TD 2 et TP 2 et 3

Chiffrement Symétrique

Consigne importante valable pour toutes vos productions ;

Pour pouvoir faire fonctionner les tests des docString de chaque fonction, ajouter au début « du code main » :

```
if  __name__== "__main__":
  import doctest
  doctest.testmod()
```

Toute les fonctions produites en Info0603 devront comporter au moins un test rédigé **avant** l'écriture du code. Cette pratique permet ainsi d'avancer sa réflexion tout en documentant le code.

C'est aussi un bon canal de communication avec son binôme ou le professeur.

Note : pour gagner du temps, reprenez les fichiers du moodle et remplacez par du code les `raise NotImplementedError`.

Déja fait : Premiers chiffreurs

Ecrire la classe ChiffreurParDecalage durant le TD, compléter son code durant le TP.

Ecrire la classe ChiffreurVigenere durant le TD, compléter son code durant le TP.

Proposer une méthode pour déchiffrer les documents DocChiffre1, DocChiffre2, DocChiffre3, DocChiffre4 puis les déchiffrer.

Les attaques sur Chiffre1,2,3 et 4, demandées au TP1 pourront être faits en fin de TP3 si nécessaire mais tout le monde doit commencer le TP2 par l'exercice suivant.

Exercice 1: Chiffreur Affine :

Ecrire à la main les sorties du programme suivant :

```
monBin=Binaire603([0x00,0x01,0x02,0x010,0x20,0x40,0x80])
for monCodeur in [ChiffreurAffine(3,5), ChiffreurAffine(1,1),
                        ChiffreurAffine(1,0), ChiffreurAffine(2,5)]:
  print(f"Codage avec monCodeur :")
  print(" Bin:",monBin)
  monBinC=monCodeur.binCode(monBin)
  print(" Bin Codé:",monBinC)
  monBinD=monCodeur.binDecode(monBinC)
  print(" Bin Décodé:",monBinD)
  print(" monBinD (décodé) est égal à Monbin ?",monBinD==monBin)
```

Quel est l'espace des clés ? Son cardinal ?

Comparer le chiffrement par décalage avec le chiffrement affine

Ecrire le code de cette classe : on gagnera à utiliser ElmtZnZ.

Tester sur un Binaire603 mais aussi un Texte603 courts.

Ecrire un algorithme d'attaque de ce chiffre.

Exercice 2: Test des bijections sur les octets

On se propose de tester différentes bijections sur des octets afin de voir leurs capacités à renvoyer un résultat statistiquement « plat », c'est-à-dire un résultat dont les caractéristiques générales du message source seront indétectables.

Pour cela, créez des classes, toutes dans un même fichier, héritées de la classe fBijOctetsCA définies comme suit :

```
class fBijOctetsCA(object):
  "Une classe abstraite de bijection de [0..255]"
  def __init__(self ):
        raise NotImplementedError
  def __repr__(self):
        raise NotImplementedError
  def __call__(self,octet):
        """Renvoie l'image de octet par la bijection"""
        raise NotImplementedError
  def valInv(self,octetC):
        "Renvoie l'antécédent de octetC"
        raise NotImplementedError
# Programmer une classe fBijParDecallage et ajouter à fBijOctetsCA une méthode affichant
un graphique permettant de voir la confusion et la diffusion générée.
# On pourra reprendre et adapter le code suivant pour construire ce graphique:
import matplotlib.pyplot as plt
lx=[-5+0.1*k for k in range(101)]
ly=[x**2 for x in lx]
plt.plot(lx,ly,"-") # ou le paramètre "." pour un graphique point par point
plt.title("La fonction carré")
plt.show()
```

- Faire le test sur une fonction affine.
- Faire le test sur une bijection inventée par vous même
- Faire le test sur une fonction appliquant un masque.

Exercice 3: (Feistel)

Programmer un chiffreur de Feistel et lui appliquer le test précédent à chaque itération. Les clés succéssives pourront être générées par "random.randint" initialisée par une graine avec "random.seed(k)."

2. Eléments de Correction

Exercice 1: Solution

```
Codage avec Chiffreur affine de avec a=3(256) et b=5(256) :
Bin : 7 octets : 00 01 02 10 20 40 80
Bin Codé : 7 octets : 05 08 0b 35 65 c5 85
Bin Décodé : 7 octets : 00 01 02 10 20 40 80
monBinD (décodé) est égal à Monbin ? True
Codage avec Chiffreur affne de avec a=1(256) et b=1(256) :
Bin : 7 octets : 00 01 02 10 20 40 80
Bin Codé : 7 octets : 01 02 03 11 21 41 81
Bin Décodé : 7 octets : 00 01 02 10 20 40 80
monBinD (décodé) est égal à Monbin ? True
Codage avec Chiffreur affine de avec a=1(256) et b=0(256) :
Bin : 7 octets : 00 01 02 10 20 40 80
Bin Codé : 7 octets : 00 01 02 10 20 40 80
Bin Décodé : 7 octets : 00 01 02 10 20 40 80
monBinD (décodé) est égal à Monbin ? True
Codage avec Chiffreur affne de avec a=2(256) et b=5(256) :
Bin : 7 octets : 00 01 02 10 20 40 80
Bin Codé : 7 octets : 05 07 09 25 45 85 05
AssertionError : a doit être inversible et donc doit être premier avec 256
# Quel est l'espace des clés ? Son cardinal ?
# Comparer le chiffrement par décalage avec le chiffrement affne
Solution : ChiffreurParDecalage(b)=ChiffreurAffine(1,b)
# Ecrire le code de cette classe : on gagnera à utiliser ElmtZnZ.
# Tester sur un Binaire603 mais aussi un Texte603 courts.
# Ecrire un algorithme d'attaque de ce chiffre.
```