THE MYSTERIOUS CASE OF CODE EQUIVALENCE: NOTES

VIOLETTA WEGER

Abstract. These notes were prepared for the VT-Swiss Summer School 1-5 July, 2024. Any comments or corrections can be sent to violetta.weger@tum.de The distribution of the notes is of course allowed. Most of the Section 1 "Basics" is copied from the survey [22].

Many thanks to Paolo Santini for helpful explanations and discussions.

Contents

1. Basics 2
1.1. Notation 2
1.2. Basics of Coding Theory 2
1.3. Basics of Cryptography 5
1.4. Basics of Complexity Theory 8
2. Applications in Code-based Cryptography 9
2.1. Idea of LESS 9
2.2. Canonical Forms 10
3. Reductions 10
3.1. Arthur and Merlin or: Code Equivalence is not NP-hard 10
3.2. Reduction from Permutation Equivalence to Graph Isomorphism 11
3.3. Reduction from Linear Equivalence to Permutation Equivalence 13
3.4. Summary 15
3.5. Open Questions 15
4. Solvers 16
4.1. Information Set Decoding 17
4.2. Leon 19
4.3. Beullens 19
4.4. Improved Beullens' Algorithm 20
4.5. Support Splitting 21
4.6. Other Solvers 21
4.7. Summary 21
4.8. Open Questions 22
5. Related Topics 22
5.1. Martix Code Equivalence 22
5.2. Subcode Equivalence 23
5.3. Other Metrics 25
5.4. Open Questions 26
References 26

1. Basics

In this section we cover the main definitions needed for the solvers and the reductions.

Let us start with some notation.

1.1. **Notation.** We denote by $\mathbb{F}_q$ the finite field with q elements, where q is a prime power and denote by $\mathbb{F}_q^\star$ its multiplicative group, i.e., $\mathbb{F}_q \setminus \{0\}$.

We denote by bold upper case or lower case letters matrices, respectively vectors, e.g. $\mathbf{x} \in \mathbb{F}_q^n$ and $\mathbf{A} \in \mathbb{F}_q^{k\times n}$.

The identity matrix of size k is denoted by Id_k. Sets are denoted by upper case letters and for a set S, we denote by $\mid S \mid$ its cardinality. By $\mathrm{GL}_n(\mathbb{F}_q)$ we denote the $n \times n$ invertible matrices over $\mathbb{F}_q$.

Other notation might be standard, forgotten to introduce or introduced right where needed.

1.2. **Basics of Coding Theory.**

Definition 1 (Linear Code)**.** Let $1 \le k \le n$ be integers. Then, an $[n,k]$ *linear code* $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$.

We will only deal with linear codes, so for the rest of these notes, we will omit the word 'linear'.

The parameter n is called the *length* of the code, the elements in the code are called *codewords* and $R = k/n$ is called the *rate* of the code.

Definition 2 (Generator Matrix)**.** Let $k \le n$ be positive integers and let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. Then, a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ is called a *generator matrix* of $\mathcal{C}$ if

$$\mathcal{C} = \left\{ \mathbf{xG} \mid \mathbf{x} \in \mathbb{F}_q^k \right\},$$

that is, the rows of $\mathbf{G}$ form a basis of $\mathcal{C}$.

We will often write $\langle \mathbf{G} \rangle$ to denote the code generated by $\mathbf{G}$.

We denote by $\langle \mathbf{x} \, \langle \mathbf{x}, \mathbf{y} \rangle$ the standard inner product, i.e.,

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i.$$

We can define the dual of an $[n,k]$ linear code $\mathcal{C}$ over $\mathbb{F}_q$ as the orthogonal space of $\mathcal{C}$.

Definition 3 (Dual Code)**.** Let $k \le n$ be positive integers and let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. The *dual code* $\mathcal{C}^\perp$ is an $[n, n-k]$ linear code over $\mathbb{F}_q$, defined as

$$\mathcal{C}^\perp = \{ \mathbf{x} \in \mathbb{F}_q^n \mid \langle \mathbf{x}, \mathbf{y} \rangle = 0 \ \forall \ \mathbf{y} \in \mathcal{C} \}.$$

Definition 4 (Parity-Check Matrix)**.** Let $k \le n$ be positive integers and let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$ with dual code $\mathcal{C}^\perp$. Then, a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ is called a *parity-check matrix* of $\mathcal{C}$, if

$$\mathcal{C} = \left\{ \mathbf{y} \in \mathbb{F}_q^n \mid \mathbf{H}\mathbf{y}^\top = \mathbf{0} \right\},$$

that is $\mathbf{H}$ is a generator matrix of $\mathcal{C}^\perp$ and

$$\mathbf{G}\mathbf{H}^\top = \mathbf{0}.$$

For any $\mathbf{x} \in \mathbb{F}_q^n$, we call $\mathbf{x}\mathbf{H}^\top$ a *syndrome*.

Definition 5 (Hamming Metric)**.** Let n be a positive integer. For $\mathbf{x} \in \mathbb{F}_q^n$, the *Hamming weight* of $\mathbf{x}$ is given by the size of its support, i.e.,

$$\mathrm{wt}_H(\mathbf{x}) = \mid \{i \in \{1, \ldots, n\} \mid x_i \neq 0\} \mid .$$

For $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, the *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$ is given by the number of positions in which they differ, i.e.,

$$d_H(\mathbf{x}, \mathbf{y}) = \mid \{i \in \{1, \ldots, n\} \mid x_i \neq y_i\} \mid .$$

Definition 6 (Minimum Distance)**.** Let $\mathcal{C}$ be a code over $\mathbb{F}_q$. The *minimum Hamming distance* of $\mathcal{C}$ is denoted by $d_H(\mathcal{C})$ and given by

$$d_H(\mathcal{C}) = \min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\}.$$

The main protagonists of code equivalence are the linear isometries.

Definition 7 (Isometry)**.** Let us consider the space V endowed with the distance d. A linear map $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is called *isometry* if it keeps the distance invariant. That is, for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ we have $d(\mathbf{x}, \mathbf{y}) = d(\varphi(\mathbf{x}), \varphi(\mathbf{y}))$.

Proposition 8. *The linear isometries of the Hamming metric consist of monomial transformations and automorphisms on $\mathbb{F}_q$, that is the maps $(\mathbb{F}_q^\star)^n \rtimes (Aut(\mathbb{F}_q) \times \mathcal{S}_n)$.*

Definition 9 (Code Equivalence)**.** Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ be linear codes. We say $\mathcal{C}_1$ is equivalent to $\mathcal{C}_2$, if there exists $\varphi \in (\mathbb{F}_q^\star)^n \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ such that $\varphi(\mathcal{C}_1) = \varphi(\mathcal{C}_2)$.

For cryptography, we mainly focus on a subset of the Hamming-metric isometries, namely the monomial transformations $M_{n,q} = (\mathbb{F}_q^\star)^n \rtimes \mathcal{S}_n$. Any map $\varphi \in M_{n,q}$ can be seen as a matrix $\mathbf{M} = \mathbf{PD}$, where $\mathbf{P}$ is a $n \times n$ permutation matrix and $\mathbf{D} = \mathrm{diag}(\mathbf{v})$ for $\mathbf{v} \in (\mathbb{F}_q^\star)^n$ is a diagonal matrix. When considering any monomial transformation, we get the *linear equivalence.*

Definition 10 (Linear Equivalence)**.** We say that two codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ are *linear equivalent*, if there exists a map $\varphi \in (\mathbb{F}_q^\star)^n \rtimes \mathcal{S}_2$, such that $\varphi(\mathcal{C}_1) = \mathcal{C}_2$.

In an even lighter version, we have the *permutation equivalence.*

Definition 11 (Permutation Equivalence)**.** We say that two codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ are *permutation equivalent*, if there exists a permutation of indices, which transforms $\mathcal{C}_1$ into $\mathcal{C}_2$, that is there exists $\sigma \in \mathcal{S}_n$, such that $\sigma(\mathcal{C}_1) = \mathcal{C}_2$.

Clearly, permutation equivalent codes are also linear equivalent codes.

Exercise 12. Consider the code $\mathcal{C}_1 \subseteq \mathbb{F}_3^3$ generated by $\mathbf{G}_1 = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{pmatrix}$ and the code $\mathcal{C}_2 \subseteq \mathbb{F}_3^3$ generated by $\mathbf{G}_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$. Are the two codes linear equivalent, permutation equivalent or not equivalent?

Proposition 13. *If $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ are permutation equivalent codes, then for any generator matrix $\mathbf{G}_1$ of $\mathcal{C}_1$ and $\mathbf{G}_2$ of $\mathcal{C}_2$, there exists a $\mathbf{S} \in \mathrm{GL}_k(q)$ and an $n \times n$ permutation matrix $\mathbf{P}$ such that*

$$\mathbf{S}\mathbf{G}_1\mathbf{P} = \mathbf{G}_2.$$

If $\mathcal{C}_1, \mathcal{C}_2$ are linear equivalent codes, then for any generator matrix $\mathbf{G}_1$ of $\mathcal{C}_1$ and $\mathbf{G}_2$ of $\mathcal{C}_2$, there exists a a $\mathbf{S} \in \mathrm{GL}_k(q)$ and an $n \times n$ permutation matrix $\mathbf{P}$ and a diagonal matrix $diag(\mathbf{v})$ for $\mathbf{v} \in (\mathbb{F}_q^\star)^n$ such that

$$\mathbf{S}\mathbf{G}_1\mathbf{P}\,diag(\mathbf{v}) = \mathbf{G}_2.$$

Exercise 14. Let $\mathcal{C}_1, \mathcal{C}_2$ be linear equivalent codes. Show that $\mathcal{C}_1^\perp$ is linear equivalent to $\mathcal{C}_2^\perp$. *Hint:* Use the fact that $\mathbf{G}_1\mathbf{H}_1^\top = \mathbf{0}$ and $\mathbf{S}\mathbf{G}_1\mathbf{P}\mathrm{diag}(\mathbf{v}) = \mathbf{G}_2$.

Note that linear equivalent codes have the same minimum distance. Even more is true.

Definition 15 (Weight Enumerator)**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code. For any $w \in \{1, \ldots, n\}$, let us denote by $A_w(\mathcal{C}) = |\{\mathbf{c} \in \mathcal{C} \mid \mathrm{wt}_H(\mathbf{c}) = w\}|$ the *weight enumerator* of $\mathcal{C}$.

Proposition 16. *Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ be linear equivalent codes, then for all $w \in \{1, \ldots, n\}$ we have that*

$$A_w(\mathcal{C}_1) = A_w(\mathcal{C}_2).$$

Note that the other direction is not true: We can have codes with the same weight enumerator, which are not equivalent!

Exercise 17. Let $\varphi \in \mathrm{Aut}(\mathcal{C})$. Show that $\varphi \in \mathrm{Aut}(\mathcal{C}^\perp)$.

Definition 18 (Hull)**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code. Then the (Euclidean) *hull* of $\mathcal{C}$ is given by

$$\mathcal{H}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^\perp.$$

Note that the hull of a random code is with high probability trivial.

Proposition 19. *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be chosen uniform at random. Then, w.h.p. $\mathcal{H}(\mathcal{C}) = \{\mathbf{0}\}$.*

Proof. For any $\mathbf{c} \in \mathcal{H}(\mathcal{C})$, we have that $\mathbf{c} \in \mathcal{C}$, thus there exists $\mathbf{m} \in \mathbb{F}_q^k$ such that $\mathbf{m}\mathbf{G} = \mathbf{c}$. We also have that $\mathbf{c} \in \mathcal{C}^\perp$, hence $\mathbf{c}\mathbf{G}^\top = \mathbf{0}$. Thus, $\mathbf{m}(\mathbf{G}\mathbf{G}^\top) = \mathbf{0}$ and counting the number of $\mathbf{c} \in \mathcal{H}(\mathcal{C})$ is equivalent to counting $\mathbf{m} \in \mathbb{F}_q^k$ with $\mathbf{m}(\mathbf{G}\mathbf{G}^\top) = \mathbf{0}$. Since $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ is a random matrix, also $\mathbf{G}\mathbf{G}^\top \in \mathbb{F}_q^{k\times k}$ is random and has with probability

$$\prod_{i=1}^{k}(1 - q^{-i})$$

full rank. Due to the rank nullity theorem, we have

$$\dim(\ker(\mathbf{G}\mathbf{G}^\top)) = k - \mathrm{rk}(\mathbf{G}\mathbf{G}^\top) = 0,$$

w.h.p. □

Another way to prove this, is to note that any $\mathbf{c} \in \mathcal{H}(\mathcal{C})$ must satisfy

$$\begin{pmatrix}\mathbf{G}\\ \mathbf{H}\end{pmatrix}\mathbf{c}^\top = \mathbf{0}.$$

Note that $\langle\begin{pmatrix}\mathbf{G}\\ \mathbf{H}\end{pmatrix}\rangle = \mathcal{C} + \mathcal{C}^\perp$, which is the smallest code containing $\mathcal{C}(\mathcal{C})$.

Again, we are interested in the dimension of the kernel of this matrix, and due to the rank-nullity theorem in its rank. We can assume that $\mathbf{G}, \mathbf{H}$ are in systematic form and perform row operations to get

$$\begin{pmatrix}\mathbf{G}'\\ \mathbf{H}'\end{pmatrix} = \begin{pmatrix}\mathrm{Id}_k & \mathbf{A}\\ \mathbf{0} & \mathbf{A}\mathbf{A}^\top + \mathrm{Id}_{n-k}\end{pmatrix}.$$

Hence its rank is given by $k + \mathrm{rk}(\mathbf{A}\mathbf{A}^\top + \mathrm{Id}_{n-k})$. Assuming $\mathbf{G}$ was a random matrix, we also have that $\mathbf{A}$ and $\mathbf{A}\mathbf{A}^\top + \mathrm{Id}_{n-k}$ are random matrices, which have with high probability full rank.

Definition 20 (Automorphism Group)**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a linear code. The *automorphism group* of $\mathcal{C}$ is given by the linear isometries that map $\mathcal{C}$ to $\mathcal{C}$.

Note that just like the hull, the automorphism group of a random linear code is w.h.p. trivial [13], i.e., $\mathrm{Aut}(\mathcal{C}) = \{\mathrm{id}\}$.

Exercise 21. Give the automorphism group of $\mathcal{C} = \langle (1,0,0), (0,1,1) \subseteq \mathbb{F}_2^3$.

Exercise 22. Let $\varphi \in \mathrm{Aut}(\mathcal{C})$. Show that $\varphi \in \mathrm{Aut}(\mathcal{C} \cap \mathcal{C}^\perp)$.

The last notion that we will make use of later, another invariant of isometries, is the support of a code.

Definition 23 (Support of a Code)**.** Let $\mathcal{C} \subset \mathbb{F}_q^n$ be a linear code. Then we define its support to be

$$\mathrm{Supp}(\mathcal{C}) = \{i \in \{1, \dots, n\} \mid \exists \mathbf{c} \in \mathcal{C} : \mathbf{c}_i \neq 0\}.$$

Clearly, for a non-degenerate code, the support will be $\{1, \dots, n\}$, however, as soon as we go to subcodes of $\mathcal{C}$, this will change.

Proposition 24. *Let $\mathcal{C}_1, \mathcal{C}_2 \subset \mathbb{F}_q^n$ be linear equivalent. For any subcode $\mathcal{D}_1 < \mathcal{C}_1$ of dimension $r < k$ and support size s there exists $\mathcal{D}_2 < \mathcal{C}_2$ of dimension r with support size s.*

1.3. **Basics of Cryptography.**

1.3.1. *Digital Signature Schemes.* Digital Signature schemes aim at giving a guarantee of the legitimate origin of an object, such as a digital message, exactly as signing a letter to prove that the sender of this letter is really you.

In this process we speak of *authentication*, meaning that a receiver of the message can (with some probability) be sure that the sender is legit, and of *integrity*, meaning that the message has not been altered.

A digital signature scheme again consists of three steps:

(1) key generation,
(2) signing,
(3) verification.

In digital signature schemes we consider two parties, one is the *prover*, that has to prove his identity to the second party called *verifier*, that in turn, verifies the identity of the prover.

As a first step, the prover constructs a secret key $\mathcal{S}$, which he keeps private and a public key $\mathcal{P}$, which is made public. The prover then chooses a message m, and creates a signature s using his secret key $\mathcal{S}$ and the message m, getting a signed message (m, s).

The verifier can easily read the message m, but wants to be sure that the sender really is the prover. Thus, he uses the public key $\mathcal{P}$ and the knowledge of the message m on the signature s to get authentication.

TABLE 1. Digital Signature Scheme

PROVER		VERIFIER
KEY GENERATION		
Construct a secret key $\mathcal{S}$		
Construct a connected public key $\mathcal{P}$		
	$\xrightarrow{\mathcal{P}}$	
SIGNING		
Choose a message m		
Construct a signature s from $\mathcal{S}$ and m		
	$\xrightarrow{m,s}$	
		VERIFICATION
		Verify the signature s using $\mathcal{P}$ and m

The security of a digital signature scheme introduces a new person, the *impersonator*. An impersonator, tries to cheat the verifier and acts as a prover, however without the knowledge of the secret key $\mathcal{S}$. An impersonator wins if a verifier has verified a forged signature. This comes with a certain probability, called *cheating probability* or *soundness error*. In order to ensure integrity a digital signature should always involve a secret key as well as the message itself.

Clearly, the secret key should still be infeasible to recover from the publicly known private key, thus one still has the usual adversary, called Eve, and a security level, as in a public-key encryption scheme.

The performance of a digital signature scheme consists of

- the *communication cost*, that is the total number of bits, that have been exchanged within the process,
- the *signature size*,
- the public key size,
- the secret key size,
- the verification time.

1.3.2. *Zero-Knowledge Protocols.* In a ZK protocol, we have two parties, a prover and a verifier. The prover wants to convince the verifier of the knowledge of a secret object, without revealing said object.

A ZK protocol consists of two stages: key generation and verification. The verification process can consist of several communication steps between the verifier and the prover, in particular, we are interested in the following 3-pass scheme:

(1) The prover prepare *commitment* c and sends it to the verifier.
(2) The verifier randomly picks a *challenge* $b \in \{0, 1\}$, and sends it to the prover.
(3) The prover provides a *response* r_b that only allows to verify c.
(4) The verifier checks the validity of c, (usually by recovering c using r_b and the public key).

TABLE 2. ZK Protocol

PROVER		VERIFIER
KEY GENERATION		
Construct a secret key $\mathcal{S}$		
Construct a connected public key $\mathcal{P}$		
	$\xrightarrow{\mathcal{P}}$	
	VERIFICATION	
Construct commitment c		
	$\xrightarrow{c}$	
		Choose $b \in \{0, 1\}$
	$\xleftarrow{b}$	
Construct response r_b		
	$\xrightarrow{r_b}$	
		Verify c using r_b

A ZK protocol has three important attributes:

(1) *Zero-knowledge:* this means that no information about the secret is revealed during the process.
(2) *Completeness:* meaning that an honest prover will always get accepted.
(3) *Soundness:* for this, we want that an impersonator has only a small cheating probability to get accepted.

Again, for the performance of the protocol, we have

- the communication cost,
- the secret key,
- the public key size,
- the verification time.

In order to achieve an acceptable cheating probability/soundness error, the protocols are often repeated several times (called *rounds*) and only if each instance was verified will the prover be accepted. Thus, if the ZK protocol previously had cheating probability α, after N such rounds we have a cheating probability of α^N.

1.3.3. *Fiat-Shamir Transform.* The *Fiat-Shamir transform* allows us to build a signature scheme from a ZK protocol. To avoid the communication with the verifier that randomly picks a challenge, the challenge is replaced with the seemingly random hash of the commitment and message.

The following table follows the general description of the Fiat-Shamir transform from [11]. We assume that we are given a zero-knowledge identification scheme and a public hash function Hash.

TABLE 3. Fiat-Shamir Transform

PROVER		VERIFIER
KEY GENERATION		
Given the public key $\mathcal{P}$ and the secret key $\mathcal{S}$ of some ZK protocol and a message m		
Choose a commitment c		
Compute $a = \mathsf{Hash}(m, c)$		
Compute a response r to the challenge a		
The signature is the pair $s = (a, r)$		
	$\xrightarrow{m,s}$	
		VERIFICATION
		Use the response r and the public key $\mathcal{P}$ to construct the commitment c
		Check if $\mathsf{Hash}(m, c) = a$

1.4. **Basics of Complexity Theory.** Complexity theory tries to arrange problems in terms of how hard it is to solve them. It usually focuses on decision problems, i.e., a problem with answer "yes" or "no". We often use the complexity terms also for computational problems, as clearly, solving the computational problem would also solve the decisional version. Hence, we will note make a difference between the computational and the decisional version.

Let $\mathcal{P}$ denote a problem. In order to estimate how hard it is to solve $\mathcal{P}$ we have two main complexity classes.

Definition 25. P denotes the class of problems that can be solved by a deterministic Turing machine in polynomial time.

Definition 26. NP denotes the class of problems that can be solved by a non-deterministic Turing machine in polynomial time.

Note that a problem $\mathcal{P}$ is in NP if and only if one can check that a candidate is a solution to $\mathcal{P}$ in polynomial time.

A polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$ follows the following steps:

(1) take any instance I of $\mathcal{R}$,
(2) transform I to an instance I' of $\mathcal{P}$ in polynomial time,
(3) assume that (using an oracle) you can solve $\mathcal{P}$ in the instance I' in polynomial time, getting the solution s',
(4) transform the solution s' in polynomial time to get a solution s of the problem $\mathcal{R}$ in the input I.

The existence of a polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$, informally speaking, means that if we can solve $\mathcal{P}$, we can also solve $\mathcal{R}$ and thus solving $\mathcal{P}$ is at least as hard as solving $\mathcal{R}$.

Definition 27. $\mathcal{P}$ is NP-hard if for every problem $\mathcal{R}$ in NP, there exists a polynomial-time reduction from $\mathcal{R}$ to $\mathcal{P}$.

Informally speaking this class contains all problems which are at least as hard as the hardest problems in NP.

Finally, NP-completeness denotes the intersection of NP-hardness and NP.

Definition 28. A problem $\mathcal{P}$ is NP-complete, if it is NP-hard and in NP.

2. Applications in Code-based Cryptography

Code equivalence is of great interest for code-based based cryptography, as some of the NIST proposals for signature schemes are based on the hardness of this problem.

Note that code equivalence is a group action.

In fact, for a group $(G, \circ)$ with neutral element e and a set X we say $\alpha : G \times X \to X, (g, x) \mapsto \alpha(g, x)$ is a group action, if

(1) $\alpha(e, x) = x$,
(2) $\alpha(g, \alpha(h, x)) = \alpha(gh, x)$.

Thus, we can consider the group of isometries $\mathcal{L}$, with group operation $\circ$ composition and identity element id and the set $X = \{\mathcal{C}' \mid \exists \varphi \in \mathcal{L} : \varphi(\mathcal{C}) = \mathcal{C}'\}$ for a fixed code $\mathcal{C}$ and the group action is given by evaluation $\alpha(\varphi, \mathcal{C}') = \varphi(\mathcal{C}')$.

This is important, as we can build Zero-Knowledge (ZK) protocols from group actions.

2.1. Idea of LESS.

We can now introduce the main signature scheme based on code equivalence in the Hamming metric: LESS [3]. This is a first round candidate for the additional call for post-quantum signature schemes by NIST.

A prover publishes $\mathbf{G}_1 \in \mathbb{F}_q^{k \times n}$ chosen at random and chooses a secret permutation matrix $\mathbf{P}$ and a $\mathbf{v} \in (\mathbb{F}_q^\star)^n$ at random. The prover computes and publishes $\mathbf{G}_2 = \mathbf{S}\mathbf{G}_1\mathbf{P}\mathrm{diag}(\mathbf{v})$, for some $\mathbf{S} \in \mathrm{GL}_k(q)$, while the monomial transformation $\mathbf{P}\mathrm{diag}(\mathbf{v})$ is kept secret.

In order to prove knowledge of the monomial transformation, the prover also computes the commitment $\mathbf{G}' = \mathbf{G}_1\mathbf{P}'\mathrm{diag}(\mathbf{v}')$ for some permutation matrix $\mathbf{P}'$ and $\mathbf{v}' \in (\mathbb{F}_q^\star)^n$. The prover can thus easily provide the monomial transformation from $\mathbf{G}_1$ to $\mathbf{G}'$ (being $\varphi_1 = \mathbf{P}'\mathrm{diag}(\mathbf{v}')$) or the linear isometry from $\mathbf{G}_2$ to $\mathbf{G}'$ (being $\varphi_2 = \mathbf{P}^{-1}\mathrm{diag}(\mathbf{v})^{-1}\mathbf{P}'\mathrm{diag}(\mathbf{v}')$) without revealing any information on the secret monomial from $\mathbf{G}_1$ to $\mathbf{G}_2$ (being $\mathbf{P}\mathrm{diag}(\mathbf{v})$).

In fact, the challenge of the verifier will consist exactly in $b \in \{0, 1\}$, and the response is given by φ_i. The verifier can then check that for the chosen b one has indeed $\varphi_b(\mathbf{G}_b) = \mathbf{G}'$.

Clearly such ZK protocol comes with a cheating probability of $1/2$.

A dishonest prover, i.e., not knowing the secret φ, seeing $\mathbf{G}_1, \mathbf{G}_2$ could simply pick a random isometry ψ and publish $\mathbf{G}' = \psi(\mathbf{G}_1)$ as commitment. If the verifier indeed asks for $b = 1$, the cheating prover gets accepted.

LESS decreases the cheating probability by using multiple public keys. In more details, one chooses several monomial transformations $\mathbf{Q}_1, \ldots, \mathbf{Q}_N$ and publishes $\mathbf{G}_1\mathbf{Q}_1, \ldots, \mathbf{G}_1\mathbf{Q}_N$. The verifier now chooses from which $\mathbf{G}_1\mathbf{Q}_i$ the monomial transformation to $\mathbf{G}'$ should be revealed, thus increasing the challenge space to $N + 1$ and the cheating probability to $\frac{1}{N+1}$.

Since the $\mathbf{G}_1$ was chosen at random it is enough to send a seed as public key. A drawback that comes with LESS is that the commitments and the responses are structured matrices, thus needing a lot of bits to be sent.

A novel version of LESS reduces the signature sizes drastically, by the use of canonical forms.

2.2. **Canonical Forms.** In [8] the authors propose to use canonical forms of matrices, this corresponds to a short representative of a certain equivalence class. As a first step, the monomial transformations are split as $(\mathbf{P},\mathbf{v},\mathbf{P}',\mathbf{v}')$ for $\mathbf{P}$ a $k\times k$ permutation matrix, $\mathbf{P}'$ a $(n-k)\times(n-k)$ permutation matrix and $\mathbf{v}\in(\mathbb{F}_q^\star)^k,\mathbf{v}'\in(\mathbb{F}_q^\star)^{n-k}$, thus getting $\mathbf{G}$ and $\mathbf{G}'$ are monomially equivalent if there exist $(\mathbf{P},\mathbf{v},\mathbf{P}',\mathbf{v}')$ such that

$$\mathbf{G}=\mathbf{S}\mathbf{G}'\begin{pmatrix}\mathbf{P}\mathrm{diag}(\mathbf{v}) & \mathbf{z}\\ \mathbf{z} & \mathbf{P}'\mathrm{diag}(\mathbf{v}')\end{pmatrix},$$

for some $\mathbf{S}\in\mathrm{GL}_k(\mathbb{F}_q)$.

Since one sends the generator matrices in systematic form, this allows the authors to restrict the monomial transformation to the redundant $k\times(n-k)$ part and only send $\mathbf{P}^{-1}\mathrm{diag}(\mathbf{v})^{-1}\mathbf{P}'\mathrm{diag}(\mathbf{v}')$.

The resulting sizes are much smaller now, however, the main question is: are the new instances now easier to solve? And of course, are there other canonical forms, i.e., different representations of $\mathbf{G}$ or φ which could be made use of to reduce the sizes (but keep the hardness of the problem)?

3. Reductions

3.1. **Arthur and Merlin or: Code Equivalence is not NP-hard.** Another complexity class is given by AM, respectively MA. In this class live the problems that can be decided through an Arthur-Merlin protocol. (The only difference between AM and MA is whether Arthur or Merlin first sends a message).

The protocol is similar to the ZK protocol we have seen before, with the prover Merlin and the verifier Arthur. The protocol is a 3 pass protocol and does not need to have the ZK property. The main difference lies in the power of the two parties: while Arthur still has polynomial computational power, Merlin has infinite computation power (indeed Merlin is a wizard).

We say that a problem $\mathcal{P}$ can be decided by the AM protocol if Merlin is able to convince Arthur that the answer upon the instance I is "no". Merlin might be cheating, i.e., the answer to I is actually "yes", and we allow for a soundness error of $\leq 1/3$.

No NP-hard problem can live in AM, else we have AM=PH (the polynomial hierarchy) and this implies a collapse of polynomial hierarchy.

Theorem 29. *Assuming PH $\neq$ AM, code-equivalence is not NP-hard.*

Proof. To show this, we construct the 3-pass Arthur-Merlin protocol. Both parties see the instance $(\mathcal{C}_1,\mathcal{C}_2)$ and Merlin wants to convince Arthur, that the two codes are not equivalent.

Arthur chooses one of the codes, $\mathcal{C}_i$, a random isometry φ and computes a generator matrix $\mathbf{G}'$ for $\varphi(\mathcal{C}_i)$ and sends $\mathbf{G}'$ to Merlin.

Merlin, with the infinite computational power, can compute which code $\mathcal{C}_i$ Arthur has chosen and reply with i. If Merlin was honest, then only one of the codes $\mathcal{C}_1,\mathcal{C}_2$ will be equivalent to the sent $\mathcal{C}'=\langle\mathbf{G}'\rangle$.

If Merlin was cheating and $\mathcal{C}_1$ is equivalent to $\mathcal{C}_2$, then Merlin has two choices and has a success probability of $1/2$.

By repeating this protocol for t rounds, we get a soundness error of 2^{-t} that Arthur accepts a cheating Merlin. □

3.2. **Reduction from Permutation Equivalence to Graph Isomorphism.** Due to Babai's algorithm [2], we know that Graph Isomorphism (GI) takes at most quasi-polynomial time to solve. Thus, a reduction from PEP to GI, i.e., showing that if we can solve GI then we can also solve PEP, implies that PEP is easier than GI. In particular, PEP should not be used for cryptography.

The reduction has been proposed in [4] and has a small drawback: it only works for codes with trivial hull. Since random codes have w.h.p. a trivial hull, we call this a "randomized" reduction, meaning that it will not work for any instance, but it works w.h.p.

Before we can give the reduction, let us recall some graph theory.
A graph $\mathcal{G}$ consists of vertices V and edges E between the vertices, i.e., $E \subset V \times V$.
We will focus on undirected graphs, thus whenever $\{u, v\} \in E$ also $\{v, u\} \in E$ and we label the edges with a weight $w(u, v)$.

We say that two weighted graphs $\mathcal{G} = (V, E)$ and $\mathcal{G}' = (V', E')$ are isomorphic, if there exists a bijective map $f : V \to V'$ with

(1) $\{u, v\} \in E \leftrightarrow \{f(u), f(v)\} \in E'$,
(2) $w(u, v) = w(f(u), f(v))$.

Thus, we can focus on $V = V' = \{1, \ldots, n\}$ and maps $f = \sigma \in \mathcal{S}_n$.

Problem 30 (Weighted Graph Isomorphism Problem). Given $\mathcal{G} = (V, E), \mathcal{G}' = (V, E')$, find $\sigma \in \mathcal{S}_n$, such that $\{u, v\} \in E \leftrightarrow \{\sigma(u), \sigma(v)\} \in E'$ and $w(u, v) = w(\sigma(u), \sigma(v))$.

Definition 31. The *adjacency matrix* of a weighted graph $\mathcal{G}$ is defined as the $n \times n$ matrix $\mathbf{A}$ with entries

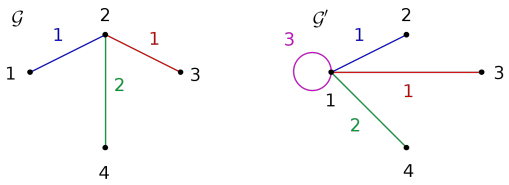
$$\mathbf{A}_{i,j} = \begin{cases} w(i, j) & \text{if } \{i, j\} \in E, \\ 0 & \text{else.} \end{cases}$$

Since we are only interested in undirected graphs, the adjacency matrices are symmetric.

Proposition 32. *Two graphs $\mathcal{G}, \mathcal{G}'$ are isomorphic if and only if there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{P}^\top \mathbf{A} \mathbf{P} = \mathbf{A}'$.*

This almost looks like what we need for PEP, except for the fact that in PEP (treating $\mathbf{A}$ as generator matrix) we also accept $\mathbf{SAP} = \mathbf{A'}$ for any invertible matrix $\mathbf{S}$, not necessarily of the form $\mathbf{P}^\top$.
In fact, one can easily make an example of two graphs, where there exists $\mathbf{S} \in \mathrm{GL}_n(q), \mathbf{P} \in S_n$ with $\mathbf{SAP} = \mathbf{A'}$ but the two graphs are clearly not isomorphic.

Example 33. Let $\mathcal{G} = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{w(1, 2) = 1, w(2, 3) = 1, w(2, 4) = 2\}$ and $\mathcal{G}' = (V, E')$ with $E' = \{w(1, 1) = 3, w(1, 2) = 1, w(1, 3) = 1, w(1, 4) = 2\}$.



These graphs can clearly not be isomorphic. However, their adjacency matrices

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{A}' = \begin{pmatrix} 3 & 1 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}$$

generate the codes $\mathcal{C} = \langle \mathbf{G} \rangle$ with $\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ and $\mathcal{C}' = \langle \mathbf{G}' \rangle$ with $\mathbf{G}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 \end{pmatrix}$, which are clearly permutation equivalent through the permutation $\sigma = (2134)$. And there exists $\mathbf{S} \in GL_4(5), \mathbf{P} \in S_4$ (the permutation matrix of σ) such that $\mathbf{SAP} = \mathbf{A}'$, namely

$$\mathbf{S} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 \end{pmatrix}.$$

Luckily, in order to reduce PEP to GI, we do not have to start with an instance of GI and transform it to an instance of PEP. Instead, we start from codes and transform them to graphs.

Hence, the main question is: given $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, how to define a symmetric matrix in $\mathbb{F}_q^{n \times n}$, which can act as adjacency matrix?

For this [4] introduced the following: For $\mathcal{C}$ with trivial hull, we define

$$\mathbf{A} = \mathbf{G}^\top (\mathbf{G}\mathbf{G}^\top)^{-1} \mathbf{G}.$$

Clearly, this matrix can only exist if $\mathbf{G}\mathbf{G}^\top$ is invertible, i.e., if the hull of $\mathcal{C}$ is trivial. The matrix $\mathbf{A} \in \mathbb{F}_q^{n \times n}$ is symmetric, $\langle \mathbf{A} \rangle = \mathcal{C}$ and, moreover, independent of the choice of $\mathbf{G}$.

In fact, taking any other generator matrix, $\mathbf{SG}$, we get

$$(\mathbf{SG})^\top (\mathbf{SG}(\mathbf{SG})^\top)^{-1} \mathbf{SG} = \mathbf{G}^\top (\mathbf{G}\mathbf{G}^\top) \mathbf{G}.$$

Theorem 34. *PEP is easier than weighted GI, for codes with trivial hull.*

Proof. Assume that the codes in the instance of PEP $(\mathcal{C}, \mathcal{C}')$ have trivial hulls. For arbitrary generator matrices $\mathbf{G}$, respectively $\mathbf{G}'$ take

$$\mathbf{A} = \mathbf{G}^\top (\mathbf{G}\mathbf{G}^\top)^{-1} \mathbf{G},$$
$$\mathbf{A}' = \mathbf{G}'^\top (\mathbf{G}'\mathbf{G}'^\top)^{-1} \mathbf{G}'.$$

And define $\mathcal{G}$, respectively $\mathcal{G}'$, to have adjacency matrices $\mathbf{A}, \mathbf{A}'$.

We now show that the answer to the constructed weighted GI instance is also the answer to the PEP instance. In fact, $\sigma(\mathcal{G}) = \mathcal{G}'$ if and only if $\sigma(\mathcal{C}) = \mathcal{C}'$.

For the first direction, note that for any choice of generator matrices, there exist $\mathbf{S} \in \mathrm{GL}_k(q)$ with $\mathbf{SGP} = \mathbf{G}'$. However, since $\mathbf{A}$ is independent of the choice of basis, we can ignore the $\mathbf{S}$ and get

$$\mathbf{A}' = (\mathbf{GP})^\top (\mathbf{GP}(\mathbf{GP})^\top)^{-1} \mathbf{GP} = \mathbf{P}^\top \mathbf{A} \mathbf{P}.$$

Thus, the two graphs are isomorphic.

The other direction is straightforward, as the $\mathbf{P}^\top \mathbf{A}\mathbf{P} = \mathbf{A}'$ implies that the two codes are equivalent. □

3.3. **Reduction from Linear Equivalence to Permutation Equivalence.** To reduce the LEP to PEP, one can use the *closure* of the code, introduced in [19].

Definition 35 (Closure of Code)**.** Let $\mathcal{C} \subset \mathbb{F}_q^n$. The closure of $\mathcal{C}$ is

$$\widetilde{\mathcal{C}} = \{(\alpha c_i)_{(i,\alpha)\in[1,n]\times\mathbb{F}_q^\star} \mid (c_i)_{i\in[1,n]} \in \mathcal{C}\} \subset \mathbb{F}_q^{n(q-1)}.$$

Thus, if $\mathcal{C} = \langle \mathbf{G} \rangle$, with

$$\mathbf{G} = \begin{pmatrix} | & & | \\ \mathbf{g}_1^\top & \cdots & \mathbf{g}_n^\top \\ | & & | \end{pmatrix}$$

then the generator matrix of the closure is given by

$$\widetilde{\mathbf{G}} = \begin{pmatrix} | & | & & | & | & | & & | \\ \mathbf{g}_1^\top & \alpha\mathbf{g}_1^\top & \cdots & \alpha^{q-2}\mathbf{g}_1^\top & \cdots & \mathbf{g}_n^\top & \alpha\mathbf{g}_n^\top & \cdots & \alpha^{q-2}\mathbf{g}_n^\top \\ | & | & & | & & | & | & & | \end{pmatrix}$$

for some primitive element $\alpha \in \mathbb{F}_q$. The closure $\widetilde{\mathcal{C}}$ has still dimension k, but now length $n(q-1)$.

Proposition 36. *If there exists $\varphi \in M_{n,q}$ with $\varphi(\mathcal{C}) = \mathcal{C}'$, then there exists $\sigma \in \mathcal{S}_n$ with $\sigma(\widetilde{\mathcal{C}}) = \widetilde{\mathcal{C}}'$.*

Hence if $\mathbf{G}, \mathbf{G}'$ are such that there exists $\mathbf{P} \in S_n$ and $\mathbf{v} \in (\mathbb{F}_q^\star)^n$ with $\mathbf{SGP}\mathrm{diag}(\mathbf{v}) = \mathbf{G}'$, for some $\mathbf{S} \in \mathrm{GL}_k(q)$, then $\widetilde{\mathbf{S}}\widetilde{\mathbf{G}}\widetilde{\mathbf{P}} = \widetilde{\mathbf{G}}'$, where

$$\widetilde{\mathbf{P}} = \begin{pmatrix} \mathbf{P}_1 & & \\ & \ddots & \\ & & \mathbf{P}_n \end{pmatrix} \mathbf{Q},$$

with $\mathbf{P}_i \in S_{q-1}$, capture the permutation of $\mathbb{F}_q$, i.e., $\mathbf{v}_i$ and $\mathbf{Q}$ is a block permutation matrix, which keeps the blocks of $(q-1)$ columns together and captures $\mathbf{P}$.

This reduction can always be done, however, depending on q it might have different outcomes.

For this, let us consider the hull of the closure. Recall, that for most solvers and for the reduction to GI, we want a small or trivial hull.

In fact, we can show that for $q \geq 4$ the closure is weakly self dual, meaning $\widetilde{\mathcal{C}} \subset \widetilde{\mathcal{C}}^\perp$. Thus, the hull has the largest possible dimension k as $\mathcal{H}(\widetilde{\mathcal{C}}) = \widetilde{\mathcal{C}}$.

This makes the reduction only interesting for $q < 4$.

Proposition 37. *If $q < 4$, then $\widetilde{\mathcal{C}}$ has w.h.p. a trivial hull. If $q \geq 4$, then $\widetilde{\mathcal{C}}$ is weakly self dual.*

Proof. To understand the hull of the closure, we have to compute

$$\mathbf{X} = \widetilde{\mathbf{G}}\widetilde{\mathbf{G}}^\top = \begin{pmatrix} | & | & & | \\ \mathbf{g}_1^\top & \alpha\mathbf{g}_1^\top & \cdots & \alpha^{q-2}\mathbf{g}_n^\top \\ | & | & & | \end{pmatrix} \begin{pmatrix} - & \mathbf{g}_1 & - \\ - & \alpha\mathbf{g}_1 & - \\ & \vdots & \\ - & \alpha^{q-2}\mathbf{g}_n & - \end{pmatrix}.$$

One can easily check that

$$\mathbf{X}_{i,j} = \sum_{\ell=1}^{n} \mathbf{g}_{\ell,i}\mathbf{g}_{\ell,j} \sum_{\beta\in\mathbb{F}_q^\star} \beta^2.$$

Now the question becomes, how does the sum of squares behave in $\mathbb{F}_q$? If there exists a $\alpha \in \mathbb{F}_q^\star$ with $\alpha^2 \neq 1$, then $\beta \mapsto \alpha\beta$ permutes $\mathbb{F}_q^\star$, hence

$$\sum_{\beta\in\mathbb{F}_q^\star} \beta^2 = \sum_{\beta\in\mathbb{F}_q^\star} (\alpha\beta)^2 = \alpha^2 \sum_{\beta\in\mathbb{F}_q^\star} \beta^2,$$

which implies that $\sum_{\beta\in\mathbb{F}_q^\star} \beta^2 = 0$.

To find such α, with $\alpha^2 \neq 1$, we need $q \geq 4$. Thus, for $q \geq 4$ we have $\mathbf{X} = \mathbf{0}$ and

$$\dim(\mathcal{H}(\widetilde{\mathcal{C}})) = k - \mathrm{rk}(\widetilde{\mathbf{G}}\widetilde{\mathbf{G}}^\top) = k.$$

Hence, for $q \geq 4$, $\widetilde{\mathcal{C}} \subset \widetilde{\mathcal{C}}^\perp$, i.e., the closure is weakly self dual. □

In fact, we can also handle $q = 4$, however, not with the classical dual defined through the standard inner product.

Definition 38 (Hermitian Inner Product)**.** For $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ let us denote by $\langle \mathbf{x}, \mathbf{y} \rangle_H$ the *Hermitian* inner product, i.e.,

$$\langle \mathbf{x}, \mathbf{y} \rangle_H = \sum_{i=1}^{n} x_i y_i^p,$$

where $p = \mathrm{char}(\mathbb{F}_q)$

Note that the Hermitian inner product is not symmetric!

Thus, to define the Hermitian dual, we have to fix on which side we place the codewords.

Definition 39 (Hermitian Dual Code)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$. The *Hermitian dual code* $\mathcal{C}^\star$ is an $[n, n-k]$ linear code over $\mathbb{F}_q$, defined as

$$\mathcal{C}^\star = \{\mathbf{x} \in \mathbb{F}_q^n \mid \langle \mathbf{x}, \mathbf{y} \rangle_H = 0 \ \forall\ \mathbf{y} \in \mathcal{C}\}.$$

Definition 40 (Hermitian Parity-Check Matrix)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_q$ with Hermitian dual code $\mathcal{C}^\star$. Then, a matrix $\mathbf{H}^\star \in \mathbb{F}_q^{(n-k)\times n}$ is called a *Hermitian parity-check matrix* of $\mathcal{C}$, if $\mathbf{H}^\star$ is a generator matrix of $\mathcal{C}^\star$ and

$$\mathbf{G}((\mathbf{H}^\star)^p)^\top = \mathbf{0}.$$

Note that if $\mathbf{H}$ is the common parity-check matrix of $\mathcal{C}$, then $\mathbf{H}^{1/p}$ is a Hermitian parity-check matrix. The Hermitian hull is then defined as $\mathcal{H}^\star(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^\star$.

The Hermitian dual and Hermitian hull are still invariants of isometries.

Exercise 41. Let $\mathcal{C} \subset \mathbb{F}_q^n$ be equivalent to $\mathcal{C}'$. Then $\mathcal{C}^\star$ is equivalent to $(\mathcal{C}')^\star$ (and thus the hulls are as well). *Hint:* Use again that $\mathbf{G}((\mathbf{H}^\star)^p)^\top = \mathbf{0}$ and $\mathbf{GP} = \mathbf{G}'$.

Having this new definition of hull, we can define

$$\mathbf{A}^\star = (\mathbf{G}^p)^\top (\mathbf{G}(\mathbf{G}^p)^\top)\mathbf{G},$$

again $\mathbf{A}^\star$ is independent of the choice of generator matrix, symmetric and exists if $\mathcal{C}$ has trivial Hermitian hull.

Similar to before, for random codes we assume that $\mathbf{G}$ is chosen uniform at random and thus $\mathbf{G}(\mathbf{G}^p)^\top$ has full rank. Thus, random codes have w.h.p. trivial Hermitian hull.

The only thing left to check is that the closure of a code in $\mathbb{F}_4$ has w.h.p. trivial Hermitian hull. For this let α be a primitive element in $\mathbb{F}_4$ and consider

$$\mathbf{X} = \widetilde{\mathbf{G}}(\widetilde{\mathbf{G}}^2)^\top = \begin{pmatrix} | & | & | & & | & | & | \\ \mathbf{g}_1^\top & \alpha\mathbf{g}_1^\top & \alpha^2\mathbf{g}_1^\top & \cdots & \mathbf{g}_n^\top & \alpha\mathbf{g}_n^\top & \alpha^2\mathbf{g}_n^\top \\ | & | & | & & | & | & | \end{pmatrix} \begin{pmatrix} - & \mathbf{g}_1^2 & - \\ - & \alpha^2\mathbf{g}_1^2 & - \\ - & \alpha\mathbf{g}_1^2 & - \\ & \vdots & \\ - & \mathbf{g}_n^2 & - \\ - & \alpha^2\mathbf{g}_n^2 & - \\ - & \alpha\mathbf{g}_n^2 & - \end{pmatrix}.$$

One can easily check that

$$\mathbf{X}_{i,j} = \sum_{\ell=1}^{n} \mathbf{g}_{\ell,i}\mathbf{g}_{\ell,j}^2(1 + \alpha \cdot \alpha^2 + \alpha \cdot \alpha^2) = \sum_{\ell=1}^{n} \mathbf{g}_{\ell,i}\mathbf{g}_{\ell,j}^2.$$

Thus, assuming $\mathbf{G}$ is random, the matrix $\mathbf{X}$ is random as well and has w.h.p. full rank.

3.4. **Summary.** We have seen the following reductions:

(1) We can always reduce LEP to PEP.
(2) If $q \leq 4$ a random LEP instance becomes a random PEP instance, i.e., w.h.p. we have trivial hull.
(3) For $q \geq 5$ a random LEP instance becomes a weakly self dual PEP instance, i.e., full hull.
(4) We can reduce PEP to weighted GI if the hull of the code is trivial.
(5) Code equivalence is not NP-hard (unless the complexity hierarchy collapses).

Thus, neither PEP nor LEP is NP-hard. However, regarding the reduction to the known quasi-polynomial time GI, this is only possible for random PEP instances, and for LEP instances with $q \leq 4$. These are thus the easiest instances of code equivalence and cryptographic systems should use random LEP instances with $q \geq 5$ or weakly self dual PEP instances.

3.5. **Open Questions.** We have seen that the Hermitian hull behaves different to the Euclidean hull and thus an obvious first question is

(1) could we consider different inner products to find more easy instances of code equivalence?
(2) Is there a different reduction of LEP/PEP to GI, which does not require trivial hulls?

4. Solvers

Historically, the first solver for code equivalence was given by Leon [14]. This solver has been improved later by Beullens in [6] and also by Santini *et al.* in [5]. The idea of these solvers is to find subsets $S \subset \mathcal{C}$ and $S' \subset \mathcal{C}'$ which are also invariant under the isometry φ, i.e., $\varphi(S) = S'$. For the smaller sets S, S' it will become easier to find an isometry between them.

Since the weight (and the weight enumerator) is invariant under an isometry, the chosen subset is a set of codewords of (relatively) small weight. To find and list all these codewords one applies Information Set Decoders (ISD).

Note that this is, however, solving a much harder problem: the problem of finding low weight codewords (equivalent to the decoding problem), which is known to be NP-hard.

For this section, we need some more definitions/notation: The q-binomial coefficient is defined as

$$\begin{bmatrix} n \\ k \end{bmatrix}_q = \prod_{i=0}^{k-1} \frac{q^n - q^i}{q^k - q^i} = \frac{\prod_{i=1}^{n}(1-q^i)}{\prod_{i=1}^{k}(1-q^i)\prod_{i=1}^{n-k}(1-q^i)}.$$

To talk about complexities, we write

(1) $f \in \mathcal{O}(g)$ if $\limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$,
(2) $f \in \Omega(g)$ if $\liminf_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| > 0$,
(3) $f \in \Theta(g)$ if $0 < \liminf_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| \leq \limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$.

Definition 42. Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$ and let $S \subseteq \{1,\ldots,n\}$ be a set of size s. Then, we define the *punctured code* $\mathcal{C}^S$ in S as follows

$$\mathcal{C}^S = \{(c_i)_{i\notin S} \mid c \in \mathcal{C}\}.$$

Definition 43 (Information Set)**.** Let $k \leq n$ be positive integers and let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. Then, a set $I \subset \{1,\ldots,n\}$ of size k is called an *information set* of $\mathcal{C}$ if

$$|\,\mathcal{C}\,| = |\,\mathcal{C}_I\,|\,.$$

Definition 44 (Systematic Form)**.** Let $k \leq n$ be positive integers and $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_q$. Then, there exist some permutation matrix $\mathbf{P}$ and some invertible matrix $\mathbf{U}$ that bring $\mathbf{G}$ in *systematic form*, i.e.,

$$\mathbf{UGP} = \begin{pmatrix} \mathrm{Id}_k & \mathbf{A} \end{pmatrix},$$

where $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$. Similarly, there exist some permutation matrix $\mathbf{P'}$ and some invertible matrix $\mathbf{U'}$, that bring $\mathbf{H}$ into systematic form as

$$\mathbf{U'HP'} = \begin{pmatrix} \mathbf{B} & \mathrm{Id}_{n-k} \end{pmatrix},$$

where $\mathbf{B} \in \mathbb{F}_q^{(n-k)\times k}$.

4.1. **Information Set Decoding.** For this section we need some additional notation: let $S \subseteq \{1, \dots, n\}$ be a set of size s, then for a vector $\mathbf{x} \in \mathbb{F}_q^n$ we denote by $\mathbf{x}_S$ the vector of length s consisting of the entries of $\mathbf{x}$ indexed by S. Whereas, for a matrix $\mathbf{A} \in \mathbb{F}_q^{k \times n}$, we denote by $\mathbf{A}_S$ the matrix consisting of the columns of $\mathbf{A}$ indexed by S. For a set S we denote by S^C its complement. For $S \subseteq \{1, \dots, n\}$ of size s we denote by $\mathbb{F}_q^n(S)$ the vectors in $\mathbb{F}_q^n$ having support in S. The projection of $\mathbf{x} \in \mathbb{F}_q^n(S)$ to $\mathbb{F}_q^s$ is then canonical and denoted by $\pi_S(\mathbf{x})$. On the other hand, we denote by $\sigma_S(\mathbf{x})$ the canonical embedding of a vector $\mathbf{x} \in \mathbb{F}_q^s$ to $\mathbb{F}_q^n(S)$.

An ISD algorithm is given a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of a code $\mathcal{C}$, a positive integer t and a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, such that there exists a vector $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight less than or equal to t with syndrome $\mathbf{s}$, i.e., $\mathbf{e}\mathbf{H}^\top = \mathbf{s}$. The aim of the algorithm is to find such a vector $\mathbf{e}$.

To adapt such algorithm to find low weight codewords, one simply sets $\mathbf{s} = \mathbf{0}$ and is thus searching for a codeword $\mathbf{c}$ of weight t.

We can focus (for now) only on Stern's algorithm [21].

(1) Choose a set $J \subset \{1, \dots, n\}$ of size $k + \ell$. Note that the set J contains with high probability an information set.
(2) Bring $\mathbf{H}$ only partially into systematic form, that is up to permutation of columns we have $\mathbf{U}\mathbf{H} = \begin{pmatrix} \mathbf{I}_{n-k-\ell} & \tilde{\mathbf{H}} \\ \mathbf{0} & \mathbf{H}' \end{pmatrix}$, for some $0 \leq \ell \leq n-k$ and $\tilde{\mathbf{H}} \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}, \mathbf{H}' \in \mathbb{F}_q^{\ell\times(k+\ell)}$ and some invertible matrix $\mathbf{U} \in \mathbb{F}_q^{(n-k)\times(n-k)}$. Thus, the codeword $\mathbf{c} = (\tilde{\mathbf{c}}, \mathbf{c}')$ with $\mathbf{c}_{J^C} = \tilde{\mathbf{c}} \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{c}_J = \mathbf{c}' \in \mathbb{F}_q^{k+\ell}$, has to satisfy two equations:
$$\begin{aligned} \tilde{\mathbf{c}} + \mathbf{c}'\tilde{\mathbf{H}}^\top &= \mathbf{0} \\ \mathbf{c}'\mathbf{H}'^\top &= \mathbf{0}. \end{aligned} \tag{4.1}$$
Assuming that $\mathbf{c}'$ has weight v, for some $0 \leq v \leq t$ and $\tilde{\mathbf{c}}$ has the remaining weight $t - v$, we can see that (4.1) forms a smaller instance of a SDP.
(3) Solve the smaller SDP instance given by $\mathbf{H}'$ and target weight v, getting a list $\mathcal{L}$ containing candidates for $\mathbf{c}'$.
(4) For all $\mathbf{c}' \in \mathcal{L}$, compute $\tilde{\mathbf{c}} = -\mathbf{c}'\tilde{\mathbf{H}}^\top$.
(5) If $\mathrm{wt}(\tilde{\mathbf{c}}) = t - v$ return the solution $\mathbf{c} = (\tilde{\mathbf{c}}, \mathbf{c}')$. Else, return to step 1 and choose a different set J.

Note that the cost of an ISD algorithm is given by the number of iterations one needs on average (this is given by the reciprocal of the success probability P of one iteration) times the cost of one iteration (C).

Stern solves the subinstance 4.1 via a collision search. We are given $\mathbf{H}' \in \mathbb{F}_q^{\ell\times(k+\ell)}$ and v. For simplicity, let us assume that $k + \ell$ and v are even. Stern's algorithm splits the sought-after vector $\mathbf{c}' = (\mathbf{c}_1, \mathbf{c}_2)$, were $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{F}_q^{(k+\ell)/2}$ are both assumed to have weight $v/2$. Analogously, we split $\mathbf{H}' = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{pmatrix}$, for $\mathbf{H}_1, \mathbf{H}_2 \in \mathbb{F}_q^{\ell\times(k+\ell)/2}$. Thus, the syndrome equation of the subinstance becomes
$$\mathbf{c}_1\mathbf{H}_1^\top + \mathbf{c}_2\mathbf{H}_2^\top = \mathbf{0}.$$

(1) Build a list of candidates $\mathbf{c}_1$,

$$\mathcal{L}_1 = \{(\mathbf{c}_1, \mathbf{c}_1\mathbf{H}_1^\top) \mid \mathbf{c}_1 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}(\mathbf{c}_1) = v/2\}$$

and similarly a list for $\mathbf{c}_2$:

$$\mathcal{L}_2 = \{(\mathbf{c}_2, -\mathbf{c}_2\mathbf{H}_2^\top) \mid \mathbf{c}_2 \in \mathbb{F}_q^{(k+\ell)/2}, \mathrm{wt}(\mathbf{c}_2) = v/2\}.$$

(2) Go through all elements in $\mathcal{L}_1 \times \mathcal{L}_2$ and search for a collision, i.e., $((\mathbf{c}_1, \mathbf{a}), (\mathbf{c}_2, \mathbf{a})) \in \mathcal{L}_1 \times \mathcal{L}_2$. For each collision add the candidate $\mathbf{c}' = (\mathbf{c}_1, \mathbf{c}_2)$ to the list $\mathcal{L}$.

The cost of one iteration is now dominated by computing the lists $\mathcal{L}_1$, respectively $\mathcal{L}_2$ and the collision search. The average number of collisions, and thus the average size of $\mathcal{L}$ is given by

$$\frac{|\mathcal{L}_1||\mathcal{L}_2|}{q^\ell} = \binom{(k+\ell)/2}{v/2}^2 (q-1)^v q^{-\ell}.$$

The success probability of Stern is given by

$$\binom{n-k-\ell}{t-v}\binom{(k+\ell)/2}{v/2}^2\binom{n}{t}^{-1}.$$

Thus, the cost of Stern is in $\mathcal{O}\left(CP^{-1}\right)$, where

$$P = \binom{n}{t}^{-1}\binom{n-k-\ell}{t-v}\binom{(k+\ell)/2}{v/2}^2,$$
$$C = \left(\binom{(k+\ell)/2}{v/2}(q-1)^{v/2} + \binom{(k+\ell)/2}{v/2}^2 (q-1)^v q^{-\ell}\right).$$

To measure costs, we also introduce the asymptotic cost. For this we assume a fixed q and want to write the cost as $q^{nf(R,q,w/n)}$, that is we are only interested in f, which depends on $R = k/n, w/n$ and q.

We introduce the notation $A = \lim_{n\to\infty} a(n)/n, B = \lim_{n\to\infty} b(n)/n$ and $F_q(A,B) = A\log_q(A) - B\log_q(B) - (A-B)\log_q(A-B)$. Note that with Stirling's formula we have that

$$\lim_{n\to\infty}\frac{1}{n}\log_q\left(\binom{a(n)}{b(n)}\right) = F_q(A,B).$$

In the case of Stern, the asymptotic cost is given by

$$F_q(1,T) - F_q(1-R-L, T-V) - F_q(R+L,V) +$$
$$\max\{F_q(\frac{R+L}{2},\frac{V}{2}) + \frac{V}{2}\log_q(q-1), F_q(R+L,V) + V\log_q(q-1) - L\},$$

where $T = \lim_{n\to\infty} t(n)/n, R = \lim_{n\to\infty} k(n)/n, L = \lim_{n\to\infty} \ell(n)/n, V = \lim_{n\to\infty} v(n)/n$.

This does not look a priori easier, as we would first need to optimize for ℓ, v and usually t is given either by $(d-1)/2$ or d, where we assume d from the Gilbert-Varshamov bound.

4.2. **Leon.** The main observation of Leon [14], is that the sought isometry has to map all codewords of weight w in $\mathcal{C}_1$ to all codewords of weight w in $\mathcal{C}_2$. Thus, Leon first constructs the sets $S_i = \{\mathbf{c} \in \mathcal{C}_i \mid \mathrm{wt}_H(\mathbf{c}) = w\}$ and then searches for $\varphi \in M_{n,q}$ such that $\varphi(S_1) = S_2$.

The cost of finding such a map φ is polynomial in the size of S_i.

Note that for a random code $\mathcal{C} \subset \mathbb{F}_q^n$ of dimension k, we expect the number of codewords having weight w to be

$$\binom{n}{w}(q-1)^w q^{k-n}.$$

The probability that a found φ mapping S_1 to S_2 does not extend to the codes, i.e., $\varphi(\mathcal{C}_1) = \mathcal{C}_2$ is then negligible.

Thus, the main factor in the cost, is constructing S_i using ISD and there is a clear trade-off: the larger we choose w, the easier ISD becomes, however, at the same time, the larger S_i becomes and thus the larger the cost of finding φ.

The main steps of Leon are thus

(1) Construct S_1, S_2 for a chosen weight w.
(2) Find an isometry φ between S_1, S_2.
(3) Check if $\varphi(\mathcal{C}_1) = \mathcal{C}_2$.

Let us compute $N_w = |S_i|$. Clearly, there is no need to store the scalar multiples of $\mathbf{c}$ as well, thus for a random code we compute

$$N_w = \binom{n}{w}(q-1)^{w-1}\frac{q^k - 1}{q^n - 1}.$$

Let us denote by $C_{ISD}(q, n, k, w)$ the cost of an ISD algorithm with these . Then, the cost of Leon's algorithm is given by

$$\mathcal{O}(\ln(N_w)C_{ISD}(q, n, k, w)).$$

Note that the cost of ISD algorithms changes depending on the number of solutions. If we are only interested in finding one, assuming there are N many (ignoring scalars), we can divide its cost by said N and in order to find all the total N is

$$\frac{C_{ISD}}{N} + \frac{C_{ISD}}{N-1} + \cdots = \ln(N)C_{ISD}.$$

If instead we only want L out of the total N, the cost is

$$C_{ISD}L/N.$$

4.3. **Beullens.** Fairly recent, Beullens introduced in [6] a refinement of Leons algorithm.

The algorithm can be split into two versions. Let us start with version 1, tackling permutation equivalence.

The main observation of Beullens, is that a permutation σ does not only fix the weight of a vector, but also the multiset of its entries. Thus, instead of seeing the elements in S_i as vectors $\mathbf{c}_i$, Beullens' algorithm treats them as multisets (now denoted by $S(\mathbf{c}_i)$) and searches for a collision in $S_1 \times S_2$, i.e., $\mathbf{c}_1 \in \mathcal{S}_1$ has the same multiset as $\mathbf{c}_2 \in S_2$. This allows us to store less elements (now multisets) in S_i.

Each found collision is then used to piece-wise reconstruct the permutation: if $\mathbf{c}_1, \mathbf{c}_2$ have the same multisets and $(\mathbf{c}_1)_i \neq (\mathbf{c}_2)_j$, then we guess $\sigma(i) \neq j$. When the number of collisions

is sufficiently high, one has enough information to fully recover σ. In fact, if $|S_i| = L$, it is enough to compute only $\Theta(\sqrt{L}\log(n))$ elements of S_1, S_2 to have on average $\Theta(\log(n))$ many pairs $(\mathbf{c}_1, \sigma(\mathbf{c}_1))$ which is enough to recover the permutation σ.

Note that this algorithm is probabilistic: either a bad collision is found, i.e., codewords $\mathbf{c}_1$ and $\mathbf{c}_2$ that have the same multisets but $\mathbf{c}_2 \neq \sigma(\mathbf{c}_1)$ or the number of collisions is too low.

By setting $L = \sqrt{2N_w n \ln(n)}$, the cost of Beullens algorithm can be approximated as

$$\mathcal{O}\left(\sqrt{\frac{n\ln(n)}{N_w}} C_{ISD}(q,n,k,w)\right).$$

In the second variant, Beullens provides an algorithm to solve LEP. In the case of linear equivalence, the multiset of the entries are clearly not preserved. However, for any subcode $\mathcal{D}_1 < \mathcal{C}_1$ there exists a $\varphi(\mathcal{D}_1) = \mathcal{D}_2 < \mathcal{C}_2$ with the same dimension and support size. In particular, Beullens proposes to use subcodes of dimension 2 and the lists now contain generator matrices in $\mathbb{F}_q^{2\times n}$.

Given the two lists $S_i = \{\mathbf{G}_i \in \mathbb{F}_q^{2\times n} \mid \langle \mathbf{G}_i \rangle < \mathcal{C}_i, |\operatorname{Supp}(\langle \mathbf{G}_i \rangle)| = s\}$ one searches now for collisions, i.e., an isometry $\varphi(\langle \mathbf{G}_1 \rangle) = \langle \mathbf{G}_2 \rangle$. In order to find such isometry, one can either use Leon's algorithm or first Beullens' algorithm to find the permutation and then reconstruct the scalar factors.

In fact, by first defining $\mathbf{G}_1\mathbf{P} = \mathbf{G}_1'$ and then finding $\mathbf{D} = \operatorname{diag}(\mathbf{v})$ such that $\mathbf{G}_1'\mathbf{D}\mathbf{H}_2^\top = \mathbf{0}$. This system has n unknowns $\mathbf{v}$ and $k(n-k)$ equations.

Note that the number of 2-dimensional subcodes of support size s is given by

$$N_s^{(2)} = \frac{\binom{n}{s}(q^2-1)^s - (q-1)^{s+1}\begin{bmatrix}k\\2\end{bmatrix}_q}{(q^2-1)(q^2-q)\begin{bmatrix}n\\2\end{bmatrix}_q}.$$

Setting $L = \sqrt{N_w^{(2)}\lceil \frac{n(n-1)}{2w(n-w)} \rceil}$, the cost of Beullens' algorithm is given by

$$\Omega\left(\frac{L}{N_s^{(2)}} C_{ISD}(q,n,k,s)\right).$$

4.4. **Improved Beullens' Algorithm.** In [5] Santini *et al.* improved the second algorithm of Beullen, using the following idea to construct less 2-dimensional subcodes, faster and with a higher probability of collision:

(1) use an ISD algorithm to find the set $\mathcal{L}$ of size L containing all codewords of weight w,
(2) form all $\binom{L}{2}$ 2-dimensional subcodes $\langle \mathbf{G}_i \rangle$ generated by these codewords,
(3) check if $\langle \mathbf{G}_i \rangle$ has support size s.

In fact, in Beullens' algorithm the two-dimensional subcodes behave like random length s codes, while the improvement forces a collision of columns in $\mathbf{G}_1$ and $\mathbf{G}_2$ of at least $2w - s$. This intersection is captured by the quantity

$$\zeta(w,s) = \begin{cases} 0 & \text{if } s < w, \\ 0 & \text{if } s > \min\{n, 2w\}, \\ \binom{w}{2w-s}\binom{n-w}{s-w}\binom{n}{w}^{-1} & \text{else.} \end{cases}$$

Setting $L = |S_i| = \left(2N_w^{(2)} n \ln(n) \zeta(w,s)^{-1}\right)^{1/4}$, the cost of the improved Beullens' algorithm is then given by

$$\mathcal{O}\left(\frac{\ln(1-L/N_w)}{N_w \ln(1-1/N_w)} C_{ISD}(q,n,k,w)\right).$$

4.5. **Support Splitting.** This algorithm was introduced by Sendrier in [18] and defines a signature function, i.e., a property for each position of the code which is invariant under the permutation. More precisely, for a given code $\mathcal{C}$, position $i \in \{1,\dots,n\}$ one wants $S(\mathcal{C},i) = S(\sigma(\mathcal{C},\sigma(i))$.

Sendrier chooses this signature function (or rather the invariant of the code) to be the weight enumerator of the hull (of a punctured code), i.e.,

$$S(\mathcal{C},i) = W(\mathcal{H}(\mathcal{C}_i)),$$

where $W(\mathcal{C}) = (A_0(\mathcal{C}),\dots,A_n(\mathcal{C}))$ are all the weight enumerators, $\mathcal{C}_i$ is the code $\mathcal{C}$ punctured in the position i.

Having defined such a signature function, one can now compare $S(\mathcal{C},i)$ with $S(\mathcal{C}',i)$ for all $i \in \{1,\dots,n\}$ and if a match is found, recover the permutation between the original codes.

Clearly, the hull is invariant under an isometry:

$$\mathcal{H}(\sigma(\mathcal{C})) = \sigma(\mathcal{C}) \cap \sigma(\mathcal{C}^\perp) = \sigma(\mathcal{C} \cap \mathcal{C}^\perp) = \mathcal{H}(\sigma(\mathcal{C})),$$

the weight enumerator of a code is also invariant:

$$A_w(\mathcal{C}) = A_w(\sigma(\mathcal{C})),$$

but the weight enumerator either involves ISD (having exponential cost) or assumes that the code has a small dimension. Since we compute the weight enumerator of the hull of a punctured code, we get that the cost of the support splitting algorithm is in

$$\mathcal{O}(q^{\dim(\mathcal{H}(\mathcal{C})}).$$

Thus, if the dimension of the hull is large, e.g. for weakly self dual codes, this becomes an exponential time solver, if the dimension is a small constant, we have a polynomial time solver and it simply fails if the hull is trivial (which we have seen happens for random codes w.h.p.).

4.6. **Other Solvers.** One could apply algebraic solvers, that is express $\sigma(\mathcal{C}) = \mathcal{C}'$ as system of equations and try to solve this with Gröbner bases, however, similar to the support splitting algorithm, these approaches have a cost exponential in the dimension of the hull.

Another idea is to use the reduction from PEP to weighted Graph Isomorphism (WGI), which allows us to use any solver of WGI to solve PEP. In fact, Babai's algorithm [2] solves WGI in quasi-polynomial time. The drawback of this approach lies again in the hull: the reduction only works if the code has a trivial hull.

4.7. **Summary.** To summarize, we can say that each solver tries to find a subset/subcode of $S_i \subset \mathcal{C}_i$ and computes invariant $W(S_i)$ of the isometry for this subset and tries to match $W(S_1)$ with $W(S_2)$.

So far, the subsets proposed were sets of small weight codewords $A_w(\mathcal{C})$ or of punctured hulls $\mathcal{H}(\mathcal{C}_i)$. While the proposed invariants were the weight enumerators and the support sizes.

Let us compare the different solvers and when they can be applied:

- To solve PEP for codes with trivial hull, we can use Babai's algorithm [2]: quasi-polynomial time.
- To solve PEP for codes with hull of constant dimension, we can use the support splitting algorithm [18]: polynomial time solver.
- To solve PEP for codes with large hull, the best solver is by [6] and has exponential cost.
- To solve LEP, the best solver is by [5] and has exponential cost.

4.8. **Open Questions.** The open questions that remain, are:

- Are there other invariants that are easier to compute?
- Are there other subcodes / subsets of the codes that lead to an easier instance?
- Are there other easy instances? (Apart from random codes for PEP).

5. Related Topics

5.1. **Martix Code Equivalence.** In the following we introduce *rank-metric codes*, for which we follow the notation of [12].

Let us denote by $\mathbb{F}_q^{n\times m}$ the $n\times m$ matrices over $\mathbb{F}_q$.

Instead of considering subspaces in $\mathbb{F}_q^n$, we can also consider subspaces in $\mathbb{F}_q^{m\times n}$, referred to as *matrix codes*.

Definition 45 (Matrix Codes)**.** An $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^{n\times m}$ is called a *matrix code*.

Thus, instead of a $k\times n$ generator matrix $\mathbf{G}\in\mathbb{F}_q^{k\times n}$, we generate the code with k generating matrices $\mathbf{G}_1,\ldots,\mathbf{G}_k\in\mathbb{F}_q^{m\times n}$, then every codeword is of the form

$$\mathbf{C}=\lambda_1\mathbf{G}_1+\cdots+\lambda_k\mathbf{G}_k,$$

for some $\lambda_i\in\mathbb{F}_q$. Since these codes are only linear over $\mathbb{F}_q$, they are also called $\mathbb{F}_q$-linear codes.

One could define the Hamming metric on such matrices, by either considering the number of non-zero columns or the number of non-zero entries. However, we will be interested in the rank metric.

The dual code of a matrix code, requires a new inner product, which extends the previous standard inner product. For this, recall that the *trace* of a matrix is the sum of the entries on its diagonal.

Definition 46. Let $\mathbf{A},\mathbf{B}\in\mathbb{F}_q^{m\times n}$, then we define their trace product as

$$\mathrm{Tr}(\mathbf{A}\mathbf{B}^\top).$$

Definition 47. Let $\mathcal{C}\subseteq\mathbb{F}_q^{m\times n}$ be a linear matrix code, then its *dual code* is given by

$$\mathcal{C}^\perp=\{\mathbf{A}\in\mathbb{F}_q^{m\times n}\mid \mathrm{Tr}(\mathbf{A}\mathbf{B}^\top)=\mathbf{z}\text{ for all }\mathbf{B}\in\mathcal{C}\}.$$

This product is compatible with the standard inner product on $\mathbb{F}_{q^m}^n$.

The new inner product is in fact also compatible with the vectorization:

Proposition 48. *Let* $\mathbf{A},\mathbf{B}\in\mathbb{F}_q^{m\times n}$*, then*

$$Tr(\mathbf{A}^\top\mathbf{B})=\langle vec(\mathbf{A}),vec(\mathbf{B})\rangle.$$

Definition 49 (Rank Metric)**.** Let $\mathbf{X}, \mathbf{Y} \in \mathbb{F}_q^{m\times n}$. The *rank weight* of $\mathbf{X}$ is given by its rank

$$\mathrm{wt}_R(\mathbf{X}) = \mathrm{rk}(\mathbf{X})$$

and the *rank distance* between $\mathbf{X}$ and $\mathbf{Y}$ is given by

$$d_R(\mathbf{X}, \mathbf{Y}) = \mathrm{wt}_R(\mathbf{X} - \mathbf{Y}).$$

Let $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ be a linear code, then its *minimum rank distance* is given by

$$d_R(\mathcal{C}) = \min\{\mathrm{wt}_R(\mathbf{C}) \mid \mathbf{C} \neq \mathbf{0}, \mathbf{C} \in \mathcal{C}\}.$$

Definition 50 (Matrix Code Equivalence)**.** Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^{m\times n}$. We say that $\mathcal{C}_1$ is equivalent to $\mathcal{C}_2$ if there exists $\varphi \in \mathrm{GL}_m(q) \rtimes \mathrm{GL}_n(q)$ such that $\varphi(\mathcal{C}_1) = \mathcal{C}_2$.

Proposition 51. *The linear isometries of the rank metric in $\mathbb{F}_q^{m\times n}$ for $m \leq n$, are given by $GL_m(q) \rtimes GL_n(q)$ and automorphisms of $\mathbb{F}_q$.*

For applications in cryptography, we again only focus on $\varphi \in \mathrm{GL}_m(q) \rtimes \mathrm{GL}_n(q)$.

Problem 52 (Matrix Code Equivalence (MCE) Problem). Given $\mathbf{G}_1, \ldots, \mathbf{G}_k \in \mathbb{F}_q^{m\times n}$ and $\mathbf{G}'_1, \ldots, \mathbf{G}'_k \in \mathbb{F}_q^{m\times n}$. Find $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q), \mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$, such that for all $\mathbf{C} \in \langle \mathbf{G}_1, \ldots, \mathbf{G}_k\rangle$ we have $\mathbf{ACB} = \mathbf{C}'$ for some $\mathbf{C}' \in \langle \mathbf{G}'_1, \ldots, \mathbf{G}'_k\rangle$.

There exists a polynomial time reduction from the Hamming code equivalence problem in [10]. A nice summary on MCE can be found in [16].

The MEDS signature scheme [7] is based on the same principle as LESS but using the matrix code equivalence instead of linear equivalence.

5.2. **Subcode Equivalence.** One can also generalize LEP/PEP to the following problem.

Problem 53 (Permuted Kernel Problem (PKP)). Given $\mathbf{G} \in \mathbb{F}_q^{k\times n}, \mathbf{H}' \in \mathbb{F}_q^{(n-k')\times n}$ find a permutation matrix $\mathbf{P}$ such that $\mathbf{H}'(\mathbf{GP})^\top = \mathbf{z}$.

This problem has first been introduced by Shamir in [20] and was formulated through parity-check matrices, thus the name *permuted kernel*. In [17] it has been observed, that the formulation of [20] is indeed equivalent to the subcode-equivalence problem.

Problem 54 (Subcode Equivalence Problem (SEP)). Given $\mathbf{G} \in \mathbb{F}_q^{k\times n}, \mathbf{G}' \in \mathbb{F}_q^{k'\times n}$, find permutation matrix $\mathbf{P}$ such that $\langle \mathbf{G}'\rangle \subset \langle \mathbf{GP}\rangle$.

Exercise 55. Show that PKP is equivalent to SEP.

In the following, we will thus only use the subcode equivalence formulation, also for PKP.

There also exists a relaxed version on PKP, which only asks to find a subcode of dimension 1.

Problem 56 (Relaxed PKP). Given $\mathbf{G} \in \mathbb{F}_q^{k\times n}$, $\mathbf{G}' \in \mathbb{F}_q^{k'\times n}$, find $\mathbf{x} \in \mathbb{F}_q^k$ and a permutation matrix $\mathbf{P}$ such that $\mathbf{xGP} \in \langle \mathbf{G}'\rangle$.

Since PKP only asks for permutation equivalence it contains PEP and clearly, PKP contains the Relaxed PKP.

Just like PEP is related to graph isomorphism, PKP is related to sub-graph isomorphism, which is known to be NP-hard [9].

In fact, a simple reduction shows that one can reduce sub-graph isomorphism to PKP, making it also an NP-hard problem.

Let us recall the reduction here.

We say that $\mathcal{G}=(V,E)$ with $|V|=v, |E|=e$ has *incidence matrix* $\mathbf{A}\in\mathbb{F}_2^{e\times v}$, if $\mathbf{A}$ has entries $a_{i,j}$ with

$$a_{i,j}=\begin{cases}1 & \text{if } i=(\ell,j)\in E,\\ 0 & \text{else.}\end{cases}$$

That is the rows correspond to the edges and the columns to the vertices. Considering the edge (a,b), we set a 1 in the position a and in the position b.

Since we consider undirected graphs, the condition $e=(\ell,j)\in E$ should be read as unordered tuple, i.e., also $e=(j,\ell)\in E$.

Example 57. The graph $\mathcal{G}$ with vertex set $V=\{1,2,3,4\}$ and edge set $E=\{(1,2),(2,3),(3,4)\}$ has incidence matrix

$$\mathbf{A}=\begin{pmatrix}1&1&0&0\\0&1&1&0\\0&0&1&1\end{pmatrix}.$$

Clearly, there are different incidence matrices, depending on the ordering of the edges.

Theorem 58. *There exists a reduction from GI to PEP.*

We follow the proof of [15].

Proof. Let $\mathcal{G}=(V,E)$ and $\mathcal{G}'=(V,E')$ be an instance of GI. Let $\mathbf{D}$ and $\mathbf{D}'$ be two incidence matrices for $\mathcal{G}$, respectively $\mathcal{G}'$. We can transform this instance to an instance of PEP, by defining the two generator matrices in $\mathbb{F}_q^{e\times(3e+v)}$

$$\mathbf{G}=\begin{pmatrix}\mathrm{Id}_e & \mathrm{Id}_e & \mathrm{Id}_e & \mathbf{D}\end{pmatrix},$$
$$\mathbf{G}'=\begin{pmatrix}\mathrm{Id}_e & \mathrm{Id}_e & \mathrm{Id}_e & \mathbf{D}'\end{pmatrix}.$$

Let us consider two cases. In the first case, the answer to GI is "yes", as there exists a $f:V\to V$, such that $\{f(u),f(v)\}\in E'$ for all $\{u,v\}\in E$. Thus, there exists a permutation of V which maps one graph to the other and the two incidence matrices $\mathbf{D}$ and $\mathbf{D}'$ are such that

$$\mathbf{QDP}=\mathbf{D}'$$

for some $e\times e$ permutation matrix $\mathbf{Q}$ and $v\times v$ permutation matrix $\mathbf{P}$. Clearly, the codes generated by $\mathbf{G}$ and $\mathbf{G}'$ are then also permutation equivalent.

In the second case, we assume that the two graphs are not isomorphic, hence there exists no permutation on V, which maps $\mathcal{G}$ to $\mathcal{G}'$. Thus, no $v\times v$ permutation matrix $\mathbf{P}$ and no $e\times e$ permutation matrix $\mathbf{Q}$ exists for which $\mathbf{QDP}=\mathbf{D}'$.

The two codes generated by $\mathbf{G}_1$ and $\mathbf{G}_2$ are only permutation equivalent, if we can find $\mathbf{S}\in\mathrm{GL}_n(\mathbb{F}_2)$ and $(3e+v)\times(3e+v)$ permutation matrix $\mathbf{P}$ such that

$$\mathbf{SGP}=\begin{pmatrix}\mathbf{S} & \mathbf{S} & \mathbf{S} & \mathbf{SD}\end{pmatrix}\mathbf{P}=\mathbf{G}'.$$

Note that the first $3e$ columns of $\mathbf{SG}$ consist of all unit vectors of length e, each appearing exactly three times. Hence, the first $3e$ columns of $\mathbf{G}_2$ are obtained by permuting the first $3e$ columns of $\mathbf{SG}$ and thus, we also have the permutation matrix $\mathbf{P}=\mathrm{diag}(\mathbf{S}^{-1},\mathbf{S}^{-1},\mathbf{S}^{-1},\mathbf{T})$, where $\mathbf{T}$ is a $v\times v$ permutation matrix. Hence, if such $\mathbf{S},\mathbf{P}$ exist, we must have $\mathbf{D}'=\mathbf{SDT}$, which is against the assumption that $\mathcal{G}$ and $\mathcal{G}'$ are not isomorphic.

□

Due to this result, we know that PEP (and thus also LEP) are at least as hard as GI.

Since PKP is a subcode-equivalence problem it is equivalent to the subgraph isomorphism problem and hence NP-complete [9]. However, the hardness of the relaxed version is not known.

The signature scheme PERK [1] uses the relaxed PKP in a ZK protocol, together with Multi-Party-Computation techniques. Also PERK is a round 1 candidate for the additional standardization process of NIST.

5.3. **Other Metrics.** There exist also many other metrics one could consider. For example the Lee metric .

Let us consider $\mathbb{F}_p$, for $p > 3$ a prime. Then we can define a different metric, called *Lee metric.*

Definition 59 (Lee Metric)**.** Let $x \in \mathbb{F}_p$, and represent $x \in \{0, \dots, p-1\}$. The *Lee weight* of x is given by

$$\mathrm{wt}_L(x) = \min\{x, |p - x|\}.$$

The largest possible Lee weight is thus $M = (p-1)/2$. Let $\mathbf{x} \in \mathbb{F}_p^n$. The Lee weight is then extended additively on the entries, that is

$$\mathrm{wt}_L(\mathbf{x}) = \sum_{i=1}^{n} \mathrm{wt}_L(x_i).$$

Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_p^n$. Their *Lee distance* is induced by the Lee weight, that is

$$d_L(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x} - \mathbf{y}).$$

Let $\mathcal{C} \subseteq \mathbb{F}_p^n$ be a linear code. The *minimum Lee distance* of $\mathcal{C}$ is given by

$$d_L(\mathcal{C}) = \min\{\mathrm{wt}_L(\mathbf{c}) \mid \mathbf{c} \in \mathcal{C}, \mathbf{c} \neq 0\}.$$

Note that the Lee metric can be defined over any integer residue ring $\mathbb{Z}/m\mathbb{Z}$, for any integer m. However, for the cryptographic purposes it is enough to consider prime fields. Since the Lee metric coincides with the Hamming metric in $\mathbb{F}_2$ and $\mathbb{F}_3$, we only focus on primes $p > 3$.

Note that, $\mathrm{wt}_H(\mathbf{v}) \leq \mathrm{wt}_L(\mathbf{v}) \leq M\mathrm{wt}_H(\mathbf{v})$ and the average Lee weight of the vectors in $\mathbb{F}_p^n$ is given by $(M/2)n$. We, thus, also get that linear code $\mathcal{C} \subseteq \mathbb{F}_p^n$ can correct more errors in the Lee metric as in the Hamming metric, i.e.,

$$d_H(\mathcal{C}) \leq d_L(\mathcal{C}).$$

Proposition 60. *The linear isometries for the Lee metric are given by* $\{\pm 1\}^n \rtimes \mathcal{S}_n$.

Since this is a subset of the linear isometries of the Hamming metric, we also get a problem in between PEP and LEP.

Problem 61 (Lee-metric Code Equivalence (LCE))*.* Given $\mathcal{C}_1, \mathcal{C}_2 \in \mathbb{F}_p^n$, find a linear isometry $\varphi \in \{\pm 1\}^n \rtimes \mathcal{S}_n$, such that $\varphi(\mathcal{C}_1) = \mathcal{C}_2$.

Clearly, if one can solve LEP, one can also solve LCE, and if one can solve LCE one can also solve PEP.

The proof that code equivalence is not NP-hard using the Arthur-Merlin protocol still applies, and hence also LCE is not NP-hard.

Note that the linear isometries of the Lee metric also do not act transitively on the Lee sphere. This makes a construction of a protocol with the ZK property difficult, as any isometry φ would leak information on the secret.

5.4. **Open Questions.** With different metrics a lot of new questions arises, such as

- How hard is relaxed PKP?
- Are there metrics for which code equivalence is NP-hard?

References

[1] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK. In *First Round Submission to the additional NIST Postquantum Cryptography Call*, 2023.

[2] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[3] Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Paolo Santini, and Robert Wallace. LESS. In *First Round Submission to the additional NIST Postquantum Cryptography Call*, 2023.

[4] 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.

[5] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. On the computational hardness of the code equivalence problem in cryptography. *Cryptology ePrint Archive*, 2022.

[6] Ward Beullens. Not enough LESS: an improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In *International Conference on Selected Areas in Cryptography*, pages 387–403. Springer, 2020.

[7] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. MEDS. In *First Round Submission to the additional NIST Postquantum Cryptography Call*, 2023.

[8] Tung Chou, Edoardo Persichetti, and Paolo Santini. On linear equivalence, canonical forms, and digital signatures. *Cryptology ePrint Archive*, 2023.

[9] Stephen A Cook. The complexity of theorem-proving procedures. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pages 143–152. 2023.

[10] Alain Couvreur, Thomas Debris-Alazard, and Philippe Gaborit. On the hardness of code equivalence problems in rank metric. *arXiv preprint arXiv:2011.04611*, 2020.

[11] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[12] Elisa Gorla. Rank-metric codes. In *Concise Encyclopedia of Coding Theory*, pages 227–250. Chapman and Hall/CRC, 2021.

[13] Hanno Lefmann, Kevin T Phelps, and Vojtěch Rödl. Rigid linear binary codes. *Journal of Combinatorial Theory, Series A*, 63(1):110–128, 1993.

[14] Jeffrey Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 1982.

[15] Erez Petrank and Ron M Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, 1997.

[16] Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Hardness estimates of the code equivalence problem in the rank metric. *Cryptology ePrint Archive*, 2022.

[17] Paolo Santini, Marco Baldi, and Franco Chiaraluce. Computational hardness of the permuted kernel and subcode equivalence problems. *IEEE Transactions on Information Theory*, 2023.
[18] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.
[19] Nicolas Sendrier and Dimitrios E Simos. How easy is code equivalence over $\mathbb{F}_q$? In *International Workshop on Coding and Cryptography-WCC 2013*, 2013.
[20] Adi Shamir. *An efficient identification scheme based on permuted kernels.* Springer, 1990.
[21] Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.
[22] Violetta Weger, Niklas Gassner, and Joachim Rosenthal. A survey on code-based cryptography. *arXiv preprint arXiv:2201.07119*, 2022.

Department of Electrical and Computer Engineering, Technical University of Munich, Germany
Email address: violetta.weger@tum.de