TP Cryptage

M'hammed Salmân Abibou & Rodrigo Ferreira Rodrigues
Université Clermont Auvergne

December 6, 2022

Contents

1 Rappel des méthodes

1.1 PGCD

Le **PGCD**, ou plus grand commun diviseur, est le plus grand entier d, diviseur commun de deux entiers, a et b.
Il se note PGCD(a, b).

Exemple : $14 = 7 \times 2$ et $21 = 7 \times 3$ donc PGCD$(14, 21) = 7$

1.2 Algorithme d'Euclide Etendu

Il se repose sur l'**Identité de Bezout**:

$$\text{si } d = Pcgd(a, b) \text{ alors } \exists (u, v) \in \mathbb{Z}^2 \text{ tels que } au + bv = d \qquad (1)$$

L'algorithme permet alors de calculer u et v.

1.3 Code inverse

Le but est d'inverser un mot : le premier caractère devient dernier, le deuxième devient avant-dernier, ...
Que ce soit pour crypter ou décrypter, le fonctionnement reste inchangé.

Exemple : *Bonjour* devient *ruojnoB*

Le but est de chiffrer un mot par décalage des caractères de l'alphabet d'une valeur fixe que l'on va ici appeler b.

On appelle n, la taille de l'alphabet.

Si on dépasse la dernier caractère de l'alphabet, on reprend au premier : on travaille alors sur modulo n.

On obtient alors la clé de chiffrement suivante :

$$f : \begin{cases} 0, .., n \to 0, .., n \\ x \to f(x) = x + b \bmod n \end{cases}$$

Posons y, le nouveau caractère avec $y = f(x)$. On cherche la fonction g tel que $g(y) = x$.

On en déduit alors que la clé de déchiffrement est :

$$g : \begin{cases} 0, .., n \to 0, .., n \\ y \to g(y) = y - b \bmod n \end{cases}$$

1.4 Code César affine

Le but est aussi de chiffrer un mot par décalage des caractères de l'alphabet. Or ici, le chiffrement se fait à l'aide d'une fonction affine.

On obtient une clé de chiffrement de la sorte :

$$f : \begin{cases} 0, .., n \to 0, .., n \\ x \to f(x) = ax + b \bmod n \end{cases}$$

(avec a et b des entiers et n indiquant toujours la taille de l'alphabet).

D'ailleurs, pour que le code fonctionne, on doit s'assurer que $PGCD(a, n) = 1$, soit que a et n sont premiers entre eux.

Cependant, dans ce cas me déchiffrement est plus difficile. On doit toujours aboutir à une fonction affine g du type $g(y) = x = a'y + b'$.

Puisque on a $y = f(x)$ alors on en déduit que :

$$g(y) = a'x + b' = a'(ax + b) + b'[n]$$

d'où

$$= a'ax + a'b + b'[n]$$

Par identification, on obtient $aa' \equiv 1[n]$ et $a'b + b' \equiv 0[n]$.

a' est donc l'inverse multiplicatif de $a[n]$ et on en déduit que $b \equiv -a'b[26]$.

On obtient a' grâce à l'algorithme d'Euclide Etendu. En effet, puisque $PGCD(a, n) = 1$ alors d'après l'identité de Bezout $\exists(u, v), au + nv = 1$, avec ici $u = a'$.

2 Programmes python & Jeux d'essais

2.1 Programmes python

2.1.1 Code inverse

```
##Code Inverse

def crypt(mess):
    traduction = "" #on créé un mot vide qui va contenir le
    message inversé
    i = len(mess) - 1 #on affecte à i la taille du message moins
    1
    while(i>-1): #i>-1 pour prendre en compte le cas où i=0
        traduction = traduction + mess[i]
        i -=1 #on décrémente de 1
    return traduction
```

Listing 1: Fonction Code Inverse.

2.1.2 Code César simple

```
## Code César Simple

def cesar(message,alphabet):
    print("Cryptage/Décryptage avec César Simple")
    b = int(input("Clé de chiffrement :")) # b représente la clé
    de chiffrement
    mode = input("Cryptage ('c') ou décryptage('d') :")
    if mode == 'd':
        b = -b%len(alphabet) # recalcul de b pour le dé
    chiffrement
    messchi = [] # création d'une liste vide qui va contenir le
    message chiffré(indices dans l'alphabet choisi)
    messdchi =[] # création d'une liste vide qui va contenir le
    message dechiffré(indices dans l'alphabet choisi)
```

```
    traduction ="" # création d'un mot vide qui va contenir la traduction du message déchiffré

    for i in message: # parcours de chaque lettre du message
        if i in alphabet:
            k = alphabet.find(i)# si la lettre se trouve dans l'alphabet choisi, on récupère et on affecte la valeur de l'indice correspondant dans k
        else :
            k=0 #si la lettre n'est pas dans l'alphabet, on met par défaut la première lettre de l'alphabet
        messchi.append(k) # actualisation de la liste du message chiffré

    for j in messchi: # parcours des chiffres dans le message chiffré
        j = (j+ b)%len(alphabet) # mise à jour de la valeur des chiffres(indices) en utilisant  y=(x-b)mod n avec n la taille de l'alphabet choisi
        messdchi.append(j) # actualisation de la liste du message déchiffré
    for i in messdchi:
        traduction += alphabet[i] # transformation des chiffres en lettres
    return traduction
```

Listing 2: Fonction Code César Simple.

2.1.3 PGCG

```
#Fonction pgcd

def pgcd(a,b):
    if b==0: # test d'arrêt
        return a
    else:
        return pgcd(b,a%b)
```

Listing 3: Fonction PGCD.

2.1.4 Algorithme d'Eucide Etendu

```
#Fonction Euclide Etendu
def euclideEtendu(a,b):
```

```
    if pgcd(a,b) == 1:
        r=[a,b] # initialisation de la liste des restes contenant la clé a et la taille de l'alphabet choisi
        q=[-1]  # initialisation de la liste des quotients (-1 pour signifier que la division par 0 est impossible avec des entiers)
        u=[1,0] # initialisation de u0 et u1
        v=[0,1] # initialisation de v0 et v1

        i = 2 # initialisation de i à 2
        while r[i-1]!=0: # test d'arrêt (lorsque le reste est nul)
            r.append(r[i-2] % r[i-1]) # mise à jour de la liste des restes
            q.append(r[i-2] // r[i-1])
            u.append(u[i-2] - q[i-1] * u[i-1])
            v.append(v[i-2] - q[i-1] * v[i-1])
            i+=1 # incrémentation de 1
```

Listing 4: Algorithme d'Euclide Etendu.

2.1.5 Code César Affine

Dans le code César affine nous allons utilisés les fonctions pgcd et Euclide étendu.

```

def cesarAffine(message,alphabet):
    print("Cryptage/Décryptage avec César Affine")
    a = int(input("Valeur de a :")) # clé de chiffrement a
    while pgcd(a,len(alphabet))!=1: # vérification que la clé a et la taille de l'alphabet sont premiers entre eux
        print("a n'est pas premier avec la taille de l'alphabet")
        a = int(input("Valeur de a :"))
    b = int(input("Valeur de b :")) # clé de chiffrement b
    mode = input("Cryptage ('c') ou décryptage('d') :")
    if mode == 'd':
        a = euclideEtendu(a,len(alphabet)) # calcul de l'inverse multiplicatif pour obtenir la clé de déchiffrement a'
        b = (-a*b)%len(alphabet) # calcul de b'
    messchi = [] #création d'une liste vide qui va contenir le message chiffré(indices dans l'alphabet choisi)
    messdchi =[] #création d'une liste vide qui va contenir le message dechiffré(indices dans l'alphabet choisi)
    traduction ="" #création d'un mot vide qui va contenir la
```

```
    traduction du message déchiffré
16
17      for i in message:
18          if i in alphabet: # parcours de chaque lettre du message
19              k = alphabet.find(i) # affectation de la valeur de l'
    indice correspondant à i dans k, si i se trouve dans l'
    alphabet choisi, on récupère et on affecte la valeur de l'
    indice correspondant dans k
20          else :
21              k=0 #si la lettre n'est pas dans l'alphabet, on met
    par défaut la première lettre de l'alphabet
22          messchi.append(k) # actualisation de la liste du message
    chiffré
23
24      for j in messchi: # parcours des chiffres dans le message
    chiffré
25          j = (a*j + b)%len(alphabet) # mise à jour de la valeur
    des chiffres(indices) en utilisant  y=(a'x+b')mod n avec n la
     taille de l'alphabet choisi
26          messdchi.append(j) # actualisation de la liste du message
     déchiffré
27      for i in messdchi:
28          traduction += alphabet[i] # transformation des chiffres
    en lettres
```

Listing 5: Fonction Code César Affine.

2.2 Jeux d'essais

Pour nos jeux d'essais on a construit un programme principal qui s'opère comme suit :

1. On demande à l'utilisateur quel alphabet il veut utiliser entre les deux qu'on lui propose;

2. Ensuite, l'utilsateur doit faire un choix du message qu'il veut crypter ou décrypter entre quatre (il saisit un chiffre entre 1 et 4) propositions de messages et a aussi le choix de personnaliser son message en le saisissant lui-même (il saisira un chiffre supérieur à 4, 5 par exemple);

3. Enfin, il lui est demandé de choisir par quel méthode il veut crypter ou décrypter le message choisi (1 pour le code inverse, 2 pour le code César simple et 3 pour le code César affine).

2.2.1 Jeu 1

Crypter/Décrypter : Si deux hommes ont la même opinion. L'un d'eux est de trop

2.2.2 Jeu 2

Décrypter : 'snoçel sed ennod em no uq sruojuot sap emia n ej euq neib erdnerppa a terp sruojuot sius ej'

2.2.3 Jeu 3

Décrypter : Cyhfibhffnhermertneqreybvaqnafyrcnffrcyhfibhfireermybvaqnafyrshghee (César : cle 13)

2.2.4 Jeu 4

Crypter/Décrypter : ""Chez moi, le secret est enfermé dans une maison aux solides cadenas dont la clé est perdue et la porte scellée."-Les milles et une nuits" (cle 2023)