PRÁCTICA 2: Criptografía (Parte 2)

Seguridad de la Información

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

RELACIÓN DE EJERCICIOS:

1. El código Python descrito en el apéndice muestra como se cifra y descifra un texto utilizando DES en modo CBC. Crear un código Python que cifre y descifre tanto el texto *Hola amig**o**s de la seguridad* como el texto *Hola amig**a**s de la seguridad* utilizando AES en modo CBC usando la misma clave e IV.

   Si se observa los textos cifrado, es posible ver que ese cambio de una "o" por una "a" (amigos → amigas) impacta en ambos textos, ¿a qué se debe ese cambio?

2. Se pide cifrar y descifrar en AES el mensaje "Hola Amigos de Seguridad" utilizando los siguientes modos de operación:

   a. ECB.
   b. CTR, pasando por parámetro únicamente el campo nonce (valor aleatorio, de tamaño (tamaño de bloque / 2)).
   c. OFB, pasando por parámetro únicamente un valor IV aleatorio.
   d. CFB, pasando por parámetro únicamente un valor IV aleatorio.
   e. GCM, pasando como parámetros el campo nonce (valor aleatorio del mismo tamaño de bloque) y mac_len (16).

   Para más información:
   https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

3. Utilizando como base el código del apartado 1 (AES en modo CBC), crear una clase llamada AES_CIPHER_CBC que tenga los siguientes métodos, y que ejecute correctamente el código de prueba:

```
class AES_CIPHER_CBC:

    BLOCK_SIZE_AES = 16 # AES: Bloque de 128 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):
        """Descifra el parámetro cifrado (de tipo binario) con una IV específica, y
           devuelve la cadena en claro de tipo String"""

key = get_random_bytes(16) # Clave aleatoria de 128 bits
IV = get_random_bytes(16)  # IV aleatorio de 128 bits
datos = "Hola Mundo con AES en modo CBC"
d = AES_CIPHER_CBC(key)
cifrado = d.cifrar(datos, IV)
descifrado = d.descifrar(cifrado, IV)
```

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 amigos de la seguridad".encode("utf-8") # Datos a cifrar
print(data)

# CIFRADO ##############################################################################

# 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)

# DESCIFRADO ###########################################################################

# 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)
```