Informe Laboratorio 4

Sección 3

Lucas Araya
e-mail: lucas.araya_t@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 8
2.6. Describe la aplicabilidad del cifrado simétrico en la vida real 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

Tanto DES como 3DES utilizan un vector de inicialización de 8 bytes, pero presentan diferencias en la longitud de sus claves efectivas: DES emplea una clave efectiva de 56 bits, mientras que 3DES tiene una clave efectiva de 112 bits. Por otro lado, AES en su versión AES-256 utiliza una clave de 256 bits y un vector de inicialización de 16 bytes (128 bits), no de 8 bytes como en DES y 3DES.

La principal diferencia entre estos algoritmos radica en sus arquitecturas y en el nivel de seguridad que ofrecen. DES y 3DES se basan en una estructura de red de Feistel, donde cada ronda realiza operaciones de permutación y sustitución de manera intercalada. En cambio, AES-256 emplea una red de sustitución-permutación de 14 rondas, en la que cada ronda involucra operaciones de sustitución de bytes, desplazamiento de filas, mezcla de columnas y adición de la clave de la ronda, lo que proporciona mayor seguridad y eficiencia en comparación con DES y 3DES.

2.2. Solicita datos de entrada desde la terminal

La solicitud de datos se realiza de la manera tradicional, utilizando la función `input` que viene por defecto en Python. A continuación, se puede observar el código que permite el ingreso de datos desde la consola.

```python
# el usuario ingresa las key's por la consola
des_key = input("Ingrese la llave de DES: ")
des3_key = input("Ingrese la llave de 3DES: ")
aes_key = input("Ingrese la llave de AES: ")

# Ingreso de vectores
des_iv = input("Ingrese el vector de inicializacion de DES: ")
des3_iv = input("Ingrese el vector de inicializacion de 3DES: ")
aes_iv = input("Ingrese el vector de inicializacion de AES: ")

text_to_cipher = input("Ingrese el texto a cifrar: ")
```

Figura 1: Implementación de ingreso de datos desde la consola

En este caso, se usarán los siguientes datos: para DES se utilizará la clave `mnbvcxza` y el vector de inicialización `abcdef12`; para 3DES, `qw1er2ty3ui4op5l` y `ghijk345`; y, por último, para AES se utilizarán `asdfghjklzxcvbn` y `pqrstuvwx9876543`. En la Figura 2 se observa cómo se ingresan estas claves desde la consola.

```
Ingrese la llave de DES: mnbvcxza
Ingrese la llave de 3DES: qw1er2ty3ui4op5l
Ingrese la llave de AES: asdfghjklzxcvbn
Ingrese el vector de inicializacion de DES: abcdef12
Ingrese el vector de inicializacion de 3DES: ghijk345
Ingrese el vector de inicializacion de AES: pqrstuvwx9876543
```

Figura 2: Ingreso de datos desde la consola

El texto a cifrar será la cadena de texto `"hola mundo"`. Como se puede observar, solo en la clave para DES se utilizó el tamaño adecuado, esto con el objetivo de ver como se generan formatean los datos.

2.3. Valida y ajusta la clave según el algoritmo

Para validar correctamente los datos, se comenzó leyendo la documentación de la biblioteca `Cipher`. Según esta, el formato que se debe recibir para el cifrado debe ser de tipo *bytes*.

Las claves y los vectores de inicialización también deben estar en *bytes*, conforme a lo descrito en la investigación. Sin embargo, para el algoritmo `3DES`, se requiere una clave de 24 bytes, que consiste en tres subclaves independientes. Esta configuración proporciona 112 bits de seguridad efectiva y previene patrones repetitivos, lo que incrementa la resistencia contra ataques.

A continuación, en las Figuras 3 y 4, se muestran las funciones utilizadas para formatear las claves y los vectores de inicialización en caso de que sea necesario. Es decir, si los datos exceden el tamaño esperado, se truncan, y si son más cortos, se completan utilizando la función `get_random_bytes` de `Cipher`.

```python
def format_key(alg, key):  3 usages
    if alg == "DES":
        max_size = 8
    elif alg == "3DES":
        max_size = 24
    elif alg == "AES":
        max_size = 32
    else:
        return None

    key = key.encode()
    if len(key) > max_size:
        key = key[:max_size]
    elif len(key) < max_size:
        diff = max_size - len(key)
        key += get_random_bytes(diff)

    return key
```

Figura 3: Implementación de función para ajustar claves

```python
def format_iv(alg, iv):  3 usages
    if alg == "DES" or alg == "3DES":
        max_size = 8
    elif alg == "AES":
        max_size = 16
    else:
        return None

    iv = iv.encode()
    if len(iv) > max_size:
        iv = iv[:max_size]
    elif len(iv) < max_size:
        diff = max_size - len(iv)
        iv += get_random_bytes(diff)

    return iv
```

Figura 4: Implementación de función para ajustar vectores de inicialización

A continuación, se pueden observar los valores de las claves y los vectores de inicialización ajustados. Las claves formateadas se muestran en formato hexadecimal, ya que al rellenar las contraseñas más cortas, el formato hexadecimal facilita su visualización de manera más clara y comprensible.

```
DES: key=6d6e627663787a61   iv=b'abcdef12'
3DES: key=717731657232747933756934 6f70356c4853d86c227991bc  iv=b'ghijk345'
AES: key=6173646667686a6b6c7a786376626e139abea093b044621235eaccb06e41559e   iv=b'pqrstuvwx9876543'
```

Figura 5: Valores ajustados de claves y vectores de inicialización

Las key's formateadas son: `6d6e627663787a61` para DES, `71773165723274793375693 46f70356c4853` para 3DES, y `6173646667686a6b6c7a786376626e139abea093b044621235eaccb06e41559e` para AES. Estos valores aseguran que las claves cumplan con el tamaño adecuado y garanticen una mayor seguridad en los cifrados correspondientes.

2.4. Implementa el cifrado y descifrado en modo CBC

Para la implementación del cifrado y descifrado, se utiliza la misma biblioteca. Las funciones son bastante simples: primero se declara el tipo de algoritmo y luego se realiza el cifrado o descifrado, dependiendo de la función. Si estamos cifrando, se utiliza la función `pad` para aplicar el padding estándar, y en el caso de descifrar, se utiliza la función `unpad`. Es importante el orden de las operaciones: para cifrar, el padding se aplica al texto antes de cifrar; para descifrar, primero se descifra y luego se elimina el padding con `unpad`. Otro punto importantes es que tanto la key, el iv y la cadena deben estar en formato de bytes, si no las funciones arrojaran error.

En las Figuras 6 y 7, se muestran las funciones de cifrado y descifrado.

```python
def aes_encrypt(text, key, iv):  1 usage
    cipher = AES.new(key, AES.MODE_CBC, iv)
    cipher_data = cipher.encrypt(pad(text, AES.block_size))
    return cipher_data


def aes_decrypt(cipher_data, key, iv):  1 usage
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = unpad(cipher.decrypt(cipher_data), AES.block_size)
    return decrypted_data


def des_encrypt(text, key, iv):  1 usage
    cipher = DES.new(key, DES.MODE_CBC, iv)
    cipher_data = cipher.encrypt(pad(text, DES.block_size))
    return cipher_data
```

Figura 6: Funciones de cifrado AES y DES, y descifrado AES

```python
def des_decrypt(cipher_data, key, iv):  1 usage
    cipher = DES.new(key, DES.MODE_CBC, iv)
    decrypted_data = unpad(cipher.decrypt(cipher_data), DES.block_size)
    return decrypted_data

def des3_encrypt(text, key, iv):  1 usage
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
    cipher_data = cipher.encrypt(pad(text, DES3.block_size))
    return cipher_data

def des3_decrypt(cipher_data, key, iv):  1 usage
    cipher = DES3.new(key, DES3.MODE_CBC, iv)
    decrypted_data = unpad(cipher.decrypt(cipher_data), DES3.block_size)
    return decrypted_data
```

Figura 7: Funciones de cifrado 3DES, y descifrado DES y 3DES

Al realizar la encriptación, se obtienen los siguientes resultados:

```
DES encrypt: 8b919edc696d02c1e5fb403827efacac
3DES encrypt: a9d91cf2eb0414090f7d7b11bd425b18
AES encrypt: 14af61c0febc4bc64a214fe386a14781
DES decrypt: hola mundo
3DES decrypt: hola mundo
AES decrypt: hola mundo
```

Figura 8: Resultados de la encriptación y desencriptación

Como se puede observar, el texto desencriptado es idéntico a la cadena original, lo que nos permite corroborar que la implementación del cifrado y descifrado es correcta.

2.5. Compara los resultados con un servicio de cifrado online

Para realizar la comparación, se utilizará la web `https://anycrypt.com/crypto/des`, una página que permite cifrar y descifrar utilizando el algoritmo DES. Lo primero será convertir el texto de formato hexadecimal a una cadena, para lo cual se usará la herramienta en línea `https://codebeautify.org/hex-string-converter`, que realiza esta conversión. Al ingresar la clave en formato hexadecimal, se obtiene el mismo valor que en la clave original, lo que corrobora la precisión de la conversión realizada en la web.

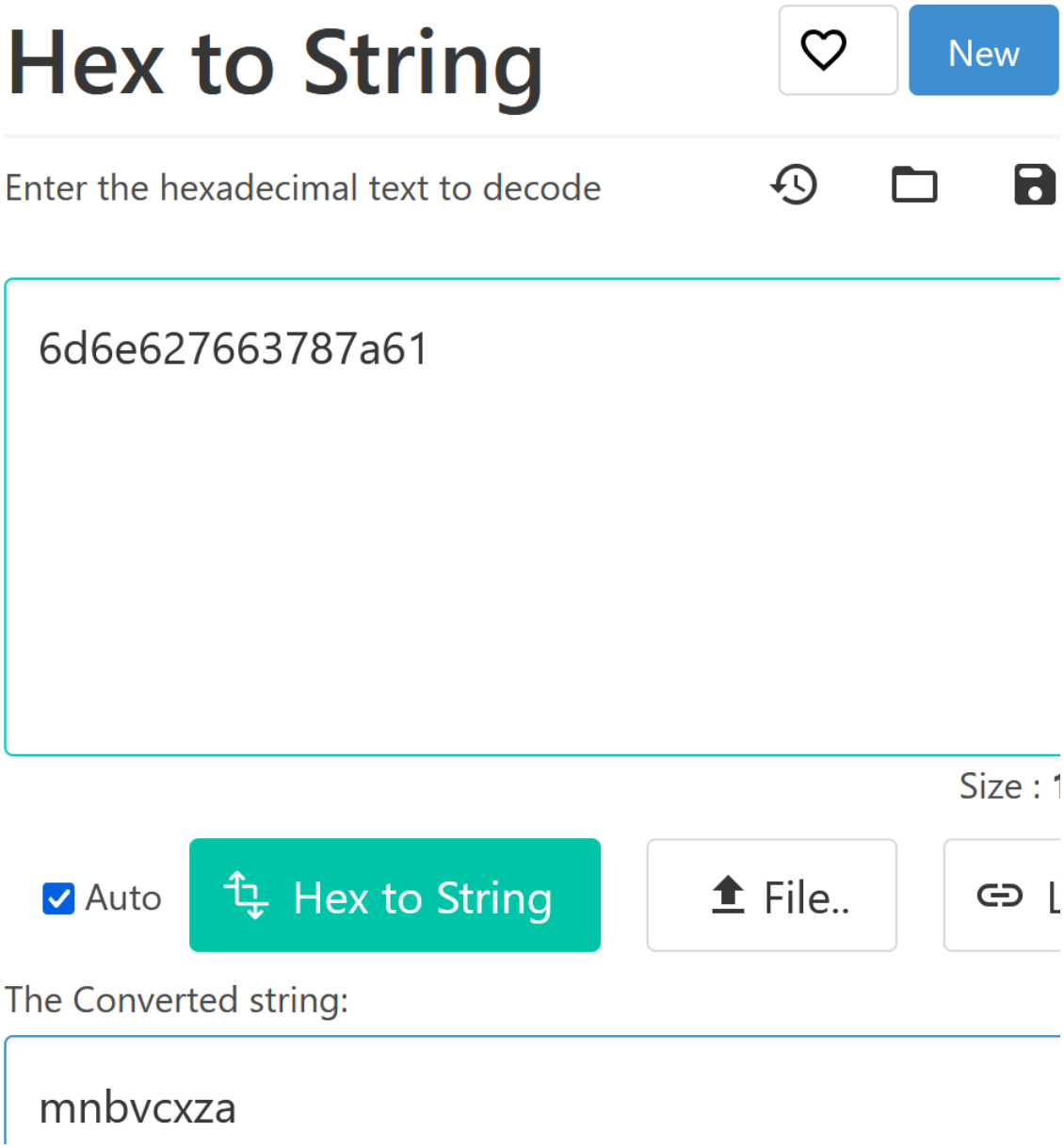

Figura 9: Conversión de hexadecimal a cadena de texto

El siguiente paso es utilizar la web de cifrado DES para verificar que los resultados sean equivalentes. La página permite ingresar tanto la clave como el vector de inicialización. En el campo de texto a cifrar, se ingresa el mensaje en formato de cadena, mientras que para descifrar, se coloca la cadena de bytes obtenida. En las Figuras 10 y 11 se muestran los resultados de cifrado y descifrado obtenidos con esta herramienta en línea.

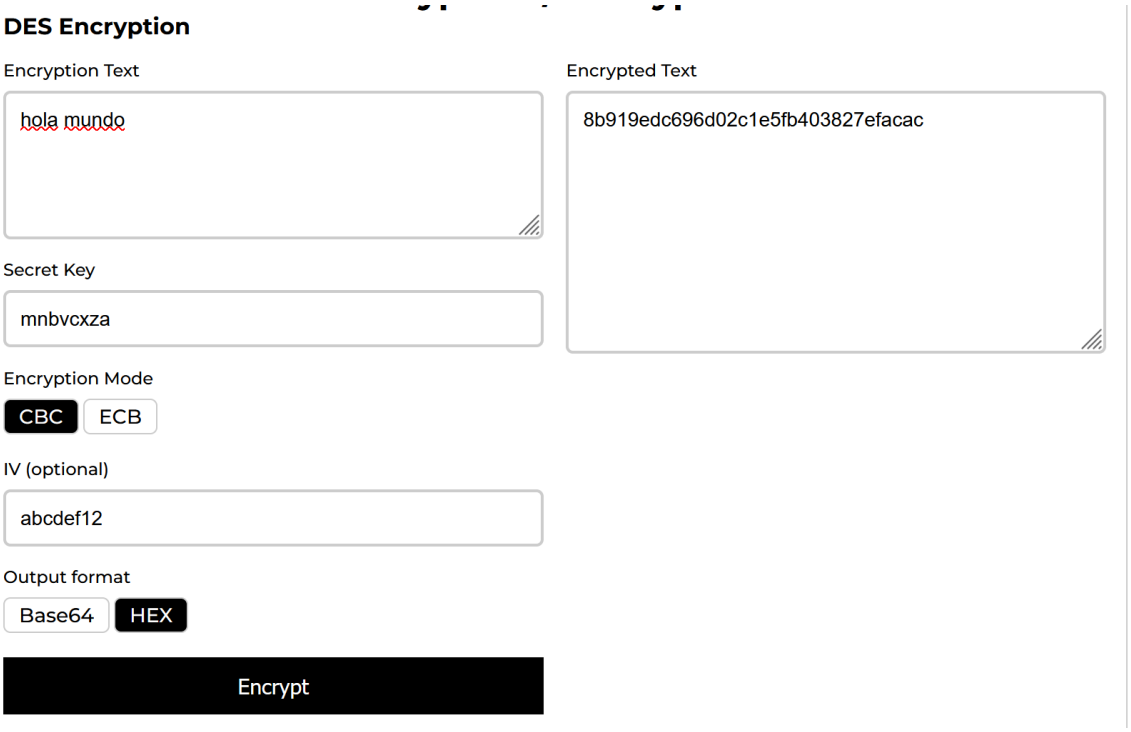

Figura 10: Cifrado con herramienta en línea

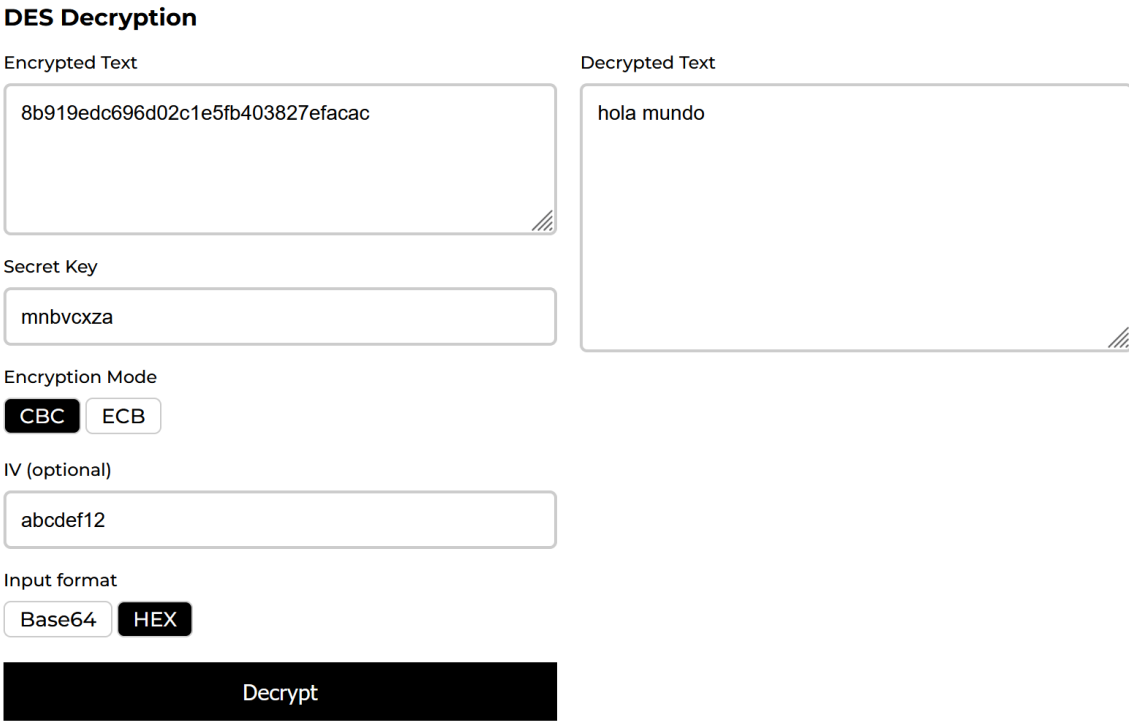

Figura 11: Descifrado con herramienta en línea

Como se observa en las imágenes, los resultados obtenidos son iguales, lo cual confirma que la implementación de cifrado y descifrado está correctamente desarrollada. Esto valida que el algoritmo DES implementado genera los mismos resultados que la herramienta en línea, demostrando su precisión y efectividad en el procesamiento de datos.

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

Un caso donde se usaría cifrado simétrico es en la comunicación segura entre un cliente y un servidor dentro de una red interna corporativa. Por ejemplo, en una empresa que maneja datos sensibles, como credenciales o registros financieros, se necesita proteger la transmisión de información para evitar su interceptación en caso de un ataque interno o externo.

Recomendaría el uso de **AES (Advanced Encryption Standard)** debido a su amplia aceptación como estándar seguro, su resistencia a ataques conocidos y su eficiencia en hardware y software modernos. Además, permite ajustar el tamaño de la clave (128, 192 o 256 bits), lo que lo hace flexible según los requerimientos de seguridad y rendimiento.

Si se sugiere reemplazar el cifrado simétrico por hashes, argumentaría que esto no es adecuado debido a que los hashes no protegen la confidencialidad. Su función principal es garantizar la integridad de los datos, pero no impedir que un atacante lea el contenido de los mensajes interceptados.

Si se busca complementar la seguridad, podría proponerse usar **HMAC (Hash-based Message Authentication Code)** para verificar integridad y autenticidad. Sin embargo, HMAC no sustituye al cifrado, por lo que el uso de un algoritmo como AES seguiría siendo necesario para garantizar la confidencialidad.

En conclusión, el uso de hashes no es una solución adecuada para proteger la confidencialidad, ya que su función es distinta y no cumple con los objetivos de seguridad en este caso.

Conclusiones y comentarios

En este laboratorio, se logró implementar exitosamente un programa en Python que utiliza los algoritmos de cifrado simétrico DES, 3DES y AES-256 en modo CBC, permitiendo la manipulación directa de claves, vectores de inicialización y textos desde la terminal. A través de este ejercicio, se verificaron los fundamentos teóricos y prácticos de cada algoritmo, incluyendo el ajuste de claves a los tamaños requeridos y el manejo de formato en bytes.

La validación realizada mediante herramientas de cifrado en línea confirmó que la implementación era precisa, ya que los resultados obtenidos en el programa coincidieron completamente con los de las herramientas externas. Esto demuestra la correcta comprensión y uso de la librería utilizada para criptografía, así como la robustez de los métodos desarrollados.

Adicionalmente, la comparación de los algoritmos permitió evidenciar diferencias importantes, como la mayor seguridad y eficiencia de AES-256 frente a DES y 3DES, aunque también se identificó que la elección del algoritmo depende del contexto y los requisitos de cada aplicación.

Este laboratorio representó una experiencia enriquecedora que combina teoría y práctica, consolidando habilidades en criptografía aplicada y su implementación en entornos reales.