PRACTICA 5

Gonzalez Jimenez Cristian Rafael

May 2025

1 REPORTE

en este texto vamos a explorar los procedimientos que se atravesaron para poder extender el codigo dado inicialmente para poder conectar una base de datos de un hospital y cifrar la informacion de los usuarios de esta.

Veamos que lo que primero se hizo fue crear la base de datos hospital que esta está dad por el archivo hopsital_shceme.sql ingresando el comando mysql ¡ hospital_scheme.sql. Una vez hecha la base de datos procedimos a extender el codigo para que podamos ingresar los datos a traves de un archivo .txt y asi agregar informacion sobre lo usuarios. Esto gracias la funcion que tenemos llamada procesar_pacientes donde ademas le damos una llave para que esto puedan ser cifrados y una vez teniendo esto leemos el archivo esto gracias a la variable "r" en el codigo y asi poder extraer los datos que seran ingresados en nuestra base de datos. Cabe destacar que esto tambien se logra usando un ciclo sobre los id de los pacientes

despues en el archivo original como en el que modificaremos usaremos el nonce generado aleatoriamento, esto con el fin de asegurar la seguridad de los datos y no ser vulnerable a un ataque donde se repita el nonce con la llave y asi poder ser atacados con el mensaje cifrador aplicando un XOR entre estos. Despues se cifra la informacion delicada con AES en modo de operacion CTR donde nosotros en el codigo entregable haremos uso de el modo de operacion GCM.

Y proximos de terminar metemos estos datos cifrado y encriptados y despues los borramos de la memoria ram

Ahora pasemos a nuestro codigo entregable de nuestra nueva base de datos donde haremos cada cosa. Esta base de datos se llamara hospitalnuevo y su archivo sera la misma hospitalnuevo.sql

donde aqui tuvimos que modificar un poco la tabla expediente y agregar unas columnas.

En este programa haremos que el usuario tenga la oportunidad de registrar la informacion o consultarla esto con la informacion de diagnostico_expediente.txt previamente determinado en el codigo.Observemos que en esta parte codificamos

nuestros datos delicados Ademas de cifrarlos y como se peude ver ahora hacemos uso de AES en modo de operacion GCM esto con las etiquetas generadas de diagnostico y de tratamiento, es decir, "diag_tag" y "treat_tag esto para podernos garantizar integridad

```
def procesar_pacientes(archivo, key):
    """Procesa el archivo de pacientes y devuelve una lista de registros cifrados"""
    registros = []

    with open(archivo, 'r') as f:
        contenido = f.read()
        try:
            datos = ast.literal_eval(contenido)
        except:
            print("Error al parsear el archivo. Asegúrese que tiene el formato correcto
            sys.exit(1)

    for id_paciente, info in datos.items():
        name = info['name']
        diagnosis = bytes(info['diagnosis'], 'utf-8')
        treatment = bytes(info['treatment'], 'utf-8')

        # Cifrar los datos con AES-GCM
        try:
            # Cifrar diagnóstico
            diag_cipher = AES.new(key, AES.MODE_GCM)
            diagnosis_ciphertext, diag_tag = diag_cipher.encrypt_and_digest(diagnosis)

            # Cifrar tratamiento
            treat_cipher = AES.new(key, AES.MODE_GCM)
            treatment_ciphertext, treat_tag = treat_cipher.encrypt_and_digest(treatment

            registros.append({
```

Una vez Hecho todo esto veamos que ahora procedemos a registrar los datos cifrados en nuestra base de datos:

Podemos obsevar que en esta parte del codigo para poder descifrar solo hacemos uso del nonce y la llave, o bien, como esta nombrado en el codigo "key" .

```
            treatment_ciphertext, treat_tag = treat_cipher.encrypt_and_digest(treatment

            registros.append({
                'name': name,
                'diagnosis': b64encode(diagnosis_ciphertext),
                'treatment': b64encode(treatment_ciphertext),
                'diag_nonce': b64encode(diag_cipher.nonce),
                'treat_nonce': b64encode(treat_cipher.nonce),
                'diag_tag': b64encode(diag_tag),
                'treat_tag': b64encode(treat_tag)
            })

        except Exception as e:
            print(f"Error al cifrar datos para {name}: {str(e)}")
            continue

        finally:
            # Limpiar variables sensibles
            clearmem(diagnosis)
            clearmem(treatment)
            del diag_cipher, treat_cipher

    return registros
```

```
def insertar_registros(registros, passwordSalt):
    """Inserta los registros en la base de datos"""
    mydb = None
    try:
        # Conexión directa (ajusta credenciales)
        mydb = MySQLdb.connect(
            host='localhost',
            user='root',      # Cambia por tu usuario
            password='',      # Cambia por tu contraseña
            database='hospitalnuevo',
            port=3306
        )
        cursor = mydb.cursor()

        for registro in registros:
            cursor.execute(
                """INSERT INTO expediente
                (nombre, diagnostico, tratamiento, passwordSalt, diag_nonce, treat_nonce, diag_tag, treat_tag)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s)""",
                (
                    registro['name'],
                    registro['diagnosis'],
                    registro['treatment'],
                    passwordSalt,
                    registro['diag_nonce'],
                    registro['treat_nonce'],
                    registro['diag_tag'],
                    registro['treat_tag']
                )
            )
            print(f"Insertado: {registro['name']}")
```

```
                    registro['diag_nonce'],
                    registro['treat_nonce'],
                    registro['diag_tag'],
                    registro['treat_tag']
                )
            )
            print(f"Insertado: {registro['name']}")

        mydb.commit()
        print(f"\nTotal insertados: {len(registros)} registros")

    except Exception as err:
        print(f"\nError al insertar: {err}")
        sys.exit(1)
    finally:
        if mydb:
            cursor.close()
            mydb.close()
            print("Conexión cerrada")
```

Ahora si queremos hacer la labor de poder hacer consulta de nuestros datos cifrados ingresando una contraseña y obteniendolos de manera clara a traves de un pequeño menu. Este codigo a continuacion hara esto:

```
def recuperar_paciente():
    """Recupera y descifra la información de un paciente específico"""
    nombre_paciente = input("\nIngrese el nombre del paciente a buscar: ")
    password = getpass("Ingrese la contraseña de cifrado: ")
    password_bytes = password.encode('utf-8')

    mydb = None
    try:
        # Conexión a la base de datos
        mydb = MySQLdb.connect(
            host='localhost',
            user='root',
            password='',
            database='hospitalnuevo',
            port=3306
        )
        cursor = mydb.cursor()

        # Buscar al paciente
        cursor.execute(
            """SELECT diagnostico, tratamiento, passwordSalt, diag_nonce, treat_nonce, diag_tag, treat_tag
            FROM expediente WHERE nombre = %s""",
            (nombre_paciente,)
        )

        resultado = cursor.fetchone()

        if not resultado:
            print(f"\nNo se encontró al paciente: {nombre_paciente}")
            return
```

```
/practica
        if not resultado:
            print(f"\nNo se encontró al paciente: {nombre_paciente}")
            return

        # Extraer datos cifrados
        (diagnostico_cifrado, tratamiento_cifrado, passwordSalt,
         diag_nonce, treat_nonce, diag_tag, treat_tag) = resultado

        # Decodificar datos desde Base64
        salt = b64decode(passwordSalt)
        key = scrypt(
            password=password_bytes,
            salt=salt,
            key_len=32,
            N=2**20,
            r=8,
            p=1
        )

        # Descifrar diagnóstico
        diag_cipher = AES.new(
            key,
            AES.MODE_GCM,
            nonce=b64decode(diag_nonce)
        )
        diagnostico = diag_cipher.decrypt_and_verify(
            b64decode(diagnostico_cifrado),
            b64decode(diag_tag)
        ).decode('utf-8')

        # Descifrar tratamiento
        treat_cipher = AES.new(
            key,
```

Ya para terminar veamos que culminamos con la llave al pedirla al usuario, llave(key) que nos ayudara a que cada paciente tenga un cifrado seguro, pues recordemos que generamos los nonce de manera aleatoria. Asi que nos prevenimos a un ataque como el que se menciono antes y ademas esta llave, como se pidio, este utiliza el algoritmo de derivacion de llaves scrypt ademas que a esta llave se le agrega la SALT o sazon con los parametros ahi indicados.

/practica

```
def main():
    print("1. Cifrar e insertar nuevos registros")
    print("2. Buscar y descifrar información de paciente")
    opcion = input("Seleccione una opción (1/2): ")

    if opcion == '1':
        # Obtener contraseña para cifrar los datos
        password = getpass("Ingrese la contraseña de cifrado: ")
        password_bytes = password.encode('utf-8')

        try:
            # Generar salt para derivación de clave
            passwordSalt = get_random_bytes(16)

            # Derivar clave de cifrado usando scrypt
            key = scrypt(
                password=password_bytes,
                salt=passwordSalt,
                key_len=32,
                N=2**20,
                r=8,
                p=1
            )

            # Procesar archivo de pacientes
            registros = procesar_pacientes("diagnosticos_tratamientos.txt", key)

            # Codificar salt para almacenamiento
            passwordSalt = b64encode(passwordSalt)

            # Insertar registros en la base de datos
            insertar_registros(registros, passwordSalt)
```

```
            )

            # Procesar archivo de pacientes
            registros = procesar_pacientes("diagnosticos_tratamientos.txt", key)

            # Codificar salt para almacenamiento
            passwordSalt = b64encode(passwordSalt)

            # Insertar registros en la base de datos
            insertar_registros(registros, passwordSalt)

            print("\nProceso completado exitosamente")

        except Exception as e:
            print(f"\nError en el proceso: {str(e)}")
            sys.exit(1)

        finally:
            # Limpiar variables sensibles
            clearmem(key) if 'key' in locals() else None
            clearmem(password_bytes)
            clearmem(password)

    elif opcion == '2':
        recuperar_paciente()

    else:
        print("Opción no válida")

if __name__ == "__main__":
    main()
```

——————————CONCLUSIONES——————————-

La verdad esta practica siento que me costo mucho pero de la misma manera me gusto mucho y es que como siempre pasa con estas practicas al ver en retrospectiva pienso que no estaba tan complicado pero diciendo esto ultimo desde un conocimiento adquirido. Ademas me parecio muy divertido manejarlo con base de datos y que nos pongan un caso que se relacione con la parte de ley, pues, esto me hace pensar que esto resulta ser algo muy apegado a la vida laboral y al dia a dia de las compañias e instituciones que protegen nuestros datos.

1.1 Enlace a github

https://github.com/rafiki04/PRACTICA-5-CYS.git