package danji;

import java.io.IOException;

import java.util.Arrays;

import java.util.Random;

//后处理

public class Post\_processing\_for\_quantum\_key\_distribution {

public static void main(String[] args) {

long startTime = System.currentTimeMillis(); //获取开始时间

//Alice的密钥

int[] aliceKeys;

//Bob的密钥

int[] bobKeys;

//密钥总长度

int n = 50000;

int n1=n;

//误码率

double p = 0.01;

//泄露信息量

int leakedMsg = 0;

//已纠正的错误数

int correctedErr = 0;

//Eve连续知道i个密钥的概率的阈值

double X=Math.pow(10,-10);

int n2=0;

// System.out.println(X);

//产生Alice，Bob的密钥

aliceKeys = Utility.aliceKeysGenerator(n);

bobKeys = Utility.bobKeysGenerator(aliceKeys,n,p);

////测试Bob的误码随机性是否达到标准

Simulation simulation=new Simulation(aliceKeys);//获取数据

Utility.writeKeyToFile("AliceKey.txt", aliceKeys);

//随机抽样检测误码

Estimate Estimate=new Estimate(aliceKeys,bobKeys,n,p);

System.out.println("随机测试误码在设定误码率范围内");//文字输出

Utility.writeKeyToFile("BobKey.txt", bobKeys);

//一致性检验

int i=0;

/\*while(循环条件){

　　\*循环操作

}\*///循环条件为true时执行循环

while(!Consistencytest.compareHash("AliceKey.txt","BobKey.txt")){//调用Consistencytest文件的compareHash对比结果，如果一致，得到true则循环条件为flase，停止循环

// System.out.println("比特不一致，退回密钥协商继续协商");//文字输出

i++;//i表示循环纠错次数

// System.out.println("第"+i+"次Cascadei");

Cascadei cascadei = new Cascadei(aliceKeys,bobKeys,n,p);//把alice和bob密钥，密钥长度，误码率输入纠错函数中

bobKeys = cascadei.cascadei();//把bob密钥刷新为纠错后密钥

Utility.writeKeyToFile("BobKey.txt", bobKeys);//调用Utility文件把密钥写进文档中

leakedMsg += cascadei.getLeakedMsg();//泄露信息量，调用纠错算法中的信息量函数结果

correctedErr +=cascadei.getcorrectedErr();//调用纠错个数作累加

}

//MD5值一致后停止循环

//该协议最终返回协商效率

System.out.println("协商效率为"+ (1 - (double)leakedMsg/n));

System.out.println("经过"+i+"轮纠错，纠正比特数为"+correctedErr+"\nMD5值一致，进入下一个环节");

System.out.println("\n泄露的信息量为："+leakedMsg);

// System.out.println("\n密钥长度为："+n1);

Random r = new Random();//产生随机数

double Y=(double) leakedMsg/n1;//Eve知道单个密钥的概率，泄露量/密钥原长度

System.out.println("\n纠错后Eve知道单个密钥的概率为"+Y);

double num0= Math.log(X) / Math.log(Y);//Eve连续知道i个密钥的概率，num0=i

System.out.println("计算出连续第"+num0+"密钥被Eve知道的概率极小");//可当作当连续第num0个密钥时，Eve不能知道超过这个长度的连续密钥

int num=(int)Math.floor(num0);//每次循环删掉的连续比特数

// System.out.println("连续"+num+"密钥，向下取整");

int delete=(int) Math.floor(leakedMsg/num);//循环次数=总泄露量/每次循环删掉的连续比特数

// System.out.println("循环次数为"+delete);

double r0=n1/num;//随机数范围，指在整串密钥中选择从哪一位的num倍开始替换删除，随机数最大值\*num＜密钥总长度

// System.out.println("随机数范围最大值为"+r0);

int r1=(int) Math.floor(r0);

System.out.println("随机数范围最大值为"+r1);//

for (int time =0; time < delete; time++) {//循环次数可改变

int index = r.nextInt(r1);//在随机数中中选择范围在r1以内的数给index幅值，随机数范围是0到r1

// System.out.println("选择的随机数为"+index);

for (int j = index\*num; j < n-num; j++) {//初始值设为index的num倍，在小于密钥长度-num时停止循环，每次循环+1

aliceKeys[j] = aliceKeys[j+num];//对密钥数组，把密钥的第j位替换成第j+num位

bobKeys[j] = bobKeys[j+num];

}//内for循环，循环目的是从随机的位数开始，该数值替换成后num位，在倒数num位的时候停下，即实际在密钥中减少了num位的有效比特

n -= num;//密钥长度减num

aliceKeys = Arrays.copyOfRange(aliceKeys, 0, n);//把替换后的密钥复制到新的密钥中，长度为原密钥-num，即原密钥中的最后一百位比特也抛弃掉了

bobKeys = Arrays.copyOfRange(bobKeys, 0, n);

}//纠错之后丢掉部分密钥增强安全性

// System.out.println(n); //删除比特后的密钥长度

System.out.println("\n随机删掉："+(delete)\*num+"比特后\n");

//外for循环，循环了delete次，总共减掉了delete\*num个比特

n2=n;//纠错后的密钥长度

int leakedMsg2=leakedMsg;//纠错后得到的泄露信息量

Utility.writeKeyToFile("AliceKey\_final.txt", aliceKeys);//把删掉后的密钥写进文件中

Utility.writeKeyToFile("BobKey\_final.txt", bobKeys);

if(Consistencytest.compareHash("AliceKey\_final.txt","BobKey\_final.txt")){//比较MD5值结果

System.out.println("MD5值相同\n删除完成\n进行保密增强");

// System.out.println("AliceKey: "+Arrays.toString(aliceKeys));//输出密钥

// System.out.println("BobKey : "+Arrays.toString(bobKeys));

}

Enhance enhance = new Enhance(n2,leakedMsg2);//获取数据,把n2和泄露量送到保密增强

// System.out.println("托普利兹矩阵:");

// enhance.showMatrix(enhance.toeplitz);//展示矩阵

int[][] Alicefinal\_Key\_Matrix = enhance.enhancealiceKeys();//建立最终密钥矩阵，调用enhancekey结果

int[][] Bobfinal\_Key\_Matrix = enhance.enhancebobKey();

// System.out.println("最终密钥:");

// enhance.showMatrix(Bobfinal\_Key\_Matrix);//展示密钥矩阵

int sum1=0;

for(int i1=0;i1<Bobfinal\_Key\_Matrix.length;i1++) {

for(int l=0;l<Bobfinal\_Key\_Matrix[i1].length;l++) {

sum1++;

}

}

// System.out.println("\n经过"+i+"轮纠错，纠正比特数为"+correctedErr);

// System.out.println("\n纠错后随机删掉："+(sum+1)\*100+"比特");

System.out.println("最终密钥长度:"+sum1);

System.out.println("安全密钥率:"+(double)sum1/n1);

Utility.writeKeyToFile2("Alicefinal\_Key\_Matrix.txt",Alicefinal\_Key\_Matrix);//把保密增强后的密钥写进文件中

Utility.writeKeyToFile2("Bobfinal\_Key\_Matrix.txt", Bobfinal\_Key\_Matrix);

long endTime = System.currentTimeMillis(); //获取结束时间

long Time=endTime - startTime;

System.out.println("程序运行时间：" + Time + "ms"); //输出程序运行时间

}

}