

▼ Chiffrement de Wolseley

Question 1 : Chiffrer "PAS TROP COMPLIQUE" avec la clé "SILENCE JE REFLECHIS"

Construction de la grille 5×5 :

```
S I L E N  
C J R F H  
U T O A D  
B K V M G  
P Q X Y Z
```

Principe : Chaque lettre est remplacée par sa symétrique par rapport au centre.

Voir code Python ci-dessous.

Question 2 : Déchiffrer "EYEV RFXICVTM"

Propriété involutive : Le chiffrement de Wolseley est involutif, donc le déchiffrement utilise la même opération.

Voir code Python ci-dessous.

Question 3 : Chiffrer le message trouvé avec "SILENCE TU REFLECHIS"

Nouvelle grille 5×5 : Construction avec la nouvelle phrase clé.

Voir code Python ci-dessous.

▼ Question 4 : Calcul du rapport de changement

Analyse : Comparaison entre le nombre de symboles modifiés dans la clé et dans le message chiffré.

Voir calculs ci-dessous.

```
import string
```

```

class WolseleyCipher:
    def __init__(self, key_phrase):
        self.grid = self.create_grid(key_phrase)
        self.substitution_map = self.create_substitution_map()

    def create_grid(self, key_phrase):
        """Crée la grille 5x5 à partir de la phrase clé"""
        # Supprimer les espaces et convertir en majuscules
        key = key_phrase.replace(" ", "").upper()

        # Éliminer les doublons tout en préservant l'ordre
        unique_key = ""
        for char in key:
            if char not in unique_key and char.isalpha():
                unique_key += char

        # Compléter avec le reste de l'alphabet (sans W)
        alphabet = string.ascii_uppercase.replace('W', '')
        for char in alphabet:
            if char not in unique_key:
                unique_key += char

        # Créer la grille 5x5
        grid = []
        for i in range(5):
            row = []
            for j in range(5):
                row.append(unique_key[i*5 + j])
            grid.append(row)

        return grid

    def create_substitution_map(self):
        """Crée le dictionnaire de substitution basé sur la symétrie centrale"""
        substitution = {}

        for i in range(5):
            for j in range(5):
                # Position symétrique par rapport au centre (2,2)
                sym_i = 4 - i
                sym_j = 4 - j

                original = self.grid[i][j]
                symmetric = self.grid[sym_i][sym_j]

                substitution[original] = symmetric

        return substitution

    def print_grid(self):
        """Affiche la grille"""
        for row in self.grid:
            print(" ".join(row))

    def encrypt_decrypt(self, text):

```

```
"""Chiffre ou déchiffre le texte (opération involutive)"""
result = ""
for char in text.upper():
    if char == ' ':
        result += ' '
    elif char == 'W':
        # Traitement spécial pour W (remplacé par V)
        result += self.substitution_map.get('V', 'V')
    elif char in self.substitution_map:
        result += self.substitution_map[char]
    else:
        result += char
return result

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

# Question 1
print("Question 1:")
cipher1 = WolseleyCipher("SILENCE JE REFLECHIS")
print("Grille avec clé 'SILENCE JE REFLECHIS':")
cipher1.print_grid()
print()

message1 = "PAS TROP COMPLIQUE"
encrypted1 = cipher1.encrypt_decrypt(message1)
print(f"Message clair: {message1}")
print(f"Message chiffré: {encrypted1}")
print()

# Question 2
print("Question 2:")
encrypted2 = "EYEV RFXICVTM"
decrypted2 = cipher1.encrypt_decrypt(encrypted2)
print(f"Message chiffré: {encrypted2}")
print(f"Message déchiffré: {decrypted2}")
print()

# Question 3
print("Question 3:")
cipher2 = WolseleyCipher("SILENCE TU REFLECHIS")
print("Grille avec clé 'SILENCE TU REFLECHIS':")
cipher2.print_grid()
print()

encrypted3 = cipher2.encrypt_decrypt(decrypted2)
print(f"Message clair: {decrypted2}")
print(f"Message chiffré avec nouvelle clé: {encrypted3}")
print()

# Question 4
print("Question 4:")
key1 = "SILENCE JE REFLECHIS".replace(" ", "")
key2 = "SILENCE TU REFLECHIS".replace(" ", "")
```

```

# Compter les changements dans la clé
key_changes = sum(1 for i in range(min(len(key1), len(key2))) if key1[i] != key2[i])
if len(key1) != len(key2):
    key_changes += abs(len(key1) - len(key2))

# Compter les changements dans le message chiffré
msg_changes = sum(1 for i in range(min(len(encrypted1), len(encrypted3))) if encrypted1[i] != encrypted3[i])
if len(encrypted1) != len(encrypted3):
    msg_changes += abs(len(encrypted1) - len(encrypted3))

print(f"Clé 1: {key1}")
print(f"Clé 2: {key2}")
print(f"Changements dans la clé: {key_changes} caractères")
print(f"Message 1: {encrypted1}")
print(f"Message 2: {encrypted3}")
print(f"Changements dans le message chiffré: {msg_changes} caractères")
if msg_changes > 0:
    print(f"Rapport: {key_changes}/{msg_changes} = {key_changes/msg_changes:.2f}")
else:
    print("Aucun changement dans le message chiffré")

```

==== Résultats des questions ===

Question 1:

Grille avec clé 'SILENCE JE REFLECHIS':

S I L E N

C J R F H

A B D G K

M O P Q T

U V X Y Z

Message clair: PAS TROP COMPLIQUE

Message chiffré: RKZ CPFR TFHRXYJNV

Question 2:

Message chiffré: EYEV RFXICVTM

Message déchiffré: VIVE POLYTECH

Question 3:

Grille avec clé 'SILENCE TU REFLECHIS':

S I L E N

C T U R F

H A B D G

J K M O P

Q V X Y Z

Message clair: VIVE POLYTECH

Message chiffré avec nouvelle clé: EYEV CTXIOVPG

Question 4:

Clé 1: SILENCEJEREFLECHIS

Clé 2: SILENCETUREFLECHIS

Changements dans la clé: 2 caractères

Message 1: RKZ CPFR TFHRXYJNV

Message 2: EYEV CTXIOVPG

Changements dans le message chiffré: 18 caractères

Rapport: 2/18 = 0.11

Analyse du chiffrement de Wolseley

Propriétés importantes :

1. **Inolutif** : $E(E(M)) = M$ (chiffrement = déchiffrement)
2. **Substitution monoalphabétique** : Chaque lettre est toujours remplacée par la même
3. **Symétrie centrale** : Basé sur la symétrie par rapport au centre de la grille 5×5

Sécurité : Plus faible que Vernam car substitution fixe (analyse fréquentielle possible).

Avantages et inconvénients

Avantages :

- Simplicité d'utilisation
- Pas besoin de clé aussi longue que le message
- Propriété involutive pratique (même opération pour chiffrer et déchiffrer)

Inconvénients :

- Vulnérable à l'analyse fréquentielle
- Substitution monoalphabétique
- Perte du caractère W (remplacé par V)

Conclusion : Le chiffrement de Wolseley est historiquement intéressant mais insuffisant pour les standards de sécurité modernes.

