CT255 Next Generation technologies

Cyber Security

Assignment 4

19/11/2022

Maxwell Maia 21236277

Diffie-Hellman Key exchange

Problem 1 & 2

Class: sender

```
/**
 * Write a description of class sender here.
 *
 * @author Maxwell Maia
 *
 */
public class sender
{
    //Personal.
    private long XA; //Calculated here. Private.
    private long YA; //Calculated here. Will be sent over public domain.

    //From other user.
    private long YB; //Calculated in receiver. Set when the receiver replies with this value.
```

```
//DB parameters agreed. Available in public domain. Make them available here too.
private long a;
private long p;

//The ultimate goal. We want this to be a shared secret number.
private long K;

/**
 * Constructor for objects of class sender
 */
public sender(long a, long p)
{
   XA = generatePrivateKey();
   System.out.println("XA privatekeySender = "+XA);

   this.a = a;
   this.p = p;
}

public void calculateK()
{
   //[YB^XA mod p]
   K = power(YB, XA, p);
   System.out.println("In sender:\nK = " + K);
}

public void setYB(long YB)
{
   this.YB = YB;
```

```
}

public void calculateYA()
{

   YA = power(a, XA, p); //[a^XA mod p]
   System.out.println("YA = "+YA);

}

public long getYA()
{
   return YA;
}



public int randomInt(int min, int max)
{
   return (int)Math.floor(Math.random()*(max-min+1)+min);
}



public int generatePrivateKey()
{
   int privateKey = randomInt(0, 9999);
   return privateKey;
}

public long power(long a, long X, long p)
```

```
    {
        long result = 1;

        //In case a is more than or equal to p.
        a = a % p;

        while (X > 0)
        {
            //Multiply a with the result mod p, if X is odd.
            if (X % 2 == 1)
            {
                result = (result * a) % p;
            }

            //Therefore, X is now even
            X = X >> 1; // y = y/2
            a = (a * a) % p;
        }
        return result;
    }
}
```

Class: receiver

```
/**
 * Write a description of class receiver here.
 *
 * @author Maxwell Maia
 */
public class receiver
{
   //Personal.
   private long XB; //Private.
   private long YB; //Calculated here. Will be sent over public domain.

   //From other user.
   private long YA; //Calculated in sender. Set when given this value at the first communication with the sender.

   //DB parameters agreed. Available in public domain. Make them available here too.
   private long a;
   private long p;

   //The ultimate goal. We want this to be a shared secret number.
   private long K;

   /**
    * Constructor for objects of class receiver
    */
   public receiver(long a, long p, long YA)
   {
```

```
    XB = generatePrivateKey();
    System.out.println("XB privatekeyReceiver = "+XB);

    this.a = a;

    this.p = p;

    this.YA = YA;
}

public long getYB()
{
    return YB;
}

public void calculateK()
{
    //[YA^XB mod p]
    K = power(YA, XB, p);
    System.out.println("In receiver:\nK = " + K);
}

public void calculateYB()
{

    YB = power(a, XB, p); //[a^XB mod p]
    System.out.println("YB = "+YB);

}
```

```
public int randomInt(int min, int max)
{
   return (int)Math.floor(Math.random()*(max-min+1)+min);
}


public int generatePrivateKey()
{
   int privateKey = randomInt(0, 9999);
   return privateKey;
}

public long power(long a, long X, long p)
{
   long result = 1;

   //In case a is more than or equal to p.
   a = a % p;

   while (X > 0)
   {
      //Multiply a with the result mod p, if X is odd.
      if (X % 2 == 1)
      {
         result = (result * a) % p;
      }

      //Therefore, X is now even
```

```
            X = X >> 1; // y = y/2
            a = (a * a) % p;
        }
        return result;
    }
}
```

The main class

Class: ke

```
import java.util.ArrayList;
import java.util.Random;

/**
 * CT255 - Assignment 4
 * Diffie-Hellman.
 *
 * @author Maxwell Maia
 * @version 1.0
 */

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

```
public class ke
{
   /**
    * Constructor for objects of class Stegano1
    */
   public ke()
   {
   }

   public static void main(String[] args)
   {
      //Welcome to diffie_hellman

      System.out.println("\n=======================\n\nWelcome to diffie_hellman");
      diffie_hellman();
   }

   static void diffie_hellman()
   {
      //PROBLEM 1

      System.out.println("\n\nGenerating DH parameters.");




      //Generate p
      // prime number in range 10^4 < p < 10^5
      System.out.println("\nGenerating a random prime number between 10 000 and 100
000.");
```

```
long p = generateP();

//Test code for p
System.out.println("Checking that "+p+" is a valid value for p");
//Check that range is correct.
boolean validRange = false;
if(10000 < p && p < 100000)
{
   System.out.println("The range is correct. 10^4 < p < 10^5");
   validRange = true;
}
else
{
   System.out.println("The range is NOT correct. 10^4 < p < 10^5");
   validRange = false;
}

//Prime number test.
if(isPrime(p))
{
   System.out.println("The number is prime.");
}
else
{
   System.out.println("The number is NOT prime.");
}

if(isPrime(p) && validRange)
{
```

```
    System.out.println(p+ " is a valid number for p.");
  }
  else
  {
    System.out.println("Therefore, "+p+ " is NOT a valid number for p.");

    //Cannot proceed
    System.out.println("Cannot proceed. Returning");
    return;
  }

  //Generate a
  // the primitive root of p
  System.out.println("\nFind the smallest primitive root of the prime number.");
  long a = 0;
  a = findPrimativeRoot(p);
  //TEST A IS PRIM ROOT (maybe needed)

  System.out.println("\nNote: the prime number (p) and the primitive root (a) are known values.\nThey are in the public domain.\n");

  System.out.println("p = "+p);
  System.out.println("a = "+a);

  //ESTABLISH A SHARED KEY BETWEEN 2 COMPUTERS
  System.out.println("\n\nESTABLISH A SHARED KEY BETWEEN 2 COMPUTERS");
```

```
        //USER 1
        System.out.println("\nUser 1. The sender. Alice.");
        System.out.println("Alice generates a private key (XA) only she can see it.");

        //Generates XA and is private. Only Alice can see it. (inside constructor of sender);
        sender alice = new sender(a, p);
        System.out.println("Sender has generated a private key that only the sender can see.\n");


        System.out.println("The sender then calculates YA, which is \"a\" to the power of XA, modulus p (a^XA mod p).");
        alice.calculateYA();

        System.out.println("\nAlice sends the YA to User 2. Bob.");
        System.out.println("Note: YA can be intercepted in the public domain.");
        System.out.println("This is okay because XA is still private.");
        System.out.println("XA cannot be easily determined even if you know p, a and YA because of the modulus.");


        //USER 2
        System.out.println("\nUser 2. The receiver. Bob.");
        System.out.println("Bob generates a private key (XB) only he can see it.");
        receiver bob = new receiver(a, p, alice.getYA());

        //Generates XA and is private. Only Alice can see it. (inside constructor of sender);
        System.out.println("Receiver has generated a private key that only the receiver can see.\n");
```

```
System.out.println("The receiver then calculates YB, which is \"a\" to the power of XB, modulus p (a^XB mod p).");

bob.calculateYB();


System.out.println("Bob sends YB to User 1. Alice.");

alice.setYB(bob.getYB());


System.out.println("\nEach user calculates their own K. [K = OtherUserY^myX mod p].");


System.out.println("\n=========================");

System.out.println("Alice/Bob\n");


alice.calculateK();

bob.calculateK();


System.out.println("\n=========================");









//PROBLEM 2




System.out.println("\n\n----------------------------------------------------\n");
```

```
System.out.println("PERFORM A MOCK MAN-IN-THE-MIDDLE ATTACK (MitM).");


System.out.println("\n\nOur nefarious character is Mallory.\n");


//Bob/Mallory and Alice/Mallory.



//Alice/Mallory

System.out.println("-------");

System.out.println("Alice attempts to establish a connection with Bob.");

System.out.println("Mallory intercepts this request, blocks it from reaching bob and establishs a shared key between Alice and Mallory.");

System.out.println("As far as Alice knows, she is talking to Bob. But she is actually talking to Mallory.");

System.out.println("\nESTABLISH A SHARED KEY BETWEEN 2 COMPUTERS");


//-----------------

//USER 1. Alice

System.out.println("\nUser 1. The sender. Alice.");

System.out.println("Alice generates a private key (XA) only she can see it.");


//Generates XA and is private. Only Alice can see it. (inside constructor of sender);

sender alice2 = new sender(a, p);

//System.out.println("Sender has generated a private key that only the sender can see.\n");



System.out.println("The sender then calculates YA, which is \"a\" to the power of XA, modulus p (a^XA mod p).");

alice2.calculateYA();
```

```
System.out.println("\nAlice sends the YA to User 2. Mallory.");



//USER 2. Mallory

System.out.println("\nUser 2. The receiver. Mallory.");

System.out.println("Mallory generates a private key (XB) only she can see it.");

receiver mallory = new receiver(a, p, alice2.getYA());

//Generates XA and is private. Only Alice can see it. (inside constructor of sender);

System.out.println("Receiver has generated a private key that only the receiver can see.\n");

System.out.println("The receiver then calculates YB, which is \"a\" to the power of XB, modulus p (a^XB mod p).");

mallory.calculateYB();

System.out.println("Mallory sends YB to User 1. Alice.");

alice2.setYB(mallory.getYB());

System.out.println("\nEach user calculates their own K. [K = OtherUserY^myX mod p].");

System.out.println("\n=========================");

System.out.println("Alice/Mallory\n");

alice2.calculateK();

mallory.calculateK();

System.out.println("\n=========================");
```

```
        //Mallory/Bob
        System.out.println("\n-------");
        System.out.println("Mallory establishs a connection with Bob.");
        System.out.println("\nESTABLISH A SHARED KEY BETWEEN 2 COMPUTERS");

        //-----------------
        //USER 1. Mallory
        System.out.println("\nUser 1. The sender. Mallory.");
        System.out.println("Mallory generates a private key (XA) only she can see it.");

        //Generates XA and is private. Only Alice can see it. (inside constructor of sender);
        sender mallory2 = new sender(a, p);
        //System.out.println("Sender has generated a private key that only the sender can see.\n");

        System.out.println("The sender then calculates YA, which is \"a\" to the power of XA, modulus p (a^XA mod p).");
        mallory2.calculateYA();

        System.out.println("\nMallory sends the YA to User 2. Bob.");

        //USER 2. Bob
        System.out.println("\nUser 2. The receiver. Bob.");
        System.out.println("Bob generates a private key (XB) only he can see it.");
        receiver bob2 = new receiver(a, p, mallory2.getYA());
```

```
        //Generates XA and is private. Only Alice can see it. (inside constructor of sender);
        System.out.println("Receiver has generated a private key that only the receiver can see.\n");

        System.out.println("The receiver then calculates YB, which is \"a\" to the power of XB, modulus p (a^XB mod p).");
        bob2.calculateYB();

        System.out.println("Bob sends YB to User 1. Mallory.");
        mallory2.setYB(bob2.getYB());

        System.out.println("\nEach user calculates their own K. [K = OtherUserY^myX mod p].");

        System.out.println("\n=========================");
        System.out.println("Mallory/Bob\n");

        mallory2.calculateK();
        bob2.calculateK();

        System.out.println("\n=========================");

    }

    //Generate prime number in range 10^4 < p < 10^5.
    static long generateP()
    {
        int min = 10001;
        int max = 99999;
```

```
    int random_int = 0;
    boolean done = false;

    while(!done)
    {
        System.out.println("Random value in int from "+min+" to "+max+ ":");
        random_int = randomInt(min, max);
        System.out.println(random_int);
        if(isPrime(random_int))
        {
            System.out.println("Aha! This random integer is prime!");
            done = true;
        }
    }

    return random_int;
}

static int randomInt(int min, int max)
{
    return (int)Math.floor(Math.random()*(max-min+1)+min);
}


//method to check if a number is prime.
static boolean isPrime(long inputNumber)
{
    boolean prime = true;
```

```
    if(inputNumber <= 1)
    {
       prime = false;
    return prime;
    }
    else
    {
       for (int i = 2; i<= inputNumber/2; i++)
       {
          if ((inputNumber % i) == 0)
          {
             prime = false;
             break;
          }
       }

       return prime;
    }
}


private static long findPrimativeRoot(long prime) {
    long primitiveRoot = 1;
    long result = 1;
    boolean flag = false;


    for(int j = 2; j < prime; j++)
    {
```

```
ArrayList<Long> primes = new ArrayList<>();
flag = false;
primitiveRoot = j;

for(int i = 0; i < prime - 1; i++)
{
   flag = false;

   result = power(j, i, prime);

   for (int k = 0; k < primes.size() - 1; k++) {
      if(result == primes.get(k)) {
         flag = true;
         break;
      }
   }

   primes.add(result);

   if(flag)
   {
      break;
   }
}

if(!flag)
{
   System.out.println("Primitive root: "+primitiveRoot);
   return primitiveRoot;
```

```
            }


        }
        return primitiveRoot;
    }

    static long power(long a, long X, long p)
    {
        long result = 1;

        //In case a is more than or equal to p.
        a = a % p;

        while (X > 0)
        {
            //Multiply a with the result mod p, if X is odd.
            if (X % 2 == 1)
            {
                result = (result * a) % p;
            }

            //Therefore, X is now even
            X = X >> 1; // y = y/2
            a = (a * a) % p;
        }
        return result;
    }
}
```