La Matematica (2025) 4:459–485
https://doi.org/10.1007/s44007-025-00152-2

REVIEW ARTICLE

The Code Equivalence Problem and Its Applications to Cryptography

Jean-François Biasse[1,2] · **Giacomo Micheli**[1,2]

Received: 6 October 2023 / Revised: 22 November 2024 / Accepted: 31 March 2025 /
Published online: 18 April 2025

Abstract
Code-based cryptographic schemes are a very popular quantum-safe alternative to current standards. Yet, designing efficient code-based signatures is a challenging task, and current proposals are still far from the target set by other post-quantum primitives (e.g. lattice-based). In this survey, we describe a code-based approach for signing, based on the hardness of the code equivalence problem. We recall the best known techniques for solving the code equivalence problem known to date, and we discuss its theoretical complexity.

Keywords Code-based cryptography · Code equivalence · Digital signature · Post-quantum cryptography

1 Introduction

In this survey, we are interested in maps between linear codes. We follow the notations of [1, Sect. 1.2] regarding the basic concepts pertaining to linear codes. An (n, M) code $\mathcal{C}$ over the finite field $\mathbb{F}_q$ is a subset of $\mathbb{F}_q^n$ of size M. If in addition, $\mathcal{C}$ is a k-dimensional subspace of of $\mathbb{F}_q^n$, then we call $\mathcal{C}$ an $[n, k]$-linear code over $\mathbb{F}_q$. In this case, $\mathcal{C}$ has $M = q^k$ elements (which we call codewords in the following). We say that $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$ is a generator matrix for $\mathcal{C}$ if the rows of $\boldsymbol{G}$ span $\mathcal{C}$, and we say that $\boldsymbol{H} \in \mathbb{F}_q^{n\times k}$ is a parity check matrix check matrix for $\mathcal{C}$ if $\boldsymbol{x} \in \mathcal{C} \Leftrightarrow \boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{0}$.

Jean-François Biasse and Giacomo Micheli: These authors contributed equally to this work

✉ Jean-François Biasse
biasse@usf.edu

✉ Giacomo Micheli
gmicheli@usf.edu

[1] Department of Mathematics & Statistics, University of South Florida, 4202 E Fowler Ave, Tampa, FL 33620, USA

[2] Center for Cryptographic Research, University of South Florida, 4202 E Fowler Ave, Tampa, FL 33620, USA

To introduce the concept of *equivalence* between two codes, we start with a special case, namely *Permutation Equivalence* (PE). We say that two linear codes $\mathcal{C}_1, \mathcal{C}_2$ are permutation equivalent if there is a permutation of coordinates $\pi \in \mathcal{S}_n$ that sends $\mathcal{C}_1$ to $\mathcal{C}_2$ (see [1, Sect. 1.6]). We denote this by $\mathcal{C}_2 = \pi(\mathcal{C}_1)$. From a matrix standpoint, π acts as a right multiplication by a matrix $\boldsymbol{P}$ that satisfies $P_{j,i} = 1$ if $\pi(i) = j$ and $P_{i,j} = 0$ otherwise. If $\boldsymbol{G}$ is a generator matrix for $\mathcal{C}_1$, and if $\boldsymbol{P}$ is a permutation matrix corresponding to π, then $\boldsymbol{GP}$ is a generator matrix for $\pi(\mathcal{C}_1)$.

Example 1 Assume that $n = 5, q = 7, \pi = (1, 2, 3)(4, 5)$. Let $\mathcal{C}$ be the code generated by

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 2 & 5 & 3 \\ 0 & 1 & 6 & 2 & 1 \end{pmatrix}.$$

Then the permutation π acts in the following way:

- $\pi(1, 0, 2, 5, 3) = (2, 1, 0, 3, 5)$.
- $\pi(0, 1, 6, 2, 1) = (6, 0, 1, 1, 2)$.

Moreover, the permutation matrix corresponding to π is given by

$$\boldsymbol{P} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

We see that a generator matrix for $\pi(\mathcal{C})$ is given by $\boldsymbol{GP} = \boldsymbol{G} = \begin{pmatrix} 2 & 1 & 0 & 3 & 5 \\ 6 & 0 & 1 & 1 & 2 \end{pmatrix}$.

The most general notion of code equivalence generalizes PE by applying a field automorphism to the coordinates of $\mathcal{C}_1$, then applying a permutation, and finally multiplying each coordinate of the resulting vector by elements of $\mathbb{F}_q$. The last two steps correspond to applying a *linear isometry* of $\mathbb{F}_q^n$, i.e. a linear map that preserves the Hamming distance.

Proposition 1 (Linear isometries of $\mathbb{F}_q^n$) *All linear isometries of $\mathbb{F}_q^n$ are of the form $\tau = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$. The image $\boldsymbol{y}$ of $\boldsymbol{x} \in \mathbb{F}_q^n$ is given by*

$$\boldsymbol{y}_{\pi(i)} = \boldsymbol{v}_{\pi(i)} \boldsymbol{x}_i, \quad \forall i \leq n.$$

A linear isometry $\tau = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ acts on row vectors as a right multiplication by a matrix of the form $\boldsymbol{Q} = \boldsymbol{DP}$ where $\boldsymbol{D}$ is the diagonal matrix whose diagonal entries are the coefficients of $\boldsymbol{v}$, and $\boldsymbol{P}$ is the permutation matrix corresponding to π. Such a matrix is called a *monomial matrix*. We denote by M_n the group of $n \times n$ monomial matrices. Below, we provide a formal definition of equivalence of codes.

Definition 1 (*Code Equivalence*) We say that two linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are *equivalent*, and write $\mathcal{C}_1 \sim \mathcal{C}_2$, if there exist a field automorphism $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and a linear

isometry $\tau = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that

$$\mathcal{C}_2 = \tau(\alpha(\mathcal{C}_1)) = \left\{ \boldsymbol{y} \in \mathbb{F}_q^n \ : \boldsymbol{y} = \tau(\alpha(\boldsymbol{x})),\ \boldsymbol{x} \in \mathcal{C}_1 \right\},$$

where $\alpha(\boldsymbol{x})$ denotes the application of α on each coordinate of $\boldsymbol{x}$.

Clearly, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are two codes with generator matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, respectively, it holds that

$$\mathcal{C}_1 \sim \mathcal{C}_2 \iff \exists((\boldsymbol{S}, \boldsymbol{Q}); \alpha) \in (\mathsf{GL}_k(\mathbb{F}_q) \times \mathsf{M}_n) \rtimes \mathsf{Aut}(\mathbb{F}_q) \quad \text{s.t. } \boldsymbol{G}_2 = \boldsymbol{S}\alpha(\boldsymbol{G}_1)\boldsymbol{Q},$$

where $\alpha(\boldsymbol{G})$ denotes the application of α on all the coefficients of $\boldsymbol{G}$.

Example 2 Assume that $n = 5$, $q = 7$, $\alpha = \mathsf{Id}$, $\pi = (1, 2, 3)(4, 5)$ and $\tau = ((2, 4, 3, 5, 2), \pi)$. Let $\mathcal{C}$ be the code generated by

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 2 & 5 & 3 \\ 0 & 1 & 6 & 2 & 1 \end{pmatrix}.$$

Then the map defined by τ is given by

- $\pi(1, 0, 2, 5, 3) = (2, 1, 0, 3, 5)$, $\tau(1, 0, 2, 5, 3) = (4, 4, 0, 1, 3)$.
- $\pi(0, 1, 6, 2, 1) = (6, 0, 1, 1, 2)$, $\tau(0, 1, 6, 2, 1) = (5, 0, 3, 5, 4)$.

Hence, the image of $\mathcal{C}$ is generated by $\begin{pmatrix} 4 & 4 & 0 & 1 & 3 \\ 5 & 0 & 3 & 5 & 4 \end{pmatrix}$.

The notion we just presented is usually known as *semilinear equivalence*. If the field automorphism is the trivial one (i.e. $\alpha = id$, as in Example 2), then the notion is simply known as *linear equivalence* (or alternatively *monomial equivalence*). If, furthermore, the monomial matrix is a permutation (i.e. $\boldsymbol{Q} = \boldsymbol{D}\boldsymbol{P}$ with $\boldsymbol{D} = \boldsymbol{I}_n$), then we obtain permutation equivalence. Note that, in practical implementation of cryptographic applications of code equivalence (e.g. [2, 3]), the fields considered are always prime for efficiency reasons, and therefore the last two notions are the only ones of interest to us. Now, let us state the Code Equivalence problem (CE).

Problem 1 (Code Equivalence) *Let $\mathcal{C}_1, \mathcal{C}_2$ be two $[n, k]$-linear code over $\mathbb{F}_q$. The Code Equivalence problem is the decision of whether $\mathcal{C}_1 \sim \mathcal{C}_2$. We state the two variants that correspond to the special notions of equivalence mentioned above*:

- *The Linear Code Equivalence (LE) problem is the decision of whether $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly equivalent.*
- *The Permutation Code Equivalence (PE) problem is the decision of whether $\mathcal{C}_1$ and $\mathcal{C}_2$ are permutationally equivalent.*

Problem 1 is formulated as a decision problem. This is the original formulation of CE and its variants LE and PE. However, for the purpose of cryptanalysis, the relevant computational problem is that of computing the map under the assumption that the codes are equivalent, i.e. the search variant of Problem 1.

Problem 2 (Search-CE) *Let $\mathcal{C}_1, \mathcal{C}_2$ be two equivalent $[n, k]$-linear code over $\mathbb{F}_q$. The search-CE problem consists in finding $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and $\tau \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $\mathcal{C}_2 = \tau(\alpha(\mathcal{C}_1))$. We present the two variants that correspond to the special notions of equivalence*:

- *Search-LE*: *find $\tau \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $\mathcal{C}_2 = \tau(\mathcal{C}_1)$.*
- *Search-PE*: *find $\pi \in \mathcal{S}_n$ such that $\mathcal{C}_2 = \pi(\mathcal{C}_1)$.*

From a linear algebra perspective, given $\boldsymbol{G}_1, \boldsymbol{G}_2 \in \mathbb{F}_q^{k \times n}$ two generator matrices for two equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, search-CE consists in finding a field automorphism $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and two matrices $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{Q} \in \mathsf{M}_n$ such that $\boldsymbol{G}_2 = \boldsymbol{S}\alpha(\boldsymbol{G}_1)\boldsymbol{Q}$.

2 A Search to Decision Reduction

In this section, we present an efficient reduction from search-PE to PE first introduced by Biasse and Micheli [4]. In a nutshell, it consists in iteratively searching for $\boldsymbol{P}(t)$ for $t = 1, 2, \ldots, n$ where $\boldsymbol{P}$ is a permutation matrix such that $\boldsymbol{SGP} = \boldsymbol{F}$ for the generator matrices $\boldsymbol{G}, \boldsymbol{F}$ spanning the two input codes. The main idea is to test $\boldsymbol{P}(t) = i_t$ by extending the generator matrix $\boldsymbol{G}$ with repetitions of its tth column, and by extending the generator matrix $\boldsymbol{F}$ with repetitions of its i_tth column. The PE oracle returns a positive answer if $\boldsymbol{P}(t) = i_t$, thus allowing us to compute the image of $\boldsymbol{P}$ at t. Further extensions of $\boldsymbol{G}$ and calls to the PE oracle allow us to find all images of the secret permutation. The main result is stated as follows:

Theorem 2 (Corollary III.6 of [4]) *There is an algorithm that solves PE between two $[k, n]$-linear codes over $\mathbb{F}_q$ by making at most n^2 calls to an oracle that solves PE on input codes of dimension k and length at most $n^2(n+1)/2$ over $\mathbb{F}_q$.*

The main technical difficulty to prove the above theorem is that once a possible $\boldsymbol{P}(t)$ has been identified, the search for $\boldsymbol{P}(t+1)$ needs to be performed accordingly. Indeed, due to the existence of automorphisms of the input codes, there may be multiple solutions to the search PE problem. Then, if one was to apply the strategy described above independently for t and $t+1$, one might obtain $\boldsymbol{P}(t)$ and $\boldsymbol{P}'(t+1)$ for two permutation matrices $\boldsymbol{P}, \boldsymbol{P}'$ that are both solution. Problems can arise if there is no solution whose values at t and $t+1$ simultaneously match $\boldsymbol{P}(t)$ and $\boldsymbol{P}'(t+1)$. To ensure that the "right" choice of $\boldsymbol{P}(t)$ is made, we use [4, Lemma III.1] that guarantees the existence of a solution whose value at t matches $\boldsymbol{P}(t)$, as well as the images of $1, 2, \ldots, t-1$ previously found.

In [4], the focus was on the resolution of Search-PE, leaving Search-LE and Search-CE instances for future work. In the case of linear code equivalence, an extra difficulty comes from the fact that the secret linear isometry τ is in $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$. The general approach fleshed out in [4] applies to the linear equivalence problem. To achieve that, care needs to be given to the identification of potential images of a given column since linear isometries also induce a scalar multiplication. From there, it is likely that this result will extend to the most general form of the code equivalence problem since the only difference with linear code equivalence is the action of a field isomorphism.

Finally, other potential consequences of the work presented in this section could include a search-to-decision reduction for the code equivalence problem in the rank metric. This problem has received some attention recently, especially since computational problems in coding theory in the rank metric have been used as the basis for potential quantum-safe cryptosystems. More generally, focusing on decisional variants of CE (while most algorithms for CE actually solve search-CE) could inspire new ways to solve search-CE via the reduction presented in this paper. This might be especially true when considering unconventional computing frameworks, in particular quantum computing.

3 Code Equivalence as a Group Action

CE and its variants can be be rephrased in terms of group actions. This framework as recently received considerable traction due to its connection with isogeny-based cryptography. In 1997, Couveignes introduced the concept of *hard homogeneous spaces* through a seminar talk at the Ecole Normale Supérieure, and a submission to the CRYPTO 97 conference. The submitted manuscript was published *as is* as a preprint [5] in 2006 when similar independent work appeared in the literature [6, 7]. Couveignes' work describes how the ideal class group of a quadratic order acted on isomorphism classes of ordinary elliptic curves over finite fields. In this original framework, a finite abelian group G acts on a space H simply and transitively. To enable the design of secure cryptosystems, the following problems need to be computationally hard:

- **Vectorization**: Given $h_1, h_2 \in H$, find $g \in G$ such that $g \star h_1 = h_2$.
- **Parallelization**: Given $h_1, h_2, h_3 \in H$, compute the unique $h_4 \in H$ such that $\exists g \in G$ with $h_2 = g \star h_1$ and $h_4 = g \star h_3$.

This framework readily extends to semigroup actions [8–10], and to non-abelian groups. Couveignes' framework was later refined by Alamati, Defeo, Montgomery, and Patranabis to produce the notion of *cryptographic group actions* [11]. This framework coincides with that of Couveignes in the case of abelian groups acting simply and transitively. In [11], an Effective Group Action (EGA) is defined as the action of a finite group G on a finite set X such that one can efficiently test membership and equality in G, sample elements from a distribution on G, perform operations and inversions on elements of G, as well as test membership in X, uniquely represent elements of X, compute the action of $g \in G$ on $x \in X$, and identify a distinguished elements $x_0 \in X$ called the origin. The hardness properties of EGAs as described in [11] are

- **One-way EGA**: given $(x, g \star x)$ where $x \in X$ and $G \in G$ are sampled uniformly at random, it must be difficult to recover g.
- **Weak Unpredictable EGA**: given polynomially many pairs $(x_i, g \star x_i)$, where $g \in G$ and the $x_i \in X$ are sampled uniformly at random, it must be difficult to compute $g \star x$ for a random $x \in X$.

- **Weak Pseudorandom EGA**: it must be difficult to distinguish pairs $(x_i, g\star x_i)$, where $g \in G$ and the $x_i \in X$ are sampled uniformly at random from pairs (x_i, y_i) where x_i, y_i are sampled uniformly at random in X.

To recast CE as a group action problem, we consider the set $X \subseteq \mathbb{F}_q^{k\times n}$ comprised of all full-rank $k \times n$ generator matrices in systematic form, i.e. each element of X uniquely identifies a $[n, k]$-linear code, and the group $G_{\mathsf{CE}} = \mathbb{F}_q^{*n} \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, which is the group of semilinear isometries [12, Lemma 9]. In X, we identify a code $\mathcal{C}$ with its generator matrix $\boldsymbol{G}$ in systematic form. Then, G_{CE} naturally acts on X via

$$\begin{array}{rccc} \star: & X \times G_{\mathsf{CE}} & \to & X \\ & (\boldsymbol{G}, ((v_1, \ldots, v_n); (\alpha, \pi))) & \to & \mathsf{SF}\left(\alpha(\boldsymbol{G})\,\mathrm{diag}(v_1, \ldots, v_n)\boldsymbol{P}_\pi\right). \end{array}$$

The codes $\mathcal{C}_1, \mathcal{C}_2$ generated by $\boldsymbol{G}_1, \boldsymbol{G}_2$ are equivalent if and only if there is $g \in G_{\mathsf{CE}}$ such that $g\star\boldsymbol{G}_1 = \boldsymbol{G}_2$.

Definition 2 We define the following subgroups of $G_{\mathsf{CE}} = \mathbb{F}_q^{*n} \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$:

- $G_{\mathsf{LE}} \subseteq G_{\mathsf{CE}} = \{(((v_1, \ldots, v_n); (\alpha, \pi))) \in G_{\mathsf{CE}} \mid \alpha = \mathsf{id}\}$,
- $G_{\mathsf{PE}} \subseteq G_{\mathsf{LE}} = \{(((v_1, \ldots, v_n); (\alpha, \pi))) \in G_{\mathsf{CE}} \mid \alpha = \mathsf{id}, (v_1, \ldots, v_n) = (1, \ldots, 1)\}$.

Proposition 3 (Code Equivalence as a group action) *Let $\mathcal{C}_1, \mathcal{C}_2$ be two $[n, k]$-linear codes over $\mathbb{F}_q$ of generator matrices in systematic form $\boldsymbol{G}_1, \boldsymbol{G}_2$. Then we have*

- *$\mathcal{C}_1$ is equivalent to $\mathcal{C}_2$ if and only if $\exists g \in G_{\mathsf{CE}}$, $\boldsymbol{G}_2 = g\star\boldsymbol{G}_1$.*
- *$\mathcal{C}_1$ is linearly equivalent to $\mathcal{C}_2$ if and only if $\exists g \in G_{\mathsf{LE}}$, $\boldsymbol{G}_2 = g\star\boldsymbol{G}_1$.*
- *$\mathcal{C}_1$ is permutationally equivalent to $\mathcal{C}_2$ if and only if $\exists g \in G_{\mathsf{PE}}$, $\boldsymbol{G}_2 = g\star\boldsymbol{G}_1$.*

Solving the code equivalence problem (and its variants) is thus equivalent to deciding whether the generator matrices of the input codes are related by the action of an element of G_{CE} (the search variants of CE consisting of finding the element that realizes this action). It is important to note that the $\star$ group action does not satisfy some useful additional properties for the design of cryptographic schemes (as formalized in [11, Sect. 3]). Indeed, only group actions that satisfy at least one of the following properties are considered: transitive, faithful, free, or regular. The CE group action is is not *transitive*, because is not always possible to connect two given element of X via a group element. It is also not free because a given code might have *automorphisms*, i.e. elements $g \in G$ such that $g\star x = x$. We denote the subgroup of G_{CE} (resp. G_{LE}, G_{PE}) of automorphisms of $x \in X$ by $\mathsf{Stab}_{G_{\mathsf{CE}}}(x)$ (resp. $\mathsf{Stab}_{G_{\mathsf{LE}}}(x)$, $\mathsf{Stab}_{G_{\mathsf{PE}}}(x)$), or simply $\mathsf{Stab}(x)$ when there is no ambiguity. The action of G_{PE} is faithful: for every non-trivial permutation $\pi \in \mathcal{S}_n$, one can find a code $\mathcal{C}$ such that $\pi(\mathcal{C}) \neq \mathcal{C}$. On the other hand, elements of G_{LE} of the form $((\lambda, \ldots, \lambda), \mathsf{Id})$ fix all codes, hence the action of G_{CE} and G_{LE} is not faithful. The quotient groups by the subgroup of elements of the form $\lambda\mathsf{Id}$ for $\lambda \in \mathbb{F}_q^*$ act faithfully on linear codes.

One of the most important restriction of the code equivalence group action is that it is not commutative. This is a considerable obstacle in the design of cryptographic protocols such as key exchange or encryption which usually rely on commutativity.

Nevertheless, a non-commutative action allows the adaption of signature schemes and some of their important optimizations and variants.

Another important negative result is the existence of a representation of the code equivalence group action [13]. This means that there is a map $\iota : X \to \mathbb{F}_q^{kn}$ together with a representation $\rho : G_{\mathsf{LE}} \to \mathsf{GL}(\mathbb{F}_q^{kn})$ that are compatible with the group action $\star$, i.e. $\rho(g)\iota(x) = \iota(g\star x)$. This has the important consequence:

Proposition 4 (Corollary 16 of [13]) *The action of G_{LE} on the space of $[k, n]$-linear codes over $\mathbb{F}_q$ is not weakly unpredictable, nor weakly pseudorandom.*

Proof Given samples of the form $(x_1, g\star x_1), \ldots, (x_m, g\star x_m)$, for a sufficiently large (yet polynomial in n) m, one can recover g. First, one recovers $\rho(g)$ from $\rho(g)\iota(x_1), \ldots, \rho(g)\iota(x_m)$ and $\iota(x_1), \ldots, \iota(x_m)$. Then, one reconstructs g from $\rho(g)$ (see [13, Proof of Corollary 16]). □

The above also implies that the action of G_{CE} is not weakly unpredictable, nor weakly pseudorandom by exhaustive search on the automorphisms of $\mathbb{F}_q$ (as in the proof of Proposition 7). This important limitation seems to prevent straightforward designs of Pseudorandom Functions (PRF) and other schemes that involve the broadcast of many samples of the form $(x_i, g\star x_i)$ for a secret g.

4 Self-Reduction of Code Equivalence

Next, we discuss random self-reducibility of CE and its variants. In a nutshell, random self-reducibility means that a good algorithm to solve average instances of a given problem implies a good algorithm to solve the hardest instances of that problem. Such a property, initially described in [14, Sect. 2.2], is desirable in the design of cryptosystems.

Definition 3 (*Random self reduction* (*as phrased in* [15]) Let $f : X \to Y$ and P be the problem of evaluating f on an arbitrary $x \in X$. A random self-reduction maps an arbitrary worst-case instance $x \in X$ to a set of random instances $y_1, \ldots, y_k$ in such a way that $f(x)$ can be computed in polynomial time given x, $f(y_1), \ldots, f(y_k)$ and the coin toss sequence used in the mapping.

CE (resp. LE and PE) are not self-reducible problems. This means that the hardness of random instances might be different. However, we can define a weaker notion of self-reducibility restricted to the orbit of a code. This is similar to the case of cryptosystems based on the lattice isomorphism problem [16, Sect. 3] where the worst case to average case reduction is realized *within* an equivalence class. Self-reduction of the code equivalence problem (within a given orbit) is made considerably easier by the fact that the problem is the inversion of the action of a finite group.

Proposition 5 (Random self-reducibility of CE) *Let G be either G_{PE}, G_{LE} or G_{CE}. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two $[k, n]$-linear codes over $\mathbb{F}_q$. Given access to an oracle that decides if there is $g \in G$ such that $\mathcal{C}_2' = g\star\mathcal{C}_1'$ for $(\mathcal{C}_1', \mathcal{C}_2') \in \mathsf{Orb}_G(\mathcal{C}_1) \times \mathsf{Orb}_G(\mathcal{C}_2)$ chosen uniformly at random in time T_0 with probability $\epsilon > 0$, one can decide if there exists $g \in G$ such that $\mathcal{C}_2 = g\star\mathcal{C}_1$ with probability ϵ in time $T_0 + \mathrm{Poly}(n, \log(q))$.*

Proof On input $\mathcal{C}_1, \mathcal{C}_2$, one draws $h_1, h_2 \in G$ uniformly at random and computes $\mathcal{C}_1' = h_1 \star \mathcal{C}_1$, $\mathcal{C}_2' = h_2 \star \mathcal{C}_2$. Then if the oracle successfully decides the existence of $g' \in G$ with $\mathcal{C}_2' = g \star \mathcal{C}_1'$, we learn whether there is $g \in G$ such that $\mathcal{C}_2 = g \star \mathcal{C}_1$. □

Proposition 6 (Random self-reducibility of Search-CE) *Let G be either G_{PE}, G_{LE} or G_{CE}. Let $\mathcal{C}_1$ be a $[k, n]$-linear code over $\mathbb{F}_q$, and let $\mathcal{C}_2 \in \mathsf{Orb}_G(\mathcal{C}_1)$. Given access to an oracle that computes $g' \in G$ such that $\mathcal{C}_2' = g' \star \mathcal{C}_1$ for an input $\mathcal{C}_2'$ chosen uniformly at random in $\mathsf{Orb}_G(\mathcal{C}_1)$ in time T_0 with probability $\epsilon > 0$, there is an algorithm that finds g such that $\mathcal{C}_2 = g \star \mathcal{C}_1$ with probability ϵ in time $T_0 + \mathrm{Poly}(n, \log(q))$.*

Proof On input $\mathcal{C}_1, \mathcal{C}_2$, one draws $h \in G$ uniformly at random and computes $\mathcal{C}_2' = h \star \mathcal{C}_2$. Then the oracle returns $g' \in G$ such that $\mathcal{C}_2' = g' \star \mathcal{C}_1$, from which we derive the solution $g = h^{-1} g'$. □

The above statements (which easily generalize to more general group actions) show that there is a worst case to average case reduction for CE *within a given orbit* of the action of G. However, these cannot rule out that the hardness of CE might differ from one orbit to the other. For example, the size of $\mathsf{Stab}(\mathcal{C})$ will influence the hardness of all inputs in $\mathsf{Orb}(\mathcal{C})$: a larger stabilizer (i.e. automorphism group) will make CE easier to solve.

Finally, note that the possible field automorphisms α that can define an instance of CE (resp. Search-CE) can be easily enumerated: they are the powers of the Frobenius automorphism. Consequently, CE reduces efficiently to LE.

Proposition 7 (Reduction from CE to LE) *CE* (*resp. Search-CE*) *efficiently reduces to LE* (*resp. Search-LE*)

Proof Assume $\mathbb{F}_q = \mathbb{F}_{p^n}$, then the automorphisms of $\mathbb{F}_q$ are of the form $\phi, \phi^2, \ldots, \phi^{n-1}$ where $\phi : x \mapsto x^p$ is the Frobenius automorphism. Hence, we can reduce an instance of CE (resp. Search-CE) to $n = O(\log(q))$ instances of LE (resp. Search-LE) by exhaustive search on the automorphism α. □

5 Is Code Equivalence a Hard Problem?

In [17], Petrank and Roth showed two important facts regarding the hardness of PE: it is unlikely to be NP-complete unless the polynomial hierarchy collapses, and the Graph Isomorphism problem reduces to PE. The latter was an argument for the hardness of PE (and CE in general) until an efficient algorithm for the Graph Isomorphism problem was found. We first present the argument justifying that PE is unlikely to be NP-complete. The proof of [17] is phrased with the permutation code equivalence problem, but it readily extends to LE (and hence CE). In the following, rephrase the method of [17] for LE. It relies on the existence of a protocol to prove that two code $\mathcal{C}_1, \mathcal{C}_2$ are non-equivalent.

Proposition 8 (Protocol to prove NON-LE) *Let $\mathcal{C}_1, \mathcal{C}_2$ be two $[n, k]$-codes over $\mathbb{F}_q$. There is an efficient m-round protocol that allows a prover to convince a verifier that $\mathcal{C}_1$ is not linearly equivalent to $\mathcal{C}_2$ with failure probability at most 2^{-m}.*

Proof We directly follow [17, Sect. 3]. We assume that the codes are given by their generating matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$.

(1) **Verifier:** Chooses $b_1, b_2, \ldots, b_m \in \{1, 2\}$ uniformly at random. For each $\ell \leq m$, the verifier computes a generating matrix $\boldsymbol{G}^{(\ell)}$ of a code linearly equivalent to $\mathcal{C}_{b_\ell}$ by multiplying $\boldsymbol{G}_{b_\ell}$ by a random $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ on the left and $\boldsymbol{M} \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ on the right. The verifier sends $\boldsymbol{G}^{(1)}, \ldots, \boldsymbol{G}^{(m)}$ to the prover.
(2) **Prover:** Computes $b'_1, \ldots, b'_m$ and sends them to the verifier. For each ℓ, the verifier chooses $b'_\ell = 1$ if the code generated by $\boldsymbol{G}^{(\ell)}$ is linearly equivalent to $\mathcal{C}_1$ and $b'_\ell = 2$ if it is linearly equivalent to $\mathcal{C}_2$.
(3) **Verifier:** Accepts the proof if $b_\ell = b'_\ell$ for all ℓ.

The above protocol is complete because an honest prover can never fail to convince the verifier. The protocol is also sound because if $\mathcal{C}_1 \sim \mathcal{C}_2$, the probability that the verifier is convinced by a dishonest prover is at most 2^{-m}. Indeed, if $\mathcal{C}_1 \sim \mathcal{C}_2$, the code generated by $\boldsymbol{G}^{(\ell)}$ is distributed uniformly at random in the equivalence class of $\mathcal{C}_1$, as well as in that of $\mathcal{C}_2$. Therefore, the distribution of $b_1, \ldots, b_m$ given $\boldsymbol{G}^{(1)}, \ldots, \boldsymbol{G}^{(m)}$ is the same as that of $b_1, \ldots, b_m$ (i.e. uniform at random). The prover has no advantage by knowing the $\boldsymbol{G}^{(\ell)}$, and therefore, their probability of success is 2^{-m}. □

Corollary 9 (LE is unlikely to be NP-complete) *If LE is* NP*-complete, then the polynomial hierarchy collapses.*

Proof We use Proposition 8 in the same way as in [17, Sect. 3]. Since we have shown a constant-round interactive proof for NON-LE, we can invoke [18] which states that this implies it also has a constant-round Arthur–Merlin protocol [19]. Then, a result of Boppana et al. [20] shows that if the complement of a problem Π has an Arthur–Merlin protocol with a constant number of rounds, and if Π is NP-complete, then the polynomial-time hierarchy collapses. □

In the following, we summarize the reduction of the Graph Isomorphism problem (GI) to PE presented in [17]. Before Babai's efficient method to solve GI [21], this was an argument in favor the hardness of PE. Now, it is no longer relevant to assess the hardness of CE and its variants, but we present it here because it is an important milestone on the way to understand the reductions between CE and other important computational problems.

Definition 4 (*Graph isomorphism*) Two graphs $\mathcal{G} = (V_G, E_G)$ and $\mathcal{H} = (V_H, E_H)$ are said to be isomorphic if there is a bijection $f : V_G \to V_H$ such that two vertices u and v of $\mathcal{G}$ are adjacent in $\mathcal{G}$ if and only if $f(u)$ and $f(v)$ are adjacent in $\mathcal{H}$.

Given $\mathcal{G}, \mathcal{H}$, GI is the problem of deciding whether there is a graph isomorphism between $\mathcal{G}$ and $\mathcal{H}$. In [17, Sect. 4], Petrank and Roth showed how to map an instance of GI to an instance of PE. This is done by using a mapping from a graph $\mathcal{G} = (V_G, E_G)$ to a generator matrix $\boldsymbol{G}$ of an $[n, k]$ code over $\mathbb{F}_2$ such that

(1) $n = 3|E_G| + |V_G|$, and $k = |E_G|$.
(2) The rows of $\boldsymbol{G}$ have Hamming weight $w \leq 5$.
(3) Any set of two rows or more of $\boldsymbol{G}$ sums up to a vector of weight $w \geq 6$.

Up to a permutation of the rows, only one matrix with the above conditions can generate a given code. Given $\mathcal{G}$, we use the incidence matrix $\boldsymbol{A} \in \mathbb{F}_2^{|E_G| \times |V_G|}$ to construct $\boldsymbol{G}$ in the following way:

$$\phi(\mathcal{G}) = \boldsymbol{G} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{A} \end{pmatrix} \in \mathbb{F}_2^{|E_G| \times (3|E_G| + |V_G|)},$$

where $\boldsymbol{I}$ is the $|E_G| \times |E_G|$ identity matrix.

Proposition 10 (GI reduces to PE) *Let $\mathcal{G}_1, \mathcal{G}_2$ be two input graphs. The graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if and only if the codes $\mathcal{C}_1, \mathcal{C}_2$ generated by $\phi(\mathcal{G}_1), \phi(\mathcal{G}_2)$ satisfy $\mathcal{C}_1 \sim \mathcal{C}_2$.*

Proof We sketch the proof whose details can be found in [17, Sect. 4]. First, if $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic with incidence matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ respectively, then there are permutation matrices $\boldsymbol{P}_1 \in \mathcal{S}_{|E_G|}, \boldsymbol{P}_2 \in \mathcal{S}_{|V_G|}$ (where $|E_G| := |E_{G_1}| = |E_{G_2}|$ and $|V_G| := |V_{G_1}| = |V_{G_2}|$) such that $\boldsymbol{A}_2 = \boldsymbol{P}_1 \boldsymbol{A}_1 \boldsymbol{P}_2$. This implies that there are permutation matrices $\hat{\boldsymbol{P}}_1, \hat{\boldsymbol{P}}_2$ such that $\phi(\mathcal{G}_2) = \hat{\boldsymbol{P}}_1 \phi(\mathcal{G}_1) \hat{\boldsymbol{P}}_2$, i.e. $\mathcal{C}_1$ and $\mathcal{C}_2$ are permutationally equivalent.

On the other hand, if $\mathcal{C}_1 \sim \mathcal{C}_2$, we necessarily have $|E_G| := |E_{G_1}| = |E_{G_2}|$ and $|V_G| := |V_{G_1}| = |V_{G_2}|$, and there are $\boldsymbol{S} \in \mathsf{GL}_{|E_G|}(\mathbb{F}_g)$ and $\hat{\boldsymbol{P}} \in \mathcal{S}_{3|E_G|+|V_G|}$ such that

$$\boldsymbol{G}_2 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{A}_2 \end{pmatrix} = \boldsymbol{S}\boldsymbol{G}_1 \hat{\boldsymbol{P}} = \begin{pmatrix} \boldsymbol{S} & \boldsymbol{S} & \boldsymbol{S} & \boldsymbol{S}\boldsymbol{A}_1 \end{pmatrix} \hat{\boldsymbol{P}},$$

where $\boldsymbol{G}_i = \phi(\mathcal{G}_i)$ for $i = 1, 2$. The generator matrices $\boldsymbol{G}_2, \boldsymbol{G}_1 \hat{\boldsymbol{P}}$ both generate the same code, and both satisfy Properties 1, 2, 3 listed above. Hence it must be that $\boldsymbol{S}$ acts by a permutation of the rows of $\boldsymbol{G}_1$, i.e., $\boldsymbol{S} \in \mathcal{S}_{|E_G|}$. Thus, the first $3|E_G|$ entries of a row of $\boldsymbol{S}\boldsymbol{G}_1$ correspond to a unit vector repeated 3 times. To end up with 3 identity blocks in the first $3|E_G|$ columns of $\boldsymbol{G}_2$, the action of $\hat{\boldsymbol{P}}$ on each block of $|E_G|$ columns of $\boldsymbol{S}\boldsymbol{G}_1$ is simply the inverse of that of $\boldsymbol{S}$ Therefore, $\hat{\boldsymbol{P}}$ must be of the form

$$\hat{\boldsymbol{P}} = \begin{pmatrix} \boldsymbol{S}^{-1} & & & \\ & \boldsymbol{S}^{-1} & & \\ & & \boldsymbol{S}^{-1} & \\ & & & \boldsymbol{T} \end{pmatrix},$$

where $\boldsymbol{T} \in \mathcal{S}_{|V_G|}$. This means that $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic since $\boldsymbol{A}_2 = \boldsymbol{S}\boldsymbol{A}_1\boldsymbol{T}$. □

6 A Zero-Knowledge Protocol for CE (LESS)

In the following, we present a zero knowledge protocol based on group actions. It is analogue to the zero knowledge proof of a discrete logarithm, or of a supersingular isogeny between curves defined over $\mathbb{F}_p$. We assume that a group G acts on the set X of generator matrices of $[n, k]$-linear codes defined over $\mathbb{F}_q$ in systematic form. This group can either be G_{CE}, G_{LE}, or G_{PE}, depending on whether we wish to have the security of the scheme rely on the hardness of Search-CE, Search-LE, or Search-PE.

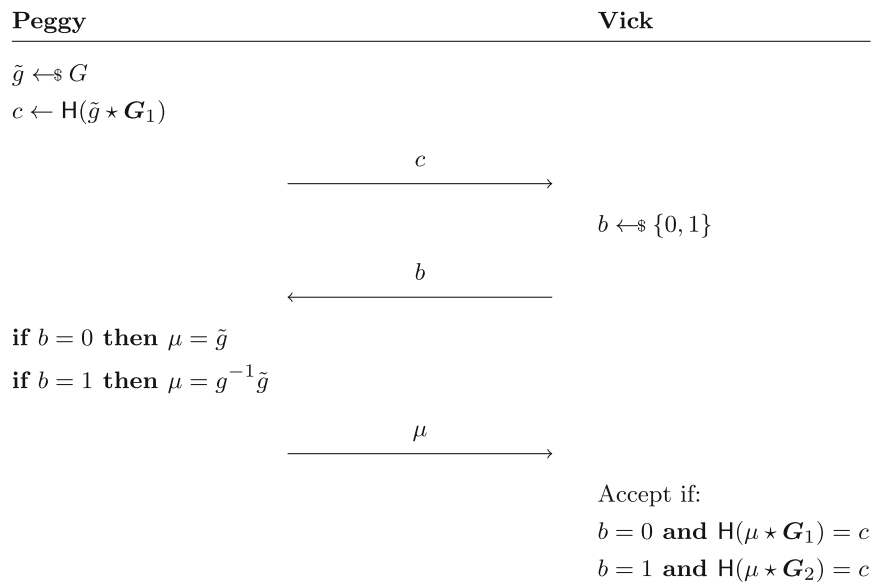

Fig. 1 ZK protocol based on group action

Assume that $\mathcal{C}_1, \mathcal{C}_2$ are given by their generator matrices $\boldsymbol{G}_1, \boldsymbol{G}_2 \in X$ and known to both a prover Peggy and a verifier Vick, and that Peggy must convince Vick that she knows $g \in G$ such that she knows $g \in G$ such that $g \star \boldsymbol{G}_1 = \boldsymbol{G}_2$. The protocol is the repetition of the round described in Fig. 1. It assumes that both players have access to a secure hash function H. We denote by $\mathsf{SF}(\boldsymbol{G})$ the systematic form of the generator matrix $\boldsymbol{G}$. It is used in this protocol to ensure that we have a unique identifier per code (there can be multiple generator matrices).

The protocol described in Fig. 1 was introduced in [3] and named Linear Equivalence Signature Scheme (LESS). The name comes from the fact that the protocol can be turned into a signature scheme via standard techniques (as we describe in the following section), and that the best set of parameters have been obtained when using the variant relying on the hardness of Search-LE. Assuming that it is hard to find $g \in G$ such that $g \star \boldsymbol{G}_1 = \boldsymbol{G}_2$ (i.e. assuming that Search-CE, Search-LE, Search-PE are hard), then the LESS Zero-Knowledge protocol satisfies the following properties:

- *Completeness* Vick accepts with probability 1.
- *2-special soundness* given two valid transcripts of the form (c, b, μ) and (c, b', μ') where $b \neq b'$, there must exist an extractor that finds $g' \in G$ such that $g' \star \boldsymbol{G}_1 = \boldsymbol{G}_2$. This is done by noticing that $(\mu'^{-1}\mu) \star \boldsymbol{G}_1 = \boldsymbol{G}_2$.
- *Honest-Verifier Zero-Knowledge* there must exist a probabilistic polynomial time simulator algorithm $\mathcal{S}$ that, without the knowledge of the secret g, is able to produce a transcript which is indistinguishable from one obtained after an interaction with an honest verifier. This derives from the fact that whether $b = 0$ or $b = 1$, the group element $\mu \in G$ produced by Peggy is distributed uniformly at random in G. Hence, $\mathcal{S}$ can produce transcripts the following way:

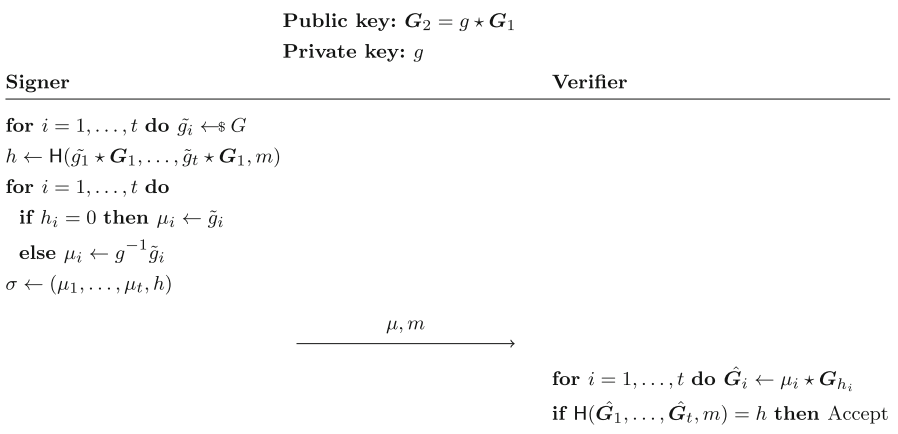

Fig. 2 The LESS signature scheme

- For $b = 0$: $\mu \leftarrow\$G$, and return $(c = \mathsf{H}(\mu \star \boldsymbol{G}_1), b = 0, \mu)$.
- For $b = 1$: $\mu \leftarrow\$G$, and return $(c = \mathsf{H}(\mu \star \boldsymbol{G}_2), b = 1, \mu)$.

Because of the uniform distribution of μ, such transcript is indistinguishable from the one coming from an interaction between Peggy and Vick.

After t rounds of the Σ-protocol described in Fig. 1, an impersonator has cheating probability close to $\frac{1}{2^t}$. This means that if an impersonator who does not possess the secret $g \in G$ is able to observe a number of transcripts from honest executions, before producing a commitment, receiving a corresponding challenge, and finally outputting its response, their probability of success of all t rounds is close to $\frac{1}{2^t}$.

Digital signature via the Fiat–Shamir transform

The 3-pass identification scheme with soundness error $1/2$ presented in the previous section can be turned into a digital signature scheme via the standard Fiat–Shamir transform [22]. In a nutshell, in consists in making the proof of knowledge of a $g \in G$ static. For that, we hash all the commitments $\tilde{g}_i \star G$, together with the message to be signed. This produces a string of bits, which are indistinguishable from random in the random oracle model. These bits are used as the challenges $b \in \{0, 1\}$, telling the signer whether to produce the response $\mu_i = \tilde{g}_i$ or $\mu_i = g^{-1}\tilde{g}_i$. In the end, the signature is the concatenation of the hash and all the μ_i.

7 Practical Optimizations

In [2, 23], practical optimizations were designed to enhance the performance of LESS. The most impactful ones can be summarized by these three orthogonal variants (which can be combined):

- LESS-M: instead of using challenges $b = \{0, 1\}$, we use multibit challenges $b \in \{0, 1\}^\ell$, and a public key consisting of ℓ generator matrices of codes equivalent to $\boldsymbol{G}$.

- LESS-F: Noting the reduced communication costs when the challenge is $b = 0$, we use a hash function H that produces hashes of fixed weight, which effectively fixes the number of times $b = 0$ and $b - 1$.
- LESS-IS: instead of transmitting $\mu = g_i^{-1}\tilde{g}$, only send partial information on μ corresponding to its action on the indices outside of an information set.

7.1 LESS-M

In LESS-M, the public key is no longer a unique generator matrix of the form $g \star \boldsymbol{G}$ for a secret $g \in G \subseteq G_{\mathsf{CE}}$. Instead, it is a collection of matrices $\boldsymbol{G}_i := g_i \star \boldsymbol{G}$ for $i = 0, \ldots, r-1$ for some r of the form $r = 2^\ell$. The ZK protocol to prove an equivalence of Fig. 1 can be turned into one to prove the knowledge of all g_i such that $g_i \star \boldsymbol{G} = \boldsymbol{G}_i$. The main change is that Vick draws $b \in \{0, 1\}^\ell$, which corresponds to an element of $0, \ldots, r-1$. Then Peggy sends the response $\mu = g_i^{-1}\tilde{g}$, which is uniformly distributed in G. The verifier accepts if the hash of $\mu \star \boldsymbol{G}_i$ is equal to the commitment. As for the original LESS signature protocol described in Fig. 2, the ZK protocol based on the round previously described can be turned into a digital signature scheme by following the Fiat–Shamir transform. The effect of choosing $b \in \{0, 1\}^\ell$ reduces the changes of success of an impersonator at each round. This means that fewer rounds are necessary to achieve the same level of security of the ZK protocol. After the Fiat–Shamir transform, this reduced communication cost translates into a shorter signature. However, the trade-off is that the public key no longer consist in a unique generator matrix of the form $g \star \boldsymbol{G}$ but rather in a list $g_0 \star \boldsymbol{G}, \ldots, g_{r-1} \star \boldsymbol{G}$. This means that reduction of the signature size comes at the expense of an increase of the size of the keys.

7.2 LESS-F

The LESS-F variant takes advantage of the fact that in the ZK protocol of Fig. 1, when the challenge bit b is 0, then Peggy returns $\mu = \tilde{g}$, which is an element of G generated uniformly at random. Instead of responding by $\tilde{g}$, Peggy can simply send the seed that was used to create the pseudo-random element. This reduces the communication cost, and therefore, after the Fiat–Shamir transform, this reduces the size of the signature. To further decrease communication cost, Vick can perform a choice of $b \in \{0, 1\}$ where $\mathsf{Pr}(b = 1) = \frac{w}{t}$ where w is a parameter and t is the number of rounds. Typically, we want that this probability be less than $1/2$ to have fewer rounds where $b = 1$ and hence reduce communication costs (and signature size). However, with such imbalance, the probability of success of an impersonator during a given round is $\frac{w}{t} \geq \frac{1}{2}$, which means that more rounds are necessary to achieve the same level of security. If λ is the security parameter, we need to ensure that $\log_2 \binom{t}{w} \geq \lambda$. Note that LESS-F and LESS-M can be combined, albeit with a slight adjustment: only $r - 1$ matrices $g_1 \star \boldsymbol{G}, \ldots, g_{r-1} \star \boldsymbol{G}$ are used for the public key. We set $\boldsymbol{G}_0 = \boldsymbol{G}$, which is already a part of the public parameters, and therefore does not need to be included in the key.

7.3 LESS-IS

In the LESS-F variant presented above, tbe challenge corresponding to $b = 1$ incurs the higher communication cost of sending $\mu = g_i^{-1}\tilde{g}$. Overall, this represents the majority of the communication costs, and it accounts for the largest portion of the signature size. In [23], Persichetti and Santini presented an optimization to reduce the size of the group element that is transmitted. It consists in only sending the action corresponding to the indices outside of the information set that has been agreed upon. This means that only $n - k$ columns of the matrix corresponding to the action of μ are required. Practical instantiations of LESS always have $k = n/2$. This means that LESS-IS effectively halves the amount of data transmitted in the case of $b = 1$. This optimization is also present in the reference implementation of the LESS submission to the NIST standardization process [24]. Note that the security assumption could be weaker since an adversary only needs to recover partial information on the secret group action. However, it was proved in [23, Theorem 1] that both assumptions were equivalent.

8 Advanced Functionalities

Some of the advanced functionalities offered by the cryptographic group action framework extend to signatures based on the hardness of CE. Note that the fact that the $\star$ action is non commutative is an obstacle to the adaptation of many exciting primitives such as one-round key exchange *à la* Diffie–Hellman, or encryption schemes. However, other signature functionalities do extend, in particular ring signatures and identity-based signatures, as originally shown in [25].

8.1 Ring Signatures

When a group of people are authorized to sign a document, one might require anonymity of the signers. The concept of ring signature was originally motivated by the case of a whistleblower who wants to remain anonymous while leaving the possibility of asserting they belong to a restricted group of potential signers with the appropriate credentials (i.e. they are no imposter). Later, the concept of ring signature was applied to other contexts, in particular to the anonymous transfer of cryptocurrencies (transactions in a regular blockchain being traceable). From a high level standpoint, a ring signature protocol has N potential signers, who each possess a pair of keys pk_i, sk_i, and a verifier. The functionalities are:

- $\mathsf{Sign}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathsf{sk}_i, m)$: the signer i of the group signs a message with their private key and the public keys of the whole group.
- $\mathsf{Verify}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \sigma, m)$: the verifier checks the validity of the signature σ with respect to the message m. They accept if one of the public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ was used to produce the signature, but they are not able to determine which individual issued σ.

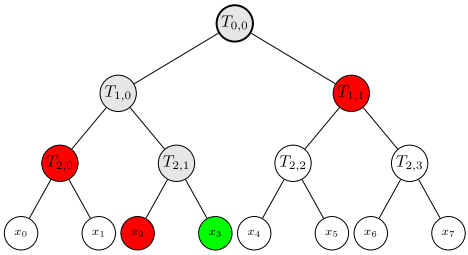

Fig. 3 Example of binary tree for $r = 7$ (from [25, Fig. 4]). The element to verify is in green. The authentication path consists of the red nodes. The reconstructed nodes are marked in gray. The goal is to reconstruct the root (thick line)

To define a ring signature scheme, we need an important cryptographic protocol: a Merkle tree [26]. Assume that $x_0, \ldots, x_{2^t-1}$ is a list of values, and that we have a cryptographically secure hash function H. Then the values $x_0, \ldots, x_{2^t-1}$ can be hashed in a specific order to create a binary tree structure. More specifically, the leaves of the tree are labelled with the $\mathsf{H}(x_i)$, and the node whose children are labeled with values x, y has label $\mathsf{H}(x||y)$. Let $\mathsf{T}_{u,l}$ be the label of the lth node of level $u \leq t$. In particular, the label of the root is $\mathsf{T}_{0,0}$ while the labels of the leaves are $\mathsf{H}(x_i) = \mathsf{T}_{i,t}$. The advantage of hashing the values x_i in this very specific order is that if offers an efficient verification procedure that a given x_i is hashed in a tree whose root is labelled by $\mathsf{T}_{0,0}$. To do that, a prover reveals a *path* to the value x_i. This path consists in the label of the sibling of the leave corresponding to x_i, then the sibling of the parent of these two nodes, and so on until we reach the root. The verifier uses x_i and the first element of the path to produce the parent of level $t - 1$, then they use the second element of the path to produce the parent of level $t - 2$, and so on until the verifier can produce the level of the root, which they can compare with the actual root. If the hash function is collision-resistant, then it is hard to produce a path that allows a verifier to accept on input the legitimate root of the tree and an element $x \notin x_1, \ldots, x_{2^t}$. Moreover, this verification procedure does not reveal the index of x_i to the verifier. All they learn is that $\mathsf{H}(x_i)$ is the label of one of the leaves. We illustrate this procedure in the case of $t = 3$ in Fig. 3. Note that it also assumes both players have access to a commitment scheme $\mathsf{Com}(r, \boldsymbol{G})$ which is typically $\mathsf{H}(r||\boldsymbol{G})$. If a player commits to $\boldsymbol{G}$ through $c = \mathsf{Com}(r, \boldsymbol{G})$, the commitment can be *opened* by a verifier once the player who has committed to $\boldsymbol{G}$ reveals r. Before that, the verifier cannot learn $\boldsymbol{G}$ from c without r.

The protocol described in Fig. 4 illustrates one round of the identification protocol that subsequently leads to a ring signature (via the Fiat–Shamir transform). Peggy proves the knowledge of one of the secret g_i (without revealing the index i). At each round, an impersonator has probability $1/2$ of deceiving Vick, and over sufficiently many rounds, an impersonator only has negligible chance of succeeding all the time.

8.2 Identity-Based Signatures

Under the presence of a trusted entity, an Identity Based Signature scheme (IBS) allows a signer to issue signatures that can be verified by using only their identity, and the public key of the trusted entity. This means that the public key of a signer is simply their identity, which has the potential to considerably simplify the distribution

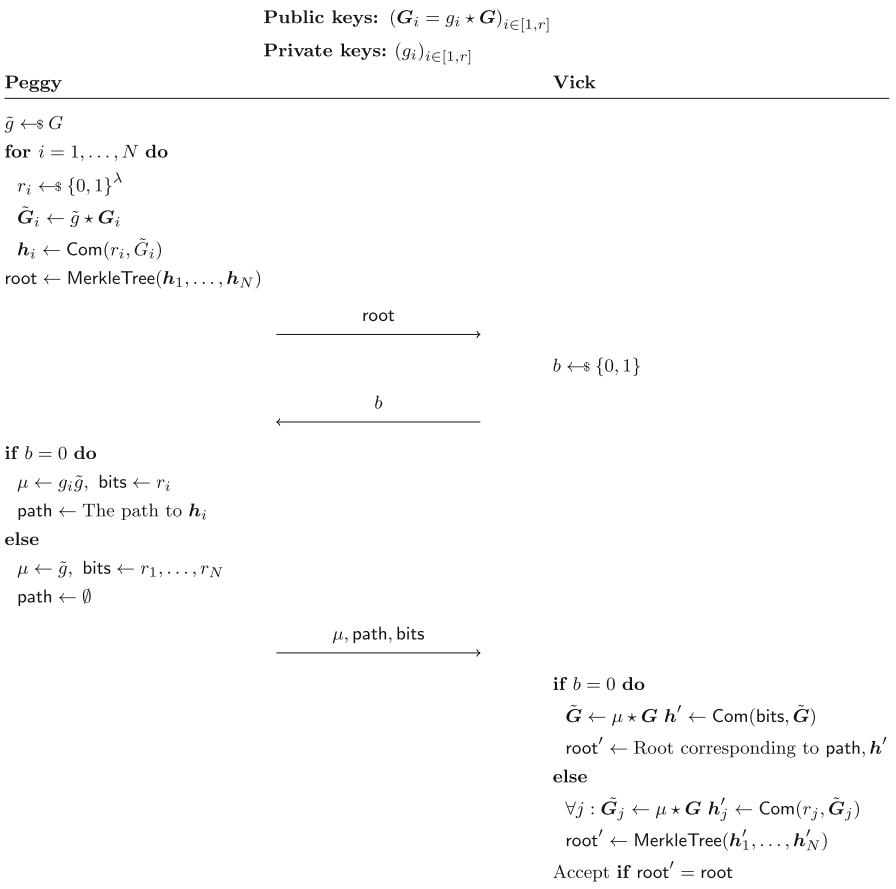

Fig. 4 Round of identification protocol corresponding to ring signature scheme

of keys. As shown in [25, Sect. 6], the LESS protocol (and its optimizations) can be used to produce an IBS via generic techniques.

- **Extraction** The trusted entity has a pair msk, mpks of private and public keys. On input ld the identity of a signer, they issue a random pair sk, pk, and digitally sign (with LESS from Fig. 2) the value (pk, ld) with msk, thus producing the signature σ_m. The trusted entity gives usk $=$ (sk, pk, σ_m) to the user.
- **Signature** The user signs a message m (with LESS from Fig. 2) with the secret key sk to obtain σ_u. Then the identity based signature is $\sigma = (\mathsf{pk}, \sigma_u, \sigma_m)$.
- **Verification** The verifier first verifies σ_u with the message m and the public verification key pk. If this verification is successful, they verify σ_m with the message (pk, ld) and the public verification key mpk.

The above protocol can be used in conjunction with LESS-FM, and it can enjoy all the trade-offs signature size/key size that we described in the previous section.

9 Solving CE with ISD

In this section, we review the most generic methods to solve Search-PE and Search-LE (with the understanding that these enable Search-CE since the latter efficiently reduces to Search-LE). These methods rely on the computation of codewords of a given weight to find the secret map. We present the resolution to Search-PE separately from that of Search-LE because the former is conceptually simpler, but both problems are solved by following the same high-level observation: let $G \subseteq G_{\mathsf{LE}}$ and $g \in G$. If $\boldsymbol{c}$ has weight w, then so does $g\star\boldsymbol{c}$ (where g acts on a codeword by evaluation of the corresponding equivalence map). This allows us to create a list L_1 of elements from $\mathcal{C}_1$ and a list L_2 of elements from $\mathcal{C}_2$ such that elements of L_2 are images of elements of L_1. With enough pairs of the form $(\boldsymbol{c}, g\star\boldsymbol{c})$ one can hope to recover the secret g.

9.1 Solving Search-PE

First, we assume that $G = G_{\mathsf{PE}}$, that is: we are given $\mathcal{C}_1$ and $\mathcal{C}_2$ such that there is $\pi \in \mathcal{S}_n$ that satisfies $\mathcal{C}_2 = \pi(\mathcal{C}_1)$. Our goal is to find π. The method we describe in this section was first introduced by Leon [27], and then improved and generalized by Beullens [28]. Assume that we know two codewords $\boldsymbol{c}_1, \boldsymbol{c}_2$ of weight w such that $\boldsymbol{c}_2 = \pi(\boldsymbol{c}_1)$. This gives us information on π. Indeed, if the i-th coordinate of $\boldsymbol{c}_1$ is zero, and the j coordinate of $\boldsymbol{c}_2$ is non-zero, then we know that $\pi(i) \neq j$. A unique pair $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ is typically not sufficient to formally identify π, but with enough such pairs, a process of elimination can identify $\pi(i)$ uniquely for each i.

Input: list L, containing m pairs $(\boldsymbol{c}, \pi(\boldsymbol{c}))$
Output: permutation π, or report failure

$\boldsymbol{U} \leftarrow n \times n$ matrix made of all ones;
for $\{\boldsymbol{c}, \boldsymbol{c}'\} \in L$ **do**
  for $i \in \{1, \cdots, n\}$ **do**
    $c_i \leftarrow i$-th entries of $\boldsymbol{c}$;
    for $j \in \{1, \cdots, n\}$ **do**
      $c'_j \leftarrow j$-th entry of $\boldsymbol{c}'$;
      /* Filter (i, j) */
      if $(c_i == 0) \neq (c'_j == 0)$ **then**
        $u_{i,j} = 0$;
/* Use $\boldsymbol{U}$ to reconstruct the permutation; if not possible, report failure */
if $\boldsymbol{U}$ is a permutation matrix **then**
  $\pi \leftarrow$ permutation described by $\boldsymbol{U}$;
  return π;
else
  report failure;

Algorithm 1: Permutation recovery from pairs of codewords [29, Alg. 2]

$$\begin{aligned}
\mathsf{Values}(1\boldsymbol{a}) &= \{1,2,2,3,3,3,4\}\\
\mathsf{Values}(2\boldsymbol{a}) &= \{1,1,1,2,3,4,4\}\\
\mathsf{Values}(3\boldsymbol{a}) &= \{1,1,2,3,4,4,4\}\\
\mathsf{Values}(4\boldsymbol{a}) &= \{1,2,2,2,3,3,4\}
\end{aligned}$$

(a)

$$\begin{aligned}
\mathbf{1}^\circ &- \mathsf{Values}(2\boldsymbol{a}) = \{1,1,1,2,3,4,4\}\\
\mathbf{2}^\circ &- \mathsf{Values}(3\boldsymbol{a}) = \{1,1,2,3,4,4,4\}\\
\mathbf{3}^\circ &- \mathsf{Values}(4\boldsymbol{a}) = \{1,2,2,2,3,3,4\}\\
\mathbf{4}^\circ &- \mathsf{Values}(1\boldsymbol{a}) = \{1,2,2,3,3,3,4\}
\end{aligned}$$

(b)

Fig. 5 Example of lexicograph ordering, for the finite field with $q = 5$ elements and a vector $\boldsymbol{a} = (0, 3, 2, 0, 0, 3, 3, 2, 4, 1)$, for which $\mathsf{Lex}(\boldsymbol{a}) = 2\boldsymbol{a}$. **a** The multisets of entries for all scalar multiples of $\boldsymbol{a}$, while **b** reports the lexicographic order of such multisets [29, Fig. 4]

The selection of the pairs of codewords being used is important to maximize the probability of success of Algorithm 1. Indeed, if $\boldsymbol{c} \in \mathcal{C}_1$, $\boldsymbol{c}' \in \pi(\boldsymbol{c}) \in \mathcal{C}_2$ and $\lambda \in \mathbb{F}_q^*$, then the pair $(\lambda\boldsymbol{c}, \lambda\boldsymbol{c}') = (\lambda\boldsymbol{c}, \pi(\lambda\boldsymbol{c}))$ does not provide any extra information about π with respect to $(\boldsymbol{c}, \boldsymbol{c}')$. Hence, Beullens introduced the notion of Lex function that on input a vector $\boldsymbol{a}$ returns $\lambda\boldsymbol{a}$, with $\lambda \in \mathbb{F}_q^*$ such that $\mathsf{Values}(\lambda\boldsymbol{a})$ comes first, in lexicographical order, among the multiset entries of all scalar multiples of $\boldsymbol{a}$ (meaning: we make a canonical choice among all the scalar multiples of $\boldsymbol{a}$). Concretely, this means that every time we draw a pair of the form $(\boldsymbol{c}, \boldsymbol{c}')$, we only keep $(\mathsf{Lex}(\boldsymbol{c}), \mathsf{Lex}(\boldsymbol{c}'))$. This makes the collection of pairs of the form $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ longer (since we only keep one canonical representative up to scalar multiplication), but this avoids the collection of redundant information. To understand how this function operates, we have reported an example in Fig. 5. Embedding the function Lex into the codewords finding algorithm, one can get rid of all unnecessary codewords.

Leon [27] first suggested to collect all codewords of weight w from $\mathcal{C}_1$ into a list L_1, and all codewords of weight w from $\mathcal{C}_2$ into a list L_2 for some weight $w \geq 1$ such that there is at least a few codewords of weight w in $\mathcal{C}_1$ (and therefore in $\mathcal{C}_2$). Then, we know that $\pi(L_1) = L_2$ since π must preserve the weight of vectors. Following Beullens' refinement [28] of this method, one can in fact only keep codewords of the form $\mathsf{Lex}(\boldsymbol{c})$ in both L_1 and L_2, thus reducing the size of each list by a factor $q - 1$ without any loss of information. Then, vectors from both lists are tentatively matched. A necessary condition for $(\boldsymbol{c}, \boldsymbol{c}') \in L_1 \times L_2$ to be of the form $(\boldsymbol{c}, \pi(\boldsymbol{c}'))$ is that both $\boldsymbol{c}$ and $\boldsymbol{c}'$ have the same multiset of entries. When the size q of the field is large enough, this condition is enough to ensure that $\boldsymbol{c}' = \pi(\boldsymbol{c})$ with high probability. However, it is never sufficient to guarantee this with absolute certainty. Therefore, there are two main sources of failure for Algorithm 1:

1. $\mathsf{Values}(\boldsymbol{c}) = \mathsf{Values}(\boldsymbol{c}')$ for $(\boldsymbol{c}, \boldsymbol{c}') \in L_1 \times L_2$, with $\boldsymbol{c}' \neq \pi(\boldsymbol{c})$.
2. L_1 and L_2 are not long enough to provide enough pairs $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ to recover π.

First, let us given an estimate on the length of the list of pairs $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ that is necessary to recover π with good probability. We deal with the case of binary code, which requires the most number of pairs $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ to recover π with Algorithm 1.

Proposition 11 *Assume that $w = cn$ for a constant $0 < c < 1$, and that entries of $\boldsymbol{c}$ are modeled as independent random variables with a probability w/n of being 1 and a probability $(n - w)/n$ of being 0, then the expected length of a list of elements of the form $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ required to identify π exactly is in $O(\ln(n))$.*

Proof Denote by c_i the ith entry of $\boldsymbol{c}$ and by c'_j the jth entry of $\boldsymbol{c}'$ at a given round. We leave $u_{i,j} = 1$ in Algorithm 1 if $c_i = c'_j$ (i.e., we do not rule out the possibility that $\pi(i) = j$). To identify $\pi(i)$ without ambiguity after k rounds, it must be that there is only one j such that at each round, $c_i = c'_j$. Given how we heuristically modeled the behavior of the entries, we have at each round:

$$\begin{aligned}\mathsf{Pr}(c_i = c'_j) &= \mathsf{Pr}(c_i = c'_j = 1) + \mathsf{Pr}(c_i = c'_j = 0)\\ &= \left(\frac{w}{n}\right)^2 + \left(\frac{n-w}{n}\right)^2\\ &= 1 - 2\left(\frac{w}{n}\right)\left(\frac{n-w}{n}\right).\end{aligned}$$

Hence, for each i, the expected number of indices j such that $c_i = c'_j$ during k consecutive rounds is $\mathsf{E_i} = \mathsf{n}\cdot\left(1 - 2\left(\frac{\mathsf{w}}{\mathsf{n}}\right)\left(\frac{\mathsf{n-w}}{\mathsf{n}}\right)\right)^{\mathsf{k}}$. Under the assumption that $w = cn$, $\mathsf{E_i} = \mathsf{n}\cdot(1 - 2\mathsf{c}(1-\mathsf{c}))^{\mathsf{k}} = \mathsf{n}\cdot\mathsf{p}^{\mathsf{k}}$ for some constant $0 < p < 1$. We have $\mathsf{E_i} = 1$ when $k = \frac{\ln(n)}{\ln(1/p)} \in O(\ln(n))$. □

Note that when $q > 2$, we can argue that number of pairs $(\boldsymbol{c}, \pi(\boldsymbol{c}))$ to collect to recover π is less than in the binary case only because we restrict ourselves to elements of the form $\mathsf{Lex(c)}$ in our list. Indeed, otherwise we would have to account for the chances that we get repeated information, which would make the analysis more complicated, and the estimate of the average number of pairs required would depend on q.

Now we turn our attention to the evaluation of the probability that two codewords $\boldsymbol{c}, \boldsymbol{c}'$ are paired because $\mathsf{Values}(\boldsymbol{c}) = \mathsf{Values}(\boldsymbol{c}')$ without having $\boldsymbol{c}' = \pi(\boldsymbol{c})$. If this happens, the algorithm for reconstructing π will likely fail. Therefore, we want to bound the probability of drawing $\boldsymbol{c}, \boldsymbol{c}'$ with $\mathsf{Values}(\boldsymbol{c}) = \mathsf{Values}(\boldsymbol{c}')$ and $\boldsymbol{c}' \neq \pi(\boldsymbol{c})$.

Proposition 12 *After the collection of $O(\ln(n))$ pairs $(\boldsymbol{c}, \boldsymbol{c}') \in \mathcal{C}_1 \times \mathcal{C}_2$ such that $\mathsf{Lex}(\boldsymbol{c}) = \mathsf{Lex}(\boldsymbol{c}')$, the probability that one of the collected pairs $(\boldsymbol{c}, \boldsymbol{c}')$ satisfies $\boldsymbol{c}' \neq \pi(\boldsymbol{c})$ is in $O\left(\ln(n)\frac{n!}{(n-w)!}q^{-n+k}\right)$.*

Proof For each multiset of values $\mathsf{Values}(\boldsymbol{c})$, the probability that there is a weight-w codeword in $\mathcal{C}_1$ with the same multiset of entries (up to multiplication by $\lambda \in \mathbb{F}_q^*$) is at most $\frac{n!}{(n-w)!}q^{-n+k}$. Hence, after $O(\ln(n))$ steps, the probability of drawing at least 1 codeword such that another codeword with the same multiset of entries exists is bounded by $O\left(\ln(n)\frac{n!}{(n-w)!}q^{-n+k}\right)$. This existence is a necessary condition for the collection of $(\boldsymbol{c}, \boldsymbol{c}')$ with $\boldsymbol{c}' \neq \pi(\boldsymbol{c})$, hence the result. □

In [28], Beullens recommends the collection of $\ln(n)$ pairs $\boldsymbol{c}, \boldsymbol{c}'$, with the choice of w low enough to guarantee that $\ln(n)\frac{n!}{(n-w)!}q^{-n+k} < \frac{1}{4}$. Note that it shows that this method is more efficient when q is large. Then comes the question of the size of the list. In the original work of Leon [27], all codewords of weight w are included in the lists L_1, L_2. To reduce the effort, Beullens suggested to collect only enough weight-w vectors to have a constant probability of creating $O(\ln(n))$ pairs $\boldsymbol{c}, \boldsymbol{c}'$. It is sufficient

to take $|L_1| = |L_2| = \sqrt{2N_w n \ln(n)}$ where $N_w = \binom{n}{w}(q-1)^{w-1}\frac{q^k-1}{q^n-1}$ is the number of codewords with Hamming weight w [29, Proposition 4].

To finish estimating the cost of the resolution of Search-PE, we need to estimate the cost of computing a weight-w codeword. This is an instance of the decoding problem (where the syndrome is trivial). The Information Set Decoding (ISD) algorithm returns a weight-w solution to the decoding problem. There is extensive literature on algorithms to perform ISD. From a fundamental perspective, the most recent developments [30, 31] leverage methods to solve the k-list problem. However, they fit the same framework as that of Prange's method [32]. Formally, ISD solve the Syndrome Decoding problem.

Definition 5 [*Syndrome Decoding Problem* (*SDP*)] Let $\mathcal{C}$ be a code of length n and dimension k defined by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, and let $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$, $w \leq n$, find $\boldsymbol{e} \in \mathbb{F}_q^n$ such that $\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s}^T$ and $\boldsymbol{e}$ is of weight w.

Given that the matrix $\boldsymbol{H}$ has n columns but only $n-k$ rows, SDP is equivalent to solving an underdetermined linear system. As long as $w < n-k$, we can hope that the solution $\boldsymbol{e}$ has k zero coefficients. Let $\pi \in \mathcal{S}_n$ be the permutation of the columns of $\boldsymbol{H}$ that brings all these zeros to the last coefficients, then we have

$$\boldsymbol{H}\boldsymbol{e}^T = \begin{pmatrix}\boldsymbol{H}_1 & \boldsymbol{H}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{e}_1^T \\ \boldsymbol{0}\end{pmatrix} = \boldsymbol{H}_1\boldsymbol{e}_1^T = \boldsymbol{s}^T,$$

where $\boldsymbol{H}_1 \in \mathbb{F}_q^{(n-k)\times(n-k)}$, and $\boldsymbol{e}_1 \in \mathbb{F}_q^{n-k}$. We can then solve for $\boldsymbol{e}_1$ and recover $\boldsymbol{e}$. The original strategy to solve SDP due to Prange (which is usually recognized as the first instance of an ISD algorithm) consists in sampling random $\pi \in \mathcal{S}_n$, and for each π, apply the permutation to $\boldsymbol{H}$ and attempt to solve the system of $n-k$ unknowns $\boldsymbol{H}_1\boldsymbol{e}_1^T = \boldsymbol{s}^T$. When an appropriate π is found (which is the difficult part), this yields a solution to SDP at little extra cost.

9.2 Solving Search-LE

Now, assume that $G = G_{\mathsf{LE}}$. This means that we are given $\mathcal{C}_1, \mathcal{C}_2$ such that there is $\tau \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ with $\mathcal{C}_2 = \tau(\mathcal{C}_1)$. The search for $\tau = (\boldsymbol{v}, \pi)$ is done in two steps: first, find π, and then use pairs $(\boldsymbol{c}, \tau(\boldsymbol{c}))$ to derive $\boldsymbol{v}$. Finding π cannot be done by simply matching $\mathsf{Lex}(\boldsymbol{c})$ with $\mathsf{Lex}(\boldsymbol{c}')$ and hoping that $\boldsymbol{c}' = \tau(\boldsymbol{c})$. Indeed, the coordinate-wise multiplication by the entries of $\boldsymbol{v}$ that occurs when we apply τ alters the multiset of entries. Therefore, two codewords that are image of each other do not, in general, have the same multiset of entries.

Instead, Beullens suggested to match subcodes from $\mathcal{C}_1$ and $\mathcal{C}_2$ that are image of each other. More specifically, we use 2-dimensional subcodes. To determine whether two subcodes are image of each other, we use an extension of Lex defined as follow.

Definition 6 *(*[Lex *of a 2-dimensional subcode)* Let V be a 2 dimensional subcode of $\mathcal{C}$. We define $\mathsf{Lex}(\mathsf{V})$ as the first basis of $g \star V$ in lexicographic order for all $g \in G_{\mathsf{LE}}$.

Clearly, if $V_2 = g \star V_1$ for some $g \in G_{\mathsf{LE}}$, we necessarily have $\mathsf{Lex}(\mathsf{V}_1) = \mathsf{Lex}(\mathsf{V}_2)$. However, when looking for pairs $(V_1, \tau(V_1))$ for the secret τ, we might accidentally collect $(V_1, \tau'(V_1))$ for $\tau' \neq \tau$. Before reviewing the high level resolution of Search-LE with Beullens method, let us illustrate the calculation of $\mathsf{Lex}(\mathsf{V})$ on a toy example over the finite field with $q = 5$ elements. Suppose we are given the initial basis

$$\begin{pmatrix} 0\,1\,0\,0\,2\,3\,2\,0\,4 \\ 1\,0\,0\,2\,0\,3\,4\,0\,2 \end{pmatrix}.$$

We act with elements in G_{LE} by scaling all columns so that the entry in the first row is a 1.

$$\begin{pmatrix} 0\,1\,0\,0\,1\,1\,1\,0\,1 \\ 1\,0\,0\,2\,0\,1\,2\,0\,3 \end{pmatrix}.$$

Then we act with elements in G_{LE} (permutations) to sort the columns:

$$\begin{pmatrix} 0\,0\,0\,0\,1\,1\,1\,1\,1 \\ 0\,0\,1\,2\,0\,0\,1\,2\,3 \end{pmatrix}.$$

Finally, we see that we have some degrees of freedom, since the third and fourth columns have a zero in the first row and a non null entry in the second row. Hence, we scale these columns and finally obtain the lexicographic minimum basis

$$\begin{pmatrix} 0\,0\,0\,0\,1\,1\,1\,2\,1 \\ 0\,0\,1\,1\,0\,0\,1\,2\,3 \end{pmatrix}.$$

Now, from a high level stand-point, the Search-LE resolution follows the same framework as the resolution of Search-PE: we generate 2-dimensional subcodes $V_1 \subseteq \mathcal{C}_1$ and $V_2 \subseteq \mathcal{C}_2$ of support of a given weight w, and we match V_1 with V_2 if $\mathsf{Lex}(\mathsf{V}_1) = \mathsf{Lex}(\mathsf{V}_2)$ in the hope that $V_2 = \tau(V_1)$. Then, with enough pairs of the form $(V_1, \tau(V_1))$, we can retrieve π by using the same argument as in the resolution of Search-PE. Namely, if V_1 has support of weight w, its basis as w non-zero columns and $n - w$ zero columns. Under τ, a zero column must be mapped to another zero column and vice versa. Thus, for each $i, j \in [1, n]$, were are able to rule out the possibility that $\pi(i) = j$ whenever the ith column of V_1 is zero and the jth column of $\tau(V_1)$ isn't (or vice versa). Similar to the case of Search-PE, an average number $O(\ln(n))$ pairs $(V_1, \tau(V_1))$ suffices to recover π. Once π is recovered, the scaling factors can be found in many efficient ways. For instance, the permutation can be applied to a generator for $\mathfrak{C}_1$, obtaining $\boldsymbol{G}'$. Then, we choose a parity-check matrix for $\mathfrak{C}_2$, and aim to determine a non-singular diagonal matrix $\boldsymbol{D}$ such that $\boldsymbol{G}'\boldsymbol{D}\boldsymbol{H}_2^\top = \boldsymbol{0}$. This linear system has $k(n-k)$ equations for n unknowns, so that in general it is over constrained and can be easily solved. The n non-null entries of $\boldsymbol{D}$ are the unknown scaling coefficients $\boldsymbol{v}$, which are used to retrieve the desired monomial as $\pi \rtimes \boldsymbol{v}$.

To construct subcodes of support with weight w, Beullens proposed to modify the ISD framework sketched in Sect. 9.1. We denote

$$X_1(w) = \{V \subset \mathcal{C}_1 \mid \dim(V) = 2 \text{ and } |\operatorname{Supp}(V)| \leq w\},$$
$$X_2(w) = \{V \subset \mathcal{C}_2 \mid \dim(V) = 2 \text{ and } |\operatorname{Supp}(V)| \leq w\}.$$

First, assume we fix $V \in X_1(w)$, and let $\pi \in \mathcal{S}_n$ be chosen at random. Then the probability that 2 indices of $\operatorname{Supp}(V)$ get mapped to $[1, k]$ while the $w-2$ remaining ones get mapped to $[k+1, n]$ is

$$P := \frac{\binom{n-k}{w-2}\binom{k}{2}}{\binom{n}{w}}.$$

For each good permutation π, we apply π to the generating matrix of the code and compute its row echelon form according to the first k columns (assuming linear independence of its restriction to these columns). Then one of the $\binom{k}{2}$ vector spaces spanned by two rows of the resulting matrix is $\pi(V)$. Thus, the cost to produce a random subcode of support with size w is in $O\left(\frac{1}{P}\left(\mathsf{Cost}(\text{row echelon}) + \binom{k}{2}\right)\right)$. Alternatively, one can draw random codewords of weight w', pair them randomly, and keep pairs that generate a 2-dimensional subcode of support with size w.

Finally, one needs to argue that the probability of "bad collisions", i.e. of finding pairs (V_1, V_2) such that $\mathsf{Lex}(\mathsf{V_1}) = \mathsf{Lex}(\mathsf{V_2})$ with $V_2 \neq \tau(V_1)$ is low. A given Lex value corresponds to at most $\frac{n!}{(n-w)!}(q-1)^w$ 2-dimensional spaces, and a random 2-dimensional subspace has a probability $\frac{(q^k-1)(q^k-q)}{(q^n-1)(q^n-q)} \approx q^{2(k-n)}$ of being a subcode of $\mathcal{C}_1$. Hence the probability that at least one of the $\ln(n)$ pairs (V_1, V_2) we collect is a bad collision is less than

$$\ln(n)\frac{n!}{(n-w)!}q^{w-1+2(k-n)}.$$

Similar to the case of Search-PE, we choose w small enough to make this probability constant, and the list size can be set to $|L_1| = \sqrt{\binom{n}{w}q^{-n+k+w-2}\ln(n)}$ to get $\ln(n)$ pairs.

10 Solving CE for Codes with Small Hull

The methods described in the previous section are the best ones for random codes. However, there are algorithms that perform better in the case of special codes, in particular when the hull of the input code is small, or even trivfffial. The first such algorithm is the Support Splitting Algorithm (SSA), introduced by Sendrier [33]. This solver is based on the idea of *signature function*, i.e. a function $\mathcal{S}$ that fixes the action of the permutation on each position in the code. A signature function is said to be *fully discriminant* if it returns a different value in each position, and this allows to reveal

the permutation linking the two codes. The signature function proposed by Sendrier in [33] is based on the *hull space* of a code, that is, the intersection between a code and its dual, for which the *weight enumerator* is computed. In particular, to create a dependence between the signature value and the code positions, one can *puncture* the code, i.e. remove coordinates from the codewords. Putting these considerations together, in [33, Sect. 5.2] Sendrier proposes to build a signature as

$$\mathcal{S}(\mathcal{C}_i) := \left\{ \mathsf{Wef}\left(\mathfrak{H}\left(\mathcal{C}_{\setminus i}\right)\right),\ \mathsf{Wef}\left(\mathfrak{H}\left(\mathcal{C}_{\setminus i}^{\perp}\right)\right) \right\},$$

where $\mathcal{C}_{\setminus i}$ is the code obtained from $\mathcal{C}$ punctured in position i, $\mathfrak{H}$ denotes the hull and Wef denotes the Weight Enumerator Function. The hull computation requires simple linear algebra, and comes with a cost of $O(n^3)$ operations in the finite field. To compute the weight enumerator of a code, one usually needs to enumerate all of its codewords: assuming that the hull has dimension h, we can use $O(nq^h)$ as an estimate for the cost of each Wef computation. On the other hand, heuristically, we observe that using $\ln(n)$ refinements is enough to obtain a fully discriminant signature. In the end, the complexity of SSA can be estimated as $O\left(n^3 + n^2 q^h \ln(n)\right)$. Thus, the hull dimension plays a crucial role in the analysis of the performance of SSA. For random codes, this dimension is with high probability equal to a small constant [34], de facto making SSA a polynomial-time solver for Search-PE. On the other hand, SSA is very inefficient for codes that have a large hull. This is, for instance, the case of (weakly) self-dual codes, for which SSA can be made arbitrarily hard by choosing codes with a sufficiently large dimension. SSA can be extended to solve the Search-LE problem as well; however, in this case, the algorithm is less efficient. In fact, such an adaptation requires applying SSA to the *closure* of the code, i.e. the linear code defined as $\{\boldsymbol{c} \otimes \boldsymbol{a}, \boldsymbol{c} \in \mathcal{C}\}$, where $\boldsymbol{a} = (a_1, \ldots, a_{q-1})$ is any ordering of the non-zero elements of $\mathbb{F}_q$. A fundamental point is that, for $q \geq 5$, the closure of a code is always weakly-self dual, and thus has a hull of maximum dimension, leading to exactly the hardest instances for SSA to solve. These results are corroborated by the analysis in [35].

In [36], Bardet et al. proposed a method to solve Search-PE, which takes advantage of the connection between Search-PE and GI in the special case of codes with empty hull. The core idea of [36] is to reduce Search-PE to an instance of the *Weighted Graph Isomorphism (WGI) problem*. This is done by building matrices of the form $\boldsymbol{A}_{\mathcal{C}_i} = \boldsymbol{G}_i^{\top} (\boldsymbol{G}_i \boldsymbol{G}_i)^{-1} \boldsymbol{G}_i$ from the codes considered, and observing that $\boldsymbol{A}_{\mathcal{C}_1} = \boldsymbol{P}^{\top} \boldsymbol{A}_{\mathcal{C}_1} \boldsymbol{P}$ allows to recover the permutation $\boldsymbol{P}$ that connects the two codes. Indeed, $\boldsymbol{A}_{\mathcal{C}_1}$ and $\boldsymbol{A}_{\mathcal{C}_2}$ are interpreted as the adjacency matrices of two graphs, and hence can be given as input to some routine which solves the WGI problem. Given that, to compute $\boldsymbol{A}_{\mathcal{C}_1}$ and $\boldsymbol{A}_{\mathcal{C}_2}$, only $O(n^{2.373})$ operations in the finite field are required thus yielding an overall complexity of $O\left(n^{2.373} C_{WGI}(n)\right)$, where $C_{WGI}(n)$ denotes the complexity of a solver for the weighted graph isomorphism problem. Using Babai's GI algorithm [21] this problem can be solved, in the worst case, with quasi-polynomial complexity.

Finally, in [35], it is shown that one can solve Search-PE by modeling it as a quadratic system. When the hull is trivial, it is possible to add several linear equations (through a technique called *block linearization*), which makes the system very easy to

solve. However, in the general case of a non-trivial hull, the methods proposed by the author (using shortened codes or searching for the closest vector in the code) always end up in exponential complexity; for example, the latter scales proportionally to q^k.

The takeaway from this section is that there exist efficient solvers for very specific instances (small or trivial hulls) of Search-CE; however, for codes with large hulls, these methods become quickly impractical. When using code equivalence in cryptography, it is easy to avoid these attacks. In fact, for the linear equivalence problem, it is enough to consider random codes defined over a large enough alphabet ($q \geq 5$), and then the value $q^h = q^k$ is already large enough for any realistic choice of code parameters. On the other hand, if one wants to use Search-PE as the hardness assumption, choosing a weakly-self dual is sufficient to guarantee maximum hull dimension.

11 Conclusion

In this survey, we have reviewed the construction of optimized digital signature schemes whose security relies on the hardness of the Code Equivalence Problem. We recalled the main results pertaining to the resolution of CE, as well as complexity statements that position CE in the landscape of potential problems to be used in the design of post-quantum cryptographic schemes. In this section, we identify the following important open questions.

First, given the fact that the Code Equivalence group action is not commutative, it is a significant challenge to design cryptographic schemes that are not digital signatures. The exploitation of subgroups or special elements with special commutativity properties could pose a significant threat to the security. Additionally, the linearization attack of [13] also prevents straightforward designs of PRFs. Hence, the design of new schemes based on CE is likely to be a significant challenge.

The known methods to solve CE rely on the search for small codewords through ISD, a decoding method. Meanwhile, the security of the McEliece cryptosystem relies on the hardness of decoding codes that are Goppa codes masked by applying a permutation. Can we establish which assumption is the most difficult? On the one hand, the LESS cryptosystem relies on random instances of CE (whose resolution is currently achieved through an oracle for the decoding problem), while the security of the McEliece cryptosystem rests on the hardness of decoding masked Goppa codes. On the other hand, the decoding problem is NP-hard while CE is unlikely to be NP-complete. An important open question is that of finding a resolution method for CE that does not involve access to a decoding oracle. If such a method exists, then this could be an argument to prove that CE is strictly easier than the decoding problem.

Finally, quantum algorithms for the resolution of CE are understudied. Quantum algorithms for decoding exist [37], but it is an open problem to decide if they can be successfully combined with quantum claw-finding methods [38] to produce an efficient quantum algorithm for the resolution of CE. Additionally, naive adaptations of existing quantum techniques to the framework presented in Sect. 9.1 will likely yield methods with significant quantum memory requirements. A sub-problem of the broader objective of producing quantum algorithms for CE will be to design quantum algorithms that are relevant to the practical analysis of the quantum security of LESS.

Per NIST's guidelines, this implies focusing on methods with low quantum memory requirements.

Acknowledgements This work was supported by NSF Grants #1846166, #2127742, and #2338424.

Data Availability The authors have no data availability declaration to make.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Huffman, W.C., Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press, Cambridge (2003). https://doi.org/10.1017/CBO9780511807077
2. Barenghi, A., Biasse, J., Persichetti, E., Santini, P.: LESS-FM: fine-tuning signatures from the code equivalence problem. In: Cheon, J.H., Tillich, J. (eds.) Post-quantum Cryptography—12th International Workshop, PQCrypto 2021, Proceedings. Lecture Notes in Computer Science, Daejeon, South Korea, 20–22 July 2021, vol. 12841, pp. 23–43. Springer (2021). https://doi.org/10.1007/978-3-030-81293-5_2
3. Biasse, J.-F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) Progress in Cryptology—AFRICACRYPT 2020—12th International Conference on Cryptology in Africa, Proceedings. Lecture Notes in Computer Science, Cairo, Egypt, 20–22 July 2020, vol. 12174, pp. 45–65. Springer (2020). https://doi.org/10.1007/978-3-030-51938-4_3
4. Biasse, J.-F., Micheli, G.: A search-to-decision reduction for the permutation code equivalence problem. In: IEEE International Symposium on Information Theory, ISIT 2023, Taipei, Taiwan, 25–30 June 2023, pp. 602–607. IEEE (2023). https://doi.org/10.1109/ISIT54713.2023.10206940
5. Couveignes, J.M.: Hard homogeneous spaces. IACR Cryptology ePrint Archive, vol. 2006, p. 291 (2006)
6. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. IACR Cryptology ePrint Archive, p. 145 (2006)
7. Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Adv. Math. Commun. **4**(2), 215–235 (2010). https://doi.org/10.3934/AMC.2010.4.215
8. Maze, G.: Algebraic methods for constructing one-way trapdoor functions. PhD Thesis, University of Notre Dame (2003)
9. Monico, C.: Semirings and semigroup actions in public-key cryptography. PhD Thesis, University of Notre Dame (2002)
10. Maze, G., Monico, C., Rosenthal, J.: Public key cryptography based on semigroup actions. Adv. Math. Commun. **1**(4), 489–507 (2007). https://doi.org/10.3934/AMC.2007.1.489
11. Alamati, N., Feo, L.D., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology—ASIACRYPT 2020—26th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part II. Lecture Notes in Computer Science, Daejeon, South Korea, 7–11 December 2020, vol. 12492, pp. 411–439. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3_14
12. Fripertinger, H.: Enumeration of the semilinear isometry classes of linear codes. Bayreuth. Math. Schr. **74**, 100–122 (2005)
13. D'Alconzo, G., Scala, A.D.: Representations of group actions and their applications in cryptography. Cryptology ePrint Archive, Paper 2023/1247 (2023). https://eprint.iacr.org/2023/1247
14. Angluin, D., Lichtenstein, D.: Provable Security of Cryptosystems: A Survey. Technical Report TR-288. Yale University (1983). http://cs-www.cs.yale.edu/publications/techreports/tr288.pdf
15. Feigenbaum, J., Fortnow, L.: On the random-self-reducibility of complete sets. In: Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, 30 June–3 July 1991, pp. 124–132. IEEE Computer Society (1991). https://doi.org/10.1109/SCT.1991.160252

16. Ducas, L., Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology—EUROCRYPT 2022— 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III. Lecture Notes in Computer Science, Trondheim, Norway, 30 May–3 June 2022, vol. 13277, pp. 643–673. Springer (2022). https://doi.org/10.1007/978-3-031-07082-2_23
17. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Trans. Inf. Theory **43**(5), 1602–1604 (1997)
18. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. Adv. Comput. Res. **5**, 73–90 (1989)
19. Babai, L., Moran, S.: Arthur–Merlin games: a randomized proof system, and a hierarchy of complexity classes. J. Comput. Syst. Sci. **36**(2), 254–276 (1988). https://doi.org/10.1016/0022-0000(88)90028-1
20. Boppana, R.B., Håstad, J., Zachos, S.: Does co-NP have short interactive proofs? Inf. Process. Lett. **25**(2), 127–132 (1987). https://doi.org/10.1016/0020-0190(87)90232-8
21. Babai, L.: Graph isomorphism in quasipolynomial time (extended abstract). In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, 18–21 June 2016, pp. 684–697. ACM (2016). https://doi.org/10.1145/2897518.2897542
22. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology—CRYPTO '86, Proceedings. Lecture Notes in Computer Science, Santa Barbara, California, USA, 1986, vol. 263, pp. 186–194. Springer (1986). https://doi.org/10.1007/3-540-47721-7_12
23. Persichetti, E., Santini, P.: A new formulation of the linear equivalence problem and shorter LESS signatures. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology—ASIACRYPT 2023—29th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part VII. Lecture Notes in Computer Science, Guangzhou, China, 4–8 December 2023, vol. 14444, pp. 351–378. Springer (2023). https://doi.org/10.1007/978-981-99-8739-9_12
24. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.-F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.-J.O., Santini, P., Wallace, R.: LESS. Additional Digital Signature Schemes—Round 1 Submissions (2023). https://www.less-project.com/
25. Barenghi, A., Biasse, J., Ngo, T., Persichetti, E., Santini, P.: Advanced signature functionalities from the code equivalence problem. Int. J. Comput. Math. Comput. Syst. Theory **7**(2), 112–128 (2022). https://doi.org/10.1080/23799927.2022.2048206
26. Merkle, R.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) Advances in Cryptology—CRYPTO '87, a Conference on the Theory and Applications of Cryptographic Techniques, Proceedings. Lecture Notes in Computer Science, Santa Barbara, California, USA, 16–20 August 1987, vol. 293, pp. 369–378. Springer (1987). https://doi.org/10.1007/3-540-48184-2_32
27. Leon, J.: Computing automorphism groups of error-correcting codes. IEEE Trans. Inf. Theory **28**(3), 496–511 (1982)
28. Beullens, W.: Not enough LESS: an improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In: Dunkelman, O., Jacobson Jr., M.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020—27th International Conference, Halifax, NS, Canada (Virtual Event), , Revised Selected Papers. Lecture Notes in Computer Science, 21–23 October 2020, vol. 12804, pp. 387–403. Springer (2020). https://doi.org/10.1007/978-3-030-81652-0_15
29. Barenghi, A., Biasse, J., Persichetti, E., Santini, P.: On the computational hardness of the code equivalence problem in cryptography. Adv. Math. Commun. **17**(1), 23–55 (2023). https://doi.org/10.3934/amc.2022064
30. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $2^{0.054n}$. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology—ASIACRYPT 2011—17th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings. Lecture Notes in Computer Science, Seoul, South Korea, 4–8 December 2011. vol. 7073, pp. 107–124. Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_6
31. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: how 1 + 1 = 0 improves information set decoding. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology—EUROCRYPT 2012—31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings. Lecture Notes in Computer Science, Cambridge,

UK, 15–19 April 2012, vol. 7237, pp. 520–536. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_31
32. Prange, E.: The use of information sets in decoding cyclic codes. IEEE Trans. Inf. Theory **8**(5), 5–9 (1962)
33. Sendrier, N.: The support splitting algorithm. IEEE Trans. Inf. Theory **46**(4), 1193–1203 (2000)
34. Sendrier, N.: On the dimension of the hull. SIAM J. Discrete Math. **10**(2), 282–293 (1997). https://doi.org/10.1137/S0895480195294027
35. Saeed, M.A.: Algebraic approach to code equivalence. PhD Thesis, University of Rouen Normandie (2018). https://theses.hal.science/tel-01678829/file/saeedtahamohamed3.pdf
36. Bardet, M., Otmani, A., Saeed-Taha, M.: Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In: IEEE ISIT 2019, 2019, pp. 2464–2468 (2019)
37. Kachigar, G., Tillich, J.: Quantum information set decoding algorithms. In: Lange, T., Takagi, T. (eds.) Post-quantum Cryptography—8th International Workshop, PQCrypto 2017, Proceedings. Lecture Notes in Computer Science, Utrecht, The Netherlands, 26–28 June 2017, vol. 10346, pp. 69–89. Springer (2017). https://doi.org/10.1007/978-3-319-59879-6_5
38. Tani, S.: An improved claw finding algorithm using quantum walk. In: Kucera, L., Kucera, A. (eds.) Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Proceedings. Lecture Notes in Computer Science, Ceský Krumlov, Czech Republic, 26–31 August 2007, vol. 4708, pp. 536–547. Springer (2007). https://doi.org/10.1007/978-3-540-74456-6_48