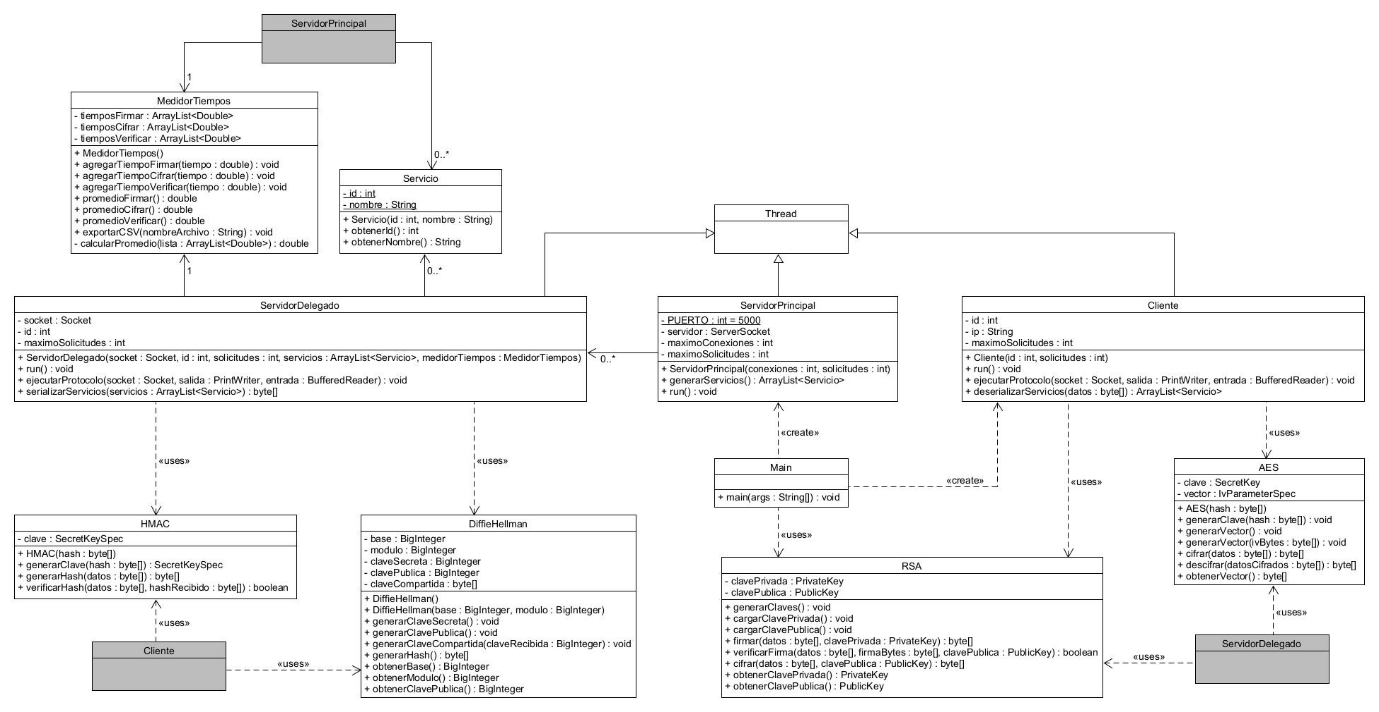
**Documentación Caso 3**

1. **Diagrama de Clases**

Figura 1. Diagrama de Clases

1. **Funcionamiento**

El proyecto inicia su ejecución en la clase Main, donde se genera un par de llaves utilizando el protocolo RSA. Estas llaves, una pública y una privada, son almacenadas en la carpeta claves/ y son utilizadas para garantizar la autenticidad de las comunicaciones mediante la firma digital de mensajes. A continuación, Main crea una instancia del ServidorPrincipal, que abre un servidor en el puerto 5000, carga la lista de servicios disponibles y queda en espera de conexiones entrantes. Paralelamente, Main crea varios hilos de Cliente, los cuales intentan establecer conexión con el ServidorPrincipal a través de la red local. Cuando un cliente establece la conexión, el ServidorPrincipal delega su atención creando un nuevo hilo ServidorDelegado, que manejará exclusivamente la comunicación con ese cliente. Cada ServidorDelegado opera de forma independiente para permitir atención concurrente a múltiples clientes.

Una vez conectados, comienza el protocolo de comunicación seguro. El cliente carga la clave pública RSA del servidor, mientras que el servidor delegado carga su clave privada RSA. El proceso inicia con el envío de un reto aleatorio por parte del cliente hacia el servidor. Este reto es firmado por el servidor utilizando su clave privada y devuelto al cliente, quien verifica la firma con la clave pública, asegurando así que la identidad del servidor es auténtica. Con la autenticidad confirmada, se procede al intercambio de claves mediante el protocolo Diffie-Hellman. El servidor delegado genera su propia clave pública junto con los parámetros de base y módulo, los cuales son enviados al cliente firmados digitalmente. El cliente utiliza la base y el módulo proporcionados por el servidor para generar su propia clave pública Diffie-Hellman, que luego es enviada al servidor. De esta forma, ambos extremos calculan de manera independiente una clave compartida secreta. A partir de esta clave compartida, tanto el cliente como el servidor derivan dos claves internas: una clave secreta utilizada para el cifrado de mensajes mediante AES, y una clave de autenticación utilizada para generar y verificar firmas de integridad mediante HMAC. De esta manera, toda la comunicación posterior está protegida tanto en confidencialidad como en integridad. Para establecer la comunicación cifrada, el cliente genera un vector de inicialización, el cual envía al servidor delegado. El servidor utiliza directamente este vector recibido para configurar su cifrado AES, asegurando que ambas partes trabajen con los mismos parámetros y puedan cifrar y descifrar la información correctamente.

Una vez establecido el canal seguro, el servidor delegado cifra la lista de servicios disponibles utilizando AES y genera una firma de integridad utilizando HMAC sobre los datos cifrados. Tanto la lista cifrada como su firma son enviadas al cliente. El cliente descifra la lista y verifica su autenticidad antes de seleccionar un servicio. El cliente selecciona un servicio (o simula un servicio inexistente para pruebas) y envía su selección cifrada y firmada de vuelta al servidor delegado, junto con su dirección IP cifrada y firmada también. El servidor delegado descifra la información recibida, verifica su integridad y responde enviando la IP y el puerto del servidor correspondiente al servicio solicitado, nuevamente cifrados y firmados.

Finalmente, tras procesar todas las solicitudes configuradas, el ServidorDelegado cierra su conexión con el cliente. Una vez que todos los clientes han sido atendidos, el ServidorPrincipal cierra el servidor y exporta un reporte en formato CSV que contiene los tiempos promedio de firma, cifrado y verificación, recopilados durante las interacciones.

1. **Organización del Proyecto**

El proyecto está organizado de la siguiente manera:

* SRC: Contiene todo el código fuente en Java. Aquí se encuentran las clases que implementan el cliente, el servidor principal, los servidores delegados, los protocolos de cifrado, firma y generación de claves, así como la medición de tiempos de procesamiento.
* Claves: Almacena las llaves RSA generadas automáticamente en la ejecución (clave\_privada.key y clave\_publica.key).
* Documentación: contiene el diagrama de clases en formato de imagen y el archivo pdf con la documentación del proyecto.
* Tiempos: Guarda los resultados de los tiempos de firma, cifrado y verificación en el archivo resultados.csv, el cual se sobrescribe en cada nueva ejecución. También se almacenan aquí el archivo de Excel con las tablas de datos y las gráficas generadas para el análisis.

1. **Instrucciones de Ejecución**

La ejecución del proyecto se realiza desde la clase Main, donde se presenta un menú con las siguientes opciones:

* Opción 1: Ejecuta el programa con un servidor principal que atiende las consultas y un solo cliente iterativo que realiza 32 solicitudes consecutivas. El servidor principal crea un único servidor delegado para atender al cliente.
* Opción 2: Ejecuta el programa con múltiples clientes y servidores concurrentes. Se puede seleccionar entre 4, 16, 32 o 64 clientes, donde cada cliente realiza una única solicitud. El servidor principal crea un servidor delegado independiente para cada cliente conectado.
* Opción 3: Compara el tiempo de cifrado entre cifrado simétrico (AES) y cifrado asimétrico (RSA). Esta opción no estaba contemplada en el enunciado inicial, pero fue implementada como una extensión para analizar la diferencia de rendimiento entre ambos tipos de cifrado, ya que el protocolo original solo consideraba el uso de clave compartida (simétrica) y no el desempeño del cifrado con clave pública.

Para ejecutar cualquiera de los casos, basta con seleccionar el número correspondiente en el menú que aparece al correr el programa. Todos los resultados de tiempos promedio (firmar, cifrar y verificar) se almacenan automáticamente en el archivo tiempos/resultados.csv.

1. **Recopilación de Datos**
2. **Tiempo que el servidor requiere para firmar**

**Tabla 1:** Tiempo promedio de firma en cada escenario

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operación | Firmar Prueba 1 | Firmar Prueba 2 | Firmar Prueba 3 | Firmar Prueba 4 | Firmar Prueba 5 | Firmar Prueba 6 | Promedio | Desviación Estándar |
| Opción 1 Promedio (ms) | 0.52 | 0.37 | 0.38 | 0.38 | 0.42 | 0.49 | 0.427 | 0.0638 |
| Opción 2, 4 Clientes, Promedio (ms) | 0.8 | 0.39 | 0.49 | 0.39 | 0.51 | 0.68 | 0.543 | 0.1646 |
| Opción 2, 16 Clientes, Promedio (ms) | 0.39 | 0.4 | 0.42 | 0.39 | 0.61 | 0.48 | 0.448 | 0.0861 |
| Opción 2, 32 Clientes, Promedio (ms) | 0.39 | 0.42 | 0.52 | 0.38 | 0.46 | 0.42 | 0.432 | 0.0515 |
| Opción 2, 64 Clientes, Promedio (ms) | 0.43 | 0.56 | 0.62 | 0.43 | 0.68 | 0.42 | 0.523 | 0.1125 |

1. **Tiempo que el servidor requiere para cifrar**

**Tabla 2:** Tiempo promedio de cifrado en cada escenario

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operación | Cifrar Prueba 1 | Cifrar Prueba 2 | Cifrar Prueba 3 | Cifrar Prueba 4 | Cifrar Prueba 5 | Cifrar Prueba 6 | Promedio | Desviación Estándar |
| Opción 1 Promedio (ms) | 0.05 | 0.1 | 0.07 | 0.06 | 0.06 | 0.14 | 0.0800 | 0.0341 |
| Opción 2, 4 Clientes, Promedio (ms) | 0.09 | 0.04 | 0.03 | 0.02 | 0.04 | 0.13 | 0.0583 | 0.0426 |
| Opción 2, 16 Clientes, Promedio (ms) | 0.03 | 0.03 | 0.04 | 0.03 | 0.02 | 0.08 | 0.0383 | 0.0214 |
| Opción 2, 32 Clientes, Promedio (ms) | 0.05 | 0.03 | 0.03 | 0.03 | 0.11 | 0.04 | 0.0483 | 0.0313 |
| Opción 2, 64 Clientes, Promedio (ms) | 0.11 | 0.07 | 0.07 | 0.03 | 0.07 | 0.06 | 0.0683 | 0.0256 |

1. **Tiempo que el servidor requiere para verificar**

**Tabla 3**: Tiempo promedio de verificación en cada escenario

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operación | Verificar Prueba 1 | Verificar Prueba 2 | Verificar Prueba 3 | Verificar Prueba 4 | Verificar Prueba 5 | Verificar Prueba 6 | Promedio | Desviación Estándar |
| Opción 1 Promedio (ms) | 0.04 | 0.02 | 0.01 | 0.01 | 0.01 | 0.03 | 0.0200 | 0.0126 |
| Opción 2, 4 Clientes, Promedio (ms) | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.0117 | 0.0041 |
| Opción 2, 16 Clientes, Promedio (ms) | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.0117 | 0.0041 |
| Opción 2, 32 Clientes, Promedio (ms) | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.03 | 0.0150 | 0.0084 |
| Opción 2, 64 Clientes, Promedio (ms) | 0.03 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 | 0.0150 | 0.0084 |

1. **Tiempo que se requiere para el cifrado simétrico y cifrado asimétrico**

**Tabla 4:** Tiempo de cifrado asimétrico vs simétrico

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operación | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 | Prueba 6 | Prueba 7 | Prueba 8 | Promedio | Desviación Estándar |
| Tiempo Asimétrico (ms) | 0.38 | 0.26 | 0.29 | 0.34 | 0.24 | 0.23 | 0.19 | 0.21 | 0.2675 | 0.0654 |
| Tiempo Simétrico (ms) | 0.23 | 0.19 | 0.23 | 0.27 | 0.22 | 0.13 | 0.26 | 0.13 | 0.2075 | 0.0537 |

1. **Análisis**

En primer lugar, se midieron los tiempos de ejecución del servidor sobre de tres operaciones dentro del protocolo de comunicación: firma, cifrado y verificación. Estas mediciones se hicieron a través de diferentes escenarios. El primero fue sobre un cliente realizando 32 consultas al servidor, y después se midieron al tener clientes concurrentes haciendo consultas, y se varió la cantidad de clientes concurrentes entre 4, 16, 32 y 64.

Para la primera operación relacionada con la firma de los datos, aunque en principio el tiempo de firma debería mantenerse estable debido a que cada operación implica un procesamiento fijo sobre mensajes pequeños, en los resultados obtenidos se observaron ligeros aumentos en escenarios de mayor concurrencia. En particular, el tiempo promedio fue más alto cuando hubo clientes concurrentes (como en el caso de 4 y 64 clientes). Por otro lado, cuando un único cliente realizó múltiples consultas secuenciales (opción 1), los tiempos se mantuvieron un poco más bajos y estables, debido a que se generan las consultas de manera secuencial y hay un uso más eficiente de la CPU y la memoria en ausencia de saturación.

Ahora bien, para los tiempos de cifrado, se esperaría que con AES el proceso sea más eficiente y mantenga tiempos bajos y estables, dado que operamos también sobre bloques pequeños de datos. En efecto, los resultados confirmaron esta expectativa en general, con tiempos promedio inferiores a 0.1 ms en todos los escenarios. Sin embargo, se observó que el mayor tiempo promedio ocurrió en la opción 1, donde un solo cliente realizó múltiples consultas secuenciales, alcanzando 0.080 ms, mientras que escenarios concurrentes como 4, 16 y 32 clientes mostraron tiempos incluso menores. Aunque hubo algo de variabilidad, especialmente en el cliente con 32 consultas, el desempeño general del cifrado se mantuvo eficiente.

Luego, con los tiempos de verificación, se obtuvieron igualmente resultados esperados con respecto a las verificaciones ya que HMAC tiene una baja complejidad del cálculo y es eficiente. En general, los tiempos promedio fueron muy bajos en todos los escenarios a comparación de los otros escenarios. Sin embargo, se observó que en la opción de 32 consultas hubo una mayor variabilidad y tiempos promedio más altos que en otros escenarios concurrentes. Esto podría deberse a que, al realizar muchas operaciones de verificación seguidas por parte de un solo cliente, se genera una carga acumulada en memoria o en el manejo de conexiones, afectando ligeramente la consistencia en los tiempos de respuesta. Aun así, todos los valores de verificación se mantuvieron dentro de rangos de milisegundos muy pequeños.

Finalmente, al comparar los tiempos de cifrado asimétrico y simétrico, se observó que el cifrado RSA es más complejo y por lo tanto más lento que el cifrado AES, como era de esperarse. El cifrado asimétrico presentó un mayor tiempo promedio y una desviación estándar más alta, indicando mayor variabilidad. Esto muestra consistencia con el hecho de que RSA involucra operaciones matemáticas más costosas que AES, que está optimizado para procesamiento rápido de bloques. De esta manera, en términos de complejidad computacional, los resultados confirmaron que el cifrado simétrico resulta menos costoso que el asimétrico, tal como se reflejó en los tiempos de ejecución medidos.

1. **Escenario**

Un escenario para poder estimar la velocidad del procesador teniendo en cuenta lo realizado anteriormente, sería realizar múltiples operaciones de cifrado consecutivas sobre un bloque de datos de tamaño fijo y medir el tiempo total de ejecución. Para esto, lo haremos usando un bloque de 90 bytes y repetiremos esto 2000 veces, para los cifrados simétrico y asimétrico. En el caso del asimétrico, se generó un par de claves de 1024 bits y se cifró el mismo bloque de datos utilizando la clave pública. Para el simétrico, se derivó una clave secreta mediante Diffie-Hellman y se realizó el cifrado en modo CBC utilizando un vector de inicialización aleatorio. Luego de que ejecutamos estas operaciones, se midió el tiempo total tomado en cada caso.

Ahora bien, en el caso asimétrico, el tiempo total registrado para las 2000 operaciones fue de 55.7803 milisegundos, mientras que para el simétrico fue de 16.869 milisegundos. Con esto, podríamos calcular la velocidad de procesamiento. Así que, se consideró la cantidad total de datos cifrados, que son 2000 bloques de 90 bytes y equivalen a 180,000 bytes. Esta cantidad la convertimos a megabytes dividiendo entre 1,048,576 bytes, resultando en aproximadamente 0.1717 MB procesados. Dividiendo los megabytes procesados entre el tiempo total en segundos, se obtiene la velocidad de cifrado alcanzada por el procesador. Así, en los casos de cifrado asimétrico y simétrico, se obtuvieron velocidades de aproximadamente 3.077 MB/s y 10.176 MB/s respectivamente.