# Permutations and Derangements

Tomasz Brengos
Committers : Aliaksei Kudzelka

## 1 Encoding and Decoding Functions

Let $p$ be a prime number. We work over the finite field $\mathbb{Z}_p$ (the integers mod $p$). For any positive integer $n$, the set $\mathbb{Z}_p^n$ denotes the set of all $n$-tuples of elements from $\mathbb{Z}_p$. Note that since $\mathbb{Z}_p$ is a field, the spaces $\mathbb{Z}_p^n$ and $\mathbb{Z}_p^k$ can be viewed as vector spaces (of dimension $n$ or $k$ respectively) over $\mathbb{Z}_p$.

**Encoding function:** A map $\xi : \mathbb{Z}_p^k \to \mathbb{Z}_p^n$ is called an *encoding function* (or *coding function*) if it is one-to-one. In other words, $\xi$ takes each possible message (an element of $\mathbb{Z}_p^k$) and encodes it as a longer string (an element of $\mathbb{Z}_p^n$). The image of $\xi$, denoted

$$\mathcal{C} = \xi(\mathbb{Z}_p^k) \subseteq \mathbb{Z}_p^n,$$

is the set of all possible encoded messages and is called the *code*. The elements of $\mathcal{C}$ are called *codewords*.

**Decoding function:** Given an encoding $\xi$, a map $\eta : \mathbb{Z}_p^n \to \mathbb{Z}_p^k$ is called a *decoding function* for $\xi$ if $\eta(\xi(u)) = u$ for every $u \in \mathbb{Z}_p^k$. In other words, $\eta$ is a (left) inverse of $\xi$ on the set of codewords, so that decoding an encoded message returns the original message (assuming no errors in transmission).

Once a message $u \in \mathbb{Z}_p^k$ is encoded as a codeword $c = \xi(u) \in \mathbb{Z}_p^n$, it is sent through a channel (transmission). During transmission, some components of $c$ might be altered due to noise or other errors. If the received word is $c' \in \mathbb{Z}_p^n$ (which may or may not equal $c$), we can describe the effect of the channel by an *error vector* $e = c' - c \in \mathbb{Z}_p^n$. Here subtraction is component-wise in $\mathbb{Z}_p$. The vector $e$ represents which positions, if any, have been corrupted (non-zero entries in $e$ indicate an error in that coordinate). After reception, the decoding function $\eta$ is applied to $c'$ in an attempt to recover the original message. If $c'$ equals the original codeword $c$ (i.e. no transmission error, so $e = \mathbf{0}$), then $\eta(c') = \eta(c) = u$. If $c'$ differs from any valid codeword (i.e. $c'$ is not in $\mathcal{C}$), the decoder can detect that an error has occurred (and possibly request retransmission, depending on the system). In general, a well-designed code will add redundancy (extra symbols in $c$ compared to $u$) so that even if some errors occur, they can be detected or corrected.

## 2 Examples of Encoding Functions (Codes)

We now present a few example encoding functions and their corresponding codes:

### Example 1: Repetition Code

One simple encoding is the *repetition code*. Here the message space is $\mathbb{Z}_p$ (messages of length 1). The encoding function $\xi_1 : \mathbb{Z}_p \to \mathbb{Z}_p^n$ for some chosen length $n$ is defined by

$$\xi_1(a) = (\underbrace{a, a, \ldots, a}_{n \text{ times}}),$$

i.e. the single-symbol message $a$ is repeated $n$ times to form the codeword. The code $\mathcal{C}_1 = \{(a, a, \ldots, a) : a \in \mathbb{Z}_p\}$ consists of all $n$-tuples with identical entries. This encoding is one-to-one (different $a$ give different constant tuples).

For example, over $\mathbb{Z}_2$ (the binary case), if $a = 1$ and $n = 5$, the codeword would be $(1, 1, 1, 1, 1)$. If a transmission error flips one of these bits (say we receive $(1, 1, 1, 0, 1)$), the decoder can notice that this is not a valid codeword in $\mathcal{C}_1$ (since not all bits are the same), hence an error is detected. In fact, the repetition code is able to detect up to $n - 1$ errors (any change in at most $n - 1$ positions will yield a tuple that is not constant and thus not in the code). This code even has error-correcting capability: intuitively, the decoder could guess that the majority value is the intended bit (for example, from $(1, 1, 1, 0, 1)$ a reasonable decoded value for $a$ would be 1, since most bits are 1). However, our focus here is on error detection criteria, which we formalize later.

## Example 2: Parity Check Code

Another useful encoding adds a *parity check* to the message. For this example, let's work in $\mathbb{Z}_p$ (and especially consider $p = 2$ for binary parity). Let the message space be $\mathbb{Z}_p^{k-1}$ (messages of length $k - 1$). Define an encoding function $\xi_2 : \mathbb{Z}_p^{k-1} \to \mathbb{Z}_p^k$ by

$$\xi_2(a_1, a_2, \dots, a_{k-1}) = \big(a_1, a_2, \dots, a_{k-1},\ a_k\big),$$

where the last component $a_k$ is chosen such that the total sum of the $k$ components is 0 in $\mathbb{Z}_p$. In other words,

$$a_k \ = \ -(a_1 + a_2 + \dots + a_{k-1}) \pmod p,$$

which ensures $a_1 + a_2 + \dots + a_{k-1} + a_k \equiv 0 \pmod p$. This extra symbol $a_k$ is a redundancy (the *parity symbol*) chosen to enforce a constraint on every codeword.

The resulting code is

$$\mathcal{C}_2 = \{(a_1, \dots, a_{k-1}, a_k) \in \mathbb{Z}_p^k : a_1 + \dots + a_{k-1} + a_k = 0\}.$$

In the special case $p = 2$ (binary), this encoding is the standard single-parity-bit code: the last bit $a_k$ is 0 or 1 chosen to make the total number of 1's in the codeword even. For example, if the message is $(1, 0, 1, 1)$ in $\mathbb{Z}_2^4$ (so $k - 1 = 4$), then the parity bit $a_5$ is chosen so that $1 + 0 + 1 + 1 + a_5 \equiv 0 \pmod 2$. Here $1 + 0 + 1 + 1 = 3 \equiv 1 \pmod 2$, so we must take $a_5 = 1$ to make the sum even. The encoded 5-bit codeword is $(1, 0, 1, 1, 1)$, which has an even number of 1's (four 1's in this case).

This parity check code allows detection of any single-error in transmission: if one bit of a codeword is flipped, the parity (sum mod $p$) will no longer be zero, so the received word will violate the code constraint and thus be recognized as invalid. For instance, if $(1, 0, 1, 1, 1)$ is sent and one bit flips, the sum of bits will be odd, alerting us to the error. In fact, more generally, if an odd number of bits are in error, the parity check will detect it (sum becomes nonzero mod 2). However, if an even number of bits are corrupted in the binary case, the parity check could fail to detect it (since an even number of flips can restore the sum to even). We will quantify the error-detecting capability of codes like this in Section 4.

## Example 3: PESEL Check Digit

The Polish national identification number, PESEL, consists of 11 decimal digits $p_1 p_2 \dots p_{11}$. Its structure is

$$p_1 p_2 p_3 p_4 p_5 p_6\, p_7 p_8 p_9 p_{10} p_{11}$$

where

- $p_1 p_2$ encode the last two digits of the birth year,
- $p_3 p_4$ encode the month (with a century–dependent offset),
- $p_5 p_6$ encode the day of the month, and
- $p_{11}$ is a *check digit* chosen for error detection.

Formally we regard the first ten digits $(p_1, \dots, p_{10}) \in \mathbb{Z}_{10}^{10}$ as the message and define an encoding

$$\xi_3 : \mathbb{Z}_{10}^{10} \longrightarrow \mathbb{Z}_{10}^{11}, \qquad \xi_3(p_1, \dots, p_{10}) = (p_1, \dots, p_{10}, p_{11}),$$

where $p_{11}$ is computed via the weighted checksum rule

$$p_{11} \equiv -\big(1p_1 + 3p_2 + 7p_3 + 9p_4 + 1p_5 + 3p_6 + 7p_7 + 9p_8 + 1p_9 + 3p_{10}\big) \pmod{10}.$$

Equivalently, the weighted sum

$$S = 1p_1 + 3p_2 + 7p_3 + 9p_4 + 1p_5 + 3p_6 + 7p_7 + 9p_8 + 1p_9 + 3p_{10} + 1p_{11}$$

must be a multiple of 10.

For concreteness, take the birth date 15 April 1997, which encodes as 150497 for the first six digits (day 15, month 04, year 97). Suppose the remaining four data digits are 1478, giving the 10–tuple $(1, 5, 0, 4, 9, 7, 1, 4, 7, 8)$. The checksum is

$$\begin{aligned} S &= 1 \cdot 1 + 3 \cdot 5 + 7 \cdot 0 + 9 \cdot 4 + 1 \cdot 9 + 3 \cdot 7 + 7 \cdot 1 + 9 \cdot 4 + 1 \cdot 7 + 3 \cdot 8 \\ &= 1 + 15 + 0 + 36 + 9 + 21 + 7 + 36 + 7 + 24 = 156 \equiv 6 \pmod{10}. \end{aligned}$$

Hence $p_{11} \equiv -6 \equiv 4 \pmod{10}$, so the full PESEL number is `15049714784`. If any single digit is altered (or two adjacent digits are swapped), the weighted sum is almost certain to be non–zero mod 10, flagging an error. The PESEL checksum therefore detects *all* single–digit errors and the vast majority of simple transpositions.