**Caso 3 - Infraestructura Computacional**

**Integrantes:**

*Juan Esteban Alvares - 202212030*

*Daniel Felipe Ortiz - 202221234*

### 

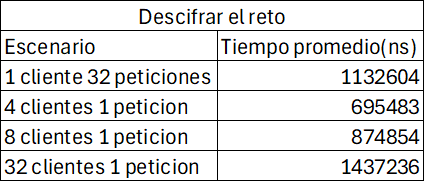
i) Organización de los archivos en el ZIP

El archivo ZIP entregado corresponde a la carpeta src, que contiene cuatro subcarpetas y dos archivos en la raíz.

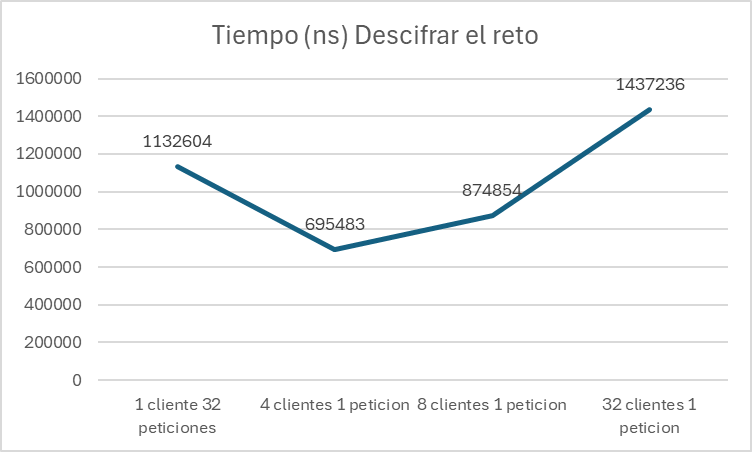
1. Archivo principal (Main.java): Este es el archivo que debe ejecutarse para iniciar el programa. Se encuentra en la raíz de src, visible para todos los componentes. A este mismo nivel también está el archivo K\_w+.txt, el cual contiene la llave pública del servidor. Al ejecutar la opción 1, este archivo se sobreescribe, al igual que el archivo correspondiente a la llave privada.
2. Carpeta server: Esta carpeta contiene toda la lógica relacionada con el servidor. En ella se encuentra el archivo de la llave privada. La clase Server.java es responsable de crear los hilos (Threads) correspondientes a los servidores delegados según el número de usuarios conectados. Por su parte, ThreadServidor.java se encarga del comportamiento de cada hilo delegado, siguiendo el protocolo definido. ProtocoloServidor.java incluye el método procesar() y gestiona el flujo de información en la interacción con el cliente, controlando los estados y respondiendo a los inputs del cliente. Adicionalmente, ProtocoloServidorTiming.java es una variante que incluye mediciones de tiempo para el mismo protocolo.
3. Carpeta client: En esta carpeta se encuentra toda la lógica correspondiente al cliente. La clase Cliente.java se encarga de crear los hilos necesarios, mientras que ProtocoloCliente.java define el comportamiento de cada hilo, coordinando las interacciones del cliente con el servidor.
4. Carpeta cipherLogic: Esta carpeta gestiona toda la lógica de cifrado. Contiene los archivos Simetrico.java y Asimetrico.java, que implementan los algoritmos de cifrado simétrico y asimétrico, respectivamente.
5. Carpeta OpenSSL: Esta carpeta es un paquete instalado que permite la creación de los valores P, G y Gx necesarios para el algoritmo de intercambio de claves de Diffie-Hellman.

2) Basta con correr el programa con Main.java y seguir las instrucciones del menú alfanumérico. En caso de querer medir tiempos, puede dirigirse a la clase ThreadServidor y descomentar la línea 34 y comentar la 33, si decide hacer esto, también debe descomentar el constructor de la clase Simetricos.java y comentar el otro constructor.

3) Tabla 1 - Descifrar el Reto



4) Gráfica 1 - Tiempo vs Descifrar

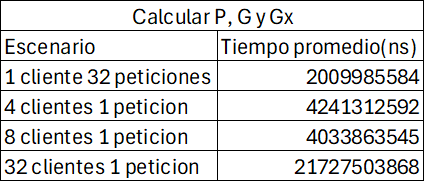


5)

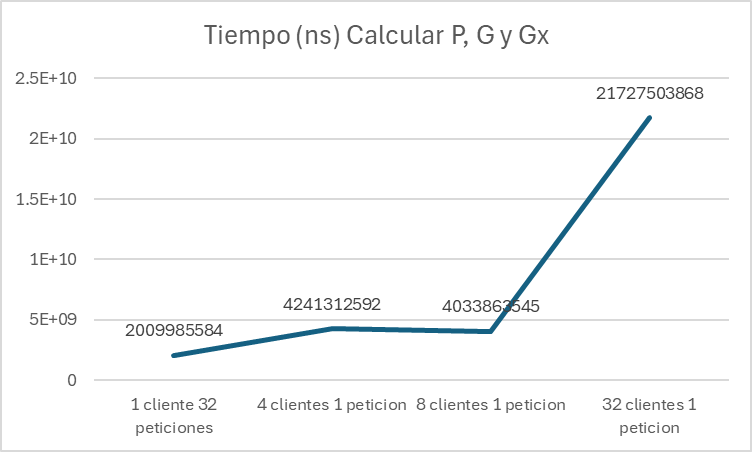
1. **1 cliente con 32 peticiones**: El tiempo de descifrado es de 1,132,604 ns. Esto sugiere que el sistema responde bien a múltiples peticiones desde un único cliente, aunque el tiempo es relativamente alto.
2. **4 clientes con 1 petición cada uno**: Aquí, el tiempo baja significativamente a 695,483 ns, lo cual sugiere que el sistema maneja mejor varias conexiones simultáneas con pocas peticiones que múltiples peticiones desde un solo cliente.
3. **8 clientes con 1 petición cada uno**: El tiempo aumenta a 874,854 ns. Este incremento indica que el sistema comienza a experimentar cierta carga adicional, probablemente debido al manejo concurrente de un mayor número de conexiones.
4. **32 clientes con 1 petición cada uno**: En este caso, el tiempo sube considerablemente a 1,437,236 ns, lo cual sugiere una carga considerable en el sistema, lo que posiblemente indica que el sistema tiene problemas de escalabilidad al manejar un alto número de conexiones simultáneas.

El comportamiento del computador muestra que el rendimiento de descifrado se deteriora con un incremento en la cantidad de clientes concurrentes. Esto podría deberse a limitaciones en el manejo de hilos, capacidad de procesamiento o problemas de contención en recursos compartidos cuando se manejan muchas conexiones simultáneas.

Tabla 2 - Calcular P, G y G^x



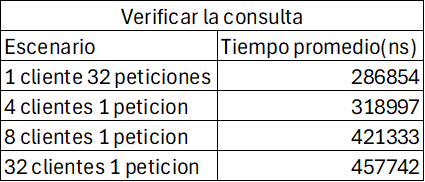
Gráfica 2 - Calcular P, G y G^x



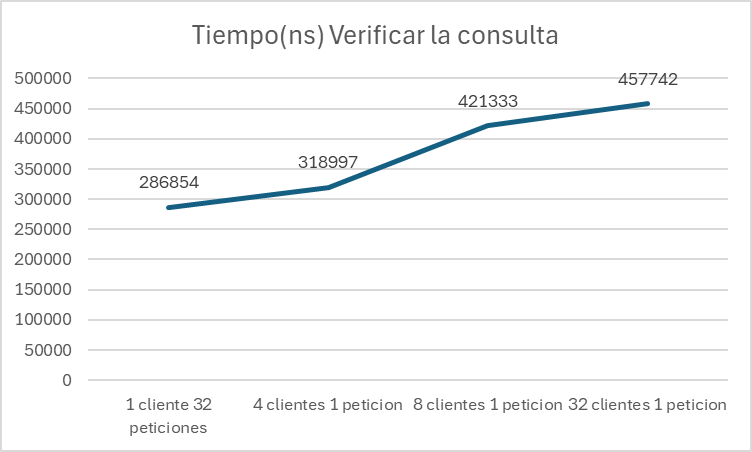
1. **1 cliente con 32 peticiones**: El tiempo es de 2,009,985,584 ns. Esto sugiere que el cálculo es intensivo, pero el sistema parece manejar un número moderado de peticiones concurrentes desde un solo cliente de manera razonable en términos de tiempo.
2. **4 clientes con 1 petición cada uno**: El tiempo se incrementa significativamente a 4,241,312,592 ns, lo cual muestra un aumento notable. Esto podría indicar una mayor carga en la administración de múltiples conexiones, afectando el tiempo de cálculo.
3. **8 clientes con 1 petición cada uno**: El tiempo es similar al caso anterior, con 4,033,865,345 ns, lo cual podría indicar que el sistema alcanza un límite en la optimización y empieza a saturarse.
4. **32 clientes con 1 petición cada uno**: Aquí, el tiempo aumenta drásticamente a 21,727,503,868 ns, lo que refleja una sobrecarga considerable cuando hay un alto número de conexiones concurrentes, resultando en una disminución notable en el rendimiento.

El comportamiento del computador sugiere que el cálculo de P, G, y Gx es altamente intensivo y sensible a la concurrencia. A medida que el número de clientes concurrentes aumenta, el tiempo de procesamiento crece exponencialmente, lo cual podría deberse a limitaciones en el procesamiento paralelo o a contención de recursos en el sistema. Esta tendencia indica que el sistema no escala eficientemente bajo altas demandas concurrentes para este tipo de cálculo, lo que representa un posible cuello de botella en su arquitectura actual.

Tabla 3 - Verificar Consulta



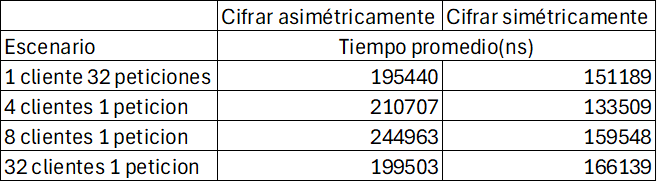
Gráfica 3 - Verificar Consulta



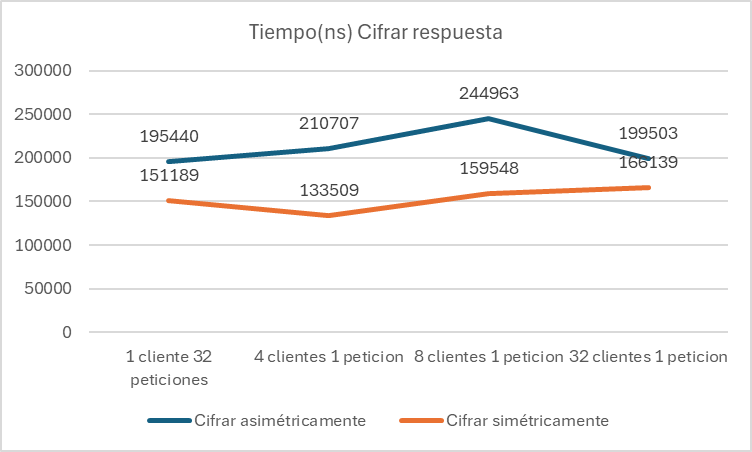
1. **1 cliente con 32 peticiones**: El tiempo es de 286,854 ns. Este valor sugiere que el sistema maneja eficientemente múltiples peticiones secuenciales desde un único cliente, con un tiempo relativamente bajo.
2. **4 clientes con 1 petición cada uno**: El tiempo aumenta a 318,997 ns, lo que indica un ligero incremento al agregar concurrencia. El impacto en el tiempo no es muy significativo, lo cual sugiere que el sistema puede manejar algunas conexiones adicionales sin grandes penalizaciones de rendimiento.
3. **8 clientes con 1 petición cada uno**: El tiempo se incrementa a 421,333 ns, reflejando una carga adicional en el sistema. Aquí se observa un aumento más notorio, probablemente debido a la mayor cantidad de hilos o recursos necesarios para manejar esta concurrencia.
4. **32 clientes con 1 petición cada uno**: El tiempo alcanza 457,742 ns. Este aumento indica que el sistema comienza a experimentar sobrecarga cuando hay muchas conexiones simultáneas, aunque el impacto sigue siendo menor en comparación con otros procesos más intensivos.

El sistema muestra una escalabilidad moderada en la tarea de "Verificar la consulta". Aunque el tiempo de respuesta incrementa con el número de clientes concurrentes, el aumento no es tan drástico como en otros procesos. Esto sugiere que la verificación de consultas es una operación menos intensiva y más optimizada para manejar concurrencia. Sin embargo, al alcanzar altos niveles de concurrencia, como 32 clientes con 1 petición cada uno, el rendimiento comienza a degradarse, lo cual podría representar un límite en la escalabilidad del sistema para este tipo de operación.

Tabla 4 - Cifrar Asimétricamente Y Simétricamente



Gráfica 4 - Cifrar Asimétricamente Y Simétricamente



1. **Cifrado asimétrico**:
   * Los tiempos son consistentemente más altos en comparación con el cifrado simétrico, lo cual es esperado debido a la mayor complejidad computacional del cifrado asimétrico.
   * **1 cliente con 32 peticiones**: Tiempo de 195,440 ns.
   * **4 clientes con 1 petición cada uno**: Incremento a 210,707 ns, lo cual indica una ligera sobrecarga al manejar múltiples conexiones.
   * **8 clientes con 1 petición cada uno**: Máximo de 244,963 ns, lo que sugiere que el sistema enfrenta una carga adicional significativa en este punto.
   * **32 clientes con 1 petición cada uno**: Tiempo de 199,503 ns, una ligera reducción en comparación con 8 clientes, lo cual podría deberse a variaciones en el manejo de concurrencia o a la optimización del sistema bajo carga alta.
2. **Cifrado simétrico**:
   * Los tiempos son consistentemente más bajos, lo que refleja la eficiencia del cifrado simétrico para este tipo de operaciones.
   * **1 cliente con 32 peticiones**: Tiempo de 151,189 ns.
   * **4 clientes con 1 petición cada uno**: Disminución a 133,509 ns, lo que indica que el sistema maneja bien esta configuración.
   * **8 clientes con 1 petición cada uno**: Tiempo de 159,548 ns, mostrando un ligero aumento bajo más concurrencia.
   * **32 clientes con 1 petición cada uno**: Tiempo de 166,139 ns, lo cual es un incremento moderado y manejable en comparación con el cifrado asimétrico.

El cifrado asimétrico resulta ser considerablemente más costoso en términos de tiempo, especialmente al aumentar la cantidad de conexiones concurrentes, lo que podría causar una sobrecarga significativa en el sistema. Por otro lado, el cifrado simétrico muestra una mejor escalabilidad y menor sensibilidad a la concurrencia, lo cual lo convierte en una opción más eficiente en escenarios donde se necesita cifrado rápido para múltiples clientes. Esta diferencia en los tiempos sugiere que, para aplicaciones que requieren una alta concurrencia y menor latencia, el cifrado simétrico sería la opción preferible, mientras que el cifrado asimétrico podría reservarse para casos donde la seguridad adicional justifique el tiempo extra de procesamiento.

**6) Velocidad Procesador:** Tiene 1.6 GHz, significa que el procesador realiza 1.6 × 10^9 ciclos por segundo.

**Para el Cifrado Simétrico**:

**Para el Cifrado Asimétrico**: