TP2 cryptographie

Arnaud Champierre de Villeneuve

M1 Cybersécurité & Management

03/06/2025

Lien GitHub pour accéder au code: Warwick001/cryptographie: Rendu TP2 Vigenere et Cesar

Objectif pédagogique

1) Pour Le **chiffrement monoalphabétique** consiste à remplacer chaque lettre d'un message clair par une unique autre lettre, toujours de manière identique tout au long du texte. En revanche, le **chiffrement polyalphabétique** applique une transformation variable, une même lettre peut être chiffrée différemment selon sa position, grâce à une clé répétitive.

2) Le chiffrement mono-alphabétique est facilement cassable : par l'analyse fréquentielle, certaines lettres comme E, S et A reviennent très souvent. A l'aide du brute force (dans le cas de César, il n'y a que 25 décalages possibles). Et, le chiffrement n'utilise qu'une clé, donc si la clé est interceptée, le message peut être facilement déchiffré.

3) Selon **Claude Shannon**, la confusion vise à rendre la relation entre la clé et le message aussi complexe et imprévisible que possible. Pour y parvenir, il serait recommandé d'utiliser des chiffrements polyalphabétiques, comme celui de Vigenère, qui repose sur des substitutions dépendantes de la position à l'aide d'une clé répétitive ou évolutive.

   Pour renforcer davantage la sécurité, il est essentiel de combiner substitution et permutation, garantissant ainsi une protection accrue contre les tentatives de décryptage.

Chiffrement César

Voir le lien Github

1) Fonction qui lorsqu'on lui passe deux valeurs : le clair et le décalage

```
# Texte clair : "Bonjour"
# Clé de décalage : 3
# Mode : 1 (chiffrement)
texte = "Bonjour"
cle = 3
mode = 1
```

```
print(cesar(texte, cle, mode))  # Résultat : "Erqmrxu"
```

La fonction consiste à décaler chaque lettre alphabétique du message clair selon une clé (ci-dessus, 3 lettres plus loin dans l'alphabet). Les caractères non alphabétiques ne sont pas modifiés.

2) Maintenant l'inverse : le chiffré et le décalage

```
if mode == 2:
   cle = -cle
```

Fonction pour le déchiffrement dans la même fonction ci-dessus.

```
# Texte chiffré : "Erqmrxu"
# Clé de décalage : 3
# Mode : 2 (déchiffrement)
texte = "Erqmrxu"
cle = 3
mode = 2
print(cesar(texte, cle, mode))  # Résultat : "Bonjour"
```

Le déchiffrement est fait simplement en inversant le décalage. Si on a chiffré avec +3, on déchiffre avec -3.

Exercice de programmation

Chiffrement Vigenere

1) Définir une fonction qui lorsqu'on lui passe deux valeurs : le clair et la clé

```
def vigenere_encrypt(plaintext, key):
```

J'utilises `cycle(key)` d'`itertools`, ce qui répète la clé indéfiniment pour l'aligner avec la longueur du texte clair.

Je calcules le décalage basé sur la lettre de la clé `(ord(next(key_cycle).lower()) - ord('a'))`.

Et j'appliques ensuite ce décalage en respectant la casse avec `(islower() / isupper())`.

```
vigenere_encrypt("Bonjour à tous", "cle")
# → Texte chiffré, dépendant de "cle" comme clé polyalphabétique
```

2) Ensuite l'inverse : le chiffré et la clé

```
def vigenere_decrypt(ciphertext, key) :
```

Même principe que encrypt, mais je soustrait le décalage au lieu de l’ajouter.

Le modulo %26 permet de gérer les boucles alphabétiques.

```
vigenere_decrypt("Fshnwrx é xwks", "cle")
# → Retourne : "Bonjour à tous"
```