Documentación Caso 2 Infracomp

Ponto Andrés Moreno Trejos - 202224525
Juan Diego Niebles Navas - 202221193

(i) Descripción de la organización de los archivos en el zip

Los archivos del ZIP se encuentran organizados de la siguiente manera:

Carpetas de pruebas de rendimiento: Contienen resultados de pruebas de tiempo para distintas configuraciones de delegados y clientes, con nombres en el formato `#Delegados#Cliente-Pruebas`, donde `#` representa el número de delegados y clientes (1, 4, 8 o 32). Cada carpeta incluye los siguientes archivos:

- `TiempoDescifrarReto`
- `TiempoGenerarGP`
- `TiempoVerificarResponde`
- `TiempoVerificarFirma` (algunas carpetas lo incluyen)

Directorio `Caso3`:

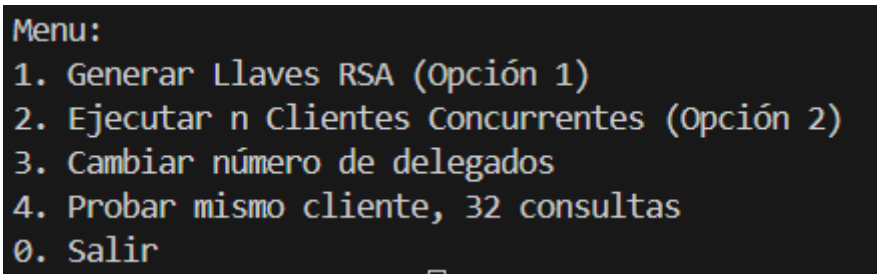
- Subcarpeta `docs`: Contiene el documento con el analisis.
- Subcarpeta `src`: Contiene los archivos fuente en Java:
  - `App.java`: Punto de entrada principal del proyecto. Se corren aqui las opciones 1 y 2.
  - `Asimetrico.java` y `Simetrico.java`: Implementan las operaciones de cifrado asimétrico y simétrico.
  - `Cliente.java` y `Servidor.java`: Manejan la lógica cliente-servidor.
  - `DiffieHellman.java`: Implementa el protocolo Diffie-Hellman.
- Subcarpetas `privado` y `publico`:
  - `privado`: Almacena la llave privada (`llavePrivada.key`).
  - `publico`: Almacena la llave pública (`llavePublica.key`).
- Archivos auxiliares:
  - `tiempos_ejecucion.txt`: Registra tiempos de ejecución para operaciones específicas.
- `OpenSSL-1.1.1h_win32`: Contiene los binarios de OpenSSL necesarios para operaciones criptográficas.

(ii) Instrucciones para correr servidor y cliente, incluyendo cómo configurar el número de clientes concurrentes

Para ejecutar el código hay que compilar y ejecutar la clase App.java. En esta clase se va a desplegar un menú que tiene las siguientes instrucciones:

Figura 1

Menú aplicación



- La opción 1 genera llaves RSA nuevas para el servidor
- La opción 2 ejecuta un n número de clientes de forma concurrente. Esta opción recibe por consola el número de clientes con los que se va a trabajar.
- La opción 3 cambia el número de delegados con los que cuenta el servidor. Esta opción recibe por consola el número de delegados con los que se va a trabajar.
- La opción 4 sirve para probar el caso 1, pues bajo un servidor con un solo delegado manda 32 peticiones del mismo cliente.

Luego de elegir y de que se ejecute la opción aparece el menú nuevamente por si uno quiere probar con alguna opción nueva. En caso de que no se quiera continuar con el programa, hay que oprimir 0.

Cabe aclarar que solo se destaca que el punto 1 y 2 corresponde a la opción 1 y 2 ya que específicamente esas dos hacen referencia a opciones que se piden en el enunciado y no son tanto para hacer pruebas.

(iii) Tablas de datos

La toma de datos se dividió en las siguientes tres partes, siendo las tres el resultado del promedio del tiempo que le tomaba a la aplicación resolver las consultas necesarias:

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

Para esta toma de datos se ejecutó varias veces la opción 4 del menú. Dentro del código se dejó un contador de tiempo para descifrar reto, generar G, P y GX y el tiempo para verificar respuesta. el resultado fue:

Tabla 2

Tiempo promedio descifrar reto

1 Delegado, 1 Cliente , 32 peticiones		
Tiempo promedio Descifrar reto	Tiempo promedio Generar G, P y Gx	Tiempo promedio verificar respuesta
2.6875	3917.25	1.4375

(ii) Un servidor y un cliente que implementen delegados. El número de delegados, tanto servidores como clientes, debe variar entre 1, 4, 8, y 32 delegados concurrentes. Cada cliente genera una sola solicitud.

Para esta toma de datos se ejecutó la opción 3 del menú para variar el número de delegados concurrentes. Así, para cada configuración de delegados, se variaron la

respectiva cantidad de clientes concurrentes que realizaban de a una petición. También se dejó un contador para el tiempo de descifrar reto, generar G, P y GX y el tiempo para verificar respuesta. el resultado fue:

Tabla 2

Tiempo promedio descifrar reto

	Tiempo Promedio Descifrar reto			
	1 Cliente	4 Clientes	8 Clientes	32 Clientes
1 Delegado	49	1.5	1	0.6875
4 Delegados	1	32.875	1.5	72.69791667
8 Delegados	1	0.75	1.5	1.09375
32 Delegados	0	1.25	1.125	8.03125

Tabla 3

Tiempo promedio generar G, P y Gx

	Tiempo Generar G, P y Gx			
	1	4	8	32
1 Delegado	1188	6300.75	3443	3399.75
4 Delegados	6050	4713.625	2368.875	7694.947917
8 Delegados	5913	3881.25	6664.625	5364.03125
32 Delegados	8952	4040.5	5644.375	11832.34375

Tabla 4

Tiempo promedio verificar respuesta

	Tiempo promedio verificar respuesta			
	1	4	8	32
1 Delegado	2	2.5	2.125	1.9375
4 Delegados	2	6	1.625	1.375
8 Delegados	1	1	1.625	1.40625
32 Delegados	1	1.25	1.25	1.375

(iii) Tiempo de cifrado Asimétrico contra el Simétrico.

Para esta toma de datos se modificó el código para tener en cuenta los tiempos de cifrado utilizando ambos tipos de datos.

Figura 2

```java
private void enviarEstadoPaquete(String estado, DiffieHellman dh, IvParameterSpec iv, String clienteId) {
    try {

        // Cifrado simétrico
        long startTimeSimetrico = System.nanoTime();
        byte[] estadoCifrado = Simetrico.cifrar(dh.llaveSimetricaAB1, estado, iv);
        long tiempoSimetrico = System.nanoTime() - startTimeSimetrico;
        System.out.println("Tiempo de cifrado simétrico: " + tiempoSimetrico + " ns");

        // Cifrado asimétrico
        long startTimeAsimetrico = System.nanoTime();
        byte[] estadoCifradoAsimetrico = Asimetrico.cifrar(llavePublica, algoritmo:"RSA", estado);
        long tiempoAsimetrico = System.nanoTime() - startTimeAsimetrico;
        System.out.println("Tiempo de cifrado asimétrico: " + tiempoAsimetrico + " ns");

        // Guardar tiempos en archivo
        guardarTiemposEnArchivo(clienteId, tiempoSimetrico, tiempoAsimetrico);


        // Generar HMAC del estado
        String hmacEstado = Simetrico.generarHMAC(dh.llaveSimetricaAB2, estado);

        // Obtener el cliente desde el mapa
        Cliente cliente = clientesConectados.get(clienteId);
        if (cliente != null) {
            System.out.println("Servidor: Enviando estado cifrado al cliente " + clienteId);
            cliente.recibirEstadoCifrado(estadoCifrado, hmacEstado);
        } else {
            System.err.println("Cliente con ID " + clienteId + " no está conectado.");
        }
    } catch (Exception e) {
        System.err.println("Error al enviar estado cifrado: " + e.getMessage());
    }
}

private synchronized void guardarTiemposEnArchivo(String clienteId, long tiempoSimetrico, long tiempoAsimetrico) {
    String nombreArchivo = "tiempos_ejecucion.txt";  // Un archivo único para todos los clientes
    File archivo = new File(nombreArchivo);
```

El resultado se puede encontrar dentro del archivo tiempos_ejecucion.txt en el repositorio:
figura 3 tiempos de ejecución entre cifrado simetrico y asimetrico

tiempos_ejecucion.txt

```
Cliente 7: Sim◆trico: 216000 ns, Asim◆trico: 229000 ns
Cliente 5: Sim◆trico: 114600 ns, Asim◆trico: 183600 ns
Cliente 4: Sim◆trico: 125600 ns, Asim◆trico: 127200 ns
Cliente 8: Sim◆trico: 141700 ns, Asim◆trico: 136400 ns
Cliente 2: Sim◆trico: 123600 ns, Asim◆trico: 121000 ns
Cliente 28: Sim◆trico: 104500 ns, Asim◆trico: 126900 ns
Cliente 13: Sim◆trico: 89600 ns, Asim◆trico: 101400 ns
Cliente 9: Sim◆trico: 119200 ns, Asim◆trico: 143700 ns
Cliente 31: Sim◆trico: 106700 ns, Asim◆trico: 131300 ns
Cliente 27: Sim◆trico: 450300 ns, Asim◆trico: 188200 ns
Cliente 11: Sim◆trico: 132200 ns, Asim◆trico: 114600 ns
Cliente 6: Sim◆trico: 97500 ns, Asim◆trico: 108600 ns
Cliente 32: Sim◆trico: 100600 ns, Asim◆trico: 120800 ns
Cliente 3: Sim◆trico: 104000 ns, Asim◆trico: 116400 ns
Cliente 30: Sim◆trico: 181900 ns, Asim◆trico: 150500 ns
Cliente 18: Sim◆trico: 131600 ns, Asim◆trico: 167100 ns
Cliente 12: Sim◆trico: 101400 ns, Asim◆trico: 116700 ns
Cliente 29: Sim◆trico: 88700 ns, Asim◆trico: 135200 ns
Cliente 15: Sim◆trico: 94300 ns, Asim◆trico: 106600 ns
Cliente 19: Sim◆trico: 86500 ns, Asim◆trico: 116300 ns
Cliente 14: Sim◆trico: 69500 ns, Asim◆trico: 116600 ns
Cliente 17: Sim◆trico: 105000 ns, Asim◆trico: 112600 ns
Cliente 20: Sim◆trico: 87800 ns, Asim◆trico: 102600 ns
Cliente 1: Sim◆trico: 60100 ns, Asim◆trico: 85300 ns
Cliente 23: Sim◆trico: 105000 ns, Asim◆trico: 117800 ns
Cliente 22: Sim◆trico: 52100 ns, Asim◆trico: 92700 ns
Cliente 10: Sim◆trico: 82700 ns, Asim◆trico: 125400 ns
Cliente 26: Sim◆trico: 85000 ns, Asim◆trico: 131400 ns
Cliente 16: Sim◆trico: 125600 ns, Asim◆trico: 109700 ns
Cliente 21: Sim◆trico: 74300 ns, Asim◆trico: 95100 ns
Cliente 24: Sim◆trico: 63300 ns, Asim◆trico: 76800 ns
Cliente 25: Sim◆trico: 132100 ns, Asim◆trico: 83400 ns
```

De estos archivos obtenemos la siguiente tabla con los tiempos en nanosegundos y sus promedios al final:

Tabla 5

Tiempos cifrados asimetricos vs simetricos

tiempos asimétrico o simétrico		
Cliente	**Tiempo Simétrico (ns)**	**Tiempo Asimétrico (ns)**
Cliente 7	216000	229000
Cliente 5	114600	183600
Cliente 4	125600	127200
Cliente 8	141700	136400
Cliente 2	123600	121000
Cliente 28	104500	126900
Cliente 13	89600	101400
Cliente 9	119200	143700
Cliente 31	106700	131300
Cliente 27	450300	188200
Cliente 11	132200	114600
Cliente 6	97500	108600
Cliente 32	100600	120800
Cliente 3	104000	116400
Cliente 30	181900	150500
Cliente 18	131600	167100
Cliente 12	101400	116700
Cliente 29	88700	135200
Cliente 15	94300	106600
Cliente 19	86500	116300
Cliente 14	69500	116600
Cliente 17	105000	112600
Cliente 20	87800	102600
Cliente 1	60100	85300
Cliente 23	105000	117800
Cliente 22	52100	92700
Cliente 10	82700	125400
Cliente 26	85000	131400
Cliente 16	125600	109700
Cliente 21	74300	95100
Cliente 24	63300	76800

Cliente 25	132100	83400
promedio	**117281.25**	**124715.625**

(iv) Gráficas

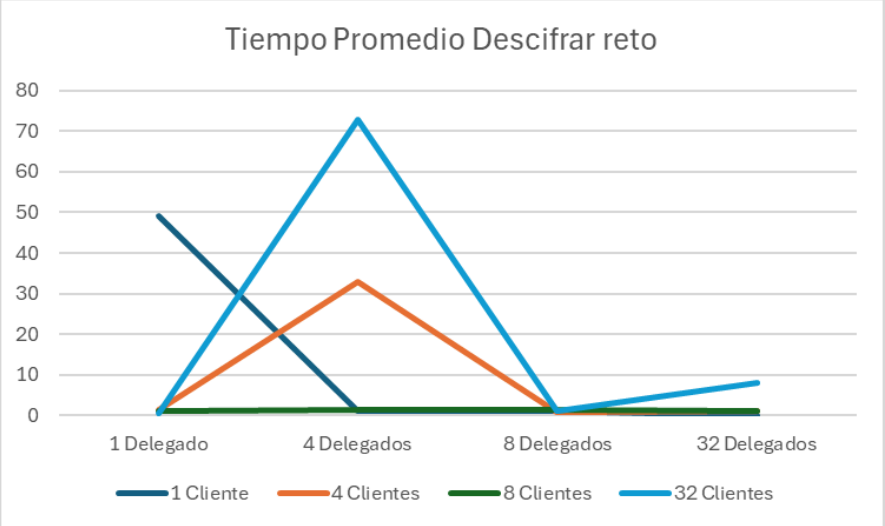
Inicialmente, antes de analizar cada gráfica es importante mencionar que existen algunas fuentes de error que pueden variar los resultados. Por ejemplo, el procesador no siempre está bajo la misma capacidad, por lo que a veces puede llegar a ser más, o menos rápido. Esto se evidencia en los picos que se encuentran de manera inusual, que se podrían considerar como datos atípicos. Sin embargo, el análisis de estas gráficas se va a enfocar en las tendencias generales que se logren ver en conjunto con la lógica detrás de los posibles resultados.

Ya habiendo dicho esto, las gráficas a analizar son:

(i) Una que compare los tiempos para descifrar el reto en los diferentes escenarios

Figura 4

Tiempo promedio Descifrar reto

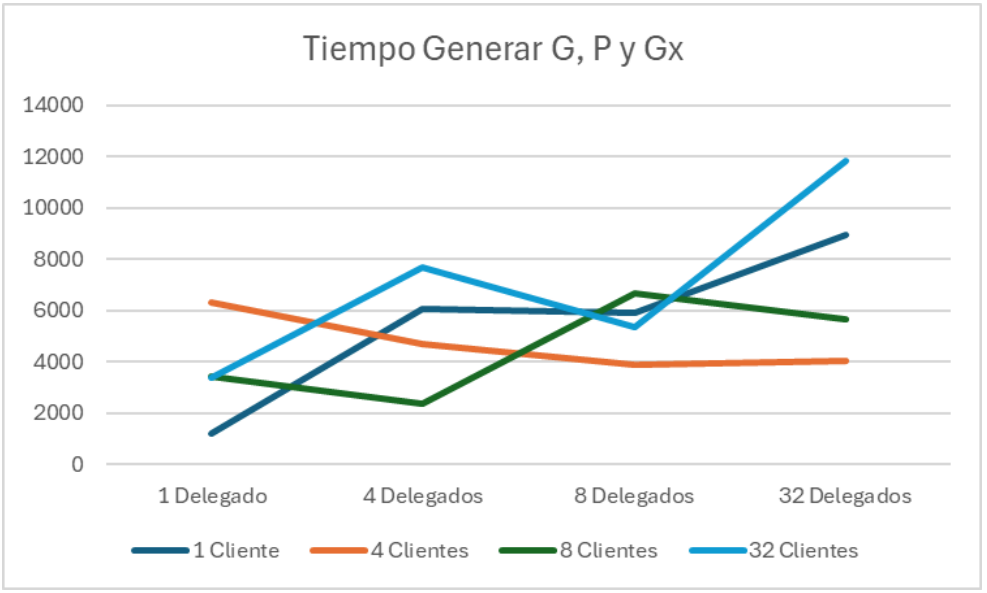


En general se puede evidenciar una dificultad del programa de descifrar altos números de retos de Clientes al tener bajos números de delegados. Esto puede deber a que está recibiendo muchas peticiones en un corto periodo de tiempo.

(ii) Una que compare los tiempos para generar G, P y Gx en los diferentes escenarios

Figura 4

Tiempo promedio generar G, P y Gx

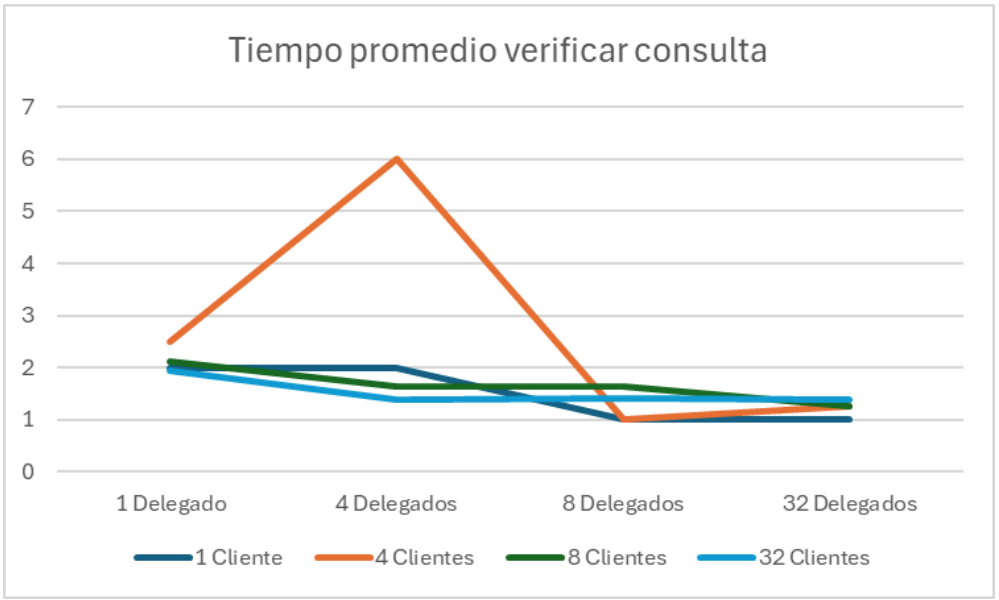


Se evidencia que en la medida que aumentan los clientes, el tiempo en el que se demora en generar las llaves es cada vez mayor. En principio estas diferencias no se notan tanto, pero al aumentar los delegados probablemente se saturen más los servidores de OpenSSL ya que se realizan más peticiones en menos tiempo, lo cual lleva a un peor rendimiento.

(iii) Una que muestre los tiempos para verificar la consulta en los diferentes escenarios

Figura 6

Tiempo promedio verificar respuesta

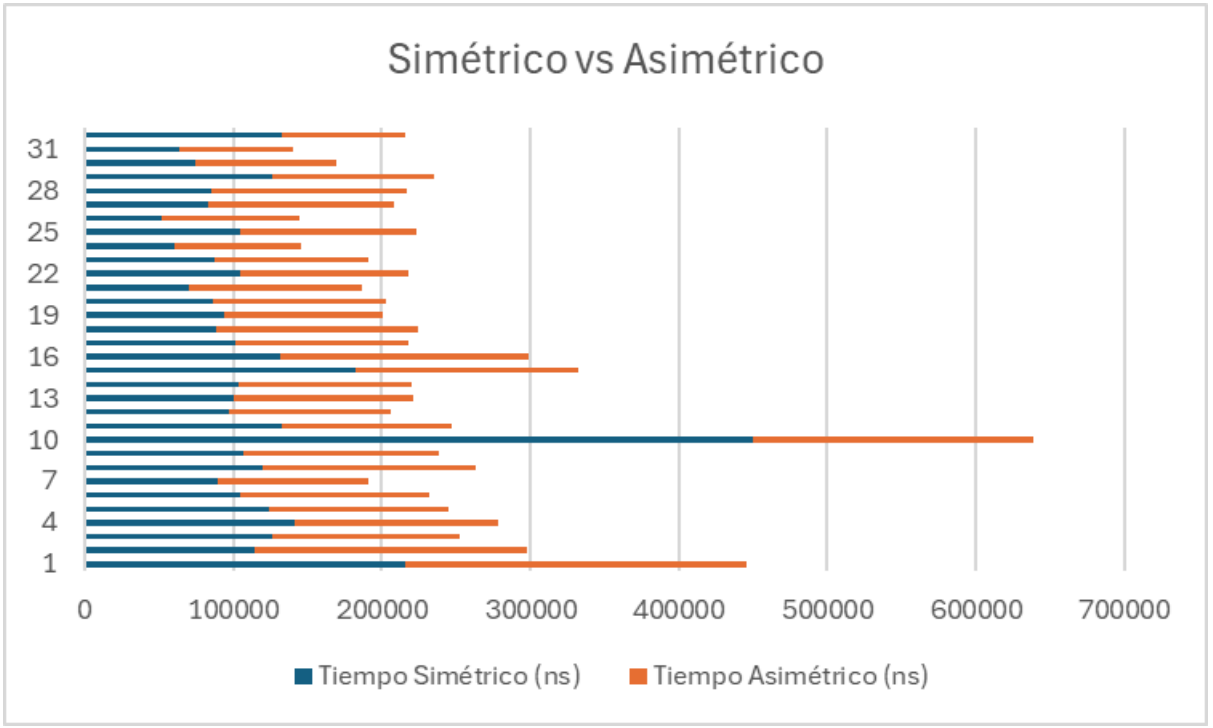


En esta gráfica se evidencia que los tiempos para verificar la consulta tienden a ser lineales. Bien es cierto que existe un pico con 4 delegados y 4 clientes, pero todo parece indicar que es un dato inusual. Con respecto a esta linealidad, esto se puede deber a que se utiliza una técnica de descifrado simétrico que tiende a ser rápido, por lo que no se alcanza a evidenciar fenómenos que afectan el rendimiento, como lo podría ser un cuello de botella.

(iv) Una que muestre los tiempos para el caso simétrico y el caso asimétrico en los diferentes escenarios

Figura 7

Tiempo simétrico contra Asimétrico



La gráfica muestra una comparación entre los tiempos de cifrado simétrico y asimétrico medidos en nanosegundos (ns) para diferentes clientes. En el eje vertical se listan los clientes numerados, mientras que en el eje horizontal se representan los tiempos de cifrado en una escala lineal. Se observa que, en general, el cifrado simétrico es más rápido que el cifrado asimétrico, con tiempos menores para la mayoría de los clientes. Sin embargo, hay ciertas variaciones donde los tiempos de cifrado asimétrico se aproximan o incluso son menores en algunos casos. Esto resalta las diferencias de rendimiento entre ambos métodos de cifrado, donde el simétrico, probablemente, debido a su menor complejidad computacional, ofrece un rendimiento superior en términos de velocidad.

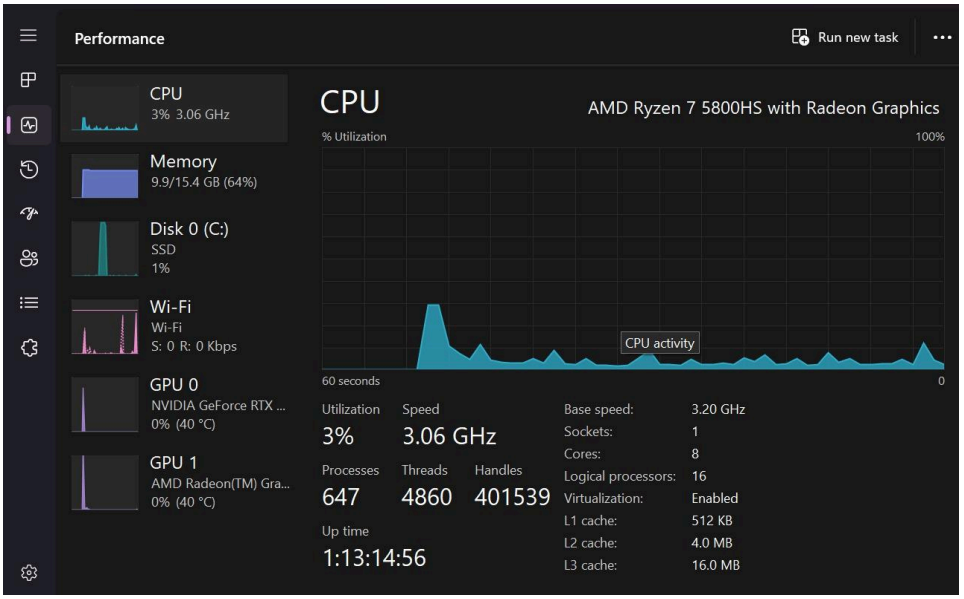
(v) Respuestas a las preguntas planteadas.

Identifique 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

Los cálculos se realizaron usando la siguiente CPU: AMD Ryzen 7 5800H con una velocidad base de 3.20 GHz, 8 cores y 16 procesadores lógicos (CPU-Monkey, n.d.).

Figura 8

Especificaciones del procesador



Para estimar cuántas operaciones de cifrado puede realizar un procesador AMD Ryzen 7 5800HS por segundo, utilizaremos los tiempos promedio de cifrado simétrico y asimétrico junto con las siguientes especificaciones del procesador.

- Modelo: AMD Ryzen 7 5800HS
- Frecuencia base: 3.20 GHz
- Núcleos: 8
- Hilos: 16
- Caché L1: 512 KB
- Caché L2: 4 MB
- Caché L3: 16 MB

También tendremos en cuenta los tiempos promedio de cifrado sacados de la tabla 5:

- Cifrado simétrico: 117,281.25 nanosegundos
- Cifrado asimétrico: 124,715.625 nanosegundos

Según Guía Hardware (2024), el rendimiento de una CPU puede medirse en términos de instrucciones por segundo (IPS) y operaciones de coma flotante por segundo (FLOPS), lo que es esencial para evaluar la capacidad de procesamiento en tareas específicas. (Guía Hardware, 2024)

Así, se realizaron los siguientes cálculos:

La frecuencia base de 3.20 GHz significa que el procesador realiza 3,200,000,000 ciclos por segundo.

Operaciones de cifrado por segundo:
Para determinar cuántas operaciones de cifrado puede realizar el procesador por segundo,

utilizamos la fórmula:
operaciones por segundo = 1/(tiempo promedio por operación (segundos)
Convertimos los tiempos promedio de nanosegundos a segundos:

Cifrado simétrico:
117 281.25 ns=117 281.25×10^−9 s = 0.00011728125 s

operaciones por segundo = 1/ 0.00011728125 s = 8,525 operaciones/ segundo

Cifrado asimétrico:
124 715.625 ns=124 715.625×10^−9 s = 0.000124715625 s

operaciones por segundo = 1/0.000124715625 s = 8018 operaciones/segundo

En conclusión, un procesador AMD Ryzen 7 5800HS puede hacer 8,525 operaciones/ segundo de cifrado simétrico y 8018 operaciones/segundo de cifrado asimétrico, lo cual demuestra que es más rápido hacer el primer tipo de cifrado que el segundo.

(vi) Referencias

CPU-Monkey. (n.d.). *AMD Ryzen 7 5800H*. Retrieved November 6, 2024, from https://www.cpu-monkey.com/en/cpu-amd_ryzen_7_5800h

Guía Hardware. (2024, 22 de enero). *El papel del IPC en las CPUs modernas: descubriendo las claves del rendimiento*. Recuperado el 6 de noviembre de 2024, de https://www.guiahardware.es/ipc-rendimiento-cpu/