**Programa Cifrado Asimétrico**

Escribir un programa que utilice cifrado asimétrico para cifrar el contenido de un fichero, usando la clave privada. el resultado del cifrado y la clave pública se deben volcar en dos ficheros adicionales de salida. Posteriormente, escribe un segundo programa que cargue dichos ficheros y descifre el archivo cifrado. Utilizar cifrado rsa.

**Solución**

La solución es similar a la del ejercicio 1, pero introduciendo las modificaciones oportunas para operar con cifrado asimétrico RSA. En este caso, el programa que cifra lee el fichero de entrada en bloques de 64 bytes.

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.PrintWriter;

**import** java.security.KeyFactory;

**import** java.security.KeyPair;

**import** java.security.KeyPairGenerator;

**import** java.security.spec.RSAPublicKeySpec;

**import** javax.crypto.Cipher;

**public** **class** cifradoasimetrico {

**public** **static** **void** main(String[] args) {

**try** {

KeyPairGenerator keygen = KeyPairGenerator.*getInstance*("RSA");

System.***out***.println("Generando par de claves");

KeyPair keypair = keygen.generateKeyPair();

System.***out***.println("Obteniendo objeto Cipher con cifrado RSA");

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

System.***out***.println("Configurando Cipher para encriptar");

desCipher.init(Cipher.***ENCRYPT\_MODE***, keypair.getPrivate());

System.***out***.println("Abriendo el fichero");

File inf = **new** File("fichero\_prueba.txt");

FileInputStream is = **new** FileInputStream(inf);

System.***out***.println("Abriendo el fichero cifrado");

FileOutputStream os = **new** FileOutputStream("fichero\_cifrado\_con\_RSA");

System.***out***.println("Cifrando el fichero...");

**byte**[] buffer = **new** **byte**[1024];

**int** bytes\_leidos = is.read(buffer);

**while** (bytes\_leidos != -1) {

**byte**[] cbuf = desCipher.doFinal(buffer, 0, bytes\_leidos);

os.write(cbuf);

bytes\_leidos = is.read(buffer);

}

os.close();

System.***out***.println("Obteniendo factoría de claves con cifrado RSA");

KeyFactory keyfac = KeyFactory.*getInstance*("RSA");

System.***out***.println("Generando keyspec");

RSAPublicKeySpec publicKeySpec = keyfac.getKeySpec(keypair.getPublic(), RSAPublicKeySpec.**class**);

System.***out***.println("Salvando la clave en un fichero");

FileOutputStream cos = **new** FileOutputStream("clave\_publica");

PrintWriter pw = **new** PrintWriter(cos);

pw.println(publicKeySpec.getModulus());

pw.println(publicKeySpec.getPublicExponent());

pw.close();

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

}

}

**Código del programa de prueba:**

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.FileReader;

**import** java.math.BigInteger;

**import** java.security.Key;

**import** java.security.KeyFactory;

**import** java.security.spec.RSAPublicKeySpec;

**import** javax.crypto.Cipher;

**public** **class** cifradoasimetrico\_prueba {

**public** **static** **void** main(String[] args) {

**try** {

BufferedReader br = **new** BufferedReader(**new** FileReader("clave\_publica"));

BigInteger modulus = **new** BigInteger(br.readLine());

BigInteger exponente = **new** BigInteger(br.readLine());

RSAPublicKeySpec keyspec = **new** RSAPublicKeySpec(modulus, exponente); KeyFactory keyfac = KeyFactory.*getInstance*("RSA");

Key public\_key = keyfac.generatePublic(keyspec);

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

desCipher.init(Cipher.***DECRYPT\_MODE***, public\_key);

File inf = **new** File("fichero\_cifrado\_con\_RSA");

FileInputStream is = **new** FileInputStream(inf);

FileOutputStream os = **new** FileOutputStream("fichero\_descifrado\_RSA.txt");

**byte**[] buffer = **new** **byte**[1024];

**int** bytes\_leidos = is.read(buffer);

**while** (bytes\_leidos != -1) {

os.write(desCipher.doFinal(buffer, 0, bytes\_leidos));

bytes\_leidos = is.read(buffer);

}

os.close();

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

} }