Teoría de la Información y Codificación

Práctica 8: Algoritmo de Clave Pública: RSA

José A. Montenegro Montes

26 de septiembre de 2014

1. Enunciado

Esta práctica tiene como objetivo mostrar el funcionamiento del algoritmo de clave pública RSA. Para ello realizaremos las funciones de creación de clave, cifrado y descifrado. Es conveniente que el alumno compare los métodos de cifrado y descifrado en la criptografía de clave simétrica y asimétrica.

2. Conclusiones

La práctica pretende que el alumno determine las diferencias entre la criptografía simétrica y asimétrica o de clave pública.

3. Código

Clase Main.

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package rsa;
7
8  import java.math.BigInteger;
9
10 /**
11  *
12  * @author monte
13  */
```

```java
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

       int tamPrimo = 1024;
        RSA rsaAlice = new RSA(tamPrimo);


        System.out.println("Tam Clave: ["+ tamPrimo + "]\n");

        System.out.println("p: [" + rsaAlice.get_p().toString(16).toUpperCase() + "]");
        System.out.println("q: [" + rsaAlice.get_q().toString(16).toUpperCase() + "]\n");

        System.out.println("Clave publica (n,e)");
        System.out.println("n: [" + rsaAlice.get_n().toString(16).toUpperCase() + "]");
        System.out.println("e: [" + rsaAlice.get_e().toString(16).toUpperCase() + "]\n");

        System.out.println("Clave publica (n,d)");
        System.out.println("n: [" + rsaAlice.get_n().toString(16).toUpperCase() + "]");
        System.out.println("d: [" + rsaAlice.get_d().toString(16).toUpperCase() + "]\n");


        String textoPlano = ("Que clase mas buena!");
        System.out.println("Texto a cifrar: ["+ textoPlano +"]\n");

        BigInteger[] textoCifrado = rsaAlice.cifrar(textoPlano);

        System.out.println("Texto cifrado: [");

        for(int i=0; i<textoCifrado.length; i++) {
            System.out.print(textoCifrado[i].toString(16).toUpperCase());
            if(i != textoCifrado.length-1)
                System.out.println("");
        }
        System.out.println("]\n");

        String recuperarTextoPlano = rsaAlice.descifrar(textoCifrado);
        System.out.println("Texto decifrado: ["+ recuperarTextoPlano +"]");

    }

}
```

Clase RSA.

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package rsa;

/**
 *
 * @author monte
 */

import java.math.BigInteger;
import java.util.*;


public class RSA {
 int tamPrimo;
    BigInteger n, q, p;
    BigInteger totient;
    BigInteger e, d;

    /** Constructor de la clase RSA */
    public RSA(int tamPrimo) {
        this.tamPrimo = tamPrimo;
        generaPrimos();             //Genera p y q
        generaClaves();             //Genera e y d
    }

    /*************
     *
     */
    public void generaPrimos()
    {
        p = new BigInteger(tamPrimo, 10, new Random());
        do q = new BigInteger(tamPrimo, 10, new Random());
            while(q.compareTo(p)==0);
    }
    /************************
     *
     */

    public void generaClaves()
```

```
44      {
45          // n = p * q
46          n = p.multiply(q);
47          // toltient = (p-1)*(q-1)
48          totient = p.subtract(BigInteger.valueOf(1));
49          totient = totient.multiply(q.subtract(BigInteger.valueOf(1)));
50
51          // Elegimos un e coprimo de y menor que n
52          do e = new BigInteger(2 * tamPrimo, new Random());
53              while((e.compareTo(totient) != -1) ||
54                    (e.gcd(totient).compareTo(BigInteger.valueOf(1)) != 0));
55          // d = e^1 mod totient
56          d = e.modInverse(totient);
57      }
58
59      /**
60       * Cifra el texto usando la clave pública
61       *
62       * @param   mensaje    Mensaje a cifrar
63       * @return   El mensaje Cifrado
64       */
65      public BigInteger[] cifrar(String mensaje)
66      {
67          int i;
68          byte[] temp = new byte[1];
69          byte[] digitos = mensaje.getBytes();
70          BigInteger[] bigdigitos = new BigInteger[digitos.length];
71
72          for(i=0; i<bigdigitos.length;i++){
73              temp[0] = digitos[i];
74              bigdigitos[i] = new BigInteger(temp);
75          }
76
77          BigInteger[] cifrado = new BigInteger[bigdigitos.length];
78
79          for(i=0; i<bigdigitos.length; i++)
80              cifrado[i] = bigdigitos[i].modPow(e,n);
81
82          return(cifrado);
83      }
84
85      /**
86       * Descifra el texto cifrado usando la clave privada
87       *
88       * @param   Array objetos
89       * @return  El texto en claro
```

```
90       */
91      public String descifrar(BigInteger[] cifrado) {
92          BigInteger[] descifrado = new BigInteger[cifrado.length];
93
94          for(int i=0; i<descifrado.length; i++)
95              descifrado[i] = cifrado[i].modPow(d,n);
96
97          char[] charArray = new char[descifrado.length];
98
99          for(int i=0; i<charArray.length; i++)
100             charArray[i] = (char) (descifrado[i].intValue());
101
102         return(new String(charArray));
103     }
104
105     public BigInteger get_p() {return(p);}
106
107     public BigInteger get_q() {return(q);}
108
109     public BigInteger get_totient() {return(totient);}
110
111     public BigInteger get_n() {return(n);}
112
113     public BigInteger get_e() {return(e);}
114
115     public BigInteger get_d() {return(d);}
116 }
```