

Universidad Galileo de Guatemala
Técnico en Desarrollo de Software
Seguridad Informática
Ingeniero Randy Fernando Juárez Najarro

ACTIVIDAD 7

INTRODUCCIÓN A LA CRIPTOGRAFÍA

Brenda Samara Escobar Avila

Carnet: 23005735

Guatemala, lunes 09 de diciembre de 2024

INTRODUCCIÓN

La criptografía es una técnica fundamental en el mundo digital actual, utilizada para proteger la información y garantizar la privacidad de las comunicaciones en un entorno cada vez más interconectado. Desde la simple transmisión de mensajes hasta la protección de transacciones financieras y la autenticación de usuarios, la criptografía juega un papel crucial en la seguridad de sistemas de todo tipo. Su evolución a lo largo de la historia, desde los métodos manuales utilizados en la antigüedad hasta los algoritmos modernos que operan a nivel de hardware y software, ha permitido que la información se mantenga a salvo de accesos no autorizados y ataques malintencionados.

La criptografía se clasifica en tres tipos principales: cifrado simétrico, cifrado asimétrico y funciones hash, cada uno con características y aplicaciones particulares. El **cifrado simétrico** es el método más antiguo y eficiente, en el cual el mismo algoritmo y clave se utilizan tanto para cifrar como para descifrar un mensaje. Esta técnica es rápida y adecuada para la protección de grandes cantidades de datos, pero su mayor desafío radica en la gestión segura de las claves compartidas entre las partes. Ejemplos de algoritmos de cifrado simétrico incluyen **AES (Advanced Encryption Standard)** y **DES (Data Encryption Standard)**.

Por otro lado, el **cifrado asimétrico** utiliza un par de claves: una clave pública, que se comparte libremente, y una clave privada, que se mantiene en secreto. Esta forma de cifrado es fundamental para la creación de sistemas de seguridad más robustos, como el **protocolo TLS/SSL**, utilizado para proteger la comunicación en la web. Los algoritmos asimétricos, como **RSA (Rivest-Shamir-Adleman)**, permiten que las partes se autenticuen y realicen intercambios de claves de manera segura, facilitando la implementación de la firma digital y la encriptación de datos sensibles.

Por último, las **funciones hash** son herramientas criptográficas que convierten un mensaje de longitud variable en una cadena de longitud fija, llamada "hash". A diferencia del cifrado, que es reversible, la función hash es un proceso unidireccional, lo que significa que no es posible obtener el mensaje original a partir de su hash. Esta característica la hace ideal para almacenar contraseñas y verificar la integridad de los datos. Algoritmos como **SHA-256 (Secure Hash Algorithm 256-bit)** son ampliamente utilizados por su resistencia a ataques de colisiones y su capacidad para generar un hash único para cada entrada de datos.

INTRODUCCIÓN A LA CRIPTOGRAFÍA

Instrucciones:

Aplicar **3 algoritmos de cifrado** uno de cada tipo:

- Cifrado simétrico
- Cifrado asimétrico
- Función hash

1. Puede utilizar cualquier lenguaje de programación y cualquier librería adicional. O bien, ejecutarlo directamente en utilizando en CMD de Windows o la terminal de Linux.
2. Coloque las pantallas mostrando el proceso y su funcionamiento. Describa brevemente que ocurre en cada imagen. Incluir imágenes del código donde se aplica (si aplica).

REPOSITORIO EN GIT

https://github.com/SamyER33/23005735_A7_SI.git

Ejemplos de Cifrado y Hashing en Python

Este repositorio contiene tres ejemplos básicos en Python que demuestran el uso de diferentes técnicas de seguridad: hashing, cifrado simétrico y cifrado asimétrico.

Requisitos

Para ejecutar estos códigos, asegúrate de tener Python 3 instalado y las siguientes bibliotecas:

- **hashlib** (incluida por defecto en Python)
- **cryptography** (instálala con ``pip install cryptography``)

Ejemplo 1: Hashing con `hashlib`

Este ejemplo utiliza el algoritmo SHA-256 para generar un hash de un mensaje.

```
python
import hashlib
```

Mensaje a encriptar

```
message = b"El espinorajo salió de la jungla"
```

Generar hash con SHA-256

```
hashObject = hashlib.sha256(message)
hashDigest = hashObject.hexdigest()
```

```
print(f"Mensaje encriptado con Hash: {hashDigest}")
```

Salida esperada

Un hash único en formato hexadecimal para el mensaje proporcionado.

![SalidaHash](Resultados/Hash.JPG)

Seguridad informática

The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project named 'CRYPTOGRAPHY' with files: `__pycache__`, `Resultados`, `cryptography_asimetrico.py`, `cryptography_hashlib.py` (selected), `cryptography_simetrico.py`, and `README.md`. The main editor displays the content of `cryptography_hashlib.py`:

```
1 import hashlib # Importa la biblioteca hashlib, que proporciona algoritmos de hash seguros como SHA-256.
2
3 # Mensaje a encriptar
4 message = b"El espinorojo salio de la jungla" # Define un mensaje como una cadena de bytes (necesario para la función hashlib).
5
6 # Mensaje encriptado con hashlib
7 hashObject = hashlib.sha256(message) # Crea un objeto hash utilizando el algoritmo SHA-256.
8 hashDigest = hashObject.hexdigest() # Convierte el resultado del hash en una representación hexadecimal legible.
9
10 # Imprime el mensaje encriptado en formato hexadecimal
11 print(f"Mensaje: {message}")
12 print(f"Mensaje encriptado con Hash: {hashDigest}")
13
```

Below the editor, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\torta\OneDrive\Documents\Sammy\Cryptography> & C:/Users/torta/AppData/Local/Programs/Python/Python310/python.exe c:/Users/torta/OneDrive/Doc
uments/Sammy/Cryptography/cryptography_hashlib.py
Mensaje: b'El espinorojo salio de la jungla'
Mensaje encriptado con Hash: 34cd59023c110b64aaf51d79f67c3d389c5aa5f03289545aa6afb822ebb6d98f
PS C:\Users\torta\OneDrive\Documents\Sammy\Cryptography>
```

The Windows taskbar at the bottom shows the time as 06:45 p. m. on 09/12/2024.

Mensaje: b'El espinorojo salio de la jungla'

Mensaje encriptado con Hash: 34cd59023c110b64aaf51d79f67c3d389c5aa5f03289545aa6afb822ebb6d98f

Ejemplo 2: Cifrado Simétrico con `cryptography.fernet`

En este ejemplo, se utiliza el cifrado simétrico Fernet para proteger y recuperar un mensaje.

```
python
from cryptography.fernet import Fernet
```

Generar una clave

```
key = Fernet.generate_key()
cipher_suite = Fernet(key)
```

Texto a cifrar

```
Mensaje = b"El espinoroyo salió de la jungla"
```

Cifrar el texto

```
mensajeCipher = cipher_suite.encrypt(Mensaje)
print(f"Mensaje cifrado: {mensajeCipher}")
```

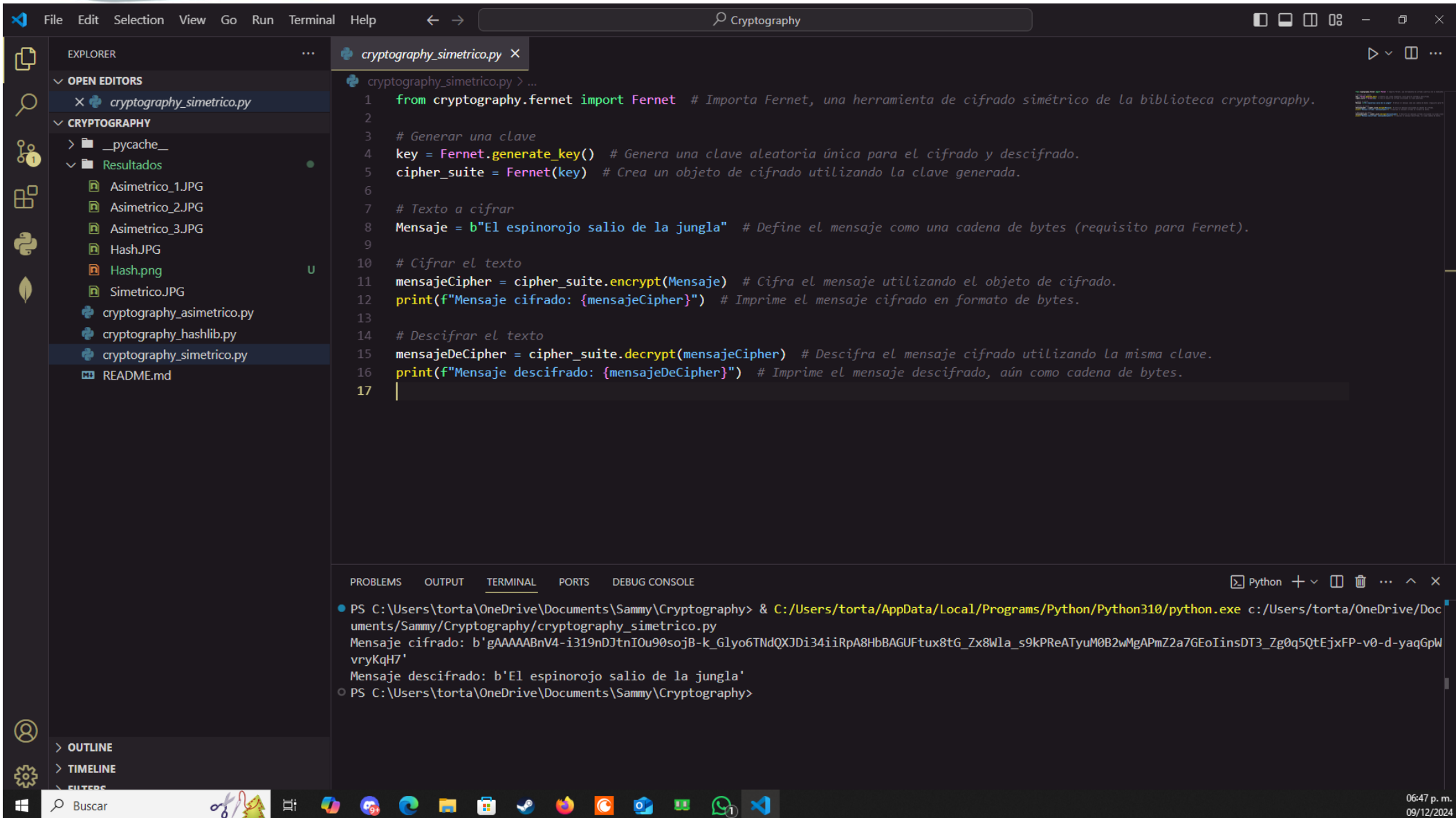
Descifrar el texto

```
mensajeDeCipher = cipher_suite.decrypt(mensajeCipher)
print(f"Mensaje descifrado: {mensajeDeCipher}")
```

Salida esperada

- **Mensaje cifrado:** Texto cifrado ilegible.
 - **Mensaje descifrado:** El mensaje original.
- ![[SalidaSim]]([Resultados/Simetrico.JPG](#))

Seguridad informática



The image shows a Visual Studio Code editor window with a Python script named `cryptography_simetrico.py` open. The script uses the `cryptography` library's `Fernet` class for symmetric encryption and decryption. The code is as follows:

```
1 from cryptography.fernet import Fernet # Importa Fernet, una herramienta de cifrado simétrico de la biblioteca cryptography.
2
3 # Generar una clave
4 key = Fernet.generate_key() # Genera una clave aleatoria única para el cifrado y descifrado.
5 cipher_suite = Fernet(key) # Crea un objeto de cifrado utilizando la clave generada.
6
7 # Texto a cifrar
8 Mensaje = b"El espinorojo salio de la jungla" # Define el mensaje como una cadena de bytes (requisito para Fernet).
9
10 # Cifrar el texto
11 mensajeCipher = cipher_suite.encrypt(Mensaje) # Cifra el mensaje utilizando el objeto de cifrado.
12 print(f"Mensaje cifrado: {mensajeCipher}") # Imprime el mensaje cifrado en formato de bytes.
13
14 # Descifrar el texto
15 mensajeDeCipher = cipher_suite.decrypt(mensajeCipher) # Descifra el mensaje cifrado utilizando la misma clave.
16 print(f"Mensaje descifrado: {mensajeDeCipher}") # Imprime el mensaje descifrado, aún como cadena de bytes.
17
```

The terminal output shows the execution of the script:

```
PS C:\Users\torta\OneDrive\Documents\Sammy\Cryptography> & C:/Users/torta/AppData/Local/Programs/Python/Python310/python.exe c:/Users/torta/OneDrive/Doc
uments/Sammy/Cryptography/cryptography_simetrico.py
Mensaje cifrado: b'gAAAAABnV4-i319nDJtnIOu90sojB-k_Glyo6TndQXJDj34iiRpA8HbBAGUftux8tG_Zx8Wla_s9kPrEATyuM0B2wMgAPmZ2a7GEoIinsDT3_Zg0q5QtEjxFP-v0-d-yaqGpW
vryKqH7'
Mensaje descifrado: b'El espinorojo salio de la jungla'
```

06:47 p. m.
09/12/2024

```
Mensaje cifrado: b'gAAAAABnVko_HVakzSgibbVg6-pouYfxk40LPZon1H1KNjz_p1nneqtTB57FA4Ssv7HxuyFM1EpRp8seNBkBEj3_h0G-3uVc_QZf2g290xAdiS-ARm3bthbP21sibEsGHioSEVW__XE1'
Mensaje descifrado: b'El espinorojo salio de la jungla'
```

Ejemplo 3: Cifrado Asimétrico con `cryptography.hazmat`

Este ejemplo utiliza el algoritmo RSA para generar un par de claves (privada y pública) y realizar operaciones de cifrado y descifrado.

```
python
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
```

Generar par de claves

```
llavePrivada = rsa.generate_private_key(
    public_exponent=65537,
    key_size=1024,
)
```

```
llavePublica = llavePrivada.public_key()
```

Serializar las claves

```
private_pem = llavePrivada.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.TraditionalOpenSSL,
    encryption_algorithm=serialization.NoEncryption()
)
```

```
public_pem = llavePublica.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)
```

Mostrar las claves

```
print(private_pem.decode('utf-8'))
print(public_pem.decode('utf-8'))
```


Texto a cifrar

mensaje = b"El espinorojo salió de la jungla"

Cifrar el mensaje con la clave pública

```
textoCifrado = llavePublica.encrypt(
    mensaje,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print(f"Mensaje cifrado: {textoCifrado}")
```

Descifrar el mensaje con la clave privada

```
textoDecifrado = llavePrivada.decrypt(
    textoCifrado,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print(f"Mensaje descifrado: {textoDecifrado}")
```

Salida esperada

- **Claves:** La clave privada y pública generadas en formato PEM.

![SalidaAsim](Resultados/Asimetrico_1.JPG "Clave Privada")

![SalidaAsim2](Resultados/Asimetrico_2.JPG "Clave Publica")

- **Mensaje cifrado:** Texto cifrado ilegible.

- **Mensaje descifrado:** El mensaje original.

![SalidaAsim3](Resultados/Asimetrico_3.JPG "Resultados")

Seguridad informática

File Edit Selection View Go Run Terminal Help

cryptography

EXPLORER

OPEN EDITORS

- cryptography_asimetrico.py

CRYPTOGRAPHY

- __pycache__
- Resultados
 - Asimetrico_1.JPG
 - Asimetrico_2.JPG
 - Asimetrico_3.JPG
 - Asimetrico.png
 - Hash.JPG
 - Hash.png
 - Simetrico.JPG
 - Simetrico.png
- cryptography_asimetrico.py
- cryptography_hashlib.py
- cryptography_simetrico.py
- README.md

cryptography_asimetrico.py

```
45 padding.OAEP( # Debe coincidir con el esquema de relleno utilizado durante el cifrado.
46 mgf=padding.MGF1(algorithm=hashes.SHA256()), # Generador de máscara basado en SHA-256.
47 algorithm=hashes.SHA256(), # Algoritmo de hash utilizado.
48 Label=None # No se utiliza etiqueta.
49 )
50 )
51 print(f"Mensaje descifrado: {textoDecifrado}") # Muestra el mensaje descifrado.
52
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

Python

```
PS C:\Users\torta\OneDrive\Documents\Sammy\Cryptography> & C:/Users/torta/AppData/Local/Programs/Python/Python310/python.exe c:/Users/torta/OneDrive/Doc
uments/Sammy/Cryptography/cryptography_asimetrico.py
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC7q4hKVL843AkURsA6xRy8TKhRYLW+qSGX0gJJUoMc+cUld1p
Eh6NzccvkuHvvT5nvHzwWTK8+s1EAYd1LoZJY2/EXqnhLzCGCvo5KDX4MFKKqqf
+ZTHP5HzkQghhTymWcImzvJSXectMeJD5QcJtM7UccChCclg4H60j6gdwIDAQAB
AoGAL2jJ6V03YXnUoVauQSQcC985tjUkNKItZ1kbUsKRdIhdPAoQAPce2NvFmd19
Uj8QgMxRE7a05qDEL3XjbmruaTT9N8dgMnJ34RNCf8wUruEuHnq1yPTwSTUwSEM4
7sbrkxvhs0BMSinY5f2uiu9cmgxcQE3AeKJRLbao4peRWECCQDfnwVhBPTaZBQN
LUnkoG9kt1GAJ1jemsfxUQBRriQvotDUB09T3iCZLxhWARTUgLT8Mff2qXeGLQYr
Ch/c7a7bAkEA1tfoq0Jtvld+IKKjs0d0ik+1R1NJaeD59rZCLTX2KZJU7RcLVnag
k4oPjLy21luPFYmDW3UMrb1BLt7CXsB1QJBALz1V2XdcfhxEX8QA30dEnvBXtci
Za+kj8E163wT3f1fS4fNo1z1BhRf8+6yeyYGmFkQ3KVAaFcVkBWBJ7SBypoECQQCF
TFTJ3V9117cj3Iug+L8cRapQhLZqAPJWRmLs1iwwQt16d0/N57Bdmi1nrqRP5Wj1
+rWfY31jMeMKeqAuE6m5AkEAtK6rrq5cwKAhWJ46X+0dn92HcE15CiJ8bFmi+C6
af2eX0+vkaXY4NKVbJOLdqNt0FDG21I0jArWB00zpeLu3g==
-----END RSA PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MIGfMA0GCsGqGSIB3DQEBAQUAA4GNADCBiQKBgQC7q4hKVL843AkURsA6xRy8TKhR
YLW+qSGX0gJJUoMc+cUld1pEh6NzccvkuHvvT5nvHzwWTK8+s1EAYd1LoZJY2/E
XqnhLzCGCvo5KDX4MFKKqqf+ZTHP5HzkQghhTymWcImzvJSXectMeJD5QcJtM7U
ccChCclg4H60j6gdwIDAQAB
-----END PUBLIC KEY-----

Mensaje cifrado: b'\xa0\x16\xc8\x9e\x93\xfc@\xc6\x17-y\x93&\xb5\xb5n1\xff_\x05'\xc3\xbd\xc8\xd5\xb6:I'\x0f\xa6\x96\xe0-\xcc\xcf9F>\xb2^\x94\x9d\x9d\x1d
j~d~:Y\x06\xc3G\xa0\xcb*md\xc7""Mr\x80\xd5\x1cJo\x10\x82\xb0\xd5B\x13+\xa9U\xf2\xdd\x98\xad\x8cY8\x9a\xb2\xbd\xba1R\x81\xc2\xf2\xd8\r\xef'\x8e\xfc'
\x1d\xac\x852\xf9\x8d\xfe\xa3!\xdb\x98\xfc\x17\x8a\x97\x7f\xca\x1d3\xce\x16\x16\xe6\x9d\x8c\xfcI'
Mensaje descifrado: b'El espinorojo salio de la jungla'
PS C:\Users\torta\OneDrive\Documents\Sammy\Cryptography>
```

OUTLINE

TIMELINE

FILTERS

Buscar

06:51 p. m.
09/12/2024


```
-----BEGIN RSA PRIVATE KEY-----  
MIICXgIBAAKBgQC7q4hKVL843AkURsA6xRy8TKhRYLW+qSGX0gJJUoMcC+cUld1p  
Eh6NzccvkuHvvT5nvHzwWTK8+s1EAYd1LoZJY2/EXqnhLzCGCvo5KDX4MFKKqqf  
+ZTHP5HzkQghhTymWcImzvJSXectMeJD5QcJtM7UccChCcc1g4H60j6gdwIDAQAB  
AoGAL2jJ6V03YXnUoVauQSqCc985tjUkNKItZ1kbUsKRdIhdPAoQAPce2NvfMd19  
Uj8QgMxRE7a05qDEL3XjbmruaTT9N8dgMnJ34RNCF8wUruEuHNq1yPTwSTUwSEM4  
7sbkrkvhzs0BMSinY5f2ui9cmgcxQE3AeKJRLbao4peRWECCQDfnwVhBPTaZBQN  
LUrkoG9kt1GAJ1jemsfxUQBRriQvotDUB09T3iCZLxhWARTUgLt8Mff2qXeGLQYr  
Ch/c7a7bAkEA1tfoq0Jtvld+IKKjs0dOik+1R1NJaeD59rZCLTX2KZJU7RcLVnag  
k4oPjLyY21luPFYmDW3UMrb1BLt7CXsB1QJBALz1V2XdCfhxEX8QA30dEnvBXtci  
Za+kj8EI63wT3f1fS4fNo1z1BhRf8+6yeyYGmFkQ3KVAAfcVkB7SBypoECQQCF  
TFTJ3V9117cj3IUg+L8cRapQhLZqAPJWRmLs1iwzQt16d0/N57Bdmi1nrqRP5Wjl  
+rWFY31jMeMKeqAuE6m5AkEAtK6rrq5cwKAHaWJ46X+0dn92HcEl5CiJ8bFmi+C6  
af2eX0+vkaXY4NKVbJOLdqNt0FDG21I0jArWB00zpeLu3g==  
-----END RSA PRIVATE KEY-----
```




```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC7q4hKVL843AkURsA6xRy8TKhR
YLW+qSGX0gJJUoMcC+cUld1pEh6NzccvkuHvvT5nvHzwWTK8+s1EAYd1LoZJY2/E
XqnhLzCGCvo5KDX4MFKKqqf+ZTHP5HzkQghhTymWcImzvJSXectMeJD5QcJtM7U
ccChCcc1g4H60j6gdwIDAQAB
-----END PUBLIC KEY-----
```

```
Mensaje cifrado: b'\xa0\x16\xc8\x9e\x93\xfc@\xc6\x17-y\x93&\xb5\xb5n1\xff_\x05\''\xc3\xbd\xc8\xd5\xb6:I\''\x0f\xa6\x96\xe0-\xcc\x9f>\xb2^\x94\x9d\xd9\x1d
j~d~:Y\x06\xc3G\xa0\xcb*md\xc7"Mr\x80\xd5\x1c]o\x10\x82\xb0\xd5B\x13+\xa9U\xf2\xdd\x98\xad\x8cY8\x9a\xb2\xb7\xbd\xaa1R\x81\xc2\xf2\xd8\r\xef\''\x8e\xfc\
x1d\xac\x852\xf9\x8d\xfe\xa3!\xdb\x98\xfc\xd7\xa8\x97\x7f\xca\xd3\xce\xd8\xde\x16\xe6\xe8\x9d\x8c\xfcI'
```

```
Mensaje descifrado: b'El espinorojo salio de la jungla'
```

Notas Importantes

- **Hashing:** Es un proceso unidireccional, ideal para verificar integridad o almacenar contraseñas.
- **Cifrado Simétrico:** La misma clave cifra y descifra los datos, por lo que es esencial mantenerla segura.
- **Cifrado Asimétrico:** Utiliza un par de claves (pública y privada) para mejorar la seguridad, pero es más lento que el cifrado simétrico.