Higher-genus McEliece

Daniel J. Bernstein[1,2], Tanja Lange[3,2], and Alex Pellegrini[3]

[1] University of Illinois at Chicago, USA
[2] Academia Sinica, Taiwan
djb@cr.yp.to
[3] Eindhoven University of Technology, Netherlands
tanja@hyperelliptic.org
alex.pellegrini@live.com

Abstract. The best attacks known against the McEliece cryptosystem have cost growing exponentially with the number of errors corrected by the error-correcting code used in the cryptosystem. One can modify the cryptosystem to asymptotically increase this number of errors, for the same key size and the same ciphertext size, by generalizing classical binary Goppa codes to subfield subcodes of algebraic-geometry codes, and then moving from genus 0 to higher genus. This paper introduces streamlined algorithms for code generation and decoding for a broad class of these codes; shows that this class includes classical binary Goppa codes; and shows that moving to higher genus within this class decodes more errors than classical binary Goppa codes for concrete sizes of cryptographic interest. A notable feature of this paper's algorithms is the use of arithmetic on the Jacobian variety of the underlying curve.

1 Introduction

The original 1978 McEliece cryptosystem [41] reaches security level λ (meaning attack cost 2^λ) against all known non-quantum attacks when public keys have

Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. This work was supported by the Dutch Ministry of Education, Culture, and Science through Gravitation project "Challenges in Cyber Security (CiCS) – 024.006.037", by the Dutch Research Council (NWO) under grant 613.009.144; by the European Commision through projects 101096435 (CONFIDENTIAL6G), 101135475 (TALER), and 101190512 (PQCSA); by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC 2092 CASA–390781972; by the Academia Sinica Grand Challenge Seed Project AS-GCS-113-M07; and by the U.S. National Science Foundation under grant 2037867. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation" (or other funding agencies). Permanent ID of this document: af1552b26f016a86c75f06894d109172c682ceab. Date: 2025.09.11.

$(K + o(1))\lambda^2(\log_2 \lambda)^2$ bits, for a particular constant K described in the next paragraph. Known quantum attacks against the cryptosystem are like quantum attacks against strong symmetric ciphers, simply changing λ by a factor $2+o(1)$.

If one assumes "rate-R systematic-form" public keys, where the rate means the ratio between the dimension and the length of the code used, then the constant K above is $R(1-R)^{-1}(\log_2(1-R))^{-2}$. The minimum K as R varies is then slightly below $3/4$, achieved for $R \approx 4/5$. Meanwhile the ciphertexts, after Niederreiter compression [45], have just $(C + o(1))\lambda \log_2 \lambda$ bits where $C = (-\log_2(1-R))^{-1}$. The full version [13, Appendix A] reviews these calculations.

One of the remarkable features of the McEliece cryptosystem is that an extensive literature on McEliece attacks (surveyed in, e.g., [2]) has made no changes in these asymptotics. Replacing today's best attacks with the best attacks known in 1978 produces the same constants K and C: the attack improvements are so small that they are covered by the $o(1)$.

There is an easy explanation for the λ^2 in the key size: the keys are matrices. There is a more complicated explanation for the $(\log_2 \lambda)^2$ in the key size and the $\log_2 \lambda$ in the ciphertext size. Internally, the cryptosystem uses a decoder capable of correcting $(1 - R + o(1))n/\log_2 n$ errors for codes over $\mathbb{F}_2$ of length n and dimension $(R + o(1))n$, namely classical binary Goppa codes. For any fixed R, the security level λ against known attacks turns out to grow linearly with the number of errors, and thus to grow as $\Theta(n/\log n)$, forcing n to grow as $\Theta(\lambda \log \lambda)$.

The literature contains various proposals of modified cryptosystems that provide smaller keys by switching to codes or lattices having public structure—perhaps losing the security that the original McEliece cryptosystem provides. For example, the public-key matrix can be compressed to its first row in a cryptosystem requiring the matrix to be quasi-cyclic or quasi-dyadic; but such structure has also enabled various attacks, as illustrated by [26], [16], and [56]. Even if there are no further attack improvements, current quasi-cyclic proposals such as BIKE [3] and HQC [1] use larger ciphertexts than the original McEliece cryptosystem, so they do not necessarily use less communication than the original system: that depends on how many ciphertexts are transmitted per public key.

A different approach in the literature, and also the focus of this paper, is to preserve n, R, key size, ciphertext size, and the lack of public structure, but to add more errors. This increases security against the best attacks known; equivalently, this reduces key size *and* ciphertext size for the same level of security against known attacks. The challenge is to be able to efficiently decode those extra errors, given the private key.

List decoding for classical binary Goppa codes (see [10], [5], [11], and [6]) corrects more errors in polynomial time, but "more" is not visible at the level of detail of $(1 - R + o(1))n/\log_2 n$. An asymptotically larger impact, quantified below, comes from replacing classical binary Goppa codes with subfield subcodes of algebraic-geometry codes (AG codes). This replacement is a generalization: classical binary Goppa codes are $\mathbb{F}_2$ subcodes of genus-0 AG codes.

Using subfield subcodes of higher-genus AG codes was suggested by Janwa and Moreno in [36]. It is important to avoid confusing this with a different

suggestion from [36], namely using AG codes *without* a subfield construction. Non-subfield AG codes were attacked in [42], [27], [20], and [19]; the latter paper reports "a general attack proving that McEliece scheme based on AG codes from any curve of any genus is totally insecure". Simpler, faster attacks are known against the McEliece scheme using generalized Reed–Solomon codes, which are genus-0 AG codes; see [57] and [58]. The attacks cited in this paragraph are, as noted in [19, Section 7.3], not attacks against the McEliece cryptosystem using *subfield subcodes* of AG codes, such as classical binary Goppa codes.

One of the attractive features of the McEliece system using classical binary Goppa codes is that, when parameters are chosen to stop all known message-recovery attacks, there is a very large security margin for all known key-recovery attacks (e.g., attacks using support splitting [52]). The literature sometimes sets a weaker goal, namely asking whether classical binary Goppa codes can merely be *distinguished* from random; see, e.g., [21] and [50]. Known distinguishers are at least as expensive in higher genus as they are in genus 0; compare, e.g., [37] to [25]. The full version [13, Appendix B] reviews how slow these algorithms are.

As noted above, moving to higher genus makes the state-of-the-art message-recovery attacks more difficult. As an example of the asymptotic improvement, using $\mathbb{F}_2$ subcodes of AG codes defined over Hermitian curves increases the number of errors decoded in polynomial time to $(3/2)(1 - R + o(1))n/\log_2 n$. For any choice of R, this improvement increases security levels against known attacks by a factor $3/2 + o(1)$, reduces the key-size constant K by a factor $9/4$, and reduces the ciphertext-size constant C by a factor $3/2$. (These asymptotics do not seem to be in prior work; we cover them in [13, Appendix A].) Replacing Hermitian curves with further curves in the Garcia–Stichtenoth tower [31] does even better asymptotically, increasing the number of errors to $\Theta(n)$.

To summarize, moving to higher genus allows smaller keys, smaller ciphertexts, and the same reasons for confidence. Why, then, are concrete cryptosystem proposals such as Classic McEliece [2] still using genus 0?

One answer is that it is challenging to optimize curve choices for concrete sizes of cryptographic interest. Here is an illustration of the difficulty. The only binary-code example in [36] says that codes of length $n = 1024$ built from the genus-24 curve $y^4 + y = x^{17}$ over $\mathbb{F}_{256}$ have dimension at least $1023 - 8(48 + 2t)$ and distance at least $49 + 4t$ (correcting $24 + 2t$ errors), where t is a parameter. We point out that this is *worse* than using genus 0. Specifically, a classical binary Goppa code of length 1024 correcting $24 + 2t$ errors will have dimension at least $1024 - 10(24 + 2t)$, which stays above $1023 - 8(48 + 2t)$ until $1023 - 8(48 + 2t)$ drops to 47 (which is too small for security). There is a comment in [36, page 302] that the genus-24 codes are expected to have better parameters than the bounds suggest; this comment is not quantified, and provides no reason to believe that these codes are competitive.

Another answer is that having polynomial-time algorithms for key generation and decoding does not mean that the algorithms run at satisfactory speed. For classical binary Goppa codes, key generation uses $n^{\omega+o(1)}$ operations where ω is the linear-algebra exponent, and decoding uses $n^{1+o(1)}$ operations; the relative

	Unstructured	Binary	Secret perm	Secret poly	More errors
Original McEliece	yes	yes	yes	yes	no
[36, Example 2]	yes	no	yes	no	no
[36, Remark 1]	yes	yes	yes	no	no
[44, Table 2]	yes	no	yes	no	no
[37, Table 1]	yes	no	yes	yes	no
This paper	yes	yes	yes	yes	yes

Table 1. Comparison of security features of various McEliece variants (including [37], which appeared after the first announcements [47,48] of this paper's results), whether or not algorithms were provided. "Unstructured": public key is an unstructured full-rank matrix. "Binary": public key is over $\mathbb{F}_2$, not a larger field. "Secret perm": private key includes a secret support sequence (whether or not the support set is secret). "Secret poly": private key includes a secret Goppa polynomial. "More errors": higher genus was shown to decrypt more errors than the original McEliece cryptosystem for the same size of public key and ciphertext.

importance of these operations again depends on the number of ciphertexts per public key. For higher genus g, the recent decoders from [9] and [7] typically use $g^{(\omega-1+o(1))/2}(n+g)^{1+o(1)}$ operations, after an unclear number of operations for precomputation: neither [9] nor [7] states a code-generation algorithm.

1.1. Contributions of this paper. This paper presents a cautious McEliece generalization that outperforms the original cryptosystem. "Outperforms" means that, for the same key size and ciphertext size, this cryptosystem corrects more errors, not just asymptotically but for specific sizes of cryptographic interest; this increases the level of security against the best attacks known. "Cautious" means that this cryptosystem avoids adding extra structure to the original McEliece cryptosystem: public keys are still unstructured full-rank matrices over $\mathbb{F}_2$, and private keys include generalizations of the secrets traditionally used in the McEliece cryptosystem, namely a secret Goppa polynomial and a secret support sequence. See Table 1 for a comparison to other proposals.

We emphasize that, in defining a complete cryptosystem, we are doing much more than simply defining a family of codes: the user needs decoding algorithms (for decryption) *and* algorithms to generate codes in the first place (for key generation). Concretely, the main work in this paper is developing algorithms for (1) generating and (2) decoding a class of AG codes, while (3) ensuring that the genus-0 case covers the traditional range of codes used in the McEliece cryptosystem, and (4) showing that moving to higher genus corrects more errors for concrete sizes.

This paper focuses on $C_{a,b}$ curves (as defined in [43]), such as elliptic curves, odd-degree hyperelliptic curves, and Hermitian curves. All $C_{a,b}$ curves are plane curves with one point at infinity, so they have at most q^2+1 points over $\mathbb{F}_q$ and cannot reduce key sizes by more than a factor $4+o(1)$ compared to classical binary Goppa codes; but considering the genus-24 example above suggests that

the optimal genera for concrete cryptographic sizes will be small enough to fit into $C_{a,b}$ curves.

As a supplement to the full version [13] of this paper, we provide Sage scripts for the full algorithms: starting from a $C_{a,b}$ curve, apply the key-generation algorithm to sample a suitable AG code, and then apply the decoder to correct errors. The point here is not to evaluate how quickly the algorithms can run (this is not optimized C software; the Sage scripts from [2] for Classic McEliece also have large overhead) but to illustrate that the algorithms work.

1.2. Streamlining the algorithms. An unusual feature of this paper's AG-code algorithms is the use of arithmetic on the Jacobian variety of the curve, reducing a general degree-0 divisor on the curve to a reduced divisor. Jacobians are familiar objects in algebraic geometry (see, e.g., [28]), and divisor reduction plays a central role in cryptosystems relying on scalar multiplication on hyperelliptic curves (see [39]), but the following application of divisor reduction to AG codes does not appear to have been pointed out before: if A is a divisor (usually not degree 0 in context, but Picard reduction works the same way as Jacobian reduction), and $B = A - (\rho)$ is the result of reducing A modulo principal divisors, then one can decode the AG code $\{(f(P_1), \ldots, f(P_n)) : f \in \mathcal{L}(A)\}$ by multiplying the ith coordinate by $\rho(P_i)$ to obtain

$$\{((\rho f)(P_1), \ldots, (\rho f)(P_n)) : f \in \mathcal{L}(A)\} = \{(f(P_1), \ldots, f(P_n)) : f \in \mathcal{L}(B)\}$$

and then applying any decoder for the B code. Reduction puts a genus-dependent limit on the number of points in B, speeding up various approaches to decoding. The closest literature that we have found is [23, page 27], which suggests reduction as a way to compute parameters for hyperelliptic codes but does not consider decoding algorithms.

To further streamline decoding, we focus on the family of differential AG codes defined in Section 3, parameterized by two rational functions h and γ whose divisors describe the support of the code and the space of differentials of the code. As motivation for choosing these codes, we show that this family naturally extends the case of classical binary Goppa codes, along with associated computational tools (for instance, this family avoids the expensive computation that would otherwise be required by Lemma 5.6). After reviewing known results on parameters of subfield subcodes in Section 3.5, we present our two main algorithms: Section 4's key-generation algorithm, which uses Jacobian arithmetic to construct a code in this family given constraints on the number and multiplicity of zeros and poles, and Section 5's decoding algorithm, which relies on computing minima of modules described by matrices over $\mathbb{F}_{q^m}[x]$ (i.e., basis reduction for polynomial lattices). The decoder can be viewed as generalizing the decoder in [46] to $C_{a,b}$ curves, combining it with the interpolation tools in [40].

Compared to [9] and [7], our asymptotics look worse for large a but better for large g, making a direct comparison difficult since a and g are only partially related: consider, e.g., hyperelliptic curves and "random" $C_{a,b}$ curves. More importantly, our analysis of the number of errors corrected indicates that both

a and g should grow slowly with n, so cryptographic costs will be determined mainly by the exponent of n, which is the same for all of these decoders.

We do not claim that the algorithms here are as fast as algorithms available for $g = 0$. For the same key size (and thus, as we show, larger t), encryption costs will be essentially the same, but decryption will be more expensive since there are more polynomials that need to be handled as g grows. On the other hand, for equal security levels against known attacks, encryption should be less expensive for $g = 1$ than for $g = 0$ (since the matrices are smaller), and it is conceivable that decryption for $g = 1$ could be faster than decryption for $g = 0$. A full cost evaluation depends on the performance of optimized software, which is beyond the scope of this paper.

1.3. Structure of this paper. Section 2 provides the necessary background on AG codes, including key definitions and notation used throughout the paper. Section 3 introduces the family of codes considered for cryptographic applications and shows that it includes classical binary Goppa codes. Sections 4 and 5 present our two main algorithms, the code-construction algorithm and the decoding algorithm. Section 6 contains a full description of a PKE that generalizes the original McEliece scheme to the higher-genus case. Section 7 provides concrete examples for the case of elliptic curves (genus-1 curves), demonstrating that, as claimed, our framework and implementation correct more errors than the genus-0 case. The full version of this paper has appendices (1) analyzing asymptotic key sizes and ciphertext sizes for both classical Goppa codes and subfield subcodes of Hermitian codes, and (2) reviewing what the literature says about the costs of key-recovery attacks and key distinguishers.

2 Background and notation

This section fixes notation for the rest of the paper, and recalls the definition of algebraic geometry codes. Standard references for AG codes (including the differential-forms perspective) are [22, 55]. See also [34, 38] for Riemann-Roch computations.

2.1. Notation. Let p be a prime and $q = p^{m'}$ with $m' \geq 1$. For $m \geq 1$ let $\mathbb{F}_{q^m}$ be the finite field of size q^m and $\overline{\mathbb{F}}_{q^m}$ its algebraic closure. Let $\mathcal{X}$ be a smooth, projective, absolutely irreducible curve of genus $g(\mathcal{X})$ defined over $\mathbb{F}_{q^m}$ and $\mathbb{F}_{q^m}(\mathcal{X})$ and $\Omega(\mathcal{X})$ be the function field and the space of rational differential forms of $\mathcal{X}$, respectively. Denote by $\mathcal{P}_\mathcal{X}$ the set of places of $\mathcal{X}$. Places on $\mathcal{X}$ correspond to orbits of the points on the curve under the action of the Galois group $\mathrm{Gal}(\overline{\mathbb{F}}_{q^m}/\mathbb{F}_{q^m})$. In literature, places are sometimes called closed points. Let v_P be the discrete valuation at a place $P \in \mathcal{P}_\mathcal{X}$. The valuation ring of P is defined as $\mathcal{O}_P := \{f \in \mathbb{F}_{q^m}(\mathcal{X}) \mid v_P(f) \geq 0\}$. Then P is the maximal ideal of $\mathcal{O}_P$, and we define the degree of P as $\deg P := [\mathcal{O}_P/P : \mathbb{F}_{q^m}]$. In the correspondence between places and orbits of points, the degree gives the size of the orbit which equals the degree of the field extension over which the points in the orbit are defined.

2.1.1. Divisors. A divisor on $\mathcal{X}$ is a formal sum $\sum_{P\in\mathcal{P}_\mathcal{X}} a_P P$, with $a_P \in \mathbb{Z}$ and almost all $a_P = 0$. Denote by $\mathrm{Div}(\mathcal{X})$ the set of divisors on $\mathcal{X}$. Together with elementwise addition $\mathrm{Div}(\mathcal{X})$ forms a group. We define the support of $D \in \mathrm{Div}(\mathcal{X})$ as $\mathrm{supp}(D) = \{P \in \mathcal{P}_\mathcal{X} \mid a_P \neq 0\}$ and its degree $\deg D = \sum_{P\in\mathrm{supp}(D)} a_P \deg P$. Let $D^+ = \sum_{P\in\mathrm{supp}(D), a_P>0} a_P P$ and $D^- = \sum_{P\in\mathrm{supp}(D), a_P<0} |a_P| P$. Then $D = D^+ - D^-$. We say that D is effective, and write $D \geq 0$, if $a_P \geq 0$ for all $P \in \mathcal{P}_\mathcal{X}$. In particular, notice that both D^+ and D^- are effective divisors. For a rational function $f \in \mathbb{F}_q(\mathcal{X})$, define the divisor of f as $(f) = \sum_{P\in\mathcal{P}_\mathcal{X}} v_P(f)P$. Divisors associated with rational functions are called principal divisors and have degree zero. Denote by $\mathrm{Princ}(\mathcal{X})$ the set of principal divisors on $\mathcal{X}$, which is a subgroup of $\mathrm{Div}(\mathcal{X})$. We say that two divisors $D, D' \in \mathrm{Div}(\mathcal{X})$ are equivalent if there exists a divisor $A \in \mathrm{Princ}(\mathcal{X})$ such that $D = D' + A$. We extend the definition of discrete valuation to a divisor D as $v_P(D) = a_P$ for every $P \in \mathcal{P}_\mathcal{X}$. The divisor (ω) associated with a differential form $\omega \in \Omega(\mathcal{X})$ is called canonical divisor and has degree $2g(\mathcal{X}) - 2$.

2.1.2. The Riemann-Roch theorem. The Riemann-Roch space of D is defined as the $\mathbb{F}_{q^m}$-linear vector space $\mathcal{L}(D) = \{f \in \mathbb{F}_{q^m}(\mathcal{X}) \mid (f) \geq -D\}$. We let $\ell(D) = \dim \mathcal{L}(D)$ be its dimension. Similarly, the space of differentials of D is defined as $\Omega(D) = \{\omega \in \Omega(\mathcal{X}) \mid (\omega) \geq D\}$. Denote by $i(D) = \dim \Omega(D)$ its dimension. The Riemann-Roch theorem [55, Theorem 1.5.15] states that $\ell(D) - i(D) = \deg D - g(\mathcal{X}) + 1$.

For $\deg D \geq 2g(\mathcal{X}) - 1$ the dimension $i(D) = 0$, hence, for those divisors D we have $\ell(D) = \deg D - g(\mathcal{X}) + 1$.

2.1.3. $C_{a,b}$ curves. For describing a cryptosystem, we will consider a particular family of curves, called $C_{a,b}$ curves, described in 1993 by Miura [43]. Let a, b be two coprime integers, then the equation

$$x^b + u_{0,a}y^a + \sum_{0\leq i<b, 0\leq j<a, ai+bj<ab} u_{i,j}x^i y^j = 0, \tag{1}$$

with $u_{i,j} \in \mathbb{F}_q, u_{0,a} \neq 0$, and uniquely defined derivatives in all points over the algebraic closure, describes the affine plane model of an absolutely irreducible, non-singular curve $\mathcal{X}$ over $\mathbb{F}_q$ called a $C_{a,b}$ curve. These curves have a unique place at infinity P_∞, and Equation (1) implies that x and y as rational functions in $\mathbb{F}_q(\mathcal{X})$ have a pole at P_∞ of order a and b, respectively. The genus of a $C_{a,b}$ curve is $g(\mathcal{X}) = (a-1)(b-1)/2$. The coordinate rings $R_\mathcal{X}$ of these curves are Dedekind domains, which is relevant for this paper, as every fractional ideal is invertible and can be generated by at most two elements. For a more in-depth study on Dedekind domains we refer the reader to [18, Section 1.2].

2.2. AG codes. In this section we consider curves, functions, and divisors defined over $\mathbb{F}_{q^m}$ so that we can define subfield subcodes defined over $\mathbb{F}_q$. Let D be a divisor on $\mathcal{X}$ with $\mathrm{supp}(D) = \{P_1, \ldots, P_n\} \subseteq \mathcal{P}_\mathcal{X}$ with $\deg P_i = 1$ for every $i = 1, \ldots, n$. Let E be a divisor on $\mathcal{X}$ such that $\mathrm{supp}(E) \cap \mathrm{supp}(D) = \emptyset$, then for every function $f \in \mathcal{L}(E)$ the evaluation of f at D is $\mathrm{ev}_D(f) = (f(P_1), \ldots, f(P_n))$.

This induces an $\mathbb{F}_{q^m}$-linear map $\mathrm{ev}_D : \mathcal{L}(E) \to \mathbb{F}_{q^m}^n$ which is injective when $\deg E < n$. Under the assumption that $\deg E < n$, the image $\mathrm{ev}_D(\mathcal{L}(E))$ of ev_D is a subspace of $\mathbb{F}_{q^m}^n$, i.e., a linear code. A code stemming from such a construction is called an AG code. It has minimum distance $d \geq n - \deg E$ and dimension $k \geq \deg E - g(\mathcal{X}) + 1$. Whenever $2g(\mathcal{X}) - 1 \leq \deg E < n$, the equality $k = \deg E - g(\mathcal{X}) + 1$ holds. We refer to $\mathrm{ev}_D(\mathcal{L}(E))$ as the functional code $C_{\mathcal{X},\mathcal{L}}(D, E)$.

The image of the linear map $\mathrm{res}_D : \Omega(D - E) \to \mathbb{F}_{q^m}^n$ defined for any $\omega \in \Omega(D - E)$ as $\mathrm{res}_D(\omega) = (\mathrm{res}_{P_1}(\omega), \ldots, \mathrm{res}_{P_n}(\omega))$, where $\mathrm{res}_{P_i}(\omega)$ is the residue of ω at P_i, defines an AG code having minimum distance $d \geq \deg E - 2g(\mathcal{X}) + 2$ and dimension $k \leq n - \deg E + g(\mathcal{X}) - 1$. Whenever $\deg E \geq 2g(\mathcal{X}) - 1$ the equality $k = n - \deg E + g(\mathcal{X}) - 1$ holds. We refer to this code as the differential code $C_{\mathcal{X},\Omega}(D, E)$. Notably we have

$$C_{\mathcal{X},\Omega}(D, E) = C_{\mathcal{X},\mathcal{L}}(D, E)^{\perp} = C_{\mathcal{X},\mathcal{L}}(D, D - E + (\eta)), \tag{2}$$

where $C_{\mathcal{X},\mathcal{L}}(D, E)^{\perp}$ denotes the dual code of $C_{\mathcal{X},\mathcal{L}}(D, E)$ and η is a Weil differential with $v_{P_i}(\eta) = -1$ and $\mathrm{res}_{P_i}(\eta) = 1$ (see [55, Prop 2.2.10]).

For both evaluation and differential codes, $\mathrm{supp}(D)$ is also called the support of the code.

Let σ be a permutation of $\{1, \ldots, n\}$. For $\mathbf{v} \in \mathbb{F}_{q^m}^n$, define $\sigma\mathbf{v} = (v_{\sigma_1}, \ldots, v_{\sigma_n})$ and $\sigma C = \{\sigma\mathbf{c} \mid \mathbf{c} \in C\}$ for a linear code $C \subseteq \mathbb{F}_{q^m}^n$. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ be an n-tuple of non-zero elements of $\mathbb{F}_{q^m}$. Define $\boldsymbol{\beta}\mathbf{c} = (\beta_1 c_1, \ldots, \beta_n c_n)$ and $\boldsymbol{\beta}C = \{\beta\mathbf{c} \mid \mathbf{c} \in C\}$. Codes $C_1, C_2 \subseteq \mathbb{F}_{q^m}^n$ are said to be isometric if there exist $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n) \in (\mathbb{F}_{q^m}^*)^n$ and a permutation σ of $\{1, \ldots, n\}$ such that $C_2 = \boldsymbol{\beta}\sigma C_1$.

Let $C \subseteq \mathbb{F}_{q^m}^n$ be a code of dimension k and minimum distance d. Then $C|_{\mathbb{F}_q} := C \cap \mathbb{F}_q^n$ is called the subfield subcode of C. We have trivial bounds on the dimension k' and minimum distance d' of $C|_{\mathbb{F}_q}$, namely $k' \leq k$ and $d' \geq d$.

3 The family of codes studied in this paper

The differential codes we describe in this section are defined by choosing two rational functions $h, \gamma \in \mathbb{F}_{q^m}(\mathcal{X})$, where $\mathcal{X}$ is defined over $\mathbb{F}_{q^m}$. The divisor $(h)^+$ is used to describe the support of the code while the divisor $(\gamma)^+$ describes the space of differential forms defining the code. We give proofs to keep the paper self-contained and to explain the construction. Readers familiar with classical binary Goppa codes should recognize the similarities.

3.1. Definition of the code. Let $h, \gamma \in \mathbb{F}_{q^m}(\mathcal{X})$ be rational functions such that $\mathrm{supp}(H^+) = \{P_1, \ldots, P_n\} \subseteq \mathcal{P}_{\mathcal{X}}$, where $H := (h)$ and $v_P(h) \deg P = 1$ for every $P \in \mathrm{supp}(H^+)$, and $\mathrm{supp}(G) \cap \mathrm{supp}(H^+) = \emptyset$, where $G := (\gamma)$. Since H is a principal divisor, $\deg H^- = n$. Let $t := \deg G^-$. Let $M > 0$ be an effective divisor such that $M \leq G^-$ and $2g(\mathcal{X}) - 1 \leq t - \mu$ where $\mu := \deg M$.

We consider the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$, i.e., a differential AG code with minimum distance $d \geq t - \mu - 2g(\mathcal{X}) + 2$ and dimension $k \leq n - t + \mu + g(\mathcal{X}) - 1$. For applications to our cryptosystem we will consider the subfield subcode

$\Gamma_{\mathcal{X}}(h,\gamma,M) := C_{\mathcal{X},\Omega}(H^+, G^+ - M)|_{\mathbb{F}_q}$, which has minimum distance $d' \geq d$ and dimension $k' \geq n - m(t - \mu - g(\mathcal{X}) + 1)$ from Delsarte's theorem [55, Theorem 9.1.2]. In this section, we mainly focus on the differential code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ instead of its subfield subcode, since the latter inherits its properties from the former. We consider subcodes of $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ in Section 3.5.

Example 3.1. This example shows how to set up a classical Goppa code $\Gamma_{\mathbb{P}^1}(h,\gamma)$ using the notation in this section. Set $\mathcal{X} = \mathbb{P}^1$, so that $g(\mathcal{X}) = 0$, and take $h := x^{q^m} - x \in \mathbb{F}_{q^m}(\mathcal{X})$ for $m \geq 2$. We have $H = (h) = P_1 + \cdots + P_n - nP_\infty$, where $n = q^m$. Let $\gamma \in \mathbb{F}_{q^m}(\mathcal{X})$ be a squarefree polynomial in x such that $\operatorname{supp}(G) \cap \operatorname{supp}(H^+) = \emptyset$, where $G = (\gamma)$, and $1 < t < n$, where $t := \deg G^-$. Set $M := P_\infty$, so that $0 \leq M \leq G^-$. We have $\mu = 1$ and $t - \mu \geq 2g(\mathcal{X}) - 1$. Then $\Gamma_{\mathbb{P}^1}(h,\gamma) = C_{\mathcal{X},\Omega}(H^+, G^+ - M)|_{\mathbb{F}_q}$. This is a linear code with minimum distance $d' \geq t - \mu - 2g(\mathcal{X}) + 2 = t + 1$ and dimension $k' \geq n - m(t - \mu - g(\mathcal{X}) + 1) = n - mt$.

3.2. Evaluation view. In this subsection, we are going to show that the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ is isometric to a functional AG code of the form $C_{\mathcal{X},\mathcal{L}}(H^+, E)$; see Section 2.2 for the definition of isometric codes. This will be crucial for the specification of our scheme as we will employ a decoder for the evaluation code $C_{\mathcal{X},\mathcal{L}}(H^+, E)$ in order to decode $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$. See Section 5 and Section 6 for further discussion.

Theorem 3.1. *Let $\mathcal{X}$ be a smooth, projective, absolutely irreducible curve of genus $g(\mathcal{X})$ defined over $\mathbb{F}_{q^m}$, $\eta \in \Omega_{\mathcal{X}}$. Fix $h, \gamma \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $\operatorname{supp}(H^+) = \{P_1, \ldots, P_n\} \subseteq \mathcal{P}_{\mathcal{X}} \setminus \{P_\infty\}$, where $H := (h)$ and $v_{P_i}(h) \deg P_i = 1$ for every $i = 1, \ldots, n$, $v_{P_\infty}(\gamma) \leq 0$ and $\operatorname{supp}(G) \cap \operatorname{supp}(H^+) = \operatorname{supp}((\eta)) \cap \operatorname{supp}(H^+) = \emptyset$, where $G := (\gamma)$. Let $M \geq 0$ be an effective divisor such that $M \leq G^-$ and set $\mu := \deg M$ and $t := \deg G^-$. Let $\boldsymbol{\beta} = \left(\gamma/\hat{h}(P_1), \ldots, \gamma/\hat{h}(P_n)\right)$ where $\hat{h} = dh/\eta$. Then we have the isometry $C_{\mathcal{X},\Omega}(H^+, G^+ - M) = \boldsymbol{\beta} C_{\mathcal{X},\mathcal{L}}(H^+, E)$ where $E = H^- - G^- + M + (\eta)$.*

Proof. Since $v_{P_i}(h) = 1$, the differential $\omega = dh/h$ has a simple pole at P_i with $\operatorname{res}_{P_i} = 1$ for every $i = 1, \ldots, n$. Let $W = (\omega)$ be the canonical divisor associated with ω, then $W = (dh) - H^+ + H^-$.

Since the vector space $\Omega(\mathcal{X})$ is one-dimensional over $\mathbb{F}_{q^m}(\mathcal{X})$, there exists $\hat{h} \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $dh = \hat{h}\eta$. Using (2), we have $C_{\mathcal{X},\Omega}(H^+, G^+ - M) = C_{\mathcal{X},\mathcal{L}}(H^+, W + H^+ - G^+ + M)$. Bearing in mind that $G^+ = G + G^-$, a rational function $f \in \mathcal{L}(W + H^+ - G^+ + M)$ is such that $(f) \geq -W - H^+ + G^+ - M = -(dh) + H^+ - H^- - H^+ + G^+ - M = -(\hat{h}) - (\eta) - H^- + G + G^- - M$. Therefore f is of the form $\hat{f} \cdot \gamma/\hat{h}$ with $\hat{f} \in \mathcal{L}(E)$ with $E = H^- - G^- + M + (\eta)$ and $\deg E = n - t + \mu + 2g(\mathcal{X}) - 2$. □

Remark 3.2. Observe that the existence of $\eta \in \Omega_{\mathcal{X}}$ with $\operatorname{supp}((\eta)) \cap \operatorname{supp}(H^+) = \emptyset$ is guaranteed by the weak approximation theorem [54, Theorem 1.3.1]. For the case of $C_{a,b}$ curves we can, and will in our examples, use $\eta = dx$ (having $(\eta) = (2g(\mathcal{X}) - 2)P_\infty$).

A sanity check on the parameters of the code $C_{\mathcal{X},\mathcal{L}}(H^+, E)$ supports the correctness of the construction, indeed the minimum distance of this functional code is $d \geq n - \deg E = t - \mu - 2g(\mathcal{X}) + 2$ while the dimension is given by $k \geq \deg E - g(\mathcal{X}) + 1 = n - t + \mu + g(\mathcal{X}) - 1$. The reader can compare these parameters with the parameters of the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$.

3.3. Generator and parity check matrices. In this section, we give explicit generator and parity check matrices for the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$.

3.3.1. Generator matrix. The standard generator matrix for these codes, and in general for all differential AG codes, is given by taking the residues at $\{P_1, \ldots, P_n\}$ of a basis $\{\omega_1, \ldots, \omega_k\}$ of the differential space $\Omega(G^+ + M - H^+)$ as

$$\mathbf{G} = \begin{pmatrix} \operatorname{res}_{P_1}(\omega_1) & \cdots & \operatorname{res}_{P_n}(\omega_1) \\ \vdots & \ddots & \vdots \\ \operatorname{res}_{P_1}(\omega_k) & \cdots & \operatorname{res}_{P_n}(\omega_k) \end{pmatrix}. \tag{3}$$

From the isometry given in Theorem 3.1 we obtain a different description of a generator matrix.

Proposition 3.3. *With notation as in Theorem 3.1, a generator matrix of the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ is*

$$\mathbf{G} = \begin{pmatrix} \frac{\gamma\varphi_1}{\hat{h}}(P_1) & \cdots & \frac{\gamma\varphi_1}{\hat{h}}(P_n) \\ \vdots & \ddots & \vdots \\ \frac{\gamma\varphi_k}{\hat{h}}(P_1) & \cdots & \frac{\gamma\varphi_k}{\hat{h}}(P_n) \end{pmatrix}, \tag{4}$$

where $\{\varphi_1, \ldots, \varphi_k\}$ is a basis of the Riemann-Roch space $\mathcal{L}(E)$, with $E = H^- - G^- + M + (\eta)$.

Proof. The claim follows easily from Theorem 3.1. Indeed, $C_{\mathcal{X},\Omega}(H^+, G^+ - M) = \boldsymbol{\beta} C_{\mathcal{X},\mathcal{L}}(H^+, E)$ with $\boldsymbol{\beta} = \left(\gamma/\hat{h}(P_1), \ldots, \gamma/\hat{h}(P_n)\right)$ where $\hat{h} = dh/\eta$, and a generator matrix of the code $C_{\mathcal{X},\mathcal{L}}(H^+, E)$ is given by

$$\begin{pmatrix} \varphi_1(P_1) & \cdots & \varphi_1(P_n) \\ \vdots & \ddots & \vdots \\ \varphi_k(P_1) & \cdots & \varphi_k(P_n) \end{pmatrix}. \qquad \square$$

3.3.2. Parity check matrix. We construct an explicit parity check matrix of $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ by constructing a generator matrix of the dual code $C_{\mathcal{X},\mathcal{L}}(H^+, G^+ - M)$. To this end, we construct a basis of the Riemann-Roch space $\mathcal{L}(G^+ - M)$. We first need some preliminary results.

Lemma 3.4. *With notation as in Theorem 3.1, the dimension of the Riemann-Roch space $\mathcal{L}(G^- - M)$ is $n-k$, where k is the dimension of $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$.*

Proof. $C_{\mathcal{X},\Omega}(H^+, G^+ - M)^\perp = C_{\mathcal{X},\mathcal{L}}(H^+, G^+ - M) = \gamma C_{\mathcal{X},\mathcal{L}}(H^+, G^- - M)$, hence $\dim \mathcal{L}(G^- - M) = \dim \gamma C_{\mathcal{X},\mathcal{L}}(H^+, G^- - M) = n - k$. $\square$

The next proposition gives an explicit parity check matrix of the code.

Proposition 3.5. *With the same notation as Theorem 3.1, a parity check matrix of the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ is*

$$\mathbf{H} = \begin{pmatrix} \frac{\varphi_1}{\gamma}(P_1) & \cdots & \frac{\varphi_1}{\gamma}(P_n) \\ \vdots & \ddots & \vdots \\ \frac{\varphi_{n-k}}{\gamma}(P_1) & \cdots & \frac{\varphi_{n-k}}{\gamma}(P_n) \end{pmatrix}, \tag{5}$$

where $\{\varphi_1, \ldots, \varphi_{n-k}\}$ is a basis of the Riemann-Roch space $\mathcal{L}(G^- - M)$.

Proof. We claim that a basis of the Riemann-Roch space $\mathcal{L}(G^+ - M)$ is given by φ_i/γ for $i = 1, \ldots, n-k$, where $\{\varphi_1, \ldots, \varphi_{n-k}\}$ is a basis of the Riemann-Roch space $\mathcal{L}(G^- - M)$. From Lemma 3.4 we obtain the size $n-k$. Fix i and consider the divisor $(\varphi_i/\gamma) = (\varphi_i) - G$. Since $\varphi_i \in \mathcal{L}(G^- - M)$ we have $(\varphi_i/\gamma) = (\varphi_i) - G \geq -G^- + M - G^+ + G^- = -G^+ + M$, meaning that $\varphi_i/\gamma \in \mathcal{L}(G^+ - M)$. This holds for every i. The fact that $\varphi_1, \ldots, \varphi_{n-k}$ form a basis of $\mathcal{L}(G^- - M)$ implies that such elements are linearly independent over $\mathbb{F}_{q^m}$. The dimension of the code $C_{\mathcal{X},\mathcal{L}}(H^+, G^+ - M) = C_{\mathcal{X},\Omega}(H^+, G^+ - M)^\perp$ is $n-k$, proving the claim. To end the proof, just note that $\mathbf{H}$ is a generator matrix of the code $C_{\mathcal{X},\mathcal{L}}(H^+, G^+ - M)$. □

Proposition 3.5 assumes the knowledge of a basis of the Riemann-Roch space $\mathcal{L}(G^- - M)$ in advance. When $\mathcal{X}$ is a $C_{a,b}$ curve, for a particular shape of the divisor M, the following corollary gives an explicit basis of the space $\mathcal{L}(G^- - M)$.

Corollary 3.6. *Let $\mathcal{X}$ be a $C_{a,b}$ curve over $\mathbb{F}_{q^m}$ and let $M = G^- + v_{P_\infty}(\gamma)P_\infty$. Then a parity check matrix for $C_{\mathcal{X},\mathcal{L}}(H^+, G^+ - M)$ is given as in (5), where a basis of the space $\mathcal{L}(G^- - M)$ is*

$$\{\varphi_1, \ldots, \varphi_{n-k}\} = \{x^{\ell_x} y^{\ell_y} \mid a\ell_x + b\ell_y \leq t - \mu, 0 \leq \ell_x, 0 \leq \ell_y < a\},$$

where $t := \deg G^-$ and $\mu := \deg M$.

Proof. We have $G^- - M = -v_{P_\infty}(\gamma)P_\infty \geq 0$. The basis is the standard basis of $\mathcal{L}((t-\mu)P_\infty)$ for $C_{a,b}$ curves. See, e.g., [8, Section II.B, Equation (II.1)]. □

Example 3.2. We describe a parity check matrix of the classical Goppa code using our tools. Recall that the dimension of the code $C_{\mathcal{X},\Omega}(H^+, G^+ - P_\infty)$ is $k = n - t$. For classical Goppa codes one chooses $\gamma \in \mathbb{F}_{q^m}[x] \subset \mathbb{F}_{q^m}(\mathbb{P}^1)$ of degree t. From Lemma 3.4 we have that $\mathcal{L}(G^- - P_\infty)$ has dimension $n - k = t$, hence it is the space of polynomials in $\mathbb{F}_{q^m}[x]$ with degree at most $t-1$, and $\{\varphi_1 = 1, \varphi_2 = x, \ldots, \varphi_t = x^{t-1}\}$ is a basis over $\mathbb{F}_{q^m}$ of $\mathcal{L}(G^- - P_\infty)$. Identifying $\{P_1, \ldots, P_n\}$ with $\{\alpha_1, \ldots, \alpha_n\}$, from Proposition 3.5 we obtain that a parity check matrix of $C_{\mathcal{X},\Omega}(H^+, G^+ - P_\infty)$ is given by

$$\mathbf{H} = \begin{pmatrix} 1/\gamma(\alpha_1) & \cdots & 1/\gamma(\alpha_n) \\ \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/\gamma(\alpha_1) & \cdots & \alpha_n^{t-1}/\gamma(\alpha_n) \end{pmatrix}, \tag{6}$$

which is a well known parity check matrix for the classical Goppa code, see for example [14].

3.4. Algorithmic constraints. As mentioned in the introduction of this paper, attacks such as [20] against McEliece variants using AG codes do not affect McEliece variants using subfield subcodes of AG codes. This motivates the choice of the code $\Gamma_{\mathcal{X}}(h, \gamma, M)$. Section 3.5 recalls some known results about subfield subcodes. We need to point out that there are intrinsic and algorithmic constraints on the choice of the rational functions h and γ and on the choice of the divisor M. To start with, the definition of $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ puts some constraints on the zeros and poles of the rational functions h, γ, namely $\operatorname{supp}((\gamma)) \cap \operatorname{supp}((h)^+) = \emptyset$ and $v_P(h) \deg P = 1$ for each $P \in \operatorname{supp}((h)^+)$. The cryptosystem described in Section 6 generates keys by choosing these two rational functions randomly within the sets satisfying the mentioned properties. Section 4 describes efficient methods to achieve this.

Furthermore, the AG code-based scheme we describe in Section 6 uses the decoder given in Section 5, which works with evaluation one-point AG codes. This translates to requiring that the rational function h can have poles only at P_∞, in other words, that $h \in R_{\mathcal{X}}$, the coordinate ring of $\mathcal{X}$. For the codes used in Section 6 arising from $C_{a,b}$ curves we can choose $\eta = dx \in \Omega_{\mathcal{X}}$, whose divisor is $(2g(\mathcal{X})-2)P_\infty$. It is interesting to observe that, in general, a differential having divisor whose support is restricted by some conditions on the places can be constructed as, for instance, in [53, Proposition 4.12]. Characterizations of curves admitting differentials whose divisor has support confined at a single place can be found in [24, 51].

Corollary 3.7 (to Theorem 3.1) says that, when h satisfies this property, the differential code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ is isometric to a one-point evaluation code.

Corollary 3.7. *With the same notation as Theorem 3.1, assume that* $(h)^- = nP_\infty$ *for some* $n \in \mathbb{N}$ *and set* $M = G^- + v_{P_\infty}(\gamma)P_\infty$. *Then* $C_{\mathcal{X},\Omega}(H^+, G^+ - M) = \boldsymbol{\beta} C_{\mathcal{X},\mathcal{L}}(H^+, (n - t + \mu + 2g(\mathcal{X}) - 2)P_\infty)$.

In Section 4.2 we describe how to build rational functions having simple zeros within a selected set of places of degree 1 and poles only at P_∞.

3.5. Subfield subcodes. Let C be an $[n, n - r, d]$-code over $\mathbb{F}_{q^m}$, and consider its subfield subcode $C|_{\mathbb{F}_q}$. We already mentioned the lower bound on the dimension arising from Delsarte's theorem, i.e., $C|_{\mathbb{F}_q}$ is an $[n, \geq n - mr, \geq d]$-code over $\mathbb{F}_{q^m}$. If we consider $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ then the subfield subcode $\Gamma_{\mathcal{X}}(h, \gamma, M)$ is an $[n, \geq n - m(t - \mu - g(\mathcal{X}) + 1), \geq t - \mu - 2g(\mathcal{X}) + 2]$-code over $\mathbb{F}_q$.

Stichtenoth [54, Theorem 4] proved a better bound on the dimension.

Theorem 3.8. *Let* G *and* M *be such that* $G^+ - M \geq qG_1$ *for some divisor* G_1 *(not necessarily effective), then* $\dim \Gamma_{\mathcal{X}}(h, \gamma, M) \geq n - m(t - \mu - g(\mathcal{X}) + 1 - \ell(G_1))$.

Corollary 3.9. $\dim \Gamma_{\mathcal{X}}(h, \gamma, M) \geq n - m(t - \mu - \deg G_1)$ *if* $\deg G_1 \geq 2g(\mathcal{X}) - 1$.

Wirtz [59, Theorem 2] proved the following equality, which gives better lower bounds on dimension and minimum distance for $G^+ - M$ of a particular shape.

Theorem 3.10. *Let G_1 be as in Theorem 3.8. Assume that $\deg G_1 \geq 2g(\mathcal{X}) - 2$ and $G_1 \geq 0$. Define divisor G_U as the sum of the places P such that $v_P(G^+ - M) \equiv q - 1 \mod q$. Then $\Gamma_{\mathcal{X}}(h, \gamma, M) = C_{\mathcal{X},\Omega}(H^+, G^+ + G_U - M)|_{\mathbb{F}_q}$.*

Theorem 3.10 gives a condition on choosing γ and M in order to have sharper lower bounds than those on the dimension and the minimum distance of the code $\Gamma_{\mathcal{X}}(h, \gamma, M)$. The strategy we propose to construct γ in Section 4 does not allow a very precise handling of the coefficients of the divisor of γ, in other words, the shape of divisor of the generated function γ is not predictable, making the use of the bound in Theorem 3.10 an unsafe estimate.

Moreover, for our applications to cryptography, we propose $M \leq G^-$ having a specific shape, which results in an efficient way to compute the parity check matrix, so the condition would fall only on the choice of γ. To construct γ, we will consider γ_1 such that $(\gamma_1)^+ = \sum_{P \in \mathrm{supp}((\gamma_1)^+)} P$, i.e. every coefficient of the divisor is 1. Finally, we use the code $\Gamma_{\mathcal{X}}(h, \gamma_1^q, qM)$, i.e. $G^+ = (\gamma_1^q)^+$, meaning that $G_U = 0$, and use the lower bound in Theorem 3.8 for the dimension.

For the case of classical binary Goppa codes, $\mathcal{X}$ is the projective line; taking each coefficient of $(\gamma_1)^+$ as 1 corresponds to taking a squarefree Goppa polynomial; and taking $G^+ = (\gamma_1^2)^+$ corresponds to taking the square of that polynomial, which, as Goppa [32, Section 4] showed, defines the same code.

4 Efficient construction of rational functions

From now on we let $\mathcal{X}$ be a $C_{a,b}$ curve; see Section 2.1.3 for the definition and basic properties of these curves. This section provides efficient methods to generate rational functions h and γ, and thus to generate a code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$, subject to constraints specified below. Section 4.1 is dedicated to an algorithm that we mainly use to construct γ. Section 4.2 shows how to build a rational function $h \in R_{\mathcal{X}}$ with simple zeros of degree 1.

4.1. Construction of rational functions with zeros of bounded multiplicity in a predetermined set of places. Let $\mathcal{P} \subset \mathcal{P}_{\mathcal{X}} \setminus \{P_\infty\}$ be a subset of the places of $\mathcal{X}$, $\{e_P \in \mathbb{N} | P \in \mathcal{P}\}$ a set of positive integers and consider the divisor $E = \sum_{P \in \mathcal{P}} e_P P \geq 0$ having degree $e := \deg E = \sum_{P \in \mathcal{P}} e_P \deg P$. In this section, we show how to construct a rational function $f \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $(f)^+ \leq E$ and $\deg(f)^+ \geq e - g(\mathcal{X})$.

Let $\mathrm{Div}^0(\mathcal{X})$ be the subgroup of $\mathrm{Div}(\mathcal{X})$ of divisors of degree 0. The set $\mathrm{Princ}(\mathcal{X})$ of principal divisors is a subgroup of $\mathrm{Div}^0(\mathcal{X})$. The following definition of reduced divisors is standard; see, e.g., [39]. The definition relies on $\mathcal{X}$ to have a unique point at infinity, as $C_{a,b}$ curves do.

Definition 4.1. *A divisor $D \in \mathrm{Div}^0(\mathcal{X})$ of the form $D = A - aP_\infty$, with $A \geq 0$, is called a semi-reduced divisor. If a is minimal for all D' in the equivalence class of D in $\mathrm{Div}^0(\mathcal{X})/\mathrm{Princ}(\mathcal{X})$, then D is called a reduced divisor.*

For elements of $\mathrm{Div}^0(\mathcal{X})$ we have the following well known theorem, for which we give a proof because the technique is relevant for the next steps.

Theorem 4.2. *Fix $D \in \mathrm{Div}^0(\mathcal{X})$. (1) There exists $r \in \mathbb{N}$ with $\ell(D + rP_\infty) > 0$. (2) Define $a = \min\{r \in \mathbb{N} : \ell(D + rP_\infty) > 0\}$. Then $a \leq g(\mathcal{X})$. (3) There is a unique divisor A such that $A \geq 0$, $\deg A = a$, and D is equivalent to $A - aP_\infty$.*

Proof. Let W be a canonical divisor, By the Riemann-Roch theorem, for each $r \in \mathbb{N}$, we have $\ell(D + rP_\infty) = \ell(W - D - rP_\infty) + \deg D + r - g(\mathcal{X}) + 1 \geq \deg D + r - g(\mathcal{X}) + 1$. In particular, $\ell(D + g(\mathcal{X})P_\infty) \geq \deg D + 1 > 0$. Hence $\{r \in \mathbb{N} : \ell(D + rP_\infty) > 0\}$ contains $g(\mathcal{X})$.

In particular, $a \leq g(\mathcal{X})$. Also, since $\ell(D + aP_\infty) > 0$, there exists a nonzero $f \in \mathcal{L}(D + aP_\infty)$. Define $A = D + aP_\infty + (f)$. Then $A \geq 0$ by definition of $\mathcal{L}$; $\deg A = a$ since D and (f) have degree 0; and D is equivalent to $A - aP_\infty$.

What remains is to prove uniqueness: i.e., to prove that if an effective degree-a divisor B has $B - aP_\infty$ equivalent to D then $B = A$.

This is easy if $a = 0$ (i.e., if D is principal): the only effective degree-0 divisor is 0. Assume from now on that $a > 0$. Then $\ell(D + (a-1)P_\infty) = 0$ by definition of a.

Note that any $D_1, D_2 \in \mathrm{Div}(\mathcal{X})$ with $D_1 \leq D_2$ have $\ell(D_1) \leq \ell(D_2)$. In particular,

$$\begin{aligned}\ell(D + aP_\infty) &= \ell(W - D - aP_\infty) + \deg D + a - g(\mathcal{X}) + 1 \\ &\leq \ell(W - D - (a-1)P_\infty) + \deg D + a - g(\mathcal{X}) + 1 \\ &= \ell(D + (a-1)P_\infty) + 1 = 1.\end{aligned}$$

By construction $\ell(D + aP_\infty) > 0$, so $\ell(D + aP_\infty) = 1$, so every element of $\mathcal{L}(D + aP_\infty)$ is an $\mathbb{F}_{q^m}$-multiple of f.

We have $B = D + aP_\infty + (g)$ for some nonzero g. Also $g \in \mathcal{L}(D + aP_\infty)$ since $B \geq 0$. Hence g is a nonzero $\mathbb{F}_{q^m}$-multiple of f, forcing $(g) = (f)$, so $B = A$ as claimed. □

Remark 4.3. The divisor $A - aP_\infty$ constructed in Theorem 4.2 is reduced.

Given a set of places $\mathcal{P}$ forming E, the strategy is to consider the semi-reduced divisor $D = E - eP_\infty$ and compute the unique reduced divisor D' equivalent to D. In particular, we construct $f \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $D = D' + (f)$ and show that $(f)^+ \leq E$ and $\deg(f)^+ \geq e - g(\mathcal{X})$. The following algorithm is a slight adaptation of [4, Algorithm 1] which finds f starting from a set of places and a set of positive integers. In the following we define, for a divisor $D \in \mathrm{Div}(\mathcal{X})$ the space $\mathcal{L}(D + \infty P_\infty) = \bigcup_{m=0}^{\infty} \mathcal{L}(D + mP_\infty)$. In particular, $\mathcal{L}(\infty P_\infty) = R_{\mathcal{X}}$.

Algorithm 1 SlowBuildRatFunc

Input : A set of places $\mathcal{P} = \{P_1, \ldots, P_r\} \subset \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$ of $\mathcal{X}$ and a set of positive integers $\{e_1, \ldots, e_r\} \subseteq \mathbb{N}$.
Output : A rational function $f \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $(f)^+ \leq \sum_{i=1}^r e_i P_i$ and $\deg(f)^+ \geq \sum_{i=1}^r e_i - g(\mathcal{X})$.

1. Put $E = \sum_{i=1}^r e_i P_i$.
2. Find $f_1 \in \mathcal{L}(-E + \infty P_\infty)$ s.t. $-v_{P_\infty}(f_1)$ is minimal.
3. Put $E' = -E + (f_1)^+$.
4. Find $f_2 \in \mathcal{L}(-E' + \infty P_\infty)$ s.t. $-v_{P_\infty}(f_2)$ is minimal.
5. Output : f_1/f_2.

We prove that Algorithm 1 produces the sought function $f \in \mathbb{F}_{q^m}(\mathcal{X})$.

Proposition 4.4. *Let $\mathcal{P} = \{P_1, \ldots, P_r\} \subset \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$, $\{e_P \in \mathbb{N} | P \in \mathcal{P}\}$ a set of positive integers. Put $E = \sum_{i=1}^r e_i P_i$ and $D = E - eP_\infty$, where $e = \sum_{i=1}^r e_i$. Then Algorithm 1 outputs a function $f \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $D - (f)$ is a reduced divisor. In particular, $(f)^+ \leq E$ and $\deg(f)^+ \geq e - g(\mathcal{X})$.*

Proof. First off, we prove that $D - (f_1/f_2)$ is a reduced divisor. Using the same arguments as in the proof of Theorem 4.2 we can prove that there exists $f_1 \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $-D + (f_1) = D'$ with $D' = E' - e'P_\infty$ a reduced divisor, hence $f_1 \in \mathcal{L}(-E + \infty P_\infty)$ with $e' \leq g(\mathcal{X})$ minimal. Similarly, we can find $f_2 \in \mathcal{L}(-E' + \infty P_\infty)$ such that $-(-D + (f_1)) + (f_2) = E'' - e''P_\infty$ is a reduced divisor. Therefore $D = E'' - e''P_\infty + (f_1/f_2)$.

To show that $(f_1/f_2)^+ \leq E$ note that $(f_1) = E + E' - (e + e')P_\infty$ and $(f_2) = E' + E'' - (e' + e'')P_\infty$. Therefore $(f_1/f_2) = E - E'' - (e - e'')P_\infty$, and since $E'' \geq 0$ it follows that $(f_1/f_2)^+ \leq E$. Finally, since $\deg E'' = e'' \leq g(\mathcal{X})$ then $\deg(f_1/f_2)^+ \geq e - g(\mathcal{X})$. Hence, put $f = f_1/f_2$. □

Typical computations involving divisors involve computing bases of Riemann-Roch spaces, which can be quite expensive in terms of computational complexity. This means that steps (2) and (4) of Algorithm 1 are, in general, not efficient. We thus follow the strategy in [4, 29] and adopt the ideal representation of divisors. To this end, put $S = \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$ and consider the holomorphy ring

$$\mathcal{O}_S = \bigcap_{P \in S} \mathcal{O}_P$$

of $\mathbb{F}_{q^m}(\mathcal{X})$; see [55, Definition 3.2.2] for the definition of holomorphy ring. For $\mathcal{X}$ being a $C_{a,b}$ curve, we have that $\mathcal{O}_S = R_\mathcal{X}$, where $R_\mathcal{X} = \mathbb{F}_{q^m}[x, y]/\langle \mathcal{X} \rangle$ is the coordinate ring of $\mathcal{X}$. Note that $R_\mathcal{X}$ is a Dedekind domain; see [43]. Also, $R_\mathcal{X}$ is the integral closure of $\mathbb{F}_{q^m}[x]$ in the extension $\mathbb{F}_{q^m}(\mathcal{X})/\mathbb{F}_{q^m}(x)$.

Proposition 4.5 ([55, Proposition 3.2.9]). *Let $\mathcal{O}_S$ be a holomorphy ring of $\mathbb{F}_{q^m}(\mathcal{X})$. Then there is a 1-1 correspondence between S and the set of maximal ideals of $\mathcal{O}_S$, given by $P \mapsto \mathfrak{p} := P \cap \mathcal{O}_S$. Moreover, the map $\mathcal{O}_S/\mathfrak{p} \to \mathcal{O}_P/P$ given by $f + \mathfrak{p} \mapsto f + P$ is an isomorphism.*

Remark 4.6. Note that the isomorphism in Proposition 4.5 allows translating the notion of degree of a place to that of degree of the corresponding ideal. Indeed, $\deg P = [\mathcal{O}_P/P : \mathbb{F}_{q^m}] = [\mathcal{O}_S/\mathfrak{p} : \mathbb{F}_{q^m}] = \dim_{\mathbb{F}_{q^m}} R_{\mathcal{X}}/\mathfrak{p}$, where the rightmost term denotes the dimension over $\mathbb{F}_{q^m}$ of the module $R_{\mathcal{X}}/\mathfrak{p}$.

We can identify divisors on $\mathcal{X}$ with ideals of $R_{\mathcal{X}}$ via the following map. Let $\mathrm{Id}(R_{\mathcal{X}})$ be the group of fractional ideals of $R_{\mathcal{X}}$ and $\mathfrak{S} = \{P \cap \mathcal{O}_S \mid P \in S\}$. Define the map

$$\begin{aligned} \phi : \mathrm{Div}(\mathcal{X}) &\to \mathrm{Id}(R_{\mathcal{X}}) \\ \sum_{P \in \mathcal{P}_{\mathcal{X}}} n_P P &\mapsto \prod_{\mathfrak{p} \in \mathfrak{S}} \mathfrak{p}^{n_P}. \end{aligned}$$

Note that the divisors that are mapped to integral ideals of $R_{\mathcal{X}}$ are those of the form $A - aP_\infty$ for some $A \geq 0$ and $a \in \mathbb{N}$, in particular, the semi-reduced divisors. Indeed, if $A - aP_\infty$ is a semi-reduced ideal with $A = \sum_{P \in S} a_P P$ then clearly $\phi(A - aP_\infty) = \prod_{\mathfrak{p} \in \mathfrak{S}} \mathfrak{p}^{a_P}$.

Remark 4.7. Note that the map ϕ "forgets" about P_∞. Indeed, for a semi-reduced divisor $D = E - eP_\infty$, we have that $\phi(D) = \phi(E)$. For our goal, we can thus forget about the semi-reduced structure of a divisor when working with the corresponding ideals and work with the effective divisor E alone.

Proposition 4.8. *Let $E \in \mathrm{Div}(\mathcal{X})$ be an effective divisor on $\mathcal{X}$. Then $\phi(E) = \mathcal{L}(-E + \infty P_\infty)$.*

Proof. Both $\phi(E)$ and $\mathcal{L}(-E+\infty P_\infty)$ contain 0, so we focus on nonzero elements.

Write $E = \sum_{P \in S} e_P P$. Let $f \in \phi(E)$, we factor the principal ideal $\langle f \rangle \subset R_{\mathcal{X}}$ as $\langle f \rangle = \prod_{\mathfrak{p} \in \mathfrak{S}} \mathfrak{p}^{e'_P}$, where $e'_P \geq e_P$ for every $P \in S$. This means that $f \in P^{e'_P} \cap \mathcal{O}_S$, i.e., $v_P(f) \geq e_P$ for every $P \in S$. Moreover, f has poles only at P_∞, hence $(f) \geq E - \infty P_\infty$; in other words $f \in \mathcal{L}(-E + \infty P_\infty)$.

On the other hand, if $f \in \mathcal{L}(-E+\infty P_\infty)$ then $(f) \geq E-\infty P_\infty$, i.e., $f \in P^{e_P}$ for each $P \in S$. Also, $f \in \mathcal{O}_S$. Therefore, $f \in P^{e_P} \cap \mathcal{O}_{\mathcal{S}}$, meaning that $\mathfrak{p}^{e_P} | \langle f \rangle$ for each $\mathfrak{p} \in \mathfrak{S}$. Hence, $\phi(E) | \langle f \rangle$, i.e., $f \in \phi(E)$. □

With these tools, we can replace steps (2) and (4) of Algorithm 1 with computations involving elements in $\mathrm{Id}(R_{\mathcal{X}})$. For instance, given a semi-reduced divisor $D = E - eP_\infty$, we can find f_1 in step (2) by searching the smallest element in $\phi(D)$ in terms of the pole order at P_∞. In particular, finding a minimal element in $\phi(D)$ w.r.t. the pole order at P_∞ is equivalent to finding a minimal element in $\mathcal{L}(-E + \infty P_\infty)$.

We will now describe an efficient procedure to compute such a minimal element. The strategy we outline is similar to that in [29, 33]. We first represent a divisor by a matrix over $\mathbb{F}_{q^m}[x]$ generating an $\mathbb{F}_{q^m}[x]^a$-submodule and then compute the (weak) Popov form of this matrix. This corresponds to lattice basis reduction over $\mathbb{F}_{q^m}[x]$. We refer the reader to [9, Section 2.3] for an introduction to Popov forms of polynomial matrices. The (shifted) Popov form of such a matrix will contain the smallest element of the module as one of its rows.

In more detail: The ring $R_{\mathcal{X}}$ has an integral $\mathbb{F}_{q^m}[x]$-basis $\{1, y, \ldots, y^{a-1}\}$. We embed $R_{\mathcal{X}}$ into $\mathbb{F}_{q^m}[x]^a$ as coefficient vectors on this basis:

$$\psi : R_{\mathcal{X}} \to \mathbb{F}_{q^m}[x]^a \tag{7}$$

$$p(x,y) = \sum_{i=0}^{a-1} p_i(x) y^i \mapsto (p_0(x), p_1(x), \ldots, p_{a-1}(x)) \,. \tag{8}$$

This mapping ψ takes each integral ideal $\mathfrak{p}$ to a submodule $\psi(\mathfrak{p}) = \{\psi(r) : r \in \mathfrak{p}\}$ of $\mathbb{F}_{q^m}[x]^a$.

Since $R_{\mathcal{X}}$ is a Dedekind domain, each ideal $\mathfrak{p}$ can be written in the form $\langle \omega_1, \omega_2 \rangle$, and one can take ω_1 as any nonzero element of $\mathfrak{p}$ if $\mathfrak{p} \neq 0$; see, e.g., [18, Proposition 1.2.3]. For definiteness, take ω_1 as the unique generator of $\mathfrak{p} \cap \mathbb{F}_{q^m}[x]$ that is monic or zero.

The $\mathbb{F}_{q^m}[x]$-module $\psi(\mathfrak{p})$ is then generated by the following elements:

$$\psi(\omega_1), \psi(\omega_2), \psi(y\omega_1), \psi(y\omega_2), \ldots, \psi(y^{a-1}\omega_1), \psi(y^{a-1}\omega_2).$$

Computationally, it is useful to reduce each coefficient of each $y^i\omega_2$ modulo ω_1, taking advantage of $\omega_1 \in \mathbb{F}_{q^m}[x]$, to obtain the following list of generators:

$$\begin{aligned} &\psi(\omega_1), \psi(y\omega_1), \ldots, \psi(y^{a-1}\omega_1), \\ &\psi(\omega_2 \bmod \omega_1), \psi(y\omega_2 \bmod \omega_1), \ldots, \psi(y^{a-1}\omega_2 \bmod \omega_1). \end{aligned} \tag{9}$$

Remark 4.9. Note that the $2a$ elements of length a listed in (9) are not linearly independent. Indeed, they generate a module of rank at most a.

Let $E \geq 0$ be an effective divisor, take $\mathfrak{p} = \phi(E)$, and write $\mathbf{M}_E$ for the matrix in $\mathbb{F}_{q^m}[x]^{2a \times a}$ whose rows are the $2a$ elements listed in (9). As we mentioned, $\mathbf{M}_E$ generates a submodule $\psi(\phi(E))$ of $\mathbb{F}_{q^m}[x]^a$ of rank at most a. We aim at finding an element of the module generated by $\mathbf{M}_E$ that is minimal w.r.t. the valuation $-v_{P_\infty}$. We first need to define a norm on $\mathbb{F}_{q^m}[x]^a$ that is consistent with $-v_{P_\infty}$, i.e., define $|\cdot|\colon \mathbb{F}_{q^m}[x]^a \to \mathbb{Z}$ as

$$\mathbf{v} = (v_0(x), \ldots, v_{a-1}(x)) \mapsto \max_{i=0,\ldots,a-1} a \deg v_i + bi, \tag{10}$$

where deg is the usual degree of a polynomial in $\mathbb{F}_{q^m}[x]$. Thanks to Proposition 4.8, it is easy to see that a module element of $\psi(\phi(E))$ that is minimal w.r.t. $|\cdot|$ corresponds to a rational function in $\mathcal{L}(-E + \infty P_\infty)$ that is minimal w.r.t. $-v_{P_\infty}$.

There exist efficient algorithms [60] that compute the minimal element of $\psi(\phi(E))$ w.r.t. $|\cdot|$. Namely, these algorithms compute the (shifted) Popov form of a matrix in $\mathbb{F}_{q^m}^{2a \times a}$, in our case. These algorithms allow specifying a so-called *shift* which is an element $\mathbf{s} \in \mathbb{Z}^a$ meant to describe a norm on the space $\mathbb{F}_{q^m}[x]^a$. In particular, for our settings we can set

$$\mathbf{s} = (0, \lfloor b/a \rfloor, \lfloor 2b/a \rfloor, \ldots, \lfloor (a-1)b/a \rfloor).$$

A matrix in Popov form generates the same module as the input matrix, and it is guaranteed to contain the smallest module element among its rows w.r.t. the norm specified by the shift $\mathbf{s}$. For a matrix $\mathbf{M} \in \mathbb{F}_{q^m}[x]^{\kappa \times \nu}$ with $\nu \leq \kappa$ of rank r we denote by $\deg_x \mathbf{M}$ the maximal degree of the entries of $\mathbf{M}$. One can compute the Popov form of $\mathbf{M}$, hence a minimal element in the submodule of $\mathbb{F}_{q^m}[x]^\nu$ spanned by $\mathbf{M}$, in $\tilde{\mathcal{O}}(\kappa \nu^{\omega-1} \deg_x \mathbf{M})$ field operations [9, Corollary 2.11], where the notation $\tilde{\mathcal{O}}(\cdot)$ is similar to $\mathcal{O}(\cdot)$ but ignores logarithmic factors and ω is the linear algebra constant. Therefore, the minimal element in the module $\psi(\phi(E))$ w.r.t. to the norm $|\cdot|$ can be found computing the $\mathbf{s}$-shifted Popov form of $\mathbf{M}_E$. This can be done in $\tilde{\mathcal{O}}(a^\omega \deg_x \mathbf{M}_E)$ field operations.

The next proposition gives a bound d on the degree $\deg_x \mathbf{M}_E$ of the entries of $\mathbf{M}_E$. We refer to [55] for concepts used in the proof such as constant field extension.

Theorem 4.10. *Let $E \geq 0$ be an effective divisor on $\mathcal{X}$. Then*

$$\deg_x \mathbf{M}_E \leq \deg E.$$

Proof. The entry $\psi(\omega_1)$ in (9) is $(\omega_1, 0, \ldots, 0)$, which has x-degree $\deg \omega_1$; recall that ω_1 is chosen in $\mathbb{F}_{q^m}[x]$. Similarly, $\psi(y\omega_1) = (0, \omega_1, \ldots, 0)$ through $\psi(y^{a-1}\omega_1) = (0, 0, \ldots, \omega_1)$ have x-degree $\deg \omega_1$. The entries $\psi(y^i \omega_2 \bmod \omega_1)$ are of x-degree below $\deg \omega_1$. It thus suffices to show $\deg \omega_1 \leq \deg E$.

Let $P \in \operatorname{supp}(E)$ and consider the corresponding ideal $\mathfrak{p} = P \cap \mathcal{O}_S$ of $\mathcal{O}_S = R_\mathcal{X}$. In the constant field extension $\mathbb{F}_{q^m}(\mathcal{X}) \cdot \mathbb{F}_{q^{m \deg P}}$, the place P splits completely into $\deg P$ extensions of P; all of these have degree 1. Hence, there exist $(\zeta_1, \zeta_2) \in \mathbb{F}^2_{q^{m \deg P}}$ such that the set of extensions of P corresponds to the orbit of (ζ_1, ζ_2) under the action of $\operatorname{Gal}(\mathbb{F}_{q^{m \deg P}}/\mathbb{F}_{q^m})$. Let $\operatorname{Orb}(\zeta_1)$ be the orbit of ζ_1 under the action of $\operatorname{Gal}(\mathbb{F}_{q^{m \deg P}}/\mathbb{F}_{q^m})$ and let $d_P = |\operatorname{Orb}(\zeta_1)|$. The monic polynomial

$$\omega_{P,1} = \prod_{\alpha \in \operatorname{Orb}(\zeta_1)} (x - \alpha) \quad \in \mathbb{F}_{q^{m \deg P}}[x] \tag{11}$$

has degree $d_P \leq \deg P$. By construction $\omega_{P,1}$ is Galois-stable, so $\omega_{P,1} \in \mathbb{F}_{q^m}[x]$. Also, $\omega_{P,1}$ vanishes at $(x, y) = (\zeta_1, \zeta_2)$, so $\omega_{P,1} \in \mathfrak{p}$.

Now multiply across P: the product $Q = \prod_{P \in \operatorname{supp}(E)} \omega_{P,1}^{e_P} \in \mathbb{F}_{q^m}[x]$, where $E = \sum_P e_P P$, is a nonzero element of the ideal $\phi(E) = \prod_\mathfrak{p} \mathfrak{p}^{e_P}$, and has degree $\sum_{P \in \operatorname{supp}(E)} e_P d_P \leq \sum_{P \in \operatorname{supp}(E)} e_P \deg P = \deg E$. But ω_1 is chosen to generate $\phi(E) \cap \mathbb{F}_{q^m}[x]$, so $\deg \omega_1 \leq \deg Q \leq \deg E$ as claimed. □

Algorithm 2 presents BuildRatFunc, an updated version of Algorithm 1.

Proposition 4.11. *Algorithm* BuildRatFunc *is correct and runs in time* $\tilde{\mathcal{O}}(a^{\omega+1}(\deg E)^2)$.

Proof. The correctness of the algorithm follows directly from that of Algorithm 1.

The most time consuming steps of the algorithm are 2, 5, 7 and 8. Step number 2 consists of the product of $\deg E$ ideals. Let $\mathbf{M}_{\mathfrak{p}_i}$ be the $2a \times a$ matrix

Algorithm 2 BuildRatFunc

Input : A set of places $\mathcal{P} = \{P_1, \ldots, P_r\} \subset \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$ of $\mathcal{X}$ and a set of positive integers $\{e_1, \ldots, e_r\} \subseteq \mathbb{N}$.
Output : A rational function $f \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $(f)^+ \leq \sum_{i=1}^r e_i P_i$ and $\deg(f)^+ \geq \sum_{i=1}^r e_i - g(\mathcal{X})$.

1. Put $E = \sum_{i=1}^r e_i P_i$;
2. Compute $\phi(E) = \prod_{i=1}^r \mathfrak{p}^{e_i}$;
3. Let $\mathbf{M}_E \in \mathbb{F}_{q^m}[x]^{2a \times a}$ be the matrix having rows $\psi(\phi(E))$;
4. Let $\mathbf{s} = (0, \lfloor b/a \rfloor, \lfloor 2b/a \rfloor, \ldots, \lfloor (a-1)b/a \rfloor)$;
5. Compute Popov form of $\mathbf{M}_E$ w.r.t. $\mathbf{s}$;
6. Let $\mathbf{v} = (v_0(x), v_1(x), \ldots, v_{a-1}(x))$ be the minimal row of $\mathbf{M}_E$. Put $f_1 = \sum_{i=0}^{a-1} v_i(x) y^i$;
7. Let $\mathbf{M}'_E$ be the matrix having rows $\psi(\langle f_1 \rangle / \phi(E))$;
8. Compute Popov form of $\mathbf{M}'_E$;
9. Let $\mathbf{u} = (u_0(x), u_1(x), \ldots, u_{a-1}(x))$ be the minimal row of $\mathbf{M}'_E$. Put $f_2 = \sum_{i=0}^{a-1} u_i(x) y^i$;
10. Return $f = f_1 / f_2$.

consisting of the rows corresponding to $\psi(\mathfrak{p}_i)$ as in (9). We then compute the Popov form of $\mathbf{M}_{\mathfrak{p}_i}$, i.e., an $a \times a$ matrix. From Theorem 4.10, this can be done in $\tilde{\mathcal{O}}(|\mathcal{P}| 2a^\omega \max_i(\deg P_i))$ for all $|\mathcal{P}|$ places P_i. The product of two ideals $\mathfrak{p}_i$ and $\mathfrak{p}_j$ can be carried out by computing the Popov form of the $a^2 \times a$ matrix over $\mathbb{F}_{q^m}[x]$ whose rows consist of the products of the functions represented by the rows of $\mathbf{M}_{\mathfrak{p}_i}$ and those of $\mathbf{M}_{\mathfrak{p}_j}$. From Theorem 4.10 we have that the product $\phi(E)$ of $\deg E$ ideals can be computed in time $\tilde{\mathcal{O}}(a^{\omega+1}(\deg E)^2)$. Step 5 and 8 compute the Popov form of $a \times a$ matrices which have entries that are $\deg_x$ bounded by $\deg E$, requiring $\tilde{\mathcal{O}}(a^\omega \deg E)$ time. Finally step 7 inverts the ideal $\phi(E)$. We can do this using [33, Algorithm 5, p. 367]. This mainly requires the computation of the Popov form of a $a^2 \times a$ matrix which has entries that are $\deg_x$ bounded by $\deg E$ which again can be carried out in time $\tilde{\mathcal{O}}(a^\omega \deg E)$. □

Remark 4.12. Our reference Sage code does not implement the procedure for step 7 pointed out in the proof of Proposition 4.11, but uses the built-in method carrying out ideal inversion. A method computing the product of ideals is implemented.

Remark 4.13. Observe that the function $f \in \mathbb{F}_{q^m}(\mathcal{X})$ computed by Algorithm 1, and therefore also by Algorithm BuildRatFunc, has at most $g(\mathcal{X})$ poles away from P_∞ (see $\deg E''$ in the proof of Proposition 4.4). Formally, $\deg((f)^- + v_{P_\infty}(f) P_\infty) \leq g(\mathcal{X})$.

In our AG code-based scheme, we will call Algorithm BuildRatFunc on a randomly chosen set of places $\mathcal{P} \subset \mathcal{P}_\mathcal{X}$ and a set of integers, typically all ones. Experiments with our software suggest that the actual results are frequently better than the bounds: for example, often $\deg((f)^- + v_{P_\infty}(f) P_\infty) \leq g(\mathcal{X})/2$.

Algorithm 3 BuildRatFuncR$_{\mathsf{X}}$

Input : A set of places $\mathcal{P}_h \subset \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$ of degree 1.
Output : A rational function $h \in R_\mathcal{X}$ having simple zeros in a subset of $\mathcal{P}_h$.

1. For every $\alpha \in \mathbb{F}_{q^m}$ let $\mathcal{P}_\alpha = \{P \in \mathcal{P}_\mathcal{X} \mid v_P(x-\alpha) = 1\}$;
2. Let $A = \{\alpha \in \mathbb{F}_{q^m} \mid \mathcal{P}_\alpha \subseteq \mathcal{P}_h, \mathcal{P}_\alpha \neq \emptyset\}$;
3. Compute $h = \prod_{\alpha\in A}(x-\alpha)$;
4. Return h.

Changes by $\tilde{\mathcal{O}}(g(\mathcal{X}))$ do not affect the asymptotic benefits of, e.g., Hermitian codes (see [13, Appendix A]) but are noticeable for concrete sizes.

4.2. Construction of rational functions in $R_\mathcal{X}$ with simple zeros in a predetermined set of places of degree one. This subsection describes a simple way to construct a function $h \in \mathbb{F}_{q^m}(\mathcal{X})$ suitable for Section 3. We start with a set of places of degree one $\mathcal{P}_h \subseteq \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$. We would like a function $h \in \mathbb{F}_{q^m}(\mathcal{X})$ such that $v_P(h) = 1$ for every $P \in \mathcal{P}_h$. The code-based scheme we will describe uses the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ with $H = (h)$ for h having poles only at P_∞, i.e., $h \in R_\mathcal{X}$. This last condition on the poles requires relaxing the requirement $v_P(h) = 1$ for every $P \in \mathcal{P}_h$. Indeed, we can compute $h \in R_\mathcal{X}$ such that $v_S(h) = 1$ for every S in a subset $\mathcal{S} \subseteq \mathcal{P}_h$.

Algorithm 3 states a short algorithm BuildRatFuncR$_{\mathsf{X}}$ that computes such a function h. The following proposition shows correctness of the algorithm.

Proposition 4.14. *Given a set $\mathcal{P}_h \subseteq \mathcal{P}_\mathcal{X} \setminus \{P_\infty\}$ of places of degree one, Algorithm* BuildRatFuncR$_{\mathsf{X}}$ *computes a function $h \in R_\mathcal{X}$ such that $v_S(h) = 1$ for every $S \in \mathcal{S}$ with*

$$\mathcal{S} = \bigcup_{\alpha\in A} \mathcal{P}_\alpha,$$

where A and $\mathcal{P}_\alpha$ are defined in Algorithm BuildRatFuncR$_{\mathsf{X}}$.

Proof. The function h is a polynomial in x, therefore the only pole of h is that of x, i.e., P_∞, so $h \in R_\mathcal{X}$. The statement follows noting that the function $x - \alpha$ has zeros at the places in $\mathcal{P}_\alpha$. □

Note that the condition on $\mathcal{P}_\alpha$ excludes points $P \in \mathcal{P}_h$ for which $x - \alpha$ intersects $\mathcal{X}$ with multiplicity > 1, i.e., is a tangent to the curve in P. For each α, $\mathcal{P}_\alpha$ contains up to a places P with $v_P(x-\alpha) = 1$. The condition on A excludes $\mathcal{P}_\alpha$ if not all of these places are contained in $\mathcal{P}_h$. For constructive applications this means that $\mathcal{P}_h$ is chosen to contain all of them if it contains one.

5 Decoding $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$

The algorithm we describe is a generalization of Gao's decoding algorithm for Reed-Solomon codes [30] to the case of $C_{\mathcal{X},\mathcal{L}}(D, E)$ where D and $E = \ell P_\infty$,

with $\ell \in \mathbb{N}$, are two disjoint divisors on $\mathcal{X}$. Given such a decoder, we can take advantage of the evaluation view of the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ with $M = (\gamma)^- + v_{P_\infty}(\gamma)P_\infty$ given in Theorem 3.1, and thus decode $C_{\mathcal{X},\mathcal{L}}(H^+, E)$ where $E = H^- - G^- + M + (\eta)$. Indeed, if C is any linear code, given the decoder for a code isometric to C, it is straightforward to obtain a decoder for C. In particular, from Theorem 3.1 we have that $C_{\mathcal{X},\Omega}(H^+, G^+ - M) = \boldsymbol{\beta} C_{\mathcal{X},\mathcal{L}}(H^+, E)$ for some $\boldsymbol{\beta} \in \mathbb{F}_{q^m}^n$. This means that given a decoder $\mathsf{Dec}_{\mathcal{L}}$ for the code $C_{\mathcal{X},\mathcal{L}}(H^+, E)$ we can decode a noisy codeword $\mathbf{w} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ by computing $\mathbf{w}' = (w_1\beta_1^{-1}, \dots w_n\beta_n^{-1})$, decoding $\mathbf{w}'$ as $\mathbf{c}' = \mathsf{Dec}_{\mathcal{L}}(w')$ and finally computing $\mathbf{c} = (c'_1\beta_1, \dots, c'_n\beta_n)$.

For this decoder, we will set up a key equation and solve it for the so-called *error locator function*. The approach we outline extends that of [46] in which only the Hermitian curve is considered and all the places in $\mathcal{P}_\mathcal{X}$ of degree 1, except for P_∞, are chosen as evaluation points of the code. In particular, we combine the strategy in [46] with the interpolation techniques in [40].

For the rest of this section, let $\mathbf{w} = \mathbf{c} + \mathbf{e}$ be a noisy codeword with $\mathbf{c} = \mathrm{ev}_E(f) \in C_{\mathcal{X},\mathcal{L}}(H^+, E)$ for some $f \in \mathcal{L}(E)$ and $\mathbf{e} \in \mathbb{F}_{q^m}^n$ being the error vector. Let also $\mathcal{P}_h = (P_1, \dots, P_n) \in (\mathcal{P}_\mathcal{X})^n$ be such that $c_i = f(P_i)$ for each $i = 1, \dots, n$ and $I_\mathbf{e} = \{i \in \{1, \dots, n\} | e_i \neq 0\}$. Let $\mathfrak{p}_i \subset R_\mathcal{X}$ be the maximal ideal corresponding to P_i under the correspondence in Proposition 4.5. Then we have that $\mathfrak{p}_\mathfrak{i} + \prod_{j=1:j\neq i}^n \mathfrak{p}_j = R_\mathcal{X}$, by coprimality of the involved ideals. This means that we can find $f_i \in \mathfrak{p}_i$ and $g_i \in \prod_{j\neq i} \mathfrak{p}_j$ such that $f_i + g_i = 1$. Finally, define the ideal $J = \prod_{i=1}^n \mathfrak{p}_i \subset R_\mathcal{X}$.

We recall the notion of Lagrange basis, introduced by Lee, Bras and Sullivan in [40].

Definition 5.1 (Lagrange basis). *Let $\mathcal{P}_h = (P_1, \dots, P_n) \in (\mathcal{P}_\mathcal{X})^n$ be a tuple of n places of degree one. The tuple $\mathrm{LB} = (g_1, \dots, g_n) \in (R_\mathcal{X})^n$ such that $g_i \in \prod_{j\neq i} \mathfrak{p}_j$ and $\exists f_i \in \mathfrak{p}_\mathfrak{i}$ s.t. $g_i + f_i = 1$ is called a Lagrange basis for $\mathcal{P}_h$.*

Clearly, $v_{P_i}(g_i) = 0$ for $i = 1, \dots, n$ and $v_{P_j}(g_i) \geq 1$ if $j \neq i$. For the code-based scheme we propose, we will only need to compute a Lagrange basis once, namely in the key generation phase, and use it in the decoding phase. For this reason we describe a dedicated algorithm computing such basis. An easy way to compute a Lagrange basis for $\mathcal{P}_h$ is given by Algorithm 4.

We give the definition of two rational functions that will be central in the decoding process described in this section. The first one is the interpolating function.

Definition 5.2 (Interpolating function). *Given a tuple of n places of degree one $\mathcal{P}_h = (P_1, \dots, P_n) \in (\mathcal{P}_\mathcal{X})^n$, an element $\mathbf{w} \in \mathbb{F}_{q^m}^n$ and a Lagrange basis $\mathrm{LB} = (g_1, \dots, g_n) \in (R_\mathcal{X})^n$ for $\mathcal{P}_h$, define the function $R \in R_\mathcal{X}$ as $R = \mathrm{LB} \cdot \mathbf{w} = \sum_{i=1}^n g_i w_i$. We call R the interpolating function of $\{(P_i, w_i) | i = 1, \dots, n\}$ under* LB.

The second one is the error locator function.

Algorithm 4 GenLagrangeBasis

Input : A tuple of n places of degree one $\mathcal{P}_h = (P_1, \ldots, P_n) \in (\mathcal{P}_\mathcal{X})^n$.
Output : A Lagrange basis $\mathrm{LB} \in (R_\mathcal{X})^n$ for $\mathcal{P}_h$.

1. Let $A = \{\alpha \in \mathbb{F}_{q^m} \mid \exists \beta \in \mathbb{F}_{q^m} \text{ s.t } \langle x-\alpha, y-\beta\rangle \in \mathcal{P}_h\}$;
2. Let $B_\alpha = \{\beta \in \mathbb{F}_{q^m} \mid \langle x-\alpha, y-\beta\rangle \in \mathcal{P}_h\}$ for every $\alpha \in A$;
3. For each $P_i = \langle x-\alpha_i, y-\beta_i\rangle$ do:
   (a) Compute $g_{i,x} = \prod_{\alpha\in A, \alpha\neq\alpha_i}(x-\alpha)$;
   (b) Compute $g_{i,y} = \prod_{\beta\in B_{\alpha_i}, \beta\neq\beta_i}(y-\beta)$
   (c) Compute $g_i = g_{i,x}g_{i,y}\cdot(g_{i,x}(\alpha_i)g_{i,y}(\beta_i))^{-1}$;
4. Return $(g_1, \ldots, g_n)$.

Definition 5.3 (Error locator function). *Given a tuple of n places of degree one $\mathcal{P}_h = (P_1, \ldots, P_n) \in (\mathcal{P}_\mathcal{X})^n$ and a vector $\mathbf{e} \in \mathbb{F}_{q^m}^n$, we define the error locator function to be the element*

$$\Lambda \in \prod_{i\in I_\mathbf{e}} \mathfrak{p}_i \subset R_\mathcal{X}$$

such that $-v_{P_\infty}(\Lambda)$ is minimal.

The value $-v_{P_\infty}(\Lambda)$ is bounded as follows.

Lemma 5.4. $\mathrm{wt}(\mathbf{e}) \leq -v_{P_\infty}(\Lambda) \leq \mathrm{wt}(\mathbf{e}) + g(\mathcal{X})$.

Proof. Observe that, as in proof of Proposition 4.8, $\Lambda \in \mathcal{L}(\sum_{i\in I_\mathbf{e}} P_i - \mathrm{wt}(\mathbf{e})P_\infty)$, hence $\mathrm{wt}(\mathbf{e}) \leq -v_{P_\infty}(\Lambda)$.

To prove $-v_{P_\infty}(\Lambda) \leq \mathrm{wt}(\mathbf{e}) + g(\mathcal{X})$, define $D = -\sum_{i\in I_\mathbf{e}} P_i + \mathrm{wt}(\mathbf{e})P_\infty \in \mathrm{Div}^0(\mathcal{X})$, and consider the smallest nonnegative integer $a \leq g(\mathcal{X})$ such that D is equivalent to $A - aP_\infty$ for some divisor $A \geq 0$ with $\deg A = a$. Such integers exist by Theorem 4.2. Now $A = D + aP_\infty + (f)$ for some $f \in \mathbb{F}_{q^m}(\mathcal{X})$, so $(f) = A - D - aP_\infty = A + \sum_{i\in I_\mathbf{e}} P_i - (\mathrm{wt}(\mathbf{e}) + a)P_\infty$, implying $f \in \prod_{i\in I_\mathbf{e}} \mathfrak{p}_i$. Finally, $-v_{P_\infty}(\Lambda)$ is minimal by definition of Λ, so $-v_{P_\infty}(\Lambda) \leq -v_{P_\infty}(f) = -v_{P_\infty}(A) + \mathrm{wt}(\mathbf{e}) + a \leq \mathrm{wt}(\mathbf{e}) + g(\mathcal{X})$. □

Fix now a Lagrange basis LB for $\mathcal{P}_h$ and let $R \in R_\mathcal{X}$ be the interpolating function of $\{(P_i, w_i) \mid i = 1, \ldots, n\}$ under LB.

Lemma 5.5. *We have the congruence $\Lambda f \equiv \Lambda R \bmod J$ over $R_\mathcal{X}$.*

Proof. We prove that $\Lambda(f-R) \in J$ by showing that $\Lambda(f-R) \in \mathfrak{p}_i$ for every $i = 1, \ldots, n$, hence $\Lambda(f-R) \in \bigcap_{i=1}^n \mathfrak{p}_i$. Then, since the ideal $\mathfrak{p}_i$ is maximal for every i and $\mathfrak{p}_i \neq \mathfrak{p}_j$ for $i \neq j$ we have that $\bigcap_{i=1}^n \mathfrak{p}_i = \prod_i^n \mathfrak{p}_i = J$.

Let $i \in \{1, \ldots, n\}$ and assume that $e_i = 0$. Then $R(P_i) = w_i = c_i + e_i = f(P_i)$, and thus $\Lambda(f-R)(P_i) = 0$, meaning that $\Lambda(f-R) \in \mathfrak{p}_i$. Assume now that $e_i \neq 0$, meaning that $i \in I_\mathbf{e}$. In this case, we have $\Lambda \in \mathfrak{p}_i$ from Definition 5.3. Hence $\Lambda(f-R) \in \mathfrak{p}_i$. □

The known terms in the key congruence are R and J, while Λ and f are unknowns. We can linearize the congruence setting $\lambda = \Lambda f$ and solve for λ and Λ. Recall that, since $R_{\mathcal{X}}$ is a Dedekind domain, we can write $J = \langle \omega_1, \omega_2 \rangle$, for some $\omega_1, \omega_2 \in R_{\mathcal{X}}$. Then we can express $\lambda = \Lambda R + \rho_1 \omega_1 + \rho_2 \omega_2$ for $\rho_1, \rho_2 \in R_{\mathcal{X}}$.

Consider the $\mathbb{F}_{q^m}[x]^{2a}$-submodule generated by the following basis matrix

$$\mathbf{M} = \left(\begin{array}{c|c} & \psi(R) \\ \mathbf{I}_a & \vdots \\ & \psi(y^{a-1}R) \\ \hline & \psi(\omega_1) \\ \mathbf{0} & \vdots \\ & \psi(y^{a-1}\omega_1) \\ \hline & \psi(\omega_2) \\ \mathbf{0} & \vdots \\ & \psi(y^{a-1}\omega_2) \end{array}\right) \in \mathbb{F}_{q^m}[x]^{3a \times 2a}, \tag{12}$$

where $\mathbf{I}_a$ is the $a \times a$ identity matrix. Since $\lambda = \Lambda f$ with $f \in \mathcal{L}(E)$, we have

$$-v_{P_\infty}(\lambda) = -v_{P_\infty}(\Lambda) - v_{P_\infty}(f) \leq -v_{P_\infty}(\Lambda) + \deg E < -v_{P_\infty}(\Lambda) + \deg E + 1.$$

The solutions of the key congruence are thus the elements (Λ', λ') of the module generated by $\mathbf{M}$ that have $-v_{P_\infty}(\lambda') < -v_{P_\infty}(\Lambda) + \deg E + 1$. We need to find such a solution (Λ', λ') with minimal $-v_{P_\infty}(\Lambda')$ and hope that $\Lambda' = \Lambda$. As in [46, Section V.B], we can express this constraint in terms of a norm on the space $\mathbb{F}_{q^m}[x]^{2a}$. We thus set $\mathbf{s} = \mathbf{s}_1 || \mathbf{s}_2 \in \mathbb{N}^{2a}$ where

$$\mathbf{s}_1 = (\lfloor (\deg E + 1)/a \rfloor, \lfloor (b + \deg E + 1)/a \rfloor, \ldots, \lfloor ((a-1)b + \deg E + 1)/a \rfloor),$$

and

$$\mathbf{s}_2 = (0, \lfloor b/a \rfloor, \ldots, (a-1)\lfloor b/a \rfloor).$$

A valid solution of the key congruence is therefore the minimal element of the Popov form of the matrix $\mathbf{M}$ w.r.t. the shift $\mathbf{s}$ having the maximum $\deg_x$ of its entries among the first a positions. Let (Λ', λ') be a solution of the key congruence. Then, the congruence can be turned into an equivalence whenever

$$-v_{P_\infty}(\lambda') = -v_{P_\infty}(\Lambda) - v_{P_\infty}(f) < n.$$

In other words, since $f \in \mathcal{L}(E)$, using Lemma 5.4 we obtain that the congruence can be regarded as an equivalence when $\mathrm{wt}(\mathbf{e}) < n - \deg E - g(\mathcal{X})$. In this case, we can recover $f = \lambda'/\Lambda'$ and $\mathbf{e} = \mathrm{ev}_E(R - f)$.

Algorithm 5 presents our decoding algorithm Decode.

The next lemma states the complexity of computing the input ideal J. This ideal encodes the support of the code we want to decode and can be computed a priori and used multiple times in the Decode algorithm. Moreover, our scheme will use the principal ideal $J = \langle h \rangle$ which completely avoids the cost of the product. We give the following lemma since the shape of J is relevant in Proposition 5.7.

Algorithm 5 Decode

Input : a matrix $\mathbf{H} \in \mathbb{F}_{q^m}^{mk \times n}$, an ideal $J = \langle \omega_1, \omega_2 \rangle \subset R_{\mathcal{X}}$, a Lagrange basis LB for $\mathcal{P}_h$ and the received word $w \in \mathbb{F}_{q^m}^n$.
Output : $\mathbf{e} \in \mathbb{F}_{q^m}$ with $\mathrm{wt}(\mathbf{e}) = t$ such that $\mathbf{w} = \mathbf{H}\mathbf{e}^\top$, or $\perp$.

1. Compute $R = \sum_{i=1}^{n} g_i w_i$;
2. Let $\mathbf{s} = \mathbf{s}_1 || \mathbf{s}_2 \in \mathbb{N}^{2a}$ where
   $\mathbf{s}_1 = (\lfloor (\deg E + 1)/a \rfloor, \lfloor (b + \deg E + 1)/a \rfloor, \ldots, \lfloor ((a-1)b + \deg E + 1)/a \rfloor)$ and
   $\mathbf{s}_2 = (0, \lfloor b/a \rfloor, \ldots, \lfloor (a-1)b/a \rfloor)$;
3. Build the matrix $\mathbf{M}$ as in (12);
4. Compute the Popov form of $\mathbf{M}$ w.r.t. $\mathbf{s}$;
5. Let $\mathbf{v} = (v_0(x), v_1(x), \ldots v_{2a-1}(x))$ be the minimal row of $\mathbf{M}$;
6. Find the smallest $i_{\max} \in \{0, 2, \ldots, 2a-1\}$ such that $\deg_x(v_{i_{\max}}(x)) \geq v_j$ for $j = 0, 1, \ldots, 2a-1$;
7. if $i_{\max} \geq a$ then return $\perp$;
8. Compute $\Lambda' = \sum_{i=0}^{a-1} v_i(x) y^i \in \mathbb{F}_{q^m}(\mathcal{X})$;
9. Compute $\lambda' = \sum_{i=0}^{a-1} v_{a+i}(x) y^i \in \mathbb{F}_{q^m}(\mathcal{X})$;
10. Compute $\mathbf{e}' = \mathbf{w} - \mathrm{ev}_E(\lambda'/\Lambda')$;
11. If $\mathbf{H}\mathbf{e}'^\top = \mathbf{w}$ then return $\mathbf{e}'$;
12. Otherwise return $\perp$.

Lemma 5.6. *Let $P_1, \ldots, P_n \in \mathcal{P}_{\mathcal{X}}$ be distinct places of degree 1. Let $\mathfrak{p}_1, \ldots, \mathfrak{p}_n \subset R_{\mathcal{X}}$ be the corresponding prime ideals. The complexity of computing the product $J = \prod_{i=1}^{n} \mathfrak{p}_i$ is $\tilde{\mathcal{O}}(a^{\omega+1} n \min(q^m, n))$.*

Proof. The complexity is that of computing the product of n ideals $\mathfrak{p}_i$ whose corresponding $a \times a$ Popov form matrices over $\mathbb{F}_{q^m}[x]$ have $\deg_x \mathbf{M}_{\mathfrak{p}_i} = 1$. The strategy is the same as that for computing $\phi(E)$ in Algorithm BuildRatFunc. We need to bound the degree of the entries of the basis matrix $\mathbf{M}_J$ representing the ideal, similarly to Theorem 4.10. Write $\mathfrak{p}_i = \langle \omega_{i,1}, \omega_{i,2} \rangle$ where $\omega_{i,1} = x - \alpha_i$ for some $\alpha_i \in \mathbb{F}_{q^m}$. Note that if $\omega_{i,1} = \omega_{j,1}$ then $\omega_{i,1} \in \mathfrak{p}_i \mathfrak{p}_j$, and similarly for the product of more than two ideals. Hence $\omega_1 \in J$ where ω_1 is the least common multiple of $\omega_{1,1}, \ldots, \omega_{n,1}$. We have $\deg \omega_1 \leq n$ since $\deg \omega_{i,1} = 1$, and we have $\deg \omega_1 \leq q^m$ since ω_1 divides $x^{q^m} - x$. Consequently $\deg_x \mathbf{M}_J \leq \min(q^m, n)$. Hence the complexity is that of computing the Popov form of n $a^2 \times a$ matrices over $\mathbb{F}_{q^m}[x]$ with $\deg_x$ bounded by $\min(q^m, n)$, i.e. $\tilde{\mathcal{O}}(a^{\omega+1} n \min(q^m, n))$. □

Proposition 5.7. *Algorithm* Decode *runs in time $\tilde{\mathcal{O}}(a^\omega \min(q^m, n))$ and corrects t errors for any $t \leq (n - \deg E - g(\mathcal{X}) - 1)/2$.*

Proof. The complexity of the algorithm is dominated by that of computing the Popov form of the matrix $\mathbf{M}$ in step 5. Every function $g \in \mathrm{LB}$ have $\deg_x g = n - 1$. Moreover as in Lemma 5.6, we can write $J = \langle \omega_1, \omega_2 \rangle$ where $\deg_x(\omega_1) \leq \min(q^m, n)$. Thus the matrix built in step 4 has $\deg_x \mathbf{M} \leq \min(q^m, n)$. The complexity of computing the Popov form of $\mathbf{M}$ is then $\tilde{\mathcal{O}}(a^\omega \min(q^m, n))$.

The decoding bound follows exactly from [46, Proposition 28]. □

6 An AG-code-based PKE

This section modifies the original McEliece [41] public-key encryption scheme (PKE) to use a code of the form $\Gamma_{\mathcal{X}}(h, \gamma, M)$ instead of a classical Goppa code $\Gamma_{\mathbb{P}^1}(h, \gamma)$. We first discuss the rationale behind the choice of the elements of the code and thereafter we give the algorithms of the scheme. We highlight again that the family of codes includes classical Goppa codes, hence is at least as rich as that used in McEliece's original scheme. We include Niederreiter's ciphertext compression [45]; this compression is not affected by the choice of code.

The generic conversion "ImplicitRejection" defined in [15] (see also [49, Section 5.3] and [35]) produces a key-encapsulation mechanism (KEM) with a tight proof of QROM IND-CCA2 security (see [17]) under the hypothesis of one-wayness (OW-CPA) for this PKE, because this is a deterministic correct PKE. We focus on the PKE here.

It is important to note that there are many variables that can be adjusted in this scheme. The description below allows the code parameters to vary by $O(g)$ around their initial targets; one can instead specify parameters in advance (using rejection sampling to fix a value of t, and adding some random parity checks to always reduce the dimension down to a prespecified target). One can even generate a secret per-user curve as part of key generation, although that would raise the question of how to efficiently generate a secret curve with many points; for simplicity, we instead fix a curve for all users (still generalizing McEliece's original system, which fixes the curve $\mathbb{P}^1$ for all users). Even without such variations, optimizing all of the variables is an interesting challenge.

Fix an $\mathbb{F}_q$-basis $\{1, z, z^2, \ldots, z^m\}$ of $\mathbb{F}_{q^m}$. Let $\mathcal{X}$ be a $C_{a,b}$ curve. Let $\mathcal{P}_\gamma \subset \mathcal{X}(\mathbb{F}_{q^m})$ be a set of places of any degree. We can build the rational function $\gamma \in \mathbb{F}_{q^m}(\mathcal{X})$ having zeros with bounded multiplicities at places in $\mathcal{P}_\gamma$ using Algorithm BuildRatFunc. Our scheme makes use of the algorithm Decode described in Section 5, which works for one-point evaluation AG codes over $C_{a,b}$ curves. We thus want the code $\Gamma_{\mathcal{X}}(h, \gamma, M)$ to be a subfield subcode of a code isometric to a one-point evaluation AG code. Corollary 3.7 gives sufficient conditions for this to happen, namely $(h)^- = nP_\infty$ and $M = (\gamma)^- + v_{P_\infty}(\gamma)P_\infty$. Finally, we need that $\operatorname{supp}(\gamma) \cap \operatorname{supp}(h)^+ = \emptyset$, hence we take $\mathcal{P}'_h \subseteq \{P \in \mathcal{X}(\mathbb{F}_{q^m}) \setminus \operatorname{supp}((\gamma)) \mid \deg P = 1\}$ and call Algorithm BuildRatFuncRX($\mathcal{P}'_h$) to compute h.

Algorithms 6, 7, 8 and 9 present, respectively, the subroutine computing the parity check matrix of a code $\Gamma_{\mathcal{X}}(h, \gamma, M)$; the key-generation algorithm KeyGen; the encryption algorithm Enc; and the decryption algorithm Dec. In Algorithm 6, RREF($\mathbf{M}$) denotes the reduced row echelon form of a matrix $\mathbf{M}$.

Remark 6.1. Using a decoder for general evaluation AG codes over $C_{a,b}$ curves, i.e. not only one-point codes, one can easily drop all the extra requirements induced by the one-point decoder. Nevertheless, without taking $M = (\gamma)^- + v_{P_\infty}(\gamma)P_\infty$, one might incur higher costs in the computation of the basis of the Riemann-Roch space $\mathcal{L}(G^- - M)$ during the construction of the parity check matrix of the codes. See the discussion in Section 3.3.

Algorithm 6 GenPCMat

Input : A tuple of n places of degree one $\mathcal{P}_h = (P_1, \ldots, P_n)$, a rational function $\gamma \in \mathcal{X}(\mathbb{F}_{q^m})$, integers t, μ.

Output : Redundancy part $\mathbf{A} \in \mathbb{F}_q^{mk \times (n-mk)}$ of the parity check matrix of the code $\Gamma_{\mathcal{X}}(h, \gamma, M)$.

1. Compute the $k \times n$ matrix $\hat{\mathbf{H}} = \{h_{i,j}\}$ over $\mathbb{F}_q$ where $h_{i,j} = \frac{\varphi_i(P_j)}{\gamma(P_j)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, n$, where $\{\varphi_1, \ldots, \varphi_k\} = \{x^{\ell_x} y^{\ell_y} \mid a\ell_x + b\ell_y \leq t - \mu, \ell_y < a\}$;
2. Replace each entry $h_{i,j}$ of $\hat{\mathbf{H}}$ with a column vector $(u_0, u_1, \ldots, u_{m-1}) \in \mathbb{F}_q^m$ such that $h_{i,j} = u_0 + u_1 z + \cdots + u_{m-1} z^{m-1}$.
3. Compute $\mathbf{H} = \mathsf{RREF}(\hat{\mathbf{H}})$. If $\mathbf{H}$ has shape $(\mathbf{I}_{mk} \mid \mathbf{A})$, return $\mathbf{A}$. Otherwise, return $\perp$.

7 Elliptic parameters

The wide range of variables supported by the family of AG codes in Section 3, starting with the choice of genus and the choice of curves, provides a large space of parameters to use in cryptosystems; see Section 6 for a PKE that replaces the classical binary Goppa decoder in the McEliece system with Algorithm 5. To demonstrate that moving to higher genus improves upon the McEliece system, this section gives concrete parameter examples specifically for genus 1, the case of elliptic curves.

We are not claiming that genus 1 is optimal. On the contrary, allowing the genus to grow with n is asymptotically better; see [13, Appendix A]. However, for purposes of demonstrating a nonzero improvement for concrete sizes, it logically suffices to show such improvements in genus 1. Focusing on genus 1 makes curve precomputations as easy as possible to carry out and to verify.

Section 7.1 collects the basic constraints on parameter selection. Section 7.2 gives a numerical example showing that Algorithm 5 already corrects significantly more errors than a classical binary Goppa decoder even for rather small n. Section 7.3 gives numerical examples in cryptographic sizes.

7.1. Parameter selection for any genus. Let $\gamma \in \mathbb{F}_{q^m}(\mathcal{X})$ be such that $\deg(\gamma)^+ = t$. Put $M = (\gamma)^- + v_{P_\infty}(\gamma) P_\infty$ so $\mu = \deg M \leq g(\mathcal{X})$. Consider the code $C_{\mathcal{X},\Omega}(H^+, G^+ - M)$ for some $h \in \mathbb{F}_{q^m}(\mathcal{X})$ as in Section 3. By Theorem 3.1 we have that this code is isometric to some evaluation code given by a divisor E of degree $\deg E = n - t + \mu + 2g(\mathcal{X}) - 2$. Our decoder, described in Algorithm 5, corrects $(n - \deg E - g(\mathcal{X}) - 1)/2 = (t - \mu - 3g(\mathcal{X}) + 1)/2$ errors.

We choose γ as a qth power. Then there is a divisor G_1 such that $qG_1 \leq (\gamma)^+ - M$ with $\deg G_1 \geq t/q - \mu$. We assume that t is large enough to have $\deg G_1 \geq 2g(\mathcal{X}) - 1$; then, by Corollary 3.9, we have $\dim \Gamma_{\mathcal{X}}(h, \gamma, M) \geq n - m(t - \mu - \deg G_1) \geq n - mt(q-1)/q$.

The code dimension is thus at least k if $t \leq ((n-k)/m)q/(q-1)$. Assume for simplicity that $((n-k)/m)q/(q-1)$ is an integer and that t is exactly this integer; then our decoder corrects $(((n-k)/m)q/(q-1) - \mu - 3g(\mathcal{X}) + 1)/2$ errors

Algorithm 7 KeyGen

Input : $\emptyset$.
Output : A public key $(\mathbf{A}, t) \in \mathbb{F}_{q^m}^{mk\times(n-mk)} \times \mathbb{N}$ and a private key $(\mathcal{P}_h, \gamma) \in \mathbb{F}_{q^m}(\mathcal{X})^2$.

1. Select $\mathcal{P}_\gamma$, a randomly chosen subset of $\mathcal{X}(\mathbb{F}_{q^m})$ consisting of places of degree larger than 1 and such that $\sum_{P\in\mathcal{P}_\gamma} \deg P = \tilde{t}$;
2. Select $\mathsf{mult} = \{e_i \in \mathbb{N}\}$ of cardinality $|\mathcal{P}_\gamma|$;
3. Compute γ as output of $\mathsf{BuildRatFunc}(\mathcal{P}_\gamma, \mathsf{mult})$;
4. Define $M = (\gamma)^- + v_{P_\infty}(\gamma)P_\infty$ and $\mu = \deg M$;
5. If $\mu > \lfloor g(\mathcal{X})/2 \rfloor$: go to step 1;
6. Define $t = \deg((\gamma)^+)$;
7. Put $k = t - \mu - g(\mathcal{X}) + 1$;
8. Select $\mathcal{P}'_h$, a randomly chosen subset of cardinality n of $\mathcal{X}(\mathbb{F}_{q^m})\backslash\mathrm{supp}(M)$ of places of degree 1;
9. Compute h as output of $\mathsf{BuildRatFuncRX}(\mathcal{P}_h)$;
10. Define an ordered tuple $P_h = (P_1, \ldots, P_n)$ from the set $\mathrm{supp}((h^+))$ containing each element only once;
11. Compute LB as output of $\mathsf{GenLagrangeBasis}(\mathcal{P}_\mathsf{h})$;
12. Compute $\mathbf{A} \in \mathbb{F}_q^{mk\times(n-mk)}$ as output of $\mathsf{GenPCMat}(\mathcal{P}_h, \gamma, t, \mu)$. If $\mathbf{A} = \perp$, restart the algorithm;
13. Return $(\mathbf{A}, t)$ as public key and $(\mathcal{P}_h, \gamma)$ as private key.

Algorithm 8 Enc

Input : A public key $(\mathbf{A}, t) \in \mathbb{F}_q^{mk\times(n-mk)} \times \mathbb{N}$; an element $\mathbf{e} \in \mathbb{F}_q^n$ with $\mathrm{wt}(\mathbf{e}) = t$.
Output : A ciphertext $\mathbf{y} \in \mathbb{F}_q^{mk}$.

1. Compute $\mathbf{H} = (\mathbf{I}_{mk} \mid \mathbf{A})$;
2. Compute $\mathbf{y} = \mathbf{H}\mathbf{e}^\top$;
3. Return ciphertext $\mathbf{y}$.

for code dimension at least k. In particular, our decoder corrects $((n-k)/m) - (3g(\mathcal{X}) + \mu - 1)/2$ errors in the binary case $q = 2$.

For comparison, a standard decoder for a classical binary Goppa code corrects $(n-k)/m$ errors for dimension k. This may sound better because it avoids the additive loss $-(3g(\mathcal{X})+\mu-1)/2$, but the constraints on m are different: a classical binary Goppa code needs $2^m \geq n$, whereas allowing larger genus allows 2^m to be replaced by a larger number of points. We will give examples below where genus $g(\mathcal{X}) = 1$ already allows our decoder to correct more errors than the classical case for the same (n, k).

More generally, to obtain an improvement over the genus-0 case, we focus on curves such that $q^m < N(\mathcal{P}_\mathcal{X})$ where $N(\mathcal{P}_\mathcal{X})$ is the number of places of degree 1, and we choose n with $q^m < n \leq N(\mathcal{P}_\mathcal{X})$. Note that the Hasse–Weil bound $|N(\mathcal{P}_\mathcal{X}) - (q^m + 1)| \leq 2g(\mathcal{X})\sqrt{q^m}$ (see e.g., [55, Theorem 5.2.3]) then forces $g(\mathcal{X}) \geq (n - (q^m + 1))/(2\sqrt{q^m})$.

Algorithm 9 Dec

Input : A ciphertext $\mathbf{y} = \mathsf{Enc}((\mathbf{A}, t), \mathbf{e}) \in \mathbb{F}_q^{mk}$ and a private key $(\mathcal{P}_h, \gamma) \in \mathbb{F}_{q^m}(\mathcal{X})^2$, for some $((\mathbf{A}, t), (h, \gamma))$ output by KeyGen and some $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathrm{wt}(\mathbf{e}) = t$.
Output : The same element $\mathbf{e} \in \mathbb{F}_q^n$.

1. Compute h to be the output of $\mathsf{BuildRatFuncRX}(\mathcal{P}_h)$;
2. Compute $\hat{h} = dh/dx$ where dh is the differential of h;
3. Let $\mathbf{y}' = (y_1(\hat{h}/\gamma)(P_1), \ldots, y_{mk}(\hat{h}/\gamma)(P_{mk}))$;
4. Compute $\bar{\mathbf{y}} = \mathbf{y}' \,||\, (0, \ldots, 0)$ appending $n - mk$ zeros to $\mathbf{y}$;
5. Let $J = \langle h \rangle$;
6. Compute $\mathbf{e}'$ as the output of $\mathsf{Decode}(\mathbf{H}, J, \mathrm{LB}, \bar{\mathbf{y}})$;
7. If $\mathbf{e}' = \perp$, return $\perp$;
8. Otherwise, return $(e'_1(\gamma/\hat{h})(P_1), \ldots, e'_n(\gamma/\hat{h})(P_n))$.

7.2. A small example in genus 1. See Section 7.4 for an elliptic curve having 289 points over $\mathbb{F}_{256}$. Our key generation (KeyGen in Section 6), starting with this curve, $n = 282$, and $\tilde{t} = 40$, consistently produces an $\mathbb{F}_2$ subcode having length 282, dimension 156, and distance at least 39. Our decoder is guaranteed to correct 18 errors—and our software is experimentally observed to do so.

For comparison, reaching length 282 with a classical binary Goppa code requires starting with $\mathbb{F}_{512}$, and then targeting dimension 156 for a random code requires taking degree $(282 - 156)/9 = 14$, where the usual decoders correct 14 errors. We correct 18 errors. Unlike the example from [36] discussed in Section 1, this example shows a clear increase in the number of errors corrected compared to the codes used in the original McEliece cryptosystem.

7.3. Big examples in genus 1. Security requires n to be larger than in Section 7.2. Also, code rates are normally chosen close to 0.8 to minimize key size (see [13, Appendix A]), while the code rate in Section 7.2 is only $156/282 \approx 0.55$. Table 2 presents concrete parameter examples addressing both of these issues. The specific elliptic curves used here appear in Section 7.4.

The larger example is clearly within cryptographic ranges. Both $n = 4209$ and $t = 77$ are about 20% larger than the $n = 3488$ and $t = 64$ from the `mceliece348864` proposal from [2]. The rate is the same as in `mceliece348864`, coming from the same $m = 12$. Recall from Section 1 that, for each choice of rate, the cost of known attacks increases exponentially with t.

The larger example also shows that we correct more errors for the same key size and the same ciphertext size, at specific sizes of cryptographic interest, as claimed in Section 1.1. At this size, using conventional decoders for classical binary Goppa codes limits t to just 71.

Finally, another way to see the advantage of $g = 1$ over $g = 0$ is to consider the key sizes required for a particular security level. Recall from Section 1 that the impact on key size is approximately quadratic compared to the impact on t. To concretely quantify this, we used CryptAttackTester from [12] to predict the security of the $g = 1$ parameter set $(12, 4209, 3286, 77)$ from Table 2, namely

Classical binary Goppa codes	Our codes
$m = 12, n = 2134, k = 1534, t = 50$	$m = 11, n = 2134, k = 1534, t = 54$
$m = 13, n = 4209, k = 3286, t = 71$	$m = 12, n = 4209, k = 3286, t = 77$

Table 2. Examples in the elliptic case of how Algorithm 5 for this paper's codes corrects more errors than conventional decoders for classical binary Goppa codes for the same length and dimension. The value t in the first column is the number of errors corrected by a Goppa decoder. The value t in the second column is the number of errors corrected by Algorithm 5.

$2^{176.27}$ bit operations, and then to search for $g = 0$ parameters that reach this security level. The smallest keys that we found for $g = 0$ were 16.5% larger than these $g = 1$ keys. The $g = 0$ parameters here are $(13, 4600, 3625, 75)$, with $2^{176.34}$ bit operations. We did not limit the parameter search to multiples of 5 (or 25): e.g., $(13, 4599, 3624, 75)$ is below the target security level.

7.4. Specific elliptic curves. Finally, this subsection presents three specific elliptic curves used in Sections 7.2 and 7.3.

For the smallest curve, represent $\mathbb{F}_{256}$ as $\mathbb{F}_2[u]/(u^8 + u^4 + u^3 + u^2 + 1)$. The curve $y^2 + y = x^3 + (u^3 + u)x$ over $\mathbb{F}_{256}$ is an elliptic curve having 289 points, as one can check with

```
k.<u> = GF(256)
assert u^8+u^4+u^3+u^2+1 == 0
E = EllipticCurve(k,[0,0,1,u^3+u,0])
assert E.cardinality() == 289
```

in Sage.

The larger curves can be checked similarly and are as follows:

$$y^2 + (u^{10} + u^7 + u^6 + u^5 + u^4 + u^3 + u^2)xy + (u^8 + u^6 + u^5 + u^4 + u^3 + u^2 + 1)y + x^3 + (u^{10} + u^9 + u^7 + u^5 + u^4 + u^3 + u^2 + u + 1)x^2 + (u^4 + u^3 + u + 1)x + (u^{10} + u^9 + u^6 + u^5 + u^4 + u^3 + u + 1)$$

where $\mathbb{F}_{2^{11}}$ is represented as $\mathbb{F}_2[u]/\langle u^{11} + u^2 + 1\rangle$;

$$y^2 + (u^{10} + u^8 + u^7 + u^4 + u^3 + u^2 + u)xy + u^7 y + x^3 + (u^8 + u^5 + u^4 + u^2 + 1)x^2 + (u^8 + u^5 + u^2)x + (u^{11} + u^{10} + u^9 + u^8 + u^7 + u^6 + u^4 + u^2 + u + 1)$$

where $\mathbb{F}_{2^{12}}$ is represented as $\mathbb{F}_2[u]/\langle u^{12} + u^7 + u^6 + u^5 + u^3 + u + 1\rangle$.

References

1. Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.
2. Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.
3. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar-Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.
4. Seigo Arita. An addition algorithm in Jacobian of C_{ab} curves. *Discrete Applied Mathematics*, 130(1):13–31, 2003. The 2000 Com2MaC Workshop on Cryptography. URL: https://www.sciencedirect.com/science/article/pii/S0166218X02005863, doi:https://doi.org/10.1016/S0166-218X(02)00586-3.
5. Daniel Augot, Morgan Barbier, and Alain Couvreur. List-decoding of binary Goppa codes up to the binary Johnson bound. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 229–233. IEEE, 2011. doi:10.1109/ITW.2011.6089384.
6. Peter Beelen, Tom Høholdt, Johan Sebastian Rosenkilde Nielsen, and Yingquan Wu. On rational interpolation-based list-decoding and list-decoding binary Goppa codes. *IEEE Trans. Inf. Theory*, 59(6):3269–3281, 2013. doi:10.1109/TIT.2013.2243800.
7. Peter Beelen and Vincent Neiger. Faster list decoding of AG codes. *IEEE Trans. Inf. Theory*, 71(5):3397–3408, 2025. doi:10.1109/TIT.2025.3550750.
8. Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast encoding of AG codes over c_{ab} curves. *IEEE Transactions on Information Theory*, 67(3):1641–1655, 2021. doi:10.1109/TIT.2020.3042248.
9. Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast decoding of AG codes. *IEEE Transactions on Information Theory*, 68(11):7215–7232, 2022. doi:10.1109/TIT.2022.3188843.
10. Daniel J. Bernstein. List decoding for binary Goppa codes. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2011. doi:10.1007/978-3-642-20901-7_4.
11. Daniel J. Bernstein. Simplified high-speed high-distance list decoding for alternant codes. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th In-*

ternational Workshop, PQCrypto 2011, pages 200–216, Tapei, Taiwan, November 29 – December 2 2011. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-25405-5_13.

12. Daniel J. Bernstein and Tung Chou. CryptAttackTester: high-assurance attack analysis. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 141–182, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68391-6_5.
13. Daniel J. Bernstein, Tanja Lange, and Alex Pellegrini. Higher-genus McEliece, 2025. Latest full version of this paper. URL: https://eprint.iacr.org/2025.
14. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings*, volume 5299 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2008. doi:10.1007/978-3-540-88403-3_3.
15. Daniel J. Bernstein and Edoardo Persichetti. Towards KEM unification. Cryptology ePrint Archive, Report 2018/526, 2018. URL: https://eprint.iacr.org/2018/526.
16. Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 893–902, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM. doi:10.1137/1.9781611974331.ch64.
17. Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 61–90, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-36033-7_3.
18. Henri Cohen. *Advanced topics in computational number theory*. Springer, 2000. URL: https://api.semanticscholar.org/CorpusID:118279568.
19. Alain Couvreur, Irene Marquez Corbella, and Ruud Pellikaan. Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes. *IEEE Trans. Inf. Theory*, 63(8):5404–5418, 2017. doi:10.1109/TIT.2017.2712636.
20. Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. Cryptanalysis of public-key cryptosystems that use subcodes of algebraic geometry codes. In Raquel Pinto, Paula Rocha Malonek, and Paolo Vettori, editors, *Coding Theory and Applications, 4th International Castle Meeting, ICMCTA 2014, Palmela Castle, Portugal, September 15-18, 2014*, volume 3 of *CIM Series in Mathematical Sciences*, pages 133–140. Springer, 2014. doi:10.1007/978-3-319-17296-5_13.
21. Alain Couvreur, Rocco Mora, and Jean-Pierre Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part IV*, volume 14441 of *Lecture Notes in Computer Science*, pages 3–38, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore. doi:10.1007/978-981-99-8730-6_1.
22. Alain Couvreur and Hugues Randriambololona. Algebraic geometry codes and some applications. In W. Cary Huffman, Jon-Lark Kim, and Patrick Solé, editors,

A Concise Encyclopedia of Coding Theory, page 998. Chapman and Hall/CRC, 2021. URL: https://hal.archives-ouvertes.fr/hal-02931167.
23. Mario A. De Boer. *Codes: their parameters and geometry*. PhD thesis, Technische Universiteit Eindhoven, 1997. URL: https://pure.tue.nl/ws/portalfiles/portal/1609368/492527.pdf.
24. Felix Delgado. The symmetry of the Weierstrass generalized semigroups and affine embeddings. *Proceedings of the American Mathematical Society*, 108(3):627–631, 1990. URL: http://www.jstor.org/stable/2047780, doi:10.1090/S0002-9939-1990-0990420-2.
25. Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. Cryptology ePrint Archive, Report 2010/331, 2010. URL: https://eprint.iacr.org/2010/331.
26. Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_14.
27. Cédric Faure and Lorenz Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic codes. In *Proceedings of the 11th international workshop on Algebraic and Combinatorial Coding Theory, ACCT*, volume 2008, pages 99–107, 2008. http://www.moi.math.bas.bg/acct2008/b17.pdf.
28. Gerhard Frey and Tony Shaska. Curves, Jacobians, and cryptography, 2018. URL: https://arxiv.org/abs/1807.05270.
29. Steven D. Galbraith, Sachar M. Paulus, and Nigel P. Smart. Arithmetic on superelliptic curves. *Math. Comput.*, 71(237):393–405, 2002. doi:10.1090/S0025-5718-00-01297-7.
30. Shuhong Gao. A new algorithm for decoding Reed-Solomon codes. In Vijay K. Bhargava, H. Vincent Poor, Vahid Tarokh, and Seokho Yoon, editors, *Communications, Information and Network Security*, pages 55–68. Springer US, 2003. doi:10.1007/978-1-4757-3789-9_5.
31. Amaldo G. Garcia and Henning Stichtenoth. Algebraic function fields over finite fields with many rational places. *IEEE Trans. Inf. Theory*, 41(6):1548–1563, 1995. doi:10.1109/18.476212.
32. Valery D. Goppa. A new class of linear error correcting codes. *Problemy Peredachi Informatsii*, 60:24–30, 1970.
33. Ryuichi Harasawa and Joe Suzuki. Fast Jacobian group arithmetic on C_{ab} curves. In *Algebraic Number Theory*, volume 1838 of *Lecture Notes in Computer Science*, pages 359–376. Springer, 2000. doi:10.1007/10722028_21.
34. Florian Hess. Computing Riemann-Roch spaces in algebraic function fields and related topics. *J. Symb. Comput.*, 33(4):425–445, 2002. doi:10.1006/jsco.2001.0513.
35. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. Cryptology ePrint Archive, Report 2017/604, 2017. URL: https://eprint.iacr.org/2017/604.
36. Heeralal Janwa and Oscar Moreno. McEliece public key cryptosystems using algebraic-geometric codes. *Designs, Codes and Cryptography*, 8(3):293–307, 1996. doi:10.1023/A:1027351723034.
37. Sabira El Khalfaoui, Mathieu Lhotel, and Jade Nardi. Goppa–like AG codes from $C_{a,b}$ curves and their behavior under squaring their dual. *IEEE Transactions on Information Theory*, 70(5):3330–3344, 2024. doi:10.1109/TIT.2023.3334096.

38. Kamal Khuri-Makdisi. Linear algebra algorithms for divisors on an algebraic curve. *Math. Comput.*, 73(245):333–357, 2004. doi:10.1090/S0025-5718-03-01567-9.
39. Neal Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, October 1989. doi:10.1007/BF02252872.
40. Kwankyu Lee, Maria Bras-Amorós, and Michael E. O'Sullivan. Unique decoding of plane AG codes via interpolation. *IEEE Trans. Inf. Theory*, 58(6):3941–3950, 2012. doi:10.1109/TIT.2012.2182757.
41. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.
42. Lorenz Minder. *Cryptography based on error correcting codes.* PhD thesis, École Polytechnique Fédérale de Lausanne (Switzerland), 2007. URL: http://infoscience.epfl.ch/record/104417.
43. Shinji Miura. Algebraic geometric codes on certain plane curves. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 76(12):1–13, 1993. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ecjc.4430761201, doi:10.1002/ecjc.4430761201.
44. Gábor P. Nagy and Sabira El Khalfaoui. Towards the security of McEliece's cryptosystem based on Hermitian subfield subcodes, 2021. URL: https://www.spsl.nsc.ru/FullText/konfe/S2021.pdf#page=176.
45. H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2):159–166, 1986.
46. Johan S. R. Nielsen and Peter Beelen. Sub-quadratic decoding of one-point Hermitian codes. *IEEE Transactions on Information Theory*, 61(6):3225–3240, 2015. doi:10.1109/TIT.2015.2424415.
47. Alex Pellegrini. Code-based cryptography and AG codes, 2022. Talk at DIAMANT Symposium Autumn 2022. URL: https://drive.google.com/file/d/1RDT455gEV2WQQcWzKZE9JbkG7tgYzMMg/view.
48. Alex Pellegrini. Code-based cryptography and AG codes, 2023. Talk at COGNAC 2023. URL: https://www.cirm-math.fr/RepOrga/2803/Slides/Pellegrini_slides.pdf.
49. Edoardo Persichetti. *Improving the efficiency of code-based cryptography.* PhD thesis, University of Auckland, 2012. URL: https://www.math.auckland.ac.nz/~sgal018/EdoardoPhD.pdf.
50. Hugues Randriambololona. The syzygy distinguisher. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part VI*, volume 15606 of *Lecture Notes in Computer Science*, pages 324–354. Springer, 2025. doi:10.1007/978-3-031-91095-1_12.
51. Avinash Sathaye. On planar curves. *American Journal of Mathematics*, 99(5):1105–1135, 1977. URL: http://www.jstor.org/stable/2374003, doi:10.2307/2374003.
52. Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Inf. Theory*, 46(4):1193–1203, 2000. doi:10.1109/18.850662.
53. Ba-Zhong Shen. *Algebraic-geometric codes and their decoding algorithm.* Phd thesis, Technische Universiteit Eindhoven, 1992. doi:10.6100/IR381307.
54. Henning Stichtenoth. On the dimension of subfield subcodes. *IEEE Transactions on Information Theory*, 36(1):90–93, 1990. doi:10.1109/18.50376.

55. Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008. doi:10.1007/978-3-540-76878-4.
56. Tianrui Wang, Anyu Wang, and Xiaoyun Wang. Exploring decryption failures of BIKE: New class of weak keys and key recovery attacks. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 70–100, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38548-3_3.
57. Christian Wieschebrink. An attack on a modified Niederreiter encryption scheme. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 14–26, New York, NY, USA, April 24–26, 2006. Springer Berlin Heidelberg, Germany. doi:10.1007/11745853_2.
58. Christian Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In Nicolas Sendrier, editor, *The Third International Workshop on Post-Quantum Cryptography, PQCRYPTO 2010*, pages 61–72, Darmstadt, Germany, May 25–28, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-12929-2_5.
59. Michael Wirtz. On the parameters of Goppa codes. *IEEE Transactions on Information Theory*, 34(5):1341–1343, 1988. doi:10.1109/18.21264.
60. Wei Zhou and George Labahn. Computing column bases of polynomial matrices. *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 379–386, 06 2013. doi:10.1145/2465506.2465947.