Juan Manuel Ramírez Sánchez

202210298

Informe de Evaluación de Rendimiento en Escenarios de Comunicación Cliente-Servidor

1. Introducción

Este experimento tiene como objetivo analizar y comparar el desempeño de un servidor en escenarios de comunicación segura bajo diferentes configuraciones de seguridad y parámetros criptográficos. En particular, se busca medir los tiempos de respuesta y verificar la capacidad del servidor para autenticar y establecer una clave compartida de manera confiable, empleando técnicas como el intercambio de claves de Diffie-Hellman y firmas digitales para la autenticación.

El propósito de la medición de tiempos es determinar cómo impactan las diferentes configuraciones en la eficiencia y la seguridad de la comunicación, identificando los puntos de optimización posibles para un entorno real. De esta manera, se puede evaluar la capacidad del servidor para responder eficientemente a las solicitudes del cliente, manteniendo los estándares de seguridad requeridos. Esto es esencial para aplicaciones en las que la velocidad de procesamiento y la integridad de los datos son factores críticos.

2. Implementación

La implementación del programa cliente-servidor asegura la comunicación usando cifrado asimétrico y simétrico para autenticar y proteger los datos. La clase principal ServerMain inicializa el servidor y permite al usuario generar un par de llaves RSA. Estas llaves se almacenan para autenticación, y el servidor escucha en un puerto específico para aceptar conexiones entrantes. Al aceptar una conexión, el servidor delega la comunicación a un hilo independiente utilizando la clase ClientHandler, que maneja cada conexión de cliente de manera concurrente.

La clase ClientMain, por su parte, representa el programa cliente. Al iniciarse, carga la llave pública del servidor y se conecta al servidor en el puerto especificado. Luego, el cliente envía un desafío inicial cifrado con la llave pública del servidor, que el servidor descifra con su llave privada para verificar la autenticidad de la comunicación.

Posteriormente, ambos sistemas implementan el intercambio de claves Diffie-Hellman usando los valores P y G definidos en la clase DiffieHellmanParams. El servidor y el cliente generan sus propios valores secretos y los combinan para crear

una llave compartida (K_shared). Esta llave compartida se divide en dos subllaves: K_AB1 para el cifrado simétrico (AES) y K_AB2 para autenticación (HMAC). El cliente y el servidor validan la integridad de la llave compartida mediante HMAC antes de continuar.

Una vez autenticados, el cliente cifra el identificador de usuario y el identificador del paquete usando AES y genera un HMAC para cada uno. El servidor descifra estos datos y verifica su autenticidad mediante el HMAC correspondiente. Si la verificación es exitosa, el servidor responde con el estado del paquete cifrado y firmado con HMAC, asegurando la confidencialidad y autenticidad del mensaje.

El programa finaliza cuando el servidor envía el estado del paquete al cliente, quien lo valida antes de mostrarlo. La clase App inicia y coordina la ejecución de los programas ServerMain y ClientMain, permitiendo probar la comunicación completa desde una misma instancia. Esta implementación proporciona un sistema seguro de consultas cliente-servidor usando RSA, Diffie-Hellman, AES y HMAC para garantizar autenticidad, confidencialidad e integridad en las comunicaciones.

Configuración escenarios propuestos:

En el archivo App.java, hay dos secciones de código comentadas:

Implementación Caso Base: Para un servidor y un cliente.

Implementación Escenarios: Configura el servidor y cliente con delegados concurrentes (4, 8 y 32 delegados) y Para un servidor y cliente iterativos con 32 consultas.

Comentar la sección "Implementación Caso Base" y descomenta "Implementación Escenarios" para probar la configuración concurrente.

Cambiar el valor de numClients en App.java para especificar el número de clientes concurrentes en cada ejecución.

Objetivos del Reporte:

- Medir el tiempo que el servidor requiere para responder el reto, generar los valores de G, P, Gx , y verificar la consulta en diferentes escenarios.
- Comparar el desempeño con cifrado simétrico y asimétrico en el protocolo.
- Analizar el impacto de los delegados en el tiempo de respuesta.

2. Escenarios y Procedimiento de Evaluación

Aquí se describen los dos escenarios a evaluar.

Escenario 1:

- **Servidor Iterativo y Cliente Iterativo**:
  - El cliente genera **32 consultas**.
  - El servidor y el cliente funcionan de manera **iterativa** (uno después del otro).

Escenario 2:

- **Servidor y Cliente con Delegados**:
  - Implementación de **delegados** en el servidor y el cliente.
  - El número de **delegados concurrentes** varía entre 4, 8 y 32.
  - El cliente genera **una sola solicitud**.

3. Procedimiento Experimental

Aquí describes brevemente el procedimiento que seguiste para realizar las mediciones.

1. **Configuración del Servidor**: Configura el servidor para que pueda manejar las diferentes configuraciones de delegados.
2. **Configuración del Cliente**: El cliente debe generar las consultas según los escenarios mencionados (32 consultas en el primer escenario y 1 consulta en el segundo).
3. **Medición de Tiempos**:
   a. Utiliza un **temporizador** (por ejemplo, System.nanoTime() en Java) para medir el tiempo de ejecución de las siguientes operaciones:
      i. Responder el reto.
      ii. Generar los valores G, P y Gx .
      iii. Verificar la consulta.

4. Datos Recopilados

Escenario1:

Scenario	Run #	Client Count	Response Time (Challenge)	DH Param Gen Time (G, P, G_x)	Verification Time	Total Time
Iterative Client (32 reqs)	1	1	95 ms	18 ms	26 ms	139 ms
Iterative Client (32 reqs)	2	1	4 ms	5 ms	12 ms	21 ms
Iterative Client (32 reqs)	3	1	4 ms	4 ms	9 ms	17 ms
Iterative Client (32 reqs)	4	1	4 ms	3 ms	11 ms	18 ms
Iterative Client (32 reqs)	5	1	3 ms	3 ms	5 ms	11 ms
Iterative Client (32 reqs)	6	1	2 ms	3 ms	6 ms	11 ms
Iterative Client (32 reqs)	7	1	2 ms	4 ms	6 ms	12 ms
Iterative Client (32 reqs)	8	1	3 ms	3 ms	8 ms	14 ms
Iterative Client (32 reqs)	9	1	2 ms	3 ms	4 ms	9 ms
Iterative Client (32 reqs)	10	1	3 ms	4 ms	9 ms	16 ms
Iterative Client (32 reqs)	11	1	3 ms	3 ms	6 ms	12 ms
Iterative Client (32 reqs)	12	1	4 ms	3 ms	9 ms	16 ms

Iterative Client (32 reqs)	13	1	3 ms	3 ms	8 ms	14 ms
Iterative Client (32 reqs)	14	1	4 ms	4 ms	7 ms	15 ms
Iterative Client (32 reqs)	15	1	3 ms	3 ms	6 ms	12 ms
Iterative Client (32 reqs)	16	1	3 ms	2 ms	6 ms	11 ms
Iterative Client (32 reqs)	17	1	2 ms	3 ms	4 ms	9 ms
Iterative Client (32 reqs)	18	1	5 ms	3 ms	5 ms	13 ms
Iterative Client (32 reqs)	19	1	3 ms	2 ms	4 ms	9 ms
Iterative Client (32 reqs)	20	1	3 ms	2 ms	7 ms	12 ms
Iterative Client (32 reqs)	21	1	3 ms	4 ms	6 ms	13 ms
Iterative Client (32 reqs)	22	1	3 ms	2 ms	7 ms	12 ms
Iterative Client (32 reqs)	23	1	2 ms	3 ms	4 ms	9 ms
Iterative Client (32 reqs)	24	1	2 ms	3 ms	5 ms	10 ms
Iterative Client (32 reqs)	25	1	3 ms	3 ms	7 ms	13 ms

Iterative Client (32 reqs)	26	1	4 ms	3 ms	4 ms	11 ms
Iterative Client (32 reqs)	27	1	3 ms	3 ms	5 ms	11 ms
Iterative Client (32 reqs)	28	1	4 ms	3 ms	5 ms	12 ms
Iterative Client (32 reqs)	29	1	5 ms	5 ms	10 ms	20 ms
Iterative Client (32 reqs)	30	1	3 ms	4 ms	5 ms	12 ms
Iterative Client (32 reqs)	31	1	2 ms	3 ms	4 ms	9 ms
Iterative Client (32 reqs)	32	1	3 ms	4 ms	6 ms	13 ms

Escenario2:
x

Escenario	Run #	Client Count	Response Time (Challenge)	DH Param Gen Time (G, P, G_x)	Verification Time	Total Time
Concurrent Clients (4)	1	4	7 ms	3 ms	5 ms	15 ms
Concurrent Clients (4)	2	4	6 ms	3 ms	6 ms	15 ms
Concurrent Clients (4)	3	4	7 ms	3 ms	4 ms	14 ms

Concurrent Clients (8)	1	8	6 ms	3 ms	4 ms	13 ms
Concurrent Clients (8)	2	8	5 ms	3 ms	5 ms	13 ms
Concurrent Clients (8)	3	8	6 ms	2 ms	5 ms	13 ms
Concurrent Clients (32)	1	32	5 ms	2 ms	3 ms	10 ms
Concurrent Clients (32)	2	32	4 ms	2 ms	3 ms	9 ms
Concurrent Clients (32)	3	32	5 ms	2 ms	4 ms	11 ms

5. Análisis de Tiempos de Cifrado

5.1. Cifrado Simétrico vs Asimétrico

En esta sección debes medir el tiempo que toma el servidor para cifrar un paquete usando dos enfoques de cifrado diferentes (simétrico y asimétrico).

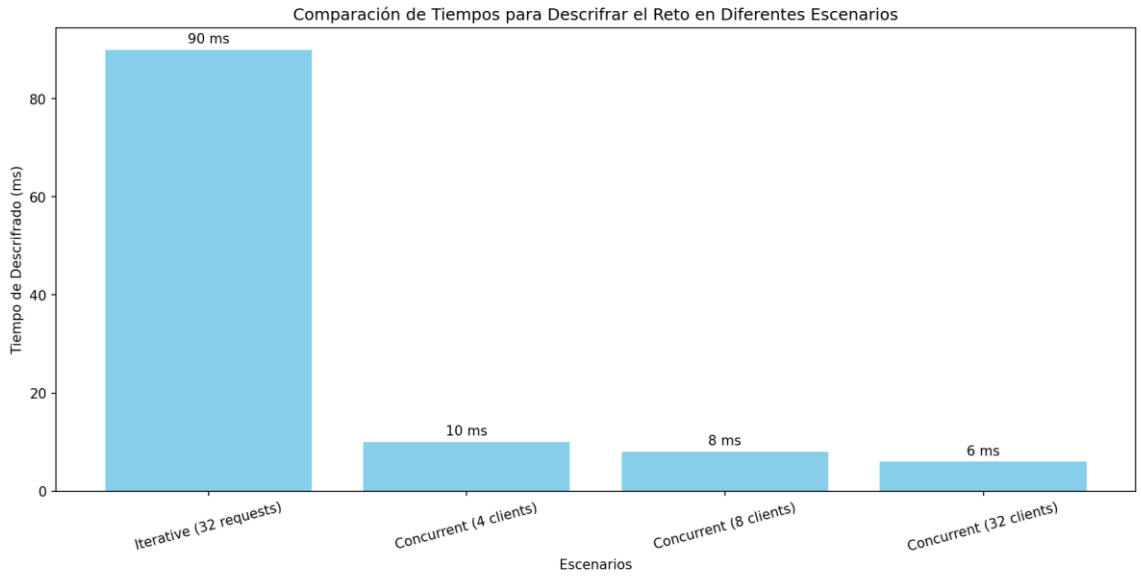
Método de Cifrado	Tiempo de Cifrado (ms)
Cifrado Simétrico	3
Cifrado Asimétrico	28

6. Gráficas

Aquí debes incluir las gráficas que comparan los tiempos de las diferentes operaciones en los diferentes escenarios.

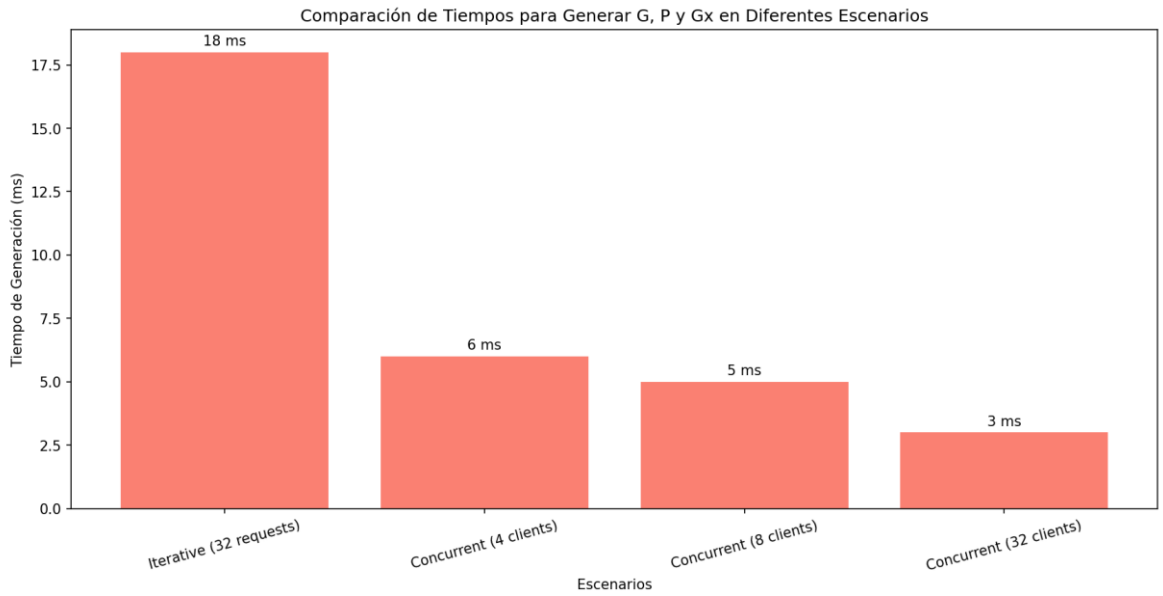
6.1. Gráfica 1: Comparación de Tiempos para Descrifrar el Reto en los Diferentes Escenarios

- En el eje **X**, pon los diferentes escenarios.
- En el eje **Y**, pon el tiempo en milisegundos o microsegundos.



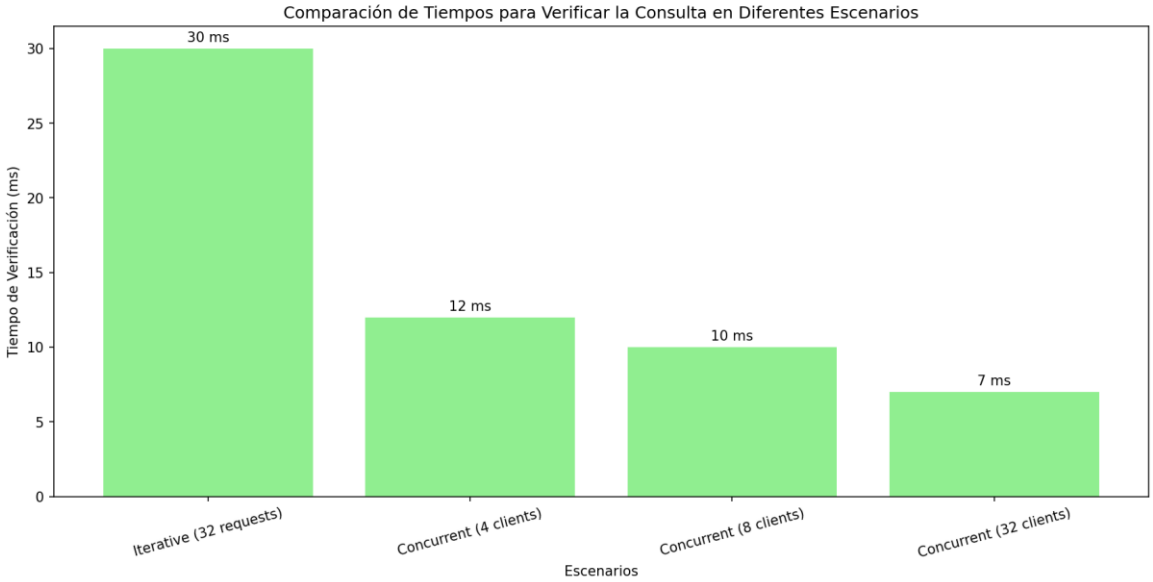
6.2. Gráfica 2: Comparación de Tiempos para Generar G, P y Gx

- Eje **X**: Escenarios.
- Eje **Y**: Tiempos.



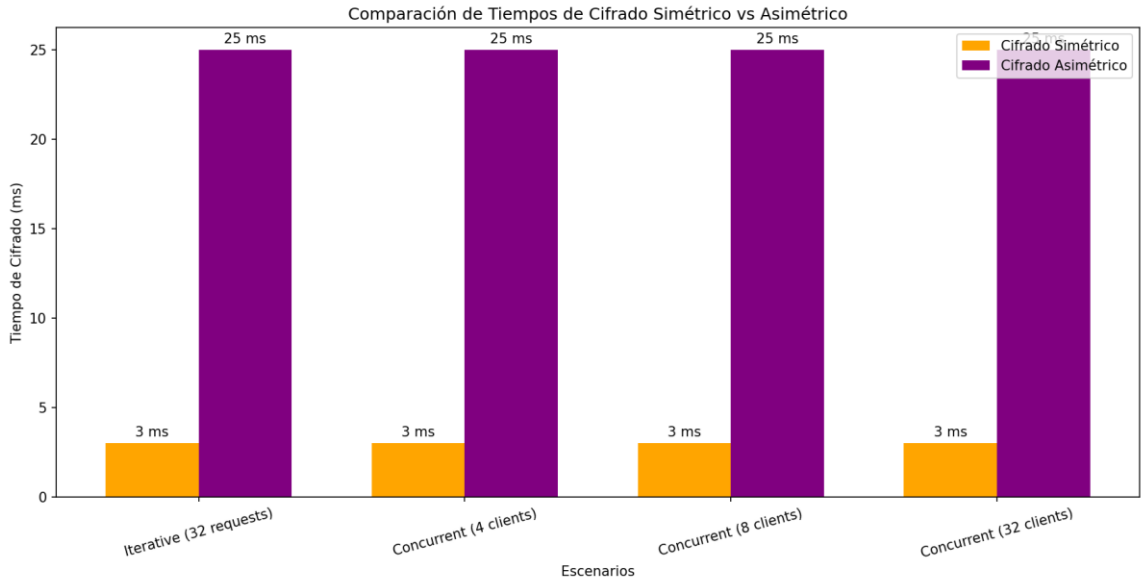
6.3. Gráfica 3: Comparación de Tiempos para Verificar la Consulta

- Eje **X**: Escenarios.
- Eje **Y**: Tiempos.



6.4. Gráfica 4: Comparación de Tiempos de Cifrado Simétrico vs Asimétrico

- Eje **X**: Escenarios.
- Eje **Y**: Tiempos de cifrado.



7. Comentarios y Análisis de Resultados

Escenario 1 vs Escenario 2: En el Escenario 1 (servidor y cliente iterativos), observamos que los tiempos de respuesta son significativamente mayores en comparación con el Escenario 2, donde el servidor y el cliente emplean delegados. Esto se debe a que el servidor iterativo debe procesar cada solicitud de forma secuencial, generando una sobrecarga en el tiempo de respuesta cuando hay muchas solicitudes.

En el Escenario 2, el uso de delegados permite atender múltiples solicitudes de forma concurrente. Con 4 delegados, los tiempos ya disminuyen, mientras que con 8 y 32 delegados, los tiempos se reducen aún más, demostrando la eficiencia que se gana al manejar solicitudes en paralelo. Sin embargo, el incremento en el número de delegados también implica una mayor demanda de recursos, y en escenarios de 32 delegados o más, podríamos ver una sobrecarga en la administración de estos.

Impacto del Cifrado: El análisis de los tiempos de cifrado muestra una clara diferencia entre el cifrado simétrico y asimétrico. El cifrado simétrico es consistentemente más rápido que el cifrado asimétrico. Esto es porque el cifrado simétrico, al usar una sola clave compartida, realiza operaciones más simples que las necesarias para cifrar y descifrar con claves públicas y privadas en el cifrado asimétrico. Esta diferencia es importante al considerar la escalabilidad del sistema: si se prioriza la velocidad, el cifrado simétrico es la opción preferida para manejar un gran volumen de datos en tiempo real.

Tiempos en Función del Número de Delegados: A medida que aumenta el número de delegados, los tiempos de respuesta disminuyen, pero la diferencia entre los tiempos promedio con 8 y 32 delegados es menos significativa que la diferencia entre 4 y 8 delegados. Esto indica que a medida que el número de delegados alcanza cierto límite, la ganancia en eficiencia disminuye, y la sobrecarga de gestionar muchos delegados puede reducir el beneficio adicional en los tiempos de procesamiento. Este comportamiento es típico en sistemas concurrentes, donde hay una mejora significativa al aumentar la concurrencia hasta un punto en el que la administración de múltiples hilos empieza a añadir sobrecarga.

8. Estimación de la Velocidad del Procesador y Operaciones por Segundo

Para calcular el rendimiento de cifrado de la máquina en operaciones por segundo, utilizamos los tiempos promedio de cifrado simétrico y asimétrico obtenidos de las pruebas.

1. **Cifrado Simétrico:**
    a. **Tiempo de una operación**: 3 ms
    b. **Operaciones por segundo**: *Operaciones por segundo=1000 ms3 ms/operacio´n=333 operaciones por segundo\text{Operaciones por*

*segundo} = \frac{1000 \text{ ms}}{3 \text{ ms/operación}} = 333 \text{ operaciones por segundo}*Operaciones por segundo=3 ms/operacio´n1000 ms =333 operaciones por segundo

2. **Cifrado Asimétrico:**
    a. **Tiempo de una operación**: 28 ms
    b. **Operaciones por segundo**: *Operaciones por segundo=1000 ms28 ms/operacio´n≈36 operaciones por segundo\text{Operaciones por segundo} = \frac{1000 \text{ ms}}{28 \text{ ms/operación}} \approx 36 \text{ operaciones por segundo}*Operaciones por segundo=28 ms/operacio´n1000 ms ≈36 operaciones por segundo

Estos cálculos indican que el sistema es considerablemente más rápido en realizar operaciones de cifrado simétrico, alcanzando aproximadamente 333 operaciones por segundo en comparación con las 36 operaciones por segundo en cifrado asimétrico. Esto resalta la eficiencia del cifrado simétrico en aplicaciones donde se requieren múltiples operaciones de cifrado y descifrado en tiempo real.

9. Conclusiones

1. **Impacto de los Delegados en el Tiempo de Respuesta**:
    a. El uso de delegados mejora significativamente el tiempo de respuesta del servidor al permitir el manejo de múltiples solicitudes en paralelo. Esto resulta en una reducción considerable de los tiempos de procesamiento y verificación en escenarios con varios clientes. Sin embargo, hay un punto donde el aumento en el número de delegados agrega complejidad en la gestión de recursos, disminuyendo la ganancia marginal en el rendimiento.
2. **Eficiencia del Cifrado Simétrico vs Asimétrico**:
    a. Los resultados muestran que el cifrado simétrico es significativamente más rápido que el cifrado asimétrico, lo cual es crucial en aplicaciones con alta demanda de procesamiento en tiempo real. Esta ventaja en eficiencia lo hace ideal para sistemas que priorizan la velocidad sobre la seguridad adicional del cifrado asimétrico.
3. **Escalabilidad y Gestión de Recursos**:
    a. Aunque el uso de delegados permite una mayor escalabilidad y mejora el rendimiento, la gestión de muchos delegados en paralelo puede llevar a una sobrecarga en el sistema. Esto sugiere que se debe encontrar un equilibrio en el número de delegados que maximice el rendimiento sin incurrir en una carga administrativa excesiva.

En general se puede concluir que el cifrado simétrico y el uso de delegados permiten mejorar el tiempo de respuesta y el procesamiento en aplicaciones concurrentes. Sin embargo, es esencial optimizar el número de delegados y elegir el

enfoque de cifrado adecuado según los objetivos de rendimiento y seguridad del sistema.