Juan David Pérez Díaz - 202210323

Samara Martínez Jacome - 202221057

Informe Caso 3

1. Descripción de la Organización de los Archivos

La carpeta de fuente ¨ProtoServer¨ cuenta con una carpeta titulada ¨src¨ dentro de esta se encuentran cuatro carpetas ¨Clientes¨, “Servidores”, “Cifrado” y “Llaves”. En cada una de estas carpetas se encuentra el código fuente como tal. A continuación, se explicará cada uno de los archivos que contiene cada carpeta.

Clientes
En esta carpeta se encuentra únicamente la clase llamada “Cliente” la cual se comporta como un Thread. Esta clase representa a un cliente el cual se conecta al servidor con el protocolo seguro de intercambio de claves con autenticación y verificación. Esta empieza estableciendo la comunicación con el servidor enviando un saludo y luego un reto. Luego, recibe el reto firmado por el servidor y verifica esta firma utilizando la clave pública del servidor. Después de esto, se hace el intercambio de claves de Diffie-Hellman en donde ambos (tanto servidor como cliente) acuerdan una clave compartida. Con esta clave el cliente deriva dos claves simétricas una para integridad (HMAC) y la otra para cifrado (AES). El cliente utiliza estas claves para descifrar la tabla de servicios que el servidor le ofrece y verifica su integridad. Por último, el cliente selecciona un servicio, luego cifra su id y su dirección IP para enviársela al servidor. El servidor le retorna estos datos cifrados, verifica la integridad y confirma al servidor si son correctos. Todo esto se hace en un solo método llamado enviarSaludo que recibe por parámetro el socket por el cual se establece la comunicación con el servidor.

Cifrado
En la carpeta de Cifrado también hay una sola clase, esta es “CifradoUtils”. Esta clase básicamente es la que proporciona todos los métodos de cifrado y descifrado tanto asimétricos como simétricos. También hay métodos para las firmas digitales y generación de resúmenes criptográficos como lo es digest. Esta clase permite cifrar y descifrar los datos usando AES en modo simétrico y el algoritmo de RSA para cifrado asimétrico. Así mismo facilita la lectura de claves públicas y privadas, los cálculos de digest utilizando SHA-256 y la generación de HMACS para la integridad y autenticidad de los datos tanto enviados como recibidos. Esta clase es la encargda de todas estas operaciones para hacer más simple y seguro el manejo de los datos cifrados en el programa.

Llaves
En esta carpeta se encuentran 3 archivos los cuales 2 de estos son la llave pública y la llave privada en archivos “.key”, el tercer archivo es la clase “CrearLlaves”. Esta clase se utiliza para generan el par de claves usando RSA, en esta crea una instancia de “KeyPairGenerator”, se

envía la configuración para que sea con 1024 bits y se generan las dos claves tanto pública como privada. Luego de que se generan estas clases se serializan y se almacenan en los archivos ".key" con el método de saveObject que utiliza un "ObjectOutputStream". Esta clase prepara las llaves para poder hacer las operaciones de cifrado, firma y autenticación que se utilizan en el programa.

Servidores

Esta carpeta cuenta con 4 archivos .java 2 de estos son los mains correspondientes al servidor y al cliente, y los otros son para el servidor.

La clase "ServidorMain" es el que activa el servidor, este inicializa los servicios disponibles que se almacenan en un "HashMap" y abre un "ServerSocket" en el puerto 65000 para poder escuchar las conexiones de los clientes. Por cada cliente que se conecta al servidor crea un nuevo hilo de "ManejadorCliente" para poder manejar la comunicación y permitir la concurrencia a varios clientes. Igualmente, esta muestra en la terminal la información básica sobre las conexiones entrantes. Esta clase básicamente organiza y mantiene la disponibilidad de los servicios: "Estado vuelo", "Disponibilidad vuelos" y "Costo de un vuelo" y expone el método para acceder a estos desde las otras clases.

Por otra parte, esta también la clase "ManejadorCliente" la cual se comporta como un Thread, esta es donde se encuentra toda la lógica del servidor, es la encargada de manejar la comunicación segura con un cliente individual. La ejecución empieza con el recibimiento del reto y del saludo enviado por el cliente que luego es firmado digitalmente usando la clave privada del servidor para poderse autenticar ante el cliente. Luego se implementa el protocolo de intercambio de llaves Diffie-Hellman para poder establecer la llave compartida simétrica que se divide para poder generar 2 claves independientes: una para cifrado AES y la otra para la verificación HMAC. Posteriormente, se envía la tabla cifrada con los servicios disponibles del servidor al cliente utilizando el cifrado simétrico AES y luego se cifra de manera asimétrico con el algoritmo de RSA para poder comparar el desempeño. Este también genera un HMAC para asegurarse que el mensaje llegue integro al cliente. Finalmente, este verifica las respuestas del cliente y completa la autenticación mutua al enviar su IP y puerto cifrados y autenticados.

En esta carpeta también se encuentra la clase "Main" este es el main del Cliente. Esta permite la comunicación entre clientes y el servidor en los dos escenarios propuestos. A través de un menú que permite escoger o el modo iterativo, en donde un solo cliente realiza 32 conexiones al servidor, o el modo concurrente en onde varios clientes (pueden ser 4, 16, 32 o 64) se conectan simultáneamente. En el escenario iterativo cada cliente se conecta ejecuta la operación y luego se desconecta antes de que el siguiente pueda comenzar. El otro escenario crea varios Threads el cual permite evaluar el comportamiento y el desempeño del servidor.

El ultimo archivo de esta carpeta es la clase "Servicio" la cual modela la representación de un servicio ofrecido por el servidor. Cada instancia de la clase contiene 4 atributos un id, un nombre, un numero de puerto asociado y una dirección IP. En esta clase están todos los getters y setters para acceder y modificar los atributos. Esta clase permite transmitir la información estructurada de los servicios disponibles y facilitando consultas.

2. Instrucciones Para Correr el Servidor y Cliente

Para iniciar el programa, primero debe ejecutarse la clase `ServidorMain.java` la cual se encuentra en el paquete Servidores y, una vez hecho esto, proceder a correr la clase `Main.java` (También dentro del paquete Servidores). Siguiendo los pasos que se presentan a continuación:

Servidor:
En la clase `ServidorMain.java`, únicamente debe ejecutarse el método `main()` que allí se encuentra y esperar a que se imprima en consola el mensaje "Servidor iniciado". Este debe realizarse cada vez que se desee ejecutar un escenario de prueba, ya que, de lo contrario, se mostrará el número total de clientes y no el número separado por escenarios. Sin embargo, no hay inconveniente si se decide no reiniciar el servidor en cada escenario de prueba.

Main:
Una vez el servidor se haya activado en el paso anterior, se pueden realizar las pruebas con los clientes. Para esto, desde la clase `Main.java`, debe ejecutarse el método `Main()` que allí se encuentra. Al hacerlo, se mostrarán las dos opciones disponibles para probar.
Si se desea probar el escenario en el que un único cliente realiza diferentes peticiones, se debe ingresar el número 1 en la consola y esperar a que se muestren los resultados.
Por otro lado, si se desea probar el escenario en el cual varios clientes interactúan de manera concurrente con el servidor, se debe ingresar el número 2. Después de esto, se solicitará ingresa la cantidad de clientes, la cual puede ser 4, 16, 32 y 64 (aunque, de ser necesario, se puede ingresar cualquier otro número). Una vez hecho esto, solo se debe esperar a que se muestren los resultados en la consola.

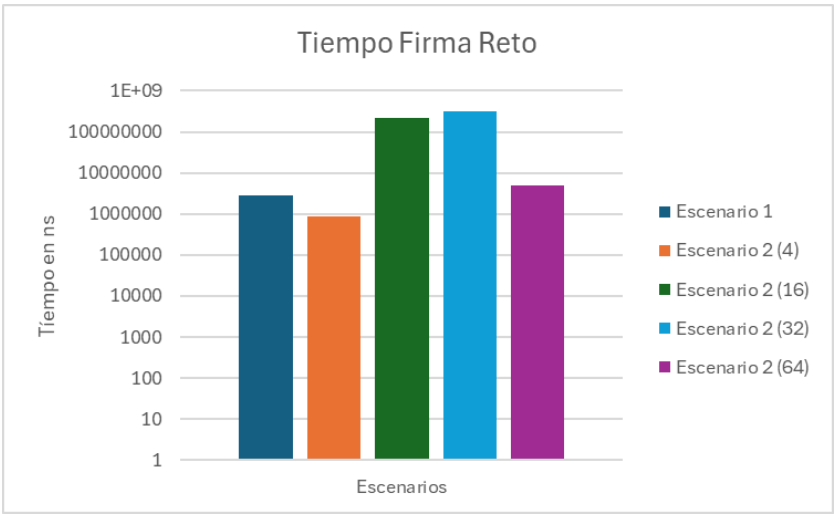
Finalmente, dado que se están ejecutando dos clases de manera simultánea, existirán dos terminales activas que imprimirán información diferente. Sin embargo, únicamente la terminal creada al ejecutar `Main.java` recibirá entradas del usuario que está probando el programa.

3. Graficas

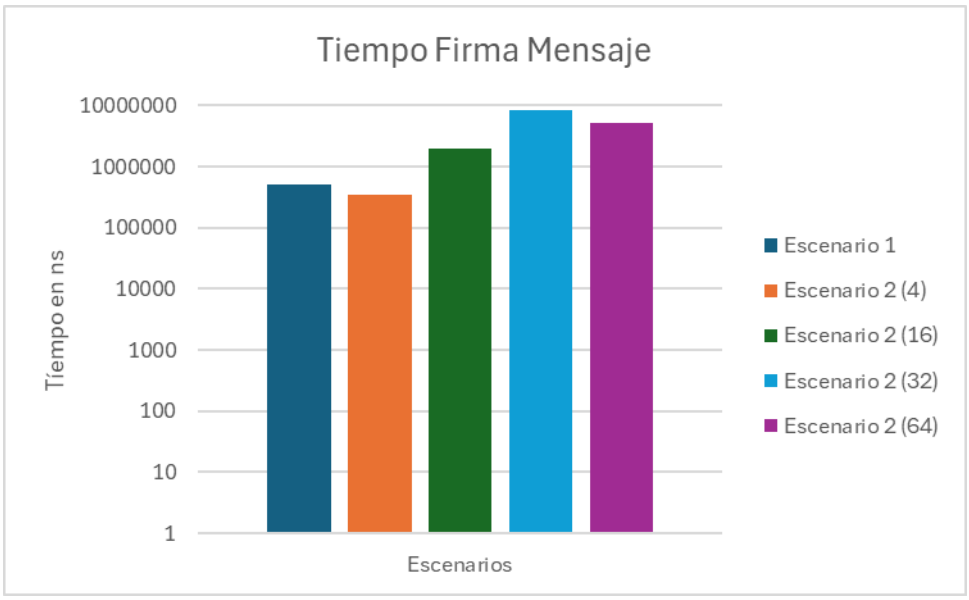
Para graficar los resultados de las pruebas, lo que se hizo fue encontrar el promedio de cada uno de los escenarios y estos fueron los que se graficaron. La ecuación para obtener el promedio fue la siguiente:

$$\bar{X} = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

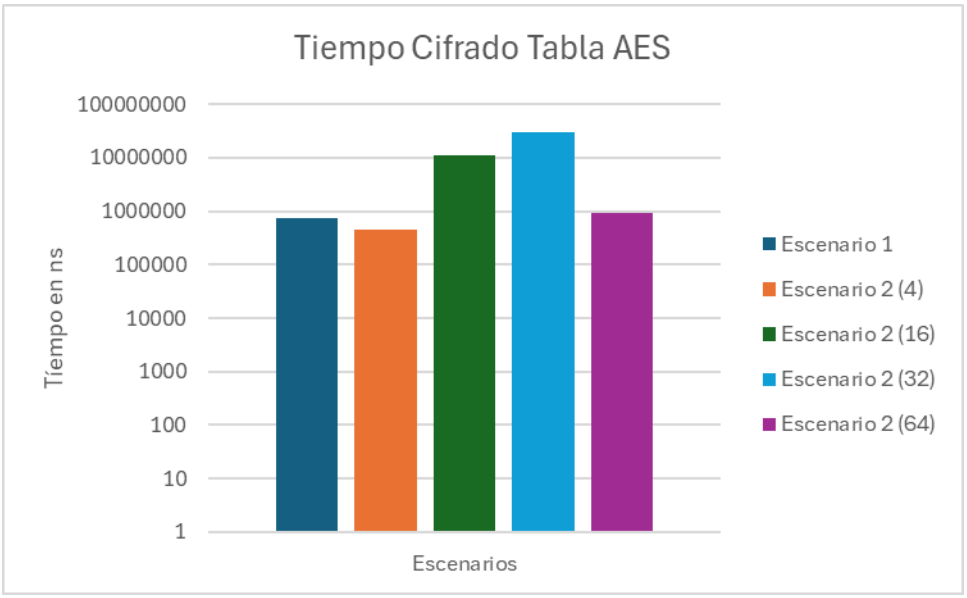
El valor que se graficó en cada escenario fue el correspondiente a X en la ecuación.



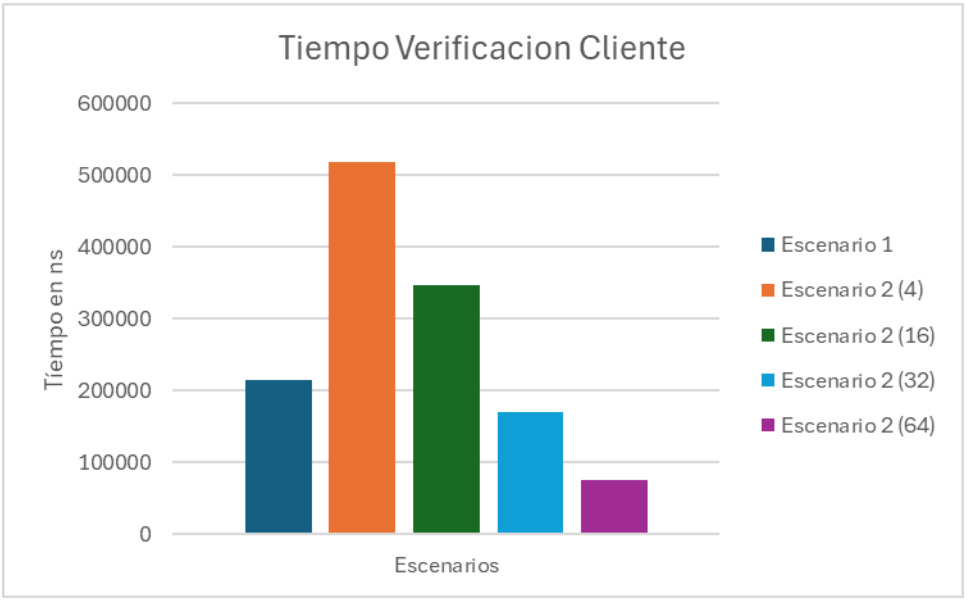
Esta grafica muestra en escala logarítmica cómo cambia el tiempo necesario para firmar digitalmente el reto bajo diferentes escenarios de carga. Se puede observar como en el escenario 2(16) y Escenario 2 (32) se encuentran los valores más altos, mientras en los demás escenarios el valor es similar.



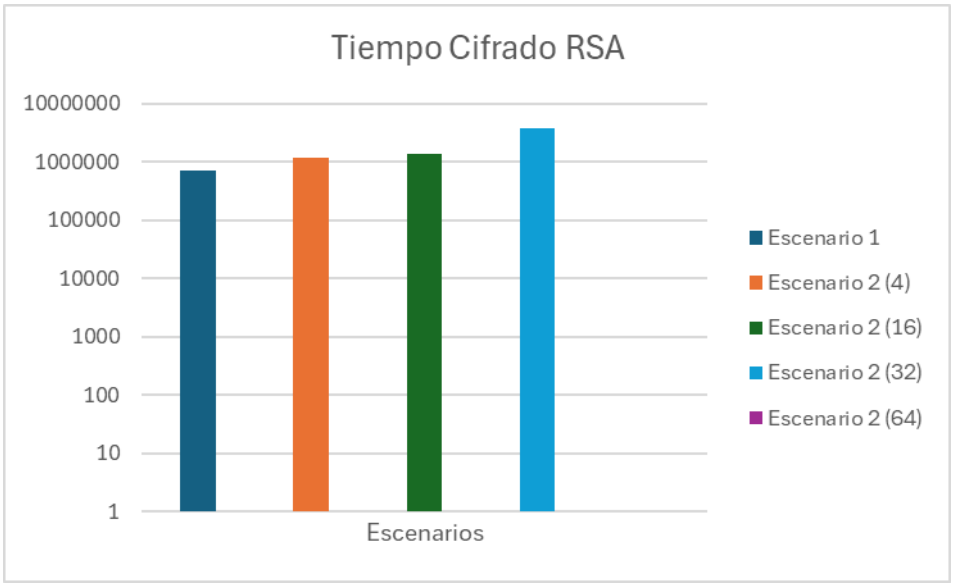
Esta grafica muestra en escala logarítmica cómo cambia el tiempo necesario para firmar digitalmente el mensaje bajo diferentes escenarios de carga. Se puede observar como en el escenario 2 (64) y Escenario 2 (32) se encuentran los valores más altos, mientras en los demás escenarios el valor es similar.



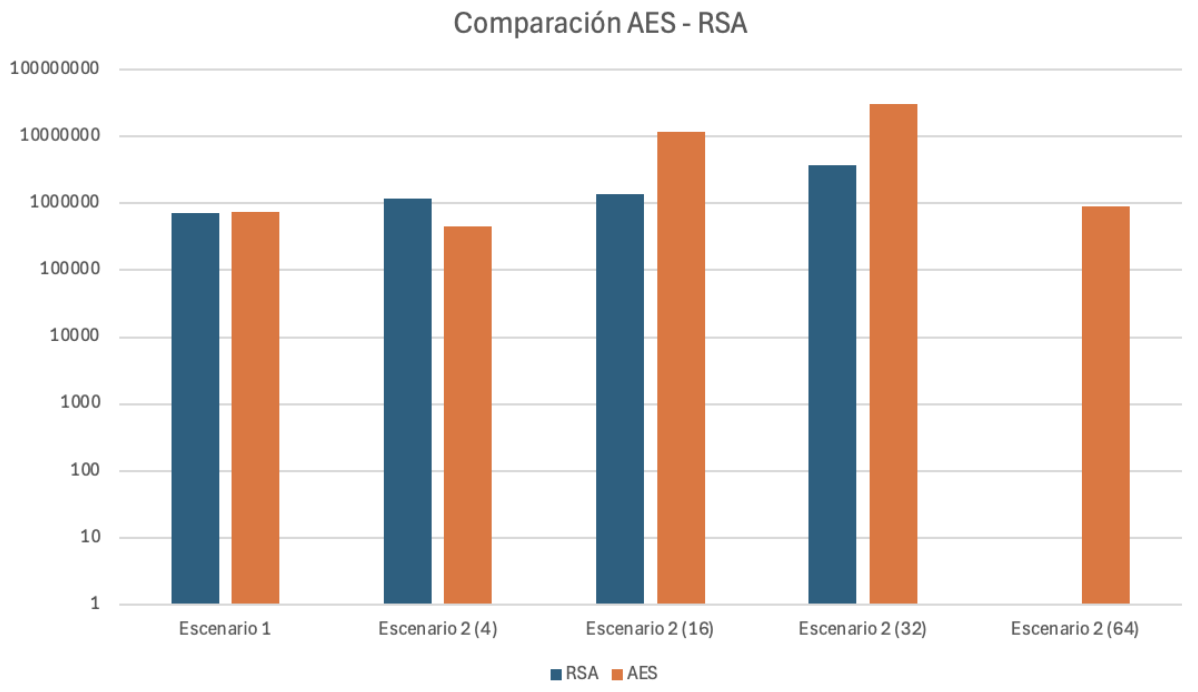
Esta grafica muestra en escala logarítmica cómo cambia el tiempo necesario para cifrar la tabla con AES bajo diferentes escenarios de carga. Se puede observar como en el escenario 2 (32) y Escenario 2 (16) se encuentran los valores más altos, mientras en los demás escenarios el valor es similar.



Esta grafica muestra en escala logarítmica cómo cambia el tiempo de verificación por parte del cliente bajo diferentes escenarios de carga. Se puede observar como en el Escenario 2 (4) se encuentran los valores más altos, mientras en los demás escenarios el valor disminuye bastante.



Esta grafica muestra en escala logarítmica cómo cambia el tiempo necesario para cifrar la tabla con AES bajo diferentes escenarios de carga. Se puede observar como en el escenario 2 (32) se encuentran los valores más altos, mientras en los demás escenarios el valor es similar.



Esta gráfica enseña en escala logarítmica una comparativa entre los datos de encriptación con RSA y con AES. Para el caso en el que es un solo cliente iterativo es igual. En los otros escenarios se observa que el AES es mejor cuando hay más clientes.

4. Preguntas Adicionales

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.

Para estimar la velocidad del procesador y calcular cuantas operaciones de cifrado puede realizar la maquina con la que se realizaron las pruebas por segundo en cifrado simétrico como asimétrico, se debe encontrar el tiempo total que tarda en cifrar un conjunto de datos y luego calcular cuantas operaciones se podrían realizar en un segundo. Para este se utilizarán los datos con clientes secuenciales.

Tiempo de cifrado de una operación con AES (promedio): 757165.625 ns/operación

Tiempo de cifrado de una operación con RSA (promedio): 727933.625 ns/operación

Se utilizará la formula:

$$OxS = \frac{1000000000\ ns}{T_{operacion}\left(\frac{ns}{op}\right)}$$

Cifrado Simétrico AES:

$$\frac{1000000000\ ns}{757165.625\ \left(\frac{ns}{op}\right)} \approx 1320\ \frac{operaciones}{segundo}$$

Cifrado Asimétrico RSA:

$$\frac{1000000000\ ns}{727933.625\ \left(\frac{ns}{op}\right)} \approx 1373\ \frac{operaciones}{segundo}$$

En este caso se pueden realizar alrededor de 1320 operaciones de cifrado Simétrico AES en la máquina de pruebas por segundo y 1373 operaciones de cifrado Asimétrico RSA con la máquina de pruebas por segundo. Esto implica que el cifrado asimétrico es un poco más rápido que el cifrado simétrico.