Security for connected devices: introduction to error correction

Jean-François COUCHOT
couchot[at]femto-st[dot]fr

September 30, 2019

Contents

Chapter 1

Practical introduction to correction codes

In 1948, Claude Shannon published a reference document "A mathematical theory of communication" [Sha01] the basis of information theory and coding theory. For each communication channel, Shannon has identified a number called *c*apacity which is the maximum reliability that can be achieved by any communication on that channel. To achieve this, it is necessary to set up data encoding/decoding processes which, in the end, will make it possible to correct certain errors.

A communication channel is shown in the figure 1.1. At the source, a x message must be sent. If x is transmitted as such in this channel, any noise would alter it and it would not be recoverable. The basic idea is to embed the message with some redundancy so that, at the reception, you can find x. Redundancy is added by encoding and the encoded message, called a c codeword, is sent on the channel. The noise, expressed as an error vector e, is added to the code producing a received vector y. This received vector is then decoded and an estimate of $\hat{x}$ of the x message is generated.

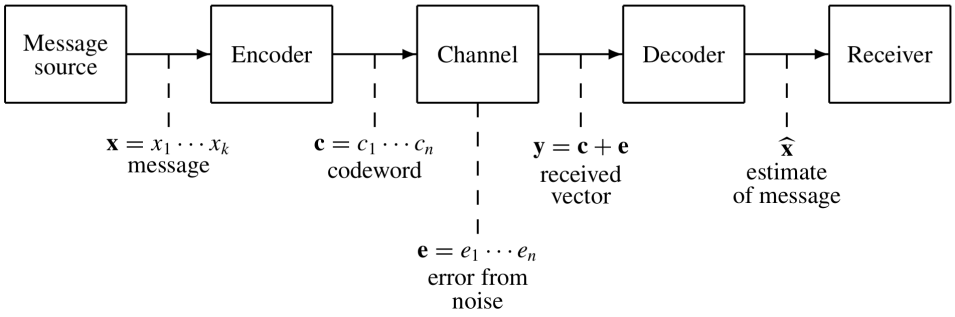

Figure 1.1: Communication Channel

1.1 Redundancy first!

To encode these messages we think of redundancy. For example, who has never heard a transmission of the type "E for Echo, R for Romeo, R for Romeo, O for Obvious, and R for Romeo" to transmit the word "error". To improve the transmission of a message, this code multiplies the information, adds redundancy which is the key to any detector and corrector code.

In the following, we consider that the encoded characters are the bits. For encoding blocks from k bits to n bits, there are 2^k different words to encode. Among the 2^n possible received messages of length n, 2^k are code words (*i.e.*, without error) and therefore $2^n - 2^k$ are not correct. The performance of a code is the rate of $\rho = \frac{k}{n}$.

1.2 Parity bit

A second example is an error detection code. The 7-bit encoding of the usual characters is considered as shown in the table 1.1.

To detect an error, the encoding adds an eighth bit to the seven-bit vector, called the parity bit which is 1 if and only if the sum of the bits initially present is odd. For example, the sum of the seven bits of B being 1 + 0 + 0 + 0 + 0 + 0 + 1 = 2, the parity bit is then 0. B is then coded in 10000010.

Char	Digit	Binary	Parity + bit
B	65	1000001	10000010
C	66	1000010	10000100
D	67	1000011	10000111
⋮			
a	97	1100001	11000011
b	98	1100010	11000101
c	99	1100011	11000110

Table 1.1: Parity bit Checksum

.

This encoding detects that an error has been made during transmission, but it does not correct it, as the receiver does not know which of the eight bits is the wrong bit. The receiver, noting the error, may, however, request a retransmission of the problematic character. Be careful, this detector code is based on the assumption that at most one bit is wrong (is it a reasonable assumption?).

Exercise 1.1. *Parity check*

- *How many errors can be detected with a simple parity check? Is it possible to correct these errors?*
- *Encode the following messages using a bit of parity:* 1101011001*,* 100*,* 1111100011111001111.
- *What is the performance of such a code?*

1.3 Data duplication

We now present an error detector and corrector code. It consists in repeating the entire message several times. Each character of a text is repeated three times. The word "011010" would thus be encoded as

$$000.111.111.000.111.000$$

If at most one error appears per character (which is reasonable), then it can be corrected:

$$100.111.110.001.110.000$$

would indeed be decoded as "011010". It is obvious that a code that would only double each character would not allow the correction in case of error (to be convinced). However, this code is not used because of its cost: all information must be transmitted in three parts.

Exercise 1.2. *Repetition code. We use a repetition code. The bits are sent 5 times with. Each time, the probability of being mistransmitted is equal to* p*.*

- *In such a 5-bit packet (i.e. 5 repetitions of the signal bit)*
  - *What is the probability that* $i = 0, 1, \ldots, 5$ *of these 5 bits are reversed during transmission?*
  - *What is the probability that the transmission error will be detected?*
  - *What is the probability that the error will be transmitted undetected?*
- *Encode the following message:* 01110*.*
- *Decode the following message:* 00100.11111.00101.01011.00100*.*
- *What is the performance of such a code?*

$w_5 = W_5$	$w_6 = W_6$	$w_7 = W_7$	wrong bit location
T	T	T	there's no wrong bit.
T	T	F	w_7 is wrong
T	F	T	w_6 is wrong
T	F	F	w_3 is wrong
F	T	T	w_5 is wrong
F	T	F	w_2 is wrong
F	F	T	w_1 is wrong
F	F	F	w_4 is wrong

Table 1.2: Correction of the wrong bit by the Hamming code$(7, 4)$.

1.4 Hamming's code$(7, 4)$

In the following, the + sum and . product operators on the field $\mathbb{F}_2 = \{0.1\}$ are considered as an "exclusive or" and "and" respectively. Their truth tables are recalled below.

$$\begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad \begin{array}{c|cc} . & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \tag{1.1}$$

In its version $C(7, 4)$, Hamming's code is limited to encoding four-letter words $(u_1, u_2, u_3, u_4) \in \mathbb{F}_2^4$ to guarantee their correct transmission. It is $(v_1, v_2, \ldots, v_7) \in \mathbb{F}_2^7$ defined by

$$(v_1, v_2, \ldots, v_7) = (u_1, u_2, u_3, u_4, u_1 + u_2 + u_4, u_1 + u_3 + u_4, u_2 + u_3 + u_4)$$

that will be sent. The redundant characters v_5, v_6 and v_7 ensure the correct transmission of any 4-bit (u_1, u_2, u_3, u_4) word when there is only one wrong bit in $(v_1, v_2, \ldots, v_7)$.

Upon receipt of $(w_1, w_2, \ldots, w_7)$, simply compare $W_5 = w_1 + w_2 + w_4$, $W_6 = w_1 + w_3 + w_4$ and $W_7 = w_2 + w_3 + w_4$ with w_5, w_6 and w_7 respectively and read the table 1.2.

To transmit the word $(1, 0, 1, 1)$, for example, it is the message $(v_1, v_2, v_3, v_4, v_5, v_6, v_7) = (1, 0, 1, 1, 0, 1, 0)$ that is sent because:

$$\begin{aligned} v_5 &= u_1 + u_2 + u_4 = 1 + 0 + 1 = 0, \\ v_6 &= u_1 + u_3 + u_4 = 1 + 1 + 1 = 1, \\ v_7 &= u_2 + u_3 + u_4 = 0 + 1 + 1 = 0. \end{aligned}$$

Suppose the receiver retrieves $(w_1, w_2, \ldots, w_7) = (1, 1, 1, 1, 0, 1, 0)$. He/she calculates

$$\begin{aligned} W_5 &= w_1 + w_2 + w_4 = 1 + 1 + 1 = 1 \neq w_5, \\ W_6 &= w_1 + w_3 + w_4 = 1 + 1 + 1 = 1 = w_6, \\ W_7 &= w_2 + w_3 + w_4 = 1 + 1 + 1 = 1 \neq w_7. \end{aligned}$$

Assuming there is only one error, and reading the correction table, he/she finds that w_2 is wrong and corrects the error.

Exercise 1.3. *Direct application of the Hamming code*(7.4)

1. *Code the following message: 0101.1001.0111*
2. *Decode the following message: 0100011.1001001.0101101.1010010.*

Chapter 2

The systematic codes of Hamming$(2^k - 1, 2^k - k - 1)$

The two numbers $2^k - 1$ and $2^k - k - 1$ indicate respectively the length of the words in the code and the size of the subspace formed by the words transmitted. For $k = 3$, we find the Hamming$code(7, 4)$.

2.1 Generator and control matrices

Two matrices are important in defining the Hamming$(2^k - 1, 2^k - k - 1)$ code:

- the **control matrix** H_k has $2^k - 1$ rows and k columns. In the case of a systematic code, this H_k matrix is defined by

$$H_k = \left(\frac{P_k}{I_k} \right), \tag{2.1}$$

where I_k is the identity matrix $k \times k$. In the upper part P_k are placed in line all non-zero $\mathbb{F}_2^k$ vectors that are not in I_k. We can therefore choose

$$H_3 = \left(\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right).$$

This H_k matrix therefore contains all non-zero vectors of $\mathbb{F}_2^k$.

- the **generator matrix** G_k has $2^k - k - 1$ rows and $2^k - 1$ columns. When H_k is defined as above, G_k is of the form

$$\left(I_{2^k - k - 1} \mid P_k \right),$$

with $I_{2^k - k - 1}$ the identity matrix of size $2^k - k - 1 \times 2^k - k - 1$. For G_3, we have for example:

$$G_3 = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right) \tag{2.2}$$

The following elements are noted:

- By construction, the product $G_k H_k$ **is the null matrix**.
- If $u = (u_1, \ldots, u_{2^k - k - 1}) \in \mathbb{F}_2^{2^k - k - 1}$ is a word to encode, $v = uG_k$ **is the associated** code word. The associated code word does not change the word u and only adds additional bits since $G_k = (I_{2^k - k - 1} \mid P_k)$. It is said that the code is *systematically*.

- For a binary word m of size $2^k - 1$, the word $\sigma(m) = mH_k$ **is called the** m syndrome. It's a word of size k.

Exercise 2.1. *Product matrix GH: null.*

1. *Show that G_3H_3 is null.*
2. *Generalize for any k, showing that G_kH_k is also null, according to the construction of H_k and G_k.*

The code of the previous example is systematic with $k = 3$. First we have $2^3 - 3 - 1 = 4$. For $u = (u_1, \ldots, u_4) = (1, 0, 1, 1)$ as in the introduction, we have $v = uG_3 = (1, 0, 1, 1, 0, 1, 0)$. The first 4 bits are copied, the last 3 are used as control. If we calculate the $\sigma(v)$ syndrome, we obtain $(0, 0, 0)$ while if we calculate the one of the $w = (1, 1, 1, 1, 0, 1, 0)$, we have $(1, 0, 1)$. The following section formalizes this.

2.2 General case: error detection and correction

PROPOSITION 2.1. Let be a code of Hamming$(2^k - 1, 2^k - k - 1)$ of control matrix H_k. A word w is a code word if and only if its syndrome $\sigma(w)$ is the null vector.

The previous proposal allows to check if there is an error.

Exercise 2.2. *Demonstration. Demonstrate the previous proposal.*

The following proposal corrects an error when it exists.

PROPOSITION 2.2. Let be a code of Hamming$(2^k - 1, 2^k - k - 1)$ of control matrix H_k. If $\sigma(m)$ is not the null vector then there is a i^{th} line in H_k that is equal to it. It is the i^{th} component w_i that is wrong.

PROOF. *Let $w = (w_1, \ldots, w_{2^k-1})$ be a word of size 2^{k-1}. Since $\sigma(w)$ is not null and since H_k contains all the $2^k - 1$ not null line of size k, there exists i such that $\sigma(w) = (H_{i,1}, \ldots, H_{i,k})$. Let us write w as*

$$w = (w_1, \ldots, w_{i-1}, w_i, \ldots, w_{2^k-1})$$

and let w' be

$$w' = (w_1, \ldots, w_{i-1}, \overline{w_i}, \ldots, w_{2^k-1})$$

i.e. w' is equal to w everywhere, but in index i. Let us prove that the syndrome of w' is null. Indeed the c^{th} component, $1 \leq c \leq k$, of $\sigma(w')$ is defined by

$$\sigma(w')_c = \sum_{l=1, l \neq i}^{2^k-1} w_l.H_{l,c} + \overline{w_i}.H_{i,c}. \tag{2.3}$$

- ***If** w_i **is 1**, $\sigma(w')_c$ is reduced to $\sum_{l=1,l\neq i}^{2^k-1} w_l.H_{l,c}$. But we know that $H_{i,c}$ is the c^{th} of $\sigma(w)$, i.e.*

$$H_{i,c} = \sum_{l=1, l \neq i}^{2^k-1} w_l.H_{l,c} + 1.H_{i,c}.$$

  We can deduce that $\sum_{l=1,l\neq i}^{2^k-1} w_l.H_{l,c}$ is null, and so $\sigma(w')_c = 0$.

- ***If** w_i **is 0**, $\overline{w_i}$ is 1. Equation (2.3) leads to*

$$\sigma(w')_c = \sum_{l=1,l\neq i}^{2^k-1} w_l.H_{l,c} + H_{i,c}.$$

  $H_{i,c}$ can be replaced by $\sigma(w)_c$ leading to

$$\sigma(w')_c = \sum_{l=1,l\neq i}^{2^k-1} w_l.H_{l,c} + \left(\sum_{l=1,l\neq i} w_l.H_{l,c}\right)$$

  which is null too.

Exercise 2.3. *Set dimensions. The following questions are about the Hamming*$code(2^k - 1, 2^k - k - 1)$.

1. *In this code, how many letters have the words* u *to transmit? How many separate words can be transmitted?*
2. *How many letters have the words encoded in* v*?*
3. *How many distinct words received* w *(wrong or not) will be decoded as the same message* u*?*
4. *Is there a* $w \in \mathbb{F}_2^{2^k-1}$ *that is not, at a possible error except, the encoding* v *of a message* $u \in \mathbb{F}_2^{2^k-k-1}$*?*

Exercise 2.4. *Let* $p = 1000^{-1}$ *be the probability that a bit is transmitted incorrectly.*

1. *What is the probability of having precisely 2 faulty bits when transmitting 7 bits, as when transmitting a word from Hamming code* $C(7, 4)$*?*
2. *What is the probability of having more than one error when transmitting 7 bits?*
3. *Rather than the Hamming code, we transmit a bit by repeating it 3 times. We decode by majority vote. Calculate the probability that the bit sent will be correctly decoded.*
4. *We transmit 4 bits by repeating each bit 3 times. What is the probability that the 4 bits will be decoded correctly? Comparing the results of this question with 2 above, we see that the simple code has a slight advantage over the Hamming code* $C(7, 4)$*, but at the cost of transmitting 12 bits rather than 7.*

Practical Work 2.1. *Hamming Implantation*$(7, 4)$*. In the following, we focus on the version of Hamming*$(7, 4)$ *seen in this chapter.*

1. *Develop a program that implements Hamming*$(7, 4)$ *by exploiting matrix products with* H_3 *and* G_3 *seen in this chapter. We'll make sure we work in* $\mathbb{F}_2$*.*
2. *Evaluate the algorithm on all words of* $\mathbb{F}_2^4$ *with all possible errors. Check that the original word is still the one that is found.*
3. *Check that the product* $G_3 \times H_3$ *is zero.*

Practical Work 2.2. *Hamming*$(2^k - 1, 2^k - k - 1)$*. In this practical work, the aim is to generalize the construction of* H_k *and* G_k*.*

1. *Develop a program that generates* H_k *and* G_k *according to the constraints outlined in this chapter.*
2. *Evaluate the algorithm on all words from* $\mathbb{F}_2^{2^k-k-1}$ *with all possible errors, for* $k = 3, \ldots, 5$*. Check that the original word is still the one that is found.*

Chapter 3

Reed-Solomon's codes $RS(n, k, m)$

This chapter uses elements from [Rou09, Wik17].

In the following, we consider m, n, k, t, each in $\mathbb{N}$ with $n = k + 2t$ and:

- m is the number of bits per used symbol; if each symbol is a character, m can be equal to 8 for example and we have bytes;
- k is the number of symbols contained in the original message;
- n is the number of symbols transmitted.
- $2t$ is the number of redundancy symbols; according to the relationship given above, the t number is deduced from n and k; it is therefore not a code parameter;

These codes are noted as $RS(n, k, m)$. Reed-Solomon codes can correct two types of errors: errors due to changes in data and errors resulting from the loss of information as long as the relation $2E + S \leq n - k$ is satisfied, where E is the number of errors and S is the number of erasures in the block. If there is nor erasure, the maximum number of corrected error is $\frac{n-k}{2}$, *i.e.* t.

3.1 Give the set $\mathbb{F}_{2^m}$ a field structure

Each symbol is a character that will be stored as a m bit vector, *i.e.* as a number in $\mathbb{F}_{2^m}$. Let A be a message consisting of k symbols $(u_1, \ldots, u_k)$. Each u_i is part of the $\mathbb{F}_{2^m}$ set.

For example, the message "13" of $k = 2$ characters would be translated into $\mathbb{F}_{2^2}$ into $u = (1, 3)$. In the following we try to correct $t = 1$ errors. We therefore work with $RS(4, 2, 2)$.

The Reed-Solomon algorithm requires a field structure for the set $\mathbb{F}_{2^m}$ in particular that the only divisor of 0 is 0. However, if we equip this set of classical operators with sum and product modulo 2^m, the set $\mathbb{F}_{2^m}$ is not a field. Indeed, for any integer i, $1 \leq i < m$, on $2^i \times 2^{m-i} \equiv 2^m \equiv 0 \bmod(m)$. Thus 2^i, not zero, is a divisor of 0. The following section shows that it is possible to build sum and product operators on this set that gives it a field structure.

3.1.1 Calculs modulo $p(x)$

In the same way that we can calculate "modulo p", it is possible to calculate "modulo a polynomial $p(x)$".

For example, $p(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$. The square of the polynomial $(x^2 + 1)$ modulo $p(x)$ in $\mathbb{F}_2[x]$ is

$$\begin{aligned}(x^2 + 1) \times (x^2 + 1) &\equiv x^4 + 2x^2 + 1 \\ &\equiv x^4 + 1 \\ &\equiv x + 1 (\bmod\, p(x)).\end{aligned}$$

Indeed, it is enough to write the Euclidean division to be convinced of this:

$$\begin{array}{rrrrr|rrr}
x^4 & & & & +1 & x^2 & +x & +1 \\
\cline{6-8}
x^4 & +x^3 & +x^2 & & & x^2 & +x & \\
\cline{1-3}
 & x^3 & +x^2 & & & & & \\
 & x^3 & +x^2 & +x & & & & \\
\cline{2-4}
 & & & x & +1 & & &
\end{array}$$

and therefore $x^4 + 1 = (x^2 + x + 1) \times (x^2 + x) + x + 1$.

Exercise 3.1. *In $\mathbb{F}_2[x]$, calculate $x^2(x^2 + 1) \bmod (x + 1)$.*

3.1.2 The set $\mathbb{F}_2[x]/p(x)$ of the polynomials modulo $p(x)$

In the following, $p(x) \in \mathbb{F}_2[x]$ is a polynomial in x of degree m whose variables and coefficients belong to $\mathbb{F}_2$. The quotient $\mathbb{F}_2[x]/p(x)$ is the set of polynomials of $\mathbb{F}_2[x]$ modulo $p(x)$. All these polynomials are of degree $q < m$. The set $\mathbb{F}_2[x]/p(x)$ therefore contains all the polynomials of the form $p_0 + p_1 x + \ldots p_{m-1} x^{m-1}$, which are 2^m since each p_i belongs to $\mathbb{F}_2$.

For example, if $p(x) = x^2 + x + 1$, we have $\mathbb{F}_2[x]/p(x) = \{0, 1, x, x + 1\}$. If we take $p'(x) = x^2 + 1$, we have $\mathbb{F}_2[x]/p'(x) = \{0, 1, x, x + 1\}$. The sets $\mathbb{F}_2[x]/p(x)$ and $\mathbb{F}_2[x]/p'(x)$ both contain 2^2 elements. However, we note that $(x + 1)^2 = x^2 + 1$ in $\mathbb{F}_2[x]$. In addition, as

$$\begin{aligned} (x+1)^2 = x^2 + 1 &\equiv 0 \bmod (x^2+1) \rightsquigarrow (x+1) \text{ is a divisor of 0,} \\ &\equiv 1 \bmod (x^2 + x + 1), \end{aligned}$$

we can conclude that $\mathbb{F}_2[x]/p'(x)$ is not a field

The following section shows how to choose $p(x)$ to build a field.

3.1.3 Construction of a field from an irreducible polynomial

In $\mathbb{F}_2[x]$ let $p(x)$ be a polynomial of degree m. The set $\mathbb{F}_2[x]/p(x)$ is a 2^m element field if and only if $p(x)$ is *irreducible* in $\mathbb{F}_2[x]$, *i.e.* $p(x)$ is not the product of 2 polynomials in $\mathbb{F}_2[x]$ of degree below m.

We immediately have that $x^2 + 1 = (x + 1)^2$ so the polynomial $x^2 + 1$ is not irreducible in $\mathbb{F}_2[x]$. Let's consider $p(x) = x^2 + x + 1$ in $\mathbb{F}_2[x]$. If it was not irreducible, it would be the product of two degree 1 polynomials that would determine its roots. However, neither 0 nor 1 are roots. Thus, it is irreducible.

Exercise 3.2. *Irreducibility. In $\mathbb{F}_2[x]$, what about the irreducibility of the polynomials x^2+1, x^3+x^2+x, x^3+x+1 and $x^3 + x^2 + x + 1$?*

By taking $p(x) = x^2 + x + 1$ and considering $\mathbb{F}_2[x]/p(x) = \{0, 1, x, x + 1\}$, we can construct the following multiplication and addition tables:

$\times$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Finally, we can build a bijection of $\mathbb{F}_{2^m}$ in $\mathbb{F}_2[x]/p(x)$ which associates the i^{th} element of $\mathbb{F}_{2^m}$ with the i^{th} element of $\mathbb{F}_2[x]/p(x)$, if the elements of a set can be ordered. Thanks to this we can define a product and a sum on $\mathbb{F}_{2^m}$ which gives it a field structure. For $\mathbb{F}_{2^2}$ it would be:

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

$+$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

(3.1)

By construction for any element $e \in \mathbb{F}_{2^m}$, we have $e + e = 0$.

Exercise 3.3. *Field from an irreducible polynomial.*

1. *Find the only irreducible polynomial on $\mathbb{F}_2$ of degree 2, both of degree 3 and all three of degree 4.*
2. *Build the addition and multiplication tables on the field $\mathbb{F}_8$ with 8 elements. We will consider the irreducible polynomial $x^3 + x + 1$.*

Practical Work 3.1. *Implement the addition and subtraction tables of $\mathbb{F}_8$. How to do for $\mathbb{F}_{2^8}$?*

3.2 Reed-Solomon Code $RS(n, k, m)$

The Reed-Solomon code $RS(n, k, m)$ will be defined as a polynomial with a degree of n or less on $\mathbb{F}_{2^m}[x]$. Let's see how to build this polynomial.

3.2.1 Polynomial $A(x)$ containing the information to be transmitted

From the message $u = (u_1, \ldots, u_k)$, where each u_i belongs to the field $\mathbb{F}_{2^m}$ (built as in the previous section) the Reed-Solomon algorithm builds the polynomial $A(x)$ defined as:

$$A(x) = u_1 x^{k-1} + u_2 x^{k-2} + \cdots + u_{k-1} x^1 + u_k. \tag{3.2}$$

This polynomial of degree less than or equal to $k-1$ belongs to $\mathbb{F}_{2^m}[x]$ which is the set of polynomials whose coefficients belong to $\mathbb{F}_{2^m}$.

In the previous example, we would have $A(x) = 1x + 3$.

3.2.2 Primary root of the field $\mathbb{F}_{2^m}$

There is a non-zero element α of $\mathbb{F}_{2^m}$ such that all other non-zero elements of this field can be expressed as a power of α. The α element is called a *primitive root* of $\mathbb{F}_{2^m}$.

$\mathbb{F}_{2^2} \setminus \{0\} = \{1, 2, 3\}$. Let's calculate the successive powers of 2 from the equation (3.1):

$$\begin{aligned} 2^0 &= 1 \\ 2^1 &= 2 \\ 2^2 &= 3. \end{aligned}$$

We can therefore conclude that 2 is a primitive root of $\mathbb{F}_{2^2}$, while 1 is not.

Exercise 3.4. *Let be defined the sum and product over $\mathbb{F}_{2^3}$ as follows.*

$+$	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

$\times$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

It is the solution of exercise 3.3. Prove that any $\alpha \in \{2, 3, 4, 5, 6, 7\}$ is a primitive root of $\mathbb{F}_8$.

3.2.3 Control polynomial $B(x)$

Let α be a primitive root of the field $\mathbb{F}_{2^m}$. First, we build the *generator* polynomial of the code as follows:

$$G(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t})$$

whose roots are $\alpha, \alpha^2, \ldots, \alpha^{2t}$. This polynomial depends only on the number of errors to be corrected (*i.e.* t) and the primitive root α of the field $\mathbb{F}_{2^m}$.

The *control* polynomial $B(x)$ is then defined by:

$$B(x) = A(x) \times x^{2t} \bmod(G(x)) \tag{3.3}$$

In the previous example, we would have successively

$$\begin{aligned} G(x) &= (x + 2)(x + 2^2) \\ &= (x + 2)(x + 3) \\ &= (x^2 + x + 1) \end{aligned}$$

In addition, $A(x) \times x^2 = (1x+3) \times x^2 = x^3 + 3x^2 = (x^2+x+1)(x+2) + 3x + 2$. Indeed, we have

$$\begin{array}{cccc|cc} x^3 & +3x^2 & & & x^2 & +x & +1 \\ \cline{5-6} x^3 & +x^2 & +x & & x & +2 & \\ \cline{1-3} & 2x^2 & +x & & & & \\ & 2x^2 & +2x & +2 & & & \\ \cline{2-4} & & 3x & +2 & & & \end{array}$$

Thus,

$$B(x) = 3x + 2 \bmod (G(x).$$

3.2.4 Code $C(x)$

The code $C(x)$ is defined by $C(x) = A(x) \times x^{2t} + B(x)$ and is naturally equal to 0 for $x = \alpha^j$, $1 \leq j \leq 2t$.

Indeed, we successively have for j such that $1 \leq j \leq 2t$:

$$\begin{array}{rcl} C(\alpha^j) & = & A(\alpha^j) \times (\alpha^j)^{2t} + B(\alpha^j) \\ & = & A(\alpha^j) \times (\alpha^j)^{2t} + A(\alpha^j) \times (\alpha^j)^{2t} + d.(G(\alpha^j)) \text{ thanks to equ. (3.3)} \\ & = & A(\alpha^j) \times (\alpha^j)^{2t} + A(\alpha^j) \times (\alpha^j)^{2t} + d.0 \text{ since } G(\alpha^j) = 0 \\ & = & 0 \end{array}$$

In the previous example the Reed-Solomon code corresponding to the message m is $C(x) = x^3 + 3x^2 + 3x + 2$.

3.3 Error detection and correction

During the transmission of the polynomial $C(x)$, errors in some coefficients may occur. $D(x)$ is considered to be the received polynomial.

For instance let us consider the received polynomial is $D(x) = 2x^3 + 3x^2 + 3x + 2$.

3.3.1 Detection only

The receiver computes syndroms $S_j = D(\alpha^j)$ in $\mathbb{F}_{2^m}$ for all j, $1 \leq j \leq t$.

- If $S_j = 0$ for all j, $1 \leq j \leq 2t$, he/she considers that there was no transmission error, *i.e.* $D(x) = C(x)$. The original message m is found using the k coefficients of the highest degree terms of the polynomial $D(x)$.
- If there is some j, $1 \leq j \leq 2t$ such that $S_j = D(\alpha^j)$ is not zero, there has been a transmission error on at least one of the coefficients that will need to be corrected.

For instance, for $\alpha = 2$ we calculate $D(\alpha)$ as given in the following table:

j	α^j	$D(\alpha^j)$	S_j
1	2	$2.1 + 3.3 + 3.2 + 2 = 2 + 2 + 1 + 2 = 3 =$	S_1
2	3	$2.1 + 3.2 + 3.3 + 2 = 2 + 1 + 2 + 2 = 3 =$	S_2

Both syndromes S_1 and S_2 are not null. There is an error, so.

3.3.2 Error correction

The final objective is to find the polynomial $E(x)$ of errors and to retrieve the initial message $C(x) = D(x) + E(x)$. Let

$$E(x) = \sum_{r=1}^{\nu} e_{i_r} x^{i_r} \tag{3.4}$$

be the error polynomial we want to find. Notice that

- $E(x)$ contains ν not null coeffecients, which is is thus the number of errors; The number of errors ν is less that t, by hypothesis.
- i_r are the idices, between 0 and $n-1$ where there is an error.
- e_{i_r} is the value of the error.

3.3.2.1 Finding the number of errors.

Since $C(x) = D(x) + E(x)$, we thus have $E(x) = D(x) + C(x)$. Particularely, for j, $1 \leq j \leq 2t$, we have

$$E(\alpha^j) = D(\alpha^j) + C(\alpha^j) = D(\alpha^j) = S_j = \sum_{r=1}^{\nu} e_{i_r} \alpha^{j.i_r} \tag{3.5}$$

thanks to (3.4).

Let us consider the polynomial $\Lambda(x) = \prod_{r=1}^{\nu}(1 + \alpha^{i_r} x)$. This polynomial can be expanded into

$$\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_\nu x^\nu$$

whose $\lambda_.$ have to be calculated. To achieve this, we note that

$$\begin{array}{rcl}
S_{\nu+j-1}\lambda_1 + S_{\nu+j-2}\lambda_2 + \cdots + S_j\lambda_\nu & = & \sum_{l=1}^{\nu} S_{v+j-l}.\lambda_l \\
& = & \sum_{l=1}^{\nu} \left(\sum_{r=1}^{\nu} e_{i_r} \alpha^{(v+j-l).i_r} \right) \lambda_l \text{ definition (3.5)}, \\
& = & \sum_{r=1}^{\nu} e_{i_r} \alpha^{(v+j).i_r} \sum_{l=1}^{\nu} \lambda_l.\alpha^{-l.i_r} \text{reordering} \\
& = & \sum_{r=1}^{\nu} e_{i_r} (\alpha^{i_r})^{v+j} \left(\Lambda(\alpha^{-i_r}) + 1 \right) \\
& = & \sum_{r=1}^{\nu} e_{i_r} (\alpha^{i_r})^{v+j} \Lambda(\alpha^{-i_r}) + \sum_{r=1}^{\nu} e_{i_r} (\alpha^{i_r})^{v+j} \\
& = & \sum_{r=1}^{\nu} e_{i_r} (\alpha^{i_r})^{v+j} \text{ since } \Lambda(\alpha^{-i_r}) \text{ is null} \\
& = & S_{v+j}
\end{array}$$

So we have a system of a ν linear equations, at most.

$$\begin{array}{rcl}
S_\nu\lambda_1 + S_{\nu-1}\lambda_2 + \cdots + S_1\lambda_\nu & = & S_{\nu+1} \\
S_{\nu+1}\lambda_1 + S_\nu\lambda_2 + \cdots + S_2\lambda_\nu & = & S_{\nu+2} \\
& \vdots & \\
S_{2\nu-1}\lambda_1 + S_{2\nu-2}\lambda_2 + \cdots + S_\nu\lambda_\nu & = & S_{2\nu}
\end{array}$$

In addition, the highest value ν less than or equal to t for which the determinant of this system is not zero is precisely the number ν equal to the number of errors transmitted. So we start from $\nu = t$, and if the determinant is zero, we decrease ν until we get a non-zero determinant.

On our example, $\nu = 1 = t$ and we have thus:

$$\begin{array}{rcl}
S_1\lambda_1 & = & S_2 \\
3\lambda_1 & = & 3
\end{array}$$

which allows to have one solution, which is $\lambda_1 = 1$.

3.3.2.2 Finding error's locations.

Once ν is found, the system is solved, which defines the polynomial Λ. Roots of this polynomial are further found. For each root r, its inverse is expressed as a power of α: there exists some i_r such that $r.\alpha^{i_r}$ is equal to 1. The error locations are $\{i_1, \ldots, i_\nu.\}$.

On our example, we can deduce that $\Lambda(x) = 1 + x$. This polynomial admits one single root, which is $r = 1$. we have thus to solve

$$\begin{array}{rcl}
r.\alpha^{i_r} & = & 1 \\
1.2^{i_1} & = & 1
\end{array}$$

whose admits one solution $i_1 = 3$.

3.3.2.3 Finding error's values

In equation (3.5), ($S_j = \sum_{r=1}^{\nu} e_{i_r} \alpha^{j.i_r}$), we can now solve this system leading to the errors $e_1, \ldots, e_r$. The polynomial $E(x)$ is thus defined and $C(x) = D(x) + E(x)$ can be extracted.

On our example, it remains to solve equations (3.5) $S_j = \sum_{r=1}^{\nu} e_{i_r} \alpha^{j.i_r}$ for j, $1 \leq j \leq 2t$. It is equivlent to the systeme

$$\left\{ \begin{array}{rcl} S_1 & = & e_3\alpha^3 \\ S_2 & = & e_3\alpha^{2\times 3} \end{array} \right. \text{ which is equivalent to } \left\{ \begin{array}{rcl} 3 & = & e_3.1 \\ 3 & = & e_3.1 \end{array} \right.$$

The error e_3 is thus equal to 3. The error polynomial is thus $E(x) = 3x^2$. Finally,

$$\begin{aligned} C(x) &= D(x) + E(x) \\ &= 2x^3 + 3x^2 + 3x + 2 + 3x^3 \\ &= x^3 + 3x^2 + 3x + 2 \end{aligned}$$

Exercise 3.5. *We want now to correct 2 errors in a message of length 2 for 4 types of caracters.*

1. *Show that $RS(6, 2, 2)$ meets the above requirements.*
2. *Show that $G(x) = x^4 + 3x^3 + x + 3$ is a correct generator polynomial.*
3. *What is the message codded into $D_1(x) = 3x^5 + 3x^4 + x^3 + 3x^2 + 3x + 1$?*
4. *What is the message codded into $D_2(x) = x^5 + 2x^4 + x^3 + x^2 + 3x + 1$?*
5. *What is the message codded into $D_3(x) = 2x^5 + 2x^4 + 2x^2 + x$?*

Practical Work 3.2. *Playing with a Reed-Solomon implementation.*

1. *Install a Reed-Solomon implementation (for example `rs.py` extracted from https://rextester.com/ZMBYT68318) on your computer.*
2. *Execute the code and understand the evaluation part.*
3. *How many errors can be corrected? How many errors are introduced in the message?*

Practical Work 3.3. *Implementation on a PyBoard with MicroPython*

1. *Implement all the previous codes (Hamming74 and Reed-solomon) on a PyBoard.*
2. *When decoded word is equal to the original one, make the green led flashing 10 times.*
3. *Evaluate them.*

Bibliography

[Rou09] *Mathématiques et Technologie*. Springer, 2009.

[Sha01] Claude E Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[Wik17] Wikipédia. Code de reed-solomon — wikipédia, l'encyclopédie libre, 2017. [En ligne; Page disponible le 16-sept-2019].