Syndrome decoding for Hermite codes with a Groebner-basis algorithm [*]

Irene I. Bouw, Sabine Kampf

July 4, 2011

Abstract

In this paper we give a new approach to syndrome decoding for AG-codes. Our algorithm is as concrete and elementary as possible. For this reason, we restrict ourselves to Hermite codes, and model our approach as much as possible on the well-known case of decoding with the euclidean algorithm for Reed–Solomon codes. A new feature of our approach is that we may decode beyond the minimal distance.

Keywords: Hermite codes, algebraic decoding, Groebner bases.
Mathematics Subject Classification (2000): Primary 14G50; Secondary 11T71.

Introduction

Algebraic-geometry codes (AG-codes) are a generalization of Reed–Solomon codes, and have many nice properties ([4],[16], [17]). For example, the family of AG-codes includes a rich variety of codes which are better than the Varshamov–Gilbert bound ([18]). Several approaches to decoding of AG-codes can be found in the literature (for example [6], [13], [15], [5]). However, up to now AG-codes are not used in practical applications. One of the reasons appears to be that understanding the proposed algorithms requires quite some background in algebraic geometry.

The present paper has two goals. On the one hand, we present a new decoding algorithm for AG-codes. Our algorithm is an extension of the algorithm by Porter et.al. [11] (see also [14], [3], [2]) and relies on a so-called key equation. We write the key equation in the same form as [11], which is much more explicit that the form it takes in [3]. However, we show that it is not necessary to exclude one place, as is needed in that paper. Our algorithm is as fast as any known general decoding algorithm, but also works up to a certain extent for decoding beyond the minimal distance.

[*]This work was supported by the German Research Council "Deutsche Forschungsgemeinschaft" (DFG) under Grant No. Bo867/22.

On the other hand, we try to present our results as elementary as possible, formulating results from algebraic geometry as much as possible as a black box, and translating the main results into easy to use and to implement statements. Moreover, our results are much more explicit than the original approaches. We have tested our results in an concrete implementation, illustrating the practicality of the proposed algorithm.

One of the main simplifications we make is that we focus on Hermite codes, rather than treat the case of general curves or general plane curves, as is usually done in the literature. Although the results extend to arbitrary curves, this makes the approach much more concrete and accessible, requiring only limited knowledge of algebraic geometry (the statement of the Riemann–Roch Theorem and a working knowledge of local rings and valuations.) We avoid for example differential forms, and state all results in terms of polynomials. Modelling the approach on the well-known case of Reed–Solomon codes, we highlight the differences and similarities between both cases.

Another reason for restricting ourselves to Hermite curves is that from a practical point of view the best AG-codes are the ones coming from maximal curves. The Hermite curves in particular and other maximal curves more generally are very suitable for coding theory, as both the rational points and a basis of the Riemann–Roch spaces are explicitly known. Moreover, these curves have a large automorphism group. This simplifies many formulas considerably. Hermite codes are therefore most likely to be among the first AG-codes that become relevant for pracitical applications.

We now describe our algorithm. Let $\mathcal{X}_q$ be the Hermite curve over $\mathbb{F}_{q^2}$ given by the affine equation $x^{q+1} = y^q + y$. It is known that $\mathcal{X}_q(\mathbb{F}_{q^2})$ has cardinality $q^3 + 1$. We denote by P the unique point at infinity, and by $\{P_1, \ldots, P_n\} := \mathcal{X}_q(\mathbb{F}_{q^2}) \setminus \{P\}$. For $2g(\mathcal{X}_q) - 1 \leq m \leq n = q^3$, we denote by $\mathcal{C}$ the image of

$$\varphi : L(mP) \to \mathbb{F}_p^m, \quad f \mapsto (f(x_i))_i,$$

where $L(mP)$ denotes the Riemann–Roch space considered as $\mathbb{F}_{q^2}$-vector space. We define the degree of elements of $L(mP)$ by minus the valuation at P (Section 1).

Given a received word $\boldsymbol{r}$, the first step of the decoding algorithm computes a minimal error locator polynomial $\Lambda \in \cup_m L(mP)$, whose zeros include the set of error locations. Here minimality is considered with respect to the degree. We show that Λ is a solution to a (modified) key equation

$$\tilde{S} \cdot \Lambda \equiv R \pmod{y^{b_m+1}},$$

where the degree of R is bounded in terms of the degree of Λ (see Section 4 for a precise statement). Here $\tilde{S}$ is the syndrome polynomial, which can be explicitly computed from $\boldsymbol{r}$ (Section 4). The modified key equation is an approximation to a conceptually easier key equation, which however uses some unknown syndromes, that cannot be computed directly from $\boldsymbol{r}$. This easier version is described in Section 3.

We a give an explicit, recursive algorithm for computing Λ and R, which is similar in spirit to the euclidean algorithm, which is used for this task in the Reed–Solomon case. Since the ring $\mathcal{R}$ of functions we work in here is not euclidean, we need to use a Gröbner-basis algorithm (Section 5). After we found an error-locator polynomial, we find the error-values by computing the residues of R/Λ at the error positions. Here (Λ, R) is the solution of the key equation we computed in the previous step.

The idea behind this statement is easiest to explain in the Reed–Solomon case. In that situation it is well-known that rather than defining RS-codes as the evaluation of polynomials of maximum degree, one may also describe them using residues of rational functions with simple poles ([10, Section 12.3]). A similar description also exists for Hermite codes; this follows easily from Serre duality. Our approach to the key equation may be thought of as combining both descriptions.

We give a practical criterion for recognizing whether the solution we found is in fact a correct solution or not. We implemented our algorithm in MAGMA and tested it on very many cases. Our results show that the practical criterion recognizes the correct solution in almost all cases. We also present a much slower algorithm, which always finds the correct solution. In Section 7 we show how to adapt our approach to decoding beyond the minimal distance. One problem here is that the first solution of the key equation we find need not be the correct solution. In this paper, we describe an algorithm which in practice finds the correct solution in most cases. In a subsequent paper, we discuss how to extend the algorithm, so that it efficiently finds a solution in all cases.

1 Preliminaries

Let $\mathbb{F}_{q^2}$ be a finite field and $\mathcal{X}_q$ the Hermite curve over $\mathbb{F}_{q^2}$, i.e. the projective curve defined by the affine equation

$$x^{q+1} = y^q + y.$$

Recall that the genus of $\mathcal{X}_q$ is $g := q(q-1)/2$. Let P be the unique point of $\mathcal{X}_q$ in infinity. We denote by

$$\mathcal{R} = \cup_{m\geq 0} L(mP)$$

the *affine ring* of $\mathcal{X}_q$ at P. For convenience, we refer to the elements of $\mathcal{R}$ as *polynomials*. It is well-known that

$$\Phi := \{\varphi_{a,b} := x^a y^b \mid 0 \leq a \leq q, 0 \leq b\}$$

is a basis of $\mathcal{R}$.

For $f \in \mathrm{Frac}(\mathcal{R})$, we define $\rho(f) = -\mathrm{ord}_P(f)$. Alternatively, one may define ρ on monomials by $\rho(x^a y^b) = -\mathrm{ord}_P(x^a y^b) = qa + (q+1)b$ and extend ρ to $f \in \mathcal{R}$ by $f = \sum_{a,b} f_{a,b} x^a y^b \in \mathcal{R}$ by defining

$$\rho(f) = \max_{(a,b):f_{a,b}\neq 0} \rho(x^a y^b),$$

and $\rho(f/g) = \rho(f) - \rho(g)$ for $f/g \in \mathrm{Frac}(\mathcal{R})$.

In analogy to the situation for $\mathbb{P}^1$, we refer to $\rho(f)$ as the *degree* of f. The term $f_{a,b}x^ay^b$ with $\rho(x^ay^b) = \rho(f)$ is called the *leading term of* f. If the leading term of f is x^ay^b, we call f *monic*.

It is well known that for every $r \in \mathbb{N}$ there is at most one monomial $x^ay^b \in \Phi$ with $\rho(x^ay^b) = r$. Therefore we can order the monomials $\varphi_{a,b}$ according to their ρ-value. We sometimes also write $\varphi_0 = \varphi_{0,0} = 1, \varphi_1 = \varphi_{1,0} = x, \varphi_2 = \varphi_{0,1} = y, \dots$ to refer to this ordering.

We denote by $\{P_0 = (0,0), \dots, P_{n-1}\} = \mathcal{X}_q(\mathbb{F}_{q^2}) \setminus \{P\}$, and let $D := \sum_{i=0}^{n-1} P_i$. It is well known that $n = q^3$. For $2g - 1 \leq m < n$, we consider the code $\mathcal{C} := \mathcal{C}_L(D, mP)$, which is defined as the image of

$$L(mP) \to \mathbb{F}^n, \qquad f \mapsto (f(P_i))_{i=0..n-1}.$$

We denote by $k = m - g + 1$ the dimension of the code, by $d^* = n - m$ the designed minimal distance, and by $k^\perp$ the dimension of the dual code. In our situation, the dual code is isomorphic to $L(m^\perp P)$, where $m^\perp = n + 2g - 2 - m$. In particular, the check matrix of $\mathcal{C}$ is

$$H := \Big(\varphi_i(P_j)\Big)_{0 \leq i \leq m^\perp, 0 \leq j \leq n-1}.$$

2 Syndrome decoding and the key equation

Suppose that $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e} = (r_i)$ is a received vector, where $\boldsymbol{c} = (c_i)$ is a codeword and $\boldsymbol{e} = (e_i)$ the corresponding error vector. Let $t = \mathrm{wt}(e)$ be the number of errors. Denote by $I \subset \{0, 1, \dots, n-1\}$ the error positions and by $Q = \sum_{i \in I} P_i$ the error divisor. An *error-locator polynomial* Λ is a function $\Lambda \in \mathcal{R}$ such that $\mathrm{ord}_{P_i}\Lambda > 0$ for all $i \in \mathcal{I}$. An error-locator polynomial of minimal degree is called a *minimal error locator*.

Definition 2.1 For every $\varphi_i \in \Phi$, we define the *syndrome element* as

$$s_i = \sum_{j=0}^{n-1} e_j \varphi_i(P_j) = s_{a,b} \text{ if } \varphi_i = \varphi_{a,b}.$$

For $m^\perp$ as in § 1, we put $a_m = q$ and $b_m = \max\{b \mid \rho(x^ay^b) \leq m^\perp\}$. We define the *syndrome polynomial*

$$S := \sum_{0 \leq a \leq a_m, 0 \leq b \leq b_m} s_{a,b} x^{a_m - a} y^{b_m - b} \in \mathcal{R}.$$

We remark that for $\rho(x^ay^b) \leq m^\perp$, we may compute the syndrome vector $\boldsymbol{s} := (s_{a,b})_{\rho(x^ay^b) \leq m^\perp}$ from the check matrix and the received vector, since $Hr^t = He^t = \boldsymbol{s}$. Sometimes, these syndromes are called the *known syndromes* in contrast to the *unknown syndromes* $s_{a,b}$ for $\rho(x^ay^b) > m^\perp$. In § 3 we will see that these unknown syndromes are not actually used in our decoding algorithm. Furthermore, we refer to the syndromes $s_{a,b}$ with $q < a \leq 2q$ as *inferred syndromes*, because these can be calculated from the syndromes with $a \leq q$ by

$$s_{a,b} = s_{a-q-1,b+1} + s_{a-q-1,b+q}, \tag{1}$$

which is a direct consequence of the defining equation of the curve. These syndromes are also known if $\rho(x^a y^b) \leq m^\perp$ and unknown otherwise.

The following lemma gives an interpretation of the syndrome polynomial, which plays a central role in our decoding algorithm. Basically the lemma states that the syndrome polynomial S is an approximation of a rational function U defined below, from which we may easily read off the error locations, and the error values.

For every $0 \leq i < n$ we define a rational function u_i, as follows. We write $P_i = (\alpha_i, \beta_i)$ and define

$$u_i = \frac{y^q + y - \alpha_i^{q+1}}{(x-\alpha_i)(y-\beta_i)} = \frac{1 + \sum_{j=0}^{q-1}(y^j \beta_i^{q-1-j})}{x-\alpha_i}. \tag{2}$$

Note that u_i has a simple pole in P_i, a pole in P with $\rho(u_i) = q^2 - q - 1$, and no other poles. Moreover, the polar part of u_i at P_i is $1/(x-\alpha_i)$. Put

$$U := -\sum_{i \in I} e_i \beta_i^{b_m+1} u_i. \tag{3}$$

Lemma 2.2 *The syndrome polynomial may also be written as*

$$S(x,y) = \sum_{i \in I} e_i (y^{b_m+1} - \beta_i^{b_m+1}) u_i.$$

Proof: Using the definition of the syndromes and (2), we can rewrite S as

$$\begin{aligned} S = & \sum_{0 \leq a \leq q, 0 \leq b \leq b_m} \Big(\sum_{i \in I} e_i \alpha_i^{a_m - a} \beta_i^{b_m - b}\Big) x^a y^b = \sum_{i \in I} e_i \Big(\sum_{u=0}^{b_m} \beta_i^{b_m - u} y^u\Big)\Big(\sum_{v=0}^{q} \alpha_i^{q-v} x^v\Big) \\ & = \sum_{i \in I} e_i \frac{(y^{b_m+1} - \beta_i^{b_m+1})(y^q + y - \alpha_i^{q+1})}{(x-\alpha_i)(y-\beta_i)} = \sum_{i \in I} e_i (y^{b_m+1} - \beta_i^{b_m+1}) u_i. \end{aligned}$$

□

Recall that $P_0 = (0,0)$. The reason why this point is special is that $\varphi_{a,b}$ for $(a,b) \neq (0,0)$ vanishes in this point. The following proposition uses an idea of [6]. The equation (4) is called the *key equation* in analogy to usual terminology for Reed–Solomon codes (see for example [10]).

Proposition 2.3 *Let $\Lambda \in \mathcal{R}$ be an error-locator polynomial. Then there exists a polynomial $R \in \mathcal{R}$ such that*

$$\begin{cases} \operatorname{ord}_{P_0}(\Lambda \cdot S - R) & \geq \operatorname{ord}_{P_0}(y^{b_m+1}), \\ \rho(R) - \rho(\Lambda) & \leq q a_m + (q+1) b_m - m^\perp - 1 =: \ell. \end{cases} \tag{4}$$

Proof: Let $\Lambda = \sum_{a,b} \lambda_{a,b}\varphi_{a,b}$ be an error-locator polynomial, and let $\mu := \rho(\Lambda)$. We write $\Lambda \cdot S = \sum_{a,b} t_{a,b}\varphi_{a,b}$.

We define

$$R := \sum_{qa+(q+1)b \leq \ell+\mu} t_{a,b}x^a y^b.$$

Then obviously $\rho(R) \leq \ell + \mu$. Consider the coefficient $t_{a,b}$ for

$$\mu + \ell < \rho(x^a y^b) < \rho(y^{b_m+1}),$$

i.e.

$$t_{a,b} = \sum \lambda_{i,j} s_{a_m-a+i,b_m-b+j}.$$

We remark that for all (i,j) with $\lambda_{i,j} \neq 0$ we have that $\rho(x^i y^j) \leq \rho(\Lambda) = \mu$. This implies that for all such (i,j), we have that

$$q(a_m - a + i) + (q+1)(b_m - b + j) < \mu + q^2 + (q+1)b_m - \ell - \mu = m^\perp + 1.$$

Therefore in this situation

$$t_{a,b} = \sum_{u=0}^{n-1} e_u \varphi_{a_m-a,b_m-b}(P_u)\Lambda(P_u) = 0,$$

since for every $u \in \{0, \ldots, n-1\}$ either e_u or $\Lambda(P_u)$ vanishes.

Since we defined $a_m = q$, the polynomial $T := \Lambda \cdot S - R$ contains only monomials $x^a y^b$ with $b > b_m$. We conclude that $\mathrm{ord}_{P_0}(T) \geq \mathrm{ord}_{P_0}(y^{b_m+1})$. □

One easily shows that in $\mathcal{R}$ the first part of the key equation (4) is equivalent to

$$y^{b_m+1} \mid (\Lambda S - R),$$

or in a notation that is more similar to that used for RS codes

$$\Lambda \cdot S \equiv R \pmod{y^{b_m+1}}.$$

We end this section with an easy lemma on the degree of the minimal error-locator polynomial.

Lemma 2.4 *Let $t = |I|$ be the number of errors and Λ the minimal error-locator polynomial. Let N be minimal such that the dimension $\ell(NP)$ of the Riemann–Roch space satisfies $\ell(NP) > t$. Then*

$$t \leq \rho(\Lambda) \leq N \leq t + g.$$

Proof: The first inequality follows since the number of zeros of Λ is less than or equal to its degree. The second follows by elementary linear algebra. The last follows from the Riemann–Roch Theorem. □

3 Solutions to the key equation

In this section, we prove a partial converse to Proposition 2.3, following [11] including some simplifications and corrections from [3]. We assume that (Λ, R) is a solution of the key equation. Our main result (Corollary 3.2) states that the solution Λ is an error-locator polynomial, provided the number of errors t is small enough.

Proposition 3.1 *Let (Λ, R) be a solution to the key equation. Write $\mu = \rho(\Lambda)$. Then*

$$R - \Lambda U \in L((\mu + \ell)P + Q - (q+1)(b_m + 1)P_0).$$

Proof: The definition of U (2, 3) implies that the poles of $R - \Lambda U$ are contained in $\{P_i\}_{i\in I} \cup \{P\}$. Moreover, in P_i with $i \in I \setminus \{0\}$, the rational function $R - \Lambda U$ has at most a simple pole. The order of the pole in P equals $-\rho(R - \Lambda U)$. Since $\rho(\Lambda) = \mu$ by definition, we conclude that

$$\rho(R - \Lambda U) \le \rho(\Lambda) + \max\left(\rho(\frac{R}{\Lambda}), \rho(U)\right).$$

Since (Λ, R) is a solution to the key equation (4), it follows that $\rho(R) - \rho(\Lambda) \le \ell$. The definition of U implies that $\rho(U) \le (q-1)(q+1) - q = q^2 - q - 1 = 2g - 1$. We conclude that

$$\rho(R - \Lambda U) \le \mu + \max(\ell, 2g - 1).$$

The definition of b_m implies that

$$(q+1)b_m \le m^\perp \le (q+1)b_m + q,$$

therefore $2g - 1 \le \ell$. We conclude that $\rho(R - \Lambda U) \le \mu + \ell$.

It remains to estimate $\mathrm{ord}_{P_0}(R - \Lambda U)$. Lemma 2.2 states that

$$S - U = (\sum_{i\in I} e_i u_i) y^{b_m+1}.$$

We conclude that $\mathrm{ord}_{P_0}(S - U) \ge \mathrm{ord}_{P_0}(y^{b_m+1}) = (q+1)(b_m+1)$ if $0 \notin I$. In the case that $0 \in I$, we have $u_0 = 1/x$, and hence

$$S - U = e_0 \frac{y^{b_m+1}}{x} + \sum_{i\in I\setminus\{0\}} e_i u_i y^{b_m+1}.$$

Since (Λ, R) is a solution to the key equation (4), it follows that $T = S\Lambda - R$ satisfies $\mathrm{ord}_{P_0}(T) \ge (q+1)(b_m+1)$. We may write $R - \Lambda U = \Lambda(S - U) - T$. Therefore it follows that

$$\mathrm{ord}_{P_0}\Lambda(S - U) \ge (q+1)(b_m+1) - 1.$$

□

The next corollary gives a necessary condition for a solution to the key equation to be an error-locator polynomial. Recall that $t = \deg(Q)$ is the number of errors.

Corollary 3.2 *Let (Λ, R) be a solution of the key equation (4) with $t+\rho(\Lambda) < d^*$. Then Λ is an error-locator polynomial.*

Proof: Let (Λ, R) be as in the statement of the corollary, and define U as before. Proposition 3.1 implies that $R-U\Lambda \in L(Q+(\mu+\ell)P-(q+1)(b_m+1)P_0)$, where $\mu = \rho(\Lambda)$. Since

$$\begin{aligned}\deg(Q + (\mu+\ell)P - (q+1)(b_m+1)P_0) =& t+\mu+\ell-(q+1)(b_m+1)\\ <& d^*+q^2-q-1-m^\perp-1 = 0,\end{aligned}$$

it follows from the Riemann–Roch Theorem that

$$L(Q + (\mu+\ell)P - (q+1)(b_m+1)P_0) = \{0\}.$$

Therefore

$$\frac{R}{\Lambda} = U. \tag{5}$$

The statement of the corollary now immediately follows, since R is a polynomial and U has poles in P_i for all $i \in I$. □

Note that in the situation of Corollary 3.2 the error values can be easily read off from $R/\Lambda = U$ and the definition of U. The corollary unfortunately only applies to situation with relatively few errors. For example, if (Λ, R) is a solution to the key equation of degree μ with $2\mu < d^*$ then Λ is an error-locator polynomial. Example 4.4 below shows that this is a real restriction. In § 5 we give a practical algorithm for computing an error-locator polynomial without restriction on the degree.

Proposition 3.3 *Suppose that Λ is an error-locator polynomial with $\rho(\Lambda) < d^*$. Let (Λ, R) be the corresponding solution of the key equation. Then*

$$\frac{R}{\Lambda} = U.$$

In particular, the error-locations $i \neq 0$ are exactly the simple poles of R/Λ and $i = 0$ is an error-location if and only if $\mathrm{ord}_{P_0}(R/\Lambda) = (q+1)(b_m+1)-1$. The error values are determined as in Lemma 4.2.

Proof: We suppose that Λ is an error-locator polynomial with $\mu := \rho(\Lambda) < d^*$. Then ΛU does not have poles outside P. As in the proof of Proposition 3.1 it follows that

$$R - \Lambda U \in L((\mu+\ell)P - (q+1)(b_m+1)P_0).$$

The assumption on μ implies that

$$\deg((\mu+\ell)P - (q+1)(b_m+1)P_0) = \mu + q^2 - q - 2 - m^\perp = \mu - d^* < 0.$$

As in the proof of Proposition 3.1, we conclude that

$$\frac{R}{\Lambda} = U.$$

□

4 The modified key equation

In this section, we address the problem that the syndrome polynomial S as defined in § 2 not only involves those syndromes which can be computed in terms of the received vector but also so-called unknown syndromes $s_{a,b}$ with $\rho(x^a y^b) > m^\perp$. The following lemma shows that to compute an error-locator polynomial Λ it suffices to use the known syndromes. To show this, we define the *modified syndrome polynomial* as :

$$\tilde{S} := \sum_{\rho(x^a y^b) \le m^\perp} s_{a,b} x^{a_m - a} y^{b_m - b}.$$

We say that $(\Lambda, \tilde{R}) \in \mathcal{R}^2$ satisfies the *modified key equation* if

$$\begin{cases} \operatorname{ord}_{P_0}(\Lambda \cdot \tilde{S} - \tilde{R}) & \ge \operatorname{ord}_{P_0}(y^{b_m+1}), \\ \rho(\tilde{R}) - \rho(\Lambda) & \le q a_m + (q+1) b_m - m^\perp - 1 =: \ell. \end{cases} \tag{6}$$

Lemma 4.1 *For $R \in \mathcal{R}$ define $\tilde{R} = R - \Lambda(S - \tilde{S})$. Then (Λ, R) is a solution to the key equation (4) if and only if $(\Lambda, \tilde{R})$ is a solution to the modified key equation.*

Proof: This follows immediately from the observation that $S - \tilde{S}$ only contains terms $x^a y^b$ with $\rho(x^a y^b) < q^2 + (q+1)b_m - m^\perp$. □

The following lemma is the modified analog to Corollary 3.2.

Lemma 4.2 *Let $(\Lambda, \tilde{R})$ be a solution to the modified key equation. Assume that $t + \rho(\Lambda) < d^*$.*

(a) Then

$$i \in I \Leftrightarrow \operatorname{ord}_{P_i} \frac{\tilde{R}}{\Lambda} = \begin{cases} -1 & \text{if } i \neq 0, \\ (q+1)(b_m+1) - 1 & \text{if } i = 0. \end{cases}$$

(b) If $i \in I$ then the polar part of $\tilde{R}/\Lambda$ in $P_i = (\alpha, \beta_i)$ is

$$\frac{e_i}{x - \alpha_i}.$$

Proof: For $i \neq 0$ this follows immediately from Corollary 3.2 and the properties of U, since

$$\frac{\tilde{R}}{\Lambda} = \frac{R}{\Lambda} + \tilde{S} - S,$$

and $\tilde{S} - S$ is a polynomial.

For $i = 0$, we recall that $\tilde{S} - S$ only contains monomials $x^a y^b$ with $\rho(x^a y^b) = qa + (q+1)b = q^2 + (q+1)b_m - m^\perp$. These monomials satisfy therefore $\operatorname{ord}_{P_0}(x^a y^b) = a + (q+1)b < q + (q+1)b_m < (q+1)(b_m+1) - 1$. □

Summarizing, we have shown that error-locator polynomials always correspond to solutions of the modified key equation. Under the assumption $t+\rho(\Lambda) < d^*$, we have shown moreover, that every solution of the key equation is also an error-locator polynomial. Example 4.4 below shows that this is not true if we omit this assumption. (A similar example can also be found in [11].)

Proposition 4.3.(a) allows in many cases to recognize whether solutions of the modified key equation are error-locator polynomials or not. To formulate it, we need to introduce some notation. Let $(\Lambda, \tilde{R})$ be a solution of the modified key equation. Let $\mathcal{Z} \subset \mathcal{X}_q(\mathbb{F}_{q^2})$ be the set of poles different from P of $\tilde{R}/\Lambda$, together with P_0 if $\mathrm{ord}_{P_0}(\tilde{R}/\Lambda) = (q+1)(b_m+1)-1$.

For $\tilde{P} \in \mathcal{Z}$, we write

$$\frac{e_{\mathcal{Z},\tilde{P}}}{x-\tilde{\alpha}}$$

for the polar part of $\tilde{R}/\Lambda$ in $\tilde{P} = (\tilde{\alpha}, \tilde{\beta})$. We denote the degree of $\mathcal{Z}$ by $t(\Lambda)$.

Note that $\tilde{R}/\Lambda$ does not have poles outside $\mathcal{X}_q(\mathbb{F}_{q^2})$ since then $\tilde{R} - \Lambda U \in L((\mu+\ell)P + Q - (q+1)(b_m+1)P_0)$ (Proposition 3.1). This implies that $t(\Lambda)$ is less than or equal to the number of $\mathbb{F}_{q^2}$-rational zeros of Λ. Recall that $\boldsymbol{r}$ denotes the received vector.

Proposition 4.3 *Suppose that $(\Lambda, \tilde{R})$ is a solution of the modified key equation, such that $\mu := \rho(\Lambda) < d^*$.*

(a) If Λ is the minimal error-locator polynomial then $\mu \leq N$, where N is minimal such that $\ell(NP) > t(\Lambda)$. In particular $\mu \leq t(\Lambda) + g$.

(b) Suppose that $\mathcal{Z} \subset \mathcal{X}_q(\mathbb{F}_{q^2}) \setminus \{P\}$, and write $\boldsymbol{e}_{\mathcal{Z}} := (e_{\mathcal{Z},P_i})$. Then Λ is an error-locator polynomial if and only if $\boldsymbol{r} - \boldsymbol{e}_{\mathcal{Z}}$ is a codeword.

Proof: The statement of (a) follows immediately from Lemma 2.4.

The forward direction of (b) was already shown in Proposition 3.3. The backward implication follows similarly. □

Example 4.4 Let $q = 4$ and $n = 64$, i.e. $\{P_0, \ldots, P_{63}\} = \mathcal{X}_q(\mathbb{F}_{q^2}) \setminus \{P\}$. We consider the code corresponding to the Riemann–Roch space $L(mP)$ with $m = 51$. One computes that $\ell = 12$, $m^\perp = 23$, $b_m = 4$ and $d^* = 13$. We represent $\mathbb{F}_{16} = \mathbb{F}_2[x]/(x^4+x+1)$. Let $\alpha \in \mathbb{F}_{16}$ be a zero of $x^4+x+1 = 0$. Note that $\alpha \in \mathbb{F}_{16}^*$ is an element of order 15. Suppose that $t = 6$ and the error positions are

$$P_1 = (1, \alpha), P_2 = (1, \alpha^2), P_3 = (1, \alpha^4), P_4 = (1, \alpha^8), P_5 = (\alpha, \alpha^6), P_6 = (\alpha^2, \alpha^3)$$

and the error values $e_i = 1$ for $i = 1, \ldots, 6$.

One computes that the modified syndrome polynomial is

$$\begin{aligned}\tilde{S} = \alpha^5 x^3 y^4 + \alpha^2 x^4 y^3 + \alpha^{10} x^2 y^4 + \alpha^{13} x^3 y^3 + \alpha^4 x^4 y^2 + \alpha^2 x y^4 \\ + \alpha^{11} x^2 y^3 + \alpha^4 x^4 y + \alpha^5 y^4 + \alpha^{11} x^2 y^2 + \alpha^6 x^3 y + \alpha^8 x^4 \\ + \alpha^{14} y^3 + \alpha^{11} x y^2 + \alpha^9 x^2 y.\end{aligned}$$

The smallest solution (Λ, R) of the modified key equation satisfies $\rho(\Lambda) = 9$; a solution is for example given by

$$\begin{aligned} \Delta_4 &= xy + \alpha^5 x^2 + y + \alpha^{13} x + \alpha^8, \\ R_4 &= \alpha y^4 + \text{ lower terms.} \end{aligned}$$

(This can for example be computed with the algorithm of the next section. The terminology Δ_i, R_i will also be explained there.) Therefore a minimal error-locator polynomial satisfies $\rho(\Lambda) \geq 9$.

Note that Δ_4 has exactly three $\mathbb{F}_{q^2}$-rational zeros. Since $\ell(8P) > 3$, the criterion of Proposition 3.3.(a) implies that Δ_4 is not an error-locator polynomial. (Note that we do not even need to consider R_4 to check this.) This is no contradiction to Lemma 4.2, since $t + \mu = 6 + 9 \geq d^*$. In the next section, we describe how to compute the minimal error-locator polynomial.

We now explain how to check the statement of Proposition 4.3.(b) in practice. For this, we define

$$T := \sum_i e_{\mathcal{Z},P_i} u_i,$$

and

$$G := \frac{\tilde{R}}{\Lambda} - T. \tag{7}$$

As in the proof of Proposition 3.3, one shows that $G \in L(\ell P)$. We may compute the coefficients of $G = \sum G_i \varphi_i$, for example by substituting suitable points P_i in (7).

We first suppose that Λ is an error-locator polynomial with $\rho(\Lambda) < d^*$. Let (Λ, R) be the solution of the key equation corresponding to Λ. Then it follows from Proposition 3.3 that $T = U = R/\Lambda$. Therefore

$$G = \frac{\tilde{R}}{\Lambda} - \frac{R}{\Lambda} = \tilde{S} - S.$$

We conclude that if Λ is an error-locator polynomial, then the coefficients of G are precisely the unknown syndromes.

Now suppose that $(\Lambda, \tilde{R})$ is any solution of the modified key equation. Recall that we already computed the values $e_i = e_{Z,P_i}$ as residues of $\tilde{R}/\Lambda$. For $0 \leq i \leq \ell$, we write $\varphi_i = x^{a_m - a} y^{b_m - b}$. We may now check whether the coefficients G_i satisfy

$$G_i = s_{a,b} = \sum_{j=0}^{n} e_{\mathcal{Z},P_j} \varphi_{a,b}(P_j). \tag{8}$$

Example 4.5 We apply this procedure to the solution (Δ_4, R_4) of Example 4.4. (Recall that we have in fact already shown that Δ_4 is not an error-locator polynomial.) We first compute the residues of R_4/Δ_4 in the three $\mathbb{F}_{q^2}$-rational zeros of Δ_4 which we denote by $\{P_1, P_2, P_3\}$. We find:

$$e_1 := \mathrm{Res}_{(\alpha,\alpha^6)} \frac{R_4}{\Delta_4} = \alpha^{10}, e_2 := \mathrm{Res}_{(\alpha^5,\alpha^{11})} \frac{R_4}{\Delta_4} = \alpha^{11}, e_3 := \mathrm{Res}_{(\alpha^2,\alpha^3)} \frac{R_4}{\Delta_4} = \alpha^2.$$

From this we compute that the polynomial G defined by (7) is given by

$$G = \alpha^{14} + \alpha^{13}x + \alpha^{12}y + x^2 + \alpha^{14}y^2.$$

This can for example by computed by substituting $x = \alpha, \alpha^2, \alpha^5$ and simplifying the rational functions in one variable one obtains.

One checks that

$$G_0 = \alpha^{14} \neq s_{4,4} = \sum_{j=1}^{3} e_j \varphi_{4,4}(P_j) = \alpha^{12}.$$

This gives a second proof that Λ is not an error-locator polynomial.

5 Calculating the minimal error-locator polynomial.

In this section we explain a practical algorithm for solving the modified key equation. The algorithm explained here is an extension of a modification of the subresultant-sequence algorithm introduced by Shen [14]. The main difference between the two algorithms is that his algorithm uses matrix operations to calculate a subresultant sequence which is known to yield the same polynomials (up to a constant factor) as the Euclidean algorithm for univariate polynomials. In contrast, our algorithm mimics the steps of the Sugiyama algorithm used for decoding RS codes more closely. Further, our algorithm easily extends to the decoding of most error patterns of weight $\left\lfloor \frac{d^*-1}{2} \right\rfloor$, such an extension is not given in [14].

The core part of the algorithm is the computation of a new basis Δ_i of $L(\mu P)$, together with polynomials R_i that satisfy the first part of the modified key equation, i.e.

$$\mathrm{ord}_{P_0}(\tilde{S}\Delta_i - R_i) \geq (q+1)(b_m + 1).$$

Here μ is a bound that will be adapted as we go along. The (Δ_i, R_i) are defined in such a way that $\rho(\Delta_i) = \rho(\varphi_i)$ and moreover the degree of R_i is minimal. Certain additional choices are made to make the polynomials uniquely determined.

1. Compute (Δ_i, R_i) **until** $\rho(R_i) - \rho(\Delta_i) \leq \ell$

The computation is based on the division of a bivariate polynomial by j other bivariate polynomials. In such a division, j quotient polynomials $\gamma_1, \ldots, \gamma_j$ and one remainder polynomial are obtained by repeatedly subtracting (multiples of) the j divisor polynomials until no term in the dividend is a multiple of the leading monomial of any divisor. However, it is necessary to fix an ordering of the divisors, as with a different ordering different quotients and remainder might be obtained. A description of the division procedure can be found in [1] or [8].

For initialization, we set $\Delta_0(x,y) = 1$, $R_0(x,y) = \tilde{S}(x,y)$. While the first part of the modified key equation (6) is trivially fulfilled for these polynomials, the second part is only fulfilled if all syndrome elements are zero which is equivalent to the received word being a codeword. Consequently, no decoding is necessary in such a case.

While the Sugiyama algorithm is the repeated division of a remainder polynomials, we use a slightly different setup in order to ensure that $\rho(\Delta_i) = \rho(\varphi_i)$ and that Δ_i is monic: first, we select the monomial φ_{i_1} where $\varphi_i = y\varphi_{i_1}$ or $\varphi_{i_1} = x^{a-1}$ if $\varphi_i = x^a$. As dividend, we then use the polynomial $\theta = yR_{i_1}$ (or xR_{i_1}) and divide θ by $R_{i-1}, \ldots, R_0 = \tilde{S}$, the ordering implicitly given as the reverse order of calculation. To ensure, that the leading term of Δ_i is not changed, we need to check the additional criterion

$$\rho(\Delta_j) + \rho(\gamma_{i,j}) < \rho(\varphi_i).$$

The following considerations explain this criterion: we obtain

$$\theta = \sum_{j<i} \gamma_{i,j} R_j + R_i.$$

Of course, using $\theta = yR_{i_1}$ (the case where $\theta = xR_{i_1}$ works completely analogously, so it is not given) and $R_j = \Delta_j \tilde{S} \mod y^{b_m+1}$, we can rewrite this to

$$y\Delta_{i_1}\tilde{S} = \sum_{j<i} \gamma_{i,j}\Delta_j \tilde{S} + R_i \mod y^{b_m+1}.$$

By setting $\Delta_i = y\Delta_{i_1} - \sum_{j<i} \gamma_{i,j}\Delta_j$, we obtain a pair (Δ_i, R_i) fulfilling the first part of the key equation. However, we further wanted to make sure that $\rho(\Delta_i) = \rho(\varphi_i)$, and because $y\Delta_{i_1}$ - which is obtained from the definition of θ - already yields the correct leading term, it is sufficient to ensure that $\sum_{j<i} \gamma_{i,j}\Delta_j$ has smaller degree. On the other hand, our algorithm works just as well if we stop each iteration as soon as one term in θ could not be canceled - no matter how many further steps we take, $\rho(R_i)$ will remain the same.

In each iteration, we check if the second part of (6) is fulfilled. If it is fulfilled, we proceed with the next step. Otherwise, we increase i and perform the next division as described before. A summary of this algorithm in pseudocode can be found in [8].

2. Obtaining the error-locator polynomial

From the previous step, we obtained (Δ_i, R_i) with $\rho(R_i) - \rho(\Delta_i) \leq \ell$. Put $\mu_i = \rho(\Delta_i)$. If there exists an error-locator polynomial Λ with $\rho(\Lambda) \leq \mu_i$ then we may write

$$\Lambda = \sum_{j \leq i} c_j \Delta_j, \tag{9}$$

since the Δ_i form a basis of $L(\mu_i P)$. The corresponding polynomial R such that (Λ, R) is a solution of the modified key equation can then be written as

$$R = \sum_{j \leq i} c_j R_j.$$

The division algorithm implies that the $\rho(R_j)$ are all distinct. Therefore for all j with $c_j \neq 0$ we have that

$$\rho(R_j) - \rho(\Delta_i) \leq \ell.$$

This greatly limits the linear combinations to consider. For all possible linear combinations $\Lambda = \sum_{j \leq i} c_j \Delta_j$ check whether (Λ, R) is an error-locator polynomial using the criteria from Proposition 3.3. If we find only one such pair, this is the solution we are looking for and the decoding algorithm stops. In the next part, we explain what to do if no solution exists, and in step 4 we explain what to do if more than one solution is available.

3. Treatment of decoding failures
If no error locator is found by the previous basis, increase μ and i to include the next (Δ_i, R_i) which is a solution to the key equation, and repeat step 2. If we can bound the number of errors (say $t \leq \left\lfloor \frac{d^*-1}{2} \right\rfloor$), then by Lemma 2.4 $\rho(\Lambda) \leq t + g$, so we know that it suffices to compute the Δ_i until $\rho(\Delta_i)$ reaches this bound, hence the number of additional steps is limited. If no further pair (Δ_i, R_i) fulfilling the key equation is found we know that more than t errors occurred and if $t \geq \left\lfloor \frac{d^*-1}{2} \right\rfloor$ unique decoding cannot be guaranteed any more. More details about this situation can be found in Section 7.

4. Choosing a single solution
As mentioned in step 2, we might find more than one possible solution to the key equation that also is an error locator. We call those solution pairs $(\Lambda, \tilde{R})$ candidates. We want to obtain a single error locator, and there are several options how to choose one of the candidates: mostly, we chose to select the candidate where Λ has the largest number of zeros, but for which $\tilde{R}/\Lambda$ has not more than $\left\lfloor \frac{d^*-1}{2} \right\rfloor$ poles.

The second option is due to Proposition 4.3 (b): for all candidate pairs that fulfill the first criterion we check if $\boldsymbol{r} - \boldsymbol{e}_{\mathcal{Z}}$ is a codeword. In simulations, all errors of weight up to $\left\lfloor \frac{d^*-1}{2} \right\rfloor$ could be uniquely (and correctly) decoded if the evaluation criterion was used. Unfortunately, performing error evaluation for all candidate codewords increases the complexity compared to the first criterion that also yields a small error rate (cf. [8, Sec. V]).

Example 5.1 We continue with Example 4.4. Since the expressions for R_i are rather long, we omit them.

i	Δ_i	$\rho(\Delta_i)$	$\rho(R_i)$
0	1	0	32
1	$x+\alpha^5$	4	36
2	$y+\alpha^{12}x+1$	5	26
3	$x^2+\alpha^5x+\alpha^3$	8	21
4	$xy+\alpha^5x^2+y+\alpha^{13}x+\alpha^8$	9	20

We see that $i = 4$ is the first value such that $\rho(\Delta_i) - \rho(R_i) \leq \ell = 12$. This implies immediately that the minimal error-locator polynomial has degree greater than or equal to 9. Moreover, we have already checked that Δ_4 is not an error-locator polynomial (Example 4.4). Note that $\rho(R_3) - \rho(\Delta_4) = 12$ but that for $i = 0, \dots, 2$ we have that $\rho(R_i) - \rho(\Delta_4) > 12$. This implies that an error-locator polynomial Λ of degree 9, if it exists, may be written as

$$\Lambda = \Delta_4 + c\Delta_3, \qquad \text{with } c \in \mathbb{F}_{16}.$$

Checking the criterion of Proposition 3.3 for all values of c yields that

$$\Delta_4 + \alpha^{14}\Delta_3 = (x-1)(y+\alpha^{12}x+1)$$

is an error-locator polynomial.

The method described in this section does not only work for the minimal error-locator polynomial. Continuing as before, one finds for example

$$\Lambda_2 = \Delta_6 + \alpha^2\Delta_4 = (x-1)(x^2+\alpha^{10}y+\alpha^{13}x+\alpha^{12}),$$

which is a further error-locator polynomial corresponding to the same codeword.

What remains to complete the decoding process is to find the error values. But once the error locator polynomial has been determined this is a relatively simple task. Basically, two different possibilities exist: the first is to use the error locator polynomial to recursively extend the syndrome polynomial, such an approach is described e.g. in [13]. The other possibility is to exploit 4.2 and calculate the residues as was done in Example 4.5 to obtain the error values. Note that the latter approach is similar to using the well-known Forney formula for RS codes [12, p. 195].

We conclude this section with a proof of correctness of our algorithm.

Theorem 5.2 *Our algorithm gives a minimal solution to the key equation.*

Proof: Our algorithm constructs a series of pairs (Δ_i, R_i), where every possible $\rho(\Delta_i)$ is obtained in increasing order. Since our algorithm stops as soon as the second part of (4) is fulfilled, it is sufficient to show that our algorithm always calculates a polynomial R_i of minimal degree given $\rho(\Delta_i)$. This, however, is a direct consequence of the fact that our algorithm is a Groebner basis calculation. □

6 The complexity of the division algorithm

First we will give an estimate for the complexity of determining an error locator polynomial. For this, we will count the number of necessary multiplications. Throughout this section, we will use the notations introduced in Section 5, and let τ be the maximum number of correctable errors. Because the check matrix can be precalculated, the computation of the syndrome polynomial has complexity $\mathcal{O}(nm^{\perp})$. The selection of i_1 has linear complexity, and the calculation of θ has $\mathcal{O}(m^{\perp})$. Next, we need to divide θ by several other polynomials. This division requires up to $\rho(\theta)\tau$ checks followed by $\rho(\theta)$ subtractions of another polynomial, where a subtraction has complexity $\mathcal{O}(m^{\perp})$. Up to τ such divisions have to be performed, hence the overall complexity of this step is $\mathcal{O}(\rho(\theta)m^{\perp}\tau) = \mathcal{O}(n^3)$. The calculation of the polynomials Δ_i can be performed with the same complexity if it is performed in line with the division. If we need to select one solution according to step 4 (we consider the selection based on the number of zeros of Λ), we have to find the zeros of up to q^2 polynomials. Finding the zeros of a polynomial with $\rho(\Delta_i) \leq \tau$ has complexity $\mathcal{O}(n\tau)$, and with $q = n^{1/3}$ and a clever setup - evaluating the basis polynomials and checking the zeros of the linear combinations in the "evaluation domain" - we get a complexity of $\mathcal{O}(n^2)$.

While the previous considerations showed cubic decoding complexity, simulations indicate that this is not the true value: the choice of the polynomial θ allows us to perform each division by subtracting only $\mathcal{O}(2q)$ instead of $\mathcal{O}(\rho(\theta))$ polynomials, hence reducing the complexity of Algorithm 1 to $\mathcal{O}(n^{7/3})$. An algorithm of the same complexity was denoted "fast" in [6] and [13].

As stated in [2], the complexity of the evaluation step is $\mathcal{O}(n^2)$, hence it does not affect the overall performance of our algorithm.

7 A basis for decoding beyond half the minimum distance

While the sum in (9) only contains one summand if $t \leq \left\lfloor \frac{d^*-1}{2} \right\rfloor$, this is no longer true if one wants to correct more than that number of errors. In this section, we will first derive the number n_b of basis elements that are obtained if $t > \left\lfloor \frac{d^*-1}{2} \right\rfloor$ errors shall be corrected. Afterwards, we discuss why it is usually not feasible to use this basis without further information about the error, e.g. reliability information or the presence of other errors in the same positions.

In the original description of our algorithm, a stopping criterion was used to determine along with the minimal error locator polynomial also an upper bound on the number of errors in the received word. Unfortunately, no such stopping criterion exists if $t > \left\lfloor \frac{d^*-1}{2} \right\rfloor$, so we have to modify our algorithm slightly: first, we fix a number t of errors that we want to correct. We then calculate all pairs (Δ_i, R_i) with $\rho(\Delta_i) \leq \rho(\varphi_t)$. Of course, Proposition 2.3 is still true if $\rho(\Lambda) > \left\lfloor \frac{d^*-1}{2} \right\rfloor$, so those pairs with $\rho(R_i) > \rho(\varphi_t) + \ell$ are not used in the basis.

Let us first assume that $\rho(R_i) = \rho_S - \rho(\varphi_i)\,\forall i$. Then the pairs not used in the basis have

$$\rho(R_i) = \rho_S - \rho(\Delta_i) > \rho(\varphi_t) + \ell$$

or $\rho(\Delta_i) < m^\perp + 1 - \rho(\varphi_t)$. We can hence give the number of basis pairs as

$$n_b = |\Phi_{\rho(\varphi_t)}| - |\Phi_{m^\perp - \rho(\varphi_t)}| = t + 1 - |\Phi_{m^\perp - \rho(\varphi_t)}|. \tag{10}$$

Now let there exist a pair $(\bar{\imath}, i)$ of indices where w.l.o.g. $\bar{\imath} < i$ and we have that $\rho(R_i) = \rho_s - \rho(\varphi_{\bar{\imath}})$ and conversely, then we can have one of the following situations:

1. Both $\rho(R_i) \leq \rho(\varphi_t) + \ell$ and $\rho(R_{\bar{\imath}}) \leq \rho(\varphi_t) + \ell$, then both pairs are selected for the basis and n_b does not change. The situation is similar if both $\rho(R_i) > \rho(\varphi_t) + \ell$ and $\rho(R_{\bar{\imath}}) > \rho(\varphi_t) + \ell$, as then neither of the pairs is selected.
2. If $i < t$ and $\rho(R_i) > \rho(\varphi_t) + \ell$ but $\rho(R_{\bar{\imath}}) \leq \rho(\varphi_t) + \ell$ we can again calculate n_b by (10), but now the pair $(\Delta_{\bar{\imath}}, R_{\bar{\imath}})$ is picked instead of (Δ_i, R_i).
3. If $i > t$, we obtain $\rho(R_{\bar{\imath}}) \leq \rho(\varphi_t) + \ell$ for sure, so the $\bar{\imath}$th pair is included in the basis. However, the pair (Δ_i, R_i) is not even calculated because of $\rho(\Delta_i)$, so in this situation n_b is larger than indicated by (10), which turns out to be only a lower bound on the number of basis elements.

Finally, we want to consider a special case: let $m^\perp - \rho(\varphi_t) > 2g - 2$ and $t \geq g$, then we know that

$$|\Phi_{m^\perp - \rho(\varphi_t)}| = m^\perp - \rho(\varphi_t) - g + 1 = m^\perp - (t + g) - g + 1 = m^\perp - t - 2g + 1.$$

In this case, we can rewrite (10) to

$$n_b = t + 1 - (m^\perp - t - 2g + 1) = 2t - (m^\perp - 2g + 2) + 2 = 2t - d^* + 2.$$

If we further write the number of errors as $t = \frac{d^*-1}{2} + t_0$, then

$$n_b = 2t_0 + 1.$$

This result coincides with known results for RS codes, see e.g. [7].

Though it is possible to use our algorithm to calculate a basis for all solutions of the key equation, the complexity for finding the optimal solution increases rapidly with an increasing number of errors: while it is quadratic if two basis polynomials are present, this complexity increases with a factor of $q^2 = n^{2/3}$ for every additional basis element. Consequently, it is desirable to be able to select one or some of the possible linear combinations before checking them for their number of zeros. Such methods are e.g. collaborative decoding of interleaved Hermitian codes - where we assume that the same positions of several codewords were corrupted during transmission - or power decoding, which works by virtually extending a Hermitian code into an interleaved code at the receiver. In [9], these two approaches are described in more detail and bounds are given for the achievable decoding radii.

Acknowledgment

We thank Martin Bossert for suggesting the project and Martin Bossert, Vladimir Sidorenko, Alexander Zeh and Antonia Wachter for helpful discussions and comments on a previous version of this paper.

References

[1] D.A. Cox, J. Little, D. O'Shea, *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, Springer, 1992.

[2] D. Ehrhard, Über das Dekodieren algebraisch-geometrischer Codes, *Dissertation, Universität Düsseldorf*, 1991.

[3] J.I. Farrán, Decoding algebraic geometry codes by the key equation, *Finite Field Appl.* 6, 207–217, 2000.

[4] V.D. Goppa, Codes on algebraic curves, *Dokl. Akad. Nauk SSSR* 259, 1289–1290, 1981.

[5] V. Guruswami, M. Sudan, Improved decoding of Reed–Solomon and algebraic-geometric codes, *39th Annual symposium on foundations of computer science*, 1998.

[6] J. Justesen, K. Larsen, H. Jensen, T. Høholdt, Fast decoding of codes from algebraic plane curves, *IEEE transactions on information theory* 38, 111–119, 1992.

[7] S. Kampf, M. Bossert, S. Bezzateev, Some results on list decoding of interleaved Reed-Solomon codes with the extended euclidean elgorithm, *Proc. Coding Theory Days in St. Petersburg 2008*, 31–36, 2008.

[8] S. Kampf, M. Bossert, I.I. Bouw, Solving the key equation for Hermitian codes with a division algorithm, *Submitted to IEEE International Symposium on Information Theory*, St. Petersburg, 2011.

[9] S. Kampf, Bounds on collaborative decoding of interleaved Hermitian codes with a division algorithm and virtual extension, Submitted to *Third international castle meeting on coding theory and applications*, Cardona, Barcelona, 2011.

[10] F. J. MacWilliams, N.J.A. Sloane, *The theory of error-correcting codes*, North-Holland Mathematical Library, 1988.

[11] S.C. Porter, B.-Z. Shen, R. Pellikaan, Decoding geometric Goppa codes using an extra place, *IEEE transactions on information theory* 38, 1663–1676, 1992.

[12] R.M. Roth, *Introduction to coding theory*, Cambridge University Press, 2006.

[13] S. Sakata, J. Justesen, Y. Madelung, H. Elbrønd, T. Høholdt, Fast decoding of algebraic-geometric codes up to the designed minimum distance, *IEEE transactions on information theory*, 41, 1672–1677, 1995.

[14] B.-Z. Shen, Solving a congruence on a graded algebra by a subresultant sequence and its application, *Journal Symbolic Computation* 14, 505–522, 1992.

[15] A. Skorobogatov, S.G. Vlăduţ, On the decoding of algebraic-geometric codes, *IEEE Trans. Inform. Theory* IT-36, 1051–1060, 1990.

[16] H. Stichenoth, *Algebraic function fields and codes*, Second edition, GTM 254, Springer-Verlag, 2009

[17] M.A. Tsfasman, S.G. Vlăduţ, *Algebraic-geometric codes*, Kluwer Academic Publishers Group, 1991.

[18] M.A. Tsfasman, S.G. Vlăduţ, T. Zink, Modular curves, Shimura curves and Goppa codes better than the Varshmov–Gilbert bound, *Math. Nachr.* 109, 21–28, 1982.

Institute of Pure Math
Ulm University
irene.bouw@uni-ulm.de

Institute of Telecommunications
and Applied Information Theory
Ulm University
sabine.kampf@uni-ulm.de