Gómez Trejo Gustavo Ali

Hernández Albino Edgar Alejandro

PROYECTO FINAL DEL SEMESTRE 2021-1

“CAJERO AUTOMÁTICO ATM”

El proyecto consiste en un cajero automático (ATM) que realiza operaciones bancarias con el servidor del Banco.

La comunicación entre el Cliente y el Servidor debe ser vía sockets TCP y debe ser CIFRADA (ENCRIPTADA), por lo que deben usar una llave privada que deben compartir entre cliente y servidor. La aplicación debe funcionar de la siguiente manera:

1. El programa Servidor debe iniciarse en el host servidor (en el puerto que decidas)

2. El programa Cliente debe iniciarse en el host cliente (pasando el nombre o IP y el puerto desde línea de comandos).

3. El Servidor acepta la conexión y una vez conectado el cliente, puede ejecutar alguno de los siguientes comandos. El comando le llega al Servidor vía socket, posteriormente se realiza la acción solicitada y debe devolver la salida al cliente:

“CONSULTAR”(enter)

Este comando será enviado por el Cliente al Servidor, será recibido por el Servidor, el Servidor obtendrá la información de la cuenta del tarjetahabiente y esa información será enviada como respuesta al Cliente. En el Cliente se desplegará la información del saldo con el que cuenta en este momento.

Después de recibir la consulta, el cliente podrá hacer uno de los siguientes comandos:

“DEPOSITAR”(espacio)cantidad_a_depositar(enter)

Se debe sumar la cantidad a depositar con el saldo actual y el resultado debe asignarse al saldo, para actualizarlo. El servidor debe enviar un mensaje de éxito o fallo al cliente.

“RETIRAR”(espacio)cantidad_a_retirar(enter)

Se debe restar la cantidad a retirar del saldo actual y el resultado debe asignarse al saldo. El servidor debe enviar un mensaje de éxito o fallo al cliente.

Nuestro proyecto fue realizado en Python haciendo uso de sockets para la comunicación cliente-servidor y la librería crypto para poder cifrar los mensajes del cliente y el servidor.

Para instalar la librería se usa el siguiente comando: pip install pycryptodome

Con eso ya se podrá correr el programa

Tambien usamos un archivo para guardar la información del tarjetahabiente, los 2 archivos (el cliente y el servidor) deben estar en la misma carpeta que el archivo.txt para poder acceder a ella.

Ahora los programas se ejecutaran siguiendo la siguiente línea

Para el cliente:

```
C:\Users\Ali Hcs\Desktop\FI\Proyecto Cliente-Servidor>python cliente-cifrado.py localhost 8000
```

Para el servidor:

```
C:\Users\Ali Hcs\Desktop\FI\Proyecto Cliente-Servidor>python servidor-cifrado.py 8000
```

Esto para que tengan la misma ip y el mismo puerto de conexión.

Ya con esto veremos la siguiente sección que será capturas del código funcionando.

Capturas del programa en funcionamiento

Servidor iniciado

```
C:\Users\Ali Hcs\Desktop\FI\Proyecto Cliente-Servidor>python servidor-cifrado.py 8000
Servidor Iniciado
-----CONEXIÓN ESTABLECIDA----- ............
```

Cliente iniciado

```
C:\Users\Ali Hcs\Desktop\FI\Proyecto Cliente-Servidor>python cliente-cifrado.py localhost 8000
Cliente iniciado
-SERVIDOR:
-----MENU-----
--CONSULTA
--DEPOSITAR (monto)
--RETIRO (monto)
--SALIR
------------

ENTRADA:
```

Como podemos ver, el servidor le manda el menú al cliente y le pide una respuesta.

```
ENTRADA: Consulta
-SERVIDOR:   Gustavo Ali GÃ³mez Trejo
1111111111111111
234444.0
```

En este caso probamos consulta, abre el archivo y nos manda la información, y podemos ver la respuesta del servidor, en la cual vemos que recibió la opción de consulta.

```
C:\Users\Ali Hcs\Desktop\FI\Proyecto Cliente-Servidor>python servidor-cifrado.py 8000
Servidor Iniciado
-----CONEXIÓN ESTABLECIDA----- ............
Se recibio la opcion:  CONSULTA
```

Ahora probaremos hacer un deposito, para esto tecleamos la opción seguido de la cantidad a depositar

```
ENTRADA: deposito 2222
-SERVIDOR:
-----MENU-----
--CONSULTA
--DEPOSITAR (monto)
--RETIRO (monto)
--SALIR
-------------

ENTRADA:
```

```
Se recibio la opcion:  DEPOSITO 2222

Saldo anterior:  234444.0
Saldo nuevo:  236666.0
```

Con esto podemos ver los resultados tando del cliente como del servidor

Consultamos otra vez y vemos el cambio.

```
ENTRADA: consulta
-SERVIDOR:   Gustavo Ali GÃ³mez Trejo
1111111111111111
236666.0
```

Por ultimo vemos la opción de retiro.

```
ENTRADA: retiro 2222
-SERVIDOR:
-----MENU-----
--CONSULTA
--DEPOSITAR (monto)
--RETIRO (monto)
--SALIR
-------------

ENTRADA:
```

```
Se recibio la opcion:  RETIRO 2222
Saldo anterior:  236666.0
Saldo nuevo:  234444.0
```

Y observamos que se ha realizo los cambios

Por ultimo hacemo una consulta mas para poder ver el movimiento y veremos lo siguiente.

```
ENTRADA: consulta
-SERVIDOR:   Gustavo Ali GÃ³mez Trejo
1111111111111111
234444.0

ENTRADA:
```

Capturas de los Códigos comentados

Código Cliente-Cifrado

```python
# Librerias que utilizaremos
import socket
import sys
import Crypto
from Crypto.PublicKey import RSA
import binascii
from Crypto.Cipher import PKCS1_OAEP
import time

# Leemos desde línea de comandos el puerto y la IP
IPServidor = sys.argv[1]
puertoServidor = int(sys.argv[2])


# Clave pública y privada
randomNum = Crypto.Random.new().read
privateKey = RSA.generate(1024, randomNum)
publicKey = privateKey.publickey()

# Exportando key publica y privada
privateKey = privateKey.exportKey(format='DER')
publicKey = publicKey.exportKey(format='DER')

# Decodificando a ascii
privateKey = binascii.hexlify(privateKey).decode('utf8')
publicKey = binascii.hexlify(publicKey).decode('utf8')
#print(publicKey)

# Codificando solo la llave privada
privateKey = RSA.importKey(binascii.unhexlify(privateKey))

# Creamos una clave de decifrado
cipher = PKCS1_OAEP.new(privateKey)
```

```python
# Iniciamos el socket
socketCliente = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socketCliente.connect((IPServidor, puertoServidor))
print("Cliente iniciado")
contador = 2

# Mandamos la llave pública al Servidor
socketCliente.send(publicKey.encode(encoding = "ascii",errors = "ignore"))

flag = True
try:
    flag = True
    while flag:
        recibidoCipher = socketCliente.recv(1024) # Recibimos el mensaje cifrado
        recibido = cipher.decrypt(recibidoCipher) # Decifamos el mensaje con cipher
        print("-SERVIDOR:  ",recibido.decode()) # Mensajes del servidor
        enviar = input("\nENTRADA: ").upper().strip() # Seleccionamos la opción desde teclado
        socketCliente.send(enviar.encode(encoding = "ascii",errors = "ignore")) # Enviamos la opción elegida

        # Terminamos conexión
        if enviar == "SALIR":
            print("CONEXIÓN FINALIZADA")
            flag = False
except ValueError:
    print("Espera un momento ")

socketCliente.close()
```

Código Servidor-Cifrado

```python
# Librerias que utilizaremos
import socket
import sys
from io import open
import Crypto
from Crypto.PublicKey import RSA
import binascii
from Crypto.Cipher import PKCS1_OAEP
import time
import itertools
import threading

done = False
# Animación de espera
def animate():
    for c in itertools.cycle(['.   ','..  ','... ','....']):
        if done:
            break
        sys.stdout.write('\rESPERANDO CONEXIÓN DEL CLIENTE '+c)
        sys.stdout.flush()
        time.sleep(0.1)
    sys.stdout.write('\r-----CONEXIÓN ESTABLECIDA-----\n')

direccionServidor = "localhost"
puertoServidor = int(sys.argv[1])
# Conección con los sockets
socketServidor = socket.socket(socket.AF_INET,socket.SOCK_STREAM);
socketServidor.bind( (direccionServidor, puertoServidor) )
socketServidor.listen()
print("Servidor Iniciado") # Mensaje que se inicio el servidor

# La animación se ejecuta hasta que un cliente se conecte
t = threading.Thread(target=animate)
t.start()
socketConexion, addr = socketServidor.accept()
done = True

# Reciviendo la llave publica del cliente
key = socketConexion.recv(1024).decode(encoding = "ascii",errors = "ignore")

# Codificando solo la llave publica
publicKey  = RSA.importKey(binascii.unhexlify(key))
cipher = PKCS1_OAEP.new(publicKey)

flag = True
#Bucle infinito
while flag:
    # Mostramos el menu disponible
    menu = """\n-----MENU-----
```

```
--CONSULTA
--DEPOSITAR (monto)
--RETIRO (monto)
--SALIR
----------"""
    # Ciframos y codificamos el mensaje antes de enviar
    menu = menu.encode()
    menuCipher = cipher.encrypt(menu)

    socketConexion.send(menuCipher) # Enviamos el menu al cliente de forma encode
    recibido = socketConexion.recv(1024)  # Esperamos la respuesta del Cliente
    opcion = recibido.decode(encoding = "ascii", errors = "ignore") # Pasamos la respuesta del cliente a cadena con decode
    print("Se recibio la opcion: ",opcion) # Mostramos que opcion escogio el cliente
    select = opcion.split(" ")

    if  select[0] == "CONSULTA" : # Si la opción que mando el cliente es CONSULTA entra en el if
        archivo = open("cuenta.txt", "r") # Abrimos el archivo como lectura (read)
        consulta = archivo.read() # Guardamos el archivo en la variable consulta
        consultaCadena = "".join(consulta) # Pasamos a cadena lo leido del archivo

        # Ciframos y codificamos el mensaje antes de enviar
        consultaCadena = consultaCadena.encode()
        consultaCipher = cipher.encrypt(consultaCadena)

        socketConexion.send(consultaCipher)  # Enviamos la cadena al cliente
        archivo.close() # Cerramos el archivo

    elif  select[0] == "DEPOSITAR": # Si la opcion que llego es deposito entra al if
        cantidad = int(select[1])

        archivo = open("cuenta.txt", 'r+' ) # Abro el archivo como lectura y escritura
        deposito = archivo.readlines() # Lee todas las lineas y las guarda como lista en deposito
        dinero = deposito[2] # Guardo la cantidad que tiene el archivo
        print("Saldo anterior: ",dinero)
        dineroFloat = float(dinero) # Paso la cantidad a flotante y la guardo en su variable
        dineroFloat = dineroFloat + cantidad # Realizo la suma  correspondiente al deposito
        dineroStr = str(dineroFloat) # Paso el monto final a string para poder escribir en el archivo

        deposito[2] = dineroStr # Escribo en la lista que remplazara al antiguo saldo
        print("Saldo nuevo: ",deposito[2])
        archivo.seek(0) # Se posiciona al principio de la linea
        archivo.writelines(deposito) # Escribe todo el texto de nuevo ya con las modificaciones
        archivo.close() # Cierra el archivo

        # Enviamos el saldo actual al Cliente
        #actual = "Su saldo nuevo es de: $"+dineroStr
        #actual = actual.encode()
        #actualCipher = cipher.encrypt(actual)
```

```
98          #socketConexion.send(actualCipher)
99
100     elif  select[0] == "RETIRO": # Si la opcion que llego es deposito entra al if
101         cantidad = int(select[1])
102
103         archivo = open("cuenta.txt", 'r+' ) # Abro el archivo como lectura y escritura
104         deposito = archivo.readlines() # Lee todas las lineas y las guarda como lista en deposito
105         dinero = deposito[2] # Guardo la cantidad que tiene el archivo
106         print("Saldo anterior: ",dinero)
107         dineroFloat = float(dinero) # Paso la cantidad a flotante y la guardo en su variable
108         dineroFloat = dineroFloat - cantidad # Realizo la suma  correspondiente al deposito
109         dineroStr = str(dineroFloat) # Paso el monto final a string para poder escribir en el archivo
110
111         deposito[2] = dineroStr # Escribo en la lista que remplazara al antiguo saldo
112         print("Saldo nuevo: ",deposito[2])
113         archivo.seek(0) # Se posiciona al principio de la linea
114         archivo.writelines(deposito) # Escribe todo el texto de nuevo ya con las modificaciones
115         archivo.close() # Cierra el archivo
116
117         # Enviamos el saldo actual al Cliente
118         #actual = "Su saldo nuevo es de: $"+dineroStr
119         #actual = actual.encode()
120         #actualCipher = cipher.encrypt(actual)
121         #socketConexion.send(actualCipher)
122
123     # Se finaliza la conexión
124     elif select[0] == "SALIR":
125         print("CONEXIÓN FINALIZADA")
126         flag = False
127
128     # En caso de que el usuario no ingrese una opcion válida
129     else:
130         error = "ERROR: Ingrese una opcion valida."
131         error = error.encode()
132         errorCipher = cipher.encrypt(error)
133         socketConexion.send(errorCipher)
134
135 socketConexion.close()
```