# The Support Splitting Algorithm

## Nicolas Sendrier

▶ **To cite this version:**

Nicolas Sendrier. The Support Splitting Algorithm. [Research Report] RR-3637, INRIA. 1999. <inria-00073037>

**HAL Id: inria-00073037**

**https://hal.inria.fr/inria-00073037**

Submitted on 24 May 2006

# *The Support Splitting Algorithm*

Nicolas Sendrier

*R**apport
de recherche*

# The Support Splitting Algorithm

Nicolas Sendrier

**Abstract:** Two linear codes are permutation-equivalent if they are equal up to a fixed permutation of the codewords coordinates. We present here an algorithm able to compute this permutation. We introduce the concept of signature: a property of a position of a code such that the set of all signatures of a given code is globally invariant by permutation. To compute the permutation between two equivalent codes, one needs a signature which is both discriminant and easy to compute. The weight enumerator of the hull of a code provides a signature which has such properties in most cases.

**Key-words:**   code, equivalence, hull, permutation, signature, invariant, weight enumerator

*(Résumé : tsvp)*

# L'algorithme de séparation du support

**Résumé :** Deux codes linéaires sont équivalents par permutation s'ils sont égaux à une permutation près de leurs coordonnées. Nous présentons ici un algorithme capable de calculer cette permutation. Nous introduisons la notion de signature comme une propriété d'une position d'un code telle que l'ensemble des signatures pour un code donné est globalement invariant par permutation. Pour déterminer la permutation entre deux codes équivalents, il suffit de trouver une signature à la fois discriminante et facile à calculer. L'énumérateur des poids du hull d'un code permet de construire une telle signature pour la plupart des codes.

# 1   Introduction

The problem we address here is to know whether or not two given linear codes are permutation-equivalent (*i.e.* one is obtained from the other by permuting the coordinates). If they are, we also want to recover this permutation, which will be unique if and only if the permutation group of the codes is reduced to the identity element.

In a recent paper [PR97], Petrank and Roth have discussed the computational difficulty of this decision problem. They showed that the CODE EQUIVALENCE PROBLEM is not NP-complete but also that it is at least as hard as the GRAPH ISOMORPHISM PROBLEM. Finding a polynomial time algorithm able to make this decision would thus be of interest.

A tentative solution to the decision problem is given by the weight enumerator: two equivalent codes have the same weight enumerator. This criterion is however incomplete. Having the same weight enumerator does not imply that two codes are equivalent, but we can then go further: we can examine the weight enumerators of all the codes obtained by puncturing both of them in one position. If the codes are permutation-equivalent then these two sets will be equal and, furthermore, the matches between the two sets of enumerators may give us some information on the permutation. Unfortunately, computing the weight enumerator of a code becomes computationally intractable when its size grows, from [BMvT78] it is in fact NP-hard. A practical solution for large length will thus make necessary the use of some other invariant property of a code, easier to compute.

First, we will introduce all the definitions necessary for a formal description of the algorithm. An invariant will be a property of a code that does not vary if the code is permuted. From any invariant, we can construct a signature, that is a set of properties, one for each position of the code, which is invariant by permutation of the code support.

If the signatures of the positions of a code are all different, then the permutation between this code and any other one, permutation-equivalent to it, can be recovered by simply matching the sets of signatures of both codes. If the signatures are not all different, an iterative procedure, which is efficient in most cases, will enable us to recover the permutation.

The main difficulty in the implementation of the algorithm lies in the choice of the invariant. The most natural one, the weight enumerator of the code, is very discriminant but rapidly becomes intractable. It is thus necessary to use another invariant: the weight enumerator of the hull. The hull of a linear code [AK90] is equal to its intersection with its dual, and it is generally of small dimension [Sen97b] though not always reduced to $\{0\}$, at least in the binary case. This will provide a signature, not as discriminant as the weight enumerator, but still discriminant enough to be successful in most cases after a few refinements. The main interest of the hull is that, because of its small dimension, it produces invariants that can be computed in polynomial average time and it thus makes the algorithm practical for large length ($> 1000$).

Finally we give some considerations about the implementation in the binary case. In particular, we show that, in the general case, the most expensive part of the algorithm is a gaussian elimination on a square matrix of size equal to the length of the code.

# Notation

- We denote by $\mathbf{F}_q$ the finite field with $q$ elements.

- For any integer $n > 0$, we denote by $I_n$ a set of cardinality $n$ used to index the coordinates of the words of $\mathbf{F}_q^n$. *For instance* $I_n = \{1, \ldots, n\}$.

- For all $x \in \mathbf{F}_q^n$ and all $i \in I_n$, we denote by $x_i$ the $i$-th coordinate of $x$.

- For all $x$ in $\mathbf{F}_q^n$ we denote by $\mathrm{supp}(x) = \{i \in I_n \mid x_i \neq 0\}$ the support of $x$.

- For all $x$ in $\mathbf{F}_q^n$ we denote by $\langle x \rangle$ the vector subspace of $\mathbf{F}_q^n$ spanned by $x$.

- For all subsets $A$ of $\mathbf{F}_q^n$ we set by $\mathrm{supp}(A) = \bigcup_{x \in A} \mathrm{supp}(x)$ the support of $A$.

# 2 Generalities

## 2.1 Equivalence of codes

Two codes $C$ and $C'$ of length $n$ over $\mathbf{F}_q$ are equivalent [MS77, p. 40] if there exist a permutation $\sigma$ on $I_n$ and a sequence $(\pi_i)_{i \in I_n}$ of $n$ permutations on $\mathbf{F}_q$ such that $C' = \phi(C)$ where $\phi : (x_i)_{i \in I_n} \mapsto (\pi_i(x_{\sigma^{-1}(i)}))_{i \in I_n}$.

If $\phi$ maps any linear code into a linear code then each $\pi_i$ is the composition of a scalar multiplication with a field automorphism. The scalar multiple may vary for each coordinate, but the field automorphism must be the same.

**Definition 2.1** *[Ber96] Let $C$ be a code of length $n$ over $\mathbf{F}_q$. For every permutation $\sigma$ on $I_n$, all vectors $\mathbf{a} = (a_i)_{i \in I_n}$ in $(\mathbf{F}_q^*)^n$ and all field automorphisms $\pi$ of $\mathbf{F}_q$, we define*

$$(\mathbf{a}; \sigma, \pi)(C) = \{(\pi(a_{\sigma^{-1}(i)} x_{\sigma^{-1}(i)}))_{i \in I_n} \mid (x_i)_{i \in I_n} \in C\}.$$

*We will say that $C$ and $(\mathbf{a}; \sigma, \pi)(C)$ are* equivalent.

For our purpose, we consider a more restrictive definition.

**Definition 2.2** *Let $C$ be a code of length $n$ over $\mathbf{F}_q$. For all permutations $\sigma$ on $I_n$, we define*

$$\sigma(C) = \{(x_{\sigma^{-1}(i)})_{i \in I_n} \mid (x_i)_{i \in I_n} \in C\}.$$

*We will say that $C$ and $\sigma(C)$ are* permutation-equivalent *or $\sigma$-equivalent, which we denote by $C \sim \sigma(C)$.*

The two definitions coincide in the binary case, otherwise they don't. The set of all permutations on $I_n$, equipped with the composition, forms the symmetric group of $I_n$, denoted by $\mathcal{S}_n$. For any code $C$ of length $n$, we can define a particular subgroup of $\mathcal{S}_n$ called the permutation group of $C$.

**Definition 2.3** *The* permutation group *of a code $C$ of length $n$, denoted by $Perm(C)$, is the subgroup of all the elements $\sigma$ of $\mathcal{S}_n$ such that $\sigma(C) = C$.*

If for every pair $(i,j)$ of elements in $I_n$ there exists $\sigma \in Perm(C)$ such that $\sigma(i) = j$, the permutation group of $C$ is said to be *transitive*. The permutation group always contains the identity permutation. If it doesn't contain any other element we will say that it is *trivial*. If the permutation group of a code $C$ is non-trivial and if $C'$ is permutation-equivalent to $C$, then several permutation will satisfy $C' = \sigma(C)$.

## 2.2 Punctured and shortened codes

For any subset $J$ of $I_n$, we denote by $\mathcal{E}_J$ the words of $\mathbf{F}_q^n$ of support included in $J$. For a singleton $J = \{i\}$ we will simply write $\mathcal{E}_i$.

**Definition 2.4** *Let $C$ be a code of length $n$ and let $J$ be a subset of $I_n$. The code $C$* punctured *in $J$ is defined by*

$$C_J = (C + \mathcal{E}_J) \cap \mathcal{E}_{I_n \setminus J}.$$

*The code $C$* shortened *in $J$ is defined by*

$$C_{\setminus J} = C \cap \mathcal{E}_{I_n \setminus J}.$$

The code $C_J$ consists of all elements of $C$ where the coordinates indexed by $J$ are replaced by zeroes. The code $C_{\setminus J}$ is the subset of all codewords of $C$ whose coordinates indexed by $J$ are equal to zero. If $C$ is linear then both $C_J$ and $C_{\setminus J}$ are linear.

In the usual definition of punctured and shortened codes [MS77, Ch. 1. §9.], the positions indexed by $J$ are removed, producing codes of length $n - |J|$. Our definition will allow the indexing of the coordinates of both $C_J$ and $C_{\setminus J}$ with the same set $I_n$ as for $C$. Furthermore, these three codes are living in the same Hamming space $\mathbf{F}_q^n$, in particular we have $C_{\setminus J} = C_J \cap C$. We list below, without proof, various properties of shortened and punctured codes.

(a) **Commutativity.** Both operations commute, that is, for all codes $C$ of length $n$ and all subsets $J$ and $J'$ of $I_n$ we have $(C_J)_{J'} = C_{J \cup J'}$ and $(C_{\setminus J})_{\setminus J'} = C_{\setminus J \cup J'}$. Furthermore, if $J \cap J' = \emptyset$ then $(C_{\setminus J})_{J'} = (C_{J'})_{\setminus J}$.

(b) **Intersections and sums.** Let $B$ and $C$ be two codes of length $n$ and let $J$ be a subset of $I_n$. We have $(B \cap C)_{\setminus J} = B_{\setminus J} \cap C_{\setminus J} = B_{\setminus J} \cap C$ and $(B + C)_{\setminus J} \supset B_{\setminus J} + C_{\setminus J}$, with equality if $J \cap \operatorname{supp}(B) = \emptyset$, and $(B + C)_J = B_J + C_J$ and $(B \cap C)_J \subset B_J \cap C_J$, with equality if $J \cap \operatorname{supp}(B^\perp) = \emptyset$.

(c) **Equivalence.** For all codes $C$ of length $n$, all subsets $J$ of $I_n$ and all permutations $\sigma$ of $I_n$, we have $\sigma(C_J) = \sigma(C)_{\sigma(J)}$ and $\sigma(C_{\setminus J}) = \sigma(C)_{\setminus \sigma(J)}$.

**(d) Duality.** Puncturing and shortening are dual to each other. For all linear codes $C$ of length $n$ and all subsets $J$ of $I_n$, we have $(C_J)^\perp = C_{\setminus J}^\perp \oplus \mathcal{E}_J$ and $(C_{\setminus J})^\perp = C_J^\perp \oplus \mathcal{E}_J = C^\perp + \mathcal{E}_J$.

*We will set $C_J^\perp = (C^\perp)_J$ and $C_{\setminus J}^\perp = (C^\perp)_{\setminus J}$.*

**(e) Hulls.** (*see Definition 5.1 page 14* ) For all linear codes $C$ of length $n$ and all subsets $J$ of $I_n$, we have $\mathcal{H}(C_J) = \mathcal{H}(C_{\setminus J}^\perp) = C_J \cap C^\perp$ and $\mathcal{H}(C_{\setminus J}) = \mathcal{H}(C_J^\perp) = C_J^\perp \cap C$.

**(f) Dimension.** For all linear codes $C$ of length $n$ and all subsets $J$ of $I_n$, we have $\dim(C_J) = \dim(C) - \dim(C \cap \mathcal{E}_J)$ and $\dim(C_{\setminus J}) = \dim(C) - |J| + \dim(C^\perp \cap \mathcal{E}_J)$.

If $J = \{i\}$ is a singleton, we will set $C_i = C_{\{i\}}$ and $C_{\setminus i} = C_{\setminus \{i\}}$.

# 3   Invariants and signatures

## 3.1   Invariants

Let $\mathcal{L}_n$ denote the set of all codes of length $n$, and let $\mathcal{L} = \bigcup_{n>0} \mathcal{L}_n$ be the set of all codes.

**Definition 3.1** *An* invariant *over a set $E$ is defined to be a mapping $\mathcal{L} \to E$ such that any two permutation-equivalent codes take the same value.*

For instance the length, the cardinality or the minimum Hamming weight are invariants over the integers. The Hamming weight enumerator is an invariant over the polynomials with integer coefficients.

Applying an invariant, for instance the weight enumerator, may help us to decide whether two codes are equivalent or not. Two codes with different weight enumerators cannot be equivalent. Unfortunately we may have inequivalent codes with the same weight enumerator, though this only occurs with a small probability when the codes are chosen randomly.

## 3.2   Signatures

Any invariant is a global property of a code,. We need to define a local property, that is a property of a code and of one of its positions. Such a property is obtained, for instance, by applying an invariant on punctured codes.

### 3.2.1   Definition

**Definition 3.2** *A* signature *$S$ over a set $F$ maps a code $C$ of length $n$ and an element $i$ of $I_n$ into an element of $F$ and is such that for all permutations $\sigma$ on $I_n$, $S(C, i) = S(\sigma(C), \sigma(i))$.*

Signatures do exist. For instance, we can associate to any invariant $\mathcal{V}$ the signature $S_{\mathcal{V}}$ : $(C, i) \mapsto \mathcal{V}(C_i)$.

Now, if we have an invariant $\mathcal{V}$, and wish to answer the question: *"Are $C$ and $C'$ permutation-equivalent?"*, we can go a little further than just compute $\mathcal{V}(C)$ and $\mathcal{V}(C')$. If these codes are permutation-equivalent with $C' = \sigma(C)$ then we have $\mathcal{V}(C_i) = \mathcal{V}(C'_{\sigma(i)})$ for all $i$ in $I_n$. Thus not only $\mathcal{V}(C) = \mathcal{V}(C')$, but also the multisets (*i.e.* a given element can appear several times) $V = \{\mathcal{V}(C_i), i \in I_n\}$ and $V' = \{\mathcal{V}(C'_i), i \in I_n\}$ are equal.

We may now start to solve to the second part of our problem: *"If $C$ and $C'$ are permutation-equivalent, find the permutation between them"*. If for any signature $S$, we have $S(C, i) \neq S(C', j)$ then $\sigma(i) \neq j$. Thus, the image of $i$ by $\sigma$ has to be chosen among the indexes $j$ verifying $S(C', j) = S(C, i)$. The number of distinct values taken by a given signature for the code $C$ is thus of crucial importance to measure how efficient it is. Furthermore, if the mapping $i \mapsto S(C, i)$ takes a different value for all elements of $I_n$ then the permutation between $C$ and any of its permuted versions can be recovered.

### 3.2.2   Partition associated to a signature and a code

For any signature $S$ and any code $C$ of length $n$, if we relate the two indexes $i$ and $j$ if and only if $S(C, i) = S(C, j)$, we define an equivalence relation. Its cosets produce a partition of $I_n$ and this partition will match with any other obtained from a code $C' \sim C$.

**Definition 3.3** *Let $S$ be a signature over $E$. For any $e$ in $E$ we define $J_e = \{i \in I_n \mid S(C, i) = e\}$. We will call $(J_e)_{e \in E}$ the $(C, S)$-partition of $I_n$. For any subset $L$ of $E$, the set $\bigcup_{e \in L} J_e = \{i \in I_n \mid S(C, i) \in L\}$ is called a $(C, S)$-discriminated subset of $I_n$.*

NOTE 1 –   Of course we have $I_n = \cup_{e \in E} J_e$ and $J_e \cap J_{e'} = \emptyset$ if $e \neq e'$. Thus the $J_e$'s form a partition of $I_n$ in the usual sense.

NOTE 2 –   If $E$ is an infinite set then almost all the $J_e$'s are equal to the empty set.

**Proposition 3.4** *Let $C$ be a code of length $n$, let $S$ be a signature over $E$ and let $(J_e)_{e \in E}$ be the $(C, S)$-partition of $I_n$. For every permutation $\sigma$ on $I_n$, $(\sigma(J_e))_{e \in E}$ is the $(\sigma(C), S)$-partition of $I_n$.*

*Proof:* Straightforward from the definition of a signature.                                                           $\diamond$

The $(C, S)$-partition of $I_n$ is the partition obtained by applying the signature $S$ to every positions of the code $C$. A $(C, S)$-discriminated subset of $I_n$ is a set of positions that we are able to recognize on any code permutation-equivalent to $C$. If $C \sim C' = \sigma(C)$, to any $(C, S)$-discriminated subset $J$ of $I_n$ corresponds the $(C', S)$-discriminated subset

$$J' = \{i \in I_n \mid S(C', i) \in S(C, J)\} = \sigma(J),$$

and then we have, in particular $C'_{J'} = \sigma(C_J)$.

### 3.2.3 Discriminant signatures

The thinner the associated partition is, the better the corresponding signature will be. In the worst case, only one element of the partition is non empty (and equal to $I_n$), which means no position can be distinguished from any other. In a favorable case, the partition contains at least two non-empty elements, and ideally, it contains $n$ singletons, which means that we were able to find a distinct property for each position of the code.

**Definition 3.5** *Let $C$ be a code of length $n$.*

- *A signature $S$ is said to be* discriminant *for $C$ if there exist $i$ and $j$ in $I_n$ such that $S(C, i) \neq S(C, j)$.*

- *A signature $S$ is said to be* fully discriminant *for $C$ if for all $i$ and $j$ distinct in $I_n$, $S(C, i) \neq S(C, j)$.*

If $C' = \sigma(C)$ and if $S$ is fully discriminant for $C$, then, for all $i$ in $I_n$, there exists a unique element $j$ in $I_n$ such that $S(C, i) = S(C', j)$, and we have $\sigma(i) = j$. We can thus obtain the permutation $\sigma$.

### Comparing signatures.

**Definition 3.6** *Let $S$ and $T$ be two signatures and let $C$ be a code of length $n$.*

- *We will say that $T$ is* more discriminant than $S$ for $C$, *and denote it by $S \trianglelefteq_C T$, if*

$$\forall i, j \in I_n, \quad T(C, i) = T(C, j) \Rightarrow S(C, i) = S(C, j).$$

  *We will say that $S$ and $T$ are* equivalent *for $C$, and denote it by $S \equiv_C T$, if $S \trianglelefteq_C T$ and $T \trianglelefteq_C S$. We will say that $T$ is* strictly more discriminant than $S$ for $C$, *denoted by $S \triangleleft_C T$, if $S \trianglelefteq_C T$ and $S \not\equiv_C T$.*

- *We will denote by $S \trianglelefteq T$, $S \equiv T$ or $S \triangleleft T$ according to whether $T$ is respectively more discriminant than, equivalent to or strictly more discriminant than $S$ for any code $C$.*

Note that the order $\trianglelefteq_C$ is not total in general, neither is $\trianglelefteq$.

### Constructing new signatures from others.

**Definition 3.7** *Let $S$ and $T$ be two signatures.*

- *The* product *of $S$ and $T$ is defined by $S \times T : (C, i) \mapsto (S(C, i), T(C, i))$.*

- *The* dual *of $S$ is defined by $S^\perp : (C, i) \mapsto S(C^\perp, i)$.*

- *The signature $S$ is said to be* self-dual *if $S \equiv S^\perp$.*

We always have $S \trianglelefteq S \times T$ and the product $S \times S^\perp$ is always self-dual. Most of the signatures we will consider are self-dual, and if not, we can consider their product with their dual signature instead.

**Existence of discriminant signatures.** It is possible to produce a signature, which has a prohibitive algorithmic cost, that will be more discriminant than any other signature, and whose associated partition is as thin as the permutation group allows.

**Proposition 3.8** *Let $\mathcal{M}$ be the signature defined for any code $C$ of length $n$ and any position $i$ in $I_n$ by*

$$\mathcal{M}(C, i) = \{(\sigma(C), \sigma(C_i)) \mid \sigma \in \mathcal{S}_n\}.$$

*The $(C, \mathcal{M})$-partition of $I_n$ is formed by the orbits of the elements of $I_n$ under the action of $Perm(C)$. Moreover, for all signatures $S$, we have $S \trianglelefteq \mathcal{M}$.*

In particular, the result above implies that

- a fully discriminant signature exists for $C$ if and only if the permutation group of $C$ is trivial,

- a discriminant signature exists for $C$ if and only if the permutation group of $C$ is not transitive.

The following technical lemma will be useful to prove the proposition.

**Lemma 3.9** *If there exists $\sigma$ in $Perm(C)$ such that $C_j = \sigma(C_i)$ then there exists $\pi$ in $Perm(C)$ such that $C_j = \pi(C_i)$ and $j = \pi(i)$.*

*Proof:* Let $i$ and $j$ be two elements of $I_n$ and let $\sigma$ be an element of $\mathcal{S}_n$ such that $C = \sigma(C)$ and $C_j = \sigma(C_i)$. We denote $j' = \sigma(i)$ and we have $j' \neq j$ or the problem is over. Let $\tau$ denote the transposition between $j$ and $j'$.

We have $C_j = \sigma(C_i) = \sigma(C)_{\sigma(i)} = C_{j'}$ and thus, in particular $\mathrm{supp}(C) \setminus \{j\} = \mathrm{supp}(C_j) = \mathrm{supp}(C_{j'}) = \mathrm{supp}(C) \setminus \{j'\}$. Since $j \neq j'$, neither $j$ nor $j'$ are in $\mathrm{supp}(C)$ and $\tau \in Perm(C)$. This implies $\pi = \tau \circ \sigma \in Perm(C)$ and $\pi(i) = j$.  $\diamond$

*Proof:* (of Proposition 3.8) Let $i$ and $j$ be two elements of $I_n$. If $i$ and $j$ are in the same orbit under the action of $Perm(C)$ then there exists $\sigma$ in $Perm(C)$ such that $j = \sigma(i)$ and thus $C = \sigma(C)$ and $C_j = \sigma(C)_{\sigma(i)} = \sigma(C_i)$, that is $\mathcal{M}(C, i) = \mathcal{M}(C, j)$.

Reciprocally, if $\mathcal{M}(C, i) = \mathcal{M}(C, j)$ then there exists $\sigma$ in $Perm(C)$ such that $C = \sigma(C)$ and $C_j = \sigma(C_i)$. From Lemma 3.9 we can assume $j = \sigma(i)$ and thus $i$ and $j$ are in the same orbit.

Finally, to prove that $S \trianglelefteq \mathcal{M}$, we have to prove that for all codes $C$ of length $n$ and all $i$ and $j$ in $I_n$ such that $\mathcal{M}(C, i) = \mathcal{M}(C, j)$, we have $S(C, i) = S(C, j)$. We have

$$\mathcal{M}(C, i) = \mathcal{M}(C, j) \Rightarrow (\exists \sigma \in Perm(C) \mid j = \sigma(i))$$
$$\Rightarrow (S(C, i) = S(\sigma(C), \sigma(i)) = S(C, j)).$$

$\diamond$

# 4   The support splitting algorithm: an algorithm to find the permutation between two equivalent codes

Let $C$ and $C'$ be two permutation-equivalent codes of length $n$. If we assume that we know a signature $S$ which is fully discriminant for $C$, then a very simple procedure, that requires the computation of $2n$ signature, will provide us with the permutation between $C$ and $C'$.

> **procedure** permutation
>     **input**: code $C$, $C'$; signature $S$
>         **for** $i, j$ **in** $I_n$ **do**
>             **if** $S(C, i) = S(C', j)$ **then**
>                 $\sigma[i] \leftarrow j$
>         **return**$(\sigma)$

Of course, a fully discriminant signature will not necessarily be available, and even if it is, it may be too expensive to compute. If a signature $S$ is discriminant but not fully discriminant for the code $C$, then we cannot obtain the permutation. However, the $(C, S)$-partition of $I_n$ gives us a classification of all elements of $I_n$ according to their signatures. We present in the next section a means to "refine" a discriminant signature, and its associated partition, to produce (hopefully) a fully discriminant one. As discrimination is relative to a code, the refinements will also depend on each particular code.

## 4.1   Refining a discriminant signature

Let $S$ be a signature over $E$ which is not fully discriminant for $C$, after computing all the $S(C, i)$ we can partition $I_n$ into $J_1, \ldots, J_s$ (with $s < n$). If $C'$ is $\sigma$-equivalent to $C$ we will obtain a partition $J'_1, \ldots, J'_s$ such that $J'_1 = \sigma(J_1), \ldots J'_s = \sigma(J_s)$. If $J$ is equal to any of the $J_l$'s, or to any union of them, $J' = \sigma(J)$ can be likewise obtained from the $J'_l$'s. The codes $C_J$ and $C'_{J'}$ are also $\sigma$-equivalent and thus, by computing all the $S(C_J, i)$ and all the $S(C'_{J'}, i)$, we may obtain additional information on $\sigma$. More formally, we have:

**Proposition 4.1** *Let $T$ be a signature over $E$. Define for any subset $L$ of $E$, for all codes $C$ of length $n$ and all elements $i$ of $I_n$ the mapping $S_{T,L}$ by*

$$S_{T,L}(C, i) = S(C_{K_{T,L}(C)}, i) \tag{1}$$

*where $K_{T,L}(C) = \{i \in I_n \mid T(C, i) \in L\}$, is a signature.*

*Proof:* We put $K = K_{T,L}(C)$, from Proposition 3.4 we have $K_{T,L}(\sigma(C)) = \sigma(K)$ for any permutation $\sigma$. From $\sigma(C_K) = \sigma(C)_{\sigma(K)}$, and the definition of a signature, we have:

$$
\begin{aligned}
S_{T,L}(\sigma(C), \sigma(i)) &= S(\sigma(C)_{\sigma(K)}, \sigma(i)) \\
&= S(\sigma(C_K), \sigma(i)) \\
&= S(C_K, i) = S_{T,L}(C, i).
\end{aligned}
$$

This proves that $S_{T,L}$ is a signature. $\diamond$

If two code codes $C$ and $C'$ are $\sigma$-equivalent, then for any $(C, T)$-discriminated subset $J = K_{T,L}(C)$ we have $J' = \sigma(J) = K_{T,L}(C')$, and the two codes $C_J$ and $C'_{J'}$ are $\sigma$-equivalent. We can thus try to apply the signature $T$ on them to obtain more information on $\sigma$. That's exactly what $S_{T,L}$ does.

Hopefully the product signature $T \times S_{T,L}$ will be strictly more discriminant than $T$ for $C$. Note that even if $S$ is self-dual, it may not be the case for $S_{T,L}$, and it may be useful to consider $\bar{S}_{T,L} = S_{T,L} \times S_{T,L}^{\perp}$. Let $S$ be a self-dual signature over $E$, we try to construct a sequence $T_i$ of signatures with

> **procedure** fd_signature
>     **input**: code $C$; signature $S$
>         $i \leftarrow 0$ ; $T_0 \leftarrow S$
>         **while** $|T_i(C, I_n)| < n$ **do**
>             $L \subset_R T_i(C, I_n)$
>             $T_{i+1} \leftarrow T_i \times \bar{S}_{T_i, L}$
>             $i \leftarrow i + 1$
>         **return**$(T_i)$

Here the inclusion $L \subset_R T_i(C, I_n)$ means that the set $L$ is chosen at random. In practice, making a random choice is not bad, since most refinements are successful, but the use of singletons for $L$ is computationally more efficient. Moreover, some other heuristic may be involved in the choice of $L$ (see section 5.3.1).

The algorithm is efficient when the instances are chosen at random, however, we did not manage to prove anything on the termination or on the complexity, even if we restrict ourselves to codes with a trivial permutation group.

At last, remark that the existence of a fully discriminant signature $S$ for $C$ and the equality $\{S(C, i), i \in I_n\} = \{S(C', i), i \in I_n\}$ will allow us to obtain a unique possible permutation $\sigma$ between $C$ and $C'$, but doesn't necessarily imply that $C' = \sigma(C)$. This last equality still has to be checked, though it seems very difficult to build a pair of inequivalent codes taking the same set of values for a fully discriminant signature.

## 4.2 Examples

### 4.2.1 Two small examples with non-linear codes

We present in this section two small example with non linear codes. Though the support splitting algorithm is designed for (large) linear codes, they are good illustration of the basic concept.

**A fully discriminant signature.** We consider the equivalent codes

$$C = \{1110, 0111, 1010\} \text{ and } C' = \{0011, 1011, 1101\},$$

and as invariant we will take the weight distribution denoted by $\mathcal{W}$. We obtain for $C$:

$$\left\{ \begin{array}{llll} C_1 = \{0110, 0111, 0010\} & \to & \mathcal{W}(C_1) = X + X^2 + X^3 \\ C_2 = \{1010, 0011\} & \to & \mathcal{W}(C_2) = 2X^2 \\ C_3 = \{1100, 0101, 1000\} & \to & \mathcal{W}(C_3) = X + 2X^2 \\ C_4 = \{1110, 0110, 1010\} & \to & \mathcal{W}(C_4) = 2X^2 + X^3 \end{array} \right.$$

and for $C'$

$$\left\{ \begin{array}{llll} C'_1 = \{0011, 0101\} & \to & \mathcal{W}(C'_1) = 2X^2 \\ C'_2 = \{0011, 1011, 1001\} & \to & \mathcal{W}(C'_2) = 2X^2 + X^3 \\ C'_3 = \{0001, 1001, 1101\} & \to & \mathcal{W}(C'_3) = X + X^2 + X^3 \\ C'_4 = \{0010, 1010, 1100\} & \to & \mathcal{W}(C'_4) = X + 2X^2 \end{array} \right.$$

From this we immediately obtain the permutation $\sigma$, such that $C' = \sigma(C)$:

$$\sigma(1) = 3, \ \sigma(2) = 1, \ \sigma(3) = 4, \ 1\sigma(4) = 2.$$

**Example of a refined signature.**   We consider the codes

$$C = \{01101, 01011, 01110, 10101, 11110\},$$

and

$$C' = \{10101, 00111, 10011, 11100, 11011\}.$$

We get for $C$,

$$\left\{ \begin{array}{llll} C_1 = \{01101, 01011, 01110, 00101\} & \to & X^2 + 3X^3 \\ C_2 = \{00101, 00011, 00110, 10101, 10110\} & \to & 3X^2 + 2X^3 \\ C_3 = \{01001, 01011, 01010, 10001, 11010\} & \to & 3X^2 + 2X^3 \\ C_4 = \{01101, 01001, 01100, 10101, 11100\} & \to & 2X^2 + 3X^3 \\ C_5 = \{01100, 01010, 01110, 10100, 11110\} & \to & 3X^2 + X^3 + X^4 \end{array} \right.$$

Positions 2 and 3 cannot be discriminated, but

$$\left\{ \begin{array}{llll} C_{\{1,2\}} = \{00101, 00011, 00110\} & \to & 3X^2 \\ C_{\{1,3\}} = \{01001, 01011, 01010, 00001\} & \to & X + 2X^2 + X^3 \end{array} \right.$$

Now for $C'$,

$$\left\{ \begin{array}{llll} C'_1 = \{00101, 00111, 00011, 01100, 01011\} & \to & 3X^2 + 2X^3 \\ C'_2 = \{10101, 00111, 10011, 10100\} & \to & X^2 + 3X^3 \\ C'_3 = \{10001, 00011, 10011, 11000, 11011\} & \to & 3X^2 + X^3 + X^4 \\ C'_4 = \{10101, 00101, 10001, 11100, 11001\} & \to & 2X^2 + 3X^3 \\ C'_5 = \{10100, 00110, 10010, 11100, 11010\} & \to & 3X^2 + 2X^3 \end{array} \right.$$

at that point positions 1 and 5 cannot be discriminated, but we know that $\sigma(\{2,3\}) = \{1,5\}$, $\sigma(1) = 2$, $\sigma(4) = 4$ and $\sigma(5) = 3$. If $C$ and $C'$ are equivalent, then $C_1$ and $C'_2$ are also equivalent. We thus compute

$$\begin{cases} C'_{\{2,1\}} = \{00101, 00111, 00011, 00100\} & \rightarrow & X + 2X^2 + X^3 \\ C'_{\{2,5\}} = \{10100, 00110, 10010\} & \rightarrow & 3X^2 \end{cases}$$

which gives us $\sigma(2) = 5$ and $\sigma(3) = 1$.

### 4.2.2 Another example with a linear code

We consider the $[20, 10]$ binary linear code of generating matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

The weight distributions of every singly punctured codes are:

$$\begin{cases} \mathcal{W}(C_1) & = & (1, 0, 0, 2, 5, 20, 54, 108, 143, 168, 196, 154, 87, 52, 22, 8, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_2) & = & (1, 0, 0, 2, 4, 21, 57, 105, 141, 170, 194, 156, 90, 49, 21, 9, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_3) & = & (1, 0, 1, 1, 5, 22, 49, 103, 151, 180, 187, 143, 95, 54, 19, 9, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_4) & = & (1, 0, 1, 1, 3, 24, 53, 97, 153, 184, 179, 147, 97, 48, 23, 11, 2, 0, 0, 0, 0) \\ \mathcal{W}(C_5) & = & (1, 0, 1, 1, 4, 24, 51, 97, 152, 184, 183, 147, 96, 48, 21, 11, 3, 0, 0, 0, 0) \\ \mathcal{W}(C_6) & = & (1, 0, 0, 2, 8, 17, 49, 110, 147, 175, 186, 150, 98, 47, 21, 10, 2, 1, 0, 0, 0) \\ \mathcal{W}(C_7) & = & (1, 0, 0, 2, 4, 21, 57, 99, 147, 186, 178, 144, 102, 49, 21, 11, 2, 0, 0, 0, 0) \\ \mathcal{W}(C_8) & = & (1, 0, 0, 3, 5, 20, 50, 102, 157, 173, 180, 159, 91, 46, 26, 8, 2, 1, 0, 0, 0) \\ \mathcal{W}(C_9) & = & (1, 0, 1, 2, 4, 18, 53, 110, 144, 172, 195, 150, 88, 50, 23, 10, 3, 0, 0, 0, 0) \\ \mathcal{W}(C_{10}) & = & (1, 0, 0, 3, 5, 21, 54, 96, 143, 186, 196, 147, 87, 49, 22, 10, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_{11}) & = & (1, 0, 0, 2, 5, 20, 54, 108, 143, 168, 196, 154, 87, 52, 22, 8, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_{12}) & = & (1, 0, 1, 2, 3, 20, 53, 108, 147, 168, 195, 154, 85, 52, 23, 8, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_{13}) & = & (1, 0, 0, 3, 6, 17, 54, 107, 140, 175, 196, 153, 90, 47, 22, 9, 3, 1, 0, 0, 0) \\ \mathcal{W}(C_{14}) & = & (1, 0, 0, 2, 6, 20, 54, 103, 140, 181, 196, 144, 90, 54, 22, 7, 3, 1, 0, 0, 0) \\ \mathcal{W}(C_{15}) & = & (1, 0, 0, 2, 5, 23, 51, 100, 154, 171, 182, 162, 93, 45, 23, 8, 3, 1, 0, 0, 0) \\ \mathcal{W}(C_{16}) & = & (1, 0, 0, 3, 3, 21, 57, 96, 150, 186, 178, 147, 99, 49, 21, 10, 3, 0, 0, 0, 0) \\ \mathcal{W}(C_{17}) & = & (1, 0, 1, 1, 7, 21, 45, 103, 153, 183, 187, 143, 93, 51, 23, 9, 2, 1, 0, 0, 0) \\ \mathcal{W}(C_{18}) & = & (1, 0, 0, 3, 4, 21, 55, 96, 149, 186, 182, 147, 98, 49, 19, 10, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_{19}) & = & (1, 0, 0, 2, 7, 21, 52, 99, 139, 186, 200, 144, 89, 49, 20, 11, 4, 0, 0, 0, 0) \\ \mathcal{W}(C_{20}) & = & (1, 0, 0, 3, 4, 19, 55, 104, 147, 174, 190, 155, 86, 47, 27, 10, 2, 0, 0, 0, 0) \end{cases}$$

All positions can be discriminated except positions 1 and 11. The corresponding partition of $I_{20}$ is

$$I_{20} = \left( \bigcup_{i \notin \{1,11\}} \{i\} \right) \cup \{1, 11\}$$

It is thus necessary to refine this signature. We will start with $L = \{\mathcal{W}(C_{13})\}$ (the largest for lexicographic order). We obtain

$$\left\{ \begin{array}{lcl} \mathcal{W}(C_{\{1,13\}}) & = & (1, 0, 0, 4, 9, 30, 79, 126, 160, 196, 182, 120, 69, 30, 11, 6, 1, 0, 0, 0, 0) \\ \mathcal{W}(C_{\{11,13\}}) & = & (1, 0, 0, 5, 8, 26, 84, 131, 150, 196, 192, 115, 64, 34, 12, 5, 1, 0, 0, 0, 0) \end{array} \right.$$

We put $W = \mathcal{W}(C_{13})$ and for any code $B$ of length $n$, we denote $J(B) = \{i \in I_n \mid \mathcal{W}(B_i) = W\}$. The signature $S$ defined for any binary code $B$ by

$$S(B, i) = \big( \mathcal{W}(B_i), \mathcal{W}(B_{J(B) \cup \{i\}}) \big)$$

is fully discriminant for $C$.

At last, remark that because of Lemma 5.8 (§5.4.2) and of the MacWilliams transform the signature $i \mapsto \mathcal{W}(C_i)$ is self-dual, and thus no improvement is obtained by considering $\mathcal{W}(C_{\backslash i})$ or $\mathcal{W}(C^{\perp}{}_i)$.

# 5   Finding good signatures

A signature can be easily built from an invariant. This signature must be discriminant, which excludes all the easy invariants such as the length or the dimension. Other invariants, as the minimum weight, or even better, the weight distribution may be discriminant [Sen94], but when the size of the code increases their computation becomes intractable.

## 5.1   Hull of a linear code

The hull was introduced in 1990 by Assmus and Key [AK90].

**Definition 5.1** *The* hull *of a linear code $C$ is defined to be its intersection with its dual. We will denote $\mathcal{H}(C) = C \cap C^{\perp}$.*

If the hull of a vector space over a field of characteristic zero is always reduced to $\{0\}$, it is, by far, not always the case for finite characteristic. For instance self-dual or weakly self-dual codes are equal to their hull.

The hull of a permuted code is obtained by applying the *same permutation* to the hull. This is of great importance, because it means that any invariant applied to the hull is still an invariant.

**Proposition 5.2** *Let $C$ be a linear code of length $n$ and $\sigma$ be a permutation on $I_n$. We have $\mathcal{H}(\sigma(C)) = \sigma(\mathcal{H}(C))$.*

*Proof:* We simply have to remark that $\sigma(C^\perp) = \sigma(C)^\perp$ and $\sigma(A \cap B) = \sigma(A) \cap \sigma(B)$. $\diamond$

**Corollary 5.3** *For any invariant $\mathcal{V}$, the mapping $C \mapsto \mathcal{V}(\mathcal{H}(C))$ is an invariant.*

NOTE – No such property exists in general for equivalent codes, the two codes $\mathcal{H}(C)$ and $\mathcal{H}((\mathbf{a}; \sigma, \pi)(C))$ are not always equivalent. The exceptions are $q = 2$ and $q = 3$. In fact, we have $(\mathbf{a}; \sigma, \pi)(C)^\perp = (\mathbf{a}^{-1}; \sigma^{-1}, \pi^{-1})(C^\perp)$ where $\mathbf{a}^{-1}$ is the component-wise inverse of $\mathbf{a}$. In the binary case equivalence and permutation-equivalence coincide, and in the ternary case, there is no non-trivial field automorphism (thus $\pi = 1$, as for any prime field) and all the non-zero field elements are their own inverse.

Thus we have at our disposal any invariant applied to the hull. The practical interest of Corollary 5.3 is due to the fact that the dimension of the hull is on average a small positive constant.

**Proposition 5.4** *[Sen97b] The average dimension of the hull of a $q$-ary $[n, k]$ code tends to a constant when the size of the code goes to infinity, this constant is equal to*

$$\bar{R} = \sum_{i>0} \frac{1}{q^i + 1}$$

*The proportion of $q$-ary $[n, k]$ code with a hull of dimension $l > 0$ is asymptotically equal to*

$$R_l = \frac{R_{l-1}}{q^l - 1} \ \ and \ R_0 = \prod_{i \geq 1} \frac{1}{1 + q^{-i}}.$$

This result is proved by counting the number codes with a given hull dimension from the number of weakly self-dual codes and by use of the generating series:

$$\mathcal{R}(z) = \sum_{l \geq 0} R_l z^l = R_0 \prod_{i \geq 1} \left(1 + zq^{-i}\right). \tag{2}$$

We give in Table 1 a part of the dimension spectrum of the hull of $q$-ary linear codes for some values of $q$.

The hull of a random $q$-ary code is of small dimension with a very high probability, and it is not always reduced to $\{0\}$. The signature associated to an invariant of the hull will thus be easy to compute, with a great probability, and may provide a non trivial partition of the support.

| $q$ | $R_0$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $\bar{R}$ |
|---|---|---|---|---|---|---|
| 2 | 0.4194 | 0.4194 | 0.1398 | 0.0200 | $1.3\,10^{-3}$ | 0.7645 |
| 3 | 0.6390 | 0.3195 | 0.0399 | $1.5\,10^{-3}$ | $1.9\,10^{-5}$ | 0.4041 |
| 4 | 0.7375 | 0.2458 | 0.0164 | $2.6\,10^{-4}$ | $1.0\,10^{-6}$ | 0.2794 |
| 16 | 0.9373 | 0.0628 | $2.5\,10^{-4}$ | $6.0\,10^{-8}$ | $9.1\,10^{-13}$ | 0.0630 |
| 256 | 0.9961 | $3.9\,10^{-3}$ | $6.0\,10^{-8}$ | $3.6\,10^{-15}$ | $8.2\,10^{-25}$ | $3.9\,10^{-3}$ |

Table 1: Hull of $q$-ary linear codes: proportion with a given dimension and average dimension

## 5.2   Building a signature from the hull

Let $\mathcal{W}(C)$ denote the weight enumerator of $C$, we consider the signature

$$S : (C, i) \mapsto \left( \mathcal{W}(\mathcal{H}(C_i)), \mathcal{W}(\mathcal{H}(C_i^\perp)) \right).$$

Compared with the weight enumerator, this signature is much easier to compute. On the other hand, since the hull has a small dimension the signature is much less discriminant.

### 5.2.1   Example: a linear code of length 20

Let $C$ be the $(20, 10)$ binary linear code used in section 4.2.2 of generating matrix

$$G = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}$$

This code has a hull equal to $\{0\}$. The weight enumerators of the hulls of all singly punctured codes are given in Table 2 and provide the first partition

$$I_{20} = \{7\} \cup \{5, 17\} \cup \{9, 12, 13\} \cup \{3, 4, 16, 18\} \cup \{1, 2, 11, 20\} \cup \{6, 8, 15\} \cup \{10, 14, 19\}.$$

At that point, we consider the codes $C_7^\perp$ and $C_7$ and repeat the process. For the code $C' = \sigma(C)$ the position corresponding to 7 can be identified; it is the only position $i_0$ such that $\mathcal{W}(\mathcal{H}(C'^\perp_{i_0})) = 1 + X^{14}$. If no or if several such positions exist then $C'$ cannot be permutation-equivalent to $C$. The result of the first refinement is given in Table 3. The partition becomes

$$I_{20} = \bigcup_{i \notin \{1,4,9,12,18,20\}} \{i\} \cup \{9, 12\} \cup \{4, 18\} \cup \{1, 20\}.$$

Three other refinements will be necessary before we obtain a fully discriminant signature. As for position 7 for the first refinement, the position used each time can be identified by an invariant property. By construction, for any code permutation-equivalent to $C$, one and only one position will have this property. For instance the position 17 used in the third refinement is the only one such that $\mathcal{W}(\mathcal{H}(C_{17})) = 1 + X^6$ and $\mathcal{W}(\mathcal{H}(C^{\perp}_{\{7,17\}})) = 1$.

Note that the second refinement does not produce any improvement of the partition. After the last refinement, we are able to produce a fully discriminant signature. Let $S$ denote the signature $(B, i) \mapsto \big(\mathcal{W}(\mathcal{H}(B_i)), \mathcal{W}(\mathcal{H}(B^{\perp}_i))\big)$. For any linear code $B$ of length 20, we define the sets:

$$J_1(B) = \big\{ i \in I_{20} \mid S(B, i) = \big(1 + X^{14}, 1\big) \big\}$$
$$J_2(B) = \big\{ i \in I_{20} \setminus J_1(B) \mid S(B, i) = \big(1 + X^6, 1\big), S(B^{\perp}_{J_1(B)}, i) = \big(1, 1 + X^6\big) \big\}$$
$$J_3(B) = \big\{ i \in I_{20} \setminus J_1(B) \mid S(B, i) = \big(1, 1 + X^8\big), S(B^{\perp}_{J_1(B)}, i) = \big(1, 1 + X^8\big) \big\}$$

and for $l = 1, 2, 3$, let $S_l : (B, i) \mapsto \Big( S(B_{J_l(B)}, i), S(B^{\perp}_{J_l(B)}, i) \Big)$. The signature $S \times S_1 \times S_2 \times S_3$ is fully discriminant for $C$. For instance $J_1(C) = \{7\}$, $J_2(C) = \{17\}$ and $J_3(C) = \{15\}$.

In practice, using the weight enumerator of the hull instead of the weight enumerator of the code will increase the number of refinement needed but will dramatically reduce the cost of each iteration.

In the case of the hull, the average cost is polynomial, we have to compute at most $n$ hulls, each of them is obtained in polynomial time, and their weight enumerators, each obtained in constant average time. Though we were not able to prove anything on the number of refinement needed, it is in practice logarithmic in $n$. Thus by using the hull, we obtain a complexity which is practically polynomial in $n$.

With the weight enumerators alone (section 4.2.2), we have to compute weight enumerators of $(n, k)$ codes, thus in practice the complexity is exponential in $k$.

In section 6 the implementation aspects are discussed. We will see that computing the hull of a code can be achieved by a Gaussian elimination and that computing the hull of a

| $i$ | $\mathcal{W}(\mathcal{H}(C_i))$ | $\mathcal{W}(\mathcal{H}(C^{\perp}_i))$ |
|---|---|---|
| 7 | $1 + X^{14}$ | 1 |
| 5, 17 | $1 + X^6$ | 1 |
| 6, 8, 15 | 1 | $1 + X^8$ |
| 10, 14, 19 | 1 | $1 + X^{10}$ |
| 9, 12, 13 | $1 + X^8$ | 1 |
| 3, 4, 16, 18 | $1 + X^{10}$ | 1 |
| 1, 2, 11, 20 | $1 + X^{12}$ | 1 |

Table 2: Initial step

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{7,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_7)_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{7,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_7^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 5 | 1 | 1 | $1+X^{12}$ | 1 |
| 17 | $1+X^6+X^{12}+X^{14}$ | $1+X^{14}$ | 1 | $1+X^6$ |
| 15 | 1 | 1 | 1 | $1+X^8$ |
| 6 | 1 | 1 | 1 | $1+X^{12}$ |
| 8 | $1+X^{14}$ | $1+X^8+X^{10}+X^{14}$ | $1+X^8$ | 1 |
| 14 | 1 | 1 | 1 | $1+X^{10}$ |
| 19 | 1 | 1 | 1 | $1+X^{14}$ |
| 10 | $1+X^{14}$ | $1+X^8+X^{10}+X^{14}$ | $1+X^{10}$ | 1 |
| 13 | 1 | 1 | $1+X^6$ | 1 |
| 9, 12 | 1 | 1 | $1+X^{10}$ | 1 |
| 16 | 1 | 1 | $1+X^8$ | 1 |
| 3 | $1+X^8+X^{10}+X^{14}$ | $1+X^{14}$ | 1 | $1+X^{10}$ |
| 4, 18 | 1 | 1 | $1+X^{12}$ | 1 |
| 2 | 1 | 1 | $1+X^{10}$ | 1 |
| 11 | $1+X^6+X^{12}+X^{14}$ | $1+X^{14}$ | 1 | $1+X^{12}$ |
| 1, 20 | 1 | 1 | $1+X^6$ | 1 |

Table 3: First refinement

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{5,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_5)_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{5,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_5^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 9, 12 | 1 | $1+X^8$ | $1+2X^6+X^8$ | $1+X^6$ |
| 4, 18 | 1 | $1+X^{10}$ | $1+X^6+X^8+X^{10}$ | $1+X^6$ |
| 1, 20 | $1+X^{10}$ | 1 | 1 | 1 |

Table 4: Second refinement

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{17,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_{17})_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{17,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_{17}^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 12 | 1 | $1+X^8$ | $1+2X^6+X^8$ | $1+X^6$ |
| 9 | 1 | $1+X^8$ | $1+X^6+X^8+X^{10}$ | $1+X^6$ |
| 4, 18 | $1+X^8$ | 1 | 1 | 1 |
| 1, 20 | 1 | $1+X^{12}$ | $1+X^6+X^{12}+X^{14}$ | $1+X^6$ |

Table 5: Third refinement

punctured code whose hull is known can be achieved at an extremely low cost. For random codes, the whole support splitting algorithm will not be much more expensive than a single Gaussian elimination.

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{15,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_{15})_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{15,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_{15}^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 18 | 1 | $1+X^8$ | 1 | 1 |
| 4 | $1+X^{10}$ | 1 | $1+X^8$ | $1+X^8+2X^{10}$ |
| 20 | 1 | $1+X^6$ | 1 | 1 |
| 1 | 1 | $1+X^{10}$ | 1 | 1 |

Table 6: Fourth refinement

### 5.2.2 Another linear example

We present here the construction of a fully discriminant signature for a code with a hull of dimension one. The generating matrix is the following:

$$
G = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0
\end{pmatrix}
$$

All the computations for this example are presented in Table 7. At each refinement, the choice of the discriminated subset is made, as for the previous example, according to predetermined sorting rules; the cosets of the partition are sorted in increasing size, then in increasing order in the previous partition, then in increasing lexicographic order for each enumerator. This defines a total ordering at each step. There is an additional heuristic rule when the hull has a strictly positive dimension, this rule explains the choice of position 6 for the last refinement and is explained in section 5.3.1.

## 5.3  Hull of punctured codes – Application to the binary case

The signatures constructed from the hull are discriminant in practice. This empirical fact was not obvious *a priori*. From Proposition 5.4 we know that the weight enumerator of the hull is, on average, easy to compute. However, we have no guaranty on how discriminant this signature is. For two different positions $i$ and $j$, the weight enumerators of $\mathcal{H}(C_i)$ and $\mathcal{H}(C_j)$ are (often) easy to compute, but how often are they inequivalent?

We will denote by $e^{(i)}$ the vector of $\mathbf{F}_q^n$ of support $\{i\}$ such that $e_i^{(i)} = 1$, and $\mathcal{E}_i = \langle e^{(i)} \rangle$ the vector space spanned by $e^{(i)}$.

**Proposition 5.5** *Let $C$ be a linear code of length $n$. For all $i$ in $I_n$ we have*

| $i$ | $\mathcal{W}(\mathcal{H}(C_i))$ | $\mathcal{W}(\mathcal{H}(C_i^\perp))$ |
|---|---|---|
| 12 | $1 + X^{10}$ | $1 + X^8 + 2X^{10}$ |
| 10 | $1 + X^{10}$ | $1 + 2X^{10} + X^{12}$ |
| 16 | $1 + X^{10}$ | $1 + 2X^{10} + X^{16}$ |
| 7 | $1 + X^8 + 2X^{10}$ | $1 + X^{10}$ |
| 1 | $1 + X^6 + X^{10} + X^{12}$ | $1 + X^{10}$ |
| 8 | $1 + X^{10} + X^{12} + X^{14}$ | $1 + X^{10}$ |
| 4 | $1 + 2X^{10} + X^{16}$ | $1 + X^{10}$ |
| $3, 17, 19$ | $1 + 2X^{10} + X^{12}$ | $1 + X^{10}$ |
| $2, 5, 6, 9, 11, 13, 14, 15, 18, 20$ | $1$ | $1$ |

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{12,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_{12})_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{12,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_{12}^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 17 | $1 + X^{10}$ | $1 + X^{10}$ | $1 + X^{10}$ | $1 + X^8 + 2X^{10}$ |
| 19 | $1 + X^8 + 2X^{10}$ | $1 + 2X^8 + 4X^{10} + X^{12}$ | $1 + 2X^{10} + X^{12}$ | $1 + X^{10}$ |
| 3 | $1 + X^8 + 2X^{10}$ | $1 + X^6 + X^8 + 3X^{10} + 2X^{12}$ | $1 + 2X^{10} + X^{12}$ | $1 + X^{10}$ |
| $2, 6, 9, 13$ | $1 + X^{10}$ | $1 + X^{10}$ | $1$ | $1$ |
| $5, 11, 14, 15, 18, 20$ | $1 + X^8$ | $1 + X^8$ | $1$ | $1$ |

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{10,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_{10})_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{10,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_{10}^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 6 | $1 + X^{12}$ | $1 + X^{12}$ | $1$ | $1$ |
| $2, 9, 13$ | $1 + X^{10}$ | $1 + X^{10}$ | $1$ | $1$ |
| $11, 15, 18$ | $1 + X^{10}$ | $1 + X^{10}$ | $1$ | $1$ |
| $5, 14, 20$ | $1 + X^{12}$ | $1 + X^{12}$ | $1$ | $1$ |

| $i$ | $\mathcal{W}(\mathcal{H}(C_{\{6,i\}}))$ | $\mathcal{W}(\mathcal{H}((C_6)_i^\perp))$ | $\mathcal{W}(\mathcal{H}(C_{\{6,i\}}^\perp))$ | $\mathcal{W}(\mathcal{H}((C_6^\perp)_i^\perp))$ |
|---|---|---|---|---|
| 2 | $1$ | $1 + X^8$ | $1$ | $1 + X^{10}$ |
| 13 | $1 + X^8$ | $1$ | $1 + X^8$ | $1$ |
| 9 | $1 + X^{12}$ | $1$ | $1 + X^8$ | $1$ |
| 15 | $1$ | $1 + X^{10}$ | $1$ | $1 + X^8$ |
| 11 | $1 + X^8$ | $1$ | $1 + X^8$ | $1$ |
| 18 | $1 + X^{12}$ | $1$ | $1 + X^{12}$ | $1$ |
| 20 | $1 + X^6$ | $1$ | $1 + X^6$ | $1$ |
| 5 | $1 + X^8$ | $1$ | $1 + X^8$ | $1$ |
| 14 | $1 + X^{10}$ | $1$ | $1 + X^6$ | $1$ |

Table 7: Second example with a hull of dimension one

**(i)** $\mathcal{E}_i \subset C \Leftrightarrow i \notin \operatorname{supp}(C^\perp)$

**(ii)** $\mathcal{E}_i \subset C + C^\perp \Leftrightarrow i \notin \operatorname{supp}(\mathcal{H}(C))$

*Proof:* Let $C$ be a linear code and let $i$ be an element of $I_n$. We have

$$e^{(i)} \in C^\perp \Leftrightarrow (\forall x \in C, \ x \cdot e^{(i)} = 0) \Leftrightarrow (\forall x \in C, \ x_i = 0) \Leftrightarrow i \notin \operatorname{supp}(C)$$

Since $\mathcal{H}(C)^\perp = C + C^\perp$, **(ii)** is obtained by applying **(i)** to $\mathcal{H}(C)$. $\diamond$

### 5.3.1 Binary case

We assume in this section that $q = 2$. Each time $i \notin \operatorname{supp}(\mathcal{H}(C))$, from Proposition 5.5, we have $e^{(i)} \in C + C^\perp$. Thus there exists $x^{(i)}$ in $C$ and $y^{(i)}$ in $C^\perp$ such that $x^{(i)} = y^{(i)} + e^{(i)}$. These elements are not unique, but any other pair $(x', y') \in C \times C^\perp$ verifying $x' = y' + e^{(i)}$ will be such that $x' - x^{(i)} = y' - y^{(i)} \in \mathcal{H}(C)$ and thus, since $i \notin \operatorname{supp}(\mathcal{H}(C))$, we have $(y'_i, x'_i) = (y_i^{(i)}, x_i^{(i)})$, $\mathcal{H}(C) \oplus \langle x^{(i)} \rangle = \mathcal{H}(C) \oplus \langle x' \rangle$ and $\mathcal{H}(C) \oplus \langle y^{(i)} \rangle = \mathcal{H}(C) \oplus \langle y' \rangle$. In the non degenerated case, *i.e.* $\operatorname{supp}(C) = \operatorname{supp}(C^\perp) = I_n$, there are three different cases:

**A** - if $i \in \operatorname{supp}(\mathcal{H}(C))$ then $\mathcal{H}(C_i) = \mathcal{H}(C_i^\perp) = \mathcal{H}(C)_{\backslash i}$.

**B** - if $i \notin \operatorname{supp}(\mathcal{H}(C))$ and $(y_i^{(i)}, x_i^{(i)}) = (0, 1)$ then $\mathcal{H}(C_i) = \mathcal{H}(C) \oplus \langle y^{(i)} \rangle$ and $\mathcal{H}(C_i^\perp) = \mathcal{H}(C)$.

**C** - if $i \notin \operatorname{supp}(\mathcal{H}(C))$ and $(y_i^{(i)}, x_i^{(i)}) = (1, 0)$ then $\mathcal{H}(C_i) = \mathcal{H}(C)$ and $\mathcal{H}(C_i^\perp) = \mathcal{H}(C) \oplus \langle x^{(i)} \rangle$.

Details and proofs can be found in section 5.4.

From what is stated above, we can compute $\mathcal{H}(C_i)$ and $\mathcal{H}(C_i^\perp)$ from $\mathcal{H}(C)$, and, much more important, we have some informations on how the hull of the different punctured codes vary.

**If the hull has dimension zero.** We have $\operatorname{supp}(\mathcal{H}(C)) = \emptyset$ and thus for all $i$ in $I_n$ there exists a unique pair $(x^{(i)}, y^{(i)})$ in $C \times C^\perp$ such that $x^{(i)} - y^{(i)} = e^{(i)}$.

- If $x_i^{(i)} = 0$ then $\mathcal{H}(C_i^\perp) = \{0, x^{(i)}\}$ and thus $\mathcal{W}(\mathcal{H}(C_i^\perp)) = 1 + X^{w_H(x^{(i)})}$.

- If $x_i^{(i)} = 1$ then $\mathcal{H}(C_i) = \{0, y^{(i)}\}$ and thus $\mathcal{W}(\mathcal{H}(C_i)) = 1 + X^{w_H(y^{(i)})}$.

If the code has no particular structure, then the Hamming weights $w_H(x^{(i)})$ (or $w_H(y^{(i)})$) is likely to vary when $i$ varies. Thus the first step will provide us with a relatively thin partition of $I_n$.

**If the hull has dimension one.** The hull $\mathcal{H}(C)$ contains one non-zero codeword, thus the average cardinality of $\operatorname{supp}(\mathcal{H}(C))$ is $n/2$. For all $i$ in $\operatorname{supp}(\mathcal{H}(C))$, we have $\mathcal{H}(C_i) = \mathcal{H}(C_i^\perp) = \{0\}$, and these positions cannot be discriminated at all in the first step. However this discrimination will be possible for the other positions, that is for about half of $I_n$.

**If the hull has dimension** $h > 1$**.** In that case $\text{supp}(\mathcal{H}(C))$ will be larger (its average cardinality will be $n - n/2^h$)), and thus the small number of positions in $I_n \setminus \text{supp}(\mathcal{H}(C))$ (possibly none) may not provide a thin enough partition. However in this case $\mathcal{H}(C_i) = \mathcal{H}(C_i^\perp)$ is obtained by shortening $\mathcal{H}(C)$ in one position. The number of distinct shortened subcodes of $\mathcal{H}(C)$ is upper-bounded by $\min(n, 2^h - 1)$. If the code has no particular structure the actual number of distinct shortened codes is likely to be close to this bound. We will thus also obtain in this case a partition thin enough to be successfully refined.

**Heuristic.** The most favorable case seems to be $\mathcal{H}(C) = \{0\}$, this was confirmed experimentally. A good strategy when $h > 0$ will then consist in searching a set $J$ of discriminated positions, such that $\mathcal{H}(C_J) = \{0\}$. This strategy proved to be efficient in practice, for random codes of length 1000 and dimension 500, the number of refinement needed before we obtain a fully discriminant signature varies roughly from 3 to 12, with an average of 7.

## 5.4 The $q$-ary case

In the binary case, the positions can be divided into the three subsets described by items **A**, **B** and **C** of §5.3.1. More generally, in the $q$-ary case, the positions can be naturally divided into $q + 1$ subsets which have a natural correspondence with the projective line $\mathbf{P}^1(\mathbf{F}_q)$.

### 5.4.1 Multiplying one coordinate by a constant

The projective line $\mathbf{P}^1(\mathbf{F}_q)$ is the set of all one-dimensional subspaces of the $\mathbf{F}_q$-vector space $\mathbf{F}_q^2$. A point of $\mathbf{P}^1(\mathbf{F}_q)$ is a vector space $\langle(\beta, \gamma)\rangle$ with $(\beta, \gamma) \neq (0, 0)$. The $q + 1$ elements of $\mathbf{P}^1(\mathbf{F}_q)$ can be represented by $\mathbf{F}_q \cup \{\infty\}$ with $\langle(\beta, 1)\rangle = \beta$ and $\langle(1, 0)\rangle = \infty$. We denote $\langle(\beta, \gamma)\rangle^{-1} = \langle(\gamma, \beta)\rangle$, this notation coincide with usual inverse, and we have $\infty^{-1} = 0$. For all $\beta$ in $\mathbf{P}^1(\mathbf{F}_q)$, we define the linear code

$$C_{\beta \cdot i} = \{y \in \mathbf{F}_q^n \mid \exists x \in C, \; x - y \in \mathcal{E}_i, \; (y_i, x_i) \in \beta\}.$$

We have $C_{\infty \cdot i} = C_{\setminus i} \oplus \mathcal{E}_i$ and for all $\beta \neq \infty$, the code $C_{\beta \cdot i}$ is obtained by multiplying the $i$-th coordinate of all elements of $C$ by $\beta$. In particular, we have $C_{1 \cdot i} = C$ and $C_{0 \cdot i} = C_i$. Note that for all $\beta$ we have $C_{\setminus i} \subset C_{\beta \cdot i}$ and $\dim(C_{\beta \cdot i}) - \dim(C_{\setminus i}) \leq 1$.

We will denote by $\mathcal{H}_{\beta \cdot i}(C)$ the intersection of $C_{\beta \cdot i}$ with $C^\perp$, that is:

$$\mathcal{H}_{\beta \cdot i}(C) = C_{\beta \cdot i} \cap C^\perp = \{y \in C^\perp \mid \exists x \in C, \; x - y \in \mathcal{E}_i, \; (y_i, x_i) \in \beta\}.$$

We have $\mathcal{H}(C)_{\setminus i} = C_{\setminus i} \cap C^\perp \subset \mathcal{H}_{\beta \cdot i}(C)$ and the difference of dimension is at most one.

If $i \notin \text{supp}(\mathcal{H}(C))$ then from Proposition 5.5 there exists $x$ in $C$ and $y$ in $C^\perp$ such that $x - y = e^{(i)}$. These elements are unique modulo $\mathcal{H}(C)$ and thus the pair $(y_i, x_i)$ constituted by their $i$-th coordinates is unique. For all $\beta$ in $\mathbf{P}^1(\mathbf{F}_q)$ we consider the following subsets of $I_n$

$$A_\beta(C) = \{i \in I_n \mid \exists (x, y) \in C \times C^\perp, \; x - y \in \mathcal{E}_i, \; \langle(y_i, x_i)\rangle = \beta\}.$$

As we will see below, the sets $A_\beta(C)$ form a partition of $I_n$. Furthermore, for all $i$, one and only one of the $\mathcal{H}_{\beta \cdot i}(C)$ is strictly greater than $\mathcal{H}(C)_{\backslash i}$, and it is precisely the one such that $i \in A_\beta(C)$.

**Proposition 5.6** *Let $C$ be a linear code of length $n$ such that* $\mathrm{supp}(C^\perp) = \mathrm{supp}(C) = I_n$. *For all $i$ in $I_n$ and all $\beta$ and $\gamma$ in $\mathbf{P}^1(\mathbf{F}_q)$ we have*

  **(i)** $(i \in A_\beta(C)) \Leftrightarrow \left( \mathcal{H}(C)_{\backslash i} \subsetneq \mathcal{H}_{\beta \cdot i}(C) \right)$

  **(ii)** *If $\beta \neq \gamma$ then $A_\beta(C) \cap A_\gamma(C) = \emptyset$.*

*Proof:*  **(i)** If $i \in A_\beta(C)$ then there exists $y$ in $C^\perp$ and $x$ in $C$ such that $x - y \in \mathcal{E}_i$ and $\langle (y_i, x_i) \rangle = \beta$. Since $i \in \mathrm{supp}(C^\perp)$, we have $\mathcal{E}_i \not\subset C$ and $y \notin C$ (Proposition 5.5). Thus in particular $y \notin \mathcal{H}(C)_{\backslash i}$. Moreover, by definition, we have $y \in \mathcal{H}_{\beta \cdot i}(C)$ and thus $\mathcal{H}(C)_{\backslash i} \neq \mathcal{H}_{\beta \cdot i}(C)$.

Reciprocally, let's assume $\mathcal{H}(C)_{\backslash i} \neq \mathcal{H}_{\beta \cdot i}(C)$. Let $y \in \mathcal{H}_{\beta \cdot i}(C) \setminus \mathcal{H}(C)_{\backslash i}$, by definition, there exists $x$ in $C$ such that $x - y \in \mathcal{E}_i$ and $(y_i, x_i) \in \beta$. We cannot have $(y_i, x_i) = (0, 0)$ because else we would have $y \in C_{\backslash i}$, thus $\langle (y_i, x_i) \rangle = \beta$. This exactly means that $i \in A_\beta(C)$.

**(ii)** We will show that if the intersection of $A_\beta(C)$ and $A_\gamma(C)$ is not empty then $\beta = \gamma$.

Let $i \in A_\beta(C) \cap A_\gamma(C)$. Let $(x, y)$ and $(x', y')$ in $C \times C^\perp$ such that $\langle (y_i, x_i) \rangle = \beta$ and $\langle (y'_i, x'_i) \rangle = \gamma$. In particular we have $(y_i, x_i) \neq (0, 0)$ and $(y'_i, x'_i) \neq (0, 0)$.

- If $i \in \mathrm{supp}(\mathcal{H}(C))$. If we had $y_i \neq x_i$ or $y'_i \neq x'_i$, we would have $x - y \in \mathcal{E}_i \setminus \{0\}$ and thus $e^{(i)} \in C + C^\perp$ which is incompatible with $i \in \mathrm{supp}(\mathcal{H}(C))$ (Proposition 5.5). We necessarily have $y_i = x_i$ and $y'_i = x'_i$ and thus $\beta = \gamma = 1$.

- If $i \notin \mathrm{supp}(\mathcal{H}(C))$. We have $x \neq y$ or else $x = y \in \mathcal{H}(C)$ and $(y_i, x_i) \neq (0, 0)$ or else $i \in \mathrm{supp}(\mathcal{H}(C))$. For the same reason $x' \neq y'$ and by a proper constant multiplication, we can assume that $x - y = e^{(i)}$ and $x' - y' = e^{(i)}$. This implies $x - x' = y - y' \in \mathcal{H}(C)$, thus $(y_i, x_i) = (y'_i, x'_i)$ and $\beta = \gamma$.

$\diamond$

**Corollary 5.7** *Let $C$ be a linear code of length $n$ over $\mathbf{F}_q$ such that* $\mathrm{supp}(C^\perp) = \mathrm{supp}(C) = I_n$.

- $A_1(C) = \mathrm{supp}(\mathcal{H}(C))$.

- *For all $\beta \neq 1$ in $\mathbf{P}^1(\mathbf{F}_q)$ and all $i$ in $A_\beta(C)$ we have*

$$\mathcal{H}_{\beta \cdot i}(C) = C^\perp \cap (\mathcal{E}_i + C).$$

*Proof:*  • We have $i \in A_1(C)$ if and only if $\mathcal{H}(C)_{\backslash i} \subsetneq \mathcal{H}_{1 \cdot i}(C) = \mathcal{H}(C)$. And for any code $U$ we have $U_{\backslash i} \neq U$ if and only if $i \in \mathrm{supp}(U)$. Thus $A_1(C) = \mathrm{supp}(\mathcal{H}(C))$.

- For all $\beta$ and all $i$, we can obtain from the definitions that:

$$\mathcal{H}(C)_{\backslash i} \subset \mathcal{H}(C) \subset \mathcal{H}_{\beta \cdot i}(C) \subset C^{\perp} \cap (\mathcal{E}_i + C) . \qquad (3)$$

If $\beta \neq 1$ and $i \in A_{\beta}(C)$ then, from the first item of the statement, we have $i \notin \mathrm{supp}(\mathcal{H}(C))$ and $\mathcal{H}(C)_{\backslash i} = \mathcal{H}(C)$. Furthermore, the difference of dimension between $\mathcal{H}(C) = C \cap C^{\perp}$ and the rightmost term of (3) is at most one and thus, at most one of the three inclusions is not an equality. From Proposition 5.6 we have $\mathcal{H}(C) \neq \mathcal{H}_{\beta \cdot i}(C)$ and thus $\mathcal{H}_{\beta \cdot i}(C) = C^{\perp} \cap (\mathcal{E}_i + C)$.

$\diamondsuit$

This result produces a means to compute $\mathcal{H}_{\beta \cdot i}(C)$ from $\mathcal{H}(C)$.

- If $i \in A_1(C) = \mathrm{supp}(\mathcal{H}(C))$ then we have $\mathcal{H}(C)_{\backslash i} \subsetneq \mathcal{H}_{1 \cdot i}(C) = \mathcal{H}(C)$ and for all $\gamma \neq 1$, we have $\mathcal{H}_{\gamma \cdot i}(C) = \mathcal{H}(C)_{\backslash i}$.

- If $i \in A_{\beta}(C)$ with $\beta \neq 1$, then $i \notin \mathrm{supp}(\mathcal{H}(C))$ and $\mathcal{H}_{\beta \cdot i}(C) = \mathcal{H}(C) \oplus \langle y \rangle$, where $y$ is an element of $\left( C^{\perp} \cap (\mathcal{E}_i + C) \right) \setminus \mathcal{H}(C)$. We will see in §6 that such a word $y$ is easy to compute.

**Final remark.** In most results of this section, we have assumed that both the supports of $C$ and $C^{\perp}$ were both equal to $I_n$. Of course, this is not always true, but positions that are not in the support of $C$ or $C^{\perp}$ can be easily identified and removed (or ignored). Thus our assumption does not imply any loss of generality.

### 5.4.2 A new discriminant signature

For any code $C$, we denote by $\mathcal{W}(C) = \sum_{x \in C} X^{w_H(x)}$ its weight enumerator where $w_H()$ is the Hamming weight.

**Lemma 5.8** *For any linear code $C$ and any $i \in \mathrm{supp}(C^{\perp})$, we have*

$$X\mathcal{W}(C_i) = \mathcal{W}(C) + (X - 1)\mathcal{W}(C_{\backslash i})$$

*Proof:* The set $L = C \setminus C_{\backslash i}$ contains the codewords of $C$ whose $i$-th coordinate is not zero and thus the weight enumerator of $L$ punctured in $i$ is $\mathcal{W}(L_i) = \mathcal{W}(L)/X$. We have $C = C_{\backslash i} \cup L$ and $C_i = C_{\backslash i} \cup L_i$, and, because $i \in \mathrm{supp}(C^{\perp})$, both these unions are disjoint. Thus

$$\mathcal{W}(C) = \mathcal{W}(C_{\backslash i}) + \mathcal{W}(L) \text{ and } \mathcal{W}(C_i) = \mathcal{W}(C_{\backslash i}) + \mathcal{W}(L_i) = \mathcal{W}(C_{\backslash i}) + \frac{\mathcal{W}(L)}{X}$$

We obtain the result by elimination of $\mathcal{W}(L)$.

$\diamondsuit$

**Proposition 5.9** *The mapping* $B : (C, i) \mapsto \beta$ *where* $i \in A_\beta(C)$ *is a signature, and the mapping*

$$S_\mathcal{W} : (C, i) \mapsto \begin{cases} (\beta, \mathcal{W}(\mathcal{H}_{\beta \cdot i}(C))) & \text{if } \beta = B(C, i) \neq 1 \\ (1, \mathcal{W}(\mathcal{H}(C)_{\setminus i})) & \text{if } B(C, i) = 1 \end{cases} \tag{4}$$

*is a self-dual signature for all $C$ such that* $\mathrm{supp}(C^\perp) = \mathrm{supp}(C) = I_n$.

*Proof:* First note that, since $A_\beta(C) = A_{\beta^{-1}}(C^\perp)$, we have $B(C^\perp, i) = B(C, i)^{-1}$.

Let $i$ and $j$ be two elements of $I_n$ such that $\beta = B(C, i) = B(C, j)$.

- If $\beta = 1$ then $S_\mathcal{W}(C, i) = S_\mathcal{W}(C^\perp, i) = \left(1, \mathcal{W}(\mathcal{H}(C)_{\setminus i}\right)$ and the result is straightforward.

- If $\beta \neq 1$ then we want to prove that for all $i$ and $j$ in $I_n$ such that $B(C, i) = B(C, j) = \beta$, we have

$$\mathcal{W}(\mathcal{H}_{\beta \cdot i}(C)) = \mathcal{W}(\mathcal{H}_{\beta \cdot j}(C)) \Rightarrow \mathcal{W}(\mathcal{H}_{\beta^{-1} \cdot i}(C^\perp)) = \mathcal{W}(\mathcal{H}_{\beta^{-1} \cdot j}(C^\perp)) \tag{5}$$

  We have $\mathcal{H}_{\beta \cdot i}(C) = \mathcal{H}(C) \oplus \langle y \rangle$ where $y \in C^\perp \cap (\mathcal{E}_i + C)$. By definition of $\mathcal{H}_{\beta \cdot i}(C)$ there exists $x$ in $C$ such that $x - y \in \mathcal{E}_i$. We have $x \in C \cap (\mathcal{E}_i + C^\perp)$, and $x \notin \mathcal{H}(C)$ because $i \in \mathrm{supp}(C)$. Thus $\mathcal{H}_{\beta^{-1} \cdot i}(C^\perp) = \mathcal{H}(C) \oplus \langle x \rangle$.

  - If $\beta \notin \{0, \infty\}$ then $\mathcal{H}(C) \oplus \langle y \rangle$ is obtained by multiplying the $i$-th coordinate of $\mathcal{H}(C) \oplus \langle x \rangle$ by $\beta$. Thus $\mathcal{W}(\mathcal{H}_{\beta \cdot i}(C)) = \mathcal{W}(\mathcal{H}_{\beta^{-1} \cdot i}(C^\perp))$. The same equality also holds for $j$ and we get (5).

  - If $\beta = 0$, we have $y_i = 0$ and $x_i \neq 0$. Thus $\mathcal{H}(C) \oplus \langle y \rangle$ is obtained by puncturing $\mathcal{H}(C) \oplus \langle x \rangle$ in $i$. Shortening $\mathcal{H}(C) \oplus \langle x \rangle$ in $i$ will produce $\mathcal{H}(C)$ and from Lemma 5.8 we get

    $$X\mathcal{W}(\mathcal{H}_{\beta \cdot i}(C)) = \mathcal{W}(\mathcal{H}_{\beta^{-1} \cdot i}(C^\perp)) + (X - 1)\mathcal{W}(\mathcal{H}(C)),$$

    and similarly for $j$

    $$X\mathcal{W}(\mathcal{H}_{\beta \cdot j}(C)) = \mathcal{W}(\mathcal{H}_{\beta^{-1} \cdot j}(C^\perp)) + (X - 1)\mathcal{W}(\mathcal{H}(C))$$

    which is enough to prove (5).

  - For $\beta = \infty$, we just interchange $C$ and $C^\perp$.

$\diamond$

In a practical point of view, the signature that will be computed is the following

$$S_\mathcal{W} : (C, i) \mapsto \begin{cases} \left(\dfrac{y_i}{x_i}, \mathcal{W}(\mathcal{H}(C) \oplus \langle y \rangle)\right) & \text{if } \exists (x, y) \in C \times C^\perp \text{ such that } x - y = e^{(i)} \\ \left(1, \mathcal{W}(\mathcal{H}(C)_{\setminus i})\right) & \text{else} \end{cases}$$

The second case corresponds to $i \in \mathrm{supp}(\mathcal{H}(C))$ and the first to $i \notin \mathrm{supp}(\mathcal{H}(C))$, in that case $x$ and $y$ can be efficiently computed (see section 6). This signature can be made more discriminant if instead of the weight enumerator we consider the complete weight enumerator $\overline{\mathcal{W}}(C) = \sum_{x \in C} \prod_{i \in I_n} X_{x_i}$. We did not investigate this possibility in detail since our main concern was the binary case.

# 6    Implementation

Our algorithm requires the computation of hulls, and the computation of weight distributions. It is an important fact that we have to make these computations on codes that are close to each other : we only consider codes that are obtained by puncturing or shortening one of the two codes given as input. The fact is of no help for the weight distributions, but, as we will see, it is very helpful for computing the hulls.

## 6.1    Computing hulls

Since $\mathcal{H}(C) = (C + C^\perp)^\perp$, the hull of a code $C$ of length $n$ and dimension $k$ is equal to the nullspace of the $n \times n$ matrix whose first $k$ rows are of a basis of $C$ and whose last $n - k$ rows are a basis of $C^\perp$. This matrix is obtained by stacking a generating matrix and a parity check matrix of $C$. Let $G = (Id, R)$ be a generating matrix of $C$ in echelon form, then $H = ({}^t R, -Id)$ is a parity check matrix of $C$. The gaussian elimination of the matrix

$$M = \left( \begin{array}{c|c} Id & R \\ \hline {}^t R & -Id \end{array} \right) \tag{6}$$

will provide a generating matrix of $\mathcal{H}(C)$.

### 6.1.1    Diagonal gaussian elimination

For any $n \times n$ square matrix $M$ over $\mathbf{F}_q$, a diagonal standard form $M$ is an $n \times n$ square matrix $D$ such that the non-zero columns of $Id - D$ form a basis of the nullspace of $M$.

For instance, $D$ is a diagonal standard form of $D'$

$$
D' = \begin{pmatrix}
1 & & & \mathrm{x} & & \mathrm{x} & \\
 & 1 & & \mathrm{x} & & \mathrm{x} & \\
 & & 1 & \mathrm{x} & & \mathrm{x} & \\
 & & & 1 & \mathrm{x} & & \\
 & & & & 1 & & \\
 & & & & & & \\
 & & & & & &
\end{pmatrix}, \quad
D = \begin{pmatrix}
1 & & & \mathrm{x} & & \mathrm{x} & \\
 & 1 & & \mathrm{x} & & \mathrm{x} & \\
 & & 1 & \mathrm{x} & & \mathrm{x} & \\
 & & & & & & \\
 & & & & 1 & \mathrm{x} & \\
 & & & & & & \\
 & & & & & & 1
\end{pmatrix}
$$

The matrix $D'$ above is an echelon matrix, this form is obtained after a gaussian elimination. We will call diagonal gaussian elimination the procedure to obtain a diagonal standard matrix. Informally, the diagonal standard form is obtained by a gaussian elimination and by keeping the '1's on the diagonal, leaving zero lines if necessary.

The procedure we need in practice does a little more, it takes as input two $n \times n$ square matrices $M$ and $U$, with $U$ non-singular and returns two $n \times n$ square matrices $D$ and $S$, where $D$ is a diagonal standard form of $M$ and $S$ is a non-singular matrix such that $U^{-1} M = S^{-1} D$. We denote $(D, S) = \mathrm{DG}(M, U)$. Note that if $U = Id$, then $SM = D$.

This procedure is almost identical to the classical gaussian elimination and requires a number of operations in $\mathbf{F}_q$ proportional to $n^3$. Furthermore, the matrix we have to deal with has the particular form of equation (6). The diagonal gaussian elimination of this matrix can be achieved with a much lower complexity than in the general case.

**Proposition 6.1** *The diagonal standard form of the matrix $M$ defined by*

$$M = \left( \begin{array}{c|c} Id & R \\ \hline {}^tR & -Id \end{array} \right)$$

*is equal to*

$$D = \left( \begin{array}{c|c} Id & R - RE \\ \hline 0 & E \end{array} \right)$$

*where $E$ is the diagonal standard form of $X = I + {}^tRR$. Furthermore, if $U$ is a non singular matrix such that $UX = E$ then*

$$S = \left( \begin{array}{c|c} Id - RU{}^tR & RU \\ \hline U{}^tR & -U \end{array} \right)$$

*is a non singular matrix which verifies $SM = D$.*

To obtain the matrices $S$ and $D$ defined in the above statement, we thus have to compute a diagonal gaussian elimination on an $(n-k) \times (n-k)$ matrix $X$ and to perform 4 matrix multiplications on matrices of about half the size. We assume that the average cost of the multiplication of a $a \times b$ matrix by a $b \times c$ matrix is $(1 - 1/q)abc$ field operations, and that the average cost of the diagonal gaussian elimination procedure on $X$ is $2(1 - 1/q)(n-k)^3$ field operations.

If $k > n/2$ and if we are given a generating matrix of the code in systematic form, the cost for computing $S$ and $D$ is then always less than one half of the cost for computing the elimination of an $n \times n$ matrix in the general case. If $k < n/2$ then we can consider the dual of $C$ instead of $C$ at no cost. If we have to compute a systematic generating matrix first, we have an additional cost of $(1 - 1/q)k^2n$ and the total cost ratio is still less than one half.

These figures can be improved by using a faster algorithm for matrix multiplication.

### 6.1.2  Computing hulls of codes modified in one position

The algorithm is basically iterative, however the first step is, by far, the most expensive. First because once a position has been discriminated, that is appeared as a singleton in one of the partition, no other computation with it is necessary. Second, because computing the hull of a punctured code $C_J$ is easier if the hull of $C$ has already been computed.

**First step.** We are given a code $C$ and we denote $G = (Id \mid R)$ and $H = ({}^tR \mid -Id)$, respectively a generating and a parity check matrix of $C$. Let $M$ be the matrix defined as in (6) by stacking $G$ and $H$. Note that, due to particular form of $G$ and $H$, the matrix $M$ is symmetric. We have assumed here, without loss of generality, that the first $k$ positions of $C$ allow the echelon form of its generating matrix $G$. Let $D$ be a diagonal standard form of $M$ and let $S$ be the non-singular matrix such that $SM = D$. We will denote respectively by $S_i$ and $D_i$ the $i$-th row of $S$ and $D$. From $D$ we derive a generating matrix of $\mathcal{H}(C)$ denoted by $B$, this matrix has $n$ columns and $h$ rows, where $h$ denotes the dimension of $\mathcal{H}(C)$.

- If $i \in \operatorname{supp}(\mathcal{H}(C)) = A_1(C)$ then we need to compute a generating matrix of $\mathcal{H}(C)_{\backslash i}$, which can be obtained from $B$.

- If $i \notin \operatorname{supp}(\mathcal{H}(C))$, then we have to find $x$ in $C$ and $y$ in $C^\perp$ such that $x - y = e^{(i)}$. We will then obtain $\beta = \langle (y_i, x_i) \rangle$ and $\mathcal{H}_{\beta \cdot i}(C) = \mathcal{H}(C) \oplus \langle y \rangle$. For all $i$ we have $S_i M = (S_i {}^tG, S_i {}^tH) = e^{(i)}$, and:

  - if $i \leq k$ then $S_i {}^tH = 0$ and $(S_i - e^{(i)}) {}^tG = 0$, thus $S_i \in C$ and $S_i - e^{(i)} \in C^\perp$. We put $x = S_i$ and $y = S_i - e^{(i)}$.
  - if $i > k$ then $S_i {}^tG = 0$ and $(S_i + e^{(i)}) {}^tH = 0$, thus $S_i \in C^\perp$ and $S_i + e^{(i)} \in C$. We put $x = S_i + e^{(i)}$ and $y = S_i$.

**Example.** We consider a binary linear $(12, 6, 3)$ code. The matrix $M$ is obtain by stacking a generating matrix of the form $(Id|R)$ and the parity check matrix $({}^tR|Id)$

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We applied to $M$ a diagonal gaussian elimination to obtain $S$ and $D$, with $D$ in diagonal standard form, such that $SM = D$. We easily deduce a generating matrix

$B$ of the hull by taking the non-zero columns of $Id - D$.

$$S = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The support of the hull is given by the non-zero columns of $B$. Note that these positions are exactly the indexes of the rows of weight 1 of $D$. This was expected, since the rows of $D$ generates the space $C + C^\perp$ and $e^{(i)}$ is in this space if and only if $i$ is not in the support of the hull.

We index the positions with the set $I_{12} = \{1, \dots, 12\}$ with the lowest indexes on the left.

- We have $\operatorname{supp}(\mathcal{H}(C)) = \{1, 3, 4, 6, 7, 8, 10, 12\}$, and
  - if $i \in \{3, 4, 8, 12\}$ then $\mathcal{H}(C)_{\backslash i} = \langle (1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0) \rangle$. It produces a weight distribution of $1 + X^4$.

- if $i \in \{6, 10\}$ then $\mathcal{H}(C)_{\backslash i} = \langle (1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1) \rangle$. It produces a weight distribution of $1 + X^6$.
- if $i \in \{1, 7\}$ then $\mathcal{H}(C)_{\backslash i} = \langle (0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1) \rangle$. It also produces a weight distribution of $1 + X^6$.

- We have $I_{12} \setminus \mathrm{supp}(\mathcal{H}(C)) = \{2, 5, 9, 11\}$, and the vector $y$ we are looking for is equal to the $i$-th row of $S$ where the $i$-th coordinate is replaced by a '1'.
  - if $i = 2$ then $x_2 = S_{2,2} = 0$ thus $y_2 = 1$ and $\beta = \langle (y_2, x_2) \rangle = \langle (1, 0) \rangle = \infty$. The code $\mathcal{H}_{\infty \cdot 2}(C)$ has a generating matrix equal to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

  It produces a weight distribution of $1 + X^4 + X^5 + 2X^6 + 2X^7 + X^9$.
  - if $i = 5$ then $x_5 = S_{5,5} = 0$ thus $y_5 = 1$ and $\beta = \langle (y_5, x_5) \rangle = \langle (1, 0) \rangle = \infty$. The code $\mathcal{H}_{\infty \cdot 5}(C)$ has a generating matrix equal to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

  It produces a weight distribution of $1 + X^4 + X^5 + 2X^6 + 2X^7 + X^9$.
  - if $i = 9$ then $y_9 = S_{9,9} = 0$ thus $x_9 = 1$ and $\beta = \langle (y_9, x_9) \rangle = \langle (0, 1) \rangle = 0$. The code $\mathcal{H}_{0 \cdot 9}(C)$ has a generating matrix equal to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

  It produces a weight distribution of $1 + X^4 + 2X^5 + 2X^6 + 2X^7$.
  - if $i = 11$ then $y_{11} = S_{11,11} = 0$ thus $x_{11} = 1$ and $\beta = \langle (y_{11}, x_{11}) \rangle = \langle (0, 1) \rangle = 0$. The code $\mathcal{H}_{0 \cdot 11}(C)$ has a generating matrix equal to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

  It produces a weight distribution of $1 + X^4 + 2X^5 + 2X^6 + 2X^7$.

At that point, the partition can be build from the weight distributions and is equal to:

$$I_{12} = \{3, 4, 8, 12\} \cup \{1, 6, 7, 10\} \cup \{2, 5\} \cup \{9, 11\}.$$

Note that the code of this example has a non trivial permutation group, in addition to the identity, it contains the permutation $(1\ 6)(7\ 10)$, product of two transpositions. The support splitting algorithm produces the orbits of the automorphism group, that is 8 singletons in addition to $\{1, 6, 7, 10\}$ in 11 refinements among which only 4 are effective.

**Refinements.** Let $J$ be a subset of $I_n$. Let $G_J$ denote the matrix obtained by replacing in $G$ the columns indexed by $J$ by zeroes, this produces a generating matrix of $C_J$. Let's consider the matrix

$$M' = \left( \frac{G_J}{H} \right) = \left( \, {}^tG_{J_1} \mid {}^tH_{J_2} \, \right)$$

The rows of $M'$ generate the space $C_J + C^\perp$ and, if $D'$ denotes its diagonal standard form, the space generated by the columns of $Id - D'$ is thus equal to $V = (C_J + C^\perp)^\perp = (C_J)^\perp \cap C$, and from we have $V_J = \mathcal{H}(C_J)$.

The matrices $M$ and $M'$ differ on a small number of columns, this will greatly simplify the computation of a diagonal standard form of $M'$ if a diagonal standard form of $M$ is known.

Let $(D, S) = \mathrm{DG}(M, Id)$, $X = M - M'$ and $(D', S') = \mathrm{DG}(D + SX, S)$. By definition of the diagonal gaussian elimination, we have $D = SM$ and $S^{-1}(D + SX) = S'^{-1}D'$, which implies that $S'M' = D'$ and thus $D'$ is a diagonal standard form of $M'$. Note that this does not implies that $(D', S') = \mathrm{DG}(M', Id)$, but for our purpose, any diagonal standard form of $M'$ is good enough.

An important computational advantage is obtained if we compute $\mathrm{DG}(D + SX, S)$ instead of $\mathrm{DG}(M', Id)$. The matrix $SX$ only has $|J|$ non-zero columns, and thus the matrix $D + SX$ differs from the identity on at most $|J| + h$ columns. In practice this reduces the cost of the elimination to $(|J| + h)O(n^2)$ (including the product $SX$) instead of $O(n^3)$ where $h$ is the dimension of the hull.

As above, we will denote respectively by $S'_i$ and $D'_i$ the $i$-th row of $S'$ and $D'$. We denote by $B'$ the matrix whose rows are equal to to the non-zero columns of $Id - D'$, it is a generating matrix of the space $C_J + C^\perp$. Note that $B'$ is not a generating matrix of $\mathcal{H}(C_J)$, but such a matrix is easily obtained by putting to zero in $B'$ the columns indexed by $J$.

- If $i \in \mathrm{supp}(\mathcal{H}(C_J))$ then we have to compute $\mathcal{H}(C_J)_{\setminus i}$ which can be obtained from $B'$.

- If $i \notin \mathrm{supp}(\mathcal{H}(C_J))$ then, as for the "normal" case, we will obtain $x$ in $C_J$ and $y$ in $C^\perp$ such that $x - y = D'_i = e^{(i)}$. It can be easily proved that $y \in C^\perp_{\setminus J} \subset (C_J)^\perp$. Thus we get $\beta = \langle(y_i, y_i + 1)\rangle$ and $\mathcal{H}_{\beta \cdot i}(C) = \mathcal{H}(C_J) \oplus \langle y \rangle$.

For $C_J^\perp$, we just have to exchange the roles of $G$ and $H$; the computation will be exactly the same with the matrix

$$M'' = \left( \frac{G}{H_J} \right)$$

where $H_J$ is obtained from $H$ by replacing by zeroes the columns indexed by $J$. The cost analysis is identical to the one for the first step.

## 6.2   Computing weight distribution

The cost in for computing the weight distribution of $q$-ary a linear code of length $n$ and dimension $h$ is proportional to $nq^h$ operations in $\mathbf{F}_q$. For random instances, the average number of operations is proportional to $n\sum_{l\geq 0}R_lq^l = n\mathcal{R}(q) = 2n\mathcal{R}(1) = 2n$ (see (2) in §5.1). In practice, for random instances, the cost of the weight distribution computation is negligible.

## 6.3   Computation time

The whole procedure has been implemented in language C for the binary case. We ran the program on 100 000 random instance of binary codes of length 1000 and dimension 500 and the average running time for recovering one permutation was a little less than half a second on a DEC Alpha PW500. The average number of refinements was 6.5, ranging from 3 to 15.

# 7   Conclusion

It is possible to recover in "practical" polynomial time the permutation between two permutation-equivalent codes when the considered codes have a trivial permutation group and a hull of small size. Any random instance will verify such assumptions.

For codes that have a non trivial permutation group, the support splitting algorithm will provide the orbits of the positions under action of the permutation group. By puncturing the codes in one or more positions, depending on the transitivity, this may provide a tool to compute the permutation group by using the techniques introduced by Leon [Leo82]. Research are in progress in this direction.

In both cases, recovering the permutation or the orbits, we did not manage to prove anything precise on the complexity, mainly because the number of refinement needed before we obtain the most discriminant signature is difficult to bound. In fact, we have no way, apart from heuristic rules, to determine *a priori* if a given refinement will be efficient, that is if it will produce a thinner partition of the support, and there is an exponential number of possible refinements ...

Implementation proved to be very efficient in the binary case, each time the hull's dimension was small enough. The algorithm is still unefficient if the hull dimension is large, but from the reduction given in [PR97] of the CODE EQUIVALENCE PROBLEM to the GRAPH ISOMORPHISM PROBLEM, the existence of difficult instances had to be expected.

# References

[AK90]     E.F. Assmus, Jr and J.D. Key. Affine and projective planes. *Discrete Mathematics*, 83:161–187, 1990.

[Ber96]    T. Berger. *Groupes d'automorphismes et de permutations des codes affine-invariants*. Habilitation à diriger des recherches, Université de Limoges, January 1996.

[BMvT78]   E.R. Berlekamp, R.J. McEliece, and H.C. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3), May 1978.

[Leo82]    J.S. Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, May 1982.

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

[PR97]     E. Petrank and R.M. Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, September 1997.

[Sen94]    N. Sendrier. On the structure of a randomly permuted concatenated code. In P. Charpin, editor, *Livre des résumé – EUROCODE 94*, pages 169–173, Abbaye de la Bussière sur Ouche, France, October 1994. INRIA.

[Sen97a]   N. Sendrier. Finding the permutation between equivalent binary codes. In *IEEE Conference, ISIT'97*, Ulm, Germany, June 1997.

[Sen97b]   N. Sendrier. On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293, May 1997.