Informe Laboratorio 4

Sección 3

Alumno Felipe Ormazábal
e-mail: felipe.ormazabal1@mail.udp.cl

Octubre de 2025

Índice

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

Algoritmo	**Tamaño de Clave (Bits)**	**Tamaño de vector(IV) (Bits)**
DES	56 + 8 (64)	64
3DES	128 0 192	64
AES-256	256 bits	128 bits

- DES: Usa una clave de solo 56 bits, es lento y ya esta obsoleto.
- 3DES: Usa una clave de 128 o 196 bits lo que lo hace un poco mas seguro, es mucho mas lento en comparación a DES debido a que aplica DES 3 veces, no esta obsoleto, pero no es recomendado.
- AES-256: Con una clave de 256 bits es mas seguros que los dos mencionados anteriormente,donde se hace uso de permutación, sustitución y mezcla, siendo también mas rápido

2.2. Solicita datos de entrada desde la terminal

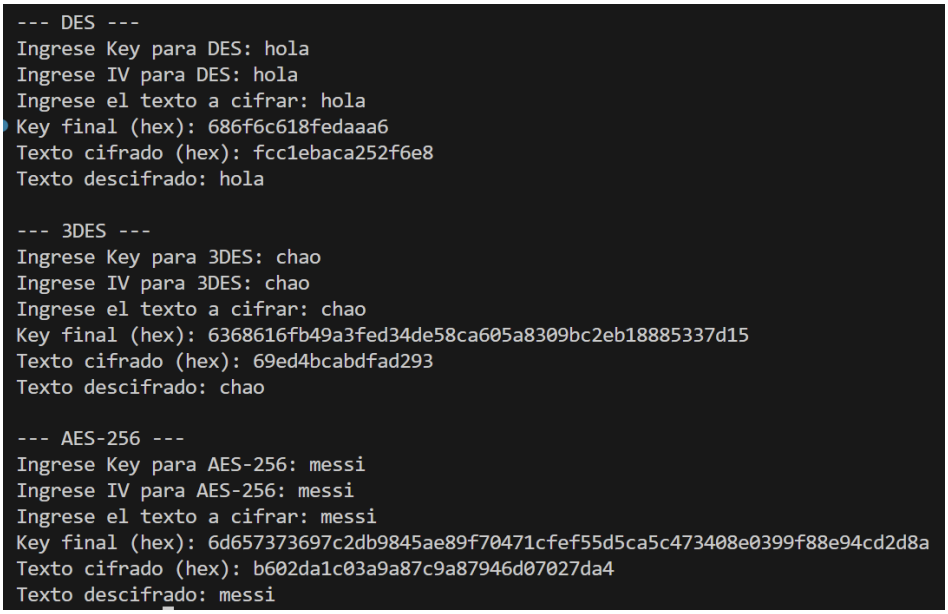

Figura 1: Datos de entrada desde terminal

Como se puede ver en la figura 1 se piden las llaves, el vector de inicialización y el texto a cifrar para cada algoritmo en la terminal.

2.3. Valida y ajusta la clave según el algoritmo

Para ajustar la clave segun el algoritmo se llama a una función creada llamada adjust_input la cual se puede ver en la figura 2 donde dependiendo del algoritmo se se le pasa el tamaño de calve que debería de tener y obviamente la clave ingresada, donde si es mas chica se rellena con bytes aleatorios y si es mas larga esta trunca.

```python
def adjust_input(data_bytes, expected_size):
    len_data = len(data_bytes)

    if len_data == expected_size:
        return data_bytes

    if len_data < expected_size:
        padding_needed = expected_size - len_data
        random_padding = get_random_bytes(padding_needed)
        return data_bytes + random_padding

    if len_data > expected_size:
        return data_bytes[:expected_size]
```

Figura 2: Funcion adjust_input

2.4. Implementa el cifrado y descifrado en modo CBC

Para el cifrado y descifrado con CBC se usa la función de la biblioteca de *pycriptodome* donde solo le pasamos la clase del algoritmo y ya hace el cifrado o descifrado correspondiente, esto se puede ver en la figura 3

```python
    cipher_encrypt = cipher_class.new(final_key, cipher_class.MODE_CBC, final_iv)
    padded_text = pad(plaintext_bytes, block_size)
    ciphertext = cipher_encrypt.encrypt(padded_text)
    print(f"Texto cifrado (hex): {ciphertext.hex()}")


    cipher_decrypt = cipher_class.new(final_key, cipher_class.MODE_CBC, final_iv)
    decrypted_padded_text = cipher_decrypt.decrypt(ciphertext)
    decrypted_text = unpad(decrypted_padded_text, block_size)
```

Figura 3: Cifrado y descifrado en modo cbc

Finalmente para correr el código en el main se llama a la función que tiene todo con cada algoritmo y sus especificaciones las cuales serian el tamaño de las llaves y vectores de inicialización en bytes.

```python
def main():
    cifrar_descifrar_demo("DES", DES, 8, 8)
    cifrar_descifrar_demo("3DES", DES3, 24, 8)
    cifrar_descifrar_demo("AES-256", AES, 32, 16)
```

Figura 4: Main

2.5. Compara los resultados con un servicio de cifrado online

Para este espacio se uso la paginahttps://anycript.com/crypto/des para cifrar online, donde se puede ver el resultado en la figura 5

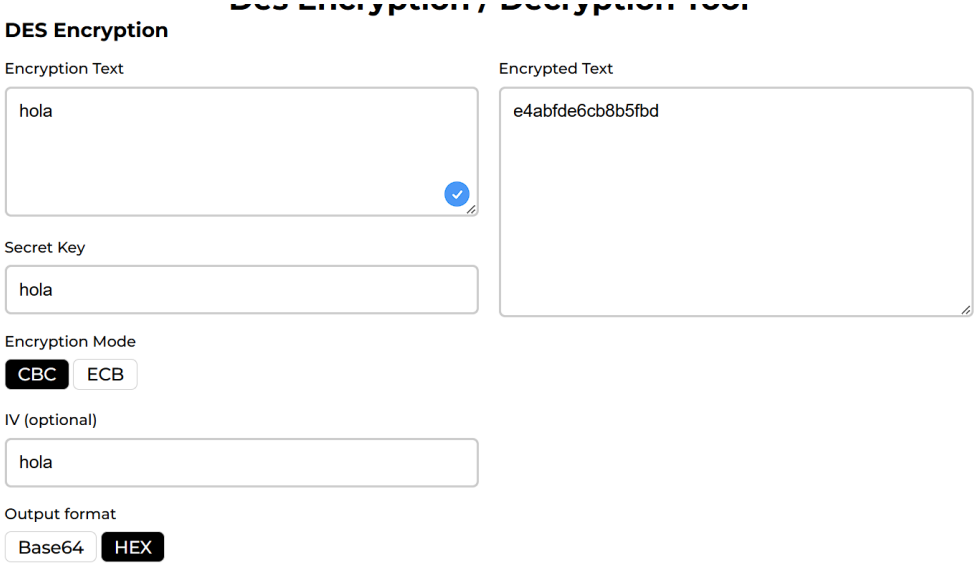

Figura 5: Encriptación online

y el resultado del codigo hecho es el siguiente: figura 6

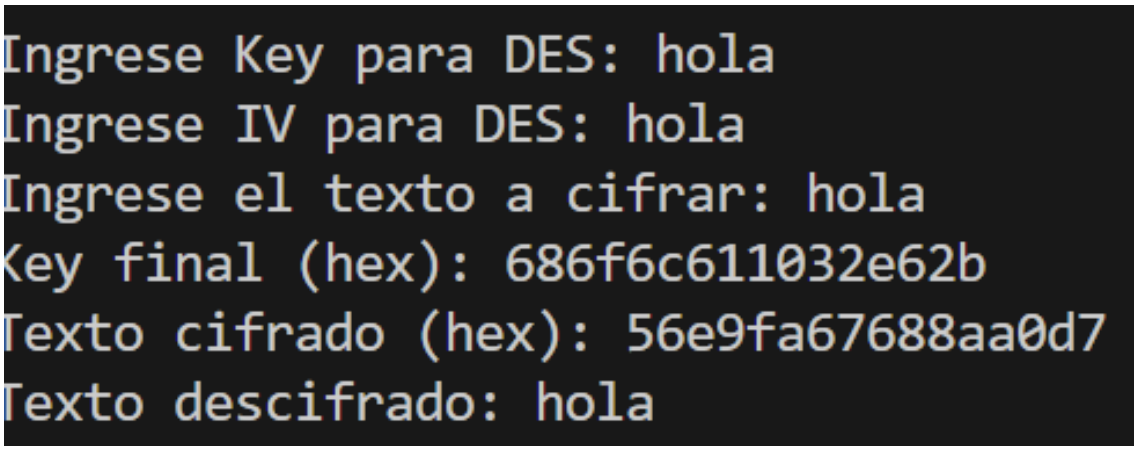

Figura 6: Encriptación por código

Como se puede observar no dan el mismo resultado, esto es posible dado a que usé una llave y una IV de un tamaño mas chico del necesario haciendo que en el caso del código creado este lo rellena con bytes aleatorios, sin embargo la encriptación de la pagina online puede estar usando otro tipo de padding haciendo que de un texto cifrado distinto en cada algoritmo.
Para comprobar esto un poco mas voy a probar nuevamente el mismo texto a cifrar, pero ahora con una llave y IV de largo 8, los resultados en cada cifrado se pueden ver en las siguientes figuras 7 y 8

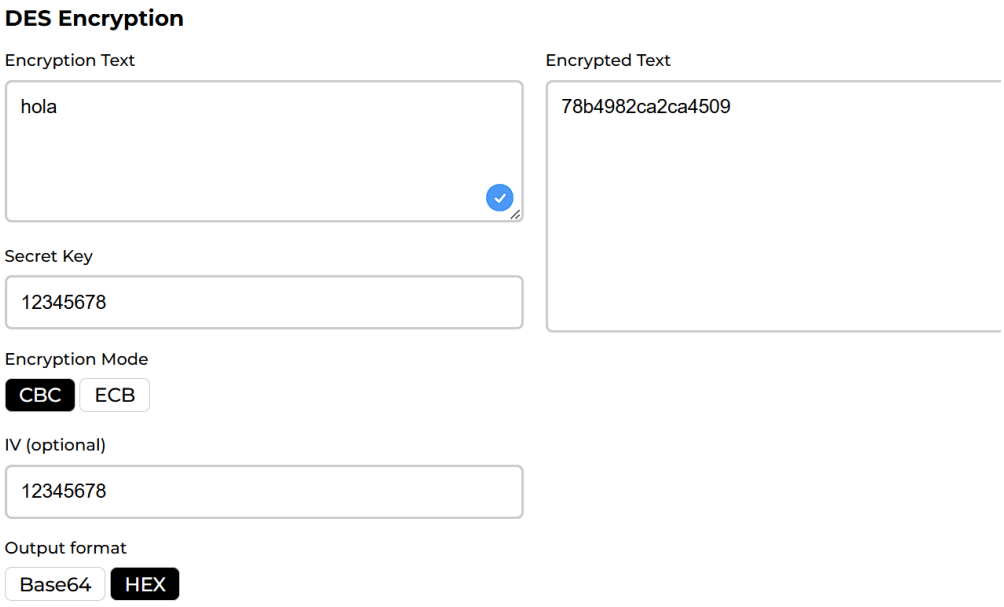

Figura 7: Encriptación online

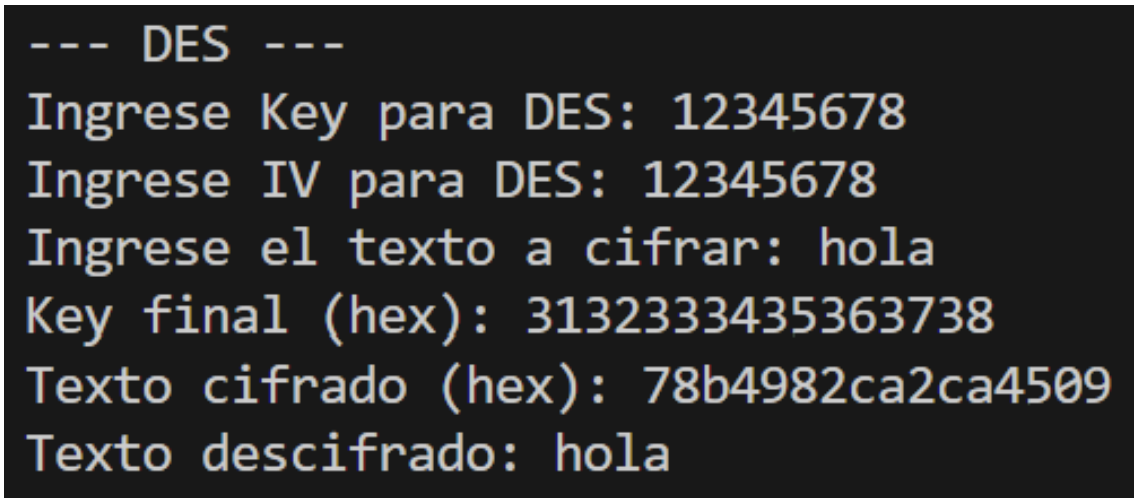

Figura 8: Encriptación por código

Ahora como se puede ver en las imágenes el texto cifrado concuerda en ambos.

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

El cifrado simétrico en la vida real se puede usar mayormente en la protección de datos, generalmente de contraseñas un ejemplo de esto seria usar openssl y encriptar la contraseña con AES-256, igualmente esto va para cualquier tipo de mensajes que tienen que ser enviados a un destinatario sin que se sepa entremedio lo que se esta enviando

Conclusiones y comentarios

En este laboratorio se aprendió a crear, usar y ver el funcionamiento interno de algoritmos de cifrado simétricos en modo CBC, lo que puede venir muy bien para un futuro cuando se tenga que trabajar con encrioptación de datos en la vida real.

3. Referencias

https://crypto.stackexchange.com/questions/76845/does-the-initialization-vector-in-des-have-to-be-8-bytes-long
https://crypto.stackexchange.com/questions/24664/key-sizes-supported-by-3des
https://stackoverflow.com/questions/31132162/what-size-of-initialization-vector-needed-for-aes-256-encryption-in-java
Repositorio: https://github.com/Xxfelipe89xX/lab04-Cripto