

Informe Laboratorio 4

Sección 3

Matías González Rojas
e-mail: matias.gonzalez8@mail.udp.cl

Octubre 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
2.2. Solicita datos de entrada desde la terminal	3
2.3. Valida y ajusta la clave según el algoritmo	5
2.4. Implementa el cifrado y descifrado en modo CBC	6
2.5. Compara los resultados con un servicio de cifrado online	7
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real	8

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 entrego 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

Para la implementación del cifrado simétrico se consideraron tres algoritmos: DES, 3DES y AES-256, todos en modo CBC. Cada uno requiere un tamaño específico de clave y de vector de inicialización (IV), los cuales determinan la seguridad y funcionamiento del cifrado.

- **DES:** utiliza una clave de 8 bytes (64 bits) y un IV del mismo tamaño. Su corta longitud de clave lo hace vulnerable frente a ataques de fuerza bruta.
- **3DES:** emplea tres claves de 8 bytes, totalizando 24 bytes (192 bits), con un IV de 8 bytes. Mejora la seguridad de DES, aunque con menor eficiencia.
- **AES-256:** usa una clave de 32 bytes (256 bits) y un IV de 16 bytes. Es actualmente uno de los algoritmos más seguros y eficientes.

En el programa desarrollado, los tamaños definidos fueron los siguientes:

```
AES -> key_size = 32 bytes, iv_size = 16 bytes
DES  -> key_size = 8 bytes,  iv_size = 8 bytes
3DES -> key_size = 24 bytes, iv_size = 8 bytes
```

El código ajusta automáticamente las claves e IV ingresados por el usuario al tamaño requerido por cada algoritmo.

2.2. Solicita datos de entrada desde la terminal

Una vez ejecutado el código, este solicitará el ingreso de la clave, IV y texto a cifrar del primer algoritmo AES-256, entregará los resultados del cifrado y descifrado, luego procederá con el siguiente algoritmo realizando el mismo proceso. Así hasta terminar la ejecución del código.

```
● postroy@postroy:~/Universidad/CRIPTOGRAFÍA Y SEGURIDAD EN REDES/Laboratorios/LAB_4$ python3 lab4.py
--- Cifrador/Descifrador (AES-256, DES, 3DES) ---

=== AES-256 ===
Ingrese la Key (texto): passwordpasswordpasswordpassword
Ingrese el IV (texto): vectorinicial16by
Ingrese el Texto a cifrar: Este es el mensaje para AES

[Ajuste] Clave original (bytes): b'passwordpasswordpasswordpassword'
[Ajuste] Clave final utilizada (32 bytes): 70617373776f726470617373776f726470617373776f726470617373776f7264
[Ajuste] IV original (bytes): b'vectorinicial16by'
[Ajuste] IV final utilizado (16 bytes): 766563746f72696e696369616c313662

--- Cifrando ---
Texto original: Este es el mensaje para AES
Texto Cifrado (hex): 50aa7b46f40235cf0e8b6e6170d9a0feb7f3de281aa310e953d3a16c2d17ad18

--- Descifrando ---
Texto Descifrado: Este es el mensaje para AES

VERIFICACIÓN: Éxito. El texto descifrado coincide con el original.
```

Figura 1: Datos solicitados para AES-256.

```
=== DES ===
Ingrese la Key (texto): password
Ingrese el IV (texto): mivector
Ingrese el Texto a cifrar: mensaje de prueba 1

[Ajuste] Clave original (bytes): b'password'
[Ajuste] Clave final utilizada (8 bytes): 70617373776f7264
[Info] IV utilizado (8 bytes): 6d69766563746f72

--- Cifrando ---
Texto original: mensaje de prueba 1
Texto Cifrado (hex): 10c76837db54273b1252c780a6e2a80e43ddb0f5fd88284a

--- Descifrando ---
Texto Descifrado: mensaje de prueba 1

VERIFICACIÓN: Éxito. El texto descifrado coincide con el original.
```

Figura 2: Datos solicitados para DES.

```
=== 3DES ===
Ingrese la Key (texto): clave123clave456clave789
Ingrese el IV (texto): vectorIV
Ingrese el Texto a cifrar: Mensaje de prueba para 3DES

[Ajuste] Clave original (bytes): b'clave123clave456clave789'
[Ajuste] Clave final utilizada (24 bytes): 626d617664313232626d617664343437626d617664373838
[Info] IV utilizado (8 bytes): 766563746f724956

--- Cifrando ---
Texto original: Mensaje de prueba para 3DES
Texto Cifrado (hex): e1af05b8d3ac03f36a812c9fc8ca2d79ccc088d09eaf96e240f6a6ff4de0bcb2

--- Descifrando ---
Texto Descifrado: Mensaje de prueba para 3DES

VERIFICACIÓN: Éxito. El texto descifrado coincide con el original.
```

Figura 3: Datos solicitados para 3DES.

Como se pueden observar en las Figuras 1, 2, 3 se piden las claves correspondientes por terminal para luego realizar el proceso del algoritmo correspondiente.

2.3. Valida y ajusta la clave según el algoritmo

En el código se realiza la validación y ajuste de la clave en tres partes distintas.

```
def ajustar_clave(key_bytes, tamaño_esperado):
    if len(key_bytes) < tamaño_esperado:
        padding = get_random_bytes(tamaño_esperado - len(key_bytes))
        return key_bytes + padding
    elif len(key_bytes) > tamaño_esperado:
        return key_bytes[:tamaño_esperado]
    return key_bytes
```

Figura 4: Función que ajusta la clave.

Como se puede observar en la Figura 4, la función se dedica a ajustar la clave, es decir, rellenar o truncar según el tamaño del algoritmo que se esté ejecutando en el momento.

```

clave_final = ajustar_clave(key_bytes, key_size)
if ajustar_paridad:
    try:
        clave_final = cipher_module.adjust_key_parity(clave_final)
    except AttributeError:
        pass
print(f"\n[Ajuste] Clave original (bytes): {key_bytes}")
print(f"[Ajuste] Clave final utilizada ({len(clave_final)} bytes): {clave_final.hex()}")

```

Figura 5: Ajuste de paridad.

En la Figura 5 se puede observar cómo esta sección del código convierte la clave ingresada a la longitud correcta con la función de la Figura 4, luego, si el algoritmo lo requiere, corrige la paridad de la clave y finalmente lo muestra en pantalla.

```

def main():
    print("--- Cifrador/Descifrador (AES-256, DES, 3DES) ---")

    ejecutar_algoritmo("AES-256", key_size=32, iv_size=16, cipher_module=AES, ajustar_paridad=False)
    ejecutar_algoritmo("DES", key_size=8, iv_size=8, cipher_module=DES, ajustar_paridad=True)
    ejecutar_algoritmo("3DES", key_size=24, iv_size=8, cipher_module=DES3, ajustar_paridad=True)

```

Figura 6: Definir que algoritmos necesitan paridad.

Finalmente, en la Figura 6 es donde se define qué algoritmos requieren paridad.

2.4. Implementa el cifrado y descifrado en modo CBC

```

print("\n--- Cifrando ---")
cipher_encrypt = cipher_module.new(clave_final, cipher_module.MODE_CBC, iv_final)
texto_cifrado = cipher_encrypt.encrypt(pad(texto_bytes, iv_size))
print(f"Texto original: {texto_str}")
print(f"Texto Cifrado (hex): {texto_cifrado.hex()}")

print("\n--- Descifrando ---")
cipher_decrypt = cipher_module.new(clave_final, cipher_module.MODE_CBC, iv_final)
texto_descifrado_padded = cipher_decrypt.decrypt(texto_cifrado)
texto_descifrado_bytes = unpad(texto_descifrado_padded, iv_size)
texto_descifrado = texto_descifrado_bytes.decode('utf-8')
print(f"Texto Descifrado: {texto_descifrado}")

```

Figura 7: Cifrado y Descifrado con CBC.

En la Figura 7 se puede observar cómo se utilizó el modo CBC para realizar el proceso de cifrado. Se creó un objeto de cifrado con el algoritmo seleccionado (AES, DES o 3DES),

utilizando la clave e IV definidos por el usuario. Luego, el texto fue rellenado con la función `pad()` y cifrado con `encrypt()`.

En el descifrado se siguió el mismo procedimiento, creando un nuevo objeto en modo CBC con la misma clave e IV. Posteriormente, se utilizó la función `decrypt()` para obtener los datos originales y `unpad()` para eliminar el relleno añadido durante el cifrado.

2.5. Compara los resultados con un servicio de cifrado online

Para este apartado del informe se va a utilizar la página <https://www.devglan.com/online-tools/aes-encryption-decryption> para realizar el cifrado y descifrado con el algoritmo AES, utilizando el mismo ejemplo de la Figura 1.

The screenshot shows the 'AES Encryption' web interface. It includes several input fields and a dropdown menu for configuration. The 'Enter Plain Text to Encrypt' field contains the text 'Este es el mensaje para AES'. The 'Select Cipher Mode of Encryption' dropdown is set to 'CBC'. The 'Select Padding' dropdown is set to 'PKCS5Padding'. The 'Enter IV (Optional)' field contains 'vectorinicial16b'. The 'Key Size in Bits' dropdown is set to '256'. The 'Enter Secret Key' field contains 'passwordpasswordpasswordpassword'. The 'Output Text Format' section has 'Base64' selected with a radio button, and 'Hex' is also present. A black 'Encrypt' button is located below the key field. At the bottom, the 'AES Encrypted Output' field displays the hexadecimal result: '50AA7B46F40235CF0E8B6E6170D9A0FEB7F3DE281AA310E953D3A16C2D17AD18'.

AES Encryption

Enter Plain Text to Encrypt

Este es el mensaje para AES

Select Cipher Mode of Encryption ?

CBC

Select Padding ?

PKCS5Padding

Enter IV (Optional) ?

vectorinicial16b

Key Size in Bits ?

256

Enter Secret Key ?

passwordpasswordpasswordpassword

Output Text Format ☐ Base64 ☒ Hex

Encrypt

AES Encrypted Output

50AA7B46F40235CF0E8B6E6170D9A0FEB7F3DE281AA310E953D3A16C2D17AD18

Figura 8: Cifrado con servicio online.

Como se puede observar en la Figura 8, el resultado obtenido es el mismo que el esperado,

confirmando que el código implementa de manera correcta el algoritmo AES-256.

AES Decryption

AES Encrypted Text

50AA7B46F40235CF0E8B6E6170D9A0FEB7F3DE281AA310E953D3A16C2D17AD18

Select Cipher Mode of Decryption ?

CBC

Select Padding ?

PKCS5Padding

Enter IV Used During Encryption(Optional) ?

vectorinicial16b

Key Size in Bits ?

256

Enter Secret Key used for Encryption ?

passwordpasswordpasswordpassword

Output Text Format ☒ Plain-Text ☐ Base64

Decrypt

AES Decrypted Output

Este es el mensaje para AES

Figura 9: Descifrado con servicio online.

Ahora se procede a realizar el proceso de descifrado con el resultado de la Figura 8 y podemos observar en la Figura 9 que el resultado corresponde al texto original que fue cifrado, indicando nuevamente que el código fue correctamente implementado. Esto es gracias a la librería PyCryptodome, la cual sigue las mismas especificaciones del estándar AES definido por el NIST (FIPS 197). El modo de operación CBC y el uso de un IV del tamaño correcto (16 bytes) aseguran que el proceso de cifrado y descifrado sea idéntico al de cualquier herramienta que cumpla el mismo estándar.

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

El cifrado simétrico se utiliza ampliamente para proteger información que debe mantenerse privada o segura durante su transmisión o almacenamiento. Un ejemplo común es el

cifrado de datos en aplicaciones de mensajería o en conexiones seguras como HTTPS, donde se usa una misma clave para cifrar y descifrar los mensajes.

Por ejemplo, cuando un usuario envía su contraseña a un servidor, el texto se cifra con un algoritmo simétrico como AES antes de ser transmitido, evitando que terceros puedan leer la información en caso de interceptarla.

Conclusiones y comentarios

El desarrollo de este laboratorio permitió implementar de manera práctica los algoritmos de cifrado simétrico DES, 3DES y AES-256, utilizando la librería pycryptodome en Python. Se cumplió con todos los requisitos, incluyendo la solicitud de datos por terminal y el uso del modo de operación CBC para todos los cifrados.

La parte más importante del laboratorio fue comparar los resultados con un servicio de cifrado en línea. Al usar los mismos datos (texto, clave e IV) con el algoritmo AES-256, el resultado del cifrado fue exactamente el mismo que el obtenido con el programa. Esto demuestra que el código funciona correctamente y sigue las reglas del estándar AES, usando el mismo tipo de relleno (padding) y el modo CBC, tal como se aplica en sistemas reales de seguridad.

Github: <https://github.com/PostRoy/lab4-crypto>

Referencias

- [1] Instituto Nacional de Estándares y Tecnología. (1999). *FIPS 46-3, Data Encryption Standard (DES)*. Departamento de Comercio de EE. UU. <https://csrc.nist.gov/pubs/fips/46-3/final>
- [2] Instituto Nacional de Estándares y Tecnología. (2001). *FIPS 197, Advanced Encryption Standard (AES)*. Departamento de Comercio de EE. UU. <https://csrc.nist.gov/pubs/fips/197/final>
- [3] The PyCryptodome Project. (s.f.-a). *AES*. En PyCryptodome 3.23.0 documentation. Recuperado el 4 de noviembre de 2025, de <https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html>
- [4] The PyCryptodome Project. (s.f.-b). *Single DES*. En PyCryptodome 3.23.0 documentation. Recuperado el 4 de noviembre de 2025, de <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des.html>

- [5] The PyCryptodome Project. (s.f.-c). *Triple DES (3DES)*. En PyCryptodome 3.23.0 documentation. Recuperado el 4 de noviembre de 2025, de <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des3.html>