Ecole Polytechnique Fédérale de Lausanne

Sciences de l'information

Série 13

Olivier Cloux

13.1

1. Nous posons nos éléments tels que : $a_1 = 0,\ a_2 = 1,\ a_3 = 2,\ a_4 = 3,\ a_5 = 4,\ a_6 = 5,\ a_7 = 6$. Depuis là, nous prenons 4 vecteur indépendants[1] de notre code, à savoir

$$(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)$$

. Cela nous donne la matrice

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ (a_1)^2 & (a_2)^2 & (a_3)^2 & (a_4)^2 & (a_5)^2 & (a_6)^2 & (a_7)^2 \\ (a_1)^3 & (a_2)^3 & (a_3)^3 & (a_4)^3 & (a_5)^3 & (a_6)^3 & (a_7)^3 \end{pmatrix}.$$

En remplaçant les éléments (toujours dans F_7), cela nous donne

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 9 & 16 & 25 & 36 \\ 0 & 1 & 8 & 27 & 64 & 125 & 216 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 2 & 2 & 4 & 1 \\ 0 & 1 & 1 & 6 & 1 & 6 & 6 \end{pmatrix}$$

2. Pour trouver la forme systématique, nous devons réduire notre matrice G, en gardant en tête que nous travaillons dans F_7 (toujours pas de division, etc,...). Notre matrice G peut se simplifier en

[1] car $k = 4$

quelques 8 étapes :

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 2 & 2 & 4 & 1 \\ 0 & 1 & 1 & 6 & 1 & 6 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 6 & 5 & 4 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 2 & 6 & 5 & 6 & 2 \\ 0 & 0 & 6 & 3 & 4 & 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 6 & 5 & 4 & 3 & 2 \\ 0 & 1 & 0 & 4 & 6 & 6 & 4 \\ 0 & 0 & 1 & 3 & 6 & 3 & 1 \\ 0 & 0 & 1 & 4 & 3 & 6 & 0 \end{pmatrix}$$
$$\sim \begin{pmatrix} 1 & 0 & 0 & 1 & 3 & 6 & 3 \\ 0 & 1 & 0 & 4 & 6 & 6 & 4 \\ 0 & 0 & 1 & 3 & 6 & 3 & 1 \\ 0 & 0 & 0 & 1 & 4 & 3 & 6 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 & 6 & 3 & 4 \\ 0 & 1 & 0 & 0 & 4 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 1 & 4 & 3 & 6 \end{pmatrix} = G' = [I_4 \ P]$$

3. En utilisant le théorème 13.1, nous pouvons poser que :

$$H = [-P^T \ I_{7-4}] = \begin{pmatrix} -6 & -4 & -1 & -4 & 1 & 0 & 0 \\ -3 & -1 & -1 & -3 & 0 & 1 & 0 \\ -4 & -1 & -4 & -6 & 0 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 3 & 6 & 3 & 1 & 0 & 0 \\ 4 & 6 & 6 & 4 & 0 & 1 & 0 \\ 3 & 6 & 3 & 1 & 0 & 0 & 1 \end{pmatrix}$$

4. Par définition, les lignes de G sont des mots de code (valides), donc leur syndrome est de 0. Ainsi, $\boxed{GH^T = 0_{4\times 3}}$ (la matrice nulle de taille adéquate, à savoir 4×3). En abusant de paranoïa et en effectuant les calculs, nous trouvons bien cette matrice nulle.

5. Comme C est un code de Reed-Solomon, il atteint la borne de Singleton. Ainsi, $\boxed{d_{min}(C) = n - k + 1 = 7 - 4 + 1 = 4}$

6. Considérons le premier effacement comme étant x_2 (car seconde position), et les suivants comme x_5 et x_7. Comme il s'agit d'un mot de code valide, son syndrome doit valoir 0. Cela nous permet de définir les équations suivantes :

$$\vec{y}H^T = 0 \rightarrow \begin{cases} 5 & +3x_2 & +30 & +12 & +x_5 & +0 & +0 & = 0 \\ 20 & +6x_2 & +30 & +16 & +0 & +3 & +0 & = 0 \\ 15 & +6x_2 & +15 & +4 & +0 & +0 & +x_7 & = 0 \end{cases} \rightarrow \begin{cases} 3x_2 + x_5 + 5 = 0 \\ 6x_2 + 6 = 0 \iff 6x_2 = 1 \\ 6x_2 + x_7 + 6 = 0 \end{cases}$$
$$\rightarrow \begin{cases} x_5 = 5 \\ x_2 = 6 \\ x_7 = 0 \end{cases}$$

En remplaçant ces corrections dans notre mot, cela nous donne

$$\boxed{\vec{y} = (5, 6, 5, 4, 5, 3, 0)}$$

7. Le syndrome se trouve comme d'habitude, avec $\vec{z}H^T$, ce qui nous donne $\boxed{(0, 1, 0)}$. Ainsi, pour de corriger l'erreur, nous devons trouver un moyen de modifier le mot afin de ne toucher que la seconde partie du syndrome, mais sans toucher à la première et troisième partie. Nous nous rendons vite compte que nous pouvons faire ça de manière simple, en modifiant une des trois dernières coordonnées du mot (à cause de la matrice identité au bas de H^T). Pour cela, il suffit de modifier l'avant dernière coordonnée de $\vec{z}$, en changeant le 0 en 6, afin d'ajouter 6 au syndrome, ce qui aura pour effet de le modifier en (0,7,0) et donc en (0,0,0). Notre mot ainsi corrigé devient $\boxed{(2, 2, 0, 4, 1, 6, 6)}$

8. Le meilleur moyen de corriger une erreur simple de manière informatique (car nous ne pouvons pas raisonner comme précédemment pour une erreur quelconque) est de tester tous les mots à distance 1 de notre mot. Cela se fait grâce à l'algorithme suivant :

Algorithm 1 correcteur.c

```
if (yH^T = 0) then
    return y                                   (▷) Déjà s'assurer que le mot donné est correcte ou non
else
    for (i = 1, i <= 7, i++) do                (▷) Afin d'inspecter indépendamment les 7 coordonnées de y
        for (j = 1, j <= 7, j++) do            (▷) Afin d'ajouter 1 x7, et faire le tour du modulo
            Modulo(y[i] + 1, 7)                (▷) La fonction Modulo(a,b) retourne a mod b
            if (yH^T = 0) then
                return y
            end if
        end for
    end for
    return noCorrectionPossible (▷) Sans retour avant, cela signifie que le mot ne peut pas être corrigé
end if
```