TP cryptographie

Jean-Baptiste LARGERON, Karam ELNASORY

November 2024

1 Introduction

Nous cherchons à utiliser des codes pour crypter un message ou à le décrypter et nous allons utiliser quelques uns d'entre eux vu en cours. Le but est de se familiariser avec ceux-ci et d'en comprendre les limites ainsi que leur avantages.

2 Sommaire des fonctions

Fichier "fonctionsCryptage.py"	Fonction code()	la fonction prend en paramètre un mot et le code dans le sens inverse.
Fichier "fonctionsCryptage.py"	Fonction Cesar()	la fonction prend en paramètre un message et un nombre b qui servira de clé de cryptage et de décryptage
Fichier "fonctionsCryptage.py	fonction Vigenere()	la fonction prend en paramètre un message et une clé, avec un argument optionelle decrypte, égale à 0 par défaut si égale à 1 on passe en mode décryptage avec la même clé de chiffrement
Fichier "fonctionsCryptage.py"	Fonction CesarAffine()	la fonction prend en paramètre un message et deux nombres a et b qui serviront de clé de cryptage
Fichier "fonctionsCryptage.py"	Fonction decrypteCesarAffine()	la fonction prend en paramètre un message et deux nombres a et b pour décrypter un code crypter par le code cesar ou cesarAffine

3 Rappel des méthodes

Code inverse

Il s'agit de crypter un message en le mettant totalement à l'envers. Le décrypter se fait de la même manière, ainsi la "clé publique " et "clé privée" est la même .

Le décryptage est très simple et se voit rapidement, il suffit de lire dans le sens inverse pour le décrypter. Donc même si l'on ne possède pas la clé, il est facile de décrypter le message.

Code César simple

Le Code de César consiste à décaler chaque lettre de l'alphabet d'un indice b. b est donc la clé de cryptage. Pour rester dans le même alphabet, il faut alors moduler à la taille de l'alphabet, ainsi :

$$y = x + b[n],$$

où y est la lettre obtenue par le cryptage, x la lettre initiale et n la taille de l'alphabet.
Pour décrypter le message, il suffit de résoudre la formule suivante :

$$x = y - b[n],$$

On peut alors voir que b est la clé de cryptage ET de décryptage.

Cela rend le code très simple à décrypter, et donc à comprendre le message même si l'on n'a pas la clé de décryptage. En effet, il suffit de faire un décalage de lettres jusqu'à obtenir un message lisible.

Code de Vigenère

Le code de Vigenère est un dérivé du code de César. Le décalage dépend d'une clé qui sera un mot et chaque lettre du message sera décaler du même décalage qui fait passer la lettre "a" à la lettre du mot correspondante et l'on répète autant de fois que nécessaire.
Ainsi, le message "il fait froid ce matin" devient avec la clé "azur" : "ikzrisziohxtelukim".
Le code est meilleur que le code de César simple mais il souffre du même problème. Le décalage est facile à retrouver.

Code César Affiné

Le code de César Affiné se compose comme le code de César simple mais prend deux clé $aetb$ tel que:

$$y = ax + b[n],$$

y étant le mot codé et n la taille de l'alphabet.

Ainsi, il est plus difficile de retrouver le message originel sans connaître les deux clés.
Le code nécessite cependant que a et n soient premiers entre eux puisque le décryptage se fait en trouvant a^{-1}.

$$x = a^{-1}(y - b)[n],$$

Si a n'a pas d'inverse, alors il n'y a pas de décryptage possible.

4 Descriptif du programme

Nous avons découpé notre programme en 2 fichiers distincts. Le fichier alphabet.py contient notre alphabet ainsi que sa taille, tandis que fonctionsCryptage.py contient les fonctions de chiffrement et main.c contient tout les tests (chiffrement et déchiffrement) sur les jeux d'essai ainsi le fichier OutilesMaths.py qui contient les outiles mathématique utilisé pour ce TP et le prochain.

Dans ce dernier, nous avons créé les fonctions suivantes:
Code(message):

la fonction prend en paramètre un message, sous la forme d'un string (tableau de caractères). Elle lit le tableau de caractères dans le sens inverse et le retourne.

Cesar(message, b):

Cette fonction code un message passé en paramètre en code césar grâce au deuxième paramètre. b est le décalage dans l'alphabet (c'est-à-dire la distance entre deux lettres de l'alphabet). Ainsi, un message comme "azur" avec la clé 1 devient "bavs" avec notre alphabet commun.

Vignere(message, key, dechiffre=0):
Comme vu dans le rappel des méthodes, le code de Vigenere est un "dérivé" du code de césar. Nous avons fait le choix d'implémenter le codage et le décodage du code dans la même fonction, ainsi il suffit de mettre le paramètre déchiffre à 0 pour coder le message ou mettre un autre chiffre pour le décoder.
CesarAffine(message, a, b) :

la fonction marche de la même manière que la fonction césar sauf que nous avons une clé plus complexe : a et b. a multiplie l'index du mot passer.

decrypteCesarAffine(message, a, b):

nous avions besoin de décoder les messages codés par césar. Nous avons donc fait le choix de faire une fonction permettant de décoder cesar (il suffit de fixer a = 1) et cesarAffine. Cette fonction traduit un message passé en paramètre grâce à la clé a,b.
Nous avions pour cela besoin de l'inverse de a et b dans le modulo de la taille de l'alphabet. Nous avons choisit d'utiliser la fonction gcd de la librairie math afin de connaître le PGCD entre la taille de l'alphabet et a pour savoir s'il existe un inverse à a dans le modulo de la taille de l'alphabet et s'il existe, connaître sa valeur. Alors nous avons crée la fonction modInv pour faire cette action.

5 Jeux d'essais

Dans cette partie nous allons tester les fonctions que nous avons coder:

Nous avons donc ajouter des lignes d'affichage après chaque fonction pour vérifier celles-ci.

message	méthode de chiffrement/déchiffrement	clé de chiffrement	message chiffré
"Hello, World!"	Code()	aucune	"!dlroW ,olleH"
"Hello, World!"	cesar()	3	"!dlroW ,olleH"
ilfaitfroidcematin	vignere()	"azur"	k0'fkg'p]kbWo/fgk1
"Hello"	CesarAffine()	3 et 5	"7]ZZ,"
"7]ZZ,"	decrypteCesarAffine()	3 et 5	"Hello"

6 Conclusion

Nous avons vu dans ce TP différente façon de crypter un message par le modulo. Ceux-ci sont très pratiques pour crypter des messages rapidement mais sont tout aussi contraignant par leur faciliter à être décrypter. En effet, les machines peuvent rapidement trouver les combinaisons (a,b) pour lesquels le message crypté par CésarAffine sera décrypté.

Il est donc préférable d'utiliser une autre façon de crypter les messages si l'on veut garder le message secret comme par exemple le code RSA.