

Improving GJJS Key Recovery for Classic McEliece

Stephen A. Weis*
Anthropic
sweis@anthropic.com

October 2026[†]

Abstract

Classic McEliece is a post-quantum encryption scheme. Its public key is a generator matrix of a secret binary Goppa code. Its security levels are set by generic decoding attacks, which cost 2^{151} to 2^{287} bit operations for the five parameter sets. Ghoshal, Ishai, Jain and Sun [20] recently gave the first test that tells a public key from a random matrix at a lower cost, an estimated 2^{114} to 2^{124} bit operations. They have since extended it to recover the secret key. The test is one large sparse linear-algebra computation, which we call a run.

We improve this attack in two ways. First, we lower the cost of a run to 2^{89} to 2^{98} bit operations. Like theirs, this count skips the zero entries of the matrix. A circuit that is the same for all keys, which is what the figures for generic decoding count, has 2^{92} to 2^{101} gates. If every gate is charged for the memory that it addresses, at square-root cost, a run costs 2^{107} to 2^{117} . This is our best estimate, and it is below the cost of generic decoding with free memory. The Classic McEliece security guide charges every operation for the whole memory. Under that rule a run costs 2^{117} to 2^{128} , against 2^{159} to 2^{316} for generic decoding. Second, we give two ways to recover the secret key. One rests on a theorem that we prove and needs 100 to 1400 runs. The other builds on the key recovery of [20] and needs a single run.

None of this is close to practical, and a run at full size cannot be tested. The costs rest on four heuristic assumptions, which we state and test on small keys. There the first attack recovered the secret key with polynomials of degree 5, and the second with degrees 5, 6 and 7. Degree 7 is the degree of the attack on Classic McEliece. Small keys deviated from the assumptions in a few ways. We identify the cause of each deviation and argue that it does not occur at full size. As a demonstration we recovered the secret key of instance 253 of the TII McEliece key-recovery challenges ($m = 8$, $t = 9$, $n = 214$), a toy-sized code that had not been solved.

1 Introduction

Background. Classic McEliece [1] is the McEliece cryptosystem [26] with binary Goppa codes, essentially unchanged since 1978. The public key is a generator (or parity-check) matrix of a binary Goppa code of length n and dimension $k = n - mt$ that corrects t errors. The secret key is the support $\alpha = (\alpha_1, \dots, \alpha_n)$, a sequence of distinct elements of $\mathbb{F}_{2^m}$, together with the Goppa polynomial $\Gamma \in \mathbb{F}_{2^m}[Z]$ of degree t .

Two kinds of attack are relevant. The first is *generic decoding*, or information-set decoding (ISD). It ignores the algebraic structure, and its cost defines the security levels of the parameter

*This was prepared with Claude. I blame it for any mistakes, but take credit for the good parts.

[†]Version 0.5.

sets [2, 9]. The second kind is *structural*: it tries to detect or exploit the hidden Goppa structure. The classical example is the attack of Sidelnikov and Shestakov [37], which recovers the support of a generalised Reed–Solomon code from ratios of codeword entries. For Goppa codes, structural work includes the high-rate distinguisher of [18], the quadratic-form approach of [13] and the syzygy distinguisher of [31]. A survey is in [20, Section 1.4]. Until 2026 no structural attack applied to the Classic McEliece parameters at a cost anywhere near ISD. The most efficient one, the higher-order-vanishing distinguisher and key recovery of [22, 21], costs 2^{298} or more [20, Table 1]. It studies polynomials that vanish to high order at the columns of a shortened parity-check matrix. Two further structural results appeared in the first half of 2026. Bardet, Lemoine and Tillich [5] gave a polynomial-time key recovery for binary Goppa codes of very high rate. It breaks the CFS signature scheme but does not apply at the rates of Classic McEliece. Briaud, Lemoine, Randriambololona and Tillich [11] gave a key recovery that they conjecture to be subexponential. By their own lowest estimate its distinguishing step costs 2^{454} to 2^{915} for the Classic McEliece sets.

Ghoshal, Ishai, Jain and Sun (GIJS) [20] changed this. Their distinguisher, which they developed independently of [22], works with a generator matrix. It views the columns of the public matrix as points on a hidden rational curve. It looks for low-degree polynomials that vanish to high order at all columns but one. On a Goppa key such polynomials automatically vanish at the remaining column too. On a random matrix they do not. The test is a single large but sparse linear-algebra problem. GIJS estimate its cost for the five parameter sets at 2^{114} to 2^{124} bit operations [20, Section 5], which is 37 to 163 bits below ISD. Section 2 explains it. A later version of their paper adds a key-recovery algorithm. Further key-recovery algorithms and cost analyses that build on the same distinguisher have appeared [40, 33, 4]. Section 8 relates them to the present work.

This paper. Under heuristic assumptions, key recovery costs about 100 to 1400 runs of the distinguisher with our first attack, and a single run with the second. Our contributions are the following.

1. *Sharper parameters* (Section 3). We prove an identity for binary Goppa codes that gives the vanishing argument t extra zeros, and we prove that the sum of all public columns is one more point of the curve. With an exact count of the nonzero entries and one heuristic assumption, a run costs 2^{94} to 2^{102} .
2. *Key recovery from gradients* (Section 4). The gradient, at a held-out column, of any polynomial that the distinguisher finds is a codeword of an explicit subcode of the public code. These subcodes determine the key by linear algebra that takes minutes at full size.
3. *Key recovery from one run* (Section 5). The key recovery of [20, Section 7] rests on the observation that the coefficients of the interpolation relation behind the distinguisher carry the support, in the manner of Sidelnikov and Shestakov [37]. For binary Goppa codes this was a conjecture. We prove it, following a reduction of Saarinen [33]. We also give, in closed form, the very large space of trivial relations that has to be removed first. This lets us read the support of a shortened code, a subcode of the public code on fewer coordinates, from about two hundred random kernel vectors of one run.
4. *Operations, data movement and memory* (Section 6). The product with the matrix decomposes into independent small dense products. The matrix need not be stored, and the long-distance traffic is smaller than the operation count by a factor of 2^8 . With Montgomery’s block Lanczos algorithm a run costs 2^{89} to 2^{98} bit operations, or 2^{92} to 2^{101} gates as a circuit that does not depend on the key.

parameter set	distinguisher		gradient attack		single-run attack			ISD
	[20]	Sec. 3	(a)	(b)	(a)	(b)	(c)	
mceliece348864	114.2	94.1	120.8	100.8	114.3	94.3	89.4	151
mceliece460896	120.3	99.3	128.9	109.8	120.3	99.3	95.4	191
mceliece6688128	124.0	102.4	131.0	110.2	124.0	102.4	98.4	257
mceliece6960119	123.0	101.6	130.3	109.8	123.0	101.6	97.6	257
mceliece8192128	124.0	102.4	131.0	110.2	124.0	102.4	98.4	287

Table 1: $\log_2$ of the number of bit operations. Columns (a) keep the parameter conditions and cost formulas of [20, Section 5], and add only the point w of Proposition 2. Columns (b) use the parameters of Section 3 with the same formulas. Column (c) uses the organisation of Section 6. It charges one operation for a nonzero matrix entry, where (a) and (b) charge two. The first column is [20, Table 2]. The ISD figures are those of [20, Table 1], based on [9]. They count the gates of a circuit. On that basis column (c) is three bits higher (Table 5).

Table 1 summarises the costs. The submission assigns the five sets to the NIST categories 1, 3, 5, 5, 5 [2], which are defined by key searches on AES that NIST estimates at 2^{143} , 2^{207} and 2^{272} gates [28]. We call these numbers the *NIST levels*.

The numbered propositions and theorems are proved. The other ingredients are dimension counts of the kind that is usual in algebraic cryptanalysis, and the behaviour of the solver. We state them as four assumptions. All four concern the large matrix, so they can only be tested on small keys (Section 7). Table 7 lists them, together with a few smaller points that we could not test. Small keys deviate from the dimension counts mainly through short linear dependencies among the columns of the shortened code, that is, dual codewords of low weight. We predict the size of this effect. The prediction was exact for separate codewords, and too high where many overlap. For the shortened Classic McEliece codes the expected number of such codewords is about 2^{-61} or less, and a search found none. Keys with $t \leq 5$ also have a few relations beyond those of our third assumption (Section 5.2).

A run of 2^{89} operations is a chain of 2^{30} dependent steps on 2^{38} unknowns, far beyond anything that can be run. The point is the comparison with ISD and with the NIST levels. Section 6 makes it for a circuit that does not depend on the key, which needs three bits more than column (c), and in several cost models for memory. For mceliece348864 the margin below 2^{143} is 51 bits without a charge for memory, and 26 to 43 bits with one.

2 Preliminaries

Binary Goppa codes. We follow the notation of [20], and Table 2 collects the symbols that recur. Let $q = 2^m$ with $m \geq 3$. The code $\text{Goppa}(\alpha, \Gamma)$ consists of the vectors $c \in \mathbb{F}_2^n$ with $\sum_j c_j/(Z - \alpha_j) \equiv 0 \pmod{\Gamma}$. Here $\Gamma \in \mathbb{F}_q[Z]$ is monic and square-free of degree t (irreducible in Classic McEliece), and $\Gamma(\alpha_j) \neq 0$ for all j . The code has dimension $k = n - mt$, and the public key is a generator matrix $Y = [y_1 | \dots | y_n] \in \mathbb{F}_2^{k \times n}$. Write $\Pi(Z) = \prod_j (Z - \alpha_j)$ and $D = n - 2t - 1$.

All attacks below use one fact. A binary Goppa code lies in a generalised Reed–Solomon (GRS) code of dimension $D + 1$ [20, Fact 2.3]. That is, with $\lambda_j = \Gamma(\alpha_j)^2/\Pi'(\alpha_j)$, every codeword c has a *code polynomial* f_c of degree at most D such that $c_j = \lambda_j f_c(\alpha_j)$ for all j . It can be written down. Let $F = \prod_{c_j=1} (Z - \alpha_j)$ and $\bar{F} = \Pi/F$. Then $\sum_j c_j/(Z - \alpha_j) = F'/F$, so c lies in the code exactly when $\Gamma \mid F'$. In characteristic 2 the derivative F' is a square, and Γ is square-free, so in fact $\Gamma^2 \mid F'$.

The code polynomial is

$$f_c = F' \bar{F} / \Gamma^2. \quad (1)$$

For the rows of Y this gives $y_j = \lambda_j p(\alpha_j)$ with a vector $p = (f_1, \dots, f_k)$ of polynomials. In other words, up to the scalars λ_j , the columns of Y are points of a rational curve of degree D in $\mathbb{F}_q^k$, given in unknown coordinates.

The instance. The distinguisher [20, Sections 3–5] does not work with the full public code. It *shortens* the code on ℓ coordinates: it keeps the codewords that are zero there and deletes these coordinates. It then brings the generator matrix to systematic form, in which $k_\ell = k - \ell$ of the $n_\ell = k_\ell + mt$ columns are unit vectors. The result is again a Goppa code, with the same Γ and the surviving support elements. We call it *the instance*. It has k_ℓ *unit columns* and mt *non-unit columns*. We set $D_\ell = n_\ell - 2t - 1$, and from now on Π and p are those of the instance.

The jet matrix. Fix a degree d . A homogeneous multilinear polynomial h of degree d in k_ℓ variables is a vector of $N = \binom{k_\ell}{d}$ coefficients h_A , one for each d -set A of variables. Its Hasse derivative $\partial^{[S]} h(y)$ at a point y , for a set S of variables, is the coefficient of X^S in $h(X + y)$, and $|S|$ is its order. We say that h vanishes to order s at y if all derivatives of order below s vanish there. For a fixed point this is a set of linear conditions on the coefficients of h . The condition of order 0 is the *value condition* $h(y) = 0$. A non-unit column y_j at which the conditions of all orders below s_j are imposed is a *constraint column* of *multiplicity* s_j . Most columns have the same multiplicity $s < d$. Unit columns need no conditions, because a multilinear form of degree d vanishes to order $d - 1$ at a unit vector.

The conditions are the columns of a binary matrix E with N rows, the *jet matrix* of [20, Section 5.3]. A column of E is indexed by a pair (j, S) . Its entry in row A is 1 exactly when $S \subseteq A$ and y_j is 1 on $A \setminus S$. The columns that belong to one y_j form the *block* of y_j .

The distinguisher. The *row kernel* $\{h : hE = 0\}$ is the space of polynomials that vanish to the prescribed orders. The distinguisher keeps one non-unit column y_τ , the *held-out column*, out of the constraint set. It computes a nonzero h in the row kernel and tests whether $h(y_\tau) = 0$.

Here is why this works [20, Theorems 3.2 and 4.1]. Restrict h to the curve: the univariate polynomial $\varphi_h(Z) = h(p(Z))$ has degree at most dD_ℓ . It vanishes to order s at α_j for every constraint column and to order $d - 1$ at the support elements of the unit columns. Suppose that these forced zeros number more than dD_ℓ . For $d = s + 1$ this is the first condition of GIJS [20, condition (6)]:

$$s(n_\ell - 1) > dD_\ell. \quad (2)$$

Then $\varphi_h \equiv 0$, and in particular $h(y_\tau) = \lambda_\tau^d \varphi_h(\alpha_\tau) = 0$. On a random matrix a nonzero h has $h(y_\tau) \neq 0$ with probability at least 2^{-d} . To guarantee a nonzero row kernel, GIJS require that E has more rows than columns [20, condition (7)]:

$$\binom{k_\ell}{d} > (mt - 1) \sum_{u < s} \binom{k_\ell}{u}. \quad (3)$$

The cost is that of finding kernel vectors of the sparse matrix E . We call one such computation *a run*. GIJS give cost formulas for a block Lanczos algorithm [20, Section 5.3], which we reuse except where stated.

m, t, n, k	field degree ($q = 2^m$), number of errors, length and dimension $k = n - mt$ of the public code
α_j, Γ	secret support elements and Goppa polynomial
k_ℓ, n_ℓ	dimension and length $n_\ell = k_\ell + mt$ of the instance, the shortened code in systematic form
D_ℓ	degree $n_\ell - 2t - 1$ of the curve on which the columns of the instance lie
d, s	degree of the polynomials h , and multiplicity at most constraint columns ($d = s + 1$)
E, N, J	the jet matrix, its number of rows, which is the number of coefficients of h , and its number of columns
P	number of constraint points
w	sum of all columns of the instance (Proposition 2)
κ	predicted dimension of the row kernel of E , equation (4)
δ	shortfall of the forced zeros against the degree (Section 3)
H	number of held-out columns in a run of the gradient attack
s'	multiplicity of the held-out columns in the single-run attack
b	block width of the solver. A run returns about b kernel vectors
T, V, M, L	bit operations, bits moved, memory of a run and memory of a tile (Section 6)
T_f, V_f	the counts T and V for a circuit that does not depend on the key
γ	exponent of the memory charge: an access to M bits costs M^γ
Ω, Ω', Ψ	random involutions of the rows and of the columns of E , and a sparse matrix with $\Psi\Psi^\top = \Omega'$, used by the solver

Table 2: Notation.

The column kernel. The same matrix has a *column kernel* $\{\rho : E\rho = 0\}$, the space of linear relations among the conditions. The two kernels are linked. Suppose that $h(y_\tau) = 0$ for every h in the row kernel. Then the value condition at y_τ is a linear combination of the other conditions. Once this value condition is added to E , the combination is a column-kernel vector. Section 4 works with the row kernel, and Section 5 with the column kernel.

Branches. The matrix E is binary, so it cannot tell the support α from its Frobenius conjugates $\alpha^{2^u} = (\alpha_1^{2^u}, \dots, \alpha_n^{2^u})$, $u < m$. Each conjugate is the support of an equivalent key. A key recovery must pick one of these m conjugates, which we call *branches*, and stay with it across all columns.

3 Sharper parameters for the distinguisher

The cost of a run grows quickly with $N = \binom{k_\ell}{d}$, so the aim is a small degree d and a small k_ℓ . Two conditions limit this. There must be enough forced zeros (2), which bounds k_ℓ from above. The row kernel must be nonzero (3), which bounds k_ℓ from below. Our reimplementations of the cost analysis of [20, Section 5] reproduces its Table 2 to both decimals. Four observations tighten the analysis. The first is a count, the second is a standard heuristic, and the last two are proved. Together they move every parameter set from $(d, s) = (8, 7)$ to $(7, 6)$ and save about twenty bits.

1. Sparsity (a count). GIJS bound the number of nonzero entries that one constraint column contributes to one row of E by $\sum_{u < s} \binom{d}{u}$. If a of the d variables of the monomial are 0 at y_j , the true number is $\sum_{u < s} \binom{d-a}{u-a}$, and $a \sim \text{Bin}(d, \frac{1}{2})$ for *balanced* columns, which have about as many

ones as zeros. Moreover, in characteristic 2 the conditions of orders $d - 2, d - 4, d - 8, \dots$ imply all the others [22, Corollary 1]. A run keeps only these, and we call the result the *reduced matrix*. The average is then 7.4 at (7, 6), instead of the bound 120. This saves about 4 bits.

2. Kernel dimension (heuristic). Condition (3) treats the conditions of different columns as independent. They are not. The standard genericity heuristic for such systems [22, Heuristic 1], [19] predicts that the row kernel has dimension

$$\kappa = [z^d] [(1 + z)^{k_\ell} (1 - z^2)^P]_+. \quad (4)$$

Here P is the number of points at which conditions are imposed, all with multiplicity $d - 1$, and $[\cdot]_+$ truncates the series at its first non-positive coefficient. A point of multiplicity $\mu < d - 1$ contributes $1 - z^{d-\mu+1}$ in place of $1 - z^2$ [20, Appendix C.2]. The distinguisher needs no more than a nonzero kernel, so we select parameters by $\kappa > 0$ instead of (3). This saves about 5 bits. Parameters are *at the threshold* if one more point, or one unit less of k_ℓ , gives $\kappa = 0$.

Assumption 1. At the parameters used in this paper, the row kernel of a run has dimension at least (4).

3. Binary Goppa codes need t zeros fewer (proved). GIJS observe that on binary Goppa keys the products of d code polynomials span a space of dimension $dD_\ell + 1 - t$, which is t less than for a generic GRS code [20, Section 7.3]. The reason is an identity. Omitted proofs are in Appendix A.

Proposition 1. *Let $h = f_1 \cdots f_d$ be a product of code polynomials (1). Then $\Pi h' \equiv d\Pi' h \pmod{\Gamma}$. Let $\mathcal{D}_d$ be the space of all polynomials φ of degree at most dD_ℓ with $\Pi\varphi' \equiv d\Pi'\varphi \pmod{\Gamma}$. Then the $\mathbb{F}_q$ -span R_d of all such products lies in $\mathcal{D}_d$, and $\dim \mathcal{D}_d = dD_\ell + 1 - t$ for $d \geq 2$.*

The polynomials φ_h lie in R_d . So they lie in a space that is t dimensions smaller than their degree suggests, and t fewer zeros should suffice to force $\varphi_h \equiv 0$. Theorem 3 below makes this precise. Equality $R_d = \mathcal{D}_d$ held in every instance with $k \geq 8$ that we computed. For a single code polynomial the identity is exact, $\Pi f' + \Pi' f = \Gamma^2 f^2$ [4, Lemma 3].

4. The point at infinity (proved). The curve has one more point with binary coordinates, and it is public.

Proposition 2. *For every codeword c of the instance, the coefficient of Z^{D_ℓ} in f_c is the parity $\sum_j c_j$. Hence the sum w of all columns of the instance is the vector of leading coefficients of p . Imposing the conditions of order below s at w forces the top s coefficients of every φ_h to vanish.*

So w is one more point at which conditions can be imposed, and it adds s zeros to (2). The constraint columns together with w are the *constraint points*.

The zero count. Let μ_j be the order of the zero of φ_h that a run forces at α_j : $d - 1$ at a unit column, the multiplicity at a constraint column, and 0 at a held-out column. Let μ_∞ be the multiplicity of w . The *shortfall* is $\delta = dD_\ell + 1 - \sum_j \mu_j - \mu_\infty$. Condition (2) says $\delta \leq 0$. The following theorem allows $\delta \leq t$.

Theorem 3. *Fix the support of the instance and the orders μ_j and μ_∞ , with $1 \leq \delta \leq t$ and with $\mu_j + d$ odd for at least δ columns. Let Γ be uniform over the monic irreducible polynomials of degree t over $\mathbb{F}_q$, where $q \geq 16$ and $t \geq 2$. With probability at least $1 - n_\ell q^{-(t-\delta+1)}$, the only polynomial in $\mathcal{D}_d$ that has a zero of order μ_j at every α_j and degree at most $dD_\ell - \mu_\infty$ is 0.*

parameter set	(m, t)	distinguisher				gradient attack (b)			
		k_ℓ	n_ℓ	$\log_2 N$	run	k_ℓ	H	runs	total
mceliece348864	(12, 64)	154	922	38.4	94.1	154	9	103	100.8
mceliece460896	(13, 96)	195	1443	40.8	99.3	195	1	1443	109.8
mceliece6688128	(13, 128)	224	1888	42.2	102.4	223	8	236	110.2
mceliece6960119	(13, 119)	216	1763	41.8	101.6	216	6	294	109.8
mceliece8192128	(13, 128)	224	1888	42.2	102.4	223	8	236	110.2

Table 3: Parameters at $(d, s) = (7, 6)$ and $\log_2$ costs with the formulas of [20]. $N = \binom{k_\ell}{7}$ is the number of unknowns of a run, and H is the number of columns that a run of the gradient attack holds out. For comparison, [20, Table 2] has $(d, s) = (8, 7)$ and $k_\ell = 217, 274, 315, 304, 315$.

Since $\varphi_h \in R_d \subseteq \mathcal{D}_d$, this gives $\varphi_h \equiv 0$ for every h in the row kernel. Classic McEliece draws Γ uniformly and independently of the support, and then rejects keys without a systematic form [1], which multiplies the failure probability by at most $1/0.29$. In Table 3 the shortfall is at most $t - 3$, so this probability is below 2^{-39} . With $d = s + 1$ and one held-out column, $\delta \leq t$ reads $k_\ell \leq t(2s + 3 - m)$, in place of $k_\ell \leq t(2s + 2 - m)$ for (2). This is what permits $s = 6$, and it saves 11 to 12 bits. For mceliece460896 the bound is 192, below the smallest k_ℓ with $\kappa > 0$, and $s = 6$ also needs the zeros of w .

Resulting parameters. Table 3 minimises the cost of a run, as given by [20, Section 5.3], subject to $\kappa > 0$ and $\delta \leq t$. One formula is adapted. Their solver reduces E to a square system whose size is bounded by the number of columns. Our E has 1.5 times as many columns as rows, so we use N .

4 Key recovery from gradients

The distinguisher only looks at the value of h at the held-out column. Its first derivatives there say more. Since h vanishes along the whole curve, its gradient at a point of the curve is orthogonal to the tangent of the curve at that point, and the tangent depends on the secret support.

Consider a run on a Goppa key in which $\varphi_h \equiv 0$ for every h in the row kernel, by (2) or by Theorem 3. Let y_τ be a column of the instance at which at most the value condition was imposed. Let $g = \nabla h(y_\tau) \in \mathbb{F}_2^{k_\ell}$ be the vector of first-order derivatives of h at y_τ . Read g as a message, and let c be the corresponding codeword of the instance, so that $c_j = \langle g, y_j \rangle$.

Theorem 4. *The codeword c lies in $C'_\tau = \text{Goppa}(\alpha \setminus \{\alpha_\tau\}, \Gamma \cdot (Z - \alpha_\tau))$. That is, $c_\tau = 0$ and $\sum_{j \neq \tau} c_j / (\alpha_j - \alpha_\tau) = 0$. These are $m + 1$ binary linear conditions.*

Proof. By Euler's identity, $c_\tau = \langle g, y_\tau \rangle = d \cdot h(y_\tau) = 0$. Next, differentiate $\varphi_h = h \circ p \equiv 0$ at α_τ . The partial derivative $\partial_i h$ is homogeneous of degree $d - 1$, and $y_\tau = \lambda_\tau p(\alpha_\tau)$, so

$$0 = \varphi'_h(\alpha_\tau) = \sum_i (\partial_i h)(p(\alpha_\tau)) f'_i(\alpha_\tau) = \lambda_\tau^{-(d-1)} \sum_i g_i f'_i(\alpha_\tau).$$

Hence the code polynomial $f_c = \sum_i g_i f_i$ satisfies $f_c(\alpha_\tau) = f'_c(\alpha_\tau) = 0$. By (1), $f_c = \bar{F} \sigma^2$ with $\sigma^2 = F'/\Gamma^2$. Since $c_\tau = 0$, the element α_τ is a simple root of $\bar{F}$. Therefore $f'_c(\alpha_\tau) = \bar{F}'(\alpha_\tau) \sigma(\alpha_\tau)^2$ with $\bar{F}'(\alpha_\tau) \neq 0$, and so $\sigma(\alpha_\tau) = 0$. This means $F'(\alpha_\tau) = 0$, that is, $F(\alpha_\tau) \sum_{c_j=1} (\alpha_\tau - \alpha_j)^{-1} = 0$, and $F(\alpha_\tau) \neq 0$. $\square$

So each column τ has its own subcode C'_τ of the public code, defined by α_τ , and every kernel vector of a run gives one element of it. The theorem proves containment. That the gradients *span* C'_τ is a genericity statement. It has a known exception. A *dual codeword* is a linear dependency among columns of the instance, and its weight is the number of these columns.

Assumption 2. At the parameters of Table 3, let τ be a column that lies on no dual codeword of weight at most $d + 1$ of the instance. Then the gradients at y_τ of $k_\ell + 16$ random elements of the row kernel span C'_τ .

On a dual codeword of weight $d + 1$ the span is $d - 1$ dimensions short [20, Appendix A.2]. At Classic McEliece sizes the expected number of such codewords through a given column is below 2^{-97} .

The attack. Repeat the following until every column of the instance has been held out once. Pick H new columns, the held-out columns of this run, and take a systematic form in which they are non-unit columns. Impose only the value condition at these H columns, and the conditions of all orders below s at all other non-unit columns and at w . Compute $k_\ell + 16$ random row-kernel vectors and record the span of their gradients at each of the H columns. These runs dominate the cost. The rest is linear algebra with at most $n_\ell m \leq 2.5 \cdot 10^4$ unknowns (Appendix B). One linear system matches the branches across the columns and yields, for one branch, an explicit GRS code of dimension $3t + 1$. The support is then read as in the attack of Sidelnikov and Shestakov [37]: two codewords of a GRS code whose polynomials share all roots but one have entries with the ratio $(\alpha_j - a)/(\alpha_j - a')$. This gives the support up to a Möbius map $x \mapsto (ax + b)/(cx + e)$, which is all that an equivalent key needs. The multipliers λ_j follow as in [41], and the shortened-away coordinates by the completion step of [23].

The number of runs. A column with only the value condition contributes one zero instead of s . So H is the slack in the zero count divided by $s - 1$, and the instance needs $\lceil n_\ell/H \rceil$ runs. The slack is counted against (2) in column (a), with the zeros of w included, which gives 110 to 381 runs at (8, 7), and against Theorem 3 in column (b). There we keep two zeros in reserve, so that all runs together fail with probability below 2^{-16} . Columns (a) and (b) assume that the solver of [20], whose blocks are 589 wide, returns the $k_\ell + 16$ kernel vectors at no extra cost, as block Krylov solvers normally do.

5 Key recovery from a single run

The gradient attack learns something only at the few columns that a run holds out. This section reads the support of *every* constraint column from one run, using the column kernel of E .

5.1 The structure of the interpolation relation

The distinguisher works because the value condition at a held-out column is a linear combination of the other conditions. On the curve this combination is Hermite interpolation: the value of a polynomial at α_τ is determined by its values and derivatives at the other α_j . The coefficients of an interpolation formula depend on the nodes in a simple way, so they reveal the support. The precise statement is due to [20, Section 7 and Appendix C.1], who prove it for GRS codes and conjecture it for binary Goppa codes. Saarinen [33, Section 7] reduced the binary Goppa case to

the nonsingularity of a $t \times t$ matrix. We give a self-contained proof along his lines and bound the probability that the matrix is singular.

Give every column j of the instance an order s_j : $d - 1$ at a unit column, the multiplicity at a constraint column. Give w an order $s_\infty \geq 0$. Designate a few constraint columns as *held-out columns*, all of the same multiplicity s' , and put $r = s' - 1$. Unlike the held-out column of the distinguisher they carry conditions. Theorem 5 expresses their value condition through the conditions at the other points. The *no-slack condition* for binary Goppa codes is

$$\sum_j s_j + s_\infty = \dim \mathcal{D}_d + s' = dD_\ell + 1 - t + s'. \quad (5)$$

A held-out column τ is *good* if $\mathcal{D}_d$ contains no nonzero polynomial of degree at most $dD_\ell - s_\infty$ with a zero of order s_j at every α_j , $j \neq \tau$. This is the situation of Theorem 3 with $\delta = t$. Let $\beta_j = \alpha_j^r$. The top r orders of the block of a column are its *locator block*.

Theorem 5. *Suppose that $r \geq 2$ is a power of two and that (5) holds. Then there are two vectors V_1, V_2 over $\mathbb{F}_q$, indexed by the columns of E , with the following properties.*

- (i) $V_2 = \beta_j V_1$ on the locator block of every column j .
- (ii) V_1 and V_2 lie in the column kernel of E . For every held-out column τ , the vector $\beta_\tau V_1 + V_2$ is zero on the block of τ , except possibly on the value condition.
- (iii) If τ is good, the coefficient of this vector on the value condition at τ is nonzero. So there is a scalar η_τ such that $\rho_\tau = \eta_\tau(\beta_\tau V_1 + V_2)$ has coefficient 1 there. This ρ_τ is the canonical relation of τ . It expresses the value condition at τ through the conditions at the other points.

For a fixed support and fixed orders, and Γ uniform as in Theorem 3, a held-out column fails to be good with probability at most $t[(n_\ell + t)/(2t)]/(q - 2)$, for $t \geq 3$ and $k_\ell \geq t$.

Since r is a power of two, β is a conjugate of the support. So V_1 and V_2 show the support element of every constraint column at once, as the ratio of their entries on that column's locator block. The column kernel of the binary matrix E is stable under Frobenius, so each of the m branches gives such a pair. The support is only defined up to a Möbius map. Three held-out columns fix this freedom: we normalise their β to $0, 1, \infty$. A fourth held-out column selects the branch. The bound on the probability is far from sharp. We measured $1/q$ at full size. The count (5) must be exact. With fewer conditions there is no relation, and with too many the relations form a larger family, in which the ratio is no longer determined.

5.2 Local relations and general position

Almost all of the column kernel of E has nothing to do with the code. It consists of relations among the conditions of a single column, and of relations that come from conditions that two columns share. These *local relations* form the space K_{loc} of [20, Section 7.1]. The algorithm of [20] removes them implicitly, by solving a fresh copy of the whole system for every column and every candidate value, about $n_\ell q$ solves. A single run suffices if K_{loc} is known explicitly.

Proposition 6. *For a multilinear h of degree d and $y \in \mathbb{F}_2^{k_\ell}$, let $\Delta_y^{(i)} h$ be the part of degree $d - i$ of $h(X + y)$. Let O_y be the span of the conditions at a column y of multiplicity s_y , and $e_y = d - s_y + 1$.*

- (i) *The condition of index S at y is the coefficient of X^S in $h(X + y)$. So O_y is spanned by the coefficients of the $\Delta_y^{(i)} h$ with $i \geq e_y$.*

- (ii) For two columns a, a' and $i \geq e_a, i' \geq e_{a'}$, every coefficient of $\Delta_a^{(i)} \Delta_{a'}^{(i')} h$ lies in $O_a \cap O_{a'}$. In the conditions of a it is, for a set C of size $d - i - i'$ and the support $\Sigma_{a'}$ of a' ,

$$g_{a \leftarrow a'}(C) = \sum_{U \subseteq \Sigma_{a'} \setminus C, |U|=i'} \partial^{[C \cup U]} h(a),$$

and symmetrically in the conditions of a' . The difference of the two expressions is a column-kernel vector of E supported on the two blocks.

At $(d, s) = (7, 6)$ two columns of multiplicity 6 have $\dim O_a \cap O_{a'} = \binom{k_\ell}{3}$, and the vectors with $(i, i') = (2, 2), (4, 2)$ and $(2, 4)$ span this space at $k_\ell = 22$, but not at $k_\ell \leq 20$.

Assumption 3. Take an instance at the parameters of Section 5.3 or Table 4 that has no dual codeword of weight at most $2d - 2$. Then the column kernel of E is spanned by three kinds of relations: relations internal to one block, the two-block relations of Proposition 6, and ms' relations lifted from univariate interpolation as in Theorem 5. The last stay linearly independent modulo the others. More precisely, the dimensions stated in Steps 2 to 4 below are correct.

This is the hypothesis of [20, Appendix C.2, equation (39)] with K_{loc} made explicit, except that they allow relations on up to three blocks. On the univariate side there are exactly s' relations per branch if some held-out column is good and $R_d = \mathcal{D}_d$, for which $R_2 = \mathcal{D}_2$ suffices [33, Section 7]. This held on every key we tried, also at full size. The lifted relations lower the rank of E , so the row kernel of such a run has dimension $\kappa + ms'$.

Small t . The lifted relations use only the values of h on the curve p . The conditions at a column j also determine the gradient of h along the curve, to order $s_j - 2$ at α_j . Relations among these quantities are column-kernel vectors of E as well, and Assumption 3 says that they give nothing new. We counted them in a model built from the secret key, for $t = 3, \dots, 8$ (Appendix D). There were no new ones for $s' < t$. For $s' = 5$ there were two per branch at $t = 4$, one at $t = 5$, and none from $t = 6$ on. One new relation per branch raises the dimension of the row kernel and the dimensions in Steps 2 and 3 by m . Classic McEliece has $t \geq 64$.

General position. Formula (4) and Assumption 3 describe points in general position. The columns of a small instance are not: they satisfy short linear dependencies, that is, the instance has dual codewords of low weight. Dependencies among $c \leq 2d - 2$ columns lower the rank of E further. Let the c points have multiplicities $\mu_1, \dots, \mu_c$, where a unit column counts with $d - 1$, and let $\sum_j h_j z^j = (1 - z)^{-(c-1)} \prod_i (1 - z^{d-\mu_i+1})$. Then the codeword removes

$$\varepsilon = \sum_{j \leq d} \max(0, -h_j) \binom{k_\ell - c + 1}{d - j} \quad (6)$$

conditions from the count of (4) (Appendix C). We write $\varepsilon(c)$ if all multiplicities are $d - 1$. The resulting relations are not among those of Assumption 3. In Step 2 below they raise the dimension by ε at each non-unit column of the codeword, and the algorithm tolerates a moderate number of them. In all our experiments these codewords and the relations for small t were the only sources of deviations from the predicted dimensions. Equation (6) gave the size exactly for separate codewords, and was too high where many of them overlap (Appendix D). Stern's algorithm [39] finds them, and an attacker who finds one can take another shortening. For a random code their expected number, $2^{-k_\ell} \sum_{c \leq 2d-2} \binom{n_\ell}{c}$, is about 2^{-61} for mceliece348864 and 2^{-98} or less for the other sets. It is not small for small keys.

5.3 The algorithm

Setting up the instance. Use all non-unit columns and w as constraint points. Choose four held-out columns of multiplicity $s' = 5$, and lower as many other columns from multiplicity 6 to 5 as (5) requires. The run also needs $\kappa > 0$: otherwise E has full row rank, and its column kernel is larger than Assumption 3 says. This gives $k_\ell = 155, 195, 224, 216, 224$ in column (b) of Table 1, and $(d, s) = (8, 7)$ with $k_\ell = 218, 274, 315, 304, 315$ in column (a).

Step 1. Perform one run that returns b random vectors of the column kernel of E . The steps below need somewhat more than $ms' \leq 65$ vectors, and we used $b = 200$ or 256 . This run dominates the cost. It uses the reduced matrix of Section 3. A kernel vector of the reduced matrix is a kernel vector of the full one, with zeros at the omitted orders, and Steps 2 to 5 work with the full blocks.

Step 2. Remove the local relations. For each constraint column j , restrict the b vectors to the block of j . Then work modulo three kinds of vectors: the relations internal to the block, the vectors $g_{j \leftarrow j'}$ for all other blocks j' , and the unit vectors of the conditions that carry no support information. The last are the value condition at a held-out column, and the orders up to $s_j - s'$ at the other columns. What remains lies in a space of dimension $(s' - 1)m$, with one dimension for each branch and each order of the locator block.

Step 3. Find the branches. Discard the combinations of the b vectors that vanish at all four held-out blocks. This keeps ms' combinations. Look for pairs (V_1, V_2) of them for which $\beta_\tau V_1 + V_2$ has no component on the derivative conditions at τ , for the three held-out columns normalised to $\beta_\tau = 0, 1, \infty$. This binary linear system has m independent solutions, one for each branch.

Step 4. Select a branch. At the fourth held-out column the m solutions give two binary matrices, A_1 from the V_1 -parts and A_2 from the V_2 -parts, with $A_2 = QA_1$ for a unique $m \times m$ matrix Q . The eigenvalues of Q are the m conjugates of the normalised β of that column, and a left eigenvector selects the branch. If the value lies in a proper subfield, which the characteristic polynomial shows, another column of multiplicity s' is used.

Step 5. Read the support. At every other constraint column j the selected branch gives two vectors $v_1, v_2 \in \mathbb{F}_q^{(s'-1)m}$ with $v_2 = \beta_j v_1$. On a dual codeword of low weight the combinations discarded in Step 3 need not vanish at j . The equation is then read modulo their span.

The unit columns. Steps 1 to 5 yield β_j for the mt non-unit columns. One more support element determines the key [23, 33], and we obtain it by a guess. Take each $x \in \mathbb{F}_q$ in turn as the value of one unit column. A greatest common divisor of two polynomials of degree about $mt/4$ then either yields the Goppa polynomial or rejects x (Appendix B). Multipliers and completion follow as in Section 4.

Cost of Steps 2 to 5. Step 2 is a sparse elimination for each column. With a block Krylov method it costs 2^{80} to 2^{88} bit operations for the whole instance, about ten bits below column (c). With a square-root charge for the memory of a block it stays eight to nine bits below the run. We used dense elimination, so this estimate is untested. The step can be prepared beside the run, because a few hundred random functionals per column that vanish on the generators suffice. Steps 3 to 5 are negligible.

6 The cost of a run: operations, data movement and memory

The formulas of [20, Section 5.3] count bit operations only, and they assume a solver that was chosen because it has a proof. By their storage formula the runs of column (b) need 2^{57} to 2^{62} bits, and 96% of this is for two auxiliary sparse matrices of that solver, its preconditioners. This section organises the run differently. By an *organisation* of the run we mean the choice of the solver, of what is stored, and of how the product with E is computed. For one and the same organisation the section counts the bit operations T , the bits V that move over long distances, and the memory M . It then compares the run with ISD in the same cost models. There are three ingredients.

1. Fewer unknowns. A column that is lowered from multiplicity 6 to 5 contributes $1 - z^3$ in place of $1 - z^2$ to (4). Taking this into account, and choosing the multiplicity of w , lowers the smallest k_ℓ with $\kappa > 0$ (Table 4). This saves up to 0.7 bits.

2. The product with E , tile by tile. The matrix E is never stored. Let $y_j[i]$ be the i -th coordinate of the constraint point y_j .

Proposition 7. *Let $h = (h_A)$ be a vector indexed by the d -sets A , and let $c = hE$. For a $(d-2)$ -set S and every point j of multiplicity $d-1$,*

$$c_{j,S} = \sum_{i < i', i, i' \notin S} y_j[i] y_j[i'] h_{S \cup \{i, i'\}}.$$

For balanced columns a quarter of the terms are nonzero. Over all S and j these are $\frac{1}{4} \binom{d}{2} PN$ terms. The conditions of the lower orders $d-4, d-8, \dots$ are sums of partial sums that occur in this computation.

So the P conditions with index S are the values of one quadratic form at the P points. We call this computation a *tile*. It gathers the $\binom{k_\ell - d + 2}{2}$ coefficients of the form and multiplies them by a binary matrix that it generates from the instance. The product with $E^\top$ is the same data flow reversed. The solver works on b vectors at once, so an entry of a vector is a word of b bits. At $(d, s) = (7, 6)$ every entry of h is needed in $\binom{d}{2} = 21$ tiles. So one product moves $21N$ words between the array that holds the vector and the tiles, and the lower orders add about $4N$ words. Inside the tiles the product performs $5.25 PN$ additions of words. So an addition is not a long-distance fetch. A word that arrives at a tile takes part in $P/4$ additions, that is, in 190 to 420 of them.

3. The solver. Kernel vectors of a large sparse matrix are found by iterative (Krylov) methods such as block Lanczos and block Wiedemann [12], the standard tools of integer factoring. They use the matrix only through products with blocks of b vectors, about N/b times in a row. GIJS assume the block Lanczos algorithm of Eberly [15], which applies the matrix two and a half times as often as the algorithms used in practice and needs two sparse preconditioners. We use Montgomery's block Lanczos algorithm [27] with $b = 256$, as in our experiments. It needs a symmetric matrix, and the usual choice is $E^\top E$. We take

$$B = \tilde{E}^\top \Omega \tilde{E} \quad \text{with} \quad \tilde{E} = E\Psi.$$

Here Ω is a random involution of the N rows of E that fixes half of them. The matrix Ψ is invertible, has at most three nonzero entries in each row, and satisfies $\Psi\Psi^\top = \Omega'$ for a random involution Ω' of the columns of the same kind (Appendix E). The column kernel of E is Ψ times that of $\tilde{E}$. So Ω and Ψ take the place of Eberly's preconditioners. They are much cheaper, but they come without a proof, and over $\mathbb{F}_2$ Montgomery's algorithm has none either.

Assumption 4. At the parameters of Tables 3 and 4, Montgomery’s block Lanczos algorithm, applied to B , terminates after $\text{rank}(E)/(b - 0.76)$ iterations. All but a few of the b vectors that it returns are nonzero elements of the column kernel of $\tilde{E}$. The same holds for the row kernel, with $E^\top$ in place of E and with the roles of Ω and Ω' exchanged.

Over $\mathbb{F}_2$ two things can go wrong. A vector z with $Bz = 0$ need not satisfy $\tilde{E}z = 0$. And there can be vectors with $B^2z = 0$ but $Bz \neq 0$, which the iteration cannot handle. Both are measured by the *defect*, the difference between $\text{rank}(E)$ and the rank of a high power of B . The algorithm returns random kernel vectors only if the defect is well below b . The assumption includes this, and we have no formula for the defect. It has two parts (Appendix E). The first is $\text{rank}(E) - \text{rank}(E^\top \Omega E)$, which depends on the column space of E . The second is essentially $\text{rank}(E) - \text{rank}(E \Omega' E^\top)$, which depends on the row space.

Runs on $E^\top E$. Most of our runs used $\Omega = 1$ and $\Psi = 1$. They had matrices with up to $1.2 \cdot 10^7$ columns, on Goppa keys and on random points. On Goppa keys all but one behaved as Assumption 4 states, and where the defect can be told apart from the effect of dual codewords it was at most 15. The exception is an instance with 25 points at degree 7 (Section 7). Its defect was 302, and the vectors that the algorithm returned were of no use. Small instances in general position at degree 7 failed in the same way when some of their points had multiplicity below 6 (Appendix E). In dense computations with at most eight points both parts of the defect reached the hundreds, and the thousands at degree 7, also close to the threshold (Appendix E). The attack has 769 to 1665 points, and κ is at most 0.5% of N , but we cannot predict its defect for $E^\top E$.

Runs with Ω and Ψ . In the dense computations the defect is then far smaller, at most 8 from three points of weight $k_\ell/2$ on. On 24 small instances at degree 7 the solver’s defect was at most 4. With Ω alone it reached 1484, and on $E^\top E$ it reached 56 608 (Appendix E). On eight small instances in general position with points of lower multiplicity, where the other two forms failed like on the instance with 25 points, it was at most 2. On the typical key at degree 7 of Section 7, with $2.0 \cdot 10^6$ rows, the runs for the column kernel and for the row kernel took 7683 and 7832 iterations, as Assumption 4 predicts, and the defects were 3 and 0. The ten runs at the threshold at degree 5 behaved in the same way. On the instance with 25 points, on which the other two forms failed, the run took 25 117 iterations, as predicted, and the defect was 2. On a second instance of this kind it took 23 728 iterations, and the defect was 3.

Cost. In the organisation by tiles Ω costs nothing, because the word of the monomial A is simply sent on to the tiles of $\Omega(A)$. The products with Ψ add 3% to the traffic (Appendix E).

The attacks need kernel vectors that are *random*. This follows from a rank condition that the run itself checks. The algorithm draws a uniformly random block U of b vectors, solves $BX = BU$, and returns combinations of the columns of $X + U$ and of its last block W that $\tilde{E}$ annihilates.

Proposition 8. *Suppose that the algorithm returns the ν columns of $(X + U)C + WC'$, where C has rank ν . Then these are ν independent, uniformly distributed elements of the column kernel of $\tilde{E}$, and their images under Ψ are such elements of the column kernel of E .*

Proof. The iteration starts from BU and uses U in no other way, so X and W are functions of this block. The matrices C and C' are computed from $\tilde{E}(X + U)$ and $\tilde{E}W$. Let K be a block of b independent uniform elements of the column kernel of $\tilde{E}$, independent of U . Replacing U by $U + K$ changes none of BU , X , W , $\tilde{E}(X + U)$, C , C' . So it adds KC to the output. For every fixed U the columns of KC are ν independent uniform elements of the kernel, because C has rank ν . Hence the output for $U + K$ is uniform. Finally $U + K$ has the same distribution as U , and Ψ maps the kernel of $\tilde{E}$ bijectively to that of E . $\square$

parameter set	k_ℓ	lowered	$\log_2 N$	skipping zeros		fixed circuit		memory	iterations
				T	V	T_f	V_f		
mceliece348864	150	40	38.1	89.4	81.8	92.3	82.4	49.7	30.1
mceliece460896	195	0	40.8	95.4	87.2	98.3	87.7	52.3	32.8
mceliece6688128	221	38	42.1	98.4	89.8	101.3	90.3	53.6	34.1
mceliece6960119	214	22	41.7	97.6	89.1	100.5	89.7	53.3	33.8
mceliece8192128	221	38	42.1	98.4	89.8	101.3	90.3	53.6	34.1

Table 4: The single run at $(d, s) = (7, 6)$ and $b = 256$. All figures from N on are $\log_2$. Lowered: columns of multiplicity 5 besides the four held-out ones. T : bit operations. V : bits moved between the tiles and the vectors. Appendix E has the formulas.

In our successful runs for the column kernel the hypothesis held for all but at most 5 of the 256 vectors.

Operations. Table 4 lists the counts. A run takes $T = 2^{89.4}$ to $2^{98.4}$ bit operations, which is column (c) of Table 1. Column (b) is the cost of the same attack by the formulas of [20] and is four to five bits higher. The difference has three sources.

- About three bits come from the solver and the product by tiles.
- Up to 0.7 bits come from the smaller number of unknowns.
- One bit is a change of unit. For a nonzero entry of E , [20] charge a multiplication and an addition, and we charge one operation.

What the counts mean. The algorithm is fixed by the parameter set, and the public key is its input. Nothing in it is chosen with knowledge of the key. The shortened coordinates and the held-out and lowered columns are arbitrary, and Ω , Ψ and the starting block are random. Setting up the instance is Gaussian elimination on the public matrix, which is negligible. We count the operations of this one algorithm in two ways.

- *Skipping zeros.* T is the number of operations AND and XOR on bits of data that the algorithm executes. For every entry of E , zero or not, it computes the bit $y_j[i]y_j[i']$ of Proposition 7 from the instance, by one AND that serves all b vectors. It adds the word only if this bit is 1. So the program is the same for all keys, and only the set of additions that it executes depends on the key. The bits that decide this are computed during the run, and T includes their cost. It is the term $2\binom{d}{2}PN$ in each iteration (Appendix E), which comes to 2^{83} to 2^{92} operations, or 1.3% to 1.4% of T . The count is for balanced columns, which generated keys match to about 1% (Appendix D). It leaves out the loops, the test of a computed bit, the computation of addresses, and reading and writing. The traffic V likewise counts only the words that are sent. The formulas of [20] also charge for the nonzero entries only, so T is the figure to set against theirs.
- *A fixed circuit.* T_f is the number of AND and XOR gates of a Boolean circuit that depends on the parameter set, on Ω and on Ψ , but not on the key. Its inputs are the public key and the starting block, and its output is the block of kernel vectors. It has to treat every entry of E and of the small dense matrices of the solver, and it has to send every word of the lower orders. It has about three bits more gates than T and moves V_f bits (Table 4). The ISD figures count the gates of a circuit as well [9].

A count like T describes one given algorithm. It is not a model of computation in which arbitrary attacks could be compared, because an algorithm can hide work in the steps that are left out. In a run these steps follow a pattern that is fixed by the parameter set, except for the tests of bits that the run has computed: the entries of E , the masks of the lower orders, and the entries of the small matrices of the solver. The fixed circuit does without these tests, at a cost of three bits. It covers the run only. Steps 2 to 5 of Section 5.3 are counted like T and are about ten bits below the run. For the five parameter sets, all comparisons with ISD and with the NIST levels use the fixed circuit. Only the example sets of Appendix F are computed with T and V , which are the lower counts.

With the same organisation the distinguisher costs the same as the single run, and the gradient attack costs $2^{96.6}$ to $2^{106.4}$ operations, counted like T . It uses up to 239 of the 256 vectors of each run.

Data movement. Within a product the 2^{29} to 2^{32} tiles are independent. What cannot be parallelised is the chain of 2^{30} to 2^{34} iterations. Between two products every word travels from the tiles to its position in the vector and back to 21 tiles. This is an exchange among all parts of a machine of 100 TiB to 1.5 PiB, and we know of no layout that keeps most of it short. We charge for it in two ways. In both, an access to a memory of M bits costs M^γ , with $\gamma = \frac{1}{3}$ or $\gamma = \frac{1}{2}$.

- *The rule behind the security guide.* The Classic McEliece security guide [2] tabulates ISD with the estimator of [14], which multiplies the whole operation count by M^γ . The same rule applied to the run gives $T_f M^\gamma$.
- *A charge per access.* The rule treats every addition as an access to the whole memory, which is not what happens in a run. We therefore also charge every operation for the memory that it addresses. This is L^γ inside a tile of $L \approx 2^{22}$ bits, and M^γ for a bit that travels between a tile and the vectors. The total is $T_f L^\gamma + V_f M^\gamma$. At $\gamma = \frac{1}{2}$ the second term is 95% of it.

For $\gamma = \frac{1}{2}$ the second charge is the cost on a two-dimensional mesh, that is, a grid of small processors, each connected to its neighbours only. About $M^{1/2}$ bits can cross the middle of such a mesh in one step, and a constant fraction of the traffic has to cross it. So the run takes at least about $V_f/M^{1/2} = 2^{58}$ to 2^{64} steps, and $V_f M^{1/2}$ is the product of area and time [10]. It grows like $N^{5/2}$, as for the matrix step of factoring [6, 25]. We regard $T_f L^{1/2} + V_f M^{1/2}$ as the most meaningful single figure for the run. It sets the constant of the routing to 1. Terms that we can size add a little under two bits (Appendix E).

More hardware saves little time. The traffic does not depend on b , so a machine with four times the block width is four times as large, twice as fast, and twice as costly. A key search, or ISD with small lists, keeps its cost when it is spread over more hardware. At 10^9 steps per second and $b = 256$ such a mesh would need 7, 110, 420, 300 and 420 years for the five sets. No machine of this kind exists.

Comparison with ISD. Table 5 and Figure 1 set the run against ISD. In each of the three cost models that have a figure for both, the run is cheaper, by 41 to 59 bits for mceliece348864 and by 77 to 193 bits for the other sets.

There is no ISD figure with a charge per access. We therefore set our best estimate of the run, the charge per access at $\gamma = \frac{1}{2}$, against ISD with free memory, which favours ISD. The run is 44 bits cheaper for mceliece348864 and 77 to 170 bits cheaper for the other sets. This leaves room for what the two figures omit. The terms outside the estimate that we can size add under two bits, and the lowest ISD estimates of the guide are ten to eleven bits lower (Appendix E). With both, 32 bits

parameter set	bit operations		times $M^{1/3}$		times $M^{1/2}$		run, per access		other ISD		NIST
	run	ISD	run	ISD	run	ISD	$\gamma = \frac{1}{3}$	$\gamma = \frac{1}{2}$	2^{64}	Grover	level
mceliece348864	92.3	151	108.9	156.4	117.1	158.6	100.2	107.3	119	101	143
mceliece460896	98.3	191	115.8	198.6	124.5	201.0	106.5	114.0	159	124	207
mceliece6688128	101.3	257	119.2	274.9	128.1	278.2	109.6	117.2	225	164	272
mceliece6960119	100.5	257	118.3	275.8	127.2	279.2	108.8	116.4	225	164	272
mceliece8192128	101.3	287	119.2	311.9	128.1	315.6	109.6	117.2	255	183	272

Table 5: The single run, as a fixed circuit, against ISD. All entries are $\log_2$ costs. Bit operations: T_f , and the ISD figures of Table 1. Times M^γ : $T_f M^\gamma$ for the run, and the smallest entry of [2, Table 1] for ISD. That table comes from another estimator, whose smallest entries without a charge for memory are 140.8, 179.8, 246.2, 245.7 and 275.6. Per access: $T_f L^\gamma + V_f M^\gamma$. 2^{64} : ISD for one of 2^{64} ciphertexts, which is the first ISD column minus 32. Grover: logical quantum gates for Grover’s algorithm on Prange’s ISD. The last three columns carry no charge for memory.

remain for mceliece348864. The same estimate is 36 bits below the NIST level for mceliece348864 and 93 to 156 bits below it for the other sets.

Two things narrow the gap (Appendix E). Given 2^{64} ciphertexts for one key, ISD finds one plaintext up to 32 bits faster, whereas a run gains nothing from further targets. Grover’s algorithm speeds up ISD, whereas we know of no quantum speed-up for a run.

- For mceliece348864 the run stays 9 to 27 bits below ISD on 2^{64} ciphertexts in the same cost model. Against the quantum gate count of 2^{101} it is 9 bits cheaper without a charge for memory, and 6 to 16 bits more expensive with the square-root charges.
- For the four larger sets the run stays below both columns. The closest case is mceliece460896 against the quantum gate count. The run is 10 bits below it with the charge per access at $\gamma = \frac{1}{2}$. Under the rule of the guide the two are level: the run is 0.7 bits above, which is within the uncertainty of the model.

7 Experiments

Our implementation uses SageMath [35], M4RI [3] for dense linear algebra, and the solver of Section 6. This section reports the main tests, and Appendix D the others. The *toy keys* are two binary Goppa codes with $m = 6$, $t = 6$, $n = 64$, $k = 28$, not shortened.

Both attacks end to end at (5, 4). For the gradient attack on the toy keys we made 64 runs per key, one for each column, with $N = 98280$. All of about 64000 gradients lay in C'_τ . They spanned it at all columns, except at the six columns of the unique dual codeword of weight $6 = d + 1$ of one key, where the span was $d - 1$ dimensions short. Holding out two of these columns together repaired this. The finishing algebra returned a key that rebuilds the public code and decodes.

For the single-run attack we took $s' = 3$ and w at multiplicity 1. A constraint column is *regular* if it is not held out. All 32 regular columns were read correctly from $b = 200$ random column-kernel vectors, on both keys and for two choices of the unit columns each. One execution gave the key. The toy instances have 14 and 17 dual codewords of weight at most 8, so Assumption 3 does not apply to them. Equation (6) predicted the effect exactly. In three runs the row kernel had dimension exactly $\kappa + ms'$ plus the sum of (6) over the codewords, which is 118, 132 and 112. At each of

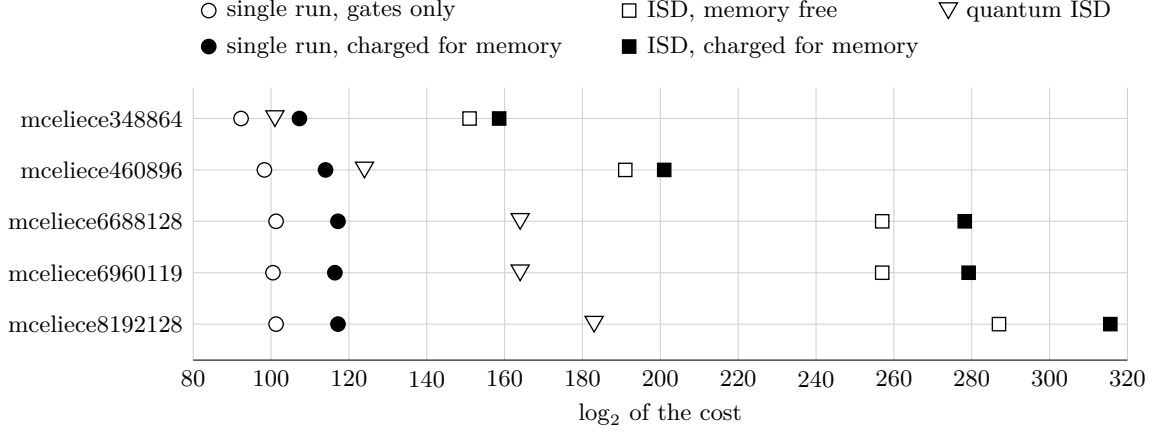


Figure 1: The main costs of Table 5 on a $\log_2$ scale. White marks count gates and leave memory free. Black marks charge for memory at square-root cost. For the run this is our best estimate, $T_f L^{1/2} + V_f M^{1/2}$, which charges every operation for the memory that it addresses. For ISD it is the estimate of the security guide, which charges every operation for the whole memory. Under that rule the run costs 2^{117} to 2^{128} . The ISD figures are for a single ciphertext, and the two come from different estimators. Quantum ISD is Grover’s algorithm on Prange’s ISD, in logical gates.

the 36 columns of a run the dimension in Step 2 was $(s' - 1)m = 12$ plus the sum of (6) over the codewords through the column. Step 5 then works modulo the additional relations.

Keys in general position at $(5, 4)$. The toy keys do not satisfy the hypothesis of Assumption 3. We therefore generated 30 keys with $m = 7$, $t = 11$, $n = 128$, without selection, and shortened them to $[n_\ell, k_\ell] = [119, 42]$, where a random code has 0.19 dual codewords of weight at most 8 on average. With $s' = 3$ and w at multiplicity 1, condition (5) holds with no column to lower, and $N = 850\,668$. All 30 keys were recovered, each from one run. On the 24 instances without such a codeword every dimension was as Assumption 3 states: $(s' - 1)m = 14$ at all 77 columns, $ms' = 21$ combinations and $m = 7$ solutions. All 73 regular columns were read correctly. The solver does not compute the rank of E . The dimensions of its blocks add up to the rank of a high power of B , and N minus this sum is the dimension of the row kernel plus the defect of Section 6. This *solver’s figure* was $\kappa + ms'$ plus 0 to 4. It tests the assumptions together, because a row kernel below the prediction and a defect of the same size would cancel.

The other six instances have one codeword of weight 8 each. For three of them (6) is 0, because the codeword passes through a held-out column, and these behaved like the 24. For the other three it is 14. There the dimension in Step 2 was 28 at the four or five non-unit columns of the codeword and 14 elsewhere, and the solver’s figure grew by 14. All 73 regular columns were again read correctly.

The threshold, lowered columns, and w at full multiplicity. These 30 instances lie away from the threshold, with κ at 14% of N , and they have no lowered column. The Classic McEliece parameters lie at the threshold. Four of the five sets have lowered columns, and three use w at multiplicity $d - 1$ (Table 4). We therefore shortened keys of the same kind to $[115, 38]$, the smallest k_ℓ with $\kappa > 0$, where $N = 501\,942$. Of 60 keys, five gave an instance without a dual codeword of weight at most 8. On each of the five we made two runs, with the solver in the form of Section 6. The first has w at multiplicity 1 and four lowered columns, and κ is 0.8% of N . The second has

w at multiplicity 4 and seven lowered columns, and κ is 2.7% of N . All ten runs gave the key. The dimension in Step 2 was 14 at all 77 columns, lowered or not, and all 73 regular columns were read correctly. The runs took 1951 to 1952 and 1913 iterations, as Assumption 4 predicts, and the solver’s figure was $\kappa + ms'$ plus 0 to 3.

A challenge instance at (6, 5). The TII challenges [24] are public keys of binary Goppa codes with $m \leq 10$, of interest only as tests of key-recovery algorithms [8]. Earlier work [29, 5, 11, 21, 32] solved instances with labels up to 252, and more than 140 remained open. We took label 253, with $m = 8$, $t = 9$, $n = 214$. Label 254 has since been solved [34]. The smallest feasible parameters are $(d, s) = (6, 5)$, $s' = 5$ and $k_\ell = 46$. The matrix has $9.4 \cdot 10^6$ rows and $1.18 \cdot 10^7$ columns, and the run took 35 903 iterations, or 21.7 hours on 32 threads. The reading went as predicted: dimension $(s' - 1)m = 32$ in Step 2, $ms' = 40$ combinations, $m = 8$ solutions, and all 68 regular columns read. The one exception is that seven columns had dimension 74 in Step 2. An exhaustive search shows that the instance has exactly one dual codeword of weight at most $2d - 2 = 10$. The seven columns are its non-unit columns, and $74 - 32 = 42 = \varepsilon(10)$. The organisers’ verification script accepts the resulting key (Appendix G). A second run with another choice of the unit columns behaved in the same way and agreed with the first on all 26 columns that they share.

Degree 7, the degree of the real attack. Keys that we can attack at $(7, 6)$ have $k_\ell \leq 37$, and a typical instance of this size has tens to hundreds of dual codewords of weight at most 12.

- *Few points.* We took three to five random points in general position at $k_\ell = 21$ to 24, with multiplicities 6 and 5. By dense linear algebra, the row kernel had dimension exactly κ , and the column kernel was exactly the span of the relations on one or two blocks.
- *A typical key* ($m = 6$, $t = 4$, $n = 64$, $k_\ell = 30$, 360 such codewords, $2.0 \cdot 10^6$ unknowns). We ran the gradient attack twice, with two forms of the solver. All gradients of the 241 and 256 kernel vectors lay in C'_τ and spanned it at both held-out columns. The excess of the row kernel over κ was 12 510, and (6) gives 12 542. For a single run, (6) predicts 270 or more additional dimensions at every column, more than the 256 kernel vectors. Indeed the vectors stayed independent in Step 2, and the support could not be read.
- *Keys in general position.* For $m = 6$, $t = 4$ and $n = 64$ the support fills the field, and the dual code has minimum distance 13 or 14. On 76 of 119 generated keys there are four coordinates such that the code shortened on them, of shape $[60, 36]$, has no dual codeword of weight at most 12. Such an instance has $8.3 \cdot 10^6$ unknowns and lies far from the threshold, with κ at 23% of N . The solver on $E^\top E$ ended after the predicted number of iterations, but its defect was 302, and nothing could be read from the vectors that it returned (Section 6). A run on $E^\top \Omega E$, with a random Ω and $\Psi = 1$, ended in the same way, because Ω does not touch the second part of the defect. With both Ω and Ψ the run took 25 117 iterations, as Assumption 4 predicts, and the attack recovered the key. Step 3 had $m = 6$ solutions, all 20 regular columns were read correctly, and the unit columns and the four shortened coordinates followed. Two dimensions were larger than in Assumption 3, both by $2m = 12$. Step 2 gave 36 instead of $(s' - 1)m = 24$ at all 24 columns, and Step 3 kept 42 combinations instead of $ms' = 30$. The solver’s figure was $\kappa + 42$ plus its defect of 2. These are the relations for small t of Section 5.2, which we found through this run. None of them vanished at a block modulo the other relations, so they did not add solutions in Step 3.

We then ran a second key of this kind, with Ω and Ψ . Its instance has w at multiplicity 6 and three lowered columns, and κ is 27% of N . Before the run ended, the model of Section 5.2

gave the same two dimensions for it, 36 and 42. The run took 23 728 iterations, as predicted, with a defect of 3. It showed these dimensions at all 24 columns, Step 3 had 6 solutions, all 20 regular columns were read correctly, and the attack recovered the key.

Assumption 1. For random points of multiplicity $d - 1$ near the threshold, the row kernel was never smaller than κ . Its excess over κ was the sum of (6), up to overlaps between codewords and the defect in the solver’s figure (Appendix D and Table 8).

Without a run. Table 7 lists what was checked at full Classic McEliece size (Appendix D). In particular, the finishing algebra of the gradient attack, with the subcodes C'_τ computed from the secret key, rebuilt a genuine mceliece348864 key and a genuine mceliece8192128 key in 40 and 25 minutes on one core.

8 Relation to the other key-recovery proposals

Besides the two algorithms in this paper, three key-recovery algorithms that build on the GIJS distinguisher have been described: the one added by GIJS themselves [20, Section 7], and two by Vedenev [40]. Saarinen [33] gives an itemised cost of the GIJS algorithm, and a note by Apon [4] analyses a step of Vedenev’s first algorithm. We refer to the versions whose dates are given in the bibliography. All algorithms start from the jet matrix of one shortened instance. They differ in three respects: which kernel of the matrix they read, what they read from it, and how they resolve the Frobenius ambiguity, that is, the m conjugate curves through the binary points. These differences are what separates their costs. Table 6 summarises them, and the rest of this section explains the entries.

	kernel of E	what is read	Frobenius branch fixed by	large solves per key
GIJS [20, §7]	column	canonical relation	feasibility tests at the 4th held-out column	$R_\ell(n_\ell+1)(q+1)$ over $\mathbb{F}_q$, $R_\ell = 1$ in [33]
Vedenev [40, §5]	row	deep derivative flags at a few columns	alignment or enumeration	one per column read
Vedenev [40, §6]	row	tangent spaces at every column	generalised minor code	as Section 4
Section 4	row	subcode C'_τ at every column	alignment system (App. B)	$[n_\ell/H] \approx 100\text{--}1400$
Section 5	column	canonical relation modulo K_{loc}	$m \times m$ eigenvalue problem	1
		$\log_2$ cost, mceliece348864	$\log_2$ cost, five sets	run on binary Goppa keys
GIJS [20, Table 3]		130.4	130–149	toy key (App. D)
Saarinen [33, Table 4]		126.8	127–145	parts, on small keys
Vedenev [40]		not given	not given	no ($\mathbb{F}_4$ -Goppa)
Section 4, (a) / (b)		120.8 / 100.8	121–131 / 101–110	toy key, finish at full size
Section 5, (a) / (b) / (c)		114.3 / 94.3 / 89.4	114–124 / 94–102 / 89–98	small keys, TII-253

Table 6: Key-recovery algorithms built on the GIJS distinguisher. Costs are $\log_2$ bit operations as reported by each source. The columns (a), (b), (c) are those of Table 1, where (c) charges one operation for a nonzero matrix entry and (a) and (b) charge two. All algorithms use $\mathbb{F}_2$ for the large solves, except that of GIJS, which uses $\mathbb{F}_q$.

GIJS [20, Section 7 and Appendix C]. This is the algorithm whose structural statement is Theorem 5. It fixes three held-out values. The held-out conditions then cut out an affine space of pairs (V_1, V_2) . For a column j and a candidate value x , one sparse linear solve over $\mathbb{F}_q$ decides whether this space contains a pair with $V_2 = x^{s'-1}V_1$ on the locator block of j . Local relations never have to be written down, because each test is a consistency question for the whole system. In exchange there are $(n_\ell + 1)(q + 1)$ solves for each of $R_\ell = 4$ or 5 shortened instances. Each solve is about twice the size of a distinguisher run and is over $\mathbb{F}_q$, which is charged a factor m^2 . Their Table 3, which already uses the sharper kernel count of Section 3, gives 2^{130} to 2^{149} . Their Remark 7.3 records the reductions proposed here, namely sampling the column kernel once, explicit local relations, and $\mathbb{F}_2$ throughout. The end-to-end binary Goppa test reported in [20, Appendix C.2] is the one of Appendix D.

Saarinen [33]. Saarinen estimates the cost of the algorithm of GIJS item by item. One shortened instance suffices in his analysis, because the completion step of [23] needs only $mt + 1$ support elements. He replaces the solver by a scalar Lanczos algorithm over an extension field, with a bound on the failure probability. His totals are 2^{127} to 2^{145} bit operations. His Section 7 contains the reduction of the canonical relations to finite rank conditions that we use in Theorem 5. He also reports small keys on which the feasibility test for a single column accepts many values. Wherever the reading of Section 5 applied, which works modulo the explicit local relations, it returned one value per column.

Vedenev [40] and Apon [4]. Vedenev reads the row kernel, as Section 4 does. At a held-out column the kernel forms determine the tangent line of the curve, which is the case of Theorem 4. Recursively they determine its osculating flag, that is, the spaces spanned by the higher derivatives of the curve, up to the Frobenius branch. His *minor-code attack* uses only the tangent spaces, but at every column. It builds an auxiliary code from 2×2 minors and, under a completeness conjecture, isolates a GRS component that is attacked as in [37]. It uses the same data as Section 4 and has the same dominant cost, namely the runs that are needed to hold out every column. The two differ in the algebra that follows. His *jet-consistency attack* uses deep flags at a few columns and guesses the support elements of these columns and their relative Frobenius twists. Both attacks were run end to end on Goppa codes over $\mathbb{F}_4$, and no figures per parameter set are given. Apon proves that reconstructing the curve uniquely from flags at c columns, a step of the jet-consistency attack, needs $c \geq 2t + 3$. His proof starts from the identity $\Pi f' + \Pi' f = \Gamma^2 f^2$ of Section 3. The two attacks of this paper do not contain such a step.

Comparison. Deep flags at a few columns need few runs, but they do not fix the global Frobenius labelling. First-order data at every column make the labelling a linear problem, but need $\lceil n_\ell/H \rceil$ runs. The column-kernel routes avoid both costs, because one canonical relation involves every column in one consistent branch. The many solves of GIJS come from not separating the canonical relation from K_{loc} . Proposition 6 makes this separation explicit.

9 Status of the assumptions, and consequences for parameters

Table 7 lists what each column of Table 1 rests on and how far each assumption was tested. What can be examined without a run was either proved or checked at Classic McEliece size.

figure	rests on	status
columns (b), (c)	Assumption 1	held near the threshold up to $(d, k_\ell) = (4, 64), (5, 44), (6, 30), (7, 26)$. Exact in general position, at $d = 7$ for up to eleven points
gradient attack	Assumption 2	held at $(5, 4)$, $k_\ell \leq 28$, and at the two columns of one run at $(7, 6)$, $k_\ell = 30$
single-run attack	dimensions of the finishing algebra good columns, $R_2 = \mathcal{D}_2$, unit columns, no dual codeword of weight ≤ 12 Assumption 3	held at full size checked at full size held in all 40 runs on keys in general position at $(5, 4)$, $k_\ell = 38$ and 42, ten of them at the threshold with lowered columns. At $d = 5$ the degree leaves no room for relations on three blocks. Their absence was tested at $(6, 5)$ on the challenge instance, and at $d = 7$ for up to five points. With dual codewords of low weight the deviations were as (6) predicts, at $(5, 4)$, $(6, 5)$ and $(7, 6)$. On two keys in general position at $(7, 6)$, far from the threshold, the attack succeeded. There the dimensions were larger by $2m$, an effect of $t = 4$ (Section 5.2)
column (c)	Assumption 4	on $E^\top E$, held on Goppa keys in all runs but one, up to $1.2 \cdot 10^7$ columns. It failed on a key in general position at $(7, 6)$, also with Ω alone. With Ω and Ψ , held in ten runs at $(5, 4)$ and two runs at $(7, 6)$, $k_\ell = 30$, and on two keys in general position at $(7, 6)$, one of them the key where the other forms failed. The defect at the parameters of the attack cannot be predicted
(a), (b), key recovery	the solver of [20] returns kernel vectors	not tested
(a), key recovery	Assumptions 2 and 3 at $(8, 7)$	not tested
cost of Step 2	a Krylov method on the local systems	not tested. We used dense elimination
memory-charged figures	the model of Section 6	estimates without an experiment

Table 7: What the figures of Table 1 rest on, and how far each ingredient was tested. The Classic McEliece sets have $k_\ell \geq 150$ and 2^{38} to 2^{42} unknowns.

If an assumption failed. Assumption 1 is the only one that enters the choice of parameters. Without it, a nonzero row kernel has to be guaranteed by (3), which together with Theorem 3 leaves only proved conditions. A run of the distinguisher, organised as in Section 6, then costs $2^{93.8}, 2^{112.6}, 2^{116.3}, 2^{115.4}, 2^{116.3}$ operations, which is 4 bits more for mceliece348864 and 17 to 18 bits more for the others. This still rests on Assumption 4. With the solver and the formulas of [20] the figures are $2^{97.9}, 2^{116.5}, 2^{120.1}, 2^{119.2}, 2^{120.1}$. The parameters of Table 4 lie at the threshold: κ is at most 0.5% of N . If the true threshold were a few units of k_ℓ higher, four of the five sets would lose about 0.1 bits per unit. For mceliece460896, $k_\ell = 195$ is the only value at which $\kappa > 0$ and (5) can both be met at $(7, 6)$. Otherwise its single run moves to $(8, 7)$ and costs $2^{105.7}$ operations.

If Assumption 4 failed, the run would fall back on the solver of [20], and its cost on column (b). We have no memory-charged figure for that solver, so Table 5 would have no counterpart. A failure of one of the other two assumptions would be visible from public data: a wrong dimension, no eigenvalue in $\mathbb{F}_q$, an inconsistent ratio, or a key that does not rebuild the code. A systematic failure of Assumption 3 would defeat the single-run reading, but not the gradient attack. Under

the assumptions, the single run fails only if a held-out column is not good or unsuitable for Step 4. Another column of multiplicity s' is then used instead, except for mceliece460896, which has no spare column and needs a second run with probability about 2^{-11} .

Dependence on the parameters. The cost of a run depends on m and t , and not on n as long as $k \geq k_\ell$. At fixed m it is polynomial in t . What raises the cost is m . The zero count forces s to grow with m , by about two units for every three units of m (Table 10). Each unit of s costs eleven to fourteen bits, so the cost grows by about eight bits per unit of m . It is therefore quasi-polynomial in n for families with $m = O(\log n)$, and not for families in which m grows faster. Nothing forces $n = 2^m$. Keep the t and the ISD cost of the present sets. Then a run costs more bit operations than the NIST level from $m = 19, 27$ and 34 on, for the count T and for the fixed circuit alike. The public keys are 2.5, 4.3 and 6.8 times the present sizes (Appendix F).

Acknowledgement. We thank Ashrujit Ghoshal, Yuval Ishai, Aayush Jain and Nuo Zhou Sun for sharing their key-recovery algorithm before publication, and for correspondence. Section 5 is built on the central observation of their algorithm. Their Remarks 5.1 and 7.3 record several of the improvements of Section 3 and the single-run reading of Section 5 as suggestions. We thank Markku-Juhani Saarinen for pointing out that an earlier version compared with superseded figures of his. We thank Daniel J. Bernstein for identifying an error in the first version's discussion of key sizes and for his questions about the operation counts (Appendix H), and Sanketh Menda for identifying errors in the cost figures of earlier versions.

AI Disclaimer. This work was generated by using Claude.

A Proofs

Proof of Proposition 1. Let $F_i, \bar{F}_i$ be as in (1) for the i -th codeword. Since F'_i is a square and $\Gamma^2 \mid F'_i$, we have $f_i = \bar{F}_i \sigma_i^2$ for a polynomial σ_i . Differentiating $\Pi = F_i \bar{F}_i$ gives $\Pi' \equiv F_i \bar{F}'_i \pmod{\Gamma}$, because $\Gamma \mid F'_i$. Multiply by $\prod_l \bar{F}_l$ and use $F_i \bar{F}_i = \Pi$:

$$\Pi' \prod_l \bar{F}_l \equiv \Pi \bar{F}'_i \prod_{l \neq i} \bar{F}_l \pmod{\Gamma} \quad \text{for each } i.$$

Summing over i gives $d \Pi' \prod_l \bar{F}_l \equiv \Pi (\prod_l \bar{F}_l)'$. Squares have zero derivative, so $h' = (\prod_i \sigma_i^2) (\prod_l \bar{F}_l)'$, while $h = (\prod_i \sigma_i^2) \prod_l \bar{F}_l$. Multiplying the last congruence by $\prod_i \sigma_i^2$ gives the identity.

The identity is $\mathbb{F}_q$ -linear in h , so it holds on all of R_d , that is, $R_d \subseteq \mathcal{D}_d$. It is one linear condition with values in $\mathbb{F}_q[Z]/(\Gamma)$, that is, t conditions over $\mathbb{F}_q$. These t conditions are linearly independent on the polynomials of degree at most dD_ℓ , because the map $h \mapsto (\Pi h' - d \Pi' h) \pmod{\Gamma}$ is onto. To see this, take $h = \Gamma u$ with $\deg u < t$, so that $\deg h \leq 2t - 1 \leq dD_\ell$. The last inequality holds because $D_\ell = k_\ell - 1 + (m - 2)t \geq t$. The value of the map is $\Pi \Gamma' u \pmod{\Gamma}$. This ranges over all residues as u does, since $\Pi \Gamma'$ is invertible modulo the square-free Γ . $\square$

Proof of Proposition 2. By interpolation, $f_c = \sum_j c_j \Pi / ((Z - \alpha_j) \Gamma(\alpha_j)^2)$. Expand $\Pi / (Z - \alpha_j)$ in powers of Z . The coefficient of $Z^{n_\ell - 1 - r}$ in f_c is then a fixed linear combination of the sums $\sum_j c_j \alpha_j^i / \Gamma(\alpha_j)^2$ with $i \leq r$, and the sum with $i = r$ has coefficient 1. These sums vanish for $i < 2t$, because they are the parity checks of $\text{Goppa}(\alpha, \Gamma^2) = \text{Goppa}(\alpha, \Gamma)$. This is why $\deg f_c \leq D_\ell$. For $r = 2t$ only $i = 2t$ survives, and

$$\sum_j c_j \alpha_j^{2t} / \Gamma(\alpha_j)^2 = \left(\sum_j c_j \alpha_j^t / \Gamma(\alpha_j) \right)^2 = \left(\sum_j c_j \right)^2.$$

The last step holds because α_j^t equals $\Gamma(\alpha_j)$ plus lower powers of α_j , and the parity checks of the lower powers vanish. For the last claim substitute $Z = 1/u$. Then $u^{dD_\ell} \varphi_h(1/u) = h(u^{D_\ell} p(1/u))$, and $u^{D_\ell} p(1/u)$ equals w at $u = 0$. $\square$

Theorem 3. The theorem follows from a more precise statement.

Proposition 9. Let W be the product of the factors $Z - \alpha_j$ over the j with $\mu_j + d$ odd, and let I_t be the number of monic irreducible polynomials of degree t over $\mathbb{F}_q$.

- (i) The polynomials in $\mathcal{D}_d$ that have a zero of order μ_j at every α_j and degree at most $dD_\ell - \mu_\infty$ are the products $u \prod_j (Z - \alpha_j)^{\mu_j}$ with $\deg u < \delta$ and $\Gamma \mid (Wu)'$.
- (ii) If $1 \leq \delta \leq t$ and $\deg W \geq \delta$, a nonzero polynomial as in (i) exists with probability at most $\frac{q^\delta - 1}{q - 1} \lfloor \frac{\deg W + \delta - 2}{2t} \rfloor \frac{1}{I_t}$, which is below $n_\ell q^{-(t-\delta+1)}$ for $q \geq 16$ and $t \geq 2$.

Proof. (i) A polynomial with these zeros and this degree is $\varphi = Qu$ with $Q = \prod_j (Z - \alpha_j)^{\mu_j}$ and $\deg u < \delta$. We have $(\Pi^d \varphi)' = \Pi^{d-1}(d\Pi' \varphi + \Pi \varphi')$, and Π is coprime to Γ . So $\varphi \in \mathcal{D}_d$ if and only if $\Gamma \mid (\Pi^d Qu)'$. Now $\Pi^d Q = \prod_j (Z - \alpha_j)^{d+\mu_j} = W\Theta^2$ for a polynomial Θ that is coprime to Γ . Squares have zero derivative, so $(W\Theta^2 u)' = \Theta^2 (Wu)'$.

(ii) Let $u \neq 0$ with $\deg u < \delta$. The polynomial Wu is not a square: W is square-free, so every root of W would have to be a root of u , and $\deg u < \deg W$. Hence $(Wu)' \neq 0$. In characteristic 2 a derivative is a square, so $(Wu)' = \sigma^2$ with $\deg \sigma \leq (\deg W + \delta - 2)/2$. An irreducible Γ divides σ^2 only if it divides σ , and σ has at most $\lfloor \deg \sigma / t \rfloor$ monic irreducible factors of degree t . The condition does not change when u is multiplied by a scalar, and there are $(q^\delta - 1)/(q - 1)$ classes of u . This gives the first bound. The second follows from $I_t \geq (q^t - 2q^{t/2})/t$, $\deg W \leq n_\ell$ and $2t < n_\ell$. $\square$

The orders have to be fixed before Γ is drawn, and a union bound covers an attacker who tries several choices. The heuristic value of the probability is $q^{-(t-\delta+1)}$, which agrees with our measurements.

Proof of Theorem 5. Univariate relations. Let $\mathcal{D}^+$ be the space of polynomials Φ of degree at most $dD_\ell + r$ with $\Pi \Phi' \equiv d\Pi' \Phi \pmod{\Gamma}$. As in Proposition 1, $\dim \mathcal{D}^+ = dD_\ell + r + 1 - t$. Consider the functionals $\Phi \mapsto \Phi^{[u]}(\alpha_j)$ for $u < s_j$, where $\Phi^{[u]}$ is the Hasse derivative, and $\Phi \mapsto$ (coefficient of $Z^{dD_\ell+r-u}$ in Φ) for $u < s_\infty$. By (5) there are $\dim \mathcal{D}^+ + 1$ of them. So they satisfy a nontrivial linear relation on $\mathcal{D}^+$, with coefficients $v(j, u)$ and $v(\infty, u)$. Put $v(j, u) = 0$ for $u \geq s_j$ and $v(\infty, u) = 0$ for $u \geq s_\infty$.

Since r is even, $(Z^r)' = 0$. Hence for $\varphi \in \mathcal{D}_d$ both φ and $Z^r \varphi$ lie in $\mathcal{D}^+$. From $Z^r = \beta_j + (Z - \alpha_j)^r$ we get $(Z^r \varphi)^{[u]}(\alpha_j) = \beta_j \varphi^{[u]}(\alpha_j) + \varphi^{[u-r]}(\alpha_j)$. Inserting φ and $Z^r \varphi$ into the relation gives two relations on $\mathcal{D}_d$ among the functionals $\varphi \mapsto \varphi^{[u]}(\alpha_j)$ and $\varphi \mapsto$ (coefficient of $Z^{dD_\ell-u}$). Their coefficients are

$$a(j, u) = v(j, u), \quad b(j, u) = \beta_j v(j, u) + v(j, u + r), \quad a(\infty, u) = v(\infty, u + r), \quad b(\infty, u) = v(\infty, u).$$

So $b = \beta_j a$ at the orders $u \geq s_j - r$ of column j . At a held-out column τ we have $s_\tau = r + 1$, so $b + \beta_\tau a$ has coefficient $v(\tau, r)$ at order 0 and coefficient 0 at the orders $1, \dots, r$.

Good columns. Let τ be good and suppose $v(\tau, r) = 0$. Then $b + \beta_\tau a$ is a relation on $\mathcal{D}_d$ among the functionals at the points other than τ . There are $\dim \mathcal{D}_d$ of these, and they are linearly independent because τ is good. So $(\beta_j + \beta_\tau)v(j, u) + v(j, u + r) = 0$ for all $j \neq \tau$ and all u . The map $x \mapsto x^r$ is a bijection, so $\beta_j \neq \beta_\tau$, and descending induction on u gives $v(j, u) = 0$. In the same way $v(\infty, u) = 0$. What is left of v is a relation on $\mathcal{D}^+$ among the derivatives of orders below

r at α_τ . But $\mathcal{D}^+$ contains $\Gamma^2 g$ for every g of degree at most $dD_\ell + r - 2t$, and $\Gamma(\alpha_\tau) \neq 0$, so these derivatives are independent. Hence $v = 0$, a contradiction.

Lifting. By the chain rule for Hasse derivatives and by homogeneity, for every form h of degree d we have $\varphi_h^{[u]}(\alpha_j) = \sum_{|S| \leq u} c_{j,u,S} \partial^{[S]} h(y_j)$. The coefficients $c_{j,u,S}$ depend on p and j only, and $c_{j,0,\emptyset} = \lambda_j^{-d}$. The same holds at w by Proposition 2. At a unit column all terms vanish for multilinear h . Put $V_1(j, S) = \sum_{u \geq |S|} a(j, u) c_{j,u,S}$, and define V_2 from b in the same way. Since $\varphi_h \in \mathcal{D}_d$, the relations a and b applied to φ_h show that V_1 and V_2 are column-kernel vectors of E . If $|S| \geq s_j - r$, then every $u \geq |S|$ has $b(j, u) = \beta_j a(j, u)$. This gives (i), and (ii) and (iii) follow from the values of $b + \beta_\tau a$ on the block of τ .

Probability. Apply Proposition 9(ii) with $\delta = t$. Its hypothesis $\deg W \geq t$ holds because the k_ℓ unit columns have order $d - 1$. Then use $\deg W \leq n_\ell$ and $I_t \geq (q^t - 2q^{t/2})/t$. $\square$

The ratio at a column j needs a nonzero entry of V_1 on the locator block. The argument for good columns gives a sufficient condition: the functionals at the points other than j , together with those of order below $s_j - 2r$ at j , are linearly independent on $\mathcal{D}_d$. For $s_j = s'$ this says that j is good. For $s_j = 6$ and $r = 4$ it says that the map $u \mapsto (Wu)'$ mod Γ of Proposition 9 has rank t on the polynomials of degree at most t .

Proof of Proposition 6. (i) is the Taylor expansion of a multilinear polynomial at a binary point. Over $\mathbb{F}_2$ we have $h(X + y) = \sum_S X^S \partial^{[S]} h(y)$. (ii) The operators Δ_a and $\Delta_{a'}$ commute, since both can be read off from $h(X + za + z'a')$. So the form in question equals $\Delta_{a'}^{(i')} (\Delta_a^{(i)} h)$. Its coefficients are sums of coefficients of $\Delta_a^{(i)} h$, and these lie in O_a by (i). With a and a' exchanged, the coefficients lie in $O_{a'}$ as well. Writing out the coefficient of X^C in $\Delta_{a'}^{(i')}$ of the polynomial $\Delta_a^{(i)} h$, whose coefficients are the conditions $\partial^{[i]} h(a)$ of order $d - i$, gives the formula. $\square$

Proof of Proposition 7. The formula is the definition of E . Let ω_j be the weight of y_j . A pair $\{i, i'\}$ inside the support of y_j occurs for the $\binom{k_\ell - 2}{d - 2}$ sets S that avoid it, and $\binom{k_\ell}{2} \binom{k_\ell - 2}{d - 2} = \binom{d}{2} N$. So the number of nonzero terms is $\binom{d}{2} N \sum_j \binom{\omega_j}{2} / \binom{k_\ell}{2}$, which is $\frac{1}{4} \binom{d}{2} P N$ for balanced columns. For the lower orders, let $a \geq 1$, let S be a set of size $d - 2^a$ and θ a bound. Let $F_a(S; \theta)_j$ be the sum of the $h_{S \cup U}$ over the sets U of size 2^a that are disjoint from S , contained in the support of y_j , and have $\min U > \theta$. The conditions of order $d - 2^a$ are $c_{j,S} = F_a(S; -\infty)_j$. If the pairs are processed in decreasing order of their smaller element, the $F_1(S; \theta)$ are partial sums of the sum for S . We claim that

$$F_{a+1}(S'; \theta)_j = \sum_V \left(\prod_{i \in V} y_j[i] \right) F_a(S' \cup V; \max V)_j,$$

where V runs over the sets of size 2^a that are disjoint from S' and have $\min V > \theta$. Indeed, a set U of size 2^{a+1} splits in exactly one way into the set V of its 2^a smallest elements and the set of the others, all of which exceed $\max V$. $\square$

So each tile sends $\binom{d-2}{2}$ masked vectors of P entries, a quarter of them nonzero, to the sets S' of size $d - 4$, and so on downwards. At $(7, 6)$ this adds $3.7N$ to $3.9N$ words to the $21N$ words of the main traffic. A point of multiplicity $d - 2$ has the top order $d - 3$, and its conditions $c_{j,S} = \sum_{v \notin S} y_j[v] F_1(S \cup \{v\}; v)_j$ are sums of partial sums as well. Generating the entries $y_j[i] y_j[i']$ adds 1.6% to the operations at $b = 256$, because one AND serves all bits of a word. The plain sparse product takes $7.4PN$ additions of words, against $5.25PN$ here.

B The finishing algebra

From the subcodes to the support (Section 4). Numbers in square brackets are the dimensions observed on a genuine mceliece348864 key with $k_\ell = 154$ and $n_\ell = 922$. Each equals the value of the formula next to it. Regard C'_τ as a space of messages. Its annihilator in $\mathbb{F}_2^{k_\ell}$ has dimension $m + 1$ and contains y_τ . Let V_τ be a complement of y_τ in it. By Theorem 4, $V_\tau \otimes \mathbb{F}_q$ is spanned modulo y_τ by the m conjugates of $\psi_\tau = \sum_{j \neq \tau} y_j / (\alpha_j - \alpha_\tau)$. Each conjugate spans a line. Looking at one column alone, one cannot tell which line belongs to which branch.

The lines are matched by one linear system. The unknowns are vectors $x_\tau \in V_\tau \otimes \mathbb{F}_q$, one for each column. For two messages g, g' with codewords c, c' , put $B_x(g, g') = \sum_\tau (c_\tau \langle x_\tau, g' \rangle + c'_\tau \langle x_\tau, g \rangle)$, and require $B_x = 0$ [11781 equations, 11064 unknowns]. If $x_\tau = \theta_\tau \psi_\tau$ for all τ , then $B_x(g, g') = \sum_\tau \theta_\tau \lambda_\tau^2 (f_c f_{c'})'(\alpha_\tau)$. By Proposition 1 with $d = 2$, this derivative is Γ^2 times the square of a polynomial of degree at most $D_\ell - 1 - t$. So $B_x = 0$ when θ lies in a GRS code of dimension $3t + 1$ whose support is a conjugate of α . This is the *pure family* of the branch. The solution space also contains solutions that mix branches, and its dimension is $m(3t + 1) + mt + m - 1$ [3095].

To separate one pure family we use three explicit identities, for $x_\tau = \alpha_\tau^e \psi_\tau$ with $e = 1, 2, 3$, whose forms B_x involve only w and the vectors $\sum_j \alpha_j^{2^u} y_j$. They lead to a small simultaneous-diagonalisation problem whose solutions are the m conjugate directions [$m = 12$ solutions]. Fixing one fixes a branch. Let X be the space of all x for which $B_x(g, g') = \sum_j \nu_j c_j c'_j$ for some scalars ν_j , enlarged by the two solutions from $e = 2, 3$ [dimension $3863 + 2$]. An element of X that vanishes on $5t + 2$ random columns is supported on the lines of the chosen branch. Two such elements determine that line at every column [all 922]. The pure family [$3t + 1 = 193$] is then an explicit GRS code, and the ratio of two of its codewords with $3t - 1$ common zeros is a Möbius image of the support.

From the support to a key. The support is known up to a Möbius map and a conjugation, which is enough for an equivalent key. The instance is the alternant code of degree $2t$ of the recovered support, that is, the binary part of a GRS code, for multipliers that are found by linear algebra [41]. The mt known non-unit columns fix the change of basis between this description and the public parity-check matrix, and applying it to the other public columns returns the support elements of the shortened-away coordinates [23].

The unit columns after a single run (Section 5.3). The β_j are the support after a Möbius map and a conjugation. Let $\tilde{\Gamma}$ be the Goppa polynomial in these coordinates, let β_∞ be the image of the point at infinity, and let $F = \prod_{c_j=1} (X - \beta_j)$ for a codeword c . If c has even weight, then $\tilde{\Gamma}^2 \mid F'$. If it has odd weight, the same holds for $(X - \beta_\infty)F$. The row of a unit column u in the systematic generator matrix is supported on u and on non-unit columns. Take a row of even weight, and let B be the product of the factors $X - \beta_j$ over its non-unit part. Then $((X - x)B)' = (XB)' + xB'$ is the square of $g_1 + \sqrt{x}g_2$ for two known polynomials, and for $x = \beta_u$ this polynomial is divisible by $\tilde{\Gamma}$. A second row of even weight, with polynomials h_1, h_2 , gives a test: modulo $\tilde{\Gamma}$ the quotient $\rho = h_1/h_2$ is a constant. So for every $x \in \mathbb{F}_q$ we compute the greatest common divisor of $g_1 + \sqrt{x}g_2$ and $\rho^q - \rho$ and remove its linear factors. For the right x what remains is $\tilde{\Gamma}$. For a wrong x an irreducible factor of degree $e > 1$ survives with probability $q^{-(e-1)}$. Once $\tilde{\Gamma}$ is known, the sum of the $1/(X - \beta_j)$ over the non-unit part of a row of even weight is congruent to $1/(X - \beta_u)$. For a row of odd weight it is congruent to $1/(X - \beta_u) + 1/(X - \beta_\infty)$, and β_∞ is the root that all odd rows share.

C General position

Derivation of (6). The points of a run are the columns of the instance, unit columns included, and w . A dependency between w and $c-1$ columns is a dual codeword of weight $n_\ell - c + 1$, because w is the sum of all columns. For a random code these are as rare as those of weight $c-1$. Let c points with multiplicities $\mu_1, \dots, \mu_c$ satisfy one linear dependency. The row kernel is the space of forms of degree d that vanish to the prescribed orders at all points. Its dimension does not change under a linear change of variables. So we may take $c-1$ of the points as the first $c-1$ unit vectors, and the last point is their sum. We complete the basis with $k_\ell - c + 1$ points of multiplicity $d-1$, so a form has degree at most 1 in each of the last variables. Write it as $\sum_T X^T F_T$, where T runs over the sets of last variables and F_T is a form of degree $j = d - |T|$ in the first $c-1$ variables. It vanishes to order μ at a point of the codeword if and only if every F_T vanishes there to order $\mu - (d - j)$. So the conditions of the c points act on each F_T separately. The series $\sum_j h_j z^j$ is that of (4) for the c points in $c-1$ variables, since $1 + z = (1 - z^2)/(1 - z)$. Formula (4) thus counts h_j admissible forms F_T of degree j , and h_j can be negative. The true number is $\max(0, h_j)$, which gives (6). In characteristic 0 this is a theorem. By [16] the admissible forms correspond to the degree- j part of the polynomial ring in $c-1$ variables modulo the powers $L_i^{d-\mu_i+1}$ of c linear forms in general position. Its dimension is given by [38]. In characteristic 2 we verified it by computation for all $j \leq 7$ and $c \leq 13$, with $c-1$ points of multiplicity $d-1$ or $d-2$ and one point of any multiplicity (1771 cases).

If all points have multiplicity $d-1$, then $h_j = \binom{c-1}{j} - \binom{c-1}{j-2}$, and all terms vanish for $c \geq 2d-1$. Lower multiplicities weaken the effect. At $d=7$ a codeword of weight 12 removes $\varepsilon(12) = 132$ conditions if all its points have multiplicity 6, and none if one of them has multiplicity 5. If $c-1$ of the points are unit columns, the additional relations are internal to the block of the last point. That the contributions of several codewords add up, in the presence of the other points, is a genericity statement like Assumption 1.

Tests. We planted a dependency of size c among one to three columns and unit columns, at $k_\ell = 20$, for $d = 5, 6, 7$ and all c from $d+1$ to $2d-1$. The columns had the multiplicities $(d-1)$, $(d-2)$, $(d-1, d-2)$, $(d-2, d-2)$, $(d-1, d-2, d-2)$ or $(d-2, d-2, d-2)$. In all 90 cases the row kernel had dimension exactly κ plus (6). Table 8 compares the sum of (6) with the measured excess on whole instances. With $\kappa = 0$ and few codewords the kernel was zero. With about a hundred codewords, at $(6, 26)$ and $(7, 24)$, a nonzero kernel can remain beyond the threshold, which agrees with Assumption 1.

D Further experiments

Several tests work in the *univariate model*: they build R_d from the secret key and test a statement about polynomials directly, without forming E .

- *Propositions 1 and 2.* On 10 codes with $d = 2, \dots, 5$ and $m = 4, \dots, 8$, shortened and not, the identity held on all products, $R_d = \mathcal{D}_d$, and the leading coefficient equalled the parity for every row.
- *Theorem 3.* For 400 random keys at each of 17 combinations of $(m, t, n_\ell, d, \delta)$ with $m \leq 6$ and $t-2 \leq \delta \leq t$, the two descriptions in Proposition 9(i) agreed, and a nonzero polynomial existed with a frequency close to $q^{-(t-\delta+1)}$ and below the bound. The binary distinguisher, on keys with $m = 6, 7$ and $t = 3, \dots, 6$, accepted Goppa keys up to shortfall t and rejected random matrices. From shortfall $t+2$ on it rejected Goppa keys too.

instance	d	$[n_\ell, k_\ell]$	codewords	sum of ε	measured excess
random points*	4	[108, 32]	0, 0, 0	0, 0, 0	0, 0, 0
random points	4	[234, 48]	0, 0, 0	0, 0, 0	3, 2, 1
random points*	5	[61, 28]	14, 12, 14	196, 168, 196	196, 168, 196
random points	5	[96, 36]	0, 0, 1	0, 0, 14	3, 1, 16
random points	6	[43, 26]	36, 31, 33	2520, 1554, 1890	2508, 1545, 1878
random points	6	[55, 30]	31, 36, 29	1918, 1820, 1218	1923, 1820, 1216
random points	7	[32, 24]	23, 25, 24	5976, 8424, 3756	5853, 8032, 3721
random points	7	[36, 26]	31, 28, 31	7452, 5712, 7452	7334, 5656, 7333
chosen points	7	[31, 24]	0, 0, 0	0, 0, 0	0, 1, 2
chosen points	7	[35, 26]	0, 0, 0	0, 0, 0	1, 2, 4
chosen points	7	[39, 28]	0	0	3
toy keys, single runs*	5	[64, 28]	17, 17, 14	118, 132, 112	118, 132, 112
$m = 7, t = 11$, 27 single runs	5	[119, 42]	0 or 1	0	0 to 4
$m = 7, t = 11$, 3 single runs	5	[119, 42]	1, 1, 1	14, 14, 14	15, 14, 17
$m = 7, t = 11$, 10 single runs at the threshold	5	[115, 38]	0	0	0 to 3
TII-253	6	[118, 46]	1	42	42
$m = 6, t = 4$, gradient run	7	[54, 30]	360	12 542	12 510
$m = 6, t = 4$, single run	7	[54, 30]	360	12 581	12 559
$m = 6, t = 4$, single runs	7	[60, 36]	0, 0	0, 0	14, 15

Table 8: Dual codewords of weight at most $2d - 2$ and their effect. Upper part: three instances of random points each, at the smallest of five values of P around the threshold, excess of the row kernel over κ . The chosen points are those of Appendix E, away from the threshold. Lower part: Goppa keys, with the multiplicities of the run. The excess is over κ , and over $\kappa + ms'$ in a single run. In the single runs at $t = 4$ it includes the 12 further relations of Section 5.2. For TII-253 it is the surplus of the dimension in Step 2. Rows with an asterisk were computed by dense linear algebra. The others are the figure of the solver, which can be a few units too high.

- *Assumption 1.* We measured the row kernel for random points of multiplicity $d - 1$, at the five values of P around the threshold and on three instances each. We used dense linear algebra at $(d, k_\ell) = (4, 32)$ and $(5, 28)$, and the solver up to $(4, 64)$, $(5, 44)$, $(6, 30)$ and $(7, 26)$. The dimension was never below κ . The solver reports the number of dimensions that the iteration covered. On the 30 instances for which we have both figures, this fell short of the rank of E by 0 to 8. Instances without dual codewords of weight at most $2d - 2$ gave κ plus 0 to 9. With codewords the excess was the sum of (6). This was exact in 17 of the 18 instances with $\kappa > 0$ that we computed by dense linear algebra. Where twenty or more codewords overlap, the excess was up to 8% less. On the instances with points of weight $k_\ell/2$ of Appendix E it was up to 4% less at $k_\ell = 24$. It was up to 22% less at $k_\ell = 20$ and 22, where every point forms a codeword with the unit columns of its support.
- *Gradient attack, variants.* Three held-out columns per run together with w , which takes 22 runs instead of 64. A shortened instance ($m = 6, t = 5, k = 34, k_\ell = 27$), including the completion. A second, independent implementation repeated the gradient checks on six further keys. The finishing algebra gave all predicted dimensions at eleven toy sizes with $m = 6, \dots, 10$ and $t = 5, \dots, 10$.
- *Theorem 5, univariate model.* On keys with (m, t) from $(6, 4)$ to $(7, 10)$ and $s' = 3, 5$, the relations were unique and had the stated form, and the ratio accepted the true support element and no other value at every column. With one condition too few there was no relation. With t conditions too many, every field element passed at all columns but one. With $s' = 2$ the

relations admitted no common pair (V_1, V_2) . We also followed the proof step by step at $(d, s) = (7, 6)$ and $s' = 5$ on ten keys.

- *Relations on the gradient along the curve* (Section 5.2). For a multilinear form h put $g_i(Z) = \partial_i h(p(Z))$. The conditions at column j determine the coefficients of φ_h at α_j up to order $s_j - 1$, and those of every g_i up to order $s_j - 2$. We evaluated these functionals on random monomials, computed all linear relations among them, and discarded the relations on one or two blocks. Let x be the left side of (5) minus $\dim \mathcal{D}_d$, so that $x = s'$ if (5) holds. We took 108 instances with $t = 3, \dots, 8$, $m = 6, 7$ and $d = 5, \dots, 8$, with w among the points. For $x < t$ exactly x relations remained, which are those of Theorem 5. From $x = t$ on for odd t , and from $x = t + 1$ on for even t , there were more. At $t = 4$ the numbers for $x = 4, \dots, 8$ were 4, 7, 10, 29, 48, the same at every degree and on every key. For both keys in general position at degree 7 (Section 7) the model gives 7 per branch, the number that the runs showed. For the second key we computed it before the run ended. Random points do not have such relations. On six instances at $d = 7$ and $k_\ell = 24$ to 28 that include the sum of all columns as a point of multiplicity 3, the solver's figure was κ plus (6) plus 1 to 3.
- *The test of GIJS*. On the toy matrix we ran the test of [20, Section 7] exhaustively over $\mathbb{F}_{64}$. With three held-out columns imposed, the values accepted at each other constraint column were exactly the m conjugates of the true value. With the fourth column imposed, every column accepted the true value alone. This is the test reported in [20, Appendix C.2].
- *Propositions 6 and 7*. On the toy matrix, $\dim O_a \cap O_b$ is $28 = k$ for two regular columns, apart from a few pairs on a common dual codeword, where it is $28 + \varepsilon(8) = 42$. The vectors of Proposition 6 spanned the intersection for random columns at $(d; s_a, s_b) = (6; 5, 5)$, $(7; 6, 6)$ and $(7; 6, 5)$ with $k_\ell = 22$. The products with E and $E^\top$, implemented tile by tile, agreed with the definition of E for $d = 5, \dots, 10$, $k_\ell = 10, \dots, 14$ and random points, and so did the counts of words and additions.
- *Full size*. The instances of Table 4 have four shapes (m, t, k_ℓ) . On the genuine mceliece348864 and mceliece8192128 keys every dimension of Appendix B came out as predicted, and the recovered keys rebuilt the public codes on all coordinates. On generated keys of each shape: the average number of nonzero entries per row and point was 7.39 to 7.52, where balanced columns give 7.44. $R_2 = \mathcal{D}_2$ held on all 20 keys. A held-out column was not good in 19 of 100 000 random choices for $m = 12$, where $1/q$ predicts 24. For the three shapes with $m = 13$ it was not good in 12, 5 and 11 of 80 000 choices, where $1/q$ predicts 10. For the unit columns we took the β_j of the non-unit columns from the secret key, after a random Möbius map. The first guess that passed the test was the right one, after 473 to 5691 guesses and 2 to 28 minutes on ten cores. A search with Stern's algorithm, with enough iterations to find any given dual codeword of weight at most 12 with probability 0.99, found none on six instances, and it found a planted one.
- *TII-253*. All 72 non-unit columns have multiplicity 5, w has multiplicity 1, and (5) holds with no column to lower. Here $\kappa = 203\,318$, and E has $3.1 \cdot 10^9$ nonzero entries. The run is 2^{57} bit operations in the count of Section 6, and its block vectors take 3 GB. Step 2 took less than two hours. The large runs used a shared server with a 64-thread Intel Xeon Platinum 8488C and 247 GB of memory.

E The cost model: omitted terms and comparisons

The counts of Table 4. An iteration consists of one product with E , one with $E^\top$, and seven dense operations on block vectors. The solver holds seven block vectors of length J , the number of columns of E , and one of length N . With P points, $I = N/(b - 0.76)$ iterations, and $\Lambda = \frac{1}{4} \binom{d-2}{2} P \binom{k_\ell}{d-2} + \dots$ words that the lower orders move,

$$\begin{aligned} T &= I \left[b \left(2 \cdot \frac{1}{4} \binom{d}{2} PN + 2\Lambda + \binom{d}{2} N \right) + 2 \binom{d}{2} PN + 3.5 Jb^2 \right], & V &= I b \left[2 \binom{d}{2} N + 2\Lambda \right], \\ M &= (7J + N) b, & L &= \left[\binom{k_\ell - d + 2}{2} + P \right] b. \end{aligned}$$

All points are charged as if they had multiplicity $d - 1$, and each of the seven dense operations of an iteration is charged $b/2$ additions of words per coordinate. The terms of T are, in this order: the additions of words in the tiles for the two products, the additions of the lower orders, the additions at the position of each entry of the vector after the exchange, the ANDs that compute the entries of E , and the dense operations. An entry of E costs one AND for all b bits of a word, whether it is zero or not. The masks of the lower orders are fewer than the entries of E by a factor of 2^{10} to 2^{11} , and T omits their computation.

The multiplicity of w is 1 for mceliece348864 and mceliece6960119 and 6 for the other sets.

The fixed circuit performs a masked addition, which is two gates per bit, for every entry of E , zero or not, and for every bit of the small matrices in the dense operations. It also sends all $\Lambda_f = \binom{d-2}{2} P \binom{k_\ell}{d-2} + \dots$ words of the lower orders, which is 4Λ at $d = 7$. So

$$T_f = I \left[b \left(4 \binom{d}{2} PN + 4\Lambda_f + \binom{d}{2} N \right) + 14 Jb^2 \right], \quad V_f = I b \left[2 \binom{d}{2} N + 2\Lambda_f \right].$$

The mask bits $y_j[i] y_j[i']$ depend neither on S nor on the iteration. In a count of gates they are computed once, by $P \binom{k_\ell}{2} < 2^{26}$ gates, and T_f omits them. If every tile computes them anew for every product, as in T , then T_f grows by $2 \binom{d}{2} PN$ gates in each iteration. This is 0.2%, or 0.003 bits. The circuit has the number of iterations of Assumption 4 built in. In each iteration the solver also selects columns and inverts a $b \times b$ matrix. This takes on the order of b^3 gates, which is negligible beside Jb^2 .

The defect. Let $\mathcal{C}$ and $\mathcal{R}$ be the column space and the row space of E , and let r be its rank. Their orthogonal complements are the row kernel and the column kernel.

- *First part.* We have $Bz = 0$ exactly when $\Omega \tilde{E} z \in \mathcal{C}^\perp$. So $r - \text{rank}(B) = \dim(\mathcal{C} \cap \Omega \mathcal{C}^\perp) = r - \text{rank}(E^\top \Omega E)$.
- *Second part.* The iteration builds a space on which the form $z^\top B z'$ is nondegenerate. This space lies in $\text{Im } B$ and meets $\ker B$ only in 0. So it misses $\text{Im } B \cap \ker B$, which has dimension $\text{rank}(B) - \text{rank}(B^2)$. Suppose that the first part is 0. Then $\text{Im } B$ is the row space $\Psi^\top \mathcal{R}$ of $\tilde{E}$, and $\ker B$ is its column kernel $\Psi^{-1} \mathcal{R}^\perp$. Multiplication by Ψ maps their intersection to $\Omega' \mathcal{R} \cap \mathcal{R}^\perp$, which has dimension $r - \text{rank}(E \Omega' E^\top)$. This number does not depend on Ω .

In our computations higher powers of B lowered the rank by a few more units, at most 10. If the defect exceeds b , the only combinations that $\tilde{E}$ annihilates are combinations of the last block W . They lie in $\text{Im } B \cap \ker B$ and are not random. In our runs nothing could be read from them.

For $\Omega = 1$ and $\Psi = 1$ the two parts are the parts of $\mathcal{C}$ and of $\mathcal{R}$ that are orthogonal to the whole space, and the structure of E makes both large. If Ω and Ω' behaved like random symmetric matrices, each part would be at most 3 with probability 0.998. An involution of the coordinates is less than that. It preserves the weight of a vector, so it can map a sparse vector of $\mathcal{C}^\perp$ to a sparse

vector of $\mathcal{C}$. Both spaces have many sparse vectors when there are few points or when the points have low weight. The involution Ω fixes half of the coordinates, because without fixed points $z^\top Bz$ would be 0 for every z . The involution Ω' needs fixed coordinates for the construction of Ψ .

The matrix Ψ is built as follows. Attach to each pair (a, a') that Ω' exchanges a coordinate a'' that it fixes, using each coordinate once. On the triple (a, a', a'') the matrix Ψ has the rows $(1, 1, 0)$, $(0, 1, 1)$ and $(1, 1, 1)$. On the other coordinates it is the identity. Then $\Psi\Psi^\top = \Omega'$.

We computed the defect by dense linear algebra for random points of multiplicity $d - 1$, on three samples at each size (Table 9). At the three sizes of the table, κ reaches 0 with about 16, eight and five points.

(d, k_ℓ)	B	number of points							
		1	2	3	4	5	6	7	8
(5, 20)	$E^\top E$	580	980	1106	1256	493	61	12	33
	$E^\top \Omega E$	29	30	25	28	27	21	8	7
	$\tilde{E}^\top \Omega \tilde{E}$	15	5	6	5	4	3	4	2
(6, 18)	$E^\top E$	1548	2131	2070	2086	575	216	154	235
	$E^\top \Omega E$	113	173	175	114	55	30	21	15
	$\tilde{E}^\top \Omega \tilde{E}$	50	9	54	5	2	4	2	6
(7, 18)	$E^\top E$	4116	5044	6394	5840	3393			
	$E^\top \Omega E$	414	929	1276	2036	1019			
	$\tilde{E}^\top \Omega \tilde{E}$	227	68	43	30	15			
(7, 18), balanced	$E^\top E$	3620	4926	7220	9326	2629			
	$E^\top \Omega E$	158	2245	2048	1077	470			
	$\tilde{E}^\top \Omega \tilde{E}$	27	13	3	1	8			

Table 9: The defect, that is $\text{rank}(E)$ minus the rank of a high power of B , for random points of multiplicity $d - 1$. Each entry is the largest value in three samples. In the last three rows all points have weight $k_\ell/2$. At (5, 20) with nine to 16 points the values are at most 15, 10 and 6.

On $E^\top E$ the defect is in the hundreds or thousands when there are few points. At degree 5 it becomes small well before the threshold. At degrees 6 and 7 it is still in the hundreds or thousands there. With Ω alone what remains is essentially $r - \text{rank}(EE^\top)$, as the second item predicts. At degree 5 this is at most 30, and Ω alone was enough in our runs at that degree. At degree 7 it is still in the hundreds or thousands.

With Ω and Ψ the defect differed from the sum of the two parts by at most 11. It was at most 8 from three points on, with two exceptions. Both are samples with points of low weight. At (7, 18) one sample has five points of weights 8, 8, 6, 8, 5, and its defect fell from 227 to 15 as the points were added. At (6, 18) one sample has a point of weight 3, which accounts for the value 54. At (7, 18) no sample is in general position, because a point of weight at most 11 and the unit columns of its support form a dual codeword of weight at most $2d - 2$. In one sample at $k_\ell = 16$ the two parts reached 120 and 26. The columns of a Classic McEliece instance have weight $k_\ell/2$ up to a few per cent of k_ℓ , and the row kernel consists of polynomials that vanish to order six at hundreds of points. We expect no sparse vectors there, but we cannot prove it.

The solver agrees with these computations. We ran its three forms on small instances at degree 7.

Random points of weight $k_\ell/2$ and multiplicity 6. We took two to eight points at $k_\ell = 20$, three to ten at $k_\ell = 22$ and four to twelve at $k_\ell = 24$. Each range ends beyond the threshold. These 24 instances have between 2 and 137 dual codewords of weight at most 12. With Ω and Ψ the defect was at most 4 on all of them. With Ω alone it was 256 or more on eight instances and reached

1484. On $E^\top E$ it was 256 or more on 15 instances and reached 56 608. At $k_\ell = 24$ the defect on $E^\top E$ was 13 968, 189 and 83 with four, five and six points. From seven points on it was at most 3 in all three forms.

Points in general position. We chose points such that the instance has no dual codeword of weight at most 12: seven points at $k_\ell = 24$, nine at $k_\ell = 26$ and eleven at $k_\ell = 28$. A greedy search found no larger sets. Here κ is 24% to 26% of N , close to the 23% of the key at [60, 36] of Section 7. With all points at multiplicity 6 the solver's figure was κ plus 0 to 4 in all three forms, on seven instances. With only the first five of seven points at $k_\ell = 24$ it was κ plus 1 to 7. With the first three, where κ is 65% of N , the defect was 67 774 on $E^\top E$, 1508 with Ω alone and 3 with Ω and Ψ .

The defect is also large when some points have a lower multiplicity, like the held-out and the lowered columns of a single run. On eight instances at $k_\ell = 24$, 26 and 28 we lowered two or four of the points to multiplicity 5, and on two of them one more point to 3. Then κ is 39% to 54% of N . On $E^\top E$ the defect was 349 to 3541. With Ω alone it was 350 to 3500, and the solver found 254 to 256 of its 512 candidates, as on the key. With Ω and Ψ the defect was 0 to 2, and the row kernel had dimension exactly κ on all eight. Here Ω alone changes little, so it is the second part of the defect that is large. Ψ removes it at these sizes.

Degree 5. On instances with two and four points at $k_\ell = 28$, the solver returned nothing on $E^\top E$ and 256 kernel vectors with Ω alone. With Ω alone we also repeated three of the runs at $k_\ell = 42$ and all ten runs at $k_\ell = 38$ of Section 7, and each gave the key. Proposition 8 also holds for the row kernel, with the same proof. In the two runs for the row kernel at degree 7 its hypothesis held for 241 of the 256 vectors with $B = EE^\top$, where the other 15 were zero, and for all 256 with both involutions.

What the memory charge leaves out. The figure $V_f M^{1/2}$ of Section 6 sets the constant factor of the routing to 1. The following terms can be sized. Together they come to about 1.7 bits.

- Every iteration sums $b \times b$ matrices over all coordinates and distributes the result. The sum and the distribution each take about $M^{1/2}$ steps, and the iteration waits for them. The exchange of one iteration takes 6.2 to 6.3 times $M^{1/2}$ steps. So time and cost grow by 32%, or 0.4 bits.
- The pattern of the traffic is fixed, but a mesh has to route it. If every word of b bits carries the $\log_2 N$ bits of its destination, the traffic grows by 0.2 bits.
- The products with Ψ and $\Psi^\top$ move about $1.5J$ words in every iteration. This is 3% of the traffic, or about 0.05 bits. Their operations are negligible.
- Every tile holds the Pk_ℓ bits of the instance. These copies come to 8% to 11% of M .
- The words in flight during one product amount to 3.1 to 3.2 times M . The exchange has to proceed in phases, or it needs buffers of this size, which would add one bit.

An undetected hardware error invalidates all later iterations, so a run of this length also needs checks and checkpoints.

ISD on many targets and on a quantum computer. Given $\mathcal{T}$ ciphertexts for one public key, ISD finds one plaintext faster by a factor of at most about $\sqrt{\mathcal{T}}$ [36], while one recovered key decrypts all of them. Table 5 takes $\mathcal{T} = 2^{64}$, the example of [2, Section 6.1]. On a quantum computer, Grover's algorithm takes the square root of the number of iterations of Prange's ISD [30, 7]. The table lists $\frac{\pi}{4} p^{-1/2} (n - k)^2 n / 2$ gates, where p is the probability that a random set of k coordinates is free of errors. This ignores error correction, memory and the depth of at least 2^{71} to 2^{150} .

For the four larger sets, the run with the charge per access at $\gamma = \frac{1}{2}$ stays below multi-target ISD without a charge for every $\mathcal{T}$ up to 2^{154} . The first ISD column of Table 5 is not the lowest estimate in the literature. The smallest entries of [2, Table 1] without a charge for memory are ten to eleven bits below it. They come from an estimator that counts an operation on a vector as one operation. With them the margin of 27 bits for mceliece348864 on 2^{64} ciphertexts becomes 17, and 2^{154} becomes 2^{132} .

F Other parameters

The cost of a run is determined by the smallest k_ℓ with $\kappa > 0$ at the smallest feasible (d, s) (Table 10). For Table 11 we keep the t of mceliece348864, mceliece460896 and mceliece6688128. We raise m until a run costs more bit operations than the NIST level of the set. We then raise n until a Stern-type estimate of ISD is back at its value for that set. This happens at $m = 19, 27$ and 34 , with public keys of 0.65, 2.2 and 7.1 MB. With the square-root charge, $m = 16, 22$ and 29 suffice, with keys of 0.47, 1.5 and 5.1 MB. All counts are those of T and V , which skip zeros and are the lower ones. These figures illustrate the scaling. They are not proposals, because they extrapolate Assumptions 1 and 4 far beyond our tests, and we did not examine other attacks at large m .

m	$t = 64$	$t = 96$	$t = 128$	$t = 160$
12	89.4 (7,6)	93.6 (7,6)	96.5 (7,6)	98.9 (7,6)
13	101.1 (8,7)	95.4 (7,6)	98.4 (7,6)	100.8 (7,6)
14	103.0 (8,7)	107.7 (8,7)	111.1 (8,7)	113.7 (8,7)
16	126.2 (10,9)	121.7 (9,8)	125.5 (9,8)	128.4 (9,8)
18	139.8 (11,10)	145.9 (11,10)	139.9 (10,9)	143.1 (10,9)
20	153.4 (12,11)	160.0 (12,11)	154.2 (11,10)	157.7 (11,10)
24	190.1 (15,14)	188.1 (14,13)	193.6 (14,13)	198.0 (14,13)
28	217.3 (17,16)	226.3 (17,16)	222.3 (16,15)	227.2 (16,15)
32	253.9 (20,19)	254.2 (19,18)	261.6 (19,18)	256.4 (18,17)
34	267.3 (21,20)	268.2 (20,19)	275.9 (20,19)	270.9 (19,18)

Table 10: $\log_2$ of the number of bit operations T of a run of the distinguisher, and the (d, s) that achieves it, in the organisation of Section 6. In the rows shown, the single run of the key recovery costs the same to within a quarter of a bit. The cost is not monotone in t , because a larger t relaxes the zero count and can make a smaller (d, s) feasible.

G The recovered key of TII-253

Public key: file pk_McEliece_253.txt of [24], with SHA-256

a66ac1368bd6d311838e3a255c2d12766818731739c53d1ec79a46c8838ee793.

Parameters: $m = 8$, $t = 9$, $n = 214$, $k = 142$, field $\mathbb{F}_{2^8} = \mathbb{F}_2[a]/(a^8 + a^4 + a^3 + a^2 + 1)$. Below, a field element is written as the byte whose bit i is the coefficient of a^i . The public code equals Goppa(L, g) for the monic irreducible

$$g = x^9 + \mathbf{a6} x^8 + \mathbf{94} x^7 + \mathbf{4a} x^6 + \mathbf{36} x^5 + \mathbf{82} x^4 + \mathbf{72} x^3 + \mathbf{38} x^2 + \mathbf{d7} x + \mathbf{be}$$

and the support L (one byte per coordinate, in the column order of the public key, 32 per line):

set	(m, n, t)	public key ciphertext		(d, s)	$\log_2 N$	$\log_2$ cost of a run		ISD (Stern)
		(bytes)	(bytes)			T	$TL^{1/2} + VM^{1/2}$	
mceliece348864	(12, 3488, 64)	261 120	96	(7, 6)	38.1	89.4	106.7	151.4
mceliece460896	(13, 4608, 96)	524 160	156	(7, 6)	40.8	95.4	113.4	193.3
mceliece6688128	(13, 6688, 128)	1 044 992	208	(7, 6)	42.1	98.4	116.6	268.0
t and ISD of 348864	(19, 5504, 64)	651 776	152	(12, 11)	68.0	151.4	183.6	151.8
t and ISD of 460896	(27, 9536, 96)	2 249 856	324	(16, 15)	98.4	214.2	260.8	193.9
t and ISD of 6688128	(34, 17344, 128)	7 067 648	544	(20, 19)	128.6	275.9	337.2	268.6

Table 11: Examples with $n < 2^m$. Sizes are $mt(n - mt)/8$ and $mt/8$ bytes. The costs use the counts T and V , which skip zeros. The last column is the $\log_2$ bit complexity of Stern’s algorithm as estimated by the CryptographicEstimators library [17], version 2.1.1. It serves only to compare rows with each other. In the last three rows n is the smallest multiple of 64 at which the estimate reaches that of the set named.

```

2256e1ae20345840a00e4ede95fd1bf97fd0f3e970211367146f510169d9c890
f7c4f65ebb97b5db2a80386052baf4c2d12fd7746278cb9d025b23c679720dec
63cf9bc5b3b05a0766aa067d4c5d541e827be3a705753d4b4d88cef2fe177c29
a2b8a19a3e819c6ebd12fac75f761dbc4864100c4fe48bc341b2b41f683fa647
31d56de6aff10bf50a7133c0ac2cdf308659fca58796a33cc937d6ad8d49fbb1
4a3b65b7dceea4b639361ab918da0477dd4332cd83cc8f57ed2b7325e2f82755
94ef351528e798619e2442ea84a9c1eb2e998593260f

```

In the two-line text format of the challenge (coefficients of g from low to high degree, then L , as polynomials in a) this key is accepted by the organisers’ script `check_solution_track2.sage`.

H Revision history

The dates are those of the versions of this paper in the Cryptology ePrint Archive, Paper 2026/1984. The first three versions carried no version number.

- *Version 0.1, 11 September 2026.* First public version.
- *Version 0.2, 18 September 2026.* Version 0.1 said that no binary Goppa parameters of usable size resist the attack. This rested on example sets with $n = 2^m$, which nothing requires, and the statement was withdrawn. The version added the dependence of the cost on m , example sets with $n < 2^m$, and a first count of the bits that a run moves. It also corrected the description of the TII challenges. No cost figure for the five Classic McEliece sets changed.
- *Version 0.3, 25 September 2026.* A constant of the solver formulas of [20, Section 5.3] was mistyped in our reimplementation. With the correct value the figures of columns (b) of Table 1 rose by 0.2 to 0.3 bits, and three figures of column (a) for the single run by 0.1.
- *Version 0.4, 5 October 2026.*
 - Two of the five assumptions of version 0.3 are now proved: the relaxed zero count (Theorem 3) and the structure of the interpolation relation (Theorem 5).
 - Assumption 1 is stated as a lower bound. Assumptions 2 and 3 exclude instances with dual codewords of low weight, and Assumption 3 includes the dimension counts of the algorithm. Equation (6) predicts the effect of such codewords.
 - Section 6 replaces the earlier discussion of storage. It contains the product by tiles (Proposition 7), the counts of operations, traffic and memory, and column (c) of Table 1.

It compares the run with ISD as a circuit that does not depend on the key, in several cost models for memory (Table 5). The behaviour of the solver is stated as Assumption 4, restricted to the parameters of the attack, together with the defect that can make it fail. The solver works on $\tilde{E}^\top \Omega \tilde{E}$ instead of $E^\top E$, on which it failed in one run at degree 7. The randomness of the solver’s output is proved (Proposition 8).

- The unit columns are recovered from the mt columns that one run reads and one guessed support element (Section 5.3). Earlier versions asked for the support elements of a few more than mt columns, which one run does not read, and their experiments used two runs with different unit columns.
- New experiments: keys in general position at (5, 4), runs at (7, 6), and further checks at full size. The comparison with [33] uses its current figures.
- The example sets of Table 11 are computed in the model of Section 6.
- The text was shortened and reorganised, and most proofs moved to Appendix A.
- *Version 0.5, October 2026, this version.* No cost figure changed.
 - A statement of versions 0.1 to 0.3 is withdrawn. Version 0.4 dropped it without saying so. By the formulas of [20, Section 5.3] a run takes 2^{94} to 2^{102} operations and stores 2^{57} to 2^{62} bits. Those versions said that a run that does not store E needs only $2^{47.8}$ to $2^{52.1}$ bits, and that its operation count “in the same model is no greater”. In other words, the smaller memory was said to cost no additional operations. They gave no derivation, and their figures with a charge for memory used both numbers.

The statement rested on an unpublished estimate for another solver, block Wiedemann, which we never ran. It gave $2^{89.8}$ to $2^{98.5}$ operations, but it charged one operation where the formulas charge two. So it was not in the same model. With the solver of the formulas, computing the entries of E anew in every product can only add operations. Section 6 now counts operations, traffic and memory for one and the same computation. It has $2^{92.3}$ to $2^{101.3}$ gates and $2^{49.7}$ to $2^{53.6}$ bits, and its solver has no proof.

- Section 6 says what T and T_f count, and the abstract and the introduction give both. Version 0.4 said of T that “the sequence of operations depends on the public key”. This could be read as a machine that is built for each key. The program is the same for all keys, and the bits that decide which additions it executes are computed during the run and counted.
- Appendix E explains the terms of T , and it sizes the gates for the mask bits and for the small matrices, which T_f omits.

References

- [1] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. Classic McEliece: conservative code-based cryptography: cryptosystem specification. Round-4 submission to the NIST post-quantum standardization process, October 2022. <https://classic.mceliece.org>.
- [2] The Classic McEliece team. Classic McEliece: conservative code-based cryptography: guide for security reviewers, October 2022. <https://classic.mceliece.org>.

- [3] M. R. Albrecht and G. V. Bard. The M4RI library. <https://github.com/malb/m4ri>.
- [4] D. Apon. An algebraic-geometry lower bound against the ePrint:2026/1747 McEliece key-recovery attack. Cryptology ePrint Archive, Paper 2026/1810, version of 26 August 2026.
- [5] M. Bardet, A. Lemoine, and J.-P. Tillich. An attack on the CFS scheme and on TII McEliece challenges. In *CRYPTO 2026, Part IV*, volume 16803 of *LNCS*, pages 380–412. Springer, 2026. Cryptology ePrint Archive, Paper 2026/430.
- [6] D. J. Bernstein. Circuits for integer factorization: a proposal. Manuscript, 2001. <https://cr.yp.to/papers.html#nfsccircuit>.
- [7] D. J. Bernstein. Grover vs. McEliece. In *PQCrypto 2010*, volume 6061 of *LNCS*, pages 73–80. Springer, 2010.
- [8] D. J. Bernstein. `break-fake-mceliece-sizes-from-tii.sh`. Script, 22 July 2026. <https://cr.yp.to/2026/20260722-break-fake-mceliece-sizes-from-tii.sh>.
- [9] D. J. Bernstein and T. Chou. CryptAttackTester: high-assurance attack analysis. In *CRYPTO 2024, Part VI*, volume 14925 of *LNCS*, pages 141–182. Springer, 2024.
- [10] R. P. Brent and H. T. Kung. The area-time complexity of binary multiplication. *Journal of the ACM*, 28(3):521–534, 1981.
- [11] P. Briaud, A. Lemoine, H. Randriambololona, and J.-P. Tillich. A heuristic subexponential attack on the McEliece cryptosystem. Cryptology ePrint Archive, Paper 2026/1232, 2026. Version of 17 September 2026.
- [12] D. Coppersmith. Solving homogeneous linear equations over $\text{GF}(2)$ via block Wiedemann algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.
- [13] A. Couvreur, R. Mora, and J.-P. Tillich. A new approach based on quadratic forms to attack the McEliece cryptosystem. In *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 3–38. Springer, 2023.
- [14] A. Esser and E. Bellini. Syndrome decoding estimator. In *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 112–141. Springer, 2022.
- [15] W. Eberly. Yet another block Lanczos algorithm: how to simplify the computation and reduce reliance on preconditioners in the small field case. In *ISSAC 2010*, pages 289–296. ACM, 2010.
- [16] J. Emsalem and A. Iarrobino. Inverse system of a symbolic power, I. *J. Algebra*, 174(3):1080–1090, 1995.
- [17] A. Esser, J. Verbel, F. Zveydinger, and E. Bellini. CryptographicEstimators: a software library for cryptographic hardness estimation. Cryptology ePrint Archive, Paper 2023/589, 2023. Python package `cryptographic-estimators`.
- [18] J.-C. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high-rate McEliece cryptosystems. *IEEE Trans. Inform. Theory*, 59(10):6830–6844, 2013.
- [19] R. Fröberg. An inequality for Hilbert series of graded algebras. *Math. Scand.*, 56:117–144, 1985.

- [20] A. Ghoshal, Y. Ishai, A. Jain, and N. Sun. Quasipolynomial cryptanalysis of the McEliece cryptosystem (or: PIR meets McEliece). Cryptology ePrint Archive, Paper 2026/1630, 2026. Version of 27 August 2026.
- [21] T. Hemmert. Key recovery for the McEliece cryptosystem using higher-order vanishing. Cryptology ePrint Archive, Paper 2026/1339, 2026.
- [22] T. Hemmert and A. Wiemers. Distinguishing Goppa codes using higher-order vanishing. In *CRYPTO 2026, Part IV*, volume 16803 of *LNCS*, pages 191–223. Springer, 2026.
- [23] E. Kirshanova and A. May. Breaking Goppa-based McEliece with hints. *Information and Computation*, 293:105045, 2023.
- [24] E. Kirshanova and A. Esser. TII McEliece challenges: scripts and instances. https://github.com/ElenaKirshanova/tii_decoding_challenge, 2023; challenge description at <https://crowdchallenge.tii.ae/mceliece-challenges/>.
- [25] A. K. Lenstra, A. Shamir, J. Tomlinson, and E. Tromer. Analysis of Bernstein’s factorization circuit. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 1–26. Springer, 2002.
- [26] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42–44, Jet Propulsion Laboratory, 1978.
- [27] P. L. Montgomery. A block Lanczos algorithm for finding dependencies over $\text{GF}(2)$. In *EUROCRYPT ’95*, volume 921 of *LNCS*, pages 106–120. Springer, 1995.
- [28] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. December 2016.
- [29] L. Panny. On breaking McEliece keys using brute force. Cryptology ePrint Archive, Paper 2025/632, 2025.
- [30] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
- [31] H. Randriambololona. The syzygy distinguisher. In *EUROCRYPT 2025, Part VI*, volume 15606 of *LNCS*, pages 324–354. Springer, 2025.
- [32] M.-J. O. Saarinen. HOVER: higher-order vanishing endomorphism recovery. Cryptology ePrint Archive, Paper 2026/1778, 2026. Recovered keys at <https://github.com/mjosaarinen/tii-solved>.
- [33] M.-J. O. Saarinen. Bit operation cost of “holdout” key-recovery attacks against Classic McEliece. Cryptology ePrint Archive, Paper 2026/1786, 2026. Version of 15 September 2026. Artifact at <https://github.com/mjosaarinen/mccost>.
- [34] M.-J. O. Saarinen. Two-anchor holdout/Hermite: solving the TII-254 McEliece key recovery challenge. Cryptology ePrint Archive, Paper 2026/1986, 2026. Version of 24 September 2026.
- [35] The Sage Developers. SageMath, the Sage Mathematics Software System. <https://www.sagemath.org>.
- [36] N. Sendrier. Decoding one out of many. In *PQCrypto 2011*, volume 7071 of *LNCS*, pages 51–67. Springer, 2011.

- [37] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed–Solomon codes. *Discrete Math. Appl.*, 2(4):439–444, 1992.
- [38] R. P. Stanley. Weyl groups, the hard Lefschetz theorem, and the Sperner property. *SIAM J. Algebraic Discrete Methods*, 1(2):168–184, 1980.
- [39] J. Stern. A method for finding codewords of small weight. In *Coding Theory and Applications 1988*, volume 388 of *LNCS*, pages 106–113. Springer, 1989.
- [40] K. Vedenev. Extending distinguishing to key recovery for subfield subcodes of GRS codes. Cryptology ePrint Archive, Paper 2026/1747, 2026. Working draft of 6 September 2026.
- [41] C. Wieschebrink. Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes. In *PQCrypto 2010*, volume 6061 of *LNCS*, pages 61–72. Springer, 2010.