CASO #3 INFRAESTRUCTURA COMPUTACIONAL

Integrantes:

- Danny Muñoz Sanabria 202215511
- Juan Pablo Barón 202210502
- Nicolas Casas Ibarra 202212190

DESARROLLO

1. Para responder las siguientes preguntas puede usar fuentes externas, solo recuerde construir las respuestas con sus propias palabras y citar las fuentes usadas:
    - En el protocolo descrito el cliente conoce la llave pública del servidor (K_w+). ¿Cuál es el método comúnmente usado para obtener estas llaves públicas para comunicarse con servidores web?

      La llave pública del servidor se usa ampliamente en la comunicación segura a través de SSL/TLS, que son protocolos para asegurar la transmisión de datos en internet. Durante el llamado "handshake" o saludo inicial de TLS, el servidor comparte su certificado SSL, que incluye su llave pública, con el cliente. El cliente utiliza esta llave pública para cifrar información, como las claves de sesión, que sólo el servidor puede descifrar usando su correspondiente llave privada. Este proceso ayuda a establecer un canal seguro sin que se expongan las claves de cifrado a interceptaciones. Las llaves de sesión generadas durante el handshake son luego usadas para cifrar y descifrar toda la comunicación entre el cliente y el servidor (Connect, Protect and Build Everywhere) (Connect, Protect and Build Everywhere) (DigiCert).

    - ¿Por qué es necesario cifrar G y P con la llave privada?

      En la práctica estándar de protocolos criptográficos como Diffie-Hellman, G (el generador) y P (el módulo) generalmente son parámetros públicos que no necesitan ser cifrados. No obstante, en ciertos escenarios específicos, podría ser necesario cifrar estos valores para proteger información adicional derivada o configuraciones específicas. Por ejemplo, si la configuración de G y P es única o adaptada a un entorno particular, revelar estos valores podría comprometer la seguridad de todo el sistema criptográfico, permitiendo ataques dirigidos que exploten conocimientos específicos de estos parámetros. En entornos altamente seguros o personalizados, ocultar estos detalles puede ser crucial para mantener la integridad y la seguridad general del sistema.

      Además, existen razones de seguridad adicionales para cifrar G y P, particularmente para proteger contra ataques de tipo "man-in-the-middle" (MitM) y para la autenticación mediante firma digital:

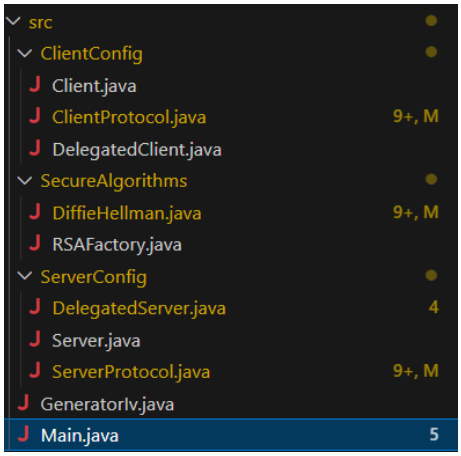
- ***Protección contra ataques MitM:*** En estos ataques, un intruso intercepta y potencialmente altera la comunicación entre dos partes. Cifrando G y P con la llave privada del servidor, se asegura que estos parámetros no sean interceptados o modificados sin autorización durante su transmisión. Esto es vital cuando la configuración de G y P no es estándar y su divulgación podría permitir a un atacante replicar o interferir en la creación de la clave compartida.
- ***Autenticación mediante firma digital:*** Cifrar datos con la llave privada del servidor actúa como una firma digital, permitiendo al cliente verificar que los datos provienen auténticamente del servidor y no han sido alterados. Este uso de la llave privada para cifrar información sensible como G y P asegura la autenticidad y la integridad de la comunicación.

- El protocolo Diffie-Hellman garantiza "Forward Secrecy", presente un caso en el contexto del sistema Banner de la Universidad donde sería útil tener esta garantía, justifique su respuesta (por qué es útil en ese caso).

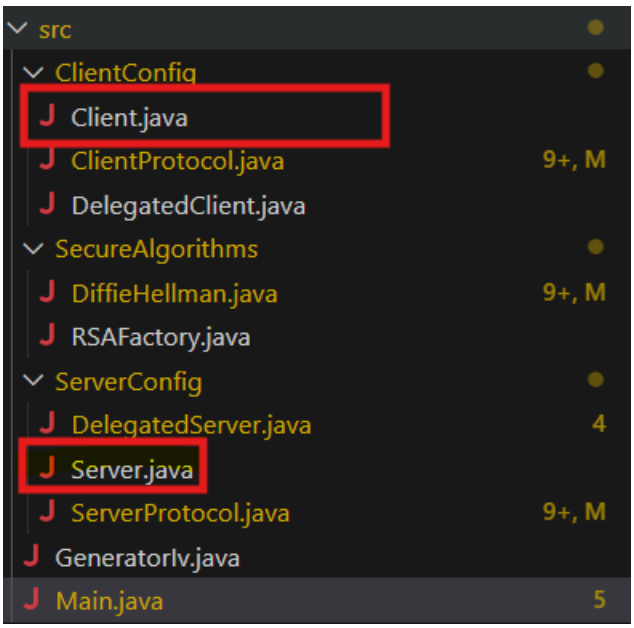
  La "Forward Secrecy" es especialmente beneficiosa para sistemas como Banner en universidades, que gestionan y almacenan grandes volúmenes de información personal y académica sensible. Esta característica del protocolo Diffie-Hellman garantiza que cada sesión de comunicación tiene claves únicas, lo cual es crucial por varias razones:

  - ***Protección de datos sensibles:*** En el sistema Banner, que incluye información como calificaciones, historiales académicos y datos personales de los estudiantes, la "Forward Secrecy" asegura que la exposición de una clave privada del servidor no comprometa la seguridad de las sesiones de comunicación anteriores. Esto significa que incluso si un atacante obtiene acceso futuro a la clave privada, no podría descifrar comunicaciones pasadas, protegiendo así los datos transmitidos anteriormente.
  - ***Mitigación de futuras vulnerabilidades:*** La tecnología y los métodos de ataque están constantemente evolucionando, lo que puede llevar a la futura vulnerabilidad de los algoritmos criptográficos utilizados hoy. La "Forward Secrecy" reduce el riesgo de que futuras debilidades en estos algoritmos expongan datos históricos, ya que las claves de cada sesión son efímeras y no se basan en la clave de cifrado a largo plazo del servidor.

Las pruebas se realizaron con un pool de threads por lo tanto se obtenian varios valores para cada escenario planteado. En la siguiente captura se muestra la clase en la que se implementa el pool de threads.



Para poder observar la interacción cliente-servidor con más detalle se puede correr el archivo server.java y el de client.java. Como se simulan dos máquinas y se comunican a través de sockets hay que ejecutar ambos programas al mismo tiempo.



2. Corra su programa en los diferentes escenarios y mida el tiempo que el cliente demora para:
    - Verificar la firma
    - Calcular G^y
    - Cifrar la consulta
    - Generar el código de autenticación

Para las pruebas realizadas se obtuvo dos valores, el tiempo promedio en que tardaba en ejecutarse cada instrucción para cada cliente y la sumatoria total de todos los tiempos de ejecución de las instrucciones para la cantidad de clientes delegados.

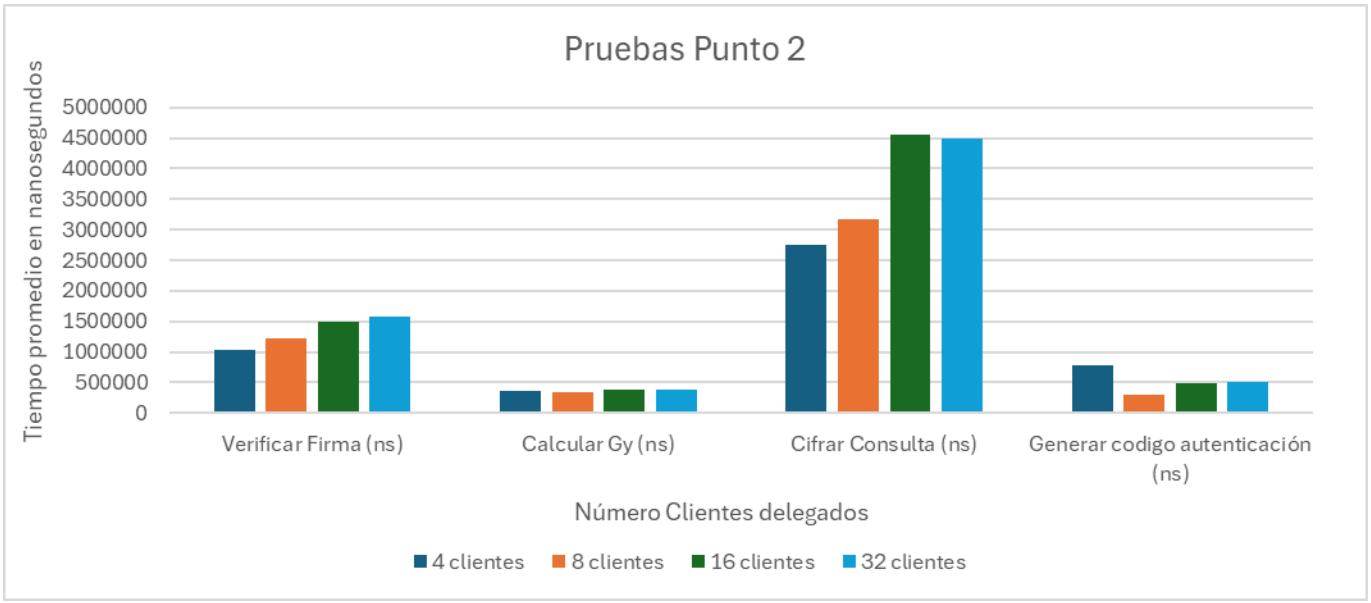
	Verificar Firma (ns)		Calcular Gy (ns)		Cifrar Consulta (ns)		Generar codigo autenticación (ns)	
#clientes delegados	Suma total (ns)	promedio (ns)	Suma total (ns)	promedio (ns)	Suma total (ns)	promedio (ns)	Suma total (ns)	promedio (ns)
4	4156625	1039156,25	1413760	353440	10977885,72	2744471,43	3126285,72	781571,43
8	9779535,52	1222441,94	2662666,64	332833,33	25197341,36	3149667,67	2326971,28	290871,41
16	23995800	1499737,5	6229600	389350	72977400	4561087,5	7934857,12	495928,57
32	50613127,36	1581660,23	12141600	379425	143445995,5	4482687,36	16187108,48	505847,14

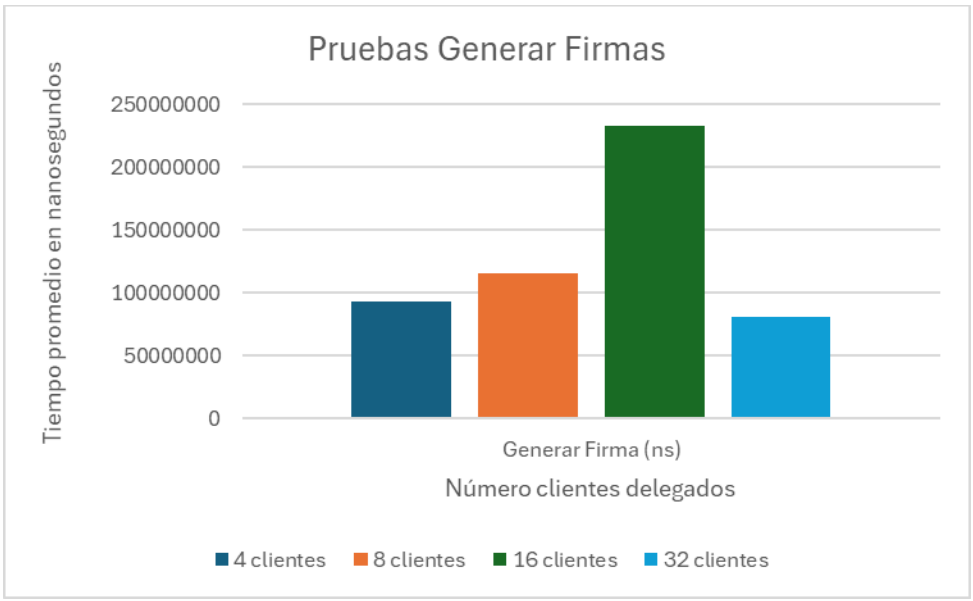
3. En los mismos escenarios considerados en el punto anterior mida el tiempo que el servidor demora para:
    - Generar la firma
    - Descifrar la consulta
    - Verificar el código de autenticación

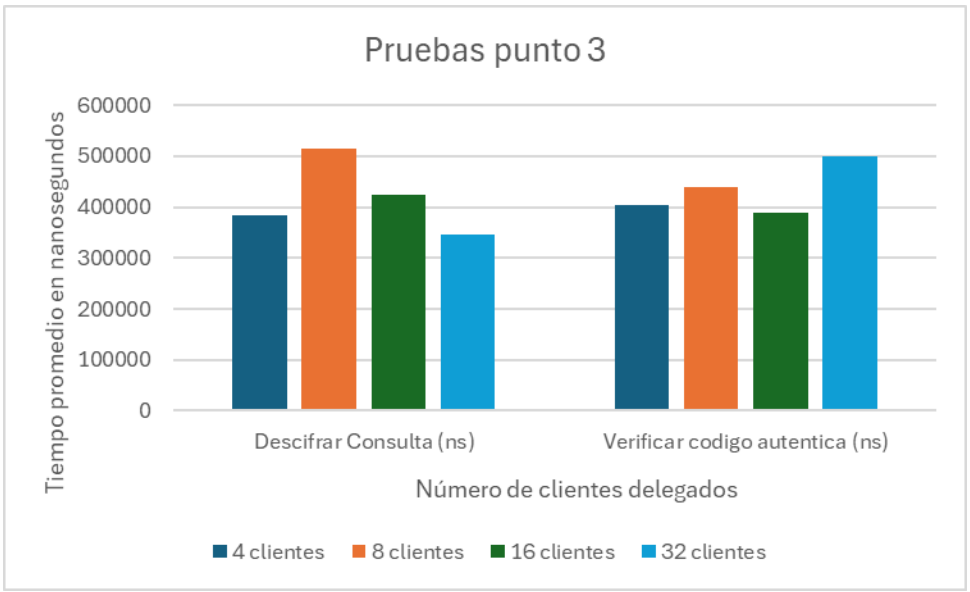
	Generar Firma (ns)		Descifrar Consulta (ns)		Verificar codigo autentica (ns)	
#clientes delegados	Suma total (ns)	promedio (ns)	Suma total (ns)	promedio (ns)	Suma total (ns)	promedio (ns)
4	372141350	93035337,5	1535013,32	383753,33	1618800	404700
8	925729000	115716125	4128000	516000	3509200	438650
16	3722756400	232672275	6769090,72	423068,17	6229485,6	389342,85
32	2602416356	81325511,11	11070221,44	345944,42	15960960	498780

4. Construya una tabla con los datos recopilados (tenga en cuenta que necesitará correr cada escenario en más de una ocasión para validar los resultados) y luego construya una gráfica con los datos de la tabla.

Las siguientes tablas contienen el tiempo promedio en el que se ejecuta la instrucción en los diferentes escenarios.


Pruebas Punto 2
Tiempo promedio en nanosegundos
5000000
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
Verificar Firma (ns)
Calcular Gy (ns)
Cifrar Consulta (ns)
Generar codigo autenticación (ns)
Número Clientes delegados
4 clientes
8 clientes
16 clientes
32 clientes


Pruebas Generar Firmas
Tiempo promedio en nanosegundos
250000000
200000000
150000000
100000000
50000000
0
Generar Firma (ns)
Número clientes delegados
4 clientes
8 clientes
16 clientes
32 clientes


Pruebas punto 3
Tiempo promedio en nanosegundos
600000
500000
400000
300000
200000
100000
0
Descifrar Consulta (ns)
Verificar codigo autentica (ns)
Número de clientes delegados
4 clientes
8 clientes
16 clientes
32 clientes

5. Escriba sus comentarios sobre los resultados.

Como se puede observar en las gráficas a medida que se aumentan el número de clientes delegados, se observa que algunas operaciones, como la verificación de firmas y el cifrado de consultas, escalan linealmente en tiempo con el aumento de la carga, lo que refleja una mayor demanda computacional. Específicamente, la verificación de firmas muestra un incremento considerable en el tiempo promedio por operación, lo que es esperado debido a la complejidad computacional de algoritmos como RSA, especialmente con claves de gran longitud.

Por otro lado, operaciones como el cálculo de Gy (Diffie-Hellman) y la generación de códigos de autenticación (HMAC) varían menos a medida que aumentan los clientes delegados, por lo que es posible que estas operaciones no consuman tantos recursos y se adapten bien a medida que aumenta la carga de clientes. Esto indica que mientras algunas tareas criptográficas pueden manejar eficientemente el aumento en carga, otras pueden requerir optimizaciones adicionales o aumento de recursos para mantener la eficacia bajo mayores demandas.

Esto se evidencia en los resultados la operación de verificación de firma aumenta de un tiempo promedio de 1,039,156.25 nanosegundos con 4 clientes a 1,581,660.23 nanosegundos con 32 clientes, lo que refleja un aumento significativo de 52% en el tiempo de procesamiento por operación a medida que se cuadruplica el número de clientes.

Sin embargo, el cálculo de Gy (Diffie-Hellman) es más estable, empezando con 353,440 nanosegundos para 4 clientes y aumentando modestamente a 379,425 nanosegundos para 32 clientes, un incremento de solo 7.3%. Esto indica una mejor eficiencia y escalabilidad en comparación con la verificación de firma.

6. Identifique la velocidad de su procesador, y estime cuántas consultas puede cifrar su máquina, cuántos códigos de autenticación puede calcular y cuántas verificaciones de firma, por segundo. Escriba todos sus cálculos.

El procesador del computador utilizado es un Core i5 9300H con una velocidad de 2.4 GHz y 8 hilos.

Cifrado de consultas

Tiempo de cifrado de consultas: 4’482.687,36 ns.

#cifrado de consultas/segundo= (velocidad procesador) / (tiempo cifrado * 8 hilos)

#cifrado de consultas/segundo= (2.4*10^9)/(4’482.687,36 *10^-9 * 8 hilos) = 66,95 cifrados /segundo

Cálculo de códigos de autenticación

Tiempo promedio de generar código: 505.847,14 ns.

#códigos de autenticación/segundo= (velocidad procesador) / (tiempo generar código* 8 hilos)

#códigos de autenticación /segundo= (2.4*10^9)/(505.847,14 *10^-9 * 8 hilos) = 593, códigos /segundo

Cálculo de verificaciones de firma

Tiempo promedio de verificar una firma : 1’581.660,23 ns.

#verificaciones de firma/segundo= (velocidad procesador) / (tiempo verificar firma* 8 hilos)

#verificaciones de firma /segundo= (2.4*10^9)/(1’581.660,23 *10^-9 * 8 hilos) = 189,65 firmas verificadas/segundo

Construcción de la solución