Caso de Estudio 3 – Canales Seguros

Sistema de rastreo de paquetes en una compañía transportadora

Laura Valentina Guiza Melo – 201920926

Diego Alejandro Mahecha Forero – 201922589

Universidad de los Andes

Ingeniería de Sistemas y Computación

ISIS 2203 Infraestructura Computacional

Semestre 2025-10

Contenido

DESCRIPCIÓN DE LA ORGANIZACIÓN DE LOS ARCHIVOS EN EL ZIP

El proyecto está organizado por subdirectorios. En primer lugar, el subdirectorio .idea, generado automáticamente, contiene los archivos encodings.xml, misc.xml y vcs.xml, que garantizan la coherencia en la codificación de caracteres, almacenan la configuración global del proyecto y definen la integración con el sistema de control de versiones.

En el subdirectorio docs, en donde se encuentra este informe, junto con el archivo Excel con todos los datos experimentales recopilados y el documento 20251-aerolinea-1.pdf, que describe el enunciado del caso de estudio y sirve de referencia para los objetivos y requisitos de la implementación.

El subdirectorio src concentra todo el código fuente bajo src/main/java/org/caso3/, organizado en tres paquetes fundamentales. El paquete cliente incluye Cliente.java, que implementa una conexión secuencial; ClienteConcurrente.java, encargado de lanzar múltiples instancias de cliente según un parámetro de concurrencia; y ClienteHilo.java, que representa cada hilo de cliente midiendo su latencia individual. El paquete servidor agrupa ServidorPrincipal.java, que pone en escucha el puerto 5000 y genera delegados para cada petición, y Delegado.java, responsable del protocolo seguro (Diffie–Hellman, AES/CBC, HMAC/SHA256, firma digital) y de la medición de tiempos de atención. Finalmente, en el paquete seguridad se encuentran GeneradorRSA.java, para la generación de pares de llaves RSA y su persistencia en archivos DER; UtilCifrado.java, con utilidades de cifrado simétrico y asimétrico, HMAC y firma digital; y UtilDH.java, que gestiona parámetros y acuerdos Diffie–Hellman para establecer la llave maestra.

Finalmente, al ejecutar, se crea el subdirectorio target, donde residen los archivos compilados en target/classes. En la raíz del ZIP también se incluye el archivo de gestión pom.xml, responsable de definir dependencias, versiones de Java y fases de construcción.

INSTRUCCIONES PARA CORRER SERVIDOR Y CLIENTE

Antes de iniciar cualquier componente, es necesario generar el par de llaves RSA que el servidor empleará para firmar respuestas y parcial descifrar intercambios del protocolo. Esto se logra ejecutando la clase GeneradorRSA, la cual produce en la raíz del proyecto los archivos llave_privada.der y llave_publica.der, listos para su carga por ServidorPrincipal.

Con las llaves disponibles, se compila todo el proyecto con Maven, lo que situará las clases resultantes en target/classes. Para poner en marcha el servicio, basta con ejecutar la clase ServidorPrincipal: ésta leerá internamente las llaves RSA, abrirá el puerto 5000 y esperará conexiones entrantes. Por cada solicitud, creará un hilo delegado que gestiona el establecimiento del canal seguro (Diffie–Hellman, cifrado AES/CBC, HMAC-SHA256 y firma digital), procesa la petición del cliente, mide el tiempo de atención y devuelve la respuesta cifrada.

Para realizar una consulta única, se levanta la clase Cliente, que inicia una conexión, envía la petición y despliega la respuesta recibida antes de cerrar el socket. Cuando se desea evaluar el rendimiento bajo concurrencia, se utiliza ClienteConcurrente, a la cual se puede pasar como parámetro el número de clientes simultáneos; en ausencia de valor explícito se lanzan 32 hilos por defecto. Cada hilo, instanciado mediante ClienteHilo, mide en milisegundos su tiempo de ida y vuelta y muestra en consola su identificador, la respuesta y la latencia correspondiente.

Esta separación clara entre el servidor, la lógica de delegado y las variantes de cliente (secuencial y concurrente) permite realizar pruebas funcionales y de rendimiento de forma sencilla y reproducible.

TAREAS Y PREGUNTAS PLANTEADAS

1. **Tiempos del servidor para firmar, cifrar la tabla y verificar la consulta.**

   Escenario 1: Un servidor de consulta y un cliente iterativo.

   En donde el cliente debe generar 32 consultas secuenciales.

   Escenario 2: Servidor y clientes concurrentes.

   En donde el número de delegados, tanto servidores como clientes, debe variar entre 4, 16, 32 y 64 delegados concurrentes. Cada servidor delegado atiende un solo cliente y cada cliente genera una sola solicitud.

2. **Tabla con los datos recopilados.**

   Tenga en cuenta que necesitará correr cada escenario en más de una ocasión para validar los resultados.

3. **Comparación del tiempo que el servidor requiere para cifrar la respuesta con cifrado simétrico y con cifrado asimétrico (con su llave pública).**

   Observe que el cifrado asimétrico de la respuesta no se usa en el protocolo, solo se calculará para posteriormente comparar los tiempos.

4. Graficas:

- **Tiempos para firmar en los escenarios**
- **Tiempos para cifrar la tabla en los escenarios**
- **Tiempos para verificar la consulta en los escenarios**
- **Tiempos caso simétrico y caso asimétrico en los diferentes escenarios.**

5. Comentarios sobre las gráficas

explicando los comportamientos observados.

6. Escenario que permita estimar la velocidad del procesador, y estimación de cuántas operaciones de cifrado puede realizar la máquina por segundo

(en el caso evaluado de cifrado simétrico y cifrado asimétrico). Escriba todos sus cálculos.