package danji;

import java.io.\*;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

// 密钥总长度

// int n = 10000;

// 误码率

// double p = 0.01;

// 块长参数

// double alpha = 0.58;

// 块长增长参数

// double beta = 2.5;

//效用

public class Utility {

//Bob的密钥发生器

//n为密钥长度

//p为误码率

public static int[] bobKeysGenerator(int[] aliceKeys, int n, double p) {

Random r = new Random();

int[] keys = Arrays.copyOf(aliceKeys, n);//复制alice的数组，长度为n（即完全复制）

//误码个数

int errorNum=(int)(n \* p);//p是误码率，是小数，这里用了一个数据类型转换，得到了总共生成的误码个数

//误码所在的下标，不能重复所以用Set

Set<Integer> errorIndex = new HashSet<>();//建立集合

while (errorIndex.size() < errorNum){//集合大小小于误码个数

errorIndex.add(r.nextInt(n));//在范围n中把随机数写入集合

}//while循环的目的是在误码个数等长的集合中写入随机数，随机数代表误码的位置

//Bob的密钥是在Alice密钥基础上增加误码

for (int index : errorIndex) {//对数组errorIndex的每个元素记为index

keys[index] ^= 1;//密钥在index的位置上把比特进行翻转

}

return keys;

}//生成Bob密钥

//Alice的密钥发生器

public static int[] aliceKeysGenerator(int n) {

//用于产生随机数

Random r = new Random();

int[] keys = new int[n];

for (int i=0; i<keys.length; i++) {

keys[i] = r.nextInt(2);//在01中随机取值写入数组

}

return keys;

}

public static void writeKeyToFile(String filename, int[] keys){

//输入流

File keysFile = new File(filename);

try (OutputStream f = new FileOutputStream(keysFile)){

for (int key : keys) {

f.write(key);

}

f.close();

}catch (IOException e) {

e.printStackTrace();

}

}

public static int[] readKeyFromFile(String filename){//读取文件数据并转成int数组

int c;

List<Integer> list = new ArrayList<Integer>();

File keysFile = new File(filename);

try (InputStream f = new FileInputStream(keysFile)){

while ((c = f.read()) != -1){

list.add(c);

}

return list.stream().mapToInt(Integer::valueOf).toArray();//把接口转回int

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static void writeKeyToFile2(String filename, int[][] keys){//把字符串写入文档中，第一个为文档名，第二个为写入的一维数组

//输入流

File keysFile = new File(filename);//创建文件流

// File file = new File("D:\\java\\content\\eclipse-win64\\S\\java\\fd.properties");

//要么是去在全路径基础上去掉项目名

try {

FileWriter out = new FileWriter(keysFile);

String line = System.getProperty("line.separator");

// String s1=""; //定义一个空字符串

// StringBuffer sb=new StringBuffer(s1); //创建字符串缓冲区对象sb，将s1对象传进去（如果不用缓冲区的话，没加一个数组元素，及重新生成一个String对象）

for (int i = 0; i < keys.length; i++) {

//对于数组key遍寻

// sb.append(keys[i]); //遍历int数组，追加到sb;

String s2=Arrays.toString(keys[i]); //返回缓冲区对象的toString(),即得到的字符串

out.write(s2);//把数据写入文件中

out.write(line);

}

out.close();

}catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* 计算MD5校验

\* @param buffer

\* @return 空串，如果无法获得 MessageDigest实例

\*/

public static String MD5(ByteBuffer buffer) {

char[] hexDigits = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',

'a', 'b', 'c', 'd', 'e', 'f'};

String s = "";

try {

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(buffer);

byte[] tmp = md.digest(); // MD5 的计算结果是一个 128 位的长整数，

// 用字节表示就是 16 个字节

char[] str = new char[16 \* 2]; // 每个字节用 16 进制表示的话，使用两个字符，

// 所以表示成 16 进制需要 32 个字符

int k = 0; // 表示转换结果中对应的字符位置

for (int i = 0; i < 16; i++) { // 从第一个字节开始，对 MD5 的每一个字节

// 转换成 16 进制字符的转换

byte byte0 = tmp[i]; // 取第 i 个字节

str[k++] = hexDigits[byte0 >>> 4 & 0xf]; // 取字节中高 4 位的数字转换, >>>,

// 逻辑右移，将符号位一起右移

str[k++] = hexDigits[byte0 & 0xf]; // 取字节中低 4 位的数字转换

}

s = new String(str); // 换后的结果转换为字符串

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

return s;

}

//比较哈希值是否一致

/\*\*

\* Get MD5 of a file (lower case)

\* @return empty string if I/O error when get MD5

\*/

/\*\*

\* 获取文件的MD5值

\*

\* @param file

\* 目标文件

\* @return MD5字符串

\*/

public static String getFileMD5(File file) {//filenameA/filenameB获取文件

// \* try {

// \*可能出现异常的代码

// \*} catch（异常类名A e）{

// \*如果出现了异常类A类型的异常，那么执行该代码

// \*} ...（catch可以有多个）

try (FileInputStream in = new FileInputStream(file)) {

FileChannel ch = in.getChannel();//获取in文件通道

String md5 = MD5(ch.map(FileChannel.MapMode.READ\_ONLY, 0, file.length()));//将此通道文件的一个区域直接映射到内存中。调用了MD5函数的结果

ch.close();//关闭ch文件

return md5;//返回md5值

} catch (IOException e) {

return e.getMessage();

}

}

}