Week 8 - Socket with Diffie-Hellman and Elgamal

Code:

elgamal.java

```
package cryptfiles;

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

import javax.crypto.SecretKey;

import java.security.*;
import java.io.*;

public class elgamal {

    final static Random sc = new SecureRandom();;
    static BigInteger secretKey;

    //setting the key that is obtained from the diffie algorithm
    public static void setSecretKey(String key) {
        secretKey = new BigInteger(key);
    }

    //the pre processing
    public static BigInteger[] calc() {
        BigInteger p, b, c;
        p = BigInteger.probablePrime(64, sc);
        b = new BigInteger("3");
        c = b.modPow(secretKey, p);

        BigInteger[] arr = { p, b, c, secretKey };
        return arr;
    }

    //performing the algorithm on the input from the user
    public String[] enc(String s) throws IOException {
```

```
        BigInteger[] calc = calc();

        BigInteger p = calc[0];
        BigInteger b = calc[1];
        BigInteger c = calc[2];
        BigInteger secretKey = calc[3];

        BigInteger X = new BigInteger(s);
        BigInteger r = new BigInteger(64, sc);
        BigInteger EC = X.multiply(c.modPow(r, p)).mod(p);
        BigInteger brmodp = b.modPow(r, p);

        String arr[] = { brmodp.toString(), secretKey.toString(), p.toString(), EC.toString() };

        return arr;
    }

    //decoding the message got
    public String dec(String brmodp_str, String secretKey_str, String p_str, String EC_str) {

        BigInteger brmodp = new BigInteger(brmodp_str);
        BigInteger secretKey = new BigInteger(secretKey_str);
        BigInteger p = new BigInteger(p_str);
        BigInteger EC = new BigInteger(EC_str);

        BigInteger crmodp = brmodp.modPow(secretKey, p);
        BigInteger d = crmodp.modInverse(p);
        BigInteger ad = d.multiply(EC).mod(p);

        return (ad.toString());
    }
}
```

diffie.java

```
package cryptfiles;

import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.SecureRandom;
```

```
import javax.crypto.spec.DHParameterSpec;
import javax.crypto.spec.DHPublicKeySpec;

public class diffie {
   public final static int pValue = 47;

   public final static int gValue = 71;

   public final static int XaValue = 9;

   public final static int XbValue = 14;

   public static String getKey() throws Exception {
      BigInteger p = new BigInteger(Integer.toString(pValue));
      BigInteger g = new BigInteger(Integer.toString(gValue));
      BigInteger Xa = new BigInteger(Integer.toString(XaValue));
      BigInteger Xb = new BigInteger(Integer.toString(XbValue));

      int bitLength = 512; // 512 bits
      SecureRandom rnd = new SecureRandom();
      p = BigInteger.probablePrime(bitLength, rnd);
      g = BigInteger.probablePrime(bitLength, rnd);

      return (createSpecificKey(p, g));
   }

   public static String createSpecificKey(BigInteger p, BigInteger g) throws Exception {
      KeyPairGenerator kpg = KeyPairGenerator.getInstance("DiffieHellman");

      DHParameterSpec param = new DHParameterSpec(p, g);
      kpg.initialize(param);
      KeyPair kp = kpg.generateKeyPair();

      KeyFactory kfactory = KeyFactory.getInstance("DiffieHellman");

      DHPublicKeySpec kspec = (DHPublicKeySpec) kfactory.getKeySpec(kp.getPublic(),
DHPublicKeySpec.class);

      System.out.println("Key Generated");

      return kspec.getY().toString();
   }
}
```

Client.java

```
import java.io.*;
import java.net.*;
import java.security.KeyStore.SecretKeyEntry;

import java.math.*;
import cryptfiles.*;

public class Client {

   // gets the message from the user
   public static String getValue() throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      System.out.print("Enter the message ->");
      String s = br.readLine();


      String s_int = "";
      for(int i = 0; i< s.length(); i++){
         s_int += (int)(s.charAt(i));
      }

      System.out.println("input in integers:" + s_int);
      return (s_int);
   }

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

         elgamal elgamal = new elgamal();
         diffie diffie = new diffie();

         // generates the diffie key
         String secretKey = diffie.getKey();
         elgamal.setSecretKey(secretKey);
         // encodes with the key got
         String arr[] = elgamal.enc(getValue());

         Socket sock = new Socket("localhost", 6666);
         DataOutputStream out = new DataOutputStream(sock.getOutputStream());

         out.writeUTF(secretKey);
```

```
            for (String str : arr) {
                out.writeUTF(str);
            }

            out.writeUTF("\n");

            out.flush();
            out.close();

            sock.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Server.java

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


import cryptfiles.*;

public class Server {

    public static void processData(String arr[]) {
        elgamal obj = new elgamal();
        obj.setSecretKey(arr[0]);
        String res = obj.dec(arr[1], arr[2], arr[3], arr[4]);

        System.out.println("Received message is: " + res);

    }

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

        try {
            ServerSocket servSock = new ServerSocket(6666);
            Socket sock = servSock.accept();// establishes connection

            DataInputStream in = new DataInputStream(sock.getInputStream());
```

```
            int receivedObjects = 5;
            String arr[] = new String[receivedObjects];

            for (int i = 0; i < receivedObjects; i++) {
                arr[i] = (String) in.readUTF();
            }

            processData(arr);

            servSock.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Output: