Juan David Duarte Yara - 202215070

Julián Santiago Rolón Toloza - 202215839

i). Descripción de la organización de los archivos:

En la carpeta “src” tiene un paquete que se llama “aplicación".

En esta carpeta se encuentran los archivos que se encargan del cliente y el servidor.

Para la ejecución del cliente

Cliente.java: Esta instancia contiene un método main que inicia el proceso de cada cliente, esta clase se puede correr varias veces para simular carios clientes. En esta clase se abre el puerto en el que el servidor se abrió (1234). Luego de esto crea una instancia de la clase “ProtocoloCliente.java”. Esta clase crea los objetos DataOutputStream y DataInutStream para que ProtocoloCliente pueda comunicarse con Protocoloservidor (escribirle al servidor y leer al servidor).

ProtocoloCliente.java: Esta instancia es creada por la clase “Cliente.java” En esta clase se realizan todos los pasos de comunicación entre (cliente - servidor) desde el lado del cliente. La comunicacion en realidad se hace entre (ProtocoloCliente - ProtocoloServidor). La comunicación se da en lo que dice la clase procesar (Que está separada por pasos según el enunciado).

Para la ejecución del servidor:

Servidor.java: Esta instancia contiene un método main que inicia el proceso del servidor. Esta clase debe ser corrida solo una vez. Esta clase abre el puerto del servidor (1234) para que los clientes se conecten por ese socket. Apenas llega un cliente, el servidor crea una instancia de la clase “InstanciaDeRespueta.java” en la que por parámetro se le ingresa el cliente.

InstanciaDeRespueta.java: A cada cliente se le asigna una instancia de respuesta. Esta clase crea los objetos DataOutputStream y DataInutStream para que ProtocoloServidor pueda comunicarse con ProtocoloCliente (Escribir y leer al cliente)

ProtocoloServidor.java: Esta instancia es creada por la clase “InstanciaDeRespuesta.java” En esta clase se realizan todos los pasos de comunicación entre (cliente - servidor) desde el lado del servidor. La comunicacion en realidad se hace entre (ProtocoloCliente - ProtocoloServidor). La comunicación se da en lo que dice la clase procesar (Que está separada por pasos según el enunciado).

Otras clases:

CifradoSimetrico.java: En esta clase están los métodos de cifrar y descifrar con las llaves secretas, con el vector inicial que se generó de manera aleatoria y usando los protocolos que se piden “AES/CBC/ PKCS5Padding”. ProtocoloCliente y ProtocoloServidor llaman a estos metodos.

En la carpeta documentos se encuentra:

- Un documento “numeroPrimo2.txt” donde se encuentra el numero primo P de tamaño 1024 y el numero G.
- Un documento “clientes.txt” donde se encuentran algunos usuarios aceptados. Si se quiere ingresar más usuarios deberían agregarse a este documento.
- El presente documento.

ii) Pasos para correr el servidor y el cliente

1. El primer paso es correr la clase “Servidor.java”. Acá se abre el puerto para que los clientes puedan consultar.
2. Después de que el servidor este corriendo, para iniciar un cliente, se corre la clase “Cliente.java”.
3. El servidor se da cuenta que llega un cliente, crea una InstanciaDeRespuesta y esta instancia crea un ProtocoloServidor. El cliente crea un ProtocoloCliente. Y la comunicación se empieza a dar en las clases de ProtocoloServidor y ProtocoloCliente.
4. Se realiza de manera automática la comunicación (El mensaje reto es un BigInteger generad aleatoriamente) (Como se indica en enunciado) todo el Diffie-Hellman y se crean las llaves secretas compartidas (Para cifrar y para generar el HMAC)
5. Luego en la consola del cliente se pide un login y un password (Estos datos deben coincidir con alguno que este almacenado en el archivo “clientes.txt” la carpeta documentos).
6. Para realizar la consulta del paso 17 se genera un numero aleatorio que es enviado.
7. En cada consola se imprimen los pasos realizados, así se puede registrar los pasos que se ejecutaron.

```
Cliente iniciado ...
Se envia al servidor "Secure init" con mensaje: "RETO"
La firma que paso el servidor esta correcto y los datos son consistentes
Se reciben los numero primos.
Se generaron las llaves secretas para cifrar y para el HMAC

Por favor, ingresa tu login: julian
Por favor, ingresa tu contraseña: 1234
Número para consultar: 3

El HMAC inicial coincide con el HMAC final

La respuesta del servidor es: 2
```

iii) Respuesta a las preguntas:

Pregunta 1:

(i) *En el protocolo descrito el cliente conoce la llave pública del servidor (K_w+). ¿Cuál es el método usado para obtener estas llaves públicas para comunicarse con servidores web?*

RTA: Generalmente, las llaves públicas de los servidores se distribuyen a través de certificados digitales emitidos por una Autoridad de Certificación (CA). Estos certificados son parte del protocolo TLS/SSL y permiten a los navegadores y otros clientes verificar la autenticidad del servidor al establecer una conexión segura.

(ii) *¿Por qué es necesario cifrar G y P con la llave privada?*

RTA: Los números G y P se cifran con la llave privada para generar una firma digital sobre estos datos y a así asegurar autenticación del servidor, ya que solo el servidor poseedor de la llave privada puede firmar el mensaje. El cliente, al conocer la llave pública, puede verificar la firma para asegurarse de que proviene del servidor correcto.

(iii) *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).*

RTA: Si un tercero malicioso quiere realizar un ataque a un estudiante. Supongamos que por algún motivo el tercero consigue la llave privada de la universidad. Como el canal de comunicación del estudiante con la universidad es público, esta persona tercera puede ver todos los mensajes. Pero aun este viendo toda la comunicación no se puede enterar cual está la llave secreta generada por Diffie-Hellman. Por lo que comunicacion que se realice con esta

llave secreta compartida entre estudiante y universidad está a salvo aunque el atacante tenga la llave privada de la universidad.

			Tiempos en ms			
			Descifrar Consulta	Generar firma	Promedio Descifrar Consulta	Promedio Generar Firma
Número de clientes	**4**		4	35	**5,75**	**40,00**
	16		4	50	**1,38**	**8,94**
	32		15	74	**0,88**	**3,69**

	Número de clientes		
Tiempo en ms	4	16	32
	5,75	1,38	0,88

Referencias:

- El Sist Support. *SHA256Encryption - Encrypt data with SHA256 algorithm*. Disponible en: https://support.elsist.biz/es/articoli/sha256encryption-encrypt-data-with-sha256-algorithm/.
- Hackers de la Programación. *Cliente-Servidor con Threads en Java*. Disponible en: https://somoshackersdelaprogramacion.es/cliente-servidor-con-threads-en-java.
- Microfocus. Cargar claves públicas de cliente en el servidor. Disponible en: https://www.microfocus.com/documentation/reflection-desktop/17-1/rdesktop-guide-es/rsitclient_client_upload_keys_pr.htm
- Ssldragon. ¿Qué es el PFS en ciberseguridad? Explicación del secreto perfecto. Disponible en: https://www.ssldragon.com/es/blog/perfect-forward-secrecy/
- Stackscale. Guía para configurar claves SSH en un servidor Linux paso a paso. Disponible en: https://www.stackscale.com/es/blog/configurar-llaves-ssh-servidor-linux/
- Universidad de los Andes. *Taller 05 - Servidores Concurrentes*. Disponible en: https://bloqueneon.uniandes.edu.co/content/enforced/248600-202410_ISIS2203_01/laboratorios/Taller%2005%20-%20Servidores%20concurrentes.pdf?isCourseFile=true&ou=248600.