UNIVERSIDAD
Panamericana

Entrega 1: “Whatsapp” seguro

Criptografía y seguridad en redes

Profesor: Carlos Pérez Leguízamo

Esteban Viniegra Pérez Olagaray

ID: 0235320

Fecha de entrega: 18 de septiembre del 2023

Introducción

La comunicación es una parte esencial de la vida humana, y cada vez más personas utilizan medios digitales para enviar y recibir mensajes. Sin embargo, estos medios pueden ser vulnerables a ataques informáticos que pongan en riesgo la privacidad y la integridad de los usuarios. Por eso, es importante contar con sistemas de mensajería segura que garanticen la confidencialidad, la autenticación y la no repudiación de los mensajes.

En este documento se presenta el desarrollo de una aplicación de mensajería segura, que utiliza técnicas de criptografía y seguridad en redes para proteger los datos. Esta es la primera entrega del proyecto, donde se implementó el cliente, el servidor y el middleware (MOM) que se encarga de distribuir la información entre los usuarios.

Desarrollo

El proyecto se divide en 2 partes:

- **Cliente:**
  El cliente es la interfaz del usuario, donde puede escribir y recibir mensajes. Se utilizó el lenguaje de programación Java y la librería JavaFX para crear una ventana gráfica con un área de texto y un botón para enviar mensajes, y una caja vertical donde se muestran los mensajes recibidos.

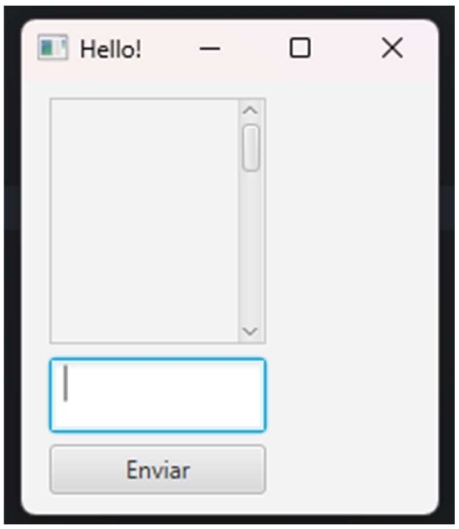


Figura 1: Interfaz gráfica del ciente

El cliente se comunica con el servidor mediante sockets TCP, usando las clases Socket, DataInputStream y DataOutputStream. El cliente envía al servidor el mensaje escrito en el área de texto, y recibe del servidor los mensajes enviados por otros usuarios.

Para mostrar los mensajes recibidos en la caja vertical, se utilizó la clase Platform.runLater, que permite ejecutar una tarea en el hilo de la interfaz gráfica. Además, se aplicó un estilo diferente a los mensajes recibidos para distinguirlos de los propios.

```
49      public void initialize() {
50
51
52          Thread socketThread = new Thread(() -> {
53              try {
54                  socket = new Socket( host: "127.0.0.1",  port: 12345);
55                  entrada = new DataInputStream(new BufferedInputStream(socket.getInputStream()));
56                  salida = new DataOutputStream(socket.getOutputStream());
57
58                  try {
59                      while (true) {
60                          String mensajeRecibido = entrada.readUTF();
61
62                          Platform.runLater(() -> {
63                              Label label = new Label(mensajeRecibido);
64                              label.setStyle("-fx-font-weight: bold;");
65                              vbox.getChildren().add(label);
66                          });
67                      }
68                  } catch (IOException error) {
69                      System.out.println("Error al recibir mensaje: " + error.getMessage());
70                  }
71              } catch (IOException error) {
72                  System.out.println(error);
73              }
74          });
75          socketThread.setDaemon(true);
76          socketThread.start();
77
78
79
80      }
```

Figura 2: Se inicializa la comunicación entre el cliente y el servidor, creando un nuevo hilo y comunicando hacia la ip y el puerto indicado para mandar y recibir mensajes.

- **Servidor y middelware (MOM)**
  El servidor y el middleware (MOM) son los componentes que se encargan de distribuir la información entre los clientes. El servidor recibe los mensajes de los

clientes mediante sockets TCP, y los envía al MOM. El MOM almacena los mensajes en una cola y los distribuye a todos los clientes conectados.

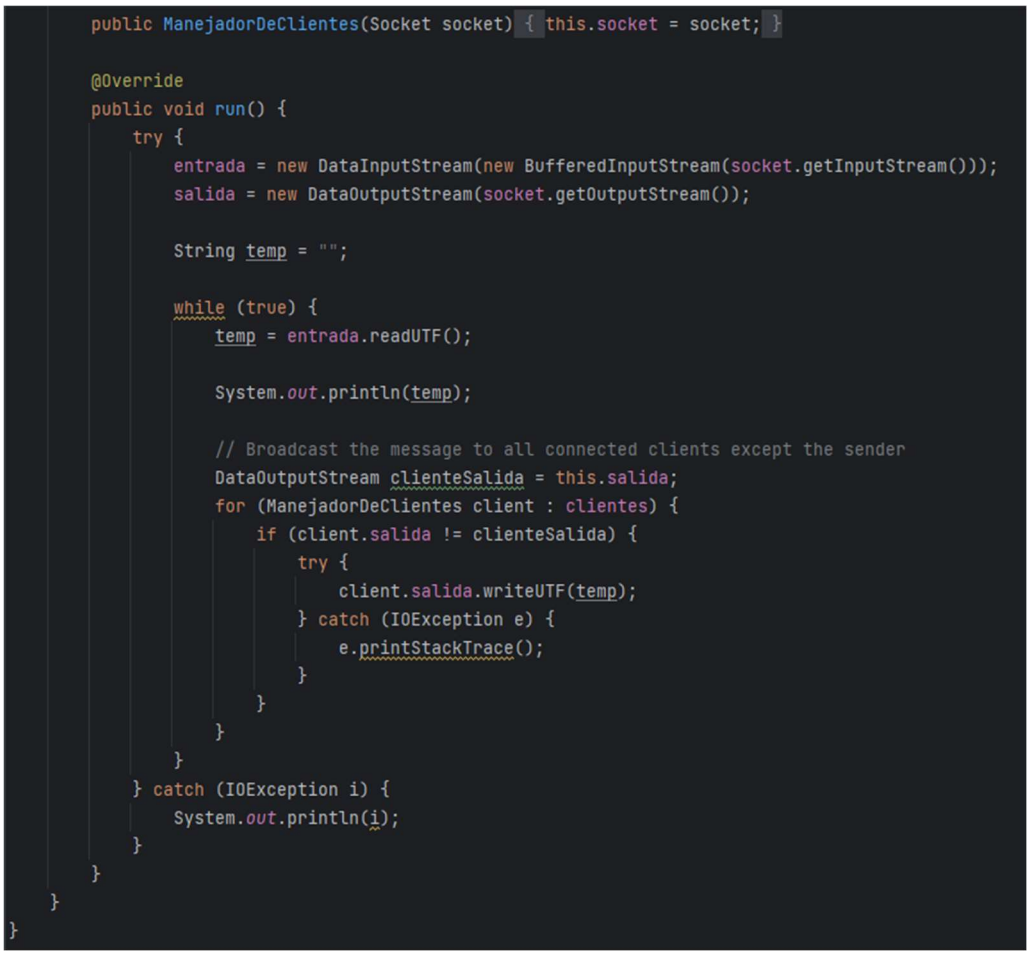
El servidor y el MOM se implementaron en Java, usando las clases ServerSocket, Socket, DataInputStream, DataOutputStream y ArrayList. El servidor crea un objeto de la clase MOM, que tiene un método para agregar clientes a una lista y otro método para enviar mensajes a todos los clientes de la lista.

El servidor acepta las conexiones de los clientes y crea un objeto de la clase ManejadorDeClientes, que implementa la interfaz Runnable. El servidor pasa el socket del cliente y el objeto del MOM al constructor del manejador. El manejador crea los flujos de entrada y salida para comunicarse con el cliente, y se registra en el MOM.

El manejador se ejecuta en un hilo independiente, y tiene un bucle infinito donde lee los mensajes del cliente y los envía al MOM. El MOM, a su vez, envía el mensaje a todos los clientes registrados, incluyendo al emisor.

```java
public class MOM {
    2 usages
    ServerSocket servidor;
    3 usages
    List<ManejadorDeClientes> clientes;

    1 usage
    public MOM(int port) {
        clientes = new ArrayList<>();

        try {
            servidor = new ServerSocket(port);
            System.out.println("Servidor Corriendo");
            while (true) {
                Socket socket = servidor.accept();
                System.out.println("Nuevo Cliente aceptado");

                ManejadorDeClientes clientHandler = new ManejadorDeClientes(socket);
                clientes.add(clientHandler);

                Thread thread = new Thread(clientHandler);
                thread.start();
            }
        } catch (IOException i) {
            System.out.println(i);
        }
    }

```

Figura 3: Inicialización de socket para el servidor y creación de hilos y sockets para cada cliente, para poder ser manejados de manera simultánea



```
    public ManejadorDeClientes(Socket socket) { this.socket = socket; }

    @Override
    public void run() {
        try {
            entrada = new DataInputStream(new BufferedInputStream(socket.getInputStream()));
            salida = new DataOutputStream(socket.getOutputStream());

            String temp = "";

            while (true) {
                temp = entrada.readUTF();

                System.out.println(temp);

                // Broadcast the message to all connected clients except the sender
                DataOutputStream clienteSalida = this.salida;
                for (ManejadorDeClientes client : clientes) {
                    if (client.salida != clienteSalida) {
                        try {
                            client.salida.writeUTF(temp);
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }
            }
        } catch (IOException i) {
            System.out.println(i);
        }
    }
}
```

Figura 4: Distribución del mensaje del cliente que mandó el mensaje a los demás clientes conectados al servidor

Concusión

En esta primera entrega del proyecto de criptografía y seguridad en redes, se desarrolló una aplicación de mensajería segura, que utiliza sockets TCP para comunicarse entre el cliente y el servidor, y un middleware (MOM) para distribuir la información entre los clientes. Se utilizó el lenguaje de programación Java y la librería JavaFX para crear la interfaz gráfica del cliente, y se implementó la lógica del servidor y el MOM en clases separadas.

En las siguientes entregas, se añadirán las funcionalidades de criptografía y seguridad, como el cifrado y descifrado de los mensajes, la generación y verificación de firmas digitales, la autenticación de los usuarios y el manejo de errores.