

# ***INFORME PRÁCTICA DE CRIPTOGRAFÍA***



**1. Tenemos un sistema que usa claves de 16 bytes. Por razones de seguridad vamos a proteger la clave de tal forma que ninguna persona tenga acceso directamente a la clave. Por ello, vamos a realizar un proceso de disociación de la misma, en el cuál tendremos, una clave fija en código, la cual, sólo el desarrollador tendrá acceso, y otra parte en un fichero de propiedades que rellenará el Key Manager. La clave final se generará por código, realizando un XOR entre la que se encuentra en el properties y en el código.**

**La clave fija en código es A1EF2ABFE1AAEEFF, mientras que en desarrollo sabemos que la clave final (en memoria) es B1AA12BA21AABB12. ¿Qué valor ha puesto el Key Manager en properties para forzar dicha clave final?**

El valor que ha puesto el Key Manager en properties para forzar la clave final es

```
1 #XOR de los datos binarios
2 #Creamos la funci'on XOR de dos datos>
3 def xor(binary_data_1, binary_data_2):
4     return bytes([b1^b2 for b1, b2 in zip(binary_data_1, binary_data_2)])
5     #Los datos deben pasar a leerse en formato bytes.
6
7
8
9 if __name__ == "__main__":
10     m = bytes("keepcoding", "utf-8")
11     k = bytes("1234567891", "utf-8")
12     print("El resultado en binario es: ", xor(m,k))
13     print("El resultado en hexadecimal es: ", xor(m,k).hex())
```

```
[Running] python -u "/home/kali/Desktop/ejercicospractica/#XOR de los datos binarios.py"
El resultado en binario es: b'ZWVDVYSQWV'
El resultado en hexadecimal es: 5a575644565953515756

[Done] exited with code=0 in 0.091 seconds
```

```
1 #XOR de los datos binarios
2 #Creamos la funci'on XOR de dos datos>
3 def xor(binary_data_1, binary_data_2):
4     return bytes([b1^b2 for b1, b2 in zip(binary_data_1, binary_data_2)])
5     #Los datos deben pasar a leerse en formato bytes.
6
7
8
9 if __name__ == "__main__":
10     m = bytes("A1EF2ABFE1AAEEFF", "utf-8")
11     k = bytes("B1AA12BA21AABB12", "utf-8")
12     print("El resultado en binario es: ", xor(m,k))
13     print("El resultado en hexadecimal es: ", xor(m,k).hex())
```

```
[Running] python -u "/home/kali/Desktop/ejerciciospractica/#XOR de los datos binarios.py"
El resultado en binario es: b'\x03\x00\x04\x07\x03s\x00\x07w\x00\x00\x00\x07\x07wt'
El resultado en hexadecimal es: 03000407037300077700000007077774

[Done] exited with code=0 in 0.043 seconds
```

```
1 from operator import xor
2 from multiplicar import *
3 def xor_data(binary_data_1, binary_data_2):
4     return bytes([b1 ^ b2 for b1, b2 in zip(binary_data_1, binary_data_2)])
5
6 m1 = bytes('España','utf-8')
7 m = bytes("keepcoding==",'utf-8')
8 k = bytes("abcdefghijklddd",'utf-8')
9
10
11
12 print(xor_data(m,k))
13 print(xor_data(m,k).hex())
14 print(xor_data(k,m).hex())
15
16 #K1 (key manager) ^ K2 (desarrollador) = K (keyStore)
17
18
19
20 K1= bytes.fromhex('A1EF2ABFE1AAEEFF')
21 K2= bytes.fromhex('B1AA12BA21AABB12')
22
23
24 K= xor_data(K1,K2)
25 K_hex = xor_data(K1,K2).hex()
26
27 print("K=K1^K2: ", K_hex)
28
29 K22 = xor_data(K,K1)
30
31 print("K2=K3 ^K1: ", K22.hex())
32
33 K1= xor_data(K2,K1)
34
35 print("K1=K2 ^ K: ", K1.hex())
36
```

```
[Running] python -u "/home/vant/Escritorio/ejerciciospractica/multiplicar.py"
b'\n\x07\x06\x14\x06\t\x03\x01\x07\rYYY'
0a07061406090301070d595959
0a07061406090301070d595959
K=K1^K2: 10453805c00055ed
K2=K3 ^K1: b1aa12ba21aabb12
K1=K2 ^ K: 10453805c00055ed
b'\n\x07\x06\x14\x06\t\x03\x01\x07\rYYY'
0a07061406090301070d595959
0a07061406090301070d595959
K=K1^K2: 10453805c00055ed
K2=K3 ^K1: b1aa12ba21aabb12
K1=K2 ^ K: 10453805c00055ed

[Done] exited with code=0 in 0.067 seconds
```

**La clave fija, recordemos es A1EF2ABFE1AAEEFF, mientras que en producción sabemos que la parte dinámica que se modifica en los ficheros de propiedades es B98A15BA31AEBB22. ¿Qué clave será con la que se trabaje en memoria?**

```
1  #XOR de los datos binarios
2  #Creamos la función XOR de dos datos>
3  def xor(binary_data_1, binary_data_2):
4      return bytes([b1^b2 for b1, b2 in zip(binary_data_1, binary_data_2)])
5      #Los datos deben pasar a leerse en formato bytes.
6
7
8
9  if __name__ == "__main__":
10     m = bytes("A1EF2ABFE1AAEEFF", "utf-8")
11     k = bytes("B98A15BA31AEBB22", "utf-8")
12     print("El resultado en binario es: ", xor(m,k))
13     print("El resultado en hexadecimal es: ", xor(m,k).hex())
```

```
[Running] python -u "/home/kali/Desktop/ejerciciospractica/#XOR de los datos binarios.py"
El resultado en binario es:  b'\x03\x08}\x07\x03t\x00\x07v\x00\x00\x04\x07\x07tt'
El resultado en hexadecimal es:  03087d07037400077600000407077474

[Done] exited with code=0 in 0.064 seconds
```

**2. Dada la clave con etiqueta “cifrado-sim-aes-256” que contiene el keystore. El iv estará compuesto por ceros binarios (“00”). Se requiere obtener el dato en claro correspondiente al siguiente dato cifrado:**

**zcFJxR1fzaBj+gVWFRAah1N2wv+G2P01ifrKejlCaGpQkPnZMiexn3WXIGYX5WnNlosyKfkNKG9GGSgG1awaZg==**

**Para este caso, se ha usado un AES/CBC/PKCS7.**

**Si lo desciframos, ¿qué obtenemos? ¿Qué ocurre si decidimos cambiar el padding a x923 en el descifrado? ¿Cuánto padding se ha añadido en el cifrado? Como truco, estudiar el resultado en hexadecimal, cuando lo imprimas en consola introducir un string para diferenciar el final, como “|\*\*\*\*\*|”.**

**Se valorará positivamente, obtener el dato de la clave desde el keystore mediante codificación en Python (u otro lenguaje).**

3. Se requiere cifrar el texto “Este curso es de lo mejor que podemos encontrar en el mercado”. La clave para ello, tiene la etiqueta en el keyStore “cifrado-sim-chacha-256”. El nonce “9Yccn/f5nJJhAt2S”.  
979df30474898787a45605ccb9b36d33b780d03cab81719d52383480dc3120

¿Cómo podríamos mejorar de forma sencilla el sistema, de tal forma, que no sólo garanticemos la confidencialidad sino, además, la integridad del mismo? Demuestra, tu propuesta por código, así como añade los datos necesarios para evaluar tu solución.

```
home > kali > Desktop > ejerciciospractica > EJERCICO3.py > ...
1  import json
2  from base64 import b64encode, b64decode
3  from Crypto.Cipher import AES
4  from Crypto.Util.Padding import pad, unpad
5  from Crypto.Random import get_random_bytes
6
7  #Cifrado
8  textoPlano_bytes = bytes('Este curso es de lo mejor que podemos encontrar en el mercado', 'UTF-8')
9
10 #clave = get_random_bytes(16)
11 clave = bytes.fromhex('979df30474898787a45605ccb9b36d33b780d03cab81719d52383480dc3120')
12 nonce = bytes.fromhex('f5871c9ff7f99c926102dd92')
13 cipher = AES.new(clave, AES.MODE_CTR, nonce=nonce)
14 texto_cifrado_bytes = cipher.encrypt(textoPlano_bytes)
15
16 nonce_b64 = b64encode(cipher.nonce).decode('utf-8')
17 texto_cifrado_b64 = b64encode(texto_cifrado_bytes).decode('utf-8')
18 mensaje_json = json.dumps({'nonce':nonce_b64, 'texto cifrado':texto_cifrado_b64})
19 print(mensaje_json)
20
21 #Descifrado
22
23 try:
24     b64 = json.loads(mensaje_json)
25     nonce_desc_bytes = b64decode(b64['nonce'])
26     texto_cifrado_bytes = b64decode(b64['texto cifrado'])
27     cipher = AES.new(clave, AES.MODE_CTR, nonce=nonce_desc_bytes)
28     mensaje_des_bytes = cipher.decrypt(texto_cifrado_bytes)
29     print("El texto en claro es: ", mensaje_des_bytes.decode("utf-8"))
30
31 except (ValueError, KeyError) as error:
32     print('Problemas para descifrar...')
33     print("El motivo del error es: ", error)
```

```
[Running] python -u "/home/kali/Desktop/ejerciciospractica/EJERCICO3.py"
{"nonce": "9Yccn/f5nJJhAt2S", "texto cifrado": "nNEpHk9g0Mh55iENpj2SjyLkdTda/LEgKehRG268VwCTG3FYxxQ92a0dwQPouppMcEHafItHyHsnCybDkA==" }
El texto en claro es: Este curso es de lo mejor que podemos encontrar en el mercado
[Done] exited with code=0 in 0.13 seconds
```

**eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoiRmVsZXBlIFJvZlJcdT  
AwZWVndWV6liwicm9sljoiaXNOb3JtYWwifQ.-  
KiAA8cjkamjwRUiNVHgGeJU8k2wiErdxQP\_iFXumM8**

El algoritmo realizado en este ejercicio es el SHA2-256, tipo JWT.

En el body del jwt se muestra los siguientes datos:

```
usuario: Felipe Rodríguez
rol: isNormal
```

Encoded

PASTE A TOKEN HERE

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmliIjoiaRmVsaXB1IFJvZHLDrWd1ZXoIlCjYb2wiOiJpc05vcml1bCJ9.vbZwgGWgbltdtCSutI1JwrtdZU1cthcovc0dXb99s8

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

```
{
  "usuario": "Felipe Rodriguez",
  "rol": "isNormal"
}
```

VERIFY SIGNATURE

HMACSHA256(  
base64UrlEncode(header) + "." +  
base64UrlEncode(payload),  

KeepCoding

)

☐ secret base64 encoded

Last build: 20 days ago
Options
About / Support

Recipe	Input
JWT Decode	<pre> eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoIcmVsaXB1IFJvZHLDrWd1Z&gt; oiLCJyb2wiOiJpc05vcm1hbCJ9.vbZwgGwgbldtdCSutI1JwrtdZU1cthcovbC0dXb99s8 </pre>

time: 9ms  
length: 60  
lines: 4

Output

```

{
  "usuario": "Felipe Rodríguez",
  "rol": "isNormal"
}

```

**Un hacker está enviando a nuestro sistema el siguiente jwt:**

**eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoIcmVsaXB1IFJvZHLJcdT  
AwZWRndWV6liwicm9sljoiaXNBZG1pbiJ9.-  
KiAA8ckamjwRUiNVHgGeJU8k2wiErdxQP\_iFXumM8**

**¿Qué está intentando realizar?**

Está intentando cambiar el rol de usuario normal a rol de administrador.



### ¿Qué ocurre si intentamos validarlo con pyjwt?

Encoded

PASTE A TOKEN HERE

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoiaWVsaXB1IFJvZHZHJcdTAwZW50bnR1bWV6Iiwicm9sIjoiaXNBZG1pb29uLWkiAA8cjkmajwRUiNVHGEJ08k2wiErdxQP\_iFXumM8

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "typ": "JWT",  "alg": "HS256"}
```

PAYLOAD: DATA

```
{  "usuario": "Felipe Rodriguez",  "rol": "isAdmin"}
```

VERIFY SIGNATURE

HMACSHA256(

base64UrlEncode(header) + "." + base64UrlEncode(payload),

KeepCoding

) ☐ secret base64 encoded

The image shows a web-based JWT decoding tool. It has a top navigation bar with a 'Recipe' tab and a 'JWT Decode' tab. The 'JWT Decode' tab is active. The 'Input' section contains a long JWT token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3VhcmIvIjoiRmVsaXB1IFJvZHLDrWd1ZXoiLCJyb2wiOiJpc0FkbWluIn0.F4tTAt91rSWwBAkLtLNosyuf2s9ZJWsL57hixOKtRGY. The 'Output' section shows the decoded payload: { "usuario": "Felipe Rodriguez", "rol": "isAdmin" }. The interface includes various icons for file operations and settings.

Si se intenta validar, nos dará error debido al cambio realizado.

**5. El siguiente hash se corresponde con un SHA3 Keccak del texto “En KeepCoding aprendemos cómo protegernos con criptografía”.**

***bced1be95fbd85d2ffcce9c85434d79aa26f24ce82fbd4439517ea3f072d56fe***

***¿Qué tipo de SHA3 hemos generado?***

Se ha generado un SHA3-256.

```
home > kali > Desktop > ejerciciospractica > EJERCICIO5.py > ...
1  import hashlib
2
3
4  # initiating the "s" object to use the
5  # sha3_256 algorithm from the hashlib module.
6  s = hashlib.sha3_256()
7
8  # will output the name of the hashing algorithm currently in use.
9  print(s.name)
10
11 # will output the Digest-Size of the hashing algorithm being used.
12 print(s.digest_size)
13
14 # providing the input to the hashing algorithm.
15 s.update(bytes("En KeepCoding aprendemos cómo protegernos con criptografía","UTF-8"))
16
17 print(s.hexdigest())
```

```
[Running] python -u "/home/kali/Desktop/ejerciciospractica/EJERCICIO5.py"
sha3_256
32
bced1be95fbd85d2ffcce9c85434d79aa26f24ce82fbd4439517ea3f072d56fe

[Done] exited with code=0 in 0.051 seconds
```

***Y si hacemos un SHA2, y obtenemos el siguiente resultado:***

***4cec5a9f85dcc5c4c6ccb603d124cf1cdc6dfe836459551a1044f4f2908aa5d63739506f646  
883 3d77c07cfd69c488823b8d858283f1d05877120e8c5351c833***

***¿Qué hash hemos realizado?***

Con la herramienta Cyberchef se ha realizado un análisis del tipo de hash.

The screenshot shows the Cyberchef web application interface. On the left, the 'Recipe' panel has 'Analyse hash' selected. The main 'Input' area contains two lines of a long hexadecimal hash. The 'Output' area displays the following information:

Hash length: 128  
Byte length: 64  
Bit length: 512

Based on the length, this hash could have been generated by one of the following hashing functions:

- SHA-512
- SHA3-512
- BLAKE-512
- ECOH-512
- FSB-512
- Grøstl-512
- JH
- MD6
- Spectral Hash

At the bottom, there is a 'BAKE!' button and an 'Auto Bake' checkbox.

Se observa que el tamaño del hash es de 128 bits y es SHA3-512.

**Genera ahora un SHA3 Keccak de 256 bits con el siguiente texto: “En KeepCoding aprendemos cómo protegernos con criptografía.” ¿Qué propiedad destacarías del hash, atendiendo a los resultados anteriores?**

```
home > kali > Desktop > ejerciciospractica > EJERCICIO5.py > ...
1  import hashlib
2
3
4  # initiating the "s" object to use the
5  # sha3_256 algorithm from the hashlib module.
6  s = hashlib.sha3_256()
7
8  # will output the name of the hashing algorithm currently in use.
9  print(s.name)
10
11 # will output the Digest-Size of the hashing algorithm being used.
12 print(s.digest_size)
13
14 # providing the input to the hashing algorithm.
15 s.update(bytes("En KeepCoding aprendemos cómo protegernos con criptografía.", "UTF-8"))
16
17 print(s.hexdigest())
```

```
[Running] python -u "/home/kali/Desktop/ejerciciospractica/EJERCICIO5.py"
sha3_256
32
302be507113222694d8c63f9813727a85fef61a152176ca90edf1cfb952b19bf

[Done] exited with code=0 in 0.042 seconds
```

En comparación con el texto y resultado anterior, la propiedad a destacar es el signo de puntuación al final del texto “.”. Esto provoca que el nuevo resultado generado con un SHA3 Keccak de 256 bits sea distinto al anterior.

Cada vez que se añade un caracter nuevo en el texto inicial, automáticamente el hash cambia.

6. Calcula el hmac-256 (usando la clave contenida en el KeyStore) del siguiente texto:

*Siempre existe más de una forma de hacerlo, y más de una solución válida.*

*Se debe evidenciar la respuesta. Cuidado si se usan herramientas fuera de los lenguajes de programación, por las codificaciones es mejor trabajar en hexadecimal.*

```
home > kali > Desktop > ejerciciospractica > JS EJERCICIO6.js > ...
1  const crypto = require("crypto");
2
3  var algoritmo = "sha256";
4  var mensaje = "Siempre existe más de una forma de hacerlo, y más de una solución válida.";
5  var key = "2712a51c997e14b4df08d55967641b0677ca31e049e672a4b06861aa4d5826eb";
6
7  var hash = crypto.createHmac(algoritmo, key);
8
9  hash.update(mensaje);
10 console.log("Mensaje:\t", mensaje);
11 console.log("Clave:\t\t", key);
12 console.log("Método:\t\t", algoritmo);
13 var miHash = hash.digest("hex");
14 console.log("\nHMAC es:\t", miHash);
15 hash = crypto.createHash(algoritmo);
16 console.log("HMAC es:\t", hash.digest("base64"));
17 console.log("Longitud of HMAC: ", miHash.length * 4, " bits");
```

```
[Running] node "/home/kali/Desktop/criptografia-main/Código fuente/Hashing y Authentication/HMAC256.js"
Mensaje:      Siempre existe más de una forma de hacerlo, y más de una solución válida.
Clave:        2712a51c997e14b4df08d55967641b0677ca31e049e672a4b06861aa4d5826eb
Método:       sha256

HMAC es:      0358b0a81f64bfbe9dcf57e9a0f5f155f41cd5bacaef477e63aa23e2f29082ee
HMAC es:      47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=
Longitud of HMAC: 256 bits
```

```
home > kali > Desktop > ejerciciospractica > ejercicio6.1.py > ...
1  from Crypto.Hash import HMAC, SHA256
2
3  secret = bytes('2712a51c997e14b4df08d55967641b0677ca31e049e672a4b06861aa4d5826eb', 'utf-8')
4
5  #Generación
6  msg0= bytes('Siempre existe más de una forma de hacerlo, y más de una solución válida.', 'utf-8')
7  h = HMAC.new(secret, msg=msg0, digestmod=SHA256)
8  #h.update(bytes('Siempre existe más de una forma de hacerlo, y más de una solución válida.', 'utf-8'))
9  print(h.hexdigest())
10 mac = h.hexdigest()
11
12 #Verificación
13 h2 = HMAC.new(secret, digestmod=SHA256)
14 msg = bytes('Siempre existe más de una forma de hacerlo, y más de una solución válida.', 'utf-8')
15 h2.update(msg)
16 try:
17     h2.verify(mac)
18     print("Mensaje validado ok")
19 except ValueError:
20     print("Mensaje validado ko")
```

```
[Running] python -u "/home/kali/Desktop/ejercicospractica/ejercico6.1.py"
0358b0a81f64bfbe9dcf57e9a0f5f155f41cd5bacaef477e63aa23e2f29082ee
Mensaje validado ok

[Done] exited with code=0 in 0.1 seconds
```

**7. Trabajamos en una empresa de desarrollo que tiene una aplicación web, la cual requiere un login y trabajar con passwords. Nos preguntan qué mecanismo de almacenamiento de las mismas proponemos.**

**Tras realizar un análisis, el analista de seguridad propone un hash SHA-1. Su responsable, le indica que es una mala opción. ¿Por qué crees que es una mala opción?**

El SHA-1 es una mala opción para utilizarlo como mecanismo de almacenamiento debido a que se considera débil. Produce un valor de hash de 160 bits.

Se descubrió que este tipo de hash tiene varios fallos teóricos que podría provocar ataques de colisión. Es una situación muy complicada ya que permitiría que los atacantes cambiar archivos, sin que se observe.

**Después de meditarlo, propone almacenarlo con un SHA-256, y su responsable le pregunta si no lo va a fortalecer de alguna forma. ¿Qué se te ocurre?**

Como se comunica en el enunciado que se propone el almacenamiento de las contraseñas con SHA-256, una de las formas para fortalecer el hash, sería:  
SHA-256 + salt.

Un salt es un valor generado por una función criptográficamente segura que se agrega a la entrada de funciones hash para crear hashes únicos para cada entrada, independientemente de que la entrada no sea única. Un salt hace que una función hash parezca no determinista, lo cual es bueno ya que no queremos revelar contraseñas duplicadas a través de nuestro hash.

Es importante utilizar sales únicas, por ejemplo si varios usuarios utilizaran la misma contraseña, ambas contraseñas que tuviesen salt tendrían el “mismo valor”, pero si se elige otro salt para la misma contraseña, se obtienen varias contraseñas únicas y más largas que tienen un valor diferente.

El almacenamiento de contraseñas sería:

Salt+ Hash + Nombre de usuario

Cuando el usuario inicie sesión se puede buscar por el nombre de usuario, agregar el salt a la contraseña proporcionada, codificarlo y verificar si el hash almacenado coincide con el hash calculado.

Es muy importante que el salt sea único para cada hash. Además si el almacenamiento lo permite usar Salt de 32 y 64 bytes con el tamaño real dependiendo de la función de protección que tenga. Cuanto más larga sea el salt aumenta la complejidad computacional de contraseñas de posible ataque.

Las contraseñas se codifican y se añade el salt con el algoritmo "bcrypt".

***Parece que el responsable se ha quedado conforme, tras mejorar la propuesta del SHA256, no obstante, hay margen de mejora. ¿Qué propondrías?***

Actualmente es un hash que no ha sido roto y ofrece seguridad, además de ser más rápido que otros hashes. Pero se podría mejorar con otro tipo de hash.

Un inconveniente que tiene SHA-2 es que no puede generar MACs, sin embargo SHA-3 no tiene esta dificultad.

***8. Tenemos la siguiente API REST, muy simple.***

***Request:***

***Post /movimientos***

Campo	Tipo	Requiere Confidencialidad	Observaciones
idUsuario	Number	N	Identificador
Usuario	String	S	Nombre y Apellidos
Tarjeta	Number	S	

***Petición de ejemplo que se desea enviar:***

***{"idUsuario":1,"usuario":"José Manuel Barrio Barrio","tarjeta":4231212345676891}***

***Response:***

Campo	Tipo	Requiere Confidencialidad	Observaciones
idUsuario	Number	N	Identificador
movTarjeta	Array	S	Formato del ejemplo
Saldo	Number	S	Tendra formato 12300 para indicar 123.00
Moneda	String	N	EUR, DOLLAR

```
{
  "idUsuario": 1,
  "movTarjeta": [{
    "id": 1,
    "comercio": "Comercio Juan",
    "importe": 5000
  }, {
    "id": 2,
    "comercio": "Rest Paquito",
    "importe": 6000
  }],
  "Moneda": "EUR",
  "Saldo": 23400
}
```

**Como se puede ver en el API, tenemos ciertos parámetros que deben mantenerse confidenciales. Así mismo, nos gustaría que nadie nos modificase el mensaje sin que nos enterásemos. Se requiere una redefinición de dicha API para garantizar la integridad y la confidencialidad de los mensajes. Se debe asumir que el sistema end to end no usa TLS entre todos los puntos.**

**¿Qué algoritmos usarías? ¿Serías capaz de hacer un ejemplo en Python de cómo resolverlo?**

Se usaría un algoritmo HS256. Se ha intentado pero no ha sido posible.

**Importante, de cara a calcular cualquier cosa, se deben quitar retornos de carro, espacios intermedios, usar las mismas comillas, etc)**



En el siguiente apartado para crear una API sencilla con los parámetros confidenciales, con la ayuda del ejemplo del archivo "servidor.py".  
Se abrirá un entorno virtual

```
home > kali > Desktop > ejerciciospractica > EJERCICIO8.py > idusuario
1
2  from flask import Flask, jsonify, request
3  import json
4  import sqlite3
5
6
7  app = Flask(__name__)
8
9  @app.route("/")
10 def idusuario():
11     return jsonify({"message": "idusuario"})
12
13 @app.route("/tarjetas", methods=['POST'])
14 def tarjetas():
15
16     #Obtenemos el body en formato json
17     request_data = request.get_json()
18
19     #Obtenemos un campo determinado
20     nombre = request_data["Jose Manuel Barrios Barrios"]
21
22     #Montamos un json para su respuesta
23     tarjetas = {"tarjeta": 4231212345676891}
24
25     #Recuperamos datos de la cabecera
26     jwt = request.headers.get('jwt')
27
28     #Escribir log
29     app.logger.info(jwt)
30     app.logger.info (nombre)
31
32
33     return tarjetas
34
35
36
37
38
39 #Iniciamos el servidor
40 if __name__ == '__main__':
41     app.run(host='0.0.0.0', debug=True)
```

```
[Running] python -u "/home/kali/Desktop/ejercicospractica/EJERCICIO8.py"
* Serving Flask app 'EJERCICIO8' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 111-639-658
10.0.2.15 - - [31/Jul/2022 10:34:06] "GET / HTTP/1.1" 200 -
```

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

message: "idusuario"

```
home > kali > Desktop > ejerciciospractica > EJERCICIO8.py > idusuario
1
2 from flask import Flask, jsonify, request
3 import json
4 import sqlite3
5
6
7 app = Flask(__name__)
8
9 @app.route("/")
10 def idusuario():
11     return jsonify([{"idusuario": "1", "movTarjeta": [{"id": "1", "comercio": "Comercio Juan", "importe": 5000}, {"id": "2", "comercio": "Rest Paquito", "importe": 6000}], "Moneda": "EUR"}])
12
13 @app.route("/tarjetas", methods=['POST'])
14 def tarjetas():
15
16     #Obtenemos el body en formato json
17     request_data = request.get_json()
18
19     #Obtenemos un campo determinado
20     nombre = request_data["Jose Manuel Barrios"]
21     moneda = request_data["EUR, DOLLAR"]
22     #Montamos un json para su respuesta
23     tarjetas = {"tarjeta": 423121234567891}
24
25     #Recuperamos datos de la cabecera
26     jwt = request.headers.get('jwt')
27
28     #Escribir log
29     app.logger.info(jwt)
30     app.logger.info(nombre)
31     app.logger.info(moneda)
32
33
34     return tarjetas
35
36
37
38
39
40 #Iniciamos el servidor
41 if __name__ == '__main__':
42     app.run(host='0.0.0.0', debug=True)
```

En la captura completa no se puede ver correctamente, a continuación se muestran las capturas por partes:

```
home > kali > Desktop > ejerciciospractica > EJERCICIO8.py > idusuario
1
2 from flask import Flask, jsonify, request
3 import json
4 import sqlite3
5
6
7 app = Flask(__name__)
8
```

```

0
7 app = Flask(__name__)
8
9 @app.route("/")
0 def idusuario():
1     return jsonify([{"idusuario": "1", "movTarjeta": [{"id": "1", "comercio": "Comercio Juan", "importe": 5000 }
2

```

```

, {"id": "2", "comercio": "Rest Paquito", "importe": 6000 }], "Moneda" : "EUR", "Saldo" : 23400}]

```

```

13 @app.route("/tarjetas", methods=['POST'])
14 def tarjetas():
15
16     #Obtenemos el body en formato json
17     request_data = request.get_json()
18
19     #Obtenemos un campo determinado
20     nombre = request_data["Jose Manuel Barrios Barr
21     moneda = request_data["EUR, DOLLAR"]
22     #Montamos un json para su respuesta
23     tarjetas = {"tarjeta": 4231212345676891}
24
25     #Recuperamos datos de la cabecera
26     jwt = request.headers.get('jwt')
27
28     #Escribir log
29     app.logger.info(jwt)
30     app.logger.info (nombre)
31     app.logger.info (moneda)
32
33
34     return tarjetas
35
36
37
38
39
40 #Iniciamos el servidor
41 if __name__ == '__main__':
42     app.run(host='0.0.0.0', debug=True)

```

```
[Running] python -u "/home/kali/Desktop/ejercicospractica/EJERCICIO8.py"
* Serving Flask app 'EJERCICIO8' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 111-639-658
10.0.2.15 - - [31/Jul/2022 11:08:02] "GET / HTTP/1.1" 200 -
```

JSON	Raw Data	Headers
Save	Copy	Collapse All Expand All Filter JSON
Moneda: "EUR"		
Saldo: 23400		
idusuario: "1"		
▼ movTarjeta:		
▼ 0:		
comercio: "Comercio Juan"		
id: "1"		
importe: 5000		
▼ 1:		
comercio: "Rest Paquito"		
id: "2"		
importe: 6000		

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

<!DOCTYPE html>  
<html class="theme-light" platform="linux" dir="ltr"> [event]  
> <head> ... </head>  
...

Filter Styles

Pseudo-elements

This Element

element {

inline

Layout Computed Changes Fonts Animat

▼ Flex Item of div.jsonPanelBox.tab-panel-inner

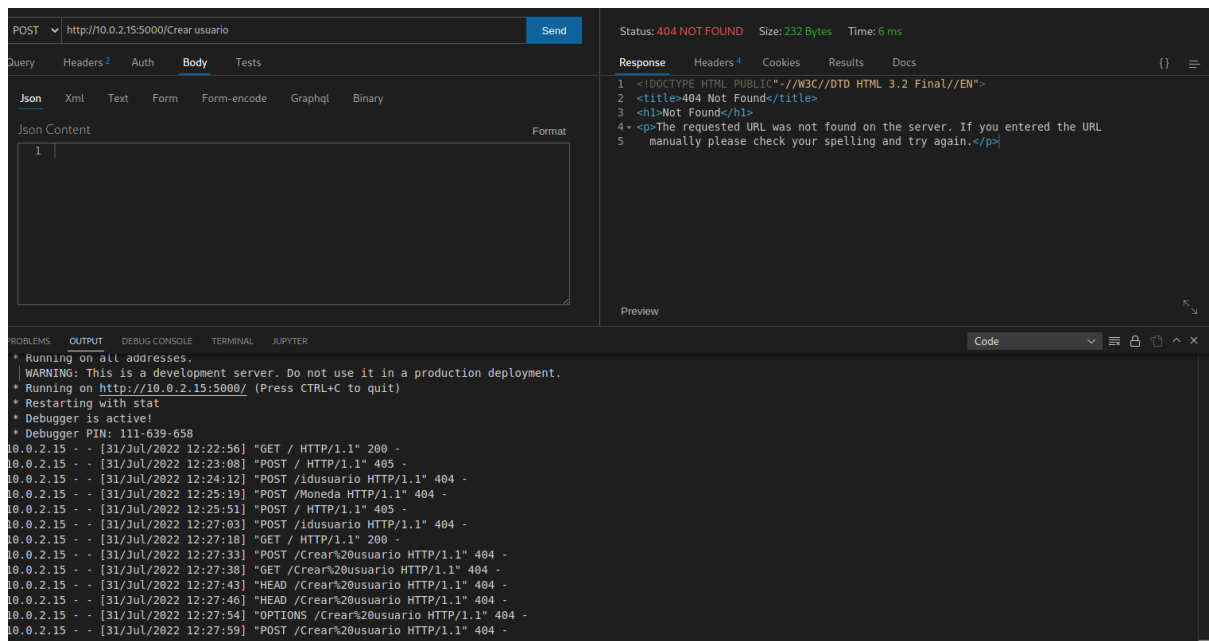
< div.toolbar

Filter Output

Errors Warnings Logs Info Debug CSS XHR Requests

Content Security Policy: The page's settings blocked the loading of a resource at http://10.0.2.15:5000/favicon.ico ("default-src"). resource:191:19

JSON	Raw Data	Headers
Save	Copy	Collapse All Expand All Filter JSON
Moneda: "EUR"		
Saldo: 23400		
idusuario: "1"		
▼ movTarjeta:		
▼ 0:		
comercio: "Comercio Juan"		
id: "1"		
importe: 5000		
▼ 1:		
comercio: "Rest Paquito"		
id: "2"		
importe: 6000		



**9. El responsable de Raúl, Pedro, ha enviado este mensaje a RRHH:**

**Se debe ascender inmediatamente a Raúl. Es necesario mejorarle sus condiciones económicas un 20% para que se quede con nosotros.**

**Lo acompaña del siguiente fichero de firma PGP (MensajeRespoDeRaulARRHH.txt.sig). Nosotros, que pertenecemos a RRHH vamos al directorio a recuperar la clave para verificarlo. Tendremos los ficheros Pedro-priv.txt y Pedro-publ.txt, con las claves privada y pública.**

**Las claves de los ficheros de RRHH son RRHH-priv.txt y RRHH-publ.txt que también se tendrán disponibles.**

**Se requiere verificar la misma, y evidenciar dicha prueba.**

**Así mismo, se requiere firmar el siguiente mensaje con la clave correspondiente de las anteriores, simulando que eres personal de RRHH.**

**Viendo su perfil en el mercado, hemos decidido ascenderle y mejorarle un 25% su salario. Saludos.**

**Por último, cifra el siguiente mensaje tanto con la clave pública de RRHH como la de Pedro y adjunta el fichero con la práctica.**

**Estamos todos de acuerdo, el ascenso será el mes que viene.**

**10. Nuestra compañía tiene un contrato con una empresa que nos da un servicio de almacenamiento de información de videollamadas. Para lo cual, la misma nos envía la clave simétrica de cada videollamada cifrada usando un RSA-OAEP. El hash que usa el algoritmo interno es un SHA-256.**

**El texto cifrado es el siguiente:**

**7edee3ec0b808c440078d63ee65b17e85f0c1adbc0da1b7fa842f24fb06b332c1560  
38062d9daa8ccfe83bace1dca475cfb7757f1f6446840044fe698a631fe882e1a6fc  
00a2de30025e9dcc76e74f9d9d721e9664a6319eaa59dc9011bfc624d2a63eb0e449  
ed4471ff06c9a303465d0a50ae0a8e5418a1d12e9392faaaf9d4046aa16e424ae1e2  
6844bcf4abc4f8413961396f2ef9ffcd432928d428c2a23fb85b497d89190e3cfa49  
6b6016cd32e816336cad7784989af89ff853a3acd796813eade65ca3a10bbf58c621  
5fdf26ce061d19b39670481d03b51bb0eccc926c9d6e9cb05ba56082a899f9aa72f9  
4c158e56335c5594fcc7f8f301ac1e15a938**

**Las claves pública y privada las tenemos en los ficheros clave-rsa-oaep-publ.pem y clave-rsa-oaep-priv.pem.**

**Si has recuperado la clave, vuelve a cifrarla con el mismo algoritmo. ¿Por qué son diferentes los textos cifrados?**

Los textos cifrados son distintos debido a que cada vez que se vuelve a cifrar se añade un padding. En cada cifrado el padding cambia, haciendo que el cifrado sea totalmente distinto.

**11. Nos debemos comunicar con una empresa, para lo cual, hemos decidido usar un algoritmo como el AES/GCM en la comunicación. Nuestro sistema, usa los siguientes datos en cada comunicación con el tercero:**

**Key:E2CFF885901B3449E9C448BA5B948A8C4EE322152B3F1ACFA0148FB3A426DB7  
2 Nonce:9Yccn/f5nJJhAt2S**

**¿Qué estamos haciendo mal?**

Los datos de comunicación que se emplean no son los correctos.

Es un cifrado con autenticación.

El nonce debe ser distinto, no se debe de reutilizar, en todos los mensajes para evitar que nos puedan hacer un ataque.