

Informe Laboratorio 4

Sección x

Alumno: Diego Martin
e-mail: diego.martin@mail.udp.cl

Junio de 2025

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades según criterio de rúbrica	3
2.1. Investiga y documenta los tamaños de clave e IV	3
2.1.1. DES	3
2.1.2. 3DES	3
2.1.3. AES-256	3
2.2. Solicita datos de entrada desde la terminal	4
2.3. Valida y ajusta la clave según el algoritmo	4
2.3.1. DES	4
2.3.2. 3DES	5
2.3.3. AES-256	5
2.4. Implementa el cifrado y descifrado en modo CBC	6
2.4.1. DES	6
2.4.2. 3DES	7
2.4.3. AES	8
2.5. Compara los resultados con un servicio de cifrado online	8
2.5.1. DES	8
2.5.2. 3DES	10
2.5.3. AES-256	11
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real	13

1. Descripción de actividades

Desarrollar un programa en Python utilizando la librería pycrypto para cifrar y descifrar mensajes con los algoritmos DES, AES-256 y 3DES, permitiendo la entrada de la key, vector de inicialización y el texto a cifrar desde la terminal.

Instrucciones:

1. Investigación

- Investigue y documente el tamaño en bytes de la clave y el vector de inicialización (IV) requeridos para los algoritmos DES, AES-256 y 3DES. Mencione las principales diferencias entre cada algoritmo, sea breve.

2. El programa debe solicitar al usuario los siguientes datos desde la terminal

- Key correspondiente a cada algoritmo.
- Vector de Inicialización (IV) para cada algoritmo.
- Texto a cifrar.

3. Validación y ajuste de la clave

- Si la clave ingresada es menor que el tamaño necesario para el algoritmo complete los bytes faltantes agregando bytes adicionales generados de manera aleatoria (utiliza `get_random_bytes`).
- Si la clave ingresada es mayor que el tamaño requerido, trunque la clave a la longitud necesaria.
- Imprima la clave final utilizada para cada algoritmo después de los ajustes.

4. Cifrado y Descifrado

- Implemente una función para cada algoritmo de cifrado y descifrado (DES, AES-256, y 3DES). Use el modo CBC para todos los algoritmos.
- Asegúrese de utilizar el IV proporcionado por el usuario para el proceso de cifrado y descifrado.
- Imprima tanto el texto cifrado como el texto descifrado.

5. Comparación con un servicio de cifrado online

- Selecciona uno de los tres algoritmos (DES, AES-256 o 3DES), ingrese el mismo texto, key y vector de inicialización en una página web de cifrado online.
- Compare los resultados de tu programa con los del servicio online. Valide si el resultado es el mismo y fundamente su respuesta.

6. Aplicabilidad en la vida real

- Describa un caso, situación o problema donde usaría cifrado simétrico. Defina que algoritmo de cifrado simétrico recomendaría justificando su respuesta.
- Suponga que la recomendación que usted entregó no fue bien percibida por su contraparte y le pide implementar hashes en vez de cifrado simétrico. Argumente cuál sería su respuesta frente a dicha solicitud.

2. Desarrollo de actividades según criterio de rúbrica

2.1. Investiga y documenta los tamaños de clave e IV

2.1.1. DES

- **Key:** la *Key* o llave de DES es de un total de 64 bits (8 bytes), de los cuales 56 bits corresponden a la clave en sí y los 8 bits restantes son utilizados como bits de paridad. Esto resulta en 56 bits efectivos
- **IV:** DES opera con bloques de 64 bits (8 bytes), por lo que su IV es del mismo tamaño (64 bits).

2.1.2. 3DES

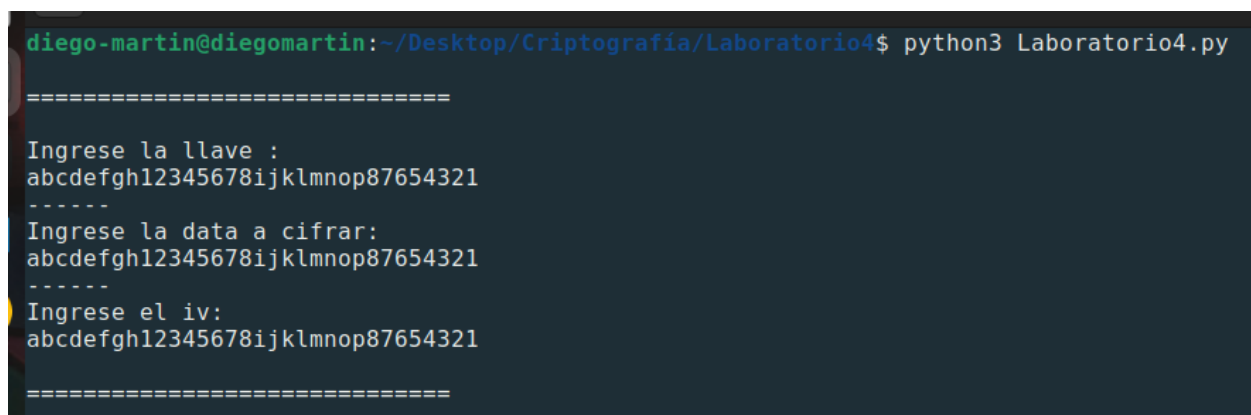
- **Key:** la llave consiste en 192 bits (24 bytes), de las cuales tiene 2 opciones de estar construida:
 1. **Two-key 3DES:** se utilizan 2 claves diferentes K_1 y K_2 de 56 bits (7 bytes) con 8 bits (1 byte) de paridad dando un total de 64 bits (8 bytes) cada una, y se aplica DES de manera cifrar-descifrar-cifrar $E_{K_1}(D_{K_2}(E_{K_1}(\text{Texto plano})))$. Esto resulta en 112 bits efectivos.
 2. **Three-key 3DES:** se utilizan 3 claves diferentes K_1 , K_2 y K_3 de 56 bits (7 bytes) con 8 bits (1 byte) de paridad, dando un total de 64 bits (8 bytes) cada una y se aplica DES para cifrar-descifrar-cifrar $E_{K_1}(D_{K_2}(E_{K_3}(\text{Texto plano})))$. Esto resulta en 168 bits efectivos
- **IV:** Al igual que en DES, 3DES opera con bloques de 64 bits (8 bytes), por lo que su IV es del mismo tamaño.

2.1.3. AES-256

- **Key:** la longitud de la llave o **Key** en AES depende del tipo de AES que se esté utilizando (128, 192 y 256 bits), pero para este caso es de una longitud de 256 bits (32 bytes)
- **IV:** como AES opera con bloques de 128 bits (16 bytes), el IV es del mismo tamaño.

2.2. Solicita datos de entrada desde la terminal

Para este caso se ingresó por terminal *abcdefgh12345678ijklmnop87654321*, para la *llave*, el *IV* y el *mensaje* a cifrar. Este texto o input ingresado se eligió porque cumple los requisitos para funcionar correctamente en AES, 3DES y DES sin tener que ingresar nuevamente los datos.



```
diego-martin@diegomartin:~/Desktop/Criptografía/Laboratorio4$ python3 Laboratorio4.py
=====
Ingrese la llave :
abcdefgh12345678ijklmnop87654321
-----
Ingrese la data a cifrar:
abcdefgh12345678ijklmnop87654321
-----
Ingrese el iv:
abcdefgh12345678ijklmnop87654321
=====
```

Figura 1: Ingreso de datos desde la terminal

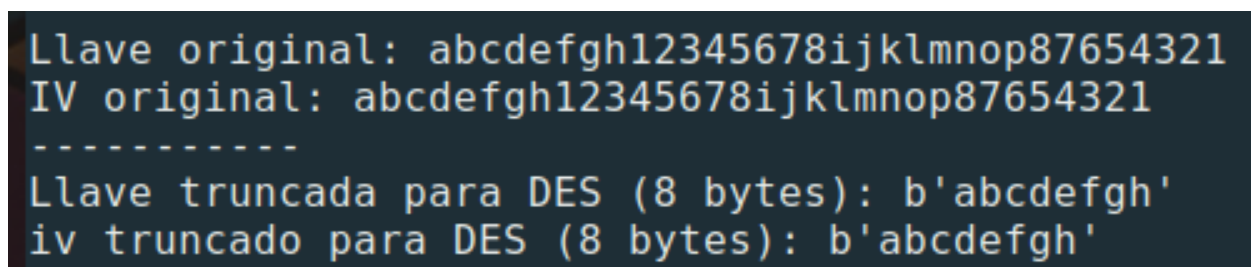
2.3. Valida y ajusta la clave según el algoritmo

Para la llave e IV elegidos *abcdefgh12345678ijklmnop87654321*, se aplican diferentes ajustes según sea necesario para cada algoritmo.

2.3.1. DES

Si la llave es mas larga que 8 bytes, esta se trunca a 8 bytes, mientras que si la llave es menor a 8 bytes se le agregan los bytes necesarios para llegar a 8 de manera aleatoria.

De la misma manera, como el algoritmo trabaja con bloques de 8 bytes el IV es truncado a 8 bytes si es mayor a eso, y si es menor se rellena aleatoriamente hasta cumplir los 8 bytes.



```
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para DES (8 bytes): b'abcdefgh'
iv truncado para DES (8 bytes): b'abcdefgh'
```

Figura 2: Ajuste DES.

2.3.2. 3DES

Si la longitud de la llave se encuentra entre 16 y 24 bytes, esta será truncada a 16, de manera que 8 bytes se utilicen para K1 y 8 bytes para K2 y así repetir los primeros 8 bytes para que la llave cumpla K1-K2-K1 y sea de 24 bytes. Si la longitud de la llave es mayor a 24 bytes, esta es truncada a 24, y si la llave es menor a 16 bytes se genera el resto de bytes aleatoriamente para completar los 24 bytes.

Como el algoritmo trabaja con bloques bloques de 8 bytes al igual que en DES, el IV es truncado a 8 bytes si es mayor a eso y si es menor se rellena aleatoriamente hasta cumplir los 8 bytes.

```
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para 3DES (24 bytes): b'abcdefgh12345678ijklmnop'
IV truncado para DES (8 bytes): b'abcdefgh'
```

Figura 3: Ajuste 3DES.

En este caso se utiliza una llave de 32 bytes y es truncada a 24 bytes y el IV es truncado a sus 8 primeros bytes.

2.3.3. AES-256

Si la longitud de la llave es menor a 32 bytes, se rellenarán los de diferencia aleatoriamente, mientras que si la longitud de la llaves es mayor o igual a 32 será truncada al byte 32.

Como el algoritmo trabaja con bloques de 16 bytes el IV es truncado al byte 16 si es de mayor longitud y si es menor se rellena aleatoriamente hasta cumplir los 16 bytes.

```
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para AES (32 bytes): b'abcdefgh12345678ijklmnop87654321'
IV truncado para AES (16 bytes): b'abcdefgh12345678'
```

Figura 4: Ajuste AES-256.

En este caso se utiliza una llave e IV iguales de largo 32 bytes, por lo que el IV es truncado a 16.

2.4. Implementa el cifrado y descifrado en modo CBC

2.4.1. DES

```
# Cifrado DES en modo CBC
cipher = DES.new(key_for_des, DES.MODE_CBC, iv_for_des)
ciphertext = cipher.encrypt(pad(data, DES.block_size))

print("Data cifrada con DES:", ciphertext)

ciphertext_b64 = base64.b64encode(ciphertext)
```

Figura 5: Cifrado DES modo CBC.

```
def descifrarDES(key_for_des: bytes, ciphertext: bytes, iv_for_des: bytes):
    cipher = DES.new(key_for_des, DES.MODE_CBC, iv_for_des)
    decrypted_padded = cipher.decrypt(ciphertext)
    # Quitar el padding
    try:
        decrypted = unpad(decrypted_padded, DES.block_size)
    except ValueError:
        print("Padding incorrecto. ¿Quizás usaste la llave equivocada?")
        return

    print("Mensaje descifrado DES:", decrypted.decode('utf-8'))
    print("\n=====\\n")
```

Figura 6: Descifrado DES modo CBC.

2.4.2. 3DES

```
# Cifrado DES en modo CBC
cipher = DES3.new(key_for_3des, DES3.MODE_CBC, iv_for_3des)
ciphertext = cipher.encrypt(pad(data, DES3.block_size))

print("Data cifrada con 3DES:", ciphertext)

ciphertext_b64 = base64.b64encode(ciphertext)
```

Figura 7: Cifrado 3DES modo CBC.

```
def descifrar3DES(key_for_3des: bytes, ciphertext: bytes, iv_for_3des: bytes):
    cipher = DES3.new(key_for_3des, DES3.MODE_CBC, iv_for_3des)
    decrypted_padded = cipher.decrypt(ciphertext)
    # Quitar el padding
    try:
        decrypted = unpad(decrypted_padded, DES3.block_size)
    except ValueError:
        print("Padding incorrecto. ¿Quizás usaste la llave equivocada?")
        return

    print("Mensaje descifrado 3DES:", decrypted.decode('utf-8'))
    print("\n=====\\n")
```

Figura 8: Descifrado 3DES modo CBC.

2.4.3. AES

```
# Cifrado AES en modo CBC
cipher = AES.new(key_for_aes, AES.MODE_CBC, iv_for_aes)
ciphertext = cipher.encrypt(pad(data, AES.block_size))

print("Data cifrada con AES:", ciphertext)

ciphertext_b64 = base64.b64encode(ciphertext)
```

Figura 9: Cifrado AES modo CBC.

```
def descifrarAES(key_for_aes: bytes, ciphertext: bytes, iv_for_aes: bytes):
    cipher = AES.new(key_for_aes, AES.MODE_CBC, iv_for_aes)
    decrypted_padded = cipher.decrypt(ciphertext)
    # Quitar el padding
    try:
        decrypted = unpad(decrypted_padded, AES.block_size)
    except ValueError:
        print("Padding incorrecto. ¿Quizás usaste la llave equivocada?")
        return

    print("Mensaje descifrado AES:", decrypted.decode('utf-8'))
    print("\n=====\\n")
```

Figura 10: Descifrado AES modo CBC.

2.5. Compara los resultados con un servicio de cifrado online

Para poder comparar los resultados, se debe utilizar una llave e IV que sean ingresados por terminal y no tengan un padding aleatorio, ya que al ser aleatorio no se podría comparar. Para esto se utiliza como llave, IV y texto a cifrar *abcdefgh12345678ijklmnop87654321* para los 3 algoritmos.

2.5.1. DES

Datos utilizados por el algoritmo y el cifrador online <https://anycrypt.com/crypto/des/>:

- Llave: *abcdefgh*.

- **IV:** *abcdefgh*.
- **Texto:** *abcdefgh12345678ijklmnop87654321*.

```

=====
--- Cifrado DES ---
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para DES (8 bytes): b'abcdefgh'
iv truncada para DES (8 bytes): b'abcdefgh'
-----
Data cifrada con DES: b'\xea\xe5\xfe\xe8\x06[-:\xb1\x9c%\xd5\xf7\x84\xb5UCJaM\xb7\x91V\xd5\xa3\x01!\xb3q\xb6>mc^\xb1\xd9D\xf1#\x1c'
--
Data cifrada a base64: 6uX+6AZbLTqxnCXV94S1VUNKYU23kVbVowEhs3G2Pm1jXrHZRPEjHA==
-----
Mensaje descifrado DES: abcdefgh12345678ijklmnop87654321
=====

```

Figura 11: Resultado DES por terminal.

DES Encryption

Encryption Text

abcdefgh12345678ijklmnop87654321

Encrypted Text

6uX+6AZbLTqxnCXV94S1VUNKYU23kVbVowEhs3G2P
m1jXrHZRPEjHA==

Secret Key

abcdefgh

Encryption Mode

CBC

ECB

IV (optional)

abcdefgh

Output format

Base64

HEX

Encrypt

Figura 12: Resultado DES cifrado online.

DES Decryption

Encrypted Text

6uX+6AZbLTqxnCXV94S1VUNKYU23kVbVowEhs3G2Pm1jXrHZRPEjHA==

Decrypted Text

abcdefgh12345678ijklmnop87654321

Secret Key

abcdefgh

Encryption Mode

CBC

ECB

IV (optional)

abcdefgh

Input format

Base64

HEX

Decrypt

Figura 13: Resultado DES descifrado online.

2.5.2. 3DES

Datos utilizados por el algoritmo y el cifrador online <https://en.metools.info/enencrypt/tripledes277.html>:

- **Llave:** *abcdefgh12345678ijklmnop*.
- **IV:** *abcdefgh*.
- **Texto:** *abcdefgh12345678ijklmnop87654321*.

```

=====
--- Cifrado 3DES ---
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para 3DES (24 bytes): b'abcdefgh12345678ijklmnop'
IV truncado para 3DES (8 bytes): b'abcdefgh'
-----
Data cifrada con 3DES: b'\x91\xa4\xe6W\xb4\xa1\x81F\x9c*Y\x89Y\x88\xc4`Y6\xf2\x8aM\xa3Y\xe27j\x7f\xe4\xab\x12+\xaf\xf6\xbd0\xef\x95V)?'
--
Data cifrada a base64: kaTmV7ShgUacKlmJWYjEYFk28opNo1niN2p/5KsSK6/2vTDvLVpPw==
-----
Mensaje descifrado 3DES: abcdefgh12345678ijklmnop87654321
=====

```

Figura 14: Resultado 3DES por terminal.

Vigenere Cipher Solver Morse Code Translator Triple DES Encryption AES Encryption

Before conversion: ✕

abcdefghijklmnopqrstuvwxyz0123456789

Encryption Mode: CBC Filling method: pkcs7padding Offset: abcdefgh

Key: abcdefgh12345678ijklmnop Triple DES Encrypt Triple DES Decrypt

After conversion: 📄

kaTmV7ShgUacklmJWYjEYFk28opNo1niN2p/5KsSK6/2vTDvIVYpPw==

Figura 15: Resultado 3DES cifrado online.

Vigenere Cipher Solver Morse Code Translator Triple DES Encryption AES Encryption

Before conversion: ✕

kaTmV7ShgUacklmJWYjEYFk28opNo1niN2p/5KsSK6/2vTDvIVYpPw==

Encryption Mode: CBC Filling method: pkcs7padding Offset: abcdefgh

Key: abcdefgh12345678ijklmnop Triple DES Encrypt Triple DES Decrypt

After conversion: 📄

abcdefghijklmnopqrstuvwxyz0123456789

Figura 16: Resultado 3DES descifrado online.

2.5.3. AES-256

Datos utilizados por el algoritmo y el cifrador online <https://en.metools.info/enencrypt/aes276.html>:

- **Llave:** *abcdefghijklmnopqrstuvwxyz0123456789*.

- **IV:** *abcdefgh12345678.*
- **Texto:** *abcdefgh12345678ijklmnop87654321.*

```

--- Cifrado AES ---
Llave original: abcdefgh12345678ijklmnop87654321
IV original: abcdefgh12345678ijklmnop87654321
-----
Llave truncada para AES (32 bytes): b'abcdefgh12345678ijklmnop87654321'
IV truncado para AES (16 bytes): b'abcdefgh12345678'
-----
Data cifrada con AES: b'4\xe0\x8d\xa8o\x8d\x1f\xffN\xc8aI\xee\r\x13\xa7\xf7\x02\x87t\xfc\xdeY\xbb\xff\xd20\xc5\x0c\x88\xae_\xebj\x1c\xdd\xde1\xa0\x8d\x8e\xcdm\xb9\x177fI'
Data cifrada a base64: NOCNqG+NH/9OyGFp7g0Tp/cCh3T83lm7/9IwxQyIrl/rfBzd3jGgjY7NbbkXN2ZJ
Mensaje descifrado AES: abcdefgh12345678ijklmnop87654321
=====
  
```

Figura 17: Resultado AES-256 por terminal.

Vigenere Cipher Solver
Morse Code Translator
Triple DES Encryption
AES Encryption and Decryption

Before conversion: ✖

abcdefgh12345678ijklmnop87654321

Encryption Mode: CBC
Filling method: pkcs7padding
Offset: abcdefgh12345678

Key: abcdefgh12345678ijklmnop8
AES Encrypt
AES Decrypt

After conversion: 📄

NOCNqG+NH/9OyGFp7g0Tp/cCh3T83lm7/9IwxQyIrl/rfBzd3jGgjY7NbbkXN2ZJ

Figura 18: Resultado AES-256 cifrado online.

Vigener Cipher Solver Morse Code Translator Triple DES Encryption AES Encryption and Decryption

Before conversion: ✕

NOCNqG+NH/9OyGFp7g0Tp/cCh3T83lm7/9lwxQyirl/rfBzd3jGgjY7NbbkXN2ZJ

Encryption Mode: CBC Filling method: pkcs7padding Offset: abcdefgh12345678

Key: abcdefgh12345678ijklmnop AES Encrypt AES Decrypt

After conversion: 📄

abcdefgh12345678ijklmnop87654321

Figura 19: Resultado AES-256 descifrado online.

2.6. Describe la aplicabilidad del cifrado simétrico en la vida real

El cifrado simétrico es una parte esencial de la criptografía debido a su velocidad y eficiencia, siendo la alternativa ideal para cifrar volúmenes grandes de datos rápidamente. Debido a esta eficiencia y velocidad es que se utiliza para proteger la confidencialidad para comunicaciones seguras, cifrado de archivos, transacciones financieras. Aún cuando el cifrado simétrico es muy eficiente, su principal problema es el manejo de sus claves, las cuales deben ser secretas y compartidas de forma protegida o se pierde el sentido del cifrado y su seguridad. Algunas de las aplicaciones más comunes del cifrado asimétrico son:

- Navegación Web con HTTPS/TLS.
- Mensajería instantánea y comunicación segura (llamadas).
- VPNs y seguridad en redes inalámbricas.
- Cifrado de archivos.
- Transacciones financieras.

Conclusiones y comentarios

El cifrado simétrico es clave en el ámbito de seguridad digital, debido a que es muy rápido y eficiente con grandes volúmenes de datos, siendo utilizado ampliamente para muchas de las aplicaciones y servicios que utilizamos día a día.

Dentro de los algoritmos de cifrado simétrico, AES es el más recomendado, debido a su robustez, ya que al compararlo con los otros algoritmos disponibles, y así AES se convirtió en el estándar actual de la mayoría de sistemas.

Si bien estos algoritmos son seguros, no basta con definir llaves e IVs cualesquiera, si no que también hay que hacer una correcta gestión de estas claves e IVs, de manera que no sean predecibles ni que se vean vulneradas al momento de intercambiarlas.

Con la implementación en Python de los algoritmos DES y 3DES se demuestra que ambos son funcionalmente operativos, pero que tienen limitaciones. DES con su llave de 56 bits queda obsoleto por su casi nula robustez. 3DES si bien es una mejora de DES al aplicarlo multiples veces, sigue teniendo las vulnerabilidades y problemas de este, siendo un ejemplo claro el utilizar una llave de 24 bytes con todos iguales (3 llaves iguales de 8 bytes), que básicamente degenera el 3DES a un DES simple. Es por esto que AES-256 surge como el mejor algoritmo al tener una llave extremadamente larga en comparación a DES y 3DES y aún incluyendo el IV para su cifrado.