Chiffre de Vernam

Question 1 : Chiffrez le message M = CRYPTO avec la clé K = VSDQLK

Calcul : C = M ⊕ K = (2+21, 17+18, 24+3, 15+16, 19+11, 14+10) mod 26
Résultat : C = XJDFLQ

Voir code Python ci-dessous.

Question 2 : Opération de déchiffrement et type de clé

Déchiffrement : M = C ⊖ K, avec mi = (ci - ki) mod 26

Type : Chiffrement à **clé secrète** (symétrique) - même clé pour chiffrer et déchiffrer.

Question 3 : Déchiffrez C = DSVSWA avec K = LOTBSH

Résultat : M = CRYPTO

Voir code Python ci-dessous.

Question 4 : Déchiffrez C = DSVSWA avec K' = OYUHOY

Résultat : M = ATTACK

Un même message chiffré peut donner différents messages plausibles selon la clé utilisée. C'est ce qui garantit la sécurité parfaite du masque jetable.

```
import string

# Définition du codage des lettres
letter_to_code = {char: i for i, char in enumerate(string.ascii_uppercase)}
code_to_letter = {i: char for i, char in enumerate(string.ascii_uppercase)}

def encrypt_vernam(message, key):
    """Chiffre un message avec le chiffre de Vernam"""
    encrypted_message = ""
    for i in range(len(message)):
        m_code = letter_to_code[message[i]]
        k_code = letter_to_code[key[i]]
        c_code = (m_code + k_code) % 26
        encrypted_message += code_to_letter[c_code]
    return encrypted_message

def decrypt_vernam(ciphertext, key):
    """Déchiffre un message avec le chiffre de Vernam"""
    decrypted_message = ""
    for i in range(len(ciphertext)):
        c_code = letter_to_code[ciphertext[i]]
        k_code = letter_to_code[key[i]]
        m_code = (c_code - k_code) % 26
        decrypted_message += code_to_letter[m_code]
    return decrypted_message

print("=== Résultats des questions ===")
print()

# Question 1 : Chiffrement CRYPTO avec VSDQLK
M = "CRYPTO"
K = "VSDQLK"
C = encrypt_vernam(M, K)
print(f"Question 1: {M} + {K} = {C}")

# Question 3 : Déchiffrement DSVSWA avec LOTBSH
C_q3 = "DSVSWA"
K_q3 = "LOTBSH"
M_q3 = decrypt_vernam(C_q3, K_q3)
print(f"Question 3: {C_q3} - {K_q3} = {M_q3}")

# Question 4 : Déchiffrement DSVSWA avec OYUHOY
K_prime_q4 = "OYUHOY"
```

```
M_q4 = decrypt_vernam(C_q3, K_prime_q4)
print(f"Question 4: {C_q3} - {K_prime_q4} = {M_q4}")
```

```
=== Résultats des questions ===

Question 1: CRYPTO + VSDQLK = XJBFEY
Question 3: DSVSWA - LOTBSH = SECRET
Question 4: DSVSWA - OYUHOY = PUBLIC
```

Question 5 : Existence d'une clé et probabilité pour un attaquant

Existence : Pour tout couple (C, M) de même longueur, il existe toujours une clé K telle que C = E_K(M). Formule : ki = (ci - mi) mod 26

Probabilité : Un attaquant sans la clé a une probabilité de 1/26^l de retrouver le bon message (l = longueur). Le chiffre de Vernam offre donc une **sécurité parfaite**.

Question 6 : Récupération d'un message avec la même clé

Problème : Si deux messages M1 et M2 sont chiffrés avec la même clé K et qu'Eve connaît M1 :

1. K = C1 ⊖ M1 (récupération de la clé)
2. M2 = C2 ⊖ K (déchiffrement du second message)

Conclusion : D'où le nom "masque **jetable**" - une clé ne doit servir qu'une seule fois.

Question 7 : Distribution et gestion de la clé

Qui doit posséder la clé : Émetteur ET destinataire doivent posséder la même clé secrète.

Problèmes pratiques :

- Distribution sécurisée de la clé (canal séparé et sûr)
- Taille = longueur du message (ingérable pour longs messages)
- Génération, stockage et destruction de grandes quantités de clés

Utilisation : Réservé aux communications très sensibles et courtes où la sécurité absolue prime.