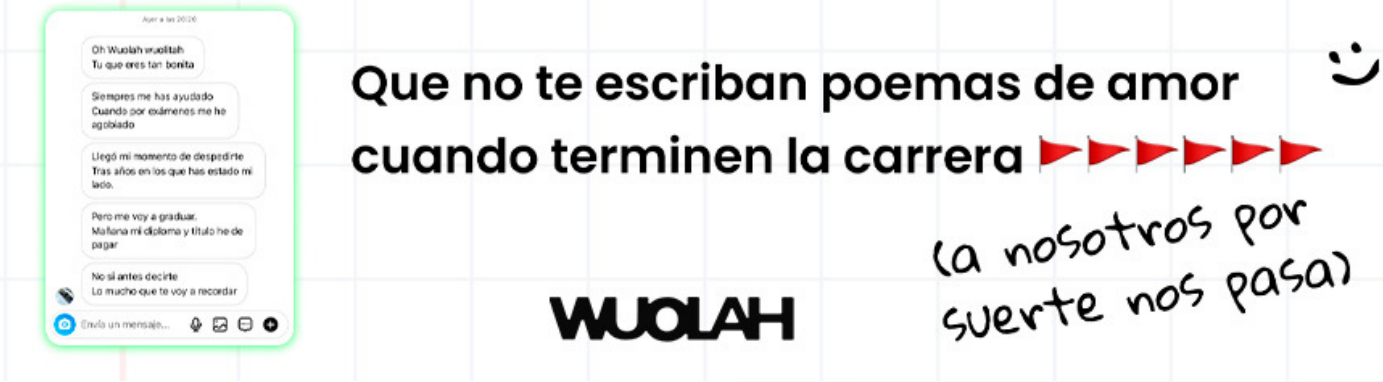
Practica-Criptografia-Simetrica.pdf

jmp__0807

Seguridad de la Información

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga



Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

PRÁCTICA 1: Criptografía

Seguridad en 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 cómo se cifra y se 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 cifrados, 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).

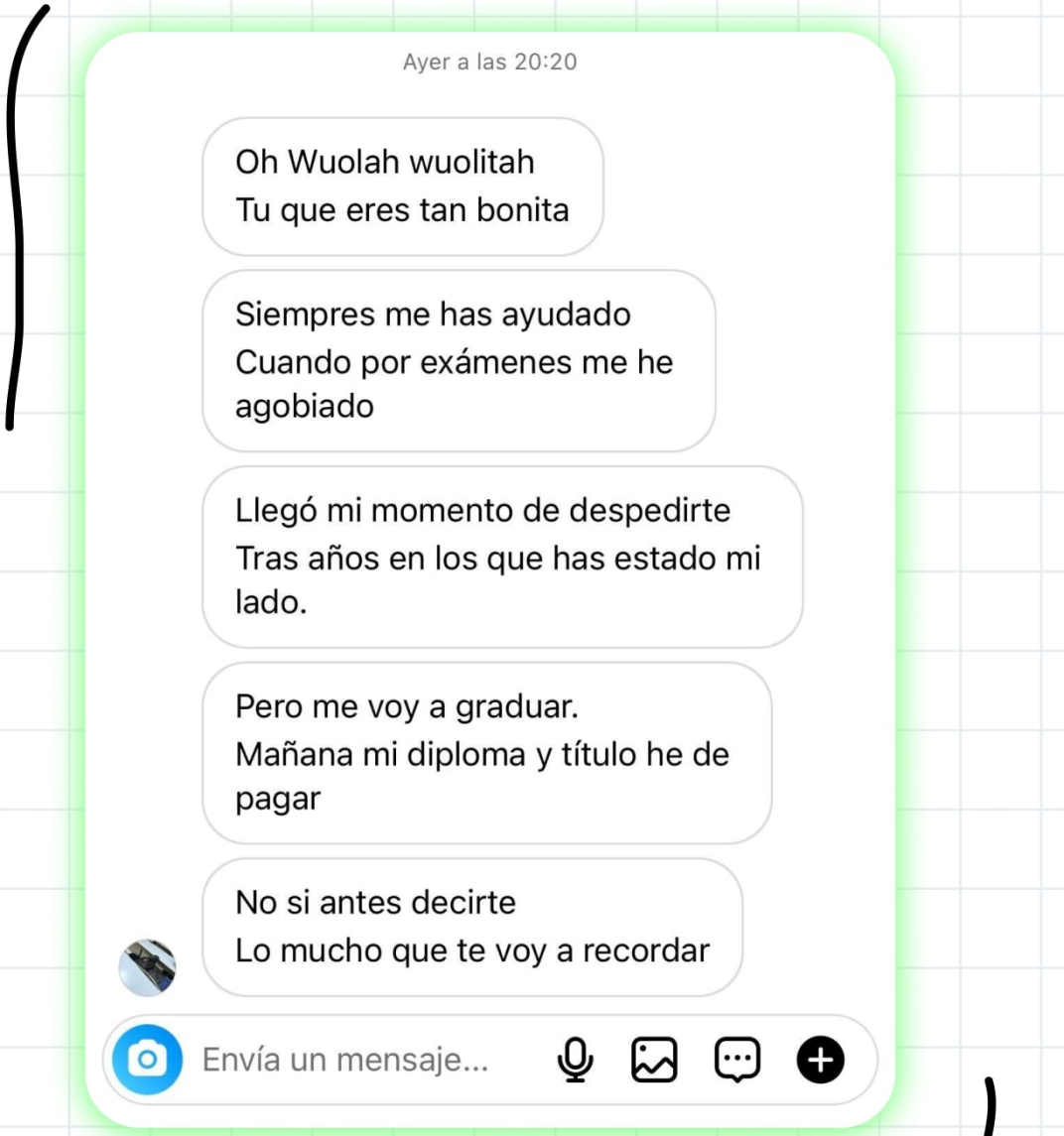
   Para más información:
   https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

3. (**OPCIONAL**) 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)
```

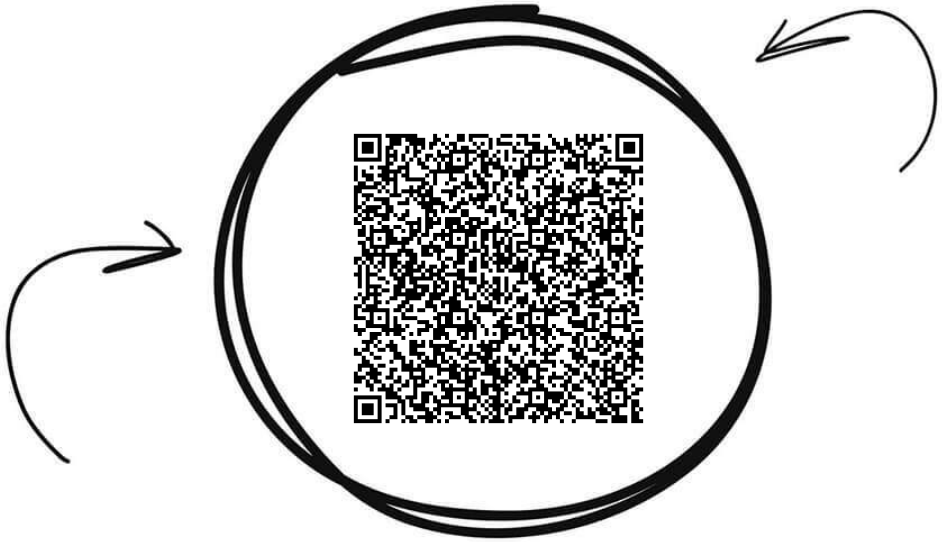
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Seguridad de la Información



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

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))
print(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)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej1.py

```python
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

key = get_random_bytes(16)
IV = get_random_bytes(16)
BLOCK_SIZE_AES = 16
text = "Hola amigos de la seguridad".encode("utf-8")
print(text)
mac_size = 16

aes_cipher = AES.new(key,AES.MODE_CBC,IV)
cipherText = aes_cipher.encrypt(pad(text,BLOCK_SIZE_AES))
print(cipherText)

decipher_aes = AES.new(key,AES.MODE_CBC,IV)
textResolved = unpad(decipher_aes.decrypt(cipherText),BLOCK_SIZE_AES).decode(
   "utf-8","ignore")
print(textResolved)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej2a.py

```python
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

key = get_random_bytes(16)
BLOCK_SIZE_AES = 16
text = "Hola Amigos de Seguridad".encode("utf-8")
print(text)
mac_size = 16

# En el modo ECB (Electronic Codebook) de cifrado simétrico, no se utiliza
# un Vector de Inicialización (IV) porque este modo de cifrado no tiene una etapa de retroalimentación

aes_cipher = AES.new(key,AES.MODE_ECB)
cipherText = aes_cipher.encrypt(pad(text,BLOCK_SIZE_AES))
print(cipherText)

decipher_aes = AES.new(key,AES.MODE_ECB)
textResolved = unpad(decipher_aes.decrypt(cipherText),BLOCK_SIZE_AES).decode("utf-8","ignore")
print(textResolved)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej2b.py

```python
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

key = get_random_bytes(16)
nonce = get_random_bytes(8) # contador de 64 bits empezando desde 0
IV = get_random_bytes(16)
BLOCK_SIZE_AES = 16
text = "Hola Amigos de Seguridad".encode("utf-8")
print(text)
mac_size = 16

aes_cipher = AES.new(key,AES.MODE_CTR, nonce = nonce)
cipherText = aes_cipher.encrypt(pad(text,BLOCK_SIZE_AES))
print(cipherText)

decipher_aes = AES.new(key,AES.MODE_CTR, nonce = nonce)
textResolved = unpad(decipher_aes.decrypt(cipherText),BLOCK_SIZE_AES).decode(
    "utf-8","ignore")
print(textResolved)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej2c.py

```python
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

key = get_random_bytes(16)
IV = get_random_bytes(16)
BLOCK_SIZE_AES = 16
text = "Hola Amigos de Seguridad".encode("utf-8")
print(text)
mac_size = 16

aes_cipher = AES.new(key,AES.MODE_OFB,IV)
cipherText = aes_cipher.encrypt(pad(text,BLOCK_SIZE_AES))
print(cipherText)

decipher_aes = AES.new(key,AES.MODE_OFB,IV)
textResolved = unpad(decipher_aes.decrypt(cipherText),BLOCK_SIZE_AES).decode(
   "utf-8","ignore")
print(textResolved)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej2d.py

```python
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

key = get_random_bytes(16)
IV = get_random_bytes(16)
BLOCK_SIZE_AES = 16
text = "Hola Amigos de Seguridad".encode("utf-8")
print(text)
mac_size = 16

aes_cipher = AES.new(key,AES.MODE_CFB,IV)
cipherText = aes_cipher.encrypt(pad(text,BLOCK_SIZE_AES))
print(cipherText)

decipher_aes = AES.new(key,AES.MODE_CFB,IV)
textResolved = unpad(decipher_aes.decrypt(cipherText),BLOCK_SIZE_AES).decode("utf-8","ignore")
print(textResolved)
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Criptografia Simetrica/Ej2e.py

```python
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

key = get_random_bytes(16) # Clave aleatoria de 128 bits
IV = get_random_bytes(16//2) # Nonce aleatorio de 64 bits para GCM
BLOCK_SIZE_AES = 16 # Bloque de 128 bits
data = "Hola Amigos de Seguridad".encode("utf-8") # Datos a cifrar
print(data)
mac_size = 16
aes_cipher = AES.new(key, AES.MODE_GCM, nonce=IV, mac_len=mac_size)
ciphertext, mac_cifrado = aes_cipher.encrypt_and_digest(pad(data,
BLOCK_SIZE_AES))
print(ciphertext)

try:
    aes_decipher = AES.new(key, AES.MODE_GCM, nonce=IV)
    text = unpad(aes_decipher.decrypt_and_verify(ciphertext,mac_cifrado),
BLOCK_SIZE_AES).decode("utf-8", "ignore")
    print(text)
except (ValueError,KeyError) as e:
    print("ERROR")
```

File - /home/javimp2003/IdeaProjects/UMAPracs/Cifrado_Descifrado_unsolved.py

```python
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

class AES_CIPHER_CBC:

    BLOCK_SIZE_AES = 16 # AES: Bloque de 128 bits

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

    def cifrar(self, cadena, IV):
        # AES en modo CBC
        """Cifra el parámetro cadena (de tipo String) con una IV específica, y
           devuelve el texto cifrado binario"""
        cadena = cadena.encode("utf-8")
        cypher = AES.new(self.key, AES.MODE_CBC,IV)

        cypherText = cypher.encrypt(pad(cadena,self.BLOCK_SIZE_AES))
        return cypherText

    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"""

        decypherAes = AES.new(self.key, AES.MODE_CBC, IV)
        textResolved = unpad(decypherAes.decrypt(cifrado),self.BLOCK_SIZE_AES).decode("utf-8","ignore")
        return textResolved

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)
print(f"Texto sin cifrar: {datos}\nTexto cifrado: {cifrado}\nTexto descifrado: {descifrado}")

##############################
##############################
##############################

"""
# 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))
```

```
59
60 # Mostramos el cifrado por pantalla en modo binario
61 print(ciphertext)
62
63 # DESCIFRADO
    ##########################################################################
64
65 # Creamos un mecanismo de (des)cifrado DES en modo CBC con un vector de
   inicialización IV para CBC
66 # Ambos, cifrado y descifrado, se crean de la misma forma
67 decipher_des = DES.new(key, DES.MODE_CBC, IV)
68
69 # Desciframos, eliminamos el padding, y recuperamos la cadena
70 new_data = unpad(decipher_des.decrypt(ciphertext), BLOCK_SIZE_DES).decode("
   utf-8", "ignore")
71
72 # Imprimimos los datos descifrados
73 print(new_data)
74 """
```