Compte Rendu

Cryptographie Appliquée

Réalisé par
EL ASRI Ayoub

Groupe
4A SAGI TD1

A- Chiffrements pré-informatique

Chiffre de César

Question 1

Programme qui permet de chiffrer le texte en clair m à l'aide de la clé k.

```
def chiffreCesar(texte,k):
    lettres = ""
    for i in range(0, len(texte)):
        rang = ord(texte[i]) - ord('A')
        rang = (rang + k) %26 + ord('A')
        lettres += chr(rang)
    return lettres

#test de la fonction
m = "AYOUB"
k = 2
print("Texte à chiffrer en Cesar : " + m)
print( "Texte chiffré en Cesar avec k = " + str(k) + " : " +
chiffreCesar(m,k))
```

```
Question 1:
Texte à chiffrer en Cesar : AYOUB
Texte chiffré en Cesar avec k = 2 : CAQWD
```

Question 2

Programme qui demande un texte chiffré s et qui affiche les 25 possibilités de texte clair.

```
def possibilitesTexte25(texte):
     possib = []
     for i in range(0, 26):
          possib.append(chiffreCesar(texte, i))
     return possib

#test de la fonction
s = input("Entrez un texte chiffré : ")
print( "les 25 possibilités du texte " + s + " sont : ")
print(possibilitesTexte25(s))
```

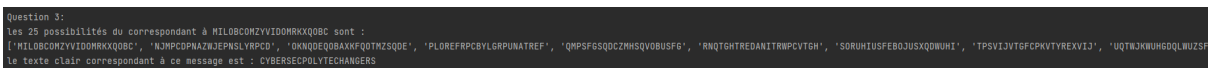
Toute la liste des 25 possibilités :

```
['CAQWD', 'DBRXE', 'ECSYF', 'FDTZG', 'GEUAH', 'HFVBI', 'IGWCJ',
'JHXDK', 'KIYEL', 'LJZFM', 'MKAGN', 'NLBHO', 'OMCIP', 'PNDJQ',
'QOEKR', 'RPFLS', 'SQGMT', 'TRHNU', 'USIOV', 'VTJPW', 'WUKQX',
'XVLRY', 'YWMSZ', 'ZXNTA', 'AYOUB', 'BZPVC']
```

Question 3

*Quel est le texte clair correspondant au message **MILOBCOMZYVIDOMRKXQOBC** ?*

```
message = "MILOBCOMZYVIDOMRKXQOBC"
print("les 25 possibilités du correspondant au message sont : ")
print(possibilitesTexte25(message))
print("le texte clair du message est : " +
str(possibilitesTexte25("message")[16]))
```



Toute la liste des 25 possibilités :

```
['MILOBCOMZYVIDOMRKXQOBC', 'NJMPCDPNAZWJEPNSLYRPCD',
'OKNQDEQOBAXKFQOTMZSQDE', 'PLOREFRPCBYLGRPUNATREF',
'QMPSFGSQDCZMHSQVOBUSFG', 'RNQTGHTREDANITRWPCVTGH',
'SORUHIUSFEBOJUSXQDWUHI', 'TPSVIJVTGFCPKVTYREXVIJ',
```

```
'UQTWJKWUHGDQLWUZSFYWJK', 'VRUXKLXVIHERMXVATGZXKL',
'WSVYLMYWJIFSNYWBUHAYLM', 'XTWZMNZXKJGTOZXCVIBZMN',
'YUXANOAYLKHUPAYDWJCANO', 'ZVYBOPBZMLIVQBZEXKDBOP',
'AWZCPQCANMJWRCAFYLECPQ', 'BXADQRDBONKXSDBGZMFDQR',
'CYBERSECPOLYTECHANGERS', 'DZCFSTFDQPMZUFDIBOHFST',
'EADGTUGERQNAVGEJCPIGTU', 'FBEHUVHFSROBWHFKDQJHUV',
'GCFIVWIGTSPCXIGLERKIVW', 'HDGJWXJHUTQDYJHMFSLJWX',
'IEHKXYKIVUREZKINGTMKXY', 'JFILYZLJWVSFALJOHUNLYZ',
'KGJMZAMKXWTGBMKPIVOMZA', 'LHKNABNLYXUHCNLQJWPNAB']
```

Chiffre affine

Question 4

L'inverse de a=4 avec m=9

```
def inverseMod(a,m):
      _a = 0
      for i in range(0, m):
            if((a*i)%m ==1):
                  return i
      return str(str(a) + " et 26 ne sont pas premiers entre eux
!!")

#Test de la fonction
a1=4
m=9
print("L'inverse de a=4 avec m=9 : ")
print(inverseMod(a1,m))
```

```
Question 4:
L'inverse de a=4 avec m=9 :
7
```

Question 5

L'inverse de a=6 avec m=9

```
#Application de la function du Q4
a2=6
m=9
print("L'inverse de a=6 avec m=9 : ")
print(inverseMod(a2,m))
```

```
Question 5:
L'inverse de a=6 avec m=9 :
6 et 26 ne sont pas premiers entre eux !!
```

Question 6

L'algorithme d'Euclide étendu.

```
def igcd(a,b):
      # Initialisation
      d,u,v,d1,u1,v1=a,1,0,b,0,1
      # Calcul
      while d1!=0:
            q=d//d1
            d,u,v,d1,u1,v1=d1,u1,v1,d-q*d1,u-q*u1,v-q*v1
      return (d,u,v)

#Application de la function
a,b=488456,18546
a,b=4445847,64545454
d,u,v=igcd(a,b)
print('pgcd(%d,%d) = %d' % (a,b,d))
print('(%d)*%d + (%d)*%d = %d' %(u,a,v,b,d))
```

```
Question 6:
pgcd(4445847,64545454) = 1
(19976707)*4445847 + (-1375982)*64545454 = 1
```

Source du code :

https://codes-sources.commentcamarche.net/source/102305-algorithme-d-euclide-etendu

Question 7

Programme de chiffrement et déchiffrement affine avec a et b en paramètres.

```
def chiffrementAffine(message, a, b):
      chiffre = ""
      for lettre in message:
            rang = ord(lettre) - ord('A')
            rang = (a * rang + b) % 26 + ord('A')
            chiffre += chr(rang)
      return chiffre

def dechiffrementAffine(chiffre, a, b):
      _a = inverseMod(a,26)
      if(type(_a) is int):
            message = ""
            for lettre in chiffre:
                  rang = ord(lettre) - ord('A')
                  rang = (_a * (rang - b)) % 26 + ord('A')
                  message += chr(rang)
            return message
      else:
            return "L'inverse de a n'existe pas !"

#Application de la function
message = "AYOUB"
a= 3
b= 2
print("Chiffrement de"+ message + " avec a="+ str(a) + " et b=" +
str(b) + " :")
print(chiffrementAffine(message,a,b))
print("Déchiffrement de " + chiffrementAffine(message,a,b) + "
avec a = " + str(a) + " et b = " + str(b) + " :")
 print(dechiffrementAffine(chiffrementAffine(message,a,b),a,b))
```

```
Question 7:
Chiffrement affine du message AYOUB avec a = 3 et b = 2 :
CWSKF
Déchiffrement affine du message chiffré CWSKF avec a = 3 et b = 2 :
AYOUB
```

B- Chiffrement RSA

Question 8

Calcul de la valeur d de la clé privée avec p = 53, q = 11 et e = 3.

```
def clePrive(p,q,e):
      n= p*q
      m=(p-1)*(q-1)
      d=inverseMod(e,m)
      return d

#Application de la fonction
p = 53
q = 11
e = 3
print("La clé privé d vaut : " + str(clePrive(p,q,e)))
```

```
Question 8:
La clé privé d vaut : 347
```

Chiffre par Bob

Question 9

Complétons le tableau des blocs

```
def decompositionMessage(message,nb):
    listDecompo=[]
    msg = ""
    for i in range (0, len(message)):
        if (len(str((ord(message[i]) - ord('A')))) == 1):
            c = '0' + str((ord(message[i]) - ord('A')))
        else:
            c = str(ord(message[i]) - ord('A'))
        msg = msg + c

    if ((len(msg)%nb)!=0):
        for i in range(0,nb-(len(msg)%nb)):
            msg='0'+msg
```

```
    w=len(msg)

    while (w!=0):
        listDecompo.insert(0,int(msg[w-nb:w]))
        w=w-nb
    return listDecompo
```

```
Question 9:
[1, 514, 112, 419, 40, 207, 1, 306, 41, 718]
```

Question 10

Complétons le message chiffré envoyé par Bob.

```
def chiffrementBob(message,e,n):
      chiffre = []
      for i in range(0, len(message)):
            chiffre.append((message[i]**e) % n)
      return chiffre

e = 3
n = 583
print(chiffrementBob(decompositionMessage("POLYTECHANGERS"),e,n))
```

```
Question 10:
[1, 303, 481, 34, 453, 564, 1, 498, 127, 115]
```

Question 11

Complétons le message déchiffré par Alice

```
def dechiffrementAlice(message,e,n):
      chiffre = []
      for i in range(0, len(message)):
            chiffre.append((message[i]**clePrive(53,11,e)) % n)
      return chiffre

e = 3
n = 583
```

```
print(dechiffrementAlice(chiffrementBob(decompositionMessage("POLY
TECHANGERS"),e,n),e,n))
```

```
Question 11:
[1, 514, 112, 419, 40, 207, 1, 306, 41, 135]
```

Question 12

Lors du chiffrement, on a choisi une taille de 3.
Ce « choix » est-il judicieux ? Que proposez-vous ?

Lorsqu'on a choisi 3 comme taille, le dernier bloc qu'on a déchiffré était erroné
Résultat souhaité : [1, 303, 481, 34, 453, 564, 1, 498, 127, 718]
Résultat obtenu : [1, 303, 481, 34, 453, 564, 1, 498, 127, 115]
Si on fait par exemple 2 comme taille, on obtient le bon résultat.

```
print("Pour taille = 3 : ")
print(dechiffrementAlice(chiffrementBob(decompositionMessage("POLY
TECHANGERS",3),e,n),e,n))

#Si on fait par exemple 2 comme taille, on obtient le bon
résultat.

print("Pour taille = 2 : ")
print(dechiffrementAlice(chiffrementBob(decompositionMessage("POLY
TECHANGERS",2),e,n),e,n))
```

```
Question 12:
Pour taille = 3 :
[1, 514, 112, 419, 40, 207, 1, 306, 41, 135]
Pour taille = 2 :
[15, 14, 11, 24, 19, 4, 2, 7, 0, 13, 6, 4, 17, 18]
```