Architecture et réseaux

Chiffrement XOR

TD n° 1

Contexte

Message intercepté

Un agent du réseau lycée a capturé un message chiffré. Après analyse, il s'avère que l'expéditeur utilise le **chiffre XOR** : un algorithme de chiffrement symétrique où le même masque (la clé) sert à la fois à chiffrer et à déchiffrer.

Votre mission est de comprendre et d'implémenter cet algorithme pas à pas, puis de **déchiffrer le message intercepté** pour identifier de qui il parle.

Partie 1 — XOR bit à bit (sans ordinateur)

Rappel : l'opérateur XOR ($\oplus$)

Le XOR (ou exclusif) vaut **1** uniquement si les deux bits sont **différents** :

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

On applique cette règle **colonne par colonne** entre deux entiers binaires.
En Python, l'opérateur XOR bit à bit s'écrit `^` : `10 ^ 12` → `6`

Question 1 — Calculer 1010 XOR 1100

Compléter le tableau en appliquant XOR colonne par colonne :

	bit 3	bit 2	bit 1	bit 0
10	1	0	1	0
12	1	1	0	0
10 ⊕ 12				

Résultat en binaire : .....

Résultat en décimal : .....

Information

En Python, l'opérateur XOR bit à bit s'écrit `^` : `10 ^ 12`

Partie 2 — Chiffrer un caractère

Principe du chiffrement XOR sur les lettres

Pour chiffrer une lettre `c` avec une clé `k`, on applique XOR sur leurs valeurs numériques recentrées (A = 1, B = 2, ..., Z = 26) :

1. Convertir : `ord(c) - 64`
2. Convertir la clé : `ord(k) - 64`
3. Appliquer XOR entre les deux valeurs
4. Reconvertir : ajouter 64 puis `chr(...)`

Exemple : chiffrer `'B'` avec la clé `'L'`

Lettre	ord()	− 64	XOR	+ 64	chr()
B	66	2	$2 \oplus 12 = 14$	78	**N**
L	76	12			

Question 2 — Chiffrement manuel de `'A'` avec la clé `'C'`

Compléter le tableau de chiffrement (A = 1, C = 3) :

Lettre	ord()	− 64	XOR	+ 64	chr()
A	_____	____	_____	_____	____
C	_____	____			

Question 3 — Fonction `chiffrer_car(c, k)`

Écrire en Python la fonction **`chiffrer_car(c, k)`** qui chiffre le caractère **`c`** avec la clé **`k`** selon la méthode ci-dessus.

Exemples

```
>>> chiffrer_car('B', 'L')
'N'
>>> chiffrer_car('A', 'C')
'B'
>>> chiffrer_car('M', 'E')
'H'
```

Réponse

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Partie 3 — Chiffrer un message

Chiffrement lettre par lettre

Pour chiffrer un message, on applique `chiffrer_car` à chaque position. Si la clé est plus courte que le message, elle **se répète** en utilisant l'opération modulo : `masque[i % len(masque)]`.

Exemple : chiffrer `"AIDE"` avec le masque `"BCHF"`

i	msg[i]	masque[i]	msg−64	clé−64	XOR	+64	résultat
0	A (65)	B (66)	1	2	3	67	**C**
1	I (73)	C (67)	9	3	10	74	**J**
2	D (68)	H (72)	4	8	12	76	**L**
3	E (69)	F (70)	5	6	3	67	**C**

`"AIDE"` chiffré avec `"BCHF"` → `"CJLC"`

Question 4 — Fonction `chiffrement(message, masque)`

Écrire la fonction `chiffrement(message, masque)` qui chiffre le message lettre par lettre en utilisant `chiffrer_car` et le masque cyclique.

Exemples

```
>>> chiffrement("AIDE", "BCHF")
'CJLC'
>>> chiffrement("BEA", "LDC")
'NAB'
>>> chiffrement("BONJOUR", "LPOSADA")
'N_AYNQS'
```

Réponse

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Partie 4 — Déchiffrer

La propriété magique du XOR

Pour tout entier a et tout entier b :

$$(a \oplus b) \oplus b = a$$

Cela se vérifie bit par bit sur toutes les lignes de la table de vérité.

Conséquence pour le chiffrement : appliquer deux fois `chiffrement` avec la même clé redonne le message d'origine. **Déchiffrer = chiffrer** (avec la même clé).

Exemple :

```
>>> chiffrement(chiffrement("BONJOUR", "LPOSADA"), "LPOSADA")
'BONJOUR'
```

Question 5 — Prédire le résultat (sans ordinateur)

Sans exécuter de code, que va renvoyer l'expression suivante ?

```
chiffrement(chiffrement("AIDE", "BCHF"), "BCHF")
```

Réponse : .....

Justification : .....

Question 6 — Fonction `dechiffrement(message, masque)`

Écrire la fonction **`dechiffrement(message, masque)`** qui déchiffre un message chiffré par XOR.

Indice : d'après la propriété ci-dessus, cette fonction est très courte…

Réponse

.....

.....

.....

.....

Question 7 — Déchiffrer le message intercepté

Le texte suivant a été chiffré avec la clé `"BOOLE"`. Utiliser `dechiffrement` pour le déchiffrer et identifier **de qui parle ce texte**.

```
texte_chiffre = "CZHYVVKJAJPHNBCM]BYIGCJ_LN@F_TWF_CWVJAXVMAACH"
cle = "BOOLE"
```

Écrire l'appel permettant de déchiffrer le texte :

Réponse

.....

.....

.....

.....

.....

Question 8 — Sur qui porte ce texte ?

.....

.....