IMPLEMENTACIÓN DEL ALGORITMO

MELANIE PINTO DOS ANJOS

PRIMERA VERSIÓN DEL PROGRAMA

Tres clases

- Inicializa.java
- Usuario.java
- Main.java

Toma de contacto con el algoritmo.

Trabajo con números enteros por simplicidad.

Comprobación de la corrección de la implementación.

PRIMERA VERSIÓN DEL PROGRAMA

Clase Inicializa.java

```java
public class Inicializa {

    private int primo;
    private int generador;

    public Inicializa(){
        int p = new Random();
        int g = new Random();
        while(p.compareTo(g)<0){ //Mientras p sea menor que g
            g = new Random();
        }
        primo = p;
        generador = g;
    }

    public int getPrimo(){
        if(primo!=null){
            return primo;
        }
        return null;
    }

    public int getGenerador(){
        if(generador!=null){
            return generador;
        }
        return null;
    }

    public boolean setPrimo(BigInteger p){
        primo = p;
        return true;
    }

    public boolean setGenerador(BigInteger g){
        generador = g;
        return true;
    }
}
```

PRIMERA VERSIÓN DEL PROGRAMA

Clase Usuario.java

```java
public class Usuario {
    private String nombre;
    private int primo;
    private int generador;
    private int secreto;
    private int clave;
    private int sharedkey;

    public Usuario(String n){
        if(n!=null){
            nombre = n;
        }
    }

    public void asignaClavesPublicas(int p, int g){
        primo = p;
        generador = g;
    }

    public BigInteger calculaSecreto(){
        secreto = new Random();
        while(primo.compareTo(secreto)<0){ //Mientras primo sea menor que s
            secreto = new Random();
        }
        return secreto;
    }

    public BigInteger calculaClave(){
        clave = pow(generador, secreto) % primo;
        return clave;
    }

    public BigInteger secretoComun(BigInteger clave){
        sharedkey = pow(clave, secreto) % primo;
        return sharedkey;
    }
```

PRIMERA VERSIÓN DEL PROGRAMA

Clase Main.java

```java
public class Main {

    //Intercambio de claves Diffe-Hellman entre dos clientes
    //Compresión y Seguridad
    public static void main(String[] args) {

        //INICIALIZAMOS EL SISTEMA Y GENERAMOS LAS DOS CLAVES PÚBLICAS
        //GUARDAMOS AMBOS VALORES PARA UTILIZARLOS POSTERIORMENTE
        Inicializa init = new Inicializa();
        BigInteger primo = init.getPrimo();
        BigInteger generador = init.getGenerador();

        System.out.print("*** Claves públicas *** \nPrimo: "+primo+"\nGenerador: "+generador+"\n");

        //DECLARAMOS E INICIALIZAMOS LOS DOS CLIENTES QUE USARÁN EL SISTEMA
        //LES ASIGNAMOS LAS CLAVES PUBLICAS GENERADAS ANTERIORMENTE
        Usuario a = new Usuario("Alberto");
        a.asignaClavesPublicas(primo, generador);
        a.calculaSecreto();
        a.calculaClave();

        Usuario b = new Usuario("Beatriz");
        b.asignaClavesPublicas(primo, generador);
        b.calculaSecreto();
        b.calculaClave();

        //AMBOS CLIENTES CALCULAN LA CLAVE COMPARTIDA
        //Esto deberia ser privado, no deberia poder acceder.
        BigInteger sharedAB = a.secretoComun(b.getClave());
        BigInteger sharedBA = b.secretoComun(a.getClave());

        if(sharedAB.compareTo(sharedBA)==0){
            System.out.println("La clave compartida es igual.");
        }
        else{
            System.out.println("La clave compartida es distinta.");
        }

    }
```

SEGUNDA VERSIÓN DEL PROGRAMA

Mismas tres clases

- Inicializa.java
- Usuario.java
- Main.java

Inclusión de la librería BigInteger.

Comprobación del funcionamiento correcto.

Programa listo para implementación de red e interfaz.

SEGUNDA VERSIÓN DEL PROGRAMA

Cambios en la clase Usuario.java (implementación de la librería BigInteger)

```
public void asignaClavesPublicas(BigInteger p, BigInteger g){
    primo = p;
    generador = g;
}

public BigInteger calculaSecreto(){
    secreto = new BigInteger(numBits, new Random());

    while(primo.compareTo(secreto)<0){ //Mientras primo sea menor que s
        secreto = new BigInteger(numBits, new Random());
    }

    return secreto;
}


/** << ALGORITMO >>
 * Alice and Bob agree to use a modulus p = 23 and base g = 5
 * (which is a primitive root modulo 23).
 * Alice chooses a secret integer a = 6,
 * then sends Bob A = g^a mod p
 * A = 5^6 mod 23 = 8
 * A = g^a mod p **/

//modPow(BigInteger exponent, BigInteger m)
//Returns a BigInteger whose value is (thisexponent mod m)
public BigInteger calculaClave(){
    clave = generador.modPow(secreto, primo);
    return clave;
}

public BigInteger secretoComun(BigInteger clave){
    sharedkey  = clave.modPow(secreto, primo);
    return sharedkey;
}
```