Sécurité Appliquée : chiffrement symétrique, TP

Jean-François COUCHOT
couchot[arobase]femto-st[point]fr

15 février 2024

1 AES par la pratique

On va utiliser la bibliothèque `pycryptodome` cryptographique pour chiffrer avec AES.

Exercice 1.1 (Exploitation d'un code pour chiffrer une image selon AES, CBC et CTR).

```
1  from Crypto.Random import get_random_bytes
2  from Crypto.Cipher import AES
3  from Crypto.Protocol.KDF import scrypt
4
5  def AESnew(pk,mode,iv=None,nonce=None):
6      match mode :
7          case AES.MODE_ECB :
8            return AES.new(pk,AES.MODE_ECB)
9          case AES.MODE_CBC :
10           return AES.new(pk,AES.MODE_CBC,iv=iv)
11         case AES.MODE_CTR :
12           return AES.new(pk,AES.MODE_CTR,nonce=nonce)
13
14 def encrypt(plain_text, password, mode):
15     salt = get_random_bytes(AES.block_size)
16     private_key = scrypt(password.encode(), salt, 16, N=2**14, r=8, p=1)
17     iv = get_random_bytes(AES.block_size)
18     nonce = get_random_bytes(int(AES.block_size/2))
19     cipher_config = AESnew(private_key,mode,iv,nonce)
20     cipher_text= cipher_config.encrypt(plain_text.encode())
21     return {'cipher_text': cipher_text,'salt': salt,'iv':iv,'nonce':nonce}
22
23 def decrypt(enc_dict, password,mode):
24     cipher_text,salt,iv,nonce=enc_dict['cipher_text'],enc_dict['salt'],enc_dict['iv'],enc_dict['nonce']
25     private_key = scrypt(password.encode(), salt, 16, N=2**14, r=8, p=1)
26     cipher_config = AESnew(private_key,mode,iv,nonce)
27     return cipher_config.decrypt(cipher_text)
28
29 def main():
30     password = input("Password: ")
31     mode = AES.MODE_ECB
32
33     encrypted = encrypt("Secrete Message.", password,mode)
34     print(encrypted)
35
36     decrypted = decrypt(encrypted, password,mode)
37     print(decrypted.decode())
38 main()
```

1. *Comprendre completement le programme précédent :*
   (a) *Objectifs des lignes 15 et 16 ? Puis 17 et 18 ?*
   (b) *Pourquoi invoque-t-on à la ligne 18 la fonction* `AESnew` *définie à la ligne 5 ? Tous les paramètres construits et passés sont-ils toujours utilisés ?*
   (c) *Que contient la variable* `cipher_text` *affectée à la ligne 20 ?*
   (d) *Que retourne-t-on à la ligne 21 ?*
   (e) *Que contiennent les variables affectées à la ligne 24 ?*
   (f) *Qu'est-ce qui est retourné à la ligne 27 ?*
   (g) *Que contient à variable* `encrypted` *affectée à la ligne 33 ?*
2. *Exécutez le programme (après avoir installé* `pycryptodome` *éventuellement).*

3. *Chiffrez et déchiffrez une chaîne de caractère de votre choix avec AES, CBC, CTR.*
4. *Jouez avec la taille des messages à chiffrer et constatez que certains modes sont robustes à la taille du message, d'autre pas.*
   (a) *Exploiter les deux fonctions* `pad` *et* `unpad` *du package* `Crypto.Util.Padding` *pour que tous les modes deviennent robustes à la taille du message.*
5. *Comprendre le code suivantqui permet de récupérer les valeurs des pixels d'une images sous la forme d'octets.*

```
from PIL import Image
import random as rd

im = Image.open("imgISIFC.png")
message = im.tobytes()
imb= Image.frombytes(im.mode,im.size,message)
imb.show()
```

   (a) *En exploitant ce code suivant, chiffrez une image avec AES et ECB.*
   (b) *Affichez l'image chiffrée. Que remarquez-vous visuellement ?*
   (c) *Déchifffez l'image chiffrée et constatez que tout s'est bien passé.*
6. *Chiffrez une image avec CBC, CTR.*
   (a) *Affichez l'image chiffrée, puis déchiffrez l'image chiffrée et constatez que tout s'est bien passé.*