PRÁCTICA 2: Criptografía (Parte 2)

Seguridad de la Información
Curso 2018-2019

Lenguajes y Ciencias de la Computación.
E.T.S.I. Informática, Universidad de Málaga

RELACIÓN DE EJERCICIOS:

1. (**5 puntos**) El código Python descrito en el apéndice muestra como se cifra y se descifra un texto utilizando para ello DES en modo CBC. Utilizando como base ese código, crear una clase llamada DES_CIPHER que tenga los siguientes métodos, y que ejecute correctamente el código de prueba:

```
class DES_CIPHER:

    BLOCK_SIZE_DES = 8 # DES: Bloque de 64 bits

    def __init__(self, key):
        """Inicializa las variables locales"""

    def cifrar(self, cadena, IV):
        """Cifra el parámetro cadena (de tipo String) con una IV específica, y
           devuelve el texto cifrado binario"""

    def descifrar(self, cifrado, IV):
        """Descrifra el parámetro cifrado (de tipo binario) con una IV específica, y
           devuelve la cadena en claro de tipo String"""

key = get_random_bytes(8) # Clave aleatoria de 64 bits
IV = get_random_bytes(8)  # IV aleatorio de 64 bits
datos = "Hola Mundo con DES en modo CBC"
d = DES_CIPHER(key)
cifrado = d.cifrar(datos, IV)
descifrado = d.descifrar(cifrado, IV)
```

2. (**3 puntos**) Se pide crear una clase AES_CIPHER, la cual tenga los mismos métodos que la clase DES_CIPHER, pero que utilice el algoritmo AES para cifrar y descifrar. Para este ejercicio, es necesario tener en cuenta tanto el tamaño de claves como el tamaño de bloques del algoritmo AES.

3. (**2 puntos**) Se pide cifrar y descifrar el mensaje “Hola Amigos de Seguridad” usando la clase AES_CIPHER y los siguientes modos de operación:

    a. ECB
    b. CTR
    c. OFB
    d. CFB

APÉNDICE: Código de ejecución de DES en modo CBC

```
from Crypto.Random import get_random_bytes
from Crypto.Cipher import DES, AES
from Crypto.Util.Padding import pad,unpad
from Crypto.Util import Counter
import base64

# Datos necesarios
key = get_random_bytes(8) # Clave aleatoria de 64 bits
IV = get_random_bytes(8)  # IV aleatorio de 64 bits para CBC
BLOCK_SIZE_DES = 8 # Bloque de 64 bits
data = "Hola Mundo con DES en modo CBC".encode("utf-8") # Datos a cifrar
print(data)

# CRIFRADO ###############################################################################

# Creamos un mecanismo de cifrado DES en modo CBC con un vector de inicialización IV
cipher = DES.new(key, DES.MODE_CBC, IV)

# Ciframos, haciendo que la variable “data” sea múltiplo del tamaño de bloque
ciphertext = cipher.encrypt(pad(data,BLOCK_SIZE_DES))

# Mostramos el cifrado por pantalla en modo binario y en modo base 64
print(ciphertext)
encoded_ciphertext = base64.b64encode(ciphertext)
print(encoded_ciphertext)

# DESCRIFRADO ############################################################################

# Creamos un mecanismo de (des)cifrado DES en modo CBC con un vector de
inicialización IV para CBC
# Ambos, cifrado y descifrado, se crean de la misma forma
decipher_des = DES.new(key, DES.MODE_CBC, IV)

# Desciframos, eliminamos el padding, y recuperamos la cadena
new_data = unpad(decipher_des.decrypt(ciphertext), BLOCK_SIZE_DES).decode("utf-8",
"ignore")

# Imprimimos los datos descifrados
print(new_data)
```