

TP chiffrement polyalphabétique

Christophe Viroulaud

Terminale - NSI

Algo 21

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Un chiffrement polyalphabétique consiste à :

- ▶ utiliser une clé de chiffrement composée de plusieurs lettres,
- ▶ recopier la clé de façon à obtenir une chaîne de la longueur du message.

B	R	A	V	O
N	S	I	N	S

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Construire un algorithme de chiffrement
polyalphabétique.

Remarque

Dans un souci de simplification, nous nous limiterons à des lettres majuscules. Chaque caractère sera donc :

- ▶ équivalent à une valeur ASCII comprise entre 65 (A) et 90 (Z),
- ▶ codé sur 7 bits.

1. Conversions

2. Convertir la clé

3. Appliquer le chiffrement XOR

4. Chiffrement, déchiffrement

5. Opérateurs binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

La première étape consiste à convertir une chaîne de caractère en sa correspondance binaire.

B	R	A	V	O
66	82	65	86	79
1000010	1010010	1000001	1010110	1001111
N	S	I	N	S
78	83	73	78	83
1001110	1010011	1001001	1001110	1010011

Tableau 1 – Conversion en binaire

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Activité 1 :

1. Télécharger et extraire le dossier compressé **TP-annexe-chiffrement.zip** sur le site <https://cviroulaud.github.io>
2. Écrire la fonction `renverser(tab: list) → list` qui renverse le contenu de `tab`
3. Écrire la fonction `int_en_bin(nb: int) → list` qui renvoie le tableau des bits de l'entier `nb`
4. Écrire la fonction `bin_en_int(paquet: list) → int` qui renvoie l'entier correspondant à la représentation binaire `paquet`.
5. Exécuter le fichier `test_conversion.py` pour tester les trois fonctions.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def renverser(tab: list) -> list:
2     """
3     renverse un tableau
4     """
5     l = len(tab)
6     res = [0 for _ in range(l)]
7     for i in range(l):
8         res[l-1-i] = tab[i]
9     return res
```

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def int_en_bin(nb: int) -> list:
2     """
3     Convertit un entier en sa représentation binaire
4     """
5     q = nb
6     r = []
7     while q > 0:
8         r.append(q % 2)
9         q = q//2
10
11     return renverser(r)
```

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def bin_en_int(paquet: list) -> int:
2     """
3     Convertit un paquet de bits en entier
4     """
5     entier = 0
6     for i in range(len(paquet)):
7         entier += int(paquet[i])*2**((len(paquet)-1-i)
8     return entier
```

1. Conversions
2. Convertir la clé
3. Appliquer le chiffrement XOR
4. Chiffrement, déchiffrement
5. Opérateurs binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Afin de pouvoir chiffrer le message lettre par lettre, il faut allonger la clé de la taille du message :

B	R	A	V	O
N	S	I	N	S

Activité 2 : Écrire la fonctionne

`creer_cle_bin(cle: str, taille: int) → list`
qui construit dans un tableau la version binaire de la
clé, à la `taille` du message.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

N	S	I	N	S
78	83	73	78	83
1001110	1010011	1001001	1001110	1010011

```

1 >>> creer_cle_bin("NSI", 5)
2 [[1, 0, 0, 1, 1, 1, 0],
3  [1, 0, 1, 0, 0, 1, 1],
4  [1, 0, 0, 1, 0, 0, 1],
5  [1, 0, 0, 1, 1, 1, 0],
6  [1, 0, 1, 0, 0, 1, 1]]

```

Code 1 – Conversion en binaire

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def creer_cle_bin(cle: str, taille: int) -> list:
2     """
3     crée la version binaire de la clé, de la taille
4     du message à chiffrer
5     """
6     cle_bin = []
7     for i in range(taille):
8         # si dépasse taille de la clé, revient à 0
9         lettre = cle[i % len(cle)]
10        cle_bin.append(int_en_bin(ord(lettre)))
11    return cle_bin
```


1. Conversions
2. Convertir la clé
3. Appliquer le chiffrement XOR
4. Chiffrement, déchiffrement
5. Opérateurs binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

La porte `xor` n'existe pas par défaut en Python. Il faut donc la créer.

Activité 3 :

1. Écrire la fonction `xor(x: int, y: int) → int`
2. Écrire alors la table de vérité de la porte logique XOR.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def xor(x: int, y: int) -> int:
2     if x == 0:
3         if y == 0:
4             return 0
5         else:
6             return 1
7     else:
8         if y == 0:
9             return 1
10        else:
11            return 0
```

```
1 print(f"0 xor 0 -> {xor(0, 0)}")
2 print(f"0 xor 1 -> {xor(0, 1)}")
3 print(f"1 xor 0 -> {xor(1, 0)}")
4 print(f"1 xor 1 -> {xor(1, 1)}")
```

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

La fonction créée permet de chiffrer la version binaire du message.

Message	1000010	1010010	1000001	1010110	1001111
\oplus	1001110	1010011	1001001	1001110	1010011
Chiffré	0001100	0000001	0001000	0011000	0011100

Tableau 2 – Application de la porte XOR

Activité 4 : Écrire la fonction
`appliquer_xor(entree_bin: list, cle_bin:
list) → list` qui renvoie un tableau de tableau
représentant la version binaire chiffrée du message.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def appliquer_xor(entree_bin: list, cle_bin: list) -> list:
2     # préparation du tableau de tableau de sortie
3     sortie_bin = [[0 for _ in range(len(entree_bin[0]))]
4                   for _ in range(len(entree_bin))]
5
6     # application de xor sur chaque bit
7     for i in range(len(entree_bin)):
8         for j in range(len(entree_bin[0])):
9             sortie_bin[i][j] = xor(entree_bin[i][j],
10                                    cle_bin[i][j])
11
12     return sortie_bin
```


1. Conversions
2. Convertir la clé
3. Appliquer le chiffrement XOR
4. Chiffrement, déchiffrement
5. Opérateurs binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

L'algorithme de chiffrement s'écrit alors :

- ▶ Convertir le message en binaire.
- ▶ Convertir la clé en binaire.
- ▶ Chiffrer le message en binaire.

Remarque

Il est inutile de convertir le message en caractères. C'est la version binaire qui transite jusqu'au destinataire.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Activité 5 : Écrire la fonction
`chiffrement(message: str, cle: str) → list`
qui applique l'algorithme précédent.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def chiffrement(message: str, cle: str) -> list:
2     # convertit le message en binaire
3     message_bin = []
4     for lettre in message:
5         message_bin.append(int_en_bin(ord(lettre)))
6
7     # convertit la clé en binaire
8     cle_bin = creer_cle_bin(cle, len(message))
9
10    # chiffre le message
11    secret_bin = appliquer_xor(message_bin, cle_bin)
12    return secret_bin
```

Pour le déchiffrement, le destinataire applique l'algorithme suivant :

- ▶ Convertir la clé en binaire.
- ▶ Déchiffrer le message en binaire.
- ▶ Convertir le message en caractères.

Activité 6 : Écrire la fonction
`dechiffrement(secret_bin: list, cle: str) → str` qui applique l'algorithme précédent.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def dechiffrement(secret_bin: list, cle: str) -> str:
2     # convertit la clé en binaire
3     cle_bin = creer_cle_bin(cle, len(secret_bin))
4
5     # déchiffre le secret binaire
6     message_bin = appliquer_xor(secret_bin, cle_bin)
7
8     # convertit le message binaire en caractères
9     message = ""
10    for lettre_bin in message_bin:
11        message = message+chr(bin_en_int(lettre_bin))
12    return message
```

```
1 cle = "NSI"
2 s = chiffrement("BRAVO", cle)
3 print(dechiffrement(s, cle))
```


1. Conversions
2. Convertir la clé
3. Appliquer le chiffrement XOR
4. Chiffrement, déchiffrement
5. Opérateurs binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

En Python, il est possible d'effectuer des opérations directement sur les bits des entiers.

$$\begin{array}{rcll} 10_{10} & = & 1 & 0 & 1 & 0 \\ \& \quad 8_{10} & = & 1 & 0 & 0 & 0 \\ \hline 8_{10} & = & 1 & 0 & 0 & 0 \end{array}$$

```
1 >>> 10 & 8
2 8
```

Code 2 – AND binaire

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

$$\begin{array}{rcccc} 10_{10} = & 1 & 0 & 1 & 0 \\ | & 8_{10} = & 1 & 0 & 0 & 0 \\ \hline 10_{10} = & 1 & 0 & 1 & 0 \end{array}$$

```
1 >>> 10 | 8
2 10
```

Code 3 – OR binaire

$$\begin{array}{rcccc} 10_{10} = & 1 & 0 & 1 & 0 \\ ^8_{10} = & 1 & 0 & 0 & 0 \\ \hline 2_{10} = & 0 & 0 & 1 & 0 \end{array}$$

```
1 >>> 10 ^ 8
2 2
```

Code 4 – XOR binaire

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

B	R	A	V	O
66	82	65	86	79
N	S	I	N	S
78	83	73	78	83

Tableau 3 – Conversion en ASCII

Activité 7 : Écrire la fonction `str_en_int(texte: str, taille: int) → list` qui renvoie le tableau des valeurs ASCII d'un `texte`. La `taille` correspond à la longueur de la chaîne `texte`.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

```
1 def str_en_int(texte: str, taille: int) -> list:
2     """
3     renvoie le tableau des valeurs ASCII d'un texte
4     """
5     texte_ascii = [0 for _ in range(taille)]
6     for i in range(taille):
7         texte_ascii[i] = ord(texte[i % len(texte)])
8     return texte_ascii
```

```
1 >>> str_en_int("BRAVO",5)
2 [66, 82, 65, 86, 79]
3 >>> str_en_int("NSI", 5)
4 [78, 83, 73, 78, 83]
```

L'algorithme de chiffrement s'écrit alors :

- ▶ Convertir le message en entiers ASCII.
- ▶ Convertir la clé en entiers ASCII, de la même taille que le message.
- ▶ Chiffrer les entiers ASCII en appliquant un **OU EXCLUSIF** binaire.

Activité 8 : Écrire la fonction
`chiffrement(message: str, cle: str) → list`
 qui applique l'algorithme précédent.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires


```
1 def chiffrement(message: str, cle: str) -> list:
2     # convertit le message en valeurs sASCII
3     message_ascii = str_en_int(message, len(message))
4
5     # convertit la clé en valeurs ASCII
6     cle_ascii = str_en_int(cle, len(message))
7
8     # chiffre les valeurs ASCII
9     secret_ascii = [0 for _ in range(len(message))]
10    for i in range(len(message_ascii)):
11        # opérateur binaire xor
12        secret_ascii[i] = message_ascii[i] ^ cle_ascii[i]
13    return secret_ascii
```

Activité 9 :

1. Écrire l'algorithme de déchiffrement.
2. Écrire la fonction
`dechiffrement(secret_ascii: list, cle:
str) → str` qui applique l'algorithme précédent.

Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ▶ Analyser les messages d'erreur,
- ▶ Demander au professeur.

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

Conversions

Convertir la clé

Appliquer le
chiffrement XOR

Chiffrement,
déchiffrement

Opérateurs
binaires

- ▶ Convertir la clé en entier ASCII, de la taille du message.
- ▶ Déchiffrer les entiers ASCII du message secret.
- ▶ Convertir les entiers ASCII en lettres

```

1 def dechiffrement(secret_ascii: list, cle: str) -> str:
2     # convertit la clé en valeurs ASCII
3     cle_ascii = str_en_int(cle, len(secret_ascii))
4
5     # déchiffre les valeurs ASCII
6     message_ascii = [0 for _ in range(len(secret_ascii))]
7     for i in range(len(secret_ascii)):
8         # opérateur binaire xor
9         message_ascii[i] = secret_ascii[i] ^ cle_ascii[i]
10
11     # convertit les valeurs ASCII en lettre
12     message = ""
13     for lettre_ascii in message_ascii:
14         message = message+chr(lettre_ascii)
15     return message

```

```

1 cle = "NSI"
2 s = chiffrement("BRAVO", cle)
3 print(dechiffrement(s, cle))

```