网络与信息安全技术丛书

Sun microsystems

Inside Java 2 Platform Security

Architecture, API Design, and Implementation

Java 2 平台安全技术

——结构、API设计和实现

(美) Li Gong 著

王运凯 石磊 曹少文 康勇 译

机械工业出版社
China Machine Press

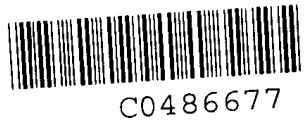
网络与信息安全技术丛书

Java 2平台安全技术

——结构、API设计和实现

(美) Li Gong 著
王运凯 石磊 曹少文 康勇 译
申波 李莹 校

机械工业出版社
China Machine Press



本书明确而详尽地阐述了Java平台安全性，探究了Java安全结构的内幕。

本书首先概述了计算机和网络安全概念并解释了Java安全模型，并在此基础上，详细描述了Java 2平台中新增加的许多安全结构方面的措施，同时对Java安全性的实施提出了使用指导，描绘了如何定制、扩展和精化安全结构以及成功实现的技术细节。

本书为建立安全、有效、强大和可移植的Java应用程序和applet提供了重要的信息，对于致力于研究Java平台的专业人员是一本必不可少的参考书。

本书版权登记号：图字：01-1999-3471

图书在版编目(CIP)数据

Java 2平台安全技术：结构、API设计和实现 /（美）李宫(Li Gong)著；王运凯等译. -北京：机械工业出版社，2000.1

(网络与信息安全技术丛书)

书名原文：Inside Java 2 Platform Security: Architecture, API Design, and Implementation

ISBN 7-111-07807-1

Ⅰ.J… Ⅱ.①李… ②王… Ⅲ.JAVA语言-程序设计-安全技术 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999)第57812号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：陈 谊

北京忠信诚胶印厂印刷 · 新华书店北京发行所发行

2000年2月第1版第1次印刷

787mm × 1092mm 1/16 · 10.25印张

印数：0 001-5 000册

定价：19.00元

前　言

给我一个杠杆和支点，我可以移动整个地球。

——阿基米德

自引入Java技术以来，尤其是1995年春季它公开露面之后，Java平台安全性引起人们越来越多的注意，同时Java使用中的新的安全问题也暴露出来。在计算机发展史上，关注安全性是一种新现象。许多新的计算机技术最初往往忽视安全问题，事后也不采取更加安全的措施。众所周知，重新配置安全性非常困难，即使可能，也常常会引起向后兼容性问题。因此，虽然人们在安全性方面做了许多尝试，但都没有获得很大的成功。

很幸运，Java技术出现在Internet舞台上，系统安全性成为一个重要的设计目标。最初的安全模型虽然非常简单，但它是一个伟大的起点，正如阿基米德所说，它是一个支点，而JavaSoft的天才工程师和强大的管理队伍是杠杆，正是由于两者的密切结合，使Java的安全体系结构变为现实。

Java技术的开发人员有两个初衷：一方面以JDK(Java Development Kit)方式为Java应用程序的运行提供一个安全平台。另一方面，通过Java编程语言提供一系列安全工具和服务，有利于更广泛的与安全性有关的应用程序的实现，如企业界的安全应用程序。

通过本书，我期望达到以下效果。首先，让读者对系统和网络安全有一个简洁而清晰的理解，特别是在Internet环境中，Java扮演的核心角色以及各种安全技术是怎样相关的。

第二，通过本书，有助于读者对当前Java 平台的安全体系结构有一个全面的理解。主要包括Java语言特征、平台API、安全策略和相应的实施机制。本书不仅讨论了许多函数，而且还深入研究它们为什么这样设计和一些可替换的函数。这些我们——Sun Microsystem的Java安全开发小组成员——都进行了验证。在示范类及其方法时精选了大量实用的例子。其中有些实例是从JDK1.2代码源树中生成的。

第三，本书中还阐述了运用安全体系过程中常见问题，如个人或企业如何管理安全体系、如何定制、扩充和丰富现存的安全体系结构。

最后一点，本书讨论了许多常见错误并提出了许多安全编程技巧，可以使开发者避免在编程过程中发生错误。

本书组织结构

第1章：关于计算机、网络和信息安全的背景知识。

第2章：回顾了Java最初的安全模型——沙盒的若干问题。

第3章：深入讲解了JDK1.2中基于策略驱动和细致访问控制的新型安全体系结构。

第4章：讲述了如何运用JDK1.2中新型的安全体系结构，包括安全策略管理、数字证书和各种安全工具。

第5章：演示了如何用户化安全体系结构的各种问题，包括如何将传统的安全代码移植到JDK1.2平台上。

第6章：评述了如何使对象更具安全性和一些关于安全方面的编程技巧。

第7章：结合常用实例，着重讲解了Java密码体系。

第8章：对Java安全性未来发展方向的展望。

本书主要适用于Java编程人员和安全方面的专业人士。从宏观(体系结构)到微观(设计和实现)角度深刻讲解了Java的安全问题，同时也适用于非专业人士，从整体框架上理解Internet的安全性，在本书中还澄清了许多关于Java安全性的错误认识。

通过阅读本书，读者可以熟悉Java语言的基本特征，希望进一步研究的读者可参阅Arnold和Gosling[2]的书，这是一本非常有益的书。

本书并未详细说明Java的所有API，有关这方面问题请参阅JDK1.2文献。

Li Gong

洛杉矶，加利福尼亚

1999年6月

第1章 计算机与网络安全基础

确保计算机安全的三条黄金法则：不要拥有计算机；不要打开计算机；不要使用计算机。

——Robert(Bob)T. Morris

安全，就是确保不发生有危害的事情。这个简洁的阐述很容易使人迷惑。实际上，安全有非常复杂的解释，对它们进行剖析有助于理解安全的真正含义。

通常情况下，某些经验性法比较适用于安全这个概念。首先，安全总是和效用有关。要想不发生危险，最好就是什么事都不做。打个比方，存放在车库里的车不可能发生交通事故，但是，汽车不跑路并不是我们的意图。我们真正的目的是保障有利的事情发生，而避免有危害的事情。

其次，安全与危险是共存的。例如，前门加锁的有效性很大程度上取决于你所要防卫的窃贼类型。对于小贼，有一定的防卫意义，而对工具齐全、熟于此道的老贼则是毫无意义的。

第三，必须从整个系统的角度去考虑安全问题。系统的安全程度由系统的最薄弱环节决定。也就是说，只保证前门安全是不够的，狡猾的小偷会从所有可能疏于防备的地方潜入房子，尤其是远离装有牢固的锁的那些地方。

第四，实施安全的措施必须简单。试想，如果你每次进门需花30分钟的艰辛去打开一个复杂的锁，你也许会不锁门。

第五，安全的实现必须注意性能价格比。例如，如果你加锁的费用比所防护的内容的价值还昂贵，那是毫无意义的。由于每个人对价值的认定是不同的，事情就变得很复杂。

最后一点但也是很重要的一点就是，安全措施必须尽可能简单。正如专家所说，越复杂的系统就容易出错。使安全系统既简单又可靠是我们的宗旨。

通过对本书的学习，你将会体会到这些符合计算机安全的经验法则。

1.1 密码学与计算机安全

在讨论特定话题之前，应先明确密码学(cryptography)与计算机安全(computer security)这两个课题间的差别。密码学是一种以秘密的方式编码信息，使只有特定的接收者才可以访问被编码的信息的方法。经过一段很长的时间，密码术的应用有了很大的发展，从古老的凯撒密码到第二次世界大战时广泛应用的密码机，一直发展到用计算机软硬件实现的现代密码系统。

直到1960年，分时用户计算机操作系统，如剑桥大学早期的计算机系统[80]和麻省理工大学的Multics系统[69]初次问世，计算机安全才首次成为研究热点。但是自那以后，除了70年代中期，计算机安全技术一直未受到重视[3, 32, 36, 75]，而且在这段时期内，计算机安全大部分是基于军事上的需求。到了90年代，Internet和电子商务得到广泛应用，尤其是Java技术的发展才使计算机安全在商业上广为应用并成为主流技术。

安全机制常常得益于密码学的应用，如基于网络的用户登录协议。不过，它也并不是必须依赖于密码学的应用，例如UNIX系统中对文件的访问控制。

反过来，密码学也依赖于系统的安全性。密码算法常常用软件或硬件实现，它们能否正

常运作关键取决于是否有一个安全的系统。比如，如果系统缺乏访问控制的安全性，攻击者可以修改密码系统的软件算法。可见，安全性的缺乏会直接影响密码术的效用。

1.2 危险和防护

在有关计算机安全的文献中，危险或攻击通常分为三类：

(1) 秘密攻击(secrecy attack)。攻击者试图窃取口令、医疗记录、电子邮件(E-mail)日志、薪水册等机密数据。其攻击手段很多，如贿赂安全卫士、寻求系统的安全漏洞和密码算法的不足之处等。

(2) 完备性攻击(integrity attack)。攻击者企图非法更改部分系统。例如，某位银行雇员为了把顾客的资金存入自己的个人帐户，修改帐户系统从而危及整个交易系统安全性[61]。又如，某位大学生侵入学校管理系统提高他的考试分数，导致危及整个数据安全。这类攻击者常常会为了隐藏“脚印”而删除系统日志。

(3) 可得性攻击(availability attack)。攻击者企图中断系统正常运作，通常也称这种攻击为拒绝服务攻击(denial-of-service attack)。如用大量的IP包“轰炸”机器，使机器与网络的其余部分隔离。这类攻击者通过破坏计算机控制系统引起通信流量阻塞。

这三类攻击是息息相关的。也就是说，某一类攻击的技术和后果经常作为另一类攻击的辅助手段。比如，一个攻击者通过秘密攻击获知口令，这样就有权访问这个系统，然后修改系统资源，最终完成拒绝服务攻击。当遭受攻击时，系统会出错，但大多数系统由于缺乏安全机制仍认为是安全的，例如，有时系统冲突导致在公开的可读目录下倾倒机密的内核信息。[⊖]

同样地，这些攻击的防护机制之间也是紧密相关的。一般来讲，防护有一种或多种目的：防止攻击、检测是否遭受攻击或恢复系统。正如本章下面讲解的，一种防护机制并不是万能的。

为了保护秘密数据，可以把数据存储到不易让攻击者发现的地方，或者是安装严密的访问控制程序拒绝非法访问；也可以用加密技术加密数据使攻击者不能访问到真正数据，除非他能破译加密系统或窃取到密钥，通常破译加密系统是极其困难的。当然，可以同时采用多种防护机制。应注意到：对于秘密数据，大多数重要技术是防止遭受攻击，因为检测数据是否丢失非常困难，而恢复数据几乎是不可能的。

为了保护数据的完备性，可以再次利用前面讲述的任何一种或所有机制。对于数据的完备性，检测是否遭受攻击较为简单，恢复数据也是可能的。比如，对于文件x，可以用著名的单向散列函数f计算其散列值，并分开存储$f(x)$，当x被修改为x'时，可以计算$f(x)$值，与$f(x')$作比较，如果两者不一致，就表明数据的完备性已被破坏。

当然如果对应的$f(x)$也被破坏，检测也许是不可能的。如果存储$f(x)$的地方不够安全，可以利用带有密钥的单向散列函数，并把$f(x,k)$与x存储在一起。如果k的保密性很好，攻击者很难于修改x和散列值[22, 52]。

为了数据的完备性被破坏后，能够恢复数据，可以提前备份数据并在安全的地方存储备份数据[61]。在安全的网络中，也可以用更复杂的分布式计算技术来备份数据[34, 64, 73, 77]。

防护可得性攻击更为复杂。这是因为它不能应用通常的防护和检测技术。这里，计算机

⊖ 当然，可以从其他角度理解攻击。例如，对于个人资料(生日和美国社会安全号)不定期、非法收集以及分布的保密性受到公众的普遍关注。

安全符合容错域计算。一些有益的研究成果在相关研究领域(有时被称作可依赖系统(dependable system))可以查到。为了进一步理解这方面的技术，可以查询[12、24、65、73]中的论文和所引用的文献。

1.3 外围防护

由于有许多潜在的薄弱环节和无穷尽的攻击，而每一种攻击又可能包含多种攻击技术。确保整个系统安全很困难，尤其是系统包括多个通过网络连接的主机时，更令人望而却步。由于系统的安全性是由它的最薄弱环节决定，因此，安全范围必须是整个系统性的。例如，在某个大企业的内部网络中，含有许多不同品牌和型号的机器，而这些机器的操作系统和应用程序又是千差万别。它们再通过不同供应商提供的不同性能和容量的网络设备和路由器连接起来。在这样不同种类、不断变化的环境中，检测整个系统和确保所有部件的安全是相当费时和复杂的。

面对这样一个混杂的场景，公司内部可以从心理和生理上比较容易地找到各部件。通常将它们划分为两大阵营Us和Them，Us包括公司拥有的、被操作的、被相关企业信任的机器，而Them则包括所有其他不友善的或不被信任的机器。一旦界限被化定，问题就在于如何使Them在安全范围之外，而Us在安全范围之内。这种情况就被称为外围防护(perimeter defense)。

构筑外围防护的一种简便方法是不要把Us和Them连接起来。的确，一些军事和商业实体拥有自己的内部网络，它们并不和更广域的网络Internet相连接。为了与外部连接，他们或许有一些独立的终端或机器，但是这些特殊用途的机器通常被守护并将之与内部网络隔离。

如果整个系统内部包含有一些分布在不同地理位置的机器，通过租赁线或共享网络可以连接各站点从而组成专用网络。如果各站点间必须通过网络相互通信，在每两个不同站点之间必须运用加密技术去组成虚拟专用网(VPN)。在图1-1的虚拟模型中，四所院校通过Internet连接，三个站点(MIT、UTAustin和UCLA)形成(VPN)，就可以使它们之间的信息流量自动地与Stanford的窃听行为相隔绝。

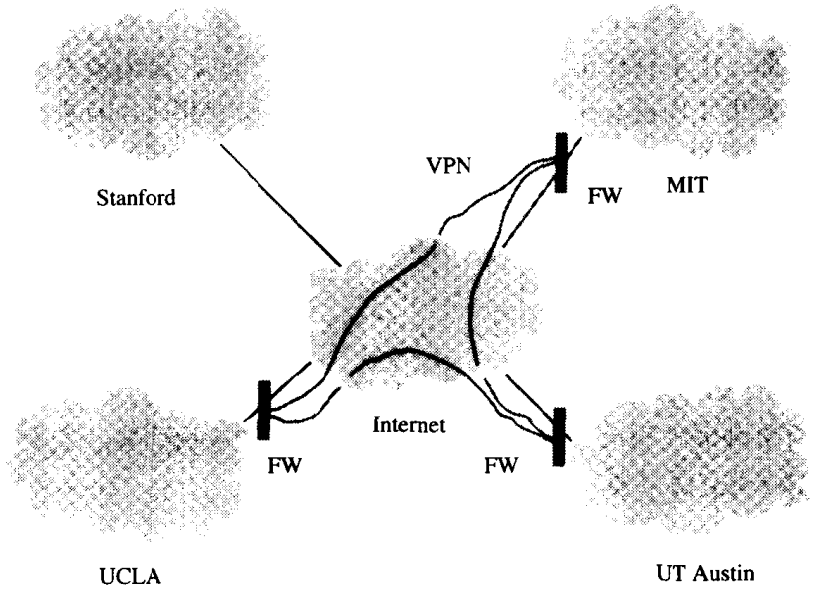


图1-1 虚拟专用网

但是，这样与外界隔离的方法并不理想。例如，随着人们通过Internet 与外界交流需求的不断增加，E-mail已经成为Internet上的“杀手”。万维网(WWW)使Internet 更为流行，漫游网络寻求有用信息对提高生产率大有裨益(当然指的是正当的、合法的)。这些趋势正驱使先前封闭的企事业单位有选择地开放它们的边界控制。在构筑更为有用的外围防护方面，防火墙(fire wall)扮演了一个重要角色。

1.3.1 防火墙

各种防火墙的形式和尺寸千差万别[6]。但总的来说，防火墙是位于专用网络与公共网络之间的一种机器，它对网络流量起着筛选的作用，符合安全策略的特定通信量在某一方向上可以通过。安全策略可以非常简单也可以非常复杂，这通常取决于信源地和信宿地、采用的通信协议和涉及的应用程序等因素。防火墙还可以为信息分配路由，甚至在让信息通过之前对其进行某种操作。作为代理服务器，它还能加密信息。有加密技术的防火墙可以被用来构成VPN。

用防火墙作为外围防护是一种有效的安全方案。防火墙作为中心控制点，易于实现和更新公司的安全策略。但是，也存在一些不足之处：第一，防火墙不能滤除和阻止所有的网络灾难。像HTTP协议这样的信息可以巧妙地通过防火墙。通常，防火墙与移动代码作用是相反的。因为前者试图阻塞或减少流入信息，包括那些后者想获得的信息。其次，防火墙目前已经成为大企业通信的瓶颈。再者，当把防火墙作为代理服务器时，桌面上许多应用程序必须重新编写。不过，对于新的应用程序，这个问题并不重要，这主要因为应用程序内嵌有代理支持。

1.3.2 仅仅使用外围防护的不足之处

作为一套安全方案，只用外围防护是不够的。这主要有以下几方面的原因：

定位和保护所有外围站点是十分困难的，比如，直接用电话线建立连接能有效穿透外围防护[61]。此外，由于企业允许雇员远程或在家中工作，必须严密监视和充分防护内部网络的远距离站点，这常常是不切实际的。

即使在企业内部，由于每件事或每个人并不是完全信任的，就需要对其加以控制。危害性很大的攻击通常发生于内部。由于内部入侵者与外部入侵者相比，有更为有利的条件，所以造成的损失更大。比如，会计部门必须加以防护，使只有被批准的雇员方可发出买单，而专利部门必须隔离开来防止信息泄露给竞争对手。

本章的其余部分回顾了在组织边界的外围和交界处大有用途的安全模型与技术。

1.4 访问控制与安全模型

安全模型(security model)是人们对访问被保护数据加以控制的方法一种抽象。像防火墙一样，安全模型也有多种形式和尺寸，对于不同的应用程序和应用环境，安全模型的要求有很大不同。安全模型分类方法很多，主要包括：

- MAC和DAC模型。
- 数据和信息安全模型。
- 静态(static)和动态(dynamic)模型。

1.4.1 MAC 和DAC模型

一类称为强制性访问控制(Mandatory Access Control，MAC)。在MAC模型中，系统内的实体可以是主题(大致相当于用户、过程、机器等概念)或者对象(大致与控制目标概念相同，例如文件、数据记录)。每一个实体属于一个秘密级别。这样的等级形成一个具有支配关系的框架。例如，“非密”、“保密”、“秘密”、“最高机密”就构成图1-2所示的支配关系。

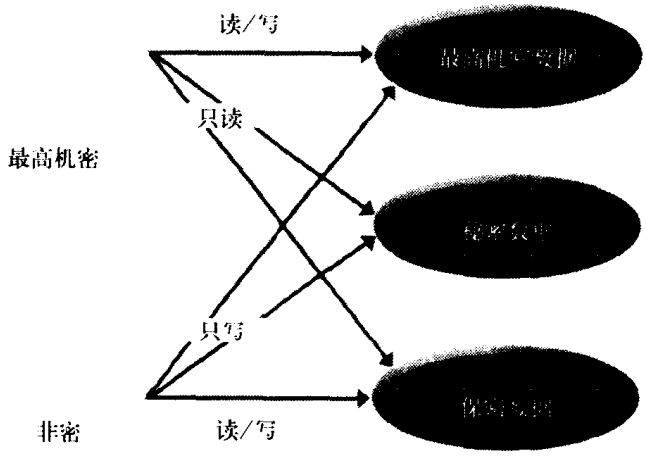


图1-2 MAC 安全模型

Bell和LaPadula[3]的书中示范了满足多种安全级别需要的MAC 模型，它是Multics系统[69]安全的数学模型。在Bell-LaPdula模型中，仅当主题对某一对象具有支配权时，它对该对象享有读许可权；仅当主题被某一对象所支配时，它对该对象享有写许可权。这被称为读下(read-down)和写下(write-down)，更准确一点可称为不读上(no read-up)和不写下(no write-up)。应注意到，在这个模型中，只有当两个实体处在同一级别上时，才可以进行双向通信或通过被信任中间体进行通信。

非Mac模型被叫做自由访问控制模型(Discretionary Access Control，DAC)。UNIX安全模型类似于DAC模型，文件的所有者通过设置文件许可权位决定谁有权对其进行访问。但在UNIX系统中读文件的人可以复制文件并供其他人阅读，而MAC模型不允许这样的自由决定。

1.4.2 对数据和信息的访问

到目前为止，对访问控制的讨论集中在指定对数据的明确访问的模型上，例如，直接读取存储在文件系统中的内容。然而，信息可以隐含地被传送，特别地，通信双方可以通过所谓的隐藏信道(covert channel，它于公开信道(overt channel)是相对的)进行通信。比如，双方共享同一磁盘分区，一方占用了整个分区空间，当另一方由于缺乏可用空间而导致建立新文件失败时，会发现分区空间已经被占用。填充或未填充磁盘空间，通过发送一个“1”或“0”告之对方。

在Lampson关于限制问题的论文[43]中首先研究了这种情况。Lampson论述了对一个应用程序加以限制的困难，使应用程序不能直接或通过发送信息影响外部世界。

这种攻击类型有多大效果与对敌人渗入的担心程度和内部攻击引起的严重损失是密切相关的。内部入侵的模式源远流长，“特洛伊木马”(从特洛伊的灭亡故事中衍生而来)在计算机安全领域指的是为破坏系统而植入计算机的任何程序(重要的计算机安全会议IEEE Symposium

on Security and Privacy已经采用“特洛伊木马”作为会议记录的封面)。最近，美国几位高级政府官员和雇员泄露国家机密给外国机构而构成犯罪的案件就采用了这种渗入方式。

关于限制问题的早期研究成果是基于信息流而非数据访问的安全模型。在这个领域，由Goguen和Meseguer提出的模型[18、19]是大多数理论工作的基础。同时，对实际系统中的隐藏信道也作了研究工作。例如，DEC的一个研究小组构造了一个通信双方共享同一磁盘的例子，一方可以有选择地读取一个文件或另一个文件，这时，如果另一方试图读第三个文件，会引起可检测到的延迟。这个延迟是由磁盘臂的运动造成的。两个不同的延迟值可分别表示为“1”、“0”。这些实际研究出的延迟值在很大程度上决定着隐藏信道的容量、可用性和危害。例如，磁盘臂隐藏信道比填充磁盘分区隐藏信道的速度快许多。

注意到隐藏信道的实际意义是难以衡量的。首先，噪声总是存在的。比如说，在磁盘臂隐藏信道中，第三方独立访问磁盘上的各种文件，会明显压缩隐藏信道的带宽。不过，对于非常机密的材料，如密钥，一个低速隐藏信道易于泄露这些秘密。第二，当一方植入特洛伊木马程序时可以利用隐藏信道。当这样的渗透发生时，利用其他通信形式就成为可能。

此外，对隐藏信道的防护仅在系统内部有效。例如，虽然不允许内部人员把信号传送到外界，但并不能防止内部人员的窃密行为。不过，一些组织，尤其是在美国政府中，比如海军研究实验室的研究人员正在开发一种称为Pump的系统，目的在于通过隐藏信道发送信息不泄漏或只有有限泄漏。JDK1.2并没有全面强调隐藏信道。

1.4.3 静态和动态模型

乍一看来，安全策略似乎是静态的，比如，一位雇员可以读或者不可以读文件A，此外没有第三种选择。实际上，安全策略是动态的，是随时间变化的。当某位雇员在公司内部调换部门时，可以获得先前并不具有的对某一文件的访问许可权。在MAC模型中，数据的机密级别和人员的许可权也是可以改变的。数据的机密级别可以降级和升级，同样人员的许可权也可以降级和升级。

下面几个著名的安全模型证明了这种动态机制。第一种称为高水印模型[44]，其中的数据机密级别随人员许可权而变化。

第二种是中国墙模型[8]。该模型实用性较强，尤其是咨询公司和金融机构可以利用它来避免利益冲突。例如，石油业的顾问可以向公司A或B进行咨询，A、B均为他的客户。这样，这位咨询人员有权访问A或B的资料，但是一旦他访问了A的资料，由于利益冲突，对B的资料的访问将会被拒绝。中国墙模型正是试图表示这种真实的策略。

第三种动态模型源于金融业的Clark Wilson一体化模型[11]，它可以满足金融交易的安全需要。比如，对某种限量货币进行交易，必须要有两位不同的人同时签名，并且签名顺序必须严格。这种模型是最先被广泛采用的安全模型，它明确表明金融业安全需求已经超过军事和政府机构的需求。这类模型主要属于MAC模型。

1.4.4 关于使用安全模型的几点考虑

安全模型用途很广，如驱动和分析计算机系统的设计，或形成系统操作的基础，但是它在使用中会产生许多有趣的问题，对这些安全问题目前已有一定程度的研究。

首先是所用模型的可决定性(decidablity)。即当给定一个实际系统的安全模型或对系统的

安全性有特殊要求和特殊条件(如某位雇员不允许访问文件A)时，能够根据实际运行情况分析系统是否安全。在通常情况下，系统是否安全是不可判决的(参见文献[32])。后来，为了解决这个问题，对模型作了一些限制，问题就可以解决了。在许多模型中，为判决系统是否安全的计算复杂度仍属于Np-难题[172]。

第二是模型的不可能性或不可实现性。一个实际系统是相当庞大的，把它作为一个整体来规定和划分，这就是可组成性(composability)。满足一套安全特性的各种部件以某种特定方式互相连接，那么由这些部件所构成的整个系统就会自动符合另一套安全特性[48, 57]。在不远的将来，就可以组建一个安全的、具有可组成性的系统。

第三，安全机制必须改进传统系统，至少安全地与传统系统连接起来，这就要求传统系统必须具有安全互用性。安全互用性就是每一个传统系统的安全特性在它最初的定义下必须是可保持的。不过，即便在非常简单的模型中判定一个特殊互用性是否安全通常也是一个Np-难题[27]。

最后，安全并不仅仅意味着机密。系统的完备性也是至关重要的。一个早期的完备模型[1]是Bell-LaPadula机密模型的双倍。有些人也把完备性看作可靠性和准确性的一方面，这样在容错域可以得到相关的解决手段。

1.5 密码系统的使用

密码系统就是信息的编码和解码。密码分析(cryptanalysis)是密码学的反过程，是一种解码或在不获知密钥的条件下对编码信息的破译方法。密码学(cryptology)指的是整个密码研究领域。

安全性和密码术是两个彼此相关但又互有差别的领域。许多人经常混淆这两者。在某种意义上它们是正交的，每一方都有自己独立的用途。虽然它们是互相渗透的。比如，到目前为止讨论的所有安全模型都不必使用密码术。密码技术可以增强保密性和完备性，与计算机安全相比它是一种抽象研究领域。不过，现代密码术大部分存在于计算机和通信系统中，访问控制在保护对密钥的访问时非常有用。实际上，入侵一个密码系统就是设法损害密钥存储的设备。

最常使用的密码概念包括：

- 单向散列函数(one-way hash function)。
- 对称密码(symmetric ciphers)。
- 非对称密码(asymmetric ciphers)。

下面三个小节对它们进行依次讲解。

应注意到：在密码术中，根据Claude Shannon理论，几乎没有一个密码系统是绝对安全的。在某种意义上，只要攻击者有足够的知识水平和计算能力就可以破译密码系统。唯一例外的是一次一密乱码本(one-time pad)系统，它与明文的长度一样而且只被使用一次。在收发双方有以安全方式互换密钥时，该系统是可行的。

目前，关于这一个论题讲解最深入的参考书目是《Handbook of Applied Cryptography》[151]，对于那些不想深入研究密码术背景和相关论题的人，《Applied Cryptography》[174](由机械工业出版社引进出版，中译本名为《应用密码学》)一书更为合适。

1.5.1 单向散列函数

单向散列函数是一种重要的构造块，它有助于获得数据的完备性。在存储和转换过程中

这些函数可以用来保护数据。

根据Knuth[38]所述，散列的思想是由IBM 的两个开发小组人员首先提出的。我所能检索到的关于单向函数的概念最早是在1968年由Wilkes[80]提出的，他论述了Needham⊖ 提出的剑桥分时计算机系统中所用的单向函数。

单向散列函数的概念可以追溯到许多年以前，众多研究者Merkle[52]、Naor和Yung[58]、Damgs ard[13]都提出一些定义方法。Meyer和Schilling[54]、Merkle[53]、Robin[64]、Rivest[67]和其他研究者还提出了一些实用的设计方法。

单向散列函数引入了许多术语，一些是可以相互替代的，另一些是为了强调不同的假设前提，如单向散列函数、自由碰撞函数、手印函数、修正检测码和消息鉴别码。

单向散列函数是正向计算简单、反向计算，而且很难找到两个不同的输入值对应于同一个输出值的一种函数。这些特性允许下面的完备性防护。假定你在磁盘上存储文件，并假想它不会被篡改，把文件内容作为函数输入，可以计算散列函数值，它通常比文件长度短许多。以后，可以把文件的当前内容作为输入计算其散列函数值，如果新旧函数值不一致，就表明文件很可能已经被篡改。在这种意义下，单向散列函数在文件内容与函数值之间充当一种非假造连接。见图1-3单向散列函数示意图。

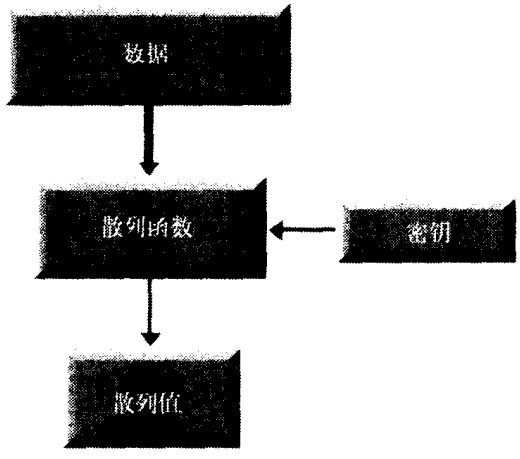


图1-3 单向散列函数

通常，可以将密钥与单向散列函数的输入组合在一起，这样，在未获悉密钥的情况下，不能正确计算或预测散列值。所以说，单向散列函数就不仅是文件内容与散列值之间的一种不可伪造链，而且还是密钥与散列函数值间的一个不可伪造链。

1.5.2 对称密码

对称密码就是在密钥的作用下，将输入(明文(plaintext))转换成输出(密文(ciphertext))，使只有拥有密钥的实体可以将密文转换成明文(密码分析学除外)(见图1-4)。

对称密码有很长的历史，在早期的凯撒系统[39]中首次被采用，从那以后，它们被广泛应用，例如，数据加密标准(DES)[62]和国际数据加密算法(IDEA)中都采用了对称密码。

⊖ Roger Needham 后来回忆说，这种思想是1967年在剑桥的鹰酒吧首次提出，但由于酒吧里人来人往，没有人能记起是谁提出的。

对称密码也称为秘密密钥密码(secret-key cipher)。在使用对称密码的系统中，通信双方必须共享同一密钥，而且，不同的通信双方间的密钥又必须不同。这必然在密钥管理和分配方面引起许多问题。当通信群体人数增加时，在理论上，所需的密钥是成指数增长的。

对称密码可以以不同模式被操作(如各种反馈模式)，也可以堆叠在一起改善整个系统的加密程度(如三重DES)。

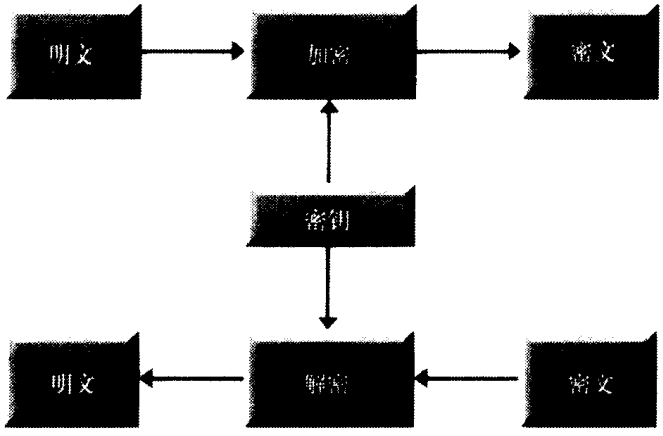


图1-4 对称密码

1.5.3 非对称密码

非对称密码类似与对称密码，不过，它必须采用一对密钥而不是一个密钥。其中一个密钥叫公钥(public key)，用来加密明文，另一个密钥叫私钥(private key)，用来破译密文。在给定私钥条件下，很容易推断出公钥，反之很难。这种性质使人们可以通过公共信道互换公钥的方法进行私人通信。在对称密码中，人们必须通过专有信道共享密钥。著名的非对称密码系统有Diffie-Hellman[114]和RSA[168]，它们通常可用来为对称密码系统加密和互换密钥。

非对称密码的另一个显著特性是：加密和解密是可逆的(reversible)。也就是说，一个人可以用私钥进行解密操作把明文变成密文，也可以用公钥进行加密操作把密文变成明文。在这种情况下，公钥是公开的，不需要机密保护。此外，由于只有私钥的拥有者可以产生密文，密文可以作为明文的数字签名(digital signature)。公钥的拥有者可以鉴别签名。RSA也许是最广泛用做数字签名的非对称密码系统。另一种系统，数字签名算法(DSA)，已被美国政府规定为国家标准，只具有数字签名功能，而不能用于加密系统。

一方为了证实另一方是否是公钥的真正所有者，可以通过被证明方为验证提供证书。公钥证书(public-key certificate)是一种来自另一方的数字签字，声明另一方的公钥(及其他信息)有一些特殊值。有时，可能有一系列证书，每一个证书含有一个需待证实的公钥，第一个证书叫根证书(root certificate)，没有其他公钥验证它。由于根证书自身含有证实自己的公钥，所以，通常又称为自签证书(self-signed certificate)。本书后几章将更深入地讨论证书。

1.6 鉴别

另一个安全问题就是鉴别(authentication)。它是对用户身份(或代表用户的机器操作)的证实。在分时计算机系统中，为了获知用户登录的身份，就出现了用户身份的鉴别问题。正如

前面提到的大多数安全模型是对某些用户是准许的，而某些用户是被拒绝的，鉴别对增强访问控制策略是至关重要的。

随着网络计算机系统的出现，鉴别显得越来越重要。网络是共享的和大众化的，获知另一端连接的用户的身份，就像用户需知将要连接的系统的身份一样同等重要。

目前，存在许多鉴别协议，但这些经过专家潜心研究的鉴别协议大多数都有微小的缺陷，这就导致鉴别技术成为一个重要的研究课题。

鉴别的基本方法就是首先让另一端的用户提供名字和口令，然后与系统记录比较。因为监控网络通信的任何一个人都可以获知口令并加以利用，这种方法易被窃听和受到猜测攻击。后来，对这种基本方法进行了一些改进，如一次性口令[40]、IETF标准所说的OTP[31]。但这些改进都很有限，因为一个人只可以携带有限个一次性口令。

这种基本方法是一种基于询问和响应的方法。当然，它们也能扩展到借助密钥分布函数来实现。不同的实体只能共享特有的密钥分配中心产生的钥匙，密钥分配中心可以动态地生成密钥。Needhan-Schroeder协议[59]是网络鉴别方面最早的协议实体A、B把密钥分配中心作为被信任的第三方来建立短期安全会话。这个协议是Kerberos系统(MIT Athena项目的一部分)的基础，后来被DCE、IETF标准所采纳[56]、[60]。

协议设计存在许多不安全之处。像Needham-Schroeder协议在许多方面逐渐暴露出缺陷[9]、[26]。对安全协议的攻击包括重放攻击(replay attack)和交叉攻击(interleaving attack)。一个攻击者窃听和记录网络信息并重新利用它们(有时需要巧妙地做一些修改)可以攻破安全体系。但这些又不易被协议的设计者发现。于是就有许多正式或非正式的协议分析技术被提出并加以利用。文献[9、15、26、50、55]中包括最新的模型检测工具型应用程序。

为应用而设计的认证协议是一个特别重要的问题，包括人们能记住的用户口令。由于这些口令通常是从相当小的范围内选出的(如字典中的词组)，它们很容易被查找和推演出来。1989年之前，由安全研究团体出版的所有认证协议都面临口令易于猜测这一问题。攻击者只要监控网络流量并设法获得认证协议的运行记录，就可以猜测每位用户的口令并验证猜测是否正确。所有工作都是离线的和不可察觉的。1989年后期，解决这个问题的技术方案[21]、[47]开始出现，包括EKE和A-EKE[4]、[5]。智能卡和其他基于硬件的安全设备有利于防止用户猜测口令。

1.7 移动代码

移动代码(mobile code)并不是一个新概念，理论上说，能引起远程系统不同动作的任何代码都可以被看作移动代码。移动代码是整个分布式计算领域正常工作的前提。它包括域名服务(DNS)数据信息、远端程序调用(RPC)的远端命令和UNIX系统远端外壳的执行脚本。本小节主要讨论可执行脚本，这种移动代码之所以非常流行，一是因为它有利于合理分配客户机/服务器上的计算负荷，二是因为它可以降低对网络带宽的要求。

PostScript文件就属于这种代码。只有在显示和查看PostScript文件时其文件内容才被执行。像微软Word文档中的宏一样，只有在文档被读时，宏才被解释执行。又例如Lisp，通过Emacs(一种功能强大的文本编辑器)人们可以阅读E-Mail，Emacs系统解释Lisp程序就像它自己所有的程序一样。当查看内部Emacs时，在E-Mail信息中嵌入的Lisp程序段才被激活。Activex控件和Java小应用程序都是这样的例子。

活动内容并没有带来新的危害，相反，它有助于暴露普通安全机制的不充分性。例如，

当利用移动代码来更新DNS时，其界面范围相当小，使得系统更安全。但是，在调用Activex控件时，其界面范围就扩展成整个Win32 API界面，那么整个API的安全漏洞都可能被发现。

移动代码的不断增加引起两种反应。一方面，人们尽力增强系统安全性以便较好地控制和利用移动代码有利的一面，另一方面，人们恐惧移动代码不利的一面，想把它阻止在外围。后者阻碍了时代发展，这是因为移动代码和活动内容流通途径很多(E-mail)，而滤掉每一个E-mail信息和部分消息是不可能的。Java技术的一个重要设计目标就是使Java成为移动代码的安全平台。

1.8 Java安全性的适用范围

前面各小节在相当广的范围内，描述了目前系统安全性方面的知识。从防火墙到访问控制，从加密到鉴别，Java安全是一种非常重要的谜。这主要因为Java技术是与平台无关的，并且是为Internet和Web编写移动代码、可执行代码的最好工具。

目前Java得到普遍应用，如在金融机构、在线电子商务软件和其他关键应用程序的部件中都采用了Java。所有这些表明，Java平台必须实现自己的诺言：Java平台是安全的Internet编程平台。[⊖]

Java平台既可被看作客户端应用程序(如在浏览器内部运行Java)、服务器端应用程序(用Java编写的服务器软件)，也可被看作是一种操作系统(在MS-DOS或裸机上直接运行的JavaOS)。不同的用途或许需要不同甚至相互冲突的安全特征。JDK1.2内含公共函数，同时有充分的界面，可以很方便地处理特殊需求。

当Java技术在操作系统内部(例如，Solaris2.6或MS-Windows)得到利用时，它的出现不会改变底层系统的安全特性。例如，在Solaris系统中，Java虚拟机(JVM)的实例只能访问正在运行JVM的用户资源。不过，如果整个应用程序界面只限于Java平台，整个系统的安全性能会得到改善。如果所有的应用程序均采用Java编写，那么在Ms-Windows平台上的许多安全问题就会消失(实际上是隐藏在Java界面的底层而不易被发现)。

在本章最后，我想强调Java平台的安全特性并不局限在JDK1.2中所讨论的，JDK的后续版本无疑会继续丰富安全特性。此外，一整套标准Java界面已经或正在设计，其中包括加密、安全套接层(SSL)、用户鉴别等，这样，Java不仅是而且实际就是许多种安全谜语。正如SSL和浏览器把密码术带到大众市场一样，总之，Java技术在使计算机安全技术成为主流技术中扮演了重要角色。

⊖ 应该澄清JavaScipt并不是以Java为基础，两者只是在名字上相似而已。JavaScipt不具有Java所包含的全面的安全性考虑和安全机制。

第2章 Java语言的基本安全特点

自从Java技术出现以来[30, 46]，人们对Java的安全性产生了极大的兴趣。部分因为Java语言从诞生之日起，安全性就被称为其重要设计目标之一。同时安全性也是把Java技术同其他技术区分开来的一个重要方面。

一项新技术的早期版本很少包括合理的安全特点。因此，Java作为Internet安全编程最佳平台地位的确立不但吸引了安全性专业人员，而且吸引了普通计算机公司的注意。长期从事安全研究的开发人员、学者以及学生已经总结出大量设计细节，并且用Sun Microsystems公司为此目的发行的JDK软件包编写出大量源代码。就连大众传媒也注意到了这个热潮。无论是《华尔街周刊》还是《纽约时报》都在显著位置加以报道。⊖

从Java技术提供者的角度来看，Java的安全性包括以下两个方面[23]：

- Java平台(主要通过JDK实现)是一个安全、现成的平台，在平台上可以以一种安全的方式实现基于Java的功能。
- Java所实现的安全工具和服务使得一些领域(如企业)对安全性广泛的需求得以满足。

Java技术的应用同时也提出了一系列有趣的问题，我们将在以后的章节讨论。本章主要侧重于Java语言和平台所提供的基本安全特点。

2.1 Java语言和平台

Java语言最初主要应用于嵌入式电子产品，如手持式设备和机顶盒。它是一般意义上的面向对象的编程语言并且非常简单，这样很多的程序员可以迅速掌握它。由于Java被特别设计成与平台无关，因此开发人员一旦用Java开发程序，就可以在Internet上的任何地方安全运行。Java同C和C++是有联系的，但又有许多区别，它包括了很多C和C++忽略的方面并且借鉴了其他语言的一些特点。

Java语言是很严格的，它没有任何不安全的结构，例如没有索引核查的数组访问，因为这类结构往往会导致不定的、不可预测的程序操作⊜。Java提供了一种主要由废物收集器实现的自动存储管理机制。Java语言还进一步避免了那些由内存重新分配命令(如C的“free”或C++的“delete”)所带来的问题。

Java程序一般被编译成二进制代码指令组，并且其二进制格式由JVM规范[30]定义。为了对编程提供更全面的支持，Java也定义了大量的包。Java程序通常以二进制文件的形式存储，这些文件表示编译过的类和界面。二进制类文件被加载入JVM，然后连接、初始化、执行，以下是一个简单的Java程序的例子。

```
class Test {
   public static void main(String[] args) {
      for (int i = 0; i < args.length; i++)
         System.out.print(i == 0 ? args[i] : " " + args[i]);
      System.out.println();
   }
}
```

⊖ 参见[49]

⊜ 近来的研究表明，CERT发布的警告大约有50%的是由缓冲器溢出造成的。