Sécurité Informatique

TP1 : Chiffrement par substitution mono-alphabétique

Dans ce TP, nous allons tout d'abord essayer de simuler le fonctionnement d'un algorithme de chiffrement mono-alphabétique. Nous intéresser spécifiquement à étudier l'algorithme de ***César***. Nous apprendrons donc à simuler le principe de chiffrement et de déchiffrement de ***César***. Ensuite nous nous intéressons à savoir comment casser un message crypté selon la méthode de césar en utilisant de techniques différentes : ***« Brut Force »*** et ***« Cryptanalytic attack »***.

1. Introduction : *Le chiffrement de César*

Lors de ses batailles, l'empereur romain ***JULES CÉSAR*** cryptait les messages qu'il envoyait à ses généraux. Sa méthode de codage consistait à décaler les lettres de ***3 rangs***, vers la droite, dans l'alphabet.
Cette méthode de cryptage est appelée chiffrement de ***César***, ou Code César.
Le chiffrement de César est un cas particulier de chiffrement ***mono-alphabétique***, c'est-à-dire un chiffrement lettre à lettre.

$$A \longmapsto D \quad B \longmapsto E \quad C \longmapsto F \quad \dots \quad W \longmapsto Z \quad X \longmapsto A \quad Y \longmapsto B \quad Z \longmapsto C$$

Le chiffrement de ***César*** est simplement une addition dans ***Z/26Z***.
Le décalage ***k*** s'appelle la ***clé de chiffrement***, c'est l'information nécessaire pour crypter le message. Il y a donc ***26 clés différentes***.

2. Algorithme de chiffrement

1) Ecrivez un script en ***Python*** d'une fonction nommée ***« caesarEncryption»*** qui récupère un texte original ***plaintext*** en ***Anglais*** et une clé de cryptage ***K*** afin de générer un nouveau texte ***ciphertext*** chiffré par la méthode de ***César*** par la clé ***K***.

$$\boldsymbol{C = E(p) = (p + K)\ mod\ (26)}$$

2) Après avoir établir la fonction ***« caesarEncryption »***, il est intéressant maintenant de la tester sur un fichier contenant un message à crypter (***plaintext***). Selon cet objectif, on vous demande d'écrire un programme ***Python*** qui permet de :

1. Récupérer le contenu (***plaintext***) d'un fichier ***F*** de type texte (.text).
2. Choisir un nombre aléatoire ***K*** compris entre ***1 et 25***.
3. Appliquer la fonction de ***« caesarEncryption »*** en chiffrant le ***plaintext*** de fichier ***F*** avec un décalage de ***K***.
4. Enregistrer le résultat dans un nouveau fichier nommé ***« caesar_Result.txt »***

3) Finalement, essayez d'établir une fonction nommée ***« caesarDeciphering »*** qui permet de déchiffrer un ***plaintext*** selon une clé de cryptage **K**.

$$\boldsymbol{P = D(c) = (c - K)\ mod\ (26)}$$

3. Méthode de cryptanalyse

Après savoir comment simuler un algorithme de cryptage ***mono-alphabétique***, il est intéressant maintenant d'écrire un programme capable de casser un message déjà chiffré ***ciphertext*** et déterminer sa clé de cryptage ***K***. A cet égard, nous allons suivre deux méthodes différentes.

1.1. Méthode n°1 : Attaque à force brute :

1) Dans cette méthode, on va essayer toutes les combinaisons possibles (sur un ***ciphertext*** pour le déchiffrer) par la clé ***K*** jusqu'à trouver la bonne. On vous demande, donc, d'écrire un script d'une fonction nommée ***« ceasarBrutForce »*** qui permet de déchiffrer un texte ***ciphertext*** en plusieurs textes.
NB. Cette fonction doit retourner une liste de textes.

2) Après avoir une liste de textes, il nous reste qu'interpréter ce que donne chacune des 25 combinaisons possibles. Cependant, cette méthode n'est pas pratique pour un système automatisé. Afin de corriger ce problème, et obtenir un système capable de détecter la clé exacte automatiquement, on vous demande d'écrire un programme qui se base sur les étapes suivantes :

1. Créez un fichier de type texte content les mots vides (***stopwords***) de langue ***anglaise*** nommé ***« stopwords.txt »***.

2. Dans un script, python écrivez les lignes de codes permettant de :

- Récupérer le contenu (***ciphertext***) d'un fichier ***F*** de type texte (.text) dans une chaine nommée ***ciphertext***.
- Récupérer tous les mots existant dans ***« stopwords.txt »*** dans une liste de mots nommée ***stopWords***.
- Enregistrer le résultat obtenu de la fonction ***« ceasarBrutForce*** » sur la chaine ***ciphertext*** avec toutes les clés de cryptages possibles dans une liste nommée ***result***.
- Créez une liste nommée ***freq*** contenant le nombre d'occurrence des mots vides pour chaque texte obtenu dans la liste ***result***. Tel que ***freq[i]*** désigne le nombre total des mots vides existant dans le texte ***result[i] (avec 0≤i≤24)***.
- Déterminer la position ***p*** de plus grand élément dans la liste ***freq***.
- La clé de cryptage est donc ***K = P+1***.

1.2. Méthode n°2 : Attaque cryptanalytique :

Cette méthode se base sur des attaques statistiques. L'idée est de calculer la fréquence d'apparition de chaque symbole dans le ***ciphertext*** et de le comparer aux fréquences d'apparition des lettres de l'alphabet dans une langue particulière. Dans notre cas, on s'intéresse uniquement sur les textes écrits en ***anglais***. A cet égard, on vous demande d'écrire un programme python capable de casser un texte ***ciphertext*** et déterminer sa clé de cryptage en se basant sur les étapes suivantes :

1. Récupérez le contenu (***ciphertext***) d'un fichier ***F*** de type texte (.text) dans une chaine nommée ***ciphertext***.
2. Créez une liste nommée ***freqEn*** contant les fréquences d'apparition des lettres pour un texte en ***Anglais***. Tel que ***freqEn[i]*** désigne la fréquence de la $i^{ème}$ lettre plus fréquente en anglais.
3. Créerz une deuxième liste nommée ***freqCr*** qui contient le nombre d'occurrence de chaque lettre alphabétique dans le texte ***ciphertext***.
4. Déterminer la lettre ayant une grande fréquence dans le ***ciphertext*** et la nommée ***maxAlpha***.
5. Calculer la différence entre la lettre ***maxAplpha*** et la première lettre dans la liste ***freqEn (frqEn[0])***. Le résultat obtenu désigne la clé de cryptage ***K***.
6. En utilisant la fonction ***« caesarDeciphering »*** déchiffrer le texte ***ciphertext*** avec la clé ***K*** obtenu et vérifier que le texte obtenu s'agit d'un texte clair et correcte en langue anglaise.
7. Si le texte obtenu et correct, arrêter le programme. Si ce n'est pas le cas votre programme doit retester et recomparer la lettre ***maxAlpha*** avec une autre lettre plus fréquente dans la liste ***freqEn***.
8. Répéter les deux dernières étapes jusqu'à obtenir un texte correct.