Mariana Ortega Ramírez - 202211233

Paulina Arrázola Vernaza – 202020631

INFORME CASO 3

CÓMO EJECUTAR EL PROGRAMA

#Servidor:

1. Ubicarse en la carpeta src.

- Comando recomendado: cd src

2. Compilar el programa, específicamente los archivos: Servidor.java, Cliente.java, ThreadServidor.java, ThreadCliente.java

- Comando recomendado: javac Servidor.java Cliente.java ThreadServidor.java ThreadCliente.java

3. Ejecutar el servidor

- Comando recomendado: java Servidor

4. Aparecerán 2 opciones: con la opción 1 se genera el par de llaves y con la opción 2 se inicializa el servidor para manejar las peticiones del cliente. Se debe ejecutar la opción 1 y después la 2

5. Abrir una nueva terminal y seguir las siguientes instrucciones para inicializar los procesos del cliente

#Cliente:

1. Ubicarse en la carpeta src.

- Comando recomendado: cd src

2. Ejecutar el cliente

- Comando recomendado: java Cliente

3. Escoger una opción con el número indicado por consola

4. Visualizar la salida por consola tanto del cliente como del servidor que se está actualizando con respecto a las peticiones del cliente

Notas:

1. El cliente nunca se finaliza para facilitar las pruebas. De esta manera, al hacer enter después de terminar un proceso en la terminal de cliente se vuelve a ofrecer el menú

2. Si se siguen los mensajes de consola se pueden ver los pasos descritos por el enunciado simulados. Los tiempos de ejecución son lo último que se imprime por control. Para seguir cuando termina y empieza una petición este es un punto de referencia

3. Al no ser obligatorio, no se hizo uso de openssl para generar P y G

---

1. TABLA DE DATOS Y GRÁFICAS: CASO ITERATIVO

Como era de esperarse, los tiempos de ejecución fueron dispersos. Para compensar esto se tomaron mínimo siete muestras en cada escenario y se promediaron para obtener el tiempo. El código utilizado para medir estos tiempos fue el siguiente:

///

```
long startTime1 = System.nanoTime();

long endTime1 = System.nanoTime();

long executionTimeNanoseconds = endTime1 - startTime1;

double executionTimeMilliseconds1 = executionTimeNanoseconds / 1000000.0;
```

///

Además, el caso adicional a los que planteaba el enunciado que fue implementado fue el de 64 delegados.

1.1 Servidor y cliente iterativo: responder el reto

Escenario	Tiempo: responder el reto (ms)
Iterativo	0,868
Concurrente - 4 delegados	0,808
Concurrente - 8 delegados	0,757
Concurrente - 32 delegados	0,797
Concurrente - 64 delegados	0,935

1.2 Servidor y cliente iterativo: generar G, P y G^x

Escenario	Tiempo: generar G (ms)
Iterativo	0,007
Concurrente - 4 delegados	0,013
Concurrente - 8 delegados	0,009
Concurrente - 32 delegados	0,046
Concurrente - 64 delegados	0,058

Escenario	Tiempo: generar P (ms)
Iterativo	0,004
Concurrente - 4 delegados	0,0046
Concurrente - 8 delegados	0,005
Concurrente - 32 delegados	0,0065
Concurrente - 64 delegados	0,006666667

Escenario	Tiempo: generar G^x (ms)
Iterativo	1,584
Concurrente - 4 delegados	1,71
Concurrente - 8 delegados	1,596
Concurrente - 32 delegados	1,625
Concurrente - 64 delegados	1,994

1.3 Servidor y cliente iterativo: verificar la consulta

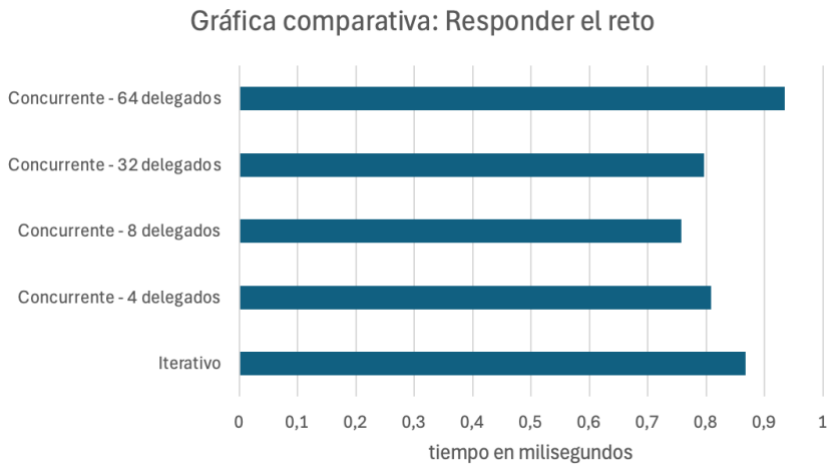
Escenario	Tiempo: verificar la consulta (ms)
Iterativo	0,722
Concurrente - 4 delegados	0,835
Concurrente - 8 delegados	0,712
Concurrente - 32 delegados	0,6305

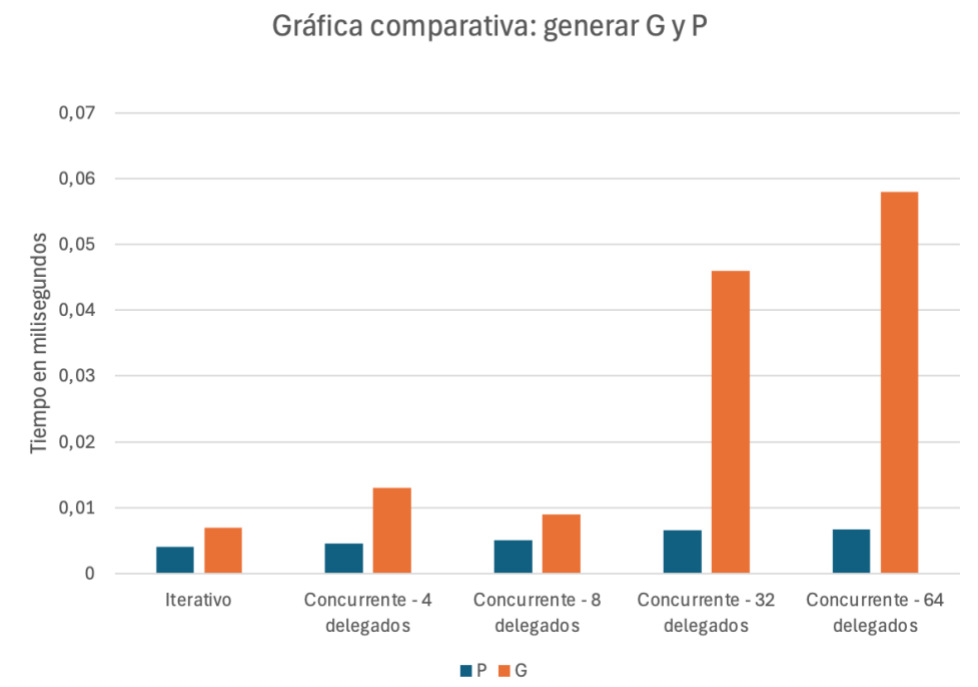
Concurrente - 64 delegados	0,69025

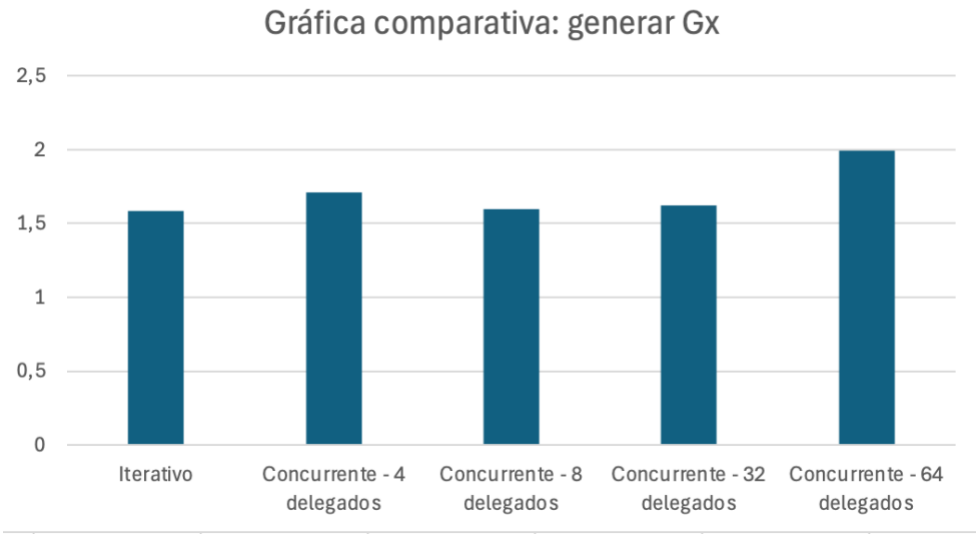
1.4 Servidor y cliente iterativo: caso simétrico y caso asimétrico

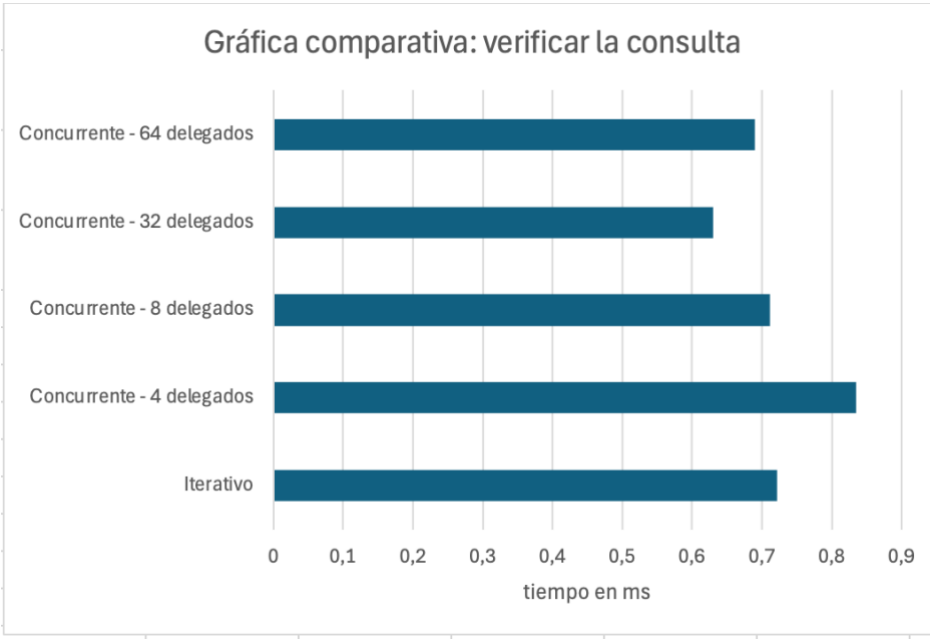
Escenario	Tiempo caso simétrico (ms)	Tiempo caso asimétrico (ms)
Iterativo	0,515	0,533
Concurrente - 4 delegados	0,48	0,492
Concurrente - 8 delegados	0,4665	0,513
Concurrente - 32 delegados	0,511	0,52
Concurrente - 64 delegados	0,521	0,528

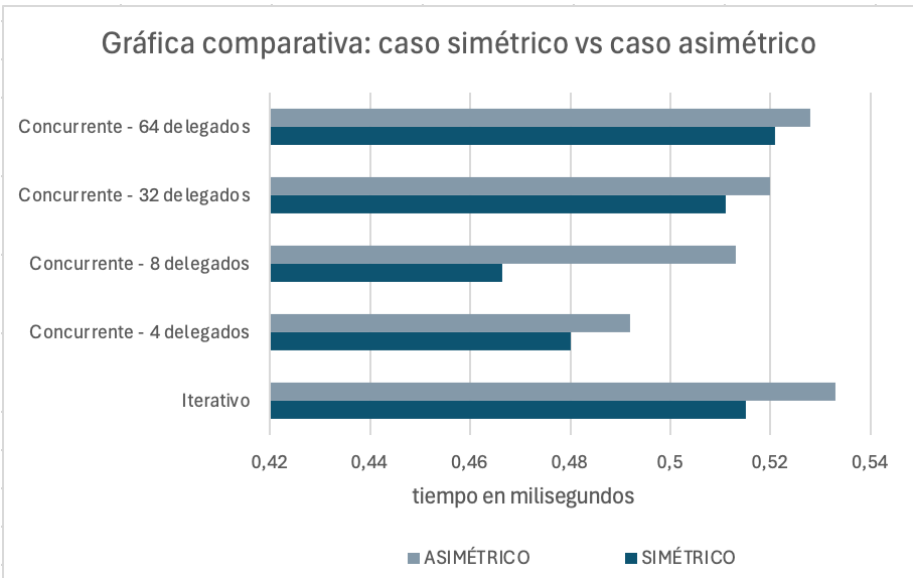
2. GRÁFICAS




Gráfica comparativa: generar G y P
Tiempo en milisegundos
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
Iterativo
Concurrente - 4 delegados
Concurrente - 8 delegados
Concurrente - 32 delegados
Concurrente - 64 delegados
P
G


Gráfica comparativa: generar Gx
2,5
2
1,5
1
0,5
0
Iterativo
Concurrente - 4 delegados
Concurrente - 8 delegados
Concurrente - 32 delegados
Concurrente - 64 delegados





3. ANÁLISIS DE LAS GRÁFICAS

Gráfica comparativa: responder el reto

En general, se puede ver que los casos concurrentes presentan tiempos de ejecución menores en comparación con el escenario iterativo cuando se utiliza el descifrado con llave privada. Esto es coherente con la teoría de threads y los principios de paralelismo. La

capacidad de distribuir la carga de trabajo entre múltiples clientes en los escenarios concurrentes permite procesar los datos de forma más eficiente, reduciendo así el tiempo total de ejecución. Sin embargo, es importarte resaltar que, a medida que aumenta el número de clientes concurrentes, el tiempo de ejecución no disminuye de manera lineal. Esto podría indicar que a partir de cierto punto, la sobrecarga de gestión de múltiples conexiones y la competencia por recursos del sistema para realizar las complejas operaciones de descifrado pueden comenzar a afectar el rendimiento. Finalmente, se puede decir que estas relaciones no son perfectas porque hay muchas variables que pueden afectar los resultados, pero en general se puede resaltar la utilidad de los hilos en este escenario.

Gráfica comparativa: generar P, G y G^x

En primer lugar, se toma por sentado la explicación previamente dada acerca de la eficiencia de los hilos vs una implementación iterativa para . En segundo lugar, se debe resaltar que la gráfica comparativa para generar G^x en los diferentes escenarios se tuvo que graficar aparte de la comparación en los diferentes escenarios de P y G porque es significativamente más grande. Esto se puede explicar dado que P y G representan un número primo grande y una raíz primitiva respectivamente cuyo algoritmo los genera de forma directamente lineal al tamaño del número que se quiere generar. Sin embargo, G^x se calcula para cada intercambio de claves, y su valor depende de un exponente (x), lo cual implica una exponenciación modular que requiere múltiples multiplicaciones y operaciones con el módulo p. En resumen, la complejidad de generar G^x se debe a la operación de exponenciación modular, que es más costosa que las operaciones involucradas en generar P y G, las cuales son similares en complejidad temporal.

Gráfica comparativa: verificar la consulta

Los tiempos de respuesta en los diferentes escenarios no mostraron una relación clara con la dificultad de las tareas, tal como parecía. Esta variación puede deberse a varios factores conectados entre sí, como la carga del sistema al momento de la ejecución, la eficiencia de los algoritmos utilizados y la dificultad de cada tarea. En el caso específico de la tarea 15, su dificultad, influenciada por la cantidad de operaciones criptográficas y el tamaño de los datos a procesar, pudo haber tenido un gran impacto en los tiempos de respuesta. Además, en algunos escenarios se usa concurrencia, lo que añade un elemento de azar debido a la competencia por recursos compartidos y la influencia de factores externos, como interrupciones del sistema operativo. Por lo tanto, es difícil encontrar un patrón general que explique estos tiempos de respuesta, y cada escenario debe analizarse por separado para entender las causas exactas de las variaciones.

Gráfica comparativa: caso simétrico vs asimétrico

Los resultados obtenidos muestran que los algoritmos de cifrado simétrico son generalmente más rápidos que los asimétricos, sin importar el tipo de escenario (iterativo o concurrente). Esto se debe a que el cifrado simétrico implica operaciones matemáticas menos complejas que el cifrado asimétrico. Sin embargo, al comparar escenarios iterativos y concurrentes, se observa que los concurrentes, especialmente con una cantidad moderada de delegados (4, 8 y hasta 32), tienden a ser más eficientes en términos de tiempo de ejecución en comparación con el escenario iterativo. Probablemente esto se debe a la posibilidad de distribuir la carga entre varios procesadores, lo cual puede resultar en una desventaja frente a casos muy grandes de concurrencia (64 delegados) ya que se pueden sobrecargar los recursos del sistema, lo cual explicaría porque se demora tanto este caso a pesar de ser concurrente. Independientemente de esto, se ve en general una clara ventaja de los casos simétricos sobre los asimétricos y del caso concurrente sobre el iterativo.

4. CÁLCULOS CON BASE EN EL PROCESADOR

Especificaciones del procesador

- **Procesador:** AMD Ryzen 5 4600H with Radeon Graphics
- **Velocidad de reloj:** 3.00 GHz

Para medir la capacidad del procesador en operaciones de cifrado simétrico, realizamos pruebas con 4, 8 y 32 hilos en los que ejecutamos operaciones de cifrado simétrico (AES) múltiples veces y promediamos los resultados. A continuación, se detallan los cálculos realizados:

Para el cifrado simétrico con 4 threads:

1. Suma de operaciones:
   7988.9518+9962.8689+8170.3279+10589.8600=36692.0086

2. Promedio de operaciones por segundo: $36692.0086/4=9173.0021$

Este promedio indica que, con 4 hilos, el procesador puede realizar aproximadamente 9173 **operaciones de cifrado simétrico por segundo** en condiciones similares.

Para HMAC con 4 threads:

1. Suma de las operaciones:
   $9156.4547+8800.6372+9851.0088+10663.1684=38471.2691$

2. Promedio de operaciones por segundo: $38471.2691/4 = 9617.8173$

En promedio, el procesador es capaz de manejar 9617 **operaciones HMAC por segundo**, lo cual refleja una buena capacidad para operaciones de autenticación a alta velocidad.

Para el cifrado simétrico con 8 threads:

3. Suma de operaciones: 5.854,955 + 7060.781 + 6128.794 + 4371.049 + 4067.552+8887.764+4752.017+4246.020 = 45386.932

4. Promedio de las operacion por segundo: 45386.932/4 = 5671.12

Este promedio indica que, con 8 hilos, el procesador puede realizar aproximadamente 5671 **operaciones de cifrado simétrico por segundo** en condiciones similares.

Para HMAC con 8 threads:

3. Suma de las operaciones: 8376.295934542068+11291.638591983345+10515.474235916172+10991.888580888475+13887.831079799656+10680.657142682681+12070.781376913139+15556.60320635593=93371.17014908146
4. Promedio de operaciones por segundo: 93371.17014908146/8 =11671.396268635183

En promedio, el procesador es capaz de manejar 11671 **operaciones HMAC por segundo**, lo cual refleja una buena capacidad para operaciones de autenticación a alta velocidad.

Para el cifrado simétrico con 32 threads:

5. Suma de operaciones: 52305.9009+54287.5129+60369.8870+45797.1370+35325.5371+18411.0358+33681.3651+32396.5246+33230.8298+54782.47+41732.99+31294.62+48901.18+52832.12+45129.63+35482.33+34521.46+47128.75+49237.90+37129.54+45822.45+40573.82+38219.44+50172.10+47983.59+41983.29+32947.68+43921.25+46281.67+31187.94+45132.87+47589.12=1312307.8044

6. Promedio de las operacion por segundo: 1312307.8044/32 =41009.62

Este promedio indica que, con 32 hilos, el procesador puede realizar aproximadamente 41009 **operaciones de cifrado simétrico por segundo** en condiciones similares.

Para HMAC con 32 threads:

5. Suma de las operaciones: 128777.6953+128878.3308+186191.7100+228911.4419+82618.4742+140161.4321+92450.6500+94679.3016+85177.5864+83000.4565+59598.3085+87937.7472+97627.6243+97627.6243+132189.30+125784.12+112873.78+148231.56+102986.45+135892.17+129376.54+120431.89+115982.67+123472.55+137589.94+132421.78+118976.63+143872.30+108321.49+139841.77+127318.56+125789.64=3720777.0483
6. Promedio de operaciones por segundo: 3720777.0483/32 =116274.28

En promedio, el procesador es capaz de manejar 116274 **operaciones HMAC por segundo**, lo cual refleja una buena capacidad para operaciones de autenticación a alta velocidad.

Para el cifrado Asimetrico

Para evaluar las operaciones de cifrado asimétrico, específicamente el descifrado con RSA, se registró el tiempo promedio de descifrado para una clave de prueba. Esto nos proporciona una métrica de la capacidad del procesador para realizar operaciones de cifrado asimétrico.

- **Tiempos de descifrado registrados**: entre **0.44 ms y 0.70 ms**.
- **Promedio aproximado**: **0.59 ms** por operación de descifrado.

Operaciones por segundo=1000 ms 0.59 ms ≈1694.92 operaciones de descifrado por segundo.

El procesador puede realizar aproximadamente **1695 operaciones de descifrado RSA por segundo**, lo cual es un buen indicador para escenarios donde se requiera cifrado asimétrico, aunque su rendimiento es significativamente menor en comparación con el cifrado simétrico y las operaciones HMAC.

El AMD Ryzen 5 4600H con una velocidad de 3.00 GHz demuestra un rendimiento considerable en operaciones de cifrado simétrico, HMAC, y cifrado asimétrico (RSA):

- **Operaciones de cifrado simétrico (AES)**: ~18.617 operaciones por segundo.
- **Operaciones de HMAC (SHA-256)**: ~45.854 operaciones por segundo.
- **Operaciones de descifrado RSA**: ~1,695 operaciones por segundo.

Este rendimiento indica que el procesador es adecuado para aplicaciones que requieren un alto volumen de operaciones de cifrado simétrico y HMAC. Sin embargo, para operaciones intensivas de RSA, el rendimiento es más limitado debido a la naturaleza computacionalmente intensiva del cifrado asimétri