

# RESUMEN-PARA-EXAMEN-PRACTICAS.pdf



**miau\_33**



**Seguridad de la Información**



**3º Grado en Ingeniería Informática**



**Escuela Técnica Superior de Ingeniería Informática  
Universidad de Málaga**



La mejor escuela de negocios en energía, sostenibilidad y medio ambiente de España.

Más información  
[www.eoi.es](http://www.eoi.es)

Formamos  
**talento** para un futuro  
**Sostenible**



**100% Empleabilidad**



**Modalidad: Presencial u online**



**Programa de Becas,  
Bonificaciones y Descuentos**



**X**  
**SIBUYA**  
URBANSUSHIBAR

Hay rollos y rollos... Me vas a comparar tu rollo de ayer  
con un Hot Roll, claramente el sushi está más bueno

HAZ CLIC PARA HACER TU

RESERVA



## RESUMEN PARA EXAMEN PRÁCTICAS

### PRÁCTICA 4

#### Explicación funciones\_aes.py que usamos:

```
crear_AESKey()
```

me devuelve una clave

- GCM
  - Cifrar

```
iniciarAES_GCM(key_16)
```

me devuelve un aes\_cifrado que usaremos en el siguiente paso

```
cifrarAES_GCM(aes_cifrado, datos)
```

me devuelve el mac, un nonce y datos\_cifrados (todo esto se necesitará para descifrar

- Descifrar

```
descifrarAES_GCM(key_16, nonce_16, datos, mac)
```

me devuelve datos\_claro

- CTR
  - Cifrar

```
iniciarAES_CTR_cifrado(key_16)
```

me devuelve un nonce y aes\_cifrado (el nonce lo necesitaremos para iniciar el descifrado así que habrá que enviarlo al receptor)

```
cifrarAES_CTR(aes_cifrado, datos)
```

me devuelve datos\_cifrado

- Descifrar

```
iniciarAES_CTR_descifrado(key_16, nonce_16_ini)
```

me devuelve aes\_cifrado

```
descifrarAES_CTR(aes_descifrado, datos)
```

me devuelve datos\_claro

#### Servidor

##### Ejemplo donde Alice recibe un mensaje de Bob

Necesitamos importar

```
from socket_class import SOCKET_SIMPLE_TCP
```

Alice

creamos socket de escucha

```
print("Esperando a Bob...")
socket =
SOCKET_SIMPLE_TCP('127.0.0.1',
5551)
socket.escuchar()
```

recibe el mensaje

```
cifrado = socket.recibir()
cifrado_mac = socket.recibir()
cifrado_nonce = socket.recibir()
```

cerramos el socket

```
socket.cerrar()
```

Bob

creamos socket de conexión con Alice (5551)

```
print("Creando conexion con
ALICE...")
socketB =
SOCKET_SIMPLE_TCP('127.0.0.1',
5551)
socketB.conectar()
```

le envía los datos

```
socketB.enviar(cifrado)
socketB.enviar(mac)
socketB.enviar(nonce)
```

WUOLAH

```
socketB.close()
```

## HMAC

Alice crea el HMAC

```
hsend = HMAC.new(K2, msg=apellido.encode("utf-8"), digestmod=SHA256)
mac = hsend.digest()
```

Bob verifica el HMAC

```
hmacB = HMAC.new(K2, digestmod=SHA256)
hmacB.update(mensaje_claro_json.encode("utf-8"))
try:
    hmacB.hexverify(mac)
    print("Mensaje correcto")
except ValueError:
    print("Mensaje manipulado")
    socket.cerrar()
    exit()
```

## JSON (poder enviar mensajes concatenados)

Alice quiere concatenar "Alice" y t\_n\_origen

```
msg = []
msg.append("Alice")
msg.append(t_n_origen.hex())
json_A = json.dumps(msg)
```

Alice cifra el `json.encode("utf-8")` y se lo envía a Bob

Bob lo recibe y descifra

```
# Decodifica el contenido: Bob, Nb
json_Recibido = datos_descifrado.decode("utf-8", "ignore")
print("A->T (descifrado): " + json_Recibido)
msg = json.loads(json_Recibido)

# Extraigo el contenido
t_alice, t_na = msg
t_na = bytearray.fromhex(t_na)
```

## FICHEROS (leer y escribir en ficheros)

Crear clave KAT y guardar en un fichero

```
KAT = funciones_aes.crear_AESKey()
fichero= open("KAT.bin", "wb")
fichero.write(KAT)
fichero.close()
```

Leer clave KAT

```
KAT = open("KAT.bin", "rb").read()
```

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en [ing.es](https://www.ing.es)

Tener todas tus compras, pagos, suscripciones **controladas desde la misma app** es...

*Very demure.*

*Very mindful.*

**Very Cuenta NoCuenta de ING.**

**¡Descúbrela!**



do your thing



## Seguridad de la Información



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



**Banco de apuntes de la**

**WUOLAH**

**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



## PRÁCTICA 2

### Datos necesarios

```
key = get_random_bytes(16) # Clave aleatoria de 64 bits
IV = get_random_bytes(16) # IV aleatorio de 64 bits para CBC
nonce = get_random_bytes(8)
BLOCK_SIZE_DES = 16 # Bloque de 64 bits
data = "Hola amigos de la seguridad".encode("utf-8") # Datos a cifrar
```

### ECB

#### Cifrado

```
cipher = AES.new(key, AES.MODE_ECB)
ciphertext = cipher.encrypt(pad(data, BLOCK_SIZE_DES))
```

#### Descifrado

```
decipher_des = AES.new(key, AES.MODE_ECB)
new_data = unpad(decipher_des.decrypt(ciphertext),
BLOCK_SIZE_DES).decode("utf-8", "ignore")
```

### CTR (hay q pasarle el nonce)

#### Cifrado

```
cipher = AES.new(key, AES.MODE_CTR, nonce=nonce)
ciphertext = cipher.encrypt(pad(data, BLOCK_SIZE_DES))
```

#### Descifrado

```
decipher_des = AES.new(key, AES.MODE_CTR, nonce=nonce)
new_data = unpad(decipher_des.decrypt(ciphertext),
BLOCK_SIZE_DES).decode("utf-8", "ignore")
```

### OFB (hay q pasarle el IV)

#### Cifrado

```
cipher = AES.new(key, AES.MODE_OFB, IV)
ciphertext = cipher.encrypt(pad(data, BLOCK_SIZE_DES))
```

#### Descifrado

```
decipher_des = AES.new(key, AES.MODE_OFB, IV)
new_data = unpad(decipher_des.decrypt(ciphertext),
BLOCK_SIZE_DES).decode("utf-8", "ignore")
```

### CFB (hay q pasarle el IV)

#### Cifrado

```
cipher = AES.new(key, AES.MODE_CFB, IV)
ciphertext = cipher.encrypt(pad(data, BLOCK_SIZE_DES))
```

#### Descifrado

```
decipher_des = AES.new(key, AES.MODE_CFB, IV)
new_data = unpad(decipher_des.decrypt(ciphertext),
BLOCK_SIZE_DES).decode("utf-8", "ignore")
```





**X**  
**SIBUYA**  
URBANSUSHIBAR

Hay rollos y rollos... Me vas a comparar tu rollo de ayer  
con un Hot Roll, claramente el sushi está más bueno

HAZ CLIC PARA HACER TU

**RESERVA**



GCM (hay q pasarle el nonce y mac\_len)

Datos necesarios

```
key = get_random_bytes(16) # Clave aleatoria de 64 bits
IV = get_random_bytes(16//2) # IV aleatorio de 64 bits para CBC
BLOCK_SIZE_DES = 16 # Bloque de 64 bits
data = "Hola amigos de la seguridad".encode("utf-8") # Datos a cifra
mac_size=16
```

Cifrado

```
cipher = AES.new(key, AES.MODE_GCM, nonce=IV, mac_len=mac_size)
ciphertext, mac_cifrado =
cipher.encrypt_and_digest(pad(data,BLOCK_SIZE_DES))
```

Descifrado

```
decipher_des = AES.new(key, AES.MODE_GCM, nonce=IV, mac_len=mac_size)
new_data =
unpad(decipher_des.decrypt_and_verify(ciphertext,mac_cifrado),
BLOCK_SIZE_DES).decode("utf-8", "ignore")
```

**WUOLAH**