

Informe Laboratorio 4

Sección 4

José Berríos Piña
e-mail: jose.berrios1@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	4
2.4. Implementa el cifrado y descifrado en modo CBC	7
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

1. DES

- **Tamaño de Clave e IV:** La clave de DES es de 64 bits (8 bytes), pero solo se utilizan 56 bits (7 bytes), ya que 8 bits se usan para verificar la integridad (paridad). El vector de inicialización (IV) también es de 64 bits (8 bytes) y se usa en ciertos modos, como CBC y OFB, para asegurar que mensajes idénticos no se cifren igual, incluso con la misma clave, lo que aumenta la seguridad [1].
- **Características:** En comparación con otros algoritmos, DES es menos seguro debido a su clave más corta.

2. AES-256

- **Tamaño de Clave e IV:** AES-256 usa una clave de 256 bits (32 bytes) y un IV de 128 bits (16 bytes) en el modo CBC. Su IV más largo mejora la seguridad al proporcionar mayor aleatoriedad [2].
- **Características:** AES-256 es más seguro y funciona más rápido en hardware adecuado, en comparación con DES y 3DES.

3. 3DES

- **Tamaño de Clave e IV:** 3DES tiene una clave de 24 bytes (192 bits), aunque solo 21 bytes (168 bits) son efectivos debido a los bits de paridad. Además, debido a vulnerabilidades en DES, la seguridad efectiva de 3DES es de 14 bytes (112 bits). Su IV es de 64 bits (8 bytes) y ayuda a mejorar la seguridad en los modos que lo requieren, aunque sigue siendo menos eficiente que AES [3].
- **Características:** 3DES es una extensión de DES, pero es más lento y menos seguro que AES-256.

2.2. Solicita datos de entrada desde la terminal

Se creará un programa que contendrá los algoritmos de cifrado DES, AES-256 y 3DES. Cada cifrado estará contenido en sus respectivas funciones, y se utilizará la biblioteca *Cryptodome*, instalada mediante el comando *pip install pycryptodome*. Además, cada programa

recibirá la llave, el vector de inicialización y el mensaje a cifrar desde la terminal. Para facilitar el análisis, se utilizarán los mismos datos de entrada para todos los algoritmos, y el programa se encargará de rellenar o truncar los atributos de acuerdo con las necesidades de cada caso, tal como se solicita en el desarrollo del laboratorio.

Datos a ingresar para cada uno de los casos:

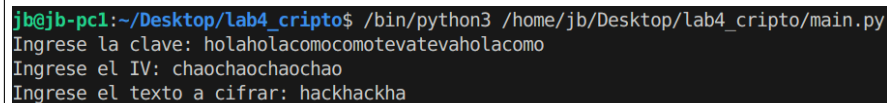
- **Llave:** holaholacomocomotevatevaholacomo.
- **Vector de inicialización (IV):** chaochaochaochao.
- **Mensaje:** hackhackha.

```

1 # Entrada de datos
2 # key_input = holaholacomocomotevatevaholacomo
3 key_input = input("Ingrese la clave: ").encode()
4 # iv_input = chaochaochaochao
5 iv_input = input("Ingrese el IV: ").encode()
6 # texto_input = hackhackha
7 texto_input = input("Ingrese el texto a cifrar: ")

```

Bloque de código 1: Extracto de código encargado de solicitar los datos mediante la terminal



```

jb@jb-pc1:~/Desktop/lab4_cripto$ /bin/python3 /home/jb/Desktop/lab4_cripto/main.py
Ingrese la clave: holaholacomocomotevatevaholacomo
Ingrese el IV: chaochaochaochao
Ingrese el texto a cifrar: hackhackha

```

Figura 1: Vista al funcionamiento de la terminal con el Bloque de código 1.

2.3. Valida y ajusta la clave según el algoritmo

1. DES

- Esta función recibe la clave proporcionada y ajusta su tamaño a 8 bytes. Si es menor, se completa con bytes aleatorios; si es mayor, se trunca. Lo mismo para IV, de 8 bytes.

```

1 # Ajustar claves y IVs seg n el algoritmo
2 key_DES = ajustar_clave(key_input, 8, "Clave_DES")
3 iv_DES = ajustar_clave(iv_input, 8, "IV_DES")
4
5 def ajustar_clave(clave, longitud_requerida, tipo):
6     #Ajusta la clave al tama o requerido y notifica si fue
7     #truncada o extendida.
8     if len(clave) < longitud_requerida:

```

```

8         clave += get_random_bytes(longitud_requerida - len(clave
9         ))
10        print(f"{tipo} ajustado a {longitud_requerida} bytes a
11        aadiendo bytes adicionales.")
12    elif len(clave) > longitud_requerida:
13        clave = clave[:longitud_requerida]
14        print(f"{tipo} exced a {longitud_requerida} bytes y fue
15        truncado.")
16    print(f"{tipo} final utilizado (hex):", binascii.hexlify(
17        clave).decode())
18    return clave

```

Bloque de código 2: Extracto de código encargado de validar y ajustar los datos de DES.

```

Clave DES excedía 8 bytes y fue truncado.
Clave DES final utilizado (hex): 686f6c61686f6c61
IV DES excedía 8 bytes y fue truncado.
IV DES final utilizado (hex): 6368616f6368616f

```

Figura 2: Validación y ajuste de los datos del algoritmo DES.

2. AES-256

- Esta función recibe la clave proporcionada y ajusta su tamaño a 32 bytes. Si es menor, se completa con bytes aleatorios; si es mayor, se trunca. Lo mismo para IV, de 16 bytes.

```

1 # Ajustar claves y IVs seg n el algoritmo
2 key_AES = ajustar_clave(key_input, 32, "Clave AES-256")
3 iv_AES = ajustar_clave(iv_input, 16, "IV AES-256")
4
5 def ajustar_clave(clave, longitud_requerida, tipo):
6     #Ajusta la clave al tama o requerido y notifica si fue
7     truncada o extendida.
8     if len(clave) < longitud_requerida:
9         clave += get_random_bytes(longitud_requerida - len(clave
10         ))
11        print(f"{tipo} ajustado a {longitud_requerida} bytes a
12        aadiendo bytes adicionales.")
13    elif len(clave) > longitud_requerida:
14        clave = clave[:longitud_requerida]
15        print(f"{tipo} exced a {longitud_requerida} bytes y fue
16        truncado.")

```

```

13     print(f"{tipo}_final_utilizado_(hex):", binascii.hexlify(
        clave).decode())
14     return clave

```

Bloque de código 3: Extracto de código encargado de validar y ajustar los datos de AES-256.

```

Clave AES-256 final utilizado (hex): 686f6c61686f6c61636f6d6f636f6d6f7465766174657661686f6c61636f6d6f
IV AES-256 final utilizado (hex): 6368616f6368616f6368616f6368616f

```

Figura 3: Validación y ajuste de los datos del algoritmo AES-256.

3. 3DES

- Esta función recibe la clave proporcionada y ajusta su tamaño a 24 bytes. Si es menor, se completa con bytes aleatorios; si es mayor, se trunca. Lo mismo para IV, de 8 bytes.

```

1  # Ajustar claves y IVs seg n el algoritmo
2  key_3DES = ajustar_clave(key_input, 24, "Clave_3DES")
3  iv_3DES = ajustar_clave(iv_input, 8, "IV_3DES")
4
5  def ajustar_clave(clave, longitud_requerida, tipo):
6      #Ajusta la clave al tama o requerido y notifica si fue
        truncada o extendida.
7      if len(clave) < longitud_requerida:
8          clave += get_random_bytes(longitud_requerida - len(clave
9              ))
10         print(f"{tipo}_ajustado_a_{longitud_requerida}_bytes_
        a_adiendo_bytes_adicionales.")
11     elif len(clave) > longitud_requerida:
12         clave = clave[:longitud_requerida]
13         print(f"{tipo}_exced a_{longitud_requerida}_bytes_y_fue
        _truncado.")
14     print(f"{tipo}_final_utilizado_(hex):", binascii.hexlify(
        clave).decode())
15     return clave

```

Bloque de código 4: Extracto de código encargado de validar y ajustar los datos de 3DES.

```

Clave 3DES excedía 24 bytes y fue truncado.
Clave 3DES final utilizado (hex): 686f6c61686f6c61636f6d6f636f6d6f7465766174657661
IV 3DES excedía 8 bytes y fue truncado.
IV 3DES final utilizado (hex): 6368616f6368616f

```

Figura 4: Validación y ajuste de los datos del algoritmo 3DES.

2.4. Implementa el cifrado y descifrado en modo CBC

Una vez ingresados la clave, el IV y el texto a cifrar, el programa ajusta la clave y IV según sea necesario, cifra el mensaje y, posteriormente, lo descifra para mostrar el texto original.

1. DES

```

1 # Cifrado y descifrado con cada algoritmo
2 print("\n---_Cifrado_y_Descifrado_con_DES_---")
3 texto_cifrado_DES = cifrar_DES(key_DES, iv_DES, texto_input)
4 descifrar_DES(key_DES, iv_DES, texto_cifrado_DES)
5
6 def cifrar_DES(clave, iv, texto):
7     #Cifra el texto utilizando DES en modo CBC.
8     cipher = DES.new(clave, DES.MODE_CBC, iv)
9     padding_length = 8 - (len(texto) % 8)
10    texto_padded = texto + (chr(padding_length) * padding_length
11    )
12    texto_cifrado = cipher.encrypt(texto_padded.encode())
13    print("Texto_cifrado_con_DES(hex):", binascii.hexlify(
14        texto_cifrado).decode())
15    return texto_cifrado
16
17 def descifrar_DES(clave, iv, texto_cifrado):
18     #Descifra el texto cifrado utilizando DES en modo CBC.
19     cipher = DES.new(clave, DES.MODE_CBC, iv)
20     texto_descifrado = cipher.decrypt(texto_cifrado)
21     padding_length = texto_descifrado[-1]
22     texto_descifrado = texto_descifrado[:-padding_length].decode
23     ()
24     print("Texto_descifrado_con_DES:", texto_descifrado)
25     return texto_descifrado

```

Bloque de código 5: Extracto de código encargado de cifrar y descifrar con el algoritmo DES.

```

--- Cifrado y Descifrado con DES ---
Texto cifrado con DES (hex): 5e50eb2190375c54c13860eb040872bc
Texto descifrado con DES: hackhackha

```

Figura 5: Cifrado y descifrado del algoritmo DES.

2. AES-256

```

1 # Cifrado y descifrado con cada algoritmo
2 print("\n---_Cifrado_y_Descifrado_con_AES-256_---")
3 texto_cifrado_AES = cifrar_AES(key_AES, iv_AES, texto_input)
4 descifrar_AES(key_AES, iv_AES, texto_cifrado_AES)
5
6 def cifrar_AES(clave, iv, texto):
7     #Cifra el texto utilizando AES-256 en modo CBC.
8     cipher = AES.new(clave, AES.MODE_CBC, iv)
9     padding_length = 16 - (len(texto) % 16)
10    texto_padded = texto + (chr(padding_length) * padding_length
11    )
12    texto_cifrado = cipher.encrypt(texto_padded.encode())
13    print("Texto_cifrado_con_AES-256(hex):", binascii.hexlify(
14        texto_cifrado).decode())
15    return texto_cifrado
16
17 def descifrar_AES(clave, iv, texto_cifrado):
18     #Descifra el texto cifrado utilizando AES-256 en modo CBC.
19     cipher = AES.new(clave, AES.MODE_CBC, iv)
20     texto_descifrado = cipher.decrypt(texto_cifrado)
21     padding_length = texto_descifrado[-1]
22     texto_descifrado = texto_descifrado[:-padding_length].decode()
23
24     print("Texto_descifrado_con_AES-256:", texto_descifrado)
25     return texto_descifrado

```

Bloque de código 6: Extracto de código encargado de cifrar y descifrar con el algoritmo AES-256.

```

--- Cifrado y Descifrado con AES-256 ---
Texto cifrado con AES-256 (hex): 8895e224fda98a27a256dc3ebf8be17e
Texto descifrado con AES-256: hackhackha

```

Figura 6: Cifrado y descifrado del algoritmo AES-256.

3. 3DES

```

1 # Cifrado y descifrado con cada algoritmo
2 print("\n---_Cifrado_y_Descifrado_con_3DES_---")
3 texto_cifrado_3DES = cifrar_3DES(key_3DES, iv_3DES, texto_input)
4 descifrar_3DES(key_3DES, iv_3DES, texto_cifrado_3DES)
5
6 def cifrar_3DES(clave, iv, texto):

```



```
7 #Cifra el texto utilizando 3DES en modo CBC.
8 cipher = DES3.new(clave, DES3.MODE_CBC, iv)
9 padding_length = 8 - (len(texto) % 8)
10 texto_padded = texto + (chr(padding_length) * padding_length
11 )
12 texto_cifrado = cipher.encrypt(texto_padded.encode())
13 print("Texto_cifrado_con_3DES(hex):", binascii.hexlify(
14     texto_cifrado).decode())
15 return texto_cifrado
16
17 def descifrar_3DES(clave, iv, texto_cifrado):
18     #Descifra el texto cifrado utilizando 3DES en modo CBC.
19     cipher = DES3.new(clave, DES3.MODE_CBC, iv)
20     texto_descifrado = cipher.decrypt(texto_cifrado)
21     padding_length = texto_descifrado[-1]
22     texto_descifrado = texto_descifrado[:-padding_length].decode(
23         ())
24     print("Texto_descifrado_con_3DES:", texto_descifrado)
25     return texto_descifrado
```

Bloque de código 7: Extracto de código encargado de cifrar y descifrar con el algoritmo 3DES.

```
--- Cifrado y Descifrado con 3DES ---
Texto cifrado con 3DES (hex): 52e8fc53177164df7eec8c0bcc591377
Texto descifrado con 3DES: hackhackha
```

Figura 7: Cifrado y descifrado del algoritmo 3DES.

```
1 from Crypto.Cipher import DES, DES3, AES
2 from Crypto.Random import get_random_bytes
3 import binascii
4
5 def ajustar_clave(clave, longitud_requerida, tipo):
6     if len(clave) < longitud_requerida:
7         clave += get_random_bytes(longitud_requerida - len(clave))
8         print(f"{tipo}_ajustado_a_{longitud_requerida}_bytes_
9               a_adiendo_bytes_adicionales.")
10    elif len(clave) > longitud_requerida:
11        clave = clave[:longitud_requerida]
12        print(f"{tipo}_exced_a_{longitud_requerida}_bytes_y_fue_
13              truncado.")
14    print(f"{tipo}_final_utilizado_(hex):", binascii.hexlify(clave).
15          decode())
16    return clave
17
18 def cifrar_DES(clave, iv, texto):
19     #Cifra el texto utilizando DES en modo CBC.
20     cipher = DES.new(clave, DES.MODE_CBC, iv)
21     padding_length = 8 - (len(texto) % 8)
22     texto_padded = texto + (chr(padding_length) * padding_length)
23     texto_cifrado = cipher.encrypt(texto_padded.encode())
24     print("Texto_cifrado_con_DES_(hex):", binascii.hexlify(
25           texto_cifrado).decode())
26     return texto_cifrado
27
28 def descifrar_DES(clave, iv, texto_cifrado):
29     #Descifra el texto cifrado utilizando DES en modo CBC.
30     cipher = DES.new(clave, DES.MODE_CBC, iv)
31     texto_descifrado = cipher.decrypt(texto_cifrado)
32     padding_length = texto_descifrado[-1]
33     texto_descifrado = texto_descifrado[:-padding_length].decode()
34     print("Texto_descifrado_con_DES:", texto_descifrado)
35     return texto_descifrado
36
37 def cifrar_AES(clave, iv, texto):
38     #Cifra el texto utilizando AES-256 en modo CBC.
39     cipher = AES.new(clave, AES.MODE_CBC, iv)
40     padding_length = 16 - (len(texto) % 16)
41     texto_padded = texto + (chr(padding_length) * padding_length)
42     texto_cifrado = cipher.encrypt(texto_padded.encode())
43     print("Texto_cifrado_con_AES-256_(hex):", binascii.hexlify(
44           texto_cifrado).decode())
45     return texto_cifrado
```

```

41
42 def descifrar_AES(clave, iv, texto_cifrado):
43     #Descifra el texto cifrado utilizando AES-256 en modo CBC.
44     cipher = AES.new(clave, AES.MODE_CBC, iv)
45     texto_descifrado = cipher.decrypt(texto_cifrado)
46     padding_length = texto_descifrado[-1]
47     texto_descifrado = texto_descifrado[:-padding_length].decode()
48     print("Texto_descifrado_con_AES-256:", texto_descifrado)
49     return texto_descifrado
50
51 def cifrar_3DES(clave, iv, texto):
52     #Cifra el texto utilizando 3DES en modo CBC.
53     cipher = DES3.new(clave, DES3.MODE_CBC, iv)
54     padding_length = 8 - (len(texto) % 8)
55     texto_padded = texto + (chr(padding_length) * padding_length)
56     texto_cifrado = cipher.encrypt(texto_padded.encode())
57     print("Texto_cifrado_con_3DES(hex):", binascii.hexlify(
58         texto_cifrado).decode())
59     return texto_cifrado
60
61 def descifrar_3DES(clave, iv, texto_cifrado):
62     #Descifra el texto cifrado utilizando 3DES en modo CBC.
63     cipher = DES3.new(clave, DES3.MODE_CBC, iv)
64     texto_descifrado = cipher.decrypt(texto_cifrado)
65     padding_length = texto_descifrado[-1]
66     texto_descifrado = texto_descifrado[:-padding_length].decode()
67     print("Texto_descifrado_con_3DES:", texto_descifrado)
68     return texto_descifrado
69
70 # Entrada de datos
71 # key_input = holaholacomocomotevatevaholacomo
72 key_input = input("Ingrese_la_clave:_").encode()
73 # iv_input = chaochaochaochao
74 iv_input = input("Ingrese_el_IV:_").encode()
75 # texto_input = hackhackha
76 texto_input = input("Ingrese_el_texto_a_cifrar:_")
77
78 print("\n-----\n")
79
80
81
82 # Ajustar claves y IVs seg n el algoritmo
83 key_DES = ajustar_clave(key_input, 8, "Clave_DES")
84 iv_DES = ajustar_clave(iv_input, 8, "IV_DES")
85 # Cifrado y descifrado con cada algoritmo

```

```

86 print("\n---Cifrado y Descifrado con DES---")
87 texto_cifrado_DES = cifrar_DES(key_DES, iv_DES, texto_input)
88 descifrar_DES(key_DES, iv_DES, texto_cifrado_DES)
89
90 print("\n-----\n\n")
91
92
93 key_AES = ajustar_clave(key_input, 32, "Clave_AES-256")
94 iv_AES = ajustar_clave(iv_input, 16, "IV_AES-256")
95 print("\n---Cifrado y Descifrado con AES-256---")
96 texto_cifrado_AES = cifrar_AES(key_AES, iv_AES, texto_input)
97 descifrar_AES(key_AES, iv_AES, texto_cifrado_AES)
98
99 print("\n-----\n\n")
100
101 key_3DES = ajustar_clave(key_input, 24, "Clave_3DES")
102 iv_3DES = ajustar_clave(iv_input, 8, "IV_3DES")
103 print("\n---Cifrado y Descifrado con 3DES---")
104 texto_cifrado_3DES = cifrar_3DES(key_3DES, iv_3DES, texto_input)
105 descifrar_3DES(key_3DES, iv_3DES, texto_cifrado_3DES)

```

Bloque de código 8: Código completo algoritmos de cifrado.

2.5. Compara los resultados con un servicio de cifrado online

Para esta sección se utilizara el servicio de cifrado AES-256 [4].

Los datos a ingresar son:

- **Llave:** holaholacomocomotevatevaholacom.
- **Vector de inicializacion (IV):** chaochaochaochao.
- **Mensaje:** hackhackha.

```

--- Cifrado y Descifrado con AES-256 ---
Texto cifrado con AES-256 (hex): 8895e224fda98a27a256dc3ebf8be17e
Texto descifrado con AES-256: hackhackha

```

Figura 8: Resultado del cifrado con los datos anteriores usando el algoritmo AES-256.

The image shows a web-based AES encryption tool. It has a title 'AES Encryption' and several input fields and dropdowns. The 'Enter Plain Text to Encrypt' field contains 'hackhackha'. 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 'chaochaochaochao'. The 'Key Size in Bits' dropdown is set to '256'. The 'Enter Secret Key' field contains 'holaholacomocomotivatevaholacom'. The 'Output Text Format' has radio buttons for 'Base64' and 'Hex', with 'Hex' selected. There is a black 'Encrypt' button. Below the button, the 'AES Encrypted Output' field displays the result: '8895E224FDA98A27A256DC3EBF8BE17E'.

AES Encryption

Enter Plain Text to Encrypt

hackhackha

Select Cipher Mode of Encryption ?

CBC

Select Padding ?

PKCS5Padding

Enter IV (Optional) ?

chaochaochaochao

Key Size in Bits ?

256

Enter Secret Key ?

holaholacomocomotivatevaholacom

Output Text Format ☐ Base64 ☒ Hex

Encrypt

AES Encrypted Output

8895E224FDA98A27A256DC3EBF8BE17E

Figura 9: Resultado del cifrado con el servicio online AES Encryption, configurado para AES-256 [4].

- **Resultado de codigo python AES-256:** 8895e224fda98a27a256dc3ebf8be17e.
- **Resultado de la pagina con AES-256:** 8895e224fda98a27a256dc3ebf8be17e.

Como se puede ver, en ambos casos se obtiene el mismo resultado, ya que se utiliza exactamente la misma clave, IV y mensaje para cifrar.

En otras páginas he realizado pruebas, y las variaciones que he encontrado en los resultados se deben principalmente al tipo de padding seleccionado para el algoritmo AES-256 que cada página utiliza.

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

Suponiendo un caso en el que se manejan datos sensibles, como en un banco o un centro médico, donde la información personal o financiera es delicada se recomienda el siguiente algoritmo de cifrado simétrico:

AES-256, con una clave de 256 bits, ofrece una cantidad inmensa de combinaciones posibles, es decir, 2^{256} combinaciones, lo que dificulta los ataques de fuerza bruta a un nivel casi imposible de resolver en la actualidad. Además, AES-256 opera con 14 rondas de cifrado que utilizan operaciones matemáticas complejas sobre los bits, de modo que un ligero cambio en el texto original genera resultados completamente diferentes [5]. Hasta el momento, no existen vulnerabilidades prácticas comprobadas de este algoritmo que lo hagan susceptible a ataques, destacando su resistencia contra técnicas avanzadas. AES-256 también es eficiente, ya que aprovecha el hardware y software disponibles en la actualidad, permitiendo manejar grandes cantidades de datos sin retrasos notables. Por último, AES-256 ha sido aprobado por el NIST y está incluido en estándares de cifrado internacionales, como el estándar de seguridad de datos de la industria de tarjetas de pago (PCI-DSS) y el Reglamento General de Protección de Datos (GDPR) de la Unión Europea [5][6][7].

Ahora, suponiendo que el cliente desea implementar *hashes* en lugar del cifrado y descifrado recomendados, tendríamos un buen nivel de seguridad y cierta facilidad de implementación. Aun así, el uso de hashes tiene el problema de que, si se filtra la información sensible de la función de hashing (como la semilla, longitud, etc.), podría replicarse fácilmente. En caso de filtrarse, sería necesario eliminar la información para proteger los datos, ya que los datos hashados no se pueden descifrar, a diferencia de los algoritmos de cifrado que se han estudiado en este informe. Además, implementar hashes limita considerablemente el uso de los datos, ya que complica la utilización de tipos de datos que no sean solo nombres de usuario o contraseñas.

Conclusiones y comentarios

Este laboratorio nos permitió probar los algoritmos DES, AES-256 y 3DES utilizando Python y la biblioteca PyCryptodome. Cada uno de estos algoritmos presenta particularidades en cuanto a la longitud de la clave y el IV, y su elección depende de factores como la velocidad y el nivel de seguridad deseado. Sin embargo, es importante notar que tanto DES como 3DES están actualmente vulnerados.

El proceso de ajustar las claves al tamaño requerido para cada algoritmo y de garantizar la configuración correcta del vector de inicialización (IV) fue tedioso y requirió un esfuerzo adicional. La investigación previa en diversas fuentes y documentación, principalmente de la propia biblioteca, resultó crucial para entender el funcionamiento de estos algoritmos.

Obtener cifrados y descifrados consistentes fue fundamental durante la actividad, en la cual también se verificó el funcionamiento con distintos servicios en línea. Esto permitió una mayor comprensión de los parámetros que utilizan los algoritmos y cómo un ligero cambio puede afectar considerablemente el resultado. Este aspecto es relevante, ya que permite observar el conocido "efecto avalancha," deseable y crucial en los algoritmos criptográficos.

En conclusión, este laboratorio evidenció que la elección del algoritmo de cifrado está directamente ligada al nivel de seguridad requerido en cada caso. Además, se destaca que, con el tiempo, surgen nuevos algoritmos más eficientes y con un mayor nivel de seguridad en comparación con los antiguos. Por lo tanto, es importante mantenerse informado y actualizado respecto a la criptografía y la seguridad en redes.

Referencias

- [1] PyCryptodome Documentation. *Single DES*. Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des.html> [Accedido: 5 de noviembre de 2024]. 2024.
- [2] PyCryptodome Documentation. *AES Cipher in PyCryptodome*. Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html> [Accedido: 5 de noviembre de 2024]. 2024.
- [3] PyCryptodome Documentation. *Triple DES (3DES) en PyCryptodome*. Disponible en: <https://pycryptodome.readthedocs.io/en/latest/src/cipher/des3.html> [Accedido: 5 de noviembre de 2024]. 2024.
- [4] Devglan. *AES Encryption and Decryption Online Tool*. Accedido: 2023-11-06. 2023. URL: <https://www.devglan.com/online-tools/aes-encryption-decryption>.
- [5] National Institute of Standards y Technology. “FIPS 197: Advanced Encryption Standard (AES)”. En: (2001). Accedido el 06 de noviembre de 2024. URL: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>.
- [6] Payment Card Industry Security Standards Council. “PCI Data Security Standard (PCI DSS)”. En: (2024). Accedido el 06 de noviembre de 2024. URL: https://www.pcisecuritystandards.org/pci_security/.
- [7] European Union. “Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo”. En: (2016). Accedido el 06 de noviembre de 2024. URL: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.