

专业课程设计I报告

（ 2019 / 2020学年 第 二 学期）

题 目： 文件完整性检测系统

|  |  |
| --- | --- |
| **专 业** | **信息安全** |
| **学 生 姓 名** | **陈柯彤** |
| **班 级 学 号** | **B17040904** |
| **指 导 教 师** | **曹晓梅** |
| **指 导 单 位** | **信息安全系** |
| **日 期** | **2020.06.08-2020.06.18** |

|  |  |  |
| --- | --- | --- |
| **支撑指标点** | **评价准则** | **计分（每项10分）** |
| 课程目标1：通过课程设计，培养学生综合应用密码学、计算机技术等领域专业知识的技能，养成良好的职业道德。（20分） | 1、能够掌握密码学的相关基础知识，并能够针对求解的工程问题，收集资料进行合理的分析与设计。。 |  |
| 2、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能。 |  |
| 课程目标2：培养学生解决信息安全领域复杂工程问题的实践创新能力。（20分） | 3、通过调研，能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现。 |  |
| 4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便。 |  |
| 课程目标3：提高学生文献查阅与资料收集能力，发现问题、研究问题、分析问题与解决问题的能力。（10分） | 5、能够对实际工程问题进行形式化描述，给出数据结构和算法的设计描述，给出关键算法的流程图或伪代码，并给出各算法之间的结构关系描述。 |  |
| 课程目标4：密码学相关的工程基础实验验证与实现能力，能够根据要求设计实验方案并开展实验，能够对实验数据进行解释与对比分析，给出实验结论。（20分） | 6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，理解相关算法，能够完成课题要求的各项任务和指标。 |  |
| 7、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试。 |  |
| 课程目标5：分组完成一次课程设计与开发的全过程，组内成员通过讨论和交流解决实验中的难题，能在实验报告中准确阐述实验内容，能够在答辩时清晰陈述观点和回答问题。（30分） | 8、组内成员之间有一定的团队合作，互通有无。 |  |
| 9、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范。 |
| 10、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的工程基础知识具有较好的理解和掌握。 |
| 专业课程设计I能力测评总分 | |  |
| **指导教师： 年 月 日** | | |
| **备注：** | | |

**文件完整性检测系统**

**一、课题内容和要求**

在网络空间，文件可能因为感染病毒、植入木马/后门、人为篡改或者传输故障等原因导致被篡改。本题目致力于设计和实现一款文件完整性检测系统，通过CRC32、MD5或SHA-1等算法来比对确定文件的状态，从而可以及时检测出文件的异常改动，使学生深入了解完整性校验算法的实现原理，提升编程能力。

题目的具体要求如下：

1. 查阅相关资料，掌握CRC32、MD5或SHA-1等完整性校验算法的原理和实现细节；

(2) 掌握Windows系统不少于两种类型文件（如DLL、DRV、SYS、OCX、EXE、COM、

PIF、SCR等）的获取、结构特点及散列值的计算和比较；

1. 选择一种编程语言和开发工具，设计和实现系统文件完整性检测系统，对不同类型、不同大小的文件进行完整性验证，分析比较检测的准确率、执行效率等指标；
2. 程序具有图形化用户界面，输出美观；
3. 可根据自己能力，在完成以上基本要求后，对程序功能进行适当扩充；
4. 撰写报告，对所采用的算法、程序结构和主要函数过程以及关键变量进行详细的说明；对程序的调试过程所遇到的问题进行回顾和分析，对测试和运行结果进行分析；总结软件设计和实习的经验和体会，进一步改进的设想；
5. 提供关键程序的清单、源程序及可执行文件和相关的软件说明。

**二、课题需求分析**

本课题目标系统“文件完整性检测系统”的功能框架图如图1所示。

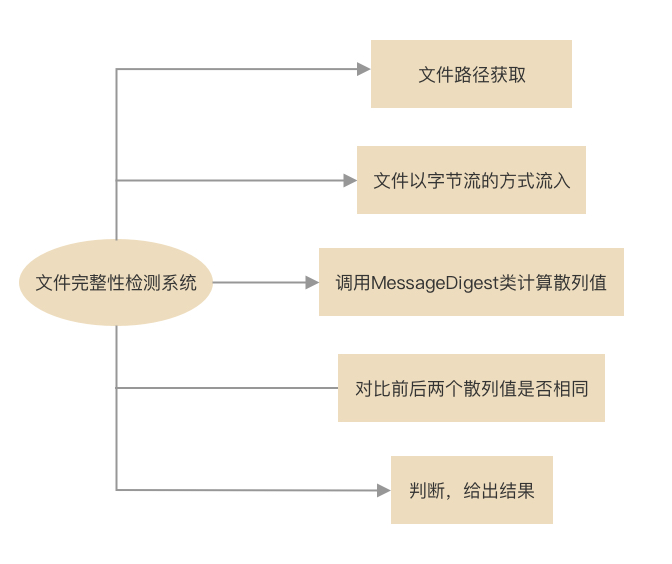


图1.功能框架

1. 能够通过路径找到文件，以文件流读入
2. 提供可用的 GUI
3. 能计算各种文件类型的散列值
4. 由于 MD5 已经无法防止碰撞攻击，所以也使用了SHA-256 算法

**三、课题相关数据结构及算法设计**

1主要类与函数

public class MD5Util extends JFrame { //软件Jframe界面

public MD5Util(){

super("MD5"); //为界面命名

setSize(600,500); //设置界面大小

setDefaultCloseOperation(EXIT\_ON\_CLOSE); //给界面添加窗口关闭选项

}

public static String md5HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath); //将文件用留的方式读入

return md5HashCode(fis);

}

public static String md5HashCode(InputStream fis) //计算文件md5的函数

public static String sha256HashCode(InputStream fis) //计算文件SHA-256的函数

MessageDigest md = MessageDigest.getInstance(""); //调用java中的MessageDigest类

BigInteger bigInt = new BigInteger(1, md5Bytes); //调用javaBigIntel类对数取绝对值

byte[] buffer = new byte[1024]; //由于文件较大，分多次将文件读入

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

md.update(buffer, 0, length);

2 主要算法流程

1. 文件读入算法： 基于字符串转md5思想如图2，为了计算文件的md5等散列值，该算法调用java中的FileInputStream类方法将文件以字节流的方式读入。这样即可按照分段的方法进行散列值的计算。算法思想入图3

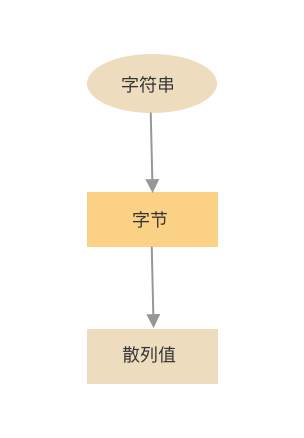


图2

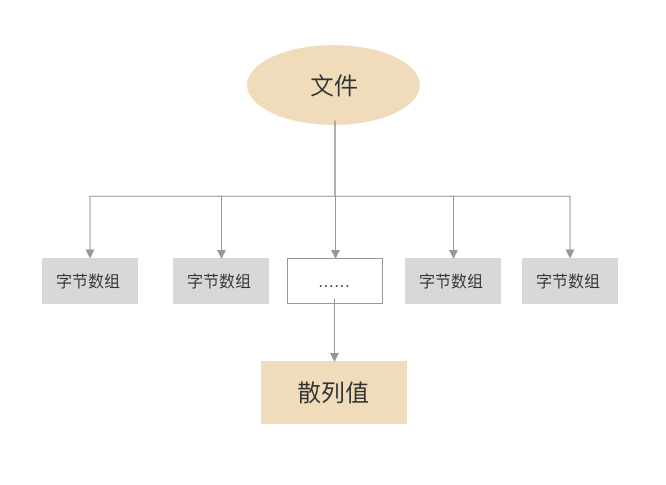


图3

1. 散列值算法：

选择了简单的 MD5 算法，MD5 输入不定长度信息，输出固定长度128-bits的算法。生成四个32位数据，最后联合起来成为一个128-bits[散列](https://zh.wikipedia.org/wiki/%E6%95%A3%E5%88%97" \o "散列)，基本操作为求余数、取余数、调整长度。但 MD5 的碰撞攻击小于等于 18 bits。

所以同样也尝试了SHA-2家族的 SHA-256 算法，SHA-256 输出散列值长度为256，循环次数为 64 次，碰撞攻击为128 bits。

两种散列算法都没有自己造轮子，而是借助了 java.security.MessageDigest 类，该类提供了常见的消息摘要算法，可用于密码散列函数，例如 SHA-1 SHA-256 MD5。通过 getInstance()方法可以指定使用的算法，例如可以传入字符串 “MD5” 来获得一个实现了 MD5 的MessageDigest 对象，之后可以调用该对象的 update 方法来处理数据。

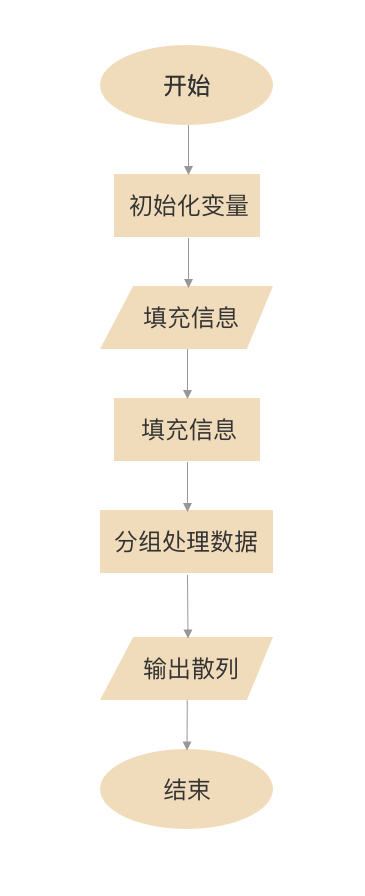


图4

**四、源程序代码**

import java.awt.Color;

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.InputStream;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import static javax.swing.JFrame.EXIT\_ON\_CLOSE;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.filechooser.FileNameExtensionFilter;

public class MD5Util extends JFrame {

String s; //前文件路径

String w; //后文件路径

public MD5Util(){

super("文件完整性检测");

setSize(600,500);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

MD5Util md5=new MD5Util();

Container contentPane = md5.getContentPane();

JPanel j=new JPanel();

j.setBackground(Color.white);

JPanel j1=new JPanel();

j1.setBackground(Color.white);

md5.setLayout(null);

j.setBounds(0, 100, 600, 50);

j1.setBounds(150, 180, 200, 50);

JLabel label1=new JLabel("输入原文件路径");

JLabel label2=new JLabel("输入后文件路径");

JButton b=new JButton("检测");

JButton b1=new JButton("浏览");

JButton b2=new JButton("浏览");

j.add(b1);

j.add(b2);

j1.add(b);

j.add(label1);

j.add(label2);

JTextField jf1 =new JTextField(400);

JTextField jf2 =new JTextField(400);

JTextArea jf3 =new JTextArea();

j.add(jf1);

j.add(jf2);

j.add(jf3);

j.setLayout(null);

label1.setBounds(0, 10, 100, 20);

label2.setBounds(0, 30, 100, 20);

b1.setBounds(500, 10, 100, 20);

b2.setBounds(500, 30, 100, 20);

jf1.setBounds(110, 10, 300, 20);

jf2.setBounds(110, 30, 300, 20);

jf3.setBounds(0, 230, 600, 200);

md5.add(j);

md5.add(j1);

md5.add(jf3);

b1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { //按钮点击事件

JFileChooser chooser = new JFileChooser("C:\\Users\\w\\Desktop"); //设置选择器

int returnVal = chooser.showOpenDialog(b1); //是否打开文件选择框

if (returnVal == JFileChooser.APPROVE\_OPTION) { //如果符合文件类型

md5.s = chooser.getSelectedFile().getAbsolutePath(); //获取绝对路径

jf1.setText(md5.s);

}

}

});

b2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { //按钮点击事件

JFileChooser chooser = new JFileChooser("C:\\Users\\w\\Desktop"); //设置选择器

int returnVal = chooser.showOpenDialog(b2); //是否打开文件选择框

if (returnVal == JFileChooser.APPROVE\_OPTION) { //如果符合文件类型

md5.w = chooser.getSelectedFile().getAbsolutePath(); //获取绝对路径

jf2.setText(md5.w);

}

}

});

b.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

if(jf1.getText().trim().equals("")||jf2.getText().trim().equals("")){

JOptionPane.showMessageDialog(null, "请输入两个对比文件路径", "检测", JOptionPane.ERROR\_MESSAGE);

}

else{

try {

md5.s=jf1.getText();

String s1=md5HashCode(md5.s);

String s2=sha256HashCode(md5.s);

md5.w=jf2.getText();

String w1=md5HashCode(md5.w);

String w2=sha256HashCode(md5.w);

jf3.setText(s1+"\t源文件MD5值\n"+w1+"\t后文件MD5值\n"+s2+"\t源文件SHA-256值\n"+w2+"\t后文件SHA-256值");

if(s1.equals(w1)&&s2.equals(w2)){

JOptionPane.showMessageDialog(null, "文件未被修改", "检测结果", JOptionPane.INFORMATION\_MESSAGE);

}

else{

JOptionPane.showMessageDialog(null, "文件已被修改", "检测结果", JOptionPane.WARNING\_MESSAGE);

}

} catch (FileNotFoundException ex) {

Logger.getLogger(MD5Util.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

});

md5.setVisible(true);

}

public static String md5HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath);

return md5HashCode(fis);

}

public static String md5HashCode(InputStream fis) {

try {

//拿到一个MD5转换器,如果想使用SHA-1或SHA-256，则传入SHA-1,SHA-256

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

//分多次将一个文件读入，对于大型文件而言，比较推荐这种方式，占用内存比较少。

byte[] buffer = new byte[1024];

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

md.update(buffer, 0, length);

}

fis.close();

//转换并返回包含16个元素字节数组,返回数值范围为-128到127

byte[] md5Bytes = md.digest();

BigInteger bigInt = new BigInteger(1, md5Bytes); //1代表绝对值

return bigInt.toString(16); //转换为16进制

} catch (Exception e) {

e.printStackTrace();

return "";

}

}

public static String sha256HashCode(String filePath) throws FileNotFoundException{

FileInputStream fis = new FileInputStream(filePath);

return sha256HashCode(fis);

}

public static String sha256HashCode(InputStream fis) {

try {

//拿到一个MD5转换器,如果想使用SHA-1或SHA-256，则传入SHA-1,SHA-256

MessageDigest sha256 = MessageDigest.getInstance("SHA-256");

//分多次将一个文件读入，对于大型文件而言，比较推荐这种方式，占用内存比较少。

byte[] buffer = new byte[1024];

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1) {

sha256.update(buffer, 0, length);

}

fis.close();

//转换并返回包含16个元素字节数组,返回数值范围为-128到127

byte[] sha256Bytes = sha256.digest();

BigInteger bigInt = new BigInteger(1, sha256Bytes);//1代表绝对值

return bigInt.toString(16);//转换为16进制

} catch (Exception e) {

e.printStackTrace();

return "";

}

}

}

**五、测试数据及其结果分析**

对算法功能、性能以及健壮性评测所使用的输入数据进行介绍和说明，并给出与输入数据相匹配的算法执行结果，并进行分析。

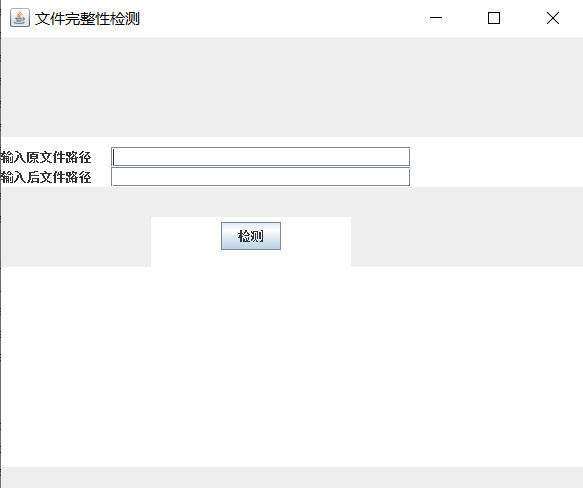
1. 界面如图5-1，用户可输入用于的两个文件的路径用于检测，当用户没有完整输入两个路径时会弹窗提示，提高用户体验

图5程序界面

图6弹窗提示路径不完整

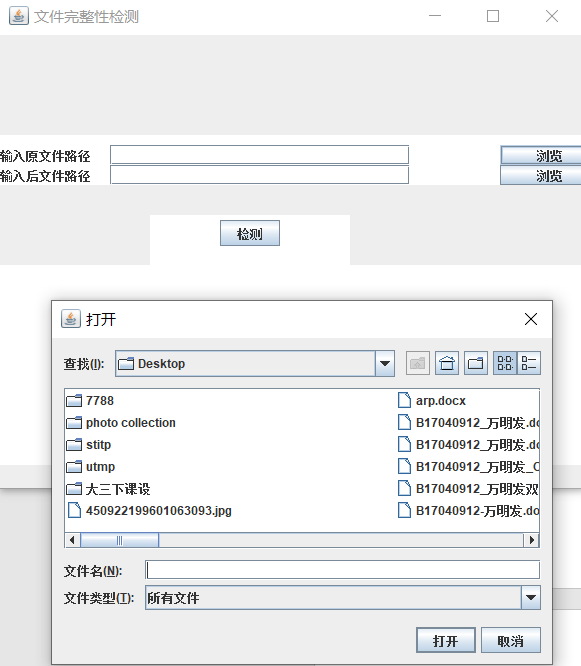
1. 检测的前后两个文件可以通过浏览按键来文件浏览弹窗选择，如图7

图7文件浏览弹窗进行文件选择

1. 如图8所示，我们将桌面的文本文件new.txt复制到一个文件硬盘中的一个文件夹中，这样我们就得到了两份一模一样的文件

图8有两个一样的txt文件

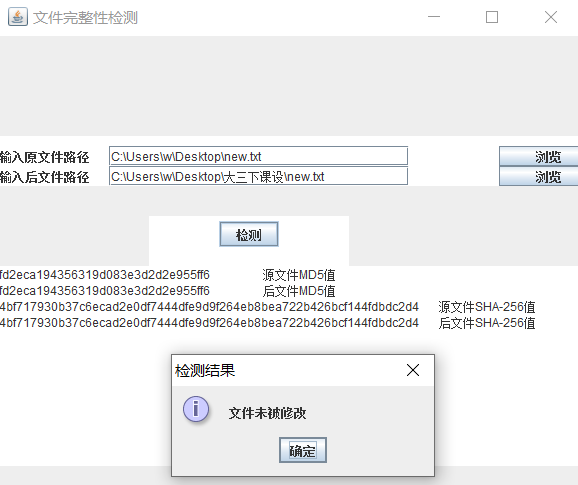
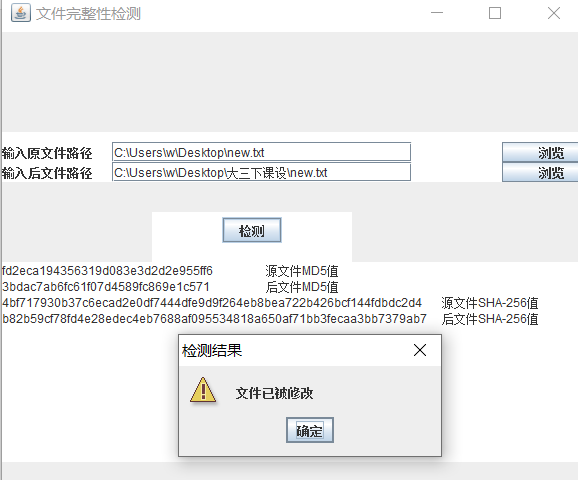
1. 将两个文件路径输入到软件框中进行检测结果如图9

图9监测两个相同内容文档结果

1. 将其中一个文档进行修改，如图10所示，对修改过后的文本再次导入到程序中进行检测，结果如图11所示。

图10修改其中一个文本

图11修改文本后的检测结果

1. 同理，用软件来检测一些被修改的文件，如图5-5所示，桌面中有两个软件，分别是从不同的网站上下载他们的路径分别为

C:\Users\w\Desktop\1806.exe

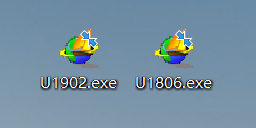
C:\Users\w\Desktop\1902.exe

图12不同的exe软件

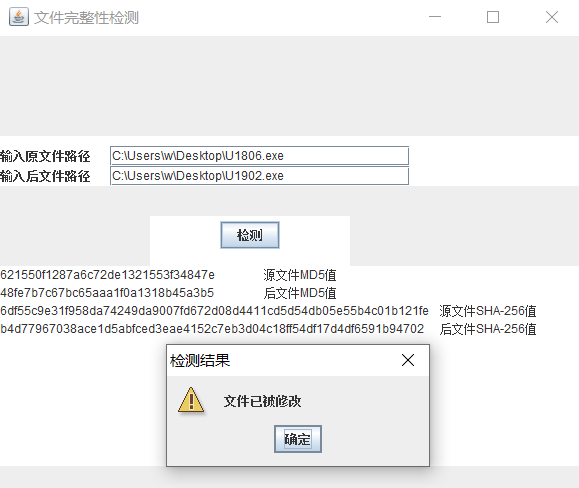
1. 将路径填入软件中用于检测，结果如图13

图13不同exe检测结果

1. 综上所述，该软件通过计算文件的散列值可以实现对不同文件完整性的检测。即到达题目要求中的文件可能因为感染病毒、植入木马/后门、人为篡改或者传输故障等原因导致被篡改，从而能够及时检测

**六、课题完成过程中遇到的问题及解决方法**

1. 没有使用 Java 编写过图形界面

解决方法：为了实现桌面端 GUI，学习了 Java 自带的 GUI 工具包 Swing，提供了各种封装好的对话框、按钮、弹窗实现。

2. 没有接触过 Java 文件读写

解决方法：学习了 Java 的文件读写操作，学习了InputStream和OutputStream两个基于字节流的读写基类

**七、总结**

通过这次的课程设计，我对 Java 更熟悉了，对面向对象的概念也理解更深，第一次尝试开发了桌面端的 GUI 程序，通过查阅 Oracle JDK 文档学习了很多内建的工具类。

除了编程语言，我还对文件篡改相关知识有了一些了解，语言作为工具，帮助我更加深刻的理解了信息安全方面的知识。

对于密码学方面，我更深入的了解了常见的散列算法，了解到 MD5 的缺点，并采用了 SHA-256。

此外，也通过这次程序设计，我查阅了许多资料，对密码学中的杂凑函数有了进一步的了解，例如CRC32、MD5或SHA-1等以及杂凑函数的各种性质有了更深的了解。函数声明由函数返回类型、函数名和形参列表组成。形参列表必须包括形参类型,但是不必对形参命名。这三个元素被称为函数原型,函数原型描述了函数的接口。也正以为这些性质，杂凑函数在信息安全中有着广泛的应用，如数字签名，口令保护，包括此次的完整性检验。

本次程序设计中，本来是用md5散列值来检测文件的完整性的，通过查阅想过资料和电子取证知识知道了md5的碰撞性，所以在本次的实验中特意添加了SHA-256和md5相结合的方法来进行验证，这样可以一定程度上提高检测的准确性。

综上所述，本次程序设计让我对信息安全有了进一步了解。