



**UNIVERSIDAD DEL BÍO-BÍO**  
**FACULTAD DE CIENCIAS EMPRESARIALES**  
**ESCUELA INGENIERÍA CIVIL INFORMÁTICA**

# INFORME I

Alumno: Esteban Sepúlveda Carrasco

Asignatura: Seguridad Informática

Profesor: Claudio Muñoz Sepúlveda

Fecha: Martes 24 de noviembre de 2020

# Índice

<b>Introducción.....</b>	<b>3</b>
<b>Desarrollo de la solución.....</b>	<b>4</b>
<b>Parte 1 .....</b>	<b>4</b>
<b>Parte 2 .....</b>	<b>6</b>
<b>Evidencias del trabajo.....</b>	<b>9</b>
<b>Parte 1: Evidencia compilación de la encriptación del mensaje plano.....</b>	<b>9</b>
<b>Parte 2: Evidencia compilación de la desencriptación de mensaje plano. ....</b>	<b>10</b>
<b>Conclusión.....</b>	<b>11</b>

## **Introducción**

Diariamente la tecnología avanza en gran paso, es por esto es importante la seguridad informática ya que se debe combatir con estas nuevas tecnologías que buscan vulnerar la seguridad de programas, mensaje, claves, etc. Es por lo que los ingenieros en informática necesitan aprender y estar actualizando constantemente sus conocimientos para enfrentar de mejor manera este tipo de situaciones.

En el presente informe se expondrán los resultados de utilizar la tecnología simétrica mediante la encriptación de un mensaje, el cual debo enviar a un compañero y él me debe responder con el mensaje descryptado. Se desarrollo de esta tecnología en el lenguaje de programación JAVA.

A continuación, se empezará con una pequeña descripción de lo que se utilizó para lograr la realización de la tarea, luego expondrán las partes de desarrollo de la solución la cual consta de 2 partes encriptar y descryptar un mensaje. Luego se presentarán pantallazos de las evidencias de compilación del programa y también correos enviados y recibidos por parte de un compañero.

## Desarrollo de la solución

En esta sección se explicará el desarrollo de la solución de las dos partes llevado a cabo para desarrollar la tarea, la implementación se desarrolló en el lenguaje de programación Java. Para el desarrollo de la solución se requirió de:

- Llaves privadas y públicas creadas en KeyTool
- Archivo con mensaje encriptado enviado por el compañero
- Certificado de compañero que nos enviara mensaje encriptado
- Librerías de JAVA, como por ejemplo Java.io.File, Java.io.FileOutputStream y Java.io.FileInputStream para el manejo de los archivos, Javax.crypto para el encriptado y desencriptado, entre otras utilizadas.

### Parte 1

- **Inicialización de archivos necesarios para la ejecución.**

En esta parte se inicializan o cargan los archivos necesarios para el desarrollo de la solución, en este caso encriptar. **archivoEncriptar** representa al archivo que tiene el texto plano que se desea encriptar, **archivoEncriptado** representa al archivo que contendrá la información encriptada, **keyStoreFile** representa el archivo.jks el cual contiene mi clave privada, **certificadoCompañero** representa el archivo.cer el cual contiene la llave pública de mi compañero, **contraseña** representa la contraseña que me permite acceder al almacén de llaves.

```
String contraseña = "esteban1998SC";
try {
    //Aquí se inicializan los archivos con sus rutas necesarios para el funcionamiento de este programa
    File archivoEncriptar = new File( pathname: "C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\texto plano.txt");
    File archivoEncriptado = new File( pathname: "C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\texto encriptado.txt");
    File keyStoreFile = new File( pathname: "C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\leisckeystore.jks");
    FileInputStream certificadoCompañero = new FileInputStream( name: "C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\leisckeystore.jks");
}
```

- **Generar llave de sesión AES**

Se genera la llave de sesión AES para la encriptación del mensaje.

```
// Generar llave de sesión AES
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
Key keyAes = keyGenerator.generateKey();
SecretKeySpec secretKeySpec = new SecretKeySpec(keyAes.getEncoded(), "AES");
```

- **Generar vector de inicialización para encriptación CBC**

Se genera el vector de inicialización de manera aleatoria haciendo de esta manera que sea más seguro ya que utiliza números criptográficos seguros, este vector será utilizado para la encriptación de la información mediante AES/CBC.

```
//Generar vector de inicialización para encriptación CBC (cipher-block chaining)
byte[] IV = new byte[16];
SecureRandom random = new SecureRandom();
random.nextBytes(IV);
IvParameterSpec ivParameterSpec = new IvParameterSpec(IV);
Cipher cipherAes = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

➤ **Encriptar mensaje plano con llave de sesión**

Se encripta mensaje plano obtenido a partir del archivo **archivoEncriptar**.

```
cipherAes.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivParameterSpec);  
byte[] mensajeEncriptado = cipherAes.doFinal(textoPlano);
```

➤ **Calcular hash con mensaje plano**

```
//Calcular hash SHA-1 del mensaje plano  
MessageDigest hash = MessageDigest.getInstance("SHA-1");  
hash.update(textoPlano);
```

➤ **Obtener llave publica de compañero**

Se obtiene llave publica de compañero, la luego encriptar la llave de sesión para que luego se le permita descryptar mi mensaje.

```
//Obtener Certificado de mi compañero y la clave publica  
CertificateFactory certificateFactory = CertificateFactory.getInstance("X509");  
Certificate certificate = certificateFactory.generateCertificate(certificadoCompañero);  
PublicKey publicKey = certificate.getPublicKey();  
  
//Encriptar llave de sesión con llave pública de mi compañero  
Cipher cipherPublicRsa = Cipher.getInstance("RSA");  
cipherPublicRsa.init(Cipher.ENCRYPT_MODE, publicKey);  
byte[] llaveSesionCompañeroEncriptado = cipherPublicRsa.doFinal(keyAes.getEncoded());
```

➤ **Obtener mi llave privada y firmar hash**

Obtengo llave privada de mi almacén de llaves, para luego firmar el hash del mensaje plano con mi llave privada.

```
//Obtener la llave privada de mi almacén de llaves  
PrivateKey privateKey = (PrivateKey) myKeyStore.getKey(alias: "rsaPrivateKey", contrasena.toCharArray());  
  
// Inicializar Objeto Cipher RSA  
Cipher PrivateRSA = Cipher.getInstance("RSA");  
PrivateRSA.init(Cipher.ENCRYPT_MODE, privateKey);  
  
//Firmar digitalmente hash del mensaje plano con mi llave privada  
byte[] hashFirmado = PrivateRSA.doFinal(hash.digest());
```

➤ **Escribir en un archivo el IV, la llave de sesión encriptada, el hash firmado y el mensaje encriptado**

Antes de finalizar el programa se escribe en el archivo que contiene la información encriptada el vector de inicialización, llave de sesión AES encriptada, hash firmado y por ultimo el mensaje encriptado, todo esto se realiza con la clase `FileOutputStream` la cual nos permite escribir en el archivo.

```
FileOutputStream escribirEncriptado = new FileOutputStream(archivoEncriptado);

System.out.println("Escribiendo mensaje encriptado...");
//Escribir vector de inicializacion
escribirEncriptado.write(IV);
//Escribir llave de sesion AES encriptada al archivo.
escribirEncriptado.write(llaveSesionCompañeroEncriptado);
// Escribir hash firmado
escribirEncriptado.write(hashFirmado);
// Escribir texto plano encriptado en el archivo.
escribirEncriptado.write(mensajeEncriptado);
```

## Parte 2

➤ **Inicialización de archivos necesarios para la ejecución.**

En esta parte se inicializan o cargan los archivos necesarios para el desarrollo de la solución, en este caso desencriptar. **archivoEncriptado** representa el archivo enviado por mi compañero el cual contiene un mensaje encriptado, **archivoDesencriptado** representa al archivo que contendrá el mensaje desencriptado por mi compañero, **keyStoreFile** representa el archivo.jks el cual contiene mi clave privada, **certificadoCompañero** representa el archivo.cer el cual contiene la llave publica de mi compañero, **contraseña** representa la contraseña que me permite acceder al almacén de llaves.

```
String contraseña = "esteban1998SC";

try {
    File archivoEncriptado = new File("C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\mensaje_Encriptado");
    File archivodesencriptado = new File("C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\textodecencriptado");
    FileInputStream certificadoCompañero = new FileInputStream("C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\certificadoCompañero.cer");
    File keyStoreFile = new File("C:\\Users\\tabi_\\Desktop\\trabajos seguridad\\informe\\eiskystore.jks");
```

### ➤ Obtener información del archivo encriptado

Para obtener la información del archivo se utilizó el método `available()` de la clase `FileInputStream` el cual me permitió tener un manejo de los bytes, lo que me permitió extraer el vector de inicialización, la key, el hash y el texto plano. Este método me permitió saber que cantidad de bytes me quedaban por leer.

```
byte[] IV = new byte[16];
byte[] llaveEncriptada = new byte[256];
byte[] hashEncriptado = new byte[256];
byte[] textoEncriptado = new byte[(int) (archivoEncriptado.length() - 528)];

while (archivoleer.available() > 0) {
    if (archivoleer.available() > (archivoEncriptado.length() - 16)) {
        //IV
        archivoleer.read(IV);
    } else if (archivoleer.available() > (archivoEncriptado.length() - 272)) {
        //La key
        archivoleer.read(llaveEncriptada);
    } else if (archivoleer.available() > (archivoEncriptado.length() - 528)) {
        //hash
        archivoleer.read(hashEncriptado);
    } else {
        //texto plano
        archivoleer.read(textoEncriptado);
    }
}
```

### ➤ Desencriptar llave de sesión con su llave privada

Lo primero se recupera mi llave privada desde mi almacen de llaves y luego se procede a desencriptar la llave de sesión con mi clave privada.

```
KeyStore myKeyStore = KeyStore.getInstance("JKS");
FileInputStream inStream = new FileInputStream(keyStoreFile);
myKeyStore.load(inStream, contrasena.toCharArray());

//Obtiene mi clave privada de mi almacen de llaves
PrivateKey privatekey = (PrivateKey) myKeyStore.getKey(alias: "rsaesickey", contrasena.toCharArray());

//Genera objeto cipher para desencriptar en RSA
Cipher cipherPrivateRSA = Cipher.getInstance("RSA");
cipherPrivateRSA.init(Cipher.DECRYPT_MODE, privatekey);

//Se desencripta de RSA a AES
byte[] decryptKey = cipherPrivateRSA.doFinal(llaveEncriptada);
```

➤ **Desencriptar archivo con la llave de sesión**

Se inicializan los objetos necesarios para la implementación del desencriptado, el código que realiza la desencriptación queda dentro de un try-catch ya que se según el enunciado de la tarea, si la desencriptación del archivo no se puede realizar, el programa debe enviar un mensaje por pantalla indicando el error.

```
//objeto cipher RSA
Cipher cipherPublicRsa = Cipher.getInstance("RSA");
cipherPublicRsa.init(Cipher.DECRYPT_MODE, publicKey);

SecretKeySpec secretKeySpec = new SecretKeySpec(llaveDesencriptada, "AES");
IvParameterSpec ivParameterSpec = new IvParameterSpec(IV);

Cipher cipherAes = Cipher.getInstance("AES/CBC/PKCS5Padding");
cipherAes.init(Cipher.DECRYPT_MODE, secretKeySpec, ivParameterSpec);

//Inicializo byte que contendra mensaje encriptado
byte[] mensajeDesencriptado = new byte[0];

try{
    //Desencripta el texto
    mensajeDesencriptado = cipherAes.doFinal(textoEncriptado);
}catch (Exception e){
    System.out.println("Ocurrio un problema");
    System.out.println("El problema es el siguiente: "+e);
}
```

➤ **Desencriptar hash**

```
byte[] hashDesencriptado = cipherPublicRsa.doFinal(hashEncriptado);
```

➤ **Calcula hash desencriptado**

```
MessageDigest hash = MessageDigest.getInstance("SHA-1");
hash.update(mensajeDesencriptado);
```

➤ **Valido firmas**

Comparo el hash extraído del archivo que contiene el mensaje de mi compañero, con el hash del mensaje, si estos son iguales entonces la firma digital es válida y el mensaje es auténtico por lo tanto muestro por pantalla y escribo en el archivo. Si son diferentes significa que el mensaje no es auténtico, por lo tanto, pudo haber sido manipulado.

```
if (Arrays.equals(hashDesencriptado,hash.digest())){
    System.out.println("Abriendo archivo a escribir: " + archivodesencriptado);
    FileOutputStream escribirDesencriptado = new FileOutputStream(archivodesencriptado);
    escribirDesencriptado.write(mensajeDesencriptado);
    System.out.println(new String(mensajeDesencriptado));
}else{
    System.out.println("El mensaje ha sido manipulado");
}
```



## Evidencias del trabajo

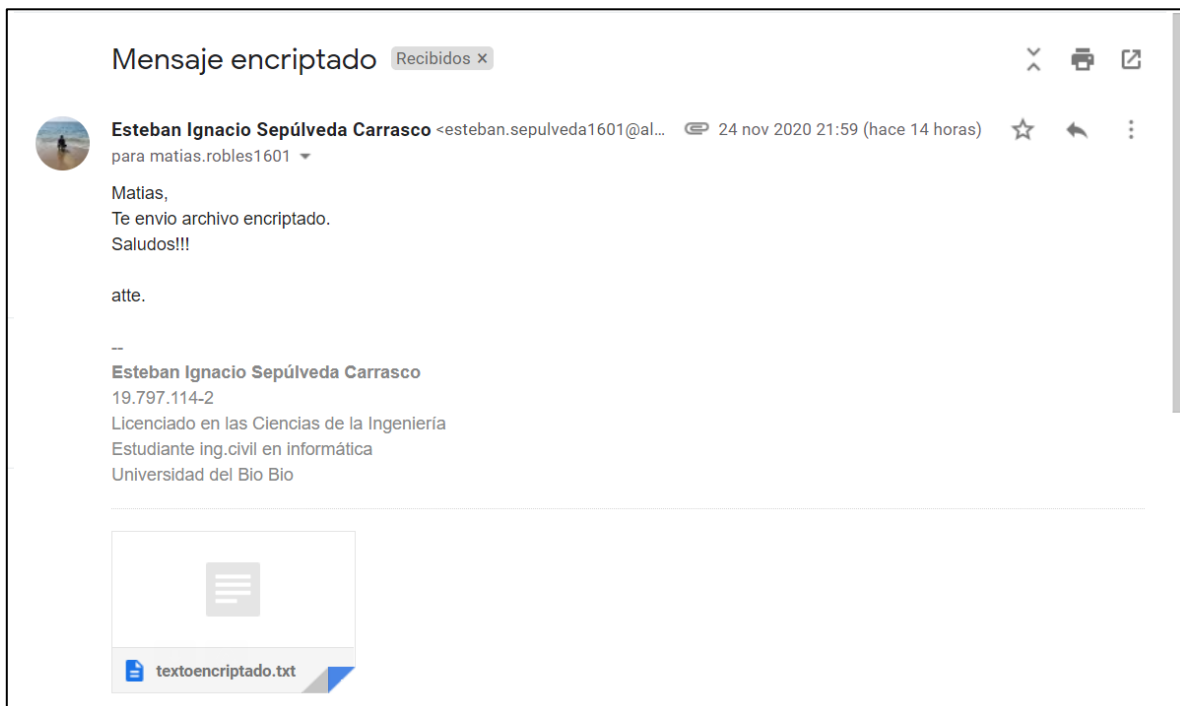
En esta sección se presentarán las evidencias de compilación del código y el envío de correo a compañero. En esta oportunidad mi compañero de trabajo es Matías Robles Díaz.

### Parte 1: Evidencia compilación de la encriptación del mensaje plano.

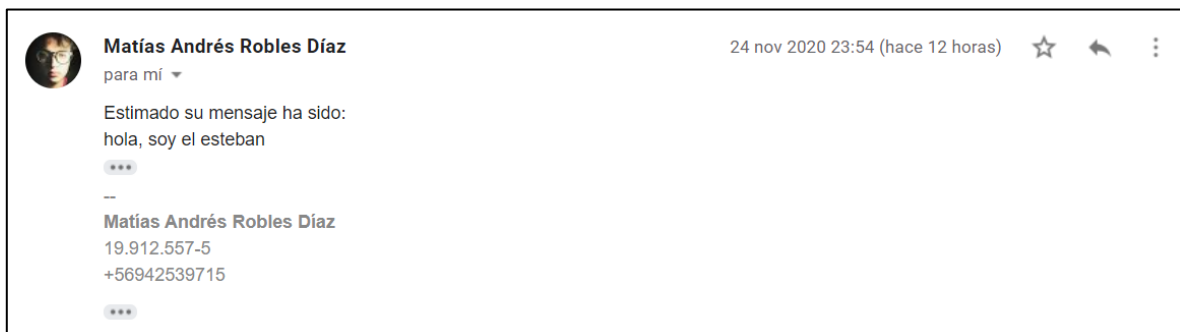
```
Leyendo archivo: C:\Users\tabi\Desktop\trabajos seguridad\informe\textoplano.txt
Obteniendo mensaje desde el archivo...
Mensaje encontrado: hola, soy el esteban
Abriendo archivo a escribir: C:\Users\tabi\Desktop\trabajos seguridad\informe\textoencryptado.txt
Escribiendo mensaje encriptado...
Finalizando...

Process finished with exit code 0
```

### Evidencia correo enviado a compañero



### Respuesta de correo por parte del compañero




## Parte 2: Evidencia compilación de la descriptación de mensaje plano.

```
Abriendo Archivo a leer: C:\Users\tabi\Desktop\trabajos seguridad\informe\mensaje_Encryptado.txt
Extrallendo datos
Abriendo archivo a escribir: C:\Users\tabi\Desktop\trabajos seguridad\informe\textodecencryptado.txt
Hola, este es un mensaje para encriptar

Process finished with exit code 0
```

## Evidencia mensaje recibido por compañero



**Matías Andrés Robles Díaz** <matias.robles1601@alumnos.ubiobio.cl> mar, 24 nov 21:56 (hace 15 horas) para mí ▾

🌐 inglés ▾ > español ▾ Traducir mensaje


Desactivar para: inglés x

Hola!,  
Te envío el archivo encriptado ;).  
Saludos!

--  
**Matías Andrés Robles Díaz**  
19.912.557-5  
+56942539715  
Estudiante ing.civil en informática  
Universidad del Bio Bio

  
 mensaje\_Encryptad...

## Evidencia respuesta que envié a mi compañero

**Esteban Ignacio Sepúlveda Carrasco** <esteban.sepulveda1601@alumn... 24 nov 2020 22:11 (hace 14 horas) para Matías ▾

Matias, su mensaje a sido desencryptado conteniendo el siguiente mensaje: " Hola, este es un mensaje para encriptar "

atte.  
...

--  
**Esteban Ignacio Sepúlveda Carrasco**  
19.797.114-2  
Licenciado en las Ciencias de la Ingeniería  
...

## **Conclusión**

Actualmente la criptografía es una herramienta muy importante puesto que ofrece confianza en la seguridad de la información, se encarga de resguardar los datos de las personas las cuales no está dirigida la información, así es como solo a la gente a la cual va dirigida la información pueda tener acceso. La criptografía ofrece claves las cuales son capaces de encriptar o desencriptar los datos usando diferentes métodos.

En este trabajo se utilizó el método de criptografía simétrica, este trabajo me ayudo a comprender el funcionamiento de este método, se comprendió de mejor manera como funcionaba las claves publicas y privadas, la firma digital, el vector de inicialización, distintos procesos de encriptación RSA, AES Y SHA-1.