INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Cryptography

"Digital signature"

Abstact

The Data Encryption Standard is a symmetric-key algorithm for the encryption of electronic data. Although insecure, it was highly influential in the advancement of modern cryptography.

By:

Meza Martínez Luis Daniel

Professor:
MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

June 2018

Index

Contenido

Introduction: 1

Literature review: 2

Software (libraries, packages, tools): 7

Procedure: 7

Results 9

Discussion: 12

Conclusions: 12

References: 12

Code 13

Introduction:

To perform this practice, we need to take some concepts such as encryption by DES / AES, HASH and in this case know how the RSA works, we need to know how to make, in this case I made this practice using JAVA.

DES Data Encryption Standar

In 1973, the National Bureau of Standards (NBS), later to become the National Institute of Standards and Technology (NIST), issued a public request seeking a cryptographic algorithm to become a national standard. IBM submitted an algorithm called LUCIFER in 1974. The NBS forwarded it to the National Security Agency, which reviewed it and, after some modifications, returned a version that was essentially the Data Encryption Standard (DES) algorithm. In 1975, NBS released DES, as well as a free license for its use, and in 1977 NBS made it the official data encryption standard.

DES has been used extensively in electronic commerce, for example in the banking industry. If two banks want to exchange data, they first use a public key method such as RSA to transmit a key for DES, then they use DES for transmitting the data. It has the advantage of being very fast and reasonably secure.

The DES algorithm is rather unwieldy to use for examples, so in the present section we present an algorithm that has many of the same features but is much smaller. Like DES, the present algorithm is a block cipher. Since the blocks are encrypted separately, we assume throughout the present discussion at the full message consists of only one block.

Advanced Encryption Standar

In 1997, the National Institute of Standards and Technology put out a call for candidates to replace DES. Among the requirements were that the new algorithm should allow key sizes of 128, 192, and 256 bits, it should operate on blocks of 128 input bits, and it should work on a variety of different hardware, for example, 8-bit processors that could be used in smart cards and the 32-bit architecture commonly used in personal computers. Speed and cryptographic strength were also important considerations. In 1998, the cryptographic community was asked to comment on 15 candidate algorithms. Five finalists were chosen: MARS (from IBM), RC6 (from RSA Laboratories), Rijndael (from Joan Daemen and Vincent Rijmen), Serpent (from Ross Anderson, Eli Biham, and Lars K nudsen), and Twofish (from Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson). Eventually, Rijndael was chosen as the Advanced Encryption Standard. The other four algorithms are also very strong, and it is likely that they will used in many future cryptosystems.

Hash functions and data integrity

A cryptographic hash function can provide assurance of data integrity. A hash function is used to construct a short "fingerprint" of some data; if the data is altered, then the fingerprint will no longer be valid. Even if the data is stored in an insecure place, its integrity can be checked from time to time by recomputing the fingerprint and verifying that the fingerprint has not changed.

Let h be a hash function and let x be some data. As an illustrative example, x could be a binary string of arbitrary length. The corresponding fingerprint is defined to be y = h(x). thins fingerprint is often referred to as a message digest. A message digest would typically be a short binary string.

We suppose that y is stored in a secure place, but x is not. If x is changed to x', say, then we hope that the "old" message digest, y is not also a message digest for x'. If this is indeed the case, then the fact that x has been altered can be detected simply by computing the message digest.

Transactions between users over the Internet require protocols to provide secrecy and authentication of both the sender's identity and the content of the message. We review the elements of digital signatures and message authentication in this chapter, and how cryptographic transformations can provide both secrecy and authentication.

Digital signatures

For years, people have been using various types of signatures to associate their identities to documents. In the Middle Ages, a nobleman sealed a document with a wax imprint of his insignia. The assumption was that the noble was the only person able to reproduce the insignia. In modern transactions, credit card slips are signed. The salesperson is supposed to verify the signature by comparing with the signature on the card. With the development of electronic commerce and electronic documents, these methods no longer suffice.

Literature review:

The federal government originally developed DES encryption over 35 years ago to provide cryptographic security for all government communications. The idea was to ensure government systems all used the same, secure standard to facilitate interconnectivity.

To show that the DES was inadequate and should not be used in important systems anymore, a series of challenges were sponsored to see how long it would take to decrypt a message. Two organizations played key roles in breaking DES: distributed.net and the Electronic Frontier Foundation (EFF).

Des description

DES is the archetypal block cipher—an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

The key is nominally stored or transmitted as 8 bytes, each with odd parity. According to ANSI X3.92-1981 (Now, known as ANSI INCITS 92-1981), section 3.5:

One bit in each 8-bit byte of the KEY may be utilized for error detection in key generation, distribution, and storage. Bits 8, 16,..., 64 are for use in ensuring that each byte is of odd parity.

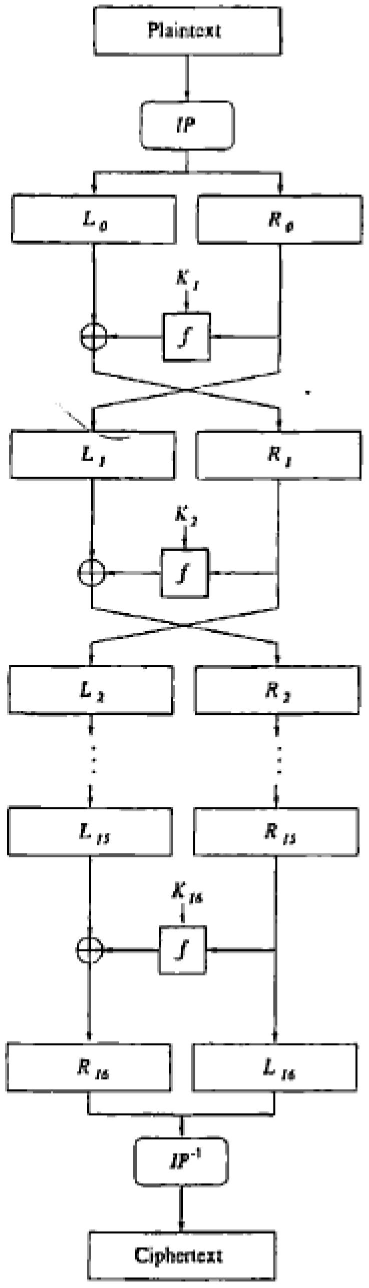


Figure 4.4: The DES Algorithm.

Like other block ciphers, DES by itself is not a secure means of encryption, but must instead be used in a mode of operation. FIPS-81 specifies several modes for use with DES. Further comments on the usage of DES are contained in FIPS-74.

Decryption uses the same structure as encryption, but with the keys used in reverse order. (This has the advantage that the same hardware or software can be used in both directions).

AES

The Advanced Encryption Standard (AES), also known by its original name Rijndael,is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

AES is based on a design principle known as a substitution–permutation network, a combination of both substitution and permutation, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4 × 4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a particular finite field.

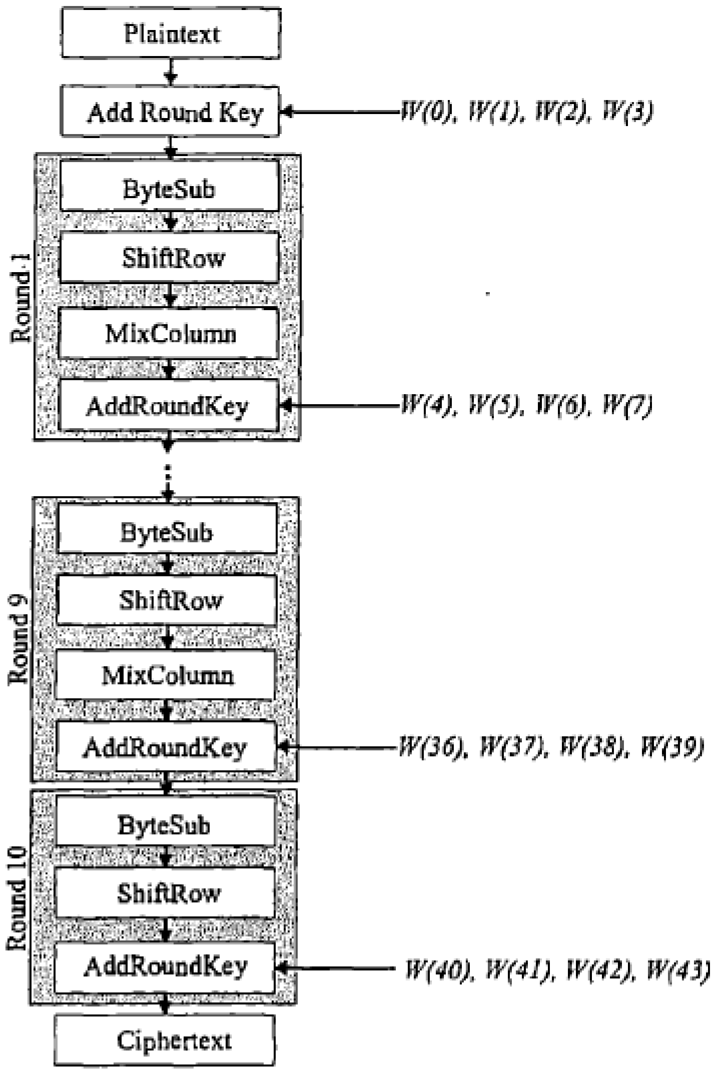


Figure 5.1: The AES-Rijndael Algorithm

Rijndael is designed for use with keys of lengths 128, 192, and 256 bits. For simplicity, we'll restrict to 128 bits. First, we give a brief outline of the algorithm, then describe the various components in more detail. The algorithm consists of 10 rounds (when the key has 192 bits, 12 rounds are used, and when the key has 256 bits, 14 rounds are used). Each round has a round key, derived from the original key. There is also a 0th round key, which is the original key. A round starts with an input of 128 bits and produces an o u tput of 128 bits.

Hash

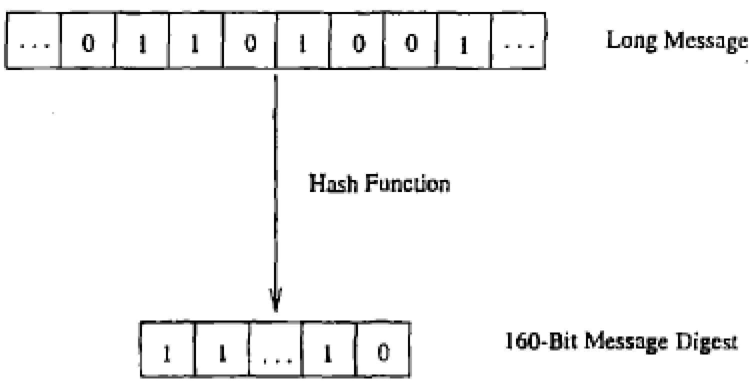


Figure 8.1: A Hash Function.

A basic component of many cryptographic algorithms is what is known as a hash function. When a hash function satisfies certain non-invertibility properties, it can be used to make many algorithms more efficient. In the following, we discuss the basic properties of hash functions and attacks on them. We also briefly discuss the random oracle model, which is a method of analyzing the security of algorithms that use hash functions. Later, in Chapter 9, hash functions will be used in digital signature algorithms. They also play a role in security protocols in Chapter 10, and in several other situations. A cryptographic hash Function h takes as input a message of arbitrary length and produces as output a message digest of fixed length.

Note that since the set of possible messages is much larger than the set of possible message digests, there should always be many examples of messages 7711 and m2 with h(m \) = h{m2). The requirement (3) says that it should be hard to find examples. In particular, if Bob produces a message m and its hash h(m), Alice wants to be reasonably certain that Bob does not know another message m 1 with h{m') = h(m), even if both m and m! arc allowed to be random strings of symbols. In practice, it is sometimes sufficient to weaken (3) to require H to be weakly collision-free. This means that given x, it is computationally infeasible to find x 1 r 1 with H{x') = H (x). This property is also called second preimage resistance.

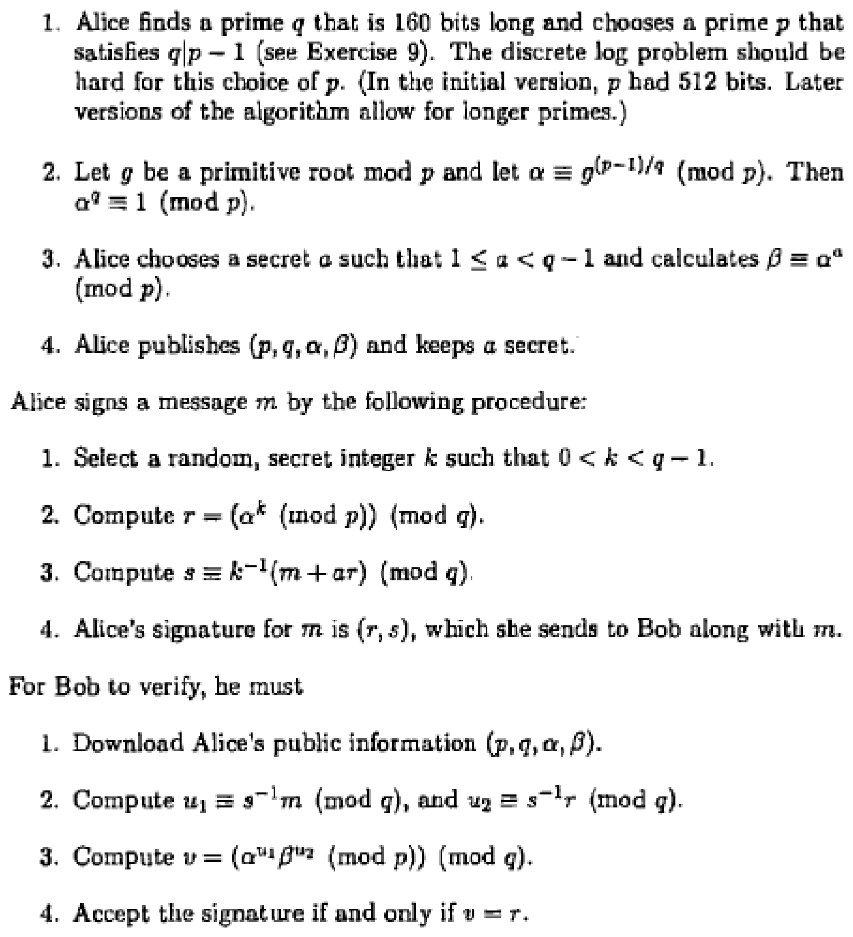
Hash functions may also be employed as a check on data integrity. The question of data integrity comes up in basically two scenarios. The first is when the data (encrypted or not) are being transmitted to another person and a noisy communication channel introduces errors to the data. The second occurs when an observer rearranges the transmission in some manner before it gets to the receiver. Either way, the data have become corrupted. For example, suppose Alice sends Bob long messages about financial transactions with Eve and encrypts them in blocks. Perhaps Eve deduces that the tenth block of each message lists the amount of money that is to be deposited to Eve’s account. She could easily substitute the tenth block from one message into another and increase the deposit. In another situation, Alice might send Bob a message consisting of several blocks of data, but one of the blocks is lost during transmission. Bob might not ever realize that the block is missing.

Digital signature.

For years, people have been using various types of signatures to associate their identities to documents. In the Middle Ages, a nobleman sealed a document with a wax imprint of his insignia. The assumption was that the noble was the only person able to reproduce the insignia. In modern transactions, credit card slips are signed. The salesperson is supposed to verify the signature by comparing with the signature on the card. With the development of electronic commerce and electronic documents, these methods no longer suffice.

For example, suppose you want to sign an electronic document. Why can't you simply digitize your signature and append it to the document? Anyone who has access to it can simply remove the signature and add it to something else, for example, a check for a large amount of money. With classical signatures, this would require cutting the signature off the document, or photocopying it, and posting it on the check. This would rarely pass for an acceptable signature. However, such an electronic forgery is quite easy and cannot be distinguished from the original.

The National Institute of Standards and Technology proposed the Digital Signature Algorithm (DSA) in 1991 and adopted it as a standard in 1994. Just like the ElGamal method, DSA is a digital signature scheme with appendix. Also, like other schemes, it is usually a message digest that is signed. In this case, the hash function produces a 160-bit output. We will assume in the following that our data message m has already been hashed. Therefore, we are trying to sign a 160-bit message. The generation of keys for DSA proceeds as follows. First, there is an initialization phase:



1. Alice finds a prime q that is 160 bits long and chooses a prime p that satisfies $q|p-1$ (see Exercise 9). The discrete log problem should be hard for this choice of p. (In the initial version, p had 512 bits. Later versions of the algorithm allow for longer primes.)
2. Let g be a primitive root mod p and let $\alpha \equiv g^{(p-1)/q} \pmod{p}$. Then $\alpha^q \equiv 1 \pmod{p}$.
3. Alice chooses a secret a such that $1 \le a < q-1$ and calculates $\beta \equiv \alpha^a \pmod{p}$.
4. Alice publishes (p, q, α, β) and keeps a secret.

Alice signs a message m by the following procedure:

1. Select a random, secret integer k such that $0 < k < q-1$.
2. Compute $r = (\alpha^k \pmod{p}) \pmod{q}$.
3. Compute $s \equiv k^{-1}(m + ar) \pmod{q}$.
4. Alice's signature for m is (r, s), which she sends to Bob along with m.

For Bob to verify, he must

1. Download Alice's public information (p, q, α, β).
2. Compute $u_1 \equiv s^{-1}m \pmod{q}$, and $u_2 \equiv s^{-1}r \pmod{q}$.
3. Compute $v = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$.
4. Accept the signature if and only if $v = r$.

Software (libraries, packages, tools):

For the realization of this practice the language that I used to carry out this practice was Java, and the IDE that I use netbeans, because I needed a graphic interface and it was easier for me to develop in netbeans.

I thought to present this practice in python, however I still need to study the graphic part of python, in the future I hope to have a better graphic interface.

And the last practices, that I will made like DES/ AES or HASH I made it in java

Libraries:

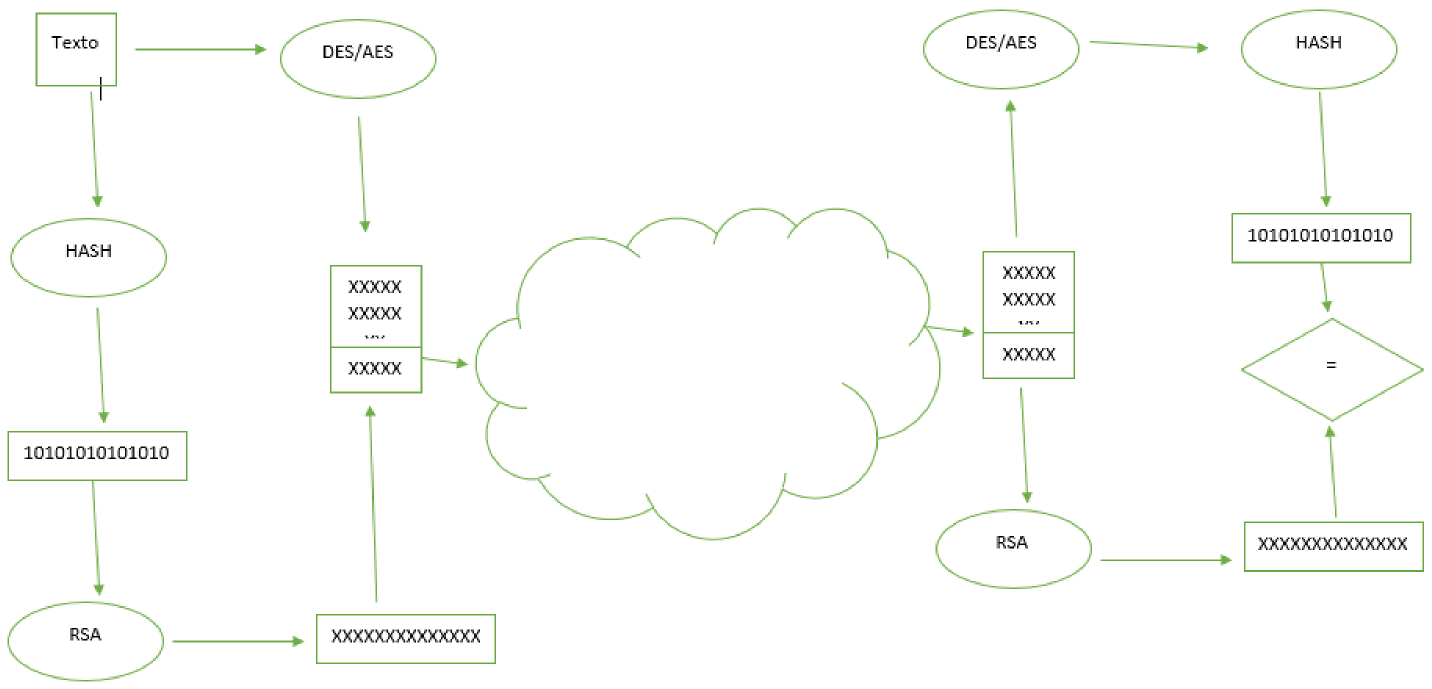
Make public and private key

```java
package firmadigitalfinal;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
```

Digital signature

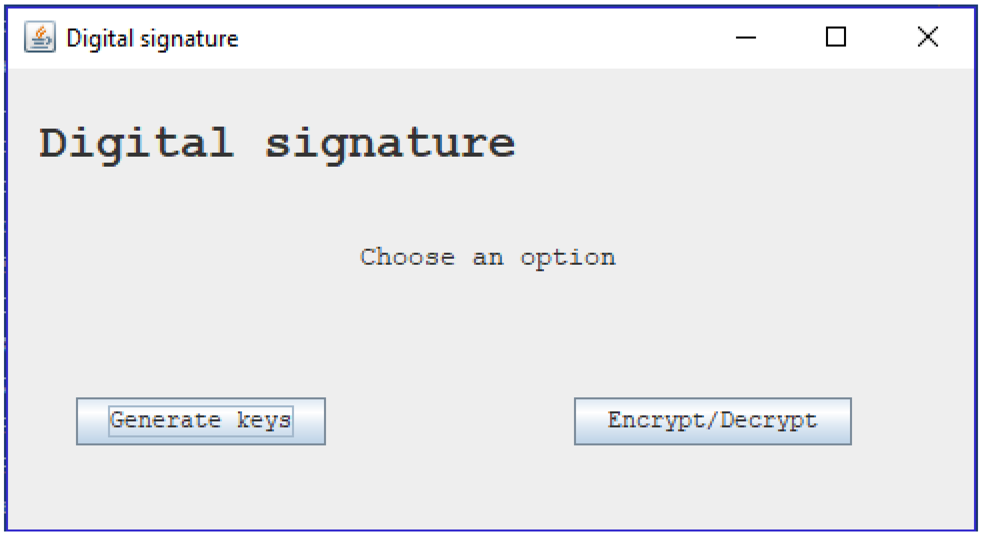
```java
package firmadigitalfinal;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.ShortBufferException;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
```

Procedure:

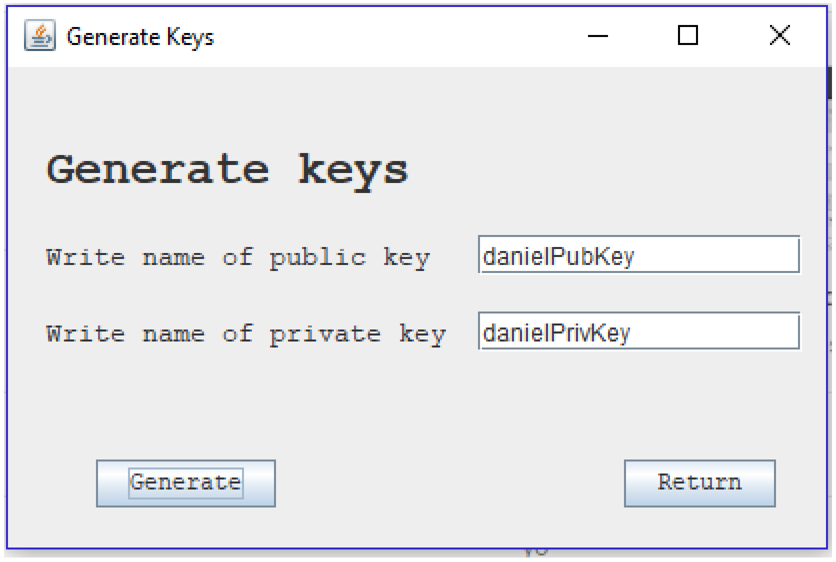


Results

My interface was like this.

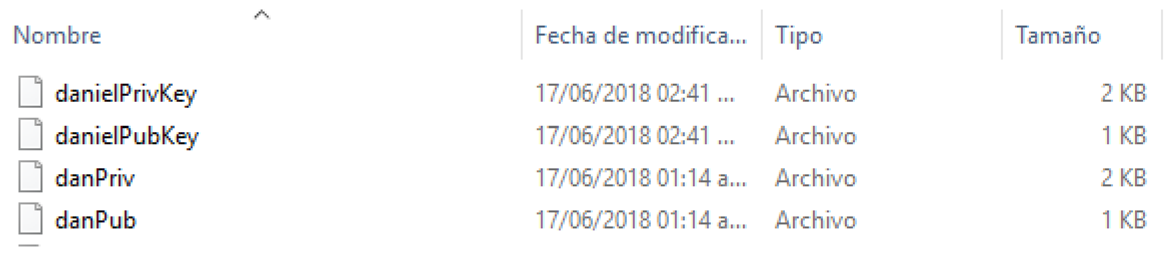


We need to generate keys to make the digital signature



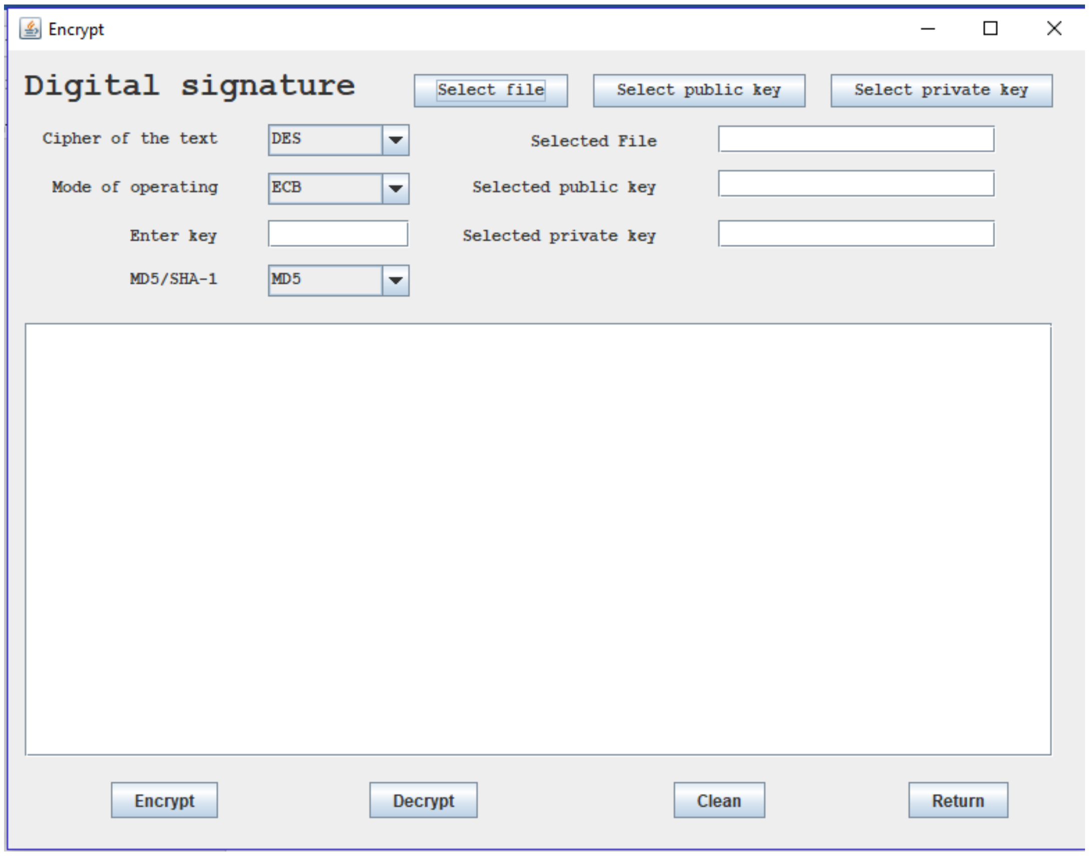
Write name of public key: danielPubKey
Write name of private key: danielPrivKey

Now, we have the keys,



Nombre	Fecha de modifica...	Tipo	Tamaño
danielPrivKey	17/06/2018 02:41 ...	Archivo	2 KB
danielPubKey	17/06/2018 02:41 ...	Archivo	1 KB
danPriv	17/06/2018 01:14 a...	Archivo	2 KB
danPub	17/06/2018 01:14 a...	Archivo	1 KB

Later we need to enter to Encrypt/Decrypt


Encrypt

Digital signature

Cipher of the text: DES

Mode of operating: ECB

Enter key:

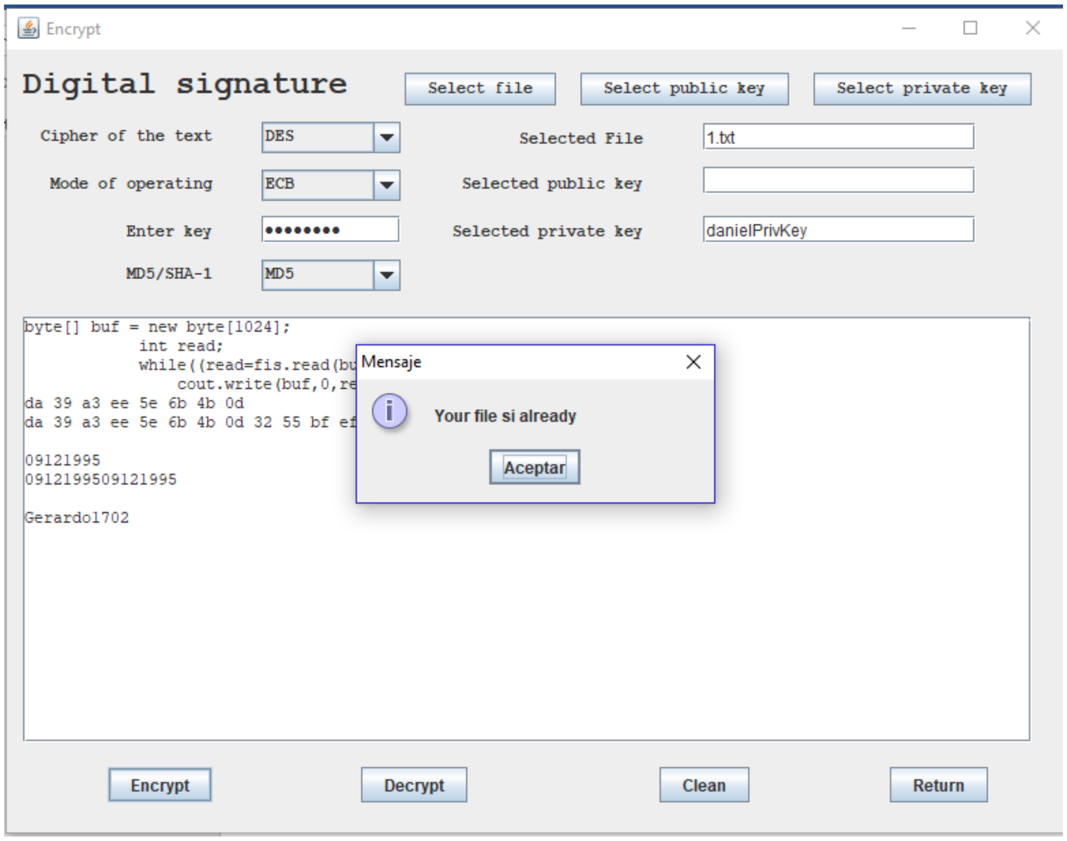
MD5/SHA-1: MD5

Selected File:

Selected public key:

Selected private key:

We need to select the file to encrypt, and the private key, to later select the type of cipher, mode of operation and his key


Encrypt

Digital signature

Cipher of the text: DES

Mode of operating: ECB

Enter key: ●●●●●●●●●

MD5/SHA-1: MD5

Selected File: 1.txt

Selected public key:

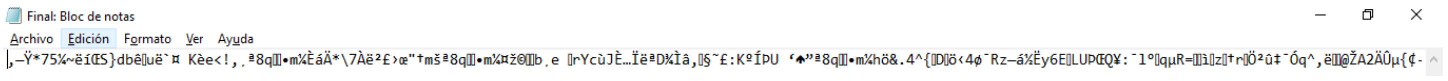
Selected private key: danielPrivKey

```
byte[] buf = new byte[1024];
            int read;
            while((read=fis.read(bu
                cout.write(buf,0,re
da 39 a3 ee 5e 6b 4b 0d
da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef

09121995
0912199509121995

Gerardo1702
```

Mensaje

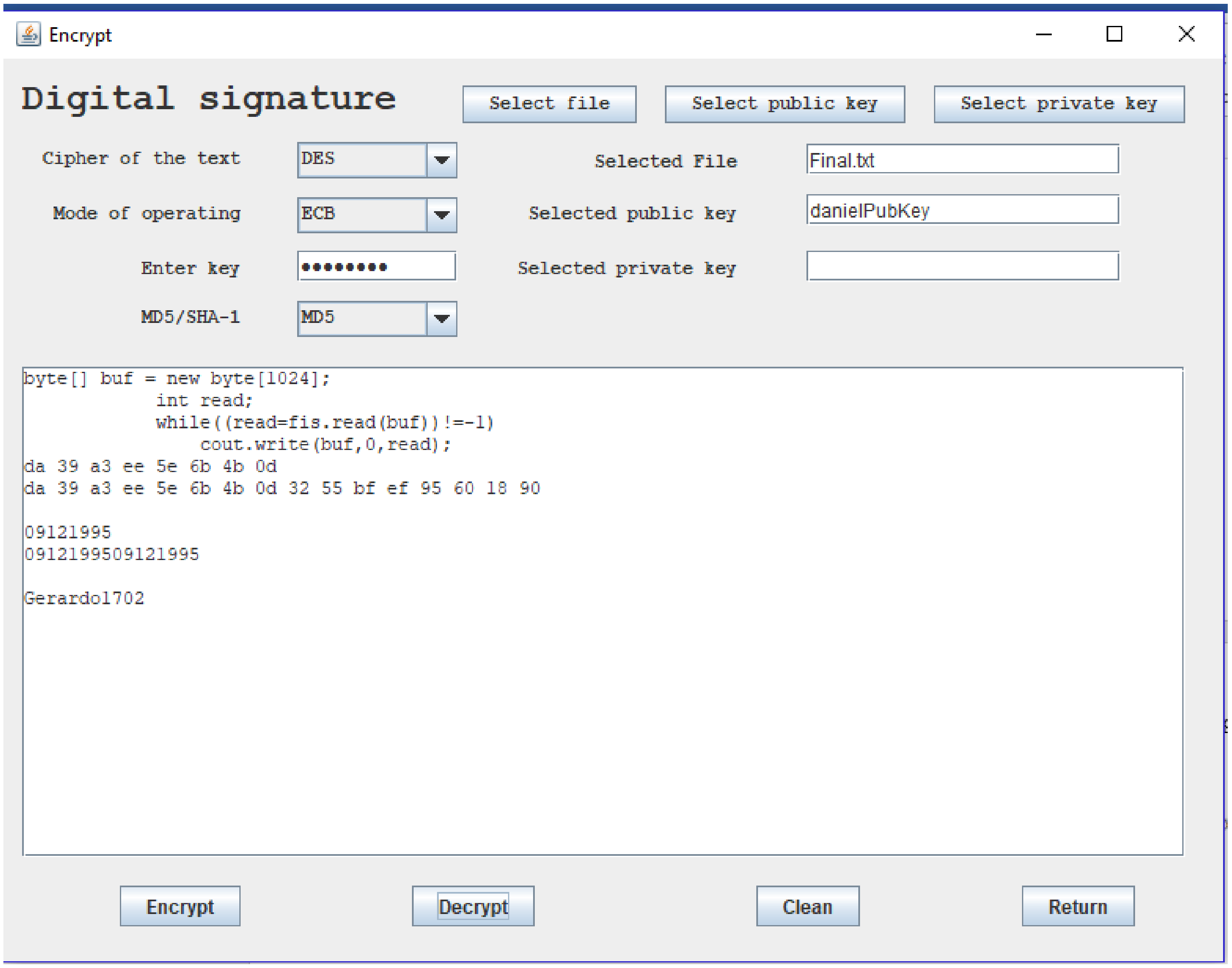
Your file si already

If we check the file, we don’t see something, now, we need to decrypt the message.



```
,–Ÿ*75¾~ëíŒS}dbê▯uë`¤ Kèe<!,¸ª8q▯▯•m¼ÈáÄ*\7Àë²£›œ"†mšª8q▯▯•m¼¤ž©▯▯b¸e ▯rYcùJÈ…ÏëªD¾Ìâ,▯§˜£:KºÍÞU ‘♠”ª8q▯▯•m¼hö&.4^{▯D▯ö‹4ϕ¯Rz–á½Ëy6E▯LUÞŒQ¥:¯1°▯qµR=▯▯ì▯z▯†r▯Ö²û‡¯Óq^,ë▯▯@ŽA2ÄÛµ{¢-
```

Now, we need to select the files to decrypt, and the correct selections



Cipher of the text: DES
Mode of operating: ECB
Enter key: ●●●●●●●●●
MD5/SHA-1: MD5
Selected File: Final.txt
Selected public key: danielPubKey
Selected private key:

```
byte[] buf = new byte[1024];
            int read;
            while((read=fis.read(buf))!=-1)
                cout.write(buf,0,read);
da 39 a3 ee 5e 6b 4b 0d
da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90

09121995
0912199509121995

Gerardo1702
```

Discussion:

With the use of AES or DES we know more about algorithms of cipher, we know all of this algorithm help us to cipher the information, the difference between AES and DES is in the year was developed, and the key length.
With this practice we could send messages in safe way.

Conclusions:

This final practice was very hard, because in the first time when I try to make, the person who I think I made this practice, he stopped entering classes, for that reason do not deliver the practice on time.
Later I’ll try to make this practice with another person, but, xD…
Well, this practice made mi think a lot.

References:

Knudsen, J. (1998). Java cryptography. Cambridge: O'Reilly.

Konheim, A. (2007). Computer security and cryptography. New York, N.Y.: Wiley.

Introduction to Cryptography with Coding Theory, 2nd edition By Wade Trappe and Lawrence C. Washington

Code

Generate keys

```java
package firmadigitalfinal;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;

public class GenerateKeys {
    private KeyPairGenerator keyGen;
    private KeyPair pair;
    private PrivateKey privateKey;
    private PublicKey publicKey;

    public GenerateKeys(int keylength) throws NoSuchAlgorithmException, NoSuchProviderException
    {
        final int keySize = 2048;
        this.keyGen = KeyPairGenerator.getInstance("RSA");
        this.keyGen.initialize(keySize);
    }

    public void createKeys()
    {
        this.pair = this.keyGen.generateKeyPair();
        this.privateKey = pair.getPrivate();
        this.publicKey = pair.getPublic();
    }

    public PrivateKey getPrivateKey()
    {
        System.out.println(privateKey);
        return this.privateKey;
    }

    public PublicKey getPublicKey()
    {
        System.out.println(publicKey);
        return this.publicKey;
    }

    public void writeToFile(String path, byte[] key) throws IOException
    {
        File f = new File(path);
        f.getParentFile().mkdirs();
        FileOutputStream fos = new FileOutputStream(f);
        fos.write(key);
        fos.flush();
        fos.close();
    }

    public static void main(String[] args)
    {
        GenerateKeys gk;
        try
        {
            final int keySize = 2048;
            gk = new GenerateKeys(keySize);
            gk.createKeys();
            gk.writeToFile("Keys/publicKey", gk.getPublicKey().getEncoded());
            gk.writeToFile("Keys/privateKey", gk.getPrivateKey().getEncoded());
        }
        catch (NoSuchAlgorithmException | NoSuchProviderException e)
        {
            System.err.println(e.getMessage());
        }
        catch (IOException e)
        {
            System.err.println(e.getMessage());
        }

    }

}
```

Make digital sign

```java
package firmadigitalfinal;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.ShortBufferException;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;


import javax.crypto.spec.IvParameterSpec;
import org.apache.commons.codec.binary.Base64;

public class Encrypt extends javax.swing.JFrame
{
    JFileChooser buscador = new JFileChooser();
    JFileChooser buscador2 = new JFileChooser();
    JFileChooser buscador3 = new JFileChooser();
    private static Cipher enc;
    private static Cipher dec;
    public String dig, comp, diges, compl;
    public static String filename, filename2, filename3, path, pathPublic, pathPrivate, txt, keyDES, keyAES, doc, pubKey, privKey, decrypted_msg;
    File archivo;
    File archivoPublicKey;
    File archivoPrivateKey;
    private File file;
    FileInputStream entrada;
    FileInputStream fis;
    //InitializationVector DES
    private static final byte[] iV = { 11, 22, 33, 44, 99, 88, 77, 66 };
    //InitializationVector AES
    private static final byte[] iVAES = { (byte)0xda, (byte)0x39, (byte)0xa3, (byte)0xee, (byte)0x5e, (byte)0x6b, (byte)0x4b, (byte)0x0d, (byte)0x32, (byte)0x55, (byte)0xbf, (byte)0xef, (byte)0x95, (byte)0x60, (byte)0x18, (byte)0x90};
    private AlgorithmParameterSpec algorithmParameterSpec;
    private AlgorithmParameterSpec algorithmParameterSpecAES;
    int auxSHAMD5;
    private Cipher cipher;
    String rsaOfText;
    private Cipher cip;
    byte[] rsaOfTxt;
    public Encrypt() throws NoSuchAlgorithmException, NoSuchPaddingException
    {
        super("Encrypt");
        initComponents();
        this.setLocationRelativeTo(null);
        algorithmParameterSpec = new IvParameterSpec(iV);
        algorithmParameterSpecAES =  new IvParameterSpec(iVAES);
        this.cip = Cipher.getInstance("RSA");
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        JBSelectFile = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jPasswordField1 = new javax.swing.JPasswordField();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jComboBox1 = new javax.swing.JComboBox();
        jComboBox3 = new javax.swing.JComboBox<>();
        jComboBox2 = new javax.swing.JComboBox();
        jLabel5 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        JBSelectFile1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel6 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel7 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        JBSelectFile2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        JBSelectFile.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
```

```
                JBSelectFile.setText("Select file");
                JBSelectFile.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        JBSelectFileActionPerformed(evt);
                    }
                });

                jLabel1.setFont(new java.awt.Font("Courier New", 1, 24)); // NOI18N
                jLabel1.setText("Digital signature");

                jPasswordField1.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jPasswordField1ActionPerformed(evt);
                    }
                });

                jLabel2.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jLabel2.setText("Mode of operating");

                jLabel3.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jLabel3.setText("Cipher of the text");

                jLabel4.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jLabel4.setText("Enter key");

                jComboBox1.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "ECB", "CBC", 
   "CFB", "OFB" }));
                jComboBox1.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jComboBox1ActionPerformed(evt);
                    }
                });

                jComboBox3.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jComboBox3.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "DES", "AES" 
    }));

                jComboBox2.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "MD5", "SHA-
   1" }));
                jComboBox2.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jComboBox2ActionPerformed(evt);
                    }
                });

                jLabel5.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
                jLabel5.setText("MD5/SHA-1");

                jButton1.setText("Encrypt");
                jButton1.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jButton1ActionPerformed(evt);
                    }
                });

                jButton2.setText("Decrypt");
                jButton2.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                        jButton2ActionPerformed(evt);
                    }
                });

                JBSelectFile1.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
```

```java
        JBSelectFile1.setText("Select public key");
        JBSelectFile1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                JBSelectFile1ActionPerformed(evt);
            }
        });

        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Courier New", 0, 12)); // NOI18N
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);

        jLabel6.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        jLabel6.setText("Selected File");

        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel7.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        jLabel7.setText("Selected public key");

        jTextField2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField2ActionPerformed(evt);
            }
        });

        jLabel8.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        jLabel8.setText("Selected private key");

        jTextField3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField3ActionPerformed(evt);
            }
        });

        JBSelectFile2.setFont(new java.awt.Font("Courier New", 1, 12)); // NOI18N
        JBSelectFile2.setText("Select private key");
        JBSelectFile2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                JBSelectFile2ActionPerformed(evt);
            }
        });

        jButton3.setText("Clean");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setText("Return");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```java
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(jScrollPane1)
                            .addGroup(layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel1)
                                    .addGroup(layout.createSequentialGroup()
                                        .addGap(13, 13, 13)
                                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                            .addGroup(layout.createSequentialGroup()
                                                .addComponent(jLabel3)
                                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                                .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE, 102, javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGroup(layout.createSequentialGroup()
                                                .addGap(7, 7, 7)
                                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                                    .addComponent(jLabel4)
                                                    .addComponent(jLabel2)
                                                    .addComponent(jLabel5))
                                                .addGap(36, 36, 36)
                                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                                        .addComponent(jComboBox2, 0, 102, Short.MAX_VALUE)
                                                        .addComponent(jPasswordField1))
                                                    .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 102, javax.swing.GroupLayout.PREFERRED_SIZE))))))
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(layout.createSequentialGroup()
                                        .addGap(3, 3, 3)
                                        .addComponent(JBSelectFile)
                                        .addGap(18, 18, 18)
                                        .addComponent(JBSelectFile1)
                                        .addGap(18, 18, 18)
                                        .addComponent(JBSelectFile2))
                                    .addGroup(layout.createSequentialGroup()
                                        .addGap(38, 38, 38)
                                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                            .addComponent(jLabel6)
                                            .addComponent(jLabel7)
                                            .addComponent(jLabel8))
                                        .addGap(44, 44, 44)
                                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 200, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, 200, javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 200, javax.swing.GroupLayout.PREFERRED_SIZE))))))
                        .addContainerGap(24, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(74, 74, 74)
                        .addComponent(jButton1)
                        .addGap(109, 109, 109)
                        .addComponent(jButton2)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton3)
                        .addGap(103, 103, 103)
                        .addComponent(jButton4)
                        .addGap(56, 56, 56))
                );
                layout.setVerticalGroup(
                    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel1)
                                    .addComponent(JBSelectFile)
                                    .addComponent(JBSelectFile1)
                                    .addComponent(JBSelectFile2))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel3)
                                    .addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(12, 12, 12)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel2)
                                    .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel7))
                                .addGap(11, 11, 11)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jPasswordField1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel4)
                                    .addComponent(jLabel8))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel5)))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(42, 42, 42)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel6))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(16, 16, 16)
                                .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(19, 19, 19)
                        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 312, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jButton1)
                            .addComponent(jButton2)
                            .addComponent(jButton3)
                            .addComponent(jButton4))
                        .addContainerGap(22, Short.MAX_VALUE))
                );

                pack();
            }// </editor-fold>

            private void JBSelectFileActionPerformed(java.awt.event.ActionEvent evt) {

                buscador.setCurrentDirectory(new File("C:\\Users\\Daniel\\Desktop"));
                if(buscador.showDialog(null,"Abrir")==JFileChooser.APPROVE_OPTION)
                {
                    archivo = buscador.getSelectedFile();
                    if(archivo.canRead())
                    {
                        if(archivo.getName().endsWith("txt"))
                        {
                            doc = AbrirArchivo(archivo);
                            filename = buscador.getSelectedFile().getName();
                            path = buscador.getSelectedFile().getAbsolutePath();
                            JOptionPane.showMessageDialog(null, "You selected " + filename);
                            jTextField1.setText(filename);
                            jTextArea1.setText(doc);
                            //jtextArea nos guarda el archivo
                        }
                        else
                        {
                            JOptionPane.showMessageDialog(null, "NOPE");
                        }
                    }
                }

            }

            public String AbrirArchivo(File archivo)
            {
                String doc="";
                try
                {
                    entrada = new FileInputStream(archivo);
                    int asc;
                    while((asc=entrada.read())!=-1)
                    {
                        char car = (char)asc;
                        doc += car;
                    }
                }
                catch (Exception e)
                {

                }
                return doc;
            }
            private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {

                // TODO add your handling code here:
            }

            private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {
                // TODO add your handling code here:
            }

            private void jPasswordField1ActionPerformed(java.awt.event.ActionEvent evt) {
                // TODO add your handling code here:
            }

            private void JBSelectFile1ActionPerformed(java.awt.event.ActionEvent evt) {
                jTextField2.setText(null);
                 jTextField3.setText(null);
                buscador2.setCurrentDirectory(new File("C:\\Users\\Daniel\\Desktop\\PracticaFinal\\Fir
 maDigitalFinal\\Keys"));
                if(buscador2.showDialog(null,"Abrir")==JFileChooser.APPROVE_OPTION)
                {
                    archivoPublicKey = buscador2.getSelectedFile();
                    if(archivoPublicKey.canRead())
                    {
                        pubKey = AbrirArchivo(archivoPublicKey);
                        //jTextArea1.setText(publicKey);
                        filename2 = buscador2.getSelectedFile().getName();
                        pathPublic = buscador2.getSelectedFile().getAbsolutePath();
                        JOptionPane.showMessageDialog(null, "You selected " + filename2);
                        jTextField2.setText(filename2);
                        //jTextArea1.setText(publicKey);
                        //jtextArea nos guarda el archivo

                }

            }
            }

            private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
                // TODO add your handling code here:
            }

            private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
                // TODO add your handling code here:
            }

            private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
                int chooseMode = jComboBox1.getSelectedIndex();
                int chooseCipher = jComboBox3.getSelectedIndex();
                int chooseMd5Sha1 = jComboBox2.getSelectedIndex();
                if(jTextArea1.getText().length() != 0 && jTextField3.getText().length() !=0)
                {
                    try
                    {
                        file=new File("C:\\Users\\Daniel\\encrypt.txt");
                        if(chooseCipher == 0)
                        {
                            FileOutputStream fos =new FileOutputStream(file);
                            if(jPasswordField1.getText().length() ==8)
                            {
                               System.out.println("----------------------------------------------------
 -------------------------------------");
                               //----------------------------------------
```

```java
char[] a = jPasswordField1.getPassword();
keyDES = (new String(a));
byte k[] = keyDES.getBytes();
SecretKeySpec skeySpec = new SecretKeySpec(k, "DES");
System.out.println("You choose DES");
try
{
    if(chooseMode == 0)
    {
        System.out.println("You choose ECB");
        try
        {
            if(chooseMd5Sha1==0)
            {
                try
                {
                    auxSHAMD5=0;
                    System.out.println("You choose MD5");
                    System.out.println("You choose Encrypt");
                    Cipher enc = Cipher.getInstance("DES/ECB/PKCS5Padding");
                    enc.init(Cipher.ENCRYPT_MODE, skeySpec);
                    cifrar(enc, fos, auxSHAMD5);

                }
                catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                {
                    e.printStackTrace();
                }

            }
            else if(chooseMd5Sha1 ==1)
            {
                try
                {
                     auxSHAMD5=1;
                     System.out.println("You choose SHA-1");
                     System.out.println("You choose Encrypt");
                     Cipher enc = Cipher.getInstance("DES/ECB/PKCS5Padding");
                     enc.init(Cipher.ENCRYPT_MODE, skeySpec);
                     cifrar(enc, fos, auxSHAMD5);
                }
                catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                {
                    e.printStackTrace();
                }
            }
        }
        catch(Exception e)
        {

        }
    }
    else if(chooseMode == 1)
    {
        System.out.println("You choose CBC");
        try
        {
            if(chooseMd5Sha1==0)
            {
                try
                {
```

```
                                                  auxSHAMD5=0;
                                                   System.out.println("You choose MD5");
                                                   System.out.println("You choose Encrypt");

                                                   Cipher enc = Cipher.getInstance("DES/CBC/PKCS5Padding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);

                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {
                                               try
                                              {
                                                  auxSHAMD5=1;
                                                   System.out.println("You choose SHA-1");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("DES/CBC/PKCS5Padding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);

                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                                  else if(chooseMode == 2)
                                  {
                                      System.out.println("You choose CFB");
                                      try
                                      {
                                          if(chooseMd5Sha1==0)
                                          {
                                              try
                                              {
                                                   auxSHAMD5=0;
                                                   System.out.println("You choose MD5");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("DES/CFB8/NoPadding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);
                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
```

```java
            }
        }
        else if(chooseMd5Sha1 ==1)
        {
            try
            {
                auxSHAMD5=1;
                 System.out.println("You choose SHA-1");
                 System.out.println("You choose Encrypt");
                 Cipher enc = Cipher.getInstance("DES/CFB8/NoPadding");
                 enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);
                 cifrar(enc, fos, auxSHAMD5);
            }
            catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
            {
                e.printStackTrace();
            }
        }
    }
    catch(Exception e)
    {

    }
}
else if(chooseMode == 3)
{
    System.out.println("You choose OFB");
    try
    {
        if(chooseMd5Sha1==0)
        {
            try
            {
                 auxSHAMD5=0;
                 System.out.println("You choose MD5");
                 System.out.println("You choose Encrypt");
                 Cipher enc = Cipher.getInstance("DES/OFB/NoPadding");
                 enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);
                 cifrar(enc, fos, auxSHAMD5);
            }
            catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
            {
                e.printStackTrace();
            }
        }
        else if(chooseMd5Sha1 ==1)
        {
           try
            {
                auxSHAMD5=1;
                 System.out.println("You choose SHA-1");
                 System.out.println("You choose Encrypt");
                 Cipher enc = Cipher.getInstance("DES/OFB/NoPadding");
                 enc.init(Cipher.ENCRYPT_MODE, skeySpec, algorithmParameterSpec);
                 cifrar(enc, fos, auxSHAMD5);
            }
```

```java
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                              }
                              catch(Exception e)
                              {

                              }
                           }
                           else
                           {
                               jPasswordField1.setText(null);
                               JOptionPane.showMessageDialog(null,"The length of the password is incorrect");
                           }
                       }
                       else if(chooseCipher == 1)
                       {
                           FileOutputStream fos =new FileOutputStream(file);
                           if(jPasswordField1.getText().length() == 16)
                           {

                               System.out.println("The length of the password is correct");
                               System.out.println("----------------------------------------------------------------------------------------");
                              //----------------------------------------
                               char[] c = jPasswordField1.getPassword();
                               System.out.println(c);
                               keyAES = (new String(c));
                               System.out.println(keyAES);
                               byte key[] = keyAES.getBytes();
                               SecretKeySpec skeySpec2 = new SecretKeySpec(key, "AES");
                               System.out.println("You choose AES");
                               try
                               {

                                   if(chooseMode == 0)
                                   {
                                       System.out.println("You choose ECB");
                                       try
                                       {

                                           if(chooseMd5Sha1==0)
                                           {

                                               try
                                               {
                                                   auxSHAMD5=0;
                                                   System.out.println("You choose MD5");
                                                   System.out.println("You choose Encrypt");
                                                   enc = Cipher.getInstance("AES/ECB/PKCS5Padding");

                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2);
                                                   cifrar(enc, fos, auxSHAMD5);
                                               }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                   e.printStackTrace();
                                              }

                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {

                                              try
                                              {
                                                  auxSHAMD5=1;
                                                  System.out.println("You choose SHA-1");
                                                  System.out.println("You choose Encrypt");
                                                  enc = Cipher.getInstance("AES/ECB/PKCS5Padding");

                                                  enc.init(Cipher.ENCRYPT_MODE, skeySpec2);
                                                  cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                   e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                                 else if(chooseMode == 1)
                                 {
                                     System.out.println("You choose CBC");
                                     try
                                     {

                                         if(chooseMd5Sha1==0)
                                         {

                                             try
                                             {
                                                  auxSHAMD5=0;
                                                  System.out.println("You choose MD5");
                                                  System.out.println("You choose Encrypt");
                                                  enc = Cipher.getInstance("AES/CBC/PKCS5Padding");

                                                  enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                  cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                   e.printStackTrace();
                                              }
                                         }
                                         else if(chooseMd5Sha1 ==1)
                                         {
                                              try
                                             {
                                                 auxSHAMD5=  1;
                                                  System.out.println("You choose SHA-1");
                                                  System.out.println("You choose Encrypt");
```

```
                                                   enc = Cipher.getInstance("AES/CBC/PKCS5Padding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                   cifrar(enc, fos, auxSHAMD5);
                                               }
                                               catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                               {
                                                    e.printStackTrace();
                                               }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                                  else if(chooseMode == 2)
                                  {
                                      System.out.println("You choose CFB");
                                      try
                                      {
                                          if(chooseMd5Sha1==0)
                                          {
                                              try
                                              {
                                                  auxSHAMD5=0;
                                                   System.out.println("You choose MD5");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("AES/CFB8/NoPadding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {
                                              try
                                              {
                                                  auxSHAMD5=1;
                                                   System.out.println("You choose SHA-1");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("AES/CFB8/NoPadding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                                  else if(chooseMode == 3)
                                  {
                                      System.out.println("You choose OFB");
                                      try
                                      {
                                          if(chooseMd5Sha1==0)
                                          {
                                              try
                                              {
                                                  auxSHAMD5=0;
                                                   System.out.println("You choose MD5");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("AES/OFB8/NoPadding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {
                                             try
                                              {
                                                  auxSHAMD5=1;
                                                   System.out.println("You choose SHA-1");
                                                   System.out.println("You choose Encrypt");
                                                   Cipher enc = Cipher.getInstance("AES/OFB8/NoPadding");
                                                   enc.init(Cipher.ENCRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                   cifrar(enc, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | IOException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                              }
                              catch(Exception e)
                              {

                              }
                           }
                           else
                           {
                               jPasswordField1.setText(null);
                               JOptionPane.showMessageDialog(null,"The length of the password is incorrect");
                           }
                       }

```

```
                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }
            }
            else
            {
                JOptionPane.showMessageDialog(null,"Please select private key");
            }
        }

        private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {

        }

        private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

            int chooseMode = jComboBox1.getSelectedIndex();
            int chooseCipher = jComboBox3.getSelectedIndex();
            int chooseMd5Sha1 = jComboBox2.getSelectedIndex();
            if(jTextField2.getText().length() !=0)
            {
                try
                {
                    file=new File("C:\\Users\\Daniel\\descifrado.txt");
                    if(chooseCipher == 0)
                    {

                        FileOutputStream fos =new FileOutputStream(file);
                        if(jPasswordField1.getText().length() ==8)
                        {
                            jTextArea1.setText(null);
                           System.out.println("The length of the password is correct");
                           System.out.println("----------------------------------------------------
------------------------------------");
                           //----------------------------------------
                           char[] a = jPasswordField1.getPassword();
                           System.out.println(a);
                           keyDES = (new String(a));
                           System.out.println(keyDES);
                           byte k[] = keyDES.getBytes();
                           SecretKeySpec skeySpec = new SecretKeySpec(k, "DES");
                           System.out.println("You choose DES");
                           try
                           {
                               if(chooseMode == 0)
                               {
                                   System.out.println("You choose ECB");
                                   try
                                   {
                                       if(chooseMd5Sha1==0)
                                       {
                                           try
                                           {
                                                auxSHAMD5=0;
                                                System.out.println("You choose MD5");
                                                System.out.println("You choose Decrypt");
                                                Cipher dec = Cipher.getInstance("DES/ECB/PKCS5Padd
ing");
                                                dec.init(Cipher.DECRYPT_MODE, skeySpec);
                                                descifrar(dec, fos, auxSHAMD5);
                                           }
                                           catch(InvalidKeyException | NoSuchAlgorithmException |
NoSuchPaddingException e)
```

```
                                              {
                                                  e.printStackTrace();
                                              }

                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {
                                              try
                                              {

                                                  auxSHAMD5=1;
                                                  System.out.println("You choose SHA-1");
                                                  System.out.println("You choose Decrypt");
                                                  Cipher dec = Cipher.getInstance("DES/ECB/PKCS5Padding");
                                                  dec.init(Cipher.DECRYPT_MODE, skeySpec);
                                                  descifrar(dec, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                      }
                                      catch(Exception e)
                                      {

                                      }
                                  }
                                  else if(chooseMode == 1)
                                  {
                                      System.out.println("You choose CBC");
                                      try
                                      {
                                          if(chooseMd5Sha1==0)
                                          {
                                              try
                                              {
                                                  auxSHAMD5=0;
                                                  System.out.println("You choose MD5");
                                                  System.out.println("You choose Decrypt");
                                                  Cipher dec = Cipher.getInstance("DES/CBC/PKCS5Padding");
                                                  dec.init(Cipher.DECRYPT_MODE, skeySpec, algorithmParameterSpec);
                                                  descifrar(dec, fos, auxSHAMD5);
                                              }
                                              catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                              {
                                                  e.printStackTrace();
                                              }
                                          }
                                          else if(chooseMd5Sha1 ==1)
                                          {
                                               try
                                              {
                                                  auxSHAMD5=1;
                                                  System.out.println("You choose SHA-1");
                                                  System.out.println("You choose Decrypt");
                                                  Cipher dec = Cipher.getInstance("DES/CBC/PKCS5Padding");
                                                  dec.init(Cipher.DECRYPT_MODE, skeySpec, algorithmParameterSpec);
```

```
                                        descifrar(dec, fos, auxSHAMD5);
                                    }
                                    catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                    {
                                        e.printStackTrace();
                                    }
                                }
                            }
                            catch(Exception e)
                            {

                            }
                        }
                        else if(chooseMode == 2)
                        {
                            System.out.println("You choose CFB");
                            try
                            {
                                if(chooseMd5Sha1==0)
                                {
                                    try
                                    {
                                        auxSHAMD5=0;
                                        System.out.println("You choose MD5");
                                        System.out.println("You choose Decrypt");
                                        Cipher dec = Cipher.getInstance("DES/CFB8/NoPadding");
                                        dec.init(Cipher.DECRYPT_MODE, skeySpec, algorithmParameterSpec);
                                        descifrar(dec, fos, auxSHAMD5);
                                    }
                                    catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                    {
                                        e.printStackTrace();
                                    }
                                }
                                else if(chooseMd5Sha1 ==1)
                                {
                                    try
                                    {
                                        auxSHAMD5=1;
                                        System.out.println("You choose SHA-1");
                                        System.out.println("You choose Decrypt");
                                        Cipher dec = Cipher.getInstance("DES/CFB8/NoPadding");
                                        dec.init(Cipher.DECRYPT_MODE, skeySpec, algorithmParameterSpec);
                                        descifrar(dec, fos, auxSHAMD5);
                                    }
                                    catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                    {
                                        e.printStackTrace();
                                    }
                                }
                            }
                            catch(Exception e)
                            {

                            }
                        }
                        else if(chooseMode == 3)
                        {
```

```
               System.out.println("You choose OFB");
               try
               {
                   if(chooseMd5Sha1==0)
                   {
                       try
                       {
                           auxSHAMD5=0;
                           System.out.println("You choose MD5");
                           System.out.println("You choose Decrypt");
                           Cipher dec = Cipher.getInstance("DES/OFB/NoPadding");
                           dec.init(Cipher.DECRYPT_MODE , skeySpec, algorithmParameterSpec);
                           descifrar(dec, fos, auxSHAMD5);

                       }
                       catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                       {
                           e.printStackTrace();
                       }
                   }
                   else if(chooseMd5Sha1 ==1)
                   {
                      try
                       {
                           auxSHAMD5=1;
                           System.out.println("You choose SHA-1");
                           System.out.println("You choose Decrypt");
                           Cipher dec = Cipher.getInstance("DES/OFB/NoPadding");
                           dec.init(Cipher.DECRYPT_MODE , skeySpec, algorithmParameterSpec);
                           descifrar(dec, fos, auxSHAMD5);
                       }
                       catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                       {
                           e.printStackTrace();
                       }
                   }
               }
               catch(Exception e)
               {

               }
           }
       }
       catch(Exception e)
       {

       }
    }
    else
    {
        jPasswordField1.setText(null);
        JOptionPane.showMessageDialog(null,"The length of the password is incorrect");
    }
}
else if(chooseCipher == 1)
{
    FileOutputStream fos =new FileOutputStream(file);
    if(jPasswordField1.getText().length() == 16)
```

```java
{

    System.out.println("The length of the password is correct");
    System.out.println("------------------------------------------------------------------------------------------");
   //----------------------------------------
    char[] c = jPasswordField1.getPassword();
    System.out.println(c);
    keyAES = (new String(c));
    System.out.println(keyAES);
    byte key[] = keyAES.getBytes();
    SecretKeySpec skeySpec2 = new SecretKeySpec(key, "AES");
    System.out.println("You choose AES");
    try
    {

        if(chooseMode == 0)
        {
            System.out.println("You choose ECB");
            try
            {

                if(chooseMd5Sha1==0)
                {
                    try
                    {
                        auxSHAMD5=0;
                        System.out.println("You choose MD5");
                        System.out.println("You choose Decrypt");
                        dec = Cipher.getInstance("AES/ECB/PKCS5Padding");
                        dec.init(Cipher.DECRYPT_MODE , skeySpec2);
                        descifrar(dec, fos, auxSHAMD5);
                    }
                    catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                    {
                         e.printStackTrace();
                    }

                }
                else if(chooseMd5Sha1 ==1)
                {
                    try
                    {
                        auxSHAMD5=1;
                        System.out.println("You choose SHA-1");
                        System.out.println("You choose Decrypt");
                        dec = Cipher.getInstance("AES/ECB/PKCS5Padding");
                        dec.init(Cipher.DECRYPT_MODE , skeySpec2);
                        descifrar(dec, fos, auxSHAMD5);
                    }
                    catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                    {
                         e.printStackTrace();
                    }
                }
            }
            catch(Exception e)
            {

            }
```

```java
                                 }
                                else if(chooseMode == 1)
                                {
                                    System.out.println("You choose CBC");
                                    try
                                    {
                                        if(chooseMd5Sha1==0)
                                        {
                                            try
                                            {
                                                auxSHAMD5=0;
                                                System.out.println("You choose MD5");
                                                System.out.println("You choose Decrypt");
                                                dec = Cipher.getInstance("AES/CBC/PKCS5Padding");
                                                dec.init(Cipher.DECRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                descifrar(dec, fos, auxSHAMD5);
                                             }
                                             catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                             {
                                                  e.printStackTrace();
                                             }
                                        }
                                        else if(chooseMd5Sha1 ==1)
                                        {
                                            try
                                            {
                                                auxSHAMD5=1;
                                                System.out.println("You choose MD5");
                                                System.out.println("You choose Decrypt");
                                                dec = Cipher.getInstance("AES/CBC/PKCS5Padding");
                                                dec.init(Cipher.DECRYPT_MODE , skeySpec2, algorithmParameterSpecAES);
                                                descifrar(dec, fos, auxSHAMD5);
                                             }
                                             catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                             {
                                                  e.printStackTrace();
                                             }
                                        }
                                    }
                                    catch(Exception e)
                                    {

                                    }
                                }
                                else if(chooseMode == 2)
                                {
                                    System.out.println("You choose CFB");
                                    try
                                    {
                                        if(chooseMd5Sha1==0)
                                        {
                                            try
                                            {
                                                auxSHAMD5=0;
                                                System.out.println("You choose MD5");
                                                System.out.println("You choose Decrypt");
                                                Cipher dec = Cipher.getInstance("AES/CFB8/NoPadding");
                                                dec.init(Cipher.DECRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                                                descifrar(dec, fos,auxSHAMD5);
```

```
            }
            catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
            {
                e.printStackTrace();
            }
        }
        else if(chooseMd5Sha1 ==1)
        {
            try
            {
                auxSHAMD5=1;
                System.out.println("You choose SHA-1");
                System.out.println("You choose Decrypt");
                Cipher dec = Cipher.getInstance("AES/CFB8/NoPadding");
                dec.init(Cipher.DECRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                descifrar(dec, fos, auxSHAMD5);
            }
            catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
            {
                e.printStackTrace();
            }
        }
    }
    catch(Exception e)
    {

    }
}
else if(chooseMode == 3)
{
    System.out.println("You choose OFB");
    try
    {
        if(chooseMd5Sha1==0)
        {
            try
            {
                auxSHAMD5=0;
                System.out.println("You choose MD5");
                System.out.println("You choose Encrypt");
                Cipher dec = Cipher.getInstance("AES/OFB8/NoPadding");
                dec.init(Cipher.DECRYPT_MODE, skeySpec2, algorithmParameterSpecAES);
                descifrar(dec, fos, auxSHAMD5);
            }
            catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
            {
                e.printStackTrace();
            }
        }
        else if(chooseMd5Sha1 ==1)
        {
           try
            {
                auxSHAMD5=1;
                System.out.println("You choose SHA-1");
                System.out.println("You choose Encrypt");
                Cipher dec = Cipher.getInstance("AES/OFB8/NoPadding");
```

```
                                               dec.init(Cipher.DECRYPT_MODE , skeySpec2, algorithmParameterSpecAES);
                                               descifrar(dec, fos, auxSHAMD5);
                                           }
                                           catch(InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException e)
                                           {
                                               e.printStackTrace();
                                           }
                                       }
                                   }
                                   catch(Exception e)
                                   {

                                   }
                               }
                           }
                           catch(Exception e)
                           {

                           }
                        }
                        else
                        {
                            jPasswordField1.setText(null);
                            JOptionPane.showMessageDialog(null,"The length of the password is incorrect");
                        }
                    }

                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }
            }
            else
            {
                JOptionPane.showMessageDialog(null,"Please select public key");
            }
        }

        private void JBSelectFile2ActionPerformed(java.awt.event.ActionEvent evt) {

            jTextField2.setText(null);
             jTextField3.setText(null);
            buscador3.setCurrentDirectory(new File("C:\\Users\\Daniel\\Desktop\\PracticaFinal\\FirmaDigitalFinal\\Keys"));
            if(buscador3.showDialog(null,"Abrir")==JFileChooser.APPROVE_OPTION)
            {
                archivoPrivateKey = buscador3.getSelectedFile();
                if(archivoPrivateKey.canRead())
                {
                    privKey = AbrirArchivo(archivoPrivateKey);
                    //jTextArea1.setText(publicKey);
                    filename3 = buscador3.getSelectedFile().getName();
                    pathPrivate = buscador3.getSelectedFile().getAbsolutePath();
                    JOptionPane.showMessageDialog(null, "You selected " + filename3);
                    jTextField3.setText(filename3);
                    //jTextArea1.setText(publicKey);
                    //jtextArea nos guarda el archivo

                }
            }
        }

        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

            jPasswordField1.setText(null);
            jTextField1.setText(null);
            jTextField2.setText(null);
            jTextField3.setText(null);
            jTextArea1.setText(null);
        }

        private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

            // TODO add your handling code here:
            new Menu().setVisible(true);
            //Cierra la ventana actual
            dispose();
        }

        private void cifrar(Cipher enc, FileOutputStream fos, int auxSHAMD5) throws IOException, IllegalBlockSizeException, BadPaddingException, ShortBufferException, Exception
        {
            try
            {
                FileInputStream fis =new FileInputStream(path);
                CipherOutputStream cout=new CipherOutputStream(fos, enc);
                byte[] buf = new byte[1024];
                int read;
                while((read=fis.read(buf))!=-1)
                    cout.write(buf,0,read);
                fis.close();
                cout.flush();
                cout.close();

            }
            catch(Exception e)
            {
                e.printStackTrace();

            }
            if(auxSHAMD5==0)
            {

                System.out.println("Elegiste MD55555");
                dig = Encrypt.md5(doc);
                comp = (doc + "\n" + dig);
                //jTextArea1.setText(comp);
                //System.out.println(dig);
                sign(dig);
            }
            else if(auxSHAMD5==1)
            {
                System.out.println("Elegiste SHAAAAAAAAA");
                diges = Encrypt.sha1(doc);
                System.out.println(diges);
                compl = (doc + "\n" + diges);
                //jTextArea1.setText(compl);
                sign(diges);
            }
        }

        public static void sign(String dig) throws NoSuchAlgorithmException, Exception
        {
            Encrypt en = new Encrypt();
            PrivateKey privateKey = en.getPrivate(pathPrivate);
            String encrypted = en.encrypt(privateKey, dig);
        System.out.println("Original Message: " + dig + "\nEncrypted Message: " + encrypted);
            try
            {
                File inFile = new File("C:\\Users\\Daniel\\encrypt.txt");
                File outFile = new File("C:\\Users\\Daniel\\Desktop\\Archivos\\Final.txt");
                FileInputStream in = new FileInputStream(inFile);
                FileOutputStream out = new FileOutputStream(outFile);
                int c;
                while( (c = in.read() ) != -1)
                out.write(c);
                in.close();
                out.close();
        } catch(IOException e) {
                System.err.println("Hubo un error de entrada/salida!!!");
            }
            FileWriter fichero = null;
            PrintWriter pw = null;
            try
            {
                fichero = new FileWriter("C:\\Users\\Daniel\\Desktop\\Archivos\\Final.txt",true);

                pw = new PrintWriter(fichero);
                pw.println("\n"+encrypted);

            } catch (Exception e) {
                e.printStackTrace();
            } finally {
               try {
               if (null != fichero)
                  fichero.close();
               } catch (Exception e2) {
                  e2.printStackTrace();
               }
            }
            JOptionPane.showMessageDialog(null,"Your file si already");
        }
        public static String encrypt(PrivateKey privateKey, String message) throws Exception
        {
            Cipher cip = Cipher.getInstance("RSA");
            cip.init(Cipher.ENCRYPT_MODE, privateKey);
            return Base64.encodeBase64String(cip.doFinal(message.getBytes("UTF-8")));
        }


        public PrivateKey getPrivate(String filename) throws Exception
        {
        byte[] keyBytes = Files.readAllBytes(new File(filename).toPath());
        PKCS8EncodedKeySpec spec2 = new PKCS8EncodedKeySpec(keyBytes);
        KeyFactory kf = KeyFactory.getInstance("RSA");
        return kf.generatePrivate(spec2);
        }
        public static String getHash(String txt, String hashType)
        {
            try
            {
                java.security.MessageDigest md = java.security.MessageDigest.getInstance(hashType)
;
                byte[] array = md.digest(txt.getBytes());
                StringBuffer sb = new StringBuffer();
                for (int i = 0; i < array.length; ++i)
                {
                    sb.append(Integer.toHexString((array[i] & 0xFF) | 0x100).substring(1, 3));
                }
                return sb.toString();
            }
            catch (java.security.NoSuchAlgorithmException e)
            {
                System.out.println(e.getMessage());
            }
            return null;
        }

        /* Retorna un hash MD5 a partir de un texto */
        public static String md5(String txt)
        {
            return Encrypt.getHash(txt, "MD5");
        }

        /* Retorna un hash SHA1 a partir de un texto */
        public static String sha1(String txt)
        {
            return Encrypt.getHash(txt, "SHA1");
        }

        private void descifrar(Cipher dec, FileOutputStream fos, int auxSHAMD5)
        {
            try
            {
                FileInputStream fis =new FileInputStream("C:\\Users\\Daniel\\encrypt.txt");
                CipherOutputStream cout=new CipherOutputStream(fos, dec);
                byte[] buf = new byte[4096];
                int read;
                while((read=fis.read(buf))!=-1)
                    cout.write(buf,0,read);
                fis.close();
                cout.flush();
                cout.close();
            }
            catch(Exception e)
            {
                e.printStackTrace();

            }

                BufferedReader br = null;
                String[] cad = new String[100];
                int lin = 0;
                try
                {
                    br = new BufferedReader(new FileReader(archivo));
                    while((cad[lin] = br.readLine()) != null)
                    {
                        lin++;
                    }
                    System.out.println("Total lines: " +lin);
                    rsaOfText= cad[lin-1];
                    //byte[] rsaOfTxt = rsaOfText.getBytes();
                    chechSign(rsaOfText);
                }
                catch (Exception e)
                {
                }
                File archivodesc = null;
                archivodesc = new File ("C:\\Users\\Daniel\\descifrado.txt");
                doc = AbrirArchivo(archivodesc);
             System.out.println(doc);
            if(auxSHAMD5==0)
            {
                System.out.println("Elegiste MD55555");
                dig = Encrypt.md5(doc);
```

```
                    //comp = (doc + "\n" + dig);
                    //jTextArea1.setText(comp);
                    System.out.println(dig);
                    //sign(dig);
                }
                else if(auxSHAMD5==1)
                {
                    System.out.println("Elegiste SHAAAAAAAAA");
                    diges = Encrypt.sha1(doc);
                    System.out.println(diges);
                    //compl = (doc + "\n" + diges);
                    //jTextArea1.setText(compl);
                    //sign(diges);
                }
                jTextArea1.setText("Error! :D algo anda maaaaaaaaaaaaaaaaaal");
                if(dig.equals(decrypted_msg))
                {
                   jTextArea1.setText(doc);
                }
                else if(diges.equals(decrypted_msg))
                {
                   jTextArea1.setText(doc);
                }


            }
            public static void chechSign(String rsaOfTxt) throws NoSuchAlgorithmException, NoSuchPaddingException, Exception
            {
                Encrypt en = new Encrypt();
                PublicKey publicKey = en.getPublic(pathPublic);
                System.out.println(rsaOfTxt);
                //System.out.println(publicKey);
                decrypted_msg = en.decryptText(rsaOfTxt, publicKey);
                //Aquí se obtiene el hash con la llave publica :D
                System.out.println("hash: "+decrypted_msg);

            }
            public String decryptText(String msg, PublicKey key) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, UnsupportedEncodingException
            {
                Cipher cip = Cipher.getInstance("RSA");
                cip.init(Cipher.DECRYPT_MODE, key);
            return new String(cip.doFinal(Base64.decodeBase64(msg)), "UTF-8");
            }
            public PublicKey getPublic(String filename) throws Exception
            {
                //System.out.println(filename);
            byte[] keyBytes = Files.readAllBytes(new File(filename).toPath());
            X509EncodedKeySpec spec3 = new X509EncodedKeySpec(keyBytes);
            KeyFactory kf2 = KeyFactory.getInstance("RSA");
            return kf2.generatePublic(spec3);
            }
            /**
             * @param args the command line arguments
             */
            public static void main(String args[]) {

                try {
                    for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                        if ("Nimbus".equals(info.getName())) {
                            javax.swing.UIManager.setLookAndFeel(info.getClassName());
                            break;
                        }
                    }
                } catch (ClassNotFoundException ex) {
                    java.util.logging.Logger.getLogger(Encrypt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
                } catch (InstantiationException ex) {
                    java.util.logging.Logger.getLogger(Encrypt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
                } catch (IllegalAccessException ex) {
                    java.util.logging.Logger.getLogger(Encrypt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
                } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                    java.util.logging.Logger.getLogger(Encrypt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
                }
                //</editor-fold>
                //</editor-fold>

                /* Create and display the form */
                java.awt.EventQueue.invokeLater(new Runnable() {
                    public void run() {
                        try {
                            new Encrypt().setVisible(true);
                        } catch (NoSuchAlgorithmException ex) {
                            Logger.getLogger(Encrypt.class.getName()).log(Level.SEVERE, null, ex);
                        } catch (NoSuchPaddingException ex) {
                            Logger.getLogger(Encrypt.class.getName()).log(Level.SEVERE, null, ex);
                        }
                    }
                });
            }


            // Variables declaration - do not modify
            private javax.swing.JButton JBSelectFile;
            private javax.swing.JButton JBSelectFile1;
            private javax.swing.JButton JBSelectFile2;
            private javax.swing.JButton jButton1;
            private javax.swing.JButton jButton2;
            private javax.swing.JButton jButton3;
            private javax.swing.JButton jButton4;
            private javax.swing.JComboBox jComboBox1;
            private javax.swing.JComboBox jComboBox2;
            private javax.swing.JComboBox<String> jComboBox3;
            private javax.swing.JLabel jLabel1;
            private javax.swing.JLabel jLabel2;
            private javax.swing.JLabel jLabel3;
            private javax.swing.JLabel jLabel4;
            private javax.swing.JLabel jLabel5;
            private javax.swing.JLabel jLabel6;
            private javax.swing.JLabel jLabel7;
            private javax.swing.JLabel jLabel8;
            private javax.swing.JPasswordField jPasswordField1;
            private javax.swing.JScrollPane jScrollPane1;
            private javax.swing.JTextArea jTextArea1;
            private javax.swing.JTextField jTextField1;
            private javax.swing.JTextField jTextField2;
            private javax.swing.JTextField jTextField3;
            // End of variables declaration
        }
```