Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
Semestre 2025-10

**Caso de Estudio 3 – Canales Seguros**
**Sistema de rastreo de paquetes en una compañía transportadora**

**Objetivos**

- Comprender ventajas y limitaciones de los algoritmos para cifrar datos.
- Construir un prototipo a escala de una herramienta para soportar confidencialidad e integridad.

**Problemática:**

Supondremos que una aerolínea presta servicios de consulta en línea para sus clientes. Los servicios son consulta del estado de un vuelo, disponibilidad de vuelos para un trayecto específico y costo de un vuelo. Para implementar estos servicios y distribuir la carga para mejorar tiempos de respuesta la aerolínea cuenta con un conjunto de servidores en los que atiende los diferentes servicios.

La Figura 1 muestra una descripción de alto nivel del protocolo (más adelante se explican los detalles). Cuando un cliente tiene una consulta, se conecta al servidor principal de la aerolínea e inicia un protocolo para establecer las llaves de cifrado de la sesión. A continuación, el servidor envía nombre e identificador de los servicios disponibles.

El cliente recibe los datos de los servicios disponibles y los despliega al usuario para que este escoja el servicio deseado. A continuación, el cliente envía al servidor el identificador del servicio elegido. El servidor responde con la dirección IP y puerto del servidor que atiende ese servicio específico.

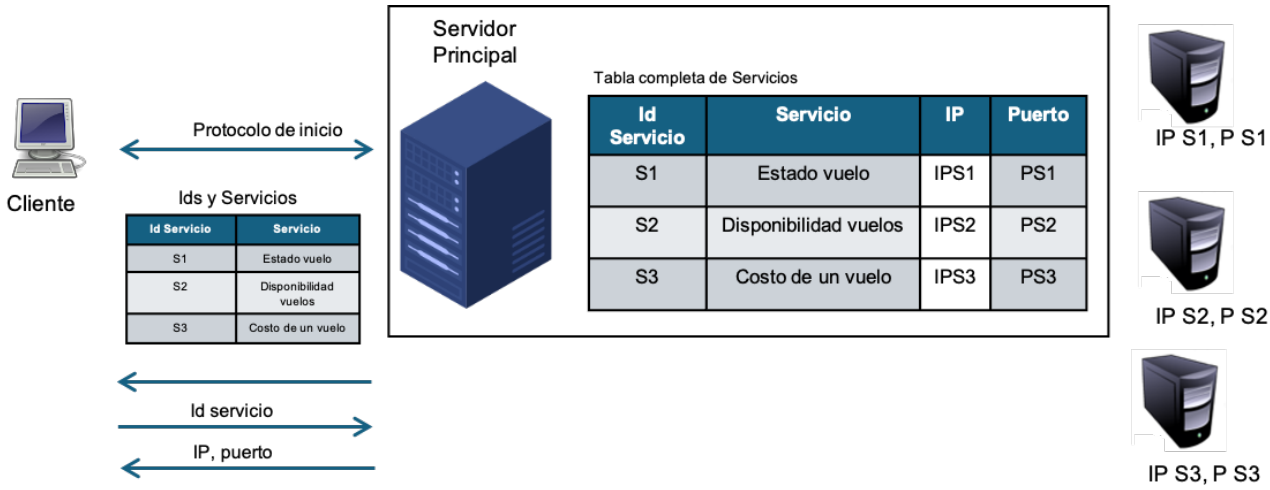


*Figura 1. Esquema de alto nivel de la comunicación entre el cliente y el servidor*

Una vez el cliente recibe la dirección y puerto del servidor que implementa el servicio de interés, el cliente establece una comunicación directa con ese servidor. Sin embargo, esta última parte (la comunicación con el servidor de interés) no hace parte del alcance de este caso.

**Implementación del Prototipo:**

Para este caso nos concentraremos en el proceso de consulta con el servidor principal. Su tarea consiste en construir los programas servidor principal y cliente, que deben comunicarse de forma segura (implementando algoritmos para garantizar confidencialidad e integridad de la sesión) y deben funcionar como se indica a continuación.

Servidor Principal:

- Guarda una tabla predefinida con la información completa de los servicios.
- Implementa servidores delegados.
- Para consultar los datos de un servicio específico, el cliente debe enviar al servidor el identificador del servicio que le interesa (de acuerdo con la información recibida previamente). Si el identificador de servicio no corresponde a una entrada en la tabla, el servidor retorna los valores -1,-1.

Cliente:

- Es un programa que inicia la comunicación.
- Cuando recibe la tabla de identificadores y servicios, permite al usuario seleccionar un servicio específico de interés y luego pide al servidor los datos de ese servicio específico, espera la respuesta y la despliega.

Tanto el servidor como el cliente deben validar los datos que reciben con HMAC. Si la validación pasa entonces el programa continúa, si no pasa entonces el programa debe desplegar el mensaje ¨Error en la consulta" y terminar.

El cliente y el servidor deben comunicarse siguiendo el protocolo descrito en la Figura 2.

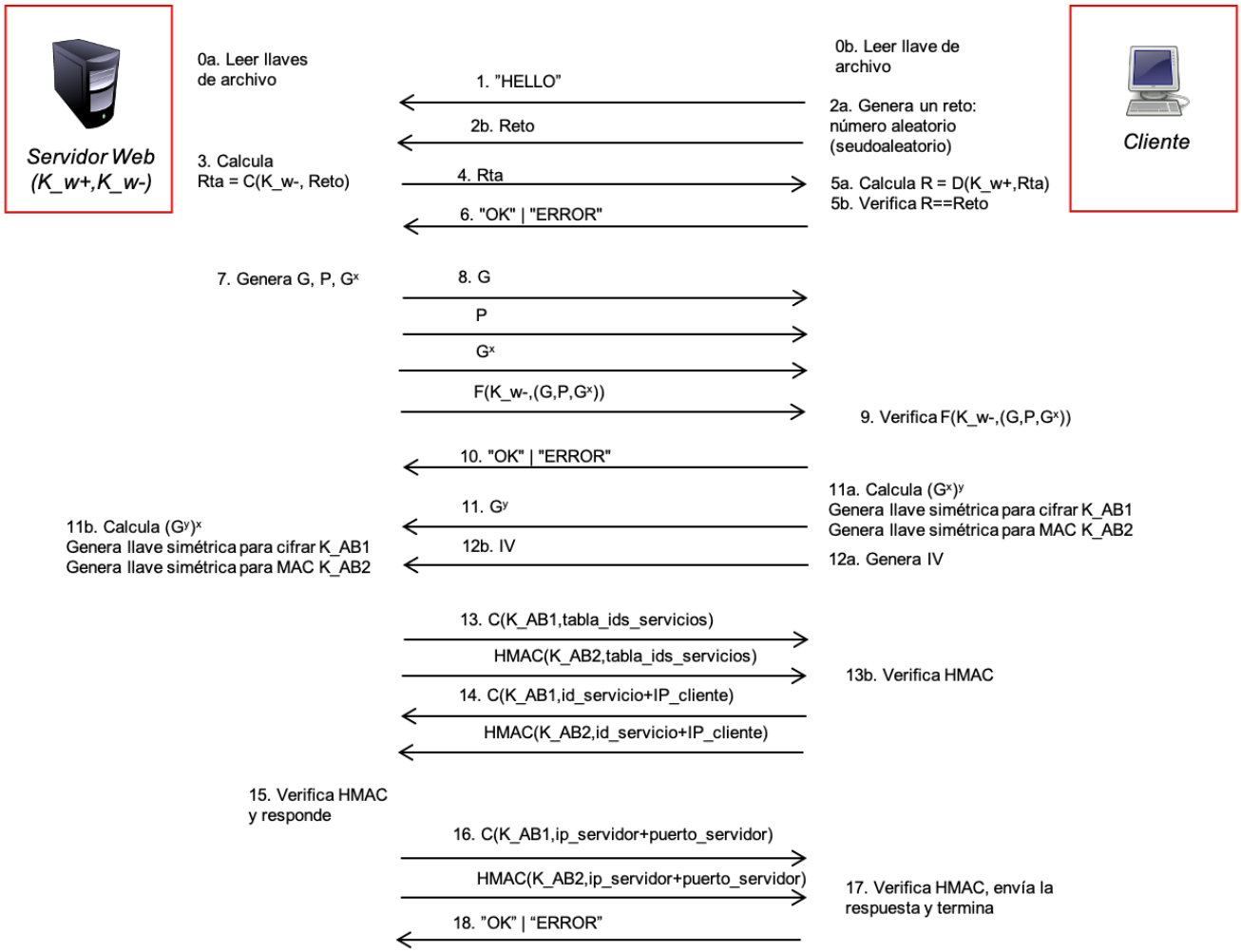


*Figura 2. Protocolo de Comunicación entre Cliente y Servidor.*

**PARA TENER EN CUENTA:**

Tanto servidor como cliente deben cumplir con las siguientes condiciones:

- Las comunicaciones entre cliente-servidor deben usar sockets y deben estar cifradas.
- No use librerías especiales, solo las librerías estándar.
- Desarrolle en Java (java.security y javax.crypto).
- Generaremos por adelantado las llaves pública y privada del servidor principal. El servidor debe tener acceso a las dos llaves, mientras el cliente solo tiene acceso a la llave pública. Tenga en cuenta que en una instalación real la llave pública puede ser leída por cualquiera, pero la llave privada debería tener acceso restringido y solo debería estar al alcance del propietario.

- Como algoritmos usaremos:
  - AES. Modo CBC, esquema de relleno PKCS5, llave de 256 bits.
  - RSA. Llave de 1024.
  - El vector de inicialización para cifrado simétrico (iv) debe tener 16 bytes generados aleatoriamente
  - Firma: SHA256withRSA
  - HMACSHA256.

- Para generar las llaves de sesión:
  - Primero calcule una llave maestra por medio de Diffie-Hellman. Para obtener el valor de p y el valor del generador g, raíz primitiva de p, puede apoyarse en el API de Java relacionada con algoritmos criptográficos, como AlgorithmParameterGenerator haciendo uso de AlgorithmParameters y DHParameterSpec.
  - Para manejar números grandes use la clase BigInteger de Java. Los números primos deben tener 1024 bits de longitud.
  - Luego use la llave maestra para calcular un digest con SHA-512.
  - Parta el digest en dos mitades para generar la llave para cifrado y la llave para código de autenticación: usar los primeros 256 bits se deben usar para construir la llave para cifrar, y los últimos 256 bits para construir la llave para generar el código HMAC.

Para evaluar la sobrecarga generada por los algoritmos de cifrado y manejo de integridad, el cliente y el servidor deben crear delegados y los delegados deben comunicarse siguiendo el protocolo descrito en la figura:

- El servidor principal crea los delegados por conexión al recibir cada cliente.
- En los clientes seleccione el servicio de interés de forma aleatoria.

Adicionalmente, resuelva las siguientes tareas:

1. Corra su programa en diferentes escenarios y mida el tiempo que el servidor requiere para: (i) Firmar, (ii) cifrar la tabla y (iii) verificar la consulta. Los escenarios son:
   (i) Un servidor de consulta y un cliente iterativo. El cliente debe generar 32 consultas secuenciales.
   (ii) Servidor y clientes concurrentes. 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. Construya una tabla con los datos recopilados. Tenga en cuenta que necesitará correr cada escenario en más de una ocasión para validar los resultados.
3. Compare el 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. Construya las siguientes gráficas:
   (i) Una que compare los tiempos para firmar en los escenarios
   (ii) Una que compare los tiempos para cifrar la tabla en los escenarios
   (iii) Una que compare los tiempos para verificar la consulta en los escenarios
   (iv) Una que muestre los tiempos para el caso simétrico y el caso asimétrico en los diferentes escenarios.
5. Escriba sus comentarios sobre las gráficas, explicando los comportamientos observados.
6. Defina un escenario que le permita estimar la velocidad de su procesador, y estime cuántas operaciones de cifrado puede realizar su máquina por segundo (en el caso evaluado de cifrado simétrico y cifrado asimétrico). Escriba todos sus cálculos.

**Entrega:**

- Cada grupo debe entregar un zip con (se reitera que la entrega es un archivo zip. No entregue enlaces a repositorios de ningún tipo, ni archivos sueltos):
    - Archivos fuente con la implementación del prototipo del cliente y el servidor y los archivos con las llaves del servidor (uno para la llave pública y uno para la llave privada).
    - Un subdirectorio *docs* con un informe que incluya: (i) la descripción de la organización de los archivos en el zip, (ii) las instrucciones para correr servidor y cliente, incluyendo cómo configurar el número de clientes concurrentes, (iii) respuestas a todas las tareas y preguntas planteadas en este enunciado.
    - Un archivo Excel con todos los datos recopilados. Incluya al comienzo de cada hoja una descripción corta del escenario correspondiente.
    - **Al comienzo del informe, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente. Sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
    - Recuerde incluir en su informe **todas las referencias** que use para resolver este proyecto.
- El trabajo se debe realizar en grupos de **2** personas y puede conformar los grupos de trabajo por su cuenta. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes). Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación será parte de la calificación de todos los miembros. Tenga en cuenta que desarrollar sus habilidades para trabajo en grupo le puede ayudar en su futuro profesional para integrarse más fácilmente a un grupo de trabajo y trabajar de forma productiva.
- En el parcial se incluirá una pregunta sobre el caso.
- El proyecto debe ser entregado por Bloque Neón.
- La fecha de entrega no será movida.
- Política de entrega tarde. Para las entregas tarde, se aplicará la siguiente política: por cada 30 minutos o fracción de retraso, con respecto a la hora de entrega establecida en Bloque Neón, habrá una penalización de 0,5/5. Adicionalmente, tenga en cuenta que para entregar por correo debe borrar los archivos .class antes de crear el archivo .zip.
- **La fecha límite de entrega es Abril 28, 2025 a las 23:50 p.m.**

Cronograma Propuesto:

- Abril 7-8: Lectura del enunciado
- Abril 9-10: Diseño de la solución
- Abril 11-23: Implementación y Pruebas
- Abril 24-26: Recolección de datos y construcción del informe
- Abril 28: Entrega

**Referencias:**


- *Cryptography and network security*, W. Stallings, Ed. Prentice Hall, 2003.
- *Computer Networks*. Andrew S. Tanenbaum. Cuarta edición. Prentice Hall 2003, Caps 7, 8.
- *Blowfish*. Página oficial es: http://www.schneier.com/blowfish.html
- *RSA*. Puede encontrar más información en: http://www.rsa.com/rsalabs/node.asp?id=2125
- *CD X509*. Puede encontrar la especificación en: http://tools.ietf.org/rfc/rfc5280.txt
- *MD5*. Puede encontrar la especificación en : http://www.ietf.org/rfc/rfc1321.txt