package danji;

import java.nio.ByteBuffer;

import java.util.\*;

import java.util.stream.Collectors;

public class Enhance {

//保密增强

int[] aliceKeys;//Alice密钥

int[] bobKeys;//Bob密钥

int[][] toeplitz;//托普利兹矩阵

int rawlength=0;//托普利兹矩阵行数

//泄露信息量

int leakedMsg;

//预设一个使安全密钥达到标准的值

double I=Math.pow(10,-100);//I概率论中指变量间的相互依赖性的量度

//纠错后密钥长度

int n;

public Enhance(int a,int b){

this.aliceKeys = Utility.readKeyFromFile("AliceKey\_final.txt");//表示aliceKeys是用了Utility类中的readKeyFromFile文件函数的"AliceKey\_final.txt"，this.表示成员变量

this.bobKeys = Utility.readKeyFromFile("BobKey\_final.txt");//读取文件中的密钥

this.leakedMsg=b;

this.n=a;

this.rawlength=getrawlength();

this.toeplitz = genToeplitz();//当前对象是genToeplitz函数结果

}//保密增强类

public int getrawlength()

{ double R00=0;//中间参数-S

double R0=0;//希望得到的最终安全密钥长度

double P=I\*Math.log(2);//信息量I和ln2的相乘，等于2的（-S）次方，S=纠错后密钥长度n2-纠错泄露信息量leakedMsg-最终密钥长度R0

// System.out.println("一个分母"+P);

R00=Math.log(P)/Math.log(2);//对上式作对数运算，提取出（-S）次方，R00为-S

// System.out.println("一个参数"+R00);

R0= R00+n-leakedMsg;//S=n2-leakedMsg-R0,但我们需要的是R0，作逆运算，R0=n2-leakedMsg-S,R00=-S,

// System.out.println("希望密钥长度"+R0);

int R=(int)Math.floor(R0);//得到安全密钥长度,取整数

System.out.println("希望密钥长度取整"+R);

int N=0;//T矩阵行长

double N1=R\*100;//列长固定为一百，列长\*希望密钥长度为N1

N=(int) N1/n;//因为最终安全密钥等于T矩阵行长乘以（纠错后密钥长度除以T矩阵列长），作逆运算。即行长等于N1除以纠错后密钥长度

System.out.println("托普利兹矩阵行长"+N);

return N;

}

//得到一个20\*20的矩阵options

public int[][] get\_options(){

int[][] options = new int[rawlength][rawlength];//二维数组，行列20\*20

int[] raw = new int[rawlength-1];//一维数组，长度19

Random random = new Random();//变量random，构造方法Random， Random()：创建一个新的随机数生成器。

for (int i = 0; i < rawlength-1; i++) {

raw[i] = random.nextInt(2);//random.nextInt（）生成某范围内的随机数字，这里的范围是0~1

}//for循环，把随机数写入raw数组中，循环19次

for (int i = 0; i < rawlength; i++) {

options[i][0] = 1;//把1写入每行的第一个位置

List<Integer> list = Arrays.stream(raw).boxed().collect(Collectors.toList());//建立一个Integer类型的接口，把int数组raw转入接口list

//List是接口， List<Integer>是指在List中Integer的类型

// Arrays.stream(arr) 可以替换成IntStream.of(arr)。

// 1.使用Arrays.stream将int[]转换成IntStream。

// 2.使用IntStream中的boxed()装箱。将IntStream转换成Stream<Integer>。

// 3.使用Stream的collect()，将Stream<T>转换成List<T>，因此正是List<Integer>。

Collections.shuffle(list);//随机打乱接口list内数据的顺序

System.arraycopy(list.stream().mapToInt(Integer::valueOf).toArray(), 0, options[i], 1, rawlength-1);//使用arraycopy（）方法来实现数组之间的复制

//这个复制是在for循环中，对options数组的第i行所在的一维数组，从第1位（第0位固定是1）开始把raw的数值（从0开始）写入19长度的数，每循环一次换一行。

//list.stream().mapToInt(Integer::valueOf).toArray()是List<Integer>转int[]的表示，转化的过程中,转换的基本数据类型要与引用数据类型保持一致

//toArray() = |1|2|3|4|...

//options[i] = |0|0|0|0|0|...

//执行System.arraycopy(toArray(), 0, options[i], 1, 19);时

//第一步：从源数组(toArray())中，从下标0开始取，取19个，也就是toArray[0]-toArray[18]，即....19个数

//第二步：把取出的数，按顺序，存放到目标数组（options[i]）中，从下标1开始存，存19个，也就是options[1]-options[19]

//所以数组options[i]为：|0|1|2|3|4|....

}//for循环，循环20次，得到的options数组的第一列都是1，其余列数由raw数组进行复制，得到20\*20的矩阵

// showMatrix(options);

return options;

}

//得到一个md5截取长度除以0.2的取模（返回余数），得到一个整型数据

public int getSeqNumber(){

Random random = new Random();//变量random，构造方法Random， Random()：创建一个新的随机数生成器。

int m = random.nextInt(this.aliceKeys.length);//m是随机一个在aliceakaeys长度内的数

int n = random.nextInt(this.aliceKeys.length);//n同上，m,n的范围在Keys.length内

int[] parts = Arrays.copyOfRange(this.aliceKeys, Math.min(m, n), Math.max(m,n));//一维数组，把Alice的密钥复制到这个数组，长度是m-n的绝对值

//Arrays.copyOfRange(T[ ] original,int from,int to)

//将一个原始的数组original，从下标from开始复制，复制到上标to，生成一个新的数组。注意这里包括下标from，不包括上标to。

byte[] bytes\_parts = new byte[parts.length];//定义一个parts长度的定长字节数组

for (int i=0; i < bytes\_parts.length;i++) {

bytes\_parts[i] = (byte) parts[i];

}//for循环，循环bytes\_parts的数组长度，把parts的数值写入bytes\_parts数组中。

String md5 = Utility.MD5(ByteBuffer.wrap(bytes\_parts));//计算MD5校验，根据一个byte[]来生成一个固定的ByteBuffer时，使用ByteBuffer.wrap()

return Integer.parseInt(md5.substring(md5.length() - 6), 16) % rawlength;//Integer.parseInt()是将String字符类型数据转换为Integer整型数据，md5是String类型

}//substring(start,end)，截取指定字符上的字符串：开始值是md5长度-6（包括），末尾值是第16位置（不包括），是左闭右开区间

public int[][] genToeplitz(){

int[][] toeplitz = new int[rawlength][100];//二维数组，new int创建动态整性数组，长度分别为20\*100，20行，100列

// System.out.println(rawlength+"个行:");//文字输出

int[] raw\_row = get\_options()[getSeqNumber()];//一维数组，调用get\_options嵌套getSeqNumber函数的函数结果，得到一个20\*20矩阵再和数值做乘法的一维数组

for (int i = 0; i < 100/rawlength; i++) {

System.arraycopy(raw\_row, 0, toeplitz[0], i\*rawlength, rawlength);//把raw-row的数据填入T矩阵的第一行的一维矩阵中，填满100个数，重复二十次

//arraycopy()的作用是复制,在这里是把raw\_row数组上复制到toeplitz数组

//arraycopy(Object src, int srcPos, Object dest, int destPos,int length)

//arraycopy(被复制的数组, 开始复制元素的下标, 要复制到的数组（列）, 目标数组的下标位置, 一共需要复制的元素个数（长度）);

}//一个for循环，从0到4，循环5次，复制目的地的下标数分别为0，20，40，60，80

//这个for循环从raw\_row 数组中选取了每行20个，每列20个复制到toeplitz数组，从下标0开始替换

// System.out.println(rawlength+"个列:");

int[] line = get\_options()[getSeqNumber()];//一维数组，调用get\_options嵌套getSeqNumber函数的函数结果，getSeqNumber用来决定选择的行数，在get\_options()的20\*20矩阵中选取第getSeqNumber行

for (int i = 0; i < rawlength; i++) {

toeplitz[i][0] = line[i];

}//for循环，循环20次，使得toeplitz二维数组上的列数替换成line数组中相应的下标

for (int i = 1; i < rawlength; i++) {

System.arraycopy(toeplitz[i - 1], 0, toeplitz[i], 1, 99);//把从第i行开始的数替换成第i-1行，列数相差1，复制99位，达到从上到下逐渐右移的效果

}//for循环，循环19次。

return toeplitz;//返回this.指向目标

}//托普利兹矩阵的构成

public void showMatrix(int[][] options){

for (int[] option : options) { //for (int[] : )for each，即for option数组的每个元素

for (int j = 0; j < options[0].length; j++) {

System.out.printf("%d ", option[j]);

}//for循环，输出矩阵的数

System.out.println();

}

}

public int[][] enhancealiceKeys(){

int [][] aliceKeysMatrix = Matrix.intArray2Matrix(this.aliceKeys, 100);//调用Alice密钥矩阵

return Matrix.matrixMul(this.toeplitz, aliceKeysMatrix);//保密增强计算

}

public int[][] enhancebobKey(){

int [][] bobKeysMatrix = Matrix.intArray2Matrix(this.bobKeys, 100);//调用Alice密钥矩阵

return Matrix.matrixMul(this.toeplitz, bobKeysMatrix);//保密增强计算

}

/\* public static void main(String[] args) {

Enhance enhance = new Enhance();//获取数据

System.out.println("托普利兹矩阵:");

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

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

System.out.println("最终密钥:");

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

}//main函数，\*/

}