

EXAMEN HASH ENCRIPTACIÓN

**MARIO JIMÉNEZ MARSET**

**ÍNDICE**

[1. ENUNCIADO – OBJETIVOS 3](#_Toc129016024)

[2. DESARROLLO – PROCEDIMIENTOS 3](#_Toc129016025)

# ENUNCIADO – OBJETIVOS

En este examen se pedía desarrollar una clase donde, a partir del fichero de texto dado, se cree otro fichero encriptado mediante el algoritmo de cifrado asimétrico RSA, mostrando las claves públicas y privadas por pantalla. Además, se pedía que, el receptor, a partir de la clave privada y del fichero creado, se desencriptase el contenido y se obtuviese el fichero original.

También se pedía desarrollar otra clase que crease un hash a partir del texto original, mostrándolo por pantalla.

Finalmente, se pedía comparar el hash creado entre este fichero y cualquier otro introducido por pantalla (mensaje comparativo).

# DESARROLLO – PROCEDIMIENTOS

Se muestra el código de las dos clases utilizadas (comentado):

Código ClaseUnoUno:

package examen;

import java.io.\*;

import java.security.\*;

import java.security.spec.InvalidKeySpecException;

import javax.crypto.\*;

public class ClaseUnoUno {

public static void main(String[] args) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, InvalidKeySpecException {

*//se crea el cifrador RSA perteneciente a la clase Cipher*

Cipher cifradorRsa;

*//se generan el par de claves RSA*

KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");

KeyPair keyPair = keyPairGenerator.generateKeyPair();

*//se guardan el par de claves pública y privada en variables*

PublicKey publicKey = keyPair.getPublic();

PrivateKey privateKey = keyPair.getPrivate();

*//se imprimen por pantalla el formato de las dos claves*

System.out.println("La clave pública es "+publicKey.getFormat());

System.out.println("La clave privada es "+privateKey.getFormat());

*//se establece el tipo de cifrado (por parámetro se establece el cifrado RSA)*

cifradorRsa = Cipher.getInstance("RSA/ECB/PKCS1Padding");

*//se escribe la ruta del fichero cuyo contenido va a ser codificado*

String ficheroTexto="C:\\Users\\jimen\\Downloads\\mensaje texto.txt";

String contenido=null;

try {

*//se crea el objeto de tipo File que va a recoger por parámetro la ruta insertada*

File f=new File(ficheroTexto);

*//se crea el FileReader y BufferedReader encargados de leer el contenido del fichero*

FileReader fr=new FileReader(f);

BufferedReader br=new BufferedReader(fr);

*//dentro del bucle que lee el contenido, se hará el cifrado y descifrado*

while((contenido=br.readLine())!=null) {

*//cada línea se almacena en otro String diferente*

String texto=contenido;

cifradorRsa.init(Cipher.ENCRYPT\_MODE, publicKey);

*//se almacena en un array de bytes la información del fichero*

byte[] encriptado = cifradorRsa.doFinal(texto.getBytes());

System.out.print("Encriptado-> ");

*//por cada byte guardado, se realiza una conversión hexadecimal*

for (byte b : encriptado) {

System.out.print(Integer.toHexString(0xFF & b));

}

*//se imprime el texto encriptado (la colección de bytes)*

System.out.println("\nEl texto encriptado es: "+encriptado.toString());

cifradorRsa.init(Cipher.DECRYPT\_MODE, privateKey);

*//en otro array de bytes, se almacena la información desencriptada*

byte[] bytesDesencriptados = cifradorRsa.doFinal(encriptado);

*//se muestra el array de bytes, convirtiéndose a String*

String textoDesencripado = new String(bytesDesencriptados);

*//se imprime por pantalla cada línea desencriptada del fichero*

System.out.println("El texto desencriptado es: "+textoDesencripado);

}

br.close();

}catch (FileNotFoundException e1) {

System.out.println(e1.getMessage());

}catch (IOException e2) {

System.out.println(e2.getMessage());

}

}

}

Código ClaseDosUno:

package examen;

import java.security.\*;

import java.util.Scanner;

import org.apache.commons.codec.binary.Hex;

public class ClaseDosUno {

public static void main(String[] args) {

*//se crean dos métodos: el primero, muestra el hash del fichero original*

*//el segundo gestiona el hash del fichero insertado y el mensaje final*

System.out.print("ALGORITMO HASH SHA-512-> ");

hash();

hashNuevo(hash());

}

public static String hash() {

*//a partir de la clase MessageDigest, se gestiona la creación de hash*

MessageDigest md = null;

String fichero="C:\\Users\\jimen\\Downloads\\mensaje texto.txt";

char[] hash=null;

try {

*//se llama al método getInstance, con el algoritmo elegido*

md=MessageDigest.getInstance("SHA-512");

*//se llama al método actualizar, consiguiendo los bytes del fichero*

md.update(fichero.getBytes());

byte[] mb=md.digest(fichero.getBytes());

hash=Hex.encodeHex(mb);

*//se imprime el hash finalmente creado*

System.out.println(Hex.encodeHex(mb));

}catch(NoSuchAlgorithmException e) {

System.out.println(e.getMessage());

}

return String.valueOf(hash);

}

public static void hashNuevo(String hash) {

*//se guarda en una variable el hash creado por el anterior método*

String hashTextoDado=hash;

System.out.print("El hash original es-> ");

System.out.println(hashTextoDado);

*//se siguen los mismos pasos que en el anterior método para crear el hash del fichero introducido*

char[] hashNuevo=null;

Scanner entrada=new Scanner(System.in);

System.out.println("Introduce el fichero, para saber si el hash es el mismo o no");

MessageDigest md = null;

String fichero=entrada.nextLine();

try {

md=MessageDigest.getInstance("SHA-512");

md.update(fichero.getBytes());

byte[] mb=md.digest(fichero.getBytes());

hashNuevo=Hex.encodeHex(mb);

System.out.print("El hash nuevo es-> ");

System.out.println(Hex.encodeHex(mb));

*//lo único nuevo es que se comparan las cadenas (el hash original y el nuevo)*

*//saldrá un mensaje u otro dependiendo de si es el mismo hash*

if(hashTextoDado.equals(String.valueOf(hashNuevo))) {

System.out.println("Es el mismo fichero. Coinciden los hashs");

}else {

System.out.println("Son ficheros diferentes");

}

}catch(NoSuchAlgorithmException e) {

System.out.println(e.getMessage());

}

}

}

CAPTURAS RESULTADOS:



