PROYECTO FINAL DE SEGURIDAD
TRANSFERENCIA CIFRADA DE ARCHIVOS
Presentado por: Marisol Giraldo y Leonardo Zambrano
Presentado al Profesor: Juan Manuel Madrid

ETAPA DE ANALISIS

Este programa permitirá enviar un archivo entre dos computadores, de manera cifrada. Se empleará para el cifrado el algoritmo AES, con clave de 128 bits. La clave por emplear será una clave de sesión, generada utilizando el algoritmo Diffie-Hellman. El programa emisor y el receptor calcularán un hash MD5 del archivo no cifrado, para garantizar la integridad del proceso.

Recursos

- Eclipse IDE for Java
- JRE 1.8

ETAPA DE DISEÑO

A continuación, se muestra el diagrama de clases de la aplicación de transferencia cifrada de archivos.

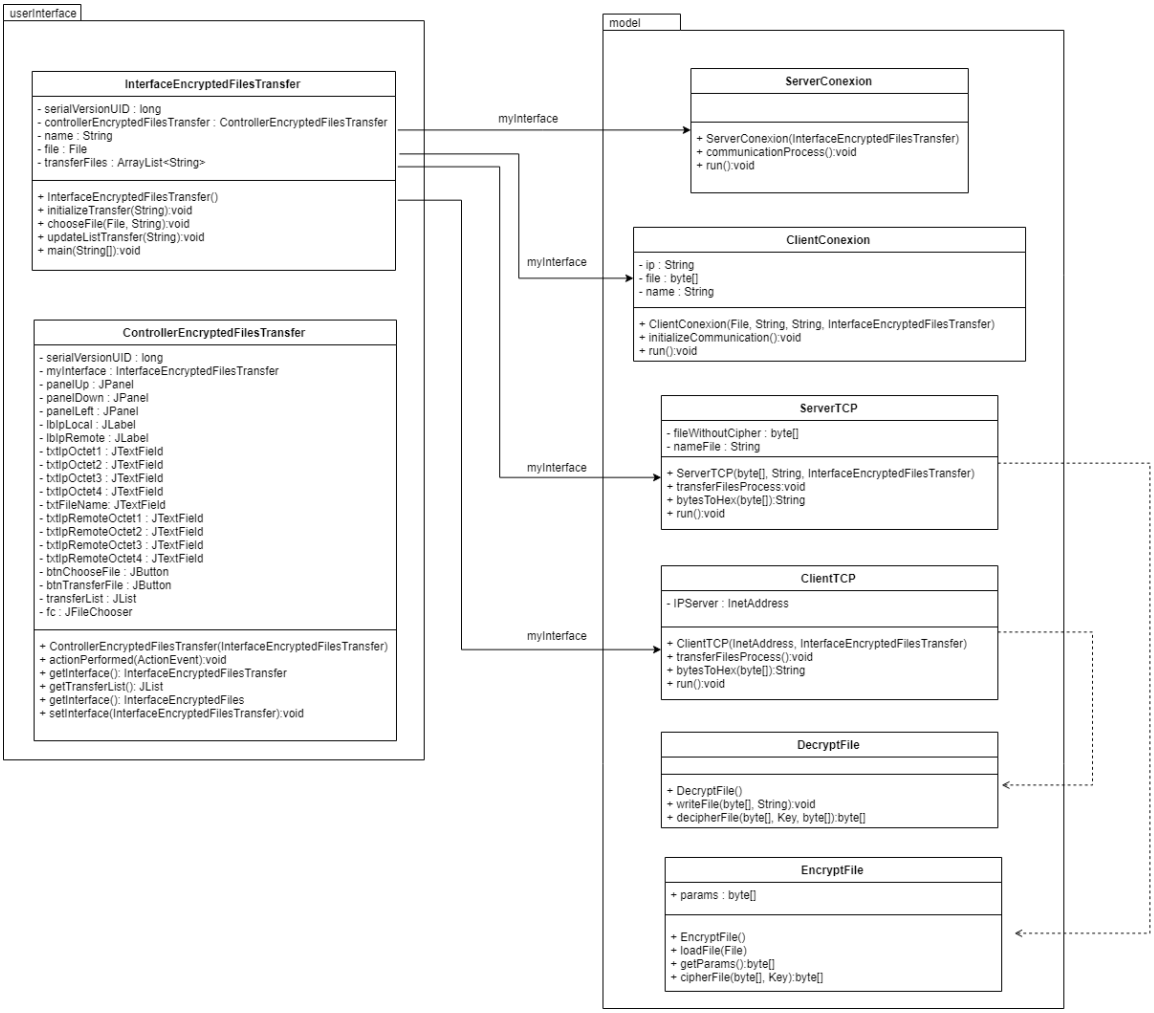

Gráfico 1: Diagrama de Clases

ETAPA DE IMPLEMENTACIÓN

Procedimiento:

1. Se ejecuta la clase principal **InterfaceEncryptedFilesTransfer** que inicializa la interfaz de la aplicación distribuida de la siguiente manera:

- **SECCIÓN INFORMATIVA DE HOST IP LOCAL Y HOST IP REMOTA**

En esta sección el programa cargará la IP del host local donde se esté ejecutando la aplicación y el usuario podrá ingresar la IP del host remoto del equipo a donde realizará la transferencia de archivo.

- **SECCIÓN SELECCIONAR ARCHIVO**

En esta sección el usuario podrá seleccionar el archivo que desea transferir.

- **SECCIÓN TRANFERENCIA DE ARCHIVOS**

En esta sección el usuario podrá enviar el archivo al host remoto y recibirá una confirmación de la lista de archivos transferidos con éxito.

A continuación, se pueden visualizar las secciones anteriormente mencionadas:

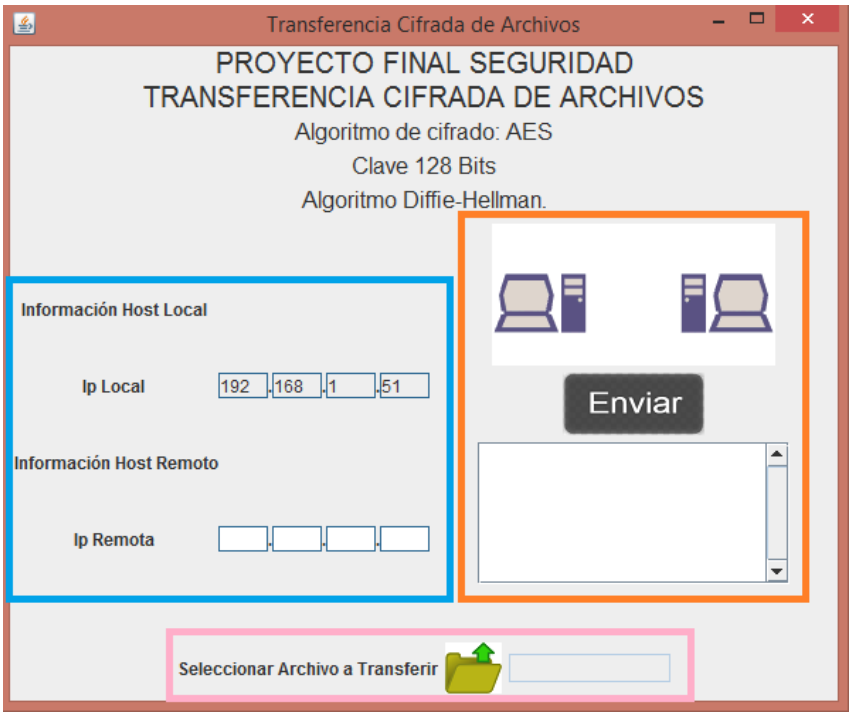

Gráfico 2: Interfaz Transferencia Cifrada de Archivos

2. Se establece la conexión entre el Servidor y el Cliente para realizar la transferencia de archivos.

El paradigma *Cliente/Servidor* se usa para describir un modelo de interacción entre dos procesos, que se ejecutan de forma simultánea, basado en una serie de preguntas y respuestas, que asegura que, si dos aplicaciones intentan comunicarse, una comienza la ejecución y espera indefinidamente que la otra le responda y luego continua con el proceso.

El servidor simplemente espera a la escucha en el *socket* a que un cliente se conecte con una petición. El cliente conoce el *nombre de la máquina* sobre la que está ejecutándose el servidor y el *número de puerto* al que está conectado.

Esta interacción se implementa en las clases **ClientConexion** y **ServerConexion**.

Para la comunicación entre estos se establece una canal público confiable que emplea el protocolo TCP/IP en la clase **ServerTCP** y **ClientTCP**. Una vez se establece la conexión, el servidor y el cliente generan una clave compartida por medio del **Algoritmo Diffie-Hellman.**

3. Se genera una clave compartida entre Servidor y Cliente mediante utilizando el Algoritmo Diffie-Hellman

La criptografía es utilizada para otorgar privacidad, confidencialidad y seguridad. En el envío de información, es importante tener en cuenta el cifrado de mensajes, puesto que evita que terceras personas conozcan el mensaje de tal manera que sea privado entre las fuentes.
El algoritmo AES (Advanced Encryption Standard) es un método criptográfico monoclave, esto quiere decir que se usa la misma clave para cifrar y descifrar, mediante una serie de bucles que se repiten: **10 ciclos para claves de 128 bits**, 12 para 192 y 14 para 256.

En la Clase **ServerTCP**, el algoritmo DH comienza con un número primo grande, **P** y un generador, **G**. Mediante el servicio **SecureRandom** generamos números pseudoaleatorios para el numero **G** de un tamaño de 1024 bits. Una vez generado el numero **G** es enviado al Cliente. En la Clase **ClientTCP** el cliente recibe el numero **G**, y procede mediante el servicio **SecureRandom** a generar el numero **P** de un tamaño de 1024 bits.

Mediante el servicio **KeyParGenerator** se genera pares de claves pública-privada. obteniendo una instancia de esta clase con el método *getInstance("DH")*. Mediante el servicio **KeyAgreement** cliente y servidor coinciden de forma segura en una clave.

Posterior a esta generación, se realiza el intercambio de claves publicas para posteriormente crear la clave privada común.

Mediante el servicio **SecretKeySpec** se genera una clave de 128 bits adecuada para AES, que tenga al menos 16 bytes. Si el array tiene más, el constructor admite dos parámetros para indicar el byte inicial (0) y cuántos bytes se deben coger (16). El último parámetro es el algoritmo para el que queremos la clave "*AES*".

4. Mediante el servicio de Cipher se realiza la encriptación del archivo para que sea transferido de manera segura.

La clase **MessageDigest** maneja el resumen de los mensajes. Esta clase tiene un constructor protegido por lo que se debe acceder a ella a través del método *getInstace(),* donde el parámetro es el algoritmo que queremos resumir. La salida son 128 bits que codifican 16 caracteres para el MD5. Se procede a encriptar del archivo a transferir mediante el método ***cipherFile()*** implementado en la Clase **EncryptFil**e.

Con el servicio **Cipher** adecuado para AES con *Cipher.getInstance("AESECB/PKCS5Padding").* Se inicializa el cifrador para modo encriptar pasando la clave con *init(Cipher.ENCRYPT_MODE, key)* y finalmente podemos encriptar el texto con *doFinal(file)* donde file está en Bytes.

El servidor envía el archivo cifrado junto con el HASH MD5 calculado previamente.

5. Si las claves son correctamente verificadas, mediante el servicio de Cipher se realiza la desencriptación del archivo para que el cliente pueda acceder a él.

El cliente acude a la clase **DecryptFile** para desencriptar el texto, utilizando el método ***decipherFile()*** inicializando el cifrador en modo desencriptar pasándole la clave con *init(Cipher.DECRYPT_MODE, key)* y se desencripta el texto con *aesCipher.doFinal(encriptedFile)* donde encriptedFile está en Bytes.

6. El archivo estará disponible en la carpeta archivos_transferidos

ETAPA DE PRUEBAS

A continuación, se muestra la verificación de que el programa funciona correctamente.

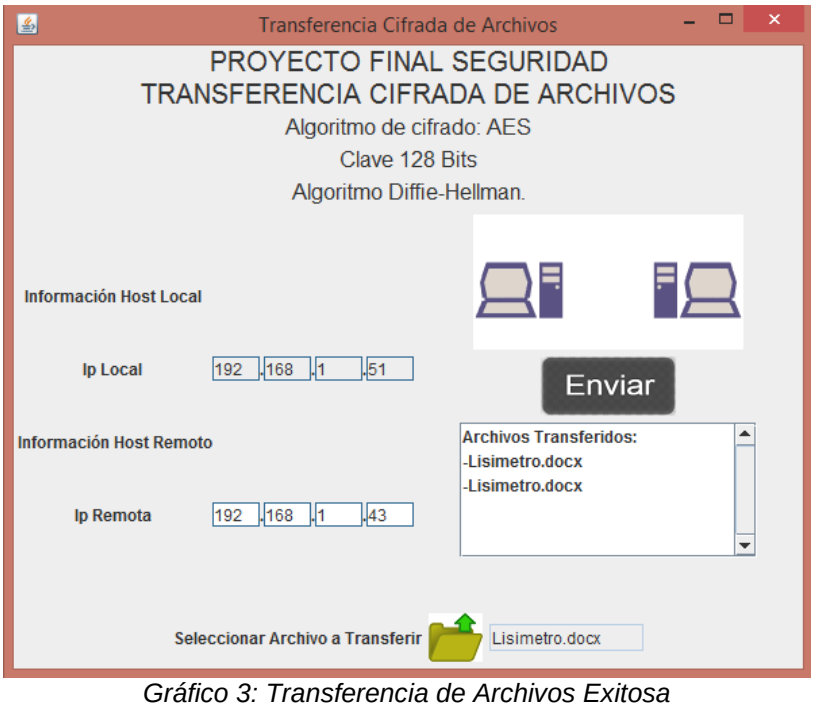

Gráfico 3: Transferencia de Archivos Exitosa

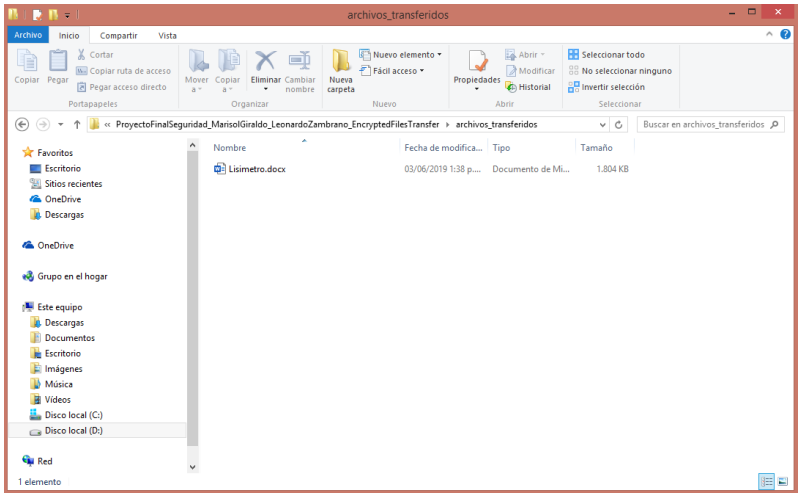

Gráfico 4: Archivo Transferido a la carpeta archivos_transferidos

```
Parametro G Generado y Enviado desde el Servidor  g = 1120301622664185829558771047121737085979119346696527002950173542589
Recibiendo desde el Servidor el Parametro G: 1120301622664185829558771047121737085979119346696527002950173542589601586892
Parametro P Generado y Enviado desde el Cliente  p = 1511242317371116612675664066329400828229989936693221908745348857569941956817
Recibiendo desde el Cliente el Parametro P: 15112423173711166126756640663294008282299899366932219087453488575699419568171
```

Gráfico 5: Verificación de Generación y Envío del Parámetro G y P

```
Recibida Clave Publica del Cliente: SunJCE Diffie-Hellman Public Key:
y:
    0c43c442 12127fc8 c883f63b f00dcfcc 33b151cc 3720f77d ff888f0f ce386da3
    71611fa0 25e935c0 48543d8a 86219153 06962d9d 96c9171a 663e9998 5eea31ba
    4ec8116c 4cf06469 246ec7b8 37058242 f31058dc 6014e830 46557c61 ec95d31b
    24028aea 5a1a158e ad9d037e 7f6b520b 32f5033f 33683cda e624372b ca3b8edb
p:
    9f894776 3a3b8740 0e451605 9c6ad264 2cde0a8a 6af866f1 8f49fb3d a7820d1d
    586de81c 5fb458e5 f42c0368 738291fe 76a869d4 16caceb2 8ace265a 1f924ffc
    0ed5d36d 2d573dcd 07194c77 9f92dffc 23dc2d67 af5b3259 5a4fc7e6 83423672
    c01e8582 4036f0da 2d1928c7 319ab1af f79deb45 2fa6d9c8 5c6509ac e7e97bd9
g:
    d73542bc 466c7eb2 36472521 44f1db25 83497b47 b3d6b309 6f76efd0 2dfd854e
    b196692a aec63a37 3665a18b 7563873d 2f7720a8 2fa70a97 1c7e16a3 da423ff5
    a9578ec4 dc1f89d0 dbf87ba5 f9c31fd9 29c103bc ee365276 115c94fb e744cf77
    c0091bc6 7ce4a7d0 95e7c37e c6ee7f36 261eb164 a6b97990 e90bb77e 86ba3c45
l:
    512
```

Gráfico 6: Clave Publica del Cliente

```
Enviada Clave Publica del Servidor: SunJCE Diffie-Hellman Public Key:
y:
    06f584ab e1026184 10020001 db1d508a d10264ac 64924d86 b7e0b6d4 145748fc
    360c21e6 35954924 b964fb10 c9f619d0 ff8f2d24 d350c2f5 582e9195 719793a7
    b16cc798 56e425e2 c42953e3 017d2b86 b544e8ed 0d116def 37880fd6 a19b9d81
    a3e4114e 57444622 7e8f0ad3 b4386bf1 9aff4b7f 94ea8334 f8c9bb88 baab3bb0
p:
    9f894776 3a3b8740 0e451605 9c6ad264 2cde0a8a 6af866f1 8f49fb3d a7820d1d
    586de81c 5fb458e5 f42c0368 738291fe 76a869d4 16caceb2 8ace265a 1f924ffc
    0ed5d36d 2d573dcd 07194c77 9f92dffc 23dc2d67 af5b3259 5a4fc7e6 83423672
    c01e8582 4036f0da 2d1928c7 319ab1af f79deb45 2fa6d9c8 5c6509ac e7e97bd9
g:
    d73542bc 466c7eb2 36472521 44f1db25 83497b47 b3d6b309 6f76efd0 2dfd854e
    b196692a aec63a37 3665a18b 7563873d 2f7720a8 2fa70a97 1c7e16a3 da423ff5
    a9578ec4 dc1f89d0 dbf87ba5 f9c31fd9 29c103bc ee365276 115c94fb e744cf77
    c0091bc6 7ce4a7d0 95e7c37e c6ee7f36 261eb164 a6b97990 e90bb77e 86ba3c45
l:
    512
```

Gráfico 7: Clave Publica del Servidor

```
Enviando INICIO para Comenzar la Transferencia de Archivo
Inicio de Transmisión al Servidor
From Client: LISTO PARA INICIO
MD5: 67D67179CCDC4EFF5D839E58709B8705
Enviando Información de Control desde el Servidor al Cliente: 0,1846848
Recibiendo Información de Control desde el Servidor: 0,1846848
From Client:OK
Archivo Transferido Correctamente
Transmision: ARCHIVO RECIBIDO
From Client: RECIBIDO
Out To Client: 0,18
Recibi 0,18
Recibi: 67D67179CCDC4EFF5D839E58709B8705
El Cliente recibió la transferencia correctamente
From Client: TRANSFERENCIA CORRECTA
Se Cierra el Socket
Conexion Cerrada
```

Gráfico 8: Confirmación MD5 y Transferencia correcta de archivo

BIBLIOGRAFIA

Encriptación y desencriptación en java
https://ictblog.luisalbertogh.net/?p=274

Uso de las Extensiones Criptográficas de Java
https://www.owasp.org/index.php/Uso_de_las_Extensiones_Criptogr%C3%A1ficas_de_Java

Diffie-Hellman Key Exchange
https://www.example-code.com/java/dh_key_exchange.asp

Encriptación con Java
https://www.scribd.com/document/296731763/Encriptacion-Con-Java