Réseau et sécurité
M1 informatique

TP sur l'implémentation et la cryptanalyse du chiffrement de Vigenère

Dans ce TP, nous allons implémenter un chiffrement de Vigenère. Puis, nous implémenterons des outils pour le casser.

1. Implémentation du chiffrement de Vigenère

Exercice 1 : Implémentez un programme (en C par exemple) qui demande à l'utilisateur de saisir un texte, et qui l'affiche.

Exercice 2 : Modifiez votre programme pour qu'il convertisse toutes les lettres en minuscules, et qu'il enlève tous les autres caractères.
Exemple:
user$ prog2
Entrez un texte : Le soleil brille!
Texte non chiffré : lesoleilbrille

Exercice 3 : Implémentez une fonction qui prend en entrée deux lettres minuscules (l'une étant une lettre du texte non chiffré, et l'autre une lettre de la clé), et qui retourne la lettre minuscule correspondante chiffrée.
Exemple :
user$ prog3 m b
b = 2
m + 2 = o

Exercice 4 : Modifiez votre program pour qu'il demande à l'utilisateur deux textes (l'un étant le texte non chiffré, et l'autre la clé), qui les convertit tous les deux (selon l'exercice 2), et qui chiffre le texte avec le chiffrement de Vigenère et la clé donnée.

Exercice 5 : Implémentez un programme qui demande à l'utilisateur deux textes (l'un étant le texte chiffré, et l'autre la clé), et qui déchiffre le texte.

2. Cryptanalyse par estimation de la longueur de la clé et analyse fréquentielle

Lorsque la longueur de la clé est connue, un chiffrement de Vigenère peut être cassé par une analyse fréquentielle.

2.1 Méthode de Babbage et Kasiki

Exercice 6 : Implémentez un programme qui prend en paramètre un texte chiffré, et qui affiche toutes les occurrences de séquences de 3 lettres ou plus qui se répétent.
Exemple :
user$ prog6
cipher: abcdefghijklmnopqrstuvwxyzabcdmnoabc
abc trouvé 3 fois
bcd trouvé 3 fois
abcd trouvé 2 fois

mno trouvé 2 fois

Exercice 7 : Modifiez votre programme pour qu'il calcule la longueur de la clé à partir des distances entre les répétitions.

Exercice 8 : Améliorez votre programme pour qu'il supprime les répétitions peu probables (par exemple, 10% des répétitions). Expliquez ce que vous considérez comme peu probable.

2.2 Test de Friedman

Exercice 9 : Soit *Tr* un grand texte, généré aléatoirement, en utilisant seulement des lettres minuscules. Quelle est la probabilité *Kr* que deux lettres choisies aléatoirement soient égales, dans *Tr* ?

Exercice 10 : Soit *Te* un grand texte rédigé en anglais, et utilisant uniquement des lettres minuscules. La probabilité que deux lettres choisies aléatoirement soient égales dans *Te* est environ $Ke \approx 0.067$. Expliquez pourquoi cette valeur est différente de la valeur de l'exercice 9.

Exercice 11 : Soit *T* un texte utilisant uniquement des lettres minuscules. Écrivez un programme qui calcule la probabilité *K* que deux lettres choisies aléatoirement soient les mêmes dans *T*.
Remarque : Vous pouvez considérer les 26 événements indépendants consistant à choisir la lettre *li* d'abord. Ainsi, *K* devient la somme des probabilités *Ki*, où *Ki* est la probabilité que deux lettres choisies aléatoirement soient égales à *li*.

Exercice 12 : Le test de Friedman estime la longueur de la clé *L* comme $(Ke-Kr)/(K-Kr)$. Calculez *L*.
Remarque : Quand $L=1$, on a $Ke=K$, puisque le chiffrement de Vigenère correspond alors au cas d'un chiffrement par substitution simple. For $L>1$, *K* est égal à la probabilité que *li* soit égale à *lj*, avec *li* et *lj* qui correspondent à la même position dans la clé, plus la probabilité que *li* soit égale à *lj*, avec *li* et *lj* qui correspondent à des positions différentes dans la clé. Ainsi, *K* est égal à Ke/L (car il y a une probabilité $1/L$ que *li* et *lj* correspondent à la même position de la clé) plus $(L-1).Kr/L$ (car il y a une probabilité $(L-1)/L$ que *li* et *lj* correspondent à des positions différentes de la clé). Dans ce cas, on a $(Ke-Kr)/(K-Kr)=L$.

Exercice 13 : Comment expliquez-vous que le test de Friedman puisse échouer ? Vous pouvez proposer plusieurs explications, par exemple en discutant sur les hypothèses de simplification faites dans la remarque de l'exercice 12.

2.3 Analyse fréquentielle

Exercice 14 : Implémentez un programme qui prend en entrée un texte chiffré et une longueur de clé, et qui casse le chiffrement de Vigenère en utilisant une analyse fréquentielle. Le programme peut demander à l'utilisateur quel caractère chiffré correspond à quel caractère en clair, mais le programme doit fournir à l'utilisateur assez d'informations.

3. Cryptanalyse par méthode du mot probable

La méthode de Bazeries consiste à deviner un mot probable, et essaye de trouver la clé en testant ce mot à toutes les positions possibles. Le mot probable doit idéalement avoir une longueur supérieure (strictement) à celle de la clé.

Exercice 15 : Implémentez un programme qui prend en entrée un texte chiffré, un mot probable et une position. Le programme essaye de décrypter le texte chiffré en utilisant le mot probable comme

clé. Si le mot probable est correctement placé, le résultat est la clé (répétée).

Exercice 16 : Modifiez votre programme pour qu'il teste toutes les positions possibles, et affiche toutes les clés possibles.
Remarque 1 : Une clé possible est un mot qui se répéte.
Remarque 2 : Prenez soin à bien faire en sorte que la clé s'affiche à partir de la bonne position (par exemple, si le mot probable est trouvé en position 2, la première lettre trouvée de la clé va être la lettre 2 ; il faut penser à réafficher la clé à partir de la première lettre).