main11

**本科毕业设计(论文)开题报告**

**题目：基于SSM的网上商场系统**

**课 题 类 型： 设计☑ 实验研究□ 论文□**

**学 生 姓 名： 郑斌**

**学 号： 3160704326**

**专 业 班 级： 软件工程163班**

**学 院： 计算机与信息学院**

**指 导 教 师： 邹 姗**

**开 题 时 间：**

1. **0 年 月 日**

一、本课题的研究意义、研究现状和发展趋势（文献综述）

1.1研究意义

随着电子商务的发展，无店铺零售悄然兴起，它是商业经济发展到一定阶段的产物，也是顺应新的消费需求和技术进步的必然结果。无店铺零售则大多不存在地段方面的制约。二是经营成本。借助日益发展的科学技术，无店铺零售通常能够省却从生产到销售过程中的许多环节，从而达到有效降低经营成本与交易成本，带来消费者、商家双赢的局面[2]。

鉴于互联网的优势以及对国内外相关电子商务网站现状的研究分析，本设计以基于Web的商城电子商务网站开发作为我的毕业设计主题。立足于设计一个在网络平台上运行的商城，以小小的力量推动中国互联网的发展。

1.2研究现状

随着计算机技术的发展和网络人口的增加，网络世界也越来越广博，越来越丰富，电子商务已经成为网上的一股潮流。相信要不了太长有时间，顾客就可以在网络世界上获得他们在现实世界上可以获得的所有商品和服务。

以下是根据CNNIC（中国互联网络信息中心）公布的中国B2C电子商务发展报告来进一步分析目前的网上购物的现状。

90%选择网上购物的人都认为网上购物节约时间和操作方便，他们分别占46.7%和44.2%，这说明随着生活节奏的加快，人们越来越希望拥有简单快捷的购物方式网上购物已经逐渐成为一种购物趋势，越来越多的人选择网上购物[8]。

消费者对消费要求的日益提高，使得网上商城必须向着专业化迈进，通过提供专业的产品和服务，区分市场。以卓越网和铛铛网为例，日前针对不同的消费者和消费需求，开通了计算机、外语、文学和艺术四个专业商城，这不仅体现了卓越网“最以客户为中心”的经验理念，相信也是中国网上商城未来发展的趋势。

尽管网络版权、阅读习惯仍是制约未来网络出版、网上商城发展的障碍，但网络环境给商品出版业带来深刻的革命，已成大势所趋。网上商城和传统店铺都在继续争取商品发行新的利润空间。[6]目前的情况是，商品发行业对网络的利用并不充分，只刚刚涉足电子商务和网站建设，对网络的其他产品，以及网络及时、一对一、跨时间、跨空间等特性，并没有充分加以利用，可以预计，随着网络进一步融入人们的生活，以技术、服务为品牌支撑的网上商城将成为市场的宠儿和业态创新的引领者。

1.2.1文献调研

文献[3]基于模型驱动的Web应用服务系统开发理论框架

文献[3]主要介绍了从软件设计实践出发,针对基于Web应用服务和设计模式的软件设计方法,提出了一种新的模型驱动开发理论框架.该框架以类型范畴理论为基础,以服务构件标识为范畴对象,以服务构件之间的交互和依赖关系为范畴态射,从而用范畴图表给出了软件体系结构模型的框架描述.设计模式看成是构件规范的模板,模板的实例化和规范的集成过程就是软件设计和实现的过程,从而建立起了模型转换和精化中特性描述和一致性研究的统一框架.应用研究表明,该框架符合模型驱动软件设计的本质要求,为模型转换和模型驱动的Web应用服务开发提供了新的认知、分析和研究的指导架构.

文献[5]基于敏捷方法的轻量级J2EE架构的应用

文献[5]主要提出Sfruls+Spring+Hibernate三个开源框架的松耦合层次结构,加之Tomcat开源服务器组成一个轻量级的J2EE架构。文章结合敏捷软件开发方法说明了轻量级J2EE架构设计实现的过程,并针对目标系统彩信短信管理平台设计了Struts+Spring+Hibernate的系统概念模型。

文献[9]基于Bootstrap+spring boot框架的在线考试系统开发

文献[9]主要介绍了该在线考试系统设计使用Bootstrap+spring boot框架,实现用户管理,题库管理、成绩管理、考试自动组合出卷与评卷管理等功能。该框架能够减少底层开发工作量,系统模块之间具有高内聚、低耦合的特点,进而提升系统的可用性和可维护性。本系统能满足各科课程教学考试考核需求,简化并改善传统纸质考试模式,提高考试效率与学院信息化建设。

文献[11]基于SSM的软件体系结构开发过程研究

文献[11]主要介绍了软件体系结构开发过程和软件体系结构产品是软件体系结构"时空"的外在表现,软件体系结构的开发过程直接影响着软件体系结构的质量;在基于SSM的SA开发过程循环图的指导下,文中系统分析了体系结构相关的设计、测试、评估等活动之间的联系,并设计出一种新的体系结构开发过程模型——"多三角"体系结构开发过程模型,充分体现出尽早测评、全面测评、全过程测评、独立的迭代测评的特点,从而以合理的过程指导优质体系结构的设计。

文献[12]基于SSH框架与AJAX技术的JavaWeb应用开发

文献[12]主要采用J2EE框架技术设计了一个高校教育资源库管理系统,并从表示层、数据层和业务层分别探讨了系统实现的关键技术。系统Web页面的设计采用AJAX技术,数据持久层的设计通过Spring与Hibernate的集成完成,业务层与表示层的设计通过Spring与Struts的集成完成。实验结果表明,SSH框架的使用不仅简化了系统的开发过程,而且提高了系统的可扩展性和可维护性。同时,AJAX技术的使用,也极大地提高了Web页面的访问效率。

1.3发展趋势

随着信息技术广泛、深入地应用到人类社会的各个领域并发挥着越来越重要的作用。[7]计算机软件技术应用于信息管理是发展的必然。作为计算机应用的一部分，充分发挥计算机的优势，将大量复杂的数据交给计算机来处理，有着手工管理所无法比拟的优点。如：查找方便、可靠性高、存储量大、保密性好、寿命长、成本低等。这些优点能够极大地提高信息管理的效率，是真正意义上的合理利用资源，也是企事业管理科学化、正规化,与世界接轨的重要条件。

二、主要设计（研究）内容

作为一个网上商城系统,就应该做到能提供强大的业务支持功能，系统能实现用户的注册功能、登录功能、商品的查询，订购等功能。该系统基本上具备一个商品销售网站应该具备的功能，该设计项目基本上体现了构建一个动态商务网站所需要的技术。

经过前期的深入调查和研究，总结出该平台需要完成的一些具体功能，分析如下：

（1）前台用户功能模块主要是实现了用户对公告、商品的浏览以及查询，在注册登录后，可以实现物品加入购物车和购买等功能。

（2）后台管理模块主要实现了管理员对公告的修改，商品的添加删除，用户的订单管理等功能

具体工作如下：

（1）需求方面：

具体分析网上商城主要的问题所在。

根据网上商城的相关问题，查阅资料总结网上商城系统的主要需求。

根据网上商城的主要需求，设计解决方案。并选择最优的解决方案。

（2）系统方面：

选择合适的开发工具以及编程语言与编程框架。

根据相关解决方案，建立对应的数据字典和数据库结构。

进行系统的编码，后端编码是要进行分层处理，分为pojo层，dao层，service层与controller层。

编码实现系统的具体功能

进行前端网页编码，其网页展现要简单，友好，

将前后端进行整合，并测试该系统网页是否正常运行。

（3）调试系统：

测试该系统是否完成任务书中的要求都功能。

测试并完善系统的友好化程度。

将整个过程总结，形成论文。

**3.研究方案及工作计划**

3.1工作重难点

1. 如何建立相关的数据字典与数据库
2. 后端技术开发及代码编写
3. 对网上商城的需求分析

3.2解决上述重难点问题拟采用方案

1. 在需求的基础上，查阅相关文献资料，设计出相关解决方案。并对解决方案进行可行性分析，找出最优的解决方案。通过最优的解决方案，设计数据字典与数据库，确定主键和外键，使数据库的设计符合第三范式。
2. 后端技术开发代码编写：后端决定采用ssm框架，SSM（Spring+SpringMVC+MyBatis）框架集由Spring、MyBatis两个开源框架整合而成（SpringMVC是Spring中的部分内容）。常作为数据源较简单的web项目的框架。此框架主要作用是页面发送请求给控制器，控制器调用业务层处理逻辑，逻辑层向持久层发送请求，持久层与数据库交互，后将结果返回给业务层，业务层将处理逻辑发送给控制器，控制器再调用视图展现数据。
3. 根据任务书的要求查阅相关文献资料，了解传统网上商场中的主要问题。通过对问题的分析，找到那些是主要问题，那些是次要问题。并对此找到网上商城的相关需求。

**3.3工作计划**

（1）课题调研：

（2）准备开题报告： 2020.2.10--2020.2.22

（3）系统分析阶段： 收集相关资料，调查研究 2020.2.23--2020.3.10

对课题进行可行性分析，完成相关文档 2020.3.10--2020.3.14

（4）系统设计阶段：概要设计： 2020.3.15--2020.3.25

详细设计： 2020.3.26--2020.4.6

（5）系统实现测试阶段：

代码编写： 2020.4.6--2020.4.25

系统测试调整：2020.4.26--2020.5.9

（）完成论文撰写： 2020.5.10--2020.5.23

**4.主要参考文献**

[1]陈海汝,何青,潘轩平,张大宇,胡铁平.基于云服务器的SSM框架后台搭建与实现[J].信息系统工程,2019(11):114-115+117.

[2]蔡呈杰,王贵鑫.基于SSM框架的学子商城系统的设计与实现[J].科学技术创新,2019(32):69-71.

[3]侯金奎,鹿旭东,陈春雷,王磊.基于模型驱动的Web应用服务系统开发理论框架[J].小型微型计算机系统,2018,39(10):2345-2352.

[4]鞠宏军,佘春燕.基于SSM的高并发慕课网的设计与实现[J].电脑知识与技术,2020,16(02):54-56+90.

[5]戚琦,廖建新,王纯,武家春.基于敏捷方法的轻量级J2EE架构的应用[J].计算机系统应用,2007(02):53-56.

[6]王钱,王蓉,张利.基于iBatis的通用数据持久层的研究与设计[J].微计算机信息,2007(12):172-173+128.

[7]王艳清,陈红.基于SSM框架的智能web系统研发设计[J].计算机工程与设计,2012,33(12):4751-4757.

[8]徐刚,翟梦娇.基于SSM的美容资讯商务网站的设计与实现[J].商丘职业技术学院学报,2019,18(06):65-71.

[9]叶惠仙,沈文杰.基于Bootstrap+spring boot框架的在线考试系统开发[J].网络安全技术与应用,2019(12):54-57.

[10]张文龙,吴林辉,杨晨耀,蒋卫祥.基于SSM框架+vue的Web网盘系统的设计与实现[J].电脑知识与技术,2019,15(34):62-63+65.

[11]张俊萍,朱小冬,侯娜,张鲁,梁欣.基于SSM的软件体系结构开发过程研究[J].计算机测量与控制,2011,19(08):2029-2032.

[12]谌湘倩,狄文辉,孙冬.基于SSH框架与AJAX技术的JavaWeb应用开发[J].计算机工程与设计,2009,30(10):2590-2592+2596.

[13]朱运乔.基于SpringBoot+SSM框架的Web应用系统搭建与实现[J].电脑编程技巧与维护,2019(10):23-25.

[14]Abdellatif, T. and F. Boyer. A node allocation system for deploying JavaEE systems on Grids. 2009. Hammemet, Tunisia.

[15].Li, Z. and Z. Weixi. Design of tourism e-business system based on JavaEE multi-pattern. 2012. Sanya, China.

**附录**

**外文文献：**

**Core Java™ Volume II–Advanced Features**

When Java technology first appeared on the scene, the excitement was not about a well-crafted programming language but about the possibility of safely executing applets that are delivered over the Internet (see Volume I, Chapter 10 for more information about applets). Obviously, delivering executable applets is practical only when the recipients are sure that the code can't wreak havoc on their machines. For this reason, security was and is a major concern of both the designers and the users of Java technology. This means that unlike other languages and systems, where security was implemented as an afterthought or a reaction to break-ins, security mechanisms are an integral part of Java technology.

Three mechanisms help ensure safety:

• Language design features (bounds checking on arrays, no unchecked type conversions, no pointer arithmetic, and so on).

• An access control mechanism that controls what the code can do (such as file access, network access, and so on).

• Code signing, whereby code authors can use standard cryptographic algorithms to authenticate Java code. Then, the users of the code can determine exactly who created the code and whether the code has been altered after it was signed.

Below, you'll see the cryptographic algorithms supplied in the java.security package, which allow for code signing and user authentication.

As we said earlier, applets were what started the craze over the Java platform. In practice, people discovered that although they could write animated applets like the famous "nervous text" applet, applets could not do a whole lot of useful stuff in the JDK 1.0 security model. For example, because applets under JDK 1.0 were so closely supervised, they couldn't do much good on a corporate intranet, even though relatively little risk attaches to executing an applet from your company's secure intranet. It quickly became clear to Sun that for applets to become truly useful, it was important for users to be able to assign different levels of security, depending on where the applet originated. If an applet comes from a trusted supplier and it has not been tampered with, the user of that applet can then decide whether to give the applet more privileges.

To give more trust to an applet, we need to know two things:

• Where did the applet come from?

• Was the code corrupted in transit?

In the past 50 years, mathematicians and computer scientists have developed sophisticated algorithms for ensuring the integrity of data and for electronic signatures. The java.security package contains implementations of many of these algorithms. Fortunately, you don't need to understand the underlying mathematics to use the algorithms in the java.security package. In the next sections, we show you how message digests can detect changes in data files and how digital signatures can prove the identity of the signer.

A message digest is a digital fingerprint of a block of data. For example, the so-called SHA1 (secure hash algorithm #1) condenses any data block, no matter how long, into a sequence of 160 bits (20 bytes). As with real fingerprints, one hopes that no two messages have the same SHA1 fingerprint. Of course, that cannot be true—there are only 2160 SHA1 fingerprints, so there must be some messages with the same fingerprint. But 2160 is so large that the probability of duplication occurring is negligible. How negligible? According to James Walsh in True Odds: How Risks Affect Your Everyday Life (Merritt Publishing 1996), the chance that you will die from being struck by lightning is about one in 30,000. Now, think of nine other people, for example, your nine least favorite managers or professors. The chance that you and all of them will die from lightning strikes is higher than that of a forged message having the same SHA1 fingerprint as the original. (Of course, more than ten people, none of whom you are likely to know, will die from lightning strikes. However, we are talking about the far slimmer chance that your particular choice of people will be wiped out.)

A message digest has two essential properties:

• If one bit or several bits of the data are changed, then the message digest also changes.

• A forger who is in possession of a given message cannot construct a fake message that has the same message digest as the original.

The second property is again a matter of probabilities, of course. Consider the following message by the billionaire father:"Upon my death, my property shall be divided equally among my children; however, my son George shall receive nothing."

That message has an SHA1 fingerprint of

2D 8B 35 F3 BF 49 CD B1 94 04 E0 66 21 2B 5E 57 70 49 E1 7E

The distrustful father has deposited the message with one attorney and the fingerprint with another. Now, suppose George can bribe the lawyer holding the message. He wants to change the message so that Bill gets nothing. Of course, that changes the fingerprint to a completely different bit pattern:

2A 33 0B 4B B3 FE CC 1C 9D 5C 01 A7 09 51 0B 49 AC 8F 98 92

Can George find some other wording that matches the fingerprint? If he had been the proud owner of a billion computers from the time the Earth was formed, each computing a million messages a second, he would not yet have found a message he could substitute.

A number of algorithms have been designed to compute these message digests. The two best-known are SHA1, the secure hash algorithm developed by the National Institute of Standards and Technology, and MD5, an algorithm invented by Ronald Rivest of MIT. Both algorithms scramble the bits of a message in ingenious ways. For details about these algorithms, see, for example, Cryptography and Network Security, 4th ed., by William Stallings (Prentice Hall 2005). Note that recently, subtle regularities have been discovered in both algorithms. At this point, most cryptographers recommend avoiding MD5 and using SHA1 until a stronger alternative becomes available. (See http://www.rsa.com/rsalabs/node.asp?id=2834 for more information.)

The Java programming language implements both SHA1 and MD5. The MessageDigest class is a factory for creating objects that encapsulate the fingerprinting algorithms. It has a static method, called getInstance, that returns an object of a class that extends the MessageDigest class. This means the MessageDigest class serves double duty:

• As a factory class

• As the superclass for all message digest algorithms

For example, here is how you obtain an object that can compute SHA fingerprints:

MessageDigest alg = MessageDigest.getInstance("SHA-1");

(To get an object that can compute MD5, use the string "MD5" as the argument to getInstance.)

After you have obtained a MessageDigest object, you feed it all the bytes in the message by repeatedly calling the update method. For example, the following code passes all bytes in a file to the alg object just created to do the fingerprinting：

InputStream in = . . .

int ch;

while ((ch = in.read()) != -1)

alg.update((byte) ch);

Alternatively, if you have the bytes in an array, you can update the entire array at once:

byte[] bytes = . . .;

alg.update(bytes);

When you are done, call the digest method. This method pads the input—as required by the fingerprinting algorithm—does the computation, and returns the digest as an array of bytes.

byte[] hash = alg.digest();

The program in Listing 9-15 computes a message digest, using either SHA or MD5. You can load the data to be digested from a file, or you can type a message in the text area.

Message Signing

In the last section, you saw how to compute a message digest, a fingerprint for the original message. If the message is altered, then the fingerprint of the altered message will not match the fingerprint of the original. If the message and its fingerprint are delivered separately, then the recipient can check whether the message has been tampered with. However, if both the message and the fingerprint were intercepted, it is an easy matter to modify the message and then recompute the fingerprint. After all, the message digest algorithms are publicly known, and they don't require secret keys. In that case, the recipient of the forged message and the recomputed fingerprint would never know that the message has been altered. Digital signatures solve this problem.

To help you understand how digital signatures work, we explain a few concepts from the field called public key cryptography. Public key cryptography is based on the notion of a public key and private key. The idea is that you tell everyone in the world your public key. However, only you hold the private key, and it is important that you safeguard it and don't release it to anyone else. The keys are matched by mathematical relationships, but the exact nature of these relationships is not important for us. (If you are interested, you can look it up in The Handbook of Applied Cryptography at http://www.cacr.math.uwaterloo.ca/hac/.)

The keys are quite long and complex. For example, here is a matching pair of public and private Digital Signature Algorithm (DSA) keys.

Public key:

Code View:

p:

fca682ce8e12caba26efccf7110e526db078b05edecbcd1eb4a208f3ae1617ae01f35b91a47e6df63413c5e12ed0899bcd132acd50d99151bdc43ee737592e17

q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5

g:678471b27a9cf44ee91a49c5147db1a9aaf244f05a434d6486931d2d14271b9e35030b71fd73da179069b32e2935630e1c2062354d0da20a6c416e50be794ca4

y:

c0b6e67b4ac098eb1a32c5f8c4c1f0e7e6fb9d832532e27d0bdab9ca2d2a8123ce5a8018b8161a760480fadd040b927281ddb22cb9bc4df596d7de4d1b977d50

Private key:

Code View:

p:

fca682ce8e12caba26efccf7110e526db078b05edecbcd1eb4a208f3ae1617ae01f35b91a47e6df63413c5e12ed0899bcd132acd50d99151bdc43ee737592e17

q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5

g:

678471b27a9cf44ee91a49c5147db1a9aaf244f05a434d6486931d2d14271b9e35030b71fd73da179069b32e2935630e1c2062354d0da20a6c416e50be794ca4

x: 146c09f881656cc6c51f27ea6c3a91b85ed1d70a

It is believed to be practically impossible to compute one key from the other. That is, even though everyone knows your public key, they can't compute your private key in your lifetime, no matter how many computing resources they have available.

It might seem difficult to believe that nobody can compute the private key from the public keys, but nobody has ever found an algorithm to do this for the encryption algorithms that are in common use today. If the keys are sufficiently long, brute force—simply trying all possible keys—would require more computers than can be built from all the atoms in the solar system, crunching away for thousands of years. Of course, it is possible that someone could come up with algorithms for computing keys that are much more clever than brute force. For example, the RSA algorithm (the encryption algorithm invented by Rivest, Shamir, and Adleman) depends on the difficulty of factoring large numbers. For the last 20 years, many of the best mathematicians have tried to come up with good factoring algorithms, but so far with no success. For that reason, most cryptographers believe that keys with a "modulus" of 2,000 bits or more are currently completely safe from any attack. DSA is believed to be similarly secure.

Figure 9-12 illustrates how the process works in practice.

Suppose Alice wants to send Bob a message, and Bob wants to know this message came from Alice and not an impostor. Alice writes the message and then signs the message digest with her private key. Bob gets a copy of her public key. Bob then applies the public key to verify the signature. If the verification passes, then Bob can be assured of two facts:

• The original message has not been altered.

• The message was signed by Alice, the holder of the private key that matches the public key that Bob used for verification.

You can see why security for private keys is all-important. If someone steals Alice's private key or if a government can require her to turn it over, then she is in trouble. The thief or a government agent can impersonate her by sending messages, money transfer instructions, and so on, that others will believe came from Alice.

The X.509 Certificate Format

To take advantage of public key cryptography, the public keys must be distributed. One of the most common distribution formats is called X.509. Certificates in the X.509 format are widely used by VeriSign, Microsoft, Netscape, and many other companies, for signing e-mail messages, authenticating program code, and certifying many other kinds of data. The X.509 standard is part of the X.500 series of recommendations for a directory service by the international telephone standards body, the CCITT.

The precise structure of X.509 certificates is described in a formal notation, called "abstract syntax notation #1" or ASN.1. Figure 9-13 shows the ASN.1 definition of version 3 of the X.509 format. The exact syntax is not important for us, but, as you can see, ASN.1 gives a precise definition of the structure of a certificate file. The basic encoding rules, or BER, and a variation, called distinguished encoding rules (DER) describe precisely how to save this structure in a binary file. That is, BER and DER describe how to encode integers, character strings, bit strings, and constructs such as SEQUENCE, CHOICE, and OPTIONAL.

**中文译文：**

**Java核心技术 卷Ⅱ高级特性**

当Java技术刚刚问世时，令人激动的并不是因为它是一个设计完美的编程语言，而是因为它能够安全地运行通过因特网传播的各种applet。很显然，只有当用户确信applet的代码不会破坏他的计算机时，用户才会接受在网上传播的可执行的applet。正因为如此，无论过去还是现在，安全都是设计人员和Java技术使用者所关心的一个重大问题。这就意味着，Java技术与其他的语言和系统有所不同，在那些语言和系统中安全是事后才想到要去实现的，或者仅仅是对破坏的一种应对措施，而对Java技术来说，安全机制是一个不可分割的组成部分。

Java技术提供了以下三种确保安全的机制：

(1)语言设计特性（对数组的边界进行检查，无不检查类型的转换，无指针算法等）。

(2)访问控制机制，用于控制代码能够执行的功能（比如文件访问，网络访问等）。

(3) 代码签名，利用该特性，代码的作者就能够用标准的加密算法来表明Java代码的身份。这样，该代码的使用者就能够准确地知道谁创建了该代码，以及代码被标识后是否被修改过。

下面，我们要介绍java.security包提供的加密算法，用来进行代码的标识和用户身份认证。

正如我们前面所说，applet 是在Java平台上开始流行起来的。实际上，人们发现尽管他们可以编写像著名的“nervous text”那样栩栩如生的applet，但是在JDK1.0安全模式下无法发挥其一整套非常有用的作用。例如，由于JDK1.0下的applet要受到严密的监督，因此，即使applet在公司安全内部网上运行时的风险相对较小，applet也无法在企业内部网上发挥很大的作用。Sun公司很快就认识到，要使applet真正变得非常有用，用户必须可以根据applet的来源为其分配不同的安全级别。如果applet来自值得信赖的提供商，并且没有被篡改过，那么applet的用户就可以决定是否给applet授予更多的运行特权。

如果要给予applet更多的信赖，你必须知道下面两件事：

(1)applet来自哪里？

(2)在传输过程中代码是否被破坏？

在过去的50年里，数学家和技术机科学家已经开发出各种各样成熟的算法，用于确保数据和电子签名的完整性，在java.security包中包含了许多这些算法的实现。在下面几节，我们将要介绍消息摘要是如何检测数据文件中的变化的，以及数字签名是如何证明签名者的身份的。

消息摘要是数据块的数字指纹。例如，所谓的SHA1（安全散列算法#1）可将任何数据块，无论其数据有多长，都压缩为160位（20字节）的序列。与真实的指纹一样，人们希望任何两条消息都不会有相同的SHA1指纹。当然这是不可能的—因为只存在2160 个SHA1指纹，所有肯定会有某些消息具有相同的指纹。因为2160 是一个很大的数字，所以存在重复指纹的可能性微乎其微，那么这种重复的可能性到底小到什么程度呢？根据James Walsh在他的《True Odds:How Risks Affect Your Everyday Life》,Merritt Publishing出版社1996年出版，一书中所阐述的，你和他们所有的人都死于雷击的概率，比伪造的消息与原来消息具有相同的SHA1指纹的概率还要高。（当然，可能有你不认识的其他10个以上的人会死于雷击，但这里我们讨论的是你选择的特定的人的死亡概率）。

消息摘要具有两个基本属性：

(1)如果数据的1位或者几位改变了，那么消息摘要也将改变。

(2）拥有给定消息的伪造者不能创建与原消息具有相同摘要的假消息。

当然，第二个属性又是一个概率问题。让我们来看看下面这位亿万富翁下的遗嘱：“我死了之后，我的财产将由我的孩子平分，但是，我的儿子George应该拿不到一个子。”

这份遗嘱的SHA1指纹为：

2D 8B 35 F3 BF 49 CD B1 94 04 E0 66 21 2B 5E 57 70 49 E1 7E

这位有疑心病的父亲将这份遗嘱交给一位律师保存，而将指纹交给另一位律师保存。现在，假设George能够贿赂那位保存遗嘱的律师，他想修改这份遗嘱，使得Bill一无所得。当然，这需要将原指纹改为下面这样完全不同的位模式：

2A 33 0B 4B B3 FE CC 1C 9D 5C 01 A7 09 51 0B 49 AC 8F 98 92

那么George能够找到与该指纹相匹配的其他文字吗？如果从地球形成之时，他就很自豪地拥有10亿台计算机，每台计算机每秒钟处理一百万条信息，他依然无法找到一个能够替换的遗嘱。

人们已经设计出大量的算法，用于计算这些消息摘要，其中最著名的两种算法是SHAI和MD5。SHAI是由美国国家标准和技术学会开发的加密散列算法，MD5是由麻省理工学院的Ronald Rivest发明的算法。这两种算法都使用了独特巧妙的方法对消息中的各个位进行扰乱。如果要了解这些方法的详细信息，请参阅William Stallings撰写的《Cryptography and Network Security》一书，该书由Prentice Hall出版社于2005年出版口值得注意的是，最近人们在这两种算法中发现了某些微妙的规律性，因此许多密码人员建议最好避免使用MD5，而应该使用SHA1算法，直到有更强的加密算法出现。（查看http://www.rsa.com/rsalabs/node.asp?id=2834以了解更多的信息）。

Java编程语言已经实现了SHA1和MD5。MessageDigest类是用于创建封装了指纹算法的对象的“工厂”，它的静态方法getInstance返回继承了MessageDigest类的某个类的对象。这意味着MessageDigest类能够承担下面的双重职责：

（1）作为一个工厂类。

（2）作为所有消息摘要算法的超类。

例如，下面是如何获取一个能够计算SHA指纹的对象的方法：

MessageDigest alg = MessageDigest.getInstance(“SHA-1”)；

（如果要获取计算MD5的对象，请使用字符串“MD5”作为getInstance的参数。）

当你已经获取MessageDigest对象之后，通过反复调用update方法，将信息中的所有字节提供给该对象。例如，下面的代码将文件中的所有字节传给上面建立的alg对象，以执行指纹算法：

InputStream in=….

int ch；

while((ch=in.read())!=-1)

alg.updat((byte) ch)；

另外，如果这些字节存放在一个数组中，那就可以一次完成整个数组的更新：

byte[] bytes =．．．；

alg.update(bytes)；

当完成上述操作后，调用digest方法。该方法填充输入信息—指纹算法需要的—并且进行相应的计算，然后以字节数组的形式返回消息摘要。

byte[] hash=alg.digest()；

程序清单9-15中的程序计算了一个消息摘要，既可以用SHA，也可以使用MD5来计算。可以从文件加载需要计算摘要的数据，也可以直接将信息输入文本区域。图9-11显示了该应用程序的画面。

消息签名

在上一节中，我们介绍了如何计算原始消息的消息摘要和指纹的方法。如果消息改变了，那么改变后的消息的指纹与原消息的指纹将不匹配。如果消息和它的指纹是分开传送的，那么接收者就可以检查消息是否被篡改过。但是，如果消息和指纹同时被截获了，对消息进行修改，再重新计算指纹，这是一件很容易的事情。毕竟，消息摘要算法是公开的，不需要使用任何密钥。在这种情况下，假消息和新指纹的接收者永远不会知道消息已经被篡改。数字签名解决了这个问题。

为了了解数字签名的工作原理，我们需要解释关于公共密钥加密技术领域中的几个概念。公共密钥加密技术是基于公共密钥和私有密钥这个两个基本概念的。它的设计思想是你可以将公共密钥告诉世界上的任何人，但是，只有自己才拥有私有密钥，重要的是你要保护你的私有密钥，不将它泄漏给其他任何入。这些密钥之间存在一定的数学关系，但是这种关系的具体性质对于实际的编程来说并不重要(如果你有兴趣，可以参阅http://www.cacr.math.uwaterloo.ca/hac/站点上的《The Handbook of Applied Cryptography》 一书)。

密钥非常长，而且很复杂。例如，下面是一对匹配的数字签名算法(DSA)公共密钥和私有密钥。

公共密钥：

p: fca682ce8e12caba26efccf7ll0e526db078b05e6ecbcdleb4a208f3ae1617ae0lf35b9la47e6df63413c5e12ed0899bcd132acd50d9915lbdc43ee737592el7

q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5

g:67847lb27a9cf44ee9la49c5147dbla9aaf244f05a434d648693ld2d1427lb9e35030b7lfd73da179069b32e2935630elc2062354d0da20a6c416e50be794ca4

y：

c0b6e67b4ac098ebla32c5f8c4clfee7e6fb9d832532e27d0bdab9ca2d2a8123ce5a8018b816la6048efadd040b927281ddb22cb9bc4df596d7de4dlb977dS0

私有密钥：

p：

fca682ce8e12caba26efccf7ll0e526db078b05edecbcdleb4a208f3ae1617ae0lf35b9la47e6df63413c5e12ed0899bcd132acd50d9915lbdc43ee737592e17

q: 962eddcc369cba8ebb260ee6b6a126d9346e38c5

g：

67847lb27a9cf44ee9la49c5147dbla9aaf244f05a434d648693ld2d1427lb9e35030b7lfd73da179069b32e2935630elc2062354d0da20a6c416e50be794ca4

x: 146c09f881656cc6c5lf27ea6c3a9lb85edld70a

在现实中，几乎不可能用一个密钥去推算出另一个密钥。也就是说，即使每个人都知道你的公共密钥，不管他们拥有多少计算资源，他们一辈子也无法计算出你的私有密钥。

任何人都无法根据公共密钥来推算私有密钥，这似乎让人难以置信。但是时至今日，还没有人能够找到一种算法，来为现在常用的加密算法进行这种推算。如果密钥足够长，那么要是使用穷举法—也就是直按试验所有可能的密钥—所需要的计算机将比用太阳系中的所有原子来制造的计算机还要多，而且还得花费数千年的时间。当然，可能会有人提出比穷举更灵活的计算密钥的算法。例如，RSA算法（该加密算法由Rivest, Shamir和Adleman发明）就利用了对数值巨大的数字进行因子分解的困难性。在最近20年里，许多优秀的数学家都在尝试提出好的因子分解算法，但是迄今为止都没有成功。据此，大多数密码学者认为，拥有2000位或者更多位“模数”的密钥目前是完全安全的，可以抵御任何攻击。DSA被认为具有类似的安全性。

图9-12展示了这项工作的处理过程。

假设Alice想要给Bob发送一个消息，Bob想知道该消息是否来自Alice，而不是冒名顶替者。Alice写好了消息，并且用她的私有密钥对该消息摘要签名。 Bob得到了她的公共密钥的拷贝，然后Bob用公共密钥对该签名进行校验。如果通过了校验，则Bob可以确认以下两个事实：

（1)原始消息没有被篡改过。

（2)该消息是由Alice签名的，她是私有密钥的持有者，该私有密钥就是Bob

与她用于校验的公共密钥相匹配的密钥。

你可以看到私有密钥的安全性为什么是最重要的。如果某个人偷了Alice的私有密钥，或者政府要求她交出私有密钥，那么她就麻烦了。小偷或者政府代表就可以假扮她的身份来发送消息和资金转账指令等等，而其他人则会相信这些消息确实来自于Alice。

X.509证书格式

为了利用公共密钥这种密码系统，必须将公共密钥分发出去。最通用的一种签名证书格式称为X.509格式。X.509格式的证书被VeriSign、微软、网景和其他许多公司广泛应用于对电子邮件消息进行签名，对程序代码进行认证，以及对许多其他类型的数据进行认证等等。 X.509标准是由国际电话标准机构，即国际电报电话咨询委员会(CCITT)提出的用于目录服务的X.500系列建议的组成部分。

X.509证书的具体结构是用一种形式化表示来描述的，称为“抽象语法表示法#1”(abstract syntax notation)即ASN.1。图9-13显示了第3版X.509格式的ASN.1定义。虽然具体的语法对我们并不重要，但是你可以看到，ASN.1为证书文件的结构给出了精确的定义。“基本编码规则”(basic encoding rules)，即BER，精确地描述了如何将该结构保存为二迸制文件。也就是说，BER描述了如何对整数、字符串、位串以及诸如SEQUENCE、CHOICE和OPTIONAL的结构进行编码的方法。

**五、指导教师意见（签名）**

**2020 年 月 日**