Informe Laboratorio 4

Sección x

Alumno x
e-mail: alumno.contacto@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 5
2.5. Compara los resultados con un servicio de cifrado online 6
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real 8
2.7. Caso de Uso de Cifrado Simétrico 8
2.8. Implementación de Hashes en Lugar de Cifrado Simétrico 9

3. Bibliografia **11**

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

Dado la investigación del tamaño en bytes de la clave y el IV, es dar como parametros para los algoritmos:

1. DES (Data Encryption Standard)

   El algoritmo DES utiliza una clave de 56 bits y un vector de inicialización (IV) de 64 bits. Fue ampliamente adoptado en su tiempo pero hoy es considerado inseguro debido a la posibilidad de ataques de fuerza bruta. A pesar de su velocidad relativamente baja en hardware moderno, marcó un hito en la criptografía antes de ser reemplazado por algoritmos más robustos.

2. AES-256 (Advanced Encryption Standard)

   El algoritmo AES-256 emplea una clave de 256 bits y un IV de 128 bits, ofreciendo una seguridad extremadamente alta. Es rápido y eficiente, convirtiéndose en el estándar de facto para la protección de datos sensibles en aplicaciones comerciales, gubernamentales y militares. Su resistencia a todos los ataques conocidos hasta la fecha lo hace muy fiable y ampliamente utilizado.

3. 3DES (Triple Data Encryption Standard)

   El algoritmo 3DES mejora la seguridad del DES aplicando tres rondas de cifrado con tres claves de 56 bits, resultando en una clave efectiva de 168 bits y un IV de 64 bits. Aunque proporciona mayor seguridad comparado con el DES, su rendimiento es inferior al de AES-256 y su uso ha disminuido en favor de algoritmos más modernos y eficientes.

Si deseamos ver esto resumido, seria algo asi:

2.2. Solicita datos de entrada desde la terminal

Se le solicita al usuario la key, el vector de inicialización (IV) y el texto a cifrar. A través del siguiente código:

Característica	DES	AES-256	3DES
Tamaño de clave	56 bits	256 bits	168 bits (tres claves de 56 bits)
Tamaño de IV	64 bits	128 bits	64 bits
Seguridad	Inseguro	Muy seguro	Relativamente seguro
Velocidad	Lento	Muy rápido	Más lento que AES-256
Uso actual	Obsoleto	Estándar de facto	Menos común
Número de rondas	16	10, 12 o 14	48 (tres veces 16 rondas de DES)

Tabla 1: Comparación detallada de DES, AES-256 y 3DES

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

# Solicitar datos al usuario
key = input("Ingrese la clave (key): ")
iv = input("Ingrese el vector de inicialización (IV): ")
texto = input("Ingrese el texto a cifrar: ")

# Convertir IV a bytes
try:
    iv_bytes = bytes.fromhex(iv)
except ValueError:
    print("El IV debe estar en formato hexadecimal.")
    exit(1)
```

Figura 1: Código que cumple con lo solicitado

2.3. Valida y ajusta la clave según el algoritmo

El programa ajusta la clave para que tenga la longitud correcta requerida por cada algoritmo. Si es necesario, agrega bytes adicionales o trunca la clave a la longitud necesaria. Luego, imprime la clave ajustada.A través del siguiente código:

```python
# Asegurar que el IV tenga la longitud correcta para cada algoritmo
def ajustar_iv(iv_bytes, tamaño_necesario):
    if len(iv_bytes) < tamaño_necesario:
        iv_bytes += get_random_bytes(tamaño_necesario - len(iv_bytes))
    elif len(iv_bytes) > tamaño_necesario:
        iv_bytes = iv_bytes[:tamaño_necesario]
    return iv_bytes

iv_des = ajustar_iv(iv_bytes, 8)
iv_aes = ajustar_iv(iv_bytes, 16)
iv_3des = ajustar_iv(iv_bytes, 8)

def ajustar_clave(key, tamaño_necesario):
    key_bytes = key.encode('utf-8')
    if len(key_bytes) < tamaño_necesario:
        key_bytes += get_random_bytes(tamaño_necesario - len(key_bytes))
    elif len(key_bytes) > tamaño_necesario:
        key_bytes = key_bytes[:tamaño_necesario]
    return key_bytes

# Ajustar claves
key_des = ajustar_clave(key, 8)
key_aes = ajustar_clave(key, 32)
key_3des = ajustar_clave(key, 24)

print(f"Clave DES ajustada: {key_des.hex()}")
print(f"Clave AES ajustada: {key_aes.hex()}")
print(f"Clave 3DES ajustada: {key_3des.hex()}")
print(f"IV ajustado para DES: {iv_des.hex()}")
print(f"IV ajustado para AES: {iv_aes.hex()}")
print(f"IV ajustado para 3DES: {iv_3des.hex()}")
```

Figura 2: Código que cumple con lo solicitado

2.4. Implementa el cifrado y descifrado en modo CBC

El programa implementa funciones para cifrar y descifrar utilizando los algoritmos DES, AES-256 y 3DES en modo CBC. Asegura que se use el IV proporcionado por el usuario y luego imprime tanto el texto cifrado como el descifrado. A través del siguiente código:

```python
def cifrar_descifrar_des(texto, key, iv):
    cipher = DES.new(key, DES.MODE_CBC, iv)
    texto_cifrado = cipher.encrypt(pad(texto.encode('utf-8'), DES.block_size))
    cipher = DES.new(key, DES.MODE_CBC, iv)
    texto_descifrado = unpad(cipher.decrypt(texto_cifrado), DES.block_size)
    return texto_cifrado, texto_descifrado.decode('utf-8')

def cifrar_descifrar_aes(texto, key, iv):
    cipher = AES.new(key, AES.MODE_CBC, iv)
    texto_cifrado = cipher.encrypt(pad(texto.encode('utf-8'), AES.block_size))
    cipher = AES.new(key, AES.MODE_CBC, iv)
    texto_descifrado = unpad(cipher.decrypt(texto_cifrado), AES.block_size)
    return texto_cifrado, texto_descifrado.decode('utf-8')Z

def cifrar_descifrar_3des(texto, key, iv):
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
    texto_cifrado = cipher.encrypt(pad(texto.encode('utf-8'), DES3.block_size))
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
    texto_descifrado = unpad(cipher.decrypt(texto_cifrado), DES3.block_size)
    return texto_cifrado, texto_descifrado.decode('utf-8')

# Ejecutar cifrado y descifrado
texto_cifrado_des, texto_descifrado_des = cifrar_descifrar_des(texto, key_des, iv_des)
texto_cifrado_aes, texto_descifrado_aes = cifrar_descifrar_aes(texto, key_aes, iv_aes)
texto_cifrado_3des, texto_descifrado_3des = cifrar_descifrar_3des(texto, key_3des, iv_3des)

# Imprimir resultados
print(f"Texto cifrado DES: {texto_cifrado_des.hex()}")
print(f"Texto descifrado DES: {texto_descifrado_des}")
print(f"Texto cifrado AES: {texto_cifrado_aes.hex()}")
print(f"Texto descifrado AES: {texto_descifrado_aes}")
print(f"Texto cifrado 3DES: {texto_cifrado_3des.hex()}")
print(f"Texto descifrado 3DES: {texto_descifrado_3des}")
```

Figura 3: Código que cumple con lo solicitado

2.5. Compara los resultados con un servicio de cifrado online

Se llevara a cabo una comparación del resultado del algoritmo AES-256 de nuestro programa de cifrado con los generados por un servicio de cifrado online. Esta comparación es crucial para validar la precisión y eficacia de nuestra implementación de los algoritmos de cifrado. Al contrastar uno de los resultados cifrados, podremos asegurar que nuestro programa está funcionando correctamente y siguiendo los estándares esperados. Para este propósito, utilizaremos herramientas de cifrado disponibles en línea que soportan el algoritmo AES-256.

Partiremos por ver la prueba desde nuestro programa:

Ahora veremos los resultados del servicio online, para lo cual utilizaremos la **Herramienta de encriptación y desencriptación Online**, los cuales fueron:

```
Ingrese la clave (key): 12345678
Ingrese el vector de inicialización (IV): 0102030405060708
Ingrese el texto a cifrar: Hola mundo
Clave DES ajustada: 3132333435363738
Clave AES ajustada: 3132333435363738ce9038369369892f892e2935f5d792ceaeb98b31c29b4bbd
Clave 3DES ajustada: 313233343536373850125d4c05b24958296d4178087cb949
IV ajustado para DES: 0102030405060708
IV ajustado para AES: 0102030405060708935f06db94f2dee6
IV ajustado para 3DES: 0102030405060708
Texto cifrado DES: a1f7fbcef3cf33773a53f1a7a3d11c2c
Texto descifrado DES: Hola mundo
Texto cifrado AES: b4f098f15665525652bc4f782b023743
Texto descifrado AES: Hola mundo
Texto cifrado 3DES: b61255a07979cc4298d977f198814548
Texto descifrado 3DES: Hola mundo
```

Figura 4: Resultados de prueba del programa con AES-256

Figura 5: Resultado de prueba con servicio de cifrado online con AES-256-CBC Encriptación

Figura 6: Resultado de prueba con servicio de cifrado online con AES-256-CBC Desencriptación

Podemos analizar que tenemos distintos resultados, dado que en el servicio online utilizado implementa un vector de manera interna, lo cual no nos permite realizar una comparativa apropiada, además de verse reflejado que no es la única key implementada.

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

2.7. Caso de Uso de Cifrado Simétrico

Un caso o situación donde se usuaría cifrado simétrico es emn transecciones bancarias, en dondela institución financiera en donde se debe asegurar que las tranferencias o transacciones bancarias se realicen de manera segura. En donde nos vemos enfrentados al manejo de datos personales que deben tener un trato confidencial, como números de cuenta, montos de

transacción y datos de identificación personal, los cuales deben ser protegidos contra accesos no autorizados.

Una solución factible es utilizar cifrado simétrico para la protección de datos, eligiendo al algoritmo Advanced Encryption Standard o más conocido como AES con una clave de 256 bits. Dado que:

1. Eficiencia

   Este algoritmo es eficiente en términos de rendimiento, lo cuál es sumamente importante en las aplicaciones que funcionan en tiempo real.

2. Seguridad

   AES-256 contiene un cifrado altamente seguro, debido a su fortaleza contra ataques criptográficos.

3. Soporte

   AES es compatible con la mayoría de las bibliotecas criptográficas modernas y plataformas de software, facilitando la integración en diferentes sistemas.

4. Normativa

   Cumple con los estándares de seguridad establecidos por entidades gubernamentales y reguladoras, garantizando el cumplimiento normativo.

2.8. Implementación de Hashes en Lugar de Cifrado Simétrico

Cuando se sugiere la implementación de hashes en lugar de cifrado simétrico, es fundamental comprender que estas dos técnicas criptográficas sirven a propósitos diferentes y se complementan en lugar de ser intercambiables. Los hashes, como los generados por algoritmos SHA-256, se utilizan principalmente para asegurar la integridad y la autenticidad de los datos. Un hash convierte los datos en una huella digital única; cualquier cambio, por pequeño que sea, en los datos originales resultará en un hash completamente diferente. Esto permite detectar alteraciones o manipulaciones en los datos, lo que es vital para la verificación de integridad.

Por otro lado, el cifrado simétrico, como el proporcionado por algoritmos como AES (Advanced Encryption Standard) y DES (Data Encryption Standard), se utiliza para mantener la confidencialidad de la información. Mediante el uso de una clave secreta compartida, el cifrado simétrico transforma datos legibles (texto plano) en una forma ilegible (texto cifrado), que solo puede revertirse a su forma original mediante la clave correcta. Esto asegura que solo las partes autorizadas puedan acceder a la información sensible, protegiéndola de accesos no autorizados.

En escenarios como las transacciones bancarias, tanto la confidencialidad como la integridad de los datos son de suma importancia. Las transacciones bancarias involucran información crítica, como números de cuenta, montos de transacción y detalles de identificación personal

(PII), que deben protegerse en todo momento. El cifrado simétrico es esencial para asegurar que esta información permanezca confidencial durante la transmisión entre servidores y aplicaciones móviles, protegiéndola de interceptaciones maliciosas.

Una solución ideal en este contexto es la combinación de cifrado simétrico y hashes. Por ejemplo, utilizando AES-256 para cifrar los datos de la transacción se asegura que la información sea confidencial. AES-256 es ampliamente reconocido por su seguridad y eficiencia, y es capaz de resistir ataques criptográficos avanzados. Además, la implementación de un HMAC (Hash-based Message Authentication Code) antes de cifrar los datos puede proporcionar autenticidad e integridad. Un HMAC se genera a partir de una clave secreta y los datos a proteger, produciendo una firma digital que garantiza que los datos no han sido alterados y que provienen de una fuente legítima.

Es importante argumentar que los hashes no deben reemplazar el cifrado simétrico, sino que deben utilizarse en conjunto para maximizar la seguridad de las transacciones bancarias. Mientras que los hashes aseguran la integridad y la autenticidad de los datos, el cifrado simétrico protege la confidencialidad de la información. La combinación de estas técnicas proporciona una solución criptográfica robusta que puede proteger contra una amplia gama de amenazas.

En resumen, cuando se considera la seguridad de las transacciones bancarias, es fundamental utilizar tanto el cifrado simétrico para proteger la confidencialidad de los datos como los hashes para asegurar su integridad. Esta combinación no solo maximiza la seguridad, sino que también cumple con los estándares y regulaciones de seguridad más estrictos, garantizando la protección de la información sensible de los usuarios en todo momento.

Conclusiones y comentarios

En este análisis detallado, hemos explorado las características y aplicaciones del cifrado simétrico, destacando los algoritmos DES, AES-256 y 3DES. Hemos visto cómo cada uno tiene sus propias ventajas y limitaciones en términos de tamaño de clave, tamaño de vector de inicialización (IV), seguridad, velocidad y uso actual. DES, aunque históricamente significativo, es ahora considerado inseguro. 3DES, con su mejora en seguridad sobre DES, sigue siendo más lento y menos eficiente que AES-256, que es el estándar de facto en la actualidad debido a su alta seguridad y eficiencia.

La clave de una implementación exitosa de cifrado simétrico radica en la correcta gestión y ajuste de las claves y IVs, asegurando que cumplan con las especificaciones del algoritmo elegido. Esto se ilustra en el código presentado, que ajusta y valida las claves y IVs para los algoritmos DES, AES-256 y 3DES, y realiza el cifrado y descifrado en modo CBC.

La comparación con servicios de cifrado online es esencial para validar la precisión de nuestras implementaciones, aunque debemos considerar las limitaciones y diferencias en los vectores de inicialización utilizados por estos servicios.

En cuanto a la aplicabilidad del cifrado simétrico en la vida real, destacamos su uso en transacciones bancarias donde la confidencialidad y la integridad de los datos son críticas. AES-256 es altamente recomendado debido a su balance óptimo entre seguridad y rendimien-

to.

Por último, la discusión sobre la implementación de hashes en lugar de cifrado simétrico resalta la importancia de comprender las diferencias y propósitos de cada técnica. Los hashes aseguran la integridad y autenticidad de los datos, mientras que el cifrado simétrico protege su confidencialidad. La combinación de ambas técnicas proporciona una solución de seguridad robusta, maximizando la protección contra una amplia gama de amenazas.

En resumen, la seguridad de la información en las transacciones bancarias y otros escenarios críticos se beneficia enormemente del uso conjunto de cifrado simétrico y hashes. Esta estrategia no solo maximiza la seguridad, sino que también asegura el cumplimiento con los estándares y regulaciones más estrictos, garantizando la protección de la información sensible en todo momento.

3. Bibliografia

1. Todo sobre criptografía: Algoritmos de clave simétrica y asimétrica
2. Product version no longer supported
3. Vector de inicialización
4. ¿Qué es el cifrado 3DES y cómo funciona?
5. 3DES
6. ¿Qué es el DES?
7. Claves y operaciones DES (Data Encryption Standard)