Joel Amores DAM2

Pràctica 5. Generació claus simètriques.

Encriptació i desencriptació de text en fitxers.

Part 1. Generació:

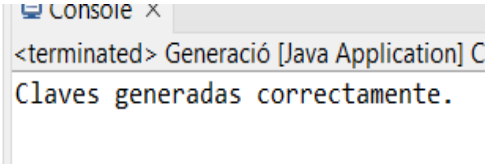
Aquesta primera part genera dues claus: clau pública i clau privada en dos fitxers diferents.

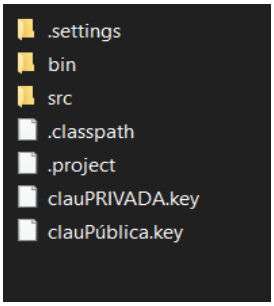
(He fet servir constants per poder modificar fàcilment els noms de sortida, tamanys…)

```java
public class Generació {

    public static final String CLAU_PUBLICA = "clauPública.key";
    public static final String CLAU_PRIVADA = "clauPRIVADA.key";
    public static final int TAMANYO_CLAVES = 2048; // Tamaños = 512, 1024 (default), 2048 (considerada segura), 4096...
    public static final String ALGORITME = "RSA";

    public static void main(String[] args) throws NoSuchAlgorithmException, NoSuchPaddingException, FileNotFoundException, IOException {

        // Esta libreria sirve para crear un par de claves (pública y privada)
        KeyPairGenerator generadorClau = KeyPairGenerator.getInstance(ALGORITME);
        generadorClau.initialize(TAMANYO_CLAVES);

        KeyPair claves = generadorClau.generateKeyPair();
        guardarClaves(claves, CLAU_PUBLICA, CLAU_PRIVADA);

        System.out.println("Claves generadas correctamente.");
    }
```

```java
    /**
     * Guarda en los archivos especificados las claves pública y privada
     * @param keyPair
     * @param archClavePublica, nombre del archivo donde se guarda la clave pública
     * @param archClavePrivada, nombre del archivo donde se guarda la clave privada
     * @throws IOException
     */
    public static void guardarClaves(KeyPair keyPair, String archClavePublica, String archClavePrivada) throws IOException {

        // Separar las claves
        PrivateKey clavePrivada = keyPair.getPrivate();
        PublicKey clavePublica = keyPair.getPublic();

        // Guardar clave pública
        X509EncodedKeySpec x509EncodedKeySpec = new X509EncodedKeySpec(clavePublica.getEncoded());
        FileOutputStream fos = new FileOutputStream(archClavePublica);
        fos.write(x509EncodedKeySpec.getEncoded());
        fos.close();

        // Guardar clave privada
        PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(clavePrivada.getEncoded());
        fos = new FileOutputStream(archClavePrivada);
        fos.write(pkcs8EncodedKeySpec.getEncoded());
        fos.close();
    }
```

A l'executar el programa ens mostra un missatge de confirmació i podem veure que ens ha generat els dos arxius:





Part 2: Encriptació

Aquesta segona part ens demana el nom de l'arxiu on tenim guardada la clau pública.

També demana un missatge, un text que s'encriptarà.

(Per agilitzar el procés he deixat el nom en una constant de configuració ràpida).

```
public class Encriptació {

    public static final String CLAU_PUBLICA = "clauPública.key";
    public static final String TEXTO_ORIGINAL = "Texto original de prueba 1, 2, 3 ñç*^^$%, lee caracteres raros";

    public static final int TAMANYO_AES = 128;

    public static final String CIPHER_PARAMS_RSA = "RSA/ECB/PKCS1Padding";
    public static final String CIPHER_PARAMS_AES = "AES/ECB/PKCS5Padding";

    public static final String ARCHIVO_CLAVE_AES_CIFRADA = "ZZZ_clau_encriptada.txt";
    public static final String ARCHIVO_TEXTO_CIFRADO = "ZZZ_missatge_encriptat.txt";

    public static void main(String[] args) throws Exception {

        Scanner teclado = new Scanner(System.in);

        // Pedir nombre archivo de la clave pública
        System.out.println("Nombre del archivo: ");
        //  String nombreArchivo = teclado.nextLine();
        String nombreArchivo = CLAU_PUBLICA;

        // Cargo la publicKey de archivo
        PublicKey publicKey = leerPublicKey(nombreArchivo);
        System.out.println("Clave cargada correctamente.");
```

Per a poder codificar el missatge, ens fa falta crear una clau AES, que generem fàcilment amb aquest mètode (en aquest cas s'utilitza un tamany de 128):

```
/**
 * Generador claves simétricas AES
 * @param keySize, Tamaño de la key en bits (128, 192, 256)
 * @return sKey, clave simétrica en algoritmo AES
 */

public static SecretKey keygenKeyGeneration(int keySize) {
    SecretKey sKey = null;
    if((keySize == 128) || (keySize == 192) || (keySize == 256)) {
        try {
            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            kgen.init(keySize);
            sKey = kgen.generateKey();
        } catch (NoSuchAlgorithmException ex) {
            System.err.println("Generador no disponible.");
        }
    }
    return sKey;
}
```

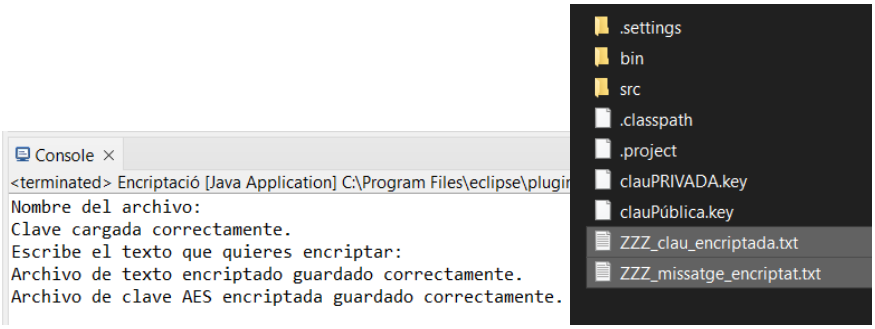
Tenint la clau AES, ja podem codificar el missatge i guardar-lo en un arxiu:

```java
// Se encripta el texto con la keyAES
aes.init(Cipher.ENCRYPT_MODE, keyAES);
byte[] textoEncriptado = aes.doFinal(texto.getBytes("UTF-8"));

// Guardar el archivo del texto encriptado
saveBytesToFile(textoEncriptado, ARCHIVO_TEXTO_CIFRADO);
System.out.println("Archivo de texto encriptado guardado correctamente.");
```

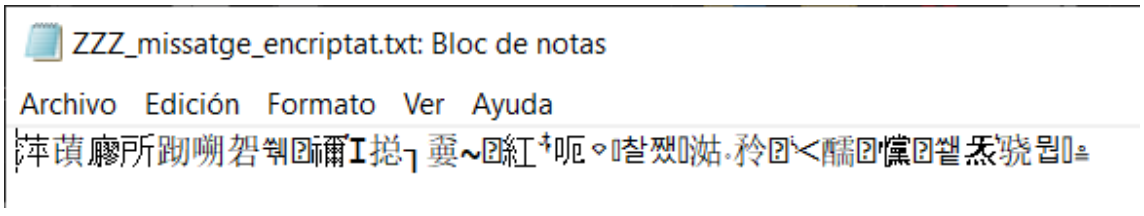
Per a que el nostre receptor pugui desencriptar el missatge, necessitarà que li passem també la clau AES encriptada:

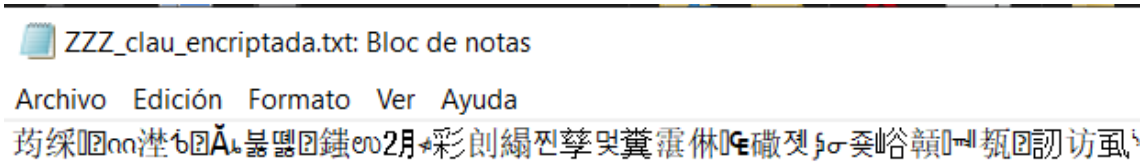
```java
// Encriptar clave simétrica AES
Cipher rsa = Cipher.getInstance("RSA");
rsa.init(Cipher.ENCRYPT_MODE, publicKey);
byte[] claveAESEncriptada = rsa.doFinal(keyAES.getEncoded());

// Guardar el archivo del texto encriptado
saveBytesToFile(claveAESEncriptada, ARCHIVO_CLAVE_AES_CIFRADA);
System.out.println("Archivo de clave AES encriptada guardado correctamente.");
```

A l'executar ens genera un arxiu amb el text codificat i un arxiu amb la clau AES també codificada



Podem afirmar que és totalment il·legible





Part 3: Desencriptació

Primerament llegeix la clau privada que teniem guardada, aquesta clau servirà per poder desencriptar la clau AES, ja que es va encriptar amb la clau pública.

```
/**
 * Carga una clave privada de un archivo
 * @param fileName
 * @return PrivateKey
 * @throws Exception
 */
private static PrivateKey loadPrivateKey(String fileName) throws Exception {

    // Leer archivo
    FileInputStream fis = new FileInputStream(fileName);
    int numBytes = fis.available();
    byte[] bytes = new byte[numBytes];
    fis.read(bytes);
    fis.close();

    // Crear Clave
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    KeySpec keySpec = new PKCS8EncodedKeySpec(bytes);
    PrivateKey clavePrivada = keyFactory.generatePrivate(keySpec);
    return clavePrivada;
}
```

Seguidament, llegim la clau AES encriptada des del fitxer i la desencriptem.

```
/**
 * Desencripta una clave AES guardada en fichero haciendo uso de la PrivateKey
 * @param clavePrivada para desencriptar
 * @param nombreFicheroAES nombre del archivo que contiene la clave AES
 * @return
 */
public static byte[] desencriptarClaveAES(PrivateKey clavePrivada, String nombreFicheroAES) {

    byte[] bytesFitxerAES = null;
    byte[] bytesDecryptedAES = null;

    File fitxerClauAES = new File(nombreFicheroAES);

    try {
        Cipher rsa = Cipher.getInstance(CIPHER_PARAMS_RSA);
        rsa.init(Cipher.DECRYPT_MODE, clavePrivada);
        bytesFitxerAES = Files.readAllBytes(fitxerClauAES.toPath());
        bytesDecryptedAES = rsa.doFinal(bytesFitxerAES);
    } catch (Exception ex) {
        System.err.println("Error cifrando los datos: " + ex);
    }
    return bytesDecryptedAES;
}
```

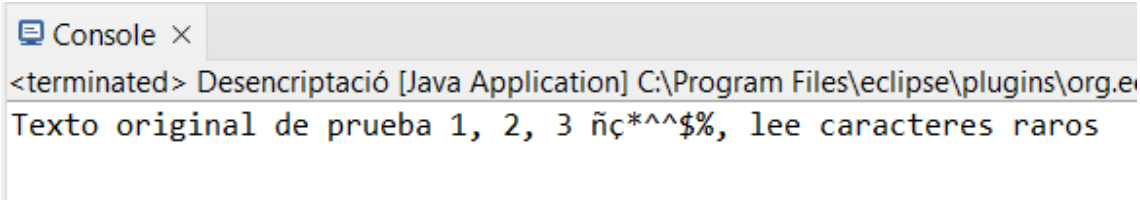
I guardem la clau AES:

```
SecretKey sKeyAES = new SecretKeySpec(claveAESDesencriptada, 0, claveAESDesencriptada.length, "AES");
```

Amb la clau AES ja poder llegir el missatge encriptat, només cal passar per paràmetre la clau AES i el nom de l'arxiu que conté el missatge codificat.

```java
/**
 * Desencripta un mensaje desde un archivo
 * @param sKey
 * @param nombreArchivoMensaje
 * @return
 * @throws IOException
 */
public static String desencriptarMensaje(SecretKey sKey, String nombreArchivoMensaje) throws IOException {

    String mensajeDesencriptado = null;
    File archivoMensaje = new File(nombreArchivoMensaje);

    byte[] bytesMensajeDesencriptado;
    byte[] bytesMensaje = Files.readAllBytes(archivoMensaje.toPath());
    try {
        Cipher aes = Cipher.getInstance(CIPHER_PARAMS_AES);
        aes.init(Cipher.DECRYPT_MODE, sKey);
        bytesMensajeDesencriptado = aes.doFinal(bytesMensaje);
        mensajeDesencriptado = new String(bytesMensajeDesencriptado, StandardCharsets.UTF_8);
    } catch (Exception ex) {
        System.err.println("Error xifrant les dades: " + ex);
    }
    return mensajeDesencriptado;
}
```

A l'executar el programa ens mostra el missatge per pantalla



Que coincideix amb el missatge original que vam crear.

```java
public class Encriptació {

    public static final String CLAU_PUBLICA = "clauPública.key";
    public static final String TEXTO_ORIGINAL = "Texto original de prueba 1, 2, 3 ñç*^^$%, lee caracteres raros";
```