

# TD : Méthodes de chiffrement

---

## I. Chiffrement par décalage

Dans cette première partie, vous allez implémenter une méthode basique de chiffrement : le chiffrement par décalage.

Pour des raisons de simplicité et d'efficacité, on supposera que toutes les chaînes de caractères manipulées ne sont composées que des symboles suivants :

- Lettres majuscules non accentuées
- Chiffres
- Symbole espace
- Signes de ponctuation

**Question 1.** Imaginer une fonction `chiffre_de_cesar_char` qui prend en argument une chaîne de caractère `c` de longueur 1 et qui renvoie

- le caractère situé 3 lettres plus loin dans l'alphabet que le caractère `c` si `c` est une lettre
- le caractère `c` si `c` n'est pas une lettre

**Indice :** N'oubliez pas que la lettre `Y` devient la lettre `B`.

**Question 2.** En déduire une fonction `chiffre_de_cesar` telle que `chiffre_de_cesar(texte)` renvoie une version chiffrée grâce au chiffre de César de la chaîne de caractère `texte`.

## II. Méthode du masque jetable

Dans cette deuxième partie, vous allez implémenter la méthode du masque jetable.

Vous manipulerez donc des messages binaires, représentés en Python par des listes d'entiers dont on suppose qu'ils ne contiennent que des 0 et des 1.

**Question 3.** Imaginer une fonction `creer_masque_aleatoire` telle que `creer_masque_aleatoire(longueur)` renvoie une liste de longueur `longueur` et composée d'une suite aléatoire de 0 et de 1.

**Question 4.** Ecrire une fonction `utiliser_masque` telle que `utiliser_masque(message, masque)` applique le masque décrit par la liste `masque` sur le message décrit par la liste `message` et renvoie la liste correspondante.

**Remarques :** Cette fonction devrait vous permettre à la fois de chiffrer et de déchiffrer un message grâce à cette méthode.

### III. Méthode à clé publique – clé privée

Pour finir, vous allez implémenter la méthode RSA.

Les messages transférés seront donc de simples entiers, mais comme nous l'avons vu en cours, tout type d'information peut être codé par une suite de bits, et à fortiori par une suite d'entiers.

#### a. Diviseurs et nombres premiers

**Question 5.** Ecrire une fonction `est_diviseur`, telle que `est_diviseur(a,b)` renvoie `True` si `a` est un diviseur de `b`, et `False` sinon.

**Question 6.** Ecrire une fonction `liste_diviseurs`, telle que `liste_diviseurs(a)` renvoie la liste des diviseurs de l'entier `a`.

**Question 7.** Ecrire une fonction `est_premier`, telle que `est_premier(n)` renvoie `True` si le nombre `n` est premier, et `False` sinon.

**Rappel :** Un nombre premier est un entier naturel qui admet exactement deux diviseurs entiers distincts : 1 et lui-même.

**Question 8.** Ecrire une fonction `sont_premiers_entre_eux`, telle que `sont_premiers_entre_eux(n,p)` renvoie `True` si les nombres `n` et `p` sont premiers entre eux, et `False` sinon.

**Rappel :** Deux nombres sont premiers entre eux s'ils admettent exactement un diviseur entier commun : 1.

**Question 9.** En utilisant les fonctions définies ci-dessus,

- Choisir deux nombres premiers distincts  $p$  et  $q$
- Calculer leur produit  $n = pq$
- Calculer  $\varphi(n) = (p - 1)(q - 1)$
- Choisir un entier  $e$  premier avec  $(n)$  et strictement inférieur à  $(n)$

### b. Inverse modulo

On dit que  $b$  est l'inverse de  $a$  modulo  $n$  si et seulement si  $a \times b = 1 \pmod n$

En d'autres termes,  $b$  est l'inverse de  $a$  modulo  $n$  si et seulement si il existe un nombre entier relatif  $k$  tel quel  $a \times b + k \times n = 1$

**Remarque :** Ce entier  $b$  n'existe pas toujours, mais s'il existe, il peut être choisi entre  $0$  et  $n - 1$ .

**Question 10.** Ecrire une fonction `trouver_inverse_modulo`, telle que `trouver_inverse_modulo a n` renvoie, s'il existe, l'inverse de  $a$  modulo  $n$ .

**Remarque :** Si ce nombre n'existe pas, utilisez l'instruction `raise Exception("message")` pour le signaler.

**Question 11.** Déduire des questions précédentes

- Un entier  $d$ , inverse de  $e$  modulo  $\varphi(n)$  et strictement inférieur à  $\varphi(n)$
- Une clé publique  $(n, e)$
- Une clé privée  $(n, d)$

### c. Exponentiation modulaire

**Question 12.** Imaginer une fonction `exponentiation_modulaire`, telle que `exponentiation_modulaire(a, k, n)` calcule la valeur de  $a^k$  modulo  $n$

**Remarque :**  $a^k \pmod n = a \times (a^{k-1} \pmod n) \pmod n$

**Question 13.** Grâce à la fonction de la question précédente,

- Choisir un message  $M$  à chiffrer (c'est-à-dire un entier strictement inférieur à  $n$ )
- Chiffrer ce message grâce à la clé publique :  $C = M^e \pmod n$
- Déchiffrer ce message grâce à la clé privée :  $R = C^d \pmod n$
- Vérifier que  $R = M$