Post-Quantum Cryptography - Codes

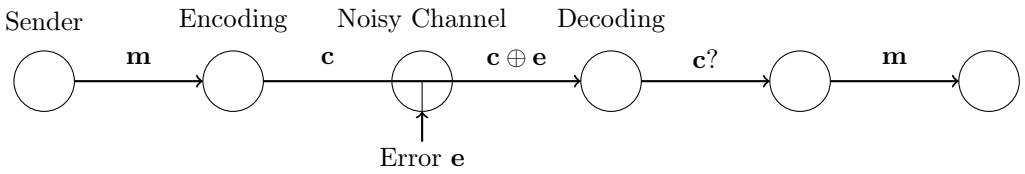
Lecture 1: An Intractable Problem Related to Codes, Decoding

Lecturer: Thomas Debris-Alazard

Introduction

In this course we will consider mathematical objects known as *linear codes*, but what is 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 $d_{\mathrm{H}}(\mathbf{c}, \mathbf{y}) \stackrel{\text{def}}{=} \sharp\left\{i \in [\![1, n]\!] : c_i \neq y_i\right\}$ 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[1] 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 this lecture note.

[1] 25 October 2021

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. 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 (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. *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. 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 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 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 $\mathbf{G}\mathbf{H}^\intercal = \mathbf{0}$. 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 $\mathbf{G}\mathbf{H}^\intercal = \mathbf{0}$, 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 $\mathbf{G}\mathbf{H}^\intercal = \mathbf{0}$. 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 $\mathbf{H}\mathbf{y}^\intercal = \mathbf{H}\mathbf{c}^\intercal + \mathbf{H}\mathbf{e}_i^\intercal = \mathbf{H}\mathbf{e}_i^\intercal$ 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 $\mathbf{H}\mathbf{y}^\intercal$.

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 $\mathbf{H}\mathbf{y}^\intercal$ could be a nice source of information when trying to decode $\mathbf{y}$. This is even more remarkable that we give a name to $\mathbf{H}\mathbf{y}^\intercal$.

Definition 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 $\mathbf{H}\mathbf{y}^\intercal$. Any element of $\mathbb{F}_q^{n-k}$ is called a syndrome.*

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

Definition 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. *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).

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 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 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 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 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 d–tuple 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 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 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 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 exist 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 later, 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 $C > 0$ 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.

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 (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 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 3. *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 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 and (ii) we would like to solve Problem 2. To this aim, let $(\mathbf{H}, \mathbf{s})$ be an input of Problem 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 6. *Show that any solver of Problem 2 can be turned in polynomial time into an algorithm solving Problem 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 this 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 Reed–Solomon 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, DVD's, BluRay's, QR codes etc... They also play an important role in code-based cryptography as we will see in lecture note 4.

Definition 7 (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 Reed–Solomon 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 efficient decoding algorithms 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 7. *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:*

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

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.

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

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 (1.3) 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 3. *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 $R \stackrel{\text{def}}{=} N_1E_2 - N_2E_1$, we have

$$\deg(R) < k + \left\lfloor \frac{n-k}{2} \right\rfloor + \left\lceil \frac{n-k}{2} \right\rceil - 1 < n.$$

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

$$\begin{aligned} R(x_i) &= N_1(x_i)E_2(x_i) - N_2(x_i)E_1(x_i) \\ &= y_iE_1(x_i)E_2(x_i) - y_iE_2(x_i)E_1(x_i) \\ &= 0 \end{aligned}$$

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.

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 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* $\mathbf{He}^\intercal = \mathbf{s}^\intercal$.

Proposition 1 ([BMvT78])**.** *Problem 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 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. *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 4. *Let* T *and* $U \subseteq T \times T \times T$ *be an instance of* 3DM *and let* $\mathbf{H}_{\text{3DM}} \in \mathbb{F}_2^{3|T| \times |U|}$ *be the associated incidence matrix. We have*

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

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.

Proof of Proposition 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 4, 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.

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] (for more details see lecture note 4) that was introduced just few months after RSA. This scheme will motivate our definition of the "cryptographic" decoding in Problem 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 4. *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 8. *Describe how the encryption scheme works with generator matrices.*

We are now ready to define the 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 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 : $(\mathbf{H}, \mathbf{s} \stackrel{def}{=} \mathbf{xH}^\intercal)$ *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* $\mathbf{eH}^\intercal = \mathbf{s}$.

Remark 5. *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 lecture note 2).*

Remark 6. *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 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 lectures 2 and 4. 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 lecture note 3 we will study algorithms solving this problem and in each case their complexity will be written as some T/ε.

Remark 7. *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:*

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

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 lecture note 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 1.1.

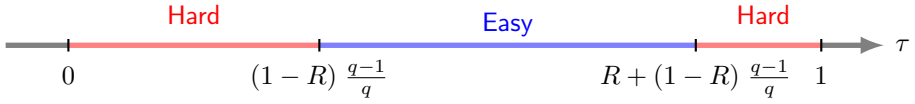


FIGURE 1.1. Hardness of $\mathsf{DP}(n,q,R,\tau)$ as function of τ.

$\mathsf{DP}(n,q,R,\tau)$ 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).

Problem 6 (Learning with Parity Noise Problem - $\mathsf{LPN}(k,\tau)$)**.**

- Input : $\mathscr{O}_{\mathbf{s},\tau}(\cdot)$ *parametrized by* $\mathbf{s} \in \mathbb{F}_2^k$ *which has been chosen uniformly at random and* τ *such that on a call it outputs* $(\mathbf{a}, \mathbf{s}\cdot\mathbf{a}+e)$ *where* $\mathbf{a} \in \mathbb{F}_2^k$ *is uniformly distributed and* e *being distributed according to a Bernoulli of parameter* τ*.*
- Output : $\mathbf{s}$

Let us stress that anyone who wants to solve this problem can ask as many samples (outputs of $\mathscr{O}_{\mathbf{s},\tau}(\cdot)$) 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 $\mathscr{O}_{\mathbf{s},\tau}(\cdot)$ 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 8. DP*could have been presented directly with generator matrix representation. For more details see lecture note* 2*, in particular Exercise* 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 (fore more details see lecture note 4).

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 $\mathscr{C}$ of length n for which we do not necessarily have an efficient decoding algorithm. The public key in Alekhnovich scheme is defined as $(\mathscr{C},\mathbf{c}+\mathbf{e})$ where $\mathbf{c}\in\mathscr{C}$ 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 $\mathscr{C}$ 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 $\mathscr{C}$ 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 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 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 9. *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 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 8. *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 according to $\mathscr{D}_b$ which is defined in $\mathsf{DDP}(n,q,R,\tau)$ (Problem 7). We define the $\mathsf{DDP}(n,q,R,\tau)$-computational success in time T as:

$$Succ^{\mathsf{DDP}(n,q,R,\tau)}(t) \stackrel{def}{=} \max_{\mathscr{A}:|\mathscr{A}|\leq T} \left(Adv^{\mathsf{DDP}(n,q,R,\tau)}(\mathscr{A}) \right).$$

where $|\mathscr{A}|$ denotes the running time of $\mathscr{A}$.

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

Exercise 9. *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 $\mathbf{e}\cdot\mathbf{r}$ 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 2.

Theorem 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 10. *Theorem 1 is stated for binary codes. However it can be extended to q-ary codes by using a generalization of Proposition 2 proved in* [GRS00].

Proof of Theorem 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 2.

Algorithm $\mathscr{A}'$:

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

$\mathbf{u} \in \mathbb{F}_2^{n-k}$ be uniformly distributed,

$\mathbf{H}' \leftarrow \mathbf{H} - \mathbf{u}^\intercal\mathbf{r}$

$b \leftarrow \mathscr{A}(\mathbf{H}',\mathbf{s})$

Output : b

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:

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

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

□

References

[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] Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, 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.

[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.

[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.

[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.

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

[Eli55] Peter Elias. Coding 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.

[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. 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.

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

[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.

[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.