

Informe Laboratorio 4

Sección 1

Diego Jaime Pastroián Márquez
e-mail: diego.pastrian@mail.udp.cl

Noviembre de 2024

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	6
2.4. Implementa el cifrado y descifrado en modo CBC	9
2.5. Compara los resultados con un servicio de cifrado online	12
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real	14

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

La información extraída de los tamaños de la key y el IV de los métodos de cifrado DES, AES-256 y 3DES provienen de la documentación de la librería *pycryptodome*, la cual sirve para aplicar tales cifrados en código Python.

- **DES [1]:**
 - *Tamaño de clave* La clave para el cifrado debe ser de 8 bytes.
 - *Tamaño de IV* El vector de inicialización tiene un tamaño de 8 bytes para el modo CBC.
- **AES-256 [2]:**
 - *Tamaño de clave:* La clave para el cifrado debe ser de 32 Bytes en AES-256, pues el nombre del algoritmo indica el uso de una clave de 256 bits (32 Bytes).
 - *Tamaño de IV:* El vector de inicialización tiene un tamaño de 16 Bytes para el modo CBC.
- **3DES [3]:**
 - *Tamaño de clave:* La clave para el cifrado debe ser de 24 Bytes de largo para utilizar DES-EDE3-CBC.
 - *Tamaño de IV* El vector de inicialización tiene un tamaño de 8 bytes para el modo CBC.

2.2. Solicita datos de entrada desde la terminal

El código completo a explicar se encuentra en el repositorio github adjunto. Para efectos de este documento, se explicarán y mostrarán las partes más importantes de este. Lo primero a realizar es el ingreso de los parámetros a utilizar; la Key, el Vector de inicialización IV y el texto a cifrar. La función *mostrar_pagina_key_iv* se encarga del ingreso del texto plano a cifrar y la selección del algoritmo de cifrado a utilizar. En base a esta selección, se cambia el valor de las variables *iv_length* y *key_length*, las cuales contienen el largo del vector de inicialización y de la clave respectivamente. Si no se ingresa un texto a cifrar, o no se selecciona

el tipo de cifrado, entonces se muestra un mensaje indicando que se deben completar todos los campos y el código termina. Lo explicado anteriormente se puede observar en la figura 1.

A screenshot of a code editor with a dark background and light blue text. The code defines a function named 'mostrar_pagina_key_iv()' which takes global variables 'key_length' and 'iv_length'. It prompts the user for a text input and a selection. If no text is entered, it shows a warning. If selection 1 is chosen, it sets 'metodo' to 'DES', 'key_length' to 8, and 'iv_length' to 8. If selection 2 is chosen, it sets 'metodo' to 'AES-256', 'key_length' to 32, and 'iv_length' to 16. If selection 3 is chosen, it sets 'metodo' to '3DES', 'key_length' to 24, and 'iv_length' to 16. For any other selection, it shows a warning. The code is as follows:

```
key_length = 0
iv_length = 0

def mostrar_pagina_key_iv():
    global key_length, iv_length
    texto = texto_entrada.get()
    seleccion = opcion_var.get()

    if not texto:
        messagebox.showwarning("Advertencia", "Por favor, ingrese un
        texto antes de seleccionar.")
        return

    if seleccion == 1:

        metodo = "DES"
        key_length = 8
        iv_length = 8

    elif seleccion == 2:

        metodo = "AES-256"
        key_length = 32
        iv_length = 16

    elif seleccion == 3:
        metodo = "3DES"
        key_length = 24
        iv_length = 16
    else:
        messagebox.showwarning("Advertencia", "Por favor, seleccione un
        método.")
        return
```

Figura 1: Función para realizar la selección de los parámetros

Después de lo anterior, se realiza el código asociado al ingreso manual de la clave y el vector de inicialización mediante **key_entry** y **iv_entry** en el código de la figura 2.

```
label_metodo = tk.Label(ventana, font=("Arial", 12, "bold"),
bg="#f0f0f5", fg="#333333")

label_key = tk.Label(ventana, text=f"Ingrese la Clave.", font=("Arial",
12), bg="#f0f0f5")
key_entry = tk.Entry(ventana, font=("Arial", 12), width=30)

label_iv = tk.Label(ventana, text=f"Ingrese el Vector de Inicialización
IV.", font=("Arial", 12), bg="#f0f0f5")
iv_entry = tk.Entry(ventana, font=("Arial", 12), width=30)
```

Figura 2: Ingreso de la key y el IV.

La compilación de lo explicado anteriormente se observa en la figura 3.

Primeramente se ingresa el texto a cifrar para posteriormente seleccionar el algoritmo de cifrado a utilizar.

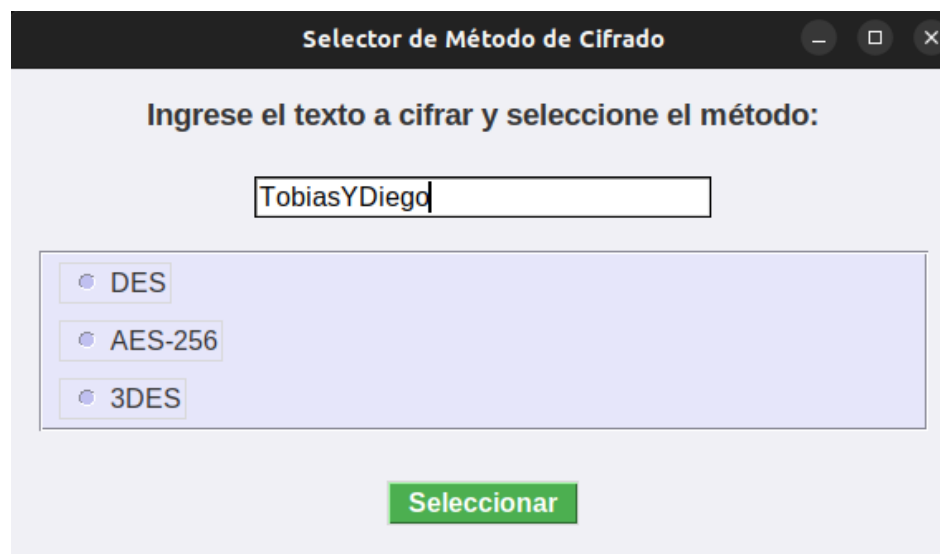


Figura 3: Ingreso del texto a cifrar y selección del cifrado a utilizar.

Luego, se realiza el ingreso de la clave y el bloque de inicialización IV en la figura 4.

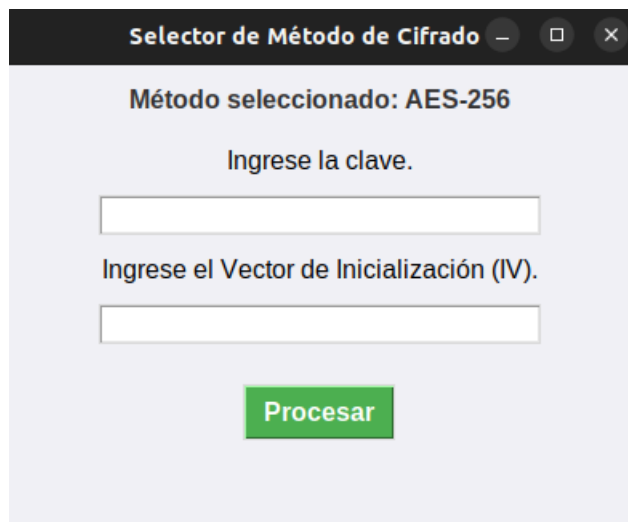
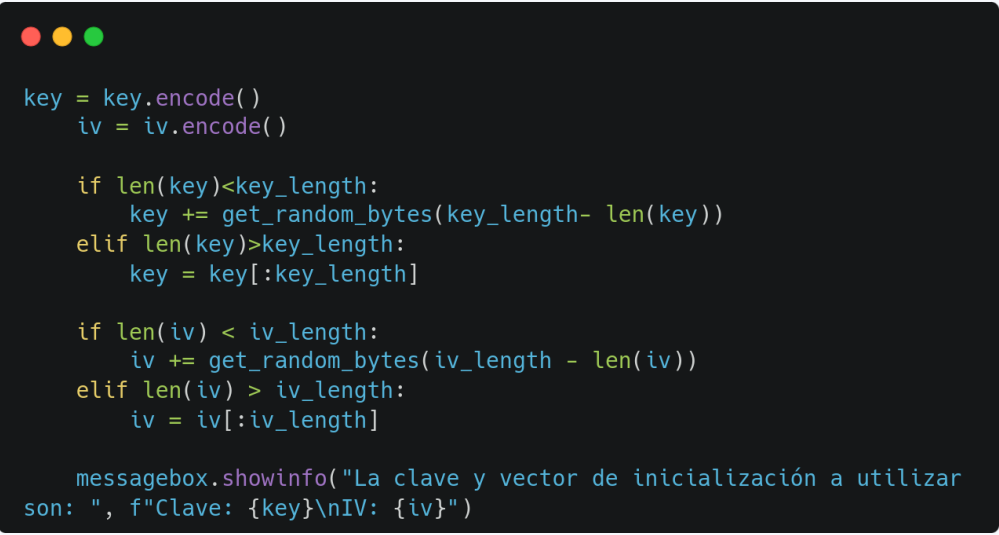


Figura 4: Ingreso de la clave y el IV.

2.3. Valida y ajusta la clave según el algoritmo

Anteriormente se definieron los tamaños necesarios para la clave (*key_length*) y el vector de inicialización(*iv_length*) en base al algoritmo seleccionado. En base a esto, la clave y el IV ingresados deben adaptarse a tales tamaños, por lo que es necesario implementar en el código el ajuste necesario. Si la clave o el vector de inicialización (IV) ingresados contienen menos bytes que los requeridos, utilizando el método *get_random_bytes* de la librería *PyCryptodome* se completarán con bytes aleatorios hasta alcanzar el tamaño necesario. Si contienen más bytes, se truncarán para ajustarse al tamaño requerido. Para lo anterior, la *key* y el *iv* ingresados se pasan a Bytes en el código para su posterior manipulación. El código asociado a lo anterior puede observarse en la Figura 5.

A screenshot of a Python code editor window with a dark background and light-colored text. The code is for adjusting the length of a key and an initialization vector (IV) to 8 bytes. It includes comments in Spanish. The code uses `key.encode()` and `iv.encode()` to convert strings to bytes. It then uses conditional logic to either append random bytes (`get_random_bytes`) if the length is less than 8 or truncate the string (`key[:key_length]`) if it is greater than 8. Finally, it uses `messagebox.showinfo` to display the resulting key and IV.

```
key = key.encode()  
iv = iv.encode()  
  
if len(key)<key_length:  
    key += get_random_bytes(key_length- len(key))  
elif len(key)>key_length:  
    key = key[:key_length]  
  
if len(iv) < iv_length:  
    iv += get_random_bytes(iv_length - len(iv))  
elif len(iv) > iv_length:  
    iv = iv[:iv_length]  
  
messagebox.showinfo("La clave y vector de inicialización a utilizar  
son: ", f"Clave: {key}\nIV: {iv}")
```

Figura 5: Código de ajuste del tamaño de IV y la Clave.

Posterior al ajuste, en pantalla se muestran todos los parámetros asociados al cifrado. La Clave y el IV se muestran en formato de Bytes.

- DES: En la Figura 6 se observa el ajuste de los parámetros mediante el ingreso de una clave “123456” y un IV “12345678999”. Como la clave y el IV en DES deben ser de 8 Bytes, a la clave se agregan bytes aleatorios y el IV se trunca para llegar al largo requerido.

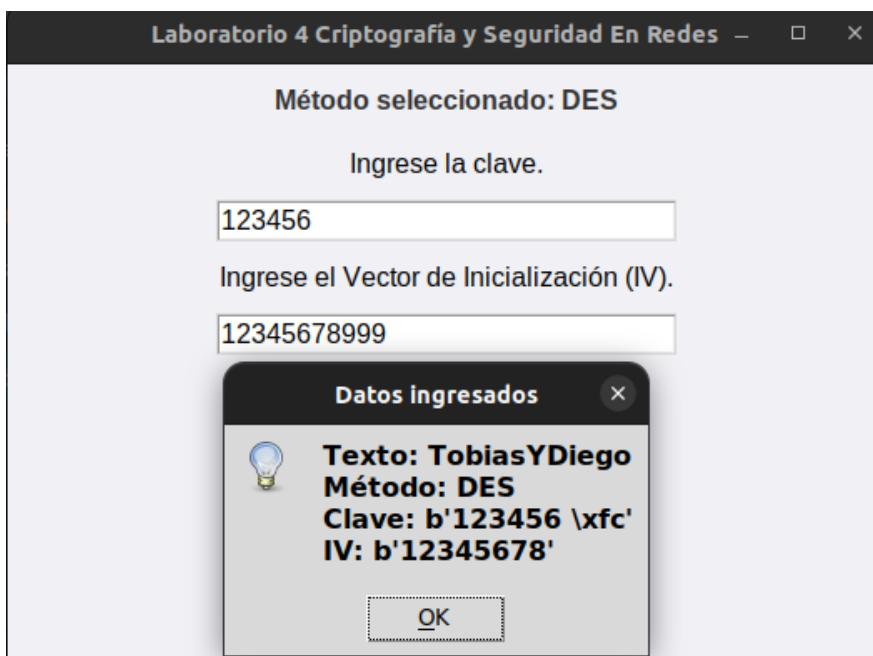


Figura 6: Ajuste de clave y despliegue de parámetros para DES.

- AES-256: En la Figura 7 se observa el ajuste para los mismos parámetros de ejemplo utilizados para DES. El largo de estos varía debido al cambio de método de cifrado (Para AES-256, la clave es de 32 Bytes y el IV de 16 Bytes).

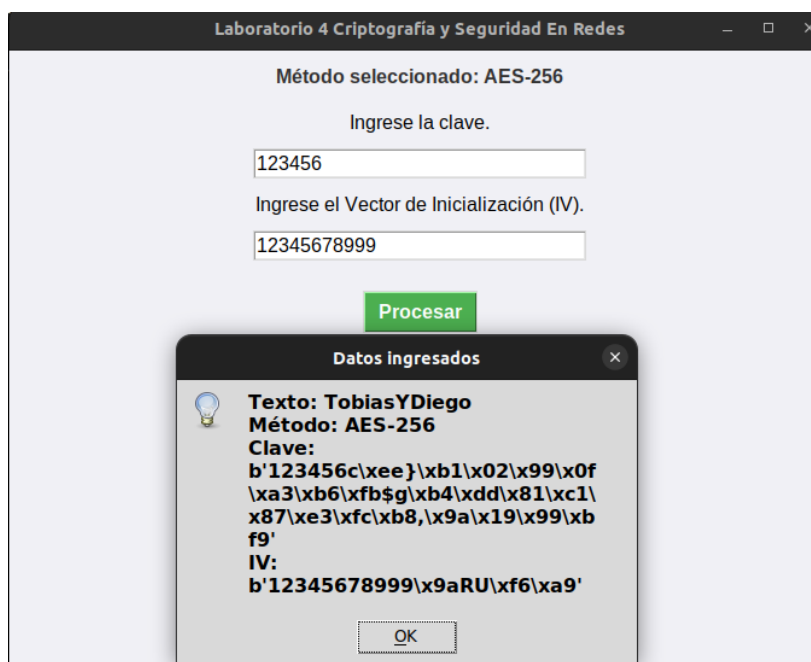


Figura 7: Ajuste de clave y despliegue de parámetros para AES-256.

- 3DES: En la Figura 8 se realiza el mismo ejemplo para 3DES, observando el ajuste de los parámetros (24 Bytes para la clave y 8 para el IV).

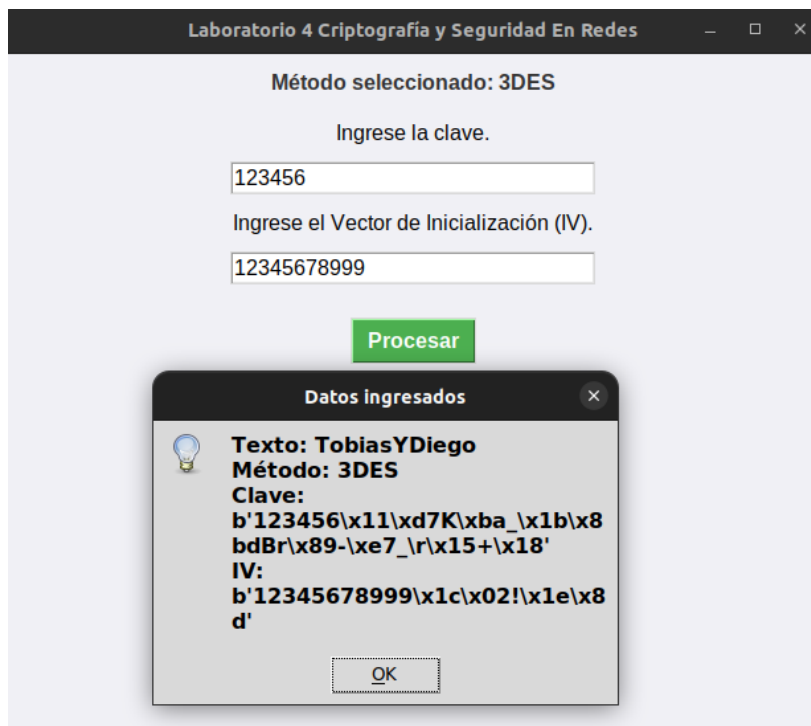


Figura 8: Enter Caption

2.4. Implementa el cifrado y descifrado en modo CBC

En la figura 9 se observa el código asociado al proceso de cifrado. De la librería *PyCryptodome* se extraen las funciones de cifrado *DES*, *AES* y *DES3*. En base a la elección realizada por el usuario, el código va a realizar el cifrado utilizando como parámetro el texto plano ingresado, la clave y el vector de inicialización. El texto cifrado resultante se muestra en pantalla en Base 64 para una representación legible.

```

seleccion = opcion_var.get()
texto = texto_entrada.get().encode() # Convertir el texto a bytes
metodo = "DES" if seleccion == 1 else "AES-256" if seleccion == 2 else "3DES"

if metodo == "DES":
    cipher = DES.new(key, DES.MODE_CBC, iv)
    ciphertext = cipher.encrypt(pad(texto, DES.block_size))
elif metodo == "AES-256":
    cipher = AES.new(key, AES.MODE_CBC, iv)
    ciphertext = cipher.encrypt(pad(texto, AES.block_size))
elif metodo == "3DES":
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
    ciphertext = cipher.encrypt(pad(texto, DES3.block_size))

# Aplicar padding y cifrar el texto

encoded_ciphertext = base64.b64encode(ciphertext).decode()

messagebox.showinfo("Resultado del Cifrado", f"Texto cifrado en {metodo}: {encoded_ciphertext}")

```

Figura 9:

Anteriormente, en el programa se mostraban los parámetros del cifrado utilizado. Ahora se puede observar adicionalmente el cifrado resultante.

A continuación se realizará la prueba para cada algoritmo de cifrado utilizando el texto plano “TobiasYDiego”.

- **DES:** Se utiliza la clave “EstaClav” y el vector de inicialización “26S7eBSu”. El resultado se observa en la Figura 10.

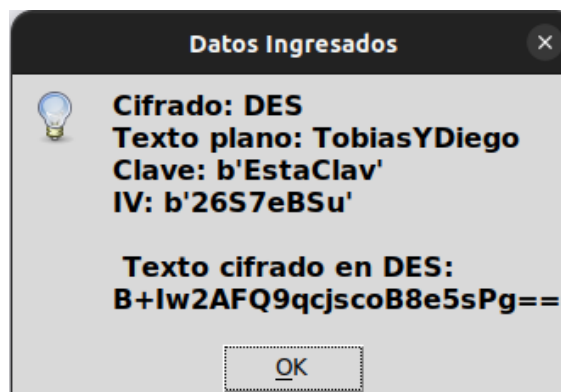


Figura 10: Cifrado DES para el texto “TobiasYDiego”.

- **AES-256:** Se utiliza la clave “EstaEsUnaClaveDe32Bytes123456789” y el vector de inicialización “26S7eBSuDYedBQUM”. El resultado obtenido se observa en la Figura 11.

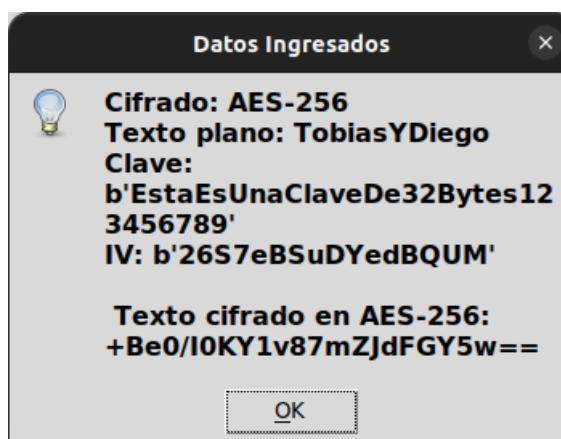


Figura 11: Cifrado AES-256 para el texto “TobiasYDiego”.

- **3DES:** Se utiliza la clave “EstaEsUnaClaveDe24Bytes1” y el vector de inicialización “26S7eBSu”. El resultado se observa en la Figura 12.

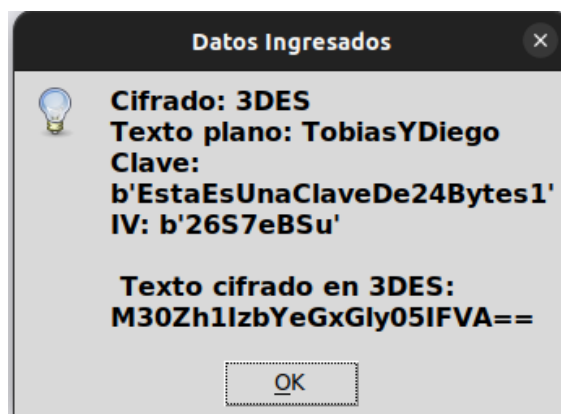


Figura 12: Cifrado 3DES para el texto “TobiasYDiego”.

Así, los cifrados obtenidos para cada algoritmo en formato Base 64 son:

- **DES:** “B+Iw2AFQ9qcjscoB8e5sPg==”
- **AES-256:** “+Be0/10KY1v87mZJdFGY5w==”
- **3DES:** “M30Zh1lzbYeGxGly05IFVA==”

2.5. Compara los resultados con un servicio de cifrado online

Con el fin de realizar una comparación con los cifrados obtenidos mediante el código realizado se utilizan servicios de cifrado online, utilizando los mismos parámetros que los utilizados en los ejemplos realizados en el punto anterior. Para 3DES se utiliza la página *Herramienta de encriptación y desencriptación DES-EDE3-CBC*. Para AES-256 se utiliza la página *Herramienta de encriptación y desencriptación AES-256-CBC*, y para DES se utiliza la página *Online DES Encryption*.

- **DES:** En la figura 13 se pueden observar los distintos parámetros utilizados y el texto cifrado resultante.
 - El texto ingresado: “TobiasYDiego”
 - El vector de inicialización: 26S7eBSu
 - La clave utilizada: “EstaClav”
 - Modo: CBC
 - Formato de salida: Base 64

Así, el texto cifrado resultante es “B+lw2AFQ9qcjscoB8e5sPg==”, obteniendo un cifrado idéntico al observado el código realizado.

Des Encryption / Decryption Tool

DES Encryption

Encryption Text: TobiasYDiego

Encrypted Text: B+lw2AFQ9qcjscoB8e5sPg==

Secret Key: EstaClav

Encryption Mode: ☒ CBC ☐ ECB

IV (optional): 26S7eBSu

Output format: ☒ Base64 ☐ HEX

Encrypt

Figura 13: Cifrado DES con un servicio de cifrado online-

■ AES-256:

En la figura 14 se puede observar en el apartado “Más información” los parámetros utilizados:

- El texto ingresado: “TobiasYDiego”
- El vector de inicialización: “26S7eBSuDYedBQUM”
- La clave utilizada: “EstaEsUnaClaveDe32Bytes123456789”
- El método utilizado: “aes-256-cbc”

En el resultado se puede observar que el texto cifrado es “+Be0/I0KY1v87mZJdFGY5w==”, texto el cual es idéntico al obtenido en el código realizado.

Resultado

+Be0/I0KY1v87mZJdFGY5w==



Más información

```
{
  "data": "TobiasYDiego",
  "method": "aes-256-cbc",
  "vector": "26S7eBSuDYedBQUM",
  "tag": "0IAggQCGUbPgmPN6lFjQ8g==",
  "key": "EstaEsUnaClaveDe32Bytes123456789",
  "dataEncrypt": "+Be0/I0KY1v87mZJdFGY5w==",
  "dataEncryptBase64Encode": "K0JlMC9JMEtZMXy4N21aSmRGR1k1dz09"
}
```

Encriptar

Figura 14: Cifrado AES-256 con un servicio de cifrado online.

■ 3DES:

En la figura 15 se puede observar en el apartado “Más información” los parámetros utilizados:

- El texto ingresado: “TobiasYDiego”
- El método utilizado: “des-ede3-cbc”
- El vector de inicialización: “26S7eBSu”
- La clave utilizada: “EstaEsUnaClaveDe24Bytes1”

En el resultado se puede observar que el texto cifrado resulta en "M30Zh1IzbYeGxGly05IFVA==", texto el cual es idéntico al obtenido en el código realizado.

Resultado

M30Zh1IzbYeGxGly05IFVA==



Más información

```
{
  "data": "TobiasYDiego",
  "method": "des-edc3-cbc",
  "vector": "26S7eBSu",
  "tag": "OIAggQCGUbPgmPN6lFjQ8g==",
  "key": "EstaEsUnaClaveDe24Bytes1",
  "dataEncrypt": "M30Zh1IzbYeGxGly05IFVA==",
  "dataEncryptBase64Encode": "TTMwWmgxSXpiWVHeEdseTA1SUZWQT09"
}
```

Encriptar

Figura 15: Cifrado 3DES con un servicio de cifrado online.

Así, se puede corroborar el correcto funcionamiento del programa creado en Python para poder realizar los distintos cifrados en base a los resultados observados en los servicios de encriptación online. La obtención de resultados iguales se debe a la utilización de los mismos parámetros, además,

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

En general, la aplicabilidad del cifrado simétrico en la vida real recae en la protección del acceso a contenido sensible presente en los datos transmitidos en la red.

Este tipo de cifrado puede tener un rol muy importante en el almacenamiento seguro de datos, ya que muchas bases de datos (o almacenamientos en la nube) cifran la información (como contraseñas o datos personales) utilizando cifrado simétrico. Esto permite proteger la información almacenada, ya que se garantiza que si alguien accede físicamente a la base de datos, los datos sensibles tienen una capa extra de seguridad, logrando que el intruso no pueda obtener la información buscada.

En cuanto a mensajería segura, *WhatsApp* utiliza un cifrado simétrico para proteger los mensajes emitidos y entrantes, donde únicamente el emisor y el receptor poseen una clave simétrica para cifrar/descifrar el contenido que existe en el mensaje, lo que permite que sólo ellos tengan acceso al contenido de los mensajes.

La seguridad de los datos tiene radical importancia en entidades financieras y bancarias, tomando un importante rol el cifrado simétrico en las transacciones monetarias por la red. La información sensible asociada como los números de las tarjetas, los pines o claves de estas, urgen de una capa extra de seguridad para que un tercero no obtenga la información y haga mal uso de esta.

Estos son algunos de los usos en la vida real del cifrado simétrico, pero en general, este siempre busca proteger el acceso de los datos sensibles mediante el uso de una llave pública que permite cifrar/descifrar el contenido presente en los datos. La selección del método de cifrado recae meramente en el usuario, pero AES-256 es uno de los algoritmos de cifrado más seguros y es el estándar recomendado por organizaciones como el NIST (Instituto Nacional de Estándares y Tecnología de EE.UU.) [4], además este algoritmo no tiene vulnerabilidades conocidas que lo comprometan a diferencia de DES (algoritmo obsoleto) y 3DES (también obsoleto, vulnerable al ataque de cumpleaños)[5].

La diferencia entre la utilización de Hashes y Cifrados simétricos recae en la diferencia en su funcionalidad. El cifrado simétrico permite cifrar y descifrar la información mediante el uso de la clave, siendo importante en la protección de información en la comunicación entre dos entidades. Los algoritmos de Hash son unidireccionales, permitiendo únicamente esconder el contenido de los mensajes, no ocultando su contenido si no creando una representación de los datos, por lo que no existe una función inversa para esto.

En base a lo anterior, si el objetivo es proteger la confidencialidad y permitir que los datos sean accesibles a futuro, la utilización de un algoritmo de Hash no permitiría recuperar los datos originales, por lo que sería adecuada la utilización del cifrado simétrico. Dicho lo anterior, es importante conocer la diferencia entre ambos para así saber de qué manera encriptar los datos en base al contexto.

Conclusiones y comentarios

La utilización del cifrado simétrico toma un rol fundamental en la protección de datos sensibles. La elección del algoritmo de cifrado debería estar respaldada por organizaciones oficiales como la NIST para asegurar la integridad de los datos y evitar posibles vulnerabilidades o ataques.

Los parámetros ingresados para el cifrado escogido también toman una vital importancia. Para la clave, se necesita de cierta Aleatoriedad y unicidad, pues en la reutilización de la misma clave existe el riesgo de que un atacante pueda percibir patrones y descifrar el contenido. También, si la clave no es lo suficientemente aleatoria, existe cierta vulnerabilidad a ataques de fuerza bruta o ataques de diccionario.

La actividad realizada permite una reflexión sobre el uso del cifrado simétrico, entendiendo los parámetros que conlleva, tal como la utilización de una única clave: En redes inseguras se realiza un mayor enfoque en el uso de cifrado asimétrico (Teniendo dos claves distintas), el cual se puede utilizar para intercambiar la clave de sesión simétrica de forma segura, realizando así un cifrado híbrido.

En general, con el uso de buenas prácticas para proteger la clave y garantizar la autenti-

cación de los mensajes, el cifrado simétrico es altamente recomendable y seguro.

Referencias

- [1] PyCryptodome, *DES - Cryptographic Services* Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des.html>. Consultado en noviembre de 2024.
- [2] PyCryptodome, *AES - Cryptographic Services*, Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html>. Consultado en noviembre de 2024.
- [3] PyCryptodome, *Triple DES - Cryptographic Services*, Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des3.html>. Consultado en noviembre de 2024.
- [4] NIST, Dworkin, M., Sonmez Turan, M., and Mouha, N., *Advanced Encryption Standard (AES)*, Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2023. Disponible en: <https://doi.org/10.6028/NIST.FIPS.197-upd1>, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936594. Consultado en noviembre de 2024.
- [5] Sofía Belmar and Tobias Guerrero, *Ayudantía 4: Cifrado Simétrico*, Universidad Diego Portales, 2024.