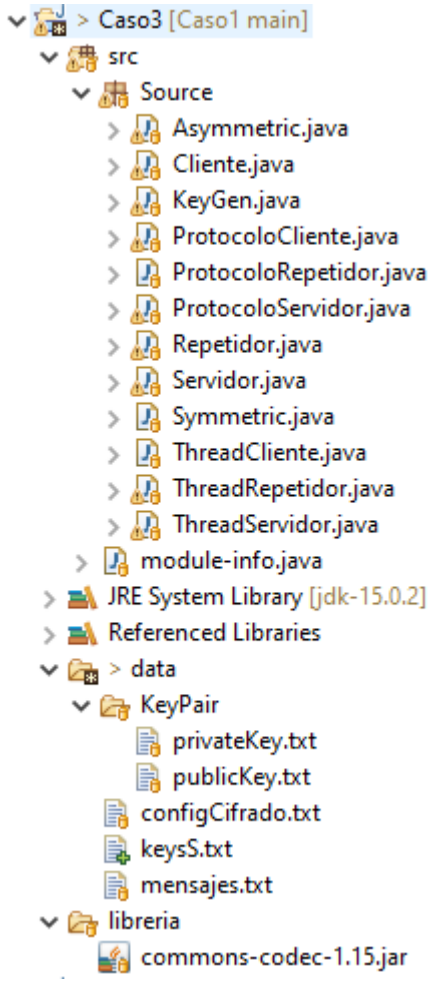
Alejandro Alcaraz 201921767

Juan Sebastián Pinzón 201915773

Caso 3 InfraComp

- **Organización de Archivos:**

En la siguiente captura podemos ver cómo fueron organizados los archivos del proyecto:



En el source se encuentran los tres programas a correr: El servidor principal, el repetidor y la clase principales de cliente. Además, cada uno de estos tiene las clases Thread y Protocolo. Las Thread se encargan de la conexión con los delegados, mientras que las clases protocolo se encargan de la comunicación de los mensajes cifrados. Por otro lado, tenemos la clase generadora de llaves asimétricas KeyGen, la clase de cifrado asimétrico Asymmetric y la clase de cifrado simétrico Symmetric.

En la sección de data tenemos la configuración de cifrado configCifrado, donde se debe elegir el tipo de cifrado entre 0: Simétrico y 1: Asimétrico previo a la ejecución. En esta carpeta también se encuentran las llaves simétricas en keyS, las cuales fueron generadas online, y carpeta KeyPair donde están las llaves asimétricas públicas y privadas.

- **Ejecución**

1. Primero se ejecuta el cliente, establecemos el cifrado con 0 para Simétrico o 1 para Asimétrico.
2. Después, indicamos el número de clientes que van a acceder al sistema.
3. Únicamente cuando se realice lo anterior, se ejecuta el servidor y después el repetidor.
4. En este momento, la consola pedirá que escribas 1 para continuar entonces lo haces.
5. A partir de aquí comienza la ejecución del esquema de comunicación Cliente-Repetidor -Servidor

- **Esquema generación de llaves**

Para las llaves simétricas, simplemente usamos un generador de texto en internet y las almacenamos en un archivo de texto denominado keysS.txt, cuya primera línea es el número de clientes que entrarán al sistema. Si el número de clientes es 16, se tendrán las 16 llaves simétricas entre cliente y repetidor y una llave simétrica más entre repetidor y servidor.

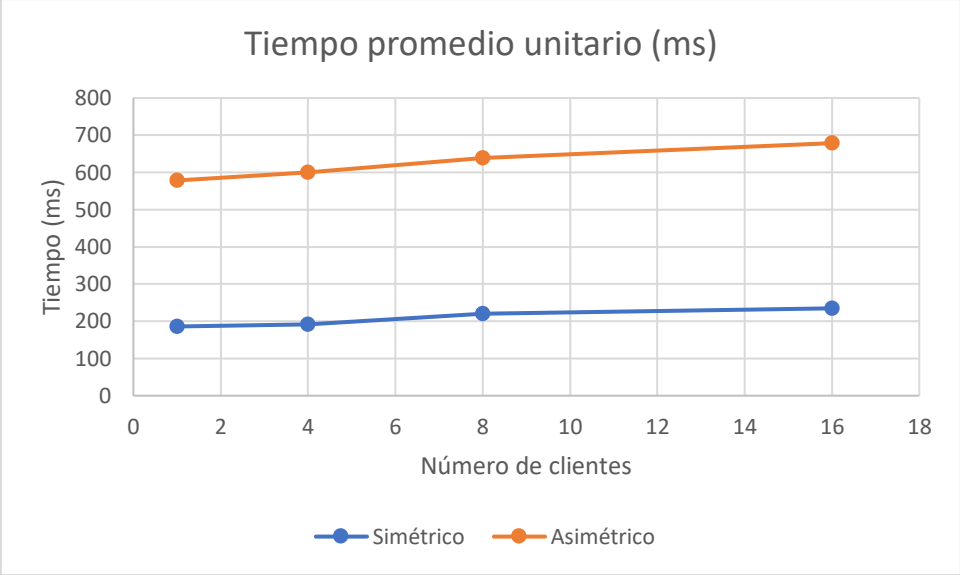
En cuanto a las llaves asimétricas, recurrimos a la creación de una clase denominada KeyGen, donde le pasamos como parámetro el número de clientes y el tamaño de la llave, la cual debe ser 1024. Esta clase genera una llave privada y una pública para cada cliente, además de otros dos pares para el repetidor y el servidor. Estas llaves son guardadas como Strings por filas en dos archivos: privateKey.txt y publicKey.txt, ambos ubicados en data/KeyPair. La primera fila de estos archivos contiene el número de clientes. Las siguientes dos filas son las llaves del servidor y el repetidor, respectivamente. A partir de ahí se encuentran las llaves de los clientes organizadas.

- **Punto 1**

1. tabla de tiempos

	Número Clientes	Simétrico				Asimétrico			
	Tipo	1	4	8	16	1	4	8	16
Intento	1	189	755	1813	3107	599	2373	5026	10659
	2	185	782	1668	3486	580	2415	5417	10974
	3	184	756	1801	4672	556	2416	4891	10944
Promedio		186	764.333333	1760.66667	3755	578.333333	2401.33333	5111.33333	10859
Promedio unitario		186	191.083333	220.083333	234.6875	578.333333	600.333333	638.916667	678.6875

2. A partir del promedio unitario encontrado, se realiza la siguiente gráfica:



3. De esta gráfica podemos concluir que el cifrado simétrico es mucho más rápido que el cifrado asimétrico, lo cual tiene sentido respecto a la complejidad de las operaciones asimétricas respecto a la de las simétricas. Además, podemos concluir que mientras más clientes, el tiempo promedio aumenta, pero por muy poco, es decir, la concurrencia es efectiva.
4. Encontramos el promedio total en simétrico y en asimétrico:

	Número Clientes	Simétrico				Asimétrico			
	Tipo	1	4	8	16	1	4	8	16
Intento	1	189	755	1813	3107	599	2373	5026	10659
	2	185	782	1668	3486	580	2415	5417	10974
	3	184	756	1801	4672	556	2416	4891	10944
Promedio		186	764.333333	1760.66667	3755	578.333333	2401.33333	5111.33333	10859
Promedio unitario		186	191.083333	220.083333	234.6875	578.333333	600.333333	638.916667	678.6875
Promedio por cifrado		207.9635417				624.0677083			

Para nuestro caso, usamos un procesador de 2Ghz, es decir, 2*10^9 ciclos por segundo. En el caso Simétrico, tuvimos un tiempo promedio de 207.96 milisegundos. Entonces, el número de ciclos de procesador que toma recibir una solicitud en el repetidor, descifrarla y volverla a cifrar para enviarla al siguiente destino es:

$$numInstruccionesSimétrico = 2 * 10^9 \text{ciclos por segundo} * 0.207963 \text{ Segundos}$$
$$= 0.4159\ 10^9 \text{ instrucciones}$$

Para el caso de cifrado asimétrico el tiempo promedio por cifrado fue de 624.07 milisegundos. Por lo tanto, el resultado es:

$$numInstruccionesSimétrico = 2 * 10^9 \text{ciclos por segundo} * 0.624068 \text{ Segundos}$$
$$= 1.248\ 10^9 \text{ instrucciones}$$

Referencias

Cryptography and network security, W. Stallings, Ed. Prentice Hall, 2003

Java - Asymmetric Cryptography example - Mkyong.com. (2021). Retrieved 8 November 2021, from https://mkyong.com/java/java-asymmetric-cryptography-example/

Caso de estudio # 3 - Seguridad - Infraestructura Computacional - Universidad de los Andes – 2021

Random string generator - Special. (2021). Retrieved 8 November 2021, from http://www.unit-conversion.info/texttools/random-string-generator/

(2021). Retrieved 8 November 2021, from https://newbedev.com/java-aes-128-ecb-mode-java-code-code-example