Santiago Vela – 202026767

Verónica Escobar – 201922197

Caso 3: Canales Seguros

2022-1

Contenido

[1. Introducción 1](#_Toc103039021)

[2. Descripción de la organización de los archivos 2](#_Toc103039022)

[3. Descripción para correr el prototipo 3](#_Toc103039023)

[4. Descripción esquema generación de llaves 3](#_Toc103039024)

[5. Escenarios Tiempo ejecución 4](#_Toc103039025)

[6. Tabla de datos recompilados 4](#_Toc103039026)

[7. Grafica simétrica 4](#_Toc103039027)

[8. Grafica asimétrica 4](#_Toc103039028)

[9. Comentarios Graficas y comportamientos 4](#_Toc103039029)

[10. Cálculos 4](#_Toc103039030)

[11. Conclusiones 5](#_Toc103039031)

# Introducción

El objetivo de este caso es entender, profundizar y poner a prueba la seguridad informática en el ámbito real, como lo es en una empresa transportadora de paquetes, en donde se tiene que mantener una confidencialidad e integridad todo el tiempo, y en donde tanto el cliente como la empresa (servidor) esperan poder tener una comunicación segura y sin intermediarios. Para esto utilizaremos los diferentes tipos de algoritmos de encriptación de datos, como los algoritmos simétricos y asimétricos, y analizaremos en su comportamiento en relación a la demanda del negocio y decidiremos que algoritmo es mejor usar según cada caso posible del negocio.



# Descripción de la organización de los archivos

Para este caso esta divido en dos partes, el primero es un servidor y cliente concurrente y en el segundo es un cliente y un servidor iterativo.

Para el caso del servidor y cliente concurrente debido a que es concurrente se pueden crear uno o más flujos de comunicación entre un cliente y un servidor es por eso que inicialmente tenemos 5 paquetes en donde en cada uno de ellos están las clases que nos ayudaran a que sea posible esta concurrencia.

Primeramente , tenemos la carpeta cliente en donde tendremos todos los archivos que son necesarios para la comunicación entre el cliente y un servidor. La primera clase es “Client” el cual nos ayudara a la creación y unos de los diferentes clientes con sus canales de comunicación con el servidor(sockets), esta clase crea la cantidad de clientes según el usuario le indique, para cada cliente se le genera su llave publica individual la cual posteriormente será utilizada por el servidor. De igual forma, con la clase “Client Thread” tenemos los métodos necesarios para comunicación efectiva entre el cliente y el servidor ,debido a que esta clase se va a ejecutar una o muchas veces concurrentemente según del flujo esperara o se comunicara con el servidor con tal de tener confidencialidad. Final mente tenemos la clase “Cliente main” en la cual hace llamado a la clase cliente en donde esta se implementa de manera mas organizada.

De la misma forma, para el caso servidor cliente concurrente, tenemos la carpeta récords en la cual nos ayudara a llevar el estado de los paquetes, para este caso tenemos 32 paquetes y 32 usuarios esta clase le permitirá al servidor consultar el estado actual de los paquetes en el momento en el que el cliente desea saber el estado. Esta. Carpeta se compone principalmente de dos clases , y un archivo csv, la primera clase llamada “record” nos ayuda a saber el log de un paquete, este log contiene: el usuario, el ID del paquete y el estatus del paquete. Por otro lado la otra clase que tenemos es “RecordList” la cual nos ayuda a leer el archivo csv y a crear una lista de los log o estados de los paquetes en tiempo real y poder almacenarlos.

Igualmente , para el caso servidor cliente concurrente tenemos la carpeta Security Utils, que le va hacer de gran ayuda tanto al cliente como al servidor en el momento de mantener confidencialidad entre ellos, por un lado tenemos la clase llamada “Key Generators ” la cual nos ayudara a crear llaves privadas y llaves asimétricas. Lo mismo pasa con la clase “Hashing and autocode” el cual le permitirán interactuar al cliente con el servidor en instantes finales como cuando se calcula el digest.

Para finalizar esta segunda parte nos encontramos con dos carpetas más, una que es Server la cual contiene tres clases que al igual que cliente ayuda en asegurar una buena conexión entre el servidor y el cliente, y la otra carpeta que encontramos se llama “Status Request” la cual es usada inicialmente por el cliente una vez los pasos de autenticación ya fueron cumplidos, esta carpeta tiene una clase la cual le permitirá al cliente enviar un request al servidor de su paquete especificando el Id del cliente y el Id del paquete.

Para el caso del servidor y cliente iterativo, se manejan las mismas carpetas sin embargo al ser iterativo no se crearan flujos de o thread si no que una vez acabe un proceso o una ejecución se puede ejecutar la otra, es por eso que tanto para el server que como para el client no vamos a tener clases que extiendan de un thread, con esto dicho la ejecución en la segunda parte será más corta, en términos de tener mas sockets o conexiones entre clientes y servidores.

# Descripción para correr el prototipo

Las instrucciones para correr el prototipo son muy simples debido a que primeramente siempre tenemos que correr la clase main del server y posteriormente tendremos que correr la clase main del cliente, esto sirve tanto para iterativo como para concurrente.

# Descripción esquema generación de llaves

Para haber podido implementar la seguridad y confidencialidad entre el cliente y el servidor, utilizamos el paquete de java “ java.security”, el cual consiste básicamente en clases abstractas e interfaces que encapsulan conceptos de seguridad como certificados, claves, resúmenes de mensajes y firmas digitales.

Los proveedores pueden implementar tres clases:

* **KeyPairGenerator**. Se emplea para crear claves públicas y privadas.
* **MessageDigest**. Prorciona la funcionalidad de algoritmos de resumen de mensajes como el MD5 y el SHA.
* **Signature**. Se emplea para el firmado digital de mensajes.

En nuestro caso utilizamos este paquete para crear tanto llaves publicas como privadas tanto del cliente como del servidor. Por otro lado usamos el paquete “javax.crypto”, el cual nos ayudo a mantener confidencialidad en las partes ya que gracias a este paquete pudimos realizar operaciones de encriptación, este paquete lo pudimos usara tanto para cifrado simétrico, como para cifrado asimétrico; para este caso usamos “KeyGenerator” Debido a que proporciona las funciones de un generador de claves simétricas.

# Escenarios Tiempo ejecución



# Tabla de datos recompilados

# Grafica simétrica

# Grafica asimétrica

# Comentarios Graficas y comportamientos

# Cálculos

# Conclusiones