Infraestructura de Computación - Caso 3

Juan David Guevara Arévalo - 202116875

Natalia Valentina Espitia Muñoz - 202215087

1. Descripción de la organización de los archivos.

El proyecto cuenta con 3 subdirectorios importantes. El primer subdirectorio “src” contiene el código fuente, el subdirectorio “data” contiene datos que son públicos, es decir, que no importa si son leidos por un cliente. El subdirectorio “privateData” contiene los archivos que solo deben ser leidos por el Servidor, y no por el cliente.

Para abordar el caso propuesto se utilizaron 3 clases principales que están en el subdirectorio *src.* La primera clase corresponde al Servidor, en la cual se cargan los datos de los paquetes, además de las llaves asimétricas, en esta clase tambien se configura el entorno a utilizar (servidor concurrente o servidor iterativo) y crea el ServerSocket utilizado para comunicar a los clientes.

La segunda clase ConnectionHandler maneja el protocolo descrito en la guía por parte del servidor. Ahí se implementan los diferentes metodos de cifrado, generación de parametros DH y se maneja el Socket de conexión para un cliente.

La tercera clase principal, corresponde al Cliente. Dado que un Cliente solo debe manejar una conexión y el protocolo de comunicación de ladop del cliente, se decidió que esta clase maneje tanto la configuración del entorno como la ejecución del protocolo. Nuevamente, en esta clase se implementan métodos para encriptar y desencriptar con la clave simetrica, además de utilizar la clave pública del servidor. Las demás clases utilizadas en el proyecto son auxiliares, no influyen directamente en la lógica de la solución.

2. Instrucciones de ejecución y configuración

Servidor: Para ejecutar el servidor, se debe ejecutar la clase Server, la cual cuenta con un método main. El Servidor recibe por entrada estandar si se debe ejecutar en modo iterativo o concurrente. En el caso de que se indique el modo concurrente, el Servidor preguntará el número de delegados que se quiere tener disponible.

Cliente: De forma similar al Servidor, se debe ejecutar la clase Client. Esta clase recibe por entrada estandar la indicación de si se debe ejecutar en modo iterativo o en modo concurrente. Independientemente de la opción seleccionada, el Cliente preguntará cuantas peticiones se quieren realizar, si se ejecutó en modo iterativo, un solo cliente realizará las peticiones ingresadas, de lo contrario, se creará un delegado concurrente para cada petición.

Nota:

- El puerto **8000** debe estár libre para que se puedan realizar las conexiones.
- El desarrollo fue realizado utilizando **Java 21**.

3. Compilación de datos de diferentes escenarios en el programa.

i . Un servidor iterativo y un cliente iterativo. El cliente genera 32 consultas.

La siguiente tabla compila el promedio de la respuesta del programa al reto, la generación de los parámetros y la verificación de consultas cuando el servidor y el cliente es iterativo con un cliente que genera 32 consultas:

	32 consultas		
	Responder al reto	Generar parametros	Verificar la consulta
PROMEDIO (ms):	1,92	1219,26	0,56

Nota importante: Este promedio es resultante de 64 tomas de datos por cada requerimiento del escenario. Para ver los datos individuales por favor diríjase al repositorio en la carpeta data/tablas_graficas.

ii. Un servidor y un cliente que implementen delegados. El número de delegados, tanto servidores como clientes, debe variar entre 4, 8, y 32 delegados concurrentes. Cada cliente genera una sola solicitud.

Se generaron las siguientes tablas con los promedios de los datos recopilados de las diferentes variaciones en los escenarios:

1. Tabla donde se muestra el promedio del tiempo de respuesta cuando el programa responde al reto con 4, 8 y 32 delegados y clientes:

Responder al reto			
	Delegados		
Clientes	4	8	32
4	24,07	24,02	19,08
8	12,27	27,42	0,91
32	3,17	7,50	51,42

2. Tabla donde se muestra el promedio del tiempo de respuesta cuando el programa genera G, P y Gx con 4, 8 y 32 delegados y clientes:

Generar parámetros			
	Delegados		
Clientes	4	8	32
4	368,25	1584,06	1009,51
8	1714,67	978,70	1693,84
32	1170,00	1958,90	4919,56

3. Tabla donde se muestra el promedio del tiempo de respuesta cuando el programa verifica la consulta con 4, 8 y 32 delegados y clientes:

Verificar la consulta			
	Delegados		
Clientes	4	8	32
4	8,18E-01	0,85	0,92
8	0,61	1,01	0,28
32	0,33	0,54	0,78

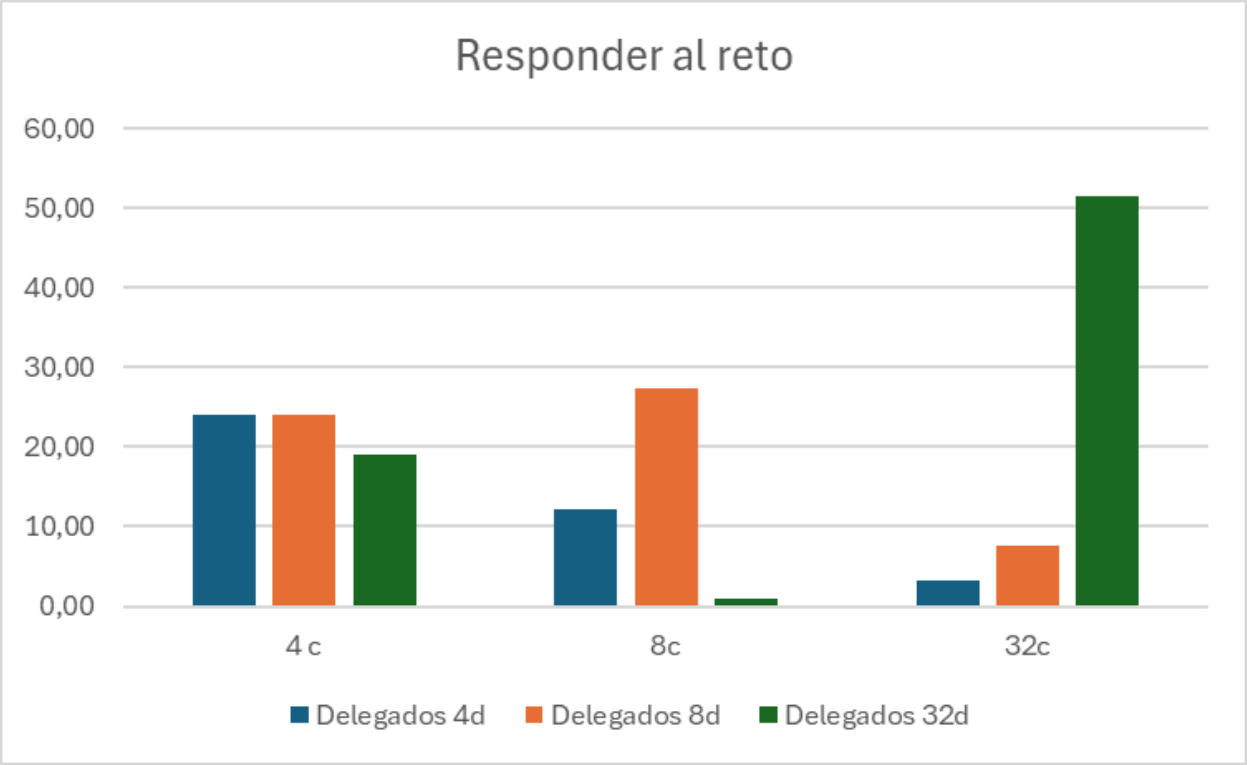
Nota importante: Este promedio es resultante de 88 tomas de datos por cada requerimiento del escenario. Para ver los datos individuales por favor diríjase al repositorio en la carpeta data/tablas_graficas.

4. Comparación de los tiempos en los diferentes cifrados:

Cifrado	Simetrico	Asimetrico
Promedio:	135432,14	819216,81

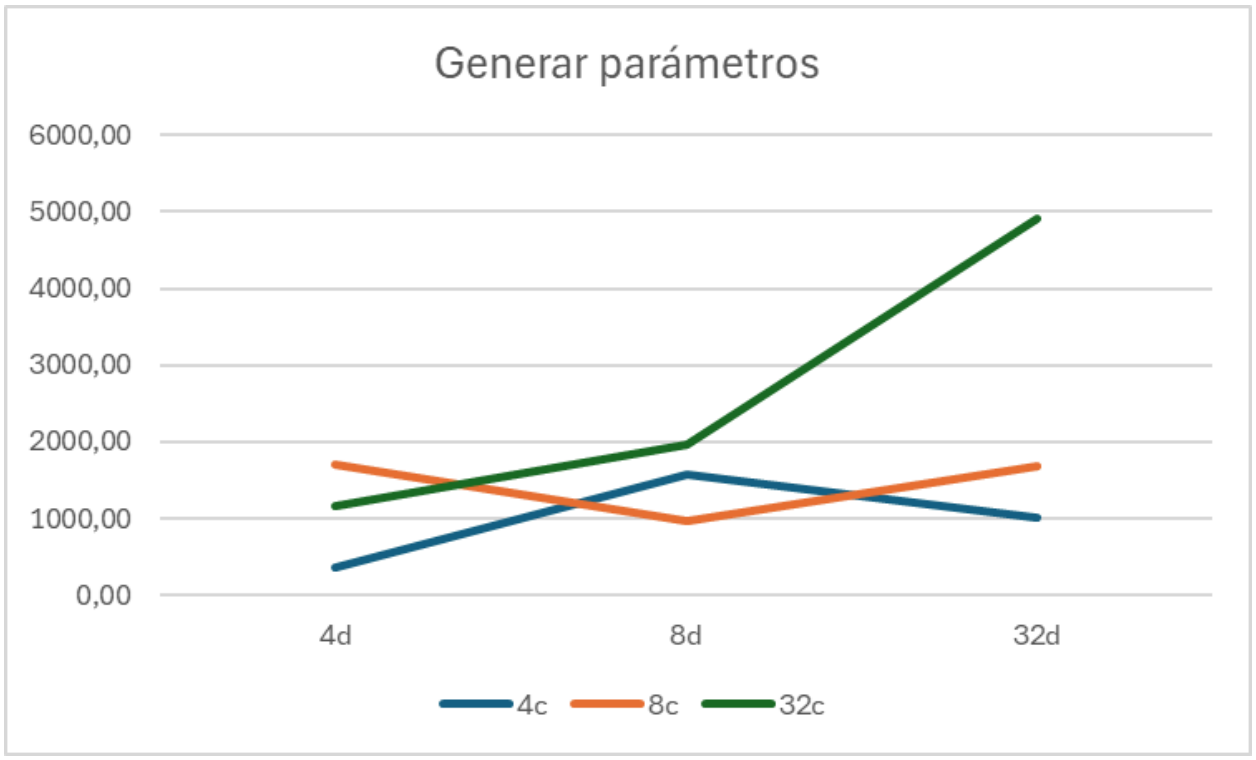
5. Gráficas y análisis de gráficas.

1. Tiempos para descifrar el reto en los diferentes escenarios



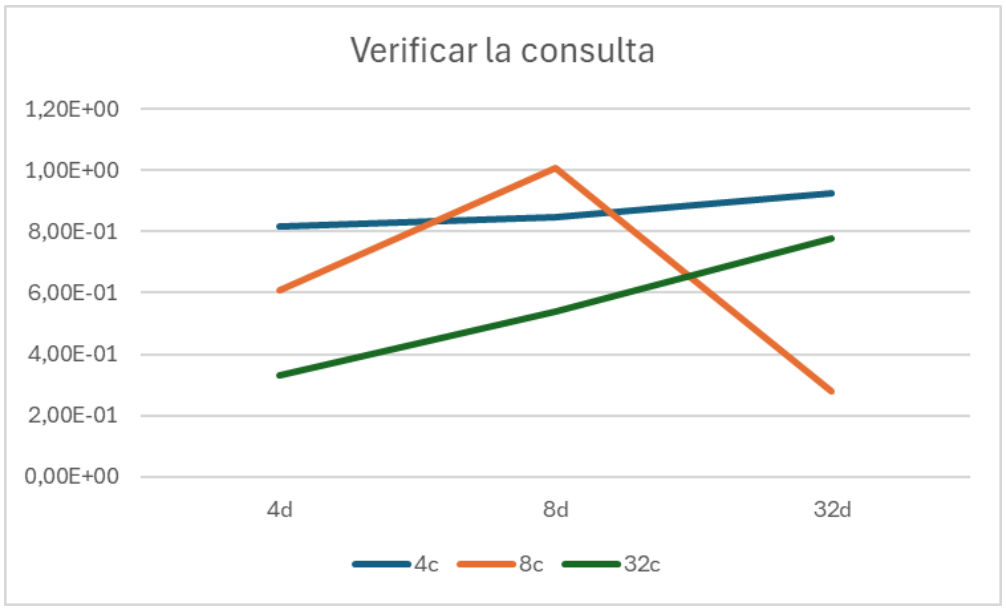
Podemos ver que la los clientes con delegados de tamaño 8 son los más estables, mientras que si el tamaño es de 32 este tiene un comportamiento más variante porque se puede decir que un tamaño de 8 puede generar más estabilidad en el programa pero puede ser menos escalable.

2. Tiempos para generar G, P y Gx en los diferentes escenarios



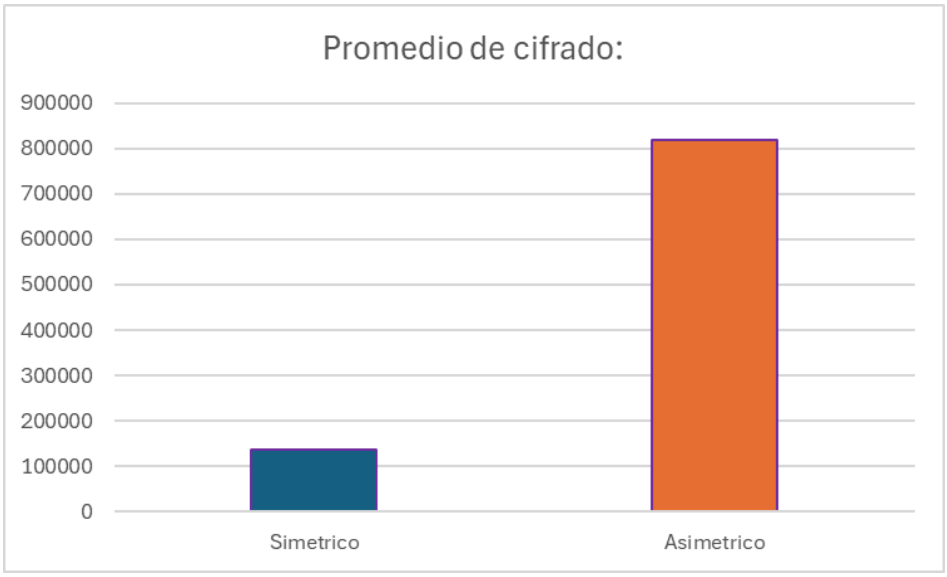
El crecimiento de la carga de los 32 clientes puede deberse al aumento en la cantidad de datos de estado enviados al servidor con el tiempo. Sin embargo, aunque los tiempos aumentan, parece ser la única que soporta el aumento de la carga.

3. Tiempos para verificar la consulta en los diferentes escenarios



Para las consultas de los clientes, la configuración de 32 parece ser la más óptima, porque permite manejar un mayor volumen de consultas con tiempos de respuesta bajos si se compara con los resultados con 4 y 8 delegados. Esto quiere decir que entre mayor sea la capacidad en los delegados mejor será el tiempo de respuesta.

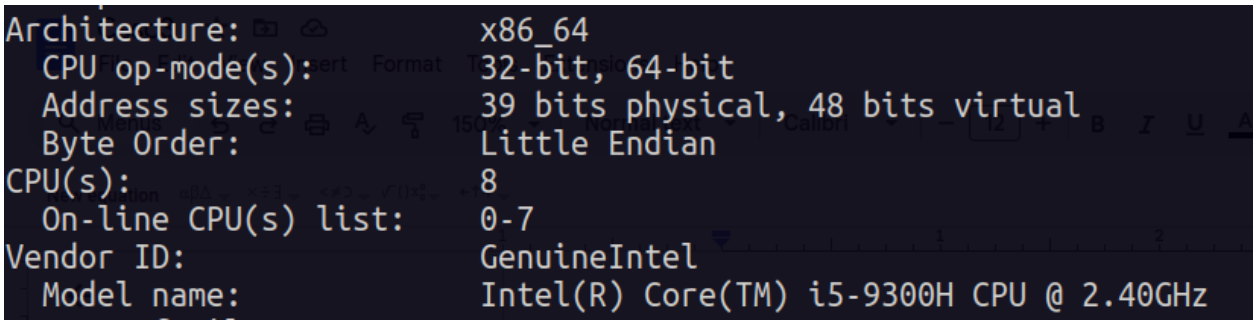
4. Tiempos para el caso simétrico y el caso asimétrico en los diferentes escenarios



El cifrado asimétrico, aunque ofrece altos niveles de seguridad, tiene una gran desventaja en términos de tiempo y consumo de recursos en comparación con el cifrado simétrico. Como podemos ver en la gráfica, mientras que el cifrado simétrico es considerablemente más rápido y eficiente, su nivel de seguridad puede ser menor, lo que lo hace más vulnerable a ciertos tipos de ataques. Por esta razón, es fundamental evaluar los requerimientos específicos de la información que se desea proteger. Si el rendimiento y la velocidad son prioritarios, el cifrado simétrico podría ser más adecuado; sin embargo, para datos extremadamente sensibles, el cifrado asimétrico ofrece una capa de seguridad adicional que puede ser indispensable a pesar de ser mucho más pesado.

6. Identificación de velocidad y cálculo de operaciones de cifrado.

Resultado del comando lscup en el computador utilizado ejecutar todas las pruebas:



Para estimar la cantidad de operaciones de cifrado que puede realizar el equipo se utilizó la herramienta OpenSSL speed:

Estimación de operaciones OpenSSL AES-256-CBC (Cifrado Simétrico):

Bloques de 16 bytes: 878320.73 KiB por segundo procesados.

Bloques de 64 bytes: 1058407.51 KiB por segundo procesados.

Bloques de 256 bytes: 1081779.11 KiB por segundo procesados.

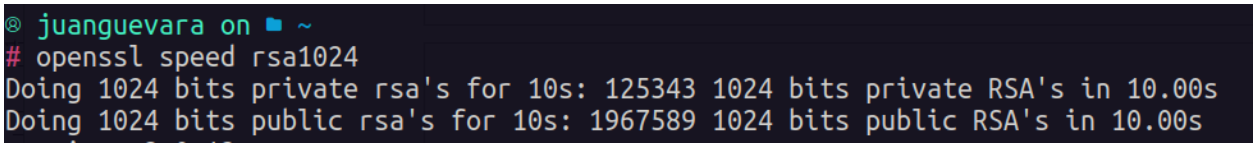
Bloques de 1024 bytes: 1087290.71 KiB por segundo procesados.

Bloques de 8192 bytes: 1090876.76 KiB por segundo procesados.

Bloques de 16384 bytes: 1088760.49 KiB por segundo procesados.

Como se puede apreciar, la cifra se estabiliza a partir de 256 bytes, por lo que tomaremos la media de dichas mediciones: 1087176.77KiB/s lo que equivale a aproximadamente a 1,04 GiB/s.

Estimación de operaciones OpenSSL RSA-1024 (Cifrado Asimétrico):



Dados los resutlados, podemos afirmar que se pueden realizar 1253,43 operaciones de firma por segundo, y 19675,9 operaciones de verificación por segundo.

Referencias

Java® Platform, Standard Edition & Java Development Kit Version 21 API specification (2024a) *java.net (Java SE 21 & JDK 21)*. Available at: https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/net/package-summary.html (Accessed: 03 November 2024).

Java® Platform, Standard Edition & Java Development Kit Version 21 API specification (2024b) *java.security (Java SE 21 & JDK 21)*. Available at: https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/security/package-summary.html (Accessed: 04 November 2024).

Java® Platform, Standard Edition & Java Development Kit Version 21 API specification (2024c) *javax.crypto (Java SE 21 & JDK 21)*. Available at: https://docs.oracle.com/en/java/javase/21/docs/api/java.base/javax/crypto/package-summary.html (Accessed: 04 November 2024).

Speed - OpenSSL Documentation. Available at: https://docs.openssl.org/1.1.1/man1/speed/#synopsis (Accessed: 05 November 2024).