On Arithmetic Invariants for Permutation Equivalence

Benjamin Benčina*

Royal Holloway, University of London
benjamin.bencina.2022@live.rhul.ac.uk

Abstract. We give an efficiently computable invariant we call the square class invariant for the (Signed) Permutation Code Equivalence ((S)PCE) problem, that was previously not recognised in coding theory, which arises from the arithmetic of quadratic forms over finite fields. Our invariant naturally yields a distinguisher for the decision version of (S)PCE as defined at Eurocrypt 2025 by Albrecht, Benčina and Lai [ABL25], breaking the hardness assumption that underpins the security of their updatable public-key encryption scheme.

We extend a 2023 result by Bruin, Ducas and Gibbons by fully classifying the genera of q-ary lattices, showing the genus is completely determined by the hull dimension and our square class invariant of the underlying linear code, and that none of these genera split non-trivially into spinor genera (as soon as the lattice dimension is at least 5), implying the genus of a q-ary lattice encodes all known efficiently computable coding-theoretic invariants for (S)PCE and vice versa. Thus our distinguisher can also be seen as comparing the genera of Construction A lattices of the (S)PCE instance in the spirit of the Lattice Isomorphism Problem. We give a complete description of the genus distribution of uniform q-ary lattices.

This motivates the definition of a genus of a linear code as the genus of the Construction A lattice of any of its generator matrices. We adapt the sampling algorithm from [ABL25] to sample from a single genus uniformly at random and can thus restrict their hardness assumption for (S)PCE to a single genus. Restricting PCE to one genus and using our sampling algorithm is then used with a slight modification to the security proof to mend the scheme from [ABL25].

1 Introduction

A key problem in the study of equivalence problems such as the Lattice Isomorphism Problem (LIP) [DvW22, BGPS23], or the many flavours of Code Equivalence [SS13] is determining efficiently computable invariants with respect to the equivalence relation. If any of the invariants of the instance do not match then the decision version of the equivalence problem is easy to answer in the negative. In the case of LIP in the quadratic form setting, a set of such invariants was captured by Ducas and van Woerden [DvW22] in the so-called arithmetic fingerprint of a quadratic form, which besides the determinant most notably contains the *genus* of the quadratic form. The genus is, essentially, a finite collection of numbers that uniquely determine quadratic form equivalence classes over the p-adic extension ring $\mathbb{Z}_p$ of $\mathbb{Z}$ for every relevant prime p. According to the weak Hasse local-global principle [CS99, Ch. 15.5.1 Thm. 4], two quadratic forms are equivalent over $\mathbb{Q}$ if and only if they are equivalent over $\mathbb{R}$ and locally at every prime p, which for integral forms is implied by $\mathbb{Z}_p$-equivalence. Since the genus is efficiently computable [CS99, Ch. 15.7], it is an invariant that must be matched for LIP to be hard to decide – if two forms are not equivalent over $\mathbb{Q}$ then they cannot be equivalent over $\mathbb{Z}$. While invariants such as the determinant of the quadratic form are relatively easy to match when sampling an instance, the same is not necessarily true for the genus. This motivates the study of the genus distribution conditional on specific distributions of useful families of lattices [BDG23], lattice properties across genera [van24] and invariant theory broadly.

A particularly useful class of lattices are q-ary lattices, these are lattices $\mathcal{L}$ such that $q \cdot \mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, where n is the dimension of $\mathcal{L}$. They feature in commonly used lattice problems such as LWE [Reg05] and SIS [Ajt96] and are normally constructed from generator matrices of linear codes through the so-called

* This research was supported by UKRI Grant EP/S021817/1.

Construction A,[1] which takes a code generator matrix $\mathbf{A} \in \mathbb{F}_q^{k \times n}$ and (informally) constructs from it the lattice $\Lambda_q(\mathbf{A}) = \mathbb{Z}^k \cdot \mathbf{A} + q \cdot \mathbb{Z}^n$. Possible genera and their distribution conditioned on the uniform distribution of $\mathbf{A}$ were first studied in [BDG23], for the particular case when $\mathbf{A}$ generates an LCD code [Mas92]. The connection of the hardness of LIP over q-ary lattices with the underlying coding-theoretic structure and (Signed) Permutation Code Equivalence ($(\mathsf{S})\mathsf{PCE}$) was then made in [DG23], and worst-case reductions from LCE and SPCE to LIP on q-ary lattices were given in [BW25]. Later, the authors of [ABL25a] studied *hollow lattice problems*, these are lattice problems on q-ary lattices whose underlying code has a particularly large hull, which they use in their updatable public-key encryption (UPKE) scheme to argue security from the hardness of PCE. They also propose the first algorithm that samples uniformly random $[n, k]$-linear codes over $\mathbb{F}_q$ with a prescribed hull dimension, which allows them to define $\Delta(\mathsf{S})\mathsf{PCE}$ as an average-case problem.

1.1 Our Contributions

The principal idea of this work is simple: if $(\mathbf{A}, \mathbf{B})$ is an instance of SPCE over $\mathbb{F}_q$, then (i) $(\mathbf{A} \cdot \mathbf{A}^\intercal, \mathbf{B} \cdot \mathbf{B}^\intercal)$ is an instance of quadratic form equivalence over the finite field $\mathbb{F}_q$, and (ii) $(\Lambda_q(\mathbf{A}), \Lambda_q(\mathbf{B}))$ is, at least in spirit, an instance of LIP (though this is not how LIP is usually defined or used). Thus we apply ideas from the arithmetic of quadratic forms commonly used in the study of LIP to $(\mathsf{S})\mathsf{PCE}$ through these mappings, in particular to decide the problem in the negative using arithmetic invariants. We first consider (i), which through Witt's theory yields a previously unrecognised invariant for $(\mathsf{S})\mathsf{PCE}$ on codes over $\mathbb{F}_q$ for odd q that are not self-orthogonal, allowing us to mount a distinguishing attack on the scheme from [ABL25a]. We then consider (ii) and compute the genus of q-ary lattices over linear codes with any hull dimension, generalising the result of [BDG23]. A precise statement on the genus distribution of q-ary lattices then allows us to refine the hardness conjecture on $(\mathsf{S})\mathsf{PCE}$ and fix the security of the scheme from [ABL25a].

The square class invariant for $(\mathsf{S})\mathsf{PCE}$. We show in Lemma 4 that mapping a code generator matrix to its corresponding quadratic space $V_{\mathbf{A}} = (\mathbb{F}_q^k, \mathbf{A} \cdot \mathbf{A}^\intercal)$ and considering its isometry class yields a previously unrecognised efficient invariant $\chi_{\mathbf{A}}$ we call the square class, that arises as a generalised version of the discriminant of a quadratic form over $\mathbb{F}_q$. We show in Theorem 1 the square classes of two random codes with the same hull dimension match with probability negligibly close to $\frac{1}{2}$ in $(k - h) \cdot \log_2(q)$, where k and h are the code and the hull dimension, respectively. To the best of our knowledge, this is the first 'new' invariant for $(\mathsf{S})\mathsf{PCE}$ since [Sen00] showed that the hull dimension is an invariant.[2] We experimentally verify that, like the hull dimension, our square class invariant is not an invariant for LCE.

Breaking the hardness assumption of [ABL25a]. The authors of [ABL25a] combined folklore beliefs on the hardness of $(\mathsf{S})\mathsf{PCE}$ from [SS13, Prob. 1] and [DG23] to define $\Delta(\mathsf{S})\mathsf{PCE}$ as distribution problems. Denoting by $\mathsf{D}^h_{n,k,q}$ the uniform distribution on the set of all $[n, k]$-linear codes over $\mathbb{F}_q$ with hull dimension h, they conjecture that given $\mathfrak{C}_0 \leftarrow\!\!\$\ \mathsf{D}^h_{n,k,q}$ and $\mathfrak{C}_1 \in \mathrm{Supp}(\mathsf{D}^h_{n,k,q})$, it is hard to decide whether $\mathfrak{C}_1$ is a (signed) permutation of $\mathfrak{C}_0$ or an independent sample from $\mathsf{D}^h_{n,k,q}$, provided the hull dimension h is big enough. They then base the security of their scheme on this problem, denoted by $\Delta\mathsf{PCE}^h_{n,k,q}$, and provide a sampling algorithm for $\mathsf{D}^h_{n,k,q}$ to instantiate the scheme. Their conjecture, however, does not account for our square class invariant, thus we show it is *false* whenever $h < k$ (which is required by their sampler and reductions) and construct a simple adversary against it in Theorem 2. Our adversary, given an instance $(\mathbf{A}, \mathbf{B})$, simply checks whether $\chi_{\mathbf{A}} = \chi_{\mathbf{B}}$. We then use our Theorem 1 to show in Theorem 2 that such an adversary has non-negligible advantage and in fact (Corollary 4) succeeds in deciding $\Delta(\mathsf{S})\mathsf{PCE}^h_{n,k,q}$ with probability at least $\frac{3}{4} - \mathsf{negl}(\lambda)$ for all the parameters given in [ABL25a, Tab. 1].

[1] Recall kernel lattices in [Ajt96] are dual up to scaling to Construction A lattices.

[2] The statement [Sen00, Prop. 12] is of course stronger, the hulls of the two codes must also be (signed) permutation equivalent with the same isometry.

A local structure theorem for q-ary lattices. Our next contribution is Theorem 3, a generalisation of [BDG23, Lem. 7] that uses square classes to precisely describe the genus of any q-ary lattice, not just those coming from LCD codes. For a q-ary Construction A lattice of a generator matrix $\mathbf{A}$ for an $[n,k]$-linear h-hollow code over the odd prime field $\mathbb{F}_q$, our theorem states that its q-adic symbol is

$$n_0 = k-h \qquad n_1 = 2h \qquad n_2 = n-k-h$$
$$\epsilon_0 = \chi_{\mathbf{A}} \qquad \epsilon_1 = \chi_q(-1)^h \qquad \epsilon_2 = \chi_{\mathbf{A}^\perp},$$

where χ_q is the quadratic character on $\mathbb{F}_q^\star$ and $\mathbf{A}^\perp$ generates the dual code. This implies (Corollary 7) that for a fixed length n, dimension k, and odd prime field $\mathbb{F}_q$, each hull dimension $h < k$ admits exactly two genera that we denote by $\mathcal{G}^{\pm 1,h}_{n,k,q}$, the membership for which is decided precisely by our square class $\chi_{\mathbf{A}} \in \{\pm 1\}$. All self-orthogonal codes (with $h = k$) give lattices that lie in a single genus denoted by $\mathcal{G}^{k}_{n,k,q}$, for a total of $2k+1$ genera. We also show in Corollary 9 that none of these genera split further into multiple spinor genera [Eic74], a finer notion of equivalence that is between integral and local equivalence, as soon as $n \geq 5$. This shows our structure theorem indeed exhausts the known local information available and indicates there are likely no further invariants for (S)PCE to obtain using arithmetic.

The genus distribution of random q-ary lattices. We follow the authors of [BDG23] a bit further and consider the distribution of genera for two particular distributions, these are $\mathsf{D}^{h}_{n,k,q}$ and $\mathcal{U}\left(\mathbb{F}_q^{k\times n}\right)$. In Corollary 10, we restate Theorem 1 for q-ary lattices and observe that the relative difference in size of $\mathcal{G}^{\pm 1,h}_{n,k,q}$ for fixed n,k,h,q is at most negligible in $(k-h)\cdot\log_2(q)$. In particular, this improves on [BDG23, Thm. 2], which states that for 0-hollow lattices (those from LCD codes), the relative difference in size of the genera $\mathcal{G}^{\pm 1,0}_{n,k,q}$ is bounded by $\mathcal{O}\left(q^{-3}\right)$, whereas our result implies $\mathcal{O}\left(q^{-\frac{k}{2}}\right)$. Using the mass formulae in [LSL25], this allows us to give precise probabilities in Corollary 11 that the Construction A lattice of a uniformly random full-rank $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ lies in either of the $2k+1$ genera.

Refined hardness conjectures. The above motivates us to consider the genus of a linear code, which we define as the genus of the Construction A lattice of any of its generator matrices. For $\chi \in \{\pm 1\}$ we define the distribution $\mathsf{D}^{\chi,h}_{n,k,q}$ as the uniform distribution on the set of generator matrices of codes in $\mathcal{G}^{\chi,h}_{n,k,q}$. Using the probabilities in Corollary 10, we turn the algorithm from [ABL25a, Def. 11] that samples from $\mathsf{D}^{h}_{n,k,q}$ into an algorithm that samples from $\mathsf{D}^{\chi,h}_{n,k,q}$ at the cost of roughly halving the success probability. We then refine the conjecture on the hardness of $\Delta(\mathsf{S})\mathsf{PCE}^{h}_{n,k,q}$ in Conjecture 1 by restricting it to one of the two possible genera, which thwarts our distinguisher, and denote it by $\Delta(\mathsf{S})\mathsf{PCE}^{\chi,h}_{n,k,q}$. Using our genus sampler, it is immediate that the security of the scheme from [ABL25a] can be based on $\Delta\mathsf{PCE}^{\chi,h}_{n,k,q}$, with the rest of the proof unchanged, which we show in Lemma 6.

1.2 Comparison to Related Work

There is a rich history of attacks on different variants of code equivalence; some rely on finding codewords of low Hamming weight with colliding multisets [Leo82,Beu20,CPS23,Now25] and typically work regardless of the hull dimension (often geared more broadly towards linear equivalence (LCE)), whereas others [Sen00,SS13,BOST19] exploit the fact that the hull of a random code is most likely trivial or very small [Sen97]. The latter attacks are exponential in the hull dimension but polynomial in every other parameter, leading the authors of [ABL25a] to consider codes with a sufficiently large hull such that the aforementioned attacks are not efficient.

Recently a new attack [BMS26] appeared that exploits the fact that the hull dimensions proposed in [ABL25a, Tab. 1] satisfy $h < \sqrt{2n}$ for n the code length. Thus we can instead consider a (S)PCE instance of *square codes*, which typically have a trivial hull, and use one of the attacks above to find the secret permutation. The issue can be fixed by increasing the hull rate, requiring $h \gg \sqrt{2n}$ (with some margin), which the reductions from [ABL25a] can accommodate with no increase in concrete sizes (say, by setting

$h = \frac{k}{2}$ for k the code dimension). We would like to emphasise that our distinguisher in Theorem 2 works just as well against [ABL25a] even if the parameters of the scheme in [ABL25a] account for [BMS26], since its success probability is (at least) negligibly close to $\frac{3}{4}$ in $k - h$, which must remain large by [ABL25a, Thm. 1], though in Corollary 4 we refer only to [ABL25a, Tab. 1] as a concrete example.

Another commonality of the above attacks, including [BMS26], is that they all target the search problem, whereas we only provide a distinguisher. It is natural to consider lifting our distinguisher to a search attack using a search-to-decision reduction, for example [BM23]. Besides the fact that [BM23] requires a distinguisher that always succeeds, whereas ours succeeds with probability about $\frac{3}{4}$, we later show in Section 5 that restricting to a genus accounts for our distinguisher completely and provide a sampler for such instances. In this sense, this work is less about the attack and more about how to avoid it through arithmetic considerations. Interestingly, the bound $h \gg \sqrt{2n}$ that thwarts [BMS26] also gives a quadratic lower bound on h for any n, thus we must include it in our refined hardness assumption in Conjecture 1, which is not the case for other attacks.

2 Preliminaries

For integers $n \leq m$ we denote $[\![n, m]\!] = [n, m] \cap \mathbb{Z}$, that is, the set of all integers between and including n and m. Denote by $\mathcal{P}_n$ the group of $n \times n$ permutation matrices, and for a commutative ring $\mathcal{R}$ let $\mathcal{O}_n(\mathcal{R}) = \{\mathbf{O} \in \mathcal{R}^{n \times n} \mid \mathbf{O} \cdot \mathbf{O}^\intercal = \mathbf{I}_n\}$. In particular, $\mathcal{O}_n(\mathbb{Z})$ is the group of signed permutation matrices. A function $f\colon X \to T$ is an *invariant* for an equivalence relation $\sim$ on X if $x \sim x'$ implies $f(x) = f(x')$ for all $x, x' \in X$. We focus on f that are efficiently computable. Tensor products are over $\mathbb{Z}$, logarithms have base 2 and $\omega \in [2, 3]$ denotes the linear algebra constant.

2.1 Finite Fields and p-adic Integers

Let q be a prime power and $\chi_q\colon \mathbb{F}_q^\star \to \{\pm 1\}$ the quadratic character $\chi_q(a) = a^{\frac{q-1}{2}}$, where $\mathbb{F}_q^\star$ is the group of multiplicative units in $\mathbb{F}_q$. The pre-images in $\chi_q^{-1}(1)$ are the quadratic residues ('squares') in $\mathbb{F}_q$, and the pre-images in $\chi_q^{-1}(-1)$ are the quadratic non-residues. For odd primes p, this is more commonly captured by the *Legendre symbol* at p that maps any integer $a \in \mathbb{Z}$ to $\left(\frac{a}{p}\right) = \chi_p(a \bmod p)$. The *square class* group $\mathbb{F}_q^\star / \left(\mathbb{F}_q^\star\right)^2$ of the field $\mathbb{F}_q$ for odd q consists of two elements $\{1, s\}$ where s is any non-residue in $\mathbb{F}_q$. We identify the square class of $a \in \mathbb{F}_q^\star$ with $\chi_q(a) \in \{\pm 1\}$ for convenience.[3]

We recommend [Ser73] as an introductory reference to p-adic integers. For a prime p, the ring $\mathbb{Z}_p$ consists of formal sums $a = \sum_{i=0}^{\infty} a_i \cdot p^i$ with $a_i \in \mathbb{F}_p$ called p-adic integers, and is equipped with the usual addition and multiplication. This is a local ring with the unique prime ideal $p \cdot \mathbb{Z}_p$, hence a is invertible if and only if $a_0 \neq 0$, that is, p does not divide a (otherwise $a \in p \cdot \mathbb{Z}_p$). The field $\mathbb{Q}_p$ of p-adic numbers is the field of fractions of $\mathbb{Z}_p$. Denote by $\nu_p\colon \mathbb{Z}_p \to \mathbb{N}_0 \cup \{\infty\}$ the *p-adic valuation*, defined as $\nu_p(0) = \infty$ and $\nu_p(a) = \min\{i \mid a_i \neq 0\}$ for $a \neq 0$. For a matrix $\mathbf{A} = [a_{i,j}]_{i,j}$ we define $\nu_p(\mathbf{A}) = \min_{i,j}\{\nu_p(a_{i,j})\}$. Algorithmically, p-adic integers are treated iteratively through operations in $\mathbb{F}_p$ by considering $a \bmod p = a_0 \in \mathbb{F}_p$, shifting $a' = \frac{a - a_0}{p^{\nu_p(a - a_0)}}$, and repeating (or the matrix analogue).

2.2 The Local-Global Principle

Denote by 1_p the element $1 \in \mathbb{Z}_p$. Two quadratic forms are equivalent over $\mathbb{Q}$ if and only if they are equivalent 'locally' over every $\mathbb{Q}_p$ for primes p, and over $\mathbb{R}$ (which is sometimes denoted as $\mathbb{Q}_\infty$ or $\mathbb{Q}_{-1}$) [CS99, Ch. 15.5.1 Thm. 4]. Two integral quadratic forms must also be equivalent over $\mathbb{Z}_p$ via extension of scalars $\mathbf{Q} \mapsto \mathbf{Q} \otimes 1_p$, which implies equivalence over $\mathbb{Q}_p$ and hence $\mathbb{Q}$. An integral quadratic form decomposes over $\mathbb{Z}_p$ as

$$\mathbf{Q} \otimes 1_p \cong \mathbf{Q}_0 \oplus p \cdot \mathbf{Q}_1 \oplus p^2 \cdot \mathbf{Q}_2 \oplus \cdots$$

[3] Though beware -1 is a non-residue only if $p = 3 \pmod 4$.

where each $\mathbf{Q}_i$ is either 0 or p-adic integral with determinant coprime to p, and can be assumed diagonal for odd p, though we will prefer to write it as a block-diagonal decomposition. This is called the *Jordan decomposition* of $\mathbf{Q}$ at p, and we call $\mathbf{Q}_i$ the i-th (or degree-i) Jordan component at p. If p does not divide $2\det(\mathbf{Q})$, the Jordan decomposition at p is trivial [Sie41, Lem. 5], so we only consider local information at p dividing $2\det(\mathbf{Q})$. We denote the *genus* of a quadratic form $\mathbf{Q}$ by $\mathcal{G}(\mathbf{Q})$ and further for a lattice $\mathcal{L}$ we denote by $\mathcal{G}(\mathcal{L})$ the genus of any of its Gram matrices, that is, two forms (of the same determinant) are in the same genus if they are equivalent locally at every prime p. Locally at odd primes p, two forms are equivalent if and only if their *p-adic symbols* match, that is the (finite) sequence of pairs $n_i = \dim(\mathbf{Q}_i)$ and $\epsilon_i = \left(\frac{\det(\mathbf{Q}_i)}{p}\right)$ (when $n_i = 0$ we write $\epsilon_i = 1$) called the dimension and the sign, respectively. Clearly, we have $\sum_{i=0}^{\infty} n_i = n$ for the dimensions, and for the signs we have the *determinant condition* [CS99, Ch. 15.7.7] that states $\prod_{i=0}^{\infty} \epsilon_i = \left(\frac{p^{-\nu_p(\det(\mathbf{Q}))} \cdot \det(\mathbf{Q})}{p}\right)$. Another useful fact is that we have

$$\det(\mathbf{Q} \otimes 1_p) = \prod_{n_i \neq 0} \det(p^i \cdot \mathbf{Q}_i) = p^{\sum_{i=0}^{\infty} i \cdot n_i} \cdot \prod_{n_i \neq 0} \det(\mathbf{Q}_i).$$

Since $\mathbf{Q}_i$ are unimodular over $\mathbb{Z}_p$, it follows that $\nu_p(\det(\mathbf{Q})) = \sum_{i=0}^{\infty} i \cdot n_i$. The p-adic symbol is efficiently computable (as is the Jordan decomposition) using the algorithm from [CS99, Ch. 15.7], implying that local equivalence of integral quadratic forms, that is genus membership, can be efficiently decided provided we can factor the determinant (which is always the case for instances of interest).

2.3 Linear Codes

Let $n, k, q \in \mathbb{Z}$ with $0 \leq k \leq n$ and q prime power. An $[n, k]$-linear code $\mathfrak{C}$ over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$. Denote by $\mathcal{X}_{n,k,q}$ the set of all $[n, k]$-linear codes over $\mathbb{F}_q$. A code is typically represented by a full-rank row-generator matrix $\mathbf{A} \in \mathbb{F}_q^{k \times n}$, denoted here by $\mathfrak{C}(\mathbf{A})$, the canonical choice for which is the unique generator matrix that is row-reduced. An *information set* of a code is a set of k coordinate indices such that the submatrix of the generator matrix at those coordinates has full rank. Every code can be permuted into one with a generator matrix of the form $[\mathbf{I}_k, \mathbf{M}]$ for some matrix $\mathbf{M}$, that is, $[\![1, k]\!]$ is its information set. The *dual code* of $\mathfrak{C}$ is defined as $\mathfrak{C}^{\perp} = \left\{\mathbf{y} \in \mathbb{F}_q^n \;\middle|\; \mathbf{y} \perp \mathfrak{C}\right\}$. If $[\mathbf{I}_k, \mathbf{M}]$ generates $\mathfrak{C}$, then $[-\mathbf{M}^{\intercal}, \mathbf{I}_{n-k}]$ generates $\mathfrak{C}^{\perp}$. The *hull* of $\mathfrak{C}$ is defined as $\mathfrak{C} \cap \mathfrak{C}^{\perp}$, and is a vector subspace of $\mathfrak{C}$ (and $\mathfrak{C}^{\perp}$). A code is called *h-hollow* if its hull has dimension h, and a full-rank matrix is called h-hollow if it generates an h-hollow code. Denote by $\mathcal{X}^h_{n,k,q}$ the set of all $[n, k]$-linear h-hollow codes over $\mathbb{F}_q$. A code is *self-orthogonal* if $\mathfrak{C} \subseteq \mathfrak{C}^{\perp}$, in which case $h = k \leq \frac{n}{2}$. Codes that are 0-hollow are called *linear complementary dual* (LCD) codes [Mas92] since then $\mathfrak{C} \oplus \mathfrak{C}^{\perp} = \mathbb{F}_q^n$. Observe that a full-rank matrix $\mathbf{A} \in \mathbb{F}_q^{k \times n}$ is h-hollow if and only if the Gram matrix $\mathbf{A} \cdot \mathbf{A}^{\intercal}$ has rank $k - h$, in particular, Gram matrices of LCD codes have full rank. Thus define a projector [Mas92, Prop. 1] to an LCD code generated by $\mathbf{A}$ by $\mathbf{P}_{\mathbf{A}} = \mathbf{A}^{\intercal} \cdot (\mathbf{A} \cdot \mathbf{A}^{\intercal})^{-1} \cdot \mathbf{A}$, satisfying symmetry $\mathbf{P}_{\mathbf{A}}^{\intercal} = \mathbf{P}_{\mathbf{A}}$ and idempotency $\mathbf{P}_{\mathbf{A}}^2 = \mathbf{P}_{\mathbf{A}}$.

Since the introduction of the so-called *hull attacks* [Sen00, SS13, BOST19] it has been conjectured that PCE is hard provided the hull dimension is large enough, indeed the complexity of each of these attacks is exponential in the hull dimension but polynomial in every other parameter (in fact involving only linear algebra). This was recently formalised by Albrecht, Benčina, and Lai [ABL25a].

Definition 1 (PCE [**ABL25a, Def. 1**]). *Let* D *be a distribution on the full-rank matrices in* $\mathbb{F}_q^{k \times n}$ *and let* $\mathbf{G} \leftarrow\!\$\; \mathsf{D}$. *Let* $\mathbf{H}_0 \leftarrow\!\$\; \mathsf{D}$, $\mathbf{H}_1 = \mathbf{U} \cdot \mathbf{G} \cdot \mathbf{P}$ *for* $\mathbf{P} \leftarrow\!\$\; \mathcal{P}_n$ *and* $\mathbf{U} \leftarrow\!\$\; \mathcal{GL}_k(\mathbb{F}_q)$, *and let* $b \leftarrow\!\$\; \{0, 1\}$.

- *The problem* $\Delta\mathsf{PCE}_{n,k,q}(\mathsf{D})$ *is to find* b *given* $(\mathbf{G}, \mathbf{H}_b)$.
- *The problem* $s\mathsf{PCE}_{n,k,q}(\mathsf{D})$ *is to find* $\mathbf{P}$ *given* $(\mathbf{G}, \mathbf{H}_1)$.

Definition 2 (SPCE [**ABL25a, Def. 2**]). *Let* D *be a distribution on the full-rank matrices in* $\mathbb{F}_q^{k \times n}$ *and let* $\mathbf{G} \leftarrow\!\$\; \mathsf{D}$. *Let* $\mathbf{H}_0 \leftarrow\!\$\; \mathsf{D}$, $\mathbf{H}_1 = \mathbf{U} \cdot \mathbf{G} \cdot \mathbf{O}$ *for* $\mathbf{O} \leftarrow\!\$\; \mathcal{O}_n(\mathbb{Z})$ *and* $\mathbf{U} \leftarrow\!\$\; \mathcal{GL}_k(\mathbb{F}_q)$, *and let* $b \leftarrow\!\$\; \{0, 1\}$.

- *The problem* $\Delta\mathsf{SPCE}_{n,k,q}(\mathsf{D})$ *is to find* b *given* $(\mathbf{G}, \mathbf{H}_b)$.

- *The problem* $s\mathsf{SPCE}_{n,k,q}(\mathsf{D})$ *is to find* $\mathbf{O}$ *given* $(\mathbf{G}, \mathbf{H}_1)$.

In Definitions 1 and 2, if $\mathsf{D}^h_{n,k,q}$ is the uniform distribution on the set of generator matrices of codes in $\mathcal{X}^h_{n,k,q}$, the authors of [ABL25a] omit D and write $\mathsf{PCE}^h_{n,k,q}$ and $\mathsf{SPCE}^h_{n,k,q}$. Most notably, they provide the first known algorithm that provably samples from $\mathsf{D}^h_{n,k,q}$ in [ABL25a, Def. 11] (with some constraints), simply called $\mathsf{SampleCode}$, which allows them to reduce the security of their UPKE scheme to the hardness of $\Delta\mathsf{PCE}^h_{n,k,q}$.

2.4 Counting Hollow Codes

Denote by $\begin{bmatrix} n \\ k \end{bmatrix}_q = \frac{\prod_{i=0}^{k-1}(q^{n-i}-1)}{\prod_{j=1}^{k}(q^j-1)}$ the q-ary Gaussian binomial coefficient, that is, the number of all $[n,k]$-linear codes over $\mathbb{F}_q$. We will need the *mass formulae* for LCD codes and codes with a particular hull dimension. While a mass formula for self-orthogonal codes has been known since [Seg59, Ple65, Ple68], we are interested in such a result for any fixed hull dimension, which the following describes asymptotically.

Proposition 1 ([Sen97, Thm. 4.18] and [Sen00, Prop. 14]). *A uniformly random* $[n,k]$*-linear code over* $\mathbb{F}_q$ *with* $k \leq \frac{n}{2}$ *has hull dimension* h *with probability*

$$P^h_{n,k,q} = \frac{1}{q^{\frac{h\cdot(h+1)}{2}}} \cdot \frac{\prod_{i=h+1}^{k}\left(1-\frac{1}{q^i}\right)}{\prod_{j=1}^{\left\lfloor\frac{k-h}{2}\right\rfloor}\left(1-\frac{1}{q^{2j}}\right)} \cdot \left(1+\mathcal{O}\left(\frac{k}{q^{\frac{n}{2}-h}}\right)\right).$$

As $k, n \to \infty$*, this probability converges to*

$$\lim_{n,k\to\infty} P^h_{n,k,q} = \prod_{i=1}^{\infty}\frac{1}{1+\frac{1}{q^i}} \cdot \prod_{j=1}^{h}\frac{1}{q^j-1} \approx \left(1-\frac{1}{q}\right)\cdot\frac{1}{q^{\binom{h+1}{2}}},$$

where the approximation is from [BMS26, Rem. 1].

This statement, as mentioned in [Sen97], is only useful when dealing with small hull dimensions. First progress in obtaining a usable and precise mass formula for codes with a particular hull dimension other than for self-orthogonal codes was made in [CMTQ19] for the case of LCD codes. This was recently extended in [LSL25] to codes of any hull dimension, concurrently to [ABL25a].

Proposition 2 ([CMTQ19, Thm. 25, Thm. 31 & Cor. 32]). *Let* $n, k, q \in \mathbb{Z}$ *with* $0 \leq k \leq n$ *and* q *odd prime power. Denote by* $\mathcal{X}^{+1,0}_{n,k,q}$ *(resp.* $\mathcal{X}^{-1,0}_{n,k,q}$*) the set of all* $[n,k]$*-linear LCD codes over* $\mathbb{F}_q$ *with a basis whose Gram matrix is* $\mathrm{diag}(1,\ldots,1)$ *(resp.* $\mathrm{diag}(1,\ldots,1,s)$ *for some quadratic non-residue* $s \in \mathbb{F}_q^\star$*), and as before by* $\mathcal{X}^0_{n,k,q}$ *the set of all* $[n,k]$*-linear LCD codes over* $\mathbb{F}_q$*. Then* $\mathcal{X}^0_{n,k,q} = \mathcal{X}^{+1,0}_{n,k,q} \sqcup \mathcal{X}^{-1,0}_{n,k,q}$*, and we have*

$$\left|\mathcal{X}^{\pm 1,0}_{n,k,q}\right| = \begin{cases} \frac{1}{2}q^{\frac{k\cdot(n-k)-1}{2}} \cdot A \cdot \left[\begin{smallmatrix} \frac{n}{2}-1 \\ \frac{k-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ odd and } n \text{ even}, \\ \frac{1}{2}q^{\frac{k\cdot(n-k)}{2}} \cdot B_\pm \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{k-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ odd and } n \text{ odd}, \\ \frac{1}{2}\cdot q^{\frac{k\cdot(n-k)}{2}} \cdot C_\pm \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{k}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ even and } n \text{ odd}, \\ \frac{1}{2}\cdot q^{\frac{k\cdot(n-k)}{2}} \cdot D_\pm \cdot \left[\begin{smallmatrix} \frac{n}{2} \\ \frac{k}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ even and } n \text{ even}, \end{cases}$$

where $A = q^{\frac{n}{2}} - \chi_q(-1)^{\frac{n}{2}}$, $B_\pm = q^{\frac{n-k}{2}} \pm \chi_q(-1)^{\frac{n-k}{2}}$, $C_\pm = q^{\frac{k}{2}} \pm \chi_q(-1)^{\frac{k}{2}}$, $D_\pm = \frac{\left(q^{\frac{k}{2}} \pm \chi_q(-1)^{\frac{k}{2}}\right) \cdot \left(q^{\frac{n-k}{2}} \pm \chi_q(-1)^{\frac{n-k}{2}}\right)}{\left(q^{\frac{n}{2}} + \chi_q(-1)^{\frac{n}{2}}\right)}$, *hence*

$$\left|\mathcal{X}^0_{n,k,q}\right| = \begin{cases} q^{\frac{k\cdot(n-k)-1}{2}} \cdot A \cdot \left[\begin{smallmatrix} \frac{n}{2}-1 \\ \frac{k-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ odd and } n \text{ even,} \\ q^{\frac{(k+1)\cdot(n-k)}{2}} \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{k-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ odd and } n \text{ odd,} \\ q^{\frac{k\cdot(n-k+1)}{2}} \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{k}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ even and } n \text{ odd,} \\ q^{\frac{k\cdot(n-k)}{2}} \cdot \left[\begin{smallmatrix} \frac{n}{2} \\ \frac{k}{2} \end{smallmatrix}\right]_{q^2} & \text{if } k \text{ even and } n \text{ even.} \end{cases}$$

Proposition 3 ([LSL25, Thm. 5 & 6]). *Let* $n, k, h, q \in \mathbb{Z}$ *with* $0 \le h \le k \le n - h$ *and* q *odd prime power. Denote by* $\mathcal{X}^{+1,h}_{n,k,q}$ *(resp.* $\mathcal{X}^{-1,h}_{n,k,q}$*) the set of all* $[n,k]$*-linear codes over* $\mathbb{F}_q$ *with hull dimension* h *and a basis whose Gram matrix is* $\mathrm{diag}(1,\dots,1,0,\dots,0)$ *(resp.* $\mathrm{diag}(1,\dots,1,s,0,\dots,0)$ *for some quadratic non-residue* $s \in \mathbb{F}_q^\star$*), the number of* 0*'s being* h*, and as before by* $\mathcal{X}^h_{n,k,q}$ *the set of all* $[n,k]$*-linear codes over* $\mathbb{F}_q$ *with hull dimension* h*. Denote also* $t = k - h$*. Then* $\mathcal{X}^h_{n,k,q} = \mathcal{X}^{+1,h}_{n,k,q} \sqcup \mathcal{X}^{-1,h}_{n,k,q}$*, and we have*

$$\left|\mathcal{X}^{\pm 1,h}_{n,k,q}\right| = \begin{cases} \left(\prod_{i=1}^h \frac{q^{n-t-2i+1}-1}{q^t\cdot(q^i-1)}\right) \cdot \left|\mathcal{X}^{\pm 1,0}_{n,k-h,q}\right| & \text{if } n-t \text{ is odd} \\ \left(\prod_{i=1}^h \frac{q^{n-t-2i+2}-1}{q^t\cdot(q^i-1)}\right) \cdot A_\pm \cdot \left|\mathcal{X}^{\pm 1,0}_{n,k-h,q}\right| & \text{if } n-t \text{ is even,} \end{cases}$$

where $A_\pm = \frac{q^{\frac{n-t}{2}-h} \pm \chi_q(-1)^{\frac{n-t}{2}}}{q^{\frac{n-t}{2}} \pm \chi_q(-1)^{\frac{n-t}{2}}}$*, hence using Proposition 2 we have*

$$\left|\mathcal{X}^h_{n,k,q}\right| = \begin{cases} \left(\prod_{i=1}^h \frac{q^{n-t-2i+1}-1}{q^t\cdot(q^i-1)}\right) \cdot q^{\frac{t\cdot(n-t)-1}{2}} \cdot B_- \cdot \left[\begin{smallmatrix} \frac{n}{2}-1 \\ \frac{t-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } t \text{ odd and } n \text{ even,} \\ \left(\prod_{i=1}^h \frac{q^{n-t-2i+2}-1}{q^t\cdot(q^i-1)}\right) \cdot q^{\frac{(t+1)\cdot(n-t)}{2}-h} \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{t-1}{2} \end{smallmatrix}\right]_{q^2} & \text{if } t \text{ odd and } n \text{ odd,} \\ \left(\prod_{i=1}^h \frac{q^{n-t-2i+1}-1}{q^t\cdot(q^i-1)}\right) \cdot q^{\frac{t\cdot(n-t+1)}{2}} \cdot \left[\begin{smallmatrix} \frac{n-1}{2} \\ \frac{t}{2} \end{smallmatrix}\right]_{q^2} & \text{if } t \text{ even and } n \text{ odd,} \\ \left(\prod_{i=1}^h \frac{q^{n-t-2i+2}-1}{q^t\cdot(q^i-1)}\right) \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot B_+ \cdot \left[\begin{smallmatrix} \frac{n}{2} \\ \frac{t}{2} \end{smallmatrix}\right]_{q^2} & \text{if } t \text{ even and } n \text{ even,} \end{cases}$$

where $B_- = q^{\frac{n}{2}} - \chi_q(-1)^{\frac{n}{2}}$ *and* $B_+ = \frac{q^{\frac{n}{2}-h} + \chi_q(-1)^{\frac{n}{2}}}{q^{\frac{n}{2}} + \chi_q(-1)^{\frac{n}{2}}}$.

2.5 Construction A Lattices

A lattice is a discrete additive subgroup of $\mathbb{R}^n$, represented algorithmically by a full-rank row-basis $\mathbf{B} \in \mathcal{GL}_n(\mathbb{R})$.[4] Define the dual lattice to $\mathcal{L}$ by $\mathcal{L}^\vee = \{\mathbf{x} \in \mathrm{Span}_{\mathbb{R}}(\mathcal{L}) \mid \forall\, \mathbf{y} \in \mathcal{L}\colon \langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{Z}\}$. The Gram matrix of $\mathbf{B}$ is defined as $\mathbf{Q} = \mathbf{B} \cdot \mathbf{B}^\intercal$, and it defines a positive-definite quadratic form. A lattice is integral if $\mathbf{Q} \in \mathbb{Z}^{n\times n}$. A lattice $\mathcal{L}$ is q-ary for $q \in \mathbb{Z}$ if $q \cdot \mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, in particular, it is integral. We are interested in q-ary lattices that are lifts of linear codes over $\mathbb{F}_q$ for an odd prime q. For a matrix $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ define its *Construction A lattice* as $\Lambda_q(\mathbf{A}) = \left\{\mathbf{y} \in \mathbb{Z}^n \;\middle|\; \mathbf{y} = \mathbf{A}^\intercal \cdot \mathbf{x} \pmod q, \mathbf{x} \in \mathbb{Z}^k\right\}$. This is a q-ary lattice and, conversely, every q-ary lattice is a Construction A lattice of some matrix obtained with the modulo q map. Notice that $\Lambda_q(\mathbf{A})$ depends only on $\mathfrak{C}(\mathbf{A})$. When $\mathbf{A} = [\mathbf{I}_k, \mathbf{M}]$ is in systematic form, we will identify $\Lambda_q(\mathbf{A})$ with its canonical row-basis

$$\begin{bmatrix} \mathbf{I}_k & \mathbf{M} \\ \mathbf{0} & q\cdot\mathbf{I}_{n-k} \end{bmatrix}.$$

A particular feature of these lattices is that $\det(\Lambda_q(\mathbf{A})) = q^{n-\mathrm{rank}(\mathbf{A})}$, which is a power of the odd prime q, meaning the local structure of Construction A lattices depends solely on their 2-adic and q-adic structure.

[4] Our lattices are assumed full-rank, since q-ary lattices always are. Since our codes are generated by rows, so are our lattices, despite column-bases being more common.

Since [BDG23, Lem. 1] shows that any two q-ary lattices are 2-adically equivalent[5] for odd q, we turn our attention to the q-adic structure. We will call a lattice h-hollow if it is a Construction A lattice of an h-hollow matrix.

2.6 Background on Quadratic Forms and Spaces

The following short exposition was greatly influenced by [Art57, Lam05] and simplified where appropriate. We recommend the notes of P. L. Clark [Claa, Clab] for a friendlier and broader introduction.

Let $\mathbb{F}$ be a field with $\mathrm{char}(\mathbb{F}) \neq 2$. For simplicity, we identify k-ary quadratic forms over $\mathbb{F}$, normally defined as homogeneous quadratic polynomials in k variables over $\mathbb{F}$, with their corresponding symmetric $k \times k$ matrices over $\mathbb{F}$. A quadratic form is called *non-degenerate* if $\det(\mathbf{Q}) \neq 0$. Two forms $\mathbf{Q}_0$ and $\mathbf{Q}_1$ are *equivalent* if there exists $\mathbf{U} \in \mathcal{GL}_k(\mathbb{F})$ such that $\mathbf{Q}_1 = \mathbf{U} \cdot \mathbf{Q}_0 \cdot \mathbf{U}^\mathsf{T}$. Taking determinants, we observe that $\det(\mathbf{Q}_1) = \det(\mathbf{U})^2 \cdot \det(\mathbf{Q}_0)$, that is, the determinants of equivalent quadratic forms always differ by a square factor in $\mathbb{F}$. For non-degenerate forms, this difference is captured by the *discriminant* $\Delta(\mathbf{Q})$, which is the coset of $\det(\mathbf{Q})$ in the square class group $\mathbb{F}^\star/(\mathbb{F}^\star)^2$.

A *quadratic space* is a pair $(V, \mathbf{Q})$, where V is a k-dimensional vector space over $\mathbb{F}$ and the k-ary quadratic form $\mathbf{Q}$ endows V with $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\mathsf{T} \cdot \mathbf{Q} \cdot \mathbf{y}$. Two quadratic spaces $(V, \mathbf{Q}_V)$ and $(W, \mathbf{Q}_W)$ are *isometric* if there is a vector space isomorphism $\phi \colon V \to W$ with $\phi(\mathbf{x})^\mathsf{T} \cdot \mathbf{Q}_W \cdot \phi(\mathbf{y}) = \mathbf{x}^\mathsf{T} \cdot \mathbf{Q}_V \cdot \mathbf{y}$ for all $\mathbf{x} \in V$ and $\mathbf{y} \in W$. A quadratic subspace of $(V, \mathbf{Q})$ is $(W, \mathbf{Q}|_W)$ where $W \leq V$. Two subspaces $W_1, W_2 \leq V$ are orthogonal if $\forall \mathbf{w}_1 \in W_1 \, \forall \mathbf{w}_2 \in W_2 \colon \langle \mathbf{w}_1, \mathbf{w}_2 \rangle = 0$, denoted as $W_1 \perp W_2$. The orthogonal complement of $W \leq V$ is defined as $W^\perp = \{\mathbf{v} \in V \mid \forall \mathbf{w} \in W \colon \langle \mathbf{v}, \mathbf{w} \rangle = 0\}$. A vector $\mathbf{v} \in V$ is *isotropic* if $\langle \mathbf{v}, \mathbf{v} \rangle = 0$, and *anisotropic* otherwise. A quadratic space is *regular* if $\mathbf{Q}$ is non-degenerate. A regular quadratic space is isotropic if it contains a non-zero isotropic vector, and anisotropic otherwise. A subspace of V is *totally isotropic* if it contains only isotropic vectors. The subspace $V^\perp \leq V$ is the *radical* of V, and V is regular if and only if $V^\perp \leq V$ is trivial. The dimension of $V^\perp$ is the co-rank of $\mathbf{Q}$.

Let $V^\perp \leq V$ have dimension h. There exists a quadratic subspace $W \leq V$ of dimension $k - h$ with $W \perp V^\perp$ such that $V = V^\perp \oplus W$. This procedure is called the *radical splitting* [Art57, Thm. III.3.3] of V, and we call such a subspace W a *regular component* of V in line with the nomenclature in [Lam05]. Clearly V is regular if and only if it is its own regular component. Assume from now on that V is regular, that is, we replace it with any of its regular components.

A *hyperbolic plane* $\mathbb{H}$ is a 2-dimensional isotropic space. All hyperbolic planes are isometric and have discriminant $-1 \cdot \mathbb{F}^2$ [Lam05, Thm. I.3.2]. A hyperbolic space is a space isometric to a direct (orthogonal) sum of hyperbolic planes [Art57, Def. III.3.8 & III.3.9]. Hyperbolic subspaces in a quadratic space control the behaviour of totally isotropic subspaces. Namely, every isotropic quadratic space contains a hyperbolic plane, and for every totally isotropic subspace $W \leq V$ of dimension $r > 0$ with V regular there exists a totally isotropic subspace $W' \leq V$ of dimension r with $W \cap W' = \{0\}$ such that $W \oplus W'$ is a hyperbolic space of dimension $2 \cdot r$. In particular, we see that the maximal dimension of a totally isotropic subspace is bounded by $\frac{k}{2}$ from above, and this bound is attained precisely when V is itself hyperbolic. It is immediate that this dimension is an invariant for quadratic space isometry. It is called the *Witt index* of $(V, \mathbf{Q})$ and denoted by $w(V, \mathbf{Q})$. By recursively removing isometric copies of $\mathbb{H}$ from V until an anisotropic subspace is reached, we arrive at Witt's decomposition theorem [Lam05, Thm. I.4.1]: any quadratic space $(V, \mathbf{Q})$ of Witt index w is isometric to an orthogonal sum of quadratic subspaces $V \cong V^\perp \oplus \bigoplus_{i=1}^{w} \mathbb{H} \oplus V_a$, where V_a is anisotropic and is called the *anisotropic component* of V. For the special case of finite fields, Artin computed the four possible *types* of regular geometry.

Lemma 1 ([Art57, Sec. III.6]). *Let $(V, \mathbf{Q})$ be a regular quadratic space of dimension k over $\mathbb{F}_q$ for an odd prime q, and let $s \in \mathbb{F}_q^\star$ be any non-residue. Then V is isometric to one of four possible Witt decompositions:*

(I) $\bigoplus_{i=1}^{\frac{k-1}{2}} \mathbb{H} \oplus \mathfrak{C}(\mathbf{v})$ *with* $\langle \mathbf{v}, \mathbf{v} \rangle = 1$ *if k odd and* $\Delta(\mathbf{Q}) = (-1)^{\frac{k-1}{2}}$,
(II) $\bigoplus_{i=1}^{\frac{k-1}{2}} \mathbb{H} \oplus \mathfrak{C}(\mathbf{v})$ *with* $\langle \mathbf{v}, \mathbf{v} \rangle = s$ *if k odd and* $\Delta(\mathbf{Q}) = (-1)^{\frac{k-1}{2}} \cdot s$,

[5] In fact, the 2-adic symbol depends only on n for odd q, but we omit this here.

(III) $\bigoplus_{i=1}^{\frac{k}{2}} \mathbb{H}$ *if* k *even and* $\Delta(\mathbf{Q}) = (-1)^{\frac{k}{2}}$,
(IV) $\bigoplus_{i=1}^{\frac{k-2}{2}} \mathbb{H} \oplus W$ *with* W *anisotropic of dim.* 2 *if* k *even and* $\Delta(\mathbf{Q}) = (-1)^{\frac{k}{2}} \cdot s$.

In particular, over a finite field there are, up to isometry, precisely two possible (anisotropic) spaces of odd dimension 1 classified by the square norm of their generating vector, and precisely two spaces of dimension 2 classified by the discriminant: $\mathbb{H}$ of discriminant $\chi_q(-1)$ and the anisotropic representative. Every quadratic space over a finite field of dimension $k \geq 3$ is isotropic as a direct corollary of the Chevalley–Warning Theorem [Che35, War35].

3 The Square Class Invariant

In this section, we interpret a linear code $\mathfrak{C} \leq \mathbb{F}_q^n$ of dimension k as a quadratic space not directly as the coordinate subspace $(\mathfrak{C}, \mathbf{I}_n) \leq (\mathbb{F}_q^n, \mathbf{I}_n)$, but by considering the coefficient space $\mathbb{F}_q^k$ endowed with the geometry of any Gram matrix of $\mathfrak{C}$. In short, the main idea of this section – borrowed from the study of LIP – is that if $(\mathbf{A}, \mathbf{B})$ generate a pair of signed permutation equivalent codes, then $(\mathbf{A} \cdot \mathbf{A}^\intercal, \mathbf{B} \cdot \mathbf{B}^\intercal)$ must be equivalent as quadratic forms over $\mathbb{F}_q$. Witt's decomposition theorem then yields that the discriminant of the coefficient space modulo the radical (which corresponds to the hull) is an invariant for (S)PCE that has not been previously considered in coding theory, giving a natural distinguisher. On the side of linear codes, we name this invariant the square class $\chi_{\mathfrak{C}} \in \{\pm 1\}$ and consider its distribution for a fixed hull dimension, observing it takes each of its possible values with probability roughly $\frac{1}{2}$. In the spirit of [NRS06, p. vii], the arithmetic correspondence is summarised in Table 1 at the end of this section.

3.1 Coefficient Quadratic Spaces

Interpreting an $[n, k]$-linear code $\mathfrak{C}$ over $\mathbb{F}_q$ as a quadratic space $(\mathfrak{C}, \mathbf{I}_n)$ dates back to the work of Pless [Ple65, Ple68]; we instead encode the code into the geometry of the space as a quadratic form by considering its coefficient quadratic spaces.

Definition 3. *For a full-rank* $\mathbf{A} \in \mathbb{F}_q^{k \times n}$, *define* $V_{\mathbf{A}} = (\mathbb{F}_q^k, \mathbf{A} \cdot \mathbf{A}^\intercal)$.

We can think of this as translating from the world of coordinates to the world of coefficients. Definition 3 may seem odd at first, since two generator matrices for the same code yield different quadratic spaces. However, we only consider linear codes up to (signed) permutation equivalence and quadratic spaces up to isometry, and we have the following.

Lemma 2. *Let* $\mathbf{A}, \mathbf{B}$ *generate* $[n, k]$*-linear codes over* $\mathbb{F}_q$ *for odd* q *and let* $V_{\mathbf{A}}, V_{\mathbf{B}}$ *be their corresponding quadratic spaces. Then* $V_{\mathbf{A}} \cong V_{\mathbf{B}}$ *if and only if the linear codes* $\mathfrak{C}(\mathbf{A})$ *and* $\mathfrak{C}(\mathbf{B})$ *are* $\mathcal{O}_n(\mathbb{F}_q)$*-equivalent, that is, there exist* $\mathbf{O} \in \mathcal{O}_n(\mathbb{F}_q)$ *and* $\mathbf{U} \in \mathcal{GL}_k(\mathbb{F}_q)$ *such that* $\mathbf{B} = \mathbf{U} \cdot \mathbf{A} \cdot \mathbf{O}$.

Proof. The if direction follows from $\mathbf{O} \cdot \mathbf{O}^\intercal = \mathbf{I}_n$, only if from [CMTQ19, Prop. 28]. □

Corollary 1. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *generate signed permutation equivalent linear codes over* $\mathbb{F}_q$ *for odd* q, *then* $V_{\mathbf{A}}$ *and* $V_{\mathbf{B}}$ *are isometric.*

Proof. $\mathcal{O}_n(\mathbb{Z}) \leq \mathcal{O}_n(\mathbb{F}_q)$ via the usual inclusion map. □

The hull of $\mathfrak{C}(\mathbf{A})$ corresponds to the radical $V_{\mathbf{A}}^\perp \leq V_{\mathbf{A}}$. The remaining invariant present in Lemma 1 is the discriminant, which we generalise to degenerate forms by first applying radical splitting and computing the discriminant of the remaining regular space.

Lemma 3. *Let* $n, k, q \in \mathbb{Z}$ *with* $0 \leq k \leq n$, q *odd prime power and* $\mathbf{A} \in \mathbb{F}_q^{k \times n}$ *full-rank. For* $\mathbf{x} \in \mathbb{F}_q^k$ *we have* $\mathbf{A}^\intercal \cdot \mathbf{x} \in \mathrm{hull}(\mathfrak{C}(\mathbf{A}))$ *if and only if* $\mathbf{x} \in V_{\mathbf{A}}^\perp$.

Proof. Immediate from the definition of the radical $V_{\mathbf{A}}^{\perp}$ in Section 2.6. □

Definition 4 (Generalised discriminant). *Let* $\mathbf{Q}$ *be a* k*-ary quadratic form over* $\mathbb{F}_q$ *of co-rank* $h < k$*, which block-diagonalises as* $\mathbf{Q} = \operatorname{diag}(\mathbf{Q}_{k-h}, \mathbf{0}^{h\times h})$*. Define* $\Delta(\mathbf{Q}) = \Delta(\mathbf{Q}_{k-h})$*. When* $h = k$ *and* $\mathbf{Q} = \mathbf{0}$*, set* $\Delta(\mathbf{Q}) = 1$*.*

Artin's classification of finite geometry (Lemma 1) reveals that the hull dimension and the generalised discriminant classify quadratic spaces up to isometry; in other words, $\mathcal{O}_n(\mathbb{F}_q)$-equivalence of linear codes is easy to decide. The full analogy is provided in Section 3.4; we now focus on translating Definition 4 to linear codes and determining its properties.

3.2 Back to Coordinates

On the side of linear codes we call the invariant in Definition 4 the square class.

Definition 5 (Square class of a linear code). *Let* $\mathfrak{C}$ *be an* $[n,k]$*-linear* h*-hollow code over* $\mathbb{F}_q$ *with* $h < k$*. The* square class *of* $\mathfrak{C}$ *is* $\chi_{\mathfrak{C}} = \chi_q(\Delta(\mathbf{Q})) \in \{\pm 1\}$ *where* $\mathbf{Q}$ *is any of its Gram matrices. When* $h = k$*, all self-orthogonal codes have square class* 1*. For a full-rank* $\mathbf{A} \in \mathbb{F}_q^{k\times n}$*, we denote* $\chi_{\mathbf{A}} = \chi_{\mathfrak{C}(\mathbf{A})}$*.*

Lemma 4. *Let* $\mathbf{A}$ *generate an* $[n,k]$*-linear code over* $\mathbb{F}_q$ *for odd* q *with square class* $\chi_{\mathbf{A}}$ *and hull dimension* h*. Then* $\chi_{\mathbf{A}}$ *and* h *determine* $V_{\mathbf{A}}$ *up to isometry.*

Proof. By Lemma 3, $V_{\mathbf{A}}^{\perp}$ is isometric to $(\mathbb{F}_q^h, \mathbf{0}^{h\times h})$. Then by Lemma 1 the Witt decomposition of $V_{\mathbf{A}}$ after radical splitting is in bijective correspondence with $\Delta(\mathbf{A}\cdot\mathbf{A}^{\mathrm{T}}) = \chi_q^{-1}(\chi_{\mathbf{A}})$ and $k-h$. □

Corollary 2 (Square class invariant). *The function* $f\colon \mathcal{X}_{n,k,q} \to \{\pm 1\}$ *that maps* $\mathfrak{C} \mapsto \chi_{\mathfrak{C}}$ *is an efficiently computable invariant for the relation of (signed) permutation equivalence.*

Proof. Combine Lemma 4 and Corollary 1. □

We need not compute the Witt decomposition of $V_{\mathbf{A}}$ to compute $\chi_{\mathbf{A}}$, the procedure amounts to computing one vector space quotient, one determinant, and one residue class test. An *LCD component* of an h-hollow code $\mathfrak{C}$, mentioned without proof in [ABL25c, App. C], is any matrix $\mathbf{T} \in \mathbb{F}_q^{(k-h)\times n}$ that modulo $\operatorname{hull}(\mathfrak{C})$ generates the vector space $\mathfrak{C}/\operatorname{hull}(\mathfrak{C})$ and is obtained efficiently by *dehulling* $\mathfrak{C}$.

Proposition 4 (Dehulling [ABL25c, App. C & Fig. 12]). *Let* $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ *be* h*-hollow with* $h < k$*, and let* $\mathfrak{C} = \mathfrak{C}(\mathbf{A})$*. There exists a deterministic algorithm running in time* $\mathcal{O}(n^{\omega})$ *that* dehulls $\mathbf{A}$*, that is, finds a matrix* $\mathbf{T} \in \mathbb{F}_q^{(k-h)\times n}$ *that generates an LCD code such that* $\mathfrak{C} = \operatorname{hull}(\mathfrak{C}) \oplus \mathfrak{C}(\mathbf{T})$*.*

Proof. The algorithm in Figure 1 using linear algebra finds representatives in $\mathfrak{C}$ for the basis for the quotient $(k-h)$-dimensional $\mathbb{F}_q$-vector space $\mathfrak{C}/\operatorname{hull}(\mathfrak{C})$ and stores them as $\mathbf{T}$. Every $\mathbf{x} \in \operatorname{hull}(\mathfrak{C}(\mathbf{T}))$ is orthogonal to all of $\mathfrak{C}(\mathbf{T})$ and to $\operatorname{hull}(\mathfrak{C})$, hence $\mathbf{x}$ is orthogonal to all of $\mathfrak{C}$. Thus $\mathbf{x} \in \operatorname{hull}(\mathfrak{C})$, and $\operatorname{hull}(\mathfrak{C}) \cap \mathfrak{C}(\mathbf{T}) = \{\mathbf{0}\}$ implies $\mathbf{x} = \mathbf{0}$. □

Corollary 3. *There is a deterministic algorithm that, given any generator matrix* $\mathbf{A}$ *for a linear code* $\mathfrak{C} \le \mathbb{F}_q^n$*, computes* $\chi_{\mathfrak{C}}$ *in time* $\mathcal{O}\big(n^{\omega} + \log^2(q)\big)$*.*

Proof. If $\mathfrak{C}$ is self-orthogonal, then output 1. Otherwise, dehulling and computing $\det(\cdot)$ take $\mathcal{O}(n^{\omega})$ by Proposition 4, and $\chi_q(\cdot)$ takes $\mathcal{O}\big(\log^2(q)\big)$ by [Coh93, 1.4.12]. □

To evaluate the advantage of our distinguisher in Section 3.3, we consider the distribution of $\chi_{\mathfrak{C}}$ for linear codes $\mathfrak{C}$ with a particular hull dimension.

Theorem 1. *Let $n, k, h, q \in \mathbb{Z}$ with $0 \le h < k \le \frac{n}{2}$ and q odd prime power. For a fixed hull dimension $h < k$, a uniformly random $[n,k]$-linear h-hollow code $\mathfrak{C}$ over $\mathbb{F}_q$ has square class $\chi \in \{\pm 1\}$ with probability negligibly close to $\frac{1}{2}$ in $(k-h)\cdot\log(q)$. Concretely, denote $p^{\chi,h}_{n,k,q} = \Pr_{\mathbf{A}\leftarrow\$\mathsf{D}^h_{n,k,q}}[\chi_{\mathbf{A}} = \chi]$. Then*

$$\left|p^{+1,h}_{n,k,q} - p^{-1,h}_{n,k,q}\right| \le \begin{cases} 0 & k-h \text{ odd, } n \text{ even} \\ q^{-\frac{n-k-h}{2}} & k-h \text{ odd, } n \text{ odd} \\ q^{-\frac{k-h}{2}} & k-h \text{ even, } n \text{ odd} \\ 2q^{-\frac{k-h}{2}} & k-h \text{ even, } n \text{ even} \end{cases} \le \mathsf{negl}((k-h)\cdot\log(q)),$$

in particular $\left|p^{\chi,h}_{n,k,q} - \frac{1}{2}\right| \le q^{-\frac{k-h}{2}}$.

Proof. Recall from Proposition 4 that $t = k-h$ is precisely the dimension of any LCD component $\mathbf{T}$ of $\mathbf{A} \leftarrow\$ \mathsf{D}^h_{n,k,q}$. We have

$$p^{+1,h}_{n,k,q} = \frac{\left|\mathcal{X}^{+1,h}_{n,k,q}\right|}{\left|\mathcal{X}^{h}_{n,k,q}\right|} \qquad \text{and} \qquad p^{-1,h}_{n,k,q} = \frac{\left|\mathcal{X}^{-1,h}_{n,k,q}\right|}{\left|\mathcal{X}^{h}_{n,k,q}\right|}.$$

Now use the formulae in Proposition 3 (from [LSL25]) and Proposition 2 (from [CMTQ19]). Denote $C_1 = \prod_{i=1}^{h} \frac{q^{n-t-2i+1}-1}{q^t\cdot(q^i-1)}$ and $C_2 = \prod_{i=1}^{h} \frac{q^{n-t-2i+2}-1}{q^t\cdot(q^i-1)}$.

Case $n-t$ odd $\bullet$ t odd and n even: We have $\left|\mathcal{X}^{+1,0}_{n,t,q}\right| = \left|\mathcal{X}^{-1,0}_{n,t,q}\right|$, hence also $\left|\mathcal{X}^{+1,h}_{n,k,q}\right| = \left|\mathcal{X}^{-1,h}_{n,k,q}\right|$ and so $p^{+1,h}_{n,k,q} = p^{-1,h}_{n,k,q}$.

Case $n-t$ odd $\bullet$ t even and n odd: We have

$$\left|p^{+1,h}_{n,k,q} - p^{-1,h}_{n,k,q}\right| = \left|\frac{C_1 \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \chi_q(-1)^{\frac{t}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t}{2}\end{smallmatrix}\right]_{q^2}}{C_1 \cdot q^{\frac{t\cdot(n-t+1)}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t}{2}\end{smallmatrix}\right]_{q^2}}\right| = \frac{1}{q^{\frac{t}{2}}} = \frac{1}{q^{\frac{k-h}{2}}}.$$

Case $n-t$ even $\bullet$ t odd and n odd: Let $A_\pm$ be as in Proposition 3. Then

$$\begin{aligned}
&\left|\left|\mathcal{X}^{+1,h}_{n,k,q}\right| - \left|\mathcal{X}^{-1,h}_{n,k,q}\right|\right| \\
&= \frac{C_2}{2} \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t-1}{2}\end{smallmatrix}\right]_{q^2} \cdot \left|A_+ \cdot \left(q^{\frac{n-t}{2}} + \chi_q(-1)^{\frac{n-t}{2}}\right) - A_- \cdot \left(q^{\frac{n-t}{2}} - \chi_q(-1)^{\frac{n-t}{2}}\right)\right| \\
&\overset{\bullet}{=} \frac{C_2}{2} \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t-1}{2}\end{smallmatrix}\right]_{q^2} \cdot \left|q^{\frac{n-t}{2}-h} + \chi_q(-1)^{\frac{n-t}{2}} - q^{\frac{n-t}{2}-h} + \chi_q(-1)^{\frac{n-t}{2}}\right| \\
&= C_2 \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t-1}{2}\end{smallmatrix}\right]_{q^2},
\end{aligned}$$

where for $\bullet$ notice that we multiply each $A_\pm$ by its denominator, and so

$$\left|p^{+1,h}_{n,k,q} - p^{-1,h}_{n,k,q}\right| = \frac{C_2 \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t-1}{2}\end{smallmatrix}\right]_{q^2}}{C_2 \cdot q^{\frac{(t+1)\cdot(n-t)}{2}-h} \cdot \left[\begin{smallmatrix}\frac{n-1}{2}\\ \frac{t-1}{2}\end{smallmatrix}\right]_{q^2}} = \frac{1}{q^{\frac{n-t}{2}-h}} = \frac{1}{q^{\frac{n-k-h}{2}}}.$$

Note that since we are only considering codes up to duality, that is $k \le \frac{n}{2}$, we may bound this from above by $q^{-\frac{t}{2}}$. Indeed, we have $n-k-h \ge k-h = t$.

Case $n-t$ even • t even and n even: We get that

$$\left|\left|\mathcal{X}^{+1,h}_{n,k,q}\right| - \left|\mathcal{X}^{-1,h}_{n,k,q}\right|\right| = \frac{C_2}{2} \cdot q^{\frac{t\cdot(n-t)}{2}} \cdot \begin{bmatrix} \frac{n}{2} \\ \frac{t}{2} \end{bmatrix}_{q^2} \cdot \left|A_+ \cdot D_+ - A_- \cdot D_-\right|,$$

where $D_\pm = \frac{\left(q^{\frac{t}{2}} \pm \chi_q(-1)^{\frac{t}{2}}\right)\cdot\left(q^{\frac{n-t}{2}} \pm \chi_q(-1)^{\frac{n-t}{2}}\right)}{\left(q^{\frac{n}{2}} + \chi_q(-1)^{\frac{n}{2}}\right)}$ as in Proposition 2. Since the denominator of $A_\pm$ is a factor in the numerator of $D_\pm$, both terms of $A_+ \cdot D_+ - A_- \cdot D_-$ have denominator $q^{\frac{n}{2}} + \chi_q(-1)^{\frac{n}{2}}$ and the remaining numerator simplifies

$$\begin{aligned}
&\left(q^{\frac{n-t}{2}-h} + \chi_q(-1)^{\frac{n-t}{2}}\right) \cdot \left(q^{\frac{t}{2}} + \chi_q(-1)^{\frac{t}{2}}\right) \\
&- \left(q^{\frac{n-t}{2}-h} - \chi_q(-1)^{\frac{n-t}{2}}\right) \cdot \left(q^{\frac{t}{2}} - \chi_q(-1)^{\frac{t}{2}}\right) \\
&= 2 \cdot \left(q^{\frac{n-t}{2}-h} \cdot \chi_q(-1)^{\frac{t}{2}} + q^{\frac{t}{2}} \cdot \chi_q(-1)^{\frac{n-t}{2}}\right).
\end{aligned}$$

Absorbing the factor of 2 into the fraction $\frac{C_2}{2}$ in front of this expression, we get

$$\left|p^{+1,h}_{n,k,q} - p^{-1,h}_{n,k,q}\right| = \left|\frac{A_+ \cdot D_+ - A_- \cdot D_-}{2B_+}\right|,$$

where $B_+ = \frac{q^{\frac{n}{2}-h} + \chi_q(-1)^{\frac{n}{2}}}{q^{\frac{n}{2}} + \chi_q(-1)^{\frac{n}{2}}}$ as in Proposition 3. Notice that B_+ and $A_+ \cdot D_+ - A_- \cdot D_-$ have the same denominator, so we cancel it. We bound from above with

$$\begin{aligned}
\left|\frac{A_+ \cdot D_+ - A_- \cdot D_-}{2B_+}\right| &= \left|\frac{q^{\frac{n-t}{2}-h} \cdot \chi_q(-1)^{\frac{t}{2}} + q^{\frac{t}{2}} \cdot \chi_q(-1)^{\frac{n-t}{2}}}{q^{\frac{n}{2}-h} + \chi_q(-1)^{\frac{n}{2}}}\right| \\
\leq \frac{q^{\frac{n-t}{2}-h} + q^{\frac{t}{2}}}{q^{\frac{n}{2}-h} - 1} &\overset{\bullet}{=} \frac{q^{-\frac{t}{2}} + q^{-\frac{n-t}{2}+h}}{1 - q^{-\frac{n}{2}+h}} \overset{\dagger}{\leq} \frac{2q^{-\frac{t}{2}}}{1-q^{-t}} \overset{\star}{\leq} \frac{9}{4} q^{-\frac{t}{2}}
\end{aligned}$$

where for $\bullet$ we divide both the numerator and the denominator by $q^{\frac{n}{2}-h}$, for $\dagger$ we use in the numerator that $-\frac{n-t}{2} + h \leq -\frac{t}{2}$ and in the denominator that $-\frac{n}{2} + h \leq -t$, both since $\frac{n}{2} \geq t + h = k$ by assumption, and for $\star$ we bound the denominator from below by its lowest possible value $\frac{8}{9}$ when $q = 3$ and $t = 2$.

Notice that this bound is tight for our estimate and it is attained exactly when also $k = \frac{n}{2}$ for any h. Otherwise, we may bound in $\star$ with $2q^{-\frac{t}{2}}$ instead (as desired); indeed since $h \leq \frac{n}{2} - t - 1$, the $\dagger$ bound is $q^{-\frac{t}{2}} \cdot \frac{1+q^{-1}}{1-q^{-t-1}} \leq 2q^{-\frac{t}{2}}$. Plugging the edge case into our original expression, we have that $\frac{n-t}{2} - h = \frac{t}{2}$ and $\chi_3(-1) = -1$, hence in the numerator we get $\left|q^{\frac{t}{2}} \cdot \chi_3(-1)^{\frac{t}{2}} \cdot (1 + \chi_3(-1)^h)\right| = q^{\frac{t}{2}} \cdot (1 + (-1)^h)$, and in the denominator we get $\left|q^t + (-1)^k\right| = q^t + (-1)^h$ (since t is even). When h is even we get $\frac{2q^{\frac{t}{2}}}{q^t+1} \leq 2q^{-\frac{t}{2}}$, and when h is odd the numerator equals 0. □

3.3 Deciding (S)PCE using Square Classes

We now propose an adversary against $\Delta(\mathsf{S})\mathsf{PCE}^h_{n,k,q}$ using the square class invariant and consider its advantage using Theorem 1.

Theorem 2 (Square class distinguisher). *Let $n, k, h, q \in \mathbb{Z}$ with $0 \leq h < k \leq \frac{n}{2}$ and q odd prime power. There exists an algorithm running in time $\mathcal{O}\left(n^\omega + \log^2(q)\right)$ with non-negligible advantage in $(k-h)\cdot\log(q)$ against $\Delta(\mathsf{S})\mathsf{PCE}^h_{n,k,q}$. Concretely, we have that the advantage of* SqClassD *in Figure 1 is*

$$\mathsf{Adv}_{\Delta(\mathsf{S})\mathsf{PCE}}(\mathsf{SqClassD}) \geq \frac{1}{4} - \mathcal{O}\left(q^{-(k-h)}\right).$$

$\mathsf{SqClassD}(\mathbf{A}, \mathbf{B})$	$\mathsf{Dehull}(\mathbf{A})$
$\mathbf{T_A}, _ \leftarrow \mathsf{Dehull}(\mathbf{A})$; $\mathbf{T_B}, _ \leftarrow \mathsf{Dehull}(\mathbf{B})$	$\mathbf{H} \coloneqq \mathsf{Hull}(\mathbf{A})$; $\mathbf{V} \coloneqq \mathfrak{C}(\mathbf{A})/\mathfrak{C}(\mathbf{H})$
return $\chi_q(\det(\mathbf{T_A} \cdot \mathbf{T_A}^\intercal)) = \chi_q(\det(\mathbf{T_B} \cdot \mathbf{T_B}^\intercal))$	$\mathbf{T} \coloneqq$ lift $\mathbf{V}$ to $\mathfrak{C}(\mathbf{A})$; **return** $(\mathbf{T}, \mathbf{H})$

Fig. 1: The square class distinguisher $\mathsf{SqClassD}$, which compares the square classes of the instance pair. The algorithm Dehull. The algorithm Hull that computes a hull generator is in [Sen00, App. A].

Proof. The algorithm $\mathsf{SqClassD}$ is given in Figure 1 and its running time is immediate from Corollary 3. Given a $\Delta(\mathsf{S})\mathsf{PCE}$ instance sampled using $\mathsf{D}^h_{n,k,q}$ for a fixed $h < k$, $\mathsf{SqClassD}$ simply computes the square classes $\chi_{\mathbf{A}}$ and $\chi_{\mathbf{B}}$ of $\mathfrak{C}(\mathbf{A})$ and $\mathfrak{C}(\mathbf{B})$, respectively, and returns whether they match. If the challenge bit in Definitions 1 and 2 is $b = 1$, then $\chi_{\mathbf{A}} = \chi_{\mathbf{B}}$ by Corollary 2, thus $\mathsf{SqClassD}$ wins with probability 1. Denote for simplicity $\mathcal{A} = \mathsf{SqClassD}$. Omitting the inputs, its normalised advantage is

$$\begin{aligned}
&2\mathsf{Adv}_{\Delta(\mathsf{S})\mathsf{PCE}}(\mathcal{A}) = 2\Pr[\mathcal{A} \text{ wins}] - 1 \\
&= 2 \cdot (\Pr[1 \leftarrow \mathcal{A} \mid b = 1] \cdot \Pr[b = 1] + \Pr[0 \leftarrow \mathcal{A} \mid b = 0] \cdot \Pr[b = 0]) - 1 \\
&= 1 + 2\Pr[0 \leftarrow \mathcal{A} \mid b = 0] \cdot \Pr[b = 0] - 1 = \Pr[0 \leftarrow \mathcal{A} \mid b = 0].
\end{aligned}$$

If the challenge bit is $b = 0$, then the probability that $\mathsf{SqClassD}$ wins is the probability that $\chi_{\mathbf{A}} \neq \chi_{\mathbf{B}}$. By Theorem 1, the probability that a code sampled from $\mathsf{D}^h_{n,k,q}$ has either square class $\chi \in \{\pm 1\}$ is at most $q^{-\frac{k-h}{2}}$ away from $\frac{1}{2}$ in absolute value. Denoting the exact difference by δ, we have

$$\begin{aligned}
\Pr[0 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{B}) \mid b = 0] &= \Pr_{\mathbf{A},\mathbf{B} \leftarrow \$ \mathsf{D}^h_{n,k,q}}[\chi_{\mathbf{A}} \neq \chi_{\mathbf{B}}] \\
&= 2\Pr_{\mathbf{A} \leftarrow \$ \mathsf{D}^h_{n,k,q}}[\chi_{\mathbf{A}} = 1] \cdot \Pr_{\mathbf{B} \leftarrow \$ \mathsf{D}^h_{n,k,q}}[\chi_{\mathbf{B}} = -1] \\
&= 2 \cdot \left(\frac{1}{2} + \delta\right) \cdot \left(\frac{1}{2} - \delta\right) = \frac{1}{2} - 2\delta^2.
\end{aligned}$$

Bounding δ by Theorem 1, our task is thus to analyse when $\delta^2 \leq q^{-(k-h)} < \frac{1}{4}$. Since $k - h \geq 1$, this condition can only fail when $q = 3$ and $t = k - h = 1$. However, recall from Theorem 1 that when t is odd, we actually have a better bound, that is $\delta \leq \frac{1}{2}q^{-\frac{t}{2}}$. Then of course $\delta^2 \leq \frac{1}{4}q^{-t} < \frac{1}{4}$ holds. □

Corollary 4. *Assume the notation of Theorem 2. For the parameters presented in [ABL25a, Tab. 1]* $\mathsf{SqClassD}$ *wins with probability at least* $\frac{3}{4} - \mathsf{negl}(\lambda)$.

We defer refining the hardness assumption for $(\mathsf{S})\mathsf{PCE}$ to take our distinguisher into account and mending the scheme from [ABL25a] to Section 5, once we have explored $(\mathsf{S})\mathsf{PCE}$ invariants further in Section 4 through the study of q-ary lattices and arithmetic of rational quadratic forms. Indeed, we will show in Section 4 that linear codes generated by full-rank matrices in $\mathbb{F}_q^{k \times n}$ for q odd prime share a hull dimension and square class if and only if their Construction A lattices are locally equivalent, that is, lie in the same genus. Recall from this section that the hull dimension and square class form a complete set of invariants for the relation of quadratic form equivalence over $\mathbb{F}_q$. However, to mend the UPKE scheme from [ABL25a] we must also consider the lattice side, in particular how the square class interacts with hollow lattice problems.

Remark 1. The square class invariant is useful only for odd q, since $\mathbb{F}_2^\star = (\mathbb{F}_2^\star)^2$, that is, all binary codes have square class 1, and more generally every element of $\mathbb{F}_{2^r}$ for $r \in \mathbb{N}$ is a quadratic residue since the squaring map $x \mapsto x^2$ is the Frobenius automorphism and hence bijective. This extinguishes any hope of using square classes to distinguish, for example, McEliece [McE78] public keys from random binary codes (cf. [Ran25]). We experimentally treat non-binary Goppa codes in Appendix B.1, observing that their genus distribution does not seem distinguishable from that of random codes, but do not attempt a formal proof.

Remark 2. We observe in Appendix B.2 that much like the hull dimension, the square class is not an invariant for LCE. For experiments see Section 6 and Appendix B.2, where we observe for a random code that the genus distribution of random codes from its orbit of the monomial action (used in LCE) does not seem distinguishable from that of random codes.

Table 1: Analogous terms for linear codes and their quadratic spaces.

Linear codes:	**Quadratic forms:**
$[n,k]$-linear code gen. $\mathbf{A}$ over $\mathbb{F}_q$	quadratic space $V_{\mathbf{A}} = (\mathbb{F}_q^k, \mathbf{Q} = \mathbf{A}\cdot\mathbf{A}^\intercal)$
square class	generalised discriminant of $\mathbf{Q}$
hull dimension	co-rank of $\mathbf{Q}$
hull, dehulling	radical, radical splitting
LCD component	regular component
self-orthogonal vector	isotropic vector
$\mathcal{O}_n(\mathbb{F}_q)$-equivalence	quadratic space isometry
genus of $\Lambda_q(\mathbf{A})$	Witt decomposition/type of $V_{\mathbf{A}}$

3.4 Summary of the Correspondence

All but the last row of Table 1 should now be clear:

- The square class $\chi_{\mathbf{A}}$ of $\mathfrak{C}(\mathbf{A})$ is by Definition 5 in bijective correspondence with the generalised discriminant $\Delta(\mathbf{A}\cdot\mathbf{A}^\intercal)$ of $V_{\mathbf{A}}$ via the multiplicative group isomorphism $\chi_q^{-1}\colon \{\pm 1\} \to \mathbb{F}_q^\star/(\mathbb{F}_q^\star)^2$.
- The hull dimension is measured by the rank deficiency of $\mathbf{A}\cdot\mathbf{A}^\intercal$, which is the dimension of the radical $V_{\mathbf{A}}^\perp$. Moreover, the hull and the radical are both defined as the subspace of vectors orthogonal to the whole space, that is $\mathbf{A}^\intercal\cdot\mathbf{x} \in \mathrm{hull}(\mathfrak{C}(\mathbf{A}))$ if and only if $\mathbf{x} \in V_{\mathbf{A}}^\perp$.
- Our dehulling procedure from Proposition 4 is simply the radical splitting procedure, producing an LCD component $\mathbf{T}$ of $\mathbf{A}$ that corresponds to the regular component $(\mathbb{F}_q^{k-h}, \mathbf{T}\cdot\mathbf{T}^\intercal) \cong V_{\mathbf{T}} \leq V_{\mathbf{A}} \cong (\mathbb{F}_q^k, \mathbf{A}\cdot\mathbf{A}^\intercal)$. In fact, LCD codes correspond to regular spaces, since a code is LCD if and only if its Gram matrix is full-rank and self-orthogonal codes correspond to quadratic spaces that are equal to their radical.
- Lemma 2 shows that $\mathcal{O}_n(\mathbb{F}_q)$-equivalence of linear codes corresponds to quadratic space isometry.

We defer explaining the last row to Section 4.5.

4 Local Structure of q-ary Lattices

We shift our attention from hollow codes over $\mathbb{F}_q$ to q-ary (hollow) lattices. In this section, we prove in Theorem 3 a *local structure theorem* for hollow lattices that shows their genus is determined completely by the hull dimension and square class of the underlying linear code. This extends [BDG23, Lem. 7], which only characterised the two largest genera corresponding to LCD codes, while we describe any hull dimension. We observe that none of the genera that arise split further into so-called spinor genera [Eic74, Cas78], which could otherwise yield further invariants for LIP on q-ary lattices and (S)PCE on the underlying linear codes. If the main idea of Section 3 was to study the (S)PCE instance $(\mathbf{A}, \mathbf{B})$ through the pair of quadratic forms $(\mathbf{A}\cdot\mathbf{A}^\intercal, \mathbf{B}\cdot\mathbf{B}^\intercal)$ over the finite field, in this section we instead consider rational quadratic forms given by Gram matrices of the lattice pair $(\Lambda_q(\mathbf{A}), \Lambda_q(\mathbf{B}))$ for odd prime q. Since $\det(\Lambda_q(\mathbf{A}))$ is always a power of the odd prime q, the local structure of Construction A lattices depends solely on their 2-adic and q-adic structure. Since [BDG23, Lem. 1] shows that any two q-ary lattices are 2-adically equivalent for odd q (the 2-adic symbol depends only on n), the genus of $\Lambda_q(\mathbf{A})$ is determined solely by its q-adic symbol.

4.1 Technical Observations

The first and key step of the proof of Theorem 3 is to prepare the Gram matrix of the Construction A lattice before computation. We do so by carefully choosing an appropriate basis and permutation for the underlying linear code. We then show in Proposition 5 that such conditioning does not change the genus of the Construction A lattice.

Lemma 5. *Every self-orthogonal code admits two disjoint information sets.*

Proof. Let $\mathbf{H}$ generate a self-orthogonal code and without loss of generality let it be in systematic form $\mathbf{H} = [\mathbf{I}_k, \mathbf{M}]$ with $\mathbf{M} \in \mathbb{F}_q^{k\times(n-k)}$ and $2k \leq n$, that is $[\![1, k]\!]$ is an information set. Then $\mathbf{M} \cdot \mathbf{M}^\mathsf{T} = -\mathbf{I}$,[6] thus $\mathbf{M}$ has full rank k and hence a full-rank $k \times k$ submatrix, whose indices form another information set disjoint from $[\![1, k]\!]$. □

Corollary 5. *Every $[n, k]$-linear self-orthogonal code $\mathfrak{C}$ over $\mathbb{F}_q$ is permutation equivalent to one with a generator matrix of the form $[\mathbf{V}, \mathbf{M}, \mathbf{I}_k]$ with $\mathbf{V} \in \mathbb{F}_q^{k\times k}$ full-rank. This can be efficiently computed from any generator matrix of $\mathfrak{C}$.*

Proof. Start with any generator matrix for $\mathfrak{C}$ and permute it such that two disjoint information sets, which exist according to Lemma 5, are at coordinates $[\![1, k]\!]$ and $[\![n - k + 1, n]\!]$. Then multiply this matrix from the left with the inverse of the full-rank $k \times k$ submatrix at coordinates $[\![n - k + 1, n]\!]$. □

Corollary 6. *Every $[n, k]$-linear h-hollow code $\mathfrak{C}$ over $\mathbb{F}_q$ with $0 < h < k$ is permutation equivalent to one with a generator matrix of the form*

$$\mathbf{A} = \begin{bmatrix}\mathbf{T}\\ \mathbf{H}\end{bmatrix} = \begin{bmatrix}\mathbf{N}'_k & \mathbf{N}_{n-k-h} & \mathbf{0}_h\\ \mathbf{M}'_k & \mathbf{M}_{n-k-h} & \mathbf{I}_h\end{bmatrix},$$

where $\mathbf{H}$ generates the hull of $\mathfrak{C}(\mathbf{A})$, $\mathbf{T}$ is an LCD component of $\mathfrak{C}(\mathbf{A})$, the subscripts on the right-hand side denote the number of columns of the matrices, and we have that $[\![1, k]\!]$ is an information set of $\mathbf{A}$. There is an efficient algorithm that computes $\mathbf{A}$ from any generator matrix of $\mathfrak{C}$.

Proof. We give an algorithm using only basis changes and permutations that will always yield $\mathbf{A}$ from any generator matrix of $\mathfrak{C}$. Start with the canonical generator matrix for $\mathfrak{C}$ and dehull it with Proposition 4 to obtain $\mathbf{T}$ and $\mathbf{H}$ that generate an LCD component and the hull of $\mathfrak{C}$, respectively. Now permute $\mathbf{H}$ (and $\mathbf{T}$ consistently) such that $[\![1, h]\!]$ and $[\![n-h+1, n]\!]$ are its disjoint information sets given by Lemma 5. Row-reduce $\mathbf{H}$ such that its $h \times h$ submatrix at coordinates $[\![n - h + 1, n]\!]$ is $\mathbf{I}_h$, that is, multiply $\mathbf{H}$ from the left with the inverse of its submatrix at coordinates $[\![n - h + 1, n]\!]$. Now reduce the rows of $\mathbf{T}$ with the rows of $\mathbf{H}$ using the last information set $[\![n - h + 1, n]\!]$ as pivots, that is subtract from $\mathbf{T}$ the matrix $\mathbf{T}_{[\![n-h+1,n]\!]} \cdot \mathbf{H}$, where $\mathbf{T}_{[\![n-h+1,n]\!]}$ is the $(k - h) \times h$ submatrix of $\mathbf{T}$ at coordinates $[\![n - h + 1, n]\!]$. The matrix $\mathbf{H}$ remains unchanged and the matrix $\mathbf{T}$ now ends with h all-zero columns.

Observe that restricted to coordinates $[\![1, n - h]\!]$, $\mathbf{H}$ and $\mathbf{T}$ generate two orthogonal LCD codes $\mathbf{T}$ because it has zeros at coordinates $[\![n - h + 1, n]\!]$ and $\mathbf{H}$ because it is of the form given in Corollary 5 and so the Gram matrix $\mathbf{H}_{[\![1,n-h]\!]} \cdot \mathbf{H}^\mathsf{T}_{[\![1,n-h]\!]} = -\mathbf{I}_h$ is full-rank by construction. We have $\mathfrak{C}(\mathbf{H})_{[\![1,n-h]\!]} \subseteq \mathfrak{C}(\mathbf{T})^\perp_{[\![1,n-h]\!]}$ and vice versa, hence they must be linearly independent, and together form an $[n - h, k]$-linear LCD code, in particular the columns of this constructed $\mathbf{A}$ have full rank k already at coordinates $[\![1, n - h]\!]$. Because the first h columns of $\mathbf{H}$ have full rank h by construction, so do the first h columns of $\mathbf{A}$. Now simply find $k - h$ columns with coordinates in $[\![h + 1, n - h]\!]$ that form an information set for $\mathbf{A}$ together with $[\![1, h]\!]$ and permute them to coordinates $[\![h + 1, k]\!]$. □

4.2 The Local Structure Theorem

The following proposition states that $\Lambda_q(\mathbf{A})$ does not depend on the basis $\mathbf{A}$ chosen for the same code or on any (signed) permutation of the code, the two operations used to condition our form.

Proposition 5. *Let $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times n}$ with q odd prime generate (signed) permutation equivalent $[n, k]$-linear codes. Then $\Lambda_q(\mathbf{A})$ and $\Lambda_q(\mathbf{B})$ are isometric.*

Proof. There exists $\mathbf{O} \in \mathcal{O}_n(\mathbb{Z})$ such that $\mathfrak{C}(\mathbf{B}) = \mathfrak{C}(\mathbf{A} \cdot \mathbf{O})$. Then $\Lambda_q(\mathbf{B}) = \Lambda_q(\mathbf{A} \cdot \mathbf{O}) = \mathbf{O}^\mathsf{T} \cdot \Lambda_q(\mathbf{A})$. □

[6] Such a matrix is sometimes called row-antiorthogonal [Mas98a, Mas98b].

With the help of square classes, we give a structure theorem that expands on [BDG23, Lem. 7] by providing a complete description of the genus of a q-ary lattice, regardless of the hull dimension of its underlying linear code.

Theorem 3 (Local structure theorem). *Let $n, k, h, q \in \mathbb{Z}$ with $0 \le h \le k \le n - h$ and q odd prime. Let $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ be h-hollow and let $\mathcal{L} = \Lambda_q(\mathbf{A})$ be its Construction A lattice with a Gram matrix $\mathbf{Q}$. Then the q-adic symbol of $\mathbf{Q}$ is given by*

$$n_0 = k - h \qquad n_1 = 2h \qquad n_2 = n - k - h$$
$$\epsilon_0 = \chi_{\mathbf{A}} \qquad \epsilon_1 = \chi_q(-1)^h \qquad \epsilon_2 = \chi_{\mathbf{A}^\perp}.$$

Proof. The proof proceeds by direct computation using the algorithm computing the q-adic symbol from [CS99, Ch. 15.7] and proving steps along the way, separating three cases. For completeness, we first consider $h = 0$, already proven in [BDG23, Lem. 7] by block-diagonalising the Gram matrix. We then consider the more complicated cases of $0 < h < k$ and $h = k$.

In this proof we will consider matrices over $\mathbb{Z}_q$ even if we normally consider them over $\mathbb{F}_q$, by interpreting their entries as q-adic integers of degree 0. We will not change notation to accommodate this, instead we will explicitly denote the modulo q when over $\mathbb{F}_q$. The 'basis complement' of a matrix $\mathbf{B} \in \mathbb{F}_q^{k\times n}$ is the matrix $\mathbf{C} \in \mathbb{F}_q^{(n-k)\times n}$ whose rows are all $\mathbf{e}_i$ for i not in the lexicographically first information set of $\mathbf{B}$.

Case $h = 0$ • degree 0: Since $\mathfrak{C}(\mathbf{A})$ is an LCD code, we have that the Gram matrix $\mathbf{A} \cdot \mathbf{A}^\intercal$ is full-rank over $\mathbb{F}_q$. Without loss of generality by Proposition 5, the matrix $\mathbf{A}$ is row-reduced and permuted into systematic form $[\mathbf{I}_k, \mathbf{N}]$, and so

$$\mathbf{Q} = \begin{bmatrix} \mathbf{A}\cdot\mathbf{A}^\intercal & q\cdot\mathbf{N} \\ q\cdot\mathbf{N}^\intercal & q^2\cdot\mathbf{I}_{n-k} \end{bmatrix}.$$

By assumption $\nu_q(\mathbf{Q}) = 0$, otherwise $\mathbf{A}$ could not be 0-hollow, and over $\mathbb{F}_q$ we get the projected Gram matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{A}\cdot\mathbf{A}^\intercal & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \pmod q,$$

which has left kernel over $\mathbb{F}_q$ of dimension $n - k$ generated by $\mathbf{B} = [\mathbf{0}_k, \mathbf{I}_{n-k}]$. Thus the degree-0 Jordan component at q yields $n_0 = k$ and $\epsilon_0 = \left(\frac{\det(\mathbf{A}\cdot\mathbf{A}^\intercal)}{q}\right) = \chi_{\mathbf{A}}$.

Case $h = 0$ • degree 2: Next, we excise the degree-0 part of $\mathbf{Q}$. The basis complement of the kernel of $\mathbf{Q}$ is just the matrix $\mathbf{C} = [\mathbf{I}_k, \mathbf{0}_{n-k}]$ and the degree-0 part of $\mathbf{Q}$ is then given by the block $\mathbf{Q}_0 = \mathbf{C}\cdot\mathbf{Q}\cdot\mathbf{C}^\intercal = \mathbf{A}\cdot\mathbf{A}^\intercal$. We denote $\mathbf{U} = \mathbf{Q}_0^{-1}$, where the inverse is over $\mathbb{Z}_q$, and compute

$$\begin{aligned}
\mathbf{Q}' &= \mathbf{B}\cdot\left(\mathbf{Q} - (\mathbf{C}\cdot\mathbf{Q})^\intercal\cdot\mathbf{U}\cdot(\mathbf{C}\cdot\mathbf{Q})\right)\cdot\mathbf{B}^\intercal \\
&= \begin{bmatrix}\mathbf{0}\ \mathbf{I}_{n-k}\end{bmatrix}\cdot\left(\begin{bmatrix} \mathbf{A}\cdot\mathbf{A}^\intercal & q\cdot\mathbf{N} \\ q\cdot\mathbf{N}^\intercal & q^2\cdot\mathbf{I}_{n-k}\end{bmatrix} - \begin{bmatrix}\mathbf{A}\cdot\mathbf{A}^\intercal \\ q\cdot\mathbf{N}^\intercal\end{bmatrix}\cdot\mathbf{U}\cdot\begin{bmatrix}\mathbf{A}\cdot\mathbf{A}^\intercal\ q\cdot\mathbf{N}\end{bmatrix}\right)\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{I}_{n-k}\end{bmatrix} \\
&= q^2\cdot\left(\mathbf{I}_{n-k} - \mathbf{N}^\intercal\cdot\mathbf{U}\cdot\mathbf{N}\right) = q^2\cdot\left(\mathbf{I}_{n-k} - \mathbf{N}^\intercal\cdot\left(\mathbf{I}_k + \mathbf{N}\cdot\mathbf{N}^\intercal\right)^{-1}\cdot\mathbf{N}\right) = q^2\cdot\left(\mathbf{I}_{n-k} + \mathbf{N}^\intercal\cdot\mathbf{N}\right)^{-1},
\end{aligned}$$

the last equality is due to the matrix inversion lemma [Ber09, Cor. 2.8.8]. Clearly $\nu_q(\mathbf{Q}') = 2$ and we get that $n_1 = 0$ (and $\epsilon_1 = 1$). The matrix $\mathbf{Q}'$, after dividing by q^2, is full-rank modulo q, as it is a Gram matrix inverse of the dual LCD code $\mathfrak{C}(\mathbf{A})^\perp$. In the second Jordan component we have $n_2 = n - k$ and

$$\epsilon_2 = \left(\frac{\det(\mathbf{I}_{n-k} + \mathbf{N}^\intercal\cdot\mathbf{N})^{-1}}{q}\right) = \left(\frac{\det(\mathbf{I}_{n-k} + \mathbf{N}^\intercal\cdot\mathbf{N})}{q}\right) = \chi_{\mathbf{A}^\perp},$$

where the equality is due to $\chi_q(a) = \chi_q(a^{-1})$ for any $a \in \mathbb{F}_q^\star$.

Case $0 < h < k$ • *degree* 0*:* Let us first prepare the matrix $\mathbf{A}$ using Corollary 6, which does not change the q-adic symbol by Proposition 5. We obtain

$$\mathbf{A} = \begin{bmatrix} \mathbf{T} \\ \mathbf{H} \end{bmatrix} = \begin{bmatrix} \mathbf{N'}_k & \mathbf{N}_{n-k-h} & \mathbf{0}_h \\ \mathbf{M'}_k & \mathbf{M}_{n-k-h} & \mathbf{I}_h \end{bmatrix}.$$

Since this $\mathbf{A}$ has an information set at coordinates $[\![1, k]\!]$, we consider the lattice generated by

$$\Lambda_q(\mathbf{A}) = \begin{bmatrix} \mathbf{N'}_k & \mathbf{N}_{n-k-h} & \mathbf{0}_h \\ \mathbf{M'}_k & \mathbf{M}_{n-k-h} & \mathbf{I}_h \\ & q \cdot \mathbf{I}_{n-k-h} & \\ & & q \cdot \mathbf{I}_h \end{bmatrix}.$$

For the degree-0 Jordan component, we consider the Gram matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{T} \cdot \mathbf{T}^\intercal & \mathbf{T} \cdot \mathbf{H}^\intercal & q \cdot \mathbf{T} \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix}^\intercal \\ \mathbf{H} \cdot \mathbf{T}^\intercal & \mathbf{H} \cdot \mathbf{H}^\intercal & q \cdot \mathbf{H} \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix}^\intercal \\ q \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix} \cdot \mathbf{T}^\intercal & q \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix} \cdot \mathbf{H}^\intercal & q^2 \cdot \mathbf{I}_{n-k} \end{bmatrix} \in \mathbb{Z}_q^{n \times n}.$$

Observe that since $\mathbf{H}$ generates the hull of $\mathfrak{C}(\mathbf{A})$ over $\mathbb{F}_q$, we have

$$\mathbf{Q} = \begin{bmatrix} \mathbf{T} \cdot \mathbf{T}^\intercal & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \pmod q,$$

and hence $n_0 = k - h$ and $\epsilon_0 = \left(\frac{\det(\mathbf{T} \cdot \mathbf{T}^\intercal)}{q}\right) = \chi_{\mathbf{A}}$.

Case $0 < h < k$ • *degree* 1*:* Continuing, the kernel of the above $\mathbf{Q}$ over $\mathbb{F}_q$ is generated by $\mathbf{B} = [\mathbf{0}_{k-h}, \mathbf{I}_{n-k+h}]$, thus its basis complement is generated by $\mathbf{C} = [\mathbf{I}_{k-h}, \mathbf{0}_{n-k+h}]$. The degree-0 Jordan component of $\mathbf{Q}$ is then the block

$$\mathbf{Q}_0 = \mathbf{C} \cdot \mathbf{Q} \cdot \mathbf{C}^\intercal = \mathbf{T} \cdot \mathbf{T}^\intercal \in \mathbb{Z}_q^{(k-h) \times (k-h)}.$$

We denote $\mathbf{U} = \mathbf{Q}_0^{-1}$, where the inverse is over $\mathbb{Z}_q$, and compute

$$\begin{aligned} \mathbf{Q'} &= \mathbf{B} \cdot \left(\mathbf{Q} - (\mathbf{C} \cdot \mathbf{Q})^\intercal \cdot \mathbf{U} \cdot (\mathbf{C} \cdot \mathbf{Q})\right) \cdot \mathbf{B}^\intercal \\ &= \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{n-k+h} \end{bmatrix}^\intercal \cdot \left(\mathbf{Q} - \begin{bmatrix} \mathbf{T} \cdot \mathbf{T}^\intercal \\ \mathbf{H} \cdot \mathbf{T}^\intercal \\ q \cdot [\mathbf{0}, \mathbf{I}_{n-k}] \cdot \mathbf{T}^\intercal \end{bmatrix} \cdot \mathbf{U} \cdot \begin{bmatrix} \mathbf{T} \cdot \mathbf{T}^\intercal \\ \mathbf{H} \cdot \mathbf{T}^\intercal \\ q \cdot [\mathbf{0}, \mathbf{I}_{n-k}] \cdot \mathbf{T}^\intercal \end{bmatrix}^\intercal \right) \cdot \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{n-k+h} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{H} \cdot \tilde{\mathbf{T}} \cdot \mathbf{H}^\intercal & q \cdot \mathbf{H} \cdot \tilde{\mathbf{T}} \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix}^\intercal \\ q \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix} \cdot \tilde{\mathbf{T}} \cdot \mathbf{H}^\intercal & q^2 \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix} \cdot \tilde{\mathbf{T}} \cdot \begin{bmatrix} \mathbf{0} \ \mathbf{I}_{n-k} \end{bmatrix}^\intercal \end{bmatrix}, \end{aligned}$$

where $\tilde{\mathbf{T}} = \mathbf{I}_n - \mathbf{T}^\intercal \cdot (\mathbf{T} \cdot \mathbf{T}^\intercal)^{-1} \cdot \mathbf{T}$ is the projector onto $\mathfrak{C}(\mathbf{T})^\perp$. Note that modulo q, we have that $\mathbf{H} \cdot \tilde{\mathbf{T}} = \mathbf{H}$, and by symmetry $\tilde{\mathbf{T}} \cdot \mathbf{H}^\intercal = \mathbf{H}^\intercal$. Thus $\nu_q(\mathbf{Q'}) = 1$ and dividing by q gives us the next form, which we relabel to $\mathbf{Q}$. Indeed, over $\mathbb{F}_q$ we get

$$\mathbf{Q} = \begin{bmatrix} \frac{\mathbf{H} \cdot \mathbf{H}^\intercal}{q} & \mathbf{M} & \mathbf{I}_h \\ \mathbf{M}^\intercal & \mathbf{0} & \mathbf{0} \\ \mathbf{I}_h & \mathbf{0} & \mathbf{0} \end{bmatrix} \pmod q.$$

The left kernel of $\mathbf{Q}$ over $\mathbb{F}_q$ is generated by $\mathbf{B} = [\mathbf{0}_h, \mathbf{I}_{n-k-h}, -\mathbf{M}^\intercal]$, since the top h rows of $\mathbf{Q}$ have full rank h, and $[\mathbf{I}_{n-k-h}, -\mathbf{M}^\intercal]$ generates the dual code of the code generated by the remaining columns $[\mathbf{M}, \mathbf{I}_h]^\intercal$ of $\mathbf{Q}$. The kernel thus has dimension $n - k - h$ and its basis complement is generated by

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_h & \mathbf{0}_{n-k-h} & \mathbf{0}_h \\ \mathbf{0}_h & \mathbf{0}_{n-k-h} & \mathbf{I}_h \end{bmatrix},$$

which gives $n_1 = 2h$. The degree-0 Jordan component of $\mathbf{Q}$ (which is the degree-1 Jordan component of the original form) is then

$$\mathbf{Q}_1 = \mathbf{C} \cdot \mathbf{Q} \cdot \mathbf{C}^\mathsf{T} = \begin{bmatrix} \frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} & \mathbf{I}_h \\ \mathbf{I}_h & q\cdot\mathbf{I}_h \end{bmatrix} \in \mathbb{Z}_q^{2h\times 2h},$$

where the bottom-right block is due to our conditioning of $\mathbf{T}$ with $\mathbf{0}_h$ as the right-most $(k-h)\times h$ submatrix, giving that the bottom-right $h\times h$ submatrix of $\tilde{\mathbf{T}}$ is $\mathbf{I}_h$. Over $\mathbb{F}_q$, we get $\epsilon_1 = \left(\frac{\det(-\mathbf{I}_h)}{q}\right) = \left(\frac{(-1)^h}{q}\right) = \left(\frac{-1}{q}\right)^h = \chi_q(-1)^h$.

Case $0 < h < k$ • *degree* 2*:* We use the known formula for the 2×2 block matrix inverse over fields of characteristic 0 when the lower-left submatrix is invertible (adapted from [Ber09, Prop. 2.8.7]) to obtain that

$$\mathbf{U} = \begin{bmatrix} -q\cdot\tilde{\mathbf{H}}, \mathbf{I}_h + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} \\ \tilde{\mathbf{H}} \qquad -\tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} \end{bmatrix}$$

is the inverse of $\mathbf{Q}_1$ over $\mathbb{Z}_q$, where $\tilde{\mathbf{H}} = (\mathbf{I}_h - \mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T})^{-1}$. First note that $\tilde{\mathbf{H}}$ is indeed well-defined, since $\mathbf{I}_h - \mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} = \mathbf{I}_h \pmod q$. We then have

$$\begin{aligned}
\mathbf{U}\cdot\mathbf{Q}_1 &= \begin{bmatrix} -q\cdot\tilde{\mathbf{H}}, \mathbf{I}_h + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} \\ \tilde{\mathbf{H}} \qquad -\tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} \end{bmatrix} \cdot \begin{bmatrix} \frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} & \mathbf{I}_h \\ \mathbf{I}_h & q\cdot\mathbf{I}_h \end{bmatrix} \\
&= \begin{bmatrix} -\tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} + \mathbf{I}_h + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}, -q\cdot\tilde{\mathbf{H}} + q\cdot\mathbf{I}_h + q\cdot\tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} \\ \tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} - \tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} \qquad \tilde{\mathbf{H}} - \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_h, q\cdot\mathbf{I}_h - q\cdot\tilde{\mathbf{H}}\cdot\left(\mathbf{I}_h - \mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}\right) \\ \mathbf{0} \qquad \tilde{\mathbf{H}}\cdot\left(\mathbf{I}_h - \mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}\right) \end{bmatrix} = \begin{bmatrix} \mathbf{I}_h, q\cdot\mathbf{I}_h - q\cdot\mathbf{I}_h \\ \mathbf{0} \qquad \mathbf{I}_h \end{bmatrix} = \begin{bmatrix} \mathbf{I}_h & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_h \end{bmatrix}.
\end{aligned}$$

We know the first h rows and columns of $\mathbf{Q}$ will be excised trivially, since the first h columns of $\mathbf{B}$ are simply $\mathbf{0}$. Thus we focus on the bottom-right $(n-k)\times(n-k)$ submatrix of $\mathbf{Q}$, which is the bottom-right $(n-k)\times(n-k)$ submatrix of $q\cdot\tilde{\mathbf{T}}$. From it, we must subtract

$$\begin{aligned}
&\left[\begin{bmatrix}\mathbf{0}\ \mathbf{I}_{n-k}\end{bmatrix}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}, q\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{I}_h\end{bmatrix}\right]\cdot\begin{bmatrix} -q\cdot\tilde{\mathbf{H}}, \mathbf{I} + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T} \\ \tilde{\mathbf{H}} \qquad -\tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}}{q} \end{bmatrix}\cdot\begin{bmatrix}\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{I}_{n-k}\end{bmatrix}\\ q\cdot\begin{bmatrix}\mathbf{0}\ \mathbf{I}_h\end{bmatrix}\end{bmatrix} \\
&= -q\cdot[\mathbf{0},\mathbf{I}_{n-k}]\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}\cdot\tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot[\mathbf{0},\mathbf{I}_{n-k}]^\mathsf{T} + q\cdot[\mathbf{0},\mathbf{I}_h]^\mathsf{T}\cdot\tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot[\mathbf{0},\mathbf{I}_{n-k}]^\mathsf{T} \\
&+ q\cdot[\mathbf{0},\mathbf{I}_{n-k}]\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}\cdot(\mathbf{I} + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T})\cdot[\mathbf{0},\mathbf{I}_h] - q\cdot[\mathbf{0},\mathbf{I}_h]^\mathsf{T}\cdot\tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\tilde{\mathbf{T}}\cdot\mathbf{H}^\mathsf{T}\cdot[\mathbf{0},\mathbf{I}_h].
\end{aligned}$$

We immediately notice that after dividing by q and looking at the new form over $\mathbb{F}_q$, the last term above is $\mathbf{0}$, and the new form modulo q will be

$$\begin{aligned}
\mathbf{Q} = &\left[\mathbf{0}_k, \mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]\cdot\tilde{\mathbf{T}}\cdot\left[\mathbf{0}_k, \mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]^\mathsf{T} \\
&+ \left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]\cdot[\mathbf{M},\mathbf{I}_h]^\mathsf{T}\cdot[\mathbf{M},\mathbf{I}_h]\cdot\left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]^\mathsf{T} \\
&- \left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]\cdot[\mathbf{0},\mathbf{I}_h]^\mathsf{T}\cdot\mathbf{H}\cdot[\mathbf{0}_k,\mathbf{I}_{n-k}]^\mathsf{T}\cdot\left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]^\mathsf{T} \\
&- \left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]\cdot[\mathbf{0},\mathbf{I}_{n-k-h}]^\mathsf{T}\cdot\mathbf{H}^\mathsf{T}\cdot[\mathbf{0},\mathbf{I}_h]\cdot\left[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]^\mathsf{T} \pmod q.
\end{aligned}$$

The last three terms are $\mathbf{0}$ due to code duality and Corollary 5: in the second term observe that $[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}]$ is dual to $[\mathbf{M},\mathbf{I}_h]$, in the third term $\mathbf{H}\cdot[\mathbf{0}_k,\mathbf{I}_{n-k}]$ is dual to $[\mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}]$, and the same in the fourth term (but transposed). Thus the last form modulo q is

$$\mathbf{Q} = \left[\mathbf{0}_k, \mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]\cdot\tilde{\mathbf{T}}\cdot\left[\mathbf{0}_k, \mathbf{I}_{n-k-h}, -\mathbf{M}^\mathsf{T}\right]^\mathsf{T} \pmod q.$$

To see that $\mathbf{Q}$ is full-rank over $\mathbb{F}_q$, recall from Section 2.2 that $2\cdot(n-k) = \sum_{i=0}^{\infty} i\cdot n_i$. If $n_2 < n-k-h$, then

$$\begin{aligned}\sum_{i=0}^{\infty} i\cdot n_i &\geq 2h + 2n_2 + 3\cdot(n-k-h-n_2) = 3n-3k-h-n_2\\ &= 2\cdot(n-k) + n - k - h - n_2 > 2\cdot(n-k),\end{aligned}$$

a contradiction. Denote $\tilde{\mathbf{M}} = [\mathbf{0}_k, \mathbf{I}_{n-k-h}, -\mathbf{M}^{\mathsf{T}}]$. To see that $\epsilon_2 = \chi_{\mathbf{A}^\perp}$, observe that the matrix $\tilde{\mathbf{M}}\cdot\tilde{\mathbf{T}}$ is an LCD component of $\mathfrak{C}(\mathbf{A})^\perp$. Indeed, its row-span is orthogonal to $\mathfrak{C}(\mathbf{A})$ because of the projection to $\mathfrak{C}(\mathbf{T})^\perp$, and its Gram matrix is precisely $\tilde{\mathbf{M}}\cdot\tilde{\mathbf{T}}\cdot\tilde{\mathbf{T}}^{\mathsf{T}}\cdot\tilde{\mathbf{M}}^{\mathsf{T}} = \tilde{\mathbf{M}}\cdot\tilde{\mathbf{T}}\cdot\tilde{\mathbf{M}}^{\mathsf{T}} = \mathbf{Q}$, where we use both the symmetry and idempotency of the projector $\tilde{\mathbf{T}}$.

Case $h = k$ • degree 1: Condition $\mathbf{A} = \mathbf{H}$ according to Corollary 5 so that there are information sets at $[\![1,k]\!]$ and $[\![n-k+1,n]\!]$, and apply the basis change such that $\mathbf{H} = [\mathbf{V}_k, \mathbf{M}_{n-2k}, \mathbf{I}_k]$, where the subscripts denote the number of columns. Then

$$\mathbf{Q} = \begin{bmatrix} \mathbf{H}\cdot\mathbf{H}^{\mathsf{T}} & q\cdot\mathbf{M} & q\cdot\mathbf{I}_k \\ q\cdot\mathbf{M}^{\mathsf{T}} & q^2\cdot\mathbf{I}_{n-2k} & \mathbf{0} \\ q\cdot\mathbf{I}_k & \mathbf{0} & q^2\cdot\mathbf{I}_k \end{bmatrix}.$$

Since $\mathbf{H}$ is self-orthogonal, we have $\nu_q(\mathbf{Q}) = 1$ and so $n_0 = 0$ (and $\epsilon_0 = 1 = \chi_{\mathbf{H}}$ by convention). Dividing by q yields a new form

$$\mathbf{Q} = \begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{M} & \mathbf{I}_k \\ \mathbf{M}^{\mathsf{T}} & q\cdot\mathbf{I}_{n-2k} & \mathbf{0} \\ \mathbf{I}_k & \mathbf{0} & q\cdot\mathbf{I}_k \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{M} & \mathbf{I}_k \\ \mathbf{M}^{\mathsf{T}} & \mathbf{0} & \mathbf{0} \\ \mathbf{I}_k & \mathbf{0} & \mathbf{0} \end{bmatrix} \pmod q,$$

which has rank $2k$ over $\mathbb{F}_q$. Indeed, as in the previous case, the left kernel of $\mathbf{Q}$ over $\mathbb{F}_q$ is generated by $\mathbf{B} = [\mathbf{0}_k, \mathbf{I}_{n-2k}, -\mathbf{M}^{\mathsf{T}}]$, and the basis complement is generated by

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_k & \mathbf{0}_{n-2k} & \mathbf{0}_k \\ \mathbf{0}_k & \mathbf{0}_{n-2k} & \mathbf{I}_k \end{bmatrix},$$

which gives $n_1 = 2k$. Thus the degree-1 part is

$$\mathbf{Q}_1 = \mathbf{C}\cdot\mathbf{Q}\cdot\mathbf{C}^{\mathsf{T}} = \begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{I}_k \\ \mathbf{I}_k & q\cdot\mathbf{I}_k \end{bmatrix},$$

whose inverse we compute as in the case $h < k$ to be

$$\mathbf{U} = \begin{bmatrix} -q\cdot\tilde{\mathbf{H}}, \mathbf{I} + \tilde{\mathbf{H}}\cdot\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}} \\ \tilde{\mathbf{H}} \qquad -\tilde{\mathbf{H}}\cdot\frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} \end{bmatrix},$$

where $\tilde{\mathbf{H}} = (\mathbf{I}_k - \mathbf{H}\cdot\mathbf{H}^{\mathsf{T}})^{-1}$, which is well-defined over $\mathbb{Z}_q$ because it is just the identity modulo q and thus invertible. Again we get $\epsilon_1 = \left(\frac{\det(-\mathbf{I}_k)}{q}\right) = \left(\frac{-1}{q}\right)^k$.

Case $h = k$ • degree 2: Excising the degree-1 part, we compute

$$\begin{aligned}\mathbf{Q}' &= \mathbf{B}\cdot\left(\mathbf{Q} - \begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{I} \\ \mathbf{M}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{I} & q\cdot\mathbf{I} \end{bmatrix}\cdot\mathbf{U}\cdot\begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{I} \\ \mathbf{M}^{\mathsf{T}} & \mathbf{0} \\ \mathbf{I} & q\cdot\mathbf{I} \end{bmatrix}^{\mathsf{T}}\right)\cdot\mathbf{B}^{\mathsf{T}} = \mathbf{B}\cdot\left(\mathbf{Q} - \begin{bmatrix} \frac{\mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}}{q} & \mathbf{M} & \mathbf{I} \\ \mathbf{M}^{\mathsf{T}} & -q\cdot\mathbf{M}^{\mathsf{T}}\cdot\tilde{\mathbf{H}}\cdot\mathbf{M} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & q\cdot\mathbf{I} \end{bmatrix}\right)\cdot\mathbf{B}^{\mathsf{T}}\\ &= \begin{bmatrix}\mathbf{0}_k & \mathbf{I}_{n-2k} & -\mathbf{M}^{\mathsf{T}}\end{bmatrix}\cdot\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & q\cdot\left(\mathbf{I} + \mathbf{M}^{\mathsf{T}}\cdot\tilde{\mathbf{H}}\cdot\mathbf{M}\right) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}\cdot\begin{bmatrix}\mathbf{0}_k \\ \mathbf{I}_{n-2k} \\ -\mathbf{M}\end{bmatrix} = q\cdot\left(\mathbf{I} + \mathbf{M}^{\mathsf{T}}\cdot\left(\mathbf{I} - \mathbf{H}\cdot\mathbf{H}^{\mathsf{T}}\right)^{-1}\cdot\mathbf{M}\right).\end{aligned}$$

Thus we obtain the degree-2 part

$$\mathbf{Q}_2 = \mathbf{I} + \mathbf{M}^{\mathsf{T}} \cdot \left(\mathbf{I} - \mathbf{H} \cdot \mathbf{H}^{\mathsf{T}}\right)^{-1} \cdot \mathbf{M},$$

and $n_2 = n - 2k$ and $\epsilon_2 = \left(\frac{\det(\mathbf{I}+\mathbf{M}^{\mathsf{T}}\cdot\mathbf{M})}{q}\right)$. Indeed, we need only verify that $\mathbf{G} = \mathbf{I} + \mathbf{M}^{\mathsf{T}} \cdot \mathbf{M}$ is full-rank over $\mathbb{F}_q$. Notice that $\mathbf{G}$ is precisely the Gram matrix of the code generated by $\mathbf{B} = [\mathbf{I}_{n-2k}, -\mathbf{M}^{\mathsf{T}}]$, thus it is sufficient (and necessary) to verify that $\mathbf{B}$ is 0-hollow. This is the case if and only if its dual code is 0-hollow, that is the one generated by $\tilde{\mathbf{M}} = [\mathbf{M}, \mathbf{I}_k]$. By construction we have $\mathbf{H} \cdot \mathbf{H}^{\mathsf{T}} = \mathbf{V} \cdot \mathbf{V}^{\mathsf{T}} + \mathbf{M} \cdot \mathbf{M}^{\mathsf{T}} + \mathbf{I}_k = \mathbf{0}$, therefore we have $\tilde{\mathbf{M}} \cdot \tilde{\mathbf{M}}^{\mathsf{T}} = \mathbf{M} \cdot \mathbf{M}^{\mathsf{T}} + \mathbf{I}_k = -\mathbf{V} \cdot \mathbf{V}^{\mathsf{T}}$, which is full-rank by Corollary 5. Observe also that $\mathbf{B}$ clearly generates an LCD subcode of $\mathfrak{C}(\mathbf{H})^{\perp}$ of dimension $n - 2k$ and is thus its LCD component, hence $\epsilon_2 = \chi_{\mathfrak{C}(\mathbf{H})^{\perp}}$. □

The upshot is the following statement on the possible genera of q-ary lattices.

Corollary 7. *Let $n, k, h, q \in \mathbb{Z}$ with $0 \leq h \leq k \leq n - h$ and q odd prime. For a fixed hull dimension $h < k$, the lattice $\Lambda_q(\mathbf{A})$ of an h-hollow matrix $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ lies in one of two genera, defined by $\chi_{\mathbf{A}} \in \{\pm 1\}$, and denoted by $\mathcal{G}_{n,k,q}^{\chi_{\mathbf{A}},h}$. When $h = k$, all Construction A lattices of generator matrices of self-orthogonal codes lie in a single genus, denoted by $\mathcal{G}_{n,k,q}^{k}$, for a total of $2k + 1$ genera.*

Remark 3. In certain parameter regimes, some of the above genera may not contain any q-ary lattices. The self-dual case is immediate: if $\chi_q(-1)^k \neq 1$, then $\mathcal{G}_{2k,k,q}^{k}$ contains no q-ary lattices as there are no self-dual codes to lift. Perhaps less obviously, the other edge case is when $k \geq \frac{n}{2}$ with hull dimension h such that $n = k + h$. Then the duals of such codes must be self-orthogonal and thus have square class 1. Hence, as we will see below in Corollary 8 from our transference statement, the only genus containing any q-ary lattices is $\mathcal{G}_{k+h,k,q}^{\chi_q(-1)^h,h}$, with the other square class empty. This will not pose a problem for us, as we only sample codes of dimension $k \leq \frac{n}{2}$.

Remark 4. It is now immediate that our distinguisher from Theorem 2 in Figure 1 can be seen from the perspective of LIP as, given a Δ(S)PCE instance $(\mathbf{A}, \mathbf{B})$, computing $(\Lambda_q(\mathbf{A}), \Lambda_q(\mathbf{B}))$ and returning whether $\mathcal{G}(\Lambda_q(\mathbf{A})) = \mathcal{G}(\Lambda_q(\mathbf{B}))$.

The symmetry of the q-adic symbol also yields a transference theorem for genera of q-ary lattices and the fact that the square class of the dual of a self-orthogonal code (which has square class 1) does not depend on the code.

Corollary 8 (Transference). *Let $n, k, h, q \in \mathbb{Z}$ with $0 \leq h \leq k \leq n - h$ and q odd prime. For any h-hollow $\mathbf{A} \in \mathbb{F}_q^{k\times n}$ we have $\chi_{\mathbf{A}^{\perp}} = \chi_q(-1)^h \cdot \chi_{\mathbf{A}}$, so*

$$\mathcal{G}(q \cdot \Lambda_q(\mathbf{A})^{\vee}) = \mathcal{G}(\Lambda_q(\mathbf{A}^{\perp})) = \mathcal{G}(\Lambda_q^{\perp}(\mathbf{A})) = \mathcal{G}_{n,n-k,q}^{\chi_{\mathbf{A}^{\perp}},h}.$$

Proof. From the determinant condition $\epsilon_0 \cdot \epsilon_1 \cdot \epsilon_2 = 1$ in [CS99, Ch. 15.7.7] (Section 2.2) we have $\chi_{\mathfrak{C}(\mathbf{A})} \cdot \chi_q(-1)^h \cdot \chi_{\mathfrak{C}(\mathbf{A})^{\perp}} = 1$ and hence $\chi_{\mathfrak{C}(\mathbf{A})^{\perp}} = \chi_q(-1)^h \cdot \chi_{\mathfrak{C}(\mathbf{A})}$. □

In terms of square classes, Corollary 8 also holds for odd prime powers [LSL25, Prop. 28]; indeed we have $\chi_{\mathbf{A}^{\perp}} = \chi_{\mathbf{A}} \cdot \chi_q(-1)^h$.

4.3 The Spinor Genus of a q-ary Lattice

To ensure we have indeed exhausted the known local information, let us recall that there is an intermediate 'practicably computable' [CS99, p. 388] notion of equivalence that is coarser than integral equivalence and finer than local equivalence (that is genus membership) recently studied in the context of cryptography in [LLM24]. Indeed, each genus may split into multiple *spinor genera* [Eic74], whereas two equivalent quadratic forms always share their spinor genus. For a standard reference, we point the interested reader to [Cas78, Ch. 11], but we omit all the details since our result is negative. This is due to the following fact.

Proposition 6 ([Cas78, Ch. 11 Thm. 1.3]). *Let $\mathcal{L}$ be an integral lattice of dimension $n \geq 3$. If the genus of $\mathcal{L}$ splits into multiple spinor genera, then either $2^{\frac{n\cdot(n-3)}{2}+\left\lfloor\frac{n+1}{2}\right\rfloor}$ divides $\det(\mathcal{L})^2$, or there exists an odd prime p such that $p^{\frac{n\cdot(n-1)}{2}}$ divides $\det(\mathcal{L})^2$.*

Corollary 9. *Let $n, k, q \in \mathbb{Z}$ with $0 \leq k \leq n$, $n \geq 5$, and q odd prime. None of the genera defined in Corollary 7 split into multiple spinor genera.*

Proof. The determinant of every Construction A lattice coming from an $[n, k]$-linear code over $\mathbb{F}_q$ is q^{n-k}. Since q is an odd prime, a necessary condition for a genus to split into multiple spinor genera is that $q^{\frac{n\cdot(n-1)}{2}}$ divides $q^{2\cdot(n-k)}$, which is false for all valid $k \geq 1$ when $n \geq 5$ (true for $k = 1$ and $n \in \{2, 3, 4\}$). □

4.4 The Genus Distribution of Random q-ary Lattices

The authors of [BDG23] were concerned less with local structure in particular than with the genus of a *random* q-ary lattice. In [BDG23, Thm. 2], they show that the two genera in which 0-hollow lattices may fall occur with about the same probability, up to a term of $\mathcal{O}\left(q^{-3}\right)$. Since the genus of a q-ary lattice depends solely on the hull dimension and square class of its underlying linear code, we have refined this statement in Theorem 1 up to a term negligible in the hull co-dimension, by simply replacing h-hollow codes with Construction A lattices of their generator matrices in the statement. We restate it here in terms of lattices for completeness.

Corollary 10 (of Theorem 1). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h < k \leq \frac{n}{2}$, q odd prime and $\chi \in \{\pm 1\}$. The Construction A lattice of a uniformly random h-hollow matrix falls in $\mathcal{G}^{\chi,h}_{n,k,q}$ with probability negligibly close to $\frac{1}{2}$ in $(k-h)\cdot\log(q)$ for both χ. Concretely, denote $p^{\chi,h}_{n,k,q} = \Pr_{\mathbf{A}\leftarrow\$\,\mathsf{D}^h_{n,k,q}}\left[\Lambda_q(\mathbf{A}) \in \mathcal{G}^{\chi,h}_{n,k,q}\right]$. Then*

$$\left|p^{+1,h}_{n,k,q} - p^{-1,h}_{n,k,q}\right| \leq 2q^{-\frac{k-h}{2}}.$$

For uniformly random q-ary lattices, Proposition 3 (from [LSL25]) completely describes the genus distribution.

Corollary 11. *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h \leq k \leq \frac{n}{2}$, q odd prime and $\chi \in \{\pm 1\}$. Denote by $p^{\chi,h}_{n,k,q} = \Pr_{\mathbf{A}\leftarrow\mathbb{F}_q^{k\times n}}\left[\operatorname{rank}(\mathbf{A}\cdot\mathbf{A}^{\mathsf{T}}) = k - h \wedge \chi_{\mathbf{A}} = \chi \;\middle|\; \operatorname{rank}(\mathbf{A}) = k\right]$ derived from the mass formulae in Proposition 3. For $\mathbf{A} \leftarrow\$\, \mathbb{F}_q^{k\times n}$ the lattice $\Lambda_q(\mathbf{A})$ falls in one of the $2k+1$ genera from Corollary 7, $\mathcal{G}^{\chi,h}_{n,k,q}$ for $h \in [\![0, k-1]\!]$ and $\mathcal{G}^{k}_{n,k,q}$, with probability*

$$\Pr_{\mathbf{A}\leftarrow\$\mathbb{F}_q^{k\times n}}\left[\Lambda_q(\mathbf{A}) \in \mathcal{G}^{\chi,h}_{n,k,q}\right] = \prod_{i=0}^{k-1}\left(1 - q^{i-n}\right)\cdot p^{\chi,h}_{n,k,q},$$

where for $h = k$ omit $\chi = 1$.

Proof. A uniformly random matrix from $\mathbb{F}_q^{k\times n}$ has full rank k with probability $\prod_{i=0}^{k-1}\left(1-q^{i-n}\right)$. Conditioned on being full-rank, it is h-hollow of square class χ with probability $p^{\chi,h}_{n,k,q} = \frac{\left|\mathcal{X}^{\chi,h}_{n,k,q}\right|}{\begin{bmatrix}n\\k\end{bmatrix}_q}$. □

To summarise, when we are intentionally sampling hollow lattices, say with $\mathsf{D}^h_{n,k,q}$ for a fixed h, Corollary 10 shows we fall into one of the two possible genera $\mathcal{G}^{\chi,h}_{n,k,q}$ for $\chi \in \{\pm 1\}$ with probability negligibly close to $\frac{1}{2}$, as long as h is not too close to k. Since the probability that a linear code has hull dimension h is negligible in h by Proposition 1, Corollary 11 shows that uniformly random q-ary lattices mostly lie in genera described by small hull dimensions.

4.5 Remarks

We remark on the connection between the Witt decomposition and genus computation, and explore why certain invariants appear to be missing.

Witt type corresponds to the genus. To connect the genus of $\Lambda_q(\mathbf{A})$ with the Witt decomposition of $V_{\mathbf{A}}$, we simply need to account for the code length n, which disappears in $V_{\mathbf{A}}$. Recall from Theorem 3 that n only appears in the dimension n_2 of the degree-2 Jordan component at q and can be recovered from the q-adic symbol for any fixed length n. This explains the last row of Table 1.

Theorem 4. *Let $n, k, q \in \mathbb{Z}$ with $0 \leq k \leq n$ and q odd prime, and let $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times n}$ full-rank. Then $\Lambda_q(\mathbf{A})$ and $\Lambda_q(\mathbf{B})$ lie in the same genus if and only if $V_{\mathbf{A}}$ and $V_{\mathbf{B}}$ are isometric.*

Proof. Combine Theorem 3 and Lemma 4. □

Discriminant over $\mathbb{Q}$. It may seem odd that the discriminant (over $\mathbb{Q}$, or perhaps appropriately generalised over $\mathbb{Z}$) of a Gram matrix does not play a role when considering $\mathbb{Z}$-equivalence of integral lattices, despite being mentioned in passing in [DvW22, Sec. 7.1]. The reason is simple: for $\mathbf{Q}$ to be $\mathbb{Z}$-equivalent to $\mathbf{Q}'$ it is necessary that $\det(\mathbf{Q}) = \det(\mathbf{Q}')$, since they must be congruent via a unimodular matrix. Hence the square classes of the determinants also match.

No Hasse–Minkowski invariant. Readers familiar with the theory of rational quadratic forms may have noticed in Theorem 3 the lack of a third invariant composing the genus, called the Clifford, Witt, Hasse or Hasse–Minkowski invariant depending on how the theory is developed (cf. [Lam05, Ch. V.3] and [Cas78, Ch. 4.1]),[7] which together with the dimension and the discriminant forms a complete system of invariants for local equivalence over $\mathbb{Q}_p$ [Cas78, Ch. 4 Thm. 1.1]. This third invariant maps a quadratic space over $\mathbb{F}$ to an element of the *Brauer group* of $\mathbb{F}$, which for finite fields is trivial [Lam05, Ch. IV Ex. 1.5], hence this invariant does not distinguish q-ary lattices due to Theorem 4.

5 A Refined Hardness Assumption

Section 4 motivates the definition of the *genus of a linear code*, defined simply as the genus of the Construction A lattice of any of its generator matrices, as it is a single convenient invariant that captures both the coding-theoretic perspective of Section 3 and the lattice perspective of Section 4, unified in Theorem 4.

Definition 6 (Genus of a linear code). *We say an $[n, k]$-linear code $\mathfrak{C}$ over $\mathbb{F}_q$ lies in $\mathcal{G}_{n,k,q}^{\chi,h}$ if it is h-hollow with $h < k$, and $\chi_{\mathfrak{C}} = \chi$. All $[n, k]$-linear self-orthogonal codes over $\mathbb{F}_q$ lie in the same genus $\mathcal{G}_{n,k,q}^{k}$, that is $\chi = 1$.*

In this section, we show how to sample uniformly from a specified genus, give a new conjecture on the hardness of (search and decision) (S)PCE, and use it to fix the security of the UPKE scheme from [ABL25a] by replacing the definition of PCE in Definition 1 with our refined hardness assumption that restricts (S)PCE instances to a single genus, which accounts for our distinguisher.

5.1 Sampling Uniformly from a Genus

Since for a fixed hull dimension $h < k$ all codes lie in two genera of the same size (up to an exponentially small relative difference in $(k - h) \cdot \log(q)$) by Corollary 10, limiting the sampling algorithm for $\mathsf{D}_{n,k,q}^{h}$ from [ABL25a, Def. 11] to a particular genus by simply rejecting codes of the wrong square class yields a sampling algorithm for the uniform distribution on that genus. We first define the genus distribution and give a generic statement, then apply it to the case of [ABL25a].

[7] The Clifford, Witt and Hasse invariant do technically refer to different functions, but turn out to be equivalent if the dimension and the discriminant match.

$\mathsf{SampleGenus}(n, k, h, q, \chi)$

$\mathfrak{C} \leftarrow \mathsf{SampleCode}(n, k, h, q)$; **if** $\mathfrak{C} = \perp$ **or** $\mathfrak{C} \notin \mathcal{G}^{\chi,h}_{n,k,q}$ **return** $\perp$ **else return** $\mathfrak{C}$

Fig. 2: The algorithm $\mathsf{SampleGenus}$ that samples from $\mathsf{D}^{\chi,h}_{n,k,q}$ using rejection sampling for the uniform distribution on a subset from a uniform distribution on the superset. The algorithm $\mathsf{SampleCode}$ is defined in [ABL25a, Def. 11].

Definition 7 (Genus distribution). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h \leq k \leq n - h$, q odd prime and $\chi \in \{\pm 1\}$. Denote by $\mathsf{D}^{\chi,h}_{n,k,q}$ the uniform distribution on the set of generator matrices of linear codes in $\mathcal{G}^{\chi,h}_{n,k,q}$, where $\chi = 1$ can be omitted when $h = k$.*

Theorem 5 (Genus sampler). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h < k \leq \frac{n}{2}$, q odd prime and $\chi \in \{\pm 1\}$. Denote $p^{\chi,h}_{n,k,q} = \Pr_{\mathbf{A} \leftarrow\$ \mathsf{D}^h_{n,k,q}}\left[\mathfrak{C}(\mathbf{A}) \in \mathcal{G}^{\chi,h}_{n,k,q}\right]$ as in Corollary 10. Suppose there exists a (t, ε)-algorithm $\mathcal{A}$ that samples from $\mathsf{D}^h_{n,k,q}$. Then there exists a $\left(t + \mathcal{O}\left(n^\omega + \log^2(q)\right), p^{\chi,h}_{n,k,q} \cdot \varepsilon\right)$-algorithm $\mathcal{A}^\chi$ that samples from $\mathsf{D}^{\chi,h}_{n,k,q}$. Moreover, if n, k, h, q are parametrised by λ and $q^{-\frac{k-h}{2}} = \mathsf{negl}(\lambda)$, then the success probability of $\mathcal{A}^\chi$ is negligibly close to $\frac{\varepsilon}{2}$ for both χ.*

Proof. We use rejection sampling for two uniform distributions, one with a known sampler $\mathcal{A}$ on the whole support of $\mathsf{D}^h_{n,k,q}$, that is $\mathcal{G}^{+1,h}_{n,k,q} \sqcup \mathcal{G}^{-1,h}_{n,k,q}$, and the other on $\mathcal{G}^{\chi,h}_{n,k,q}$ for $\chi \in \{\pm 1\}$. In this particular setting, the algorithm $\mathcal{A}^\chi$ always aborts when the sampled code is in the wrong genus, and always accepts when it is in the correct genus, conditioned on $\mathcal{A}$ succeeding (if $\mathcal{A}$ fails then the probability its output is in either genus is 0). Thus the running time increase is due to Corollary 3 and the success probability of $\mathcal{A}^\chi$ is the success probability of $\mathcal{A}$ multiplied by the probability that we sampled from the correct genus, that is $p^{\chi,h}_{n,k,q}$. The case parametrised by λ is due to Corollary 10. □

Corollary 12 ($\mathsf{SampleGenus}$). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $k \leq \frac{n}{2}$, $2h \leq k$,[8] q odd prime, and $\chi \in \{\pm 1\}$. There exists a PPT algorithm $\mathsf{SampleGenus}$ that on input n, k, h, q, χ samples from $\mathsf{D}^{\chi,h}_{n,k,q}$ with success probability*

$$\varepsilon \geq \left(\frac{1}{2} - \mathsf{negl}((k-h) \cdot \log(q))\right) \cdot \left(1 - \frac{1}{q} - \frac{1}{q^2}\right) \cdot (1 - \mathsf{negl}(n)) .$$

Proof. Apply Theorem 5 to the algorithm $\mathsf{SampleCode}$ from [ABL25a, Def. 11] that samples from $\mathsf{D}^h_{n,k,q}$ with probability at least $\left(1 - \frac{1}{q} - \frac{1}{q^2}\right) \cdot (1 - \mathsf{negl}(n))$ by [ABL25a, Lem. 6]. The decrease in the success probability then follows from Corollary 10. The algorithm is given in Figure 2. □

5.2 Permutation Equivalence on a Genus

We refine the hardness assumption on (S)PCE from Definitions 1 and 2 by restricting the problem to a single genus, analogous to the study of LIP. Restricting to a genus thwarts our distinguisher in Figure 1 and we conjecture the restricted (S)PCE problem is hard for every genus parametrised by a large enough hull dimension h regardless of square class.

Definition 8 ((S)PCE). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h \leq k \leq \frac{n}{2}$, q odd prime and $\chi \in \{\pm 1\}$. Denote by $(\mathsf{S})\mathsf{PCE}^{\chi,h}_{n,k,q}$ the (search or decision) problem $(\mathsf{S})\mathsf{PCE}_{n,k,q}(\mathsf{D}^{\chi,h}_{n,k,q})$ on the genus $\mathcal{G}^{\chi,h}_{n,k,q}$ when $h < k$, and, as before, by $(\mathsf{S})\mathsf{PCE}^k_{n,k,q}$ the problem $(\mathsf{S})\mathsf{PCE}_{n,k,q}(\mathsf{D}^k_{n,k,q})$.*

[8] We note that the proof of [ABL25a, Lem. 6] allows for $(1 + c) \cdot h \leq k$ for $c > 0$.

On the one hand the hull dimension and square class form an arithmetically complete system of invariants for quadratic form equivalence over $\mathbb{F}_q$. On the other hand, the (spinor) genus is, to the best of our knowledge, the finest efficiently computable equivalence relation on integral quadratic forms. This increases our confidence in the hardness of (S)PCE when restricted to a particular genus.

Conjecture 1 (Fixed-genus (S)PCE*).* Let $n, k, h, q, \chi \in \mathbb{Z}$ be parametrised by λ with $h \geq \Omega(\lambda)$, $0 \leq h \leq k \leq \frac{n}{2}$, q odd prime, $\chi \in \{\pm 1\}$, $h > \sqrt{2n}$ and $\binom{n}{\lceil h - \sqrt{2n} \rceil} \geq 2^\lambda$ (due to [BMS26], see Section 1.2). The advantage of any PPT adversary against search or decision $(\mathsf{S})\mathsf{PCE}^{\chi,h}_{n,k,q}$ (with $\chi = 1$ when $h = k$) is $\mathsf{negl}(\lambda)$.

5.3 Mending the UPKE Scheme from [ABL25a]

We now replace the $\Delta\mathsf{PCE}^{h}_{n,k,q}$ hardness assumption underpinning the security of the UPKE scheme from [ABL25a] with either of its genus restrictions $\Delta\mathsf{PCE}^{\chi,h}_{n,k,q}$ for $\chi \in \{\pm 1\}$. Concretely, we will mend the game hop in their security proof of [ABL25a, Thm. 3] that uses $\Delta\mathsf{PCE}^{h}_{n,k,q}$, which is conveniently isolated as [ABL25a, Lem. 11], instead relying on Conjecture 1 for security.

Lemma 6 (Restriction of [ABL25a, Lem. 11] to a genus). *Let $n, k, h, q, \chi \in \mathbb{Z}$ with $0 \leq h \leq k \leq \frac{n}{2}$, q odd prime, and $\chi \in \{\pm 1\}$ but always $\chi = 1$ if $h = k$. If there exists a (t, ε)-algorithm $\mathcal{A}$ that distinguishes*

$$\left(\mathbf{A} \leftarrow\!\$\, \mathsf{D}^{\chi,h}_{n,k,q}, \mathbf{u} \coloneqq \mathbf{A} \cdot \mathbf{r}, \mathbf{B} \coloneqq \mathbf{U} \cdot \mathbf{A} \cdot \mathbf{O}, \mathbf{u}' \coloneqq \mathbf{B} \cdot \mathbf{r}', \mathbf{r}' \coloneqq \mathbf{O}^\intercal \cdot \mathbf{r} \;\middle|\; \mathbf{r} \leftarrow\!\$\, \{\pm 1\}^n, \mathbf{O} \leftarrow\!\$\, \mathcal{O}_n(\mathbb{Z}), \mathbf{U} \leftarrow\!\$\, \mathcal{GL}_k(\mathbb{F}_q)\right)$$

from

$$\left(\mathbf{A} \leftarrow\!\$\, \mathsf{D}^{\chi,h}_{n,k,q}, \mathbf{u} \coloneqq \mathbf{A} \cdot \mathbf{r}, \mathbf{B} \leftarrow\!\$\, \mathsf{D}^{\chi,h}_{n,k,q}, \mathbf{u}' \coloneqq \mathbf{B} \cdot \mathbf{r}', \mathbf{r}' \leftarrow\!\$\, \{\pm 1\}^n \;\middle|\; \mathbf{r} \leftarrow\!\$\, \{\pm 1\}^n\right),$$

then there exists a $(t + \mathcal{O}(n^\omega), \varepsilon)$-algorithm $\mathcal{B}$ against $\Delta\mathsf{PCE}^{\chi,h}_{n,k,q}$.

Proof. The proof is exactly the same as that for [ABL25a, Lem. 11], since the reduction is genus preserving (Corollary 2), though we will strip the scheme-specific terminology from the idea. The reduction $\mathcal{B}$ on input a $\Delta\mathsf{PCE}^{\chi,h}_{n,k,q}$ instance $(\mathbf{A}, \mathbf{B})$ assigns to $\mathbf{A}$ the tuple $(\mathbf{A}, \mathbf{a}, [1]^n)$ where $\mathbf{a} \in \mathbb{F}_q^k$ is the sum of the columns of $\mathbf{A}$, and to $\mathbf{B}$ the tuple $(\mathbf{B}, \mathbf{b}, [1]^n)$ where $\mathbf{b} \in \mathbb{F}_q^k$ is the sum of the columns of $\mathbf{B}$, and $[1]^n$ is the all-1's vector. It is clear that $\mathbf{a} = \mathbf{A} \cdot [1]^n$ and $\mathbf{b} = \mathbf{B} \cdot [1]^n$. The reduction then re-randomises the instance by sampling $\mathbf{O_A}, \mathbf{O_B} \leftarrow\!\$\, \mathcal{O}_n(\mathbb{Z})$ and $\mathbf{U_A}, \mathbf{U_B} \leftarrow\!\$\, \mathcal{GL}_k(\mathbb{F}_q)$, and computing

$$\begin{aligned} \tilde{\mathbf{A}} &= \mathbf{U_A} \cdot \mathbf{A} \cdot \mathbf{O_A}, & \tilde{\mathbf{a}} &= \mathbf{U_A} \cdot \mathbf{a}, & \mathbf{r_A} &= \mathbf{O_A^\intercal} \cdot [1]^n, \\ \tilde{\mathbf{B}} &= \mathbf{U_B} \cdot \mathbf{B} \cdot \mathbf{O_B}, & \tilde{\mathbf{b}} &= \mathbf{U_B} \cdot \mathbf{b}, & \mathbf{r_B} &= \mathbf{O_B^\intercal} \cdot [1]^n. \end{aligned}$$

$\mathcal{B}$ then calls $\mathcal{A}$ on $(\tilde{\mathbf{A}}, \tilde{\mathbf{a}}, \tilde{\mathbf{B}}, \tilde{\mathbf{b}}, \mathbf{r_B})$, and outputs the bit it receives.

Since $[1]^n$ is fixed by any permutation, we have that if $\mathbf{B} = \mathbf{U} \cdot \mathbf{A} \cdot \mathbf{P}$, then $\mathbf{O_A^{-1}} \cdot \mathbf{P} \cdot \mathbf{O_B}$ is a uniformly random signed permutation (since $\mathbf{O_A}$ and $\mathbf{O_B}$ are sampled independently and uniformly) that maps $\mathfrak{C}(\tilde{\mathbf{A}})$ to $\mathfrak{C}(\tilde{\mathbf{B}})$, and its transpose maps $\mathbf{r_A}$ to $\mathbf{r_B}$. Thus the tuple submitted to $\mathcal{A}$ follows the first distribution. If $\mathbf{A}$ and $\mathbf{B}$ are independently sampled from $\mathsf{D}^{\chi,h}_{n,k,q}$, then the submitted tuple clearly follows the second distribution. □

It is immediate from Theorem 1 that as soon as $q^{k-h} \gtrsim 2^\lambda$, the Hollow LWE assumption [ABL25a, Thm. 1] and the Hollow LHL [ABL25a, Lem. 9] also hold if h-hollow matrices are additionally restricted to a particular square class χ. From the perspective of Section 4, if hollow lattice problems are hard for q-ary lattices of a particular hull dimension, then they are also hard on q-ary lattices from a genus parametrised by that hull dimension.

The UPKE scheme from [ABL25a] can now be fixed in two ways. The simplest way is to pick a genus in advance, that is $\mathcal{G}^{\chi,h}_{n,k,q}$ for $\chi \in \{\pm 1\}$ and an appropriately large h to account for known attacks, and then replace the algorithm SampleCode from [ABL25a, Def. 11] with SampleGenus from Corollary 12 everywhere

it appears, that is in the key generation procedure KGen and in the security proof of [ABL25a, Thm. 3] in the game hop that uses [ABL25a, Lem. 11]. This lemma is then replaced with our Lemma 6. The other way is to leave KGen as is and condition the security proof on the genus of the KGen output, that is, we only replace SampleCode with SampleGenus for the correct $\chi \in \{\pm 1\}$ (based on the output of KGen) in the security proof. While the first option is conceptually simpler, it incurs a factor of 2 running time increase in KGen (in expectation), which the second option avoids. In both cases the adversary's advantage in [ABL25a, Thm. 3] against $\Delta\mathsf{PCE}^{h}_{n,k,q}$ is simply replaced by its advantage against $\Delta\mathsf{PCE}^{\chi,h}_{n,k,q}$.

6 Experiments

Our experiment code is in Appendix C, and also attached to this document, and consists of a SageMath [The25] script `local.py` containing all the following functions. We also include a slightly modified copy of `selforthogonal.py` [ABL25b] that our code depends on with the dehulling procedure from Proposition 4 implemented.

We implement our distinguisher from Figure 1 in `SqClassD` and observe in `test_SqClassD` that it indeed wins with probability roughly $\frac{3}{4}$ for any valid parameters we give it, as claimed in Theorem 2 and Corollary 4. We observe that our distinguisher behaves the same even for the edge cases outlined in Theorem 2. By Corollary 3, our distinguisher is efficient and works for any parameters that work with [ABL25a], even accounting for [BMS26], and thus we do not report its practical performance. In fact most of the running time in `test_SqClassD` is used to sample the instances. The sampler from Figure 2 is in `SampleGenus`. We expand on Remarks 1 and 2 in Appendix B by experimentally comparing the genus distribution of random non-binary Goppa codes and codes obtained through LCE to that of random codes. Experiments can be re-run using `produce_data`, which produces numbers similar to Table 2 and plots similar to Figures 3 and 4.

7 Future Directions

We outline potential future directions and consider the limitations of our approach.

Characteristic 2. As discussed briefly in Remark 1, our square class invariant is not useful in characteristic 2 since all binary codes have square class 1. However, there is an additive version of the discriminant that works over fields of characteristic 2 called the *Arf invariant* [Arf41], which plays essentially the same role as the discriminant in characterising equivalent quadratic forms. It is not immediate how the analogous generalisation of the Arf invariant corresponds to genera of binary lattices (cf. Theorem 3), as the arithmetic theory locally over $\mathbb{Z}_2$ is anomalous; the Jordan decomposition is not necessarily unique, and, besides dimensions and signs, we also have to consider *oddity* in the dyadic symbol. Moreover, the algorithmic theory of hollow codes would need to be extended to make any results cryptographically relevant, for example the sampler from [ABL25a, Def. 11] would have to be extended to characteristic 2 so that a binary version of Theorem 5 could even be applied.

Linear equivalence. We noted in Remark 2 that neither the hull dimension nor the square class are invariants for LCE, but Section 3 reveals a simple reason why this is the case: the group of monomial matrices $\mathcal{M}_n(\mathbb{F}_q)$ is not a subgroup of $\mathcal{O}_n(\mathbb{F}_q)$, in fact $\mathcal{M}_n(\mathbb{F}_q) \cap \mathcal{O}_n(\mathbb{F}_q) = \mathcal{O}_n(\mathbb{Z})$. Invariants do exist for LCE, for example the Hamming weight enumerator polynomial, though to the best of our knowledge none have been found that can be computed efficiently. Studying potential arithmetic invariants for LCE thus marks a target for future work.

Relation to LIP. The connection between flavours of LIP and SPCE can be formalised, namely [DG23, Lem. 9] provides a reduction from another type of LIP on q-ary lattices to $\mathbb{Z}$LIP [BGPS23] and SPCE, but only for 0-hollow lattices. Similarly to how we extended [BDG23] in Theorem 3 to arbitrary hull dimension, extending the above reduction of LIP on q-ary lattices to $\mathbb{Z}$LIP and SPCE to all hollow lattices remains an open problem. We note this was recently solved in [BHJ26] for uniformly random codes with a Monte Carlo-style algorithm

when $q < n^{\frac{1}{2}-\varepsilon}$ prime for $\varepsilon > 0$. This bound assures the isometric image of the sublattice $q \cdot \mathbb{Z}^n$ is expected to contain the shortest vectors of the lattice, which the algorithm finds, avoiding the hull. However, it is not immediate how this algorithm behaves for codes sampled from a fixed genus and as q grows (for example $q > n$ in [ABL25a]).

Acknowledgements

We thank Martin R. Albrecht for fruitful discussions and helpful feedback.

References

ABL25a. Martin R. Albrecht, Benjamin Benčina, and Russell W. F. Lai. Hollow LWE: A new spin - unbounded updatable encryption from LWE and PCE. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 363–392. Springer, Cham, May 2025. https://doi.org/10.1007/978-3-031-91101-9_13. 2, 3, 4, 5, 6, 13, 22, 23, 24, 25, 26

ABL25b. Martin R. Albrecht, Benjamin Benčina, and Russell W. F. Lai. Hollow LWE: A new spin - unbounded updatable encryption from LWE and PCE, May 2025. https://doi.org/10.1007/978-3-031-91101-9_13. Artifact available at https://artifacts.iacr.org/eurocrypt/2025/a9. 25

ABL25c. Martin R. Albrecht, Benjamin Benčina, and Russell W. F. Lai. Hollow LWE: A new spin, unbounded updatable encryption from LWE and PCE. Cryptology ePrint Archive, Report 2025/340, 2025. https://eprint.iacr.org/2025/340. 10

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. https://doi.org/10.1145/237814.237838. 1, 2

Arf41. Cahit Arf. Untersuchungen über quadratische Formen in Körpern der Charakteristik 2. (Teil I.). *Journal für die reine und angewandte Mathematik*, 1941(183):148–167, 1941. https://doi.org/10.1515/crll.1941.183.148. 25

Art57. Emil Artin. *Geometric Algebra*, volume 3 of *Interscience Tracts in Pure and Applied Mathematics*. Interscience Publishers, Inc., 1957. 8

BBPS21. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. LESS-FM: Fine-tuning signatures from the code equivalence problem. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, pages 23–43. Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-81293-5_2. 30

BDG23. Peter Bruin, Léo Ducas, and Shane Gibbons. Genus distribution of random q-ary lattices. *Banach Center Publications*, 126:137–159, 2023. https://doi.org/10.4064/bc126-9. 1, 2, 3, 8, 14, 16, 21, 25

Ber09. Dennis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas*. Princeton University Press, second edition, 2009. 16, 18

Beu20. Ward Beullens. Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 387–403. Springer, Cham, October 2020. https://doi.org/10.1007/978-3-030-81652-0_15. 3

BGPS23. Huck Bennett, Atul Ganju, Pura Peetathawatchai, and Noah Stephens-Davidowitz. Just how hard are rotations of $\mathbb{Z}^n$? algorithms and cryptography with the simplest lattice. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 252–281. Springer, Cham, April 2023. https://doi.org/10.1007/978-3-031-30589-4_9. 1, 25

BHJ26. Jean-Francois Biasse, Alexandra Hostetler, and Anuvrat Jaindungarwal. On the hull attacks against Construction A lattices. In *Selected Areas in Cryptography*, 2026. https://sacworkshop.org/SAC26/preproceedings/SAC2026_1-final42.pdf. 25

BLP11. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Wild McEliece incognito. In Bo-Yin Yang, editor, *PQCrypto 2011*, pages 244–254. Springer, Berlin, Heidelberg, November / December 2011. https://doi.org/10.1007/978-3-642-25405-5_16. 29

BM23. Jean-François Biasse and Giacomo Micheli. A Search-to-Decision Reduction for the Permutation Code Equivalence Problem. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 602–607. IEEE, 2023. https://doi.org/10.1109/isit54713.2023.10206940. 4

BMS26. Michele Battagliola, Rocco Mora, and Paolo Santini. Using the Schur Product to Solve the Code Equivalence Problem. *IEEE Transactions on Information Theory*, 72(7):4675–4694, 2026. https://doi.org/10.1109/TIT.2026.3694630. 3, 4, 6, 24, 25

BOST19. Magali Bardet, Ayoub Otmani, and Mohamed Saeed-Taha. Permutation Code Equivalence is Not Harder Than Graph Isomorphism When Hulls Are Trivial. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2464–2468. IEEE, 2019. https://doi.org/10.1109/ISIT.2019.8849855. 3, 5

BW25. Huck Bennett and Kaung Myat Htay Win. Relating code equivalence to other isomorphism problems. *DCC*, 93(3):701–723, 2025. https://doi.org/10.1007/s10623-024-01542-3. 2

Cas78. John William Scott Cassels. *Rational Quadratic Forms*, volume 13 of *London Mathematical Society Monographs*. Academic Press, 1978. 14, 20, 21, 22

Che35. Claude Chevalley. Démonstration d'une hypothèse de M. Artin. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):73–75, 1935. https://doi.org/10.1007/BF02940714. 9

Claa. Pete L. Clark. Quadratic Forms Chapter I: Witt's Theory. https://people.reed.edu/~ormsbyk/kgroup/resources/Pete_Clark_Quadratic_Forms.pdf. 8

Clab. Pete L. Clark. Quadratic Forms Chapter II: Structure of the Witt Ring. https://people.reed.edu/~ormsbyk/kgroup/resources/Pete_Clark_Quadratic_Forms_2.pdf. 8

CMTQ19. Claude Carlet, Sihem Mesnager, Chunming Tang, and Yanfeng Qi. New Characterization and Parametrization of LCD Codes. *IEEE Transactions on Information Theory*, 65(1):39–49, 2019. https://doi.org/10.1109/TIT.2018.2829873. 6, 9, 11, 28

Coh93. Henri Cohen. *A Course in Computional Algebraic Number Theory*. Graduate Texts in Mathematics. Springer, 1993. https://doi.org/10.1007/978-3-662-02945-9. Third, Corrected Printing 1996. 10

COT14. Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial time attack on wild McEliece over quadratic extensions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 17–39. Springer, Berlin, Heidelberg, May 2014. https://doi.org/10.1007/978-3-642-55220-5_2. 29

CPS23. Tung Chou, Edoardo Persichetti, and Paolo Santini. On linear equivalence, canonical forms, and digital signatures. Cryptology ePrint Archive, Report 2023/1533, 2023. https://eprint.iacr.org/2023/1533. 3

CS99. John H. Conway and Neil J.A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290 of *Grundlehren der mathematischen Wissenschaften*. Springer, 3rd edition, 1999. https://doi.org/10.1007/978-1-4757-6568-7. 1, 4, 5, 16, 20

DG23. Léo Ducas and Shane Gibbons. Hull attacks on the lattice isomorphism problem. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 177–204. Springer, Cham, May 2023. https://doi.org/10.1007/978-3-031-31368-4_7. 2, 25

DvW22. Léo Ducas and Wessel P. J. van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 643–673. Springer, Cham, May / June 2022. https://doi.org/10.1007/978-3-031-07082-2_23. 1, 22

Eic74. Martin Eichler. *Quadratische Formen und orthogonale Gruppen*, volume 63 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2 edition, 1974. https://doi.org/10.1007/978-3-642-80764-0. 3, 14, 20

Gop70. Valery Denisovich Goppa. A New Class of Linear Correcting Codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970. http://mi.mathnet.ru/ppi1748. 29

Gop88. Valery Denisovich Goppa. *Geometry and Codes*, volume 24 of *Mathematics and its Applications (Soviet Series)*. Springer, 1988. https://doi.org/10.1007/978-94-015-6870-8. 29

Lam05. Tsit-Yuen Lam. *Introduction to Quadratic Forms over Fields*, volume 67 of *Graduate Studies in Mathematics*. American Mathematical Society, 2005. https://doi.org/10.1090/gsm/067. 8, 22

Leo82. Jeffrey Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 1982. https://doi.org/10.1109/TIT.1982.1056498. 3

LLM24. Cong Ling, Jingbo Liu, and Andrew Mendelsohn. On the spinor genus and the distinguishing lattice isomorphism problem. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 329–358. Springer, Singapore, December 2024. https://doi.org/10.1007/978-981-96-0894-2_11. 20

LSL25. Shitao Li, Minjia Shi, and San Ling. A Mass Formula for Linear Codes With Prescribed Hull Dimension and Related Classification. *IEEE Transactions on Information Theory*, 71(1):273–286, 1025. https://doi.org/10.1109/TIT.2024.3425058. 3, 6, 7, 11, 20, 21

Mas92. James L. Massey. Linear codes with complementary duals. *Discrete Mathematics*, 106/107:337–342, 1992. https://doi.org/10.1016/0012-365X(92)90563-U. 2, 5

Mas98a. James L. Massey. On Antiorthogonal Matrices and Their Codes. In *1998 IEEE International Symposium on Information Theory (ISIT)*, page 64, Cambridge, MA, USA, 1998. IEEE. https://doi.org/10.1109/isit.1998.708647. 15

Mas98b. James L. Massey. Orthogonal , Antiorthogonal and Self-Orthogonal Matrices and their Codes. In M. Darnell and B. Honary, editors, *Communications and Coding*, pages 3–9, Somerset, England, 1998. Research Studies Press. 15

McE78. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. 13, 29

Now25. Julian Nowakowski. An improved algorithm for code equivalence. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part I*, pages 71–103. Springer, Cham, April 2025. https://doi.org/10.1007/978-3-031-86599-2_3. 3

NRS06. Gabriele Nebe, Eric M. Rains, and Neil J.A. Sloane. *Self-Dual Codes and Invariant Theory*, volume 17 of *Algorithms and Computation in Mathematics*. Springer, 2006. https://doi.org/10.1007/3-540-30731-1. 9

Ple65. Vera Pless. The Number of Isotropic Subspaces in a Finite Geometry. *Atti della Accademia Nazionale dei Lincei. Classe di Scienze Fisiche, Matematiche e Naturali. Rendiconti, Serie VIII*, 39(6):418–421, 1965. http://www.bdim.eu/item?id=RLINA_1965_8_39_6_418_0. 6, 9

Ple68. Vera Pless. On the uniqueness of the Golay codes. 5(3):215–228, 1968. https://doi.org/10.1016/S0021-9800(68)80067-5. 6, 9

Ran25. Hugues Randriambololona. The syzygy distinguisher. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VI*, volume 15606 of *LNCS*, pages 324–354. Springer, Cham, May 2025. https://doi.org/10.1007/978-3-031-91095-1_12. 13

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. https://doi.org/10.1145/1060590.1060603. 1

Seg59. Beniamino Segre. Le geometrie di Galois. *Annali di Matematica Pura ed Applicata*, 48(1):1–96, 1959. https://doi.org/10.1007/bf02410658. 6

Sen97. Nicolas Sendrier. On the Dimension of the Hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293, 1997. https://doi.org/10.1137/S0895480195294027. 3, 6, 29, 30

Sen00. Nicolas Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000. https://doi.org/10.1109/18.850662. 2, 3, 5, 6, 13, 29

Ser73. Jean-Pierre Serre. *A Course in Arithmetic*, volume 7 of *Graduate Texts in Mathematics*. Springer, 1973. https://doi.org/10.1007/978-1-4684-9884-4. 4

Sie41. Carl Ludwig Siegel. Equivalence of Quadratic Forms. *American Journal of Mathematics*, 63(3):658–680, 1941. https://doi.org/10.2307/2371381. 5

SS13. Nicolas Sendrier and Dimitris E. Simos. The hardness of code equivalence over and its application to code-based cryptography. In Philippe Gaborit, editor, *PQCrypto 2013*, pages 203–216. Springer, Berlin, Heidelberg, June 2013. https://doi.org/10.1007/978-3-642-38616-9_14. 1, 2, 3, 5

The25. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.9)*, 2025. https://www.sagemath.org. 25

van24. Wessel P. J. van Woerden. Dense and smooth lattices in any genus. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 386–417. Springer, Singapore, December 2024. https://doi.org/10.1007/978-981-96-0894-2_13. 1

War35. Ewald Warning. Bemerkung zur vorstehenden Arbeit von Herrn Chevalley. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):76–83, 1935. https://doi.org/10.1007/BF02940715. 9

A Elementary Proof of the Square Class Invariant

Let us show the square class of $\det(\mathbf{T} \cdot \mathbf{T}^{\mathsf{T}})$ is an invariant for (S)PCE without the arithmetic theory of quadratic forms over finite fields.

Lemma 7 (Square class invariant). *Let $\mathfrak{C}_0$ and $\mathfrak{C}_1$ be signed permutation equivalent $[n,k]$-linear codes over $\mathbb{F}_q$ with any LCD components $\mathbf{T}_0$ and $\mathbf{T}_1$, respectively. Then $\chi_q(\det(\mathbf{T}_0 \cdot \mathbf{T}_0^{\mathsf{T}})) = \chi_q(\det(\mathbf{T}_1 \cdot \mathbf{T}_1^{\mathsf{T}}))$.*

Proof. By [CMTQ19, Thm. 25], there exists a basis for $\mathfrak{C}(\mathbf{T}_i)$ for $i \in \{0,1\}$ such that its Gram matrix is either $\mathrm{diag}(1,\dots,1)$ or $\mathrm{diag}(1,\dots,1,s)$ where s is any quadratic non-residue of $\mathbb{F}_q$. The determinant of this

Gram matrix is then either 1 or s, and since a basis change $\mathbf{U} \in \mathcal{GL}_k(\mathbb{F}_q)$ adds a square factor $\det(\mathbf{U})^2$ to the determinant, this determines the square class of the determinant of any LCD component. There exists some basis for $\mathfrak{C}_0$ and $\mathfrak{C}_1$ with LCD components $\mathbf{T}_0$ and $\mathbf{T}_1$ such that if $\mathbf{O} \in \mathcal{O}_n(\mathbb{Z})$ maps $\mathfrak{C}_0$ to $\mathfrak{C}_1$, then $\mathbf{T}_1 = \mathbf{U} \cdot \mathbf{T}_0 \cdot \mathbf{O}$ for some $\mathbf{U} \in \mathcal{GL}_k(\mathbb{F}_q)$. Indeed, let $\mathbf{H}_i$ be any basis of $\mathrm{hull}(\mathfrak{C}_i)$ for $i \in \{0, 1\}$, and let $\mathbf{A}$ be the basis of $\mathfrak{C}_0$ that is $\mathbf{T}_0$ stacked on $\mathbf{H}_0$ (say the one obtained from Proposition 4). Then $\mathbf{A} \cdot \mathbf{O}$ generates $\mathfrak{C}_1$, but this matrix is just $\mathbf{T}_0 \cdot \mathbf{O}$ stacked on $\mathbf{H}_0 \cdot \mathbf{O}$. Since $\mathbf{H}_0 \cdot \mathbf{O}$ generates $\mathrm{hull}(\mathfrak{C}_1)$ by [Sen00, Prop. 12], $\mathbf{T}_0 \cdot \mathbf{O}$ must be an LCD component of $\mathfrak{C}_1$. Thus the square class of the determinant of the Gram matrix of $\mathbf{T}_0$ equals that of $\mathbf{T}_1$, since $\mathbf{O} \cdot \mathbf{O}^\mathsf{T} = \mathbf{I}_n$ and so $\det(\mathbf{T}_1 \cdot \mathbf{T}_1^\mathsf{T}) = \det(\mathbf{U})^2 \cdot \det(\mathbf{T}_0 \cdot \mathbf{T}_0^\mathsf{T})$. □

B Extended Remarks and Experiments

B.1 On Remark 1: Non-binary Goppa Codes

As discussed in Remark 1, our square class invariant cannot be used to distinguish binary codes, in particular we cannot use it to distinguish the class of the permutations of binary Goppa codes [Gop70], that is McEliece [McE78] public-keys, from uniformly random binary codes of the same length and dimension. We consider whether a local distinguishing attack can be mounted on *non-binary Goppa codes*, in other words, is the genus distribution of uniformly random Goppa codes over a finite field of odd characteristic with a fixed set of parameters noticeably different from that of uniformly random codes of the same length and dimension? We do not attempt to answer this question formally and leave it as potential future work, however we provide in this section experiment data suggesting the answer to our question is *negative*, that is the two distributions may not be efficiently distinguishable using local invariants. We note that while we are not aware of random non-binary Goppa codes being used in cryptographic constructions, a special subclass of such codes was used in the so-called 'Wild McEliece' scheme [BLP11] and broken (over quadratic extensions) using a distinguishing attack [COT14].

We compared the genus distribution of random Goppa codes over $\mathbb{F}_q$ for $q \in \{3, 5, 7, 11\}$ with parameters $m = 4$, $t = 9$, $n = 72$ and hence $k = n - m \cdot t = 36$ (in general it is possible that $k \geq n - m \cdot t = 36$, but we did not observe this event), and that of uniformly random $[n, k]$-linear codes over $\mathbb{F}_q$, using 10^5 independent samples for each code family and prime q. By a 'random' Goppa code with parameters (m, t, n), we mean that we pick a uniformly random square-free monic polynomial g of degree t over $\mathbb{F}_{q^m}$ with zero set $\mathcal{V}_g$ and a uniformly random set $L \subseteq \mathbb{F}_{q^m} \setminus \mathcal{V}_g$ of size $|L| = n$, and consider the $[n, k \geq n - t \cdot m]$-linear code

$$\Gamma(g, L) = \left\{ \mathbf{v} \in \mathbb{F}_q^n \;\middle|\; \sum_{i=1}^{n} \frac{\mathbf{v}[i]}{X - L[i]} = 0 \pmod{g} \right\} \leq \mathbb{F}_q^n.$$

To see that this is well-defined, observe that since $L[i] \notin \mathcal{V}_g$, we have that $\gcd(X - L[i], g) = 1$ in $\mathbb{F}_{q^m}[X]$ and so the rational function $(X - L[i])^{-1}$ is in fact a polynomial modulo g, inducing linear constraints on the entries of $\mathbf{v}$.[9] We observe through the above experiment that for the above distribution $\mathsf{D}^{\Gamma}_{n,k,q}$ of Goppa codes and any genus $\mathcal{G}$ of a q-ary lattice from Corollary 7 we have

$$\Pr_{\mathbf{A} \leftarrow \$ \mathbb{F}_q^{k \times n}}[\mathcal{G} = \mathcal{G}(\Lambda_q(\mathbf{A}))] \approx \Pr_{\mathbf{A} \leftarrow \$ \mathsf{D}^{\Gamma}_{n,k,q}}[\mathcal{G} = \mathcal{G}(\Lambda_q(\mathbf{A}))].$$

The distributions are plotted in Figure 3, with Goppa codes on the left and random codes on the right. We also compare both numbers to the asymptotically expected number of codes with a particular hull dimension [Sen97] in a sample size of 10^5 in Table 2, but divided by 2, showing the 2 possible genera for a fixed hull dimension described in Corollary 7 are indeed close in size.

[9] Codes defined by rational curves are sometimes called *rational codes* [Gop88], though the term algebraic geometry (AG) codes is more common.

B.2 On Remark 2: Non-Applicability to LCE

A natural consideration is whether using the square class of the code can help us distinguish *Linear Code Equivalence* (LCE), used for example in the LESS digital signature scheme [BBPS21] submitted to NIST as a potential candidate for post-quantum signatures. Linear equivalence is the action of monomial matrices $\mathcal{M}_n(\mathbb{F}_q)$ on linear codes, these are matrices that have exactly one non-zero element from $\mathbb{F}_q^\star$ in each row and column. We experimentally verify that the square class of a linear code is *not an invariant* for linear equivalence, and further observe that the genus distribution of the codes obtained by sampling $\mathbf{A} \leftarrow\!\$\ \mathbb{F}_q^{k\times n}$ and, provided it is full rank, acting on it with monomial matrices $\mathcal{M}_n(\mathbb{F}_q)$ looks like the genus distribution of uniformly random linear codes. That is, we observe through the above experiment that for $\mathbf{B} \leftarrow\!\$\ \mathbb{F}_q^{k\times n}$ full-rank with $q \geq 5$ and any genus $\mathcal{G}$ of a q-ary lattice from Corollary 7 we have

$$\Pr_{\mathbf{A}\leftarrow\!\$\mathbb{F}_q^{k\times n}}[\mathcal{G} = \mathcal{G}(\Lambda_q(\mathbf{A}))] \approx \Pr_{\mathbf{M}\leftarrow\!\$\mathcal{M}_n(\mathbb{F}_q)}[\mathcal{G} = \mathcal{G}(\Lambda_q(\mathbf{B}\cdot\mathbf{M}))].$$

The parameters we pick are $n = 72$, $k = 36$ and $q \in \{5, 7, 11\}$ for comparability with our experiments on non-binary Goppa codes in Appendix B.1. We omit $q = 3$ since of course $\mathcal{M}_n(\mathbb{F}_3) = \mathcal{O}_n(\mathbb{Z})$, whose action preserves the genus; such a genus distribution is constant, giving $\mathcal{G}(\Lambda_q(\mathbf{B}))$ with probability 1. The distributions are plotted in Figure 4, with the codes obtained with the action of monomial matrices on the left and random codes on the right. We also compare both numbers to the asymptotically expected number of codes with a particular hull dimension [Sen97] in a sample size of 10^5 in Table 2, as in Appendix B.1.

B.3 Tables and Figures

Table 2: Experiment data from observing the genus distribution of random non-binary Goppa codes in Appendix B.1 and that for random codes from the orbit of the linear equivalence action on a random code in Appendix B.2. Numbers represent the observed number of Goppa, linearly equivalent, and random codes in a particular genus over 10^5 samples per type. Expected values are rounded to the nearest integer (there were no ties).

Type	q	$\chi_{\mathfrak{C}}$	$h=0$	$h=1$	$h=2$	$h=3$	$h=4$
Goppa	3	1	31926	15940	2098	77	0
Goppa	3	-1	31992	15867	2030	69	1
random	3	1	31874	16087	2026	93	1
random	3	-1	31916	15991	1939	72	1
expected	3	–	31950	15975	1997	77	1
Goppa	5	1	39742	9964	403	5	0
Goppa	5	-1	39641	9839	403	3	0
LCE	5	1	39734	9915	417	4	0
LCE	5	-1	39528	10018	382	2	0
random	5	1	39702	10009	413	1	0
random	5	-1	39544	9921	408	2	0
expected	5	–	39667	9917	413	3	0
Goppa	7	1	42833	7112	156	0	0
Goppa	7	-1	42750	7000	149	0	0
LCE	7	1	42941	7044	153	1	0
LCE	7	-1	42669	7046	146	0	0
random	7	1	42733	7234	147	0	0
random	7	-1	42558	7192	136	0	0
expected	7	–	42730	7122	148	0	0
Goppa	11	1	45545	4574	34	0	0
Goppa	11	-1	45364	4441	42	0	0
LCE	11	1	45394	4444	35	0	0
LCE	11	-1	45568	4524	35	0	0
random	11	1	45615	4466	40	0	0
random	11	-1	45368	4478	33	0	0
expected	11	–	45420	4542	38	0	0

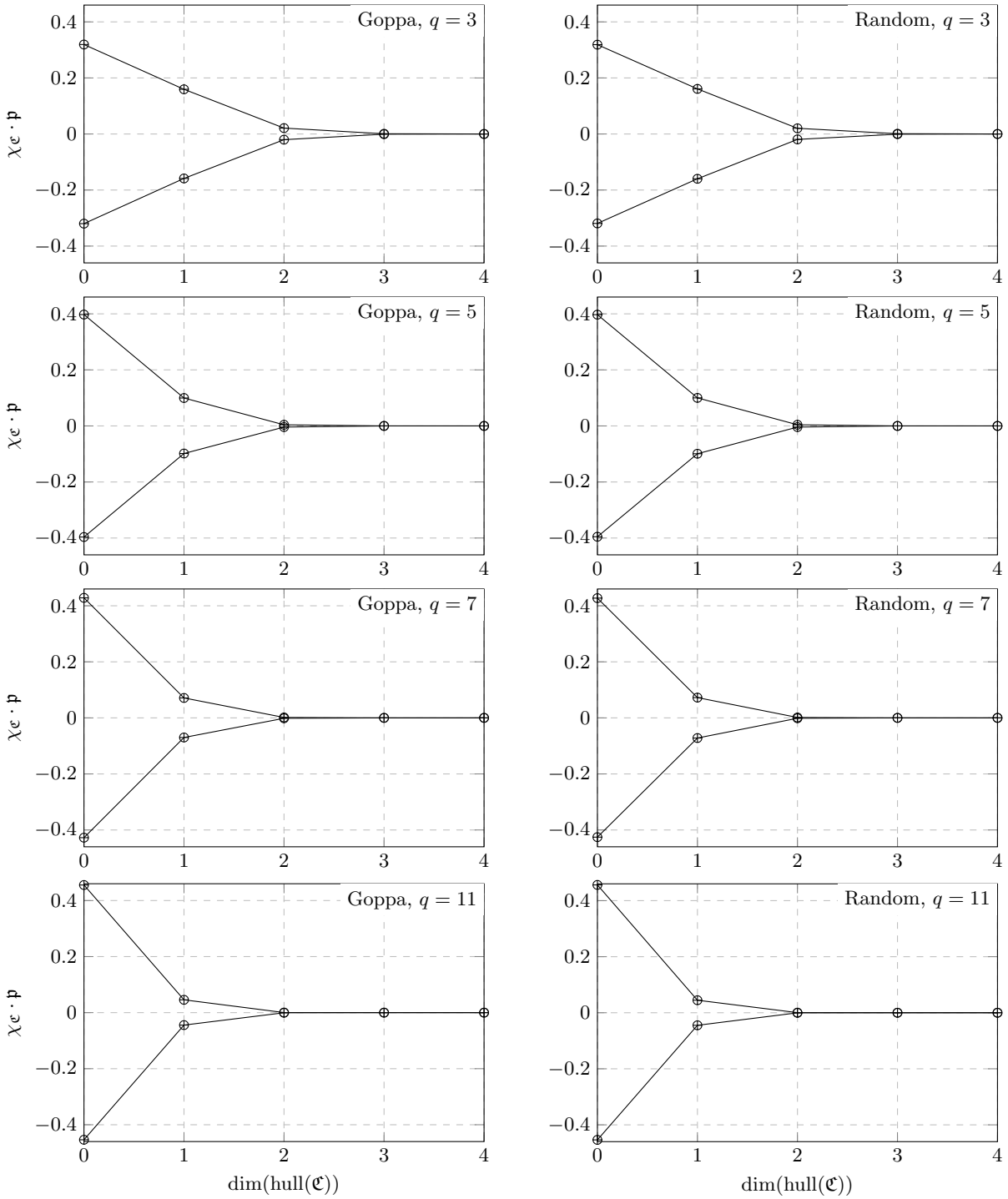


Fig. 3: Comparison of the genus distribution of random Goppa codes (on the left) over $\mathbb{F}_q$ for $q \in \{3, 5, 7, 11\}$ with parameters $m = 4$, $t = 9$, $n = 72$ and hence $k = 36$, and that of uniformly random $[n, k]$-linear codes over $\mathbb{F}_q$ (on the right). A genus is represented by the square class $\chi_{\mathfrak{C}} \in \{\pm 1\}$ and the hull dimension h, with $\mathfrak{p}$ being the measured probability of the genus $\mathcal{G}_{n,k,q}^{\chi_{\mathfrak{C}},h}$ over 10^5 samples.

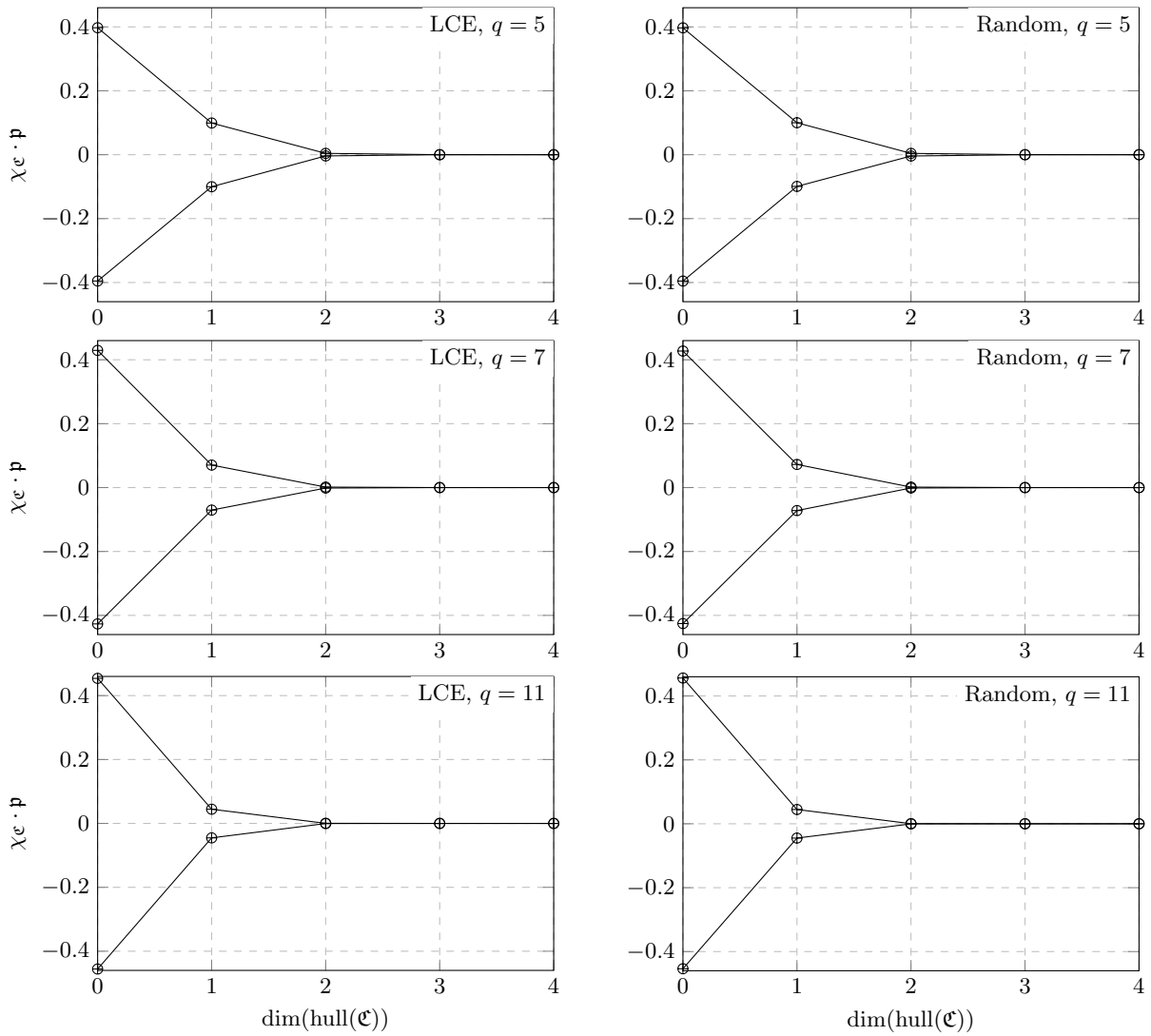


Fig. 4: Comparison of the genus distribution of random elements in the orbit of the linear equivalence action on a random code (on the left) over $\mathbb{F}_q$ for $q \in \{5, 7, 11\}$, and that of uniformly random $[n, k]$-linear codes over $\mathbb{F}_q$ (on the right), both for $n = 72$ and $k = 36$. A genus is represented by the square class $\chi_{\mathfrak{C}} \in \{\pm 1\}$ and the hull dimension h, with $\mathfrak{p}$ being the measured probability of the genus $\mathcal{G}_{n,k,q}^{\chi_{\mathfrak{C}},h}$ over 10^5 samples. Data for random codes identical to Figure 3.

C Code Listings

C.1 local.py

```
"""
This module contains experiments regarding the local structure of q-ary lattices and its
applications to code equivalence, including helper functions. It relies heavily on the module
selforthogonal.py that accompanied [EC:AlbBenLai25], which should be provided, and can also be found
here: https://artifacts.iacr.org/eurocrypt/2025/a9/

NOTE: There is a bug in selforthogonal.py with newer versions of sage that triggers an exception. We
do not fix it, we just wrap the problematic calls in try, since the exception is rare enough.
"""

from sage.all import (
    GF,
    PolynomialRing,
    QQ,
    bar_chart,
    random_matrix,
)
from sage.all import codes

import selforthogonal as so

import random
import sys


# distinguisher and sampling from genus

def SqClassD(A, B):
    """
    Implements the PCE distinguisher via the square class invariant.
    """
    TA, HA = so.dehull(A)
    TB, HB = so.dehull(B)
    assert HA.nrows() == HB.nrows()
    chiA = (TA * TA.T).det().is_square()
    chiB = (TB * TB.T).det().is_square()
    return chiA == chiB


def test_SqClassD(reps=10000, n=72, k=36, h=2, q=11, signed=False):
    """
    Test the attack on (S)PCE via the square class invariant, i.e. compare genera.
    """
    assert h < k
    r_count = 0
    wins = 0
    while r_count < reps:
        b = so.random_bit()
        try:
            A = so.SampleCode(n, k, h, q)
            if A is None:
                continue
            isom = so.random_signed_permutation_matrix(n) if signed else so.random_permutation_matrix(n)
            B = (A * isom).echelon_form() if b else so.SampleCode(n, k, h, q)
            if B is None:
                continue
        except:
            continue
        r_count += 1
        sys.stdout.write("\r{}/{}".format(r_count, reps))
        sys.stdout.flush()

        b_ = SqClassD(A, B)
        if b == b_:
            wins += 1
    print(f"Wins: {wins}/{reps} ~ {QQ(wins / reps).n(digits=4)}")


def compute_genus(A):
    """
    Simple functions that computes the genus of a linear code.
    """
```

```
    T, H = so.dehull(A)
    h = H.nrows()
    chi = 1 if not T else 2 * int((T * T.T).det().is_square()) - 1
    return (h, chi)


def SampleGenus(n, k, h, q, s, halt=256):
    """
    Returns a random [n,k]-linear code over FF_q with 2*k <= n and hull dimension h <= k and square
    class s.
    """
    assert s in [1, -1]
    for _ in range(halt):
        try:
            A = so.SampleCode(n, k, h, q)
        except:
            continue
        if A is None:
            continue
        gA = compute_genus(A)
        if gA != (h, s):
            continue
        return A
    return None


# Goppa and LCE experiments

def plot_goppa(reps=10000, p=3, m=4, n=3 ** 4 - 9, t=9, fname=None):
    """
    Sample random Goppa codes with parameters (t,m,n) and compute their genus. A log file and a
    chart are produced.
    """
    if n is None:
        n = p ** m - t
    k = n - m * t
    assert k >= 0
    assert p ** m - n >= 0
    F = GF(p ** m)
    R = PolynomialRing(F, "x")
    r_count = 0
    genera = {}
    while r_count < reps:
        g = R.random_element(degree=t, monic=True)
        if not g.is_squarefree():
            continue
        L_all = [a for a in F if g(a) != 0]
        L = random.sample(L_all, n)
        C = codes.GoppaCode(g, L)
        assert k == C.dimension()
        (h, chi) = compute_genus(C.generator_matrix())
        if (h, chi) in genera:
            genera[(h, chi)] += 1
        else:
            genera[(h, chi)] = 1
        r_count += 1
        sys.stdout.write(f"\r{r_count}/{reps}")
        sys.stdout.flush()
    if fname is None:
        print()
        print(k, genera, sep="\n")
    datp = [genera.get((h, 1), 0) / reps for h in range(k+1)]
    datm = [-genera.get((h, -1), 0) / reps for h in range(k+1)]
    G = bar_chart(datp, width=0.2) + bar_chart(datm, width=0.2)
    if fname is not None:
        fd = open(fname + ".log", "w")
        for g in genera:
            fd.write(f"{g}: {genera[g]}\n")
        fd.close()
        G.save(fname + ".png", xmax=5)
    else:
        G.show(xmax=5)
    return genera


def plot_lce(reps=10000, n=3 ** 4 - 9, k=3 ** 4 - 9 - 4 * 9, p=3, fname=None):
    """
```

```
    Sample a random code with parameters (n,k) and compute the genus of its diagonal orbit. A log file and a
        ↪ chart are produced.
    """
    r_count = 0
    A = random_matrix(GF(p), k, n)
    while A.rank() < k:
        A = random_matrix(GF(p), k, n)
    genera = {}
    while r_count < reps:
        D = so.random_diagonal_matrix(n, p)
        (h, chi) = compute_genus(A*D)
        if (h, chi) in genera:
            genera[(h, chi)] += 1
        else:
            genera[(h, chi)] = 1
        r_count += 1
        sys.stdout.write(f"\r{r_count}/{reps}")
        sys.stdout.flush()
    if fname is None:
        print()
        print(k, genera, sep="\n")
    datp = [genera.get((h, 1), 0) / reps for h in range(k+1)]
    datm = [-genera.get((h, -1), 0) / reps for h in range(k+1)]
    G = bar_chart(datp, width=0.2) + bar_chart(datm, width=0.2)
    if fname is not None:
        fd = open(fname + ".log", "w")
        for g in genera:
            fd.write(f"{g}: {genera[g]}\n")
        fd.close()
        G.save(fname + ".png", xmax=5)
    else:
        G.show(xmax=5)
    return genera


def plot_random(reps=10000, n=3 ** 4 - 9, k=3 ** 4 - 9 - 4 * 9, p=3, fname=None):
    """
    Sample random codes with parameters (n,k) and compute their genus. A log file and a chart are produced.
    """
    r_count = 0
    genera = {}
    while r_count < reps:
        A = random_matrix(GF(p), k, n)
        if A.rank() != k:
            continue
        (h, chi) = compute_genus(A)
        if (h, chi) in genera:
            genera[(h, chi)] += 1
        else:
            genera[(h, chi)] = 1
        r_count += 1
        sys.stdout.write(f"\r{r_count}/{reps}")
        sys.stdout.flush()
    if fname is None:
        print()
        print(k, genera, sep="\n")
    datp = [genera.get((h, 1), 0) / reps for h in range(k+1)]
    datm = [-genera.get((h, -1), 0) / reps for h in range(k+1)]
    G = bar_chart(datp, width=0.2) + bar_chart(datm, width=0.2)
    if fname is not None:
        fd = open(fname + ".log", "w")
        for g in genera:
            fd.write(f"{g}: {genera[g]}\n")
        fd.close()
        G.save(fname + ".png", xmax=5)
    else:
        G.show(xmax=5)
    return genera


def produce_data():
    """
    Produces data for the Goppa experiments appendix. Takes a long time.
    """
    plot_goppa(reps=100000, p=3, fname="goppa-3")
    plot_random(reps=100000, p=3, fname="random-3")
    plot_goppa(reps=100000, p=5, fname="goppa-5")
```

```
    plot_lce(reps=100000, p=5, fname="lce-5")
    plot_random(reps=100000, p=5, fname="random-5")
    plot_goppa(reps=100000, p=7, fname="goppa-7")
    plot_lce(reps=100000, p=7, fname="lce-7")
    plot_random(reps=100000, p=7, fname="random-7")
    plot_goppa(reps=100000, p=11, fname="goppa-11")
    plot_lce(reps=100000, p=11, fname="lce-11")
    plot_random(reps=100000, p=11, fname="random-11")
```

C.2 Modified `selforthogonal.py`

```
"""
This module implements the algorithms for sampling uniformly random linear codes with a specified hull
    ↪ dimension,
presented in [EC:AlbBenLai25]. The main function to do this is SampleCode(n,k,h,q).

The module also implements some standard operations such as computing the dual, the closure, and the
hull, along with some non-standard operations such as the signed closure, and "dehulling" a code, that
is computing a basis of the subcode T <= C such that C = T + hull(C).
"""
from sage.all import (
    GF,
    Permutation,
    PolynomialRing,
    RR,
    diagonal_matrix,
    identity_matrix,
    is_prime_power,
    matrix,
    random_matrix,
    random_vector,
    span,
    vector,
)
from sage.all import randint, random, shuffle
from sage.all import copy


# utility functions

def _cprod(a, b):
    """
    Entry-wise multiplication of vectors.
    """
    assert len(a) == len(b)
    return vector([a[i] * b[i] for i in range(len(a))])


# basic samplers

def random_bit():
    """
    Random bit {0,1} as int.
    """
    return randint(0, 1)


def random_invertible(q):
    """
    Return a random element of FF_q^*.
    """
    F = GF(q)
    z = F.multiplicative_generator()
    i = randint(1, q-1)
    return z ** i


def random_fullrank_matrix(n, q):
    """
    Return a random full-rank n x n matrix over FF_q.
    """
    while True:
        A = random_matrix(GF(q), n, n)
        if A.rank() == n:
            return A
```

```
def random_permutation_matrix(n):
    """
    Return a random element from the group P_n(ZZ).
    """
    ind = list(range(1, n+1))
    shuffle(ind)
    return Permutation(ind).to_matrix()


def random_signed_permutation_matrix(n):
    """
    Return a random element from the group O_n(ZZ).
    """
    P = random_permutation_matrix(n)
    v = vector([(-1) ** random_bit() for i in range(n)])
    return matrix([_cprod(p, v) for p in P])


def random_diagonal_matrix(n, q):
    """
    Return a random element of diag(FF_q^*).
    """
    v = [random_invertible(q) for i in range(n)]
    return diagonal_matrix(v)


def random_monomial_matrix(n, q):
    """
    Return a random element from the group M_n(FF_q).
    """
    P = random_permutation_matrix(n)
    v = vector([random_invertible(q) for i in range(n)])
    return matrix([_cprod(p, v) for p in P])


# sanity check

def _check_code(A, n=None, k=None, q=None):
    """
    Return parameters n,k,F of [n,k]-linear code over F, and check health. Optional parameters
    override reality.
    """
    if n is None:
        n = A.ncols()
    if k is None:
        k = A.nrows()
    F = None
    if q is None:
        F = A.base_ring()
    else:
        assert is_prime_power(q)
        F = GF(q)
    assert n >= k
    assert A.rank() == k
    assert F.is_field()
    assert F.is_finite()
    return n, k, F


# algorithms on linear codes

def dual(A, q=None):
    """
    Given a row generator matrix for a [n,k]-linear code over a finite field F, return a
    generator matrix of its dual.
    """
    _ = _check_code(A, q=q)
    return A.right_kernel().basis_matrix()


def hull(A):
    """
    Given a row generator matrix for a [n,k]-linear code over a finite field F, return a
    generator matrix of its hull.
    """
    _ = _check_code(A)
```

```
    M = A.stack(dual(A))
    H = M.right_kernel().basis_matrix()
    return H


def hulldim(A):
    """
    Compute hull dimension of the linear code generated by A.
    """
    _, k, _ = _check_code(A)
    return k - (A * A.T).rank()


def dehull(A):
    """
    Given an [n,k]-linear h-hollow code A with hull H, compute a [n,k]-linear 0-hollow code T such
    that A = H + T algebraically. Returns (T, H).

    EXAMPLES:
        sage: import selforthogonal as so
        sage: A = so.SampleCode(16, 8, 4, 41)
        sage: T, H = so.dehull(A)
        sage: assert span(A) == span(T) + span(H)
        sage: assert span(H) == span(so.hull(A))
    """
    _, k, F = _check_code(A)

    H = hull(A)
    h = H.nrows()
    # edgecases
    if h == 0:
        return (A, H)
    if h == k:
        return (matrix(F, []), A)

    # compute basis of A/hull(A) and lift to A
    V = span(A)
    W = span(H)
    Q = V / W
    T = matrix(F, [Q.lift(b) for b in Q.basis()])

    assert hulldim(T) == 0
    return (T, H)


# various closures

def closure(A):
    """
    Given a row generator matrix for a [n,k]-linear code over a finite field F, return a
    generator matrix for its closure.
    """
    _, _, F = _check_code(A)
    A_ = copy(A)
    for x in F:
        if x == 0 or x == 1:
            continue
        A_ = A_.augment(x * A)
    return A_


def signed_closure(A):
    """
    Given a row generator matrix for a [n,k]-linear code over a finite field F, return a
    generator matrix for its signed closure.
    """
    _ = _check_code(A)
    A_ = copy(A)
    A_ = A_.augment(-1 * A)
    return A_


def _QR(F):
    """
    Output the list of all distinct quadratic residues of the finite field F.
    """
    return [x for x in F if x.is_square()]
```

```
# samplers for hollow codes

def MaxSols(D, F):
    """
    Return the maximal number of solutions a smooth conic with discriminant D over the finite field
    F can have.

    [EC:AlbBenLai25]: Fig. 3
    """
    assert D != 0
    assert D in F
    q = len(F)
    return 2 * q - 1 if D.is_square() else q + 1


def SampleSelfOrthogonal(A, halt=128, n=None, q=None):
    """
    Sample a uniformly random self-orthogonal vector from a code generated by A.

    EXAMPLES:
        sage: import selforthogonal as so
        sage: A = random_matrix(GF(41), 8, 16)
        sage: v = so.SampleSelfOrthogonal(A)  # sample s.o. vector v from Span(A) <= FF_{41}^{16}
        sage: assert v * v == 0
        sage: assert v in span(A)
        sage:
        sage: v = so.SampleSelfOrthogonal(identity_matrix(GF(41), 16))  # sample s.o. vector v from FF_
            ↪ {41}^{16}
        sage: v = so.SampleSelfOrthogonal([], n=16, q=41)  # this does the same thing
        sage: assert v * v == 0

    [EC:AlbBenLai25]: Def. 10, Lem. 5, Fig. 3
    """
    # edge case to just return a random so vector if A is empty
    if not A:
        if n is None or q is None:
            return None
        assert n > 0
        assert is_prime_power(q)
        A = identity_matrix(GF(q), n)
    _, k, F = _check_code(A)

    # reorder basis to get smooth conic
    G = copy(A)
    v, w = G[k-2], G[k-1]
    D = (v*w)**2 - (v*v) * (w*w)
    if D == 0:
        found = False
        for i in range(k):
            for j in range(i+1, k):
                v, w = G[i], G[j]
                D = (v*w)**2 - (v*v) * (w*w)
                if D != 0:
                    found = True
                    break
            if found:
                break
        G[i], G[k-2] = G[k-2], G[i]
        G[j], G[k-1] = G[k-1], G[j]

    # set max number of solutions
    M = MaxSols(D, F)

    R, X = PolynomialRing(F, "X").objgen()
    for i in range(halt):
        pre_x = [F.random_element() for i in range(k-2)]
        # find all conic solutions
        roots = []
        for a in F:
            x = vector(R, pre_x + [a, X])
            rs = (x * G * G.T * x).roots(multiplicities=False)
            for r in rs:
                roots.append((a, r))
        L = len(roots)
        # assert L in {M, M-len(F)}
```

```
        # possibly reject
        if random() > L / M:
            continue
        # sample random solution
        sol = list(roots[randint(0, L-1)])
        # construct so vector
        cs = vector(F, pre_x + sol)
        v = cs * G
        assert v * v == 0
        return v
    return None


def SampleCode(n, k, h, q, randomize=True):
    """
    Returns a random [n,k]-linear code over FF_q with 2*k <= n and hull dimension h <= k.

    EXAMPLES:
        sage: import selforthogonal as so
        sage: A = so.SampleCode(16, 8, 4, 41)  # sample 8-dim. subspace of FF_{41}^{16} with hull dim. 4
        sage: assert so.hulldim(A) == 4
        sage:
        sage: A = so.SampleCode(16, 8, 4, 41, randomize=False)  # get hull basis as last 4 rows
        sage: H = so.hull(A)
        sage: assert span(A[4:]) == span(H)

    [EC:AlbBenLai25]: Def. 11, Lem. 6, Fig. 3
    """
    assert k >= 1
    assert h >= 0
    # proof of Lem. 6 requires 2*h <= k, but this also works
    assert h <= k
    assert 2 * k <= n
    F = GF(q)

    # sample trivial hull part
    vs = []
    for i in range(k - h):
        v = random_vector(F, n)
        vs.append(v)

    # check rank and hull dimension
    A_0 = matrix(F, vs)
    if A_0.rank() != k - h:
        return None
    if hulldim(A_0) != 0:
        return None

    # sample hull part
    A_i = A_0
    for i in range(h):
        # sample new hull vector
        v = SampleSelfOrthogonal(dual(A_i, q=q), halt=n, n=n, q=q)
        if v is None:
            return None
        vs.append(v)
        # check rank increments
        A_i = matrix(F, vs)
        if A_i.rank() != k - h + i + 1:
            return None
    A_h = A_i
    assert A_h.rank() == k
    assert hulldim(A_h) == h

    # optionally randomise basis
    S = random_fullrank_matrix(k, len(F)) if randomize else 1
    return S * A_h


def random_selforthogonal_code(n, k, q):
    """
    Return a random self-orthogonal i.e. weakly self-dual [n,k]-linear code over FF_q.
    """
    return SampleCode(n, k, k, q)


# misc
```

```
def prob_hull(n, k, h, q):
    """
    Returns the probability that a uniformly random [n,k]-linear code over FF_q has hull dimension h.

    Sendrier (1997): On the Dimension of the Hull, Thm. 4.18
    """
    p = RR(1)
    p /= q ** (h * (h + 1) / 2)
    for i in range(k - h):
        p *= (RR(1) - (RR(1) / (q ** (k - i))))
    for i in range(1, ((k - h) // 2) + 1):
        p /= (RR(1) - (RR(1) / (q ** (2*i))))
    return p


if __name__ == "__main__":
    print("This is a library for sampling linear codes with a specified hull dimension, and not a callable
        ↪ program.")
    print("")
    print("To use this library, import it as a module into your project with:")
    print("import selforthogonal")
    print("")
    print("To read the help manual, import this library into the Sage REPL and run:")
    print("sage: import selforthogonal")
    print("sage: help(selforthogonal)")
```