Seed Code for Project 1 – Secure Client-Server Communication using RSA Algorithm

Instructor: Dr. Natarajan Meghanathan, CSC 439/539 Advanced Information Security, Spring 2013

(1) RSA Key generation

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

class RSAKeyGen{

  public static void main(String[] args){

    /* Creating two random number generators, each with a different seed */
    Random rand1 = new Random(System.currentTimeMillis());
    Random rand2 = new Random(System.currentTimeMillis()*10);

    int pubKey = Integer.parseInt(args[0]); /* public key is at least a
                                        certain value input by the user */

/* returns a BigInteger that is not prime with probability less than 2^(-100)
*/
    BigInteger bigB_p = BigInteger.probablePrime(32, rand1);
    BigInteger bigB_q = BigInteger.probablePrime(32, rand2);

    BigInteger bigB_n = bigB_p.multiply(bigB_q);

    BigInteger bigB_p_1 = bigB_p.subtract(new BigInteger("1")); //p-1
    BigInteger bigB_q_1 = bigB_q.subtract(new BigInteger("1")); //q-1
    BigInteger bigB_p_1_q_1 = bigB_p_1.multiply(bigB_q_1); // (p-1)*(q-1)

    // generating the correct public key

    while (true){
        BigInteger BigB_GCD = bigB_p_1_q_1.gcd(new BigInteger(""+pubKey));

        if (BigB_GCD.equals(BigInteger.ONE)){
             break;
        }

        pubKey++;
    }

    BigInteger bigB_pubKey = new BigInteger(""+pubKey);
    BigInteger bigB_prvKey = bigB_pubKey.modInverse(bigB_p_1_q_1);

    System.out.println(" public key : "+bigB_pubKey+" , "+bigB_n);
    System.out.println(" private key: "+bigB_prvKey+" , "+bigB_n);
   }
}
```

Compilation and Execution

```
C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>javac RSAKeyGen.java

C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>java RSAKeyGen 34
 public key : 37 , 134295695861281171O9
 private key: 9799956179097816673 , 134295695861281171O9
```

(2) A program that inputs a public key-private key pair, the ascii value of the character to encrypt and prints the cipher text as well as the decrypted plaintext

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

class RSAEncDec{

  public static void main(String[] args){


    BigInteger bigB_pubKey = new BigInteger(args[0]);
    BigInteger bigB_prvKey = new BigInteger(args[1]);
    BigInteger bigB_n = new BigInteger(args[2]);

    int asciiVal = Integer.parseInt(args[3]);


    /* encrypting an ASCII integer value using the public key and decrypting
the cipher value using the private key and extracting the ASCII value back */

    BigInteger bigB_val = new BigInteger(""+asciiVal);
    BigInteger bigB_cipherVal =  bigB_val.modPow(bigB_pubKey, bigB_n);

    System.out.println("ciphertext: "+bigB_cipherVal);

    BigInteger bigB_plainVal = bigB_cipherVal.modPow(bigB_prvKey, bigB_n);
    int plainVal = bigB_plainVal.intValue();

    System.out.println("plaintext:  "+plainVal);

   }
}
```

Compilation and Execution

```
C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>javac RSAEncDec.java

C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>java RSAEncDec 37 9799956179097
816673 13429569586128117109 65
ciphertext: 106074961001006150 26
plaintext:  65

C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>
```

(3) Socket programs to send a sequence of 10 Big Integers from the client to the server; the server receives and prints them

Server program (run/start the server first, followed by the client)

```
import java.math.BigInteger;
import java.net.*;
import java.io.*;

class BigIntServer{

   public static void main(String[] args){

   try{

       ServerSocket serv = new ServerSocket(Integer.parseInt(args[0]));
       Socket socket = serv.accept();
       ObjectInputStream ois = new
ObjectInputStream(socket.getInputStream());

       Object object = null;

       BigInteger big_neg_1 = new BigInteger("-1");

       while ( true){

         BigInteger bigInt = (BigInteger) ois.readObject();
         if (bigInt.equals(big_neg_1))
            break;
         System.out.println("big integer: "+bigInt);

       }

       socket.close();
       serv.close();



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

   }

}
```

Client program

```
import java.math.BigInteger;
import java.net.*;
import java.io.*;
import java.util.*;

class BigIntClient{

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

```
   try{

      Random rand = new Random(System.currentTimeMillis());

      InetAddress serverHost = InetAddress.getByName(args[0]);
      int serverPortNum = Integer.parseInt(args[1]);

      Socket clientSocket = new Socket(serverHost, serverPortNum);
      ObjectOutputStream oos = new
ObjectOutputStream(clientSocket.getOutputStream());

      for (int i=0; i<10; i++){
       BigInteger bigInt = BigInteger.probablePrime(32, rand);

       oos.writeObject(bigInt);
       oos.flush();
      }

       BigInteger bigInt = new BigInteger("-1");
       oos.writeObject(bigInt);
       oos.flush();

      clientSocket.close();


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

   }

}
```

Compilation and Sample Screenshots for Execution

Server Side

```
C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>javac BigIntServer.java

C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>java BigIntServer 1234
big integer: 4292419583
big integer: 2959199183
big integer: 2531628403
big integer: 2207063779
big integer: 3725635987
big integer: 3193217243
big integer: 2978664373
big integer: 2737799749
big integer: 2776626871
big integer: 2811379297

C:\Spring-2013\CSC439-AIS\Projects\Project-1-RSA>
```

Client Side

```
C:\Spring-2013\CSC439-AIS>javac BigIntClient.java
C:\Spring-2013\CSC439-AIS>java BigIntClient localhost 1234
C:\Spring-2013\CSC439-AIS>_
```