Code-based Cryptography: Lecture Notes

Thomas Debris-Alazard

Code-based Cryptography:

Lecture Notes

Thomas Debris-Alazard

Inria

These lecture notes have been written for courses given at École normale supérieure de Lyon and summer school 2022 in post-quantum cryptography that took place in the university of Budapest. Our objective is to give a general introduction to the foundations of code-based cryptography which is currently known to be secure even against quantum adversaries. In particular we focus our attention to the decoding problem whose hardness is at the ground of the security of many cryptographic primitives, the most prominent being McEliece and Alekhnovich' encryption schemes.

Comments and criticism are very welcome and can be sent to thomas.debris@inria.fr

Acknowledgments. We would like to thank Maxime Bombar and Alain Couvreur for their helpful comments and corrections.

Contents

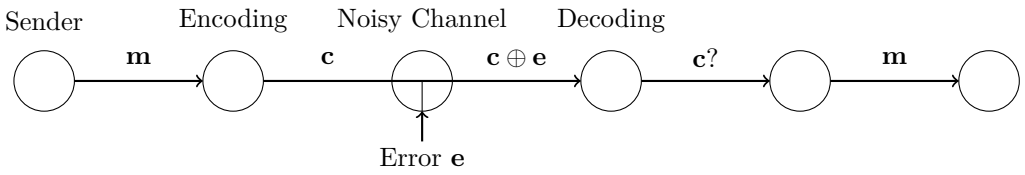
CHAPTER 1

An Intractable Problem Related to Codes: Decoding

Introduction

In this course we will consider *linear codes*, but what are these mathematical objects known as a linear code? It is a subspace of any n-dimensional space over some finite field. Linear codes were initially introduced to preserve the quality of information stored on a physical device or transmitted across a noisy channel. The key principle for achieving such task is extremely simple and natural: *adding redundancy*. A trivial illustration is when we try to spell our name over the phone: S like Sophie, T like Terence, E like Emily, ...

In a digital environment the basic idea to mimic our example is as follows, let $\mathbf{m}$ be a message of k bits that we would like to transmit over a noisy channel. Let us begin by fixing a linear code $\mathcal{C}$ (a subspace) of dimension k over $\mathbb{F}_2^n$ (the words of n bits). By linearity it is easy to map (and to invert) any k-bits word to some n-bits codeword. The task consisting in adding $n-k$ bits of redundancy is commonly called encoding. Once our message $\mathbf{m}$ to transmit is encoded into some codeword $\mathbf{c}$, we send it across the noisy channel. The receiver will therefore get a corrupted codeword $\mathbf{c} \oplus \mathbf{e}$ where some bits of $\mathbf{c}$ have been flipped. Receiver's challenge lies now in recovering $\mathbf{c}$, and thus $\mathbf{m}$, from $\mathcal{C}$ and $\mathbf{c} \oplus \mathbf{e}$, a task called *decoding*. The situation is described in the following picture.



A first, but quite simple, realistic and natural modelization for the noisy channel is the so-called binary symmetric channel: each bit of $\mathbf{c}$ is independently flipped with some probability $p \in [0, 1/2)$. In such a case, given a received word $\mathbf{y} = (y_1, \ldots, y_n)$, the probability that $\mathbf{c} = (c_1, \ldots, c_n)$ was sent is given by:

$$\mathbb{P}\left(\mathbf{c} \text{ was sent} \mid \mathbf{y} \text{ is received}\right) = p^{d_{\mathrm{H}}(\mathbf{c},\mathbf{y})}(1-p)^{n-d_{\mathrm{H}}(\mathbf{c},\mathbf{y})}$$

where $d_{\mathrm{H}}(\mathbf{c}, \mathbf{y}) \stackrel{\text{def}}{=} \sharp\{i \in [\![1, n]\!] : c_i \neq y_i\}$ is known as the *Hamming distance* between $\mathbf{c}$ and $\mathbf{y}$. Using this probability, it is easily verified that any decoding candidate $\mathbf{c} \in \mathcal{C}$ is even more likely as it is close to the received message $\mathbf{y}$ *for the Hamming distance*. It explains why "decoding" has historically consisted, given an input, to find the closest codeword *for the Hamming distance* (usually called maximum likelihood decoding).

Obviously, the naive procedure enumerating the $\sharp\mathcal{C} = 2^k$ codewords is to avoid. Coding theory aims at proposing family of codes with an explicit and efficient decoding procedure. Until today two families were roughly proposed: (i) codes derived from strong algebraic structures like Reed-Solomon codes [**MS86**, Chapter 10] and their their natural generalization known as algebraic geometry codes [**Gop81**], Goppa codes [**MS86**, Chapter 12] or (ii) those equipped with a probabilistic decoding algorithm like convolutional codes [**Eli55**], LDPC codes [**Gal63**] or more recently polar codes [**Arı09**] (which are used in the $5G$). It has been necessary to introduce all these structures because (even after 70 years of research)

decoding a linear code without any "peculiar" structure is an intractable problem, topic of these lecture notes.

Basic notation. The notation $x \stackrel{\text{def}}{=} y$ means that x is defined to be equal to y. Given a finite set $\mathscr{E}$, we will denote by $\sharp\mathscr{E}$ its cardinality. We denote by $\mathbb{F}_q$ the finite field with q elements, $\mathbb{F}_q^n$ will denote its n-dimensional version for some $n \in \mathbb{N}$. Vectors will be written with bold letters (such as $\mathbf{e}$) and upper-case bold letters are used to denote matrices (such as $\mathbf{H}$). If $\mathbf{e}$ is a vector in $\mathbb{F}_q^n$, then its components are $(e_1, \ldots, e_n)$. *Let us stress that vectors are in row notation.*

1.1. Codes: Basic Definitions and Properties

The main objective of this section is to introduce the basic concept of a code and define some of its parameters.

DEFINITION 1.1.1 (Linear code, length, dimension, rate, codewords). *A linear code $\mathcal{C}$ of length n and dimension k over $\mathbb{F}_q$ – for short, an $[n,k]_q$-code – is a subspace of $\mathbb{F}_q^n$ of dimension k. The rate of $\mathcal{C}$ is defined as k/n and elements of $\mathcal{C}$ are called codewords.*

Notice that the rate measures the amount of redundancy introduced by the code.

REMARK 1.1.1. *A code is more generally defined as a subset of $\mathbb{F}_q^n$. However in these lecture notes we will only consider* linear *codes. It may happen that we confuse codes and linear codes but for us a code will always be linear.*

EXERCISE 1.1.1. Give the dimension of the following linear codes:

1. $\{(f(x_1), \ldots, f(x_n)) : f \in \mathbb{F}_q[X] \text{ and } \deg(f) < k\}$ where the x_i's are distinct elements of $\mathbb{F}_q$,
2. $\{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U \text{ and } \mathbf{v} \in V\}$ where U (*resp.* V) is an $[n, k_U]_q$-code (*resp.* $[n, k_V]_q$-code).

DEFINITION 1.1.2 (Inner product, dual code). *The inner product between $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is defined as $\mathbf{x} \cdot \mathbf{y} \stackrel{def}{=} \sum_{i=1}^n x_i y_i \in \mathbb{F}_q$. The dual of a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is defined as $\mathcal{C}^* \stackrel{def}{=} \left\{\mathbf{c}^* \in \mathbb{F}_q^n : \forall \mathbf{c} \in \mathcal{C},\ \mathbf{c} \cdot \mathbf{c}^* = 0\right\}$.*

The dual of an $[n,k]_q$-code $\mathcal{C}$ is an $[n, n-k]_q$-code. Although $\dim(\mathcal{C}) + \dim(\mathcal{C}^*) = n$, it may happen that $\mathcal{C}$ and $\mathcal{C}^*$ are not in direct sum. This is even more remarkable that coding theorists have given a name to $\mathcal{C} \cap \mathcal{C}^*$: the hull of $\mathcal{C}$.

Representation of a code. To represent an $[n,k]_q$-code $\mathcal{C}$ we may take any basis of it, namely a set of k linearly independent vectors $\mathbf{g}_1, \ldots, \mathbf{g}_k \in \mathcal{C}$, and form the matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ whose rows are the $\mathbf{g}_i$'s. Then $\mathcal{C}$ can be written as

$$\mathcal{C} = \left\{\mathbf{mG} : \mathbf{m} \in \mathbb{F}_q^k\right\}.$$

Conversely, any matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ of rank k defines a code with the previous representation. Such matrix $\mathbf{G}$ is usually called *a generator matrix of* $\mathcal{C}$.

Another representation of $\mathcal{C}$ is by a so-called *parity-check matrix*. Let $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ be such that its rows form a basis of $\mathcal{C}^*$, the linear code $\mathcal{C}$ can be written as

$$\mathcal{C} = \left\{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{Hc}^\intercal = \mathbf{0}\right\}.$$

Conversely, any matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ of rank $n-k$ defines an $[n,k]_q$-code with the previous representation. We call $\mathbf{H}$ a *parity-check matrix* of $\mathcal{C}$.

Notice now by basic linear algebra that for any non-singular matrix $\mathbf{S}$ of size $k \times k$ (*resp.* $(n-k) \times (n-k)$), $\mathbf{SG}$ (*resp.* $\mathbf{SH}$) is still a generator (*resp.* parity-check) matrix of $\mathcal{C}$. Therefore, left multiplication by an invertible matrix "does not change the code", it just gives another basis. This is not the case if we perform

some right multiplication. For instance if $\mathbf{P}$ denotes an $n \times n$ permutation matrix, then $\mathbf{GP}$ will be the generator matrix of the code $\mathcal{C}$ permuted, namely $\{\mathbf{cP} : \mathbf{c} \in \mathcal{C}\}$.

EXERCISE 1.1.2. *Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be a generator matrix of some code $\mathcal{C}$. Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of rank $n-k$ such that $\mathbf{GH}^\intercal = \mathbf{0}$. Show that $\mathbf{H}$ is a parity-check matrix of $\mathcal{C}$.*

Among our two representations of a code, a natural question arises: are we able from one representation to compute the other one? The answer is obviously yes. To see this let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be a generator matrix of $\mathcal{C}$. As $\mathbf{G}$ has rank k, by a Gaussian elimination we can put it into *systematic form*, namely to compute a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{k\times k}$ such that $\mathbf{SG} = (\mathbf{1}_k \mid \mathbf{A})$ (up to a permutation of the columns). The matrix $\mathbf{SG}$ is still a generator matrix of $\mathcal{C}$. Now it is readily seen that $\mathbf{H} \stackrel{\text{def}}{=} (-\mathbf{A}^\intercal \mid \mathbf{1}_{n-k})$ verifies $\mathbf{GH}^\intercal = \mathbf{0}$, has rank $n-k$ and therefore is a parity-check matrix of $\mathcal{C}$.

Parity-check matrices may seem unnatural to represent a code when comparing to generator matrices. However, even if both representations are equivalent, the parity-check representation is in many applications more relevant (particularly in code-based cryptography). Let us give some illustration.

The Hamming code. Let $\mathcal{C}_{\text{Ham}}$ be the binary code of generator matrix:

$$\mathbf{G} \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

This code has length 7 and dimension 4. The following matrix:

$$\mathbf{H} \stackrel{\text{def}}{=} \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

has rank 3 and verifies $\mathbf{GH}^\intercal = \mathbf{0}$. It is therefore a parity-check matrix of our code $\mathcal{C}_{\text{Ham}}$. Notice that $\mathbf{H}$ has a quite nice structure, its columns are the integers from 1 to 7 written in binary.

Suppose now that we would like to recover $\mathbf{c} \in \mathcal{C}_{\text{Ham}}$ from $\mathbf{y} \stackrel{\text{def}}{=} \mathbf{c} + \mathbf{e}_i$ where $\mathbf{e}_i$ is the vector of all zeros except a single 1 at the i'th position. It is not clear how to use $\mathbf{G}$ to recover $\mathbf{c}$. However note that $\mathbf{Hy}^\intercal = \mathbf{Hc}^\intercal + \mathbf{He}_i^\intercal = \mathbf{He}_i^\intercal$ which is the i'thm column of $\mathbf{H}$. Therefore the position i of the error is given by the integer in the binary representation $\mathbf{Hy}^\intercal$.

The code $\mathcal{C}_{\text{Ham}}$ is in fact known as the Hamming code of length 7. It belongs to the family of Hamming codes which are the $[2^r - 1, 2^r - 1 - r]_2$-codes built from the $r \times (2^r - 1)$ parity-check matrix whose i-th column is the binary representation of i. Hamming codes are the caricatural example of codes that are better understood with their parity-check matrices. Furthermore we can easily correct one error by using their parity-check matrix instead of their generator matrix.

Our example has shown the relevance of parity-check matrices to understand a code. In particular, we saw how $\mathbf{Hy}^\intercal$ could be a nice source of information when trying to decode $\mathbf{y}$. This is even more remarkable that we give a name to $\mathbf{Hy}^\intercal$.

DEFINITION 1.1.3 (Syndrome). *Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. The syndrome of $\mathbf{y}$ with respect to $\mathbf{H}$ is defined as $\mathbf{Hy}^\intercal$. Any element of $\mathbb{F}_q^{n-k}$ is called a syndrome.*

Syndromes are a natural set of representatives of the code cosets:

DEFINITION 1.1.4 (Coset). *Let $\mathcal{C}$ be a linear code over $\mathbb{F}_q$ and $\mathbf{a} \in \mathbb{F}_q^n$. The coset of $\mathbf{a}$ (relatively to $\mathcal{C}$) is defined as*

$$\mathcal{C}(\mathbf{a}) \stackrel{def}{=} \mathbf{a} + \mathcal{C}.$$

Notice that any parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ of an $[n,k]_q$-code $\mathcal{C}$ has rank $n-k$. Therefore for any syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, there exists some $\mathbf{a_s} \in \mathbb{F}_q^n$ such that $\mathbf{Ha_s^\intercal} = \mathbf{s}^\intercal$.

LEMMA 1.1.1. *Let $\mathcal{C}$ be an $[n,k]_q$-code of parity-check matrix $\mathbf{H}$. Then for any $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^n$,*

$$\mathcal{C}(\mathbf{a}) = \mathcal{C}(\mathbf{b}) \iff \mathbf{Ha}^\intercal = \mathbf{Hb}^\intercal.$$

PROOF. Notice that,

$$\begin{aligned} \mathbf{Ha}^\intercal = \mathbf{Hb}^\intercal &\iff \mathbf{H}(\mathbf{a}-\mathbf{b})^\intercal = \mathbf{0} \\ &\iff \mathbf{a}-\mathbf{b} \in \mathcal{C} \end{aligned}$$

which concludes the proof. □

Cosets play an important role in the geometry of a code. They partition the space $\mathbb{F}_q^n$ according to $\mathcal{C}$: they are the representatives of the "torus" $\mathbb{F}_q^n/\mathcal{C}$. Notice now that syndromes are a nice set of representatives of $\mathbb{F}_q^n/\mathcal{C}$ via the isomorphism for some parity-check matrix of $\mathcal{C}$: $\mathbf{x} \in \mathbb{F}_q^n/\mathcal{C} \mapsto \mathbf{Hx}^\intercal \in \mathbb{F}_q^{n-k}$ (which is well defined and one to one by Lemma 1.1.1). In particular we can partition $\mathbb{F}_q^n$ as follows

$$\mathbb{F}_q^n = \bigsqcup_{\mathbf{s}\in\mathbb{F}_q^n} (\mathbf{a_s} + \mathscr{C})$$

where $\mathbf{a_s} \in \mathbb{F}_q^n$ is such that $\mathbf{Ha_s^\intercal} = \mathbf{s}^\intercal$.

Minimum distance. Let us define now an important parameter for a code: its minimum distance. It measures the quality of a code in terms of "decoding capacity", namely how many errors has to be added before a noisy codeword could be confused with another noisy codeword.

The minimum distance of a code relies on the definition of Hamming weight.

DEFINITION 1.1.5 (Hamming weight, distance). *The Hamming weight of $\mathbf{x} \in \mathbb{F}_q^n$ is defined as the number of its non-zero coordinates,*

$$|\mathbf{x}| \stackrel{def}{=} \sharp\{i \in [\![1,n]\!] \,:\, x_i \neq 0\}.$$

The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as $|\mathbf{x}-\mathbf{y}|$.

REMARK 1.1.2. *Notice that the Hamming metric is a coarse metric which can only take $n+1$ values. Furthermore, it does not distinguish "small" and "large" coefficients contrary to the Euclidean metric. For instance, in $\mathbb{F}_{11}^3$, vectors $(5,3,0)$ and $(1,0,1)$ have the same Hamming weight.*

In what follows $\mathbb{F}_q^n$ will always be embedded with the Hamming distance. However one may wonder if other metrics could be interesting for telecommunication or cryptographic purposes. The answer is yes, we can cite the Lee or rank metrics but this is out of the scope of these lecture notes.

DEFINITION 1.1.6 (Minimum distance). *The minimum distance of a linear code $\mathcal{C}$ is defined as the shortest Hamming weight of non-zero codewords,*

$$d_{\min}(\mathcal{C}) \stackrel{def}{=} \min\{|\mathbf{c}| \,:\, \mathbf{c} \in \mathcal{C}\backslash\{\mathbf{0}\}\}.$$

EXERCISE 1.1.3. Give the minimum distance of the following codes:

1. $\{(f(x_1), \ldots, f(x_n)) : f \in \mathbb{F}_q[X] \text{ and } \deg(f) < k\}$ where the x_i's are distinct elements of $\mathbb{F}_q$.
2. $\{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U \text{ and } \mathbf{v} \in V\}$ where U (*resp.* V) is a code of length n over $\mathbb{F}_q$ and minimum distance d_U (*resp.* d_V).
3. The Hamming code of length $2^r - 1$.

   Hint: *A code has minimum distance d if and only if for some parity-check matrix $\mathbf{H}$ every $(d-1)$-tuple of columns are linearly independent and there is at least one linearly linked dtuple of columns.*

The following elementary lemma asserts that for a code of minimum distance d, if a received word has less than $(d-1)/2$ errors (the error has an Hamming weight smaller than $(d-1)/2$), then it can be successfully decoded: the exhaustive search of the closest codeword will output the "right" codeword. We stress here that this does not show the existence of an efficient decoding algorithm, which is far from being guaranteed. Furthermore we will see later that for random codes of minimum distance d, balls centered at codewords and with radius $\approx d$ typically do not intersect, showing that decoding can theoretically be done for these codes up to distance $\approx d$ and not $(d-1)/2$.

LEMMA 1.1.2. *Let $\mathcal{C}$ be a code of minimum distance d, then balls of radius $\frac{d-1}{2}$ centered at codewords are disjoint,*

$$\forall \mathbf{c}, \mathbf{c}' \in \mathcal{C},\ \mathbf{c} \neq \mathbf{c}',\ \mathscr{B}\left(\mathbf{c}, \frac{d-1}{2}\right) \bigcap \mathscr{B}\left(\mathbf{c}', \frac{d-1}{2}\right) = \emptyset$$

where $\mathscr{B}(\mathbf{x}, r)$ denotes the ball of radius r and center $\mathbf{x}$ for the Hamming distance.

PROOF. Let $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$ be two distinct codewords. Let us assume that there exists $\mathbf{x} \in \mathscr{B}\left(\mathbf{c}, \frac{d-1}{2}\right) \cap \mathscr{B}\left(\mathbf{c}', \frac{d-1}{2}\right)$. By using the triangle inequality we obtain

$$\begin{aligned} |\mathbf{c} - \mathbf{c}'| &\leq |\mathbf{c} - \mathbf{x}| + |\mathbf{x} - \mathbf{c}'| \\ &\leq \frac{d-1}{2} + \frac{d-1}{2} \\ &= d - 1 \end{aligned}$$

which contradicts the fact that $\mathcal{C}$ has minimum distance d. □

EXERCISE 1.1.4. *Let $\mathbf{H}$ be a parity-check matrix of a code $\mathcal{C}$ of minimum distance d. Show that the $\mathbf{He}^\intercal$'s are distinct when $|\mathbf{e}| \leq \frac{d-1}{2}$.*

EXERCISE 1.1.5. *Let $\mathcal{C} \subseteq \mathbb{F}_2^n$ be a code of minimum distance d and $t > n - \frac{d}{2}$. Show that there exists at most one codeword $\mathbf{c} \in \mathcal{C}$ of weight t.*

From this lemma we see that a code with a large minimum distance is a "good" code in terms of decoding ability. However there is another parameter to take into account: the rate. A code of small rate asks for adding a lot of redundancy when encoding a message to send, thing that we would like to avoid in telecommunications (where the perfect situation corresponds to not adding any redundancy). Therefore we would like to find a code with large minimum distance and large rate. As it might be expected these two considerations are diametrically opposed to each other. There exists many bounds to quantify the relations between the rate and the minimum distance but this is out of the scope of these lecture notes.

As we will see in Chapter 2, a "random code" with a constant rate $k/n \in (0,1)$ has a very good minimum distance, namely $d \sim Cn$ for some constant $C > 0$ (known as the relative *Gilbert-Varshamov* bound) when $n \to +\infty$. However, while we expect a typical code to have a minimum distance linear in its length given some rate, it is a hard problem to explicitly build linear codes with such minimum distance.

1.2. The Decoding Problem

Now that linear codes are defined we are ready to present more formally the decoding problem. Below are presented two equivalent versions of this problem. The first presentation is natural when dealing with noisy codewords (as we did until now) but we will mostly consider in these lecture notes the second one (with syndromes) which is more suitable for cryptographic purposes. For each problem a code is given as input but with a generator or parity-check representation.

PROBLEM 1.2.1 (Noisy Codeword Decoding). *Given* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ *of rank* k, $t \in [\![0, n]\!]$, $\mathbf{y} \in \mathbb{F}_q^n$ *where* $\mathbf{y} = \mathbf{c} + \mathbf{e}$ *with* $\mathbf{c} = \mathbf{mG}$ *for some* $\mathbf{m} \in \mathbb{F}_q^k$ *and* $|\mathbf{e}| = t$, *find* $\mathbf{e}$.

PROBLEM 1.2.2 (Syndrome Decoding). *Given* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *of rank* $n-k$, $t \in [\![0, n]\!]$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ *where* $\mathbf{He}^\intercal = \mathbf{s}^\intercal$ *with* $|\mathbf{e}| = t$, *find* $\mathbf{e}$.

REMARK 1.2.1. *Solving the decoding problem comes down to solve a linear system* but *with some non-linear constraint on the solution (here its Hamming weight). Notice that without such constraint it would be easy to solve the problem with a Gaussian elimination.*

It turns out that these two (worst-case) problems are strictly equivalent, if we are able to solve one of them, then we can turn our algorithm into another algorithm that solves the other one in the same running time (up to some small polynomial time overhead).

Suppose that (i) we have an algorithm solving Problem 1.2.1 and (ii) we would like to solve Problem 1.2.2. To this aim, let $(\mathbf{H}, \mathbf{s})$ be an input of Problem 1.2.2. First, as $\mathbf{H}$ has rank $n-k$ we can compute a matrix $\mathbf{G}$ of rank k such that $\mathbf{GH}^\intercal = \mathbf{0}$. It is equivalent to computing a generator matrix of the code $\mathcal{C}$ with parity-check matrix $\mathbf{H}$. This can be done in polynomial time (over n) by performing a Gaussian elimination. Then, by solving a linear system (which also can be done in polynomial time) we can find $\mathbf{y}$ such that $\mathbf{Hy}^\intercal = \mathbf{s}^\intercal$. Notice now that $\mathbf{H}(\mathbf{y} - \mathbf{e})^\intercal = \mathbf{0}$ where $\mathbf{He}^\intercal = \mathbf{s}^\intercal$ and $|\mathbf{e}| = t$. Therefore $\mathbf{y} = \mathbf{c} + \mathbf{e}$ for some $\mathbf{c} \in \mathcal{C}$, namely $\mathbf{c} = \mathbf{mG}$ for some $\mathbf{m} \in \mathbb{F}_q^k$. From this we can use our algorithm solving the noisy codeword version of the decoding problem to recover the error $\mathbf{e}$.

EXERCISE 1.2.1. *Show that any solver of Problem 1.2.2 can be turned in polynomial time into an algorithm solving Problem 1.2.1.*

In what follows we will mainly consider the syndrome version of the decoding problem. Furthermore, we will call *decoding algorithm*, any algorithm solving this problem (or its equivalent version with noisy codewords).

A little bit about the decoding problem hardness. Our aim in these lecture notes is to show that decoding is *hard*

- in the worst case (NP–complete),
- in average (it will be defined in a precise manner later).

However, even though the decoding problem is hard in the "worst case" and in "average", let us stress that there are codes that we know how to decode efficiently (hopefully for telecommunications...). It may seem counter-intuitive at first glance: is the decoding problem hard or not? All the subtlety lies in the inputs that are given. Is the code given as input particular? How is the decoding distance t (for instance with $t = 1$ we have an easy problem)? In fact the hardness of the decoding problem relies on how we answer to these questions. There exists some codes and decoding distances for which the problem is easy to solve. The NP–completeness shows that we cannot hope to solve the decoding problem in polynomial time for all inputs while the average hardness ensures (for well chosen t) that for almost all code the problem

is intractable. Our aim in what follows is to show this. But we will first exhibit a family of codes with associate decoding distances t for which decoding is easy. The existence of such codes is at the foundation of code-based cryptography.

Codes that we know to decode: Reed-Solomon codes. The family of ReedSolomon codes is of central interest in coding theory: many algebraic constructions of codes that we know how to decode efficiently such as BCH codes [**MS86**, Chapter 3], Goppa codes [**MS86**, Chapter 12] derive from this family. Reed-Solomon codes are practically used for instance in compact discs, DVDs, BluRays, QR codes etc...

DEFINITION 1.2.1 (Generalized Reed-Solomon Codes). *Let $\mathbf{z} \in (\mathbb{F}_q^\star)^n$ and $\mathbf{x}$ be an n-tuple of pairwise distinct elements of $\mathbb{F}_q$ (in particular $n \leq q$) and let $k \leq n$. The code $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ is defined as*

$$\mathsf{GRS}_k(\mathbf{x}, \mathbf{z}) \stackrel{def}{=} \{(z_1 f(x_1), \ldots, z_n f(x_n)) \ : \ f \in \mathbb{F}_q[X] \text{ and } \deg(f) < k\}\,.$$

Generalized ReedSolomon codes $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ are $[n, k]_q$-codes with many remarkable properties. Among others, they are said "MDS", *i.e.* their minimum distance equals $d \stackrel{\text{def}}{=} n - k + 1$ and there exists an efficient decoding algorithm to correct any pattern of $\lfloor \frac{d-1}{2} \rfloor$ errors as we will see below. However, a major drawback of Generalized Reed-Solomon codes is that their length is upper-bounded by the size of the alphabet $\mathbb{F}_q$.

EXERCISE 1.2.2. *Show that $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})^* = \mathsf{GRS}_{n-k}(\mathbf{x}, \mathbf{z}')$ where $z'_i = \frac{1}{z_i \prod_{j \neq i}(x_i - x_j)}$. Deduce that $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ has a parity-check matrix of the following form:*

$$\mathbf{H} \stackrel{def}{=} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{n-k-1} & x_2^{n-k-1} & \cdots & x_n^{n-k-1} \end{pmatrix} \begin{pmatrix} z'_1 & & & 0 \\ & z'_2 & & \\ & & \ddots & \\ 0 & & & z'_n \end{pmatrix} \tag{1.1}$$

Furthermore, show that $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ has minimum distance $n - k + 1$.

Decoding Generalized Reed-Solomon codes at distance $\leq \lfloor \frac{n-k}{2} \rfloor$. Suppose that $\mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ is given as input, namely that the x_i's and z_i's are known (or equivalently $\mathbf{H}$ given in Equation (1.1)). Let $\mathbf{y}$ be a noisy codeword that we would like to decode:

$$\mathbf{y} \stackrel{\text{def}}{=} \mathbf{c} + \mathbf{e}$$

where $\mathbf{c} = (z_1 f(x_1), \cdots, z_n f(x_n)) \in \mathsf{GRS}_k(\mathbf{x}, \mathbf{z})$ with $f \in \mathbb{F}_q[X]$ such that $\deg(f) < k$ and $\mathbf{e} \in \mathbb{F}_q^n$ be an error of Hamming weight $t \leq \lfloor \frac{n-k}{2} \rfloor$. Let us suppose without loss of generality that $z_1 = \cdots = z_n = 1$ (for the general case multiply each coordinate of $\mathbf{y}$ by z_i^{-1} which does not change the weight of the error term). Our aim is to recover f (or equivalently $\mathbf{e}$). Let us introduce the following (unknown) polynomial:

$$E(X) \stackrel{\text{def}}{=} \prod_{i : e_i \neq 0} (X - x_i).$$

Notice that $\deg(E) = t$. The key ingredient of the decoding algorithm is the following fact

FACT 1.2.1.

$$\forall i \in [\![1, n]\!], \quad y_i E(x_i) = f(x_i) E(x_i). \tag{1.2}$$

Coordinates y_i and x_i are known while f and E are unknown. System (1.2) is not linear and the basic idea to decode is to *linearize* it (to bring us to a pleasant case). Let,

$$N \stackrel{\text{def}}{=} Ef$$

Equation (1.2) can be rewritten as:

$$\forall i \in [\![1, n]\!], \quad y_i E(x_i) = N(x_i) \tag{1.3}$$

where coefficients of the polynomial $N \in \mathbb{F}_q[X]$ of degree $< k + t$ and $E \in \mathbb{F}_q[X]$ of degree t are unknowns. This system has a non-trivial solution: (E, Ef) but it may have many other solutions. The following lemma asserts that any other non-trivial solution enables to recover f.

LEMMA 1.2.1. *Let $E_1, E_2 \in \mathbb{F}_q[X]$ of degree $\leq \lfloor \frac{n-k}{2} \rfloor$ and $N_1, N_2 \in \mathbb{F}_q[X]$ of degree $< k + \lceil \frac{n-k}{2} \rceil$ such that (E_1, N_1) and (E_2, N_2) are non-zero and solutions of Equation (1.3). Then,*

$$\frac{N_1}{E_1} = \frac{N_2}{E_2} = f.$$

PROOF. First, if $E_i = 0$ then N_i has n roots by Equation (1.3) while its degree is smaller than n. Therefore we get that $E_i \neq 0$ as (E_i, N_i) is non-zero. Let $R \stackrel{\text{def}}{=} N_1 E_2 - N_2 E_1$, we have

$$\deg(R) < k + \left\lfloor \frac{n-k}{2} \right\rfloor + \left\lceil \frac{n-k}{2} \right\rceil \leq n.$$

By using now that (E_1, N_1) and (E_2, N_2) are solutions of Equation (1.3) we obtain for all $i \in [\![1, n]\!]$,

$$\begin{aligned} R(x_i) &= N_1(x_i)E_2(x_i) - N_2(x_i)E_1(x_i) \\ &= y_i E_1(x_i)E_2(x_i) - y_i E_2(x_i)E_1(x_i) \\ &= 0 \end{aligned}$$

Therefore R has n roots while its degree is smaller than n. It shows that $R = 0$ and $\frac{N_1}{E_1} = \frac{N_2}{E_2}$. It concludes the proof as (E, Ef) is also a non-zero solution of Equation (1.3). □

The algorithm we just described to decode a generalized Reed-Solomon code up to the distance $\lfloor \frac{n-k}{2} \rfloor$ is known as the *Berlekamp-Welch* algorithm.

1.2.1. Worst Case Hardness. The aim of this subsection is to show that the decoding problem is hard in the *worst case*, namely NP–complete. We have to be careful with this kind of statement. First, NP–completeness is designed for decisional problems: "is there a solution given some input?". Furthermore, we need to be very cautious with inputs that are being fed to our problem. The NP–completeness "only" shows (under the assumption $\mathsf{P} \neq \mathsf{NP}$) that we cannot hope to have an algorithm solving our problem in polynomial time *for all inputs*. The set of possible inputs is therefore important, it may happen that a problem is easy to solve when its inputs are drawn from a set A while it becomes hard (NP–complete) when its inputs are taken from some set $B \supsetneq A$. This remark has to be carefully taken into consideration when using the NP–completeness in cryptography as a safety guaranty. It is quite possible that the security of a cryptosystem relies on the difficulty to solve an NP–complete problem but at the same time breaking the scheme amounts to solve the problem on a subset of inputs for which it is easy. To summarize, the NP–completeness of a problem for a cryptographic use is a nice property but it is not the panacea to ensure its hardness.

The foregoing discussion has shown that we have to rephrase the decoding problem as a decisional problem. Furthermore, it will be important to have a careful look on the set of inputs.

PROBLEM 1.2.3 (Decisional Decoding Problem).

- Input: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ *where* $n, k \in \mathbb{N}$ *with* $k \leq n$ *and an integer* $t \leq n$.
- Decision: *it exists* $\mathbf{e} \in \mathbb{F}_q^n$ *of Hamming weight* t *such* $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$.

PROPOSITION 1.2.1 ([BMvT78]). *Problem 1.2.3 for $q = 2$ is* NP*–complete.*

The proof of this proposition relies on a reduction of the following combinatorial decision problem, which is known to be NP–complete.

PROBLEM 1.2.4 (Three Dimensional Matching (3DM)).

- Input: *a subset* $U \subseteq T \times T \times T$ *where* T *is a finite set.*
- Decision: *it exists* $V \subseteq U$ *such that* $\sharp V = \sharp T$ *and for all* $(x_1, y_1, z_1), (x_2, y_2, z_2) \in V$ *we have* $x_1 \neq x_2, y_1 \neq y_2$ *et* $z_1 \neq z_2$.

The formalism of this problem may seem at first sight to be far away from the decoding problem. However we can restate it with incidence matrices. We proceed as follows: first we take each first coordinate of elements that belong to U, then we build an incidence matrix relatively to T of size $\sharp T \times \sharp U$ and similarly for the two remaining coordinates. After that we vertically concatenate our three matrices. Therefore we get in polynomial time a matrix of size $3\sharp T \times \sharp U$, that we will call a 3DM-*incidence matrix*. But now, as shown in the following lemma, we have a solution to the 3DM-problem associated to U and T if and only if there are $\sharp T$ columns that sum up to the all one vector (which corresponds to our decoding problem). But let us first give an example to illustrate this discussion.

EXAMPLE 1.2.1. *Let* $T = \{1, 2, 3\}$ *and* $U = \{u_1, u_2, u_3, u_4, u_5\}$ *such that:*

$$u_1 = (1, 1, 2), \quad u_2 = (2, 3, 1), \quad u_3 = (1, 2, 3)$$

$$u_4 = (3, 1, 2) \quad \textit{and} \quad u_5 = (2, 2, 2).$$

The 3DM-*incidence matrix associated to these sets is given by:*

	112	*231*	*123*	*312*	*222*
1	*1*	*0*	*1*	*0*	*0*
2	*0*	*1*	*0*	*0*	*1*
3	*0*	*0*	*0*	*1*	*0*
1	*1*	*0*	*0*	*1*	*0*
2	*0*	*0*	*1*	*0*	*1*
3	*0*	*1*	*0*	*0*	*0*
1	*0*	*1*	*0*	*0*	*0*
2	*1*	*0*	*0*	*1*	*1*
3	*0*	*0*	*1*	*0*	*0*

We obtain the all one vector by summing columns $2, 3$ *and* 4. *Therefore,* $V = \{u_2, u_3, u_4\}$ *is a solution.*

LEMMA 1.2.2. *Let* T *and* $U \subseteq T \times T \times T$ *be an instance of* 3DM *and let* $\mathbf{H}_{\text{3DM}} \in \mathbb{F}_2^{3\sharp T \times \sharp U}$ *be the associated incidence matrix. We have*

$$\textit{There is a solution for the instance } T, U \iff \exists \mathbf{e} \in \mathbb{F}_2^{\sharp U} \ : \ |\mathbf{e}| = \sharp T \quad \textit{and} \quad \mathbf{H}_{\text{3DM}}\mathbf{e}^\intercal = \mathbf{1}^\intercal \textit{(all one vector).}$$

PROOF. By definition, columns of $\mathbf{H}_{\text{3DM}}$ have length $3\sharp T$ and Hamming weight 3. Therefore, $\sharp T$ columns sum up to the all one vector if and only if their supports are pairwise distinct. □

We are now ready to prove Proposition 1.2.1.

PROOF OF PROPOSITION 1.2.1. Let T, U be an instance of the three dimensional matching problem. We can build in polynomial time the matrix $\mathbf{H}_{\text{3DM}}$. Now, by Lemma 1.2.2, there is a solution for T and U if and only if there is a solution of the decoding problem for the input $(\mathbf{H}_{\text{3DM}}, \mathbf{1})$ and $t \stackrel{\text{def}}{=} \sharp T$. □

We have just proven that decoding is an NP–complete problem but when is given as input a binary matrix and a decoding distance. In other words, we cannot reasonably hope to find a polynomial time algorithm to solve the decoding problem for all codes over $\mathbb{F}_2$ and for all decoding distances. But can we find a proof that fits with a restricted set of inputs? The answer is yes. Below is presented an incomplete list of some improvements. The decoding problem is still NP–complete if we restrict:

- the decoding distance at $t = n/\log_2 n$ [**Fin09**] or $t = Cn$ for any constant $C \in (0, 1)$ [**Deb19**].
- the input codes are restricted to Reed-Solomon codes [**GV05**]
- etc...

There are many other NP–complete problems related to codes. For instance, computing the minimum distance of a code [**Var97**] or some codewords of weight w [**BMvT78**] are NP–complete.

1.2.2. Average Case Hardness. The decoding worst-case hardness makes it a suitable problem for cryptographic applications. However we have to be careful when dealing with the decoding problem in this context. Recall that the aim of any cryptosystem is to base its security on the "hardness" of solving some problem. However to study and to ensure the hardness (thus the security) it would be preferable first to define the problem *exactly* as it is stated when wanting to break the crypto-system. It leads us to the following question: "how the decoding problem is used in cryptography?". To answer this question let us briefly present the McEliece public key encryption scheme [**McE78**] that was introduced just few months after RSA. This scheme will motivate our definition of the "cryptographic" decoding in Problem 1.2.5.

McEliece encryption scheme. McEliece's idea to build a public key encryption scheme based on codes is as follows: Alice, the secret key owner, has a code $\mathcal{C}$ that she can efficiently decode up to some distance t (some "quantity" that enables to decode is the secret). Alice publicly reveals a parity-check matrix of her code, let us say $\mathbf{H}$, as well as its associated decoding distance t. For obvious security reasons Alice does not want $\mathbf{H}$ to reveal any information on how she decodes $\mathcal{C}$. In that case, the perfect situation corresponds to a matrix $\mathbf{H}$ *which is uniformly distributed*. Now Bob wants to send a message $\mathbf{m}$ to Alice. First he associates with a public one to one mapping (in a sense to define) his message $\mathbf{m}$ to some vector $\mathbf{e}$ of Hamming weight t. Then he computes $\mathbf{He}^\intercal$ and sends it to Alice. Once again, for obvious security reasons, Bob does not want $\mathbf{e}$ to share any information with $\mathbf{m}$ that could be used when observing $\mathbf{He}^\intercal$. The perfect situation corresponds to a mapping such that $\mathbf{e}$ *is uniformly distributed over words of Hamming weight* t. Now Alice who got $\mathbf{He}^\intercal$ recovers $\mathbf{e}$ and $\mathbf{m}$ thanks to her decoding algorithm.

One may wonder why Bob has associated its message to some word of weight t and not $\leq t$ as Alice can decode up to the distance t. The reason is that any malicious person looking at the discussion between Alice and Bob observes $\mathbf{He}^\intercal$ and to recover the message she/he has to find $\mathbf{e}$. However, decoding is harder if $|\mathbf{e}|$ is larger. Therefore it is preferable if $\mathbf{e}$ has a Hamming weight as large as possible, thus t.

Remark 1.2.2. *McEliece encryption scheme relies on the use of generator matrices. We have actually presented Niederreiter encryption scheme* [**Nie86**]. *The security of both schemes is the same. The only differences are in term of efficiency, depending of the context.*

Exercise 1.2.3. *Describe how the encryption scheme works with generator matrices.*

We are now ready to define the (average) decoding problem for cryptographic applications. In what follows q will denote a fixed field size while $R(n)$ and $\tau(n)$ will be functions taking their values in $(0, 1)$. To simplify notation, since n is clear here from the context, we will drop the dependency in n and simply write R and τ.

Problem 1.2.5 (Decoding Problem - $\mathsf{DP}(n, q, R, \tau)$). *Let* $k \stackrel{def}{=} \lfloor Rn \rfloor$ *and* $t \stackrel{def}{=} \lfloor \tau n \rfloor$.

- Input: $(\mathbf{H}, \mathbf{s} \stackrel{def}{=} \mathbf{x}\mathbf{H}^\intercal)$ *where* $\mathbf{H}$ *(resp.* $\mathbf{x}$*) is uniformly distributed over* $\mathbb{F}_q^{(n-k)\times n}$ *(resp. words of Hamming weight* t *in* $\mathbb{F}_q^n$*).*
- Output: *an error* $\mathbf{e} \in \mathbb{F}_q^n$ *of Hamming weight* t *such that* $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$.

REMARK 1.2.3. *This problem really corresponds to decode a code of rate* R *and parity-check matrix* $\mathbf{H}$*. We call such a code a random code as its parity-check matrix is uniformly distributed (for more details see Chapter 2).*

REMARK 1.2.4. *In our definition of* DP*, we ask given a code and a syndrome obtained via a vector* $\mathbf{x}$ *of weight* t*, to find a vector* $\mathbf{e}$ *with the same weight that reaches the syndrome. In particular, we do not ask to recover* $\mathbf{x}$*. It may seem confusing when looking at the original definition of decoding problem in telecommunications where it is requested to recover exactly* $\mathbf{x}$ *and thus the message that was sent. But such definition imposes some constraints over* t*, for instance* t *smaller than the minimum distance of the code out of* 2*, which ensures the uniqueness of the solution (see Lemma 1.1.2). However, in cryptography our constraints are not the same. Sometimes we ask* DP *to have a unique solution given some instance (like in encryption schemes), sometimes not (like in signatures). When thinking about the decoding problem in cryptography we have to forget the "telecommunication context". For now, our concern is the hardness of* DP*, whatever is the choice of* t*, whatever is the number of solutions. We will further discuss this (important) remark in Chapter 2 . As we will see, all the subtlety lies in the choice of* t*.*

We could have defined DP without any distribution on its inputs. However we are interested in the algorithmic hardness of this problem in the following way. Let us assume that we have a probabilistic algorithm $\mathscr{A}$ that solves (sometimes) the decoding problem at distance t. Furthermore, let us suppose that a single run of this algorithm costs a time T. Inputs of $\mathscr{A}$ are a parity-check matrix $\mathbf{H}$ and a syndrome $\mathbf{s}$. We denote by $\mathbf{w} \in \{0,1\}^\ell$ the internal coins of $\mathscr{A}$ which tries to output some $\mathbf{e}$ of weight t that reaches the syndrome $\mathbf{s}$ with respect to $\mathbf{H}$. We are interested in its probability of success:

$$\varepsilon = \mathbb{P}_{\mathbf{H},\mathbf{x},\mathbf{w}}\left(\mathscr{A}(\mathbf{H}, \mathbf{s} = \mathbf{x}\mathbf{H}^\intercal, \mathbf{w}) = \mathbf{e} \text{ s.t } |\mathbf{e}| = t \text{ and } \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\right)$$

where the probability is computed over the internal coins of $\mathscr{A}$ and $\mathbf{H}$ (*resp.* $\mathbf{x}$) being uniformly distributed over $\mathbb{F}_q^{(n-k)\times n}$ (*resp.* words of Hamming weight t in $\mathbb{F}_q^n$). This leads us to say that $\mathscr{A}$ solves the decoding problem in *average* time

$$T/\varepsilon.$$

In Chapter 3 we will study algorithms solving this problem and in each case their complexity will be written as some T/ε.

REMARK 1.2.5. *We have spoken of "average time complexity", it comes from the fact that* ε *is the average success probability of* $\mathscr{A}$ *over all its possible inputs. By using the law of total probability it can be verified that:*

$$\begin{aligned}\varepsilon &= \frac{1}{q^{(n-k)n}\,(q-1)^t\binom{n}{t}} \sum_{\substack{\mathbf{H}\in\mathbb{F}_q^{(n-k)\times n}\\ |\mathbf{x}|=t}} \mathbb{P}_{\mathbf{w}}\left(\mathscr{A}(\mathbf{H}, \mathbf{s} = \mathbf{x}\mathbf{H}^\intercal, \mathbf{w}) = \mathbf{e} \ s.t\ |\mathbf{e}| = t \ and\ \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\right)\\ &= \mathbb{E}_{\mathbf{H},\mathbf{x}}\left(\mathbb{P}_{\mathbf{w}}\left(\mathscr{A}(\mathbf{H}, \mathbf{s} = \mathbf{x}\mathbf{H}^\intercal, \mathbf{w}) = \mathbf{e} \ s.t\ |\mathbf{e}| = t \ and\ \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\right)\right)\end{aligned}$$

In particular we are interested in the probability to solve the decoding problem in average over all $[n,k]_q$*-codes.*

DP is a problem parametrized by n and two functions of n: R and τ. In the overwhelming majority of cryptographic applications the rate $R \in (0,1)$ is chosen as a constant. But it may be also interesting to consider the case where $R \underset{n\to+\infty}{\longrightarrow} 0$. Actually this regime of parameters is basically the LPN problem that will be discussed at the end of this subsection. Considering now the other parameter τ, that we will call the

relative decoding distance, many choices can be made but this greatly varies the difficulty DP. For instance, when $\tau = O(\log n/n)$, there is at most a polynomial number of errors of weight τn and a simple enumeration is enough to solve DP in polynomial time (over n). But surprisingly there are also many other non-trivial regimes of parameters for which DP can be solved in polynomial time. We will see in Chapter 3 that $\mathsf{DP}(n, q, R, \tau)$ can be solved in polynomial time as soon as $\tau \in \left[(1-R)\,\frac{q-1}{q}, R + (1-R)\,\frac{q-1}{q}\right]$. However, despite many efforts, the best algorithms to solve DP (even after 70 years of research) are all exponential in τn for other relative distances τ, namely $T/\varepsilon \underset{n\to+\infty}{=} 2^{n\tau\,(\alpha(q,R,\tau)+o(1))}$ for some function $\alpha(q, R, \tau)$ which depends of the used algorithm $\mathscr{A}$, q, R and τ. Situation is depicted in Figure 3.1.

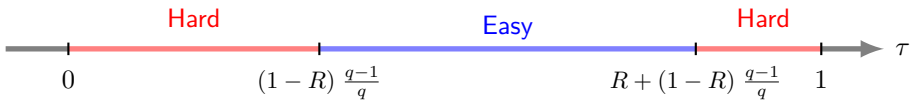


FIGURE 1.1. Hardness of $\mathsf{DP}(n, q, R, \tau)$ as function of τ.

$\mathsf{DP}(n, q, R, \tau)$ is hard in average but for well chosen relative distances τ. Therefore anyone who wants to design a crypto-system whose security relies on the hardness of solving DP has to carefully choose τ (in most cases the choice is constrained by the design itself). We list below some choices that have been made according to the designed (asymmetric) primitive:

- McEliece encryption [**McE78**]: $\tau = \Theta\left(\frac{1}{\log n}\right)$,
- Encryption schemes [**Ale03, MTSB13, AAB+17**]: $\tau = \Theta\left(\frac{1}{\sqrt{n}}\right)$,
- Authentication protocol [**Ste93**]: $\tau = C$ for some constant C quite small,
- Signature [**DST19**]: $\tau = C$ for some constant C large, $C \approx 0.95$.

The Learning Parity with Noise Problem. In the cryptographic literature, a problem closely related to DP and referred to as Learning Parity with Noise (LPN) is sometimes considered. It is a problem where is given as input an oracle that is function of some secret quantity. The aim is then to recover this secret but with as many samples as wanted (outputs of the oracle).

DEFINITION 1.2.2 (LPN-oracle). *Let $k \in \mathbb{N}$, $\tau \in [0, 1/2)$ and $\mathbf{s} \in \mathbb{F}_2^k$. We define the $\mathsf{LPN}(k, \tau)$-oracle $\mathscr{O}_{\mathbf{s},\tau}^{\mathsf{LPN}}$ as follows: on a call it outputs $(\mathbf{a}, \langle \mathbf{s}, \mathbf{a}\rangle + e)$ where $\mathbf{a} \leftarrow \mathbb{F}_2^k$ is uniformly distributed and e being distributed according to a Bernoulli of parameter τ.*

PROBLEM 1.2.6 (Learning with Parity Noise Problem - $\mathsf{LPN}(k, \tau)$).

- Input: *$\mathscr{O}_{\mathbf{s},\tau}^{\mathsf{LPN}}$ be an $\mathsf{LPN}(k, \tau)$-oracle parametrized by $\mathbf{s} \in \mathbb{F}_2^k$ which has been chosen uniformly at random.*
- Output: $\mathbf{s}$.

Let us stress that anyone who wants to solve this problem can ask as many samples (outputs of $\mathscr{O}_{\mathbf{s},\tau}^{\mathsf{LPN}}$) as he wants. However, each call to the oracle costs one. All the game consists in finding efficient algorithms that solves $\mathsf{LPN}(k, \tau)$ with as few queries as possible. Notice that the difficulty greatly varies with τ. The noise parameter τ deeply affects the gain of information on $\mathbf{s}$ that we obtain with each sample.

When $\tau = 0$, it is necessary to make at least k queries and then to solve a square linear system which has a complexity roughly given by k^3. On the other hand, when $\tau \in (0, 1)$ is some constant, best algorithms [**BKW03**] have a sub-exponential time complexity $2^{O(k/\log_2 k)}$ and for them the number of queries is

roughly the running time.

LPN: a special case of DP. It turns out that solving $\mathsf{LPN}(k,\tau)$ with n samples basically corresponds to solving $\mathsf{DP}(n,2,R,\tau)$ where $R = k/n$. Therefore, as the number of samples n is a priori unlimited, LPN really amounts to solve DP where the rate can be chosen arbitrarily close to 0.

Suppose that an algorithm asks for n samples to solve $\mathsf{LPN}(k,\tau)$, here $\mathscr{O}^{\mathsf{LPN}}_{\mathbf{s},\tau}$ outputs the sequence:

$$\mathbf{s}\cdot\mathbf{a}_1 + e_1,\ \ \ldots,\ \mathbf{s}\cdot\mathbf{a}_n + e_n.$$

These n samples can be rewritten as $\mathbf{sG}+\mathbf{e}$ where columns of $\mathbf{G}\in\mathbb{F}_2^{k\times n}$ are the $\mathbf{a}_i$'s and $\mathbf{e}\stackrel{\text{def}}{=}(e_1,\ldots,e_n)$. Now notice that $\mathbb{E}(|\mathbf{e}|)=\tau n$ as each e_i is a Bernoulli distribution of parameter τ. The algorithm that recovers $\mathbf{s}$ and thus $\mathbf{e}$ decodes at distance $|\mathbf{e}|$ the code of generator matrix $\mathbf{G}$. It corresponds to solve $\mathsf{DP}\left(n,q,\frac{k}{n},\frac{|\mathbf{e}|}{n}\right)$ where is given as input a matrix $\mathbf{H}\in\mathbb{F}_q^{(n-k)\times n}$ such that $\mathbf{GH}^\intercal=\mathbf{0}$ and the syndrome $\mathbf{eH}^\intercal$.

REMARK 1.2.6. DP *could have been presented directly with generator matrix representation. For more details see Chapter 2, in particular Exercise 2.2.1.*

1.2.3. Search to Decision Reduction. It is common in cryptography to consider for a same problem two variants: search or decision/distinguish. Roughly speaking, for some one-way function f (easy to compute but hard to invert) we ask in the search version given $f(x)$ to recover x while in the decision version we ask to distinguish between $f(x)$ and a uniform string. Obviously, the decision version is easier and therefore to rely a cryptosystem security on the hardness of some decision problem instead of its search counterpart is a strongest assumption to make. It turns out that Diffie-Hellman [**DH76**] or El Gamal [**ElG84**] primitives rely on this kind of assumption. But as we will see in this subsection, constructions based on codes do not suffer from this flaw, it has been shown in [**FS96**], through a reduction that the decision and search versions of the decoding problem are equivalent. The interesting direction has been to show that if there is an algorithm solving the decision version, then there is an algorithm that solves (in essentially the same time) the search part. We call such a result *a search-to-decision reduction.*

However it may be tempting to say that obtaining a search-to-decision reduction for the decoding problem is "only interesting" but not crucial for any security guarantee. This is not true and to see this let us present Alekhnovich scheme [**Ale03**], which is after McEliece scheme the second way of building encryption schemes based on codes and the decoding problem.

Alekhnovich encryption scheme. By contrast with McEliece's idea, Alekhnovich did not seek to build a public key encryption scheme based on the use of a decoding algorithm as a secret key. He proposed to start from a code $\mathcal{C}$ of length n for which we do not necessarily have an efficient decoding algorithm. The public key in Alekhnovich scheme is defined as $(\mathcal{C},\mathbf{c}+\mathbf{e})$ where $\mathbf{c}\in\mathcal{C}$ and $|\mathbf{e}|\ll n$ while the secret key is $\mathbf{e}$. Now if someone wants to encrypt *some bit* $b\in\{0,1\}$ into $\mathsf{Enc}(b)$ he proceeds as follows:

- $\mathsf{Enc}(1)\stackrel{\text{def}}{=}\mathbf{u}$ where $\mathbf{u}$ is a uniform vector,
- $\mathsf{Enc}(0)\stackrel{\text{def}}{=}\mathbf{c}^*+\mathbf{e}'$ where $|\mathbf{e}'|\ll n$ and $\mathbf{c}^*$ belongs to the dual of the code spanned by $\mathcal{C}$ and $\mathbf{c}+\mathbf{e}$.

Now to decrypt we just compute the inner product $\mathsf{Enc}(b)\cdot\mathbf{e}$. The correction of this procedure relies on the fact that

$$\mathbf{e}\cdot\mathsf{Enc}(0)=\mathbf{e}\cdot(\mathbf{c}^*+\mathbf{e}')=\mathbf{e}\cdot\mathbf{e}',$$

where in the last equality we used that $\mathbf{e}$ belongs to the code spanned by $\mathcal{C}$ and $\mathbf{c}+\mathbf{e}$ while $\mathbf{c}^*$ is in its dual. But now, with a high probability, $\mathbf{e}\cdot\mathbf{e}'=0$ as both vectors have a very small hamming weight ($\ll n$). On the other hand, $\mathbf{e}\cdot\mathsf{Enc}(1)=\mathbf{e}\cdot\mathbf{u}$ will be a uniform bit. Therefore, to securely send a bit b, it is enough to

repeat this procedure a small amount of times and to choose the most likely input according to the most probable outcome.

Notice now that a natural strategy for an adversary to decrypt is to distinguish between $\mathsf{Enc}(1)$ and $\mathsf{Enc}(0)$, namely a uniform string and a noisy codeword. Therefore the security of Alekhnovich scheme critically relies on the decision/distinguish version of the decoding problem.

Our aim now is to show how to obtain a search-to-decision reduction for the decoding problem. However to explain how to get this result, let us come back to the viewpoint with one-way functions. Let $\mathscr{A}$ be an algorithm that can distinguish between a random string u and $f(x)$ be some one-way function f. Given $f(x_0)$ we would like to use $\mathscr{A}$ to glean some information about x_0. A natural idea is to disturb a little bit $f(x_0)$ and to feed $\mathscr{A}$ with it with the hope, when looking at its answer, to gain some information on x_0 after repeating a small amount of times the operation. Here the key is the meaning of "information", we have to be careful about this. For instance, does it make sense to have a direct proposition like: if given $\mathscr{A}$ and $f(x_0)$ we can deduce a bit of x_0 then we are able to invert f? In fact not. Given f, the following function $g(b,x) = (b, f(x))$ is also a one-way function but its first input bit is always revealed. In other words, to hope to be able to invert f, we need to obtain another information than obtaining directly an input bit from $\mathscr{A}$, but which information? A first answer to this question has been given in [**Gol01**, Proposition 2.5.4]. Roughly speaking, it has been proven that if someone can extract from $f(x)$ and a uniform string r the value $x \cdot r$, then one can invert f.

PROPOSITION 1.2.2 ([**GL89**, **Gol01**]). *Let $f : \{0,1\}^* \to \{0,1\}^*$, $\mathscr{A}$ be a probabilistic algorithm running in time $T(n)$ and $\varepsilon(n) \in (0,1)$ be such that*

$$\mathbb{P}\left(\mathscr{A}(f(\mathbf{x}_n), \mathbf{r}_n) = \mathbf{x}_n \cdot \mathbf{r}_n\right) = \frac{1}{2} + \varepsilon(n)$$

where the probability is computed over the internal coins of $\mathscr{A}$, $\mathbf{x}_n$ and $\mathbf{r}_n$ that are uniformly distributed over $\{0,1\}^n$. Let $\ell(n) \stackrel{def}{=} \log(1/\varepsilon(n))$. Then, it exists an algorithm $\mathscr{A}'$ running in time $O\left(n^2\ell(n)^3T(n)\right)$ that satisfies

$$\mathbb{P}\left(\mathscr{A}'(f(\mathbf{x}_n) = \mathbf{x}_n)\right) = \Omega\left(\varepsilon(n)^2\right)$$

where the probability is computed over the internal coins of $\mathscr{A}'$ and $\mathbf{x}_n$.

PROOF. A nice proof of this proposition can be found here https://www.math.u-bordeaux.fr/~gzemor/alekhnovich.pdf □

REMARK 1.2.7. *Interestingly, the proof of this proposition relies on the use of linear codes (and their associated decoding algorithm) that are some Reed-Muller like codes of order one* [**MS86**, Chapitre 13].

This proposition will be at the core of the search-to-decision reduction of the decoding problem. Let us start by the formal definition of the decision decoding problem.

PROBLEM 1.2.7 (Decision Decoding Problem - $\mathsf{DDP}(n,q,R,\tau)$). *Let $k \stackrel{def}{=} \lfloor Rn \rfloor$ and $t \stackrel{def}{=} \lfloor \tau n \rfloor$.*

- Distributions:
  - $\mathscr{D}_0$ *:* $(\mathbf{H}, \mathbf{s})$ *be uniformly distributed over* $\mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$,
  - $\mathscr{D}_1$ *:* $(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal)$ *where* $\mathbf{H}$ *(resp.* $\mathbf{x}$*) being uniformly distributed over* $\mathbb{F}_q^{(n-k)\times n}$ *(resp. words of Hamming weight t).*
- Input: $(\mathbf{H}, \mathbf{s})$ *distributed according to* $\mathscr{D}_b$ *where* $b \in \{0,1\}$ *is uniform,*
- Decision: $b' \in \{0,1\}$.

A first, but trivial, way to solve DDP would be to output a random bit b'. It would give the right solution with probability $1/2$ which is not very interesting. The efficiency of an algorithm solving this problem is measured by the difference between its probability of success and $1/2$. This quantity is the right one to consider and is defined as the *advantage*.

DEFINITION 1.2.3. *The* $\mathsf{DDP}(n,q,R,\tau)$*-advantage of an algorithm* $\mathscr{A}$ *is defined as:*

$$Adv^{\mathsf{DDP}(n,q,R,\tau)}(\mathscr{A}) \stackrel{def}{=} \frac{1}{2}\left(\mathbb{P}\left(\mathscr{A}(\mathbf{H},\mathbf{s})=1 \mid b=1\right) - \mathbb{P}\left(\mathscr{A}(\mathbf{H},\mathbf{s})=1 \mid b=0\right)\right) \tag{1.4}$$

where the probabilities are computed over the internal randomness of $\mathscr{A}$*, a uniform* $b \in \{0,1\}$ *and inputs be distributed according to* $\mathscr{D}_b$ *which is defined in* $\mathsf{DDP}(n,q,R,\tau)$ *(Problem 1.2.7).*

For the sake of simplicity we will omit the dependence in the parameters (n,q,R,τ).

EXERCISE 1.2.4. *Prove that when* $(\mathbf{H},\mathbf{s})$ *is distributed according to* $\mathscr{D}_b$ *(for a fixed* $b \in \{0,1\}$*) we have:*

$$\mathbb{P}\left(\mathscr{A}(\mathbf{H},\mathbf{s})=b\right) = \frac{1}{2} + Adv^{\mathsf{DDP}}(\mathscr{A}).$$

Our aim now is to prove the following theorem which shows how an algorithm solving DDP can be turned into an algorithm solving DP. More precisely, we will show how to turn $\mathscr{A}$ with advantage $Adv^{\mathsf{DDP}}(\mathscr{A})$ into an algorithm that computes $\mathbf{x}\cdot\mathbf{r}$ with probability $1/2 + Adv^{\mathsf{DDP}}(\mathscr{A})$ given as input $\mathbf{x}\mathbf{H}^\intercal$ and $\mathbf{r}$. To conclude it will simply remain to apply Proposition 1.2.2.

THEOREM 1.2.1. *Let* $\mathscr{A}$ *be a probabilistic algorithm running in time* $T(n)$ *whose* $\mathsf{DDP}(n,2,R,\tau)$*-advantage is given by* $\varepsilon(n)$ *and let* $\ell(n) \stackrel{def}{=} \log(1/\varepsilon(n))$*. Then it exists an algorithm* $\mathscr{A}'$ *that solves* $\mathsf{DDP}(n,2,R,\tau)$ *in time* $O(n^2\ell(n)^3)T(n)$ *and with probability* $\Omega(\varepsilon(n)^2)$*.*

REMARK 1.2.8. *Theorem 1.2.1 is stated for binary codes. However it can be extended to* q*-ary codes by using a generalization of Proposition 1.2.2 proved in* [**GRS00**].

PROOF OF THEOREM 1.2.1. Let $(\mathbf{H},\mathbf{s} \stackrel{\text{def}}{=} \mathbf{x}\mathbf{H}^\intercal)$ be an instance of $\mathsf{DP}(n,q,R,\tau)$. In what follows, $\mathscr{A}'$ is an algorithm such that on input $(\mathbf{H},\mathbf{x}\mathbf{H}^\intercal,\mathbf{r})$ it outputs $\mathbf{x}\cdot\mathbf{r}$ with probability $1/2+\varepsilon$. To end the proof it will be enough to apply Proposition 1.2.2.

Algorithm $\mathscr{A}'$:

Input : $(\mathbf{H},\mathbf{s}) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k}$ and $\mathbf{r} \in \mathbb{F}_2^n$,

1. $\mathbf{u} \in \mathbb{F}_2^{n-k}$ be uniformly distributed
2. $\mathbf{H}' \stackrel{\text{def}}{=} \mathbf{H} - \mathbf{u}^\intercal\mathbf{r}$
3. $b \stackrel{\text{def}}{=} \mathscr{A}(\mathbf{H}',\mathbf{s})$

Output : b

The matrix $\mathbf{H}$ is uniformly distributed by definition, therefore $\mathbf{H}'$ is also uniformly distributed. Notice now,

$$\mathbf{s} = \mathbf{x}\mathbf{H}^\intercal = \mathbf{x}\mathbf{H}'^\intercal + (\mathbf{x}\cdot\mathbf{r})\,\mathbf{u}.$$

Let,

$$\mathbf{s}' \stackrel{\text{def}}{=} \mathbf{x}\mathbf{H}'^\intercal + \mathbf{u}.$$

It is readily verified that $\mathbf{s}'$ is uniformly distributed. Therefore, according to $b = \mathbf{x} \cdot \mathbf{r} \in \{0, 1\}$, we obtain distributions of DDP. The probability that $\mathscr{A}'$ outputs $\mathbf{x} \cdot \mathbf{r}$ is given by:

$$\begin{aligned}
\mathbb{P}(\mathscr{A}'(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal, \mathbf{r}) = \mathbf{x} \cdot \mathbf{r}) &= \frac{1}{2}\mathbb{P}\left(\mathscr{A}'(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal, \mathbf{r}) = 0 \mid \mathbf{r} \cdot \mathbf{x} = 0\right) + \frac{1}{2}\mathbb{P}\left(\mathscr{A}'(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal, \mathbf{r}) = 1 \mid \mathbf{r} \cdot \mathbf{x} = 1\right) && (1.5)\\
&= \frac{1}{2}\left(\mathbb{P}\left(\mathscr{A}(\mathbf{H}', \mathbf{x}\mathbf{H}'^\intercal) = 0\right) + \mathbb{P}\left(\mathscr{A}(\mathbf{H}', \mathbf{s}') = 1\right)\right)\\
&= \frac{1}{2} + \frac{1}{2}\left(\mathbb{P}\left(\mathscr{A}(\mathbf{H}', \mathbf{s}') = 1\right) - \mathbb{P}\left(\mathscr{A}(\mathbf{H}', \mathbf{x}\mathbf{H}'^\intercal) = 1\right)\right)\\
&= \frac{1}{2} + \varepsilon && (1.6)
\end{aligned}$$

where we used in (1.5) the fact that $\mathbf{r}$ is uniformly distributed and in (1.6) the DDP-advantage definition. □

CHAPTER 2

Random Codes

Introduction

We study in this course *random codes*, *i.e.* codes whose parity-check or generator matrix is drawn uniformly at random. However, in light of the history of error correcting codes which has consisted in finding codes becoming more and more complex and structured, one may wonder but why then are we wasting our time to study random codes? That may come as a surprise but random codes enlighten about what we could expect or not in a simple fashion, and even better what is optimal or not. The most prominent example of the interest of random codes is the famous Shannon theorem about the capacity of some "noisy channels". Roughly speaking, Shannon gave (for some error models) the maximum amount of errors that "can" be theoretically decoded with codes of fixed rate. Shannon made his proof by using random codes and he has shown that they are precisely those which reach the optimality.

Our aim in these lecture notes is to study carefully these kind of codes and to show that they enable to answer many questions like for instance:

- How many vectors of Hamming weight t do we expect in a code?
- What is the typical minimum distance of a code?
- etc...

Our study will have an important consequence for cryptographic purposes: a better understanding of the Decoding Problem $\mathsf{DP}(n, q, R, \tau)$ that was defined in Chapter 1. We will be able to predict with a very good accuracy the number of solutions of this problem as a function of its parameters. This will be particularly helpful to understand the behaviour of algorithms solving it.

2.1. Prerequisites

Basic notation. In all these lecture notes, q will denote a fixed field size while R will be a *constant* in $(0, 1)$. On the other hand, $\tau(n)$ will denote any function of n taking its values in $(0, 1)$. To simplify notation, since n is clear from the context, we will drop the dependency in n and simply write τ. Furthermore, parameters k and t will always (even implicitly) be defined as $k \stackrel{\text{def}}{=} \lfloor Rn \rfloor$ and $t \stackrel{\text{def}}{=} \lfloor \tau n \rfloor$. A function $f(n)$ is said to be negligible, and we denote this by $f \in \text{negl}(n)$, if for all polynomial $p(n)$, $|f(n)| < |p(n)|^{-1}$ for all sufficiently large n.

Many asymptotic results will be given. As all our parameters are functions of n, our asymptotic results will always hold for:

$$n \longrightarrow +\infty.$$

The parameter n is in most cryptographic applications roughly given by several thousands.

The following function h_q, known as the *$q-$ary entropy*, will play an important role:

$$h_q : x \in [0, 1] \longmapsto -x \log_q \left(\frac{x}{q-1} \right) - (1-x) \log_q (1-x) \quad \text{(extended by continuity in 0 and 1).}$$

It is equal to the entropy of a random variable e over $\mathbb{F}_q$ distributed like the error for a $q-$ary symmetric channel of crossover probability x, *i.e.* $\mathbb{P}(e=0)=1-x$ and $\mathbb{P}(e=\alpha)=\frac{x}{q-1}$ for any $\alpha\in\mathbb{F}_q^\star$.

It can be verified that h_q is an increasing function over $\left[0,\frac{q-1}{q}\right]$ and a decreasing function over $\left[\frac{q-1}{q},1\right]$. The $q-$ary entropy is involved in the estimation of $\sharp\mathscr{S}_t$ where $\mathscr{S}_t$ is defined as the sphere of radius t for the Hamming distance $|\cdot|$, namely

$$\mathscr{S}_t \stackrel{\text{def}}{=} \left\{\mathbf{x}\in\mathbb{F}_q^n \ : \ |\mathbf{x}|=t\right\}.$$

The following elementary lemma will be at the core of most of our asymptotic results.

LEMMA 2.1.1. *Let* $t \stackrel{def}{=} \lfloor \tau n\rfloor$. *We have* $\sharp\mathscr{S}_t=\binom{n}{t}(q-1)^t$ *and*

$$q^{n\left(h_q(\tau)+O\left(\frac{\log_q(n)}{n}\right)\right)} \leq \binom{n}{t}(q-1)^t \leq q^{nh_q(\tau)}.$$

Asymptotically,

$$\frac{1}{n}\log_q\left(\binom{n}{t}(q-1)^t\right)=h_q(\tau)+O\left(\frac{\log_q n}{n}\right).$$

Probabilistic notation. During these lecture notes we wish to emphasize on which probability space the probabilities or the expectations are taken. Therefore we will denote by a subscript the random variable specifying the associated probability space over which the probabilities or expectations are taken. For instance the probability $\mathbb{P}_X(A)$ of the event A is taken over Ω the probability space over which the random variable X is defined, *i.e.* if X is for instance a real random variable, X is a function from a probability space Ω to $\mathbb{R}$, and the aforementioned probability is taken according to the measure chosen for Ω.

Statistical distance. An essential tool for many cryptographic applications is the *statistical distance*, sometimes called the *total variational distance*. It is a distance for probability distributions, which in the case where X and Y are two random variables taking their values in a same finite space $\mathscr{E}$ is defined as

$$\Delta(X,Y) \stackrel{\text{def}}{=} \frac{1}{2}\sum_{a\in\mathscr{E}}\left|\mathbb{P}\left(X=a\right)-\mathbb{P}\left(Y=a\right)\right|. \tag{2.1}$$

An equivalent definition is given by

$$\Delta\left(X,Y\right) \stackrel{\text{def}}{=} \max_{A\subseteq\mathscr{E}}\left|\mathbb{P}_X(A)-\mathbb{P}_Y(A)\right|. \tag{2.2}$$

Depending on the context, (2.1) or (2.2) is the most useful. A direct consequence of (2.2) is that given any event A, we have $|\mathbb{P}_X(A)-\mathbb{P}_Y(A)|\leq\Delta(X,Y)$. Therefore, computing probabilities over X or Y will differ by at most $\Delta(X,Y)$. Furthermore, given a single observation, coming from X or Y with probability $1/2$, we will be able to guess which with probability at most $1/2+\Delta(X,Y)/2$ and there is a strategy to reach this probability of guessing correctly.

The statistical distance enjoys many interesting properties. Among others, it cannot increase by applying a function f,

$$\Delta(f(X),f(Y))\leq\Delta(X,Y) \quad (\textit{data processing inequality}).$$

The function f can be randomized, but its internal randomness has to be independent from X and Y for the data processing inequality to hold. In particular, it implies that the "success" probability of any algorithm $\mathscr{A}$ for inputs distributed according to X or Y, can only differ by at most $\Delta(X,Y)$.

In our applications we will focus on distributions X, Y such that their statistical distance is negligible. It will show (as a consequence of the data processing inequality) that X and Y are computationally indistinguishable[(1)] without requiring any computational argument with a reduction.

One can define various other distances for capturing in a cryptographic context the differences between two distributions. For instance, we can cite the family Renyi divergences, but this is out of the scope of these lecture notes.

2.2. Random Codes

The model of random codes. In these lecture notes we will use two probabilistic models that will be referred to as *random $[n,k]_q$-codes*. The first one is by choosing a code $\mathcal{C}$ by picking uniformly at random a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ (*i.e.* $\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{mG} : \mathbf{m} \in \mathbb{F}_q^k\right\}$). However, all the probabilistic results of these lecture notes are easier to prove if, instead, we choose $\mathcal{C}$ by picking uniformly at random a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ (*i.e.* $\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{Hc}^\intercal = \mathbf{0}\right\}$). We will denote $\mathbb{P}_{\mathbf{G}}$ and $\mathbb{P}_{\mathbf{H}}$ respectively the probabilities in these two models.

It may be pointed out that in both models we don't pick uniformly at random an $[n,k]_q$-code. Indeed, the first model always produces codes of dimension $\leq k$ whereas in the second model codes are always of dimension $\geq k$. One may wonder why don't we pick $\mathbf{G}$ (*resp.* $\mathbf{H}$) uniformly at random among the $k \times n$ (*resp.* $(n-k)\times n$) matrices of rank k (*resp.* $n-k$)? First, computations are much more complicated in this "exact" model. Furthermore, it turns out that it is pointless. Roughly speaking, the $\mathbf{G}$-model produces codes of dimension $= k$ with probability $1 - O(q^{-(n-k)})$ while in the $\mathbf{H}$-model we get a code of dimension $= k$ with probability $1 - O(q^{-k})$. As shown in the following lemma this result can even be expressed in a stronger way, our probabilistic models are exponentially close, *for the statistical distance*, to the "exact" model. Therefore all our computations in the $\mathbf{G}$ or $\mathbf{H}$ models can really be thought as by picking uniformly at random an $[n,k]_q$-code $\mathcal{C}$.

LEMMA 2.2.1. *Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ (resp. $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$) be a uniformly random matrix and $\mathbf{G}_k \in \mathbb{F}_q^{k\times n}$ (resp. $\mathbf{H}_{n-k} \in \mathbb{F}_q^{(n-k)\times n}$) be a uniformly random matrix of rank k (resp. $n-k$). We have:*

$$\Delta\left(\mathbf{G}, \mathbf{G}_k\right) = O\left(q^{-(n-k)}\right) \quad \left(\textit{resp. } \Delta\left(\mathbf{H}, \mathbf{H}_{n-k}\right) = O\left(q^{-k}\right)\right).$$

PROOF. Let us prove the lemma for $(\mathbf{G}, \mathbf{G}_k)$, the other case will be similar. First it is a classical fact that the density of rank k matrices among $\mathbb{F}_q^{k\times n}$ is equal to $1 - O\left(q^{-(n-k)}\right)$. Therefore, given some rank k matrix $\mathbf{R} \in \mathbb{F}_q^{k\times n}$, we have:

$$\mathbb{P}\left(\mathbf{G}_k = \mathbf{R}\right) = \frac{1}{q^{k\times n}\left(1 - O(q^{-(n-k)})\right)}.$$

It leads to the following computation:

$$\begin{aligned}
2\Delta\left(\mathbf{G}, \mathbf{G}_k\right) &= \sum_{\substack{\mathbf{R}\in\mathbb{F}_q^{k\times n}\\ \text{rank}(\mathbf{R})=k}} \left|\mathbb{P}(\mathbf{G}=\mathbf{R}) - \mathbb{P}(\mathbf{G}_k=\mathbf{R})\right| + \sum_{\substack{\mathbf{R}\in\mathbb{F}_q^{k\times n}\\ \text{rank}(\mathbf{R})\neq k}} \mathbb{P}(\mathbf{G}=\mathbf{R}) \\
&= \sum_{\substack{\mathbf{R}\in\mathbb{F}_q^{k\times n}\\ \text{rank}(\mathbf{R})=k}} \left|\frac{1}{q^{k\times n}}\left(1 - \frac{1}{1-O\left(q^{-(n-k)}\right)}\right)\right| + \sum_{\substack{\mathbf{R}\in\mathbb{F}_q^{k\times n}\\ \text{rank}(\mathbf{R})\neq k}} \frac{1}{q^{k\times n}} \\
&= O\left(q^{-(n-k)}\right)
\end{aligned}$$

(1) See here for a definition: https://www.cs.princeton.edu/courses/archive/spr10/cos433/lec4.pdf

which concludes the proof. □

Now one may wonder why do we consider two models for random $[n,k]_q$-codes? It turns out that depending of the context, computations might be easier and/or more natural in one model rather than in the other one. In addition, for the same reasons as those given in the previous lemma, $\mathbf{G}$ and $\mathbf{H}$ models are closely related, computations in both probabilistic models will outcome the same results up to an additive exponentially small factor.

LEMMA 2.2.2. *Let $\mathscr{E}$ be a set of linear codes of length n in $\mathbb{F}_q$ which is defined as an event. We have,*

$$|\mathbb{P}_{\mathbf{G}}(\mathscr{E}) - \mathbb{P}_{\mathbf{H}}(\mathscr{E})| = O\left(q^{-\min(k,n-k)}\right).$$

PROOF. Let $\mathbf{G}_k$ and $\mathbf{H}_{n-k}$ be defined as in Lemma 2.2.1. Notice that $\mathbb{P}_{\mathbf{G}_k}(\mathscr{E}) = \mathbb{P}_{\mathbf{H}_{n-k}}(\mathscr{E})$, in both models, we exactly pick uniformly at random an $[n,k]_q$-code. It leads to the following computation:

$$\begin{aligned}|\mathbb{P}_{\mathbf{G}}(\mathscr{E}) - \mathbb{P}_{\mathbf{H}}(\mathscr{E})| &\leq |\mathbb{P}_{\mathbf{G}}(\mathscr{E}) - \mathbb{P}_{\mathbf{G}_k}(\mathscr{E})| + \left|\mathbb{P}_{\mathbf{H}_{n-k}}(\mathscr{E}) - \mathbb{P}_{\mathbf{H}}(\mathscr{E})\right| \\ &\leq \Delta(\mathbf{G},\mathbf{G}_k) + \Delta(\mathbf{H},\mathbf{H}_{n-k})\end{aligned}$$

where in the last line we used Equation (2.2). It concludes the proof by using Lemma 2.2.1. □

EXERCISE 2.2.1. *Let us introduce the following variant of* DP *(see Problem 1.2.5) with generator matrices instead of parity-check matrices*

$\mathsf{DP}'(n,q,R,\tau)$. *Let $k \stackrel{def}{=} \lfloor Rn \rfloor$ and $t \stackrel{def}{=} \lfloor \tau n \rfloor$.*

- Input: $(\mathbf{G}, \mathbf{y} \stackrel{def}{=} \mathbf{sG} + \mathbf{x})$ *where $\mathbf{G}, \mathbf{s}$ and $\mathbf{x}$ are uniformly distributed over $\mathbb{F}_q^{k\times n}$, $\mathbb{F}_q^k$ and words of Hamming weight t in $\mathbb{F}_q^n$.*
- Output: *an error $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight t such that $\mathbf{y} - \mathbf{e} = \mathbf{mG}$ for some $\mathbf{m} \in \mathbb{F}_q^k$.*

Show that for any algorithm $\mathscr{A}$ solving this problem with probability ε and time T, there exists an algorithm $\mathscr{B}$ which solves $\mathsf{DP}(n,q,R,\tau)$ in time $O\left(n^3 + T\right)$ with probability $\geq \varepsilon - O\left(q^{-\min(k,n-k)}\right)$. Show that we can exchange DP' by DP *in the previous question.*

REMARK 2.2.1. *The above exercise shows that defining* DP *with generator or parity-check matrices is just a matter of personal taste, it does not change the average hardness.*

A first computation with random codes. Now that random codes are well defined, we are ready to make our first computation in this probabilistic model. The following elementary lemma gives the probability (over the codes) that a fixed non-zero word $\mathbf{y}$ reaches some syndrome $\mathbf{s}$ according to the code. In particular, by setting $\mathbf{s}$ to $\mathbf{0}$, we obtain the probability that $\mathbf{y}$ belongs to the code. This lemma will be at the core of all our results about random codes.

LEMMA 2.2.3. *Given $\mathbf{s} \in \mathbb{F}_q^{n-k}$ and $\mathbf{y} \in \mathbb{F}_q^n$ such that $\mathbf{y} \neq \mathbf{0}$, we have for $\mathbf{H}$ being uniformly distributed at random in $\mathbb{F}_q^{(n-k)\times n}$,*

$$\mathbb{P}_{\mathbf{H}}(\mathbf{yH}^\intercal = \mathbf{s}) = \frac{1}{q^{n-k}}.$$

PROOF. Let $h_{i,j}$ be the coefficient of $\mathbf{H}$ at position (i,j). Without loss of generality, we can suppose that $y_1 = 1$ (by permuting $\mathbf{y}$ if $y_1 = 0$ and then multiplying by y_1^{-1} which is possible as we work in $\mathbb{F}_q$). The probability that we are looking for is the probability of the following event:

$$\forall i \in [\![1, n-k]\!], \quad h_{i,1} = s_i - \sum_{j=2}^{n} h_{i,j} y_j$$

Recall that $\mathbf{H}$ is uniformly distributed: the $h_{i,j}$'s are independent and equidistributed. Therefore the above $n-k$ equations will be independently true with probability $1/q$ which concludes the proof. □

EXERCISE 2.2.2. *Show that for any non-zero* $\mathbf{y} \in \mathbb{F}_q^n$,

$$\mathbb{P}_{\mathbf{G}}(\mathbf{y} \in \mathcal{C}^*) = \frac{1}{q^k}.$$

2.3. Weight Distribution of Cosets of Random Codes

The aim of this section is to answer the following question: given a random code $\mathcal{C}$ and a fixed vector $\mathbf{y} \in \mathbb{F}_q^n$, how many codewords $\mathbf{c} \in \mathcal{C}$ do we expect to be at Hamming distance t from $\mathbf{y}$? Or equivalently, given a parity-check matrix of our random code $\mathcal{C}$ and a fixed syndrome $\mathbf{s}$, how many vectors $\mathbf{e}$ of Hamming weight t do we expect to reach the syndrome $\mathbf{s}$ according to $\mathbf{H}$? Notice that deriving an answer to these questions in the particular cases $\mathbf{y} = \mathbf{0}$ and $\mathbf{s} = \mathbf{0}$ enables to compute the expected number of codewords of weight t in $\mathcal{C}$. It will be useful to compute the expected minimum distance of a code.

These results will have an important consequence: a better understanding of the Decoding Problem (DP) that was defined in Problem 1.2.5. According to our probabilistic model, this problem really corresponds to decode a random $[n,k]_q$-code of parity-check matrix $\mathbf{H}$. In that case it is natural to wonder how many vectors $\mathbf{e} \in \mathscr{S}_t$ are expected to reach the syndrome $\mathbf{s}$ according to $\mathbf{H}$, but why? To understand this let us take a toy example. A trivial solution to solve DP is to pick a random error $\mathbf{e} \in \mathscr{S}_t$ with the hope that it gives a solution. By definition there is a solution to our problem (here $\mathbf{x}$). If there is exactly one solution, our success probability is given by $\frac{1}{\binom{n}{t}(q-1)^t}$. But now imagine that we expect N solutions to our problem. In that case we would expect our success probability to be equal to $\approx \frac{N}{\binom{n}{t}(q-1)^t}$. It is therefore important to know the value of N to be able to predict the running time of our algorithm. It is the aim of what follows.

Notation. Given $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$, let

$$N_t(\mathcal{C}, \mathbf{s}) \stackrel{\text{def}}{=} \sharp\left\{\mathbf{e} \in \mathscr{S}_t : \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\right\}$$

where implicitly $\mathcal{C}$ is defined as $\{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{H}\mathbf{c}^\intercal = \mathbf{0}\}$. Notice that $N_t(\mathcal{C}, \mathbf{s})$ is a random variable that gives the number of solutions of DP with input $(\mathbf{H}, \mathbf{s})$ (where $\mathbf{s} \in \mathbb{F}_q^{n-k}$ is fixed and not necessarily computed as some $\mathbf{x}\mathbf{H}^\intercal$ for $\mathbf{x} \in \mathscr{S}_t$). On the other hand, $N_t(\mathcal{C}, \mathbf{0})$ is a random variable that gives the number of codewords $\mathbf{c} \in \mathcal{C}$ of Hamming weight t.

Expected weight distribution of cosets. From now on, our main objective is to compute the expected number of Hamming weight t vectors in a given coset, namely to compute the expectation of $N_t(\mathcal{C}, \mathbf{s})$ over $\mathcal{C}$. To avoid any suspense, we will prove that for any syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, the expectation of $N_t(\mathcal{C}, \mathbf{s})$ is given by $\binom{n}{t}(q-1)^t/q^{n-k}$. However, before showing this result, let us start to understand how this quantity behaves as function of it parameters, namely $\tau = t/n$ and $R = k/n$. By Lemma 2.1.1, we have

$$\frac{1}{n}\log_q \binom{n}{t}(q-1)^t/q^{n-k} = h_q(\tau) - (1-R) + O\left(\frac{\log_q n}{n}\right). \tag{2.3}$$

Recall now that $x \in [0,1] \mapsto h_q(x)$ is an increasing function over $\left[0, \frac{q-1}{q}\right]$ and a decreasing function over $\left[\frac{q-1}{q}, 1\right]$. Furthermore, $h_q(0) = 0$ and $h_q(1) = \log_q(q-1)$. This shows that $N_t(\mathcal{C}, \mathbf{s})$ is expected (according to τ) to be exponentially small or large (in n) at the exception of one value τ^- and potentially a second one in the case where $(1-R) \geq \log_q(q-1)$, that we will denote τ^+. We summarize the picture by drawing in Figure 2.1 the logarithm in basis 3 (for n large enough) of $\binom{n}{t}(q-1)^t/q^{n-k}$ when $q = 3$ and $k/n = 1/4$.

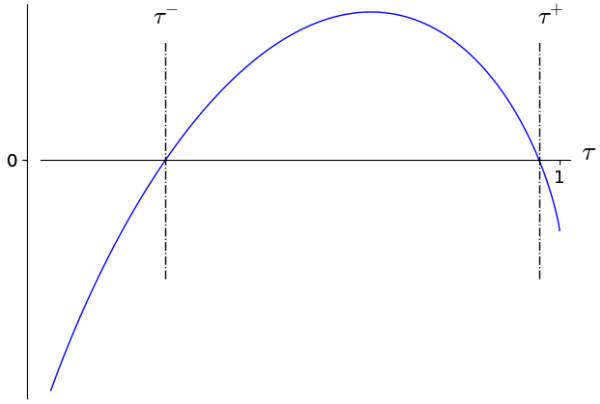


FIGURE 2.1. $\lim_{n\to+\infty} \frac{1}{n} \log_q \binom{n}{t}(q-1)^t/q^{n-k}$ when $q = 3$ and $k/n = 1/4$ as function of $\tau = t/n$.

It turns out that an analytic expression of τ^- and τ^+ can be given,

$$\tau^- \stackrel{\text{def}}{=} g_q^-(1-R) \quad \text{and} \quad \tau^+ \stackrel{\text{def}}{=} g_q^+(1-R) \text{ when } R \leq 1 - \log_q(q-1) \tag{2.4}$$

where g_q^- (*resp.* g_q^+) denotes the inverse of h_q over $\left[0, \frac{q-1}{q}\right]$ $\left(\textit{resp.} \left[\frac{q-1}{q}, 1\right]\right)$.

REMARK 2.3.1. *As we will see in Section 2.4, τ^- is commonly called the relative Gilbert-Varshamov distance or bound.*

Quantities τ^- and τ^+ give the boundaries between which we expect $N_t(\mathcal{C}, \mathbf{s})$ to be exponentially large as we show now.

PROPOSITION 2.3.1. *Let $k \stackrel{def}{=} \lfloor Rn \rfloor$, $t \stackrel{def}{=} \lfloor \tau n \rfloor$ and $\mathbf{s} \in \mathbb{F}_q^{n-k}$. We have:*

$$\mathbb{E}_{\mathbf{H}}(N_t(\mathcal{C}, \mathbf{s})) = \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}. \tag{2.5}$$

When $\tau \in \begin{cases} (\tau^-, \tau^+) & \text{if } R \leq 1 - \log_q(q-1) \\ (\tau^-, 1) & \text{otherwise} \end{cases}$, *we expect $N_t(\mathcal{C}, \mathbf{s})$ to be exponentially large:*

$$\mathbb{E}_{\mathbf{H}}(N_t(\mathcal{C}, \mathbf{s})) = q^{\alpha n(1+o(1))} \quad \text{where} \quad \alpha \stackrel{def}{=} h_q(\tau) - 1 + R > 0.$$

In the case where $\tau = \begin{cases} \tau^- \text{ or } \tau^+ & \text{if } R \leq 1 - \log_q(q-1) \\ \tau^- & \text{otherwise.} \end{cases}$, *the expectation of $N_t(\mathcal{C}, \mathbf{s})$ equals $P(n)(1+o(1))$ for some polynomial P.*

PROOF. Let $\mathbb{1}_{\mathbf{e}}$ be the indicator function of the event "$\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$". It is readily verified that by definition,

$$N_t(\mathcal{C}, \mathbf{s}) = \sum_{\mathbf{e} \in \mathscr{S}_t} \mathbb{1}_{\mathbf{e}}.$$

We have the following computation,

$$\begin{aligned}\mathbb{E}_{\mathbf{H}}(N_t(\mathcal{C},\mathbf{s})) &= \mathbb{E}_{\mathbf{H}}\left(\sum_{\mathbf{e}\in\mathscr{S}_t}\mathbb{1}_{\mathbf{e}}\right)\\ &= \sum_{\mathbf{e}\in\mathscr{S}_t}\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}}) \quad \text{(by linearity of the expectation)}\\ &= \sum_{\mathbf{e}\in\mathscr{S}_t}\mathbb{P}_{\mathbf{H}}\left(\mathbf{e}\mathbf{H}^{\intercal}=\mathbf{s}\right)\end{aligned}$$

which gives (2.5) by using Lemma 2.2.3. The second part of the proposition is a consequence of Lemma 2.1.1. □

EXERCISE 2.3.1. *Show that the average number of solutions of* $\mathsf{DP}(n,q,R,\tau)$ *(over the input distribution) is given by*

$$1+\frac{\binom{n}{t}(q-1)^t-1}{q^{n-k}}$$

REMARK 2.3.2. *Proposition 2.3.1 can actually be stated more generally. Given some set* $\mathscr{E}\subseteq\mathbb{F}_q^n$*, we can show that the expected number of vectors* $\mathbf{e}\in\mathscr{E}$ *that reach some syndrome for a random* $[n,k]_q$*-code is given by* $\sharp\mathscr{E}/q^{n-k}$*.*

REMARK 2.3.3. *The expected number of codewords of weight* t *in a random* $[n,k]_q$*-code is given by* $\binom{n}{t}(q-1)^t/q^{n-k}$ *(by setting* $\mathbf{s}$ *to* $\mathbf{0}$ *in Proposition 2.3.1). This statement, in the same manner as in the previous remark, can be generalized to give the expected number of codewords in any set* $\mathscr{E}\subseteq\mathbb{F}_q^n$*. In particular, it can be used to obtain the expected number of codewords of "weight"* t *that belong to a random code, for any notion of weight and therefore any metric.*

EXERCISE 2.3.2. *Show that* $(t>0)$*,*

$$\mathbb{E}_{\mathbf{G}}\left(\sharp\left\{\mathbf{m}\in\mathbb{F}_q^k:|\mathbf{m}\mathbf{G}|=t\right\}\right)=\frac{q^k-1}{q^n}\binom{n}{t}(q-1)^t \quad \textit{and} \quad \mathbb{E}_{\mathbf{H}}\left(\sharp\left\{\mathbf{c}\in\mathcal{C}:|\mathbf{c}| \textit{ is odd}\right\}\right)=\frac{1}{2}\,\frac{q^n-(2-q)^n}{q^{n-k}}.$$

Hint: For the first part of the exercise first show that $\mathbf{m}\mathbf{G}$ is uniformly distributed over $\mathbb{F}_q^n$ when $\mathbf{m}\in\mathbb{F}_q^k\backslash\{\mathbf{0}\}$.

At this point our work has given the *expected* number of solutions of $\mathsf{DP}(n,q,R,\tau)$. Situation is depicted in Figure 2.2.

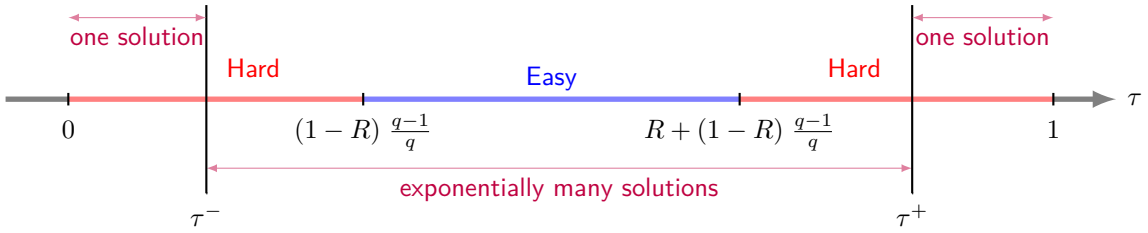


FIGURE 2.2. Hardness and expected number of solutions of $\mathsf{DP}(n,q,R,\tau)$ as function of τ.

However, can we be much more precise? For instance, can we give with an overwhelming probability, and therefore for almost all codes, the number of solutions of DP? The answer is yes. Below is given two techniques for achieving this, the first one uses Markov's inequality (*first moment technique*) and the second one, which is more accurate, uses Bienaymé-Tchebychev's inequality (*second moment technique*).

PROPOSITION 2.3.2 (First Moment Technique). *Let $\mathbf{s} \in \mathbb{F}_q^{n-k}$. For any $a > 0$, we have*

$$\mathbb{P}_{\mathbf{H}}(N_t(\mathcal{C}, \mathbf{s}) > a) \leq \frac{1}{a} \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}.$$

PROOF. By Proposition 2.3.1, $\mathbb{E}_{\mathbf{H}}(N_t(\mathcal{C}, \mathbf{s})) = \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}$. It remains to apply Markov's inequality to conclude the proof. □

Notice that Proposition 2.3.2 is not very accurate. One has to choose $a \gg \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}$ to obtain a negligible probability that $N_t(\mathcal{C}, \mathbf{s})$ is larger than a. But the expectation of $N_t(\mathcal{C}, \mathbf{s})$ is exactly given by $\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}$. A meaningful result would be

$$\mathbb{P}_{\mathbf{H}}\left(\left|N_t(\mathcal{C}, \mathbf{s}) - \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right| > a\right) < \varepsilon,$$

for some $\varepsilon \in \mathrm{negl}(n)$ and a be such that $a \in \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\mathrm{negl}(n)$. It is precisely the aim of the following proposition.

PROPOSITION 2.3.3 (Second Moment Technique). *Let $\mathbf{s} \in \mathbb{F}_q^{n-k}$. For any $a > 0$, we have*

$$\mathbb{P}_{\mathbf{H}}\left(\left|N_t(\mathcal{C}, \mathbf{s}) - \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right| \geq a\right) \leq \frac{(q-1)\binom{n}{t}(q-1)^t}{a^2 q^{n-k}}.$$

PROOF. Let $\mathbb{1}_{\mathbf{e}}$ be the indicator function of the event "$\mathbf{eH}^\intercal = \mathbf{s}$". By using Bienaymé-Tchebychevs inequality with the random variable $N_t(\mathcal{C}, \mathbf{s}) = \sum_{\mathbf{e} \in \mathscr{S}_t} \mathbb{1}_{\mathbf{e}}$, we obtain

$$\begin{aligned}
\mathbb{P}_{\mathbf{H}}\left(\left|N_t(\mathcal{C}, \mathbf{s}) - \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right| \geq a\right) &\leq \frac{\mathbf{Var}_{\mathbf{H}}(N_t(\mathcal{C}, \mathbf{s}))}{a^2} \\
&= \frac{1}{a^2}\left(\sum_{\mathbf{e} \in \mathscr{S}_t} \mathbf{Var}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}}) + \sum_{\substack{\mathbf{x},\mathbf{y} \in \mathscr{S}_t \\ \mathbf{x} \neq \mathbf{y}}} \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}) - \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}})\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{y}})\right) \\
&\leq \frac{1}{a^2}\left(\sum_{\mathbf{e} \in \mathscr{S}_t} \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}}) + \sum_{\substack{\mathbf{x},\mathbf{y} \in \mathscr{S}_t \\ \mathbf{x} \neq \mathbf{y}}} \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}) - \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}})\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{y}})\right) \\
&= \frac{1}{a^2}\left(\frac{\binom{n}{t}(q-1)^t}{q^{n-k}} + \sum_{\substack{\mathbf{x},\mathbf{y} \in \mathscr{S}_t \\ \mathbf{x} \neq \mathbf{y}}} \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}) - \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}})\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{y}})\right) \qquad (2.6)
\end{aligned}$$

where we used that $\mathbf{Var}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}}) \leq \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}}^2) = \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{e}})$. Let us now upper-bound the second term of the inequality. To this aim let us prove the following lemma.

LEMMA 2.3.1. *We have*

$$\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}) \leq \begin{cases} 1/q^{n-k} & \text{if } \mathbf{x} \text{ and } \mathbf{y} \text{ are colinear} \\ 1/q^{2(n-k)} & \text{otherwise.} \end{cases}$$

PROOF. The result is clear when $\mathbf{x}$ and $\mathbf{y}$ are colinear by Lemma 2.2.3. Let us suppose that $\mathbf{x}$ and $\mathbf{y}$ are not colinear and define

$$\varphi : \mathbf{h} \in \mathbb{F}_q^n \longmapsto (\mathbf{h} \cdot \mathbf{x}, \mathbf{h} \cdot \mathbf{y}).$$

It is readily verified that this linear application has a kernel of $\mathbb{F}_q$-dimension $n-2$. Therefore, for any $a, b \in \mathbb{F}_q$ and $\mathbf{h} \in \mathbb{F}_q^n$ being uniformly distributed,

$$\mathbb{P}_{\mathbf{h}}\left(\varphi(\mathbf{h}) = (a,b)\right) = \frac{q^{n-2}}{q^n} = \frac{1}{q^2} \tag{2.7}$$

Let us remark now that

$$\begin{aligned}\mathbb{E}_{\mathbf{H}}\left(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}\right) &= \mathbb{P}_{\mathbf{H}}\left(\mathbf{x}\mathbf{H}^{\intercal} = \mathbf{s} \ \textit{and} \ \mathbf{y}\mathbf{H}^{\intercal} = \mathbf{s}\right)\\ &= \mathbb{P}_{\mathbf{h}}\left(\varphi(\mathbf{h}) = (a,b)\right)^{n-k} \end{aligned} \tag{2.8}$$

where in the last line we used that each rows of $\mathbf{H}$ are independent and uniformly distributed. To conclude the proof it remains to plug Equation (2.7) in Equation (2.8). □

Lemma 2.3.1 enables us to deduce that

$$\begin{aligned}\sum_{\substack{\mathbf{x},\mathbf{y}\in\mathscr{S}_t\\ \mathbf{x}\neq\mathbf{y}}} \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}}\mathbb{1}_{\mathbf{y}}) - \mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{x}})\mathbb{E}_{\mathbf{H}}(\mathbb{1}_{\mathbf{y}}) &\leq \sum_{\mathbf{x}\in\mathscr{S}_t} \sum_{\substack{\mathbf{y}\in\mathscr{S}_t\setminus\mathbf{x}:\\ \text{colinear to } \mathbf{x}}} \frac{1}{q^{n-k}} - \frac{1}{q^{2(n-k)}}\\ &\leq \sum_{\mathbf{x}\in\mathscr{S}_t} \sum_{\substack{\mathbf{y}\in\mathscr{S}_t\setminus\mathbf{x}:\\ \text{colinear to } \mathbf{x}}} \frac{1}{q^{n-k}}\\ &\leq \frac{(q-2)\binom{n}{t}(q-1)^t}{q^{n-k}} \end{aligned} \tag{2.9}$$

It gives by plugging (2.9) in (2.6)

$$\begin{aligned}\mathbb{P}_{\mathbf{H}}\left(\left|N_t(\mathcal{C},\mathbf{s}) - \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right| \geq a\right) &\leq \frac{1}{a^2}\left(\frac{\binom{n}{t}(q-1)^t}{q^{n-k}} + \frac{(q-2)\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\\ &= \frac{(q-1)\binom{n}{t}(q-1)^t}{a^2 q^{n-k}}\end{aligned}$$

which concludes the proof. □

The point of this proposition is that, for relative weights $t/n \in (\tau^-, \tau^+)$ the term $\binom{n}{t}(q-1)^t/q^{n-k}$, is exponentially large. Therefore, by carefully setting a in this case we deduce $N_t(\mathcal{C},\mathbf{s})$ with a very good precision. For instance, if $t/n \in (\tau^-,\tau^+)$, let $a = \left(\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)^{3/4}$ to obtain:

$$\mathbb{P}_{\mathbf{H}}\left(\left|N_t(\mathcal{C},\mathbf{s}) - \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right| \geq \left(\frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)^{3/4}\right) \leq (q-1)\sqrt{\frac{q^{n-k}}{\binom{n}{t}(q-1)^t}} \in \mathrm{negl}(n).$$

The number of errors of weight t that reach some syndrome is with an overwhelming probability equal to its expectation $\binom{n}{t}(q-1)^t/q^{n-k}$ up to an additive factor $\left(\binom{n}{t}(q-1)^t/q^{n-k}\right)^{3/4}$ which is exponentially small with respect to $\binom{n}{t}(q-1)^t/q^{n-k}$.

2.4. Expected Minimum Distance of Codes

We are now interested in computing the expected minimum distance of a random code. As we will see it is given by the so-called *Gilbert-Varshamov* distance. This result is for cryptographic purposes very important. Suppose that one finds in a code a word of Hamming weight much smaller than it is expected. This would mean that the code is peculiar and maybe even worse, this codeword of small weight may reveal some secret information.

In view of the foregoing, it may be tempting to say that the expected minimum distance of a random $[n,k]_q$-code is given by the largest t such that $\sum_{\ell\leq t}\frac{\binom{n}{\ell}(q-1)^\ell}{q^{n-k}}\leq 1$ and that is indeed what happens. It turns out that this value of t plays an important role in coding theory and is known as the Gilbert-Varshamov distance.

DEFINITION 2.4.1 (Gilbert-Varshamov Distance). *Let $k\leq n$ and q be integers. The Gilbert-Varshamov distance $t_{\mathrm{GV}}(q,n,k)$ is defined as the largest integer such that*

$$\sum_{\ell=0}^{t_{\mathrm{GV}}(q,n,k)}\binom{n}{\ell}(q-1)^\ell\leq q^{n-k}.$$

REMARK 2.4.1. *The Gilbert-Varshamov distance gives the maximum r such that the volume of the ball of radius r is smaller than the inverse density of any $[n,k]_q$-code. Its analogue for lattices (in the context of lattice-based cryptography) is called the Gaussian heuristic.*

It can be verified (by using Lemma 2.1.1)

$$\frac{t_{\mathrm{GV}}(q,n,Rn)}{n}=\tau^-+o(1)$$

where τ^- is defined in Equation (2.4). It explains why τ^- is commonly called the *relative Gilbert-Varshamov distance.* In the following proposition we show that the minimum distance of almost all $[n,k]_q$-codes is given by τ^-. Interestingly, the proof that $d_{\min}(\mathcal{C})/n>\tau^-$ happens with a negligible probability relies on Proposition 2.3.3 that used the second moment technique.

PROPOSITION 2.4.1. *Let $\varepsilon>0$. We have*

$$\mathbb{P}_{\mathbf{H}}\left((1-\varepsilon)\tau^-<\frac{d_{\min}(\mathcal{C})}{n}<(1+\varepsilon)\tau^-\right)\geq 1-q^{-\alpha n(1+o(1))}$$

where $\alpha\stackrel{def}{=}\min\left((1-R)-h_q\left((1+\varepsilon)\tau^-\right),h_q\left((1-\varepsilon)\tau^-\right)-(1-R)\right)>0$.

PROOF. First notice that,

$$\begin{aligned}\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n}\notin\left((1-\varepsilon)\tau^-,(1+\varepsilon)\tau^-\right)\right)&\leq\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n}\leq(1-\varepsilon)\tau^-\right)\\&\quad+\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n}\geq(1+\varepsilon)\tau^-\right).\end{aligned}\tag{2.10}$$

Let us upper-bound independently both terms of the above inequation. First,

$$\begin{aligned}\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n}\leq(1-\varepsilon)\tau^-\right)&=\mathbb{P}_{\mathbf{H}}\left(\exists\mathbf{x}\ :\ \frac{|\mathbf{x}|}{n}\leq(1-\varepsilon)\tau^-\text{ and }\mathbf{x}\in\mathcal{C}\right)\\&\leq\sum_{\substack{\mathbf{x}:\\ \frac{|\mathbf{x}|}{n}\leq(1-\varepsilon)\tau^-}}\mathbb{P}_{\mathbf{H}}\left(\mathbf{x}\in\mathcal{C}\right)\\&=\sum_{\ell=0}^{(1-\varepsilon)n\tau^-}\frac{\binom{n}{\ell}(q-1)^\ell}{q^{n-k}}\quad\text{(By Lemma 2.2.3.)}\\&=\frac{q^{n\left(h_q\left((1-\varepsilon)\tau^-\right)+o(1)\right)}}{q^{n-k}}\qquad(2.11)\\&=q^{n\left(h_q\left((1-\varepsilon)\tau^-\right)-(1-R)+o(1)\right)}\qquad(2.12)\end{aligned}$$

where we used in (2.11) Lemma 2.1.1 and the fact that $x\mapsto h_q(x)$ is an increasing function for $x\in[0,\tau^-]\subseteq\left[0,\frac{q-1}{q}\right]$.

Let us now upper-bound the second term of (2.10). Let $\delta \in (0,\varepsilon)$ and $u \stackrel{\text{def}}{=} (1+\delta)n\tau^-$. Notice now that,

$$\begin{aligned}\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n} \geq (1+\varepsilon)\tau^-\right) &\leq \mathbb{P}_{\mathbf{H}}\left(N_u(\mathcal{C},\mathbf{0}) = 0\right)\\ &\leq \mathbb{P}_{\mathbf{H}}\left(\left|N_u(\mathcal{C},\mathbf{0}) - \frac{\binom{n}{u}(q-1)^u}{q^{n-k}}\right| \geq \frac{\binom{n}{u}(q-1)^u}{q^{n-k}}\right)\\ &\leq (q-1)\frac{q^{n-k}}{\binom{n}{u}(q-1)^u} \qquad (2.13)\end{aligned}$$

where in the last line we used Proposition 2.3.3 by setting $a = \frac{\binom{n}{u}(q-1)^u}{q^{n-k}}$. But now by using Lemma 2.1.1 and that $u = (1+\delta)\tau^-$ we obtain:

$$\frac{q^{n-k}}{\binom{n}{u}(q-1)^u} = q^{n\left(1-R-h_q\left((1+\delta)\tau^-\right)+o(1)\right)}.$$

By plugging this in Equation (2.13) we obtain for any $\delta \in (0,\varepsilon)$:

$$\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n} \geq (1+\varepsilon)\tau^-\right) \leq q^{n\left(1-R-h_q\left((1+\delta)\tau^-\right)+o(1)\right)}.$$

Therefore, by letting $\delta \to \varepsilon$ we get:

$$\mathbb{P}_{\mathbf{H}}\left(\frac{d_{\min}(\mathcal{C})}{n} \geq (1+\varepsilon)\tau^-\right) \leq q^{n\left(1-R-h_q\left((1+\varepsilon)\tau^-\right)+o(1)\right)}. \qquad (2.14)$$

To conclude the proof it remains to put together Equations (2.12) and (2.14) in Equation (2.10). □

REMARK 2.4.2. *In coding theory, the relative Gilbert-Varshamov distance τ^- is known as a "lower-bound", there exists a family of $[n, Rn]_q$-codes with a relative minimum distance $\geq \tau^-$. What we have actually proven is that almost all families of $[n, Rn]_q$-codes have asymptotically a relative minimum distance $= \tau^-$. Let us stress that it does not show that the relative Gilbert-Varshamov distance τ^- is an "upper-bound", all families of $[n, Rn]_q$-codes are such that their asymptotically relative minimum distance is $\leq \tau^-$. This is a widely open conjecture in the case of $\mathbb{F}_2$ but which is not true for $\mathbb{F}_q$ as long as q is a square ≥ 49.*

About the optimality of random codes. We have seen in Chapter 1 that balls centred at codewords $\mathbf{c} \in \mathcal{C}$ and whose radius is $< \frac{d_{\min}(\mathcal{C})}{2}$ never overlap. This condition over the radius ensures that when an error $\mathbf{e}$ of Hamming weight smaller than $< \frac{d_{\min}(\mathcal{C})}{2}$ occurs, we are sure that computing the closest codeword from $\mathbf{c}+\mathbf{e}$ will outcome $\mathbf{c}$ (we say that the *maximum likelihood decoding* succeeds). However, for random codes this property can be made even stronger. In that case we can show that $\mathbf{c}$ is indeed the closest codeword from $\mathbf{c}+\mathbf{e}$ (with overwhelming probability) if $\mathbf{e} \approx d_{\min}(\mathcal{C})$ where $d_{\min}(\mathcal{C}) \approx \tau^- n$ as we show now in the following proposition (by using Markov's inequality).

PROPOSITION 2.4.2. *Let $t \stackrel{def}{=} n(1-\varepsilon)\tau^-$ for some $\varepsilon > 0$ and let $\eta > 0$. We have*

$$\mathbb{P}_{\mathbf{H}}\left(\frac{\sharp\left\{\mathbf{c},\mathbf{c}' \in \mathcal{C},\ \mathbf{e},\mathbf{e}' \in \mathscr{S}_t\ :\ \mathbf{c}+\mathbf{e} = \mathbf{c}'+\mathbf{e}'\right\}}{q^k\binom{n}{t}(q-1)^t} \geq 1+\eta\right) \leq \frac{1}{\eta}\,\frac{\binom{n}{t}(q-1)^t}{q^{n-k}} = \frac{1}{\eta}\,q^{-\alpha n(1+o(1))}$$

where $\alpha \stackrel{def}{=} h_q\left((1-\varepsilon)\tau^-\right) - 1 + R > 0$.

Notice that,

$$\frac{\sharp\left\{\mathbf{c},\mathbf{c}' \in \mathcal{C},\ \mathbf{e},\mathbf{e}' \in \mathscr{S}_t\ :\ \mathbf{c}+\mathbf{e} = \mathbf{c}'+\mathbf{e}'\right\}}{q^k\binom{n}{t}(q-1)^t} = 1$$

means that there are no collisions between noisy codewords with errors of Hamming weight t (and that balls of radius t and centered at codewords do not overlap). Therefore, the above proposition shows that for random codes, the maximum likelihood decoding will succeed for almost all noisy codewords, up to the Gilbert-Varshamov distance if η is chosen sufficiently small. However, once again, we could be more accurate by using the second moment technique with Bienaymé-Tchebychev's inequality.

Proof. Let,

$$Z \stackrel{\text{def}}{=} \sharp\left\{\mathbf{c},\mathbf{c}' \in \mathcal{C},\ \mathbf{e},\mathbf{e}' \in \mathscr{S}_t : \mathbf{c}+\mathbf{e} = \mathbf{c}'+\mathbf{e}'\right\}.$$

We have the following computation

$$\begin{aligned} Z &= \sum_{\mathbf{c}\in\mathcal{C},\mathbf{e}\in\mathscr{S}_t} 1 + \sum_{\substack{\mathbf{c},\mathbf{c}'\in\mathcal{C},\mathbf{e},\mathbf{e}'\in\mathscr{S}_t \\ (\mathbf{c},\mathbf{e})\neq(\mathbf{c}',\mathbf{e}') \\ \mathbf{c}+\mathbf{e}=\mathbf{c}'+\mathbf{e}'}} 1 \\ &= q^k \binom{n}{t}(q-1)^t + q^k \sum_{\substack{\mathbf{e},\mathbf{e}'\in\mathscr{S}_t \\ \mathbf{e}\neq\mathbf{e}':(\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}}} 1 \\ &= q^k \binom{n}{t}(q-1)^t \left(1 + \frac{1}{\binom{n}{t}(q-1)^t} \sum_{\substack{\mathbf{e},\mathbf{e}'\in\mathscr{S}_t \\ \mathbf{e}\neq\mathbf{e}':(\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}}} 1\right). \end{aligned}$$

Let,

$$X \stackrel{\text{def}}{=} \frac{1}{\binom{n}{t}(q-1)^t} \sum_{\substack{\mathbf{e},\mathbf{e}'\in\mathscr{S}_t \\ \mathbf{e}\neq\mathbf{e}':(\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}}} 1$$

By using Lemma 2.2.3,

$$\begin{aligned} \mathbb{E}_{\mathbf{H}}(X) &= \frac{1}{\binom{n}{t}(q-1)^t} \sum_{\substack{\mathbf{e},\mathbf{e}'\in\mathscr{S}_t \\ \mathbf{e}\neq\mathbf{e}'}} \frac{1}{q^{n-k}} \\ &\leq \frac{1}{\binom{n}{t}(q-1)^t} \frac{\left(\binom{n}{t}(q-1)^t\right)^2}{q^{n-k}} \\ &= \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}. \end{aligned}$$

To conclude the proof it is enough to apply Markov's inequality to upper-bound $\mathbb{P}_{\mathbf{H}}(X > \eta)$. □

2.5. Uniform Distribution of Syndromes

In Chapter 1 we have seen that the decoding problem is defined (for cryptographic purposes) with some distribution in input, namely $(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal)$ where $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and $\mathbf{x} \in \mathscr{S}_t$ are uniformly distributed. Our aim in what follows is to show that when $t/n \in (\tau^-, \tau^+)$ we could replace $\mathbf{x}\mathbf{H}^\intercal$ by $\mathbf{s} \in \mathbb{F}_q^{n-k}$ being uniformly distributed, without changing our problem. More precisely, we are going to show that distributions $(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal)$ and $(\mathbf{H}, \mathbf{s})$ are *statistically close* under the condition $t/n \in (\tau^-, \tau^+)$, showing that for any algorithm solving $\mathsf{DP}(n, q, R, \tau)$ we could choose its inputs as $(\mathbf{H}, \mathbf{s})$ without changing at much its success probability. This result may seem useless but in some applications where $\tau \in (\tau^-, \tau^+)$ it is much more comfortable to consider directly $(\mathbf{H}, \mathbf{s})$ rather than $(\mathbf{H}, \mathbf{x}\mathbf{H}^\intercal)$.

PROPOSITION 2.5.1. *Let* $k \stackrel{def}{=} \lfloor Rn \rfloor$, $t \stackrel{def}{=} \lfloor \tau n \rfloor$ *and* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $\mathbf{e} \in \mathscr{S}_t$ *being uniformly distributed. We have*

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}\right)\right) \leq \frac{1}{2}\sqrt{\frac{q^{n-k}-1}{\binom{n}{t}(q-1)^t}}. \tag{2.15}$$

In particular, when $\tau \in (\tau^-, \tau^+)$

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}\right)\right) \leq q^{-\alpha n(1+o(1))} \quad \textit{where} \quad \alpha \stackrel{def}{=} \frac{1}{2}\left(h_q(\tau) - 1 + R\right) > 0.$$

EXERCISE 2.5.1. *Let* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *being uniformly distributed,* $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $\mathbf{e} \in \mathscr{S}_t$ *be some random variables. Show that*

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}\right)\right) = \frac{1}{q^{(n-k)\times n}} \sum_{\mathbf{H}_0 \in \mathbb{F}_q^{(n-k)\times n}} \Delta\left(\mathbf{e}\mathbf{H}_0^{\intercal}, \mathbf{s}\right)$$

The proof of Proposition 2.5.1 relies on the following lemma which is a rewriting in our context of the result known as the *left over hash lemma*. Roughly speaking, it shows that if the outputs of a function collide with a probability ε-close to the case where they would be randomly distributed, then this function is $\sqrt{\varepsilon}$-close to a random function.

LEMMA 2.5.1. *Let* $\mathscr{H} = (h_i)_{i\in I}$ *be a finite family of applications from* E *in* F. *Let* ε *be the "collision bias"*

$$\mathbb{P}_{h,e,e'}(h(e) = h(e')) = \frac{1}{\sharp F}(1+\varepsilon)$$

where h *is uniformly drawn in* $\mathscr{H}$, e *and* e' *be distributed according to some random variable* X *taking its values* E. *Let* $\mathscr{U}$ *be the uniform distribution over* F *and* $\mathscr{D}(h)$ *be the distribution* $h(e)$ *when* e *is distributed according to* X. *We have,*

$$\mathbb{E}_h\left(\Delta(\mathscr{D}(h), \mathscr{U})\right) \leq \frac{1}{2}\sqrt{\varepsilon}.$$

PROOF. By definition of the statistical distance we have

$$\begin{aligned}
\mathbb{E}_h\left(\Delta(\mathscr{D}(h), \mathscr{U})\right) &= \sum_{h\in\mathscr{H}} \frac{1}{\sharp\mathscr{H}} \Delta\left(\mathscr{D}(h), \mathscr{U}\right) \qquad (2.16)\\
&= \frac{1}{2}\sum_{h\in\mathscr{H}} \frac{1}{\sharp\mathscr{H}} \sum_{f\in F} \left|\mathbb{P}_e(h(e) = f) - \frac{1}{\sharp F}\right| \\
&= \frac{1}{2}\sum_{(h,f)\in\mathscr{H}\times F} \left|\mathbb{P}_{h_0,e}(h_0 = h, h_0(e) = f) - \frac{1}{\sharp\mathscr{H}\ \sharp F}\right| \\
&= \frac{1}{2}\sum_{(h,f)\in\mathscr{H}\times F} \left|q_{h,f} - \frac{1}{\sharp\mathscr{H}\ \sharp F}\right|. \qquad (2.17)
\end{aligned}$$

where $q_{h,f} \stackrel{\text{def}}{=} \mathbb{P}_{h_0,e}\left((h_0, h_0(e)) = (h, f)\right)$ with h_0 being uniformly chosen at random in $\mathscr{H}$ and e be distributed according to X. Using the Cauchy-Schwarz inequality, we obtain

$$\sum_{(h,f)\in\mathscr{H}\times F} \left|q_{h,f} - \frac{1}{\sharp\mathscr{H}\ \sharp F}\right| \leq \sqrt{\sum_{(h,f)\in\mathscr{H}\times F} \left(q_{h,f} - \frac{1}{\sharp\mathscr{H}\ \sharp F}\right)^2}\ \sqrt{\sharp\mathscr{H}\ \sharp F}. \tag{2.18}$$

Let us observe now that

$$\begin{aligned}\sum_{(h,f)\in\mathscr{H}\times F}\left(q_{h,f}-\frac{1}{\sharp\mathscr{H}\,\sharp F}\right)^2 &= \sum_{h,f}\left(q_{h,f}^2-2\frac{q_{h,f}}{\sharp\mathscr{H}\,\sharp F}+\frac{1}{\sharp\mathscr{H}^2\,\sharp F^2}\right)\\ &= \sum_{h,f}q_{h,f}^2-2\frac{\sum_{h,f}q_{h,f}}{\sharp\mathscr{H}\,\sharp F}+\frac{1}{\sharp\mathscr{H}\,\sharp F}\\ &= \sum_{h,f}q_{h,f}^2-\frac{1}{\sharp\mathscr{H}\,\sharp F}. \qquad (2.19)\end{aligned}$$

Consider for $i\in\{0,1\}$ independent random variables h_i and e_i that are drawn uniformly at random in $\mathscr{H}$ and according to X respectively. We continue this computation by noticing now that

$$\begin{aligned}\sum_{h,f}q_{h,f}^2 &= \sum_{h,f}\mathbb{P}_{h_0,e_0}(h_0=h,h_0(e_0)=f)\mathbb{P}_{h_1,e_1}(h_1=h,h_1(e_1)=f)\\ &= \mathbb{P}_{h_0,h_1,e_0,e_1}\left(h_0=h_1,h_0(e_0)=h_1(e_1)\right)\\ &= \frac{\mathbb{P}_{h_0,e_0,e_1}\left(h_0(e_0)=h_0(e_1)\right)}{\sharp\mathscr{H}}\\ &= \frac{1+\varepsilon}{\sharp\mathscr{H}\,\sharp F}. \qquad (2.20)\end{aligned}$$

By substituting for $\sum_{h,f}q_{h,f}^2$ the expression obtained in (2.20) into (2.19) and then back into (2.18) we finally obtain

$$\begin{aligned}\sum_{(h,f)\in\mathscr{H}\times F}\left|q_{h,f}-\frac{1}{\sharp\mathscr{H}\,\sharp F}\right| &\le \sqrt{\frac{1+\varepsilon}{\sharp\mathscr{H}\,\sharp F}-\frac{1}{\sharp\mathscr{H}\,\sharp F}}\;\sqrt{\sharp\mathscr{H}\,\sharp F}\\ &= \sqrt{\frac{\varepsilon}{\sharp\mathscr{H}\,\sharp F}}\;\sqrt{\sharp\mathscr{H}\,\sharp F}\\ &= \sqrt{\varepsilon}.\end{aligned}$$

□

We are now ready to prove Proposition 2.5.1.

Proof of Proposition 2.5.1. Let us compute the "collision bias" in our case. We have:

$$\begin{aligned}\mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left(\mathbf{e}\mathbf{H}^\intercal=\mathbf{e}'\mathbf{H}^\intercal\right) &= \mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left((\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}\right)\\ &= \mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left((\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}\mid\mathbf{e}\neq\mathbf{e}'\right)\mathbb{P}\left(\mathbf{e}\neq\mathbf{e}'\right)+\mathbb{P}_{\mathbf{e},\mathbf{e}'}(\mathbf{e}=\mathbf{e}')\\ &= \frac{1}{q^{n-k}}\left(1-\frac{1}{\binom{n}{t}(q-1)^t}\right)+\frac{1}{\binom{n}{t}(q-1)^t}\quad\text{(by Lemma 2.2.3)}\\ &= \frac{1}{q^{n-k}}\left(1+\varepsilon\right)\quad\text{where}\quad\varepsilon\stackrel{\text{def}}{=}\frac{q^{n-k}-1}{\binom{n}{t}(q-1)^t}\end{aligned}$$

which shows (2.15) by using Lemma 2.5.1. The second part of the proposition easily follows from the definition of τ^- and τ^+. □

Exercise 2.5.2. *Show that if one replaces in the binary case ($q=2$) $\mathbf{e}$ in Proposition 2.5.1 by $\mathbf{e}^{\mathrm{Ber}}$, where the e_i^{Ber}'s are distributed according to a Bernoulli distribution of parameter τ, we would obtain*

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}^{\mathrm{Ber}}\mathbf{H}^\intercal,\mathbf{s}\right)\right)\le\frac{1}{2}\sqrt{2^{-k}\left(1+(1-2\tau)^2\right)^n}.$$

What can you deduce when comparing both results with $\mathbf{e}$ *or* $\mathbf{e}^{\mathrm{Ber}}$*? What is (according to Proposition 2.5.1) the "best" choice of error* $\mathbf{x}$ *to ensure that* $\mathbf{x}\mathbf{H}^\intercal$ *is uniformly distributed?*

EXERCISE 2.5.3. *Let* $\mathcal{C}$ *be a fixed* $[n,k]_q$*-code of parity-check matrix* $\mathbf{H}$ *and* $\mathbf{y},\mathbf{s},\mathbf{e} \in \mathbb{F}_q^n \times \mathbb{F}_q^{n-k} \times \mathscr{S}_t$ *be uniformly distributed. Our aim in this exercise is to show that* $\Delta(\mathbf{c}+\mathbf{e},\mathbf{y}) = \Delta(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s})$.

1. *Given* $\mathbf{s} \in \mathbb{F}_q^{n-k}$*, let* $\mathbf{y}(\mathbf{s}) \in \mathbb{F}_q^n$ *be such that* $\mathbf{y}(\mathbf{s})\mathbf{H}^\intercal = \mathbf{s}$*. Show that*
$$\sum_{\mathbf{y}\in\mathbb{F}_q^n} \left| \mathbb{P}_{\mathbf{e},\mathbf{c}}(\mathbf{c}+\mathbf{e}=\mathbf{y}) - \frac{1}{q^n} \right| = \sum_{\mathbf{s}\in\mathbb{F}_q^{n-k}} \sum_{\mathbf{c}'\in\mathcal{C}} \left| \mathbb{P}_{\mathbf{e},\mathbf{c}}(\mathbf{c}+\mathbf{e}=\mathbf{y}(\mathbf{s})+\mathbf{c}') - \frac{1}{q^n} \right|.$$
2. *Deduce that* $\Delta(\mathbf{c}+\mathbf{e},\mathbf{y}) = \Delta(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s})$.

Up to now, we have proven in Proposition 2.5.1 that syndromes $\mathbf{e}\mathbf{H}^\intercal$ are statistically close to the uniform distribution when $\mathbf{e}$ is picked uniformly at random in $\mathscr{S}_t$ with t larger than the Gilbert-Varshamov distance (more precisely, when $t/n \in (\tau^-,\tau^+)$) *but* in average over $\mathbf{H}$. One may ask if the result still holds for a fixed matrix $\mathbf{H}_0$? Actually we can prove, without too much effort, that the above result is true for almost all matrices but with a loss given by a square root as shown by the following proposition.

PROPOSITION 2.5.2. *Let* $k \overset{def}{=} \lfloor Rn \rfloor$, $t \overset{def}{=} \lfloor \tau n \rfloor$, $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *being uniformly distributed and* $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $\mathbf{e} \in \mathbb{F}_q^n$ *be some random variables. Suppose that*
$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s}\right)\right) \leq \varepsilon$$
Then, we have
$$\frac{\sharp\left\{\mathbf{H}_0 \in \mathbb{F}_q^{(n-k)\times n} \ : \ \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) \geq \sqrt{\varepsilon}\right\}}{q^{(n-k)\times n}} \leq \sqrt{\varepsilon}$$

PROOF. First,
$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s}\right)\right) = \frac{1}{q^{(n-k)\times n}} \sum_{\mathbf{H}_0\in\mathbb{F}_q^{(n-k)\times n}} \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right). \tag{2.21}$$
The idea of the proof is to split in two parts this sum according to the terms that are larger and smaller than $\sqrt{\varepsilon}$
$$\begin{aligned}
\sum_{\mathbf{H}_0\in\mathbb{F}_q^{(n-k)\times n}} \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) &= \sum_{\substack{\mathbf{H}_0 \ : \\ \Delta(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s})<\sqrt{\varepsilon}}} \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) + \sum_{\substack{\mathbf{H}_0 \ : \\ \Delta(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s})\geq\sqrt{\varepsilon}}} \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) \\
&\geq \sum_{\substack{\mathbf{H}_0 \ : \\ \Delta(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s})<\sqrt{\varepsilon}}} \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) + \sum_{\substack{\mathbf{H}_0 \ : \\ \Delta(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s})\geq\sqrt{\varepsilon}}} \sqrt{\varepsilon} \\
&\geq \sum_{\substack{\mathbf{H}_0 \ : \\ \Delta(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s})\geq\sqrt{\varepsilon}}} \sqrt{\varepsilon} \\
&= \sqrt{\varepsilon}\,\sharp\left\{\mathbf{H}_0 \in \mathbb{F}_q^{(n-k)\times n} \ : \ \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) \geq \sqrt{\varepsilon}\right\} \qquad (2.22)
\end{aligned}$$
By plugging Equation (2.22) in (2.21), we obtain
$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s}\right)\right) \geq \sqrt{\varepsilon}\,\frac{\sharp\left\{\mathbf{H}_0 \in \mathbb{F}_q^{(n-k)\times n} \ : \ \Delta\left(\mathbf{e}\mathbf{H}_0^\intercal,\mathbf{s}\right) \geq \sqrt{\varepsilon}\right\}}{q^{(n-k)\times n}}$$
But by assumption we have $\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathbf{e}\mathbf{H}^\intercal,\mathbf{s}\right)\right) \leq \varepsilon$. It concludes the proof. □

CHAPTER 3

Information Set Decoding Algorithms

Introduction

The aim of any code-based cryptosystem is to rely its security on the hardness of the decoding problem when the input code is random. It is therefore crucial to study the best algorithms, usually called *generic* decoding algorithms, for solving this problem. Despite many efforts on this issue the best ones [**BJMM12**, **MO15**, **BM17**] are exponential in the number of errors that have to be corrected and all can be viewed as a refinement of the original Prange's algorithm [**Pra62**]. They are actually referred to as *Information Set Decoding* (ISD). The aim of these lecture notes is to describe the "first" ISD algorithms. Our description will be mainly algorithmic with the use of parity-check matrices. However in this (too long) introduction we try to give another point of view, more related to the inherent mathematical structure of codes: linear subspaces of some $\mathbb{F}_q^n$.

Prange's approach: use linearity. Given an $[n,k]_q-$code $\mathcal{C}$ and one word $\mathbf{y} \in \mathbb{F}_q^n$, we are looking for some codeword $\mathbf{c} \in \mathcal{C}$ at distance t from $\mathbf{y}$, namely $|\mathbf{y}-\mathbf{c}| = t$. Here $\mathcal{C}$ is defined as a *linear* subspace of $\mathbb{F}_q^n$ of dimension k. Therefore one can check that there exists some set of positions $\mathscr{I} \subseteq [\![1,n]\!]$ of size k, *called an information set*, which uniquely determines every codewords, more precisely

$$\forall \mathbf{x} \in \mathbb{F}_q^k : \quad \exists! \mathbf{c} \in \mathcal{C} \text{ (that we can easily compute by linear algebra) such that } \mathbf{c}_{\mathscr{I}} = \mathbf{x}$$

where $\mathbf{c}_{\mathscr{I}}$ denotes the vector whose coordinates are those of $\mathbf{c} = (c_i)_{1\le i\le n}$ which are indexed by $\mathscr{I}$, *i.e.* $\mathbf{c}_{\mathscr{I}} = (c_i)_{i\in\mathscr{I}}$.

Exercise 3.0.1. *Let $\mathcal{C}$ be an $[n,k]_q$-code and $\mathscr{I} \subseteq [\![1,n]\!]$ be of size k. Show that,*

$$\begin{aligned} \mathscr{I} \text{ is an information set for } \mathcal{C} &\iff \forall \mathbf{G} \text{ generator matrix of } \mathcal{C},\ \mathbf{G}_{\mathscr{I}} \text{ is invertible} \\ &\iff \forall \mathbf{H} \text{ parity-check matrix of } \mathcal{C},\ \mathbf{H}_{\overline{\mathscr{I}}} \text{ is invertible} \end{aligned}$$

where given $\mathbf{M} \in \mathbb{F}_q^{r\times n}$, $\mathbf{M}_{\mathscr{I}}$ denotes the matrix whose columns *are those of $\mathbf{M}$ which are indexed by $\mathscr{I}$.*

Prange's idea to recover some solution $\mathbf{c}^{\text{sol}} \in \mathcal{C}$, where $\mathbf{y} = \mathbf{c}^{\text{sol}} + \mathbf{e}^{\text{sol}}$ and $|\mathbf{e}^{\text{sol}}| = t$, is as follows. First we pick some random information set $\mathscr{I}$ and we hope that it contains no error positions, namely:

$$\mathbf{c}^{\text{sol}}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}} \quad \left(\iff \mathbf{e}^{\text{sol}}_{\mathscr{I}} = \mathbf{0} \right). \tag{3.1}$$

If this is true, we are done. It remains to compute *the unique* codeword $\mathbf{c}$ such that $\mathbf{c}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}}$ as by uniqueness we get $\mathbf{c} = \mathbf{c}^{\text{sol}}$. In other words, Prange's idea (when looking for a close codeword) simply consists in picking some information set $\mathscr{I}$, computing the unique codeword $\mathbf{c}$ equal to $\mathbf{y}$ on those coordinates, and then to check if the constraint (3.1) is verified, namely if $|\mathbf{y}-\mathbf{c}| = t$. The average number of times we have to pick a set $\mathscr{I}$ in Prange's algorithm until finding a solution is therefore given by $1/p_{\text{pr}}$ where p_{pr} is the probability that Equation (3.1) is verified. As we will see, $1/p_{\text{pr}}$ is

- polynomial for $t/n = \left(\frac{q-1}{q}\right)\left(1-\frac{k}{n}\right)$,

- exponential in t when $t/n \in \left]0, \left(\frac{q-1}{q}\right)\left(1-\frac{k}{n}\right)\right[$

as long as $n \to +\infty$ and k/n is some constant. Interestingly, all improvements of Prange's algorithm, since sixty years, have the same behaviour with respect to t/n. Even though Prange's algorithm is quite naive, it really shows where decoding is easy, where it is not.

Let us now describe how these improvements, in particular ISD algorithms, were obtained as they start from the same key idea. For this let us back up a bit.

Dumer's approach: a collision search. The simplest way to find a codeword $\mathbf{c}$ at distance t from $\mathbf{y}$ is basically enumerating all the errors $\mathbf{e}$ of weight t until finding one that reaches $\mathbf{y}-\mathbf{e} \in \mathcal{C}$. This naive approach will obviously cost $\binom{n}{t}(q-1)^t$. However, taking advantage of the birthday paradox, this exhaustive enumeration can be improved as Dumer showed [Dum86]. Dumer's idea was to notice that if one splits in two parts[(1)] of the same size some parity-check matrix $\mathbf{H} = \begin{pmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{pmatrix}$ of $\mathcal{C}$, then solving the decoding problem boils down to finding $\mathbf{e}_1$ and $\mathbf{e}_2$ of Hamming weight $t/2$ such that $\mathbf{H}_1\mathbf{e}_1^\intercal + \mathbf{H}_2\mathbf{e}_2^\intercal = \mathbf{H}\mathbf{y}^\intercal$. A natural strategy to compute a solution $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ reduces to compute the following lists of $\mathbb{F}_q^{n-k}$

$$\mathscr{L}_1 \stackrel{\text{def}}{=} \left\{\mathbf{H}_1\mathbf{e}_1^\intercal : |\mathbf{e}_1| = \frac{t}{2}\right\} \quad \text{and} \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{-\mathbf{H}_2\mathbf{e}_2^\intercal + \mathbf{H}\mathbf{y}^\intercal : |\mathbf{e}_2| = \frac{t}{2}\right\}$$

and then to compute their "collision"

$$\mathscr{L}_1 \bowtie \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{(\mathbf{e}_1, \mathbf{e}_2) \in \mathscr{L}_1 \times \mathscr{L}_2, \quad \mathbf{H}_1\mathbf{e}_1^\intercal = -\mathbf{H}_2\mathbf{e}_2^\intercal + \mathbf{H}\mathbf{y}^\intercal\right\}.$$

This new list trivially leads to solutions of the decoding problem. However, what is the cost of this procedure? By using classical techniques such as hash tables or sorting lists, computing $\mathscr{L}_1 \bowtie \mathscr{L}_2$ costs, up to a polynomial factor, $\max(\sharp\mathscr{L}_1, \sharp\mathscr{L}_2) + \sharp(\mathscr{L}_1 \bowtie \mathscr{L}_2)$. Notice now that both lists $\mathscr{L}_1$ and $\mathscr{L}_2$ have the same size, namely

$$\binom{n/2}{t/2}(q-1)^{t/2} = \widetilde{O}\left(\sqrt{\binom{n}{t}(q-1)^t}\right).$$

To estimate the cost of this procedure it remains to estimate the size of $\mathscr{L}_1 \bowtie \mathscr{L}_2$. One can check that Dumer's approach finds all the solutions of the decoding problem (given by $\approx \max\left(1, \binom{n}{t}(q-1)^t/q^{n-k}\right)$ as shown in Chapter 2) but up to some polynomial loss, given by the probability that a solution $\mathbf{e}$ is not split into two equal parts. Then, $\mathscr{L}_1 \bowtie \mathscr{L}_2$ is the set of solution(s) of the considered decoding problem. To summarize, Dumer's approach enables to roughly find (up to polynomial factors)

$$\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right) \quad \text{solutions in time} \quad \sqrt{\binom{n}{t}(q-1)^t} + \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}. \tag{3.2}$$

Notice in the case where t is equal to the Gilbert-Varshamov distance, namely when $\binom{n}{t}(q-1)^t \approx q^{n-k}$, Dumer's algorithm has a quadratic gain compared to the exhaustive search. However, it is even better, as shown by the following proposition.

Proposition 3.0.1. *The running time of Prange's algorithm for solving* $\mathsf{DP}(n, q, R, \tau)$ *when* $\tau = h_q^{-1}(1-R)$[(2)] *and* $R \to 1$ *is given by:*

$$q^{n\,(1-R)(1+o(1))}$$

while Dumer's algorithm will cost:

$$q^{n\,\frac{1-R}{2}(1+o(1))}.$$

(1) To simplify the presentation, the cut is explained by taking the first $\frac{n-k}{2}$ positions for the first part and the other $\frac{n-k}{2}$ else for the second part, but of course in general these two sets of positions are randomly chosen.

(2) The relative Gilbert-Varshamov distance.

Dumer's algorithm has therefore a quadratic gain over Prange when the code rate tends to one and decoding at the Gilbert-Varhsamov distance. Though, the primary interest of this approach is not here. First, Dumer's algorithm finds (almost) all solutions of the decoding problem even if there are many of them. Furthermore, the distance t can be chosen such that it finds (almost) all of them in *amortized time one*.

DEFINITION 3.0.1 (Amortized time one). *An algorithm that outputs S solutions in time T of some problem is said to be in amortized time one if $S = \frac{T}{P(n)}$ for some polynomial P. In the sequel we will always neglect this polynomial factor.*

Dumer's algorithm works in amortized time one when t is beyond the Gilbert-Varshamov bound and verifies:

$$\sqrt{\binom{n}{t}(q-1)^t} = \frac{\binom{n}{t}(q-1)^t}{q^{n-k}} \iff \binom{n}{t}(q-1)^t = \left(q^{n-k}\right)^2. \tag{3.3}$$

As we are going to explain, most of the ideas to improve Prange's algorithm were based on these two remarks. The key idea is to reduce the initial decoding problem to a "denser" decoding problem where there are an exponential number of solutions but which can be found in amortized time one.

A mixed approach: ISD. The key point to improve Prange's algorithm starts from the following idea. Given some set of positions $\mathscr{J} \subseteq [\![1, n]\!]$ of size $k+\ell$ where $\ell > 0$, compute first a set $\mathscr{S}$ of *decoding candidates* which are some vectors at distance p from the target $\mathbf{y}$ when their coordinates are restricted to $\mathscr{J}$, namely:

$$\mathscr{S} \subseteq \left\{\mathbf{c}_{\mathscr{J}} : |\mathbf{c}_{\mathscr{J}} - \mathbf{y}_{\mathscr{J}}| = p \text{ and } \mathbf{c} \in \mathcal{C}\right\}. \tag{3.4}$$

Notice that $\mathscr{S}$ is a subset of the solutions of a decoding problem at distance p when it is given as input the target $\mathbf{y}_{\mathscr{J}}$ and the code

$$\mathscr{D} \stackrel{\text{def}}{=} \left\{\mathbf{c}_{\mathscr{J}} \in \mathbb{F}_q^{k+\ell} : \mathbf{c} \in \mathcal{C}\right\}. \tag{3.5}$$

It turns out that $\mathscr{D}$ is a code known as the *punctured* code of $\mathcal{C}$ at the positions $\overline{\mathscr{J}}$. Its length is $k+\ell$ and its dimension is k if $\mathscr{J}$ is an *augmented information set*, namely it contains some information set of $\mathcal{C}$, which will be assumed in what follows. Under this condition, $\mathbf{c}_{\mathscr{J}}$ uniquely determines its "lift" $\mathbf{c} \in \mathcal{C}$ which can be easily computed by linear algebra.

EXERCISE 3.0.2. *Let $\mathcal{C}$ be an $[n,k]_q$-code and $\mathscr{J} \subseteq [\![1,n]\!]$ be of size $k+\ell$. Show that,*

$$\mathscr{J} \text{ is an augmented information set for } \mathcal{C} \iff \mathscr{D} \text{ defined in Equation (3.5) has dimension } k.$$

Now, for the codeword $\mathbf{c} \in \mathcal{C}$, such that $\mathbf{c}_{\mathscr{J}} \in \mathscr{S}$, to be a solution of the original decoding problem, it has necessarily to verify

$$\left|\mathbf{c}_{\overline{\mathscr{J}}} - \mathbf{y}_{\overline{\mathscr{J}}}\right| = t - p. \tag{3.6}$$

This condition is weaker than of Prange algorithm (see Equation (3.1)): by picking our set $\mathscr{J}$ we do not hope to remove all the errors but only some fraction of it. Furthermore, contrary to Prange's approach we have many decoding candidates for each draw of the augmented information set $\mathscr{J}$. However, notice that smaller is p, harder it will be to compute even one decoding candidate. Therefore we cannot reasonably hope to choose p too small if we seek to test many decoding candidates at each draw of $\mathscr{J}$. It also turns out that if p is too small (below the Gilbert-Varshamov bound of the punctured code $\mathscr{D}$) no solutions are expected while on the other hand, if p is just above the Gilbert-Varshamov distance, we expect an exponential number of solutions.

So all in all, we have reduced our problem to decode a code of length n and dimension k to the bet made in (3.6) *and* the computation of $\mathscr{S}$ (*i.e.* the decoding candidates) which is nothing else than decoding a

"sub"-code of length $k+\ell$ and dimension k. This whole approach is known as Information Set Decoding (ISD). Note that we are completely free to choose our favourite algorithm to compute $\mathscr{S}$. Each ISD is then "parametrized" by the algorithm used as a subroutine for computing this set and, the better the algorithm, the better the ISD. However one may ask our meaning of a "better" algorithm for computing $\mathscr{S}$. To understand this let us introduce the probability $\alpha_{p,\ell}$ that a fixed $\mathbf{c}_{\mathscr{J}} \in \mathscr{S}$ leads to $\mathbf{c} \in \mathcal{C}$ which verifies Equation (3.6). We will show that the overall probability (after computing $\mathscr{S}$) to get a solution is given by $\approx \min(1, \sharp\mathscr{S}\ \alpha_{p,\ell})$. It will lead to the following proposition that gives the running time of the whole algorithm to solve DP[3].

PROPOSITION 3.0.2. *Assume that, given a random code $\mathscr{D}$ of length $k+\ell$, dimension k and a target $\mathbf{z} \in \mathbb{F}_q^{k+\ell}$, we can compute in time T a set of size S of codewords $\mathbf{d} \in \mathscr{D}$ at distance p from $\mathbf{z}$. Then, we can solve $\mathsf{DP}(n, q, R, \tau)$ in average time (up to a polynomial factor in n)*

$$T\ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right). \tag{3.7}$$

The overall cost for solving DP is therefore crucially parametrized by the cost for decoding a code $\mathscr{D}$ of rate $k/(k+\ell)$ at distance p, but notice that we need to find S solutions in time T and a priori not only one. If we want to design algorithms achieving this task such that the ISD improves original Prange's algorithm we have first to understand how parameters p, ℓ and quantities T, S interact.

Let us admit that $p \mapsto \alpha_{p,\ell}$ is a decreasing function. Notice now that, the larger p, the larger the number of solutions and the easier the decoding of $\mathscr{D}$ at distance p. Therefore we can reasonably suppose that $p \mapsto T$ is also a decreasing function. These two facts lead to a contradictory situation to minimize the ISD cost, we need to choose p as small as possible for minimizing $1/\alpha_{p,\ell}$ while at the same time we need to choose a large p to decrease T. Notice now, as $T \geq S$, that we have

$$T\ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right) \geq \frac{1}{\alpha_{p,\ell}}$$

Therefore we do not really have the choice, to minimize the cost of the ISD we have in the best case to design a sub-routine such that for parameters p and ℓ we have above an equality instead of an inequality. In particular it shows that our decoding algorithm at distance p (as small as possible) needs to find solutions in amortized time one, *i.e.* $S = T$. If this can be done we would get an improvement over Prange. Indeed, we have to remember that $\alpha_{p,\ell}$, the probability that our decoding candidate verifies Equation (3.6), is exponentially larger than the probability to verify Equation 3.1 as in Prange's algorithm (our bet is weaker).

Our discussion has just shown that it is theoretically possible to improve Prange's algorithm if we succeed, given a code $\mathscr{D}$ of length $k+\ell$, dimension k and any target $\mathbf{z}$, to compute in amortized time one many codewords $\mathbf{d} \in \mathscr{D}$ at distance p (as small as possible) from $\mathbf{z}$. The fundamental remark here is that $\mathscr{D}$ has a rate given by $k/(k+\ell) \approx 1$ when ℓ is not too large. It corresponds exactly to the range of parameters where Dumer's algorithm (that we have described earlier) can decode in amortized time one and can also have a quadratic gain over the original Prange algorithm. However parameters p and ℓ have to be carefully chosen as in Equation (3.3) (where replace t by p and n by $k+\ell$). In particular p cannot be chosen too small. Even though this choice of parameters is extremely constrained, the ISD using Dumer's algorithm improves Prange algorithm. But the better was yet to come. More sophisticated algorithms were designed, enabling to change the balance of parameters between p and ℓ by still decoding in amortized time one (in particular decreasing $p/(k+\ell)$ but also increasing ℓ to move away the rate $k/(k+\ell)$ from one). In these lecture notes we will restrict our study to the improvement given by the generalized birthday algorithm [**Wag02**]. But nowadays there exists far better techniques, such as "representations technique" (originally used for solving

[3] The following proposition is the equivalent of Proposition 3.3.3 with the "noisy codeword" point of view.

subset-sum problems) [**BJMM12**] or nearest neighbours search [**MO15, BM17**] but this is out of scope of these lecture notes.

Basic notation. Given $\mathbf{H} \in \mathbb{F}_q^{r \times n}$ and $\mathscr{I} \subseteq [\![1, n]\!]$ we will denote by $\mathbf{H}_{\mathscr{I}}$ the matrix whose *columns* are those of $\mathbf{H}$ which are indexed by $\mathscr{I}$.

During all these lecture notes both $R \in (0, 1)$ and the field size q will be supposed to be constants.

Described algorithms to solve DP. We will describe three ISD algorithms to solve $\mathsf{DP}(n, q, R, \tau)$ (Problem 1.2.5 in Chapter 1). In each case we will show that their running time (over the input distribution) is of the form $2^{n\,\alpha(n,q,R,\tau)(1+o(1))}$. For all of them (and all known algorithms), their exponent $\alpha(n, q, R, \tau)$ is > 0 as long as τ does not belong to $\left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$ as roughly described in Figure 3.1. Our aim during this lecture will be to compute the exponents of the three described algorithms. We draw them in Figures 3.2, 3.3 and 3.4 as function of τ for some rates R and field sizes q.

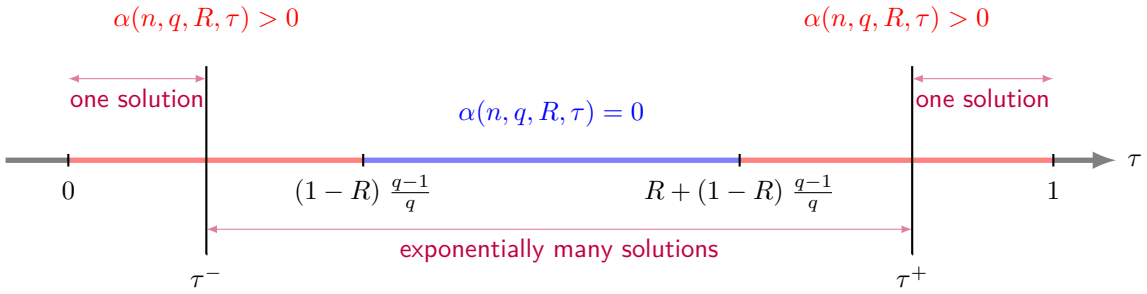


FIGURE 3.1. Exponents of the best generic decoding algorithms and expected number of solutions of $\mathsf{DP}(n, q, R, \tau)$ as function of τ.

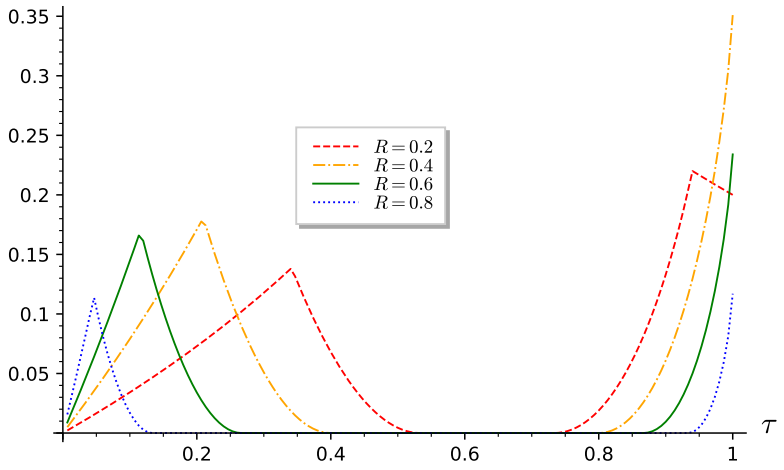


FIGURE 3.2. Exponent of Prange's algorithm (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $q = 3$ and different rates R as function of τ.

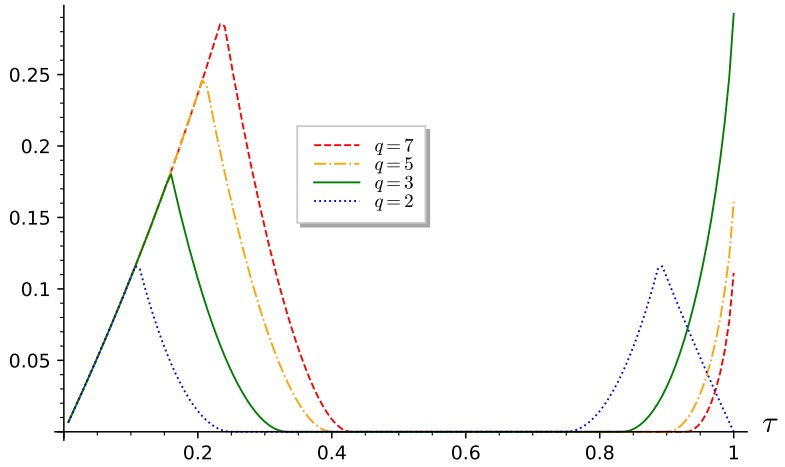


FIGURE 3.3. Exponent of Prange's algorithm (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.5$ and different field sizes q as function of τ.

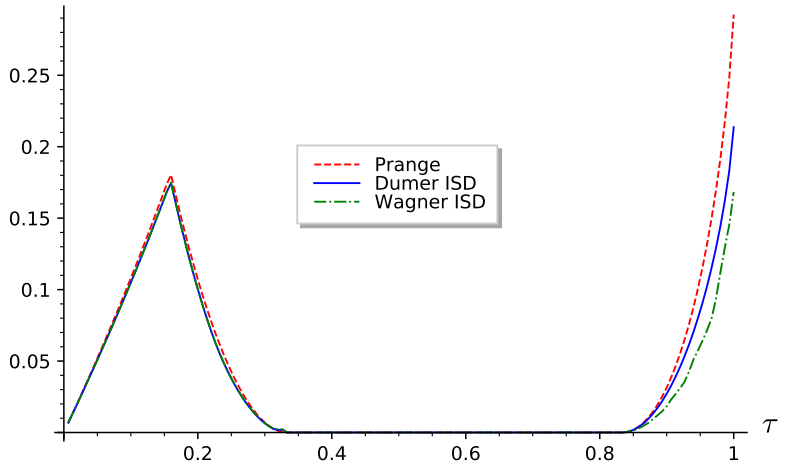


FIGURE 3.4. Exponents of Prange's algorithm and ISD with Dumer and Wagner' algorithms (in base 2) to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.5$ and $q = 3$ as function of τ.

3.1. Prange Algorithm

From now on, let us fix some instance $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ of the decoding problem $\mathsf{DP}(n, q, R, \tau)$ where recall that $k = \lfloor Rn \rfloor$. Our aim is to find $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight $t = \lfloor \tau n \rfloor$ such that $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ and we know that by definition there is at least one solution.

It corresponds to solving a linear system with $n-k$ equations and n unknowns with a constraint on the Hamming weight of the solution. Prange's idea simply consists in "fixing" k unknowns and then solving a square linear system of size $(n-k)\times(n-k)$ hoping that the solution will have the correct Hamming weight; if not, we just repeat by fixing k other unknowns[(4)]. More precisely, Prange's algorithm is as follows. Let us first introduce the following distribution $\mathscr{D}_t$ over vectors of $\mathbb{F}_q^k$, for reasons that will be clear in the sequel.

The distribution $\mathscr{D}_t$.

- If $t < \frac{q-1}{q}(n-k)$, $\mathscr{D}_t$ only outputs $\mathbf{0} \in \mathbb{F}_q^k$,
- if $t \in [\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]$, $\mathscr{D}_t$ outputs uniform vectors of weight $t - \frac{q-1}{q}(n-k)$,
- if $t > k + \frac{q-1}{q}(n-k)$, $\mathscr{D}_t$ outputs uniform vectors of weight k.

The algorithm.

1. *Picking the information set.* Let $\mathscr{I} \subseteq [\![1, n]\!]$ be a random set of size k. If $\mathbf{H}_{\overline{\mathscr{I}}} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ is not of full-rank, pick another set $\mathscr{I}$.
2. *Linear algebra.* Perform a Gaussian elimination to compute a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ such that $\mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}} = \mathbf{1}_{n-k}$.
3. *Test Step.* Pick $\mathbf{x} \in \mathbb{F}_q^k$ according to the distribution $\mathscr{D}_t$ and let $\mathbf{e} \in \mathbb{F}_q^n$ be such that

$$\mathbf{e}_{\overline{\mathscr{I}}} = \left(\mathbf{s} - \mathbf{x}\mathbf{H}_{\mathscr{I}}^\intercal\right)\mathbf{S}^\intercal \quad ; \quad \mathbf{e}_{\mathscr{I}} = \mathbf{x}. \tag{3.8}$$

   If $|\mathbf{e}| \neq t$ go back to Step 1, otherwise it is a solution.

Correction of the algorithm. It easily follows from the following computation,

$$\begin{aligned}
\mathbf{S}\mathbf{H}\mathbf{e}^\intercal &= \mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}}\,\mathbf{e}_{\overline{\mathscr{I}}}^\intercal + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{e}_{\mathscr{I}}^\intercal \\
&= \mathbf{e}_{\overline{\mathscr{I}}}^\intercal + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{e}_{\mathscr{I}}^\intercal \qquad \text{(by definition } \mathbf{S}\mathbf{H}_{\overline{\mathscr{I}}} = \mathbf{1}_{n-k}\text{)} \\
&= \mathbf{S}\left(\mathbf{s}^\intercal - \mathbf{H}_{\mathscr{I}}\,\mathbf{x}^\intercal\right) + \mathbf{S}\mathbf{H}_{\mathscr{I}}\,\mathbf{x}^\intercal \qquad \text{(by Equation (3.8))} \\
&= \mathbf{S}\mathbf{s}^\intercal
\end{aligned}$$

which corresponds to $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$ as $\mathbf{S}$ is non-singular. Furthermore the end of Step 3 is here to ensure that $\mathbf{e}$ will have the correct Hamming weight once the algorithm terminates.

Remark 3.1.1. *Let $\mathbf{y} \in \mathbb{F}_q^n$ be such that $\mathbf{y}\mathbf{H}^\intercal = \mathbf{s}$. Notice that $\mathbf{y} - \mathbf{e} = \mathbf{c} \in \mathcal{C}$ and by definition of $\mathbf{e}$ output by Prange's algorithm we have $\mathbf{c}_{\mathscr{I}} = \mathbf{y}_{\mathscr{I}}$ when $\mathbf{x} = \mathbf{0}$. In other words, when $\mathbf{x} = \mathbf{0}$, we recover the interpretation given in introduction to find a close codeword.*

Exercise 3.1.1. *Describe Prange's algorithm with the generator matrix formalism in the same fashion as above (with also three steps and the distribution $\mathscr{D}_t$).*

Far or close codeword? One may ask why did we pick some vector $\mathbf{x}$ in Step 3 of the algorithm? Notice that it corresponds to fixing k unknowns to the value that we want. Suppose now that we would like to find a solution $\mathbf{e}$ of small Hamming weight. Obviously fixing $\mathbf{x}$ to be a non-zero vector is both needless and

(4) Another interpretation of Prange's algorithm.

counterproductive as it would increase the weight of the decoding candidate. It is therefore better to choose $\mathbf{x}$ as $\mathbf{0}$ if we seek a solution of small Hamming weight. But now what happens if someone is looking for an error $\mathbf{e}$ of large Hamming weight, let us say close to n? The exact opposite: we need to choose $\mathbf{x}$ as a non-zero Hamming weight vector to increase the weight of the potential solution and therefore improving our success probability.

In summary, the vector $\mathbf{x}$ that we pick relies on what we want to do, finding a "short" or a "large" solution. The distribution $\mathscr{D}_t$, upon which $\mathbf{x}$ is picked, is precisely chosen according to the aforementioned aim. Equivalently, if one takes the generator matrix point of view, the distribution $\mathscr{D}_t$ enables to find a close or a far away codeword from a given target.

Rough analysis of the algorithm. Before giving a precise analysis of the running time of Prange's algorithm let us start by a rough analysis about what we "expect". First, let us assume that $\mathbf{s}$ *is uniformly distributed over* $\mathbb{F}_q^{n-k}$. Notice that it is, according to Proposition 2.5.1 in Chapter 2, equivalent to assuming that t/n belongs to $[\tau^-, \tau^+]$. Therefore, according to Equation (3.8) the expected Hamming weight of the decoding candidate $\mathbf{e}$ is given by ($\mathbf{S}$ is non-singular hence it keeps invariant the uniform distribution of $\mathbf{s}$)

$$\mathbb{E}\left(|\mathbf{e}|\right) = |\mathbf{x}| + \frac{q-1}{q}\,(n-k).$$

By choosing $\mathbf{x} = \mathbf{0}$ we expect $\mathbf{e}$ to have a Hamming weight equals to $\frac{q-1}{q}\,(n-k)$. In other words, if one seeks a solution of DP with the aforementioned weight, its probability of success (in Step 3) is roughly $p_{\mathrm{pr}} \approx 1$ and the number of repetitions of the whole algorithm will be given by $1/p_{\mathrm{pr}} \approx 1$. On the other hand, if one wants a weight smaller than $(1-\varepsilon)\,\frac{q-1}{q}\,(n-k)$ or larger than $(1+\varepsilon)\,\frac{q-1}{q}\,(n-k)$, its probability of success will be exponentially small in $\varepsilon\,(n-k)$ since $\mathbf{s}$ is uniformly distributed. In that case we will need to repeat the three steps an exponential number of times before succeeding. However we can turn the above strategy into a stronger one: by carefully choosing $|\mathbf{x}| \in [\![0, k]\!]$ (recall that $\mathbf{x}$ is a vector of length k, the co-dimension of our "constrained" linear system to solve), we can easily reach any weight in

$$\left[\!\!\left[\frac{q-1}{q}\,(n-k),\ k + \frac{q-1}{q}\,(n-k)\right]\!\!\right].$$

It explains why there is a whole interval in which DP is claimed to be easy to solve (as drawn in Figure 3.1). Let us stress once again that no algorithm is known to solve DP in polynomial time outside this range of parameters (up to an additive logarithmic factors in the above interval).

Precise analysis of the algorithm. All the challenge in the analysis of Prange's algorithm running time relies on the computation of the success probability in Step 3. From now on we will we make the following assumption concerning Step 1 of the algorithm

Assumption 3.1.1. *The success probability of Prange's algorithm is equal (up to a polynomial factor) to the probability of success when $\mathscr{I}$ is supposed to be uniformly distributed in Step* 1.

It is an usual assumption (or heuristic) to make when studying the complexity of Prange's algorithm. Notice that we did not suppose that $\mathscr{I}$ is uniformly distributed, but that our probabilities will be well approximated by making this assumption. It would be obviously false to suppose directly that $\mathscr{I}$ is uniformly distributed as $\mathbf{H}_{\overline{\mathscr{I}}}$ will be non-singular for some sets $\mathscr{I}$ (at the exception of very particular cases). However, when $\mathbf{H}$ is random, $\mathbf{H}_{\overline{\mathscr{I}}}$ is typically non-singular.

In the following lemma we give the success probability of Prange's algorithm. Our proof is written with a lot of details. We will not repeat this and we will process in a simpler way. The idea is that we study

algorithms to solve DP *in average* and from a cryptographic point of view we are on the cryptanalysis side. Our aim is to show that solving the decoding problem requires at least some number of operations. It is why we can suppose to live in the best world for a cryptanalyst. For instance, an event that is expected or that occurs with a probability given by the inverse of a polynomial, *always* happens and we are not concerned with approximation factors (although some heuristics may be hidden). The rationale behind the following proof is to show that what follows during these lecture notes could be stated and proved very precisely but at the price of significantly increasing the complexity of statements and their proofs while at the same time without changing conclusions.

PROPOSITION 3.1.1. *Let p_{pr} be the success probability in Step 3 of the above algorithm. Under Assumption 3.1.1, we have*

$$p_{\mathrm{pr}} = \Theta\left(\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}\right)$$

for a density $1-2^{-\Omega(n)}$ of matrices $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, where

$$j \stackrel{def}{=} \begin{cases} 0 & \text{if } t < \frac{q-1}{q}(n-k) \\ t - \frac{q-1}{q}(n-k) & \text{if } t \in [\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!] \\ k & \text{otherwise.} \end{cases}$$

PROOF. Notice that input $(\mathbf{H}, \mathbf{s})$ is fixed, the randomness of the algorithm comes from $\mathbf{x}$ picked according to $\mathscr{D}_t$ and the drawing of the information set $\mathscr{I}$. Under Assumption 3.1.1, our probability computations over $\mathscr{I}$ are up to a polynomial factor given by the case where $\mathscr{I}$ is uniformly distributed (we will not write the polynomial during the computations).

Let us fix $\mathbf{e}^{(1)}$ to be a solution of our decoding problem (we know that there is at least one). To compute the success probability of Prange's algorithm let us first notice that an iteration will succeed if $\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}$, namely

$$\mathbb{P}\left(\text{an iteration of Prange finds } \mathbf{e}^{(1)}\right) = \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}\right) \tag{3.9}$$

It comes from the fact that $\mathbf{e}^{(1)}$ is uniquely determined by $\mathbf{e}^{(1)}_{\mathscr{I}}$ as necessarily $\mathbf{e}^{(1)}_{\overline{\mathscr{I}}} = \left(\mathbf{s} - \mathbf{e}^{(1)}_{\mathscr{I}}\mathbf{H}^{\intercal}_{\mathscr{I}}\right)\mathbf{S}^{\intercal}$. Furthermore, $\mathscr{D}_t$ only outputs vectors $\mathbf{x}$ of Hamming weight j. To find $\mathbf{e}^{(1)}$ it is necessary that during an iteration we have $\left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j$ as $|\mathbf{e}^{(1)}| = t$ and $|\mathbf{x}| = j$. Therefore, using the law of total probability, we obtain the following computation

$$\mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}\right) = \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}} \mid \left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j\right)\mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j\right) \tag{3.10}$$

The probability to find $\mathbf{e}^{(1)}$ in one iteration is given by the probability that

- (i) $\mathscr{I}$ is such that $\left|\mathbf{e}^{(1)}_{\mathscr{I}}\right| = t - j$
- (ii) $\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}}$ supposing (i).

Under Assumption 3.1.1, the probability of (i) is $\frac{\binom{t}{j}\binom{n-t}{k-j}}{\binom{n}{k}}$ while the probability of (ii) is given by $\frac{1}{\binom{k}{j}(q-1)^j}$ as $\mathbf{x} \in \mathbb{F}_q^k$ picked according to $\mathscr{D}_t$ is uniformly distributed among words of Hamming weight j. Plugging this in Equation (3.9) and using Equation (3.10) we obtain the following computation

$$\begin{aligned}\mathbb{P}\left(\text{an iteration of Prange finds } \mathbf{e}^{(1)}\right) &= \mathbb{P}_{\mathscr{I},\mathbf{x}}\left(\mathbf{x} = \mathbf{e}^{(1)}_{\mathscr{I}} \mid |\mathbf{e}_{\mathscr{I}}| = t-j\right)\mathbb{P}_{\mathscr{I}}\left(|\mathbf{e}_{\mathscr{I}}| = t-j\right)\\ &= \frac{1}{\binom{k}{j}(q-1)^j}\,\frac{\binom{t}{j}\binom{n-t}{k-j}}{\binom{n}{k}}\\ &= \frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\end{aligned}$$

where the last equality follows from a simple computation.

Recall now that we are sure that there is at least one solution of the decoding problem. However, depending on t, it may happen that there are more. Let us denote by N the number of solutions. According to the above equation, the probability to find none of these in one iteration of the algorithm is given by

$$\left(1 - \frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right)^N = 1 - \Theta\left(N\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right)$$

Here we used that the randomness $\mathscr{I}$, $\mathbf{x}$ of the algorithm is independent of the solutions. Therefore, the probability p_{pr} that Prange's algorithm succeeds is

$$p_{\mathrm{pr}} = \Theta\left(N\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}(1+o(1))\right) \tag{3.11}$$

But now recall from Chapter 2 [(5)] that for any constant C,

$$\mathbb{P}_{\mathbf{H}}\left(\left|N - \max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\right| > C\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\right) = 2^{-\Omega(n)}$$

Therefore, since $\mathbf{H}$ is uniformly distributed in the above probability, we have for a density $1-2^{-\Omega(n)}$ of matrices $\mathbf{H}$,

$$p_{\mathrm{pr}} = \Theta\left(\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)\,\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\binom{n}{t}(q-1)^t}\right) = \Theta\left(\frac{\binom{n-k}{t-j}(q-1)^{t-j}}{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}\right)$$

where we used Equation (3.11). It concludes the proof. □

We are now ready to give the running-time of Prange's algorithm to solve DP. It will be a simple consequence of the above proposition.

COROLLARY 3.1.1. *Under Assumption 3.1.1, the complexity $C_{\mathrm{Prange}}(n,q,R,\tau)$ of Prange's algorithm to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*

$$\frac{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}{\binom{n-k}{t-j}(q-1)^{t-j}}$$

where $k \overset{def}{=} \lfloor Rn \rfloor$, $t \overset{def}{=} \lfloor \tau n \rfloor$ and

$$j \overset{def}{=} \begin{cases} 0 & \text{if } t < \frac{q-1}{q}(1-R)\\ t - \frac{q-1}{q}(n-k) & \text{if } t \in [\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]\\ k & \text{otherwise.}\end{cases}$$

[(5)] Depending on which term achieves the maximum, we use Propositions 2.3.2 or 2.3.3.

PROOF. The cost of an iteration of Prange's algorithm is dominated by the time to perform a Gaussian elimination. Let p_{pr} be the probability of success of an iteration which is given in Proposition 3.1.1. The number of iterations is (up to a polynomial in n) $1/p_{\text{pr}}$ with a probability exponentially close to one (in n). The latter affirmation comes from the fact that the number of iterations is a geometric distribution. □

To conclude this section let us briefly study the asymptotic complexity (in n) of Prange's algorithm.

Asymptotic complexity: use the entropy function. When studying the asymptotic complexity of ISD algorithms it will be important to be familiar with the q-ary entropy function and its properties. Recall from Chapter 2 that it is defined as (and extended by continuity)

$$h_q : x \in [0,1] \longmapsto -x \log_q \left(\frac{x}{q-1}\right) - (1-x)\log_q(1-x).$$

The q-ary entropy is an increasing function over $\left[0, \frac{q-1}{q}\right]$ and a decreasing function over $\left[\frac{q-1}{q}, 1\right]$. It reaches its maximum 1 in $\frac{q-1}{q}$.

This function has the nice property to describe the asymptotic behaviour of binomials, namely (see Lemma 2.1.1)

$$\frac{1}{n}\log_q \binom{n}{t}(q-1)^t \underset{n\to+\infty}{=} h_q(\tau) + O\left(\frac{\log_q n}{n}\right) \tag{3.12}$$

where $\tau = t/n$. From this we easily deduce the exponent of Prange's algorithm

$$\frac{1}{n}\log_q C_{\text{Prange}}(n,q,R,\tau) \underset{n\to+\infty}{=} \min\left(1-R, h_q(\tau)\right) - (1-R)\, h_q\left(\frac{\tau-\gamma}{1-R}\right) + O\left(\frac{\log_q n}{n}\right) \tag{3.13}$$

where,

$$\gamma \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } \tau < \frac{q-1}{q}(n-k) \\ \tau - \frac{q-1}{q}(1-R) & \text{if } \tau \in \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right] \\ R & \text{otherwise.} \end{cases}$$

There are particular ranges of parameters for which Equation (3.12) simplifies. First, in the case where $\tau \in \left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$, this exponent is a $O\left(\frac{\log_q n}{n}\right)$ (it is easily verified using that $h_q((q-1)/q) = 1$). It corresponds to what we have expected as we claimed that Prange's algorithm is polynomial in this range of parameters.

We used Equation (3.13) multiplied by a factor $\log_2(q)$ (exponents are in base 2) to draw Figures 3.2, 3.3 and 3.4. Notice in the case $q = 2$ that the complexity of Prange's algorithm is symmetric around $1/2$ which is expected in that case. Considering short or large weight in the binary case is equivalent as you have to show in the following exercise. In particular, in the sequel we will not compute the exponents of algorithms for solving $\mathsf{DP}(n, 2, R, \tau)$ with $\tau > 1/2$.

EXERCISE 3.1.2. *Let $\tau \in [0, 1/2]$. Show how from an algorithm solving $\mathsf{DP}(n, 2, R, \tau)$ we can deduce an algorithm solving $\mathsf{DP}(n, 2, R, 1-\tau)$ in the same running-time (and reciprocally).*

Let us consider now the case $\tau = o(1)$ (and therefore $\gamma = 0$). It corresponds to parameters of all code-based public-key encryption schemes (for instance [**McE78**, **Ale03**, **MTSB13**, **AAB+17**]). Using

$$h_q(x) \underset{x\to 0}{=} -x\log_q\left(\frac{x}{q-1}\right) + x + o(x) \tag{3.14}$$

and Equation (3.13) (with $\gamma = 0$) we obtain the following computation,

$$\begin{aligned}\frac{1}{n}\log_q C_{\text{Prange}}(n,q,R,\tau) &= \min\left(1-R, h_q(\tau)\right) - (1-R)\, h_q\left(\frac{\tau}{1-R}\right) + O\left(\frac{\log_q n}{n}\right)\\ &= h_q(\tau) - (1-R)\, h_q\left(\frac{\tau}{1-R}\right) + O\left(\frac{\log_q n}{n}\right)\\ &= -\tau\log_q\left(\frac{\tau}{q-1}\right) + \tau\log_q\left(\frac{\tau}{1-R}\,\frac{1}{q-1}\right) + o(\tau) + O\left(\frac{\log_q n}{n}\right) \text{ (see Eq. (3.14))}\\ &= -\tau\log_q(1-R) + o(\tau) + O\left(\frac{\log_q n}{n}\right).\end{aligned}$$

Therefore, when $\tau = o(1)$ ($t = \tau n$), the complexity of Prange algorithm is given by (for some constant C)

$$C_{\text{Prange}}(n,q,R,\tau) = n^C\, q^{-t\log_q(1-R)}.$$

It is even more remarkable that no algorithm is known to have a complexity $q^{ct(1+o(1))}$ with $c < -\log_q(1-R)$ as soon as $t = o(n)$. Furthermore, all known ISD (even the most sophisticated) have the same asymptotic complexity than Prange's algorithm for these parameters [**CS16**]. Despite its extreme simplicity, Prange's algorithm is the best known algorithm to solve asymptotically $\mathsf{DP}(n,q,R,\tau)$ when the decoding distance is sub-linear, namely $\tau = o(1)$.

3.2. Birthday Paradox Techniques

We present in this section two algorithms for solving $\mathsf{DP}(n,q,R,\tau)$. Both rely on the following *crucial* lemma which is essentially an average version of the birthday paradox

LEMMA 3.2.1. *Let $\mathscr{L}_1, \mathscr{L}_2$ be two lists of L random and independent elements in $\mathbb{F}_q^r$. We have,*

$$\mathbb{E}\left(\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2\right) = \frac{L^2}{q^r}.$$

Notice that we expect one element in the intersection of the two lists included in $\mathbb{F}_q^r$ when their size verifies $L = \sqrt{q^r}$. Recall that the birthday paradox, asserts that when there are $\sqrt{N}$ elements picked uniformly at random among a set of size N, we will get with a good probability two equal elements. It explains why we refer to the above lemma as the birthday paradox.

PROOF. By definition, $\mathscr{L}_1 = \{X_1, \ldots, X_L\}$ and $\mathscr{L}_2 = \{Y_1, \ldots, Y_L\}$ where the X_i's and Y_j's are independent and uniformly distributed random variables taking their values in $\mathbb{F}_q^r$. We have

$$\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2 = \sum_{i,j=1}^{L} \mathbb{1}_{\{X_i = Y_j\}}.$$

By linearity of the expectation we have the following computation,

$$\mathbb{E}\left(\sharp\, \mathscr{L}_1 \cap \mathscr{L}_2\right) = \sum_{i,j=1}^{L} \mathbb{E}\left(\mathbb{1}_{\{X_i = Y_j\}}\right) = \sum_{i,j=1}^{L} \frac{1}{q^r} = \frac{L^2}{q^r}$$

which concludes the proof. □

3.2.1. Dumer's Algorithm. Let us now quickly present Dumer's algorithm [Dum86] to solve $\mathsf{DP}(n,q,R,\tau)$. This short subsection may be skipped as the description of this algorithm has already been given in the introduction (in the same fashion).

The algorithm.

1. *Splitting in two parts.* First we randomly select a set $\mathscr{S} \subseteq [\![1,n]\!]$ of $n/2$ positions.
2. *Building lists step.* We build,
$$\mathscr{L}_1 \stackrel{\text{def}}{=} \left\{ \mathbf{H}_{\mathscr{S}} \mathbf{e}_1^\intercal : |\mathbf{e}_1| = \frac{t}{2} \right\} \quad ; \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{ -\mathbf{H}_{\overline{\mathscr{S}}} \mathbf{e}_2^\intercal + \mathbf{s}^\intercal : |\mathbf{e}_2| = \frac{t}{2} \right\}.$$
3. *Collisions step.* We merge the above lists (with an efficient technique like hashing or sorting)
$$\mathscr{L}_1 \bowtie \mathscr{L}_2 \stackrel{\text{def}}{=} \left\{ (\mathbf{e}_1, \mathbf{e}_2) \in \mathscr{L}_1 \times \mathscr{L}_2, \quad \mathbf{H}_{\mathscr{S}} \mathbf{e}_1^\intercal = -\mathbf{H}_{\overline{\mathscr{S}}} \mathbf{e}_2^\intercal + \mathbf{s}^\intercal \right\}.$$
and output this new list. If it is empty we go back to Step 1 and pick another set of $n/2$ positions.

PROPOSITION 3.2.1. *The complexity $C_{\text{Dumer}}(n,q,R,\tau)$ of Dumer's algorithm to solve $\mathsf{DP}(n,q,R,\tau)$ is up to a polynomial factor (in n) given by*
$$\sqrt{\binom{n}{t}(q-1)^t} + \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}$$
Furthermore, Dumer's algorithm finds $\max\left(1, \frac{\binom{n}{t}(q-1)^t}{q^{n-k}}\right)$ *solutions (up to a polynomial factor in n) where $k \stackrel{def}{=} \lfloor Rn \rfloor$ and $t \stackrel{def}{=} \lfloor \tau n \rfloor$.*

PROOF. Dumer's algorithm will find a fixed solution of the considered decoding problem with probability
$$\frac{\binom{t}{t/2}\binom{n-t}{n/2-t/2}}{\binom{n}{n/2}} = 2^{th_2(1/2)+(n-t)h_2(1/2)-nh_2(1/2)+O(\log_2 n)} = 2^{O(\log_2 n)}$$
which is polynomial. Therefore the number of iterations of Dumer's algorithm to find a solution will be polynomial.

The cost of one iteration is given by the time to build lists $\mathscr{L}_1, \mathscr{L}_2$, namely $\binom{n/2}{t/2}(q-1)^{t/2}$ plus the time to merge them. With efficient techniques such as sorting or hashing this can be done in time $\sharp\, \mathscr{L}_1 \bowtie \mathscr{L}_2$. But, according to Lemma 3.2.1, the expected size of $\mathscr{L}_1 \bowtie \mathscr{L}_2$ is in average over $\mathbf{H}$ given by $\left(\binom{n/2}{t/2}(q-1)^{t/2}\right)^2 / q^{n-k}$ which is equal to (up to a polynomial factor) $\binom{n}{t}(q-1)^t/q^{n-k}$ as collisions are made on vectors which belong to $\mathbb{F}_q^{n-k}$. It concludes the proof. □

REMARK 3.2.1. *We have presented Dumer's algorithm to find all solutions of* DP. *But one can also tweak this algorithm to build less solutions in one iteration.*

EXERCISE 3.2.1. *We have made the choice when presenting Dumer's algorithm to build lists of maximum size, namely $\binom{n/2}{t/2}(q-1)^{t/2}$ (why is it the largest possible list size?). Let $(\mathbf{H},\mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ be an instance of a decoding problem that we would like to solve at distance t. Show that a slight variation of Dumer's algorithm enables to compute* $\max\left(1, \frac{L^2}{q^{n-k}}\right)$ *solutions in time $L + \max\left(1, \frac{L^2}{q^{n-k}}\right)$ (up to polynomial factors). How L needs to be chosen for this algorithm to output solutions in amortized time one? Deduce a necessary condition over t for this to be possible.*

3.2.2. Wagner's Algorithm. We have just seen that Dumer's algorithm finds all solutions of DP in roughly one iteration. It is an extremely nice property but that may be an impediment in some contexts. Suppose that one needs M solutions of DP to achieve some task. The best situation would be to find them in amortized time one. Suppose now that Dumer's algorithm is able to compute N solutions of DP in amortized time one, namely it builds lists of size N which verify

$$N = \frac{N^2}{q^{n-k}} \iff N = q^{n-k}$$

but unfortunately $N \gg M$. In other words, Dumer's algorithm finds too many solutions. To avoid this useless situation we may be tempted to decrease the size of the built lists, namely N, to decrease the number of output solutions. However by doing this we would not produce decoding solutions in amortized time one, which would be less efficient for our purpose. To improve this situation, the fundamental remark is that Dumer's algorithm produces all its solutions with a shape $(\mathbf{e}_1, \mathbf{e}_2)$, where $|\mathbf{e}_1| = |\mathbf{e}_2| = t/2$, and by looking at collisions directly on $n-k$ symbols. The idea to produce less solutions, still in amortized time one, is to look for solutions with more constraints on their shapes and the way that collisions are built. It is precisely the idea of Wagner's algorithm [**Wag02**] (producing less solutions in amortized time one by decimating the search space) that we are going to present precisely in the sequel. However, as a picture is better than a long discourse, let us first describe in Figure 3.5 a simplified version of this algorithm when we try to find $\mathbf{e}$ of Hamming weight t such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{0}$.

The output of Wagner's algorithm described in Figure 3.5 is $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)$. It is a solution as by construction it reaches the syndrome $\mathbf{0}$ with respect to $\mathbf{H}$ and it has the right Hamming weight since each $\mathbf{e}_i$ has weight $t/4$. Notice that this solution has a particular "shape" when compared to the output of Dumer's algorithm. During Steps 2 and 3 we do not perform collisions on all the $n-k$ symbols of the syndromes but on $(n-k)/2$ symbols. Therefore solutions have the following property: $(\mathbf{H}_1\ \mathbf{H}_2)\,(\mathbf{e}_1, \mathbf{e}_2)^\intercal$ and $(\mathbf{H}_3\ \mathbf{H}_4)\,(\mathbf{e}_3, \mathbf{e}_4)^\intercal$ are equal to 0 on the last $(n-k)/2$ positions. If one splits an output of Dumer's algorithm and the parity-check matrix in four parts, there is no reason that it verifies the above property. It will be true only for an exponentially small fraction of the solutions. It explains why Wagner's algorithm "decimates" the solutions. At the same time this algorithm has the advantage to be able to produce solutions in amortized time one. The idea in that case is to build the lists $\mathscr{L}_i$'s with size L such $L^2/q^{(n-k)/2} = L$, *i.e.* $L = \sqrt{q^{n-k}}$. Therefore, each collision step has the same cost given by the size L of the lists that are output. However it may happen that the number of solutions is still too large. If so, the next idea of Wagner's algorithm is to consider more lists at the beginning, like 8 (and not 4) and then to make collisions on $(n-k)/3$ symbols. It enables smaller lists, namely $L^2/q^{(n-k)/3} = L$, *i.e.* $L = \sqrt[3]{q^{n-k}}$. However, in that case, there will be three steps of collisions. Then we can extend this by considering a number of lists given by some 2^a and making a steps of collisions. Nonetheless, it is not possible to do this for any a. If one uses Wagner's algorithm with initially 2^a lists, all of them need to be built from vectors $\mathbf{e}_i$ of Hamming weight $t/2^a$. If a is too large it will be impossible to build lists large enough to produce collisions in amortized time one.

Let us emphasize that the above discussion is not only a thought exercise. It turns out that the above situation happens with ISD algorithms (Dumer's algorithm produces too many solutions in one iteration). It explains why the ISD with Wagner's algorithm outperforms the ISD with Dumer's algorithm for some parameters as we can see in Figure 3.4 (in particular when DP is such that there are a lot ofh solutions).

PROPOSITION 3.2.2. *Wagner's algorithm solves* $\mathsf{DP}(n, q, R, \tau)$ *by (where* $k \stackrel{def}{=} \lfloor Rn \rfloor$*)*

$$\mathbf{H} = (\mathbf{H}_1\ \mathbf{H}_2\ \mathbf{H}_3\ \mathbf{H}_4) \text{ where the } \mathbf{H}_i \in \mathbb{F}_q^{(n-k)\times n/4}$$

1st Step: compute the following lists

$$\mathscr{L}_1 \stackrel{\text{def}}{=} \{\mathbf{H}_1\mathbf{e}_1^\intercal : |\mathbf{e}_1| = t/4\} \quad \mathscr{L}_2 \stackrel{\text{def}}{=} \{\mathbf{H}_2\mathbf{e}_2^\intercal : |\mathbf{e}_2| = t/4\} \quad \mathscr{L}_3 \stackrel{\text{def}}{=} \{\mathbf{H}_3\mathbf{e}_3^\intercal : |\mathbf{e}_3| = t/4\} \quad \mathscr{L}_4 \stackrel{\text{def}}{=} \{\mathbf{H}_4\mathbf{e}_4^\intercal : |\mathbf{e}_4| = t/4\}$$

2nd Step: compute the following lists obtained by collision

$(\mathbf{H}_1\mathbf{e}_1^\intercal, \mathbf{H}_2\mathbf{e}_2^\intercal) \in \mathscr{L}_1 \times \mathscr{L}_2$ s.t

$\mathbf{H}_1\mathbf{e}_1^\intercal =$  $\mathbf{H}_2\mathbf{e}_2^\intercal =$ - 

$\longrightarrow (\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal =$ 

$(\mathbf{H}_3\mathbf{e}_3^\intercal, \mathbf{H}_4\mathbf{e}_4^\intercal) \in \mathscr{L}_3 \times \mathscr{L}_4$ s.t

$\mathbf{H}_3\mathbf{e}_3^\intercal =$  $\mathbf{H}_4\mathbf{e}_4^\intercal =$ -  $(n-k)/2$

$\longrightarrow (\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal =$ 

3rd Step: compute the following lists obtained by collision

$(\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal$ and $(\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal$ in the above lists s.t

$(\mathbf{H}_1\ \mathbf{H}_2)(\mathbf{e}_1, \mathbf{e}_2)^\intercal =$ 

$(\mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_3, \mathbf{e}_4)^\intercal =$ - 

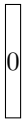
$\longrightarrow (\mathbf{H}_1\ \mathbf{H}_2\ \mathbf{H}_3\ \mathbf{H}_4)(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4)^\intercal =$ 

FIGURE 3.5. Simplified version of Wagner's algorithm with four layers to find $\mathbf{e}$ of weight t such that $\mathbf{He} = \mathbf{0}$. The same colours on vectors means that they are equal (be careful on the minus signs).

(1) finding one solution in time and space $q^{\frac{n-k}{a+1}}$ (up to a polynomial factor in n) for any integer a such that $q^{\frac{n-k}{a+1}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$ which asymptotically can be written as,

$$\frac{1-R}{h_q(\tau)} \leq \frac{a+1}{2^a}.$$

(2) finding $q^{\frac{n-k}{a}}$ solutions in amortized time 1 (up to a polynomial factor in n) for any integer a such that $q^{\frac{n-k}{a}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$ which asymptotically can be written as,

$$\frac{1-R}{h_q(\tau)} \leq \frac{a}{2^a}.$$

During the description of Wagner's algorithm that follows (which will give the proof of the above proposition) we will use Lemma 3.2.1 to estimate the size of the lists after merging.

Wagner's algorithm. The first step is to split $\mathbf{H}$ in 2^a parts of the same size, for a parameter a that is called *depth of the algorithm*. For the sake of simplicity let us split $\mathbf{H}$ as (we can choose the partition)

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & \dots & \mathbf{H}_{2^a} \end{pmatrix}$$

where for all i we have $\mathbf{H}_i \in \mathbb{F}_q^{(n-k)\times \frac{n}{2^a}}$. Then we build the following 2^a-lists for some parameter L that will be fixed later ($t = \lfloor \tau n \rfloor$)

$$\forall i \in [\![1, 2^a]\!], \quad \mathscr{L}_i \subseteq \left\{ \mathbf{e}\mathbf{H}_i^\intercal \,:\, \mathbf{e} \in \mathbb{F}_q^{n/2^a},\ |\mathbf{e}| = \frac{t}{2^a} \right\} \quad \textit{and} \quad \sharp\, \mathscr{L}_i = L.$$

Notice that by construction we have the following constraint,

$$L \leq \binom{n/2^a}{t/2^a} (q-1)^{t/2^a}. \tag{3.15}$$

Let $\ell \in [\![1, n-k]\!]$ be a parameter that will be chosen later. Then, Wagner's algorithm performs the collision of these lists two by two on their last ℓ symbols [(6)] to build the new lists $\mathscr{L}_{i,i+1}$'s, namely

$$\mathscr{L}_{i,i+1} \stackrel{\text{def}}{=} \{\mathbf{s}_i + \mathbf{s}_{i+1} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } \ell \text{ symbols of } \mathbf{s}_i + \mathbf{s}_{i+1} \text{ are } \mathbf{0}\},$$

where by construction we have access to the errors $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ of Hamming weight $t/2^a$ that reach $\mathbf{s}_i$ and $\mathbf{s}_{i+1}$ through $\mathbf{H}_i$ and $\mathbf{H}_{i+1}$. The last list $\mathscr{L}_{2^a-1,2^a}$ is built by merging $\mathscr{L}_{2^a}$ and $\mathscr{L}_{2^a-1}$ but this time according to the last ℓ symbols of $\mathbf{s}$, namely

$$\mathscr{L}_{2^a-1,2^a} \stackrel{\text{def}}{=} \{\mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } \ell \text{ symbols of } \mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \text{ are equal to those of } \mathbf{s}\}.$$

Using Lemma 3.2.1, these new lists built after merging on ℓ symbols will be of the same size,

$$\frac{L^2}{q^\ell}.$$

Furthermore, we produce them at cost $L + \frac{L^2}{q^\ell}$ (up to a polynomial factor). Once this is done, we start again this process $a-2$ times by merging each time on the ℓ next new symbols. Wagner proposed to choose L such that at each step, *the time for merging is the same than the one to build lists*, namely

$$L = q^\ell. \tag{3.16}$$

This implies under Constraint (3.15) that the parameter ℓ is such that

$$q^\ell \leq \binom{n/2^a}{t/2^a} (q-1)^{t/2^a}. \tag{3.17}$$

Remark 3.2.2. *One may ask if this strategy of an amortized time one at each merge is optimal. It turns out that the answer is yes as proved in* [MS09].

Up to now we have made $a-1$ merges and we still have two lists, that we denote by $\mathscr{S}_1$ and $\mathscr{S}_2$. They are such that

$$\mathscr{S}_1 = \{\mathbf{s}_1 + \dots + \mathbf{s}_{2^{a-1}} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } (a-1)\ell \text{ symbols of } \mathbf{s}_1 + \dots + \mathbf{s}_{2^{a-1}} \text{ are equal to } \mathbf{0}\}$$

$$\mathscr{S}_2 = \{\mathbf{s}_{2^{a-1}+1} + \dots + \mathbf{s}_{2^a} \,:\, \mathbf{s}_i \in \mathscr{L}_i \text{ and the last } (a-1)\ell \text{ symbols of } \mathbf{s}_1 + \dots + \mathbf{s}_{2^{a-1}} \text{ are equal to those of } \mathbf{s}\}$$

$$\text{where,} \qquad \sharp\, \mathscr{S}_1 = \sharp\, \mathscr{S}_2 = \frac{L^2}{q^\ell} = L = q^\ell.$$

[(6)] Given a vector $\mathbf{x} \in \mathbb{F}_q^m$, it denotes $x_{m-\ell+1}, \dots, x_m$.

Therefore, it remains to merge these two lists on the last $(n-k)-(a-1)\ell$ first symbols. It yields in time $\frac{q^{\ell(a+1)}}{q^{(n-k)}}$ a list of solutions of size

$$\frac{q^{2\ell}}{q^{(n-k)-(a-1)\ell}} = \frac{q^{\ell(a+1)}}{q^{(n-k)}}. \tag{3.18}$$

Now the parameter ℓ has to be set whether one wants only one solution or many solutions in amortized time one.

Wagner to reach one solution. According to Equation (3.18), it remains to choose parameters such that

$$\ell = \frac{n-k}{a+1}.$$

All the lists in the $a-1$ first steps of the algorithm have the same size, namely $L = q^{\ell}$ (Equation (3.16)), therefore the algorithm has a cost given by

$$q^{\frac{n-k}{a+1}}.$$

However, we have to be careful with the depth a of the algorithm, unfortunately it cannot be chosen too large. According to Equation (3.15)

$$q^{\ell} = q^{\frac{n-k}{a+1}} \leq \binom{n/2^a}{t/2^a}(q-1)^{t/2^a}$$

which leads to the following asymptotic constraint,

$$\frac{1-R}{a+1} \leq \frac{1}{2^a}\, h_q(\tau) \iff \frac{1-R}{h_q(\tau)} \leq \frac{a+1}{2^a}.$$

It concludes the proof of (1).

Wagner to compute many solutions in amortized time one. In this case, according to Equation (3.18), we just need to choose ℓ such that

$$q^{\ell} = \frac{q^{\ell(a+1)}}{q^{n-k}} \iff \ell = \frac{n-k}{a}$$

As above we obtain the claimed constraint on a. It concludes the proof. □

We draw in Figure 3.6 the exponent of Wagner's algorithm to solve $\mathsf{DP}(n, q, R, \tau)$ as function of $\tau \geq \tau^-$ (the relative Gilbert-Varshamov distance defined in Equation (2.4)). We choose parameters of the algorithm to output one solution and a being the largest integer that satisfies the constraint (1) of Proposition 3.2.2 (to have an optimal complexity). As it can be seen the exponent is a decreasing function of τ. Indeed, when τ increases, a can be chosen larger. Furthermore, there is a discontinuity in the exponent. It comes from the fact that a is an *integer*. It is possible to adapt the algorithm to "smooth" its complexity but this is out of scope of these lecture notes.

3.3. Combining Linear Algebra and Birthday Paradox Techniques

We are now ready to present the general framework (introduced in [**FS09**]) of *Information Set Decoding* (ISD) algorithms.

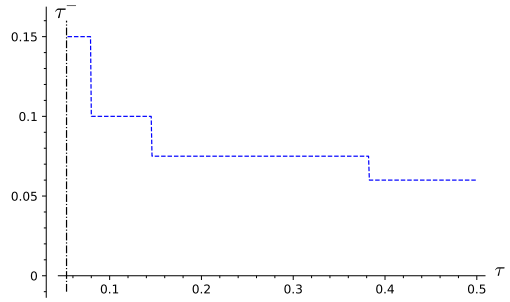


FIGURE 3.6. Exponent of Wagner's algorithm to solve $\mathsf{DP}(n, q, R, \tau)$ for $R = 0.7$ as function of τ.

The algorithm. Let us introduce the following parameters,

$$\ell \in [\![0, n-k]\!] \quad \text{and} \quad p \in [\![0, \min(t, k+\ell)]\!]$$

1. *Picking the augmented information set.* Let $\mathscr{J} \subseteq [\![1, n]\!]$ be a random set of size $k+\ell$. If $\mathbf{H}_{\overline{\mathscr{J}}} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ is not of full-rank, pick another set $\mathscr{J}$.
2. *Linear algebra.* Perform a Gaussian elimination to compute a non-singular matrix $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ such that $\mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}} = \begin{pmatrix} \mathbf{1}_{n-k-\ell} \\ \mathbf{0}_{\ell\times(n-k-\ell)} \end{pmatrix}$. Let $\mathbf{H}' \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$, $\mathbf{H}'' \in \mathbb{F}_q^{\ell\times(k+\ell)}$, $\mathbf{s}' \in \mathbb{F}_q^{n-k-\ell}$ and $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$ be such that

$$\mathbf{S}\mathbf{H}_{\mathscr{J}} = \begin{pmatrix} \mathbf{H}' \\ \mathbf{H}'' \end{pmatrix} \quad \text{and} \quad \mathbf{S}\mathbf{s}^\intercal = (\mathbf{s}', \mathbf{s}'')^\intercal \tag{3.19}$$

3. *Sub-decoding problem.* Compute a set,

$$\mathscr{S} \subseteq \left\{ \mathbf{e}'' \in \mathbb{F}_q^{k+\ell} \ : \ \mathbf{e}''\mathbf{H}''^\intercal = \mathbf{s}'' \text{ and } |\mathbf{e}''| = p \right\}. \tag{3.20}$$

4. *Test.* Find $\mathbf{e}'' \in \mathscr{S}$ such that $\left|\mathbf{s}' - \mathbf{e}'\mathbf{H}''^\intercal\right| = t - p$. If not, return to Step 1; otherwise output $\mathbf{e} \in \mathbb{F}_q^n$ such that

$$\mathbf{e}_{\overline{\mathscr{J}}} = \mathbf{s}' - \mathbf{e}''\mathbf{H}'^\intercal \quad ; \quad \mathbf{e}_{\mathscr{J}} = \mathbf{e}'' \tag{3.21}$$

Correction of the algorithm. It easily follows from the following computation,

$$\begin{aligned}
\mathbf{S}\mathbf{H}\mathbf{e}^\intercal &= \mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}}\mathbf{e}_{\overline{\mathscr{J}}}^\intercal + \mathbf{S}\mathbf{H}_{\mathscr{J}}\mathbf{e}_{\mathscr{J}}^\intercal \\
&= \begin{pmatrix} \mathbf{1}_{n-k-\ell} \\ \mathbf{0}_{\ell\times(n-k-\ell)} \end{pmatrix} \left(\mathbf{s}'^\intercal - \mathbf{H}''\mathbf{e}''^\intercal\right) + \begin{pmatrix} \mathbf{H}' \\ \mathbf{H}'' \end{pmatrix} \mathbf{e}''^\intercal \quad \text{(By definition of } \mathbf{S}\mathbf{H}_{\overline{\mathscr{J}}}, \mathbf{S}\mathbf{H}_{\mathscr{J}}, \mathbf{e}_{\overline{\mathscr{J}}} \text{ and } \mathbf{e}_{\mathscr{J}}) \\
&= \begin{pmatrix} \mathbf{s}'^\intercal - \mathbf{H}''\mathbf{e}''^\intercal \\ \mathbf{0}_{\ell\times(n-k-\ell)} \end{pmatrix} + \begin{pmatrix} \mathbf{H}'\mathbf{e}''^\intercal \\ \mathbf{H}''\mathbf{e}''^\intercal \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{s}'^\intercal \\ \mathbf{s}''^\intercal \end{pmatrix} \qquad \text{(By definition of } \mathbf{e}'' \in \mathscr{S}\text{, see Equation (3.20))} \\
&= \mathbf{S}\mathbf{s}^\intercal \qquad \text{(By Equation (3.19))}
\end{aligned}$$

which corresponds to $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal$ since $\mathbf{S}$ is non-singular. Furthermore, by definition $\mathbf{e}''$ has Hamming weight p and the test ensures that $\mathbf{s}' - \mathbf{e}''\mathbf{H}''^\intercal$ has weight $t-p$. Therefore, once the algorithm terminates, $\mathbf{e}$ reaches $\mathbf{s}$ with respect to $\mathbf{H}$ and has Hamming weight t.

EXERCISE 3.3.1. *Let*

$$\mathscr{D} \stackrel{def}{=} \left\{ \mathbf{c}'' \in \mathbb{F}_q^{k+\ell} \ : \ \mathbf{c}''\mathbf{H}''^\intercal = \mathbf{0} \right\}.$$

Show that $\mathscr{D}$ is a code of length $k+\ell$ and dimension k.

REMARK 3.3.1. *The code of parity-check matrix $\mathbf{H}''$ is known as the punctured code (defined by the parity-check matrix $\mathbf{H}$) at the positions $\overline{\mathscr{J}}$. Computing $\mathscr{S}$ in Equation (3.20) amounts to solve a decoding problem at distance p with this input code and the syndrome $\mathbf{s}''$. Therefore, for each drawn of the augmented information set $\mathscr{J}$ we test many decoding candidates (given by elements of the list $\mathscr{S}$ and with associated lift defined in Equation (3.21)). We recover the interpretation of* ISD *algorithms given in the introduction.*

Far or close codeword? One may wonder why don't we use the distribution $\mathscr{D}_t$ in ISD algorithms to be able to produce "short" or "large" solutions? To answer this question let us take a look at the typical weight of a vector $\mathbf{e}$ that will pass the test at the end of an iteration (see Equation (3.21)). By supposing that $\mathbf{s}$ is uniformly distributed, we have

$$\mathbb{E}\left(|\mathbf{e}|\right) = p + \frac{q-1}{q}\ (n-k-\ell) \tag{3.22}$$

The $\frac{q-1}{q}\ (n-k-\ell)$ term comes from the fact that $\mathbf{s}'$ is uniformly distributed over $\mathbb{F}_q^{n-k-\ell}$ while p is here as by definition $\mathbf{e}''$ has weight p. If one wants to get a solution of small weight, the best approach is to decode the punctured code at a small as possible distance p. On the other hand, if one seeks a solution of large weight, one has to decode this punctured code at the largest as possible distance, namely $p = k+\ell$. Therefore the strategy to reach short or large error relies on how we choose the parameter p.

The above discussion hints us why we can not reasonably hope, with ISD algorithms, to solve DP in polynomial time outside the interval $[\![\frac{q-1}{q}\ (n-k), k + \frac{q-1}{q}\ (n-k)]\!]$. For instance, if one is looking for an ISD algorithm solving DP in polynomial time for some $t < \frac{q-1}{q}(n-k)$, one has according to Equation (3.22) to find a subroutine decoding in polynomial time a random code of length $k+\ell$ and dimension k at distance p such that

$$p - \frac{q-1}{q}\ell < 0.$$

But at the same time, the smaller p for which we known how to decode in polynomial a random $[k+\ell, k]_q$-code is precisely $\frac{q-1}{q}\ (k+\ell-k) = \frac{q-1}{q}\ \ell$ which is the above limit to get an improvement. Therefore, if one seeks an ISD enlarging the interval of weights in which Prange's algorithm is polynomial, one has to first enlarge this interval.

Analyse of the algorithm. As in Prange's algorithm, all the challenge in the analysis of ISD algorithms running time relies on the computation of the success probability in Step 4. However, contrary to Prange's algorithm it will not be necessary to make an assumption on how the augmented information sets are picked. We can suppose directly, when $\ell = \Theta(n)$ (which will be the case in our applications), that they are uniformly distributed as shown by the following proposition.

PROPOSITION 3.3.1. *Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and $\mathscr{J} \subseteq [\![1,n]\!]$ being uniformly distributed over the sets of size $k+\ell$ (where $\ell > 0$) such that $\mathbf{H}_{\overline{\mathscr{J}}}$ is non-singular. Let $\mathscr{J}_{\text{unif}} \subseteq [\![1,n]\!]$ being uniformly distributed over the sets of size $k+\ell$. We have,*

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathscr{J}, \mathscr{J}_{\text{unif}}\right)\right) = O\left(\frac{1}{q^\ell}\right)$$

where Δ denotes the statistical distance.

PROOF. Let us index from 1 to $\binom{n}{k+\ell}$ the subset of size $k+\ell$ of $[\![1,n]\!]$ and let X_i be the indicator of the event "the subset $\mathscr{J}_i$ of index i is such that $\mathbf{H}_{\overline{\mathscr{J}_i}}$ has not a full rank". Let,

$$N \stackrel{\text{def}}{=} \sum_i X_i$$

It can be verified that we have

$$\mathbb{E}_{\mathbf{H}}\left(\Delta\left(\mathscr{J}, \mathscr{J}_{\text{unif}}\right)\right) = \mathbb{E}_{\mathbf{H}}\left(\frac{N}{\binom{n}{k+\ell}}\right) = \frac{1}{\binom{n}{k+\ell}} \sum_{i=1}^{\binom{n}{k+\ell}} \mathbb{E}_{\mathbf{H}}(N_i) \tag{3.23}$$

where the last equality follows from the linearity of the expectation.
Notice now that $\mathbf{H}_{\overline{\mathscr{J}}} \in \mathbb{F}_q^{(n-k)\times(n-k-\ell)}$ has not a full rank with probability (over $\mathbf{H}$) given by a $O\left(\frac{1}{q^\ell}\right)$. Therefore,

$$\mathbb{P}\left(X_i = 1\right) = O\left(\frac{1}{q^\ell}\right)$$

Plugging this in Equation (3.23) concludes the proof. □

However, although the above proposition enables to avoid an assumption, there will be as for Prange, an assumption to make when studying ISD algorithms.

An important quantity. Let us use notation of the above algorithm. Let $\alpha_{p,\ell}$ be the probability that given a fixed $\mathbf{x} \in \mathbb{F}_q^{k+\ell}$ be such that $\begin{cases} \mathbf{x}\mathbf{H}''^{\intercal} = \mathbf{s}'' \\ |\mathbf{x}| = p \end{cases}$, the vector $\mathbf{e}' \stackrel{\text{def}}{=} \mathbf{s}' - \mathbf{x}\mathbf{H}''^{\intercal}$ has Hamming weight $t-p$, namely

$$\alpha_{p,\ell} \stackrel{\text{def}}{=} \mathbb{P}\left(|\mathbf{e}'| = t-p\right).$$

In other words, $\alpha_{p,\ell}$ denotes the probability that given a solution $\mathbf{x}$ of the decoding problem at distance p with input $(\mathbf{H}'', \mathbf{s}'')$, then its lift gives a solution of weight t of the initial decoding problem with input $(\mathbf{H}, \mathbf{s})$. Notice that we did not suppose that $\mathbf{x} \in \mathscr{S}$.

PROPOSITION 3.3.2. *The probability $\alpha_{p,\ell}$ is up to a polynomial factor (in n) given by,*

$$\frac{\binom{n-k-\ell}{t-p}(q-1)^{t-p}}{\min\left(q^{n-k-\ell}, \binom{n}{t}(q-1)^t q^{-\ell}\right)}$$

The proof of this proposition is similar to the one of Proposition 3.1.1 and here we only provide a sketch of it.

SKETCH OF PROOF. One can remark that the only difference between formulas of Propositions 3.1.1 and 3.3.3 is the factor $q^{-\ell}$ in the denominator. The difference comes from the fact that the probability (over $\mathbf{H}$) that an error $(\mathbf{e}', \mathbf{e}'')$ of weight t verifies $\mathbf{e}''\mathbf{H}''^{\intercal} = \mathbf{s}''$, where $\mathbf{s}'' \in \mathbb{F}_q^\ell$, is $q^{-\ell}$. Therefore we have to consider a fraction $q^{-\ell}$ of possible solutions in our probability, which roughly explains the factor $q^{-\ell}$ in the denominator. □

We are now ready to give the running time of ISD algorithms to solve DP. It will use the following assumption

ASSUMPTION 3.3.1. *Let us use notation of* ISD *algorithm that is described above. Given* $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(S)}$ *be solution of* $\begin{cases} \mathbf{x}\mathbf{H}''^{\intercal} = \mathbf{s}'' \\ |\mathbf{x}| = p \end{cases}$ *Then, the vectors* $\mathbf{e}^{(i)}$*'s in* $\mathbb{F}_q^n$ *be defined as*

$$\mathbf{e}^{(i)}_{\mathscr{J}} = \mathbf{s}' - \mathbf{x}^{(i)}\mathbf{H}'^{\intercal} \quad ; \quad \mathbf{e}^{(i)}_{\mathscr{J}} = \mathbf{x}^{(i)}$$

are independent random variables (where $\mathscr{J}$ *is a random augmented information set).*

PROPOSITION 3.3.3. *Let* $\ell \in [\![0, n-k]\!]$*. Let* $\mathscr{A}$ *be an algorithm that can compute* S *solutions in time* T *of the following problem* $\begin{cases} \mathbf{x}\mathbf{H}''^{\intercal} = \mathbf{s}'' \\ |\mathbf{x}| = p \end{cases}$ *where* $\mathbf{H} \in \mathbb{F}_q^{\ell \times (k+\ell)}$ *and* $\mathbf{s}'' \in \mathbb{F}_q^{\ell}$*. Furthermore, we suppose that outputs of* $\mathscr{A}$ *verify Assumption 3.3.1. Then, the* ISD *algorithm using* $\mathscr{A}$ *in Step* 3 *solves* $\mathsf{DP}(n, q, R, \tau)$ *up to a polynomial factor (in* n*) in time*

$$T \ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right)$$

where $\alpha_{p,\ell}$ *is given in Proposition 3.3.2.*

As in Proposition 3.3.2 we only provide a sketch of proof of this proposition.

SKETCH OF PROOF. The number of iterations of ISD algorithms is, as for Prange's algorithm, up to a polynomial factor (in n) given by $1/p_{\text{ISD}}$ where p_{ISD} is the probability of success of an iteration. Furthermore, each iteration has a cost given by the time to computing $\mathscr{S}$ in Step 3 (the cost of a Gaussian elimination is polynomial). Therefore the cost of the ISD using $\mathscr{A}$ is given by $T\ \frac{1}{p_{\text{ISD}}}$.

Let us compute p_{ISD}. Let $\mathbf{e}''_1, \dots, \mathbf{e}''_S$ be the outputs of $\mathscr{A}$. The probability that any $\mathbf{e}''_i$ does not lead to a solution is given by $1 - \alpha_{p,\ell}$. Using the independence given by Assumption 3.3.1, the probability that none of the $\mathbf{e}''_i$'s leads to a solution is given by

$$1 - (1 - \alpha_{p,\ell})^S = 1 - \Theta\left(\min(1, S\alpha_{p,\ell})\right).$$

Therefore, $p_{\text{ISD}} = \Theta\left(\min(1, S\alpha_{p,\ell})\right)$ and

$$T\ \frac{1}{p_{\text{ISD}}} = T\ \frac{1}{\Theta\left(\min(1, S\alpha_{p,\ell})\right)} = \Theta\left(T\ \max\left(1, \frac{1}{S\ \alpha_{p,\ell}}\right)\right)$$

which concludes the proof. □

We are now ready to "instantiate" ISD algorithms with Dumer and Wagner algorithms that we have described in Subsections 3.2.1 and 3.2.2.

3.3.1. ISD with Dumer's algorithm. A slight variation of Proposition 3.2.1 shows that, given an instance $(\mathbf{H}'', \mathbf{s}'') \in \mathbb{F}_q^{\ell \times (k+\ell)} \times \mathbb{F}_q^{\ell}$ of a decoding problem at distance p, Dumer's algorithm find

$$\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}$$

solutions in average time

$$\sqrt{\binom{k+\ell}{p}(q-1)^p} + \frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}.$$

Here there is no maximum in the fomula as we are not sure that there is always a solution to our decoding problem.

Therefore we easily deduce the following proposition which gives the complexity of the ISD using Dumer's algorithm.

PROPOSITION 3.3.4. *The complexity* $C_{\text{Dumer}}(n,q,R,\tau)$ *of the* ISD *using Dumer's algorithm (described in Subsection 3.2.1) to solve* $\mathsf{DP}(n,q,R,\tau)$ *is up to a polynomial factor (in* n*) given by*

$$\left(\sqrt{\binom{k+\ell}{p}(q-1)^p} + \frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell}\right) \cdot \max\left(1, \frac{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}{\binom{n-k-\ell}{t-p}(q-1)^{t-p}\binom{k+\ell}{p}(q-1)^p}\right) \tag{3.24}$$

This complexity is parametrized by p and ℓ. According to our wish, finding a short or large solution, the optimization will not be the same. Let us describe our strategy for both of them but before let us fix the relative quantities that we will consider

$$R \stackrel{\text{def}}{=} \frac{k}{n}, \quad \tau \stackrel{\text{def}}{=} \frac{w}{n}, \quad \lambda \stackrel{\text{def}}{=} \frac{\ell}{n} \quad \text{and} \quad \pi \stackrel{\text{def}}{=} \frac{p}{n}.$$

These quantities will be useful as we are interested in the *asymptotic complexity* of the ISD's.

Strategy to reach short solutions. Our first choice is to force Dumer's algorithm to produce decoding solutions in amortized time one. Let us stress that here we give a method to optimize the complexity of ISD's, but we do not claim that it will lead to optimal parameters. Anyway, Dumer's algorithm computes solutions in amortized time one if

$$\sqrt{\binom{k+\ell}{p}(q-1)^p} = \frac{\binom{k+\ell}{p}}{q^\ell} \iff q^\ell = \sqrt{\binom{k+\ell}{p}(q-1)^p}$$

Using Equation (3.12), it implies asymptotically the following equality

$$\lambda = \frac{R+\lambda}{2} h_q\left(\frac{\pi}{R+\lambda}\right) \iff \pi = (R+\lambda) h_q^{-1}\left(\frac{2\lambda}{R+\lambda}\right) \tag{3.25}$$

Let $\pi(\lambda)$ be the parameter π that reaches the above equality. We can now verify that according to Equations (3.12), (3.24) and (3.25) that

$$\frac{1}{n}\log_q(C_{\text{Dumer}}) = f(\lambda)(1+o(1))$$

where

$$f(\lambda) \stackrel{\text{def}}{=} \lambda + \max\left(0, \min(1-R, h_q(\tau)) - (1-R-\lambda) h_q\left(\frac{\tau-\pi(\lambda)}{1-R-\lambda}\right) - 2\lambda\right).$$

To optimize $\lambda \mapsto f(\lambda)$, a good approximation (which can be verified for many parameters) is to suppose that it is an unimodal function. Then its minimization is easy to obtain with for instance the golden section search (see https://en.wikipedia.org/wiki/Golden-section_search). We used this method to draw the exponent (for relative weights $\tau \leq (q-1)/q(1-R)$) of the ISD with Dumer's algorithm given in Figures 3.4, 3.7 and 3.8 . Furthermore we multiplied the above formula by a term $\log_2(q)$ to get exponents in base 2.

Strategy to reach large solutions. Let us suppose that $q > 2$. Otherwise we can symmetrize the complexity of the algorithm from the short case as shown in Exercise 3.1.2. Contrary to the strategy to get short solutions, if one wants to use an ISD to compute solutions with a large weight, one has to choose p as $k+\ell$ (see the discussion in the beginning of this section entitled "Far or close codeword"). Therefore, with Dumer's algorithm we will choose parameters such that

$$\lambda = \frac{R+\lambda}{2} h_q\left(\frac{\pi}{R+\lambda}\right) \quad \text{and} \quad \pi = R+\lambda$$

which leads to (as $h_q(1) = \log_q(q-1)$),

$$\lambda = \frac{R+\lambda}{2}\log_q(q-1) \iff \lambda = \frac{R}{2}\,\frac{\log_q(q-1)}{1-\frac{1}{2}\log_q(q-1)}$$

However if one uses this strategy directly with Dumer's algorithm it would lead to very high exponent as build lists of size $q^{\lambda n}$ would be too large. The idea (before using Wagner's algorithm as we are going to do) is to change Dumer's algorithm and to use the variation given in Exercise 3.2.1. Suppose that one build lists of size S in Dumer's algorithm. Then, according to Proposition 3.3.3, the complexity of the ISD with this algorithm is given (up to polynomial factor by) (we fixed p to $k+\ell$)

$$\left(S + \frac{S^2}{q^\ell}\right) \cdot \max\left(1, \frac{\min\left(q^{n-k}, \binom{n}{t}(q-1)^t\right)}{\binom{n-k-\ell}{t-k-\ell}(q-1)^{t-k-\ell}\, S^2}\right)$$

Let $\sigma \stackrel{\text{def}}{=} \frac{1}{n}\log_q S$. Using this algorithm leads to the following asymptotic complexity

$$g(\lambda,\sigma) \stackrel{\text{def}}{=} \max(\sigma, 2\sigma-\lambda) + \max\left(0, \min(1-R, h_q(\tau)) - (1-R-\lambda)h_q\left(\frac{\tau-R-\lambda}{1-R-\lambda}\right) - 2\sigma\right) \tag{3.26}$$

However we do not have to forget that we have a constraint on the size of built lists, namely $S \leq \binom{(k+\ell)/2}{p/2}(q-1)^{p/2}$, therefore σ has necessarily to verify

$$\sigma \leq \frac{R+\lambda}{2}\, h_q\left(\frac{\pi}{R+\lambda}\right). \tag{3.27}$$

To optimize (3.26) we used the golden section search to first finding $\sigma(\lambda)$ "minimizing" (according to the method) $\sigma \mapsto g(\lambda,\sigma)$ for a fixed λ and σ verifying Constraint (3.27). Then we also used the golden section search to "minimize" $\lambda \mapsto g(\lambda,\sigma(\lambda))$. We draw in Figures 3.7 and 3.8 the exponent of Prange and the ISD with Dumer's algorithm for a fixed rate and as function of τ. As we see Dumer's algorithm provides an improvement over Prange's algorithm. Even if the improvement seems slight, don't forget that it means an *exponential* improvement as we draw exponents.

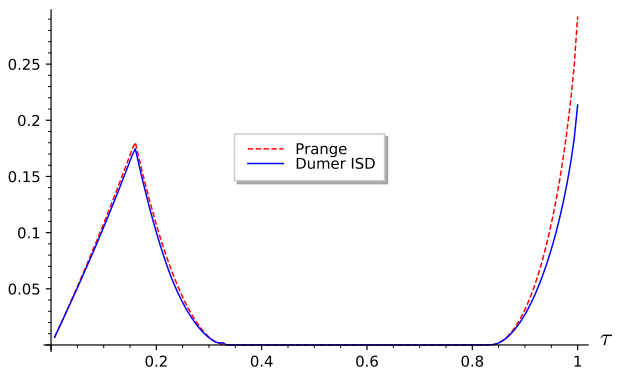


FIGURE 3.7. Exponents in base 2 of Prange's algorithm and ISD with Dumer's algorithm (in base 2) to solve $\mathsf{DP}(n,q,R,\tau)$ for $q=3$ and $R=1/2$ as function of $\tau \in [0,1]$.

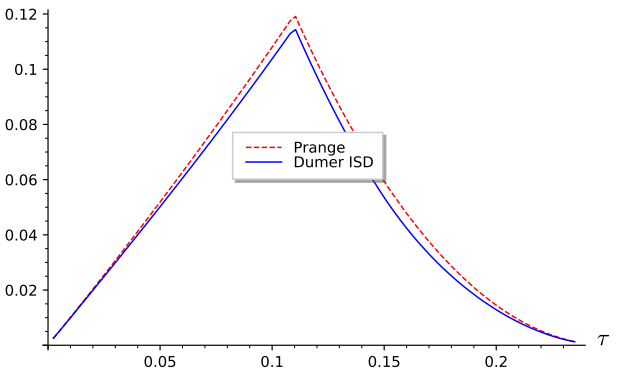


FIGURE 3.8. Exponents in base 2 of Prange's algorithm and ISD with Dumer's algorithm (in base 2) to solve $\mathsf{DP}(n,q,R,\tau)$ for $q=2$ and $R=1/2$ as function of $\tau \in \left[0, \frac{q-1}{q}(1-R)\right]$.

3.3.2. ISD with Wagner's algorithm. We are now ready to instantiate an ISD with Wagner's algorithm as a subroutine. In this case we choose to parametrize the algorithm to output solutions in amortized time one. Combining Propositions 3.3.3 and 3.2.2 (assertion (2)) leads to the following proposition

PROPOSITION 3.3.5. *The complexity* $C_{\text{Dumer}}(n, q, R, \tau)$ *of the* ISD *using Wagner's algorithm (described in Subsection 3.2.2) to solve* $\mathsf{DP}(n, q, R, \tau)$ *is up to a polynomial factor (in* n*) given by*

$$q^{\frac{\ell}{a}} \max\left(1, \frac{\min\left(q^{n-k-\ell}, \binom{n}{t}(q-1)^t q^{-\ell}\right)}{\binom{n-k-\ell}{t-p}(q-1)^{t-p}\, q^{\frac{\ell}{a}}}\right) \tag{3.28}$$

where a *is the largest integer such that* $q^{\frac{\ell}{a}} \leq \binom{(k+\ell)/2^a}{p/2^a}(q-1)^{p/2^a}$.

We used this proposition (with the same kind of strategy that above) to draw the exponent of the ISD with Wagner's algorithm. As we can see in Figure 3.4 the ISD with Wagner's algorithm has far better exponent compared to the ISD with Dumer's algorithm for large weight; otherwise exponents are the same.

Bibliography

[AAB+17] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC, November 2017. NIST Round 1 submission for Post-Quantum Cryptography.

[Ale03] Alekhnovich, Michael. More on Average Case vs Approximation Complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.

[Arı09] Erdal Arıkan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inform. Theory*, 55(7):3051–3073, 2009.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1+1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, LNCS. Springer, 2012.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[BM17] Leif Both and Alexander May. Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security. In *WCC Workshop on Coding and Cryptography*, September 2017.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.

[CS16] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography 2016*, LNCS, pages 144–161, Fukuoka, Japan, February 2016.

[Deb19] Thomas Debris-Alazard. *Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse. (Code-based Cryptography: New Approaches for Design and Proof ; Contribution to Cryptanalysis)*. PhD thesis, Pierre and Marie Curie University, Paris, France, 2019.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *Advances in Cryptology - ASIACRYPT 2019*, LNCS, Kobe, Japan, December 2019. Springer.

[Dum86] Ilya Dumer. On syndrome decoding of linear codes. In *Proceedings of the 9th All-Union Symp. on Redundancy in Information Systems, abstracts of papers (in russian), Part 2*, pages 157–159, Leningrad, 1986.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. 1984.

[Eli55] Peter Elias. oding for noisy channels. *IRE conv. Rec.*, 3:37, 1955.

[Fin09] Matthieu Finiasz. NP-completeness of certain sub-classes of the syndrome decoding problem, 2009. arXiv:0912.0453.

[FS96] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *LNCS*, pages 245–255. Springer, 1996.

[FS09] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gop81] Valerii D. Goppa. Codes on algebraic curves. *Dokl. Akad. Nauk SSSR*, 259(6):1289–1290, 1981. In Russian.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.

[GV05] Venkatesan Guruswami and Alexander Vardy. Maximum-likelihood decoding of reed-solomon codes is np-hard. *IEEE Trans. Inf. Theory*, 51(7):2249–2256, 2005.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[MS86] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. North–Holland, Amsterdam, fifth edition, 1986.

[MS09] L. Minder and A. Sinclair. The extended k-tree algorithm. In C. Mathieu, editor, *Proceedings of SODA 2009*, pages 586–595. SIAM, 2009.

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

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

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, 1993.

[Var97] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6):1757–1766, November 1997.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.