

## Introduction à la sécurité – TP 0

11/09/2023

Ce TP n'est pas à rendre — il fait office d'entraînement.

Les fichiers utiles sont disponibles à l'adresse suivante :  
<https://lvz1.fr/teaching/2023-24/is.html>

**Exercice 1. Chiffrement affine.**

Dans cet exercice, on souhaite **implanter et attaquer un chiffrement affine simple**. On considère un ensemble de symboles (l'alphabet) formé des 26 lettres courantes en majuscule A-Z, et de trois caractères spéciaux : l'espace « » , l'apostrophe « ' » et le point « . ».

On souhaite encoder ces 29 caractères avec les entiers allant de 0 à 28, en commençant par le A et en terminant par le point.

**Question 1.**– Écrire des fonctions `encode` et `decode`, qui permettent respectivement de coder du texte en une liste d'entiers, et de décoder une liste d'entiers en du texte, selon la règle d'encodage énoncée ci-dessus.

Le chiffrement de Cesar vu en cours s'applique aussi bien sur l'alphabet classique à 26 éléments, que sur notre alphabet à 29 éléments.

**Question 2.**– Écrire des fonctions `chiffre_cesar` et `dechiffre_cesar`, qui permettent respectivement de chiffrer un texte clair et de déchiffrer un texte chiffré avec une clé représentée sous la forme d'un entier compris entre 0 et 28.

Dans notre alphabet à 29 éléments, la distribution des caractères est différente de celle présentée en cours, car on a ajouté trois éléments. Dorénavant, les caractères les plus fréquents sont les suivants :

caractère	espace	E	A	S	I
fréquence	0.165	0.136	0.076	0.069	0.065

**Question 3.**– Écrire une fonction `plus_frequent` qui retourne le caractère le plus fréquent d'un texte.

**Question 4.**– Retrouver le texte clair associé au texte chiffré par un chiffrement de Cesar, qui est donné dans le fichier `enc_cesar.txt`.

On s'intéresse maintenant à une autre type de chiffrement, appelé chiffrement affine. L'idée est de généraliser le chiffrement de Cesar à d'autres types de permutations de lettres. Pour Cesar, la permutation des lettres était donnée par :

$$x \mapsto x + \Delta \pmod{29}$$

où  $x$  est la lettre à chiffrer, et  $\Delta$  est la clé de décalage.

Pour un chiffrement affine, la permutation est de la forme :

$$x \mapsto ax + b \pmod{29}$$

où le couple  $(a, b)$  forme la clé, avec comme seule contrainte  $a \neq 0$ . Pour déchiffrer, on doit alors appliquer la permutation inverse, à savoir :

$$y \mapsto (y - b) \cdot a^{-1} \pmod{29}$$

**Remarque.** Pour obtenir l'inverse modulaire «  $a^{-1} \pmod{29}$  », en python il suffit d'exécuter `pow(a, -1, 29)`.

**Question 5.**– Écrire des fonctions `chiffre_affine` et `dechiffre_affine`, qui permettent respectivement de chiffrer un texte clair et de déchiffrer un texte chiffré avec une clé `cle` représentée sous la forme d'un couple d'entiers.

**Question 6.**– Vérifier que le chiffré de INFORMATIQUE par la clé  $(a = 13, b = 12)$  est AHTUBXM'ARLG, puis que le déchiffrement se déroule correctement.

Pour retrouver la clé  $\Delta$  du chiffrement de Cesar, il a suffi de reconnaître le chiffré d'une seule lettre (en l'occurrence, la plus fréquente). Pour retrouver la clé du chiffrement affine, qui est formée de deux entiers  $(a, b)$ , il nous faudra la connaissance du chiffrement de deux lettres.

Supposons que l'on connaisse  $y$  le chiffré d'un caractère  $x$ , et  $y'$  le chiffré de  $x'$ . Alors on a le système de deux équations à deux inconnues  $(a, b)$  :

$$\begin{cases} y = ax + b \pmod{29} \\ y' = ax' + b \pmod{29} \end{cases}$$

que l'on sait résoudre pour obtenir

$$a = (y - y') \cdot (x - x')^{-1} \pmod{29} \quad \text{puis} \quad b = y - ax \pmod{29}.$$

**Question 7.**– Écrire une fonction `deux_plus_frequents` qui retourne les deux caractères les plus fréquents d'un texte.

**Question 8.**– Retrouver le texte clair associé au texte chiffré par un chiffrement affine, qui est donné dans le fichier `enc_affine.txt` du dossier `affine`.

## Exercice 2. Attaque sur le chiffrement de Vigenère.

On souhaite attaquer le chiffrement de Vigenère vu en cours, c'est-à-dire avec un alphabet de 26 lettres.

**Question 1.**– Implanter les fonctions de chiffrement et de déchiffrement de Vigenère, puis tester ces fonctions sur un exemple.

**Question 2.**– Implanter une fonction `IC` qui calcule l'indice de coïncidence d'un texte.

**Question 3.**– Implanter une fonction `calcule_longueur_cle` qui prend en entrée un texte chiffré, et qui retourne la longueur de clé probable utilisée pour obtenir ce chiffré.

**Question 4.**– Implanter une fonction `calcule_decalage` qui calcule le décalage probable d'un morceau de texte chiffré avec Cesar.

**Question 5.**– Implanter une fonction `attaque_vigenere` qui décrypte un chiffré de Vigenère (sans la clé, donc).

**Question 6.**– Décrypter les chiffrés contenus dans les fichiers `file<i>_encrypted.txt` du dossier `vigenere`.