MT10/P13 - TP3 : Corps finis et codes correcteurs

Table des matières

Répondre aux questions posées au fil du texte. Les démonstrations des résultats cités sont rédigées dans le document [1].

1 Construction de corps finis

1.1 Rappels théoriques sur les corps finis

Si $p \in \mathbb{P}$, l'anneau $\mathbb{Z}/p\mathbb{Z}$ des entiers *modulo* p est un corps qu'on notera [1] $\mathbb{F}_p$. Un corps fini est nécessairement de cardinal $q = p^n$, où $p \in \mathbb{P}$ est sa caractéristique et $n \in \mathbb{N}$. Un tel corps admet $\mathbb{F}_p$ comme sous-corps, et peut être vu comme un $\mathbb{F}_p$-espace vectoriel de dimension n. On sait aussi que tout corps fini est commutatif (théorème de Wedderburn).

Si $P \in \mathbb{F}_p[X]$ est un polynôme irréductible, alors l'anneau quotient $\mathbb{F}_p[X]/(P)$ des polynômes *modulo* P est un corps. Si P est de degré n, le corps ainsi construit est de cardinal $q = p^n$. Pour tout $p \in \mathbb{P}$, il existe des polynômes de $\mathbb{F}_p[X]$ irréductibles de tout degré. De plus, deux corps

1. Le $\mathbb{F}$ est mis pour *field* qui veut dire champ. Les corps étaient auparavant appelés "champs de Galois", d'où aussi la notation `GF`, pour *Galois field*, utilisé dans `Sage`.

finis de même cardinal sont isomorphes. A isomorphisme près, tout corps fini s'obtient donc comme $\mathbb{F}_p[X]/(P)$ pour un certain $P \in \mathbb{F}_p[X]$ irréductible.

1.2 Rudiments de Sage pour les corps finis

De façon générale, on utilise la complétion (touche TAB) pour obtenir les méthodes qui s'appliquent à un objet. Voici une liste non exhaustive de commandes utiles dans le contexte des corps finis.

- les corps premiers $\mathbb{F}_p$: `F5=GF(5)`, faire `F5.`+TAB pour la liste des méthodes associées
- les autres corps finis $\mathbb{F}_q$ avec $q = p^n$: `Fq.<a>=GF(q,name='a')` et Sage choisit le *modulo* (polynôme irréductible de degré n), qu'on peut récupérer par `Fq.modulus()`
- $\mathbb{F}_p[X]$ et les polynômes : `R.<X>=GF(p)['X']`, ou `PolynomialRing`
- l'espace vectoriel $\mathbb{F}_p^k$: `V=VectorSpace(Fp,k)`

Remarque : si `Fq` désigne le corps fini à $q = p^n$ éléments, alors `Fq.vector_space()` renvoie le $\mathbb{F}_p$-espace vectoriel de dimension n isomorphe à $\mathbb{F}_q$ (comme $\mathbb{F}_p$-espace vectoriel), autrement dit $\mathbb{F}_p^n$.

1.3 Dénombrement des polynômes irréductibles et unitaires de $\mathbb{F}_p[X]$

On dit qu'un polynôme $P \in \mathbb{F}_p[X]$ est unitaire si son cœfficient dominant est égal à 1. Tout polynôme non nul de $\mathbb{F}_p[X]$ est égal au produit d'un polynôme unitaire par un élément de $\mathbb{F}_p^*$.

1.3.1 Factorisation de $X^q - X$ dans $\mathbb{F}_p[X]$

Soit $q = p^n$. Alors $X^q - X$ est le produit dans $\mathbb{F}_p[X]$ des polynômes $P \in \mathbb{F}_p[X]$ irréductibles et unitaires dont le degré divise n.

$$X^q - X = \prod_{\substack{P \in \mathbb{F}_p[X] \\ \text{irréductible, unitaire} \\ \deg(P) | n}} P(X) \tag{1}$$

Question 1: A l'aide de Sage, vérifier cette factorisation pour $X^q - X$ avec $q = 2^6$.

Désignons par $\mathrm{Irr}_p(k)$ le nombre de polynômes de degré k, irréductibles et unitaires, de $\mathbb{F}_p[X]$.

$$\mathrm{Irr}_p(k) \overset{\text{déf}}{=} \mathrm{card}\,\{P \in \mathbb{F}_p[X] \ : \quad \deg(P) = k, \text{irréductible, unitaire}\} \tag{2}$$

L'égalité des degrés dans la factorisation de $X^q - X$ donne la relation :

$$p^n = \sum_{d|n} d\ \mathrm{Irr}_p(d). \tag{3}$$

Pour calculer les $\mathrm{Irr}_p(k)$, il suffit "d'inverser cette relation", et cela se fait grâce à la fonction de Möbius.

1.3.2 La fonction de Möbius

La fonction de Möbius est définie sur $\mathbb{N} \setminus \{0\}$ par :

$$\begin{array}{rcl} \mu(1) & = & 1 \\ (\forall n > 1) \qquad \mu(n) & = & \left| \begin{array}{ll} 0 & \text{si } n \text{ a un facteur carré} \\ (-1)^r & \text{si } n \text{ est produit de } r \text{ nombres premiers} \\ & \quad \text{deux à deux distincts} \end{array} \right. \end{array} \tag{4}$$

Comme l'indicatrice d'Euler, la fonction de Möbius est multiplicative, au sens où :

$$(\forall n, n' \in \mathbb{N} \setminus \{0\}) \qquad \text{pgcd}(n, n') = 1 \implies \mu(nn') = \mu(n)\mu(n') \tag{5}$$

On a aussi une sorte de formule d'Euler pour la fonction de Möbius :

$$(\forall n \in \mathbb{N} \setminus \{0\}) \qquad \sum_{d|n} \mu(d) = \left| \begin{array}{lll} 1 & \text{si} & n = 1 \\ 0 & \text{si} & n > 1 \end{array} \right. \tag{6}$$

Cette formule d'Euler très spéciale permet d'obtenir la formule d'inversion souhaitée.

Proposition 1 (formule d'inversion de Möbius) *Soit une application $f : \mathbb{N} \setminus \{0\} \to \mathbb{C}$. Alors, si pour tout $n \in \mathbb{N} \setminus \{0\}$ on a posé $F(n) = \sum_{d|n} f(d)$, on a :*

$$(\forall n \in \mathbb{N} \setminus \{0\}) \qquad f(n) = \sum_{d|n} \mu(\frac{n}{d}) F(d)$$

Question 2: `Sage` possède la fonction `moebius` qui calcule les $\mu(n)$. Consulter l'aide, puis vérifier :

1. que $\mu(n) \in \{-1; 0; 1\}$ pour les 100 premiers entiers naturels ;
2. la formule d'Euler pour la fonction de Möbius pour les 100 premiers entiers naturels ;
3. la formule d'inversion de Möbius pour le calcul de $\phi(100)$, où ϕ désigne l'indicatrice d'Euler, dont on rappelle qu'elle vérifie la formule d'Euler $n = \sum_{d|n} \phi(d)$.

1.3.3 Calcul du nombre de polynômes unitaires irréductibles de degré d dans $\mathbb{F}_p[X]$

La formule d'inversion de Möbius fournit finalement le résultat souhaité :

$$(\forall n \in \mathbb{N} \setminus \{0\}) \qquad \text{Irr}_p(n) = \frac{1}{n} \sum_{d|n} \mu(\frac{n}{d}) p^d \tag{7}$$

Question 3: Ecrire une fonction `Irr` qui retourne $\text{Irr}_p(n)$ quand p et n sont donnés en entrée. Dresser un tableau de $\text{Irr}_p(n)$ pour $p = 2, 3, 5$ et $n = 1, \ldots, 10$.

1.4 Calcul de polynômes unitaires irréductibles de $\mathbb{F}_p[X]$

Question 4: Déterminer tous les polynômes irréductibles de $\mathbb{F}_2[X]$ de degré inférieur ou égal à 10.

2 Les codes de Reed et Solomon

2.1 Définition des codes de Reed-Solomon généralisés (GRS)

Le message (numérisé) à transmettre est formé de mots de k lettres de $\mathbb{F}_q$; chaque mot $\mathbf{x} \in \mathbb{F}_q^k$ est interprété comme un polynôme $f \in \mathbb{F}_q[X]_k$ de degré $< k$ (c'est la signification du k en indice de $\mathbb{F}_q[X]_k$) dont les cœfficients sont les lettres du mot.

Définition 2 (code de Reed-Solomon généralisé)
Soient $0 \leq k \leq n \leq q$, $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_q^{*n}$ *et*

$$\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}_q^n \quad \textit{tel que } i \neq j \Longrightarrow \alpha_i \neq \alpha_j. \tag{8}$$

On définit alors le code de Reed-Solomon par

$$GRS_{n,k}(\alpha, \mathbf{v}) \stackrel{déf}{=} \{ev_{\alpha,\mathbf{v}}(f) \ : f \in \mathbb{F}_q[X]_k\} \subset \mathbb{F}_q^n, \tag{9}$$

où l'évaluation se fait par

$$ev_{\alpha,\mathbf{v}}(f) = (v_0 f(\alpha_0), v_1 f(\alpha_1), \ldots, v_{n-1} f(\alpha_{n-1})) \ \in \mathbb{F}_q^n. \tag{10}$$

Question 5: Ecrire une fonction `codeGRS` qui prenne en entrée un bloc de message $\mathbf{x} \in \mathbb{F}_q^k$ et qui retourne le code correspondant $\mathbf{y} = ev_{\alpha,\mathbf{v}}(f) = (v_0 f(\alpha_0), v_1 f(\alpha_1), \ldots, v_{n-1} f(\alpha_{n-1})) \ \in \mathbb{F}_q^n$. Cette fonction `codeGRS` admettra aussi comme argument les paramètres $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in \mathbb{F}_q^{*n}$ et $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}_q^n$.

2.2 Cas sans erreur : décodage des GRS par interpolation de Lagrange

Le codage GRS est une évaluation d'un polynôme en certains points. Son décodage est donc une interpolation : il s'agit de retrouver le polynôme à partir de ses valeurs prises en ces mêmes points. Pour cela, on utilise les polynômes de Lagrange.

Soit $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}_q^n$ tel que $i \neq j \Longrightarrow \alpha_i \neq \alpha_j$. Les polynômes de Lagrange sont définis par

$$(\forall i \in [\![0, n-1]\!]) \qquad \boxed{L_i(X) \ = \prod_{\substack{j \in [\![0,n-1]\!] \\ j \neq i}} (X - \alpha_j)} \quad \in \mathbb{F}_q[X]_n \tag{11}$$

Ces polynômes rendent facile l'interpolation polynomiale par le lemme ci-dessous.

Lemme 3 (interpolation de Lagrange)
Pour tout polynôme $f \in \mathbb{F}_q[X]_n$, *on a l'identité*

$$f(X) = \sum_{i \in [\![0,n-1]\!]} f(\alpha_i) \left(L_i(\alpha_i)\right)^{-1} L_i(X) \tag{12}$$

Question 6: Ecrire une fonction `decodeGRS` qui réalise la réciproque de la fonction `codeGRS`. Tester sur quelques exemples. Que se passe-t-il si les points α_i ne sont pas deux à deux distincts ?

2.3 Simulation d'erreurs de transmission

Résumons les étapes du processus de codage-décodage par GRS :

1. Le message (numérisé) à transmettre est formé de mots de k lettres de $\mathbb{F}_q$; chaque mot $\mathbf{x} \in \mathbb{F}_q^k$ est interprété comme un polynôme $f \in \mathbb{F}_q[X]_k$ dont les cœfficients sont les lettres du mot.
2. Chaque $f \in \mathbb{F}_q[X]_k$ est codé par
$$\mathbf{y} = ev_{\alpha,\mathbf{v}}(f) = (v_0 f(\alpha_0), v_1 f(\alpha_1), \ldots, v_{n-1} f(\alpha_{n-1})) \ \in \mathbb{F}_q^n, \tag{13}$$
et $\mathbf{y}$ est envoyé dans le canal de transmission.
3. Des éventuelles erreurs font qu'à l'extrémité du canal de transmission on reçoit
$$\mathbf{y}' = \mathbf{y} + \mathbf{e} \ \in \mathbb{F}_q^n. \tag{14}$$
4. Il s'agira ensuite de récupérer le mot de code transmis $\mathbf{y}$ à partir du mot reçu $\mathbf{y}'$, et c'est l'objet de la section suivante.

Question 7: Ecrire une fonction `errTrans` qui simule des erreurs de transmission. `errTrans` prendra en entrée le mot de code $\mathbf{y}$ et retournera un mot $\mathbf{y}' \in \mathbb{F}_q^n$. Cette fonction admettra aussi comme argument un entier `Nb_err` indiquant le nombre d'erreurs de transmission, chaque erreur sera tirée aléatoirement dans $\mathbb{F}_q^*$.

Question 8: Vérifier sur un exemple que l'interpolation de Lagrange donne n'importe quoi dès qu'il y a une erreur de transmission.

3 Correction d'erreurs grâce aux GRS

Dans toute cette section, $0 \leq k \leq n \leq q$, et on pose $r = n - k$. On notera aussi $C = GRS_{n,k}(\alpha, \mathbf{v})$.

3.1 Le polynôme syndrome

Le mot reçu $\mathbf{y}'$ est éventuellement différent du mot de code envoyé $\mathbf{y}$. On peut tester si $\mathbf{y}' \in C$ grâce à la formulation de Goppa (voir [1, Théorème 26 p.18]). Ceci conduit à définir le **polynôme syndrome** par :

$$\begin{aligned} S(X) = S_{\mathbf{y}'}(X) &\overset{\text{déf}}{=} \sum_{i \in [\![0,n-1]\!]} y'_i u_i (1 - \alpha_i X)^{-1} \qquad [X^r] \\ &= \sum_{i \in [\![0,n-1]\!]} y'_i \left(v_i^{-1} L_i(\alpha_i)^{-1} \right) \left(\sum_{j \in [\![0,r-1]\!]} (\alpha_i X)^j \right) \\ &= \underbrace{S_{\mathbf{y}}(X)}_{=0} + S_{\mathbf{e}}(X) = S_{\mathbf{e}}(X) \quad \in \mathbb{F}_q[X]_r \end{aligned} \tag{15}$$

On a alors l'alternative :

(i) soit $S(X) = 0$, et donc $\mathbf{y}' \in C$ et on décode par $\boxed{\widehat{\mathbf{y}} = \mathbf{y}'}$ (on ne fait rien !).
Dans ce cas, on sait que :

– si $w(\mathbf{e}) \leq d-1 = r$, alors[2] $\mathbf{e} = 0$, et le décodage est un succès : $\widehat{\mathbf{y}} = \mathbf{y}$.
– si $w(\mathbf{e}) \geq d = r+1$, alors il est possible que, bien que $\mathbf{y}' \in C$, $\mathbf{y}'$ ne soit pas le mot de code envoyé. Le décodage peut être un échec.

(ii) soit $S(X) \neq 0$, donc $\mathbf{y}' \notin C$ et $w(\mathbf{e}) \geq 1$. On sait que si $w(\mathbf{e}) \leq \frac{d-1}{2} = \frac{r}{2}$, alors $\mathbf{y}$ est l'élément de C le plus proche de $\mathbf{y}'$. On décodera donc par $\boxed{\widehat{\mathbf{y}} = \mathbf{y}' - \widehat{\mathbf{e}}}$, où $\widehat{\mathbf{e}}$ reste à déterminer parmi les éléments de $\mathbb{F}_q^n$ de même syndrome que $\mathbf{e}$ (et donc que $\mathbf{y}'$). C'est l'objet de ce qui va suivre.

Question 9: Ecrire une fonction `Syndrome` qui prend en entrée un mot reçu $\mathbf{y}' \in \mathbb{F}_q^n$, ainsi que les paramètres α et $\mathbf{v}$, et retourne le polynôme syndrome correspondant $S(X)$.

Question 10: Vérifier sur des exemples que $\mathbf{y}' \in C \iff S(X) = 0$.

3.2 L'équation clef

On va voir que la seule connaissance du polynôme syndrome $S(X)$ suffit à reconstruire le mot de code transmis. Désignons par B l'ensemble des positions des erreurs, *i.e.*

$$B \stackrel{\text{déf}}{=} \{i \in [\![0, n-1]\!] \ : \quad e_i \neq 0\}\,. \tag{16}$$

On a alors

$$S(X) = \sum_{b \in B} e_b u_b (1 - \alpha_b X)^{-1} \qquad [X^r]. \tag{17}$$

Chassons les dénominateurs pour obtenir :

$$\underbrace{\prod_{b \in B} (1 - \alpha_b X)}_{=\sigma(X)} S(X) = \underbrace{\sum_{b \in B} e_b u_b \prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'} X)}_{=\omega(X)} \qquad [X^r]. \tag{18}$$

C'est la fameuse **équation clef** :

$$\sigma(X) S(X) = \omega(X) \qquad [X^r], \tag{19}$$

où, pour des raisons qui seront bientôt clarifiées,
– le polynôme $\sigma(X)$ s'appelle le **polynôme localisateur** des erreurs ;
– le polynôme $\omega(X)$ s'appelle le **polynôme évaluateur** des erreurs.

Remarquons que les polynômes σ et ω ne sont pas calculables à partir de leur définition puisqu'on ne connait ni B, ni les e_b. Par contre, on espérer trouver des solutions de l'équation clef qui vérifient en plus $\sigma(0) = 1$ et $\text{pgcd}(\sigma, \omega) = 1$ (lire [1, p.19-20]). Si on écrit l'équation clef sous la forme

$$u(X) X^r + \sigma(X) S(X) = \omega(X), \tag{20}$$

on pense irrésistiblement à l'algorithme d'Euclide, et c'est l'objet de la section suivante.

2. Par définition de la distance minimale, voir [1].

3.3 Résolution de l'équation clef par Euclide

Ce résultat fondamental est démontré dans [1, Théorème 29 p.22].

Théorème 4 (résolution de l'équation clef par Euclide)
Soit un code $GRS_{n,k}(\alpha, \mathbf{v})$ de paramètres (q, n, k, d). Posons $r = n - k$. Soit un polynôme syndrome non nul $S \in \mathbb{F}_q[X]_r$. Alors l'algorithme d'Euclide étendu tronqué se termine :

$$\begin{array}{l} r_0(X) = X^r \ ; \ u_0(X) = 1 \ ; v_0(X) = 0 \ ; \\ r_1(X) = S(X) \ ; \ u_1(X) = 0 \ ; v_1(X) = 1 \ ; \ j = 1 \ ; \\ \textit{Tant que } \deg(r_j) \geq \frac{r}{2} \textit{ faire } : \\ \quad r_{j-1}(X) = r_j(X)q_j(X) + r_{j+1}(X) \ ; \quad (\textit{division euclidienne}) \\ \quad u_{j+1}(X) = u_{j-1}(X) - u_j(X)q_j(X) \ ; \\ \quad v_{j+1}(X) = v_{j-1}(X) - v_j(X)q_j(X) \ ; \\ \quad j = j + 1 \ ; \\ \widetilde{\sigma}(X) = v_j(X) \ ; \\ \widetilde{\omega}(X) = r_j(X) \ ; \end{array} \tag{21}$$

A la fin de cet algorithme, on obtient

$$u_j(X)X^r + \widetilde{\sigma}(X)S(X) = \widetilde{\omega}(X) \quad et \quad \deg(\widetilde{\omega}) < \frac{r}{2}. \tag{22}$$

Si, de plus, le mot d'erreur $\mathbf{e}$ *est de poids* $w(\mathbf{e}) \leq \frac{d-1}{2} = \frac{r}{2}$*, alors on a*

$$\left\{ \begin{array}{rcl} \sigma(X) & = & \widetilde{\sigma}(0)^{-1}\widetilde{\sigma}(X) \\ \omega(X) & = & \widetilde{\sigma}(0)^{-1}\widetilde{\omega}(X) \end{array} \right. \tag{23}$$

où σ *et* ω *sont respectivement les polynômes localisateur et évaluateur pour l'erreur* $\mathbf{e}$*.*

Question 11: Ecrire une fonction `Clef` qui, à partir du polynôme syndrome $S(X)$, retourne les polynômes localisateur et évaluateur σ et ω. Vérifier sur un exemple que l'équation clef est satisfaite.

3.4 Localisation et évaluation des erreurs de transmission

La résolution de l'équation clef a permis de calculer les polynômes σ et ω. Il reste à en déduire l'erreur de transmission.

• Le polynôme localisateur des erreurs σ porte bien son nom puisqu'effectivement sa connaissance permet de calculer

$$\boxed{B = \left\{ b \in [\![0, n-1]\!] \ : \quad \sigma(\alpha_b^{-1}) = 0 \right\}}. \tag{24}$$

• Pour le calcul des erreurs e_b, on a (voir [1, Théorème 29 p.20])

$$(\forall b \in B) \qquad \omega(\alpha_b^{-1}) = e_b u_b \prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'}\alpha_b^{-1}), \tag{25}$$

et il s'agit donc de calculer $\prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'}\alpha_b^{-1})$. Pour cela, remarquons que

$$(\forall b \in B) \qquad \sigma(X) = (1 - \alpha_b X) \prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'} X). \tag{26}$$

Par dérivation, puis par évaluation, il vient successivement

$$\begin{aligned} \sigma'(X) &= -\alpha_b \prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'} X) + (1 - \alpha_b X) \left(\prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'} X) \right)' \\ \sigma'(\alpha_b^{-1}) &= -\alpha_b \prod_{\substack{b' \in B \\ b' \neq b}} (1 - \alpha_{b'}\alpha_b^{-1}). \end{aligned} \tag{27}$$

On en déduit $-\alpha_b \omega(\alpha_b^{-1}) = e_b u_b \sigma'(\alpha_b^{-1})$, et on calculera finalement les erreurs par

$$(\forall b \in B) \qquad \boxed{e_b = -\alpha_b \omega(\alpha_b^{-1}) \left(u_b \sigma'(\alpha_b^{-1}) \right)^{-1}}. \tag{28}$$

Question 12: Ecrire une fonction `Erreur` qui, à partir des polynômes localisateur et évaluateur σ et ω, renvoie l'erreur de transmission $\mathbf{e}$. Vérifier sur un exemple que ça fonctionne, à condition que $w(\mathbf{e})$ soit suffisamment petit.

4 Conclusion : une chaîne de transmission cryptée robuste

Question 13: Utiliser vos fonctions RSA du TP précédents pour simuler une chaîne de transmission de message :

1. message alphabétique clair ;
2. message numérique crypté RSA ;
3. message codé GRS envoyé ;
4. message reçu (avec erreurs éventuelles) ;
5. message décodé GRS ;
6. message numérique décrypté ;
7. message alphabétique reconstitué.

Exemple de paramètres : $(q, n, k, d) = (2^8, q - 1, k, n - k + 1)$, faire des essais avec différents $k < n$.

Références

[1] V. Robin. *Les codes correcteurs d'erreurs de Reed et Solomon.* Manuscript UTC, mai 2012.