Computing Roots of Polynomials over Function Fields of Curves

Shuhong Gao[1] and M. Amin Shokrollahi[2]

[1] Department of Mathematical Sciences, Clemson University, Clemson, SC 29634 USA
[2] Bell Labs, Rm. 2C-353, 700 Mountain Ave, Murray Hill, NJ 07974, USA

Abstract We design algorithms for finding roots of polynomials over function fields of curves. Such algorithms are useful for list decoding of Reed-Solomon and algebraic-geometric codes. In the first half of the paper we will focus on bivariate polynomials, i.e., polynomials over the coordinate ring of the affine line. In the second half we will design algorithms for computing roots of polynomials over the function field of a nonsingular absolutely irreducible plane algebraic curve. Several examples are included.

1 Introduction

In this paper we will study the following problem: given a nonsingular absolutely irreducible plane curve $\mathcal{X}$ over the finite field $\mathbb{F}_q$, a divisor G on $\mathcal{X}$, and a polynomial H defined over the function field of $\mathcal{X}$, compute all zeros of H that belong to $L(G)$. Our interest in this problem stems mainly from recent list decoding algorithms [5,9,11] for Reed-Solomon and algebraic geometric codes. Originally, those algorithms found the roots of H by completely factoring it and looking for factors of degree one. This method is, however, not very efficient, especially if $\mathcal{X}$ is not a rational curve.

We will design more efficient algorithms by utilizing the fact that we are interested in *roots* of H rather than a complete factorization. For instance, suppose that $\mathcal{X}$ is the projective line, $L(G)$ is the space of univariate polynomials of degree $\leq k$ over $\mathbb{F}_q$, and $H(x, y)$ is a bivariate polynomial over $\mathbb{F}_q$. The problem is then that of finding polynomials f of degree $\leq k$ in the variable x such that $H(x, f) = 0$. For this problem we will design an algorithm that runs in time $O(k^2 b^3)$ where b is the degree of H in the variable y.

In the next section we review some well-known facts on the running times for certain operations on polynomials over finite fields and introduce an algorithm for computing roots of bivariate polynomials and demonstrate its use by means of several examples. In Section 3 we will attack the more general problem stated at the beginning of the introduction.

2 Roots of Polynomials over Rational Function Fields

In this paper we will mainly deal with probabilistic algorithms. The measure of an algorithm, usually called "time," will be the (expected) number of

operations in $\mathbb{F}_q$, and usually we will use the "Soft O" notation to ignore logarithmic factors: $g = \tilde{\mathcal{O}}(n)$ means that $g = O(n \log^c n)$ for some constant c. The term "deterministic time" of an algorithm is meant to imply that the algorithm in question is deterministic.

We briefly recall some well-known results. Two polynomials of degree $< n$ over $\mathbb{F}_q$ can be multiplied in deterministic time $\tilde{\mathcal{O}}(n)$ [2, Th. 2.13]. The same is true for computing the division with remainder [2, Cor. 2.26], and the gcd of two such polynomials [2, Th. 3.13]. In particular, arithmetic operations in a given extension field $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$ can be done in deterministic time $\tilde{\mathcal{O}}(d)$. Furthermore, given two polynomials f and g, both of degree $< n$, and an integer ℓ, one can compute $f^\ell \bmod g$ in deterministic time $\tilde{\mathcal{O}}(n \log \ell)$ using the "binary method" [2, pp. 3–4]. The roots of a polynomial of degree $< n$ over $\mathbb{F}_q$ can be computed in time $\tilde{\mathcal{O}}(n \log q)$ [1, Theorem 5]. Without using fast algorithms, the running time for this task is $O(n^2 \log n \log q)$. Moreover, for any given d one can find an irreducible polynomial of degree d over $\mathbb{F}_q$ and hence construct the field $\mathbb{F}_{q^d}$ via an algorithm that runs in time $\tilde{\mathcal{O}}(n^2 \log q)$ [1, Theorem 3].

In this section we present an algorithm which solves the following problem: given a polynomial $H(x, y)$ in two variables of degree m in x and degree b in y over a finite field $\mathbb{F}_q$ and a positive integer k, find all polynomials f in x of degree at most k such that $H(x, f(x)) \equiv 0 \bmod x^{k+1}$. For simplifying assertions on the running time we will assume the following.

Assumption 1. *H is a bivariate polynomial whose degree b in y satisfies $b \leq k$. We further assume that $\log q \leq k$, and that H is not divisible by x.*

Our algorithm is a modification of Kaltofen's [6] and is based on the following simple idea: let $H = \sum_{i=0}^{m} H_i(y) x^i$. We are looking for $f_0, \ldots, f_k \in \mathbb{F}_q$ and $\psi_0, \ldots, \psi_k \in \mathbb{F}_q[y]$ such that

$$\begin{aligned}(y - f_0 - f_1 x - \cdots - f_k x^k)(\psi_0 + \psi_1 x + \cdots + \psi_k x^k) = \\ H_0 + H_1 x + \cdots + H_k x^k \bmod x^{k+1}.\end{aligned} \tag{1}$$

f_0 is found by factoring H_0 over $\mathbb{F}_q$. If H_0 is squarefree, then multiplying out and comparing the "coefficients" of x^i for $i = 0, \ldots, k$ will successively reveal $f_1, \ldots, f_k$.

Algorithm 2. *On input a bivariate polynomial $H = \sum_{i=0}^{m} H_i(y) x^i$ over the field $\mathbb{F}_q$ such that $H_0(y)$ is squarefree, and an integer $k \geq 1$, the algorithm outputs a list $f^{(1)}, \ldots, f^{(s)}$ of polynomials of degree $\leq k$ such that $H(x, f^{(j)}(x)) \equiv 0 \bmod x^{k+1}$ for $j = 1, \ldots, s$.*

(1) *Find all roots of H_0 in $\mathbb{F}_q$. Call them $\beta_1, \ldots, \beta_s$. If $s = 0$, then terminate the algorithm and output the empty set.*
(2) *For $\ell = 1, \ldots, s$ do*
   (a) *Set $\beta := \beta_\ell$.*
   (b) *For $i = 0, \ldots, k$ compute $h_i := H_i(\beta)$.*

(c) *Set* $f_0 := \beta$, $\varphi_0 := (y-\beta)$, $\psi_0 := H_0/(y-\beta)$, $\eta_0 := H_0'(\beta)$.
(d) *For* $i = 1, \ldots, k$ *compute*

$$\begin{aligned}
\varphi_i &:= \frac{h_i - \varphi_1\eta_{i-1} - \cdots - \varphi_{i-1}\eta_1}{\eta_0}, \\
\psi_i &:= \frac{H_i - \varphi_i\psi_0 - \cdots - \varphi_1\psi_{i-1}}{\varphi_0}, \\
\eta_i &:= \psi_i(\beta), \\
f_i &:= -\varphi_i.
\end{aligned}$$

Theorem 3. *The above algorithm computes its output in time* $O(k^2b^2)$.

Proof. Let us first prove correctness. Fix ℓ. We will show by induction on i that

$$\left(\sum_{j=0}^{i}\varphi_j x^j\right)\left(\sum_{j=0}^{i}\psi_j x^j\right) \equiv H(x,y) \bmod x^{i+1}.$$

The assertion is true for $i = 0$: $\varphi_0\psi_0 = H_0$. Suppose now that the assertion is true for $i-1$. We only need to show that $\varphi_0\psi_i + \cdots + \varphi_i\psi_0 = H_i$. But, since $\eta_0 \neq 0$ by the assumption of squarefreeness of H_0, this is equivalent to

$$\begin{aligned}
\varphi_i &= \frac{H_i(\beta) - \varphi_1\eta_{i-1}(\beta) - \cdots - \varphi_{i-1}\eta_1(\beta)}{\eta_0(\beta)} \\
\psi_i &= \frac{H_i - \varphi_i\psi_0 - \cdots - \varphi_1\psi_{i-1}}{\varphi_0}
\end{aligned}$$

which is exactly what is computed in the most inner loop of the algorithm. Stated in terms of f, this result shows that

$$(y - f_0 - f_1x - \cdots - f_ix^i)(\psi_0 + \psi_1x + \cdots + \psi_ix^i) \equiv H(x,y) \bmod x^{i+1}.$$

Hence, $H(x,f) \equiv 0 \bmod x^{i+1}$.

For assessing the running time, note first that computing the β_i uses $O(b^2 \log b \log q)$ operations. Computation of the h_i takes at most kb operations using Horner's rule. In the inner loop (d) computing φ_i uses $O(i)$ operations, computing ψ_i uses $O(bi)$ operations (note that each H_i and each ψ_j is of degree at most b) and computing η_i uses another $O(b)$ operations. Hence, steps (a) to (d) use $O(k^2b)$ operations, which shows that the cost of Step (2) is $O(k^2b^2)$. Since $\log q \leq k$ by our general assumption, the result follows. □

Remark 4. Even if H_0 is not squarefree, the above algorithm works for a particular root β of H_0 as long as β is a simple root. In that case, the algorithm finds a solution f of $H(x,f) \equiv 0 \bmod x^{k+1}$ with $f(0) = \beta$ in time $O(bk^2)$.

We proceed with an example. Let

$$\begin{aligned} H(x,y) &= x^7 + y^3x^5 + y^3x^4 + (y^4 + y^2 + y + 1)x^3 + (y^3 + y^2 + 1)x^2 + \\ &\quad (y^2 + y)x + y^5 + y^4 + y^3 + y \\ &=: H_7x^7 + H_6x^6 + H_5x^5 + H_2x^2 + H_1x + H_0 \end{aligned}$$

over the base field $\mathbb{F}_2$. As $H_0(y)$ is squarefree, we can apply the foregoing algorithm. We set $k := 3$, i.e., we are looking for those polynomials $f \in \mathbb{F}_2[x]$ such that $H(x, f) \equiv 0 \bmod x^4$. One root of $H_0(y)$ over $\mathbb{F}_2$ is $\beta := 0$. Applying our algorithm we then obtain

$$\begin{array}{llll} \varphi_0 = y, & \psi_0 = y^4 + y^3 + y^2 + 1, & \eta_0 = 1, & f_0 = 0, \\ \varphi_1 = 0, & \psi_1 = y + 1, & \eta_1 = 1, & f_1 = 0, \\ \varphi_2 = 1, & \psi_2 = y^3, & \eta_2 = 0, & f_2 = 1, \\ \varphi_3 = 0, & & & f_3 = 0. \end{array}$$

Hence, this setting of β yields the polynomial $f = x^2$. Another root of H_0 is 1. For this root we obtain

$$\begin{array}{llll} \varphi_0 = y + 1, & \psi_0 = y^4 + y^2 + y, & \eta_0 = 1, & f_0 = 1, \\ \varphi_1 = 0, & \psi_1 = y, & \eta_1 = 1, & f_1 = 0, \\ \varphi_2 = 1, & \psi_2 = y^3 + 1, & \eta_2 = 0, & f_2 = 1, \\ \varphi_3 = 1, & & & f_3 = 1, \end{array}$$

which yields $f = x^3 + x^2 + 1$. In both these cases we have in fact $H(x, f) = 0$. Polynomial division results in the factorization

$$H(x,y) = (y + x^2)\left(y + (x^3 + x^2 + 1)\right)\left(y^3 + y + (x^2 + x + 1)\right).$$

Let us now consider the case when H_0 is not squarefree. We will use the method of Newton polygons. Since we seek solutions modulo an arbitrary power of x, it is natural to work in the ring $\mathbb{F}_q[[x]]$ of formal power series in x. Denote by $\mathbb{F}_q[[x]][y]$ the ring of polynomials in y with coefficients in $\mathbb{F}_q[[x]]$, and by $\mathbb{F}_q[[x, y]]$ the ring of formal power series in x and y. Note that $\mathbb{F}_q[x, y] \subset \mathbb{F}_q[[x]][y] \subset \mathbb{F}_q[[x, y]]$, and they are all unique factorization domains. For any $H \in \mathbb{F}_q[[x, y]]$, its **Newton polygon** is defined to be the lower convex hull of all points (i, j) with $c_{ij} \neq 0$ and the point $(+\infty, +\infty)$ at infinity in the real Euclidean plane. For example, the Newton polygon of $H = y^5 + (y^4 + y^3)x^2 + y^5x^4 + y^3x^6 + x^8 + x^9$ is shown in Figure 1. A Newton polygon consists of (finite) line segments with nonzero slopes, called edges of the polynomial. For the above example, H has two edges with slopes 1 and 1/2 respectively. The definition of Newton polygon given here is equivalent to that of Cassels [3, Sections 6.3 and 6.4] (note that $\mathbb{F}_q[[x]]$ is a complete local ring in which x is a prime). Denote each edge by a pair (ρ, ℓ) where ρ is its slope and ℓ its length on the x-axis, and denote a Newton polygon by a list of pairs $[(\rho_1, \ell_1), \ldots, (\rho_t, \ell_t)]$ of all the edges. So the above Newton

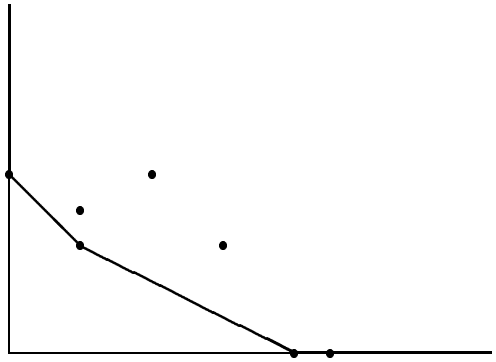

Figure1. Newton Polygon of H

polygon is denoted by $[(1,2),(1/2,6)]$. The notation also implicitly implies that we are interested in the Newton polygon only up to a translation in the real Euclidean plane.

The Newton polygon of a power series H carries a lot of information about the factors of H. The following result is from Cassels [3].

Lemma 5. *Let $H \in \mathbb{F}_q[[x]][y]$.*

(i) *If $G \in \mathbb{F}_q[[x]][y]$ divides H then the slope of every edge of G is also a slope of H.*
(ii) *Suppose that the Newton polygon of H is of the form $[(\rho_1,\ell_1),\ldots,(\rho_t,\ell_t)]$. Then there exist $G_i \in \mathbb{F}_q[[x]][y]$ with Newton polygon of the form $[(\rho_i,\ell_i)]$, $1 \le i \le t$, such that $H = G_1 \cdots G_t$.*

In particular, each edge of H corresponds to a distinct factor of H. These factors may still be reducible. To describe how they factor, we follow McCallum [7]. We need some more terminology. Let w be a rational number. For a monomial $x^i y^j$, we define its w-degree to be $i + wj$. For a power series $H \in \mathbb{F}_q[[x,y]]$, its w-order is defined to be the minimum w-degree of all nonzero terms of H, denoted by $o_w(H)$. For a polynomial $H \in \mathbb{F}_q[x,y]$, its w-degree is defined to be the maximum w-degree of all nonzero terms of H, denoted by $d_w(H)$. A polynomial is called a w-form if all of its nonzero terms have the same w-degree. Obviously, any polynomial can be written as a sum of w-forms. The initial w-form of H is the w-form in H that has the smallest w-degree. It is straightforward to see that if ρ is a slope of H and $w = 1/\rho$, then the edge with slope ρ corresponds exactly to the initial w-form of H (see below). The next result from McCallum [7, Theorem 2.2] says that the factors of the initial w-form give factors of H itself.

Lemma 6. *Let $w > 0$ be a rational number and $H \in \mathbb{F}_q[[x]][y]$. Suppose that the initial w-form h_0 of H is not divisible by x. If $h_0 = f_0 g_0$ for $f_0, g_0 \in$*

$\mathbb{F}_q[x,y]$ *and* $\gcd(f_0,g_0)=1$ *then there exist* $F,G\in\mathbb{F}_q[[x]][y]$ *such that* $H=FG$, $\deg_y F=\deg_y f_0$, *and* f_0 *(resp.* g_0*) is the initial* w*-form of* F *(resp.* G*).*

We describe below more explicitly how an initial w-form factors. Let $H\in\mathbb{F}_q[[x]][y]$. Consider a typical edge of H, say from $A=(t,h)$ to $B=(t-u,h+v)$ where $t\geq u>0$, $h\geq 0$ and $v>0$ are integers. The slope of the edge is $\rho=v/u$. Let $\ell=\gcd(u,v)$, $u_1=u/\ell$, $v_1=v/\ell$ and $w=1/\rho=u/v=u_1/v_1$. Any integral point on the edge AB must be of the form $A+(-u_1 i,v_1 i)$ for some $0\leq i\leq\ell$. All the terms of H that lie on the edge have the same w-degree $t+wh$. Any point above the egde has higher w-degree. Thus the initial w-form of H is

$$H_0=\sum_{i=0}^{\ell}c_i x^{t-u_i}y^{h+v_1 i}=x^{t-u}\cdot y^h\cdot x^u\sum_{i=0}^{\ell}c_i\left(\frac{y^{v_1}}{x^{u_1}}\right)^i=x^{t-u}\cdot y^h\cdot x^u\tilde{H}_0(z)$$

for some $c_i\in\mathbb{F}_q$, where $z=y^{v_1}/x^{u_1}$ and $\tilde{H}_0(z)=\sum_{i=0}^{\ell}c_i z^i$. Note that $x^u\tilde{H}_0(z)\in\mathbb{F}_q[[x]][y]$. In the following, we call $\tilde{H}_0$ the reduced polynomial of H_0. Note that $\tilde{H}_0$ has degree ℓ and $\tilde{H}_0(0)\neq 0$, since H must have two nonzero terms corresponding to the vertices A and B on its Newton polygon. If $\gcd(u,v)=1$ then there is no integral point on the edge AB except the end points, and so AB is the shortest line segment with slope $\rho=v/u$. By Lemma 5 (i), $x^u\tilde{H}_0$ can not factor (in $\mathbb{F}_q[[x]][y]$). In this case $x^u\tilde{H}_0$ must be (absolutely) irreducible and can be lifted to a factor of H by Lemma 5. Now suppose $\ell=\gcd(u,v)>1$. Since $\tilde{H}_0$ is a univariate polynomial, it factors into linear factors over an extension field of $\mathbb{F}_q$. Each linear factor $z-\beta$ of $\tilde{H}_0$ gives an absolutely irreducible factor $y^{v_1}-\beta x^{u_1}$ of H_0.

Lemma 7. *Let* $H\in\mathbb{F}_q[[x]][y]$ *with* $H(0,0)=0$ *and* H *not divisible by* y. *Then any factor* $y-f(x)$ *of* H, *where* $f(x)\in\mathbb{F}_q[[x]]$ *and* $f(0)=0$, *is of the form*

$$y-(\beta x^w+\textit{terms of higher degrees in } x)$$

where $w>0$ *is an integer and* $\beta\in\mathbb{F}_q$ *such that* $1/w$ *is a slope of the Newton polygon of* H *and* β *is a root of the reduced polynomial of the initial* w*-form of* H.

Proof. Suppose $f(x)=f_1x+f_2x^2+\cdots\in\mathbb{F}_q[[x]]$ and $y-f(x)$ divides H. Let $w>0$ be the smallest integer such that $f_w\neq 0$. Then the Newton polygon of $y-f(x)$ has only one edge starting at $(0,1)$ and ending at $(w,0)$ whose slope is obviously $1/w$. By Lemma 5 (i), $1/w$ is also a slope of H. Note that the initial w-forms are multiplicative, i.e., $(FG)_0=F_0G_0$ for $F,G\in\mathbb{F}_q[[x]][y]$. Let H_0 be the initial w-form of H. As $y-f_wx^w$ is the initial w-form of $y-f(x)$, we see that $y-f_wx^w$ divides H_0. By the above argument, f_w is a root of the reduced polynomial $\tilde{H}_0$ of H_0. □

Lemma 7 shows clearly how to find solutions for our problem. Let $H=H_0+H_1x+\cdots+H_mx^m\in\mathbb{F}_q[x,y]$. We want to find all solutions $f(x)\in\mathbb{F}_q[x]$

for (1). Suppose that $y = \beta$ is a root of H_0 of multiplicity $v > 1$. Make a change of variables $y_1 = y - \beta$ and $G = H(x, y_1 + \beta)$. To lift $y - \beta$, we need to find all factors of G of the form $y_1 - f_1 x - \cdots - f_k x^k$. If $y_1 \mid G$ then $y_1 = y - \beta$ is a solution. In this case, we can divide out y_1 in G and denote the resulting polynomial by G'. If $G'(0,0) \neq 0$ then G has no other factor of the form $y_1 - f_1 x - \cdots - f_k x^k$. So we may assume that G is not divisible by y_1 and $G(0,0) = 0$. Thus G is of the form of H as in Lemma 7 with respect to x and y_1. Compute the Newton polygon of G (with respect to x and y_1). We find all the egdes of G with slopes of the form $1/w$ for some integers w. For each such edge, find all the linear factors $y_1 - \beta_1 x^w$ of the initial w-form G_0 of G where $\beta_1 \in \mathbb{F}_q$ is a root of the reduced polynomial $\tilde{G}_0$ of g_0. When G has no such edges or $\tilde{G}_0$ has no roots in $\mathbb{F}_q$ then $y_1 = y - \beta$ can not be lifted to a factor $y - f(x)$ of H with $f_0 = \beta$. If β_1 is a simple root of $\tilde{G}_0$, we will show below how to lift such a partial solution. So suppose that β_1 is a multiple root. Make another change of variables $y_2 = y_1 - \beta_1 x^w$ and let $G_1 = G(x, y_2 + \beta_1 x^w)$. We can compute the Newton polygon again for G_1 and repeat the above procedure. For G_1, we need only to consider the edges of slopes $1/w_1$ with $w_1 > w$, so that higher powers of x will be added in the changes of variables. Since the w's increase at least by 1 each time and we only need powers of x up to k, this procedure will stop after at most k iterations. As is described below, all such partial solutions can be lifted to true solutions.

We illustrate this method by means of an example. Let H be as in the above example and we compute over $\mathbb{F}_2$. The first change of variables is not needed. H has two edges of slopes 1 and $1/2$ respectively. For the edge of slope 1, $w = 1$ and the initial w-form of H equals $H_0 = y^5 + y^3x^2 = y^3(y^2 + x^2) = y^3(y + x)^2$. So $\beta = 1$ is a multiple root. Let $y_1 = y - x$ and

$$G = H(x, y_1 + x) = x^9 + x^8 + y^2x^7 + (y^3 + 1)x^6 + y^4x^5 + y^5x^4 + y^2x^3 + (y^4 + y^3)x^2 + y^4x + y^5$$

The Newton polygon of G is shown in Figure 2. Note that G has two edges of slopes 1 and $2/3$ respectively, none of them is of the form $1/w_1$ with w_1 an integer $> w = 1$. Hence $y - x$ can not be lifted to a factor $y - f(x)$ of H with $f_0 = 0$ and $f_1 = 1$. Consider the edge of H with slope $1/2$. Then $w = 2$ and the intial w-form is $G_0 = y^3x^2 + x^8 = x^8(z^3 + 1) = x^8(z + 1)(z^2 + z + 1)$ where $z = y/x^2$. As $\beta = 1$ is a simple root of $z^3 + 1$, $y + x^2$ can be lifted to a factor of H. Therefore H has only one solution which is a lift of $y + x^2$.

Algorithm 8. *(Finding partial solutions) On input $H \in \mathbb{F}_q[x, y]$ and an integer $k > 1$, this algorithm compute a list L of all triples (w, β, g) where w is ∞ or an integer > 0, $\beta \in \mathbb{F}_q$, and $g \in \mathbb{F}_q[x]$ is a polynomial such that if $w = \infty$ then $H(x, g) \equiv 0 \bmod x^{k+1}$, and if $w < \infty$ then β is a simple root of the reduced initial w-form of $H(x, y + g)$.*

(0) *Initialization: $w := 0$, $g := 0$, and $L = \{\}$.*

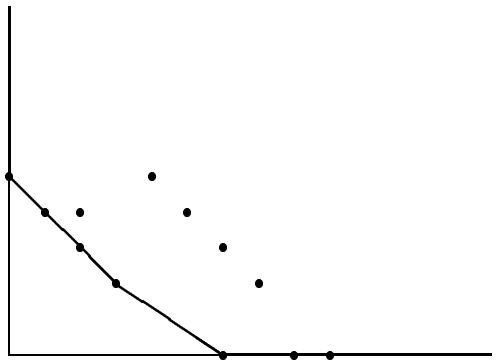

Figure2. Newton Polygon of G

(1) *Compute the initial w-form H_0 of H and write it in the reduced form $\tilde{H}_0 \in \mathbb{F}_q[z]$. Find all the roots β of $\tilde{H}_0$ in $\mathbb{F}_q$.*
(2) *For all roots β from Step 1, do the following:*
   (a) *If β is a simple root then $L := L \cup \{(w, \beta, g)\}$;*
   (b) *If β is a multiple root then compute*

$$H := H(x, y + \beta x^w) \bmod x^{k+1}, \text{ and } g := g + \beta x^w.$$

If y divides H then $L := L \cup (\infty, \beta, g)$ and set $H := H/y^a$ where a is the largest integer such that $y^a \mid H$.
   (i) *Compute the Newton polygon of H and the slopes of the edges.*
   (ii) *For each slope of the form $1/d$ with $d > w$ where d is a positive integer, set $w := d$ and go to Step (1).*
(3) *Return the list L.*

It is important to note that Step (2) of the algorithm is executed in parallel for all roots β. This means that the algorithm traverses the computation tree in a breadth-first fashion. A partial solution is built up on each path separately. One can implement the algorithm more efficiently in a depth-first fashion.

The algorithm returns two types of partial solutions (w, β, g). For $w < \infty$, we will show below how to lift g to a solution f modulo any power of x. For $w = \infty$, g is already a solution modulo x^{k+1}. In the latter case, however, g may not in general be liftable to a solution modulo a higher power of x.

Theorem 9. *Algorithm 8 correctly returns all partial solutions with $O(b^3k^2)$ operations.*

Proof. The correctness follows from the discussion above. On the running time, the dominant cost is at Step (2b) for updating H and computing Newton polygons. Since H has at most bk nonzero terms, $H(x, y+\beta x^w) \bmod x^{k+1}$

can be computed by Horner's rule (on y) in time $O(b^2k)$, and the Newton polygon of H can also be computed in this time. Each β here represents a term in a potential solution. Since H has at most b solutions and each one has at most k terms, Step (2b) is executed at most bk times. So the whole algorithm runs in time as claimed. □

Remark 10. When the degree b in y of H is large, Algorithm 8 can be improved by the following strategy. By Lemma 5, each edge of H corresponds to a factor of H. Only the factors of edges with slopes of the form $1/w$ with w an integer can have factors linear in y. Hence one can factor H at each stage according to the edges of the Newton polygon. Then for each factor of an edge with slope $1/w$, make a translation of variables and repeat the same procedure to the new polynomial. Since the degree in y of the factors decreases at each stage, this modified version of Algorithm 8 will be faster.

Finally, we show how to lift the partial solutions (w, β, g), $w < \infty$, returned by Algorithm 8. McCallum [7] discusses a more general case. Since we are dealing with linear factors, the algorithm here will be much simpler and in fact it will be exactly Algorithm 2.2. Let $G = H(x, y + g)$, which is computed at Step 2.b in Algorithm 8. Write G into a sum of w-forms

$$G = G_u + G_{u+1} + \cdots + G_{u+v}$$

where $u = o_w(G)$, $u + v = d_w(G)$, and G_i is either zero or a w-form of w-degree i for $u \le i \le u + v$. So G_u is the initial w-form of G. Note that G_i is of the form

$$G_i = x^i \sum_{j=0}^{\ell} c_j \left(\frac{y}{x^w}\right)^j$$

for some integer ℓ, where $c_j \in \mathbb{F}_q$. Let

$$\tilde{G}_i = \sum_{j=0}^{\ell} c_j z^j \in \mathbb{F}_q[z]$$

where $z = y/x^w$. Then

$$G = x^u(\tilde{G}_u + \tilde{G}_{u+1}x + \cdots + \tilde{G}_{u+v}x^v).$$

Note that $\tilde{G}_u$ is equal to the reduced polynomial of G_u up to a factor of a power of y. By the design of Algorithm 8, β is a simple root of $\tilde{G}_u$ and $y - \beta x^w$ is a factor of G_u. We want to find $f_0 = \beta, f_1, \ldots, f_{k-w} \in \mathbb{F}_q$ such that

$$(y - f_0x^w - f_1x^{w+1} - \cdots - f_{k-w}x^k)\psi \equiv G \bmod x^{k+1} \tag{2}$$

for some $\psi \in \mathbb{F}_q[x, y]$. Since $o_w(G) = u$ and $o_w(y - f(x)) = w$, we have $o_w(\psi) = u - w$. If we write ψ into a sum of w-forms and use the reduced form as we did for G, then we have

$$\psi = x^{u-w}(\tilde{\psi}_0 + \tilde{\psi}_1x + \cdots + \tilde{\psi}_{k-w}x^{k-w})$$

where $\tilde{\psi}_i \in \mathbb{F}_q[z]$ of appropriate degrees. Now divide the equation (2) on both sides by x^u, we have

$$(z - f_0 - f_1 x - \cdots - f_{k-w}x^{k-w})(\tilde{\psi}_0 + \tilde{\psi}_1 x + \cdots + \tilde{\psi}_{k-w}x^{k-w}) \equiv$$

$$\tilde{G}_u + \tilde{G}_{u+1}x + \cdots + \tilde{G}_{u+v}x^v \bmod x^{k+1-u}.$$

This is exactly the type of the equation (1) we started with. Since β is a simple root of $\tilde{G}_u$, Algorithm 2.2 can be applied to find a solution $f(x) = f_0 + f_1 x + \cdots + f_{k-w}x^{k-w} \in \mathbb{F}_q[x]$ with $f_0 = \beta$ for the above equation. Then $g + x^w f(x)$ is a solution for the equation (1).

Theorem 11. *Algorithms 2 and 8 find all solutions of the Equation (1) in time $O(b^3k^2)$.*

Proof. Since Algorithm 2 lifts a partial solution in time $O(bk^2)$ and there are at most b solutions, all the solutions of (1) can be found in time $O(b^3k^2)$. □

Example. Consider the polynomial

$$H = x^6 + (y+1)x^5 + x^4 + x^3 + (y^3 + y)x^2 + y^2x + (y^4 + y^3)$$

over $\mathbb{F}_2$. $H_0 = y^3 + y^4 = y^3(y+1)$ has a simple root $y = 1$ and a triple root $y = 0$. The first one can be lifted to a true solution by Algorithm 2.2. To lift the second one, we need to find the Newton polygon of H, which happens to have only one edge of slope 1. So let $w = 1$. The initial w-form of H is

$$h_0 = y^3 + y^2x + yx^2 + x^3 = x^3(\frac{y}{x} + 1)^3.$$

Thus $\beta = 1$ is a triple root of $\tilde{h}_0$. Make a translation of variables $y_1 = y - x$. Then

$$\begin{aligned} G = H(x, y_1 + x) &= x^5y_1 + x^2y_1^3 + y_1^2x^3 + x^4y_1 + y^3 + y_1^4 \\ &= y_1(x^5 + x^2y_1^2 + y_1x^3 + x^4 + y_1^2 + y_1^3). \end{aligned}$$

Hence $y_1 = y + x$ is a solution of H (modulo any power of x). Let $G_1 = G/y_1$. Its Newton polygon has one edge of slope $1/2$. Let $w = 2$. Then the initial w-form of G_1 is

$$g_0 = y_1^2 + x^4 = x^4(y_1/x^2 + 1)^2.$$

So $\beta = 1$ is a double root of $\tilde{g}_0$. Make another translation of variables $y_2 = y_1 - x^2$ and

$$G_2 = G_1(x, y_2 + x^2) = x^4y_2 + y_2x^3 + y_2^2 + y_2^3 = y_2(x^4 + x^3 + y_2 + y_2^2).$$

So $y_2 = y_1 - x^2 = y - x - x^2$ is solution (modulo any power of x). The Newton polygon of $G_3 = G_2/y_2$ has an edge of slope $1/3$. Let $w = 3$. Then the initial w-form of G_3 is $y_2 + x^3 = x^3(y^2/x^3 + 1)$ for which $\beta = 1$ is a simple root. So $y_2 - x^3 = y - x - x^2 - x^3$ can be lifted to a solution modulo any power of x. In total there are four solutions: $y - x$, $y - x - x^2$, and lifts of $y - 1$ and $y - x - x^2 - x^3$.

3 The General Case

In this section we assume familiarity with basic concepts from the theory of algebraic curves. (See, e.g., [10].) Let $\mathcal{X}$ be a nonsingular curve given as the zeroset of an absolutely irreducible polynomial $F \in \mathbb{F}_q[x,y]$ and let $R := \mathbb{F}_q[x,y]/(F)$ denote its coordinate ring. Let G be a divisor on $\mathcal{X}$ defined over $\mathbb{F}_q$ and let $L(G)$ denote the linear space of G. Assume that we are given a basis $\varphi_1, \ldots, \varphi_\ell$ of $L(G)$ such that each $\varphi_i \in R$. We are interested in computing the roots in $L(G)$ of a polynomial

$$H(T) = u_b T^b + \cdots + u_1 T + u_0$$

with coefficients $u_i \in R$. The algorithm we will present below is a generalization and simplification of that stated in [9] for polynomials of degree $b = 2$.

Assumption 12. *For the rest of this section we will assume that* $\deg F =: D \geq 3$, *that* $k := \deg G \geq 2D^2$, *that the basis functions* φ_i *of* $L(G)$ *are represented modulo* F *as bivariate polynomials of degree* $\leq B$, *and that the functions* u_i *are represented modulo* F *as bivariate polynomials of degree* $\leq C$. *Furthermore, we assume that* $b, \log q \leq k$.

The first step of the algorithm to be presented below consists of finding an $\mathbb{F}_{q^d}$-rational solution $\mathfrak{p} = (a,b)$ of $F(x,y) = 0$ where $d > k$ and where either a or b is a primitive element of the extension $\mathbb{F}_{q^d}/\mathbb{F}_q$. We call $\mathfrak{p}$ an (affine) point of $\mathcal{X}$ (or of F) of degree d over $\mathbb{F}_q$.

Algorithm 13. *On input an irreducible nonsingular bivariate polynomial* $F(x,y)$ *over* $\mathbb{F}_q$ *of degree* D *and an integer* $d \geq 2D^2$ *the algorithm computes an affine point* $\mathfrak{p}$ *of the zeroset of* F *of degree* d *over* $\mathbb{F}_q$.

(1) *Construct the field* $\mathbb{F}_{q^d}$.
(2) *Randomly select an element* ζ *of* $\mathbb{F}_{q^d}$ *until a primitive element of* $\mathbb{F}_{q^d}/\mathbb{F}_q$ *is found.*
(3) *Test whether* $g(\zeta, y)$ *has a zero* y_0 *in* $\mathbb{F}_{q^d}$. *If yes, then output* $\mathfrak{p} = (\zeta, y_0)$. *If not, then test whether* $g(x,\zeta)$ *has a zero* x_0 *in* $\mathbb{F}_{q^d}$. *If yes, then output* $\mathfrak{p} = (x_0, \zeta)$. *If not, then go back to Step* (2).

Theorem 14. *The above algorithm correctly computes its output in time*

$$\tilde{\mathcal{O}}(d^2 D \log q + d^3).$$

Proof. Let N_i denote the number of solutions in $\mathbb{F}_{q^i}^2$ of $F(x,y) = 0$. We first prove that

$$|N_i - q^i| \leq D^2 q^{i/2}. \tag{3}$$

Since F is nonsingular, the genus g of the zeroset of F is $(D-1)(D-2)/2$ [4, Chap. 8, Prop.5]. The number $\tilde{N}_i$ of $\mathbb{F}_{q^i}$-rational points of the zeroset $\mathcal{X}$ of F

in the projective plane over $\mathbb{F}_q$ satisfies $|\tilde{N}_i - q^i - 1| \leq 2gq^{i/2}$ by the Hasse-Weil inequality. Let $\tilde{F}(X,Y,Z)$ be the homogenized version of F. The number of $\mathbb{F}_{q^i}$-rational points of $\mathcal{X}$ in the projective plane over $\mathbb{F}_{q^i}$ which have $Z = 0$ is obviously upper bounded by $2D$. As a result we have $\tilde{N}_i \leq N_i + 2D$, which gives us

$$q^i + 1 - (D-1)(D-2)q^{i/2} - 2D \leq N_i \leq q^i + 1 + (D-1)(D-2)q^{i/2} \leq q^i + D^2 q^{i/2}.$$

It remains to show that $q^i + 1 - (D-1)(D-2)q^{i/2} - 2D \geq q^i - D^2 q^{i/2}$. A simple manipulation leads to the equivalent condition $q^{i/2} \geq (2D-1)/(3D-1)$ which is trivially true, as $D \geq 3$ by Assumption 12.

Next, we compute a lower bound for the number N of those solutions (a,b) of $F(a,b) = 0$ such that a or b is primitive. Obviously, $N = N_d - \sum_{\ell | d, \ell < d} N_\ell \geq N_d - \sum_{\ell=1}^{\lfloor d/2 \rfloor} N_\ell$, since an element of $\mathbb{F}_{q^d}$ is primitive iff it does not belong to any proper subfield of $\mathbb{F}_{q^d}$. Use of (3) yields

$$\begin{aligned} N &\geq q^d - D^2 q^{d/2} - q\frac{q^{d/2}-1}{q-1} - \sqrt{q}D^2\frac{q^{d/4}-1}{\sqrt{q}-1} \\ &\geq q^d - q^{d/2}(D^2 + q + \sqrt{q}D^2 q^{-d/4}). \end{aligned}$$

The number of primitive elements of $\mathbb{F}_{q^d}$ is $q^d - \sum_{\ell | d, \ell < d} q^\ell \geq q^d - \sum_{\ell=1}^{\lfloor d/2 \rfloor} q^\ell \geq q^d - q^{d/2+1}$. As a result, a random element in $\mathbb{F}_{q^d}$ is primitive over $\mathbb{F}_q$ with at most a constant probability. This shows that Step (2) is performed, on average a constant number of times. After this step we will have obtained a uniform randomly chosen primitive element of $\mathbb{F}_{q^d}$. The probability p that a random primitive element of $\mathbb{F}_{q^d}$ is either the x- or the y-coordinate of a solution of $F(x,y) = 0$ satisfies

$$p \geq \frac{1 - q^{-d/2}(D^2 + q + \sqrt{q}D^2 q^{-d/4})}{1 - q^{-d/2+1}} \geq \frac{1 - q^{-d/2}(2D^2 + q)}{1 - q^{-d/2+1}}.$$

(Note that $q^{-d/4} < q^{-1/2}$.) Now observe that $2D^2 \leq d \leq q^{d/4}$ and that $(2D^2 + q) \leq 2q^{d/4}$. This implies

$$p \geq \frac{1 - 2q^{-d/4}}{1 - q^{-d/2+1}}.$$

Hence, Step (3) is performed on average a constant number of times.

Let us now focus on the running time. Step (1) uses $\tilde{\mathcal{O}}(d^2 \log q)$ operations. Testing primitivity of an element ζ is done by computing $1, \zeta, \ldots, \zeta^{d-1}$ in the polynomial basis ($\tilde{\mathcal{O}}(d^2)$ operations), and testing linear independence of these elements as vectors ($O(d^3)$ operations). So, Step (2) uses $O(d^3)$ operations. Each iteration of Step (3) consists of computing $F(\zeta, y)$ (or $F(x, \zeta)$) which uses $O(D^2)$ operations over $\mathbb{F}_{q^d}$, i.e., $\tilde{\mathcal{O}}(dD^2)$ operations over $\mathbb{F}_q$, and of computing the roots of a univariate polynomial of degree $\leq D$ over $\mathbb{F}_{q^d}$ which takes $\tilde{\mathcal{O}}(D \log q^d)$ operations over $\mathbb{F}_q$, i.e., $\tilde{\mathcal{O}}(d^2 D \log q)$ $\mathbb{F}_q$-operations. □

Remark 15. The assumption $d \geq 2D^2$ in Algorithm 13 is related to applications in coding theory where one assumes that $k = \deg G > 2g-2$, where g is the genus of the curve. It can be weakened at the expense of a more tedious analysis. However, we remark that points of degree d may not exist for all values of d. For instance, the Hermitian curve $x^3 = y^2 + y$ does not have any points of degree 2 over $\mathbb{F}_4$.

The final algorithm now follows.

Algorithm 16. *Given an irreducible algebraic nonsingular bivariate polynomial F, a divisor G of the zeroset of F defined over $\mathbb{F}_q$, basis functions $\varphi_1, \ldots, \varphi_\ell$ of $L(G)$ and functions $u_0, \ldots, u_b \in \mathbb{F}_q[x,y]/(F)$ satisfying the conditions in Notation 12, the algorithm computes a list $f^{(1)}, \ldots, f^{(s)}$ of at most b functions in $L(G)$ which includes any $f \in L(G)$ such that $H(f) = 0$, where $H = \sum_{i=0} u_i T^i$.*

(1) *Using Algorithm 13 compute an affine point $\mathfrak{p}$ of degree $d = k+1$ over $\mathbb{F}_q$ of the zeroset of F.*
(2) *Compute $\varphi_1(\mathfrak{p}), \ldots, \varphi_\ell(\mathfrak{p})$ and represent them as d-dimensional vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ over $\mathbb{F}_q$.*
(3) *Compute the values $u_0(\mathfrak{p}), \ldots, u_b(\mathfrak{p})$.*
(4) *Compute the zeros $\beta_1, \ldots, \beta_s$ of the polynomial $u_0(\mathfrak{p}) + u_1(\mathfrak{p})x + \cdots + u_b(\mathfrak{p})x^b$ in the field $\mathbb{F}_{q^d}$ and represent them as d-dimensional vectors $\mathbf{b}_1, \ldots, \mathbf{b}_s$ over $\mathbb{F}_q$.*
(5) *Compute vectors $\mathbf{h}_i = (h_{1,i}, \ldots, h_{\ell,i})^\perp \in \mathbb{F}_q^\ell$, $i = 1, \ldots, s$ such that*

$$(\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_\ell)\,(\mathbf{h}_1 \mid \cdots \mid \mathbf{h}_s) = (\mathbf{b}_1 \mid \cdots \mid \mathbf{b}_s)$$

over $\mathbb{F}_q$ and output $f^{(i)} = h_{1,i}\varphi_1 + \cdots + h_{\ell,i}\varphi_\ell$.

Theorem 17. *The above algorithm correctly computes its output in time*

$$\tilde{\mathcal{O}}(k^2 D \log q + k^3 + k^2 B^2 + kbC^2 + k^2 b \log q).$$

Proof. We first prove correctness. If $f \in L(G)$ is such that $H(f) = 0$, then we have that $\sum_{i=0}^b u_i(\mathfrak{p}) f(\mathfrak{p}) = 0$, i.e., $f(\mathfrak{p})$ is one of the β_i. Writing $f = \sum_i h_i \varphi_i$, we see that $h_1, \ldots, h_\ell$ satisfy the equations in Step (5). We now prove that, for each i, the solution to this system is unique. Indeed, two solutions would give rise to functions $f, g \in L(G)$ defined over $\mathbb{F}_q$ such that $(f-g)(\mathfrak{p}) = 0$. But then $(f-g)(\mathfrak{p}^\sigma) = 0$ for all the d different automorphisms σ of $\mathbb{F}_{q^d}/\mathbb{F}_q$. This shows that $f - g$ has more zeros than poles, which implies that $f = g$. We infer that f is one of the $f^{(i)}$'s.

Step (1) of the algorithm uses $\tilde{\mathcal{O}}(d^2 D \log q)$ operations in $\mathbb{F}_q$ by Theorem 14. Each φ_i is represented by a bivariate polynomial of degree $\leq B$. So, computing $\varphi_i(\mathfrak{p})$ uses, in the worst case, $O(B^2)$ operations in $\mathbb{F}_{q^d}$, i.e., $\tilde{\mathcal{O}}(dB^2) = \tilde{\mathcal{O}}(kB^2)$ operations in $\mathbb{F}_q$. Since there are ℓ of these functions and $\ell \leq k+1$, Step (2) requires $O(k^2 B^2)$ time. Similarly, computing the $u_i(\mathfrak{p})$

uses $\tilde{\mathcal{O}}(bkC^2)$ $\mathbb{F}_q$-operations. Step (4) uses $\tilde{\mathcal{O}}(b \log q^d) = \tilde{\mathcal{O}}(db \log q)$ operations in $\mathbb{F}_{q^d}$, i.e., it requires $\tilde{\mathcal{O}}(k^2 b \log q)$ $\mathbb{F}_q$-operations. The cost of Step (5) is $O(b^2 k)$: it consists of reducing a $d \times 2s$-matrix to echelon form using row operations, and $s \leq b$. □

Remark 18. (1) In applications to coding theory one usually has a fixed divisor G (corresponding to fixing the code) and one wants to compute zeros in $L(G)$ for different polynomials H. In this case one can compute the point $\mathfrak{p}$ and the evaluation of the φ_i at $\mathfrak{p}$ in advance. Neglecting the cost of this preconditioning, the running time of the algorithm would then be $\tilde{\mathcal{O}}(kbC^2 + b^2 k + k^2 b \log q)$. Assuming that b is a constant and that $C, \log q \leq k$ (both reasonable assumptions in list decoding scenarios), this gives a running time of $\tilde{\mathcal{O}}(k^3)$.

(2) If the functions u_i and φ_i are not polynomials in x and y, it is still possible (though tedious) to analyze the running time of the algorithm. The only major change in the algorithm is to ensure that the point $\mathfrak{p}$ found has the property that the functions u_i can be evaluated at it.

(3) The assumption that the curve $\mathcal{X}$ has a *nonsingular* plane model was only needed to bound the number of solutions of $F(x, y)$ over extensions of $\mathbb{F}_q$. One can also bound these numbers without this assumption [8] and can obtain similar (though a little worse) results.

As was pointed out in Remark 15, the assumption $d \geq 2D^2$ for the existence of points of degree d can be weakened. In the next example, we will find a point of degree 6 of the degree 5 Hermitian curve over $\mathbb{F}_2$ given by the equation $x^5 = y^4 + y$. Let Q be the common pole of x and y. We are interested in zeros of the polynomial

$$\begin{aligned} H(T) &= T^3 + (x + y + 1)T^2 + (x^2 + y)T + (x^2 y + x^3 + xy + x^2) \\ &=: T^3 + u_2 T^2 + u_1 T + u_0 \end{aligned}$$

among the elements of $L(5Q) = \langle 1, x, y \rangle$. The first step consists of finding a point of degree 6. We represent $\mathbb{F}_{2^6}$ as $\mathbb{F}_2(\zeta)$ with $\zeta^6 + \zeta + 1 = 0$. Applying Algorithm 13, we find $\mathfrak{p} = (\zeta^2 + \zeta, \zeta^4 + \zeta^2)$.

The next step of the algorithm is to find the zeros of the polynomial

$$T^3 + u_2(\mathfrak{p})T^2 + u_1(\mathfrak{p})T + u_0(\mathfrak{p}) = T^3 + (\zeta^4 + \zeta + 1)T^2 + \zeta^3$$

in $\mathbb{F}_{2^6}$. They turn out to be $\beta_1 = \zeta^2 + \zeta$, $\beta_2 = \zeta^2 + \zeta + 1$, and $\beta_3 = \zeta^4 + \zeta$. Now we represent elements of $\mathbb{F}_{2^6}$ with respect to the $\mathbb{F}_q$-basis $1, \zeta, \ldots, \zeta^5$ and solve the system of equations given in Step (5):

$$\begin{pmatrix} 1&0&0 \\ 0&1&0 \\ 0&1&1 \\ 0&0&0 \\ 0&0&1 \\ 0&0&0 \\ 0&0&0 \end{pmatrix} \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{pmatrix} = \begin{pmatrix} 0&1&0 \\ 1&1&0 \\ 1&1&0 \\ 0&0&0 \\ 0&0&1 \\ 0&0&0 \\ 0&0&0 \end{pmatrix}.$$

Solving this system yields the solutions $(0,1,0)^{\perp}$, $(1,1,0)^{\perp}$, and $(0,1,1)^{\perp}$ for $(h_1,h_2,h_3)^{\perp}$ which leads to the functions $f^{(1)} = x$, $f^{(2)} = x+1$, and $f^{(3)} = x+y$.

References

1. M. Ben-Or. Probabilistic algorithms in finite fields. In *Proceedings of the 22nd IEEE Symposium on Foundations of Computer Science*, pages 394–398, 1981.
2. P. Bürgisser, M. Clausen, and M.A. Shokrollahi. **Algebraic Complexity Theory**, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer Verlag, Heidelberg, 1996.
3. J.W.S. Cassels. **Local Fields**. Number 3 in London Mathematical Society Student Texts. Cambridge University Press, London, 1986.
4. W. Fulton. **Algebraic Curves**. Addison-Wesley, 1989.
5. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 28–37, 1998.
6. E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.*, 14:469–489, 1985.
7. S. McCallum. On testing a bivariate polynomial on analytic reducibility. *J. Symb. Comp.*, 24:509–535, 1997.
8. W. M. Schmidt. **Equations over Finite Fields: An Elementary Approach**. Number 536 in Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1976.
9. M.A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 45:432–437, 1999.
10. H. Stichtenoth. **Algebraic Function Fields and Codes**. Universitext. Springer Verlag, 1993.
11. M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *J. Compl.*, 13:180–193, 1997.