TD 2: Clés secrètes

1 Algèbre polynomiale

(1) Calculer la division Euclidienne de x^4+x+1 par $x+1$. Déduisez-en le pgcd des polynômes et l'inverse de $x+1$ dans $\mathbb{F}_2[x]/x^4+x+1$.

(2) Calculer la table de multiplication des éléments de $\mathbb{F}_2[x]/x^2+x+1$. Quel est ce corps?

2 Cryptanalyses

(1) Faire la cryptanalyse statistique du texte suivant chiffré par un chiffre multiplicatif:

`RAWFEJBANAREQSSQBDAWKRSKWK`

Les lettres françaises les plus fréquentes sont `ETIANS` (18%,7%,6%6%,6%,6%)

(2) Vous pourrez faire une cryptanalyse par recherche exhaustive de la clé en écrivant la fonction de déchiffrement correspondant au code de chiffrement suivant:

```
def chiffre_mult(msg,t):
    res=''
    for car in msg:
        if car.isupper():
            res+=chr((((ord(car)-ord('A'))*t)%26)+ord('A'))
        else: res+=car
    return res
```

(3) En vous indiquant que les premières lettres du clair sont `LA`, comment la cryptanalyse en est-elle simplifiée?

3 Chiffre de Vernam

Au moyen d'un progamme `Python`, récupérez une suite aléatoire depuis `/dev/random` pour chiffrer un message (et déchiffrez-le ensuite). Pour simplifier, on chiffrera un caractère ASCII avec une suite aléatoire sur des entiers (modulo 256). Vous pourrez suivre les indications sur le web décrivant l'utilisation de la fonction `ord()` pour convertir la suite aléatoire extraite en un entier modulo 256.

4 Modes de chiffrement

En utilisant la librairie cryptography de `Python` et le source suivant, chiffrez Beastie en mode ECB puis en mode CBC.

```
import sys, base64, io, os
from cryptography.hazmat.primitives.ciphers import Cipher,
                algorithms, modes
from cryptography.hazmat.backends import default_backend
from django.utils.encoding import force_bytes, force_text

cles = "et celle-ci?"
backend = default_backend()
key = force_bytes(base64.urlsafe_b64encode(force_bytes(cles))[:32])

aesCipher = Cipher(algorithms.AES(key), modes.ECB(),backend)
aesEncryptor = aesCipher.encryptor()
aesDecryptor = aesCipher.decryptor()

plain=b'un texte simple a chiffrer'
crypto=aesEncryptor.update(plain)
```

Indications: Pour retirer l'en-tête de l'image `bmp` et obtenir une suite d'octets, faites appel à la bibliothèque Pillow qui permet de charger l'image, d'en extraire la suite d'octets et la taille; on peut alors chiffrer la suite d'octets puis reconstruire une image et l'enregistrer.

```
from PIL import Image
plain=Image.open(ifile)
w, h = plain.size
print('larg. , haut. , mode:', plain.size,plain.mode)
bplain=plain.tobytes("raw","RGBA")
benc=                    #on fait le chiffrement de bplain
enc=Image.frombytes('RGBA',(w,h),benc)
enc.save(ofile)
```

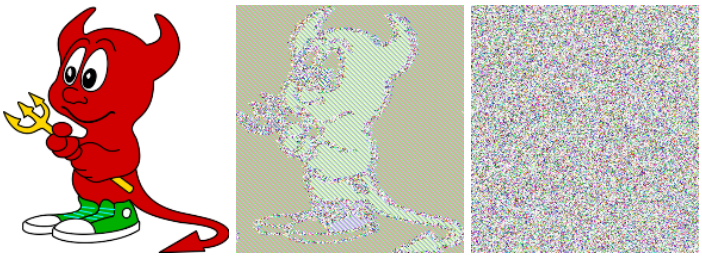

Figure 1: Beastie ECB et CBC