Key Recovery Against RLCE via the GIJS Distinguisher

Jacob Alperin-Sheriff*
Independent Researcher
jacobmas@gmail.com

November 25, 2026

Abstract

We present an experimental cryptanalysis of the RLCE cryptosystem [Wan15, Wan] at the heretofore secure $w = n-k$ parameter regime. Our attack relies crucially on the recent Ghoshal–Ishai–Jain–Sun (GIJS) distinguisher [GIJS26]. The attack shortens the public key at random positions, runs the GIJS distinguisher with *every* column as hold-out, and uses this distinguisher to identify twin pairs based on how they appear across trials. Three pairs so identified are unmixed by a root-finding cascade, and Sidelnikov–Shestakov on the resulting $k \times (k+3)$ pure matrix recovers the key.

We experimentally verify our attack by breaking the ID-0 (128-bit) in 30.5 core-hours (about 3 hours of wall time) on a standard higher-end laptop.

1 Introduction

The RLCE cryptosystem of Wang [Wan15, Wan] was one of the 69 schemes submitted to NIST's postquantum cryptography standardization process. RLCE builds its public key from a generalized Reed–Solomon code $GRS_{n,k}$ by injecting w *twin pairs*: for each pair, two GRS columns are mixed with two random columns via a 2×2 invertible matrix, and all $N = n + w$ columns are permuted. The three unbroken parameter crucially all have $w = n - k$, the exact boundary where the CLT square distinguisher provably fails.

This scheme was one of many [Wie09, BBC13, Wie16, ABNS21, Kro20] attempts to patch Niederreiter's original attempt to instantiate the McEliece [McE78] cryptosystem with GRS codes [Nie86], which was itself broken by the Sidenikov–Shestakov attack [SS92]. The original scheme and all of these other previous attempts have fallen to cryptanalysis.

Due to security concerns raised by the Couvreur–Lequesne–Tillich (CLT) [CLT18] attack on smaller parameters, NIST did not advance RLCE to the second round; it appears among the schemes for which NIST "lacked full confidence in security" [Moo19]. See [Nat19] for the Round 1 report. No subsequent public attack has been advanced against the $w = n-k$ parameter sets since, and due to the large parameter sizes the scheme may not have received any attention at all in the intervening period. However, with the recent improvements in AI cryptanalysis [SW26, Saa26] causing concerns from eminent cryptographers that key exchange/key encapsulation itself may be at risk [Gre26], it is likely important to subject even large parameter schemes to renewed cryptanalysis attempts in order to enable a potential broadening of standardized schemes for cryptographic agility as a hedge against AI breakthroughs against lattices, isogenies or quasi-cyclic codes.

*The author was a member of the NIST PQC standardization team during Round 1 and participated in the evaluation of RLCE.

There has been recent hubbub over the Ghoshal–Ishai–Jain–Sun [GIJS26] provable distinguisher and heuristic key recovery for the venerable and ancient (by cryptographic standards) McEliece scheme [McE78] and subsequent improvement papers [Wei26]. Most notably it has led to the German Federal Office for Information Security (BSI) shifting its guidance from wholesale approval to recommending against the use of the Classic McEliece scheme in new deployments [Fed26]. In the course of a very long thread on NIST's PQC forum over exactly how devastating or not these attacks are, in practice, Dr. Wang commented that "I do not think these attacks on McEliece will work on RLCE. I still believe RLCE has sufficient randomness in the generator matrix and is a good alternative for McEliece schemes," [Wan26] which piqued the interest of this work's author.

We contradict Wang's claim for *both* distinguishing and key recovery by using the GIJS distinguisher: the distinguishing step is proved, and the key-recovery step is an experimentally verified cryptanalysis [GIJS26].

1.1 The Attack

In slightly more detail, our attack proceeds as follows.

1. **Shorten** the public key at $\ell = k - k'$ random positions, for some *small* k'.
2. **Distinguish** using the recent Ghoshal–Ishai–Jain–Sun [GIJS26] distinguisher (Theorem 1), which operates beyond the Schur-square barrier, applied *directly* with every column as holdout.
3. **Group** columns across trials via co-occurrence with twin pairs.
4. **Recover** the original GRS columns from 3 twin pairs.
5. **Solve** using Sidelnikov-Shestakov.

A key insight is that GIJS will still work even with some random columns, and that, furthermore, the pairs of noisy columns provide useful information to the GIJS distinguisher. We show in Section 4 that the GIJS distinguisher for GRS subcodes should work on shortened RLCE public keys whenever $sC + U > dD$, where s is the jet parameter, d is the degree of the multivariate polynomials, C is the number of clean (either from shortening at one of a twin pair's columns or from an initially clean column) columns in the shortened code, and U is the number of twin pairs remaining in the shortened code.

Extensive experimental testing shows that the GRS distinguisher on the parameter sets of interest essentially never fails to diagnose a "clean" GRS column as GRS and a noisy column as "random." This leads to practical attacks.

In short, we simply choose uniformly at random columns to shorten at for a relatively small (less than 100 for the sets considered) number of separate *trials*. In each trial, we run the GIJS distinguisher on the shortened code, with each column held out separately as the holdout column. This enables us to learn which columns in the shortening remain part of a twin pair (meaning both it and its pair column are in the shortened code) in that shortening, and refer to these columns as *flagged*.

To recover twin pairs, we observe that (assuming no errors in the distinguisher) a column is *flagged* in a trial if and only its twin pair column is also *flagged* in that trial. This fact means that after a relatively small number of randomized trials, for a given column A, there will be only other column B that is flagged in and only in trials in which column A is flagged, and that this column B is the twin pair for column A.

After recovering all twin pairs, we use some clever ideas in5 to unmix 3 of these twin pairs back into actual GRS columns. We note that a simpler route to do so would involve simply trying all $(q-1)^3$ non-degenerate linear combinations of the pairs, one of which will correctly unmix all 3.

Finally, after we have recovered a $k \times (k+3)$ pure GRS matrix, we are able to run the celebrated [SS92] to recover the code structure (the scrambling matrix S). At this point, unmixing the remainder of the twin pairs becomes trivial (see [CLT18]). We end up with the full private key.

1.2 Experimental Success

We were able to run a full key recovery (see Section 6) at the 128 bit parameters (ID-0) in the specification that took approximately 30.5 hours of core work on a standard high-end laptop to recover the key. We have also given key recovery data for several other smaller parameter sets for comparison in the $w = n - k$ regime; see Section 6 for details.

While we did not execute an experimental attack on $ID-1$ and $ID-2$ due to a lack of desire to spend personal funds, we estimate that a cloud based attack should be possible for no more than a few hundred dollars; the biggest issues, as mentioned in [GIJS26], are the memory costs of the attack. However, these are easily solvable in the cloud for the given parameter sets.

1.3 Further Work

The scheme is completely broken in the $w = n - k$ parameter regime, as a back of the envelope computation estimates that a public key would need to be in the petabytes with hundreds of megabytes per ciphertext in order to reach 128 bits of security. However, there is a natural defensive option; push the number of twin pairs w to its maximum $w = n$, eliminating pure columns entirely ($N = 2n$). Our attack does not work in this regime when $2k \geq n$, so one could potentially propose this as a patch. However, due to the matrix becoming wider while the error correction regime does not change, ISD attack perform better, necessitating a bump in k, and significantly raising parameter sizes.

Below, we have calculated parameter sets that seem to be immune to the attacks given in this paper, as well as to ISD-based attacks.

Table 1: Proposed $w = n$ parameter sets immune to our attack and ISD

	n	k	t	N	field	Stern	pk (KB)	ct (kB)
128-fixed	1191	619	286	2382	2^{11}	140.2	1980	3.2
192-fixed	1774	922	426	3548	2^{11}	204.0	4393	4.8
256-fixed	2355	1225	565	4710	2^{12}	268.1	8452	6.9

We do not claim that these parameters are secure, and if forced, we would bet against their security given the long, long history of attacks on GRS-McEliece schemes and various attempts to patch. We merely give them as new targets and the smallest scheme parameters secure given our new attacks for future research.

2 Preliminaries

Definition 1 (Generalized Reed–Solomon code)**.** Let q be a prime power, $n \leq q$, $1 \leq k \leq n$. Choose distinct $x_1, \ldots, x_n \in \mathbb{F}_q$ (evaluation points) and nonzero $y_1, \ldots, y_n \in \mathbb{F}_q^*$ (column multipliers). The

generalized Reed–Solomon code $\mathrm{GRS}_{n,k}(x, y)$ is the $[n, k]$ linear code with generator matrix

$$(G_{\mathrm{GRS}})_{i,j} = y_j x_j^i, \qquad i = 0, \ldots, k-1,\ j = 1, \ldots, n,$$

i.e., the evaluation of all polynomials of degree $< k$ at the x_j, scaled by y_j. GRS codes are MDS: minimum distance $d = n - k + 1$. See [MS77, Ch. 10] or [HP03, Ch. 5].

Definition 2 (Shortening)**.** Let C be an $[n, k]$ code and $L \subset \{1, \ldots, n\}$. The *shortening* of C at L, denoted C_L or $\mathcal{S}_L(C)$, is obtained by restricting to codewords that vanish on L, then deleting those positions. If the generator columns at L are linearly independent, $\dim C_L = k - |L|$ and the length is $n - |L|$.

2.1 The RLCE scheme

We use the $S \cdot G_s \cdot P_1 \cdot A \cdot P_2$ formulation from the RLCE specification [Wan16, Wan]. Everything happens over the finite field $\mathbb{F}_q$ for $q = 2^m$.

Let G_s be a $k \times n$ GRS generator matrix. Key generation:

- S: uniform random $k \times k$ invertible matrix (scrambler).
- P_1: uniform random $n \times n$ permutation matrix.
- A: $n \times (n + w)$ *expansion* matrix. For w designated positions $j_1, \ldots, j_w$, column j_t of $G_s P_1$ is replaced by two columns
$$a_t g_{j_t} + c_t r_t, \qquad b_t g_{j_t} + d_t r_t,$$
where $r_t \in \mathbb{F}_q^k$ is uniform random and $\left(\begin{smallmatrix} a_t & b_t \\ c_t & d_t \end{smallmatrix}\right) \in \mathrm{GL}_2(\mathbb{F}_q)$ is uniform random. All other columns pass through unchanged. The pairs $(a_t g_{j_t} + c_t r_t,\ b_t g_{j_t} + d_t r_t)$ are the *twin pairs*.
- P_2: uniform random $(n + w) \times (n + w)$ permutation matrix.

The public key is $G_{\mathrm{pub}} = S \cdot G_s \cdot P_1 \cdot A \cdot P_2$, a $k \times (n+w)$ matrix. The secret key is (S, G_s, P_1, A, P_2). Encryption is $c = mG_{\mathrm{pub}} + e$ with $\mathrm{wt}(e) \le t$; decryption unpermutes by P_2^{-1}, unmixes each twin pair via the 2×2 inverse, unpermutes by P_1^{-1}, and decodes the GRS code. Security of the post-CLT paper updated version relies on the indistinguishability of G_{pub} from random solely at $w = n - k$; for a more precise statement of security, consult the specification.

For completeness, we reproduce the specification's parameter sets below.

Set	n	k	t	w	m
ID-0	630	470	80	160	10
ID-1	1000	764	118	236	10
ID-2	1360	800	280	560	11

Table 2: Parameter sets from the RLCE specification

2.2 The CLT Attack

We work with G_{GRS} in canonical form as above. The RLCE public key has $N = n + w$ columns: $n - w$ pure GRS columns and w twin pairs, each $(c_i, c_{\tau(i)})$ of the form

$$c_i = aG_j + cR, \qquad c_{\tau(i)} = bG_j + dR,$$

with G_j a GRS column, $R \in \mathbb{F}_q^k$ random, and $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ invertible, all permuted by secret π.

Shortening at L deletes those positions after restricting to codewords vanishing there. By [CLT18, Lemma 24], shortening at one twin-pair member *derandomizes* the other. If L hits every pair, $C_L = M \cdot G'_{\mathrm{GRS}}$ with M generic $k' \times k_{\mathrm{GRS}}$, $k' = k - |L|$. If L misses j pairs, those $2j$ columns remain mixed (noisy).

The ambient GRS dimension is $k_{\mathrm{GRS}} = k' + w - j$ for j unhit pairs.

High-level overview. The Couvreur–Lequesne–Tillich attack [CLT18] proceeds in three phases. First, *distinguish*: the Schur square $C^{\star 2}$ has abnormally low dimension when the gap $c = (n-k)-w$ exceeds $(3+\sqrt{16w+1})/2$, revealing twin-pair positions. Second, *unmix*: for each identified pair, the 2×2 mixing is inverted to recover pure GRS columns. Third, *recover*: Sidelnikov–Shestakov [SS92] on the reconstructed GRS code yields the secret support and multipliers. The attack *succeeds* for w sufficiently below $n-k$ but *fails* at the shipped $w = n-k$ boundary, where the square is provably full and no distinguisher was known.

2.3 Hasse Derivatives, Multiplicity Codes, and Jets

We follow the treatment of [GIJS26]. For $\mathbf{a} = (a_1, \ldots, a_{k'}) \in \mathbb{Z}_{\geq 0}^{k'}$ write $|\mathbf{a}| = \sum_i a_i$ for its ℓ_1-norm, and let

$$A_{<s,k'} := \{\mathbf{a} \in \mathbb{Z}_{\geq 0}^{k'} : |\mathbf{a}| < s\}$$

be the set of multi-indices of order below s. We write $\mathbf{X}^{\mathbf{a}} := \prod_i X_i^{a_i}$ for the corresponding monomial.

Definition 3 (Hasse derivatives [GIJS26, Def. 2.7])**.** Let $F \in \mathbb{F}[X_1, \ldots, X_{k'}]$ over a field $\mathbb{F}$. The *Hasse derivative* of F with respect to $\mathbf{a} \in \mathbb{Z}_{\geq 0}^{k'}$ is

$$\partial^{[\mathbf{a}]} F = \sum_{\mathbf{b} \geq \mathbf{a}} \binom{\mathbf{b}}{\mathbf{a}} \cdot \mathrm{Coeff}_{\mathbf{b}}(F)\, \mathbf{X}^{\mathbf{b}-\mathbf{a}},$$

where $\mathrm{Coeff}_{\mathbf{b}}(F)$ is the coefficient of $\mathbf{X}^{\mathbf{b}}$ in F and $\binom{\mathbf{b}}{\mathbf{a}} = \prod_i \binom{b_i}{a_i}$. Equivalently, Hasse derivatives are the finite-field analogue of ordinary partial derivatives, characterized by the formal Taylor expansion

$$F(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{a}} (\partial^{[\mathbf{a}]} F)(\mathbf{X})\, \mathbf{Z}^{\mathbf{a}}.$$

For a univariate polynomial $f(Z) = \sum_{j=0}^{d} c_j Z^j$ we write $\partial^{[r]} f(Z) = \sum_{j=r}^{d} \binom{j}{r} c_j Z^{j-r}$; in characteristic 0 this equals $f^{(r)}(Z)/r!$. The Hasse-derivative chain rule holds in any characteristic, which is what makes multiplicities usable over finite fields.

Definition 4 (Hasse jet [GIJS26, Def. 2.9])**.** For $\mathbf{x} \in \mathbb{F}_q^{k'}$ and $s \geq 1$, the *Hasse jet of order* $< s$ of F at $\mathbf{x}$ (also the *order-s evaluation* of F at $\mathbf{x}$) is the vector of all Hasse derivatives of order below s evaluated at $\mathbf{x}$:

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

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

Definition 5 (Multiplicity code [GIJS26, Def. 2.10], [KSY14])**.** The multiplicity code of order-s evaluations of polynomials of total degree at most d in k' variables over $\mathbb{F}_q$ has length $q^{k'}$ over the alphabet $\Sigma = \mathbb{F}_q^{\binom{k'+s-1}{s-1}}$, indexed by $\mathbf{x} \in \mathbb{F}_q^{k'}$, with codeword $\mathrm{Encode}_{s,d,k',q}(F) = \left(J_{\mathbf{x}}^{<s}(F)\right)_{\mathbf{x} \in \mathbb{F}_q^{k'}}$ for each F of total degree at most d.

The key analytic fact is Hermite interpolation: over any field, a univariate polynomial of degree at most d is determined by its order-$< s$ Hasse jets at enough points.

Lemma 1 (Hermite interpolation of Hasse derivatives [GIJS26, Lem. 2.8])**.** *Let f be a univariate polynomial of degree at most d over a finite field $\mathbb{F}$, let $\alpha_1, \ldots, \alpha_h$ be distinct elements of $\mathbb{F}$, and let $\mu_1, \ldots, \mu_h$ be positive integers with $\sum_i \mu_i > d$. Then f is recovered in $\mathrm{poly}(d, h, \log|\mathbb{F}|)$ time from the values $(\partial^{[j]} f)(\alpha_i)$ for $i \in [h]$, $0 \le j < \mu_i$.*

Use in the GIJS distinguisher. Let $Y = [\mathbf{y}_1 \mid \cdots \mid \mathbf{y}_{n'}] \in \mathbb{F}_q^{k' \times n'}$ be the input matrix and fix a hold-out column τ. The distinguisher computes the kernel

$$K_\tau := \Big\{ H \in \mathcal{H}_{k',d} : J^{<s}_{\mathbf{y}_j}(H) = 0 \text{ for all } j \neq \tau \Big\},$$

where $\mathcal{H}_{k',d}$ is the space of *homogeneous* degree-d polynomials in k' variables, of dimension $\binom{k'+d-1}{d}$. It outputs "GRS" iff every $H \in K_\tau$ vanishes at $\mathbf{y}_\tau$. Restricting to homogeneous polynomials kills the unknown GRS column multipliers: for homogeneous H of degree d,

$$(\partial^{[\mathbf{a}]} H)(\lambda \mathbf{x}) = \lambda^{d-|\mathbf{a}|} \cdot (\partial^{[\mathbf{a}]} H)(\mathbf{x}) \qquad (\lambda \neq 0),$$

so $J^{<s}_{\lambda\mathbf{x}}(H) = 0 \iff J^{<s}_{\mathbf{x}}(H) = 0$ [GIJS26, Lem. 3.1]. The two feasibility conditions are the *reconstruction* inequality $s(n'-1) > dD$, which forces every $H \in K_\tau$ to vanish at a clean hold-out via Hermite interpolation along the hidden degree-D curve, and the *kernel* inequality $(n'-1)\binom{k'+s-1}{s-1} < \binom{k'+d-1}{d}$, which guarantees $K_\tau \neq \{0\}$ on a random input so that Schwartz–Zippel applies.

2.4 The GIJS distinguisher

The Ghoshal–Ishai–Jain–Sun [GIJS26] distinguisher tests whether a given matrix spans a subcode of a generalized Reed–Solomon code. Given n' columns in $\mathbb{F}_q^{k'}$, it interpolates: it seeks a nonzero homogeneous degree-d polynomial vanishing to order s at $n'-1$ of the columns (the interpolation set), then checks whether it also vanishes at the remaining hold-out column. For a true GRS subcode, the hidden rational-curve structure forces this vanishing; for random columns, Schwartz–Zippel says it fails with high probability. The two inequalities below encode feasibility: the first ensures enough interpolation constraints to force vanishing on the curve, the second that the interpolation kernel is nonzero.

Theorem 1 (Ghoshal et al., Theorem 3.2)**.** *Let $n' \le q$, $k' \le K$, $d < q$, $D := K - 1$. If*

$$s(n'-1) > d \cdot D \tag{1}$$

$$(n'-1)\binom{k'+s-1}{s-1} < \binom{k'+d-1}{d} \tag{2}$$

then there is a $\mathrm{poly}\Big(\binom{k'+d-1}{d}, n', \log q\Big)$-time algorithm distinguishing a random k'-dimensional subcode of random $GRS_{n',K}$ from uniform random, with gap $\ge 1 - d/q$.

The algorithm (Algorithm 1 of [GIJS26]): hold out an arbitrary column τ, compute the kernel K_τ of Hasse-jet evaluations (order $< s$) at the other $n'-1$ columns in the space of homogeneous degree-d polynomials, and output "GRS" iff every $H \in K_\tau$ vanishes at y_τ, "Random" otherwise.

Variables. k' is the dimension of the subcode; K the hidden ambient GRS dimension ($k' + w$ here); s the jet multiplicity; d the test-polynomial degree. The first inequality is *reconstruction*; the second the *kernel* condition.

3 Phase 1

Here we describe the first phase of the attack, which uses random code shortenings and the GIJS distinguisher to recover and unmix all the noisy pairs. Given the $k \times (n + w)$ RLCE public matrix, we proceed as follows. Let (n', k', s, d, D, q) be GIJS parameters satisfying the kernel inequality $(n' - 1)\binom{k'+s-1}{s-1} < \binom{k'+d-1}{d}$. Here, $D = k' + w$ is the degree of the hidden rational curve.

3.1 Choosing the GIJS parameters

Given the RLCE parameters (n, k, w, q) with public-key length $N = n + w$, the attacker first fixes a target shortened length n'. Shortening at $N - n'$ positions drops the dimension by the same amount, so

$$k' = k - (N - n').$$

After shortening, the hidden ambient GRS code has dimension $K = k' + w$, so the hidden rational curve has degree

$$D = K - 1 = k' + w - 1.$$

It remains to choose the jet multiplicity s and the test-polynomial degree d. We search over small integer pairs with $1 \le s \le d < q$ satisfying both GIJS feasibility inequalities:

1. *Reconstruction:* $s(n' - 1) > d \cdot D$, which forces every $H \in K_\tau$ to vanish at a clean hold-out;
2. *Kernel:* $(n' - 1)\binom{k'+s-1}{s-1} < \binom{k'+d-1}{d}$, which guarantees the interpolation kernel is nonzero on random input.

Among feasible pairs (s, d), we prefer smaller d: the GIJS matrix has $\binom{k'+d-1}{d}$ columns, so d dominates the per-run cost. Subject to that constraint, we prefer a larger relative margin on the kernel inequality, since a near-tight kernel is numerically fragile.

There is a tension in choosing n' itself. A smaller k' (hence smaller n') makes each GIJS run cheaper, but GIJS also needs a hard column-count floor: n' must be large enough that the reconstruction inequality can hold at all. In practice n' cannot be pushed far below the boundary where $s(n' - 1) \approx dD$.

3.2 The Core of the Attack

Let t' denote the number of noisy columns among the $n' - 1$ interpolation columns (all columns except the holdout column) at any given point in the following algorithm.

Dirty column tolerance. We define $t'_{\max}$ to be the maximum number of noisy columns such that PIR persistence (see Section 4) guarantees acceptance on a clean hold-out:

$$t'_{\max} = \max\{t' : s \cdot ((n' - 1) - t') > d \cdot D\}. \tag{3}$$

A trial is *recorded* — its flagged set contributes to the frequency table — iff $|V| \le T_{\text{reject}} := t'_{\max}$.

We deliberately obfuscate an inner step, namely the computation of the set of holdout columns τ on which the GIJS subroutine outputs random. While this can be trivially computed by separately

running the GIJS subroutine as a black box on each holdout column, we have a quicker method, described in Section 3.3.

Algorithm 1: Phase 1

Input: Original code parameters n, w, k, k', q; raw trial count T; recording threshold T_{reject}
Output: All w pairs (c_1, c_2) of noisy columns in the original code
$W \leftarrow \emptyset$;
for *iter* $\leftarrow 1$ **to** T **do**
  Choose a random subset $\mathcal{L} \subset [1, n+w]$ where $|\mathcal{L}| = k - k'$;
  Compute the $k' \times n'$ shortened code by shortening at $\mathcal{L}$;
  $V \leftarrow \emptyset$;
  Compute V as the set of holdout columns $\tau \in \{1, \dots n'\}$ on which the GIJS subroutine outputs "random." (see Algorithm 2)
  $W \leftarrow W \cup \{V\}$;
Use W to compute all w pairs of noisy columns (see Algorithm 3)
return w twin pairs (c_1, c_2)

3.3 Reuse Speedup

We can speed up the attack by noticing that the jet matrix for each holdout attack has only small differences from the others. Consider a specific shortening. In the language of the GIJS attack we have one shortened point set $y_1, \dots, y_{n'} \in \mathbb{F}_q^{k'}$. For a hold-out τ, the GIJS decision is computed from the jet matrix M_τ: its rows are indexed by $(j, \mathbf{a})$ with $j \neq \tau$ and $\mathbf{a} \in A_{<s,k'}$, its columns by the monomials $\mathbf{X}^{\mathbf{m}}$ of degree d, and its entry is the Hasse derivative $(\partial^{[\mathbf{a}]}\mathbf{X}^{\mathbf{m}})(y_j)$. Writing z for the coefficient vector of $H \in V_d$, we have that $(M_\tau z)_{(j,\mathbf{a})} = (\partial^{[\mathbf{a}]}H)(y_j)$, so $\ker M_\tau$ is the interpolation kernel K_τ. Let $\mathbf{v}_\tau \in \mathbb{F}_q^{N_d}$ be the evaluation vector $\mathbf{v}_\tau = (y_\tau^{\mathbf{m}})_{\mathbf{m}}$ at which the GIJS distinguisher is executed.

Now, let M_{full} be the jet matrix of *all* n' points, with $E_F := n'J$ rows grouped in n' consecutive blocks of J rows (block j holds the jets at y_j, with $\mathbf{a} = \mathbf{0}$ first). Note that M_τ is M_{full} with block τ deleted, and that $\mathbf{v}_\tau$ is exactly the $(\tau, \mathbf{0})$-row of M_{full}, since $(\partial^{[\mathbf{0}]}\mathbf{X}^{\mathbf{m}})(y_\tau) = y_\tau^{\mathbf{m}}$.

Call a vector $\mathbf{s} = M_{\text{full}} z$ a *syndrome*. As z ranges over $\mathbb{F}_q^{N_d}$, the syndromes range over the column space of M_{full}, which is the kernel of any matrix $P^\perp$ whose rows form a basis of the *left* kernel of M_{full} (the row relations $\mathbf{u}^T M_{\text{full}} = 0$). Write $h := E_F - \operatorname{rank}(M_{\text{full}})$ for the number of such basis rows.

Theorem 2 (Exactness of hold-out reuse)**.** *For every $\tau \in [n']$, the following are equivalent:*

(i) the per-hold-out GIJS decision flags τ;

(ii) there is a syndrome $\mathbf{s} = M_{\text{full}} z$ supported only on block τ with $s_{(\tau,\mathbf{0})} \neq 0$;

(iii) $\operatorname{rank}(P^\perp|_\tau) = \operatorname{rank}(P^\perp|_\tau^{-\mathbf{0}})$*, where $P^\perp|_\tau$ is $P^\perp$ restricted to the J columns of block τ and $P^\perp|_\tau^{-\mathbf{0}}$ is the same restriction with the $\mathbf{a} = \mathbf{0}$ column removed. If $P^\perp$ is empty ($h = 0$), then* (iii) *is declared to hold for every τ.*

Hence all n' decisions of a trial are obtained from one left-kernel basis $P^\perp$ and two rank computations per hold-out.

Proof. (i) $\iff$ (ii): τ is flagged iff there is $z \in \ker M_\tau$ with $\langle z, \mathbf{v}_\tau \rangle \neq 0$. For such a z, the syndrome $\mathbf{s} = M_{\text{full}} z$ vanishes outside block τ (that is $M_\tau z = 0$) and $s_{(\tau,\mathbf{0})} = \langle z, \mathbf{v}_\tau \rangle \neq 0$. Conversely, if $\mathbf{s} = M_{\text{full}} z$ is supported on block τ with $s_{(\tau,\mathbf{0})} \neq 0$, then $M_\tau z = 0$ and $\langle z, \mathbf{v}_\tau \rangle = s_{(\tau,\mathbf{0})} \neq 0$.

(ii) $\iff$ (iii): the syndromes supported on block τ are exactly the vectors $\mathbf{u} \in \mathbb{F}_q^J$ with $P^\perp|_\tau \mathbf{u} = 0$, i.e. $\ker(P^\perp|_\tau)$, embedded in block τ: indeed $\mathbf{s} \in \ker P^\perp$ iff $\mathbf{s} = M_{\text{full}} z$ for some z. Such a pattern with $u_{\mathbf{0}} \neq 0$ exists iff the first column of $P^\perp|_\tau$ lies in the span of the remaining columns: if $\mathbf{u} \in \ker(P^\perp|_\tau)$ has $u_{\mathbf{0}} \neq 0$, then $\text{col}_{\mathbf{0}} = -\sum_{i \geq 1} (u_i/u_{\mathbf{0}}) \, \text{col}_i$; if $\text{col}_{\mathbf{0}} = \sum_{i \geq 1} w_i \, \text{col}_i$, then $(1, -w_1, \dots) \in \ker(P^\perp|_\tau)$ has nonzero first entry. And $\text{col}_{\mathbf{0}}$ lies in the span of the others iff removing it does not drop the rank, which is (iii). If $h = 0$, then M_{full} has full row rank, $\ker P^\perp = \mathbb{F}_q^{E_F}$, the pattern $\mathbf{e_0}$ is achievable on block τ, and (ii) holds for every τ directly. □

Algorithm 2 turns Theorem 2 into a three phase procedure. In the first phase it assembles the single matrix M_{full}. Next, it extracts *every* linear relation among the rows of M_{full}, in one pass. Forward elimination is run on the augmented matrix $[M_{\text{full}} \mid I_{E_F}]$. The identity block is used as a ledger to record combinations of the original rows formed by forward elimination. Whenever a row of the left block has been reduced to zero, the ledger row opposite it is therefore an explicit combination of the rows of M_{full} that cancels them. Collecting all such ledger rows gives the basis $P^\perp$ of the left kernel. This elimination step has a cost comparable to one per-hold-out GIJS run. The third phase, the loop over τ, merely reads $P^\perp$ locally, one block at a time.

Algorithm 2: Hold-out reuse: all GIJS decisions of one trial

Input: Shortened columns $y_1, \dots, y_{n'} \in \mathbb{F}_q^{k'}$; GIJS parameters (s, d).
Output: The set $V \subseteq [n']$ of flagged hold-outs, identical to n' separate GIJS runs.
Build M_{full} $(E_F \times N_d)$, rows grouped in blocks of J, $\mathbf{a} = \mathbf{0}$ first in each block;
Forward-eliminate the augmented matrix $[M_{\text{full}} \mid I_{E_F}]$; let $P^\perp$ be the rows of the accumulated transform opposite the zero rows of the echelon form;
if $h = 0$ **then**
  return $V = [n']$;
$F \leftarrow \emptyset$;
for $\tau \leftarrow 1$ **to** n' **do**
  $A \leftarrow P^\perp$ restricted to the J columns of block τ;
  if $\text{rank}(A) = \text{rank}(A$ *without its first column*$)$ **then**
    $V \leftarrow V \cup \{\tau\}$;
return V;

Restricting $P^\perp$ to the J columns of block τ discards everything a relation says about the other points and keeps only its entries on τ's own rows. According to Theorem 2, τ is flagged by the GIJS distinguisher as random if and only if the first of these J columns is already a linear combination of the remaining $J - 1$. As a result, computing the rank of the matrix of the full J columns and comparing it to the rank of the last $J - 1$ columns is sufficient; the rank drops if the distinguisher outputs GRS while it stays the same if it outputs random.

The early exit for $h = 0$ is a degenerate case: with no row relations at all, every syndrome is achievable and thus every hold-out is flagged as random.

3.4 Unmixing Pairs Via Clustering

We now show how to take flagged sets of columns from the tria runs and recover pairs. Here, the flagged sets of columns mean the set W to which sets of columns flagged as "random" were added in Algorithm 1 after each good trial.

The idea is simple once the flagged set of a single trial is looked at the right way. In a trial shortening, a twin pair can be in only three states. If both members are kept, then both are

dirty columns, so both are flagged with very very high probability. If exactly one member is kept, and then the survivor is derandomized to a clean column per the analysis of the Couvrier et al paper [CLT18], so that column is not flagged, and the other member is absent from the trial altogether. The final case is where both members are hit by the shortening, and neither appears. In all three cases it is with very very high probability the case that either both are flagged or neither are flageed.

Finally, a column that belongs to no twin pair, (a pure GRS column), is never flagged at all.

As a result, if both members of a pair are correctly flagged with high enough probability, then the list of trials in which column A was flagged is the same as the list of columns for which its twin column B was flagged. Meanwhile, it is an empty list for a pure column, and for other twin pairs, the list is non-empty but with high probability over the random choices of shortenings distinct. Pair identification thus becomes grouping columns by their *pattern*.

3.4.1 The algorithm

For each original public column i, record its pattern $b_i \in \{0,1\}^T$, where $b_i(t) = 1$ if i was flagged in trial t and 0 otherwise. Then proceed as follows.

Algorithm 3: Flag-pattern clustering

Input: Flagged sets $V_1, \ldots, V_T$ of T trials, mapped back to original public-column indices
Output: Sets of twin pairs and pure columns; anomalies, if any
For each column i, form its pattern b_i, with $b_i(t) = 1$ iff $i \in V_t$;
Group columns by *exact equality* of patterns;
A group of size 2 with a nonzero pattern is a twin pair;
The group with the all-zero pattern is the pure columns;
A group of size 1 with a nonzero pattern is an orphan;
A group of size 3 or more is oversized, and is reported, not paired;
Compare the size of the all-zero group with the expected number $n - w$ of pure columns, and report any excess;

3.4.2 Trials for Success

We now compute how many trials (assuming that pairs are with overwhelming probability correctly flagged together) are necessary to identify and unmix pairs with high probability. Fix a pair. The probability both columns of the pair appear in the shortening is

$$p \;=\; \frac{(n')(n'-1)}{(n+w)(n+w-1)},$$

the probability that a given pair survives a random shortening to n' out of N columns. Two distinct pairs separate in a recorded trial except with probability at most $p^2 + (1-p)^2$, when either both pairs appear in the shortening, or neither pair appears in the shortening.

Note that we heuristically treat the pairs as independent here. A union bound over the $\binom{w}{2}$ pairs of pairs then says that T trials separate every pair from every other pair with failure probability at most ϵ as soon as

$$\binom{w}{2}\left(p^2 + (1-p)^2\right)^T \;\leq\; \epsilon,$$

and we take T_{sep} to be the smallest such T. A pair is invisible, and so hides in the pure set, if it is never unhit in T trials, which happens with probability $(1-p)^T$ per pair; a union bound over

the w pairs gives the visibility requirement

$$w(1-p)^T \;\le\; 0.01,$$

with T_{vis} the smallest such T. Clustering runs $T \;=\; \max(T_{\text{sep}}, T_{\text{vis}})$.

3.4.3 Empirical Validation

Clustering was validated at several points, where the truth was known from the key and every trial could be checked against it.

These parameters include the actual ID-0 parameters.

Scheme Parameters (q, n, k, w)	Attack Parameters $(n', (s, d))$	Trials	Flagging exact	Pairs recovered
$(256, 198, 165, 33)$, $(90, 2, 3)$	23/23	33/33		
$(256, 119, 79, 31)$, $(101, 2, 3)$	25/25	31/31		
$(1024, 630, 470, 160)$, $(366, 2, 3)$	41/41	160/160		

Table 3: Cluster validation at reduced parameters. "Flagging exact" means the flagged set equalled the unhit pairs' columns in that many trials.

ID-0, end to end. The shipped ID-0 set has since been run end to end with cluster Phase 1, on a fresh key generated with the NIST DRBG key generation of the implementation, at the adopted point $(n', k', (s, d)) = (366, 46, (2, 3))$, with 41 trials on 12 threads. The run succeeded completely: cluster aggregation over the 41 trials returned 160 pairs, 470 pure columns, 0 orphans, and 0 oversized groups; Phase 2 unmixed 160/160 pairs, Sidelnikov–Shestakov recovered the correct GRS code, and verification passed with 160/160 non-degenerate pairs correct and 0 false positives. The run took a little over 3 hours of wall time across those 12 threads to break ID-0. That is the Phase 1 trial time dominating the run: 41 trials in batches of 12, each trial a single reuse elimination, with aggregation, Phase 2 (SS bootstrap 0.25 s), and verification taking minutes by comparison. ID-1 and ID-2 have not been run end to end with cluster Phase 1; their figures below remain model estimates.

4 The GIJS Distinguisher In the Our Context

In this section we analyze the performance of the GIJS distinguisher [GIJS26] in our attack algorithms.

We now show that under plausible and empirically validated heuristics, the GIJS distinguisher algorithms succeeds in distinguishing clean columns (shortened from a pure GRS column) from dirty columns (one of two elements of a twin pair, both of which appear in the shortened code) under certain conditions even with noisy RLCE columns.

The shortened clean columns lie on a projected rational curve

$$\varphi : \mathbf{P}^1 \longrightarrow \mathbf{P}^{k'-1},$$

of degree $D = k' + w - 1$, given by homogeneous polynomials of degree D in the decoding parameter t. Below, we work over the algebraic closure $\overline{\mathbf{F}}_q$ when counting roots.

Each original column has a distinct evaluation parameter $\alpha \in \mathbf{F}_q$, and this distinct parameter is unchanged by shortening.

Fix a holdout column and let I be the set of interpolation columns, $|I| = n' - 1$. Partition I as follows:

- *Clean points:* kept pure columns and singleton-hit survivors. Each lies on the curve $\varphi(\mathbf{P}^1)$, at the parameter of its original column. Let $\mathcal{C}$ denote this set, and $C = |\mathcal{C}|$.
- *Dirty points:* kept columns belonging to a pair both of whose members are kept. Such a pair is called *both-kept*. Let $\mathcal{U}$ denote the set of non-degenerate both-kept pairs, and $U = |\mathcal{U}|$. Non-degenerate means that neither mixing coefficient is zero, so neither member of the pair is itself a curve point; in the RLCE specification the mixing coefficients are chosen to always be non-degenerate.

Thus, counting kept columns, $n' = C + 2U$ plus any degenerate both-kept pairs, which we exclude from $\mathcal{U}$ and treat separately; in the ID-0 run below there are none.

Let H be a homogeneous polynomial of degree d in k' variables. We say H *vanishes to order* s at a point x if all partial derivatives of H of order $< s$ vanish at x, i.e. the jet of H at x of order $s-1$ is zero. The GIJS kernel, for a fixed interpolation set, is the space of such H vanishing to order s at every point of I. A clean holdout τ is accepted, i.e. not flagged, if every H in the kernel vanishes at τ; it is flagged if some H in the kernel does not vanish at τ.

4.1 PIR persistence: rigorous acceptance with noise

First, we show that the pairs of dirty columns (which are the only type of non-clean columns present in the shortened code) are actually useful to the GIJS attack. As we will show, the reason is precisely due to the linear relation existing between pairs that results in a pure GRS column when inverting the A map from public key generation.

Theorem 3 (PIR persistence)**.** *Let H be a homogeneous polynomial of degree d vanishing to order s at every interpolation point, and assume*

$$2s > d. \tag{4}$$

Then H vanishes identically on the line through the two dirty columns of every non-degenerate both-kept pair, and in particular H vanishes at the underlying GRS point of that pair. Consequently, if

$$sC + U > dD, \tag{5}$$

then H vanishes identically on the curve $\varphi(\mathbf{P}^1)$.

As a result, it vanishes at every clean holdout column, meaning that running the GIJS distinguisher with a clean holdout column outputs "GRS" deterministically if the above conditions are fulfilled.

Proof. Fix a non-degenerate both-kept pair with underlying GRS point P and random point R. After shortening, its two columns are

$$Q_1 = aP + cR, \qquad Q_2 = bP + dR,$$

in projective coordinates, with $ad - bc \neq 0$ and $c, d \neq 0$ by non-degeneracy. Hence $Q_1 \neq Q_2$ in $\mathbf{P}^{k'-1}$, and Q_1, Q_2, P are collinear on the line L spanned by P and R; L is not contained in the curve, since R is not a curve point.

Restrict H to L. The restriction is a binary form of degree d on L, vanishing to order s at the two distinct points Q_1 and Q_2. A binary form of degree d with two distinct roots of multiplicity s each is identically zero as soon as $2s > d$, by degree counting. This is (4). Hence H vanishes identically on L, and in particular $H(P) = 0$.

Now consider the pullback $h(t) = H(\varphi(t))$. This is a binary form of degree dD, unless it is identically zero, which is the conclusion we want. We count its roots over $\overline{\mathbf{F}}_q$, with multiplicity.

First, let t_0 be the parameter of a clean interpolation point $X = \varphi(t_0)$. Since H vanishes to order s at X in ambient space, its Taylor expansion at X has no terms of degree $< s$. Composing with $\varphi(t) - X$, which has zero constant term at t_0, gives a power series in $t - t_0$ with no terms of degree $< s$; if φ is ramified at t_0, the order is only larger. Hence h vanishes to order at least s at t_0.

Second, let t_i be the parameter of the underlying GRS point P_i of a non-degenerate both-kept pair. By the line argument above, $H(P_i) = 0$, so $h(t_i) = 0$, i.e. h has a root at t_i of order at least 1. The parameters t_i are distinct across pairs, since the original evaluation parameters are distinct, and they are distinct from the clean parameters, since a both-kept pair's parameter is not the parameter of any clean interpolation column. Hence the roots counted in the first and second steps are distinct, except that a clean root already carries multiplicity s.

Therefore h has total root multiplicity at least $sC + U$. If (5) holds, this exceeds the degree dD of h, so h cannot be a nonzero binary form. Hence $h \equiv 0$, i.e. H vanishes at every point of the curve $\varphi(\mathbf{P}^1)$. A clean holdout column is a curve point, so every kernel polynomial H vanishes at it, and the GIJS test accepts it. □

4.2 Random Holdout Columns

We now show that with high probability (and heuristically with overwhelming probability) that a random and independent column will be flagged by the GIJS distinguisher, meaning that at least one kernel polynomial H will not vanish at it.

Theorem 4 (Independent uniform random holdout)**.** *Fix the interpolation set I, and let K be its GIJS kernel as above. Assume $K \neq \{0\}$. Let τ be uniform in $\mathbf{F}_q^{k'}$, independent of I and hence independent of K. Then*

$$\Pr[\tau \text{ is flagged}] \geq 1 - \frac{d}{q}. \tag{6}$$

Proof. Since $K \neq \{0\}$, pick any nonzero $H_0 \in K$. The flagging event contains the event $\{H_0(\tau) \neq 0\}$, because if H_0 does not vanish at τ, the GIJS test has a witness and flags τ. Hence

$$\Pr[\tau \text{ is flagged}] \geq \Pr[H_0(\tau) \neq 0] = 1 - \Pr[H_0(\tau) = 0].$$

It remains to bound $\Pr[H_0(\tau) = 0]$. This is the Schwartz–Zippel bound: a nonzero polynomial of total degree d in k' variables over $\mathbf{F}_q$ vanishes at at most a d/q fraction of points in $\mathbf{F}_q^{k'}$. Indeed, the bound follows by induction on k', writing H_0 as a polynomial in one variable with coefficients in the remaining variables; a univariate polynomial of degree at most d has at most d roots unless it is identically zero. Applying it to H_0 gives $\Pr[H_0(\tau) = 0] \leq d/q$, and (6) follows. □

4.2.1 Non-zero Kernel Conditions

It remains to discuss conditions on which the kernel is non-zero.

Let $J = \binom{k'+s-1}{s-1}$ be the number of jet conditions, and $N_d = \binom{k'+d-1}{d}$ the number of monomials of degree d in k' variables.

Corollary 1 (Sufficient condition for a nonzero kernel)**.** *If*

$$(n' - 1)J < N_d, \tag{7}$$

then $K \neq \{0\}$ for every interpolation set, and hence (6) holds for every trial.

Proof. K is the kernel of an $(n'-1)J \times N_d$ matrix. Its rank is at most the number of rows, $(n'-1)J$. By rank–nullity,

$$\dim K \geq N_d - (n'-1)J > 0$$

under (7), so $K \neq \{0\}$. Theorem 3 then applies. □

4.2.2 Heuristic Bounds on Dirty Columns

As our experiments (see below) show, we have not had *any* dirty columns fail to flag (i.e. the GIJS distinguisher outputs "GRS" when they are a holdout column).

Under a reasonable heuristic and using the $N_d-(n'-1)J$ lower bound for the kernel dimension, it seems reasonable to assume that the probability of failure in practice is more on the order of $(d/q)^{\dim K}$, meaning the probability is essentially independent across kernel polynomials. We stress that this is a pure heuristic, but that it has been experimentally shown to be reasonable on parameter sets up to the lowest set given in the RLCE specification.

5 Phase 2: unmixing three pairs

Phase 1 hands us two things: the twin pairings, and the pure columns. However, this is insufficient for Sidelnikov–Shestakov [SS92], as that requires $k+3$ pure columns, and at the $w = n-k$ parameters, Phase 1 finds exactly k pure columns.

A trivial method of finding the pairs would be to compute linear combination $c_{1,i} + a_i * c_{2,i}$ for $a \neq 0$ for each of 3 pairs $(c_{1,i}, c_{2,i})$ for which we wish to recover pure columns. Exactly 1 of the $(q-1)^3$ possibilities of choices of 3 a_i will successfully unmix all the 3 columns, enabling the Sidelnikov–Shestakov attack to succeed.

Instead, we use some clever ideas to *solve* for the triple. It costs only about q^2 cheap steps.

5.1 Putting the pairs in systematic form

We begin with the matrix

$$G = [P \mid c_{1,1} \mid c_{1,2} \mid c_{2,1} \mid c_{2,2} \mid c_{3,1} \mid c_{3,2}]$$

where P is the $k \times k$ matrix of the pure columns. After a left multiply of P^{-1}, the generator looks like $G = [I_k \mid \mathbf{g}_1\ \mathbf{g}_2\ \mathbf{g}_3]$, where for pair i with public columns $c_{i,1}, c_{i,2}$ we set $\mathbf{a}_i = P^{-1}c_{i,1}$, $\mathbf{b}_i = P^{-1}c_{i,2}$, and

$$\mathbf{g}_i(\lambda_i) \;=\; \mathbf{a}_i + \lambda_i \mathbf{b}_i, \qquad \lambda_i \in \mathbb{P}^1(\mathbb{F}_q),$$

with $\mathbf{g}_i(\infty) := \mathbf{b}_i$.

5.2 The Test

The dual $D(\lambda)$ is generated by the rows of

$$H \;=\; \left[\, -\mathbf{g}_1^T \;;\; -\mathbf{g}_2^T \;;\; -\mathbf{g}_3^T \;\middle|\; I_3 \,\right]$$

For $j \in [k]$ write $\gamma_j = (\mathbf{g}_1[j], \mathbf{g}_2[j], \mathbf{g}_3[j]) \in \mathbb{F}_q^3$ for the j-th column of the left block, and

$$\mathbf{m}_j \;=\; \big(\gamma_{j1}\gamma_{j2},\ \gamma_{j1}\gamma_{j3},\ \gamma_{j2}\gamma_{j3}\big) \in \mathbb{F}_q^3, \qquad M^*(\lambda) := \big[\mathbf{m}_1 \mid \cdots \mid \mathbf{m}_k\big] \in \mathbb{F}_q^{3\times k}.$$

In words, $\mathbf{m}_j$ is just the three pairwise products of the entries of γ_j. The test is about the rank of this $3 \times k$ matrix.

Lemma 2 (Dual-square characterization)**.** *Assume $k \geq 2$. Then*

$$\dim D(\lambda)^2 \;=\; 3 + \operatorname{rank} M^*(\lambda),$$

and $C(\lambda)$ is a GRS code with distinct support if and only if

(i) $\operatorname{rank} M^*(\lambda) = 2$*;*

(ii) the (then unique up to scale) vector $\mathbf{c} \in \mathbb{F}_q^3$ *with* $\mathbf{c}\, M^*(\lambda) = 0$ *satisfies* $c_{12}c_{13}c_{23} \neq 0$*;*

(iii) the $k+3$ columns of H are pairwise non-proportional.

Proof. Pairwise products of the three generators of $D(\lambda)$ span $D(\lambda)^2$. The evaluation map sending a quadric $Q \in \mathrm{Sym}^2(\mathbb{F}_q^3)$, a space of dimension 6, to the vector of its values on the columns of H has as its kernel exactly the quadrics vanishing at all $k+3$ columns of H, viewed as points of $\mathbb{P}^2$. Vanishing at $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ forces the squared coefficients of Q to be zero, so every quadric in the kernel looks like $Q_{\mathbf{c}} = c_{12}X_1X_2 + c_{13}X_1X_3 + c_{23}X_2X_3$, and $Q_{\mathbf{c}}(\gamma_j) = \mathbf{c} \cdot \mathbf{m}_j$. Hence the kernel has dimension $3 - \operatorname{rank} M^*$, and $\dim D(\lambda)^2 = 6 - (3 - \operatorname{rank} M^*) = 3 + \operatorname{rank} M^*$, in any characteristic.

If $C(\lambda)$ is GRS with distinct support, so is its dual $D(\lambda)$ [MS77], a $\mathrm{GRS}_{k+3,3}$ whose square is the evaluation code of degree at most 4 polynomials, of dimension $\min(5, k+3) = 5$; hence $\operatorname{rank} M^* = 2$. The columns of H are the support points of $D(\lambda)$, hence pairwise non-proportional, and they all lie on a smooth conic. That conic is a quadric through $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$, so it is $Q_{\mathbf{c}}$ for the unique relation $\mathbf{c}$, and smoothness is exactly (ii): if, say, $c_{12} = 0$ then $Q_{\mathbf{c}} = X_3(c_{13}X_1 + c_{23}X_2)$ is a union of two lines, whereas if all $c_{ij} \neq 0$ the gradient system in characteristic 2 has the unique candidate singular point, the nucleus $\nu = (c_{23}, c_{13}, c_{12})$, and $Q_{\mathbf{c}}(\nu) = c_{12}c_{13}c_{23} \neq 0$, so ν is not on the conic and the conic is smooth. While we do not need it for the purposes of this paper, the same conclusion in odd characteristic follows from the determinant $c_{12}c_{13}c_{23}/4 \neq 0$.

Conversely, given (i)–(iii), all $k+3$ columns of H lie on $Q_{\mathbf{c}}$, which is a smooth conic by (ii), and a 3-dimensional code whose generator columns are pairwise distinct points of a smooth conic is a GRS_3 code, by parametrizing the conic by $\mathbb{P}^1$. So $D(\lambda)$ is GRS and $C(\lambda) = D(\lambda)^{\perp}$ is GRS with distinct support by (iii). □

5.3 Solving for the triple, instead of trying it

Now, we present our root-finding idea. Fix a triple $J = \{j_1, j_2, j_3\} \subset [k]$ of coordinate positions (for simplicity we may take $(0, 1, 2)$), and let

$$\Delta_J(\lambda) \;=\; \det\big[\, R_{ab}(j_t) \,\big]_{(a,b) \in \{(1,2),(1,3),(2,3)\},\; t \in \{1,2,3\}}$$

be the corresponding 3×3 minor of $M^*(\lambda)$, where $R_{ab}(j) = \mathbf{g}_a[j]\, \mathbf{g}_b[j]$. Each R_{ab} is a product of two affine forms, so every term of the determinant has degree at most 2 in each λ_i: Δ_J has multidegree at most $(2, 2, 2)$. By the lemma, Δ_J vanishes at the true triple λ^*, and indeed every 3×3 minor of $M^*(\lambda^*)$ does, since $M^*(\lambda^*)$ has rank 2.

Fix (λ_1, λ_2) and call that a *cell*. The row $(R_{12}(j))_{j \in J}$ does not involve λ_3; call it $\mathbf{e}$. The other two rows are affine in λ_3: $(R_{13}(j))_{j \in J} = \mathbf{p} + \lambda_3 \mathbf{q}$ and $(R_{23}(j))_{j \in J} = \mathbf{r} + \lambda_3 \mathbf{s}$. Multilinearity of the determinant in the last two rows gives

$$\Delta_J \;=\; A\,\lambda_3^2 + B\,\lambda_3 + C,$$

with A, B, C each a 3×3 determinant in $\mathbf{e}, \mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}$. So at each cell, the admissible λ_3 are just the roots of one explicit quadratic, costing about 150 field operations per cell, independent of k. Solving

that quadratic over $\mathbb{F}_{2^m}$ is closed-form: if $A = 0$ it is linear (or constant); if $A \neq 0$ and $B = 0$ the unique root is $\lambda = (C/A)^{2^{m-1}}$; if $A, B \neq 0$, substituting $\lambda = (B/A)\mu$ gives $\mu^2 + \mu = AC/B^2$, solvable iff $\mathrm{Tr}(AC/B^2) = 0$, with roots $\mu, \mu + 1$. Heuristically, we have observed that the number of roots over all q^2 cells is about q^2 in expectation (with 68,406 observed in a test at $q = 256$).

The rest is a cascade, and each stage is cheaper than the last in the only sense that matters, the number of survivors. **Stage 1** computes the roots at every cell. **Stage 2** evaluates a second minor $\Delta_{J'}$, for a disjoint triple J', at each root (about 60 operations), leaving about q survivors. **Stage 3** applies the exact test of Lemma 2 to each survivor ($O(k)$ operations each). **Stage 4** rebuilds each surviving candidate in original coordinates and runs SS, accepting only a triple that also passes the attack's deterministic full-key verification. The remaining $w - 3$ pairs are then unmixed against the recovered curve, and the key completed, exactly as in the original Phase 2 (including the CLT post-processing [CLT18]); none of that is changed.

Algorithm 4: Root-finding unmixing (one orientation)

Input: The k pure columns; three pairs $(c_{i,1}, c_{i,2})$; disjoint triples $J, J' \subset [k]$.
Output: A verified unmixing triple, or failure.
$P \leftarrow$ pure-column matrix; compute P^{-1} and $\mathbf{a}_i, \mathbf{b}_i$ for $i = 1, 2, 3$;
for $(\lambda_1, \lambda_2) \in \mathbb{F}_q^2$ **do**
  Form $\mathbf{e}, \mathbf{p}, \mathbf{q}, \mathbf{r}, \mathbf{s}$ on J; compute A, B, C;
  if $A = B = C = 0$ **then**
    $\mathcal{R} \leftarrow \{\lambda_3 \in \mathbb{F}_q : \Delta_{J'}(\lambda_1, \lambda_2, \lambda_3) = 0\}$; `// degenerate cell: scan`
  else
    $\mathcal{R} \leftarrow \{\lambda_3 : \Delta_{J'}(\lambda_1, \lambda_2, \lambda_3) = 0,\ \lambda_3$ a root of $A\lambda_3^2 + B\lambda_3 + C\}$;
  for $\lambda_3 \in \mathcal{R}$ **do**
    if $(\lambda_1, \lambda_2, \lambda_3)$ *passes Lemma 2 and SS + verification on original coordinates* **then**
      return $(\lambda_1, \lambda_2, \lambda_3)$;
return failure

6 Experimental Costs for Key Recovery

Experiments were run on a single-socket, single-NUMA-node x86_64 machine with an Intel Core Ultra 7 265H processor (family 6, model 197, stepping 2): 16 cores and 16 hardware threads (one thread per core), with a maximum frequency of 5.3 GHz, 30 GiB of RAM and 8 GiB of swap, a 24 MiB shared L3 cache, 28 MiB of L2 cache in total, and 480 KiB of L1 data cache and 768 KiB of L1 instruction cache in total.

All experiments below were successful in key recovery. Code will be available on Github shortly if the author gets around to it.

6.1 Parameter Sets Tested

In addition to the ID-0 (128-bit) parameter set from the RLCE specification, we also tested several smaller parameter sets, and present them and name them here to avoid cluttering the timing table below; all sets below have $w = n - k$.

6.2 Timing tables

Below, phase 1 refers to the total time taken summed across threads.

Table 4: Parameter Sets Tested

Name	q	n	k	n'
256	256	198	165	90
512	512	500	375	293

Table 5: Timing Results

Set	Trials	Threads	Phase 1 Time	Phase 2 Time
256	50	10	288.27 (s)	0.02 (s)
512	44	12	37278.15 (s)	0.09 (s)
ID-0	41	12	109673.21 (s)	0.23 (s)

References

[ABNS21] Paulo Almeida, Miguel Beltrá, Diego Napp, and Cláudia Sebastião. Smaller keys for the McEliece cryptosystem: A convolutional variant with GRS codes. *arXiv preprint arXiv:2104.06809*, 2021.

[BBC13] Marco Baldi, Giovanni Bianchi, and Franco Chiaraluce. A new cryptosystem based on McEliece with generalized Reed-Solomon codes. *Des. Codes Cryptogr.*, 69(3):373–386, 2013.

[CLT18] Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Tillich. Recovering short secret keys of RLCE in polynomial time. *IACR Cryptol. ePrint Arch.*, 2018:528, 2018.

[Fed26] Federal Office for Information Security (BSI). Notes on recent developments concerning classic mceliece. Technical report, Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik), Bonn, Germany, sep 2026. Accessed: October 14, 2026.

[GIJS26] Ashrujit Ghoshal, Yuval Ishai, Aayush Jain, and Nuozhou Sun. Quasipolynomial cryptanalysis of the mceliece cryptosystem (or: PIR meets mceliece). *IACR Cryptol. ePrint Arch.*, 2026:1630, 2026.

[Gre26] Matthew Green. I think we might lose public key cryptography. X post, October 2026.

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.

[Kro20] Evgenii A. Krouk. New code-based cryptosystem based on binary image of generalized Reed-Solomon code. In *XVII International Symposium on Problems of Redundancy in Information and Control Systems*, pages 66–69, 2020.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014.

[McE78] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report 42-44*, pages 114–116, 1978. Pasadena, California.

[Moo19] Dustin Moody. Round 1 PQC status, 2019. PQCrypto 2019 NIST presentation.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[Nat19] National Institute of Standards and Technology. Status report on the first round of the NIST post-quantum cryptography standardization process. Technical Report NISTIR 8240, NIST, 2019.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control. Inf. Theory*, 15(2):159–166, 1986.

[Saa26] Markku-Juhani O. Saarinen. Chinese NGCC algorithms: The first week of AI cryptanalysis. Cryptology ePrint Archive, Paper 2026/2266, 2026.

[SS92] Vladimir M. Sidelnikov and Serguei O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[SW26] Zygimantas Straznickas and Stephen A. Weis. HAWK-n key recovery reduces to SVP in dimension $n/2 + 1$. Cryptology ePrint Archive, Paper 2026/1593, 2026.

[Wan] Yongge Wang. RLCE specification and implementation.

[Wan15] Yongge Wang. Quantum resistant random linear code based public key encryption scheme RLCE. *CoRR*, abs/1512.08454, 2015.

[Wan16] Yongge Wang. Quantum resistant random linear code based public key encryption scheme RLCE. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 2519–2523. IEEE, 2016.

[Wan26] Yongge Wang. Security of RLCE against mceliece attacks. NIST PQC-Forum Mailing List, October 2026. Posted October 1, 2026, 8:35 PM. Cleaned text: "I do not think these attacks on McEliece will work on RLCE...".

[Wei26] Stephen A. Weis. Improving GIJS key recovery for classic McEliece. Cryptology ePrint Archive, Paper 2026/1984, 2026.

[Wie09] Christian Wieschebrink. Cryptanalysis of the McEliece cryptosystem based on Goppa codes. In *Post-Quantum Cryptography - Third International Workshop, PQCrypto 2009*, volume 5628 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2009.

[Wie16] Christian Wieschebrink. Variations of the McEliece cryptosystem with GRS codes and weight 2 masking matrices. In *Workshop on Coding and Cryptography, WCC 2016*, 2016.

AI Usage Statement

Muse AI (Meta) was used in the preparation of this paper for proofs, calculations, and writing, and the creation of the faster non-trivial Phase 2 as well as, with careful guidance from the author, the reuse idea to speed up Phase 1.

The other key ideas and insights, including that the Ghoshal–Ishai–Jain–Sun distinguisher can tolerate noisy rows, the shortening and counting appearances of twin pairs idea for twin pair recovery are the author's.

The author takes full responsibility for the contents.