

Informe Laboratorio 4

Sección 1

Matías Vásquez

e-mail: matias.vasquez1@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. AES-256:	3
2.2. Solicita datos de entrada desde la terminal	3
2.3. Valida y ajusta la clave según el algoritmo	4
2.4. Implementa el cifrado y descifrado en modo CBC	5
2.5. Compara los resultados con un servicio de cifrado online	5
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real	7
2.7. conclusión	7
2.8. comentarios	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

2.1.1. DES

: la clave requerida para que el algoritmo DES funcione es de 64 bits o 8 byte, dado que la clave en total son 56 bits mas 8 bits de paridad, o 7 byte de clave y 1 de paridad. El vector de inicialización corresponde a una función encargada de evitar que el texto se repita, y a excepción de usar un modo de cifrado que no use un IV, este debe tener la misma cantidad de byte que el bloque del algoritmo requerido.

3DES: 3DES corresponde a el uso consecutivo de DES tres veces, cada vez usando una clave diferente, lo que requiere el uso de al menos 2 claves diferentes, hasta un máximo de 3, esto se determina por el tamaño del bloque de clave solicitado, 16 bytes se asume que son 2 claves y usan el orden de K1, K2, K1, en caso de pedir 24 bytes se están utilizando 3 claves y usan el orden de K1, K2, K3. y nuevamente cualquier modo que requiera un IV tiene que tener un tamaño igual al bloque del algoritmo usado, en este caso 16-24 bits.

2.1.2. AES-256:

AES fue pensado con 3 niveles de seguridad AES-128, AES-192 y finalmente AES-256, a cada nivel se le nombro con la cantidad de bite requeridos para la contraseña, en este caso EAS-256 utiliza 256 bites o 32 bytes, y como se investigo anteriormente cualquier modo que pida un IV pide necesariamente uno del mismo tamaño del bloque del algoritmo requerido, en este caso 16 bytes.

2.2. Solicita datos de entrada desde la terminal

El código presentado si tiene la capacidad de solicitar la información al usuario, mostrado en el siguiente fragmento de código:

```
1 print("Seleccione una operacin:")
2 print("1. Cifrar texto")
3 print("2. Descifrar texto")
4 op = input("Opcin (1 o 2): ")
5
6 print("\nSeleccione el algoritmo:")
7 print("1. DES")
8 print("2. 3DES")
9 print("3. AES-256")
10 algoritmo_op = input("Opcin (1, 2 o 3): ")
11
12 algoritmo = {"1": "DES", "2": "3DES", "3": "AES"}[algoritmo_op]
13 tamao_clave_3des = None
14
```

```

15 if algoritmo == "3DES":
16     tipo = input("Desea usar 2 claves (16 bytes) o 3 claves (24 bytes)? (Ingrese 16 o 24): ")
17     tamao_clave_3des = 16 if tipo == "16" else 24
18
19 clave = input("\nIngrese la clave: ")
20 iv = input("Ingrese el IV: ")
21
22 if op == "1":
23     mensaje = input("Ingrese el texto a cifrar: ")
24     texto_cifrado, key_final, iv_final = cifrar(mensaje, clave, iv, algoritmo, tamao_clave_3des)
25     print(f"\n[{algoritmo}] Cifrado")
26     print(f"Clave final usada (hex): {key_final.hex()}")
27     print(f"IV usado (hex): {iv_final.hex()}")
28     print(f"Texto cifrado (Base64): {texto_cifrado}")

```

Y se puede comparar con las siguientes imágenes como solicita información al usuario.

```

73 print("\nSeleccione el algoritmo:")
74 print("1. DES")
75 print("2. 3DES")
76 print("3. AES-256")
77 algoritmo_op = input("Opción (1, 2 o 3): ")
78
79 algoritmo = {"1": "DES", "2": "3DES", "3": "AES"}[algoritmo_op]
80 tamaño_clave_3des = None
81
82 if algoritmo == "3DES":
83     tipo = input("Desea usar 2 claves (16 bytes) o 3 claves (24 bytes)? (Ingrese 16 o 24): ")
84     tamaño_clave_3des = 16 if tipo == "16" else 24
85
86 clave = input("\nIngrese la clave: ")
87 iv = input("Ingrese el IV: ")
88
89 if op == "1":
90     mensaje = input("Ingrese el texto a cifrar: ")
91     texto_cifrado, key_final, iv_final = cifrar(mensaje, clave, iv, algoritmo, tamaño_clave_3des)
92     print(f"\n[{algoritmo}] Cifrado")
93     print(f"Clave final usada (hex): {key_final.hex()}")
94     print(f"IV usado (hex): {iv_final.hex()}")

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

[AES] Cifrado
Clave final usada (hex): 6d695f636c6176655f736563726574615fd695f636c6176655f736563726574
IV usado (hex): 696e696369616c766563746f72313233
Texto cifrado (Base64): +C0BhCVMS:SSM15Ca10fA==
(venv) PS C:\Users\Walter\Desktop\laboratorio-cripto\laboratorio_4> python cifrado.py
Seleccione una operación:
1. Cifrar texto
2. Descifrar texto
Opción (1 o 2): 1

Seleccione el algoritmo:
1. DES
2. 3DES
3. AES-256
Opción (1, 2 o 3): 3

Ingrese la clave: hola
Ingrese el IV: hola
Ingrese el texto a cifrar:

```

Figura 1: ingreso de datos desde terminal

2.3. Valida y ajusta la clave según el algoritmo

el código es capaz de adaptar la clave según la cantidad de bytes que necesita para completar el bloque, tal como lo muestra el siguiente código.

```

1 def cifrar(texto, clave, iv, algoritmo, tamao_clave_3des=None):
2     if algoritmo == "DES":
3         key = ajustar_clave(clave, 8)
4         iv = ajustar_iv(iv, 8)
5         cipher = DES.new(key, DES.MODE_CBC, iv)
6     elif algoritmo == "3DES":
7         key_size = 16 if tamao_clave_3des == 16 else 24
8         key = ajustar_clave(clave, key_size)
9         key = DES3.adjust_key_parity(key)
10        iv = ajustar_iv(iv, 8)
11        cipher = DES3.new(key, DES3.MODE_CBC, iv)
12    elif algoritmo == "AES":
13        key = ajustar_clave(clave, 32)
14        iv = ajustar_iv(iv, 16)
15        cipher = AES.new(key, AES.MODE_CBC, iv)
16    else:
17        raise ValueError("Algoritmo no vlido.")
18
19    texto_padded = pad(texto.encode(), cipher.block_size)
20    cifrado = cipher.encrypt(texto_padded)
21    return base64.b64encode(cifrado).decode(), key, iv

```

2.4. Implementa el cifrado y descifrado en modo CBC

Eso se muestra en la sección anterior, mas específicamente en las siguientes líneas:

```
1 cipher = DES.new(key, DES.MODE_CBC, iv)
2
3 cipher = DES3.new(key, DES3.MODE_CBC, iv)
4
5 cipher = AES.new(key, AES.MODE_CBC, iv)
```

Esa es la implementación explícita de CBC en el código entregada por chatGPT

2.5. Compara los resultados con un servicio de cifrado online

Comparando los resultados en diferentes sitios online, tales como "Chef" para AES-256 y Cifrado y descifrado DES en línea para des, pude observar que me era imposible obtener el mismo resultado, sin importar como lo configurara, después de un tiempo e análisis y discusiones con la IA generativa ChatGPT para confirmar dudas, se teoriza que la causa de error era rellenar los byte faltantes de manera aleatoria con la función de padding `get.random_bytes()`, función que ofusca el cifrado dado que cada iteración del código generara un resultado diferente.

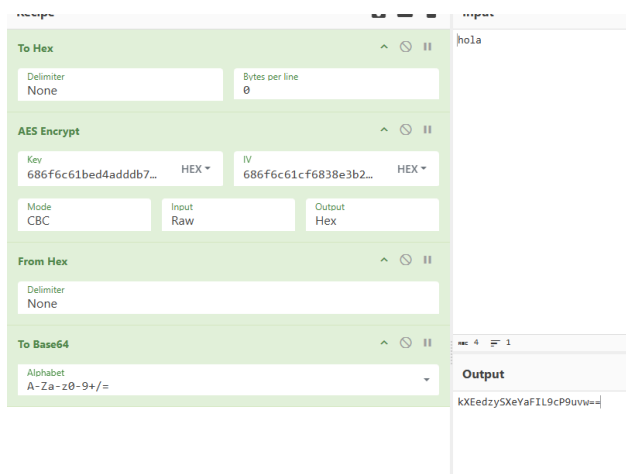


Figura 2: uso de la página del chef para intentar obtener el mismo resultado

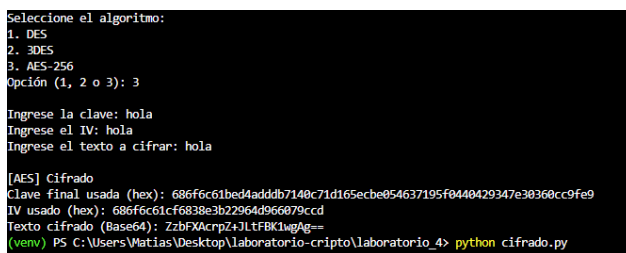


Figura 3: resultados obtenidos al usar el cifrado del código creado

Paralelamente se pudo descifrar el mensaje obtenido usando un descifrador online.

```
Opción (1, 2 o 3): 3

Ingrese la clave: mi_clave_secreta_mi_clave_secret
Ingrese el IV: inicialvector123
Ingrese el texto a cifrar: mensaje secreto

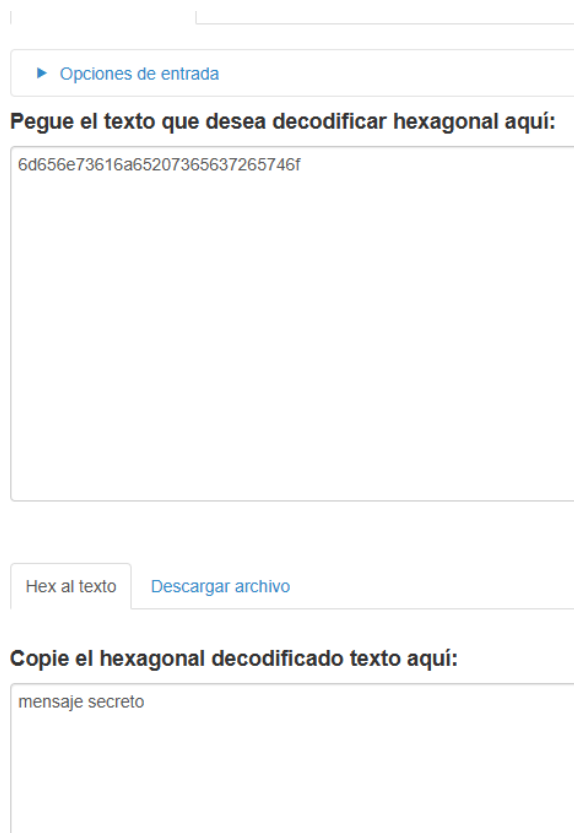
[AES] Cifrado
Clave final usada (hex): 6d695f636c6176655f736563726574615f6d695f636c6176655f736563726574
IV usado (hex): 696e696369616c766563746f72313233
Texto cifrado (Base64): +C0DhCYR5zGSW1sCGaI0fA==

(venv) PS C:\Users\Matias\Desktop\laboratorio-cripto\laboratorio_4> python cifrado.py
```

Figura 4: texto cifrado obtenido usando el codigo generado

The image shows a web-based AES decryption tool. The 'Input' field contains the Base64 encoded ciphertext '+C0DhCYR5zGSW1sCGaI0fA=='. The 'AES Decrypt' section shows the key '6d695f636c6176655f...' and IV '696e696369616c7665...' in hexadecimal. The 'Output' field displays the decrypted message '6d65e73616a65207365637265746f'.

Figura 5: uso de la pagina del chef para descifrar el mensaje obtenido



A screenshot of a web application interface for decoding hexagonal text. At the top, there is a button labeled 'Opciones de entrada'. Below it, a text prompt reads 'Pegue el texto que desea decodificar hexagonal aquí:'. A text input field contains the hex string '6d656e73616a65207365637265746f'. Below the input field, there are two buttons: 'Hex al texto' and 'Descargar archivo'. Below these buttons, another text prompt reads 'Copie el hexagonal decodificado texto aquí:'. A text output field displays the decoded message 'mensaje secreto'.

Figura 6: pasar el resultado del descifrado a base64

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

El cifrado simétrico tiene muchas aplicaciones y muy diversas, dado que se usa una sola clave para cifrar como descifrar lo hace ideal para situaciones de emisor y receptor, puede usarse para redes VPN para cifrar el canal de mensaje sin que terceros puedan leer los mensajes, mensajería instantánea como Whatsapp o Messenger, otras opciones son el cifrado de bases de datos o de SSD.

Como se pudo comprobar y demostrar, existen muchas opciones para el cifrado simétrico, sobretodo se contabilizan sus ventajas, entre ellas: fácil implementación en dispositivos de bajos recursos e ideal para manejar grandes volúmenes de datos

Conclusiones y comentarios

2.7. conclusión

Como se puede demostrar durante el transcurso del laboratorio se puede afirmar la correcta realización de la experiencia o al menos la mayoría de esta, como se puede demostrar durante la explicación de la cantidad de bytes en los claves e IV en los diferentes métodos de

cifrados. Por otro lado no se pudo resolver adecuadamente la razón de por que no se obtiene el mismo texto cifrado y solo se obtuvo una suposición.

Con todo lo comentado se puede dar por concluido este informe del laboratorio 4 de criptografía y seguridad en redes.

2.8. comentarios

Para iniciar los comentarios, se uso chatGPT para generar y editar el código utilizado.

Claro, aquí tienes una versión **mejorada** de tu código, que:

1. ☒ Pregunta primero si el usuario quiere **cifrar** o **descifrar**.
2. ☒ Luego pregunta qué algoritmo usar: **DES**, **3DES**, o **AES-256**.
3. ☒ Si elige **3DES**, pregunta si usará una clave de 16 o de 24 bytes.

☒ CÓDIGO ACTUALIZADO:

```
python
from Crypto.Cipher import DES, DES3, AES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad
import base64

def ajustar_clave(clave, tamaño_requerido):
    clave_bytes = clave.encode()
    if len(clave_bytes) < tamaño_requerido:
        clave_bytes += get_random_bytes(tamaño_requerido - len(clave_bytes))
    elif len(clave_bytes) > tamaño_requerido:
        clave_bytes = clave_bytes[:tamaño_requerido]
    return clave_bytes

def ajustar_iv(iv, tamaño_requerido):
```

Figura 7: chatGPT generando el menú interactivo

Las paginas de donde se saco la investigación sobre los bytes por metodo de cifrado:
Algoritmo DES y 3DES algoritmo AES-256
Github.