CY Cergy-Paris Université **Enseignant : Kévin Carrier**

Introduction à la Cryptographie kevin.carrier@cyu.fr

TD 4 : Chiffrement Symétrique par Bloc (DES,AES) et modes de chiffrement

Nous allons utiliser la librairie `pycrypto` en Python qui implémente notamment les chiffrements symétriques classiques tels que DES et AES. Pour pouvoir l'utiliser, il faut tout d'abord l'installer. Pour cela, vous pouvez utiliser la commande :

```
pip3 install pycryptodome
```

À titre d'exemple, voici un code qui permet de créer un chiffreur DES et de chiffrer/déchiffrer un message :

```
from Crypto.Cipher import DES

def pad(text):
    n = 8 - len(text) % 8
    if n == 8:
        return text
    return text + (b' ' * n)

key = b'password'
des = DES.new(key, DES.MODE_ECB)

plaintext = pad(b'Hello world !!!')
ciphertext = des.encrypt(plaintext)

print([hex(x) for x in ciphertext])
print(des.decrypt(ciphertext))
```

1. Implémenter un générateur de clé qui retourne une clé tirée uniformément dans l'ensemble des chaines de 64 bits. La clé devra être au format `byte array` (type `bytes` en Python).

La fonction suivante calcule le xor bit à bit entre deux `byte array` :

```
def xor_bytes(s1, s2):
    if len(s1) != len(s2):
        exit("xor_bytes error : len(s1) != len(s2) !")
    return bytes([a ^ b for (a,b) in zip(s1, s2) ])
```

2. Choisir une clé secrète $\mathbf{k}$ et un message clair $\mathbf{m}$ de 64 bits chacun. Tester le chiffrement et déchiffrement DES en mode ECB sur le message $\mathbf{m}$ avec la clé $\mathbf{k}$ puis avec la clé $\mathbf{k}' := \mathbf{k} \oplus 0x0101010101010101$. Expliquer le résultat obtenu.

3. Implémenter un générateur de clé DES qui retourne une clé de 64 bits respectant la parité sur chaque octet.

4. On note $\overline{\mathbf{x}}$ le vecteur binaire complémentaire de $\mathbf{x}$. Autrement dit,

$$\overline{\mathbf{x}} := \mathbf{x} \oplus 0b111\cdots1$$

Appliquer le chiffrement DES en mode ECB sur le message $\overline{\mathbf{m}}$ avec la clé $\overline{\mathbf{k}}$. Comparer le résultat obtenu avec le cryptogramme obtenu à la question 2. Le résultat vous parait-il normal? Est-ce une faille de sécurité du chiffrement DES?

5. Une clé secrète $\mathbf{k}$ est dite *faible* si elle génère des clés partielles $\mathbf{k}_i$ identiques. Il y a exactement 4 clés faibles pour DES. Proposer un algorithme qui permet de les trouver.

   Aide. On remarque que les clés partielles $\mathbf{k}_i$ sont toutes égales dès lors que les permutations PC-2 prennent en entrée le même vecteur $\mathbf{x}_1||\mathbf{x}_2$ où $\mathbf{x}_1$ et $\mathbf{x}_2$ sont soit le vecteur tout à 0 soit le vecteur tout à 1.

6. On veut tester la propriété de diffusion sur DES. Choisir deux messages $\mathbf{m}_1$ et $\mathbf{m}_2$ de 64 bits tels que $\mathbf{m}_1$ et $\mathbf{m}_2$ diffèrent seulement sur un bit. Appliquer le chiffrement DES en mode ECB sur $\mathbf{m}_1$ et $\mathbf{m}_2$ avec une même clé. Comparer les résultats.

7. On veut tester la sensibilité de DES aux erreurs de transmission.

   Appliquer le chiffrement DES en mode ECB sur un message $\mathbf{m}$ de 64 bits ; on note $\mathbf{c}$ le chiffré obtenu.

   Déchiffrer un message $\mathbf{c}'$ qui ne diffère de $\mathbf{c}$ que d'un seul bit. Comparer le message déchiffré avec le message $\mathbf{m}$. Répéter l'opération plusieurs fois en changeant le bit modifié. Que peut-on en conclure?

8. Résumer les différentes faiblesses du chiffrement DES mises en lumière depuis le début de l'exercice.

double-DES.

On voudrait étudier si le chiffrement double-DES (2-DES) améliore le niveau de sécurité par rapport au simple DES.

Le protocole 2-DES utilise deux clés secrètes, $\mathbf{k}_1$ et $\mathbf{k}_2$. Pour chiffrer un message $\mathbf{m}$, Alice calcule :

$$\mathbf{c} = Enc_{\mathbf{k}_2}(Enc_{\mathbf{k}_1}(\mathbf{m}))$$

Bob, à son tour, déchiffre $\mathbf{m}$ par l'opération suivante :

$$\mathbf{m} = Dec_{\mathbf{k}_1}(Dec_{\mathbf{k}_2}(\mathbf{c}))$$

9. Chiffrer deux messages $\mathbf{m}_1$ et $\mathbf{m}_2$ par 2-DES pour obtenir deux messages chiffrés $\mathbf{c}_1$ et $\mathbf{c}_2$.

10. On a donc deux couples de messages : $(\mathbf{m}_1, \mathbf{c}_1)$ et $(\mathbf{m}_2, \mathbf{c}_2)$.

    Démontrer sur un exemple que

    $$Enc_{\mathbf{k}_1}(\mathbf{m}_i) = Dec_{\mathbf{k}_2}(\mathbf{c}_i)$$

    pour $i \in \{1, 2\}$.

11. Justifier que le nombre moyen de clés $(\mathbf{k}_1, \mathbf{k}_2)$ tels que $Enc_{\mathbf{k}_1}(\mathbf{m}_1) = Dec_{\mathbf{k}_2}(\mathbf{c}_1)$ est $2^{112-64} = 2^{48}$.

12. Justifier que le nombre moyen de clés $(\mathbf{k}_1, \mathbf{k}_2)$ tels que $Enc_{\mathbf{k}_1}(\mathbf{m}_1) = Dec_{\mathbf{k}_2}(\mathbf{c}_1)$ ET $Enc_{\mathbf{k}_1}(\mathbf{m}_2) = Dec_{\mathbf{k}_2}(\mathbf{c}_2)$ est 1.

13. Proposer une attaque sur le double-DES.

AES et modes de chiffrement.

14. Adapter les questions 6 et 7 au chiffrement AES-128 en mode ECB. Les faiblesses de DES vous semblent-elles corrigées avec AES?

15. Implémenter une fonction de chiffrement qui prend en entrée :

    - `key` : la clé de chiffrement ;
    - `image` : un fichier au format bitmap (`.bmp`) ;
    - `algo` : "DES" ou "AES" ;
    - `mode` : "MODE_ECB", "MODE_CBC" ou "MODE_CTR" ;
    - `iv` : un éventuel vecteur d'initialisation (nécessaire selon le mode de chiffrement considéré).

    et qui retourne un fichier au format bitmap qui représente le contenu chiffré de `image.bmp`. Attention, il est demandé de ne chiffrer que le contenu de l'image ; on ne modifiera donc pas l'entête du fichier bitmap.

16. Comparer les différents modes de chiffrement de DES et AES sur le fichier `image.bmp` (ne pas chiffrer l'entête qui est de 74 octets pour ce fichier). Quel mode vous semble le plus pertinent? Justifier.