Decoding square-free Goppa codes over $\mathbb{F}_p$

Paulo S. L. M. Barreto[1]*, Richard Lindner[2], and Rafael Misoczki[1]

[1] Departamento de Engenharia de Computação e Sistemas Digitais (PCS),
Escola Politécnica, Universidade de São Paulo, Brazil.
`{pbarreto,rmisoczki}@larc.usp.br`
[2] Department of Computer Science,
Technische Universität Darmstadt, Germany.
`rlindner@cdc.informatik.tu-darmstadt.de`

Abstract. We propose a new, efficient decoding algorithm for square-free (irreducible or otherwise) Goppa codes over $\mathbb{F}_p$ for any prime p. If the code in question has degree t and its average code distance is at least $(4/p)t + 1$, the proposed decoder can uniquely correct up to $(2/p)t$ errors with high probability. The correction capability is higher if the distribution of error magnitudes is not uniform, approaching or reaching t errors when any particular error value occurs much more often than others or exclusively. This makes the method interesting for (semantically secure) cryptosystems based on the decoding problem for permuted and punctured Goppa codes.

Keywords: coding theory, error correction, efficient algorithms, coding-based cryptosystems

Mathematics Subject Classification (2000): 94A60, 14G50, 94B35

1 Introduction

Public-key cryptosystems based on coding theory, known for nearly as long as the very concept of asymmetric cryptography itself, have recently been attracting renewed interest because of their apparent resistance even against attacks mounted with the help of quantum computers, constituting a family of so-called post-quantum cryptosystems [4]. However, not all error-correcting codes are suitable for cryptographic applications. The most commonly used family of codes for such purposed is that of Goppa codes, which remain essentially unharmed by cryptanalysis efforts despite considerable efforts and progress in the area.

Goppa codes [9] were introduced in 1970 as a subfamily of Generalized Reed-Solomon codes, which in turn are subfield subcodes of alternant codes. Let $q = p^m$ for some prime p and some $m > 0$. A Goppa code $\Gamma(L, g)$ over $\mathbb{F}_p$ is determined by a sequence $L \in \mathbb{F}_q^n$ of distinct values, and a polynomial $g \in \mathbb{F}_q[x]$ whose roots are disjoint from L. Goppa codes have by design a minimal distance

* Supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under research productivity grant 303163/2009-7.

at least $\deg(g)+1$ by virtue of being alternant. Certain codes are known to have better minimum distances than this lower bound. Thus, binary Goppa codes where g is square-free are known to have a larger minimum distance of at least $2\deg(g)+1$ instead. A family of codes where g is not square-free have minimum distance at least $\deg(g)+\gamma-1$ for some $2<\gamma<\deg(g)-1$, which is known as the Hartmann-Tzeng bound for Goppa codes [16]. Codes where $g=h^{r-1}$ for some irreducible monic polynomial $h \in \mathbb{F}_q[x]$ and some power r of p dividing q, dubbed "wild" codes [5], have minimum distance at least $r\deg(h)+1$ rather than $(r-1)\deg(h)+1$ [20]. Determining the true minimum distance of a general Goppa code remains largely an open problem, yet it is an important metric as it determines not only how many errors can always be uniquely corrected, but indirectly the security level and the key sizes of the cryptosystems based on each given code.

Apart from brute force, known decoding methods for alternant codes can in general correct only about half as many errors as a binary square-free Goppa code is in principle able to correct [1, 19] (see also [11]). Even the Guruswami-Sudan algorithm [10], which exceeds the $t/2$ limit, can only correct about $n-\sqrt{n(n-t)} \approx t/2+(t/2)^2/(2n-t)$ errors. In contrast, Patterson's algorithm can correct all t design errors of binary Goppa codes, as can an alternant decoder using the equivalence $\Gamma(L,g)=\Gamma(L,g^2)$ albeit at a larger computational cost. Bernstein's list decoding method [3] goes somewhat further, attaining a correction capability of $n-\sqrt{n(n-2t-2)} \approx t+1+(t+1)^2/2(n-t-1)$ errors for binary irreducible Goppa codes, although decoding is ambiguous if the actual distance is not proportionally higher. Similar techniques can in principle correct about $n-\sqrt{n(n-rt)} \approx rt/2+(rt/2)^2/(2n-rt)$ errors for wild codes [5].

Our contribution in this paper is a decoding algorithm for square-free (irreducible or otherwise) Goppa codes over $\mathbb{F}_p$ for any prime p. The method generalizes Patterson's approach and can potentially correct up to $(2/p)t$ errors, on the condition that a suitable short vector can be found in a certain polynomial lattice. In particular, our method corrects $(2/3)t$ errors in characteristic 3, exceeding the $t/2$ barrier when the average code distance is at least $(4/3)t+1$. The correction is observed to be unique with overwhelming probability for irreducible ternary Goppa codes chosen uniformly at random, hinting that the average (as opposed to the minimum) code distance is high enough for the vast majority of such codes. Besides, our proposal can probabilistically correct a still larger number of errors that approaches and reaches t depending on the distribution of error magnitudes. For instance, the method corrects up to t errors with high probability if all error magnitudes are known to be equal.

This feature outperforms even wild codes and the associated decoding methods described in [5], and is particularly interesting for cryptographic applications like McEliece encryption [12] under the Fujisaki-Okamoto or similar semantic security transform [8], where error magnitudes can be chosen by convention to be all or nearly all equal. In that case, even if an attacker could somehow derive a generic alternant decoder from the public code that is typical in such systems (a strategy exploited e.g. in [7]), he will not be able to correct more than about

$t/2$ errors out of roughly t that can be corrected with the private trapdoor enabled by our proposal, facing an infeasible workload of $\binom{n}{t/2}/\binom{t}{t/2}$ guesses to mount a complete attack. This makes Goppa codes in odd characteristic, which have already been shown to sport some potential security advantages over binary ones [18], even more attractive in practice.

The remainder of this document is organized as follows. We provide theoretical preliminaries in Section 2. We recapitulate Patterson's decoding algorithm for binary irreducible Goppa codes in Section 3, and extend it to square-free codes in characteristic p in Section 4, showing that it can correct $(2/p)t$ errors in general and up to t errors depending on the distribution of error magnitudes. We conclude in Section 5.

2 Preliminaries

2.1 Error correcting codes

Let p be prime and $q = p^m$ for some $m > 0$. Let $L = (L_0, \ldots, L_{n-1}) \in \mathbb{F}_q^n$ be a sequence (called the *support*) of $n \leqslant q$ distinct elements, and let $g \in \mathbb{F}_q[x]$ be an irreducible monic polynomial of degree t such that $g(L_i) \neq 0$ for all i. For any word $e \in \mathbb{F}_p^n$ we define the corresponding *Goppa syndrome* polynomial $s_e \in \mathbb{F}_q[x]$ to be:

$$s_e(x) = \sum_{i=0}^{n-1} \frac{e_i}{x - L_i} \mod g(x).$$

Thus the syndrome is a linear function of e. The $[n, n-mt, \geqslant 2t+1]$ *Goppa code* over $\mathbb{F}_p$ with support L and generator polynomial g is the kernel of the syndrome function, i.e. the set $\Gamma(L, g) := \{e \in \mathbb{F}_p^n \mid s_e \equiv 0 \mod g\}$.

Writing $s_e(x) := \sum_i s_i x^i$ for some $s \in \mathbb{F}_q^n$, one can show that $s^{\mathrm{T}} = He^{\mathrm{T}}$ with

$$H = \begin{bmatrix} g_t & 0 & \ldots & 0 \\ g_{t-1} & g_t & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & \ldots & 1 \\ L_0 & L_1 & \ldots & L_{n-1} \\ L_0^2 & L_1^2 & \ldots & L_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ L_0^{t-1} & L_1^{t-1} & \ldots & L_{n-1}^{t-1} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{g(L_0)} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(L_1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{g(L_{n-1})} \end{bmatrix} \tag{1}$$

Thus $H = TVD$ where T is a $t \times t$ Toeplitz matrix, V is a $t \times n$ Vandermonde matrix, and D is an $n \times n$ diagonal matrix.

The *syndrome decoding problem* consists of computing the error pattern e given its syndrome s_e. Knowledge of the code structure in the form of the support L and the polynomial g makes this problem solvable in polynomial time, with some constraints relating the weight of e to the degree of g.

2.2 Polynomial lattices

Let $A \in \mathbb{F}_q[x]^{n\times m}$ be a polynomial matrix of rank r. The (polynomial) lattice $\Lambda(A)$ over $\mathbb{F}_q[x]$ spanned by the rows of A is

$$\Lambda(A) = \{(u_0, \dots, u_n)A \in \mathbb{F}_q[x]^m \mid (u_0, \dots, u_n) \in \mathbb{F}_q[x]^n\}.$$

The notion of length which we will use for $f \in \mathbb{F}_q[x]$ is $|f| = \deg(f)$. For polynomial vectors $v \in \mathbb{F}_q[x]^k$ we adopt the notion of maximal degree length: $|v| = \max_i |v_i|$. This notion is coarse enough that, contrary to integer lattices where finding even an approximation to the shortest vector by a constant factor is a hard problem [14], reducing a basis for a polynomial lattice can be achieved in polynomial time. The following result by Mulders and Storjohann holds [15]:

Theorem 1. *There exists an algorithm which finds the shortest nonzero vector in the $\mathbb{F}_q[x]$-module generated by the rows of A with $O(mnrd^2)$ operations in $\mathbb{F}_q$, where $d = \max\{\deg(A_{ij}) \mid 1 \leqslant i \leqslant n, \leqslant j \leqslant m\}$.*

The algorithm whose existence is established by Theorem 1 is based on converting a given lattice basis to the so-called weak Popov form, formally defined in Appendix A. Accordingly, we will refer to it as WeakPopovForm. Interestingly, the actual complexity of WeakPopovForm for the kind of lattices we will encounter is lower than the estimate provided by Theorem 1. Appendix A also contains a description of the WeakPopovForm algorithm and its cost behaviour in the context of decoding.

3 Patterson's decoding method

We briefly recapitulate Patterson's decoding algorithm [17], which will provide the basis for the general algorithm we propose. The goal, of course, is to compute the error pattern e given its syndrome s_e and the structure of $\Gamma(L, g)$.

Let $q = 2^m$, and assume we are given a binary Goppa code $\Gamma(L, g)$ where the monic polynomial g is irreducible. We define the Patterson locator polynomial $\sigma \in \mathbb{F}_q[x]$ as:

$$\sigma(x) := \prod_{e_i=1} (x - L_i). \tag{2}$$

The name *locator polynomial* comes from the fact that the roots of σ clearly indicate where errors occurred, since $\sigma(L_j) = 0 \Leftrightarrow e_j = 1$. Lifting to $\mathbb{F}_q(x)$ we obtain

$$\sigma'(x) = \sum_{e_i=1} \prod_{\substack{e_j=1 \\ j\neq i}} (x - L_j) = \sum_{e_i=1} \frac{1}{x - L_i} \prod_{e_j=1} (x - L_j) = \sigma(x) \sum_i \frac{e_i}{x - L_i},$$

and hence, in $\mathbb{F}_q[x]/g(x)$,

$$\sigma'(x) = \sigma(x) s_e(x) \mod g(x). \tag{3}$$

This is called the *key equation*. It only determines σ' mod g (up to a constant factor that is chosen to make σ monic), but if the number of errors is restricted not to exceed t, $\deg(\sigma')$ is guaranteed to be bound by $t-1$ and hence $\sigma' \mod g = \sigma'$. By integration, σ is determined up to an arbitrary squared polynomial, i.e. $\sigma = \sigma_0 + h^2$ for some h where σ_0 is the unique primitive of σ_0 that is free of squared terms. But since we require that $\deg(\sigma) \leqslant t$, necessarily $\deg(h) \leqslant \lfloor t/2 \rfloor < t$, and hence h itself (rather than only h mod g) is uniquely determined as well. This means that the whole σ of degree up to t is uniquely determined, enabling the correction of t errors.

Being a binary polynomial, $\sigma(x)$ modulo $g(x)$ can be written as

$$\sigma(x) = a_0(x)^2 + x a_1(x)^2$$

for some $a_0(x)$, $a_1(x)$ with $\deg(a_0) \leqslant \lfloor t/2 \rfloor$ and $\deg(a_1) \leqslant \lfloor (t-1)/2 \rfloor$, and hence

$$\sigma'(x) = 2\, a_0(x)\, a_0'(x) + a_1(x)^2 + 2\, a_1(x)\, a_1'(x)\, x = a_1(x)^2,$$

since the characteristic is 2. Therefore

$$a_1(x)^2 = \sigma'(x) = \sigma(x) s_e(x) = \left(a_0(x)^2 + x a_1(x)^2\right) s_e(x) \mod g(x),$$

whence

$$a_0(x) = a_1(x)\sqrt{x + 1/s_e(x)} \mod g(x). \tag{4}$$

The last equation is actually a Bézout relation $a_0(x) = a_1(x)v(x) + \lambda(x)g(x)$ with $v(x) := \sqrt{x + 1/s_e(x)} \mod g(x)$, which can be solved for $a_0(x)$ and $a_1(x)$ with the restriction $\deg(a_j) \leqslant \lfloor (t-j)/2 \rfloor$ (and also $\lambda(x)$ but it is not used) using the extended Euclidean algorithm. Solutions (a_0, a_1) can also be seen as short vectors in the lattice spanned by the rows of the following matrix:

$$A = \begin{bmatrix} g & 0 \\ v & 1 \end{bmatrix}$$

in the sense that the degrees of these polynomials are much smaller than uniformly random vectors, since $(\lambda, a_1)A = (\lambda g + a_1 v, a_1) = (a_0, a_1)$ for some $\lambda \in \mathbb{F}_q[x]$, by virtue of Equation 4. Therefore, algorithm WeakPopovForm can be used to find candidate solutions (a_0, a_1).

At first glance there is no guarantee that a short vector in the lattice generated by A yields the desired solution; in other words, being short is a necessary condition, but in principle not a sufficient one. However, the fact that in the binary case the minimum code distance is known to be at least $2t + 1$ actually restricts σ to a single candidate, so that algorithm WeakPopovForm is bound to find it. Thus, decoding is always successful up to t introduced errors.

4 Decoding codes over $\mathbb{F}_p$

We now show how to generalize Patterson's decoding algorithm so as to correct errors for codes defined over general prime fields. Thus, let $q = p^m$ for some prime p and some $m > 0$, and assume we are given an irreducible Goppa code $\Gamma(L, g)$ over $\mathbb{F}_p$.

Let $\phi \in \mathbb{F}_p \setminus \{0\}$ be a constant scalar. We define the generalized error locator polynomial to be

$$\sigma(x) := \prod_i (x - L_i)^{e_i/\phi}, \tag{5}$$

where the value e_i/ϕ is lifted from $\mathbb{F}_p$ to $\mathbb{Z}$ (i.e. to the corresponding integer representative in range $0 \dots p-1$) upon exponentiation. One can easily see that this definition actually coincides with Patterson error locator polynomials as defined by Equation 2 for $p = 2$. Lifting Equation 5 to $\mathbb{F}_q(x)$ and taking the derivative, we have

$$\begin{aligned}\sigma'(x) &= \sum_j (e_j/\phi)(x - L_j)^{e_j/\phi - 1} \prod_{i \neq j} (x - L_i)^{e_i/\phi} \\ &= (1/\phi) \sum_j \frac{e_j}{x - L_j} \prod_i (x - L_i)^{e_i/\phi} \\ &= (1/\phi)\sigma(x) \sum_j \frac{e_j}{x - L_j},\end{aligned}$$

which over $\mathbb{F}_q[x]$ reduces to

$$\phi\sigma'(x) = \sigma(x)s(x) \mod g(x). \tag{6}$$

This is the key equation of the proposed method, which generalizes Patterson's decoder to Goppa codes over $\mathbb{F}_p$. The actual ϕ must be chosen so as to minimize the degree of σ (and hence maximize the number of correctable errors). One cannot expect to know *a priori* the value of ϕ, but since there are only $p - 1$ possibilities, the error correction strategy will be to try each of them in turn.

As in the binary case, solving Equation 6 with the right ϕ only determines $\sigma' \bmod g$ (up to a constant factor that is chosen to make σ monic), but this remainder coincides with σ' by imposing the restriction that $\deg(\sigma')$ be less than t. By integration, σ is determined up to an arbitrary p-th power, i.e. $\sigma = \sigma_0 + h^p$ for some h where σ_0 is the unique primitive of σ_0 that is free of p-th power terms. But since we require that $\deg(\sigma) \leqslant t$, necessarily $\deg(h) \leqslant \lfloor t/p \rfloor < t$, and hence h itself (rather than only $h \bmod g$) is uniquely determined as well. This means that the whole σ of degree up to t is uniquely determined. Notice that the maximum number of correctable errors can be, and usually is, less than t, since the degree of σ exceeds the number of roots in the presence of multiple roots.

Indeed, let w denote the maximum number of actually correctable errors, and let w_v denote the number of times the magnitude v occurs in an error pattern of weight w, so that $\sum_v w_v \leqslant w$. The constraint for correctability is thus $\deg(\sigma) = \sum_v (v/\phi)w_v \leqslant t$. In the extreme situation when the weight of the error pattern reaches w, the most often error magnitude occurs $w_{\max} \geqslant w/(p-1)$ times, attaining the lower bound when all error magnitudes occur with equal frequency. In that case, $\deg(\sigma) \leqslant \sum_v (v/\phi)w_{\max} = (1+2+\dots+(p-1))w/(p-1) = wp/2 \leqslant t$, and hence the maximum number of errors that can be corrected independently of the distribution of magnitudes is $w = (2/p)t$.

Since the method coincides with Patterson's for $p = 2$, it is not surprising that t errors can be corrected in that case. However, in characteristic 3 the number of potentially correctable errors is $(2/3)t$, exceeding the limit of $t/2$ errors attainable by previously known decoding methods for codes of degree t except when the Goppa polynomial is a $(p-1)$-th power of an irreducible polynomial (our method, by contrast, applies when that polynomial is square-free, as we will see in Section 4.1).

Despite the low general limit of $(2/p)t$ correctable errors for $p > 3$, it is still possible to exceed that limit in any odd characteristic if the distribution of error magnitudes is unbalanced. Indeed, all that is required to get a chance of uniquely decoding a word containing $w \leqslant t$ errors is that $\deg(\sigma) \leqslant t$ for some choice of ϕ and that the actual distance from the right codeword to any other codeword be at least $2w + 1$. If the code is not equidistant, this possibility remains open even when the minimum code distance is not high enough.

The actual number of correctable errors depends heavily on the distribution of error magnitudes and has to be computed in a case-by-case basis, always laying in the range $(2/p)t$ to t. In particular, if all error magnitudes are equal, up to t errors can be corrected. This is especially useful for cryptographic applications involving an all-or-nothing transform, as it happens e.g. for a semantically secure encryption scheme involving the McEliece one-way trapdoor function [12] and the Fujisaki-Okamoto conversion [8]. In such scenarios, the magnitudes of the introduced errors can be chosen to be all or nearly all equal by convention, making the proposed decoder attractive for its higher decodability bound.

4.1 Solving the key equation

We now focus on actually solving Equation 6. Being a polynomial in characteristic p, $\sigma(x)$ can be written as

$$\sigma(x) = \sum_{k=0}^{p-1} x^k a_k(x)^p \tag{7}$$

for some $a_k(x)$ with $\deg(a_k) \leqslant \lfloor (t-k)/p \rfloor$, $0 \leqslant k \leqslant p-1$, and hence

$$\sigma'(x) = \sum_{k=0}^{p-1} \left(k x^{k-1} a_k(x)^p + p x^k a_k(x)^{p-1} a_k'(x) \right) = \sum_{k=1}^{p-1} k x^{k-1} a_k(x)^p$$

since the characteristic is p. Therefore

$$\phi \sum_{k=1}^{p-1} k x^{k-1} a_k(x)^p = \phi\sigma'(x) = \sigma(x) s_e(x) = \left(\sum_{k=0}^{p-1} x^k a_k(x)^p \right) s_e(x) \mod g(x),$$

whence

$$a_0 + \sum_{k=1}^{p-1} a_k(x) v_k(x) = 0 \mod g(x) \tag{8}$$

where $v_k(x) := \sqrt[p]{x^k + \phi k x^{k-1}/s_e(x)} \mod g(x)$. This Diophantine equation has to be solved for $a_k(x)$ with the stated restriction on the degrees. Solutions $(a_0, a_1, \ldots, a_{p-1})$ can be seen as short vectors in the lattice spanned by the rows of the matrix

$$A = \begin{bmatrix} g & 0\,0 \ldots 0 \\ -v_1 & 1\,0 \ldots 0 \\ -v_2 & 0\,1 \ldots 0 \\ \vdots & \vdots\,\vdots \ddots \vdots \\ -v_{p-1} & 0\,0 \ldots 1 \end{bmatrix}, \tag{9}$$

since $(\lambda, a_1, \ldots, a_{p-1})A = (\lambda g - \sum_{k=1}^{p-1} a_k(x)v_k(x), a_1, \ldots, a_{p-1}) = (a_0, a_1, \ldots, a_{p-1})$ for some $\lambda \in \mathbb{F}_q[x]$, by virtue of Equation 8. Therefore, algorithm WeakPopovForm can be used to find candidate solutions $(a_0, \ldots, a_{p-1})$.

The method is applicable whenever one can actually invert $s \bmod g$ and then compute the p-roots needed to define the v_k polynomials. This is always the case when g is irreducible, but not exclusively so. Indeed, to compute the v_k it suffices that g is square-free and that s is invertible modulo each of the irreducible factors of g, since in this case the v_k can be easily computed modulo those irreducible factors and finally recovered via the Chinese Remainder Theorem (see e.g. [13, Algorithm 2.121]).

Theorem 1 predicts a cost of $O(p^3t^2)$ $\mathbb{F}_q$ operations for computing short vectors in $\Lambda(A)$. However, as discussed in Appendix A the actual cost for the particular structure of A is only $O(p^2t^2)$ $\mathbb{F}_q$ operations.

4.2 Estimating the success probability

Regrettably the ability to find shorts vectors in lattice $\Lambda(A)$ does not mean that any such vector yields a solution to Equation 6. We will now see that, fortunately, the proposed method has a surprisingly favourable probability of finding the right $(a_0, \ldots, a_{p-1})$ that solves Equation 6.

In a successful decoding, the reduced basis for lattice $\Lambda(A)$ leads to candidates for σ with degree t onward, of which of course only the candidate with the smallest degree is the correct one. Spurious candidates of degree close to t result from random-looking short (albeit not shortest) vectors in the reduced basis and are usually harmless. But the fact that those short vectors are "random-looking" means they are also a threat: if by chance they are such that the coefficient of the highest-degree term in the associated spurious σ vanishes, $\deg(\sigma)$ becomes t or less. Since this is connected with the vanishing of a coefficient from $\mathbb{F}_q$, this event happens with probability $1/q$ assuming that short spurious vectors in $\Lambda(A)$ are approximately uniformly distributed.

In general, when trying to correct $w < t$ errors of equal magnitude for a uniformly random irreducible code, the top $t+1-w$ coefficients in the spurious σ must vanish to interfere with the decoding process, whence the probability of successful decoding is roughly $1 - 1/q^{t+1-w}$. This matches the empirically observed behaviour of the proposed method in odd characteristic. Not surprisingly, the method always succeeds for binary codes, since it reduces to Patterson's algorithm and the minimum code distance is known to be at least $2t+1$.

The probability of decoding $w \leqslant t' := (2/p)t$ errors of uniformly random magnitude for a uniformly random irreducible code is better still. Failure occurs if the top $t'+1-w$ coefficients in spurious σ polynomials vanish for all of the $p-1$ values of ϕ, since each of them can now lead to a valid σ. Thus, successful decoding happens with probability at least $1-(1/q^{t'+1-w})^{p-1}$, unless t' is already smaller than the minimum code distance in which case decoding is unconditionally successful. Therefore, the success probability is $1-(1/q^{(2/3)t+1-w})^2$ in characteristic 3 and simply 1 for $p > 3$ as the minimum distance is known beforehand to exceed $(4/p)t+1$, namely, it is at least $t+1$, and hence the method always yields a correct decoding.

4.3 Computing the error magnitudes

In contrast to generic alternant decoding methods, there is no need to compute an error evaluator polynomial to obtain the error magnitudes in the current proposal. After obtaining $\sigma(x)$ and finding its roots L_j, all that is needed to compute the corresponding error values e_j is to determine the multiplicity μ_j of each root, since one can see from Equation 5 that $e_j = \phi\mu_j$.

Computing μ_j is accomplished by determining how many times $(x-L_j) \mid \sigma(x)$ whenever $\sigma(L_j) = 0$, or alternatively by finding the highest derivative of σ such that $\sigma^{(h)}(L_j) = 0$ (and setting $\mu_j \leftarrow h$).

Since the value of ϕ is not known *a priori*, and even in scenarios where it is actually known beforehand, an additional syndrome check is necessary for each guessed ϕ, and the process usually must check all possible values of ϕ anyway since more than one solution may exist.

4.4 The completed decoder

We are finally ready to state the full decoding method explicitly in Algorithm 1. The polynomial decomposition of Equation 7 immediately suggests a simple and efficient way to compute the p-th roots needed at Step 12, namely, precompute $r(x) \leftarrow \sqrt[p]{x} \bmod g(x)$ and $r(x)^k \bmod g(x)$, and then compute the p-th root of $z(x) := \sum_k x^k z_k(x)^p$ as $\sqrt[p]{z(x)} \bmod g(x) \leftarrow \sum_k r(x)^k z_k(x)$. The tests in Steps 2 and 9 are unnecessary if g is irreducible. To find the zeroes of σ in Step 24 one can use the Chien search [6], in which case the multiplicities of each root can be determined as part of the search, or the Berlekamp trace algorithm [2].

5 Conclusion

We described a new decoding algorithm for square-free (in particular, irreducible) Goppa codes of degree t over $\mathbb{F}_p$ that can correct $(2/p)t$ errors in general, and up to t errors for certain distributions of error magnitudes of cryptographic interest. By attaining an correction capability of $(2/3)t$ errors in characteristic 3

Algorithm 1 Decoding p-ary square-free Goppa codes

```
INPUT: $\Gamma(L,g)$, a Goppa code over $\mathbb{F}_p$ where $g$ is square-free.
INPUT: $H \in \mathbb{F}_q^{r\times n}$, a parity-check matrix in the form of Equation 1.
INPUT: $c' = c + e \in \mathbb{F}_p^n$, the received codeword with errors.
OUTPUT: set of corrected codeword $c \in \Gamma(L,g)$ ($\varnothing$ upon failure).
 1: $s^{\mathrm{T}} \leftarrow Hc'^{\mathrm{T}} \in \mathbb{F}_q^n$, $s_e(x) \leftarrow \sum_i s_i x^i$. ▷ N.B. $Hc'^{\mathrm{T}} = He^{\mathrm{T}}$.
 2: if $\nexists\ s_e^{-1}(x) \bmod g(x)$ then
 3:     return $\varnothing$ ▷ this can only happen if $g(x)$ is composite
 4: end if
 5: $S \leftarrow \varnothing$
 6: for $\phi \leftarrow 1$ to $p-1$ do ▷ guess the correct scale factor $\phi$
 7:     for $k \leftarrow 1$ to $p-1$ do
 8:         $u_k(x) \leftarrow x^k + \phi k x^{k-1}/s_e(x) \mod g(x)$
 9:         if $\nexists\ \sqrt[p]{u_k(x)} \bmod g(x)$ then
10:             try next $\phi$ ▷ this can only happen if $g(x)$ is composite
11:         end if
12:         $v_k(x) \leftarrow \sqrt[p]{u_k(x)} \bmod g(x)$
13:     end for
14:     Build the lattice basis $A$ defined by Equation 9.
15:     Apply WEAKPOPOVFORM (Algorithm 2) to reduce the basis of $\Lambda(A)$.
16:     for $i \leftarrow 1$ to $p$ do
17:         $a \leftarrow A_i$ ▷ with $a_j$ component indices numbered in range $0 \ldots p-1$
18:         for $j \leftarrow 0$ to $p-1$ do
19:             if $\deg(a_j) > \lfloor (t-j)/p \rfloor$ then
20:                 try next $i$ ▷ not a solution
21:             end if
22:         end for
23:         $\sigma(x) \leftarrow \sum_j x^j a_j(x)^p$
24:         Compute the set $J$ such that $\sigma(L_j) = 0$, $\forall j \in J$.
25:         for $j \in J$ do
26:             Compute the multiplicity $\mu_j$ of $L_j$.
27:             $e_j \leftarrow \phi\mu_j$
28:         end for
29:         if $He^{\mathrm{T}} = s^{\mathrm{T}}$ then
30:             $S \leftarrow S \cup \{c' - e\}$
31:         end if
32:     end for
33:     return $S$
34: end for
```

with high probability, our method outperforms the best previously known decoder for that case, and suggests that the corresponding average distance is at least $(4/3)t+1$ for most irreducible ternary Goppa codes. Regardless of the characteristic, our proposal can correct a still larger number of errors that approaches and reaches t as the distribution of error magnitudes becomes ever more skewed toward the predominance of some individual value. The method can be viewed as generalizing Patterson's binary decoding procedure, and is similarly efficient in practice.

A further increase in the number of correctable errors may be possible by resorting to list decoding and by extracting more information from the decoding process along the lines proposed by Bernstein [3]. This in principle might enable the correction of approximately $n - \sqrt{n(n-(4/p)t)}$ errors in general, and possibly as many as $n - \sqrt{n(n-2t-2)}$ errors depending on the distribution of error magnitudes. Furthermore, the ability to correct close to t errors with high probability means that smaller keys might be adopted for coding-based cryptosystems. Properly chosen parameters would keep the probability of decoding failure below the probability of breaking the resulting schemes by random guessing, while maintaining the security at the desired level. We leave the investigation of such possibilities for future research.

References

1. E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill, New York, USA, 1968.
2. E. R. Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–715, 1970.
3. D. J. Bernstein. List decoding for binary Goppa codes. Preprint, 2008. `http://cr.yp.to/papers.html#goppalist`.
4. D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography.* Springer, 2008.
5. D. J. Bernstein, T. Lange, and C. Peters. Wild McEliece, 20108. Preprint.
6. R. T. Chien. Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10(4):357–363, 1964.
7. J.-C. Faugère, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology – Eurocrypt'2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010.
8. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – Crypto'1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
9. V. D. Goppa. A new class of linear error correcting codes. *Problemy Peredachi Informatsii*, 6:24–30, 1970.
10. V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.
11. F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*, volume 16. North-Holland Mathematical Library, 1977.
12. R. McEliece. A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report, DSN PR 42–44, 1978. `http://ipnpr.jpl.nasa.gov/progressreport2/42-44/44N.PDF`.

13. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, USA, 1999.
14. D. Micciancio. The shortest vector problem is np-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001.
15. T. Muldersa and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35:377–401, 2003.
16. C.-S. Park, G.-L. Feng, and K. K. Tzeng. The new minimum distance bounds of Goppa codes and their decoding. *Designs, Codes and Cryptography*, 9(2):157–176, 1996.
17. N. J. Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.
18. C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In *Post-Quantum Cryptography Workshop – PQCrypto'2010*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010.
19. Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding Goppa codes. *Information and Control*, 27(1):87–99, 1975.
20. Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. Further results on Goppa codes and their applications to constructing efficient binary codes. *IEEE Transactions on Information Theory*, 22(5):518–526, 1976.

A The weak Popov form

For ease of reference, we provide here a concise description of the Mulders-Storjohann polynomial lattice reduction algorithm based on the weak Popov form. We closely follow the exposition in [15].

Definition 1. *For* $1 \leqslant i \leqslant n$ *the* i-th pivot index vector I^M *of a matrix* $M \in \mathbb{F}_q[x]^{n\times m}$ *is defined as follows: if* $M_{ij} = 0$ *for all* $1 \leqslant j \leqslant m$*, then* $I_i^M = 0$*, otherwise*

1. $\deg(M_{ij}) \leqslant \deg(M_{i,I_i^M})$ *for* $1 \leqslant j < I_i^M$*,*
2. $\deg(M_{ij}) < \deg(M_{i,I_i^M})$ *for* $I_i^M < j \leqslant m$*.*

Definition 2. *The* carrier set C^M *of a matrix* $M \in \mathbb{F}_q[x]^{n\times m}$ *is the set* $\{1 \leqslant i \leqslant n \mid I_i^M \neq 0\}$*. The* i-th pivot element *of* M*, denoted* P_i^M*, is the element* $P_i^M := M_{i,I_i^M}$ *when* $I_i^M \neq 0$*, otherwise* $P_i^M := 0$*.*

Definition 3. *A matrix* $M \in \mathbb{F}_q[x]^{n\times m}$ *is said to be in* weak Popov form *if the positive pivot indices of* M *are all different, i.e. if* $\forall k, \ell \in C^M : k \neq \ell \Rightarrow I_k^M \neq I_\ell^M$*.*

The following theorem establishes that writing a matrix in weak Popov form yields short vectors in the lattice spanned by its rows.

Theorem 2 ([15]). *If matrix* $M \in \mathbb{F}_q[x]^{n\times m}$ *is in weak Popov form and* ℓ *is such that* $\deg(P_\ell^M) = \min_{1\leqslant i\leqslant n}\{\deg(P_i^M)\}$*, then all vectors in the* $\mathbb{F}_q[x]$*-module generated by the rows of* M *have degree at least* $\deg(P_\ell^M)$*.*

Proof. See [15, Lemma 8.1]. $\square$

If $k \in C^M$, $\ell \neq k$ and $\deg(M_{\ell,I_k^M}) \geqslant \deg(P_k^M)$, there are unique $c \in \mathbb{F}_q$ and $e \in \mathbb{N}$ such that $\deg(M_{\ell,I_k^M} - cx^e P_k^M) < \deg(M_{\ell,I_k^M})$. In that case we call the operation $M_\ell \leftarrow M_\ell - cx^e M_k$ the *simple transformation* of row k on row ℓ. If $I_\ell^M = I_k^M$, the transformation is called of the *first kind*. Then an efficient algorithm to put a matrix in weak Popov form stems from the following observation:

Theorem 3 ([15]). *$M \in \mathbb{F}_q[x]^{n\times m}$ is not in weak Popov form iff one can apply a simple transformation of the first kind on M, that is, not all non-zero pivot indices of M are different.*

Proof. See [15, Lemma 2.1]. $\square$

Therefore, all one has to do to obtain the weak Popov form of a matrix M is to repeatedly check if M is already in the weak Popov form (by testing if all nonzero pivot indices are different) and, if it is not, apply a simple transformation of the first kind on it. This surprisingly simple algorithm has complexity $O(mnrd^2)$ $\mathbb{F}_q$ operations if the rank of M is r and d is a bound on the degree of all components of M.

In the case of the lattice basis A defined in Section 4.1 where $n = m = r = p$ and $d = t$, this would appear to amount to $O(p^3t^2)$ $\mathbb{F}_q$ operations. However, matrix A is quite sparse, containing only $O(p)$ rather than $O(p^2)$ elements, and as it becomes denser its components converge to short vectors of expected degree $O(t/p)$. Therefore the expected number of times a simple transformation of the first kind has to be applied decreases by a factor $O(p)$ on average, and the expected complexity of the algorithm for a lattice basis in the form of Equation 9 is only $O(p^2t^2)$ $\mathbb{F}_q$ operations.

This process is summarised in Algorithm 2, where $\mathrm{lead}(P)$ denotes the leading coefficient of $P \in \mathbb{F}_q[x]$ and $\mathrm{rep}(I^A)$ denotes the number of occurrences of the most frequent value among the nonzero components of I^A, i.e. $\mathrm{rep}(I^A) := \max\{\#\{j \mid I_j^A = v\} \mid v \neq 0\}$. Algorithm 2 is strikingly similar to the variant of the extended Euclidean algorithm usually employed in the decoding of alternant codes [19], and actually coincides with that method for $p = 2$.

B Decoding other families of codes?

For completeness, we briefly discuss whether and how one might attempt to use similar methods to decode a different family of alternant codes, including BCH codes and their permuted and/or punctured versions.

Let $L \in \mathbb{F}_q^n$ be a sequence of $n \leqslant q$ distinct nonzero elements, let $D \in \mathbb{F}_q^n$ be a sequence of nonzero elements, and let $H = \mathrm{vdm}(L)\,\mathrm{diag}(D)$. For any word $e \in \mathbb{F}_p^n$ we define the corresponding *alternant r-syndrome* polynomial $s_e \in \mathbb{F}_q[x]$ to be $s_e(x) := \sum_{i=0}^{r-1} s_i x^i$ where $s^{\mathrm{T}} := He^{\mathrm{T}}$, i.e.

$$s_i = \sum_{j=0}^{n-1} e_j D_j L_j^i.$$

Algorithm 2 (*aka* WeakPopovForm) Computing the weak Popov form

```
Input: $A \in \mathbb{F}_q[x]^{p\times p}$ in the form of Equation 9.
Output: weak Popov form of $A$.
 1: $\triangleright$ Compute $I^A$:
 2: for $j \leftarrow 1$ to $p$ do
 3:     $I_j^A \leftarrow$ if $\deg(A_{j,1}) > 0$ then $1$ else $j$
 4: end for
 5: $\triangleright$ Put $A$ in weak Popov form:
 6: while $\mathrm{rep}(I^A) > 1$ do
 7:     $\triangleright$ Find suitable $k$ and $\ell$ to apply simple transform of first kind:
 8:     for $k \leftarrow 1$ to $p$ such that $I_k^A \neq 0$ do
 9:         for $\ell \leftarrow 1$ to $p$ such that $\ell \neq k$ do
10:             while $\deg(A_{\ell,I_k^A}) \geqslant \deg(A_{k,I_k^A})$ do
11:                 $c \leftarrow \mathrm{lead}(A_{\ell,I_k^A})/\,\mathrm{lead}(A_{k,I_k^A})$
12:                 $e \leftarrow \deg(A_{\ell,I_k^A}) - \deg(A_{k,I_k^A})$
13:                 $A_\ell \leftarrow A_\ell - cx^e A_k$
14:             end while
15:             $\triangleright$ Update $I_\ell^A$ and hence $\mathrm{rep}(I^A)$ if necessary:
16:             $d \leftarrow \max\{\deg(A_{\ell,j}) \mid j = 1, \ldots, p\}$
17:             $I_\ell^A \leftarrow \max\{j \mid \deg(A_{\ell,j}) = d\}$
18:         end for
19:     end for
20: end while
21: return $A$
```

The alternant code $\mathcal{A}(L, D, r)$ consists of the set $\{e \in \mathbb{F}_p^n \mid s_e(x) \equiv 0\}$.

Using the formula for the sum of a geometric sequence $\sum_{i=0}^{r-1} u^i = (1 - u^r)/(1-u)$ whereby $\sum_{i=0}^{r-1} L_j^i x^i = (1 - x^r L_j^r)/(1 - xL_j) \equiv 1/(1 - xL_j) \bmod x^r$, one can see that

$$s_e(x) = \sum_{i=0}^{r-1} \sum_{j=0}^{n-1} e_j D_j L_j^i x^i = \sum_{j=0}^{n-1} e_j D_j \sum_{i=0}^{r-1} L_j^i x^i \equiv \sum_{j=0}^{n-1} \frac{e_j D_j}{1 - xL_j} \mod x^r.$$

The subfamily we will be interested in is that of alternant codes satisfying the restriction $D_j/L_j \in \mathbb{F}_p \setminus \{0\}$ for all j, so that each value D_j/L_j can be lifted to $\mathbb{Z}$ with a representative in range $1 \dots p-1$.

Let $\phi \in \mathbb{F}_p \setminus \{0\}$ be a constant scalar. We define the generalized error locator polynomial for this family as

$$\sigma(x) := \prod_i (1 - xL_i)^{e_i(D_i/L_i)/\phi}. \tag{10}$$

The error positions are revealed by the *inverses* of the components of L, which are the roots of this polynomial. This definition coincides with the usual alternant error locator polynomial when $p = 2$, in which case $D = L$ (hence, a permuted and/or punctured subcode of a binary BCH code).

Lifting Equation 10 to $\mathbb{F}_q(x)$ and taking the derivative of σ we get

$$\begin{aligned}\sigma'(x) &= \sum_j (e_j(D_j/L_j)/\phi)(1 - xL_j)^{e_j(D_j/L_j)/\phi - 1}(-L_j) \prod_{i \neq j} (1 - xL_i)^{e_i(D_i/L_i)/\phi} \\ &= -(1/\phi) \sum_j \frac{e_j D_j}{1 - xL_j} \prod_i (1 - xL_i)^{e_i(D_i/L_i)/\phi} \\ &= -(1/\phi)\sigma(x) \sum_j \frac{e_j D_j}{1 - xL_j},\end{aligned}$$

which over $\mathbb{F}_q[x]$ reduces to

$$-\phi\sigma'(x) = \sigma(x) s_e(x) \mod x^r. \tag{11}$$

This is the key equation for this family of codes.

Now most of the techniques developed for Goppa codes can be applied to solve Equation 11. The main difference is that the error magnitudes are computed as a function of the multiplicity μ_j of a root $1/L_j$ of σ as $e_j = \phi\mu_j/(D_j/L_j)$.

Writing $\sigma = \sum_{k=0}^{p-1} x^k a_k(x)^p$ for some $a_k(x)$ with $\deg(a_k) \leqslant \lfloor (r-k)/p \rfloor$, solutions to Equation 11 can be found as short vectors $(a_0, a_1, \dots, a_{p-1})$ in the polynomial lattice spanned by the rows of the matrix

$$A = \begin{bmatrix} x^r & 0 \dots 0 \\ -v_1 & 1 \dots 0 \\ \vdots & \vdots \ddots \vdots \\ -v_{p-1} & 0 \dots 1 \end{bmatrix}$$

where $v_k(x) := \sqrt[p]{x^k - \phi k x^{k-1}/s_e(x)} \bmod x^r$, provided that these p-th roots exist.

Here the major obstacle for this technique becomes apparent: inverting $s_e(x) \bmod x^r$ is usually fine, but computing the necessary p-th roots mod x^r is only very seldom possible. Specifically, assuming that the radicands are uniformly distributed polynomials in $\mathbb{F}_q[x]/x^r$ for a random code of this family, the probability that it is a p-th power mod x^r is only about $(q^{r/p}/q^r)^{p-1} = p^{-mr(p-1)^2/p}$, corresponding to the vanishing of all but a fraction $1/p$ of the r coefficients of each of the $p-1$ radicands needed to build matrix A.

Therefore there is scant chance that this would work in practice, except possibly for some highly contrived code whose syndromes lead to suitable radicands with high probability. It it an open problem whether such codes exist and, if so, what they might look like.