

**TP DE CRYPTOGRAPHIE INFORMATIQUE 2**

**THEME :**

Implémentation du cryptage par transposition et sa cryptanalyse

Groupe 2 :

|  |  |
| --- | --- |
| Noms et prénoms | matricules |
| TEUFACK KRISAN BAOROS | CM-UDS-23SCI0710 |
| KELEKO SABZE YANNICK | CM-UDS-23SCI0568 |
| FOTSING TCHEDJOU ALAIN | CM-UDS-23SCI0538 |
| KENNE DASSE JOVIAL | CM-UDS-23SCI1109 |
| KAMGA KAMENI PIERRE JUNIOR | CM-UDS-23SCI0108 |
| DJIOTSA NANGUE RAHIM |  |

Sous la supervision de : Dr. FOKOU PELAP Géraud

Année académique 2024/2025

# Introduction

Le chiffrement par transposition est l’une des cryptographies anciennes qui consiste à modifier les positions des lettres d’un texte en clair selon une règle prédéfinie.

Son principe est basé sur un mot définie comme clé.

Objectifs :

* Implémenter le chiffrement par transposition.
* Réaliser le déchiffrement
* Analyser les vulnérabilités à l'aide de la cryptanalyse.
* Tutoriel d’applications

# Méthodologie

## Chiffrement par transposition

Le chiffrement par transposition consiste à réorganiser les lettres d'un texte clair : le texte est écrit ligne par ligne dans une grille de largeur égale à la clé, puis les colonnes sont réorganisés en fonction des positions des lettres de la clé dans l’alphabet.

Le code est le suivant :

def crypter\_message(texte, clef):

    cle = clef.upper()

    txt = texte.upper()

    cle\_classe = key\_index(cle)

    texte\_en\_claire = sup\_speciaux(txt)

    colone = len(cle)

    if len(texte\_en\_claire) % colone == 0:

        ligne = int(len(texte\_en\_claire) / colone)

    else:

        ligne = int(math.floor(len(texte\_en\_claire) / colone) + 1)

    matrice = creer\_tableau(texte\_en\_claire, ligne, colone)  # on crait notre matrice classe en ordre

    matrice\_chifre = reoganiser\_colones(matrice, cle\_classe)

    texte\_chifre = text\_a\_afficher(matrice\_chifre)

    result\_f = []

    for d in texte\_chifre:

        if d != "\*":

            result\_f.append(d)

    resul= ''.join(result\_f)

    return resul

Exemple :

Texte clair : SALUT LES PETITS POTS

Clé : GRAIN

Grille :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **R** | **A** | **I** | **N** |
| **2** | **5** | **1** | **3** | **4** |
| **S** | **A** | **L** | **U** | **T** |
| **L** | **E** | **S** | **P** | **E** |
| **T** | **I** | **T** | **S** | **P** |
| **O** | **T** | **S** |  |  |

Réorganisation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **G** | **I** | **N** | **R** |
| **L** | **S** | **U** | **T** | **A** |
| **S** | **L** | **P** | **E** | **E** |
| **T** | **T** | **S** | **P** | **I** |
| **S** | **O** |  |  | **T** |

Texte chiffre :

**LSTSSLTOUPSTEPAEIT**

## Déchiffrement

Le déchiffrement consiste à reconstruire la grille et lire le texte ligne par ligne.

Le code est le suivant :

def decrypter\_message(txt, cle):

    texte = txt.upper()

    texte\_chif = sup\_speciaux(texte)

    colones = len(cle)

    cle\_indice = index\_key(cle)

    if (len(texte\_chif) / colones == 0):

        lignes = int(len(texte\_chif) / colones + 1)

        reste = 0

        matrice\_decryp = decrypt\_LVLUP(texte\_chif, cle\_indice, lignes, colones, reste)

        mat = ''.join(matrice\_decrypt)

    else:

        reste = len(texte\_chif) % colones

        ligne = int(math.floor(len(texte\_chif) / colones) + 1)

        matrice\_ch = decrypt\_LVLUP(texte\_chif, cle\_indice, ligne, colones, reste)

        mat = ''.join(matrice\_ch)

    return mat

Exemple :

Texte chiffré : **LSTSSLTOUPSTEPAEIT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |
| **A** | **G** | **I** | **N** | **R** |
| **L** | **S** | **U** | **T** | **A** |
| **S** | **L** | **P** | **E** | **E** |
| **T** | **T** | **S** | **P** | **I** |
| **S** | **O** |  |  | **T** |

Reorganisation :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **R** | **A** | **I** | **N** |
| **2** | **5** | **1** | **3** | **4** |
| **S** | **A** | **L** | **U** | **T** |
| **L** | **E** | **S** | **P** | **E** |
| **T** | **I** | **T** | **S** | **P** |
| **O** | **T** | **S** |  |  |

Texte clair : SALUT LES PETITS POTS

## Cryptanalyse

La technique de cryptanalyse qui a été utilisées pour déchiffrer les messages est la force brute : le programme réalise une combinaison envoie le résultat et l’utilisateur dit s’il est satisfait du résultat si non le programme fait la combinaison suivante. Ainsi de suite. Le programme utilise un dictionnaire qui contient la liste des mots les plus courants en début de phrases pour effectuer les vérifications.

Exemple :

Texte chiffré : EIORJUBRSSUE

Texte déchiffré : JE SUIS BOURRE

Clé :[1, 0, 2]

Le code est le suivant :

def force\_s\_cle(texte):

    for n in range(3,8):

        trouve = False

        ENS = creer\_ensemble(n)

        for element in ENS:

            if (len(texte) / n == 0):

                ligne = int(len(texte) / n + 1)

                reste = 0

            else:

                reste = len(texte) % n

                ligne = int(math.floor(len(texte) / n) + 1)

            proba = bibliotheque.decrypt\_LVLUP(texte, element, ligne, n, reste)

            if verification(proba) :

                print("cette combinaison de mots peut etre probable")

                for i in range(ligne):

                    for j in range(n):

                        if proba[i][j] is not None:

                            print(f"{proba[i][j]}", end="")

                print()

                print(f"la clef utilise est{element}")

                tt = int(input("apuiyer sur '1' pour continuer ou '2'pour arreter la boucle de test"))

                if tt == 1:

                    print("continuons la boucle de test")

                    continue

                else:

                    print("felicitation vous avez arrete la boucle de test")

                    trouve = True

                    break

            elif not verification(proba):

                continue

            else:

                print("message trop court . le programme ne saurais le traiter")

                break

        if trouve == True:

            break

Nous avons testé toutes les clés possibles pour un texte court. Avec la clé correcte, le texte clair a été retrouvé en quelques secondes.

Pour améliorer la sécurité, il serait possible de combiner la transposition avec un chiffrement par substitution ou d'augmenter la complexité de la clé.

# Conclusion

Ce TP nous a permis de comprendre le fonctionnement du chiffrement par transposition, ses forces et ses limites. Les techniques de cryptanalyse utilisées ont montré que ce type de chiffrement est peu adapté aux applications modernes, mais reste une introduction intéressante à la cryptographie.

À l'avenir, nous pourrions explorer des algorithmes plus robustes comme AES ou RSA.

**Tutoriel d’applications**

Le langage de codage utilise est le python

* **Logiciel**
* Vscode et installer extensions nécessaires a l’exécution des codes python
* Python 3.13
* **Installation**
* Installer les package PySide6 :

Pour cela, il faut taper la commande « pip install PySide6 » dans son invite de commande de son SE.

* **Utilisations**

Ouvrir le fichier test pour le cryptage et le décryptage

Lancer le fichier cryptanalyse pour tester la cryptanalyse

Lancer le fichier main\_window pour exécuter le programme en fenêtre graphique

Si vous n’arrivez pas à installer tous les logiciels et fichiers cités ci-haut vous pouvez utiliser notre application nommée transpoapp

.