Lycée Pierre-Gilles de Gennes 2025-2026

TP5 Python

Partage de secret

Ce TP est basé sur le partage des clés secrètes de Shamir, qui permet de crypter un code secret formé de chiffres en distribuant à plusieurs personnes différentes des fragments d'informations permettant, une fois regroupés, de reconstituer le code initial.

1 Le principe du codage de Shamir

On dispose d'un code secret à N chiffres. Cependant, les informations pour le reconstituer on été dispersées en $n \geq 2$ fragments.
Heureusement, la totalité des n fragments n'est pas nécessaire pour reconstituer le code !

Le code secret est un code composé d'une série de chiffres $(0, 1, \dots, 9)$.
Ce code sera représenté par un polynôme P dont :

- le degré est $N-1$ (N est le nombre de caractères du code) ;
- les coefficients sont les caractères du code.

Exemple : le code 007 sera codé par le polynôme $P = 7X^2$.

Les fragments d'information nécessaires au décodage du code sont des couples de réels $((x_k, y_k))_{k\in[\![0,n-1]\!]}$ avec des abscisses deux à deux distinctes tels que

$$P(x_k) = y_k.$$

Reconstituer le code grâce aux fragments consiste alors à retrouver le polynôme P vérifiant :

$$(*) \quad \forall k \in [\![0, n-1]\!], \quad P(x_k) = y_k.$$

Pour être sûr de retrouver le code, il faut donc qu'il y ait un unique polynôme P vérifiant $(*)$: il faut donc que le nombre n de fragments soit supérieur ou égal à la longueur N du code.

Exemple : on considère le code à 3 chiffres (c, b, a) représenté par un polynôme $P = c + bX + aX^2$.

- Si on dispose des deux fragments $(0, 0)$ et $(1, 7)$ nous ne serons pas en mesure de décoder. Par exemple pour $P_1 = 7X^2$ et pour $P_2 = X + 6X^2$ on a :

  $$P_1(0) = P_2(0) = 0 \qquad P_1(1) = P_2(1) = 7$$

  donc on ne peut pas savoir si le code est 007 ou s'il s'agit de 016.
- Si on dispose des trois fragments $(0, 0)$, $(1, 7)$, $(-1, 7)$ nous serons en mesure de décoder car on peut montrer que le seul polynôme vérifiant :

  $$P(0) = 0 \quad ; \quad P(1) = 7 \quad ; \quad P(-1) = 7$$

  est $7X^2$.

2 Les polynômes avec Python

La classe `Polynomial` du module `numpy.polynomial` permet de travailler avec des polynômes.

Travail demandé

1. Importer la fonction `Polynomial` en recopiant :

```
from numpy.polynomial import Polynomial
```

2. Pour créer un polynôme, il suffit de lister ses coefficients par degré croissant. Par exemple, pour le polynôme $P = X^3 + 2X - 3$, on utilise la commande :

```
P = Polynomial([-3,2,0,1])
```

   (a) Définir le polynôme $Q = X^4 - 2X^2 + 1$.
   (b) Tester la commande P.`coef` (et de même Q.`coef`).
   Quelle est son rôle ?
   (c) Déterminer les coefficients des polynômes PQ et $P + Q$.

3. Tester la commande P.`degree()` (et de même Q.`degree()`).
   Quelle est son rôle ?
4. Tester les commandes `Q(1)` , `Q(-1)`.
   Expliquer ces commandes.

3 Les maths au secours du décodage

Soit $n \geq N \geq 2$ et $((x_k, y_k))_{k\in[\![0,n-1]\!]}$ avec les x_k deux à deux distincts .

Décoder un code dont les fragments sont $((x_k, y_k))_{k\in[\![0,n-1]\!]}$ revient donc à trouver l'unique polynôme $P \in \mathbb{R}_{n-1}[X]$ tel que

$$\forall k \in [\![0, n-1]\!], \quad P(x_k) = y_k.$$

Pour tout $k \in [\![0, n-1]\!]$, on pose L_k le polynôme de $\mathbb{R}_{n-1}[X]$ défini par :

$$L_k = \prod_{\substack{j=0 \\ j\neq k}}^{n-1} \frac{X - x_j}{x_k - x_j}.$$

Exemple : si $n = 3$ avec $x_0 = 0$, $x_1 = 1$ et $x_2 = 4$ alors

$$L_0 = \frac{1}{4}(X-1)(X-4) \quad ; L_1 = -\frac{1}{3}X(X-4) \quad ; \quad L_2 = \frac{1}{12}X(X-1).$$

Travail demandé

1. Écrire une fonction `def Poly_elem(a,b)` prenant un entrée deux réels distincts et renvoyant le polynôme $\dfrac{X-b}{a-b}$.
2. Avec la fonction précédente, déterminer la liste des coefficients des polynômes L_0, L_1 et L_2 de l'exemple.
3. (a) Écrire une fonction `Lagrange(X,i)` qui prend en entrée une liste `X`$= [x_0, \dots, x_{n-1}]$, un entier $i \in [\![0, n-1]\!]$ et qui renvoie le polynôme L_i.
   Indications : on pourra utiliser la fonction `Poly_elem(a,b)` et une boucle pour construire L_i par produits successifs.
   (b) Tester la fonction avec la liste $[0, 1, 4]$ et $i \in \{0, 1, 2\}$.
4. (a) Écrire une fonction `def test(X,i)` prenant en entrée une liste `X`$=[x_0, \dots, x_{n-1}]$ et un entier `i`$\in [\![0, n-1]\!]$ et qui renvoie la liste $[L_i(x_0), \dots, L_i(x_{n-1})]$.
   (b) Tester la fonction pour différentes listes X et différentes valeurs de i.
   Que constate-on ?
   (c) Conjecturer pour tout $k, j \in [\![1, n]\!]$ la valeur de $L_k(x_j)$ puis prouver cette conjecture.
5. Dans cette question on prend $n = 2$.
   (a) Écrire une fonction `comb_lin(X,a,b,c)` prenant en argument trois réels a, b, c, une liste $X=[x_0, x_1, x_2]$ de trois éléments distincts, qui définit les polynômes $P = aX^2 + bX + c$ et $P - \displaystyle\sum_{k=0}^{2} P(x_k)L_k$ et qui renvoie la liste des coefficients de ce dernier.
   (b) Tester cette fonction pour $X = [0, 1, 4]$ et différentes valeurs de a, b, c.
   Que remarque-t-on ? Que peut-on en déduire sur la famille L_0, L_1, L_2 de $\mathbb{R}_2[X]$?
6. **Dans un premier temps on pourra admettre les résultats ci-dessous notamment celui de la question 6.(b) et passer à la suite.**
   (a) Déduire de la question 4.(c) que la famille $(L_k)_{k\in[\![0,n-1]\!]}$ est une base de $\mathbb{R}_{n-1}[X]$.
   (b) En déduire qu'il existe un unique polynôme $P \in \mathbb{R}_{n-1}[X]$ tel que :
   $$\forall k \in [\![0, n-1]\!], \quad P(x_k) = y_k$$
   et qu'il est défini par $P = \displaystyle\sum_{k=0}^{n-1} y_k L_k$.
   (c) Que représente le n-uplet $(y_0, \dots, y_{n-1})$ par rapport à P ?

4 Programmation du décodage

Travail demandé

Grâce au résultat de la question 6.(b), écrire une fonction `decode(Fragment)` telle que :

— l'entrée `Fragment` est une liste dont les éléments sont des listes $[x_k, y_k]$ représentants les fragments du code ;

— la sortie est le code (c'est-à-dire la liste des coefficients du polynôme représentant le code).