# Extending Distinguishing to Key Recovery for Subfield Subcodes of GRS codes

a preprint

Kirill Vedenev*

November 13, 2026

**Abstract**

The McEliece cryptosystem based on binary Goppa codes has resisted both classical and quantum cryptanalysis for nearly 50 years. Recently, Ghoshal, Ishai, Jain, and Sun proposed an efficient algorithm that distinguishes generator matrices of subcodes of generalized Reed–Solomon (GRS) codes, including Goppa codes, from random ones. From all columns but one, this distinguisher finds a set of multivariate polynomials, called a hold-out kernel, that must vanish at the held-out column if the tested code is indeed a subcode of a GRS code.

In this paper, we propose a full key-recovery attack working essentially from a single run of this distinguisher. The attack takes a relatively small sample of polynomials from the hold-out kernel and recovers a local jet (i.e., a truncated Hasse expansion up to reparametrization and scaling) of the hidden curve associated with the code. This jet can then be used in one of two finishing routes proposed in the paper. The first route uses the jet to obtain the quadratic forms vanishing on the curve and recover the tangent planes at all columns. From them, it builds a special auxiliary code, called the minor code, which turns out to be a full GRS code with the same secret support, allowing the standard Sidelnikov–Shestakov attack to recover it. The second route seeks two special quadratic forms and recovers the support from them directly. From this support, simple linear algebra recovers the rest of the secret key.

The attack relies on heuristic assumptions, mainly the stated sufficient rank conditions, which we test experimentally. We solved the McEliece challenges TII-173, TII-249, and TII-253, the first two previously unsolved. We also successfully tested the attack on randomly generated small binary Goppa codes, wild Goppa codes, alternant codes, and GRS subcodes. Under these assumptions, the cost of key recovery is dominated by the single distinguisher run (about $2^{90}$ to $2^{101}$ bit operations for Classic McEliece, excluding memory access, depending on the parameter set and the employed sparse solver).

*CryptoPro, Moscow, Russia
Southern Federal University, Rostov-on-Don, Russia
vedenevk@yandex.ru

## Contents


*1. Introduction* . . . 2
*2. Preliminaries* . . . 11
*3. The distinguisher* . . . 16
*4. Recovering local jets of the curve* . . . 24
*5. Bootstrapping* . . . 37
*6. Key recovery via the minor code* . . . 42
*7. Direct key recovery* . . . 49
*8. Experimental results* . . . 53
*9. Complexity* . . . 58
*10. Comparison with other attacks* . . . 63
*11. Conclusion* . . . 71
*References* . . . 72
*A. Derivative spaces and multiplicity shortenings* . . . 79
*B. Earlier attack routes* . . . 82


## 1. Introduction

In 1978, R. McEliece [58] proposed the first code-based public-key encryption scheme, whose security relies on the hardness of decoding general linear codes. The public key is a generator matrix $\mathbf{G}_{\text{pub}}$ of a $t$-error-correcting $[n, k]$-code $C$ from a chosen family (originally, binary Goppa codes), disguised to hide the structure enabling efficient decoding. In the original scheme, $\mathbf{G}_{\text{pub}} = \mathbf{SGP}$, where $\mathbf{G}$ is a canonical generator matrix of $C$, $\mathbf{S}$ is a random invertible matrix, and $\mathbf{P}$ is a random permutation matrix.

A message $\mathbf{m} \in \mathbb{F}_q^k$ is encrypted as $\mathbf{y} = \mathbf{m}\mathbf{G}_{\text{pub}} + \mathbf{e}$, where $\mathbf{e} \in \mathbb{F}_q^n$ is a random error vector of weight $t$. The secret key is a decoding trapdoor: an algebraic description of $C$ enabling efficient decoding (for a Goppa code, its support and Goppa polynomial). The receiver uses it to recover $\mathbf{m}$ from $\mathbf{y}$, whereas an adversary must decode a general linear code or recover a decoding trapdoor from the public matrix.

Niederreiter [64] later proposed an equivalent dual variant with a disguised parity-check matrix $\mathbf{H}_{\text{pub}}$ of $C$ as the public key. The message is encoded as a weight-$t$ error vector $\mathbf{e}$, and the ciphertext is the syndrome $\mathbf{H}_{\text{pub}}\mathbf{e}^{\mathsf{T}}$. Classic McEliece [3], a key encapsulation mechanism recently standardized by ISO, uses this formulation with binary Goppa codes.

For the code-based schemes, generic message-recovery attacks rely mainly on information-set decoding, introduced by Prange [65] and developed in [14, 30, 56, 57, 74]. The classical cost estimates for generic decoding have changed comparatively little over several decades [4], so this relative stability is an advantage of code-based cryptography over lattice-based cryptography.

The history of key-recovery attacks is less reassuring for encryption schemes based on algebraic codes. The large public keys of Goppa-based schemes prompted many attempts to replace Goppa codes with more compact alternatives. Structural attacks broke many of the proposals based on algebraic codes, including generalized Reed–Solomon codes [64,

73], Reed–Muller codes [19, 60, 72], algebraic geometry codes [24], twisted Reed–Solomon codes [51], BCH codes [32], concatenated codes [71] and some variants of generalized concatenated codes [66], polar codes [11], convolutional codes [48], quasi-cyclic and quasi-dyadic alternant codes [12, 13, 34–36], and wild Goppa codes over quadratic extensions [26].

Other variants kept the broken codes, mostly generalized Reed–Solomon codes, and improved the hiding of the code. They published random subcodes [17], inserted random columns [82], replaced the permutation matrix by a more general invertible matrix [8], or mixed the code columns with random columns [78]. Other variants used subspace subcodes [16, 18] and their modifications [42]. IKKR-type schemes added a second public matrix that introduces a decodable error of large weight [43, 47, 50]. Later attacks broke many of these variants as well [22, 23, 27, 28, 49, 52, 76, 77, 81]. Among the schemes based on algebraic codes, the original binary Goppa construction remained a notable exception at cryptographic parameter sizes for almost fifty years.

Recently, Ghoshal, Ishai, Jain, and Sun [37] proposed a distinguisher for subcodes of GRS codes, a family that includes random GRS subcodes, alternant codes, and Goppa codes. Its estimated cost for the Classic McEliece parameter sets fell well below that of information-set decoding.

The distinguisher exploits the fact that, for a subcode of a GRS code, each row of the public generator matrix consists of evaluations of a polynomial, with the same support and multipliers across all rows. Writing these polynomials as $f_1, \dots, f_k$, the public columns therefore lie, up to scalars, on the hidden curve $\mathbf{F}(Z) = (f_1(Z), \dots, f_k(Z))^{\mathsf{T}}$. Neither these polynomials nor the support values are known. The distinguisher leaves out one public column and uses linear equations to find homogeneous polynomials $h$ that vanish with prescribed multiplicity at the remaining columns. These conditions give roots of the univariate polynomial $h(\mathbf{F}(Z))$. If the number of roots, counted with multiplicity, exceeds its degree, then $h(\mathbf{F}(Z))$ is identically zero. By homogeneity, $h$ also vanishes at the held-out column. For a uniformly random matrix, the held-out column is independent of the columns used to construct $h$, so this additional vanishing is not forced. Testing it gives the distinguisher.

The first version of [37] left open whether the secret key can be recovered from these polynomials. Three independent lines of work have since addressed this question. In this paper, we develop key-recovery attacks based on recovering local jets of the hidden curve from the polynomials of the distinguisher. The second revision of [37] added a different key-recovery attack based on the ideas of the Sidelnikov–Shestakov attack [73], for which Saarinen [68, 70] gave cost estimates and a practical implementation. Weis [80] proposed two further attacks, an optimized parameter selection for the distinguisher, and a cost analysis of the sparse linear algebra at its core.

### 1.1. Contribution

We present two key-recovery attacks against subcodes of GRS codes, including alternant and Goppa codes. Both require one set of polynomials from the hold-out kernel of Ghoshal et al. [37] at a single position.

Our key-recovery pipeline consists of a local jet recovery phase, for which we also derive refinements for binary Goppa codes, followed by a choice between two independent finishing routes (Figure 1):

- *Local jet recovery.* We recover a *local jet*: the first coefficients of a Hasse expansion of the hidden curve at the support element of the held-out column, after an unknown reparametrization and local scaling. For subfield subcodes, the hold-out polynomials vanish on the Frobenius conjugate branches of the curve. We use quadratic equations to separate these branches and then extend the jet of one branch by solving recursive linear equations. We prove sufficient conditions for recovery in both the non-subfield and subfield cases.

- *Local jet recovery for binary Goppa codes.* By exploiting the differential structure of binary Goppa codes, we prove that every jet of order one extends to a normalized jet, whose even-order coefficients are coordinatewise squares of earlier ones. We use this relation to provide additional equations that strengthen both the initial branch separation and the recursive recovery via backward solving. We prove sufficient conditions for this refined procedure as well.

- *Route 1: Minor-code attack.* We bootstrap the local jet to recover, under a rank condition, aligned tangent planes at all public columns. From these planes, we construct an auxiliary *minor code*, which we prove is contained in a GRS code on the original support. In characteristic two, taking entrywise square roots reduces the dimension bound. When the minor code equals the ambient GRS code, as our fullness heuristic predicts, the Sidelnikov–Shestakov attack recovers an equivalent support, provided its dimension and codimension are at least two.

- *Route 2: Direct support recovery.* From a deeper local jet, we seek two low-degree homogeneous forms that vanish on the curve at the held-out support element to adjacent orders and whose quotient yields a non-degenerate Möbius transformation of the support. For binary square-free Goppa codes, adjacent vanishing orders are obstructed. We instead use even vanishing orders differing by two and take the square root of the quotient. We prove a sufficient condition for the required quadratic forms to exist in this case.

*Implementation and experimental results.* We implemented both attacks end-to-end and recovered full secret keys for three public McEliece challenges published by the Technology Innovation Institute (TII) [45]: TII-173, TII-249, and TII-253. To our knowledge [9, 20, 40, 69], TII-173 and TII-249 had not previously been solved. TII-253 was also solved by Weis [80] with a different attack. We also tested end-to-end recovery on randomly generated binary Goppa codes, wild Goppa codes over $\mathbb{F}_4$, random alternant codes, and non-subfield GRS subcodes. Key recovery succeeded on every code tested.

Since computing the hold-out polynomials is out of reach for large codes, we also tested the finishing steps in synthetic experiments, including at shortened Classic McEliece sizes. These steps succeeded on every code tested. The results support the feasibility of key recovery at these sizes, conditional on obtaining hold-out polynomials satisfying

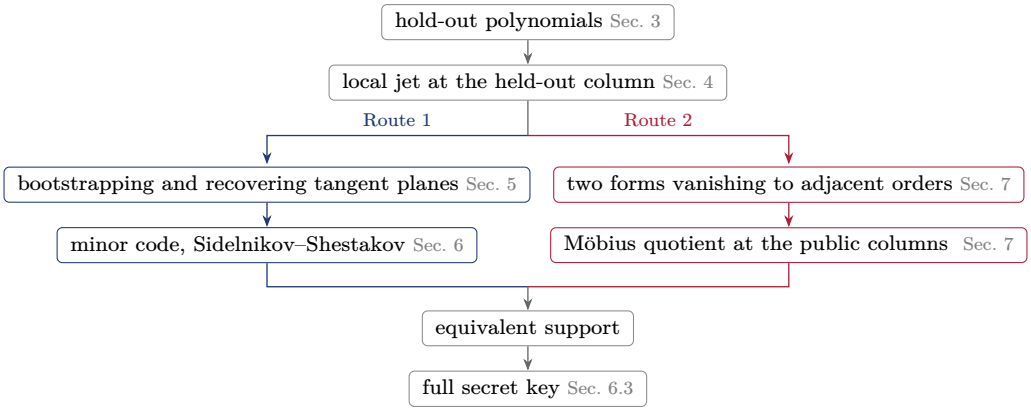



Figure 1: The two key-recovery routes.

our jet-recovery conditions and, for the minor-code attack, on the rank condition of bootstrapping.

*Complexity on Classic McEliece.* Under the assumptions stated in Section 9, full key recovery costs essentially no more than one hold-out computation supplying the required polynomials. Using a block Wiedemann sampler for the hold-out polynomials, we estimate about $2^{92.7}$ to $2^{101.2}$ Boolean gates for the whole key recovery on the Classic McEliece parameter sets, excluding memory access.

We also discuss the relation of local jets to multiplicity shortenings and earlier filtration attacks.

### 1.2. Technical overview

Recall that for a GRS subcode, each row of the public generator matrix consists of evaluations of a polynomial of degree at most $D$, with common support and multipliers across the rows. Writing these polynomials as $f_1, \dots, f_k$, the $n$ public columns have the form $\mathbf{y}_j = \lambda_j \mathbf{F}(\alpha_j)$, where $\mathbf{F} = (f_1, \dots, f_k)^{\mathsf{T}}$. The columns are known, while the curve $\mathbf{F}$, the support values $\alpha_j$, and the multipliers $\lambda_j$ are hidden.

1. **Computing the vanishing polynomials.** The distinguisher of Ghoshal et al. [37] constructs homogeneous polynomials $h$ of degree $d$ that vanish identically on the hidden curve, meaning $h(\mathbf{F}(Z)) \equiv 0$. To find them, it leaves out a public column $\mathbf{y}_i$ and solves linear equations imposing a zero of multiplicity $s \geq 2$ on the multivariate polynomial $h$ at every other column. For each retained position $j \neq i$, this guarantees

$$(Z - \alpha_j)^s \mid h(\mathbf{F}(Z)).$$

   The polynomial $h(\mathbf{F}(Z))$ has degree at most $dD$, so $s(n-1) > dD$ forces it to be identically zero. For binary Goppa codes, total multiplicity $dD + 1 - t$ suffices under Assumption 1, where $t$ is the degree of the Goppa polynomial.

2. **Recovering a local jet** (Section 4). We begin the local reconstruction at the omitted, or *held-out*, column $\mathbf{y}_i$, since the imposed multiplicities make $\nabla h(\mathbf{y}_j) = 0$ at every other public column. Differentiating $h(\mathbf{F}(Z)) \equiv 0$ at $\alpha_i$ gives

$$\nabla h(\mathbf{y}_i)\, \mathbf{F}'(\alpha_i) = 0.$$

Since $h$ and $\mathbf{y}_i$ are known, these equations give a computable linear system for tangent information. Homogeneity also makes $\mathbf{y}_i$ a solution, so the object we seek is the tangent plane spanned by the public column and a tangent vector.

For subfield subcodes, the gradient equations have coefficients in the base field $\mathbb{F}_q$, while the tangent vectors lie in the extension field containing the support. Raising the coordinates of any solution to $q$-th powers gives another solution. Consequently, the solution space contains all Frobenius conjugates of the desired tangent plane and mixtures of their directions.

We separate these planes using the second-order terms of the identities $h(\mathbf{F}(Z)) \equiv 0$. At this order, the candidate tangent vector enters quadratically, and the next unknown coefficient is multiplied by $\nabla h(\mathbf{y}_i)$. We eliminate that coefficient by taking linear combinations of the $h$ so that their gradients cancel at $\mathbf{y}_i$, leaving quadratic conditions on the tangent direction. Under the conditions of Section 4, their solution set is the union of the conjugate tangent planes.

Any Frobenius conjugate of the hidden key is an equivalent key, so we can continue with any one of these planes. We use $\mathbf{F}$ and $\alpha_j$ for the corresponding curve and support.

After fixing a tangent direction in one of these planes, we recover successive coefficients of a local expansion from the higher-order equations. With the preceding coefficients fixed, the new coefficient enters linearly. For subfield subcodes, adding a tangent direction from another conjugate branch to a solution still satisfies these equations. We therefore use equations one order higher, where gradient cancellation eliminates the following coefficient and leaves further linear constraints on the current one. Under the continuation conditions of Section 4, the result is a *local jet*

$$\mathbf{V}(T) = \mathbf{y}_i + \mathbf{v}_1 T + \cdots + \mathbf{v}_r T^r,$$

representing the first $r+1$ coefficients of the curve after an unknown invertible change of parameter and nonvanishing local scaling.

For binary square-free Goppa codes, every first-order jet of a branch extends to a normalized expansion with $\mathbf{v}_{2r} = \mathbf{v}_r^{\circ 2}$ for $r \geq 1$, where squaring is coordinatewise (Section 4.6). At order two, this gives branch-separation equations from every hold-out polynomial, without gradient cancellation. When recovering a higher odd coefficient, the next even coefficient is already known, so the equations at that next order give further linear constraints from all the hold-out polynomials. We call this *backward solving*.

3. With the jet now known, we use it as data in a new linear system, whose unknowns

are the coefficients of a homogeneous form $Q$. Requiring the first $r+1$ coefficients of $Q(\mathbf{V}(T))$ to vanish imposes a root of multiplicity at least $r+1$ on $Q(\mathbf{F}(Z))$ at $\alpha_i$. The unknown change of parameter and local scaling preserve this multiplicity. Thus we can prescribe a root at the hidden support element by solving linear equations, and both recovery routes use this ability.

- **Route 1: bootstrapping and the minor code** (Sections 5 and 6). To recover the missing tangent planes, the first route uses the jet to construct new quadratic forms $Q$ satisfying $Q(\mathbf{F}(Z)) \equiv 0$. At the retained columns we impose only $Q(\mathbf{y}_j) = 0$; we no longer require the gradients to vanish there. The additional multiplicity needed to force vanishing on the curve comes from the known jet.

  We require a quadratic form $Q$ to vanish at all $n$ public columns and to order at least $r+1$ along the jet. The polynomial $Q(\mathbf{F}(Z))$ then has degree at most $2D$, while the public columns and the jet give $n+r$ roots counted with multiplicity: one at each column, with $r$ additional copies at $\alpha_i$. Hence $n+r > 2D$ forces $Q(\mathbf{F}(Z)) \equiv 0$.

  Solving these conditions over the field of the ambient GRS code recovers all quadratic forms that vanish identically on the selected branch. Allowing coefficients in this field lets the new forms distinguish one branch from its Frobenius conjugates. Under the rank condition of Section 5, their gradients recover the tangent planes at all public columns. These planes are aligned because they all come from the branch selected by the single jet.

  *The minor code (Section 6).* We turn these tangent planes into an evaluation code by choosing a basis of each plane consisting of the public column $\mathbf{y}_j$ and a second vector $\hat{\mathbf{y}}_j$. The $2 \times 2$ minors form column $j$ of a new matrix, with entries

  $$y_{a,j}\hat{y}_{b,j} - y_{b,j}\hat{y}_{a,j} = \rho_j(f_a f_b' - f_b f_a')(\alpha_j), \qquad \rho_j \neq 0.$$

  Changing the second basis vector multiplies all entries in that column by the same nonzero scalar. This changes the column multiplier $\rho_j$, while every row still consists of evaluations at the same support values $\alpha_j$.

  In characteristic two, $f_a f_b' - f_b f_a' = (f_a f_b)'$ contains only even powers and is a square. We therefore take entrywise square roots to lower the degree bound in characteristic two, while in odd characteristic we use the original minor matrix. The row span of the resulting matrix, called the *minor code*, is contained in a GRS code on the hidden support. Our fullness heuristic predicts equality. When this equality holds, the Sidelnikov–Shestakov attack recovers an equivalent support, provided the code's dimension and codimension are at least two.

- **Route 2: direct support recovery** (Section 7). The second route recovers the support from a quotient of two quadratic forms. We seek quadratic forms $R, S$ for which $R(\mathbf{F}(Z))$ and $S(\mathbf{F}(Z))$ vanish at $\alpha_i$ to exact orders $2D$ and $2D-1$. The vanishing conditions are linear, with nonvanishing checks to ensure that the orders are exact. When the required forms exist, the degree bound $2D$ leaves only a constant

factor for the first polynomial and a linear factor for the second:

$$R(\mathbf{F}(Z)) = c_R(Z-\alpha_i)^{2D}, \qquad S(\mathbf{F}(Z)) = (Z-\alpha_i)^{2D-1}(aZ+b).$$

The common power cancels in their quotient. Evaluating at the other public columns also cancels the column multipliers, giving

$$\frac{R(\mathbf{y}_j)}{S(\mathbf{y}_j)} = \frac{c_R(\alpha_j-\alpha_i)}{a\alpha_j+b} \quad \text{for } j \neq i.$$

The exact orders ensure that this is a non-degenerate Möbius transformation, so its values give an equivalent support. The held-out support value is mapped to zero, and a zero denominator at another column is interpreted as infinity.

For binary square-free Goppa codes with nonconstant Goppa polynomial $G$ and every quadratic form $Q$, the polynomial $U = Q(\mathbf{F}(Z))$ satisfies $G^2 \mid U'$. If $U$ vanishes at $\alpha_i$ to order at least $2D-2$, differentiation in characteristic two preserves the factor $(Z-\alpha_i)^{2D-2}$ and gives degree at most $2D-2$. Thus $U'$ is a scalar multiple of this factor. Since $G(\alpha_i) \neq 0$, the divisibility forces $U' = 0$, so $U$ is a square.

We therefore use exact orders $2D$ and $2D-2$, giving

$$S(\mathbf{F}(Z)) = (Z-\alpha_i)^{2D-2}(aZ+b)^2.$$

Taking the square root of the quotient now gives $\sqrt{c_R}(Z-\alpha_i)/(aZ+b)$, a Möbius transformation. Proposition 17 gives a sufficient condition for the required forms to exist.

4. **Completing the key** (Section 6.3). To recover the multiplier, we require the rescaled public rows to satisfy the Reed–Solomon parity checks on the recovered support. This gives linear equations in the inverse multipliers. If preliminary shortening was used to reduce the initial computation, the completion procedure of Section 6.3 reconstructs the removed support elements and the full key under its stated conditions.

### 1.3. Related works

*Schur products and filtrations.* Most structural attacks on the variants based on GRS codes use the component-wise product of codes. The square of a GRS code of dimension $k$ is a GRS code of dimension $2k-1$, whereas the square of a random code of the same dimension usually has dimension $\min\{n, \binom{k+1}{2}\}$. This gap distinguishes GRS codes and their large subcodes from random codes. Wieschebrink [81] used it against the subcode variant of [17]: the square of a random subcode is usually the square of the ambient GRS code, and the Sidelnikov–Shestakov attack recovers the latter. Couvreur et al. [28] turned the product into a filtration attack. Shortening and products give a nested sequence of subcodes of the GRS code, from which the support and the multipliers are read off without the Sidelnikov–Shestakov attack. Filtrations also broke wild Goppa codes over quadratic extensions [26] and algebraic geometry codes and their subcodes [24].

*Algebraic modelling.* Faugère, Otmani, Perret and Tillich [34] wrote key recovery for alternant codes as a system of polynomial equations in the support and the multipliers. For quasi-cyclic and quasi-dyadic codes, the symmetry reduces the number of unknowns, and Gröbner bases solve the system; this broke the compact-key variants [34, 35]. Without such symmetry the system is out of reach, but its linearization gave the first distinguisher for alternant and Goppa codes [33]: when the rate is high, as in the CFS signature scheme, the linearized system has a smaller rank than for a random code. Later, Mora and Tillich [63] explained this rank defect by a tight upper bound on the dimension of the square of the dual of an alternant or Goppa code.

*Quadratic relations of the dual code.* The later line of attacks against alternant codes starts from quadratic relations in the dual code. Over the large field, the dual contains GRS codes, whose codewords satisfy relations such as $\mathbf{c}_1 \star \mathbf{c}_3 = \mathbf{c}_2 \star \mathbf{c}_2$, where $\star$ is the componentwise product. Couvreur, Mora and Tillich [25] collect all relations $\sum_{i \le j} c_{ij}\, \mathbf{v}_i \star \mathbf{v}_j = \mathbf{0}$ among products of a basis $\mathbf{v}_1, \dots, \mathbf{v}_k$ of the dual code into the code of quadratic relations. Each defines a quadratic form $Q$ in $k$ variables. The codewords above give $Q = x_i x_j - x_\ell^2$. They associate to each $Q$ the matrix $\mathbf{M}$ of the bilinear form

$$Q(\mathbf{x} + \mathbf{y}) - Q(\mathbf{x}) - Q(\mathbf{y}) = \mathbf{x}\mathbf{M}\mathbf{y}^{\mathsf{T}}.$$

This matrix is symmetric in odd characteristic and skew-symmetric with zero diagonal in characteristic two. For the form above, its rank is three in odd characteristic and two in characteristic two. Basis changes in the dual code act by congruence, so matrices of this rank exist in every basis. Finding them is a MinRank problem.

In characteristic two, they add the linear conditions defining the matrix code to the known Gröbner basis of skew-symmetric matrices of rank at most two. This modelling was later called Pfaffian. The behaviour of the Gröbner basis computation distinguishes alternant and Goppa codes from random codes at rates above $2/3$. Rank-defective matrices also reveal the polynomial structure, giving polynomial-time key recovery for alternant codes of degree below $q+1$ and Goppa codes of degree below $q-1$ distinguished by [33]. Bardet, Mora and Tillich [10] obtained polynomial-time key recovery for generic alternant codes over $\mathbb{F}_2$ and $\mathbb{F}_3$ throughout the regime of [33]. Together, the two attacks cover every distinguishable generic alternant code. Shortening and products reduce the alternant degree to three, and a Gröbner basis computation recovers the resulting code's structure.

Lemoine, Mora and Tillich [54] computed the Hilbert function of the Pfaffian modelling in degree two and determined the rates it distinguishes. Bardet, Lemoine and Tillich [9] combined this modelling with shortening and a search for squares in the ideal. Their polynomial-time attack on very-high-rate binary Goppa codes breaks the CFS signature scheme and solved TII challenges with claimed security up to 210 bits. Briaud, Lemoine, Randriambololona and Tillich [20] replaced the Pfaffian modelling by a new modelling for rank-two forms in even characteristic. Their attack has solved several previously unsolved TII challenges. They conjecture subexponential complexity and estimate at least $2^{454}$ bit operations for the smallest Classic McEliece parameter set.

*The syzygy distinguisher.* Randriambololona [67] uses finer invariants of the same

quadratic relations to distinguish alternant and Goppa codes. These are the graded Betti numbers of the homogeneous coordinate ring of a shortened dual code, which count the quadratic relations among its columns and their syzygies. The cost is subexponential in $t$, making this the first analysis of the McEliece cryptosystem to break the exponential barrier of generic decoding. The distinguisher applies to the Classic McEliece parameter sets, at an estimated cost of $2^{528}$ operations for the smallest set in [67], lowered to $2^{401}$ in its extended version, and between $2^{683}$ and $2^{834}$ for the other four sets there.

*Higher-order vanishing and tangent spaces.* This line of work also starts from the dual code. Hemmert and Wiemers [41] compute polynomials of a chosen degree vanishing to higher order at the columns of a parity-check matrix. Alternant and Goppa codes admit a larger space of such polynomials than random codes. This distinguisher applies to the Classic McEliece parameter sets at a lower estimated cost than earlier distinguishers. Hemmert [40] turned it into key recovery by searching this polynomial space for low-rank directions. The attack solved many TII challenges. Its cost at Classic McEliece parameters is conjectured to be close to the distinguisher's, about $2^{300}$ for the smallest set.

Saarinen's HOVER [69] replaces this search by computing an algebra of endomorphisms of the same polynomial space. These operators preserve linear relations among first partial derivatives, represented by their coefficient matrices. Under the paper's structural assumptions, the algebra is isomorphic to the hidden extension field. A field generator's eigenlines recover the hidden polynomial directions. HOVER solved five TII challenges, including TII-252.

Lemoine's earlier tangent space attack [53] also recovers the hidden extension field through an endomorphism algebra. For a generic alternant code, the quadrics through the parity-check columns define a variety that inherits the rational normal curve's structure by Weil restriction. Lemoine computes the algebra of operators preserving its tangent spaces at these columns. Under the paper's heuristics, this yields the underlying GRS code in polynomial time when the rate is high enough.

*Key recovery from the hold-out distinguisher.* The hold-out distinguisher of [37] also uses higher-order vanishing, but on the public code itself: the conditions are imposed at the columns of a generator matrix, not of a parity-check matrix. The second version of [37] added key recovery by extending the Sidelnikov–Shestakov attack to interpolation with multiplicities. Saarinen gave itemized bit-operation estimates for this attack on the Classic McEliece parameter sets [68] and reported solving TII-254 with a two-anchor variant [70]. Weis [80] improved the parameter selection of the distinguisher for binary Goppa codes, lowering the cost per run by about 20 bits, and proposed two attacks. His gradient attack requires 103 to 1,443 distinguisher runs. His single-run attack improves the key recovery of [37]; its cost is close to that of one distinguisher run, and it solved TII-253.

Earlier versions of this paper [75] recovered local jets from the hold-out polynomials and used them either at a few positions with guessed support values or, through tangent planes, at all positions (Appendix B). The two methods of this paper need a local jet at a single position only.

Our methods cost essentially the same as the single-run attack of Weis [80], but they are more transparent and conceptually simpler. Each step recovers a geometric object of

the hidden curve under its own sufficient condition: a local jet, then its tangent planes, then a GRS code on the original support. They are also universal: they use only that the public code is a subcode of a GRS code, and they are not tied to binary Goppa codes or to characteristic two. Their heuristic assumptions differ from those of [80]. Section 10 compares all these attacks in detail and gives their history.

### 1.4. Organization

The rest of the paper is organized as follows. Section 2 provides the necessary notation and preliminaries. Section 3 recalls the hold-out distinguisher of [37] together with its optimizations and cost [80].

In Section 4, we describe the recovery of a local jet of the hidden curve at a held-out position from the hold-out polynomials. This includes the separation of Frobenius conjugate branches for subfield subcodes, refinements for binary Goppa codes, and sufficient conditions for recovery to succeed. Section 5 describes how bootstrapping uses the jet to construct quadratic forms vanishing on the curve and recover aligned tangent planes at all public columns. In Section 6, we construct the minor code from these planes and prove that it is contained in a GRS code on the original support. We conjecture that this containment is an equality and give a heuristic argument supporting the conjecture. We then describe support recovery with the Sidelnikov–Shestakov attack, followed by recovery of the multiplier and the coordinates removed by shortening. Section 7 presents direct key recovery from a single deep jet, with the modification for binary Goppa codes.

Section 8 reports the end-to-end and synthetic experiments, and Section 9 estimates the cost of key recovery for the Classic McEliece parameter sets. Section 10 compares our attacks with the other attacks arising from the hold-out distinguisher of [37] and gives their history. Section 11 concludes the paper. Appendix A relates local jets to derivative spaces, multiplicity shortenings and filtration attacks. Appendix B describes two earlier attack routes: partial-support guessing and extended minor codes.

## 2. Preliminaries

**Notation.** $\mathbb{F}_q$ is the finite field with $q$ elements and $\mathbb{F}_{q^m}$ its degree-$m$ extension. We write $\mathbb{F}_q[x]_{<k}$ for the space of polynomials of degree less than $k$, and $\mathrm{diag}(\boldsymbol{\beta})$ for the diagonal matrix with the entries of $\boldsymbol{\beta}$ on its diagonal. A linear $[n,k,d]_q$-code is a $k$-dimensional subspace of $\mathbb{F}_q^n$ of minimum distance $d$; it corrects up to $\lfloor (d-1)/2 \rfloor$ errors.

### 2.1. Generalized Reed–Solomon codes

Let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ be a tuple of $n$ distinct elements of $\mathbb{F}_{q^m}$ and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n) \in (\mathbb{F}_{q^m}^*)^n$. For $1 \le k \le n$, the *Generalized Reed–Solomon (GRS) code* of dimension $k$ with support $\boldsymbol{\alpha}$ and multiplier $\boldsymbol{\beta}$ is

$$\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \{(\beta_1 f(\alpha_1), \dots, \beta_n f(\alpha_n)) \mid f \in \mathbb{F}_{q^m}[x]_{<k}\}.$$

The choice $\boldsymbol{\beta} = \mathbf{1}_n$ gives the Reed–Solomon code $\mathrm{RS}_k(\boldsymbol{\alpha})$. The following standard facts can be found in [55, Ch. 10]. The code $\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is an $[n, k, d = n - k + 1]$ MDS code, generated by the *Vandermonde-type* matrix

$$\mathbf{V}_k(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{bmatrix} \mathrm{diag}(\boldsymbol{\beta}).$$

Its dual is again a GRS code, $\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta})^\perp = \mathrm{GRS}_{n-k}(\boldsymbol{\alpha}, \boldsymbol{\beta}^\perp)$, with the same support and multiplier $\beta_i^\perp = \left(\beta_i \prod_{j \neq i} (\alpha_i - \alpha_j)\right)^{-1}$. In particular, a parity-check matrix of $\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the Vandermonde-type matrix $\mathbf{V}_{n-k}(\boldsymbol{\alpha}, \boldsymbol{\beta}^\perp)$.

**Remark 1.** *GRS codes are stable under affine transformations of the support and scaling of the multiplier: for any $\lambda, \nu \in \mathbb{F}_{q^m}^*$ and $\mu \in \mathbb{F}_{q^m}$,*

$$\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathrm{GRS}_k\left((\lambda\alpha_1 + \mu, \ldots, \lambda\alpha_n + \mu),\, \nu\boldsymbol{\beta}\right).$$

*Such transformations yield distinct support/multiplier pairs describing the same code.*

### 2.2. Sidelnikov–Shestakov attack

The use of GRS codes in the McEliece cryptosystem to reduce the public key size was proposed in [64]; however, Sidelnikov and Shestakov [73] described an algorithm that efficiently recovers the secret structure of these codes. Below, we briefly describe a modified Sidelnikov–Shestakov attack that, given a generator matrix $\mathbf{G}$ of a GRS code $C = \mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\beta})$ over a finite field $\mathbb{K}$, recovers an equivalent support and multiplier. Assume $2 \leq k \leq n - 2$.

Let $\mathbf{M} = [\mathbf{I}_k \mid \mathbf{R}] = (m_{i,j})$ be a systematic generator matrix of $C$. Each row of $\mathbf{M}$ has $k - 1$ zero positions. Its representing polynomial therefore has the form

$$f_i(x) = a_i \prod_{\substack{1 \leq s \leq k \\ s \neq i}} (x - \alpha_s), \qquad a_i \in \mathbb{K}^*.$$

Consequently,

$$m_{i,j} = \beta_j f_i(\alpha_j) = a_i \beta_j \prod_{\substack{1 \leq s \leq k \\ s \neq i}} (\alpha_j - \alpha_s). \tag{1}$$

For distinct $i, i' \in [\![1, k]\!]$ and distinct $j, j' \in [\![k+1, n]\!]$, division gives the relation [17, Cor. 1]

$$\frac{m_{i,j} m_{i',j'}}{m_{i',j} m_{i,j'}} = \frac{(\alpha_j - \alpha_{i'})(\alpha_{j'} - \alpha_i)}{(\alpha_j - \alpha_i)(\alpha_{j'} - \alpha_{i'})}. \tag{2}$$

The constants $a_i, a_{i'}$ and the multipliers $\beta_j, \beta_{j'}$ cancel, as do all factors indexed by $s \notin \{i, i'\}$. The four matrix entries are shown in Figure 2.

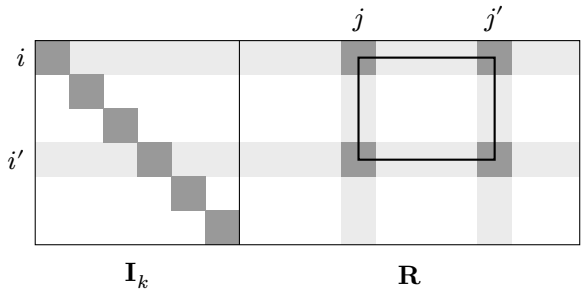



Figure 2: The four entries of $\mathbf{M} = [\mathbf{I}_k \mid \mathbf{R}]$ used in (2).

By Remark 1, we may normalize two support elements to $\alpha_1 = 0$ and $\alpha_2 = 1$. We try all values of $\alpha'_{k+1} \in \mathbb{K} \setminus \{0, 1\}$. For each trial, set $\alpha'_1 = 0$ and $\alpha'_2 = 1$, and recover the other entries from (2), with $\boldsymbol{\alpha}$ replaced by $\boldsymbol{\alpha}'$:

(i) For $j = k+2, \dots, n$, take $i = 1$, $i' = 2$, and $j' = k+1$ to recover $\alpha'_j$.

(ii) For $i = 3, \dots, k$, take $i' = 1$, $j = k+1$, and $j' = k+2$ to recover $\alpha'_i$.

After clearing denominators, each equation is linear in its one unknown. We discard a trial if an equation has no finite solution or if the recovered support has repeated entries.

Once $\boldsymbol{\alpha}'$ is found, let $\mathbf{H}_{\mathrm{RS}}$ be a parity-check matrix of $\mathrm{RS}_k(\boldsymbol{\alpha}')$. The inverse multipliers are recovered by solving the linear system

$$\mathbf{G}\,\mathrm{diag}(z_1, \dots, z_n)\mathbf{H}_{\mathrm{RS}}^{\mathsf{T}} = 0 \tag{3}$$

over $\mathbb{K}$. For a valid support, the solution space is one-dimensional and every nonzero solution has no zero coordinates. Normalize $z_1 = 1$ and set $\beta'_i = z_i^{-1}$. Indeed, (3) gives $C\,\mathrm{diag}(\mathbf{z}) \subseteq \mathrm{RS}_k(\boldsymbol{\alpha}')$; since both codes have dimension $k$, this inclusion is an equality. Hence $C = \mathrm{GRS}_k(\boldsymbol{\alpha}', \boldsymbol{\beta}')$. The normalized original support occurs among the trials, so the procedure returns an equivalent GRS description.

Reduction to systematic form costs $O(k^2 n)$ operations over $\mathbb{K}$. Each support trial takes $O(n)$ operations, and at most $|\mathbb{K}| - 2$ values are tried. Multiplier recovery requires solving a linear system with $n$ unknowns and $k(n-k)$ equations.

### 2.3. Subfield subcodes and alternant codes

**Definition 1.** For a linear $[n, k, d]$-code $C \subseteq \mathbb{F}_{q^m}^n$, its *subfield subcode* over $\mathbb{F}_q$ is $C_{|\mathbb{F}_q} = C \cap \mathbb{F}_q^n$.

For subfield subcodes, we have the following dimension and distance bounds

$$\dim_{\mathbb{F}_q}(C_{|\mathbb{F}_q}) \geq n - m(n-k), \qquad d(C_{|\mathbb{F}_q}) \geq d(C).$$

**Definition 2.** Let $1 \le r < n$, let $\boldsymbol{\alpha} \in \mathbb{F}_{q^m}^n$ be a tuple of distinct elements, and let $\boldsymbol{\beta} \in (\mathbb{F}_{q^m}^*)^n$. The *alternant code* $\mathcal{A}_r(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of degree $r$ is the subfield subcode of $\mathrm{GRS}_{n-r}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ over $\mathbb{F}_q$:

$$\mathcal{A}_r(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathrm{GRS}_{n-r}(\boldsymbol{\alpha}, \boldsymbol{\beta})_{|\mathbb{F}_q}.$$

In particular, its dimension is at least $n - mr$ and its minimum distance is at least $r + 1$.

### 2.4. Goppa codes

**Definition 3.** Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$ have distinct entries and let $G(x) \in \mathbb{F}_{q^m}[x]$ be of degree $r$ with $G(\alpha_i) \neq 0$ for all $i$. The *$q$-ary Goppa code* [38] with support $\boldsymbol{\alpha}$ and Goppa polynomial $G$ is

$$\Gamma(\boldsymbol{\alpha}, G) = \left\{ \mathbf{c} \in \mathbb{F}_q^n \;\middle|\; \textstyle\sum_{i=1}^n \frac{c_i}{x - \alpha_i} \equiv 0 \pmod{G(x)} \right\}.$$

**Proposition 1.** *Assume $r < n$. The matrix*

$$\mathbf{H} = \mathbf{V}_r(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_n \\ \beta_1\alpha_1 & \beta_2\alpha_2 & \cdots & \beta_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1\alpha_1^{r-1} & \beta_2\alpha_2^{r-1} & \cdots & \beta_n\alpha_n^{r-1} \end{bmatrix}, \qquad \beta_i = G(\alpha_i)^{-1}, \tag{4}$$

*is a parity-check matrix of $\Gamma(\boldsymbol{\alpha}, G)$ over $\mathbb{F}_{q^m}$. Hence $\Gamma(\boldsymbol{\alpha}, G) = \mathcal{A}_r(\boldsymbol{\alpha}, \boldsymbol{\beta}^\perp)$, and in particular $\dim_{\mathbb{F}_q} \Gamma(\boldsymbol{\alpha}, G) \ge n - mr$ and $d(\Gamma(\boldsymbol{\alpha}, G)) \ge r + 1$.*

Among Goppa codes, *binary square-free codes* are of particular interest, since a degree-$t$ Goppa polynomial gives minimum distance at least $2t + 1$.

**Proposition 2.** *Let $q = 2$, let $G \in \mathbb{F}_{2^m}[x]$ be square-free of degree $t$ with $G(\alpha_i) \neq 0$ for all $i$, and assume $2t < n$. Then*

$$\Gamma(\boldsymbol{\alpha}, G) = \Gamma(\boldsymbol{\alpha}, G^2),$$

*and consequently*

$$\Gamma(\boldsymbol{\alpha}, G) \subseteq \mathrm{GRS}_{n-2t}(\boldsymbol{\alpha}, \boldsymbol{\lambda}'), \qquad \lambda_i' = \frac{G(\alpha_i)^2}{\Pi_{\boldsymbol{\alpha}}'(\alpha_i)}, \qquad \Pi_{\boldsymbol{\alpha}}(x) = \prod_{j=1}^n (x - \alpha_j),$$

*so that $\dim_{\mathbb{F}_2} \Gamma(\boldsymbol{\alpha}, G) \ge n - mt$ and $d(\Gamma(\boldsymbol{\alpha}, G)) \ge 2t + 1$.*

*Wild Goppa codes* generalize square-free binary Goppa codes to non-binary alphabets.

**Remark 2.** *Let $g \in \mathbb{F}_{q^m}[x]$ be square-free of degree $t$, with $g(\alpha_i) \neq 0$ for all $i$. The code $C = \Gamma(\boldsymbol{\alpha}, g^{q-1})$ is called a* wild Goppa code. *The identity [26, Prop. 3]*

$$\Gamma(\boldsymbol{\alpha}, g^{q-1}) = \Gamma(\boldsymbol{\alpha}, g^q)$$

*gives the bounds*

$$\dim_{\mathbb{F}_q} C \ge n - m(q-1)t, \qquad d(C) \ge qt + 1.$$

*If $qt < n$, then $C$ is a subfield subcode of a GRS code of dimension $n - qt$.*

### 2.5. Hasse derivatives

Below, we recall the definition of Hasse derivatives, which will be used in the distinguisher of [37] and in our attacks.

For a multi-index $\mathbf{a} = (a_1, \dots, a_k) \in \mathbb{Z}_{\geq 0}^k$, let $|\mathbf{a}| = \sum_{i=1}^k a_i$ and $\mathbf{X}^{\mathbf{a}} = \prod_{i=1}^k X_i^{a_i}$. Let

$$\mathcal{A}_{w,k} = \{\mathbf{a} \in \mathbb{Z}_{\geq 0}^k \mid |\mathbf{a}| = w\}, \qquad \mathcal{A}_{<s,k} = \bigcup_{w=0}^{s-1} \mathcal{A}_{w,k}.$$

Then $|\mathcal{A}_{<s,k}| = \binom{k+s-1}{s-1}$.

Let $\mathbb{K}$ be a field, and let $F \in \mathbb{K}[X_1, \dots, X_k]$. For a multi-index $\mathbf{a} \in \mathbb{Z}_{\geq 0}^k$, the $\mathbf{a}$-*th Hasse derivative* $\partial^{[\mathbf{a}]} F$ is defined as the coefficient of $\mathbf{Z}^{\mathbf{a}}$ in the formal identity

$$F(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{a} \in \mathbb{Z}_{\geq 0}^k} (\partial^{[\mathbf{a}]} F)(\mathbf{X}) \mathbf{Z}^{\mathbf{a}}. \tag{5}$$

Equivalently, if $F = \sum_{\mathbf{b}} c_{\mathbf{b}} \mathbf{X}^{\mathbf{b}}$, then

$$\partial^{[\mathbf{a}]} F = \sum_{\mathbf{b} \geq \mathbf{a}} c_{\mathbf{b}} \left( \prod_{i=1}^k \binom{b_i}{a_i} \right) \mathbf{X}^{\mathbf{b}-\mathbf{a}}.$$

For a univariate polynomial $f \in \mathbb{K}[x]$ and $a \in \mathbb{K}$,

$$f(a + U) = \sum_{r \geq 0} (\partial^{[r]} f)(a) U^r. \tag{6}$$

We sometimes write $f'(x)$ in place of $(\partial^{[1]} f)(x)$ to simplify the notation. For a point $\mathbf{x} \in \mathbb{K}^k$ and an integer $s \geq 1$, the *Hasse jet of order less than* $s$ is

$$J_{\mathbf{x}}^{<s}(F) = \left( (\partial^{[\mathbf{a}]} F)(\mathbf{x}) \right)_{\mathbf{a} \in \mathcal{A}_{<s,k}}.$$

We say that $F$ *vanishes with multiplicity at least* $s$ *at* $\mathbf{x}$ if $J_{\mathbf{x}}^{<s}(F) = 0$.

The following lemmas are used in the distinguisher of [37, Sec. 2.4].

**Lemma 1.** *Let $\mathbb{K}$ be a field, and let $\phi \in \mathbb{K}[Z]$ have degree at most $D$. If $\phi$ has distinct roots $\gamma_1, \dots, \gamma_h$ of multiplicities at least $s_1, \dots, s_h$ and $\sum_i s_i > D$, then $\phi$ is the zero polynomial.*

**Lemma 2.** *Let $\mathbb{K}$ be a field, and let $H \in \mathbb{K}[X_1, \dots, X_k]$ be homogeneous of degree $d$. For every $\lambda \in \mathbb{K}^*$, $\mathbf{x} \in \mathbb{K}^k$, and $\mathbf{a} \in \mathbb{Z}_{\geq 0}^k$ with $|\mathbf{a}| \leq d$,*

$$(\partial^{[\mathbf{a}]} H)(\lambda \mathbf{x}) = \lambda^{d - |\mathbf{a}|} (\partial^{[\mathbf{a}]} H)(\mathbf{x}).$$

*Consequently, $J_{\lambda \mathbf{x}}^{<s}(H) = 0$ if and only if $J_{\mathbf{x}}^{<s}(H) = 0$.*

**Lemma 3.** *Let $\mathbb{K}$ be a field, let $H \in \mathbb{K}[X_1, \dots, X_k]$, let $f_1, \dots, f_k \in \mathbb{K}[Z]$, and fix $\gamma \in \mathbb{K}$. Put $\mathbf{p} = (f_1(\gamma), \dots, f_k(\gamma))$ and $\phi(Z) = H(f_1(Z), \dots, f_k(Z))$. If $J_{\mathbf{p}}^{<s}(H) = 0$, then $\gamma$ is a root of $\phi$ of multiplicity at least $s$.*

**Remark 3.** *For multilinear $h \in \mathbb{F}_2[X_1, \dots, X_k]$ and $\mathbf{x} \in \mathbb{F}_2^k$, let*

$$\mathcal{B}_{<s}(\mathbf{x}) = \{\mathbf{z} \in \mathbb{F}_2^k : d_H(\mathbf{x}, \mathbf{z}) < s\}.$$

*For such polynomials, Hasse-jet vanishing is equivalent to vanishing on the corresponding Hamming ball:*

$$J_{\mathbf{x}}^{<s}(h) = 0 \quad \Longleftrightarrow \quad h(\mathbf{z}) = 0 \text{ for every } \mathbf{z} \in \mathcal{B}_{<s}(\mathbf{x}). \tag{7}$$

*Indeed, identifying a subset $S \subseteq [\![1, k]\!]$ with its indicator vector $\mathbf{1}_S$, one has*

$$(\partial^{[\mathbf{1}_S]} h)(\mathbf{x}) = \sum_{T \subseteq S} h(\mathbf{x} + \mathbf{1}_T), \qquad h(\mathbf{x} + \mathbf{1}_T) = \sum_{S \subseteq T} (\partial^{[\mathbf{1}_S]} h)(\mathbf{x}).$$

## 3. The distinguisher

Ghoshal, Ishai, Jain, and Sun [37] proposed a distinguisher that, under suitable parameter conditions, distinguishes generator matrices of GRS subcodes from uniformly random matrices of the same size. It applies both to subcodes over the field of the ambient GRS code and to subfield subcodes.

Let $C \subseteq \mathbb{F}_q^n$ be an $[n, k]$-code over $\mathbb{F}_q$ that is a subcode of an ambient $[n, K = D+1]$ GRS code $\mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \subseteq \mathbb{F}_{q^m}^n$, where $\boldsymbol{\alpha} \in \mathbb{F}_{q^m}^n$ has pairwise distinct entries and $\boldsymbol{\lambda} \in (\mathbb{F}_{q^m}^*)^n$. When $m = 1$, this captures generic GRS subcodes over the base field; when $m \geq 2$, it captures subfield subcodes, including alternant codes and Goppa codes.

Let

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \\ | & | & & | \end{bmatrix} \in \mathbb{F}_q^{k \times n}$$

be a generator matrix of $C$. Since each row of $\mathbf{Y}$ belongs to $\mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, there exist polynomials $f_1, \dots, f_k \in \mathbb{F}_{q^m}[Z]$ of degree at most $D$ such that the $j$-th column of $\mathbf{Y}$ is

$$\mathbf{y}_j = \lambda_j \mathbf{F}_j, \qquad \mathbf{F}_j = \mathbf{F}(\alpha_j) = (f_1(\alpha_j), \dots, f_k(\alpha_j))^{\mathsf{T}} \tag{8}$$

for any $j \in [\![1, n]\!]$. Thus, the columns of $\mathbf{Y}$ are scalar multiples of points lying on the algebraic curve in $\mathbb{F}_{q^m}^k$ parametrized by $\mathbf{F}(Z) = (f_1(Z), \dots, f_k(Z))^{\mathsf{T}}$.

**The idea of the distinguisher.** The idea is to find nonzero homogeneous polynomials $h \in \mathbb{F}_q[X_1, \dots, X_k]$ that vanish on the hidden curve:

$$h(\mathbf{F}(Z)) \equiv 0.$$

It is enough to find some such polynomials. The distinguisher builds them using only part of the public matrix: it leaves out a small set $R$ of columns and uses all columns outside $R$ (typically, a single column $\mathbf{y}_\tau$ is held out). Denote the resulting polynomial space by $\mathcal{K}_R$, or by $\mathcal{K}_\tau$ when $R = \{\tau\}$.

The distinguisher then evaluates the polynomials in $\mathcal{K}_R$ at the held-out columns. For a GRS subcode, the conditions given below ensure that these polynomials vanish on $\mathbf{F}$. By homogeneity, they also vanish at its scaled points, so

$$h(\mathbf{y}_\tau) = 0 \qquad \text{for all } h \in \mathcal{K}_R,\ \tau \in R.$$

The matrix is accepted if all these evaluations are zero and rejected otherwise.

For a uniformly random matrix, $\mathcal{K}_R$ depends only on the retained columns, while the held-out columns remain independent uniformly random points. Acceptance therefore requires these points to be common zeros of the polynomials in $\mathcal{K}_R$. Bounds on the zeros of polynomials in the chosen space then control the probability of accepting a random matrix.

### 3.1. A sufficient condition for vanishing on the curve

Let $h$ be homogeneous of degree $d$ and put

$$g(Z) = h(\mathbf{F}(Z)) = h(f_1(Z), \dots, f_k(Z)).$$

Since each $f_i$ has degree at most $D$, we have $\deg g \le dD$. Thus, to prove $g \equiv 0$, it suffices to give more than $dD$ roots of $g$, counted with their multiplicities.

Imposing $h(\mathbf{y}_j) = 0$ for $j \notin R$ gives the $n - |R|$ distinct roots $\alpha_j$ of $g$, because $h(\mathbf{y}_j) = \lambda_j^d g(\alpha_j)$. To increase the number of roots counted with multiplicity, the distinguisher requires each of these roots to have multiplicity at least $s$, for an integer $s \ge 1$. The sufficient condition is then

$$s(n - |R|) > dD. \tag{9}$$

It remains to impose these multiplicities without knowing the support or the curve.

**Remark 4.** *Over a field of characteristic* 0, *$\alpha$ is a root of $g$ of multiplicity at least $s$ if and only if*

$$g(\alpha) = g'(\alpha) = \cdots = g^{(s-1)}(\alpha) = 0.$$

*The chain rule expresses these derivatives in terms of the partial derivatives of $h$ and the derivatives of $\mathbf{F}$. For example, writing $\mathbf{p} = \mathbf{F}(\alpha)$ gives*

$$g'(\alpha) = \sum_{i=1}^{k} \frac{\partial h}{\partial X_i}(\mathbf{p}) f_i'(\alpha).$$

*Even when the values $f_i'(\alpha)$ are unknown, we can ensure $g'(\alpha) = 0$ by requiring all first partial derivatives of $h$ to vanish at $\mathbf{p}$. The higher-order chain rule gives the same conclusion for the remaining derivatives: if all partial derivatives of $h$ of total order less*

*than $s$, including $h$ itself at order zero, vanish at $\mathbf{p}$, then*

$$g(\alpha) = g'(\alpha) = \cdots = g^{(s-1)}(\alpha) = 0.$$

*Note that requiring all these partial derivatives to vanish is a very restrictive sufficient condition, not a necessary one.*

The ordinary-derivative criterion can fail in positive characteristic. For example, in characteristic 2, the polynomial $g(Z) = Z^2$ satisfies

$$g(0) = g'(0) = g''(0) = 0,$$

but the root at zero has multiplicity only 2.

The distinguisher [37] uses Hasse derivatives in place of ordinary derivatives [37, Sec. 2.4]. Over any field, $\alpha$ is a root of $g$ of multiplicity at least $s$ if and only if

$$(\partial^{[r]} g)(\alpha) = 0, \qquad 0 \le r < s.$$

The sufficient condition from the preceding remark also holds for Hasse derivatives. Namely, for homogeneous $h$ and $\mathbf{y}_j = \lambda_j \mathbf{F}(\alpha_j)$, Lemmas 2 and 3 imply that

$$(\partial^{[\mathbf{a}]} h)(\mathbf{y}_j) = 0 \quad \text{for all } |\mathbf{a}| < s \quad \Longrightarrow \quad (Z - \alpha_j)^s \mid h(\mathbf{F}(Z)).$$

Homogeneity allows the conditions to be imposed at the scaled point $\mathbf{y}_j$. The resulting conditions $(\partial^{[\mathbf{a}]} h)(\mathbf{y}_j)$ involve only the known column and the coefficients of $h$; neither the support nor the derivatives of $\mathbf{F}$ are needed.

Fix a linear space $\mathcal{P}$ of homogeneous degree-$d$ polynomials over $\mathbb{F}_q$. Imposing these conditions at every column outside $R$ gives the *hold-out kernel*

$$\mathcal{K}_R^{\mathcal{P}}(\mathbf{Y}) = \{h \in \mathcal{P} : (\partial^{[\mathbf{a}]} h)(\mathbf{y}_j) = 0 \text{ for all } j \notin R,\ |\mathbf{a}| < s\}. \tag{10}$$

It can be computed by solving homogeneous linear equations in the coefficients of $h$. For $R = \{\tau\}$, write $\mathcal{K}_\tau^{\mathcal{P}}(\mathbf{Y})$.

Every $h$ in this kernel gives a polynomial $h(\mathbf{F}(Z))$ of degree at most $dD$ with at least $s(n - |R|)$ roots counted with multiplicity. Hence the zero-count condition gives the following result.

**Proposition 3.** *If $s(n - |R|) > dD$, then every $h \in \mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$ satisfies*

$$h(\mathbf{F}(Z)) \equiv 0.$$

*In particular, $h(\mathbf{y}_\tau) = 0$ for every $\tau \in R$.*

**Remark 5.** *A polynomial may have nonzero coefficients but vanish at every point of $\mathbb{F}_q^k$. For example, the homogeneous polynomial $X_1^q X_2 - X_1 X_2^q$ vanishes on $\mathbb{F}_q^2$, since $a^q = a$ for every $a \in \mathbb{F}_q$. Any such polynomial occurring in the hold-out kernel is a spurious nonzero solution. To exclude these solutions, restrict $\mathcal{P}$ to homogeneous $q$-*reduced *polynomials of*

*degree $d$, whose degree in each variable is less than $q$, i.e.,*

$$\mathcal{P} = \mathcal{P}^{\mathsf{red}}_{=d,q} = \operatorname{span}_{\mathbb{F}_q}\{\mathbf{X}^{\mathbf{a}} : |\mathbf{a}| = d,\ 0 \le a_i < q \text{ for all } i\}.$$

*A nonzero $q$-reduced polynomial cannot vanish at every point of $\mathbb{F}_q^k$. For $q = 2$, these are precisely the multilinear homogeneous polynomials.*

**Remark 6.** *Recall that for binary multilinear homogeneous polynomials $h$ and binary $\mathbf{y}$, the conditions $J_{\mathbf{y}}^{<s}(h) = 0$ are equivalent to vanishing on the Hamming ball of radius $s-1$ centred at $\mathbf{y}$; see (7).*

### 3.2. Nontriviality of the hold-out kernel

The coefficients of $h$ in a basis of $\mathcal{P}$ are the unknowns in (10). There are $\dim_{\mathbb{F}_q} \mathcal{P}$ such coefficients. If each retained column imposes at most $M_s$ equations, then

$$\dim_{\mathbb{F}_q} \mathcal{K}_R^{\mathcal{P}}(\mathbf{Y}) \ge \dim_{\mathbb{F}_q} \mathcal{P} - (n - |R|)M_s.$$

Consequently, a *very coarse sufficient condition* for a nonzero kernel is

$$(n - |R|)M_s < \dim_{\mathbb{F}_q} \mathcal{P}. \tag{11}$$

This equation count ignores linear dependencies. The rank of the system can be much smaller than the number of equations, so the kernel may be nonzero even when (11) fails. In [80, Sec. 3], parameters for binary Goppa codes are instead chosen using a heuristic estimate of the kernel dimension.

For $d < q$, the space $\mathcal{P}^{\mathsf{red}}_{=d,q}$ is the full homogeneous degree-$d$ space, also denoted by $\mathcal{P}^{\mathsf{hom}}_{=d}$. For $q = 2$, write $\mathcal{P}^{\mathsf{red}}_{=d} = \mathcal{P}^{\mathsf{red}}_{=d,2}$. Table 1 gives the two counts used in [37] and the general $q$-reduced count. Here $N_{q,k}(r)$ is the number of $q$-reduced monomials of degree $r$, given by

$$N_{q,k}(r) = [T^r](1 + T + \cdots + T^{q-1})^k,$$

with the coefficient taken over the integers. Only Hasse derivative indices with $a_i < q$ for every $i$ need to be counted, giving the sum in the last row.

Table 1: The number $\dim_{\mathbb{F}_q} \mathcal{P}$ of unknowns and the bound $M_s$ on the number of equations imposed by each retained column.

| | $d < q$ | $q = 2$ | General $q$ |
|---|---|---|---|
| $\dim_{\mathbb{F}_q} \mathcal{P}$ | $\binom{k+d-1}{d}$ | $\binom{k}{d}$ | $N_{q,k}(d)$ |
| $M_s$ | $\binom{k+s-1}{s-1}$ | $\sum_{r=0}^{s-1}\binom{k}{r}$ | $\sum_{r=0}^{s-1} N_{q,k}(r)$ |

In the binary case, the first count is the number of square-free monomials of degree $d$, and the second is the size of a Hamming ball of radius $s-1$.

### 3.3. The distinguishing test

Assume that $(d, s, R)$ satisfy (9) and (11) for $\mathcal{P} = \mathcal{P}^{\mathsf{red}}_{=d,q}$. On input $\mathbf{Y} \in \mathbb{F}_q^{k\times n}$, the test computes a basis of $\mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$ and outputs `structured` if every basis element vanishes at $\mathbf{y}_\tau$ for every $\tau \in R$, and `random` otherwise. By Proposition 3, a generator matrix of a code as in (8) is always accepted.

For uniform $\mathbf{Y}$, condition on the columns outside $R$. The kernel is then fixed and nonzero, while the columns $\mathbf{y}_\tau$, $\tau \in R$, remain independent and uniform on $\mathbb{F}_q^k$. Fix a nonzero $h \in \mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$. By Remark 5, its evaluation vector is a nonzero codeword of the generalized Reed–Muller code $\mathrm{RM}_q(d,k)$. Writing $d = \mu(q-1)+\nu$ with $0 \le \nu < q-1$ and assuming $d < k(q-1)$, the minimum distance $(q-\nu)q^{k-1-\mu}$ of this code [29, 44] gives

$$\Pr[\text{output } \texttt{structured}] \le \left(1 - (1-\tfrac{\nu}{q})q^{-\mu}\right)^{|R|}.$$

This equals $(d/q)^{|R|}$ for $d < q$ [37, Thm. 3.2] and $(1-2^{-d})^{|R|}$ for $q = 2$. Sharper bounds can be obtained using generalized Hamming weights of Reed–Muller codes [39, 79] to account the test for all kernel elements instead of single-polynomial estimate, or from counting low-weight codewords [1].

### 3.4. Optimizations

Ghoshal et al. [37, Sec. 5] use two optimizations for concrete parameters.

(i) *Systematic form.* Put $\mathbf{Y}$ in systematic form with respect to an information set contained in the retained columns. Let $q = 2$ and $d > s$. By (7), the conditions at a unit column $\mathbf{e}_i$ say that $h$ vanishes at every binary vector within Hamming distance $s-1$ of $\mathbf{e}_i$. These vectors have weight at most $s < d$, and a multilinear form of degree $d$ vanishes at every binary vector of weight less than $d$. Hence the conditions at the $k$ unit columns hold for every $h \in \mathcal{P}^{\mathsf{red}}_{=d}$. They add no equations to the linear system, but these columns still count in (9). The coarse sufficient condition (11) for a nonzero kernel becomes

$$(n-k-|R|)\sum_{r=0}^{s-1}\binom{k}{r} < \binom{k}{d}. \tag{12}$$

This generalizes to any $q$. At a unit column $\mathbf{e}_i$, the conditions of order less than $s$ say exactly that no monomial of $h$ contains $X_i$ with exponent greater than $d-s$. Imposing them therefore amounts to replacing $\mathcal{P}$ by its subspace spanned by the monomials with all exponents at most $d-s$. On this subspace, the conditions at the unit columns hold automatically. For $s = d-1$ this subspace is $\mathcal{P}^{\mathsf{red}}_{=d}$, and (12) applies for every $q$ [41, Lemma 2].

(ii) *Shortening.* Shortening the public code on $\ell$ information-set coordinates outside $R$ gives an $[n-\ell, k-\ell]$ code. In the multilinear case, the number of unknowns decreases from $\binom{k}{d}$ to $\binom{k-\ell}{d}$, which can greatly reduce the cost of computing the hold-out kernel.

Shortening also lowers the degree bound of the hidden curve from $D$ to $D-\ell$, so the reconstruction condition becomes

$$s(n-\ell-|R|) > d(D-\ell). \tag{13}$$

For $d > s$, increasing $\ell$ relaxes this condition.

The sufficient kernel conditions apply with $(n,k)$ replaced by $(n-\ell, k-\ell)$. In [37, Sec. 5.3], the parameters $(\ell, d, s)$ are chosen to minimize $\binom{k-\ell}{d}$ subject to the reconstruction condition and the systematic-form kernel condition.

**Further optimizations.** In [80, Sec. 3], several further optimizations were proposed, resulting in lower complexity. Two of them were transferred from [41], where similar higher-order vanishing conditions are imposed on the dual side, at the columns of a parity-check matrix. Sparsity gives a sharper cost estimate, while removing redundant derivative orders reduces the number of equations. The remaining refinements relax the parameter conditions, allowing smaller $d$ and $s$ and larger $\ell$. The first three optimizations below apply to all GRS subcodes. For them we assume that $s = d-1$ and that $\mathbf{Y}$ is in systematic form, so that $\mathcal{P} = \mathcal{P}^{\mathsf{red}}_{=d}$ by (i). The last two are specific to Goppa codes.

(iii) *Sparsity.* The complexity estimate in [80, Sec. 3] uses a sharper count of the nonzero entries of the constraint matrix. Consider a multilinear monomial of degree $d$ and a retained column $\mathbf{y}$. Suppose that $a$ of the variables in the monomial are zero at $\mathbf{y}$. A Hasse derivative can be nonzero at $\mathbf{y}$ only if all these $a$ variables are differentiated. Hence the number of nonzero entries contributed by this monomial and column is

$$\sum_{u=a}^{s-1} \binom{d-a}{u-a},$$

where the sum is zero if $a \geq s$.

For a uniformly random binary column, $a$ has distribution $\mathrm{Bin}(d, 1/2)$. The expected count is approximately 12.6 for $(d,s) = (7,6)$ and 20.6 for $(d,s) = (8,7)$, compared with the bounds 120 and 247 obtained from $\sum_{u=0}^{s-1}\binom{d}{u}$. These smaller counts reduce the estimated cost of solving the system by sparse linear algebra.

(iv) *Redundant derivative orders.* As noted above, to build the hold-out kernel we impose the conditions $(\partial^{[\mathbf{a}]}h)(\mathbf{y}) = 0$ at every retained column $\mathbf{y}$, for all $\mathbf{a}$ with $|\mathbf{a}| < s$. We call $|\mathbf{a}|$ the order of such a condition. Since $s = d-1$, the orders range from 0 to $d-2$. Some of these conditions follow from others at the same column. For example,

since $h$ is homogeneous of degree $d$, Euler's identity gives

$$\sum_{i=1}^{k} y_i \frac{\partial h}{\partial X_i}(\mathbf{y}) = d\, h(\mathbf{y}).$$

Let $p$ be the characteristic of $\mathbb{F}_q$. If $p \nmid d$, the condition $h(\mathbf{y}) = 0$ of order 0 therefore follows from the conditions of order 1. Similar identities relate other orders. By [41, Lemma 5, Remark 4], the conditions of orders $d-2$ and $d-p^\ell \geq 0$, $\ell \geq 1$, imply all the others. In characteristic 2 these are the orders $d-2, d-4, d-8, \ldots$ (for instance, for $d = 7$, only the orders 5 and 3 are sufficient). Dropping the other orders does not change the kernel, but it reduces the number of equations.

(v) *Kernel dimension.* The sufficient condition (12) ignores all dependencies between the conditions. For random matrices, Hemmert and Wiemers give a heuristic formula for the kernel dimension. Let $\tilde{n} = n - k - |R|$ be the number of retained columns that are not unit vectors. For a power series $A = \sum_{i \geq 0} a_i z^i$, let $[A]_+$ be the sum of the terms $a_i z^i$ with $a_j > 0$ for all $j \leq i$.

**Heuristic 1** ([41, Heuristic 1])**.** Suppose that $\tilde{n} > k$. If $\mathbf{Y}$ is uniformly random and $k$ is large enough, then with high probability

$$\dim_{\mathbb{F}_q} \mathcal{K}_R^{\mathcal{P}^{\mathrm{red}}_{=d}}(\mathbf{Y}) = \kappa(k, d, \tilde{n}) := [z^d]\left[(1+z)^k (1-z^2)^{\tilde{n}}\right]_+.$$

Hemmert and Wiemers derive this formula from regularity assumptions when $p = 2$ or $p > d$ and conjecture it in general. Experiments in [41] and [80] support it. For a GRS subcode, the columns of $\mathbf{Y}$ lie on a curve, and the conditions are expected to be at least as dependent as for random points. Thus $\kappa(k, d, \tilde{n})$ is also a reasonable lower estimate for the kernel dimension of a GRS subcode. The experiments with binary Goppa codes in [80] agree with this. Since the random case of the test needs only a nonzero kernel, parameters in [80] are selected by the condition $\kappa(k, d, \tilde{n}) > 0$ instead of (12).

(vi) *The point at infinity.* Let $C = \Gamma(\boldsymbol{\alpha}, G)$, where $G$ is monic and square-free of degree $t$, and use the GRS representation of Proposition 2, with $D = n - 2t - 1$. Appending the column

$$\mathbf{w} = \sum_{j=1}^{n} \mathbf{y}_j$$

to $\mathbf{Y}$ gives a generator matrix of the extended Goppa code. Its ambient GRS code retains dimension $D+1$ and gains one support position, conventionally denoted by $\infty$. Thus $\mathbf{w}$ supplies an additional evaluation column of the hidden curve [15].

In [80, Prop. 2], it is shown that imposing the order-$< s$ Hasse conditions at this additional column gives $s$ more zeros in the interpolation count. The reconstruction condition therefore becomes

$$s(n - |R| + 1) > dD. \tag{14}$$

The point $\mathbf{w}$ adds one constraint point, so $\tilde{n}$ grows by one in Heuristic 1. In [80, Sec. 3], this gain is reported to matter only for `mceliece460896`.

**Remark 7.** *The sum of the public columns also provides an additional evaluation column for nonbinary Goppa codes [15]. For a Goppa polynomial of degree $r < n$, condition (14) applies with $D = n - r - 1$. For wild Goppa codes, the representation with $G = g^q$ from Remark 2 gives $D = n - q \deg g - 1$, provided $q \deg g < n$.*

(vii) *Hidden linear constraints for binary Goppa codes.* Let $q = 2$ and $C = \Gamma(\boldsymbol{\alpha}, G)$, where $G$ is square-free of degree $t$. Let $\mathbf{F}$ come from the GRS representation of Proposition 2, so that $D = n - 2t - 1$. For $h$ in the hold-out kernel, put $\phi(Z) = h(\mathbf{F}(Z))$. Then $\deg \phi \le dD$, and $\phi$ vanishes with multiplicity at least $s$ at each $\alpha_j$ with $j \notin R$. These are $s(n - |R|)$ linear conditions on the $dD + 1$ coefficients of $\phi$, and Proposition 3 uses only them. By [80, Prop. 1], $\phi$ also satisfies

$$\Pi_{\boldsymbol{\alpha}} \phi' \equiv d \, \Pi'_{\boldsymbol{\alpha}} \phi \pmod{G}, \tag{15}$$

which gives $t$ further linear conditions. Assuming that they are independent of the vanishing conditions, so that the combined system has maximal rank, we obtain $\phi = 0$ as soon as $s(n - |R|) + t > dD$.

**Assumption 1** ([80, Sec. 3 and Thm. 3])**.** For binary Goppa codes as above, every $h \in \mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$ satisfies $h(\mathbf{F}(Z)) \equiv 0$ whenever

$$s(n - |R|) \ge dD + 1 - t.$$

For a fixed support, [80, Thm. 3] proves the assumption with high probability over the Goppa polynomial. Let $\delta = dD + 1 - s(n - |R|)$ be the number of missing zeros, so that $1 \le \delta \le t$, and let $s + d$ be odd, as it is for $s = d - 1$. If $G$ is uniformly random among the monic irreducible polynomials of degree $t \ge 2$ over $\mathbb{F}_{2^m}$, $m \ge 4$, then the assumption fails with probability at most $n \, 2^{-m(t-\delta+1)}$. Classic McEliece draws $G$ in this way, independently of the support. Including the column at infinity replaces $n - |R|$ by $n - |R| + 1$ in the assumption and in $\delta$.

### 3.5. Complexity

The cost of the distinguisher is dominated by the computation of the hold-out kernel, that is, by the solution of a homogeneous linear system with $\dim_{\mathbb{F}_q} \mathcal{P}$ unknowns. The system is sparse, and for large instances Ghoshal et al. [37, Sec. 5.3] solve it by the block Lanczos algorithm of Eberly [31]. They also give formulas for the resulting number of bit operations. Memory is not counted. Weis [80, Sec. 6] instead uses Montgomery's block Lanczos algorithm [62], computes the entries of the matrix during the run instead of storing it, and counts the operations, memory and data movement of this computation.

Table 2 gives the estimates of [37] and conditional block Lanczos estimates for uniform-multiplicity systems, using the counting convention of [80]. The latter use the optimizations

(iii)–(vii) and therefore rely on Heuristic 1 and Assumption 1. For comparison, Weis reports 94.1 to 102.4 bits using the formulas of [37] at his parameters [80, Table 3], and attributes about 11 of the roughly 20 bits gained to Assumption 1 [80, Sec. 3]. In that comparison, about three of the further 4 bits come from the solver and the organization of the sparse products, and one bit from charging one operation instead of two for each nonzero entry.

The Lanczos entries here use the cost formulas of [80, Sec. 6 and App. E] for our polynomial-coefficient kernel. They assume that a run supplies enough samples, as in [80, Assumption 4], and exclude retries, memory access and address generation. They count additions at nonzero entries under Weis's balanced-density convention. Section 9 specifies the sampling requirement for key recovery and gives estimates for minimum-zero mixed systems using a block Wiedemann sampler. Those estimates count every AND and XOR of a fixed Boolean circuit. The two tables therefore use different systems and counting conventions.

Table 2: Estimated number $T$ of bit operations for one run of the distinguisher on the Classic McEliece parameter sets, from [37, Table 2] and the conditional Lanczos calculation described above. The latter uses uniform multiplicities. Here $k_\ell$ is the dimension of the shortened code and $N = \binom{k_\ell}{d}$ is the number of unknowns. The last column is $\log_2$ of the cost of information-set decoding quoted in [37, Table 1].

| | | [37], $(d,s)=(8,7)$ | | | Lanczos, $(d,s)=(7,6)$ | | | |
|---|---|---|---|---|---|---|---|---|
| Parameter set | $(m,n,t)$ | $k_\ell$ | $\log_2 N$ | $\log_2 T$ | $k_\ell$ | $\log_2 N$ | $\log_2 T$ | ISD |
| `mceliece348864` | $(12,3488,64)$ | 217 | 46.6 | 114.2 | 154 | 38.4 | 89.9 | 151 |
| `mceliece460896` | $(13,4608,96)$ | 274 | 49.3 | 120.3 | 195 | 40.8 | 95.4 | 191 |
| `mceliece6688128` | $(13,6688,128)$ | 315 | 51.0 | 124.0 | 224 | 42.2 | 98.6 | 257 |
| `mceliece6960119` | $(13,6960,119)$ | 304 | 50.6 | 123.0 | 216 | 41.8 | 97.8 | 257 |
| `mceliece8192128` | $(13,8192,128)$ | 315 | 51.0 | 124.0 | 224 | 42.2 | 98.6 | 287 |

## 4. Recovering local jets of the curve

Let $\mathbf{Y}$ be a generator matrix as in (8), possibly after shortening, and let $\mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$ be a hold-out kernel with held-out set $R$. If (9) holds, then by Proposition 3 every $h$ in this hold-out kernel satisfies

$$h(\mathbf{F}(Z)) = 0 \tag{16}$$

identically in $Z$. For binary Goppa codes with the parameters of [80], (16) holds under Assumption 1, with the column at infinity counted. The distinguisher uses (16) only at the held-out points $Z = \alpha_j$, $j \in R$. The identity, however, holds along the whole curve, so the same polynomials carry information about the curve itself. We obtain this information from the derivatives of (16) at a held-out point.

Fix $i \in R$, let $\mathbf{p} = \mathbf{y}_i$, and put $\mathbb{K} = \mathbb{F}_{q^m}$. Let $\mathcal{H} \subseteq \mathcal{K}_R^{\mathcal{P}}(\mathbf{Y})$ be a space spanned by polynomials from the hold-out kernel (for example the whole hold-out kernel, or the vectors returned by a sparse solver), and let $h_1, \dots, h_M$ span $\mathcal{H}$.

### 4.1. Gradient equations

Our first aim is to find the tangent vector $\mathbf{F}'(\alpha_i)$ of the curve at the held-out point. The first derivative of (16) gives linear equations for it. For a polynomial $h \in \mathbb{K}[X_1, \dots, X_k]$ let

$$\nabla h = \Big( \frac{\partial h}{\partial X_1}, \dots, \frac{\partial h}{\partial X_k} \Big)$$

denote the *gradient*, i.e., the row vector of first partial derivatives. The first partial derivatives coincide with the Hasse derivatives of order one. Let $h \in \mathcal{H}$. Differentiating (16) gives

$$\nabla h(\mathbf{F}(Z))\, \mathbf{F}'(Z) = 0.$$

The partial derivatives of $h$ are homogeneous of degree $d-1$, and $\mathbf{p} = \lambda_i \mathbf{F}(\alpha_i)$ with $\lambda_i \neq 0$, so $\nabla h(\mathbf{p}) = \lambda_i^{d-1} \nabla h(\mathbf{F}(\alpha_i))$. Hence, at $Z = \alpha_i$,

$$\nabla h(\mathbf{p})\, \mathbf{F}'(\alpha_i) = 0. \tag{17}$$

The coefficients $\nabla h(\mathbf{p})$ are computed from $h$ and the public column. Let $\mathbf{J}$ be the $M \times k$ matrix with rows

$$\nabla h_1(\mathbf{p}), \dots, \nabla h_M(\mathbf{p}).$$

Taking $h = h_1, \dots, h_M$ in (17), we see that the unknown $\mathbf{F}'(\alpha_i)$ is a solution of the linear system $\mathbf{J}\mathbf{v} = \mathbf{0}$; by linearity in $h$, this system contains all equations (17) for $h \in \mathcal{H}$. We would like to find $\mathbf{F}'(\alpha_i)$ by solving it.

The system cannot determine $\mathbf{F}'(\alpha_i)$ uniquely. Besides its scalar multiples, the kernel of $\mathbf{J}$ contains the public column $\mathbf{p}$ itself: by Euler's identity $\sum_l X_l\, \partial h/\partial X_l = d\, h$ and (16),

$$\nabla h(\mathbf{p})\, \mathbf{p} = d\, h(\mathbf{p}) = d\lambda_i^d\, h(\mathbf{F}(\alpha_i)) = 0.$$

Hence

$$\operatorname{span}\{\mathbf{p}, \mathbf{F}'(\alpha_i)\} \subseteq \ker \mathbf{J}. \tag{18}$$

We assume from now on that $\mathbf{p}$ and $\mathbf{F}'(\alpha_i)$ are linearly independent, so that this span is a plane. Thus the gradient equations determine $\mathbf{F}'(\alpha_i)$ at best up to a nonzero scalar factor and an added multiple of $\mathbf{p}$. In Section 4.2 we show that this ambiguity comes from reparametrizations of the curve, and that no equation derived from (16) removes it.

**Remark 8.** *The gradient equations are useful only at held-out positions: at a retained column $\mathbf{y}_j$ and for $s \geq 2$, the conditions defining the hold-out kernel include $\nabla h(\mathbf{y}_j) = 0$. Even at a held-out column all gradients may vanish. Since $\partial^{[\mathbf{a}]}(\partial h/\partial X_l) = (a_l + 1)\, \partial^{[\mathbf{a}']} h$, where $\mathbf{a}'$ is obtained from $\mathbf{a}$ by increasing $a_l$ by one, each $\partial h/\partial X_l$ vanishes with multiplicity at least $s-1$ at every retained column. If $(s-1)(n-|R|) > (d-1)D$, the argument of Proposition 3 in degree $d-1$ gives $(\partial h/\partial X_l)(\mathbf{F}) = 0$, and hence $\mathbf{J} = 0$. Thus*

$$(s-1)(n-|R|) \leq (d-1)D$$

*is necessary for nonzero gradient equations; after shortening, $n$ and $D$ are replaced by*

*$n-\ell$ and $D-\ell$. All parameter sets of Table 2, with $|R|=1$, satisfy this condition, also when the column at infinity and the $t$ conditions (15) are counted.*

### 4.2. Reparametrization

We now explain the ambiguity in (18). It arises because the identities (16) do not depend on the parametrization of the curve. To compare parametrizations, we normalize the expansion of $\mathbf{F}$ at $\alpha_i$ so that its constant term is the public column:

$$\mathbf{P}(T) = \lambda_i \mathbf{F}(\alpha_i + T) = \sum_{r\geq 0} \boldsymbol{\tau}_r T^r, \qquad \boldsymbol{\tau}_r = \lambda_i (\partial^{[r]}\mathbf{F})(\alpha_i),$$

by (6). Thus $\boldsymbol{\tau}_0 = \mathbf{p}$ and $\boldsymbol{\tau}_1 = \lambda_i \mathbf{F}'(\alpha_i)$. We call $V_i^{[1]} = \operatorname{span}\{\boldsymbol{\tau}_0, \boldsymbol{\tau}_1\}$, the plane in (18), the *derivative space* of order one at position $i$.

**Definition 4.** A *local representative* of $\mathbf{P}$ is a series $u(T)\,\mathbf{P}(\psi(T))$ with $u, \psi \in \mathbb{K}[[T]]$, $u(0)=1$, $\psi(0)=0$ and $\psi'(0)\neq 0$. The coefficients $(\mathbf{v}_0, \dots, \mathbf{v}_r)$ of $T^0, \dots, T^r$ in a local representative form a *local jet of order $r$* of $\mathbf{P}$.

Every local representative has constant term $\mathbf{p}$. For a local representative $u\mathbf{P}(\psi)$ and $h \in \mathcal{H}$, homogeneity gives

$$h(u\mathbf{P}(\psi)) = u^d\, h(\mathbf{P}(\psi)) = 0.$$

Hence no equation obtained from (16) distinguishes a local representative from $\mathbf{P}$.

At first order, local representatives account for the whole ambiguity in (18). The first coefficient of $u\mathbf{P}(\psi)$ is $u'(0)\mathbf{p} + \psi'(0)\boldsymbol{\tau}_1$, and for $a \in \mathbb{K}$ and $b \in \mathbb{K}^*$ the local representative $(1+aT)\mathbf{P}(bT)$ has first coefficient $a\mathbf{p} + b\boldsymbol{\tau}_1$. Hence the first coefficients of local representatives are exactly the vectors of $V_i^{[1]} \setminus \mathbb{K}\mathbf{p}$, and the equations obtained from (16) cannot single out $\boldsymbol{\tau}_1$ among them. Instead of $\mathbf{F}'(\alpha_i)$ we therefore aim to recover the plane $V_i^{[1]}$, or equivalently one local jet $(\mathbf{p}, \mathbf{v}_1)$ of order one: every such local jet gives $V_i^{[1]} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$, and every $\mathbf{v}_1 \in V_i^{[1]} \setminus \mathbb{K}\mathbf{p}$ is the first coefficient of such a jet.

### 4.3. Conjugate branches

By Section 4.2, the best we can expect from the gradient equations is $\ker \mathbf{J} = V_i^{[1]}$; then $V_i^{[1]}$ is found by linear algebra. For subfield subcodes ($m \geq 2$) there is an obstruction. Let $\varphi\colon x \mapsto x^q$ act coordinatewise on $\mathbb{K}^k$ and coefficientwise on $\mathbb{K}[[T]]^k$, with $T$ fixed. Since $\mathbf{J}$ has entries in $\mathbb{F}_q$, its kernel is stable under $\varphi$. Hence $\ker \mathbf{J} = V_i^{[1]}$ is possible only if $\varphi(V_i^{[1]}) = V_i^{[1]}$, and by (18),

$$E_i = \operatorname{span}\{\mathbf{p}, \boldsymbol{\tau}_1, \varphi(\boldsymbol{\tau}_1), \dots, \varphi^{m-1}(\boldsymbol{\tau}_1)\} \subseteq \ker \mathbf{J}. \tag{19}$$

The conjugate vectors come from other curves through $\mathbf{p}$. For $0 \leq \sigma < m$ put $\mathbf{P}_\sigma = \varphi^\sigma(\mathbf{P})$. Every $h \in \mathcal{H}$ has coefficients in $\mathbb{F}_q$, so $h(\mathbf{P}_\sigma) = \varphi^\sigma(h(\mathbf{P})) = 0$. Moreover $\mathbf{P}_\sigma(0) = \varphi^\sigma(\mathbf{p}) = \mathbf{p}$, because $\mathbf{p} \in \mathbb{F}_q^k$. Explicitly, $\mathbf{P}_\sigma(T) = \varphi^\sigma(\lambda_i)\,\mathbf{F}^{(\sigma)}(\varphi^\sigma(\alpha_i) + T)$, where

$\mathbf{F}^{(\sigma)}$ is obtained by applying $\varphi^\sigma$ to the coefficients of $\mathbf{F}$. We call $\mathbf{P}_0 = \mathbf{P}, \mathbf{P}_1, \dots, \mathbf{P}_{m-1}$ the *branches* at $\mathbf{p}$. Definition 4 and the results of Section 4.2 apply to each branch. In particular, the first coefficients of local representatives of $\mathbf{P}_\sigma$ are the vectors $a\mathbf{p} + b\varphi^\sigma(\boldsymbol{\tau}_1)$ with $b \neq 0$; we call them the *tangent vectors* of $\mathbf{P}_\sigma$. The derivative space of order one of $\mathbf{P}_\sigma$ is $\varphi^\sigma(V_i^{[1]})$. The key-recovery methods below use the derivative spaces only up to an unknown power of $\varphi$ at each position. It is therefore enough to recover a local jet of one branch, without knowing which.

We assume from now on that the $m+1$ vectors $\mathbf{p}$ and $\varphi^\sigma(\boldsymbol{\tau}_1)$, $0 \leq \sigma < m$, are linearly independent, so that $\dim E_i = m+1$. The difficulty is that $\ker \mathbf{J}$ does not separate the branches: a vector of $E_i \setminus \mathbb{K}\mathbf{p}$ is the first coefficient of a local representative of some branch if and only if it lies in one of the $m$ planes $\operatorname{span}\{\mathbf{p}, \varphi^\sigma(\boldsymbol{\tau}_1)\}$. We call a vector of $E_i$ *mixed* if, in the basis $\mathbf{p}, \boldsymbol{\tau}_1, \dots, \varphi^{m-1}(\boldsymbol{\tau}_1)$ of $E_i$, at least two of its coordinates at the vectors $\varphi^\sigma(\boldsymbol{\tau}_1)$ are nonzero. Mixed vectors satisfy all gradient equations but are tangent to no branch. If a mixed vector $\mathbf{v}$ were used, the plane $\operatorname{span}\{\mathbf{p}, \mathbf{v}\}$ would not be a derivative space of any branch, and no local jet extending $(\mathbf{p}, \mathbf{v})$ would be a local jet of a branch. The mixed vectors must therefore be excluded before the higher coefficients are computed.

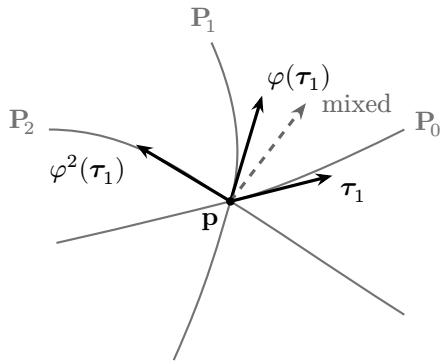



Figure 3: Schematic view of the branches at $\mathbf{p}$ for $m = 3$. Their tangent vectors $\varphi^\sigma(\boldsymbol{\tau}_1)$ lie in $E_i \subseteq \ker \mathbf{J}$, so the gradient equations do not separate them. A mixed vector also lies in $E_i$ but is tangent to no branch.

Since $E_i \subseteq \ker \mathbf{J}$, we have $\operatorname{rank} \mathbf{J} \leq k - m - 1$. We call the case of equality, $\operatorname{rank} \mathbf{J} = k - m - 1$ or equivalently $\ker \mathbf{J} = E_i$, the *maximal-rank case*. For every $m$, the kernel of $\mathbf{J}$ may also be larger than $E_i$, for instance when $\dim \mathcal{H} < k - m - 1$, since $\operatorname{rank} \mathbf{J} \leq \dim \mathcal{H}$.

### 4.4. Separating the branches

To exclude the mixed vectors we use the second-order terms of (16). For $h \in \mathcal{H}$ let

$$Q_h(\mathbf{v}) = [T^2]\, h(\mathbf{p} + T\mathbf{v}) = \sum_{|\mathbf{a}|=2} (\partial^{[\mathbf{a}]} h)(\mathbf{p})\, \mathbf{v}^{\mathbf{a}}$$

be the quadratic part of $h$ at $\mathbf{p}$. Let

$$\mathbf{p} + \mathbf{v}_1 T + \mathbf{v}_2 T^2 + \cdots$$

be a local representative of a branch $\mathbf{P}_\sigma$, $0 \le \sigma < m$. By (5), the coefficients of $T$ and $T^2$ in $h(\mathbf{p} + \mathbf{v}_1 T + \mathbf{v}_2 T^2 + \cdots) = 0$ give

$$\nabla h(\mathbf{p})\,\mathbf{v}_1 = 0, \qquad \nabla h(\mathbf{p})\,\mathbf{v}_2 + Q_h(\mathbf{v}_1) = 0. \tag{20}$$

When we solve for $\mathbf{v}_1$, the second coefficient $\mathbf{v}_2$ is in general unknown. The second equation does not involve $\mathbf{v}_2$ if $\nabla h(\mathbf{p}) = 0$. We therefore use the second equation only for $h$ in the subspace

$$\mathcal{H}^0 = \{h \in \mathcal{H} : \nabla h(\mathbf{p}) = 0\},$$

where it becomes $Q_h(\mathbf{v}_1) = 0$. The tangent vectors of all branches are therefore solutions of

$$\mathbf{J}\mathbf{v} = \mathbf{0}, \qquad Q_h(\mathbf{v}) = 0 \quad \text{for all } h \in \mathcal{H}^0. \tag{21}$$

The system (21) has coefficients in $\mathbb{F}_q$, since the $h \in \mathcal{H}$ do and $\mathbf{p} \in \mathbb{F}_q^k$. Like the hold-out kernel, it is computed over $\mathbb{F}_q$. Its solution set is therefore stable under $\varphi$, so the quadratic equations cannot distinguish the branches from each other. In practice we compute a basis of $\ker \mathbf{J}$ consisting of vectors in $\mathbb{F}_q^k$, with $m+1$ elements in the maximal-rank case $\operatorname{rank} \mathbf{J} = k - m - 1$, and solve the quadratic equations in the coordinates of this basis.

**Remark 9.** *In (21) we use the second equation of (20) only for $h \in \mathcal{H}^0$. For the other $h \in \mathcal{H}$ it involves the unknown $\mathbf{v}_2$, and these equations do not restrict $\mathbf{v}_1$ further. To see this, choose a subspace $\mathcal{G} \subseteq \mathcal{H}$ with $\mathcal{H} = \mathcal{H}^0 \oplus \mathcal{G}$. Since $\nabla h(\mathbf{p}) = 0$ only for $h \in \mathcal{H}^0$, the linear map $h \mapsto \nabla h(\mathbf{p})$ is injective on $\mathcal{G}$. The gradients at $\mathbf{p}$ of a basis of $\mathcal{G}$ are therefore linearly independent vectors of $\mathbb{F}_q^k$, also over $\mathbb{K}$, and for every $\mathbf{v}_1$ there is a $\mathbf{v}_2$ with*

$$\nabla g(\mathbf{p})\,\mathbf{v}_2 = -Q_g(\mathbf{v}_1) \quad \text{for all } g \in \mathcal{G}.$$

*Every $h \in \mathcal{H}$ is the sum of an element of $\mathcal{H}^0$ and an element of $\mathcal{G}$. Hence, if $Q_h(\mathbf{v}_1) = 0$ for all $h \in \mathcal{H}^0$, this $\mathbf{v}_2$ satisfies the second equation of (20) for all $h \in \mathcal{H}$. The equations with $h \notin \mathcal{H}^0$ restrict $\mathbf{v}_1$ only when more is known about $\mathbf{v}_2$.*

**Remark 10.** *Other equations for $\mathbf{v}_1$ can be used. For example, one may solve for $\mathbf{v}_1, \mathbf{v}_2, \ldots$ jointly, using the equations of order three and higher. For binary Goppa codes, $\mathbf{J}\mathbf{v}_2 = \mathbf{0}$ for every local representative of a branch (Proposition 8), so the second equation in (20) gives $Q_h(\mathbf{v}_1) = 0$ for all $h \in \mathcal{H}$, not only for $h \in \mathcal{H}^0$.*

Besides the tangent vectors of the branches, (21) has the solution $\mathbf{p}$: $\mathbf{J}\mathbf{p} = \mathbf{0}$ by Euler's identity, and $Q_h(\mathbf{p}) = \binom{d}{2} h(\mathbf{p}) = 0$. Since the equations are homogeneous, the solution set of (21) contains the $m$ planes $\operatorname{span}\{\mathbf{p}, \varphi^\sigma(\boldsymbol{\tau}_1)\}$.

Further equations that hold on these planes may be added to (21), such as those in Remark 10. We call (21), together with such added equations, the *first-order system*.

**Assumption 2** (Branch separation)**.** The solution set of the first-order system is

$$\bigcup_{\sigma=0}^{m-1} \operatorname{span}\{\mathbf{p}, \varphi^{\sigma}(\boldsymbol{\tau}_1)\}.$$

Under Assumption 2, every solution $\mathbf{v} \notin \mathbb{K}\mathbf{p}$ of the first-order system is a tangent vector of some branch $\mathbf{P}_\sigma$, and $\operatorname{span}\{\mathbf{p}, \mathbf{v}\} = \varphi^{\sigma}(V_i^{[1]})$. For $m = 1$ the assumption says that the solution set is $V_i^{[1]}$; the case $\ker \mathbf{J} = V_i^{[1]}$ implies it.

In the maximal-rank case $\dim \ker \mathbf{J} = m + 1$, a rank computation gives a sufficient condition for the assumption.

**Proposition 4.** *Suppose the maximal-rank case* $\ker \mathbf{J} = E_i$*. If the restrictions to* $\ker \mathbf{J}$ *of the quadratic forms in the first-order system span over* $\mathbb{K}$ *a space of dimension* $\binom{m}{2}$*, then Assumption 2 holds.*

*Proof.* Write the vectors of $\ker \mathbf{J} = E_i$ as $z\mathbf{p} + \sum_\sigma z_\sigma \varphi^{\sigma}(\boldsymbol{\tau}_1)$. Every quadratic form $Q$ in the first-order system vanishes on the planes $\operatorname{span}\{\mathbf{p}, \varphi^{\sigma}(\boldsymbol{\tau}_1)\}$. On such a plane the restriction of $Q$ to $\ker \mathbf{J}$ is a quadratic form in $z$ and $z_\sigma$ that vanishes identically, so the restriction contains none of the monomials $z^2$, $zz_\sigma$, $z_\sigma^2$. It therefore lies in the span of the $\binom{m}{2}$ monomials $z_\sigma z_\rho$, $\sigma < \rho$, and by hypothesis the restrictions span this space. Hence a vector of $\ker \mathbf{J}$ is a common zero of these forms if and only if at most one $z_\sigma$ is nonzero, that is, if and only if it lies in one of the planes. All equations of the first-order system hold on the planes, and every solution lies in $\ker \mathbf{J}$, so the solution set is the union of the planes. □

**Explicit algorithm.** Under the hypotheses of Proposition 4, we can recover the tangent planes by an eigenvalue computation. Take a basis $\mathbf{p}, \mathbf{b}_1, \dots, \mathbf{b}_m$ of $\ker \mathbf{J}$ consisting of vectors in $\mathbb{F}_q^k$, and let $y_1, \dots, y_m$ be the coordinates at $\mathbf{b}_1, \dots, \mathbf{b}_m$. The restricted quadratic forms involve only these coordinates. Their span $W$ has dimension $\binom{m}{2}$ and codimension $m$ in the space of quadratic forms in $y_1, \dots, y_m$.

To construct the matrices, choose linear forms $\ell_1 \neq 0$ and $\ell_2$ over $\mathbb{F}_q$. Fix a basis of quadratic forms modulo $W$ represented by forms over $\mathbb{F}_q$. For $j = 1, 2$, let $A_j \in \mathbb{F}_q^{m \times m}$ have as its $a$-th column the coordinates of $\ell_j y_a$ modulo $W$ in this basis.

**Proposition 5.** *The matrix* $A_1$ *is invertible. If* $B = A_1^{-1} A_2$ *has* $m$ *distinct eigenvalues in* $\mathbb{K}$*, then every nonzero left eigenvector* $\mathbf{u} = (u_1, \dots, u_m)$ *gives a tangent vector*

$$\mathbf{v} = \sum_{a=1}^{m} u_a \mathbf{b}_a.$$

*The planes* $\operatorname{span}\{\mathbf{p}, \mathbf{v}\}$ *obtained from the* $m$ *eigenspaces are precisely the tangent planes of the branches.*

*Proof.* In the branch coordinates $z_\sigma$ from the preceding proof, $W$ is spanned by the mixed monomials $z_\sigma z_\rho$, $\sigma < \rho$. The classes of $z_\sigma^2$ therefore form a basis of quadratic forms modulo $W$. Multiplication by $\ell_j$ maps $z_\sigma$ to $\ell_j(\varphi^{\sigma}(\boldsymbol{\tau}_1)) z_\sigma^2$ modulo $W$. Thus $A_1$ and $A_2$ are diagonal in these bases.

To see that $A_1$ is invertible, suppose a linear form in $y_1, \dots, y_m$ over $\mathbb{F}_q$ vanishes at $\boldsymbol{\tau}_1$. It then vanishes at every conjugate vector $\varphi^\sigma(\boldsymbol{\tau}_1)$. These vectors form a basis modulo $\mathbb{K}\mathbf{p}$, so the form is zero. Thus the nonzero form $\ell_1$ vanishes at none of the conjugates, and all diagonal entries of $A_1$ are nonzero.

In the branch coordinates, $B$ is diagonal with entries

$$\theta_\sigma = \frac{\ell_2(\varphi^\sigma(\boldsymbol{\tau}_1))}{\ell_1(\varphi^\sigma(\boldsymbol{\tau}_1))}.$$

If these entries are distinct, every nonzero left eigenvector in the branch coordinates has exactly one nonzero coordinate. In the original coordinates, the eigenspace for $\theta_\sigma$ is therefore spanned by the row $(y_1(\varphi^\sigma(\boldsymbol{\tau}_1)), \dots, y_m(\varphi^\sigma(\boldsymbol{\tau}_1)))$. The corresponding $\mathbf{v}$ differs from a nonzero scalar multiple of $\varphi^\sigma(\boldsymbol{\tau}_1)$ by a multiple of $\mathbf{p}$. It is therefore a tangent vector of that branch. □

The proof also shows that, for fixed $\ell_1$, the map $\ell_2 \mapsto \ell_2(\boldsymbol{\tau}_1)/\ell_1(\boldsymbol{\tau}_1)$ is injective. Both spaces have dimension $m$ over $\mathbb{F}_q$, so every element of $\mathbb{K}$ occurs as a ratio. If the ratio has degree $m$ over $\mathbb{F}_q$, its Frobenius conjugates give $m$ distinct eigenvalues of $B$. Such elements exist, so we can keep $\ell_1$ fixed and choose new forms $\ell_2$ until the eigenvalues are distinct.

**Remark 11** (Linearization)**.** *For $m \geq 2$, we may also solve the quadratic equations by guessing one coordinate and then using linear algebra. Normalize by setting $y_1 = 1$, and try $y_2 = \theta$ for $\theta \in \mathbb{K}$. For each guess, replace the products $y_a y_b$, $3 \leq a \leq b \leq m$, by independent unknowns. The equations are then linear in $y_3, \dots, y_m$ and these new unknowns. If $y_2/y_1$ takes distinct values on the branches, each consistent guess determines one normalized tangent vector, which we check in the original quadratic equations.*

### 4.5. Higher-order coefficients

Our next aim is to extend a local jet $(\mathbf{p}, \mathbf{v}_1)$ of order one of a branch $\mathbf{P}_\sigma$, found as in Section 4.4, to local jets of higher order of the same branch. Once $\mathbf{v}_1$ is fixed, each further coefficient satisfies linear equations with known right-hand sides, and we find the coefficients one at a time. The difficulty is to make sure that the solutions stay on $\mathbf{P}_\sigma$.

The equations for the higher coefficients come from the chain rule, applied to $h$ composed with a local representative $\mathbf{p} + \mathbf{v}_1 T + \mathbf{v}_2 T^2 + \cdots$. Expand $h$ at $\mathbf{p}$ as in (5). A term of this expansion that contains the coefficients $\mathbf{v}_{e_1}, \dots, \mathbf{v}_{e_j}$ contributes to the power $T^{e_1+\cdots+e_j}$. Hence $\mathbf{v}_r$ reaches the power $T^r$ only through the gradient term, as in the first-order equation (17). To write the remaining terms, let $\Phi_{h,r}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1})$ and $\Phi_{h,r+1}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1})$ be the coefficients of $T^r$ and $T^{r+1}$ in $h(\mathbf{p} + \mathbf{v}_1 T + \cdots + \mathbf{v}_{r-1} T^{r-1})$. The second subscript is the power of $T$, and the arguments are the coefficients already known. Then

$$[T^r]\, h(\mathbf{p} + \mathbf{v}_1 T + \mathbf{v}_2 T^2 + \cdots) = \nabla h(\mathbf{p})\, \mathbf{v}_r + \Phi_{h,r}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1}).$$

Suppose that $\mathbf{v}_1, \dots, \mathbf{v}_{r-1}$ are known for some $r \geq 2$. Then $\mathbf{v}_r$ satisfies the linear system

$$\nabla h(\mathbf{p})\, \mathbf{v}_r = -\Phi_{h,r}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1}) \quad \text{for all } h \in \mathcal{H}. \tag{22}$$

Both sides are linear in $h$, so it suffices to take $h = h_1, \dots, h_M$. Since $\mathbf{J}$ has the rows $\nabla h_j(\mathbf{p})$, the system has the matrix form

$$\mathbf{J}\mathbf{v}_r = -\big(\Phi_{h_1,r}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1}), \dots, \Phi_{h_M,r}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1})\big)^{\mathsf{T}},$$

with the same matrix as the gradient equations and a right-hand side computed from the known coefficients. In particular, $\Phi_{h,2}(\mathbf{v}_1) = Q_h(\mathbf{v}_1)$, and for $r = 2$ this is the second equation of (20).

The system (22) has the same matrix $\mathbf{J}$ as the gradient equations, so it inherits their ambiguity: when it is consistent, its solutions form a coset of $\ker \mathbf{J}$. By Sections 4.1 and 4.3, $\ker \mathbf{J}$ contains the plane $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$ and, for subfield subcodes, also the tangent vectors of the other branches. The two parts behave differently. Adding a vector of the plane to the coefficient of order $r$ of a local representative gives the coefficient of another local representative, as at first order. Adding one of the other directions may give a vector that is not the coefficient of any local representative of $\mathbf{P}_\sigma$.

We first make the statement about the plane precise. Suppose that $\mathbf{p}, \mathbf{v}_1, \dots, \mathbf{v}_{r-1}$ are the first coefficients of a local representative

$$\mathbf{W} = \mathbf{p} + \mathbf{v}_1 T + \cdots + \mathbf{v}_{r-1} T^{r-1} + \mathbf{w}_r T^r + \cdots$$

of $\mathbf{P}_\sigma$. Every $h \in \mathcal{H}$ vanishes on $\mathbf{W}$, so $\mathbf{w}_r$ solves (22), and the coset of solutions is $\mathbf{w}_r + \ker \mathbf{J}$. At first order, the representatives $(1 + aT)\mathbf{P}(bT)$ gave all first coefficients $a\mathbf{p} + b\boldsymbol{\tau}_1$ (Section 4.2). The analogous change at order $r$ leaves the lower coefficients unchanged.

**Lemma 4.** *Let $\mathbf{W} = \mathbf{p} + \mathbf{w}_1 T + \mathbf{w}_2 T^2 + \cdots$ be a local representative of a branch $\mathbf{P}_\sigma$, and let $r \geq 2$ and $a, b \in \mathbb{K}$. Then $(1 + aT^r)\, \mathbf{W}(T + bT^r)$ is again a local representative of $\mathbf{P}_\sigma$. It has the same coefficients as $\mathbf{W}$ below order $r$, and its coefficient of order $r$ is $\mathbf{w}_r + a\mathbf{p} + b\mathbf{w}_1$.*

*Proof.* Write $\mathbf{W} = u\, \mathbf{P}_\sigma(\psi)$. The new series is $(1 + aT^r)\, u(T + bT^r)\, \mathbf{P}_\sigma(\psi(T + bT^r))$. Its scalar factor has constant term 1, and its parameter has the same linear coefficient as $\psi$, so it is a local representative. Substituting $T + bT^r$ for $T$ changes a term $\mathbf{w}_e T^e$ only at orders $e + r - 1$ and higher. Below order $r$ nothing changes, and at order $r$ only $\mathbf{w}_1 T$ contributes, adding $b\mathbf{w}_1 T^r$. Multiplying by $1 + aT^r$ adds $a\mathbf{p}T^r$ and terms of higher order. □

Thus every vector of $\mathbf{w}_r + \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$ is the coefficient of order $r$ of a local representative of $\mathbf{P}_\sigma$ with the same lower coefficients. These solutions are harmless, and no equation obtained from (16) excludes them. If $\ker \mathbf{J}$ contains nothing else, the recursion never has to branch.

**Proposition 6.** *Let* $(\mathbf{p}, \mathbf{v}_1)$ *be a local jet of order one of a branch* $\mathbf{P}_\sigma$*, and suppose that* $\ker \mathbf{J} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$*. Then the systems* (22)*, solved successively for* $r = 2, 3, \ldots$*, are consistent, and every resulting sequence* $\mathbf{p}, \mathbf{v}_1, \mathbf{v}_2, \ldots$ *is the sequence of coefficients of one local representative of* $\mathbf{P}_\sigma$*.*

*Proof.* Suppose that $\mathbf{p}, \mathbf{v}_1, \ldots, \mathbf{v}_{r-1}$ are the first coefficients of a local representative $\mathbf{W}$ of $\mathbf{P}_\sigma$, and let $\mathbf{w}_r$ be its coefficient of order $r$. Then $\mathbf{w}_r$ solves (22), so its solutions form the affine space $\mathbf{w}_r + \ker \mathbf{J} = \mathbf{w}_r + \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$. By Lemma 4, each solution is the coefficient of order $r$ of a local representative of $\mathbf{P}_\sigma$ whose lower coefficients are $\mathbf{p}, \mathbf{v}_1, \ldots, \mathbf{v}_{r-1}$. By induction on $r$, every initial segment of the sequence is a local jet of $\mathbf{P}_\sigma$. The change made at step $r$ affects the scalar factor and the parameter only from order $r$ on, so the compositions of these changes converge coefficientwise and give one local representative for the whole sequence. □

**Remark 12.** *For non-subfield subcodes in the maximal-rank case* $\dim \ker \mathbf{J} = 2$*, the hypothesis holds, since then* $\ker \mathbf{J} = V_i^{[1]} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$*. In general* $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\} \subseteq \ker \mathbf{J}$*, so the hypothesis is equivalent to* $\operatorname{rank} \mathbf{J} = k - 2$*.*

For subfield subcodes the proposition is not directly applicable, since $\ker \mathbf{J}$ contains $E_i$, which has dimension $m + 1 \geq 3$. In this case, and whenever $\ker \mathbf{J}$ is larger than $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$, the other solutions of (22) differ from $\mathbf{w}_r$ by further vectors of $\ker \mathbf{J}$, such as tangent vectors of the other branches, and must be excluded. For this we can use the equations at the next order.

Counting powers of $T$ as for (22), we get

$$[T^{r+1}]\, h(\mathbf{p} + \mathbf{v}_1 T + \mathbf{v}_2 T^2 + \cdots) = \nabla h(\mathbf{p})\, \mathbf{v}_{r+1} + B_h(\mathbf{v}_1, \mathbf{v}_r) + \Phi_{h,r+1}(\mathbf{v}_1, \ldots, \mathbf{v}_{r-1}),$$

where $B_h(\mathbf{x}, \mathbf{y}) = Q_h(\mathbf{x}+\mathbf{y}) - Q_h(\mathbf{x}) - Q_h(\mathbf{y})$ is the polar form of $Q_h$. Besides the gradient term, $\mathbf{v}_r$ now appears once more, paired with $\mathbf{v}_1$ in the quadratic part of $h$; all other terms containing $\mathbf{v}_r$ reach only $T^{r+2}$ and higher powers. Hence every $h \in \mathcal{H}$ gives the equation

$$\nabla h(\mathbf{p})\, \mathbf{v}_{r+1} + B_h(\mathbf{v}_1, \mathbf{v}_r) = -\Phi_{h,r+1}(\mathbf{v}_1, \ldots, \mathbf{v}_{r-1}). \tag{23}$$

It contains the next coefficient $\mathbf{v}_{r+1}$. If nothing is known about $\mathbf{v}_{r+1}$, we can only use $h \in \mathcal{H}^0$, for which the first term vanishes, as in Section 4.4. This gives the further linear equations

$$B_h(\mathbf{v}_1, \mathbf{v}_r) = -\Phi_{h,r+1}(\mathbf{v}_1, \ldots, \mathbf{v}_{r-1}) \quad \text{for all } h \in \mathcal{H}^0. \tag{24}$$

**Remark 13.** *For some codes more is known about* $\mathbf{v}_{r+1}$*. For instance, for binary Goppa codes, a suitable local representative has* $\mathbf{v}_{r+1}$ *determined by the earlier coefficients when* $r$ *is odd (Section 4.6), and then* (23) *can be used for all* $h \in \mathcal{H}$*.*

Let $\mathbf{L}_0$ be the matrix whose rows represent the linear forms $\mathbf{x} \mapsto B_g(\mathbf{v}_1, \mathbf{x})$, where $g$ runs over a basis of $\mathcal{H}^0$, and put

$$\mathbf{A} = \begin{bmatrix} \mathbf{J} \\ \mathbf{L}_0 \end{bmatrix}.$$

By linearity in $h$, it suffices to impose (22) for $h_1, \dots, h_M$ and (24) for a basis of $\mathcal{H}^0$. The two systems then form one linear system for $\mathbf{v}_r$ with matrix $\mathbf{A}$. This matrix depends only on $\mathbf{p}$, $\mathcal{H}$ and $\mathbf{v}_1$. Only the right-hand side changes with $r$, so one row reduction of $\mathbf{A}$ serves all orders.

**Remark 14.** *The polar form is the Hessian form of $h$ at $\mathbf{p}$: $B_h(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\mathsf{T}} \mathbf{H}_h \, \mathbf{y}$, where $\mathbf{H}_h$ is the matrix of second partial derivatives $(\partial^2 h / \partial X_j \partial X_l)(\mathbf{p})$. The rows of $\mathbf{L}_0$ are therefore the vectors $\mathbf{v}_1^{\mathsf{T}} \mathbf{H}_g$. Moreover, $B_h(\mathbf{x}, \mathbf{x}) = 2Q_h(\mathbf{x})$, so in characteristic two $B_h$ is alternating.*

**Proposition 7** (Continuation with next-order equations)**.** *Proposition 6 remains true with $\mathbf{J}$ replaced by $\mathbf{A}$ and with the systems (22) replaced by the systems formed by (22) and (24).*

*Proof.* The proof of Proposition 6 uses two facts: the coefficient of order $r$ of a local representative with the given lower coefficients solves the system, and the kernel of the matrix is $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$. The first fact holds because (24), like (22), is obtained from (16); the second is the hypothesis. □

By Lemma 4, $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\} \subseteq \ker \mathbf{A}$, so the hypothesis is equivalent to $\operatorname{rank} \mathbf{A} = k - 2$. It is checked once, before the recursion starts. This is a sufficient condition: if $\ker \mathbf{A}$ is larger, other solutions may appear at some order, and the proposition gives no guarantee for them. In the maximal-rank case $\dim \ker \mathbf{J} = m + 1$, the rank condition of Proposition 4 for the forms from $\mathcal{H}^0$ is a sufficient condition for $\ker \mathbf{A} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$.

**Corollary 1.** *Suppose that $\ker \mathbf{J} = E_i$ and that the restrictions to $\ker \mathbf{J}$ of the forms $Q_h$, $h \in \mathcal{H}^0$, span over $\mathbb{K}$ a space of dimension $\binom{m}{2}$. Then $\ker \mathbf{A} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$ for every tangent vector $\mathbf{v}_1$ of a branch.*

*Proof.* Let $\mathbf{v}_1 = a\mathbf{p} + b\,\varphi^{\sigma}(\boldsymbol{\tau}_1)$ with $b \neq 0$, and use the coordinates $z, z_0, \dots, z_{m-1}$ on $\ker \mathbf{J}$ from the proof of Proposition 4. There the restrictions of the forms $Q_h$, $h \in \mathcal{H}^0$, were shown to lie in the span of the monomials $z_\rho z_\pi$, $\rho < \pi$; by hypothesis they span it. For $Q = z_\sigma z_\rho$ with $\rho \neq \sigma$, the polar form satisfies $B(\mathbf{v}_1, \mathbf{x}) = b\, z_\rho(\mathbf{x})$. Hence the rows of $\mathbf{L}_0$, restricted to $\ker \mathbf{J}$, span the functionals $z_\rho$ with $\rho \neq \sigma$, and $\ker \mathbf{A}$ consists of the vectors of $\ker \mathbf{J}$ with $z_\rho = 0$ for all $\rho \neq \sigma$. This is $\operatorname{span}\{\mathbf{p}, \varphi^{\sigma}(\boldsymbol{\tau}_1)\} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$. □

The condition of Corollary 1 is sufficient, but not necessary. For a fixed branch $\mathbf{P}_\sigma$ it is enough that the forms from $\mathcal{H}^0$ give the $m - 1$ functionals $z_\rho$, $\rho \neq \sigma$, and $\ker \mathbf{A} = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$ may also hold when $\ker \mathbf{J}$ is larger than $E_i$. Only the forms from $\mathcal{H}^0$ enter, since for $h \notin \mathcal{H}^0$ the equation at order $r + 1$ contains the unknown $\mathbf{v}_{r+1}$.

### 4.6. Binary Goppa codes

For binary Goppa codes it is useful to consider local representatives characterized by $\mathbf{v}_{2r} = \mathbf{v}_r^{\circ 2}$ for $r \geq 1$, where $\circ 2$ denotes coordinatewise squaring. This relation gives more equations than the general procedure of Sections 4.4 and 4.5: all $h \in \mathcal{H}$ can be used,

not only $h \in \mathcal{H}^0$, both in the equations $Q_h(\mathbf{v}_1) = 0$ at first order and in the next-order equation (23) at odd orders.

Let $q = 2$, and consider a binary Goppa code $\Gamma(\boldsymbol{\alpha}, G)$, where $G$ is square-free of degree $t$ and $2t < n$. Shortening such a code gives the binary Goppa code with the same Goppa polynomial on the remaining support, so we may ignore shortening. We use the GRS representation of Proposition 2: $D = n - 2t - 1$ and $\lambda_j = G(\alpha_j)^2/\Pi'_{\boldsymbol{\alpha}}(\alpha_j)$. For $q = 2$ the Frobenius map $\varphi$ of Section 4.3 is coordinatewise squaring, and we write $\mathbf{v}^{\circ 2} = \varphi(\mathbf{v})$. For a series $\mathbf{W}(T)$, $\mathbf{W}(T)^{\circ 2}$ denotes the series whose coordinates are the squares of those of $\mathbf{W}(T)$.

**Lemma 5.** *The series*

$$\mathbf{S}(T) = \sum_{j=1}^{n} \frac{T\,\mathbf{y}_j}{T + \alpha_i - \alpha_j} = \mathbf{p} + \sum_{r \geq 1} \boldsymbol{\gamma}_r T^r, \qquad \boldsymbol{\gamma}_r = \sum_{j \neq i} \frac{\mathbf{y}_j}{(\alpha_i - \alpha_j)^r},$$

*is a local representative of* $\mathbf{P}$*, and* $\boldsymbol{\gamma}_{2r} = \boldsymbol{\gamma}_r^{\circ 2}$ *for all* $r \geq 1$*.*

*Proof.* Both sides of

$$G(X)^2 \mathbf{F}(X) = \Pi_{\boldsymbol{\alpha}}(X) \sum_{j=1}^{n} \frac{\mathbf{y}_j}{X - \alpha_j}$$

are polynomial vectors of degree at most $n - 1$. At $X = \alpha_j$ both equal $\Pi'_{\boldsymbol{\alpha}}(\alpha_j)\mathbf{y}_j$, by the formula for $\lambda_j$, so they are equal. Put $X = \alpha_i + T$ and write $\Pi_{\boldsymbol{\alpha}}(\alpha_i + T) = T\,\Theta(T)$, where $\Theta(0) = \Pi'_{\boldsymbol{\alpha}}(\alpha_i)$. Then

$$\mathbf{S}(T) = \frac{G(\alpha_i + T)^2}{\Theta(T)}\,\mathbf{F}(\alpha_i + T) = \frac{G(\alpha_i + T)^2}{\lambda_i \Theta(T)}\,\mathbf{P}(T),$$

and the scalar factor has constant term $G(\alpha_i)^2/(\lambda_i \Pi'_{\boldsymbol{\alpha}}(\alpha_i)) = 1$. Hence $\mathbf{S}$ is a local representative of $\mathbf{P}$. The term with $j = i$ is $\mathbf{p}$, and in characteristic two $T/(T + \delta) = \sum_{r \geq 1} T^r/\delta^r$, which gives the formula for $\boldsymbol{\gamma}_r$. Since the $\mathbf{y}_j$ are binary, squaring $\boldsymbol{\gamma}_r$ squares the scalars $(\alpha_i - \alpha_j)^{-r}$ and leaves the $\mathbf{y}_j$ fixed. This gives $\boldsymbol{\gamma}_{2r} = \boldsymbol{\gamma}_r^{\circ 2}$. □

The series $\mathbf{S}$ involves the hidden support. We use it, and the representatives constructed from it below, only in proofs. The recovery algorithm imposes the relation between the coefficients and does not need the support.

The relation for $r = 1$ already has a consequence at first order.

**Proposition 8.** *Every local representative of every branch has a second coefficient* $\mathbf{v}_2$ *with* $\mathbf{J}\mathbf{v}_2 = \mathbf{0}$*. Consequently,* $Q_h(\mathbf{v}_1) = 0$ *for all* $h \in \mathcal{H}$ *and every tangent vector* $\mathbf{v}_1$ *of a branch.*

*Proof.* The vector $\boldsymbol{\gamma}_1$ is the first coefficient of $\mathbf{S}$, so $\mathbf{J}\boldsymbol{\gamma}_1 = \mathbf{0}$. Since $\mathbf{J}$ has entries in $\mathbb{F}_2$, $\mathbf{J}\boldsymbol{\gamma}_2 = \mathbf{J}\boldsymbol{\gamma}_1^{\circ 2} = (\mathbf{J}\boldsymbol{\gamma}_1)^{\circ 2} = \mathbf{0}$. Every local representative of $\mathbf{P}$ has the form $u\mathbf{S}(\psi)$ with $u = 1 + u_1 T + u_2 T^2 + \cdots$ and $\psi = \psi_1 T + \psi_2 T^2 + \cdots$, $\psi_1 \neq 0$. Its second coefficient is $u_2\mathbf{p} + (u_1\psi_1 + \psi_2)\boldsymbol{\gamma}_1 + \psi_1^2\boldsymbol{\gamma}_2$, which lies in $\ker \mathbf{J}$. For a branch $\mathbf{P}_\sigma$, apply $\varphi^\sigma$. The second equation of (20) now reads $Q_h(\mathbf{v}_1) = -\nabla h(\mathbf{p})\,\mathbf{v}_2 = 0$. □

The equations $Q_h(\mathbf{v}) = 0$, $h \in \mathcal{H}$, also hold for $\mathbf{p}$, so they hold on the planes $\operatorname{span}\{\mathbf{p}, \varphi^\sigma(\boldsymbol{\tau}_1)\}$. They may therefore be added to the first-order system, and Proposition 4 applies to this larger family of quadratic forms.

At higher orders we use the relation between orders $r$ and $2r$ for all $r$. It holds for $\mathbf{S}$, but not for every local representative: for instance, multiplying $\mathbf{S}$ by $1 + aT^2$ keeps $\boldsymbol{\gamma}_1$ and replaces $\boldsymbol{\gamma}_2$ by $\boldsymbol{\gamma}_2 + a\mathbf{p}$. We call a local representative $\mathbf{W} = \mathbf{p} + \mathbf{w}_1 T + \mathbf{w}_2 T^2 + \cdots$ of a branch *normalized* if $\mathbf{w}_{2r} = \mathbf{w}_r^{\circ 2}$ for all $r \geq 1$. Every first coefficient admits such a representative.

**Proposition 9.** *Every local jet $(\mathbf{p}, \mathbf{v}_1)$ of order one of a branch $\mathbf{P}_\sigma$ extends to a normalized local representative of $\mathbf{P}_\sigma$.*

*Proof.* First let $\sigma = 0$. The first coefficients of local representatives of $\mathbf{P}$ are the vectors of $V_i^{[1]} \setminus \mathbb{K}\mathbf{p}$, and $V_i^{[1]} = \operatorname{span}\{\mathbf{p}, \boldsymbol{\gamma}_1\}$ (Section 4.2). Hence $\mathbf{v}_1 = a\mathbf{p} + b\boldsymbol{\gamma}_1$ with $a \in \mathbb{K}$ and $b \in \mathbb{K}^*$. Put

$$\mathbf{W}(T) = \frac{1}{1 + aT}\, \mathbf{S}\Big(\frac{bT}{1 + aT}\Big).$$

This is a local representative of $\mathbf{P}$ with first coefficient $a\mathbf{p} + b\boldsymbol{\gamma}_1 = \mathbf{v}_1$. With $\mathbf{p}_\infty = \sum_{j=1}^n \mathbf{y}_j$, partial fractions give

$$\mathbf{W}(T) = \frac{\mathbf{p}_\infty}{1 + aT} + \sum_{j \neq i} \frac{\mathbf{y}_j}{1 + (a + b/(\alpha_i - \alpha_j))T},$$

so that

$$[T^r]\, \mathbf{W} = a^r \mathbf{p}_\infty + \sum_{j \neq i} \Big(a + \frac{b}{\alpha_i - \alpha_j}\Big)^r \mathbf{y}_j.$$

The vectors $\mathbf{p}_\infty$ and $\mathbf{y}_j$ are binary, so the coefficient of order $2r$ is the square of the coefficient of order $r$. For a branch $\mathbf{P}_\sigma$, apply this construction to $\varphi^{-\sigma}(\mathbf{v}_1)$ and then apply $\varphi^\sigma$, which commutes with squaring. □

Along a normalized representative, every even coefficient is determined by an earlier one: $\mathbf{v}_2 = \mathbf{v}_1^{\circ 2}$, $\mathbf{v}_4 = \mathbf{v}_2^{\circ 2}$, and in general $\mathbf{v}_{2r} = \mathbf{v}_r^{\circ 2}$. We therefore compute only the odd coefficients. At an odd order $r \geq 3$, the next coefficient $\mathbf{v}_{r+1} = \mathbf{v}_{(r+1)/2}^{\circ 2}$ is already known, since $(r+1)/2 < r$. The equation (23) then holds for all $h \in \mathcal{H}$ and is linear in $\mathbf{v}_r$:

$$B_h(\mathbf{v}_1, \mathbf{v}_r) = -\Phi_{h,r+1}(\mathbf{v}_1, \dots, \mathbf{v}_{r-1}) - \nabla h(\mathbf{p})\, \mathbf{v}_{(r+1)/2}^{\circ 2} \quad \text{for all } h \in \mathcal{H}. \tag{25}$$

Let $\mathbf{L}$ be the matrix whose rows represent the linear forms $\mathbf{x} \mapsto B_{h_j}(\mathbf{v}_1, \mathbf{x})$, $1 \leq j \leq M$, and put

$$\mathbf{A}_\Gamma = \begin{bmatrix} \mathbf{J} \\ \mathbf{L} \end{bmatrix}.$$

By linearity in $h$, (22) and (25) form one linear system for $\mathbf{v}_r$ with matrix $\mathbf{A}_\Gamma$. We call this use of the next-order equations *backward solving*. The rows of $\mathbf{L}_0$ are linear combinations of the rows of $\mathbf{L}$, so $\ker \mathbf{A}_\Gamma \subseteq \ker \mathbf{A}$.

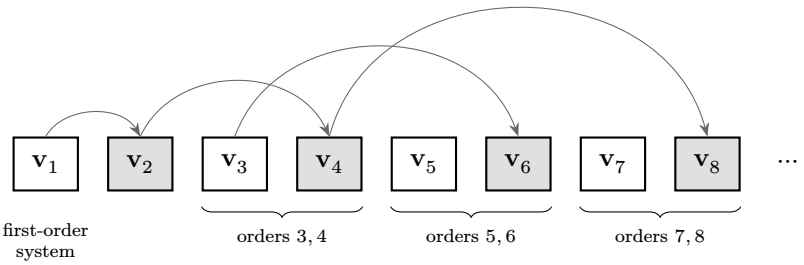



Figure 4: Backward solving along a normalized representative. The shaded coefficients are determined by $\mathbf{v}_{2r} = \mathbf{v}_r^{\circ 2}$ (arrows). Each coefficient $\mathbf{v}_r$ with odd $r \geq 3$ is found from the equations at orders $r$ and $r+1$ (braces), using the known $\mathbf{v}_{r+1}$.

As in Section 4.5, part of the solution set comes from changing the representative, but now the change must keep it normalized. Normalization can be written as the identity

$$T\mathbf{W}'(T) = \mathbf{W}(T) + \mathbf{W}(T)^{\circ 2},$$

where $\mathbf{W}'$ is the formal derivative. At odd orders both sides agree for every series, and at order $2r$ the identity says $\mathbf{w}_{2r} = \mathbf{w}_r^{\circ 2}$, including $\mathbf{p}^{\circ 2} = \mathbf{p}$ for $r = 0$.

**Lemma 6.** *Let $\mathbf{W} = \mathbf{p} + \mathbf{w}_1 T + \mathbf{w}_2 T^2 + \cdots$ be a normalized local representative of a branch $\mathbf{P}_\sigma$, let $r \geq 3$ be odd, and let $a, b \in \mathbb{K}$. For $\psi(T) = T + bT^r + aT^{r+1}$ the series*

$$\widetilde{\mathbf{W}}(T) = \frac{T\psi'(T)}{\psi(T)}\,\mathbf{W}(\psi(T))$$

*is a normalized local representative of $\mathbf{P}_\sigma$. It has the same coefficients as $\mathbf{W}$ below order $r$ and at order $r+1$, and its coefficient of order $r$ is $\mathbf{w}_r + a\mathbf{p} + b\mathbf{w}_1$.*

*Proof.* Put $u_\psi = T\psi'/\psi$. It has constant term 1, so $\widetilde{\mathbf{W}}$ is a local representative of $\mathbf{P}_\sigma$. Since $\psi'' = 0$ in characteristic two, $Tu_\psi' = u_\psi + u_\psi^2$. Using $T\psi' = u_\psi \psi$ and the identity for $\mathbf{W}$ at $\psi$, we get

$$\begin{aligned} T\widetilde{\mathbf{W}}' &= (u_\psi + u_\psi^2)\,\mathbf{W}(\psi) + u_\psi^2\,\psi\mathbf{W}'(\psi) \\ &= (u_\psi + u_\psi^2)\,\mathbf{W}(\psi) + u_\psi^2(\mathbf{W}(\psi) + \mathbf{W}(\psi)^{\circ 2}) = \widetilde{\mathbf{W}} + \widetilde{\mathbf{W}}^{\circ 2}, \end{aligned}$$

so $\widetilde{\mathbf{W}}$ is normalized. Since $r$ is odd, $T\psi' = T + bT^r$, and

$$u_\psi = \frac{1 + bT^{r-1}}{1 + bT^{r-1} + aT^r} = 1 + aT^r + O(T^{r+2}).$$

In $\mathbf{W}(\psi)$, the term $\mathbf{w}_1\psi$ contributes $b\mathbf{w}_1 T^r + a\mathbf{w}_1 T^{r+1}$. The term $\mathbf{w}_2\psi^2 = \mathbf{w}_2(T^2 + b^2T^{2r} + a^2T^{2r+2})$ changes nothing below order $2r$, and the terms $\mathbf{w}_e\psi^e$ with $e \geq 3$ change only orders $e + r - 1 \geq r + 2$. Multiplication by $u_\psi$ adds $a\mathbf{p}T^r + a\mathbf{w}_1 T^{r+1}$ up to terms of order $r+2$. At order $r$ the total change is $a\mathbf{p} + b\mathbf{w}_1$, and at order $r+1$ the two

contributions $a\mathbf{w}_1$ cancel. □

**Proposition 10** (Normalized continuation)**.** *Let* $(\mathbf{p}, \mathbf{v}_1)$ *be a local jet of order one of a branch* $\mathbf{P}_\sigma$*, and suppose that* $\ker \mathbf{A}_\Gamma = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$*. Put* $\mathbf{v}_2 = \mathbf{v}_1^{\circ 2}$*. For odd* $r = 3, 5, \ldots$*, solve the system formed by* (22) *and* (25) *for* $\mathbf{v}_r$*, and put* $\mathbf{v}_{r+1} = \mathbf{v}_{(r+1)/2}^{\circ 2}$*. Then every step is consistent, and every resulting sequence* $\mathbf{p}, \mathbf{v}_1, \mathbf{v}_2, \ldots$ *is the sequence of coefficients of one normalized local representative of* $\mathbf{P}_\sigma$*.*

*Proof.* By Proposition 9, $\mathbf{p}, \mathbf{v}_1, \mathbf{v}_1^{\circ 2}$ are the first coefficients of a normalized local representative. Suppose that $\mathbf{p}, \mathbf{v}_1, \ldots, \mathbf{v}_{r-1}$, with $r \geq 3$ odd, are the first coefficients of a normalized local representative $\mathbf{W}$. Its coefficient of order $r+1$ is $\mathbf{v}_{(r+1)/2}^{\circ 2}$, so its coefficient $\mathbf{w}_r$ of order $r$ solves the system, and the solutions form $\mathbf{w}_r + \ker \mathbf{A}_\Gamma = \mathbf{w}_r + \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$. By Lemma 6, each solution is the coefficient of order $r$ of a normalized local representative with the same coefficients below order $r$ and at order $r+1$. This proves the induction step. As in Proposition 6, the successive changes converge coefficientwise and give one normalized local representative for the whole sequence. □

By Lemma 6, $\operatorname{span}\{\mathbf{p}, \mathbf{v}_1\} \subseteq \ker \mathbf{A}_\Gamma$, so the hypothesis is equivalent to $\operatorname{rank} \mathbf{A}_\Gamma = k - 2$ and is checked once. Since $\ker \mathbf{A}_\Gamma \subseteq \ker \mathbf{A}$, Proposition 10 applies whenever Proposition 7 does, so the additional equations can only help. The sufficient condition of Corollary 1 also has a binary Goppa version.

**Corollary 2.** *Suppose that* $\ker \mathbf{J} = E_i$ *and that the restrictions to* $\ker \mathbf{J}$ *of the forms* $Q_h$*,* $h \in \mathcal{H}$*, span over* $\mathbb{K}$ *a space of dimension* $\binom{m}{2}$*. Then* $\ker \mathbf{A}_\Gamma = \operatorname{span}\{\mathbf{p}, \mathbf{v}_1\}$ *for every tangent vector* $\mathbf{v}_1$ *of a branch.*

*Proof.* The proof of Corollary 1 applies with $\mathcal{H}$ in place of $\mathcal{H}^0$. The forms $Q_h$, $h \in \mathcal{H}$, belong to the first-order system by Proposition 8, so their restrictions lie in the span of the monomials $z_\rho z_\pi$. For $\mathbf{x} \in \ker \mathbf{J}$ we still have $B_h(\mathbf{p}, \mathbf{x}) = (d-1)\nabla h(\mathbf{p})\,\mathbf{x} = 0$. □

**Remark 15** (Wild Goppa codes)**.** *For a wild Goppa code* $\Gamma(\boldsymbol{\alpha}, g^{q-1}) = \Gamma(\boldsymbol{\alpha}, g^q)$ *with* $q > 2$*, the proofs of Lemma 5 and Proposition 9 apply with* $G^2$ *replaced by* $g^q$*, with the signs of odd characteristic. Every tangent vector of a branch is the first coefficient of a local representative with* $\mathbf{v}_{qr} = \varphi(\mathbf{v}_r)$ *for all* $r \geq 1$*. This relation fixes only the coefficients of orders divisible by* $q$*. The proof of Proposition 10 does not carry over directly, so we do not use it.*

## 5. Bootstrapping

The previous section established how to recover a deep local jet of one branch of the curve at a single held-out position. While the polynomials from the original hold-out kernel already vanish on the curve, they give no linear equations for the tangents at the retained positions, because the distinguisher forced their gradients to zero there (Remark 8).

This section introduces *bootstrapping*: using the recovered jet to efficiently construct lower-degree polynomials that vanish on the same branch but have non-trivial gradients at all positions.

The method relies on changing how the vanishing conditions are imposed. In the distinguisher, requiring the Hasse derivatives in all directions to vanish at every retained column created a large number of linear constraints, forcing the use of polynomials of a large degree. By contrast, the known jet provides necessary and sufficient linear conditions for a high-multiplicity root at the held-out position: only $s$ conditions for a root of multiplicity $s$ (Section 5.1).

Combined with simple evaluations $h(\mathbf{y}_j) = 0$ at the remaining public columns, these conditions along the curve are sufficient to recover the full space of equations of a given degree of the branch (Section 5.2). Because this new system is far less restrictive, we can drop the polynomial degree significantly—specifically, down to quadratic forms. Evaluating their gradients provides, under a rank condition, aligned tangent planes at every public column (Section 5.3), setting up the minor-code attack of Section 6.

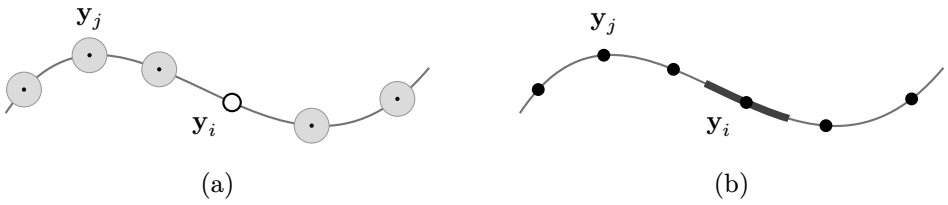



Figure 5: Vanishing conditions imposed on $h$. (a) The hold-out kernel requires multiplicity at least $s$ in all directions at the retained columns $\mathbf{y}_j$ (shaded disks), up to $M_s$ linear conditions each, and nothing at the held-out column $\mathbf{y}_i$. (b) Bootstrapping requires $h$ to vanish at every public column and, at $\mathbf{y}_i$, to vanish to order at least $r+1$ along the branch of the jet (thick arc), using a local jet of order $r$; these are $n+r$ linear conditions.

### 5.1. Multiplicity conditions from local jets

We begin by turning a recovered local jet into vanishing conditions at its position. Although known only up to scaling and reparametrization (Definition 4), a jet of order $r$ provides necessary and sufficient conditions for $h$ to vanish to order at least $s \le r+1$ along its branch.

**Lemma 7.** *Let $(\mathbf{v}_0, \dots, \mathbf{v}_r)$ be a local jet of order $r$ of $\mathbf{P}$ at position $i$ (i.e., the first $r+1$ coefficients of a local representative $u(T)\,\mathbf{P}(\psi(T))$), and put $\mathbf{V}(T) = \mathbf{v}_0 + \mathbf{v}_1 T + \cdots + \mathbf{v}_r T^r$. For every homogeneous $h \in \mathbb{K}[X_1, \dots, X_k]$ of degree $d$ and every $s$ with $1 \le s \le r+1$,*

$$(Z - \alpha_i)^s \mid h(\mathbf{F}(Z)) \quad \Longleftrightarrow \quad T^s \mid h(\mathbf{V}(T)). \tag{26}$$

*Proof.* By Definition 4, $\mathbf{V}$ consists of the first $r+1$ coefficients of a local representative $\mathbf{W}(T) = u(T)\,\mathbf{P}(\psi(T))$ with $u(0) = 1$, $\psi(0) = 0$ and $\psi'(0) \neq 0$. Since $\mathbf{V} \equiv \mathbf{W} \pmod{T^{r+1}}$

and $s \le r+1$, $T^s$ divides $h(\mathbf{V})$ if and only if it divides $h(\mathbf{W}) = u^d\, h(\mathbf{P}(\psi))$. Multiplying by the unit $u^d$ does not change the order of a series at $T = 0$, and neither does substituting $\psi$: if $f = T^e f_1$ with $f_1(0) \neq 0$, then $f(\psi) = \psi^e f_1(\psi)$ again has order $e$. Hence $T^s$ divides $h(\mathbf{W})$ if and only if it divides $h(\mathbf{P}(T)) = \lambda_i^d\, h(\mathbf{F}(\alpha_i + T))$, that is, if and only if $(Z - \alpha_i)^s$ divides $h(\mathbf{F}(Z))$. □

The right-hand side of (26) says that the coefficients of $T^0, \dots, T^{s-1}$ in $h(\mathbf{V}(T))$ vanish. These are $s$ linear equations, not necessarily independent, in the coefficients of $h$, computed from the jet alone: they do not involve $u$ or $\psi$. The equation for $T^0$ is $h(\mathbf{y}_i) = 0$, since $\mathbf{v}_0 = \mathbf{y}_i$.

**Example 1.** For $r \ge 2$, write $\mathbf{p} = \mathbf{v}_0$. The conditions for multiplicity at least three are

$$h(\mathbf{p}) = 0, \qquad \nabla h(\mathbf{p})\, \mathbf{v}_1 = 0, \qquad \nabla h(\mathbf{p})\, \mathbf{v}_2 + Q_h(\mathbf{v}_1) = 0,$$

where $Q_h$ is the quadratic part of $h$ at $\mathbf{p}$, as in Section 4.4. Here the jet vectors are known and the coefficients of $h$ are the unknowns, so all three equations are linear in these unknowns.

For instance, consider the curve $\mathbf{F}(Z) = (1, Z, Z^2)^\mathsf{T}$ at $Z = 0$. A local jet of order two is $\mathbf{V}(T) = (1, T, T^2)^\mathsf{T}$. Write a quadratic form as

$$h = c_{11}X_1^2 + c_{12}X_1X_2 + c_{13}X_1X_3 + c_{22}X_2^2 + c_{23}X_2X_3 + c_{33}X_3^2.$$

Substitution gives

$$h(\mathbf{V}(T)) = c_{11} + c_{12}T + (c_{13} + c_{22})T^2 + c_{23}T^3 + c_{33}T^4.$$

Thus multiplicity at least three imposes exactly

$$c_{11} = 0, \qquad c_{12} = 0, \qquad c_{13} + c_{22} = 0.$$

In characteristic two, the square term $X_2^2$ still contributes to the third equation, although its gradient is zero. The conditions do not force the whole gradient of $h$ to vanish: for instance, $h = X_1X_3 - X_2^2$ satisfies them and has $\nabla h(1, 0, 0) = (0, 0, 1)$. △

**Remark 16.** *For non-subfield subcodes ($m = 1$), the Frobenius map $\varphi$ is the identity on $\mathbb{K} = \mathbb{F}_q$, so $\mathbf{P}$ is the only branch at $\mathbf{y}_i$ (Section 4.3). For subfield subcodes, the recovered jet may belong to a Frobenius conjugate branch $\mathbf{P}_\sigma = \varphi^\sigma(\mathbf{P})$ with $\sigma \neq 0$. Since $\mathbf{y}_j \in \mathbb{F}_q^k$, we have $\mathbf{y}_j = \varphi^\sigma(\lambda_j)\mathbf{F}^{(\sigma)}(\varphi^\sigma(\alpha_j))$ for all $j$. The conjugate data $\varphi^\sigma(\boldsymbol{\alpha})$, $\varphi^\sigma(\boldsymbol{\lambda})$ and $\mathbf{F}^{(\sigma)}$ therefore describe the same public columns, and in this description $\mathbf{P}_\sigma$ is the branch $\mathbf{P}_0$. Lemma 7 then gives*

$$\big(Z - \varphi^\sigma(\alpha_i)\big)^s \mid h(\mathbf{F}^{(\sigma)}(Z)) \quad \iff \quad T^s \mid h(\mathbf{V}(T)),$$

*with the same equations on the right, so $\sigma$ need not be known. In the rest of this section we assume that the jet belongs to $\mathbf{P}$. Replacing $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$ and $\mathbf{F}$ by their conjugates changes the recovered key only by a power of the Frobenius automorphism.*

### 5.2. Quadratic forms from a local jet

Our aim is to compute all quadratic forms that vanish on $\mathbf{F}$, using the public columns and one recovered local jet. Quadratic forms are the first possible equations of the curve: a nonzero linear form vanishing on $\mathbf{F}$ would give a linear relation among the rows of $\mathbf{Y}$. We work over $\mathbb{K} = \mathbb{F}_{q^m}$ and denote the space of quadratic forms with $h(\mathbf{F}(Z)) \equiv 0$ by $\mathcal{J}$.

Let $(\mathbf{v}_0, \dots, \mathbf{v}_r)$ be the recovered local jet of $\mathbf{P}$ at position $i$, and put

$$\mathbf{V}(T) = \mathbf{v}_0 + \mathbf{v}_1 T + \cdots + \mathbf{v}_r T^r.$$

Write the unknown quadratic form as

$$h(\mathbf{X}) = \sum_{1 \le a \le b \le k} c_{ab} X_a X_b, \qquad c_{ab} \in \mathbb{K},$$

including the square monomials $X_a^2$, also in characteristic two. We require $h$ to vanish at every public column and to vanish to order at least $r+1$ along the jet:

$$\begin{aligned} h(\mathbf{y}_j) &= 0 \quad \text{for } 1 \le j \le n, \\ [T^e]\, h(\mathbf{V}(T)) &= 0 \quad \text{for } 1 \le e \le r. \end{aligned} \tag{27}$$

The coefficient of $T^0$ is already covered by $h(\mathbf{y}_i) = 0$, since $\mathbf{v}_0 = \mathbf{y}_i$. These are $n + r$ linear equations in the $\binom{k+1}{2}$ unknowns $c_{ab}$, computed from $\mathbf{Y}$ and the jet alone. The equations need not be independent.

**Proposition 11.** *If $n + r > 2D$, then the solution space of (27) is exactly $\mathcal{J}$.*

*Proof.* Every $h \in \mathcal{J}$ satisfies (27) by homogeneity and Lemma 7. Conversely, suppose that a quadratic form $h$ satisfies these conditions and put $g(Z) = h(\mathbf{F}(Z))$, so that $\deg g \le 2D$. Since $h(\mathbf{y}_j) = \lambda_j^2 g(\alpha_j)$ and $\lambda_j \neq 0$, the evaluation conditions give a root of $g$ at each $\alpha_j$. At $\alpha_i$, the jet conditions increase the multiplicity to at least $r+1$ by Lemma 7. Thus $g$ has at least $(n-1) + (r+1) = n + r > 2D$ roots counted with multiplicity, and $g = 0$. □

A jet of order $r \ge \max\{0, 2D - n + 1\}$ therefore suffices to recover $\mathcal{J}$. Its dimension satisfies

$$\dim_{\mathbb{K}} \mathcal{J} \ge \binom{k+1}{2} - (2D+1). \tag{28}$$

Indeed, $\mathcal{J}$ is the kernel of the linear map $h \mapsto h(\mathbf{F}(Z))$ from quadratic forms to polynomials of degree at most $2D$, a space of dimension $2D+1$.

**Remark 17.** *For subfield subcodes, solving over $\mathbb{K}$ is essential. A form with coefficients in $\mathbb{F}_q$ that vanishes on $\mathbf{F}$ also vanishes on every conjugate curve. Allowing coefficients in $\mathbb{K}$ removes this restriction: the jet selects the curve whose quadratic equations we recover. Their gradients need not annihilate the tangents of the other branches. Under the rank condition of Section 5.3, they determine the tangent planes of this same curve at every public column.*

### 5.3. Tangent planes from quadratic forms

The forms in $\mathcal{J}$ give linear equations for the tangent plane at each public column $\mathbf{y}_j$, collected in the matrix $\mathbf{J}_j$ whose rows are $\nabla h(\mathbf{y}_j)$ for a basis of $\mathcal{J}$. The gradient argument of Section 4.1 gives

$$V_j^{[1]} = \operatorname{span}\{\mathbf{y}_j, \mathbf{F}'(\alpha_j)\} \subseteq \ker \mathbf{J}_j. \tag{29}$$

As in Section 4.1, we assume that $\mathbf{y}_j$ and $\mathbf{F}'(\alpha_j)$ are linearly independent, so that $V_j^{[1]}$ is a plane. Then (29) is an equality if and only if $\operatorname{rank} \mathbf{J}_j = k - 2$.

**Heuristic 2.** For random codes of the families considered here, if $\dim_{\mathbb{K}} \mathcal{J}$ is sufficiently larger than $k-2$, then $\operatorname{rank} \mathbf{J}_j = k-2$ at each public column $\mathbf{y}_j$ with probability close to 1.

At every column with $\operatorname{rank} \mathbf{J}_j = k-2$, the kernel gives the tangent plane. These planes are aligned: all belong to the curve selected by the recovered jet, so the unknown Frobenius power of Remark 16 is the same at every position. Unlike in Section 4.4, no branches have to be separated at the other positions.

The rank condition is checked by one rank computation at each column, so failures are detected. It needs $\dim_{\mathbb{K}} \mathcal{J} \geq k-2$, which by (28) holds when $\binom{k}{2} \geq 2D-1$, and it fails at columns lying on a trisecant line of the curve. At such columns, forms of higher degree can be used as described in the next remark.

**Remark 18.** *If the quadratic equations do not give the required gradient rank, we can use forms of higher degree, for instance cubic forms. The same construction and proof recover all homogeneous degree-$\delta$ equations of the curve when $n + r > \delta D$. The system still has $n + r$ equations, now in $\binom{k+\delta-1}{\delta}$ unknowns, and a jet of order $r \geq \max\{0, \delta D - n + 1\}$ suffices.*

### 5.4. Other variants of bootstrapping

The required contact order can be distributed among several positions. Given local jets of orders $r_i$ at $i \in S$, all on the same Frobenius conjugate of the curve, impose their coefficient conditions together with the public-point equations. The bound of Proposition 11 becomes

$$n + \sum_{i \in S} r_i > 2D.$$

If only the Frobenius closures $E_i$ of the tangent planes are known, solve instead for quadratic forms over $\mathbb{F}_q$ satisfying

$$h(\mathbf{y}_j) = 0 \quad \text{for } 1 \leq j \leq n, \qquad \nabla h(\mathbf{y}_i)\, \mathbf{v} = 0 \quad \text{for all } i \in S \text{ and } \mathbf{v} \in E_i.$$

Every quadratic curve equation over $\mathbb{F}_q$ satisfies these conditions. Conversely, the conditions give at least $n + |S|$ roots of $h(\mathbf{F})$ counted with multiplicity, so $n + |S| > 2D$ suffices to recover all such equations. They vanish on all conjugate curves; under the rank conditions of Section 4, they allow branch separation and jet recovery. Section 8.2 reports a test of this variant.

Once $\mathcal{J}$ is recovered over $\mathbb{K}$, the continuation argument of Proposition 6 gives local jets of arbitrary order at every position where $\operatorname{rank} \mathbf{J}_j = k - 2$. These jets belong to the same selected curve and require no further hold-out computation.

## 6. Key recovery via the minor code

Section 5 gives the tangent planes $V_j^{[1]}$ at all public columns, all on the same Frobenius conjugate of the curve. In this section, we construct from these planes an auxiliary code—the *minor code*—which in general turns out to be a full GRS code with the same support as the public code. Applying the Sidelnikov–Shestakov attack to this code recovers an equivalent support; we then describe how to recover the GRS multiplier and any support elements removed by preliminary shortening.

Note that the minor-code construction applies without rate limitations in characteristic 2; in odd characteristic it requires the ambient GRS rate $(D+1)/n$ to be below about $1/2$.

### 6.1. Minor code

Let $\mathbb{K}$ be the field of the ambient GRS code (i.e., $\mathbb{K} = \mathbb{F}_q$ in the non-subfield case, and $\mathbb{K} = \mathbb{F}_{q^m}$ in the subfield-subcode case). We use $n, k, D$ for the parameters after any preliminary shortening. In the subfield case, we absorb the common Frobenius conjugation into the hidden data as in Remark 16. In both cases, the known tangent planes are

$$V_j^{[1]} = \operatorname{span}_{\mathbb{K}}\{\mathbf{F}(\alpha_j), \mathbf{F}'(\alpha_j)\}, \qquad \dim_{\mathbb{K}} V_j^{[1]} = 2 \quad \text{for } 1 \le j \le n.$$

For each position $j$, choose $\hat{\mathbf{y}}_j \in V_j^{[1]} \setminus \operatorname{span}_{\mathbb{K}}\{\mathbf{y}_j\}$, so that $[\mathbf{y}_j \mid \hat{\mathbf{y}}_j]$ is a basis matrix of the plane. We collect its $2 \times 2$ minors into a single column, as illustrated in Figure 6.

**Definition 5.** The *minor matrix* $\mathbf{M} \in \mathbb{K}^{\binom{k}{2} \times n}$ has rows indexed by $1 \le a < b \le k$ and entries

$$m_{(a,b),j} = \det \begin{pmatrix} y_{a,j} & \hat{y}_{a,j} \\ y_{b,j} & \hat{y}_{b,j} \end{pmatrix} = y_{a,j}\hat{y}_{b,j} - y_{b,j}\hat{y}_{a,j}. \tag{30}$$

Define

$$\mathbf{T} = \begin{cases} \sqrt{\mathbf{M}}, & \operatorname{char} \mathbb{K} = 2, \\ \mathbf{M}, & \operatorname{char} \mathbb{K} \neq 2, \end{cases} \qquad \mathcal{T} = \operatorname{RowSpan}_{\mathbb{K}}(\mathbf{T}).$$

In characteristic two, the square root denotes entrywise application of the inverse of the squaring automorphism of $\mathbb{K}$. We call $\mathcal{T}$ the *minor code*.

**Remark 19.** *The minor code depends only on the tangent planes, up to nonzero coordinate scalings. Indeed, put* $\mathbf{B}_j = [\mathbf{y}_j \mid \hat{\mathbf{y}}_j]$. *Any other basis matrix* $\widetilde{\mathbf{B}}_j$ *of* $V_j^{[1]}$ *satisfies*

$$\widetilde{\mathbf{B}}_j = \mathbf{B}_j \mathbf{A}_j, \qquad \mathbf{A}_j \in \mathrm{GL}_2(\mathbb{K}).$$

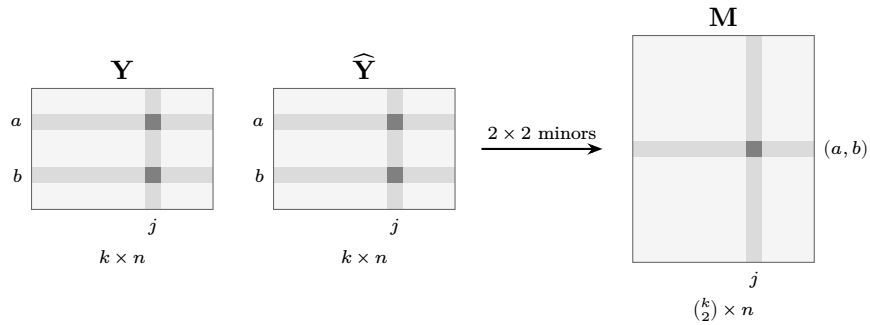



Figure 6: The minor matrix. Rows $a$ and $b$ of $\mathbf{Y}$ and $\widehat{\mathbf{Y}} = [\hat{\mathbf{y}}_1 \mid \cdots \mid \hat{\mathbf{y}}_n]$ give row $(a, b)$ of $\mathbf{M}$: its entry in column $j$ is the $2 \times 2$ determinant formed by the four dark entries.

*Restricting to rows $a, b$ and using multiplicativity of the determinant gives*

$$\widetilde{m}_{(a,b),j} = \det(\mathbf{A}_j)\, m_{(a,b),j}.$$

*Thus column $j$ of $\mathbf{T}$ is multiplied by $\sqrt{\det(\mathbf{A}_j)}$ in characteristic two, and by $\det(\mathbf{A}_j)$ otherwise. In either case, this is a nonzero column scaling.*

The following proposition identifies a GRS code containing $\mathcal{T}$ and explains the use of square roots in characteristic two.

**Proposition 12.** *The minor code $\mathcal{T}$ is contained in a GRS code with support $\boldsymbol{\alpha}$, of dimension $D$ if* $\operatorname{char} \mathbb{K} = 2$ *and of dimension* $\min\{2D-1, n\}$ *otherwise.*

*Proof.* Apply the change-of-basis argument of Remark 19 to the chosen basis and the hidden basis $(\mathbf{F}(\alpha_j), \mathbf{F}'(\alpha_j))$. Write

$$[\mathbf{y}_j \mid \hat{\mathbf{y}}_j] = [\mathbf{F}(\alpha_j) \mid \mathbf{F}'(\alpha_j)]\, \mathbf{A}_j, \tag{31}$$

where $\mathbf{A}_j \in \mathrm{GL}_2(\mathbb{K})$, and put $\rho_j = \det(\mathbf{A}_j) \neq 0$. Taking minors gives

$$m_{(a,b),j} = \rho_j\, \omega_{a,b}(\alpha_j), \qquad \omega_{a,b} = f_a f_b' - f_b f_a'. \tag{32}$$

Since $\deg f_a, \deg f_b \leq D$ and the coefficient of $Z^{2D-1}$ cancels, $\deg \omega_{a,b} \leq 2D-2$. This proves the odd-characteristic inclusion; when $2D-1 \geq n$, the ambient evaluation code is all of $\mathbb{K}^n$.

In characteristic two, $\omega_{a,b} = (f_a f_b)'$ contains only even powers of $Z$. It therefore has a unique polynomial square root $g_{a,b} \in \mathbb{K}[Z]$, with

$$\omega_{a,b} = g_{a,b}^2, \qquad \deg g_{a,b} \leq D-1.$$

Put $\theta_j = \sqrt{\rho_j}$. Taking square roots in (32) yields

$$(\mathbf{T})_{(a,b),j} = \theta_j \, g_{a,b}(\alpha_j), \tag{33}$$

which proves the characteristic-two inclusion. □

**Fullness.** The matrix $\mathbf{T}$ contains $\binom{k}{2}$ rows, all belonging to the ambient GRS code of Proposition 12 (of dimension $D$ in characteristic two, $\min\{2D-1, n\}$ in odd characteristic, and $D-t$ for binary Goppa codes by Proposition 13). Because $\binom{k}{2}$ is generally much larger than this dimension, $\mathbf{T}$ provides ample generators to span this GRS code. Fullness is checked by a single rank computation: $\mathcal{T}$ equals the ambient GRS code if and only if $\operatorname{rank}\mathbf{T}$ attains the upper bound.

**Heuristic 3.** For random codes in the families considered here, suppose that all tangent planes are two-dimensional and that $\binom{k}{2}$ is sufficiently larger than the applicable upper bound on $\dim_{\mathbb{K}}\mathcal{T}$. Then, with high probability, $\mathcal{T}$ is a full GRS code with support $\boldsymbol{\alpha}$.

Our experiments suggest that the minor code is a full GRS code with probability almost 1 when the number of rows of the minor matrix exceeds the dimension of the ambient GRS code.

Once $\mathcal{T}$ is confirmed to be a full GRS code of dimension $k_{\mathcal{T}} = \dim_{\mathbb{K}}\mathcal{T}$, the Sidelnikov–Shestakov attack (see Section 2.2) applied to a row basis of $\mathbf{T}$ recovers an equivalent support, provided $2 \le k_{\mathcal{T}} \le n-2$.

**Remark 20.** *In odd characteristic, generic instances have $k_{\mathcal{T}} = \min\{2D-1, n\}$. The condition $k_{\mathcal{T}} \le n-2$ requires $2D-1 \le n-2$, which restricts the ambient rate $(D+1)/n$ to at most roughly $1/2$. In characteristic two, the dimension bound $D$ imposes no such rate restriction.*

### 6.2. Minor code of a binary Goppa code

For binary square-free Goppa codes, the ambient GRS code of the minor code has an even smaller dimension.

**Proposition 13.** *Let $C = \Gamma(\boldsymbol{\alpha}, G)$ be a binary square-free Goppa code, with $\deg G = t$, and use the representation of Proposition 2, so that $D = n - 2t - 1$. If $D > t$, then $\mathcal{T}$ is contained in a GRS code with support $\boldsymbol{\alpha}$ and dimension $D - t$.*

*Proof.* Put $\Pi = \Pi_{\boldsymbol{\alpha}}$. The interpolation identity in the proof of Lemma 5 gives, for each row $a$,

$$R_a(Z) := \frac{G(Z)^2 f_a(Z)}{\Pi(Z)} = \sum_{j=1}^{n} \frac{y_{a,j}}{Z - \alpha_j}.$$

Since $y_{a,j} \in \mathbb{F}_2$, we have $R_a' = R_a^2$. Clearing denominators gives

$$\Pi f_a' + \Pi' f_a = G^2 f_a^2.$$

Multiplying this identity by $f_b$ and the analogous identity for $f_b$ by $f_a$, then adding, gives

$$\Pi\,\omega_{a,b} = G^2(f_a^2 f_b + f_a f_b^2).$$

As $\gcd(G, \Pi) = 1$, it follows that $G^2 \mid \omega_{a,b} = g_{a,b}^2$, and hence $G \mid g_{a,b}$. Write $g_{a,b} = G h_{a,b}$, where $\deg h_{a,b} < D - t$. Equation (33) becomes $(\mathbf{T})_{(a,b),j} = \eta_j h_{a,b}(\alpha_j)$ with $\eta_j = \theta_j G(\alpha_j)$, proving the inclusion. The multipliers $\eta_j$ are nonzero because $G(\alpha_j) \neq 0$. □

For binary Goppa codes, an explicit condition can take the place of Heuristic 3. The inclusion of Proposition 13 is an equality when the products $f_a f_b$ span a space of the largest possible dimension. In that case, the tangent planes are also two-dimensional at all positions, as Section 6.1 assumes.

**Proposition 14.** *Let $C$, $G$, $t$ and $D$ be as in Proposition 13, with $D > t$, and let*

$$W = \operatorname{span}_{\mathbb{K}}\{f_a f_b : 1 \le a \le b \le k\}.$$

*Then $\dim_{\mathbb{K}} W \le 2D - t + 1$. If equality holds, then $\dim_{\mathbb{K}} V_j^{[1]} = 2$ for $1 \le j \le n$, and the minor code $\mathcal{T}$ is a GRS code with support $\boldsymbol{\alpha}$ and dimension $D - t$.*

*Proof.* Each product $f_a f_b$ has degree at most $2D$. In characteristic two, $(f_a^2)' = 0$ and $(f_a f_b)' = \omega_{a,b}$ for $a < b$, so differentiation maps $W$ onto the span of the polynomials $\omega_{a,b}$. The kernel of this map is contained in the space of polynomials of degree at most $2D$ with only even powers of $Z$, which has dimension $D + 1$.

By the proof of Proposition 13, $\omega_{a,b} = G^2 h_{a,b}^2$ with $\deg h_{a,b} < D - t$. Squaring is an automorphism of $\mathbb{K}$, so the polynomials $G^2 A^2$ with $\deg A < D - t$ form a $\mathbb{K}$-vector space of dimension $D - t$. It contains every $\omega_{a,b}$, so

$$\dim_{\mathbb{K}} W \le (D + 1) + (D - t) = 2D - t + 1.$$

Suppose that equality holds. Then both bounds are attained, so the polynomials $\omega_{a,b}$ span this space. For every polynomial $A$ of degree less than $D - t$, we therefore have

$$G^2 A^2 = \sum_{a<b} \zeta_{a,b}\,\omega_{a,b} \quad \text{for some } \zeta_{a,b} \in \mathbb{K}. \tag{34}$$

Take $A = 1$. At $Z = \alpha_j$, the left-hand side of (34) is $G(\alpha_j)^2 \neq 0$, so $\omega_{a,b}(\alpha_j) \neq 0$ for some $a < b$. This value is the minor of rows $a$ and $b$ of the matrix $[\mathbf{F}(\alpha_j) \mid \mathbf{F}'(\alpha_j)]$. The matrix therefore has rank two, and $\dim_{\mathbb{K}} V_j^{[1]} = 2$.

Now let $A$ be arbitrary. Substituting $\omega_{a,b} = G^2 h_{a,b}^2$ in (34) gives

$$G^2 A^2 = G^2 \Big( \sum_{a<b} \sqrt{\zeta_{a,b}}\, h_{a,b} \Big)^2, \qquad \text{and hence} \qquad A = \sum_{a<b} \sqrt{\zeta_{a,b}}\, h_{a,b}.$$

The polynomials $h_{a,b}$ therefore span all polynomials of degree less than $D - t$. By the proof of Proposition 13, row $(a, b)$ of $\mathbf{T}$ is $(\eta_j h_{a,b}(\alpha_j))_{1 \le j \le n}$ with $\eta_j \neq 0$. Hence $\mathcal{T} = \mathrm{GRS}_{D-t}(\boldsymbol{\alpha}, \boldsymbol{\eta})$, where $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$. □

In the synthetic experiments of Section 8.2, the condition $\dim_{\mathbb{K}} W = 2D - t + 1$ held for every shortened Classic McEliece code on which we tested it. The same condition guarantees that the quadratic forms required for direct recovery exist (Proposition 17).

**Remark 21.** *As noted in Section 4.6, shortening a binary Goppa code gives the binary Goppa code with the same Goppa polynomial on the remaining support. Propositions 13 and 14 therefore apply to a shortened code, with $n$, $D = n - 2t - 1$ and $\mathbf{F}$ taken for the shortened code.*

For a binary Goppa code with $\dim_{\mathbb{K}} W = 2D - t + 1$, the rank condition of Heuristic 2 holds at a public column if and only if the code shortened at that column satisfies the same dimension condition. This condition is easy to check experimentally for a code family, even at shortened Classic McEliece sizes: $W$ and $\widetilde{W}$ are computed from the secret key of a randomly generated code.

**Proposition 15.** *Let $C$, $G$, $t$, $D$ and $W$ be as in Proposition 14, and suppose that $\dim_{\mathbb{K}} W = 2D - t + 1$. Let $1 \le j \le n$, and let $\widetilde{W}$ be the space $W$ defined for the code obtained by shortening $C$ at position $j$ (Remark 21). Then the matrix $\mathbf{J}_j$ of Section 5.3 has rank $k - 2$ if and only if $\dim_{\mathbb{K}} \widetilde{W} = 2(D-1) - t + 1$.*

*Proof.* The space $W$ and the rank of $\mathbf{J}_j$ do not depend on the choice of the generator matrix of $C$. Since $\dim_{\mathbb{K}} V_j^{[1]} = 2$ by Proposition 14, $\mathbf{y}_j \neq \mathbf{0}$, and we may choose the generator matrix so that $\mathbf{y}_j$ is the first unit vector. A quadratic form $h = \sum_{a \le b} c_{ab} X_a X_b$ in $\mathcal{J}$ has $c_{11} = h(\mathbf{y}_j) = 0$, and therefore

$$\nabla h(\mathbf{y}_j) = (0, c_{12}, \dots, c_{1k}).$$

Hence $\nabla h(\mathbf{y}_j) = \mathbf{0}$ if and only if $h$ is a form in $X_2, \dots, X_k$. The forms of $\mathcal{J}$ in these variables are the kernel of the map $h \mapsto h(f_2, \dots, f_k)$ on the quadratic forms in $X_2, \dots, X_k$. Its image is

$$W_j = \operatorname{span}_{\mathbb{K}} \{ f_a f_b : 2 \le a \le b \le k \},$$

so the space of these forms has dimension $\binom{k}{2} - \dim_{\mathbb{K}} W_j$. Similarly, $\mathcal{J}$ is the kernel of the map $h \mapsto h(\mathbf{F}(Z))$, whose image is $W$, so $\dim_{\mathbb{K}} \mathcal{J} = \binom{k+1}{2} - \dim_{\mathbb{K}} W$. The rank of $\mathbf{J}_j$ is $\dim_{\mathbb{K}} \mathcal{J}$ minus the dimension of the space of forms $h \in \mathcal{J}$ with $\nabla h(\mathbf{y}_j) = \mathbf{0}$. Therefore

$$\operatorname{rank} \mathbf{J}_j = k - \dim_{\mathbb{K}} W + \dim_{\mathbb{K}} W_j. \tag{35}$$

After column $j$ is deleted, rows $2, \dots, k$ of the generator matrix generate the shortened code. For $a \ge 2$, the polynomial $f_a$ vanishes at $\alpha_j$, so $f_a = (Z - \alpha_j)\tilde{f}_a$ with $\deg \tilde{f}_a \le D - 1$. Put $\Pi = \Pi_{\boldsymbol{\alpha}}$ and $\widetilde{\Pi} = \Pi/(Z - \alpha_j)$. For every position $j' \neq j$,

$$y_{a,j'} = \frac{G(\alpha_{j'})^2}{\Pi'(\alpha_{j'})} (\alpha_{j'} - \alpha_j) \tilde{f}_a(\alpha_{j'}) = \frac{G(\alpha_{j'})^2}{\widetilde{\Pi}'(\alpha_{j'})} \tilde{f}_a(\alpha_{j'}).$$

Hence the $\tilde{f}_a$ are the polynomials of the shortened code in the representation of Proposition 2. Therefore $W_j = (Z - \alpha_j)^2 \widetilde{W}$ and $\dim_{\mathbb{K}} W_j = \dim_{\mathbb{K}} \widetilde{W}$. With $\dim_{\mathbb{K}} W = 2D - t + 1$,

(35) gives $\operatorname{rank} \mathbf{J}_j = k-2$ if and only if $\dim_{\mathbb{K}} \widetilde{W} = 2D-t-1 = 2(D-1)-t+1$. □

### 6.3. Recovering the full key

After the Sidelnikov–Shestakov attack recovers the support of the minor code, it remains to find a multiplier for the public code and, if preliminary shortening was used, to reconstruct the removed support elements.

Throughout this subsection, $n, k, D$ denote the parameters before any shortening, with $C \subseteq \mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ and $c = n-D-1$. All linear algebra below is over $\mathbb{K}$; for subfield subcodes, $C$ denotes the $\mathbb{K}$-linear span of the public code.

**Recovering the multiplier.** Assume first that no preliminary shortening was used, so the full support $\boldsymbol{\alpha}$ is already known. Let $\mathbf{G}_C$ be a generator matrix of $C$, and let $\mathbf{H}_{\mathrm{RS}}$ be a parity-check matrix of $\mathrm{RS}_{D+1}(\boldsymbol{\alpha})$. To find a valid multiplier, we solve the linear system

$$\mathbf{G}_C \operatorname{diag}(\mathbf{z}) \mathbf{H}_{\mathrm{RS}}^{\mathsf{T}} = \mathbf{0} \tag{36}$$

for a vector $\mathbf{z} \in (\mathbb{K}^*)^n$. This condition ensures that $C \operatorname{diag}(\mathbf{z}) \subseteq \mathrm{RS}_{D+1}(\boldsymbol{\alpha})$. Any solution $\mathbf{z}$ without zero coordinates yields a suitable multiplier via $\lambda_i = z_i^{-1}$, placing $C$ inside a GRS supercode of the required dimension (also note that this recovered multiplier does not necessarily equal the multiplier of the minor code itself).

**Recovering coordinates removed by shortening.** Now consider the case where the attack was mounted on a shortened code $C_J = \mathrm{Sh}_J(C)$, with $J$ comprising $\ell$ information-set positions of the public code. Let $I = [\![1, n]\!] \setminus J$ be the set of retained coordinates. We first compute the multiplier $\boldsymbol{\mu}$ of the corresponding ambient GRS code by applying (36) to the known shortened support $\boldsymbol{\alpha}_I$ and targeting an ambient dimension of $D+1-\ell$. Assuming this solution space is one-dimensional, $\boldsymbol{\mu}$ is determined up to a global scalar factor. The missing support elements are then recovered by lifting two specifically chosen dual codewords to their full length following ideas of [46, §3.3].

Let $\mathbf{H}_C \in \mathbb{F}_q^{(n-k)\times n}$ be a parity-check matrix of the public code. Extending scalars does not change the dimension, so the rows of $\mathbf{H}_C$ span over $\mathbb{K}$ the dual $C^\perp$ of the $\mathbb{K}$-linear span $C$; all duals below are taken over $\mathbb{K}$. Let $\mathbf{H}$ denote the restriction of $\mathbf{H}_C$ to the columns in $I$. By shortening–puncturing duality,

$$C_J^\perp = \mathrm{Pct}_J(C^\perp),$$

which means $\mathbf{H}$ generates $C_J^\perp$. Since $\dim C_J = k-\ell$, its dual has dimension $n-k$, implying that $\mathbf{H}$ retains full row rank. Consequently, every codeword in $C_J^\perp$ lifts uniquely to a codeword in $C^\perp$. To lift a word, one expresses it as a linear combination of the rows of $\mathbf{H}$ and then applies those same coefficients to the full matrix $\mathbf{H}_C$.

To recover the missing coordinates, we lift the two dual codewords corresponding to the first two rows of the Vandermonde parity-check matrix. Let $\boldsymbol{\nu} = \boldsymbol{\mu}^\perp$ be the dual multiplier of the shortened ambient code, so that $\mathrm{GRS}_{D+1-\ell}(\boldsymbol{\alpha}_I, \boldsymbol{\mu})^\perp = \mathrm{GRS}_c(\boldsymbol{\alpha}_I, \boldsymbol{\nu})$ by Section 2.1.

On the other hand,

$$\mathrm{Sh}_J(\mathrm{GRS}_{D+1}(\boldsymbol{\alpha},\boldsymbol{\lambda}))^{\perp} = \mathrm{Pct}_J(\mathrm{GRS}_c(\boldsymbol{\alpha},\boldsymbol{\lambda}^{\perp})) = \mathrm{GRS}_c(\boldsymbol{\alpha}_I,(\lambda_i^{\perp})_{i\in I}),$$

and a GRS code of dimension $c < n-\ell$ with a given support determines its multiplier up to a scalar. Hence $\boldsymbol{\nu} = (\lambda_i^{\perp})_{i\in I}$ after absorbing a common scalar into $\boldsymbol{\lambda}$. For $c \geq 2$, the vectors $\boldsymbol{\nu}$ and $(\nu_i\alpha_i)_{i\in I}$ belong to $C_J^{\perp}$ and are the restrictions of $\boldsymbol{\lambda}^{\perp}$ and $(\lambda_i^{\perp}\alpha_i)_{i=1}^n$.

We determine their unique lifts by solving

$$\mathbf{aH} = \boldsymbol{\nu}, \qquad \mathbf{bH} = (\nu_i\alpha_i)_{i\in I}, \tag{37}$$

and setting $\mathbf{u} = \mathbf{aH}_C$ and $\mathbf{v} = \mathbf{bH}_C$. By uniqueness, we must have $\mathbf{u} = \boldsymbol{\lambda}^{\perp}$ and $\mathbf{v} = (\lambda_i^{\perp}\alpha_i)_{i=1}^n$. The full support is then recovered via

$$\alpha_i = \frac{v_i}{u_i}, \quad \text{for } 1 \leq i \leq n, \tag{38}$$

and the multiplier is $\boldsymbol{\lambda} = \mathbf{u}^{\perp}$, the dual multiplier of $\mathbf{u}$ with respect to $\boldsymbol{\alpha}$ as in Section 2.1; indeed, the map $\boldsymbol{\beta} \mapsto \boldsymbol{\beta}^{\perp}$ is an involution.

If $u_j = 0$ for some $j$, then $\alpha_j = \infty$ in the recovered projective frame; since support elements are distinct, this happens for at most one $j$. Choose $p \in \mathbb{K}$ different from all $v_i/u_i$ with $u_i \neq 0$, replace $\boldsymbol{\alpha}_I$ by its image under $x \mapsto 1/(x-p)$, and repeat the multiplier and lifting steps. In the new frame all $n$ support elements are finite.

As a final verification step, we check that all recovered support elements are mutually distinct, that no $u_i$ is zero, and that

$$\mathbf{G}_C\mathbf{V}_c(\boldsymbol{\alpha},\mathbf{u})^{\mathsf{T}} = \mathbf{0}.$$

This confirms the inclusion $C \subseteq \mathrm{GRS}_{D+1}(\boldsymbol{\alpha},\boldsymbol{\lambda})$. If the system for $\boldsymbol{\mu}$ admits multiple independent solutions, each candidate is tested against this condition.

**Remark 22.** *An alternative approach is the support-extension technique utilized in the attack on the Ivanov–Krouk–Zyablov cryptosystem [77, Sec. 5.1]. For a removed coordinate $j \in J$, let $\mathbf{G}_j$ generate $\mathrm{Sh}_{J\setminus\{j\}}(C)$ with the $j$-th coordinate placed last, and set $N = n-\ell+1$. We test each candidate support value $w \in \mathbb{K} \setminus \{\alpha_i : i \in I\}$ by attempting to solve*

$$\mathbf{G}_j\,\mathrm{diag}(\boldsymbol{\eta})\,\mathbf{V}_c((\boldsymbol{\alpha}_I,w),\mathbf{1}_N)^{\mathsf{T}} = \mathbf{0}. \tag{39}$$

*The candidate $w$ is retained if and only if there exists a solution $\boldsymbol{\eta} \in (\mathbb{K}^*)^N$. The correct support value is guaranteed to pass this test; the rejection of all incorrect values relies on heuristic arguments. To account for a potentially removed point at infinity, one can either test it directly using the Vandermonde column $(0,\ldots,0,1)^{\mathsf{T}}$, or simply switch to a different affine representative. Once the full finite support is reconstructed, (36) cleanly recovers a multiplier for $C$.*

## 7. Direct key recovery

In this section, we present an alternative key-recovery method that bypasses both the minor code construction and the Sidelnikov–Shestakov attack. The method reconstructs the support directly from a single deep jet at one position $i$. Throughout this section, $n, k, D$ refer to the parameters after any preliminary shortening. Thus, if shortening was used, the recovered support covers only the retained positions; the removed support elements are recovered as in Section 6.3.

The approach finds two homogeneous polynomials $R$ and $S$ of the same degree $\delta$ (typically $\delta = 2$) such that $R(\mathbf{F}(Z))$ and $S(\mathbf{F}(Z))$ vanish at $\alpha_i$ to orders $\delta D$ and $\delta D - 1$, respectively. The quotient $R(\mathbf{y}_j)/S(\mathbf{y}_j)$ then equals $\mu(\alpha_j)$ for one Möbius transformation $\mu$ at every public column $\mathbf{y}_j$ with $j \neq i$, directly exposing an equivalent support. For binary Goppa codes, this construction needs the modification described in Section 7.4.

### 7.1. The Möbius quotient

Let $R, S \in \mathbb{K}[X_1, \dots, X_k]$ be homogeneous polynomials of degree $\delta$. Since each coordinate of $\mathbf{F}(Z)$ has degree at most $D$, both $R(\mathbf{F}(Z))$ and $S(\mathbf{F}(Z))$ are univariate polynomials in $\mathbb{K}[Z]$ of degree at most $\delta D$.

**Proposition 16.** *Suppose that*

$$(Z - \alpha_i)^{\delta D} \mid R(\mathbf{F}(Z)) \qquad \text{and} \qquad (Z - \alpha_i)^{\delta D - 1} \mid S(\mathbf{F}(Z)),$$

*with $R(\mathbf{F}(Z)) \not\equiv 0$. Then there exist constants $c_R \in \mathbb{K}^*$ and $a, b \in \mathbb{K}$ such that*

$$R(\mathbf{F}(Z)) = c_R (Z - \alpha_i)^{\delta D}, \qquad S(\mathbf{F}(Z)) = (Z - \alpha_i)^{\delta D - 1}(aZ + b). \tag{40}$$

*Furthermore, if $a\alpha_i + b \neq 0$, the rational function*

$$\mu(Z) := \frac{R(\mathbf{F}(Z))}{S(\mathbf{F}(Z))} = \frac{c_R (Z - \alpha_i)}{aZ + b} \tag{41}$$

*is a non-degenerate Möbius transformation. In addition, for every public column $\mathbf{y}_j = \lambda_j \mathbf{F}(\alpha_j)$ with $j \neq i$,*

$$\frac{R(\mathbf{y}_j)}{S(\mathbf{y}_j)} = \mu(\alpha_j), \tag{42}$$

*where the quotient is $\infty$ when $S(\mathbf{y}_j) = 0$; at the position $i$, $\mu(\alpha_i) = 0$.*

*Proof.* Since $\deg R(\mathbf{F}(Z)) \leq \delta D$ and $(Z - \alpha_i)^{\delta D}$ divides it, the quotient must be a constant $c_R$, which is nonzero because $R(\mathbf{F}) \not\equiv 0$. Similarly, dividing $S(\mathbf{F}(Z))$ by $(Z - \alpha_i)^{\delta D - 1}$ leaves a polynomial $aZ + b$ of degree at most one, proving (40).

The rational function $\mu(Z) = \frac{c_R(Z - \alpha_i)}{aZ + b}$ is a non-degenerate Möbius transformation because $c_R \neq 0$ and the root of the numerator is not a root of the denominator ($a\alpha_i + b \neq 0$).

Finally, since $R$ and $S$ are homogeneous of degree $\delta$, evaluating at $\mathbf{y}_j = \lambda_j \mathbf{F}(\alpha_j)$ gives

$$\frac{R(\mathbf{y}_j)}{S(\mathbf{y}_j)} = \frac{\lambda_j^\delta R(\mathbf{F}(\alpha_j))}{\lambda_j^\delta S(\mathbf{F}(\alpha_j))} = \mu(\alpha_j). \qquad \square$$

### 7.2. Linear constraints from the jet

As in Remark 16, we assume that the recovered jet belongs to $\mathbf{P}$. By Lemma 7, the conditions of Proposition 16 are equivalent to two nested linear systems on the $\binom{k+\delta-1}{\delta}$ unknown coefficients of degree-$\delta$ homogeneous forms:

$$W_R = \{h : [T^e]\, h(\mathbf{V}(T)) = 0 \text{ for } 0 \le e < \delta D\},$$
$$W_S = \{h : [T^e]\, h(\mathbf{V}(T)) = 0 \text{ for } 0 \le e < \delta D - 1\}.$$

Notice that $W_R \subseteq W_S$ is a subspace defined by the single extra equation $[T^{\delta D-1}]\, h(\mathbf{V}(T)) = 0$. Both systems are constructed from a jet of order $r \ge \delta D - 1$.

**Remark 23.** *To satisfy the nonvanishing hypotheses of Proposition 16, we choose $R \in W_R$ with $R(\mathbf{y}_j) \neq 0$ for some $j \neq i$. Indeed, $R(\mathbf{F}(Z))$ is a scalar multiple of $(Z - \alpha_i)^{\delta D}$, so this evaluation condition is equivalent to $R(\mathbf{F}(Z)) \not\equiv 0$.*

*We then choose $S \in W_S \setminus W_R$. Its jet coefficient of order $\delta D - 1$ is nonzero, so Lemma 7 implies that $S(\mathbf{F}(Z))$ vanishes at $\alpha_i$ to order exactly $\delta D - 1$. Hence $a\alpha_i + b \neq 0$, and $\mu$ is non-degenerate.*

### 7.3. Key recovery algorithm

Let $C$ denote the code used to recover the jet, after any preliminary shortening, and let $\mathbf{G}_C$ be a generator matrix of $C$. All linear algebra below is over $\mathbb{K}$; for subfield subcodes, $C$ denotes its $\mathbb{K}$-linear span. Assuming that the required forms $R$ and $S$ exist, an equivalent support and multiplier are reconstructed as follows:

1. Find polynomials $R$ and $S$ as in Remark 23.
2. Because $\mu(\alpha_i) = 0$, set $\beta_i = 0$. For each $j \neq i$, evaluate
$$\beta_j = \begin{cases} R(\mathbf{y}_j)/S(\mathbf{y}_j), & \text{if } S(\mathbf{y}_j) \neq 0 \\ \infty, & \text{if } S(\mathbf{y}_j) = 0. \end{cases}$$
3. If every entry of $\boldsymbol{\beta}$ is finite, set $\boldsymbol{\alpha}' = \boldsymbol{\beta}$. Otherwise, choose $p \in \mathbb{K} \setminus \{\beta_j : \beta_j \neq \infty\}$ and apply $\beta_j \mapsto (\beta_j - p)^{-1}$ to every entry, with $\infty \mapsto 0$, obtaining a finite support $\boldsymbol{\alpha}' \in \mathbb{K}^n$.
4. Let $\mathbf{H}_{\mathrm{RS}}$ be a parity-check matrix of $\mathrm{RS}_{D+1}(\boldsymbol{\alpha}')$. Solve the linear system
$$\mathbf{G}_C \operatorname{diag}(\mathbf{z}) \mathbf{H}_{\mathrm{RS}}^{\mathsf{T}} = \mathbf{0} \tag{43}$$

and choose a solution $\mathbf{z} \in (\mathbb{K}^*)^n$. Set $\lambda'_j = z_j^{-1}$ for $1 \le j \le n$. Then

$$C \subseteq \mathrm{GRS}_{D+1}(\boldsymbol{\alpha}', \boldsymbol{\lambda}').$$

This is (36) applied to the recovered support. By Proposition 16, $\boldsymbol{\alpha}'$ is the image of the secret support $\boldsymbol{\alpha}$ under a Möbius transformation. Since both supports are finite, the code $\mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ containing $C$ is also a GRS code with support $\boldsymbol{\alpha}'$, so a solution with no zero coordinates exists.

5. If preliminary shortening was used, apply the procedure of Section 6.3 to reconstruct the removed support elements and the complete multiplier.

### 7.4. Direct recovery for binary Goppa codes

For binary Goppa codes, we show below that the direct recovery procedure above cannot be applied with adjacent vanishing orders $\delta D$ and $\delta D - 1$. The reason for this is that the differential structure of the code rules out such quotients for forms of any degree.

However, the procedure can be adjusted for these codes by setting $\delta = 2$ and targeting vanishing orders $2D$ and $2D - 2$ instead. This forces the quotient to be the *square* of a Möbius transformation, from which an equivalent support is recovered via square roots in $\mathbb{K}$.

**The differential obstruction.** Let $C = \Gamma(\boldsymbol{\alpha}, G)$ be a binary square-free Goppa code with $t = \deg G \ge 1$, represented as in Proposition 2 with $D = n - 2t - 1$ and $\Pi = \Pi_{\boldsymbol{\alpha}}$. Recall from the proof of Proposition 13 that each coordinate of $\mathbf{F}$ satisfies $\Pi f'_a + \Pi' f_a = G^2 f_a^2$.

Let $R$ and $S$ be homogeneous polynomials of the same degree $\delta$, and set $U_R = R(\mathbf{F})$ and $U_S = S(\mathbf{F})$. Applying the chain rule and Euler's identity $\sum_a X_a \frac{\partial h}{\partial X_a} = \delta h$ yields

$$\Pi U'_R + \delta \Pi' U_R \equiv 0 \pmod{G^2}, \qquad \Pi U'_S + \delta \Pi' U_S \equiv 0 \pmod{G^2}.$$

Multiplying the first relation by $U_S$, the second by $U_R$, and adding them in characteristic two cancels the $\Pi'$ terms, giving $\Pi(U_R U'_S + U'_R U_S) \equiv 0 \pmod{G^2}$. Since $\gcd(\Pi, G) = 1$, it follows that

$$G^2 \mid (U_R U'_S + U'_R U_S). \tag{44}$$

This condition rules out adjacent vanishing orders. Indeed, if $U_R$ and $U_S$ satisfied (40), a direct calculation in characteristic two would give

$$U_R U'_S + U'_R U_S = c_R (a\alpha_i + b)(Z - \alpha_i)^{2\delta D - 2}.$$

For a non-degenerate transformation, the scalar $c_R(a\alpha_i + b)$ is nonzero. Since $G(\alpha_i) \neq 0$, the polynomial $(Z - \alpha_i)$ is coprime to $G$, so $G^2$ cannot divide this expression. Hence pairs with these adjacent vanishing orders do not exist for binary Goppa codes in any degree.

**Recovery via quadratic squares.** The obstruction is ruled out when the quotient $U_R/U_S$ is a perfect square, as its derivative is identically zero in characteristic two. To

produce such a quotient, we fix $\delta = 2$ and relax the vanishing order at $\alpha_i$ of $S$ to $2D-2$, while maintaining order $2D$ for $R$.

Since $\delta = 2$ is even, Euler's identity in characteristic two gives $\Pi U' \equiv 0 \pmod{G^2}$ for any quadratic form $h$ and $U = h(\mathbf{F})$, which implies

$$G^2 \mid U'. \tag{45}$$

Setting $x = Z - \alpha_i$, the degree bound $2D$ and the prescribed multiplicities at $\alpha_i$ force the two polynomials to take the form

$$U_R = c_R x^{2D}, \qquad U_S = x^{2D-2}(a + bx + cx^2), \qquad c_R a \neq 0.$$

Differentiating $U_S$ gives $U_S' = bx^{2D-2}$. Since $\gcd(G, x) = 1$, (45) forces $b = 0$. Both $U_R$ and $U_S$ are therefore squares, and taking the square root of their quotient yields the non-degenerate Möbius transformation

$$\mu(Z) := \sqrt{\frac{U_R(Z)}{U_S(Z)}} = \frac{\sqrt{c_R}(Z - \alpha_i)}{\sqrt{a} + \sqrt{c}\,(Z - \alpha_i)}. \tag{46}$$

To recover the quadrics, fix a public position $j \neq i$ and solve

$$\begin{aligned} R(\mathbf{V}(T)) &\equiv 0 \pmod{T^{2D}}, & R(\mathbf{y}_j) &= 1, \\ S(\mathbf{V}(T)) &\equiv T^{2D-2} \pmod{T^{2D-1}}, \end{aligned}$$

for homogeneous quadratic forms $R, S$. These normalizations impose no restriction: an admissible $R$ is nonzero at every public position other than $i$, and the leading jet coefficient of $S$ is nonzero, so both forms can be scaled independently.

The public evaluation ensures that $R(\mathbf{F})$ is nonzero and therefore has vanishing order exactly $2D$. The second congruence imposes order exactly $2D-2$ on $S(\mathbf{F})$. Both systems require only a jet of order $2D-1$.

For each $j \neq i$, homogeneity cancels the column multiplier in $R(\mathbf{y}_j)/S(\mathbf{y}_j)$, so

$$\beta_i = 0, \qquad \beta_j = \sqrt{\frac{R(\mathbf{y}_j)}{S(\mathbf{y}_j)}} = \mu(\alpha_j),$$

with $\beta_j = \infty$ when $S(\mathbf{y}_j) = 0$. Square roots in $\mathbb{K}$ are unique, and the non-degeneracy of $\mu$ ensures that the recovered support values are distinct. The remaining steps of Section 7.3 recover the multiplier and any support elements removed by shortening.

**A sufficient condition for direct recovery.** The construction above assumes that the two linear systems have solutions. The following dimension condition on the span of the products $f_a f_b$ suffices to guarantee solvability.

**Proposition 17.** *Suppose that $D \geq t$, and let $W = \operatorname{span}_{\mathbb{K}}\{f_a f_b : 1 \leq a \leq b \leq k\}$. If $\dim_{\mathbb{K}} W = 2D - t + 1$, then the quadratic forms required for direct support recovery exist*

*at every public position.*

*Proof.* By (45), $W$ is contained in

$$\mathcal{S} = \{U \in \mathbb{K}[Z] : \deg U \le 2D,\ G^2 \mid U'\}.$$

The kernel of differentiation on $\mathcal{S}$ is the space of square polynomials of degree at most $2D$, of dimension $D+1$. Since $\mathbb{K}$ is perfect, its image consists exactly of the polynomials $G^2A^2$ with $\deg A < D - t$, a space of dimension $D - t$ (zero when $D = t$). Thus $\dim_{\mathbb{K}} \mathcal{S} = 2D - t + 1$, and the dimension hypothesis gives $W = \mathcal{S}$. In particular, $W$ contains both $(Z-\alpha_i)^{2D}$ and $(Z-\alpha_i)^{2D-2}$, so there are homogeneous quadratic forms $R, S$ with $R(\mathbf{F}(Z)) = (Z-\alpha_i)^{2D}$ and $S(\mathbf{F}(Z)) = (Z-\alpha_i)^{2D-2}$. Their vanishing orders are exactly $2D$ and $2D-2$, respectively, so independent nonzero rescalings give the normalizations in the two systems above. □

**Remark 24.** *As in Remark 21, Proposition 17 applies to a shortened binary Goppa code, with $n$, $D$ and $\mathbf{F}$ taken for the shortened code.*

## 8. Experimental results

We implemented both key-recovery routes and ran them in two settings. In the end-to-end experiments the only input is public data, and recovery passes through every step: computing a hold-out kernel, obtaining a local jet, and then direct or minor-code recovery. In the synthetic experiments, we tested the finishing routes in isolation, supplying aligned tangent planes or local jets computed directly from the ground-truth curves. This included shortened Classic McEliece codes, for which hold-out-kernel computation was out of practical reach. Implementation and artifacts are available online.[1]

We experimentally validated the proposed attack by recovering full keys for three public McEliece key-recovery challenges published by the Technology Innovation Institute (TII) [45], all for binary Goppa codes. To our knowledge [9, 20, 40, 69], TII-173 and TII-249 had not previously been solved. TII-253 was also solved by Weis [80] using a different key-recovery attack based on the same hold-out distinguisher [37]. We also recovered keys for all 920 randomly generated codes from four families: binary Goppa codes, wild Goppa codes, alternant codes, and GRS subcodes.

At shortened Classic McEliece sizes, all minor codes in the synthetic experiments were full GRS codes, as Heuristic 3 predicts, and direct recovery succeeded on every code tested. At these sizes, every code tested also satisfied the condition of Proposition 17, which guarantees the forms that direct recovery needs for binary Goppa codes.

### 8.1. End-to-end key recovery

All end-to-end experiments followed the same pipeline. We chose a position $i$ outside the information set of the public generator matrix in systematic form. We then omitted column $i$ from the constraints used to compute the hold-out kernel (Section 3.4). For the

[1] https://github.com/kirill-vedenev/breaking-grs-subcodes

generated codes, we initially selected the first such position. We computed $M$ polynomials in the hold-out kernel $\mathcal{K}_i$ and recovered a local jet at $i$ following Section 4.

For direct recovery, we used a jet of order $2D-1$. In characteristic two, we also applied the minor-code attack: we first used bootstrapping (Section 5) with a jet of order $r = \max\{0, 2D-n+1\}$, the smallest nonnegative integer satisfying $n+r > 2D$ (Proposition 11), and then constructed the minor code and applied the Sidelnikov–Shestakov attack (Section 6). In odd characteristic, the minor-code method requires $2D-1 \le n-2$. None of the tested odd-characteristic parameter sets satisfied this condition, so we applied only direct recovery in those cases.

For binary Goppa codes, we chose the parameters for computing the hold-out kernels based on Assumption 1, allowing up to $t$ fewer imposed zeros than the standard requirement of $dD+1$. For wild Goppa codes, alternant codes, and GRS subcodes, we relied on the standard reconstruction condition (9), $s(n-1) > dD$.

**TII challenges.** The TII challenges consist of public keys for the Niederreiter cryptosystem based on binary Goppa codes with $5 \le m \le 10$. Each challenge is named by a nominal security level assigned by the organizers (corresponding to a rough key-recovery cost using the brute-force search and the support splitting algorithm). Table 3 lists the three challenges we solved.

We shortened the three challenge codes to the parameters in Table 3 (optimization (ii) in Section 3.4). When computing the hold-out kernels, we also imposed the order-$\prec s$ Hasse conditions at the additional column $\mathbf{w} = \sum_j \mathbf{y}_j$ (optimization (vi)).

Direct recovery, followed by reconstruction of the support elements removed by shortening (Section 6.3), yielded verified full keys for all three challenges. The minor-code method also recovered the shortened supports, which we extended to the full secret keys using the same reconstruction procedure. For each recovered key in GRS form, we applied a Möbius transformation sending the support value corresponding to $\mathbf{w}$ to $\infty$ and then interpolated the Goppa polynomial from the multipliers. Each resulting polynomial was monic, irreducible, and of degree $t$.

Table 3: Code parameters and hold-out-kernel settings used for full key recovery of the TII challenges. Here $[n_0, k_0]$ and $[n, k]$ are the original and shortened code parameters, respectively, and $D = n - 2t - 1$.

| Challenge | $(m, t)$ | $[n_0, k_0]$ | $[n, k]$ | $D$ | $(d, s)$ | $M$ |
|---|---|---|---|---|---|---|
| TII-173 | $(7, 8)$ | $[96, 40]$ | $[91, 35]$ | 74 | $(5, 4)$ | 256 |
| TII-249 | $(8, 16)$ | $[235, 107]$ | $[178, 50]$ | 145 | $(5, 4)$ | 280 |
| TII-253 | $(8, 9)$ | $[214, 142]$ | $[118, 46]$ | 99 | $(6, 5)$ | 280 |

For each challenge, we sampled $M$ linearly independent polynomials from the hold-out kernel and used the same set for jet recovery and both key-recovery methods.[2] The sample sizes were chosen with roughly 5–10-fold redundancy; smaller samples may also suffice. Computing $M$ linearly independent polynomials from the TII-173 hold-out kernel

[2]The computed sets of hold-out polynomials are available in the same repository.

by dense elimination took three minutes on two NVIDIA H200 GPUs. For TII-249 and TII-253, we instead used block Wiedemann [21], with code adapted from Saarinen's GPU implementation of it [70]. Each GPU stage used one H200; the CPU linear-generator stage used 32 and 48 threads, respectively. Computing the polynomials took about 1 hour for TII-249 and 14 hours for TII-253. After computing the hold-out polynomials, the remaining steps took less than two minutes per challenge on a standard laptop.

**Randomly generated codes.** In addition to the public TII challenges, we tested 920 randomly generated codes from the four families: toy-size binary Goppa codes, wild Goppa codes, alternant codes, and high-codimension GRS subcodes. Their parameters are listed in Table 4.

For each toy-size binary Goppa code $\Gamma(\boldsymbol{\alpha}, G)$, we used all elements of $\mathbb{F}_{64}$ in random order as the support. For each wild Goppa code $\Gamma(\boldsymbol{\alpha}, g^3)$ over $\mathbb{F}_4$, we sampled $n$ distinct elements of $\mathbb{F}_{64}$ as the support. We chose $G$ and $g$ as random monic irreducible polynomials of degree $t$ over $\mathbb{F}_{64}$. For the GRS subcodes and alternant codes, we chose a random support $\boldsymbol{\alpha}$ and a random multiplier $\boldsymbol{\lambda}$. For the GRS subcodes, we then took a random $k$-dimensional subcode of $\mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$; for the alternant codes, we took $\mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) \cap \mathbb{F}_q^n$.

For each GRS subcode, we also checked the dimensions of the Schur squares of the code, its dual, and sampled ordinary shortenings of both. We considered all shortening dimensions, using eight random information sets per code. All tested squares had the dimensions expected for random codes; the standard square-code and filtration attacks [26, 28, 81] therefore do not apply directly.

Table 4: Code parameters and hold-out-kernel settings for end-to-end key recovery of randomly generated codes. Here $t$ is the degree of $G$ or $g$; "all" in the $M$ column means that a basis of the full hold-out kernel was used, with its dimension given in parentheses.

| Family | $q$ | $m$ | $t$ | $[n, k]$ | $D$ | $(d, s)$ | $M$ | Codes |
|---|---|---|---|---|---|---|---|---|
| *Characteristic two: direct and minor-code recovery* | | | | | | | | |
| Binary Goppa | 2 | 6 | 6 | $[64, 28]$ | 51 | $(5, 4)$ | all (372–706) | 100 |
| Wild Goppa | 4 | 3 | 3 | $[55, 28]$ | 42 | $(5, 4)$ | 256 | 20 |
| Wild Goppa | 4 | 3 | 4 | $[63, 27]$ | 46 | $(4, 3)$ | 256 | 100 |
| Wild Goppa | 4 | 3 | 4 | $[64, 28]$ | 47 | $(4, 3)$ | 256 | 100 |
| Alternant | 8 | 2 | | $[60, 20]$ | 39 | $(3, 2)$ | 48 | 100 |
| GRS subcode | 256 | 1 | | $[48, 16]$ | 27 | $(3, 2)$ | 48 | 100 |
| GRS subcode | 256 | 1 | | $[72, 20]$ | 42 | $(3, 2)$ | 48 | 100 |
| *Odd characteristic: direct recovery only* | | | | | | | | |
| Alternant | 11 | 2 | | $[45, 15]$ | 29 | $(3, 2)$ | all (21) | 100 |
| GRS subcode | 257 | 1 | | $[48, 16]$ | 27 | $(3, 2)$ | 48 | 100 |
| GRS subcode | 257 | 1 | | $[72, 20]$ | 42 | $(3, 2)$ | 48 | 100 |

Using the initially chosen held-out position and the standard procedure, both methods recovered keys for all 220 wild Goppa codes and for 99 of the 100 binary Goppa codes. The remaining binary Goppa code required an adjustment, but key recovery succeeded in

this case as well. Each recovered wild Goppa key contains a monic irreducible polynomial of degree $t$ whose third and fourth powers both reproduce the public code. For all 600 GRS subcodes and alternant codes, the recovered support matched the original up to a Möbius transformation and, for alternant codes, Frobenius conjugation. The remaining binary Goppa code is the exception discussed below.

*Rank conditions.* The methods of Sections 4 to 6 rely on rank conditions that we do not prove in general. We checked these conditions for the TII challenges and the 920 generated codes. In every reported attempt, $\operatorname{rank} \mathbf{J} = k - m - 1$. For wild Goppa codes, alternant codes, and GRS subcodes, the restrictions to $\ker \mathbf{J}$ of the quadratic forms $Q_h$ with $h \in \mathcal{H}^0$ spanned a space of dimension $\binom{m}{2}$. Thus the first-order system separated the $m$ branches (Proposition 4), and $\operatorname{rank} \mathbf{A} = k - 2$ on each branch (Corollary 1). For binary Goppa codes, the same held with $\mathcal{H}$ and $\mathbf{A}_\Gamma$ in place of $\mathcal{H}^0$ and $\mathbf{A}$ (Corollary 2) in every reported attempt except the one described below.

In the characteristic-two experiments, the bootstrapped quadratic forms gave $\operatorname{rank} \mathbf{J}_j = k - 2$ at every public column (Heuristic 2). Every minor code was a full GRS code (Heuristic 3): its dimension reached the upper bound $D - t$ of Proposition 13 for binary Goppa codes and $D$ of Proposition 12 otherwise.

The exception occurred at the first held-out position of the remaining binary Goppa code. There, the restricted quadratic forms spanned a space of dimension 9 instead of $\binom{6}{2} = 15$. For each $\sigma = 0, \dots, 5$, mixed vectors in $\operatorname{span}\{\mathbf{p}, \varphi^\sigma(\boldsymbol{\tau}_1), \varphi^{\sigma+1}(\boldsymbol{\tau}_1)\}$ also satisfied the first-order system, so the system did not separate the branches. Adding the order-three equations to the first-order system (Remark 10) separated all six branches using the same hold-out kernel. Alternatively, when we chose another held-out position and recomputed the hold-out kernel, all rank conditions held and both key-recovery methods recovered the full key.

**Degree seven.** The cost estimates of Section 9 for Classic McEliece use hold-out polynomials of degree seven, higher than in the experiments above. Since the hold-out computation is out of practical reach at these sizes, we tested this degree on four random toy-size binary Goppa codes with $m = 6$ and $t = 4$, shortened from $[64, 40]$ to $[54, 30]$. As in Section 9, twelve retained points had multiplicity five and the others multiplicity six. This imposed exactly $t$ fewer zeros than $dD + 1$, the largest reduction that Assumption 1 allows. We had fixed this choice on a pilot key, where fewer points of multiplicity five made the rank conditions fail. For each code, we computed $M = 256$ polynomials in the hold-out kernel. This took about 43 minutes per code on one H200 GPU. The rank conditions held on all four codes, and both methods recovered their full keys.

The four shortened codes had 373, 360, 370 and 350 dual codewords of weight at most $2d - 2 = 12$, respectively. On a similar code, Weis [80, Sec. 7] reported a failure of his single-run method. This failure required a fallback to his multi-run method or to less aggressive shortening. Our methods showed no such obstruction on these four codes.

### 8.2. Synthetic experiments

The hold-out computation is out of practical reach for large codes (including Classic McEliece codes with necessary shortenings applied as in Section 3.5). We therefore tested the next steps in isolation (i.e., minor-code recovery and direct recovery), supplying aligned tangent planes and local jets computed from the ground-truth curve. The jets were taken at random public positions and reparametrized at random.

Table 5 lists the tested codes. We chose the supports, multipliers, Goppa polynomials, and GRS subcodes at random. For Classic McEliece, we used the binary Goppa parameters $(m, n, t)$ in Table 2 and shortened the codes to the listed dimensions $k_\ell$. Both methods were tested on codes in characteristic two; direct recovery was also tested in odd characteristic. The minor-code rows of Table 5 cover 20 codes per Classic McEliece shortening in Table 2 and 100 codes per parameter set for the other families.

Table 5: Synthetic tests of minor-code and direct recovery. The last column counts GRS minor codes obtained without puncturing in the minor-code rows and recovered keys in the direct-recovery rows. Here $n$ is the length after shortening.

| Method | Family | $q$ | $n$ | Codes | Successes |
|---|---|---|---|---|---|
| Minor code | Binary Goppa | 2 | 16–220 | 1,100 | 1,100 |
| | Classic McEliece, shortened | 2 | 922–1,979 | 200 | 200 |
| | Wild Goppa | 4, 8, 16 | 55–256 | 1,500 | 1,500 |
| | Alternant | 2, 4, 8, 16 | 14–220 | 1,100 | 1,099 |
| | GRS subcode | $2^5$–$2^9$ | 20–300 | 700 | 700 |
| Direct | Binary Goppa | 2 | 100, 400 | 120 | 120 |
| | Classic McEliece, shortened | 2 | 922, 1,888 | 40 | 40 |
| | Wild Goppa | 3, 4, 5 | 100–800 | 360 | 360 |
| | Alternant | 2, 3, 4, 5 | 100–1,000 | 540 | 540 |

All minor codes but one were full GRS codes with support $\boldsymbol{\alpha}$, as Heuristic 3 predicts, and the Sidelnikov–Shestakov attack recovered the support from each of them. The exception was a binary $[60, 12]$ alternant code with a one-dimensional tangent space at one position, which violates the hypothesis of Heuristic 3. After we punctured this position, its minor code was a $[59, 50]$ GRS code, which gave the support at the other 59 positions. We restored the missing support element by testing the remaining candidates against the public code. Direct recovery succeeded on all codes. We also tested the condition $\dim_{\mathbb{K}} W = 2D - t + 1$ of Proposition 17 on binary Goppa codes using $W$ computed from the ground-truth curve: it held for all 10,000 shortened Classic McEliece codes (1,000 for each parameter set and each value of $k_\ell$ in Table 2). For 20 of these codes per parameter set and value of $k_\ell$, we also shortened the code at 20 random positions. The condition held for all 4,000 shortened codes, so $\operatorname{rank} \mathbf{J}_j = k - 2$ at these positions by Proposition 15.

Finally, we tested the variant of Section 5.4 in which only the Frobenius closures $E_i$ are known. We computed the closures from the ground-truth curve at a random set $S$ of positions, with $|S| = 2D + 1 - n$ for binary alternant codes. For binary Goppa codes, we chose $|S| = n - 5t + 1$ based on Assumption 1, allowing up to $t$ fewer roots than

the bound $n + |S| > 2D$ requires. We recovered a local jet from the resulting quadratic forms as in Section 4 and then applied bootstrapping and minor-code recovery. The support was recovered for all 500 binary Goppa codes with $64 \le n \le 256$, all 40 shortened Classic McEliece codes with $n = 922$ and $n = 1{,}888$, and all 300 alternant codes with $64 \le n \le 240$. With $|S| = n - 5t - 1$ (equality in the bound of Assumption 1), recovery failed for 5 of the 540 binary Goppa codes.

## 9. Complexity

The cost of either key-recovery method is the cost of obtaining hold-out polynomials at one position, followed by jet recovery and the finishing steps. We specify the required number and distribution of polynomials independently of the solver. For the concrete sampling estimates, we use a block Wiedemann sampler, which is analysed in a note in the repository.[3] At these parameters, sampling dominates the remaining work. The estimates assume the conditions listed at the end of this section.

We choose the shortening and the multiplicities using the mixed version of the kernel-dimension heuristic in Heuristic 1, as in [80, Sec. 6]. For each shortening, we impose only the total multiplicity required by the vanishing criterion with the $t$ additional Goppa conditions of (15). This allows some retained points to have multiplicity five instead of six.

Throughout this section, $n, k, D$ are the parameters after shortening, with $n = k + mt$ and $D = n - 2t - 1$. We use homogeneous multilinear polynomials of degree seven and hold out one non-unit column $i$. We include the public point $\mathbf{w}$ at infinity at multiplicity six. There are $P = mt$ retained non-unit constraint points, counting $\mathbf{w}$. Choose $P_5$ finite retained points at multiplicity five; the remaining $P_6 = P - P_5$ points have multiplicity six. The $k$ unit columns already have multiplicity at least six for every polynomial in $\mathcal{P}^{\mathsf{red}}_{=7}$. The total imposed multiplicity is therefore $6(k + P) - P_5$.

We take the largest feasible $P_5$ for which

$$6(k + P) - P_5 \ge 7D + 1 - t.$$

At the parameters considered here, equality is feasible and gives

$$P_5 = (15 - m)t - k + 6, \qquad 0 \le P_5 \le P - 1. \tag{47}$$

For each parameter set, choose the smallest $k$ for which the mixed kernel-dimension prediction

$$[z^7]\big[(1 + z)^k (1 - z^2)^{P_6} (1 - z^3)^{P_5}\big]_+ \tag{48}$$

is positive. The subscript means truncation at the first nonpositive coefficient. Write $\mathcal{K} = \ker \mathbf{K}$ for this mixed hold-out kernel and put $N = \binom{k}{7}$. The chosen parameters appear in Table 6.

[3] *Sampling binary kernels with block Wiedemann*, https://github.com/kirill-vedenev/breaking-grs-subcodes.

Key recovery uses only the polynomials of $\mathcal{K}$ that vanish at the held-out column $\mathbf{p} = \mathbf{y}_i$,

$$\mathcal{K}' = \{h \in \mathcal{K} : h(\mathbf{p}) = 0\}.$$

Every polynomial that vanishes on the curve vanishes at $\mathbf{p}$, so this condition excludes no polynomial that satisfies (16). It adds one zero at $\alpha_i$. For $h \in \mathcal{K}'$, the polynomial $h(\mathbf{F}(Z))$ therefore has $7D + 2 - t$ imposed zeros, one more than the minimum $7D + 1 - t$ of Assumption 1. With one zero in reserve, [80, Thm. 3], which allows a different order at each point, applies: for a fixed support and fixed orders, (16) holds for every $h \in \mathcal{K}'$ with probability at least $1 - n\,2^{-2m}$ over a uniform monic irreducible Goppa polynomial of degree $t$. At the five parameter sets, $n\,2^{-2m} < 2^{-14}$. The orders are six at all retained points except the $P_5$ points of multiplicity five and the held-out column. They are therefore fixed by choosing these positions and the shortened positions in advance.

### 9.1. The number of polynomials

Jet recovery of Section 4 does not need the whole basis of $\mathcal{K}'$. Its rank conditions depend on the polynomials only through their gradients and their quadratic parts at the held-out column $\mathbf{p}$. It therefore suffices to sample enough polynomials to preserve the images of $\mathcal{K}'$ under these two linear maps.

The maps are $h \mapsto \nabla h(\mathbf{p})$ and $h \mapsto Q_h|_{E_i}$. We write the latter in a basis of $E_i$ consisting of vectors in $\mathbb{F}_2^k$. Both maps are $\mathbb{F}_2$-linear and take binary values, so their ranks over $\mathbb{F}_2$ equal the dimensions over $\mathbb{K}$ of the spans of their images. Suppose that $\mathcal{K}'$ is in the maximal-rank case and satisfies the rank condition of Corollary 2. These ranks are then $k - m - 1$ and $\binom{m}{2}$. Samples that span both images satisfy the hypotheses of Proposition 4 and Corollary 2.

**Lemma 8.** *Let $\pi$ be an $\mathbb{F}_2$-linear map on $\mathcal{K}'$ of rank $a$, and let $h_1, \dots, h_M$ be independent uniform elements of $\mathcal{K}'$. Then $\pi(h_1), \dots, \pi(h_M)$ span $\pi(\mathcal{K}')$ with probability greater than $1 - 2^{a-M}$.*

*Proof.* The vectors $\pi(h_1), \dots, \pi(h_M)$ are independent and uniform in $\pi(\mathcal{K}')$, a space of dimension $a$. They fail to span it if and only if some nonzero linear functional on $\pi(\mathcal{K}')$ vanishes at all of them, which has probability $2^{-M}$ for each of the $2^a - 1$ functionals. By the union bound, they fail to span it with probability at most $(2^a - 1)2^{-M} < 2^{a-M}$. □

The sampler of Section 9.2 returns independent uniform elements of $\mathcal{K}$. Since $d = 7$ is odd, Euler's identity gives $h(\mathbf{p}) = \nabla h(\mathbf{p})\,\mathbf{p}$, so the values at $\mathbf{p}$ are read off the gradients. If some sample takes the value 1 at $\mathbf{p}$, we add one such sample to every other sample with value 1 and discard it. The samples with value 1 are independent uniform elements of $\mathcal{K} \setminus \mathcal{K}'$, so these sums are independent uniform elements of $\mathcal{K}'$. At least $M - 1$ independent uniform elements of $\mathcal{K}'$ remain.

We choose

$$M = \max\left\{k - m - 1, \binom{m}{2}\right\} + 32.$$

By Lemma 8 applied to both maps and to $M-1$ samples, both images are then spanned with probability greater than $1-2^{-30}$. At the parameters considered here, $k-m-1>\binom{m}{2}$, so $M=k-m+31$. The values of $M$ are given in Table 6. The polynomials need not be linearly independent. Jet recovery computes both ranks, so a failure is detected.

### 9.2. Computing the polynomials

The matrix $\mathbf{K}$ can be applied to vectors using the public columns, without storing its entries. The multiplicity-six points contribute derivative orders five and three; the multiplicity-five points contribute orders four and three. Hence

$$J_{\mathcal{K}}=P_6\binom{k}{5}+P_5\binom{k}{4}+P\binom{k}{3}$$

is the number of rows of $\mathbf{K}$, and its unknown vectors have length $N=\binom{k}{7}$.

The sampling task is to obtain $M$ independent uniform elements of $\mathcal{K}$. The returned vectors are checked against $\mathbf{K}$, and jet recovery checks the ranks of their gradients and quadratic parts as in Section 9.1. Block Lanczos and block Wiedemann are both possible solvers. The randomized block Lanczos construction of [80, Prop. 8] gives uniform samples under its checked output-rank condition. Its iteration count and supply of enough samples require a solver-success assumption of the type in [80, Assumption 4], applied to our coefficient kernel and sample requirement.

The block Wiedemann sampler of the note returns $M$ independent uniform elements of $\mathcal{K}$ conditional on acceptance, with a proved rejection bound for every fixed binary matrix. Table 6 gives its costs on the mixed hold-out systems. Independent retries do not change the displayed exponents. The solver guarantee does not require an assumption on the geometric ranks of $\mathcal{K}$; key recovery still requires the conditions listed below.

The first cost column counts the AND and XOR gates of the sampler. The second also charges each gate and each transferred bit the square root of the size of the memory region it uses, as in [80, Sec. 6]. The sampler stores about 104 to 2,323 TiB of coefficients and arithmetic workspace, depending on the parameter set.

Table 6: Cost of sampling $M$ polynomials from the mixed hold-out kernel with block Wiedemann. Here $k$ is the dimension after shortening and $P_5$ is the number of retained points of multiplicity five.

| | | | | $\log_2$ of the cost | |
|---|---|---|---|---|---|
| Parameter set | $k$ | $P_5$ | $M$ | Gates | With memory access |
| `mceliece348864` | 150 | 48 | 169 | 92.68 | 109.21 |
| `mceliece460896` | 195 | 3 | 213 | 98.19 | 115.21 |
| `mceliece6688128` | 221 | 41 | 239 | 101.21 | 119.06 |
| `mceliece6960119` | 214 | 30 | 232 | 100.35 | 117.68 |
| `mceliece8192128` | 221 | 41 | 239 | 101.21 | 119.06 |

### 9.3. Cost after sampling

Extending the local jet dominates the cost after sampling. We bound this cost using schoolbook field arithmetic and Gaussian elimination over $\mathbb{K} = \mathbb{F}_{2^m}$. We charge $3m^2$ bit operations for a multiplication, $m$ for an addition, and $6m^3$ for an inversion in $\mathbb{K}$.

Accumulating the gradients and quadratic parts of the $M$ polynomials at $\mathbf{p}$ takes $MN(1 + d + \binom{d}{2})$ additions over $\mathbb{F}_2$. The eigenvalue computation of Proposition 5 takes at most $2^m m^4 + m^6$ operations in $\mathbb{K}$.

Direct recovery needs a jet of order $2D - 1$ (Section 7.4). The minor-code method needs order $2D - n + 1$ for recovering the quadratic equations of the curve (Proposition 11). The continuation of Proposition 10 computes only the odd coefficients, since each even coefficient is the square of an earlier one.

At an odd order $r$, the right-hand sides of (22) and (25) use the coefficients of $T^r$ and $T^{r+1}$ in the substituted polynomials. For each of the $N$ monomials, we multiply the corresponding $d$ coordinate series to order $r + 1$. The two resulting coefficients serve all $M$ polynomials. Recomputing these products at each odd order avoids storing the series of all $N$ monomials. For direct recovery, they take at most

$$(d-1)N \sum_{\substack{3 \le r \le 2D-1 \\ r \text{ odd}}} \frac{(r+2)(r+3)}{2}$$

multiplications in $\mathbb{K}$ and as many additions. For the minor-code method, the sum stops at $2D - n + 1$. Combining the monomial coefficients takes another $2MN$ additions per order. All orders share the $2M \times k$ matrix $\mathbf{A}_\Gamma$, which is eliminated once.

The systems for the quadratic forms in direct recovery and for the quadratic equations of the curve have $\binom{k+1}{2}$ unknowns and at most $2D + 1$ rows. We charge Gaussian elimination for these systems and for the multiplier and completion systems of Section 6.3. For the minor-code method, we also charge the tangent-plane computations at all public columns, the construction of the minor code, and the Sidelnikov–Shestakov attack. These costs are additional to the sampling costs in Table 6.

For a common upper bound on these smaller computations, let $n_0$ be the length before shortening. At the five parameter sets, we charge $128n_0^4$ field operations for forming and reducing the quadratic systems, solving the jet equations, branch separation, computing tangent planes and the minor code, recovering the support, and completing and checking the key. This includes repeating the multiplier and lifting steps after a change of affine frame. We charge each of these operations at the inversion cost $6m^3$. Adding the monomial-series products, the combinations at each odd order, and the initial derivative accumulation gives Table 7. Memory access is not charged.

### 9.4. Conditions

In addition to the conditions used to select the distinguisher parameters (see Section 3.4), the cost estimates for key recovery rely on the following assumptions.

(i) *Ranks for jet recovery.* We assume $\operatorname{rank} \mathbf{J} = k - m - 1$ and that the restrictions $Q_h|_{E_i}$

Table 7: Upper bounds on the number of bit operations after sampling, for the parameters of Table 6. Entries are base-two logarithms.

| Parameter set | Direct recovery | Minor-code recovery |
|---|---|---|
| `mceliece348864` | 77.78 | 74.04 |
| `mceliece460896` | 82.69 | 78.98 |
| `mceliece6688128` | 85.12 | 81.38 |
| `mceliece6960119` | 84.49 | 80.77 |
| `mceliece8192128` | 85.12 | 81.38 |

for $h \in \mathcal{K}'$ span a space of dimension $\binom{m}{2}$. These conditions give branch separation and continuation by Proposition 4 and corollary 2. The sampling guarantee in Section 9.1 assumes these ranks for the full kernel $\mathcal{K}'$.

*Status:* These conditions have not been proved at Classic McEliece sizes, and the cost of computing the hold-out kernel makes testing them at those sizes impractical. However, experiments on the TII challenges and randomly generated codes, including the toy degree-seven examples, provide evidence that the conditions are likely to hold (cf. Section 8.1).

(ii.a) *Sufficient conditions for direct support recovery.* The equality $\dim_{\mathbb{K}} W = 2D - t + 1$ guarantees the quadratic forms required for direct support recovery (see Proposition 17).

*Status:* We have not proved that this condition holds for every code in the family. However, it can be tested at Classic McEliece sizes when the curve is known. It held for all 10000 shortened Classic McEliece codes in Section 8.2, providing experimental evidence that it holds with high probability at these parameters. Whenever it holds, successful recovery of a jet of order $2D - 1$ suffices for direct support recovery.

(ii.b) *Additional condition for the minor-code attack.* For the minor-code attack, we assume both the dimension condition in (ii.a) and the rank condition $\operatorname{rank} \mathbf{J}_j = k - 2$ at every public column. Here $\mathbf{J}_j$ is the gradient matrix of the bootstrapped quadrics at $\mathbf{y}_j$. By Proposition 15, this additional condition is equivalent to $\dim_{\mathbb{K}} \widetilde{W} = 2(D-1) - t + 1$ for each one-point shortening. These ranks allow the tangent planes to be recovered. Condition (ii.a) guarantees that the resulting minor code is full (see Proposition 14).

*Status:* The additional condition was tested at shortened Classic McEliece sizes and held at all 4,000 sampled positions. All 200 synthetic minor-code tests at these sizes also succeeded (see Section 8.2).

(iii) *Multiplier uniqueness.* To recover the coordinates removed by shortening, we assume that the shortened multiplier system has a one-dimensional solution space (see Section 6.3).

*Status:* The solution space was one-dimensional in all experiments where this was tested, including synthetic runs at shortened Classic McEliece sizes (Section 8).

## 10. Comparison with other attacks

Since the introduction of the hold-out distinguisher in the first version of [37], several concurrent works have developed and analyzed key-recovery attacks built on it: a later revision of [37], Saarinen [68, 70], Weis [80], and this paper together with its earlier working drafts.

These key-recovery attacks fall into two categories according to how they use the linear system of Hasse-derivative conditions $(\partial^{[\mathbf{a}]}h)(\mathbf{y}_j) = 0$ from Section 3:

- *Attacks from vanishing polynomials* compute polynomials $h$ in a hold-out kernel and differentiate the identity $h(\mathbf{F}(Z)) \equiv 0$ at held-out positions to recover local subspaces or local jets of the hidden curve.
- *Attacks from interpolation relations* compute linear dependencies among the Hasse-derivative conditions and read cross-ratios of the support from univariate Hermite interpolation coefficients.

The two main methods of this paper belong to the first category and need a hold-out kernel at one position. Our earlier methods (see Appendix B) also belong to the first category and require hold-out kernels at several positions. The gradient attack of Weis [80] is another attack from vanishing polynomials. The key-recovery attack of [37], Saarinen's variants [68, 70], and the single-run attack of Weis [80] fall into the second category. Below, we compare these routes in detail.

### 10.1. Attacks from vanishing polynomials

All attacks in this category start by collecting polynomials $h$ from a hold-out kernel, which satisfy $h(\mathbf{F}(Z)) \equiv 0$ (Section 4). They differ mostly in what information is recovered from these polynomials and in how many held-out positions are needed. Below, we sort them by the amount of information recovered from the polynomials at a held-out position $i$ (see also Figure 7).

- *Weis's gradient attack* [80, Sec. 4 and App. B] uses the hold-out polynomials only to obtain the kernel of $\mathbf{J}$, which in our notation is the Frobenius closure of the tangent plane (under the maximal rank assumption):

  $$E_i = \operatorname{span}\{\mathbf{y}_i, \mathbf{F}'(\alpha_i), \varphi(\mathbf{F}'(\alpha_i)), \dots, \varphi^{m-1}(\mathbf{F}'(\alpha_i))\}.$$

  It is computed at every column, which takes 103 to 1,443 distinguisher runs at the Classic McEliece parameters [80, Table 3]. For the remaining work, it employs an intricate multi-step procedure [80, App. B], which is tailored to binary Goppa codes only and recovers the support of the shortened code.
- *Our earlier attack based on extended minor codes* (Section B.2) used the hold-out polynomials, through the local quadratic equations of Section 4.4, to recover a tangent plane at each public column, which requires holding out every column.

These planes could lie on different, unknown branches:

$$W_i^{[1]} = \varphi^{\sigma_i}(V_i^{[1]}) \quad \text{for } 1 \le i \le n.$$

Here $\sigma_i$ is unknown and may depend on $i$. The minor-code method of Section 6, however, requires tangent planes on a common branch.

To handle these unknown branch choices, we introduced the extended minor code, which includes all $m$ conjugates under $\varphi$. Let $\widetilde{\mathbf{T}}$ be the matrix of Definition 5 computed from the planes $W_i^{[1]}$, and put

$$\mathbf{T}_{\text{ext}} = [\widetilde{\mathbf{T}} \mid \varphi(\widetilde{\mathbf{T}}) \mid \cdots \mid \varphi^{m-1}(\widetilde{\mathbf{T}})].$$

The extended minor code is the row space of $\mathbf{T}_{\text{ext}}$. For each public position $i$, the corresponding $m$ columns, one from each block, represent all $m$ branches; the unknown $\sigma_i$ determines only a cyclic shift in their order. Thus, up to an unknown column permutation that groups columns by branch, the extended minor code is contained in the direct sum of the minor codes of the $m$ branches.

For the generic alternant codes we tested, this containment was an equality, and row reduction separated the branches. For binary Goppa codes, the extended minor code was a proper subcode of this direct sum, and we separated the branches using the Schur square of its dual. Once the minor code of one branch is isolated, the Sidelnikov–Shestakov attack recovers the support.

We also gave a variant of this attack for binary Goppa codes that applies the minor construction twice. This variant requires a local jet of order 3 at every public column.

- *Our earlier partial-support guessing attack* (Section B.1) used the hold-out polynomials to recover local jets of high order at a few held-out positions. A recovered jet consists of the first coefficients of a local representative $u_i(T)\mathbf{P}_\sigma(\psi_i(T))$, where the series $u_i$ and $\psi_i$ are unknown and differ from position to position. The attack therefore used the jets only through their derivative flags (Section A.1), the nested spaces

$$V_i^{[r]} = \operatorname{span}\{(\partial^{[e]}\mathbf{F})(\alpha_i) : 0 \le e \le r\} \quad \text{for } 0 \le r \le r_i,$$

which are spanned by the coefficients of the jet and do not depend on $u_i$ and $\psi_i$.

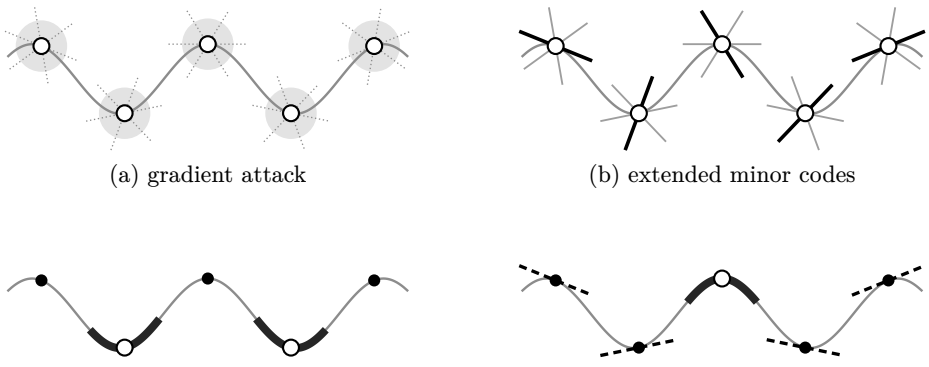
The flags give equations for the curve. The hidden polynomial vector satisfies

$$(\partial^{[e]}\mathbf{F})(\alpha_i) \in V_i^{[e]} \quad \text{for } 0 \le e \le r_i,$$

and if $\alpha_i$ is known, these are homogeneous linear equations in the coefficients of $\mathbf{F}$.

The support values $\alpha_i$ at the held-out positions are not known, so the attack guessed them. In the subfield case, it also guessed the branch of each recovered flag. A guess was rejected if the resulting linear system had only the zero solution. Each guessed value ranges over $\mathbb{K}$, so the cost of guessing grows exponentially with the number of positions used, which made this attack expensive.

- *The two attacks of this paper* use the hold-out polynomials to recover a deep local jet of one branch at a single held-out position (Section 4), and they use the coefficients of this jet themselves, not only the spaces they span. As a result, a single hold-out kernel suffices: no other column is held out and no support value is guessed. There is no alignment problem either, because bootstrapping derives the tangent planes at all columns from the jet of one branch (Section 5), and direct recovery works at one position only (Section 7). The cost is essentially that of computing the polynomials of one hold-out kernel (Table 6), which is the lowest among the attacks from vanishing polynomials.



(a) gradient attack (b) extended minor codes

(c) partial-support guessing (d) this paper

Figure 7: Local data used by the attacks from vanishing polynomials, for $m = 3$. Hollow circles mark held-out positions. (a) The Frobenius closure $E_i$ (shaded) at every position; the tangent planes inside it (dotted) are not known individually. (b) The tangent planes of all $m$ branches at every position: the recovered plane (black) and its conjugates (gray). Which plane belongs to which branch is unknown. (c) Derivative flags (thick arcs) at a few positions. (d) A local jet (thick arc) at one position, and the tangent planes of the same branch (dashed) that bootstrapping derives from it.

**Remark 25.** *Our methods can also start from the input of the gradient attack. Given Frobenius closures $E_i$ at positions $S$ with $n + |S| > 2D$, the Frobenius-closure variant of Section 5.4 computes the quadratic equations of the curve over $\mathbb{F}_q$. Under the rank conditions of Section 4, these give a local jet. Both methods then proceed as before. This is not specific to binary Goppa codes and does not need closures at every column.*

*In the synthetic experiments of Section 8.2, with closures computed from the ground-truth curve, bootstrapping and the minor code recovered the support of all alternant codes tested with $|S| = 2D + 1 - n$ and all binary Goppa codes tested with $|S| = n - 5t + 1$.*

### 10.2. Attacks from interpolation relations

The attacks in this category solve a linear system of a different kind, in which no column is held out. Consider the linear forms $h \mapsto (\partial^{[\mathbf{a}]}h)(\mathbf{y}_j)$ on $\mathcal{P}$ at all $n$ columns, where

$1 \le j \le n$ and $|\mathbf{a}| < s_j$ for some multiplicities $s_1, \dots, s_n$. The unknowns of the system are scalars $\zeta_{j,\mathbf{a}} \in \mathbb{K}$, one for each of these forms, and its equations state that

$$\sum_{j=1}^{n} \sum_{|\mathbf{a}|<s_j} \zeta_{j,\mathbf{a}} \, (\partial^{[\mathbf{a}]} h)(\mathbf{y}_j) = 0 \quad \text{for all } h \in \mathcal{P}. \tag{49}$$

We call the solutions $\boldsymbol{\zeta} = (\zeta_{j,\mathbf{a}})$ *relations*; they are the linear dependencies among the forms. In the system of a hold-out kernel (10), by contrast, the unknowns are the coefficients of $h$, and the equations are $(\partial^{[\mathbf{a}]} h)(\mathbf{y}_j) = 0$ at the columns that are not held out.

For $d = 1$ and $s_j = 1$, the relations are the codewords of the dual code, and the Sidelnikov–Shestakov attack (Section 2.2) uses those given by the systematic generator matrix of a GRS code. In the notation of (1), the columns $\mathbf{g}_1, \dots, \mathbf{g}_n$ of a generator matrix satisfy

$$\mathbf{g}_i = \sum_{j=1}^{k} m_{j,i} \, \mathbf{g}_j \quad \text{for } k < i \le n, \qquad m_{j,i} = a_j \, b_i \, (\alpha_i - \alpha_j)^{-1}, \tag{50}$$

where $b_i = \beta_i \prod_{j'=1}^{k} (\alpha_i - \alpha_{j'})$. The factors $a_j$ and $b_i$ cancel in the ratios (2), which are cross-ratios of the support. For the public code, these relations, with $D+1$ in place of $k$ and $\boldsymbol{\lambda}$ in place of $\boldsymbol{\beta}$, are still codewords of the dual code, but the dual code of a subcode is larger than their span, and they cannot be singled out in it.

For larger $d$, a relation of the same kind comes from Hermite interpolation of the polynomial $\phi(Z) = h(\mathbf{F}(Z))$, which has degree at most $dD$. We have $h(\mathbf{y}_j) = \lambda_j^d \phi(\alpha_j)$, and by the chain rule every derivative $(\partial^{[e]}\phi)(\alpha_j)$ with $e < s_j$ is a linear combination of the forms of column $j$. If $\sum_{j \ne i} s_j > dD$, then $\phi(\alpha_i)$ is determined by the derivatives of $\phi$ of order less than $s_j$ at the points $\alpha_j$ with $j \ne i$ (Lemma 1), which gives the relation

$$h(\mathbf{y}_i) = \sum_{j \ne i} \sum_{|\mathbf{a}|<s_j} \zeta_{j,\mathbf{a}}^{(i)} \, (\partial^{[\mathbf{a}]} h)(\mathbf{y}_j) \quad \text{for all } h \in \mathcal{P}. \tag{51}$$

We call it the *interpolation relation* of column $i$. Its right-hand side vanishes on the hold-out kernel of $i$, so $h(\mathbf{y}_i) = 0$ there, as in Proposition 3, and the distinguisher uses nothing else of it.

The attacks of this category use its coefficients. Suppose that $\sum_{j \ne i} s_j = dD + 1$, and let $s$ be the multiplicity of column $i$, which is then the same for all columns $i$ with this property. Then, by the Hermite interpolation formula,

$$\zeta_{j,\mathbf{a}}^{(i)} = \xi_i \, \nu_{j,\mathbf{a}} \, (\alpha_i - \alpha_j)^{s-1} \quad \text{for } |\mathbf{a}| = s_j - 1, \tag{52}$$

where $\xi_i$ depends only on $i$, and $\nu_{j,\mathbf{a}}$ only on $j$ and $\mathbf{a}$ [37, Sec. 7.1], [80, Thm. 5]. For binary Goppa codes, the same holds with $dD + 1 - t$ in place of $dD + 1$, provided the interpolation map on the space of compositions $h(\mathbf{F})$ is nonsingular. This is the form of the entries $m_{j,i}$ in (50), with the exponent $s - 1$ in place of $-1$.

Both [37] and [80] take characteristic two and $s - 1$ a power of two; for Classic McEliece, [80] takes $s = 5$ at four columns $i$ and multiplicities five and six at the other columns.

Then $(\alpha_i - \alpha_j)^{s-1} = \alpha_i^{s-1} + \alpha_j^{s-1}$, so (52) is linear in $\alpha_i^{s-1}$: there are two relations $\boldsymbol{\zeta}'$ and $\boldsymbol{\zeta}''$ such that the interpolation relation of each such column $i$ is a multiple of $\alpha_i^{s-1}\boldsymbol{\zeta}' + \boldsymbol{\zeta}''$, and

$$\zeta''_{j,\mathbf{a}} = \alpha_j^{s-1}\, \zeta'_{j,\mathbf{a}} \quad \text{for } 1 \le j \le n \text{ and } s_j - s < |\mathbf{a}| < s_j \tag{53}$$

[37, App. C], [80, Thm. 5]. Since $x \mapsto x^{s-1}$ is an automorphism of $\mathbb{K}$, (53) determines the support values $\alpha_j$.

**Remark 26.** *The coefficients (52) also give cross-ratios of the support, as the entries $m_{j,i}$ do in the Sidelnikov–Shestakov attack. For two columns $i \neq i'$ with the property above and two further columns $j \neq j'$, the factors $\xi$ and $\nu$ cancel in the ratio of four coefficients:*

$$\frac{\zeta^{(i)}_{j,\mathbf{a}}\, \zeta^{(i')}_{j',\mathbf{a}'}}{\zeta^{(i')}_{j,\mathbf{a}}\, \zeta^{(i)}_{j',\mathbf{a}'}} = \left( \frac{(\alpha_i - \alpha_j)(\alpha_{i'} - \alpha_{j'})}{(\alpha_{i'} - \alpha_j)(\alpha_i - \alpha_{j'})} \right)^{s-1} \qquad \textit{for } |\mathbf{a}| = s_j - 1 \textit{ and } |\mathbf{a}'| = s_{j'} - 1. \tag{54}$$

*The right-hand side is a power of the cross-ratio in (2). As there, these ratios determine the support up to a Möbius transformation. The attacks of [37] and [80] exploit the relations differently. They prescribe the support values at three columns, which fixes the Möbius transformation, and obtain $\alpha_j^{s-1}$ at every other constraint column $j$ from the pair $(\boldsymbol{\zeta}', \boldsymbol{\zeta}'')$ of (53): [37] by testing candidate values, and [80] as the ratio of the entries of the two relations.*

The interpolation relation (51) is not determined by its entries at column $i$. The relations with zero entries at column $i$ include relations that hold for arbitrary points $\mathbf{y}_j$: by Euler's identity,

$$\sum_{l=1}^{k} y_{j,l}\, \frac{\partial h}{\partial X_l}(\mathbf{y}_j) = d\, h(\mathbf{y}_j) \quad \text{for all } h \in \mathcal{P}$$

at every column $j$, where $y_{j,l}$ are the coordinates of $\mathbf{y}_j$, and similar identities relate the forms of higher orders (item (iv) of Section 3.4). Such relations, which involve the forms of a few columns only, are called *local relations* [37, Sec. 7.1]. Adding them to (51) changes the coefficients (52), and (53) no longer holds. For a subfield subcode, the forms have coefficients in $\mathbb{F}_q$, so the interpolation relations of the other Frobenius conjugate branches are relations as well.

In contrast to the dual code in the case $d = 1$, the relations are expected to contain nothing else: for a suitable choice of the multiplicities, the local relations and the relations that come from interpolation on the $m$ branches are conjectured to span all relations [37, App. C], [80, Assumption 3]. The attacks differ in how they separate (51) from the local relations.

- *The key recovery of* [37, Sec. 7] tests candidate support values. It prescribes a support value $x_i$ at each of three columns $i$ of multiplicity $s$, which is possible because the support is determined only up to a Möbius transformation. For a further column $j$ and a candidate value $x \in \mathbb{K}$ of $\alpha_j$, it looks for two relations $\boldsymbol{\zeta}'$ and $\boldsymbol{\zeta}''$ such that, for each prescribed column $i$, the relation $x_i^{s-1}\boldsymbol{\zeta}' + \boldsymbol{\zeta}''$ has zero entries at all positive-order forms of column $i$. At one of these three columns, its entry at the

value form $h(\mathbf{y}_i)$ is fixed to 1. The relations must also satisfy

$$\zeta''_{j,\mathbf{a}} = x^{s-1}\,\zeta'_{j,\mathbf{a}} \quad \text{for } s_j - s < |\mathbf{a}| < s_j.$$

For a given $x$, these are linear equations in the entries of $\boldsymbol{\zeta}'$ and $\boldsymbol{\zeta}''$, and the candidate is retained if they have a solution [37, Alg. 4]. The correct value is retained by (53). The attack relies on the conjecture that the other values are rejected, apart from those of the other Frobenius conjugate branches [37, Conjecture C.3]; a fourth prescribed column selects one branch. Saarinen reported a binary Goppa code with $m = 6$ and $n = 60$ on which every candidate value was retained at every tested column, so that the test did not determine the support there [68, Sec. 7].

The attack solves such a system for each column and each candidate, which gives an estimated cost of $2^{130.4}$ to $2^{149.0}$ bit operations for Classic McEliece [37, Table 3], or $2^{126.8}$ to $2^{145.2}$ in the count of [68, Table 4]. The two-anchor variant of Saarinen [70], which solved TII-254, replaces the candidate tests by linear algebra on computed relations.

- *The single-run attack of Weis* [80, Sec. 5] removes the local relations before reading the coefficients. It describes the local relations at one column and between two columns explicitly [80, Prop. 6]. For codes with no dual codeword of weight at most $2d - 2$, it assumes that they span all relations together with the relations that come from interpolation on the $m$ Frobenius conjugate branches [80, Assumption 3]. The pairs $(\boldsymbol{\zeta}', \boldsymbol{\zeta}'')$ of the $m$ branches are then computed, modulo the local relations, from 256 solutions of the linear system (49) over $\mathbb{F}_2$, and linear algebra at the prescribed columns separates the branches [80, Sec. 5.3]. These solutions are obtained in one run of a sparse solver, which is why the estimated cost of the attack, $2^{89.4}$ to $2^{98.4}$ bit operations [80, Table 4], is close to that of one run of the distinguisher. The remaining support is recovered by completion after guessing one support value at a unit column.

### 10.3. Vanishing polynomials versus interpolation relations

The two methods of this paper and the single-run attack of Weis [80, Sec. 5] are the cheapest attacks of their categories, and for Classic McEliece they cost essentially the same (Table 6, [80, Table 4]). At the same cost, recovering the key from vanishing polynomials has several advantages.

- *Universality.* Our methods are not tied to Goppa codes or to characteristic two, because they use only that the public code is a subcode of a GRS code. In the experiments of Section 8.1, they recovered keys of binary Goppa codes, wild Goppa codes, alternant codes and GRS subcodes in characteristic two, and direct recovery also did so in odd characteristic. The attacks from interpolation relations need characteristic two, and the attack of Weis is limited to binary Goppa codes [80, Sec. 5].

- *Modularity, simplicity and interpretability.* Our methods are modular and conceptually simple, because each intermediate step corresponds to a clean geometric object of the hidden curve: a local jet, its tangent planes, a GRS code on the original support. Each step therefore has a geometric interpretation and its own sufficient condition (Section 9.4), and it depends only on the object produced by the previous step, not on how this object was obtained. Bootstrapping, for instance, works equally well from Frobenius closures (Remark 25). As far as we know, the attacks from interpolation relations have neither such modularity nor such a geometric interpretation: their intermediate data are relations modulo local relations.

- *More natural starting point.* Vanishing polynomials are the more natural object to start from: every polynomial in a hold-out kernel is an equation of the hidden curve, so there is no noise to remove. In contrast, relations are linear dependencies among the conditions, and almost all of them are local relations, which have nothing to do with the code and have to be removed before the support can be read [80, Sec. 5.2].

- *Complexity of post-processing.* After sampling, our methods use only dense linear algebra and series arithmetic over $\mathbb{K}$, which take at most $2^{74.0}$ to $2^{85.1}$ bit operations, depending on the method and the parameter set (Table 7). The attack of Weis removes the local relations by a sparse elimination at every column, estimated at $2^{80}$ to $2^{88}$ bit operations [80, Sec. 5.3]. The key recovery of [37] does not remove them. Instead, it solves a sparse linear system for every column and every candidate value, which is why it costs $2^{126.8}$ to $2^{145.2}$ bit operations [68, Table 4].

- *Heuristic assumptions.* Both approaches rely on heuristic assumptions. Ours are the conditions of Section 9.4. The attack of Weis assumes that the linear system has no relations besides the local relations and the interpolation relations [80, Assumption 3]. We do not know how these assumptions are related, and so far it is hard to tell which of them are less restrictive. There is, however, some evidence that our methods are less fragile in edge cases (see, e.g., the degree-seven experiments in Section 8.1).

- *Fallbacks.* A failure of our stated conditions does not completely block key recovery. If the branches are not separated at the held-out position, one can add the equations of higher order, which needs no new computation, or move to another held-out position, which needs one more hold-out kernel (see Section 8.1). If a deep jet cannot be recovered at all, tangent information could be enough: bootstrapping also works from jets of low order at several positions, or from Frobenius closures of tangent planes alone (Section 5.4).

  Even partial information of this kind can be reinforced. A shallow jet or a Frobenius closure imposes roots on the curve with far fewer equations than Hasse-derivative conditions (Lemma 7), so adding it to a new hold-out system lowers the multiplicities needed at the other columns and gives a less restrictive hold-out kernel. We do not know of comparable fallbacks for the attacks from interpolation relations.

- *Parameters.* Our methods restrict the parameters less. The attacks from interpolation relations need an exact number of imposed zeros, and the multiplicity $s$ of the columns

with prescribed support values must be such that $s-1$ is a power of two [80, Thm. 5]. Our methods need neither. The conditions on the imposed zeros are inequalities (Proposition 3, Assumption 1), which leaves room to choose the multiplicities for the lowest cost, and several experiments used $s = 4$ (Tables 3 and 4).

Nevertheless, the attacks from interpolation relations give a complementary view on the breakthrough distinguisher of [37]. Several independent routes from this distinguisher to the secret key, found concurrently by different authors, only strengthen the evidence that it leads to key recovery.

### 10.4. History

The first version of [37], dated 7 August 2026, left key recovery open. The initial working draft of this paper [75], dated 20 August, introduced jet recovery from the hold-out polynomials, with quadratic equations separating the Frobenius conjugate branches. Key recovery used the derivative spaces spanned by the jets, which do not depend on the local parametrization or scaling: we guessed part of the support and built a linear system for the curve from the derivative spaces at the guessed positions (Section B.1). This approach was experimentally validated on small non-subfield subcodes and small non-binary alternant codes.

Subsequent experiments showed that, for binary Goppa codes, determining the curve uniquely needed too many correct guesses, so the 26 August version used the system only to reject incorrect partial support guesses, which still worked with few guessed points. The same day, Apon [6] published a proof that at least $2t + 3$ correct support values are needed before the system uniquely determines the curve, and guessing that many points, as the attack of the 20 August draft would require, costs far more than generic decoding.

On 27 August, the second version of [37] added key recovery from interpolation relations, extending the Sidelnikov–Shestakov attack to interpolation with multiplicities (Section 10.2). Our 2 September version introduced the minor-code attack from tangent planes at all positions. Since the branches were separated at each position independently, it required aligning the branches recovered at different positions. That version also included an alignment based on squares of multiplicity shortenings, which, however, imposed restrictive parameter conditions. Our 6 September version replaced it by a more efficient attack based on extended minor codes, which adjoins the planes of all conjugate branches and then separates the Frobenius blocks, for binary Goppa codes through the square of the dual code (Section B.2).

On 11 September, Weis published the first version of [80]. In [80], he improved the parameter selection of the distinguisher for binary Goppa codes and lowered the cost per run by about 20 bits. Additionally, [80] proposed two attacks: a gradient attack and a further improvement of the key-recovery attack of [37]. Like our initial minor-code attack, the gradient attack uses tangent information at each position. Unlike it, the gradient attack does not separate the Frobenius conjugate branches locally: it computes only the Frobenius closures of the tangent planes (Section 10.1), and its further key recovery relies heavily on properties of Goppa codes. Computing the closures at all columns requires 103

to 1,443 distinguisher runs. The second attack of [80] improves the key recovery of [37] by identifying and removing the local relations that obstruct it at concrete parameters. Its cost is close to that of a single distinguisher run. It solved the TII-253 challenge.

The same day, Saarinen [70] reported solving TII-254 with a two-anchor variant of the attack of [37], and his 15 September revision of [68] gave itemized bit-operation estimates for that attack on the Classic McEliece parameter sets. Later versions of [80], up to 6 October, revised the cost analysis and proved two of its assumptions.

Our 8 October version introduced the two methods of this paper. The present version adds the degree-seven experiments (Section 8.1), a sufficient condition for full minor codes of binary Goppa codes (Proposition 14), and this comparison.

## 11. Conclusion

We presented two methods that extend the hold-out distinguisher of [37] for subcodes of GRS codes to full key recovery. Their common starting point is a local jet of the hidden curve, recovered from a sample of hold-out polynomials at one position. This local information yields the secret support either through bootstrapping and the minor code, or directly through a quotient of quadratic forms. Under the stated conditions, both methods use a single hold-out computation, whose cost dominates the remaining steps: about $2^{92.7}$ to $2^{101.2}$ Boolean gates for Classic McEliece, excluding memory access. The construction applies to several families of GRS subcodes, while the differential structure of binary Goppa codes provides additional equations and stronger recovery criteria.

Our implementations recovered full secret keys for TII-173, TII-249, and TII-253, as well as for randomly generated binary Goppa codes, wild Goppa codes, alternant codes, and GRS subcodes. Synthetic experiments at shortened Classic McEliece sizes also support the conditions for the finishing steps.

Several questions remain open. The first is to theoretically establish when the required rank conditions of Section 9.4 hold and bound their failure probabilities, as well as to analyze the fallback options for the cases where they could fail (cf. Section 10.3). Another is to understand the connection between the attacks from vanishing polynomials and the attacks from interpolation relations. The last is to lower the cost of the hold-out computation itself, which now dominates key recovery, or to find some cheaper approaches for supplying the required information for key recovery (e.g., this information could be leveraged via some variants of bootstrapping).

Given the results of this paper and of the concurrent works [37, 80], binary Goppa codes, as well as other relatives of GRS codes, no longer appear to be a conservative foundation for code-based cryptography, even though the attacks are still conditional and not yet practical. For almost fifty years, binary Goppa codes were the exception among the algebraic codes proposed for the McEliece cryptosystem, but this is no longer the case. Moving to larger parameters (for instance, by increasing $m$) would require even larger public keys without guaranteeing resistance to further cryptanalytic advances, particularly given the rapid progress against Goppa and alternant codes over the past few years [9, 10, 25, 37, 40, 41, 67]. By contrast, neither of these concerns applies to schemes whose

security rests on less structured or random codes, such as the approach of Alekhnovich [5], HQC [2, 59], and QC-MDPC cryptosystems [7, 61].

## References


[1] Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. "Reed–Muller Codes for Random Erasures and Errors". In: *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2015, pp. 297–306.

[2] Carlos Aguilar-Melchor et al. "Efficient encryption from random quasi-cyclic codes". In: *IEEE Transactions on Information Theory* 64.5 (2018), pp. 3927–3943.

[3] Martin R. Albrecht et al. *Classic McEliece: conservative code-based cryptography*. Round-4 submission to the NIST Post-Quantum Cryptography Standardization Project. Submission package. Oct. 23, 2022. URL: https://classic.mceliece.org/nist.html.

[4] Martin R. Albrecht et al. *Classic McEliece: conservative code-based cryptography: guide for security reviewers*. Supporting documentation, Round-4 submission to the NIST Post-Quantum Cryptography Standardization Project. Oct. 23, 2022. URL: https://classic.mceliece.org/mceliece-security-20221023.pdf.

[5] M. Alekhnovich. "More on average case vs approximation complexity". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* 2003, pp. 298–307.

[6] Daniel Apon. *An Algebraic-Geometry Lower Bound against the ePrint:2026/1747 McEliece Key-Recovery Attack*. Cryptology ePrint Archive, Paper 2026/1810. 2026. URL: https://eprint.iacr.org/2026/1810.

[7] N. Aragon et al. *BIKE: Bit Flipping Key Encapsulation, Round 4 Submission*. NIST. Oct. 2024. URL: https://bikesuite.org.

[8] Marco Baldi et al. "Enhanced Public Key Security for the McEliece Cryptosystem". In: *Journal of Cryptology* 29.1 (Aug. 2014), pp. 1–27. ISSN: 1432-1378. DOI: 10.1007/s00145-014-9187-8.

[9] Magali Bardet, Axel Lemoine, and Jean-Pierre Tillich. *An Attack on the CFS Scheme and on TII McEliece Challenges*. Cryptology ePrint Archive, Paper 2026/430. 2026. URL: https://eprint.iacr.org/2026/430.

[10] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. *Polynomial Time Key-Recovery Attack on High Rate Random Alternant Codes*. arXiv:2304.14757. 2023. URL: https://arxiv.org/abs/2304.14757.

[11] Magali Bardet et al. "Cryptanalysis of the McEliece Public Key Cryptosystem Based on Polar Codes". In: *Post-Quantum Cryptography (PQCrypto 2016)*. Lecture Notes in Computer Science. Springer, 2016, pp. 118–143. DOI: 10.1007/978-3-319-29360-8_9.

[12] Magali Bardet et al. "Practical Algebraic Attack on DAGS". In: *Code-Based Cryptography (CBCrypto 2019)*. Lecture Notes in Computer Science. Springer, 2019, pp. 86–101. DOI: 10.1007/978-3-030-25922-8_5.

[13] Élise Barelli and Alain Couvreur. “An Efficient Structural Attack on NIST Submission DAGS”. In: *Advances in Cryptology – ASIACRYPT 2018*. Lecture Notes in Computer Science. Springer, 2018, pp. 93–118. DOI: 10.1007/978-3-030-03326-2_4.

[14] Anja Becker et al. “Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding”. In: *Advances in Cryptology – EUROCRYPT 2012*. Springer, 2012, pp. 520–536. DOI: 10.1007/978-3-642-29011-4_31.

[15] Thierry P. Berger. “On the Cyclicity of Goppa Codes, Parity-Check Subcodes of Goppa Codes, and Extended Goppa Codes”. In: *Finite Fields and Their Applications* 6.3 (2000), pp. 255–281. DOI: 10.1006/ffta.2000.0277.

[16] Thierry P. Berger, Cheikh Thiecoumba Gueye, and Jean Belo Klamti. “Generalized Subspace Subcodes With Application in Cryptology”. In: *IEEE Transactions on Information Theory* 65.8 (Aug. 2019), pp. 4641–4657. ISSN: 0018-9448. DOI: 10.1109/TIT.2019.2909872.

[17] Thierry P. Berger and Pierre Loidreau. “How to mask the structure of codes for a cryptographic use”. In: *Designs, Codes and Cryptography* 35.1 (2005), pp. 63–79. DOI: 10.1007/s10623-003-6151-2.

[18] Thierry P. Berger et al. “Designing a Public Key Cryptosystem Based on Quasi-cyclic Subspace Subcodes of Reed-Solomon Codes”. In: *Algebra, Codes and Cryptology (A2C 2019)*. Vol. 1133. Communications in Computer and Information Science. Springer, 2019, pp. 97–113. DOI: 10.1007/978-3-030-36237-9_6.

[19] Mikhail A. Borodin and Ivan V. Chizhov. “Effective attack on the McEliece cryptosystem based on Reed-Muller codes”. In: *Discrete Mathematics and Applications* 24.5 (2014), pp. 273–280. DOI: 10.1515/dma-2014-0024.

[20] Pierre Briaud et al. *A Heuristic Subexponential Attack on the McEliece Cryptosystem*. Cryptology ePrint Archive, Paper 2026/1232. 2026. URL: https://eprint.iacr.org/2026/1232.

[21] Don Coppersmith. “Solving Homogeneous Linear Equations over GF(2) via Block Wiedemann Algorithm”. In: *Mathematics of Computation* 62.205 (1994), pp. 333–350.

[22] Alain Couvreur and Matthieu Lequesne. “On the Security of Subspace Subcodes of Reed–Solomon Codes for Public Key Encryption”. In: *IEEE Transactions on Information Theory* 68.1 (Jan. 2022), pp. 632–648. ISSN: 0018-9448. DOI: 10.1109/TIT.2021.3120440.

[23] Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Tillich. “Recovering short secret keys of RLCE in polynomial time”. In: *Post-Quantum Cryptography (PQCrypto 2019)*. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 133–152. DOI: 10.1007/978-3-030-25510-7_8.

[24] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. “Cryptanalysis of McEliece Cryptosystem Based on Algebraic Geometry Codes and Their Subcodes”. In: *IEEE Transactions on Information Theory* 63.8 (2017), pp. 5404–5418. DOI: 10.1109/TIT.2017.2712636.

[25] Alain Couvreur, Rocco Mora, and Jean-Pierre Tillich. "A New Approach Based on Quadratic Forms to Attack the McEliece Cryptosystem". In: *Advances in Cryptology – ASIACRYPT 2023*. Springer Nature Singapore, 2023, pp. 3–38. ISBN: 9789819987306. DOI: 10.1007/978-981-99-8730-6_1.

[26] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. "Polynomial Time Attack on Wild McEliece Over Quadratic Extensions". In: *IEEE Transactions on Information Theory* 63.1 (Jan. 2017), pp. 404–427. ISSN: 1557-9654. DOI: 10.1109/tit.2016.2574841.

[27] Alain Couvreur et al. "A Polynomial-Time Attack on the BBCRS Scheme". In: *Public-Key Cryptography – PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Berlin: Springer, 2015, pp. 175–193. DOI: 10.1007/978-3-662-46447-2_8.

[28] Alain Couvreur et al. "Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes". In: *Designs, Codes and Cryptography* 73.2 (Apr. 2014), pp. 641–666. ISSN: 1573-7586. DOI: 10.1007/s10623-014-9967-z.

[29] Philippe Delsarte, Jean-Marie Goethals, and F. Jessie MacWilliams. "On Generalized Reed–Muller Codes and Their Relatives". In: *Information and Control* 16.5 (1970), pp. 403–442.

[30] Ilya Dumer. "On Minimum Distance Decoding of Linear Codes". In: *Proceedings of the 5th Joint Soviet–Swedish International Workshop on Information Theory*. Moscow, 1991, pp. 50–52.

[31] Wayne Eberly. "Yet Another Block Lanczos Algorithm: How to Simplify the Computation and Reduce Reliance on Preconditioners in the Small Field Case". In: *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC 2010)*. ACM, 2010, pp. 289–296.

[32] Freja Elbro and Christian Majenz. "An Algebraic Attack Against McEliece-like Cryptosystems Based on BCH Codes". In: *2023 IEEE Information Theory Workshop (ITW)*. IEEE, 2023, pp. 70–75. DOI: 10.1109/ITW55543.2023.10161620.

[33] Jean-Charles Faugère et al. "A Distinguisher for High-Rate McEliece Cryptosystems". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6830–6844. DOI: 10.1109/TIT.2013.2272036.

[34] Jean-Charles Faugère et al. "Algebraic Cryptanalysis of McEliece Variants with Compact Keys". In: *Advances in Cryptology – EUROCRYPT 2010*. Lecture Notes in Computer Science. Springer, 2010, pp. 279–298. DOI: 10.1007/978-3-642-13190-5_14.

[35] Jean-Charles Faugère et al. "Structural Cryptanalysis of McEliece Schemes with Compact Keys". In: *Designs, Codes and Cryptography* 79.1 (2016), pp. 87–112. DOI: 10.1007/s10623-015-0036-z.

[36] Valérie Gauthier-Umaña and Gregor Leander. *Practical Key Recovery Attacks On Two McEliece Variants*. Cryptology ePrint Archive, Paper 2009/509. 2009. URL: https://eprint.iacr.org/2009/509.

[37] Ashrujit Ghoshal et al. *Quasipolynomial Cryptanalysis of the McEliece Cryptosystem (or: PIR Meets McEliece)*. Cryptology ePrint Archive, Paper 2026/1630. 2026. URL: https://eprint.iacr.org/2026/1630.

[38] V. D. Goppa. "A New Class of Linear Correcting Codes". In: *Probl. Peredachi Inf.* 6.3 (1970), pp. 24–30. URL: http://mi.mathnet.ru/ppi1748.

[39] Petra W. Heijnen and Ruud Pellikaan. "Generalized Hamming Weights of $q$-ary Reed–Muller Codes". In: *IEEE Transactions on Information Theory* 44.1 (1998), pp. 181–196.

[40] Tobias Hemmert. *Key Recovery for the McEliece Cryptosystem Using Higher-Order Vanishing*. Cryptology ePrint Archive, Paper 2026/1339. 2026. URL: https://eprint.iacr.org/2026/1339.

[41] Tobias Hemmert and Andreas Wiemers. "Distinguishing Goppa Codes Using Higher-Order Vanishing". In: *Advances in Cryptology – CRYPTO 2026*. Lecture Notes in Computer Science. Full version: Cryptology ePrint Archive, Paper 2025/1661. Springer, 2026, pp. 191–223. DOI: 10.1007/978-3-032-35398-6_7.

[42] Fedor Ivanov, Eugenii Krouk, and Victor Zyablov. "New Code-Based Cryptosystem Based on Binary Image of Generalized Reed-Solomon Code". In: *2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*. 2021, pp. 66–69. ISBN: 978-1-6654-3308-2. DOI: 10.1109/REDUNDANCY52534.2021.9606467.

[43] Fedor Ivanov et al. "A New Code-Based Cryptosystem". In: *Code-Based Cryptography (CBCrypto 2020)*. Ed. by Marco Baldi, Edoardo Persichetti, and Paolo Santini. Vol. 12087. Lecture Notes in Computer Science. Springer, 2020, pp. 41–49. ISBN: 9783030540746. DOI: 10.1007/978-3-030-54074-6_3.

[44] Tadao Kasami, Shu Lin, and W. Wesley Peterson. "New Generalizations of the Reed–Muller Codes–I: Primitive Codes". In: *IEEE Transactions on Information Theory* 14.2 (1968), pp. 189–199. DOI: 10.1109/TIT.1968.1054127.

[45] Elena Kirshanova and Andre Esser. *TII McEliece Challenges: Scripts and Instances*. GitHub repository. Challenge description at https://crowdchallenge.tii.ae/mceliece-challenges/. 2023. URL: https://github.com/ElenaKirshanova/tii_decoding_challenge.

[46] Elena Kirshanova and Alexander May. "Breaking Goppa-based McEliece with hints". In: *Information and Computation* 293 (Aug. 2023), p. 105045. ISSN: 0890-5401. DOI: 10.1016/j.ic.2023.105045.

[47] Evgenii Krouk, Grigory Kabatiansky, and Cedric Tavernier. "McEliece-type cryptosystem based on correction of errors and erasures". In: *2023 XVIII International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY)*. 2023, pp. 173–177. DOI: 10.1109/redundancy59964.2023.10330197.

[48] Grégory Landais and Jean-Pierre Tillich. “An Efficient Attack of a McEliece Cryptosystem Variant Based on Convolutional Codes”. In: *Post-Quantum Cryptography (PQCrypto 2013)*. Lecture Notes in Computer Science. Springer, 2013, pp. 102–117. DOI: 10.1007/978-3-642-38616-9_7.

[49] Terry Shue Chien Lau and Chik How Tan. “Polynomial-time plaintext recovery attacks on the IKKR code-based cryptosystems”. In: *Advances in Mathematics of Communications* 17.2 (2023), pp. 353–366. ISSN: 1930-5338. DOI: 10.3934/amc.2020132.

[50] Terry Shue Chien Lau et al. “New code-based cryptosystems via the IKKR framework”. In: *Journal of Information Security and Applications* 76 (Aug. 2023), p. 103530. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2023.103530.

[51] Julien Lavauzelle and Julian Renner. “Cryptanalysis of a System Based on Twisted Reed–Solomon Codes”. In: *Designs, Codes and Cryptography* 88.7 (2020), pp. 1285–1300. DOI: 10.1007/s10623-020-00747-6.

[52] Yongwoo Lee et al. “Cryptanalysis of the Ivanov-Kabatiansky-Krouk-Rumenko Cryptosystems”. In: *IEEE Communications Letters* 24.12 (Dec. 2020), pp. 2678–2681. ISSN: 2373-7891. DOI: 10.1109/lcomm.2020.3019054.

[53] Axel Lemoine. *The Tangent Space Attack*. arXiv:2505.10184. 2025. URL: https://arxiv.org/abs/2505.10184.

[54] Axel Lemoine, Rocco Mora, and Jean-Pierre Tillich. “Understanding the New Distinguisher of Alternant Codes at Degree 2”. In: *Designs, Codes and Cryptography* (2025). DOI: 10.1007/s10623-025-01626-8.

[55] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-correcting Codes*. Mathematical Library. Amsterdam: North-Holland Publishing Company, 1977. ISBN: 0444850090.

[56] Alexander May, Alexander Meurer, and Enrico Thomae. “Decoding Random Linear Codes in $\tilde{\mathcal{O}}(2^{0.054n})$”. In: *Advances in Cryptology – ASIACRYPT 2011*. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 107–124. DOI: 10.1007/978-3-642-25385-0_6.

[57] Alexander May and Ilya Ozerov. “On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes”. In: *Advances in Cryptology – EUROCRYPT 2015*. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 203–228. DOI: 10.1007/978-3-662-46800-5_9.

[58] Robert J. McEliece. “A public-key cryptosystem based on algebraic coding theory”. In: *The Deep Space Network Progress Report* 42-44 (1978), pp. 114–116.

[59] Carlos Aguilar Melchor et al. *Hamming Quasi-Cyclic (HQC), Round 4 Submission*. NIST. 2024. URL: https://pqc-hqc.org.

[60] Lorenz Minder and Amin Shokrollahi. “Cryptanalysis of the Sidelnikov cryptosystem”. In: *Advances in Cryptology – EUROCRYPT 2007*. Vol. 4515. Lecture Notes in Computer Science. Springer, 2007, pp. 347–360. DOI: 10.1007/978-3-540-72540-4_20.

[61] Rafael Misoczki et al. “MDPC-McEliece: New McEliece variants from moderate density parity-check codes”. In: *2013 IEEE International Symposium on Information Theory*. IEEE. 2013, pp. 2069–2073. DOI: 10.1109/ISIT.2013.6620590.

[62] Peter L. Montgomery. “A Block Lanczos Algorithm for Finding Dependencies over GF(2)”. In: *Advances in Cryptology – EUROCRYPT ’95*. Vol. 921. Lecture Notes in Computer Science. Springer, 1995, pp. 106–120.

[63] Rocco Mora and Jean-Pierre Tillich. “On the Dimension and Structure of the Square of the Dual of a Goppa Code”. In: *Designs, Codes and Cryptography* 91.4 (2023), pp. 1351–1372. DOI: 10.1007/s10623-022-01153-w.

[64] Harald Niederreiter. “Knapsack-type cryptosystems and algebraic coding theory”. In: *Problems of Control and Information Theory* 15.2 (1986), pp. 159–166.

[65] Eugene Prange. “The Use of Information Sets in Decoding Cyclic Codes”. In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9. DOI: 10.1109/TIT.1962.1057777.

[66] Sven Puchinger et al. “Code-Based Cryptosystems Using Generalized Concatenated Codes”. In: *Applications of Computer Algebra*. Springer International Publishing, 2017, pp. 397–423. ISBN: 9783319569321. DOI: 10.1007/978-3-319-56932-1_26.

[67] Hugues Randriambololona. “The Syzygy Distinguisher”. In: *Advances in Cryptology – EUROCRYPT 2025*. Springer Nature Switzerland, 2025, pp. 324–354. ISBN: 9783031910951. DOI: 10.1007/978-3-031-91095-1_12.

[68] Markku-Juhani O. Saarinen. *Bit Operation Cost of “Holdout” Key-Recovery Attacks Against Classic McEliece*. Cryptology ePrint Archive, Paper 2026/1786. 2026. URL: https://eprint.iacr.org/2026/1786.

[69] Markku-Juhani O. Saarinen. *HOVER: Higher-Order Vanishing Endomorphism Recovery*. Cryptology ePrint Archive, Paper 2026/1778. 2026. URL: https://eprint.iacr.org/2026/1778.

[70] Markku-Juhani O. Saarinen. *Two-Anchor Holdout/Hermite: Solving the TII-254 McEliece Key Recovery Challenge*. Cryptology ePrint Archive, Paper 2026/1986. Version of September 24, 2026. 2026. URL: https://eprint.iacr.org/2026/1986.

[71] Nicolas Sendrier. “On the Concatenated Structure of a Linear Code”. In: *Applicable Algebra in Engineering, Communication and Computing* 9.3 (1998), pp. 221–242. DOI: 10.1007/s002000050104.

[72] V. M. Sidelnikov. “A public-key cryptosystem based on binary Reed-Muller codes”. In: *Discrete Mathematics and Applications* 4.3 (1994), pp. 191–208. DOI: 10.1515/dma.1994.4.3.191.

[73] V. M. Sidelnikov and S. O. Shestakov. “On insecurity of cryptosystems based on generalized Reed-Solomon codes”. In: *Discrete Mathematics and Applications* 2.4 (1992), pp. 439–444. ISSN: 0924-9265. DOI: 10.1515/dma.1992.2.4.439.

[74] Jacques Stern. “A Method for Finding Codewords of Small Weight”. In: *Coding Theory and Applications*. Vol. 388. Lecture Notes in Computer Science. Springer, 1989, pp. 106–113. DOI: 10.1007/BFb0019850.

[75] Kirill Vedenev. *Extending Distinguishing to Key Recovery for Subfield Subcodes of GRS codes*. Cryptology ePrint Archive, Paper 2026/1747. 2026. URL: https://eprint.iacr.org/2026/1747.

[76] Kirill Vedenev. “On the security of two IKKR-type code-based cryptosystems”. In: *Designs, Codes and Cryptography* 93 (2025), pp. 4779–4798. DOI: 10.1007/s10623-025-01683-z.

[77] Kirill Vedenev and Yury Kosolapov. “Cryptanalysis of Ivanov–Krouk–Zyablov Cryptosystem”. In: *Code-Based Cryptography (CBCrypto 2022)*. Ed. by Jean-Christophe Deneuville. Lecture Notes in Computer Science. Springer Nature Switzerland, 2023, pp. 137–153. DOI: 10.1007/978-3-031-29689-5_8.

[78] Yongge Wang. “Quantum resistant random linear code based public key encryption scheme RLCE”. In: *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2016, pp. 2519–2523. DOI: 10.1109/ISIT.2016.7541753.

[79] Victor K. Wei. “Generalized Hamming Weights for Linear Codes”. In: *IEEE Transactions on Information Theory* 37.5 (1991), pp. 1412–1418.

[80] Stephen A. Weis. *Improving GIJS Key Recovery for Classic McEliece*. Cryptology ePrint Archive, Paper 2026/1984. Version 0.5, October 2026. 2026. URL: https://eprint.iacr.org/2026/1984.

[81] Christian Wieschebrink. “Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes”. In: *Post-Quantum Cryptography (PQCrypto 2010)*. Vol. 6061. Lecture Notes in Computer Science. Springer, 2010, pp. 61–72. DOI: 10.1007/978-3-642-12929-2_5.

[82] Christian Wieschebrink. “Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography”. In: *2006 IEEE International Symposium on Information Theory*. 2006, pp. 1733–1737. DOI: 10.1109/isit.2006.261651.

## Appendix A. Derivative spaces and multiplicity shortenings

### A.1. Derivative spaces

For $r \geq 0$, let

$$V_i^{[r]} = \operatorname{span}\{\boldsymbol{\tau}_0, \dots, \boldsymbol{\tau}_r\} = \operatorname{span}\{(\partial^{[e]}\mathbf{F})(\alpha_i) : 0 \leq e \leq r\}$$

be the *derivative space* of order $r$ at position $i$. We call the nested sequence $(V_i^{[r]})_{r\geq 0}$ the *derivative flag* at that position.

Recall that, at position $i$, the jet-recovery procedure of Section 4 returns the coefficients of a local representative of the form

$$\mathbf{B}_i(T) = u_i(T)\mathbf{P}_\sigma(\psi_i(T)), \qquad u_i(0) = 1, \quad \psi_i(0) = 0, \quad \psi_i'(0) \neq 0.$$

The series $u_i$ and $\psi_i$ are part of the ambiguity in the recovered representative, and independent runs at different positions may return representatives corresponding to different pairs $(u_i, \psi_i)$. The next lemma shows that the derivative flag is independent of these choices and can therefore be obtained from any local jet recovered at position $i$.

**Lemma 9.** *Let $\mathbf{B}(T) = \sum_{r\geq 0} \mathbf{b}_r T^r \in \mathbb{K}[[T]]^k$ and $\widetilde{\mathbf{B}}(T) = u(T)\mathbf{B}(\psi(T))$, where $u, \psi \in \mathbb{K}[[T]]$ satisfy $u(0) \neq 0$, $\psi(0) = 0$ and $\psi'(0) \neq 0$. Then for every $r \geq 0$,*

$$\operatorname{span}\{[T^e]\widetilde{\mathbf{B}} : 0 \leq e \leq r\} = \operatorname{span}\{\mathbf{b}_0, \dots, \mathbf{b}_r\}.$$

*Proof.* Since $\psi(T)$ is divisible by $T$, the coefficient of $T^r$ in $u(T)\mathbf{B}(\psi(T))$ is a linear combination of $\mathbf{b}_0, \dots, \mathbf{b}_r$, in which $\mathbf{b}_r$ has the coefficient $u(0)\psi'(0)^r \neq 0$. The two families are therefore related by an invertible triangular matrix. □

**Remark 27.** *For non-subfield subcodes, $m = 1$, so there is no Frobenius-branch ambiguity. A recovered local jet $(\mathbf{v}_0, \dots, \mathbf{v}_r)$ gives $V_i^{[e]} = \operatorname{span}\{\mathbf{v}_0, \dots, \mathbf{v}_e\}$ for $0 \leq e \leq r$. For subfield subcodes, the recovered jet may lie on any branch $\mathbf{P}_\sigma$ and gives $\varphi^\sigma(V_i^{[e]})$ instead. The same $\sigma$ applies to the whole flag, but it may vary with the position $i$.*

### A.2. Multiplicity shortening

Derivative spaces are closely related to multiplicity shortenings of the public code. In particular, multiplicity shortenings can be computed from the derivative spaces, without knowing the support values.

For $i \in [\![1, n]\!]$ and $s \geq 0$, define the *multiplicity shortening* of $C$ at $i$ by

$$\mathcal{Z}_i^s(C) = \left\{(\lambda_1 g(\alpha_1), \dots, \lambda_n g(\alpha_n)) \in C \;\middle|\; g \in \mathbb{K}[Z]_{\leq D},\ (Z - \alpha_i)^s \mid g(Z)\right\}.$$

We have $\mathcal{Z}_i^0(C) = C$, and $\mathcal{Z}_i^1(C)$ is ordinary shortening with the zero coordinate retained. Put $V_i^{[-1]} = \{\mathbf{0}\}$ and

$$U_i^{[r]} = \sum_{\sigma=0}^{m-1} \varphi^\sigma(V_i^{[r]}) \quad \text{for } r \geq -1. \tag{55}$$

**Proposition 18.** *For every $i \in [\![1, n]\!]$ and $s \geq 0$,*

$$\mathcal{Z}_i^s(C) = \{\mathbf{mY} : \mathbf{m} \in \mathbb{F}_q^k,\ \mathbf{m} \perp V_i^{[s-1]}\}.$$

*If $L_i^s = \{\mathbf{m} \in \mathbb{F}_q^k : \mathbf{mY} \in \mathcal{Z}_i^s(C)\}$, then*

$$U_i^{[s-1]} = (L_i^s \otimes_{\mathbb{F}_q} \mathbb{K})^\perp, \qquad \dim_{\mathbb{F}_q} \mathcal{Z}_i^s(C) = k - \dim_{\mathbb{K}} U_i^{[s-1]}. \tag{56}$$

*Proof.* The representing polynomial of $\mathbf{mY}$ is $g_{\mathbf{m}} = \mathbf{m} \cdot \mathbf{F}$. Since $(\partial^{[e]} g_{\mathbf{m}})(\alpha_i) = \mathbf{m} \cdot (\partial^{[e]}\mathbf{F})(\alpha_i)$, its first $s$ Hasse coefficients vanish exactly when $\mathbf{m} \perp V_i^{[s-1]}$.

For $\mathbf{m} \in \mathbb{F}_q^k$, this is equivalent to $\mathbf{m} \perp U_i^{[s-1]}$, because $\varphi(\mathbf{m}\cdot\mathbf{v}) = \mathbf{m}\cdot\varphi(\mathbf{v})$. The Frobenius-stable space $(U_i^{[s-1]})^\perp$ is spanned by its $\mathbb{F}_q$-rational vectors, so $L_i^s \otimes_{\mathbb{F}_q} \mathbb{K} = (U_i^{[s-1]})^\perp$. Taking orthogonal complements and dimensions proves the remaining identities. □

Conditions at several positions are imposed by intersecting the shortened subcodes, or equivalently by annihilating the sum of their derivative spaces.

For $m = 1$ (i.e., the non-subfield case), we have $U_i^{[r]} = V_i^{[r]}$, so multiplicity shortenings and derivative spaces determine each other by linear algebra.

**Extended multiplicity shortening.** In the subfield-subcode case, ordinary multiplicity shortening depends only on the Frobenius closure $U_i^{[r]}$, and therefore does not distinguish the individual conjugates of $V_i^{[r]}$. To impose the condition on one conjugate, we extend the code to $\mathbb{K}$. The resulting extended multiplicity shortenings correspond one-to-one to the conjugate derivative spaces $\varphi^\sigma(V_i^{[r]})$. Write

$$C_{\mathrm{ext}} = C \otimes_{\mathbb{F}_q} \mathbb{K} = \{\mathbf{mY} : \mathbf{m} \in \mathbb{K}^k\}.$$

**Definition 6.** For $i \in [\![1, n]\!]$, $0 \leq \sigma < m$, and $s \geq 0$, let

$$\mathcal{Z}_i^s(C_{\mathrm{ext}}; \sigma) = \{\mathbf{mY} : \mathbf{m} \in \mathbb{K}^k,\ \mathbf{m} \perp \varphi^\sigma(V_i^{[s-1]})\}$$

be the *extended multiplicity shortening* associated with the Frobenius conjugate $\varphi^\sigma(V_i^{[s-1]})$.

The map $\mathbf{m} \mapsto \mathbf{mY}$ is injective. Hence the extended shortening determines the conjugate derivative space itself:

$$\varphi^\sigma(V_i^{[s-1]}) = \{\mathbf{m} \in \mathbb{K}^k : \mathbf{mY} \in \mathcal{Z}_i^s(C_{\mathrm{ext}}; \sigma)\}^\perp, \qquad \dim_{\mathbb{K}} \mathcal{Z}_i^s(C_{\mathrm{ext}}; \sigma) = k - \dim_{\mathbb{K}} V_i^{[s-1]}.$$

On the chosen branch, the extended shortening is a subcode of a multiplicity shortening of the ambient GRS code. Let $A = \mathrm{GRS}_{D+1}(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, and define $\mathcal{Z}_i^s(A)$ by the same formula as $\mathcal{Z}_i^s(C)$. Then

$$\mathcal{Z}_i^s(C_{\mathrm{ext}}; \sigma) = C_{\mathrm{ext}} \cap \varphi^\sigma\big(\mathcal{Z}_i^s(A)\big).$$

Indeed, $\mathbf{y}_j = \varphi^\sigma(\lambda_j)\mathbf{F}^{(\sigma)}(\varphi^\sigma(\alpha_j))$ for all $j$, so the representing polynomial of $\mathbf{mY}$ in $\varphi^\sigma(A)$ is $\mathbf{m} \cdot \mathbf{F}^{(\sigma)}$. Its first $s$ Hasse coefficients at $\varphi^\sigma(\alpha_i)$ are $\mathbf{m} \cdot \varphi^\sigma\big((\partial^{[e]}\mathbf{F})(\alpha_i)\big)$, $0 \leq e < s$, and

vanish exactly when $\mathbf{m} \perp \varphi^{\sigma}(V_i^{[s-1]})$. The other branches are unconstrained: for $\sigma' \neq \sigma$, the extended shortening lies in $\varphi^{\sigma'}(A)$, as $C_{\mathrm{ext}}$ does.

### A.3. Relation to filtration attacks

The filtration attack against full GRS codes in [28] recursively computes multiplicity shortenings of increasing order. By Proposition 18, computing $\mathcal{Z}_i^1(C), \dots, \mathcal{Z}_i^{r+1}(C)$ is equivalent to recovering the derivative flag through order $r$ when $m = 1$. Thus filtration recovers the flag spanned by the local jet constructed in Section 4, using a different recursion.

For codes $B, B' \subseteq \mathbb{K}^n$, write

$$B \star B' = \mathrm{span}_{\mathbb{K}}\{\mathbf{a} \star \mathbf{b} : \mathbf{a} \in B,\ \mathbf{b} \in B'\},$$

where $\mathbf{a} \star \mathbf{b} = (a_j b_j)_{j=1}^n$, and put $B^{\star 2} = B \star B$.

For a full GRS code $C = \mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ over $\mathbb{K} = \mathbb{F}_q$, with $3 \leq k \leq n/2$, put $C_s = \mathcal{Z}_i^s(C)$. Starting from the public code $C_0$ and its ordinary shortening $C_1$, [28, Theorem 21 and the following construction] gives

$$C_{s+1} = \{\mathbf{c} \in C_s : \mathbf{c} \star C_{s-1} \subseteq C_s^{\star 2}\} \quad \text{for } 1 \leq s \leq k-2. \tag{57}$$

Each step is a linear system built from the preceding two terms. The identity uses $C_{s+1} \star C_{s-1} = C_s^{\star 2}$ and injectivity of evaluation for products of degree at most $2k-2 < n$.

The attack on wild Goppa codes over quadratic extensions in [26] carries the filtration approach to subfield subcodes. The recursion is different: it uses shortening conditions and heuristic product and spanning steps [26, Sections 4.4–4.6]. Here the recovered subcodes determine the Frobenius closures $U_i^{[r]}$, so they do not in general identify the derivative spaces of individual branches.[4]

Both attacks recover the support from deep terms of the filtration: by quotients of bottom codewords for GRS codes, and by norm identities for quadratic wild-Goppa codes. These finishing steps use the full GRS code or the full special alternant code of the wild-Goppa construction. They do not directly apply to arbitrary proper subcodes. Our direct method in Section 7 also extracts support from deep local information, but uses the coefficients of a jet, which a derivative flag need not determine.

Filtration can nevertheless supply the inputs of our methods, which then give alternative finishing steps when their hypotheses hold. By Proposition 18, the multiplicity-two shortening at $i$ gives the Frobenius closure of the tangent plane,

$$E_i = U_i^{[1]} = (L_i^2 \otimes_{\mathbb{F}_q} \mathbb{K})^{\perp}. \tag{58}$$

For $m = 1$, $E_i$ is the tangent plane $V_i^{[1]}$ itself. If it is known at every public column, the minor-code method of Section 6 uses these planes directly.

[4] The cited attack assumes that $g$ is irreducible of degree greater than one. Its $g^{q+1}$ representation has polynomial vector $\widehat{\mathbf{F}}$ with $\mathbf{F} = g\widehat{\mathbf{F}}$ for the $g^q$ representation of Remark 2. Since $g$ does not vanish on the support, Lemma 9 gives the same derivative flags.

More generally, suppose that $E_i$ is known at positions $i \in S$ with $n + |S| > 2D$. From these spaces, the Frobenius-closure variant of Section 5.4 computes all quadratic equations of the curve over $\mathbb{F}_q$. Under the branch-separation and continuation hypotheses of Section 4, the method of that section recovers a local jet from these equations. We can continue this jet for direct key recovery (Section 7), or use it to bootstrap aligned tangent planes for the minor-code method (Section 5).

## Appendix B. Earlier attack routes

### B.1. Partial-support guessing

Our initial attack was based on the following principle. For simplicity, consider first the non-subfield case. Suppose the derivative flag at position $i$ is known through order $r_i$. The hidden polynomial vector must then satisfy

$$(\partial^{[e]}\mathbf{F})(\alpha_i) \in V_i^{[e]} \quad \text{for } 0 \le e \le r_i.$$

Let $\mathbf{H}_i^{[e]}$ be a parity-check matrix of $V_i^{[e]}$ considered as a linear code. The conditions can also be written as

$$\mathbf{H}_i^{[e]}(\partial^{[e]}\mathbf{F})(\alpha_i) = \mathbf{0} \quad \text{for } 0 \le e \le r_i.$$

If $\alpha_i$ is known, these are homogeneous linear equations in the coefficients of $\mathbf{F}$. At order $e$, the matrix $\mathbf{H}_i^{[e]}$ contributes $k - \dim_{\mathbb{K}} V_i^{[e]}$ equations.

Since $\dim V_i^{[e]} \le e + 1$, the flag cannot saturate before order $k - 1$. Recovering it up to saturation therefore supplies at least

$$\sum_{e=0}^{k-2}(k - e - 1) = \binom{k}{2}$$

equations.

We first recover flags at a small set $I$ of positions, choosing the number of positions and the recovered orders so that $|I| \ge 4$ and

$$N_{\mathrm{eq}} = \sum_{i \in I}\sum_{e=0}^{r_i}(k - \dim_{\mathbb{K}} V_i^{[e]}) > k(D+1). \tag{59}$$

The quantity $N_{\mathrm{eq}}$ counts the equations without accounting for possible dependencies among them. The support values $\alpha_i$ for $i \in I$ are unknown, so we guess a tuple $(a_i)_{i \in I}$, after affinely normalizing two of its entries to 0 and 1. For each guess, let $\mathcal{W}$ be the space of polynomial vectors $\mathbf{G} \in \mathbb{K}[Z]^k_{\le D}$ satisfying

$$(\partial^{[e]}\mathbf{G})(a_i) \in V_i^{[e]} \quad \text{for all } i \in I \text{ and } 0 \le e \le r_i. \tag{60}$$

For the correct guess, $\mathcal{W}$ contains the corresponding parametrization of the hidden curve.

In the subfield case, each recovered flag is determined only up to a Frobenius shift, which may vary with the position. Write $W_i^{[e]}$ for the recovered flag at position $i$. We

guess the relative shifts $\sigma_i$, fixing $\sigma_i = 0$ at one reference position, and replace $V_i^{[e]}$ in (60) by $\varphi^{\sigma_i}(W_i^{[e]})$. In our tests on generic GRS subcodes and random alternant codes, the correct support and shift guesses yielded a one-dimensional solution space. In the first version of the attack, we used a generator of $\mathcal{W}$ to recover the curve parametrization up to a scalar.

For binary Goppa codes, we experimentally observed that unique curve recovery typically required about $2t + 3$ support values, where $t$ is the degree of the Goppa polynomial. Still, with only a few values, we could reject every incorrect support-and-alignment guess we tested. The second version of the attack therefore used the system only as a rejection test; it did not require a one-dimensional solution space for the correct guess. Our heuristic is that an incorrect guess of the support and alignment (one that does not describe a valid ambient GRS representation) makes the equations from different positions sufficiently independent to give the system full column rank. Then $\mathcal{W} = \{\mathbf{0}\}$, so the guess is rejected. Independently, Apon gave an algebraic-geometric proof that, under a nondegeneracy hypothesis, at least $2t + 3$ correct support values are needed for the binary-Goppa system to determine the curve up to a scalar [6, Theorem 1]. Apon's work appeared concurrently with the updated version of the partial-support-guessing attack.

The refined attack also proposed a way to extend a surviving partial guess without computing new hold-out kernels. For a public column $\mathbf{y}_j$ and each unused support value $a$, form

$$\mathcal{W}_{j,a} = \{\mathbf{G} \in \mathcal{W} : \mathbf{G}(a) \in \mathbb{K}\mathbf{y}_j\}.$$

These are further linear conditions, so the dimension is obtained by a rank computation. We select $a$ if $\dim \mathcal{W}_{j,a}$ has a unique positive maximum, and replace $\mathcal{W}$ by $\mathcal{W}_{j,a}$. This selection is heuristic. If the maximum is not unique, we leave position $j$ aside and try another column. Each selected value extends the partial support, even if the curve is not yet uniquely determined. If $\mathcal{W}$ becomes one-dimensional, its generator lets us test the remaining support values against the public columns. Any resulting ambient GRS representation is checked against the public code.

### B.2. Extended minor codes

The minor-code method of Section 6 needs tangent planes on a common Frobenius conjugate branch. Without bootstrapping, Section 4 can recover a tangent plane at each public column separately, but each on an unknown branch:

$$W_j^{[1]} = \varphi^{\sigma_j}(V_j^{[1]}) \quad \text{for } 1 \le j \le n, \text{ with } \sigma_j \text{ unknown.}$$

The extended minor code uses these planes without aligning the branches. It adjoins the minor columns of all Frobenius conjugate planes at every position, and the branches are separated afterwards.

Let $\widetilde{\mathbf{T}}$ be the matrix of Definition 5 computed from the planes $W_j^{[1]}$ instead of $V_j^{[1]}$, and put

$$\mathbf{T}_{\text{ext}} = [\widetilde{\mathbf{T}} \mid \varphi(\widetilde{\mathbf{T}}) \mid \cdots \mid \varphi^{m-1}(\widetilde{\mathbf{T}})] \in \mathbb{K}^{\binom{k}{2} \times mn}.$$

We call $\mathcal{B} = \mathrm{RowSpan}_{\mathbb{K}}(\mathbf{T}_{\mathrm{ext}})$ the *extended minor code*. Since $\mathbf{y}_j \in \mathbb{F}_q^k$, column $j$ of $\varphi^{\sigma}(\widetilde{\mathbf{T}})$ is computed from the plane $\varphi^{\sigma+\sigma_j}(V_j^{[1]})$ with the basis $[\mathbf{y}_j \mid \varphi^{\sigma}(\hat{\mathbf{y}}_j)]$. Thus the $m$ columns at position $j$ belong to the $m$ branches, in an order shifted by the unknown $\sigma_j$. Let $\mathbf{P}$ be the permutation matrix that sorts the columns of $\mathbf{T}_{\mathrm{ext}}$ into $m$ column groups, one for each branch. Then

$$\mathcal{B}\mathbf{P} \subseteq \mathcal{T}_0 \oplus \mathcal{T}_1 \oplus \cdots \oplus \mathcal{T}_{m-1},$$

where $\mathcal{T}_\sigma$ is the minor code of the planes $\varphi^{\sigma}(V_j^{[1]})$, $1 \le j \le n$, computed with the bases used in $\mathbf{T}_{\mathrm{ext}}$. Under Heuristic 3, each $\mathcal{T}_\sigma$ is a full GRS code with support $\varphi^{\sigma}(\boldsymbol{\alpha})$ and dimension $k_{\mathcal{T}}$.

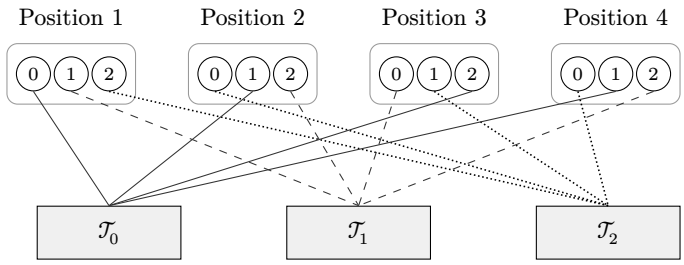



Figure 8: Illustrative Tanner graph of the extended minor code for $m = 3$ and four positions. Circles represent the symbols of one row of $\mathbf{T}_{\mathrm{ext}}$. The circle labelled $\sigma$ at position $j$ comes from column $j$ of $\varphi^{\sigma}(\widetilde{\mathbf{T}})$. The symbols joined to each bottom box must form a codeword of the indicated code. The unknown shifts $\sigma_j$ determine the connections.

For the generic alternant codes we tested, $\mathcal{B}\mathbf{P}$ was the whole direct sum whenever $\binom{k}{2} > mk_{\mathcal{T}}$. Row reduction then separates the branches. Call two columns linked if they are nonzero in a common row of the reduced row echelon form of $\mathbf{T}_{\mathrm{ext}}$. Since each $\mathcal{T}_\sigma$ is MDS, the classes of the equivalence relation generated by these links are exactly the $m$ column groups. Restricted to one group, $\mathbf{T}_{\mathrm{ext}}$ generates the full GRS code $\mathcal{T}_\sigma$, and the Sidelnikov–Shestakov attack recovers an equivalent support of $\varphi^{\sigma}(\boldsymbol{\alpha})$. The group contains exactly one column from each position, so every support value is assigned to its position. The multiplier and the shortened positions are then recovered as in Section 6.3.

For binary Goppa codes, $\mathcal{B}\mathbf{P}$ was observed to be a proper subcode of the direct sum (some of the specific coupling relations were also explicitly described in previous revisions), so the separation above does not apply. The column groups can be found in two ways.

The first uses the square of the dual code, with $\star$ as in Section A.3. In our tests with $m \ge 3$ and $n > 12t + 3$, $(\mathcal{B}^{\perp})^{\star 2}$ was a direct sum of $m$ codes supported on the column groups, none of which split further. Row reduction therefore recovered the groups. Restricted to each group, $\mathbf{T}_{\mathrm{ext}}$ generated a full GRS code of dimension $n - 3t - 1$.

The second applies the minor construction twice and needs a local jet of order 3 at every public column instead of a tangent plane. For $\mathbf{u}, \mathbf{v} \in \mathbb{K}^k$, write $\mathbf{u} \wedge \mathbf{v}$ for the vector of $2 \times 2$ minors of $[\mathbf{u} \mid \mathbf{v}]$, indexed as the rows of the minor matrix. Let $(\mathbf{v}_0, \ldots, \mathbf{v}_3)$ be a local jet of order 3 at position $j$, on an unknown branch. The *doubly minor matrix* $\mathbf{T}^{(2)}$

has column $j$ equal to

$$\sqrt[4]{(\mathbf{v}_0 \wedge \mathbf{v}_1) \wedge (\mathbf{v}_0 \wedge \mathbf{v}_3 + \mathbf{v}_1 \wedge \mathbf{v}_2)}.$$

By the Hasse product rule, this column equals $\sqrt{\mathbf{g}(\alpha_j) \wedge \mathbf{g}'(\alpha_j)}$ up to a nonzero factor and the Frobenius power of the branch, where $\mathbf{g} = (g_{a,b})_{a<b}$ is the polynomial vector of (33). Thus $\mathbf{T}^{(2)}$ plays the role of $\widetilde{\mathbf{T}}$ for the curve $\mathbf{g}$ instead of $\mathbf{F}$. The *extended doubly minor code* is the row space of $[\mathbf{T}^{(2)} \mid \varphi(\mathbf{T}^{(2)}) \mid \cdots \mid \varphi^{m-1}(\mathbf{T}^{(2)})]$. We conjectured that, given enough rows, it is the full direct sum of $m$ GRS codes also for binary Goppa codes. The branches then separate by row reduction as for alternant codes, without the assumptions of the dual-square method.