本资源来自数缘社区

http://maths.utime.cn:81

欢迎来到数缘社区。本社区是一个高等数学及密码学的技术性论坛，由山东大学数学院研究生创办。在这里您可以尽情的遨游数学的海洋。作为站长，我诚挚的邀请您加入，希望大家能一起支持发展我们的论坛，充实每个版块。把您宝贵的资料与大家一起分享！

数学电子书库

每天都有来源于各类网站的与数学相关的新内容供大家浏览和下载，您既可以点击左键弹出网页在线阅读，又可以点右键选择下载。现在书库中藏书 1000 余本。如果本站没有您急需的电子书，可以发帖说明，我们有专人负责为您寻找您需要的电子书。

密码学论文库

国内首创信息安全专业的密码学论文库，主要收集欧密会（Eurocrypt）、美密会（Crypto）、亚密会（Asiacrypt）等国内外知名论文。现在论文库中收藏论文 4000 余篇（包括论文库版块 700 余篇、论坛顶部菜单“密码学会议论文集”3000 余篇）。如果本站没有您急需的密码学论文，可以发帖说明，我们有专人负责为您寻找您需要的论文。

提示：本站已经收集到 1981－2003 年欧密会、美密会全部论文以及 1997 年－2003 年五大会议全部论文（欧密会、美密会、亚密会、PKC、FSE）。

数学综合讨论区

论坛管理团队及部分会员来源于山东大学数学院七大专业（基础数学、应用数学、运筹学、控制论、计算数学、统计学、信息安全），在数学方面均为思维活跃、成绩优秀的研究生，相信会给您的数学学习带来很大的帮助。

密码学与网络安全

山东大学数学院的信息安全专业师资雄厚，前景广阔，具有密码理论、密码技术与网络安全技术三个研究方向。有一大批博士、硕士及本科生活跃于本论坛。本版块适合从事密码学或网络安全方面学习研究的朋友访问。

网络公式编辑器

数缘社区公式编辑器采用 Latex 语言，适用于任何支持图片格式的论坛或网页。在本论坛编辑好公式后，您可以将自动生成的公式图片的链接直接复制到你要发的帖子里以图片的形式发表。

如果您觉得本站对您的学习和成长有所帮助，请把它添加到您的收藏夹。如果您对本论坛有任何的意见或者建议，请来论坛留下您宝贵的意见。

附录 A：本站电子书库藏书目录

http://maths.utime.cn:81/bbs/dispbbs.asp?boardID=18&ID=2285

附录 B：版权问题

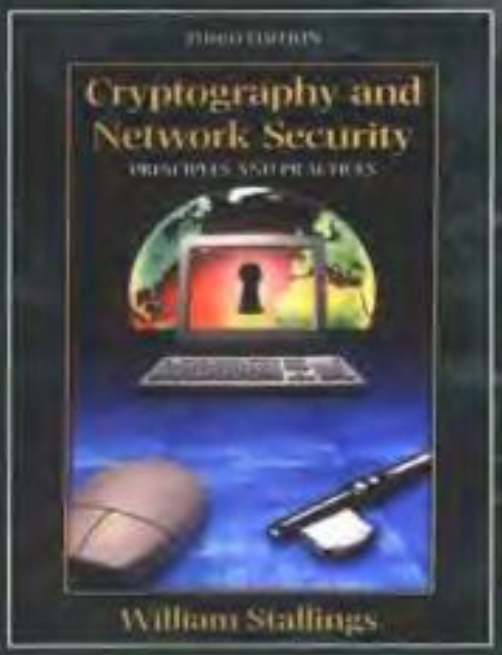
国外计算机科学教材系列

密码编码学与网络安全

——原理与实践（第三版）

Cryptography and Network Security

Principles and Practices, Third Edition



[美] William Stallings 著
刘玉珍 王丽娜 傅建明 等译
张焕国 审校

PEARSON
Prentice Hall

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

经典教材

密码编码学与网络安全
——原理与实践（第三版）

Cryptography and Network Security
Principles and Practices, Third Edition

William Stallings 为读者提供了一本关于密码编码学和网络安全的最优秀书籍。本书详细讨论了网络安全的实际应用。既适合用做学生的教材，又适合专业技术人员作为技术参考书。

主要特色

- 新增内容——高级加密标准（AES）：新的美国数据加密标准
- 新增内容——RC4：一种应用最广泛的流密码
- 新增内容——有限域：信息安全的一种数学理论基础
- 扩充内容——椭圆曲线密码及其应用
- 叙述清楚——便于教学，通俗易懂，便于理解
- 第一手的实践经验——为读者的研究项目、编程项目和阅读/报告提供强有力的支持
- 详细资料和信息可通过网站 http://www.WilliamStallings.com/StudentSupport.html 获得

作者简介

William Stallings：在计算机网络和计算机体系结构领域作出了独特的、广泛的贡献。他在18个专题方面编写出版了48本书籍，五次获得教材和著作家协会颁发的优秀计算机科学和工程教材奖。他还作为独立顾问为计算机网络制造商、软件开发商、研究机构和计算机用户提供咨询服务。William Stallings 获得了麻省理工学院计算机科学博士学位。他在 Prentice Hall 出版的著作都可以从 Prentice Hall 的网站 http://www.prenhall.com/stallings 上找到。

审校者简介

张焕国：教授，博士生导师，武汉大学计算机科学与技术学院副院长。主要从事信息安全、容错计算和计算机应用方面的教学和科研工作。现任中国密码学会理事，中国计算机学会容错专业委员会委员，湖北省电子学会副理事长，湖北省暨武汉市计算机学会理事。

ISBN 7-5053-9395-2

9 787505 393950 >

责任编辑：谭海平
封面设计：毛惠庚

ISBN 7-5053-9395-2　定价：49.00 元

国外计算机科学教材系列

密码编码学与网络安全

——原理与实践（第三版）

Cryptography and Network Security
Principles and Practices
Third Edition

[美] William Stallings 著

刘玉珍 王丽娜 傅建明 等译

张焕国 审校

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书系统地介绍了密码编码学与网络安全的基本原理和应用技术。全书主要包括下列四个部分。传统密码部分详细讨论了传统密码算法和设计原理，包括使用传统密码来保证秘密性。公钥密码和 hash 函数部分详细讨论了公钥密码算法和设计原理、消息认证码和 hash 函数的应用，以及数字签名和公钥证书。网络安全部分讨论了系统层的安全问题，包括入侵者和病毒造成的威胁及相应的对策、防火墙和可信系统的应用。本书第三版与第二版相比，在继续广泛涵盖密码编码学与网络安全领域内容的同时，新增了有限域、高级加密标准（AES）、RC4 密码、CTR 模式等内容，并对椭圆曲线密码部分的内容做了很多扩充。此外，对于基本内容的讲述方法也有许多变化和更新。本书内容全面，讲述深入浅出，便于理解，尤其适合于课堂教学和自学，是一本难得的好书。特别是本书后面讨论的网络安全在现实世界中的应用，包括已经实现的和正在使用的提供网络安全的实际应用。

本书可作为研究生和高年级本科生的教材，也可供从事信息安全、计算机、通信、电子工程等领域的科技人员参考。

版权贸易合同登记号　图字：01-2002-5703

图书在版编目（CIP）数据

密码编码学与网络安全：原理与实践：第二版/（美）斯托林斯（Stallings, W.）著；刘玉珍等译.
-北京：电子工业出版社，2004.1
（国外计算机科学教材系列）
书名原文：Cryptography and Network Security: Principles and Practices, Third Edition
ISBN 7-5053-9395-2

Ⅰ.密…　Ⅱ.①斯…　②刘…　Ⅲ.①电子计算机-密码-理论　②计算机网络-安全技术
Ⅳ.①TP309.7　②TP393.08

中国版本图书馆 CIP 数据核字（2003）第 108188 号

责任编辑：谭海平
印　　刷：北京兴华印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 1092　1/16　　印张：32　字数：819 千字
印　　次：2004 年 8 月第 2 次印刷
定　　价：49.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。
联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳－希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯托林斯（William Stallings）、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

译 者 序

随着计算机与数据通信网络的高速发展和广泛应用，社会对计算机和数据通信网络的依赖越来越大。如果计算机和数据通信网络的安全受到危害，将会危及国家安全，引起社会混乱，造成重大损失。因此，确保计算机和数据通信网络的安全成为世人关注的社会问题，并成为计算机科学技术的热点领域。

为了适应信息科学技术发展的这一新特点，我国政府和科技界已将信息安全技术列为今后一段时期的重点发展领域。许多大专院校都开办了信息安全专业，开设了信息安全课程，因此迫切需要一本合适的教课书。为此，电子工业出版社组织我们翻译出版了这本优秀的教科书。

本书的作者 William Stallings 先后获得了 Notre Dame 电气工程学士学位和麻省理工学院计算机科学博士学位。他编辑出版了 48 本计算机网络和计算机结构领域的书籍，在帮助人们了解计算机网络和计算机结构的技术发展方面做出了卓越的贡献。William Stallings 的著作不仅学术造诣很高，而且十分实用，连续五次获得了教材和著作家协会颁发的优秀计算机科学和工程教材奖。

本书系统地介绍了密码编码学和网络安全的基本原理和应用技术。全书主要包含下列四个部分：传统密码部分详细讨论了传统密码算法和设计原理，包括使用传统密码来保证秘密性；公钥密码和 hash 函数部分详细讨论了公钥密码算法和设计原理、消息认证码和 hash 函数的应用，以及数字签名和公钥证书；网络安全实现部分讨论了重要的网络安全工具和应用软件；系统安全部分讨论了系统层的安全问题，包括入侵者和病毒造成的威胁及相应的对策、防火墙和可信系统的应用。

本书第三版与第二版相比，在继续广泛涵盖密码学与网络安全领域内容的同时，新增了有限域、高级加密标准（AES）、RC4 密码、CTR 模式等内容，并对椭圆曲线密码部分的内容做了很多扩充。除此之外，对于基本内容的讲述方法也有许多变化和更新。

本书内容全面，讲述深入浅出，便于理解，尤其适合于课堂教学和自学，是一本难得的好书。特别是本书后面讨论的网络安全在现实世界中的应用，包括已经实现的和正在使用的提供网络安全的实际应用。本书可作为研究生和高年级本科生的教材，也可供从事信息安全、计算机、通信、电子工程等领域的科技人员参考。

本书前言、第 1 章和第二部分由刘玉珍翻译，第一部分由曾祥勇、王张宜翻译，第三部分及附录由傅建明翻译，第四部分由王丽娜翻译，全书由张焕国统稿和审校。

由于译者的专业知识和外语水平有限，书中错误在所难免，敬请读者指正，译者在此先致感谢之意。

前　言

在当前全球电子互连互通的时代，由于病毒、黑客、电子窃听和电子欺诈，使得信息安全性在任何时候都十分重要。第一，由于计算机系统的大量增加以及计算机系统通过网络互联，使得组织和个人越来越依赖于存储的信息和利用计算机系统传输的信息。这样就需要保护数据和资源不被泄露，保证数据和消息的真实性，保护系统不受基于网络的攻击。第二，密码和网络安全的学科已经成熟，这样可开发出方便实用的应用软件来加强网络安全。由于这两种发展趋势，本书所讨论的内容就显得十分重要。

本书的目的

本书的目的是概述密码编码学和网络安全的原理和应用。前两部分讨论密码编码学和网络安全技术，阐述网络安全的基本内容。其他部分讨论网络安全的应用，包括已经实现或正用于提供网络安全的实用应用软件。

因此本书涉及多个学科。特别地，要想理解本书讨论的某些技术的精髓，必须要有数论的基本知识和概率论中的某些结果。然而本书试图自成体系，不仅给出了必需的数论知识，而且让读者对这些知识有直观的理解。采用的方法是，在需要的时候才引入这些背景知识。这样有助于读者理解讨论这些知识的动机，作者认为这种方法比把所有的数学知识一次性全部放在本书开头要好。

本书适用的对象

本书适合于教师和专业人员使用。本书可作为计算机科学、计算机工程、电气工程专业本科生密码编码学和网络安全方面课程的教材，学时为一学期。本书也可作为参考用书或作为自学教材。

本书的组织

本书由四个部分组成：

1. **传统密码**：详细讨论了传统密码算法和设计原理，包括使用传统密码来保证秘密性。
2. **公钥密码和 hash 函数**：详细讨论了公钥密码算法和设计原理，该部分还讨论了消息认证码和 hash 函数的应用，以及数字签名和公钥证书。
3. **网络安全实现**：讨论了重要的网络安全工具和应用软件，包括 Kerberos、X.509v3、PGP、S/MIME、IP Security、SSL/TLS 和 SET。
4. **系统安全**：讨论了系统层的安全问题，包括入侵者和病毒造成的威胁及相应的对策、防火墙和可信系统的应用。

另外，本书还给出了术语表、常用的首字母缩略词表和参考文献。每一章中都有课后习题、思考题和关键术语表、推荐读物和网址。

在每一部分开头，按章详细介绍了该章的主要内容。

教师和学生的 Internet 服务

本书的网页可给学生和教师提供支持，该网页包括一些相关的站点、以 PDF 格式存储的本书中出现的图片和表格、有关本书的 Internet 邮件列表的签名信息。该网页是 WilliamStallings.com/Crypto3e.html。Internet 邮件列表的建立使得使用本书的教师可以相互或与作者交换信息、建议或讨论问题。若发现印刷或其他错误，则在 WilliamStallings.com 处可找到本书的勘误表。另外在计算机科学专业学生资源网址 WilliamStallings.com/StudentSupport.html 提供了有关计算机专业学生和专业人员的信息和链接。

讲授密码编码学和网络安全的计划

对许多教师来说，密码编码学或信息安全课程的一个重要组成部分就是制定一个或一系列计划，使得学生有机会亲手实践，以加深从课本中学到的知识。本书在很大程度上对该课程的讲授计划提供支持。教师手册不仅包含如何布置和安排计划，而且还包括一系列涵盖本书内容的推荐教学计划：

- **研究计划**：一系列指导学生研究 Internet 有关课题以及撰写研究报告的课外研究课题。
- **程序设计计划**：一系列涵盖大部分课程内容且可在任何平台上用任何适当的语言实现的程序设计项目。
- **课外阅读/报告**：每一章在参考文献中都包含有论文列表，可让学生阅读并写出简短报告。

第三版新增内容

本书第二版出版后的四年中，该领域仍处于不断的变革之中。该新版中，我试图在继续广泛涵盖本领域内容的同时，增加这些新的变化。进行本次修订之初，本书由许多讲授该领域课程的教授仔细审阅过。而且，许多研究该领域的专业人员也审阅过某些章节。这使得许多地方的叙述变得清晰、紧凑，对插图也进行了改进，而且增加了许多新的“现场试验”问题。

除了这些为改进教学法和用户友好性所做的修改以外，还有一些实质性的变化贯穿本书，最主要包括下列几个方面：

- **新增内容——高级加密标准(AES)**：该领域在过去四年中发生的最重要的事件莫过于采用了高级加密标准(AES)。设计该传统加密算法是为了替代 DES 和三重 DES。该算法可能很快会成为最为广泛使用的传统加密算法。本书新增了对 AES 的详细讨论。
- **新增内容——有限域**：AES 和椭圆曲线密码学都使用有限域。本书新增加了一章，简洁且清晰地描述了该领域中那些必不可少的概念。
- **新增内容——RC4 密码**：RC4 是使用最为广泛的流密码。它是为网络浏览器和服务器

间通信而定义的 SSL/TLS(安全套接层/传输层安全)标准的一部分,它也用于 WEP(Wired Equivalent Privacy)协议中,该协议是 IEEE 802.11 无线 LAN 标准的一部分。

- **新增内容——CTR 模式**:NIST 最近批准了分组密码加密的计数器模式,以适应对加密速度要求高的应用。
- **扩充内容——椭圆曲线密码学**:ECC 是一种愈来愈重要的、愈来愈被广泛使用的公钥技术,因此本书对 ECC 部分的内容做了很多扩充。

致谢

本次修改得益于许多人的审阅,他们花费了大量的时间和精力。下列这些人员审阅了所有或大部分手稿:Martha Sloan(Michigan Tech)、Xiangyang Li(Illinois Institute of Technology)、Ed Fernandez(Florida Atlantic University)、Dan Warren(Naval Postgraduate School)、Phillip Enslow(Georgia Tech)和 Cetin Koc(Oregon State)。

我还要感谢那些详细审阅其中某一章的人员:Steve Tate、Breno de Medeiros、Daniel Kifer、Sam Staton、Thiébaut Mochel、Mads Sig Ager Jensen、Alexey Kudravtsev、Stefan Katzenbeisser 和 Iulian Dragos。Joan Daemen 审阅了关于 AES 的章节。

下列人员在教师手册中的课程计划方面做了工作:Henning Schulzrinne(Columbia University)、Cetin Kaya Koc(Oregon State University)和 David Balenson(Trusted Information Systems and George Washington University)。

同时,我还要感谢那些在家庭作业方面做了工作的人员:Luke O'Connor;MITRE 的 Joseph Kusmiss;Stratus 的 Carl Ellison;Shunmugavel Rajarathinam;Jozef Vyskoc,他在 Sherlock Holmes 问题上做了很好的工作。

在这么多帮助面前,我几乎没有什么可以居功自傲的。但我可以自豪地说,没有这些帮助,我也会选择所有这些内容。

目　　录

第 1 章　引言 …… 1
1.1　服务、机制和攻击 …… 2
1.2　OSI 安全框架 …… 4
1.3　网络安全模型 …… 9
1.4　本书概览 …… 10
1.5　推荐读物 …… 11
1.6　Internet 和 Web 资源 …… 11

第一部分　对 称 密 码

第 2 章　传统加密技术 …… 14
2.1　对称密码的模型 …… 14
2.2　代换技术 …… 18
2.3　置换技术 …… 29
2.4　转轮机 …… 30
2.5　隐写术 …… 32
2.6　推荐读物和网址 …… 33
2.7　关键术语、思考题和习题 …… 34
第 3 章　分组密码与数据加密标准 …… 38
3.1　简化 DES …… 38
3.2　分组密码原理 …… 44
3.3　数据加密标准 …… 50
3.4　DES 的强度 …… 58
3.5　差分分析和线性分析 …… 59
3.6　分组密码的设计原理 …… 62
3.7　分组密码的工作模式 …… 65
3.8　推荐读物 …… 71
3.9　关键术语、思考题和习题 …… 71
第 4 章　有限域 …… 75
4.1　群、环和域 …… 75
4.2　模运算 …… 78
4.3　Euclid 算法 …… 83
4.4　有限域 GF(p) …… 85
4.5　多项式运算 …… 88
4.6　有限域 GF(2^n) …… 93

4.7 推荐读物和网址 …… 99
4.8 关键术语、思考题和习题 …… 100
第 5 章 高级加密标准 …… 103
5.1 高级加密标准的评估准则 …… 103
5.2 AES 密码 …… 106
5.3 推荐读物和网址 …… 123
5.4 关键术语、思考题和习题 …… 123
附录 5A 系数在 GF(2^8)中的多项式 …… 125
第 6 章 对称密码 …… 128
6.1 三重 DES 算法 …… 128
6.2 Blowfish 算法 …… 132
6.3 RC5 算法 …… 136
6.4 高级对称分组密码的特点 …… 140
6.5 RC4 流密码 …… 141
6.6 推荐读物和网址 …… 144
6.7 关键术语、思考题和习题 …… 145
第 7 章 用对称密码实现保密性 …… 148
7.1 密码功能的设置 …… 148
7.2 传输保密性 …… 153
7.3 密钥分配 …… 154
7.4 随机数的产生 …… 160
7.5 推荐读物和网址 …… 165
7.6 关键术语、思考题和习题 …… 166

第二部分 公钥加密与 hash 函数

第 8 章 数论入门 …… 172
8.1 素数 …… 172
8.2 Fermat 定理和 Euler 定理 …… 174
8.3 素性测试 …… 177
8.4 中国剩余定理 …… 179
8.5 离散对数 …… 181
8.6 推荐读物和网址 …… 185
8.7 关键术语、思考题和习题 …… 186
第 9 章 公钥密码学与 RSA …… 188
9.1 公钥密码体制的基本原理 …… 188
9.2 RSA 算法 …… 195
9.3 推荐读物和网址 …… 203
9.4 关键术语、思考题和习题 …… 204

附录 9A　算法复杂性 …… 207
第 10 章　密钥管理和其他公钥密码体制 …… 210
10.1　密钥管理 …… 210
10.2　Diffie-Hellman 密钥交换 …… 215
10.3　椭圆曲线算术 …… 218
10.4　椭圆曲线密码学 …… 224
10.5　推荐读物和网址 …… 227
10.6　关键术语、思考题和习题 …… 228
第 11 章　消息认证和 hash 函数 …… 231
11.1　对认证的要求 …… 231
11.2　认证函数 …… 232
11.3　消息认证码 …… 241
11.4　hash 函数 …… 243
11.5　hash 函数和 MAC 的安全性 …… 248
11.6　推荐读物 …… 251
11.7　关键术语、思考题和习题 …… 251
附录 11A　生日攻击的数学基础 …… 253
第 12 章　hash 算法 …… 258
12.1　MD5 消息摘要算法 …… 258
12.2　安全 hash 算法 …… 265
12.3　RIPEMD-160 …… 272
12.4　HMAC …… 278
12.5　推荐读物和网址 …… 281
12.6　关键术语、思考题和习题 …… 282
第 13 章　数字签名和认证协议 …… 284
13.1　数字签名 …… 284
13.2　认证协议 …… 287
13.3　数字签名标准 …… 293
13.4　推荐读物 …… 295
13.5　关键术语、思考题和习题 …… 295

第三部分　网络安全应用

第 14 章　认证的实际应用 …… 300
14.1　Kerberos …… 300
14.2　X.509 认证服务 …… 314
14.3　推荐读物和网址 …… 320
14.4　关键术语、思考题和习题 …… 321
附录 14A　Kerberos 加密技术 …… 322

第 15 章　电子邮件安全 …… 325
15.1　PGP …… 325
15.2　S/MIME …… 338
15.3　推荐网址 …… 351
15.4　关键术语、思考题和习题 …… 351
附录 15A　用 ZIP 压缩数据 …… 352
附录 15B　基数 64 转换 …… 354
附录 15C　PGP 随机数生成 …… 355
第 16 章　IP 安全性 …… 358
16.1　IP 安全性概述 …… 358
16.2　IP 安全体系结构 …… 360
16.3　认证头 …… 364
16.4　封装安全载荷 …… 367
16.5　安全关联组合 …… 371
16.6　密钥管理 …… 373
16.7　推荐读物和网址 …… 380
16.8　关键术语、思考题和习题 …… 381
附录 16A　互联网络和互联网协议 …… 382
第 17 章　Web 安全性 …… 389
17.1　Web 安全性思考 …… 389
17.2　安全套接层和传输层的安全 …… 391
17.3　安全电子交易 …… 404
17.4　推荐读物和网址 …… 412
17.5　关键术语、思考题和习题 …… 413

第四部分　系统安全性

第 18 章　入侵者 …… 416
18.1　入侵者 …… 416
18.2　入侵检测 …… 418
18.3　口令管理 …… 427
18.4　推荐读物和网址 …… 434
18.5　关键术语、思考题和习题 …… 435
附录 18A　基于比率的错误 …… 436
第 19 章　恶意软件 …… 440
19.1　病毒及相关的威胁 …… 440
19.2　计算机病毒的防治策略 …… 449
19.3　推荐读物和网址 …… 452
19.4　关键术语、思考题和习题 …… 453

第 20 章　防火墙 …… 454
20.1　防火墙的设计原理 …… 454
20.2　可信系统 …… 463
20.3　推荐读物和网址 …… 467
20.4　关键术语、思考题和习题 …… 468
附录 A　标准和标准化组织 …… 469
A.1　标准的重要性 …… 469
A.2　标准和规则 …… 469
A.3　互联网标准和国际互联网协会 …… 470
A.4　美国标准与技术研究所 …… 473
A.5　本书引用的标准和说明 …… 473
附录 B　用于密码编码学与网络安全教学的项目 …… 475
B.1　研究项目 …… 475
B.2　编程项目 …… 475
B.3　阅读/报告作业 …… 476
术语表 …… 477
参考文献 …… 482

第1章 引 言

最近几十年中，企业对**信息安全**的需求经历了两个重要变革。在广泛使用数据处理设备之前，企业主要是依靠物理和行政手段来保证重要信息的安全。采用的物理手段如将重要的文件放在上锁的文件柜里，采用的行政手段如对雇员的检查制度。

很显然，由于计算机的应用，需要有自动工具来保护存于计算机中的文件和其他信息。对于共享系统，如时间共享系统，以及通过公共电话网、数据网或 Internet 可访问的系统，尤其如此。用来保护数据和阻止黑客的工具一般称为**计算机安全**。

影响安全的第二个变革是，分布式系统、终端用户与计算机之间以及计算机与计算机之间传送数据的网络和通信设施的应用。在信息传输时，需要有网络安全措施来保护数据传输。事实上，术语**网络安全**容易引起误解，因为实际上所有的商业、政府和学术组织都将其数据处理设备与互联网相连，该互联网称为 internet[①]，并使用术语 **internet 安全**。

上述两种形式的安全没有明确的界限。例如，对信息系统最常见的攻击就是计算机病毒，它可能已先感染磁盘，然后才加载到计算机上，从而进入系统；也可能是通过 internet 进入系统。无论是哪一种情况，一旦病毒驻留在计算机系统中，就需要内部的计算机安全工具来检查病毒并恢复数据。

本书主要讨论 internet 的安全，包括阻止、防止、检测和纠正信息传输中出现的安全问题的措施，所涉及的内容相当广泛。为使读者对本书中讨论的领域有所了解，我们先举出几个有关安全问题的例子：

1. 用户 A 向用户 B 传送文件，该文件包含不能泄密的敏感信息（如工资单），用户 C 无权读取文件，但能够监视传输过程并截获该文件。
2. 网络管理员 D 向计算机 E 传输一条消息，命令计算机 E 更新权限文件以允许新用户可访问 E。用户 F 截获并修改该消息，如增加或删除一些用户，然后将消息转发给 E，而 E 误以为是管理员 D 发来的消息并更新权限文件。
3. 用户 F 也可以不截获消息，而是按自己的意愿构造消息并发送给 E，同样 E 误以为是管理员 D 发来的消息并更新权限文件。
4. 雇员事先未得到警告就被解雇。管理人员向服务器系统发送消息以注销该雇员的账号。账号注销后，服务器将通知贴到该雇员的文件中以确认注销。该雇员可以截获并延时这条消息，直至他有足够的时间访问服务器来获取敏感信息，然后再转发这条消息，以确认注销账号。雇员的这些活动在相当长的时间内不会被察觉。
5. 顾客向股票经纪人发送消息，请求完成各种交易。后来，这些投资失败而顾客否认发送过该消息。

---

① internet（"i"小写时）是指任何网络互联，如企业内部网就是 internet 的一个示例；Internet（"I"大写时）是指企业构造其 internet 时所使用的设施之一。

尽管上述举例不能穷尽所有可能的安全威胁类型，但足以表明网络安全所关注的范围。Internet 的安全同样错综复杂。理由如下：

1. 安全涉及到通信和网络，它不是像初次接触这个领域的人想像的那样简单。对网络安全的要求看起来似乎很明显。的确，对安全服务的绝大多数要求都可用自明其意的词语来描述，如保密性、认证、真实性和完整性等。但是，实现满足这些要求的安全机制却非常复杂。要想理解这些安全机制，需要进行缜密的推理。
2. 倘若让你设计一个安全机制或算法，你必须考虑各种各样的潜在攻击。很多情况下，从一个与设计完全不同的角度和方法出发，可能使攻击成功。这些方法都利用了你设计的机制中存在的意想不到的弱点。
3. 根据第二点，设计安全机制的过程通常采用逆向思维：不是从对安全性的要求出发来确定需要哪些安全措施，而是从可能有哪些攻击方法出发来确定需要哪些安全措施。
4. 倘若已经设计好了安全机制，接下来就是要确定在哪里使用这些安全机制，包括物理位置（在网络的什么地方）和逻辑位置（如像 TCP/IP 这样的网络协议的哪一层）。
5. 安全机制所使用的算法和协议通常不止一个。这些协议和算法需要通信双方使用一些秘密信息（如加密密钥），这就出现了对秘密信息的产生、分配、保护等问题。它们所依赖的通信协议可能会使得设计安全机制的过程复杂化。例如，安全机制需要对通信时间进行限制，而任何协议和网络都存在不确定且不可预知的通信时延，因此可能使这种限制毫无意义。

因此，需考虑的问题还有很多，这一章概述本书所要讨论的主要问题。首先讨论网络安全、服务和机制以及对网络的攻击类型，然后给出安全设施和安全机制的一般模型。

1.1 服务、机制和攻击

为了对系统的安全需求进行评估以及评价各种有关安全的产品和政策，负责安全的主管人需要用一些系统的方法来定义对安全的要求以及描述如何满足这些要求。一种方法是考虑信息安全的三个方面：

- **安全攻击**：任何危及系统信息安全的活动。
- **安全机制**：用来保护系统免受侦听、阻止安全攻击及恢复系统的机制。
- **安全服务**：加强数据处理系统和信息传输的安全性的一种服务。其目的在于利用一种或多种安全机制阻止安全攻击。

1.1.1 服务

我们先简单考虑一下上述三个方面，首先我们讨论安全服务。我们可以想到一些信息安全服务，如正常的与物理文本有关的某些服务。大多数人类活动，如商业、外交、军事以及人际交往等，都使用了文本，并且依赖交易双方对文本完整性的信赖。通常文件都有签名和日期，同时为防止它们被泄漏、篡改或破坏，要有公证和现场见证人，要被记录或被允许访问，等等。

因为信息系统已经越来越深入到我们的日常生活，电子信息在很多方面已经取代了传统的纸文本的作用。然而，以下这些方面使得电子信息有时不太方便：

1. 一般说来，要区分出纸文本的原件和复印件是可能的，然而电子信息只不过是一些二进制位串，无法区分所谓的原件和复印件。
2. 更改纸文本必然会留下一些物理痕迹，比如擦除可能导致表面粗糙或留下一个小槽，而在内存中改变一些二进制位却不会留下任何物理痕迹。
3. 所有与纸文本有关的证据都来自于文本本身的物理特征，比如手写签名、阴文或阳文的公证印章，等等；而电子信息若要进行此类认证，只能依靠本身所记录的二进制信息。

表 1.1 列举了传统纸文本的一些常见功能以及电子信息类似的功能。这些功能实际上是我们对电子信息所提出的安全性要求。

表 1.1 的内容太长，其本身不适合指导我们设计安全机制。计算机和网络安全的研究与开发着重于一些通用的安全服务，包括信息安全机制所要求的各种功能。我们将在下一节中探讨这些问题。

表 1.1　部分常见的信息完整性功能［SIMM92］

• 身份证明	• 签注
• 认证	• 访问(外出权)
• 许可与/或证书	• 确认
• 签名	• 发生时间
• 见证人(公证)	• 认证软件——软件与/或文件
• 同时发生	• 投票
• 责任	• 所有权
• 收条	• 登记
• 发送方与/或接收方的证书	• 赞成/否决
	• 隐秘(秘密)

1.1.2　机制

没有哪一种安全机制能提供上述的所有安全服务。本书将讲述几种安全机制。然而，我们会注意到，许多安全机制都包含一种特殊的技术：密码技术。加密或类加密信息传输(如 hash 函数)是提供安全性最常用的手段。因此，本书集中讨论了密码技术的发展、应用和管理。

1.1.3　攻击

正如 G. J. Simmons 所指出的，信息安全是指如何阻止对信息系统中物理存在的信息的攻击，或在阻止失败后如何检测这些攻击并从中恢复信息，其中这些信息本身是没有意义的物理存在[SIMM92]。

表 1.2 列举了一些攻击的例子，在现实世界中有许多这种攻击，即组织或个人（或代表其雇员的组织）都需要对付的攻击。对组织的攻击随着环境的变化而变化。幸运的是，我们能够通过观察可能遇到的攻击类型，从不同的角度来处理该问题。这就是下一节所要讨论的主题。

表 1.2 安全性攻击举例[SIMM92]

1. 获取对信息的非授权访问(即侵犯秘密性或隐秘性)。
2. 冒充别的用户以推卸责任或使用他人的许可证,以达到以下目的:
   a. 制造欺诈信息。
   b. 篡改合法信息。
   c. 使用欺诈性的身份来获得非授权访问。
   d. 欺诈性地对交易进行认证或签注。
3. 抵赖欺诈引起的责任。
4. 声称已经收到了别的用户的信息,其实这是由他自己伪造的(即责任的欺诈属性)。
5. 声称已经(在某个特定时间)将信息发送给接收方,而在当时他根本没有发送这条信息(或是在不同的时间发送的)。
6. 否认接收到的信息或接收信息的时间。
7. 扩大他的合法权限(如访问、创建、分发,等等)。
8. (未经授权)修改他人的权限(欺诈性地登录、限制或扩大他人的权限)。
9. 隐藏某些信息(秘密通信)于其他通信(公开通信)之中。
10. 将自身作为中继插入到其他用户的通信链路中。
11. 了解谁在什么时候访问了什么信息(资源、文件等),即使信息本身未被泄露,也可进行访问(例如,归纳总结分析从通信信道到数据库、软件等的流量)。
12. 通过揭示欺诈者以为(根据协议)仍然保密的信息来对信息完整性协议提出质疑。
13. 使软件功能反常,通常是加入一个秘密函数。
14. 通过加入错误信息使他人侵犯协议。
15. 让系统明显失败以让人们失去信心。
16. 阻止其他用户间的通信,特别是通过秘密介人,使合法通信被拒绝。

应该注意,在文献中,术语**威胁**和**攻击**的意义通常是相同的。表 1.3 给出了 RFC 2828(Internet Security Glossary)提供的定义。

表 1.3 威胁和攻击(RFC 2828)

威胁

侵犯安全的可能性,在破坏安全或引起危害的环境、可能性、行为或事件的情况下,会出现这种威胁。也就是说,威胁是利用脆弱性的潜在危险。

攻击

对系统安全的攻击,它来源于一种具有智能的威胁;也就是说,有意违反安全服务和侵犯系统安全策略的(特别是在方法或技巧的)智能行为。

1.2 OSI 安全框架

为了有效评价一个机构的安全需求,以及对安全产品和政策进行评估和选择,负责安全的管理员需要某种系统的方法来定义对安全的要求并刻画满足这些要求的措施。在集中式数据处理环境下做到这一点非常困难。随着局域网和广域网的使用,问题变得更加复杂。

ITU-T[①] 推荐方案 X.800,即 OSI 安全框架,定义了这样一种系统方法。OSI 安全框架对管理员来说是提供安全的一种组织方法。而且,这个框架是作为国际标准开发的,计算机和电信销售商已经在他们的产品和服务上开发了这些安全特性,这些产品和服务与服务和安全机制的结构化定义有关联。

对我们来说,OSI 安全框架对本书将要涉及的许多概念提供了一个有用的(可能是抽象的)概貌。该框架主要关注安全服务、机制和攻击。

① 国际电信联盟(ITU)电信标准化机构是联合国资助的制定这些标准的机构,其提出的所谓推荐标准涉及远程通信和开放式系统互联(OSI)。

1.2.1 安全服务

X.800 将安全服务定义为通信开放系统协议层提供的服务，从而保证系统或数据传输有足够的安全性。也许在 RFC 2828 中可找到一种更清楚的定义：一种由系统提供的对系统资源进行特殊保护的处理或通信服务；安全服务通过安全机制来实现安全策略。X.800 将这些服务分为五类共十四个特定服务(表 1.4)。我们下面将逐类进行讨论[①]。

表 1.4 安全服务(X.800)

认 证	**数据完整性**
保证通信的实体是它所声称的实体。	保证收到的数据确是授权实体所发出的数据(即没有修改、插入、删除或重放)。
同等实体认证	**具有恢复功能的连接完整性**
用于逻辑连接时为连接的实体的身份提供可信性。	提供一次连接中所有用户数据的完整性，检测整个数据序列内存在的修改、插入、删除或重放，且试图恢复之。
数据源认证	**无恢复的连接完整性**
在无连接传输时保证收到的信息来源是声称的来源。	同上，但仅提供检测，无恢复。
存取控制	**选择域连接完整性**
阻止对资源的非授权使用(即这项服务控制谁能存取资源，在什么条件下可以存取，这些存取的资源可用于做什么)。	提供一次连接中传输的单个数据块用户数据中选定部分的数据完整性，并判断选定域是否有修改、插入、删除或重放。
数据保密性	**无连接完整性**
保护数据免于非授权泄漏。	为单个无连接数据块提供完整性保护，并检测是否有数据修改。另外，提供有限的重放检测
连接保密性	**选择域无连接完整性**
保护一次连接中所有的用户数据。	为单个无连接数据块内选定域提供完整性保护；判断选定域是否被修改。
无连接保密性	**不可否认性**
保护单个数据块里的所有用户数据。	防止整个或部分通信过程中，任一通信实体进行否认的行为
选择域保密性	**源不可否认**
对一次连接或单个数据块里选定的数据部分提供保密性。	证明消息是由特定方发出的。
流量保密性	**宿不可否认性**
保护那些可以通过观察流量而获得的信息。	证明消息被特定方收到。

认证

认证服务与保证通信的真实性有关。在单条消息的情况下，如一条警告或报警信号，认证服务功能是向接收方保证消息来自所声称的发送方。对于正在进行的交互，如终端和主机连接，就涉及两个方面。首先，在连接的初始化阶段，认证服务保证两个实体是可信的，也就是说，每个实体都是他们所声称的实体。其次，认证服务必须保证该连接不受第三方如下方式的干扰：第三方能够伪装成两个合法实体中的一个进行非授权传输或接收。

该标准还定义了两个特殊的认证服务：

- **同等实体认证**：为连接中的同等实体提供身份确认，用于连接的建立或数据传输阶段。该服务想提供这样的保证：一个实体不能试图进行伪装或以前连接的非授权重放。

① 信息安全文献中使用的许多术语尚未达成广泛的一致。例如，完整性有时是指信息安全性，认证有时既用来指实体的验证，又用来指下面列出的关于完整性的各种函数。我们这里使用的术语与 X.800 和 RFC 2828 是相一致的。

- **数据源认证**:为数据的来源提供确认,但对数据的复制或修改不提供保护。这种服务支持电子邮件的如下应用:在这种应用的背景下,通信实体间没有预先的交互。

存取控制

在网络安全中,存取控制是一种限制、控制那些通过通信连接对主机和应用进行存取的能力。为此,每个试图获得存取控制的实体必须被识别或认证后,才能获取其相应的存取权限。

数据保密性

保密性是防止传输的数据遭到被动攻击(定义见后)。关于数据传输,可以有几层保护。最广泛的服务在一段时间内为两个用户间所传输的所有用户数据提供保护。例如,如果两个系统间建立了 TCP 连接,则这种广泛的保护将阻止在 TCP 连接上传输的任何用户数据的泄漏。也可以定义一种较窄的保密性服务,可以是对单条消息或对单条消息内某个特定的范围提供保护。这种细化比起广泛的方法用处要少,而且实现起来更复杂、更昂贵。

保密性的另一个方面是防止流量分析。这要求攻击者不能观察到消息的源和宿、频率、长度或通信设施上的其他流量特征。

数据完整性

与保密性相比,完整性可应用于消息流、单条消息或消息的选定部分。同样,最有用也最直接的方法是对整个数据流提供保护。

处理消息流的面向连接的完整性服务保证收到的消息和发出的消息一致,没有复制、插入、修改、更改顺序或重放。该服务也涉及对数据的破坏。因此,面向连接的完整性服务处理消息流的修改和拒绝服务两个问题。另一方面,仅仅处理单条消息而不管大量信息的无连接的完整性服务,通常仅仅防止对单条消息的修改。

我们可以区分有恢复和无恢复的服务。因为完整性服务和主动攻击有关,我们更关心检测而不是阻止攻击。如果检测到完整性遭破坏,那么服务可以简单地报告这种破坏,并通过软件的其他部分或人工干预来恢复被破坏的部分。另外,我们下面会看到,有些机制可用来恢复数据完整性。通常,自动恢复机制是一种更具吸引力的选择。

不可否认性

不可否认性防止发送方或接收方否认传输或接收过某条消息。因此,当消息发出后,接收方能证明消息是由声称的发送方发出的。同样,当消息接收后,发送方能证明消息事实上确实由声称的接收方收到。

可用性服务

X.800 和 RFC 2828 都将可用性定义为:根据系统的性能说明,能够按授权的系统实体的要求存取或使用系统或系统资源的性质(即当用户请求服务时,根据系统设计,若系统提供这些服务,则系统是可用的)。许多攻击可导致可用性的损失或减少。一些自动防御措施,如认证、加密,可对付某些攻击。面其他的一些攻击需要一些物理措施来阻止或恢复分布式系统中

损失的可用性。

X.800 将可用性看做是和各种安全服务相关的性质。但是，单独说明可用性服务是颇有意义的。可用性服务使系统确保可用性。这种服务处理由拒绝服务攻击引起的安全问题。它依赖于对系统资源的恰当管理和控制，因此依赖于存取控制服务和其他安全服务。

1.2.2　安全机制

表 1.5 列出了 X.800 中定义的安全机制。由表可知，这些安全机制可分成两类：一类在特定的协议层实现，一类不属于任何的协议层或安全服务。本书后面将讨论这些机制，在此除了讨论加密的定义外，我们不详论之。X.800 区分可逆和不可逆加密机制。可逆加密机制是一种简单的加密算法，使数据可以加密和解密。不可逆加密机制包括 hash 算法和消息认证码，用于数字签名和消息认证应用。

表 1.5　安全机制(X.800)

特定安全机制
可以嵌入合适的协议层以提供一些 OSI 安全服务。
加密
运用数学算法将数据转换成不可知的形式。数据的变换和复原依赖于算法和零个或多个加密密钥。
数字签名
附加于数据元之后的数据，是对数据元的密码变换，以使得(如接收方)可证明数据源和完整性，并防止伪造
存取控制
对资源行使存取控制的各种机制。
数据完整性
用于保证数据元或数据元流的完整性的各种机制。
认证交换
通过信息交换来保证实体身份的各种机制。
流量填充
在数据流空隙中插入若干位以阻止流量分析。
路由控制
能够为某些数据选择特殊的物理上安全的路线并允许路由变化(尤其是在怀疑有侵犯安全的行为时)。
公证
利用可信的第三方来保证数据交换的某些性质。
普遍的安全机制
不局限于任何 OSI 安全服务或协议层的机制。
可信功能
据某些标准被认为是正确的(例如，根据安全策略所建立的标准)。
安全标签
资源(可能是数据元)的标志，指明该资源的安全属性。
事件检测
检测与安全相关的事件。
安全审计跟踪
收集的及潜在用于安全审计的数据，它是对系统记录和行为的独立回顾和检查。
安全恢复
处理来自安全机制的请求，如事件处理、管理功能和采取恢复行为。

基于 X.800 中的定义，表 1.6 给出了安全服务和安全机制的关系。

表 1.6 安全服务与机制间的关系

服务	机制							
	加密	数字签名	访问控制	数据完整性	认证交换	流量填充	路由控制	公证
同等实体认证	Y	Y			Y			
数据源认证	Y	Y						
访问控制			Y					
保密性	Y						Y	
流量保密性	Y					Y	Y	
数据完整性	Y	Y		Y				
不可否认性		Y		Y				Y
可用性				Y	Y			

1.2.3 安全性攻击

X.800 和 RFC 2828 都使用了一种有用的方式来对安全性攻击进行分类，即被动攻击和主动攻击。被动攻击试图了解或利用系统的信息但不影响系统资源。主动攻击试图改变系统资源或影响系统运作。

被动攻击

被动攻击的特性是对传输进行窃听和监测。攻击者的目标是获得传输的信息。信息内容泄漏和流量分析就是两种被动攻击。

信息内容泄漏攻击很易理解。电话、电子邮件消息和传输的文件都可能含有敏感或秘密 HTH 的信息。我们希望能阻止攻击者了解传输的内容。

第二种被动攻击是**流量分析**。设想我们已有一种方法来隐藏消息内容或其他信息的流量，使得攻击者即使捕获了消息也不能从消息里获得信息。加密是隐藏内容的常用技巧。即使我们恰当地进行了加密保护，攻击者仍可能获得这些消息模式。攻击者可以决定通信主机的身份和位置，可以观察传输的消息的频率和长度。这些信息可以用于判断通信的性质。

被动攻击由于不涉及对数据的更改，所以很难察觉。然而，通过加密的手段阻止这种攻击却是可行的。因此处理被动攻击的重点是预防，而不是检测。

主动攻击

主动攻击包括对数据流进行篡改或伪造数据流，可分为四类：伪装、重放、消息篡改和拒绝服务。

伪装是指某实体假装别的实体。伪装攻击的例子有：捕获认证信息，并在其后利用认证信息进行重放，这样它就可能获得其他实体所拥有的权限。

重放是指将获得的信息再次发送以在非授权情况下进行传输。

消息篡改是指修改合法消息的一部分或延迟消息的传输以获得非授权作用。例如，将消息“Allow John Smith to read confidential file *accounts*”修改为“Allow Fred Brown to read confidential file *accounts*”。

拒绝服务阻止或禁止正常的使用或管理通信设施。这种攻击可能有具体的目标。比如，某实体可能会查禁所有发向某目的地的消息。拒绝服务的另一种形式是破坏某实体网络，它或者是使网络失效，或者是使其过载以降低其性能。

主动攻击与被动攻击相反。被动攻击虽然难以被检测到但可以防止，而主动攻击却难以防止(因为这样做需要保护所有的物理通信设施)，但容易检测，所以重点在于检测并从破坏中恢复。因为检测主动攻击有一种威慑效果，所以也可在某种程度上阻止主动攻击。

1.3 网络安全模型

我们要讨论的大多通信模型如图 1.1 所示，通信一方要通过 internet 将消息传送给另一方，那么通信双方(称为交易的主体)必须协调努力共同完成消息交换。我们可以通过定义 internet 上从源到宿的路由以及通信主体共同使用的通信协议(如 TCP/IP)来建立逻辑信息通道。

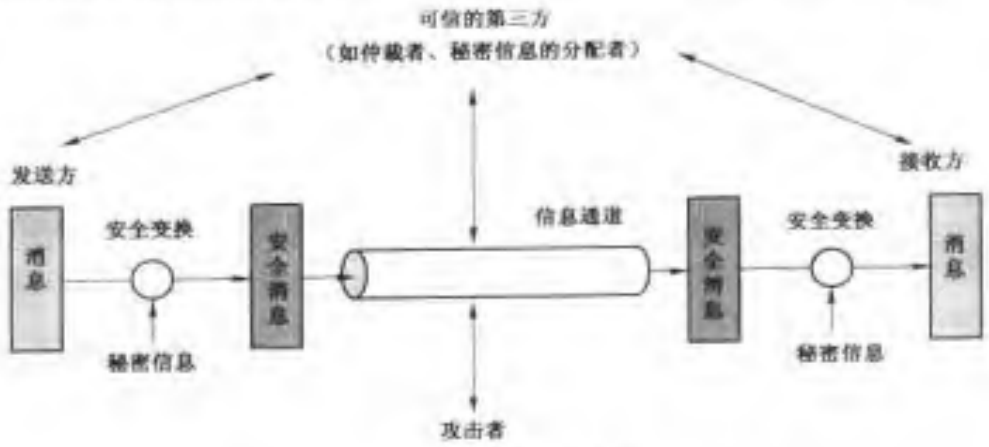


图 1.1 网络安全模型

在需要保护信息传输以防攻击者危害信息的保密性、真实性的时候，就会涉及信息安全。任何用来保证安全的方法都包含两个方面：

- 被发送信息的相关安全变换。如对消息加密，它打乱消息使得攻击者不能读懂消息，或者将基于消息的编码附于消息后，用于验证发送方的身份。
- 双方共享某些秘密信息，并希望这些信息不为攻击者所知。如接收方在变换和恢复消息之前用来打乱消息的加密密钥[①]。

为了实现安全传输，需要有可信的第三方。例如，第三方负责将秘密信息分配给通信双方，而对攻击者保密，或者当通信双方关于信息传输的真实性发生争执时，由第三方来仲裁。

上述模型说明，设计安全服务应包含下列四个方面的内容：

1. 设计执行安全相关传输的算法。该算法应是攻击者无法攻破的。
2. 产生算法所使用的秘密信息。

① 第二部分讨论的公钥密码中，只需发送方或者接收方拥有秘密信息。

3. 设计分配和共享秘密信息的方法。

4. 指明通信双方使用的协议，该协议利用安全算法和秘密信息实现安全服务。

本书主要讨论的是安全机制和服务，它们遵循图 1.1 所示的模型。但是，还有其他与安全有关的情形不完全符合该模型，本书将讨论这些内容，它们的一般模型如图 1.2 所示，该模型希望保护信息系统不受有害的访问。大多数读者都熟悉黑客引起的问题，黑客试图渗入到通过网络可访问的系统，他们可能没有恶意，而只是满足于闯入或进入计算机系统的人。入侵者可能是想进行破坏的雇员，或者是想利用计算机获利的人(如获取信用卡号或者进行非法的资金转账)。

另一种类型的有害访问是在计算机系统中加入程序，它利用系统的弱点来影响应用程序和实用程序，如编辑程序和编译程序。程序引起的威胁有两种：

- **信息访问威胁**。以非授权用户的名义截获或修改数据。
- **服务威胁**。利用计算机中的服务缺陷禁止合法用户使用这些服务。

病毒和蠕虫是两种软件攻击，这些攻击可通过含有隐藏在有用的软件中的有害程序的磁盘进入系统，也可以通过网络进入系统。网络安全更关心的是通过网络进入系统的攻击。

对付有害访问所需的安全机制分为两大类(见图 1.2)。第一类称为门卫功能，它包含基于口令的登录过程，该过程只允许授权用户的访问并检测和拒绝蠕虫、病毒等攻击的扫描程序访问系统，一旦非法用户或软件获得了访问权，那么应有各种监视活动和分析存储信息的内部控制机制来检测非法入侵者。

图 1.2 网络访问安全模型

1.4 本书概览

本章是全书的导引。书的其余部分分成四部分：

第一部分。给出了对称加密的综述，包括古典算法和现代算法，着重讨论了 DES(数据加密标准)和 AES(高级加密标准)这两个重要算法。

第二部分。给出了公钥算法的综述，包括 RSA 和椭圆曲线，也讨论了公钥密码的应用，包括数字签名和密钥交换。

第三部分。讨论了那些用于网络和 Internet 的密码算法和安全协议。主要包括用户认证、电子邮件、IP 安全和 Web 安全。

第四部分。讨论了用于保护计算机系统的一些安全设施，使系统免受攻击，如入侵、病毒和蠕虫，还讨论了防火墙技术。

本书中给出的许多密码算法、网络安全协议和应用都已经被指定为标准。其中最重要的是在 RFC 中定义的 Internet 标准和由美国国家标准技术研究(NIST)所发布的联邦信息处理标准(FIPS)。附录 A 讨论了标准的制定过程，列出了本书引用的标准。

1.5 推荐读物

[PFLE97]是计算机和网络安全的很好的入门书。[NICH99]是一本优秀的综述性书籍。[SCHN00]对从事计算机和网络安全的实践者来说是很有价值的读物。该书讨论了技术和密码学的局限性，特别是密码学在提供安全方面的局限性，还讨论了对硬件和软件实现、网络及从事安全与攻击的人员方面的要求。

NICH99 Nichols, R. ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.

PFLE97 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York: Wiley 2000.

1.6 Internet 和 Web 资源

有许多 Internet 和 Web 资源对本书提供支持，以帮助读者跟上该领域的最新发展。

1.6.1 为本书提供的 Web 站点

为本书建立的网页是 WilliamStallings.com/Crypto3e.html。该站点包括如下内容：

- **有用的 web 站点**：含有指向其他相关站点的链接。
- **勘误表**：据需要维护和更新本书的勘误表。请把你发现的错误以电子邮件方式寄给我们。
- **图**：使用 PDF 格式存储的本书中的图。
- **表**：使用 PDF 格式存储的本书中的表。
- **幻灯片**：按章组织的 PowerPoint 幻灯片。
- **Internet 邮件列表**：本站点包括该书的 Internet 邮件列表的签收信息。
- **密码学和网络安全课程**：含有指向基于本书的课程主页，这些主页对一些老师如何安排他们的课程计划有帮助。

我还维护了一个计算机科学学生资源站点，位于 WilliamStallings.com/StudentSupport.html。目的是为计算机科学的师生提供文档、信息和链接。链接和文档分成四类：

- **数学**：包括一些基本的数学复习，排队分析初步，数制系统初步，以及一些指向其他数学站点的连接。
- **如何做**：在课后作业、撰写技术报告、准备技术演讲等方面提供建议和指导。
- **研究资源**：指向论文集、技术报告和目录的链接。
- **其他**：其他一些有用的文档和链接。

1.6.2 其他站点

有许多站点提供和本书有关的信息。在后续章节，指向特定站点的链接可以在推荐读物和网址中找到。由于网址经常变化，本书没有包括这些地址。在本书的网页上可找到书中列出的网址的链接。书中没有提及的链接也可能会添加到本书的网页上。

如下的一些站点与密码学和网络安全有关：

- **COAST**：有关密码学和网络安全的链接。
- **IETF 安全域**：有关网络安全的标准化方面的材料。
- **计算机和网络安全参考索引**：它给出了销售商、商业产品、常见问题、新闻组文档、论文和其他站点的索引。
- **密码学常见问题**：覆盖密码学所有领域的较长较有价值的常见问题。
- **Tom Dunigan 的安全主页**：指向密码学和网络安全站点的列表。
- **IEEE 安全保密技术委员会**：含有其新闻和与 IEEE 有关的活动信息的副本。
- **计算机安全资源中心**：美国国家标准技术局维护；包含大量有关安全威胁、技术和标准的信息。

1.6.3 USENET 新闻组

大量的 USENET 新闻组致力于网络安全或密码学。事实上和所有的 USENET 新闻组一样，有很高的噪信比，但确实值得浏览，以了解是否能满足自己的需要。最相关的有如下一些：

- **sci.crypt.research**：最好的新闻组。这是一个受限的研究性新闻组；张贴的东西必须和密码技术有关。
- **sci.crypt**：有关密码学和相关主题的一般性讨论。
- **sci.crypt.random-numbers**：讨论密码学意义的随机性。
- **alt.security**：安全主题的一般性讨论。
- **comp.security.misc**：计算机安全主题的一般性讨论。
- **comp.security.firewalls**：对防火墙和技术的讨论。
- **comp.security.announce**：CERT 发布的新闻和公告。
- **comp.risks**：讨论计算机和用户对公众的威胁。
- **comp.virus**：一个受限的计算机病毒讨论组。

第一部分　对称密码

第一部分涉及的内容

迄今为止，确保网络与通信安全的最重要的工具是加密。广泛使用的两种加密形式是传统(或对称)加密和公钥(或非对称)加密。第一部分讨论了对称加密的基本原理、广泛使用的算法以及对称密码的应用。

第一部分内容浏览

第 2 章　传统加密技术

第 2 章描述了传统对称加密技术。对密码编码学和密码分析学进行了细致有趣的介绍，并着重强调了某些重要的概念。

第 3 章　分组密码与数据加密标准

第 3 章介绍了现代对称密码学的原理，讨论了广泛使用的加密技术：数据加密标准(DES)。本章还讨论了密码设计、密码分析学和大多数现代对称加密方案使用的基本结构：Feistel 密码。

第 4 章　有限域

有限域在密码学中变得日益重要。很多密码算法在很大程度上依赖于有限域的性质，其中比较著名的有高级加密标准(AES)和椭圆曲线密码。本章在讨论 AES 之前先介绍与 AES 有关的概念，为理解有限域 GF(2^n)上的运算提供必要的背景知识。

第 5 章　高级加密标准

近年来密码学上最重要的发展就是选取 AES 作为新的对称密码标准。第 5 章对这个密码进行了全面透彻的讨论。

第 6 章　对称密码

除了 DES 和 AES 外，还有许多其他广泛使用的对称密码。第 6 章讨论了其中最为重要的一些对称密码，并一般性地讨论了对称分组密码的设计原理。此外，第 6 章还讨论了对称流密码及其最为重要的一个例子：RC4。

第 7 章　用对称密码实现保密性

本章讨论对称加密算法的实际结构和应用对称加密确保机密性的一些设计问题。其中既包括端到端加密技术、链路加密技术、密钥分配技术，又包括随机数产生技术。

第 2 章　传统加密技术

对称加密，也称传统加密或单钥加密，是公钥密码产生之前惟一的一种加密技术[①]。迄今为止，它仍是两种类型的加密中使用最为广泛的一种。第一部分将讨论许多对称密码。本章中，我们首先介绍对称加密过程的一般模型，了解传统加密算法的使用环境。然后，我们讨论计算机出现之前的许多算法。最后，简要地介绍隐写术。第 3 章则详细讨论当今使用最广泛的加密算法：DES。

首先，我们来定义一些术语。原始的消息称为**明文**，而加密后的消息称为**密文**。从明文到密文的变换过程称为**加密**；从密文到明文的变换过程称为**解密**。研究各种加密方案的学科称为**密码编码学**，而加密方案则称为**密码体制**或**密码**。研究破译密码获得消息的学科称为**密码分析学**。密码分析学即外行所说的“破译”。密码编码学和密码分析学统称**密码学**。

2.1　对称密码的模型

对称加密方案有五个基本成分（见图 2.1）：

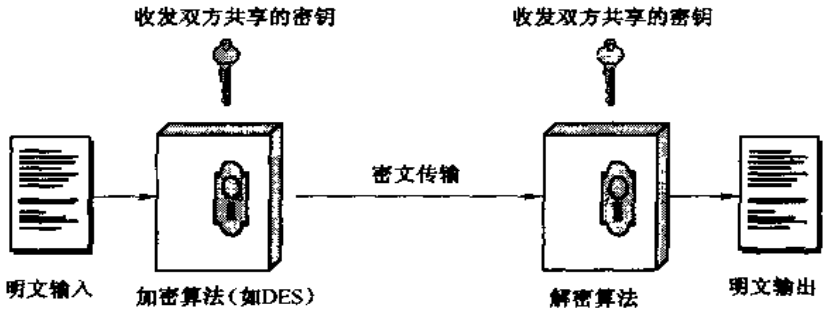


图 2.1　传统密码的简化模型

- **明文**：作为算法的输入，原始可理解的消息或数据。
- **加密算法**：加密算法对明文进行各种代换和变换。
- **密钥**：密钥也是加密算法的输入。密钥独立于明文。算法将根据所用的特定密钥而产生不同的输出。算法所用的代换和变换也依靠密钥。
- **密文**：作为算法的输出，看起来完全随机而杂乱的数据，依赖于明文和密钥。对于给定的消息，不同的密钥将产生不同的密文，密文是随机的数据流，并且其意义是不可理解的。
- **解密算法**：本质上是加密算法的逆。输入密文和密钥可以用解密算法恢复出明文。

---

① 公钥加密于 1976 年第一次在公开文献中提出，但是 NSA（National Security Agency）宣称他们几年前就发现了这种加密体制。

传统密码的安全使用要满足如下两个要求：

1. 加密算法必须是足够强的。至少，我们希望这个算法在敌手知道它并且能够得到一个或者多个的密文时也不能破译密文或计算出密钥。这个要求通常用一种更强的形式表述为：即使敌手拥有一定数量的密文和产生这些密文的明文，他(或她)也不能破译密文或发现密钥。
2. 发送者和接收者必须在某种安全的形式下获得密钥并且必须保证密钥的安全。如果有人发现该密钥，而且知道相应的算法，那么就能读出使用该密钥加密的所有通信内容。

我们假设基于已知密文和加密/解密算法的知识而能破译消息是不实际的。换句话说，我们并不需要保密算法，而仅需要保密密钥。对称密码的这些特点使其能够广泛地应用。算法不需要保密这一事实使得制造商可以开发出低成本的芯片，以实现数据加密算法。这些芯片能够广泛地使用、适用于大规模生产。因此，采用对称密码，首要的安全问题就是密钥的保密性。

我们从图 2.2 中可以更清楚地理解对称加密方案的基本成分。发送方产生明文消息 $X=[X_1, X_2, \cdots, X_M]$，$X$ 的 M 个元素是某个字母表中的字母。一般地，字母表由 26 个大写字母组成。而现在，最常用的是基于二进制字母表{0,1}的二进制串。加密的时候，先产生一个形如 $K=[K_1, K_2, \cdots, K_J]$的密钥。如果密钥是由信息的发送方产生的，那么它要通过某种安全渠道发送到接收方；另一种方法是由第三方生成密钥后再安全地分发给发送方和接收方。

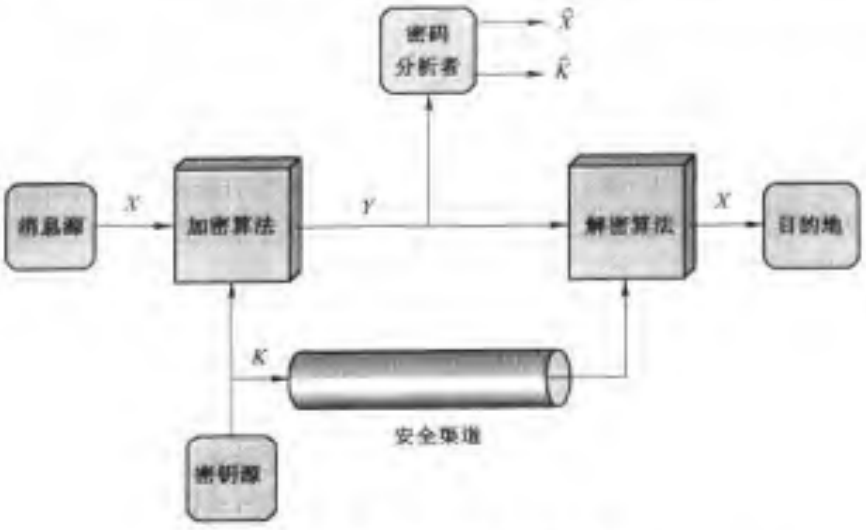


图 2.2　传统密码体制的模型

加密算法根据输入信息 X 和密钥 K 生成密文 $Y=[Y_1, Y_2, \cdots, Y_N]$，即

$$Y = \mathrm{E}_K(X)$$

该式表明密文 Y 是明文 X 的函数，而具体的函数由密钥 K 的值决定。

拥有密钥 K 的接收者，可以进行以下转换，以得到明文：

$$X = D_K(Y)$$

假设某敌手窃得 Y 但是并不知道 K 或 X，而企图得到 K 或 X，或 K 和 X。假设他知道加密算法 E 和解密算法 D，如果他只是对某些特定信息感兴趣，那么他将注意力集中在计算明文的估计值 $\hat{X}$ 来恢复 X；不过，攻击者往往对进一步的信息同样有兴趣，这种情况下，他企图通过计算密钥的估计值 $\hat{K}$ 来恢复 K。

2.1.1 密码编码学

密码编码学系统具有以下三个独立的特征：

1. **转换明文为密文的运算类型**。所有的加密算法都基于两个原理：代换和置换。代换是将明文中的每个元素（如位、字母、位组或字组等）映射成另一个元素；置换是将明文中的元素重新排列。上述运算的基本要求是不允许有信息丢失（即所有的运算都是可逆的）。大多数密码体制都使用了多层代换和置换。
2. **所用的密钥数**。如果发送方和接收方使用相同的密钥，这种密码就称为对称密码、单密钥密码或传统密码。如果发收双方使用不同的密钥，这种密码就称为非对称密码、双钥密码或公钥密码。
3. **处理明文的方法**。分组密码每次处理一个输入分组，相应地输出一个输出分组。而流密码则是连续地处理输入元素，每次输出一个元素。

2.1.2 密码分析学

攻击传统的密码体制有两种一般的方法：

- **密码分析学**：密码分析学的攻击依赖于算法的性质和明文的一般特征或某些明密文对。这种形式的攻击企图利用算法的特征来推导出特别的明文或使用的密钥。如果这种攻击能成功地推导出密钥，那么影响将是灾难性的：将会危及所有未来和过去使用该密钥加密消息的安全。
- **穷举攻击**：攻击者对一条密文尝试所有可能的密钥，直到把它转化为可读的有意义的明文。平均而言，获得成功至少要尝试所有可能密钥的一半。

我们首先考虑密码分析学，然后介绍穷举攻击。

基于密码分析者知道信息的多少，表 2.1 概括了密码攻击的几种类型。表中惟密文攻击难度最大。有些情况下，攻击者甚至不知道加密算法，但是我们通常假设敌手知道。这种情况下，一种可能的攻击是试遍所有可能密钥的穷举攻击。如果密钥空间非常大，这种方法就不太实际。因此攻击者必须依赖于对密文本身的分析，而这一般要运用各种统计方法。使用这种方法，攻击者对隐含的明文类型必须有所了解，比如说明文是英文文本或法文文本、Windows 的 EXE 执行文件、Java 源列表文件、会计文件，等等。

表 2.1　基于加密信息的攻击类型

攻击类型	密码分析者已知的信息
惟密文攻击	• 加密算法 • 要解密的密文
已知明文攻击	• 加密算法 • 要解密的密文 • 用(与待解的密文)同一密钥加密的一个或多个明密文对
选择明文攻击	• 加密算法 • 要解密的密文 • 分析者任意选择的明文,用(与待解的密文)同一密钥加密的密文
选择密文攻击	• 加密算法 • 要解密的密文 • 分析者有目的选择的一些密文,用(与待解的密文)同一密钥解密的对应明文
选择文本攻击	• 加密算法 • 要解密的密文 • 分析者任意选择的明文,用(与待解的密文)同一密钥加密的对应密文 • 分析者有目的选择的一些密文,用(与待解的密文)同一密钥解密的对应明文

惟密文攻击是最容易防范的,因为攻击者拥有的信息量最少。不过在很多情况下,分析者可以得到更多的信息。分析者可以捕获到一段或更多的明文信息及相应密文,也可能知道某段明文信息的格式等。比如,按照 Postscript 格式加密的文件总是以相同的格式开头,电子金融消息往往有标准化的文件头或者标志等。这些都是已知明文攻击的例子。拥有这些知识的分析者就可以从转换明文的方法入手来推导出密钥。

与已知明文攻击紧密相关的是可能词攻击。如果攻击者处理的是一般散文信息,他可能对信息的内容一无所知,但是如果他处理的是一些特定的信息,他就可能知道其中的部分内容。比如说,对于一个完整的会计文件,攻击者可能知道放在文件最前面的是某些密钥词。又比如,某某公司开发的程序源代码可能含有该公司的版权信息,并且放在某个标准位置。

如果分析者能够通过某种方式,让发送方在发送的信息中插入一段由他选择的信息,那么选择明文攻击就有可能实现。一个例子是差分密码分析,这在第 3 章中将会讲到。一般说来,如果分析者有办法选择明文加密,那么他将故意选取那些最有可能恢复出密钥的数据。

表 2.1 还列举了另外两种类型的攻击方法:选择密文攻击和选择文本攻击。它们在密码分析技术中很少用到,但是不失为两种较好的可能攻击方法。

只有相对较弱的算法才抵挡不住密文攻击。一般地,加密算法起码要能经受得住已知明文攻击才行。

此外,还有两个概念值得注意。如果无论有多少可使用的密文,都不足以惟一地确定由该体制产生密文所对应的明文,则加密体制是**无条件安全的**。也就是说无论花多少时间,攻击者都无法将密文解密,这仅仅因为他(或她)没有所需要的信息。除了一次一密(在以后的章节中将会讲到)之外,所有的加密算法都不是无条件安全的。因此,加密算法的使用者应挑选尽量满足以下标准的算法:

- 破译密码的代价超出密文信息的价值。
- 破译密码的时间超出密文信息的有效生命期。

如果满足了上述两条标准则加密体制是**计算上安全的**。估计攻击者成功破译密文所需的工作量是非常困难的。

试遍所有密钥直到有一个合法的密钥能够把密文还原成明文，这就是穷举攻击。我们可以从这种方法入手，考虑其所需的时间代价。一般说来，要获得成功必须尝试所有可能密钥的一半。表 2.2 给出了对于不同密钥空间所耗用的时间，结论按照 4 个二进制密钥长度给出。DES(数据加密标准)算法使用的是 56 位密钥。3-DES 使用的是 168 位密钥。AES(高级加密标准)的最小密钥长度是 128 位。表中最后一行还列出了采用 26 个字母的排列组合作为密钥的替换密码的一些结果。执行一次加(解)密若需要 1 μs 的话(这是今天普通计算机的速度)，表中数据说明了对于不同长度密钥执行穷尽搜索所需的时间。随着大规模并行计算机的应用，可能对于同样规模的密钥空间有更快的穷举速度。表 2.2 最后一列列举了每微秒能处理 100 万个密钥的系统所需的时间。正如你所见，对于这种性能的计算机，DES 算法不再是计算上安全的。

表 2.2 穷尽密钥空间所需的时间

密钥大小(位)	密钥个数	每微秒执行一次加密所需的时间	每微秒执行 100 万次加密所需的时间
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ 分	2.15 毫秒
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ 年	10.01 小时
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ 年	5.4×10^{18} 年
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ 年	5.9×10^{30} 年
26 个字符的排列组合	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ 年	6.4×10^{6} 年

对称密码体制的所有分析方法都利用了这样一个事实：明文的结构和模式在加密之后仍然保存了下来，并能在密文中找到一些蛛丝马迹。在本章后面，这一点将会变得很显然。在第二部分，我们将会看到对公钥密码体制的分析是依据一个完全不同的假设，即密钥对的数学性质使得从一个密钥推出另一个密钥成为可能。

2.2 代换技术

在本节和下一节中，我们将举例探讨古典加密方法。研究这些方法可以使我们清楚今天所用的传统密码的一些基本方法，以及随之而来的密码攻击的类型等。

首先我们看一下几乎所有传统加密都要用到的两种基本技巧：代换和置换。然后再来讨论将两种技巧综合应用的密码系统。

代换法是将明文字母替换成其他字母、数字或符号的方法。[①] 如果把明文看做是二进制序列的话，那么代换就是用密文位串来代换明文位串。

① 当涉及到字母时，本书约定：用小写字母表示明文，用大写字母表示密文，用斜体小写字母表示密钥。

2.2.1 Caesar 密码

已知最早的代换密码是由 Julius Caesar 发明的 Caesar 密码。它非常简单，就是对字母表中的每个字母，用它之后的第 3 个字母来代换。例如：

```
明文：    meet me after the toga party
密文：    PHHW PH DIWHU WKH WRJD SDUWB
```

注意到字母表是循环的，即认为紧随 Z 后的是字母 A。我们可通过列出所有的可能来定义如下变换：

```
明文：    a b c d e f g h i j k l m n o p q r s t u v w x y z
密文：    D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

如果我们让每个字母等价于一个数值：

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

那么加密算法可以如下表达。对每个明文字母 p，代换成密文字母 C：[①]

$$C = \mathrm{E}(p) = (p + 3) \bmod (26)$$

移位可以是任意整数 k，这样就得到了一般的 Caesar 算法：

$$C = \mathrm{E}(p) = (p + k) \bmod (26)$$

这里 k 的取值范围从 1 到 25。解密算法是：

$$p = \mathrm{D}(C) = (C - k) \bmod (26)$$

如果已知某给定的密文是 Caesar 密码，那么穷举攻击密码学分析是很容易实现的：只要简单地测试所有 25 种可能的密钥。图 2.3 给出了应用这种策略解密的结果。对于这个例子，显然明文出现在第三行。

Caesar 密码的三个重要特征使我们可以采用穷举攻击分析方法。

1. 已知加密和解密算法。

2. 需测试的密钥只有 25 个。

3. 明文所用的语言是已知的，且其意义易于识别。

在大多数网络情况下，我们假设密码算法是已知的。一般说来，密钥空间很大的算法使得穷举攻击分析方法不太可能。例如第 6 章我们将会介绍的 3-DES 算法，它的密钥长度是 168 位，其密钥空间是 2^{168}，或者说有大于 3.7×10^{50} 种可能的密钥。

---

① 我们定义 $a \bmod n$ 为用 n 除 a 的余数。例如，$11 \bmod 7 = 4$。关于模算术的进一步讨论参见第 4 章。

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

图 2.3 对 Caesar 密码的穷举密码学分析

上述的第三个特征也是非常重要的。如果明文所用的语言不为我们所知，那么明文输出就不可识别。而且，输入可能按某种方式经过缩写或压缩，则识别就更加困难。例如图 2.4 给出的是经过 ZIP 压缩之后的部分文本文件。如果这个文件用一种简单的代换密码来加密（将字母集合扩充为不止包含 26 个英文字母），那么即使用穷举攻击进行密码分析，恢复出来的明文也是不可识别的。

```
˜+Wµ"− Ω−O)≤4{∞‡. ë~Ω%ràu·¯Í ◊¯Z−
Ú≠2Ò#Åæð œ«q7.Ωn·®3NÛÚ Œz'Y-ƒ∞Í[±Û_ èΩ,<NO¬±«ˇxã ÅäfèÜ3Å
x}ö§k°Â
_yÍ ^ΔÉ] .¤ J/'iTê&ı 'c<uΩ-
 ÄD(G WÄC~y_ïõÄW PÔı«ÎÜ†ç],¤¡ˇÌ^üÑπ˜≈ˇL˙9OgflO˜&Œ≤ ¬≤ ØÔ§˝:
 ˇŒ!SGqèvo^ ú\.S>h<-*6ø‡%x´˝⌋fiÓ#≈˜my‰ˇ≥ñP<,ƒi Áj ÅÛ¿˝Zù-
Ω¨Õ¯6ŒŸ{% „ΩÊó .ï π+ÁÎ˚úO2çSÿ´O-
2ÄflßiÍ /@^˝∏K°*PŒπ.úé^´3∑˜ö˙ÔZÌ"Y¬ŸΩœY> Ω+eô/˙<K£¿*+˜"≤Û˜
B ZøK˜Qßÿüƒ.!ÒflÎzsS/]>ÈQ ü
```

图 2.4 经过压缩的文本的样本

2.2.2　单表代换密码

Caesar 密码仅有 25 种可能的密钥，是很不安全的。通过允许任意代换，密钥空间将会急剧增大。回忆 Caesar 密码的对应：

```
明文：  a b c d e f g h i j k l m n o p q r s t u v w x y z
密文：  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

如果密文行是 26 个字母的任意置换，那么就有 26!或大于 4×10^{26} 种可能的密钥，这比 DES 的密钥空间要大 10 个数量级，应该可以抵挡穷举攻击了。这种方法称为**单表代换密码**，这是因为每条消息用一个字母表(给出从明文字母到密文字母的映射)加密。

不过，攻击办法仍然存在。如果密码分析者知道明文(例如，未经压缩的英文文本)的属性，他就可以利用语言的一些规律进行攻击。为了说明分析过程，我们在这里给出一段从文献[SINK66]中摘选出来的例子。需要解密的密文是：

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

首先把字母使用的相对频率统计出来，与英文字母的使用频率分布进行比较，见图 2.5(来自文献[LEWA00])。如果已知消息足够长的话，只用这种方法就已经足够了，但是如果这段消息相对较短，我们就不能得到准确的字母匹配。密文中字母的相对频率(按百分比)如下：

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

将这种统计规律与图 2.5 比较，可以得出结论：密文字母中的 P 和 Z 可能相当于明文中的 e 和 t，但是并不能确定 P 对应 e(Z 对应 t)还是 Z 对应 e(P 对应 t)。密文中的 S、U、O、M 和 H 相对频率都比较高，可能与明文中的字母集{a,h,i,n,o,r,s}中的某一个相对应。相对频率较低的字母(如 A,B,G,Y,I,J)可能对应着明文字母集{b,j,k,q,v,x,z}中的某个元素。

据此我们可以从以下几种方法入手。我们可以尝试着将明文中的字母替换，看看是否像一个消息的轮廓。更系统一点的方法是寻找其他的规律。例如，明文中有某些词可能是已知的，或者寻找密文字母中的重复序列，推导它们的等价明文。

统计双字母组合(比如代表单音节的两个字母)的频率是一个很有效的工具。由此可以得到一个类似于图 2.5 的字母相关频率图。最常用的一个字母组合是 th。而在我们的密文中，用得最多的双字母组合是 ZW，它出现了三次。所以我们可以估计 Z 对应明文 t，而 W 对应明文 h。根据先前的假设，我们可以认为 P 对应 e。现在我们意识到密文中的 ZWP 很可能就是 the，这是英语中最常用的三字母组合，这表明我们的思路是正确的。

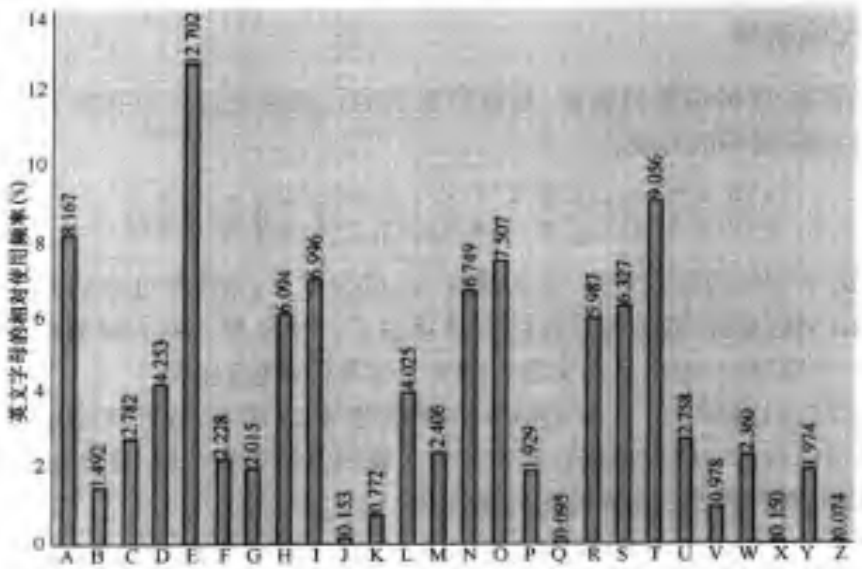


图 2.5 英文字母的相对使用频率

接下来注意到第一行中的序列 ZWSZ。我们并不知道它是否为一个完整的单词，若是这样的话，它应该翻译成 th_t 的形式，因此 S 很可能是明文 a。

至此我们有以下结果：

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a         e  e te  a that e e a        a

VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  e t   ta t ha e ee  a e  th    t  a

EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 e  e e tat  e   the   t
```

虽然只确定了 4 个字母，但是我们已经有眉目了。继续进行类似的分析、测试，很快就可以得出完整的明文，加上空格后如下：

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

单表代换密码较容易被攻破，因为它带有原始字母使用频率的一些统计学特征。一种对策是对每个字母提供多种代换，就像一个读音可以代表多个单词的同音词一样，一个明文单元也可以变换成不同的密文单元。比如字母 e 可以替换成 16、74、35 和 21，等等，循环或随机地选取其中一个即可。如果对每个明文元素(字母)分配的密文元素(如数字等)的个数与此明文元素(字母)的使用频率成一定比例关系，那么使用频率信息就完全被隐藏起来了。伟大的数学家 Carl Friedrich Gauss 认为他已经用这种"同音词"方法设计出一种牢不可破的密码。然而，即使采用了同音词方法，明文中的每个元素仅仅只对密文中的一个元素产生影响，多字母语法模式(比如双字母音节)仍然残留在密文中，这样就降低了密码分析者分析的难度。

代换密码必须考虑的一个问题是，明文的语法模式和结构有多少仍然保存在密文中。有两种基本方法可以减少这种残留：一种是对明文中的多个字母一起加密，另一种是采用多表代换密码。每一种我们都会简单地提及。

2.2.3 Playfair 密码

最著名的多表代换密码是 Playfair 密码，它把明文中的双字母音节作为一个单元并将其转换成密文的"双字母音节"。[①]

Playfair 算法基于一个由密钥词构成的 5 × 5 字母矩阵。下面的例子由 Lord Peter Wimsey 给出。

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

本例使用的密钥词是 monarchy。填充矩阵的方法是：首先将密钥词从左至右、从上至下填在矩阵格子中，再将剩余的字母按字母表的顺序从左至右、从上至下填在矩阵剩下的格子里。字母 I 和 J 暂且当做一个字母。对明文按如下规则一次加密两个字母：

1. 如果该字母对的两个字母是相同的，那么在它们之间加一个填充字母，比如 x。例如 balloon 先把它变成 ba lx lo on 这样四个字母对。
2. 落在矩阵同一行的明文字母对中的字母由其右边的字母来代换，每行中最右边的一个字母用该行中最左边的第一个字母来代换，比如 ar 变成 RM。
3. 落在矩阵同一列的明文字母对中的字母由其下面的字母来代换，每列中最下面的一个字母用该列中最上面的第一个字母来代换，比如 mu 变成 CM。
4. 其他的每组明文字母对中的字母按如下方式代换：它所在的行是该字母的所在行，列则为另一字母的所在列。比如 hs 变成 BP，ea 变成 IM(或 JM)。

Playfair 密码相对于简单的单表密码是一个巨大的进步。首先，因为有 26 个字母，故有 $26 \times 26 = 676$ 个字母对，因此对单个的字母对进行判断要困难得多。此外，单个字母在使用频率的统计规律上比字母对要强得多。这样，利用使用频率分析字母对就更困难一些。因为这些原因，Playfair密码在很长一段时间内被认为是牢不可破的。第一次世界大战中英军就使用它作为陆军的标准加密体制，并且在第二次世界大战中，美军及其他一些盟国军队用它来进行加密。

尽管 Playfair 密码被认为是较安全的，它仍然是相对容易攻破的，因为它的密文仍然完整地保留了明文语言的结构。几百个字母的密文就足够我们分析出规律了。

图 2.6 说明了 Playfair 密码和其他一些密码的有效性(来自文献[SIMM93])。标有"明文"的曲线画出了大不列颠百科辞典中关于密码的词条中超过 70 000 个字母的频率分布。[②] 这也就是单表代换密码的频率分布。曲线代表这样的含义：对文章中出现的每个字母计数，计数结

① 这个密码实际上是由英国科学家 Charles Wheatstone 于 1854 年发明的，但是却挂着他的朋友 Baron Playfair 的名字。在英国的外事机构中，Baron Playfair 在密码方面的成就是首屈一指的。

② 我感激 Gustavus Simmons 提供这些曲线并解释它们的构造方法。

果除以使用频率最高的字母 e 的出现次数。设 e 出现的频率为 1，那么 t 就大约为 0.76，等等。水平轴上的点对应着按使用频率递降的字母。

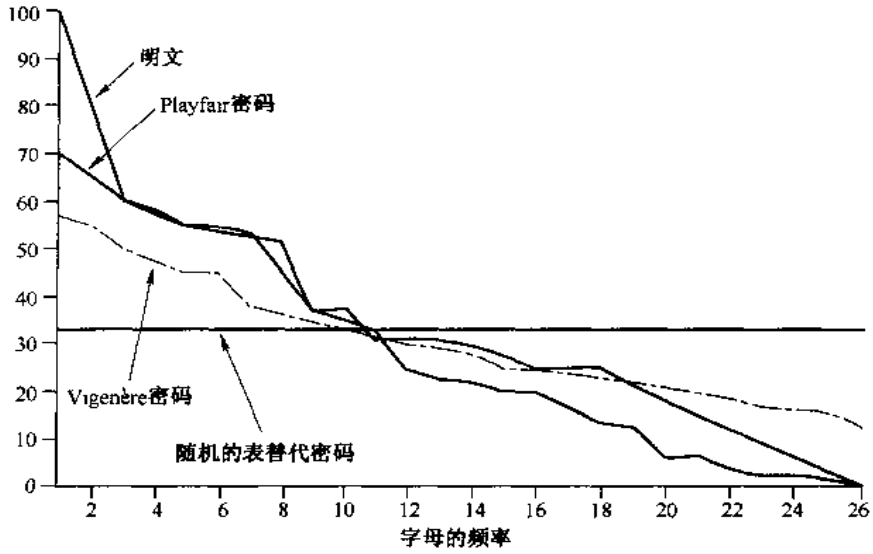


图 2.6 字母出现的相对频率

图 2.6 也表明了使用 Playfair 密码加密后文本的字母频率分布情况。为了便于比较，密文中的每个字母的使用频率仍然除以明文中 e 的使用频率。因此，(加密后的)曲线体现了加密后字母频率分布被掩盖的程度，而按字母的频率分布来分析代换密码是一个基本方法。如果频率分配的信息完全被加密过程给隐藏了，那么密文的频率曲线应该是一条水平的线，密码分析由此下手将一无所获。图中所示表明 Playfair 密码虽然有比明文稍平坦的频率分布曲线，但是仍然透露了大量的信息给密码分析者。

2.2.4 Hill 密码[①]

另一个有趣的多表代换密码是 Hill 密码，是 1929 年由数学家 Lester Hill 发明的。加密算法将 m 个连续的明文字母替换成 m 个密文字母。这是由 m 个线性等式决定的，在等式里每个字母被指定为一个数值($a=0, b=1, \cdots, z=25$)。例如 $m=3$，系统可以描述为：

$$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$$
$$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$$
$$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$$

用列矢量和矩阵表示如下：

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

① 该密码理解起来比本章的其他密码稍微困难一些，但是我重点讲述后面将要用到的密码分析方法。初学者可以跳过这一节。

或

$$\mathbf{C} = \mathbf{KP} \bmod 26$$

这里 **C** 和 **P** 是长度为 3 的列矢量，分别代表密文和明文，**K** 是一个 3 × 3 矩阵，代表加密密钥。运算按模 26 执行。

例如，对明文“paymoremoney”，用加密密钥：

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

明文的前面 3 个字母用矢量(15 0 24)表示，则有：

$$\mathbf{K}(15\ 0\ 24) = (375\ 819\ 486) \bmod 26 = (11\ 13\ 18) = \text{LNS}$$

照此方式转换余下字母，可得整段明文对应的密文是 LNSHDLEWMTRW。

解密则需要用到矩阵 **K** 的逆。矩阵 **K** 的逆矩阵 $\mathbf{K}^{-1}$ 由等式 $\mathbf{K}\mathbf{K}^{-1} = \mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$ 定义，其中 **I** 是单位矩阵。不一定所有的矩阵都有逆，但是如果有逆则一定满足上式。对于刚才这个例子，**K** 的逆是：

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

可以验证如下：

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}\begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

很明显可以看出，将 $\mathbf{K}^{-1}$ 应用于密文可以恢复出明文。为了解释矩阵的逆是如何决定的，我们非常简要地提及线性代数中的部分内容，有兴趣的读者可以找这方面的书仔细阅读。对于任意一个方阵($m \times m$)，其行列式的值等于不同行、不同列上的元素的乘积的代数和。对于一个 2 × 2 的矩阵：

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

其行列式的值为 $k_{11}k_{22} - k_{12}k_{21}$。对于一个 3 × 3 的矩阵，其行列式的值为 $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$。如果一个方阵 **A** 是非奇异的，那么矩阵的逆由 $[\mathbf{A}^{-1}]_{ij} = (-1)^{(i+j)}(D_{ij})/\det(\mathbf{A})$ 计算得出，其中 (D_{ij}) 是 **A** 去掉第 i 行第 j 列后的子行列式的值。det(**A**)是 **A** 的行列式。对本例而言，所有的运算都是对 26 取模。

用一般术语，Hill 密码可以表示如下：

$$\mathbf{C} = E_{\mathbf{K}}(\mathbf{P}) = \mathbf{KP} \bmod 26$$
$$\mathbf{P} = D_{\mathbf{K}}(\mathbf{C}) = \mathbf{K}^{-1}\mathbf{C} \bmod 26 = \mathbf{K}^{-1}\mathbf{KP} = \mathbf{P}$$

同 Playfair 密码相比，Hill 的优点是完全隐蔽了单字母频率特性。实际上，Hill 用的矩阵越大，所隐藏的频率信息就越多。因此，一个 3 × 3 的 Hill 密码不仅隐藏了单字母的频率特性，还隐藏了双字母的频率特性。

尽管 Hill 足以抗惟密文攻击，但是它较易被已知明文攻击破解。对于一个 $m \times m$ 的 Hill 密码，假如我们有 m 个明密文对，每个长度都是 m，定义 $\mathbf{P}_j = (p_{1j}, p_{2j}, \cdots, p_{mj})$和 $\mathbf{C}_j = (c_{1j}, c_{2j}, \cdots, c_{mj})$，使得对每个 $\mathbf{C}_j$ 和 $\mathbf{P}_j (1 \leqslant j \leqslant m)$都有 $\mathbf{C}_j = \mathbf{KP}_j$，其中 $\mathbf{K}$ 是未知的矩阵形密钥。现在定义两个 $m \times m$ 的矩阵 $\mathbf{X} = (p_{ij})$和 $\mathbf{Y} = (c_{ij})$。那么我们可以得出矩阵等式 $\mathbf{Y} = \mathbf{KX}$。若 $\mathbf{X}$ 可逆，则可得 $\mathbf{K} = \mathbf{YX}^{-1}$。若 $\mathbf{X}$ 不可逆，那么我们可以另找 $\mathbf{X}$ 直至得到一个可逆的 $\mathbf{X}$。

我们举一个例子来说明(选自文献[STIN02])。假设明文“friday”经过一个 2×2 的 Hill 加密生成密文 PQCFKU。因此我们知道 $\mathbf{K}(5\ 17) = (15\ 16)$；$\mathbf{K}(8\ 3) = (2\ 5)$；$\mathbf{K}(0\ 24) = (10\ 20)$。那么由这两个明密文对可得：

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \mathbf{K} \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

$\mathbf{X}$ 的逆是：

$$\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

$$\mathbf{K} = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

该结果可以由第 3 个明密文对来验证。

2.2.5 多表代换密码

改进简单的单表代换的方法是在明文消息中采用不同的单表代换。这种方法一般称之为**多表代换密码**。所有这些方法都有以下的共同特征：

1. 采用相关的单表代换规则集。
2. 由密钥决定给定变换的具体规则。

此类算法中最著名、最简单的是 Vigenère 密码。它的代换规则集由 26 个类似 Caesar 密码的代换表组成，其中每一个代换表是对明文字母表移位 0 到 25 次后得到的代换单表。每个密码代换表由一个密钥字母来表示，这个密钥字母就是用来代换明文字母 a 的那个字母。对于 Caesar 密码，代换表是移位 3 次得到的，所以 Caesar 密码的代换表由密钥值 d 来代表。

为了帮助理解上述密码机制和用法，我们构造一个 Vigenère 密码表的矩阵(表 2.3)。26 个密码水平置放，最左边是其密钥字母，顶部排列的是明文的标准字母表。加密过程很简单：给定密钥字母 x 和明文字母 y，密文字母是位于 x 行和 y 列的那个字母，在本例中密文是 V。

加密一条消息需要与消息一样长的密钥。通常，密钥是一个密钥词的重复，比如密钥词是 *deceptive*，那么消息“we are discovered save yourself”将被这样加密：

```
key:          deceptivedeceptivedeceptive
plaintext:    wearediscoveredsaveyourself
ciphertext:   ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

解密同样简单。密钥字母决定行。密文字母所在列的顶部字母就是明文字母。

这种密码的强度在于每个明文字母对应着多个密文字母，且每个使用惟一的字母作为密钥词，因此字母出现的频率信息被隐蔽了，不过并非所有的明文结构信息都隐蔽了。例如，图 2.6 给出了9 个字母密钥词的 Vigenère 密码频率分布特征。尽管它对于 Playfair 密码是一个较大的改进，却依然保留了许多频率信息。

表 2.3　现代Vigenère 替换表

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

略述这种密码的攻击方法很有益，因为它体现了密码分析学中的一些数学原理。

首先，假设敌手认为密文是用单表代换或 Vigenère 密码来加密的。可以用一个简单的测试来做个区分。如果用单表代换，那么密文的统计特性应该与明文语言的统计特性相同，因此参照图 2.5，应该有一个密文字母的出现频率大约是 12.7%，一个大约是 9.06%等。如果只有一条消息可用于密码分析，那么我们并不期望它所体现出来的统计规律与明文的统计规律非常接近。然而当它们的统计规律很接近时，我们就认为是用单表代换加密的。

另一方面，如果认为是用 Vigenère 密码加密的，那么它会依赖于所用密钥词的长度，我们很快就会意识到这一点。现在我们集中精力寻找密钥词长度。洞察到这个事实是很重要的：如果两个相同的明文序列之间的距离是密钥词长度的整数倍，那么产生的密文序列也是相同的。在前面的例子里，“red”的两次出现相隔 9 个字母。在两种情况下，r 都是用 e 加密，e 都是用 p 加密，d 都是用 t 加密，因此得到了两个相同的密文序列 VTW。

分析者只需发现重复序列 VTW，而重复的 VTW 之间相隔 9 个字符，那么他（或她）就认为密钥词的长度是 3 或 9。VTW 的两次出现也可能是偶然的，而不一定是用相同密钥加密相同明文序列导致的。然而，如果信息足够长，就会有大量重复的密文序列出现。通过计算重复密文序列间距的公因子，分析者就很有可能猜出密钥词的长度。

破解密码也依靠于另一个重要的观察。如果密钥词的长度是 N，那么密码实际上包含了 N 个单表代换。例如，以 DECEPTIVE 作为密钥词，那么处在位置 1，10，19，…的字母的加密实际上是单表密码。因此，我们可以用明文语言的频率特征对这样的单表密码分别进行攻击。

密钥词的周期性可以用与明文信息一样长的不重复密钥词来消除。Vigenère 提出用一个所谓的"密钥自动生成系统"（比如明文自身）来提供不重复的密钥词。请看下面的例子：

```
密钥：    deceptivewearediscoveredsav
明文：    wearediscoveredsaveyourself
密文：    ZICVTWQNGKZEIIGASXSTSLVVWLA
```

即使采用这个方案，它也是易受攻击的。因为密钥和明文具有相同频率的分布特征，所以我们可以应用统计学的方法。例如，用 e 加密 e，由图 2.5 可以估计出其发生的频率为 $(0.127)^2 \approx 0.016$，而用 t 加密 t 发生的频率大概只有它的一半。密码分析者利用这些规律能够成功地进行分析。[①]

最终的反破译措施也许只有选择一个与明文毫无统计关系且和明文一样长的密钥了。1918 年工程师 Gilbert Vernam 首先引入了这种体制，其运算基于二进制数据而非字母。该体制可以简明地表述为：

$$c_i = p_i \oplus k_i$$

其中：

p_i = 明文第 i 个二进制位

k_i = 密钥第 i 个二进制位

c_i = 密文第 i 个二进制位

$\oplus$ = 异或(XOR)运算符

因此密文是通过对明文和密钥的逐位异或而成。根据异或运算的性质，解密过程为：

$$p_i = c_i \oplus k_i$$

这种技巧的本质在于构造密钥的方式。Vernam 提出使用周期很大的循环密钥。尽管周期很长对于密码分析增添了相当大的难度，但是如果有足够的密文，使用已知或可能的明文序列，或者联合使用二者，该方案是可以被破解的。

2.2.6 一次一密

陆军情报军官 Joseph Mauborgne 提出了一种对 Vernam 密码的改进方案，从而达到了最完善的安全性。Mauborgne 建议使用与消息一样长且无重复的随机密钥来加密消息，这就是著名的一次一密密码，它是不可攻破的。它产生的随机输出与明文没有任何统计关系。因为密文不包含明文的任何信息，所以无法可破。

下面的例子能够说明我们的观点。假设我们使用的是 27 个字符（第 27 个字符是空格）的 Vigenère 密码，但这里使用的一次密钥和消息一样长。因此表 2.3 的规模应扩展为 27×27。请看下面的密文：

---

① 虽然破译 Vigenère 密码的技术并不复杂，但是 1917 年的一期《科学美国人》杂志上却称之为不可破译的。当对现代密码算法做出类似论断时，这是值得汲取的教训。

ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

现在我们用两种不同的密钥解密：

密文：　ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
密钥：　*pxlmvmsydoftyrvzwc tnlebnecvgdupahfzzlmnyih*
明文：　mr mustard with the candlestick in the hall

密文：　ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
密钥：　*mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt*
明文：　miss scarlet with the knife in the library

假设密码分析者已设法找到了这两个密钥，于是就产生了两个似是而非的明文。分析者如何确定正确的解密(正确的密钥)呢？如果密钥是在真正随机的方式下产生的，那么分析者就不能说密钥更有可能是哪一种。因此没有办法确定正确的密钥，也就是说没有办法确定正确的明文。

事实上，给出任何长度与密文一样的明文，都存在着一个密钥产生这个明文。因此，如果你用穷举法搜索所有可能的密钥，就会得到大量可读、清楚的明文，但是没有办法确定哪一个才是真正所需的，因而这种密码是不可破的。

一次一密的安全性完全取决于密钥的随机性。如果构成密钥的字符流是真正随机的，那么构成密文的字符流也是真正随机的。因此分析者没有任何攻击密文的模式和规则可用。

理论上，对一次一密已经讲得很清楚了。但是在实际中，要一次一密提供完全的安全性，存在两个基本难点：

1. 产生大规模随机密钥的实际困难。任何经常使用的系统都需要建立在某个规则基础上的百万个随机字符，提供这样规模的真正随机字符是相当艰巨的任务。
2. 更令人担忧的是密钥的分配和保护。对每一条发送的消息，需要提供给发送方和接收方等长度的密钥。因此，存在庞大的密钥分配问题。

因为上面这些困难，一次一密在实际中很少使用，而主要用于安全性要求很高的低带宽信道。

2.3　置换技术

到现在为止我们所讨论的都是将明文字母代换为密文字母。与之极不相同的另一种加密方法是通过置换而形成新的排列，这种技术称为置换密码。

最简单的例子是栅栏技术，按照对角线的顺序写入明文，而按行的顺序读出作为密文。例如，用深度为 2 的栅栏技术加密信息“meet me after the toga party”，可写为：

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

加密后的信息是 MEMATRHTGPRYETEFETEOAAT。

这种技巧对密码分析者来说实在微不足道。一种更复杂的方案是把消息一行一行地写成矩形块，然后按列读出，但是把列的次序打乱。列的次序就是算法的密钥。例如：

```
密钥:      4 3 1 2 5 6 7
明文:      a t t a c k p
           o s t p o n e
           d u n t i l t
           w o a m x y z

密文:      TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

单纯的置换密码因为有着与原始明文相同的字母频率特征而易被识破。如同列变换所示，密码分析可以直接从将密文排列成矩阵人手，再来处理列的位置。双字母音节和三字母音节分析办法可以派上用场。

多步置换密码相对来讲要安全得多。这种复杂的置换是不容易构造出来的。因此，如果前面那条消息用相同算法再加密一次，则有：

```
密钥:      4 3 1 2 5 6 7
明文:      t t n a a p t
           m t s u o a o
           d w c o i x k
           n l y p e t z

密文:      NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

为了更清晰地看出经过双重置换后的结果，我们用字母所在位置的序号来代换原始明文信息。于是，共 28 个字母的原始消息序列是：

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

经过第一次置换后变成了：

```
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28
```

这多少还是有些规律，但是经过第二次置换变成

```
17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28
```

之后，排列结构已经没有什么规律了，分析者攻击它要困难得多。

2.4 转轮机

刚才的例子表明，用多步置换得到的算法对密码分析有很大的难度。这对代换密码也适用。在介绍 DES 之前，我们先来介绍对多层加密原理有着重要应用的例子，即转轮机密码系统。[①]

转轮机的基本原理如图 2.7 所示。转轮机包括一组相互独立的旋转轮，电脉冲可以由它通过。每个圆筒有 26 个输人引脚和 26 个输出引脚。内部连线使每一个输人仅同惟一一个输出连接，为简明起见，图中只画出了三条内部连接。

如果我们把每个输人输出引脚当做字母表中的一个字母，那么一个圆筒就定义了一个单

① 在第二次世界大战中，德国(Enigma 密码机)和日本(Purple 密码机)都使用了基于转轮原理的密码机。盟军破译了这两种密码，对二战的结局产生了重要的影响。

表替换。如图 2.7 所示，如果操作员按下代表字母 A 的键，那么电信号就加在了第一个圆筒的第一个输入引脚上，经过内部连线被传送到第 25 个输出引脚。

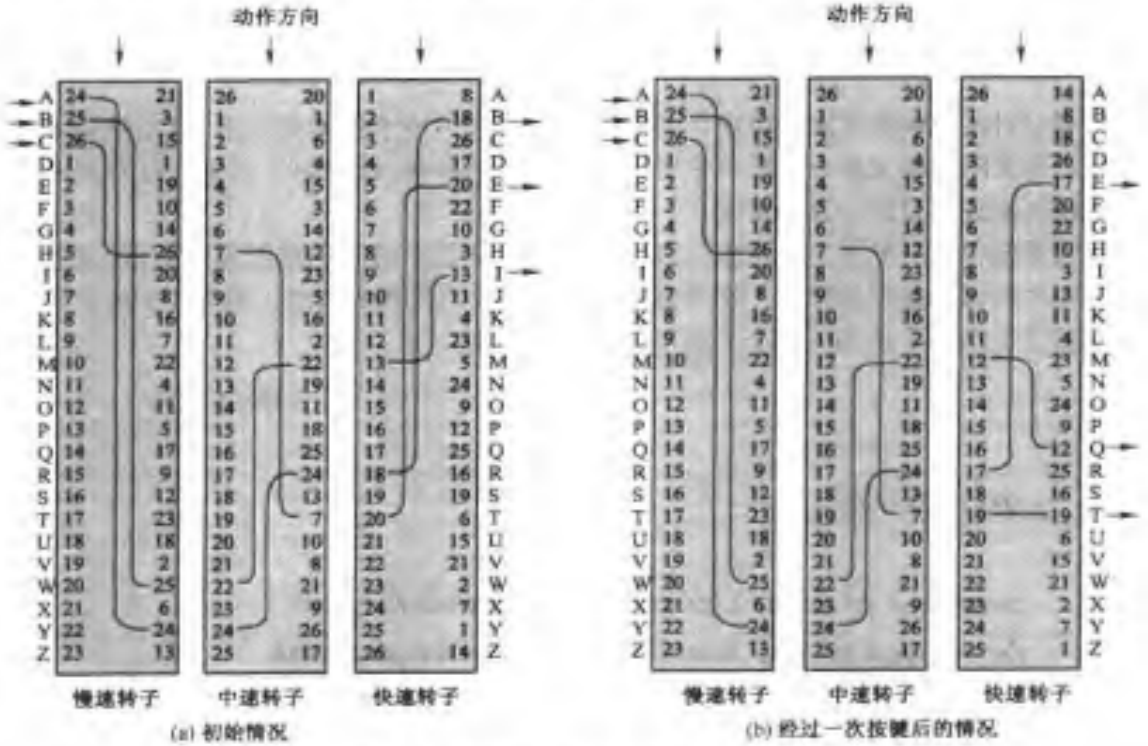


图 2.7　用编号表示内部引线的三转子机

考虑只有一个圆筒的转轮机。每次按下一个输入键，圆筒就旋转一个位置，所以内部连线也就相应改变了。因此就定义了不同的单表代换密码。经过 26 个明文字母后，圆筒回到初始位置。于是我们可以得到一个周期为 26 的多表代换算法。

单筒系统还是比较简单的，容易对付。转轮机的威力在于它使用了多个圆筒，每个筒的输出引脚连接到下一个筒的输入引脚。图 2.7 所示的是一个三筒系统。图中左半部分，操作员从第一个引脚输入(明文字母 a)，经过 3 个圆筒之后，电信号出现在第 3 个圆筒的第 2 个引脚(密文字母 B)。

多筒系统中，操作员每按一次输入键，最后一个转轮就旋转一个引脚的位置。图 2.7 的右半部分表明了经过一次按键后的情况。最后一个转轮旋转完一圈之后，它前面一个圆筒就旋转一个引脚的位置，依此类推。它的原理看起来就像是里程表，所以整个系统重复使用 $26 \times 26 \times 26 = 17\,576$ 个不同的替换字母表，如果用 4 个或 5 个转轮，周期数将分别为 456 976 和 11 881 376。David Kahn在文献[KAHN96]的第 413 页对五筒转轮机做了详细的论述。其中一段是：

> 足够大的周期可以挫败任何想直接通过字母的频率特征来解密的实际可能。这种普通攻击方法对每个密文表大约需要 50 个字母，这意味着所有 5 个转轮将必须经历 50 次循环。这需要有多长的密文呢？密文长度相当于连续三届国会在美国参议院发表演讲的长度。没有任何密码分析者可能在一生中戴上这样的桂冠，即使是外交家和最善于辞令的政客也不可能达到这种程度。

今天,转轮机的意义在于它曾经给最为广泛使用的密码——数据加密标准 DES 指明了方向。我们将在第 3 章中对 DES 进行讨论。

2.5 隐写术

我们通过讨论一种技术来结束这一章,严格说来,这种技术并不是加密,而是隐写术。有两种办法可用来隐藏明文信息。隐写术可以隐藏信息的存在,而密码学则是通过对文本信息的不同转换而实现信息的对外不可读。①

一种简单却很耗时的隐写术可由一段明显无伤大雅文本的字词重新排列而成。例如,在一整段文本中用每个单词的第一个字母连起来就可以拼出隐藏的消息。图 2.8 中的例子可以利用一整段文本信息中单词的某个子集来表达隐藏信息。

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours,

图 2.8 侦探 Morse 的疑惑。源于 Colin Dexter 的《Nicholas Quinn 的沉默世界》

历史上还用过其他一些技术。下面给出一些例子(参见文献[MYER91]):

- **字符标记**:选择一些印刷字母或打字机打出的文本,用铅笔在其上书写一遍。这些标记

① 隐写术是一种过去的说法,现在 David Kahn 将其修正并且给出了新的含义,见文献[KAHN96]。

需要做得在一般场合下辨认不出，除非将纸张按某个角度对着亮光看。

- **不可见墨水**：有些物质用来书写后不留下可见痕迹，除非加热或加之以某种化学物质。
- **针刺**：在某些字母上刺上小的针孔，这一般是分辨不出来的，除非对着光线。
- **打字机的色带校正**：用黑色的色带在行之间打印。用这种色带打印后的东西只在强光下可见。

尽管以上这些方法显得古老，但是却具有时代的意义。文献[WAYN93]陈述了如何用 CD 每一帧的最后一位来隐藏信息。例如，柯达 CD 格式的最大分辨率是 2048 × 3072，其中每一个像素包含有 24 位的 RGB 颜色信息。改动这 24 位像素的最低一位对整个画质影响不大。结果是你可以在每一张数字快照中隐藏 2.3 兆字节的信息。有很多软件包就是采用诸如此类的办法来进行信息隐蔽的。

同加密相比，隐写术有一些缺点。它需要许多额外的付出来隐藏相对较少的信息。尽管采用一些诸如上述方案也许很有效，但是一旦被破解，整个方案就毫无价值了。不过这个问题也可以克服，比如具体方法由密钥来决定（例如，习题 2.11）。另外一种方法是，将消息先加密，再将其隐写。

隐写术的优点是可以应用于通信双方宁愿他们的秘密通信被发现而不愿其中的重要内容丢失的情况。加密标识的传输也是重要和秘密的，通过它可以找出隐藏消息的发送方（或接收方）。

2.6　推荐读物和网址

对密码编码学和密码破译学的历史感兴趣的读者，可参阅文献[KAHN96]。尽管这本书更注重于密码学的影响而非密码技术的发展，但是它的确是一本密码学的优秀介绍读物，读来饶有趣味。另一本介绍密码学发展历史的读物是[SING99]。

[GARD72]一书则简洁地介绍了本章和其他章节所涉及的技术。涉及古典密码技术并更侧重于技术的文献有很多：[SINK66]是最的好文献之一。[KORN96]读来让人感到很愉快，它用很长篇幅的一节讲述传统技巧。[GARR01]和[NICH99]这两本密码书包含大量传统技术的材料。[NICH96]详细地介绍了大量的传统密码，提供了许多密文及其解，对于真正有兴趣的读者有较大的参考价值。

转轮机的详细数学描述可参考文献[KOHN81]，仍在销售中。另一本文献[KUMA97]对转轮机进行了合理的处理，并讨论了它们的密码分析。

文献[KATZ00]详细地介绍了隐写术。[WAYN96]是另一个很好的资源。

GARD72　Gardner, M. *Codes, Ciphers, and Secret Writing*. New York: Dover, 1972.

GARR01　Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.

KAHN96　Kahn, D. *The Codebreakers: The Story of Secret Writing*. New York: Scribner, 1996.

KATZ00　Katzenbeisser, S., ed. *Information Hiding Techniques for Steganography and Digital Watermarking*. Boston: Artech House, 2000.

KOHN81 Konheim, A. *Cryptography: A Primer*. New York: Wiley, 1981.

KORN96 Korner, T. *The Pleasures of Counting*. Cambridge, England: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.

NICH96 Nichols, R. *Classical Cryptography Course*. Laguna Hills, CA: Aegean Park Press, 1996.

NICH99 Nichols, R. ed. *ICSA Guide to Cryptography*. New York: McGraw-Hill, 1999.

SING99 Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary Cryptanalysis: A Mathematical Approach*. Washington, DC: The Mathematical Association of America, 1966.

WAYN96 Wayner, P. *Disappearing Cryptography*. Boston: AP Professional Books, 1996.

推荐网址：

- **American Cryptogram Association(美国密码电文协会)**：一个业余密码爱好者的协会。该网上有传统密码学的信息，并可以链接到有相关信息的网上。

2.7 关键术语、思考题和习题

2.7.1 关键术语

分组密码	密码编码学	明文
密文	加密过程	单钥加密
密码体制	一次一密	对称加密
解密	栅栏密码	密码
单表代换密码	流密码	密码分析学
多表代换密码	Vigenere 密码	解密过程
隐写术	Caesar 密码	Hill 密码
无条件安全	传统加密	Playfair 密码
穷举攻击	密码学	置换密码
计算上的安全	加密	

2.7.2 思考题

2.1 什么是对称密码的本质成分?

2.2 密码算法中两个基本函数是什么?

2.3 用密码进行通信的两个人需要多少密钥?

2.4 分组密码和流密码的区别是什么?

2.5　攻击密码的两种一般方法是什么?
2.6　列出并简要地定义基于攻击者所知道信息的密码分析攻击类型。
2.7　无条件安全密码和计算上安全密码的区别是什么?
2.8　简要地定义 Caesar 密码。
2.9　简要地定义单表代换密码。
2.10　简要地定义 Playfair 密码。
2.11　单表代换密码和多表代换密码的区别是什么?
2.12　一次一密的两个问题是什么?
2.13　什么是置换密码?
2.14　什么是隐写术?

2.7.3　习题

2.1　已知下面的密文由单表代换算法产生:

```
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8¶60))85;;]8*;:‡*8†83
(88)5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956*2(5*-4)8¶8*
;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4)485†528806*81
(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;
```

请将它破译。提示:

1. 正如你所知,英文中最常见的字母是 e。因此,密文第 1 个或第 2 个(或许第 3 个)最高出现频率的字符应该代表 e。此外,e 经常成对出现(如 meet、fleet、speed、seen、been、agree,等等)。找出代表 e 的字符,并首先将它译出来。
2. 英文中最常见的单词是"the"。利用这个事实猜出什么字符代表字母 t 和 h。
3. 根据已经得到的结果破译其他部分。

注意:最终得到的英文明文第一眼看起来可能不太好懂。

2.2　解决密钥分配问题的一个办法是使用收发双方都有的一本书中的某行文字。至少在某些侦探小说中经常把一本书的第一句话作为密钥。这里就从一本富于悬念的侦探小说——Ruth Rendell 的《与陌生人的谈话》中找到了一个例子。请不要找到这本书之后再来做这道题!

给定下列消息:

```
SIDKHKDM AF HCRKIABIE SHIMC KD LFEAILA
```

这段密文是用《沉默的背后》(一本有关间谍 Kim Philby 的小说)一书的第一句话和单表代换方法产生的,这句话是:

The snow lay thick on the steps and the snowflakes driven by the wind looked black in the headlights of the cars.

使用的是简单的代换密码。

a. 加密算法是什么样的?
b. 它的安全性怎么样?
c. 为了使密钥分配问题简单化,通信双方都同意使用一本书的第一句话或最后一句话作为密钥。要想改变密钥,他们只需更换一本书就行了。使用第一句话比

使用最后一句话要好。为什么?

2.3 在Sherlock Holmes的案件中,有一例是这样的。有一段消息:

534 C2 13 127 36 31 4 17 21 41
DOUGLAS 109 293 5 37 BIRLSTONE
26 BIRLSTONE 9 127 171

尽管Watson被难住了,但Holmes一下子就推出了所用密码的类型。你能吗?

2.4 普通的单表代换密码的一个缺点是,收发双方都必须将打乱的密码序列记住。为了避免这一点,经常使用这种方法:采用可推出整个密码序列的一个密钥。例如,使用密钥词CIPHER,然后将字母表中没有使用的字母按正常的顺序排在CIPHER的后面:

```
密文:  a b c d e f g h i j k l m n o p q r s t u v w x y z
明文:  C I P H E R A B D F G J K L M N O Q S T U V W X Y Z
```

如果这样做不能产生足够的混淆,那么可以将字母表中没有使用的字母按密钥词排成几行,再按列读出:

```
C I P H E R
A B D F G J
K L M N O Q
S T U V W X
Y Z
```

这就得到了序列:

```
C A K S Y I B L T Z P D M U H F N V E G O W R J Q X
```

在2.3节的例子中使用了这种方法。试确定密钥词。

2.5 Playfair密码可用的密钥有多少个?要求写成接近2的乘方形式。

2.6 **a.** 用Hill加密消息"meet me at the usual place at then rather than eight oclock",密钥为 $\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$。要求写出计算过程和结果。

b. 写出从密文恢复为明文所做的解密计算。

2.7 我们知道只要有足够多的明密文对,Hill是不能抗已知明文攻击的。如果可使用选择明文攻击,Hill就更脆弱了。请描述这种攻击方案。

2.8 当海军上尉John F. Kennedy下令日本毁灭者击沉美国巡逻船PT-109时,在澳大利亚的无线站截获了一条用Playfair密码加密的消息:

```
KXJEY UREBE ZWEHE WRYTU HEYFS
KREHE GOYFI WTTTU OLKSY CAJPO
BOTEI ZONTX BYBWT GONEY CUZWR
GDSON SXBOU YWRHE BAAHY USEDQ
```

密钥为 *royal new zealand navy*。请解密这条消息。

2.9 本题探讨了Vigenère密码的一次一密版本的用途。在这种方案中,密钥是0至26间的随机数流。例如,如果密钥是3 19 5 …,则明文的首个字母使用3个字母的移位加密,第二个字母使用19个字母的移位加密,第三个字母使用5个字母的移位加

密，依此类推。

a. 使用密钥流 9 0 1 7 23 15 21 14 11 11 2 8 9 加密明文 sendmoremoney。

b. 使用(a)中产生的密文找到一个密钥，以便该密文解密为 cashnotneeded。

2.10　隐藏在图 2.8 中的消息是什么？

2.11　在多罗西的怪诞小说中，有一个故事是这样的：地主彼得遇到了图 2.9 所示的消息，他找到了密钥，是一段整数：

78765654343211234345656787878765654
34321123434565678788787656544332112 34

a. 破译这段消息。提示：最大的整数是什么？

b. 如果只知道算法而不知道密钥，这种加密方案的安全性怎么样？

c. 如果只知道密钥而不知道算法，这种加密方案的安全性又怎么样？

I thought to see the fairies in the fields, but I saw only the evil elephants with their black backs. Woe! how that sight awed me! The elves danced all around and about while I heard voices calling clearly. Ah! how I tried to see—throw off the ugly cloud—but no blind eye of a mortal was permitted to spy them. So then came minstrels, having gold trumpets, harps and drums. These played very loudly beside me, breaking that spell. So the dream vanished, whereat I thanked Heaven. I shed many tears before the thin moon rose up, frail and faint as a sickle of straw. Now though the Enchanter gnash his teeth vainly, yet shall he return as the Spring returns. Oh, wretched man! Hell gapes, Erebus now lies open. The mouths of Death wait on thy end.

图 2.9　地主彼得的迷惑

第 3 章　分组密码与数据加密标准

本章的目的在于阐明现代对称密码的基本原理。因此，我们主要探讨使用最广泛的对称密码：数据加密标准(DES)。尽管引入 DES 之后涌现了大量的对称密码，尽管它注定要被高级加密标准(AES)所取代，但是 DES 依然是一个极其重要的算法。而且，详细地讨论 DES 算法，也可以帮助我们深刻地理解其他对称密码的原理。在第 5 章、第 6 章，我们也将讨论包括 AES 在内的其他重要的对称密码。

与像 RSA 这样的公钥密码相比，DES 以及大多数传统加密算法的结构都非常复杂，不像 RSA 或类似的算法能给予简洁的描述。因此，我们先讨论 DES 的一个简化版本。这样可以让我们手工执行加、解密运算并对算法的详细工作过程有较好的理解。课堂经验表明，对简化版本的研究能提高对 DES 的理解。[①]

讨论完 DES 的简化版之后，本章将研究对称分组密码的一般原理，本书所说的对称密码是指对称分组密码(除第 6 章的流密码 RC4 外)。接下来我们研究完整的 DES。然后来看一个具体的算法，再回到对分组密码设计的一般性讨论上来。

3.1　简化 DES

简化 DES 是由圣达卡拉大学的爱德华·施菲尔教授提出的(见文献[SCHA96])，为用做教学而不能算一个安全的加密算法。它和 DES 有着相似的性质和结构，但是参数要小得多。读者将发现随着本节的讨论，如果同时用手工计算一个例子会加深对 DES 的理解。[②]

3.1.1　概论

图 3.1 说明了简化 DES 的整体结构，我们称之为 S-DES。S-DES 的加密算法输入为一个 8 位的明文组(例如 10111101))和一个 10 位的密钥，输出为 8 位的密文组。S-DES 的解密算法的输入则是一个8 位的密文组和一个 10 位的密钥，输出为 8 位的明文组。

加密算法包括 5 个函数：初始置换(IP)；复杂函数 f_K，它包含有置换和代换运算，并且依赖于输入的密钥；用以转换数据两个部分的简单置换函数(SW)；再一次运用函数 f_K；最后的一个置换函数是 IP 的逆，记为 IP^{-1}。正如我们在第 2 章提到过的，采用多层置换和代换将使算法更加复杂，能增加密码分析的难度。

f_K 的输入不仅包括通过加密算法的数据，而且包括一个 8 位的密钥。算法可以设计成 16 位密钥的工作模型，包含两个8 位的子密钥，分别用在 f_K 的两次出现中。另一种方法是将 8 位密钥在算法中使用两次。折衷的方案如图 3.1 所示，它是用一个 10 位的密钥来产生两个

---

① 读者完全可以跳过 3.1 节，至少在初次阅读时可以这样做。如果因为陷入 DES 的技术细节而感到迷惑，那么读者可以回过头来再从简化 DES 开始。

② 本节基于 Schaefer 的论文。

8 位的子密钥。在这个例子中，密钥先经过置换(P10)和移位，移位操作的输出经过置换产生 8 位的输出(P8)作为第一个子密钥(K_1)。移位操作的输出经过另一个移位和另一个 P8 产生第二个子密钥(K_2)。

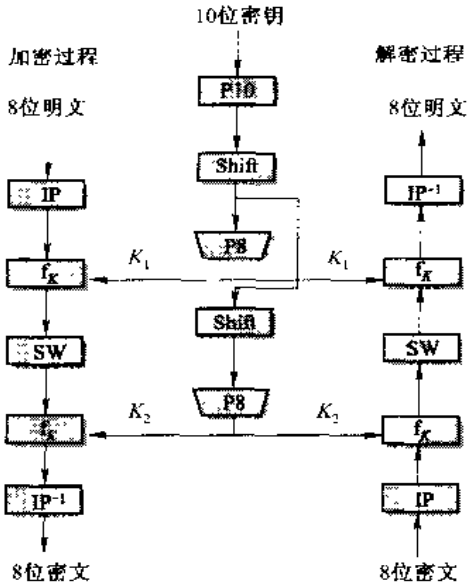


图 3.1　简化的 DES 体制

我们可用函数的复合[①] 把加密过程简单地表示为：

$$IP^{-1} \circ f_{K_2} \circ SW \circ f_{K_1} \circ IP$$

也可写为：

$$密文 = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(明文)))))$$

其中：

$$K_1 = P8(Shift(P10(key)))$$
$$K_2 = P8(Shift(Shift(P10(key))))$$

图 3.1 中也给出了解密过程，它本质上是加密的逆过程：

$$明文 = IP^{-1}(f_{K_1}(SW(f_{K_2}(IP(密文)))))$$

现在我们详细地研究 S-DES 的各个部分。

3.1.2　S-DES 的密钥产生

S-DES 依赖于收、发双方共享的 10 位密钥，它产生的两个 8 位子密钥分别用在加、解密的不同阶段。图 3.2 描述了这个过程。

① 定义：如果 f 和 g 是两个函数，且 $y = F(x) = g[f(x)]$，则称函数 F 是 f 和 g 的复合函数，并记为 $F = g \circ f$。

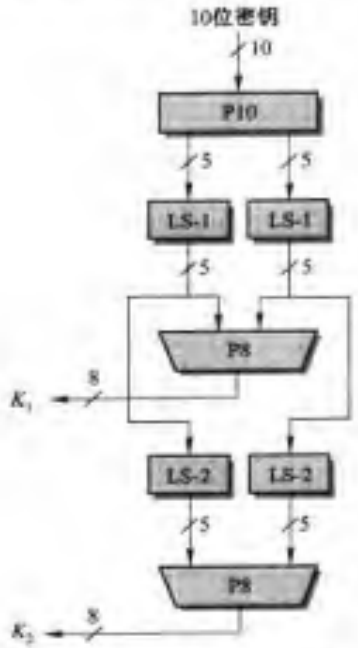


图 3.2 简化 DES 的密钥产生

首先，按如下方式置换密钥。10 位密钥表示成$(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$。置换 P10 定义为：

$$P10(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)$$

也可按下面的排列简单地定义：

P10									
3	5	2	7	4	10	1	9	8	6

上表从左向右读；表中的位置给出了产生这个位置上的输出位的输入位的位置，于是第 1 个输出位是输入的第 3 位，第 2 个输出位是输入的第 5 位，等等。例如，密钥(1010000010)经过置换后变成了(1000001100)。接下来分别将前面 5 位和后面 5 位循环左移(LS-1)一位，本例中结果为(00001 11000)。

接下来，选取 10 位中的 8 位按下面的规则运用置换 P8：

P8							
6	3	7	4	8	5	10	9

得到子密钥 1(K_1)，本例中 K_1 = (10100100)。

再回到由两个 LS-1 函数产生的 5 位串对，它们分别循环左移 2 位；本例中(00001 11000)就变成了(00100 00011)；然后根据置换 P8 产生 K_2，本例的结果为(01000011)。

3.1.3 S-DES 加密

图 3.3 更加详细地描述了 S-DES 的加密算法。正如刚才所提到的，加密过程包括五个函数。我们分别考虑每一个函数。

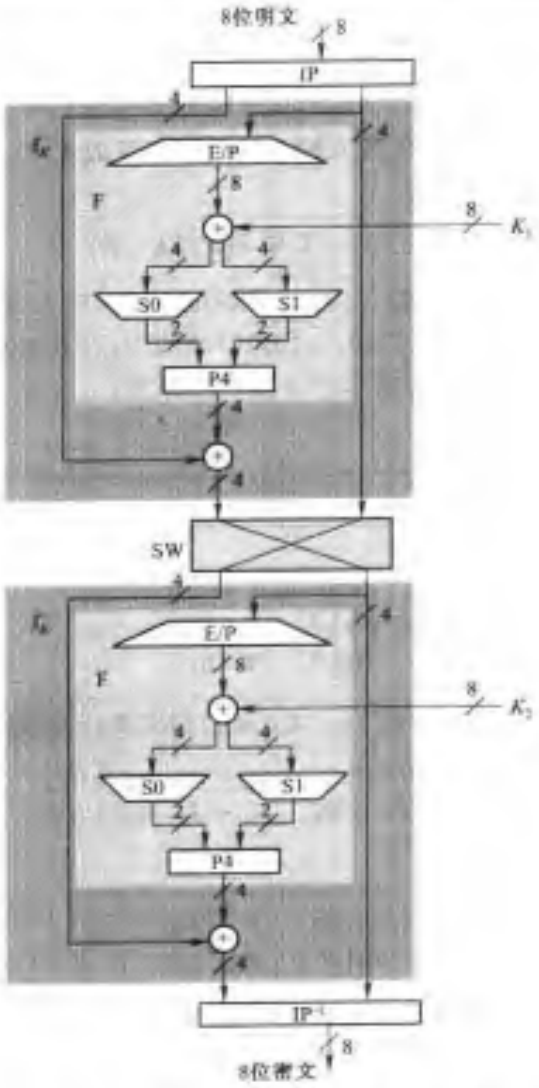


图 3.3　简化 DES 密码体制的加密过程

初始置换和末尾置换

算法的输入是一个 8 位的明文分组，我们首先使用 IP 函数对它进行置换：

IP							
2	6	3	1	4	8	5	7

它保留了明文的所有 8 位信息但将其混淆。算法最后使用的逆置换是：

IP^{-1}							
4	1	3	5	7	2	8	6

易证第二个置换是第一个置换的逆，即 $IP^{-1}(IP(X)) = X$。

函数 f_K

S-DES 算法最复杂的部分是函数 f_K，它由置换函数和代换函数组合而成。函数可按如下方式表达。设 L 和 R 分别是 f_K 的 8 位输入的左边 4 位和右边 4 位。F 是一个 4 位串到 4 位串的映射(不一定是一一映射)。那么设

$$f_K(L, R) = (L \oplus F(R, SK), R)$$

其中，SK 是子密钥，$\oplus$ 是按位异或函数。例如，设经过图 3.3 中 IP 置换的输出为(10111101)，F(1101, SK) = (1110)。因为(1011)$\oplus$(1110) = (0101)，所以 f_K(10111101) = (01011101)。

现在我们描述映射 F。它的输入是一个 4 位二进制数($n_1 n_2 n_3 n_4$)。第一个操作是扩展/置换操作：

E/P							
4	1	2	3	2	3	4	1

用如下方式表达也许会更清楚一些：

$$\begin{array}{c|c|c|c} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{array}$$

将 8 位子密钥 $K_1 = (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18})$与之进行异或运算后得：

$$\begin{array}{c|c|c|c} n_4 + k_{11} & n_1 + k_{12} & n_2 + k_{13} & n_3 + k_{14} \\ n_2 + k_{15} & n_3 + k_{16} & n_4 + k_{17} & n_1 + k_{18} \end{array}$$

将其记为：

$$\begin{array}{c|c|c|c} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \end{array}$$

前面 4 位(矩阵的第一行)输入到 S 盒 S0 中得到一个 2 位的输出，其余 4 位(第 2 行)输入到 S1 中产生另一个 2 位的输出。这两个 S 盒定义为：

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

S 盒的操作如下：第 1 位和第 4 位作为一个 2 位二进制数决定 S 盒的行，第 2 位和第 3 位决定 S 盒的列。输出也是一个 2 位的二进制数。例如，若 ($p_{0,0}$ $p_{0,3}$) = (00)，($p_{0,1}$ $p_{0,2}$) = (10)，则输出是 S0 盒第 0 行第 2 列的数 3，或写成二进制(11)。类似地，($p_{1,0}$ $p_{1,3}$)和($p_{1,1}$ $p_{1,2}$)也用于代表 S1 盒的行和列产生的一个 2 位二进制输出。

接下来，由 S0 和 S1 的输出组成的 4 位再进行一次如下的置换：

P4			
2	4	3	1

P4 的输出就是函数 F 的输出。

交换函数

函数 f_K 只改变输入的最左边 4 位。交换函数 SW 将输入的左 4 位和右 4 位交换，这样 f_K 再次作用的就是不同的 4 位了。但在这次作用中，E/P，S0，S1 和 P4 都是相同的，输入的密钥为 K_2。

3.1.4　简化 DES 的分析

对简化 DES 进行穷举攻击当然是可行的，因为 10 位的密钥只存在 $2^{10} = 1024$ 种可能。给定一段密文，攻击者可以尝试所有可能的密钥，对结果进行分析，从而得到合理的明文。

如何进行密码分析呢？先来考虑已知明文攻击。假设一条明文为 $(p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$，其密文为 $(c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$，而密钥 $(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$ 是未知的。那么 c_i 是 p_j 和 k_j 的多项式函数 g_i。因此我们可以将加密算法表示成 10 个未知量的 8 个非线性等式。可能有许多解，但是每一个都可以计算出来并用于分析。算法中的置换和加法都是线性映射。非线性成分来自 S 盒。写下这些 S 盒对应的等式是很有用的。为清楚起见，我们把 $(p_{0,0}, p_{0,1}, p_{0,2}, p_{0,3})$ 写做 (a, b, c, d)，把 $(p_{1,0}, p_{1,1}, p_{1,2}, p_{1,3})$ 写做 (w, x, y, z)，它们的四位输出是 (q, r, s, t)，于是 S0 的操作可以定义为如下方程：

$$\begin{aligned} q &= abcd + ab + ac + b + d \\ r &= abcd + abd + ab + ac + ad + a + c + 1 \end{aligned}$$

这里，所有的加法都是模 2 运算。类似地可以定义 S1 的方程。线性和非线性映射的交替作用使密文位的多项式表达非常复杂，从而使得密码分析很困难。为了清楚问题的规模，请注意 10 个未知二进制位所表示的多项式有 2^{10} 个可能项。平均而言，我们可以假定 8 个方程中的每一个都有 2^9 个项。有兴趣的读者可以尝试用符号处理器将这些方程都找出来，不过这件事对读者和软件来说都不是那么容易实现的。

3.1.5　与 DES 的联系

DES 对 64 位分组输入进行操作。加密可以定义为：

$$\mathrm{IP}^{-1} \circ \mathrm{f}_{K_{16}} \circ \mathrm{SW} \circ \mathrm{f}_{K_{15}} \circ \mathrm{SW} \circ \cdots \circ \mathrm{SW} \circ \mathrm{f}_{K_1} \circ \mathrm{IP}$$

算法采用 56 位的密钥，由此产生 16 个 48 位的子密钥。对 56 位数据进行初始置换后，接着对 48 位数据进行一系列的移位和置换。

加密算法中的 F 不再作用在 4 位二进制数 $(n_1 n_2 n_3 n_4)$ 上，而作用在 32 位 $(n_1 \cdots n_{32})$ 上。经过初始的扩散和置换，48 位输出可以表示成：

$$\begin{array}{c|cccc|c} n_{32} & n_1 & n_2 & n_3 & n_4 & n_5 \\ n_4 & n_5 & n_6 & n_7 & n_8 & n_9 \\ \vdots & & \vdots & & & \vdots \\ n_{28} & n_{29} & n_{30} & n_{31} & n_{32} & n_1 \end{array}$$

然后将该矩阵加到（异或运算）48 位的子密钥上。它的 8 行分别对应 8 个 S 盒，每个 S 盒有 4 行 16 列。上面矩阵的每行第一位和最后一位决定一个 S 盒中某元素的行数，而中间 4 位决定这个 S 盒中某元素的列数，这个元素就是本次运算的输出。

3.2 分组密码原理

事实上，现在使用的所有对称分组加密算法都基于 Feistel 分组密码结构[FEIS73]。因为这个原因，研究 Feistel 密码的设计原理就很重要了。我们将从比较流密码和分组密码开始，然后讨论它的一些含义。

3.2.1 流密码与分组密码

流密码每次加密数据流的一位或一个字节。古典流密码的例子有密钥自动产生的 Vigenère密码和 Vernam 密码。**分组密码**是将一个明文组作为整体加密且通常得到的是与之等长的密文组。典型的分组大小是 64 位或 128 位。本章后面还要讲到使用某些操作模式，分组密码可以获得与流密码相同的效果。

人们已经对分组密码进行了大量的研究。一般来说，分组密码的应用范围比流密码要广泛。绝大部分基于网络的对称密码应用使用的是分组密码。因此，我们在本章及本书中讨论的对称密码将集中在分组密码。

3.2.2 Feistel 密码结构的设计动机

分组密码作用在 n 位明文组，以产生 n 位密文组。共有 2^n 个不同的明文组，且由于加密是可逆的(即解密是可能的)，每一个明文组将惟一地对应一个密文组。这样的变换称为可逆变换，或非奇异变换。下面这个例子解释了 $n=2$ 时的非奇异变换和奇异变换。

可逆映射		不可逆映射	
明文	密文	明文	密文
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

在后一种变换中，密文 01 可由两个明文组中的任意一个产生。所以，如果我们限定在可逆映射上，不同变换的总数是 $2^n!$ 个。

图 3.4 给出了 $n=4$ 时的一个普通代换密码的结构。4 位的输入有 16 种可能的输入状态，每一种被代换密码映射成 16 种可能输出状态中的惟一一个，每一个表示 4 位的密文输出。加密和解密映射可如表3.1一样用表来定义。这是分组密码的最一般形式，能用来定义明密文之间的任意可逆变换。

但应用这种方法有着实际上的困难。对于像 $n=4$ 这样的较小分组，密码系统等价于传统代换密码。正如我们所见，用明文的统计分析方法攻击它是轻而易举的。这种脆弱性并非来自于代换密码，而是因为使用的分组规模太小。如果 n 充分大，并且有一个明密文之间的任意可逆变换，那么明文的统计特征将被掩盖，从而使得用来攻击这种体制不可行。

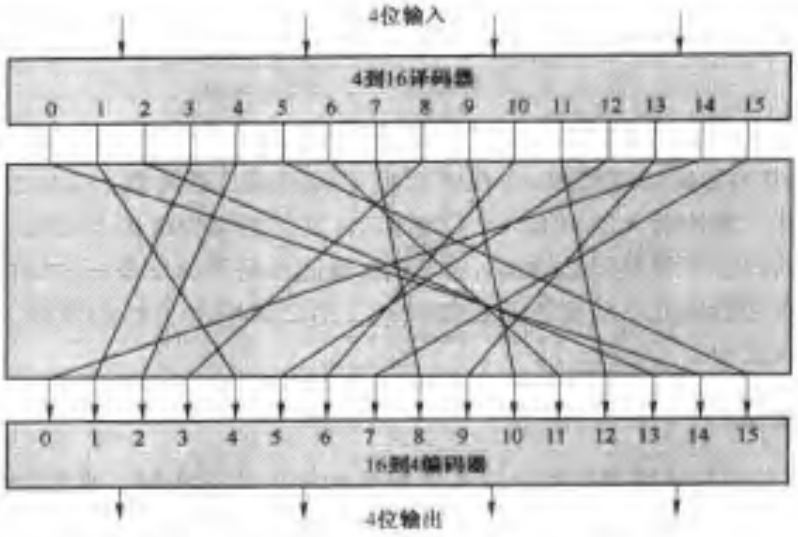


图 3.4 $n=4$ 时的一个 n 位-n 位分组密码

然而，从实现和运行的角度来看，使用大规模分组的任意可逆代换密码是不可行的。对于这样的变换，映射本身就是密钥。再考虑表 3.1，它定义了 $n=4$ 时的某个可逆映射。这个映射可以直接由表中的第二列来定义，它给出了每个明文对应的密文。本质上它决定所有可能映射中某一个映射的密钥。在这种情况下，密钥要求为 64 位。一般地，对于 n 位的代换分组密码，密钥的规模是 $n\times 2^n$ 位。一个 64 位的分组密码，若分组有抗统计攻击的理想长度，其密钥大小将需要 $64\times 2^{64}=2^{70}\approx 10^{23}$ 位。

表 3.1 图 3.4 所示代换密码的加密与解密表

明 文	密 文
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

明 文	密 文
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

考虑到这些困难，Feistel 指出我们所需的是对应较大 n 的理想分组密码体制的一种近似体制而已，它可以在容易实现部件的基础上逐步建立起来[FEIS75]。在讨论 Feistel 的方法之前，我们来进行其他的讨论。我们将讨论限定在一般分组代换密码，但为了实现起来方便，我们只讨论 $2^n!$ 个不同可逆映射的一个子集。例如，设我们按照线性方程的集合来定义映射。在 $n=4$ 时，我们有：

$$y_1 = k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4$$
$$y_2 = k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4$$
$$y_3 = k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4$$
$$y_4 = k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4$$

其中 x_i 是明文组中的四位二进制数，y_i 是密文组中的四位二进制数。k_{ij} 是二进制系数，所有运算都是模 2 运算。密钥的大小只是 n^2，这里为 16 位。如果为攻击者所知，这种公式表示的密码在密码分析的攻击下将是很危险的。本例中，我们只是将第 2 章所讨论的希尔(Hill)密码的知识运用到二进制数据而不是字符。正如我们在第 2 章中所了解的那样，这样一个简单的线性系统是较易攻破的。

3.2.3 Feistel 密码

Feistel 建议[FEIS73]使用乘积密码的概念来逼近简单代换密码。乘积密码是指依次使用两个或两个以上的基本密码，所得结果的密码强度将强于所有单个密码的强度。特别地，Feistel 建议交替地使用代换和置换。实际上，这是 Claude Shannon 提出的交替使用混淆和扩散乘积密码的实际应用。我们接下来看一看混淆和扩散的概念，然后研究 Feistel 密码。但是，首先请注意这样一个不平凡的事实：四分之一个世纪以前提出的、基于 1945 年 Shannon 理论的 Feistel 密码结构，是当前使用的大多数重要对称分组密码的基本结构。

混淆和扩散

Shannon 引进混淆和扩散这两个术语来刻画任何密码系统的两个基本构件[SHAN49]。[①] 他关注的是如何挫败基于统计方法的密码分析。理由是这样的：假设攻击者拥有明文统计特征的知识。例如某种人类语言的可读信息，其不同字母的频率分布是已知的。或者已知信息中极有可能出现某些单词或短语。如果这些特征以任何形式体现在密文中，密码分析者就有可能推导出密钥或其一部分，至少是包含确切密钥的一个密钥集。在 Shannon 所指的理想密码中，密文所有的统计特征都是独立于所用密钥的。我们前面所讲的任意代换密码就是这样一种密码，不过正如我们所见(图 3.4)，它是不可能获得实际应用的。

抛却求助于理想系统，Shannon 建议了两种对付统计分析的方法：扩散和混淆。**扩散**就是指使明文的统计特征消散在密文中，可以让每个明文数字尽可能地影响多个密文数字获得，等价于说每个密文数字被许多明文数字影响。一个扩散的例子是用如下方法加密一段消息 $M = m_1, m_2, m_3, \cdots$，其中 m 为字母。

$$y_n = \sum_{i=1}^{k} m_{n+i} (\text{mod } 26)$$

将 k 个连续字母相加得到密文字母 y_n。明文的统计结构因而消失了。故密文中各字母的频率比明文更趋于一致；双字母频率也是如此，等等。在二进制分组密码中，对明文进行置换后再用某个函数作用，重复多次就可获得较好的扩散效果；这来自于原始明文信息中不同位对

① Shannon 于 1949 年发表的论文，最初是 1945 年的一个保密报告。他在计算机和信息科学史上占有杰出的地位。他不仅奠定了现代密码学的基本思想，而且他还是信息论的开创者。他还创立了另一门学科，即把布尔代数用于数字电路的研究，这是他在自己的硕士论文中提出来的。

密文某一位产生的影响。[①]

每一个分组密码都是明文组到密文组的变换，而这个变换又是依赖于密钥的。扩散的方法是尽可能地使明文和密文间的统计关系变得复杂，以挫败推导出密钥的企图。另一方面，**混淆**则是尽可能使密文和加密密钥间的统计关系更加复杂，以阻止攻击者发现密钥。因此，即使攻击者拥有一些密文的统计特征信息，利用密钥产生密文的方法复杂性使得推导密钥极其困难。这可以用一些复杂的代换算法来实现，简单的线性代换函数几乎增加不了混淆。

正如文献[ROBS95b]指出的那样，扩散和混淆正是抓住了设计分组密码的本质而成为现代分组密码设计的里程碑。

Feistel 密码结构

图 3.5 描述了 Feistel 提出的结构。加密算法的输入是长为 $2w$ 位的明文组和密钥 K。明文组被分为两部分：L_0 和 R_0。这两半数据经过 n 轮迭代后组合成密文组。第 i 轮迭代的输入 L_{i-1} 和 R_{i-1} 来自于上轮迭代的输出；而于密钥 K_i 是由整个密钥 K 推导出的。一般地，K_i 不同于 K，也互不相同。

每轮迭代都有相同的结构。**代换**作用在数据的左半部分。它通过用轮函数 F 作用数据的右半部分后，与左半部分数据进行异或来完成。每轮迭代的轮函数是相同的，但是输入的子密钥 K_i 不同。代换之后，交换数据的左右两半完成**置换**。[②] 这种网络是 Shannon 提出的代换置换网络(substitution-permutation network，SPN)的一种特别形式。

Feistel 结构的具体实现依赖于以下参数和特征：

- **分组长度**：分组越长意味着安全性越高(其他数据不变)，但是会降低加、解密的速度。64 位的分组长度比较合理，能迎合广泛分组密码的要求。高级加密标准(AES)使用的是 128 位的分组长度。
- **密钥长度**：密钥较长同样意味着安全性较高，但会降低加、解密的速度。现在一般认为 64 位的密钥还不够。通常使用的密钥长度是 128 位。
- **迭代轮数**：Feistel 密码的本质在于单轮不能提供足够的安全性，而多轮加密可取得很高的安全性。迭代轮数的典型值是 16。
- **子密钥产生算法**：子密钥产生越复杂，密码分析攻击就越困难。
- **轮函数**：同样，轮函数越复杂，抗攻击的能力就越强。

设计 Feistel 密码还有两个其他方面的考虑：

- **快速软件加/解密**：许多情况下，加密算法被植到应用程序工具中，而做成硬件不太方便。因此，算法执行的速度很重要。
- **简化分析难度**：尽管我们喜欢把算法设计得不易受到密码分析的攻击，但是设计较简单的算法却有利于我们进行分析。也就是说，如果算法描述起来简洁清楚，那么分析其

① 某些密码学的书把置换和扩散等同起来，这是不正确的。置换本身并不改变明文在单个字符或置换块上的统计特性。例如，在 DES 中用置换交换两个 32 位数据块，因而 32 位数据块内的统计特性将保持不变。

② 在最后一轮迭代后有一个交换，以抵消在最后一轮迭代中的那个交换。你可以从图中去掉这两个交换，但这样会使问题的表述不一致。我们将会看到，在任何情况下去掉最后一轮中的交换将使解密更简单。

脆弱性也就容易一些，因而可以开发出更强的算法。不过 DES 并没有容易的分析办法。

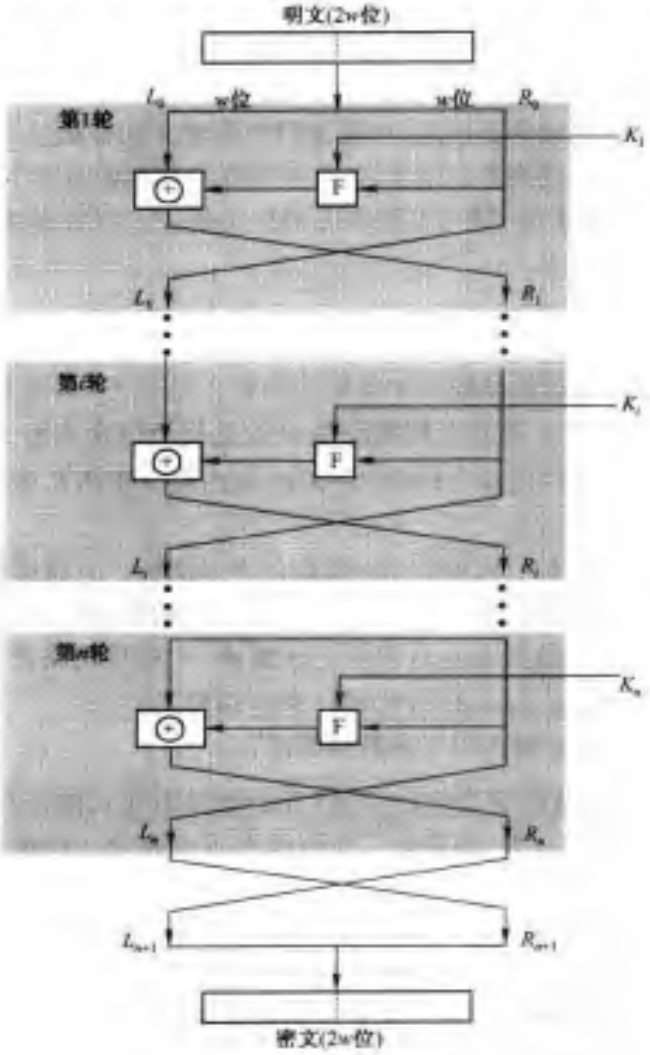


图 3.5 古典 Feistel 网络

再回来研究图 3.1 和图 3.3，我们可以看出两轮 S-DES 具有的 Feistel 结构。与典型的 Feistel 结构相比，算法有所不同，那就是在开始和结束处都有一个置换函数。这个差别也出现在完整的 DES 中。

Feistel 解密算法

Feistel 密码的解密过程本质上与加密过程一致。其规则如下：将密文作为算法的输入，但是逆序使用子密钥 K_i。也就是说，第一轮使用 K_n，第二轮使用 K_{n-1}，直到最后一轮使用 K_1。这是一个很好的特点，因为这意味着我们不需要实现两个算法：一个用做加密，另一个用做解密。

图 3.6 表示了用逆序密钥和相同算法解密的过程。加密过程在图的左边，自上而下，而解密过程在右边，自下而上进行了 16 轮算法(无论多少轮结果都是相同的)。为清楚起见，我们

用 LE_i 和 RE_i 表示加密过程的中间数据，而用 LD_i 和 RD_i 表示解密过程的中间数据。图中表明，每轮的解密过程中间值与加密过程中间值左右互换的结果是相同的。换句话说，第 i 轮加密的输出是 $LE_i \| RE_i$，解密的第$(16-i)$轮的相应输入则是 $RE_i \| LE_i$ 或 $RD_{16-i} \| LD_{16-i}$。

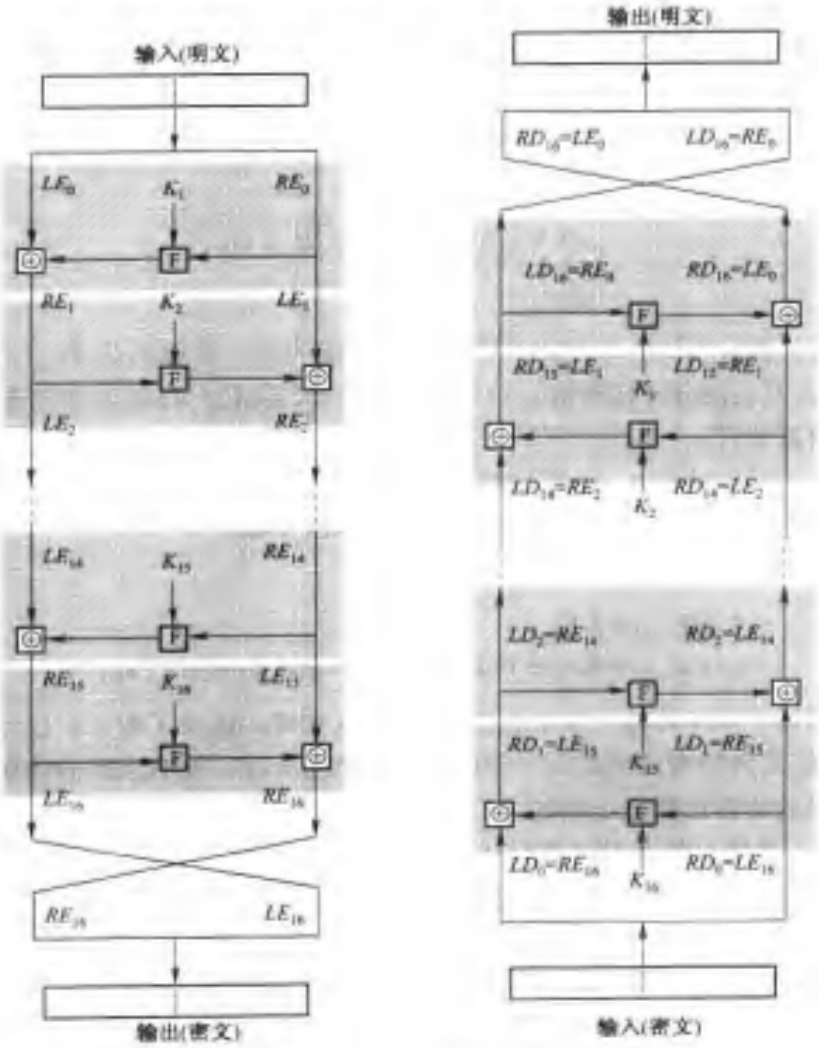


图 3.6　Feistel 加密与解密

我们按照图 3.6 来证明上面所说的正确性。[①] 加密过程的最后一轮迭代后，输出数据的左右两部分互换，所以密文是 $RE_{16} \| LE_{16}$。现在将它作为同一个算法的输入。第一轮的输入就是 $RE_{16} \| LE_{16}$，它应等于加密过程第 16 轮输出左右部分互换的值。

现在我们来证明解密过程第一轮的输入等于加密过程第 16 轮输出左右部分互换的值。首先，对于加密过程有：

① 为简单起见，这里已把图拉开，且未画出每一轮迭代后的交换。但是请注意，加密过程第 i 轮的中间结果是由 LE_i 和 RE_i 连接组成的 $2w$ 位数据。而解密过程第 i 轮的中间结果是由 LD_i 和 RD_i 连接组成的 $2w$ 位数据。

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$$

对于解密则有：

$$\begin{aligned} LD_1 &= RD_0 = LE_{16} = RE_{15} \\ RD_1 &= LD_0 \oplus \mathrm{F}(RD_0, K_{16}) \\ &= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16}) \end{aligned}$$

XOR 运算有以下性质：

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$
$$D \oplus D = 0$$
$$E \oplus 0 = \mathrm{E}$$

因此我们有 $LD_1 = RE_{15}$及 $RD_1 = LE_{15}$。所以解密过程的第一轮输出为 $LE_{15} \parallel RE_{15}$，正是加密过程第 16 轮输入左右部分互换的值。对于其他各轮亦是如此，我们可把它表示成一般形式。对于第 i 轮加密算法有：

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, \mathrm{K}_i)$$

又可写为，

$$RE_{i-1} = LE_i$$
$$LE_{i-1} = RE_i \oplus \mathrm{F}(RE_{i-1}, \mathrm{K}_i) = RE_i \oplus \mathrm{F}(LE_i, \mathrm{K}_i)$$

因此我们已经描述了第 i 轮输入输出的函数关系，这些等式证实了图 3.6 右半部分的正确性。

最后，我们注意到解密过程最后一轮的输出是 $RE_0 \parallel LE_0$，左右互换的结果正是原始明文，说明 Feistel 密码的解密过程是正确的。

注意，我们在推导过程并没有要求 F 函数是可逆的。为了解这一点，我们取一种极端情况，这种情况下 F 的输出为一个常数(例如全为 1)，等式依然成立。

3.3 数据加密标准

使用最广泛的加密体制是数据加密标准(DES)，它于 1977 年被美国国家标准局即现在的国家标准和技术研究所(NIST)采纳为联邦信息处理标准 46(FIPS PUB 46)。这个算法本身被称为数据加密算法(DEA)。[①] DES 采用了 64 位的分组长度和 56 位的密钥长度，它将 64 位的输入经过一系列变换得到 64 位的输出。解密则使用了相同的步骤和相同的密钥。

DES 使用得很广泛。DES 的安全强度究竟如何一直是争论的话题。为了理解争论的原因，我们先简要回顾一下 DES 的历史。

在 20 世纪 60 年代后期，IBM 公司成立了一个由 Horst Feistel 负责的计算机密码编码学研

① 术语有点混乱，直到现在还经常将 DES 和 DEA 互换使用。最近 DES 的文件中关于 DEA 的描述增加了 3 重 DEA (TDEA)，关于 3 重 DEA 我们将在第 6 章讲述。DEA 和 TDEA 都是 DES 的组成部分。最近，官方术语采用了 TDEA，并把 3 重 DEA 称为 3 重 DES，简写为 3DES。为了方便，我们采用 3DES 的说法和写法。

究项目。1971 年设计出算法 LUCIFER 后[FEIS73],该项目宣告结束。LUCIFER 被卖给了伦敦的 Lloyd 公司,用在同样由 IBM 公司开发的现金发放系统上。LUCIFER 是分组长度为 64 位、密钥长度为 128 位、具有 Feistel 结构的分组密码。因为 LUCIFER 非常成功,IBM 决定开发一个适合于芯片实现的商业密码产品。这一次由 Walter Tuchman 和 Carl Meyer 牵头,参与者不仅有 IBM 公司的研究人员,而且有 NSA 的技术顾问。这次努力的结果是给出了 LUCIFER 的一个修订版,它的抗分析攻击能力更强而且密钥长度减小为 56 位,因而很适合制成一个专用 DES 密码芯片。

1973 年美国国家标准局(NBS)征求国家密码标准方案。IBM 将 Tuchman-Meyer 方案提交给了 NBS,它是所有应征方案中最好的一个,所以 1977 年 NBS 将它采纳为数据加密标准,即 DES。

DES 在被采纳为标准之前,曾受到过激烈的批评,其实直到今天亦未平息。批评主要集中在两个方面:第一,IBM 公司最开始的 LUCIFER 算法的密钥长度是 128 位,而 DES 却只使用了 56 位的密钥,整整减少了 72 位。批评者一直担心密钥太短而不能抗穷举攻击。第二,DES 的内部结构,即 S 盒的设计标准被列入官方机密。所以,用户不能确信 DES 的内部结构是没有弱点的,NSA 有可能利用这些弱点在没有密钥的情况来解密。后来的事件,特别是近年来差分分析方面的工作表明,DES 的内部结构是强健的。而且按照 IBM 的参与者所说,原始算法中惟一做过改动的是 S 盒,这是由 NSA 建议的,它去掉了测试过程中所发现的算法的一些脆弱性。

不管这件事的价值如何,DES 已经风行全世界了,特别是在金融领域的应用。1994 年 NIST 重新决定将 DES 联邦使用期延长 5 年。NIST 推荐在一般商业应用中使用 DES,而不用 DES 来保护官方机密。1999 年 NIST 颁布了标准(FIPS PUB 46-3)的新版本,新版本规定了 DES 只能用于(历史)遗留系统以及 3DES 的使用情况。我们将会在第 6 章研究 3DES。因为 DES 与 3DES 的加、解密算法是相同的,所以理解 DES 算法仍然有很重要的意义。

3.3.1　DES 加密

图 3.7 表明了 DES 加密的整个机制。对于任意加密方案,总有两个输入:明文和密钥。DES 的明文长为 64 位,密钥长为 56 位。[①]

从图中左半部分,可见明文的处理经过了三个阶段。首先,64 位的明文经过初始置换(IP)而被重新排列。然后进行 16 轮的相同函数的作用,每轮的作用中都有置换和代换。最后一轮迭代的输出有 64 位,它是输入明文和密钥的函数。将其左半部分和右半部分互换产生预输出。最后,预输出再被与初始置换(IP)互逆的逆初始置换(IP^{-1})作用产生 64 位的密文。除了初始和末尾置换外,DES 的结构与图 3.5 所示的 Feistel 密码结构完全相同。

图 3.7 的右半部分给出了使用 56 位密钥的过程。开始时,密钥经过一个置换,然后经过循环左移和另一个置换分别得到子密钥 K_i,供每一轮的迭代加密使用。每轮的置换函数都一样,但是由于密钥位的重复迭代使得子密钥互不相同。

① 实际上这个密码函数希望采用 64 位的密钥。然而却仅仅采用了 56 位,其余 8 位可以用做奇偶校验或随意设置。

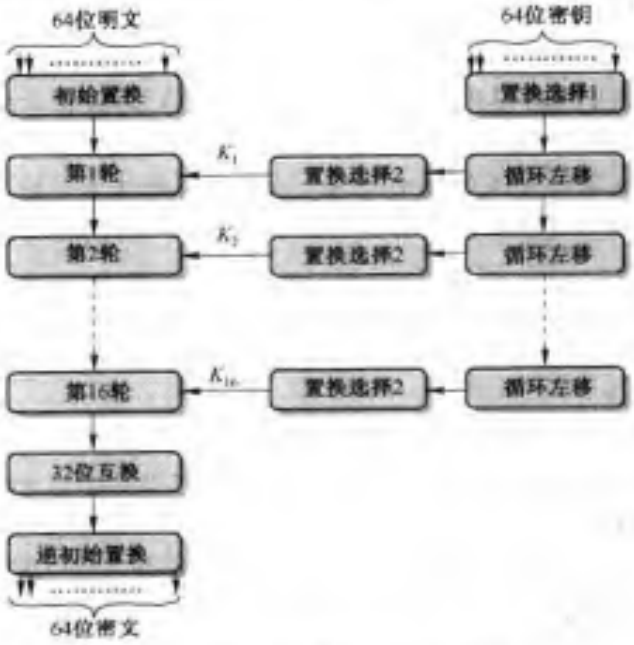


图 3.7 DES 加密算法的一般描述

初始置换

表 3.2(a)和 3.2(b)分别定义了初始置换及其逆置换,其解释如下。表的输入标记从 1 到 64,共 64 位。置换表中 64 个元素代表从 1 到 64 这些数的一个置换。置换表中的每个元素表明了某个输入位在 64 位输出中的位置。

表 3.2 DES 的置换表

(a) 初始置换 (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) 逆初始置换 (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) 扩充置换 (E)　　(续表)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) 置换函数 (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

为了说明这两个变换的确是互逆的，考虑下面这个 64 位的输入 M：

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

这里 M_i 是二进制数。经过置换 $X = \mathrm{IP}(M)$之后，得到的 X 是：

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

如果我们再对它作用逆初始置换 $Y = \mathrm{IP}^{-1}(X) = \mathrm{IP}^{-1}(\mathrm{IP}(M))$，就会恢复出 M。

单轮变换的详细过程

图 3.8 给出了一轮变换的内部结构。我们同样先看图的左半部分。64 位中间数据的左右两部分作为独立的 32 位数据，分别记为 L(左)和 R(右)。在任何古典 Feistel 密码中，每轮变换的整个过程可以写为下面的公式：

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

该轮的密钥 K_i 长 48 位。R 输入是 32 位。首先将 R 用一个表定义的置换扩展为 48 位，其中有 16 位是重复的[见表 3.2(c)]。这 48 位与 K_i 异或，再用一个代换函数[见表 3.2(d)]作用产生 32 位的输出。

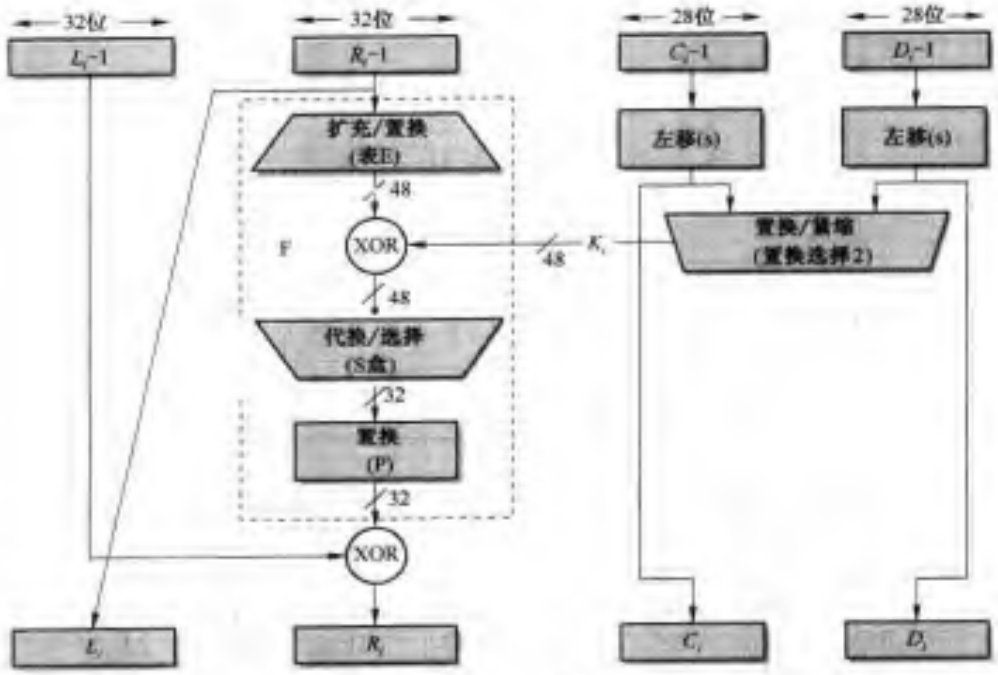


图 3.8 DES 算法一轮迭代的过程

图 3.9 解释了 S 盒在函数 F 中的作用。代换函数由 8 个 S 盒组成，每个 S 盒都由 6 位输入产生 4 位输出。这些变换见表 3.3，其解释如下：盒 S_i 输入的第一位和最后一位组成一个 2 位的二制数，用来选择 S 盒 4 行中的一行，中间 4 位用来选择 16 列中的某一列。行列交叉处的十进制值转换为二进制之后可得到输出的 4 位二进制表示。例如，在 S_1 中，若输入为 011001，则行是 01(1)，列是 1100(12)。该处的值是 9，所以输出为 1001。

表 3.3 DES 的 S 盒的定义

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

（续表）

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

S 盒的每行都定义了一个普通的可逆代换。图 3.4 对于理解这里面的映射关系可能有帮助。图中显示了盒 S_1 第 0 行的代换关系。

S 盒的操作值得进一步评论。暂且不管子密钥 K_i 的作用。如果仔细研究扩展表，就会发现 32 位输入被 4 位 4 位地分成了 8 组，然后每组从相邻的两组中取得位置靠近的 4 位，而变成 6 位。例如，如果输入字的部分为：

... efgh ijkl mnop ...

它将变为：

... defghi hijklm lmnopq ...

每组的外面两位实际上决定了4种选择中的一种(S盒中的一行)。那么4位输出代换了输入中特殊的4位(输入的中间4位)。8个S盒的32位输出经过置换,使得每个S盒的输出在下一轮中尽可能地影响更多(数据的)其他位。

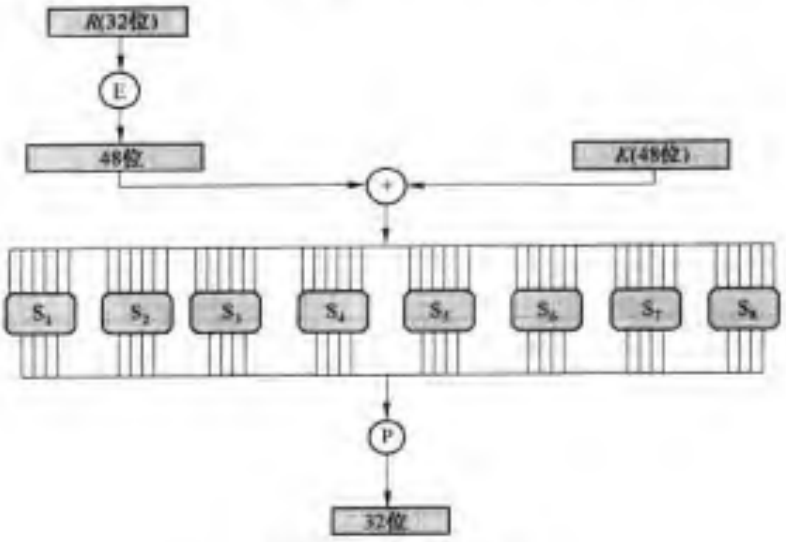


图3.9 F(R,K)的计算

密钥产生

回到图3.7和图3.8中,我们知道56位密钥作为算法的输入。密钥的各位分别标记为1到64。如表3.4(a)中没有阴影的部分,也就是每行第8个位被忽略。首先用置换选择1的表[表3.4(b)]进行置换。所得56位密钥分为两个28位数据 C_0 和 D_0。每轮迭代中,C_{i-1} 和 D_{i-1} 分别左循环移位一位或两位,见表3.4(d)。移位后的值作为下一轮的输入。它们同时也作为置换选择2[表3.4(c)]的输入,产生一个48位的子密钥,并作为函数 $F(R_{i-1}, K_i)$ 的输入。

表3.4 DES密钥的使用

(a) 输入密钥

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) 置换选择 1 (PC-1)　(续表)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) 置换选择 2 (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) 对左移次数的规定

迭代轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
移位次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3.3.2 DES 解密

作为 Feistel 密码，DES 的解密算法与加密算法是相同的，只是子密钥的使用次序相反。

3.3.3 雪崩效应

明文或密钥的微小改变将对密文产生很大的影响是任何加密算法所期望的一个好性质。特别地，明文或密钥的某一位发生变化会导致密文的很多位发生变化。如果相应的改变很小，可能会给分析者提供缩小搜索密钥或明文空间的渠道。

DES 显示出很强的雪崩效应。表 3.5 给出了文献[KONH81]中的一些结果。表 3.5(a)使用的两条仅有一位不同的明文是：

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

所用的密钥为：

0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

结果表明仅经过 3 轮迭代后，两段数据有 21 位不同。全部迭代完成后得到的两段密文则有 34 位不同。

表 3.5(b)做了一个近似的实验。同一段明文：

01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

用两个仅相差一位的密钥

1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100
0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

加密。结果表明,经过数层变换后发生了雪崩效应,即两段密文中有半数的位不相同。

表 3.5 DES 的雪崩效应

(a)明文的变化		(b)密钥的变化	
轮数	改变的位数	轮数	改变的位数
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

3.4 DES 的强度

自从 DES 被采纳为联邦标准,对它的安全性就一直争论不休,焦点主要集中于密钥的长度和算法本身的安全性。

3.4.1 56 位密钥的使用

56 位的密钥共有 2^{56} 种可能,这个数字大约为 7.2×10^{16}。所以穷举攻击明显是不太实际的。平均而言,搜索一半密钥空间,一台每毫秒执行一次 DES 加密的计算机需要用一千年才能破译出密文(见表 2.2)。

然而,每毫秒执行一次加密运算的假设过于保守。早在 1977 年 Diffie 和 Hellman 就指出,若用现有的技术去制造一台并行机,它带有 100 万个加密器,每一个加密器都可以在 1 毫秒内执行一次加密运算[DIFF77],那么平均的穷举时间将减少为大约 10 个小时。他们估计这台计算机的造价在当时可能为 2000 万美元。

1998 年 7 月,当 EFF(Electronic Frontier Foundation)宣布一台造价不到 25 万美元的专用"DES 破译机"破译了 DES 时,DES 终于清楚地被证明是不安全的。这次攻击所花的时间不到

三天。EFF 还公布了这台机器的细节，使其他人也能建造自己的破译机[EFF98]。当然，随着速度的提高，硬件的造价将会下降，最终会导致 DES 毫无价值。

不过，要进行真正的穷举攻击，仅仅靠简单地将所有可能的密钥代入到程序中去执行是不够的。如果不提供已知明文，分析者必须能够从不同密钥解密的所谓明文中辨认出真正的明文。如果消息是用英文写的，即使要达到自动辨认，也是很好办的。如果文本信息在加密之前进行了压缩，那就太困难了。或者原始信息是更一般的数据类型，比如数据文件，还进行了压缩，那么辨认起来就几乎不可能实现自动化。因此，要进行穷举攻击，需要事先知道一些有关期望明文的知识，并且需要将正确的明文从可能的明文堆里辨认出来的自动化方法。EFF 也介绍了在很多环境中很有效的自动化技术。

幸运的是，有大量 DES 的代换算法，最重要的有高级加密标准(AES)和 3DES，我们将在第 5 章、第 6 章分别加以讨论。

3.4.2　DES 算法的性质

人们关心的另外一件事是，密码分析者有没有利用 DES 算法本身的特征来攻击它的可能性。问题集中在每轮迭代所用的 8 个代换表，即 S 盒身上。因为这些 S 盒的设计标准，实际上包括整个算法的设计标准是不公开的，因此人们怀疑密码分析者若是知道 S 盒的构造方法，就可能知道 S 盒的弱点。这种说法很令人困惑，多年来，人们的确发现了 S 盒的许多规律和一些缺点。尽管如此，至今还没有人发现 S 盒存在致命的弱点。[①]

3.4.3　计时攻击

由于计时攻击与公钥算法有关，我们将在第二部分详细讨论；然而它也与对称密码相关。本质上计时攻击是通过对执行给定的多种密文解密所需时间的观察，来获得关于密钥与明文的信息。计时攻击所利用的事实是加密或解密算法对于不同的输入所花的时间有着细微的差别。文献[HEVI99]给出了产生密钥汉明重量的一种方法。这离知道实际密钥还很遥远，但这的确是很能激发人们兴趣的一步。作者认为 DES 能够很好地抵抗计时攻击，但是建议从其他途径考虑利用它。尽管这是很有趣的攻击路线，但是到目前为止，它还不可能成功地攻击 DES，更不能攻击如 3DES 和高级加密标准(AES)等的对称密码。

3.5　差分分析和线性分析

自从 DES 出台后，人们就很关心它究竟能否经受住穷举攻击。毕竟，它的密钥长度只有 56 位。不过，用密码分析的方法来攻击 DES 也是很有趣的事情。随着包括 3DES 在内的长密钥分组密码的逐渐增多，穷举攻击自然是越来越不实际。因此用密码分析的方法来攻击 DES 及其他分组密码就越来越受人们重视了。本节中，我们将对其中两种最有希望的强有力分析方法做一个概述，它们是差分密码分析和线性密码分析。

---

① 至少没有人公开宣布这样的发现。

3.5.1 差分密码分析

近年来密码分析领域最重要的发展就是差分分析。本节中我们对此进行讨论，并将其应用于 DES。

历史

直到 1990 年，差分密码分析才公开发表。已知最早的是 Murphy 用它来分析 FEAL [MURP90]。接着 Biham 和 Shamir 发表了数篇文章，对多种加密算法和哈希函数进行了此类攻击，结果汇集在文献[BIHA93]中。

大多数发表的结果被用于分析 DES。差分密码分析是第一个公开的、能对 DES 在小于 2^{55} 种复杂性情况下攻击成功的方法。[BIHA93]中有整个攻击方案。它表明，若有 2^{47} 个选择明文，用差分密码分析就可以在 2^{47} 次内成功攻击 DES。尽管 2^{47} 比 2^{55} 小得多，但是要拥有 2^{47} 个选择明文的条件使得这种方法只具有理论上的意义。

尽管差分密码分析是一个强有力的密码攻击方法，但它对 DES 并不奏效。根据设计 DES 的 IBM 小组一位成员所述[COPP94]，原因是该小组早在 1974 年就知道了差分密码分析。为了提高 DES 抗差分攻击的能力，在设计 S 盒和置换 P 的时候做了充分考虑。[BIHA93]中对这些做了相应的论述，可以证明知道差分密码分析之后对设计 DES 会产生很大的影响。用差分密码分析对 8 层迭代的LUCIFER算法只需 256 个选择明文，而若只有 8 层迭代的 DES 则需 2^{14} 个选择明文。

差分密码分析攻击

差分密码分析攻击非常复杂，文献[BIHA93]对此有详细的描述。这里，我们只对它进行简单的描述，只要能看出它的特点就行了。

下面讨论 DES 的算法中符号有所改变。设原始明文组 m 被分成两部分，即 m_0、m_1。DES 的每一轮迭代都将其右半部分映射为下一轮迭代中的左半部分，而将左半部分与于密钥进行运算后作为下一轮迭代的右半部分。所以在每一轮迭代中，仅产生 32 位新数据。如果我们将其记为 $m_i(2 \leqslant i \leqslant 17)$，可得以下关系式：

$$m_{i+1} = m_{i-1} \oplus \mathrm{f}(m_i, K_i), \qquad i = 1, 2, \cdots, 16$$

在差分密码分析中，我们从两条已知异或差分 $\Delta m = m \oplus m'$ 的明文 m 和 m' 开始，考虑中间数据的差分 $\Delta m_i = m_i \oplus m_i'$。我们有：

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]\end{aligned}$$

现在，假设函数 f 有许多对具有相同差分的输入，当使用相同的子密钥时产生相同差分的输出。更精确地说，若差分为 X 的所有输入，产生差分为 Y 的输出占所有输出的百分比为 p，我们就称由差分 X 导致差分 Y 的概率为 p。我们假设有许多 X 都会以很高的概率产生某些特定的差分。因此，若我们以很高的概率知道 Δm_{i-1} 和 Δm_i，就能以很高的概率知道 Δm_{i+1}。而且，如果知道很多有关差分的数据，那么确定函数 f 所使用的子密钥就是可行的。

基于单轮的以上考虑就是差分密码分析的策略。步骤是这样的：两段具有给定差分的明文 m 和 m'，跟踪它们在每一轮迭代后产生的差分与之前的差分形成的概率模型。实际上输出的两个 32 位数据有两个概率差分($\Delta m_{17} \| \Delta m_{16}$)。然后我们给出 m 和 m' 在未知密钥下加密的实际差分，将结果与概率差分比较。如果它们是匹配的，即

$$E_K(m) \oplus E_K(m') = (\Delta m_{17} \| \Delta m_{16})$$

那么我们可猜测所有中间各轮的所有概率模型是正确的。根据这个假设，我们可以对密钥的某些位进行推导。这个过程可能要重复很多次，才能确定整个密钥。

图 3.10 来自于文献[BIHA93]，它表明了 DES 经过三轮迭代后差分的扩散情况。图右边的概率指给定的中间差分作为输入差分的函数的概率。经过三轮迭代后如图给出的输出差异的概率为 $0.25 \times 1 \times 0.25 = 0.0625$。

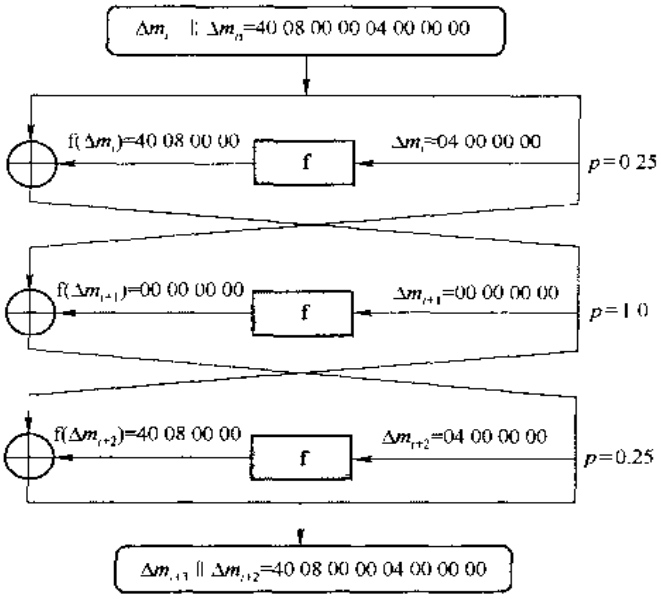


图 3.10　经过三轮迭代的差分传播(图中的数字为 16 进制)

3.5.2　线性密码分析

最近的一个密码分析方法是[MATS93]中所描述的线性密码分析。这种方法通过寻找 DES 变换的线性近似来进行攻击。这种方法只需知道 2^{47} 个明文就可以找出 DES 的密钥，而用差分密码分析则需要知道 2^{47} 个选择明文。尽管获取明文比获取选择明文容易得多，但这只是一个小的改进，线性密码分析对于攻击 DES 还是不可行的。到现在为止，几乎没有取得线性密码分析的有效进展。

先简要叙述一下线性密码分析的基本原理。对于 n 位的明文组和密文组以及 m 位密钥的密码，记明文组为 P[1],…,P[n]，密文组为 C[1],…,C[n]，密钥为 K[1],…,K[m]。那么定义：

$$A[i, j, \cdots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

线性密码分析的目标是找到一个下列形式的有效线性等式：

$$P[\alpha_1, \alpha_2, \cdots, \alpha_a] \oplus C[\beta_1, \beta_2, \cdots, \beta_b] = K[\gamma_1, \gamma_2, \cdots, \gamma_c]$$

(其中 $x=0$ 或 1；$1 \leqslant a, b \leqslant n, 1 \leqslant c \leqslant m$，$\alpha$、$\beta$ 和 γ 代表固定且惟一的位位置)要求上式成立的概率为 $p \neq 0.5$。p 离 0.5 越远，等式就越有效。一旦确定了这种关系，实际过程就是对大量明密文对计算上述等式的左半部分。若多数情况下结果为 0，则假设 $K[\gamma_1, \gamma_2, \cdots, \gamma_c] = 0$，若多数情况下结果为 1，则假设 $K[\gamma_1, \gamma_2, \cdots, \gamma_c] = 1$。这就给出了密钥位的一个线性等式。尽量获得较多的这种等式，以便我们从中解出密钥。因为我们处理的是线性等式，所以可以一次处理一轮迭代，把这些结果组合起来就能解决我们的问题。

3.6 分组密码的设计原理

尽管在设计强分组密码方面有了很大的进展，但是自 20 世纪 70 年代早期 Feistel 和 DES 设计小组所做的工作以来，基本设计原理并没有改进。在讨论之前，我们先来看看 DES 的公开设计准则，然后再看看设计分组密码的三个主要问题：加密轮数、函数 F 的设计以及密钥的使用方案。

3.6.1 DES 的设计准则

文献[COPP94]给出了 DES 的设计准则，主要针对 S 盒及 P 函数的设计。S 盒的设计准则如下：

1. 任何 S 盒的输出位都不应太接近于一个输入位的线性函数。特别地，如果我们选择了任何一个输出位和任何 6 位输入的集合，输入位的异或值等于这个输出位的输入占总输入的比例既不应接近 0 也不应接近 1，而应该接近于 0.5。
2. S 盒的每行(由输入的最左和最右位决定的固定值)应包含所有 16 种可能的输出位组合。
3. 对 S 盒的 2 个输入，若仅有一位不同，输出至少有 2 位不同。
4. 对 S 盒的 2 个输入，若中间 2 位不同，输出至少有 2 位不同。
5. 对 S 盒的 2 个输入，若前面 2 位不同而后 2 位相同，输出不应相同。
6. 对任意有着 6 位非零差分的输入对，32 对输入中至多有 8 对有着相同的输出差分。
7. 这条准则与前一条类似，但是对三个 S 盒的情形而言。

Coppersmith 指出，上述第 1 条准则的必要性在于 S 盒是 DES 中惟一的非线性部分。如果 S 盒是线性的(即 S 盒的输出是输入的线性组合)，那么整个算法就是线性的，很容易攻破。这种现象我们在讨论线性 Hill 密码时已经看到了。其余的准则则提供较好的混淆性质来抗差分攻击。

置换 P 的准则如下：

1. 在第 i 轮迭代 S 盒的 4 位输出中，有 2 位影响第 $i+1$ 层的中间 2 位，另 2 位则影响两端位。在 S 盒的 6 位输入中，中间 2 位不与相邻的 S 盒共用，而左右各 2 位则与相邻的 S 盒共用。
2. S 盒的 4 位输出影响下一轮的 6 个不同 S 盒，且没有 2 位能影响相同的 S 盒。

3. 对 S 盒 j 和 k，若 S_j 的某位输出影响了下一轮 S_k 的中间某位，那么 S_k 的某位输出不能影响 S_j 的中间某位。这隐含着对于 $j = k$，S_j 的输出某位不影响它的中间某位。

这些准则的目的在于增加算法的扩散程度。

3.6.2　迭代轮数

Feistel 密码的密码编码强度来自于三个方面：迭代轮数、函数 F 和密钥使用的算法。首先我们来看轮数的选择。

迭代轮数越多，密码分析就越困难，即使是 F 相对较弱也同样适用。一般说来，迭代轮数的选择准则是使密码分析的难度大于简单穷举攻击的难度。这个准则当然也用在了 DES 的设计中。Schneier(见[SCHN96])观察到，对 16 层迭代的 DES，差分密码分析比穷举攻击的效率要差一点：差分密码分析需要 $2^{55.1}$ 次操作，[①] 而穷举攻击则平均需要 2^{55} 次操作。如果 DES 只有 15 层迭代或更少，差分密码分析比穷举攻击的效率就要高一些。

这个准则是很有吸引力的，因为它使得判别算法强度和比较算法优劣变得很容易。如果在密码分析方面没有突破，则任何满足这个准则的算法强度仅需要从密钥长度判断。

3.6.3　函数 F 的设计

Feistel 密码的核心是函数 F。正如我们在 DES 中所见，这个函数依赖于 S 盒的使用。大多数分组对称密码皆是如此，我们将会在第 6 章中看到这一点。这里我们先对函数 F 的设计准则做一般评价，再来专门讨论 S 盒的设计。

函数 F 的设计准则

函数 F 给Feistel密码注入了混淆的成分，所以必须难以将函数 F 实行代换后的结果破译为原码。函数 F 的明显准则是非线性，这一点在前面我们已经讨论过。函数 F 的非线性成分越多，任何形式的分析就会越困难。非线性化的方法有很多种，这都超出了本书的主题，在此不多赘述。粗略说来，越难将函数 F 近似表示为某些线性等式，函数 F 的非线性度就越高。

设计函数 F 时还应考虑其他几个准则。我们希望算法有较好的雪崩效应，即输入的一位变化应该引起输出的很多位变化。一个更严格的定义是**严格雪崩效应准则**(strict avalanche criterion，SAC)，见文献[WEBS86]，对于所有的 i 和 j，它要求若 S 盒的输入的任意一位 j 发生变化，输出的任意一位 i 发生变化的可能性为 1/2。尽管 SAC 是对 S 盒而言的，作为一个准则它同样适用于整个函数 F，即使函数 F 中不含 S 盒。

文献[WEBS86]中建议的另一项准则是**位独立准则**(bit independence criterion，BIC)，它指对于所有的 i、j、k，当输入中 i 位变化时，输出中 j 位和 k 位的变化应是彼此无关的。SAC 和 BIC 明显是为了加强混淆的有效性。

---

① 前面讲过，对 DES 进行差分密码分析需要 2^{47} 个选择明文。如果仅有已知明文，就需要稍微多一点的已知明密文对。这样就使工作量上升到 $2^{55.1}$。

S 盒的设计

对称分组密码的研究中，S 盒的设计是最热门的领域之一，有关方面的文献亦是浩如烟海。[①] 这里我们叙述几个一般原理。本质上，我们希望输入矢量的任何变化导致 S 盒的输出产生近乎随机的变化，这种关系应是非线性的，并且难以用线性函数逼近。

S 盒的明显特征是其大小。$n \times m$ 的 S 盒有 n 位输入，m 位输出。DES 的 S 盒是 6×4，而第 6 章将讨论的 Blowfish 算法采用 8×32 的 S 盒。一般说来，S 盒越大，抗差分密码分析和线性密码分析的能力就越强[SCHN96]。而另一方面，n 越大，查找表也就越大。此外，S 盒越大，设计起来也就越困难。所以，出于实际应用考虑，n 通常在 8 到 10 之间。

S 盒的组织通常与 DES 中 S 盒的方式不同。一个 $n \times m$ 的 S 盒一般包括 2^n 行，每行有一个 m 位的元素，输入的 n 位可用来选择行数，而该行的 m 位元素则是输出。例如，一个 8×32 的 S 盒，输入为 00001001，输出则是第 9 行的那个 32 位元素（第 1 行标为行 0）。

Mister 和 Adams[MIST96]提出了一系列 S 盒的设计准则，包括 S 盒应该满足 SAC 和 BIC。他们还建议 S 盒各列的线性组合应该是符合 bent 函数的。bent 函数是一种特殊的布尔函数，根据一定的数学准则，它具有很高的非线性。用 bent 函数来设计和分析 S 盒已引起了人们极大的兴趣。

文献[HEYS95]提出并分析了 S 盒的一个相关设计准则。作者如下定义**保证雪崩准则**（GAC）：S 盒满足序为 γ 的 GA 是指，1 位输入的变化，至少引起 γ 位输出的变化，作者的结论是：若算法能够提供序为 2 至 5 之间的 GA，则它具有强扩散特征。

对于像 8×32 这样的较大 S 盒，如何找到最好方法来选择 S 盒符合上述准则，是很重要的问题。Nyberg 撰写了许多有关 S 盒设计理论和实现的文章，提出了下列方法：

- **随机性**：使用伪随机数发生器或者随机数表来产生 S 盒的元素。在 S 盒较小时这可能导致一些不太想要的特征，比如 6×4，但是在 S 盒较大时（如 8×32）还是可以接受的。
- **测试随机性**：随机选择 S 盒的元素，按照各种标准进行测试，然后舍弃那些不能通过的元素。
- **人工构造**：利用简单的数学知识，再或多或少用些手工的方法。DES 的 S 盒的设计明显就是这种方法。在 S 盒较大时用这种方法很困难。
- **数学方法**：根据数学原理生成 S 盒，用这种办法生成的 S 盒安全性很高，可以抵抗差分密码分析和线性密码分析，且有很好的扩散效果。

上述第一种方法可以做一些改动，即 S 盒是随机且与密钥相关的。第 6 章中讨论的 Blowfish算法便是如此，它的 S 盒最初是由伪随机数填充的，根据密钥而改变内容。与密钥相关的 S 盒的最大优点是，因为它们不固定，所以预先分析 S 盒以找出其弱点不可行。

3.6.4 密钥扩展算法

分组密码设计的最后一个方面是密钥扩展算法，它没有受到像 S 盒那样的关注。所有的 Feistel分组密码，密钥在每轮迭代都要产生一个子密钥。一般说来，子密钥的选择方案应该确

① 文献[SCHN96]是一篇关于 1996 年前 S 盒研究的较好综述。

保推导出子密钥及密钥的难度很大。还没有这方面的一般原理见诸报道。

Hall指出[ADAM94]密钥扩展算法至少应保证密钥和密文符合严格雪崩效应准则和位独立准则。

3.7　分组密码的工作模式

DES 算法是提供数据安全的基本构件。为了将 DES 应用于实际，人们定义了四种工作模式(FIPS 81)。这四种模式实际上覆盖了所有使用 DES 的应用程序。由于新的应用和要求已经出现，在特别公布 800-38A 中NIST已将推荐模式扩展为 5 个。这些模式可用于包括 3DES 和高级加密标准(AES)在内的任何分组密码。表 3.6 对这些模型做了一个总结，在余下的章节中将会做简单的描述。

表 3.6　DES 的工作模式

模　式	描　述	典 型 应 用
电码本(ECB)	用相同的密钥分别对明文组加密	• 单个数据的安全传输(如一个加密密钥)
密码分组链接(CBC)	加密算法的输入是上一个密文组和下一个明文组的异或	• 普通目的的面向分组的传输 • 认证
密码反馈(CFB)	一次处理 j 位。上一个分组密文作为产生一个伪随机数输出的加密算法的输入，该输出与明文异或，作为下一分组的输入	• 普通目的的面向分组的传输 • 认证
输出反馈(OFB)	与 CFB 基本相同，只是加密算法的输入是上一次 DES 的输出	• 噪声通道上的数据流的传输(如卫星通信)
计数器(CTR)	每个明文组是与加密的计数器的异或。对每个后续的组，计数器是累的	• 普通目的面向分组的传输 • 用于高速需求

3.7.1　电码本模式

最简单的模式是电码本(ECB)模式，它一次处理 64 位明文，[①] 每次使用相同的密钥加密(见图 3.11)。使用电码本这个词是因为对于给定的密钥，任何 64 位的明文组只有惟一的密文与之对应，所以，可以想像存在一个很厚的密码本，根据任意 64 位明文都可以查到相应的密文。

明文若长于 64 位，则可简单地将其分成 64 位分组，必要时可对最后一个分组进行填充。解密也是一次执行一个分组，且使用相同的密钥。图 3.11 中的明文由一串 64 位分组成，记做 $P_1, P_2, \cdots, P_N$，相应的密文分组依次是 $C_1, C_2, \cdots, C_N$。

ECB 模式特别适合于数据较少的情况，比如加密密钥。因此，若想安全地传输一个 DES 密钥，选择这种模式是合适的。

ECB 最重要的特征是一段消息中若有几个相同的明文组，那么密文也将出现几个相同的片断。

对于很长的消息，ECB 模式可能不安全。如果消息是非常结构化的，密码分析者可能利用其结构特征来破译。例如，若已知这段消息总是以某些固定的字符开头，密码分析者就可以拥

① 这一节我们假设明文分组长度为 64 位，DES、3DES和许多商业分组密码都采用这个明文分组长度。对于其他的明文分组长度，如 AES 采用的 128 位分组，这里讨论的工作模式同样正确。

有大量明密文对以展开攻击。若消息有重复的成分，且重复的周期正好是 64 位的倍数，分析者就能辩认出这些成分，然后可以用代换或重排这些分组的方法进行攻击。

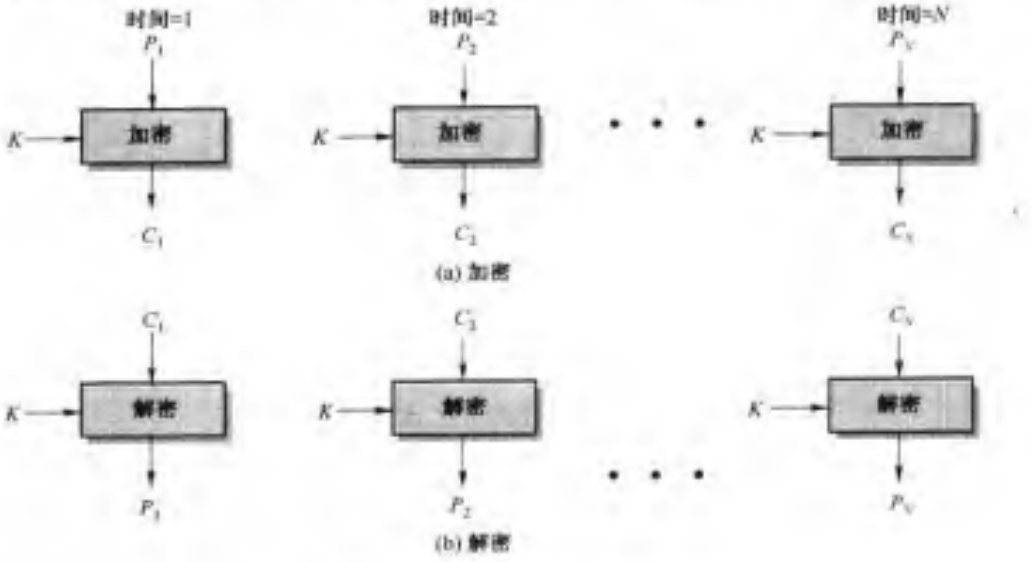


图 3.11 电码本(ECB)模式

3.7.2 密码分组链接模式

为了克服 ECB 的这些弱点，我们需要将重复的明文组加密成不同的密文组。CBC 模式能满足这个要求(见图 3.12)。这种模式下加密算法的输入是当前的明文组和上一个密文组的异或，而使用的密钥是相同的。这就相当于将所有的明文组链接起来了。加密算法的每次输入

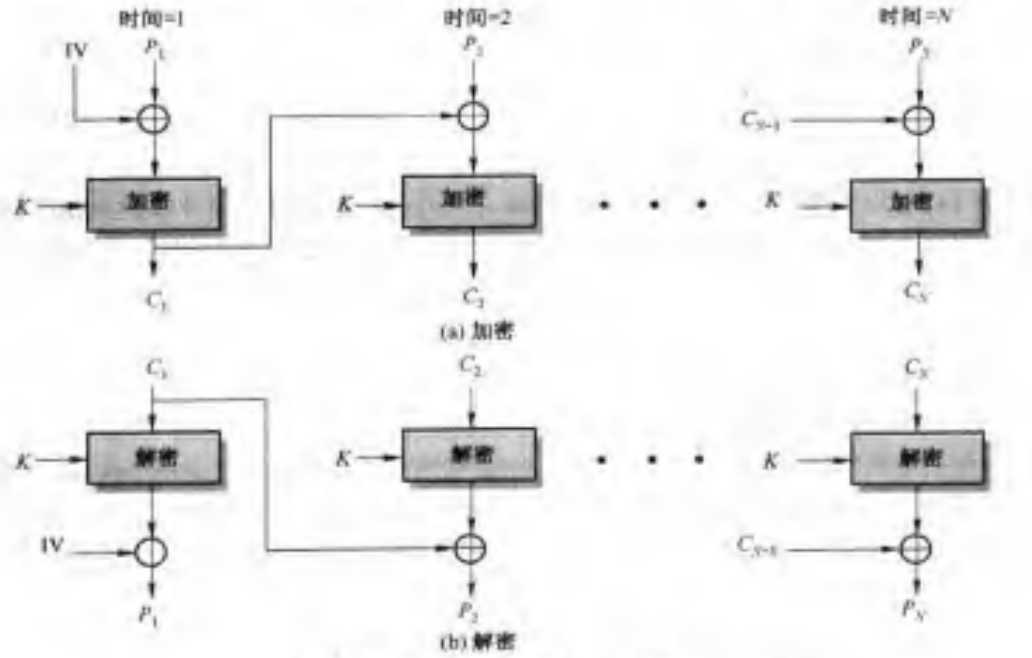


图 3.12 密码分组链接(CBC)模式

与本明文组没有固定的关系。因此，若有重复的 64 位明文组，加密后就看不出来了。

解密时，每个密码组分别进行解密，再与上一块密文异或就可恢复出明文。下面对这个过程的正确性给出证明。

$$C_j = \mathrm{E}_K[C_{j-1} \oplus P_j]$$

则

$$\mathrm{D}_K[C_j] = \mathrm{D}_K[\mathrm{E}_K(\mathrm{C}_{j-1} \oplus \mathrm{P}_j)]$$
$$\mathrm{D}_K[\mathrm{C}_j] = (\mathrm{C}_{j-1} \oplus \mathrm{P}_j)$$
$$\mathrm{C}_{j-1} \oplus \mathrm{D}_K[\mathrm{C}_j] = \mathrm{C}_{j-1} \oplus \mathrm{C}_{j-1} \oplus \mathrm{P}_j = \mathrm{P}_j$$

第一块明文可以和一个初始矢量(IV)异或后再加密。解密时将第一块密文解密的结果与 IV 异或而恢复出第一块明文。

IV 必须为收发双方共享。为了增加安全性，IV 应该和密钥一样加以保护，比如用 ECB 加密来保护 IV。要保护 IV 的一个原因是：攻击者可以欺骗接收者，让他使用不同的 IV，然后将第一个明文组的某些位取反。为了解这一点，考虑如下的等式：

$$C_1 = \mathrm{E}_K(\mathrm{IV} \oplus P_1)$$
$$P_1 = \mathrm{IV} \oplus \mathrm{D}_K(C_1)$$

用 X[i]表示 64 位 X 的第 i 位，则有：

$$P_1[i] = \mathrm{IV}[i] \oplus \mathrm{D}_K(C_1)[i]$$

使用 XOR 的性质，我们将上式重写为：

$$P_1[i]' = \mathrm{IV}[i]' \oplus \mathrm{D}_K(C_1)[i]$$

撇号表示取反。这意味着攻击者可以预先改变 IV 中的某些位，从而接收者收到的 P_1 相应也就改变了。

还可能有其他利用 IV 的一些攻击方法，见文献[VOYD83]。

总之，CBC 的链接机制使得它适合于加密长度大于 64 位的消息。

CBC 除了用来获得保密性，亦可用于认证。这种使用将在第二部分描述。

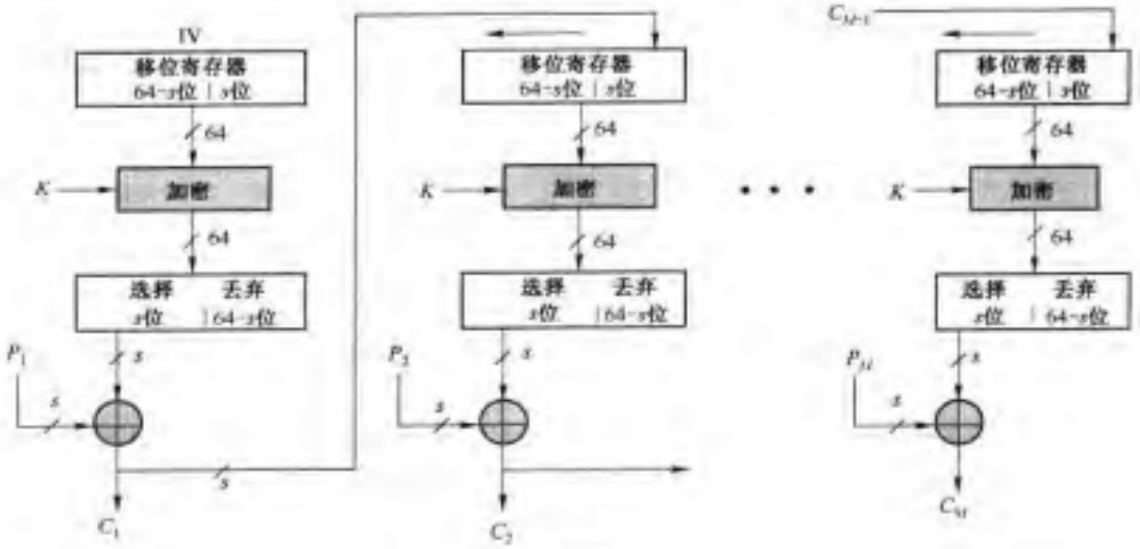
3.7.3　密码反馈模式

DES 本质上是一个 64 位的分组密码，但是利用密码反馈模式(CFB)或输出反馈模式(OFB)，亦可作为流密码使用。流密码不需要明文长度是分组长度的整数倍，且可以实时操作。所以，待发送的字符流中任何一个字符都可以用面向字符的流密码加密后立即发送。

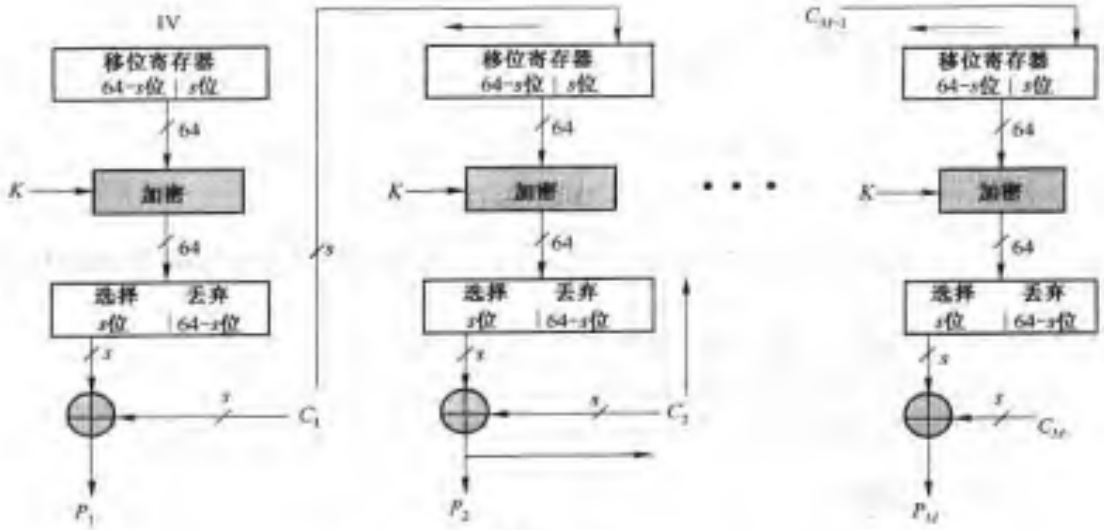
流密码有一个让人心动的性质，即密文与明文等长。所以如果要发送的是 8 位的字符，加密时也是用 8 位。如果多于 8 位，传输能力就浪费了。

图 3.13 描述了 CFB 模式。假设传输单元是 s 位，s 通常为 8。如果用 CBC 模式，明文的各个单元要链接起来，所以任意个明文单元的密文都是前面所有明文的函数。在这种情况下，明文被分成 s 位的片段而不是 64 位的单元。

首先来考虑加密。加密函数的输入是 64 位的移位寄存器，它的值为初始矢量 IV。加密函数输出最左边的 s 位与明文 P_1 异或得到第一个密文单元 C_1，然后将 C_1 发送出去；接着，移位寄存器左移 s 位，C_1 填入移位寄存器的最右边 s 位。就这样，直到所有明文单元被加密完成。



(a) 加密



(b) 解密

图 3.13 s 位密码反馈(CFB)链接

解密使用相同的办法，只是有一点不同：将收到的密文单元与加密函数的输出异或得到明文单元。注意，这里使用的是加密函数而非解密函数，这一点很容易解释。设 $S_s(X)$ 表示 X 的最左边 s 位，则有：

$$C_1 = P_1 \oplus S_s(E_K(IV))$$

从而有：

$$P_1 = C_1 \oplus S_s(E_K(IV))$$

对后续单元亦同理可得。

3.7.4　输出反馈模式

如图 3.14 所示，输出反馈模式的结构和 CFB 很相似。正如我们所见的那样，它用加密函数的输出填充 OFB 中的移位寄存器，而在 CFB 中，则用密文单元来填充移位寄存器。

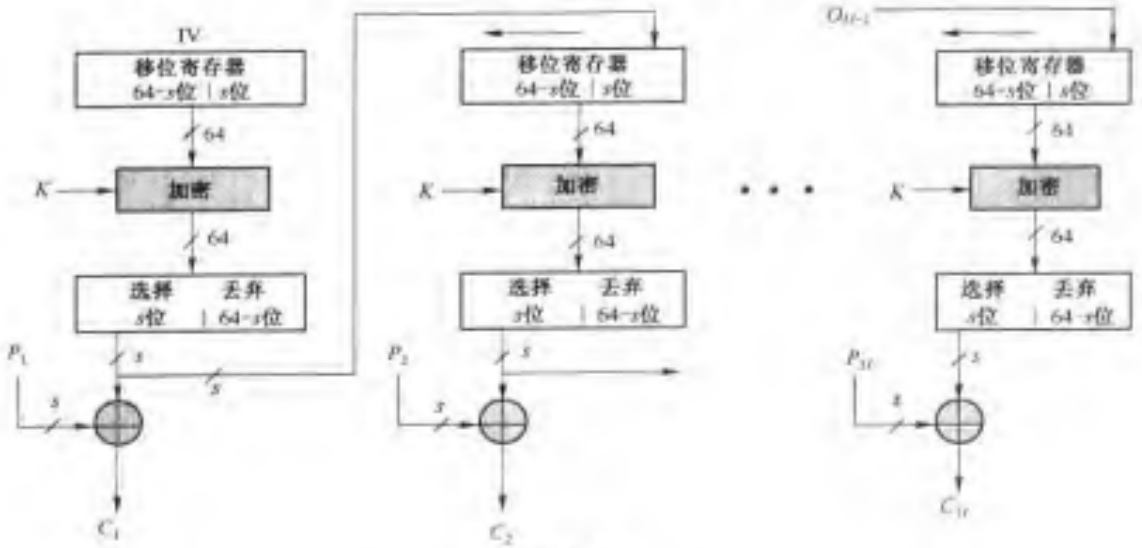


图 3.14　s 位输出反馈链接

OFB 的一个优点是传输过程中在某位上发生的错误不会影响其他位。比如，C_1 中有 1 位发生了错误，只会影响到 P_1 的恢复，后续的明文单元不受影响。而 CFB 中 C_1 还作为了移位寄存器的输入，所以影响了后续的所有消息。

OFB 的缺点是，抗消息流篡改攻击的能力不如 CFB。即密文中的某位取反，恢复出的明文

相应位也取反。所以攻击者有办法控制对恢复明文的改变。这样，攻击者可以根据消息的改动而改动校验和，以使改动不被纠错码发现。读者若想深入了解，请参阅文献[VOYD83]。

3.7.5 计数器模式

尽管随着计数器模式(CTR)在 ATM 网络安全与 IPSec 中的应用使得人们最近才对它产生了浓厚的兴趣，但这种模式在很早以前就已经提出来了[DIFF79]。

图 3.15 描述了 CTR 模式。计数器使用与明文分组规模相同的长度。SP 800-38A 的惟一要求是加密不同的明文组计数器对应的值必须是不同的(模 2^b，其中 b 为分组大小)。典型地，计数器首先被初始化为某一值，然后随着消息块的增加计数器的值加 1。计数器加 1 后与明文组异或得到密文组。解密使用具有相同值的计数器序列，用加密后的计数器的值与密文组异或来恢复明文组。

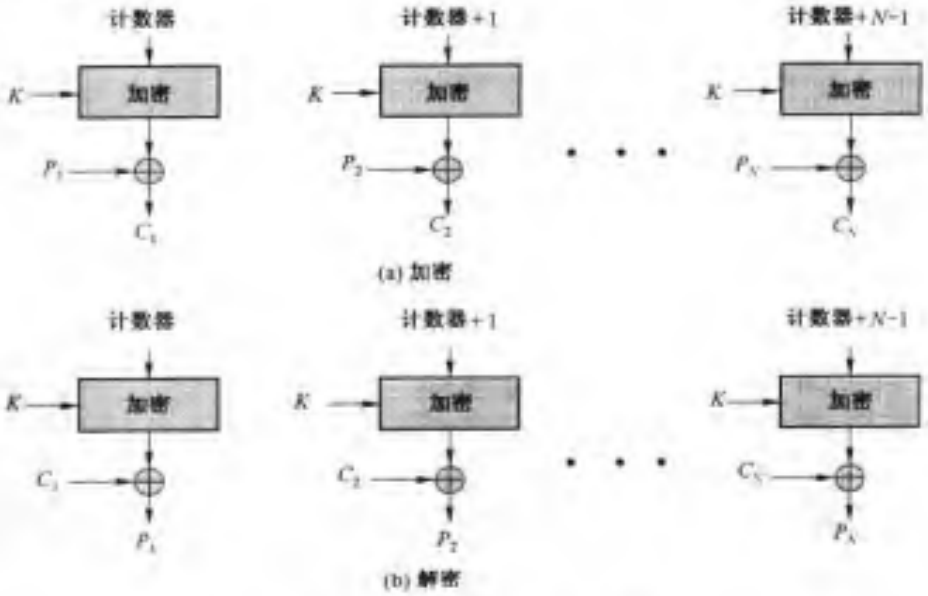


图 3.15 计数器(CTR)模式

文献[LIPM00]列出了计数器模式的如下优点：

- **硬件效率**：与三种链接模式不同，CTR 模式能够并行处理多块明文(密文)的加密(解密)。链接模式在处理下一块数据之前必须完成当前数据块的计算，这就限制了算法的吞吐量。在 CTR 模式中，吞吐量仅受可使用并行数量的限制。
- **软件效率**：类似地，因为 CTR 模式能够进行并行计算，处理器能够很好地用来提供像流水线、每个时钟周期的多指令分派、大数量的寄存器和 SIMD 指令等并行特征。
- **预处理**：基本加密算法的执行并不依靠明文或密文的输入。因此，如果有充足的存储器可用且能提供安全，预处理能够用来准备如图 3.15 所示的用于 XOR 函数的加密盒的输出。当给出明文或者密文时，所需的计算仅是进行一系列的异或。这样的策略能够极大地提高吞吐量。
- **随机访问**：密文的第 i 个明文组能够用一种随机访问的方式处理。在链接模式下，前面

的 $i-1$ 块密文计算出来后才能计算密文 C_i。有很多应用情况是全部密文已存储好了，只需要破解其中的某一块密文。对于这种情形，随机访问的方式很有吸引力。

- **可证明安全性**：能够证明 CTR 模式至少和本节讨论的其他模式一样安全。
- **简单性**：与 ECB 和 CBC 不同，CTR 模式要求实现加密算法，但不要求实现解密算法，像高级加密标准一样，当加密算法与解密算法本质上不同时，就更能体现这种模式的简单性。另外，也不用实现解密密钥扩展。

3.8　推荐读物

有关对称加密算法的文献浩如烟海，这里列出其中较有价值的一部分。基本参考书有[SCHN96]，这部巨著描述了它出版之前几乎所有发表的密码算法和协议。作者搜集了期刊、会议论文集、政府出版物和标准文件等，并做了综合整理。另外一本有价值且较详细的文献是[MENE97]。文献[STIN02]进行了严格的数学处理。

前面的参考文献也覆盖了公钥密码学。

或许对 DES 描述最详细的文献是[SIMO95]；这本书也广泛讨论了 DES 的差分密码分析和线性密码分析。文献[BARK91]给出了关于 DES 的有趣密码学分析，也给出了它的潜在密码学分析方法。[EFF98]详细介绍了DES最有效的穷举攻击。文献[COPP94]考察了 DES 的内在强度和抗密码学分析的能力。

BARK91　Barker, W. *Introduction to the Analysis of the Data Encryption Standard (DES)*. Laguna Hills, CA: Aegean Park Press, 1991.

COPP94　Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." *IBM Journal of Research and Development*, May 1994.

EFF98　Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA: O'Reilly, 1998.

MENE97　Menezes, A.; Oorshcot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

SCHN96　Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SIMO95　Simovits, M. *The DES: An Extensive Documentation and Evaluation*. Laguna Hills, CA: Aegean Park Press, 1995.

STIN02　Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

3.9　关键术语、思考题和习题

3.9.1　关键术语

雪崩效应	差分分析	置换
分组密码	扩散	乘积密码

密码分组链接模式	电码本模式	可逆映射
密码反馈模式	Feistel 密码	轮
混淆	不可逆映射	轮函数
计数器模式	密钥	流密码
数据加密标准	线性密码分析	子密钥
输出反馈模式	代换	

3.9.2 思考题

3.1 为什么说研究 Feistel 密码很重要?

3.2 分组密码和流密码的差别是什么?

3.3 为什么使用表 3.1 所示的任意可逆代换密码不实际?

3.4 什么是乘积密码?

3.5 混淆与扩散的差别是什么?

3.6 哪些参数与设计选择决定了 Feister 密码的实际算法?

3.7 设计 DES S 盒的目的是什么?

3.8 解释什么是雪崩效应。

3.9 差分密码分析与线性密码分析的区别是什么?

3.10 为什么某些分组密码的操作模式仅使用加密算法,而其他的模式既使用加密算法又使用解密算法?

3.9.3 习题

3.1 参考有关描述 S-DES 密钥产生的图 3.2,回答:

a. 初始 P10 置换函数有多重要?

b. 两个 LS-1 左移函数有什么重要作用?

3.2 在对 S-DES 的分析一节中,定义了一个关于变量 q 和 r 的等式。请给出 s 和 t 的等式。

3.3 使用 S-DES,用密钥(0111111101)手工解密二进制串(10100010)。给出执行 IP、F_K、SW、F_K、IP^{-1}后的中间值。再把前 4 位译码为某个字母,后 4 位译码为另一个字母(例如,将字母 A,B,…,P 分别对应为 0000,0001,…,1111)。提示:执行 SW 后,串的形式为(00010011)。

3.4 考虑由表 3.3 中 S 盒 S_i的第一行定义的代换,并给出对应这个代换、类似于图 3.4 的分块图。

3.5 设 π 表示整数 0,1,2,…,(2^n-1)的一个置换。$\pi(m)$表示 m 的置换值,其中$0 \leqslant m \leqslant 2^n$。换句话说,$\pi$ 将 n 位的整数集映射到其自身,且没有两个整数映射到同一整数。DES 就是 64 位整数的一个置换。若存在 m 满足 $\pi(m)=m$,我们称 π 有一个不动点。即,如果 π 是一个加密映射,则不动点意味着一段消息加密后仍为它自己。我们感兴趣的是 π 没有不动点的可能性有多大。证明多于 60% 的映射将至少有一个不动点。

3.6 考虑分组长度为 n 的分组加密算法,设 $N=2^n$。比如说我们有 t 个明密文对 P_i,$C_i=E_K(P_i)$,这里我们假设密钥 K 选择了 $N!$ 种映射中的一种。假设我们希望用穷举的

方法找出 K。我们可以产生密钥 K'，并对这 t 个明密文对测试 $C_i = E_{K'}(P_i)$ $(1 \leqslant i \leqslant t)$是否成立。若 K'将每个 P_i 加密成其正确的 C_i，我们就有理由说 $K = K'$。然而，可能 K 与 K'不同，但恰好 $E_K(\cdot)$和 $E_{K'}(\cdot)$对这 t 个明密文对 P_i，C_i 的结果一样。

a. $E_K(\cdot)$和 $E_{K'}(\cdot)$事实上表示不同映射的概率有多大?

b. $E_K(\cdot)$和 $E_{K'}(\cdot)$对另外 $t'(0 \leqslant t' \leqslant N - t)$个明密文对正好都成立的概率有多大?

3.7 这个问题给出了用一轮的 DES 加密的具体数字的例子。我们假设明文和密钥有相同的位模式，即：

用十六进制表示：0 1 2 3 4 5 6 7 8 9 A B C D E F

用二进制表示：　0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 0100 1101 1110 1111

a. 推导第一轮的子密钥 K_1。

b. 推导 L_0，R_0。

c. 扩展 R_0 求 $E[R_0]$。

d. 计算 $A = E[R_0] \oplus K_1$。

e. 把(d)的 48 位结果分成 6 位(数据)的集合并求对应 S 盒代换的值。

f. 利用(e)的结论来求 32 位的结果，B。

g. 应用置换求 $P(B)$。

h. 计算 $R_1 = P(B) \oplus L_0$。

i. 写出密文。

3.8 验证下面来自 3.2 节的陈述：一般来说，对于 n 维代换分组密码，密钥大小为 $n \times 2^n$。

3.9 证明 DES 解密算法的确是 DES 加密算法的逆。

3.10 DES 算法第十六轮之后的 32 位互换，使得 DES 的解密过程与加密过程一样，只是密钥的使用不同。习题 3.6 可说明这一点。然而，为什么需要这 32 位的互换，你可能并不非常清楚，所以看看下面的练习。首先给出一些记号：

$A \| B$　　= 将串 A 和串 B 连接起来

$T_i(R \| L)$　= 加密过程第 i 轮迭代所定义的变换$(1 \leqslant i \leqslant 16)$

$TD_i(R \| L)$　= 解密过程第 i 轮迭代所定义的变换$(1 \leqslant i \leqslant 16)$

$T_{17}(R \| L)$　= $L \| R$。加密过程第 16 轮迭代之后的变换

a. 证明下式：

$$TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15} \| R_{15}))))) = R_{15} \| L_{15}$$

b. 请判断下式是否成立：

$$TD_1(IP(IP^{-1}(T_{16}(L_{15} \| R_{15})))) = L_{15} \| R_{15}$$

3.11 比较初始置换表[表 3.2(a)]和置换选择 1 表[表 3.4(b)]，它们的结构是否相似? 如果相似，请说明它们的相似之处。通过分析你可以得出什么结论?

3.12 16 个密钥$(K_1, K_2, \cdots, K_{16})$在 DES 解密过程中是逆序使用的。因此，图 3.8 的右半部分不再正确。请模仿表 3.4(d)设计一个合适移动的密钥扩展方案。

3.13 **a**. 设 M' 是对 M 按位取反的结果。证明如果明文和密钥都取反，则密文取反。即：

如果 $Y = \mathrm{DES}_K(X)$

那么 $Y' = \mathrm{DES}_{K'}(X')$

提示：首先证明对任意两个相同长度的串 A 和 B，有：

$$(A \oplus B)' = A' \oplus B$$

b. 据说对 DES 的穷举攻击需要搜索 2^{56} 个密钥的密钥空间。(a)中的结论对此是否有影响？

3.14 证明 DES 中每个子密钥的前 24 位均来自于初始密钥的 28 位，而后 24 位来自于初始密钥的另外 28 位。

3.15 在 DES 的 ECB 模式中，若在密文的传输过程中，某一块发生了错误，则只有相应的明文组会受影响。然而，在 CBC 模式中，这种错误具有扩散性。比如，图 3.12 中传输 C_1 时发生的错误将会影响明文组 P_1 和 P_2。

a. P_2 以后的所有块是否会受到影响？

b. 假设 P_1 本来就有一位发生了错误。这个错误要扩散至多少个密文组？对接收者解密后的结果有什么影响？

3.16 在 8 位的 CFB 模式中，若传输中一个密文字符发生了一位错误，这个错误将传播多远？

3.17 填满下表中的剩余位置。

操作模式	加密	解密
电码本模式	$C_j = \mathrm{E}_K[P_j]\ j = 1, \cdots, N$	$P_j = \mathrm{D}_K[C_j]\ j = 1, \cdots, N$
密码分组链接模式	$C_1 = \mathrm{E}_K[P_1 \oplus \mathrm{IV}]$ $C_j = \mathrm{E}_K[P_j \oplus C_{j-1}]\ j = 2, \cdots, N$	$P_1 = \mathrm{E}_K[C_1] \oplus \mathrm{IV}$ $P_j = \mathrm{E}_K[C_j] \oplus C_{j-1}\ j = 2, \cdots, N$
密码反馈模式		
输出反馈模式		
计数器模式		

第4章 有限域

有限域在密码编码学中的地位越来越重要。许多密码算法都对有限域的性质有很大的依赖性，特别是高级加密标准(AES)和椭圆曲线加密算法。本章先对群、环和域的概念做一个简要介绍。然后，我们介绍模运算和 Euclid 算法的基本背景知识。我们将讨论形如 GF(p)的有限域，其中 p 为素数。随后我们了解有关多项式运算的知识。本章最后讨论形如 GF(2^n)的有限域，其中 n 是一个正整数。

数论中的概念和方法都非常抽象，如果没有具体的实例，想要直观地掌握它们是比较困难的[RUBI97]。因此，在本章和第 8 章包含了大量的例子，而且每个例子都以加框线的形式得到了突出。

4.1 群、环和域

群、环和域都是数学理论中一个分支，即抽象代数或称为近世代数的基本元素。在抽象代数中，我们关心的是其元素能进行代数运算的集合，也就是说，我们可以通过很多种方法，使集合上的两个元素组合得到集合中的第三个元素。这些运算方法都遵守特殊的规则，而这些规则又能确定集合的性质。根据约定，集合上元素的两种主要运算符号与普通数字的加法和乘法所使用的符号是相同的。然而，我们必须指出，在抽象代数中，我们并不仅仅限于普通的算术操作。在后面的章节中，这将变得很清楚。

4.1.1 群

群 G，有时记做$\{G,\cdot\}$，是定义了一个二元运算的集合，这个二元运算可表示为$\cdot$，G 中每一个序偶(a,b)通过运算生成 G 中的元素$(a\cdot b)$，并满足以下公理[①]：

(**A1**) 封闭性：如果 a 和 b 都属于 G，则 $a\cdot b$ 也属于 G。

(**A2**) 结合律：对于 G 中任意元素 a、b、c，$a\cdot(b\cdot c)=(a\cdot b)\cdot c$ 都成立。

(**A3**) 单位元：G 中存在一个元素 e，对于 G 中任意元素 a，都有 $a\cdot e=e\cdot a=a$ 成立。

(**A4**) 逆元：对于 G 中任意元素 a，G 中都存在一个元素 a'，使得式 $a\cdot a'=a'\cdot a=e$ 成立。

我们用 N_n 表示 n 个不同符号的集合，为方便起见，将其表示成$\{1,2,\cdots,n\}$。n 个不同符号的一个置换是一个 N_n 到 N_n 的一一映射。定义 S_n 为 n 个不同符号的所有置换组成的集合。S_n 中的每一个元素都代表集合$\{1,2,\cdots,n\}$的一个置换。可以很容易地验证 S_n 是一个群：

A1：如果 $\pi,\rho\in S_n$，则合成映射 $\pi\cdot\rho$ 根据置换 π 来改变 ρ 中元素的次序而形成。例如，$\{3,2,1\}\cdot\{1,3,2\}=\{2,3,1\}$。显然，$\pi\cdot\rho\in S_n$。

① 操作符$\cdot$具有一般性：可以指加法、乘法或某些其他的数学运算。

A2:映射的合成显而易见满足结合律。

A3:恒等映射就是不改变 n 个元素位置的置换。对于 S_n,单位元是{1,2,…,n}。

A4:对于任意 $\pi \in S_n$,抵消由 π 定义置换的映射就是 π 的逆元。这个逆元总是存在的。例如,{2,3,1}·{3,1,2} = {1,2,3}。

如果一个群的元素是有限的,则该群称为**有限群**。并且群的阶等于群中元素的个数。否则,称该群为**无限群**。

一个群如果还满足以下的条件,则称为**交换群**:

(**A5**) 交换律:对于 G 中任意的元素 a、b,都有 $a \cdot b = b \cdot a$ 成立。

在加法运算下的整数集合(包括正整数、负整数和零)是一个交换群。在乘法运算下的非零实数集合是一个交换群。前面例子中的集合 S_n 是一个群,但对于 $n > 2$,它不是交换群。

当群中的运算符是加法时,其单位元是 0;a 的逆元是 $-a$,并且减法用以下的规则定义:$a - b = a + (-b)$。

循环群

我们在群中定义求幂运算为重复运用群中的运算,如 $a^3 = a \cdot a \cdot a$。而且,我们定义 $a^0 = e$ 作为单位元,并且 $a^{-n} = (a')^n$。如果群中的每一个元素都是一个固定元素 a($a \in G$)的幂 a^k(k 为整数),则称群 G 是**循环群**。我们认为元素 a **生成了**群 G,或者说 a 是群 G 的**生成元**。循环群总是交换群,它可能是有限群或无限群。

整数的加法群是一个无限循环群,它由 1 生成。在这种情况下,幂被解释为用加法合成的,因此 n 是 1 的 n 次幂。

4.1.2 环

环 R,有时记为{R, +, ×},是一个有两个二元运算的集合,这两个二元运算分别称为加法和乘法[①],且对于 R 中的任意元素 a、b、c,满足以下公理:

(**A1 – A5**) R 关于加法是一个交换群;也就是说,R 满足从 A1 到 A5 的所有原则。对于此种情况下的加法群,我们用 0 表示其单位元,$-a$ 表示 a 的逆元。

(**M1**) 乘法的封闭性:如果 a 和 b 都属于 R,则 ab 也属于 R。

(**M2**) 乘法的结合律:对于 R 中的任意元素 a、b、c,有 $a(bc) = (ab)c$ 成立。

(**M3**) 分配律:对于 R 中的任意元素 a、b、c,式 $a(b + c) = ab + ac$ 和式 $(a + b)c = ac + bc$ 总成立。

本质上说,环就是一个集合,我们可以在其上进行加法、减法[a – b = a + (– b)]和乘法,而不脱离该集合。

① 一般地,我们并不用 × 表示乘法,而是用两个元素的连接表示。

实数上所有 n 阶方阵的集合关于加法和乘法够成一个环 R。

环如果还满足以下条件，则称为**交换环**：

(**M4**)乘法的交换律：对于 R 中的任意元素 a、b，有 $ab = ba$ 成立。

偶整数集合(包括正数、负数和0)记为 S，在普通加法和乘法运算下 S 是交换环。前面一个例子中定义的所有 n 阶方阵的集合则不是交换环。

下面，我们定义**整环**，它是满足以下公理的交换环：

(**M5**) 乘法单位元：在 R 中存在元素 1，使得对于 R 中的任意元素 a，有 $a1 = 1a = a$ 成立。

(**M6**) 无零因子：如果有 R 中元素 a、b，且 $ab = 0$，则必有 $a = 0$ 或 $b = 0$。

将普通加法和乘法运算下的整数集合(包括正数、负数和0)记为 S，则 S 是一个整环。

4.1.3 域

域 F，有时记为 $\{F, +, \times\}$，是有两个二元运算的集合，这两个二元运算分别称为加法和乘法，且对于 F 中的任意元素 a、b、c，满足以下公理：

(**A1 – M6**) F 是一个整环；也就是说 F 满足从 A1 到 A5 以及从 M1 到 M6 的所有原则。

(**M7**) 乘法逆元：对于 F 中的任意元素 a(除 0 以外)，F 中都存在一个元素 a^{-1}，使得式 $a\,a^{-1} = (a^{-1})a = 1$ 成立。

本质上说，域就是一个集合，我们可以在其上进行加法、减法、乘法和除法而不脱离该集合。除法又按以下规则来定义：$a/b = a(b^{-1})$。

有理数集合、实数集合以及复数集合都是我们所熟悉的域的例子。需要指出的是，所有整数的集合并不是一个域，因为并不是集合中所有的元素都有乘法逆元；实际上，整数集合中只有元素 1 和 – 1 有乘法逆元。

图 4.1 总结了定义群、环和域的公理。

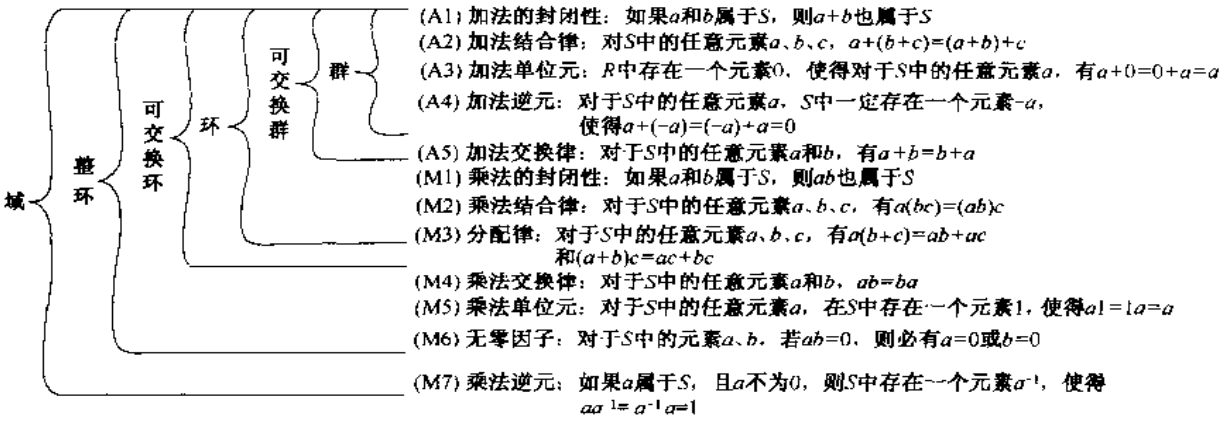


图 4.1 群、环和域

4.2 模运算

给定任意一个正整数 n 和任意整数 a，如果将用 n 除 a，则得到整数商 q 和整数余数 r，且它们之间满足以下关系：

$$a = qn + r \qquad 0 \leqslant r < n; q = \lfloor a/n \rfloor$$

其中$\lfloor x \rfloor$是小于或等于 x 的最大整数。

图 4.2 表明，对于给定的 a 和正整数 n，总可以找到满足上面关系式的 q 和 r。在数轴上描绘出整数，a 就会落在轴上的某个位置（a 为正整数的情况已经表示出来，a 为负数的情况可以类似地表示）。从 0 开始，然后到 n，$2n$，一直到 qn，使得 $qn \leqslant a$ 且 $(q+1)n > a$。从 qn 到 a 的距离即为 r，我们可以找到惟一的 q 和 r 的值。余数 r 称为**剩余**。

$a = 11;$	$n = 7;$	$11 = 1 \times 7 + 4;$	$r = 4$
$a = -11;$	$n = 7;$	$-11 = (-2) \times 7 + 3;$	$r = 3$

图 4.2 关系 $a = qn + r, 0 \leqslant r < n$

如果 a 是整数，n 是正整数，则我们定义 a 除以 n 所得的余数为 a 模 n。因此，对于任意整数 a，我们总可以写出：

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$11 \bmod 7 = 4;$ $\qquad -11 \bmod 7 = 3$

如果$(a \bmod n) = (b \bmod n)$，则我们称整数 a 和 b 是。可以表示为 $a \equiv b \bmod n$。

$73 \equiv 4 \bmod 23;$ $\qquad 21 \equiv -9 \bmod 10$

4.2.1 因子

设 a、b 和 m 均为整数，如果存在某个 m 使得 $a = mb$，则我们称 b 整除 a。也就是说，如果做除法时余数为零，我们则认为 b 整除 a。符号 $b \mid a$ 通常用来表示 b 整除 a。同样，如果 $b \mid a$，我们就说 b 是 a 的一个因子。

24 的正因子有 1,2,3,4,6,8,12 和 24。

以下关系成立：

- 如果$a|1$,那么 $a=\pm 1$。
- 如果$a|b$,且$b|a$,那么 $a=\pm b$。
- 任何 $b\neq 0$ 整除 0。
- 如果$b|g$ 且$b\nmid h$,那么对任意整数 m 和n,都有$b|(mg+nh)$。

我们来证明最后一个关系式,注意到:

如果$b|g$, 那么 $g=b\times g_1$, g_1 为整数。
如果$b|h$, 那么 $h=b\times h_1$, h_1 为整数。
于是有 $mg+nh=mbg_1+nbh_1=b\times(mg_1+nh_1)$, 因此 b 整除 $mg+nh$。

$b=7; g=14; h=63; m=3; n=2$
$7|14$ 和$7|63$。证明:$7|(3\times 14+2\times 63)$
我们知道$(3\times 14+2\times 63)=7(3\times 2+2\times 9)$,显然,$7|(7(3\times 2+2\times 9))$

注意,如果 $a\equiv 0 \bmod n$,那么$n|a$。

4.2.2 模运算的性质

模运算有如下性质:

1. 如果$n|(a-b)$, 那么 $a\equiv b \bmod n$。
2. $a\equiv b \bmod n$ 隐含$b\equiv a \bmod n$。
3. $a\equiv b \bmod n$ 和$b\equiv c \bmod n$ 隐含$a\equiv c \bmod n$。

先证明第一条。如果$n|(a-b)$, 那么存在某个 k 使得$(a-b)=kn$。于是我们知道 $a=b+kn$。因此,$(a \bmod n)=(b+kn$ 除以 n 的余数$)=(b$ 除以 n 的余数$)=(b \bmod n)$。

$23\equiv 8(\bmod 5)$	因为 $23-8=15=5\times 3$
$-11\equiv 5(\bmod 8)$	因为 $-11-5=-16=8\times(-2)$
$81\equiv 0(\bmod 27)$	因为 $81-0=81=27\times 3$

剩下的两条同样可很容易地证明。

4.2.3 模算术运算

注意,由定义知(图 4.2),(mod n)运算符将所有的整数映射到$\{0,1,\cdots,(n-1)\}$组成的集合。于是出现了这样的一个问题:能否限制在这个集合上进行算术运算? 可以,这种技术称为**模算术**。

模算术有如下性质:

1. $[(a \bmod n)+(b \bmod n)] \bmod n=(a+b) \bmod n$
2. $[(a \bmod n)-(b \bmod n)] \bmod n=(a-b) \bmod n$
3. $[(a \bmod n)\times(b \bmod n)] \bmod n=(a\times b) \bmod n$

我们来证明第一个性质。定义$(a \bmod n)=r_a$, $(b \bmod n)=r_b$, 于是存在整数 j 和k,使得 $a=r_a+jn$, $b=r_b+kn$。那么有:

$$\begin{aligned}(a+b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k+j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

剩下的性质容易证明。下面是关于这三个性质的一些例子。

$11 \bmod 8 = 3; 15 \bmod 8 = 7$

$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$

$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$

$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$

$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

幂运算和普通算术一样，可以通过重复乘法实现。

求 $11^7 \bmod 13$，我们可以如下进行：

$11^2 = 121 \equiv 4 \bmod 13$

$11^4 \equiv 4^2 \equiv 3 \bmod 13$

$11^7 \equiv 11 \times 4 \times 3 \equiv 132 \equiv 2 \bmod 13$

因此，普通算术的加、减、乘运算规则可以平移到模算术中。

表 4.1 给出了模 8 的加法和乘法的例子。请看模 8 的加法例子，结果非常简单并且矩阵有一定的规律。就像在普通的加法运算中一样，模运算中对于每个整数也存在加法逆元素，或称为负数。在模运算中，整数 x 的负数 y 是使得 $x + y \equiv 0 \bmod 8$ 成立的值。可以这样求出左边列中整数的加法逆元素：浏览矩阵的对应行，并找出值 0；0 所在列顶部的整数就是所求的加法逆元素；例如 $2 + 6 = 0 \bmod 8$。同样地，在乘法表中的元素的意义明确。在普通运算中，每个整数都有其乘法逆元素，或称为倒数。在模 8 乘法运算中，整数 x 的乘法逆元 y 是使得 $x \times y \equiv 1 \bmod 8$ 成立的值。下面我们在乘法表中求一个整数的乘法逆元：浏览矩阵中整数所在的行并寻找值 1，1 所在列顶部的整数就是所求的乘法逆元素；例如 $3 \times 3 = 1 \bmod 8$。需要指出的是，并不是所有的整数模 8 都有乘法逆元；后面我们将会经常提到。

4.2.4 模算术的性质

定义比 n 小的非负整数集合为 Z_n：

$$Z_n = \{0, 1, \cdots, (n-1)\}$$

这个集合称为**剩余集**，或模 n 的**剩余类**。更准确地说，Z_n 中每一个整数都代表一个剩余类，我们可以将模 n 的剩余类表示为 $[0], [1], [2], \cdots, [n-1]$，其中：

$$[r] = \{a: a \text{ 是一个整数}, a \equiv r \bmod n\}$$

模4剩余类为：

$[0] = \{\cdots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \cdots\}$

$[1] = \{\cdots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \cdots\}$

$[2] = \{\cdots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \cdots\}$

$[3] = \{\cdots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \cdots\}$

表4.1 模8运算

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a)模8加法

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b)模8乘法

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c)模8加法和乘法逆元素

在剩余类所有的整数中，我们通常用最小非负整数来代表这个剩余类。寻找与 k 是模 n 同余的最小非负整数的过程，称为**模 n 的 k 约化**。

如果我们在 Z_n 中进行模运算，表4.2中所列的性质对于 Z_n 中的整数同样适用。因此，Z_n 是有乘法单位元的交换环(图4.1)。

模运算有一个特性使它区分于普通运算。首先，在普通运算中，我们可以采用下式：

$$\text{如果}(a+b)\equiv(a+c)\bmod n \quad \text{那么} \quad b\equiv c \bmod n \tag{4.1}$$

$(5+23)\equiv(5+7) \bmod 8;$ $\qquad 23\equiv 7 \bmod 8$

表 4.2 Z_n 中整数模运算的性质

性质	表达式
交换律	$(w+x) \bmod n = (x+w) \bmod n$ $(w\times x) \bmod n = (x\times w) \bmod n$
结合律	$[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$ $[(w\times x)\times y] \bmod n = [w\times(x\times y)] \bmod n$
分配律	$[w\times(x+y)] \bmod n = [(w\times x)+(w\times y)] \bmod n$ $[w+(x\times y)] \bmod n = [(w+x)+(w+y)] \bmod n$
恒等式	$(0+w) \bmod n = w \bmod n$ $(1\times w) \bmod n = w \bmod n$
加法逆元$(-w)$	对于 Z_n 中的任意 w，存在一个 z 使得 $w+z\equiv 0 \bmod n$

等式 4.1 与加法逆元的存在性是一致的。将 a 的加法逆元加在等式 4.1 的两边，我们得到：

$$((-a)+a+b)\equiv((-a)+a+c) \bmod n$$

$$b\equiv c \bmod n$$

然而，下面的陈述只在有附加条件时才是成立的：

$$\text{若 } a \text{ 与 } n \text{ 是互素的：如果}(a\times b)\equiv(a\times c) \bmod n, \text{那么 } b\equiv c \bmod n \tag{4.2}$$

其中互素的概念这样定义：如果两个整数的最大公因子是 1，则称它们是互素的。同等式 4.1 的情况类似，我们也可以说等式 4.2 同乘法逆元的存在性是一致的。将 a 的乘法逆元素施加在等式 4.1 的两边，我们得到：

$$((a^{-1})ab)\equiv((a^{-1})ac) \bmod n$$

$$b\equiv c \bmod n$$

要明白这点，我们来看一个使等式 4.2 的条件不成立的例子。6 和 8 不是互素的，因为它们有公因子 2。我们有以下式子：

$$6\times 3 = 18\equiv 2 \bmod 8$$

$$6\times 7 = 42\equiv 2 \bmod 8$$

然而，$3\not\equiv 7 \bmod 8$。

得到这个奇怪结果的原因是：对于任何一般的模数 n，如果 a 和 n 有任何公因子的话，用乘数 a 依次作用于 0 到 $n-1$ 的所有整数将不会产生 a 的一个完整剩余集。

如果 $a=6, n=8$,则有:

Z_8	0	1	2	3	4	5	6	7
乘以 6	0	6	12	18	24	30	36	42
余数	0	6	4	2	0	6	4	2

因为乘以 6 时,我们得不到一个完整的剩余集,Z_8 中多于一个整数映射到相同的剩余集。特别地,$6\times 0 \bmod 8 = 6\times 4 \bmod 8$;$6\times 1 \bmod 8 = 6\times 5 \bmod 8$;依次类推。因为这是一个多对一的映射,所以乘法运算并没有惟一的逆运算。

然而,如果我们令 $a=5, n=8$,二者惟一的公因子是 1:

Z_8	0	1	2	3	4	5	6	7
乘以 5	0	5	10	15	20	25	30	35
余数	0	5	2	7	4	1	6	3

余数这一行包含了 Z_8 行的所有整数,只是顺序不同。

一般来说,如果一个整数与 n 互素,那么它在 Z_n 中有一个乘法逆元。从表 4.1(c)可以看出,整数 1、3、5、7 在 Z_8 中有一个乘法逆元,而 2、4 和 6 则没有。

4.3 Euclid 算法

数论的一个最基本的技巧是 Euclid 算法,它是求两个正整数的最大公因子的简单过程。

4.3.1 最大公因子

我们用 $\gcd(a,b)$ 表示 a 和 b 的最大公因子。正整数 c 称为 a 和 b 的最大公因子,如果:

1. c 是 a 和 b 的因子。
2. a 和 b 的任何因子都是 c 的因子。

以下是一个等价的定义:

$$\gcd(a, b) = \max[k, \text{其中 } k|a \text{ 且 } k|b]$$

因为我们所求的最大公因子是正数,所以 $\gcd(a,b)=\gcd(a,-b)=\gcd(-a,b)=\gcd(-a,-b)$。一般来说,$\gcd(a,b)=\gcd(|a|,|b|)$。

$$\gcd(60,24)=\gcd(60,-24)=12$$

同样地,因为 0 可被所有非零整数整除,所以 $\gcd(a,0)=|a|$。

如果 a 和 b 只有惟一的正公因子 1,我们则称整数 a 和 b 是互素的。也就是说,如果 $\gcd(a,b)=1$,那么 a 和 b 互素。

8 和 15 互素。因为 8 的正因子是 1,2,4 和 8,15 的正因子是 1,3,5 和 15,所以 1 是二者惟一的公因子。

4.3.2 求最大公因子

Euclid 算法基于以下的定理：对于任意非负整数 a 和任意正整数 b，有：

$$\gcd(a, b) = \gcd(b, a \bmod b) \tag{4.3}$$

> $\gcd(55,22) = \gcd(22,55 \bmod 22) = \gcd(22,11) = 11$

为保证等式(4.3)成立，使 d 等于 $\gcd(a,b)$。然后，根据 gcd 的定义，有 $d|a$ 和 $d|b$ 成立。对于任意正整数 b，a 可以表示为如下形式：

$$a = kb + r \equiv r \bmod b$$

$$a \bmod b = r$$

其中 k，r 为整数。因此，对某个整数 k，有 $(a \bmod b) = a - kb$。因为 $d|b$，$d|kb$，且 $d|a$，所以有 $d|(a \bmod b)$。这说明 d 是 b 和 $(a \bmod b)$ 的公因子。反之，如果 d 是 b 和 $(a \bmod b)$ 的公因子，那么 $d|kb$ 并且由此可知 $d|[kb + (a \bmod b)]$，即 $d|a$。因此，a 和 b 公因子的集合与 b 和 $a \bmod b$ 公因子的集合相等。因此，两对数的 gcd 相同，定理得证。

为了求出最大公因子，可多次重复使用(4.3)式。

> $\gcd(18,12) = \gcd(12,6) = \gcd(6,0) = 6$
> $\gcd(11,10) = \gcd(10,1) = \gcd(1,0) = 1$

如下所示的 Euclid 算法通过重复使用(4.3)式来求最大公因子。算法中假设 $a > b > 0$。因为 $\gcd(a,b) = \gcd(|a|,|b|)$，所以仅考虑正整数的情况即可。

EUCLID(a, b)

1. A ← a; B ← b
2. 若B = 0　　则返回 A = gcd(a, b)
3. R = A mod B
4. A ← B
5. B ← R
6. 转到 2

算法过程如下：

$$A_1 = B_1 \times Q_1 + R_1$$
$$A_2 = B_2 \times Q_2 + R_2$$
$$A_3 = B_3 \times Q_3 + R_3$$
$$A_4 = B_4 \times Q_4 + R_4$$

> 求gcd(1970,1066)
> 1970 = 1 × 1066 + 904　　gcd(1066,904)
> 1066 = 1 × 904 + 162　　gcd(904,162)

$904 = 5 \times 162 + 94$	gcd(162,94)
$162 = 1 \times 94 + 68$	gcd(94,68)
$94 = 1 \times 68 + 26$	gcd(68,26)
$68 = 2 \times 26 + 16$	gcd(26,16)
$26 = 1 \times 16 + 10$	gcd(16,10)
$16 = 1 \times 10 + 6$	gcd(10,6)
$10 = 1 \times 6 + 4$	gcd(6,4)
$6 = 1 \times 4 + 2$	gcd(4,2)
$4 = 2 \times 2 + 0$	gcd(2,0)
因此,gcd(1970,1066) = 2	

机警的读者可能会问,我们怎样才能知道这个过程会结束呢?也就是说,我们怎样才能确保在某个时刻 B 能整除 A?如果这个时候不会来临,我们将得出无穷无尽的正整数序列,每一个都比前面一个小,这显然是不可能的。

4.4 有限域 GF(p)

在 4.1 节中,我们把域定义为一个满足图 4.1 中所有公理的集合,并给出了一些无限域的例子。无限域在密码编码学中没有特别的意义,然而,有限域却在许多密码编码学算法中扮演着重要的角色。可以看出,有限域的元素个数必须是一个素数的幂 p^n,n 为正整数。我们将在第 8 章详细讨论素数。在这里,我们只需要知道素数是只有两个正整数因子(即 1 和它本身)的整数。

元素个数为 p^n 的有限域一般记为 GF(p^n);GF 代表 Galois field,以第一位研究有限域的数学家的名字命名。我们在此要关注两种特殊的情形。$n = 1$ 时的有限域 GF(p);它和 $n > 1$ 的有限域相比有着不同的结构,我们将在本节对它进行研究。在 4.6 节将研究 GF(2^n)形式的有限域。

4.4.1 元素个数为 p 的有限域

给定一个素数 p,元素个数为 p 的有限域 GF(p)定义为整数{0,1,…,$p-1$}的集合 Z_p,其运算为模 p 的代数运算。

我们曾在 4.2 节讲到,整数{0,1,…,$n-1$}的集合 Z_n,在模 n 的代数运算下,构成一个交换环(表 4.2)。我们进一步发现:Z_n 中的任一整数有乘法逆元,当且仅当该整数与 n 互素时[见(4.2)式的讨论]。若 n 为素数,Z_n 中所有的非零整数都与 n 互素,因此 Z_n 中所有非零整数都有乘法逆元。因此,我们可以在表 4.2 中列出的 Z_p 的性质加上下面一条:

乘法逆元(w^{-1})	任意 $w \in Z_p$,$w \neq 0$,存在 $z \in Z_p$,使得 $w \times z \equiv 1 \bmod p$

因为 w 和 p 互素,如果我们用 w 乘以 Z_p 的所有元素,得出的余数是 Z_p 中所有元素的另一种排列。因此,恰好只有一个余数值为 1。因而,Z_p 中有这样的整数,当它乘以 w,得余数 1。这个整数就是 w 的乘法逆元,记做 w^{-1},所以 Z_p 其实是一个有限域。而且,(4.2)式和乘法逆元的存在是一致的,不需要附加条件就可以被改写为:

$$\text{如果}(a \times b) \equiv (a \times c) \bmod p \text{ 那么} b \equiv c \bmod p \tag{4.4}$$

将(4.4)式的两边同乘以 a 的乘法逆元，可得：

$$((a^{-1}) \times a \times b) \equiv ((a^{-1}) \times a \times c) \bmod p$$

$$b \equiv c \bmod p$$

最简单的有限域是 GF(2)，它的代数运算可以简单地描述如下：

+	0	1
0	0	1
1	1	0

加

×	0	1
0	0	0
1	0	1

乘

w	$-w$	w^{-1}
0	0	—
1	1	1

求逆

在此，加等价于异或运算，乘等价于逻辑与运算。

表 4.3 描述了 GF(7)的代数运算。

表 4.3 GF(7)的代数运算

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) 模 7 加法

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) 模 7 乘法

w	$-w$	w^{-1}
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) 模 7 加法逆元和乘法逆元

4.4.2 在 GF(p)中求乘法逆元

当 p 值较小时，求 GF(p)中元素的乘法逆元很容易。我们只需构造一个乘法表，如表 4.3(b)，

所要的结果就可以从中直接读出。但是，当 p 值比较大时，这种方法是不切实际的。

如果 $\gcd(m,b)=1$，那么 b 有模 m 的乘法逆元。即，对于正整数 $b<m$，存在 $b^{-1}<m$ 使 $bb^{-1}=1 \bmod m$。Euclid 算法可以被扩展如下：求出 $\gcd(m,b)$ 之后，当 gcd 为 1 时，算法返回 b 的乘法逆元。

扩展的 Euclid(m, b)

1. (A1, A2, A3) ← (1, 0, m); (B1, B2, B3) ← (0, 1, b)
2. 若B3 = 0　　则返回A3 = gcd(m, b); 不可逆
3. 若B3 = 1　　则返回B3 = gcd(m, b); B2 = $b^{-1} \bmod m$
4. $Q = \left\lfloor \frac{A3}{B3} \right\rfloor$
5. (T1, T2, T3) ← (A1 − QB1, A2 − QB2, A3 − QB3)
6. (A1, A2, A3) ← (B1, B2, B3)
7. (B1, B2, B3) ← (T1, T2, T3)
8. 转到 2

在计算中有以下关系成立：

$$mT1 + bT2 = T3 \qquad mA1 + bA2 = A3 \qquad mB1 + bB2 = B3$$

为了保证算法正确返回 $\gcd(m,b)$，请注意，如果我们在 Euclid 算法中使 A 和 B 与扩展 Euclid算法中的 A3 和 B3 相等，那么对这两个变量的处理是一致的。在 Euclid 算法中的每一次迭代中，A 被赋值为先前的 B，B 被赋值为先前的 A mod B。类似地，在扩展 Euclid 算法中的每一步，A3 被赋值为先前的 B3，B3 被赋值为先前的 A3 减去 A3 的整数商乘以 B3。A3 的整数商为 A3 除以 B3 的余数，即 A3 mod B3。

同时注意到，如果 $\gcd(m,b)=1$，那么在最后一步我们将得到 B3 = 0 和 A3 = 1。因此，在上一步，B3 = 1。但是如果 B3 = 1，那么我们可以说：

$$mB1 + bB2 = B3$$
$$mB1 + bB2 = 1$$
$$bB2 = 1 + mB1$$
$$bB2 \equiv 1 \bmod m$$

且 B2 为 b 的模 m 乘法逆元。

表 4.4 是算法执行的一个例子。如表所示，$\gcd(550,1759)=1$，550 的乘法逆元是 355，即 $550 \times 350 \equiv 1 \bmod 1759$。

要得到该算法的详细证明，请参阅文献[KNUT97]。

表 4.4　在 GF(1759)中求 550 的乘法逆元

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	−3	109
5	1	−3	109	−5	16	5
21	−5	16	5	106	−339	4
1	106	−339	4	−111	355	1

4.5 多项式运算

在继续讨论有限域之前，我们需要介绍一个有趣的问题——多项式算术。我们只讨论一元多项式 x，且可把多项式运算分为三种：

- 使用代数基本规则的普通多项式运算
- 系数运算是模 p 运算的多项式运算，即系数在 Z_p 中
- 系数在 Z_p 中，且多项式被定义为模一个 n 次多项式 $m(x)$ 的多项式运算

本节讨论前两种多项式运算，下一节再讨论最后一种。

4.5.1 普通多项式运算

一个 n 次多项式（$n \geqslant 0$）的表达形式如下：

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

其中 a_i 是某个指定数集 S 中的元素，该数集叫做系数集，且 $a_n \neq 0$。我们称 $f(x)$ 是定义在系数集 S 上的多项式。

零次多项式称为**常数多项式**，是系数集里的一个元素。如果 $a_n = 1$，那么对应的 n 次多项式就被称为**首 1 多项式**。

在抽象代数中，我们一般不给多项式中的 x 赋一个特定值[例如 $f(7)$]。为了强调这一点，变元 x 有时被称做**不定元**。

多项式运算包含加法、减法和乘法，这些运算是把变量 x 当做集合 S 中一个元素来定义的。除法也以类似的方式定义，但这时要求 S 是域。这些域包括实数、有理数和素数 p 的 Z_p。注意整数集并不是域，也不支持多项式除法运算。

加法、减法的运算规则是把相应的系数相加减。因此，如果

$$f(x) = \sum_{i=0}^{n} a_i x^i; \qquad g(x) = \sum_{i=0}^{m} b_i x^i; \qquad n \geqslant m$$

那么加法运算定义为：

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i)x^i + \sum_{i=m+1}^{n} a_i x^i$$

乘法运算定义为：

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

其中：

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0$$

在最后一个公式中，我们令 $a_i = 0 (i > n)$，$b_i = 0 (i > m)$。注意结果的次数等于两个多项式的次数之和。

例如，令 $f(x)=x^3+x^2+2, g(x)=x^2-x+1$，则

$$f(x)+g(x)=x^3+2x^2-x+3$$
$$f(x)-g(x)=x^3+x+1$$
$$f(x)\times g(x)=x^5+3x^2-2x+2$$

手工计算过程见图 4.3(a)～图 4.3(c)。

4.5.2 系数在 Z_p 中的多项式运算

现在我们考虑这样的多项式，它的系数是域 F 的元素。这种情况下，容易看出这样的多项式集合是一个环，称为**多项式环**。也就是说，如果我们把每个不同的多项式看做集合中的元素，则这个集合是一个环[①]。

对域上的多项式进行多项式运算时，除法运算是可能的。注意这并不说能进行整除。我们来阐明两者的区别。在一个域中，给定两个元素 a 和 b，a 除以 b 的商也是这个域中的一个元素。然而，在非域的环 R 中，普通除法将得到一个商式和余式，这并不是整除。

考虑集合 S 中的除法运算 5/3。如果 S 是有理数集合(是一个域)，那么结果可以简单地表示成 5/3 且是 S 中的一个元素。现在假设 S 是域 Z_7，在这种情况下，用表 4.3(c)计算：

$$5/3=(5\times 3^{-1}) \bmod 7=(5\times 5) \bmod 7=4$$

这是一个整除。最后假设 S 是整数集(是环但不是域)，那么 5/3 的结果是商为 1 余数为 2：

$$5/3=1+2/3$$
$$5=1\times 3+2$$

因此，除法在整数集上并不是整除。

现在，如果试图在非域系数集上进行多项式除法，我们会发现除法运算并不总是有定义的。

如果系数集是整数集，那么$(5x^2)/(3x)$没有结果，因为需要一个值为 5/3 的系数，这在系数集中是没有的。同一个多项式，如果在 Z_7 上进行除法运算，有$(5x^2)/(3x)=4x$，这在 Z_7 中是一个合法的多项式。

然而，正如我们现在所见，即使系数集是一个域，多项式除法也不一定是整除。一般说来，除法会产生一个商式和一个余式：

$$\frac{f(x)}{g(x)}=q(x)+\frac{r(x)}{g(x)} \tag{4.5}$$
$$f(x)=q(x)g(x)+r(x)$$

如果 $f(x)$的次数是 n，$g(x)$的次数是 $m(m\leqslant n)$，那么商式 $q(x)$的次数是 $n-m$，余式的次数至多为 $m-1$。

① 事实上，系数在交换环中的多项式集合构成多项式环，但这个结论在此处并无多大意义。

与整数运算相似，我们可以在等式(4.5)中把余式 $r(x)$写为 $f(x) \bmod g(x)$。即 $r(x) = f(x) \bmod g(x)$。如果这里没有余式[即 $r(x) = 0$]，那么我们可以说 $g(x)$**整除** $f(x)$，记为 $g(x) | f(x)$；等价地，我们可以说 $g(x)$是 $f(x)$的一个**因式或除式**。

对于上个例子[$f(x) = x^3 + x^2 + 2$ 和 $g(x) = x^2 - x + 1$]，$f(x)/g(x)$产生一个商式 $q(x) = x + 2$ 和一个余式 $r(x) = x$，如图 4.3(d)所示。这很容易证实，只要注意到：

$$q(x)g(x) + r(x) = (x + 2)(x^2 - x + 1) + x = (x^3 + x^2 - x + 2) + x = x^3 + x^2 + 2 = f(x)$$

$$\begin{array}{r} x^3+x^2 \quad +2 \\ +\ (x^2-x+1) \\ \hline x^3+2x^2-x+3 \end{array}$$

(a) 加法

$$\begin{array}{r} x^3+x^2 \quad +2 \\ -\ (x^2-x+1) \\ \hline x^3 \quad +x+1 \end{array}$$

(b) 减法

$$\begin{array}{r} x^3+x^2 \quad +2 \\ \times\ (x^2-x+1) \\ \hline x^3+x^2 \quad +2 \\ -x^4-x^3 \quad -2x \\ \hline x^5 \quad +3x^2-2x+2 \end{array}$$

(c) 乘法

$$\begin{array}{r|l} & x+2 \\ \hline x^2-x+1 & x^3+x^2 \quad +2 \\ & x^3-x^2+x \\ \hline & 2x^2-x+2 \\ & 2x^2-2x+2 \\ \hline & x \end{array}$$

(d) 除法

图 4.3 多项式运算的一个例子

对我们而言，GF(2)上的多项式最有意义的。4.4 节中提到过，在 GF(2)中，加法等价于 XOR 运算，乘法等价于逻辑与运算。此外，模 2 的加法和减法是等价的：$1 + 1 = 1 - 1 = 0$；$1 + 0 = 1 - 0 = 1$；$0 + 1 = 0 - 1 = 1$。

图 4.4 给出了 GF(2)上多项式运算的一个例子。$f(x) = (x^7 + x^5 + x^4 + x + 1)$和 $g(x) = (x^3 + x + 1)$，图中显示了 $f(x) + g(x)$；$f(x) - g(x)$；$f(x) \times g(x)$和 $f(x)/g(x)$。注意 $g(x) | f(x)$。

域 F 上的多项式$f(x)$被称为**不可约的**(既约的)，当且仅当 $f(x)$不能表示为两个多项式的积[两个多项式都在 F 上，次数都低于 $f(x)$的次数]。与整数相似，一个不可约多项式也称为**素多项式**。

GF(2)上的多项式[①] $f(x) = x^4 + 1$ 是可约的，因为 $x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$。

① 如无标明，本章其余部分的多项式均是 GF(2)上的多项式。

考虑多项式 $f(x)=x^3+x+1$，易观察到 x 不是 $f(x)$ 的一个因式，很容易得出 $x+1$ 也不是 $f(x)$ 的一个因式：

$$
\begin{array}{r|l}
x+1 & x^2+x \\
\hline
 & x^3+x+1 \\
 & x^3+x^2 \\
\hline
 & x^2+x \\
 & x^2+x \\
\hline
 & 1
\end{array}
$$

因此 $f(x)$ 没有一次因式。但是很容易观察到：如果 $f(x)$ 是可约的，它一定有一个二次因式和一个一次因式。因此，$f(x)$ 是不可约的。

$$
\begin{array}{r}
x^7+x^5+x^4+x^3+x+1 \\
+(x^3+x+1) \\
\hline
x^7+x^5+x^4
\end{array}
$$

(a) 加法

$$
\begin{array}{r}
x^7+x^5+x^4+x^3+x+1 \\
-(x^3+x+1) \\
\hline
x^7+x^5+x^4
\end{array}
$$

(b) 减法

$$
\begin{array}{r}
x^7+x^5+x^4+x^3+x+1 \\
\times(x^3+x+1) \\
\hline
x^7+x^5+x^4+x^3+x+1 \\
x^8+x^6+x^5+x^4+x^2+x \\
x^{10}+x^8+x^7+x^6+x^4+x^3 \\
\hline
x^{10}+x^4+x^2+1
\end{array}
$$

(c) 乘法

$$
\begin{array}{r|l}
x^3+x+1 & x^4+1 \\
\hline
 & x^7+x^5+x^4+x^3+x+1 \\
 & x^7+x^5+x^4 \\
\hline
 & x^3+x+1 \\
 & x^3+x+1 \\
\hline
 & 0
\end{array}
$$

(d) 除法

图 4.4　GF(2)上的多项式运算示例

4.5.3　求最大公因式

我们可以通过定义最大公因式来扩展域上的多项式运算和整数运算之间的类比。如果：

1. $c(x)$能同时整除 $a(x)$和 $b(x)$。

2. $a(x)$和 $b(x)$的任何因式都是 $c(x)$的因式。

就称多项式 $c(x)$为 $a(x)$和 $b(x)$的最大公因式。

下面是一个等价的定义：$\gcd[a(x), b(x)]$是能同时整除 $a(x)$和 $b(x)$的多项式中次数最高的一个。

我们可以改写 Euclid 算法来计算两个多项式的最大公因式。式(4.3)中的等式可以改写为如下的定理：

$$\gcd[a(x), b(x)] = \gcd[b(x), a(x) \bmod b(x)] \tag{4.6}$$

用于多项式的 Euclid 算法可以表述如下：算法假设 $a(x)$的次数大于 $b(x)$的次数。那么，为了求 $\gcd[a(x), b(x)]$：

EUCLID$[a(x), b(x)]$

1. $A(x) \leftarrow a(x)$; $B(x) \leftarrow b(x)$
2. 若$B(x) = 0$　则返回 $A(x) = \gcd[a(x), b(x)]$
3. $R(x) = A(x) \bmod B(x)$
4. $A(x) \leftarrow B(x)$
5. $B(x) \leftarrow R(x)$
6. 转到 2

求 $\gcd[a(x), b(x)]$，其中 $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$，$b(x) = x^4 + x^2 + x + 1$。

$A(x) = a(x)$; $B(x) = b(x)$

$$\begin{array}{r|l} x^4+x^2+x+1 & x^2+x \\ \hline x^6+x^5+x^4+x^3+x^2+x+1 & \\ x^6\quad\ +x^4+x^3+x^2 & \\ \hline x^5\qquad\qquad +x+1 & \\ x^5\quad\ +x^3+x^2+x & \\ \hline x^3+x^2+1 & \end{array}$$

$R(x) = A(x) \bmod B(x) = x^3 + x^2 + 1$

$A(x) = x^4 + x^2 + x + 1$; $B(x) = x^3 + x^2 + 1$

$$\begin{array}{r|l} x^3+x^2+1 & x+1 \\ \hline x^4\quad\ +x^2+x+1 & \\ x^4+x^3\quad\ +x & \\ \hline x^3+x^2\quad\ +1 & \\ x^3+x^2\quad\ +1 & \\ \hline \end{array}$$

$R(x) = A(x) \bmod B(x) = 0$

$\gcd[a(x), b(x)] = A(x) = x^3 + x^2 + 1$

4.6 有限域 GF(2^n)

在这一章的前几节中,我们提到过有限域的元素个数必须为 p^n,其中 p 为素数,n 为正整数。在 4.4 节中,我们讨论了元素个数为 p 的有限域这一特殊情况。我们发现在使用 Z_p 上的模运算时,它满足域的所有条件(如图 4.1)。而以 p^n 为模的模运算却不一定能产生域。在本节中,我们将看到在一个具有 p^n 个元素的集合中,什么样的结构满足域的所有条件,并集中讨论 GF(2^n)。

4.6.1 动机

实际上所有的加密算法(包括对称密钥和公开密钥算法)都涉及到整数集上的算术运算。如果某种算法使用的运算之一是除法,那么我们就必须使用定义在域上的运算。为了方便使用和提高效率,我们希望这个整数集中的数与给定的二进制位所能表达的信息一一对应,而不出现浪费。也就是说,我们希望这个整数集的范围是从 0 到 $2^n - 1$,这样正好对应一个 n 位的字。

假设我们要构造一个传统加密算法,该算法一次处理 8 位的数据并且要使用除法。我们可以用 8 个二进制位来表示 0 到 255 之间的整数。但由于 256 不是素数,如果使用 Z_{256} 上的模运算(以 256 为模的运算),这个集合就不是一个域了。小于 256 的素数中最大的是 251,所以在使用以 251 为模的运算时,集合 Z_{251} 是一个域。然而在这种情况下,在 8 个二进制位所能表示的全部整数中,251 到 255 不能使用,造成了存储空间的浪费。

上面的例子指出,如果我们既要使用所有的运算,又要使用 n 个二进制位所能表示的全部整数,那么以 2^n 为模的运算是不行的。也就是说,当 $n > 1$ 时,以 2^n 为模的整数集不是一个域。退一步说,即使加密算法只使用加法和乘法而不使用除法以 Z_{2^n} 为模的整数集仍然存在问题,如下例所示。

假设我们在加密算法中要使用 3 位的信息块,并且只使用加法和乘法。如表 4.1 所示,以 8 为模的运算的定义是合理的。但是请注意,在乘法运算表中,各非零元素出现的次数是不同的。比如 3 只出现了 4 次而 4 却出现了 12 次。另一方面,还可以使用本节提到的形式为 GF(2^n)的有限域,即具有 $2^3 = 8$ 个元素的特定的有限域。这个域上的运算如表 4.5 所示。这样的话,各非零元素的出现次数就是一致的。两种情况对比如下:

非零元素	1	2	3	4	5	6	7
在 Z_8 中的出现次数	4	8	4	12	4	8	4
在 GF(2^3)中的出现次数	7	7	7	7	7	7	7

表 4.5 GF(2^3)上的运算

+	000	001	010	011	100	101	110	111
	0	1	2	3	4	5	6	7
000 0	0	1	2	3	4	5	6	7
001 1	1	0	3	2	5	4	7	6
010 2	2	3	0	1	6	7	4	5
011 3	3	2	1	0	7	6	5	4
100 4	4	5	6	7	0	1	2	3
101 5	5	4	7	6	1	0	3	2
110 6	6	7	4	5	2	3	0	1
111 7	7	6	5	4	3	2	1	0

(a) 加法

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

(b) 乘法

w	$-w$	w^{-1}
0	0	—
1	1	1
2	2	5
3	3	6
4	4	7
5	5	2
6	6	3
7	7	4

(c) 加法和乘法的逆运算

直觉告诉我们,一个将整数集不平均地映射到自身的算法用于加密时可能要弱于一个提供一一映射的算法。正因为如此,有限域 GF(2^n)对加密算法是很有吸引力的。

4.6.2 多项式模运算

设集合 S 由域 Z_p 上次数小于 $n-1$ 的所有多项式组成。每一个多项式具有如下形式:

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

其中,a_i 在集合{0, 1,…, $p-1$}上取值。S 中共有 p^n 个不同的多项式。

当 $p=3, n=2$ 时，集合中共有 $3^2=9$ 个多项式，分别是：

0	x	$2x$
1	$x+1$	$2x+1$
2	$x+2$	$2x+2$

当 $p=2, n=3$ 时，集合中共有 $2^3=8$ 个多项式，分别是：

0	$x+1$	x^2+x
1	x^2	x^2+x+1
x	x^2+1	

如果定义了合适的运算，那么每一个这样的集合 S 都是一个有限域。定义由如下几条组成：

1. 该运算遵循基本代数规则中的普通多项式运算规则。
2. 系数运算以 p 为模，即遵循有限域 Z_p 上的运算规则。
3. 如果乘法运算的结果是次数大于 $n-1$ 的多项式，那么必须将其除以某个次数为 n 的既约多项式 $m(x)$并取余式。对于多项式 $f(x)$，这个余式可表示为 $r(x)=f(x) \bmod m(x)$。

高级加密标准(AES)使用有限域 GF(2^8)上的运算，其中既约多项式 $m(x)=x^8+x^4+x^3+x+1$。考虑多项式 $f(x)=x^6+x^4+x^2+x+1$ 和 $g(x)=x^7+x+1$，则：

$$
\begin{aligned}
f(x)+g(x) &= x^6+x^4+x^2+x+1+x^7+x+1 \\
&= x^7+x^6+x^4+x^2 \\
f(x)\times g(x) &= x^{13}+x^{11}+x^9+x^8+x^7+ \\
&\quad x^7+x^5+x^3+x^2+x+ \\
&\quad x^6+x^4+x^2+x+1 \\
&= x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1
\end{aligned}
$$

$$
\begin{array}{r|l}
 & x^5+x^3 \\
\hline
x^8+x^4+x^3+x+1 & x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \\
 & x^{13}\quad\ +x^9+x^8\quad\ +x^6+x^5 \\
\hline
 & \quad +x^{11}\quad\ +x^4+x^3+1 \\
 & \quad +x^{11}\quad\ +x^7+x^6\quad\ +x^4+x^3 \\
\hline
 & \qquad\qquad\ +x^7+x^6+1
\end{array}
$$

所以，$f(x)\times g(x) \bmod m(x)=x^7+x^6+1$。

和简单模运算类似，多项式模运算中也有剩余集的概念。设 $m(x)$为 n 次多项式，则模 $m(x)$剩余集中有 p^n 个元素，其中每个元素都可以表示成一个 p^n 次多项式($m<n$)。

以 $m(x)$为模的剩余类$[x+1]$由所有满足 $a(x)\equiv(x+1) \bmod m(x)$的多项式 $a(x)$组成。也就是说，剩余类$[x+1]$中的所有多项式 $a(x)$满足等式 $a(x) \bmod m(x)=x+1$。

由此可见，以 n 次既约多项式 $m(x)$ 为模的所有多项式组成的集合满足图 4.1 的所有公理，于是可以形成一个有限域。还可以进一步看到，所有给定元素个数的有限域都是同构的，即任意两个给定元素个数的有限域具有相同的结构，但是元素的表示和标记可能不同。

为了构造有限域 GF(2^3)，我们需要选择一个 3 次既约多项式。只有两个满足条件的多项式：x^3+x^2+1 和 x^3+x+1。若选择后者，则 GF(2^3)上的加法和乘法表如表 4.6 所示。

表 4.6 以 x^3+x+1 为模的多项式运算

		000	001	010	011	100	101	110	111
	+	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) 加法

		000	001	010	011	100	101	110	111
	×	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	0	0	0	0	0	0	0	0	0
000	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) 乘法

4.6.3 求乘法逆元

正如 Euclid 算法可以用来求两个多项式的最大公因式一样，扩展 Euclid 算法可以用来求一个多项式的乘法逆元。如果多项式 $b(x)$ 的次数小于 $m(x)$ 且 $\gcd[m(x), b(x)]=1$，那么该算法能求出 $b(x)$ 以 $m(x)$ 为模的乘法逆元。若 $m(x)$ 为既约多项式，即除了本身与 1 之外没有其他因式，则始终有 $\gcd[m(x), b(x)]=1$。算法过程如下：

EXTENDED EUCLID[$m(x), b(x)$]

1. [A1(x), A2(x), A3(x)] ← [1, 0, $m(x)$]; [B1(x), B2(x), B3(x)] ← [0, 1, $b(x)$]
2. 若B3(x) = 0　　则返回A3(x) = gcd[$m(x), b(x)$]; 无乘法逆元
3. 若B3(x) = 1　　则返回B3(x) = gcd[$m(x), b(x)$]; B2(x) = $b(x)^{-1}$ mod $m(x)$
4. Q(x) = A3(x)/B3(x)的商
5. [T1(x), T2(x), T3(x)] ← [A1(x) − Q(x)B1(x), A2(x) − Q(x)B2(x), A3(x) − QB3(x)]
6. [A1(x), A2(x), A3(x)] ← [B1(x), B2(x), B3(x)]
7. [B1(x), B2(x), B3(x)] ← [T1(x), T2(x), T3(x)]
8. 转到2

表4.7给出了计算$(x^7 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1)$的乘法逆元的过程。结果是$(x^7 + x + 1)^{-1} = (x^7)$，即$(x^7 + x + 1)(x^7) \equiv 1 \bmod (x^8 + x^4 + x^3 + x + 1)$。

表4.7 推展的Euclid[$(x^7 + x + 1), (x^8 + x^4 + x^3 + x + 1)$]

初始化	$A1(x) = 1; A2(x) = 0; A3(x) = x^8 + x^4 + x^3 + x + 1$ $B1(x) = 0; B2(x) = 1; B3(x) = x^7 + x + 1$
迭代1	$Q(x) = x$ $A1(x) = 0; A2(x) = 1; A3(x) = x^7 + x + 1$ $B1(x) = 1; B2(x) = x; B3(x) = x^4 + x^3 + x^2 + 1$
迭代2	$Q(x) = x^3 + x^2 + 1$ $A1(x) = 1; A2(x) = x; A3(x) = x^4 + x^3 + x^2 + 1$ $B1(x) = x^3 + x^2 + 1; B2(x) = x^2 + 1; B3(x) = x$
迭代3	$Q(x) = x^3 + x^2 + x$ $A1(x) = x^3 + x^2 + 1; A2(x) = x^2 + 1; A3(x) = x$ $B1(x) = x^6 + x^2 + x + 1; B2(x) = x^7; B3(x) = 1$
迭代4	$B3(x) = \gcd[(x^7 + x + 1), (x^8 + x^4 + x^3 + x + 1)] = 1$ $B2(x) = (x^7 + x + 1)^{-1} \bmod (x^8 + x^4 + x^3 + x + 1) = x^7$

4.6.4 计算上的考虑

GF(2^n)中的多项式$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n-1} a_i x^i$可以由它的$n$个二进制系数$(a_{n-1}a_{n-2}\cdots a_0)$惟一地表示。因此，GF($2^n$)中的每个多项式都可以表示成一个$n$位的二进制整数。

表4.5和表4.6给出了GF(2^3)上以$m(x) = (x^3 + x + 1)$为模的加法和乘法表。其中，表4.5用二进制整数表示，而表4.6用多项式表示。

加法

我们发现这里的多项式加法是将相应的系数分别在Z_2上相加，而Z_2上的加法其实就是异或(XOR)运算。所以，GF(2^n)中两个多项式的加法等同于按位异或运算。

考虑前面例子中 GF(2⁸)上的两个多项式 $f(x)=x^6+x^4+x^2+x+1$ 和 $g(x)=x^7+x+1$。

$(x^6+x^4+x^2+x+1)+(x^7+x+1)$	=	$x^7+x^6+x^4+x^2$	(多项式表示)
$(01010111)\oplus(10000011)$	=	(11010100)	(二进制表示)
$\{57\}\oplus\{83\}$	=	$\{d4\}$	(十六进制表示)①

乘法

简单的异或运算不能完成 GF(2ⁿ)上的乘法。但是可以使用一种合理且容易实现的技巧。我们将在高级加密标准使用的有限域中讨论该技巧。这个有限域是 GF(2⁸),其中模多项式为 $m(x)=x^8+x^4+x^3+x+1$。

这个技巧基于下面的等式:

$$x^8 \bmod m(x) = [m(x) - x^8] = (x^4+x^3+x+1) \tag{4.7}$$

通过观察不难证明等式(4.7)是正确的。一般地,在 GF(2ⁿ)上对于 n 次多项式 $p(x)$,有 $x^n \bmod p(x)=[p(x)-x^n]$。

现在考虑 GF(2⁸)上的多项式 $f(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0$,将它乘以 x,可得:

$$x \times f(x) = (b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_7x^8+b_1x^2+b_0x) \bmod m(x) \tag{4.8}$$

如果 $b_7=0$,那么结果就是一个次数小于 8 的多项式,不需要进一步计算。如果 $b_7=1$,那么可以通过等式(4.7)进行除 $m(x)$取余运算:

$$x \times f(x) = (b_6x^7+b_7x^8+b_4x^5+b_3x^4+b_7x^8+b_1x^2+b_0x)+(x^4+x^3+x+1)$$

这表明乘以 x(如 00000010)的运算可以通过左移一位后按位异或 00011011 来实现,其表示为 (x^4+x^3+x+1)。总结如下:

$$x \times f(x) = \begin{cases} (b_6b_5b_4b_3b_2b_1b_00) & \text{若 } b_7=0 \\ (b_6b_5b_4b_3b_2b_1b_00)\oplus(00011011) & \text{若 } b_7=1 \end{cases} \tag{4.9}$$

乘以一个高于一次的多项式可以通过重复使用等式(4.9)来实现。这样一来,GF(2⁸)上的乘法可以用多个中间结果相加的方法实现。

在前面的例子中,我们给出了当 $f(x)=x^6+x^4+x^2+x+1$, $g(x)=x^7+x+1$, $m(x)=x^8+x^4+x^3+x+1$ 时,$f(x)\times g(x) \bmod m(x)=x^7+x^6+1$ 的计算过程。现在我们用二进制运算的方法重做一遍,即计算(01010111)×(10000011)。首先要求出 x 幂乘的中间结果:

(01010111) × (00000010) = (10101110)
(01010111) × (00000100) = (01011100) ⊕ (00011011) = (01000111)

① 在计算机科学学生资源网上的 WilliamStallings.com/StudentSupport.html 中有关于数制系统(十进制、二进制、十六进制)的相关资料。用一个十六进制字符表示 4 位二进制,一个字节的二进制数用括弧括起来的两个十六进制字符表示。

$$(01010111) \times (00001000) = (10001110)$$
$$(01010111) \times (00010000) = (00011100) \oplus (00011011) = (00000111)$$
$$(01010111) \times (00100000) = (00001110)$$
$$(01010111) \times (01000000) = (00011100)$$
$$(01010111) \times (10000000) = (00111000)$$

所以，$(01010111) \times (10000011) = (01010111) \times [(00000001) \oplus (00000010) \oplus (10000000)]$
$= (01010111) \oplus (10101110) \oplus (00111000) = (11000001)$

11000001 等价于 $x^7 + x^6 + 1$。

4.7 推荐读物和网址

[HERS75]是抽象代数的经典参考资料，严谨精密，可读性强，并且仍在继续印刷。[DESK92] 是另一个很好的参考资料来源。[KNUT98]是关于多项式运算比较全面的参考资料。[GARR01] 是一本涵盖内容广泛的参考资料。[BERL84] 是本章内容的最佳参考资料之一，仍在继续印刷。关于有限域的全面而严谨的参考资料是[LIDL94]。

BERL 84 Berlekamp, E. *Algebraic Coding Theory*. Laguna Hills, CA: Aegean Park Press, 1984.

DESK92 Deskins, W. *Abstract Algebra*. New York: Dover, 1992.

GARR01 Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology*. Upper Saddle River, NJ: Prentice Hall, 2001.

HERS75 Herstein, I. *Topics in Algebra*. New York: Wiley, 1975.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1998.

LIDL 94 Lidl, R., and Niederreiter, H. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press, 1994.

推荐网址：

- **PascGalois Project**：包含一系列巧妙的举例和演示，能帮助学生形象化地理解抽象代数的重要概念。

4.8 关键术语、思考题和习题

4.8.1 关键术语

交换群	结合律	交换律
交换环	循环群	因子
Euclid 算法	域	有限群
有限环	有限域	生成元
最大公因子	群	单位元
无限群	无限环	无限域
整环	逆元	既约多项式
模运算	多项式模运算	取模运算
首 1 多项式	多项式运算	多项式环
素数	素多项式	互素
余数	环	

4.8.2 思考题

4.1 定义一个简单的群。
4.2 定义一个简单的环。
4.3 定义一个简单的域。
4.4 解释“b 是 a 的因子”的含义。
4.5 模运算与普通运算的区别是什么?
4.6 列举三类多项式运算。

4.8.3 习题

4.1 设 S_n 是 n 个不同符号的所有置换组成的群。

a. S_n 中有多少个元素?

b. 说明当 $n > 2$ 时,S_n 不是交换群。

4.2 对于下列运算,判断模 3 剩余类组成的集合是否能构成一个群。

a. 加法

b. 乘法

4.3 集合 $S = \{a, b\}$ 上的加法和乘法定义如下:

+	a	b
a	a	b
b	b	a

×	a	b
a	a	a
b	a	b

判断 S 是否构成环,并证明你的结论。

4.4 在 4.2 节中我们定义同余关系如下:整数 a 和 b 模 n 同余,当且仅当($a \bmod n$) =

$(b \bmod n)$。然后证明若$n \mid (a-b)$,则 $a \equiv b \bmod n$。有些教材将后者作为同余关系的定义:整数 a 和 b 模 n 同余,当且仅当 $n \mid (a-b)$。以后一个定义为出发点,证明如果$(a \bmod n) = (b \bmod n)$,那么 n 整除$(a-b)$。

4.5 证明以下结论:

a. $a \equiv b \bmod n \rightarrow b \equiv a \bmod n$

b. $(a \equiv b \bmod n)$和$(b \equiv c \bmod n) \rightarrow a \equiv c \bmod n$

4.6 证明以下等式:

a. $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

b. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

4.7 求 Z_5 中各非零元素的乘法逆元。

4.8 证明任意十进制整数 N 和它的各位数字之和模 9 同余。例如,$475 \equiv 4+7+5 \equiv 16 \equiv 1+6 \equiv 7 \bmod 9$。

4.9 **a.** 求 gcd(24 140, 16 762)。

b. 求 gcd(4655, 12 075)。

4.10 本题的目的在于给 Euclid 算法的迭代轮数设置一个上限。

a. 设 $m = qn + r$,其中 $q > 0, 0 \leqslant r < n$,证明 $m/2 > r$。

b. 设 A_i 是 Euclid 算法进行 n 次迭代后 A 的值,证明 $A_{i+2} < A_i/2$。

c. 证明如果整数 m, n, N 满足 $1 \leqslant m,\ n \leqslant 2^N$,那么 Euclid 算法至多进行 $2N$ 次迭代就可求出 gcd$(m,\ n)$。

4.11 Euclid 算法已经有 2000 多年的历史并且一直都被数论学者们公认为最佳算法。经过这么多年以后,J.Stein 于 1961 年发明了一种有潜力与 Euclid 算法竞争的算法。该算法求 gcd$(A,\ B)(A,\ B \geqslant 1)$的过程如下。

第 1 步 令 $A_1 = A, B_1 = B, C_1 = 1$

第 n 步 (1) 若 $A_n = B_n$,则返回 $gcd(A、B) = A_n C_n$

(2) 若 A_n 和 B_n 均为偶数,则令 $A_{n+1} = A_n/2, B_{n+1} = B_n/2, C_{n+1} = 2C_n$

(3) 若 A_n 是偶数且 B_n 是奇数,则令 $A_{n+1} = A_n/2, B_{n+1} = B_n, C_{n+1} = C_n$

(4) 若 A_n 是奇数且 B_n 是偶数,则令 $A_{n+1} = A_n, B_{n+1} = B_n/2, C_{n+1} = C_n$

(5) 若 A_n 和 B_n 均为奇数,则令 $A_{n+1} = |A_n - B_n|, B_{n+1} = \min(B_n, A_n)$, $C_{n+1} = C_n$

继续第 $n+1$ 步

a. 分别用 Euclid 算法和 Stein 算法计算 gcd(2152, 764)。

b. Stein 算法相对于 Euclid 算法的一个明显优势是什么?

4.12 **a.** 证明若 Stein 算法在第 n 步前没有终止,则:

$$C_{n+1} \times \gcd(A_{n+1}, B_{n+1}) = C_n \times \gcd(A_n, B_n)$$

b. 证明若 Stein 算法在第 $n+1$ 步前没有终止,则:

$$A_{n-2}B_{n+2} \leqslant \frac{A_n B_n}{2}$$

c. 证明若 $1 \leqslant A,\ B \leqslant 2^N$,则 Stein 算法至多需要 $4N$ 步就能求出 gcd(m, n)。

d. 证明 Stein 算法的返回值确实是 gcd(A, B)。

4.13 用扩展 Euclid 算法求下列乘法逆元：

a. 1234 mod 4321 **b**. 24 140 mod 40 902 **c**. 550 mod 1769

4.14 证明由系数在域上的多项式组成的集合是一个环。

4.15 判断下列关于域上多项式的说法是否正确，并加以证明。

a. 首 1 多项式的乘积是首 1 多项式。

b. 次数分别为 m 和 n 的两个多项式的乘积的次数为 $m+n$。

c. 次数分别为 m 和 n 的两个多项式的和的次数为 $\max[m, n]$。

4.16 对于系数在 Z_{10} 上取值的多项式运算，分别计算：

a. $(7x+2)-(x^2+5)$

b. $(6x^2+x+3)\times(5x^2+2)$

4.17 判断下列多项式在 GF(2)上是否可约：

a. x^3+1

b. x^3+x^2+1

c. x^4+1

4.18 求下列各对多项式在相应有限域上的最大公因式。

a. x^3+x+1 和 x^2+x+1 在 GF(2)上。

b. x^3-x+1 和 x^2+1 在 GF(3)上。

c. $x^5+x^4+x^3-x^2-x+1$ 和 x^3+x^2+x+1 在 GF(3)上。

d. $x^5+88x^4+73x^3+83x^2+51x+67$ 和 $x^3+97x^2+40x+38$ 在 GF(101)上。

4.19 求 x^3+x+1 在 GF(2^4)上的乘法逆元，$m(x)=x^4+x+1$。

第 5 章　高级加密标准

美国国家标准技术局（NIST)在 2001 年发布了高级加密标准（AES)。AES 是一个对称分组密码算法,用来取代 DES,从而成为广泛使用的新标准。在本章中,我们首先介绍 NIST 挑选 AES 候选算法所使用的评估准则,然后讨论 AES 算法本身。

5.1　高级加密标准的评估准则

5.1.1　高级加密标准的起源

我们在第 3 章曾提及 NIST 在 1999 年发布了一个新版本的 DES 标准(FIPS PUB 46-3),该标准指出 DES 仅能用于遗留的系统,同时 3DES 将取代 DES 成为新的标准。我们将在第 6 章中描述 3DES。3DES 有两个显著的优点,确保了其能在未来得到广泛的应用。首先它的密钥长度是 168 位,故能克服 DES 所面临的穷举攻击问题。其次,3DES 的底层加密算法与 DES 的加密算法相同,该加密算法比任何其他加密算法受到分析的时间要长得多,也未能发现有比穷举攻击更有效的、基于算法本身的密码分析攻击方法。相应地，3DES 对密码分析攻击有很强的免疫力。如果仅考虑算法安全,3DES 能成为未来数十年加密算法标准的合适选择。

3DES 的根本缺点在于用软件实现该算法的速度比较慢。起初,DES 是为 20 世纪 70 年代中期的硬件实现所设计的,难以用软件有效地实现该算法。在 3DES 中轮的数量三倍于 DES 中轮的数量,故其速度慢得多。另一个缺点是 DES 和 3DES 的分组长度均为 64 位。就效率和安全性而言,分组长度应更长。

由于这些缺陷,3DES 不能成为长期使用的加密算法标准。故 NIST 在 1997 年公开征集新的高级加密标准(AES),要求安全性能不低于 3DES,同时应具有更好的执行性能。除了这些通常的要求之外,NIST 特别提出了高级加密标准必须是分组长度为 128 位的对称分组密码,并能支持长度为 128 位、192 位、256 位的密钥。

15 个候选算法通过了第一轮评估,而仅有 5 个候选算法通过了第二轮评估。NIST 在 2001 年 11 月完成了评估并发布了最终标准(FIPS PUB 197)。NIST 选择 Rijndael 作为 AES 算法。Rijndael 的作者是比利时的密码学家 Joan Daemen 博士和 Vincent Rijmen 博士。

高级加密标准最终会取代 3DES,但这个过程需要几年时间。NIST 估计 3DES 在可预见的未来将仍是一个被认可的算法(对美国政府的使用)。

5.1.2　AES 的评估

我们首先介绍 NIST 评估高级加密标准候选算法所使用的准则。这些准则涉及了现代对称分组密码在现实中的应用。事实上,这涉及到两套准则。NIST 在 1997 年开始征集候选算法时要求候选算法应具有表 5.1 所示的特征(特征按其重要性排列)[NIST97]。该准则中的三个类别如下:

- **安全性**:指用密码分析方法分析一个算法所需的代价。评估的重点在于能否防止实际的

攻击。AES 最短的密钥长度是 128 位，故使用目前技术的穷举攻击方式是不用考虑的。

- **成本**：NIST 期望 AES 能够广泛地应用于各种实际应用。故 AES 必须具有很高的计算效率，以便其能用于各种高速应用。
- **算法和执行特征**：它包含了各方面需要关注的事项，其中有：算法灵活性；算法适合于多种硬件和软件方式实现；算法的简洁性，即更便于分析算法的安全性。

表 5.1 NIST 对 AES 的评估准则(1997 年 9 月 12 日)

安全性

- **实际安全**：与其他候选算法相比(在密钥和分组长度一样的情况下)。
- **随机性**：算法的输出与输入块经过随机置换所得的输出是不可区别的。
- **可靠性**：算法的安全性要基于相应的数学基础。
- **其他的安全因素**：在评估过程中公众提出的因素，包括各种攻击方法。这些攻击方法可能使算法的实际安全强度低于算法提交者所声明的安全强度。

成本

- **专利要求**：NIST 要求当 AES 发布时，AES 使用的算法可以在全世界免费使用。
- **计算效率**：计算效率的评估将基于硬件与软件实现的评估。NIST 的第一轮分析将主要集中在软件实现的评估上，并特定于某一密钥 - 分组长度(128 位 - 128 位)的评估。第二轮分析将关注硬件实现和更多的密钥 - 分组长度的评估。计算效率主要指算法的执行速度。NIST 也会关注公众对每个候选算法执行效率的评论(特别是在各种平台和应用情况下)。
- **存储空间要求**：在评估过程中，NIST 也会关注执行每一个候选算法所需存储空间的大小(硬件和软件实现)。第一轮分析评估中，NIST 将主要考虑算法用软件实现时对存储空间的要求。第二轮主要考虑算法用硬件实现时对存储空间的要求。存储空间要求包括用硬件实现时所需的门数以及用软件实现时代码的长度和执行时所需 RAM 大小等因素。

算法和执行特征

- **灵活性**：具有更多灵活性的算法将会满足更多用户的要求，因此，相对而言是更可取的。然而，这并不要求算法提供一些不太实际的功能(如非常短的密钥长度)。灵活性包括但不限于如下的一些性质：
  a. 算法能够适应附加的密钥和分组长度[如 64 位的分组长度和其他可以接受的安全密钥长度(如在 128 位到 256 位之间且为 32 的倍数的密钥长度)等]。
  b. 算法能在各种平台和应用(如 8 位的 CPU、ATM 网络、语音和卫星通信、HDTV、B-ISDN 等)中安全有效地执行。
  c. 算法能够作为流密码、消息认证码发生器、随机数发生器、hash 算法等。
- **硬件与软件平台的适应性**：候选算法并不限定在只能用硬件实现。如果能在固件上得以实现，将会增加算法的灵活性。
- **简洁性**：候选算法的实现过程是否简洁。

根据这些准则，从最初的 21 个候选算法中选出了 15 个候选算法，后来又减少到了 5 个候选算法。在进行最终评估时，使用了最终评估准则(在[NECH00]中有相应的描述)。最终评估中使用了如下准则：

- **一般安全性**：为了评估一般安全性，NIST 依赖于密码学界的公共安全分析。在三年的评估过程中，许多密码专家发表了他们对各个密码算法的优点和缺点的分析报告。他们特别强调用诸如差分和线性密码分析等已知的攻击方式来分析各个候选算法。然而，与分析 DES 相比，分析 Rijndael 算法所用的时间及密码专家都非常有限。既然 AES 已选定，我们期待着密码学界中有对此的更多、更广泛的分析。
- **软件实现**：主要关注软件执行速度、跨平台执行能力以及随密钥长度的改变而使执行速度变化的情况。
- **受限空间环境**：在诸如智能卡等应用中，只有比较少的 RAM 和/或 ROM 空间可用于代

码存储(通常在 ROM 中)、诸如 S 盒(依据是否进行预计算以存储在 ROM 或 RAM 中)的数据目标存储方式以及子密钥的存储(在 RAM 中)。

- **硬件实现**:像软件一样,硬件执行提高执行速度或缩短代码长度。但硬件实现比起软件实现,其代码长度与费用更为相关。在具有大内存的普通计算机上,即使将加密程序的长度加倍也几乎不造成费用增加,但把硬件设备的存储区域加倍显然会大大地增加该设备的费用。
- **对执行的攻击**:一般安全性准则涉及到了依据密码算法所使用的数学性质来进行密码分析攻击。另一类攻击方式是在算法执行时利用某些物理措施来获取诸如密钥之类的定量信息。这种攻击方式结合了算法内部特性和算法执行时所依赖的环境特性。这种攻击的例子有计时攻击和能量分析攻击,其中计时攻击可参见第 3 章。由于智能卡执行时所消耗的能量与其执行的指令和数据的存取有关,能量分析攻击[KOCH98, BIHA00]的基本思想就是通过观测智能卡上密码算法在执行过程中任何特定时间所消耗的能量来获得一些有用的信息。例如,执行乘法比执行加法消耗更多能量,向存储器中写 1 比写 0 需要的能量要多。
- **加密与解密**:这个准则涉及到了与加密和解密相关的几个问题。如果加密和解密算法不同,那么解密代码也要占用存储空间。同时,无论两个算法是否一样,都可能在执行时存在时间上的差异。
- **密钥灵活性**:这是指利用少量的资源快速改变密钥的能力。这包括计算子密钥以及在子密钥已存在时,在正执行的不同安全集合中转变密钥的能力。
- **其他的多功能性和灵活性**:[NECH00]指出了有关这个方面的两个领域。参数灵活性包括易于支持其他的密钥和分组长度,也包括易于增加轮的数量以应对新发现的攻击。执行灵活性涉及到在特定环境下对密码元素进行优化。
- **指令级并行执行的潜力**:这个准则涉及到在当前和未来处理器中利用 ILP 特征的能力。

表 5.2 给出了 NIST 对 Rijndael 进行评估时所采用的准则。

表 5.2　NIST 对 Rijndael 的最终评估(2000 年 10 月 2 日)

一般安全性
没有已知的攻击方法能攻击 Rijndael。它用 S 盒作为非线性组件，Rijndael 表现出足够的安全性能，但一些专家认为可能利用该算法使用的数学结构来攻击该算法。另一方面，其简单的结构便于在评估高级加密标准候选算法过程中分析该算法的安全性能。
软件执行
Rijndael 非常利于在包括 8 位和 64 位以及 DSP 在内的各种平台上执行的加密和解密算法。然而，因执行轮数的增加而引起的密钥长度变长会降低算法的执行性能。Rijndael 固有的分布执行机制能够充分有效地利用处理器资源，甚至在不能分布执行的模型下仍能达到非常好的软件执行性能。同时，Rijndael 的密钥安装速度非常快。
受限空间环境
通常，Rijndael 非常适合在受限空间环境中执行加密或解密操作(不是两者都如此)。它对 RAM 和 ROM 的要求很低。在这样的环境中，如果既要执行加密操作又要执行解密操作，则其缺陷是它需要更大的 ROM 空间。因为加密和解密的主要步骤是不一样的。
硬件执行
在最后的五个候选算法中，Rijndael 在反馈模型下执行的速度最快，在非反馈模型下的执行速度位居第二。但当该算法的密钥长度为 192 位和 256 位时，因执行的轮数增加，其执行速度就变得很慢了。当用完全的流水线实现时，则该算法需要更多的存储空间，但不会影响其执行速度。

（续表）

对执行的攻击

Rijndael 所采用的实现方式非常利于防止能量攻击和计时攻击。与其他最终候选算法相比，Rijndeal 算法利用掩码技术使其具有防止这些攻击的能力，并未显著地降低该算法的执行性能。同时，它对 RAM 的需求仍在合理的范围内。当使用这些防攻击措施时，Rijndael 算法比其他的候选算法在执行速度上更有优势。

加密与解密

Rijndael 的加密函数和解密函数不同。一份 FPGA 研究报告指出，同时实现加密算法和解密算法所占用的存储空间比仅仅实现加密算法占用的存储空间要多约 60%。尽管在解密算法中密钥安装速度比在加密算法中速度要慢，但 Rijndael 执行加密和解密算法的速度差不多。

密钥灵活性

Rijndael 支持加密中的快速子密钥计算。Rijndael 要求在加密前用特定密钥产生所有子密钥。这给 Rijndael 的密钥灵活性稍微增加了一点资源负担。

其他的多功能性和灵活性

Rijndael 支持分组和密钥长度分别为 128 位、192 位和 256 位的各种组合。原则上，该算法结构能通过改变轮数来支持任意长度为 32 的倍数的分组和密钥长度。

指令级并行执行的潜力

Rijndael 对于单个分组加密有很好的并行执行能力。

5.2 AES 密码[①]

在 Rijndael 算法中，分组长度和密钥长度均能分别被指定为 128 位、192 位或 256 位。在高级加密标准规范中，密钥的长度可以使用三者中的任意一种，但分组长度只能是 128 位。高级加密标准中众多参数与密钥长度（见表 5.3）有关。在本章中，我们假定密钥的长度为 128 位，这可能是使用最广泛的实现方式。

表 5.3 AES 的参数

密钥长度（words/bytes/bits）	4/16/128	6/24/192	8/32/256
明文分组长度（words/bytes/bits）	4/16/128	4/16/128	4/16/128
轮数	10	12	14
每轮的密钥长度（words/bytes/bits）	4/16/128	4/16/128	4/16/128
扩展密钥长度（words/bytes）	44/176	52/208	60/240

Rijndael 具有如下特性：

- 所有已知的攻击具有免疫性。
- 在各种平台上，其执行速度快而且代码紧凑。
- 设计简单。

图 5.1 显示了 AES 的完整结构。加密算法的输入分组和解密算法的输出分组均为 128 位。在 FIPS PUB 197 中，输入分组是用以字节为单位的正方形矩阵描述。该分组被复制到 **State** 数组，这个数组在加密或解密的每个阶段都会被改变。在执行了最后的阶段后，**State** 被复制到输出矩阵中。这些操作在图 5.2(a)中描述。同样，128 位的密钥也是用以字节为单位的矩阵描述的。然后这个密钥被扩展到一个以字为单位的密钥序列数组中；每个字由 4 个字节组成，128 位的密钥最终扩展为

① 这一节的很多资料来源于[STAL02]。

44 字的序列[见图 5.2(b)]。注意在矩阵中字节排列顺序是从上到下从左到右排列的。加密算法中每个 128 位分组输入的前四个字节被按顺序放在了 in 矩阵的第一列，接着的四个字节放在了第二列，等等。同样，扩展密钥的前四个字节(一个字)被放在 w 矩阵的第一列。

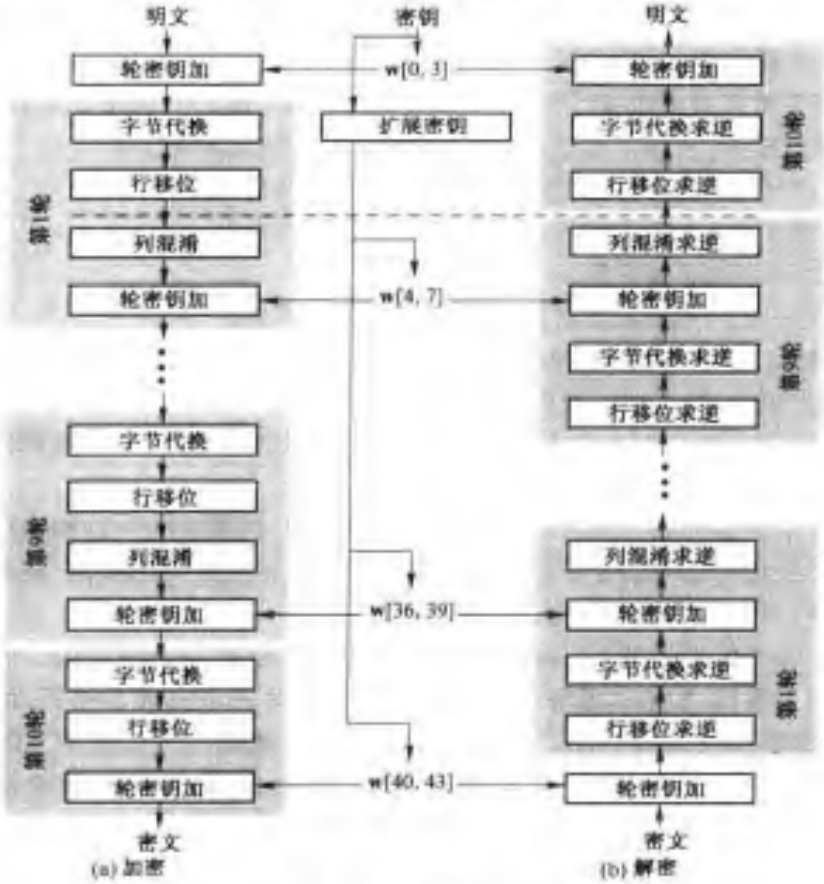


图 5.1　AES 的加密与解密

(a) 输入、state数据组和输出

(b) 密钥和扩展密钥

图 5.2　AES 的数据结构

在讨论 AES 的全部细节之前，我们先谈谈 AES 的结构：

1. AES 结构的一个显著特征是它不是 Feistel 结构。回想一下经典的 Feistel 结构，数据分组中的一半被用来修改数据分组中的另一半，然后交换这两部分。包括 Rijndael 在内的两个 AES 最终候选算法没有使用 Feistel 结构，而是每一轮都使用代换和混淆并行地处理整个数据分组。
2. 输入的密钥被扩展成由 44 个 32 位字所组成的数组 w[i]。从图 5.1 中可以看出，在每轮加解密过程中，有四个字(128 位)的密钥作为该轮的轮密钥。
3. 该结构由四个不同的阶段组成，包括一个混淆和三个代换：
   - **字节代换**：用一个 S 盒完成分组中的按字节的代换。
   - **行移位**：一个简单的置换。
   - **列混淆**：一个利用在域 GF(2^8)上的算术特性的代换。
   - **轮密钥加**：利用当前分组和扩展密钥的一部分进行按位 XOR。
4. 算法结构非常简单。对加密和解密操作，算法由轮密钥加开始，接着执行九轮迭代运算，每轮都包含所有四个阶段的代换，然后执行只包含三个阶段的最后一轮运算。图 5.3描述了包含全部加密轮的结构。
5. 仅仅在轮密钥加阶段中使用密钥。由于这个原因，该算法的开始和结束都有轮密钥加阶段。如果将其他不需要密钥的运算用于算法开始或结束的阶段，在不知道密钥的情况下就能计算其逆，故不能增加算法的安全性。
6. 轮密钥加实质上是一种 Vernam 密码形式，就其本身是不难破译的。而另外三个阶段一起提供了混淆、扩散以及非线性功能。因这些阶段没有涉及密钥，故就它们自身而言，并未提供算法的安全性。我们可把该算法看成是一个分组的 XOR 加密(轮密钥加)，接着是对这个分组的混淆(其他的三个阶段)，再接着又是 XOR 加密等交替执行的操作。这种方式非常有效且非常安全。
7. 每个阶段均可逆。对字节代换、行移位和列混淆，在解密算法中用与它们相对应的逆函数。轮密钥加的逆就是用同样的轮密钥和分组相异或，其原理就是 A⊕A⊕B = B。
8. 与大多数分组密码一样，解密算法按逆序方式利用了扩展密钥。然而，AES 的解密算法和加密算法并不一样。这是由 AES 的特定结构所决定的。
9. 一旦将所有的四个阶段求逆，很容易证明解密函数的确可以恢复原来的明文。图 5.1 中加密和解密流程在纵向上是相反的。在每个水平点上(如图中的虚线)，**State** 在加密和解密函数中是一样的。
10. 加密和解密过程的最后一轮均只包含三个阶段。这是由 AES 的特定结构所决定的，而且也是密码算法可逆性所要求的。

我们现在开始分别讨论 AES 中使用的四个阶段。对每个阶段我们首先描述正向算法(加密算法)、逆算法(解密算法)，接着讨论该阶段的合理性。最后我们讨论密钥扩展。

如我们在第 4 章提到的，AES 使用了基于有限域 GF(2^8)的不可约多项式[①] $m(x) = x^8 + x^4 + x^3 + x + 1$。Rijndael 的开发者说他们从 30 个 8 次不可约多项式中选择 $m(x)$，因为它是

① 在本章的其余部分，GF(2^8)均指由此多项式所确定的有限域。

[LIDL94]中给出的第一个不可约多项式。

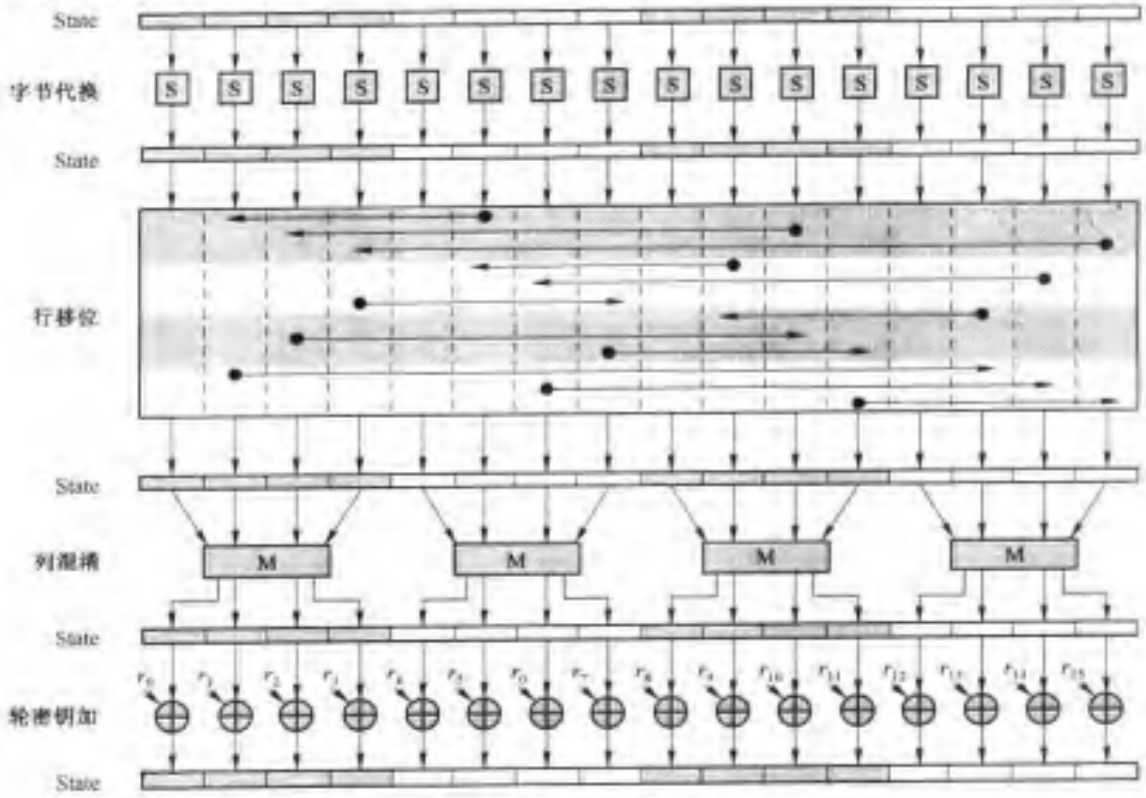


图 5.3　AES 的一轮加密过程

5.2.1　字节代换变换

正向和逆向变换

被称为字节代换的**正向字节代换变换**是一个简单的查表操作[图 5.4(a)]。AES 定义了一个 S 盒，它是由 16 × 16 个字节组成的矩阵，包含了 8 位值所能表达的 256 种可能的变换。**State** 中每个字节按照如下的方式映射为一个新的字节：把该字节的高 4 位作为行值，低 4 位作为列值，然后取出 S 盒中对应行列的元素作为输出。例如，十六进制值|95|[①] 所对应的 S 盒的行值是 9，列值是 5，S 盒中在此位置的值是|2A|。相应地，|95|被映射为|2A|。

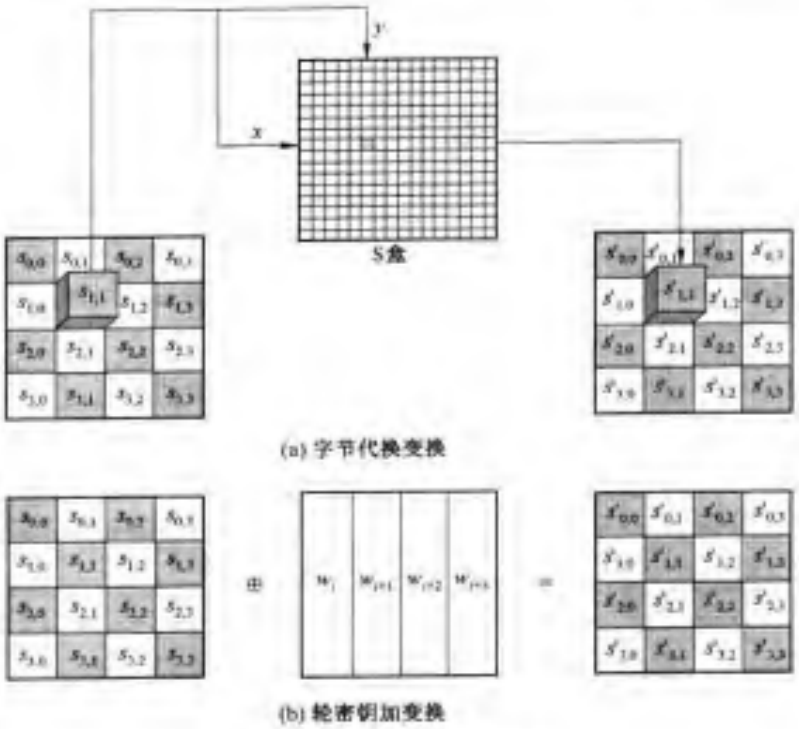
下面是一个字节代换变换的例子：

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

① 在 FIPS PUB 197 中，十六进制数由外加花括号的形式表示。本章中我们采用这种表示方式。



(a) 字节代换变换

(b) 轮密钥加变换

图 5.4 AES 的字节层操作

S 盒按如下的方式构造：

1. 逐行按升序排列的字节值初始化 S 盒。第一行是|00|，|01|，|02|，…，|0F|；第二行是|10|，|11|，…，|1F|等。因此，在行 x 列 y 的字节值是|xy|。
2. 把 S 盒中的每个字节映射为它在有限域 GF(2^8)中的逆；|00|被映射为它自身|00|。
3. 把 S 盒中的每个字节记成($b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$)。对 S 盒中每个字节的每位做如下的变换：

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \tag{5.1}$$

这里的 c_i 是指值为|63|的字节 c 的第 i 位。即($c_7\ c_6\ c_5\ c_4\ c_3\ c_2\ c_1\ c_0$) = (01100011)。符号(′)表示更新后的变量的值。AES 以如下的方式用矩阵描述了这个变换：

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{5.2}$$

我们解释一下方程(5.2)。在通常的矩阵乘法中,乘积矩阵中的每个元素是一行和一列相对应元素乘积的和。在这个例子中,乘积矩阵中每个元素是一个行和列所对应元素乘积按位异或的值。进一步而言,方程(5.2)中最终的加法是按位异或的。

我们考虑输入值为{95}的情况。在 $GF(2^8)$ 中{95}的乘法逆为$\{95\}^{-1} = \{8A\}$,用二进制表示就是 10001010。用方程(5.2)表示,就是:

$$\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}0\\1\\0\\1\\0\\0\\0\\1\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}1\\0\\0\\1\\0\\0\\1\\0\end{bmatrix}\oplus\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}0\\1\\0\\1\\0\\1\\0\\0\end{bmatrix}$$

得到的结果是{2A},这是 S 盒中行号为{09}、列号为{05}所对应的元素。这可以从表 5.4(a)中得到证实。

逆字节代换变换利用了表 5.4(b)中所示的逆 S 盒。例如,输入{2A}到逆 S 盒中,输出为{95};输入{95}到 S 盒中,输出为{2A}。逆 S 盒的构造方法是利用方程(5.1)中的逆变换,该逆变换是由 $GF(2^8)$ 上的逆变换得来的。该逆变换是:

$$b_i' = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

这里,字节 $d = \{05\}$,或者 00000101。我们可以用如下的方式来描述这个转换:

$$\begin{bmatrix}b_0'\\b_1'\\b_2'\\b_3'\\b_4'\\b_5'\\b_6'\\b_7'\end{bmatrix}=\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}+\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$$

为了理解逆字节代换变换是字节代换变换的逆,我们分别令字节代换变换和逆字节代换变换为 **X** 和 **Y**,常量 c、d 的矩阵表示方法为 **C** 和 **D**。对某个 8 位的矢量 **B**,方程(5.2)变成 **B′** = **XB**⊕**C**。我们需要证明 **Y**(**XB**⊕**C**)⊕**D** = **B**。将括号里的内容乘出来,即我们应该证明 **YXB**⊕**YC**⊕**D** = **B**。这变成了:

表 5.4 AES 的 S 盒

(a) S 盒

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) 逆 S 盒

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	23	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

$$\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus$$

$$\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=$$

$$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}$$

我们验证了 **YX** 等于单元矩阵，且 **YC** = **D**，于是 **YC**⊕**D** 等于零矢量。

评价

S 盒被设计成能防止已有的各种密码分析攻击。Rijndael 的开发者特别寻求在输入位和输出位之间几乎没有相关性的设计，且输出值不能通过利用一个简单的数学函数变换输入值[DAEM01]所得到。另外，在方程(5.1)中所选择的常量使得在 S 盒中没有不动点[S 盒(a) = a]，也没有“反不动点”[S 盒(a) = $\bar{a}$]。

当然，S 盒必须是可逆的，即逆 S 盒[S 盒(a)] = a。然而，因 S 盒(a) = 逆 S 盒(a)不成立，在这个意义上 S 盒不是自逆的。例如，S 盒(|95|) = |2A|，但逆 S 盒(|95|) = |AD|。

5.2.2　行移位变换

正向和逆向变换

图 5.5(a)描述了**正向行移位变换**。**State** 的第一行保持不变。把 **State** 的第二行循环左移一个字节，**State** 的第三行循环左移两个字节，**State** 的第四行循环左移三个字节。行移位变换的一个例子如下：

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

逆向行移位变换将 **State** 中的后三行执行相反方向的移位操作，如第二行向右循环移一个字节等。

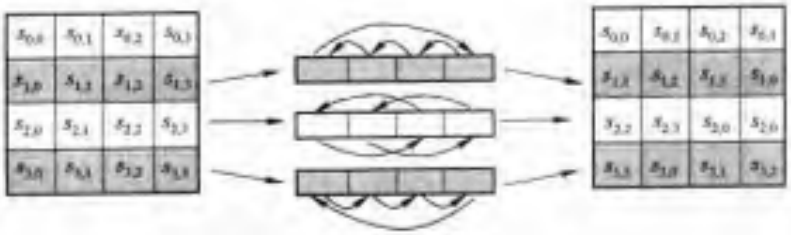
评价

行移位变换要比它看起来有用得多。这是因为 **State** 和密码算法的输入输出数据一样，是一个由 4 列所组成的数组，其中每列由 4 个字节组成。因此在加密过程中，明文的前四个字节直接被复制到 **State** 的第一列中，接着的四个字节被复制到 **State** 的第二列中，等等。进一步而言，如下面将要看到的那样，轮密钥也是逐列地应用到 **State** 上的。因此，行移位就是将某个字节从一列移到另一列中，它的线性距离是 4 字节的倍数。同时请注意这个变换确保了某列中的四字节被扩展到了 4 个不同的列。图 5.3 说明了这一点。

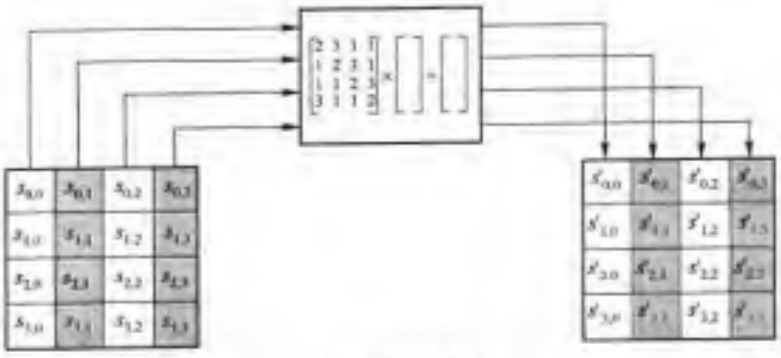
5.2.3 列混淆变换

正向和逆向变换

列混淆变换的**正向列混淆变换**对每列独立地进行操作。每列中的每个字节被映射为一个新值，此值由该列中的四个字节通过函数变换得到。这个变换可由下面的、基于**State**的矩阵乘法表示[见图 5.5(b)]：



(a) 行移位变换



(b) 列混淆变换

图 5.5 AES 的行与列操作

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.3}$$

乘积矩阵中的每个元素均是一行和一列中所对应元素的乘积之和。在这里的乘法和加法[①]都是定义在 GF(2^8)上的。**State** 中第 j 列($0 \leqslant j \leqslant 3$)的列混淆变换可表示为：

$$\begin{aligned} s'_{0,j} &= (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j}) \\ s'_{3,j} &= (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j}) \end{aligned} \tag{5.4}$$

列混淆变换的一个例子如下：

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

让我们来验证一下该例子的第一列。我们在第 4 章第 6 节中提到，在 GF(2^8)中，加法就是按位 XOR 操作，乘法是根据在方程(4.9)上所示的规则执行的。特别地，将某值乘上 x(如{02})，其结果就是将该值向左移一位，如果该值的最高位为 1，那么在移位后还要异或(0001 1011)。所以，为了验证第一列的列混淆变换，我们需要做：

$$\begin{aligned} (\{02\} \bullet \{87\}) &\oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} &&= \{47\} \\ \{87\} &\oplus (\{02\} \bullet \{6E\}) \oplus (\{03\} \bullet \{46\}) \oplus \{A6\} &&= \{37\} \\ \{87\} &\oplus \{6E\} \oplus (\{02\} \bullet \{46\}) \oplus (\{03\} \bullet \{A6\}) &&= \{94\} \\ (\{03\} \bullet \{87\}) &\oplus \{6E\} \oplus \{46\} \oplus (\{02\} \bullet \{A6\}) &&= \{ED\} \end{aligned}$$

对第一个方程，我们有 $\{02\} \cdot \{87\} = (0000\ 1110) \oplus (0001\ 1011) = (0001\ 0101)$；$\{03\} \cdot \{6E\} = \{6E\} \oplus (\{02\} \cdot \{6E\}) = (0110\ 1110) \oplus (1101\ 1100) = (1011\ 0010)$。于是有：

$$\begin{aligned} \{02\} \bullet \{87\} &= 0001\ 0101 \\ \{03\} \bullet \{6E\} &= 1011\ 0010 \\ \{46\} &= 0100\ 0110 \\ \{A6\} &= \underline{1010\ 0110} \\ & \quad 0100\ 0111 = \{47\} \end{aligned}$$

其他的方程也可以通过类似的方式得以验证。

① 我们遵从 FIPS PUB 197 的协定，使用符号 • 表示有限域 GF(2^8)上的乘法；使用 ⊕ 表示按位 XOR，这对应于 GF(2^8)中的加法。

逆向列混淆变换可由如下的矩阵乘法定义：

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{5.5}$$

从直观上，不容易看出方程(5.5)是方程(5.3)的逆。我们需要进行如下运算：

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} =$$

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

这等价于：

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6}$$

即逆变换矩阵乘以正向变换矩阵的结果为单位矩阵。为了验证方程(5.6)的第一列，我们需要进行如下运算：

$$\begin{aligned} (\{0E\} \bullet \{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\} \bullet \{03\}) &= \{01\} \\ (\{09\} \bullet \{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\} \bullet \{03\}) &= \{00\} \\ (\{0D\} \bullet \{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\} \bullet \{03\}) &= \{00\} \\ (\{0B\} \bullet \{02\}) \oplus \{0D\} \oplus \{09\} \oplus (\{0E\} \bullet \{03\}) &= \{00\} \end{aligned}$$

对第一个等式，我们有$\{0E\} \cdot \{02\} = 00011100$；而且$\{09\} \cdot \{03\} = \{09\} \oplus (\{09\} \cdot \{02\}) = 00001001 \oplus 00010010 = 00011011$。于是：

$$\begin{aligned} \{0E\} \bullet \{02\} &= 00011100 \\ \{0B\} &= 00001011 \\ \{0D\} &= 00001101 \\ \{09\} \bullet \{03\} &= \underline{00011011} \\ &\quad 00000001 \end{aligned}$$

利用类似的方法可以验证其他的方程。

AES 文档描述了另一种刻画列混淆变换的方式，即利用数学上的多项式来表示。在 AES 中，列混淆变换可以通过将 **State** 的每列考虑为一个系数在 GF(2^8)上的四次多项式来定义。每列乘上一个固定的多项式 $a(x)$，然后模(x^4+1)，$a(x)$的定义如下：

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{5.7}$$

附录 5A 论述了将 **State** 的每列乘上 $a(x)$能写成方程(5.3)中的矩阵乘法。相似地，能把方程(5.5)中的变换的每列看成是一个四次多项式且把该列乘上 $b(x)$。$b(x)$的定义如下：

$$b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} \tag{5.8}$$

显而易见 $b(x) = a^{-1}(x) \bmod (x^4 + 1)$。

评价

在方程(5.3)中，矩阵的系数基于码字间有最大距离的线性编码，这使得在每列的所有字节中有良好的混淆性。列混淆变换和行移位变换使得在经过几轮变换后，所有的输出位均与所有的输入位相关。关于此方面的详细讨论参见[DAEM99]。

此外，列混淆变换的系数，即{01}、{02}、{03}受算法执行效率的影响。正如上文所述，这些系数的乘法涉及到至多一次移位和一次 XOR。于是，为了与列混淆变换中所选择的系数相对应，逆向列混淆变换中的系数并不是出于效率的考虑。然而，加密被视为比解密更重要，原因如下：

1. 仅在加密过程中使用 CFB 和 OFB 密码模型(图 3.13 和图 3.14)。
2. 与任何其他的分组密码一样，AES 能用于构造消息验证码(第二部分)，这仅仅用在了加密过程中。

5.2.4　轮密钥加变换

正向和逆向变换

在**正向轮密钥加变换**中，128 位的 **State** 按位与 128 位的密钥 XOR。如图 5.4(b)中所示的那样，我们能把这个操作看成是基于 **State** 列的操作，即把 **State** 的一列中的四个字节与轮密钥的一个字进行异或；我们也能将其视为字节级别的操作。下面是轮密钥加的一个例子：

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

例子中第一个矩阵是 **State**，第二个矩阵是轮密钥。

逆向轮密钥加变换与正向轮密钥加变换相同，因为异或操作是其本身的逆。

评价

轮密钥加变换非常简单，却能影响 **State** 中的每一位。密钥扩展的复杂性和 AES 的其他阶段运算的复杂性，确保了该算法的安全性。

5.2.5 AES的密钥扩展

密钥扩展算法

AES密钥扩展算法的输入值是4字(16字节)密钥,输出值是一个44字(156字节)的一维线性数组。这足以为初始轮密钥加阶段和算法中的其他10轮中的每一轮提供4字的轮密钥。下面用伪码描述了这个扩展:

```
KeyExpansion(byte key[16], word w[44])
{
    word temp
    for(i=0;i<4;i++)  w[i]=(key[4*i],key[4*i+1], key[4*i+2] ,key[4*i+3]);
    for(i=4;i<44;i++)
    {
        temp = w[i-1];
        if(i mod 4=0)  temp = SubWord(RotWord(temp))⊕Rcon[i/4];
        w[i]=w[i-4] ⊕temp;
    }
}
```

输入密钥直接被复制到扩展密钥数组的前四个字。然后每次用四个字填充扩展密钥数组余下的部分。在扩展密钥数组中,w[i]的值依赖于w[i-1]和w[i-4]。在四种情形中,三种使用了异或。对w数组中下标为4的倍数的元素,采用了更复杂的函数来计算。图5.6阐明了如何计算扩展密钥数组的前8个字节,其中使用符合g来表示这个复杂函数。函数g由下述的子功能组成:

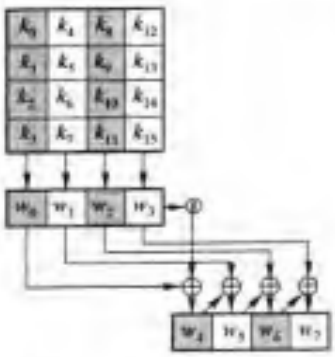


图5.6 AES的密钥扩展

1. 字循环的功能是使一个字中的四个字节循环左移一个字节。即将输入字[b0,b1,b2,b3]变换成[b1,b2,b3,b0]。
2. 字节代换利用S盒对输入字中的每个字节进行字节代换[见表5.4(a)]。
3. 步骤1和步骤2的结果再与轮常量Rcon[j]相异或。

轮常量是一个字，这个字最右边三个字节总为 0。因此与 Rcon 中的一个字相异或，其结果只是与该字最左的那个字节相异或。每轮的轮常量均不同，其定义为 Rcon[j] = (RC[j],0,0,0)，其中 RC[1] = 1，RC[j] = 2·RC[j − 1]，且乘法定义在域 GF(2^8)上。RC[j]的值以十六进制表示为：

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

例如，假设第八轮的轮密钥为：

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

那么第九轮的轮密钥的前四个字节(第一列)能按如下的方式计算：

i(十进制)	temp	RotWord 后	SubWord 后	Rcon(9)	与 Rcon 进行 XOR 后	w[i − 4]	w[i] = temp⊕w[i − 4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

评价

Rijndael 的开发者设计了密钥扩展算法来防止已有的密码分析攻击。使用与轮相关的轮常量，是为了防止不同轮中产生的轮密钥的对称性或相似性。[DAEM99]中使用的标准如下：

- 知道密钥或轮密钥的部分位不能计算出轮密钥的其他位。
- 它是一个可逆的变换[即知道扩展密钥中任何连续的 *Nk* 个字能够重新产生整个扩展密钥(*Nk* 是构成密钥所需的字数)]。
- 能够在各种处理器上有效地执行。
- 使用轮常量来排除对称性。
- 将密钥的差异性扩散到轮密钥中的能力；即密钥的每个位能影响到轮密钥的一些位。
- 足够的非线性，以防止轮密钥的差异完全由密钥的差异所决定。
- 易于描述。

作者并未量化上述列表的第一点，但指出了如果你知道的密钥或某个轮密钥少于 *Nk* 个连续字，那么你将难于构造出其余的未知位。知道的密钥的位数量越少，就越难于重构出或推测出密钥扩展中的其他位。

5.2.6　对应的逆算法

如上所述，AES 的解密算法和加密算法不同(图 5.1)。尽管在加密和解密中密钥扩展的形式一样，但在解密中变换的顺序与加密中变换的顺序不同。其缺陷在于对同时需要加密和解密的应用而言，需要两个不同的软件或固件模块。然而，解密算法的一个等价版本与加密算法有同样的结构。这个版本与加密算法的变换顺序相同(用逆变换取代正向变换)。为了达到这个目标，需要对密钥扩展进行改进。

两处改进使解密算法的结构与加密算法的结构一致。在加密过程中，其轮结构为字节代换、行移位、列混淆、轮密钥加。在标准的解密过程中，其轮结构为逆向行移位、逆向字节代换、轮密钥加、逆向列混淆。因此，在解密轮中的前两个阶段应交换，后两个阶段也需交换。

交换逆向行移位和逆向字节代换

逆向行移位影响在 **State** 中字节的顺序，但并不更改字节的内容，同时也不依赖字节的内容来进行它的变换。逆向行移位影响 **State** 中字节的内容，但不更改字节的顺序，同时也不依赖字节的顺序来进行它的变换。因此，这两个操作可以交换。如对一个给定的 **State** S_i，有：

$$\text{InvShiftRows}\,[\text{InvSubBytes}\,(S_i)] = \text{InvSubBytes}\,[\text{InvShiftRows}\,(S_i)]$$

交换轮密钥加和逆向列混淆

轮密钥加和逆向列混淆并不更改 **State** 中字节的顺序。如果我们将密钥看成是字的序列，那么轮密钥加和逆向列混淆每次都对 **State** 的一列进行操作。这两个操作对列输入是线性的。即对给定的 **State** S_i 和给定的轮密钥 w_j，有：

$$\text{InvMixColumns}\,(S_i \oplus w_j) = [\text{InvMixColumns}\,(S_i)] \oplus [\text{InvMixColumns}\,(w_j)]$$

假定 **State** S_i 的第一列是序列 (y_0, y_1, y_2, y_3)，轮密钥 w_j 的第一列为 (k_0, k_1, k_2, k_3)。则我们有：

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus$$

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

我们来论证第一列。我们有：

$$\begin{aligned} &[\{0E\} \bullet (y_0 \oplus k_0)] \oplus [\{0B\} \bullet (y_1 \oplus k_1)] \oplus [\{0D\} \bullet (y_2 \oplus k_2)] \oplus [\{09\} \bullet (y_3 \oplus k_3)] = \\ &\quad [\{0E\} \bullet y_0] \oplus [\{0B\} \bullet y_1] \oplus [\{0D\} \bullet y_2] \oplus [\{09\} \bullet y_3] \oplus \\ &\quad [\{0E\} \bullet k_0] \oplus [\{0B\} \bullet k_1] \oplus [\{0D\} \bullet k_2] \oplus [\{09\} \bullet k_3] \end{aligned}$$

我们可以看到这个方程是正确的。因此，假使先对轮密钥应用逆向列混淆，我们可以交换轮密钥加和逆向列混淆。注意我们无需对第一次或最后一次轮密钥加变换的输入进行逆向列混淆操作，因为这两个轮密钥加变换并不与逆向列混淆可交换，来产生等价的解密算法。

图 5.7 描述了等价的解密算法。

5.2.7 实现

文献[DAEM99]针对如何用像智能卡等 8 位处理器、个人电脑等 32 位处理器有效实现 Rijndael算法提出了一些建议。

8 位处理器

AES 能在 8 位处理器上非常有效地实现。轮密钥加是按位异或操作。行移位是简单的移

字节操作。字节代换是在字节级别上进行操作的，而且只要求一个 256 字节的表。

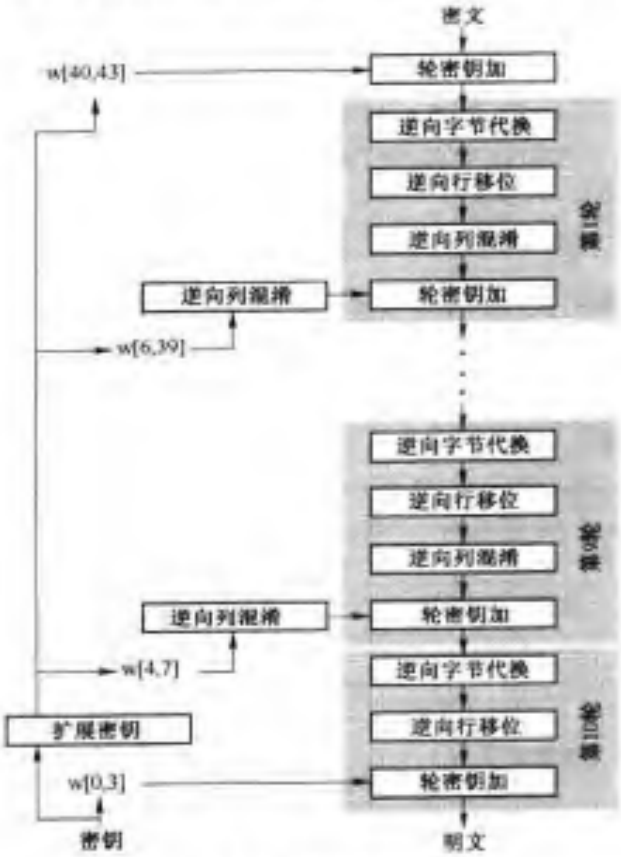


图 5.7　等价逆密码

列混淆变换要求在域 GF(2^8)上的乘法，即所有的操作都是基于字节的。列混淆仅要求乘以|02|和|03|，正如我们所看到的一样，这涉及到简单的移位、条件异或和异或。若不用移位或条件异或操作，算法的执行将更有效。方程集合(5.4)是说明在单列上进行列混淆变换的等式。利用性质 $|03| \cdot x = (|02| \cdot x) \oplus x$，我们能用如下的方式重写方程集合(5.4)：

$$\begin{aligned} Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\ s'_{0,j} &= s_{0,j} \oplus Tmp \oplus [2 \bullet (s_{0,j} \oplus s_{1,j})] \\ s'_{1,j} &= s_{1,j} \oplus Tmp \oplus [2 \bullet (s_{1,j} \oplus s_{2,j})] \\ s'_{2,j} &= s_{2,j} \oplus Tmp \oplus [2 \bullet (s_{2,j} \oplus s_{3,j})] \\ s'_{3,j} &= s_{3,j} \oplus Tmp \oplus [2 \bullet (s_{3,j} \oplus s_{0,j})] \end{aligned} \tag{5.9}$$

方程集合(5.9)能通过扩展和约去项的方式加以验证。

乘以|02|包含一次移位和一次条件异或操作。这种实现方式可能易于受到 3.4 中所描述的计时攻击。为了防止这种攻击并提高执行的效率，可以使用查表的方式取代乘法操作，其代价是需要花费一定存储空间。我们定义一个包含 256 字节的表 X2，使得 $X2[i] = |02| \cdot i$。因此，等式集合(5.9)可被描述为：

$$Tmp = s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{0,j} = s_{0,j} \oplus Tmp \oplus X2[s_{0,j} \oplus s_{1,j}]$$
$$s'_{1,j} = s_{1,j} \oplus Tmp \oplus X2[s_{1,j} \oplus s_{2,j}]$$
$$s'_{2,j} = s_{2,j} \oplus Tmp \oplus X2[s_{2,j} \oplus s_{3,j}]$$
$$s'_{3,j} = s_{3,j} \oplus Tmp \oplus X2[s_{3,j} \oplus s_{0,j}]$$

32 位处理器

上一节所描述的实现方式仅是基于在 8 位处理器上的操作。对 32 位处理器，若将操作定义在 32 位的字上，就能执行更有效的操作。为了论述这一点，我们首先利用代数形式定义一轮的四个变换。假如我们利用 $a_{i,j}$ 表示 **State** 矩阵，利用 $k_{i,j}$ 表示轮密钥矩阵，那么我们能用如下的方式来描述这些变换：

字节代换	$b_{ij} = \mathrm{S}[a_{ij}]$
行移位	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
列混淆	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
轮密钥加	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

在行移位等式中，列下标需要模 4。我们能把所有的这些表达式表示成一个等式：

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} =$$

$$\left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet \mathrm{S}[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet \mathrm{S}[a_{1,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet \mathrm{S}[a_{2,j-2}] \right) \oplus$$

$$\left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet \mathrm{S}[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

在第二个方程中，我们用矢量的线性组合来表示矩阵的乘法。我们定义 4 个 256 字（1024 字节）的表如下：

$$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[x] \right) \quad T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[x] \right) \quad T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[x] \right) \quad T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[x] \right)$$

因此，每个表接受一个字节的输入，同时输出一个列矢量（一个 32 位的字），我们可将该字节值看成是 S 盒输入的一个函数。这些表能被事先计算。

我们用如下的方式定义轮函数操作：

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

结果，基于上述等式的实现仅需要四个查找表、每轮每列的四次异或，以及存储这些表所需的 4 K 字节存储空间。Rijndael 的开发者们认为这种紧凑、有效的实现方式可能是选择 Rijndael 作为高级加密标准的最重要因素之一。

5.3　推荐读物和网址

到目前为止，对 AES 的最完整描述是由 AES 的开发者提供的[DAEM02]一书。作者也在[DAEM01]中给出了简单的描述和设计原理。

DAEM 01　Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard." *Dr. Dobb's Journal*, March 2001.

DAEM 02　Daemen, J., and Rijmen, V. *The Design of Rijndael: The Wide Trail Strategy Explained*. New York, Springer-Verlag, 2000.

推荐网址：

- **AES 主页**：NIST 关于高级加密标准的页面。包含这个标准及其有关的一些文档。
- **Rijndael 主页**：由 Rijndael 的开发者所维护。包含文档、到执行的链接以及其他相关的链接。

5.4　关键术语、思考题和习题

5.4.1　关键术语

高级加密标准（AES）
美国国家标准技术研究所（NIST）
能量攻击
Rjindael 算法
S 盒

5.4.2 思考题

5.1 NIST 评估 AES 的最初标准是什么?
5.2 NIST 评估 AES 的最终标准是什么?
5.3 什么是能量分析?
5.4 Rijndael 和 AES 有何不同?
5.5 使用 **State** 数组的目的是什么?
5.6 如何构造 S 盒?
5.7 简述什么是字节代换。
5.8 简述什么是行移位变换。
5.9 行移位变换影响了 **State** 中的多少字节?
5.10 简述什么是列混淆。
5.11 简述什么是密钥加变换。
5.12 简述密钥扩展算法。
5.13 字节代换和字代换有何不同?
5.14 行移位变换和字循环函数有何不同?
5.15 AES 解密算法和 AES 的逆算法之间有何不同?

5.4.3 习题

5.1 在讨论列混淆和逆向列混淆时,提到了:

$$b(x) = a^{-1}(x) \bmod (x^4 + 1)$$

在这里,$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$,且 $b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$。请证明这个等式的正确性。

5.2 **a**. 在 GF(2^8)上{01}的逆是什么?
b. 验证{01}在 S 盒中的输入。

5.3 当 128 位的密钥是全 0 时,给出密钥扩展数组中的前 8 个字节。

5.4 若明文是{000102030405060708090A0B0C0D0E0F},密钥是{01010101010101010101010101010101},
a. 用 4×4 的矩阵来描述 **State** 的最初内容。
b. 给出初始化轮密钥加后 **State** 的值。
c. 给出字节代换后 **State** 的值。
d. 给出行移位后 **State** 的值。
e. 给出列混淆后 **State** 的值。

5.5 验证等式(5.11)。即验证 $x^i \bmod (x^4 + 1) = x^{i \bmod 4}$。

5.6 比较 AES 和 DES。对如下所述的 DES 中的元素,指出 AES 中与之相对应的元素或解释 AES 中为什么不需要该元素:
a. f 函数的输入与子密钥相异或。

b. f 函数的输出与分组最左的部分相异或。

c. f 函数。

d. 置换 P。

e. 交换分组长度相等的两部分。

5.7 在关于实现的小节中，描述了利用表的方式来防止计时攻击。请提出另一种可防止计时攻击的技术。

5.8 在关于实现小节中，提出了仅用一个代数方程描述加密算法一个典型轮的全部四个阶段。请给出与第 10 轮等价的方程。

附录 5A　系数在 GF(2^8)中的多项式

在 4.5 节中，我们讨论了系数定义在 Z_p 上的多项式算术和模 $M(x)$的多项式，其中 $M(x)$的次数为 n。在这里，多项式系数的加法和乘法都定义在域 Z_p 上，即加法和乘法都要模 p。

AES 文档描述的多项式算术中的多项式的系数在 GF(2^8)上，且次数不大于 3。其中使用了如下的规则：

1. 加法操作就是两个多项式的系数在 GF(2^8)上相加。正如 4.5 节中所指出的，如果我们把 GF(2^8)中的元素看成是 8 位的串，那么加法就等价于异或操作。于是我们有：

$$a(x) = a_3x^3 + a_3x^3 + a_1x + a_0 \tag{5.10}$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \tag{5.11}$$

那么

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

2. 乘法操作和一般多项式乘法差不多，但有两个限制：

a. 系数是在 GF(2^8)上相乘。

b. 结果多项式需要模多项式($x^4 + 1$)。

我们应对所讨论的多项式有清醒的认识。回顾 4.6 节，GF(2^8)中的每个元素可以表示成系数为{0}或{1}、次数不大于 7 的多项式。多项式乘积的结果需要模一个次数为 8 的多项式。相应地，GF(2^8)中的每个元素都能看成是一个 8 位的字节，该字节中每一位都与相对应多项式的系数对应。对这一章所定义的集合，我们定义了一个多项式环，该环中的每个多项式的系数在 GF(2^8)中，次数不大于 3。多项式乘积的结果需要模一个次数为 4 的多项式。相应地，这个环中的每个元素均可看成是一个 4 字节的字，该字中的每一个字节值都与该元素所对应多项式的 8 位系数相对应。

我们定义 $a(x)$和 $b(x)$的模乘积为 $a(x) \otimes b(x)$。为计算 $d(x) = a(x) \otimes b(x)$，首先执行没有模操作的乘法并将具有相同次数的项相合并。我们把这个过程称为 $c(x) = a(x) \times b(x)$。然后有：

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_3x^3 + c_1x + c_0 \tag{5.12}$$

这里，

$$c_0 = a_0 \bullet b_0$$
$$c_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1)$$
$$c_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2)$$
$$c_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$
$$c_4 = (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$
$$c_5 = (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$
$$c_6 = a_3 \bullet b_3$$

最后一步是执行模操作：

$$d(x) = c(x) \bmod (x^4 + 1)$$

即 $d(x)$必须满足方程：

$$c(x) = [(x^4 + 1) \times q(x)] \oplus d(x)$$

以使 $d(x)$的次数不大于 3。

在多项式环上进行乘法的一种实现方式基于：

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4} \tag{5.13}$$

如果我们联合方程(5.10)和方程(5.11)，可得到：

$$\begin{aligned} d(x) &= c(x) \bmod (x^4 + 1) = [c_6x^2 + c_5x + c_4 + c_3x^3 + c_3x^3 + c_1x + c_0] \bmod (x^4 + 1) \\ &= c_3x^3 + (c_2 \oplus c_6)x^2 + (c_1 \oplus c_5)x + (c_0 \oplus c_4) \end{aligned}$$

将系数 c_i 进行扩展，我们将得到关于 $d(x)$系数的如下方程：

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$
$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$
$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$$
$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

其矩阵表达方式可写为：

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{5.14}$$

列混淆变换

在讨论列混淆变换时，我们说明了有两种等价的方式来定义这个变换。第一个是方程(5.3)中所示的矩阵乘法，即：

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

第二种方式是将 **State** 中的每列看成是一个系数在 GF(2^8)中的四次多项式。每列乘上固定的多项式 $a(x)$，然后模$(x^4 + 1)$，其中 $a(x)$为：

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

据方程(5.8),我们有 $a_3 = \{03\}$; $a_2 = \{01\}$; $a_1 = \{01\}$; $a_0 = \{02\}$。对 **State** 中的第 j 列,我们有多项式 $\text{col}_j(x) = s_{3,j}x^3 + s_{2,j}x^2 + s_{1,j}x + s_{0,j}$。代入方程(5.12),我们能将 $d(x) = a(x) \times \text{col}_j(x)$ 表示为:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} s_{o,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{o,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

这与方程(5.3)等价。

乘以 x

考虑环中一个多项式乘上 x 的情况:$c(x) = x \otimes b(x)$。我们有:

$$\begin{aligned} c(x) &= x \otimes b(x) = [x \times (b_3x^3 + b_2x^2 + b_1x + \mathrm{b}_0)] \bmod (x^4 + 1) \\ &= (b_3x^4 + b_3x^3 + b_1x^2 + \mathrm{b}_0x) \bmod (x^4 + 1) \\ &= b_3x^3 + b_1x^2 + \mathrm{b}_0\,x + b_3 \end{aligned}$$

因此,多项式乘上 x 就相当于将这个多项式所对应的、由四字节所组成的字循环左移一个字节。如果把这个多项式作为一个四字节的列矢量,那么就有:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

第6章 对称密码

本章探讨当前最重要的几种对称密码算法。这些算法的选择基于以下标准：

1. 有相当高的密码强度。
2. 广泛应用于 Internet 上。
3. 代表了自 DES 以来的现代对称密码技术。

本章讨论的算法有：三重 DES、Blowfish 和 RC5。随后，总结了优秀对称分组密码的重要特征。本章的最后讨论了一种流行的对称流密码，即 RC4。

6.1 三重 DES 算法

DES 在穷举攻击之下相对比较脆弱，因此很多人希望用某种算法替代它。一种方案是设计全新的算法，本章将给出几个这样的例子。还有一种方案，能够保护已有软件和硬件使用 DES 的投资，那就是用 DES 进行多次加密，且使用多个密钥。我们以第二种替代方案的一个简单例子开始，再讨论已被广泛接受的三重 DES(3DES)算法。

6.1.1 双重 DES

多次加密的最简单形式是进行两次加密，每次使用不同的密钥[图 6.1(a)]。给定明文 P 及密钥 $K1$ 和 $K2$，密文 C 按下述方式生成：

$$C = \mathrm{E}_{K_2}[\mathrm{E}_{K_1}[P]]$$

解密时逆序使用这两个密钥：

$$P = \mathrm{D}_{K_1}[\mathrm{D}_{K_2}[C]]$$

对于 DES，这种方法的密钥长度为 $56 \times 2 = 112$ 位，密码强度增加了。不过我们还是要对它进行仔细分析。

对单步加密进行推导

考虑下述说法：对所有的 56 位密钥 K_1 和 K_2，可能存在密钥 K_3，使得：

$$\mathrm{E}_{K_2}[\mathrm{E}_{K_1}[P]] = \mathrm{E}_{K_3}[P] \tag{6.1}$$

若该说法成立，问题就来了，那就是实际上不管用 DES 进行了多少次加密运算，都是没有用的，因为它的效果等同于用一个 56 位密钥进行一次 DES 加密的效果。

表面看来，等式(6.1)不太可能成立。DES 加密是 64 位分组之间的映射，实际上映射就是重排。这就是说，对 2^{64} 个可能的明文分组，DES 用某个特定密钥加密后都惟一对应一个 64 位的密文分组。否则，即使两个输入块映射为一个输出块，解密也不可能。那么，2^{64} 个可能的输入，有多少个一对一映射呢？这个数量是：

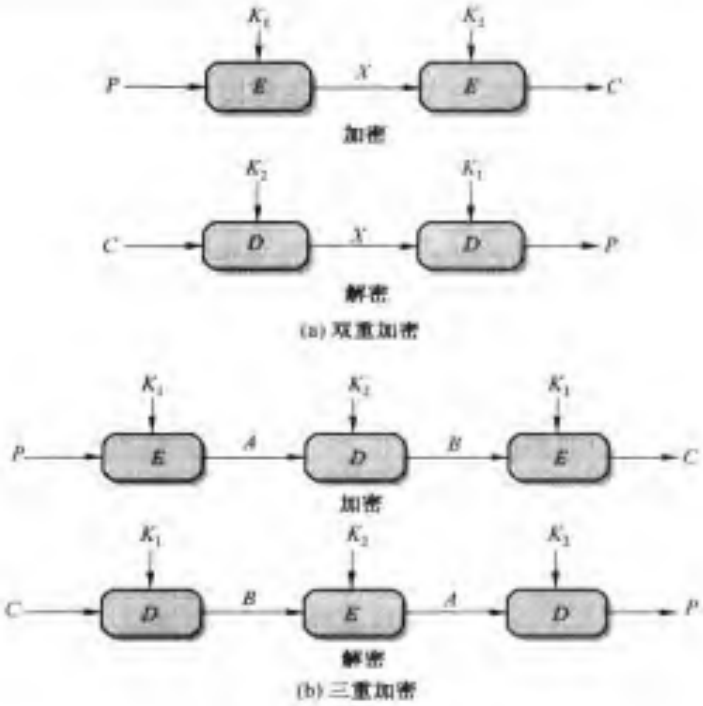


图 6.1 多重加密

$$(2^{64})! = 10^{3473800000000000000} > (10^{10^{20}})$$

另一方面，DES 为每个密钥定义了一个映射，映射总数为：

$$2^{56} < 10^{17}$$

所以，完全有理由认为，双重 DES 所对应的映射，绝大部分不为单个 DES 的应用所定义。尽管有很多支持该假设的证据，但直到 1992 年该假设才被证明，见文献[CAMP92]。

中间相遇攻击

虽然双重 DES 对应的映射与单 DES 对应的映射不同，但是还有另外一种攻击方法，这种方法不是针对 DES 的，而是针对所有分组密码的。

这就是基于观察的中间相遇攻击算法，文献[DIFF77]首次对它进行了描述。假设：

$$C = \mathrm{E}_{K_2}[\mathrm{E}_{K_1}[P]]$$

则有[见图 6.1(a)]：

$$X = \mathrm{E}_{K_1}[P] = \mathrm{D}_{K_2}[C]$$

给定明密文对(P, C)，攻击如下展开：首先，将 P 按密钥 K_1 所有可能的 2^{56} 个值加密，得到的结果按 X 的值排序放在表 T 内，然后将 C 用密钥 K_2 所有可能的 2^{56} 个值解密，每解密一次，就将解密结果与 T 中的值比较，如果有相等的，就用刚才测试的两个密钥对 P 加密，若结果为 C，则认定这两个密钥是正确的密钥。

对任意给定的明文 P，采用双重 DES 加密后可能得到 2^{64} 个密文中的一个，双重 DES 使用了 112 位密钥，所以密钥空间为 2^{112}。因此对明文 P，可产生密文 C 的密钥个数平均为 $2^{112}/2^{64}$

$=2^{48}$。故上述攻击过程对第一个(P, C)对将产生2^{48}个错误的结果。而对第二个(P, C)对，错误结果的概率就降为$2^{48-64}=2^{-16}$。换句话说，中间相遇攻击使用两组已知明密文对时，其猜出正确密钥的概率为$1-2^{-16}$。结论是：已知明文攻击可以成功对付密钥长度为112位的双重DES。其付出的数量级为2^{56}，比攻击单DES所需的数量级2^{55}多不了多少。

6.1.2 使用两个密钥的三重DES

对付中间相遇攻击的一个明显方法是使用三个密钥进行三次加密。这样，已知明文攻击的代价将升为2^{112}数量级，这远远超出了人们现行甚至是未来的能力。然而，它需要长为$56\times3=168$位的密钥，这无疑是个缺点。

Tuchman建议仅使用两个密钥进行三次加密[见图6.1(b)]了[TUCH79]。具体运算过程是加密－解密－加密(EDE)，写成式子为：

$$C=\mathrm{E}_{K_1}[\mathrm{D}_{K_2}[\mathrm{E}_{K_1}[P]]]$$

第二步采用解密运算并没有什么密码学上的深层含义，这仅是为了使用三重DES的用户可以利用该算法解密单DES加密的数据。因为：

$$C=\mathrm{E}_{K_1}[\mathrm{D}_{K_2}[\mathrm{E}_{K_1}[P]]]=\mathrm{E}_{K_1}[P]$$

使用两个密钥的三重DES已经广泛地替代了DES，并已用于密钥管理标准ANS X9.17和ISO 8732中。

当前，还没有对3DES的可行攻击方法。Coppersmith[COPP94]分析后认为，对3DES的穷举攻击的代价是$2^{112}\approx(5\times10^{33})$数量级，且估计用差分密码分析的代价是按指数增长的，与单DES比较超出10^{52}倍。

我们不妨看看几种对3DES的攻击方法，虽然都不实际，但对以后产生好的攻击方法也许有所裨益。

Merkle和Hellman[MERK81]提出了一个有启发性的攻击方法，即挨个查看可能的明文，看哪一个明文的第一个中间值[见图6.1(b)]$A=0$，然后再使用中间会合攻击来得到密钥。这种攻击方法的代价是2^{56}数量级，但是它需要2^{56}个选择明密文对，这是不可能的。

文献[OORS90]中概述了一种已知明文攻击，相对于选择明文攻击，这种方法有些进步，但付出的代价也增多了。攻击基于如下观察：若已知A和C[见图6.1(b)]，那问题相当于对双DES的攻击。当然，只要不知道密钥，即使知道了P和C，攻击者还是得不到A。攻击者可选择A的一个可能值，再试着找出一个可产生A的已知(P, C)对。具体的攻击过程如下：

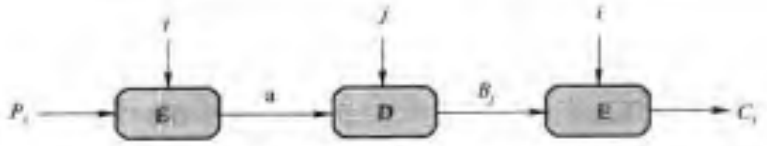
1. 获取n个(P, C)对，将这些已知(P, C)对按P排序放在表中[见图6.2(b)]了。
2. 对A随意选择值a，创建第2张表[见图6.2(c)]，表的入口按下列方式定义。对每个可能的$K_1=i$，计算可产生a的明文值P_i：

$$P_i=\mathrm{D}_i[a]$$

对所有能与表1中的入口匹配的P_i，在表2中创建一个入口，其中有K_1的值以及该(P, C)对产生的B的值，即：

$$B=\mathrm{D}_i[C]$$

最后将表2按B的值排序。



(a) 两个密钥三重加密与候选密钥对

(b) n个已知明文—密文对的表，按P排序

(c) 中间值和候选密钥表

图 6.2 对 3DES 的已知明文攻击

3. 现在表 2 中有了一批 K_1 的候选值，它们可用来搜索 K_2。对任意可能的 $K_2 = j$，对选定的 a 计算第 2 个中间值：

$$B_j = \mathrm{D}_j[a]$$

在表 2 中查找 B_j，若有相等者，则相应的密钥 i 及 j 的这个值可认为是未知密钥(K_1，K_2)的候选值。为什么？因为我们找到了一个密钥对(i,j)，它可以生成已知的(P,C)对[见图 6.2(a)]。

4. 对这样的(i,j)再进行几个其他的(P,C)对的测试，要是测试出来用某些 P 按密钥(i,j)不能产生 C，则返回步骤 1 选择一个新的 a。否则，便大功告成：(i,j)就是密钥。

对给定的(P,C)，选择 a 成功的可能性为 2^{-64}。所以，给定 n 个(P,C)对，单个选择的 a 成功的可能性为 $n/2^{64}$。概率论中有这样的结果：从一个装有 n 个红球和 $N-n$ 个绿球的箱子里，平均要摸$(N+1)/(n+1)$次才可能摸到一个红球。所以在 n 非常大的时候，尝试的 a 的个数为：

$$\frac{2^{64}+1}{n+1} \approx \frac{2^{64}}{n}$$

因此，攻击的期望运算时间的数量级为：

$$(2^{56})\frac{2^{64}}{n} = 2^{120-\log_2 n}$$

6.1.3 使用三个密钥的三重 DES

尽管上述攻击方法不可行，但是所有使用双密钥 3DES 算法的人还是感觉有点悬。因此，很多人觉得采用三密钥 3DES 算法才是最好方案(例如文献[KALI96a])。三密钥 3DES 的密钥长度为 168 位，定义成：

$$C = E_{K_3}[D_{K_2}[E_{K_1}[P]]]$$

要想和 DES 兼容，只需设 $K_3 = K_2$ 或 $K_1 = K_2$ 就可以了。

有些基于 Internet 的应用已经采纳了这种三密钥的 3DES，例如第 14 章要讨论的 PGP 和 S/MIME。

6.2 Blowfish 算法

Blowfish 算法是由 Bruce Schneier 设计的一种对称分组密码（参考文献[SCHN93, SCHN94]）。Blowfish 算法有如下性质：

- **快速**：使用 32 位微处理器，Blowfish 算法可以达到每 18 个时钟周期加密 1 字节的速度。
- **紧凑**：运行 Blowfish 算法只需 5 K 的内存。
- **简单**：Blowfish 算法结构简单，易于实现，且易于确定算法强度。
- **安全性可变**：密钥长度可变，最长可达 448 位，这使得用户可在安全性和速度之间进行权衡。

Blowfish 算法的分组长度是 64 位，它已在很多产品上得到了实现，并接受了相当多的考验。至今为止，Blowfish 算法的安全性还未受到挑战。

6.2.1 子密钥和 S 盒的产生

Blowfish 算法的密钥长度可在 32 位到 448 位之间变化，即字长若为 32 位，密钥长度可为 1 个字到 14 个字。这个密钥用来产生 18 个 32 位的子密钥和 4 个 8 × 32 的 S 盒，这些 S 盒总共有 1024 个 32 位项，加上子密钥一共是 1042 × 32 位，或 4168 字节。

密钥可表示为数组 K：

$$K_1, K_2, \cdots, K_j \qquad 1 \leqslant j \leqslant 14$$

子密钥可表示为数组 P：

$$P_1, P_2, \cdots, P_{18}$$

4 个 S 盒，每个有 256 项，每项 32 位，可表示为：

$$S_{1,0}, S_{1,1}, \cdots, S_{1,255}$$
$$S_{2,0}, S_{2,1}, \cdots, S_{2,255}$$
$$S_{3,0}, S_{3,1}, \cdots, S_{3,255}$$
$$S_{4,0}, S_{4,1}, \cdots, S_{4,255}$$

产生数组 P 和 S 盒的步骤如下：

1. 依次初始化 $P_1, \cdots, P_{18}, S_{1,0}, \cdots, S_{1,255}, S_{2,0}, \cdots, S_{2,255}, S_{3,0}, \cdots, S_{3,255}, S_{4,0}, \cdots S_{4,255}$。方法是：将常数 π 的小数部分，按每 32 位依次分配给这些项。所以，P_1 是 π 的小数部分的最左 32 位，等等。写成十六进制：

$$
\begin{aligned}
P_1 &= 243F6A88\\
P_2 &= 85A308D3\\
&\cdots\\
S_{4,254} &= 578FDFE3\\
S_{4,255} &= 3AC372E6
\end{aligned}
$$

2. 将 P 和 K 按位异或，并重复使用 K。例如，对最大长度的密钥（14 个 32 位），$P_1 = P_1 \oplus K_1$，$P_2 = P_2 \oplus K_2$，$\cdots$，$P_{14} = P_{14} \oplus K_{14}$，$P_{15} = P_{15} \oplus K_1$，$\cdots$，$P_{18} = P_{18} \oplus K_4$。
3. 用当前的 P 和 S 加密 64 位的全 0 数据块，用输出结果替换 P_1 和 P_2。
4. 用当前的 P 和 S 加密步骤 3 的输出结果，用这次的输出结果替换 P_3 和 P_4。
5. 重复执行上述步骤，直到 P 和 S 全部被更新为止，每一步所使用的 Blowfish 算法都在不断变化。

整个更新 P 和 S 的过程如下所示：

$$
\begin{aligned}
P_1, P_2 &= \mathrm{E}_{P,S}[0]\\
P_3, P_4 &= \mathrm{E}_{P,S}[P_1 \| P_2]\\
&\cdots\\
P_{17}, P_{18} &= \mathrm{E}_{P,S}[P_{15} \| P_{16}]\\
S_{1,0}, S_{1,1} &= \mathrm{E}_{P,S}[P_{17} \| P_{18}]\\
&\cdots\\
S_{4,254}, S_{4,255} &= \mathrm{E}_{P,S}[S_{4,252} \| S_{4,253}]
\end{aligned}
$$

其中 $\mathrm{E}_{P,S}[Y]$ 表示使用 S 和 P 时 Blowfish 算法加密 Y 的结果。

产生最终的 P 和 S 需要整整执行 521 次 Blowfish 算法的加密过程。所以 Blowfish 算法不太适合于密钥经常改动的应用。此外，为了快速执行，每次使用算法时都要把 P 和 S 重算一遍，不如把 P 和 S 存起来，只需要大约 4 K 字节的存储空间。所以，Blowfish 算法也不适合于存储容量有限的应用，例如智能卡。

6.2.2　加密和解密

Blowfish 算法使用两个基本的运算：

- **加法**：字的模 2^{32} 加法，记为 +。
- **按位异或**：记为 $\oplus$。

这两种运算不可转换。这个性质很重要，它使得密码分析更困难。

图 6.3(a)描述了加密的过程。明文分成两半，即 LE_0 和 RE_0，各为 32 位。我们用变量 LE_i 和 RE_i 来表示第 i 轮迭代后数据的左半部分和右半部分。算法可以用下面的伪代码定义：

```
for i = 1 to 16 do
        RE_i = LE_(i-1) ⊕ P_i;
        LE_i = F[RE_i] ⊕ RE_(i-1);
LE_17 = RE_16 ⊕ P_17;
RE_17 = LE_16 ⊕ P_18;
```

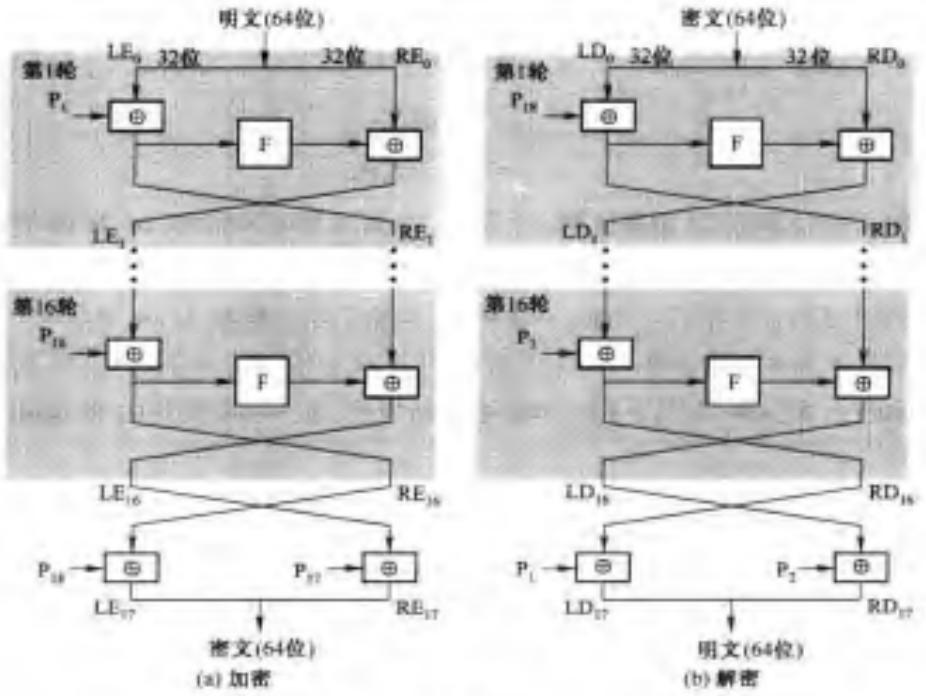


图 6.3 Blowfish 算法的加密和解密

得到的密文即变量 LE_{17} 和 RE_{17}。函数 F 如图 6.4 所示。F 的 32 位输入分成 4 个字节，记为 a、b、c 和 d。F 定义为：

$$F[a, b, c, d] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$$

所以，每一轮迭代中含有模 2^{32} 加、异或以及用 S 盒实现的替代。

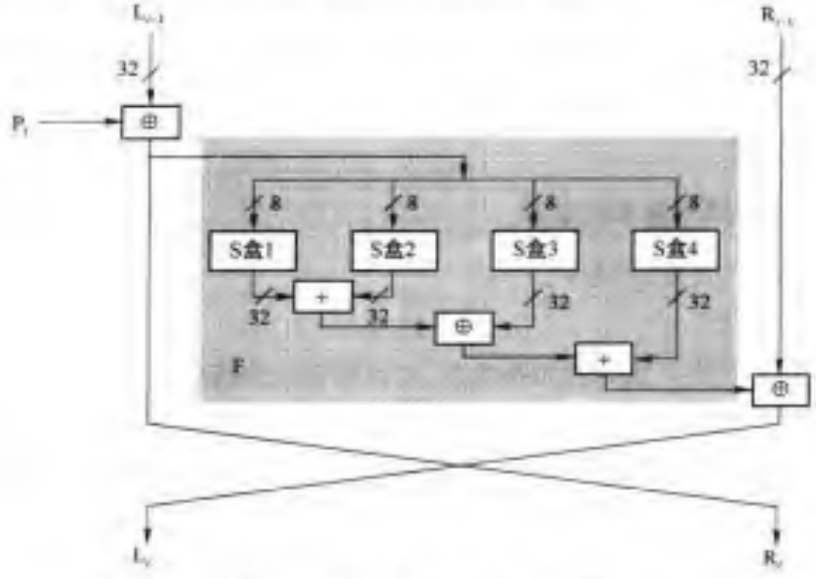


图 6.4 Blowfish 算法的一轮迭代

解密如图6.3(b)所示，很容易从加密算法推导出来。这里，64位的密文分成 LD_0 和 RD_0，我们用 LD_i 和 RD_i 来表示第 i 轮迭代后输出的左半部分和右半部分。像大多数分组密码那样，Blowfish 算法的解密过程也是把子密钥倒过来使用。不过，和大多数分组密码不同的是，Blowfish算法解密的顺序和加密是一样的，而不是倒过来的。算法可以表示为下面这段程序：

```
for i = 1 to 16 do
        RD_i = LD_(i-1) ⊕ P_(16-i+1);
        LD_i = F[RD_i] ⊕ RD_(i-1);
LD_17 = RD_16 ⊕ P_1;
RD_17 = LD_16 ⊕ P_2;
```

6.2.3 讨论

Blowfish算法也许是本书讨论的传统加密算法中最难以对付的一个。跟DES不同，Blowfish算法的S盒是跟密钥有关的，不仅如此，其他的一些算法，如RC5，每轮迭代执行的函数是与数据相关的，而Blowfish算法的子密钥和S盒都是用Blowfish算法本身生成的。这使得数据完全不可辩认，对它的密钥分析也异常困难。到现在为止，对Blowfish算法进行分析的文章发表了不少，但还没有找出它的任何实际弱点。

Blowfish 算法设计的另一个有趣方面是：每轮运算都是对数据的左右两部分同时执行，而不是像古典 Feistel 密码那样，每轮迭代只对数据的一半进行运算。即使附加的运算(XOR)是线性的，密码的强度还是因此增加不少。文献[HEYS95]也提到，在替代－置换网络(SPN)中的每一轮迭代包含有线性变换将增加分组密码的雪崩效应。

Blowfish 算法完全不担心受到穷举攻击，因为它的密钥长度可以达到448位。

Blowfish 算法执行起来也是非常快的，Schneier 总结出表 6.1，比较了在奔腾机上不同算法执行的时间。其中，Blowfish 算法明显是最快的。

表 6.1 几种分组密码在奔腾机上的速度

算法	每轮所用的时钟周期数	轮数	加密一字节所用的时钟周期数
Blowfish 算法	9	16	18
RC5	12	16	23
DES	18	16	45
IDEA	50	8	50
3DES	18	48	108

在文献[SCHN93]中，Schneier 给出了设计 Blowfish 算法时所考虑的一些因素。我们择其重要的摘抄如下：

1. 用穷举攻击的难度不仅仅来自于密钥的长度，因为子密钥的产生过程非常耗时。要测试一个密钥，实际上一共要执行 522 次加密算法。
2. 函数 F 作为一个 Feistel 网络给 Blowfish 算法带来了很好的雪崩效应。在第 i 轮，L_{i-1} 的每位会影响 R_{i-1} 的每位。此外，每个子密钥位会受每个密钥位的影响，因此，在每轮结

束后，函数 F 在密钥(P_i)和数据的右半部分(R_i)间会有很好的雪崩效应。

3. F 的每一位输入仅用做一个 S 盒的一位输入。而在 DES 中，有很多位用做两个 S 盒的输入，并以此大大加强了算法抗差分分析的能力。Schneier 觉得使用了与密钥相关的 S 盒之后，增加这种复杂性没有必要。
4. 与 CAST 算法不同，Blowfish 算法的函数 F 与第几轮没有什么关系。Schneier 觉得将 F 设计成与轮数相关没有什么密码学意义，因为 P 的替换已经与轮数相关了。

6.3 RC5 算法

RC5 是 Ron Rivest 设计的一种对称加密算法(参考文献[RIVE94, RIVE95])。RC5 具有以下特点：

- **适于软件和硬件实现**：RC5 算法只用了一些微处理器上常见的基本运算。
- **快速**：为了加快运算速度，RC5 设计成面向字的简单算法。基本运算是对整个字同时执行的。
- **可用于字长不同的处理器**：RC5 中的字长是算法的参数，不同的字长对应不同的算法。
- **迭代次数可变**：迭代次数也是 RC5 的参数，这样使用者可在高速与高安全性两方面做出权衡。
- **密钥长度可变**：密钥长度也是 RC5 的参数，它同样影响到速度和安全性。
- **简单**：RC5 的结构简单，易于实现，易于确定算法强度。
- **对存储量要求低**：这一点使得 RC5 可以在智能卡及其他存储容量有限的设备上使用。
- **安全性高**：RC5 可以通过设置参数来提供高度安全。
- **与数据相关的循环**：RC5 采用了与数据相关的循环(循环移位)，这明显加强了算法抗分析的强度。

RC5 已被 RSA 数据安全公司采纳，在他们的一些主要产品中使用了 RC5，如 BSAFE、JSAFE 和 S/MAIL。

6.3.1 RC5 的参数

RC5 实际上是由 3 个参数确定的一个加密算法族。

参　数	定　义	允许的值
w	字长以位为单位，RC5 的分组长度为 2 个字	16,32,64
r	迭代次数	0,1,…,255
b	密钥 K 的 8 位字节数十进制	0,1,…,255

所以，RC5 的分组长度可为 32、64 和 128，密钥长度则可以从 0 位到 2040 位。RC5 的参数确定之后记为 RC5 – $w/r/b$。例如，RC5 – 32/12/16 的字长为 32 位，分组长度为 64 位，加解密算法中的迭代次数为 12，密钥长度是 16 字节(128 位)。Rivest 建议的版本是 RC5 – 32/12/16。

6.3.2 密钥扩展

RC5对密钥执行一系列的复杂运算，以产生总共 t 个子密钥。每一轮迭代要用2个子密钥，另外还需要2个子密钥在迭代之外使用，所以 $t = 2r + 2$。每个子密钥的长度都是1个字（w 位）。

图6.5给出了产生子密钥的方法。子密钥表示为S[0],S[1],…,S[$t-1$]，这是一个 t 个字的数组。根据参数 r 和 w，将这个数组初始化成固定的伪随机数。然后将 b 字节的密钥K[0…$b-1$]转换成一个 c 个字的数组L[0…$c-1$]。在低端结构上可以这样实现：把数组L初始化为全0，然后把串K直接赋值给L。如果 b 不是 w 的整数倍，那么L最后一个单元的一部分仍是0。最后，执行一个复杂运算，根据L和初始化的S来得到最终的S。

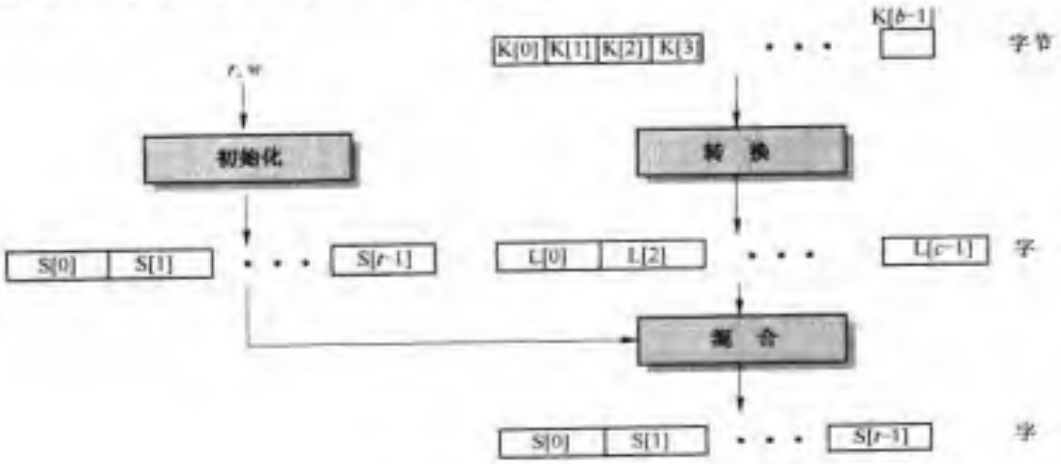


图6.5 RC5算法的密钥扩展

初始化操作使用了下面这两个1字长的常量：

$$P_w = \text{Odd}[(e - 2)2^w]$$
$$Q_w = \text{Odd}[(\phi - 1)2^w]$$

其中

$$e = 2.718281828459\ldots\text{(自然对数的底)}$$
$$\phi = 1.618033988749\ldots\text{(黄金分割比)} = \left(\frac{1 + \sqrt{5}}{2}\right)$$

Odd[x]表示与 x 最接近的奇数。比如，Odd[e] = 3，Odd[Φ] = 1。对于 w 的可能取值，这两个常数如下（十六进制）：

w	16	32	64
P_w	B7E1	B7E15163	B7E151628AED2A6B
Q_w	9E37	9E3779B9	9E3779B97F4A7C15

使用这两个常数将S如下初始化：

```
S[0] = P_w;
for i = 1 to t - 1 do
        S[i] = S[i - 1] + Q_w;
```

其中的加法是模 2^w 操作。密钥数组 L 与初始数组 S 随后将混合，以产生最终的子密钥数组 S。在进行混合时，对较大的数组要进行三轮操作，对较小的数组则可能操作更多次：

```
i = j = X = Y = 0
do 3 × max(t, c) times:
    S[i] = (S[i] + X + Y) <<< 3; X = S[i]; i = (i + 1) mod (t);
    L[j] = (L[j] + X + Y) <<< (X + Y); Y = L[j]; j = (j + 1) mod (c);
```

Rivest 在文献[RIVE94]中指出这个密钥扩展函数具有一定的单向性：从 S 推出 K 不太容易。

6.3.3 加密

RC5 使用了 3 种基本运算(及它们的逆)：

- **加法**：字的模 2^w 加法。记为 +。逆运算是模 2^w 减法，记为 −。
- **按位异或**：记为"⊕"。
- **循环左移**：字 x 循环左移 y 位记为 $x <<< y$。其逆运算是循环右移，记为 $x >>> y$。

图 6.6(a)描述了加密的过程。请注意，这并非一个古典 Feistel 密码。设开始时明文存在两个 w 位的寄存器 A 和 B 中，我们用变量 LE_i 和 RE_i 来表示第 i 轮迭代完成之后数据的左半部分和右半部分。算法可以用下面的伪代码来描述：

```
LE_0 = A + S[0];
RE_0 = B + S[1];
for i = 1 to r do
        LE_i = ((LE_{i-1} ⊕ RE_{i-1}) <<< RE_{i-1}) + S[2 × i];
        RE_i = ((RE_{i-1} ⊕ LE_i) <<< LE_i) + S[2 × i + 1];
```

得出的密文即变量 LE_r 和 RE_r。每轮迭代 r 都对数据的所有字进行了替代和置换。注意，运算过程异常简洁，只用 5 行代码就可以定义。同时应注意到每轮迭代都对数据的左右两半部分进行了更新，所以 RC5 的一轮迭代几乎相当于 DES 的两轮迭代。

6.3.4 解密

解密过程[见图 6.6(b)]很容易由加密过程推导出来。这里，$2w$ 位密文开始时指定为两个字变量 LD_r 和 RD_r，我们用 LD_i 和 RD_i 来表示第 i 轮迭代开始之前数据的左半部分和右半部分。这里的迭代轮数编号为从 r 到 1。

```
for i = r downto 1 do
        RD_{i-1} = ((RD_i - S[2 × i + 1] >>> LD_i) ⊕ LD_i);
        LD_{i-1} = ((LD_i - S[2 × i] >>> RD_{i-1}) ⊕ RD_{i-1});
B = RD_0 - S[1];
A = LD_0 - S[0];
```

RC5 的两个最显著的特点是，算法简单且移位与数据相关。移位是算法中惟一的非线性部分。Rivest 认为，正是这个特点使得对算法进行线性分析和差分分析都很困难。许多研究都证实了这一点(参阅文献[YIN97])。

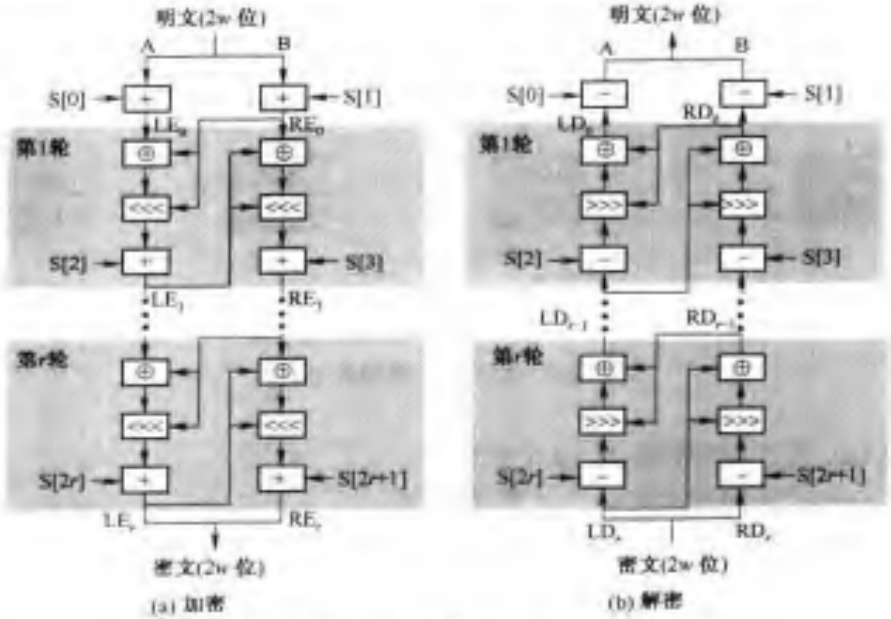


图 6.6 RC5 算法的加密与解密

6.3.5 RC5 的运行模式

为了增强 RC5 的实现效率，RFC 2040 定义了 4 种不同的工作模式：

- **RC5 分组密码**：算法取固定长度的明文块（$2w$ 位），再用同一个密钥所产生的变换来产生相同长度的密文块。即 ECB 模式（参见图 3.11）。
- **RC5-CBC**：这是 RC5 的密码分组链接模式。第 3 章中已经讨论了 CBC 模式（参见图 3.12）。CBC 模式下，处理的消息长度为 RC5 分组长度的整数倍（长度 $2w$ 位）。第 3 章中已经提到，CBC 模式比 ECB 模式的安全性要好，因为用同样的明文可以得到不同的密文块。
- **RC5-CBC-Pad**：属 CBC 模式，它可以处理任意长的明文。密文最多比明文长 RC5 的分组长度。
- **RC5-CTS**：属 CBC 模式，称为密文挪用模式，它处理任意长度的明文并得到等长的密文。

后两种这里需要详细讨论一下。

用 CBC 模式来加密消息的话，需要采取一些方法来处理不为分组长度整数倍长的消息。最简单的方法是使用填充。RC5 假设消息长度为整数个字节，在消息的末尾，需要填充 1 到 bb 个字节，bb 就是 RC5 的分组长度，按字节计就是 $2w/8$。填充的字节值相同，等于填充字节的个数。比如填充 8 个字节，则每个字节就是 00001000。这种模式就是 RC5-CBC-Pad 模式。

用这种办法有些时候不太合适。例如，要把密文存放在和原始明文相同的区域里，这样，就要求密文与原始明文等长，用 RC5-CTS 模式就可以做到这一点（见图 6.7）。设明文的最后一块只有 L 字节。其中 $L<2w/8$。加密过程如下所示：

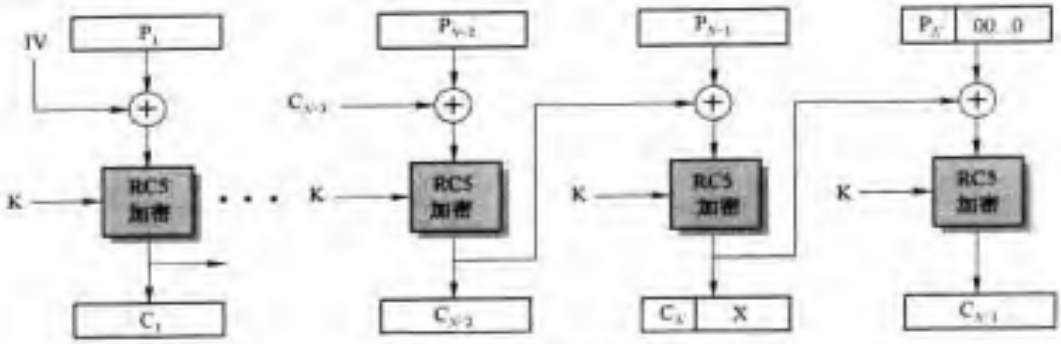


图 6.7 RC5 密文挪用模式

1. 用传统 CBC 模式加密前面 $N-2$ 个分组。
2. 将 P_{N-1} 与刚产生的密文块 C_{N-2} 异或，得到 Y_{N-1}。
3. 加密 Y_{N-1} 得到 E_{N-1}。
4. 以 E_{N-1} 的前面 L 个字节为 C_N。
5. P_N 在后面补 0 后与 E_{N-1} 异或，得到 Y_N。
6. 将 Y_N 加密，得到 C_{N-1}。

最后两块密文是 C_{N-1} 和 C_N。

6.4 高级对称分组密码的特点

实际上，现在所有的对称分组密码在很多方面与 DES 和基本的 Feistel 分组密码结构相似。这确凿地证明了 IBM 和 NSA 的 DES 设计者们的功绩。然而，密码学在发展，快速软件加密的需求已经显现，一些高级的对称密码应运而生。本章旨在阐明这些最重要的现代对称密码算法的先进性。本节，我们会强调这些算法中为 DES 所没有的关键特征。

- **密钥长度可变**：倘若对某个加密算法进行密码分析极端困难，那么该算法的强度就取决于其密钥长度了。密钥越长，越能抗穷举攻击。Blowfish 算法和 RC5 中的密钥长度都可变，且允许很长。
- **复合运算**：使用多个算术和布尔运算使密码分析复杂化，尤其是这些运算不满足分配律和结合律的时候。这种方法提供的非线性可以替代 S 盒。本章中除了 3DES 以外的所有算法都使用了复合运算。
- **与数据相关的循环移位**：另外一种可替代 S 盒的有趣方法是与数据相关的循环移位。只要有充分的迭代轮数，这种方法就可提供很好的混淆性和扩散性。此外，循环移位还与进入这一轮迭代的数据有关，而非子密钥，这使得恢复出子密钥更为困难。RC5 就使用了与数据相关的循环移位。
- **与密钥相关的 S 盒**：不像 DES 或 CAST-128 那样，试图设计出一个有着令人满意的密码特征的固定 S 盒，而是使用其内容随密钥变化而变化的 S 盒。这种方法，尤其是对于较大的 S 盒（如 8×32），可以获得高度的非线性，使得密码分析更为困难。Blowfish 算法就

使用了与密钥相关的S盒。

- **复杂的密钥生成算法**：这是Blowfish算法所使用的一种独特的策略。子密钥的生成甚至比一次加密或解密要花长得多的时间。这样就使穷举攻击的代价增加不少。
- **可变的分组长度**：分组长度越大，密码分析越困难。并且，由于分组长度可变，使得算法在使用上更灵活，可适应不同的应用。RC5即是如此。
- **迭代次数可变**：同样，增加迭代次数亦可增加密码强度。当然，这要花去更多时间。迭代次数可变允许用户在安全性和执行速度两方面进行折衷。RC5就是这样。
- **每一轮迭代对数据的左右两半部分同时运算**：古典Feistel密码的每一轮迭代只对数据的一半进行运算。如果另外一半进行很简单的运算，这只会多花一点点时间，但增加了安全性。Blowfish算法和RC5都采用了这种策略。
- **函数F可变**：每轮迭代使用不同的F，以使密码分析更为困难。
- **与密钥相关的循环移位**：可以采用不与数据相关而与密钥相关的循环移位。

6.5 RC4流密码

本节我们讨论一种可能被广泛推广的对称流密码——RC4。首先介绍流密码的整体结构，然后探讨RC4。

6.5.1 流密码的结构

一个典型的流密码每次加密一个字节的明文，当然流密码也可被设计为每次操作一比特或者大于一个字节的单元。图6.8给出了一个典型的流密码的结构图。在该结构中，密钥输入到一个伪随机数(比特)发生器，该伪随机数发生器产生一串随机的8比特数。在第7章中，我们将详细讨论伪随机数发生器。简单地说，一个伪随机流就是在不知道输入密钥的情况下不可预知的流。发生器的输出称为密钥流，通过与同一时刻一个字节的明文流进行异或(XOR)操作产生密文流。例如，如果发生器产生的下一字节为01101100，而下一明文字节为11001100，则得出的密文字节为：

```
  11001100    明文
⊕ 01101100    密钥流
  10100000    密文
```

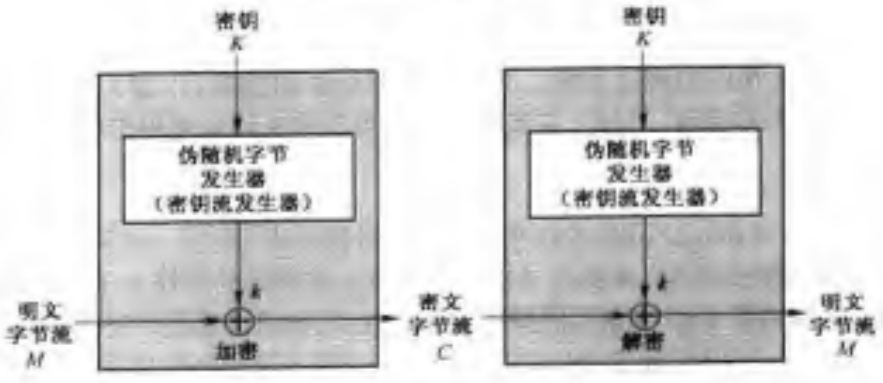


图6.8 流密码结构图

解密需要使用相同的伪随机序列：

```
  10100000   明文
⊕ 01101100   密钥流
  11001100   密文
```

流密码类似于第 2 章讨论的“一次一密”。不同的是，“一次一密”使用的是真正的随机数流，而流密码使用的是伪随机数流。

[KUMA97]列出了设计流密码需要考虑的主要因素：

1. 加密序列的周期要长。伪随机数发生器实质上使用的是产生确定的比特流的函数，该比特流最终将出现重复。重复的周期越长，密码分析的难度就越大。这与讨论维吉尼亚密码的考虑从本质上是一致的，即密钥越长密码分析越困难。
2. 密钥流应该尽可能地接近于一个真正的随机数流的特征。例如，1 和 0 的个数应近似相等。若密钥流为字节流，则所有 256 种可能的字节值出现的频率应近似相等。密钥流的随机特性越好，则密文越随机，密码分析就越困难。
3. 注意图 6.8 中伪随机数发生器的输出取决于输入密钥的值。为了防止穷举攻击，密钥应该足够长，对于分组密码也要有同样的考虑。因此，从目前的软硬件技术发展来看，至少应当保证密钥长度不小于 128 位。

通过设计合适的伪随机数发生器，流密码可以提供和相应密钥长度分组密码相当的安全性。流密码的主要优点是，其相当于分组密码来说，往往速度更快而且需要编写的代码更少。例如本节介绍的 RC4，仅仅数行代码就可实现。根据[RESC01]给出的数据，表 6.2 列出了 RC4 与三种常见的对称分组密码执行速度的对比。分组密码的优点是可以重复使用密钥，然而，如果用流密码对两个明文加密时使用相同的密钥，则密码分析就会相当容易[DAWS96]。如果对两个密文流进行异或，得出的结果就是两个原始明文的异或。如果明文仅仅是文本串、信用卡号或者其他已知特征的字节流，则密码分析极易成功。

表 6.2 奔腾 II 上对称密码的速度对比

密码	密钥长度	速度(Mbps)
DES	56	9
3DES	168	3
RC2	可变	0.9
RC4	可变	45

对于需要对数据流进行加密解密的应用，比如通过一个数据通信信道或者网页浏览器/Web 链接，流密码就是很好的解决方案。而对于处理成块的数据，比如文件传输，电子邮件和数据库，分组密码则更为适用。当然，在实际中两种类型的密码都可用于各种应用。

6.5.2 RC4 算法

RC4 是Ron Rivest为 RSA 公司在 1987 年设计的一种流密码。它是一个可变密钥长度、面向字节操作的流密码。该算法以随机置换为基础。分析显示该密码的周期大于 10^{100}[ROBS95a]。每输出一字节的结果仅需要 8 到 16 条机器操作指令。RC4可能是应用最广泛的流密码。它被用于 SSL/TLS(安全套接字协议/传输层安全协议)标准，该标准是为网络浏览器和服务器间通信而制定的。它也应用于作为IEEE 802.11无线局域网标准一部分的WEP(Wired Equivalent Privacy)协议。

当时，RC4作为RSA公司的商业机密并没有公开。直到 1994 年 9 月，RC4 算法才通过Cypherpunks匿名邮件列表匿名地公开于Internet上。

RC4 算法非常简单，易于描述：用从 1 到 256 个字节(8 到 2048 位)的可变长度密钥初始化一个 256 个字节的状态矢量 S，S 的元素记为 S[0]，S[1]，…，S[255]，从始至终置换后的 S 包含从 0 到 255 的所有 8 比特数。对于加密和解密，字节 k(图 6.8)由 S 中 255 个元素按一定方式选出一个元素而生成。每生成一个 k 的值，S 中的元素就被重新置换一次。

初始化 S

开始时，S 中元素的值被置为按升序从 0 到 255，即 S[0] = 0，S[1] = 1，…，S[255] = 255。同时建立一个临时矢量 T。如果密钥 K 的长度为 256 字节，则将 K 赋给 T。否则，若密钥长度为 keylen 字节，则将 K 的值赋给 T 的前 keylen 个元素，并循环重复用 K 的值赋给 T 剩下的元素，直到 T 的所有元素都被赋值。这些预操作可概括如下：

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

然后用 T 产生 S 的初始置换。从 S[0]到 S[255]，对每个 S[i]，根据由 T[i]确定的方案，将 S[i]置换为 S 中的另一字节：

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

因为对 S 的操作仅是交换，所以惟一的改变就是置换。S 仍然包含所有值为 0 到 255 的元素。

密钥流的生成

矢量 S 一旦完成初始化，输入密钥就不再被使用。密钥流的生成是从 S[0]到 S[255]，对每个 S[i]，根据当前 S 的值，将 S[i]与 S 中的另一字节置换。当 S[255]完成置换后，操作继续重复，从 S[0]开始：

```
/* Stream Generation */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

加密中，将 k 的值与下一明文字节异或；解密中，将 k 的值与下一密文字节异或。

图 6.9 总结了 RC4 的逻辑结构。

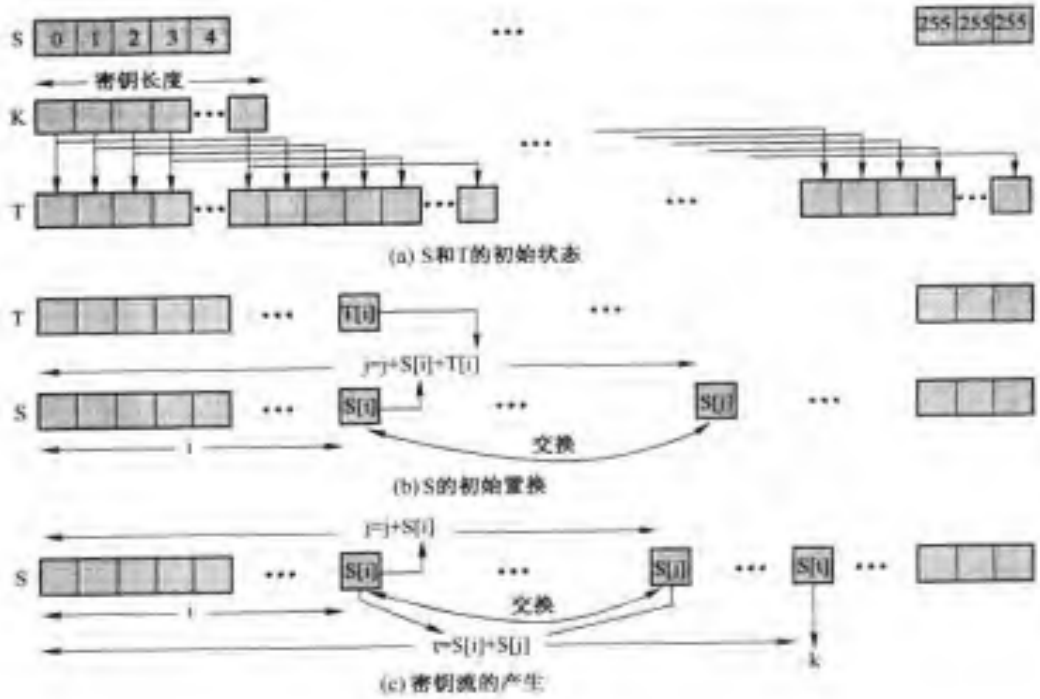


图 6.9 RC4

RC4 的强度

关于分析RC4的攻击方法有许多公开发表的文献(如[KNUD98]、[MIST98]、[FLUH00]、[MANT01])。但没有哪种方法对于攻击足够长度密钥(如 128 位)的 RC4 有效。值得注意的是[FLUH01]中的报告,作者指出用于为 802.11 无线局域网络提供机密性的 WEP 协议,易于受到一种特殊的攻击方法的攻击。从本质上讲,这个问题并不在于 RC4 本身,而是作为 RC4 中输入的密钥的产生途径有漏洞。这种特殊的攻击方法不适用于其他使用 RC4 的应用,通过修改 WEP 中密钥产生的途径也可以避免这种攻击。这个问题恰恰说明了设计一个安全系统的困难性不仅包括密码编码函数,还包括协议如何正确地使用这些密码编码函数。

6.6 推荐读物和网址

[SCHN96]提供了许多对称分组密码及一些流密码的详细介绍。[ROBS95b]中关于对称分组密码设计的论述很引人入胜。

[KUMA97]对于流密码的设计准则提出了很好的意见。另一个更数学化的相关读物是[RUEP92]。[ROBS95a]给出了关于流密码设计的论述。

KUMA97 Kumar, I. *Cryptology*. Laguna Hills, CA: Aegean Park Press, 1997.

ROBS95a Robshaw, M. *Stream Ciphers*. RSA Laboratories Technical Report TR-701, July 1995. http://www.rsasecurity.com/rsalabs/index.html

ROBS95b Robshaw, M. *Block Ciphers*. RSA Laboratories Technical Report TR-601,

August 1995. http://www.rsasecurity.com/rsalabs/index.html

RUEP92 Rueppel, T. "Stream Ciphers." In [SIMM92].

SCHN96 Schneier, B. *Applied Cryptography*. New York: Wiley, 1996.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.

推荐网址：

- **Block Cipher Lounge**：当前分组密码的概览，可链接到许多相关的论文。

6.7 关键术语、思考题和习题

6.7.1 关键术语

Blowfish	RC4	三重DES(3DES)
中间相遇攻击	RC5	

6.7.2 思考题

6.1 什么是三重加密？

6.2 什么是中间相遇攻击？

6.3 在三重加密中用到多少个密钥？

6.4 为什么3DES的中间部分采用了解密而不是加密？

6.5 Blowfish的密钥长度是多少？

6.6 Blowfish中用到的初始操作是什么？

6.7 RC5中用到了哪些普通的算术运算？

6.8 RC5中用到的初始操作是什么？

6.9 请列出设计流密码要考虑的重要因素。

6.10 为什么流密码的密钥不能重复使用？

6.11 RC4中用到的初始操作是什么？

6.7.3 习题

6.1 假设你想做一个用CBC模式进行分组加密的硬件设备，要求算法强度比DES强。3DES是一个很好的候选算法，但它只在ECB模式中定义。现在还没有标准将3DES用于CBC模式，图6.10给出了两种方案，使用的都是CBC模式。你将选择哪一个？

a. 从安全性角度考虑。

b. 从性能上考虑。

6.2 仅使用3DES芯片和一些异或函数，你能对图6.10中给出的两种方案进行安全性改进吗？假设仍旧使用两个密钥。

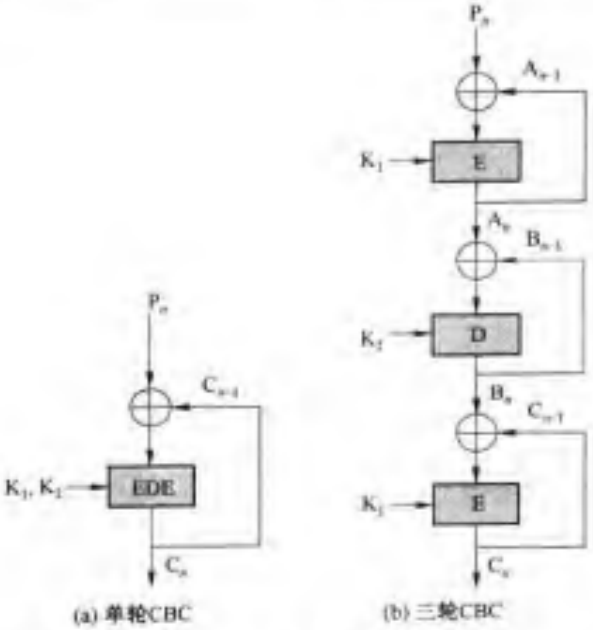


图 6.10 将 3DES 用于 CBC 模式

6.3 对 3DES 的 Merkle-Hellman 攻击是这样的：首先令 A = 0[见图 6.1(b)]，然后，对 K_1 的所有 2^{56} 个取值，找出令 A = 0 的明文 P。请描述算法的剩下部分。

6.4 证明 Blowfish 算法的解密是加密的逆。

6.5 证明 RC5 算法的解密是加密的逆。

6.6 在 RC5 的 CBC 填充模式中，可以填充 1 个到 bb 个字节。为什么不允许填充 0 个字节？也就是说，若消息正好是分组长度的整数倍，为什么还要填充？

6.7 图 6.11 给出了当明文不是分组长度整数倍时，使产生的密文与明文等长的RC5-CTS的一种替代方案。

a. 对算法进行解释。

b. 解释为什么 RC5-CTS 比图 6.10 所示的方案要好。

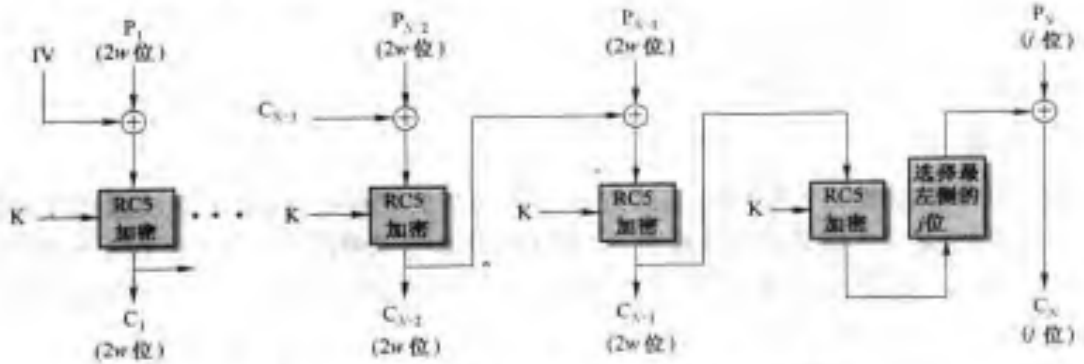


图 6.11 用 RC5 的 CBC 模式加密最后的短块示意图

6.8　说明 RC5-CTS 模式如何解密 C_{n-1} 和 C_n。

6.9　在任意时刻，RC4 流发生器输出的下一字节取决于发生器的状态，该状态依次由 i 和 j 在算法的当前值以及 S 的当前置换决定。对于发生器有多少种可能的状态？

6.10　RC4 的密钥值是什么的时候，使得 S 在初始化过程中没有变化？即在对 S 进行初始置换后，S 的元素的值按升序分别等于 0 到 255。

第 7 章　用对称密码实现保密性

历史上，密码学的主要作用是用传统密码来实现保密性。只是最近几十年才出现了其他应用领域，如认证、完整性、数字签名以及公钥密码，这些都成了密码学的理论和应用内容。

在讨论这些之前，本章先是主要探讨一下用传统密码来实现保密性，这一直都是一个重要方面。此外，充分掌握这里面的原理有助于发展公钥密码和理解牵涉到加密的其他问题，例如认证。

我们先讨论一下密码所处的逻辑位置，这里主要是指在链路加密和端对端加密之间做出选择。接下来看看怎么用密码来对付传输过程中所遇到的分析和攻击。然后讨论一个难题：密钥分配。最后，我们阐述一下实现保密性过程中要使用的一个重要工具——随机数产生的基本原理。

7.1　密码功能的设置

如果要用加密来对抗危害保密性的攻击，我们就必须确定加密功能在系统中的设置位置。本节，我们先看一看通常系统中的安全隐患，然后给出加密功能设置的两种方法：链路加密和端对端加密。

7.1.1　安全隐患

作为一个例子，考虑典型商业组织的一个用户工作站。图 7.1 给出了它所用的通信设施。我们可以找一找它的安全性弱点。

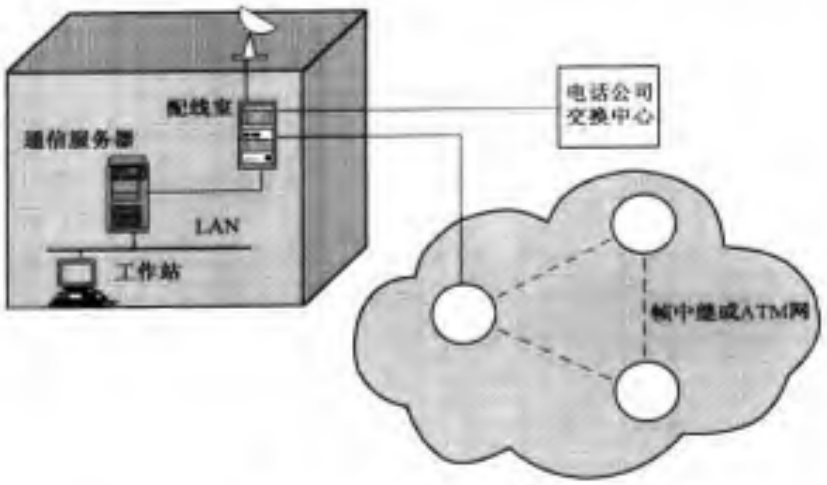


图 7.1　数据安全隐患示意图

在大部分组织中，工作站都连到局域网上。用户通常可以访问到直接连到同一个局域网

上或通过网桥或路由器互连在同一栋建筑中其他局域网上的其他工作站、主机和服务器。那么这里就有第一个易受攻击的地方，在这里的主要担心是被其他雇员窃听。通常局域网是一个广播网络：从任意一个站点到任意另一个站点的传输内容在局域网媒体上对所有站点都是可见的。数据是以帧的形式传输的，每一帧中都包含源地址和目的地址。偷听者可以监视局域网上的通信量，并依据源地址和目的地址获取想要的任何通信量。

更有甚者，偷听者并不一定必须是在这栋建筑物内工作的雇员。如果局域网通过一个通信服务器或局域网上的主机提供了拨号接入能力，那么外部侵入者就可能访问这个局域网并监视通信量。

访问局域网外面的站点几乎总是可以通过一个路由器、一组拨出调制解调器或某种通信服务器来进行。从通信服务器到配线室有一条线路，配线室为互连内部数据和电话线提供了一个接插板，并为外部通信提供了一个中转接点。

配线室本身就易受攻击。入侵者潜入配线室就可侦听任何数据传输线路，他可以使用有线或无线的办法，将有用数据发送到附近别的地方。

具体说来，将消息从配线室发送出去有以下途径：标准配置一般提供有对最近的电话公司交换中心的访问设备。配线室的数据线汇成电缆，通往大楼的地基，从那里再通往电话公司交换中心。

此外，不管是卫星链接的地面工作站，还是区域内点对点的微波链接，其配线室都提供微波天线。天线是内部网的一部分，或者用于搭上远距离传输媒体。

配线室还可提供与分组交换网络的连接。这个连接可以是一条租用线，或是一条直通专用线，也可以是诸如 ISDN 的公众远程通信网的交换连接。在网络内部，数据经过一些节点和节点之间的链路到达目的节点。

攻击可以在上述任何通信链路进行。对于主动攻击，攻击者首先需要获得对一部分链路的控制以便侦听和介入，而对于被动攻击，攻击者只要能观察到数据传输就够了。通信链路可以是电缆（双绞线、同轴电缆、光纤）、微波链路或卫星频道。对于双绞线和同轴电缆，可以通过非法侵入或者用感应设备来监视电磁辐射的方法来攻击，主动攻击和被动攻击都可以使用非法侵入。而用感应设备来监视则只是被动攻击，对于光纤这种较先进的媒体，还没有一种有效的窃听办法。光纤不产生电磁辐射，因此不易被感应侦听。要是把光缆给割破了，那将会至少降低信号质量，因此很容易被发现，相反，攻击者侦听微波和卫星传输的信息就冒不了多少险。特别是卫星传输，它覆盖的地理面积太大了。想在微波和卫星信道上做手脚也是可能的，只是技术难度很大，且花费很昂贵。

除了不同通信线路易受攻击之外，处理数据的软件和硬件也可能受到攻击。攻击者可以试图修改软件和硬件，以访问到数据存储器或监视到电磁辐射。这种攻击方法不像无链路攻击那么常见，但的确存在。

所以，可以说，攻击者无处不在。此外，对于广域网，许多易受攻击的地方不为终端用户所控制。即使是局域网，不安全的因素也有很多。

7.1.2　链路加密与端对端加密

要解决上述安全问题，加密是最强有力且最常见的方法。我们现在要搞清楚的是：加密什么？在什么地方可以用加密？图 7.2 给出了两种基本选择：链路加密与端对端加密。

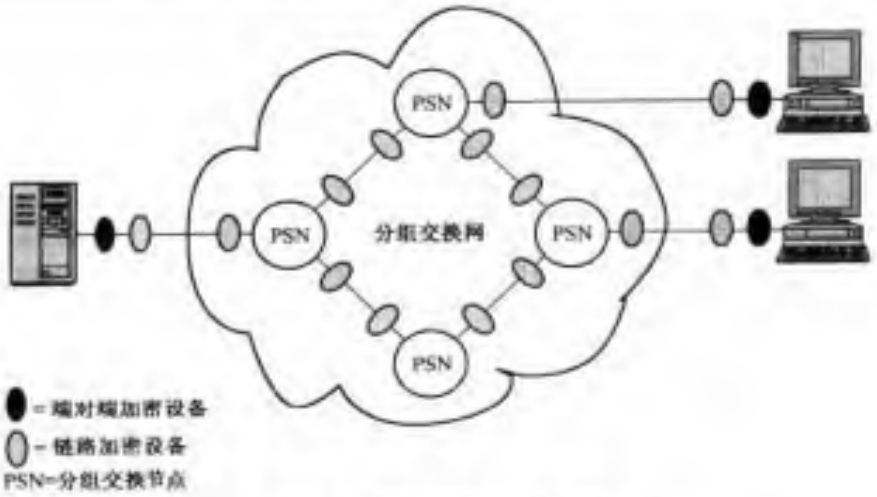


图 7.2 分组交换网上的加密

基本方法

使用链路加密，是在通信链路两端加上加密设备。这样，链路上的信息传输可以是安全的。尽管这种办法对于一个很大的网络来说需要很多加密设备，但其作用是显而易见的。它的缺点是：每次分组交换都需要将消息解密。因为交换时要用到数据包头地址，以便寻径。所以在每次分组交换时，消息是易受攻击的。如果是公众分组交换网，用户对节点的安全性不能控制。

要让这个策略有效，所有潜在链路都要使用链路加密。共享一条链路的每对节点应共享惟一的密钥。这样，整个网络上密钥的数目就会很大。

端对端加密的过程在两端系统中进行。由源主机或终端加密数据。密文经由网络传送到目的主机或终端。目的主机与源主机共享一个密钥以便解密。这种办法看起来可以使网络连接及交换的信息传输安全，所以，端对端加密能让用户对网络信息传输的安全大为宽心。然而，遗憾的是，这种方法依然存在弱点。

看看这种情况：一主机通过 X.25 分组交换网与另一主机建立起虚拟信道，并准备用端对端加密发送数据到第三台主机。数据包含有信息头和一些用户数据。主机应加密每个数据包的哪个部分呢？对整个包加密（当然包括信息头），那是不行的，因为只有第三台主机才能解密。而分组交换节点接收到密文包后不能读信息头，这样就不能实现包的路由。现在假设只对用户数据进行加密，信息头保持明文，这样，用户数据安全了，但是传输过程又不安全了，所以端对端加密需提供认证。如果两个末端系统共享一个加密密钥，那么接收者可以确信接收到的任何消息是从合法发送者处来的。链路加密机制当然不含这种认证的。

为了提高安全性，应该同时使用链路加密和端对端加密，见图 7.2。这时，主机使用端对端加密密钥来加密用户数据，整个分组则使用链路加密。分组在网络中穿梭时，每个节点用链路加密密钥来解密它，读取信息头，然后再对它加密，发送到下一条链路上。这时，除了在分组交换节点的存储器逗留的时间里信息头是明文以外，整个分组一直都是安全的。

表 7.1 对两种加密策略进行了总结。

表 7.1　链路加密和端对端加密的特点[PFLE97]

链路加密	端对端加密
末端系统和中间系统的安全性	
消息在发送主机时为明文	消息在发送主机时为密文
消息在中间节点时为明文	消息在中间节点时为密文
用户的角色	
与发送主机交互	与发送过程交互
加密过程对用户透明	用户自己使用加密
主机含加密设备	用户决定加密算法
所有用户使用一个加密设备	用户选择加密体制
可以硬件实现	软件实现
所有消息被加密或都不被加密	用户决定每条消息是否加密
实现时要注意的	
每个中间主机节点和每两个中间主机节点需一个密钥	每对用户需一个密钥
提供用户认证	提供用户认证

端对端加密函数的逻辑位置

链路加密的加密函数在整个网络中处于较低的层次，在开放式系统互联(OSI)模型中，它处在物理层或链路层。而端对端加密的加密函数的逻辑位置可有几种选择。应用级别较低的话，它可以用在网络层。例如，附着在 X.25 协议上，这样，所有 X.25 数据包的用户数据都将被加密。

对于网络层加密，可识别并被保护的实体数与网络的末端系统数相对应。任何一个末端系统都可以进行加密数据的交换，只需与对方末端系统共享一个密钥。该末端系统所有的用户进程和应用都使用相同的密钥和加密机制来与某目标末端系统联系。对于这种安全保护，可以使用某种带有加密函数的前端处理器(FEP)来实现，通常是末端系统的网卡。

图 7.3 是一个示意图，在主机这边，FEP 接收数据包，数据包的用户数据部分被加密了，而信息头则没有，处理过后被发送到网络上。而另一方，数据到达之后，用户数据部分被解密，再将整个包发送给主机。如果传输层的功能(如 TCP)在前端实现，那么传输层的数据头也应该是明文，而传输层协议的用户数据部分应被加密。

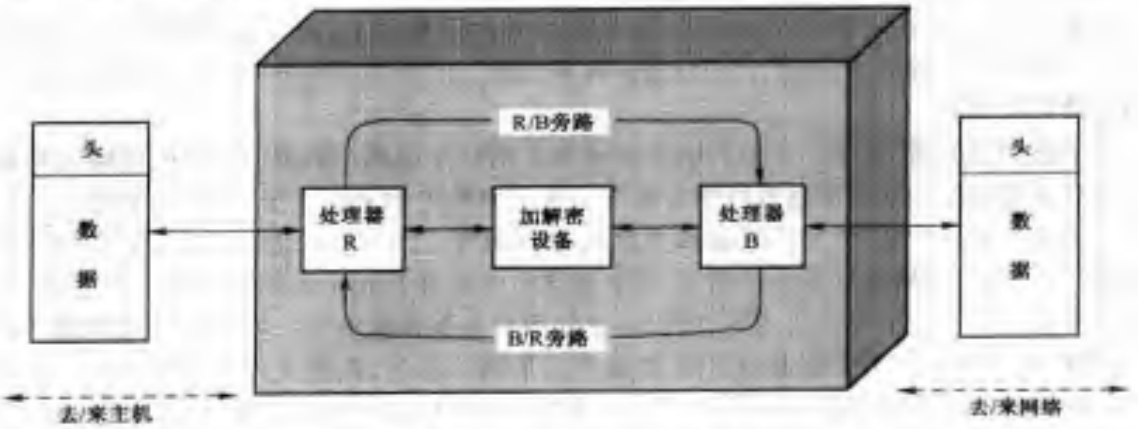


图 7.3　前端处理器(FEP)的功能

将加密设备用于端对端协议，比如网络层帧中继或 TCP 协议，可以提供整个网络的端对端传输的安全性。但是，这种方案不能用于网络之间的服务，比如电子邮件、电子数据交换(EDI)以及文件传输等。

图 7.4 说明了这个问题。图中，用一个电子邮件网关来进行一个基于 OSI 协议的内部网和一个基于 TCP/IP 体系的内部网之间的连接。在应用层上没有端对端协议。这两个内部网的末端节点的传输层和网络层传输到邮件网关上就终止了，由邮件网关建立起新的连接。此外，这样的情况不仅会在两个基于不同体系结构的网络之间的网关上出现，即使它们都使用 TCP/IP 或是都使用 OSI，也会有类似的问题。所以，对于像具有存储转发功能的电子邮件这样的应用，获得端对端加密只能在应用层上进行。

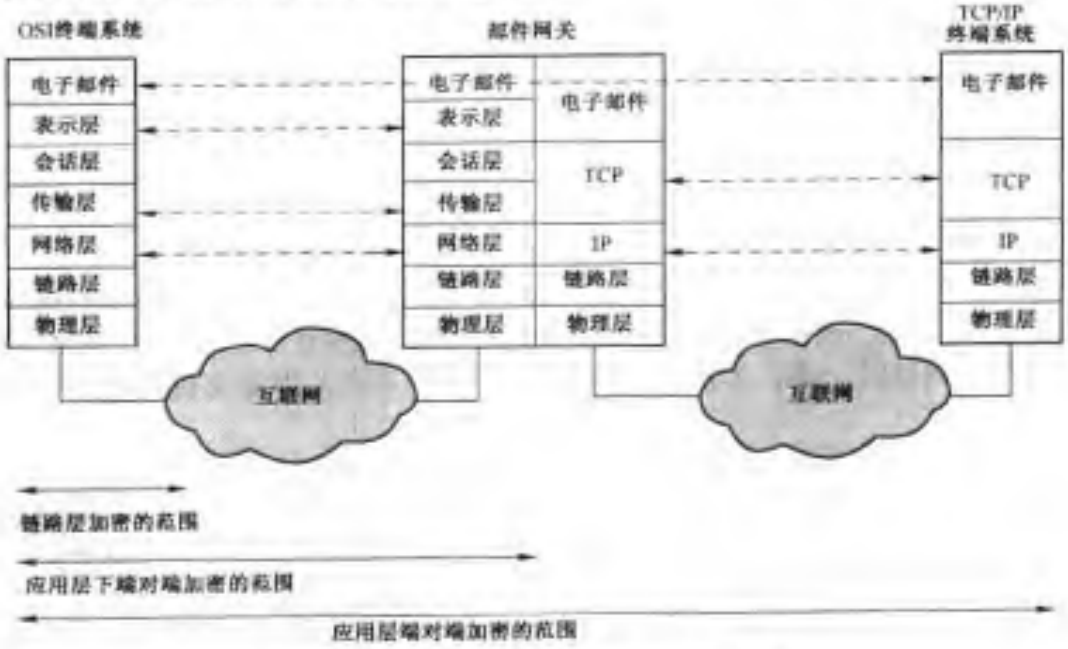


图 7.4 存储转发通信网络中的加密范围示意图

应用层加密的缺点是牵连的实体个数太多。支持几百个主机的网络可能有几千个用户或应用程序。所以，要生成和分配的密钥数也就太大。

注意，我们要是往通信层次上边走，那么要加密的信息少了，而安全性却更高。从图 7.5 中我们可以看得很清楚，其中使用了 TCP/IP 体系结构。图中的应用层网关是指在应用层上运行的存储转发设备。

对于应用层加密[见图 7.5(a)]，则只加密 TCP 信息片的用户数据，TCP、IP、网络层和链路层的信息头链路层的附加信息都以明文形式出现。如果在 TCP 这一层上加密[见图 7.5(b)]，那么对于单个端对端连接，用户数据和 TCP 头都被加密。因为路由器的需要，IP 层的信息头还是明文。然而，如果消息要经过网关，TCP 连接将终止并开始一段新的连接。此外，网关被看做是下面一层——IP 层的目的地，所以加密的用户数据将被解密。如果下一段网络还是基于 TCP/IP 的，传输之前用户数据和 TCP 要被再次加密。不过，在网关内部的缓冲区内数据是明文。最后，对于链路层加密[见图 7.5(c)]，除了链路头及附加信息外，所有数据都将被加密，只是在路由器和网关中才暂时恢复为明文。

图 7.5　不同加密策略的实现

7.2　传输保密性

第 1 章我们曾提到，有些场合用户很关注数据传输的安全性。拥有节点间的消息数目、消息长度等知识，都有助敌手判断出通信双方是谁。这在军事冲突中有明显的意义，即使是商业应用，数据传输分析也可以有助于得到传输方想隐蔽的信息。文献[MUFT89]列举了通过数据传输分析可以得到的信息的几种类型：

- 传输双方的标识
- 传输双方的联系频率
- 消息格式、消息长度或者消息频繁交换可以暗示出消息的重要性
- 与传输双方的某些谈话相关的事件

与消息传输相关的、另一个值得关注的事是**隐通道**。隐通道指的是采用某种方式进行通信，这种方式是通信设备的设计者所不知道的。通常，这种隐通道是违反安全规定的。例如，雇员想与外面的人进行联络，而不想被管理人员发现。参与双方可以对表面合法的消息构造一种编码，比如一段短的消息代表二进制 0，长的消息代表二进制 1，等等。

7.2.1　链路加密的方法

链路加密时消息头被加密了，减少了传输分析的机会，但是攻击者依然可以观察到网络上的数据流量。传输填充是一种有效对策，如图 7.6 所示。

传输填充持续地产生密文，即使没有明文输入。这种对策连续不断地产生随机数据流。有明文输入时，就将明文加密，然后发送；没有明文输入时，就把随机数加密并发送。这使得攻击者难以区分真实数据和无用数据，故不能分析出传输流量。

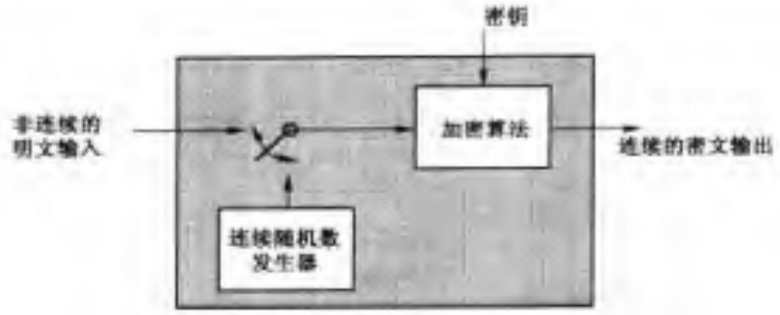


图 7.6 传输填充加密设备

7.2.2 端对端加密的方法

传输填充本质上是链路加密的方法。如果仅使用端对端加密，有效的防范办法会受到很大的限制。例如，若在应用层上实现加密，敌手可以判断出通信双方是谁，而若在传输层上加密，敌手可以弄到网络层的地址和传输格式。

有一个可能有用的办法，就是按统一长度的数据填充传输层和应用层传输的消息。此外，还可随机插入空消息到消息流中。这些策略隐藏了终端用户之间数据流量的知识并且让传输格式难以辨认。

7.3 密钥分配

对称密码要求消息交换双方共享密钥，并且此密钥不为他人所知。此外，密钥要经常变动，以防攻击者知道。因此，任何密码系统的强度都与密钥分配方法有关。密钥分配方法是指将密钥发放给希望交换数据的双方而不让别人知道的方法。对于参与者 A 和 B，密钥的分配有以下几种办法：

1. 密钥由 A 选择，并亲自交给 B。
2. 第三方选择密钥后亲自交给 A 和 B。
3. 如果 A 和 B 以前或最近使用过某密钥，其中一方可以用它加密一个新密钥后再发送给另一方。
4. A 和 B 与第三方 C 均有秘密渠道，则 C 可以将一密钥分别秘密发送给 A 和 B。

方法 1 和方法 2 需要人工传送密钥。对于链路加密，这个要求并不过分。因为每个链路加密设备仅同链路另一方进行数据交换，但是对于端对端加密，这样做未免有些笨拙。分布式系统中，任何主机或终端可能需要和其他许多主机或终端经常交换数据。所以，每个设备需要大量动态产生的密钥，特别是对于那些广域分布系统。

问题的规模取决于通信双方的个数。如果端对端加密在网络层或 IP 层实现，那么想相互通信的每对主机都得有一个密钥。这样 N 台主机需要的密钥数为 $[N(N-1)]/2$。若加密在应用层实现，则每对要相互通信的用户或进程需要一个密钥。一个网络可能有几百台主机，而用户和进程数则成千上万。图 7.7 表明了用端对端加密密钥分配的任务是何等繁重。可以想

像，1000 个节点的网络需要分配 50 万个密钥，若它还支持 10 000 个应用，则应用层加密还得有 5000 万个密钥。

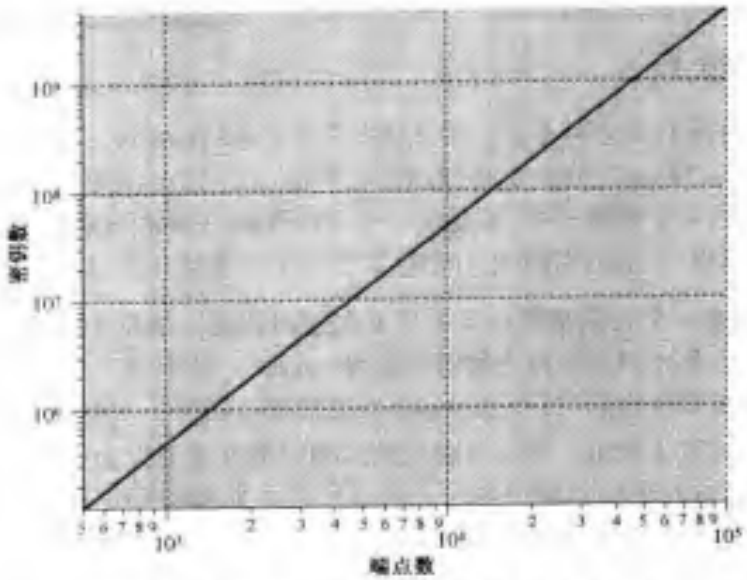


图 7.7　用于支持任意端点间连续所需的密钥数

再看看其余的方法。我们给出的方法 3 既可用于链路加密，也可用于端对端加密。但是，如果攻击者曾经成功获取一个密钥，则所有的子密钥都暴露了。此外，成千上万个初始密钥还是要想办法分发下去。

对于端对端加密，方法 4 稍做变动后被广泛应用。这时候，有一个密钥分配中心负责分发密钥给需要的用户（主机、进程、应用）。每个用户与密钥分配中心共享一个密钥，此密钥用于密钥分配。

密钥分配中心的工作基于对密钥的层次式使用。打个比方，使用两层密钥（见图 7.8）的情况：末端系统之间的通信用一个临时密钥加密，这个临时密钥常被称为**会话密钥**。一般情况下，会话密钥在建立逻辑连接时被使用，比如帧中继连接或传输连接，使用后该密钥就作废了。每个会话密钥经过用于末端用户通信的相同网络设备，从密钥分配中心获得。因此，会话密钥本身被加密后才被传送，加密它的密钥是由密钥分配中心和末端系统或用户共享的**主密钥**。

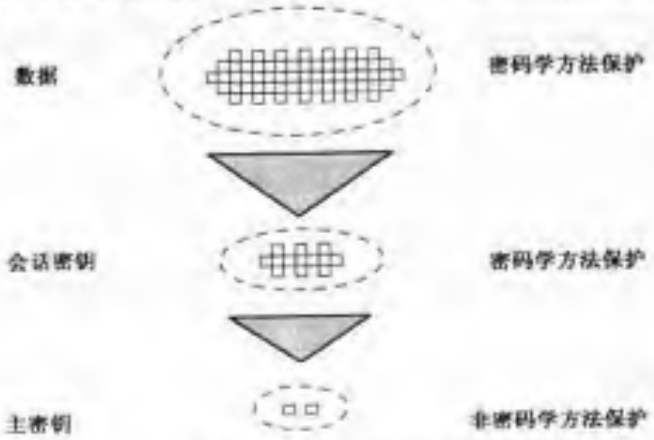


图 7.8　层次式密钥

对于每个末端系统或用户，主密钥是惟一的。当然，这些主密钥必须以某种方式分发。不过，问题的规模大大减小了。如果有 N 个希望互相通信的实体，刚才提到每次通信都需要有 $[N(N-1)]/2$ 个会话密钥，而现在只需要 N 个主密钥。主密钥可以通过非密码途径分发。

7.3.1 密钥分配模式

具体实现密钥分配有很多种办法。图 7.9 来自于文献[POPE79]，它给出了一种典型模式。这种模式假定每个用户与密钥分配中心(KDC)共享惟一的一个主密钥。

设 A 要与 B 建立一个逻辑连接，需要用一个一次性的会话密钥来保护数据的传输。A 有一个除了它外只有 KDC 知道的密钥 K_a；同样，B 有一个主密钥 K_b。具体过程是这样的：

1. A 向 KDC 请求一个会话密钥以保护与 B 的逻辑连接。消息中有 A 和 B 的标识及惟一的标识 N_1，这个标识我们称为**临时交互号**(nonce)。临时交互号可以是一个时间戳、计数值或者随机数什么的，只要每次通话时不同就可以了。当然，为了伪装，临时交互号也应尽量不让敌手知道。所以以随机数为临时交互号不失为一种好办法。
2. KDC 以用 K_a 加密的消息做出响应。所以只有 A 能够得到正确的消息，并可知它来自于 KDC。消息中有两项内容是给 A 的：
   - 一次性会话密钥 K_s，用于会话。
   - 原始请求消息，包括临时交互号，以使 A 使用适当的请求匹配这个响应。

   因此，A 可验证其原始请求在被 KDC 接收前不会更改，且由于临时交互号，这不是一些以前的请求的重放。

   此外，消息中有两项内容是给 B 的：
   - 一次性会话密钥 K_s，用于会话。
   - A 的标识符 ID_A(比如 A 的网络地址)。

   这两项用 K_b(KDC 与 B 共享的主密钥)加密。它们将发送给 B，以建立连接并证明 A 的标识。
3. A 存下会话密钥备用，并将消息的后两项发给 B，即 $E_{K_b}[K_S \| ID_A]$。因为这两项用 K_b 加密了，所以可防止窃听。现在 B 已知道会话密钥 K_S，知道它稍后的通话伙伴是 A(来自 ID_A)，且知道这些消息来自于 KDC，因为只有 KDC 与它共享 E_{K_b}。

这样，会话密钥就安全地发给了 A 和 B。它们可以开始受保护的信息交换了。不过，还有两步：

4. B 发送一个临时交互号 N_2 的密文给 A。
5. A 发送 $f(N_2)$的密文(以 K_s 加密)给 B，其中 $f(N_2)$是 N_2 的一个函数，比如加 1。

这两步保证 B 原来所收到(第 3 步)的报文不是一个重放。

注意，实际的密钥分配过程只包含第 1 步至第 3 步，而第 4 步和第 5 步以及第 3 步起了认证的作用。

7.3.2 层次式密钥控制

没有必要把密钥分配功能限定在单个 KDC 上。实际上，网络规模很大的时候，这样做是

不实际的。这时候，可以使用层次式 KDC。例如，有一些本地 KDC，各自负责整个网络中的一个小区域，比如说一个局域网或一栋大楼。如果两个实体在不同的区域，它们想共享一个密钥，就得让相应的本地 KDC 通过全局 KDC 进行通信。层次式的概念可以扩展到三层或更多层，这要看网络的用户数目和地域范围。

层次式密钥分配使主密钥分配的代价变小了，因为绝大部分主密钥是由本地 KDC 及其本地实体所共享。此外，如果一个本地 KDC 出错，或被攻破，则破坏只限于一个区域，而不会影响到全局。

7.3.3　会话密钥的生命期

会话密钥改变得越频繁，通信就越安全，因为敌手对任何会话密钥只拥有很少的密文。而另一方面，会话密钥的分配耽误了通信的时间，而且是网络容量的一个负担。安全管理者必须对这两方面进行折衷考虑，以决定会话密钥的生命期。

对于面向连接的协议，有一个自然而然的选择，那就是在开放连接期间，使用相同的会话密钥，改变连接后再用新的会话密钥。如果逻辑连接的时间特别长，则周期性地改变会话密钥必须十分小心，也许协议数据单元 PDU 的顺序号总在变化。

对于非面向连接的协议，比如面向事务的协议，它没有明显的连接开始和结束。因此，隔多长时间改变会话密钥不是那么明显。每次数据交换都用一个新的会话密钥是最安全的，不过，这样做显然是触动了无连接协议的根本好处，即延迟了交易时间，较好的策略是隔一段时间更改一次会话密钥，这段时间内只有一定的交易次数。

7.3.4　一种透明的密钥控制方案

图 7.9 给出的方法有多种变化，本小节讨论其中一种变化。这种方案（见图 7.10）可以在网络层和传输层提供一种对用户透明的端对端加密。假设使用的是面向连接的端对端协议，例如 X.25 或 TCP。图中的 FEP 即前端处理器很重要，它负责执行端加密并获得相应的会话密钥。

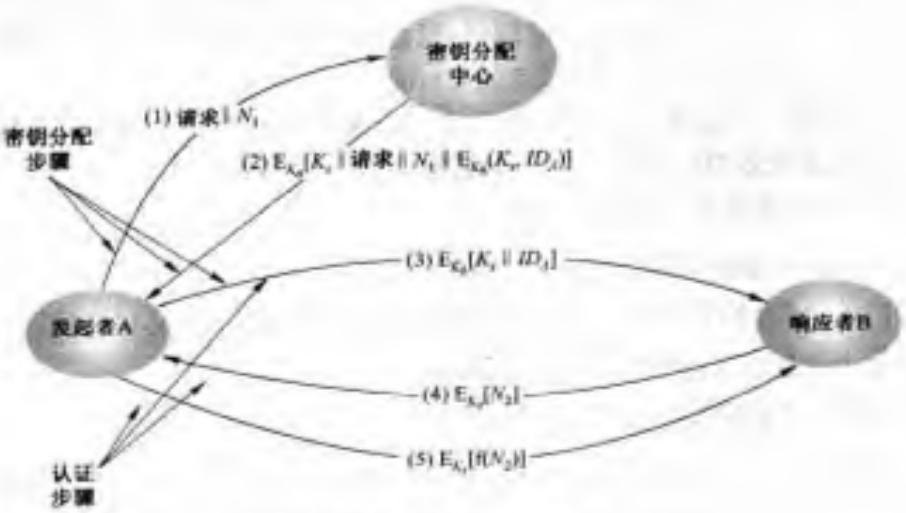


图 7.9　密钥分配过程

图 7.10 给出了建立连接的步骤。一主机若想与另一主机建立连接，就发送一个请求连接包(步骤 1)。FEP 和 KDC 之间的通信受它们共享的主密钥保护，如果 KDC 同意请求，就产生一个会话密钥并同时发给两个 FEP，且对每个 FEP 使用惟一的临时密钥(步骤 3)。现在发出请求的 FEP 可以释放连接请求包，两个末端系统之间的连接已经建立(步骤 4)。它们之间的所有用户数据都由各自的 FEP 用会话密钥加密。

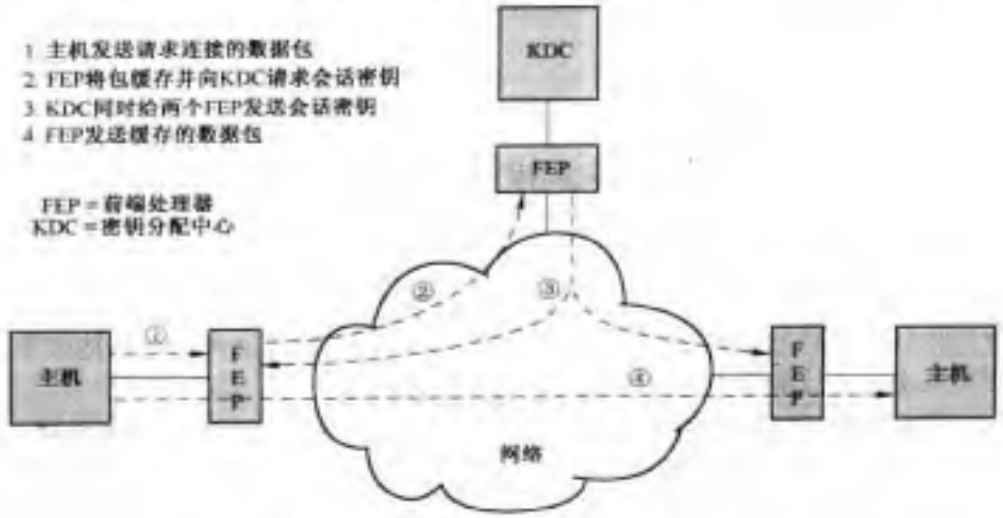


图 7.10 面向连接协议的自动密钥分析

这种方法的好处是减小了对末端系统的影响。从主机的角度看，FEP 如同一个分组交换节点，主机同网络的连接没有改变；从网络的角度看，FEP 如同一个主机，主机间的分组交换也没有改变。

7.3.5 分散式密钥控制

使用密钥分配中心必须保证 KDC 是可信任的，且不会受到破坏。如果密钥分配是完全分散的，则不用担这一点。尽管对于很大的网络，只使用传统密码来实现分散的密钥分配是不实际的，但是在某些特定场合下，这种方法还是可行的。

分散式密钥控制的方法需要每个末端系统能以安全的方式与潜在的通话伙伴通信。因此，对于 n 个末端系统须有 $[n/(n-1)]/2$ 个主密钥。

会话密钥按下列步骤生成(如图 7.11)：

1. A 向 B 请求会话密钥，同时发送临时交互号 N_1。
2. B 做出响应，回送消息包含 B 选择的会话密钥、B 的标识、$f(N_1)$以及另一个临时交互号 N_2。整个消息用 A、B 共享的主密钥加密。
3. 使用新的会话密钥加密，A 发 $f(N_2)$给 B。

因此，尽管每个节点至多只保存$(n-1)$个主密钥，但是可产生很多会话密钥，因为用主密钥加密的消息很短，所以密码分析困难。而会话密钥只使用有限的时间，不易被攻破。

图 7.11　分散式密钥分配

7.3.6　密钥的使用方法

使用层次式密钥控制及自动密钥生成方法，极大地减小了管理和分配密钥的工作量。但如何使用自动分配的密钥，还值得探讨。例如，除了将主密钥从会话密钥中分离出来，我们还希望定义几种类型的会话密钥，如：

- 数据加密密钥，用于网络中的一般通信。
- PIN 加密密钥，用于电子货币应用中的个人识别号(PIN)。
- 文件加密密钥，用于加密存储在大家都能访问到的地方的文件。

为了说明如此分类的用意，请考虑这种情况：将主密钥作为数据加密密钥用在某设备中。正常情况下，主密钥加密的会话密钥受到密钥分配中心和末端系统的密码硬件保护。用主密钥加密的会话密钥可用于应用程序，然后可用会话密钥加密数据。但是，如果将主密钥当做一个会话密钥，则有可能让未授权的用户利用主密钥获得其他会话密钥的明文。

所以，很有必要研究一下密钥在系统中的使用方法，这要根据密钥本身的特征来进行。简单的方法是给每个密钥一个标志(见文献[JONE82]及[DAVI89])。有人建议利用 DES 中64 位密钥的额外 8 位作为密钥标志，这 8 位原来是做校验用的。各位的解释如下：

- 一位表示密钥是主密钥还是会话密钥。
- 一位表示密钥可否用于加密。
- 一位表示密钥可否用于解密。
- 其余位留待将来使用。

因为这些标志都含在密钥中，所以密钥分配时被一起加密了。缺点是标志位只有 8 位，限制了其灵活性和功能，还有一个缺点是因为标志不能以明文形式传送，要用时必须将它解密，这限制了对密钥的管理。

控制矢量方案要灵活得多，其描述见文献[MATY91a 和 b]。这种方案给每个会话密钥一个关联控制矢量，它用一些域来指明会话密钥的使用及限制条件。控制矢量的长度可变。

控制矢量在 KDC 生成密钥时的使用过程如图 7.12 所示。第一步，控制矢量经过某个 hash 函数产生一个与加密密钥等长的值。第 10 章将详细讨论 hash 函数。hash 函数本质上是一个从大空间到小空间的有规律的映射。比如，1 到 100 之间的数经过 hash 函数映射为 1 到 10 之间的数。这样，每个目标值大约可由 10%的映射源得到。

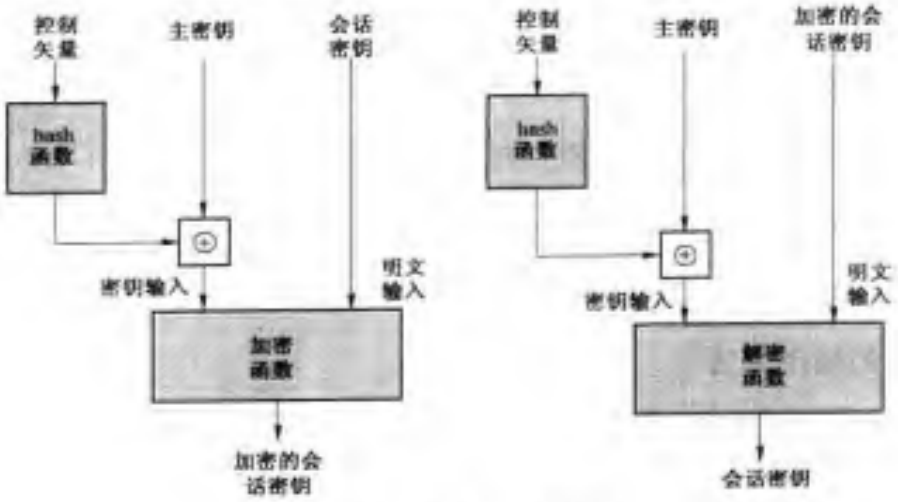


图 7.12 控制矢量的加密和解密

然后，将这个 hash 值与主密钥异或，将输出作为密钥来加密会话密钥。即：

$$\begin{aligned} \text{hash 值} &= H = \mathrm{h}(CV) \\ \text{密钥输入} &= K_m \oplus H \\ \text{密文} &= \mathrm{E}_{K_m \oplus H}[K_s] \end{aligned}$$

其中，K_m 是主密钥，K_s 是会话密钥。经过以下逆变换可恢复出明文：

$$K_s = \mathrm{D}_{K_m \oplus H}[\mathrm{E}_{K_m \oplus H}[K_s]]$$

KDC 将会话密钥发送给用户的同时，也将控制矢量以明文形式发送给用户。用户只有同时拥有主密钥和控制矢量才能恢复出会话密钥。这样，会话密钥及其控制矢量间的链接就得到了保持。

这种控制矢量的方案与 8 位标志的方案相比，有两个优点：一是对控制矢量的长度没有限制，可以实现对密钥的任意复杂的控制；二是控制矢量在操作过程中始终是明文，所以可多次运用对密钥的控制要求。

7.4 随机数的产生

随机数在许多网络安全应用中扮演着重要的角色。本节中，我们大致看一看它在网络安全中的作用，然后讨论几种生成随机数的方法。

7.4.1 随机数的使用

基于密码学的大量网络安全算法都使用了随机数。例如：

- 图 7.9 和图 7.11 所给出的相互认证方案。在密钥分配时，用临时交互号来作为握手信息之一，以阻止重复攻击。
- 会话密钥产生，可由密钥分配中心或是由委托方产生。
- RSA 公钥加密算法中密钥的产生（详见第 9 章）。

这些应用对随机数产生提出了两个不同的要求：随机性和不可预测性。

随机性

一般认为随机序列应有良好的统计特性。下面是两个评价标准：

- **分布一致性**：序列中随机数的分布应是一致的，即出现频率大约相等。
- **独立性**：序列中任何数不能由其他数推导出。

对随机序列的分布一致性已有较好的测试方法。但是，尽管有许多测试方法可以用于表明一个序列的独立性不好，还没有某种方法可以表明一个序列的独立性好。通常的策略是多进行一些测试，直至可认为它的独立性是足够强的。

密码算法大量使用了这种“似乎随机”的随机数序列。例如，第 9 章将要讨论 RSA 公开密钥加密方案的一个基本要求就是产生素数的能力。一般来说，判断一个给定的大数是否是素数是很难的。采用穷举测试的方法需要把 N 除以 $N^{1/2}$ 以内的数，如果 N 是 10^{150} 这种数量级的，这种大数在公钥密码中并非罕见，但是要用穷举测试它是否为素数则超出了人和计算机的实际能力。然而，有许多有效的算法，可以随机地选择一些随机数来进行相对简单的运算，只要随机数足够多[但是远比 $(10^{150})^{1/2}$ 小]，就几乎可判定这个大数是否为素数。这类方法称为不确定性方法，在算法设计中经常用到。本质上说，若某问题的精确求解很难或很耗时，可以采用简单的、所谓不确定的方法来取得某种可信程度的解。

不可预测性

在相互认证或会话密钥生成之类的应用中，对随机数的统计随机性的要求并不很高，但是要求产生的随机数序列是不可预测的。所谓的“真随机数序列”，是各个数之间的统计独立性而使序列不可预测。不过，真正的随机数序列很少用，一般的随机数序列是由算法产生的，只要敌手不能够从先前的随机数推导出后面的随机数就行了。

随机数的源

真随机数的源难以获得，物理噪声发生器可以算是这样的源，比如离子辐射事件、闸流管、漏电容的脉冲检测设备等。不过，这样的东西在网络安全应用中使用得有限。其实它们产生的随机数的随机性和精确度存在问题[BRIG79]，再说在网络中加上这些笨拙的设备也不太好。另一种选择是使用已经公布的好的随机数(参见文献[RAND55]、[TIPP27])，不过这些随机数与网络的需求相比，数目太少。此外，尽管本书中的随机数显示了良好的统计随机性，但它们是可预测的，因为知道这本书的敌手当然可以也搞到一本。

所以，密码应用大多使用算法来生成随机数。这些算法是确定的，所以产生的序列并非统计随机的。不过，要是算法好的话，产生的序列可以经受住随机性检测。这样的数一般称为**伪随机数**。

你也许对这种方法持怀疑态度，但是，只要我们不追求所谓的哲学上的完美性，这种方法的确有效。一位专家在概率论中谈到了这一点(参见文献[HAMM91])。

实际应用中，我们被迫尴尬地接受“相对随机”的概念。

7.4.2 伪随机数的产生

到现在为止,产生伪随机数的最广泛的方法是由 Lehmer 首先提出的算法(参见文献[LEHM51]),即线性拟合法。算法有以下 4 个参数:

m	模	$m > 0$
a	乘数	$0 < a < m$
c	增量	$0 \leqslant c < m$
X_0	初始值或种子	$0 \leqslant X_0 < m$

随机数序列$\{X_n\}$按下面的迭代方程获得:

$$X_{n+1} = (aX_n + c) \bmod m$$

若 m,a,c 和 X_0 都是整数,那么这种方法将产生一个随机数序列,且每个随机数都满足 $0 \leqslant X_n < m$。

要想设计一个好的随机数发生器,对 a,c 和 m 的选择至关重要。例如,假设 $a = c = 1$,产生的序列明显不行。假设 $a = 7$,$c = 0$,$m = 32$ 且 $X_0 = 1$,产生的序列是$\{7, 17, 23, 1, 7, \cdots\}$,这也明显不行,就是说 32 个可能的值,它只用了 4 个;即随机数序列的周期为 4。如果把 a 改成 5,序列就成了$\{5, 25, 29, 17, 21, 9, 13, 1, \cdots\}$,周期为 8。

m 一般都很大,因此可产生很多不同的随机数。一个常见的评价标准是 m 与给定计算机可表示的最大非负整数的值接近相等。这样,对于 32 位计算机,就有接近 2^{31} 个 m 值可供选择。

文献[PARK88]中提出了评价随机数发生器的三个标准:

T_1: 生成函数应是全周期的,即重复周期与 m 相等,亦即 0 ~ m 之间的所有数都可能。

T_2: 产生的序列应显得随机,因为是采用确定性生成随机数的办法,所以并非真随机,但是有多种统计测试方法可以评估其随机程度。

T_3: 生成函数可以用 32 位运算器方便地实现。

选择合适的 a、c 和 m,可以同时满足这三点。对于条件 T_1,可以证明若 m 是素数且 $c = 0$,则 a 的某些取值可以使产生函数的周期为 $m - 1$,只是不能得到 0 这个数。对于 32 位算术运算,$2^{31} - 1$ 就是一个常用的素数,这时的产生函数为:

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

a 的可能取值超过 20 亿个,但满足上述条件的只有其中一部分。a 取值为 $7^5 = 16\ 807$ 时可以满足上述条件,这个数最开始是在 IBM 360 系列计算机中使用的(见文献[LEWI69]),利用这个参数做成的伪随机数发生器使用得最广泛且经过了细致的测试,尤其适合于统计和仿真方面(参阅文献[JAIN91]和[SAUE81])。

若乘数和模选择恰当,用线性拟合算法产生的随机数序列的统计特性几乎与真随机数相当。但是除了初始值 X_0 之外,算法没有任何其余的东西是随机的,一旦 X_0 选定了,后续产生的随机数也就确定了,这一点对密码分析有帮助。

如果敌手知道了上述算法及参数(例如 $a = 7^5$、$c = 0$、$m = 2^{31} - 1$),只要他知道一个随机数,就可获得后续的所有序列。即使他只知道采用了线性拟合算法,那么只根据随机数序列中的一小部分就可以找到这些参数。设敌手可确定 X_0,X_1,X_2 和 X_3,则有:

$$X_1 = (aX_0 + c) \bmod m$$
$$X_2 = (aX_1 + c) \bmod m$$
$$X_3 = (aX_2 + c) \bmod m$$

由这三个等式可求解出 a、c 和 m。

因此，尽管这种办法用做伪随机数发生器有很多优点，但最理想的还是产生真正的随机数，这样的话敌手才不能由部分序列求得以后的序列。有几种办法可以达到这个目标。例如，文献[BRIG79]建议使用内部系统时钟来修正随机数流，一种方法是每隔 N 个数就以时钟值对 m 取模，以作为新的种子来产生新的序列。还有一种方法是直接将随机数加上时钟值再对 m 取模。

7.4.3　用密码编码学方法生成随机数

对于密码编码学应用，利用适当的加密逻辑来生成随机数还是很有意义的。已有一些方法得到了应用，本小节介绍三种典型的方法。

循环加密

图 7.13 给出了文献[MEYE82]所提出的一种方法。这里使用了用主密钥生成会话密钥的过程，用一个周期为 N 的计数器作为加密逻辑的输入。例如，若要产生 56 位的 DES 密钥，可以用一个周期为 2^{56} 的计数器。每产生一个密钥，计数器就增 1。因此，这种方法产生的伪随机数是全周期的，所有的输出 $X_0, X_1, \cdots, X_{N-1}$ 都是由不同的计数值而来，所以它们互不相同。因为主密钥被保护，所以由生成的随机数不能推出后续的随机数。为增加算法的强度，可以用一个全周期的伪随机数发生器来代替简单的计数器。

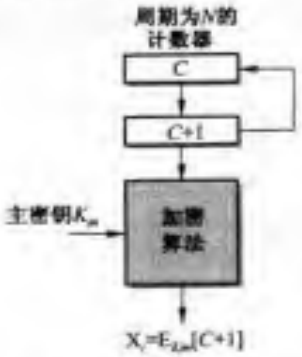


图 7.13　由计数器生成伪随机数

DES 输出反馈模式

图 3.14 中 DES 的输出反馈(OFB)模式可以用来产生密钥及流密码。注意其运算的每阶段的输出都是 64 位的值，其中最左 j 位被反馈作为加密输入。连续的 64 位输出构成有着良好统计性质的伪随机数序列。此外，还可用刚才讲的循环加密的方法，用一个受保护的主密钥来生成会话密钥，以作为最终的随机数。

ANSI X9.17 伪随机数发生器

ANSI X9.17 中所给出的伪随机数发生器是最强的伪随机数发生器之一。有一些应用使用了这种方法，如金融安全应用和 PGP(详见第 14 章)。

图 7.14 说明了算法流程，它使用了 3DES 作为加密。其组成如下：

- **输入**：用 2 个伪随机数输入来驱动发生器。一个是 64 位的数，代表当前的日期和时间，每产生一个伪随机数它均要改变。另一个是 64 位的种子，它可为任意值，并在生成随机数的过程中被更新。
- **密钥**：所用的 3DES 加密算法使用一个 56 位密钥对，这个密钥对必须保密，且只在产生随机数时才使用。
- **输出**：输出包括 64 位的伪随机数和 64 位的种子。

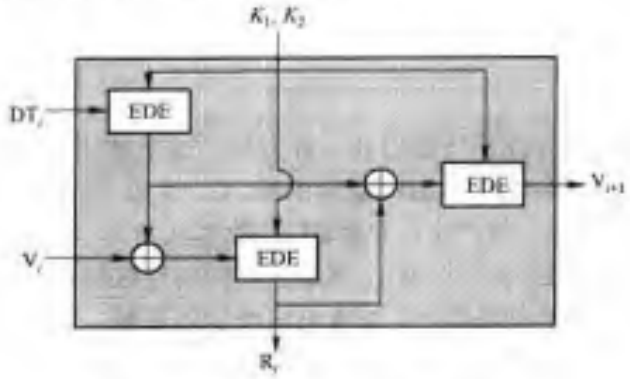


图 7.14 ANSI X9.17 伪随机数发生器

定义以下变量：

DT_i：算法第 i 轮的初始日期/时间值。

V_i：算法第 i 轮的初始种子值。

R_i：算法第 i 轮所产生的伪随机数。

K_1, K_2：算法所用的 DES 密钥。

则有：

$$R_i = EDE_{K_1,K_2}[V_i \oplus EDE_{K_1,K_2}[DT_i]]$$
$$V_{i+1} = EDE_{K_1,K_2}[R_i \oplus EDE_{K_1,K_2}[DT_i]]$$

其中 EDE 代表 3DES 加密的过程。

该方法的密码强度来自几个方面，包括 112 位密钥和 9 次 DES 加密。整个方案的输入是两个伪随机数，它们并非仅与该算法相关，所以敌手要分析的东西太多了，即使他知道了一个 R_i，也不能由 R_i 推导出 V_{i+1}，因为产生 V_{i+1} 时又用了一次 EDE 运算。

7.4.4 BBS 发生器

BBS 发生器是现在产生安全伪随机数的普遍方法。BBS 是 3 名设计者名字的首字母：

Blum, Blum, Shub, 它的密码强度也许是最强的[BLUM86]。产生过程如下：首先，选择 2 个大素数 p 和 q，且要求：

$$p \equiv q \equiv 3 (\mathrm{mod}\ 4)$$

例如，$7 \equiv 11 \equiv 3 (\mathrm{mod}\ 4)$，且 7 和 11 是素数。设 $n = p \times q$。接着，选择一个随机数 s，但要求 s 与 n 互素，即 s 与 p 或 q 皆无公因子。然后，BBS 按下列算法产生随机数序列：

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$\quad X_i = (X_{i-1})^2 \bmod n$$
$$\quad B_i = X_i \bmod 2$$

以 B_i 作为随机数的 1 位。表 7.2 给出了 BBS 运算的一个例子。这里，$n = 192\ 649 = 383 \times 503$，种子 $s = 101\ 355$。

表 7.2　BBS 伪随机数发生器的一个例子

s	X_i	B_i	s	X_i	B_i
0	20 749		11	137 922	0
1	143 135	1	12	123 175	1
2	177 671	1	13	8 630	0
3	97 048	0	14	114 386	0
4	89 992	0	15	14 863	1
5	174 051	1	16	133 015	1
6	80 649	1	17	106 065	1
7	45 663	1	18	45 870	0
8	69 442	0	19	137 171	1
9	186 894	0	20	48 060	0
10	177 046	0			

BBS 被称为**密码安全伪随机数位发生器**（CSPRBG）。它能经受住续位测试，在文献[MENE97]中续位测试定义如下："称某伪随机数位发生器可通过续位测试，若不存在多项式时间复杂度的算法，对子某输出序列的最初 k 位输入，可以以超过 1/2 的概率预测出第 $(k+1)$ 位"。换句话说，给定序列的最开始 k 位，没有有效算法可以让你以超过 1/2 的概率确定下一位是 1 还是 0。所以对于实际应用，这个序列是不可预测的。BBS 的安全性是基于对 n 的因子分解的困难性上的，即给定 n，我们不能确定它的素因子 p 和 q。

7.5　推荐读物和网址

文献[FUMY93]对密钥管理的原理做了很好的概述。至于伪随机数产生，最好的读物应该是[KNUT98]。文献[BRIG79]则详细解释了线性循环算法，而线性循环算法有些时候可以替代标准的线性拟合算法。文献[ZENG91]则将不同类型的伪随机数产生方法用在产生 Vernam 密码的变长密钥上，并对其优劣进行了评估。文献[MENE97]讨论了安全伪随机数发生器。RFC 1750 也很不错，不过它的重点在于实际应用。[KELS98]探讨了安全 PRNG 技术及攻击 PRNG 的密码学分析。

BRIG79 Bright, H., and Enison, R. "Quasi-Random Number Sequences from Long-Period TLP Generator with Remarks on Application to Cryptography." *Computing Surveys*, December 1979.

FUMY93 Fumy, S., and Landrock, P. "Principles of Key Management." *IEEE Journal on Selected Areas in Communications*, June 1993.

KELS98 Kelsey, J.; Schneier, B; and Hall, C. "Cryptanalytic Attacks on Pseudorandom Number Generators." *Proceedings, Fast Software Encryption*, 1998. http://www.counterpane.com/pseudorandom_number.html.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1998.

MENE97 Menezes, A.; Oorshcot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

ZENG91 Zeng. K.; Yang, C,; Wei, D.; and Rao, T. "Pseudorandom Bit Generators in Stream-Cipher Cryptography." *Computer*, February 1991.

推荐网址：

- **NIST Random Number Generation Technical Working Group**：包括 NIST 关于密码应用的 PRNG 的文档和试验，并且还有许多有用的链接。

7.6 关键术语、思考题和习题

7.6.1 关键术语

BBS 发生器	线性同余	伪随机数发生器
秘密渠道	主密钥	会话密钥
端对端加密	链路加密	传输填充
密钥分配	临时交互号	配线室
密钥分配中心(KDC)		

7.6.2 思考题

7.1 对于一个典型商业环境中的用户工作站，请列出对其窃密攻击的可能位置，即其中的安全隐患。

7.2 链路加密与端对端加密的区别是什么？

7.3 通过对数据传输的分析攻击能够获得什么类型的信息？

7.4 什么是传输填充？其作用是什么？

7.5 请列出将密钥分配到通信双方的几种方法。

7.6 会话密钥和主密钥的区别是什么?

7.7 什么是临时交互号?

7.8 什么是密钥分配中心?

7.9 统计随机性和不可预测性的区别是什么?

7.6.3 习题

7.1 不同电子邮件系统对于有多个接收者的情况的处理是不同的。有些系统将原始邮件做成多个拷贝,然后给每个接收者发送一份。另外一种选择是先确定给每个接收者的路径,然后将邮件在公共的路径上只发送一份,在路径分支上按路径分支的个数发送这么多份,这个过程我们称之为邮件装袋。

a. 不考虑安全性,讨论这两种方法的优缺点。

b. 讨论这两种方法的安全性。

7.2 在 7.2 节中,我们讨论了利用消息长度构造秘密渠道的方法。请给出三种其他用传输模式构造秘密渠道的方法。

7.3 某局域网销售商提供了如图 7.15 所示的密钥分配设备。

a. 描述其工作原理。

b. 将它与图 7.9 所示的密钥分配机制进行比较,它们各有哪些优缺点?

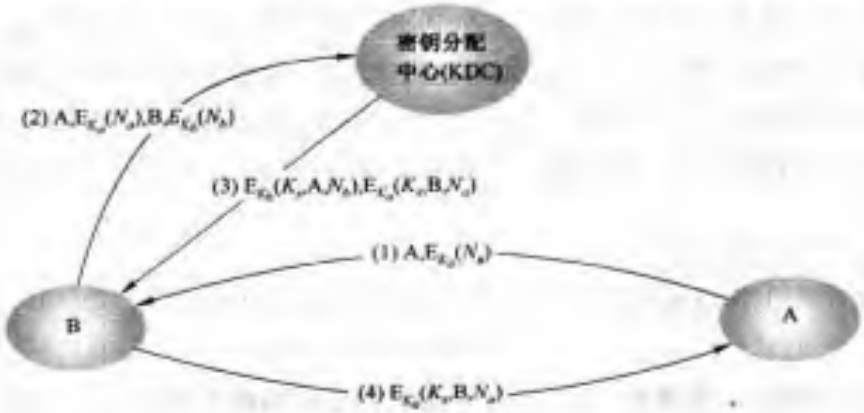


图 7.15　题 7.3 的图

7.4 "我们面临的压力很大,福尔摩斯。"侦探莱斯垂德神情焦虑地说,"我们已经知道了政府重要文件的拷贝存在某外国驻伦敦大使馆的电脑里。而正常情况下,只在政府少数几台电脑里面才有这些文件,而且这些电脑戒备森严。当然,有时候这些文件也会经过网络发送到政府其他电脑上。但是网络上的所有信息都已经被最强悍的加密算法加密过,而这些算法是由我们最好的密码专家设计的,即使是美国国家安全局和前苏联克格勃也没办法攻破。可是这些文件竟然出现在这样一个小国,甚至可以说是不值一提的小国手里。这是怎么一回事呢?"

"你总想到了一些可疑人物吧,有没有?"福尔摩斯问道。

"有,我们做了一些例行的调查。有一名男子合法地使用了政府的某台电脑,并且经常接触那个大使馆的文件。但是他用的电脑不是核心电脑,正常情况下没有那些文

件。他有些可疑，但是我们无法得知他是怎么搞到文件拷贝的。而且，即使他搞到了一个加密后的拷贝，他也没有办法解密呀。”

“嗯。你给我描述一下网络所用的通信协议吧。”福尔摩斯睁开了眼睛，这表示莱斯垂德的话已经吸引了他的注意力，使他从睡意朦胧中精神起来。

“好的。协议是这样的。网络的每个节点 N 都分配了一个惟一的密钥 K_n，用这个密钥来保证节点和可信任的服务器之间的通信安全。就是说，服务器上也存储了所有的密钥。用户 A 要想发送密文 M 给用户 B，使用以下协议：

1. A 产生一个随机数 R 并且将自己的名字、用户 B 的名字以及 $E_{K_a}[R]$ 发送给服务器。

2. 服务器做出响应，发送 $E_{K_b}[R]$ 给 A。

3. A 发送 $E_R[M]$ 和 $E_{K_b}[R]$ 给 B。

4. B 知道 K_b，因此将 $E_{K_b}[R]$ 解密，从而知道了 R，然后用 R 解密 $E_R[M]$ 得到 M。

你看，每次发送一段信息都要产生一个随机数。我承认那名男子能够从信道上截取到信息，但是我看他无法解密。”

“好，莱斯垂德，我想你找到了你需要的人了。通信协议是不安全的，因为服务器没有对向它发出请求的用户进行认证。显然协议的设计者相信发送 $E_{K_X}[R]$ 就意味着对用户 X 进行了认证，而且只有 X（和服务器）知道 K_X。但你知道 $E_{K_X}[R]$ 可以被侦听到，并在以后使用。一旦你知道了漏洞所在，你肯定可以通过监视该名男子对电脑的使用情况来获得证据。他最有可能这样进行工作：侦听到 $E_{K_a}[R]$ 和 $E_R[M]$ 之后（见协议的步骤 1 和步骤 3），该男子，我们且称它为 Z 吧，将会假扮成 A，然后……”

请继续讲述该故事。

7.5 如果将产生伪随机数的线性拟合法的附加项设为 0，即：

$$X_{n+1} = (aX_n) \bmod m$$

则可证明若 m 是素数，且给定的 a 使得算法获得最大周期 $m-1$，那么只要 k 小于 m 且 $m-1$ 不可被 k 整除，则 a^k 也使得算法获得最大周期 $m-1$。令，$X_0 = 1$，$m = 31$，$a = 3$ 时产生的序列为 $3, 3^2, 3^3, 3^4$。请证明之。

7.6 **a.** 下述伪随机数发生器可获得的最大周期是多少？

$$X_{n-1} = (aX_n) \bmod 2^4$$

b. 这时 a 为多少？

c. 对种子有什么要求？

7.7 你也许对线性拟合算法中选用模数 $m = 2^{31} - 1$ 而不是 2^{31} 感到奇怪，因为后者表示起来简单一些，且执行模运算也要简单一些，不过一般认为 $2^k - 1$ 比 2^k 要好，这是为什么？

7.8 在线性拟合算法中，选择的参数即使可提供全周期，也不一定能提供很好的随机性。例如，有下面两个伪随机数产生器：

$$X_{n-1} = (6X_n) \bmod 13$$
$$X_{n+1} = (7X_n) \bmod 13$$

写出随机数序列以证明它们都是全周期的。哪一个的随机性更好一些呢?

7.9 对伪随机数的任何使用,比如加密、仿真或是静态设计,盲目地相信计算机系统中的伪随机数产生器是危险的。文献[PARK88]发现当代许多教材和软件包中使用的伪随机数产生器有缺陷。下面这道题可使你对你的系统进行测试。

测试基于 Ernesto Cesaro 的理论(见文献[KNUT98]),他证明了两个随机选择的整数的最大公因子为 1 的可能性是 $6/\pi^2$。根据这个理论及统计学方法可以用程序来计算 π 的值。主程序要调用三个子程序:一个从系统中找到一些伪随机数产生器,并用它们产生一些伪随机数;一个计算两个整数的最大公因子;一个计算平方根。后面两个子程序如果已经有程序库可被调用就不用自己编写了。主程序应该有一个循环,且循环次数很大,以给出上述概率估计。这样,你就可以用这种简单的方法来估计 π 的值了。若结果接近于 3.14,那么祝贺你。如果不是,结果可能偏小,通常是在 2.7 附近。为什么得到的结果与 π 的值大相径庭呢?

7.10 有人提出一种方法,可用来确认你的密钥是否与另一人的相同。你首先产生一个与密钥等长的随机二进制位串,将其与密钥异或,并将结果通过某渠道发出。对方将收到的位串与自己的密钥(应与你的密钥相同)异或,并将结果发送回来。你将结果与开始的随机位串核对,就可以知道你的密钥是否与对方的相同,且在整个过程中没有发送真正的密钥。这种方案中有一个缺点,你看出来了吗?

第二部分　公钥加密与 hash 函数

第二部分涉及的内容

公钥加密或非对称加密是继对称加密之后的又一加密方法，它对通信安全具有革命性的意义。与密码有关的领域之一是密码 hash 函数，它与非对称密码一起用于数字签名中。除此之外，hash 函数还用于消息认证中。非对称密码也可用于密钥管理。第二部分将讨论上述这些相关领域。

第二部分内容浏览

第 8 章　数论入门

大多数公钥体制都基于数论。虽然读者可以放心地使用数论中的结论，但是掌握数论中的有关观念是非常有用的。第 8 章概述了数论的有关知识，并给出了大量的例子来阐述这些概念。

第 9 章　公钥密码和 RSA

第 9 章介绍公钥密码，重点介绍其在提供保密性方面的用途。本章还讨论了使用最为广泛的公钥密码 RSA 算法。

第 10 章　密钥管理和其他公钥密码体制

第 10 章根据非对称密码的特点重新讨论了密钥管理问题。本章还讨论了被广泛使用的 Diffie-Hellman 密钥交换技术，以及近来提出的、基于椭圆曲线的公钥密码。

第 11 章　消息认证和 hash 函数

认证作为一种安全措施与保密性具有同等的重要性。至少消息认证可保证消息来自所声称的源。除此之外，认证还可保护消息不被修改、延时、替代和重定序。第 11 章首先分析认证所应满足的要求，然后给出了一种用于认证的对称方法。认证方法中的一个关键因素是使用认证符，它通常是消息认证码(MAC)或 hash 函数。本章讨论了设计上述两种类型的算法应考虑的问题，并分析了几个特例。

第 12 章　hash 算法

第 12 章在前一章讨论的基础上，更加深入地讨论了当今最重要的 hash 函数 MD5、SHA-1 和 RIPEMD-160 算法，以及称为 HMAC 的 Internet 标准 MAC。

第 13 章　数字签名和认证协议

数字签名是一种重要的认证方法。第 13 章讨论了构造数字签名时所使用的方法，以及数字签名标准(DSS)这一重要标准。

认证算法中使用了各种基于数字签名的认证方法。这些算法的设计涉及到对一些攻击的分析，这些攻击可以攻破许多安全协议。第 14 章中还会讨论这一问题。

第8章 数论入门

数论中的许多概念在设计公钥密码算法时是必不可少的。本章讨论其他章节中所用到的一些概念，熟悉这些内容的读者可略过本章。

8.1 素数[①]

数论主要关心的是素数，实际上所有关于数论的书都是围绕这一主题来写的（如[CRAN01]、[RIBE96]）。本节简要介绍本书中所要用到的有关数论知识。

整数 $p > 1$ 是素数，当且仅当它只有因子 ± 1 和 $\pm p$。素数在数论和本章所讨论的各种方法中起着重要作用。

表 8.1 列出了 2000 以内的素数。

表 8.1 2000 以内的素数

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71	73	79	83	89	97
101	103	107	109	113	127	131	137	139	149	151	157	163	167	173	179	181	191	193	197	199				
211	223	227	229	233	239	241	251	257	263	269	271	277	281	283	293									
307	311	313	317	331	337	347	349	353	359	367	373	379	383	389	397									
401	409	419	421	431	433	439	443	449	457	461	463	467	479	487	491	499								
503	509	521	523	541	547	557	563	569	571	577	587	593	599											
601	607	613	617	619	631	641	643	647	653	659	661	673	677	683	691									
701	709	719	727	733	739	743	751	757	761	769	773	787	797											
809	811	821	823	827	829	839	853	857	859	863	877	881	883	887										
907	911	919	929	937	941	947	953	967	971	977	983	991	997											
1009	1013	1019	1021	1031	1033	1039	1049	1051	1061	1063	1069	1087	1091	1093	1097									
1103	1109	1117	1123	1129	1151	1153	1163	1171	1181	1187	1193													
1201	1213	1217	1223	1229	1231	1237	1249	1259	1277	1279	1283	1289	1291	1297										
1301	1303	1307	1319	1321	1327	1361	1367	1373	1381	1399														
1409	1423	1427	1429	1433	1439	1447	1451	1453	1459	1471	1481	1483	1487	1489	1493	1499								
1511	1523	1531	1543	1549	1553	1559	1567	1571	1579	1583	1597													
1601	1607	1609	1613	1619	1621	1627	1637	1657	1663	1667	1669	1693	1697	1699										
1709	1721	1723	1733	1741	1747	1753	1759	1777	1783	1787	1789													
1801	1811	1823	1831	1847	1861	1867	1871	1873	1877	1879	1889													
1901	1907	1913	1931	1933	1949	1951	1973	1979	1987	1993	1997	1999												

任意整数 $a > 1$ 都可以惟一地因子分解为：

$$a = p_1^{a_1} p_2^{a_2} \cdots p_t^{a_t}$$

其中，$p_1, p_2, \cdots, p_t$ 均是素数，$p_1 < p_2 < \cdots < p_t$，且对每一 a_i，有 $a_i > 0$。

$$91 = 7 \times 13; 11011 = 7 \times 11^2 \times 13$$

① 若无特别说明，则本节只讨论非负整数，使用负整数不会有本质上的不同。

下面介绍另一种有用的表示方法。设 P 是所有素数的集合，则任意正整数可惟一地表示为：

$$a = \prod_{p \in P} p^{a_p} \qquad \text{其中每个 } a_p \geqslant 0$$

上式右边是所有素数之积。对某一整数 a，其大多数指数 a_p 为 0。

$$3600 = 2^4 \times 3^2 \times 5^2$$

我们可以通过列出上述公式中所有非零指数来惟一地表示正整数。

整数 12 可用 $\{a_2 = 2,\ a_3 = 1\}$ 来表示。
整数 18 可用 $\{a_2 = 1,\ a_3 = 2\}$ 来表示。

两数相乘即是对应指数相加：

$$k = mn \rightarrow \qquad k_p = m_p + n_p \qquad \text{对任意 } p \in P$$

$$k = 12 \times 18 = 216$$
$$k_2 = 2 + 1 = 3;\ k_3 = 1 + 2 = 3$$
$$216 = 2^3 \times 3^3$$

从素因子的角度看，a 整除 b，即 $(a \mid b)$ 意味着什么呢[①]？由于任意形为 p^k 的整数，只能被 p^j 整除（其中 $j \leqslant k$），所以有：

$$a \mid b \rightarrow \qquad a_p \leqslant b_p \qquad \text{对所有 } p$$

$$a = 12;\ b = 36;\ 12 \mid 36;\ 12 = 2^2 \times 3;\ 36 = 2^2 \times 3^2$$
$$a_2 = 2 = b_2$$
$$a_3 = 1 \leqslant 2 = b_3$$

若将整数表示为素数之积，则很容易确定两个正整数的最大公因子[②]。

$$300 = 2^2 \times 3^1 \times 5^2$$
$$18 = 2^1 \times 3^2$$
$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

一般而言，

$$k = \gcd(a, b) \quad \rightarrow \quad k_p = \min(a_p, b_p) \text{ 对所有 } p$$

确定一个大数的素因子不是一件容易的事，因此利用上述关系式并不能直接得出计算最大公因子的实用方法。

---

① 若 a 除以 b 的余数为 0，则称 a 整除 b，见第 4 章。

② 若 c 能整除 a 和 b，且 a 和 b 的任何公因子均是 c 的因子，则称 c 为整数 a 和 b 的最大公因子，记为 $\gcd(a, b)$。见第 4 章。

8.2 Fermat 定理和 Euler 定理

Fermat 定理和 Euler 定理在公钥密码学中占有重要地位。

8.2.1 Fermat 定理[①]

Fermat 定理可如下描述:若 p 是素数,a 是正整数且不能被 p 整除,则:

$$a^{p-1} \equiv 1 \bmod p \tag{8.1}$$

证明:由第 4 章可知,若用 a 乘以 Z_p 中的所有元素并对 p 取模(其中 Z_p 为整数集合$\{0,1,\cdots,p-1\}$),则所得的结果将包含 Z_p 中的所有元素,而且 $a \times 0 \equiv 0 \bmod p$。所以$(p-1)$个数$\{a \bmod p, 2a \bmod p, \cdots, (p-1)a \bmod p\}$恰好是$\{1,2,\cdots,(p-1)\}$,将上述两个集合中的元素相乘并对 p 取模得到:

$$\begin{aligned} a \times 2a \times \cdots \times ((p-1)a) &\equiv [(a \bmod p) \times (2a \bmod p) \times \cdots \times \\ &\quad ((p-1)a \bmod p)] \bmod p \\ &\equiv [1 \times 2 \times \cdots \times (p-1)] \bmod p \\ &\equiv (p-1)! \bmod p \end{aligned}$$

但是

$$a \times 2a \times \cdots \times ((p-1)a) = (p-1)!a^{p-1}$$

所以

$$(p-1)!a^{p-1} \equiv (p-1)! \bmod p$$

因为$(p-1)!$ 与 p 互素[②],所以我们可以消去$(p-1)!$ 项[见等式(4.2)],故等式(8.1)成立。

$$\begin{gathered} a = 7,\ p = 19 \\ 7^2 = 49 \equiv 11 \bmod 19 \\ 7^4 \equiv 121 \equiv 7 \bmod 19 \\ 7^8 \equiv 49 \equiv 11 \bmod 19 \\ 7^{16} \equiv 121 \equiv 7 \bmod 19 \\ a^{p-1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \bmod 19 \end{gathered}$$

Fermat 定理的另一种有用的表示形式是:若 p 是素数且 a 是任意正整数,则:

$$a^p \equiv a \bmod p \tag{8.2}$$

$$\begin{gathered} p = 5,\ a = 3,\ 3^5 = 243 \equiv 3 \bmod 5 \\ p = 5,\ a = 10,\ 10^5 = 100\,000 \equiv 10 \bmod 5 \equiv 0 \bmod 5 \end{gathered}$$

8.2.2 Euler 函数

在给出 Euler 定理之前,我们需要引入数论中一个非常重要的概念,即 Euler 函数$\phi(n)$,它是指小于 n 且与 n 互素的正整数的个数。

① 该定理有时称为 Fermat 小定理。

② 见第 4 章。两个数若无公共素因子,即它们的惟一公因子为 1,则称这两个数是互素的;也就是说,若两个数的最大公因子为 1,则它们是互素的。

确定φ(37)和φ(35)的值。
因为 37 是素数，所以从 1 到 36 的所有正整数均与 37 互素。故φ(37) = 36。
要计算φ(35)，我们列出所有小于 35 且与 35 互素的正整数如下：
1,2,3,4,6,8,9,11,12,13,16,17,18,19,22,23,24,26,27,29,31,32,33,34
由上可知共有 24 个数，因此φ(35) = 24。
表 8.2 列出了φ(n)前 30 项的值，虽然φ(1)的值无意义，但我们仍将它定义为 1。

表 8.2　Euler 函数φ(n)的前 30 项的值

n	φ(n)	n	φ(n)	n	φ(n)
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

显然，对素数 p 有：

$$\phi(p) = p - 1$$

假设有两个素数 p 和 q，$p \neq q$，那么对 $n = pq$，有：

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

为了证明$\phi(n) = \phi(pq)$，考虑 Z_n 中的余数集合$\{0,1,\cdots,(pq-1)\}$。不与 n 互素的余数集合是$\{p,2p,\cdots,(q-1)p\}$，$\{q,2q,\cdots,(p-1)q\}$和 0，所以有：

$$\begin{aligned}\phi(n) &= pq - [(q-1) + (p-1) + 1] \\ &= pq - (p+q) + 1 \\ &= (p-1) \times (q-1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

$\phi(21) = \phi(3) \times \phi(7) = (3-1) \times (7-1) = 2 \times 6 = 12$
其中这 12 个整数是$\{1,2,4,5,8,10,11,13,16,17,19,20\}$

8.2.3　Euler 定理

Euler 定理说明，对任意互素的 a 和 n，有：

$$a^{\phi(n)} \equiv 1 \bmod n \tag{8.3}$$

$a = 3;\ n = 10;\ \phi(10) = 4;\ 3^4 = 81 \equiv 1 \bmod 10$
$a = 2;\ n = 11;\ \phi(11) = 10;\ 2^{10} = 1024 \equiv 1 \bmod 11$

证明:若 n 是素数,根据$\phi(n)=(n-1)$和 Fermat 定理,则有等式(8.3)成立。但实际上等式(8.3)对任意整数 n 都是成立的。请注意$\phi(n)$是指小于 n 且与 n 互素的正整数的个数。下面我们考虑这些整数所组成的集合:

$$R=\{x_1,x_2,\cdots,x_{\phi(n)}\}$$

用 a 与 R 中的每个元素模 n 相乘:

$$S=\{(ax_1 \bmod n),(ax_2 \bmod n),\cdots,(ax_{\phi(n)} \bmod n)\}$$

集合 S 是 R 的一个置换,原因如下:

1. a 与 n 互素,且 x_i 与 n 互素,所以 ax_i 必与 n 互素。这样,S 中的所有元素均小于 n,且与 n 互素。
2. S 中没有重复元素,参见等式(4.2)。若 $ax_i \bmod n = ax_j \bmod n$,则 $x_i=x_j$。

因此,我们有:

$$\prod_{i=1}^{\phi(n)}(ax_i \bmod n)=\prod_{j=1}^{\phi(n)}x_i$$

$$\prod_{i=1}^{\phi(n)}ax_i \equiv \prod_{i=1}^{\phi(n)}x_i \pmod n$$

$$a^{\phi(n)}\times\left[\prod_{i=1}^{\phi(n)}\right]x_i \equiv \prod_{i=1}^{\phi(n)}x_i \pmod n$$

$$a^{\phi(n)} \equiv 1 \pmod n$$

Euler 定理的另一种有用的表达形式是:

$$a^{\phi(n)+1} \equiv a \pmod n \tag{8.4}$$

下面给出 Euler 定理的一个推论,它对证明 RSA 算法(见第 9 章)的有效性非常有用。给定两个素数 p 和 q,整数 $n=pq$ 和 m,其中 $0<m<n$,则下列关系式成立:

$$m^{\phi(n)+1}=m^{(p-1)(q-1)+1} \equiv m \bmod n \tag{8.5}$$

若 $\gcd(m,n)=1$,即 m 与 n 互素,则由 Euler 定理[见等式(8.3)]可知上述关系成立。假定 $\gcd(m,n)\neq 1$。这是什么意思呢?由于 $n=pq$,所以若 m 是 p 的倍数,则 m 与 n 有公共的素因子 p,这样 m 与 n 不是互素的;若 m 是 q 的倍数,则 m 与 n 有公共的素因子 q,这样 m 与 n 也不是互素的。因此,$\gcd(m,n)=1$ 等价于逻辑表达式(m 不是 p 的倍数)且(m 不是 q 的倍数)。故 $\gcd(m,n)\neq 1$ 等价于上述逻辑表达式的否定,从而 $\gcd(m,n)\neq 1$ 等价于逻辑表达式(m 是 p 的倍数)或(m 是 q 的倍数)。

假设 m 是 p 的倍数,那么有某正整数 c 使得 $m=cp$,此时 $\gcd(m,q)=1$;否则 m 既是 p 的倍数,也是 q 的倍数,与 $m<pq$ 矛盾。由于 $\gcd(m,q)=1$,所以 Euler 定理成立且有:

$$m^{\phi(q)} \equiv 1 \bmod q$$

面根据模算术的性质有:

$$[m^{\phi(q)}]^{\phi(p)} \equiv 1 \bmod q$$

$$m^{\phi(n)} \equiv 1 \bmod q$$

因此有某整数 k 使得：

$$m^{\phi(n)} = 1 + kq$$

两边同乘 $m = cp$ 后得：

$$m^{\phi(n)+1} = m + kcpq = m + kcn$$
$$m^{\phi(n)+1} \equiv m \bmod n$$

同理可证 m 是 q 的倍数的情形。故等式(8.5)得证。该推论的另一种表达形式与 RSA 有直接联系：

$$\begin{aligned} m^{k\phi(n)+1} &\equiv [(m^{\phi(n)})^k \times m] \bmod n \\ &\equiv [(1)^k \times m] \bmod n \\ &\equiv m \bmod n \end{aligned} \tag{8.6}$$

8.3　素性测试

在许多加密算法中，需要随机选择一个或多个非常大的素数，因此我们必须确定一个给定的大数是否是素数。目前还没有简单有效的方法解决该问题。

本节介绍一种较好的常用算法，该算法产生的数不一定是素数，你也许为此会感到惊讶，然而，该算法产生的数几乎可以肯定是素数。我们将在下面解释这一点。Miller 和 Rabin 提出的这一算法[MILL75, RABI80]，利用了 Fermat 定理[等式(8.1)]，即若 n 是素数，则 $a^{n-1} \equiv 1 \bmod n$。

下面我们来解释该算法。对一候选的奇整数 $n \geqslant 3$，考虑偶数 $(n-1)$，它可表示为 2 的某次幂与一个奇数之积：

$$n - 1 = 2^k q \qquad k > 0, q \text{ 为奇数}$$

即我们用 2 除 $(n-1)$，直至所得结果为奇数，此处共做了 k 次除法。

接下来，我们选一个整数 a，$1 < a < n-1$。该算法计算下述幂序列模 n 的余数：

$$a^q, a^{2q}, \cdots, a^{2^{k-1}q}, a^{2^k q} \tag{8.7}$$

若 n 是素数，则由 Fermat 定理可知，$a^{2^k q} \bmod n = a^{n-1} \bmod n = 1$。序列(8.7)中可能在 $a^{2^k q}$ 前就存在余数为 1 的元素。我们可将序列(8.7)表示为 $\{a^{2^j q},\ 0 \leqslant j \leqslant k\}$，那么若 n 是素数，则有某最小的 $j(0 \leqslant j \leqslant k)$ 使得 $a^{2^j q} \bmod n = 1$。考虑下列两种情形：

- **情形 1($j = 0$)**：此时我们有 $a^q - 1 \bmod n = 0$，或 n 整除 $(a^q - 1)$。
- **情形 2($1 \leqslant j \leqslant k$)**：此时我们有 $(a^{2^j q} - 1) \bmod n = (a^{2^{j-1} q} - 1)(a^{2^{j-1} q} + 1) \bmod n = 0$。这意味着 n 整除 $(a^{2^{j-1} q} - 1)$ 或 $(a^{2^{j-1} q} + 1)$。因为根据假设，j 是使 n 整除 $(a^{2^j q} - 1)$ 的最小整数，所以 n 不能整除 $(a^{2^{j-1} q} - 1)$，因此 n 整除 $(a^{2^{j-1} q} + 1)$，即 $a^{2^{j-1} q} \bmod n = (-1) \bmod n = n - 1$。

由上可知，若 n 是素数，则要么余数序列 $(a^q,\ a^{2q}, \cdots, a^{2^{k-1} q},\ a^{2^k q})$ 的第一个元素为 1，要么该序列中某元素为 $n-1$。否则 n 是合数(即不是素数)。然而，若条件成立，并不一定意味着 n 是素数。

我们可将上述算法形式化为过程 TEST。该过程的输入为待测试的整数 n，并且若 n 肯定不是素数，则返回“合数”；若 n 可能是也可能不是素数，则返回“不确定”。

TEST(n)

1. 找出整数 k,q,其中 $k>0,q$ 是奇数,使($n-1=2^kq$)。
2. 随机选取整数 $a,1<a<n-1$。
3. **if** $a^q \bmod n=1$, **then** 返回"不确定"。
4. **for** $j=0$ **to** $k-1$ **do**。
5. **if** $a^{2^jq} \bmod n=n-1$ **then** 返回"不确定"。
6. 返回"合数"。

对素数 $n=29$ 应用上述测试算法。我们有($n-1$) $=28=2^2(7)=2^kq$。首先选取 $a=10$,计算 $10^7 \bmod 29=17$,它既不为 1 也不为 28,所以我们继续测试。然后计算$(10^7)^2=\bmod 29=28$,因此测试算法返回"不确定"。我们再选取 $a=2$。由计算可知 $2^7 \bmod 29=12$, $2^{14} \bmod 29=28$;因此仍返回"不确定"。如果我们对 1 到 28 间的所有整数 a 执行测试算法,则得到的同样是"不确定"。这一点与 n 为素数是相一致的。

现在对合数 $n=13\times17=221$ 应用上述测试,则($n-1$) $=220=2^2(55)=2^kq$。选取 $a=5$,则 $5^{55} \bmod 221=112$,它既不是 1 也不是 220;$(5^{55})^2 \bmod 221=168$,因为我们已对 TEST 算法第 4 行中所有的 j(0,1)进行了测试,所以算法返回"合数",这表明 221 肯定是合数。但是假设我们选取 $a=21$,则 $21^{55} \bmod 221=200$,$(21^{55})^2 \bmod 221=220$,则测试法返回"不确定",表明 221 可能是素数。事实上,在 1 到 220 之间的这 220 个整数中,有 6 个整数会返回"不确定",它们是 1,21,47,174,200 和 220。

8.3.1 概率方面的考虑

有两个概率问题需要考虑。第一,我们如何利用 Miller-Rabin 算法确定某整数是或不是素数,并且要具有很高的可信度;第二,随机选取大素数的难度如何?

重复使用Miller-Rabin算法

我们已经知道,若 TEST(n)返回"合数",则 n 不是素数。可以证明(例如[ROSE00],[KNUT98]),对给定的不是素数的整数 n,Miller 测试至多对($n-1$)/4 个整数 $a(1\leqslant a\leqslant n-1)$返回"不确定"。因此给定合数 n,并随机选取整数 a,TEST 返回"不确定"(即不能确定 n 不是素数)的概率为:

$$\frac{(n-1)/4}{n}<\frac{1}{4}$$

这为我们决定一个奇整数是素数且具有合理的可信度奠定了基础。其过程如下:对随机选取的 a,重复调用 TEST(n),如果某时刻 TEST 返回"合数",则 n 一定不是素数;若 TEST 连续 t 次返回"不确定",则 n 是素数的概率至少是 $1-4^{-t}$。例如对 $t=10$,n 是素数的概率大于 0.999 999。这样,对足够大的 t,若 Miller 算法总是返回"不确定",则我们可以相信 n 是素数。

素数的分布

值得注意的是,在利用 Miller-Rabin 测试或任何其他的素数测试算法发现一个素数之前已

经测试了多少个素数？由数论中的素数定理可知，n 附近的素数分布情况为，平均每 $\ln(n)$ 个整数中有一个素数。这样，平均而言，在找到一个素数之前，我们必须测试约 $\ln(n)$ 个整数。因为所有偶数以及以 5 结尾的整数肯定不是素数，所以需测试的整数个数为 $0.4\ln(n)$。例如，若要找 2^{200} 左右的素数，则约需 $0.4\ln(2^{200}) = 55$ 次测试。然而，这只是平均值。在数轴上的某些位置，素数非常密集，而在其他有些位置，素数非常稀疏。

> 两个相邻的奇数 1 000 000 000 061 和 1 000 000 000 063 都是素数；而另一方面，1001! + 2，1001! + 3，…，1001! + 1000，1001! + 1001这 1000 个连续的整数均是合数。

8.4　中国剩余定理

中国剩余定理(CRT)[①] 是数论中最有用的定理之一。CRT 说明某一范围内的整数可通过它对两两互素的整数取模所得的余数来重构。

> $Z_{10}(0,1,\cdots,9)$中的 10 个整数可通过它们对 2 和 5(10 的素因子)取模所得的两个余数来重构。假设已知十进制数 x 的余数 $r_2 = 0$ 且 $r_5 = 3$，即 $x \bmod 2 = 0$ 且 $x \bmod 5 = 3$，则 x 是 Z_{10}中的偶数且被 5 除余 3，故惟一解为 $x = 8$。

CRT 可有几种不同的表示形式，这里我们给出其中一种最有用的表示形式，习题 8.10 给出了另一种表示形式。令

$$M = \prod_{i=1}^{k} m_i$$

其中 m_i 是两两互素的，即对 $1 \leqslant i, j \leqslant k, i \neq j$ 有 $\gcd(m_i, m_j) = 1$。我们可将 Z_M 中的任一整数对应一个 k 元组，该 k 元组的元素均在 Z_m 中，这种对应关系为：

$$A \leftrightarrow (a_1, a_2, \cdots, a_k) \tag{8.8}$$

其中 $A \in Z_M$，对 $1 \leqslant i \leqslant k$，$a_i \in Z_{m_i}$，且 $a_i = A \bmod m_i$。CRT 说明下列两个断言成立：

1. 等式(8.8)中的映射是 Z_M 到笛卡儿积 $Z_{m_1} \times Z_{m_2} \times \cdots \times Z_{m_k}$ 的一一对应(称为**双射**)，也就是说，对任何 A，$0 \leqslant A < M$，有惟一的 k 元组 $(a_1, a_2, \cdots, a_k)$ 与之对应，其中 $0 \leqslant a_i \leqslant m_i$，并且对任何这样的 k 元组 $(a_1, a_2, \cdots, a_k)$，Z_M 中有惟一的 A 与之对应。
2. Z_M 中元素上的运算可等价于对应的 k 元组上的运算，即在笛卡儿积的每一个分量上独立地执行运算。

上述第二个断言也可如下描述。若：

$$A \leftrightarrow (a_1, a_2, \cdots, a_k);\ B \leftrightarrow (b_1, b_2, \cdots, b_k)$$

则

---

① 因为人们认为这个定理是公元 100 年由中国数学家孙子(SunTse)发现的，所以称为 CRT。

$$(A + B) \bmod M \leftrightarrow ((a_1 + b_1) \bmod m_1, \cdots, (a_k + b_k) \bmod m_k)$$
$$(A - B) \bmod M \leftrightarrow ((a_1 - b_1) \bmod m_1, \cdots, (a_k - b_k) \bmod m_k)$$
$$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, \cdots, (a_k \times b_k) \bmod m_k)$$

下面证明第一个断言。只需取 $a_i = A \bmod m_i$，由 A 到$(a_1, a_2, \cdots, a_k)$的转换显然是惟一确定的。对给定的$(a_1, a_2, \cdots, a_k)$，可如下计算 A。对 $1 \leqslant i \leqslant k$，令 $M_i = M/m_i$，因为 $M_i = m_1 \times m_2 \times \cdots \times m_{i-1} \times m_{i+1} \times \cdots \times m_k$，所以对所有的 $j \neq i$，有 $M_i \equiv 0 \pmod{m_j}$。令

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \qquad 1 \leqslant i \leqslant k \tag{8.9}$$

根据 M_i 的定义有 M_i 与 m_i 互素，所以 M_i 有惟一的模 m_i 的乘法逆元，因此等式(8.9)是良定的，这样可得到惟一的 c_i。我们现在计算

$$A = \left(\sum_{i=1}^{k} a_i c_i \right) \bmod M \tag{8.10}$$

要证明等式(8.10)产生的 A 是正确的，必须证明对 $1 \leqslant i \leqslant k$ 有 $a_i = A \bmod m_i$。由于 $j \neq i$ 时，$c_j \equiv M_j \equiv 0 \pmod{m_i}$，而且 $c_i \equiv 1 \pmod{m_i}$，故 $a_i = A \bmod m_i$。

CRT 的第二个断言与模运算有关，根据模算术的性质即可知其成立。

中国剩余定理的用途之一是，它给出了一种方法，使非常大的数对 M 的模运算转化到更小的数上来进行运算，当 M 为 150 位或 150 位以上时，这种方法非常有效。

下面将 973 mod 1813 表示为模 37 和 49 的两个数。我们定义[①]

$$m_1 = 37$$
$$m_2 = 49$$
$$M = 1813$$
$$A = 973$$

则 $M_1 = 49$ 且 $M_2 = 37$。利用扩展的欧几里德算法有 $M_1^{-1} = 34 \bmod m_1$ 且 $M_2^{-1} = 4 \bmod m_2$(注意每个 M_i 和 M_i^{-1} 只需计算一次)。对 37 和 49 取模，因为 973 mod 37 = 11，973 mod 49 = 42，所以 973 可表示为(11,42)。

假定要计算 678 加 973。如何处理(11,42)呢？首先计算(678)↔(678 mod 37，678 mod 49) = (12,41)，然后将二元组的元素相加并化简(11 + 12 mod 37，42 + 41 mod 49) = (23,34)。这个结果是正确的，因为

$$\begin{aligned}(23,34) &\leftrightarrow a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \bmod M \\ &= [(23) \times (49) \times (34) + (34) \times (37) \times (4)] \bmod 1813 \\ &= 43\,350 \bmod 1813 \\ &= 1651\end{aligned}$$

而 $(973 + 678) \bmod 1813 = 1651$。

① 本例由乔治亚工学院的 Ken Calvert 教授提供。

假定要计算 73 乘 1651(mod 1813)。我们首先用 73 乘(23,24)并化简得(23 × 73 mod 37,34 × 73 mod 49) = (14,32)。很容易验证:

$$\begin{aligned}(14,32) &\leftrightarrow [(14)\times(49)\times(34)+(32)\times(37)\times(4)] \bmod 1813\\ &= 856\\ &= 1651\times 73 \bmod 1813\end{aligned}$$

8.5 离散对数

离散对数是包括 Diffie-Hellman 密钥交换和数字签名算法(DSA)在内的许多公钥算法的基础。本节简要介绍离散对数的一般知识,有兴趣者可参阅[ORE67]和[LEVE90],它们更详细地介绍了这方面的知识。

8.5.1 模 n 的整数幂

Euler 定理[等式(8.3)]告诉我们,对任何互素的 a 和 n,有:

$$a^{\phi(n)} \equiv 1 \bmod n$$

其中 Euler 函数$\phi(n)$是指小于 n 且与 n 互素的正整数的个数。下面我们考虑 Euler 定理更一般的表示形式:

$$a^m \equiv 1 \bmod n \tag{8.11}$$

若 a 与 n 互素,则至少有一个整数 m 满足等式(8.11),即 $m=\phi(n)$,我们称使等式(8.11)成立的最小正幂 m 为下列之一:

- a(模 n)的阶
- a 所属的模 n 的指数
- a 所产生的周期长

为说明最后一个概念,我们考虑下面 7 模 19 的各次幂:

$$\begin{aligned}7^1 &= 7 \bmod 19\\ 7^2 &= 49 = 2\times 19+11 \equiv 11 \bmod 19\\ 7^3 &= 343 = 18\times 19+1 = 1 \bmod 19\\ 7^4 &= 2401 = 126\times 19+7 = 7 \bmod 19\\ 7^5 &= 16\,807 = 884\times 19+11 = 11 \bmod 19\end{aligned}$$

由于上述计算中出现了重复,所以不必再往下计算,因为由 $7^3 \equiv 1 \pmod{19}$可得 $7^{3+j} \equiv 7^3 7^j \equiv 7^j \pmod{19}$,这说明若 7 的两个指数相差 3(或 3 的倍数),则以它们为指数的 7 的(模 19)幂是相同的。换句话说,该序列是周期性的,且其周期长度是使 $7^m \equiv 1 \pmod{19}$成立的最小正幂 m。

表 8.3 给出了所有小于 19 的正整数 a 模 19 的各次幂，阴影部分指明了每个底 a 的幂序列长。请注意：

1. 所有序列均以 1 结束。这与前面的推导是一致的。
2. 每个序列的长度都可以整除$\phi(19) = 18$。也就是说，表中每一行都有整数个子序列。
3. 有些序列长为 18。这时我们说底 a（通过幂运算）生成模 19 的非零整数集，并称这样的整数为模数 19 的本原根。

更一般地，我们说$\phi(n)$是一个数所属的模 n 的可能的最高指数。如果一个数的阶为$\phi(n)$，则称之为 n 的**本原根**。本原根的重要之处在于，若 a 是 n 的本原根，则其幂

$$a, a^2, \cdots, a^{\phi(n)}$$

是(模 n)各不相同的，且均与 n 互素。特别地，对素数 p，若 a 是 p 的本原根，则

$$a, a^2, \cdots, a^{p-1}$$

是(模 p)各不相同的。素数 19 的本原根为 2,3,10,13,14 和 15。

并不是所有的整数都有本原根。事实上，只有形为 2,4,p^n 和 $2p^n$ 的整数才有本原根，这里 p 是任何奇素数，a 是正整数。

表 8.3 模 19 的整数幂

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

8.5.2　指标

对于普通正实数，对数函数是指数函数的逆函数。对模算术，也有类似的函数。

我们先简要讨论普通对数的性质。给定一个数，它的对数是满足下列条件的幂：以某个正数（除 1 外）为底的该次幂恰好等于给定的这个整数，即对底 x 和数 y，有：

$$y = x^{\log_x(y)}$$

对数具有下列性质：

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z) \tag{8.12}$$

$$\log_x(y^r) = r \times \log_x(y) \tag{8.13}$$

对某素数 p（对非素数亦可）的本原根 a，a 的 1 到 $(p-1)$ 的各次幂恰可产生 1 到 $(p-1)$ 的每个整数一次且仅一次。而对任何整数 b，根据模算术的定义，b 可表示为如下形式：

$$b \equiv r \bmod p \qquad \text{其中 } 0 \leqslant r \leqslant (p-1)$$

因此，对任何整数 b 和素数 p 的本原根 a，有惟一的幂 i 使得：

$$b \equiv a^i \bmod p \qquad \text{其中 } 0 \leqslant i \leqslant (p-1)$$

该指数 i 称为以底 a（模 p）的 b 的指标，记为 $\text{ind}_{a,p}(b)$。

请注意下列各式：

$$\text{ind}_{a,p}(1) = 0, \text{因为} a^0 \bmod p = 1 \bmod p = 1 \tag{8.14}$$

$$\text{ind}_{a,p}(a) = 1, \text{因为} a^1 \bmod p = a \tag{8.15}$$

下面的例子使用的是非素数模，$n = 9$。这里 $\phi(n) = 6$，$a = 2$ 是一个本原根，计算 a 的各次幂得：

$$2^0 = 1 \qquad 2^4 = 7$$
$$2^1 = 2 \qquad 2^5 = 5 \pmod 9$$
$$2^2 = 4 \qquad 2^6 = 1$$
$$2^3 = 8$$

因此，对给定的本原根 $a = 2$（模 9）的指标，有下列余数表：

指标	0	1	2	3	4	5
数	1	2	4	8	7	5

给定一个数，要得到其指标，我们可以重排上表，使与 9 互素的余数作为主索引项：

数	1	2	4	5	7	8
指标	0	1	2	5	4	3

由于

$$x = a^{\text{ind}_{a,p}(x)} \bmod p \qquad y = a^{\text{ind}_{a,p}(y)} \bmod p$$

$$xy = a^{\text{ind}_{a,p}(xy)} \bmod p$$

由模乘法的性质有：

$$xy \bmod p = (x \bmod p)(y \bmod p)$$

$$a^{\text{ind}_{a,p}(xy)} \bmod p = (a^{\text{ind}_{a,p}(x)} \bmod p)(a^{\text{ind}_{a,p}(y)} \bmod p)$$

$$= (a^{\text{ind}_{a,p}(x)+\text{ind}_{a,p}(y)}) \bmod p$$

根据 Euler 定理，对任何互素的 a 和 n，有：

$$a^{\phi(n)} \equiv 1 \bmod n$$

任何正整数 z 可表示为 $z = q + k\phi(n)$，其中 $0 \leqslant q < \phi(n)$，所以由 Euler 定理有：

$$a^z \equiv a^q \bmod n \quad \text{如果 } z = q \bmod \phi(n)$$

将其代入前面的等式，则有：

$$\text{ind}_{a,p}(xy) \equiv [\text{ind}_{a,p}(x) + \text{ind}_{a,p}(y)] \bmod \phi(p)$$

由此可得：

$$\text{ind}_{a,p}(y^r) \equiv [r \times \text{ind}_{a,p}(y)] \bmod \phi(p)$$

这说明了普通对数和指标之间的相似性。由于这一原因，指标通常称为离散对数。

注意，仅当 a 是 m 的本原根时，才存在惟一的、以 a 为底的模 m 的离散对数。

表 8.4 可直接由表 7.6 推出，它列出了模 19 的离散对数集。

表 8.4 模 19 的离散对数表

(a) 以 2 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) 以 3 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) 以 10 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) 以 13 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) 以 14 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	14	9

(f) 以 15 为底，模 19 的离散对数

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\text{ind}_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	12	9

8.5.3　离散对数的计算

考虑方程

$$y = g^x \bmod p$$

对给定的 g，x 和 p，可直接计算出 y。在最坏情况下需执行 x 次乘法，并且存在计算 y 的有效算法。

但是，给定 y，g 和 p，计算 x 一般非常困难(即求离散对数)，其难度与 RSA 中因子分解素数之积的难度具有相同的数量级。在本书写作时，已知最快求模数为素数的离散对数的异步算法的难度为[BETH91]：

$$e^{((\ln p)^{1/3}(\ln(\ln p))^{2/3})}$$

对大素数而言，该算法是不可行的。

8.6　推荐读物和网址

有许多教材都非常详细地讨论了数论。[ORE67]是一本有用的初等数论的入门书籍；若读者要更深入地学习数论，可参阅[KUMA98]和[ROSE00]这两本优秀教材；另外，[LEVE90]也详细介绍了数论的有关知识。以上这些教材都附有习题及其解答，特别适合于自学。

[BURN97]包含许多习题并附有解答，有时间的读者可通过学习[BURN97]并完成这些习题逐步掌握数论中的基本概念，并牢固掌握数论的基本知识。

BURN97　Burn, R. *A Pathway to Number Theory*. Cambridge, England: Cambridge University Press, 1997.

KUMA98　Kumanduri, R., and Romero, C. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

LEVE90　Leveque, W. *Elementary Theory of Numbers*. New York: Dover, 1990.

ORE67　Ore, O. *Invitation to Number Theory*. Washington, DC: The Mathematical Association of America, 1967.

ROSE00　Rosen, K. *Elementary Number Theory and Its Applications*. Reading, MA: Addison-Wesley, 2000.

推荐网址：

- **The Prime Pages**：有关素数的研究及相关资料。

8.7 关键术语、思考题和习题

8.7.1 关键术语

双射	Euler 定理	阶
合数	Euler 函数	素数
中国剩余定理	Fermat 定理	本原根
离散对数	指标	

8.7.2 思考题

8.1 什么是素数?

8.2 表达式 a 整除 b 的意义是什么?

8.3 什么是 Euler 函数?

8.4 Miller-Rabin 测试可确定一个数不是素数,但不能确定一个数是素数。该算法是如何用来进行素性测试的?

8.5 一个数的本原根是什么?

8.6 指标和离散对数的区别是什么?

8.7.3 习题

8.1 本题的目的是要证明两个随机数互素的概率约为 0.6。

a. 令 $P = \Pr[\gcd(a,b) = 1]$。证明 $\Pr[\gcd(a,b) = d] = P/d^2$。提示:请考虑 $\gcd(a/d, b/d)$。

b. 对所有可能的 d,(a)中结果之和为 1,即:

$$\sum_{d \geq 1} \Pr[\gcd(a,b) = d] = 1$$

利用上述等式确定 P 的值。提示:请利用恒等式

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$$

8.2 对两个连续整数 n 和 $n+1$,为什么 $\gcd(n, n+1) = 1$?

8.3 利用 Fermat 定理计算 $3^{201} \bmod 11$。

8.4 在表 8.2 中,$n > 2$ 时 $\phi(n)$ 是偶数。这一点对所有 $n > 2$ 都成立。请证明之。

8.5 证明,若 p 是素数,则 $\phi(p^i) = p^i - p^{i-1}$。提示:请考虑哪些数与 p^i 有公因子。

8.6 可以证明(参见任何有关数论的书),若 $\gcd(m,n) = 1$,则 $\phi(mn) = \phi(m)\phi(n)$。利用该性质和前面习题中的有关性质,以及对素数 p,$\phi(p) = p - 1$ 这些性质,对任何 n 可直接取得 $\phi(n)$ 的值。请计算:

a. $\phi(41)$ **b.** $\phi(27)$ **c.** $\phi(231)$ **d.** $\phi(440)$

8.7 虽然中国古代数学家提出了中国剩余定理,取得了很好的结果,但是他们并不总是

正确的。他们提出的素性测试方法断言：n 是素数当且仅当 n 能整除 $(2^n - 2)$。

a. 给出一个满足上述条件的奇素数。

b. $n = 2$ 时条件显然为真。证明 n 是奇素数时条件也为真(即证明充分条件)。

c. 给出一个奇数但不是素数，它不满足上述条件。该非素数可能非常大，这也是导致中国数学家误以为条件为真时 n 是素数的原因。

d. 遗憾的是，中国古代数学家并没有测试 $n = 341$ 的情形。341 不是素数($341 = 11 \times 31$)但能被 $2^{341} - 2$ 整除。证明 $2^{341} \equiv 2 \bmod 341$(即必要条件不成立)。提示：可利用同余定理证明之，而不必计算 2^{341}。

8.8 证明：若 n 是奇合数，则对 $a = 1$ 和 $a = (n - 1)$，Miller-Rabin 测试将返回“不确定”。

8.9 若 n 是合数，且对底 a 通过了 Miller-Rabin 测试，则 n 称为对底 a 的强伪素数。证明 2047 是对底 2 的强伪素数。

8.10 中国剩余定理的常用表示形式为：令 $m_1, \cdots, m_k$ 是两两互素的整数，且 $1 \leqslant i, j \leqslant k$，且 $i \neq j$。定义 M 为所有 m_i 的乘积。设 $a_1, \cdots, a_k$ 是整数，则同余方程组：

$$\begin{aligned} x &\equiv a_1 (\bmod m_1) \\ x &\equiv a_2 (\bmod m_2) \\ &\vdots \\ x &\equiv a_k (\bmod m_k) \end{aligned}$$

有模 M 惟一解。证明定理的这种表示形式是正确的。

8.11 下面是孙子用来说明 CRT 的一个例子：

$$x \equiv 2 \pmod 3; \quad x \equiv 3 \pmod 5; \quad x \equiv 2 \pmod 7$$

试求解 x。

8.12 六位教授分别在周一至周六开始授课，且分别每隔 2，3，4，1，6 和 5 天授一次课，该大学禁止周日上课(所以周日的课必须停止)。什么时候所有六位教授首次发现必须停课？提示：利用 CRT。

8.13 找出 25 的所有本原根。

8.14 给定 29 的本原根 2，构造指标表，并利用该表解下列同余方程：

a. $17x^2 \equiv 10 \bmod 29$

b. $x^2 - 4x - 16 \equiv 0 \bmod 29$

c. $x^7 \equiv 17 \bmod 29$

第9章　公钥密码学与RSA

公钥密码学的发展是整个密码学发展历史中最伟大的一次革命，也许可以说是惟一的一次革命。从密码学产生至今，几乎所有的密码体制都是基于替换和置换这些初等方法。几千年来，对算法的研究主要是通过手工计算来完成的。随着转轮加密/解密机器的出现，传统密码学有了很大进展，利用电子机械转轮可以开发出极其复杂的加密系统，利用计算机甚至可以设计出更加复杂的系统，最著名的例子是 Lucifer 在 IBM 实现数据加密标准(DES)时所设计的系统。转轮机和 DES 是密码学发展的重要标志，但它们都基于替换和置换这些初等方法之上。

公钥密码学与其前的密码学完全不同。首先，公钥算法基于数学函数而不基于替换和置换。更重要的是，与只使用一个密钥的对称传统密码不同，公钥密码学是非对称的，它使用两个独立的密钥。我们将会看到，使用两个密钥在消息的保密性、密钥分配和认证领域有着重要意义。

下面我们先讨论有关公钥密码的几种常见的误解。一种误解是，从密码分析的角度看，公钥密码比传统密码更安全。例如，Gardner 在“Scientific American”上发表的一篇著名的文章中[GARD77]就提出了这样一种观点。事实上，任何加密方法的安全性依赖于密钥的长度和破译密文所需要的计算量。从抗密码分析的角度看，原则上不能说传统密码优于公钥密码，也不能说公钥密码优于传统密码。

第二种误解是，公钥密码是一种通用的方法，所以传统密码已经过时。其实正相反，由于现有的公钥密码方法所需的计算量大，所以取缔传统密码似乎不太可能。就像公钥密码的发明者之一所说的[DIFF88]，“公钥密码学仅限于用在密钥管理和签名这类应用中，这几乎是已被广泛接受的事实。”

最后一种误解是，传统密码中与密钥分配中心的握手是一件异常麻烦的事情，与之相比，用公钥密码实现密钥分配则非常简单。事实上，使用公钥密码也需要某种形式的协议，该协议通常包含一个中心代理，并且它所包含的处理过程既不比传统密码中的那些过程更简单，也不比之更有效。

本章简要介绍公钥密码。我们先介绍其中的基本概念。有趣的是，这些概念在用于公钥密码中之前就已经提出；然后我们讨论 RSA 算法，它是公钥密码中最重要的一种切实可行的加密/解密算法；第 10 章将进一步讨论公钥加密。

很多公钥密码体制的理论都基于数论。如果你接受本章中给出的结论，则可不必理解数论的有关知识。但是要完全理解公钥算法，则需要理解这些数论知识。第 8 章中概要介绍了数论的有关知识。

9.1　公钥密码体制的基本原理

公钥密码学的概念是为了解决传统密码中最困难的两个问题而提出的。第一个问题是在

第 7 章中已经详细讨论过的密钥分配问题。

我们知道，用对称密码进行密钥分配要求(1)通信双方已经共享一个密钥，而该密钥已通过某种方法分配给通信双方；或者(2)利用密钥分配中心。公钥密码的发明人之一 Whitfield Diffie(和 Martin Hellman 当时同在斯坦福大学工作)认为，第二个要求有悖于密码学的精髓，即应在通信过程中完全保持保密性。正如 Diffie 所说[DIFF88]，"如果必须要求用户与 KDC 共享他们的密钥，这些密钥可能因为盗窃或索取而被泄密，那么设计不可破的密码体制究竟还有什么意义呢?"

Diffie 考虑的第二个问题是"数字签名"问题，该问题显然与第一个问题无关。如果密码学不是仅仅用于军事上，而是广泛用于商业或个人目的，那么像手写签名一样，电子消息和文件也需要签名，也就是说，能否设计一种方法，它能确保数字签名是出自某特定的人，并且各方对此均无异议？对数字签名的要求比对认证的要求更为广泛，在第 13 章中我们将讨论数字签名的特性及其相关的问题。

1976 年 Diffie 和 Hellman 针对上述两个问题提出了一种方法，这种方法与以前四千多年来密码学中的所有方法有着根本区别[①]，它是密码学中一个惊人的成就[DIFF76 a,b]。

在下一小节中，我们先讨论公钥密码学的基本框架，然后我们讨论加密/解密算法应满足的一些条件，这些条件是公钥体制的核心内容。

9.1.1　公钥密码体制

公钥算法依赖于一个加密密钥和一个与之相关的不同的解密密钥，这些算法都具有下述重要特点：

- 仅根据密码算法和加密密钥来确定解密密钥在计算上是不可行的。

另外，有些算法(如 RSA)还有以下特点：

- 两个密钥中的任何一个都可用来加密，另一个用来解密。

公钥密码体制有 6 个组成部分[图 9.1(a)；请对照图 2.1]：

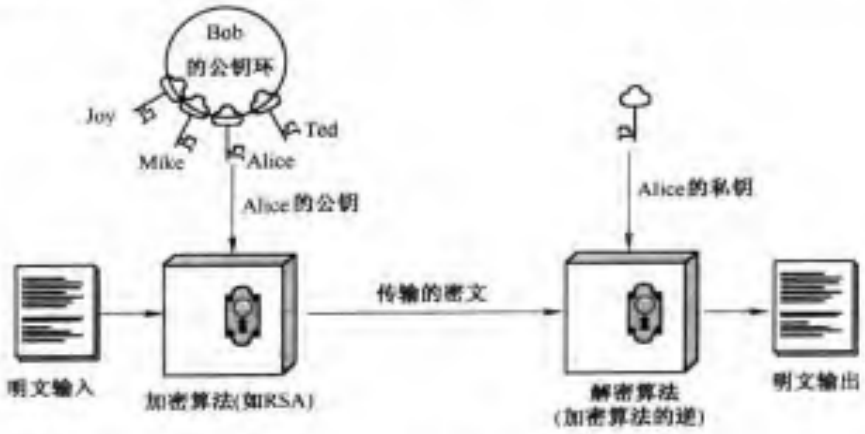
- **明文**：算法的输入。它们是可读信息或数据。
- **加密算法**：加密算法对明文进行各种转换。
- **公钥和私钥**：算法的输入。这对密钥中一个用于加密，一个用子解密。加密算法执行的变换依赖于公钥和私钥。
- **密文**：算法的输出。它依赖于明文和密钥，对给定的消息，不同的密钥产生的密文不同。
- **解密算法**：该算法接收密文和相应的密钥，并产生原始的明文。

其主要步骤如下：

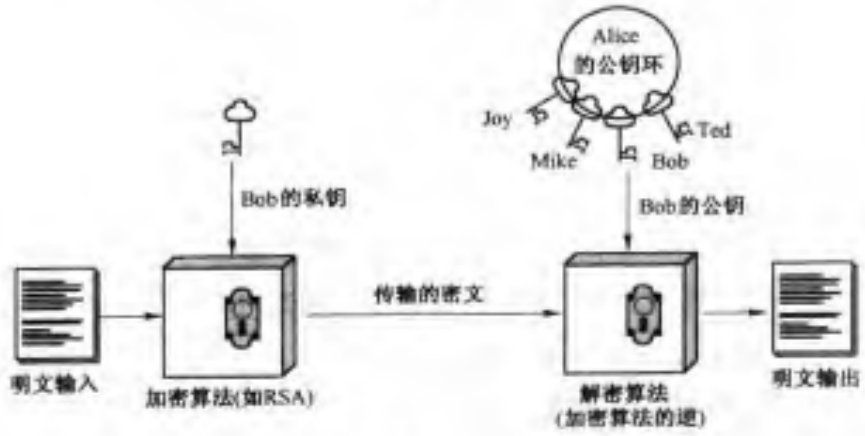
---

① Diffie 和 Hellman 于 1976 年首次公开提出了公钥密码学的概念，但是公钥密码学的概念并不是 1976 年才出现的，英国国家安全局(NSA)的负责人 Admiral Bobby Inman 声称 NSA 于 20 世纪 60 年代中期就已提出了公钥密码学[SIMM93]。这些概念的最早文字记录是在 James Ellis 于 1970 年所做的一份机密报告中[ELLI70]，这些概念出自与 NSA 类似的英国机构，即通信电子安全研究组(The Communications-Electronics Security Group)，Ellis 称这种技术为非保密的加密方法，并在[ELLI99]中描述了这一发现。

1. 每一用户产生一对密钥，用来加密和解密消息。
2. 每一用户将其中一个密钥存于公开的寄存器或其他可访问的文件中，该密钥称为公钥，另一密钥是私有的。如图 9.1(a)所示，每一用户可以拥有若干其他用户的公钥。
3. 若 Bob 要发消息给 Alice，则 Bob 用 Alice 的公钥对消息加密。
4. Alice 收到消息后，用其私钥对消息解密。由于只有 Alice 知道其自身的私钥，所以其他的接收者均不能解密出消息。



(a) 加密



(b) 认证

图 9.1 公钥密码体制

利用这种方法，通信各方均可访问公钥，而私钥是各通信方在本地产生的，所以不必进行分配。只要系统控制了其私钥，那么它的通信是安全的。在任何时刻，系统可以改变其私钥，并公布相应的公钥以替代原来的公钥。

表 9.1 总结了对称密码和公钥密码的一些重要特征。为了区分二者，我们一般将对称密

码中使用的密钥称为**秘密钥**，将公钥密码中使用的两个密钥分别称为**公钥**和**私钥**[①]。私钥总是保密的，但是为了避免与对称密码中的密钥混淆，我们将之称为私钥而不是秘密钥。

表 9.1　对称密码和公钥密码

对称密码	公钥密码
一般要求	**一般要求**
1. 加密和解密使用相同的密钥	1. 同一算法用于加密和解密，但加密和解密使用不同密钥
2. 收发双方必须共享密钥	2. 发送方拥有加密或解密密钥，而接收方拥有另一密钥
安全性要求	**安全性要求**
1. 密钥必须是保密的	1. 两个密钥之一必须是保密的
2. 若没有其他信息，则解密消息是不可能或至少是不可行的	2. 若没有其他信息，则解密消息是不可能或至少是不可行的
3. 知道算法和若干密文不足以确定密钥	3. 知道算法和其中一个密钥以及若干密文不足以确定另一密钥

下面我们利用图 9.2(比较图 2.2)进一步分析公钥密码体制的基本要素。消息源 A 产生明文消息 $X=[X_1, X_2, \cdots, X_M]$，其中 X 的 M 个元素是某有穷字母表上的字符，A 欲将消息 X 发送给 B。B 产生公钥 KU_b 和私钥 KR_b，其中只有 B 知道 KR_b，而 KU_b 则是公开可访问的，所以 A 也可访问 KU_b。

对作为输入的消息 X 和加密密钥 KU_b，A 生成密文 $Y=[Y_1, Y_2, \cdots, Y_N]$：

$$Y = \mathrm{E}_{KU_b}(X)$$

期望的接收方因为拥有相应的私钥，所以可进行逆变换：

$$X = \mathrm{D}_{KR_b}(Y)$$

攻击者可观察到 Y 并且可访问 KU_b，但是他不能访问 KR_b 或者 X，所以攻击者肯定会想方设法恢复 X 和/或 KR_b。假定攻击者知道加密(E)和解密(D)算法，如果他只关心 X 这一条消息，那么他会集中精力试图通过产生明文估计值 $\hat{X}$ 来恢复 X。但是，通常攻击者也希望能获得其他消息，所以他会通过产生估计值 $\hat{KR}_b$ 来试图恢复 KR_b。

前面曾提到过，两个密钥中的任何一个都可用来加密，而另一个用来解密。这样，我们可利用公钥密码实现其他一些功能。图 9.2 所示的方法可提供保密性，而图 9.1(b)和图 9.3 说明公钥密码可用于认证：

$$Y = \mathrm{E}_{KR_a}(X)$$

$$X = \mathrm{D}_{KU_a}(Y)$$

在这种方法中，A 向 B 发送消息前，先用 A 的私钥对消息加密，B 则用 A 的公钥对消息解密。由于是用 A 的私钥对消息加密，所以只有 A 可加密消息，因此，整个加密后的消息就是数字签名。除此之外，因为只有拥有 A 的私钥才能产生上述加密后的消息，因此该消息可用于认证源和数据完整性。

上述方法是对整条消息加密，尽管这种方法可以验证发送方和消息的有效性，但却需要大

---

① 我们将使用下述记号。秘密钥用 K_m 表示，其中 m 是修饰符；例如，K_s 表示会话密钥，用户 A 的公钥用 KU_a 表示，其相应的私钥用 KR_a 表示。分别用 $E_{K_m}[P]$、$E_{KU_a}[P]$和 $E_{KR_a}[P]$表示用秘密钥、公钥和私钥对明文 P 加密。同样，分别用 $D_{K_m}[C]$、$D_{KU_a}[C]$和 $D_{KR_a}[C]$表示用秘密钥、公钥和私钥对密文 C 解密。

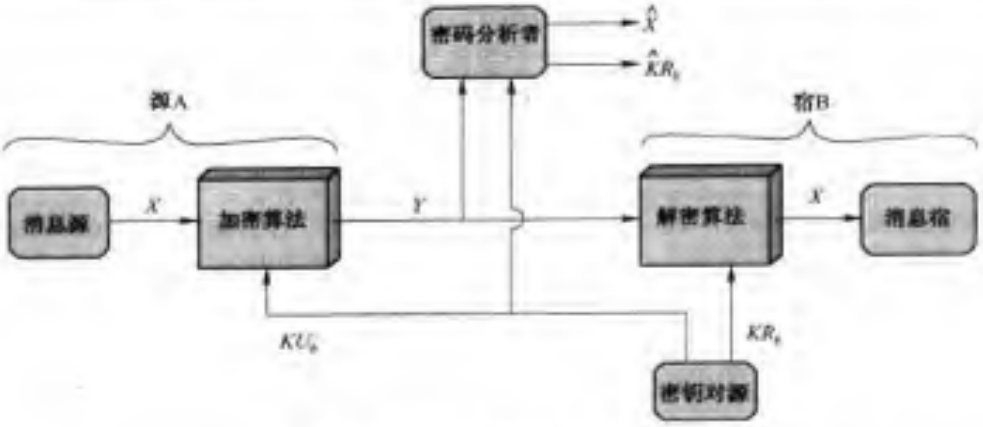


图 9.2 公钥密码体制:保密性

量的存储空间。在实际使用中,每个文件必须既要以明文形式保存,又要以密文形式保存,以便在发生争执时可以验证源及其发送的消息。解决这个问题的更有效途径是,只对一个称为认证符的小数据块加密,它是该消息的函数,对该消息的任何修改必然会引起认证符的变化。如果用发送方的私钥对认证符加密,那么加密的结果就可作为数字签名,它能验证源、消息和通信序列的有效性。我们将在第 13 章中详细讨论这种方法。

需要强调指出的是,上述加密过程不能保证消息的保密性。也就是说,它可以防止发送的消息被修改,但不能防止被搭线窃听。在基于对部分消息签名方法中,因为消息的其余部分是以明文形式传输的,所以这种方法显然不能保证保密性。由于任何人都可以用发送方的公钥对消息解密,所以即使用图 9.3 所示的方法对整条消息加密,也不能保证被发送消息的保密性。

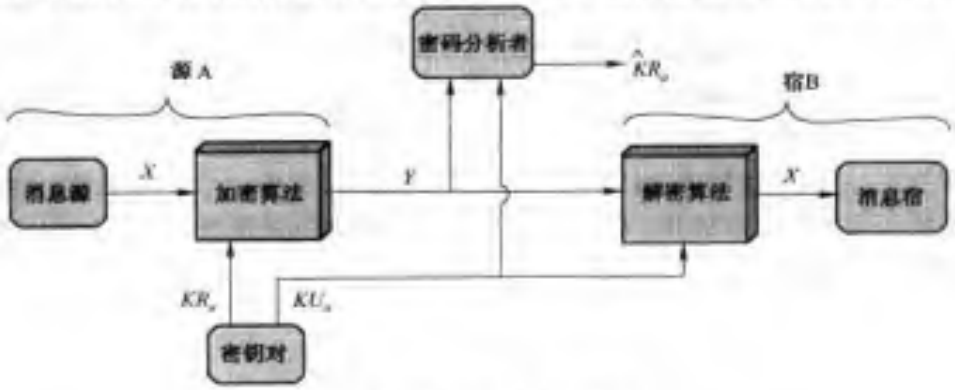


图 9.3 公钥密码体制:认证

但是,如果两次使用公钥方法,则既可提供认证功能,又可保证被发送消息的保密性(见图 9.4):

$$Z = \mathrm{E}_{KU_b}[\mathrm{E}_{KR_a}(X)]$$
$$X = \mathrm{D}_{KU_a}[\mathrm{D}_{KR_b}(Z)]$$

在这种方法中,发送方首先用其私钥对消息加密,得到数字签名,然后再用接收方的公钥加密,所得的密文只能被拥有相应私钥的接收方解密,这样可保证消息的保密性。但这种方法

的缺点是，在每次通信中要执行四次复杂的公钥算法而不是两次。

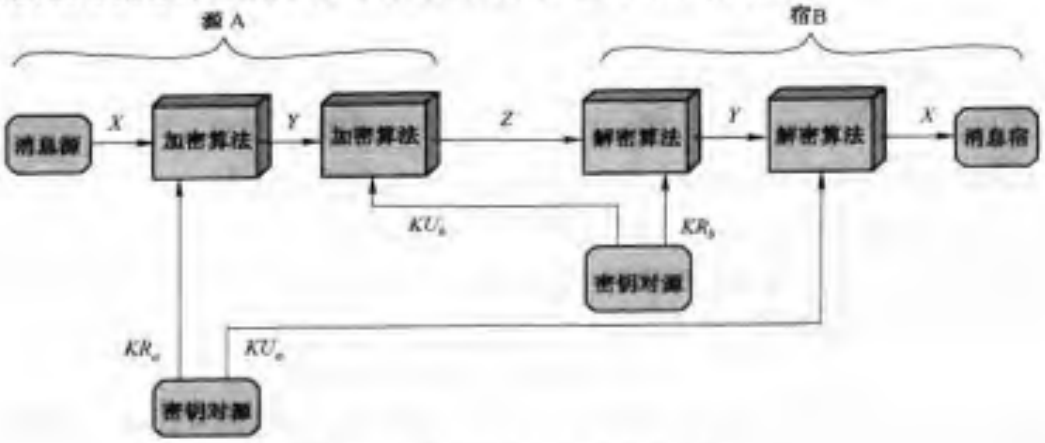


图 9.4　公钥密码体制：保密性和认性

9.1.2　公钥密码体制的应用

我们首先必须澄清公钥密码体制中容易引起混淆的一个问题。公钥密码体制的特点是使用具有两个密钥的密码算法，其中一个密钥是私有的，另一个是公有的。根据不同的应用，发送方可使用其私钥或者接收方的公钥或同时使用二者来执行密码功能。一般地，公钥密码体制的应用可分为三类：

- **加密/解密**：发送方用接收方的公钥对消息加密。
- **数字签名**：发送方用其私钥对消息"签名"。签名可以通过对整条消息加密或者对消息的一个小的数据块加密来产生，其中该小数据块是整条消息的函数。
- **密钥交换**：通信双方交换会话密钥。有几种不同的方法可用于密钥交换，这些方法都使用了通信一方或双方的私钥。

有些算法可用于上述三种应用，而其他一些算法则只适用其中一种或两种应用。表 9.2 列出了本书中所讨论的算法及其所支持的应用。

表 9.2　公钥密码体制的应用

算　法	加密/解密	数字签名	密钥交换
RSA	是	是	是
椭圆曲线	是	是	是
Diffie-Hellman	否	否	是
DSS	否	是	否

9.1.3　对公钥密码的要求

图 9.2 至图 9.4 所示的密码体制建立在基于两个相关密钥的密码算法之上。Diffie 和 Hellman 假定这一体制是存在的，但没有证明这种算法的存在性，不过他们给出了这些算法应满足的条件[DIFF76b]：

1. B 产生一对密钥(公钥 KU_b，私钥 KR_b)在计算上是容易的。

2. 已知公钥和要加密的消息 M，发送方 A 产生相应的密文在计算上是容易的：

$$C = \mathrm{E}_{KU_b}(M)$$

3. 接收方 B 使用其私钥对接收的密文解密以恢复明文在计算上是容易的：

$$M = \mathrm{D}_{KR_b}(C) = \mathrm{D}_{KR_b}[\mathrm{E}_{KU_b}(M)]$$

4. 已知公钥 KU_b 时，攻击者要确定私钥 KR_b 在计算上是不可行的。

5. 已知公钥 KU_b 和密文 C，攻击者要恢复明文 M 在计算上是不可行的。

我们还可以增加一个条件，尽管该推荐很有用，但并不是所有的公钥密码应用都必须满足该条件：

6. 加密和解密函数的顺序可以交换：

$$M = \mathrm{E}_{KU_b}[\mathrm{D}_{KR_b}(M)] = \mathrm{D}_{KU_b}[\mathrm{E}_{KR_b}(M)]$$

在公钥密码学概念提出后的几十年中，只有两个满足这些条件的算法（RSA，椭圆曲线密码体制）为人们普遍接受，这一事实表明要满足上述条件是不容易的。

下面我们先对上述条件做进一步分析，然后再详细说明为什么这些条件很难满足。事实上，要满足上述条件即是要找一个单向陷门函数。单向函数[①] 是满足下列性质的函数：每个函数值都存在惟一的逆，并且计算函数值是容易的，但求逆却是不可行的：

$$Y = \mathrm{f}(X) \qquad \text{容易}$$

$$X = \mathrm{f}^{-1}(Y) \qquad \text{不可行}$$

通常，“容易”是指一个问题可以在输入长度的多项式时间内得到解决，即若输入长度为 n 位，则计算函数值的时间与 n^a 成正比，其中 a 是一个固定的常数，这样的算法称为 P 类算法。“不可行”的定义比较模糊，一般而言，若解决一个问题所需的时间比输入规模的多项式时间增长更快，则称该问题是不可行的。例如，若输入长度是 n 位，计算函数的时间与 2^n 成比，则认为是不可行的。遗憾的是，我们很难确定算法是否具有这种复杂性。另外，传统的计算复杂性注重于算法的最坏情况或平均情况复杂性，但是最坏情况复杂性和平均情况复杂性这种方法不适用于密码学，因为密码学要求对任何输入都不能求出函数的逆，而不是在最坏情况或平均情况下不能求出函数的逆。附录 9A 简要介绍了上述有关概念。

下面给出单向陷门函数的定义。一个函数，若计算函数值很容易，并且在缺少一些附加信息时计算函数的逆是不可行的，但是已知这些附加信息时，可在多项式时间内计算出函数的逆，那么我们称这样的函数为单向陷门函数，即单向陷门函数是满足下列条件的一类不可逆函数 f_k：

若 k 和 X 已知，则容易计算 $Y = \mathrm{f}_k(X)$

若 k 和 Y 已知，则容易计算 $X = \mathrm{f}_k^{-1}(Y)$

若 Y 已知但 k 未知，则计算出 $X = \mathrm{f}_k^{-1}(Y)$ 是不可行的

由此可见，寻找合适的单向陷门函数是公钥密码体制应用的关键。

9.1.4 公钥密码分析

与对称密码一样，公钥密码也易受穷举攻击，其解决方法也是使用长密钥。但同时也应考

① 请不要将单向函数与单向 hash 函数混淆。单向 hash 函数的输入可以是任意长的数据，其输出是定长的，这些函数用于消息认证中（见第 11 章）。

虑使用长密钥的利弊，公钥体制使用的是某种可逆的数学函数，计算函数值的复杂性可能不是密钥长度的线性函数，而是比线性函数增长更快的函数，因此，为了抗穷举攻击，密钥必须足够长；同时为了便于实现加密和解密，密钥又必须足够短。在实际中，现已提出的密钥长度确实可以抗穷举攻击，但是它也使加/解密速度太慢，所以公钥密码目前仅限于密钥管理和签名中。

对公钥密码的另一种攻击方法是，找出一种从给定的公钥计算出私钥的方法。到目前为止，还未在数学上证明对一特定公钥算法这种攻击是不可行的，所以包括已被广泛使用的 RSA 在内的任何算法都是值得怀疑的。密码分析的历史表明，同一个问题从一个角度看是不可解的，但从另一个不同的角度来看则可能是可解的。

最后，还有一种攻击形式是公钥体制中所特有的，这种攻击本质上就是穷举消息攻击。例如，假定要发送的消息是 56 位的 DES 密钥，那么攻击者可以用公钥对所有可能的密钥加密，并与传送的密文匹配，从而可解密任何消息。因此，无论公钥体制的密钥有多长，这种攻击都可以转化为对 56 位密钥的穷举攻击。抗这种攻击的方法是，在要发送的消息后附加上一个随机数。

9.2　RSA 算法

Diffie 和 Hellman 在其早期的著名论文[DIFF76b]中提出了一种新的密码学方法，事实上，它对密码学家提出了一种挑战，即要去寻找满足公钥体制要求的密码算法。MIT 的 Ron Rivest、Adi Shamir 和 Len Adleman 于 1977 年提出并于 1978 年首次发表的算法[RIVE78][①]，可以说是最早提出的满足要求的公钥算法之一。Rivest-Shamir-Adleman(RSA)算法自其诞生之日起，就成为被广泛接受且被实现的通用公钥加密方法。

RSA 体制是一种分组密码，其明文和密文均是 0 至 $n-1$ 之间的整数，通常 n 的大小为 1024 位二进制数或 309 位十进制数。本节我们详细讨论 RSA 算法。首先我们给出该算法的描述，然后讨论 RSA 算法的计算问题和密码分析问题。

9.2.1　算法描述

Rivest、Shamir 和 Adleman 提出的算法使用了乘方运算。明文以分组为单位进行加密，每个分组的二进制值均小于 n。也就是说，分组的大小必须小于或等于 $\log_2(n)$位；在实际应用中，分组的大小是 k 位，其中 $2^k < n \leqslant 2^{k+1}$。对明文分组 M 和密文分组 C，加密和解密过程如下：

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

其中收发双方均已知 n，发送方已知 e，只有接收方已知 d，因此公钥加密算法的公钥为 $KU = \{e, n\}$，私钥为 $KR = \{d, n\}$。该算法要能用做公钥加密，必须满足下列条件：

1. 可以找到 e，d 和 n，使得对所有 $M < n$，有 $M^{ed} = M \bmod n$。
2. 对所有 $M < n$，计算 M^e 和 C^d 是比较容易的。

---

① 很明显。加密/解密的首个可用公钥系统由英国 CESG 的 Clifford Cocks 于 1973 年提出[COCK73]；Cocks 的方法几乎与 RSA 相同。

3. 由 e 和 n 确定 d 是不可行的。

我们先讨论第一个问题，其他问题将在以后讨论。我们需要找出下列关系式：

$$M^{ed} = M \bmod n$$

由第 8 章中给出的 Euler 定理的推论[等式(8.6)]可知，给定素数 p 和 q，整数 n 和 m，以及整数 k，其中 $n = pq$，$0 < m < n$，则有下列关系式成立：

$$m^{k\phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \bmod n$$

其中 $\phi(n)$ 是 Euler 函数，即小于 n 且与 n 互素的正整数的个数。在第 8 章中已经证明，对素数 p 和 q，有 $\phi(pq) = (p-1)(q-1)$。因此若

$$ed = k\phi(n) + 1$$

则我们可得到期望的关系式。上式等价于：

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

也就是说，d 和 e 是模 $\phi(n)$ 的乘法逆元。根据模算术的性质，仅当 d 与 $\phi(n)$ 互素[因此 e 也与 $\phi(n)$ 互素]，即 $\gcd(\phi(n), d) = 1$ 时，d 和 e 是模 $\phi(n)$ 的乘法逆元。

下面我们介绍 RSA 算法，该算法中用到下列元素：

两个素数 p, q	(保密的，选定的)
$n = pq$	(公开的，计算得出的)
e, $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(公开的，选定的)
$d \equiv e^{-1} \bmod \phi(n)$	(保密的，计算得出的)

这里，私钥为 $\{d, n\}$，公钥为 $\{e, n\}$。假定用户 A 已公布了其公钥，用户 B 要发送消息 M 给 A，那么用户 B 计算 $C = M^e \pmod n$，并发送 C；在接收端，用户 A 计算 $M = C^d \pmod n$ 以解密出消息 M。

上述算法是正确的，因为所选择的 e 和 d 满足：

$$d \equiv e^{-1} \bmod \phi(n)$$

所以

$$ed \equiv 1 \bmod \phi(n)$$

因此，ed 是形为 $k\phi(n) + 1$ 的公式。根据第 8 章中 Euler 定理的推论，对给定的素数 p 和 q，整数 $n = pq$ 和 M，其中 $0 < M < n$：

$$M^{k\phi(n)+1} = M^{k(p-1)(q-1)+1} \equiv M \bmod n$$

所以 $M^{ed} \equiv M \bmod n$。这样我们有：

$$C = M^e \bmod n$$
$$M = C^d \bmod n \equiv (M^e)^d \bmod n \equiv M^{ed} \bmod n \equiv M \bmod n$$

图 9.5 归纳总结了 RSA 算法。图 9.6 所示的是[SING99]中给出的一个例子。其密钥产生过程如下：

1. 选择两个素数，$p = 7$ 和 $q = 11$。

2. 计算 $n = pq = 17 \times 11 = 187$。
3. 计算$\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$。
4. 选择 e 使其与$\phi(n) = 160$ 互素且小于$\phi(n)$；这里选择 $e = 7$。
5. 确定 d 使得 $de = 1 \bmod 160$ 且 $d < 160$。因为 $23 \times 7 = 161 = 10 \times 160 + 1$，所以 $d = 23$。d 可利用扩展的 Euclid 算法来计算（第 4 章）。

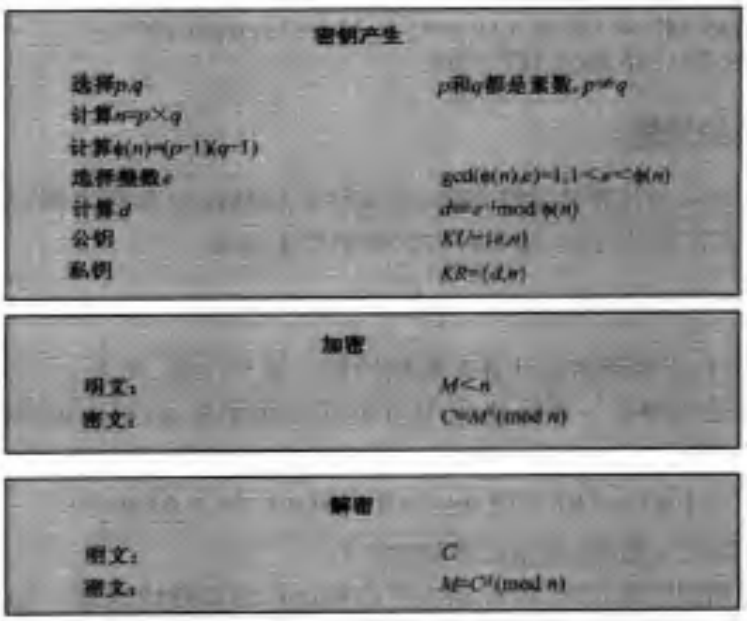


图 9.5　RSA 算法

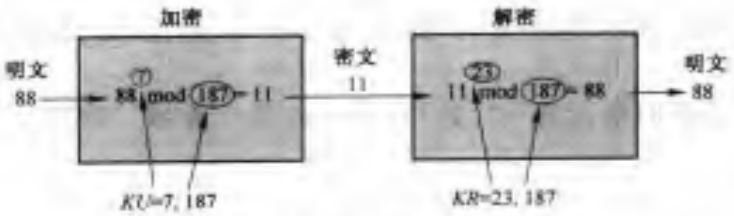


图 9.6　RSA 算法举例

所得的公钥 $KU = |7, 187|$，私钥 $KR = |23, 187|$。该例说明了输入明文 $M = 88$ 时这些密钥的使用情况。加密时，需计算 $C = 88^7 \bmod 187$。利用模算术的性质，我们如下计算：

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59\,969\,536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894\,432 \bmod 187 = 11$$

解密时，我们计算 $M = 11^{23} \bmod 187$：

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14\,641 \bmod 187 = 55$$
$$11^8 \bmod 187 = 214\,358\,881 \bmod 187 = 33$$
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79\,720\,245 \bmod 187 = 88$$

9.2.2 计算方面的问题

下面我们讨论 RSA 的计算复杂性问题，它实际上包括两方面的问题：密钥产生和加密/解密。我们先讨论加密和解密过程，然后再讨论密钥产生问题。

加密和解密

在 RSA 中，加密和解密都需要计算某整数的模 n 整数次幂，如果先求出整数的幂，再对 n 取模，那么中间结果会非常大。幸运的是，正如前面的例子所示，我们可利用模算术的下列性质来进行模幂运算：

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

这样我们将中间结果对 n 取模，使得计算切实可行。

因为 RSA 中所用到的幂很大，所以还应考虑幂运算的有效性问题。为说明如何增强有效性，以计算 x^{16} 为例。若直接计算则需进行 15 次乘法：

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

如果重复计算每个中间结果的平方，得到 x^2，x^4，x^8 和 x^{16}，那么只需 4 次乘法即可计算出 x^{16}。

更一般地，假定我们要计算 a^m，其中 a 和 m 是正整数。若将 m 表示为二进制数 $b_k b_{k-1} \cdots b_0$，则有：

$$m = \sum_{b_i \neq 0} 2^i$$

所以

$$a^m = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i \neq 0} [a^{(2^i)} \bmod n]\right) \bmod n$$

下面我们讨论计算 $a^b \bmod n$ 的算法[①]，如图 9.7 所示。图 9.8 举例说明了该算法的执行过程。请注意，这里的变量 c 不是必需的，引入它只是为了便于解释算法，c 的终值即是幂的值。

① 该算法已有较长的历史；这个特殊的伪码表达式摘自[CORM01]。

```
c ← 0; d ← 1
for i ← k downto 0
    do  c ← 2 × c
        d ← (d × d) mod n
        if  b_i = 1
            then  c ← c + 1
                  d ← (d × a) mod n
return d
```

图 9.7　计算 $a^b \bmod n$ 的算法

密钥产生

在应用公钥密码进行通信之前，通信各方都必须产生一对密钥，即需做以下工作：

- 确定两个素数 p 和 q。
- 选择 e 或者 d，并计算 d 或者 e。

我们首先考虑 p 和 q 的选择问题。由于任何攻击者可以知道 $n = pq$，所以为了避免攻击者用穷举法求出 p 和 q，应该从足够大的集合中选取 p 和 q（即 p 和 q 必须是大素数）。另一方面，选择大素数的方法必须是有效的。

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

图 9.8　计算 $a^b \bmod n$ 的快速模幂算法，其中 $a = 7, b = 560 = 1000110000, n = 561$

目前还没有有效的方法可以产生任意大素数，因此需要使用其他方法来解决这一问题。通常使用的方法是，随机挑选一个期望大小的奇数，然后测试它是否是素数。若不是，则挑选下一个随机数直至检测到素数为止。

各种素性测试方法因此应运而生（如[KNUT98]中介绍了几种素性测试方法），它们几乎全都是概率测试方法；也就是说，这些测试只能确定一个给定的整数可能是素数。尽管存在这种不确定性，但是以某种方式执行这些测试可使得一个整数是素数的概率接近 1.0。例如第 8 章中介绍的 Miller-Rabin 算法就是非常有效且被广泛使用的素数测试方法。Miller-Rabin 算法及其他许多类似算法，测试一个给定的数 n 是否是素数的过程即是执行某种计算，这些计算涉及 n 和一个随机选择的整数 a。若 n“未通过”测试，则 n 不是素数；若 n 通过了测试，则 n 可能是素数，也可能不是素数。若对许多随机选择的不同 a，n 均能通过测试，则我们几乎可以相信 n 就是素数。

归纳起来，挑选素数的过程如下：

1. 随机选择一个奇整数 n（如利用伪随机数产生器）。
2. 随机选择一个整数 $a < n$。
3. 执行诸如 Miller-Rabin 之类的概率素数测试。若 n 未通过测试，则拒绝 n，并转到步骤 1。
4. 若 n 通过测试足够多次，则接受 n；否则转到步骤 2。

这个过程有些繁琐，但是，一般不会很频繁地执行这个过程，因为只有在需要一对新的密钥(KU, KR)时才会执行它。

值得注意的是，在找到一个素数之前可能有多少个整数会被拒绝。由数论中的素数定理可知，在 N 附近平均每隔 ($\ln N$) 个整数就有一个素数，这样在找到一个素数之前，平均要测试大约 $\ln(N)$ 个整数。由于每个偶数会被立刻拒绝，所以实际上只需测试大约 $\ln(N)/2$ 个整数。例如，要找一个 2^{200} 左右的素数，则在找到素数之前大约要进行 $\ln(2^{200})/2 = 70$ 次尝试。

若确定了素数 p 和 q，则可通过选择 e 并计算 d 或者选择 d 并计算 e 来产生密钥。假定是前者，那么我们需要选择满足 $\gcd(\phi(n), e) = 1$ 的 e，并计算 $d \equiv e^{-1} \bmod \phi(n)$。幸运的是，存在求两个整数的最大公因子的算法，并且在它们的最大公因子为 1 时，该算法还能同时求出其中一数对另一数取模的逆元，这个算法称为扩展的 Euclid 算法，已在第 8 章中讨论过。由上可知，产生密钥的过程就是要产生若干个随机数，直至找到与 $\phi(n)$ 互素的数为止。我们再次提出这样一个问题：在找到一个可用的数，即与 $\phi(n)$ 互素的数之前，要测试多少个随机数呢？可以很容易地证明，两个随机数互素的概率约为 0.6，因此找到一个恰当的数之前只需进行非常少的测试(见习题 8.1)。

9.2.3 RSA 的安全性

对 RSA 算法的攻击可能有如下三种方式：

- **穷举攻击**：这种方法试图穷举所有可能的私钥。
- **数学攻击**：有多种数学攻击方法，它们的实质都是试图分解两个素数的乘积。
- **计时攻击**：这类方法依赖于解密算法的运行时间。

像其他的密码体制一样，RSA 抗穷举攻击的方法也是使用大密钥空间，所以 e 和 d 的位数越大越好，但是密钥产生过程和加/解密过程都包含复杂的计算，因此密钥越大，系统运行速度越慢。

在本小节中，我们简要介绍数学攻击和计时攻击。

因子分解问题

用数学方法攻击 RSA 的途径有以下三种：

- 分解 n 为两个素因子。这样就可以计算出 $\phi(n) = (p-1) \times (q-1)$，从而可以确定 $d = e^{-1} (\bmod \phi(n))$。
- 直接确定 $\phi(n)$而不先确定 p 和 q。这同样也可以确定 $d = e^{-1} (\bmod \phi(n))$。
- 直接确定 d，而不先确定 $\phi(n)$。

对 RSA 的密码分析的讨论大都集中于第一种攻击方法，即将 n 分解为两个素数因子。由给定的 n 来确定 $\phi(n)$等价于因子分解 n[RIBE96]。现在已知的、从 e 和 n 确定 d 的算法至少和因子分解问题一样费时[KALI95]。因此，我们通过将因子分解的性能作为基准来评价 RSA 的安全性。

尽管因子分解具有大素数因子的数 n 仍然是一个难题，但已不像以前那么困难。下面我们来看一个著名的例子。1977 年，RSA 的三位发明者让杂志“Scientific American”的读者对他们发表在 Martin Gardner 上“数学游戏”专栏中的密文进行解密，解得明文则可获得 100 美元奖金，

他们预言需要 4×10^{16} 年才能解得明文。但是，一个研究小组利用因特网只用了 8 个月的时间，于 1994 年 4 月解决了这个问题。这里所使用的公钥大小（n 的长度）是 129 位十进制数，即约 428 位二进制数。与对 DES 的算法一样，RSA 实验室同时还发布了用位数为 100，110 或 120 等的密钥加密的密文供有兴趣者解密。最近被解密的是 RSA-155，其密钥长度为 155 十进制位或 512 位。表 9.3 给出了迄今为止得出的一些结果。我们用 MIPS 年来描述计算的代价。MIPS 年是指一台每秒执行百万条指令的处理器运行一年，即执行约 3×10^{13} 条指令。一台 1 GHz的奔腾机约等于一台 250 MIPS 机器。

表 9.3　因子分解问题的进展情况

十进制位数	二进制位数（近似值）	完 成 日 期	MIPS 年	算　　法
100	332	1991 年 4 月	7	二次筛法
110	365	1992 年 4 月	75	二次筛法
120	398	1993 年 6 月	830	二次筛法
129	428	1994 年 4 月	5000	二次筛法
130	431	1996 年 4 月	1000	一般数域筛法
140	465	1999 年 2 月	2000	一般数域筛法
155	512	1999 年 8 月	8000	一般数域筛法

请注意表 9.3 所使用的因子分解方法。在 20 世纪 90 年代中期以前一直是用二次筛法来进行因子分解，对 RSA-130 的攻击使用了称为一般数域筛（GNFS）的新算法，该算法能够因子分解比 RSA-129 更大的数，但计算代价仅是二次筛法的 20%。

计算能力的不断增强和因子分解算法的不断改进，给大密钥的使用造成威胁。我们已了解到，更换一种算法可使速度显著增加。我们期望 GNFS 还可以进一步改进并能设计出更好的算法。事实上，对某种特殊形式的数，用特殊数域筛（SNFS）算法进行因子分解比用一般数域筛法要快得多，图 9.9 对这两种算法的性能进行了比较。我们可以期望算法上会有所突破，使一般的因子分解的性能在时间上大约与 SNFS 一样或者甚至比 SNFS 更快[ODLY95]。因此我们在选择 RSA 的密钥大小时应谨慎小心。在最近一段时间里，密钥大小取在 1024 到 2048 位是合适的。

除了要指定 n 的大小外，研究者还提出了其他一些限制条件。为了防止可以很容易地分解 n，RSA 算法的发明者建议 p 和 q 还应满足下列限制条件：

1. p 和 q 的长度应仅相差几位。这样对 1024 位（309 十进制位）的密钥而言，p 和 q 都应约在 10^{75} 到 10^{100} 之间。
2. $(p-1)$ 和 $(q-1)$ 都应有一个大的素因子。
3. $\gcd(p-1, q-1)$ 应该较小。

另外，已经证明，若 $e < n$ 且 $d < n^{1/4}$，则 d 很容易被确定[WIEN90]。

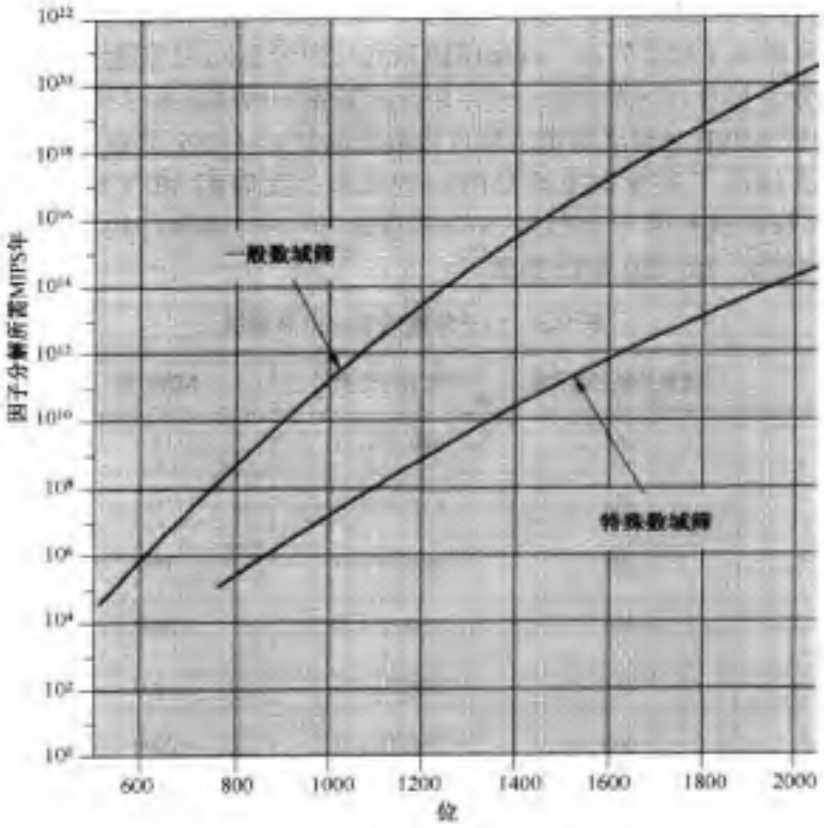


图 9.9 因子分解所需的 MIPS 年

计时攻击

如果你想知道评价密码算法的安全性有多难，那么计时攻击的出现就是最好的例子。密码学家 Paul Kocher 已证明，攻击者可以通过记录计算机解密消息所用的时间来确定私钥[KOCH96, KALI96b]。计时攻击不仅可用于攻击 RSA，而且可以用于攻击其他的公钥密码系统，由于这种攻击的完全不可预知性以及它仅依赖于明文，所以计时攻击具有很大的威胁。

计时攻击类似于窃贼通过观察他人转动保险柜拨号盘的时间长短来猜测密码，我们可以通过图 9.7 中的模幂算法来说明这种攻击，但这种攻击可以攻击任何运行时间可变的算法。图 9.7 的算法中，模幂运算是通过一位一位来实现的，每次迭代执行一次模乘运算，且若该位为 1，则还需执行一次模乘运算。

正如 Kocher 在其论文中所指出的，在下述极端情况下，我们很容易理解计时攻击的含义。假定在模幂算法中，模乘函数的执行时间只在几种情形中其执行时间比整个模幂运算的平均执行时间要长得多，但在大多情形下其执行速度相当快。计时攻击是从最左位 b_k 开始，一位一位地进行的。假设攻击者已知前面的 j 位(为了得到整个指数，攻击者可以从 $j=0$ 开始，重复攻击直至已知整个指数为止)，则对给定的密文，攻击者可以完成 **for** 循环的前 j 次迭代，其后的操作依赖于未知的指数位。若该位为 1，则要执行 $d \leftarrow (d \times a) \bmod n$。对有些 a 和 d 的值，模乘运算的执行速度异常慢，假定攻击者知道是哪些值。由于位为 1 时，对这些值执行迭

代的速度很慢，若攻击者观察到解密算法的执行总是很慢，则可认为该位为 1；若攻击者多次观察到整个算法的执行都很快，则可认为该位为 0。

在实际中，模幂运算的实现并没有这样大的时间差异，一次迭代的执行时间会超过整个算法的平均执行时间，但存在足够大的差异使得计时攻击切实可行。关于这个问题的详细讨论，请参见[KOCH96]。

尽管计时攻击会造成严重的威胁，但是有一些简单可行的解决方法，包括：

- **不变的幂运算时间**：保证所有的幂运算在返回结果前执行的时间都相同。这种方法虽然很简单，但会降低算法的性能。
- **随机延时**：通过在求幂算法中加入随机延时来迷惑计时攻击者可提高性能。Kocher 认为，如果不增加足够的干扰，那么攻击者可以通过收集额外的观察数据来抵消随机延时，仍然可能攻击成功。
- **隐蔽**：在执行幂运算之前先将密文乘上一个随机数，这一过程可使攻击者不知道计算机正在处理的是密文的哪些位，这样可防止攻击者一位一位地进行分析，而这种分析正是计时攻击的本质所在。

RSA 数据安全算法（RSA Data Security）在乘积中就使用了隐蔽方法，它用私钥实现操作 $M = C^d \pmod n$ 的过程如下：

1. 产生 0 至 $n-1$ 之间的秘密的随机数 r。
2. 计算 $C' = C(r^e) \bmod n$，其中 e 是公开的指数。
3. 像通常的 RSA 运算一样，计算 $M' = (C')^d \bmod n$。
4. 计算 $M = M'r^{-1} \bmod n$，其中 r^{-1} 是 r 模 n 的乘法逆元；关于乘法逆元的讨论见第 8 章。根据 $r^{ed} \bmod n = r \bmod n$，可以证明结论是正确的。

RSA 数据安全算法由于使用了隐蔽方法，其性能降低了 2% ~ 10%。

9.3　推荐读物和网址

第 3 章中给出的有关密码的推荐读物既适用于对称密码，也适用于公钥密码。

[DIFF88]详细描述了几种安全的双钥密码算法和基于这些算法的各种协议的发展过程。[SALO96]是一本关于公钥密码学的优秀书籍。[CORM01]是一本简明易读的书，它总结了与验证、计算和 RSA 密码分析有关的所有算法。[BONE99]讨论了对 RSA 的各种密码分析攻击。

BONE99　Boneh, D. "Twenty Years of Attacks on the RSA Cryptosystem." *Notices of the American Mathematical Society*, February 1999.

CORM01　Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

DIFF88　Diffie, W. "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92].

SALO96　Salomaa, A. *Public-Key Cryptography*. New York: Springer-Verlag, 1996.

SIMM92　Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.


推荐网址：

- **RSA Laboratories**：广泛收集了密码学中有关 RSA 和其他方面的技术资料。

9.4 关键术语、思考题和习题

9.4.1 关键术语

数字签名	密钥交换	单向函数
私钥	公钥	公钥密码学
公钥密码体制	公钥加密	RSA
时间复杂性	计时攻击	单向陷门函数

9.4.2 思考题

9.1 公钥密码体制的主要成分是什么？
9.2 公钥和私钥的作用是什么？
9.3 公钥密码体制的三种应用是什么？
9.4 为得到安全算法，公钥密码体制应满足哪些要求？
9.5 什么是单向函数？
9.6 什么是单向陷门函数？
9.7 试描述一个挑选素数的有效过程。

9.4.3 习题

9.1 在任何诸如 RSA 的公钥体制出现之前，就已经有了关于公钥体制的存在性证明，其目的是为了说明公钥密码在理论上是可行的。考虑函数 $f_1(x_1)=z_1$，$f_2(x_2,y_2)=z_2$，$f_3(x_3,y_3)=z_3$，其中的值都是整数，且 $1\leqslant x_i, y_i, z_i \leqslant N$。函数 f_1 可以用长为 N 的矢量 M1 表示，其中 M1 的第 k 个分量即是 $f_1(k)$；同样，f_2 和 f_3 可分别用 $N\times N$ 矩阵 M2 和 M3 表示。这样表示的目的是希望通过查表实现加/解密过程。N 是一个很大的数，所以这些表也非常大，在实际中不可能实现这些表，但是原则上这些表是可构造的。其构造原理如下：首先，取 M1 为 1 到 N 上的一个随机置换，即 1 到 N 之间的每个整数在 M1 中恰好出现一次；M2 的每一行都是前 N 个整数的随机置换；最后按下述条件生成 M3：

$$f_3(f_2(f_1(k),p),k)=p \qquad \text{对所有} k,p \text{有} 1\leqslant k,p\leqslant N$$

也就是说，

1. M1 的输入为 k，输出为 x。
2. M2 的输入为 x 和 p，输出为 z。

3. M3 的输入为 z 和 k，输出为 p。

然后，公布已构造好的三张表。

a. 显然，可以构造出满足这些条件的 M3。例如，在下述情况下可填充表 M3：

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

约定：M1 的第 i 个分量对应 $k = i$；M2 的第 i 行对应 $x = i$，M2 的第 j 列对应 $p = j$；M3 的第 i 行对应 $z = i$，M3 的第 j 列对应 $k = j$。

b. 说明如何使用上述各表在两个用户间实现加密和解密。

c. 说明这是一种安全的方法。

9.2 用图 9.6 所示的 RSA 算法对下列数据实现加密和解密：

a. $p = 3$；$q = 11$，$e = 7$；$M = 5$

b. $p = 5$；$q = 11$，$e = 3$；$M = 9$

c. $p = 7$；$q = 11$，$e = 17$；$M = 8$

d. $p = 11$；$q = 13$，$e = 11$；$M = 7$

e. $p = 17$；$q = 31$，$e = 7$；$M = 2$

提示：解密并不像你想像的那么难，请使用一些技巧。

9.3 在使用 RSA 的公钥体制中，已截获发给某用户的密文 $C = 10$，该用户的公钥 $e = 5$，$n = 35$，那么明文 M 等于多少？

9.4 在 RSA 体制中，某给定用户的公钥 $e = 31$，$n = 3599$，那么该用户的私钥等于多少？

9.5 使用 RSA 算法时，可以从少量重复的编码中恢复出明文，其可能的原因是什么？

9.6 假定我们已知若干用 RSA 算法编码的分组但不知私钥，假设 $n = pq$，e 是公钥。若某人告诉我们说他知道其中有一个明文分组与 n 有公因子，这对我们有帮助吗？

9.7 在 RSA 公钥密码体制中，每个用户都有一个公钥 e，一个私钥 d。假定 Bob 的私钥已泄密。Bob 决定产生新的公钥和新的私钥，而不是产生新的模数，请问这样安全吗？

9.8 “我想告诉你，福尔摩斯，”华生激动地说，“你最近进行的网络安全活动让我对密码学产生了浓厚的兴趣，就在昨天，我还发现一次一密的加密方法是可行的。”

“噢？真的吗？”福尔摩斯从睡意朦胧中醒过来。“这么说，你找到了一种生成强密码序列的确定性方法了？”

“千真万确，福尔摩斯。这个想法倒是挺简单的。对给定的单向函数 F，通过将 F 应用于某标准的变量参数序列，我产生了一个长伪随机数序列。假使密码分析者知道了 F 和序列的一般性质，这个性质可能很简单，比如 S，S + 1，S + 2，… ，而不知道 S，由于 F 的单向性，没人能够对某 i，从 F(S + i)推出 S，即使他得到了序列的一段，他也不能确定其他部分。”

“华生，我担心你的想法并非无懈可击，至少它要求 F 满足一些附加条件。我们考虑一下。例如，RSA 加密函数 $F(M) = M^K \bmod N$，K 是保密的，这个函数被认为是单向的，但是我不赞成将这种方法用于类似 $M = 2,3,4,5,6,\cdots$ 这样的序列。”

“为什么，福尔摩斯？”华生大惑不解，“为什么你认为，如果 K 是保密的，那么像 2^K

mod N, 3^K mod N, 4^K mod N,… 这样的序列不适合于一次一密?"

"因为它至少是部分可预测的,亲爱的华生,即使 K 是保密的,如你刚才所说,假定密码分析者知道了 F 和序列的一般性质,再假设他能截获一小段输出序列,在密码学界这种假设是很常见的。对于该输出序列,已知最前面的两个元素,即使他不能预测出所有元素,但他可预测出该序列中后续的许多元素。因此这样的序列在密码学上不能被认为是强序列。利用预测出的较长的序列段,他可预测出序列中更多的元素。瞧,已知序列的一般性质和序列的前面两个元素:2^K mod N, 3^K mod N,就很容易计算出后续元素……"

9.9 说明如何通过习题 9.1 中的矩阵 M1,M2 和 M3 来描述 RSA。

9.10 考虑下列方法:

1. 挑选一个奇数 E。
2. 挑选两个素数 P 和 Q,其中$(P-1)(Q-1)-1$ 是 E 的偶数倍。
3. P 和 Q 相乘得 N。
4. 计算 $D=\dfrac{(P-1)(Q-1)(E-1)+1}{E}$。

这种方法是否与 RSA 等价? 请说明原因。

9.11 B 用下述方法对发送给 A 的消息加密:

1. A 选择两个大素数 P 和 Q,它们与$(P-1)$和$(Q-1)$均互素。
2. A 公布其公钥 $N=PQ$。
3. A 计算 P'和Q',使得 $PP'\equiv 1(\text{mod } Q-1)$且 $QQ'\equiv 1 (\text{mod} P-1)$。
4. B 计算 $C=M^N(\text{mod } N)$。
5. 求解 $M\equiv C^{P'}(\text{mod } Q)$和 $M=C^{Q'}(\text{mod } P)$得出 M。

a. 试说明这种方法的工作原理。

b. 它与 RSA 有什么不同?

c. 与这种方法相比,RSA 有哪些优点?

d. 说明如何通过习题 9.1 中的矩阵 M1,M2 和 M3 描述这种方法。

9.12 "这是一个非常有趣的案例,华生。"福尔摩斯说,"这个年轻人爱上了一个女孩,这个女孩也爱他。但是女孩的父亲非常怪,他坚持要求他未来的女婿在公钥密码体制上设计一个简单安全的协议,以便他在公司的计算机网络中使用。这个年轻人提出了下列两方通信协议:假设用户 A 要将消息 M 发送给用户 B[交换的消息形为(发送方的姓名,消息正文,接收方的姓名)]。"

1. A 将$(A,E_{KU_b}[M,A],B)$发送给 B。
2. B 发送应答$(B,E_{KU_a}[M,B],A)$给 A。

"这个协议确实很简单,但是女孩的父亲还是认为该协议不够简单,因为这种协议中存在一些冗余,可进一步简化为:"

1. A 将$(A,E_{KU_b}[M],B)$发送给 B。
2. B 发送应答$(B,E_{KU_a}[M],A)$给 A。

"由于这个原因,女孩的父亲不许他的女儿与年轻人结婚,这使得他们非常不愉快,

因此年轻人来我这里请求我的帮助。”

“嗯，我不知道你会怎样帮助他。”华生想到年轻人要失去他心爱的人，显得有些不快。

“我想我可以帮助他，你知道，华生，冗余有时候对保证协议的安全性是有好处的，因此女孩父亲简化后的协议容易受到一种攻击，而年轻人设计的协议能够抗这种攻击。”福尔摩斯若有所思地说，“有办法了，华生。瞧，攻击者必须是网络用户中的一员，且能够截获 A 和 B 交换的消息。因为是网络中的用户，所以他自己也有公钥，并且他可以发消息给 A 或 B，也可以接收 A 或 B 发出的消息。如果使用这个简化后的协议，那么他可以按下述过程得出 A 以前发送给 B 的消息 M……”

9.13 下面是快速乘方算法的另一种实现方法。证明它与 9.2 节中的方法是等价的。

1. d←1; T←a; E←b
2. **if** odd(e), **then** d←d × T
3. E←⌊E/2⌋
4. T←T × T
5. **if** E > 0, **then goto** 2
6. **output** d

9.14 改进附录 9A 中的算法 P1。

a. 设计一个执行 $2n$ 次乘法和 $n+1$ 次加法的算法。提示：$x^{i+1} = x^i \times x$。

b. 设计一个仅执行 $n+1$ 次乘法和 $n+1$ 次加法的算法。提示：$P(x) = a_0 + x \times q(x)$，其中 $q(x)$ 是次数为 $(n-1)$ 的多项式。

附录 9A　算法复杂性

评价密码算法抗密码分析能力的核心问题是这种攻击所需要的时间。通常，我们不能肯定所找到的攻击算法是最有效的方法，而至多只能说对某特定算法，攻击所需的代价具有多大规模。我们通过将该规模与当前或预知的处理器的速度进行比较，来决定算法的安全程度。

算法时间复杂性是衡量算法有效性的常用标准。如果对所有的 n 和所有长度为 n 的输入，算法的执行至多需要 $f(n)$ 步，则我们定义算法的时间复杂性为 $f(n)$。因此，对给定的输入规模和处理器的速度，时间复杂性是算法执行时间的上界。

这里有几个概念的定义不够明确。第一，步的定义不精确。一步可以是图灵机的一次操作、一条处理器机器指令、一条高级语言机器指令，等等。不过，步的上述各种定义都可以通过一个简单的乘法常量把它们联系起来。当 n 很大时这些常量并不重要，重要的是相对执行时间增长得有多快。例如，如果我们关心的是使用 50 位（$n = 10^{50}$）还是 100 位（$n \approx 10^{100}$）的 RSA 密钥，那么我们不必精确地知道破译这些密钥所需的时间，而是感兴趣于所需时间的近似值，以及破译更长的密钥还需多少额外的代价。

第二，一般地，我们并不给出 $f(n)$ 的精确公式，而只是给出它的近似公式。但我们主要感兴趣的是，当 n 很大时 $f(n)$ 的变化速度。

算法时间复杂性常用称为“大 O”的标准数学符号来刻划，其定义为：$f(n) = O(g(n))$ 当且仅当存在两个数 a 和 M 使得：

$$|f(n)| \leqslant a \times |g(n)|, \qquad n \geqslant M \tag{9.1}$$

下面我们通过例子来说明该符号的使用。假定要计算下述形式的多项式：

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

[POHL81]中给出了下列简单的算法：

```
algorithm P1;
   n, i, j: integer; x, polyval: real;
   a, S: array [0..100] of real;
   begin
     read(x, n);
     for i := 0 upto n do
     begin
        S[i] := 1; read(a[i]);
        for j := 1 upto i do S[i] := x × S[i];
        S[i] := a[i] × S[i]
     end;
     polyval := 0;
     for i ;= 0 upto n do polyval := polyval + S[i];
     write ('value at', x, 'is', polyval)
end.
```

该算法分别计算每个子表达式的值。对每个 $S[i]$需要$(i+1)$次乘法：计算 $S[i]$需要 i 次乘法，用 $a[i]$乘以 $S[i]$需要一次乘法。计算所有的 n 项需要

$$\sum_{i=0}^{n}(i+1) = \frac{(n+2)(n+1)}{2}$$

次乘法。虽然算法还执行了$(n+1)$次加法，但是由于乘法次数更多，所以相对于乘法次数而言我们可以忽略这些加法运算，因此算法的时间复杂性 $f(n) = (n+2)(n+1)/2$。下面我们证明 $f(n) = O(n^2)$。根据等式(9.1)中的定义，我们证明对 $a=1, M=4, g(n) = n^2$ 时关系式成立。施归纳于 n，因为$(4+2)(4+1)/2 = 15 < 4^2 = 16$，所以 $n = 4$ 时关系式成立。假设对小于等于 k 的所有 n，关系式都成立，即$(k+2)(k+1)/2 < k^2$，那么 $n = k+1$ 时，

$$\begin{aligned}\frac{(n+2)(n+1)}{2} &= \frac{(k+3)(k+2)}{2}\\ &= \frac{(k+2)(k+1)}{2} + k + 2\\ &\leqslant k^2 + k + 2\\ &\leqslant k^2 + 2k + 1 = (k+1)^2 = n^2\end{aligned}$$

所以 $n = k+1$ 时结论成立。

一般地，大 O 符号使用增长速度最快的项，例如

1. $O(ax^7 + 3x^3 + \sin(x)) = O(ax^7) = O(x^7)$
2. $O(e^n + an^{10}) = O(e^n)$
3. $O(n! + n^{50}) = O(n!)$

大 O 符号还有许多其他意义，有兴趣的读者请参阅文献[GRAH94]。

设输入规模为 n，称算法是

- **线性的**:若运行时间为 $O(n)$
- **多项式的**:若有某常量 t,使得运行时间为 $O(n^t)$
- **指数的**:若有某常量 t 和多项式 $h(n)$,使得运行时间为 $O(t^{h(n)})$

一般来说,在多项式时间内可解的问题被认为是可行的,而任何比多项式时间更坏,尤其是指数时间可解的问题,则被认为是不可行的。使用这些术语时必须小心。第一,若输入的规模太小,则即使很复杂的算法也会变成是可行的。例如,假定一个系统在单位时间可执行 10^{12} 次操作。表 9.4 列出了几种不同复杂性的算法在单位时间内能处理的输入规模的大小。对指数或阶乘时间复杂性的算法,只适合于非常小的输入规模。

表 9.4　几种不同复杂性的算法的代价

复　杂　性	规　　模	操　　作
$\log_2 n$	$2^{10^{12}} = 10^{3\times10^{11}}$	10^{12}
n	10^{12}	10^{12}
n^2	10^6	10^{12}
n^6	10^2	10^{12}
2^n	39	10^{12}
$n!$	15	10^{12}

第二,要注意对输入的刻画方式。例如,对加密算法进行密码分析的复杂性可以通过可能的密钥数,同样也可以通过密钥的长度来刻画。如对 DES 算法可能的密钥数是 2^{56} 个,密钥长度为 56 位。若我们将一次加密看做是一"步",可能的密钥数是 $N = 2^n$,则算法的时间复杂性是密钥数的线性函数 $O(N)$,但却是密钥长度的指数函数 $O(2^n)$。

第 10 章　密钥管理和其他公钥密码体制

本章继续简要介绍公钥密码，讨论公钥密码体制的密钥分配和密钥管理，包括 Diffie-Hellman 密钥交换，最后介绍椭圆曲线密码学。

10.1　密钥管理

在第 7 章中，我们曾讨论了传统密码体制的密钥分配问题。公钥密码的主要作用之一就是解决密钥分配问题，在这方面，公钥密码实际上可用于下列两个不同的方面：

- 公钥的分配
- 公钥密码用于传统密码体制的密钥分配

接下来，我们就来讨论这两方面的内容。

10.1.1　公钥的分配

人们已经提出了几种公钥分配方法，所有这些方法本质上可归结为下列几种方法：

- 公开发布
- 公开可访问目录
- 公钥授权
- 公钥证书

公钥的公开发布

表面上看，公钥密码的特点就是公钥可以公开，因此如果有像 RSA 这样为人们广泛接受的公钥算法，那么任一通信方可以将他的公钥发送给另一通信方或广播给通信各方（如图 10.1）。例如，越来越为人们广泛使用的 PGP（pretty good privacy，该方法将在第 15 章中讨论）中使用了 RSA 算法，所以许多 PGP 用户在给诸如 USENET 新闻组和 Internet 邮件列表这样的一些公开论坛发送消息时，都将其公钥附加在要发送的消息之后。

虽然这种方法比较简便，但它有一个较大的缺点，即任何人都可以伪造这种公钥的公开发布。也就是说，某个用户可以假冒是用户 A 并将一个公钥发送给通信的另一方或广播该公钥，在用户 A 发现这种假冒并通知其他各方之前，该假冒者可以读取所有本应发送给 A 的加密后的消息，并且可以用伪造的密钥进行认证（见图 9.3）。

公开可访问的目录

维护一个动态可访问的公钥目录可以获得更大程度的安全性。某可信的实体或组织负责这个公开目录的维护和分配（见图 10.2），这种方法包含下面几方面的内容：

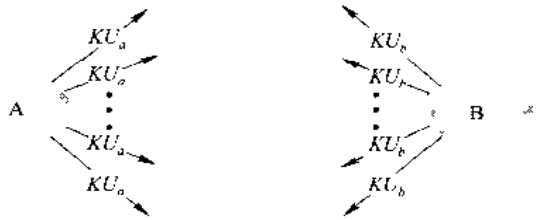


图 10.1　自由的公钥分配

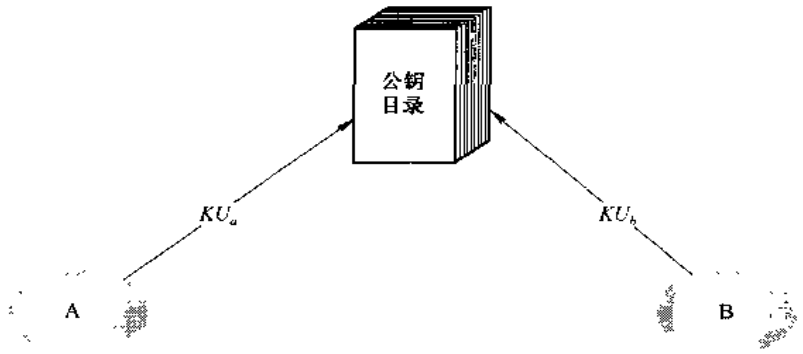


图 10.2　公开的公钥发布

1. 管理员通过对每一通信方建立一个目录项{姓名，公钥}来维护该目录。
2. 每一通信方通过目录管理员来注册一个公钥。注册必须亲自或通过安全的认证通信来进行。
3. 通信方在任何时刻可以用新的密钥替代当前密钥。这可能是因为公钥已用于大量的数据，因而用户希望更换公钥，也可能是因为相应的私钥已经泄密。
4. 管理员定期发布或更新该目录。例如，像电话号码簿一样，可以公开出版目录的硬拷贝，也可以在广泛订阅的报纸上列出更新的密钥。
5. 通信方也可以访问电子目录。为实现这一目标，必须有从管理员到他的安全的认证通信。

这种方法显然比由个人公开发布公钥要安全，但是它也存在缺点。一旦攻击者获得或计算出目录管理员的私钥，则他可以传递伪造的公钥，因此他可以假冒任何通信方，以窃取发送给该通信方的消息。另外，攻击者也可以通过修改目录管理员保存的记录来达到这一目的。

公钥授权

通过更加严格地控制目录中的公钥分配，可使公钥分配更加安全。图 10.3 举例说明了一个典型的公钥分配方案，它基于[POPE79]中给出的图示。像前面一样，该方案中假定中心管理员负责维护通信各方公钥的动态目录，除此之外，每一通信方可靠地知道该目录管理员的公钥，并且只有管理员知道相应的私钥。这种方案包含以下步骤(与图 10.3 中的序号对应)：

1. A 发送一条带有时间戳的消息给公钥管理员，以请求 B 的当前公钥。
2. 管理员给 A 发送一条用其私钥 KR_{auth} 加密的消息，这样 A 就可用管理员的公钥对接收

到的消息解密，因此 A 可以确信该消息来自管理员。这条消息包括下列内容：

- B 的公钥 KU_b。A 可用它对要发送给 B 的消息加密。
- 原始请求。这样 A 可以将该请求与其最初发出的请求进行比较，以验证在管理员收到请求之前，其原始请求未被修改。
- 原始时间戳。这样 A 可以确定它收到的不是来自管理员的旧消息，该旧消息中包含的不是 B 的当前公钥。

3. A 保存 B 的公钥，并用它对包含 A 的标识（ID_A）和临时交互号（N_1）的消息加密，然后发送给 B。这里，临时交互号是用来惟一标识本次交易的。

4,5. 与 A 检索 B 的公钥一样，B 以同样的方法从管理员处检索出 A 的公钥。

至此公钥已被安全地传递给 A 和 B，他们之间的信息交换将受到保护。尽管如此，但是最好还包含下面两步：

6. B 用 KU_a 对 A 的临时交互号（N_1）和 B 所产生的新临时交互号（N_2）加密，并发送给 A。因为只有 B 可以解密消息(3)，所以消息(6)中的 N_1 可以使 A 确信其通信伙伴就是 B。

7. A 用 B 的公钥对 N_2 加密并发送给 B，以使 B 相信其通信伙伴是 A。

这样，总共需要发送七条消息，但是由于 A 和 B 可保存另一方的公钥以备将来使用（这种方法称为暂存），所以并不会频繁地发送前面 4 条消息。不过为了保证通信中使用的是当前公钥，用户应定期地申请对方的当前公钥。

公钥证书

图 10.3 的方案虽然不错，但它还是有缺陷。因为只要用户与其他用户通信，就必须向目录管理员申请对方的公钥，因此公钥管理员就会成为系统的瓶颈。像前面一样，管理员所维护的、含有姓名和公钥的目录也容易被篡改。

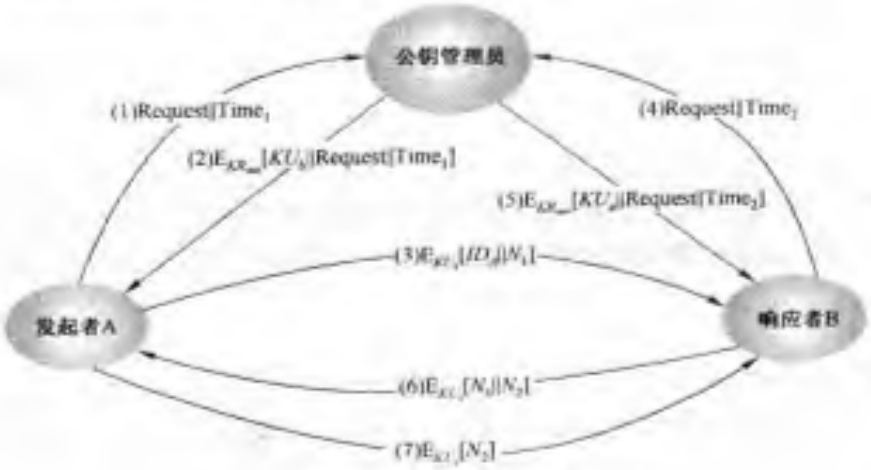


图 10.3 公钥分配方案

最早由 Kohnfelder 提出了使用证书的方法[KOHN78]。通信各方使用证书来交换密钥而不是通过公钥管理员。在某种意义上，这种方案与直接从公钥管理员处获得密钥的可靠性相同。

证书包含公钥和其他一些信息，它由证书管理员产生，并发给拥有相应私钥的通信方。通信一方通过传递证书将密钥信息传递给另一方，其他通信各方可以验证该证书确实是由证书管理者产生的。这种方法应满足下列要求：

1. 任何通信方可以读取证书并确定证书拥有者的姓名和公钥。

2. 任何通信方可以验证该证书出自证书管理员，而不是伪造的。

3. 只有证书管理员才可以产生并更新证书。

[KOHN78]中最初提出的方法能满足上述条件。后来 Denning[DENN83]又增加了下列要求：

4. 任何通信方可以验证证书的当前性。

图 10.4 举例说明了证书方法。每一通信方向证书管理员提供一个公钥并提出申请证书请求。申请必须由当事人亲自或通过某种安全的认证通信提出。对于申请者 A，管理员提供如下形式的证书：

$$C_A = \mathrm{E}_{KR_{auth}}[T, ID_A, KU_a]$$

其中 KR_{auth} 是证书管理员的私钥。A 将该证书发送给其他通信各方，他们读取并如下验证证书：

$$\mathrm{D}_{KU_{auth}}[C_A] = \mathrm{D}_{KU_{auth}}[\mathrm{E}_{KR_{auth}}[T, ID_A, KU_a]] = (T, ID_A, KU_a)$$

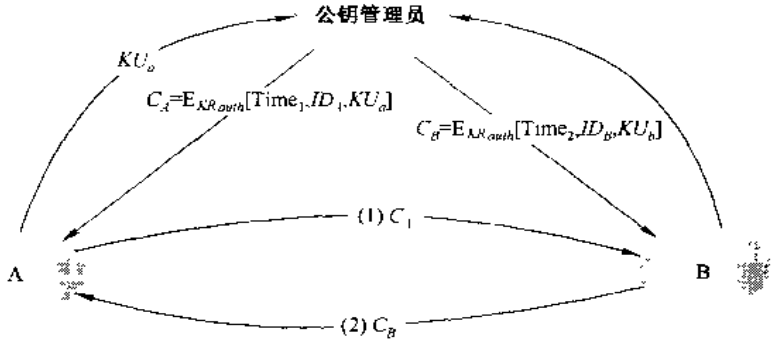


图 10.4　公钥证书的交换

接收方用管理员的公钥 KU_{auth} 对证书解密。因为只用管理员的公钥即可读取证书，因此接收方可验证证书确实是出自证书管理员；ID_A 和 KU_a 向接收方提供证书拥有者的姓名和公钥；时间戳 T 用来验证证书的当前性。时间戳可以在攻击者已知 A 的私钥的情况下抗攻击。假设 A 产生新的公/私钥对并向证书管理员申请新的证书；同时，攻击者重放 A 的旧证书给 B，若 B 用伪冒的旧公钥加密消息，则攻击者可读取消息。

在这种情形下，私钥的泄密就如同信用卡丢失一样，卡的持有者会注销信用卡号，但只有在所有可能的通信方均已知旧信用卡已过时的时候，才能保证卡的持有者的安全。因此，时间戳有些像截止日期。若一个证书太旧，则认为证书已失效。

10.1.2 利用公钥密码分配传统密码体制的密钥

如果已分配了公钥或者公钥是可访问的，那么我们就可以进行安全的通信，这种通信可以抗窃听（见图 9.2）、篡改（见图 9.3），或者同时抗窃听和篡改攻击。但是由于公钥密码速度较慢，几乎没有用户愿意在通信中完全使用公钥密码，因此公钥密码更适合作为传统密码中实现秘密钥分配的一种手段。

简单的秘密钥分配

Merkle 提出了一种极其简单的方法[MERK79]，如图 10.5 所示。若 A 要与 B 通信，则执行下列操作：

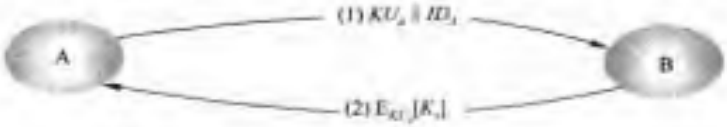


图 10.5 公钥密码用于建立会话密钥的例子

1. A 产生公/私钥对 $\{KU_a, KR_a\}$，并将含有 KU_a 和其标识 ID_A 的消息发送给 B。
2. B 产生秘密钥 K_s，并用 A 的公钥对 K_s 加密后发送给 A。
3. A 计算 $D_{KR_a}[E_{KU_a}[K_s]]$得出秘密钥 K_s。因为只有 A 能解密该消息，所以只有 A 和 B 知道 K_s。
4. A 放弃 KU_a 和 KR_a，B 放弃 KU_a。

这样，A 和 B 就可利用传统密码和会话密钥 K_s 安全地通信。密钥交换完成后，A 和 B 均放弃 K_s。上述协议尽管简单，但却很诱人。由于在通信前和通信完成后都没有密钥存在，所以密钥泄密的可能性最小，同时这种通信还可以抗窃听攻击。

不过该协议容易受主动攻击。如果攻击者 E 能够控制通信信道，那么他可用下列方式对通信造成危害但又不被发现：

1. A 产生公/私钥对 $\{KU_a, KR_a\}$，并将含有 KU_a 和其标识 ID_A 的消息发送给 B。
2. E 截获该消息，产生其公/私钥对 $\{KU_e, KR_e\}$，并将 $KU_e \| ID_A$ 发送给 B。
3. B 产生秘密钥 K_s，并发送 $E_{KU_e}[K_s]$。
4. E 截获该消息，并通过计算 $D_{KR_e}[E_{KU_e}[K_s]]$得出 K_s。
5. E 发送 $E_{KU_a}[K_s]$给 A。

结果是，A 和 B 均已知 K_s，但他们不知道 E 也已知道 K_s。A 和 B 用 K_s 来交换消息；E 不再主动干扰通信信道而只需窃听即可。由于 E 也已知 K_s，所以 E 可解密任何消息，但是 A 和 B 却毫无察觉，因此上述简单协议只能用于仅有窃听攻击的环境中。

具有保密性和真实性的密钥分配

图 10.6 中给出的方法建立在[NEED78]中提出的一种方法之上，它既可抗主动攻击又可抗被动攻击。假定 A 和 B 已通过本节前面所讲到的某种方法交换了公钥，并执行下列操作：

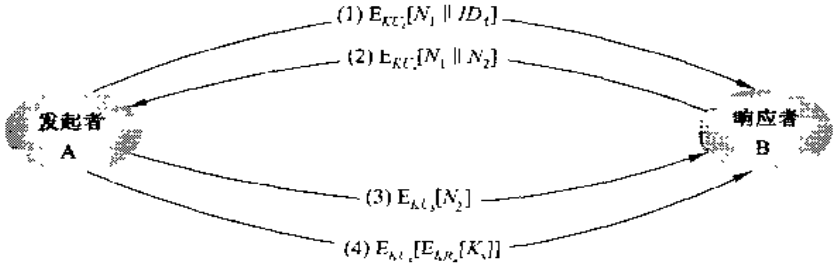


图 10.6　使用公钥密码分配传统密码体制的密钥

1. A 用 B 的公钥对含有其标识 ID_A 和临时交互号(N_1)的消息加密,并发送给 B。其中 N_1 用来惟一标识本次交易。
2. B 发送一条用 KU_a 加密的消息,该消息包含 A 的临时交互号(N_1)和 B 产生的新临时交互号(N_2)。因为只有 B 可以解密消息(1),所以消息(2)中的 N_1 可使 A 确信其通信伙伴是 B。
3. A 用 B 的公钥对 N_2 加密,并返回给 B,这样可使 B 确信其通信伙伴是 A。
4. A 选择密钥 K_s,并将 $M = E_{KU_b}[E_{KR_a}[K_s]]$发送给 B。使用 B 的公钥对消息加密可以保证只有 B 才能对它解密;使用 A 的私钥加密可以保证只有 A 才能发送该消息。
5. B 计算 $D_{KU_a}[D_{KR_b}[M]]$得到密钥。

请注意,本方法的前三步与图 10.3 中的后三步相同,但是它们在传统密码体制密钥交换过程中,既可保证保密性又可保证真实性。

混合方法

混合方法也是利用公钥密码来进行密钥分配,在 IBM 计算机上曾使用了这种方法[LE93]。这种方法也需要密钥分配中心(KDC),该 KDC 与每一用户共享一个秘密的主密钥,通过用该主密钥加密来实现秘密的会话密钥的分配。公钥方法在这里只用来分配主密钥。使用这种三层结构方法的依据如下:

- **性能**:许多应用,特别是面向交易的应用,需要频繁地交换会话密钥。因为公钥加密和解密计算量大,所以若用公钥密码进行会话密钥的交换,则会降低整个系统的性能。利用三层结构方法,公钥密码只是偶尔用来在用户和 KDC 间更新主密钥。
- **向后兼容性**:只需花很小的代价或在软件上做一些修改,我们就可以很容易地将混合方法用于现有的 KDC 方法中。

增加公钥层是分配主密钥的一种安全有效的手段,它对于一个 KDC 对应许多分散用户的系统而言具有其优越性。

10.2　Diffie-Hellman 密钥交换

Diffie 和 Hellman 在一篇具有独创意义的论文中首次提出了公钥算法,给出了公钥密码学的

定义，该算法通常称为 Diffie-Hellman 密钥交换[①]。许多商业产品都使用了这种密钥交换技术。

该算法的目的是使两个用户能安全地交换密钥，以便在后续的通信中用该密钥对消息加密。该算法本身只限于进行密钥交换。

Diffie-Hellman 算法的有效性建立在计算离散对数是很困难的这一基础之上。简单地说，我们可如下定义离散对数。首先我们定义素数 p 的本原根。素数 p 的本原根是一个整数，且其幂可以产生 1 到 $p-1$ 之间的所有整数。也就是说，若 α 是素数 p 的本原根，则

$$\alpha \bmod p, \alpha^2 \bmod p, \cdots, \alpha^{p-1} \bmod p$$

各不相同，它是整数 1 到 $p-1$ 的一个置换。

对任意整数 b 和素数 p 的本原根 α，我们可以找到惟一的指数 i，使得：

$$b \equiv \alpha^i \bmod p \qquad \text{这里 } 0 \leqslant i \leqslant (p-1)$$

指数 i 称为 b 的以 α 为底的模 p 离散对数或指标，记为 $\text{ind}_{\alpha,p}(b)$。有关离散对数的进一步讨论请见第 8 章。

下面我们给出 Diffie-Hellman 密钥交换，如图 10.7。在这种方法中，素数 q 及其本原根 α 是两个公开的整数。假定用户 A 和 B 希望交换密钥，那么用户 A 选择一个随机整数 $X_A < q$，并计算 $Y_A = \alpha^{X_A} \bmod q$。类似地，用户 B 也独立地选择一个随机整数 $X_B < q$，并计算 $Y_B = \alpha^{X_B} \bmod q$。A 和 B 保持其 X 是私有的，但对另一方而言，Y 是公开可访问的。用户 A 计算 $K = (Y_B)^{X_A} \bmod q$ 并将其作为密钥，用户 B 计算 $K = (Y_A)^{X_B} \bmod q$ 并将其作为密钥。这两种计算所得的结果是相同的：

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q && \text{根据模型算术的运算规律} \\
&= \alpha^{X_B X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

至此 A 和 B 完成了密钥的交换。此外，由于 X_A 和 X_B 是私有的，所以攻击者只能通过 q，α，Y_A 和 Y_B 来进行攻击。这样，他就必须求离散对数才能确定密钥。例如，要对用户 B 的密钥进行攻击，攻击者就必须先计算：

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

然后他就可以像用户 B 那样计算出密钥 K。

Diffie-Hellman 密钥交换的安全性建立在下述事实之上：求关于素数的模幂运算相对容易，而计算离散对数却非常困难；对于大素数，求离散对数被认为是不可行的。

下面给出的例子中，密钥交换中所使用的素数 $q = 353$ 和它的一个本原根 $\alpha = 3$。A 和 B

① 英国 CESG 的 Williamson 几个月前在一份机密文档中提出了相同的方法[WILL76]，并声称是在此几年前设计的，请参见[ELLI99]的有关讨论。

分别选择密钥 $X_A = 97$ 和 $X_B = 233$，并计算相应的公钥：

$$\text{A 计算 } Y_A = 3^{97} \bmod 353 = 40$$
$$\text{B 计算 } Y_B = 3^{233} \bmod 353 = 248$$

A 和 B 交换公钥后，双方均可计算出公共的密钥：

$$\text{A 计算 } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$$
$$\text{B 计算 } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$$

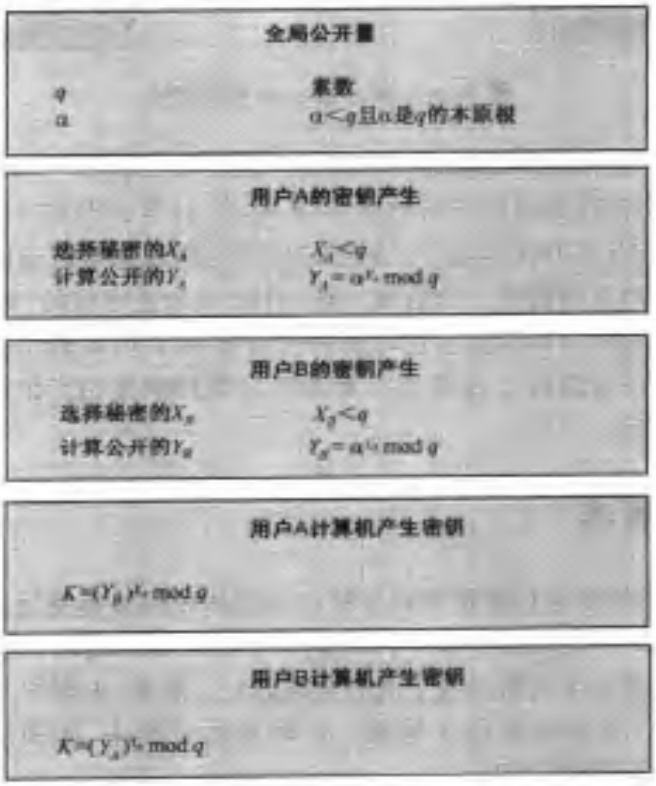


图 10.7　Diffie-Hellman 密钥交换算法

我们假定攻击者能够得到下列信息：

$$q = 353;\ \alpha = 3;\ Y_A = 40;\ Y_B = 248$$

在这个简单的例子中，用穷举攻击确定密钥 160 是可能的。特别地，攻击者可以通过寻找方程 $3^a \bmod 353 = 40$ 或 $3^b \bmod 353 = 248$ 的解来确定该公共密钥。穷举攻击方法即是要计算 3 模 353 的若干幂，当计算结果等于 40 或 248 时则停止。因为 $3^{97} \bmod 353 = 40$，所以幂值为 97 时可得到期望的结果。

对于大数，上述方法实际是不可行的。

图 10.8 给出的简单协议使用了 Diffie-Hellman 计算方法。假定 A 希望与 B 建立连接，并使用密钥对该次连接中的消息加密。用户 A 产生一次性私钥 X_A，计算 Y_A，并将 Y_A 发送给 B；用户 B 也产生私钥 X_B，计算 Y_B，并将 Y_B 发送给 A。这样，A 和 B 都可以计算出密钥。当然，在通信前 A 和 B 都应已知公开的 q 和 α，如可由用户 A 选择 q 和 α，并将 q 和 α 放入第一条消息中。

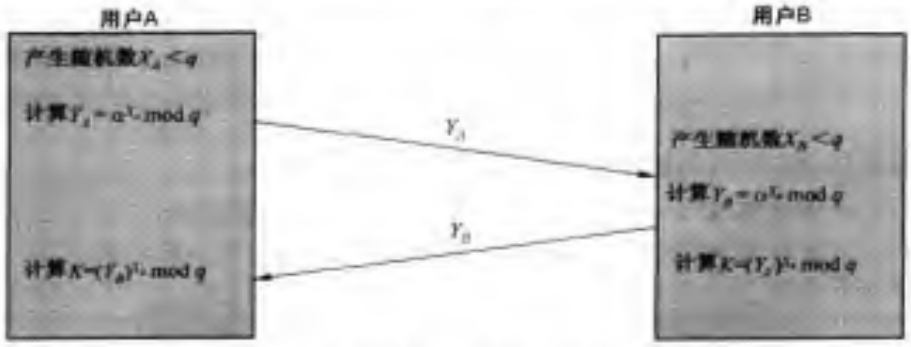


图 10.8 Diffie-Hellman 密钥交换

下面是使用 Diffie-Hellman 算法的另一个例子。假定有一组用户(如 LAN 上的所有用户),且每个用户都产生一个在较长时间内有效的私钥 X_A,并计算公开的 Y_A。这些公开值与公开的全局量 q 和 α 一起存于某中心目录中,在任何时刻用户 B 都可以访问用户 A 的公开值,计算出密钥,并用密钥对消息加密后发送给 A。若该中心目录是可信的,则这种形式的通信既可保证保密性,又可保证某种程度的真实性。因为只有 A 和 B 可以确定密钥,所有其他用户均不能读取该消息(保密性);接收方 A 知道只有用户 B 能用该密钥产生消息(真实性)。但是,这种方法不能抗重放攻击。

10.3 椭圆曲线算术

大多数使用公钥密码学进行加密和数字签名的产品和标准都使用 RSA 算法。我们知道,为了保证 RSA 使用的安全性,最近这些年来密钥的位数一直在增加,这对使用 RSA 的应用是很重的负担,对进行大量安全交易的电子商务更是如此。近来,出现的一种具有强大竞争力的椭圆曲线密码学(ECC)对 RSA 提出了挑战。在标准化过程中,如关于公钥密码学的 IEEE P1363 标准中,人们也已考虑了 ECC。

与 RSA 相比,ECC 的主要诱人之处在于,它可以使用比 RSA 短得多的密钥得到相同的安全性,因此可以减少处理负荷。另一方面,虽然关于 ECC 的理论已很成熟,但直到最近才出现这方面的产品,对 ECC 的密码分析也刚刚起步,因此 ECC 的可信度还没有 RSA 高。

ECC 比 RSA 或 Diffie-Hellman 更难阐述,关于 ECC 的完整数学描述已超出本书范围。本节和下一节中,我们只给出有关椭圆曲线和 ECC 的一些背景知识。我们首先简要讨论 Abel 群,然后讨论定义在有限域上的椭圆曲线,最后讨论椭圆曲线密码。

在下述讨论之前,读者可思考第 4 章中关于有限域的内容。

10.3.1 Abel 群

由第 4 章可知,Abel 群 G 由元素的集合及其上的二元运算·组成,有时记为 $\{G,\cdot\}$。将 G 中的元素序偶(a,b)与 G 中的元素$(a\cdot b)$对应,使得下述公理[①] 成立:

① 运算符·是通用运算,它可以是加法、乘法或其他的数学运算。在本节中,我们用加法运算符 + 表示群的运算。

(A1) 封闭性	若 a 和 b 属于 G，则 $a \cdot b$ 也属于 G。
(A2) 结合性	对 G 中任意的 a，b 和 c，$a \cdot (b \cdot c) = (a \cdot b) \cdot c$。
(A3) 单位元	G 中存在元素 e，使得对 G 中所有的 $a \cdot e = e \cdot a = a$。
(A4) 逆元	对 G 中任何 a，存在 G 中元素 a'，使得 $a \cdot a' = a' \cdot a = e$。
(A5) 交换性	对 G 中任何 a 和 b，有 $a \cdot b = b \cdot a$。

许多公钥密码都使用了 Abel 群。例如 Diffie-Hellman 密钥交换包含若干非零整数对素数 q 的取模运算，通过幂运算来产生密钥，其中幂运算定义为重复相乘。如 $a^k \bmod q = \underbrace{(a \times a \times \cdots \times a)}_{k次} \bmod q$。要攻击 Diffie-Hellman，攻击者必须对给定的 a 和 a^k 确定 k，这即是求离散对数问题。

椭圆曲线密码学使用椭圆曲线上称为加法的运算，乘法被定义为重复相加。例如 $a \times k = \underbrace{(a + a + \cdots + a)}_{k次}$，其中加法是椭圆曲线上的加法。密码分析者需要从给定的 a 和 $(a \times k)$ 来确定 k。

椭圆曲线是一个具有两个变元的方程。对于密码学而言，变元和系数均限制于有限域上。有限域的定义涉及有限 Abel 群的定义。在讨论这些之前，我们先讨论变元和系数均是实数的椭圆曲线，这种情况也许更常见。

10.3.2　实数域上的椭圆曲线

椭圆曲线并不是椭圆，之所以称为椭圆曲线，是因为它们与计算椭圆周长的方程相似，也是用三次方程来表示的。一般地，椭圆曲线的三次方程形为：

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

其中 a，b，c，d 和 e 是实数，x 和 y 在实数集上取值[①]。对我们而言，将方程限制为下述形式就已足够：

$$y^2 = x^3 + ax + b \tag{10.1}$$

因为方程中的指数最高是 3，所以我们称之为三次方程，或者说方程的次数为 3。椭圆曲线的定义中还包含一个称为无穷远点或零点的元素，记为 O，我们以后再讨论这个概念。为了画出该曲线，我们需要计算：

$$y = \sqrt{x^3 + ax + b}$$

对给定的 a 和 b，对 x 的每一个值，需画出 y 的正值和负值，这样每一曲线都关于 $y = 0$ 对称。图 10.9 给出了椭圆曲线的两个例子，由图中可知，上述方程有时对应的是一条怪异的曲线。

现在我们考虑满足等式(10.1)的所有点 (x, y) 和元素 O 所组成的点集 $E(a, b)$。(a, b) 的值不同，则相应的集合 $E(a, b)$ 也不同。使用上述术语，图 10.9 中的两条曲线可分别用集合 $E(-1, 0)$ 和 $E(1, 1)$ 分别表示。

① 注意，x 和 y 是真正的变元，它们具有值。这与第 4 章中讨论的多项式环和域不同，其时 x 被看做是待定元。

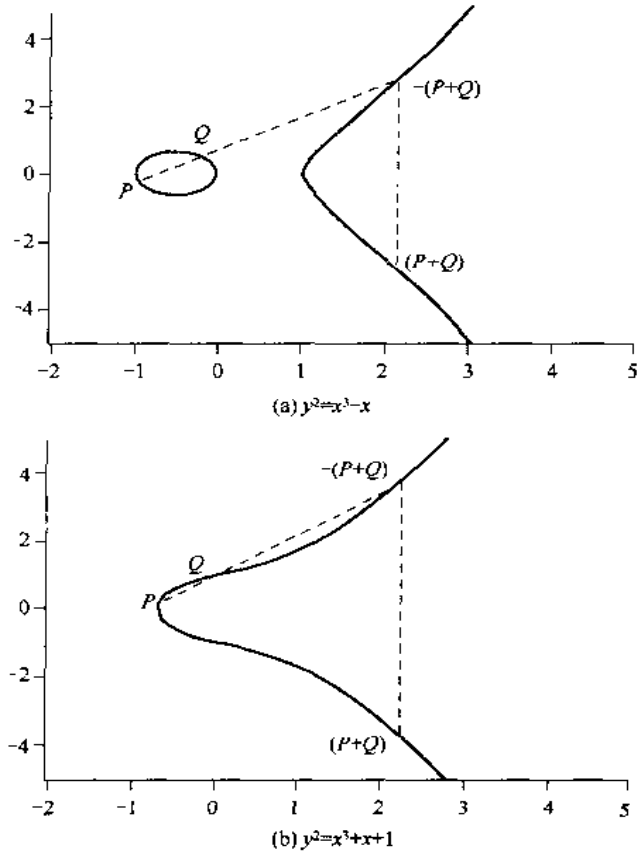


图 10.9 椭圆曲线举例

加法的几何描述

可以证明，只要 $x^3 + ax + b$ 无重复因子，则可基于集合 $E(a,b)$ 定义一个群。这等价于条件：

$$4a^3 + 27b^2 \neq 0 \tag{10.2}$$

下面在 $E(a,b)$ 上定义加法运算，用 + 表示，其中 a 和 b 满足等式(10.2)。用几何术语可如下定义加法的运算规则：若椭圆曲线上的三个点同在一条直线上，则它们的和为 O。从这个定义出发，我们可以定义椭圆曲线加法的运算规则：

1. O 是加法的单位元。这样有 $O = -O$；对椭圆曲线上的任何一点 P，有 $P + O = P$。下面假定 $P \neq Q$ 且 $Q \neq O$。
2. 点 P 的负元是具有相同 x 坐标和相反 y 坐标的点[即若 $P = (x, y)$，则 $-P = (x, -y)$]。注意这两个点可用一条垂直的线连接起来，并且 $P + (-P) = P - P = O$。
3. 要计算 x 坐标不相同的两点 P 和 Q 之和，则在 P 和 Q 间作一条直线并找出第三个交点 R，显然存在有惟一的交点 R(除非这条直线在 P 或 Q 处与该椭圆曲线相切，此时我们分别取 $R = P$ 或 $R = Q$)。要形成群，需要定义如下三个点上的加法：$P + Q = -R$。也就是说，定义 $P + Q$ 为第三个交点(相对于 x 轴)的镜像。图 10.9 说明了这一情形。

4. 上述术语的几何解释也适用于具有相同 x 坐标的两个点 P 和 $-P$ 的情形。用一条垂直的线连接这两点，这也可看做是在无穷远点处与曲线相交，因此有 $P+(-P)=O$，与上述(2)相一致。
5. 为计算点 Q 的两倍，画一条切线并找出另一交点 S，则 $Q+Q=2Q=-S$。

利用前述的运算规则，可以证明集合 $E(a,b)$ 是 Abel 群。

加法的代数描述

在本小节中，我们给出一些用于椭圆曲线上加法的结论[①]。对不是互为负元的两个不同的点 $P=(x_P,y_P)$ 和 $Q=(x_Q,y_Q)$，连接它们的曲线 l 的斜率 $\Delta=(y_Q-y_P)/(x_Q-x_P)$。$l$ 恰与椭圆曲线相交于另一点，即 P 与 Q 之和的负元。利用某些代数运算，我们可如下表示和 $R=P+Q$：

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

上述加法满足普通加法的性质，如交换律和结合律。我们定义正整数 k 乘以椭圆曲线上的点 P 为 k 个 P 之和，因此有 $2P=P+P$，$3P=P+P+P$，等等。

我们也需要能够计算一个点与它自身相加：$P+P=2P=R$。当 $y_P\neq 0$ 时，该表达式为：

$$\begin{aligned} x_R &= \left(\frac{3x_P^2+a}{2y_P}\right)^2 - 2x_P \\ y_R &= \left(\frac{3x_P^2+a}{2y_P}\right)(x_P - x_R) - y_P \end{aligned} \tag{10.4}$$

10.3.3　Z_p 上的椭圆曲线

椭圆曲线密码体制使用的是变元和系数均为有限域中元素的椭圆曲线。密码应用中所使用的两类椭圆曲线是定义在 Z_p 上的素曲线(prime curves)和在 $GF(2^n)$ 上构造的二元曲线。[FERN99]指出，因为不需要二元曲线所要求的位混淆(bit-fiddling)运算，对软件应用最好使用素曲线；而对硬件应用最好使用二元曲线，它可以用异常少的门电路来得到快速且功能强大的密码体制。我们将在本节和下一节中讨论这两类曲线。

对有限域上的椭圆曲线算术没有显而易见的几何解释。用于有限域上椭圆曲线算术的代数解释就要完成，在此也不做讨论。

对 Z_p 上的椭圆曲线，我们使用变元和系数均在 0 到 $p-1$ 的整数集上取值的三次方程，其中 p 是素数，所执行的计算均是模 p 运算。与关于实数时的情形一样，我们限制方程具有等式(10.1)所示的形式，但此处系数和变元均限制在 Z_p 中：

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \tag{10.5}$$

例如，$a=1$，$b=1$，$x=9$，$y=7$，$p=23$ 时可满足等式(10.5)：

---

① 关于这些结论的推导请见[KOBL94]或其他关于椭圆曲线的数学资料。

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

下面考虑有所有满足等式(10.5)的整数对(x, y)和无穷远点 O 组成的集合 $E_p(a, b)$。

例如,取 $p = 23$。考虑椭圆曲线方程 $y^2 = x^3 + x + 1$,这里 $a = b = 1$。注意,该方程与图 10.9(b)中的方程是相同的。图中显示了所有满足方程的实点。对 $E_{23}(1,1)$,我们只感兴趣与满足模 p 方程的、从(0,0)到$(p-1, p-1)$的象限中的非负整数。表 10.1 列出了若干点(除 O 外),这些点是 $E_{23}(1,1)$的一部分,图 10.10 给出了 $E_{23}(1,1)$上的点。注意这些点除了一个之外,均关于 $y = 11.5$ 对称。

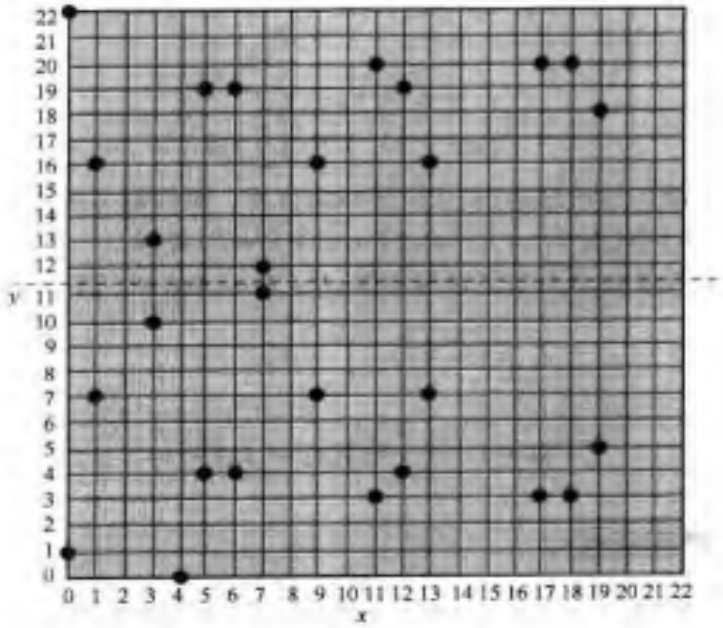


图 10.10 椭圆曲线 $E_{23}(1,1)$

表 10.1 椭圆曲线 $E_{23}(1,1)$上的点

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

可以证明,若$(x^3 + ax + b) \bmod p$ 无重复因子,则基于集合 $E_p(a, b)$可定义一个有限 Abel

群。这等价于下列条件：

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \tag{10.6}$$

注意等式(10.6)和等式(10.2)具有相同的形式。

$E_p(a,b)$上的加法运算构造与定义在实数上的椭圆曲线中描述的代数方法是一致的。对任何点 $P,Q \in E_p(a,b)$，有：

1. $P + O = P$。
2. 若 $P = (x_P, y_P)$，则 $P + (x_P, -y_P) = O$。点$(x_P, -y_P)$是 P 的负元，记为 $-P$。例如，对 $E_{23}(1,1)$上的点 $P = (13,7)$，有 $-P = (13,-7)$。而 $-7 \bmod 23 = 16$，因此，$-P = (13,16)$，该点也在 $E_{23}(1,1)$上。
3. 若 $P = (x_P, y_P)$，$Q = (x_Q, y_Q)$，且 $P \neq -Q$，则 $R = P + Q = (x_R, y_R)$由下列规则确定：

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

其中

$$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{若} P \neq Q \\ \left(\dfrac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{若} P = Q \end{cases}$$

4. 乘法定义为重复相加。如 $4P = P + P + P + P$。例如取 $E_{23}(1,1)$上的 $P = (3,10)$，$Q = (9,7)$，那么

$$\lambda = \left(\frac{7-10}{9-3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$
$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3-17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

所以 $P + Q = (17,20)$。为计算 $2P$，先求

$$\lambda = \left(\frac{3(3^2) + 1}{2 \times 10}\right) \bmod 23 = \left(\frac{5}{20}\right) \bmod 23 = \left(\frac{1}{4}\right) \bmod 23 = 6$$

上述等式的最后一步中需求 4 在 Z_{23}中的乘法逆元。

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$
$$y_R = (6(3-7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

可见 $2P = (7,12)$。

为了确定各种椭圆曲线密码的安全性，需要知道定义在椭圆曲线上的有限 Abel 群中点的个数。在有限群 $E_p(a,b)$中，点的个数 N 的范围是：

$$p + 1 - 2\sqrt{p} \leqslant N \leqslant p + 1 + 2\sqrt{p}$$

所以对于大数 p，$E_p(a,b)$上点的个数约等于 Z_p 中元素的个数。

10.3.4 GF(2^m)上的椭圆曲线

由第4章可知,有限域GF(2^m)由2^m个元素及定义在多项式上的加法和乘法运算组成。给定某m,对GF(2^m)上的椭圆曲线,我们使用变元和系数均在GF(2^m)上取值的三次方程,且利用GF(2^m)中的算术运算规则来进行计算。

可以证明,GF(2^m)上适合于椭圆曲线密码应用的三次方程与Z_p上的三次方程有所不同,其形为:

$$y^2 + xy = x^3 + ax^2 + b \tag{10.7}$$

其中变元x和y以及系数a和b是GF(2^m)中的元素,且所有计算均在GF(2^m)中进行。

考虑由满足等式(10.7)的所有整数对(x,y)和无穷远点组成的集合$E_{2^m}(a,b)$。可以证明,只要$b \neq 0$,则可基于集合$E_{2^m}(a,b)$定义一个有限Abel群。加法的运算规则如下所述。对所有点$P,Q \in E_{2^m}(a,b)$:

1. $P + O = P$。
2. 若$P = (x_P, y_P)$,则$P + (x_P, x_P + y_P) = O$。点$(x_P, x_P + y_P)$是P的负元,记为$-P$。
3. 若$P = (x_P, y_P)$,$Q = (x_Q, y_Q)$,且$P \neq -Q$,$P \neq Q$,则$R = P + Q = (x_R, y_R)$由下列规则确定:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$
$$y_R = \lambda(x_P + x_R) + x_R + y_P$$

其中

$$\lambda = \left(\frac{y_Q + y_P}{x_Q + x_P}\right)$$

4. 若$P = (x_P, y_P)$,则$R = 2P = (x_R, y_R)$由下列规则确定:

$$x_R = \lambda^2 + \lambda + a$$
$$y_R = x_P^3 + (\lambda + 1)x_R$$

其中

$$\lambda = x_P + \frac{y_P}{x_P}$$

10.4 椭圆曲线密码学

我们将ECC中的加法运算与RSA中的模乘运算相对应,将ECC中的乘法运算与RSA中的模幂运算对应。要建立基于椭圆曲线的密码体制,我们需要类似因子分解两个素数之积或求离散对数这样的"难题"。

考虑方程$Q = kP$,其中Q, $P \in E_p(a,b)$且$k < p$。对给定的k和P计算Q比较容易,而对给定的Q和P计算k则比较困难。

我们引用Certicom网站(www.certicom.com)中给出的例子来说明这一问题。考虑由方程

$y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$ 所定义的群 $E_{23}(9,17)$。以 $P=(16,5)$为底的 $Q=(4,5)$的离散对数 k 为多少？穷举攻击方法通过计算 P 的倍数来寻找 Q。这样

$$P=(16,5);\ 2P=(20,20);\ 3P=(14,14);\ 4P=(19,20);\ 5P=(13,10);$$
$$6P=(7,3);\ 7P=(8,7);\ 8P=(12,17);\ 9P=(4,5)$$

因为 $9P=(4,5)=Q$，所以以 $P=(16,5)$为底的 $Q=(4,5)$的离散对数 $k=9$。在实际应用中，k 的值非常大，从而使穷举攻击方法不可行。

下面，我们介绍 ECC 的两种应用。

10.4.1　用椭圆曲线密码实现 Diffie-Hellman 密钥交换

利用椭圆曲线可按如下方式实现密钥交换。首先，挑选一个大整数 q 以及等式(10.5)或等式(10.7)中的椭圆曲线参数 a 和 b，这里 q 为素数 p 或是形为 2^m 的整数。由此可以定义出点的椭圆群 $E_q(a,b)$；其次，在 $E_q(a,b)$中挑选基点 $G=(x_1,y_1)$，G 的阶为一个非常大的数 n。椭圆曲线上点 G 的阶 n 是使得 $nG=O$ 成立的最小正整数。$E_q(a,b)$和 G 是该密码体制中通信各方均已知的参数。

用户 A 和用户 B 之间完成密钥交换的过程如下(见图 10.11)：

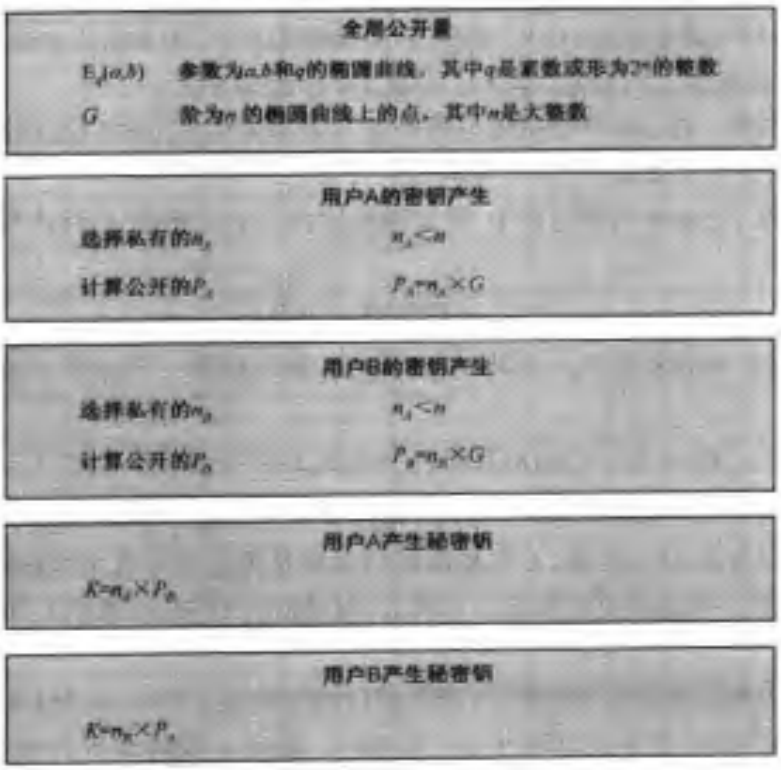


图 10.11　ECC 密钥交换

1. A 选择一个小于 n 的整数 n_A 作为其私钥，然后产生其公钥 $P_A=n_A\times G$；该公钥是 $E_q(a,b)$中的一个点。
2. B 可类似地选择私钥 n_B 并计算公钥 P_B。

3. A产生秘密钥 $K = n_A \times P_B$，B产生秘密钥 $K = n_B \times P_A$。

步骤3中 K 的两种计算结果是相同的，因为

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

要破译这种体制，攻击者必须由 G 和 kG 计算 k，这被认为是非常难的。

例如[①]，取 $p = 211$，$E_p(0,-4)$，$G = (2,2)$，这里 $E_p(0,-4)$ 即是曲线 $y^2 = x^3 - 4$，则计算可得 $240G = O$。A的私钥 $n_A = 121$，所以A的公钥 $P_A = 121(2,2) = (115,48)$。B的私钥 $n_B = 203$，所以B的公钥为 $203(2,2) = (130,203)$，它们共享的秘密钥为 $121(130,203) = 203(115,48) = (161,69)$。

请注意，这里密钥是一对数字。若将它用做传统密码中的会话密钥，则必须是一个数字。我们可以简单地选取 x 坐标或者 x 坐标的某简单函数。

10.4.2 椭圆曲线加/解密

一些文献中分析了几种用椭圆曲线实现加/解密的方法，本小节介绍的也许是其中最简单的一种方法。首先，我们必须将要发送的消息明文 m 编码为形为 x-y 的点 P_m，并对点 P_m 进行加密和其后的解密。注意，不能简单地将消息编码为点的 x 坐标或 y 坐标，因为并不是所有的坐标都在 $E_q(a,b)$ 中，请参见表10.1。将消息 m 编码为点 P_m 的方法有多种，我们不打算在这里讨论这些方法，但需要说明的是，存在有比较直接的编码方法。

像密钥交换系统一样，加/解密系统也需要点 G 和椭圆群 $E_q(a,b)$ 这些参数。每个用户A选择一个私钥 n_A，并产生公钥 $P_A = n_A \times G$。

若A要将消息 P_m 加密后发送给B，则A随机选择一个正整数 k，并产生密文 C_m，该密文是一个点对：

$$C_m = \{kG, P_m + kP_B\}$$

注意，此处A使用了B的公钥 P_B。B要对密文解密，则需用第二个点减去第一个点与B的私钥之积：

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

A通过将 kP_B 与 P_m 相加来伪装消息 P_m，因为只有A知道 k，所以即使 P_B 是公钥，除A外任何人均不能除去伪装 kP_B；但是，A也在伪装后的消息中包含了有关“线索”，使得已知私钥 n_B 时可以除去伪装。攻击者要想恢复消息明文，则必须从 G 和 kG 求出 k，但这被认为是很困难的。

下面举例说明椭圆曲线的加密过程，该例摘自[KOBL94]。取 $p = 751$；$E_p(-1,188)$，即其椭圆曲线方程为 $y^2 = x^3 - x + 188$，$G = (0,376)$。假定A要将编码为椭圆曲线上点 $P_m = (562,201)$ 的消息发送给B，且A挑选的随机数 $k = 386$，B的公钥为 $P_B = (201,5)$，那么有 $386(0,376) = (676,558)$ 和 $(562,201) + 386(201,5) = (385,328)$。于是A发送的密文是 $\{(676,558),(385,328)\}$。

① 本例由Santa Clara大学的Ed Schaefer提供。

10.4.3　椭圆曲线密码的安全性

ECC 的安全性是建立在由 kP 和 P 确定 k 的困难程度之上的，这个问题称为椭圆曲线对数问题。Pollard rho 方法是已知求椭圆曲线对数的最快方法。表 10.2 对这种方法和分解两个素数之积的一般数域筛法进行了比较。由表可知，ECC 使用的密钥比 RSA 中使用的密钥要短得多，而且在密钥长度相同时，ECC 与 RSA 所执行的计算量也差不多[JURI97]。因此，与具有同等安全性的 RSA 相比，由于 ECC 使用的密钥更短，所以 ECC 所需的计算量比 RSA 少。

表 10.2　椭圆曲线密码和 RSA 在计算量上的比较

密钥大小	MIPS 年
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) 用 Pollard rho 方法求椭圆曲线对数

密钥大小	MIPS 年
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) 使用一般数域筛法进行整数因子分解

10.5　推荐读物和网址

[ROSI99]介绍了椭圆曲线密码学，该书简单易懂，其重点在于软件实现。[BLAK99]和[ENGE99]讨论了某些非常难的数学问题。[KUMA98]、[STIN02]和[KOBL94]也是非常好的书，其文字更为简洁。[FERN99]和[JURI97]综述了椭圆曲线密码学。

BLAK99　Blake, I.; Seroussi, G.; and Smart, N. *Elliptic Curves in Cryptography*. Cambridge: Cambridge University Press, 1999.

ENGE99　Enge, A. *Elliptic Curves and Their Applications to Cryptography*. Norwell, MA: Kluwer Academic Publishers, 1999.

FERN99　Fernandes, A. "Elliptic Curev Cryptography." *Dr. Dobb's Journal*, December 1999.

JURI97　Jurisic, A., and Menezes, A. "Elliptic Curves and Cryptography." *Dr. Dobb's Journal*, April 1997.

KOBL94　Koblitz, N. *A Course in Number Theory and Cryptography*. New York: Springer-Verlag, 1994.

KUMA98　Kumanduri, R., and Romero, C. *Number Theory with Computer Applications*. Upper Saddle River, NJ: Prentice Hall, 1998.

ROSI99　Rosing, M. *Implementing Elliptic Curve Cryptography*. Greenwich, CT: Manning Publications, 1999.

STIN02　Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.


推荐网址：

- **Certicom**：广泛收集了椭圆曲线密码学和密码学其他领域的技术资料。

10.6 关键术语、思考题和习题

10.6.1 关键术语

Abel 群	椭圆曲线算术	本原根
Diffie-Hellman 密钥交换	椭圆曲线密码学	公钥证书
离散对数	有限域	公钥目录
椭圆曲线	密钥分配	零点
密钥管理		

10.6.2 思考题

10.1 公钥密码学有关密钥分配的两种不同用途是什么？
10.2 列出 4 种公钥分配方法。
10.3 公钥目录的主要成分是什么？
10.4 什么是公钥证书？
10.5 使用公钥证书应满足哪些要求？
10.6 简要说明 Diffie-Hellman 密钥交换。
10.7 什么是椭圆曲线？
10.8 什么是椭圆曲线的零点？
10.9 椭圆曲线上同在一条直线上的三个点的和是什么？

10.6.3 习题

10.1 用户 A 和 B 使用 Diffie-Hellman 密钥交换技术来交换密钥，设公用素数 $q = 71$，本原根 $\alpha = 7$。

a. 若用户 A 的私钥 $X_A = 5$，则 A 的公钥 Y_A 为多少？

b. 若用户 B 的私钥 $X_B = 12$，则 B 的公钥 Y_B 为多少？

c. 共享的密钥为多少？

10.2 设 Diffie-Hellman 方法中，

a. 公用素数 $q = 11$，本原根 $\alpha = 2$。

b. 若用户 A 的公钥 $Y_A = 9$，则 A 的私钥 X_A 为多少？

c. 若用户 B 的公钥 $Y_B = 3$，则共享的密钥 K 为多少？

10.3 "但是,"华生说,"你的客户在他们的网络中使用的是 Diffie-Hellman 协议,这种协议基于离散对数,而这是公认的难题,不是吗?"

"是的,华生,"福尔摩斯点头道,"对于选择的某些参数,离散对数问题确实很难,我的客户知道这一点,所以他们选择了这种方法进行密钥分配。遗憾的是,他们的安全顾问没有意识到,主动攻击比被动防范更容易成功。攻击者也知道他不能在合理的时间内解决离散对数问题,所有他会试图使用其他方法,因为我相信 Moriarty 本人也对我的客户的通信感兴趣,所以我必须设想在他们的网络上存在某种主动攻击,Moriarty 决不会仅仅用被动攻击的,华生。"

"你认为,福尔摩斯,"华生非常惊讶地说道,"Moriarty 能找到一种攻破 Diffie-Hellman 密钥交换方法的办法吗?"

"噢,这并不太难,华生,"福尔摩斯微笑着说,"Moriarty 要做的只是隐藏在通信信道上的某个地方,使他不仅能截获而且能修改所有的信息,我相信 Moriarty 完全有能力做到这一点,他可以在这个地方……"

10.4 1985 年,T. ElGamal 提出了有关与 Diffie-Hellman 技术密切相关的一种基于离散对数的公钥密码体制。和 Diffie-Hellman 技术一样,ElGamal 体制的全局量为素数 q 和其本原根 α。用户 A 选择其私钥 X_A 并像在 Diffie-Hellman 中一样计算公钥 Y_A,用户 A 对要发送给用户 B 的消息 $M < q$ 按如下方式加密:

1. 选择随机整数 k,$1 \leqslant k \leqslant q-1$。
2. 计算 $K = (Y_B)^k \pmod q$。
3. 加密 M 为一对整数(C_1, C_2),其中

$$C_1 = \alpha^k \pmod q \quad C_2 = KM \pmod q。$$

用户 B 按如下方式恢复明文:

1. 计算 $K = (C_1)^{X_B} \pmod q$。
2. 计算 $M = (C_2 K^{-1}) \pmod q$。

试说明该体制的工作原理,即解密过程能够恢复出明文。

10.5 设 ElGamal 体制的公用素数 $q = 71$,其本原根 $\alpha = 7$。

a. 若 B 的公钥 $Y_B = 3$,A 选择的随机整数 $k = 2$,则 $M = 30$ 的密文是什么?

b. 若 A 选择的 k 值使得 $M = 30$ 的密文为 $C = (59, C_2)$,则整数 C_2 是多少?

10.6 根据实数域上椭圆曲线中的规则(5),要计算点 Q 的两倍,则画一条切线并找出另一交点 S。那么 $Q + Q = 2Q = -S$。若该切线不是垂直的,则恰好只有一个交点。但若切线是垂直的,那么在这种情况下,$2Q$ 的值是多少? $3Q$ 的值是多少?

10.7 证明图 10.9 中的两条实数域上的椭圆曲线均满足群的条件。

10.8 设实数域上的椭圆曲线为 $y^2 = x^3 - 36x$,令 $P = (-3.5, 9.5)$, $Q = (-2.5, 8.5)$。计算 $P + Q$ 和 $2P$。

10.9 考虑椭圆曲线 $E_{11}(1,6)$,即由 $y^2 = x^3 + x + 6$ 定义的曲线,其模数 $p = 11$。确定 $E_{11}(1,6)$ 上所有的点。提示:对 x 的所有可能值计算方程右边的值。

10.10 对 $E_{11}(1,6)$ 上的点 $G = (2,7)$,计算 $2G$ 到 $13G$ 的值。

10.11 利用 10.4 节中给出的方法实现椭圆曲线的加/解密。该密码体制的参数是 $E_{11}(1,6)$和

$G=(2,7)$，B 的私钥 $n_B=7$。

a. 找出 B 的公钥 P_B。

b. A 要加密消息 $P_m=(10,9)$，其选择的随机值 $k=3$，试确定密文 C_m。

c. 试给出 B 由 C_m 恢复 P_m 的计算过程。

第 11 章　消息认证和 hash 函数

与消息认证和数字签名有关的问题也许是网络安全中最易引起混淆的领域，攻击和抗攻击对策的发展呈螺旋式，这使人们想起那些古老的天文学家，他们试图通过在本轮上建造本轮的方法来解释所有的问题。值得庆幸的是，不像那些古老的天文学家，今天的密码协议的设计者是在完全合理的模型上来进行研究的。

只用几章的篇幅来讨论已提出或实现的有关消息认证与数字签名的密码功能和协议是不可能的，本章和后续两章只是概要介绍上述内容，并给出一种用以描述各种方法的手段。

本章首先介绍对认证和数字签名的要求以及可能遇到的攻击类型，然后归纳总结包括安全 hash 函数在内的一些基本方法。安全 hash 函数已成为越来越重要的研究领域，我们将在第 12 章中讨论一些特殊的 hash 函数。

11.1　对认证的要求

在网络通信环境中，可能有下述攻击：

1. **泄密**：将消息透露给没有合法秘密钥的任何人或程序。
2. **传输分析**：分析通信双方的通信模式。在面向连接的应用中，确定连接的频率和持续时间；在面向连接或无连接的环境中，确定双方的消息数量和长度。
3. **伪装**：欺诈源向网络中插入一条消息。如攻击者产生一条消息并声称这条消息是来自某合法实体，或者非消息接收方发送的关于收到或未收到消息的欺诈应答。
4. **内容修改**：对消息内容的修改，包括插入、删除、转换和修改。
5. **顺序修改**：对通信双方消息顺序的修改，包括插入、删除和重新排序。
6. **计时修改**：对消息的延时和重放。在面向连接的应用中，整个消息序列可能是前面某合法消息序列的重放，也可能是消息序列中的一条消息被延时或重放；在面向无连接的应用中，可能是一条消息（如数据报）被延时或重放。
7. **发送方否认**：发送方否认发送过某消息。
8. **接收方否认**：接收方否认接收到某消息。

对付前两种攻击的方法属于消息保密性范畴，我们已在第一部分讨论了消息保密性；对付第 3 种至第 6 种攻击的方法一般称为消息认证；对付第 7 种攻击的方法属于数字签名。一般而言，数字签名方法也能够抗第 3 种至第 6 种中的某些或全部攻击，对付第 8 种攻击需要使用数字签名和为抗此种攻击而设计的协议。

归纳起来，消息认证就是验证所收到的消息确实是来自真正的发送方且未被修改的消息，它也可验证消息的顺序和及时性。数字签名是一种认证技术，其中的一些方法可用来抗发送方否认攻击。

11.2 认证函数

任何消息认证或数字签名方法在功能上基本可看做有两层。下层中一定有某种产生认证符的函数,认证符是一个用来认证消息的值;上层协议中将该函数作为原语使接收方可以验证消息的真实性。

本节讨论可以用来产生认证符的函数类型,这些函数可以分为如下三类:

- **消息加密**:整个消息的密文作为认证符。
- **消息认证码(MAC)**:它是消息和密钥的公开函数,它产生定长的值,以该值作为认证符。
- **hash 函数**:它是将任意长的消息映射为定长的 hash 值的公开函数,以该 hash 值作为认证符。

下面我们简要讨论这些问题。11.3 节和 11.4 节将更加详细地讨论 MAC 和 hash 函数。

11.2.1 消息加密

消息加密本身提供了一种认证手段。传统密码和公钥密码体制中,对消息加密的分析是不相同的。

对称加密

考虑一个使用传统加密的简单例子[图 11.1(a)]。发送方 A 用 A 和 B 共享的秘密钥 K 对发送到接收方 B 的消息 M 加密,如果没有其他方知道该密钥,那么可提供保密性,因为任何其他方均不能恢复出消息明文。

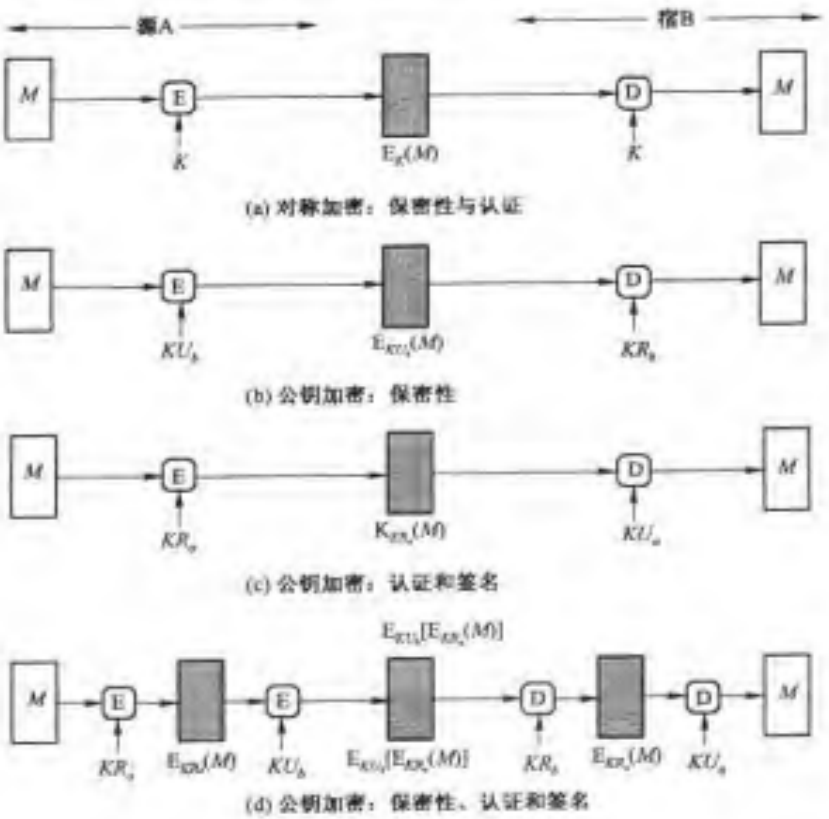
B 可确信该消息是由 A 产生的。为什么呢? 因为除 B 外只有 A 拥有 K,A 能产生出用 K 可解密的密文,所以该消息一定来自 A。由于攻击者不知道密钥,他也就不知如何改变密文中的信息位才能在明文中产生预期的改变。因此,若 B 可以恢复出明文,则 B 可以认为 M 中的每一位都未被改变。

所以我们可以说,对称密码既可提供认证又可提供保密性,但这不是绝对的。考虑在 B 方所发生的事件,给定解密函数 D 和密钥 K,接收方可接收任何输入 X,并产生输出 $Y = D_K(X)$。若 X 是用相应的加密函数对合法消息 M 加密生成的密文,则 Y 就是明文消息 M;否则 Y 可能是无意义的位串。因此在 B 端需要有某种方法能确定 Y 是合法的明文以及消息确实是发自 A。

从认证的角度来看,上述推理存在这样一个问题。如果消息 M 可以是任何的位模式,那么接收方无法确定接收到的消息是合法明文的密文。若 M 可以是任意的位模式,那么不管 X 的值是什么,$Y = \mathrm{D}_K(X)$都会作为真实的密文被接受。

一般地,我们要求合法明文只是所有可能位模式的一个小子集。这样,由任何伪造的密文都不太可能得出合法的明文。例如,假定 10^6 种位模式中只有一种是合法明文的位模式,那么随机选择一个位模式作为密文,它产生合法明文消息的概率只有 10^{-6}。

许多应用和加密方法都满足上述条件。例如,假定我们利用具有一次移动($K = 1$)的 Caesar 密码来传递英文消息,A 发送下列合法的消息:



(a) 对称加密：保密性与认证

(b) 公钥加密：保密性

(c) 公钥加密：认证和签名

(d) 公钥加密：保密性、认证和签名

图 11.1　消息加密的基本用途

```
nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz
```

B 解密并产生下列明文：

```
mareseatoatsanddoeseatoatsandlittlelambseativy
```

通过简单的频率分析可以发现这个消息具有普通英语的特点。若攻击者产生下列随机的字符序列：

```
zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq
```

则它被解密为：

```
ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp
```

这个序列不具有普通英语的特点。

对接收到的密文解密，再对所得明文的合法性进行判别，是一件困难的事情。比如，若明文是二进制文件或数字化的 X 射线，那么很难确定解密后的消息是正确生成的，即真实的明文。此攻击者可以简单地发布任何消息并伪称是发自合法用户的消息，从而造成某种程度的破坏。

解决这个问题的方法之一是，要求明文具有某种易于识别的结构，并且不通过加密函数是不能重复这种结构的。例如，可以在加密前对每个消息附加一个错误检测码，也称之为帧校验序列(FCS)或校验和，如图 11.2(a)所示。A 准备发送明文消息 M，那么 A 将 M 作为函数 F 的

输入，产生 FCS，将 FCS 附加在 M 后并对 M 和 FCS 一起加密。在接收端，B 解密其收到的信息，并将其看做是消息和附加的 FCS，B 用相同的函数 F 重新计算 FCS。若计算得到的 FCS 和收到的 FCS 相等，则 B 认为消息是真实的。任何随机的位串不可能产生 M 和 FCS 之间的上述联系。

请注意，FCS 和加密函数执行的顺序很重要。[DIFF79]中将图 11.2(a)所示的这种序列称为内部错误控制，以与外部错误控制[图 11.2(b)]对应。对于内部错误控制，由于攻击者很难产生密文，使得解密后其错误控制位是正确的，因此内部错误控制可以提供认证；如果 FCS 是外部码，那么攻击者可以构造具有正确错误控制码的消息，虽然攻击者不知道解密后的明文是什么，但他可以造成混淆并破坏通信。

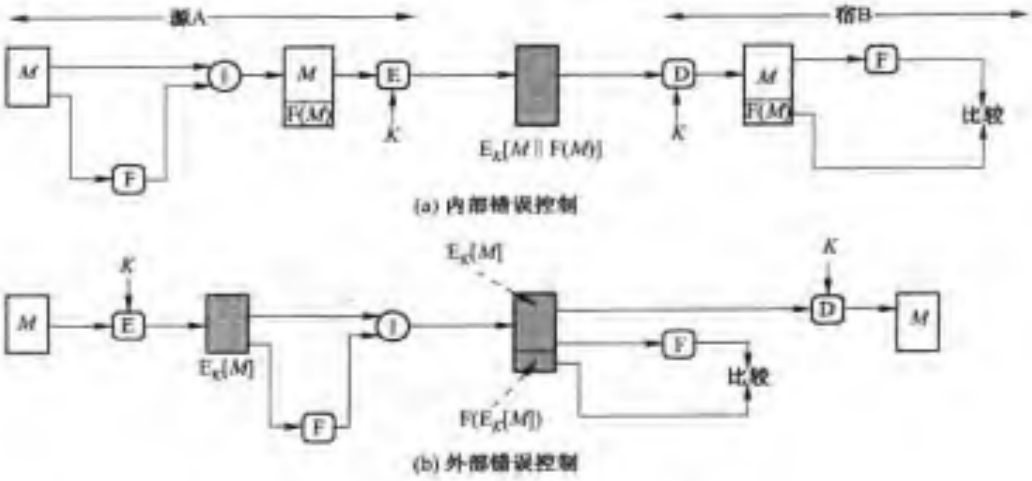


图 11.2 内部和外部错误控制

错误控制码仅是具有上述结构的一个例子。事实上，在要发送的消息中加入任何类型的结构信息都会增强认证能力。分层协议通信体系可以提供这种结构，例如考虑使用 TCP/IP 协议传输的消息的结构，图 11.3 给出的 TCP 段的格式说明了 TCP 报头的结构，假定每对主机共享一个密钥，并且无论是何种应用，每对主机间都使用相同的密钥进行信息交换，那么我们可以对除 IP 报头外的所有数据报加密(见图 7.5)，如果攻击者用一条消息替代加密后的 TCP 段，那么解密后所得出的明文将不等于原 IP 报头。在这种方法中，头不仅包含校验和，而且还含有其他一些有用的信息，如序列号。因为对给定连接，连续的 TCP 段是按顺序编号的，所以加密使得攻击者不能延时、删除任何段或改变段的顺序。

公钥加密

使用公钥加密[图 11.1(b)]可提供保密性，但不能提供认证。发送方(A)使用接收方(B)的公钥 KU_b 对 M 加密，由于只有 B 拥有相应的私钥 KR_b，所以只有 B 能对消息解密。但是任何攻击者可以假冒 A 用 B 的公钥对消息加密，所以这种方法不能保证真实性。

若要提供认证，则 A 用其私钥对消息加密，而 B 用 A 的公钥对接收的消息解密[图 11.1(c)]。因为只有 A 拥有 KR_a，能产生用 KU_a 可解密的密文，所以该消息一定来自 A。同样，对明文也必须有某种内部结构以使接收方能区分真实的明文和随机的位串。

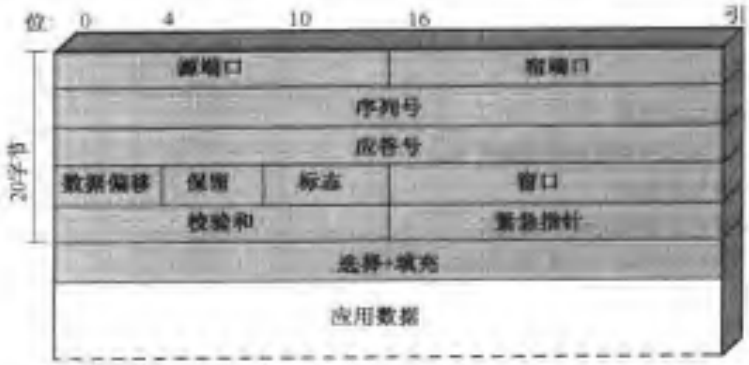


图 11.3　TCP 段

假定明文具有这种结构，那么图 11.1(c)的方法既可提供认证，又可提供数字签名功能[①]。由于只有 A 拥有 KR_a，所以只有 A 能够产生密文，甚至接收方 B 也不能产生密文，因此若 B 接收到密文消息，则 B 可以确认该消息来自 A。事实上，A 通过用其私钥对消息加密来对该消息"签名"。

注意，这种方法不能提供保密性，因为任何拥有 A 的公钥的人都可将密文解密。

如果既要提供保密性又要提供认证，那么 A 可先用其私钥对 M 加密，这就是数字签名；然后 A 用 B 的公钥对上述结果加密，这可保证保密性[图 11.1(d)]。但这种方法的缺点是，一次通信中要执行四次复杂的公钥算法而不是两次。

表 11.1 归纳总结了各种消息加密方法在提供保密性和认证方面的特点。

表 11.1　各种消息加密方法在提供保密性和认证方面的特点(见图 11.1)

$A \rightarrow B: E_K[M]$

- 提供保密性
  - —只有 A 和 B 共享 K
- 提供认证
  - —只能发自 A
  - —传输中未被改变
  - —需要某种数据组织形式/冗余
- 不能提供数字签名
  - —接收方可以伪造消息
  - —发送方可以否认消息

(a)对称加密

$A \rightarrow B: E_{KU_b}[M]$

- 提供保密性
  - —只有 B 拥有用于解密的密钥 KR_b
- 不能提供认证
  - —任何一方都可用 KU_b 对消息加密并假称是 A

(b)公钥(非对称)加密

① 这不是构造数字签名的一般方法，但是我们将会看出，它们的基本原理是相同的。

（续表）

$A \rightarrow B: E_{KR_a}[M]$ ● 提供认证和签名 —只有 A 拥有用于加密的密钥 KR_a —传输中未被改变 —需要某种数据组织形式/冗余 —任何一方可用 KU_a 来验证签名 (c)公钥加密：认证和签名
$A \rightarrow B: E_{KU_b}[E_{KR_a}(M)]$ ● 提供保密性(因为 KU_b) ● 提供认证和签名(因为 KR_a) (d)公钥加密：保密性、认证和签名

11.2.2 消息认证码

消息认证码，又称 MAC，也是一种认证技术，它利用密钥来生成一个固定长度的短数据块，并将该数据块附加在消息之后。在这种方法中假定通信双方，比如 A 和 B，共享密钥 K。若 A 向 B 发送消息时，则 A 计算 MAC，它是消息和密钥的函数，即 $MAC = C_K(M)$，其中：

M = 输入消息
C = MAC 函数
K = 共享的密钥
MAC = 消息认证码

消息和 MAC 一起将被发送给接收方。接收方对收到的消息用相同的密钥 K 进行相同的计算得出新的 MAC，并将接收到的 MAC 与其计算出的 MAC 进行比较[图 11.4(a)]，如果我们假定只有收发双方知道该密钥，那么若接收到的 MAC 与计算得出的 MAC 相等，则有：

1. 接收方可以相信消息未被修改。如果攻击者改变了消息，但他无法改变相应的 MAC，所以接收方计算出的 MAC 将不等于接收到的 MAC。因为我们已假定攻击者不知道密钥，所以他不知道应如何改变 MAC 才能使其与修改后的消息相一致。
2. 接收方可以相信消息来自真正的发送方。因为其他各方均不知道密钥，因此他们不能产生具有正确 MAC 的消息。
3. 如果消息中含有序列号(如 HDLC、X.25 和 TCP 中使用的序列号)，那么接收方可以相信消息顺序是正确的，因为攻击者无法成功地修改序列号。

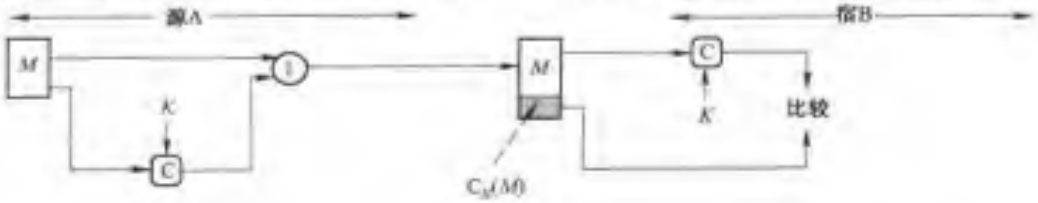
MAC 函数与加密类似。其区别之一是，MAC 算法不要求可逆性，而加密算法必须是可逆的。一般而言，MAC 函数是多对一函数，其定义域由任意长的消息组成，而值域由所有可能的 MAC 和密钥组成。若使用 n 位长的 MAC，则有 2^n 个可能的 MAC，而有 N 条可能的消息，其中 $N >> 2^n$。而且若密钥长为 k，则有 2^k 种可能的密钥。

例如，假定使用 100 位的消息和 10 位的 MAC，那么总共有 2^{100} 种不同的消息，但仅有 2^{10} 种不同的 MAC。所以平均而言，同一 MAC 可以由 $2^{100}/2^{10} = 2^{90}$ 条不同的消息产生。若使用的密钥

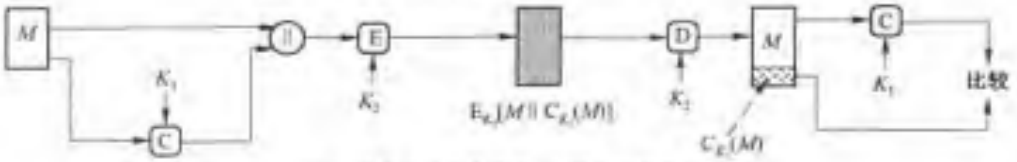
长为 5 位，则从消息集合到 MAC 值的集合有 $2^5 = 32$ 种不同的映射。

可以证明，由于认证函数的数学性质，与加密相比，认证函数更不易被攻破。

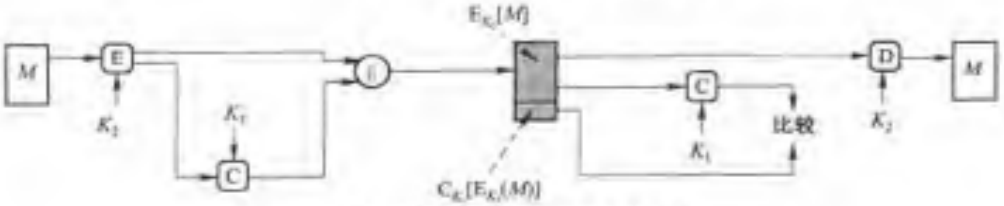
图 11.4(a)所示的过程可以提供认证但不能提供保密性，因为整个消息是以明文形式传送的。若在 MAC 算法之后[图 11.4(b)]或之前[图 11.4(c)]对消息加密，则可以获得保密性。这两种情形都需要两个独立的密钥，并且收发双方共享这两个密钥。在第一种情形中，先将消息作为输入，计算 MAC，并将 MAC 附加在消息后，然后对整个信息块加密；在第二种情形中，先将消息加密，然后将此密文作为输入，计算 MAC，并将 MAC 附加在上述密文之后形成待发送的信息块。一般而言，将 MAC 直接附加于明文之后要更好一些，所以通常使用图 11.4(b)中的方法。



(a) 消息认证



(b) 消息认证和保密性：与明文有关的认证



(c) 消息认证和保密性：与密文有关的认证

图 11.4　消息认证码(MAC)的基本用途

对称加密可以提供认证，且它已被广泛用于现有产品之中，那么为什么不直接使用这种方法而要使用分离的消息认证码呢？[DAVI89]中提出了三种使用消息认证码的情形：

1. 有许多应用是将同一消息广播给很多接收者。例如需要通知各用户网络暂时不可使用，或一个军事控制中心要发一条警报。这种情况下，一种经济可靠的方法就是只要一个接收者负责验证消息的真实性，所以消息必须以明文加上消息认证码的形式进行广播。上述负责验证的接收者拥有密钥并执行认证过程，若 MAC 错误，则他发警报通知其他各接收者。
2. 在信息交换中，可能有这样一种情况，即通信某一方的处理负荷很大，没有时间解密收到的所有消息，他应能随机选择消息并对其进行认证。
3. 对明文形式的计算机程序进行认证是一种很有意义的服务。运行一个计算机程序而不必每次对其解密，因为每次对其解密会浪费处理器资源。若将消息认证码附于该程序之后，则可在需要保证程序完整性的时候才检验消息认证码。

除此以外，还有下述三种情形：

4. 一些应用并不关心消息的保密性，而关心消息认证。例如简单网络管理协议版本3(SNMPv3)就是如此，它将提供保密性和提供认证分离开来。对这些应用，管理系统应对其收到的 SNMP 消息进行认证，这一点非常重要，尤其是当消息中包含修改系统参数的命令时更是如此，但对这些应用不必伪装 SNMP 传输。
5. 将认证和保密性分离开来，可使层次结构更加灵活。例如，在应用层我们可能希望对消息进行认证，而在更低层上，如传输层，我们可能希望提供保密性。
6. 仅在接收消息期间对消息实施保护是不够的，用户可能希望延长对消息的保护时间。就消息加密而言，消息被解密后就不再受任何保护，这样只是在传输中可以使消息不被修改，而不是在接收方系统中保护消息不被修改。

由于收发双方共享密钥，因此 MAC 不能提供数字签名。

表 11.2 归纳总结了图 11.4 中所示的各种方法在提供保密性和认证方面的特点。

表 11.2 消息认证码的基本用途(见图 11.4)

$A \rightarrow B: M \| C_K(M)$ ● 提供认证 —只有 A 和 B 共享 K (a)消息认证
$A \rightarrow B: E_{K_2}[M \| C_{K_1}(M)]$ ● 提供认证 —只有 A 和 B 共享 K_1 ● 提供保密性 —只有 A 和 B 共享 K_2 (b)消息认证和保密性：与明文有关的认证
$A \rightarrow B: E_{K_2}[M] \| C_{K_1}(E_{K_2}[M])$ ● 提供认证 —使用 K_1 ● 提供保密性 —使用 K_2 (c)消息认证和保密性：与密文有关的认证

11.2.3 hash 函数

单向 hash 函数是消息认证码的一种变形。与消息认证码一样，hash 函数的输入是可变大小的消息 M，输出是固定大小的 hash 码 $H(M)$。与 MAC 不同的是，hash 码并不使用密钥，它仅是输入消息的函数。hash 码有时也称为消息摘要，或 hash 值。hash 码是所有消息位的函数，它具有错误检测能力，即改变消息的任何一位或多位，都会导致 hash 码的改变。

图 11.5 给出了将 hash 码用于消息认证的各种方法，如下所述：

a. 用对称密码对消息及附加在其后的 hash 码加密。这种方法的结构与图 11.2(a)所示的内部错误控制码方法相同。由于只有 A 和 B 共享密钥，所以消息一定是来自 A 且未被

修改过。hash 码提供了认证所需的结构或冗余，并且由于该方法是对整个消息和 hash 码加密，所以也提供了保密性。

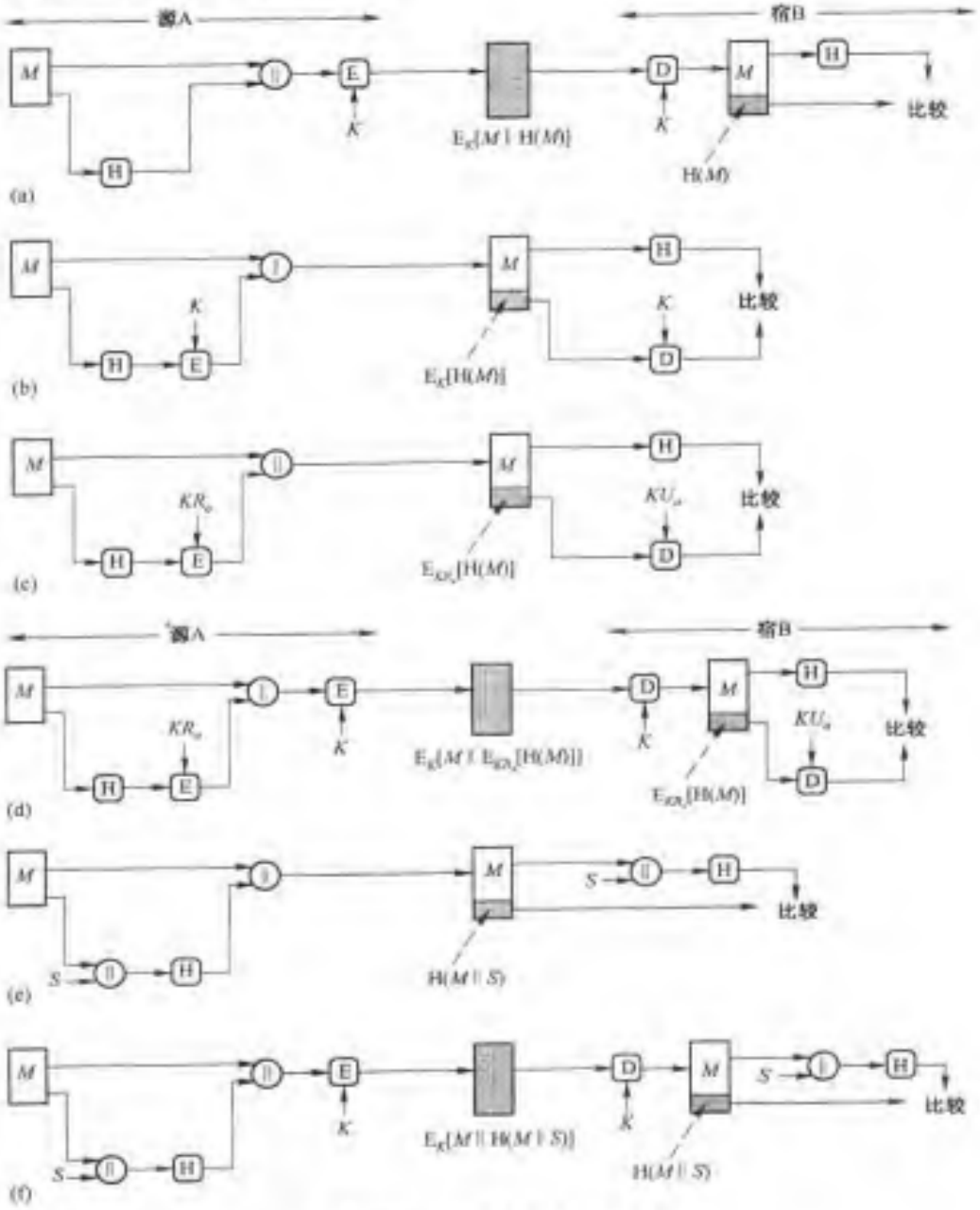


图 11.5　hash 函数的基本用途

b. 用对称密码仅对 hash 加密。对那些不要求保密性的应用，这种方法会减少处理代价。注意，hash 函数和加密函数的合成函数即是 MAC[图 11.4(a)]。也就是说，$E_K[H(M)]$ 是变长消息 M 和密钥 K 的函数，它产生定长的输出值，若攻击者不知道密钥，则他无法得出这个值。

c. 用公钥密码和发送方的私钥仅对 hash 码加密。同(b)一样，这种方法可提供认证；由于

只有发送方可以产生加密后的 hash 码，所以这种方法也提供了数字签名。事实上，这就是数字签名技术的本质所在。

d. 若既希望保证保密性又希望有数字签名，则先用发送方的私钥对 hash 码加密，再用对称密码中的密钥对消息和上述加密结果进行加密。这种技术比较常用。

e. 该方法使用 hash 函数但不使用加密函数。这里，假定通信双方共享公共的秘密值 S，A 将 M 和 S 联接后再计算 hash 值，并将其附于 M 后。由于 B 也知道 S，所以 B 可以计算 hash 值，并验证其正确性。由于秘密值本身并不传送，所以攻击者无法修改所截获的消息，也不能伪造消息。

f. 如果对整个消息和 hash 码加密，则(e)中的方法可提供保密性。

如果不要求保证保密性，那么由于(b)和(c)中所需的计算较少，而且人们越来越对那些不含加密函数的方法感兴趣[图 11.5(e)]，所以(b)和(c)比那些对整条消息加密的方法要好一些。[TSUD92]中给出了几种不希望使用加密函数的理由：

- 加密软件速度慢。即使每条消息需要加密的数据量不大，但是总有消息串会输入到加密系统中或由系统输出。
- 加密硬件成本不容忽视。尽管已有实现 DES 的低成本芯片，但是若网络中所有节点都必须有该硬件，则总成本可能很大。
- 加密硬件的优化是针对大数据块的。对小数据块，大量时间花费在初始化/调用之上。
- 加密算法可能受专利保护。某些加密算法，如 RSA 公钥算法，已申请专利，必须经批准后才能使用，这也会增加成本。
- 加密算法受美国出口控制的限制。

表 11.3 总结了图 11.5 中所示的各种方法在提供保密性和认证方面的特点。下面我们来更详细地讨论 MAC 和 hash 码。

表 11.3 hash 函数的基本用途

$A \rightarrow B: E_k[M \| H(M)]$ ● 提供保密性 —只有 A 和 B 共享 K ● 提供认证 —H(M)受密码保护 (a)加密消息及 hash 码	$A \rightarrow B: E_K[M \| E_{KR_a}[H(M)]]$ ● 提供认证和数字签名 ● 提供保密性 —只有 A 和 B 共享 K (d)加密(c)的结果—共享的密钥
$A \rightarrow B: M \| E_K[H(M)]$ ● 提供认证 —H(M)受密码保护 (b)加密 *hash* 码——共享的密钥	$A \rightarrow B: M \| H(M \| S)$ ● 提供认证 —只有 A 和 B 共享 S (e)计算消息和秘密值的 *hash* 码
$A \rightarrow B: M \| E_{KR_a}[H(M)]$ ● 提供认证和数字签名 —H(M)受密码保护 —只有 A 能产生 $E_{KR_a}[H(M)]$ (c)加密 *hash* 码——发送方私钥	$A \rightarrow B: E_K[M \| H(M) \| S]$ ● 提供认证 —只有 A 和 B 共享 S ● 提供保密性 —只有 A 和 B 共享 K (f)加密(e)的结果

11.3　消息认证码

MAC 也称密码校验和，它由如下形式的函数 C 产生：

$$\mathrm{MAC} = C_K(M)$$

其中 M 是一个变长消息，K 是收发双方共享的密钥，$C_K(M)$是定长的认证符。在假定或已知消息正确时，将 MAC 附于发送方的消息之后；接收方可通过计算 MAC 来认证该消息。

本节我们先介绍函数 C 应满足的条件，然后讨论一个消息认证的例子，其他例子将在第 12 章中讨论。

11.3.1　对 MAC 的要求

为了获得保密性，可用对称或非对称密码对整个消息加密，这种方法的安全性一般依赖于密钥的位长。除了算法中本身的某些弱点外，攻击者可以对所有可能的密钥进行穷举攻击。一般对 k 位的密钥，穷举攻击需要 $2^{(k-1)}$ 步，特别地，对仅依赖于明文的攻击，若给定密文 C，攻击者要对所有可能的 K_i 计算 $P_i = D_{K_i}(C)$，直到产生的某 P_i 具有适当的明文结构为止。

对 MAC 情况则完全不一样。一般地，MAC 函数是多对一函数。攻击者如何用穷举方法找到密钥呢？如果没有提供保密性，那么攻击者可访问明文形式的消息及其 MAC。假定 $k > n$，即假定密钥位数比 MAC 长，那么，对满足 $\mathrm{MAC}_1 = C_{K_i}(M_1)$的 M_1 和 MAC_1，密码分析者要对所有可能的密钥值 K_i 计算 $\mathrm{MAC}_i = C_{K_i}(M_1)$，那么至少有一个密钥会使得 $\mathrm{MAC}_i = \mathrm{MAC}_1$。注意，这里总共会产生 2^k 个 MAC，但只有 $2^n < 2^k$ 个不同的 MAC 值，所以许多密钥都会产生正确的 MAC，而攻击者却不知哪一个是正确的密钥。平均来说，有 $2^k/2^n = 2^{(k-n)}$个密钥会产生正确的 MAC，因此攻击者必须重复下述攻击：

- **循环 1**

  给定 M_1，$\mathrm{MAC}_1 = C_K(M_1)$

  对所有 2^k 个密钥，判断 $\mathrm{MAC}_i = C_{K_i}(M_1)$

  匹配数 $\approx 2^{(k-n)}$

- **循环 2**

  给定 M_2，$\mathrm{MAC}_2 = C_K(M_2)$

  对余下的 $2^{(k-n)}$个密钥判断 $\mathrm{MAC}_i = C_{K_i}(M_2)$

  匹配数 $\approx 2^{(k-2\times n)}$

等等。平均来讲，若 $k = \alpha \times n$，则需 α 次循环。例如，如果使用 80 位的密钥和长为 32 位的 MAC，那么第一次循环会得到约 2^{48}个可能的密钥，第二次循环会得到约 2^{16}个可能的密钥，第三次循环则得到惟一一个密钥，这个密钥就是发送方所使用的密钥。

如果密钥的长度小于或等于 MAC 的长度，则很可能第一次循环中就得到一个密钥，当然也可能得到多个密钥，这时攻击者还需对新的(消息，MAC)对执行上述测试。

由此可见，用穷举方法来确定认证密钥不是一件容易的事，而且确定认证密钥比确定同样

长度的加密密钥更困难。不过可能存在有不需要去寻找密钥的其他攻击。

考虑下面的 MAC 算法。令消息 $M = (X_1 \| X_2 \| \cdots \| X_m)$是由 64 位分组 X_i 连接而成。定义：

$$\Delta(M) = X_1 \oplus X_2 \oplus \cdots \oplus X_m$$
$$C_K(M) = E_K[\Delta(M)]$$

其中⊕是异或(XOR)运算，加密算法是电子密码本方式 DES，那么密钥长为 56 位，MAC 长为 64 位。若攻击者知道$\{M \| C_K(M)\}$，则确定 K 的穷举攻击需执行至少 2^{56}次加密，但是攻击者可以用任何期望的 Y_1 至 Y_{m-1}替代 X_1 至 X_{m-1}，用 Y_m 替代 X_m 来进行攻击，其中 Y_m 是如下计算的：

$$Y_m = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{m-1} \oplus \Delta(M)$$

攻击者可以将 Y_1 至 Y_m与原来的 MAC 连接成一个新的消息，而接收方却会认为该消息是真实的。用这种办法，攻击者可以随意插入任意长为 $64 \times (m-1)$位的消息。

因此，评价 MAC 函数的安全性时，我们应考虑对该函数的各种类型的攻击。下面我们介绍 MAC 函数应满足的要求。假定攻击者知道 MAC 函数 C，但不知道 K，那么 MAC 函数应具有下列性质：

1. 若攻击者已知 M 和 $C_K(M)$，则他构造满足 $C_K(M') = C_K(M)$的消息 M'在计算上是不可行的。
2. $C_K(M)$应是均匀分布的，即对任何随机选择的消息 M 和 M'，$C_K(M) = C_K(M')$的概率是 2^{-n}，其中 n 是 MAC 的位数。
3. 设 M'是 M 的某个已知的变换，即 $M' = f(M)$。例如，f 可表示逆转 M 的一位或多位，那么 $\Pr[C_K(M) = C_K(M')] = 2^{-n}$。

前面已讲过，攻击者即使不知道密钥，也可以构造出与给定的 MAC 匹配的新消息，第一个要求就是针对这种情况提出的。第二个要求是为了阻止基于选择明文的穷举攻击，也就是说，假定攻击者不知道 K，但是他可以访问 MAC 函数，能对消息产生 MAC，那么攻击者可以对各种消息计算 MAC，直至找到与给定 MAC 相同的消息为止。如果 MAC 函数具有均匀分布的特征，那么穷举方法平均需要 $2^{(n-1)}$步才能找到具有给定 MAC 的消息。

最后一条要求，认证算法对消息的某部分或位不应比其他部分或位更弱。否则，已知 M 和 $C_K(M)$的攻击者可以对 M 的已知"弱点"处进行修改，然后再计算 MAC，这样有可能更早得出具有给定 MAC 的新消息。

11.3.2 基于 DES 的消息认证码

数据认证算法(FIPS PUB 113)建立在 DES 之上，是使用最广泛的 MAC 算法之一，它也是一个 ANSI 标准(X9.17)。

数据认证算法采用 DES 运算的密文块链接(CBC)方式(见图 3.12)，其初始矢量为 0，需要认证的数据(如消息、记录、文件或程序)分成连续的 64 位的分组 $D_1, D_2, \cdots, D_N$，若最后分组不足 64 位，则在其后填 0 直至成为 64 位的分组。利用 DES 加密算法 E 和密钥 K，计算数据认证码(DAC)的过程如图 11.6 所示。

$$O_1 = \mathrm{E}_K(D_1)$$
$$O_2 = \mathrm{E}_K(D_2 \oplus O_1)$$
$$O_3 = \mathrm{E}_K(D_3 \oplus O_2)$$
$$\vdots$$
$$O_N = \mathrm{E}_K(D_N \oplus O_{N-1})$$

DAC 可以是整个块 O_N，也可以是其最左边的 M 位，其中 $16 \leqslant M \leqslant 64$。

数据认证算法似乎可以满足前面提出的要求。

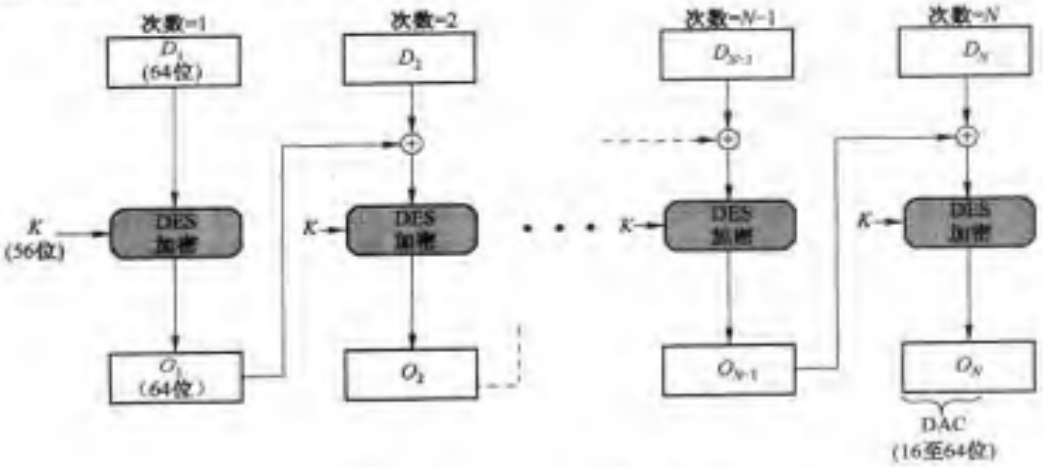


图 11.6　数据认证算法(FIPS PUB 113)

11.4　hash 函数

hash 值 h 由下述形式的函数 H 生成：

$$h = \mathrm{H}(M)$$

其中 M 是一个变长消息，$\mathrm{H}(M)$是定长的 hash 值。消息正确时，将 hash 值附于发送方的消息后；接收方通过重新计算 hash 值可认证该消息。由于 hash 函数本身是不保密的，所以需要有某些方法来保护 hash 值(见图 11.5)。

我们首先讨论用于消息认证的 hash 函数应满足的要求，因为 hash 函数一般很复杂，所以讨论一些简单的 hash 函数有助于理解 hash 函数有关的内容；然后我们介绍 hash 函数设计的几种方法。

11.4.1　对 hash 函数的要求

hash 函数的目的就是要产生文件、消息或其他数据块的“指纹”。hash 函数要能够用于消息认证，它必须具有下列性质(这些性质摘自[NECH92])：

1. H 可应用于任意大小的数据块。
2. H 产生定长的输出。
3. 对任何给定的 x，计算 $\mathrm{H}(x)$比较容易，用硬件和软件均可实现。
4. 对任何给定的 hash 值 h，找到满足 $\mathrm{H}(x) = h$ 的 x 在计算上是不可行的，有些文献中称之为**单向性**。

5. 对任何给定的分组 x，找到满足 $y \neq x$ 且 $H(y) = H(x)$ 的 y 在计算上是不可行的，我们有时称之为**抗弱碰撞性**。
6. 找到任何满足 $H(x) = H(y)$ 的偶对 (x, y) 在计算上是不可行的。我们有时称之为**抗碰撞性**[①]。

前三个条件是 hash 函数实际应用于消息认证中所必须满足的。

第四个条件单向性是指，由消息很容易计算出 hash 码，但是由 hash 码却不能计算出相应的消息。对使用一个秘密值的认证方法[见图 11.5(e)]，这个性质非常重要，虽然该秘密值本身并不传送，但若 hash 函数不是单向的，则攻击者可以很容易地找出这个秘密值：若攻击者能够观察或截获到传送的消息，则他可以得到消息 M 和 hash 码 $C = H(S_{AB} \| M)$，然后求出 hash 函数的逆，从而得出 $S_{AB} \| M = H^{-1}(C)$，由于攻击者已知 M 和 $S_{AB} \| M$，所以可得出 S_{AB}。

第五条性质可以保证，不能找到与给定消息具有相同 hash 值的另一消息，因此可以在使用对 hash 码加密的方法中防止伪造[见图 11.5(b)和图 11.5(c)]。在这些方法中，攻击者可以读取消息并产生其 hash 码，但是由于攻击者不知道密钥，所以他不可能改变消息而又不被察觉。如果第五条不成立，那么攻击者可以先观察或截获一条消息及其加密的 hash 码，然后由消息产生一个未加密的 hash 码，最后产生另一个具有相同 hash 码的消息。

第六条性质涉及 hash 函数抗生日攻击这类攻击的能力强弱问题。我们在后面会讨论生日攻击。

11.4.2 简单 hash 函数

所有hash函数的输入(消息、文件等)都可看做是一个 n 位分组的序列，其输出是 n 位的 hash 值。hash 函数每次处理一个分组，重复该过程直至处理完所有的输入分组。

最简单的 hash 函数之一是将每个分组逐位异或(XOR)，这个函数可如下描述：

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

其中：

C_i = hash 码的第 i 位，$1 \leqslant i \leqslant n$
m = n 位输入块的个数
b_{ij} = 第 j 块的第 i 位
$\oplus$ = 异或运算

一种简单的改进方法是，每处理完一个分组后将 hash 值循环移位一次。图 11.7 说明了上述运算过程，它对每一位产生一个简单的奇偶校验，我们称之为经度冗余校验，这种方法对将随机数用于数据完整性检查比较有效。如果每个 n 位 hash 值出现的概率都相同，那么数据出错而不引起 hash 值改变的概率为 2^{-n}；若数据格式不是随机的，则会降低函数的有效性，例如，通常大多数文本文件中每个8位字节的高位总为0，若使用128位的hash值，则对这类数据，hash

① 这些述语的定义不惟一。有些文献中定义为单向 hash 函数(性质 4 和 5)；具有抗碰撞能力的 hash 函数(性质 4、5 和 6)；弱单向 hash 函数(性质 4 和 5)；强单向 hash 函数(性质 4、5 和 6)。读者在阅读文献时应注意所使用的术语的特定含义。

函数的有效性是 2^{-112} 而不是 2^{-128}。

	位1	位2	...	位n
分组1	b_{11}	b_{21}		b_{n1}
分组2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
分组m	b_{1m}	b_{2m}		b_{nm}
hash码	C_1	C_2		C_n

图 11.7　使用按位异或的简单 hash 函数

这个过程可归纳如下：

1. n 位 hash 值的初值为 0。
2. 如下处理每个 n 位的分组：
   a. 将当前的 hash 值循环左移一次。
   b. 将该分组与 hash 值异或。

这样，可使输入更加完全地“随机”，从而消除输入数据的规则性。图 11.8 给出了两种产生 16 位 hash 值的 hash 函数。

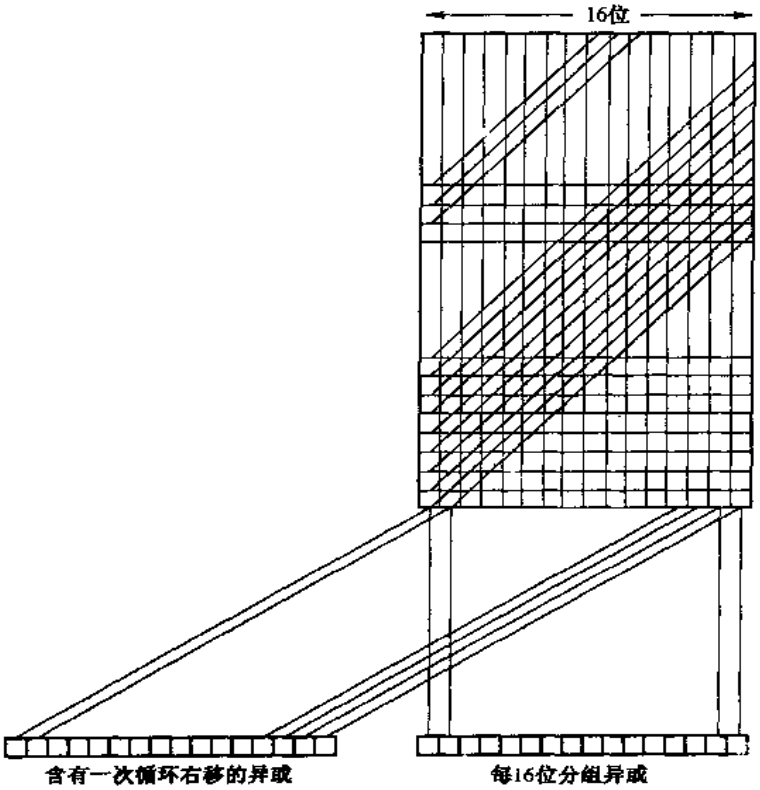


图 11.8　两个简单的 hash 函数

虽然这种改进的方法可以很好地保证数据的完整性，但是如果使用图 11.5(b)和图 11.5(c)中所示的方法，即将加密后的 hash 码附加在明文之后，那么该方法不能保证数据的安全性。

因为很容易产生一条新消息，使它与给定的消息具有相同的 hash 码：先选定某消息，然后在其后附加一个 n 位分组，使它们与给定的消息具有相同的 hash 码。

如果只对 hash 码加密，那么上述简单异或或循环异或(RXOR)方法不能保证数据的安全性，但是如果对消息和 hash 码均加密，那么这些方法仍然可能存在问题[见图 11.5(a)]。美国国家标准局最初提出了一种方法，这种方法对 64 位的分组执行简单异或操作，使用密文块链接(CBC)方式对整个消息加密。给定消息 $X_1, X_2, \cdots, X_N$，其中 X_i 是 64 位的分组，其 hash 码 C 为所有分组的异或，并且将该 hash 码作为最后一个分组：

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

然后使用 CBC 方式对消息和 hash 码加密得到 $Y_1, Y_2, \cdots, Y_{N+1}$。文献[JUEN85]中给出了几种改变消息密文而 hash 码无法检测的攻击。例如，根据 CBC 的定义(见图 3.12)，我们有：

$$\begin{aligned} X_1 &= IV \oplus \mathrm{D}_K(Y_1) \\ X_i &= Y_{i-1} \oplus D_K(Y_i) \\ X_{N+1} &= \mathrm{Y}_N \oplus \mathrm{D}_K(Y_{N+1}) \end{aligned}$$

且 hash 码 X_{N+1} 为：

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \cdots \oplus X_N \\ &= (IV \oplus \mathrm{D}_K(Y_1)) \oplus (Y_1 \oplus \mathrm{D}_K(Y_2)) \oplus \cdots \oplus (Y_{N-1} \oplus \mathrm{D}_K(Y_N)) \end{aligned}$$

由于上述等式中异或可以按任意顺序计算，所以改变密文分组的顺序，hash 码仍然不变。

11.4.3 生日攻击

有人可能认为使用 64 位的 hash 码会很安全。例如，如果将加密后的 hash 码 C 与未加密的消息 M 一起传输[见图 11.5(b)或图 11.5(c)]，那么攻击者必须找到满足 $\mathrm{H}(M') = \mathrm{H}(M)$ 的 M' 来替代 M。平均来讲，攻击者要找到这样的消息大约需要进行 2^{63} 次尝试[参见附录 11A，等式(11.1)]。

但是，可能有另一种类型的攻击，这种攻击建立在生日悖论的基础之上(参见附录11A)。Yuval 提出了以下对策[YUVA79]：

1. 发送方 A 准备对消息“签名”，其使用的方法是，用其私钥对 m 位的 hash 码加密并将加密后的 hash 码附于消息之后[见图 11.5(c)]。
2. 攻击者产生该消息的 $2^{m/2}$ 种变式，且每一种变式表达相同的意义。攻击者再伪造一条消息，并产生该伪造消息的 $2^{m/2}$ 种变式，攻击者准备用该伪造消息替代真实消息。
3. 比较上述两个集合，找出产生相同 hash 码的一对消息。根据生日悖论，找到这对消息的概率大于 0.5，如果找不到这样的消息，那么再产生一条伪造的消息直至成功为止。
4. 攻击者将该有效变式提供给 A 签名，将该签名附于伪造消息的有效变式后并发送给预期的接收方。因为上述两个变式的 hash 码相同，所以它们产生的签名也相同，因此攻击者即使不知道加密密钥也能攻击成功。

这样，如果使用 64 位的 hash 码，那么所需代价的数量级仅为 2^{32}[见附录 11A，等式(11.7)]。

产生多个具有相同意义的变式并不困难。例如，攻击者可以在文件的词与词之间插入若

干“空格 – 空格 – 退格”字符对，然后在实例中用“空格 – 退格 – 空格”替代这些字符从而产生各种变式。攻击者也可以简单地改变消息中词的顺序但不改变消息的意义，图 11.9[DAVI89]举例说明了这种方法。

由此可以得出结论，hash 码的长度应该较长，我们将在 11.5 节中进一步讨论这个问题。

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}
Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}
Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}
responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}
in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}
to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the
market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the
{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}
of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}
as proof of identity. An order with his signature, which is {appended / attached}
{authorizes / allows} you to charge the cost to this company at the {above / head office}
address. We {fully / --} expect that our {level / volume} of orders will increase in
the {following / next} year and {trust / hope} that the new appointment will {be / prove}
{advantageous / an advantage} to both our companies.

图 11.9　具有 2^{37} 种变式的信[DAVI89]

11.4.4　分组链接技术

人们已经提出的 hash 函数中，许多基于密文分组链接方法，但没有密钥。Rabin 所提出的方法[RABI78]，将消息 M 分成固定大小的分组 $M_1, M_2, \cdots, M_N$，并使用对称加密体制，如 DES，计算其 hash 码 G：

$$H_0 = \text{初始值}$$
$$H_i = E_{M_i}[H_{i-1}]$$
$$G = H_N$$

这种方法类似于 CBC 方法，但不使用密钥。像任何 hash 码一样，这种方法也易受到生日攻击，如果使用 DES 加密算法而且仅产生 64 位 hash 码，那么系统会非常脆弱。

即使攻击者只能截获一条消息及其签名，他仍然可以进行另一种形式的生日攻击。例如，假定攻击者截获了一条已签名的消息，该签名是加密后的 hash 码，加密前 hash 码长度为 m 位：

1. 用本小节开始部分定义的算法计算该未加密的 hash 码 G。
2. 构造任何形为 $Q_1, Q_2, \cdots, Q_{N-2}$的消息。
3. 对 $1 \leqslant i \leqslant (N-2)$，计算 $H_i = \mathrm{E}_{Q_i}[H_{i-1}]$。
4. 产生 $2^{m/2}$个随机分组，对每一分组 X，计算 $E_X[H_{N-2}]$。再产生 $2^{m/2}$个随机分组，对每一分组 Y，计算 $\mathrm{D}_Y[G]$，其中 D 是对应 E 的解密函数。
5. 根据生日悖论，X 和 Y 满足 $E_X[H_{N-2}] = D_Y[G]$的概率较大。
6. 构造消息 $Q_1, Q_2, \cdots, Q_{N-2}, X, Y$。因该消息的 hash 码也为 G，因此可以将该消息与截获的签名一起发送。

这种形式的攻击称为"中间相遇"攻击。为了加强密文分组链接方法的抗攻击能力，许多研究者提出了修改建议。例如，Davies 和 Price[DAVI89]提出了下列修改方法：

$$H_i = \mathrm{E}_{M_i}[H_{i-1}] \oplus H_{i-1}$$

[MEYE88]中提出的修改方法是：

$$H_i = \mathrm{E}_{H_{i-1}}[M_i] \oplus M_i$$

已经证明上述两种方法都易受到各种攻击[MIYA90]。更一般地，可以证明，只要 hash 码较短(如不超过 64 位)，或者 hash 码很长但可分解为独立的子码，就存在有生日攻击，它能成功地攻击任何使用密文分组链接但不使用密钥的 hash 方法[JUEN87]。

因此我们需要其他的 hash 方法。但是已经证明，许多这种方法都存在有一些弱点[MITC92]。我们将在第 12 章中讨论强 hash 函数。

11.5 hash 函数和 MAC 的安全性

与对称密码和公钥密码一样，我们也可将对 hash 函数和 MAC 的攻击分为两类：穷举攻击和密码分析。

11.5.1 穷举攻击

对 hash 函数的穷举攻击与对 MAC 的穷举攻击有所不同。

hash 函数

hash 函数抗穷举攻击的能力仅仅依赖于算法所产生的 hash 码的长度。前面我们曾讨论了 hash 函数应满足的三个性质：

1. **单向性**：对任何给定的 hash 码 h，找到满足 $\mathrm{H}(x) = h$ 的 x 在计算上是不可行的。
2. **抗弱碰撞性**：对任何给定的块 x，找到满足 $y \neq x$ 且 $\mathrm{H}(y) = \mathrm{H}(x)$的 y 在计算上是不可行的。
3. **抗强碰撞性**：找到任何满足 $\mathrm{H}(x) = \mathrm{H}(y)$的偶对$(x, y)$在计算上是不可行的。

我们知道，对长度为 n 的 hash 码，找到上述性质的元素所需的代价分别与下表中的相应量成正比：

单 向 性	2^n
抗弱碰撞性	2^n
抗强碰撞性	$2^{n/2}$

如果要求抗强碰撞能力(通常期望安全 hash 码具有该性质),那么值 $2^{n/2}$ 决定了该 hash 码抗穷举攻击的强度。Oorschot 和 Wiener[OORS94]耗资一千万美元,为攻击 MD5 设计了一台碰撞搜寻机器,它能在 24 天内找到一个碰撞。MD5 使用的是 128 位 hash 码,因此一般认为 128 位hash 码是不够的。如果将 hash 码看做是一串 32 位字,那么以后将要使用 160 位 hash 码。对 160 位 hash 码,用相同的搜寻机器则需要四千年才能找到一个碰撞。我们在第 12 章中将讨论 SHA-1 和 RIPEMD-160,这两个最流行的 hash 码都是 160 位。

消息认证码

对 MAC 的穷举攻击由于需要知道消息 – MAC 对,所以这种攻击会更加困难。下面我们来分析其原因。攻击者可以按下述方式对 hash 码进行攻击:对给定的消息 x,其 n 位 hash 码 $h = H(x)$,寻找碰撞的穷举攻击方法可以随机挑选一个位串 y,检查是否有 $H(y) = H(x)$。攻击者可以以离线方式重复上述操作,但对 MAC 算法是否能使用离线攻击则依赖于密钥和 MAC 的长度。

在进一步讨论之前,我们先讨论 MAC 算法所应具有的安全性质,该性质可描述如下:

- **抗计算性**:给定一个或多个消息 – MAC 对$(x_i, C_K(x_i))$,对任何新的输入 $x \neq x_i$,计算消息 – MAC 对$(x, C_K(x))$在计算上是不可行的。

换句话说,对给定的消息 x,攻击者可通过攻击密钥空间和攻击 MAC 值这两种方法来找出其 MAC。下面我们来讨论这些攻击。

如果攻击者能够确定 MAC 密钥,那么他就可以对任何输入 x 产生有效的 MAC。假定密钥长为 n,并且攻击者已知一个消息 – MAC 对,那么攻击者可用所有可能的密钥对该消息计算其 n 位的 MAC,这样至少有一个密钥会产生正确的 MAC,该密钥就是原来用来产生该消息 – MAC对的密钥,此处所需的代价与 2^k 成正比(即对 2^k 个可能的密钥中的每一个执行一次操作)。但是,如前所说,因为 MAC 是多对一映射,所以可能有其他一些密钥也会产生正确的 MAC 值,因此,如果不止一个密钥产生正确值,那么必须检查其他一些消息 – MAC 对。可以证明,检查这些消息 – MAC 对所需的代价会迅速减小,并且总的代价约为 2^k[MENE97]。

攻击者也可以攻击 MAC 而不试图去找出密钥,这种攻击的目的是,对给定的消息产生其有效的 MAC,或者对给定的 MAC 产生相应的消息。这两种情形中,其代价为 2^n,与攻击具有单向性或抗弱碰撞能力的 hash 码所需的代价相同。对 MAC 攻击时,因为攻击者需要有已选择的消息 – MAC 对或者已知密钥,所以这种攻击不能离线进行。

总之,对 MAC 算法的穷举攻击所需的代价为 $\min(2^k, 2^n)$。其强度的评价与对称密码算法中的讨论类似。密钥长度和 MAC 的长度应满足关系式 $\min(k, n) \geq N$,其中 N 可以为 128 位。

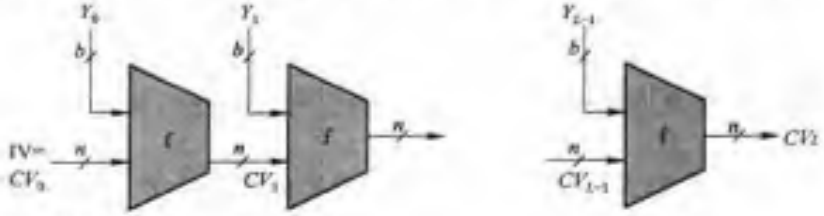
11.5.2　密码分析

与对密码算法的攻击一样,对 hash 函数和 MAC 算法的密码分析攻击,也是利用算法的某种性质而不是通过穷举来进行攻击的。评价 hash 或 MAC 算法抗密码分析能力的方法是,将

其与穷举攻击所需的代价相比，也就是说，理想的 hash 函数和 MAC 算法要求密码分析攻击所需的代价大于或等于穷举攻击所需的代价。

hash 函数

最近这些年，人们在研究对 hash 函数的密码分析攻击方面做了大量的工作，其中有些攻击是成功的。讨论这些问题之前，我们需要先讨论一下典型的安全 hash 函数的一般结构，如图 11.10 所示。这种结构称为迭代 hash 函数，它是由 Merkle[MERK79, MERK89]提出的。包括 MD5、SHA-1 和 RIPEMD-160 在内的目前所使用的大多数 hash 函数也是这种结构，我们将在第 12 章中讨论这些 hash 函数。hash 函数将输入消息分为 L 个固定长度的分组，每一分组长为 b 位，最后一个分组不足 b 位时需要将其填充为 b 位，最后一个分组为输入的总长度。由于输入中包含长度，所以攻击者必须找出具有相同 hash 值且长度相等的两条消息，或者找出两条长度不等但加入消息长度后 hash 值相同的消息，从而增加了攻击的难度。



IV = 初始值
CV = 链接变量
Y_i = 第 i 个输入分组
f = 压缩算法
L = 输入分组数
n = hash 码的长度
b = 输入分组的长度

图 11.10 安全 hash 码的一般结构

hash 算法中重复使用了压缩函数 f，它的输入是前一步中得出的 n 位结果（称为链接变量）和一个 b 位分组，输出为一个 n 位分组。链接变量的初值由算法在开始时指定，其终值即为 hash 值。通常 $b > n$，因此成为压缩。hash 函数可归纳如下：

$$CV_0 = IV = \text{初始}n\text{位值}$$
$$CV_i = \mathrm{f}(CV_{i-1}, Y_{i-1}) \quad 1 \leqslant i \leqslant L$$
$$\mathrm{H}(M) = CV_L$$

其中 hash 函数的输入为消息 M，它由分组 $Y_0, Y_1, \cdots, Y_{L-1}$ 组成。

Merkle[MERK89]和 Damgard[DAMG89]发现，如果压缩函数具有抗碰撞能力，那么迭代 hash 函数也具有抗碰撞能力[①]。因此，hash 函数常使用上述迭代结构，这种结构可用于对任意长度的消息产生安全 hash 函数。由此可见，设计安全 hash 函数可归纳为设计具有抗碰撞能力的压缩函数问题，并且该压缩函数的输入是定长的。

对 hash 函数的密码分析主要集中在对 f 的内部结构进行分析，并试图找到能与 f 的执行

① 其逆不一定为真。

产生碰撞的有效方法，如果能找到这种方法，那么它必定要考虑固定值 IV。像对称分组密码一样，f 由若干轮组成，因此要攻击 f，攻击者就必须分析轮与轮之间信息变化的规律。

因为我们是将长度至少为分组大小 b 的消息映射为长度为 n 的 hash 码，其中 $b > n$，所以任何 hash 函数都存在有碰撞，因此应该要求找出碰撞在计算上是不可行的。

对 hash 函数的攻击相当复杂，这些内容已超出了我们的讨论范围，有兴趣的读者可参阅[DOBB96a]和[BELL97]。

消息认证码

与 hash 函数相比，MAC 的结构种类更多，而且对 MAC 的密码分析攻击的研究很少，所以很难归纳总结对 MAC 的密码分析。[PREN96]中概述了对某些特定 MAC 的密码分析攻击。

11.6　推荐读物

[JUEN85]和[JUEN87]给出了消息认证的背景知识，集中讨论了 MAC 和 hash 函数；[STIN02]和[MENE97]也详细地讨论了 hash 函数和消息认证码；新近发表的[PREN99]对此做了较全面的综述。

JUEN85　Jueneman, R.; Matyas, S.; and Meyer, C. "Message Authentication." *IEEE Communications Magazine*, September 1988.

JUEN87　Jueneman, R. "Electronic Document Authentication." *IEEE Network Magazine*, April 1987.

MENE97　Menezes, A.; Oorshcot, P.; and Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

PREN99　Preneel, B. "The State of Cryptographic Hash Functions." *Proceedings*, *EUROCRYPT'96*, 1996; published by Springer-Verlag.

STIN02　Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2002.

11.7　关键术语、思考题和习题

11.7.1　关键术语

认证符	密码校验和	消息摘要
生日攻击	hash 函数	单向 hash 函数
生日悖论	消息认证	抗强碰撞
压缩函数	消息认证码(MAC)	抗弱碰撞

11.7.2　思考题

11.1　消息认证是为了对付哪些类型的攻击？

11.2 消息认证或数字签名方法有哪两层功能?

11.3 产生消息认证有哪些方法?

11.4 对称加密和错误控制码一起用于消息认证时,这两个函数必须以何种顺序执行?

11.5 什么是消息认证码?

11.6 消息认证码和 hash 函数之间的区别是什么?

11.7 为提供消息认证,应以何种方式保证 hash 值的安全?

11.8 为了攻击 MAC 算法,必须要恢复密钥吗?

11.9 安全 hash 函数需要具有哪些特性?

11.10 抗弱碰撞和抗强碰撞之间的区别是什么?

11.11 hash 函数中压缩函数的作用是什么?

11.7.3 习题

11.1 如果 F 是错误检测函数,那么无论是用做内部函数还是外部函数(见图 11.2),它都具有错误检测能力。如果被传输的消息的任何位被改变,那么不管是在加密函数内或外执行 FCS,接收到的 FCS 和计算出的 FCS 都将会不一致。有些编码还具有错误纠正能力,如果在传输中有一位或少数位被改变,则纠错码应含有足够的冗余信息可确定错误位并纠正之。显然,纠错码用在加密函数外时,它具有纠错能力。如果纠错码用在加密函数内,那么它具有纠错能力吗?

11.2 可以将 11.3 节中给出的数据认证算法定义为使用密文分组链接(CBC)方式的 DES 运算,其初始矢量为 0(见图 11.6)。证明通过密文反馈方式可以得出同样的结果。

11.3 高速传输协议 XTP(Xpress Transfer Protocol)使用 32 位校验和函数,它是两个 16 位函数 XOR 和 RXOR 的连接,XOR 和 RXOR 是 11.4 节图 11.8 中所示的"两个简单的 hash 函数"。

a. 该校验和能否检测出由奇数位错所引起的所有错误?请说明原因。

b. 该校验和能否检测出由偶数位错所引起的所有错误?若不能,请说明该校验和所不能检测出的错误类型的特征。

c. 若将该函数作为消息认证中的 hash 函数,试分析其效率。

11.4 **a.** 考虑 11.4 节中给出的、Davies 和 Price 提出的 hash 码方法,假定使用 DES 作为加密函数:

$$H_i = \mathrm{E}_{M_i}[H_{i-1}] \oplus H_{i-1}$$

前面讲过 DES 的互补性(习题 3.10):若 $Y = \mathrm{DES}_K(X)$,则 $Y' = \mathrm{DES}_{K'}(X')$。利用该性质说明如何修改分组为 $M_1, M_2, \cdots, M_N$ 的消息而不改变其 hash 码。

b. 证明存在类似的攻击可成功地攻击[MEYE88]中提出的方法:

$$H_i = \mathrm{E}_{H_{i-1}}[M_i] \oplus M_i$$

11.5 可以利用 hash 函数构造类似 DES 结构的分组密码。但 hash 是单向的,面分组密码是可逆的(解密),那么如何用 hash 码构造上述分组密码呢?

11.6 考虑相反的问题:利用加密算法构造单向 hash 函数。考虑使用有一个已知密钥的 RSA 算法。如下处理含有若干分组的消息:加密第一分组,将加密结果与第二分组

异或并加密之，等等。说明在解决下面的问题时该方法是不安全的。给定两个分组消息 B1，B2，其 hash 码为：

$$RSAH(B1,B2) = RSA(RSA(B1) \oplus B2)$$

给定任一分组 C1，选择 C2 使得 RSAH(C1，C2) = RSAH(B1，B2)。

附录 11A　生日攻击的数学基础

在本附录中，我们给出生日攻击的数学证明。首先我们介绍一个与 hash 函数相关的问题，然后讨论生日攻击问题，并说明生日攻击这个名称的来历。

相关问题

下面给出通常与 hash 函数有关的一个问题。给定 hash 函数 H 和某 hash 值 $H(x)$，假定 H 有 n 种可能的输出，如果 H 作用于 k 个随机的输入，那么至少有 y 使得 $H(y) = H(x)$ 的概率为 0.5 的 k 的值是多少？

对某 y 值，$H(y) = H(x)$ 的概率恰为 $1/n$。反过来，$H(y) \neq H(x)$ 的概率为 $[1 - (1/n)]$。如果我们产生 k 个随机的 y 值，那么它们均不能与 x 匹配的概率等于每个值不与 x 匹配的概率之积，即 $[1 - (1/n)]^k$，因此至少有一个匹配的概率是 $1 - [1 - (1/n)]^k$。

二项式定理可描述如下：

$$(1-a)^k = 1 - ka + \frac{k(k-1)}{2!}a^2 - \frac{k(k-1)(k-2)}{3!}a^3 \cdots$$

a 很小时上式约为 $(1 - ka)$。因此，至少有一个匹配的概率约为 $1 - [1 - (1/n)]^k \approx 1 - [1 - (k/n)] = k/n$。要使概率为 0.5，则 $k = n/2$。

特别地，若 hash 码为 m 位，则可能有 2^m 个 hash 码，使上述概率为 1/2 的 k 为：

$$k = 2^{(m-1)} \tag{11.1}$$

生日悖论

在初等概率课程中，常用生日悖论来说明一些违背直觉的结果。我们可以如下描述这类问题：使 k 个人中至少有两个人生日相同的概率大于 0.5 的最小 k 值是多少？我们不考虑 2 月 29 日并且假定每个生日出现的概率相同。要解决上述问题，我们定义：

$P(n,k) = Pr[k$ 个元素中至少有一个元素重复出现，其中每个元素出现的概率均为 $1/n]$

所以，我们就是要找使得 $P(365,k) \geqslant 0.5$ 的最小 k。我们用 $Q(365,k)$ 表示没有重复的概率，若 $k > 365$ 则不会出现任何重复，所以我们假定 $k \leqslant 365$。设 k 个元素均不重复的数为 N，那么，第一个元素有 365 种取值，第二个元素有 364 种取值，等等。因此，不同的方式数为：

$$N = 365 \times 364 \times \cdots (365 - k + 1) = \frac{365!}{(365-k)!} \tag{11.2}$$

如果允许重复，那么每一元素有 365 种取法，共有 365^k 种取法。所以不重复的概率为无重复数除以总数：

$$Q(365,k) = \frac{365!/(365-k)!}{(365)^k} = \frac{365!}{(365-k)!(365)^k}$$

且

$$P(365,k) = 1 - Q(365,k) = 1 - \frac{365!}{(365-k)!(365)^k} \tag{11.3}$$

该函数如图 11.11 所示，该概率比一些人想像的要大得多，许多人以为要使至少有一个重复的概率大于 0.5，那么大约应有 100 人。事实上，因为 P(365,23) = 0.507 3，所以人数为 23。k = 100 时至少有一个重复的概率为 0.999 999 7。

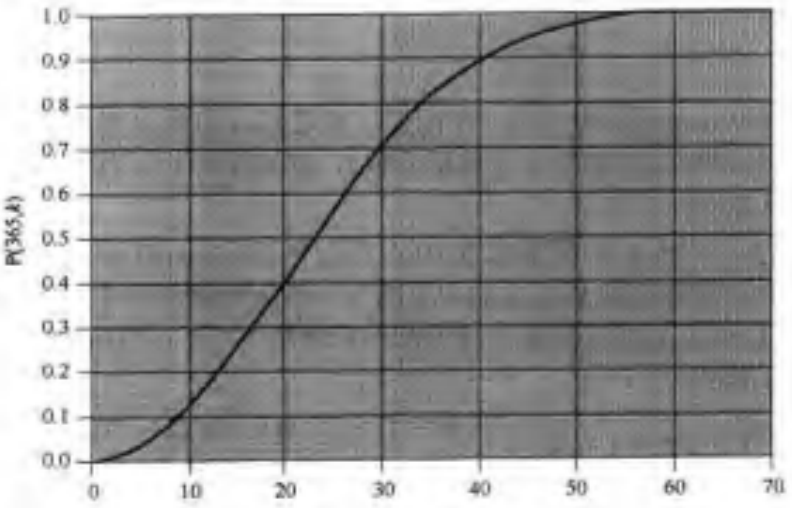


图 11.11 生日悖论

虽然某人与另一人生日相同的概率较小，但是这里我们考虑的是任何两个人生日相同的概率。在 23 个人中，有(23(23 - 1))/2 = 253 种不同的双人组合，因此生日相同的概率较大。

常用不等式

在讨论生日悖论的一般问题之前，我们先推导一个有用的不等式：

$$(1-x) \leqslant e^{-x} \qquad x \geqslant 0 \tag{11.4}$$

图 11.12 给出了该不等式的图像。下面的直线在 $x = 0$ 处与 e^{-x} 相切，其斜率为 e^{-x} 在 $x = 0$ 处的导数：

$$f(x) = e^{-x}$$

$$f'(x) = \frac{d}{dx} e^{-x} = -e^{-x}$$

$$f'(0) = -1$$

这条切线是形为 $ax + b$ 的直线，其中 $a = -1$，它在 $x = 0$ 处的值等于 $e^{-0} = 1$，因此这条切线即是函数$(1 - x)$，这说明不等式(11.4)成立，并且对较小的 x，有$(1 - x) \approx e^{-x}$。

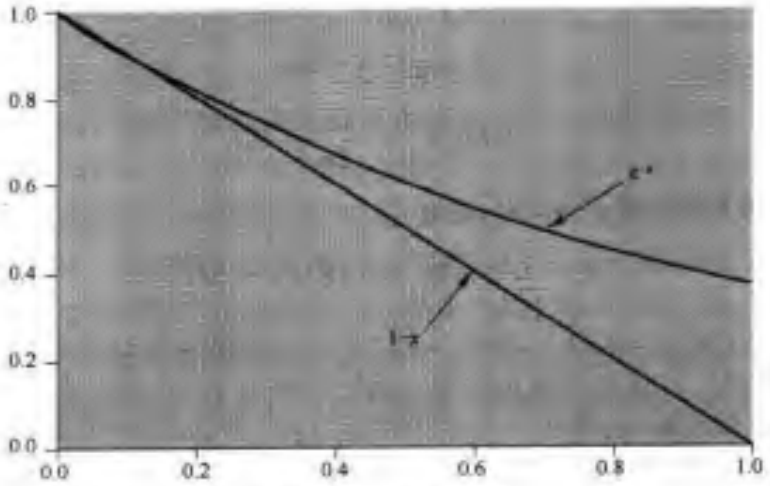


图 11.12　常用不等式

元素重复的一般情形

生日悖论可推广为下述情形：给定一个在 1 到 n 之间均匀分布的随机整数变量以及它的 k 个实例($k \leqslant n$)，那么至少有一个重复的概率 P(n，k)是多少？生日悖论是其在 $n = 365$ 时的特例。和前面的推导过程一样，我们可推广等式(11.3)为：

$$\mathrm{P}(n, k) = 1 - \frac{n!}{(n-k)!n^k} \tag{11.5}$$

即

$$\mathrm{P}(n, k) = 1 - \frac{n \times (n-1) \times \cdots \times (n-k+1)}{n^k}$$

$$= 1 - \left[\frac{n-1}{n} \times \frac{n-2}{n} \times \cdots \times \frac{n-k+1}{n}\right]$$

$$= 1 - \left[\left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times \cdots \times \left(1 - \frac{k-1}{n}\right)\right]$$

根据不等式(11.4)有：

$$\mathrm{P}(n,k) > 1 - [(e^{-1/n}) \times (e^{-2/n}) \times \cdots \times (e^{-(k-1)/n})]$$

$$> 1 - e^{-[(1/n) + (2/n) + \cdots + ((k-1)/n)]}$$

$$> 1 - e^{-(k \times (k-1))/2n}$$

下面我们来看一下 k 为多少时，P(n，k) > 0.5。要使 P(n，k) > 0.5，则：

$$1/2 = 1 - e^{-(k \times (k-1))/2n}$$

$$2 = e^{(k \times (k-1))/2n}$$

$$\ln(2) = \frac{k \times (k-1)}{2n}$$

k 较大时，我们可用 k^2 取代 $k \times (k-1)$，则有：

$$k = \sqrt{2(\ln 2)n} = 1.18\sqrt{n} \approx \sqrt{n} \tag{11.6}$$

$n = 365$ 时，我们有 $k = 1.18 \times \sqrt{365} = 22.54$，它与正确结果 23 非常接近。

下面我们来描述生日攻击的基本原理。假定函数 H 有 2^m 种可能的输出（即输出为 m 位），H 作用于 k 个随机输入，那么 k 为多少时至少有一个重复出现[即对某输入 x 和 y，有 $H(x) = H(y)$]？利用等式(11.6)中的近似公式有：

$$k = \sqrt{2^m} = 2^{m/2} \tag{11.7}$$

两个集合中元素的重复

我们上面讨论的生日问题可以推广为下列问题：给定一个在 1 到 n 之间均匀分布的随机整数变量以及它的两个实例集合，每个集合中有 k 个元素（$k \leqslant n$），那么这两个集合相交（即至少有一个元素同属两个集合）的概率 $R(n, k)$ 是多少？

假设这两个集合 X 和 Y 分别为 $\{x_1, x_2, \cdots, x_k\}$ 和 $\{y_1, y_2, \cdots, y_k\}$，对给定的 x_1，$y_1 = x_1$ 的概率恰为 $1/n$，所以 y_1 不等于 x_1 的概率为 $[1 - (1/n)]$，若在 Y 中随机取 k 个值，则这些值均不等于 x_1 的概率为 $[1 - (1/n)]^k$，因此至少有一个值等于 x_1 的概率为 $1 - [1 - (1/n)]^k$。

如果 n 和 k 均较大（如 k 约为 $\sqrt{n}$），那么 k 个值中只有少数重复，大多数值都不相同。我们假定 X 中的元素均不相同，则有：

$$\Pr[Y\text{中没有元素与}x_1\text{匹配}] = \left(1 - \frac{1}{n}\right)^k$$

$$\Pr[Y\text{中没有元素与}X\text{中的元素匹配}] = \left(\left(1 - \frac{1}{n}\right)^k\right)^k = \left(1 - \frac{1}{n}\right)^{k^2}$$

$$R(n,k) = \Pr[Y\text{中至少有一个元素与}X\text{中的元素匹配}] = 1 - \left(1 - \frac{1}{n}\right)^{k^2}$$

由不等式(11.4)，我们有：

$$R(n, k) > 1 - (e^{-1/n})^{k^2}$$

$$R(n, k) > 1 - (e^{-k^2/n})$$

下面我们来看这样一个问题：k 为多少时，$R(n, k) > 0.5$？要使 $P(n, k) > 0.5$，则：

$$1/2 = 1 - (e^{-(k^2/n)})$$

$$2 = e^{k^2/n}$$

$$\ln(2) = \frac{k^2}{n}$$

$$k = \sqrt{(\ln(2))n} = 0.83\sqrt{n} \approx \sqrt{n} \tag{11.8}$$

我们可以将上述问题用与生日攻击有关的术语描述如下：假定函数 H 有 2^m 种可能的输出(即输出为 m 位)，H 作用于 k 个随机输入得到集合 X，H 作用于另外 k 个随机输入得到集合 Y，那么 k 为多少时，这两个集合中至少有一个匹配[即对某输入 $x \in X$ 和 $y \in Y$，有 $H(x) = H(y)$]？由等式(11.8)中的近似公式有：

$$k = \sqrt{2^m} = 2^{m/2}$$

第 12 章　hash 算法

hash 函数的发展过程和对称分组密码的发展过程类似。第一，穷举攻击能力和密码分析能力的不断增强，减少了 DES 算法的使用，同时，设计具有更长密钥且能抗密码分析攻击的新算法也不像以前那么流行；同样，计算能力和对 hash 函数的密码分析能力的增强，减少了 MD4 和 MD5 算法的使用，所以人们设计出具有更长 hash 码且能抗密码分析的新 hash 算法。第二，hash 函数和对称分组密码都遵循已被证明的结构。DES 算法基于 Feistel 密码，而 Feistel 密码基于 Shannon 提出的替换 - 置换。因为 Feistel 设计的结构能够对付新近发现的密码分析攻击，因此许多重要的分组密码都遵循 Feistel 结构，如果用完全不同的新方法设计对称分组密码，那么不一定能保证安全性；同样，目前最重要的 hash 函数都遵循图 11.10 所示的基本结构，已经证明这种结构是合理的，设计新的 hash 函数只是改进这种结构以及增加 hash 码长。

本章中，我们先讨论三个重要的 hash 函数：MD5、SHA-1 和 RIPEMD-160；然后讨论 Internet 标准的消息认证码 HMAC，其中包含了 hash 函数。

12.1　MD5 消息摘要算法

MD5 消息摘要算法（RFC 1321）是由 MIT 的 Ron Rivest（RSA［Rivest-Shamir-Adleman］公钥密码算法中的"R"）提出的。近几年，随着穷举攻击和密码分析的发展，最为广泛使用的安全 hash 函数 MD5 不再像以前那么流行。

12.1.1　MD5 算法步骤

MD5 算法的输入可以是任意长度的消息，对输入按 512 位的分组为单位进行处理，算法的输出是 128 位的消息摘要。

图 12.1 描述了产生消息摘要的处理过程，它遵循图 11.10 所示的一般结构。该处理过程包含下列步骤：

- **步骤 1：增加填充位。**填充消息使其长度与 448 模 512 同余（即长度≡448 mod 512）。也就是说，填充后的消息长度比 512 的某整数倍少 64 位。即使消息本身已经满足上述长度要求，仍然需要进行填充。例如，若消息长为 448 位，则仍需要填充 512 位使其长度为 960 位，因此填充位数在 1 到 512 之间。填充由一个 1 和后续的 0 组成。
- **步骤 2：填充长度。**用 64 位表示填充前消息的长度，并将其附在步骤 1 所得结果之后（最低有效位在前）。若填充前消息的长度大于 2^{64}，则只使用其低 64 位，即它包含的是填充前消息的长度对 2^{64} 取模的结果。

  上述两步所得消息的长度是 512 的整数倍，图 12.1 中用 512 位的分组 $Y_0, Y_1, \cdots, Y_{L-1}$ 表示填充后的消息，所以填充后的消息总长为 $L \times 512$ 位。消息总长也可通过长为 32 位的字来表示，用 $M[0 \cdots N-1]$ 表示这些字，则有 $N = L \times 16$。

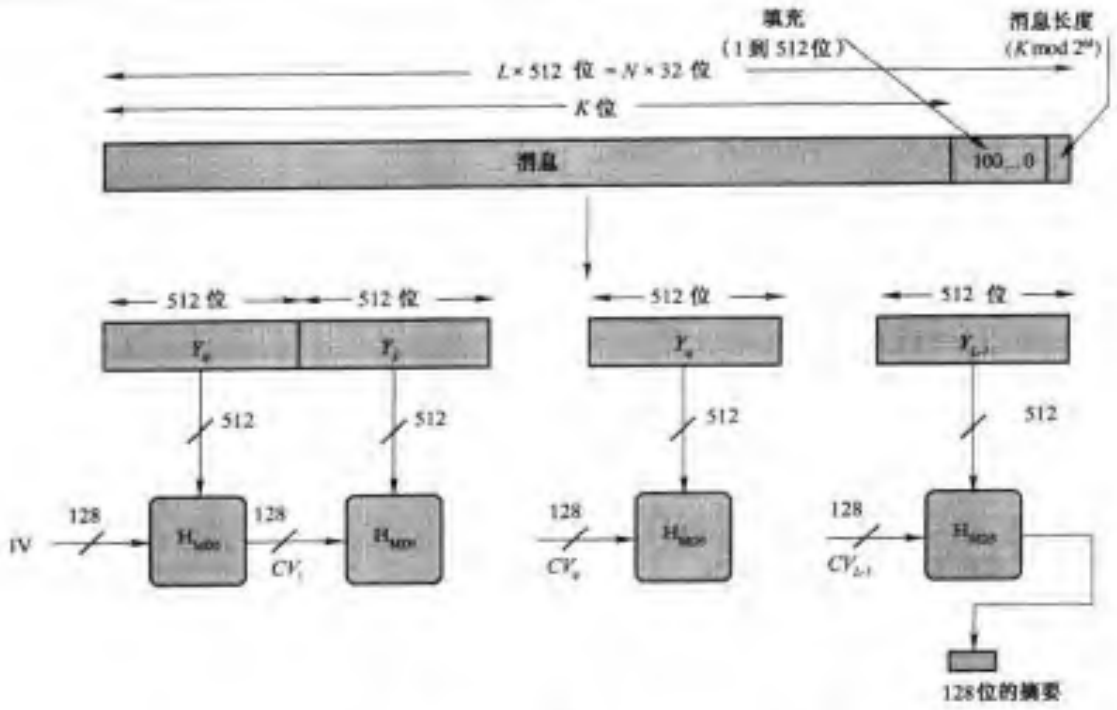


图 12.1　利用 MD5 算法产生消息摘要

- **步骤 3：初始化 MD 缓冲区。**hash 函数的中间结果和最终结果保存于 128 位的缓冲区中，缓冲区用 4 个 32 位的寄存器(A,B,C,D)表示，并将这些寄存器初始化为下列 32 位的整数(十六进制值)：

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476

上述初始值以低端格式存储，也就是说，字的最低有效字节存储在低地址字节位置，即如下存储(十六进制)：

字 A: 01 23 45 67
字 B: 89 AB CD EF
字 C: FE DC BA 98
字 D: 76 54 32 10

- **步骤 4：以 512 位的分组(16 个字)为单位处理消息。**算法的核心是压缩函数，它由四"轮"运算组成，在图 12.1 中压缩函数模块标记为 H_{MD5}，其处理步骤如图 12.2 所示。四轮运算结构相同，但各轮使用不同的基本逻辑函数，我们分别称之为 F,G,H 和 I。
每轮的输入是当前要处理的 512 位的分组(Y_q)和 128 位缓冲区 ABCD 的内容。表 T 有 64 个元素，每轮使用表 T[1…64]中的 16 个元素，并更新缓冲区。表 T 是通过正弦函数来构造的，T 的第 i 个元素，记为 T[i]，其值为 $2^{32} \times \text{abs}(\sin(i))$的整数部分，$i$ 是

弧度。因为abs(sin(i))在0到1之间,所以T的每个元素都可用32位表示。表T"随机化"32位的输入数据,即消除输入数据的规律性。表12.1(b)列出了T的所有值。

第四轮的输出与第一轮的输入(CV_q)相加得到 CV_{q+1},这里的加法是指缓冲区中的4个字与 CV_q 中对应的4个字分别模 2^{32} 相加。

- **步骤5:输出**。所有的 L 个512位的分组处理完后,第 L 个分组的输出即是128位的消息摘要。

我们可将MD5的处理过程归纳如下:

$$CV_0 = \text{IV}$$
$$CV_{q+1} = \text{SUM}_{32}[CV_q, \text{RF}_\text{I}(Y_q, \text{RF}_\text{H}(Y_q, \text{RF}_\text{G}(Y_q, \text{RF}_\text{F}(Y_q, CV_q))))]$$
$$MD = CV_{L-1}$$

其中

IV = 第三步定义的缓冲区ABCD的初值

Y_q = 消息的第 q 个512位分组

L = 消息分组的个数(包括填充位和长度域)

CV_q = 处理消息的第 q 个分组时所使用的链接变量

RF_x = 使用基本逻辑函数x的轮函数

MD = 消息摘要

SUM_{32} = 对输入字分别执行模 2^{32} 加法

表12.1 MD5算法的重要成分

(a)基本逻辑函数的真值表

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

(b)从正弦函数构造的表T

T[1]	=	D76AA478	T[17]	=	F61E2562	T[33]	=	FFFA3942	T[49]	=	F4292244
T[2]	=	E8C7B756	T[18]	=	C040B340	T[34]	=	8771F681	T[50]	=	432AFF97
T[3]	=	242070DB	T[19]	=	265E5A51	T[35]	=	699D6122	T[51]	=	AB9423A7
T[4]	=	C1BDCEEE	T[20]	=	E9B6C7AA	T[36]	=	FDE5380C	T[52]	=	FC93A039
T[5]	=	F57COFAF	T[21]	=	D62F105D	T[37]	=	A4BEEA44	T[53]	=	655B59C3

（续表）

T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBC8	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

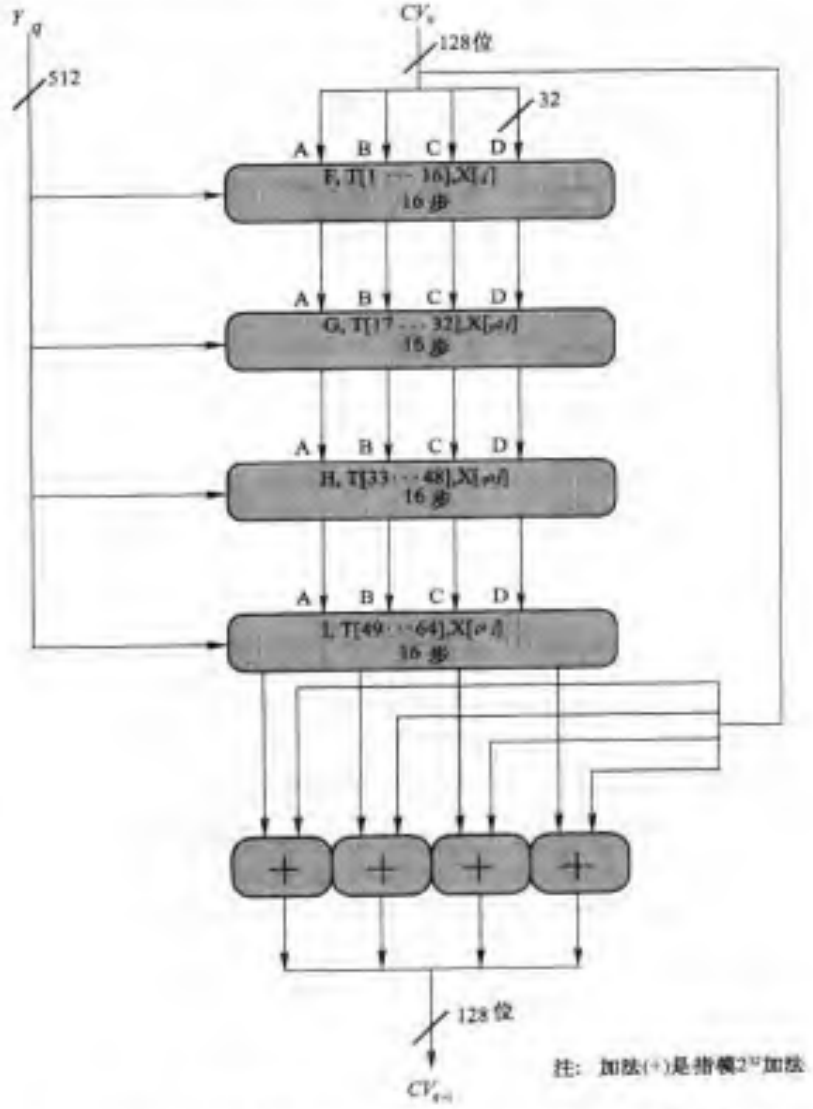


图 12.2　MD5 算法处理 512 位的分组的过程(MD5 压缩函数)

12.1.2 MD5 压缩函数

下面我们详细讨论每一轮处理 512 位的分组的过程。MD5 中每一轮要对缓冲区 ABCD 进行 16 步迭代，每步迭代形为：

$$a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i]) <<< s)$$

其中

a,b,c,d = 缓冲区的四个字，它按一定次序随迭代步变化
g = 基本逻辑函数 F,G,H,I 之一
$<<< s$ = 32 位的变量循环左移 s 位
$X[k]$ = $M[q \times 16 + k]$ = 消息第 q 个 512 位分组的第 k 个 32 位字
$T[i]$ = 矩阵 T 中的第 i 个 32 位字
+ = 模 2^{32} 加法

图 12.3 阐明了上述操作，每步中将(a,b,c,d)循环右移一个字。

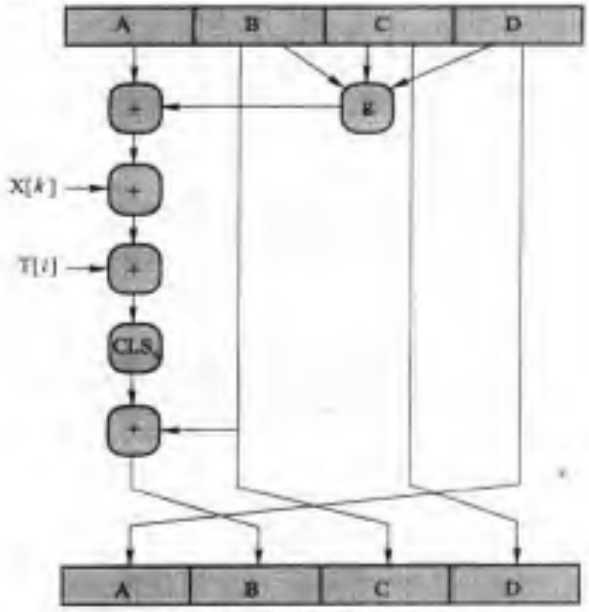


图 12.3 MD5 的基本操作(单步)

每轮使用一个基本逻辑函数，每个基本逻辑函数的输入是三个 32 位的字，输出是一个 32 位的字，它执行位逻辑运算，即输出的第 n 位是其三个输入的第 n 位的函数，这些函数可归纳如下：

轮	基本函数 g	g(b,c,d)
1	F(b,c,d)	$(b \wedge c) \vee (\bar{b} \wedge d)$
2	G(b,c,d)	$(b \wedge d) \vee (c \wedge \bar{d})$
3	H(b,c,d)	$b \oplus c \oplus d$
4	I(b,c,d)	$c \oplus (b \vee \bar{d})$

我们用符号(∧, ∨, ¯, ⊕)表示逻辑操作(AND,OR,NOT,XOR)。函数 F 是条件函数:如果 b 则 c 否则 d;函数 G 也是条件函数:如果 d 则 b 否则 c;函数 H 产生一个奇偶位。这些函数的真值表见表 12.1(a)。

图 12.4(摘自 RFC 1321)给出了步骤 4 的处理过程。当前要处理的 512 位的分组保存于数组 X[0…15]中,其元素是 32 位的字。X[i]在每轮中恰好被使用一次,不同轮中其使用顺序不相同。第一轮中,其使用顺序为初始顺序,第二轮至第四轮中其使用顺序由下列置换确定:

$$\rho_2(i) = (1 + 5i) \bmod 16$$
$$\rho_3(i) = (5 + 3i) \bmod 16$$
$$\rho_4(i) = 7i \bmod 16$$

T 中的每个字在每轮中恰好被使用一次,而且每步迭代只更新缓冲区 ABCD 中的一个字,因此,缓冲区的每个字在每轮中被更新四次。这里,每轮都使用了循环左移,且不同轮中循环左移的次数不相同,这些复杂变换的目的是为了保证很难产生碰撞(即不同的分组产生相同的输出)。

12.1.3　MD4

MD4 和 MD5 都是由 Ron Rivest 提出的,但 MD4 出现在 MD5 之前。MD4 最初作为 RFC 发表于 1990 年 10 月,其修订版和 MD5 同时发表于 1992 年 4 月(RFC 1320)。Rivest 在其论文[RIVE90]中提到 MD5 和 MD4 的设计目标相同,所以我们有必要简要讨论一下 MD4。MD4 的设计目标是:

- **安全性**:对 hash 码的通常要求是,找到两个摘要相同的消息在计算上是不可行的。在 MD4 公布之时,由于摘要长度足够长,所以 MD4 对于穷举攻击是安全的。基于当时的技术和 MD4 的复杂性,Rivest 认为 MD4 对密码分析攻击也是安全的。
- **速度**:算法应有利于快速的软件实现。特别地,算法的快速实现是针对 32 位机的,因此算法基于的是字长为 32 位的基本操作。
- **简单和简洁性**:算法应易于描述且易于编程,不需使用大的程序或置换表。这些特点不仅对程序设计有利,而且从安全的角度来看,也希望算法具有这些性质。
- **倾向于使用低端结构**:某些处理器(如 *Intel* 80xxx 和 Pentium 系列)将字的最低有效字节存于低地址字节位置(低端),而其他的处理器(如 SUN Sparcstation)将字的最高有效位存于低地址字节位置(高端)。当将消息作为 32 位的字的序列进行处理时,这两种处理方式有着明显区别,因为其中一种结构需要将每个要处理的字节反向。Rivest 注意到使用高端结构的处理器一般比较快,因而能够承受这种处理代价,所以他选择使用低端结构将消息表示为 32 位的字的序列。

这些设计目标同样也适用于 MD5。由于 MD5 比 MD4 更复杂,所以其执行速度也更慢,Rivest 认为增加复杂性可以增加安全性。MD4 和 MD5 的主要区别如下:

1. MD4 使用三轮运算,每轮 16 步;MD5 使用四轮运算,每轮 16 步。
2. MD4 的第一轮运算没有使用加法常量,第二轮运算中每步迭代使用的加法常量相同,第三轮运算中每步迭代使用的加法常量也相同,但不同于第二轮中使用的加法常量;

MD5 的 64 步迭代中，每步迭代使用的都是不同的加法常量 T[i]。

```
/*处理16个字(512位)的分组*/
For q = 0 to (N/16) - 1 do
   /*复制分组q到 X */
   For j = 0 to 15 do
         置 X[j] 为 M[q*16 + j]
   end /*j 循环*/

   /*分别存 A,B,C,D到 AA, BB, CC,DD 中*/
   AA = A
   BB = B
   CC = C
   DD = D

   /* 第一轮 */
   /* 令 [abcd  k  s  i]表示
   a = b + ((a + F(b,c,d) + X[k] + T[i]) <<<s)
   执行下面16 步迭代 */
   [ABCD   0   7   1]
   [DABC   1   12  2]
   [CDAB   2   17  3]
   [BCDA   3   22  4]
   [ABCD   4   7   5]
   [DABC   5   12  6]
   [CDAB   6   17  7]
   [BCDA   7   22  8]
   [ABCD   8   7   9]
   [DABC   9   12  10]
   [CDAB   10  17  11]
   [BCDA   11  22  12]
   [ABCD   12  7   13]
   [DABC   13  12  14]
   [CDAB   14  17  15]
   [BCDA   15  22  16]

   /* 第二轮 */
   /* 令 [abcd  k  s  i]表示
   a = b + ((a + G(b,c,d) + X[k] + T[i]) <<<s)
   执行下面16 步迭代 */
   [ABCD   1   5   17]
   [DABC   6   9   18]
   [CDAB   11  14  19]
   [BCDA   0   20  20]
   [ABCD   5   5   21]
   [DABC   10  9   22]
   [CDAB   15  14  23]
   [BCDA   4   20  24]
   [ABCD   9   5   25]
   [DABC   14  9   26]
   [CDAB   3   14  27]
   [BCDA   8   20  28]
   [ABCD   13  5   29]
   [DABC   2   9   30]
   [CDAB   7   14  31]
   [BCDA   12  20  32]

   /* 第三轮 */
   /* 令 [abcd  k  s  i]表示
   a = b + ((a + H(b,c,d) + X[k] + T[i]) <<<s)
   执行下面16 步迭代 */
   [ABCD   5   4   33]
   [DABC   8   11  34]
   [CDAB   11  16  35]
   [BCDA   14  23  36]
   [ABCD   1   4   37]
   [DABC   4   11  38]
   [CDAB   7   16  39]
   [BCDA   10  23  40]
   [ABCD   13  4   41]
   [DABC   0   11  42]
   [CDAB   3   16  43]
   [BCDA   6   23  44]
   [ABCD   9   4   45]
   [DABC   12  11  46]
   [CDAB   15  16  47]
   [BCDA   2   23  48]

   /* 第四轮 */
   /* 令 [abcd  k  s  i]表示
   a = b + ((a + I (b,c,d) + X[k] + T[i]) <<<s)
   执行下面16 步迭代 */
   [ABCD   0   6   49]
   [DABC   7   10  50]
   [CDAB   14  15  51]
   [BCDA   5   21  52]
   [ABCD   12  6   53]
   [DABC   3   10  54]
   [CDAB   10  15  55]
   [BCDA   1   21  56]
   [ABCD   8   6   57]
   [DABC   15  10  58]
   [CDAB   6   15  59]
   [BCDA   13  21  60]
   [ABCD   4   6   61]
   [DABC   11  10  62]
   [CDAB   2   15  63]
   [BCDA   9   21  64]

   /*  将四个寄存器的值分别加上其初始值 */

   A = A + AA
   B = B + BB
   C = C + CC
   D = D + DD

end /* q循环 */
```

图 12.4　基本的 MD5 更新算法(RFC 1321)

3. MD5 使用四个基本逻辑函数，每轮运算使用一个基本逻辑函数；MD4 中使用三个基本函数，每轮运算使用一个基本逻辑函数。
4. MD5 中每步迭代的结果都与前一步的结果相加。例如，步骤 1 更新字 A，步骤 2 将循环左移的结果与 A 相加，所得结果存于 D 中，同样地，步骤 3 将循环左移的结果与 D 相加，所得结果存于 C 中；MD4 中没有执行这次加法。Rivest 认为包含前面步骤的结果会产生更大的雪崩效应。

12.1.4　MD5 的强度

MD5 算法有这样一个特点，即 hash 函数的每一位都是输入的每一位的函数。基本逻辑函数(F,G,H,I)的复杂迭代使得输出对输入的依赖非常小；也就是说，随机选择的两条消息，即使它们具有相同的规律性，也不可能产生相同的 hash 码。Rivest 猜测，MD5 可能是 128 位的 hash 码中最强的算法，即找到 hash 码相同的两条消息所需的代价为 2^{64} 数量级，找到具有给定摘要的消息所需的代价为 2^{128} 数量级。

到目前为止，还没有分析说明 Rivest 的猜测不成立。但是现有的一些攻击对 MD5 算法非常不利：

1. Berson[BERS92]已经证明，对单轮的 MD5 算法，利用差分密码分析，可以在合理的时间内找出摘要相同的两条消息，这一结果对 MD5 的四轮运算中的每一轮都成立。但是 Berson 尚不能说明如何将这种攻击推广到具有四轮运算的 MD5 之上。
2. Boer 和 Bosselaers[BOER93]说明了如何找到消息分组 X 和两个有关的链接变量使得它们产生相同的输出，也就是说，对一个 512 位的分组，MD5 压缩函数对缓冲区 ABCD 的不同值产生相同的输出，我们称之为伪碰撞。目前尚无法用上述方法成功地攻击 MD5 算法。
3. Dobbertin[DOBB96a]提出的攻击对 MD5 最具威胁，它可使 MD5 压缩函数产生碰撞，也就是说，给定一个 512 位的分组，这种方法可以找到另一个 512 位的分组，使得它们的 MD5 运算结果相同。到目前为止，尚不能用 Dobbertin 提出的方法对使用初值(IV)的整个消息进行攻击。

由此可见，MD5 算法抗密码分析的能力较弱，对 MD5 的生日攻击所需的代价为 2^{64} 数量级，因此，应该使用 hash 码更长且抗密码分析能力更强的 hash 函数替代 MD5，其中两种常用的这种函数是我们在下两节中将讨论的 SHA-1 和 RIPEMD-160。

12.2　安全 hash 算法

安全 hash 算法(SHA)由美国标准与技术研究所(NIST)设计并于 1993 年作为联邦信息处理标准(FIPS 180) 发布，修订版于 1995 年发布(FIPS 180-1)，通常称之为 SHA-1。该标准称为安全 hash 函数。SHA 算法建立在 MD4 算法之上，其基本框架与 MD4 类似。RFC 3174 也给出了 SHA-1，它基本上是复制 FIPS 180-1 中的内容，但增加了 C 代码实现。

12.2.1　SHA-1 算法步骤

SHA-1 算法的输入是长度小于 2^{64} 位的消息，输出是 160 位的消息摘要，输入消息以 512 位的分组为单位进行处理。

与图 12.1 中 MD5 处理消息的过程一样，SHA-1 算法也将消息按 512 位分组，但 hash 值和链接变量长为 160 位。SHA-1 算法包含下列步骤：

- **步骤 1：增加填充位。**填充消息使其长度与 448 模 512 同余(即长度≡448 mod 512)，即使消息已经满足上述长度要求，仍然需要进行填充，因此填充位数在 1 到 512 之间。填充由一个 1 和后续的 0 组成。
- **步骤 2：填充长度。**在消息后附加 64 位，将其看做 64 位的无符号整数(最高有效字节在前)，它包含填充前消息的长度。
- **步骤 3：初始化 MD 缓冲区。**hash 函数的中间结果和最终结果保存于 160 位的缓冲区中，缓冲区用 5 个 32 位的寄存器(A，B，C，D，E)表示，并将这些寄存器初始化为下列 32 位的整数(十六进制值)：

  A = 67452301
  B = EFCDAB89
  C = 98BADCFE
  D = 10325476
  E = C3D2E1F0

  其中，前四个值与 MD5 中使用的值相同，但在 SHA-1 中这些值以高端格式存储，也就是说，字的最高有效字节存于低地址字节位置，即按如下方式存储上述初始值(十六进制)：

  字 A: 67 45 23 01
  字 B: EF CD AB 89
  字 C: 98 BA DC FE
  字 D: 10 32 54 76
  字 E: C3 D2 E1 F0

- **步骤 4：以 512 位的分组(16 个字)为单位处理消息。**算法的核心是具有四轮运算的模块，每轮执行 20 步迭代，其处理过程如图 12.5 所示。四轮运算结构相同，但各轮使用不同的基本逻辑函数，分别称为 f_1，f_2，f_3 和 f_4。

  每轮的输入是当前要处理的 512 位的分组(Y_q)和 160 位缓冲区 ABCDE 的内容，且每轮都对缓冲区进行更新。每轮使用一个加法常量 K_t，其中 t 表示步数，$0 \leqslant t \leqslant 79$，但实际上只使用四个不同的加法常量，下面是这些常量的十六进制和十进制表示：

步　骤	十六进制	取整数部分
$0 \leqslant t \leqslant 19$	K_t = 5A827999	$[2^{30} \times \sqrt{2}]$
$20 \leqslant t \leqslant 39$	K_t = 6ED9EBA1	$[2^{30} \times \sqrt{3}]$
$40 \leqslant t \leqslant 59$	K_t = 8F1BBCDC	$[2^{30} \times \sqrt{5}]$
$60 \leqslant t \leqslant 79$	K_t = CA62C1D6	$[2^{30} \times \sqrt{10}]$

第四轮第 18 步的输出与第一轮的输入(CV_q)相加得到 CV_{q+1}，这里的加法是指缓冲区中的 5 个字与 CV_q 中对应的 5 个字分别模 2^{32} 相加。

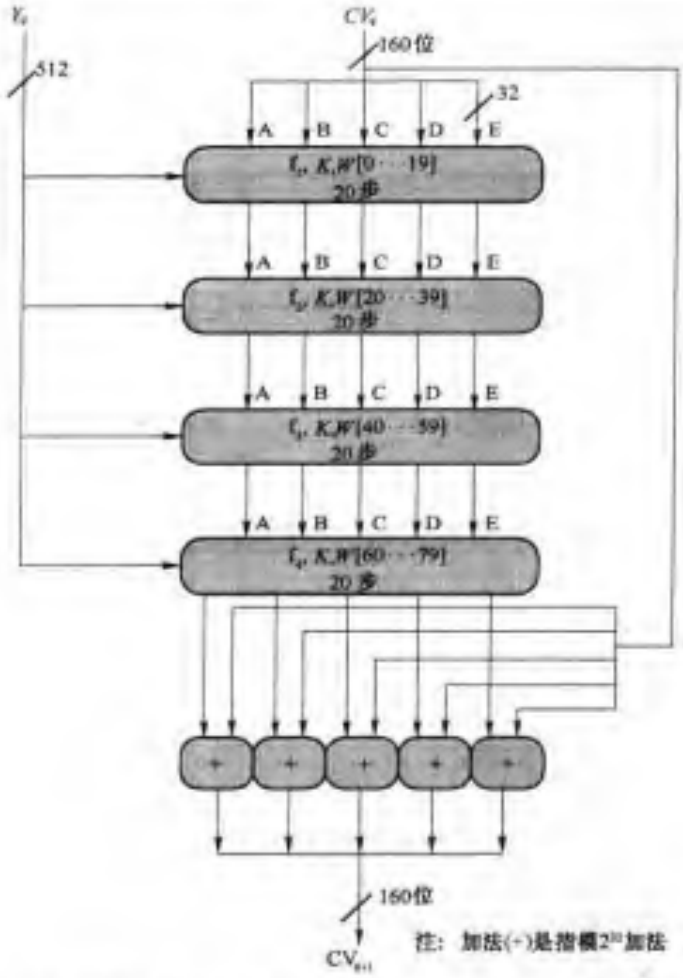


图 12.5　SHA-1 算法处理 512 位的分组的过程(SHA-1 压缩函数)

- **步骤 5:输出**。所有的 L 个 512 位的分组处理完后,第 L 个分组的输出即是 160 位的消息摘要。

我们可将 SHA-1 的处理过程归纳如下:

$$CV_0 = \text{IV}$$
$$CV_{q+1} = \text{SUM}_{32}(CV_q, \text{ABCDE}_q)$$
$$MD = \text{CV}_L$$

其中:

IV	=	第三步定义的缓冲区 ABCDE 的初值
ABCDE_q	=	处理第 q 个消息分组时最后一轮的输出
L	=	消息中分组的个数(包括填充位和长度域)
SUM_{32}	=	对输入字分别执行模 2^{32} 加法
MD	=	消息摘要

12.2.2 SHA-1 压缩函数

处理一个 512 位的分组要执行 80 步，下面我们详细讨论每步的处理过程。每步具有下述形式(见图 12.6)：

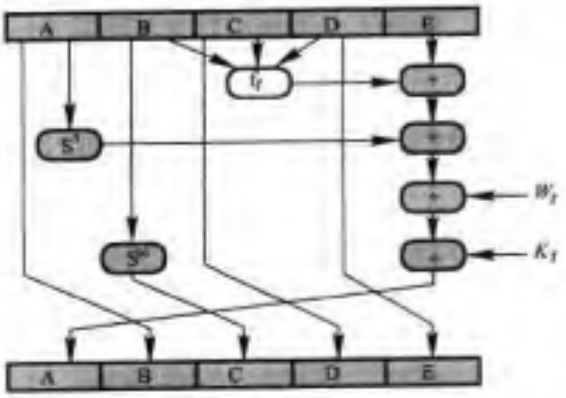


图 12.6 SHA 的基本操作(单步)

$$A,B,C,D,E \leftarrow (E + f(t,B,C,D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D$$

其中：

A,B,C,D,E = 缓冲区的 5 个字
t = 步骤编号，$0 \leqslant t \leqslant 79$
$f(t,B,C,D)$ = 第 t 步使用的基本逻辑函数
S^k = 32 位的变量循环左移 k 位
W_t = 从当前 512 位输入分组导出的 32 位字
K_t = 加法常量。如前所述，共使用了 4 个不同的加法常量
$+$ = 模 2^{32} 加法

每个基本逻辑函数的输入是三个 32 位字，输出是一个 32 位字。每个函数执行位逻辑运算，即输出的第 n 位是三个输入的第 n 位的函数。这些函数可归纳如下：

步 骤	函数名称	函 数 值
$0 \leqslant t \leqslant 19$	$f_1 = f(t,B,C,D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$20 \leqslant t \leqslant 39$	$f_2 = f(t,B,C,D)$	$B \oplus C \oplus D$
$40 \leqslant t \leqslant 59$	$f_3 = f(t,B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \leqslant t \leqslant 79$	$f_4 = f(t,B,C,D)$	$B \oplus C \oplus D$

我们用符号($\wedge$, $\vee$, $-$, $\oplus$)表示逻辑操作(AND, OR, NOT, XOR)。从表中可知，实际上只使用了三个不同的函数。$0 \leqslant t \leqslant 19$ 时，对应的函数是条件函数：若 B 则 C 否则 D；$20 \leqslant t \leqslant 39$ 和 $60 \leqslant t \leqslant 79$ 时，对应的函数产生一个校验位；$40 \leqslant t \leqslant 59$ 时，若至少有两个变量为真，则对应的函数为真。这些函数的真值表见表 12.2。

表 12.2　SHA-1 中基本逻辑函数的真值表

B	C	D	$f_{0..19}$	$f_{20..39}$	$f_{40..59}$	$f_{60..79}$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

图 12.7 说明如何从 512 位的消息分组导出 32 位的字 W_t。W_t 的前 16 个值即是当前分组的 16 个字，其他值的定义如下：

$$W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$

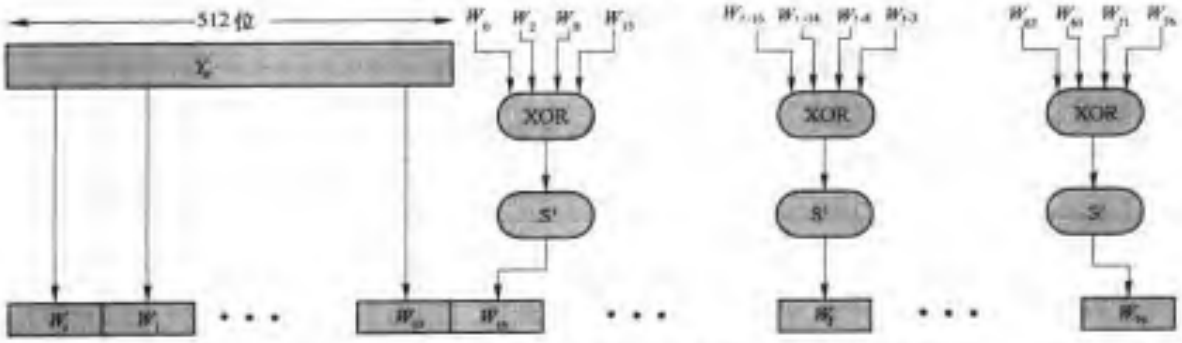


图 12.7　为 SHA-1 处理一个分组而产生的 80 字输入序列

因此，前 16 步迭代中 W_t 的值等于消息分组的第 t 个字，余下的 64 步迭代中 W_t 等于前面四个 W_t 值异或后循环左移一位的结果，这是与 MD5 和 RIPEMD-160 的重要区别。MD5 和 RIPEMD-160 虽然在不同轮中字的使用顺序不同，但每步迭代的输入，只使用消息分组中的一个字。SHA-1 将消息分组的 16 个字扩展为 80 个字供压缩函数使用，这种大量冗余使被压缩的消息分组相互独立，所以，对给定的消息，找出具有相同压缩结果的消息会非常复杂。

12.2.3　SHA-1 和 MD5 的比较

由于 SHA-1 和 MD5 都基于 MD4，所以它们的性质非常相似。下面我们从前面讲到的 MD4 的设计目标方面来比较这两种算法。

- **抗穷举攻击的能力**：SHA-1 和 MD5 最重要的区别之一是，SHA-1 的消息摘要比 MD5 的消息摘要长 32 位。对 MD5 用穷举攻击方法产生具有给定摘要的消息，其代价为 2^{128} 数量级，而对 SHA-1，代价为 2^{160} 数量级；对 MD5 用穷举攻击方法产生两个摘要相同的消息，

其代价为 2^{64} 数量级，而对 SHA-1，代价为 2^{80} 数量级，所以 SHA-1 抗穷举攻击的能力要强得多。

- **抗密码分析的能力**：由上节的讨论可知，MD5 抗已知密码分析攻击的能力较弱，而SHA-1 抗这些攻击的能力似乎并不弱，但是，由于 SHA-1 的设计原则尚未公开，所以很难评价其强度。
- **速度**：由于这两个算法都基于模 2^{32} 加法，所以在 32 位机上这些算法的执行速度较快，SHA-1 要执行 80 步迭代且缓冲区为 160 位，而 MD5 要执行 64 步迭代且缓冲区为 128 位，因此在同样的环境下，SHA-1 的执行速度要比 MD5 的速度慢得多。
- **简单和简洁性**：两个算法都易于描述和实现，不需要使用大的程序或置换表。
- **低位结构和高位结构**：MD5 使用低位在前的方式将消息映射为 32 位字的序列，而 SHA-1 使用高位在前的方式将消息映射为 32 位字的序列[①]，这两种结构没有本质的差异。

12.2.4 安全 hash 函数的修改

1981 年，NIST 修改了 FIPS 180-1，并发布了其推荐的修订版，即 FIPS 180-2，其中除 SHA-1 外新增了三个 hash 算法，它们是 SHA-256，SHA-384 和 SHA-512，其消息摘要的长度分别为 256，384 和 512。NIST 指出 FIPS 180-2 的目的是要与由于使用 AES 而增加的安全性相适应。

表 12.3 列出了上述四种 hash 算法的基本性质。它们之间的最大区别是消息摘要的长度。它决定了算法抗穷举攻击的能力。这四种算法的结构和算法的细节描述非常相似，所以新增的三种算法应和 SHA-1 具有相同的抗密码分析攻击的能力。

表 12.3 SHA 性质比较

	SHA-1	SHA-256	SHA-384	SHA-512
消息摘要大小	160	256	384	512
消息大小	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
分组大小	512	512	1024	1024
字长	32	32	64	64
步数	80	80	80	80
安全性	80	128	192	256

注：1. 大小均指位数。

2. 安全性是指：对大小为 n 的消息摘要的生日攻击，产生一个碰撞所需的代价约为 $2^{n/2}$。

表 12.4 总结了上述三种新增算法的主要组成成分。函数 $\mathrm{ROTL}^n(X)$与 FIPS 180-1 中的函数 $S^n(X)$相同，字母(a,b,c,d,e,f,g)与 FIPS 180-1 中的字母(A,B,C,D,E)的作用相同。也就是说，这些字母均是单字长的缓冲区。

① 这也许是因为 SHA-1 的设计者 NSA 当初采用 Sun 计算机来实现的。

表 12.4 推荐的 SHA 标准

	SHA-256	SHA-384	SHA-512
函数	$Ch(x,y,z) = (x \wedge y) \oplus (\bar{x} \wedge z)$ $Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (x \wedge z)$ $\sum_0^{256}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$ $\sum_1^{256}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$ $\sigma_0^{256}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$ $\sigma_1^{256}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$	$Ch(x,y,z) = (x \wedge y) \oplus (\bar{x} \wedge z)$ $Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (x \wedge z)$ $\sum_0^{512}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$ $\sum_1^{512}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$ $\sigma_0^{512}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$ $\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$	$Ch(x,y,z) = (x \wedge y) \oplus (\bar{x} \wedge z)$ $Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (x \wedge z)$ $\sum_0^{512}(x) = ROTR^{34}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$ $\sum_1^{512}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$ $\sigma_0^{512}(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$ $\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$
常量	最开始的 64 个素数的立方根的小数部分的前 32 位	最开始的 80 个素数的立方根的小数部分的前 64 位	最开始的 80 个素数的立方根的小数部分的前 64 位
填充	添加一个 1 和若干个 0 使得填充后与 448 模 512 同余	添加一个 1 和若干个 0 使得填充后与 896 模 1024 同余	添加一个 1 和若干个 0 使得填充后与 896 模 1024 同余
长度	添加原始消息长度值(64 位)	添加原始消息长度值(128 位)	添加原始消息长度值(128 位)
初始化缓冲区	6A09E667 BB67AE85 3C6EF372 A54FF53A 510E527F 9B05688C 1F83D9AB 5BE0CD19	CBBB9D5DC1059ED8 629A292A367CD507 9159015A3070DD17 152FECD8F70E5939 67332667FFC0CB31 8EVB44A8768581511 DB0C2E0D64F98FA7 47B5481DBEFA4FA4	6A09E667F3BCC908 BB67AE8584CAA73B 3C6EF372FE94F82B A54FF53A5F1D36F1 510E527FADE682D1 9B05688C2B3E6C1F 1F83D9ABFB41BD6B 5BE0CDI9913E2179
压缩函数	$T_1 = h + \sum_1^{256}(e) + Ch(e,f,g) + K_t^{256} + W_t$ $T_1 = \sum_0^{256}(e) + Maj(a,b,c)$ $(a,b,c,d,e,f,g) =$ $(T_1 + T_2, a,b,c,d + T_1, e,f,g)$	$T_1 = h + \sum_1^{512}(e) + Ch(e,f,g) + K_t^{512} + W_t$ $T_1 = \sum_1^{512}(e) + Maj(a,b,c)$ $(a,b,c,d,e,f,g) =$ $(T_1 + T_2, a,b,c,d + T_1, e,f,g)$	$T_1 = h + \sum_1^{512}(e) + Ch(e,f,g) + K_t^{512} + W_t$ $T_1 = \sum_1^{512}(e) + Maj(a,b,c)$ $(a,b,c,d,e,f,g) =$ $(T_1 + T_2, a,b,c,d + T_1, e,f,g)$

12.3 RIPEMD-160

RIPEMD-160 消息摘要算法[DOBB96b, BOSS97]是欧洲 RIPE(RACE Integrity Primitives Evaluation) 计划下由一组研究人员设计的,这些研究人员曾对 MD4 和 MD5 进行了一些成功的攻击。这个小组最初设计的是 128 位的算法。RIPE 计划完成后,H. Dobbertin(他并未参加 RIPE 计划)发现了对两轮 RIPEMD 的攻击,后来又发现了对 MD4 和 MD5 的攻击。因此,RIPE 的研究人员决定修改 RIPEMD,并和 Dobbertin 一起完成了该算法的修改工作。

12.3.1 RIPEMD-160 算法步骤

RIPEMD-160 算法的输入可以是任意长的消息,输出是 160 位的消息摘要。输入以 512 位的分组为单位进行处理。

与图 12.1 中 MD5 处理消息的过程一样,RIPEMD-160 算法也将消息按 512 位分组,但 hash 值和链接变量长为 160 位。RIPEMD-160 算法包含下列步骤:

- **步骤 1:增加填充位。**填充消息使其长度与 448 模 512 同余(即长度≡448 mod 512),即使消息已经满足上述长度要求,仍然需要进行填充,因此填充位数在 1 到 512 之间。填充由一个 1 和后续的 0 组成。
- **步骤 2:填充长度。**在消息后附加一个 64 位,将其看做 64 位的无符号整数(最低有效字节在前),它包含原始消息分组(填充前)的长度模 2^{64} 的结果。和 MD5 一样,RIPEMD-160 使用的是低端结构,注意这一点与 SHA-1 不同。
- **步骤 3:初始化 MD 缓冲区。**hash 函数的中间结果和最终结果保存于 160 位的缓冲区中,缓冲区用 5 个 32 位的寄存器(A,B,C,D,E)表示,并将这些寄存器初始为下列 32 位的整数(十六进制值):

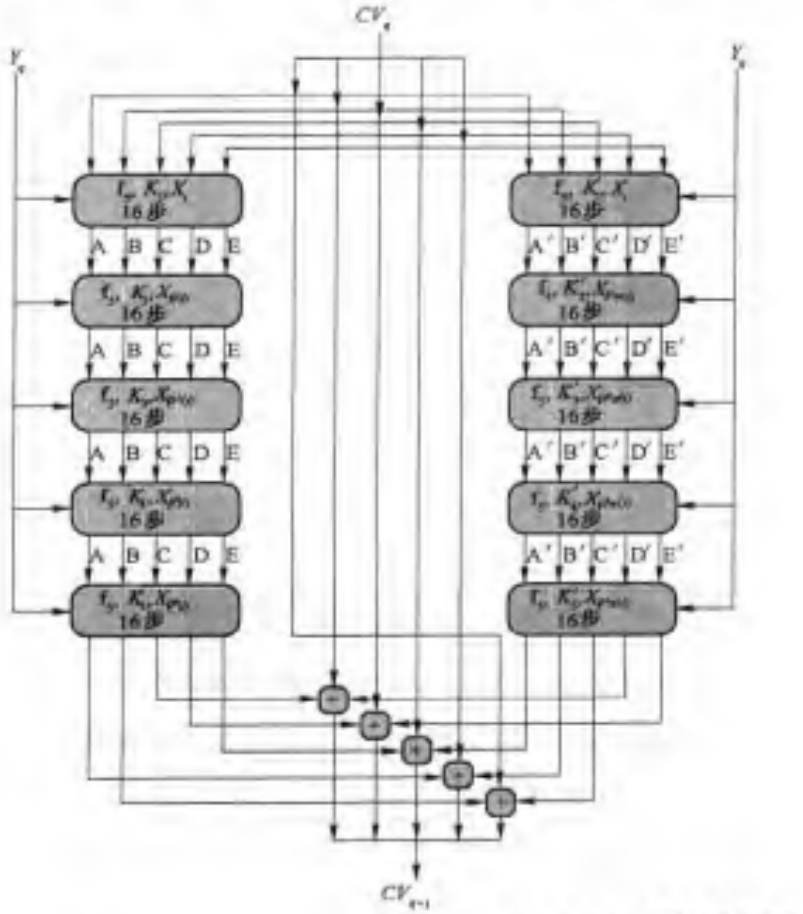
  A = 67452301
  B = EFCDAB89
  C = 98BADCFE
  D = 10325476
  E = C3D2E1F0

  这些初始值与 SHA-1 中使用的值相同,前四个值与 MD5 中使用的值相同。这些值的存放格式与 MD5 中一样,是按低位在前格式存放的。
- **步骤 4:以 512 位的分组(16 个字)为单位处理消息。**算法的核心是具有十轮运算的模块,将这十轮运算分成两组,每组五轮,每轮执行 16 步迭代,其处理过程如图 12.8 所示。每轮运算结构相同,但使用不同的基本逻辑函数,分别称为 f_1, f_2, f_3, f_4 和 f_5。注意图中右列中函数的使用顺序与左列正好相反。

  每轮的输入是当前要处理的 512 位的分组(Y_q)和 160 位缓冲区ABCDE的内容(左边)或者A′B′C′D′E′的内容(右边),且每轮都对缓冲区进行更新。每轮使用一个加法常量,但实际上只使用了九个不同的加法常量,且其中之一为 0,它们的十六进制和十进制表示如表 12.5 所示。

表 12.5　RIPEMD-160 所使用的常量

	左　列		右　列	
步骤编号	十六进制	取整数部分	十六进制	取整数部分
$0 \leqslant j \leqslant 15$	$K_1 = K(j) = 00000000$	0	$K'_1 = K'(j) = 50A28BE6$	$2^{30} \times \sqrt[3]{2}$
$16 \leqslant j \leqslant 31$	$K_2 = K(j) = 5A827999$	$2^{30} \times \sqrt{2}$	$K'_2 = K'(j) = 5C4DD124$	$2^{30} \times \sqrt[3]{3}$
$32 \leqslant j \leqslant 47$	$K_3 = K(j) = 6ED9EBA1$	$2^{30} \times \sqrt{3}$	$K'_3 = K'(j) = 6D703EF3$	$2^{30} \times \sqrt[3]{5}$
$48 \leqslant j \leqslant 63$	$K_4 = K(j) = 8F1BBCDC$	$2^{30} \times \sqrt{5}$	$K'_4 = K'(j) = 7A6D76E9$	$2^{30} \times \sqrt[3]{7}$
$64 \leqslant j \leqslant 79$	$K_5 = K(j) = A953FD4E$	$2^{30} \times \sqrt{7}$	$K'_5 = K'(j) = 00000000$	0



注：加法（+）是指模2^{32}加法

图 12.8　RIPEMD-160 算法处理 512 位的分组的过程（RIPEMD-160 压缩函数）

第五轮（即第 80 步）的输出与第一轮的输入（CV_q）相加得到 CV_{q+1}，这里的加法是指每一列缓冲区中的 5 个字与 CV_q 中对应的 5 个字分别模 2^{32} 相加，并且：

$$CV_{q+1}(0) = CV_q(1) + C + D'$$
$$CV_{q+1}(1) = CV_q(2) + D + E'$$
$$CV_{q+1}(2) = CV_q(3) + E + A'$$
$$CV_{q+1}(3) = CV_q(4) + A + B'$$
$$CV_{q+1}(4) = CV_q(0) + B + C'$$

- **步骤 5:输出。**所有的 L 个 512 位的分组处理完后,第 L 个分组的输出即是 160 位的消息摘要。

12.3.2 RIPEMD-160 压缩函数

处理一个 512 位的分组需要 10 轮,下面我们详细讨论每轮的处理过程。每轮有 16 步,图 12.9说明了每步的运算过程,5 个字(A,B,C,D,E)用来实现整字的循环右移,这种结构和 MD5 的结构完全相同。

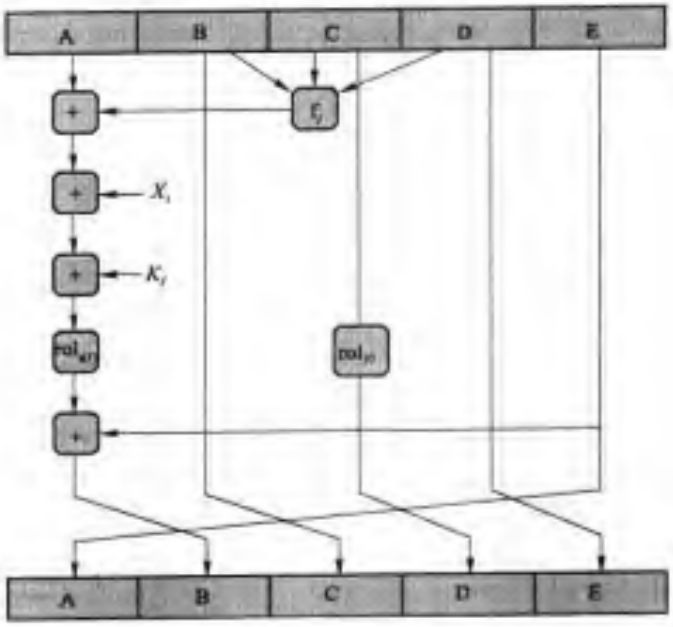


图 12.9 RIPEMD-160 的基本操作(单步)

每轮使用一个基本逻辑函数,且右列中函数的使用顺序与左列中相反(见图 12.8),每个基本逻辑函数的输入是三个 32 位的字,输出是一个 32 位的字,每个函数执行的是位逻辑操作,即输出的第 n 位是三个输入的第 n 位的函数,这些函数可归纳如下:

步骤编号	函数名称	函数值
$0 \leqslant j \leqslant 15$	$f_1 = f(j,B,C,D)$	$B \oplus C \oplus D$
$16 \leqslant j \leqslant 31$	$f_2 = f(j,B,C,D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$32 \leqslant j \leqslant 47$	$f_3 = f(j,B,C,D)$	$(B \vee \overline{C}) \oplus D$
$48 \leqslant j \leqslant 63$	$f_4 = f(j,B,C,D)$	$(B \wedge D) \vee (C \wedge \overline{D})$
$64 \leqslant j \leqslant 79$	$f_5 = f(j,B,C,D)$	$B \oplus (C \vee \overline{D})$

我们用符号($\wedge$, $\vee$, $-$, $\oplus$)表示逻辑操作(AND,OR,NOT,XOR)。函数 f_1 产生奇偶校验位。函数 f_2 是条件函数:如果 B 则 C 否则 D。同样,f_4 也可描述为:如果 D 则 B 否则 C。这五个函数的真值表见表 12.6。

表 12.6　RIPEMD-160 逻辑函数的真值表

B	C	D	f_1	f_2	f_3	f_4	f_5
0	0	0	0	0	1	0	1
0	0	1	1	1	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	1	1	0	1
1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	1
1	1	0	0	1	1	1	0
1	1	1	1	1	0	1	0

[DOBB96b]中的下述伪码说明了每轮的处理过程：

$A := CV_q(0); B := CV_q(1); C := CV_q(2); D := CV_q(3); E := CV_q(4);$
$A' := CV_q(0); B' := CV_q(1); C' := CV_q(2); D' := CV_q(3); E' := CV_q(4);$
for $j := 0$ to 79 **do**
　　$T := rol_{s(j)}(A + f(j, B, C, D) + X_{r(j)} + K(j)) + E;$
　　$A := E; E := D; D := rol_{10}(C); C := B; B := T;$
　　$T := rol_{s'(j)}(A' + f(79 - j, B', C', D') + X_{r'(j)} + K'(j)) + E';$
　　$A' := E'; E' := D'; D' := rol_{10}(C'); C' := B'; B' := T';$
enddo
$CV_{q+1}(0) = CV_q(1) + C + D'; CV_{q+1}(1) = CV_q(2) + D + E'; CV_{q+1}(2) = CV_q(3) + E + A';$
$CV_{q+1}(3) = CV_q(4) + A + B'; CV_{q+1}(4) = CV_q(0) + B + C';$

其中

A, B, C, D, E	= 左列缓冲区的五个字
A′, B′, C′, D′, E′	= 右列缓冲区的五个字
j	= 步骤编号；$0 \leqslant j \leqslant 79$
$f(j, B, C, D)$	= 左列第 j 步和右列第 $79 - j$ 步使用的基本逻辑函数
$rol_{s(j)}$	= 32 位的变量循环左移；循环移动次数由函数 $s(j)$ 决定
$X_{r(j)}$	= 当前 512 位的分组中的 32 位的字；字的选择由置换函数 $r(j)$ 决定
$K(j)$	= 第 j 步使用的加法常量
+	= 模 2^{32} 加法

当前要处理的 512 位的分组保存于 32 位的字数组 X[0…15]中。每轮中每个 X[i]恰好被使用一次，不同轮中这些字的使用顺序不同，表 12.7(a)给出了每列中各轮运算所使用的置换。置换 π 可描述为 $\pi(i) = 9i + 5 \pmod{16}$。表 12.7(b)定义了每轮中使用的循环左移，它指的是处理 32 位的字时每步中使用的移动次数。

表 12.7 RIPEMD-160 算法的重要成分

(a) 字的置换

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\pi(i)$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12

列	第 1 轮	第 2 轮	第 3 轮	第 4 轮	第 5 轮
左	恒等	ρ	ρ^2	ρ^3	ρ^4
右	π	$\rho\pi$	$\rho^2\pi$	$\rho^3\pi$	$\rho^4\pi$

(b) 字的循环左移(两列)

轮	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
2	12	13	11	15	6	9	9	7	12	15	11	13	7	8	7	7
3	13	15	14	11	7	7	6	8	13	14	13	12	5	5	6	9
4	14	11	12	14	8	6	5	5	14	12	15	14	9	9	8	6
5	15	12	13	13	9	5	8	6	15	11	12	11	8	6	5	5

12.3.3 RIPEMD-160 的设计思想

了解 RIPEMD-160 设计者的设计思想,对于设计强 hash 函数具有指导意义。下面是[DOBB96a]中给出的设计思想:

1. 使用两列的处理方法来增加在轮与轮之间寻找碰撞的复杂性,因为可以通过找轮与轮之间的碰撞来找压缩函数的碰撞。
2. 为简单起见,左右两列的处理过程本质上是相同的,但设计者认为有必要使两列的处理尽可能不同。他们认为不久可能出现对一列,甚至是对两列的攻击(每列三轮)。但是,若使两列的处理过程不同,则可以抗攻击。这些不同点是:
   a. 两列中使用的加法常量不同(见表 12.5)。
   b. 两列中基本逻辑函数(f_1 至 f_5)的使用顺序相反。
   c. 对消息分组的 32 位的字的处理顺序不同[见表 12.7(a)]。
3. RIPEMD-160 包含五轮而不是四轮运算,每步迭代中字 C 循环移动 10 次,除此以外,RIPEMD-160(见图 12.9)和 MD5 完全相同(见图 12.3)。因为其他循环次数均不为 10,所以字 C 的循环移动次数选择为 10 次。字 C 的这种循环移动可以避免在 MD5 中对最高有效位的攻击[BOER93]。
4. 置换 ρ[见表 12.7(a)]的作用是,使得在某轮中相近的两个字在下一轮中相差甚远;置换 π[见表 12.7(a)]的作用是,使得在左列中相近的两个字在右列中至少有 7 位不同。
5. 循环左移[见表 12.7(b)]的选择基于下述设计标准:
   a. 移动范围是 5 到 15。一般认为少于 5 次的移动较弱。
   b. 每个字在五轮中循环移动的次数不同,并且移动次数的奇偶性不完全相同。

c. 每个字的移动不应有特定规律(如总的移动次数不应被 32 整除)。

d. 不应有太多的移动次数能被 4 整除。

12.3.4　RIPEMD-160 与 MD5 和 SHA-1 的比较

RIPEMD-160、MD5 和 SHA-1 都基于 MD4,所以 RIPEMD-160 在许多方面与 MD5 和 SHA-1 相似,表 12.8 比较了它们的异同。下面我们从几个方面来评价这些算法:

- **抗穷举攻击的能力**:三个算法都不易受到对抗弱碰撞性的攻击。由于 MD5 使用的是 128 位的摘要,所以它极易受基于抗强碰撞性的生日攻击,而 SHA-1 和 RIPEMD-160 在不久的将来对基于抗强碰撞性的生日攻击是安全的。
- **抗密码分析的能力**:前面已经讨论过,对 MD5 的密码分析已取得了很大进展,RIPEMD-160 正是为抗已知密码分析攻击而设计的。虽然对 SHA-1 的设计原则知之甚少,但它抗已知的密码分析攻击的能力似乎很强。RIPEMD-160 使用平行的两列结构,执行的步数增加了一倍,也就增加了复杂性,所以与 SHA-1 相比,对 RIPEMD-160 的密码分析更加困难。

表 12.8　MD5、SHA-1 和 RIPEMD-160 的比较

	MD5	SHA-1	RIPEMD-160
摘要长度	128 位	160 位	160 位
基本处理单位	512 位	512 位	512 位
步数	64(4 轮,每轮 16 步)	80(4 轮,每轮 20 步)	160(5 轮,每轮 16 步)
最大消息长度	∞	$2^{64}-1$ 位	$2^{64}-1$ 位
基本逻辑函数	4	4	5
使用的加法常量	64	4	9
低位/高端位结构	低位在前结构	高位在前结构	低位在前结构

- **速度**:三个算法都基于模 2^{32} 加法和简单的位逻辑运算,所以它们在 32 位机上执行速度较快。由于 SHA-1 和 RIPEMD-160 比 MD5 更复杂且迭代步数更多,所以它们的执行速度比 MD5 要慢。表 12.9 给出了在 266 MHz Pentium 机上得到的一些结果。[BOSS96]中也给出了类似的结果。
- **低位在前结构与高位在前结构**:MD5 和 RIPEMD-160 使用低位在前的方法将消息分组解释为 32 位的字的序列,而 SHA-1 使用的是高位在前的方式。

表 12.9　几个 hash 函数的性能比较(在 850 MHz Celeron 上用 C++ 编写)

算　法	Mbps
MD5	26
SHA-1	48
RIPEMD-160	31

注:程序代码由 Wei Dai 编写,见 http://www.eskimo.com/~weidai/benchmarks.html。

12.4 HMAC

在第 11 章中,我们讨论了使用对称分组密码的消息认证码(MAC),即 FIPS PUB 113 中定义的消息认证算法,该算法一直是构造 MAC 的最常用方法。近年来,人们越来越感兴趣于利用密码 hash 函数来设计 MAC,因为:

1. 一般像 MD5 和 SHA-1 这样的 hash 函数,其软件执行速度比诸如 DES 这样的对称分组密码要快。
2. 可利用密码 hash 函数代码库。
3. 美国或其他国家对密码 hash 函数没有出口限制,而对即使是用于 MAC 的对称分组密码都有出口限制。

诸如 MD5 这样的 hash 函数并不是专为 MAC 而设计的,由于 hash 函数不依赖于秘密钥,所以它不能直接用于 MAC。目前,将密钥加到现有的 hash 函数中已经提出了许多方案,HMAC(RFC 2104)是最受欢迎的方案之一[BELL96a,BELL96b],它被选为 IP 安全中实现 MAC 必须使用的方法,并且其他 Internet 协议中(如 SSL)也使用了 HMAC。HMAC 也已作为 FIPS 草案公布(尚未赋予其序号)。

12.4.1 HMAC 设计目标

RFC 2104 给出了 HMAC 的设计目标:

- 不必修改而直接使用现有的 hash 函数。特别地,很容易免费得到软件上执行速度较快的 hash 函数及其代码。
- 如果找到或者需要更快或更安全的 hash 函数,应能很容易地替代原来嵌入的 hash 函数。
- 应保持 hash 函数的原有性能,不能过分降低其性能。
- 对密钥的使用和处理应较简单。
- 如果已知嵌入的 hash 函数的强度,则完全可以知道认证机制抗密码分析的强度。

前两个目标是 HMAC 为人们所接受的重要原因。HMAC将hash函数看做是"黑盒"有两个好处。第一,实现HMAC时可将现有hash函数作为一个模块,这样可以对许多HMAC代码预先封装,并在需要时直接使用;第二,若希望替代HMAC中的hash函数,则只需删去现有的hash函数模块并加入新的模块,例如需要更快的hash函数时就可如此处理。更重要的是,如果嵌入的hash函数的安全受到威胁,那么只需用更安全的hash函数替换嵌入的hash函数(如用SHA-1替代MD5),仍然可保持HMAC的安全性。

上述最后一个设计目标实际上是 HMAC 优于其他基于 hash 的一些方法的主要方面。只要嵌入的 hash 函数有合理的密码分析强度,则可以证明 HMAC 是安全的。本节后面再来讨论这个问题,下面我们先讨论 HMAC 的结构。

12.4.2 HMAC 算法

图 12.10 给出了 HMAC 的总体结构。定义下列符号:

H = 嵌入的 hash 函数(如 MD5,SHA-1,RIPEMD-160)

IV　= 作为 hash 函数输入的初始值
M　= HMAC 的消息输入(包括嵌入的 hash 函数中定义的填充位)
Y_i　= M 的第 i 个分组,$0 \leqslant i \leqslant (L-1)$
L　= M 中的分组数
b　= 每一分组所含的位数
n　= 嵌入的 hash 函数所产生的 hash 码长
K　= 密钥;若密钥长度大于 b,则将密钥作为 hash 函数的输入,来产生一个 n 位的密钥;建议密钥长度 $\geqslant n$
K^+　= 为使 K 为 b 位长而在 K 左边填充 0 后所得的结果
ipad　= 00110110(十六进制数 36)重复 $b/8$ 次的结果
opad　= 01011100(十六进制数 5C)重复 $b/8$ 次的结果

HMAC 可描述如下:

$$\mathrm{HMAC}_K(M) = \mathrm{H}[(K^+ \oplus \mathrm{opad}) \| \mathrm{H}[(K^+ \oplus \mathrm{ipad}) \| M]]$$

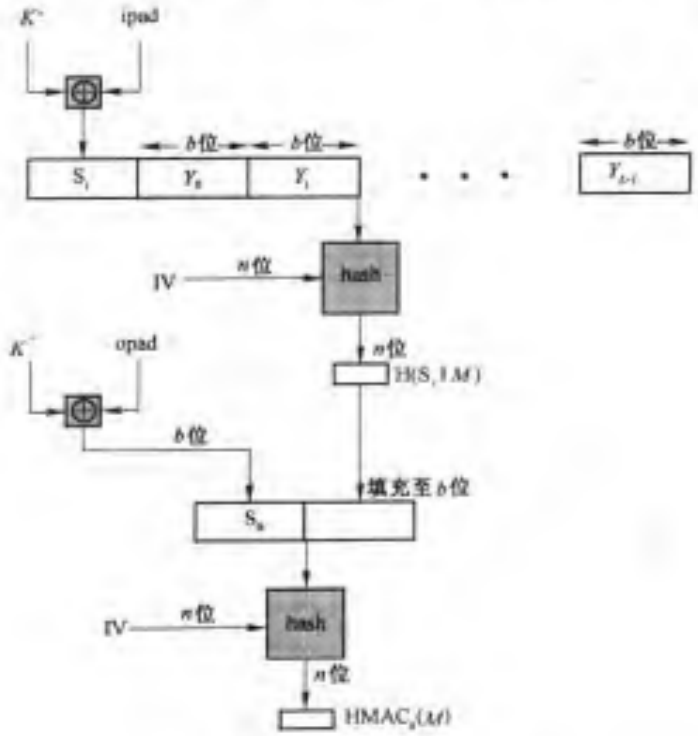


图 12.10　HMAC 的结构

也就是说,

1. 在 K 左边填充 0,得到 b 位的 K^+(例如,若 K 是 160 位,$b = 512$,则在 K 中加入 44 个 0 字节 0x00)。
2. K^+ 与 ipad 执行异或运算(位异或)产生 b 位的分组 S_i。
3. 将 M 附于 S_i 后。

4. 将 H 作用于步骤 3 所得出的结果。

5. K^+ 与 opad 执行异或运算(位异或)产生 b 位的分组 S_o。

6. 将步骤 4 中的 hash 码附于 S_o 后。

7. 将 H 作用于步骤 6 所得出的结果,并输出该函数值。

注意,K 与 ipad 异或后,其信息位有一半发生了变化;同样,K 与 opad 异或后,其信息位的另一半也发生了变化,这样,通过将 S_i 与 S_o 传给 hash 算法中的压缩函数,我们可以从 K 伪随机地产生出两个密钥。

HMAC 多执行了三次 hash 压缩函数(对 S_i、S_o 和内部的 hash 产生的分组),但是对于长消息,HMAC 和嵌入的 hash 函数的执行时间应该大致相同。

实现 HMAC 有更为有效的方法,如图 12.11 所示。我们可以预计算两个值:

$$f(IV, (K^+ \oplus ipad))$$

$$f(IV, (K^+ \oplus opad))$$

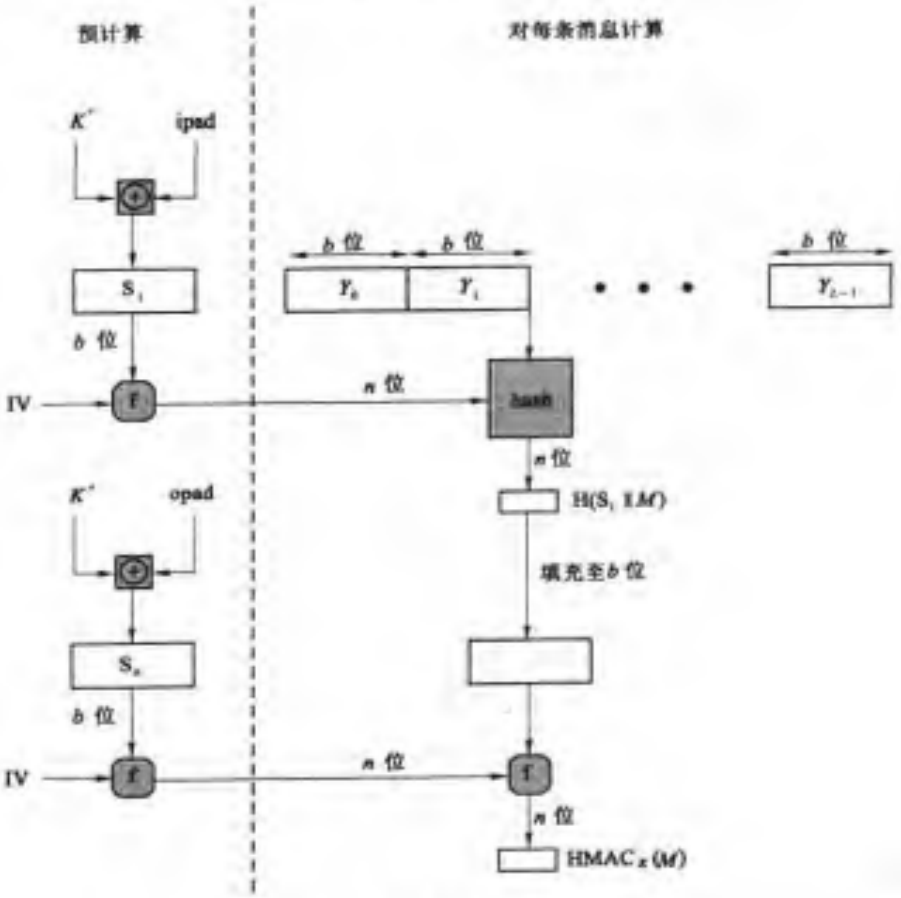


图 12.11 HMAC 的有效实现方案

其中 f(cv, block)是 hash 函数的压缩函数,其输入是 n 位的链接变量和 b 位的分组,输出是 n 位的链接变量。上述这些值只在初始化或密钥改变时才需计算,实际上,这些预先计算的值取代

了 hash 函数中的初值(IV)。这样,只多执行了一次压缩函数,在大多数产生 MAC 的消息都较短的情况下,这种实现特别有意义。

12.4.3 HMAC 的安全性

任何建立在嵌入 hash 函数基础上的 MAC,其安全性在某种程度上依赖于该 hash 函数的强度。HMAC 的好处在于,其设计者可以证明嵌入的 hash 函数的强度与 HMAC 的强度之间的联系。

根据伪造者在给定时间内伪造成功和用相同密钥产生给定数量的消息-MAC 对的概率,可以来描述 MAC 函数的安全性。本质上,这些消息是由合法用户产生的。[BELL96a]中已经证明,如果攻击者已知若干(时间,消息-MAC)对,则成功攻击 HMAC 的概率等价于对嵌入 hash 函数的下列攻击之一:

1. 即使对攻击者而言,IV 是随机的、秘密的和未知的,攻击者也能计算压缩函数的输出。
2. 即使 IV 是随机的和秘密的,攻击者也能找到 hash 函数中的碰撞。

在第一种攻击中,我们可将压缩函数看做是将 hash 函数应用于只含有一个 b 位分组的消息,hash 函数的 IV 被一个 n 位秘密的随机值代替。攻击该 hash 函数或者是对密钥的穷举攻击(其代价为 2^n 数量级),或者是生日攻击(这是第二种攻击的特例,请见下面的讨论)。

在第二种攻击中,攻击者要找两条消息 M 和 M',它们产生相同的 hash 码:$H(M) = H(M')$,这就是第 11 章中所讨论的生日攻击,我们已经证明其所需的代价为 $2^{n/2}$ 数量级。根据现在的技术,若代价为 2^{64} 数量级,则被认为是可行的,所以 MD5 的安全性不能得到保证。但是,这是否意味着像 MD5 这样的 128 位的 hash 函数不能用于 HMAC 呢?回答是否定的,因为要攻击 MD5,攻击者可以选择任何消息集,并用专用计算机离线计算来寻找碰撞,由于攻击者知道 hash 算法和默认的 IV,因此攻击者可以对其产生的任何消息计算 hash 码。但是,攻击 HMAC 时,由于攻击者不知道 K,所以他不能离线产生消息/hash 码对,他必须观察 HMAC 用相同的密钥产生的消息序列,并对这些消息进行攻击。hash 码长为 128 位时,攻击者必须观察 2^{64} 个由同一密钥产生的分组(2^{73} 位),对于 1 Gbps 连接,要想攻击成功,攻击者约需 250 000 年来观察用同一密钥产生的连续的消息流。因此,当注重执行速度时,用 MD5 而不是 SHA-1 和 RIPEMD-160 作为 HMAC 的嵌入 hash 函数,完全是可以接受的。

12.5 推荐读物和网址

[DOBB96a]讨论了 MD5 的安全性。[TOUC95]描述了 MD5 的有效实现技术。[ROBS95c]对 MD5 和 SHA 进行了比较。[BOSS97]和[PREN97]讨论了 RIPEMD-160。

BOSS 97 Bosselaers, A.; Dobbertin, H.; and Preneel, B. "The RIPEMD-160 Cryptographic Hash Function." *Dr. Dobb's Journal*, January 1997.

DOBB96a Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

PREN 97 Preneel B.; Bosselaers, A.; and Dobbertin, H. "The Cryptographic Hash Function RIPEMD-160." *CryptoBytes*, Autumn 1997.

ROBS95c Robshaw, M. *MD2, MD4, MD5, SHA and Other Hash Functions*. RSA Laborato-

ries Technical Report TR-101, July 1995. http://www.rsasecurity.com/rsalabs/index.html

TOUC95 Touch, J. "Performance Analysis of MD5." *Proceedings*, SIGCOMM'95, October 1995.

推荐网址：

- **NIST Secure Hashing Pages**：SHA FIPS 和有关文档。
- **RIPEMD-160 Page**：有关 RIPEMD-160 的资料。

12.6 关键术语、思考题和习题

12.6.1 关键术语

高位在前格式	MD4	SHA-256
压缩函数	MD5	SHA-384
HMAC	RIPEMD-160	SHA-512
低位在前格式	SHA-1	

12.6.2 思考题

12.1 高位在前格式和低位在前格式的区别是什么？

12.2 MD5 中使用的基本算术和逻辑函数是什么？

12.3 SHA-1 中使用的基本算术和逻辑函数是什么？

12.4 RIPEMD-160 中使用的基本算术和逻辑函数是什么？

12.5 为什么人们一直用密码 hash 函数而不是用对称密码来构造消息认证码？

12.6 为了用一个 hash 函数替代另一个 hash 函数，HMAC 中需要进行哪些改变？

12.6.3 习题

12.1 说明 MD4 和 MD5 的不同点。特别地，你认为在哪些方面 MD5 强于 MD4，为什么？

12.2 图 12.7 中假定可用 80 个 32 位的字的数组来存储 W_t 的值，所以在开始处理分组时，可以预先计算这些值。现在假定只有 16 个字的循环缓冲区可用，其初值为 W_0 到 W_{15}。试设计一个算法，对每个 t，计算所需的输入值 W_t。

12.3 给出 SHA-1 中 W_{16}、W_{17}、W_{18}、W_{19}的值。

12.4 假定 $a_1 a_2 a_3 a_4$ 是字长为 32 位的字的 4 个字节，将每个 a_i 看做是 0 到 255 之间的二进制整数，在高位在前结构中，该字表示整数

$$a_1 2^{24} + a_2 2^{16} + a_2 2^8 + a_4$$

在低位在前结构中，该字表示整数

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$$

a. MD5 和 RIPEMD-160 使用的是低位在前结构。注意，消息摘要不依赖于低层结构这一点很重要，因此要在高位在前结构上执行 MD5 和 RIPEMD-160 的模 2 加法运算，必须进行调整。假定 $X = x_1 x_2 x_3 x_4$ 且 $Y = y_1 y_2 y_3 y_4$，说明 MD5 的加法运算 $(X + Y)$ 如何在高位在前结构的机器上执行。

b. SHA-1 使用高位在前结构，说明 SHA-1 的运算 $(X + Y)$ 如何在低位在前结构的机器上执行。

第 13 章　数字签名和认证协议

数字签名是公钥密码学发展过程中最重要的概念之一，它可以提供其他方法难以实现的安全性。本章先简要介绍数字签名的有关内容，然后介绍认证协议，其中许多协议与数字签名有关，最后介绍数字签名标准(DSS)。

13.1　数字签名

13.1.1　对数字签名的要求

消息认证可以保护信息交换双方不受第三方的攻击，但是它不能处理通信双方自身发生的争攻击，这种攻击可以有多种形式。

例如，假定 John 使用图 11.4 中的某种方法给 Mary 发送一条认证消息。考虑下面两种情形：

1. Mary 可以伪造一条消息并称该消息发自 John。Mary 只需产生一条消息，用 John 和Mary 共享的密钥产生认证码，并将认证码附于消息之后。
2. John 可以否认曾发送过某条消息。因为 Mary 可以伪造消息，所以无法证明 John 确实发送过该消息。

这两种情形都是法律关注的。例如，对于第一种情形，在进行电子资金转账时，接收方可以增加转账资金，并声称这是来自发送方的转账资金额；对于第二种情形，股票经纪人收到有关电子邮件消息，要他进行一笔交易，而这笔交易后来失败了，但是发送方可以伪称从未发送过这条消息。

在收发双方不能完全信任的情况下，就需要除认证之外的其他方法来解决这些问题。数字签名是解决这个问题的最好方法，它的作用相当于手写签名。数字签名必须具有下列特征：

- 它必须能验证签名者、签名日期和时间。
- 它必须能认证被签的消息内容。
- 签名应能由第三方仲裁，以解决争执。

因此，数字签名具有认证功能。

根据这些特征，数字签名应满足下列条件：

- 签名必须是与消息相关的二进制位串。
- 签名必须使用发送方某些独有的信息，以防伪造和否认。
- 产生数字签名比较容易。
- 识别和验证签名比较容易。
- 伪造数字签名在计算上是不可行的。无论是从给定的数字签名伪造消息，还是从给定的消息伪造数字签名，在计算上都是不可行的。

- 保存数字签名的拷贝是可行的。

图 11.5(c)或图 11.5(d)所示的方法所使用的安全 hash 函数满足上述条件。

数字签名的方法有多种，这些方法可分为两类：直接数字签名和仲裁数字签名。

13.1.2　直接数字签名

直接数字签名只涉及通信双方。假定接收方已知发送方的公钥，则发送方可以通过用自己的私钥对整个消息[见图 11.1(c)]或消息的 hash 码[见图 11.5(c)]加密来产生数字签名。

用接收方的公钥(公钥密码)和共享的密钥(对称密码)再对整个消息和签名加密，则可以获得保密性，请参见图 11.1(d)和图 11.5(d)。注意这里是先进行签名，然后才执行外层的加密，这样在发生争执时，第三方可以查看消息及其签名。这种先后次序非常重要，若先对消息加密，然后才对消息的密文签名，那么第三方必须知道解密密钥才能读取原始消息。但是签名若是在内层进行，那么接收方可以存储明文形式的消息及其签名，以备将来解决争执时使用。

上述直接签名方法都有这样一个弱点，即这些方法的有效性依赖于发送方私钥的安全性。如果发送方想否认以前曾发送过某条消息，那么他可以称其私钥已丢失或被盗用，其他人伪造了他的签名。虽然在某种程度上这种威胁可能存在，但我们可以通过在私钥的安全性方面进行控制来阻止或至少减少这种情况的发生。例如，可以要求每条要签名的消息都包含一个时间戳(日期和时间)，以及在密钥被泄密后应立即向管理中心报告。

另一种可能的威胁是，X 的私钥可能在时刻 T 被盗用，但攻击者可用 X 的签名签发一条消息并加盖一个在 T 或 T 之前的时间戳。

13.1.3　仲裁数字签名

仲裁签名可以解决直接数字签名中出现的问题。

与直接签名方法一样，也有多种仲裁签名方法。一般来说，这些方法都是这样执行的：从发送方 X 到接收方 Y 的每条已签名的消息都先发送给仲裁者 A，A 对消息及其签名进行检查以验证消息源及其内容，然后给消息加上日期并发送给 Y，同时指明该消息已通过仲裁者的检验。A 的加入解决了直接数字签名所面临的问题，即 X 可能否认发送过这条消息。

在这种类型的方法中，仲裁者起着关键作用，通信各方都应非常信任仲裁机制。使用第 20 章中讲到的可信系统可以满足这个要求。

表 13.1 给出了几个仲裁签名的例子[①]，它们都基于[AKL83]和[MITC92]中讲到的方法。第一个例子使用的是对称密码，假定发送方 X 和仲裁者 A 共享密钥 K_{ax}，A 和 Y 共享密钥 K_{ay}。X 产生消息 M 并计算出其 hash 值 H(M)，然后 X 将消息 M 和签名发送给 A，其签名由 X 的标识 ID_X 和 hash 值组成，并且用 K_{ax} 加密。A 对签名解密后，通过检查 hash 值来验证该消息的有效性，然后 A 用 K_{ay} 对 ID_X、发自 A 的原始消息 M、签名和时间戳加密后传给 Y。Y 对 A 发来的消息解密即可恢复消息 M 和签名。时间戳告诉 Y 该消息是及时的消息，而不是重放的消息。Y 可以存储 M 和签名，发生争执时 Y 可将下列消息发给 A，以证明曾收到过来自 X 的消息：

① 我们使用下述格式：用 P→Q: M 表示 P 发送消息 M 给 Q 的通信。

$$E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)]]$$

表 13.1 仲裁数字签名方法

(a) 传统加密，仲裁者能阅读消息

(1) X → A: $M \| E_{K_{xa}}[ID_X \| H(M)]$
(2) A → Y: $E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$

(b) 传统加密，仲裁者不能阅读消息

(1) X → A: $ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])]$
(2) A → Y: $E_{K_{ay}}[ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])] \| T]$

(c) 公钥加密，仲裁者不能阅读消息

(1) X → A: $ID_X \| E_{KR_x}[ID_X \| E_{KU_y}(E_{KR_x}[M])]$
(2) A → Y: $E_{KR_a}[ID_X \| E_{KU_y}[E_{KR_x}[M]] \| T]$

其中：X = 发送方，Y = 接收方，A = 仲裁者，M = 消息，T = 时间戳。

仲裁者先用 K_{ay} 恢复出 ID_X、M 和签名，然后用 K_{xa} 解密该签名并验证其 hash 码。在这种方法中，签名只是用于解决争执，Y 不能直接读取 X 的签名。因为消息来自 A，所以 Y 认为来自 X 的消息是真实的，这种方法中双方应对 A 高度信任：

- X 必须相信 A 不会泄漏 K_{xa}，并且不会产生形为 $E_{K_{xa}}[ID_X \| H(M)]$的伪造签名。
- Y 必须相信 A 只在 hash 值正确并且签名确实是由 X 产生时，才发送 $E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$。
- 双方必须相信 A 会公正地解决争执。

如果仲裁者 A 确实能做到这一点，那么 X 可以确信没人能伪造他的签名，Y 可以确信 X 无法否认他的签名。

上面这种方法中，A 可以读取 X 发送给 Y 的消息，实际上任何窃听者都能读取该消息。表 13.1(b)给出的方法可以提供仲裁签名，而且还能保证消息的保密性。在这种方法中，假定 X 和 Y 共享密钥 K_{xy}。X 用 K_{xa} 对其标识、用 K_{xy} 加密后的消息的 hash 值加密产生签名，然后将其标识、用 K_{xy} 加密后的消息和签名发送给 A。和前面一样，A 先对签名解密并通过检验 hash 值来验证消息的有效性，由于 A 处理的是加密后的消息，因此 A 不能读取消息。A 用 K_{ay} 对 X 发来的所有内容以及一个时间戳加密后发送给 Y。

尽管仲裁者不能读取消息，但他仍然可以防止 X 或 Y 的欺诈行为。但这种方法和第一种方法都存在这样一个问题，即仲裁者可能与发送方共同否认一个已签名的消息，或者与接收方共同伪造发送方的签名。

公钥方法可以解决上述所有问题，其中一种公钥方法如表 13.1(c)所示。X 对消息 M 两次加密，即先用其私钥 KR_x 对消息 M 加密，然后再用 Y 的公钥 KU_y 加密，得到加密后的签名；X 再用 KR_x 对其标识和上述加密后的签名加密并连同 ID_X 一起发送给 A。上述内层两次加密后的消息对仲裁者(以及除 Y 外的所有人)是秘密的，但是 A 可以对外层的加密进行解密以验证消息确实发自 X(因为只有 X 有私钥 KR_x)。A 检查 X 的公/私钥对是否仍然有效，若是，则消息是有效的。然后 A 再用 KR_a 对 ID_X、两次加密后的消息和一个时间戳加密后传送给 Y。

与前面两种方法相比，这种方法有许多优点。第一，通信各方在通信前不共享任何信息，从而避免联合欺诈；第二，即使 KR_x 已泄密，但 KR_a 没有泄密，那么时间戳不正确的消息是不能被发送的；最后，X 发送给 Y 的消息对 A 或其他人来讲是保密的。但最后这种方法中用公钥对消息加密了两次。我们后面会讨论更实用的方法。

13.2　认证协议

第 11 章中讨论的那些基本方法已越来越广泛地用于各种应用之中，如 13.1 节中讨论的数字签名。本节我们主要讨论相互认证和单向认证，并介绍其中的一些认证方法。

13.2.1　相互认证

相互认证是一个重要的应用领域，相互认证协议可以使通信双方达成一致并交换会话密钥。7.3 节（对称方法）和 10.1 节（公钥方法）讨论这个问题时，侧重的是密钥分配，本节我们讨论有关认证的更广泛的内容。

保密性和及时性是认证的密钥交换中两个重要的问题。为防止假冒和会话密钥的泄密，用户标识和会话密钥这样的重要信息必须以密文的形式传送，这就需要事先已有能用于这一目的的密钥或公钥。因为可能存在消息重放，所以及时性非常重要，在最坏情况下，攻击者可以利用重放攻击威胁会话密钥或者成功地假冒另一方。成功的消息重放所提供的消息看似正确，但实际并不正确，这至少会影响正常的操作。

[GONG93]列出了如下一些重放攻击的例子：

- **简单重放**：攻击者只是简单地复制消息并在以后重放这条消息。
- **可检测的重放**：攻击者在有效的时限内重放有时间戳的消息。
- **不可检测的重放**：由于原始消息可能被禁止而不能到达其接收方，所以只有通过重放消息才能将消息发送到接收方，此时可能出现这种攻击。
- **不加修改的逆向重放**：这是向消息发送方的重放。如果使用对称密码并且发送方不能根据内容来区分发出的消息和接收的消息，那么可能出现这种攻击。

对付重放攻击的方法之一是，在每个用于认证交换的消息后附加一个序列号，只有序列号正确的消息才能被接受。但是这种方法存在这样一个问题，即它要求每一通信方都要记录其他通信各方最后的序列号。因此，认证和密钥交换一般不使用序列号，而是使用下列两种方法之一：

- **时间戳**：仅当消息包含时间戳并且在 A 看来这个时间戳与其所认为的当前时间足够接近时，A 才认为收到的消息是新消息，这种方法要求通信各方的时钟应保持同步。
- **随机数/响应**：若 A 要接收 B 发来的消息，则 A 首先给 B 发送一个临时交互号（随机数），并要求 B 发来的消息（响应）包含该临时交互号。

由于时间戳方法本身存在的一些困难，有人认为时间戳方法不适合于面向连接的应用（如[LAM92a]）。第一，它需要某种协议保持通信各方的时钟同步，为了能够处理网络错误，该协议必须能够容错，并且还应能抗恶意攻击；第二，如果由于通信一方时钟机制出错而使同步失

效，那么攻击成功的可能性就会增大；第三，由于各种不可预知的网络延时，不可能保持各分布时钟精确同步。因此，任何基于时间戳的程序都应有足够长的时限以适应网络延时，同时应有足够短的时限以使攻击的可能性最小。

另一方面，随机数/响应不适合于无连接的应用，因为它要求在任何无连接传输之前必须先握手，这与无连接交易的主要特征相违背。对这种应用，最好的方法是通过某种安全的时间服务器以及通信各方的协调努力来保持时钟同步（如[LAM92b]）。

对称加密方法

正如5.3节所讲到的，可以通过使用两层传统加密密钥结构来保证分布式环境中通信的保密性。通常，这种方法要使用一个可信的密钥分配中心（KDC）。在网络中，各方与KDC共享一个称为主密钥的密钥，KDC负责产生通信双方通信时短期使用的密钥（称为会话密钥），并用主密钥保护这些会话密钥的分配。这种方法的使用非常普遍，例如第14章中的Kerberos系统使用的就是这种方法。我们通过本小节的讨论来理解Kerberos系统。

图7.9给出了利用KDC进行密钥分配的协议[NEED78]，该协议最初是由Needham和Schroeder提出的，第7章中已讲过这个协议具有认证功能。该协议可归纳如下：

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E_{K_a}[K_s \| ID_B \| N_1 \| E_{K_b}[K_s \| ID_A]]$
3. A → B: $E_{K_b}[K_s \| ID_A]$
4. B → A: $E_{K_s}[N_2]$
5. A → B: $E_{K_s}[f(N_2)]$

A、B和KDC分别共享密钥K_a和K_b，该协议的目的是要保证将会话密钥K_s安全地分配给A和B。A在步骤2可以安全地获得新的会话密钥；因为只有B可以解密步骤3中的消息，所以只有B可理解该消息，步骤4说明B已知K_s，步骤5使B确信A已知K_s，由于使用了临时交互号N_2，B可确信该消息是一条新消息，这是因为根据第5章中的讨论，步骤4和步骤5的目的就是要防止重放攻击，特别地，如果攻击者可以截获步骤3中的消息并重放这条消息，则可能会破坏B的操作。

尽管有步骤4和步骤5的握手，但这种协议仍然容易受重放攻击。假定攻击者X已知一个旧会话密钥，虽然这种情况比攻击者只是截获步骤3中的消息的可能性要小得多，但这确是一种潜在的威胁。X可以模仿A重放步骤3中的消息并诱使B用旧会话密钥进行通信，除非B明确记得以前与A通信所使用的所有会话密钥，否则B无法确定是否是重放的消息。若X能够截获步骤4中的握手消息，则他可以模仿步骤5中A的应答，从这时开始，X可以发送伪造的消息给B，而B以为这是发自A的用已认证的会话密钥加密后的消息。

Denning[DENN81, DENN82]对Needham/Schroeder协议进行了修改，她在步骤2和步骤3中加入了时间戳，从而解决了上述问题。在Denning提出的协议中，假定主密钥K_a和K_b是安全的，该协议包括下列步骤：

1. A → KDC: $ID_A \| ID_B$
2. KDC → A: $E_{K_a}[K_s \| ID_B \| T \| E_{K_b}[K_s \| ID_A \| T]]$
3. A → B: $E_{K_b}[K_s \| ID_A \| T]$
4. B → A: $E_{K_s}[N_1]$
5. A → B: $E_{K_s}[f(N_1)]$

时间戳 T 使 A 和 B 确信该会话密钥是刚刚产生的，这样 A 和 B 都知道此次交换的是一个新会话密钥。A 和 B 通过检验下式来验证及时性：

$$|\text{时钟} - T| < \Delta t_1 + \Delta t_2$$

其中 Δt_1 是 KDC 的时钟与(A 或 B 的)本地时钟正常误差的估计值，Δt_2 是预计的网络延时。每个节点可以根据某个标准的参考源设置其时钟。由于是用保密的主密钥对时间戳加密，所以即使攻击者知道旧的会话密钥，他也不能成功地重放消息，因为 B 可以根据消息的及时性检测出步骤 3 中的重放消息。

Denning 最初提出的协议[DENN81]中并没有包含步骤 4 和步骤 5，后来才将它们加入到协议中[DENN82]，用来确认 B 收到了会话密钥。

与 Needham/Schroeder 协议相比，Denning 协议的安全程度似乎更高。但是 Denning 协议中又出现了新的问题，即它依赖于时钟，而这些时钟应在整个网络上保持同步。[GONG92]指出了其中存在的危险，如对时钟或同步机制出错或者被破坏，分布的时钟就可能会不同步①。若发送方的时钟超前于接收方的时钟，则攻击者可以截获消息并在消息内的时间戳是接收方的当前时间时重放该消息，这种重放将会引起不可预知的结果，Gong 称这种重放攻击为**禁止-重放攻击**。

解决禁止-重放攻击的一种方法是，要求通信各方必须根据 KDC 的时钟周期性地校验他们的时钟。另一种方法建立在使用临时交互号的握手协议之上，它不要求时钟同步，并且接收方选择的临时交互号对发送方而言是不可预知的，所以不易受禁止 - 重放的攻击。Needham/Schroeder 协议使用了临时交互号，但是我们已经了解它存在其他弱点。

在[KEHN92]中曾尝试解决禁止 - 重放攻击问题，并且同时解决 Needham/Schroeder 协议中出现的问题结果发现后一协议中存在不一致性，因此在[NEUM93a]中提出了如下协议②：

1. A → B:　$ID_A \| N_a$
2. B → KDC:　$ID_B \| N_b \| E_{K_b}[ID_A \| N_a \| T_b]$
3. KDC → A:　$E_{K_a}[ID_B \| N_a \| K_s \| T_b] \| E_{K_b}[ID_A \| K_s \| T_b] \| N_b$
4. A → B:　$E_{K_b}[ID_A \| K_s \| T_b] \| E_{K_s}[N_b]$

下面我们逐条分析该协议。

1. A 初始化该认证交换。A 产生临时交互号 N_a，并将其标识和 N_a 以明文的形式发送给 B。该临时交互号将和会话密钥等一起被加密后返回给 A，以使 A 确信消息的及时性。
2. B 向 KDC 申请一个会话密钥。B 发送给 KDC 的消息包括 B 的标识和临时交互号 N_b。该临时交互号将和会话密钥等一起加密后返回给 B，以使 B 确信消息的及时性。B 发送给 KDC 的消息中还包括用 B 和 KDC 共享的密钥加密后的信息，该信息用于请求 KDC 给 A 发证书，它指定了证书接收方、证书的有效期和收到的 A 的临时交互号。
3. KDC 将 B 的临时交互号、用 KDC 和 B 共享的密钥加密后的信息发送给 A。我们将会看到该信息可用做 A 进行后续认证的一张"证明书"。同时 KDC 将用它和 A 共享的密钥

① 这种情况可能发生，而且确实发生过。最近这些年中，许多计算机和其他的电子系统使用有缺陷的芯片来记录日期和时间，这些芯片的计时有可能会超前一天[NEUM90]。

② 构造正确的协议的确很难。

加密后的信息发送给 A，该信息可用来验证 B 曾收到过 A 最初发出的消息（ID_B），并可说明该消息是及时的消息，而不是重放的消息（N_a），同时 A 可从中得出会话密钥（K_s）及其使用时限（T_b）。

4. A 将证书和用会话密钥加密后的 B 的临时交互号发送给 B。B 可由该证书求得解密 $E_{K_s}[N_b]$ 的密钥，从而得出 N_b。用会话密钥对 B 的临时交互号加密可保证该消息是来自 A 的非重放消息。

上述协议提供了 A 和 B 通过会话密钥建立会话的安全有效手段。该协议使 A 拥有可用于对 B 进行后续认证的密钥，避免了与认证服务器的重复联系。假定 A 和 B 用上述协议建立并结束了一个会话，并且在该协议所建立的时限内 A 希望与 B 进行新的会话，则可使用下述协议：

1. $A \rightarrow B$: $E_{Kb}[ID_A \| K_s \| T_b], N'_a$
2. $B \rightarrow A$: $N'_b, E_{K_s}[N'_a]$
3. $A \rightarrow B$: $E_{K_s}[N'_b]$

B 在步骤 1 收到消息后，可以验证证书没有失效。新产生的 N'_a 和 N'_b 使双方确信没有重放攻击。

在前面的讨论中，T_b 指的是相对于 B 的时钟的时间，因为 B 只校验自身产生的时间戳，所以并不要求时钟同步。

公钥加密方法

在第 10 章，我们给出了用公钥密码进行会话密钥分配的方法（见图 10.6）。该协议假定通信的每一方都拥有另一方的当前公钥，但这个假设可能是不切实际的。

[DENN81]中给出了一种使用时间戳的协议：

1. $A \rightarrow AS$: $ID_A \| ID_B$
2. $AS \rightarrow A$: $E_{KR_{as}}[ID_A \| KU_a \| T] \| E_{KR_{as}}[ID_B \| KU_b \| T]$
3. $A \rightarrow B$: $E_{KR_{as}}[ID_A \| KU_a \| T] \| E_{KR_{as}}[ID_B \| KU_b \| T] \| E_{KU_b}[E_{KR_{as}}[K_s \| T]]$

这里的中心系统并不真正负责密钥分配，而是提供公钥证书，所以称为认证服务器（AS）。因为 A 选择并加密会话密钥，因此不存在会话密钥被 AS 泄密的危险；时间戳可防止用已泄密的密钥进行重放攻击。

上述协议简洁明了，但仍要求时钟同步。Woo 和 Lam[WOO92a]提出了一种使用临时交互号的方法。该协议包含如下步骤：

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E_{KR_{auth}}[ID_B \| KU_b]$
3. $A \rightarrow B$: $E_{KU_b}[N_a \| ID_A]$
4. $B \rightarrow KDC$: $ID_B \| ID_A \| E_{KU_{auth}}[N_a]$
5. $KDC \rightarrow B$: $E_{KR_{auth}}[ID_A \| KU_a] \| E_{KU_b}[E_{KR_{auth}}[N_a \| K_s \| ID_B]]$
6. $B \rightarrow A$: $E_{KU_a}[E_{KR_{auth}}[N_a \| K_s \| ID_B] \| N_b]$
7. $A \rightarrow B$: $E_{K_s}[N_b]$

在步骤 1 中，A 告诉 KDC 他想与 B 建立安全连接，KDC 将 B 的公钥证书的副本返回给 A（步骤 2），A 通过 B 的公钥告诉 B 想与之通信，同时将临时交互号 N_a 发送给 B(步骤 3)。在步骤 4，B 向 KDC 索要 A 的公钥证书并请求会话密钥，由于 B 发送的消息中包含 A 的临时交互号，因此 KDC 可以用该临时交互号对会话密钥加戳，其中该临时交互号受 KDC 的公钥保护。在步骤 5，KDC 将 A 的公钥证书的副本和消息 $\{N_a, K_s, ID_B\}$ 一起返回给 B。这条消息说明，K_s 是 KDC 为 B 产生的且与 N_a 有关的密钥。K_s 和 N_a 使 A 确信 K_s 是新会话密钥。用 KDC 的私钥对三元组 $\{N_a, K_s, ID_B\}$ 加密，使得 B 可以验证该三元组确实发自 KDC；由于是用 B 的公钥对该三元组加密，因此其他各方均不能利用该三元组与 A 建立假冒连接。在步骤 6，B 用 A 的公钥对 $E_{KR_{auth}}[N_a \| K_S \| ID_B]$ 和 B 产生的 N_b 加密后发送给 A。A 先解密得出会话密钥 K_s，然后用 K_s 对 N_b 加密后发送给 B，这样可使 B 确信 A 已经知道了会话密钥。

这个协议看起来对各种攻击都是安全的，然而 Woo 和 Lam 自己发现了问题并在[WOO92b]中给出了修改后的算法：

1. A → KDC: $ID_A \| ID_B$
2. KDC → A: $E_{KR_{auth}}[ID_B \| KU_b]$
3. A → B: $E_{KU_b}[N_a \| ID_A]$
4. B → KDC: $ID_B \| ID_A \| E_{KU_{auth}}[N_a]$
5. KDC → B: $E_{KR_{auth}}[ID_A \| KU_a] \| E_{KU_b}[E_{KR_{auth}}[N_a \| K_s \| ID_B]]$
6. B → A: $E_{KU_a}[E_{KR_{auth}}[N_a \| K_s \| ID_A \| ID_B] \| N_b]$
7. A → B: $E_{K_s}[N_b]$

在步骤 5 和步骤 6 中，加入了 A 的标识 ID_A，从而就将会话密钥 K_s 与会话双方的标识连接在一起。增加 ID_A 说明，临时交互号值 N_a 仅在 A 所产生的所有临时交互号中惟一，而不是在通信各方产生的所有临时交互号中惟一，因此 $\{ID_A, N_a\}$ 惟一标识了 A 的连接请求。

从这个例子和前面讲到的协议中可知，看似安全的协议在经过进一步分析后可能仍需修改，可见在认证领域要保证正确性的困难程度。

13.2.2　单向认证

电子邮件是广泛使用加密的应用之一，它的基本特点和主要益处是，收发双方不需同时在线联系，而是电子邮件消息发送到接收方的电子邮箱中，并一直存放在邮箱中，直至接收方读取为止。

“信封”或消息的报头必须是明文形式，这样它们才可由诸如简单邮件传输协议（SMTP）或 X.400 的存储－转发电子邮件协议来处理。但是，通常希望邮件处理协议不要访问明文形式的消息，因为这会要求可信的邮件处理机制，所以应该对电子邮件消息加密，并且邮件处理系统不拥有解密密钥。

另外还应有认证。一般来说，接收方希望能够保证其接受的消息确实来自真正的发送方。

对称加密方法

图 7.11 所示的用传统密码实现密钥分配的方法是不可行的，它要求发送方向预期的接收方提出请求，并等待一个包含会话密钥的响应，只有收到响应后才能发送消息。

对上述方法稍做修改，可将图 7.9 所示的 KDC 方法用于加密的电子邮件应用中。因为我们希望不必要求接收方(B)和发送方(A)同时在线，所以应删去步骤 4 和步骤 5。对内容为 M 的消息，发送序列如下：

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E_{K_a}[K_s \| ID_B \| N_1 \| E_{K_b}[K_s \| ID_A]]$
3. A → B: $E_{K_b}[K_s \| ID_A] \| E_{K_s}[M]$

该方法保证只有真正的消息接收方才能读取消息，同时也可证明发送方确实是 A。前面已经说过，该协议不能抗重放攻击，但通过在消息中加入时间戳可以提供一定的抗攻击能力。由于对电子邮件的处理中可能存在大量延时，因此时间戳的用途是有限的。

公钥加密方法

我们已经讨论过适用于电子邮件的公钥密码方法，如对整条消息直接加密以获得保密性[见图 11.1(b)]或者真实性[见图 11.1(c)]或者同时获得保密性和真实性[见图 11.1(d)]。但这些方法要求发送方已知接收方的公钥(保密性)或者接收方已知发送方的公钥(真实性)，或者要求通信双方同时已知对方的公钥；对于长消息，这些方法也必须使用一次或两次公钥算法。

如果主要关心的是保密性，那么下面的方法可能更有效：

$$A \rightarrow B: E_{KU_b}[K_s] \| E_{K_s}[M]$$

这里，A 使用一次性密钥对消息加密，且用 B 的公钥对该一次性密钥加密，而只有 B 能用相应的私钥恢复该一次性密钥，然后用它解密消息。这种方法比直接用 B 的公钥对整个消息加密更有效。

如果主要关心的是真实性，那么像图 11.5(c)所示的那样，使用数字签名就足够了：

$$A \rightarrow B: M \| E_{KR_a}[H(M)]$$

这种方法可保证 A 以后不能否认曾发送过消息，但它完全不能防止下述伪冒。Bob 要发送消息给 Alice，该消息中包含可以节约公司资金的计划，Bob 将其数字签名附在消息后并将它发送到电子邮件系统。这条消息最终会送到 Alice 的邮箱中。但是，假定 Max 得知了 Bob 的计划并且他在邮件发到 Alice 的邮箱之前能够访问邮件队列，那么他可以找到 Bob 的消息，用自己的签名替代 Bob 的后，再把消息加入到发给 Alice 的邮件队列中，从面 Max 因 Bob 的计划得到信任。

要解决这个问题，可以用接收方的公钥同时对消息和签名加密：

$$A \rightarrow B: E_{KU_b}[M \| E_{KR_a}[H(M)]]$$

后两种方法均要求 B 已知 A 的公钥，并且确信该公钥是 A 的现用公钥。第 10 章中所讨论的数字证书方法是满足上述要求的有效方法。我们有：

$$A \rightarrow B: M \| E_{KR_a}[H(M)] \| E_{KR_{as}}[T \| ID_A \| KU_a]$$

A 用私钥对其签名加密并用认证服务器的私钥对其证书加密，然后将它们连同消息一起发送给 B。接收方 B 先从证书中获得发送方的公钥并验证它的真实性，然后使用公钥来验证消息。如果要求保密性，一种方法是用 B 的公钥对整条消息加密，另一种方法是使用一次性密钥对整条消息加密，并用 B 的公钥对该密钥加密后一起发送给 B，我们将在第 15 章中讨论这种方法。

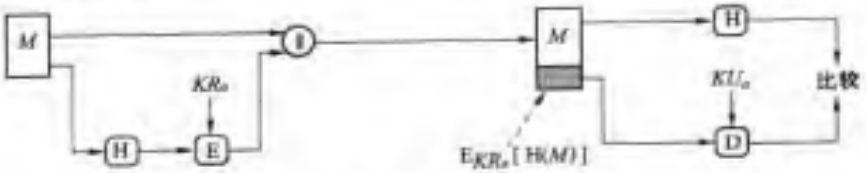
13.3　数字签名标准

美国国家标准与技术研究所(NIST)发布的联邦信息处理标准 FIPS 186,称为数字签名标准(DSS)。DSS 使用第 12 章中讨论的安全 hash 算法(SHA),给出了一种新的数字签名方法,即数字签名算法(DSA)。DSS 最初提出于 1991 年,1993 年根据公众对于其安全性的反馈意见进行了一些修改。1996 年又稍做修改。2000 年发布了该标准的扩充版,即 FIP 186-2。该最新版本还包括基于 RSA 和椭圆曲线密码的数字签名算法。本节我们讨论最初的 DSS 算法。

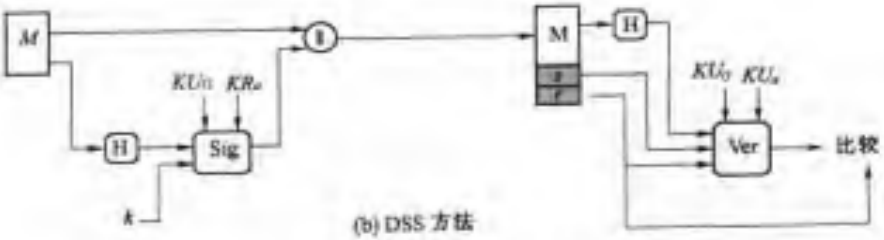
13.3.1　DSS 方法

DSS 使用的是只提供数字签名功能的算法。与 RSA 不同,DSS 是一种公钥方法,但不能用于加密或密钥分配。

图 13.1 对用 DDS 产生数字签名和用 RSA 产生数字签名这两种方法进行了对比。在 RSA 方法中,hash 函数的输入是要签名的消息,输出是定长的 hash 码,用发送方的私钥对该 hash 码加密形成签名,然后发送消息及其签名。接收方用发送方的公钥对签名解密,如果计算出的 hash 码与解密出的结果相同,则认为签名是有效的。因为只有发送方拥有私钥,因此只有发送方能够产生有效的签名。



(a) RSA 方法



(b) DSS 方法

图 13.1　两种数字签名方法

DSS 方法也使用 hash 函数,它产生的 hash 码和为此次签名而产生的随机数 k 作为签名函数的输入,签名函数依赖于发送方的私钥(KR_a)和一组参数,这些参数为一组通信伙伴所共有,我们可以认为这组参数构成全局公钥(KU_G)[①]。签名由两部分组成,标记为 s 和 r。

接收方对接收到的消息产生 hash 码,这个 hash 码和签名一起作为验证函数的输入,验证

① 对不同的用户,这些附加参数可以不同,以便使它们是用户公钥的一部分。在实际应用中,更可能的情形是将全局公钥与每个用户的公钥分开使用。

函数依赖于全局公钥和发送方公钥，若验证函数的输出等于签名中的 r 成分，则签名是有效的。签名函数保证只有拥有私钥的发送方才能产生有效签名。

下面我们详细讨论数字签名算法。

13.3.2 数字签名算法

DSA 建立在求离散对数的困难性（见第 8 章）以及 ElGamal[ELGA85]和 Schnorr[SCHN91]最初提出的方法之上。

图 13.2 归纳总结了 DSA 算法，其中有三个公开参数为一组用户所共有。选择一个 160 位的素数 q；然后选择一个长度在 512 到 1024 之间且满足 q 能整除 $(p-1)$ 的素数 p；最后选择形为的 $h^{(p-1)/q} \bmod p$ 的 g，其中 h 是 1 到 $p-1$ 之间的整数且 g 大于 1[①]。

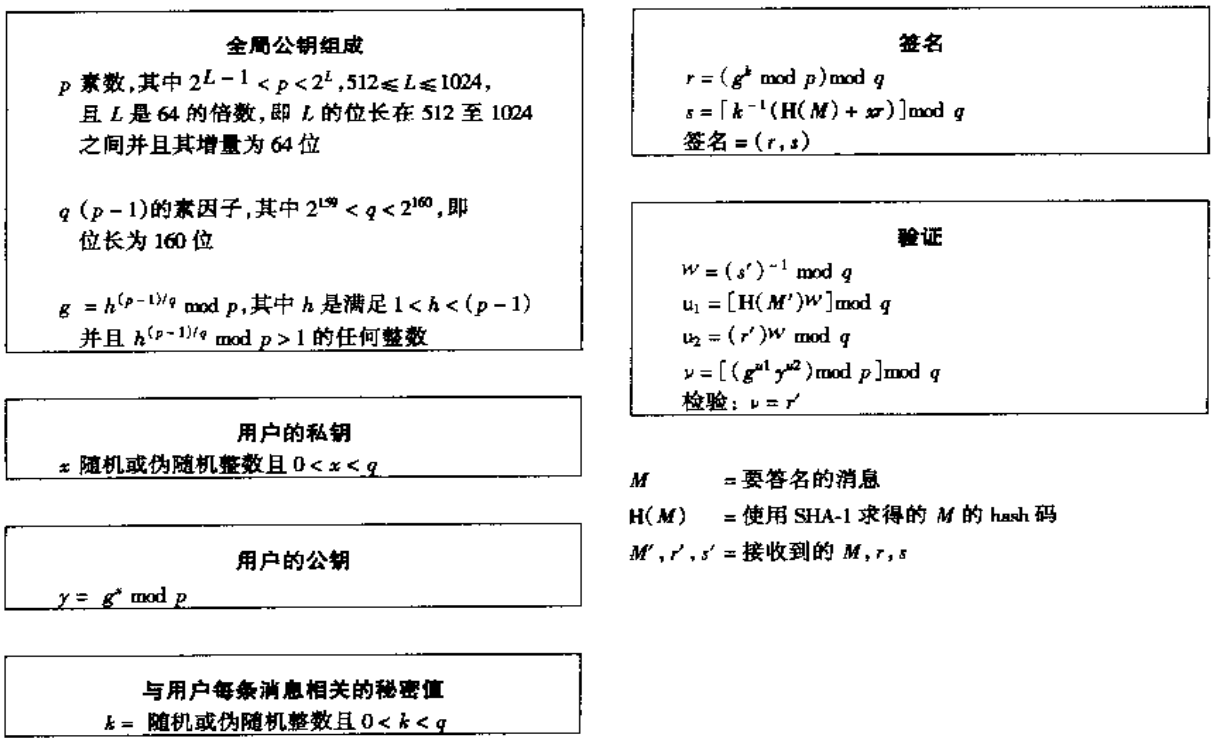


图 13.2 数字签名算法（DSA）

选定这些参数后，每个用户选择私钥并产生公钥。私钥 x 必须是随机或伪随机选择的 1 到 $q-1$ 之间的数，由 $y = g^x \bmod p$ 计算出公钥。由给定的 x 计算 y 比较简单，而由给定的 y 确定 x 则在计算上是不可行的，因为这就是求 y 的以 g 为底的模 p 的离散对数（见第 8 章）。

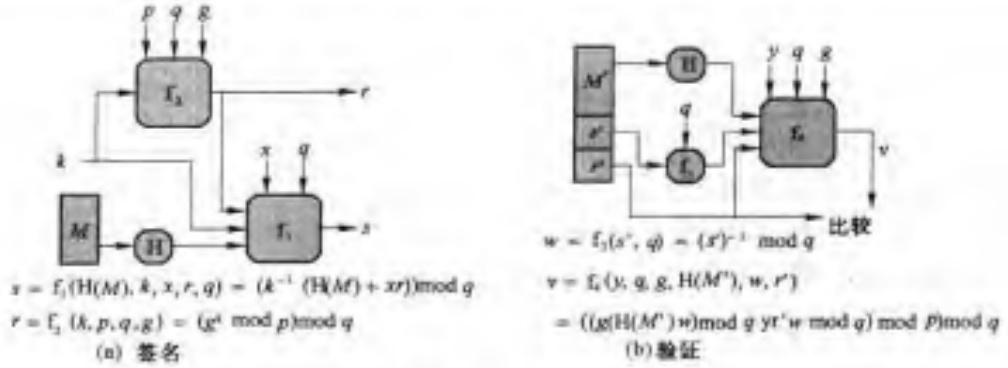
要进行签名，用户需计算两个量 r 和 s。r 和 s 是公钥 (p, q, g)、用户私钥 (x)、消息的 hash 码 $H(M)$ 和附加整数 k 的函数，其中 k 是随机或伪随机产生的，且 k 对每次签名是惟一的。

接收端用图 13.2 所示的公式进行验证。接收方计算值 v，它是公钥 (p, q, g)、发送方公钥、接收到的消息的 hash 码的函数，若 v 与签名中的 r 相同，则签名是有效的。

图 13.3 描述了上述签名和验证函数。

---

① 在数论术语中，g 是 $q \bmod p$ 的阶，见第 8 章。



$s = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) 签名

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{H(M')w} y^{r'w} \bmod q) \bmod p) \bmod q$

(b) 验证

图 13.3　DSS 签名和验证

图 13.3 所示的算法有这样一个特点，接收端的验证依赖于 r，但是 r 却根本不依赖于消息，它是 k 和全局公钥的函数。k 模 p 的乘法逆元传给函数 f_1，f_1 的输入还包含消息的 hash 码和用户私钥。函数的这种结构使接收方可利用其收到的消息和签名、它的公钥以及全局公钥来恢复 r。从图 13.2 和图 13.3 不容易看出这种方法的正确性，本书的网站中给出了证明。

由于求离散对数的困难性，攻击者从 r 恢复出 k 或从 s 恢复出 x 都是不可行的。

另一点需要注意的是，产生签名中需要进行的复杂指数运算 $g^k \bmod p$，但由于它不依赖于被签名的消息，因此可以预先计算。实际上，用户甚至可以根据需要预先计算许多个用于签名的 r。惟一负责的是确定乘法逆元 k^{-1}。需再次提请注意的是，这些值中有许多是可以预先计算的。

13.4　推荐读物

[AKL83]是关于数字签名的经典论文，并且引用率仍然很高。[MITC92]是关于数字签名的优秀综述文章。

AKL83　Akl, S. "Digital Signatures: A Tutorial Survey." *Computer*, February, 1983.

MITC92　Mitchell, C.; Piper, F.; and Wild, P. "Digital Signatures." In [SIMM92a].

13.5　关键术语、思考题和习题

13.5.1　关键术语

仲裁者	仲裁数字签名	直接数字签名
数字签名	数字签名算法(DSA)	数字签名标准(DSS)
相互认证	临时交互号	单向认证
重放攻击	禁止－重放攻击	时间戳

13.5.2 思考题

13.1 列出消息认证中出现的两种争议。
13.2 数字签名应该具有哪些性质?
13.3 数字签名应满足哪些要求?
13.4 直接数字签名和仲裁数字签名的区别是什么?
13.5 签名函数和保密函数应以何种顺序作用于消息? 为什么?
13.6 直接数字签名方法中会遇到哪些威胁?
13.7 举例说明重放攻击。
13.8 列出对付重放攻击的三种通用办法。
13.9 什么是禁止 - 重放攻击?

13.5.3 习题

13.1 在 13.2 节中,我们简要介绍了[WOO92a]中提出的用于密钥分配的公钥方法,其修改协议的步骤 5 和步骤 6 中包含 ID_A,这种修改主要是为了对付何种攻击?

13.2 习题 13.1 中的协议可由 7 步归约为 5 步,它包含下述序列:

1. A→B:
2. B→KDC:
3. KDC→B:
4. B→A:
5. A→B:

请给出每步中传输的消息。提示:该协议的最后一条消息与原协议的最后一条消息相同。

13.3 修改表 13.1(a)和表 13.1(b)中的数字签名方法,使得接收方可以验证签名。

13.4 修改表 13.1(c)中的数字签名方法,以避免对整条消息进行三次加密。

13.5 讨论表 13.1(c)时,曾讲过不可能存在联合抵赖,但事实上这种抵赖是可能存在的。请说明这种可能性,并解释为什么这种可能性很小以至于我们可以忽略它而不影响安全性。

13.6 13.2 节中讨论了禁止 - 重放攻击。

a. 当通信一方的时钟超前于 KDC 的时钟时,举例说明这种攻击。
b. 当通信一方的时钟超前于另一方的时钟时,举例说明这种攻击。

13.7 将随机数作为临时交互号有三种典型的方法。假定 N_a 是 A 产生的临时交互号,A 和 B 共享密钥 K,f()是函数,如增值函数。这三种使用方法如下:

方法 1	方法 2	方法 3
(1) A → B: N_a (2) B → A: $E_K[N_a]$	(1) A → B: $E_K[N_a]$ (2) B → A: N_a	(1) A → B: $E_K[N_a]$ (2) B → A: $E_K[f(N_a)]$

试说明每种方法适用的情况。

13.8 华生耐心地等待福尔摩斯，直至福尔摩斯退出系统后才问："有什么有趣的问题需要解决吗？"

"噢，没有，我只是看一看邮件，并做了几个网络实验，而不是通常的化学实验。我现在只有一个客户，并且我已经解决了他的问题。我清楚地记得，你曾说过你也喜欢进行密码研究，所以你也许会感兴趣的。"

"我只是业余的密码学家，福尔摩斯，但我想我会感兴趣的。你说的是什么问题呢？"

"我的客户是 Hosgrave 先生，他是一家小银行的董事，他的银行已完全计算机化，当然也广泛地使用网络通信，他们使用 RSA 来保护数据，并对传输的文件进行数字签名。现在，他们打算修改一些过程，特别地，对某些文件需要两个签名者来签名，使得：

1. 第一签名者对文件签名，并传送给第二签名者。
2. 第二签名者首先验证该文件确实已由第一签名者签名，然后他将其签名加到文件中，任何接收者都可验证两次签名后的文件，但只有第二签名者可验证步骤 1 中的签名。而且，银行希望利用现有支持 RSA 数字签名的模块。"

"嗯，我知道如何将 RSA 用于只有一个签名者的数字签名中，福尔摩斯。我猜你已经适当推广了 RSA 数字签名并解决了 Hosgrave 先生的问题。"

"是的，华生。"福尔摩斯点头道，"RSA 数字签名是用签名者的私钥(d)加密产生的，任何人都可用公钥(e)解密来验证签名，验证签名 S 是由已知 d 的惟一的签名者产生的。对这个过程稍加推广，就可解决 Hosgrave 先生的问题，也就是说……"

13.9 DSA 中指出，如果签名产生过程中出现 $s=0$，则必须产生新的 k 并重新计算签名，为什么？

13.10 如果用于产生 DSA 签名的 k 已被泄密，则会出现什么问题？

13.11 DSS 包括一个推荐的素性测试算法，该算法如下：

(1) [**选择 w**] 令 w 是随机的奇数，则 $(w-1)$ 是偶数且可表示为 $2^a m$，其中 m 是奇数，也就是说，2^a 是整除 $(w-1)$ 的 2 的最大幂。

(2) [**产生 b**] 令 b 是随机整数，$1<b<w$。

(3) [**求幂**] 置 $j=0$ 且 $z=b^m \bmod w$。

(4) [**完成?**] 若 $j=0$ 且 $z=1$ 或者 $z=w-1$，则 w 可能是素数，故应测试 w，转到步骤 8。

(5) [**终止?**] 若 $j>0$ 且 $z=1$，则 w 不是素数，对该 w 算法终止。

(6) [**增值 j**] 置 $j=j+1$。若 $j<a$，则置 $z=z^2 \bmod w$ 并转到步骤 4。

(7) [**终止**] w 不是素数，对该 w 算法终止。

(8) [**继续测试?**] 若已测试足够多的 b，则认为该 w 是素数并终止算法，否则转到步骤 2。

a. 说明该算法的工作原理。

b. 证明该算法与第 7 章中给出的 Miller-Rabin 测试是等价的。

13.12 因为 DSS 对每个签名产生一个 k，所以即使对同一消息签名，在不同的情况下签名也不相同，但 RSA 签名则不能做到这一点。这种区别有什么实际意义？

13.13 在 Diffie-Hellman 算法的基础上，设计可用于数字签名的方法是很有意义的。下面的方法比 DSA 更简单，它需要私钥但不需要秘密的随机数。

公开量：

q 素数

α $\alpha < q$ 且 α 是 q 的本原根

私钥：

X $X < q$

公钥：

$$Y = \alpha^X \bmod q$$

要对消息 M 签名，则先计算该消息的 hash 码 $h = \mathrm{H}(M)$。我们要求 $\gcd(h, q-1) = 1$，若 $\gcd(h, q-1)$ 不为 1，则将该 hash 码附于消息后再计算 hash 码，继续该过程直至产生的 hash 码与 $(q-1)$ 互素；然后计算满足 $Z \times h \equiv X \bmod (q-1)$ 的 Z，并将 α^Z 作为对该消息的签名。验证签名即是验证 $Y = (\alpha^Z)^h = \alpha^X \bmod q$。

a. 证明该体制能正确运行，即证明如果签名是有效的，那么在验证过程中将有上述等式成立。

b. 给出一种对给定的消息伪造用户签名的简单方法，以证明这种体制是不可接受的。

第三部分　网络安全应用

第三部分涉及的内容

在前两部分中，我们探讨了各种密码及其在保密性、认证、密钥交换等方面的应用。第三部分探讨重要的网络安全工具和应用。这些应和可用在单个网络中、企业局域网中或互联网中。

第三部分内容浏览

第 14 章　认证的实际应用

第 14 章将介绍目前正在使用的两个最重要的认证规范。Kerberos 是一个基于传统加密的认证协议，它受到了广泛的支持，并用于各种系统中。X.509 提出了一种认证算法，并定义了一种认证方式。后者可确保用户获得公钥认证，以便用户团体可确信公钥的有效性。这种方法已在许多应用中采用为构件分组。

第 15 章　电子邮件安全

使用最广泛的分布式应用是电子邮件，将认证和保密性作为电子邮件方式的一部分，已越来越受到人们的关注。第 15 章将研究两个近期支配电子邮件安全的两种方法。Pretty Good Privacy(PGP)是一种广泛使用的方案，它不依赖于任何组织或授权。因此，它同样适用于个人。S/MIME 已开发为一种互联网标准。

第 16 章　IP 安全性

互联网协议(IP)是 Internet 和私用内部网的核心元素。相应地，IP 层的的安全对任何其于内联网的安全方案的设计是很重要的。第 16 章将探讨 IP 安全方案，它是为当前的 IP 以及正在出现的下一代 IP(即 IPv6)开发的。

第 17 章　Web 安全性

WWW 在电子商务应用方面的爆炸式增长，导致了人们对基于 Web 的安全性的强劲需求。第 17 章探讨了这个重要的新安全领域，并研究了两种密钥标准，即安全套接字层(SSL)和安全电子事务处理(SET)。

第 14 章　认证的实际应用

本章介绍一些用于应用级认证和数字签名的认证功能。首先介绍最早且被最广泛使用的服务之一 Kerberos，然后介绍 X.509 目录认证服务，该标准是目录认证的一个重要组成部分，而且它还被应用于其他标准，如第 15 章中介绍的 S/MIME。

14.1　Kerberos

Kerberos[①] 是作为 MIT 的 Athena 计划的认证服务而开发的。Kerberos 阐述了这样一个问题：假设有一个开放的分布环境，工作站用户想通过网络对分布在网络中的各种服务提出请求，那么，希望服务器能够只对授权用户提供服务，并能鉴别服务请求的种类。但在这种环境下，一个工作站无法准确判定它的终端用户和请求的服务是否合法。特别是存在以下三种威胁：

- 用户可能通过某种途径进入工作站并假装成其他用户操作工作站。
- 用户可以通过变更工作站的网络地址，从该机上发送伪装的请求。
- 用户可以监听信息交换并使用重播攻击，以获得服务或破坏正常操作。

在上述任何一种情况下，一个非授权用户均可能获得未授权的服务或数据。针对上述情况，Kerberos 通过提供一个集中的认证服务器来负责用户对服务器的认证和服务器对用户的认证，而不是为每个服务器提供详细的认证协议。与本书中其他认证方案不同的是，Kerberos 仅仅依赖于对称加密体制而没有使用公钥加密体制。

目前常用的 Kerberos 有两个版本。版本 4[MILL88, STEI88]被广泛使用，而版本 5 [KOHL94] 改进了版本 4 中的安全性，并成为 Internet 标准草案(RFC 1510)[②]。

我们首先简要介绍 Kerberos 的动机，然后通过版本 4 中的认证协议来理解 Kerberos 策略的本质，最后阐述版本 5。

14.1.1　动机

当一组用户使用相互独立的计算机而不用网络相连时，每个用户的资源和文件就可以利用物理手段保护；当一组用户使用一个集中式分时系统时，则必须由分时操作系统来提供安全保护，操作系统可以使用基于用户身份的操作控制策略，并通过登录过程来认证用户。

今天，更为普遍的情况是由许多用户工作站和分布(或集中)的服务器所组成的分布式体系结构。在这种环境下，有三种安全方案可以采用：

---

① “在希腊神话中，Kerberos 是地狱入口的守卫者，长着许多头，通常是三个头，带有毒蛇似的尾巴。”摘自“Dictionary of Subjects and Symbols in Art”, James Hall, Harper & Row, 1979。像希腊神话中的 Kerberos 有三个头一样，现代的 Kerberos 也要有三个“头”来守卫网络的大门，那就是认证、清算和审计。现在，后面的两个“头”还未实现。

② 第 1 版到第 3 版是内部开发版本，第 4 版才是“原始”版本。

1. 由客户端工作站确保用户或用户组的身份，由服务器提供基于用户标识 ID 的安全策略。
2. 要求客户端系统向服务器提供身份认证，但要相信客户端系统对自己用户的身份认证。
3. 不仅要求客户向服务器提供身份认证，同时需要服务器向客户提供身份认证。

在一个小型、封闭的环境中，所有的系统属于同一个组织且被其访问，则应选择第一方案或第二方案。[①] 但对更为开放的网络互联环境而言，则应选择第三种方案保护用户信息和服务器资源。Kerberos 支持第三方案。Kerberos 假设其体系结构为分布的客户/服务器体系结构，并拥有一个或多个 Kerberos 服务器提供认证服务。

Kerberos 发布的第一个报告[STEI88]中列出了以下 Kerberos 的需求：

- **安全性**：网络监听者不可能通过冒充其他用户获取有用信息。通常，Kerberos 应有足够的坚固性，使得潜在攻击者无法找到其中的薄弱环节。
- **可靠性**：对依赖于 Kerberos 访问控制的所有服务而言，Kerberos 服务缺乏可用性意味着其所支持的服务缺乏可用性。因此，Kerberos 必须具有高可靠性，且应使用分布式服务结构，即一个系统可以支持其他系统。
- **透明性**：理想状况下，用户除了要输入口令外，不需要知道认证的发生。
- **可伸缩性**：系统应能支持大量的客户端和服务器，这需要采用模块化、分布式的体系结构。

为了支撑这些要求，Kerberos 的整体设计方案是基于协议的可信第三方认证服务（在第 7 章中讨论过），客户与服务器均信任 Kerberos 的认证。假设 Kerberos 协议设计充分，则认证服务的安全性取决于 Kerberos 服务器的安全性。[②]

14.1.2　Kerberos 版本 4

Kerberos 的版本 4 在协议中使用 DES 来提供认证服务，但从整体上很难看出使用 DES 的必要性。因此，我们使用了 Athena 计划中被 Bill Bryant[BRYA88]用过的策略，通过建立一系列假设会话来帮助理解该协议，后一个会话在前一个会话的基础上增加了一些安全性措施，以解决暴露的安全漏洞。

一个简单的认证会话

在一个无保护的网络中，任何客户端可以向任意服务器提出服务请求。显然，安全风险在于可能假冒，一个攻击者可以假装成其他客户获得未授权的服务。因此，服务器必须确定申请服务的客户身份。每次服务器必须在客户/服务器的每次交互中进行认证。但在一个开放式的环境中，这将对服务器带来许多额外的负担。

一种变通的方法是使用一个认证服务器（AS）将所有用户的口令存储在一个集中式数据

---

① 然而，即使是封闭式环境也面临怀有不满情绪的雇员的攻击。

② 要记住，Kerberos 服务器并不能自动具有安全性，而是要采取措施确保其安全（如安装到上锁的房间里）。还要记住希腊神话中 Kerberos 的命运。Eurystheus 命令大力神捕捉 Kerberos，以作为他的第 12 个劳奴："大力神发现那只戴着链子的大狗，就卡住它的脖子。那时 Kerberos 的三个头企图反攻，还用它那强有力的尾巴来回鞭打。大力神冷酷地坚持不懈，最终 Kerberos 失去知觉，被制伏。Eurystheus 看见 Kerberos 的三个头流着口水，惊讶地发现大力神还活着。已经属于他的那条大狗惧怕他的智慧，就跳回到他那安全的铜缸里。"摘自 Michael Stapleton，Hamlyn，1982 年版的"The Hamlyn Concise Dictionary of Greek and Roman Mythology"一书。

库中，并且 AS 与每一个服务器共享惟一的一个密钥，密钥按物理方法分发或按其他安全方法分发。考虑以下包含 3 个消息的会话：①

$$\text{(1) C} \rightarrow \text{AS:} \quad ID_C \| P_C \| ID_V$$
$$\text{(2) AS} \rightarrow \text{C:} \quad Ticket$$
$$\text{(3) C} \rightarrow \text{V:} \quad ID_C \| Ticket$$
$$Ticket = E_{K_V}[ID_C \| AD_C \| ID_V]$$

其中，

C = 客户端
AS = 认证服务器
V = 应用服务器
ID_C = C 上的用户标识
ID_V = V 的标识
P_C = C 上的用户口令
AD_C = C 的网络地址
K_V = AS 与 V 共享的加密密钥
$\|$ = 连接符

此时，用户登录到工作站，并请求访问服务器 V，用户工作站上的客户端模块 C 要求用户输入口令，并将包含用户标识 ID、服务器标识 ID 和用户口令的消息送往 AS。AS 查询数据库，检查用户口令是否与用户标识相符，并判断此用户是否有访问服务器 V 的权限。如果两个检测均通过，则 AS 认为此用户合法，并通知服务器该用户是合法的。为了达到这一目的，AS 创建一个包含该用户 ID、用户网络地址和服务器 ID 的票据(Ticket)，用 AS 和此服务器共享的密钥加密，并将加密后的票据返回 C。由于票据被加密，因此不可能被 C 或其他攻击者修改。

使用该票据，C 可以向 V 提出服务请求。C 向 V 发出包含 C 的 ID 和票据的消息，由 V 解密票据，并验证票据里的用户 ID 是否与消息中未加密的用户 ID 一致。如果验证通过，则服务器认为该用户真实，并为其提供服务。

消息(3)的每一个成分都非常重要。票据加密后可防止更改或伪造。将服务器标识(ID_V)包含在票据中，使得服务器可以验证它是否能正确地解密该票据；包含在票据中的标识 ID_C 可以说明该票据是从 C 发布的；最后，用 AD_C 来对抗下述威胁。如果票据中没有网络地址，攻击者可能在消息(2)的传输过程中捕获该票据，然后使用标识 ID_C 以消息(3)的格式从另一个工作站上发送消息，这样，服务器就可以得到一个与该用户 ID 相匹配的合法票据，将相应的权利授给其他工作站。为了防止这种攻击，AS 可以在票据中加入消息来源的网络地址。这时，从票据生成的工作站发出的票据才是合法的。

一个更安全的认证会话

虽然上述会话解决了开放网络环境中认证的一些问题，但其中还存在两个突出问题。首

① 冒号左边的部分表示发送者和接收者，冒号右边的部分表示消息内容，符号 $\|$ 表示连接。

先，我们可能希望能使用户输入口令的时间最小化。假设每个票据仅能使用一次，那么，如果用户 C 早晨在一个工作站上登录，希望查看他在邮件服务器上的邮件，C 为了与邮件服务器通信就必须提供口令得到票据。但如果 C 想在一天中多次查看邮件，则每一次都需要它重新输入口令。我们可以通过重用票据来解决这一问题。对一个简单的登录会话，工作站可以将它收到的邮件服务器的票据存储，用户可以在多次访问邮件服务器时使用相同的票据。

但在这种模式下，用户必须为每种不同的服务创建一个新的票据。如果一个用户想同时访问打印服务器、邮件服务器、文件服务器等，则每次访问必须输入用户口令，以获得新的票据。

第二个问题是，上述会话中包含对口令[消息(1)]的明文传输，网络窃听者捕获口令后可以使用受害者的任何服务。

为了解决这些问题，我们引进了一个避免口令明文传输的模式和一个票据授权服务器(TGS)。新的会话如下：

每个用户仅一次的登录会话：

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E_{K_c}[Ticket_{tgs}]$

每种服务类型仅一次：

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

每次服务会话仅一次：

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E_{K_{tgs}}[ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1]$$

$$Ticket_v = E_{K_v}[ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2]$$

TGS 将票据发给通过 AS 认证的用户，于是，用户首先应向 AS 申请得到一个票据授权票据 $Ticket_{tgs}$，由用户工作站的客户端模块保存。每当用户申请一项新的服务，客户端则用该票据证明自己的身份，向 TGS 发出申请，由 TGS 向特定的服务授予一个票据，客户将每个服务授权票据保存后，在每次请求特定服务时使用该票据证实自己的身份：

1. 客户端将用户标识、TGS 标识一起送往 AS，申请得到票据授权票据。
2. AS 用一个从用户口令得到的密钥加密票据，并将其返回给客户端，由客户端提示用户输入口令，生成密钥，解密传来的消息，如果口令正确；则可以成功地恢复票据。

由于只有正确的用户知道口令，只有该用户能恢复票据。因此，可以在 Kerberos 中使用口令来获得可信度，避免口令的明文传输。票据本身由用户标识、用户网络地址和 TGS 标识组成，使得票据可以在客户端申请服务授权票据时多次使用，因此票据授权票据是可以重用的。考虑以下情况：一个攻击者捕获票据后等待，直至原用户退出网络。此时，攻击者可以通过使用原用户工作站或将其自己的工作站配置为同样的网络地址来使用该票据欺骗 TGS。为防止这种情况发生，颁发的票据中应包含带有日期和时间的时间戳，并包含票据合法使用时间长度(如 8 小时)的生命期(Life time)，这样，客户端既可以重用票据，又可以不用为每个新的服务保存一个口令。最后，票据授权票据是被一个仅有 AS 和 TGS 知道的密钥加密的，防止了对票据

授权票据的篡改，且该票据被用用户口令的密钥再加密，确保了只有正确的用户才能恢复该票据，起到了认证的作用。

现在，客户端得到了票据授权票据，通过步骤3、步骤4来对各种服务器进行访问：

3. 客户端根据用户请求申请一个服务授权票据。为此，客户端将包含由用户标识ID、服务标识ID和票据授权票据组成的消息送往TGS。
4. TGS对得到的票据授权票据解密，通过其标识验证该票据的正确性。确定该票据的生命期未超时，并比较用户ID和网络地址是否用消息来源一致来认证用户。如果用户被允许访问服务器V，则TGS发布相应服务的服务授权票据。

服务授权票据与票据授权票据具有相同的结构。实际上，由于TGS是服务器，我们可以认为TGS认证客户端与认证应用服务器所需要的元素是相同的。其票据也应包含时间戳和生命周期。如果用户想在后来再次使用同一个服务，该客户端可以简单地使用原来获得的服务授权票据，而不需要用户重新输入口令。注意，加密票据授权票据的密钥 K_c 只有TGS和特定服务器知道，可防止篡改。

最后，使用特定的服务授权票据，客户端即可通过步骤5请求相应的服务：

5. 客户端请求服务。为此，客户端将包含用户标识和服务授权票据的消息送往服务器。服务器通过该票据的内容认证用户的合法性。

这种新的模式可以满足在每次用户会话时仅询问一次口令和保护用户口令的需求。

版本4的认证会话

虽然上述会话与第一种方式相比在安全性方面有所增强，但仍然存在两个问题。第一个问题与票据授权票据的生命期有关。如果生命期太短(例如几分钟)，则用户将被要求重复输入口令；而如果生命期太长(如几小时)，则为攻击者提供了大量攻击机会。攻击者在网络上监听、捕获票据授权票据后等待合法用户退出网络。此时，攻击者可以伪造合法用户的网络地址，并按照步骤(3)发送消息给TGS，因此将会使攻击者得到合法使用该用户资源和文件的机会。

同样地，如果攻击者获取了服务授权票据并在其过期之前使用，也可获得相应的服务。

因此，我们提出了一个额外的需求。一个网络服务(TGS或应用服务)必须能够证实票据使用者与票据所有者的一致性。

第二个问题是服务器有向用户证实自己身份的需求。没有这样的认证，攻击者即可伪造配置使得送往服务器的消息被定向到其他节点。假冒的服务器即可假装成真正的服务器来捕获用户信息，从而对用户提供虚假服务。

我们将逐个讨论这些问题，表14.1描述了实际的Kerberos协议。

首先考虑票据授权票据的捕获问题和确定票据使用者与票据所有者一致的需求。其威胁是攻击者窃取票据并在其有效期间内使用。为了解决这一问题，我们让AS为客户端和TGS同时用安全方式提供一条秘密信息，然后客户端用该秘密信息以同样的安全方式向TGS证实自己的身份。Kerberos的会话密钥使用的是加密密钥。

表 14.1　Kerberos 版本 4 的消息交换

(a) 服务认证交换：获得票据授权票据

(1) $C \rightarrow AS$: $ID_c \| ID_{tgs} \| TS_1$

(2) $AS \rightarrow C$: $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]$$

(b) 服务授权票据交换：获取服务授权票据

(3) $C \rightarrow TGS$: $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C$: $E_{K_{c,tgs}}[K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]$$
$$Ticket_v = E_{K_v}[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4]$$
$$Authenticator_c = E_{K_{c,tgs}}[ID_C \| AD_C \| TS_3]$$

(c) 客户/服务器认证交换：获取服务

(5) $C \rightarrow V$: $Ticket_v \| Authenticator_c$

(6) $V \rightarrow C$: $E_{K_{c,v}}[TS_5 + 1]$　　(相互认证)

$$Ticket_v = E_{K_v}[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4]$$
$$Authenticator_c = E_{K_{c,v}}[ID_C \| AD_C \| TS_5]$$

表 14.1(a)描述了分发会话密钥的技术。如前所述，客户端向 AS 发送消息请求访问 TGS，由 AS 回送应答消息，并用从用户口令派生的密钥(K_c)将其加密。加密的消息中还含有一份会话密钥 $K_{c,tgs}$，其下标表示该密钥是属于 C 和 TGS 共享的会话密钥。由于会话密钥包含在用 K_c 加密的消息中，只有该用户的客户端可以解密。此会话密钥同样也存在于票据中，而只能被 TGS 解密。因此，此会话密钥可以安全地在 C 和 TGS 间传递。

在继续进行会话前还有一些附带的信息要加到会话的第一阶段中。由于消息(1)包含时间戳，使得 AS 知道该消息是及时的；消息(2)包含 C 可以访问票据中的若干元素，这些元素使得 C 能确定此票据是送往 TGS 的，并知道它的有效期。

有了票据和会话密钥，C 就可以与 TGS 通话。同样，由 C 向 TGS 发送消息，消息中包含票据和所申请服务的标识 ID，如表 14.1(b)中的消息(3)。另外，C 还发送一条认证消息，包括 C 的用户标识 ID、网络地址和时间戳。与票据的可重用性不同，此认证消息仅能使用一次且生命期极短。当 TGS 接到消息后，用与 AS 共享的密钥解密该票据，票据中包含的信息说明用户 C 已得到会话密钥 $K_{c,tgs}$，即相当于宣布“任何使用 $K_{c,tgs}$ 的用户必为 C”。接着，TGS 使用该会话密钥解密认证消息，用得到的信息检查消息来源的网络地址，如匹配，则 TGS 确认该票据的发送者与票据的所有者是一致的。实际上，认证消息说明“在时间 TS_3，AD_c 使用 $K_{c,tgs}$”。注意，票据并不能证明任何人的身份，而只是安全分发密钥的一种形式；而认证消息则用于证明用户的身份。由于认证消息仅能使用一次且生命期极短，攻击者同时窃取票据和认证消息的威胁将在后面论述。

TGS 的应答消息为消息(4)，与消息(2)的格式相同。该消息用 TGS 和 C 的共享会话密钥加密，且包含 C 与服务器 V 的共享密钥、服务器 V 的标识 ID 以及票据的时间戳。票据自身包含相同的会话密钥。

现在，C 拥有了服务器 V 的一个可重用的服务授权票据。当 C 按消息(5)的方式使用票据时，同样需要一个认证消息。由服务器解密票据，恢复会话密钥，并解密认证消息。

如果需要双向认证，则服务器应按表 14.1 中的消息(6)发送一个应答消息。服务器返回的消息中，时间戳的值为认证消息时间戳的值加 1，并用会话密钥加密。C 解密消息后可得到增加后的时间戳，由于消息是被会话密钥加密的，C 可以确信此消息只可能由服务器 V 生成。消息中的内容确保该应答不是一个对以前消息的应答。

最后，客户端与服务器共享一个密钥，该密钥可以用于加密在它们之间传递的消息，或交换新的随机会话密钥。

表 14.2 总结了 Kerberos 协议各元素的合理性，图 14.1 简要概括了整个会话的过程。

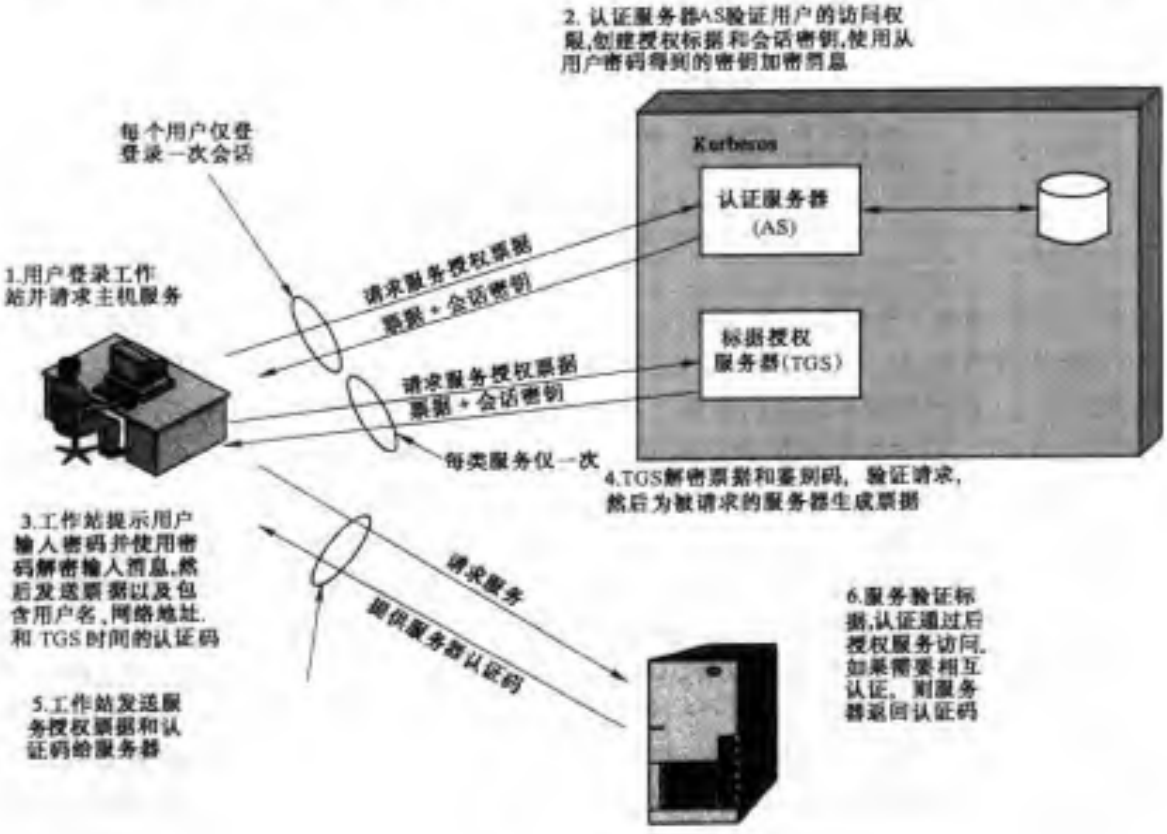


图 14.1 Kerberos 概述

Kerberos域和多重Kerberi

Kerberos环境包括Kerberos服务器、若干客户端和若干应用服务器：

1. Kerberos服务器必须有存放用户标识(UID)和用户口令的数据库。所有用户必须在Kerberos服务器注册。
2. Kerberos服务器必须与每个应用服务器共享一个特定的密钥，所用应用服务器必须在Kerberos服务器注册。

表 14.2　Kerberos 版本 4 协议中所用到元素的作用

	(a) 认证服务交换
消息(1)	客户端申请票据授权票据
ID_C：	告知 AS 客户端的用户标识
ID_{tgs}：	告知 AS 用户请求访问 TGS
TS_1：	使 AS 能验证客户端时钟是否与 AS 时钟同步
消息(2)	AS 返回票据授权票据
E_{K_c}：	基于用户口令的密钥使得 AS 与客户端能验证口令，保护消息(2)的内容
$K_{c,tgs}$：	客户端可访问的会话密钥，由 AS 创建，使得客户端和 TGS 在不需要共享永久密钥的前提下安全交换信息
ID_{tgs}：	标识该票据是为 TGS 生成的
TS_2：	通知客户端票据发放的时间戳
$Lifetime_2$：	通知客户端票据的生命期
$Ticket_{tgs}$：	客户端用于访问 TGS 的票据
	(b) 服务授权票据交换
消息(3)	客户端申请服务授权票据
ID_V：	告知 TGS 用户希望访问的服务器 V
$Ticket_{tgs}$：	告知 TGS 该用户已被 AS 认证
$Authenticator_c$：	客户端生成的合法票据
消息(4)	TGS 返回服务授权票据
$E_{K_{c,tgs}}$：	用 C 与 TGS 共享的密钥保护消息(4)的内容
$K_{c,v}$：	客户端可访问的会话密钥，由 TGS 创建，使得客户端和服务器在不需要共享永久密钥的前提下安全交换信息
ID_V：	标识该票据是为服务器 V 生成的
TS_4：	通知客户端票据发放的时间戳
$Ticket_V$：	客户端用于访问服务器 V 的票据
$Ticket_{tgs}$：	重用，以免用户重新输入口令
$E_{K_{tgs}}$：	由 AS 和 TGS 共享的密钥加密的票据，防止伪造
$K_{c,tgs}$：	TGS 可访问的会话密钥，用于解密认证消息，即认证票据
ID_C：	标识票据的合法所有者
AD_C：	防止票据在与申请票据时的不同工作站上使用
ID_{tgs}：	向服务器确保票据解密正确
TS_2：	通知 TGS 票据发放的时间
$Lifetime_2$：	防止票据过期后继续使用
$Authenticator_c$：	向 TGS 确保此票据的所有者与票据发放时的所有者相同，用短生命期防止重用
$E_{K_{c,tgs}}$：	用客户端与 TGS 共享的密钥加密认证消息，防止伪造
ID_c：	票据中必须与认证消息匹配的标识 ID
AD_c：	票据中必须与认证消息匹配的网络地址
TS_2：	通知 TGS 认证消息的生成时间

（续表）

(c) 客户/服务器认证交换	
消息(5)	客户端申请服务
$Ticket_V$：	向服务器证明该用户通过了 AS 的认证
$Authenticator_c$：	客户端生成的合法票据
消息(6)	可选的客户端认证服务器
$E_{K_{c,v}}$：	向客户端 C 证明该消息来源于服务器 V
TS_5+1：	向客户端 C 证明该应答不是对原来消息的应答
$Ticket_v$：	可重用，使得用户在多次使用同一服务器时不需要向 TGS 申请新票据
E_{K_v}：	用 TGS 与服务器共享的密钥加密的票据，防止仿造
$K_{c,v}$：	客户端可访问的会话密钥，用于解密认证消息
ID_C：	标识票据的合法所有者
AD_C：	防止票据在与申请票据时的不同工作站上使用
ID_V：	确保服务器能正确解密票据
TS_4：	通知服务器票据发放的时间
$Lifetime_4$：	防止票据超时使用
$Authenticator_c$：	向服务器确保此票据的所有者与票据发放时的所有者相同，用短生命期防止重用
$E_{K_{c,v}}$：	用客户端与服务器共享的密钥加密的认证消息，防止假冒
ID_C：	票据中必须与认证消息匹配的标识 ID
AD_C：	票据中必须与认证消息匹配的网络地址
TS_5：	通知服务器认证消息的生成时间

这种环境称为一个域。隶属于不同行政机构的客户/服务器网络通常构成了不同域，在一个Kerberos服务器中注册的客户与服务器属于同一个行政区域，但由于一个域中的用户可能需要访问另一个域中的服务器，而某些服务器也希望能给其他域的用户提供服务，所以也应该为这些用户提供认证。

Kerberos提供了一种支持这种域间认证的机制。为支持域间认证，应满足第三个需求：

3. 每个互操作域的Kerberos服务器应共享一个密钥，双方的Kerberos服务器应相互注册。

这种模式要求一个域的Kerberos服务器必须信任其他域的Kerberos服务器对其用户的认证。另外，其他域的应用服务器也必须信任第一域中的Kerberos服务器。

有了以上规则，我们可以用图 14.2 来描述该机制：当用户访其他域的服务时，须获得其他域中该服务的服务授权票据，用户按照通常的程序与本地 TGS 交互，并申请获得远程 TGS(另一个域的 TGS)的票据授权票据。客户端可以向远程 TGS 申请远程 TGS 域中服务器的服务授权票据。

图 14.2 中交换的消息细节如下(与表 14.1 比较)：

(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C: $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

(3) C → TGS: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $E_{K_{c,tgs}}[K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}]$

(5) C → TGS_{rem}: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(6) TGS_{rem} → C: $E_{K_{c,tgsrem}}[K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}]$

(7) C → V_{rem}: $Ticket_{vrem} \| Authenticator_c$

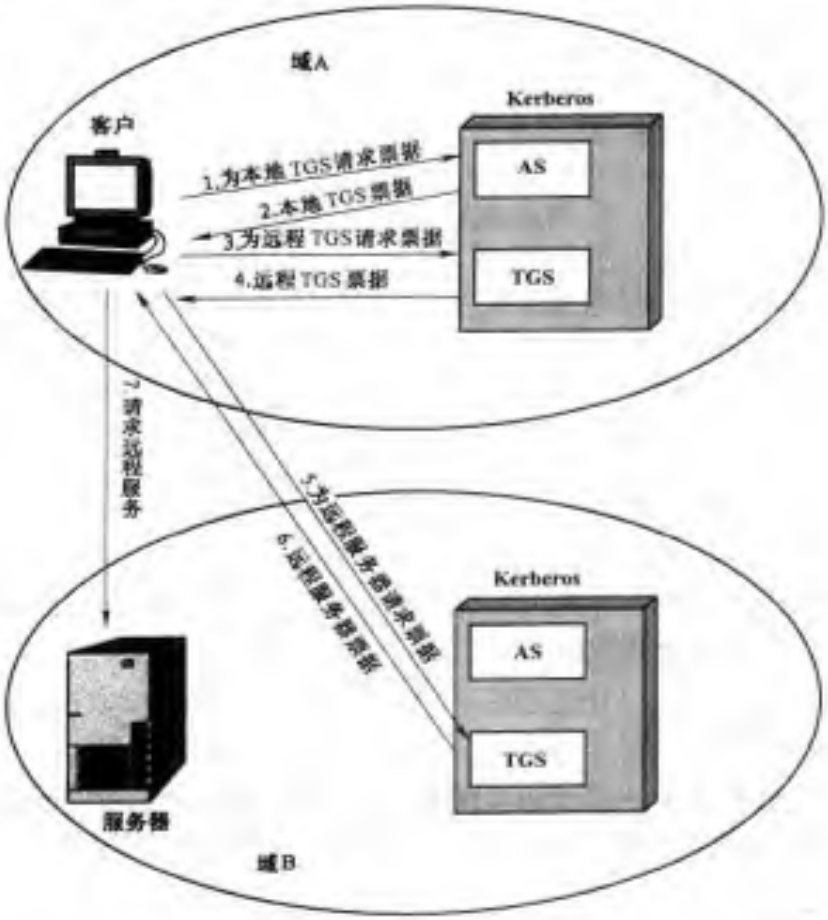


图 14.2　请求另一个域中的服务

送往远程服务器(V_{rem})的票据表明了用户原认证所在的域，服务器可以决定是否接收远程请求。

上述方式带来的一个问题是，对多域的可伸缩性不大。如果有 N 个域，则必须有 $N(N-1)/2$次安全密钥交换，每个 Kerberos 域可以与其他 Kerberos 域交互。

14.1.3　Kerberos 版本 5

Kerberos 版本 5 在 RFC 1510 中有详细描述，介绍了版本 4 的一些改进[KOHL94]。在此，我们将首先介绍版本 4 与版本 5 的不同，接着讨论版本 5 协议。

版本 4 与版本 5 的区别

版本 5 从两个方面阐述了版本 4 的局限性：环境缺陷和技术不足。我们先简单介绍这两个方面的改进。[①]

Kerberos 版本 4 是为在 Athena 项目环境中使用而开发的，而没有过多地考虑一般情况，这就导致了以下一些**环境缺陷**：

1. **加密系统依赖性**：版本 4 使用 DES，因此，它依赖于 DES 的强度和输出限制。版本 5 用加密类型标记密文，使得可以使用任何加密技术。用类型和长度标记的密钥允许同一个密钥在不同算法中使用，并允许在给定算法中具有不同的描述。
2. **Internet 协议依赖性**：版本 4 需要使用 IP 地址，不支持其他地址类型，如 ISO 网络地址。版本 5 用类型和长度标记网络地址，允许使用任何类型的网络地址。
3. **消息字节顺序**：版本 4 中，由消息的发送者用标记说明规定消息的字节顺序，而不遵循已有的惯例。在版本 5 中，所有消息结构按照抽象语法表示(ASN.1)和基本编码规则(BER)规定，确定消息字节顺序。
4. **票据的生命期**：版本 4 的生命期用一个 8 位表示，每单位代表 5 分钟。因此，最大生命期为 $2^8 \times 5 = 1280$ 分钟，约为 21 小时，但这对某些应用不够长。在版本 5 中，票据中包含了精确的起始时间和终止时间，允许票据拥有任意长度的生命期。
5. **向前认证**：版本 4 中，发给客户端的证书不能转发给其他用户进行其他相关操作。此操作是指服务器为了完成客户端请求的服务而请求其他服务器协作的能力。例如，客户端申请打印服务器的服务，而打印服务器需要利用客户端证书访问文件服务器得到客户文件。版本 5 提供这项功能。
6. **域间认证**：版本 4 中，N 个域的互操作需要 $O(N^2)$ 个 Kerberos-to-Kerberos 关系。版本 5 中支持一种需要较少连接的方法。

除了这些环境限制，版本 4 中还存在一些**技术不足**。大多数不足在[BELL90]中有详细叙述，版本 5 试图予以解决。其不足如下：

1. **两次加密**：在表 14.1 中，提供给客户端的票据需要经过两次加密[消息(2)和(4)]，第一次使用的是目标服务器的密钥，第二次使用的是客户端密钥，面第二次加密并不是必须的。
2. **PCBC 加密**：版本 4 加密使用 DES 的非标准模式 PCBC (Propagating Cipher Block Chaining)，[②] 此种模式已被证明易受交换密码块攻击[KOHL89]。PCBC 试图在加密操作中提供完整性检查。版本 5 提供了精确的完整性检查机制，并用标准的 CBC 模式加密。
3. **会话密钥**：每一个包含会话密钥的票据均被客户端用于加密与票据一起送往服务器的认证消息。另外，会话密钥还可用于保护客户端与服务器间会话的消息。然而，多次使用同一个票据获得特定服务器的服务，将会增大攻击者从客户端或服务器获得以前的会话

① 下面的讨论参照[KOHL94]。

② 这将在附录 14A 中介绍。

消息的可能性。在版本 5 中，客户端与服务器可以协商得到子会话密钥，使得每个密钥仅被使用一次。这种新的客户端访问方式将会导致一种新的子密钥生成。

4. **口令攻击**：这两种版本均易受到口令攻击。从 AS 发往客户端的消息是用基于客户端用户口令的密钥加密的，[①] 攻击者可以捕获消息，并通过尝试各种口令解密。如果某一次解密成功，则攻击者即可得到客户端口令，进而使用该口令从Kerberos获取认证证书。这与第 18 章中描述的口令攻击类型相同，用相同的对策处理。版本 5 提供了一种称为预认证的机制，使得口令攻击更为困难，但无法杜绝它。

版本 5 认证会话

表 14.3 描述了版本 5 的基本会话，可以与版本 4 的表 14.1 进行对比。

表 14.3　Kerberos 版本 5 消息交换

(a) 认证服务交换：获取票据授权票据

(1) C → AS: $\text{Options} \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS → C: $Realm_c \| ID_C \| Ticket_{tgs} \| E_{Kc}[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}]$

$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times]$

(b) 服务授权票据交换：获取服务授权票据

(3) C → TGS: $\text{Options} \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V]$

$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times]$

$Ticket_v = E_{K_v}[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times]$

$Authenticator_c = E_{K_{c,tgs}}[ID_C \| Realm_c \| TS_1]$

(c) 客户/服务器认证交换：获取服务

(5) C → V: $\text{Options} \| Ticket_v \| Authenticator_c$

(6) V → C: $E_{K_{c,v}}[TS_2 \| \text{Subkey} \| \text{Seq\#}]$

$Ticket_v = E_{K_v}[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times]$

$Authenticator_c = E_{K_{c,v}}[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#]$

首先考虑**认证服务交换**。消息(1)是客户端请求票据授权票据过程。如前所述，它包括用户和 TGS 的标识。新增的元素包括：

- **Realm**：标识用户所属的域。
- **Options**：用于请求在返回的票据中设置指定的标识位，如下所述。
- **Times**：用于客户端请求在票据中设置时间：
  - from：请求票据的起始时间
  - till：请求票据的过期时间
  - rtime：请求 till 更新时间
- **Nonce**：在消息(2)中重复使用的临时交互号，用于确保应答是刷新的，且未被攻击者使用。

① 附录 14A 将讲述口令与加密密钥的映射。

消息(2)返回票据授权票据,标识客户端信息和一个用用户口令形成的密钥加密的数据块。该数据块包含客户端和 TGS 间使用的会话密钥、消息(1)中设定的时间和临时交互号以及 TGS 的标识信息。票据本身包含会话密钥、客户端的标识信息、需要的时间值、影响票据状态的标志和选项。这些标志为版本 5 带来的一些新功能将在以后讨论。

比较版本 4 和版本 5 的**服务授权票据交换**。两者的消息(3)均包含认证码、票据和请求服务的名字。在版本 5 中,还包括与消息(3)类似的票据请求的时间、选项和一个临时交互号;认证码的作用与版本 4 中的相同。

消息(4)与消息(2)结构相同,返回票据和一些客户端需要的信息,后者被客户端和 TGS 共享的会话密钥加密。

最后,版本 5 对客户/服务器认证交换进行了一些改进,如在消息(5)中,客户端可以请求选择双向认证选项。认证也增加了以下新域:

- **Subkey**:客户端选择一个子密钥保护某一特定应用会话,如果此域被忽略,则使用票据中的会话密钥 $K_{c,v}$。
- **Seq #**:序号,可选域,用于说明在此次会话中服务器向客户端发送消息的序列号。将消息排序可以防止重播攻击。

如果请求双向认证,则服务器按消息(6)应答。该消息中包含从认证消息中得到的时间戳。在版本 4 中该时间戳被加 1,而在版本 5 中,由于攻击者不可能在不知道正确密钥的情况下创建消息(6),因此不需要对时间戳进行上述处理。如果有子密钥域存在,则覆盖消息(5)中相应的子密钥域。而选项序列号则说明了客户端使用的起始序列号。

票据标志

版本 5 票据中的标志域支持许多版本 4 中没有的功能。表 14.4 总结了票据中可能包含的标志。

表 14.4 Kerberos 版本 5 标志

INITIAL	按照 AS 协议发布的服务授权票据,而不是基于票据授权票据发布的
PRE-AUTHENT	在初始认证中,客户在授予票据前即被 KDC 认证
HW-AUTHENT	初始认证协议要求带名客户端独占硬件资源
RENEWABLE	告知 TGS 此票据可用于获得最近超时票据的新票据
MAY-POSTDATE	告知 TGS 事后通知的票据可能基于票据授权票据
POSTDATED	表示该票据是事后通知的,终端服务器可以检查 authtime 域,查看认证发生的时间
INVALID	不合法的票据在使用前必须通过 KDC 使之合法化
PROXIABLE	告知 TGS 根据当前票据可以发放给不同网络地址新的服务授权票据
PROXY	表示该票据是一个代理
FORWARDABLE	告知 TGS 根据此票据授权票据可以发放给不同网络地址新的票据授权票据
FORWARDED	表示该票据或是经过转发的票据,或是基于转发的票据授权票据认证后发放的票据

标志 INITIAL 用于表示票据是由 AS 发放的，而不是由 TGS 发放的。当客户端向 TGS 申请服务授权票据时，必须拥有 AS 发放的票据授权票据。在版本 4 中，这是惟一获得服务授权票据的方法。版本 5 提供了一种可以直接从 AS 获得服务授权票据的手段，其机制是：一个服务器（如口令变更服务器）希望知道客户端口令近来已被验证。

标志 PRE-AUTHENT 如果被设置，则表示当 AS 接收初始请求[消息(1)]时，在发放票据前应先对客户端进行认证，其预认证的确切格式在此未做详细说明。例如，MIT 实现版本 5 时，加密的时间戳默认设置为预认证。当用户想得到一个票据时，它将一个带有临时交互号的预认证块、版本号和时间戳用基于客户口令的密钥加密后送往 AS。AS 解密后，如果预认证块中的时间戳不在允许的时间范围之内（时间间隔取决于时钟迁移和网络延迟），则 AS 不返回票据授权票据。另一种可能性是使用智能卡生成不断变化的口令，将其包含在预认证消息中。卡所生成的口令基于用户口令，但经过了一定的变换，使得生成的口令具有随机性，防止了简单猜测口令的攻击。如果使用了智能卡或其他相似设备，则设置 HW-AUTHENT 标志。

当票据的生命期较长时，就在相当一段时间内存在票据被攻击者窃取并使用的威胁。而缩短票据的生命期可降低这种威胁，因此，开销将主要依赖于获取新票据，如对票据授权票据而言，客户端可以存储用户密钥（危险性较大）或重复向用户询问口令来解决。一种解决方案是使用可重新生成的票据。一个具有标志 RENEWABLE 的票据中包含两个有效期：一个是此特定票据的有效期，另一个是最大许可值的有效期。客户端可以通过将票据提交给 TGS 申请得到新的有效期的方法获得新票据。如果这个新的有效期在最大有效期的范围之内，TGS 即发放一个具有新的会话时间和有效期的新票据。这种机制的好处在于，TGS 可以拒绝更新已报告为被盗用的票据。

客户端可请求 AS 提供一个具有标志 MAY-POSTDATE 的票据授权票据。客户端可以使用此票据从 TGS 申请一个具有标志 POSTDATED 或 INVALID 的票据，然后，客户端提交合法的超时票据。这种机制在服务器上运行批处理任务和经常需要票据时特别有用。客户端可以通过一次会话得到一组具有扩展性时间值的票据。但第一个票据被初始化为非法标志，当执行进行到某一阶段需要某一特定票据时，客户端即将相应的票据合法化。用这种方法，客户端就不再需要重复使用授权票据去获取服务授权票据。

在版本 5 中，服务器可以作为客户端的代理，获取客户端的信任和权利，并向其他服务器申请服务。如果客户端想使用这种机制，需要申请获得一个带有 PROXIABLE 标志的票据授权票据。当此票据传给 TGS 时，TGS 发布一个具有不同网络地址的服务授权票据。该票据的标志 PROXY 被设置，接收到这种票据的应用可以接收它或请求进一步认证，以提供审计跟踪。[①]

代理在转发时有一些限制。如果票据被设置为 FORWARDABLE，TGS 给申请者发放一个具有不同网址和 FORWARDED 标志的票据授权票据，于是此票据可以被送往远程 TGS。这使得用户端在不需要每个 Kerberos 都包含与其他各不同域中的 Kerberos 共享密钥的前提下，可以访问不同域的服务器。例如，各域具有层次结构时，客户端可以向上遍历到一个公共节点后再向下到达目标域。每一步都是转发票据授权票据到图中的下一个 TGS。

① 对于代理能力应用的讨论参见[NEUM93b]。

14.2 X.509 认证服务

ITU-T 建议书中的 X.509 是 X.500 系列中定义目录服务的一部分。实际上，目录是指管理用户信息数据库的服务器或一组分布服务器，用户信息包括用户名到网络地址的映射等用户信息或其他属性。

X.509 定义了 X.500 用户目录的一个认证服务框架，该目录可以提供第 9 章中公钥证书库类型的服务，每个证书包含该用户的公钥并由一个可信的认证中心用私钥签名。另外，X.509 还定义了基于使用公钥证书的一个认证协议。

X.509 是关于证书结构和认证协议的一种重要标准，并被广泛使用。例如，X.509 证书格式用于 S/MIME(第 15 章)、IP 安全性(第 16 章)和 SSL/TLS 与 SET(第 17 章)。

X.509 首次于 1988 年发布，随后在安全性方面被修正，参阅文献[IANS90]和[MITC90]，修订稿于 1993 年发布，1995 年发布第三版，2000 年被再次修改。

X.509 是基于公钥密码体制和数字签名的服务。其标准中并未规定使用某个特定的算法，但推荐使用 RSA；其数字签名需要用到 hash 函数，但并没有规定具体的 hash 算法。1988 年的建议书中推荐的 hash 算法被证明不安全后，在 1993 年的建议书中被删除。

14.2.1 证书

X.509 的核心是与每个用户相关的公钥证书。这些用户证书由一些可信的认证中心(CA)创建并被 CA 或用户放入目录服务器中。目录服务器本身不创建公钥和证书，仅仅为用户获得证书提供一种简单的存取方式。

图 14.3(a)介绍了一般证书的格式，它包含以下元素：

- **版本(Version)**：区分合法证书的不同版本，默认设置为 1。如果存在发行商惟一标识或主体惟一标识，则版本号为 2；如果存在一个或多个扩展，则版本号为 3。
- **序列号(Serial number)**：一个整数，在 CA 中惟一标识证书。
- **签名算法标识(Signature algorithm identifier)**：带参数的、用于给证书签名的算法，由于此信息在证书尾部的域 Signature 中还会出现，这里很少含该信息。
- **发行商名字(Issuer name)**：X.500 中创建、签名证书的认证中心 CA 的名字。
- **有效期(Period of validity)**：包含两个日期，即证书的生效日期和终止日期。
- **证书主体名(Subject name)**：获得证书的用户名，证明拥有相应私钥的主体是公钥的所有者。
- **证书主体的公钥信息(Subject's public-key information)**：主体的公钥以及将被使用的密钥的算法标识，带有相关的参数。
- **发行商惟一标识(Issuer unique identifier)**：由于 X.500 的名字被许多不同实体引用，因此用可选位串惟一标识认证中心。
- **证书主体惟一标识(Subject unique identifier)**：由于 X.500 的名字被许多不同实体引用，因此用可选位串惟一标识证书主体。
- **扩展(Extensions)**：一个或多个扩展域集，扩展域是在版本 3 中增加的，将在以后讨论。
- **签名(Signature)**：覆盖证书的所有其他域，以及其他域被 CA 私钥加密后的 hash 代码，以及签名算法标识。

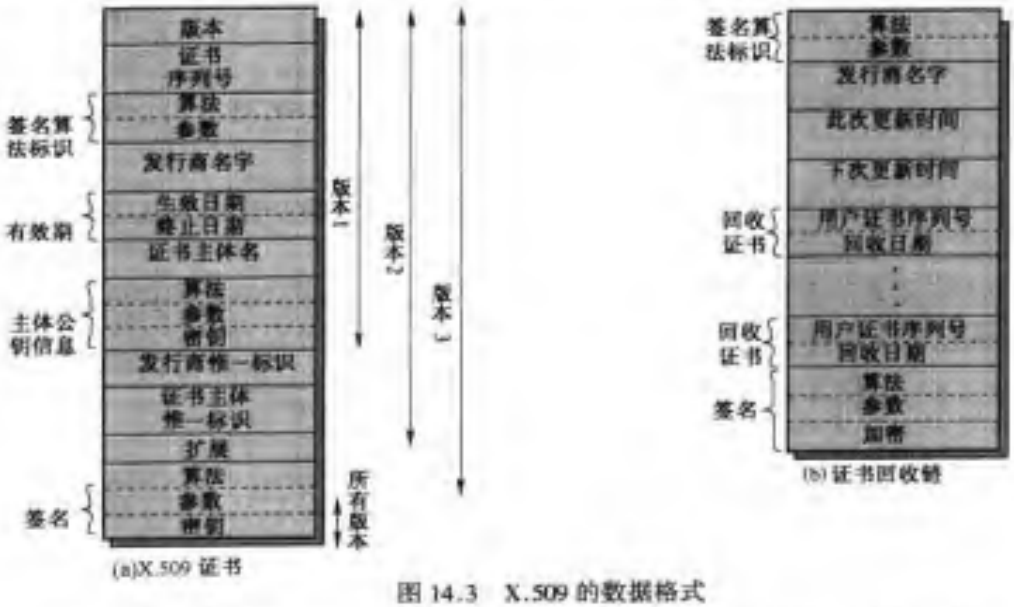


图 14.3　X.509 的数据格式

惟一标识域是在版本 2 中加入的，在证书主体或发行商名字出现重名时使用，一般很少使用。

该标准使用如下标注定义证书：

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, T_A, A, Ap\}$$

其中，

$Y\langle\langle X\rangle\rangle$ = 用户 X 的证书是认证中心 Y 发放的

$Y\{I\}$ = Y 签名 I，包含 I 和 I 被加密后的 hash 代码

CA 用它的私钥对证书签名，如果用户知道相应的公钥，则用户可以验证 CA 签名的证书的合法性，这是一种典型的数字签名方法，参见图 11.5(c)所示。

获得一个用户证书

CA 生成的用户证书具有以下特点：

- 任何可以访问 CA 公钥的用户均可获得证书中的用户公钥。
- 只有 CA 可以修改证书。

由于证书不可伪造，因此证书可以存放在目录中，而不需要对目录进行特别保护。

如果所有用户都属于同一个 CA，则说明对该 CA 有普遍信任，所有用户的证书均可存放于同一个目录中，以被所有用户存取。另外，用户也可以直接将其证书传给其他用户。一旦 B 拥有了 A 的证书，B 即可确信用 A 的公钥加密的消息是安全的，不可能被窃取；同时，用 A 的私钥签名的消息也不可能仿造。

如果用户数量很多，不可能期望所有用户从同一个 CA 获得证书。由于证书是由 CA 签发的，每一个用户都需要拥有一份 CA 的公钥来验证签名，该公钥必须用一种绝对安全的方式提供给每个用户，使得用户可以信任该证书。因此，拥有许多 CA，并要求每个 CA 按一种安全的方式给其用户群提供该 CA 的公钥。

现在，假设 A 获得了认证中心 X_1 的证书，而 B 获得了认证中心 X_2 的证书，如果 A 无法安全地获得 X_2 的公钥，则由 X_2 发放的 B 的证书对 A 而言就无法使用，A 只能读取 B 的证书，但无法验证其签名。然而，如果两个 CA 之间能安全地交换它们的公钥，则 A 可以通过下述过程使 A 获得 B 的公钥：

1. A 从目录中获得由 X_1 签名的 X_2 的证书，由于 A 知道 X_1 的公钥，A 可从证书中获得 X_2 的公钥，并用 X_1 的签名来验证证书。
2. A 再到目录中获取由 X_2 颁发的 B 的证书，由于 A 已经得到了 X_2 的公钥，A 即可利用它验证签名，安全地获得 B 的公钥。

A 使用了一个证书链来获得 B 的公钥，在 X.509 中，该链表示如下：

$$X_1\langle\langle X_2\rangle\rangle\ X_2\langle\langle B\rangle\rangle$$

同样，B 可以逆向链获得 A 的公钥：

$$X_2\langle\langle X_1\rangle\rangle\ X_1\langle\langle A\rangle\rangle$$

上述模式并不仅仅限于两个证书，对经过长度为 N 的 CA 链的认证过程可表示如下：

$$X_1\langle\langle X_2\rangle\rangle\ X_2\langle\langle X_3\rangle\rangle\cdots X_N\langle\langle B\rangle\rangle$$

在这种情况中，链中的每对 $CA(X_i, X_{i+1})$ 必须互相发放证书。

所有由 CA 发放给 CA 的证书必须放在一个目录中，用户必须知道如何找到一条路径获得其他用户的公钥证书。在 X.509 中，推荐采用层次结构放置 CA 证书，以利于建立强大的导航机制。

图 14.4 描述了 X.509 中推荐的层次结构，相连的圆圈表示 CA 间的层次结构，相连的方框表示每个 CA 发放的证书所在的目录，每个 CA 目录入口中包含两种证书：

- **向前证书**：由其他 CA 发给 X 的证书。
- **向后证书**：X 发给其他 CA 的证书。

例如，用户 A 可以通过创建一条到 B 的路径获得相关证书：

$$X\langle\langle W\rangle\rangle\ W\langle\langle V\rangle\rangle\ V\langle\langle Y\rangle\rangle\ Y\langle\langle Z\rangle\rangle\ Z\langle\langle B\rangle\rangle$$

当 A 获得相关证书后，可以通过顺序展开证书路径获得 B 的公钥，用这个公钥，A 可将加密消息送往 B，如果 A 想得到 B 返回的加密消息或对发往 B 的消息签名，则 B 需要按照下述证书路径获得 A 的公钥：

$$Z\langle\langle Y\rangle\rangle\ Y\langle\langle V\rangle\rangle\ V\langle\langle W\rangle\rangle\ W\langle\langle X\rangle\rangle\ X\langle\langle A\rangle\rangle$$

B 可以获得目录中的证书集，或 A 可在它发给 B 的初始消息中将其包含进去。

证书的撤销

回想图 14.3，每一个证书都有一个有效期，这与信用卡相似。通常，新的证书会在旧证书失效前发放。另外，还可能由于以下原因提前撤销证书：

1. 用户密钥被认为不安全。
2. 用户不再信任该 CA。

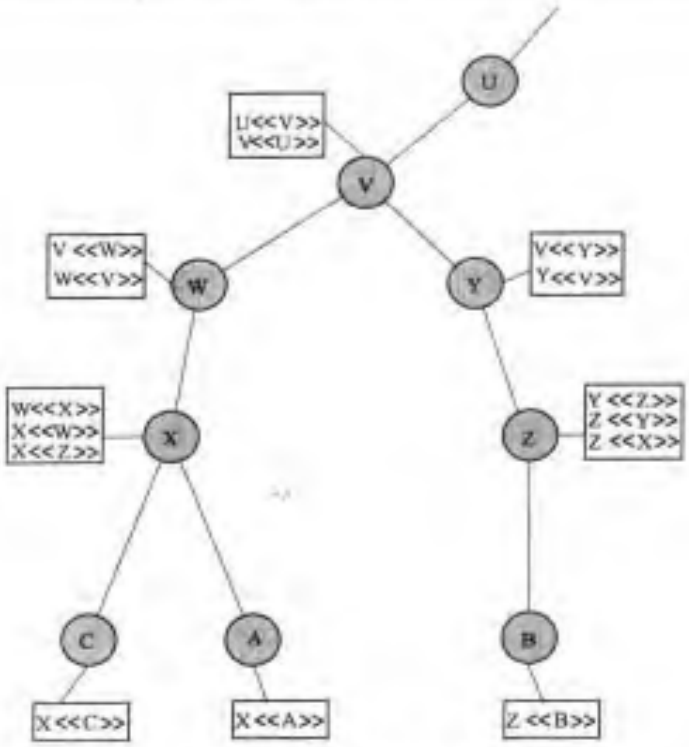


图 14.4　描述了 X.509 中层次结构的一个例子

3. CA 证书被认为不安全。

每个 CA 必须保留一张表，其中包含所有被 CA 撤销且还未到期的证书，包括发给用户和其他 CA 的证书，这张表也应被放在目录中。

每个放在目录中的证书撤销表(CRL)均被其发行商签名，并包含发行商的名字、表创建的时间、下一张 CRL 表发放的时间以及每个撤销证书的入口。每个入口中包含该证书的序列号和撤销时间。由于一个 CA 中的序列号是惟一的，因此序列号足以表示一张证书。

当一个用户在一个消息中接收了一个证书，该用户必须确定该证书是否已被撤销。用户可以在接到证书时检查目录，为了避免目录搜索时的延迟，用户可以将证书和 CRL 缓存。

14.2.2　认证的过程

X.509 也包含三种可选的认证过程，这些过程可以应用于各种应用程序。三种方法均采用了公钥签名。假设双方知道对方的公钥，可以通过目录服务获得证书或证书由初始消息携带。

图 14.5 给出了三种过程。

单向认证

单向认证包含一个从用户 A 到 B 的简单信息传递，包括：

1. A 的标识和 A 创建的消息。

2. B 所需要的消息。

3. 消息的完整性和原创性(不能多次发送)。

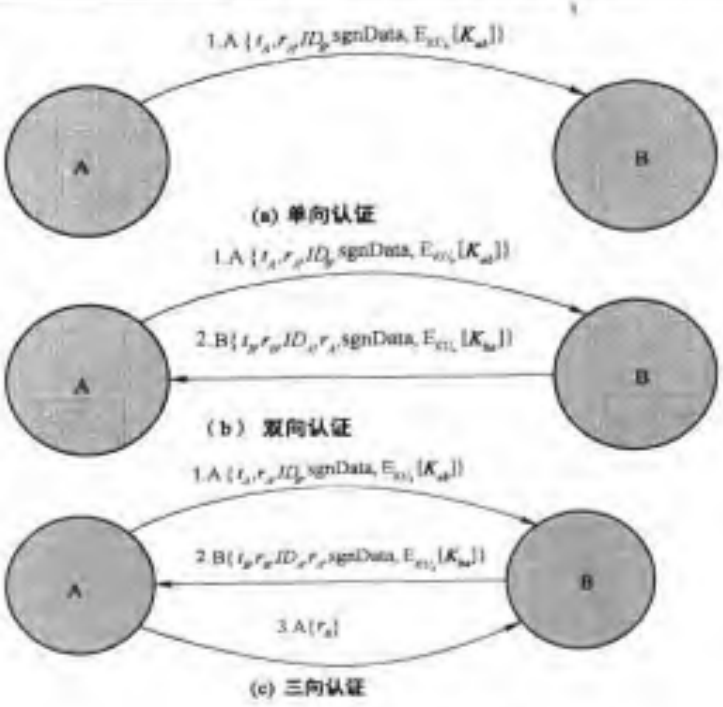


图 14.5 X.509 的强认证过程

注意，初始实体的标识在这个过程中被验证，而不是应答实体被验证。

至少，消息中应包括时间戳 t_A、临时交互号 r_A、B 的标识，并用 A 的私钥加密。时间戳由起始时间(可选)和终止时间组成，可以防止消息的延迟传递；临时交互号(nonce)用于防止重放攻击，其值在消息的起止时间之内惟一。这样，B 即可存储临时交互号直至它过期，并拒绝接受其他具有相同临时交互号的新消息。

对于纯认证而言，消息被用做简单地向 B 提供证书。消息可以包含要传送的信息。将信息放在签名的范围内，保证其真实性和完整性。消息也可以用 B 的公钥将会话密钥加密后传送给 B。

双向认证

除了上述三个元素外，双向认证还需要建立以下元素：

4. B 的标识和 B 生成的应答消息。
5. A 需要的消息。
6. 应答的完整性和真实性。

双向认证允许通信双方验证对方的身份。

应答消息中包括从 A 中得到的临时交互号、时间戳和 B 生成的临时交互号。与前面类似，消息中可以包含签名的其他信息和用 A 的公钥加密过的会话密钥。

三向认证

在三向认证中，包括最后从 A 发往 B 的消息中包含签名了的临时交互号 r_B。这样，其中

的时间戳就不用检查了，因为双方的临时交互号均被回送给了对方，各方可以使用回送的临时交互号来防止重播攻击。这种方法在没有同步时钟时使用。

14.2.3　X.509 版本 3

X.509 的版本 2 中没有将设计和实践当中所需要的所用信息均包含进去。[FORD95]列出了版本 2 中未满足的需求：

1. 证书主体域不足以将密钥所有者转换为公钥用户。X.509 中的名字相对较短，不能满足用户需要知道标识细节的要求。
2. 证书主体域在许多应用中不足以满足需要，通常需要用互联网邮件地址、URL 和其他与互联网相关的标识。
3. 需要标明安全策略信息。这可以将安全应用或安全功能，如 IPSec 等，与 X.509 联系，将策略应用于证书。
4. 需要对证书的使用范围进行限定，以缩小 CA 失误或恶意破坏造成的影响。
5. 能区分同一个用户在不同时刻使用不同密钥是非常重要的。这一特性支持密钥的生命周期管理，特别是在一般或特殊情况下更新用户和 CA 密钥对的能力。

与向固定格式中继续增加字段相比，标准的开发者认为需要一种更方便的方法。于是，版本 3 的格式增加了一些可选的扩展项。每一个扩展项有一个扩展标识、一个危险指示和一个扩展值。危险指示用于指出该扩展项是否能安全地被忽略，如果值为 TRUE 且实现时未处理它，则其证书将会被当做非法的证书。

证书扩展项有三类：密钥和策略信息、证书主体和发行商属性以及证书路径约束。

密钥和策略信息

此类扩展项传递的是与证书主体和发行商密钥相关的附加信息，以及证书策略的指示信息。一个证书策略是一个带名的规则集，在普通安全级别上描述特定团体或应用类型证书的使用范围。例如，某个策略可用于电子数据交换(EDI)在一定价格范围内的贸易认证。

这个范围包括：

- **授权密钥标识符**：标识用于验证证书或 CRL 上的签名的公钥。同一个 CA 的不同密钥得以区分，该字段的一个用法是用于更新 CA 密钥对。
- **主体密钥标识符**：标识被证实了的公钥，用于更新主体的密钥对。同样，一个主体对不同目的的不同证书可以拥有许多密钥对(例如，数字签名和加密密钥协议)。
- **密钥使用**：说明被证实的公钥的使用范围和使用策略。可以包含以下内容：数字签名、非抵赖、密钥加密、数据加密、密钥一致性、CA 证书的签名验证和 CA 的 CRL 签名验证。
- **私钥使用期**：表明与公钥相匹配的私钥的使用期。通常，私钥的使用期与公钥不同。例如，在数字签名密钥中，签名私钥的使用期一般比其公钥短。
- **证书策略**：证书可以在应用多种策略的各种环境中使用。该扩展项中列出了证书所支持的策略集，包括可选的限定信息。
- **策略映射**：仅用于其他 CA 发给 CA 的证书中。策略映射允许发行 CA 将其一个或多个策略等同于主体 CA 域中的某个策略。

证书主体和发行商属性

该扩展支持证书主体或发行商以可变的形式拥有可变的名字，并可传递证书主体的附加信息，使得证书所有者更加确信证书主体是一个特定的人或实体。例如，一些信息如邮局地址、公司位置或一些图片。

扩展域包括：

- **主体可选名字**：包括使用任何格式的一至两个可选名字。该字段对特定应用如电子邮件、EDI、IPSec 等使用自己的名字形式非常重要。
- **发行商可选名字**：包括使用任何格式的一至两个可选名字。
- **主体目录属性**：将 X.500 目录的属性值转换为证书的主体所需要的属性值。

证书路径约束

该扩展项允许在 CA 或其他 CA 发行的证书中包含限制说明。这些限制信息可以限制主体 CA 所能发放的证书种类或证书链中的种类。

扩展域包括：

- **基本限制**：标识该主体是否可作为 CA，如果可以，证书路径长度被限制。
- **名字限制**：表示证书路径中的所有后续证书的主体名的名字空间必须确定。
- **策略限制**：说明对确定的证书策略标识的限制，或证书路径中继承的策略映射的限制。

14.3 推荐读物和网址

为掌握 Kerberos 概念，可参考[BRYA88]；Kerberos 详细叙述可参考[KOHL94]。[TUNG99]从用户的角度描述了 Kezberos。

BRYA88 Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Project Athena document, February 1988. Available at http://web.mit.edu/kerberos/www/dialogue.html。

KOHL94 Kohl, J.; Neuman, B.; and Ts'o, T. "The Evolution of the Kerberos Authentication Service." In Brazier, F., and Johansen, D. *Distributed Open Systems*. LosAlamitos, CA: IEEE Computer Society Press, 1994. Available at http://web.mit.edu/kerberos/www/papers.html。

TUN G99 Tung, B. *Kerberos: A Network Authentication System*. Reading, MA: Addison-Wesley, 1999。

推荐网址：

- **MIT Kerberos Site**：Kerberos 的有关信息，包括 FAQ、论文、文档以及相关商业产品。
- **USC/ISI Kerberos Page**：Kerberos 素材。
- **Public-Key Infrastructure Working Group**：开发基于 X.509v3 的标准的 IETF 组。

- **Verisign**：X.509 相关产品的开发商，该站点上有白皮书和其他有价值的资料。

14.4　关键术语、思考题和习题

14.4.1　关键术语

认证	临时交互号	序列号
认证服务器	传播密码块链接模式	子密钥
Kerberos	公钥证书	票据
Kerberos 服务器	域	票据授权服务器(TGS)
生命周期		X.509 证书

14.4.2　思考题

14.1　Kerberos 是为解决什么问题而设计的？

14.2　在网络环境下用户认证面临哪三种威胁？

14.3　列出在分布式环境下进行安全用户认证的三种方法。

14.4　为 Kerberos 定义了哪四个需求？

14.5　Kerberos 认证环境由哪些实体构成？

14.6　根据 Kerberos 的上下文，什么是域？

14.7　Kerberos 的版本 4 和版本 5 之间的主要区别是什么？

14.8　X.509 标准的目的是什么？

14.9　什么是认证链？

14.10　X.509 是如何撤销证书的？

14.4.3　习题

14.1　在 PCBC 模式下，如果某块密文出现了随机错，则错误将会传播到后继的明文块中(参见图 14.7)，请说明之。

14.2　假设在 PCBC 模式下，块 Ci 和 Ci + 1 在传送过程中交换了，请证明其影响仅涉及到解密块 Pi 和 Pi + 1，与后续块无关。

14.3　图 14.5(c)描述的 X.509 中的三向认证过程中存在安全缺陷。该协议的实质如下：

$$\begin{aligned} &A \rightarrow B: \quad A\{t_A, r_A, ID_B\} \\ &B \rightarrow A: \quad B\{t_B, r_B, ID_A, r_A\} \\ &A \rightarrow B: \quad A\{r_B\} \end{aligned}$$

X.509 中规定在三向认证中时间戳 t_A 和 t_B 是可选的，但考虑下述情况：假设 A 和 B 在以前曾用过上述协议，攻击者 C 截获了上述三条消息。另外，假设未使用时间戳，并均设置为 0。最后，假设 C 想冒充 A 与 B 通信。C 开始向 B 发送捕获的第一条消息：

$$C \rightarrow B: \quad A\{0, r_A, ID_B\}$$

B 应答，认为是在和 A 通话，但实际上是和 C 通话：

$$B \rightarrow C: \quad B\{0, r'_B, ID_A, r_A\}$$

C 用某种方式使得 A 与 C 进行初始认证，因此，A 向 C 发送消息：

$$A \rightarrow C: \quad A\{0, r'_A, ID_C\}$$

C 应答 A，使用 B 发给 A 的临时交互号 r'_B：

$$C \rightarrow A: \quad C\{0, r'_B, A, r'_A\}$$

A 应答：

$$A \rightarrow C: \quad A\{r'_B\}$$

这就是 C 需要向 B 证实它正在与 A 通话所需要的临时交互号，因此，C 可以重复将此消息发给 B：

$$C \rightarrow B: \quad A\{r'_B\}$$

这样，B 就会相信它是在和 A 通话，而实际上它是在和 C 通话。这个问题的一个简单的解决方法可以不涉及到使用时间戳，请给出该方法。

14.4 1988 年 X.509 指出了 RSA 的密钥必须满足安全性，即用目前已有的知识分解大数是困难的。这在公共指数和模 n 上导出约束：必须确保 $e > \log_2(n)$，通过采用模 n 的第 e 个根来阻止发现明文的攻击。虽然约束是正确的，但给出的原因是错误的。什么原因是正确的，什么原因是错误的？

附录 14A Kerberos 加密技术

Kerberos 包括一个支持各种加密操作的加密库。

从口令到密钥的转换

Kerberos 中，口令被限制在可见的 7 位 ASCII 字符范围以内。任何长度的口令都可以被转换成为加密密钥并存储于 Kerberos 数据库中。图 14.6 描述了这个过程。

首先，口令字符串 s 被紧凑装入一个二进制串 b，s 的每个字符仅占用 7 个二进制位，顺序排放，如：

$$\begin{aligned}
b[0] &= \text{bit 0 of } s[0] \\
&\vdots \\
b[6] &= \text{bit 6 of } s[0] \\
b[7] &= \text{bit 0 of } s[1] \\
&\vdots \\
b[7i+m] &= \text{bit } m \text{ of } s[i] \qquad 0 \leq m \leq 6
\end{aligned}$$

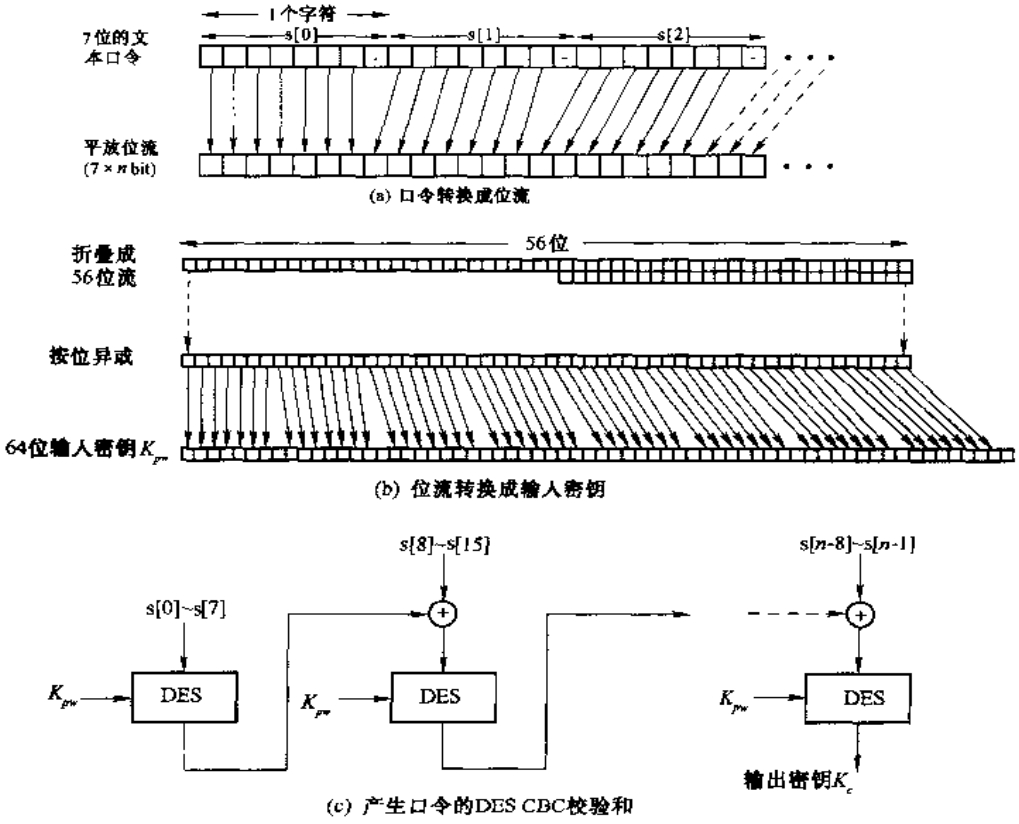


图 14.6　从口令生成加密密钥

然后，这个位串 b 通过 XOR 操作被折叠压缩成为 56 位。例如：设 b 是 59 位，则：

$$b[55] = b[55] \oplus b[56]$$
$$b[54] = b[54] \oplus b[57]$$
$$b[53] = b[53] \oplus b[58]$$

这样便得到了一个 56 位的 DES 密钥，为了得到 64 位的 DES 密钥的格式，再将 56 位的位串中的每 7 位一组，分别扩展成 8 位，得到一个 64 位的输入密钥 K_{pw}。

最后，以 K_{pw} 为密钥，原始口令的 s 位明文采用密文块链接（CBC）模式加密。最后一块的 64 位密文，作为 CBC 校验和，成为了与口令 s 相对应的输出密钥。

这整个过程可看做是从任意一个口令到一个 64 位串的 hash 函数。

传播密码分组块链接模式

回忆第 3 章中 DES 的 CBC 模式，在相同的密钥下，DES 的每一段输入都是由明文 XOR 前一段加密结果的密文构成的（参见图 3.12）。与电子密码本（ECB）模式相比，它的优势在于：前者，明文中相同的段，被加密后，密文不相同；后者，明文段相同，密文相同。

CBC 有这样的特性：如果在传输中密文 C_i 发生错误，那么该错误会传播到解密后的明文 P_i 和 P_{i+1}。

Kerberos 版本 4 使用了一种扩展的 CBC，称为传播密文块链接模式（PCBC）[MEYE82]。它

可以使密文中出现一段错误将导致其后的所有段全部错误，数据无法使用。这样可以在加密的同时保证完整性。

图 14.7 描述了 PCBC。这种模式下，DES 的每一段输入都是由明文 XOR 前一段的明文 XOR 前一段的密文构成的。

$$C_n = E_K[C_{n-1} \oplus P_{n-1} \oplus P_n]$$

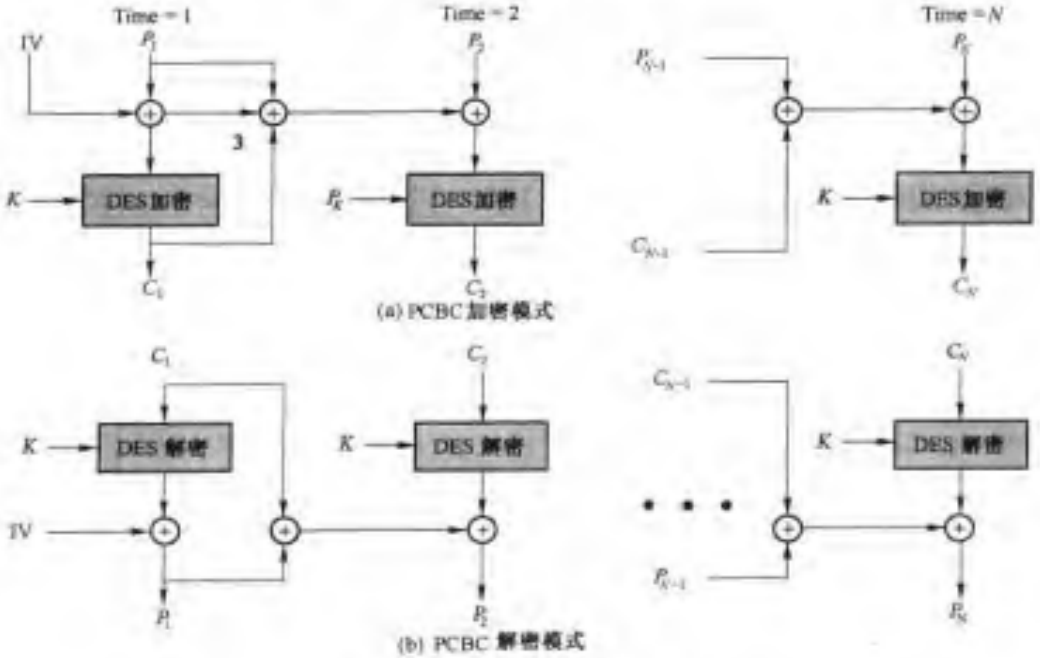


图 14.7 传播密码块链接模式

解密时，每次解密的结果 XOR 前一段的明文 XOR 前一段的密文，最终得到这一段的明文。我们可以这样看：

$$\begin{aligned} D_K[C_n] &= D_K[E_K[C_{n-1} \oplus P_{n-1} \oplus P_n]] \\ &= C_{n-1} \oplus P_{n-1} \oplus P_n \end{aligned}$$

$$C_{n-1} \oplus P_{n-1} \oplus D_K[C_n] = P_n$$

第 15 章　电子邮件安全

电子邮件是最广泛的网络应用，也是在异构环境下惟一跨平台的、通用的分布式系统。用户希望并能够直接或间接地给互联网相连的其他人发邮件，而不论双方使用的是何种操作系统或通信协议。

随着对电子邮件依赖的爆炸式增长，其认证和保密性的需求也日益增长。本章讨论两种广泛应用的方法：PGP(Pretty Good Privacy)和 S/MIME。

15.1　PGP

由 Phil Zimmermann 等人提出的 PGP 可以在电子邮件和文件存储应用中提供保密和认证服务。实际上，Zimmermann 做了如下工作：

1. 在构造块时，选择了可获得的、最好的密码算法。
2. 将这些算法集成到通用应用程序，这些应用独立于操作系统和处理器，而只依赖于一个易用的小规模指令集。
3. 将其封装成包和文档，包括源码，可在公告牌和商用网络如 AOL(America On Line)等平台上自由使用。
4. 与公司达成协议(Viacrypt：现在的 Network Associates)，提供完全兼容且低价的商用 PGP 版。

PGP 的应用呈爆炸性增长，并迅速普及，原因可大致归纳如下：

1. 提供世界范围内免费的各种版本，可运行于各种平台，包括 Windows、UNIX、Macintosh 等。另外，其商用版能使用户得到销售商的技术支持。
2. 使用的算法经过了充分的公众检验，且被认为是非常安全的算法。特别是，软件包包含 RSA、DSS、Diffie-Hellman 等公钥加密算法，以及 CAST-128、IDEA 和 3DES 对称钥加密算法，hash 编码算法 SHA-1。
3. 应用范围广泛，既可用于为加密文件选择和使用一种标准模式的公司，也可用于希望与互联网或其他网络上的其他人安全地通信的个人。
4. 不由任何政府或标准制定机构控制。因为对上述机构控制的协议，人们有本能的不信任，这使得 PGP 更有吸引力。
5. PGP 现已成为 Internet 标准文档(RFC 3156)，而且 PGP 是反对主流派努力的结果。

我们将从 PGP 的整体操作入手，接着介绍如何创建和存储密钥，最后讲述公钥管理的相关问题。

15.1.1　符号约定

本章中所使用的大部分符号在前面的章节中已经出现过，只有少量新的标识符，我们首先介绍这些符号。使用的符号如下：

K_s = 会话密钥,用于传统加密体制中

KR_a = 用户 A 的私钥,用于公钥加密体制中

KU_a = 用户 A 的公钥,用于公钥加密体制中

EP = 公钥加密

DP = 公钥解密

EC = 对称加密

DC = 对称解密

H = hash 函数

|| = 串联

Z = 用 ZIP 算法压缩

R64 = 转换为基数 64 的 ASCII 码格式

PGP 文档通常使用术语**密钥**来指公钥密码体制中与公钥相对的密钥,容易与前述对称密钥中的密钥混淆。因此,我们使用术语**私钥**来替代。

15.1.2 操作描述

PGP 的实际操作与密钥管理相对,包括五种服务:认证、保密、压缩、电子邮件兼容性和分段(表 15.1)。以下我们将逐个讨论。

表 15.1 PGP 服务概述

功 能	使用的算法	描 述
数字签名	DSS/SHA 或 RSA/SHA	消息的 hash 码利用 SHA-1 产生,将此消息摘要和消息一起用发送方的私钥按 DSS 或 RSA 加密
消息加密	CAST 或 IDEA,或使用 Diffie-Hellman 的 3DES 或 RSA	将消息用发送方生成的一次性会话密钥按 CAST-128 或 IDEA 或 3DES 加密。用接收方公钥按 Diffie-Hellman 或 RSA 算法加密会话密钥,并与消息一起加密
压缩	ZIP	消息在传送或存储时可用 ZIP 压缩
电子邮件兼容性	基数 64 转换	为了对电子邮件应用提供透明性,一个加密消息可以用基数 64 转换为 ASCII 串
分段	—	为了符合最大消息尺寸限制,PGP 执行分段和重新组装

认证

图 15.1(a)描述了 PGP 提供的数字签名服务。这种数字签名模式曾在第 13 章中讨论过,参见图 11.5(c)。过程如下:

1. 发送方创建消息。
2. 用 SHA-1 生成消息的 160 位 hash 码。
3. 用发送方的私钥按 RSA 加密 hash 码,并将结果放入消息中。
4. 接收方使用发送方的公钥按 RSA 解密,恢复 hash 码。
5. 接收方从消息中生成新的 hash 码,并与解密得到的 hash 码比较。如果匹配,则接收到的消息是真实的。

SHA-1 和 RSA 的组合提供了一种有效的数字签名模式。由于 RSA 的强度，接收方可以确信只有匹配私钥的拥有者才能生成签名。由于 SHA-1 的强度，接收方可以确信其他人都不可能生成与该 hash 编码匹配的新消息，从而确保是原始消息的签名。

也可使用 DSS/SHA-1 作为可选方案。

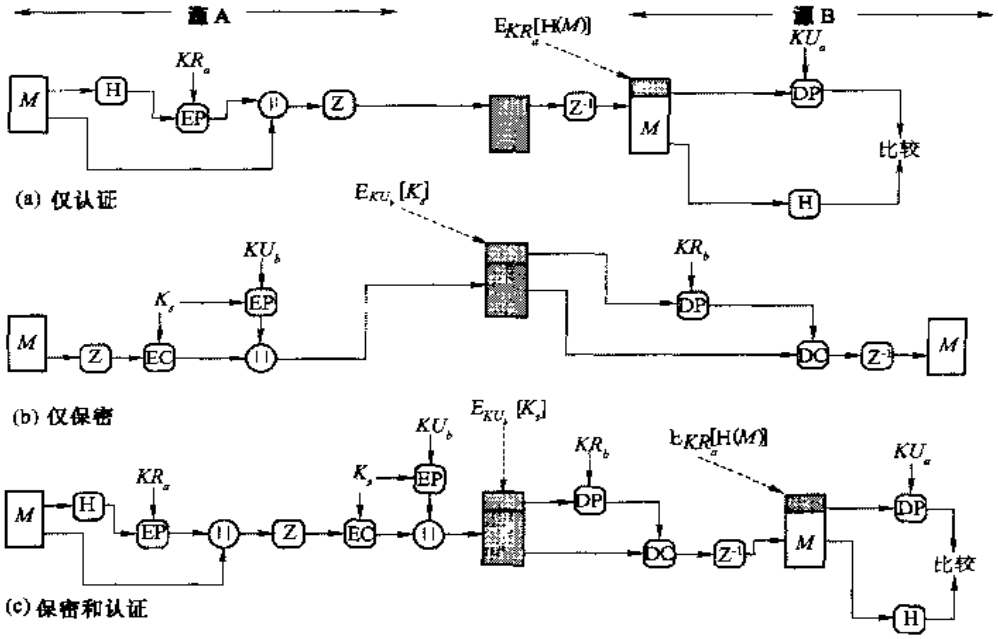


图 15.1　PGP 的密码功能

虽然签名通常与被签名的消息和文件绑定，但并不总是如此；也支持分离的签名。分离签名可以与其签名消息分开存储和传送，这在许多上下文中都是有用的。例如，用户也许会希望为所有发送和接收的消息保存分离的签名日志。一个执行程序的分离签名可测程序以后是否被病毒感染。最后，分离签名可用于对一个合法合同等文档的多方签名。每个人的签名彼此独立，并只与该文档相关。否则，签名必须嵌套，第二个签名的对象必须是文档和第一个签名，依此类推。

保密

PGP 的另一个基本服务是保密，通过对要传递的消息或要存储的本地文件实施加密。在两种情况下，可以使用对称加密算法 CAST-128，也可以使用 IDEA 或 3DES。使用 64 位的密码反馈模式(CFB)加密。

通常，必须讨论密钥分配问题。在 PGP 中，每个对称密钥仅使用一次，即为每一个消息生成一个 128 位的随机数作为新密钥。在文档中，作为会话密钥的该密钥是一次性密钥，只使用一次，必须将此密钥与消息一起绑定传送。为了保护此会话密钥，需要使用接收方的公钥对其加密。图 15.1(b)示例了该过程，描述如下：

1. 发送方生成消息和作为会话密钥的 128 位随机数。
2. 用会话密钥按 CAST-128(或 IDEA、3DES)加密消息。
3. 用接收方的公钥按 RSA 加密会话密钥，放入消息中。
4. 接收方使用其私钥按 RSA 解密和恢复会话密钥。

5. 使用会话密钥解密消息。

PGP 提供的 Diffie-Hellman 算法可替代 RSA 进行加密。如第 10 章所述，Diffie-Hellman 是一种密钥交换算法。实际上，PGP 使用 Diffie-Hellman 的变体 ElGamal 来进行加密/解密，详细叙述参见习题 6.19。

我们会获得某些发现。首先，为了减少加密时间，通常使用对称和公钥加密的组合方式来简化直接使用 RSA 或 ElGamal 加密消息：CAST-128 和其他传统算法比 RSA 或 ElGamal 算法快。其次，使用公钥算法解决了会话密钥的分配问题，因为只有接收方能恢复绑定在消息中的会话密钥。注意，我们不需要使用第 10 章中讨论的会话密钥交换协议，因为我们不会从正在进行的会话中开始，而是每个消息都使用与事件无关的一次性密钥。另外，由于电子邮件具有存储并转发的属性，使用握手协议确保双方拥有相同的会话密钥是不实际的。最后，使用一次性的对称密钥加强了已经是强加密算法的加密方法。每个密钥仅仅只加密少量原文，并且密钥之间没有联系。在这种程度下，公钥算法是安全的，从而整个模式是安全的。同时，PGP 提供了一个 768 位到 3072 位的密钥大小范围（DSS 的签名密钥限制在 1024 位）。

保密和认证

如图 15.1(c)所示，可以将保密和认证两种服务同时应用于一个消息。首先，为原始消息生成签名。然后，用 CAST-128（或 IDEA、3DES）加密带签名的明文消息，并用 RSA（或 ElGamal）加密会话密钥。与先加密消息、而后为加密后的消息签名的方式相比，这种方式要好一些，因为将签名与原始消息一起存储比较方便，且在三方认证中，如果先签名，则第三方在验证签名时不需要考虑对称密钥。

总之，当同时使用保密和认证时，发送方首先用自己的私钥加密签名消息，然后用会话密钥加密消息和签名，再用接收方的公钥加密会话密钥。

压缩

默认情况下，PGP 在签名之后、加密之前要对消息进行压缩。这可以为电子邮件传输和文件存储节约空间。

压缩算法中用于压缩的 Z 和用于解压的 Z^{-1} 是关键如图 15.1 所示：

1. 在压缩前生成签名的原因有两个：
   a. 对未压缩的消息签名可以将未压缩的消息和签名一起存放，以在将来验证时使用。而如果对一个压缩的文档签名，则将来要么将消息的压缩版本存放下来用于验证，要么在需要验证时再对消息进行压缩。
   b. 如果想动态重新压缩消息进行验证，用 PGP 现有的压缩算法将比较困难。因为该算法是不确定的；当在运行速度和压缩比之间权衡时，为了获得不同的比值，算法会有不同的实现，从而产生不同的压缩格式。然而，这些不同的压缩算法是可以互操作的，因为任何一个版本的算法均可以被其他版本正确解压。而在压缩之后应用 hash 函数和签名，必须要求 PGP 实现时采用同一种压缩算法。
2. 在压缩后，加密消息可以加强密码的安全性。由于压缩后消息的冗余信息比原文少，使得密码分析更加困难。

使用的压缩算法是 ZIP，将在附录 15A 中介绍。

电子邮件兼容性

使用 PGP 时，至少要加密块中将要被传递的部分。如果只使用签名服务，就必须用发送方的私钥对消息摘要加密。如果使用保密服务，就需要将消息和签名（如果有）以一次性对称密钥加密。因此，得到的部分或全部块由任意的 8 位字节流组成。然而，许多电子邮件系统仅仅允许使用由 ASCII 码组成的块。为了适应这个限制，PGP 提供了将 8 位二进制流转换为可打印的 ASCII 码字符的功能。

为此目的服务的模式称为基数 64 转换。一组三个 8 位二进制数据映射为四个 ASCII 码字符，同时加上 CRC 校验以检测传送错误，详细描述参见附录 15B。

使用基数 64 将导致消息大小增加 33%。但幸运的是，会话密钥和消息的签名部分都相当紧凑，且原始消息将被压缩。实际上，压缩的效果不仅可以补偿基数 64 转换导致的膨胀，还可以大大减少占有的空间。例如，[HELD96]中指出使用 ZIP 的压缩率平均为 2.0。如果我们忽略相对长度很小的签名和密钥部分，则压缩后的典型长度将为原始长度的 $1.33 \times 0.5 \times X = 0.665 \times X$。因此，仍然有三分之一被压缩。

一个基数 64 转换算法盲目地将输入串转化为基数 64 格式，它与上下文无关，即使在输入是 ASCII 文本时也是如此。因此，如果一个消息被签名但未加密，且转换作用于整个块，则输出对窃听者不可读，从而提供了一定程度的保密性。PGP 也可以选择只对消息的签名部分进行基数 64 转换，使得接收方可以不使用 PGP 直接阅读消息。PGP 也可以用于验证签名。

图 15.2 描述了上述四种服务之间的关系。在传输中，如果需要，可用压缩明文的 hash 编码生成签名，再将签名和明文一起压缩。接着，如果需要保密，可对由压缩的明文或压缩的签名与明文所构成的块加密，与用公钥加密的对称加密密钥一起，转换为基数 64 格式。

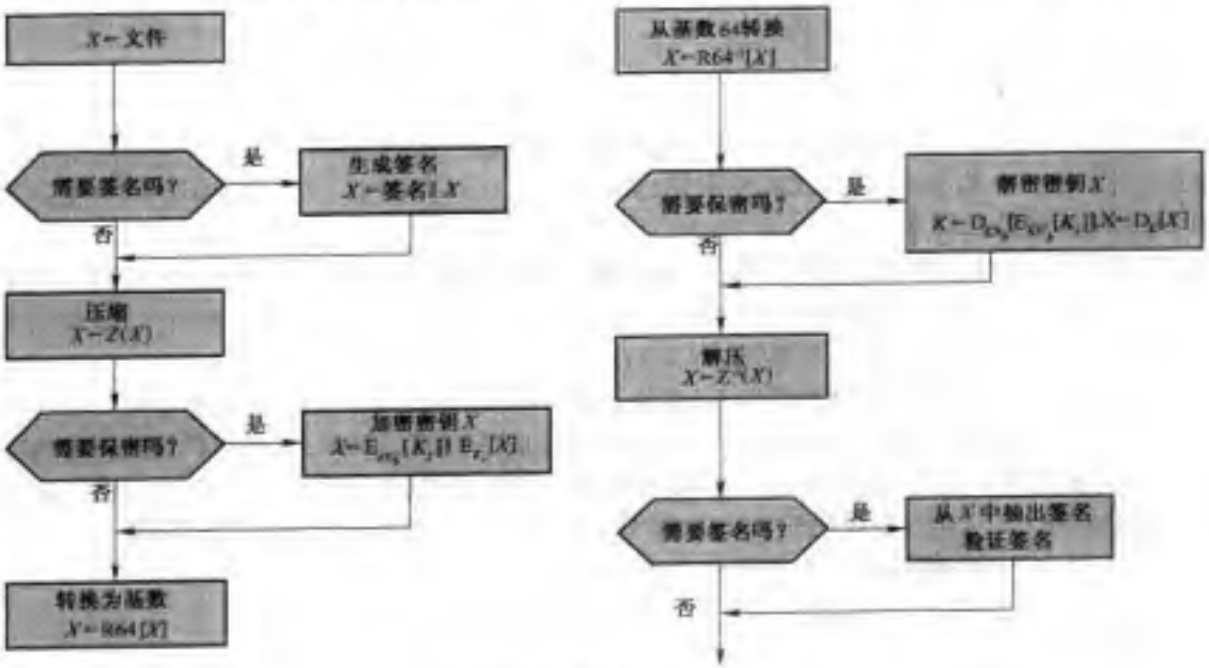


图 15.2　PGP 消息的发送与接收

在接收端,将收到的块首先从基数 64 转换为二进制。然后,如果消息加密过,则接收方恢复会话密钥,解密消息,再将得到的块解压。如果消息被签名,则接收方恢复传送过来的 hash 码,并与原 hash 码比较。

分段和组装

电子邮件工具通常限制消息的最大长度。例如,在互联网上提供的许多工具都将最大长度限制在 50 000 个 8 位字节,任何大于该长度的消息必须分成若干小段,单独发送。

为了适应这个限制,PGP 自动将长消息分段,使之可以通过电子邮件发送。分段在所有其他操作之后进行,包括基数 64 转换。因此,会话密钥和签名部分仅在第一段的段首出现。在接收方,PGP 必须剥掉所有的电子邮件头,并按图 15.2(b)的步骤组装。

15.1.3 加密密钥和密钥环

PGP 使用四种类型的密钥:一次性会话对称密钥、公钥、私钥和基于对称密钥的口令。这些密钥需要三种需求:

1. 需要生成不可预测的会话密钥。
2. 希望能允许用户拥有多个公钥/私钥对。因为用户可能希望能经常更换他(她)的密钥对。而当更换时,许多流水线中的消息往往仍使用已过时的密钥。另外,接收方在更新到达之前只知道旧的公钥。为了能改变密钥,用户希望在某一时刻拥有多对密钥与不同的人进行应答,或限制用一个密钥加密消息的数量,以增强安全性。所有这些情况导致了用户与公钥之间的应答关系不是一对一的,因此,需要能鉴别不同的密钥。
3. 每个 PGP 实体必须管理一个自己的公钥/私钥对的文件和一个其他用户公钥的文件。

下面我们依次讨论各个需求。

会话密钥的产生

每个会话密钥与一个消息对应,并加密和解密该消息。回想消息加密/解密使用的对称加密算法:CAST-128 和 IDEA 使用 128 位密钥,3DES 使用 168 位密钥。在以下的讨论中,假设采用 CAST-128 算法。

使用 CAST-128 产生 128 位的随机数,随机数生成器的输入包括一个 128 位的密钥和两个 64 位待加密的明文块。使用密码反馈机制,CAST-128 产生两个 64 位的密文块,它们串联起来构成 128 位的会话密钥。此算法基于 ANSI X12.17 说明文档。

输入随机数生成器的明文由两个 64 位块组成,来源于一个 128 位的随机数据流。这些数据是用户击键产生的,击键的时间和击键本身用于生成随机流。因此,如果用户以其正常速度击键的话,就会得到合理的随机输入,将此随机输入与以前 CAST-128 产生的会话密钥组合得到生成器的输入。因此,不规则的 CAST-128 算法将产生一系列不可预测的会话密钥。

附录 15C 详细地叙述了 PGP 随机数生成技术。

密钥标识

如上所述,加密后的消息将与加密消息使用的会话密钥的密文一起传送。会话密钥将使用接收方的公钥加密,而只有接收方可以恢复会话密钥从而进一步恢复消息。如果每个用户

只使用一个公钥/私钥对，则接收方可以自动知道该用哪个密钥解密会话密钥：接收方惟一的私钥。然而，如何满足给定用户可以拥有多个公钥/私钥对的需求呢？

那么，接收方如何知道使用的是哪一个公钥加密会话密钥的呢？一个简单的解决方案是将公钥和消息一起传送。于是，接收方可以通过验证来确定使用的公钥。这种方式可以工作，但却浪费了不必要的空间。一个 RSA 的公钥可以长达几百个十进制数。另一种解决方案是每个用户的不同公钥与惟一的标识一一对应，即用户标识和密钥标识组合足以惟一标识一个密钥。这时，只需传送较短的密钥标识即可。但这个方案产生了管理和开销问题：密钥标识必须确定并存储，使发送方和接收方能获得密钥标识和公钥间的映射关系。

PGP 采用的解决方案是给每个公钥分配一个密钥标识，并在很大的概率上与用户标识一一对应。[①] 与公钥相关的密钥标识至少为 64 位，即公钥 KU_a 的密钥标识重复的可能性为（$KU_a \bmod 2^{64}$），这种可能性非常小。

PGP 的数字签名也需要使用密钥标识。因为发送方需要使用一个私钥加密消息摘要，接收方必须知道应使用哪个公钥解密。相应地，消息的数字签名部分必须包括公钥对应的 64 位密钥标识。当接收到消息后，接收方用密钥标识指示的公钥验证签名。

由于引进了密钥标识的概念，我们可以进一步讨论传递的消息格式，如图 15.3 所示。消息由三个部分组成：消息部分、签名部分（可选）和会话密钥部分（可选）。

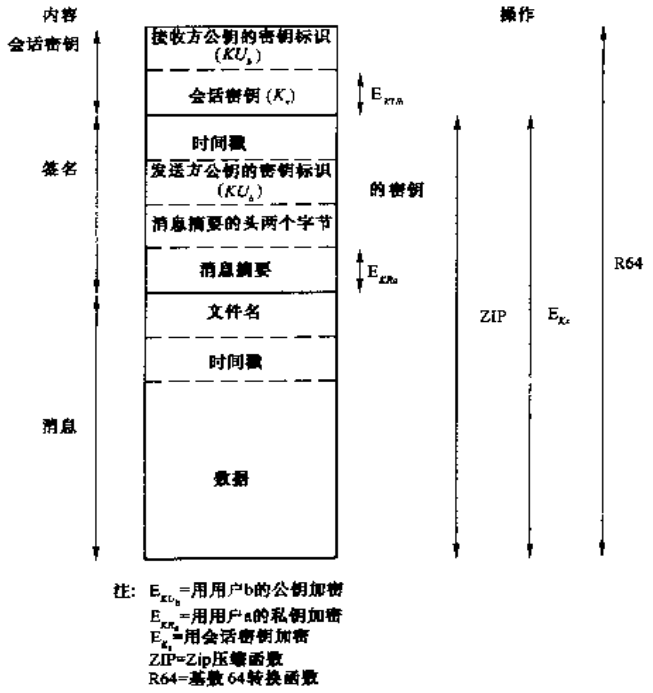


图 15.3　PGP 消息（从 A 到 B）的一般格式

消息部分包括将要存储或传输的实际数据，以及文件名、指明文件创建时间的时间戳等。

① 早在 8.4 节，我们就看到了概率性的概念，其时用概率性的算法检验一个数是否为素数。在设计算法时，经常使用概率技术降低时间消耗，降低复杂性，或同时降低时间消耗和复杂性。

签名部分包括如下内容：

- **时间戳**：签名时的时间。
- **消息摘要**：160 位的 SHA-1 摘要，用发送方的私钥加密。摘要是计算签名时间戳和消息的数据部分得到的。摘要中包含的时间戳可以防止重播攻击。不包括消息部分的文件名和时间戳保证了分离后的签名与分离前的签名一致。分离的签名基于单独的文件计算，该文件不包含消息头。
- **消息摘要的头两个字节**：为使接收方能够判断是否使用了正确的公钥解密消息摘要，可以通过比较明文中的头两个字节和解密后摘要中的头两个字节。这两个字节也可作为消息的 16 位帧校验序列。
- **发送方公钥的密钥标识**：标识解密所应使用的公钥，从而标识加密消息摘要的私钥。

消息和可选的签名可以使用 ZIP 压缩后再用会话密钥加密。

会话密钥部分包括会话密钥和发送方加密会话密钥时所使用的接收方公钥的标识。

整个块使用基数 64 转换编码。

密钥环

密钥标识对 PGP 操作是关键的，任何 PGP 消息中包含的两个密钥标识可以提供保密性和认证功能。这些密钥必须采用有效且系统的方式存储、组织，以供各方使用。PGP 为每个节点提供一对数据结构，一个用于存放本节点自身的公钥/私钥对，另一个用于存放本节点知道的其他用户的公钥，即私钥环和公钥环。

图 15.4 列出了**私钥环**的一般结构，我们可以将环看成是一个表结构，其中每一行表示用户拥有的一对公/私密钥。每一行包含如下内容：

私钥环

时间戳	密钥标识*	公钥	加密的私钥	用户标识*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	用户 i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

公钥环

时间戳	密钥标识*	公钥	信任度	用户标识*	合法密钥	签名	签名信任度
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$KU_i \bmod 2^{64}$	KU_i	$trust\_flag_i$	用户 i	$trust\_flag_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = 使用索引表的域

图 15.4 公钥环和私钥环的一般结构

- **时间戳**：密钥对生成的日期/时间。
- **密钥标识**：至少 64 位的公钥标识。
- **公钥**：密钥对的公钥部分。
- **私钥**：密钥对的私钥部分，此域被加密。
- **用户标识**：一般使用用户的电子邮件地址（如 stallings@acm.org）。但用户可以为不同密钥对选择不同的用户标识（如 Stallings、WStallings、WilliamStallings 等），也可以多次重复使用同一个用户标识。

私钥环可用用户标识或密钥标识索引；稍后我们将讨论这两种索引。

虽然私钥环只在用户创建和拥有密钥对的机器上存储并只能被该用户存取，但私钥的存储应尽可能地安全。因此，私钥并不直接存储在密钥环中，而是用 CAST-128（或 IDEA、3DES）加密后存储。处理过程如下：

1. 用户选择加密私钥的口令。
2. 当系统使用 RSA 生成新的公钥/私钥对后，向用户询问口令。使用 SHA-1 为口令生成 160 位的 hash 码，并废弃口令。
3. 系统用 CAST-128 和作为密钥的 128 位 hash 码加密私钥，并废弃该 hash 码，将加密后的私钥存于私钥环。

接着，当用户从私钥环中重新取得私钥时，他必须提供口令。PGP 将生成口令的 hash 码，并用 CAST-128 和 hash 码一起解密私钥。

这是一种非常紧凑而有效的模式。在任何基于口令的系统中，系统的安全性依赖于口令的安全性。为了避免将口令写出来，用户应使用不易猜测但容易记忆的口令。

图 15.4 也描述了**公钥环**的结构。此数据结构用来存储该用户知道的其他用户的公钥。以下仅讨论其中的部分域：

- **时间戳**：该项生成的日期/时间。
- **密钥标识**：该项至少 64 位的公钥标识。
- **公钥**：该项的公钥部分。
- **用户标识**：标识公钥的拥有者。多个用户标识可与一个公钥相关。

公钥环可以通过用户标识或密钥标识索引，稍后进行讨论。

现在，我们将论述如何在消息传递和接收中使用密钥环。为简化论述，这里忽略压缩和基数 64 转换。首先考虑消息传递（如图 15.5 所示），假设应对消息进行签名和加密，则发送的 PGP 实体执行下列步骤：

1. 签名消息：
   a. PGP 以用户标识作为索引从发送方的私钥环中取出选定的私钥，如果在命令中不提供用户标识，则取出私钥环中的第一个私钥。
   b. PGP 提示用户输入口令以恢复私钥。
   c. 创建消息的签名。
2. 加密消息：
   a. PGP 生成会话密钥并加密消息。

b. PGP 用接收方的用户标识作为索引，从公钥环中获得接收方的公钥。

c. 创建消息的会话密钥。

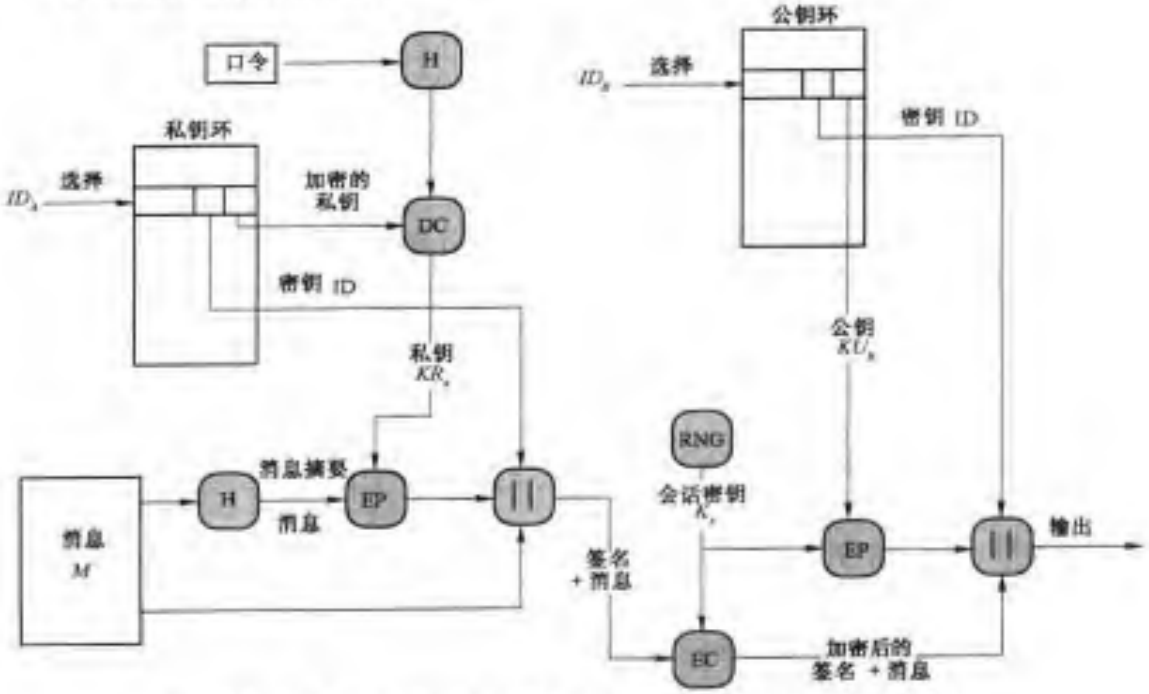


图 15.5 PGP 消息生成(从用户 A 到用户 B,不包括压缩和基数 64 转换)

接收方 PGP 实体执行下列步骤(如图 15.6):

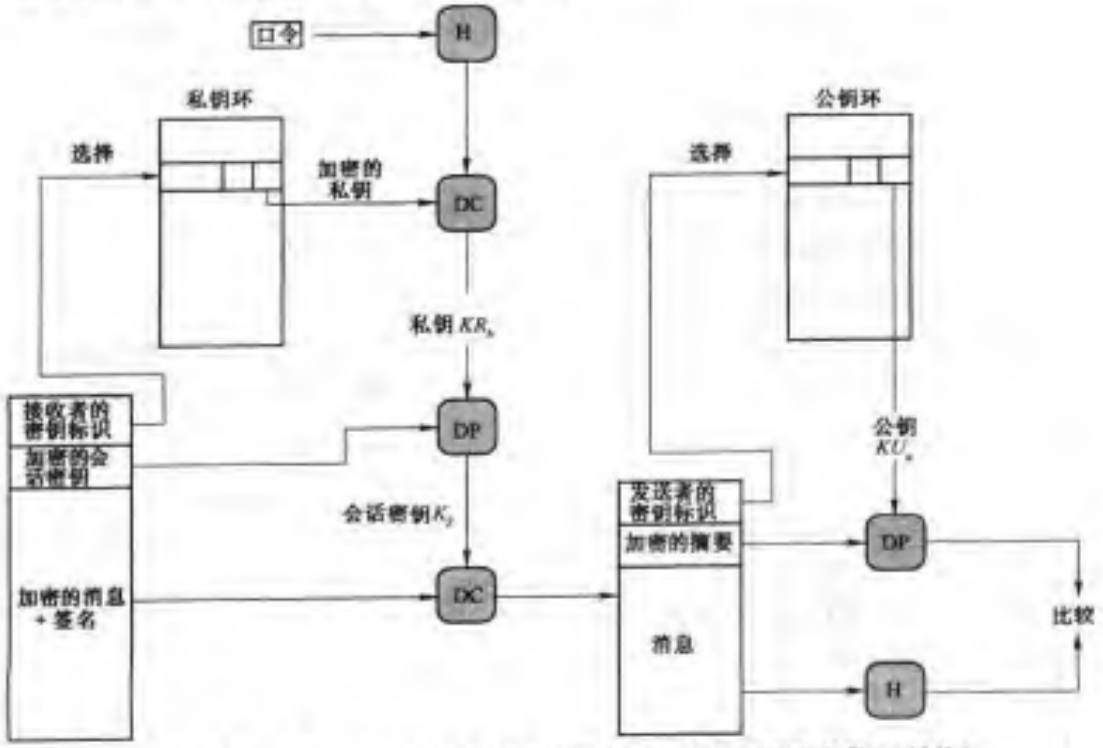


图 15.6 PGP 消息接收(从用户 A 到用户 B,不包括压缩和基数 64 转换)

1. 解密消息：
   a. PGP 用消息的会话密钥的密钥标识域作为索引，从私钥环中获取接收方的私钥。
   b. PGP 提示用户输入口令以恢复私钥。
   c. PGP 恢复会话密钥，解密消息。
2. 认证消息：
   a. PGP 用消息的签名密钥中包含的密钥标识，从公钥环中获取发送方的公钥。
   b. PGP 恢复消息摘要。
   c. PGP 计算接收到的消息的消息摘要，并将其与恢复的消息摘要进行比较来认证。

15.1.4　公钥管理

如上所述，PGP 包含一组精巧的、有效的、连锁的功能和形式，以提供保密和认证功能。为了完善该系统，还需要对公钥管理进行论述。PGP 文档描述了这部分的重要性：

> 在实际的公钥应用中，防止公钥的篡改是最困难的问题之一，有许多公钥密码机制和软件的复杂性都是为了解决这一问题而引入的。

PGP 提供了解决此问题的一种结构，并有各种可选方案。PGP 可用于许多正式和非正式环境中，没有 S/MIME 拥有的严格公钥管理模式。

公钥管理的方法

公钥管理的实质是：用户 A 为了与其他用户用 PGP 互操作，必须建立一个拥有其他用户公钥的公钥环。假设 A 的公钥环中包含一个属于 B 的公钥，但该公钥实际上是 C 的公钥。例如，A 从 B 张贴公钥的公告牌系统(BBS)获得了一个公钥，但该公钥是 C 伪造的，则此时 A 就会以为该公钥是属于 B 的。这样，就存在如下两种威胁：其一，C 向 A 发送消息且伪造 B 的签名，这样 A 就会以为该消息来源于 B；其二，任何 A 发往 B 的加密消息 C 均可以阅读。

有许多方法可以减少用户公钥环中包含错误公钥的可能性。假设 A 想获得 B 的可靠的公钥，可使用如下方法：

1. 物理上从 B 获得密钥：B 可以将自己的公钥(KU_b)存放于一张软盘上，并将其交给 A，A 再从软盘上将该密钥复制到系统中。这种方法虽然非常安全，但也有明显的局限性。
2. 利用电话验证密钥：如果 A 可以通过电话与 B 联系，A 可以直接通过电话向 B 询问基数 64 格式的密钥。或者，B 可以通过电子邮件传递密钥给 A，A 再用 PGP 为该密钥用 SHA-1 生成 160 位的摘要，并用十六进制表示作为密钥的指纹。于是 A 通过电话向 B 询问密钥的指纹，如果相符，则密钥得到了校验。
3. 从共同信任的个体 D 处获得 B 的公钥。引进 D 创建签名证书。证书包括 B 的公钥、密钥的创建时间，并用 D 的私钥加密，将签名放入证书。由于只有 D 可以创建签名，其他人不可能伪造公钥并假装 D 签名，因此，签名证书可由 B 或 D 直接送往 A，或发布在公告牌上。
4. 从信任的认证机构中获取 B 的公钥。同样创建公钥证书，并由认证机构签名，A 可以向认证机构提供用户名并接收签名证书。

对第三种、第四种方案，A 必须已经拥有了第三方的公钥并相信密钥的合法性。从根本上说，取决于 A 对第三方的信任程度。

信任关系

虽然 PGP 不包括建立认证机构或建立信任的机制，但它提供了使用信任的便利手段，利用信任信息将公钥与信任相联系。

基本结构如下。公钥环中的每项是一个公钥证书，如前所述。与各证书相关的是**密钥合法性域**使得 PGP 相信该用户的公钥是正确的。信任级别越高，用户标识与密钥间的绑定关系越强。此域的值由 PGP 计算。每个签名有一个**签名信任域**，该域表示 PGP 用户信任该签名的程度。与每个实体相关的是密钥环拥有者可以收集零个或多个签发证书的签名。密钥合法性域来源于该项中的签名信任域。最后，每个表项定义一个与特定所有者相关的公钥，**所有者信任域**包含对本公钥签名的其他公钥证书的信任程度，此信任级别由用户设定。我们可以认为签名信任域是所有者信任域的一个备份。

上述的三个域各包含一个信任标志字节，标志的内容如表 15.2 所示。假设处理的是用户 A 的公钥环，则信任操作的处理流程如下：

表 15.2 信任标志字节的内容

(a)赋给公钥所有者的信任值（在密钥包后，由用户定义）	(b) 赋给公钥/用户标识的信任值(在用户标识后，由 PGP 计算)	(c) 赋给签名的信任值(在签名包后，为此签名者缓存 OWNERTRUST 的备份)
OWNERTRUST 域 —未定义信任 —未知用户 —通常不信任对其他密钥的签名 —通常信任对其他密钥的签名 ——直信任对其他密钥的签名 —该密钥出现在秘密密钥环中(绝对信任) BUCKSTOP 位 —若密钥出现在秘密密钥环中，则置位	KEYLEGIT 域 —未知或未定义信任 —密钥所有者身份不被信任 —密钥所有者身份一般信任 —密钥所有者身份绝对信任 WARNONLY 位 —当用户想在使用的加密密钥不完全合法时提出警告，则置位	SIGTRUST 位 —未定义信任 —未知用户 —通常不信任对其他密钥的签名 —通常信任对其他密钥的签名 ——直信任对其他密钥的签名 —该密钥出现在秘密密钥环中(绝对信任) CONTIG 位 —如果签名在信任证书链中被证实，则置位

1. 当 A 向公钥环中插入一个新的公钥时，PGP 必须为该公钥的所有者设定一个信任标志。如果所有者为 A，则该公钥必然也出现于私钥环中，因此信任域中的值自动设为 ultimate trust。否则，PGP 询问 A 应为该标志置何值，A 必须键入相应的信任级别。用户可将该公钥的所有者设为 unknown、untrusted、marginally trusted 或 completely trusted。
2. 当新的公钥进入环中，必须与一个或多个签名相联系，多个签名可逐个加入。当插入一个签名时，PGP 将搜索公钥环，看该签名的作者是否是已知公钥的所有者。如果是，则该所有者的 OWNERTRUST 域值置为 SIGTRUST；否则，置为 unknown user。
3. 根据表项中签名信任域的值计算密钥合法性域。如果至少有一个签名的签名信任值为 *ultimate*，则将密钥合法性域置为 complete。否则，PGP 计算加权的信任值。A 将 always trusted 的签名权值置为 $1/X$，将 usually trusted 的签名权值置为 $1/Y$，X 和 Y 是用户设置的参数。当第三方的密钥/用户标识组合的权值达到 1 时，则认为该绑定是值得信

任的，并将密钥合法性域置为 complete。因此在没有 ultimate trusted 时，至少需要 *X* 签名的 always trusted 或 *Y* 签名的 usually trusted 或其他组合消息。

PGP 周期性地处理公钥环来获得一致性。本质上，这是一个自顶向下的过程。对每个 OWNERTRUST 域，PGP 扫描整个环得到该所有者的全部签名，并根据 OWNERTRUST 域来更新 SIGTRUST 域。此过程起始于具有 ultimate trust 值的密钥，接着计算与签名相关的所有 KEYLEGIT 域。

图 15.7 提供了一个签名信任和密钥合法性相关的例子。[①] 图中描述了一个公钥环的结构，用户从它们的所有者或作为密钥服务器的第三方获得了若干公钥。

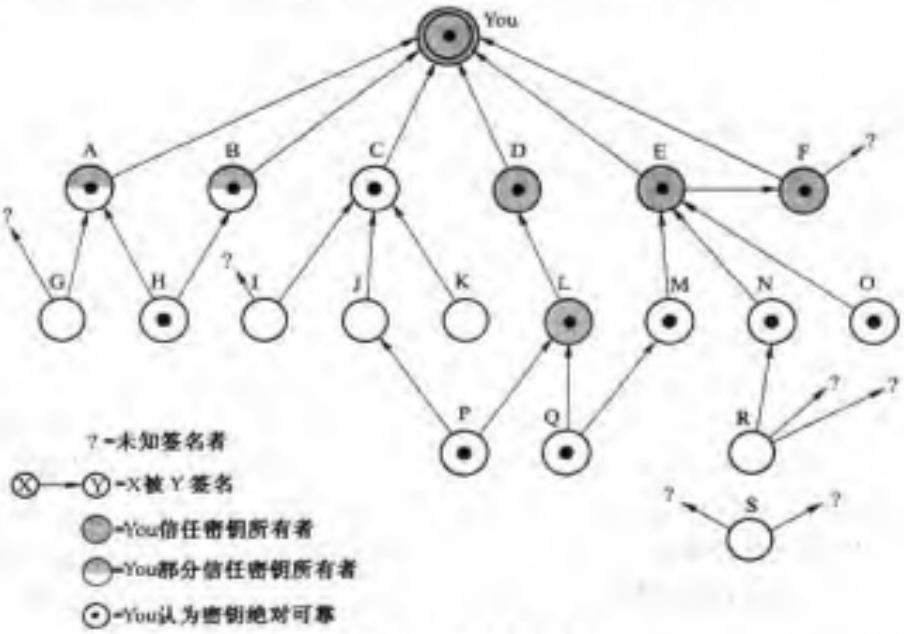


图 15.7　PGP 的信任模型示例

标识为"You"的节点表示公钥环中该项所属的用户。该密钥是合法的，且其域 OWNERTRUST 的值为 ultimate trust。公钥环中的其他节点在用户没有赋值前，域 OWNERTRUST 的值为 undefined。此例中，该用户通常相信用户 D、E、F、L 签名的密钥，并部分相信 A、B 签名的密钥。

从图 15.7 也可以看出该用户的信任级别。树结构表示出其他用户签名的密钥，如果一个对密钥进行签名的用户，其密钥也在这个密钥环中，则用箭头连接签名密钥和签名者。如果一个对密钥进行签名的用户，其密钥不在这个密钥环中，则用箭头连接签名密钥和问号"?"，表示该用户不知道签名者是谁。

图 15.7 说明了如下几点：

1. 除了节点 L 外，该用户全部信任和部分信任的密钥所有者的所有密钥均被签名。这种用户签名并不总是必要的（如节点 L），但实际上，大多数用户都会为其信任的所有者签名。因此，即使 E 的密钥是被信任的引进者 F 签名的，用户仍会选择直接对 E 的密钥进行签名。
2. 假设可以用双方信任的签名验证一个密钥，则当 A 和 B 同时部分信任对 H 的密钥签名时，PGP 认为此密钥合法。

① 该图由 Phil Zimmermann 提供给作者。

3. 如果一个密钥被一方完全信任的签名者或被两方部分信任的签名者签名，则认为该密钥合法，但可能在签名其他密钥时不被信任。例如，由于该用户信任的 E 签名 N 的密钥，使得 N 的密钥合法。但由于该用户并没有赋信任值给 N，该用户不相信 N 对其他密钥签名。因此，虽然 R 的密钥由 N 签名，PGP 认为 R 的密钥是不合法的。这种情况产生了很好的效果。如果你想向某人发送私人消息，你并不需要在任何方面都信任他，而只需要确信你拥有他的公钥。
4. 图 15.7 也说明了一个拥有两个 unknown 标识的孤立节点 S。这种密钥可能来自密钥服务器。PGP 并不会因为它来源于有信誉的服务器就简单地认为它合法，用户必须对它签名或告知 PGP 它希望信任该公钥的签名者来使该密钥合法。

最后要说明的是，前面提到可以用多个用户标识与公钥环中的一个公钥相关联。由于一个人可以改变名字，或使用不同的名字签名，或使用不同的 E-mail 地址，从而可能出现上述情况。因此，可以用公钥做为树的根，此树拥有多个用户标识，每个用户标识又拥有多个签名的公钥。公钥与特定用户标识的绑定关系可用依赖于公钥和用户标识的签名来区分，而该密钥的信任级别将依赖于所有相关的签名。

撤销公钥

用户可能因为防止攻击或在某一特定时期不想使用原来的密钥，而希望撤销当前使用的公钥。当攻击者获得了用户未加密的私钥或获得了私钥环中的私钥和口令时，用户必须撤销该公钥。

通常，撤销一个公钥需要所有者签发一个密钥撤销证书。该证书与一般的签名证书相同，并包括一个表明该证书的目的是用来撤销该公钥的指示信息，且必须用相应的私钥来签名该公钥撤销证书，然后所有者尽快广泛地散发该证书，使得相关用户可以更新他们的公钥环。

同样，一个获得所有者私钥的攻击者也可以发布一个这样的证书，但这会使得攻击者和合法所有者一样无法再使用该公钥，因此，这种恶意窃取私钥的威胁性不大。

15.2 S/MIME

S/MIME(Secure/Multipurpose Internet Mail Extension)在 RSA 数据安全性的基础上，加强了互联网电子邮件格式标准 MIME 的安全性。虽然 PGP 和 S/MIME 都是 IETF 工作组推出的标准，但 S/MIME 侧重于作为商业和团体使用的工业标准，而 PGP 则倾向于为许多用户提供个人电子邮件的安全性。RFC 为 S/MIME 定义了许多文档，重要的有 RFC 2630、RFC 2632 和 RFC 2633。

为了理解 S/MIME，我们首先要对它使用的电子邮件格式(即 MIME)有所了解。为了理解 MIME 的重要性，讲述仍在广泛使用的传统电子邮件格式标准 RFC 822。因此，本节介绍 RFC 822 和 MIME，接着讲述 S/MIME。

15.2.1 RFC 822

RFC 822 定义了一种用电子邮件传输的文本消息格式，是基于互联网传递的文本邮件消息标准，并被广泛使用。在 RFC 822 标准中，消息包括信封和内容两部分，信封包含用于传递和发送的任何信息，内容是指要发给接收方的对象。RFC 822 标准仅应用在内容上，而内容标

准包括一组用于邮件系统创建信封的报头域，以及简化程序获取该信息的步骤。

RFC 822 描述的消息结构非常简单。一个消息包含若干行的报头和无限制的正文。报头与正文之间用一个空行隔开，消息是 ASCII 码的文本，而从第一行到第一个空行的内容由邮件系统的用户代理使用。

报头通常由关键字、冒号":"和该关键字的参数组成，并允许一个长行被分割为若干短行，最常用的关键字为 From、To、Subject 和 Date。例如：

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 822
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is
delimited from the message heading by a blank line.
```

RFC 822 报头包含的另一个域是消息标识 Message-ID。该域包含与该消息相关的一个惟一标识。

15.2.2 MIME

MIME(Multipurpose Internet Mail Extensions)是对 RFC 822 框架的一个扩展，解决了使用 SMTP(Simple Mail Transfer Protocol)或其他邮件传输协议、RFC 822 电子邮件存在的一些问题和限制。[MURH98]列出了 SMTP/822 模式的如下限制：

1. SMTP 不能传输可执行文件或其他二进制对象。为了使用 SMTP 邮件系统，可以采用许多方式将二进制文件转化为文本格式，如流行的 UNIX UUencode/UUdecode 模式，但没有一个标准或事实上的标准。
2. SMTP 不能传递包括国际语言字符的文本数据，因为该字符集用 8 位描述，其值为 128 或更大，而 SMTP 限于 7 位 ASCII 码。
3. SMTP 服务器可以拒绝超过一定大小的邮件消息。
4. SMTP 网关在 ASCII 码和 EBCDIC 码之间转换时没有使用一致的映射。
5. 到 X.400 电子邮件网络的 SMTP 网关不能处理包含在 X.400 消息中的非文本数据。
6. 一些 SMTP 的实现与定义在 RFC 821 中的 SMTP 标准不完全一致。这些问题包括：
   - 回车和换行的增加、删除和重排
   - 长于 76 个字符的截断和转行
   - 去掉空白字符(制表符和空格符)
   - 将消息中的行填充为等长
   - 将制表符转换为若干空格符

MIME 期望以 RFC 822 实现兼容的方式解决以上问题，详细说明参见 RFC 2045 到 2049。

概述

MIME 说明书包含如下要素：

1. 定义了五个新的报头域，这些域提供正文信息（可能在 RFC 822 报头中定义）。
2. 定义了若干内容格式，从而标准化支持多媒体的电子邮件。
3. 定义了编码转换方式，使得任何内容格式均可转化为邮件系统认可的格式。

在本小节中，我们将介绍五个报头域，而在随后的两小节中介绍内容格式和编码转换。MIME 中定义的五个报头域如下：

- **MIME-Version（MIME 版本）**：其值必须为 1.0，表明该消息符合 RFC 2045 和 2046。
- **Content-Type（内容类型）**：描述正文中包含的数据类型，使得接收方代理可以选择合适的代理或机制表示数据或正确处理数据。
- **Content-Transfer-Encoding（内容转换编码）**：描述将消息正文转换为可以传输的类型转换方式。
- **Content-ID（内容标识）**：在多重上下文中惟一标识 MIME 实体的标识。
- **Content-Description（内容描述）**：正文对象的文本描述，在该对象不可读时使用（如音频数据）。

RFC 822 报头可以出现一个或多个域。在实现中必须支持 MIME-Version、Content-Type 和 Content-Transfer-Encoding 域，可以选择支持 Content-ID 和 Content-Description 域，并且接收方可以忽略这两个域。

MIME 内容类型

MIME 中有大量关于内容类型定义的说明，体现了在多媒体环境下需要提供多种信息表示方法的需求。

表 15.3 列出了 RFC 2046 中说明的内容类型，主要有 7 种不同的类型和 15 种子类型。一般来说，用内容类型描述数据的通用类型，用子类型描述该数据类型的特殊格式。

表 15.3 MIME 内容类型

类型	子类型	描述
Text	Plain	无格式文本，为 ASCII 码或 ISO 8859 码
	Enriched	提供丰富的文档信息
Multipart	Mixed	各部分相互独立但一起传送。接收方看到的形式与邮件消息中的形式相同
	Parallel	除在传送时各部分无序外，其余与 Mixed 相同
	Alternative	不同部分是同一消息的不同表现形式，按增序排列，接收方系统将按照最佳方式显示
	Digest	与 Mixed 相似，但每部分默认的类型/子类型为 message/rfc822
Message	rfc822	封装消息的正文与 RFC 822 一致
	Partial	允许按接收方透明的方式对大邮件分段
	External-body	包含一个对存在于别处的对象的指针
Image	jpeg	JFIF 编码的 JPEG 图像
	gif	图像为 GIF 格式
Video	mpeg	MPEG 格式
Audio	Basic	单通道 8 位 ISDN μ 律，8 kHz 采样率
Application	PostScript	Adobe PostScript
	Octet-stream	一般的 8 位二进制数据

若正文为**文本（text）类型**，则除了要支持指定字符集外不需要特殊的软件来获取文本的全部含义，因此主要的子类型为纯文本（plain text），即由简单的 ASCII 码或 ISO 8859 字符组成的

字符串。子类型 enriched 则允许拥有更多的格式信息。

类型为**多部分(multipart)**时,表示正文中包含多个相互独立的部分。域 Content-Type 中包含一个称为分界符的参数,该分界符定义了正文中各部分的分隔符。此分界符不能随意出现,而必须从一个新行开始,并在两个连字符后跟分界符值,最后一个分界符表示最后一个部分的结尾,并有两个连字符做后缀。在每部分,可以有一个通常的 MIME 报头。

以下是一个多正文消息的简单例子,包含由简单文本组成的两个部分(见 RFC 2046):

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Sample message
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="simple boundary"

This is the preamble. It is to be ignored, though it is a handy
place for mail composers to include an explanatory note to non-
MIME conformant readers.--simple boundary

This is implicitly typed plain ASCII text. It does NOT end with a
linebreak.--simple boundary
Content-type: text/plain; charset=us-ascii

This is explicitly typed plain ASCII text. It DOES end with a
linebreak.

--simple boundary--
This is the epilogue. It is also to be ignored.
```

Multipart 类型有四种子类型,其语法相同。**子类型 multipart/mixed** 用于将相互独立的各部分按一定的顺序绑定。**子类型 multipart/parallel** 表示各部分的顺序无关紧要。如果接收系统合适,则各部分可并行接收。例如,在图片或文本显示时,可同时播放音频。

子类型 multipart/alternative 表示这若干个部分是同一个信息的不同表示。例如:

```
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Formatted text mail
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary=boundary42
--boundary42

Content-Type: text/plain; charset=us-ascii

... plain text version of message goes here ....

--boundary42
Content-Type: text/enriched

.... RFC 1896 text/enriched version of same message goes here ...
--boundary42--
```

在此子类型中，各部分按优先级递增的方式排序。例如，如果接收方可以处理 text/enriched 格式的信息，则按此格式处理，否则就使用纯文本方式。

子类型 multipart/digest 用于将正文的每个部分作为一个带有头的 RFC 822 消息，从而使得一个消息中可以包含多个独立的消息。例如，可以用一个组缓冲器收集组中各成员的电子邮件消息并将其封装到一个 MIME 消息中发送。

类型 message 提供了许多 MIME 的重要特性。**子类型 message/rfc822** 表明该正文是一个完整的消息，包括了头和正文。除子类型的名字外，封装的消息既可以是 RFC 822 消息，也可以是任何 MIME 消息。

子类型 message/partial 使得可以将一个大消息分段为若干部分，并可在目的地重新组装。对这个子类型而言，Content-Type: Message/Partial 域需要说明三个参数：一个对同一个消息所有分片一致的标识 id、一个每段惟一的序列号 sequence number 和总的段数 total。

子类型 message/external-body 表示消息中需要转换但不包含在正文中的实际数据，而正文中包含了存取该数据所需的信息。与其他消息类型一样，此子类型有一个外部报头和封装在消息内的报头。外部头仅需要 Content-Type 域，该域表示消息/外部头的此子类型。内部头是该封装消息的消息头。域 Content-Type 中必须包含一个描述存取方式的类型参数，如 FTP。

类型 application 指的是其他类型的数据，如不能解释的二进制数据或由邮件应用程序处理的信息。

MIME 转换编码

MIME 说明书中另一个主要部分是定义消息正文的转换编码。其目的是在庞大的网络环境中提供可靠的投递方式。

MIME 标准定义了两种编码方式，但域 Content-Transfer-Encoding 可以取六个值，如表 15.4 所示。其中当值为 7 bit、8 bit 和 binary 时，并不进行编码，而是提供一些与数据属性相关的信息。对 SMTP 传输而言，用 7 bit 较安全，而在其他邮件传输协议中可以使用 8 bit 和 binary。Content-Transfer-Encoding 域也可取值 x-token，用来表示其他编码方式，并提供该方式的名字。这种方式可以是销售商定义的或应用程序自定义的方式。目前，有两种方式：quoted-printable 和 base64，它们都是为了提供一种将所有数据类型的紧凑表示转换为便于人类阅读形式而设计的。

表 15.4 MIME 转换编码

7 bit	所有数据都用短行的 ASCII 码字符表示
8 bit	短行，但可以有非 ASCII 码字符（高位位的 8 位）
binary	不仅可以有非 ASCII 码字符，且可以存在 SMTP 无法传送的长行
quoted-printable	如果被编码的数据大多是 ASCII 码字符，则数据编码后的形式大部分仍是能被人识别的一种数据编码方式
base64	将 6 位输入转为 8 位输出的方式，转换后的数据均为可打印的 ASCII 码字符
x-token	非标准编码方式的名称

quoted-printable 转换编码在数据由大量可打印的 ASCII 字符组成时使用。其本质上是一种不安全的 16 进制字符表示方法，并引入软回车来解决每行不得超过 76 个字符的限制。

base64 转换编码是一种基数 64 的编码方法，是一种对二进制数据进行编码的通用方法，在邮件传输程序中无懈可击。也用于 PGP，附录 15B 中有详细叙述。

一个多部分的例子

图 15.8(见 RFC 2045)描述了一个复杂的多部分消息。该消息有依次出现的五个部分:两个介绍性的纯文本,一个嵌入的多部分消息,一个 richtext 部分和一个封装了非 ASCII 字符集的文本消息。其中,嵌入消息由两个需要并行播放的部分(图片和声音片断)组成。

```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: A multipart example
Content-Type: multipart/mixed;
    boundary=unique-boundary-1

This is the preamble area of a multipart message. Mail readers that understand multipart format should ignore
this preamble. If you are reading this text, you might want to consider changing to a mail reader that understands
how to properly display multipart messages.

--unique-boundary-1

   ...Some text appears here...
[Note that the preceding blank line means no header fields were given and this is text, with charset US ASCII.
It could have been done with explicit typing as in the next part.]

--unique-boundary-1
Content-type: text/plain; charset=US-ASCII

This could have been part of the previous part, but illustrates explicit versus implicit typing of body parts.

--unique-boundary-1
Content-Type: multipart/parallel;   boundary=unique-boundary-2

--unique-boundary-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

   ... base64-encoded 8000 Hz single-channel mu-law-format audio data goes here....

--unique-boundary-2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

   ... base64-encoded image data goes here....

--unique-boundary-2--

--unique-boundary-1
Content-type: text/enriched

This is <bold><italic>richtext.</italic></bold> <smaller>as defined in RFC 1896</smaller>

Isn't it <bigger><bigger>cool?</bigger></bigger>

--unique-boundary-1
Content-Type: message/rfc822

From: (mailbox in US-ASCII)
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

   ... Additional text in ISO-8859-1 goes here ...

--unique-boundary-1--
```

图 15.8　MIME 报文格式

规范格式

MIME 和 S/MIME 有一个规范格式的重要概念。规范格式指的是一种与内容类型相对应的格式,可在系统中作为标准使用。表 15.5(摘自RFC 2049)可阐明此问题。

表 15.5 本地格式和规范格式

本地格式	被传送的正文部分是按系统的本地格式创建的,使用本地字符集和行结束标记。正文可以是 UNIX 风格的文本文件、Sun 光栅图片或 VMS 索引文件,或依赖于系统的仅存储在内存中的声音文件等任何信息。从根本上说,数据依据媒体类型按照本地格式创建
规范格式	整个正文部分,包括外部信息如记录长度和可能的文件属性信息,均转化为一致的规范格式。正文中特定媒体类型和其属性将按本地格式使用。将其转换为合适的本地格式将涉及到字符集的转换、声音数据的转换、压缩或其他与媒体类型相关的操作。如果涉及到字符集的转换,则应注意理解与字符集密切相关的媒体类型的语义(如,text 子类型中的 plain 和其他含义丰富的字符集)

15.2.3 S/MIME 的功能

就一般功能而言,S/MIME 与 PGP 非常相似。两者都提供了签名和加密消息的能力。在本节中,我们将简要介绍 S/MIME 的功能,然后再从消息格式和消息准备方面详细叙述各功能。

功能

S/MIME 提供如下功能:

- **封装数据**:由任何类型的加密内容和加密该内容所用的加密密钥组成,密钥可以是与一个或多个接收方对应的多个密钥。
- **签名数据**:数字签名通过提取待签名内容的数字摘要,并用签名者的私钥加密得到。然后,用 base64 编码方法重新对内容和签名编码。因此,一个签名了的数据消息只能被具有 S/MIME 能力的接收方处理。
- **透明签名数据**:签名的数据形成了内容的数字签名。但在这种情况下,只有数字签名采用了 base64 编码。因此,没有 S/MIME 功能的接收方虽然无法验证签名,但却可以看到消息内容。
- **签名并封装数据**:仅签名实体和仅封装实体可以嵌套,以便对加密后的数据进行签名,以及对签名数据或透明签名数据进行加密。

密码算法

表 15.6 总结了在 S/MIME 中使用的密码算法。S/MIME 中使用了如下术语(摘自 RFC 2119)来说明需求级别:

表 15.6　S/MIME 中使用的密码算法

功　能	要　求
创建用于数字签名的数字摘要	必须支持 SHA-1，接收方应该支持 MD5，以便向后兼容
加密数字摘要形成数字签名	发送代理和接收代理必须支持 DSS 发送代理应该支持 RSA 加密 接收代理应该支持验证密钥大小在 512 位至 1024 位的 RSA 签名
为传送消息加密会话密钥	发送代理和接收代理必须支持 Diffie-Hellman 发送代理应该支持密钥大小在 512 位至 1024 位的 RSA 加密 接收代理应该支持 RSA 解密
用一次性会话密钥加密消息	发送代理应该支持 3DES 和 RC2/40 加密 接收代理必须支持用 3DES 解密，应该支持 RC2/40 解密

- **MUST**：在规格说明书中表示一定要满足的需求。实现时，必须包括其修饰的特性或与规格说明中的功能一致。
- **SHOULD**：如果在特定条件下有合理的理由，可以忽略其修饰的特性或功能，但推荐其实现包含其特性或功能。

S/MIME 组合三种公钥算法。第 13 章的 DSS(Digital Signature Standard)是其推荐的数字签名算法。Diffie-Hellman 是其推荐的密钥交换算法；实际上，在 S/MIME 中使用的是其能加密解密的变体 ElGamal(参见习题 6.19)。第 9 章讲述的 RSA 既可以用做签名，也可以加密会话密钥。这些算法与 PGP 中使用的算法相同，从高层提供了其安全性。规格说明推荐使用 160 位的 SHA-1 算法作为数字签名的 hash 函数，但要求接收方能支持 128 位的 MD5 算法。第 12 章对 MD5 安全性的讨论指出 SHA-1 的安全性更好一些，但由于 MD5 已经有了广泛的应用，因此必须提供支持。

对消息加密而言，推荐使用 3DES。但实现时也应支持 40 位的 RC2，后者是一种弱加密算法，美国允许出口该算法。

S/MIME 规格说明包括如何决定采用何种加密算法。从本质上说，一个发送方代理需要进行如下两个抉择：一是，发送方代理必须确定接收方代理是否能够解密该加密算法。二是，如果接收方代理只能接收弱加密的内容，发送方代理必须确定弱加密方式是否可以接受。为达到上述要求，发送方代理可以在它发送消息之前先宣布它的解密能力，由接收方代理将该消息存储，留给将来使用。

发送方代理必须按照如下顺序遵守如下规则：

1. 如果发送方代理有一个接收方解密性能表，则它应该选择表中的第一个(即优先级最高的)性能。
2. 如果发送方代理没有接收方的解密性能表，但曾经接收到一个或多个来自于接收方的消息，则应该使用与最近接收到的消息一样的加密算法，加密将要发送给接收方的消息。
3. 如果发送方代理没有接收方的任何解密性能方面的知识，并且想冒险一试(接收方可能无法解密消息)，则应该选择 3DES。
4. 如果发送方代理没有接收方的任何解密性能方面的知识，并且不想冒险，则发送方代理必须使用 RC2/40。

如果消息需要发给多个接收方，并且它们没有一个可以接受的、共同的加密算法，则发送方代理需要发送两条消息。此时，该消息的安全性将由于安全性低的一份拷贝而易受到攻击。

15.2.4 S/MIME 消息

S/MIME 使用了一系列新的 MIME 内容类型，如表 15.7 所示。所有的新类型都使用 PKCS (Public-Key Cryptography Specifications)指示，PKCS 是纪念为 S/MIME 做出贡献的 RSA 实验室发布的一组公钥密码规格说明。

表 15.7 S/MIME 内容类型

类型	子类型	smime 参数	描 述
Multipart	Signed		两部分的透明签名消息：一部分是消息，另一部分是签名
Application	pkcs7-mime	签名数据	签名的 S/MIME 实体
	pkcs7-mime	封装数据	加密的 S/MIME 实体
	pkcs7-mime	退化的签名数据	仅包含公钥证书的实体
	pkcs7-signature	—	签名子部分的内容类型为 multipart/signed 的消息
	pkcs10-mime	—	证书注册请求消息

下面逐一介绍 S/MIME 消息处理过程。

保护 MIME 实体

S/MIME 用签名和/或加密保护 MIME 实体。一个 MIME 实体可以是一个整体消息(除 RFC 822 报头外)，或当 MIME 内容类型为 multipart 时，MIME 实体是一个或多个消息的子部分。MIME 实体将按照 MIME 消息准备规则进行准备工作，然后将 MIME 实体和一些与安全相关的数据(如算法标识符、证书)一起用 S/MIME 处理，得到所谓的 PKCS 对象。最后，将 PKCS 对象作为消息内容封装到 MIME 中(提供合适的 MIME 头)。后面我们将举例说明。

总之，将要发送的消息需要转换为规范形式。特别地，对给定的类型和子类型面言，相应的规范形式将作为消息内容。对一个多部分的消息而言，合适的规范形式被用于每个子部分。

使用编码转换时应注意，在大多数情况下，使用安全算法后将产生部分或全部用二进制数据表示的对象，将该对象放入外部 MIME 消息后，一般用 base64 转换编码对其进行转换。而对一个多部分签名的消息，安全处理过程并不改变子部分的消息内容，除了内容用 7 位表示的以外，其他代码转换应使用 base64 或 quoted-printable，使得应用签名的内容不会被改变。下面逐个讨论 S/MIME 内容类型。

封装数据

子类型 application/pkcs7-mime 拥有一个 smime-type 参数。其结果(对象)采用 ITU-T 推荐的 X.209 中定义的 BER(Basic Encoding Rules)表示法。BER 格式由 8 位字符串组成，即为二进制数据，因此，此对象可在外部 MIME 消息中用 base64 转换算法编码。首先，看看封装的数据。

MIME 实体准备封装数据的步骤如下：

1. 为特定的对称加密算法(RC2/40 或 3DES)生成伪随机的会话密钥。
2. 用每个接收方的 RSA 公钥分别加密会话密钥。

3. 为每个接收方准备一个接收方信息块(RecipientInfo),其中包含接收方的公钥证书标识、[①] 加密会话密钥的算法标识和加密后的会话密钥。
4. 用会话密钥加密消息内容。

接收方信息块(RecipientInfo)后面紧跟着构成封装数据的加密内容,然后用 base64 编码,例如(不包括 RFC 822 报头):

```
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
      name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
0GhIGfHfQbnj756YT64V
```

为了恢复加密的消息,接收方首先去掉 base64 编码,然后用其私钥恢复会话密钥。最后,用会话密钥解密得到消息内容。

签名数据

smime 类型的签名数据可以被一个或多个签名者使用。为了简单起见,我们将讨论范围限定在单个数字签名。MIME 实体准备签名数据的步骤如下:

1. 选择消息摘要算法(SHA 或 MD5)。
2. 计算待签名内容的消息摘要或 hash 函数。
3. 用签名者的私钥加密数字摘要。
4. 准备一个签名者信息块(SignerInfo),其中包含签名者的公钥证书、消息摘要算法的标识符、加密消息摘要的算法标识符和加密的消息摘要。

签名数据实体包含一系列块,包括一个消息摘要算法标识符、被签名的消息和签名者信息块(SignerInfo)。签名数据实体可以包含一组公钥证书,该证书可以构成一个从上级认证机构或更高级的认证机构证明该签名者的一条链路。最后将这些数据用 base64 转换编码。例如(不包括 RFC 822 报头):

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
       name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

① 这是一种 X.509 证书,本节后面还要讲到。

为了恢复签名消息并验证签名，接收方首先去掉 base64 编码，然后用签名者的公钥解密消息摘要，接收方独立计算消息摘要，并将其与解密得到的消息摘要进行比较，从而验证签名。

透明签名

透明签名在对多部分内容类型的子类型签名时使用。签名过程并不涉及对签名消息的转换，因此该消息发送时是明文。因此，具有 MIME 能力而不具备 S/MIME 能力的接收方也能阅读输入的消息。

一个 multipart/signed 消息由两部分组成。第一部分可以是任何 MIME 类型，但必须做好准备，使之在从源端到目的端的传送过程中不被改变，这意味着第一部分不能是 7 位，需要用 base64 或 quoted-printable 编码。而后，其处理过程与签名数据相同，但签名数据格式的对象中消息内容域为空，该对象与签名相分离，再将它用 base64 编码，作为 multipart/signed 消息的第二部分。第二部分的 MIME 内容类型为 application，子类型为 pkcs7-signature。例如：

```
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--boundary42--
```

协议的参数表明它是一个由两部分组成的透明签名实体。参数 micalg 表明使用的是消息摘要类型。接收方可以从第一部分获得消息摘要，并同第二部分中恢复得到的消息摘要进行比较来进行认证。

注册请求

典型地，一个应用或用户向认证中心申请公钥证书。S/MIME 的实体 application/pkcs10 用于传递证书请求。证书请求包括证书请求信息块、公钥加密算法标识符、用发送方私钥对证书请求信息块的签名。证书请求信息块包含证书主体的名字（拥有待证实的公钥的实体）和该用户公钥的标识位串。

仅证书消息

仅包含证书或证书撤销表(CRL)的消息在应答注册请求时发送。该消息的类型/子类型为 application/pkcs7-mime,并带一个退化的 smime-type 参数。其步骤除没有消息内容和签名者信息块为空以外,与创建签名数据消息相同。

15.2.5　S/MIME 证书处理过程

S/MIME 使用公钥证书的方式与 X.509 的版本 3 一致(参见第 14 章)。S/MIME 使用的密钥管理模式是严格的 X.509 证书层次和 PGP 的基于 Web 信任方式的一种混合方式。按照 PGP 模式,S/MIME 的管理者和/或用户必须配置每个客户端的信任密钥表和证书撤销表。也就是说,验证接收到的签名和对输出消息的签名工作都是通过本地维护证书实现的。另一方面,证书由认证机构颁发。

用户代理职责

一个 S/MIME 用户需要执行若干密钥管理职能:

- **密钥生成**:与一些行政管理机构相关的用户(如与局域网管理相关)必须能生成单独的 Diffie-Hellman 和 DSS 密钥对,并且应该能生成 RSA 密钥对。每个密钥对必须利用非确定的随机输入生成,并以安全的方式保护。用户代理应该能生成长度在 768 位到 1024 位的 RSA 密钥对,且禁止生成长度小于 512 位的 RSA 密钥对。
- **注册**:为了获得 X.509 公钥证书,用户的公钥必须到认证机构注册。
- **证书存储和检索**:为了验证接收到的签名和加密输出消息,用户需要存取本地的证书列表。该列表必须由用户自己维护,或一些本地行政实体为一部分用户维护。

VeriSign 证书

有不少公司都可以提供证书认证授权服务。例如,Nortel 设计了一种企业认证授权解决方案,能够在组织内部提供 S/MIME 支持。还有一些基于互联网的认证中心,包括 VeriSign、GTE 和 U.S. Postal Service。其中使用最广泛的是 VeriSign 认证服务。

VeriSign 提供与 S/MIME 和其他一系列应用兼容的一种认证授权服务,颁发称为 VeriSign 数字证书(VeriSign Digital ID)的 X.509 证书。到 1998 年初,已有 35 000 家商业 Web 站点使用 VeriSign 数字证书,一百多万使用 Netscape 和 Microsoft 浏览器的用户拥有 VeriSign 数字证书。

VeriSign 数字证书包含的内容依赖于数字证书的类型和它的用途。但每个数字证书至少包含如下内容:

- 用户公钥
- 用户名或别名
- 数字证书的有效期
- 数字证书的序列号
- 颁发数字证书的认证中心名
- 颁发数字证书的认证中心的数字签名

数字证书还可以包含其他用户提供的信息：

- 地址
- 电子邮件地址
- 基本注册信息(国家、邮政编码、年龄和性别)

VeriSign 提供了公钥证书的三种安全级别，如表 15.8 所示。用户可以通过在线方式向 VeriSign 的 Web 站点或其他相关站点申请证书。第一类和第二类请求可以在线处理，而且大多数只需要几秒钟就可以完成。下面将简述所使用的处理过程：

表 15.8 VeriSign 公钥证书类型

	证书确认	IA 私钥保护	证书申请者的私钥保护	用户实现或期望的应用
第 1 类	姓名和电子邮件地址的自动搜索	PCA:信任的硬件 CA:信任的硬件或软件	推荐使用加密软件(PIN 保护)	Web 浏览和使用电子邮件
第 2 类	第 1 类检查，登记信息检查和自动地址检查	PCA 和 CA:信任的硬件	要求使用加密软件(PIN 保护)	个人、公司内或公司间的电子邮件、在线定购、更换密码和软件确认
第 3 类	第 1 类检查、本人出席、ID 文档、第 2 类自动身份检查。组织、团体、的业务记录	PCA 和 CA:信任的硬件	要求使用加密软件(PIN 保护)。推荐使用硬件令牌	电子银行、公司数据存取、个人银行、基于会员的在线服务、内容集成服务、电子贸易服务、软件确认。LRAA 的认证、对服务的强加密

IA: Issuing Authority
CA: Certification Authority
PCA: VeriSign Public Primary Certification Authority
PIN: Personal Identification Number
LRAA: Local Registration Authority Administrator

- 对第一类数字证书而言，VeriSign 通过发送一个 PIN 命令和从应用中提取电子邮件地址确认用户的电子邮件地址。
- 对第二类数字证书而言，VeriSign 除了进行与第一类数字证书相同的检查外，还使用一个用户数据库自动比较应用中提供的信息。最后，向给定的邮件地址发送一个确认信息，告诉该用户已经按其名字颁发了一个数字证书。
- 对第三类数字证书而言，VeriSign 需要更高级的身份证实。个人必须提供有效证明来证实其身份。

15.2.6 增强安全性服务

互联网草案还提出了三种增强安全性服务，其细节可能会发生变动，也可能增加一些新的服务。这三种服务分别是：

- **签收**：对签名数据对象要求进行签收。返回一条签收消息可以告知消息的发送方已经收到消息，并通知第三方接收方已收到消息。本质上说，接收方将对整个原始消息和发送方的原始签名进行签名，并将此签名与消息一起形成一个新的 S/MIME 消息。
- **安全标签**：在签名数据对象的认证属性中可以包括安全标签。安全标签是一个描述被 S/MIME 封装的信息的敏感度的安全信息集合。该标签既可用于存取控制，描述该对象

能被哪些用户存取，还可描述优先级（秘密、机密、受限等）或角色（哪种人可以查看信息，如患者的病历等）。

- **安全邮寄列表**：当用户向多个接收方发消息时，需要进行一些与每个接收方相关的处理，包括使用各接收方的公钥。用户可以通过使用 S/MIME 提供的邮件列表代理（Mail List Agent）来完成这一工作。邮件列表代理可以对一个输入消息为各接收方进行相应的加密处理，而后自动发送消息。消息的发送方只需将用 MLA 的公钥加密过的消息发给 MLA 即可。

15.3　推荐网址


推荐网址：

- **PGP Home Page**：Network Associates 提供的 PGP Web 站点，PGP 主流销售商。
- **MIT Distribution Site for PGP**：免费 PGP 发放，包括 FAQ、其他信息和其他 PGP 站点的链接。
- **S/MIME Charter**：S/MIME 的最新 RFC 和 Internet 草案。
- **S/MIME Central**：S/MIME 的 RSA Inc 的 Web 站点。包括 FAQ 和其他有用信息。

15.4　关键术语、思考题和习题

15.4.1　关键术语

分离签名	PGP	S/MIME
电子邮件	R64	信任
MIME	会话密钥	ZIP

15.4.2　思考题

15.1　PGP 提供的 5 个服务是什么？
15.2　分离签名的用途是什么？
15.3　PGP 为什么在压缩前生成签名？
15.4　什么是 R64 转换？
15.5　电子邮件应用为什么使用 R64 转换？
15.6　PGP 为什么需要分段和重组？
15.7　PGP 如何使用信任关系？
15.8　RFC 822 是什么？
15.9　MIME 是什么？
15.10　S/MIME 是什么？

15.4.3　习题

15.1　PGP 使用了 CAST-128 的密文反馈模式（CFB），而传统的加密应用（不是指密钥加

密)使用密文块链接模式。我们有:

$$\text{CBC: } C_i = E_K[C_{i-1} \oplus P_i]; \qquad P_i = C_{i-1} \oplus D_K[C_i]$$
$$\text{CFB: } C_i = P_i \oplus E_K[C_{i-1}]; \qquad P_i = C_i \oplus E_K[C_{i-1}]$$

这两种模式看起来提供了相当的安全性,PGP 使用 CFB 的原因是什么?

15.2 在 PGP 体制中,在前一次会话密钥生成后,希望生成多少个会话密钥?

15.3 在 PGP 中,一个拥有 N 个公钥的用户有多大的可能性拥有至少一个重复的密钥 ID?

15.4 PGP 签名的 128 位消息摘要中的前 16 位是以明文方式解释的。

a. 这对 hash 算法的安全性有多大威胁?

b. 这对其设计功能有多少帮助? 也就是说,帮助判断解密摘要的 RSA 密钥是否正确。

15.5 图 15.4 中公钥环的每一项都有一个所有者信任域可以指明其所有者的信任度。它为什么不够用? 所有者是被信任的,而且这也是该所有者的公钥,为什么还不能被 PGP 足够信任使用这个公钥?

15.6 基数 64 转换是一种加密形式,它没有密钥。但假设对手知道某种加密文本的替代算法,这种算法对抗密码分析的能力如何?

15.7 Phil Zimmermann 选择了 IDEA、三密钥 3DES 和 CAST-128 作为 PGP 的对称钥加密算法。试列举本书所提及的其他加密算法在 PGP 中合适或不合适的原因,如 DES、二密钥 3DES、Blowfish、RC5。

附录 15A 用 ZIP 压缩数据

PGP 使用一个名为 ZIP 的压缩软件包,它是由 Jean-lup Gailly、Mark Adler 和 Richard Wales 编写的。ZIP 是一个运行于 UNIX 和其他系统上的免费软件包,用 C 语言编写。ZIP 的功能与 PKWARE 公司出品的、运行于 Windows 系统的共享软件 PKZIP 等效。ZIP 算法可能是全世界最流行的跨平台压缩算法,不论是 Windows 和 UNIX,还是 Macintosh 或者其他系统上,都有免费的或共享的软件。

ZIP 和类似算法是 Jacob Ziv 和 Abraham Lempel 研究出的。1977 年,他们描述了一种基于滑动窗口缓冲区的机制,存放最近处理过的文字[ZIV77]。这种算法通常称为 LZ77。ZIP 压缩体制使用了这种算法。

LZ77 及其变体假设一个文字流中的词或词组(GIF 中的图形模式)是有可能重复的。每当重复发生,重复的部分被一个较短的代码取代。压缩程序扫描重复的部分并动态构造代码表,以取代重复部分。以后,代码将被用来捕获新的序列。解压程序的算法被设计成为能够重新演绎出代码与原始数据之间的映射。

在看 LZ77 的细节之前,我们先看一个例子[①]:

```
the brown fox jumped over the brown foxy jumping frog
```

这段话有 424 位长,53 字节。算法从左向右处理。开始时,将每个字符转换为 9 位模式,首位为 1,后面 8 位是字符的数据。在转换中,算法同时检查重复的序列。当重复发生时,算法继

① [WEIS93]中的一个例子。

续搜索，直至重复结束。换句话说，算法搜索尽可能长的重复序列。第一个这样的序列是 **the brown fox**。这个重复的序列被一个前面序列的指针和它的长度所替代。在这个例子中，**the brown fox** 出现在 26 个字符前，长度为 13。假定有两种编码方式：8 位指针、4 位长度或 12 位指针、6 位长度；用 2 位的头部指示编码方式，00 表示第一种编码方式，01 表示第二种。则 **the brown fox** 的第二次出现被编码为：$<00_b><26_d><13_d>$，或 00 00011010 1101。

剩下的部分是：字符 **y**；**jump** 被替代成 $<00_b><27_d><5_d>$；以及 **ing frog**。

图 15.9 描述了这个压缩映射。压缩后的消息由 35 个 9 位字符和 2 个代码组成，总共 $35 \times 9 + 2 \times 14 = 343$位。与 424 位的未压缩数据相比，压缩比为 1.24。

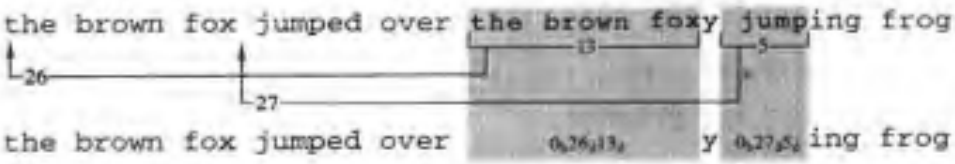
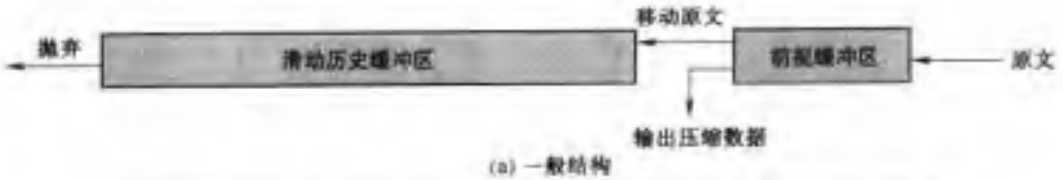


图 15.9　LZ77 模式的实例

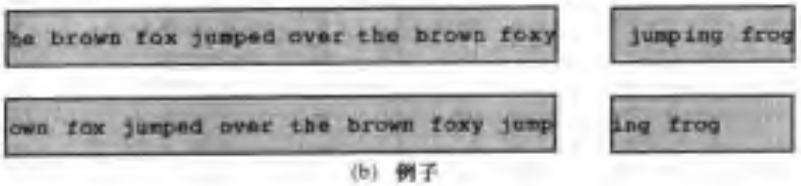
压缩算法

LZ77 压缩算法及其变体使用两个缓冲区：一个**滑动历史缓冲区**记录最后 N 个被处理的字符，一个**前视缓冲区**装下 L 个将被处理的字符[见图 15.10(a)]。算法从前视缓冲区的第一个字符开始向前寻找在滑动历史缓冲区中出现过的、长度不低于 2 的字符串，如果找不到，则将前视缓冲区中的第一个字符如上所述改造为 9 位字符推入滑动历史缓冲区，并将该缓冲区中最老的一个字符挤出缓冲区。如果找到了，算法将继续搜索最长的重复字符串，这个重复的字符串被一个三元组(指示位、指针、长度)所取代。如果重复串的长度为 K，则滑动缓冲区中最老的 K 个字符被挤出，刚被编码的 K 个字符被转入该缓冲区。

图 15.10b 显示了操作过程。这个例子假设滑动窗为 39 个字符，前视缓冲区有 13 个字符。在上面的例子中，前 40 个字符已经被处理过了，其中后 39 个字符以未被编码的形式存放于滑动窗中。剩下的部分在前视缓冲区中。算法寻找下一个重复字符串，5 个字符从前视缓冲区移入滑动窗，输出这个 5 字符串的编码。缓冲器的状态显示如下。



(a) 一般结构



(b) 例子

图 15.10　LZ77 模式

LZ77 速度快,有自适应能力,但也有缺陷。算法定义了一个有限长度的窗口查询前面的文本。对于相对较长的文本,有些潜在的重复串被遗漏了。这个窗口可以增大,但有两个代价:(1)算法比较重复字符串的时间变长,(2)指针的长度也要变长,以满足长距离的跳转。

解压算法

LZ77 的解压算法很简单。解压算法要保存被解压出的 N 个字符。当碰到压缩编码时,解压算法使用 < 指针 > 和 < 长度 > 域来恢复原文。

附录 15B 基数 64 转换

PGP 和 S/MIME 都用基数 64 转换的编码技术。这种技术可以将任意二进制输入转换成为可打印字符。这种转换有如下相关特性:

1. 函数输出的字符即使在所有场合都可以表示出来,而不是某个字符集的二进制编码。这样,这些字符就可以被特定的系统编码为任何它们需要的形式。例如:字母"E"在 ASCII 系统中被编码为 16 进制 45,在 EBCDIC 系统中则是 16 进制 C5。
2. 这个字符集有 65 个可打印字符。其中有一个用来填充,其他 $2^6 = 64$ 个字符可以组成任意的 6 位输入。
3. 字符集中不包含任何控制字符。这样的基数 64 编码就可以通过那些在报文中搜索控制符的邮件系统。
4. 不包含"-"符号。这个符号在 RFC 822 中有特殊含义,应该被回避。

表 15.9 列举了 6 位输入到字符的映射关系。字符集包括数字、字符和"+"、"/"。符号"="作为填充符。

表 15.9 基数 64 编码

6 位数值	编码字符	6 位数值	编码字符	6 位数值	编码字符	6 位数值	编码字符
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

图 15.11 演示了映射的过程。输入以每 3 个 8 位组或 24 位为一块。每 6 位一组，映射为一个字符。图中，这些字符用 8 位表示。通常 24 位输入会被扩展为 32 位输出。

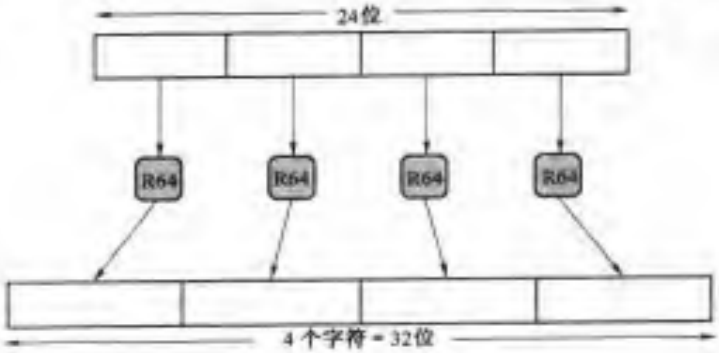


图 15.11　R64 编码

例如，24 位串 00100011 01011100 10010001，十六进制表示为 235C91。按 6 位一组排列为：

001000 110101 110010 010001

这些 6 位数用十进制表示为 8，53，50，17。查表 15.9 得到如下字符：I1yR。如果用 8 位 ASCII 码表示（奇偶校验位置零），我们有：

01001001 00110001 01111001 01010010

以十六进制表示为 49317952。总结如下：

输入数据	
二进制表示	00100011 01011100 10010001
十六进制表示	235C91
输入数据的 R64 编码	
字符表示	I1yR
ASCII 码（8 位，0 奇偶校验）	01001001 00110001 01111001 01010010
十六进制表示	49317952

附录 15C　PGP 随机数生成

PGP 使用一套复杂而强有力的体制生成随机数和伪随机数，提供给各种用途。PGP 以敲击键盘的时间和击键内容生成随机数。伪随机数生成是基于 ANSI X9.17 的一种算法。PGP 使用这些数的目的如下：

- 真随机数：
  - 生成 RSA 密钥对
  - 为伪随机数发生器提供种子
  - 为生成伪随机数提供其他输入
- 伪随机数：
  - 生成会话密钥
  - 为 CFB 模式提供初始化矢量

真随机数

PGP 维护一个 256 字节的随机位缓冲区。每次，PGP 等待一个键盘输入，记下 32 位格式的时间，从开始等待计时。接收到键盘输入后，记下 8 位格式的输入内容。用时间和内容生成一个密钥，再去加密随机位缓冲区。

伪随机数

伪随机数从 24 个 8 位组种子生成出 16 个 8 位组会话密钥和 8 个 8 位组初始化矢量，以及一个用来生成下一个伪随机数的种子。这个算法基于第 7 章(参见图 7.14)中描述的 X9.17 算法，但 3DES 被替换成了 CAST-128。该算法使用了如下数据结构：

1. 输入
   - 随机种子(24 个 8 位组)：如果为空，用 24 个真随机 8 位组填充。
   - 消息：用来加密消息的会话密钥和初始化矢量(IV)其实也是一种对该消息的函数。这进一步增加了密钥和初始化矢量的随机性，而且如果对手已经获得了消息的明文，也就不需要捕获会话密钥了。
2. 输出
   - K(24 个 8 位组)：前 16 字节 K[0..15]装会话密钥，后 8 字节 K[16..23]装初始化矢量。
   - 随机种子(24 个 8 位组)：置于此文档中的新种子。
3. 内部数据
   - dtbuf(8 个 8 位组)：前 4 字节 dtbuf[0..3]被初始化为当前日期/时间。这个缓冲区相当于 X12.17 算法的 DT 变量。
   - rkey(16 个 8 位组)：CAST-128 加密密钥。
   - rseed(8 个 8 位组)：相当于 X12.17 的 V_i。
   - rbuf(8 个 8 位组)：本算法生成的伪随机数，相当于 X12.17 的 R_i 变量。
   - K'(24 个 8 位组)：装随机种子的临时空间。

这个算法包括 9 步，即 G1 到 G9。第一步和最后一步是为了减少对手得到随机种子的可能性而设置的混杂步骤。剩下的步骤其实是 X12.17 算法的三重循环，如图 15.12 所示(比较图 7.14)。

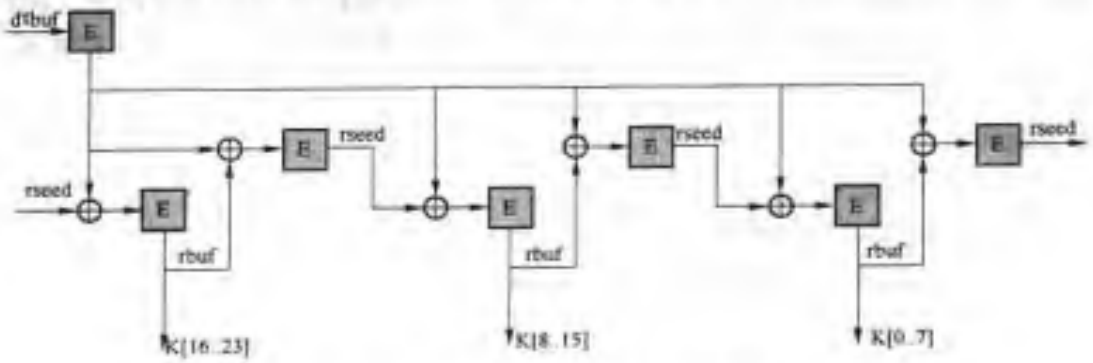


图 15.12 PGP 会话密钥和 IV 的产生(步骤 G2 - G8)

G1. [预洗上一次的种子]

a. 拷贝随机种子到 K[0..23]。

b. 取消息摘要(如果该消息已经被签名,就不再计算,否则就使用消息的前 4000 个 8 位组),将结果当做密钥,设初始化矢量为空,用 CFB 模式加密 K。结果存回 K。

G2. [设初始化种子]

a. 设 dtbuf[0..3]为本地 32 位时间,设 dtbuf[4..7]为全零。拷贝 K[0..15]到 rkey。拷贝 K[16..23]到 rseed。

b. 用 128 位 rkey 做密钥按 ECB 模式加密 64 位的 dtbuf;结果存回 dtbuf。

G3. [准备生成随机数]设 rcount 为 0,k 为 23。重复执行 G4 ~ G7 24 次(k = 23...0)。每次,将随机数存于 K 中。rcount 表示 rbuf 中未使用的随机数。它从 8 到 0 计算 3 次,共生成 24 个 8 位组。

G4. [可用字节?]若 rcount = 0,跳到 G5,否则跳到 G7。G5 和 G6 用 X12.17 算法生成了新的一批 8 个 8 位组。

G5. [生成新的随机数]

a. rseed←rseed ⊕ dtbuf。

b. rbuf←E_{rkey}[rseed],按 ECB 模式。

G6. [生成下一个种子]

a. rseed ← rbuf ⊕ dtbuf。

b. rseed ← E_{rkey}[rseed],按 ECB 模式。

c. rcount ← 8 。

G7. [一次从 rbuf 转换一个字节到 K]

a. rcount ← rcount - 1。

b. 生成一个真随机数字节 b,并没 K[k] ← rbuf [rcount] ⊕ b。

G8. [完成?]若 k = 0,跳到 G9;否则 k ← k - 1,跳到 G4。

G9. [后洗种子,返回结果]

a. 用 G4 ~ G7 生成的 24 字节(除了 G7 中不要 XOR 随机数),将结果存入 K′。

b. 以 K[0..15]为密钥,K[16..23]为初始化矢量,按 CFB 模式加密 K′,结果存入随机种子。

c. 返回 K。

通过 G9.a 生成的 24 个新 8 位组数得到会话密钥,应该是不可能的。但为了保证被存储下来的随机种子文件不会提供任何有关最近被使用的会话密钥的信息,新的 24 个 8 位组被作为新的种子加密存储。

这个完善的算法提供了强伪随机数。

第16章 IP安全性

互联网社团开发了各种应用的具体安全机制，如电子邮件(S/MIME、PGP)、客户/服务器(Kerberos)、Web访问(SSL)等。然而，用户还有与协议层相关的安全要求。例如，一个企业可以通过不允许访问不信任站点来运行一个安全的内部TCP/IP网络。通过实现IP级安全性，企业不仅可以保证各种带安全机制的应用程序，而且可以保护许多与安全无关的应用。

IP级安全性包括三个方面的内容：认证、保密和密钥管理。认证机制确保收到的包是从包头标识的源端发出的。另外，这个机制还确保了该包在传输过程中未被篡改。保密性是将报文加密后传送，防止第三方的窃听。密钥管理机制与密钥的安全交换相关。

本章简要介绍IP安全性(IPSec)和IPSec的体系结构，然后依次详细介绍上述三种功能。附录介绍互联网的一些协议。

16.1 IP安全性概述

1994年，互联网体系结构委员会(IAB)发布了“互联网体系结构中的安全性”报告(RFC 1636)。该报告指出，大多数人认为互联网需要更多、更好的安全性，并指明了使用密钥领域的安全机制。因此产生了对网络基础设施的非授权监控和网络传输控制的需求，以及使用认证和加密保护端到端用户的传输。

2001年，CERT(Computer Emergency Response Team)在其年度报告中列举出52 000多个安全事件。最严重的攻击包括IP欺诈，攻击者使用假的IP地址创建数据包，欺骗基于IP认证的应用程序，以及使用各种手段在信息传送过程中窃听登录信息和数据库信息等内容。

根据以上情况，IAB认为下一代IP应包括认证和加密等必要的安全特性。幸运的是，这些安全能力在现在的IPv4和以后的IPv6中都可以使用。这意味着供应商现在可以开始提供这些功能，且事实上许多供应商已经提供了这些功能。

16.1.1 IPSec的应用

IPSec提供了在LAN、WAN和互联网中安全通信的能力。它的用途包括如下各方面：

- **分支机构通过互联网安全互联**：一个公司可以在互联网和公用WAN上建立安全的虚拟专用网，使得依赖于互联网的交易成为可能，并减少了对专用网络的需求，节省了开销和网络管理费用。
- **远程安全访问互联网**：使用了IP安全协议的终端用户可以通过本地调用访问互联网服务提供商(ISP)，以获得对公司网络的安全访问，减少了雇员上班和远程通信的费用。
- **与合作者建立外联网和内联网联系**：IPSec可以与其他组织进行安全通信，确保认证、保密和提供密钥交换机制。
- **加强电子商务的安全性**：虽然一些Web和电子商务应用程序是建立在安全协议之上的，但使用IPSec能加强其安全性。

IPSec 能支持各种应用的原理在于它可以在 IP 层加密和/或认证所有流量，这样就可以保护所有的分布式应用，包括远程登录、客户/服务器、电子邮件、文件传输、Web 访问等。

图 16.1 是使用 IPSec 的一个典型方案。一个使用 LAN 的组织可以分布在各地。非安全的 IP 流量在各 LAN 内部使用，通过专用或公用 WAN 的外部流量则使用 IPSec 协议。这些协议在网络设备如路由器或防火墙中运行，它们将各 LAN 与外部世界相连。IPSec 网络设备将对所有进入 WAN 的流量加密、压缩，并解密和解压来自 WAN 的流量，这些操作对 LAN 上的工作站和服务器是透明的。当然，对于使用拨号上网的个人用户，也可以使用安全传输。这些用户的工作站必须使用 IPSec 协议来提供安全性。

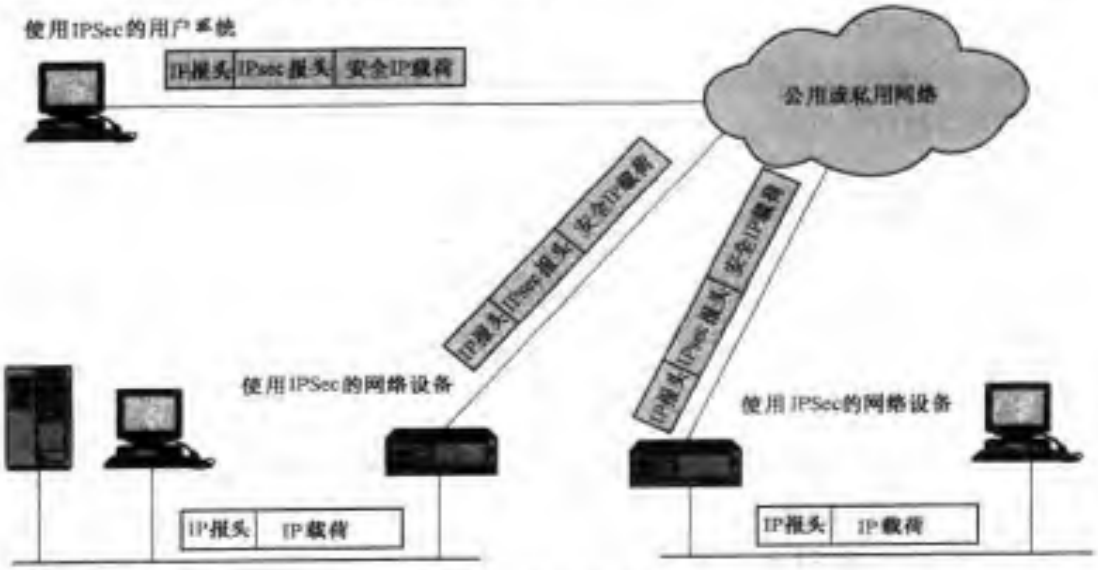


图 16.1　一个 IP 安全的方案

16.1.2　IPSec 的优点

[MARK97]列举了 IPSec 的如下优点：

- 当防火墙或路由器使用 IPSec 时，它提供了对通过其边界的所有通信的强安全性。公司或工作组内部的通信不会导致与安全处理相关的开销。
- 防火墙内的 IPSec 能在所有外部流量均使用 IP 时阻止旁路，因为防火墙是从互联网进入组织内部的惟一通道。
- 位于传输层(TCP、UDP)之下的 IPSec 对所有应用都是透明的。因此，当防火墙或路由器使用 IPSec 时，不需要对用户系统或服务系统做任何改变，即使在末端系统中使用 IPSec，也不需要改变上层的软件和应用。
- IPSec 可以对终端用户透明，不需要对用户进行安全机制培训，如对每个用户发布一个密钥，在用户离开组织时回收密钥等。
- 如果需要，IPSec 可以为个体用户提供安全性。这对网上上班族非常有用，它可以在敏感的应用中为用户和组织建立一个虚拟子网。

16.1.3 路由应用

除了支持终端用户和保护上述系统和网络外，IPSec 在互联网的路由结构中扮演了一个非常重要的角色。[HUIT98]列举了使用 IPSec 的例子。IPSec 可确保：

- 路由广播(一个新的路由器公告它的存在)来自于授权路由器。
- 邻居广播(路由器寻找建立或维护与其他路由区域中路由器的相邻关系)来源于授权路由器。
- 重定向报文来源子发送包的初始路由器。
- 路由更新无法伪造。

没有以上安全措施，攻击者可以中断通信或转发某些流量。路由协议如 OSPF 必须在用 IPSec 安全协议的路由器间运行。

16.2 IP 安全体系结构

IPSec 规范相当复杂，为了能了解其整体结构，我们将从定义 IPSec 的文档开始介绍，然后讨论 IPSec 的服务并介绍安全方面的概念。

16.2.1 IPSee 文档

IPSec 有许多文档，重要的有 1998 年发布的 RFC 2401、2402、2406 和 2408。

- RFC 2401：安全结构概述。
- RFC 2402：IP 扩展的包认证描述(IPv4 和 IPv6)。
- RFC 2406：IP 扩展的包加密描述(IPv4 和 IPv6)。
- RFC 2408：特定加密机制。

IPv6 必须支持这些特性，而 IPv4 可以选择支持。在这两种情况中，均在主 IP 头中使用扩展头实现安全特性。认证的扩展头称为认证头，加密的扩展头称为载荷安全性封装头(ESP header)。

除了这四个文档外，IETF 设立的 IP 安全协议工作组又做了大量工作。整个文档分为七部分，如图 16.2 所示(RFC 2401)：

- **体系结构**：包括一般概念、安全需求、定义和 IPSec 的机制。
- **载荷安全封装(ESP)**：包括包格式和使用 ESP 加密/认证包的一些相关约定。
- **认证头(AH)**：包括包格式和使用 AH 认证包的一些相关约定。
- **加密算法**：一系列描述 ESP 中使用的各种加密算法的文档。
- **认证算法**：一系列描述 AH 中使用的各种认证算法和 ESP 认证选项的文档。
- **密钥管理**：描述密钥管理模式的文档。
- **解释域(DOI)**：包括与其他文档相关的一些值，如被认可的加密、认证算法标识和密钥生存周期参数。

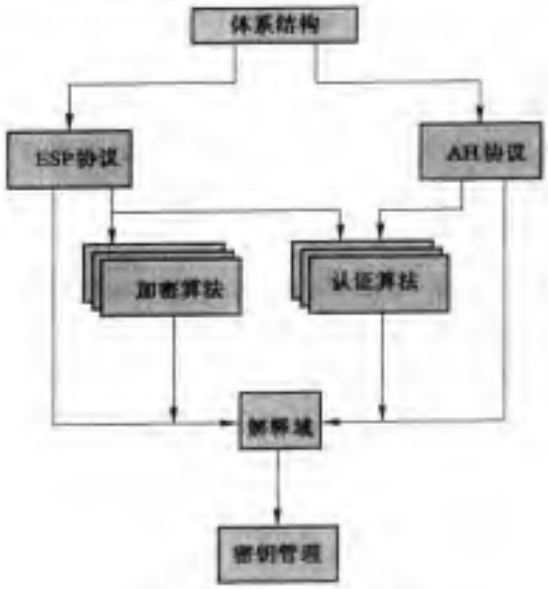


图 16.2　IPSec 文档概述

16.2.2　IPSec 服务

IPSec 通过允许系统选择所需的安全协议、决定服务所使用的算法和提供任何服务需要的密钥来提供 IP 级的安全服务。两种协议都提供安全性：一个是使用 AH 协议报头的认证协议，另一个是为数据包设计的加密/认证协议 ESP。其提供的服务如下：

- 访问控制
- 无连接完整性
- 数据源认证
- 拒绝重放包(部分顺序完整性格式)
- 保密性(加密)
- 限制流量保密性

表 16.1 描述了 AH 和 ESP 协议所支持的服务。对 ESP 而言，可分为支持和不支持认证两种；AH 和 ESP 均支持基于分布密钥的访问控制。而流量管理则与安全协议相关。

表 16.1　IPSec 服务

	AH	ESP(仅包括加密)	ESP(加密 + 认证)
访问控制	√	√	√
无连接完整性	√		√
数据源认证	√		√
拒绝重放包	√	√	√
保密性		√	√
限制流量保密性		√	√

16.2.3 安全关联

在IP的认证和保密机制中出现的一个核心概念是安全关联(SA)。关联是发送方和接收方之间的单向关系,该关联为双方的通信提供安全服务。如果需要双向安全交换,则需要建立两个安全关联。安全服务可以由AH或ESP提供,但不能两者都提供。

一个安全关联由三个参数惟一确定:

- **安全参数索引(SPI)**:一个与SA相关的位串,仅在本地有意义。SPI由AH和ESP携带,使得接收系统能选择合适的SA处理接收包。
- **IP目的地址**:目前,只允许使用单一地址,表示SA的目的地址,可以是用户末端系统、防火墙或路由器。
- **安全协议标识**:标识该关联是一个AH安全关联或ESP安全关联。

然而,在任何IP包中,[①] 安全关联由IPv4或IPv6报头中的目的地址惟一标识,SPI被封装于AH或ESP扩展头中。

SA参数

在任何IPSec实现中,有一个名义上的[②] 安全关联数据库(SAD),它定义与每个SA相关联参数。一个安全关联通常用以下参数定义:

- **序列号计数器**:生成AH或ESP报头中序列号的32位值,详见16.3节(必须实现)。
- **序列号溢出标志**:标志序列号计数器是否溢出,生成审核事件。溢出时,阻止在此SA上继续传输包(必须实现)。
- **反重放窗口**:用于判定一个内部AH或ESP数据包是否是重放,详见16.3节(必须实现)。
- **AH信息**:认证算法、密钥、密钥生存期和AH的相关参数(AH必须实现)。
- **ESP信息**:加密和认证算法、密钥、初始值、密钥生存期和ESP的相关参数(ESP必须实现)。
- **安全关联的生存期**:一个特定的时间间隔或字节计数。超过后,必须终止或由一个新的SA(和新SPI)替代,并加上应进行何种操作的指示(必须实现)。
- **IPSec协议模式**:隧道、传输或通配符(必须实现),这些模式将在后面详细讨论。
- **Path MTU**:最大传输单元(不需要分段传输的最大包长度)路径和迟滞变量(必须实现)。

认证和保密机制与密钥管理机制相互独立。分布密钥使用的密钥管理机制只通过安全参数索引与认证和保密机制相联系。

SA选择子

IPSec为用户提供多种方式在IP通信中实现IPSec服务。SA可以根据用户的意愿进行配置。并且,IPSec对需要IPSec保护的流量和不需要IPSec保护的流量进行大粒度区分。

---

① 在本章中,术语IP指IPv4报文或IPv6包。

② 名义上是指由安全联数据库提供的功能必须在IPSec的任何实现中都存在,但提供功能的方式由实现者决定。

IP 流量与特定 SA 相关(或在不需要 IPSec 时无 SA)是通过安全策略数据库(SPD)实现的。SPD 至少应包括定义 IP 流量子集的入口、指向该流量 SA 的指针。在更复杂的情况下,多个入口可与一个 SA 相连,或多个 SA 与一个 SPD 入口相关。读者可以参阅相关的 IPSec 文档。

每个 SPD 入口由一个 IP 集合和上层协议定义,称为选择子。实际上,这些选择子用于过滤输出流量,并将它们映射到某个特定的 SA。每个 IP 包的输出过程遵守如下过程:

1. 在 SPD 中,比较包中相应域的值(选择子域),寻找匹配的 SPD 入口,这可能指向零个或多个 SA。
2. 如果该包存在 SA,则选定 SA 和其关联的 SPI。
3. 执行所需的 IPSec 处理(如 AH 或 ESP 处理)。

SPD 入口由以下选择子决定:

- **目的 IP 地址**:可以是单一 IP 地址、一组或一定范围的地址和地址掩码。后两者要求多个目的系统共享相同的 SA(例如,位于防火墙之后)。
- **源 IP 地址**:可以是单一 IP 地址、一组或一定范围的地址和地址掩码。后两者要求多个源系统共享相同的 SA(例如,位于防火墙之后)。
- **用户标识**:来自操作系统的用户标识。用户操作系统提供该标识,而不是 IP 或上层报头域提供。
- **数据敏感性级别**:用于系统提供信息流安全级别(如秘密或未分类)。
- **传输层协议**:从 IPv4 或 IPv6 的邻接头域中获得,可以是单个协议号,也可以是一组或一定范围的协议号。
- **IPSec 协议{AH 或 ESP 或 AH/ESP}**:如果有,则从 IPv4 或 IPv6 协议中的邻接头域中获得。
- **源端口和目的端口**:可以是单个 TCP 或 UDP 端口、一组端口和端口通配符。
- **IPv6 报类**:从 IPv6 报头中获得,可以是一个特定的 IPv6 类型或通配类型。
- **IPv6 报流标签**:从 IPv6 报头中获得,可以是一个特定的 IPv6 流标签或通配类型。
- **IPv4 报服务类型**:从 IPv4 报头中获得,可以是一个特定的 IPv4 服务类型或通配类型。

16.2.4 传输模式和隧道模式

AH 和 ESP 均支持两种模式:传输模式和隧道模式,详细描述分别参见 16.3 节和 16.4 节,在此作简要介绍。

传输模式

传输模式主要为上层协议提供保护,同时增加了 IP 包载荷的保护。例如,TCP 段或 UDP 段、ICMP 包均是在主机协议栈的 IP 层进行操作。典型地,传输模式用于在两台主机(如服务器与工作站之间、两个工作站之间)进行的端到端通信。当一个主机在 IPv4 上运行 AH 或 ESP 时,其载荷是跟在 IP 报头后面的数据,对 IPv6 而言,其载荷是跟在 IP 报头后面的数据和 IPv6 的任何扩展头。

传输模式的 ESP 加密和认证(可选)IP 载荷,但不包括 IP 报头。传输模式的 AH 认证 IP 载荷和 IP 报头的选中部分。

隧道模式

隧道模式对整个 IP 包提供保护。为了达到这个目的，当 IP 包加 AH 或 ESP 域之后，整个数据包加安全域被当做一个新 IP 包的载荷，并拥有一个新的外部 IP 报头。原来或内部的整个包利用隧道在网络之间传输，沿途路由器不能检查内部 IP 报头。由于原来的包被封装，新的、更大的包可以拥有完全不同的源地址与目的地址，以增强安全性。当 SA 的一端或两端为安全网关时使用隧道模式，如使用 IPSec 的防火墙或路由器。防火墙外的主机在没有 IPSec 时也可以实现安全通信。而当主机生成的未保护包通过本地网络边缘的防火墙或安全路由器时，IPSec 提供隧道模式的安全性。

IPSec 如何操作隧道模式的例子如下。网络中的主机 A 生成以另一个网络中主机 B 作为目的地址的 IP 包，该包选择的路由是从源主机到 A 网络边界的防火墙或安全路由器；再由防火墙过滤所有的外部包。根据对 IPSec 处理的请求，如果从 A 到 B 的包需要 IPSec 处理，则防火墙执行 IPSec 处理并在外部 IP 报头中封装包，外部 IP 包中的源 IP 地址为此防火墙的 IP 地址，目的地址可能为 B 本地网络边界的防火墙的地址。这样，包被传送到 B 的防火墙，而其间经过的中间路由器仅检查外部 IP 报头；在 B 的防火墙处，除去外部 IP 报头，内部的包被送往主机 B。

ESP 在隧道模式中加密和认证(可选)整个内部 IP 包，包括内部 IP 报头。AH 在隧道模式中认证整个内部 IP 包和外部 IP 报头中的选中部分。

表 16.2 总结了传输模式和隧道模式的功能。

表 16.2 传输模式和功能

	传输模式 SA	隧道模式 SA
AH	认证 IP 载荷和 IP 报头中的选中部分、IPv6 扩展头	认证整个内部 IP 包(内部报头和 IP 载荷)和部分选中的外部 IP 报头、外部 IPv6 扩展头
ESP	加密 IP 载荷和跟在 ESP 报头后的所有 IPv6 扩展头	加密内部 IP 包
带认证的 ESP	加密 IP 载荷和跟在 ESP 头后的所有 IPv6 扩展头；认证 IP 载荷，但没有 IP 报头	加密内部 IP 包 认证内部 IP 包

16.3 认证头

认证头支持数据完整性和 IP 包的认证。数据完整性确保在包的传输过程中内容不可更改。认证确保末端系统或网络设备能对用户或应用程序进行认证，并相应地提供流量过滤功能，同时还能防止地址欺诈攻击和重放攻击。

认证基于第 11 章讲述的报文认证编码(MAC)，双方必须共享同一个密钥。

认证头由如下域组成(见图 16.3)：

- **邻接头(8 位)**：标识紧跟在此报头后面的头类型。
- **载荷长度(8 位)**：字长为 32 位的认证头长度减 2。例如，认证数据域的默认长度是 96 位或 3 个 32 位字，另加三个字长的固定头，总共六个字，则载荷长度域的值为 4。

- **保留(16 位)**：备用。
- **安全参数索引(32 位)**：标识安全关联。
- **序列号(32 位)**：单调递增计数值，以后讨论。
- **认证数据(变量)**：变长域(必须是整数个 32 位字)，包含完整性校验值(ICV)或包的 MAC，以后讨论。

图 16.3　IPSec 认证头

16.3.1　反重放服务

重放攻击是指攻击者在得到一个经过认证的包后，在后来将其传送到目的站点的行为。而重复接收经过认证的 IP 包可能会以某种方式中断服务或产生不可预料的后果。序列号域就可以防止上述攻击。首先，了解发送方如何生成序列号，再讨论接收方如何处理它。

当建立了一个新的 SA 时，**发送方**将序列号初值设为 0，每次在 SA 上发送一个包，则计数器加 1 并将值放入序列号域，这样，使用的第一个值即为 1。如果要求反重放(默认设置)，则发送方不允许循环计数(到达 $2^{32}-1$ 后返回 0)，否则，同一个序列号可以有多个合法的包。如果该序列号达到 $2^{32}-1$，则 SA 必须终止，用新的密钥协商生成新的 SA。

由于 IP 是无连接、不可靠的服务，协议不能保证包按顺序传输，也不能保证所有的包均被传输。因此，IPSec 认证文档声明，接收方应实现一个大小为 W 的窗口(W 默认为 64)。窗口的右边界代表最大的序列号 N，记录目前收到的合法包的最大序列号，任何序列号在 $N-W+1$ 到 N 之间的包均可以被正确接收(如被验证)，并标记窗口的正确位置(如图 16.4)。接收到包后的处理过程如下：

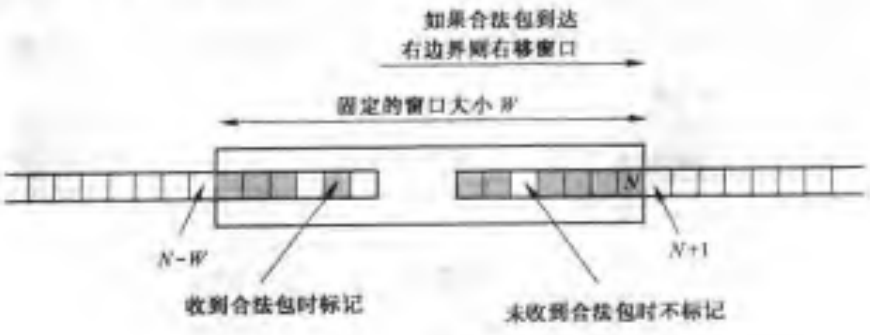


图 16.4　反重放机制

1. 如果所接受的包在窗口中且是新包，则验证 MAC。如果验证通过，则在窗口中标记相应位置。
2. 如果所接受的包超过窗口的右边界且是新包，则验证 MAC。如果验证通过，则以这个序列号为窗口的右边界并在窗口中标记相应的位置。
3. 如果所接受的包超过窗口的左边界或未通过验证，则忽略包，并产生审核事件。

16.3.2 完整性校验值

认证数据域包含完整性校验值(ICV)。ICV 是一种报文认证编码 MAC 或 MAC 算法生成的截断代码。有许多规格说明描述了这一规范：

- HMAC-MD5-96
- HMAC-SHA-1-96

上述两种规格说明均使用了 HMAC 算法，而前者使用 MD5 散列函数，后者使用 SHA 散列函数(参见第 12 章)。两者均先计算全部的 HMAC 值，然后截断前 96 位(认证数据域的默认长度)。

MAC 根据如下部分进行计算：

- IP 报头：传输过程中不变的部分和 AH SA 终点可以预测的部分，对可变部分或不可预计的部分全部置 0，便于在源端和目的端计算。
- AH 报头不包括认证数据域。认证数据域被置 0，便于在源端和目的端计算。
- 整个上层协议数据，假设在传输过程不变，如 TCP 段或隧道模式中的内部 IP 包。

对 IPv4 而言，不变域为 Internet 报头长度，可变但可预测的域为目的地址。对 ICV 计算具有 0 优先级的可变域为生存时间域及头校验和域。源地址和目的地址域均被保护，以防地址欺诈。

对 IPv6 而言，不变域为基本头的版本号，可变但可预测的域为目的地址，具有 0 优先级的可变域为流标签。

16.3.3 传输模式和隧道模式

图 16.5 描述了使用 IPSec 认证服务的两种途径。其一，直接在服务器和客户工作站间提供认证，工作站可以与服务器在同一个网络中，也可以在其他外部网中。只要工作站和服务器共享同一个受保护的密钥，认证即是安全的，此时，使用传输模式 SA；其二，在服务器不支持认证时，远程工作站向防火墙证明自己的身份以访问整个内部网络，此时，使用隧道模式 SA。

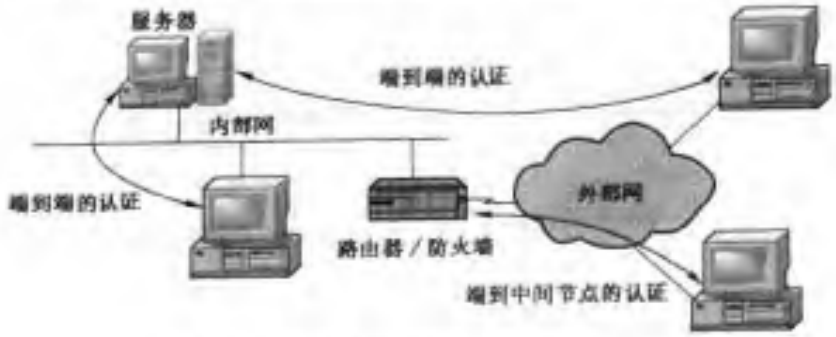


图 16.5 端到端的认证和端到中间节点的认证

本小节介绍 AH 提供的认证范围和两种模式的认证头。图 16.6(a)是典型的 IPv4 和 IPv6 包。此时，IP 载荷为 TCP 分段，也可以是任何使用 IP 的数据单元，如 UDP 或 ICMP。

在 IPv4 的传输模式 AH 中，AH 插入到原始 IP 报头之后、IP 载荷(如 TCP 分段)之前。如图 16.6(b)所示。认证包括了除 IPv4 报头中可变的、被 MAC 计算置为 0 的域以外的整个包。

在 IPv6 中，AH 被作为端到端载荷，即不被中间路由器检查或处理。因此，AH 出现在 IPv6 基本头、跳、路由和分段扩展头之后。目的地址作为可选报头在 AH 前面或后面，由特定语义决定。同样，认证包括了除 IPv4 报头中可变的、被 MAC 计算置为 0 的域以外的整个包。

对隧道模式 AH 而言，整个原始 IP 包被认证，AH 被插入到原始 IP 报头和新外部 IP 报头之间[如图 16.6(c)所示]。内部 IP 报头包含源地址和目的地址，而外部 IP 报头可包含不同的 IP 地址(如防火墙或其他安全网关的地址)。

使用隧道模式，整个内部 IP 包，包括整个内部 IP 报头均被 AH 保护。外部 IP 报头(IPv6 中的外部 IP 扩展头)除了可变且不可预测的域之外，均被保护。

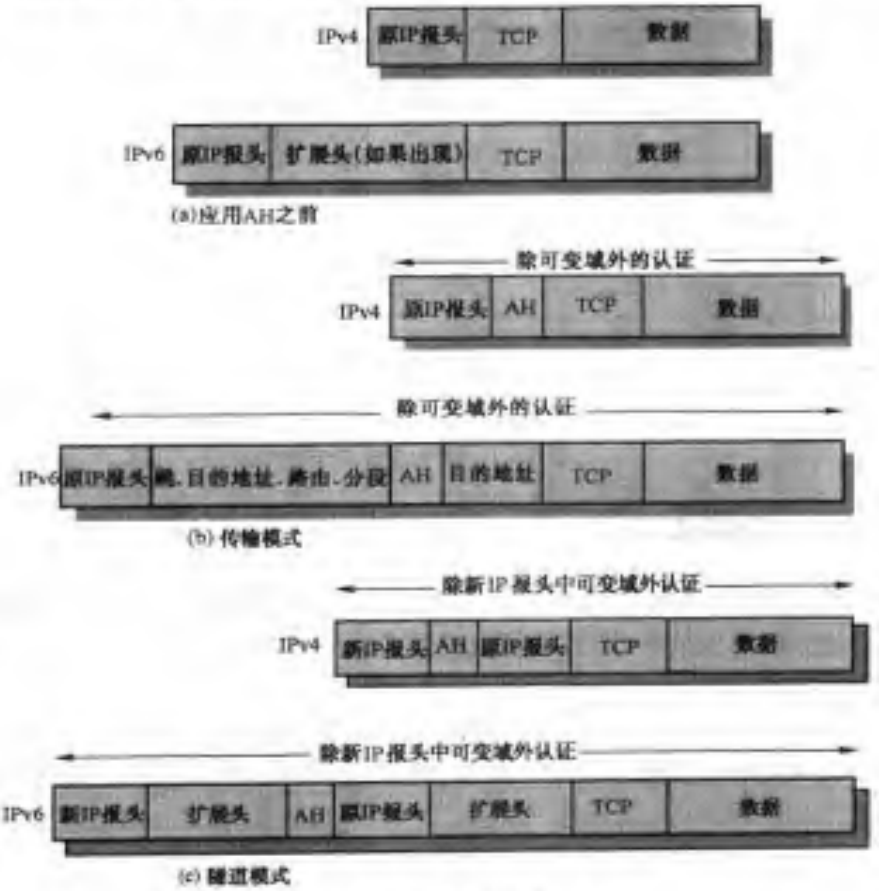


图 16.6　AH 认证的作用域

16.4　封装安全载荷

封装安全载荷提供的保密服务包括报文内容保密和流量限制保密，ESP 还可以提供与 AH 相同的认证服务。

16.4.1 ESP 格式

图 16.7 是 ESP 包的格式，它包含如下各域：

- **安全参数索引(32 位)**：标识安全关联。
- **序列号(32 位)**：单调递增计数值，提供反重放功能。
- **载荷数据(变量)**：被加密保护的传输层分段(传输模式)或 IP 包(隧道模式)。
- **填充域(0～255 字节)**：此域的目的稍后讨论。
- **填充长度(8 位)**：填充数据的长度。
- **邻接头(8 位)**：标识载荷中第一个报头的数据类型(如 IPv6 中的扩展头或上层协议 TCP 等)。
- **认证数据(变量)**：变长域(必须为 32 位字长的整数倍)，包含根据除认证数据域外的 ESP 包计算的完整性校验值。

图 16.7 IPSec ESP 格式

16.4.2 加密和认证算法

载荷数据、填充数据、填充长度和邻接头域在 ESP 中均被加密。如果加密载荷的算法需要初始矢量 IV 这样的同步数据，则必须从载荷数据域头部取。如果包括 IV，IV 通常作为密文的开头，但并不被加密。

目前的规约规定一个兼容实现必须支持 DES 算法按 CBC 加密(参见第 3 章)，且 DOI 文档对其他算法定义了标识符，因而容易用于加密，可以用来加密的算法如下(参见第 6 章)：

- 3DES
- RC5
- IDEA
- 三密钥三重 IDEA
- CAST
- Blowfish

与 AH 相同，ESP 支持使用默认为 96 位的 MAC，且应支持 HMAC-MD5-96 和 HMAC-SHA-1-96。

16.4.3　填充

填充域功能如下：

- 如果加密算法需要明文是某个数目字节的倍数(例如，用于块密文的单个块的倍数)，则填充域可用于扩展明文长度(包括载荷数据、填充数据、填充长度和邻接头域)到所需长度。
- ESP 格式需要填充长度和邻接头域为右对齐的 32 位字，密文长度必须是 32 位的整数倍。填充域用来确保这种排列。
- 增加额外的填充域可以隐藏载荷的实际长度，并提供部分流量保护。

16.4.4　传输和隧道模式

图 16.8 是两种 IPSec ESP 服务使用的途径。在图 16.8(a)中，加密和认证(可选)直接由两个主机提供；在图 16.8(b)中，描述了如何用隧道模式操作创建一个虚拟专用网。在本例中，企业拥有通过互联网互连的四个专用网，内部网络的主机使用互联网传输数据，但不和其他基于互联网的主机交互，各内部网络的安全网关终止隧道，配置允许主机没有保护能力。前者使用传输模式 SA 来支持，后者使用隧道模式 SA 来支持。

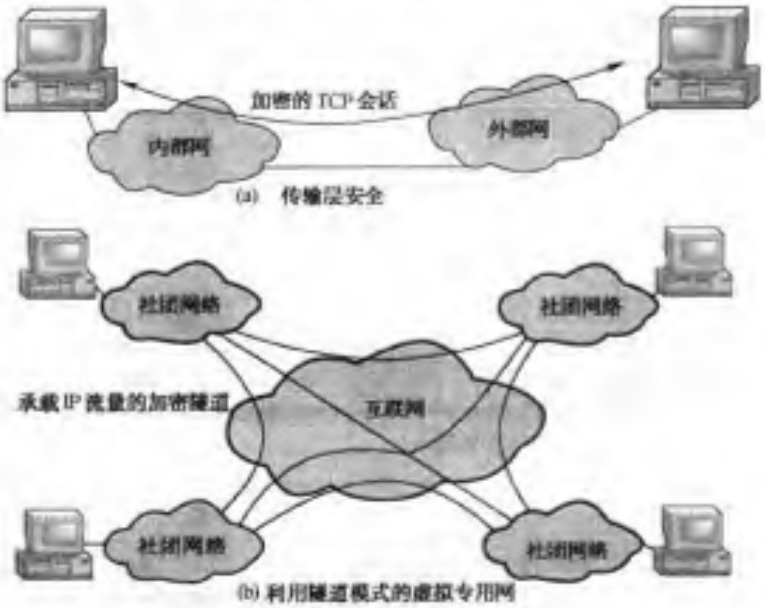


图 16.8　传输模式加密和隧道模式加密

在本小节中，我们将介绍这两种模式的 ESP 的范围，对于 IPv4 和 IPv6 的考虑有所不同。与 AH 一样，我们使用图 16.6(a)中的包格式作为开始。

传输模式 ESP

传输模式 ESP 用于加密和认证(可选)IP 携带的数据(如 TCP 分段)，如图 16.9(a)所示。在此模式下使用 IPv4，ESP 头位于传输头(TCP、UDP、ICMP)之前的 IP 包中，ESP 尾(填充数据、填充长度和邻接头域)放入 IP 包尾部。如果选择了认证，则将 ESP 的认证数据域置于 ESP 尾之后。整个传输层分段和 ESP 尾一起加密。认证覆盖了 ESP 头和所有密文。

在 IPv6 中，ESP 被视为端到端载荷，即不被中间路由器校验和处理。因此，ESP 头出现在 IPv6 基本头、跳、路由和分段扩展头之后，目的可选扩展头可根据愿望出现在 ESP 头之前或之后。如果可选扩展头在 ESP 头之后，则加密包括整个传输段、ESP 尾和目的可选扩展头。认证覆盖了 ESP 头和所有密文。

传输模式操作可归纳如下：

1. 在源端，包括 ESP 尾和整个传输层分段的数据块被加密，块中的明文被密文替代，形成要传输的 IP 包。如果选择了认证，则加上认证。
2. 将包送往目的地。中间路由器需要检查和处理 IP 报头以及任何附加的 IP 扩展头，但不需要检查密文。
3. 目的节点对 IP 报头和任何附加的 IP 扩展头进行处理后，利用 ESP 头中的 SPI 解密包的剩余部分，恢复传输层分段数据。

传输模式操作为任何使用它的应用提供保护，而不需要在每个单独的应用中实现。同时，这种方式也是高效的，仅增加了少量的 IP 包长度。它的一个弱点是可能对传输包进行流量分析。

隧道模式 ESP

隧道模式 ESP 用于加密整个 IP 包[如图 16.9(b)所示]。在此模式中，将 ESP 头作为包的前缀，并在包后附加 ESP 尾，然后对其进行加密。该模式用于对流量计数分析。

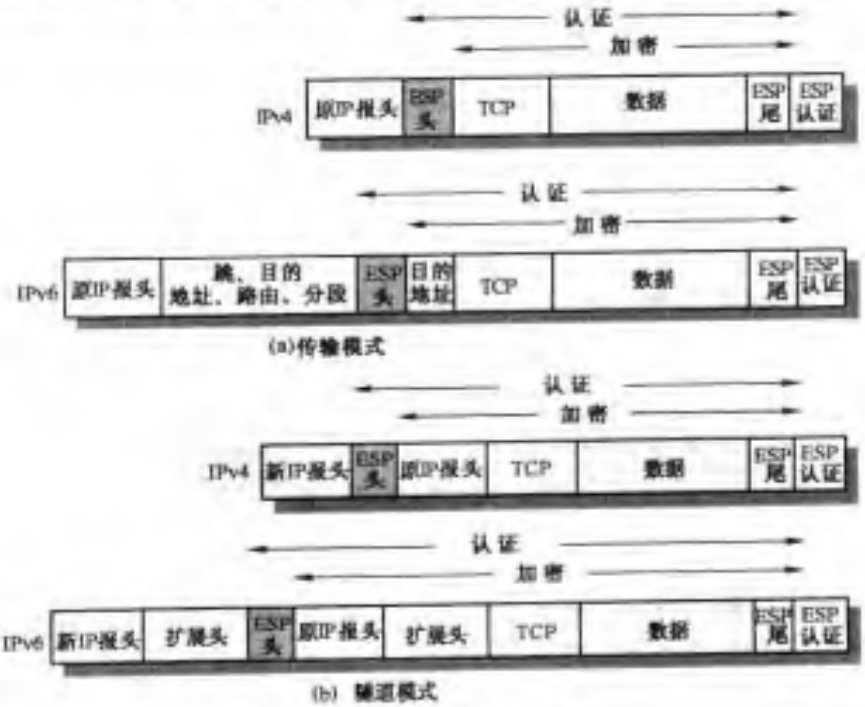


图 16.9 ESP 加密和认证的作用域

由于 IP 报头中包含目的地址和可能的路由以及跳信息，不可能简单地传输带有 ESP 头且被加密的 IP 包，因为这样中间路由器就不能处理该数据包。因此，必须用新的 IP 报头封装整个数据块(ESP 头、密文和可能的认证数据)，其中拥有足够的路由信息，却没有为流量分析提供信息。

然而，传输模式适合于保护支持 ESP 特性的主机之间的连接，而隧道模式则适用于防火墙或其他安全网关，保护内部网络，隔离外部网络。后者加密仅发生在外部网络和安全网关之间或两个安全网关之间，从而内部网络的主机不负责加密工作，通过减少所需密钥数目简化密钥分配任务。另外，它阻碍了基于最终目的地址的流量分析。

考虑这样一种情况，外部主机想与防火墙保护的内部网络中的一台主机进行通信，则在将传输层分段从外部主机传到内部主机过程中采取以下步骤：

1. 源端将目标内部主机的 IP 地址作为目的地址准备一个内部 IP 包，以 ESP 头为前缀，再将包和 ESP 尾加密，并可以加上认证数据。然后新的 IP 报头封装数据块(基本头和可选的扩展，如 IPv6 的路由和跳信息)，目的地址为防火墙的地址，生成外部 IP 包。
2. 外部 IP 包到达目的防火墙，其中经过的中间路由器应检查和处理外部 IP 报头及任何外部 IP 扩展头，而不需要检查密文。
3. 目的防火墙检查和处理外部 IP 报头及任何外部 IP 扩展头，然后根据 ESP 头中的 SPI，解密包的剩余部分，恢复内部 IP 包的明文，然后在内部网络中传输包。
4. 内部包经过 0 个或多个路由器到达目的主机。

16.5 安全关联组合

单个 SA 可以实现 AH 或 ESP 协议，但不能两者都实现。有时，特定的流量需要在主机间提供 IPSec 服务，并在安全网关间(如防火墙间)为相同流量提供分离的服务。此时，为了达到理想的 IPSec 服务，需要为相同流量提供多个 SA。**安全关联束**是指为提供特定的 IPSec 服务集所需的一个 SA 序列。安全关联束中的 SA 可以在不同节点终止，也可以在同一个节点终止。

安全关联可以通过两种方式组合成束：

- **传输邻接**：指在不调用隧道的情况下，对一个 IP 包使用多个安全协议。组合 AH 和 ESP 的方法仅允许一级组合，因为对一个 IPSec 实例进行多次嵌套没有任何好处。
- **隧道迭代**：指通过 IP 隧道应用多层安全协议。由于每个隧道可以在路径上的不同 IPSec 节点起始和终止，因此，该方法允许多层嵌套。

两种方法可以组合使用(例如，在主机间使用传输 SA 而在安全网关之间的路径上使用隧道 SA)。

当考虑 SA 束时，一个值得一提的问题是认证和加密的顺序和方法，我们将在后面详细讨论。下而了解至少涉及一个隧道的 SA 组合。

16.5.1 认证加保密

加密和认证在主机间要求传送经过加密和认证的 IP 包时是可以组合的，以下是一些组合途径。

带认证选项的 ESP

该方法如图 16.9 所示。用户首先对要保护的数据应用 ESP，然后加上认证数据，这又分为两种情况：

- **传输模式 ESP**：认证和加密仅作用于传送到主机的 IP 载荷，不保护 IP 报头。
- **隧道模式 ESP**：认证作用于整个要发往外部 IP 目的地址(如防火墙)的整个 IP 包，并在

目的地进行验证。整个内部 IP 包在传往内部 IP 目的地时按专用机制保护。

两种情况下,认证都只对密文使用而不是应用于明文。

传输邻接

另一种加密后认证的方法是使用两个捆绑在一起的传输 SA,内部是 ESP SA,外部是 AH SA。此时,ESP 没有认证。由于内部 SA 是一个传输 SA,仅对 IP 载荷加密。得到的包包含一个 IP 报头(可能有 IPv6 扩展头)和 ESP。然后,使用传输模式的 AH,使得认证覆盖 ESP 和除可变域外的原始 IP 报头。与简单地使用带 ESP 认证选项的 ESP SA 相比,此方法的好处在于认证覆盖了更多域,包括源 IP 地址和目的 IP 地址,缺点在于两次 SA 的开销大于一次 SA 的开销。

传输 – 隧道束

在加密之前认证有如下优点。首先,由于认证数据被加密保护,因此任何人都不可能从报文中截取和改变认证数据而不被发现;其次,可能希望在报文的目的地接收并保留认证信息以备后用。此时,在加密之前进行报文验证将更方便,否则,需要再次加密报文来保留认证的签名信息。

在两个主机间先认证后加密可以使用包含内部 AH 的传输 SA 和外部 ESP 的隧道 SA 的安全关联束。此时,认证作用于 IP 载荷和除可变域外的 IP 报头(包括扩展),然后将隧道模式的 ESP 作用于得到的 IP 包,该数据包含经过认证的整个内部密文和新的外部 IP 报头(及扩展)。

16.5.2 安全关联的基本组合

IPSec 结构文档列举了四种 SA 组合,IPSec 的主机(如工作站、服务器)或安全网关(如防火墙、路由器)必须支持这些组合。如图 16.10 所示。图的下部表示元素的物理连接,上部表示一个或多个嵌套的 SA 逻辑连接。每个 SA 可以是 AH 或 ESP。对主机到主机的 SA 而言,模式可以是传输的或隧道的;否则,必须是隧道模式。

情况 1 为实施 IPSec 的末端系统间提供所需的安全机制。通过 SA 通信的任何两个末端系统,必须共享特定的密钥,可能的组合如下:

a. 传输模式的 AH
b. 传输模式的 ESP
c. 传输模式的 AH 后紧跟 ESP(ESP SA 内置于 AH SA)
d. 在 AH 或 ESP 隧道模式中拥有 a、b 或 c 中的任一个

我们已经讨论了如何用各种关联支持认证、加密、认证前加密和认证后加密。

在**情况 2** 中,仅在网关间提供安全保护,主机不实现 IPSec。此时,支持简单的虚拟专用网。隧道可支持 AH、ESP 或带认证选项的 ESP。由于 IPSec 将作用于整个内部包,故而不需要隧道嵌套。

情况 3 在情况 2 的基础上增加了端到端保护。情况 1 和情况 2 讨论的组合在此都允许。网关到网关的隧道对末端系统间的所有通信提供认证和/或保密性。网关到网关的隧道 ESP 可对流量提供一定的保密性。单个主机可以根据特定应用实现任何额外的 IPSec 服务或为用户提供端到端的 SA。

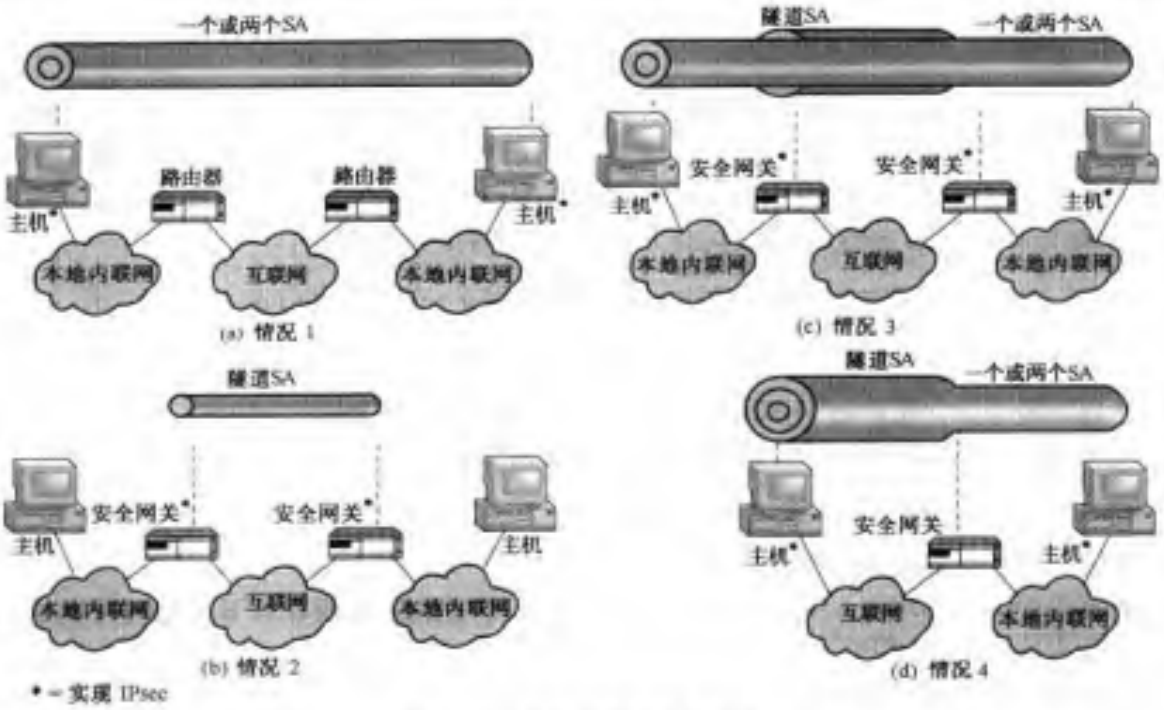


图 16.10 安全关联的基本组合

情况 4 支持远程主机使用互联网到达企业的防火墙，访问防火墙后的某些服务器或工作站。在防火墙和远程用户之间仅需要隧道模式。如情况 1 一样，可在远程主机和本地主机间使用一到两个 SA。

16.6 密钥管理

IPSec 的密钥管理包括密钥的确定和分发。典型的需求是两个应用需要四个密钥：发送和接收对的 AH 和 ESP。IPSec 体系结构文档要求支持两种密钥管理类型：

- **手动的**：系统管理员手动地为每个系统配置自己的密钥和其他通信系统密钥，应用于小规模、相对静止的环境。
- **自动的**：自动系统可以在大型分布系统中使用可变配置为 SA 动态按需创建密钥。

默认的 IPSec 自动密钥管理协议指的是 ISAKMP/Oakley，它由以下元素组成：

- **Oakley 密钥确定协议**：Oakley 提供增值安全性，是基于 Diffie-Hellman 算法的密钥交换协议。Oakley 是通用的，没有任何特别格式。
- **互联网安全关联和密钥管理协议(ISAKMP)**：ISAKMP 提供互联网密钥管理框架和特定协议支持，包括格式和安全属性协商。

ISAKMP 自身不包含特定的交换密钥算法，而是由一系列使用各种交换密钥算法的报文类型集合组成。Oakley 是 ISAKMP 的第一个版本规定使用的交换密钥算法。

16.6.1 Oakley 密钥确定协议

Oakley 是 Diffie-Hellman 交换密钥算法的细化。Diffie-Hellman 涉及用户 A 和用户 B 间的交互。在两个全局参数上首先达成一致：大素数 q 和 q 的生成元 α。A 选择一个随机整数 X_A 作为它的私钥，将其公钥 $Y_A = \alpha^{X_A}$ 模 q 传给 B。同样，B 也选择一个随机整数 X_B 作为它的私钥，将其公钥 $Y_B = \alpha^{X_B}$ 模 q 传给 A。这样，双方即可计算它们的会话密钥：

$$K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q = \alpha^{X_A X_B} \bmod q$$

Diffie-Hellman 算法有两个很好的特性：

- 仅在需要时创建密钥，不需要长时间地存放密钥，减少了泄漏的可能性。
- 除了利用全局参数达成协议外，不需要额外的交换开销。

然而，[HUIT98]指出该算法也有一些缺点：

- 不提供任何标识各方身份的信息。
- 易受中间人的攻击。第三方 C 可以在与 A 通信时冒充 B，而在与 B 通信时冒充 A。中间人攻击过程如下：
  1. B 给 A 发送公钥 Y_B(参见图 10.8)。
  2. 敌方 E 窃听到该消息，将 B 的公钥保存下来，并向 A 以 B 的用户标识发送带有 E 的公钥 Y_E 的报文。该报文伪装成从 B 的主机发送出来的形式，于是 A 接收了 E 的报文，将 E 的公钥和 B 的用户标识一起存储；同样地，E 伪装成 A 向 B 发送一个带有 E 公钥的报文。
  3. B 在 B 的私钥和 Y_E 的基础上计算会话密钥 K_1，A 在 A 的私钥和 Y_E 的基础上计算会话密钥 K_2，E 使用 E 的私钥 X_E 和 Y_B 计算 K_1，使用 X_E 和 Y_A 计算 K_2。
  4. 此后，E 就可以通过转接从 A 到 B 和从 B 到 A 的消息来获得 A 和 B 的通信内容，而 A 和 B 却无法知道他们在与 E 共享通信。
- 具有很强的可计算性。当攻击者申请许多密钥时，受害者很容易受阻塞攻击。受害者需要花费相当多的资源做无意义的乘方、取模运算。

Oakley 算法则是一种保持了 Diffie-Hellman 优点而去掉了其缺点的一种算法。

Oakley 的特性

Oakley 算法有如下五个重要特性：

1. 可以防止阻塞攻击。
2. 双方可以协商得到一个组。本质上这与 Diffie-Hellman 密钥交换的全局参数一样。
3. 使用临时交互号防止重放攻击。
4. 可以交换 Diffie-Hellman 的公钥值。
5. 对 Diffie-Hellman 交换进行认证，防止中间人攻击。

我们已经讨论过 Diffie-Hellman，再来看看一些其他属性。首先，在阻塞攻击中，攻击者伪装成合法用户的源地址，发送一个 Diffie-Hellman 密钥给受害者。受害者于是执行乘方取模运

算计算密钥，重复这类消息可以阻塞受害者的机器，使其系统无法正常工作。**cookie 交换**要求各方在发给对方确认的初始报文中发送一个伪随机数 cookie，此确认必须在 Diffie-Hellman 密钥交换的第一个报文中重复。如果源地址是伪造的，则攻击者不能得到该 cookie。因此，攻击者只能要求用户生成确认报文，但不可能要求用户执行 Diffie-Hellman 计算。

ISAKMP 要求 cookie 的生成应满足三个基本需求：

1. cookie 必须与特定的通话方相关。这样可以防止攻击者使用合法的 IP 地址和 UDP 端口获得 cookie 后，将它用于其他 IP 地址和端口。
2. 被某个实体承认的 cookie 只能由其发行实体生成，不可能由其他实体生成，使得发行实体使用本地秘密信息生成 cookie，继而验证它。并且，该秘密信息不可能从其他 cookie 中推导出来。其本质在于发行实体不需要保存它所发行的 cookie，当在需要时能验证收到的 cookie，从而降低了被发现的可能性。
3. cookie 的生成和验证方法必须足够快，以防范企图占用处理器资源的攻击。

推荐创建 cookie 的方法是，根据 IP 的源地址、目的地址和 UDP 的源端口、目的端口以及一个本地生成的秘密值快速生成 hash 值(如 MD5)。

Oakley 支持在 Diffie-Hellman 密钥交换时使用不同的组，每组包含两个全局参数和算法标识。目前，规范中包括如下组：

- 768 位的模进行模取幂

$$q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149\,686)$$
$$\alpha = 2$$

- 1024 位的模进行模取幂

$$q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129\,093)$$
$$\alpha = 2$$

- 1536 位的模进行模取幂
  - 参数待定
- 2^{155} 的椭圆曲线组
  - 生成子(十六进制)：X = 7B，Y = 1C8
  - 椭圆曲线参数(十六进制)：A = 0，Y = 7338F
- 2^{185} 的椭圆曲线组
  - 生成子(十六进制)：X = 18，Y = D
  - 椭圆曲线参数(十六进制)：A = 0，Y = 1EE9

前三种是使用模取幂的传统 Diffie-Hellman 算法，后两种是使用椭圆曲线模拟 Diffie-Hellman，参见第 10 章。

Oakley 使用 **nonce** 来防止重放攻击。每个 nonce 是一个本地生成的伪随机数，nonce 在应答中出现，并在交换的特定部分加密以对它进行保护。

在 Oakley 中使用了三种认证方法：

- **数字签名**：对双方均可取到的 hash 进行签名来验证交换，各方用自己的私钥加密 hash。hash 在生成时使用重要的参数，如用户 ID、nonce 等。

- **公钥加密**:使用发送方的私钥对参数如 ID、nonce 加密来验证交换。
- **对称密钥加密**:使用其他方法传送密钥,再使用该密钥和对称加密算法对交换信息加密来验证交换。

Oakley 交换实例

Oakley 规范包括许多协议允许的交换实例,这里将以规范中称为"主动密钥交换"的实例为例,因为其中只涉及到三次报文交换。

图 16.11 描述了一个主动密钥交换协议。第一步,发起者 I 发送了一个在组中使用的 cookie 和 I 为此次交换准备的 Diffie-Hellman 公钥,同时 I 还声明在此次交换中使用的公钥加密算法、hash 算法和认证算法,以及 I 和响应者 R 的标识、此次交换的 I 的 nonce。最后,I 使用 I 的私钥对两个标识、nonce、组、Diffie-Hellman 公钥和提供的算法进行签名,并将签名附于其后。

$I \rightarrow R$: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

$R \rightarrow I$: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

$I \rightarrow R$: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

记号:

I = 发起者
R = 响应者
CKY_I, CKY_R = 发起者和响应者的 cookie
OK_KEYX = 密钥交换报文类型
GRP = 此交换的 Diffie-Hellman 组的名字
g^x, g^y = 发起者和响应者的公钥;g^{xy} = 此交换的会话密钥
EHAO, EHAS = 提供和选择的加密、散列、认证函数
NIDP = 声明报文的剩余部分没有加密
ID_I, ID_R = 发起者和响应者的标识
N_I, N_R = 此次交换发起者和响应者的随机数 nonce
$S_{KI}[X]$, $S_{KR}[X]$ = 表明对 X 的签名使用的是发起者和响应者的专用签名密钥

图 16.11 主动 Oakley 密钥交换实例

当 R 接收到报文时,R 使用 I 的公开签名密钥验证签名,然后,R 将从报文中得到的 I 的 cookie、标识和 nonce 作为对此报文的应答,同时在报文中包含一个 cookie、R 的 Diffie-Hellman 公钥、所选的算法(包含在提供的算法中)、R 的标识、R 为此次交换准备的 nonce。最后,R 使用 R 的私钥对两个标识、nonce、组、两个 Diffie-Hellman 公钥和选择的算法进行签名,并将签名附于其后。

当 I 收到第二个报文时,I 使用 R 的公钥验证签名。报文中的 nonce 值确保了这不是一个对旧消息的重放。为了完成交换,I 必须发送一个应答消息给 R,证实 I 已经收到了 R 的公钥。

16.6.2 ISAKMP

ISAKMP 定义建立、协商、修改和删除安全关联的过程和包格式。ISAKMP 定义了生成交换密钥的载荷和认证数据。载荷的格式提供了与特定密钥交换协议、加密算法和认证机制无关的一致性框架。

ISAKMP 头格式

ISAKMP 报文由 ISAKMP 头和一个或多个载荷组成，并包含在传输协议之中。规范指明了在实现时必须在传输协议中支持 UDP。

图 16.12(a)指明了 ISAKMP 报文的头格式，它由以下域组成：

- **发起者 cookie(64 位)**：发起 SA 创建、SA 通知或 SA 删除的实体的 cookie。
- **响应者 cookie(64 位)**：应答实体的 cookie，在发起者的第一个报文中为空。
- **邻接载荷(8 位)**：表明报文中第一个载荷的类型，载荷将在下一小节中讨论。
- **主版本(4 位)**：使用的 ISAKMP 的主版本。
- **从版本(4 位)**：使用的从版本。
- **交换类型(8 位)**：表明交换类型，在本节中稍后讨论。
- **标志(8 位)**：ISAKMP 交换的可选项集合，目前定义了两位：当跟在头后面的所有载荷都使用此 SA 的加密算法加密后，则设置加密位；在 SA 创建完成之前没有接收到任何加密消息时设置提交位。
- **报文标识(32 位)**：报文的惟一标识。
- **长度(32 位)**：报文(头 + 所有载荷)的总字节长度。

ISAKMP 载荷类型

所有 ISAKMP 载荷开始于如图 16.12(b)所示的载荷头，而报文中最后一个载荷的邻接载荷域的值为 0。载荷长度域标明载荷头的该载荷字节长度。

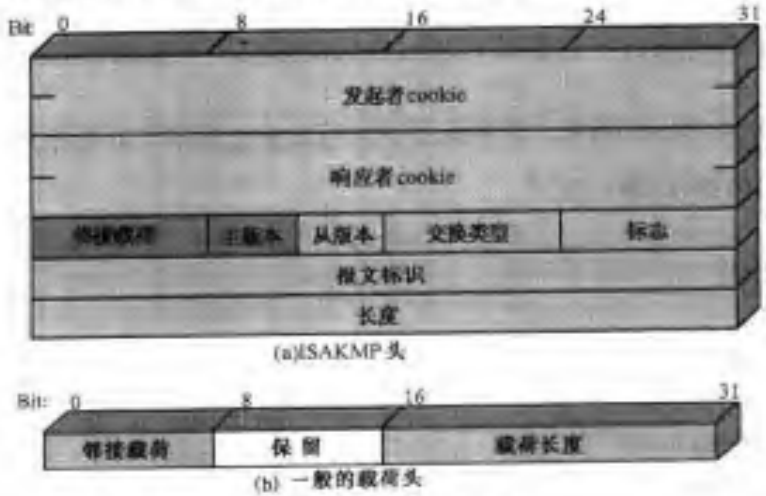


图 16.12　ISAKMP 格式

表 16.3 总结了在 ISAKMP 中定义的载荷类型，列举了每种载荷的部分域和参数。**SA 载荷**用于开始创建一个 SA，其中的解释域参数定义了协商发生的 DOI 标识，如 IPSec DOI；位置参数定义了协商使用的安全策略，而加密和保密的安全级别可以规定(如敏感级、安全间隔等)。

表 16.3 ISAKMP 载荷类型

类 型	参 数	描 述
安全关联(SA)	解释和位置域	用于协商安全属性和表明协议发生的 DOI 和位置
建议(P)	建议号、协议标识、SPI 大小、转换号、SPI	用于 SA 协商中,标明使用的协议和转换数
转换(T)	转换号、转换标识、SA 属性	用于 SA 协商中,标明转换和相关的 SA 属性
密钥交换(KE)	密钥交换数据	支持各种密钥交换技术
标识(ID)	标识类型、标识数据	用于交换标识信息
证书(CERT)	证书编码、证书数据	用于传输证书和其他与证书相关的信息
证书请求(CR)	证书类型号、证书类型、证书认证号、认证中心	用于请求证书,标明所请求证书的类型和可接受的认证中心
散列(HASH)	散列数据	散列函数生成的数据
签名(SIG)	签名数据	数字签名函数生成的数据
Nonce(NONCE)	*Nonce* 数据	包含 nonce
通知(N)	DOI、协议标识、SPI 大小、通知报文类型、SPI、通知数据	用于传输通知数据,如出错条件
删除(D)	DOI、协议标识、SPI 大小、SPI 号、一个或多个 SPI	标明一个 SA 不再合法

建议载荷包含在 SA 协商中需要使用的信息。该载荷表明了 SA(ESP 或 AH)协商使用的协议、服务和机制。此载荷还包括发送实体的 SPI 和转换次数。每个转换包含在转换载荷之中。发起者可以通过使用多个转换载荷来提供多种可能性,而响应者可以从中选择一个或予以拒绝。

转换载荷定义了用于保护指定协议通信通道的安全转换方式,参数转换号用于标识该载荷,使得响应者可以使用它来表示接受了该转换方式;转换标识和属性域标识了特定的转换(如 ESP 使用的 3DES、AH 使用的 HMAC-SHA-1-96)和与之相联系的属性(散列长度)。

密钥交换载荷可以被各种密钥交换技术使用,包括 Oakley、Diffie-Hellman 和 PGP 使用的基于 RSA 的密钥交换。密钥交换数据域包括生成会话密钥所需的数据,与所使用的密钥交换算法相关。

标识载荷用于确定通信方的标识和使用的认证信息。一般地,标识数据域包含 IPv4 或 IPv6 地址。

证书载荷用于传送公钥证书。证书编码域标明证书类型或与证书相关的如下信息:

- PKCS # 7 包装的 X.509 证书
- PGP 证书
- DNS 签名密钥
- X.509 签名证书
- X.509 密钥交换证书
- Kerberos 令牌
- 证书撤销表(CRL)
- 权利撤销表(ARL)
- SPKI 证书

在任何 ISAKMP 交换中,发送方可以使用证书请求载荷去请求其他通信实体的证书。载荷必须列举可接受的多种证书类型和可接受的多个认证中心 CA。

散列载荷包含由散列函数根据部分报文和/或 ISAKMP 状态生成的数据。此载荷用于验证报文中数据的完整性或认证正在与之对话的实体。

签名载荷包含由数字签名函数根据部分报文和/或 ISAKMP 状态生成的数据。此载荷用

于验证报文中数据的完整性或提供不可抵赖服务。

Nonce 载荷包含用于交互的随机数据，以防止重放攻击。

通知载荷包含与 SA 或 SA 协商相关的出错或状态信息，ISAKMP 定义了以下出错报文：

无效载荷类型	无效协议标识	无效认证编码
不支持的 DOI	无效 SPI	无效认证
不支持的情况	无效变换标识	无效认证请求语法
无效 cookie 程序	不支持的属性	无效认证权利
无效主版本	不建议的选择	无效散列信息
无效从版本	无效建议语法	认证失败
无效交换类型	载荷残缺	无效签名
无效标志	无效密钥信息	地址通知
无效报文标识		

目前定义的 ISAKMP 状态报文只有连接报文。另外，还使用了一些 ISAKMP 通知、DOI 通知。IPSec 定义了如下的状态报文：

- **响应者生命期**：响应者选择的 SA 生命期。
- **重放状态**：响应者选择是否执行反重放检测的确认。
- **初始联系**：通知对方这是与远程系统建立联系的第一个 SA，收到这个通知的接收方可以假设发送系统重新启动，不再需要以前的 SA，并从该系统中删除。

删除载荷表明发送方将一个或多个 SA 从它的数据库中删除，从而不再合法。

ISAKMP 交换

ISAKMP 提供了一个带有载荷类型的报文交换框架，定义了五种默认的交换类型，如表 16.4所示。表中，SA 指与协议和转换载荷相关的 SA 载荷。

表 16.4　ISAKMP 交换类型

交　换	注　释
(a) 基本交换	
(1) I→R：SA；NONCE	开始 ISAKMP-SA 协商
(2) R→I：SA；NONCE	基本 SA 同意生成的密钥
(3) I→R：KE；ID_I；AUTH	生成密钥；响应者验证的发起者标识
(4) R→I：KE；ID_R；AUTH	发起者验证的响应者标识；生成密钥；SA 建立
(b) 标识保护交换	
(1) I→R：SA	开始 ISAKMP-SA 协商
(2) R→I：SA	基本 SA 同意生成的密钥
(3) I→R：KE；NONCE	生成密钥
(4) R→I：KE；NONCE	生成密钥
(5)* I→R：ID_I；AUTH	响应者验证的发起者标识
(6)* R→I：ID_R；AUTH	发起者验证的响应者标识；SA 建立
(c) 单认证交换	
(1) I→R：SA；NONCE	开始 ISAKMP-SA 协商

（续表）

交　换	注　释
(2) R→I:SA; NONCE; ID_R; AUTH	基本 SA 建立；发起者验证的响应者标识
(3) I→R:ID_I; AUTH	响应者验证的发起者标识；SA 建立
(d) 主动交换	
(1) I→R:SA; KE; NONCE; ID_I	开始 ISAKMP-SA 协商和密钥交换
(2) R→I:SA; KE; NONCE; ID_R; AUTH	响应者验证的发起者标识；生成密钥；基本 SA 建立
(3) * I→R:AUTH	发起者验证的响应者标识；SA 建立
(e) 信息交换	
(1) * I→R:N/D	出错或状态通知或删除

记号：

I = 发起者

R = 响应者

* = 加密 ISAKMP 报头后的载荷

基本交换允许将密钥交换和认证材料一起传送，减少了交换的次数，但代价是不提供标识保护。头两个报文提供 cookie，建立达成一致协议和转换的 SA，双方使用 nonce 阻止重放攻击。最后两个报文交换密钥和用户标识，使用从头两个报文中得到的认证密钥、标识和 nonce 的认证载荷。

标识保护交换通过提供用户标识扩展了基本交换。头两个报文建立 SA，接下来的两个报文使用 nonce 提供反重放保护，执行密钥交换，一旦计算出会话密钥，双方则交换包括数字签名、公钥证书等内容的认证信息。

单认证交换用于执行双方认证而不进行密钥交换。头两个报文建立 SA。另外，响应者使用第二个报文传输它的标识，并使用认证保护。发起者发送第三个报文传送其认证标识。

主动交换不提供标识保护，从而交换的次数最少。第一个报文由发起者提供一个带协议和转换选项的 SA，同时开始密钥交换并提供它的标识。在第二个报文中，响应者在确认接收到 SA 的报文时，表明它所接收的协议和转换，完成密钥交换和对传送信息的认证。第三个报文由发起者使用共享的会话密钥对先前接收的信息加密，发回认证结果。

信息交换用于单向传输 SA 管理信息。

16.7 推荐读物和网址

[STAL00]详细叙述了 IPv6 和 IPv4。有关 IPv4 和网络互联的读物见[COME00]和[STEV94]。[HUIT98]从技术性上阐述了构成 IPv6 规范的各种 RFC，这本书讨论了各种特征的目的和协议的操作。[MILL98]重点介绍 IPv6 的实现情况。[CHEN98]很好地讨论了 IPSec 设计。[FRAN01]和[DORA99]是 IPSec 的综合读物。

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.

COME00 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture*. Upper Saddle River, NJ: Prentice Hall, 2000.

DORA99 Doraswamy, N., and Harkins, D. *IPSec*. Upper Saddle River, NJ: Prentice Hall, 1999.

FRAN01 Frankel, S. *Demystifying the IPSec Puzzle*. Boston: Artech House, 2001.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

MILL98 Miller, S. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall, 1998.

STAL00 Stallings, W. *Data and Computer Communications*, 6th edition. Upper Saddle River, NJ: Prentice Hall, 2000.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1 : The Protocols*. Reading, MA: Addison-Wesley, 1994.

推荐网址：

- **IP Security Protocol（ipsec）Charter**：最新的 RFC 和有关 IPSec 的草案。
- **IP Security Working Group News**：工作组文档、邮件文件、相关技术论文和其他一些有价值的材料。
- **IP Security（IPSEC）Resources**：应用 IPSec 的公司列表，应用调查和其他一些有价值的材料。

16.8 关键术语、思考题和习题

16.8.1 关键术语

抗重放服务
认证头(AH)
载荷安全封装（ESP）
互联网安全关联和
密钥管理协议（ISAKMP）
IPSec
IPv4
IPv6
Oakley 密钥确定协议
重放攻击
安全性
安全关联(SA)
传输模式
隧道模式

16.8.2 思考题

16.1 给出 IPSec 的一个应用实例。

16.2 IPSec 提供哪些服务？

16.3 哪些参数标识 SA，哪些参数刻画一个特定 SA 的本质？

16.4 指出传输模式和隧道模式的区别？

16.5 什么是重放攻击？

16.6 为什么 ESP 含填充域？

16.7 形成 SA 束有哪些基本方法？

16.8 IPSec 中 Oakley 密钥确定协议和 ISAKMP 的作用是什么？

16.8.3 习题:

16.1 在讨论 AH 的处理中,曾经提到,并非 IP 报头中的所有域都参与 MAC 的计算。

a. 对于 IPv4 头的每个域,指明哪些是不变的,哪些会变但可以预计、哪些是随意的(在 ICV 计算前置 0)。

b. 对 IPv6 头的每一个域完成同样的工作。

c. 对 IPv6 扩展头的每一个域完成同样的工作。

在每种情况下,判断你对每个域的理解。

16.2 当使用隧道模式时,构造一个新的外部 IP 报头。IPv4 和 IPv6 都指明了外部包的外部 IP 报头的每个域和每个扩展头与内部 IP 包对应域或扩展头的关系。请指出哪些外部数据是从内部数据继承的,哪些是由独立于内部数据重新构造的?

16.3 端对端地认证和加密应该在两台主机之间。参照图 16.6 和图 16.9 画图表示:

a. 传输邻接,认证前加密。

b. 一个隧道 SA 中有一个传输 SA,认证前加密。

c. 一个隧道 SA 中有一个传输 SA,加密前认证。

16.4 IPSec 构架文档中说,当两个传输模式 SA 被捆绑,在同一个端对端流中允许 AH 和 ESP 两种协议,看起来只有一种安全协议比较合适:先实施 ESP 协议再实施 AH 协议。为什么不推荐先认证后加密?

16.5 **a.** 哪一种 ISAKMP 交换类型(见表 16.4)与主动 Oakley 密钥交换对应?

b. 对于 Oakley 主动密钥交换,指出每个消息的哪个参数与 ISAKMP 载荷类型一一对应?

附录 16A 互联网络和互联网协议

这个附录概括了互联网协议。我们首先总结互联网协议在互连网络中的主要作用,然后介绍 IPv4 和 IPv6。

互联网协议的作用

互联网协议为末端系统通过多种网络互连提供功能。为了这个目的,IP 被应用于所有连接网络的末端系统和路由器。源端的高层数据被封装为 IP 协议数据单元(PDU)进行传输。然后 PDU 途径多个网络和路由器到达目的端。

路由器必须能够兼容各种网络,包括:

- **编址方式**:不同的网络可能使用不同的编址方式。例如:IEEE 802 LAN 为其设备指定 16 位或 48 位地址;X.25 公共包交换网络使用 12 位十进制地址(48 位地址,每 4 位一个数字)。除了目录服务以外,还需要一个全局的编址方式。
- **最大包长度**:包从一个网络传到另一个网络时,需要被切分成多个小包,这个过程称为拆分。例如,以太网允许的最大包长度为 1500 字节,而 X.25 最大允许 1000 字节。一个包从以太网传向 X.25 时,可能被拆分成两个包。
- **接口**:软硬件接口是依网络不同而异的。路由器的概念独立于这些区别。

- **可靠性**:异种网络可能提供从可靠的端对端虚电路到不可靠的服务。路由器的操作不应假设网络的可靠性。

路由器的操作如图16.13所示,这种操作依赖于互联网协议。例如,TCP/IP协议中的IP协议可以完成该功能。IP必须在网络的所有终端系统和路由器中实现,每个终端必须兼容IP上层协议才能成功通信。而中间路由器可以只实现IP。

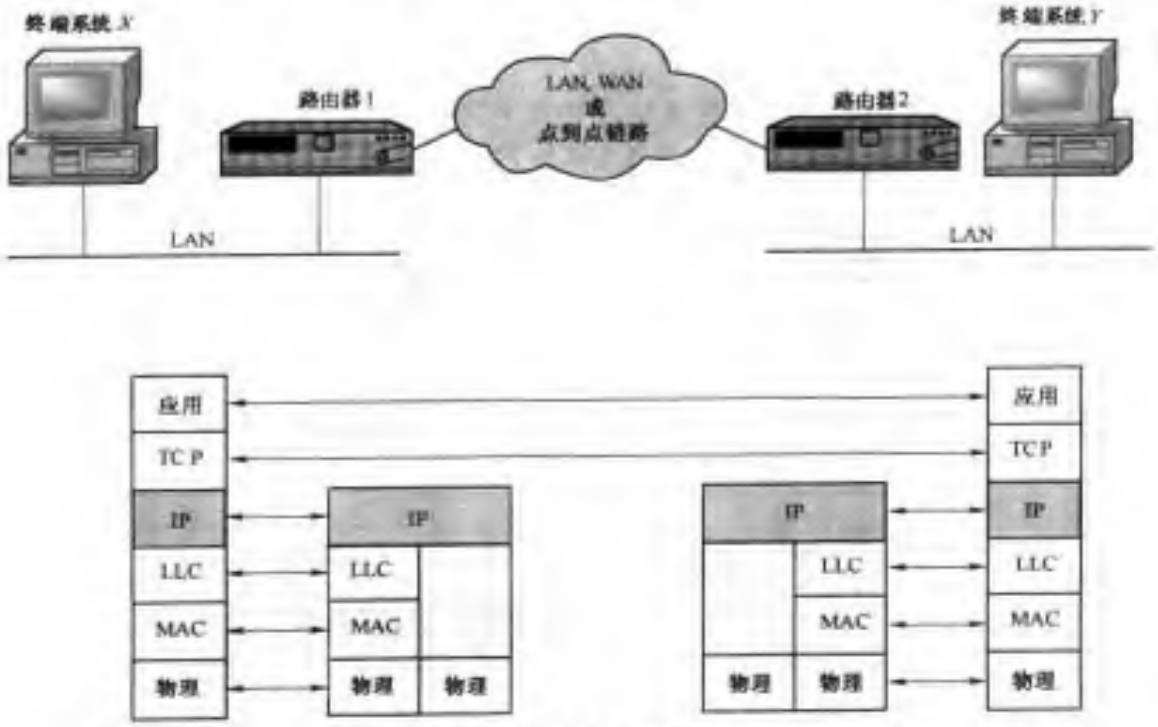


图16.13 TCP/IP的结构

考虑如图16.13所示的终端 X 向终端 Y 传送数据块。X 中的IP从TCP中接收将要传送给 Y 的数据,IP层给它附加一个带有 Y 的全局互联网地址(网络标识和终端标识),形成一个IP包;接着,IP识别出 Y 属于另一个子网,于是首先将包发往路由器,如图16.13中的路由器1。为了完成这个操作,IP将带有正确地址信息的数据单元下传给LLC,LLC创建一个LLC PDU,再传给MAC层,由MAC层构造一个包头带有路由器1地址的MAC包。

其次,包通过局域网到达路由器1,该路由器分离该包、LLC头、尾,并分析IP报头得到数据的最终地址,在此例中为 Y,然后,该路由器做出一个路由决策,这有两种可能性:

1. 目的终端系统 Y 直接与该路由器相连的一个子网相连。
2. 为了到达目的地,必须遍历一个或多个其他的路由器。

在此例中,包在到达目的地之前必须经过路由器2。于是路由器1通过中间网络将IP包送往路由器2。为了达到这个目的,需要用到那个网络的协议,例如,如果该中间网络是一个X.25网络,则IP数据必须被封装到一个X.25包中,带上到达路由器2的合适的地址信息。当包到达路由器2时,去掉包头,路由器发现该包的目的地 Y 的子网直接与它相连。因此,路由器创建一个带有目的地址 Y 的包,然后直接通过局域网发送。包最后到达 Y,然后去掉包、

LLC 和互联网头得到数据。

IP 提供的这个服务是一种不可靠服务。也就是说，IP 不保证所有的数据都会被传送，或所传送的数据都会按照正确的顺序到达。这就需要上层协议 TCP 来发现并恢复这些错误。这种方法带来了很大的机动性，因为不保证传送，不需要对子网提出任何特别的可靠性要求，使得协议可以与任何类型的子网相连；不保证传送顺序使得相继的包可以通过互联网上的不同路径传送，允许协议在互联网网络发生阻塞和出错时可以重新选择路由。

IPv4

数十年来，TCP/IP 协议结构的里程碑是 IP 协议的第四版。图 16.14(a)显示了 IP 报头的格式，它至少包含 20 个字节或 160 位。其各个域如下所示：

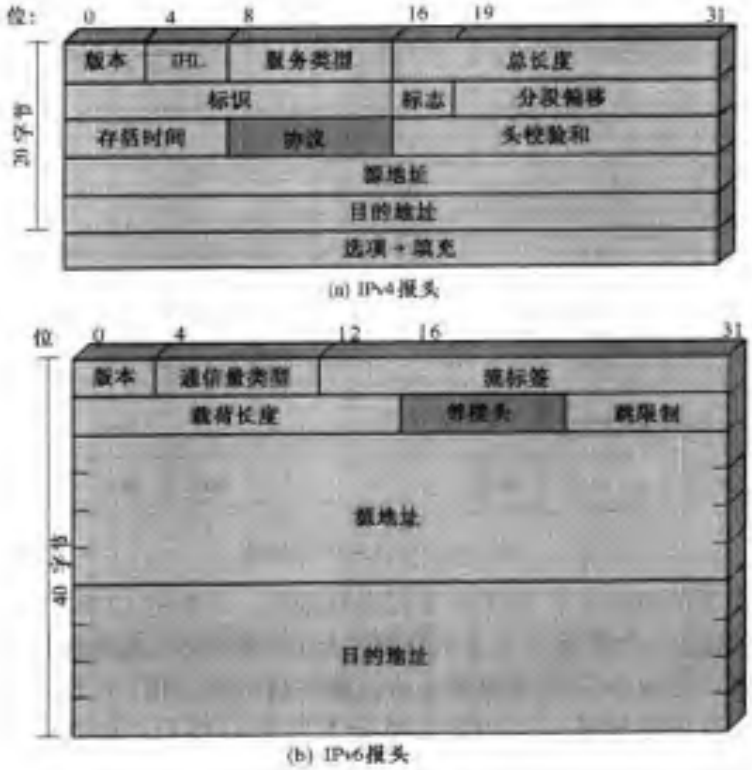


图 16.14 IP 报头

- **版本(4 位)**：表明版本号，允许协议升级，值为 4。
- **互联网头长度(IHL)(4 位)**：32 位字长的头长度，最小值为 5，表示最小头长度为 20 个字节。
- **服务类型(8 位)**：为末端系统的 IP 模块提供指导，根据包的相对优先级为包提供路由。
- **总长度(16 位)**：总的 IP 包字节长度，以 8 位组为单位。
- **标识(16 位)**：结合源地址、目的地址和用户协议的一个序列号，用于惟一标识数据包。当该包在互联网中存在时，该包的标识对源地址、目的地址和用户协议应该是惟一的。
- **标志(3 位)**：目前只定义了 2 位。当包被分段时，More 位表示该包是否为原包的最后一个分段。不允许分段位设置后将禁止分段，当知道目的节点没有重组分段的能力时，使用此位。

然而，一旦设置了该位，当该包的长度超过了所经子网的最大允许长度时，数据包会被丢弃。因此，如果设置了该位，必须使用源路由避免经过具有最小包长度的子网。

- **分段偏移(13 位)**：表明该分段在原包中的位置，以 64 位为单位，这隐含除了最后一个分段外，其余分段必须包含 64 位的整数倍长度的数据。
- **存活时间(TTL)(8 位)**：规定包在互联网中存在的最长时间，以秒为单位。每个转发的路由器至少将 TTL 值减 1，使得 TTL 在某种程度上与跳数相似。
- **协议(8 位)**：表明邻接的上层协议，此域用于目的端接收数据，因此，该域标识了该包中紧接在 IP 报头后面的邻接头。
- **头校验和(16 位)**：仅应用于报头的检错代码，由于有些头域在传送过程中可以改变(如存活时间、分段域)，因此该域在每个路由器处都要重新验证、重新计算。校验和是头中的所有 16 位字之和的补码。为了便于计算，校验和域的初值为 0。
- **源地址(32 位)**：按照各种网络和终端允许的方式编码(7 和 24 位、14 和 16 位、21 和 8 位)。
- **目的地址(32 位)**：与源地址相同。
- **选项(变量)**：对发送方要求的选项编码，可以包括安全标签、源路由、记录路由和时间戳。
- **填充(变量)**：用于确保包头的长度是 32 位的整数倍。

IPv6

1995 年，为互联网开发协议标准的 IETF 提出了下一代 IP 的规范，名为 IPng。这个规范在 1996 年成为 IPv6 标准。IPv6 提供了许多在现有 IP 上增强的功能，以适应现代网络高速且日益流行的包括音频、视频的多种数据流。但开发新协议的推动力量是需要更多的地址空间。IPv4 使用 32 位地址指明源和目的地址。随着互联网的爆炸式发展和专用网络连入互联网，地址长度已经不能为所有需要地址的系统提供足够的空间。如图 16.14(b)所示，IPv6 有 128 位地址。最终，所有使用 TCP/IP 的应用都要从现有 IP 移植到 IPv6，但这个过程可能需要许多年。

IPv6 报头

IPv6 报头定长为 40 字节，包含以下各域[如图 16.14(b)所示]：

- **版本(4 位)**：表明版本号，允许协议升级，值为 6。
- **流量类型(8 位)**：源节点和/或路由器用来标识和区分不同类型和优先级的 IPv6 包，此域的使用还在研究之中。
- **流标签(20 位)**：可以用于主机标志需要路由器进行特殊处理的包。流标签可以辅助资源预留和实时流量处理。
- **载荷长度(16 位)**：IPv6 包的包头后所跟的剩余部分的字节长度，也就是说，是所有扩展头加传输层 PDU 的总长度。
- **邻接头(8 位)**：标识紧接在 IPv6 报头后的头类型，可以是一个 IPv6 扩展头或一个高层协议的头，如 TCP 或 UDP。
- **跳限制(8 位)**：包所允许的剩余跳数，跳限制被源端设置为某个最大值，网络节点每转发一次减 1，当跳限制的值到达 0 时，包即被丢掉。
- **源地址(128 位)**：包的源节点地址。

- **目的地址(128 位)**:包的接收方的地址,如果存在路由扩展头,该地址可以不是最终的目的地址。

虽然 IPv6 报头比 IPv4 报头长(40 字节对 20 字节),但包含的域少(8 对 12),路由器对每个头所进行的处理要少一些,从而加速路由。

IPv6 扩展头

一个 IPv6 包可以包含 IPv6 报头和一个或多个扩展头,IPSec 定义了如下扩展头:

- **逐跳选项头**:定义需要跳处理的选项。
- **路由头**:提供附加的路由,与 IPv4 的源路由相似。
- **分段头**:包含分段和重组信息。
- **认证头**:提供包的完整性和认证。
- **封装安全载荷头**:提供秘密性。
- **目的选项头**:包含被目的节点检查的可选信息。

IPv6 标准声称,当使用了多个扩展头时,IP 报头按如下顺序出现:

1. IPv6 报头:必要且必须出现在第一项。
2. 逐跳选项头。
3. 目的选项头:用于处理出现在 IPv6 目的地址域的第一个目的地址和路由头中列出的后继地址列表的选项。
4. 路由头。
5. 分段头。
6. 认证头。
7. 封装安全载荷头。
8. 目的选项头:仅用于处理包的最终目的地址的选项。

图 16.15 是一个包含每个非安全头的 IPv6 包的例子。注意,每个 IPv6 报头和扩展头中都包含一个邻接头域,此域标识了下一个头的类型,如果下一个头是扩展头,则在此域中填充该头的类型标识;否则,即填写使用 IPv6 的上层协议的协议标识(传输层协议),与在 IPv4 协议中使用的值相同。在图 16.15 中,上层协议为 TCP,因此,IPv6 包所带的上层数据是 TCP 报头后接一个应用数据块。

逐跳选项头携带的是可选信息,如果存在,则将会被途径的所有路由器检测。该头中包含的域如下:

- **邻接头(8 位)**:标识紧接在此头后面的头的类型。
- **头扩展长度(8 位)**:以 64 位为单位的头长度,不包括第一个 64 位。
- **选项**:包含一个或多个选项,每个选项由三个子域组成:表明选项类型的标志、长度和值。

到目前为止,只定义了一个选项,即 Jumbo 载荷选项,它用于发送载荷长度大于 $2^{16} - 1 = 65\,535$个字节的 IPv6 包。此选项的可选数据域长度为 32 位,表示包含 IPv6 头的包的字节长度。对这种包而言,IPv6 报头的载荷长度域必须设置为 0 且不包含分段头。使用这个选项,

IPv6 支持总长达 4 亿字节的包，便于传送大的视频包，并使得 IPv6 能最大限度地利用传输介质提供的能力。

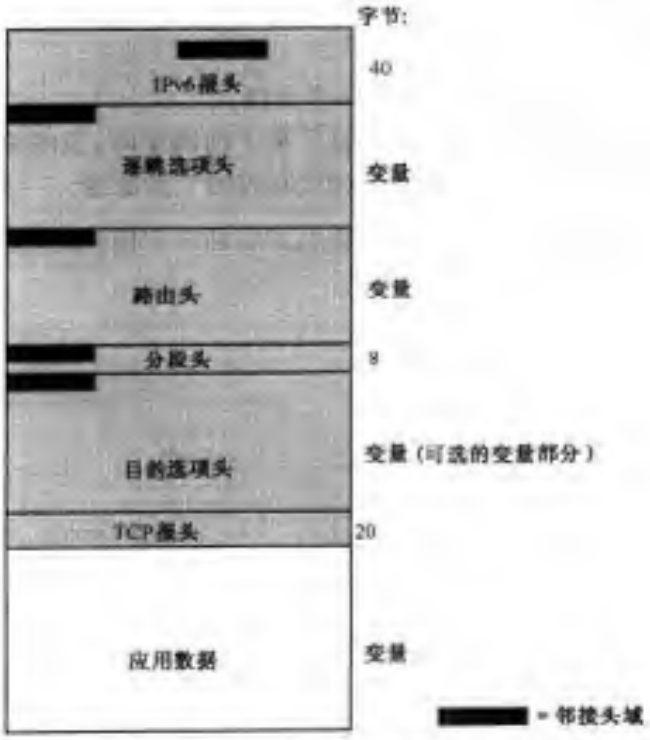


图 16.15 带扩展头的 IPv6 包(包含 TCP 段)

路由头包含一个或多个到达包目的节点所需经过的中间节点列表。所有路由头开始于四个 8 位域组成的 32 位数据块，后接某种特定路由类型的路由数据，这四个 8 位域分别为邻接头、头扩展长度、路由类型以及剩余分段：

- **路由类型**：标识特定路由头变量，如果路由器无法识别该路由类型值，则它必须丢弃此包。
- **剩余分段**：在到达目的节点前还需要访问的精确的中间节点数量。

除了这种一般的头定义外，IPv6 规范还定义了 0 类型路由头。当使用 0 类型路由头时，源节点不将最终目的地址放入 IPv6 报头，而将其放在路由头列表的最后，在 IPv6 报头中放入路径上第一个路由的目的地址。当到达 IPv6 报头所指定的节点后再检查路由头，根据它更新 IPv6 报头和路由头，并将包继续转发。更新的内容包含将下一个要访问的地址送入 IPv6 报头中，将路由头中的剩余分段值减 1。

IPv6 要求 IPv6 节点在接收到包含路由头的包时逆向路由，向发送方返回一个包。

当源节点需要分段时使用**分段头**。在 IPv6 中，只有源节点可以执行分段操作，而传输包的路由器不可以对包进行分段。为了充分利用互联网环境的特点，节点必须能执行路径发现算法，以找到路径上的各子网所能传送的最大传输单元(MTU)，也就是使用路径发现算法找出路径中瓶颈子网的 MTU，使得源节点能按每个给定目的节点的需要分段。否则，源节点必须将包长度限制在任何子网必须支持的 1280 字节以内。

除邻接头域外，分段头中还必须包含以下域：

- **分段偏移(13 位)**：表明此分段的载荷在原包中的位置，必须以 64 位为单位，也就是说，除了最后一个分段外，所有分段必须包含 64 位的整数倍长度的数据域。
- **保留(2 位)**：备用。
- **M 标志(1 位)**：1 = 更多分段；0 = 最后一个分段。
- **标识(32 位)**：惟一标识原数据包。在包存在于网络中时，包的标识必须惟一。具有同一个源节点和目的节点且标识相同的分段在目的节点重组。

目的选项头包含可选信息，如果存在，仅由包的目的节点检查此项。此头的格式与逐跳选项头相同。

第 17 章　Web 安全性

实际上，几乎所有的商业企业、大多数政府机构和许多个人都有 Web 站点。访问互联网的个人和公司的数量增长速度非常快，且他们均使用图形界面的 Web 浏览器。因此，许多公司都热衷于在 Web 上进行电子商务。但是，现实情况是互联网和 Web 容易受到攻击。商家越来越意识到这种现实，因此安全 Web 服务应运而生。

Web 安全性非常广泛，很容易写满一本书（本章后面推荐了一些书）。本章首先讨论 Web 安全性的普遍需求，然后集中讨论两种应用与 Web 商业的标准模式：SSL/TLS 和 SET。

17.1　Web 安全性思考

WWW 本质上是一种运行于互联网和 TCP/IP 上的一种客户/服务器程序。同样地，到目前为止，本书讨论的安全工具和方法也适用于 Web 安全性。但与[GARF97]中指出的一样，Web 带来了与一般计算机和网络安全不太一样的挑战：

- 互联网是双向的。与传统的发布环境不同，电子发布系统将涉及到文字电视广播、语音回答或传真反馈等，使得 Web 服务器容易受到来自于互联网的攻击。
- Web 越来越多地作为商业合作和产品信息的发布窗口以及商务交易的平台。如果 Web 服务器被破坏，就可能发生信誉受损和金钱失窃等问题。
- 虽然 Web 浏览器非常易于使用，Web 服务器相对而言易于配置和管理，Web 内容也易于开发，但其底层的软件却非常复杂。复杂的软件可能隐藏着潜在的安全漏洞。在 Web 使用的短短历史中，各种新的和升级的系统容易受到各种各样的安全性攻击。
- Web 服务器可以作为公司或机构整个计算机系统的核心。一旦 Web 服务器被攻陷，攻击者不仅可以访问 Web 服务，也可获得与之相连的整个本地站点服务器的数据和系统访问权限。
- 通常基于 Web 服务的用户是一些突发的、未受训练的用户，这些用户不需要知道隐藏在服务背后的安全隐患，因此也没有有效防范的工具和知识。

17.1.1　Web 安全性威胁

表 17.1 总结了在使用 Web 时将要面临的一些威胁安全的类别。一种归类的方式是将它们区分为被动攻击和主动攻击：被动攻击包括在浏览器和服务器通信时窃听，获得原本被限制使用的权限；主动攻击包括伪装成其他用户、篡改客户和服务器之间的消息或篡改 Web 站点的信息。

另一种分类方法是按威胁的位置分类：Web 服务器、Web 浏览器和服务器与浏览器之间的网络通信。服务器与浏览器的安全问题是计算机系统自身的安全性问题，本书第四部分论述的系统安全性问题也适用于 Web 系统的安全性，通信的安全性则是本章将要论述的重点。

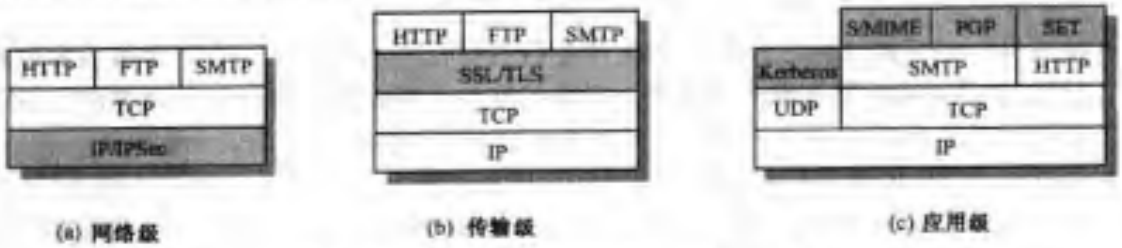
表 17.1 Web 上威胁的比较[RUBI97]

	威胁	后果	对策
完整性	• 修改用户数据 • 特洛伊木马浏览器 • 内存修改 • 传送中的消息修改	• 信息丢失 • 机器损害 • 易受所有其他威胁的攻击	加密的校验和
保密性	• 网上窃听 • 盗窃服务器数据 • 盗窃客户端数据 • 盗窃网络配置信息 • 盗窃客户端与服务器通话信息	• 信息失窃 • 秘密失窃	加密、Web 代理
拒绝服务	• 破坏用户线程 • 用假消息使机器做出 • 装满硬盘或内存 • 使用 DNS 攻击来孤立机器	• 破坏 • 干扰 • 阻止正常工作	难于防止
认证	• 伪装成合法用户 • 伪造数据	• 用户错误 • 相信虚假信息	加密技术

17.1.2 Web 流量安全性方法

现在已有许多提供 Web 安全性的方法。这些方法的使用机理是相似的，只是各自的应用范围及在 TCP/IP 协议栈中的相对位置不同。

图 17.1 说明了这种区别。提供 Web 安全性的一种方法是使用 IP 安全性[如图 17.1(a)]。使用 IPSec 的优点在于，它对终端用户和应用均是透明的，并且提供通用的解决方案。另外，IPSec 还具有过滤功能，以便仅用 IPSec 处理所选的流量。



(a) 网络级　(b) 传输级　(c) 应用级

图 17.1 TCP/IP 协议栈中安全功能的相对位置

另一种解决方案是在 TCP 之上实现安全性[如图 17.1(b)]。这种方法最先的例子是安全套接层(SSL)，接着是称为传输层安全协议(TLS)的互联网标准。此时，有两种实现方法。一般来说，SSL(或 TLS)可以作为潜在的协议对应用透明，也可以在特定包中使用，如 Netscape 和 IE 浏览器均提供 SSL，大多数 Web 服务器都实现了此协议。

特定安全服务在特定应用中得以体现。图 17.1(c)是一个示意图。这种方法的好处在于它是为给定应用定制的。在 Web 安全性方面一个典型的例子是安全电子交易(SET)。[①]

后面将讨论 SSL/TLS 和 SET。

① 图 17.1(c)表明 SET 处于 HTTP 的顶层，这是一种常见的实现方法。在另外一些实现中，SET 直接使用 TCP。

17.2　安全套接层和传输层的安全

SSL 源于 Netscape 公司。协议第 3 版在通过公开评论和工业界使用后成为互联网草案，接着在达到共识之后由 IETF 的 TLS 工作组将其开发为一般标准。目前 TLS 工作的目标是开发互联网标准，TLS 开发的第一个版本 SSLv3.1 将非常接近 SSLv3，且与 SSLv3 兼容。

本节将主要讨论 SSLv3，然后介绍 SSLv3 和 TLS 的主要区别。

17.2.1　SSL 体系结构

SSL 被设计成使用 TCP 来提供可靠的端到端安全服务。SSL 不是简单的单个协议，而是两层协议，如图 17.2 所示。

图 17.2　协议栈

SSL 记录协议(SSL Record Protocol)为高层协议提供基本的安全服务。特别是为 Web 客户端/服务器交互提供传送服务的 HTTP 协议可以在上层访问 SSL。SSL 协议上定义了三个高层协议：握手协议、修改密码规范协议和警报协议。这些 SSL 上层协议用于对 SSL 交换进行管理。

SSL 中包含两个重要概念：SSL 会话和 SSL 连接。在规范中定义如下：

- **连接**：连接是提供合适服务类型的一种传输(OSI 层次模型定义)。对 SSL 来说，连接表示的是对等网络关系，且连接是短暂的，每个连接与一个会话相关。
- **会话**：SSL 会话是一个客户端和服务器间的关联，会话是通过握手协议创建的，定义了一组多个连接共享的密码安全参数。会话可用于减少为每次连接建立安全参数的昂贵协商费用。

在多方会谈中(如客户端和服务器上的 HTTP 应用)，需要多个安全连接。从理论上说，可以在多方之间同时发生会话，但还没有在实际中使用。

每个会话实际上与多种状态相关。一旦会话建立，则进入针对读和写(如接收和发送)的当前操作状态。另外，在握手协议中，创建了读挂起状态和写挂起状态。在握手协议成功完成后，挂起状态成为当前状态。

一个会话状态由以下参数定义(见 SSL 规范)：

- **会话标识**：服务器用于标识活动的或恢复的会话状态所选的一个随机字节序列。
- **同位体证书**：同位体的 X509.v3 证书，此状态元素可以为空。
- **压缩方法**：在加密前使用的压缩数据的算法。

- **密码规范**:描述主要数据加密算法(如 null、DES 等)和计算 MAC 的散列算法(如 MD5 或 SHA-1),同时也定义如散列大小等加密属性。
- **主密码**:客户和服务器间 48 字节的共享密码。
- **可恢复性**:表明会话是否可被用于初始化新连接的标志。

连接状态可用以下参数定义:

- **服务器和客户端随机数**:服务器和客户端为每个连接选择字节序列。
- **服务器写 MAC 密码**:服务器发送数据时在 MAC 操作中使用的密码。
- **客户端写 MAC 密码**:客户端发送数据时在 MAC 操作中使用的密码。
- **服务器写密钥**:服务器加密和客户端解密数据时使用的传统加密密钥。
- **客户端写密钥**:客户端加密和服务器解密数据时使用的传统加密密钥。
- **初始化矢量**:使用 CBC 时,需要为每个密钥维护一个初始化矢量(IV)。该域首先被 SSL 握手协议初始化,其后,每个记录的最后一个密码块被保存,以作为后续记录的 IV。
- **序列号**:会话的各方为每个连接传送和接收消息维护一个单独的序列号。当接收或发送一个修改密码规范协议报文时,序列号被设为 0。序列号不能超过 $2^{64}-1$。

17.2.2 SSL 记录协议

SSL 记录协议为 SSL 连接提供两种服务:

- **保密性**:握手协议定义了加密 SSL 载荷的传统加密共享密钥。
- **消息完整性**:握手协议也定义了生成消息认证代码(MAC)的共享密钥。

图 17.3 示例了 SSL 记录协议的整个操作过程。记录协议接收一个要传送的应用消息,将其段分为块,压缩(可选),加上 MAC,加密,再加上一个 SSL 头,将得到的最终数据单元放入一个 TCP 段中。接收的数据被解密、验证、解压、重组后,再传递给更高级的用户。

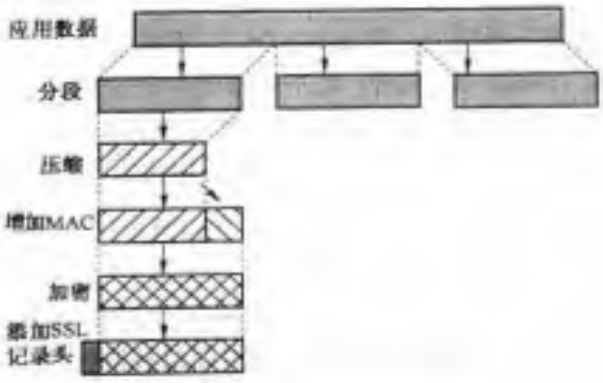


图 17.3 SSL 记录协议的操作

第一步是**分段**。每个上层消息被分成若干小于或等于 2^{14} 字节(16 384 字节)的段。接着

进行可选地压缩。压缩必须采用无损压缩方法，并且增加长度不能超过 1024 个字节。[①] 在 SSLv3(和当前的 TLS)中，没有制定压缩算法，所以默认的压缩算法为空。

接着对压缩数据计算其消息认证代码(MAC)。为此，需要使用共享密钥。其计算方式如下：

```
hash(MAC_write_secret || pad_2 ||
        hash (MAC_write_secret || pad_1 || seq_num || SSLCompressed.type ||
        SSLCompressed.length || SSLCompressed.fragment))
```

其中：

||	=	连接
MAC_write_secret	=	共享密钥
hash	=	hash 算法；MD5 或 SHA-1
pad_1	=	字节 0x36(0011 0110)，对 MD5 重复 48 次(384 位)，对 SHA-1 重复 40 次(320 位)
pad_2	=	字节 0x5C(0101 1100)，对 MD5 重复 48 次；对 SHA-1 重复 40 次
seq_num	=	此消息的序列号
SSLCompressed.type	=	处理此分段的上层协议
SSLCompressed.length	=	压缩后的分段长度
SSLCompressed.fragment	=	压缩后的分段(如果没有压缩，则为明文段)

注意，这与第 12 章定义的 HMAC 算法非常相似，其区别在于在 SSLv3 中两个填充域是连接关系，而在 HMAC 中是异或关系。SSLv3 MAC 算法基于 HMAC 的原始互联网草案，且使用连接关系。而在 HMAC 的最后版本 RFC 2104 中，使用异或关系。

接下来，将压缩消息和 MAC 用对称加密方法加密。加密对内容增加的长度不能超过 1024 个字节，以便整个长度不能超过 $2^{14} + 2048$。以下的加密算法是允许的：

分组密码		流密码	
算　法	密钥大小	算　法	密钥大小
IDEA	128	RC4-40	40
RC2-40	40	RC4-128	128
DES-40	40		
DES	56		
3DES	168		
Fortezza	80		

Fortezza 可被用于智能卡加密模式。

对流加密而言，压缩消息和 MAC 一起被加密。注意，MAC 在加密之前计算，然后将 MAC 和明文或压缩后的明文一起加密。

对分组加密而言，填充应在 MAC 之后、加密之前进行。填充的格式是一定长度的填充字节后跟一个字节的填充长度。整个填充域的长度是使得总长度(明文 + MAC + 填充域的长度)为规定的加密分组长度整数倍的最小长度。例如，明文(或如果使用了压缩则为压缩文本)长度为

① 当然，我们希望压缩能够减少数据，而不是增加数据。然而，由于压缩算法数据格式的限定，对于非常短的数据块，压缩算法的实际输出可能长于输入数据。

58 个字节，MAC 长度为 20 个字节(使用 SHA-1)，使用分组长度为 8 个字节的加密算法(如 DES)，则加上填充长度域的 1 个字节总共有 79 个字节。为了达到 8 的整数倍，需要增加一个字节的填充。SSL 记录协议的最后一步是加上一个由如下域组成的 SSL 头：

- **内容类型(8 位)**：封装段使用的高层协议。
- **主版本(8 位)**：表明 SSL 使用的主版本，如 SSLv3 的值为 3。
- **从版本(8 位)**：表明 SSL 使用的从版本，如 SSLv3 的值为 0。
- **压缩长度(16 位)**：明文段(如果使用了压缩，则为压缩段)的字节长度，最大值为$2^{14}+2048$。

已经定义的内容类型包括修改密码规范、警报、握手和应用数据。接下来讨论前三个类型。注意，在各种应用中使用 SSL 并没有什么限制，它们提供的数据内容对 SSL 来说是不透明的。

图 17.4 示例了 SSL 记录的格式。

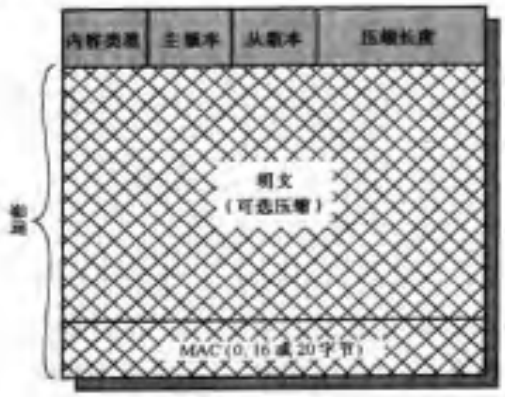


图 17.4 SSL 记录的格式

17.2.3 修改密码规范协议

修改密码规范协议是 SSL 三个特定协议之一，也是最简单的一个。协议由一个仅包含一个字节的、值为 1 的消息组成[如图 17.5(a)所示]，此消息使得挂起状态被拷贝到当前状态中，用于更新此连接使用的密码组。

17.2.4 警报协议

警报协议用于向对等实体传递 SSL 相关的警报。和使用 SSL 的其他应用一样，警报消息按照当前状态压缩和加密。

此协议的每个消息由两个字节组成[如图 17.5(b)所示]。第一个字节，值 1 表示警告，值 2 表示致命错误来传递消息出错的严重程度。如果级别为致命，则 SSL 将立即终止连接，而会话中的其他连接将继续进行，但不会在此会话中建立新连接。第二个字节包含描述特定警报信息的代码。首先，我们将通常导致致命错误的警报列举如下(见 SSL 规范定义)：

- **意外消息**：接收到不正确的消息。
- **MAC 记录出错**：接收到不正确的 MAC。
- **解压失败**：解压函数接收到不正确的输入(如不能解压或解压长度大于允许值的长度)。
- **握手失败**：发送者无法在给定选项中协商出一个可以接受的安全参数集。
- **非法参数**：握手消息中的某个域超出范围或与其他域不一致。

剩下的警报列举如下：

- **结束通知**：通知接收者，发送者将不再用此连接发送任何消息。各方在关闭连接的写端时均需发送结束通知。
- **无证书**：如果无适当的证书可用，可能作为证书请求的响应来发送。
- **证书出错**：接受的证书被破坏（如签名无法通过验证）。
- **不支持的证书**：不支持接收的证书类型。
- **证书撤销**：证书被其签名者撤销。
- **证书过期**：证书超过使用期限。
- **未知证书**：在处理证书时，出现其他错误，使得证书不被接受。

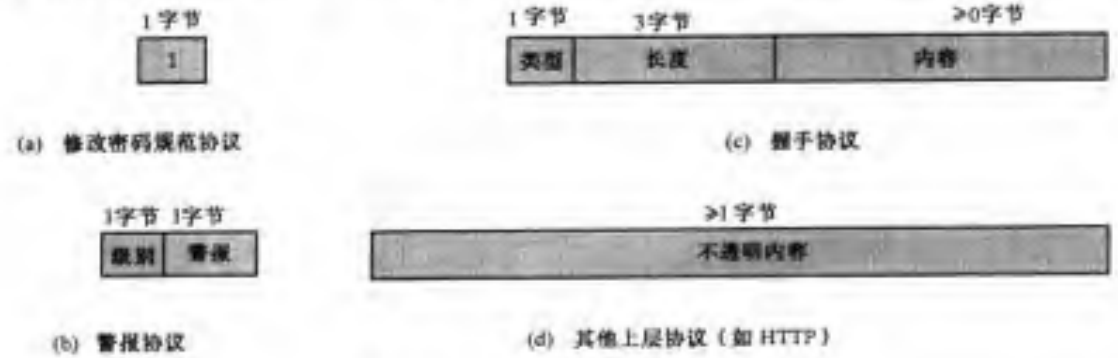


图 17.5　SSL 记录协议的有效载荷

17.2.5　握手协议

SSL 的部分复杂性来自于握手协议。此协议允许客户端和服务器端相互认证、协商加密和 MAC 算法，保护数据使用的密钥通过 SSL 记录传送。握手协议在传递应用数据之前使用。

握手协议由客户端和服务器间交换的一系列消息组成，这些消息的格式如图 17.5(c)所示。每个消息由三个域组成：

- **类型(1 字节)**：表明 10 种消息中的一种，表 17.2 列举了所定义的消息类型。
- **长度(3 字节)**：消息的字节长度。
- **内容(≥1 字节)**：与消息相关的参数，如表 17.2 所示。

表 17.2　握手协议消息类型

消息类型	参　数
hello_request	空
client_hello	版本号、随机数、会话标识、密码组、压缩方法
server_hello	版本号、随机数、会话标识、密码组、压缩方法
certificate	X.509v3 证书链
server_key_exchange	参数、签名
certificate_request	类型、认证机构
server_done	空
certificate_verify	签名
client_key_exchange	参数、签名
finished	hash 值

图 17.6 表明了在客户端与服务器之间建立逻辑连接的初始交换。此交换由四个部分组成：

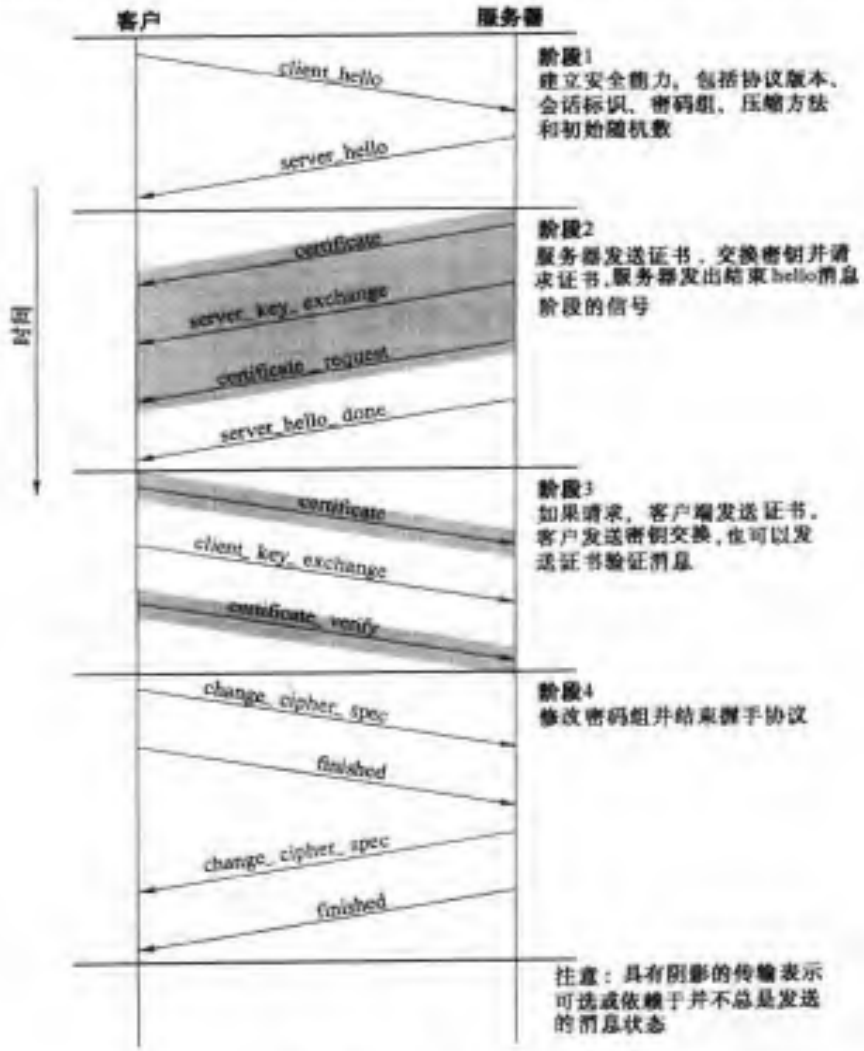


图 17.6 握手协议的处理过程

阶段 1：建立安全能力

此阶段用于建立初始的逻辑连接，并建立与之相连的安全能力。客户端发起这个交换，发送具有如下参数的 **client _ hello 消息**：

- **版本**：客户端所支持的最高 SSL 版本。
- **随机数**：由客户端生成的随机数结构，由 32 位时间戳和一个安全随机数生成器生成的 28 字节随机数组成。这些值作为 nonce，在密钥交换时防止重放攻击。
- **会话标识**：一个变长的会话标识。非 0 值意味着客户端想更新已存在连接的参数，或为此会话创建一个新的连接；0 值意味着客户端想在新会话上创建一个新连接。
- **密码组**：按优先级降序排列的、客户端支持的密码算法列表。表的每个元素定义了一个密钥交换算法和一个密码说明。

- **压缩方法**：一个客户端支持的压缩方法列表。

客户端发出消息 client _ hello 后，会等待包含与消息 client _ hello 参数相同的 **server _ hello** 消息的到来。对 server _ hello 消息而言，应用了如下惯例：版本域中包含的是客户端支持的最低版本号和服务器支持的最高版本号。随机数域是由服务器生成的，与客户端的随机数域相互独立。如果客户端会话标识非 0，则服务器使用与之相同的值，否则，服务器的会话标识域包含新对话的值。密码组域包含着服务器从客户端所给的密码组中选出的密码组。压缩域包含的是服务器从客户端所给的压缩方法中选出的压缩方法。

密码组参数的第一个元素是密钥交换方法（如传统加密密钥和 MAC 交换的方法）。支持下述密钥交换方法：

- **RSA**：用接收者的 RSA 公钥加密的密钥，必须拥有接收者公钥的公钥证书。
- **固定 Diffie-Hellman**：Diffie-Hellman 密钥交换，其中包含认证中心签发的 Diffie-Hellman 公钥参数的服务器证书，也就是说，公钥证书包含 Diffie-Hellman 公钥参数。客户端在证书中提供它的 Diffie-Hellman 公钥参数，或需要进行客户端认证时，在密钥交换消息中提供证书。
- **瞬时 Diffie-Hellman**：此技术用于创建瞬时（临时、一次性）的密钥。在这种情况下，Diffie-Hellman 公钥在交换时使用发送者的 RSA 或 DSS 私钥签名。接收者使用相应的公钥验证签名。由于它使用的是临时的认证密钥，因此在三种 Diffie-Hellman 选项中最安全。
- **匿名 Diffie-Hellman**：使用基本的 Diffie-Hellman 算法，没有认证。即在向对方发送其 Diffie-Hellman 公钥参数时，不进行认证。这种方法容易受到中间人攻击，攻击者可以使用匿名 Diffie-Hellman 与双方进行通话。
- **Fortezza**：为 Fortezza 模式定义的技术。

密钥交换方法定义之后的是 CipherSpec，其中包含以下域：

- **密码算法**：任何前面提及的算法：RC4、RC2、DES、3DES、DES40、IDEA、Fortezza。
- **MAC 算法**：MD5 或 SHA-1。
- **密码类型**：流或分组。
- **可出口**：真或假。
- **散列长度**：0，16（MD5）字节或 20（SHA-1）字节。
- **密钥材料**：字节序列，包含生成写密钥所使用的数据。
- **IV 大小**：密码分组链接（CBC）加密使用的初始矢量的大小。

阶段 2　服务器认证和密钥交换

如果需要进行认证，则服务器开始于发送其证书；消息包含一个或一组 X.509 证书。除匿名 Diffie-Hellman 方法外，其他密钥交换方法均需要**证书消息**（**certificate message**），注意，如果使用固定 Diffie-Hellman，此证书消息将由于包含了服务器 Diffie-Hellman 公钥参数而作为服务器的密钥交换消息。

接着，如果需要，可以发送**服务器密钥交换消息**（**server _ key _ exchange message**）。在以下两种情况下，不需要此消息：（1）服务器发送了带有固定 Diffie-Hellman 参数的证书；（2）使用

RSA 密钥交换。以下情况需要 server _ key _ exchange 消息：

- **匿名 Diffie-Hellman**：消息内容包含两个全局 Diffie-Hellman 值（一个素数和它的原根）和服务器 Diffie-Hellman 公钥（如图 10.7 所示）。
- **瞬时 Diffie-Hellman**：消息内容包含三个 Diffie-Hellman 参数，包括匿名 Diffie-Hellman 中的两个参数和它们的参数签名。
- **RSA 密钥交换，服务器在使用 RSA 时仅用了 RSA 签名密钥**：因此，客户端不能简单地通过服务器公钥加密其密钥后传送，而服务器必须创建一个临时 RSA 公钥/私钥对，并使用服务器密钥交换消息发送公钥。消息内容包含两个临时的 RSA 公钥参数（指数和模，见图 9.5）和参数签名。
- **Fortezza**。

签名的其他细节均能得到保证。通常情况下，通过对消息使用散列函数并使用发送者私钥加密获得签名。在此，散列函数定义如下：

```
hash (ClientHello.random ‖ ServerHello.random ‖ ServerParams)
```

散列不仅包含 Diffie-Hellman 或 RSA 参数，还包含初始 hello 消息中的两个 nonce，可以防止重放攻击和伪装。对 DSS 签名而言，散列函数使用 SHA-1 算法；对 RSA 签名而言，将要计算 MD5 和 SHA-1，再将两个散列结果串接（36 字节）后，用服务器私钥加密。

接下来，一个非匿名服务器（服务器不使用匿名 Diffie-Hellman）需要向客户端申请证书。**证书请求消息**（**certificate _ request message**）包含两个参数：证书类型和认证中心。证书类型表明了公钥算法和它的用途：

- RSA，仅用于签名。
- DSS，仅用于签名。
- 固定 Diffie-Hellman 的 RSA；此时，发送 RSA 签名证书，其签名仅用于认证。
- 固定 Diffie-Hellman 的 DSS；仅用于认证。
- 瞬时 Diffie-Hellman 的 RSA。
- 瞬时 Diffie-Hellman 的 DSS。
- Fortezza。

证书请求消息中的第二个参数是一个可接受的认证中心名字表。

第二阶段中的最后一个消息是**服务器完成消息**（**server _ done message**），它通常是需要的。此消息由服务器发送，表明服务器的 hello 和相关消息结束。在此消息发送之后，服务器将等待客户端应答。此消息不带参数。

阶段 3　客户端认证和密钥交换

在接收到服务器完成消息之后，如果请求了证书，客户端需要验证服务器是否提供了合法证书，并且检查 server _ hello 参数是否可接受。如果所有的条件均满足，则客户端向服务器发回一个或多个消息。

如果服务器请求了证书，则在此阶段客户端开始发送一条**证书消息**（**certificate message**）。如果未提供合适的证书，则客户端将发送一个“无证书警报”。

接下来是此阶段必须要发送的**客户端密钥交换消息**(**client _ key _ exchange message**),消息的内容依赖于密钥交换的类型:

- **RSA**:客户端生成 48 字节的次密钥,并使用服务器证书中的公钥或服务器密钥交换消息中的临时 RSA 密钥加密。它用于生成稍后介绍的主密钥计算。
- **瞬时或匿名 Diffie-Hellman**:发送客户端的 Diffie-Hellman 公钥参数。
- **固定 Diffie-Hellman**:由于证书消息中包括 Diffie-Hellman 公钥参数,因此,此消息内容为空。
- **Fortezza**:发送客户端的 Fortezza 参数。

在此阶段的最后,客户端可以发送一个**证书验证消息**(**certificate _ verify message**)来提供对客户端证书的精确认证。此消息只有在客户端证书具有签名能力时发送(如除带有固定 Diffie-Hellman 参数外的所有证书)。此消息对一个基于前述消息的散列编码的签名,其定义如下:

```
CertificateVerify.signature.md5_hash
    MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1));
Certificate.signature.sha_hash
    SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1));
```

其中,pad _ 1 和 pad _ 2 是前面 MAC 定义的值,握手消息指的是所有发送的握手协议消息或接收到的从 client _ hello 消息开始不包括此消息的所有消息。主密钥的计算方法将在以后介绍。如果用户私钥是 DSS,则被用于加密 SHA-1 散列;如果用户私钥是 RSA 的密钥,则被用于加密 MD5 和 SHA-1 散列连接。不管在哪种情况下,其目的都是为了使用私钥验证客户证书的客户所有权。即使有人误用了客户证书,它也无法发送消息。

阶段 4　完成

此阶段完成安全连接的设置。客户端发送**修改密码规范消息**(**change _ cipher _ spec message**)并向当前的 CipherSpec 中拷贝挂起 CipherSpec。注意,此消息不是握手协议的一部分,而是使用修改密码规范协议发送的。于是,客户端立即使用新的算法、密钥和密码发送新的**完成消息**(**finished message**)。完成消息对密钥交换和认证过程的正确性进行验证。完成消息的内容包含两个散列值的连接:

```
MD5(master_secret | pad2 | MD5(handshake_messages | Sender | master_secret | pad1))
SHA(master_secret | pad2 | SHA(handshake_messages | Sender | master_secret | pad1))
```

其中发送者(Sender)表示发送者为客户端,握手消息(handshake _ message)包括除此消息之外的所有握手消息数据。

在应答这两个消息时,服务器发送自己的修改密码规范消息(change _ cipher _ spec),并向当前的 CipherSpec 中拷贝挂起 CipherSpec,发送完成消息。此时,握手完成,客户端和服务器即可开始交换应用层数据。

17.2.6　密码计算

下面介绍通过密钥交换创建共享主密钥和使用主密钥生成密码参数。

主密钥的创建

共享主密钥是利用安全密钥交换为此会话建立的一个一次性 48 字节的值(384 位)。生成此密钥共分为两个阶段:首先,交换次密钥 pre_master_secret;其次,双方共同计算主密钥 master_secret。对于次密钥交换,有两种可能:

- **RSA**:由客户端生成 48 字节的次密钥,用服务器的 RSA 公钥加密后,发往服务器。服务器用其私钥解密密文,得到次密钥。
- **Diffie-Hellman**:客户端和服务器同时生成 Diffie-Hellman 公钥。密钥交换后,各方执行 Diffie-Hellman 计算,创建共享次密钥。

然后,双方按如下方法计算主密钥:

```
master_secret = MD5(pre_master_secret || SHA ('A'|| pre_master_secret ||
                    ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA ('BB'|| pre_master_secret ||
                    ClientHello.random || ServerHello.random)) ||
                MD5(pre_master_secret || SHA ('CCC'|| pre_master_secret ||
                    ClientHello.random || ServerHello.random))
```

其中,ClientHello.random 和 ServerHello.random 是在初始 hello 消息中交换的两个 nonce。

生成密码参数

CipherSpecs 需要的客户端写 MAC 密钥、服务器写 MAC 密钥、客户端写密钥、服务器写密钥、客户端写初始矢量和服务器写初始矢量,均是通过主密钥生成的。主密钥通过散列函数把所有参数映射为足够长的安全字节序列。

从主密钥生成各主要参数的方法与从次密钥中生成主密钥的方法相同:

```
key_block = MD5(master_secret || SHA ('A'|| master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA ('BB'|| master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA ('CCC'|| master_secret ||
                ServerHello.random || ClientHello.random)) || ...
```

直到生成足够的输出。此算法结构的结果是一个伪随机函数,可以将主密钥看成是该函数的伪随机种子值。客户端和服务器随机数则可看成是复杂密码分析方法的敏感值(参见 18 章)。

17.2.7 传输层安全

传输层安全(TLS)是 IETF 标准的初衷,其目的是编写 SSL 的互联网标准。当前 TLS 的草案 RFC 2246 与 SSLv3 非常相似。在本节中我们将把主要精力集中在它们的区别上。

版本号

TLS 记录格式与 SSL 记录格式相同(如图 17.4),头中各域的含义也相同。其区别在于版本值。在 TLS 当前版本中,其主版本号为 3,从版本号为 1。

消息认证代码

SSLv3 和 TLS 的 MAC 模式有两点不同:实际算法和 MAC 计算的范围。TLS 使用 RFC 2104 中定义的 HMAC 算法。回想第 12 章中的定义:

$$\text{HMAC}_K(M) = \text{H}[(K^+ \oplus \text{opad}) \| \text{H}[(K^+ \oplus \text{ipad}) \| M]]$$

其中:

H　= 嵌入的散列函数(对 TLS 而言,为 MD5 或 SHA-1)

M　= 输入给 HMAC 的消息

K^+　= 左边添 0 的密钥,使得其长度等于散列代码的分组长度(对 MD5 和 SHA-1 而言,分组长度为 512 位)

ipad　= 00110110(十六进制的 36)重复 64 次(512 位)

opad　= 01011100(十六进制的 5C)重复 64 次(512 位)

SSLv3 除填充分组与密钥连接外,它们使用相同算法。两者的安全级别相同。

对 TLS 而言,MAC 计算包括以下表达式的各域:

```
HMAC_hash(MAC_write_secret, seq_num||TLSCompressed.type||
    TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment))
```

MAC 计算不仅覆盖了 SSLv3 中 MAC 计算的各域,还增加了一个体现协议版本号的域 TLSCompressed。

伪随机函数

TLS 使用称为 PRF 的伪随机函数将密码扩展成为生成密钥的数据分组,目的是使用相对较小的共享密码值,生成较长的数据分组,防止对散列函数和 MAC 的攻击。伪随机函数基于下述数据扩展函数(如图 17.7):

```
P_hash(secret, seed) = HMAC_hash(secret,A(1)||seed)||
                       HMAC_hash(secret,A(2)||seed)||
                       HMAC_hash(secret,A(3)||seed)||...
```

其中,A()定义为:

$$A(0) = \text{seed}$$
$$A(i) = \text{HMAC\_hash}(\text{secret}, A(i-1))$$

数据扩展函数使用以 MD5 或 SHA-1 为基本散列函数的 HMAC 算法。P_hash 可以迭代任意次,产生所需的数据。例如,如果使用 P_SHA-1 生成 64 个字节的数据,就需要迭代 4 次,产生 80 个字节的数据,略去最后的 16 个字节。如果使用 P_MD5,则也需要迭代 4 次,恰好产生 64 个字节的数据。注意,每次叠代执行两次 HMAC,每次 HMAC 执行两次基本散列函数。

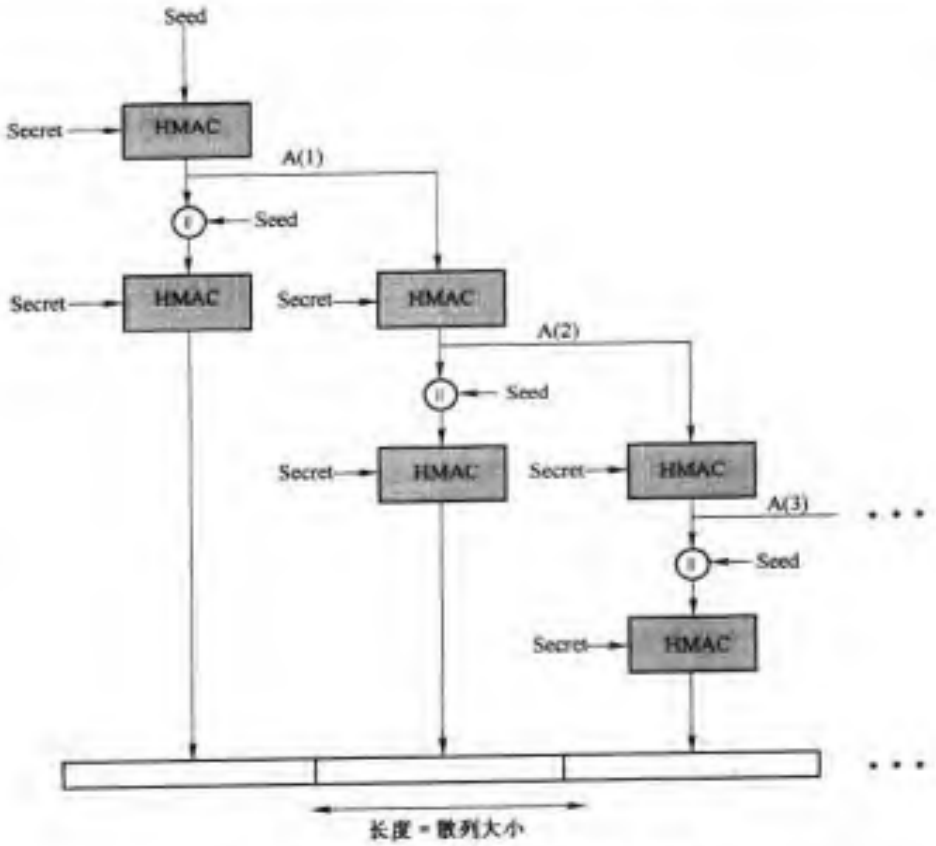


图 17.7 TLS 函数 P_hash (secret, seed)

为了使 PRF 足够安全，PRF 同时使用两种散列函数。只要有一种算法是安全的，则 PRF 就是安全的。PRF 定义如下：

$$PRF(secret, label, seed) = P\_MD5(S1, label \| seed) \oplus P\_SHA\text{-}1(S2, label \| seed)$$

PRF 以密码值 secret、标识标签 label 和种子值 seed 为输入，产生任意长度的输出。通过将 secret 分成两半（S1 和 S2），并对各部分执行 P_hash 函数，一部分使用 MD5，另一部分使用 P_SHA，再将两部分的结果异或得到输出。因此，为了使异或操作产生相同的数据量，P_MD5就应该比 P_SHA-1 迭代的次数多。

警报代码

TLS 支持除无证书以外的 SSLv3 中定义的所有警报代码，并且还定义了许多附加的代码。以下列举出了其中的主要部分：

- **解密失败（decryption_failed）**：使用不正确的方法解密密文，或者长度不是分组长度的整数倍，或填充值不正确。
- **记录溢出（record_overflow）**：接收的 TLS 记录中的载荷（密文）长度超过 $2^{14} + 2048$ 个字节，或者密文解密后长度超过 $2^{14} + 1024$。
- **未知的认证中心（unknown_ca）**：接收到正确的证书链或部分链，但由于证书不能定位或不能与可信任的认证中心匹配而不接收证书。

- **拒绝访问(access _ denied)**:接收到合法证书,但发送者拒绝进行协商访问。
- **解码出错(decode _ error)**:由于域超出了指定范围或消息长度不正确使得消息不被解码。
- **输出限制(export _ restriction)**:密钥长度的输出限制不能达成一致。
- **协议版本(protocol _ version)**:客户端试图协商的协议版本可以识别但不被支持。
- **安全不足(insufficient _ security)**:服务器需要的安全级别客户端无法支持时协商失败的返回值,代替握手失败。
- **中间出错(internal _ error)**:与对方或协议正确性无关的中间环节出错,使得无法继续操作。

新的警报代码如下:

- **解密出错(decrypt _ error)**:握手密码访问失败,包括无法验证签名、解密密钥交换或校验完成的消息。
- **用户取消(user _ canceled)**:握手由于某些与协议错误的无关原因而被取消。
- **不再重新协商(no _ renegotiation)**:由客户端响应 hello 请求或服务器在初始握手后响应客户端的 hello。通常应答消息都会导致重新协商,但此警报表明发送者不能重新协商。此消息通常是警告消息。

密码组

SSLv3 和 TLS 提供的密码组有一些小的差别:

- **密钥交换**:TLS 支持除 Fortezza 外的所有 SSLv3 的密钥交换技术。
- **对称加密算法**:TLS 包括除 Fortezza 外的所有 SSLv3 的对称加密算法。

客户端证书类型

TLS 定义了以下证书请求消息中需要的证书类型:rsa _ sign、dss _ sign、rsa _ fixed _ dh 和 dss _ fixed _ dh,这些都是 SSLv3 中定义了的类型。另外,SSLv3 中还包括 rsa _ ephemeral _ dh、dss _ ephemeral _ dh 和 fortezza _ kea。瞬时 Diffie-Hellman 使用 RSA 或 DSS 对 Diffie-Hellman 参数加密。对 TLS 而言,不包括 Fortezza 模式,仅使用 rsa _ sign 和 dss _ sign 类型对 Diffie-Hellman 参数加密。

证书验证和完成消息

在 TLS 证书验证(certificate _ verify)消息中,MD5 和 SHA-1 的散列值只根据握手消息计算。SSLv3 中散列值的计算还包括主密钥和填充域,但这些额外的域不能增加它的安全性。

与 SSLv3 中的完成消息(finished)相比,TLS 中的完成消息基于共享主密钥、先前的握手消息和标识服务器或客户端的标签的散列数据。某些计算不同。在 TLS 中,我们有:

```
PRF(master _ secret, finished _ label, MD5(handshake _ messages) ‖ SHA-1(handshake _
messages))
```

其中,finished _ label 或为客户端的字符串"client finished",或为服务器端的字符串"server finished"。

密码计算

TLS 中次密钥的计算方法与 SSLv3 中的相同。与 SSLv3 一样,TLS 的主密钥是次密钥和两个随机 hello 值的散列,但计算方法如下:

```
master_secret =
  PRF(pre_master_secret,"master secret", ClientHello.random ‖ServerHello.random)
```

算法执行直到产生 48 个字节的伪随机数。密钥分组元素(MAC 密钥、会话加密密钥和初始矢量)的计算方法如下:

```
key_block =
    PRF(master_secret, "key expansion",
        SecurityParameters.server_random ‖ SecurityParameters.client_random)
```

直到产生足够多的输出。在 SSLv3 中,key_block 是一个关于 master_secret、客户端和服务器随机数的函数,而与 TLS 中实际使用的算法不同。

填充

在 SSL 加密之前,使用填充域使得用户数据为加密所需的分组长度的最小整数倍。而在 TLS 中,填充域长度不超过 255 个字节,使用户数据为加密所需的分组长度的任意整数倍的值。例如,如果明文(或压缩明文)加上 MAC 和填充长度域共为 79 个字节,则填充域可以是 1、9、17,…,249 位。可变的填充可以防止基于交换消息长度分析的攻击。

17.3 安全电子交易

安全电子交易(SET)是一种开放的加密安全规范,用于保护互联网上的信用卡交易。当前的版本 SETv1 是应 MasterCard 和 Visa 公司的要求于 1996 年 2 月提出的安全标准。在开发最初规范时,涉及了许多公司,包括 IBM、Microsoft、Netscape、RSA、Terisa 和 Verisign。在 1996 年初,对此概念进行了许多尝试,到 1998 年,第一个与 SET 兼容的产品问世。

SET 本身并不是一个支付系统,而是一个安全协议和格式集,用户以此安全方式在开放网络如互联网中使用现存的信用卡支付基础设施。从本质上说,SET 提供三种服务:

- 为交易各方提供安全的信道。
- 通过使用 X.509v3 数字证书提供信任。
- 由于信息只在需要的时间和地方提供,因而要确保私密性。

SET 规范相当复杂,在 1997 年 5 月出版的如下三本书中有其详细定义:

- **书 1**: Business Description (80 页)
- **书 2**: Programmer's Guide (629 页)
- **书 3**: Formal Protocol Definition (262 页)

该规范共有 971 页。相反,SSLv3 规范只有 63 页,而 TLS 规范仅有 80 页。本节仅总体介绍该协议。

17.3.1 SET 综述

我们将从 SET 的商业需求、主要特性和 SET 交易各方来讨论 SET。

需求

SET 规范列举了在互联网或其他网络上用信用卡进行安全支付的商业需求:

- **提供付款和订购信息的保密性**：向持卡人确保信息的安全性且信息只能被指定的接收方访问。保密性也减少了被任何一方或怀有恶意的第三方欺诈的危险。SET 使用加密提供保密性。
- **确保传送数据的完整性**：确保在 SET 消息的传送过程中消息内容不被改变。使用数字签名提供完整性。
- **为持卡人是否为信用卡账号的合法用户提供认证**：将持卡人和特定账号相联系的机制减少了欺诈的发生和支付处理的总开销。使用数字签名和证书来验证持卡人是否为合法账号的合法用户。
- **可以根据它与金融机构的关系通过接受信用卡交易为商家提供认证**：这是前述需求的补充。持卡人需要能够识别他们将要进行安全交易的商家。仍然使用数字签名和证书。
- **确保使用最好的安全模式和系统设计技术保护电子交易中所有合法方的利益**：SET 是基于高度安全的密码算法和协议的、经过严格测试的规范。
- **创建一个不依赖于传输安全机制也不妨碍其使用的协议**：SET 可以在原始的 TCP/IP 栈上实现安全访问。且 SET 不干预其他安全机制如 IPSec 和 SSL/TLS 的使用。
- **在软件和网络提供者之间提供功能设施和互操作性**：SET 协议和格式与硬件平台、操作系统及 Web 软件无关。

SET 的主要特性

为了达到上述要求，SET 具有下述特性：

- **信息保密性**：持卡人账号和支付信息在网络传输过程中是安全的。SET 的一个有趣而重要的特性是它提供了一种防止获知持卡人信用卡号的机制，只将其提供给发行银行。传统加密方法 DES 用于提供保密性。
- **数据完整性**：持卡人发往商家的支付信息包括订购信息、个人数据和支付指令。SET 保证这些消息内容在传送过程中不被改动。RSA 数字签名使用 SHA-1 散列编码提供消息完整性。某些消息也使用 SHA-1 的 HMAC 保护。
- **持卡人账号认证**：SET 使得商家能够验证持卡人是否为合法卡账号的合法用户。为此，SET 使用带有 RSA 签名的 X.509v3 数字证书。
- **商家认证**：SET 使得持卡人能验证商家是否与允许接受支付卡的金融机构建立了联系。为此，SET 使用带有 RSA 签名的 X.509v3 数字证书。

注意，与 IPSec 和 SSL/TLS 不同，SET 仅为每种密码算法提供一种选择。这是因为 SET 是针对需求集合提供的单一应用，而 IPSec 和 SSL/TLS 则是为了支持一组应用。

SET 参与各方

图 17.8 例示了 SET 系统的参与各方，包括：

- **持卡人(Cardholder)**：在电子环境中，消费者和公司购买者使用个人电脑与商家通过互联网进行交互。持卡人是由发行机构发行的、经过授权的支付卡(如 MasterCard、Visa)的持有者。
- **商家(Merchant)**：商家是拥有持卡人所需商品或服务的个人或组织。一般地，这些商品和服务是通过 Web 站点或电子邮件提供的。能接受支付卡的商家必须与清算银行有联系。
- **发卡机构(Issuer)**：是一个向持卡人提供支付卡的金融机构，如银行。一般地，通过邮件或个人申请账号。最终，由发卡机构为持卡人的支付账务负责。

- **清算银行(Acquirer)**:为商家建立账号的金融机构,处理支付卡认证和付款。商家通常可以接受多种品牌的信用卡,但并不想与所有发卡机构打交道。清算银行向商家提供认证,提供给定卡号是否合法和信用卡的消费限额等信息。清算银行还将支付信息传送到商家的账户中。随后,发卡行还要为付款网络中的电子资金流动向清算银行提供补偿。
- **支付网关(Payment Gateway)**:由清算银行或指定的第三方提供的功能,处理商家支付信息。支付网关是 SET 和现存的银行卡支付网络的接口,提供认证和支付功能。商家通过互联网使用支付网关交换 SET 消息,而支付网关与清算银行的金融处理系统具有某种直接的连接或网络连接。
- **认证中心(Certification Authority, CA)**:被信任的、为持卡人、商家和支付网关发行 X.509v3 公钥证书的实体。SET 的成功依赖于为此目的服务的 CA 基础设施。如前所述,使用层次 CA,使得各方不需要直接被根 CA 认证。

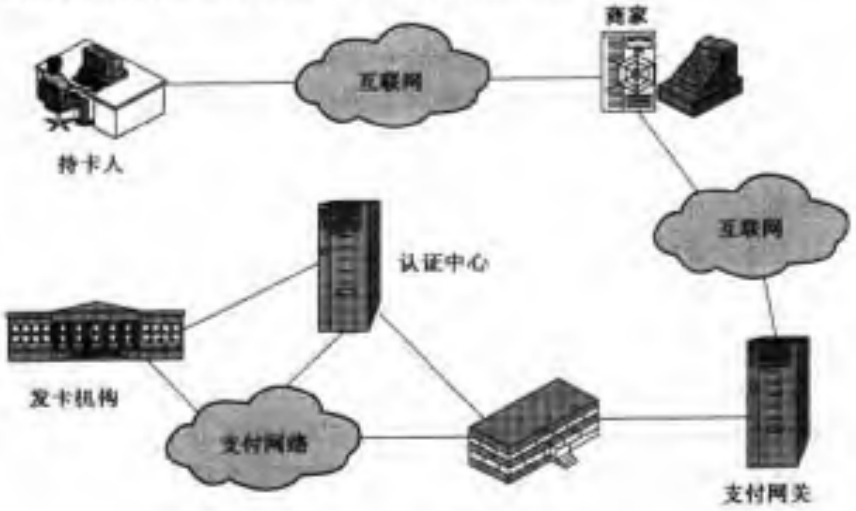


图 17.8 安全电子商务的组件

现在,我们简单描述进行一次交易需要进行的操作,然后再讨论一些密码细节。

1. **顾客开通账号**:顾客从一个支持电子支付和 SET 的银行获得一个信用卡账号,如 MasterCard 或 Visa。
2. **顾客收到证书**:在通过适当的身份验证后,顾客收到一个银行签发的 X.509v3 数字证书。证书验证了顾客的 RSA 公钥和有效期限,并建立了一个由银行保证的用户密钥对和信用卡之间的联系。
3. **商家拥有他们自己的证书**:接收某品牌信用卡的商家必须拥有两种公钥证书:一个用于签名消息,一个用于密钥交换。商家还需要一个支付网关公钥证书的备份。
4. **顾客进行订购**:这是一个涉及到用户首先浏览商家的 Web 站点选择商品和决定价格的过程。然后,用户向商家发送一份购买清单,商家发回一个带有各种商品、单价、总金额和订购号的订购单。
5. **商家被验证**:除了订购单,商家还发给客户一份他自己的证书,使得用户可以验证他正在和一个合法的商店进行交易。

6. **发送订购和付款信息**：顾客发送带有其证书的订购和支付信息给商家。其订购信息确认了订购单中要购买的项目。支付信息包括信用卡细节，并用商家无法解密的方法加密。顾客的证书可以使商家验证顾客。
7. **商家请求付款认证**：商家将付款信息发给支付网关，请求认证顾客提供的信用卡可以支付此次购买。
8. **商家确认订购**：商家向客户发送订购确认消息。
9. **商家提供商品或服务**：商家向顾客提供商品或服务。
10. **商家请求付款**：此请求发给支付网关，由支付网关处理所有的支付操作。

17.3.2　双向签名

在讨论 SET 协议细节之前，我们首先就 SET 中引进的一个重要创新进行讨论：双向签名。双向签名的目的在于将两个接收者的不同消息连接起来。在本例中，客户想给商家发送订购信息(OI)，给银行发送支付信息(PI)。商家不需要知道客户的信用卡号，银行不需要知道客户订购的细节。因此，为客户提供了分离这两者的额外保护。然而，必须将这两个部分连接在一起，以解决一些可能发生的纠纷，因为这个连接可以使客户证明这笔支付是为了这次订购，而不是为其他商品或服务的付款。

考虑此连接的需求，假设客户向商家发送了两个消息：一个签名的 OI1 和一个签名的 PI1，然后由商家将 PI1 传送给银行。如果商家从此客户获得了另一个 OI2，则商家可以说这个 OI2 是与 PI1 配套的，而不是原来的那个 OI1 配套。连接即可防止此类事件的发生。

图 17.9 说明使用双向签名满足了前述的要求。客户使用 SHA-1 从 PI 和 OI 中获得散列值。将两个散列值连接后，再对结果使用散列。最后，用户使用其签名私钥加密最后的散列值，创建双向签名。这些操作可以总结如下：

$$DS = E_{KR_C}[H(H(PI) \| H(OI))]$$

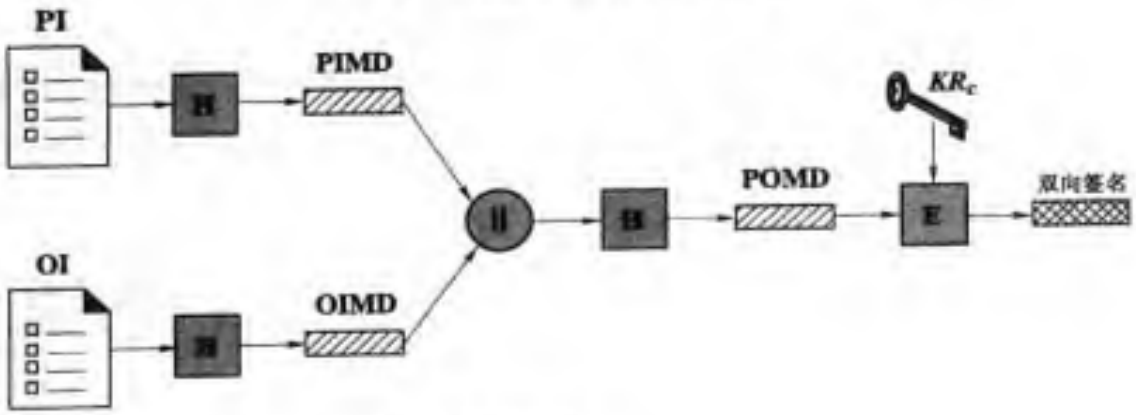


图 17.9　双向签名的构造

其中 KR_c 是用户的签名私钥。现在，假设商家得到了双签名(DS)、OI 和 PI 的数字摘要(PIMD)。由于商家拥有从客户证书得到的客户公钥，使得商家可以计算如下两个值：

$$H(PIMD \| H(OI)) 和 D_{KU_c}[DS]$$

其中 KU_c 是用户的签名公钥。如果这两个值相等，商家即验证了签名。同样，如果银行拥有 DS、PI 和 OI 的数字摘要(OIMD)、用户公钥，则银行可以计算：

$$H(H(PI) \| OIMD) 和 D_{KU_c}[DS]$$

如果两个值相等，银行即验证了签名。总之，

1. 商家接收 OI 并验证签名。
2. 银行接收 PI 并验证签名。
3. 客户链接 OI 和 PI，可以证明此连接。

例如，假设商家为了自身的利益想用另一个 OI 替换交易中的 OI，他就必须找到与现存散列值 OIMD 相同的 OI。对 SHA-1 而言，这是不可行的。因此，商家不能将那个 OI 与此 PI 相连。

17.3.3 支付处理

表 17.3 列举了 SET 支持的交易类型。我们将分别对以下交易类型进行阐述：

- 购买请求
- 支款认可
- 支付获取

表 17.3 SET 交易类型

持卡人注册	持卡人在向商家发送 SET 消息之前必须到 CA 注册
商家注册	商家在与顾客和支付网关交换 SET 消息之前必须到 CA 注册
支付请求	顾客发给商家的消息，包括给商家的 OI 和给银行的 PI
支付认可	商家和支付网关间交换的消息，验证给定信用卡账号能够支持一次购买
支付获取	允许商家向支付网关申请支付
证书询问和状态	如果 CA 无法快速地完成证书请求处理，它将给持卡人或商家发送一个应答，说明将在以后核对。持卡人或商家发送证书询问消息查询证书请求的状态，如果请求被通过，则收到证书
购买询问	允许持卡人在收到购买应答后查询订购处理的状态。注意，此消息不包含如退货等状态，但能表明认证、获取和信用处理等状态
撤销认可	允许商家更正以前的认可请求。如果订购未完成，则商家撤销所有的认可，如果部分订购未完成(如退货)，则商家撤销部分认可
撤销获取	允许商家更正获取请求中的错误，如店员输入了不正确的交易数据
信用	允许商家在退货或商品在运输过程中损坏时向持卡人账号中发布退还。注意，SET 的信用消息通常是由商家而不是持卡人发送的，商家和持卡人之间的通信使得在 SET 外处理退还
撤销信用	允许商家修正前一个退还请求
支付网关证书请求	允许商家询问支付网关，得到它的密钥交换和签名证书
批管理	允许商家根据批命令与付款网关交换信息
出错信息	表明由于格式或内容验证问题，接收者拒绝消息

购买请求

在开始购买请求之前，持卡人必须完成浏览、选择和订购，接着商家才发给顾客一份完整的订购单。以上所有这些操作都不需要使用 SET。

购买请求交换由四条消息组成：初始请求、初始应答、购买请求和购买应答。

为了给商家发送 SET 消息，持卡人必须拥有一份商家和支付网关的证书。因此，持卡人在给商家的**初始请求**消息中申请得到这些证书，该消息包括顾客使用的信用卡品牌、代表此请求/应答对的标识以及与时间相关的 nonce。

商家生成应答消息，并用签名私钥签名。应答消息包括从顾客请求中得到的 nonce、将在下一条消息中返回的新产生的 nonce 和此次购买交易的交易标识。除对此应答签名外，**初始应答**消息还包括商家的签名证书和付款网关的密钥交换证书。

持卡人通过他们信任的 CA 验证商家和网关的证书，并生成 OI 和 PI。商家设置的交易标识放入 OI 和 PI 中，OI 不包含具体的订购数据如商品的数量和单价。OI 还包含在第一个 SET 消息发送前的选购活动中为商家和顾客之间交换信息的订购索引。接着，持卡人准备**购买请求**消息(如图 17.10)。为此，持卡人生成一次性的对称加密密钥 K_s。

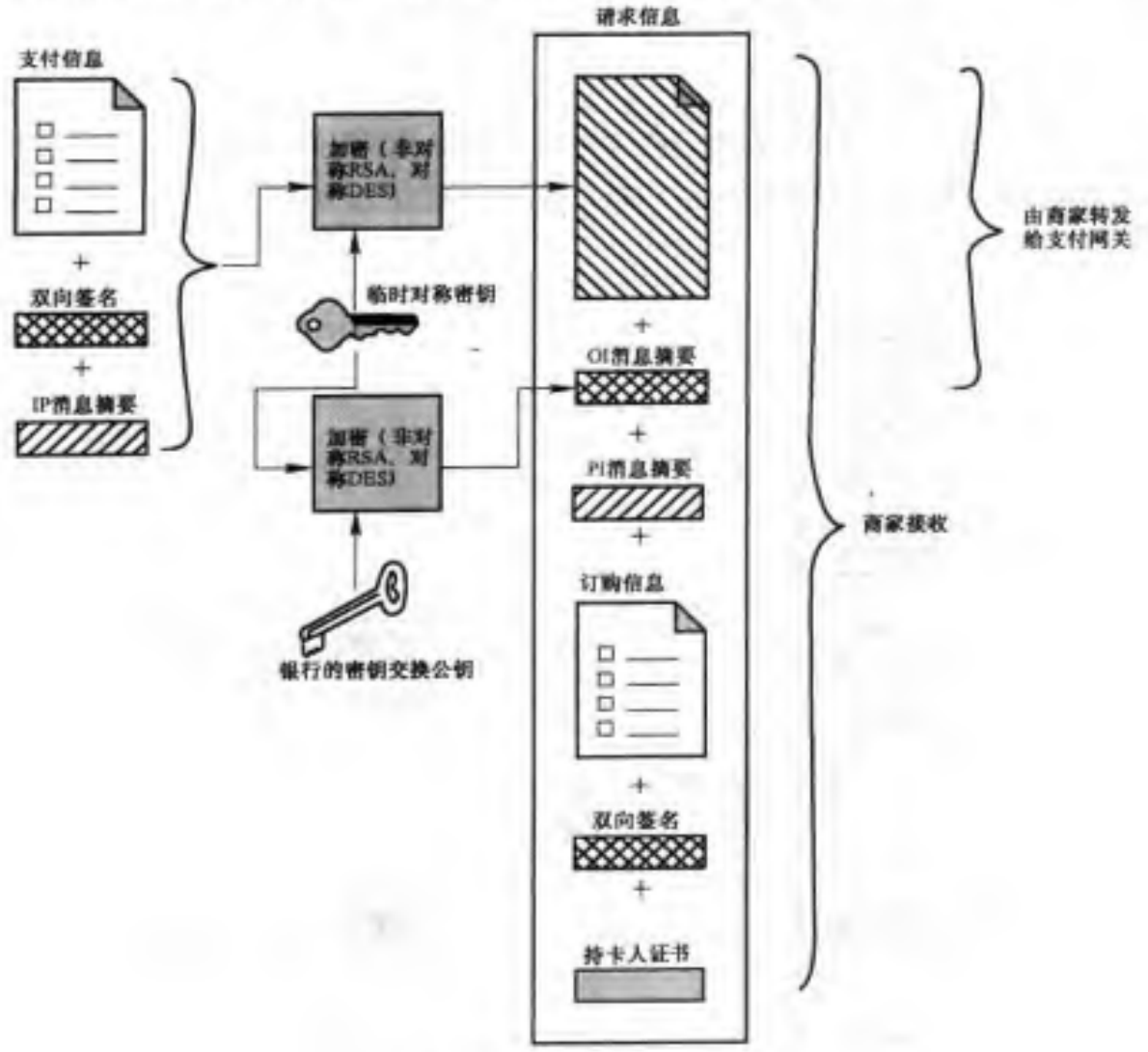


图 17.10　持卡人发送购买请求

消息包括：

1. **与购买相关的信息**。此信息将由商家转发给支付网关，其组成如下：
   - PI。
   - 根据 PI 和 OI 计算得到的双向签名，并使用顾客的签名私钥签名。
   - OI 消息摘要(OIMD)。

   付款网关需要 OIMD 验证双向签名，所有这些项都使用了密钥 K_S 加密。
   - 数字信封：用付款网关的公开交换密钥加密 K_S，由于在得到前述各项之前必须先解密此项，因此将之称为数字信封。

   商家无法知道 K_S 的值，因此，商家无法知道任何与该支付相关的信息。
2. **与订购相关的信息**。此信息是商家需要的，其组成如下：
   - OI。
   - 根据 PI 和 OI 计算得到的双向签名，并使用顾客的签名私钥签名。
   - PI 消息摘要(PIMD)。

   商家需要 PIMD 验证双向签名，注意 OI 是以明文传送的。
3. **持卡人证书**。包含持卡人的公开签名密钥。商家和支付网关都需要它。

当商家接收到购买请求消息后，它执行如下步骤(如图 17.11 所示)：

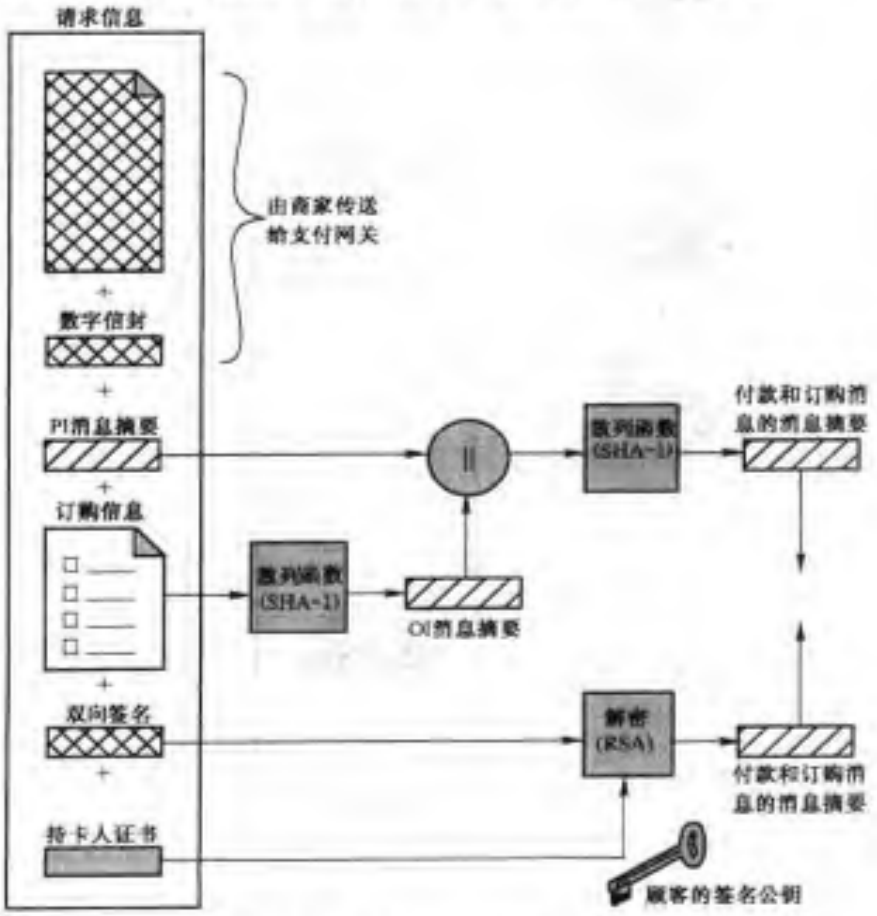


图 17.11 商家验证顾客的购买请求

1. 通过它的 CA 签名验证持卡人证书。
2. 使用顾客的签名公钥验证双向签名，确保在传输过程中订购未被篡改，用持卡人的签名私钥签名。
3. 处理订购并将支付信息转发给支付网关进行验证。
4. 向持卡人发送购买应答。

购买应答消息包括承认订购的应答分组和相应的交易号索引，使用商家的签名私钥签名，并将数据分组、签名和商家签名证书一起发给顾客。

当持卡人软件收到购买应答消息后，使用商家证书验证应答分组上的签名。最后，基于应答进行某些操作，如给用户一个消息或更新订购库的状态。

支付认可

在持卡人订购消息的处理中，商家与支付网关一起认可交易。支付认可使得发卡银行对交易进行担保。认可能确保商家收到付款，而使商家能向顾客提供服务或商品。付款授权交换由两个消息组成：认可请求和认可应答。

商家向支付网关发送的认可请求消息包括：

1. **与购买相关的信息**。从客户获得的信息，包括：
   - PI。
   - 根据 PI 和 OI 计算得到的双向签名，并使用顾客的签名私钥签名。
   - OI 消息摘要(OIMD)。
   - 数字信封。
2. **与认可相关的信息**。由商家生成的信息，包括：
   - **认证分组**：包括使用商家签名私钥签名的交易标识，并使用商家生成的一次性对称密钥加密。
   - **数字信封**：使用支付网关的公开交换密钥加密一次性密钥形成数字信封。
3. **证书**。商家包括持卡人签名密钥证书(用于验证双向签名)、商家签名密钥证书(用于验证商家签名)以及商家密钥交换证书(在支付网关的应答中需要)。

支款网关执行如下任务：

1. 验证所有的证书。
2. 解密认证分组的数字信封，获得对称密钥，解密认证分组。
3. 验证认证分组的商家签名。
4. 解密支付分组的数字信封，获得对称密钥，解密支付分组。
5. 验证支付分组的双向签名。
6. 验证从商家接收到的交易标识，与从客户端接收(间接)的 PI 的交易标识比较。
7. 请求和接收来自于发卡行的认证。

获得发卡行的认可后，支付网关返回**认可应答**消息给商家，包含如下元素：

1. **与认可相关的信息**。包含用网关签名私钥签名的认可分组，并用网关生成的一次性对称密钥加密。同时还包含用商家交换密钥的公钥加密的一次性密钥组成的数字信封。

2. **获取标记信息**。此信息用于以后的支付。此分组与(1)中的格式相同,即签名、加密的获取标记和数字信封。此标记不由商家处理,而必须在支付请求中返回。
3. **证书**。网关的签名密钥证书。

有了网关的认可,商家即可向顾客提供商品或服务。

支付获取

为了获得支付款,商家向支付网关请求支付款获取交易,由获取请求和获取应答两个消息组成。

对**获取请求**消息而言,商家对获取请求分组(包括付款金额、交易标识)签名、加密。消息还包括在认可应答消息中收到的被加密的获取令牌、商家的签名密钥和交换密钥的密钥证书。

当支付网关接收到获取请求消息时,解密和验证获取请求分组和捕获标记。然后验证获取请求与获取令牌的一致性。接着,创建一个通过专用支付网络传送的请求消息,使得资金能够转到商家的账号。

然后,网关在**获取应答**消息中通知商家已支付。消息包括由网关签名和加密的获取应答分组,还包括网关的签名密钥证书。商家软件存储获取应答,便于与从清算银行获得的支付进行验证。

17.4 推荐读物和网址

[RESC01]是有关 SSL 和 TSL 的详细读物。MasterCard 的 SET Web 站点上有关于 SET 协议的详尽描述。此外,[MACG97]是优秀的综述材料。另外,[DREW99]也值得一看。

DREW99 Drew, G. *Using SET for Secure Electronic Commerce*. Upper Saddle River, NJ: Prentice Hall, 1999.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L.; and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*. IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems*. Reading, MA: Addison-Wesley, 2001.

推荐网址:

- **Netscape's SSL Page**:包括 SSL 规范。
- **Transport Layer Security Charter**:最新的 RFC 和 TLS 的互联网草案。
- **MasterCard SET Site**:最新的 SET 文档、术语和应用信息。
- **SETCo**:最新的 SET 文档、术语和其他信息。

17.5　关键术语、思考题和习题

17.5.1　关键术语

清算银行	发卡机构	安全电子交易(SET)
持卡人	商家	安全套接层(SSL)
认证中心(CA)	支付网关	传输层安全(TLS)
双向签名		

17.5.2　思考题

17.1　指出图 17.1 中每种方法的优点。

17.2　SSL 包含哪些协议?

17.3　SSL 连接和 SSL 会话的区别是什么?

17.4　列举并简单定义 SSL 会话状态的参数。

17.5　列举并简单定义 SSL 会话状态的连接。

17.6　SSL 记录协议提供哪些服务?

17.7　SSL 记录协议传输有哪些步骤?

17.8　列举并简单定义 SET 交易各方。

17.9　双向签名的定义和目的是什么?

17.5.3　习题

17.1　在 SSL 和 TLS 中,为什么有单独的修改密码规范协议,而不是在握手协议中包含修改密码规范消息?

17.2　考虑以下 Web 安全性威胁,并描述 SSL 是如何防止这些威胁的。

a. 穷举密码分析攻击:穷举传统加密算法的密钥空间。

b. 已知明文字典攻击:许多消息中包含的是可以预测的明文,如 HTTP 中的 GET 命令。攻击者构造一个包含各种可能的已知明文加密字典。截获加密消息,攻击者可以将包含已知明文的加密部分和字典中的密文进行比较。如果多次匹配成功,可以得到正确的密码。此攻击对于小尺寸的密钥空间非常有效(如 40 位的密钥)。

c. 重放攻击:重放先前的 SSL 握手消息。

d. 中间人攻击:在密钥交换时,攻击者向服务器假扮客户端,向客户端假扮服务器。

e. 密码窃听:HTTP 或其他应用流量的密码被窃听。

f. IP 欺诈:使用伪造的 IP 地址使主机接收伪造的数据。

g. IP 劫持:中断两个主机间活动的、经过认证的连接,攻击代替一方的主机进行通信。

h. SYN 泛滥:攻击者发送 TCP SYN 消息来请求连接,但不回答建立连接的最后一条消息。被攻击的 TCP 模块通常为此预留几分钟的"半开连接",重复的 SYN 消息可以阻塞 TCP 模块。

17.3　利用本章中学到的知识,回答在 SSL 中接收者在接收到顺序紊乱的 SSL 记录分组时能为它们排序吗? 如果可以,解释它是如何做的;如果不可以,为什么?

第四部分　系统安全性

第四部分涉及的内容

第四部分将探讨系统级安全性问题，包括入侵者与病毒所造成的危害与应对这种危害的策略，以及防火墙和可信系统的使用。

第四部分内容浏览

第 18 章　入侵者

第 18 章将探讨由黑客引发的各种信息访问以及设备危害。本章先探讨了未授权用户或入侵者引发的各种类型的攻击，并分析了预防与检测的各种方法。此外，本章还涉及了口令管理的相关问题。

第 19 章　恶意软件

本章探讨软件对系统的危害，重点在于病毒与蠕虫。本章先介绍了各种恶意的软件，随后详细地探讨了病毒和蠕虫的特性。剩下的内容则探讨应对的策略。

第 20 章　防火墙

保护本地计算机不受外部伤害的标准方法是使用防火墙。本章探讨了防火墙的设计原理，并研究了特定的技术。此外，本章还涉及了可信系统的相关问题。

第18章 入 侵 者

网络系统面临的一个严重的安全问题是用户或软件的恶意入侵。用户入侵方式包括未授权登录,或者授权用户非法取得更高级别的权限和操作。软件入侵方式包括病毒、蠕虫和特洛伊木马。

上述的攻击都与网络安全有关,因为攻击是通过网络实现的。然而,攻击不仅限于基于网络的攻击。使用本地终端的用户入侵本地系统可以不通过网络;病毒或特洛伊木马可以通过磁盘传播到系统中;只有蠕虫才是网络特有的现象。因此,系统入侵是网络安全和系统安全的交叉领域。

本书的重点是网络安全,所以我们只是简要介绍有关系统入侵的攻击及其相应的安全对策,不对其进行详细的分析。

本章首先介绍入侵的特征,然后讲述防范策略,最后讨论口令管理问题。

18.1 入侵者

安全性两大威胁之一是入侵者(另一个是病毒),入侵者通常指黑客和解密高手。Anderson在[ANDE80]中把入侵者分为三类:

- **假冒者**:指未经授权使用计算机的人和穿透系统的存取控制冒用合法用户账号的人。
- **非法者**:指未经授权访问数据、程序和资源的合法用户;或者已经获得授权访问,但是错误使用权限的合法用户。
- **秘密用户**:夺取系统超级控制并使用这种控制权逃避审计和访问控制或者抑制审计记录的个人。

假冒者可能是外部使用者,非法者一般是内部人员,秘密用户可能是外部使用者,也可能是内部人员。

入侵者的攻击可能是友善的,也可能是用心险恶的。很多善意攻击者只是想探索一下网络,看看那里是什么;而有些图谋不轨的用户企图读取受权限保护的数据,未经授权修改数据,或者扰乱系统。

入侵者威胁已是众所周知,特别是1986年~1987年发生的著名的"威利黑客"事件[STOL88,STOL89]。1990年,美国对违法的计算机黑客进行了一次全国范围的打击:逮捕、刑事拘留、一次引人注目的公审、几次罪行辩护、没收大量的数据和计算机设备[STER92]。于是许多人相信问题已经在控制之中了。

事实上,入侵行为并没有得到控制。贝尔实验室[BELL92,BELL93]的一个小组报告称,在很长时间内,贝尔实验室的计算机受到来源不同的持续和频繁的攻击。报告指出,贝尔实验室的计算机受到了如下攻击:

- 每隔一天都有一次以上的复制口令文件的企图。

- 每一周有不只一次的可疑的远程过程调用(RPC)请求。
- 至少每两周有一次要求连接到不存在的计算机的企图。

善意入侵者尽管占用资源,降低了合法使用者的系统的性能,但是还是可以容忍的。问题在于,无法事先预知入侵者是善意的还是恶意的。所以,即使系统没有特别敏感的数据,仍然要控制入侵问题。

得克萨斯 A&M 大学曾经出现了备受关注的系统入侵案例[SAFF93]。1992 年 8 月,该校计算机中心发现该校的一台计算机被用于通过网络攻击其他地方的计算机。通过监视,计算机中心管理人员发现,有多个外部的入侵者使用解密程序攻击校内计算机。计算中心断开受影响的计算机,堵上已知的安全漏洞,然后恢复正常操作。几天以后,一位局域系统管理员发现又有入侵者攻击。此次攻击比预想的复杂得多。管理员发现一个文件包含多个已被截获的口令,甚至有曾经被认为安全的服务器的口令。除此之外,有一台本地机器还被当成一个黑客公告牌,黑客们经常用这个公告牌来彼此联络,讨论技术和进展。

对这个攻击的分析显示实际上有两种黑客。高级黑客对技术的了解非常透彻;而低级黑客则是一些只会使用破坏应用程序的"步兵",他们几乎不知道该程序如何运作。而这次联合攻击使用了两个最厉害的武器:洞悉如何入侵和愿意花无数小时来找系统弱点。

计算机紧急响应小组(CERT)的建立正是基于意识到入侵者问题的日益严重性。该小组收集与系统脆弱性有关的信息,将这些信息通告给系统管理员。遗憾的是,黑客也能获得 CERT 的通告。在得克萨斯 A&M 大学事件中,黑客就是根据 CERT 提供的漏洞报告进行攻击的。如果一台计算机没有及时堵上 CERT 通告的漏洞,就可能受到攻击。

除了运行破解口令程序,入侵者试图修改登录软件,以获得此后用户登录系统的口令。

本节介绍入侵技术,然后讲述入侵检测方法。最后,引入基于口令的入侵防范技术。

18.1.1 入侵技术

入侵者的动机是获取系统访问权或是扩大在系统中的权限范围。通常,入侵者需要获得系统保护的信息。在大多情况下,这些信息以用户口令的形式存在。如果知道一些其他用户的口令,入侵者可以登录系统,使用合法用户的所有权限。

系统通常保存一份授权用户和授权用户口令的文件。如果这个文件没有采取保护措施,攻击者很容易访问文件,获取口令。口令文件可以采用以下的保护方法:

- **单向加密**:系统只保存用户口令的密文。当用户提出验证口令请求时,系统加密此口令,并与保存的密文对比,判断真假。实际上,系统通常只进行单向变换(不可逆),在这个单向变换过程中,用口令产生一个密钥来作用于加密,该函数的输出是一个固定长度的值。
- **访问控制**:对口令文件的访问仅限于少数几个账号。

如果采取以上两种措施的一种或全部,入侵者要获取口令的难度就会相当大。根据对一些口令攻击者的采访调查,[ALVA90]给出了如下的、获取口令的技术:

1. 使用系统提供的标准账户和默认口令。许多管理员不愿意改变这些默认值。
2. 穷尽所有的短口令(1 到 3 个字符)。
3. 尝试系统在线词典中的单词或看似口令的单词列表。后者可以很容易地在黑客的公告牌上获得。

4. 收集用户的信息，诸如用户的全称、他们的配偶或孩子的名称、他们办公室中的图片和与他们兴趣有关的书。
5. 尝试用户的电话号码、社会保障号码和房间号码。
6. 尝试本国所有合法牌照号码。
7. 使用特洛伊木马逃避访问限制。
8. 窃听远程用户和主机系统之间的线路。

前六种方法是猜测口令的不同方法。如果入侵者是通过登录系统验证猜测口令的结果，那么这个过程对攻击者是乏味的，也比较容易防范。例如，三次口令验证失败，系统可断开本次登录连接，这使入侵者必须再次连接主机。这种情况下，入侵者不可能测试大量的口令。当然，入侵者不可能使用这种拙劣的方法。比如，如果入侵者在较低的权限下就可访问口令文件，则攻击策略变成了获得文件，然后从容地研究这个特定系统的加密机制，直至获得一个可以提供更高权限的有效口令。

在不考虑检测猜测进程的情况下，如果猜测进程可以自动尝试并检验大量口令时，则猜测攻击是可行的和高效的。本章后面部分将详细讨论如何挫败猜测攻击。

第七种方法——特洛伊木马是比较难以防范的。[ALVA90]提到了逃避访问控制的程序的例子。低权限用户制作一个游戏程序，邀请系统操作员在其空闲时间用。这个程序的确是一个游戏，但是游戏程序中部分代码的作用是将没有加密但受访问控制的口令文件拷贝到用户文件。因为游戏运行于操作员的高权限模式下，所以程序能够访问口令文件。

第八种攻击是线路窃听。这是物理安全问题，可以使用 7.1 节的线路加密技术加以防范。

现在我们讨论两个主要的防范措施：检测和阻止。检测注重掌握攻击成功前后所采用的思路。阻止是具有挑战性的安全问题，在任何时候都是不断升级的战争。阻止在于挫败所有可能的攻击，而攻击者总是查找防御链最薄弱的点进行攻击。

18.2 入侵检测

显然，不存在最好的入侵阻止系统。系统防御的第二道防线是入侵检测，这也是近年来研究的热点。入侵检测成为热点的原因包括：

1. 如果检测到入侵的速度足够快，则在迫害发生或危及数据之前，就可识别出入侵者，并将他们驱逐出系统。即使没有非常及时地检测到入侵者，那么入侵检测越快，破坏的程度会越低，恢复得也越快。
2. 有效的入侵检测系统可以看成是阻止入侵的屏障。
3. 入侵检测可收集入侵技术的信息，用以增强入侵阻止工具。

入侵检测建立的前提是假设入侵者的行为在某些情况下不同于合法用户的行为。当然，入侵的攻击和合法用户正常使用资源的差别不可能十分明显；甚至，他们的行为还有相似之处。

图 18.1 非常抽象地表明了入侵检测系统的设计者所面临的任务性质。虽然入侵者的典型行为有别于授权用户的典型行为，但两者仍有重叠部分。放宽定义入侵者行为，会发现更多的入侵者，也会产生大量的“错误肯定”，即将合法用户判定为入侵者。另一方面，为了减少错误肯定，严格定义入侵者的行为，将导致错误否定的增加，即无法判断出入侵者。

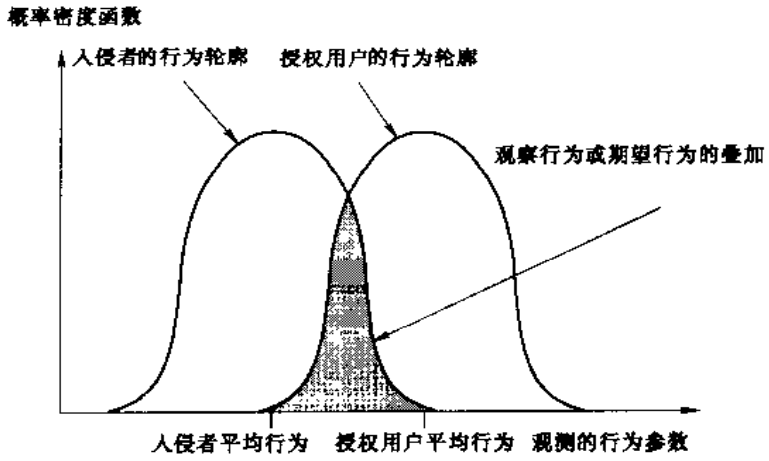


图 18.1　入侵者和授权用户的行为轮廓

在Anderson的研究[ANDE80]中,假定一个人可以非常自信地区分一个假冒者和一个合法用户。通过观察合法用户的合理历史行为来建立其行为模式,明显偏离此模式的行为会被检测到。Anderson指出检测到非法者(行使未授权的行为的合法用户)要困难得多,因为正常行为与异常行为的区别很小。Anderson总结认为单凭查找异常行为不能检测出这种侵犯。但通过巧妙定义表示非授权使用的条件集,仍可以检测出非法者。最后,秘密使用者的检测好像远超出纯自动技术的范围。1980 年所做的这些研究至今仍适用。

[PORR92]定义了如下的入侵检测方法:

1. **统计异常检测**:收集一段时间内合法用户的行为,然后用统计测试来观测其行为,判定该行为是否是合法用户行为。
   a. 阈值检测:这种方法根据各种事件发生的频率确定阈值,阈值的确定与用户无关。
   b. 基于轮廓的检测:为每个用户建立活动轮廓,用于检测每个账号行为的变化。
2. **基于规则的检测**:定义一个规则集,用于判定给定行为是否为入侵者行为。
   a. 异常检测:定义规则,用于检测现在的行为与以前使用的模式的偏差。
   b. 渗透鉴别:用专家系统来查找可疑的行为。

简而言之,统计方法试图定义正常的、期望的行为,而基于规则的方法定义正确的行为。

对于上文所列举的攻击者类型,统计方法能有效对付假冒者,因为假冒者不可能模仿他们盗用账号的行为模式,但此方法无法对付合法者。对于合法者的攻击,基于规则的方法可以识别攻击事件和顺序,从而发现入侵。事实上,很多系统将这两种方法结合起来有效对付更大范围的攻击。

18.2.1　审计记录

入侵检测的一个基本工具是审计记录。用户活动的记录应作为入侵检测系统的输入。一般采用下面两种方法:

- **原始审计记录**:几乎所有的多用户操作系统都有收集用户活动信息的审计软件。使用这些信息的好处是不需要再额外使用收集软件。其缺点是审计记录可能没有包含所需的信息,或者信息没有以方便的形式保存。
- **检测专用的审计记录**:使用的收集工具可以只记录入侵检测系统所需要的审计记录。此方法的优点在于提供商的软件可适用于不同的系统。缺点是一台机器要运行两个审计包管理软件,需要额外的开销。

Dorothy Denning 在[DENN87]中给出了使用检测专用审计记录的例子。每个审计记录包含如下几个域:

- **主体**:行为的发起者。主体通常是终端用户,也可是充当用户或用户组的进程。所有的活动来自主语发出的命令。主语分为不同的访问类别,类别之间可以重叠。
- **动作**:主语对一个对象的操作或联合一个对象完成的操作,如登录、读、I/O 操作、执行。
- **客体**:行为的接收者。客体包括文件、程序、消息、记录、终端、打印机、用户或程序创建的结构。当一个客体是一个活动的接收者时,则主体也可看成是客体,如电子邮件。客体可根据类型分类。客体的粒度可根据客体类型和环境发生变化。例如,数据库行为的审计可以以数据库整体或以记录为粒度进行审计。
- **异常条件**:若返回时有异常,则标识出该异常情况。
- **资源使用**:指大量元素的列表。每个元素都给出某些资源使用的数量(例如,打印或显示的行数,读写记录的次数,处理器时钟,使用的 I/O 单元,会话占用的时间)。
- **时间戳**:当动作发生时用来标识的惟一的时间日期戳。

大多数用户的操作由基本动作构成。比如,用户执行复制文件命令,包括确定访问、建立副本、读源文件、写目标文件。考虑下面的命令:

```
COPY GAME.EXE TO <Library>GAME.EXE
```

该命令由 Smith 发出,将一个可执行文件 GAME 从当前目录复制到 < Library > 目录下。审计记录如下:

Smith	execute	< Library > COPY.EXE	0	CPU = 00002	11058721678
Smith	read	< Smith > GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	< Library > COPY.EXE	write – viol	RECORDS = 0	11058721680

因为 Smith 没有 < Library > 的写权限,所以复制过程终止。

将用户的操作系统分解为基本动作有三个好处:

1. 因为客体是系统保护的实体,使用基本动作可以对影响一个客体的所有行为进行审计。这样,系统可以检测出试图破坏访问控制的行为(根据返回的意外情况的数目来表明一个异常)。在一个主体可访问的客体集中,标识出一个异常即可检测出已发生的破坏行为。
2. 单个客体、单个动作的审计记录简化了模型,也易于实现。

3. 由于专用审计记录结构简单，格式统一，它可以直接将现有的原审计记录转换为检测专用的审计记录，从而相对容易地获取信息或至少部分信息。

18.2.2 统计异常检测

正如我们所说的，统计异常检测分为两类：阈值检测和基于轮廓的检测。阈值检测与一段时间间隔内特殊事件发生的次数有关。如果发生次数超过期望的合理次数，则认为可能存在入侵。

阈值分析本身是粗糙而效率不高的检测器，即使面对并不老练的攻击。阈值和时间间隔必须事先选定。因为用户是不断变化的，阈值检测很可能导致大量错误肯定或错误否定。但是，只有和其他高级技术共同使用，阈值检测才会有效。

基于轮廓的异常检测归纳出每个用户过去的行为特征或用户组过去行为的特征，用于发现有重大偏差的行为。轮廓可能包含多个参数集，所以只发生一个参数的偏差不足以产生一个警告。

统计异常检测是基于对审计记录的分析。审计记录以两种方法作为入侵检测函数的输入。第一种，设计者需制定度量用户行为的量化机制。一段时间内对审计记录的分析可作为一般用户的活动轮廓。这样，审计记录可用于定义典型的行为。第二种，当前审计记录作为入侵检测的输入，即入侵检测模型分析当前审计记录，判定当前行为与典型行为的偏差。

可用于基于轮廓入侵检测的度量机制如下：

- **计数器**：是一个非负整数。如果没有管理员的干预，它只能增加，不能减少。通常，某些事件类型的发生次数在一段时间内不发生变化。例如，一小时内一个用户登录的次数，一个用户会话期间执行命令的次数，一分钟内口令验证失败的次数。
- **标准值**：是一个可增加也可减少的非负整数。通常，标准值用于衡量一些实体的当前值。例如，分配给用户应用的逻辑连接次数和用户进程发出消息的数目。
- **间隔计时器**：指两个相关事件相继发生的时间。例如，连续两次成功登录到一个账号的时间间隔。
- **资源利用**：一定时间内资源消耗的数量。例如，一次用户会话中打印的页数，程序执行花费的时间。

利用以上度量机制，存在多种方法判定当前活动是否可接受。[DENN87] 列出了可能采取的一些方法：

- 均值和标准偏差
- 多变量
- 马尔可夫处理
- 时间序列
- 操作

最简单的统计测试是测量一段历史时期一个参数的均值和标准偏差。这反映一般行为和行为的变化。均值和标准偏差可广泛用于计数器、计时器、资源利用。使用这个方法只能用于粗糙的入侵检测判断。

多变量模型建立在两个或多个变量的关联之上。考虑这种关联可以更确定地定义入侵者

行为(例如,处理器时间和资源使用,登录频率和会话结束时间)。

马尔可夫处理模型用于确定各种状态的转化概率。例如,检查两个命令相继发生的概率。

时间序列模型以时间间隔为基础,查找发生过快或过慢的时间序列。存在多种统计测试用于提取异常行为的时间特征。

最后一种方法是操作模型,它用于判断什么被认为是异常,而非自动分析以前的审计记录。一般地,定义一个固定的范围,超出此范围的则被怀疑为入侵者。此方法适用于入侵者行为可由某类活动推导出来的情况。例如,短时间内有大量的登录企图,则表明存在入侵。

表 18.1 是斯坦福研究院(SRI)入侵检测系统(IDES)[DENN87, JAVI91, LUNT88]采用的度量机制或模型。

表 18.1 可用于入侵检测的度量

度量	模型	检测到的入侵类型
登录和会话活动		
每天每小时的登录频率	均值和标准偏差	入侵者可能在非登录高峰期登录成功
不同位置的登录频率	均值和标准偏差	入侵者可能从特殊用户很少或从不使用的登录地点登录成功
从上次登录到现在的时间	操作模型	用一个"死"账号入侵
每次对话占用的时间	均值和标准偏差	偏差过大表明存在假冒者
站点的输出量	均值和标准偏差	对远程地址发送过多的数据可能意味着敏感数据泄漏
会话资源的使用	均值和标准偏差	非正常的 CPU 使用或 I/O 操作表明存在入侵
登录时口令失败	操作模型	试图用猜测口令的方法入侵
从特定终端登录失败	操作模型	试图入侵
命令和程序执行活动		
执行频率	均值和标准差	可能检测到使用不同命令的入侵者,或者已经获得使用特权命令的合法用户的入侵
程序资源利用	均值和标准偏差	异常值表明存在病毒或特洛伊木马,增加了 I/O 或处理器的使用
拒绝执行	操作模型	可能检测出试图获得高权限的用户的入侵
文件访问活动		
读、写、创建、删除频率	均值和标准偏差	一个用户读写异常表明存在假冒或者正在浏览文件
记录的读和写	均值和标准偏差	异常行为表示试图通过推论和聚合获取敏感数据
读、写、创建、删除文件	操作模型	可以检测到一直试图访问未授权文件的用户
失败的次数	操作模型	

利用统计轮廓的最大优点是不用预先知道安全漏洞的知识。检测程序首先学习什么是正常行为,然后再去查找行为的偏差。此方法与系统特征和脆弱性无关。因此,它可广泛用于多种系统。

18.2.3 基于规则的入侵检测

基于规则的入侵检测系统是通过观察系统中的事件，运用规则集判断一个给定的活动模式是否可疑。在一般情况下，我们将所有的方法分为异常检测和渗透鉴别，尽管这两类方法有重叠部分。

基于规则的异常检测在方法与强度上类似统计异常检测。使用基于规则的方法分析历史审计记录，确定使用模式，并自动产生描述此模式的规则。规则是表示用户、程序、特权、时间槽、终端等过去行为的模式。通过观察当前行为，将每个行为与匹配规则集进行匹配，判定它是否符合某个历史行为模式。

与统计异常检测一样，基于规则的异常检测不用知道系统内部的安全脆弱性知识。而且，此方法建立在对过去行为的观察之上，假设将来的行为类似于过去的行为。这个方法若要有效，需要包含大量规则的数据库。例如，[VACC89]的数据库包含 10^4 到 10^6 条规则。

基于规则的渗透鉴别是与入侵检测不同的方法，它建立在专家系统技术之上。这种系统的关键特征在于要使用规则去鉴别已知的渗透或利用已知弱点的渗透。规则可用来识别可疑行为，即使这个行为符合已建立的使用模式。通常，这些系统中的规则与特定的机器和操作系统有关。而且，这些规则不是由分析审计记录产生的，而是由专家定义的。通常，规则建立需询问系统管理员和安全分析师，用于收集一组已知渗透方案和危及目标系统安全的关键事件。[①] 因此，此方法的好坏取决于建立这些规则的技术。

应用于 NIDX 中的一个使用启发式规则的早期系统就是使用这类规则的一个例子，这些启发式规则给每个活动分配一个可疑度[BAUE88]。启发式的例子如下所列：

1. 用户不能读其他用户的个人目录下的文件。
2. 用户不能改写其他用户的文件。
3. 几个小时内重复登录的用户通常访问他们前一次访问过的文件。
4. 用户一般不能直接使用磁盘设备，只能利用高层的操作系统提供的工具使用磁盘设备。
5. 同一系统，用户只能登录一次。
6. 用户不能复制系统程序。

IDES 使用的渗透鉴别机制运行的过程如下：审计记录一旦生成就与规则集进行匹配。如果找到匹配项，则可疑率增加。如果有足够的匹配，可疑率增加超过阈值，则报告异常。

IDES 方法建立在对审计记录的检查之上。该方法的不足之处是缺乏灵活性。对于某个渗透片断，可能产生大量有稍微差别的审计记录。但是很难将这些差别明确表示在规则中。另一个方法是建立独立于专用审计记录的高层模型。众所周知的状态转换模型——USTAT [ILGU93]，就是这样一个例子。USTAT 处理一般活动而非 UNIX 审计机制记录的具体消息的动作。USTAT 在一个可对 239 个事件进行审计记录的 SunOS 系统上实现。其中，预处理器使用的 28 个事件映射到 10 个一般动作中，见表 18.2。利用这些动作和每个动作调用的参数，可建立表征可疑活动的状态转换图。

---

① 这种询问甚至可以扩大到已悔改的或未悔改的解密高手，让他们提供经验，并付给他们酬金[FREE93]。

表 18.2 USTAT 动作和 SunOS 事件类型

USTAT 动作	SunOS 事件类型
Read	open _ r, open _ rc, open _ rtc, open _ rwc, open _ rwtc, open _ rt, open _ rw, open _ rwt
Write	truncate, ftruncate, creat, open _ rtc, open _ rwc, open _ rwtc, open _ rt, open _ rw, open _ rwt, open _ w, open _ wt, open _ wc, open _ wct
Create	mkdir, creat, open _ rc, open _ rtc, open _ rwc, open _ rwtc, open _ wc, open _ wtc, mknod
Delete	rmdir, unlink
Execute	exec, execve
Exit	exit
Modify _ Owner	chown, fchown
Modify _ Perm	chmod, fchmod
Rename	rename
Hardlink	link

因为将大量不同审计事件映射到少数几个动作中，所以简化了规则生成过程。而且，状态转换图模型可以根据最近的入侵行为灵活地进行修改。

18.2.4 基于比率的错误

入侵检测系统要想实用化，则需保证在可接受的虚假报警率下，检测到大量的入侵。如果只能检测到少量入侵，则系统给人以安全的假象。反之，如果没有入侵而系统频繁报警（虚假警报），系统管理员可能开始忽略这些报警，或浪费大量时间分析本次虚假报警。

可惜，由于概率自身的特点，故很难符合既要有高检测率又要有低虚假报警率的标准。一般来讲，对比于合法使用系统的次数，如果入侵的实际次数比较少，则在严格区分合法行为和入侵行为的条件下，虚假报警率会比较高。[AXEL00]是对已有的入侵检测系统的研究报告。它暗示，目前的入侵检测系统没有克服基于比率的错误问题。附录 18A 简洁地讲了此问题的数学背景。

18.2.5 分布式入侵检测

至今为止，入侵检测的研究都集中在单机系统的孤立工具上。然而，一个典型的组织需要保护局域网内或内部互联网内所有主机的安全。虽然不可能做到每台机器安装一个单机入侵检测系统，但是入侵检测系统可通过网络合作达到有效保护网内机器的目的。Porras 指出了下述分布入侵检测系统设计中存在的主要问题[PORR92]：

- 分布式入侵检测系统可能要处理不同格式的审计记录。在异构环境中，不同的系统会采用不同的原审计收集系统。如果还装有入侵检测系统，则可能还有与安全性相关的其他格式的审计记录。

- 网络中一个或多个节点负责收集和分析网络中各系统的数据。这样，未加工的数据或汇总性数据将在网络之间传递。因此，有必要保证数据的真实性和保密性。要求真实性是为了防止入侵者通过篡改传输的审计信息掩饰其行为。保证保密性是因为审计信息是有价值的。
- 可使用集中式结构和非集中式结构。在集中式结构中，所有审计数据的收集和分析由一个节点负责。此结构减轻了对输入报告综合处理的任务，产生了一个潜在的瓶颈和单点失败问题。对于非集中式结构，有多个分析中心，但他们之间必须协调活动和交换信息。

加利福尼亚大学建立的[HEBE92, SNAP91]分布式入侵检测系统就是一个很好的例子。图 18.2 是该系统的整体结构，主要有三个部分：

- **主机代理模块**：审计收集模块作为后台进程运行在监测系统上。它的作用是收集有关主机安全事件的数据，并将这些数据传至中心管理员。
- **局域网监视代理模块**：其运作方式与主机代理模块相同，但它还分析局域网的流量，将结果报告给中心管理员。
- **中心管理员模块**：接收局域网监视模块和主机代理模块送来的报告，分析报告，并对其进行综合处理用以判断是否存在入侵。

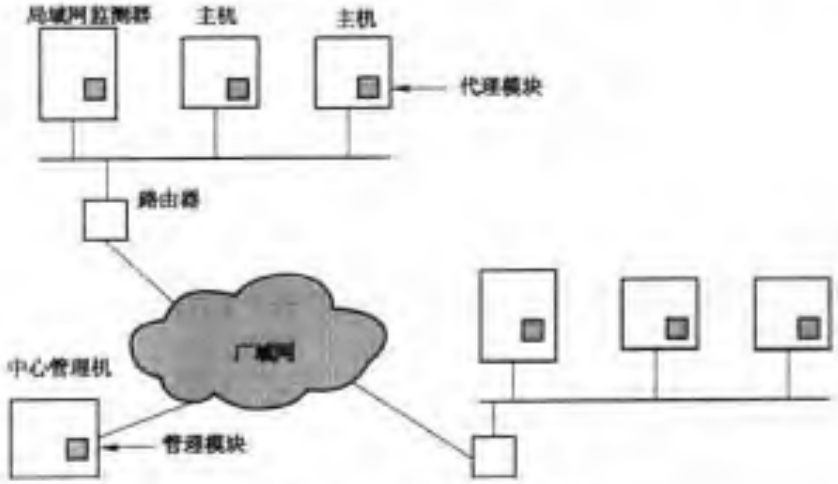


图 18.2 分布式入侵检测的结构

这个机制与操作系统和审计系统实现无关。图 18.3[SNAP91]给出了采用的一般方法。代理截获由原审计收集系统产生的每个审计记录。通过过滤器过滤，只保留与安全性有关的记录。这些记录按主机审计记录(HAR)格式重新组装。然后，使用模板驱动的逻辑模块分析可疑活动的记录。在最底层，代理扫描出与以前事件截然不同的事件。例如，失败的文件访问，访问系统文件，改变文件访问控制。再向上一层，代理查找事件序列，如已知的攻击模式。最后，代理根据用户历史轮廓查找每个用户的异常行为。比如，程序执行次数，文件访问次数等。

当检测到可疑活动，就向中心管理员发出警报。中心管理员使用专家系统，根据传来的信息推导出可能的后果。中心管理员也会主动要求单个主机提供 HAR 副本，与其他代理进行关联。

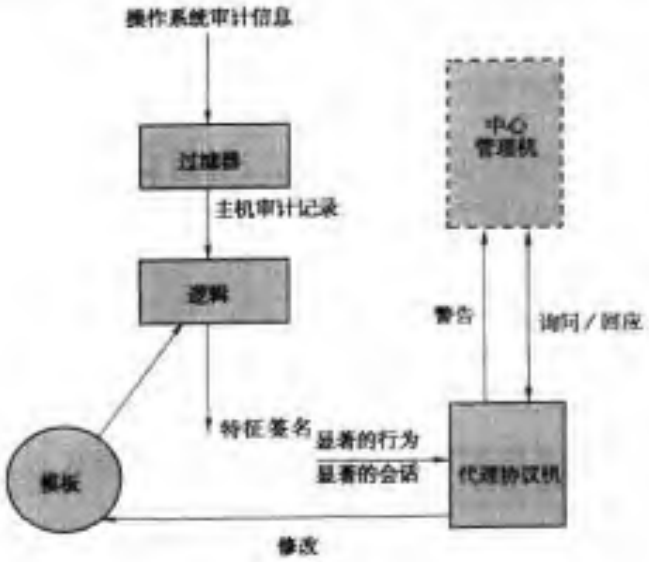


图 18.3 代理结构

局域网监视代理也向中心管理员提供信息。局域网监视代理模块审计主机间的连接、采用的服务和网络流量搜寻重大的事件，如网络负载突然变化、使用与安全性相关的服务以及诸如 rlogin 的网络活动。

图 18.2 和图 18.3 描述的结构非常通用和灵活。它为建立一个与机器无关的检测方法提供了基础，这个方法可以将检测系统从单机扩展到一个可以协作的系统，并对许多站点和网络的活动进行综合处理，以检测到更多的可疑活动。

18.2.6 蜜罐

入侵检测技术中相对较新的创新是蜜罐。蜜罐是诱导潜在的攻击者远离重要系统的一个圈套。蜜罐的功能是：

- 转移攻击重要系统的攻击者
- 收集攻击者活动的信息
- 希望攻击者在系统中逗留足够的时间，使管理员能对此攻击做出响应。

这些系统充满合法用户无法访问、但表面看起来有价值的虚假信息。因此，任何对蜜罐的访问都是可疑的。蜜罐系统使用的工具包括灵敏的监视器和事件日志。事件日志用以检测访问和收集攻击者活动的信息。因为任何对蜜罐的攻击，系统都给出攻击成功的假象，所以系统管理员可以在不暴露真正在工作的系统的条件下，有时间转移、记录、跟踪攻击者。

初始工作是使用一台有 IP 地址的机器作为蜜罐，引诱黑客。最近的研究集中在建立整个蜜罐网络，用来模拟一个企业，其中会使用到实际的或模拟的通信量和数据。一旦黑客进入这个网络，管理员就能详细观察到他们的行为，找出防范措施。

18.2.7 入侵检测交换格式

为了有利于分布式入侵检测系统的发展，使之能运行在各种不同的系统和环境下，需要制

定标准支持协同工作的能力。这样的标准由 IETF 下属的入侵检测工作小组(Intrusion Detection Working Group)负责制定。该工作组的目的是为关系入侵检测和响应系统,以及要与其他机器进行交互的管理系统的共享信息定义数据格式和交换过程。该工作组的成果如下:

1. 需求文档定义:描述入侵检测系统之间通信的高层功能要求;描述入侵检测系统和管理系统间通信的要求,包括这些要求的基本原理。会使用特定的片断举例说明这些要求。
2. 一般入侵语言定义:描述满足需求的数据格式。
3. 框架文件:确定目前用于入侵检测系统之间最好的通信协议;描述如何涉及与入侵检测系统相关的数据格式。

到写本书为止,所有这些文档仍处于互联网草稿文档阶段。

18.3 口令管理

18.3.1 口令保护

入侵者面对的第一条防线是口令系统。基本上所有多用户系统都要求用户提供 ID 和口令。口令用来验证登录到系统的用户 ID。然后,ID 又以如下方式提供安全:

- ID 决定用户是否被授权访问系统。在有些系统中,只有那些在系统中已经申请 ID 的人才能获得访问。
- ID 决定用户权限。少数用户拥有管理权或"超级用户"权限。他们可以读被操作系统保护的文件,执行被操作系统特别保护的功能。一些系统有来客(guest)账号和匿名账号。使用这些账号的用户的权限比其他的用户更受限制。
- ID 用于自主访问控制中。比如,通过列出用户 ID 列表,一个用户就可以授予他们读属于他的文件的权限。

口令的脆弱性

为了了解对密码系统攻击的特征,我们考虑一个广泛用于 UNIX 系统的方案,其中口令从不以明文存储。该方案采用如下的过程[见图 18.4(a)]。每个用户选择长度小于 8 位的可打印字符作为口令。加密过程会将它们转化为 56 位的二进制串(使用 7 位的 ASCII 码),作为输入加密程序的密钥。加密算法 crypt(3)是用一个 12 位的"盐"值修改 DES 算法而得到的。通常,盐值与用户口令产生的时间有关。修改了的 DES 算法的输入是 64 位的全零分组。输出结果再作为第 2 次加密的输入。此过程重复 25 次。最后,将 64 位的输出再转换为 11 个字符序列。口令密文和盐的明文及用户 ID 一起保存在口令文件中。

使用盐有三个目的:

- 防止口令文件中出现相同口令。即使两个用户选用相同口令,也会分配不同的时间值。因此,两个用户"扩展"后的口令不会相同。
- 无需用户额外记住两个字符,就能有效增加口令长度。这样,可能的口令数量增加了 4096 倍,这增加了猜测口令的难度。
- 阻止了用硬件实现 DES,而硬件实现会减少猜测攻击的难度。

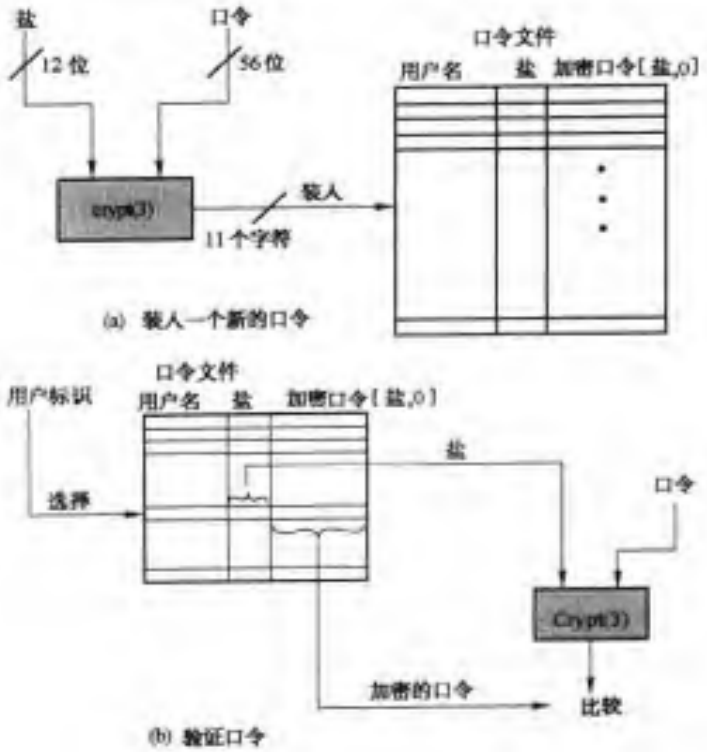


图 18.4 UNIX 口令方案

当一个用户试图登录到 UNIX 系统，他必须提供 ID 和口令。操作系统使用 ID 索引口令文件，取回盐的明文和口令的密文。将盐和用户提供的口令作为加密程序的输入。如果结果匹配存储的值，则通过口令验证。

加密程序是为了阻止猜测攻击。软件实现的 DES 速度慢于硬件实现，其 25 次迭代使计算时间增加了 25 倍。但从算法设计至今，已经出现了两个变化。第一，新的 DES 算法运算速度加快。比如，第 19 章的网络蠕虫使用比 UNIX 系统中的标准加密算法更加高效的加密算法，在较短时间内能猜出几百个在线口令。第二，计算机硬件性能增强，任何一种软件算法的执行速度都加快。

综上所述，UNIX 口令系统存在两个威胁。第一，用户可用来客(guest)账号或其他方式访问一台机器，然后运行猜测口令的程序，这样的用户称为这台机器的口令攻击者。攻击者只需消耗很少的资源就可尝试上百甚至上千个可能的口令。除此之外，如果攻击者能获得口令文件，则可在另一台空闲的机器上运行解密程序。这使攻击者能在一段可接受的时间内尝试数千个可能的口令。

例如，文献[MADS93]中记载了 1993 年 8 月在思维机器公司(Thinking Machines Corporation)的一台并行计算机上运行口令攻击者程序，这台计算机的每个矢量单元每秒可执行1560 次加密。拥有 128 个节点的机器，每个执行节点有 4 个矢量单元(基本配置)，每秒可计算800 000次加密；拥有 1024 个节点的机器每秒可计算 640 万次加密。

对试图尝试所有可能包含的字符的攻击者，即使使用这样惊人的猜测率也不能轻松得手。所以，口令攻击者在进行攻击时，会利用一些用户选择简单易猜的口令这一习惯进行攻击。

有些用户在系统允许其选择自己的口令时，选用很短的口令。普渡大学调查的结果如表 18.3。这个调查观察了 54 台机器上的大约 7000 个用户账号设置口令的选择。3%的口令长度小于或等于 3。对这种口令的攻击可穷尽所有可能的口令。对此，系统简单的补救方法是拒绝接受长度少于某个值的口令，比如 6 个字符的口令，或干脆规定口令必须为 8 个字符。大部分用户都不会抱怨这种限制。

表 18.3 口令长度([SPAF92a])

长度	数目	所占总数的比率
1	55	.004
2	87	.006
3	212	.02
4	449	.03
5	1260	.09
6	3035	.22
7	2917	.21
8	5772	.42
总计	13 787	1.0

口令长度仅仅是其中的一个问题。当允许用户选择自己的口令时，很多人选用容易猜测的口令，如他们的名字，他们住的街道的名字，字典中常见的单词等。这让口令攻击变得简单。因为很多用户使用易被猜测的口令，故这种攻击方法对几乎所有系统都有效。

[KLEI90]中给出了口令猜测有效性的事例。作者收集的 UNIX 口令文件共有近 14 000 个加密的口令。作者公正的研究结果令人震惊，如表 18.4 所示。共有近 3/4 的口令被猜出来。猜测口令使用如下策略：

18.4 口令破解统计(13 797 个账号[KLEI90])

口令类型	查找尺寸	匹配次数	口令匹配百分比(%)	代价/收益比率*
用户/账号	130	368	2.7	2.830
字符序列	866	22	0.2	0.025
数字	427	9	0.1	0.021
中文	392	56	0.4	0.143
地点名称	628	82	0.6	0.131
普通名称	2239	548	4.0	0.245
女性名称	4280	161	1.2	0.038
男性名称	2866	140	1.0	0.049
不常用的名称	4955	130	0.9	0.026
神话和传说	1246	66	0.5	0.053
莎士比亚戏剧中的人物	473	11	0.1	0.023
运动术语	238	32	0.2	0.134
科学小说	691	59	0.4	0.085
电影和演员	99	12	0.1	0.121
动画片	92	9	0.1	0.098
名人	290	55	0.4	0.190
短语和模式	933	253	1.8	0.271
姓氏	33	9	0.1	0.273
生物	58	1	0.0	0.017

（续表）

口令类型	查找尺寸	匹配次数	口令匹配百分比(%)	代价/收益比率
系统字典	19 683	1027	7.4	0.052
机器名称	9018	132	1.0	0.015
记忆术	14	2	0.0	0.143
James Bible 国王	7525	83	0.6	0.011
混合的单词	3212	54	0.4	0.017
犹太人的单词	56	0	0.0	0.000
行星名称	2407	19	0.1	0.007
总计	62 727	3340	24.2	0.053
* 代价/收益比率 = 匹配数目/查找单词数。匹配一个口令需要测试的单词越多，代价/收益比率越低。				

1. 尝试用户姓名、姓、账号名和其他相关的个人信息。对每个用户总共尝试了 130 个不同排列的口令。
2. 尝试各种字典中的单词。作者建立了一个大约 60 000 个单词的词典，包括系统本身的在线字典和所示的其他列表。
3. 测试第 2 种方法中不同排列的单词。这包括大写第一个字母或加一个扩展符，所有字母大写，反写单词，用'0'替代'o'等。这些排列又给列表增加了 100 万个单词。
4. 尝试用大写字母排列第三步中未考虑的但出现在第二步中的单词。这给列表又增加 200 万个单词。

测试使用大约 300 万个单词。使用思维公司最快的计算机，用所有可能的盐值加密所有上述单词所用的时间不到一个小时。这个搜寻的成功率可达 25%，而只要找到一个正确口令，就足以获取很大的系统权限。

访问控制

阻止口令攻击的一个方法是拒绝攻击者访问口令文件。若只有特权用户才能访问口令文件，则入侵者要想访问文件，首先得知道特权用户的口令。[SPAF92a]指出了这种方法的缺陷：

- 包括 UNIX 在内的许多系统很容易受到不可预知的入侵。一旦入侵者通过某种方式获得访问权限，他将希望获得尽量多的口令，这样可以使用不同的账号登录系统，以降低风险。或有账号的用户则希望得到另一个用户的账号来访问特权数据或破坏系统。
- 保护性事故可能导致文件对外可读，这会损害所有用户账号。
- 有些用户在所有机器上的账号使用相同的口令。因此，如果一台机器上的口令文件能被任意访问，则另一台机器上的口令也不安全。

总之，比较有效的方法是强迫用户选择难以猜测的口令。

18.3.2 口令选择策略

上述两个试验（表 18.3 和表 18.4）表明，撇开他们自己的设备不谈，许多用户选用口令过短或易于猜测。在另一个极端情况下，若分配给用户采用 8 个随机的可打印字符作为口令，几乎不能被解密。但是，多数用户难以记住这样的口令。幸运的是，即使我们规定口令是容易记忆的，这个口令范围对攻击者来说仍然很大。于是，我们的目的是选择用户容易记忆而又不容易被猜测的口令。采用以下四种方法：

- 告诉用户应该如何选择口令
- 计算机产生口令
- 口令自检查
- 口令预检查

第一个策略是要让用户知道使用难猜口令的重要性，给他们提供选择强口令的指导。这种**用户教育**方法无法在大多数系统上成功，特别是在存在大量用户或人员流动量很大的情况下。许多用户会无视口令选择指导，还有些人不能判断什么是强口令。比如，许多用户相信，将单词顺序颠倒，或大写最后一个字母，会使此口令成为不易猜测的口令。

计算机生成口令的方法同样存在问题。若口令太随机了，用户不容易记住。就算口令是由一些音节组成，可以发音，用户可能仍然记不住，所以得写下来。计算机生成口令方案一直很难被用户接受。FIPS PUB 181 中定义了一个设计得很好的口令自动生成器。这个标准不仅给出了这个方法的描述，而且给出了该算法的全部 C 语言源代码。该算法可生成可发音的音节，再将它们连接成一个单词。它用一个随机数生成器来产生用来构成音节和单词的随机字符流。

口令自检查方法是系统周期性地运行它自己的口令攻击程序去查找易猜口令。系统取消任何猜中的口令，并告知用户。但这个策略有许多缺点。首先，若此程序正常运作需要很多资源。因为专业的入侵者窃得口令文件后，可以几小时甚至数天的时间以及全部的资源来运行攻击程序，有效的口令自检查程序与之相比就明显无优势了。而且，在口令自检查器发现脆弱口令之前，现存的脆弱口令会一直存在于系统之中，也就存在着被攻击的危险。

增强口令安全性最可行的方法是**口令预检查器**。在这个机制中，可以允许用户选择他自己的口令，但在口令被选择后，系统会检查该口令是否可以接受，若不可以，则拒绝该口令。这种检查器使得用户在系统的充分指导下，可以从一个相当大的口令库中选择易记的口令而不会被字典攻击猜中。

口令预检查器是在用户可接受性和口令强度之间的一个权衡策略。若系统拒绝太多口令，用户会抱怨选择口令太难了。若系统用一个简单的算法定义可接受的口令，则为攻击者改进其猜测技术提供了指导。在后文中，我们将看到一些可行的预口令检查方法。

第一个方法是使用强制规则执行的简单系统。例如，需要执行下面的一些规则：

- 所有口令长度不小于 8
- 口令头 8 个字符中，至少有一个大写，一个小写，一个数字，一个标点符号

这些规则应该作为建议提供给用户。虽然这个方法比第一种策略高级，但它仍不能完全阻止口令攻击。这个机制提醒用户不要选择那些口令，但仍有可能被攻击。

另一个可能的程序是简单地编辑一个“坏”口令太字典。当用户选择口令时，系统检查口令是否在该字典中。但这个方法有两个问题：

- **空间**：这个字典若有效必须非常大。例如，如文献[SPAF92a]中使用的字典占用了超过 30 M 字节的空间。
- **时间**：查询这个大字典的时间很长。除此之外，坏口令该可能有多种排列，为了检查出所有的坏口令，每个查询都必须做相当大的处理。

两种可用来设计基于拒绝列表中单词的有效且高效率的口令预检查器程序的方法都很好。其中之一是建立一个马尔可夫(Markov)模型，用来生成可猜测的口令[DAVI93]。图 18.5

所给出的就是一个简化了的马尔可夫模型。这个模型表示的语言基于由 3 个字符组成的字母表。任何时刻系统的状态表示为当前字母。从一个状态转向另一个状态的转换值表示了一个字母在另一个字母之后的概率。例如，当前字符 a 的下一个字符 b 的概率是 0.5。

通常，马尔可夫模型是一个四元组[m, A, $\mathbf{T}$, k]，其中 m 是模型中的状态数，A 是状态空间，$\mathbf{T}$ 是转换概率矩阵，k 是模型的阶。一个 k 阶的模型，转换到一个特定的字母需要根据前面已产生的 k 个字母。图 18.5 所示的是一个简单的一阶模型。

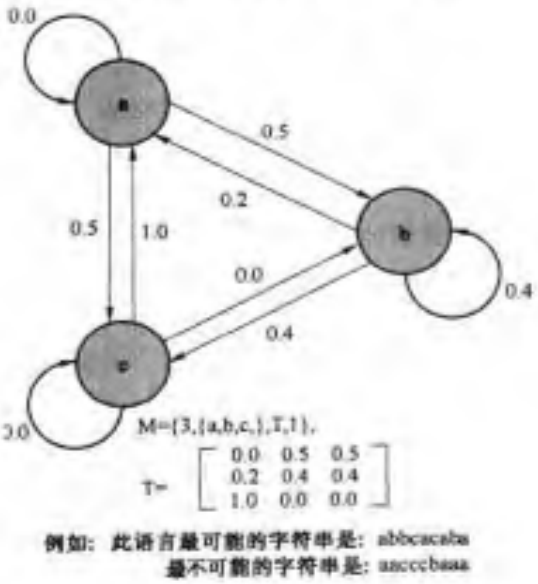


图 18.5 马尔可夫模型的例子

作者建立了一个二阶的模型。首先建立一个易猜口令的字典，然后如下方式计算转换矩阵：

1. 确定频率矩阵 $\mathbf{f}$，$\mathbf{f}(i,j,k)$是单词中的第 i、第 j、第 k 个字符组成的三字符组出现的次数。例如，口令 parsnips 的三字母组有 par，ars，rsn，sni，nip 和 ips。
2. 对每个二字母组 ij，用 $\mathbf{f}(i,j,\infty)$表示以 ij 开头的三字符组出现的次数。例如，$\mathbf{f}$(a,b,∞)是三字母组 aba，abb，abc…的出现次数。
3. $\mathbf{T}$ 的计算如下：

$$\mathbf{T}(i,j,k) = \frac{\mathbf{f}(i,j,k)}{\mathbf{f}(i,j,\infty)}$$

此方法产生的模型表示了字典中单词的结构。在此模型中，问题“这是一个坏口令”变为“马尔可夫模型能否产生这个字符串”。文献[DAVI93]的作者称，使用第 2 顺序模型，几乎完全可检测出他们建立坏口令字典中的所有单词。

Spafford 提出了另一个不同的方法[SPAF92a, SPAF92b]。它基于一个 Bloom 过滤器[BLOO70]。首先，我们解释 Bloom 过滤器的操作。k 阶的 Bloom 过滤器由 k 个独立的 hash 函数 $H_1(x), H_2(x), \cdots, H_k(x)$组成，每个函数将一个口令映射成一个 0 到 N 之间的 hash 值。即：

$$H_i(\mathbf{X}_j) = y \qquad 1 \leq i \leq k; \qquad 1 \leq j \leq D; \qquad 0 \leq y \leq N-1$$

其中：

X_j = 口令字典中的第 j 个单词

D = 口令字典中的单词数

对字典进行以下处理：

1. 定义一个 N 位的 hash 表，每一位初始化为 0。
2. 对每个口令计算出 k 个 hash 值，并将 hash 表中的相应位置为 1。如对某个 (i,j)，若 $H_i(X_j) = 67$，则 hash 表中的第 67 位置 1。

对任何提交给检查器的新口令，计算其 hash 值。若 hash 表的相应位为 1，则拒绝该口令。字典中的所有口令都应被拒绝，但仍会有一些“错误判断”（即不在字典中的口令也可能产生了与 hash 表某位匹配的 hash 值）。对此，我们可以考虑使用两个 hash 函数的方法，并假设 undertaker 和 hulkhogan 在字典中，但 xG% # jj98 不在。进一步假设：

$H_1(\text{undertaker}) = 25$ $H_1(\text{hulkhogan}) = 83$ $H_1(\text{xG\% \# jj98}) = 665$

$H_2(\text{undertaker}) = 998$ $H_2(\text{hulkhogan}) = 665$ $H_2(\text{xG\% \# jj98}) = 998$

我们可以看到若口令 xG% # jj98 提交给系统，虽然它不在字典中却仍将被拒绝。如果有许多这样的错误判断，则用户很难选择口令。因此，我们必须设计的 hash 函数应尽量减少错误判断的发生。出现一个错误判断的概率大约是：

$$P \approx (1 - e^{kD/N})^k = (1 - e^{k/R})^k$$

或

$$R \approx \frac{-k}{\ln(1 - P^{1/k})}$$

其中：

k = hash 函数数目

N = hash 表的位数

D = 字典中的单词数

R = N/D，hash 表大小（位）与字典大小（字）的比率

图 18.6 给出了不同 k 值的函数 P。假设有一个 100 万个单词的字典，我们希望口令的错误拒绝率只有 0.01。若选择 6 个 hash 函数，则所需的比率 $R = 9.6$，这样我们需要 9.6×10^6 位 hash 表，即大约 1.2 M 字节的存储空间。相比之下，存储整个字典需要 8 M 字节，比前者多了几乎 7 倍。而且，计算口令检查所用的 6 个 hash 函数直接且简单，与字典大小无关，而使用整个字典，口令搜索的代价很大。[①]

① 马尔可夫模型和 Bloom 过滤器都使用了概率技术。对于马尔可夫模型，存在字典中的一些口令却未被抓住的小概率，也存在不在字典中的一些口令却被拒绝的小概率。对于 Bloom 过滤器，也存在不在字典中的一些口令却被拒绝的小概率。我们又一次看到采用概率技术可以简化求解（参看第 15 章中的第 1 个脚注）。

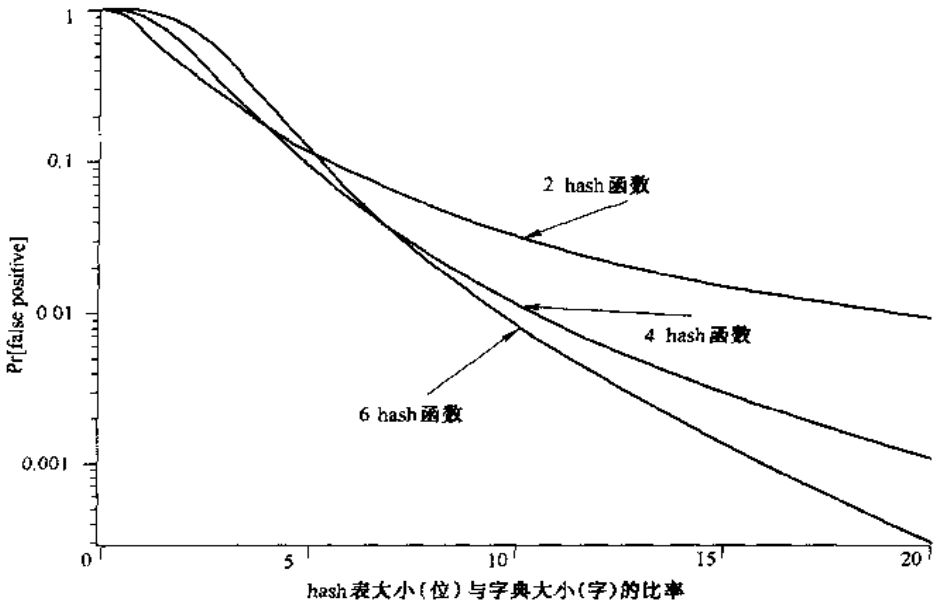


图 18.6 Bloom 过滤器的性能

18.4 推荐读物和网址

[BACE00]和[PROC01]完整地介绍了入侵检测方案。[BACE01]介绍的方案简洁有用。[KENT00]和[MCHU00]是两篇文章,虽然精短,但是非常有用的总结性文章。[HONE01]详细讲述了蜜罐技术,对黑客攻击的方法和工具做了详尽的分析。

BACE00 Bace, R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems*. NIST Special Publication SP 800-31, November 2000.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

MCHU00 McHugh, J.; Christie, A.; and Allen, J. "The Role of Intrusion Detection Systems" *IEEE Software*, September/October 2000.

PROC01 Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

推荐网址:

- **CERT Coordination Center(CERT 协调中心)**:该组织产生于 DARPA(Defense Advanced Research Projects Agency)建立的计算机紧急响应小组。它的网址提供的信息包括:因特网安全受到的威胁、脆弱性和攻击统计数据。

- **Honeypot Project(蜜罐项目)**:分析高明的黑客的技术,制作蜜罐产品。
- **Intrusion Detection Working Group(入侵检测工作小组)**:提供该组制定的所有文档。

18.5 关键术语、思考题和习题

18.5.1 关键术语

审计记录	贝叶斯定理	基于比率的错误
入侵者	入侵检测	入侵检测交换格式
口令	基于规则的入侵检测	盐
		统计异常检测

18.5.2 思考题

18.1 简述三种主要的攻击者。
18.2 保护口令文件有哪两种常用方法?
18.3 入侵检测系统的三个优点是什么?
18.4 统计异常检测和基于规则入侵检测的差别是什么?
18.5 基于轮廓的入侵检测采用的机制是什么?
18.6 基于规则的异常检测和基于规则的渗透鉴别的差别是什么?
18.7 蜜罐的含义是什么?
18.8 UNIX 口令管理中,盐指的是什么?
18.9 简述避免猜测口令的四种技术。

18.5.3 习题

18.1 一辆出租车涉及夜间发生的一起致命的交通事故。该城市有两家出租车公司,分别是蓝和绿。已知:

- 绿公司的出租车数量占全城的 85%,蓝公司的占 15%
- 目击证人证实,肇事车辆属于蓝公司

法庭为了验证目击证人证词的可靠性,模拟事发当时的环境,发现目击证人正确识别出租车颜色的概率为 80%。请问肇事出租车是蓝公司而非绿公司的概率是多少?

18.2 假设口令是 26 个英文字母中任意四个字符的组合,且攻击者猜测口令的速度为每秒一个口令。

a. 假设每次猜测时,在所有字符都测试完成后再反馈给攻击者,那么发现正确口令的时间要多长?
b. 假设当口令中一旦有一个字符错误时就反馈给攻击者,那么发现正确口令的时间要多长?

18.3 长度为 k 的源元素以固定格式转化为长度为 p 的目标元素。每个位有 r 种取值，可知源元素的个数为 r^k，目标元素的个数是 r^p。设存在源元素 x_i 和对应的目标元素 y_j。

a. 如果攻击者只有一个机会猜测源元素，则猜到正确源元素的概率是多少？

b. 攻击者选用不同源元素 $x_k(x_i \neq x_k)$但得到相同目标元素 y_j 的概率是多少？

c. 如果攻击者只有一次机会猜测目标元素，则猜到正确目标元素的概率是多少？

18.4 语音口令产生器对每个 6 个字母的口令随机产生 2 个分段。每个分段的格式为 CVC(辅音，元音，辅音)，其中 V = < a,e,i,o,u > ，C = $\overline{\text{V}}$。

a. 口令总数是多少？

b. 攻击者猜测一个正确口令的概率是多少？

18.5 设口令长度为 10 个字符，且只能用 95 个可显 ASCII 字符。解密者解密的速度为 640 万次加密/秒，那么穷尽 UNIX 系统的所有口令需要多少时间？

18.6 对于 UNIX 口令系统存在的问题，SunOS-4.0 建议将口令文件替换成名为/etc/publickey的公共可读文件。用户 A 在该文件中的保存格式为用户的标识 ID_A，用户的公钥 KU_a 和对应的私钥 KR_a。私钥使用用户登录口令 P_a 和 DES 算法加密。当用户 A 进入系统，系统解密 $E_{P_a}[KR_a]$得到 KR_a。

a. 系统如何确定 P_a 是正确的？

b. 如何攻击这样的系统？

18.7 UNIX 口令所用的加密机制是单向的，即不可逆。是否可以据此说明口令密文是散列值？

18.8 UNIX 口令管理使用了盐，将猜测口令的难度增加了 4096 倍。盐值以明文的形式和对应的口令密文一起保存在文件中。如果攻击者得到口令文件，则得到了口令密文和盐值。为什么说盐的使用增强了安全性？

18.9 假设你能够正确回答问题 18.8，知道盐的含义。为什么不将盐扩大到 24 位或 48 位，用以防范所有的口令解密者？

18.10 考虑 18.3 节的 Bloom 过滤器。定义 k = 散列函数数目，N = 散列表的位数，D = 字典中的单词数目。证明：

a. 散列表中期望位数为 0 可表示为：

$$\phi = \left(1 - \frac{k}{N}\right)^D$$

b. 输入非字典单词而被错误认为是字典中单词的概率为：

$$\text{P} = (1 - \phi)^k$$

c. 上一表达式可近似表示为：

$$\text{P} \approx (1 - e^{-kD/N})^k$$

附录 18A 基于比率的错误

我们将回顾由概率理论推出的重要结论，并举例说明基于比率的错误。

条件概率和独立性

我们通常想知道一些事件在条件限制下的概率。事件发生的条件将去掉部分抽样空间的结果。例如，设有两个骰子，如果我们已知至少一个骰子点数为偶数，则两个是骰子点数之和为 8 的概率是多少？推导如下。因为一个骰子点数为偶数，而总的点数也为偶数，则第二个骰子的点数必为偶数。故可知有三个可能的情况：(2,6)，(4,4)和(6,2)。两个骰子点数都为偶数的可能情况为[36 – 两个骰子点数都为奇数的事件] = 36 – 3 × 3 = 27。最终的概率为 3/27 = 1/9。

事件 B 发生的条件下，事件 A 发生的条件概率公式 $\Pr[A/B]$表示如下：

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]}$$

其中 $\Pr[B]$不为 0。

在我们的例子中，A = {和数为 8}，B = {至少一个骰子的点数为偶数}。$\Pr[AB]$包含所有符合总和数为 8 和至少一个骰子的点数为偶数的条件的所有结果。正如我们所看到的，存在三种结果。因此，$\Pr[AB] = 3/36 = 1/12$。稍加思考，你应该确信 $\Pr[B] = 3/4$。现在，我们计算

$$\Pr[A|B] = \frac{1/12}{3/4} = \frac{1}{9}$$

这符合我们前面的推理。

如果 $\Pr[AB] = \Pr[A]\Pr[B]$，则事件 A 和事件 B 是独立的。很容易看出，如果 A 和 B 是独立的，则 $\Pr[A|B] = \Pr[A]$和 $\Pr[B|A] = \Pr[B]$。

贝叶斯定理

概率论中一个最重要的结论是贝叶斯定理。首先我们有必要陈述全概率公式。给定不相容事件 $E_1, E_2, \cdots, E_n$，这些事件的并集覆盖所有可能的结果，对于任意一个事件 A，可以表示为：

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|E_i]\Pr[E_i] \tag{18.1}$$

贝叶斯定理表示如下：

$$\Pr[E_i|A] = \frac{\Pr[A|E_i]\mathrm{P}[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i]\mathrm{P}[E_i]}{\sum_{j=1}^{n} \Pr[A|E_j]\Pr[E_j]} \tag{18.2}$$

图 18.7(a)表示了全概率公式和贝叶斯定理。

贝叶斯定理计算“较晚的概率”，即在具有有利于某个事件的条件下，事件一定发生的概率。例如，我们要在由噪声的传输线路上传送 0,1 序列。在某一时刻发送 0 的事件表示为 S0，在某一时刻发送 1 的事件表示为 S1，收到 0 的事件表示为 R0，收到 1 的事件表示为 R1。假设我们知道发送源的概率，即 $\Pr[S1] = p$ 和 $\Pr[S0] = 1 - p$。通过观察线路决定在发送 0,1 的时

候，错误发生的频率有多高，概率的计算如下：$\Pr[R0|S1] = p_a$ 和 $\Pr[R1|S0] = p_b$。如果收到 0，我们然后计算发生错误的条件概率，即发送信号是 1 而接收到的信号是 0 的条件概率，使用贝叶斯定理得到：

$$\Pr[S1|R0] = \frac{\Pr[R0|S1]\Pr[S1]}{\Pr[R0|S1]Pr[S1] + \Pr[R0|S0]\Pr[S0]} = \frac{p_a p}{p_a p + (1 - p_b)(1 - p)}$$

(a) 原理　　(b) 示例

图 18.7　全概率和贝叶斯定理图例

图 18.7(b)表示公式的计算过程。图中，抽样空间是最小整数的正方形。正方形的一半对应 S0，另一半对应 S1，故 $\Pr[S0] = \Pr[S1] = 0.5$。类似地，正方形的一半对应 R0，另一半对应 R1，所以 $\Pr[R0] = \Pr[R1] = 0.5$。在 S0 域中，1/4 区域对应 R1，故 $\Pr[R1|S0] = 0.25$。其他的条件概率同样是显而易见的。

基于比率的错误举例

考虑下面的情况。病人的一些疾病测试结果为肯定性的(表明他患有疾病)。已知：

- 测试准确性是 87%(即如果病人患有疾病，87%的时间中，测试产生正确的结果，如果病人未患疾病，87%的时间中，测试产生正确的结果)
- 人口中疾病发生率为 1%

假设测试结果是肯定的，而病人实际上没有患有此疾病的概率是多少？也就是问虚假报警的概率是多少？我们用贝叶斯定理得到正确答案：

$$\Pr[\text{well/positive}] = \frac{\Pr[\text{positive/well}]\Pr[\text{well}]}{\Pr[\text{positive/disease}]\Pr[\text{disease}] + \Pr[\text{positive/well}]\Pr[\text{well}]}$$

$$= \frac{(0.13)(0.99)}{(0.87)(0.01) + (0.13)(0.99)} = 0.937$$

因此，在绝大多数情况下，当发现一个疾病条件，那么它就是虚假报警。

[PIAT91]使用的这个问题是很多人面临的问题。多数人给出的答案是 13%，包括很多医生在内的绝大多数认为答案应该低于 50%。推断结果是错误的，医生悲叹道："如果你没有问题，没有必要做临床测试！"对于大多数认为自己有问题的人来说，出现这种情况的原因是在用

直觉解决问题时，没有考虑事件发生的基本率。这类错误称为基于比率的错误。

如何确定这个问题？假设我们能将两种结果的正确率提高到 99%。也就是说，我们有 Pr[positive/disease] = 0.99，Pr[negative/well] = 0.99。代入公式(18.2)，我们得到 Pr[well/positive] = 0.01。因此，如果我们以 99%的概率准确地发现疾病和准确地发现没有疾病，则虚假报警率只是 1%。但是，现在假设在人口中，疾病事件只是 1/10 000 = 0.000 1，则我们以 99%的虚假报警率结束。在实际情况下，[AXEL00]发现入侵检测系统中的概率是这样一种情况：虚假报警率并不令人满意。

第19章　恶意软件

这一章主要分析恶意软件，尤其是病毒和蠕虫。

19.1　病毒及相关的威胁

对计算机系统来说，最复杂的威胁可能就是那些利用计算机系统的弱点来进行攻击的恶意程序。在这一章中，我们主要讨论应用程序，如编辑和编译程序。

在本章的开始，我们概述了软件威胁的范围，其余部分主要讨论病毒，首先介绍它们的本质，然后介绍一些相应的措施。

19.1.1　恶意程序

图 19.1 列出了软件威胁（恶意程序）的所有分类。这些威胁大致可以分为两类：依赖于宿主程序的和独立于宿主程序的威协。前者本质上来说是不能独立于应用程序或系统程序的程序段，后者是可以被操作系统调度和这行的自包含程序。

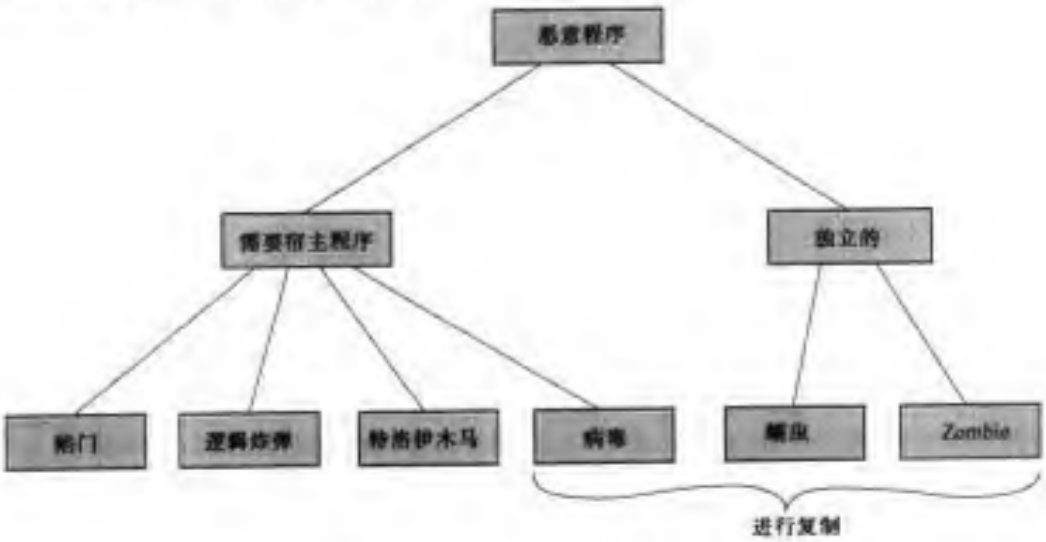


图 19.1　恶意程序的分类

我们也可以按其是否进行复制而将软件威胁分成两类：不进行复制的和进行复制的威胁。前者是在宿主程序被调用来执行某一特定功能时被激活的程序段；后者是指一个程序段（病毒）或一个独立的程序（蠕虫病毒、细菌），当它执行时，可能会对自身进行复制，而且这些复制品将会在该系统或其他系统中被激活。

尽管图 19.1 的分类对我们所讨论的问题是有用的，但它并不是一种完整的描述。事实上，逻辑炸弹和特洛伊木马也可以属于病毒的一部分。在这一节的其他部分中，我们简要的介绍除病毒以外的其他一些恶意程序。

陷门

陷门是程序的一个秘密入口，通过它，用户可以不按照通常的访问步骤就能获得访问权。许多年来，陷门一直被程序员合理地用在程序的调试和测试中。程序员在开发一个含有验证过程或一个经过长期计划的、需要用户输入一些不同值来运行的应用程序时，要使用陷门。为了调试程序，开发者通常希望能得到特权或避免所有必须进行的计划或验证。程序开发者还希望能确信存在着某种激活程序的方法，它被错误地用在应用程序的验证过程中。陷门是用来识别一些特殊的输入次序的代码，它在以某个用户身份证运行时或事件以某一不可能的次序发生时被激发。

当陷门被一些无耻之徒用来作为获得未授权的访问权的手段时，它就成为一种威胁。在影片战争游戏中，陷门就被描述成为对系统进行攻击的基本思想。关于陷门的另一个例子就是在 Multics 操作系统的开发期间，美国空军“老虎队”（模拟的敌手）使用了入侵检测，所用到的一个战略就是向正在运行 Multics 的站点发送操作系统的虚假更新，此更新含有一种能被陷门激活并使老虎队获得访问权的特洛伊木马程序（后面会讲到的一种恶意程序）。这种威胁是如此地隐蔽，使得 Multics 开发者很难发现它，即使获悉该威胁的存在，他们也很难找出它[ENGE80]。

要实现操作系统对陷门的控制是很困难的。安全的策略必须集中在程序的开发和软件的更新活动上。

逻辑炸弹

逻辑炸弹是最早出现的程序威胁之一，它预先规定了病毒和蠕虫的发作时间。逻辑炸弹是嵌在合法程序中的、只有当特定的事件出现时才会进行破坏（爆炸）的一组程序代码。例如，可以将特定文件是否存在、一个星期中的某一天或某个特定用户使用电脑等作为逻辑炸弹的触发条件。一旦这些条件被满足，逻辑炸弹就会激发，从而破坏硬盘数据乃至整个文件，甚至会引起“死机”或其他的一些危害。逻辑炸弹的一个典型事例就是 Tim Lloyd 事件。Tim Lloyd 所设计的逻辑炸弹使他所在的公司蒙受了超过 1000 万美元的损失，并导致 80 位员工失业[GAUD00]。最后，Tim Lloyd 被处以 41 个月的监禁以及 200 万美元的赔偿金。

特洛伊木马

特洛伊木马程序是一种实际上或表面上有某种有用功能的程序，它内部含有隐蔽代码，当其被调用时会产生一些意想不到的后果。

特洛伊木马程序使计算机潜伏执行非授权功能。例如，为了在某个共享系统中获得对其他用户文件的访问权，某一用户可以设计一个木马程序，当该程序被执行时，它会通过改变调用用户文件的许可权，从面获得对该文件的访问权。该设计者通过将本马程序放在普通的目录中并以看起来有用的程序名为其命名，来促使其他用户运行该程序。例如，程序表面上会以一个令人满意的形式产生一个用户文件列表，当其他用户运行该程序时，设计者就能获得对其他用户文件中信息的访问权。再例如，如果利用修改了的编译程序向正被编译的程序中插入一些附加的程序代码，如系统逻辑程序[THOM84]，那么这种特洛伊木马程序就很难被检测出来。设计者还在此逻辑程序中设计了一个陷门，使得他能够通过某种特殊的路径在网络上

进行连接主机(服务器)的操作,而我们是不可能从逻辑程序的原代码中找到该木马程序的。

特洛伊木马程序另一个比较普遍的危害是对数据的破坏。程序表面上看起来是在执行某种有用的功能(如一个计算程序),但实际上却在悄悄地删除用户文件。例如,哥伦比亚广播公司的一名执行经理就曾受到木马程序的攻击,结果他机器上所有的数据都被毁坏了[TIME90]。特洛伊木马程序是按某种图解程序被转移到电子布告栏系统中的。

Zombie

Zombie 是这样一种程序:它秘密地接管其他依附在 Internet 上的计算机,并使用该计算机发动攻击,而且这种攻击是很难通过追踪 Zombie 的创建者查出来的。Zombie 被用在对拒绝服务的攻击上,尤其是对 Web 站点的攻击。它被放置在成百上千的、属于可信任第三方的计算机中,通过向 Internet 发动不可抵抗的攻击取得对目标 Web 站点的控制。

19.1.2 病毒的特性

病毒是一种可以通过修改自身来感染其他程序的程序,这种修改包括对病毒程序的复制,复制后生成的新病毒同样具有感染其他程序的功能。

生物病毒是一种微小的基因代码段(DNA 或 RNA),它能掌管活细胞机构并采用欺骗性手段生成成千上万的原病毒的复制品。和生物病毒一样,计算机病毒执行使自身能完美复制的程序代码。通过寄居在宿主程序上,计算机病毒可以暂时控制该计算机的操作系统盘。没有感染病毒的软件一经在受染机器上使用,就会在新程序中产生病毒的新拷贝。因此,通过可信任用户在不同计算机间使用磁盘或借助于网络向他人发送文件,病毒是可能从一台计算机传到另一台计算机的。在网络环境下,访问其他计算机的某个应用或系统服务的功能,给病毒的传播提供了一个完美的条件。

病毒程序可以执行其他程序所能执行的一切功能,惟一不同的是它必须将自身附着在其他程序(宿主程序)上,当运行该宿主程序时,病毒也跟着悄悄地执行了。一旦病毒程序被执行,它就能执行一些意想不到的功能,如删去文件。

在其生命周期中,病毒一般会经历如下四个阶段:

- **潜伏阶段**:这一阶段的病毒处于休眠状态,这些病毒最终会被某些条件(如日期、某特定程序或特定文件的出现或内存的容量超过一定范围)所激活。并不是所有的病毒都会经历此阶段。
- **传染阶段**:病毒程序将自身复制到其他程序或磁盘的某个区域上,每个被感染的程序又因此包含了病毒的复制品,从而也就进入了传染阶段。
- **触发阶段**:病毒在被激活后,会执行某一特定功能从而达到某种既定的目的。和处于潜伏期的病毒一样,触发阶段病毒的触发条件是一些系统事件,包括病毒复制自身的次数。
- **发作阶段**:病毒在触发条件成熟时,即可在系统中发作。由病毒发作体现出来的破坏程度是不同的:有些是无害的,如在屏幕上显示一些干扰信息;有些则会给系统带来巨大的危害,如破坏程序以及文件中的数据。

多数病毒是基于某种特定的方式进行工作的,如某个特定的操作系统或某个特定的硬件平台。因此,攻击者们经常利用某特定系统的细节和弱点来设计病毒程序。

病毒的结构

病毒可以被放在可执行文件首部、尾部或中间，关键是要保证当调用受染文件时，首先被执行的应是病毒程序，然后才是原主程序。

病毒结构的一般性描述如图 19.2 所示[COHE94]，在该程序中，病毒程序 V 位于受染文件的首部，现假设调用该程序时，程序的入口指向该程序的第一行，即 V 取得了系统控制权。

```
        program V := 

{goto main;
      1234567;

          subroutine infect-executable := 
          {loop:
            file := get-random-executable-file;
             if (first-line-of-file = 1234567)
                    then goto loop
                    else prepend V to file;}

          subroutine do-damage := 
               {whatever damage is to be done}

          subroutine trigger-pulled := 
               {return true if some condition holds}

main: main-program := 
               {infect-executable;
                if trigger-pulled then do-damage;
                goto next;}

next:
}
```

图 19.2　一个简单的病毒

受染程序的工作机理如下：程序第一行代码的功能是跳转执行病毒程序，第二行代码是个特殊的标记，它被用来判断一个可能被感染文件是否已经感染病毒。当该程序被调用后，很快又将控制交给病毒程序模块，病毒程序首先寻找未被感染过的可执行文件并感染之，然后，病毒通常会对系统进行破坏活动，系统程序每被调用一次，该破坏活动就会进行一次。如果该破坏行为是个逻辑炸弹，那它就仅在某个特定条件满足时才会被激发。最后，病毒程序把控制权交还给原主程序。如果文件受染时间很短，那么用户一般是不会察觉到文件受染前、后的变化的。

上面所讲述的这类病毒是很容易被检测出来的，因为受染文件的长度与受染前相比变大了。一个逃避这种检测的方法就是将受染文件压缩到受染前的长度。图 19.3 [COHE94]显示了上述过程。在此病毒程序中，关键行都用数字进行了标记。具体操作过程如图 19.4 所示，我们假设程序 P_1 被病毒程序 CV 感染了，当 P_1 被调用时，CV 就获取了系统控制权，具体步骤如下所示：

```
program CV : =

{goto main;
      01234567;

      subroutine infect-executable : =
                  {loop:
                        file : = get-random-executable-file;
                  if (first - line - of - file = 01234567) then goto loop;
            (1)compress file;
            (2)prepend CV to file;
            }

main:main = program : =
                  {if ask-permiseion then infect-executable;
            (3)uncompress rest-of-file;
            (4)run uncompressed file;}
            }
```

图 19.3 压缩病毒的逻辑

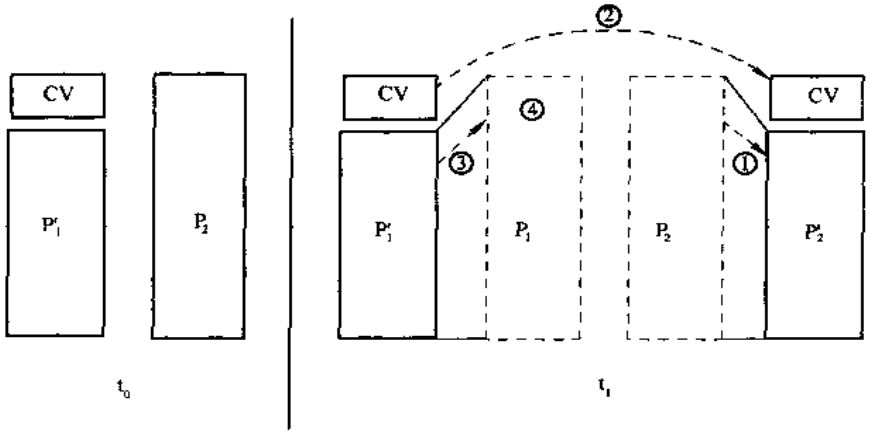


图 19.4 病毒的压缩过程

1. 将那些已发现的未受染文件 P_2 压缩成 P'_2，使 P'_2 和 CV 的总长度小于 P_2 的长度。
2. 将病毒的复制体放到被压缩文件 P'_2 的首部。
3. 将压缩后的原始受染文件 P'_1 解压。
4. 执行解压后的原始程序。

在这个例子中，病毒程序只进行了传染。正如前面所述，这种病毒可能含有逻辑炸弹。

病毒的早期传染方式

病毒一旦通过感染某文件获得入侵系统的入口，那么当受染文件被执行时，它就会感染该系统中的其他一部分或全部的可执行文件。因此，我们完全可以通过预防病毒进入系统的方法来阻止病毒的传播。遗憾的是，由于病毒程序本身可能会作为文件的一个组成部分，所以预防就变得特别困难。所以，除非你愿意自己做每块硬件，并亲手编写系统程序以及应用程序，否则你的机器就是脆弱的(易受攻击的)。

多数病毒的传播都开始于使用磁盘将程序拷贝到某计算机上，这些磁盘中可能含有游戏文件，也可能含有一些简单但很便利的、有用的东西。例如，某人可以从自己家中的计算机上将它复制并安装到他办公室的计算机中。更让人难以置信的是，当我们从某应用软件制造商那里买到用收缩性薄膜包装的磁盘时，该磁盘中可能就已经含有病毒了。只有少数病毒是通过网络进行传播的。例如，当某人下载了游戏软件或实用软件后，他可能会发现这些软件中已含有病毒。

19.1.3　病毒的种类

自病毒问世以来，病毒制造者和反病毒软件制造者之间的斗争就从未间断过。我们不但有对付现有病毒的有效技术，而且还发展了新的技术。[STEP93]一书将现有的比较典型的病毒按如下方式进行了分类：

- **寄生性病毒**：这是一种比较传统但仍然常见的病毒。寄生性病毒将自身附着在可执行文件上并对自身进行复制。当受染文件被执行后，它又会继续寻找其他的可执行文件并对其进行感染。
- **常驻存储器病毒**：这种病毒以常驻系统的程序的形式寄居在主存储器上，从这点看，这类病毒会感染所有类型的文件。
- **引导扇区病毒**：此类型的病毒感染主引导记录或引导记录，在系统从含有病毒的磁盘上引导装入程序时进行传播。
- **隐蔽性病毒**：设计这种病毒的目的就是为了躲避反病毒软件的检测。
- **多态性病毒**：这种病毒在每次感染时，放入宿主程序的代码互不相同，不断变化，因此采用特征代码法的检测工具是不能识别它们的。

早先就有关于隐蔽性病毒的讨论：有些病毒通过将受染文件压缩，使其长度恰好等于未感染时的长度；有些则可能使用许多手段更为高超的技术，例如，有些病毒会在磁盘的输入/输出例程上放置中途拦截逻辑，一旦用户通过例程试图访问磁盘中的可疑部分，它就会将原始的未感染病毒的程序呈现到用户面前。因此，隐蔽性并不是使用在上面所讨论的病毒中的专业术语，而是病毒用来逃避检测的一种技术。

多态性病毒在进行感染时，产生功能相似但字节排列方式截然不同的多个样本。和隐蔽性病毒一样，它的目的也是反跟踪。多态性病毒每次感染时，放入宿主程序的代码互不相同，不断变化。为了实现这种变化，必须在病毒程序中插入一些附加指令或改变程序代码的组合方式。一个很有效的方法就是对病毒程序进行加密。病毒程序中有这样一部分（我们通常称之为“变异引擎”）会随机产生一个加密密钥，用来加密程序的其余部分。密钥是和病毒程序存放在一起的，而且“变异引擎”本身也在变化。当受染文件被调用时，病毒程序就会使用现有的随机密钥来加密自身。当病毒再次进行复制时，又会产生新的随机密钥。

病毒制造者的另一个秘密武器是病毒制造工具。借助于这些工具，病毒初学者可以很快地生成各种不同的病毒程序。尽管利用工具制造出来的病毒无法与那些病毒制造大师制造出来的病毒相比，但仅就这些新型的病毒就足以让反病毒工作人员头疼了。

19.1.4　宏病毒

这些年来，病毒总数急剧上升。事实上，引起这种变化的一个极其重要的原因就是一种新

型病毒——“宏病毒”的出现。据国际计算机安全中心(www.ncsa.com)报道,在现有的计算机病毒中,宏病毒就占据了三分之二。

由于以下原因,宏病毒显示出了极其严重的危害:

1. 宏病毒不依赖于单一的平台。事实上,所有的宏病毒都会感染 Microsoft Word 文档,任何一种支持 Word 的硬件平台或操作系统都可能被感染。
2. 宏病毒只感染文档文件,不感染程序文件。而绝大多数信息都是以文档形式(而非程序形式)输入到计算机系统中的。
3. 宏病毒很容易被传播。一个非常常见的方法就是通过电子邮件的形式进行传播。

宏病毒有利于人们发现 Word 和其他办公软件(如 Microsoft Excel,也就是宏)的特点。就其本质来讲,宏是 Word 文档或其他类型文件中的可执行程序代码。特别是用户可以使用宏来自动完成某种重复的任务,而不必重复敲击键盘。宏语言通常是用 Basic 语言编写的。用户可以在宏中定义一连串的按键操作,这样,当输入某一功能键或特殊的功能键的组合时,该宏就被调用了。

自动执行宏为宏病毒的产生提供了可能性。宏被自动调用时,是不需要明确的用户输入的。常见的自动执行事件有打开文件、关闭文件或开始某个应用。一旦宏被执行,它就会将自身复制到其他文档中,删除文件并对用户系统造成其他危害。在 Microsoft Word 中,有三种形式的自动执行宏:

- **自动执行**:如果某个被称为“自动执行批处理文件”的宏处在“normal.dot”模板或存放在 Word 启动目录的通用模板中,那么只要 Word 一启动,它就会被执行。
- **自动宏**:当所定义的事件(如打开文件,关闭文件,新建一个文档,或停止 Word 的运行)发生时,自动宏就会执行。
- **命令宏**:如果通用宏文件中或附着在文档上的宏包含有 Word 的命令名称,则只要用户一调用该命令(如保存文件),该宏就被调用。

宏病毒常用的传播技术如下所示:附着在 Word 文档中的自动宏或命令宏通过电子邮件或移动磁盘的方式进入系统。用户一打开文档,宏就被执行,并将自身复制到通用宏文件中。当再次打开 Word 时,受染的通用宏就被激活了。当这种宏执行时,它就能复制自身并破坏系统。

后续发布的 Word 提高了预防宏病毒的可能性。例如,Microsoft 提供了一种可选的宏病毒检测工具,该工具可以用来检测可疑的 Word 文件并能去除用户打开宏文件时所潜在的危险。各种反病毒工具制造商也提供了检测并防治宏病毒的工具。和其他类型的病毒一样,宏病毒仍在发展。

19.1.5 电子邮件病毒

近几年来发展比较快的恶意程序之一是电子邮件病毒。传播迅速的电子邮件病毒如“Melissa”(美丽沙)是利用 Microsoft Word 宏嵌在电子邮件附件中。一旦接受者打开邮件附件,该 Word 宏就被激活:

1. 电子邮件病毒向用户电子邮件地址簿中的地址发送被感染文件。
2. 病毒就地发作。

1999 年，曾出现一种破坏力极强的电子邮件病毒。用户根本不必打开邮件附件而只要打开含毒邮件，这种新型的病毒就会被激活。该病毒使用电子邮件包支持的 Visual Basic 脚本语言编写。

我们可以看到新一代的恶意程序都是利用电邮特征或以电邮方式在 Internet 上进行传播的。病毒一旦被激活，就会通过打开邮件或邮件附件的方式，将自身向它所知道的所有地址传播。因此，过去常常要花几个月甚至几年时间来传播的病毒现在只要花几小时就能完成传染任务。这就使得反病毒软件要在病毒大面积发作之前对其做出回应变得困难了。所以我们必须要加强 Internet 领域中计算机应用软件的安全程度，以对付日益增长的威胁。

19.1.6　蠕虫病毒

电子邮件病毒将自身从一台计算机传播到另一台计算机，因此它具有蠕虫病毒的一些特征。但由于它仍需要人为地手工移动，所以我们把它另外作为一类。蠕虫病毒会时刻寻找更多的计算机并伺机对其传染。那些被感染的机器又会作为一种自动发射缓冲器，以进一步感染其他机器。

网络蠕虫病毒利用互联网将自身从一台计算机传播到另一台计算机。只要系统中的蠕虫病毒处于活动状态，它就能像计算机病毒或可移动的木马程序那样对系统做出巨大的破坏行为。

网络蠕虫病毒利用以下网络工具进行传播：

- **电子邮件设备**：蠕虫病毒将自身的拷贝以邮件的形式发送到其他系统。
- **远程执行功能**：蠕虫病毒在其他系统中执行自身的拷贝。
- **远程登录功能**：蠕虫病毒以用户身份进入远程系统，然后使用命令将自身从一台计算机复制到另一台计算机。

新的蠕虫病毒的拷贝会在远程计算机上进一步运行并执行一些功能，而且以同样的方式继续传播。

网络蠕虫病毒显示出类似于计算机病毒的一些特征，它同样也具有四个阶段，即潜伏阶段、传染阶段、触发阶段和发作阶段。在传染阶段，蠕虫病毒一般执行如下操作：

1. 通过检查主表格或远程计算机的地址库，找到可进一步传染的其他计算机。
2. 和远程计算机连接。
3. 将自身拷贝到远程计算机并运行该拷贝。

网络蠕虫病毒在将自身复制到某台计算机之前，也会试图判断该计算机先前是否已被感染过。在分布式系统中，蠕虫病毒可能会以系统程序名或不易被操作系统察觉的名字来为自己命名，从而伪装自己。

和计算机病毒一样，网络蠕虫病毒也很难被反击。但是，如果我们能进一步加强网络系统和单机系统的安全，则可以减少蠕虫病毒的攻击。

Morris 蠕虫病毒

最为人们所熟知的蠕虫病毒可能就是 1998 年 Robert Morris 所设计的病毒。Morris 蠕虫病毒通过 Internet 并使用多种不同的技术在 UNIX 系统中传播。当某个拷贝开始执行时，它的首

要任务就是找到当前主机所知道的其他系统，从而可以从当前主机进入到其他系统。Morris 蠕虫病毒是通过检查各种各样的目录来完成该任务的。这些目录中可能含有以下内容：表明主机信任其他哪些系统的系统目录、用户邮件地址文件、用户赋予自身登录远程账户的许可权的目录，以及用来报告网络连接状态的文件。一旦找到其他系统，Morris 蠕虫病毒就会想方设法以获得对它们的访问权。

1. Morris 蠕虫病毒试图以合法用户的身份登录远程系统。病毒首先会试图打开局部口令文件，然后再使用它所得到的口令/密码和相应的用户标识部分。该设想基于许多用户会在不同的系统中使用相同的密码。为了得到用户的密码，蠕虫病毒执行一种专门用来破解密码的程序，并努力得到：
   (a) 每个用户的账户名及其简单的排列。
   (b) 包含 432 个内置口令的列表。Morris 蠕虫病毒认为该列表是可能的候选者。
   (c) 当前系统目录中的所有指令。
2. 在 finger 协议上安装窃听器，从而能知道远程用户的所在之处。
3. 在远程处理操作的调试选择权中设置陷门，这些远程处理操作是指接受或发送用户邮件。

上面所提到的这些攻击只要有一个能成功，Morris 蠕虫病毒就会和操作系统命令解释器进行通信。它会向该命令解释器发送一个比较短的引导程序，并发布某命令来执行该程序，然后在网络上终止连接主机(服务器)的操作。接下来该引导程序便收回父程序并下载病毒的其余部分。最后，新病毒就开始执行了。

近期的蠕虫病毒攻击

2001 年 7 月出现的红色代码病毒开创了病毒史的新纪元。红色代码病毒利用微软 IIS (Microsoft Internet Information Server)的安全漏洞进行病毒的传播，它使得系统在 Windows 状态下不能检查系统文件。红色代码病毒利用任意探测到的 IP 地址，对其他主机进行传播。在某个特定的时间内，它只进行传播，然后再从不同主机向政府 Web 站点发送大量的包以发动“拒绝服务”的攻击。红色代码病毒也会周期性地暂停活动和恢复活动。第二次大规模的攻击中，红色代码病毒仅用 14 小时就感染了近 36 万台计算机，加上对目标服务器攻击所造成的危害，它消耗了大量的 Internet 资源并损坏了大量的服务器。

红色代码 2 是攻击微软 IIS 时产生的变种。此外，该改良版建立了一个后门，使得被攻击的机器“后门大开”，从而也使得攻击者可以直接控制被攻击的机器。

在 2001 年末，出现了像“尼姆达”病毒那样破坏力极强的蠕虫病毒。“尼姆达”病毒以多种方式从一台计算机传播到另一台计算机：

- 通过电子邮件的传递。
- 通过开放的网络传递。
- 通过浏览不安全的网站从 Web 服务器感染。
- 通过主动扫描或利用 Microsoft IIS 4.0/5.0 directory traversal 的缺陷感染。
- 通过扫描“红色代码”病毒中的后门进行感染。

“尼姆达”病毒通常修改如“.htm、.html 和 .asp”类型文件的 Web 文档或受染系统中的可执行文件，并以不同的文件名对自身进行大量的拷贝。

19.2　计算机病毒的防治策略

19.2.1　反病毒方法

解决病毒攻击的理想方法是对病毒进行预防，即在第一时间阻止病毒进入系统。尽管预防可以降低病毒攻击成功的概率，但一般说来，上面的目标是不可能实现的。下面列出了一个比较有效的、可行的方法：

- **检测**：一旦系统被感染，就立即断定病毒的存在并对其进行定位。
- **鉴别**：对病毒进行检测后，辨别该病毒的类型。
- **清除**：在确定病毒的类型后，从受染文件中删除所有的病毒并恢复程序的正常状态。清除被感染系统中的所有病毒，目的是阻止病毒的进一步传染。

如果对病毒检测成功但鉴别或清除没有成功，则必须删除受染文件并重新装入无毒文件的备份。

病毒和反病毒技术都在不断发展。早期的病毒是一些相对简单的代码段，可以用相应的较简单的反病毒软件来检测和清除。随着病毒技术的发展，病毒和反病毒软件都变得越来越复杂化和经验化。

[STEP93]一书将反病毒软件的发展分为四代：

- **第一代**：简单的扫描程序。
- **第二代**：启发式的扫描程序。
- **第三代**：主动设置陷阱。
- **第四代**：全面的预防措施。

第一代扫描软件要求知道病毒的特征以鉴别之。病毒可能含有“通配符”，但就其本质而言，所有的拷贝都具有相同的结构和排列方式。那些基于病毒具体特征的扫描软件只能检测已知的病毒。另一种类型的第一代扫描软件包含文件长度的记录，通过比较文件长度的变化来确定病毒的种类。

第二代扫描软件不依赖于病毒的具体特征，而是利用自行发现的规律来寻找可能存在的病毒感染。有一种扫描软件就是用来寻找和病毒相联系的代码段的。例如，一种扫描软件可以用来寻找多态性病毒中用到的加密圈的起点，并发现加密密钥。一旦该密钥被发现，扫描软件就能对病毒进行解密，从而鉴别该病毒的种类，然后就可以清除这种病毒并将该程序送回到服务器。

第二代扫描软件的另一种方法是进行完整的检查。“校验和”可以附加在文件上。如果文件感染了某种病毒，但“校验和”没有改变，则可以用完整性检查的方法来找出变化。为了对付一种在感染文件时能改变“校验和”的病毒，我们必须使用 hash 函数来进行加密。我们还必须将加密密钥和程序代码分开存储以防止病毒产生新的 hash 代码并进行加密。通过使用 hash 函数（而不是一个简单的“校验和”），可以防止病毒调整程序产生同前面一样的 hash 代码。

第三代程序是存储器驻留程序，它可以通过受染文件中的病毒的行为（而非其特征）来鉴别病毒。这种程序的优点是不需要知道大量的病毒的特征以及启发式的论据，它只需要鉴别一小部分的行为，该行为表明了某一正试图进入系统的传染行为。

第四代产品是一组含有许多和反病毒技术联系在一起的包，它包括扫描软件和主动设置陷阱。此外，该包还包括一种访问控制功能，这就限制了病毒入侵系统的能力和病毒为了进行传播更新文件的能力。

反病毒的技术还在不断发展。利用第四代检测包，我们可以运用一些综合的防御策略，拓宽防御范围，以适应多功能计算机上的安全需要。

19.2.2 高级反病毒技术

大量经验丰富的反病毒手段和产品相继出现。在这一小节中，我们要强调两种最主要的技术。

通用解密技术

只要保持快速的扫描，通用解密技术(GD)就能使反病毒程序很容易地检测到病毒，即使是最为复杂的多态性病毒[NACH97]。当我们执行感染了多态性病毒的文件时，我们知道病毒必须在对自身解密后才能工作。为了检测出这种特征，我们必须使用 GD 扫描器对该可执行文件进行扫描。这里提到的扫描器包括以下几个要素：

- **CPU 仿真器**：一种实际上基于软件的计算机。可执行文件中的指令并不是由基本的处理器处理的，而是被仿真器解释的。仿真器包括所有的寄存器软件版本和其他处理器硬件的软件版本。因此，经仿真器解释的程序一般不会影响基本的处理器。
- **病毒特征扫描器**：一种通过扫描目标代码来寻找现有病毒特征的模块。
- **仿真控制模块**：控制目标代码的执行。

在每次开始伪装时，仿真器都要对目标程序进行解释，而且一次只能解释一个。因此，如果代码中含有解密病毒程序和提高病毒暴露概率的解密例程，那么该代码就会被解释。事实上，病毒程序就是通过暴露自身来对反病毒程序起作用的。控制模块会周期性地中断解释，来扫描目标程序以找出病毒的特征。

由于目标程序是在完全被控制的环境下被解释的，所以它可以使个人计算机环境免受破坏。

设计 GD 扫描器最大的困难就是必须确定每次进行解释的时间到底应该多长。病毒程序通常是在执行某程序后被激活的，但这并不是我们所需要知道的。我们知道扫描器仿真某特定程序的时间越长，它就越有可能找出隐藏的病毒。但是，反病毒程序只能占据有限的时间和资源，否则便会引起用户的抱怨。

数字免疫系统

IBM 公司开发的数字免疫系统是一种综合的预防病毒的方法[KEPH97a, KEPH97b]，开发的原因是 Internet 上传播的病毒日益严重的威胁。我们首先介绍这种病毒所构成的威胁，然后再概述 IBM 公司所使用的方法。

从传统上讲，新的病毒或病毒的变种以相对较慢的速度传播，形成一种较小的威胁，只要反病毒软件每月都进行更新，就足以对付这种威胁，Internet 在病毒的传播上只担当相对较轻的角色。但正如[CHES97]一书所述，近几年 Internet 技术中有两种主要趋势增大了病毒传播的速率：

- **完整的邮件系统**：Lotus Notes 和 Microsoft Outlook 这类系统的出现使得向任何人发送任何信息和用所接受的信息来进行工作都变得非常简单。
- **可移动程序系统**：Java 和 ActiveX 这类软件的兼容性使得程序可以将自身从一个系统迁移到另一个系统。

为了对付基于 Internet 进行传播的病毒所构成的威胁，IBM 公司开发了数字免疫系统原型，该系统阐述了前面已讨论过的程序仿真的用途，并提供一个用途广泛的仿真系统和病毒检测系统，该系统的具体目标是对病毒做出快速的反应，使得病毒一入侵系统，立即就被标记。当有新的病毒进入系统时，免疫系统会自动将其捕获并对其进行分析，然后增加一些相应的检测和防御措施并清除病毒。最后将这种新病毒的有关信息送到 IBM AntiVirus 在线系统，使该病毒再次出现时立即就被检测出来。

图 19.5 给出了数字免疫系统操作的典型步骤：

1. 每台 PC 机上的监视程序利用大量的启发式论据对现有的病毒进行推断。这些启发式论据源自于系统行为、程序可疑的变化和同类病毒的特征。监视程序将受感染程序的拷贝发送到该组织的管理机上。
2. 管理机对该样本进行加密，并将加密后的样本送到病毒中心分析机。
3. 为了进一步进行分析，该分析机创造了一个受染程序能够运行的环境。出于这一目的所用到的技术有：模拟或创造一个保护环境，在该环境下的可疑程序能被执行或监视。然后，病毒分析机会生成鉴别和清除病毒的指令。
4. 将指令执行的结果送回到管理机。
5. 管理机将该指令送往已感染病毒的客户机。
6. 管理机还将该指令送往系统的其他客户机。
7. 全球的用户接收反病毒软件的定期更新，以避免遭到新病毒的攻击。

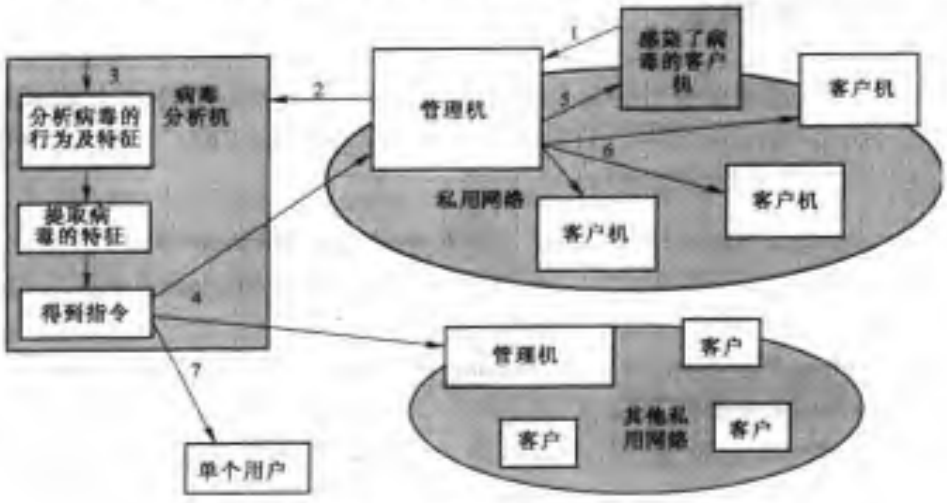


图 19.5　数字免疫系统

数字免疫系统是否成功取决于病毒分析机侦测新种病毒的能力。通过对新出现的病毒不断地分析和监视，可以经常更新数字免疫软件，从而减少相关的威胁。

19.2.3 行为 - 阻塞软件

和启发式论据以及指纹扫描器不同,行为 - 阻塞软件把主机操作系统和受到恶意攻击时对程序的实时监视行为结合在一起,并在那些恶意的行为对系统进行攻击之前就将它们阻塞。监视行为的内容包括:

- 试图打开、查看、删除或修改文件。
- 试图格式化磁盘和其他不可恢复的磁盘操作。
- 对可执行文件和程序代码进行修改。
- 对关键性的系统设置进行修改,例如系统的启动设置。
- 电子邮件的脚本程序以及通知客户发送可执行内容。
- 网络通信的初始化。

如果行为阻塞者检测到某程序可能有恶意行为,它就实时阻塞该程序或结束该程序的执行。这就给那些已制定的反病毒检测技术(如"指纹"识别、探索性方法)提供了基本的有利条件。由于病毒程序可以用数以亿计的方法来打乱并重组自身,所以它很可能逃避"指纹"识别或探索性方法的检测,最终可以向系统发出一个合理的请求。假使行为阻塞者能中途拦截这些请求,那么无论该程序如何混乱,他都能够确认并阻塞该行为。

对正在运行的软件进行实时监控的能力给予行为阻塞者莫大的好处,但它还是存在缺点。由于恶意代码在被确认之前,肯定要在该系统中运行,所以就可能在被行为阻塞系统检测并阻塞前,对系统造成巨大的危害。例如,一个新的病毒可能会在感染某文件并被阻塞前将硬盘上一些看似不太重要的文件移走。用户即使制止了病毒的进一步传染,他也不能定位文件,从而会导致系统效率的降低或可能引起一些错误。

19.3 推荐读物和网址

[HARL01]一书有助于读者对病毒的精确理解。如果读者想对病毒有个基本的了解,则建议读者阅读以下几本书:[CASS01]、[FORR97]、[KEPH97]和[NACH97]。而[MEIN01]一书则为读者提供了对付红色代码病毒的方法。

CASS01 Cass, S. "Anatomy of Malice." *IEEE Spectrum*, November 2001.

FORR97 Forrest, S.; Hofmeyr, S.; and Somayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

HARL01 Harley, D.; Slade, R.; and Gattiker, U. *Viruses Revealed*. New York: Osborne/McGraw-Hill, 2001.

KEPH97 Kephart, J.; Sorkin, G.; Chess, D.; and White, S. "Fighting Computer Viruses." *Scientific American*, November 1997.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.


推荐网址：

- **AntiVirus On-line**：IBM 公司关于病毒信息的站点，它是最好的网站之一。

19.4　关键术语、思考题和习题

19.4.1　关键术语

数字免疫系统	电子邮件病毒	逻辑炸弹
宏病毒	恶意软件	多态性病毒
隐蔽性病毒	陷门	特洛伊木马
病毒	蠕虫	zombie

19.4.2　思考题

19.1　简要地定义图 19.1 中各个类型的恶意软件。

19.2　在病毒操作期间压缩的任务(目的)是什么?

19.3　病毒运行期间加密的任务(目的)是什么?

19.4　病毒运行期间最典型的阶段是哪一个?

19.5　在整个运行周期中，蠕虫病毒是怎样进行传播的?

19.6　什么是数字免疫系统?

19.7　行为－阻塞软件是怎样工作的?

19.4.3　习题

19.1　图 19.2 的病毒程序中有一个错误，请指出。

19.2　这样一个问题：是否能利用某个程序对软件进行分析，以此来判定该软件是否是病毒程序。现在我们假设这样的程序 D 已经存在，也就是说对某一程序 P，如果我们运行 D(P)，那么返回结果就是真(P 是个病毒程序)或假(P 不是病毒程序)。请思考如下程序：

```
Program CV :=
  { ...
  main-program :=
         {if D(CV) then goto next:
                else infect-executable;
         }
next:
  }
```

在上面的程序中，infect-executable 是这样一个函数，它扫描可执行文件并在这些文件中对自身进行复制，从而决定 D 是否能正确地判定 CV 是否是个病毒程序。

第20章 防 火 墙

防火墙是一种有效的防御工具，当通过广域网和 Internet 访问内部的主机或者网络时，以及通过内部的主机或者网络访问外部系统时，防火墙可使系统免于受到网络安全方面的威胁。

在本章的开始，我们概述了防火墙的功能和设计原理。接着，分析了防火墙自身的安全，特别介绍了可信系统的概念或安全操作系统。

20.1 防火墙的设计原理

企业、政府和其他组织里的信息系统已经经历了一番稳定的发展：

- 中心数据处理系统，它处于一台中心主机之中，这台主机支持一定数量直接连接的终端。
- 局域网(LAN)，将个人计算机、终端和主机相互连接。
- 驻地网，由一些局域网、互联的个人计算机、服务器组成，这其中还有可能包括一到两台中央主机。
- 企业内部网络，包括一些通过专用广域网连接起来的多重地理分布式驻地网。
- Internet 连接，在这个连接里，各个不同的驻地网都连入了 Internet，它们彼此之间可以依然通过专用广域网连接，也可以不用。

如今，对于大多数机构而言，是否连入 Internet 已经不再是一个需要考虑的问题，Internet 上大量有用的信息和服务对于他们而言是必须的。而且，机构内部的个人用户同样需要访问 Internet，如果他们自己所处的局域网不能够提供这种服务，那么他们也可以通过拨号上网的方法联入 Internet。话说回来，Internet 在向机构提供便利的同时，也使得外面的世界能够接触到本地网络并对其产生影响。这便对机构产生了威胁。虽然给每个工作站和驻地网都配置强大的安全特性是可能的，但却并不是一个实际的办法。设想一个拥有着成百上千个系统的网络，运行着各种不同版本的 UNIX 和 Windows 操作系统，一旦发现某个安全漏洞，那么每个有可能受到影响的系统都必须进行升级以弥补这个漏洞，这显然是一项庞大的工作。一种越来越为人们所接受的替代方法是防火墙。防火墙被嵌入在驻地网和 Internet 之间，从而建立受控制的连接并形成外部安全墙或者说是边界。这个边界的目的在于防止驻地网收到来自 Internet 的攻击，并在安全性将受到影响的地方形成阻塞点。防火墙可以是一台计算机系统，也可以由两台或更多的系统协同工作起到防火墙作用。

这一章，我们先了解防火墙的一般特性，然后介绍目前通用的防火墙的分类，最后叙述最常见的防火墙结构。

20.1.1 防火墙的特性

[BELL94]列出了防火墙的设计目标：

1. 所有的通信，无论是从内部到外部还是从外部到内部的，都必须经过防火墙。这一点可以通过阻塞所有未通过防火墙的对于本地网络的访问来实现。后文将要详述各种不同结构的防火墙。

2. 只有被授权的通信才能通过防火墙，这些授权将在本地安全策略中规定。不同类型的防火墙实现不同的安全策略，后面将要说明。
3. 防火墙本身对于渗透必须是免疫的。这意味着必须使用运行安全操作系统的可信系统。这部分内容将在 20.2 节讨论。

[SMIT97]列出了防火墙用以控制访问和加强站点安全策略的四项常用技术。起初，防火墙主要关心的是服务控制技术，到后来，它已经发展为以下四项：

- **服务控制**：决定哪些 Internet 服务可以被访问，无论这些服务是从内而外还是从外而内。防火墙可以以 IP 地址和 TCP 端口为基础过滤通信；也可以提供代理软件，在服务请求通过防火墙时接收并解释它们；或者执行服务器软件的功能，比如邮件服务。
- **方向控制**：决定在哪些特定的方向上服务请求可以被发起并通过防火墙。
- **用户控制**：根据用户正在试图访问的服务器，来控制他的访问。这个技术特性主要应用于防火墙网络内部的用户（本地用户）。它也可以应用到来自外部用户的通信；后者要求某种形式的安全认证技术，例如 IPSec（16 章提到过）。
- **行为控制**：控制一个具体的服务怎样被实现。举例来说，防火墙可以通过过滤邮件来清除垃圾邮件。它也可能只允许外部用户访问本地服务器的部分信息。

在进入防火墙的分类和配置的细节问题之前，我们最好是对于防火墙究竟能够做什么有个大概的认识。下面这些性能是防火墙力所能及的：

1. 防火墙定义了单一阻塞点，它使得未授权的用户无法进入网络，禁止了潜在的易受攻击的服务进入或是离开网络，同时防止了种种形式的 IP 欺骗和路由攻击。单一阻塞点的使用简化了安全管理，因为安全措施都被集中到了单个的或是成套的系统中。
2. 防火墙提供了一个监控安全事件的地点。对于安全问题的检查和警报可以在防火墙系统上实施。
3. 防火墙还是一个便利的平台，这个平台提供了一些与网络安全无关的功能。比如地址转换器，它把本地地址映射为 Internet 地址，又如网络管理功能，它用来审查和记录 Internet 的使用。
4. 防火墙可以作为 IPSec 的平台。利用隧道模式方法（16 章介绍过），防火墙可以用来实现虚拟专用网络。

防火墙也有它的局限性，其中包括：

1. 防火墙不能防御绕过了它的攻击。网络内部可能会有通过拨号连入 ISP 的能力。一个内部局域网，如果支持机架式调制解压器，那么它就为那些移动雇员和远程办公者提供了拨号接入网络的能力。
2. 防火墙不能消除来自内部的威胁，比如某个心怀不满的雇员或者某个私下里与网络外部攻击者联手的雇员。
3. 防火墙不能防止病毒感染过的程序和文件进出网络。事实上，安装了防火墙的网络系统内部，运行着多种多样的操作系统和应用程序，想通过扫描所有进出网络的文件、电子邮件以及信息来检测病毒的方法，是不实际的，也是不大可能实现的。

20.1.2 防火墙的分类

图 20.1 描述了三种最为常用的防火墙：包过滤路由器、应用级网关和电路级网关。我们依次进行探讨。

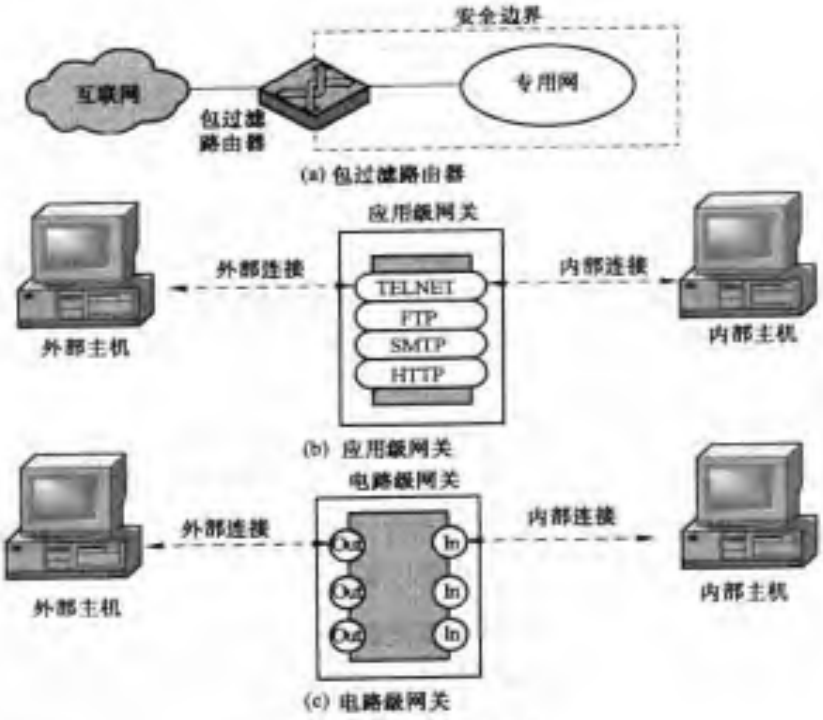


图 20.1 防火墙的类型

包过滤路由器

包过滤路由器依据一套规则对收到的 IP 包进行处理，决定是转发还是丢弃。路由器被特别设置成对两个方向（进入内部网络和从内部网络发出）的数据包进行过滤。过滤的具体处理方法根据数据包所包含的信息而定：

- **源 IP 地址**：产生 IP 数据包的系统的 IP 地址（例如 192.168.1.1）。
- **目的 IP 地址**：IP 数据包所要到达地系统的 IP 地址（例如 192.168.1.2）。
- **源和目的传输层地址**：指数据包在源系统和目的系统经过的传输层端口数，一些应用如 SNMP 和 TELNET 必须在这里进行。
- **IP 协议域**：对传输协议的定义。
- **接口**：对那些有两到三个端口的路由器，这部分信息规定了哪个是数据包的进入端口，哪个是预留给数据包的端口。

包过滤器根据事实上可看做一个规则表、由规则表和 IP 报头或 TCP 数据头内容的匹配情况来执行过滤操作。如果有一条规则和数据包的状态匹配，就按照这条规则来执行过滤操作。如果没有一条规则匹配，就执行默认操作。默认的策略可能是：

- **默认值 = 丢弃**:所有未明确允许转发的数据包都将被丢弃。
- **默认值 = 转发**:所有未明确规定需要丢弃的数据包都将被转发。

默认丢弃策略显得比较保守。起初,所有的服务都会被阻塞,服务必须依靠实例的积累逐步扩展,这时的防火墙在用户看来,更像是一个障碍物。默认转发策略方便了用户的使用,但也相应降低了安全性;网络安全管理员基本上要对每一个被发现的安全威胁立刻做出反应。

表 20.1 给出了包过滤规则表的一些例子。在每个表中,规则从上到下依次应用。“*”号是一个通配符,用来表示符合要求的每一种可能。这里我们假设使用默认丢弃策略。

表 20.1　包过滤的实例

A

处　理	内部主机	端　口	外部主机	端　口	说　明
阻塞	*	*	SPIGOT	*	这些人不被信任
通过	OUR-GW	25	*	*	与内部主机的 SMTP 端口有连接

B

处　理	内部主机	端　口	外部主机	端　口	说　明
阻塞	*	*	*	*	默认

C

处　理	内部主机	端　口	外部主机	端　口	说　明
通过	*	*	*	25	与外部主机的 SMTP 端口有连接

D

处　理	出发地	端　口	目的地	端　口	标　志	说　明
通过	本地主机	*	*	25		发往外部 SMTP 端口的包
通过	*	25	*	*	ACK	外部主机的回复

E

处　理	出发地	端　口	目的地	端　口	标　识	说　明
通	主机	*	*	*		本地主机的输出请求
通过	*	*	*	*	ACK	对本地请求的回复
通过	*	*	*	> 1024		到非服务器的通信

A:进入防火墙内部的邮件被允许通过(端口 25 专门供 SMTP 进入内部使用),但是只能发往一台特定的网关主机,从特定的外部主机 SPIGOT 发来的邮件将被阻塞,因为这台主机曾经在邮件中发送过邮件炸弹。

B:这是默认策略的一个清楚的描述。所有的规则表实际上都把这条规则当做最后的规则。

C:这个规则表的目的在于规定内部的每一台主机都可以向外部发送邮件。一个目的端口为 25 的 TCP 包将被路由到目的机器上的 SMTP 服务器。这条规则的问题在于把端口 25 用来作为 SMTP 接收只是一个默认设置;而外部机器的 25 端口可能被设置用来做其他的应用。从这条规则可以看出,一个攻击者可以通过发送一个 TCP 源端口为 25 的数据包来获得对内部机器的访问权。

D:这个规则表达到了表 C 所没有达到的效果。它利用了 TCP 连接的优点,一旦建立一个连接,那么 TCP 段被设置一个 ACK 标志,表示是另一方发来的数据段。因此,这个规则表就允许那些源 IP 地址是给定的某些主机,而目标 TCP 端口数是 25 的数据分组通过。并同时允许那些源端口数为 25 并且包含一个 ACK 标志的数据包通过。当然,我们必须清楚地指定源系统和目的系统,才能有效地定义这些规则。

E：这个规则表是一种处理 FTP 连接的方法。为实现 FTP，需要建立两个 TCP 连接：控制连接负责建立文件传输，数据连接负责实际文件的传输过程。数据连接使用与控制连接不同的端口，这个端口是在传输时动态分配的。大多数服务器使用低端口，它们往往是攻击者的目标；大多数对外部系统的呼叫则倾向于使用高端口，特别是大于 1023 的。因此，这个规则表在下列情况允许通过：

○ 从内部发出的数据包。

○ 对一个内部机器所建立的连接进行响应的数据包。

○ 内部机器上发向高端口的数据包。

这个方案要求系统设置为只有某些适当的端口可用。

规则表 E 表明了在包过滤层上处理应用程序存在着困难。处理 FTP 和类似应用程序的另一种方法是应用层网关，我们将在后面介绍。

包过滤路由器的优点在于它的简单性。因此，包过滤器对用户而言几乎是透明的，处理速度也很快。[WACK02]指出了包过滤器防火墙的缺点：

- 包过滤器防火墙不检查上层数据，因此，对于那些利用特定应用的攻击，防火墙无法防范。例如，包过滤防火墙不能阻塞具体的应用程序指令；它一旦允许某个应用程序通过，那么程序内所有的操作都将被允许通过。
- 由于防火墙可用的信息有限，它所提供的日志功能也十分有限。包过滤器日志一般只记载那些曾经做出过访问控制决定的信息（源地址、目的地址和通信类型）。
- 多数包过滤防火墙不支持高级用户认证方案。又是这种局限性，导致了防火墙缺少上层功能。
- 这种防火墙通常容易受到利用 TCP/IP 规定和协议栈漏洞的攻击，例如网络层地址欺骗。许多包过滤防火墙不能察觉对数据包 OSI 第三层的地址信息的修改。入侵者通常会采用欺骗攻击来躲过防火墙的安全控制。
- 最后，由于在这种防火墙作出安全控制决定时，起作用的只是少数几个因素，包过滤器防火墙对那种由于不恰当的设置而导致的安全威胁显得十分脆弱。换句话说，偶然性的改动可能会导致防火墙允许某些传输类型、源地址和目的地址的数据包通过，而事实上按照该系统的安全策略，这些数据包是应该被阻塞的。

下面是一些针对包过滤器防火墙的攻击以及相应的对策：

- **IP 地址欺骗**：入侵者从防火墙外部发送一个源地址为内部主机的数据包。攻击者试图利用假的地址来进入那些仅对源地址信赖的系统，在这些系统里，一旦数据包的源地址为防火墙内部的可信主机，它就将被允许通过。应对这种攻击的方法是，一旦在防火墙的外部接口处发现源地址是内部地址的数据包，就将它丢弃。
- **源路由攻击**：攻击者在来源位置注明数据包在 Internet 上传输时所应该采用的路由，由此希望绕过那些安全措施。应对措施是丢弃所有使用了这个选项的数据包。
- **微分片攻击**：入侵者使用 IP 分片选项来制造出非常小的分片，分片如此之小，使得 TCP 头信息只能被放在一个独立的分片中。这种攻击方法用来对付那些过滤规则只能依赖于 TCP 头信息的防火墙很是有效。攻击者的如意算盘是，过滤路由器仅仅检查第一个

分片，然后将后面的所有分片统统放行。然而，如果防火墙将那些协议类型为 TCP，IP 碎片偏移为 1 的小分片都丢弃，这种攻击也就失效。

状态检查防火墙

传统的包过滤器仅仅依据各个数据包的信息就对其实行过滤操作，而不去考虑上层的上下文内容。要知道上下文意味着什么，为什么传统的包过滤防火墙在这方面有局限性，必须了解一些背景知识。大多数运行在 TCP 协议之上的标准应用程序遵循一种客户机/服务器的工作模式。举例来说，在简单邮件传输协议(SMTP)里，电子邮件从一个客户系统发送到服务系统。客户系统发起一个新的邮件信息，它通常通过用户的输入来实现，服务器收到这个信息后存放到相应的客户的邮箱里。SMTP 的具体实现过程首先是在客户机和服务器之间建立一个 TCP 连接，在这个连接里，服务器端口数是 25，这个端口是专门提供给 SMTP 的服务器用的。SMTP 中客户机的端口数则是 1024 到 16 383 之间的一个。

通常，当使用 TCP 协议的应用程序创建一个同远端主机的会话时，也就建立了一个 TCP 连接，它分配给远端(服务器)应用程序的 TCP 端口是一个小于 1024 的数，面分配给本地(客户机)程序的是一个介于 1024 和16 383 之间的数。小于 1024 的那些众所周知的端口数是永久性地分配给某些特别的应用程序的(比如 25 是给 SMTP 中的服务器用的)。介于 1024 和 16 383 之间的端口数是动态的，也是暂时分配的，一旦 TCP 连接中断，分配也就不再有效。

简单包过滤防火墙必须允许所有使用这些高端口的基于 TCP 的通信通过。这就使得它容易受到未授权的用户的使用。

状态检查防火墙的包过滤器通过建立外向 TCP 连接字典，加强了处理 TCP 通信的规则。就像表 20.2 所显示的那样，每个当前建立的连接都记录在字典里。如果一个数据包的目的地是系统内部的一个介于 1024 和 16 383 之间的端口，而且它的信息与连接字典里某一条记录相符，包过滤器才允许它进入。

表 20.2　状态检查防火墙的状态表实例

源 地 址	源 端 口	目的地址	目的端口	连接状态
192.168.1.100	1030	210.9.88.29	80	已建立
192.168.1.102	1031	216.32.42.123	80	已建立
192.168.1.101	1033	173.66.32.122	25	已建立
192.168.1.106	1035	177.231.32.12	79	已建立
223.43.21.231	1990	192.168.1.6	80	已建立
219.22.123.32	2112	192.168.1.6	80	已建立
210.99.212.18	3321	192.168.1.6	80	已建立
24.102.32.23	1025	192.168.1.6	80	已建立
223.212.212	1046	192.168.1.6	80	已建立

应用级网关

应用级网关也叫做代理服务器，它在应用级的通信中扮演着一个消息传递者的角色[图 20.1(b)]。用户使用 Telnet 和 FTP 之类的 TCP/IP 应用程序时，建立了一个到网关的连接，这个网关要求用户出示将要访问的异地机器的正确名称。如果用户给出了一个有效的用户 ID 和验证信息，网关就建立一个到异地机器的应用级连接，并开始在访问者和被访问者之间传递包含着应用数据的 TCP 数据段。如果网关无法执行某个应用程序的代理码，服务就无法

执行，也不能通过防火墙发送。而且，网关可以被设置成为只能支持网络管理员所愿意接受的某些应用程序，而拒绝所用其他的服务。

应用级网关看上去要比包过滤器更加安全。它不再去试图处理 TCP/IP 层可能发生的所有情况，一一考虑它们是否应被允许通过，而是只需要去考虑一小部分那些允许进行的应用程序。而且，在应用级上进行日志管理和通信过程的审查要容易得多。

这种网关的一个主要不足是它在每次连接中有多余的处理开销。具体地说，两个终端用户通过代理取得连接，代理必须检查并转发通信中两个方向上的所有数据。

电路级网关

第三种防火墙是电路级网关[见图 20.1(c)]，它是一个独立系统或是说某项具体的功能，这项功能事实上也可以由应用级网关在某个应用中执行。电路级网关不允许一个端到端的直接 TCP 连接；它由网关建立两个 TCP 连接，一个连接网关与网络内部的 TCP 用户，一个连接网关与网络外部的 TCP 用户。连接建立之后，网关就起着一个中继的作用，将数据段从一个连接转发到另一个连接。它通过决定哪个连接被允许建立来实现其对安全性的保障。

电路级网关的具体应用是在一个系统管理员信任内部用户的环境里。配置网关，使得它在与内部用户的连接上支持应用级或代理服务，而在与外部用户的连接上支持电路级功能。这样，虽然网关在对进入内部的数据检查以发现系统禁止的操作时，仍然会有处理开销，但在处理到网络外部的数据时，则不会有此开销。

电路级网关的一个应用的例子是 SOCKS 包[KOBL92]，RFC 1928 定义了 SOCKS 版本 5。RFC 定义的 SOCKS 格式如下：

> 本协议为 TCP 和 UDP 领域的客户端/服务器应用程序提供一个框架，使得它们更为安全和便利地使用网络防火墙。从概念上看，本协议可以看做是应用层和传输层之间的一个"薄片层"，尽管它并不提供网络层网关服务，例如 ICMP 信息的转发等。

SOCKS 由以下成分组成：

- SOCKS 服务器，它运行在一个基于 UNIX 系统的防火墙上。
- SOCKS 客户库，它运行在防火墙保护着的网络内部主机上。
- 一些标准客户端程序，如 FTP 和 TELNET 的 SOCKS 版本。执行 SOCKS 协议的过程包括基于 TCP 的客户端程序的重编译或重链接，从而能在 SOCKS 库中使用合适的封装程序。

当一个基于 TCP 的客户端希望与一个只有通过防火墙才能到达的目标建立连接时(这个判断需要首先执行)，它首先必须打开一个与 SOCKS 服务系统上某个合适的 SOCKS 端口的 TCP 连接。SOCKS 服务使用的 TCP 端口数是 1080。如果请求成功，客户端与服务系统就对将要使用的验证方法进行协商，然后发送一个转发请求，这个请求必须用协商得出的方法进行验证。SOCKS 服务器评估这个请求决定是否建立相应的连接。UDP 交换也以类似的步骤来进行。一般来说，断开一个 TCP 连接，就可以使用户验证或者收发一个 UDP 段，只要 TCP 连接还断开着，UDP 段就可以一直转发下去。

堡垒主机

堡垒主机是由防火墙的管理人员所指定的某个系统，它是网络安全的一个临界点。它通

常是作为应用级网关和电路级网关的服务平台。通常,堡垒主机有如下特点:

- 堡垒主机硬件平台使用的操作系统是该操作系统的安全版本,这使得它成为一个可信系统。
- 堡垒主机上只安装网络管理员认为必需的服务。这其中包括代理应用程序,如 Telnet、DNS、FTP、SMTP 和用户验证方法。
- 每个用户在访问代理服务之前,堡垒主机要对其进行额外的验证。另外,每个代理在用户对其访问之前也可以对其进行验证。
- 各代理设置为只能支持标准应用程序指令基的一个子集。
- 各代理只允许对特殊主机的访问。这意味着只能在受保护网络内的部分主机上使用部分指令。
- 各代理通过对通信、连接以及连接持续时间的日志管理,来取得详细的审查信息。对日志的审查是发现和终结入侵者攻击的重要方法。
- 各代理模块是专为网络安全而设计的很小软件包。由于它相对比较简单,检查安全漏洞也比较容易。举例来说,一个典型的 UNIX 邮件应用程序可能包含超过 20 000 行的代码,而一个邮件代理仅有不到 1000 行。
- 堡垒主机上的代理彼此之间是独立的。如果对某个代理的操作出现问题,或者潜在的弱点被发现,那么完全可以卸载这个代理而不会影响到其他代理的使用。另一方面,如果用户需要新的服务,网络管理员可以很容易地在堡垒主机上安装所需的代理。
- 代理通常使用无盘访问而不让用户直接读取它的初始配置文件。这使得入侵者很难在堡垒主机上安装木马程序或者其他危险的文件。
- 各代理运行在堡垒主机上私有且安全的目录下,它们之间没有优先权的划分。

20.1.3 防火墙的配置

除了使用简单的系统如单一的包过滤路由器或网关(见图 20.1)的防火墙之外,还有着配置更为复杂的防火墙,事实上这类防火墙更为常用。图 20.2 给出了三种常见的防火墙配置。下面一一介绍。

在**屏蔽主机防火墙的单宿堡垒主机**结构中,防火墙包含两个系统:一个包过滤路由器和一台堡垒主机。特别地,路由器的配置如下:

1. 对来自 Internet 的通信,只允许发往堡垒主机的 IP 包通过。
2. 对来自网络内部的通信,只允许经过了堡垒主机的 IP 包通过。

堡垒主机执行着验证和代理的功能。这种配置比单一包过滤路由器或者单一的应用级网关更为安全。理由有二:第一,这种配置实现了包级和应用级的过滤,在系统安全策略允许的范畴内又有着相当的灵活性;第二,入侵者必须攻破两个独立的系统才有可能威胁到内部网络的安全。

这种配置较为灵活,可以提供直接的 Internet 访问。一个例子是,内部网络可能有一个如 Web 服务器之类的公共信息服务器,在这个服务器上,高级的安全不是必需的,这样,就可以将路由器配置为允许信息服务器与 Internet 之间的直接通信。

然而,从这种配置我们可以清楚地看到,如果包过滤路由器被攻破,那么通信就可以越过路由器在 Internet 和内部网络的其他主机之间直接进行。**屏蔽主机防火墙双宿堡垒主机**结构在物理上防止了这种安全漏洞的产生[见图 20.2(b)]。第一种配置所带来的双重安全性的好处在这

种配置里依然存在。而且，信息服务器或者其他的主机在安全策略允许的范围内都可以和路由器直接通信。

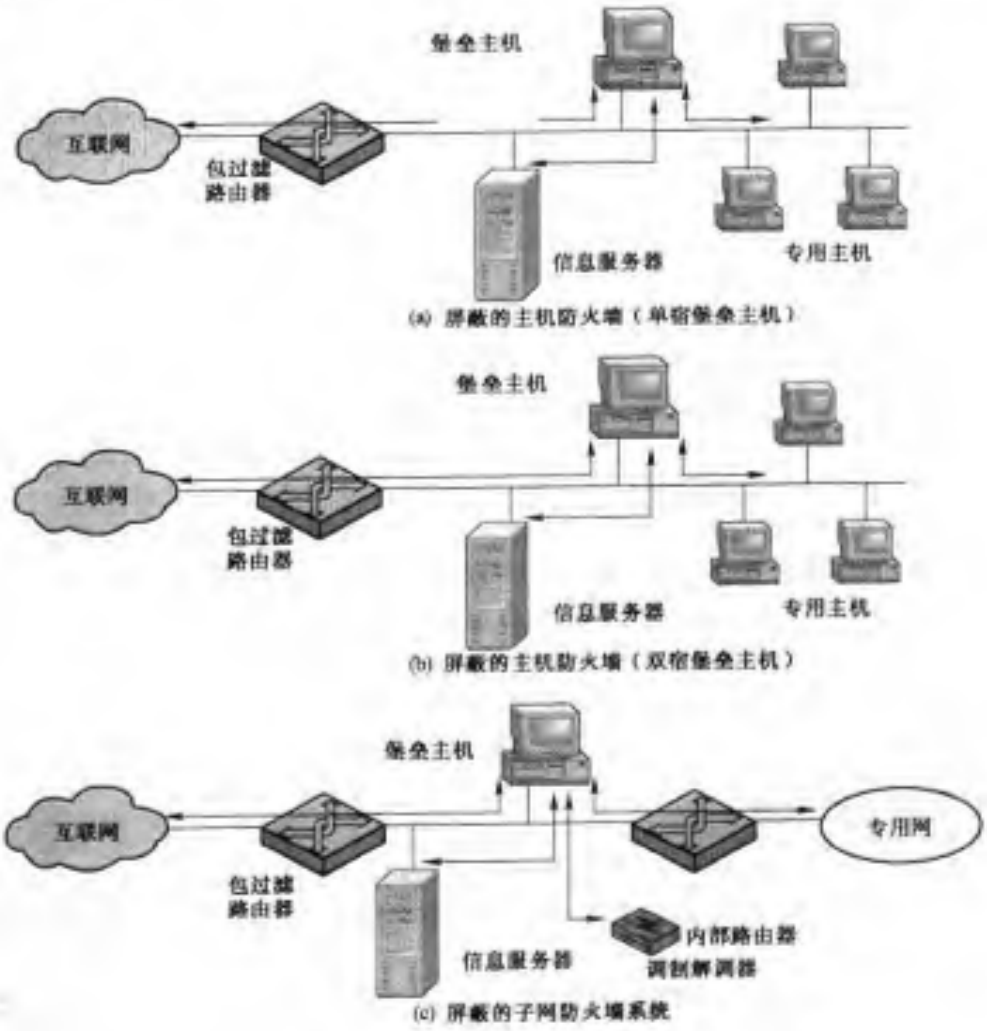


图 20.2 防火墙配置

如图 20.2(c)所示，屏蔽子网防火墙是我们所探讨的配置里最为安全的一种。在这种配置中，使用了两个包过滤路由器，一个在堡垒主机和 Internet 之间，一个在堡垒主机和内部网络之间。这种配置创造出一个独立的子网，子网可能只包括堡垒主机，也可能还包括一台或者更多的信息服务器以及为了满足拨入功能而配置的调制解调器。在这里，Internet 和内部网络都有权访问子网里的主机，但是要通过子网的通信则被阻塞。这种配置有如下优点：

- 有三层防御来抵御入侵者。
- 在 Internet 上通过外部路由器只能得知子网的存在；因此，内部网络对于 Internet 而言是不可见的。
- 类似地，从内部网络通过内部路由器也只能得知子网的存在；因此，网络内部的系统无法构造直接到 Internet 的路由。

20.2 可信系统

可信系统是加强系统防御入侵者和恶意程序能力的方法之一。这一部分对这个问题做一个概述。我们首先考察数据访问控制的一些基本概念。

20.2.1 数据访问控制

成功登录之后,用户就获得了对主机和应用程序的访问权。对于那些数据库里存有敏感数据的主机而言,这样是远远不够的。通过用户访问控制程序,系统可以识别用户。对于每个用户,还必须配置一个专门的文档,用来规定这个用户被允许进行的操作和可以访问的文件。这样,操作系统就可以通过管理用户文档来控制用户对数据的访问。数据库管理系统则需要对特定记录或者一部分记录进行访问控制。举例说,也许管理层的每一个人都有权得到一份公司全体职员的名单,但却只有少数人有权对职员的工资信息进行访问。这决不仅仅是个细节问题。不然的话,操作系统可能允许一个用户访问文件和应用程序,而不对其进行进一步的安全检查,数据库管理系统则需要对每个用户的访问请求进行判断。判断的取得依赖于用户的身份,依赖于将要被访问的数据,甚至依赖于已经泄漏给这个用户的那部分数据。

数据库管理系统或文件系统通常以控制矩阵[图 20.3(a)]的形式来描述访问控制。这种模式有如下几个要素:

- **主体**:一个有权访问客体的实体。通常主体的概念等同于进程。任何得到对客体访问权的用户或应用程序,都会以进程的形式来存在。
- **客体**:需要对其进行访问控制的实体。例如文件、文件的某一部分、程序、内存段。
- **访问权**:主体对客体进行访问的具体形式。例如读、写和执行。

矩阵的一条轴包含了所有被认证过的有权进行数据访问的主体。通常,这份名单可能包括单个用户和用户群体,另外一条轴则列出了所有可以被访问的客体。从最大程度的细化来看,对象可以是单个的数据域。更为庞大的数据群体,比如记录、文件,甚至整个数据库,也可以是矩阵的客体。矩阵的每一个元素指明了某个主体对相应的客体所拥有的访问权。

事实上,访问矩阵通常是稀疏矩阵,通常我们以两种方式分解并使用它。矩阵可以按列来分解,生成所谓**访问控制序列**[见图 20.3(b)]。这样,对每个客体而言,访问控制序列列出了它的用户和他们所允许的访问方式。访问控制序列可以包含一个默认的公共实体。这样,那些没有在矩阵中标明的用户就可以拥有某些权力,对客体进行一些默认的操作。名单的元素可以是单个用户,也可以是用户的群体。

将矩阵按行分解就得到**能力票据**[见图 20.3(c)]。能力票据标明了一个用户所拥有的所有授权的客体和操作。每个用户都拥有一定数量的票据,在经过批准后可以将其借给或者给予其他用户。也正是因为票据可以在系统内部流通,这就提出了比访问控制名单更高的安全性问题,特别是票据的不可伪造性。解决这一问题的方法是让操作系统替用户来保管这些票据,把它们放在用户无法访问的内存段中。

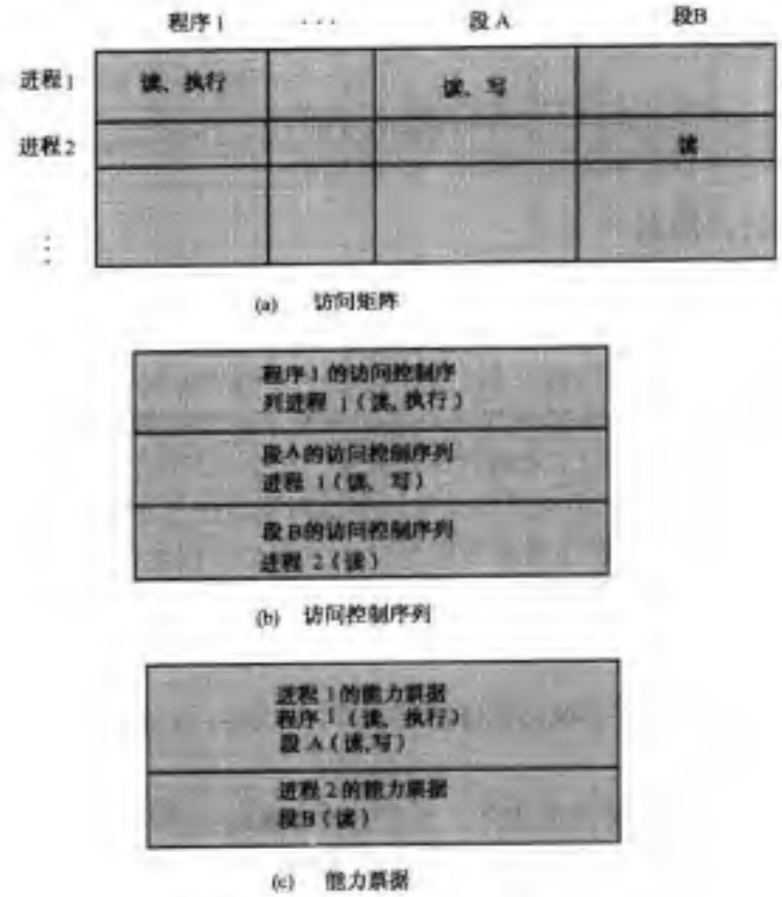


图 20.3 存取控制结构

20.2.2 可信系统的概念

到现在为止，我们主要讨论的是如何防止一个给定的用户对于给定的信息或目标的主动或被动的攻击。一个稍有不同但应用性很强的要求是在安全级的基础上来保护数据。通常军方使用这种方法，他们将信息划分为不保密的(U)、机密的(C)、秘密的(S)、绝密的(TS)和最高机密的。这种观念同样应用在其他领域，在那里信息可能被划分为一些大类，用户也被相应地划分，他们对与他们相对应的信息类有访问权。例如，需要最大程度安全的信息是公司战略计划的文档和数据，只有公司的高级官员和他的参谋机构有权访问；往下去的可能是敏感的金融和个人数据，只有管理人员、公司官员之类的人才可以访问；依此类推地往下划分各种信息。

一旦定义了数据的各种安全分层，就必须严格执行**分级安全策略**。该策略的一个基本要求是高级数据的主体不能将数据传给低级或非可比级的主体，除非这种数据流动是准确按照授权用户的要求进行的。出于应用的目的，这种要求由两个部分来规定。一个分级安全系统必须做到：

- **禁止向上读**：主体只能读与之同级或比它低的安全级的数据。这叫做**简单安全特性**。

- **禁止向下写**：主体只能在同级或比它高的安全级上写数据。这叫做 * **特性**。[①]

这两条规则如果正确执行，就可以实现分级安全。对一个数据处理系统而言，它所采用的方法事实上也是许多研究和开发的对象，这种方法基于参考监控器概念。图 20.4 描绘了这种方法。参考监控器是硬件和操作系统上的一个控制元，它根据主体和客体的安全参数来调控主体对客体的访问。参考监控器有权访问安全内核数据库，该数据库列出了每个主体的访问优先级（安全划分）和每个客体的保护属性（保密分级）。参考监控器执行上面列出的安全规则（禁止向上读，禁止向下写），并有如下特性：

- **完全性**：安全规则作用于每一次访问，而不仅仅作用于个别情况，如打开了一个文件。
- **隔离性**：在未授权的情况下，无法修改参考监控器和数据库。
- **验证性**：参考监控器的正确性一定是可以证明的。也就是说，一定可以从数学的证明：参考监控器执行了安全规则，从而满足了完全性和隔离性。

这些都是硬性要求。完全性要求意味着对内存和硬盘的每一次访问都必须受到调控。纯软件实现需要太高的资源开销，因此必须部分地依赖硬件。隔离性要求意味着无论一个人侵者多么聪明，都无法改变参考监控器的逻辑结构和安全核心数据库的内容。而对数学证明的要求对常规用途的计算机来说是十分困难的。能够给出这样的证明的系统被认为是**可信系统**。

图 20.4 所描绘的基本内容是一份审计文件。重要的安全事件，例如察觉对安全的侵害和授权对安全核心数据库的更改，都存储在这份文件里。

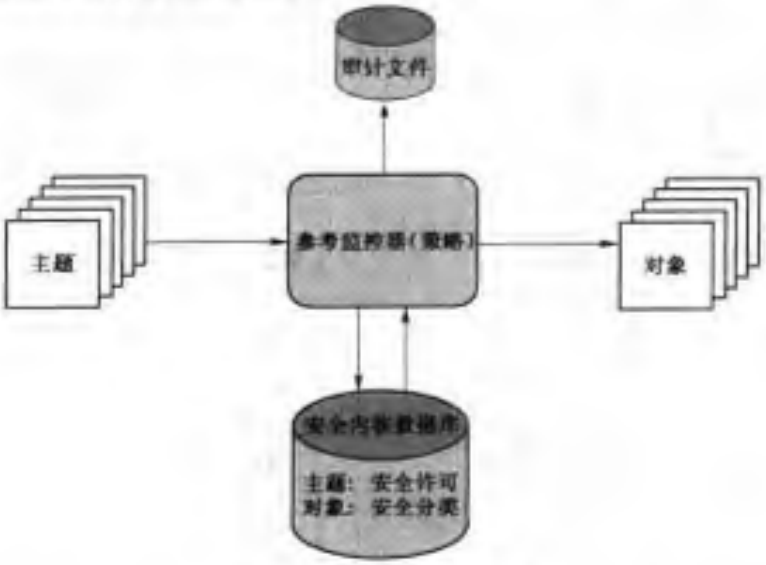


图 20.4　参考监控器的概念

为了满足自身的需求，也为了为公众提供服务，美国国防部 1981 年在国家安全局以推广可信计算机系统的应用为目的建立了计算机安全中心。这个目的通过中心的商业产品评估程序得到了实现。事实上，中心试图评估那些商业上可行的产品是否可以满足前面所描述的安全要求。

---

① 这里的 * 号不代表任何含意。最初写这个模型的报告时，没有给这一特性取一个合适的名字。在草稿中写适这个星号是为了使文本编辑器能够迅速发现并用一个合适的名字来代替它。由于没有命名，所以报告发表时用了星号。

中心将评估过的产品按照他们所提供的安全性能的不同来分类。评估结果是国防部所需要的，但它同时也公布于众，供公众使用。因此，它可以作为用户在选用商品时的一个参考。

20.2.3 对特洛伊木马的防御

防御特洛伊木马攻击的方法之一是使用可信操作系统。图 20.5 给出了一个示例。在这个例子里，有人对访问控制序列这一大多数文件管理系统和操作系统都使用的标准安全机制使用了特洛伊木马。用户 Bob 使用了一个程序，这个程序含有一份带有敏感字符串"CPE170KS"的文件。这份文件只有代表 Bob 的执行程序才有权读/写它。也就是说，只有属于 Bob 的进程有权访问这份文件。

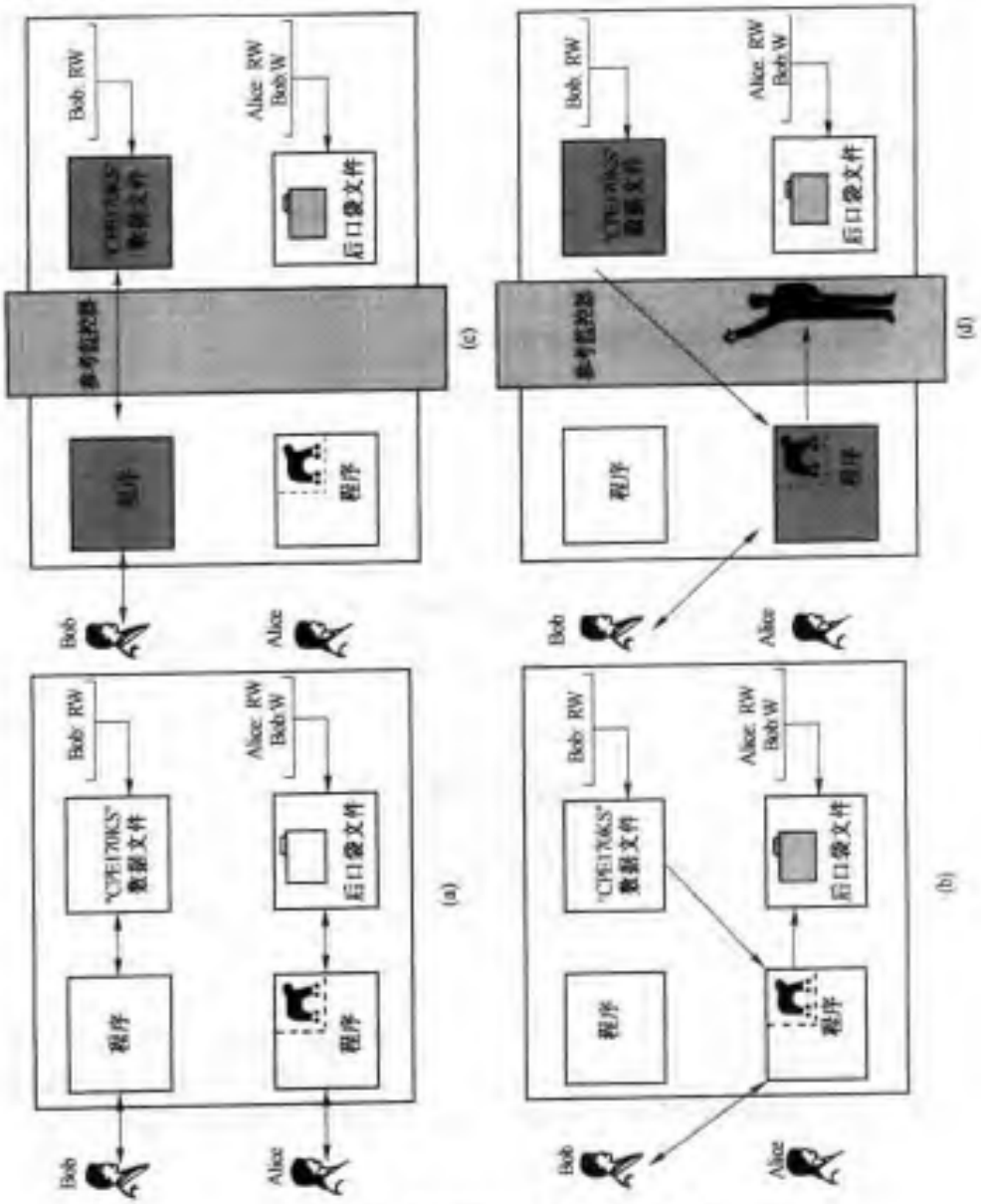


图 20.5 特洛伊木马和安全操作系统

当一个恶意用户 Alice 获得了对系统的合法访问权，并且安装了一个木马程序和一份私人文件(在攻击中叫做"后口袋")时，木马攻击随之开始。Alice 给她自己对这份文件的读/写权，同时给 Bob 只写权[见图 20.5(a)]。现在 Alice 诱使 Bob 调用木马程序，例如通过宣扬这个程序很有用处。当程序探知它正由 Bob 调用，就从前面提到的 Bob 的文件中读敏感字符串并拷贝到 Alice 的"后口袋"文件里[见图 20.5(b)]。无论是读操作还是写操作，都在访问控制列表所允许的范围内。Alice 只能在稍后的时间里从 Bob 的文件中获得字符串的值。

现在来考虑此种情形下安全操作系统的作用[见图 20.5(c)]。安全分级按照某些标准来划分主体，该标准可以是按照用户访问计算机时使用的终端，也可以是用户的 ID。在这里，有两个安全级，敏感级和公共级，敏感级高于公共级。属于 Bob 和 Bob 的数据文件的进程划入敏感级。Alice 的文件和进程则被限制在公共级。如果 Bob 调用了木马程序[见图 20.5(d)]，程序就获得了 Bob 的安全级。因此，根据简单安全特性，它就可以读敏感字符串。当程序试图存储字符串到一个公共级的文件(后口袋文件)中去时，* 特性起了作用，这个请求为参考监控器所不允许。因此，试图对后口袋文件的写操作被拒绝，尽管访问控制列表允许：在这里，安全策略要比访问控制列表的优先权高。

20.3 推荐读物和网址

防火墙方面的经典读物[CHAP95]至今仍值得一读。另一本经典是最近已经再版的[CHES00]。[LODI98]、[OPPL97]和[BELL94]也是这方面不错的论文。[WACK02]是对防火墙技术和策略的优秀总结。[GASS88]对可信计算机系统做了全面的研究。[PFLE97]和[GOLL99]也是如此。

BELL94 Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

CHAP95 Chapman, D., and Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 1995.

CHES00 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2000.

GASS88 Gasser, M. *Building a Secure Computer System*. New York: Van Nostrand Reinhold, 1988.

GOLL99 Gollmann, D. *Computer Security*. New York: Wiley, 1999.

LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.

OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.

PELE97 Pfleeger, C. *Security in Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.

WACK02 Wack, J.; Cutler, K.; and Pole, J. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41, January 2002.


推荐网址：

- **Firewall.com**：有许多关于防火墙参考资料和软件资源的链接。

20.4 关键术语、思考题和习题

20.4.1 关键术语

访问控制列表(ACL)	能力票据	包过滤路由器
访问矩阵	电路级网关	参考监控器
访问权限	防火墙	状态检查防火墙
应用级网关	分级安全	主体
堡垒主机	客体	可信系统

20.4.2 思考题

20.1 列出设计防火墙的三个目标。
20.2 试述防火墙通过哪四项技术来控制访问，执行安全策略。
20.3 典型的包过滤路由器使用哪些信息？
20.4 包过滤路由器有哪些不足？
20.5 包过滤路由器和状态检查防火墙有何不同之处？
20.6 什么是应用级网关？
20.7 什么是电路级网关？
20.8 图 20.2 的三种配置之间有何不同？
20.9 在访问控制的上下文中，主体和客体有何不同？
20.10 访问控制序列和能力票据有何不同？
20.11 参考监控器执行哪两条规则？
20.12 参考监控器应具备那些特性？

20.4.3 习题

20.1 对一个分级安全系统而言，禁止向上读规则的必要性是显而易见的，那么禁向下写的重要性何在？
20.2 在图 20.5 中，特洛伊木马的稍后读写链(copy-and-observe-later chain)是断开的。攻击者 Alice 还可以从两个角度来攻击：Alice 将“后口袋”文件划为敏感级，并试图直接读取字符串。这时，参考监控器还能抵御攻击吗？

附录 A　标准和标准化组织

“标准”是本书经常提到的一个非常重要的概念。本附录提供了一些有关标准特性和与之关联的背景知识，并介绍了几个建立网络和通信标准的主要机构。

A.1　标准的重要性

长期以来，在通信行业，标准被认为需要用来统一通信设备的物理、电气和过程特性。在过去，这种观点并没有被计算机行业所接受。虽然通信设备商们公认它们的设备和其他供应商的设备留有接口，并且可以相互通信，但是计算机供应商传统上企图垄断它们的客户。计算机数量的急剧增长和不同的制作过程使它处于一个不令人期望的位置。随着协议标准的不断更新发展，不同供应商的计算机必须能够互相通信，顾客们将不再接受仅满足特定协议的软件。其结果就导致了本书中所提到的标准化。

建立标准会带来很多好处，但也有一些弊端。它所带来的主要好处如下：

- 标准确保了在硬件或软件的某个领域将具有极大的市场。这将刺激大批量生产，并且通过采用大规模或超大规模生产技术，降低了成本。
- 标准允许不同供应商的产品可以相互兼容，这就给用户在对设备的选择和使用方面带来了极大的便利。

建立标准也存在如下几个弊端：

- 标准会滞缓技术的发展。为了达到一个标准，就要接受审查，在技术上也会有一些妥协，并且还要公开颁布。但如果没有这些限制，就可能发明更加高效的技术。
- 同一个事物可能有多套标准，这本不该是标准的一个弊端。但从目前情况看，它是。幸运的是，近几年各种标准建立机构开始更加紧密地合作。尽管如此，某些领域仍然存在标准互相冲突的现象。

A.2　标准和规则

本节对读者分清以下三个概念是很有帮助的：

- 自愿标准。
- 统一标准。
- 自愿标准的统一使用。

自愿标准是由这个附录中所提到的那些标准化组织制定的。这些标准是自愿使用的，并不带有强迫性。也就是说，如果制造商们认为这个标准对他们有利，他们的产品便采用这个标准，并没有规定一定要遵守。这些标准是自愿的，也意味着这些标准是由一些自愿者制订的。

负责这个工作的标准设立机构也不会给这些志愿者的工作付任何报酬。这些志愿者一般都是一些感兴趣的团体组织的成员，譬如工厂和政府办事处。

自愿标准的出现是因为这是建立在广泛意见的基础之上的，并且顾客需要标准使供应商们的产品得到统一。

相反，统一标准是由政府部门特定机构为了满足某些社会需求，譬如经济、卫生和安全产品而专门制定的。这些标准具有强制效力并且该标准涉及的有关部门必须遵守该标准。关于这方面的一个熟悉的例子就是防火和卫生标准。但这些规则能适用于大部分产品，包括计算机和通信有关的一些产品。例如，美国联邦通信委员会建立的电磁辐射标准。

一个比较新的、至少是目前比较流行的现象是自愿标准的统一。一个典型的例子就是官方购买的产品都符合自愿标准的规定。

这种做法有如下几点好处：

- 减少政府办事机构制定标准的负担。
- 鼓励政府和标准化组织共同制定具有广泛应用的标准。
- 减少供应商需要满足的标准的数量。

A.3 互联网标准和国际互联网协会

一些构成 TCP/IP 协议组的协议已经标准化，有一部分还在标准化过程中。经过一致同意，一个叫做国际互联网协会的组织负责这些协议的开发和颁布。国际互联网协会是一个监督互联网开发、标准化组织和工作组的专业成员机构。

本节简要描述了 TCP/IP 协议组的标准化过程。

A.3.1 互联网机构与 RFC 的发布

国际互联网协会是互联网设计、施工和管理的协调委员会。其涉及的领域包括互联网自身的操作与互操作平台。国际互联网协会属下的以下三个机构负责标准的开发与颁布等实际工作：

- **互联网架构委员会(IAB)**：负责定义互联网的整体架构，并给 IETF 提供指导和工作方向。
- **互联网工程任务组(IETF)**：互联网的协议施工与开发。
- **互联网工程指导小组(IESG)**：负责 IETF 的技术管理和互联网标准化过程的技术管理。

IETF 许可的工作组实施新标准和协议的实际开发。工作组的每个成员都是自愿参加的，任何有兴趣的团体都可以参加。在说明书的设计过程中，工作组将会设计一份文档作为放在 IETF“互联网草案集”筹建目录中的互联网草案。这份文档作为互联网草案保留 6 个月，这段时间中，有兴趣的团体可以审查和评论这份草案。其间，IESG 可能将其作为请求文档(RFC)发布。如果这份草案在 6 个月之内没有达到成为 RFC 的要求，它将从目录中删除，工作组可以最后发布这份草案的修正版本。

IETF 在 IESG 的批准下负责发布 RFC，RFC 是互联网研究和开发机构的工作记录。RFC 可能是与计算机通信紧密相关的关键议题，也可能是某些关于标准详述的会议报告。

IETF 的工作分为八个领域，每个领域都有一个机构负责人(AD)，每个机构由若干的工作组构成。表 A.1 列出了 IETF 的各个机构以及它们的主要力量和工作重点。

表 A.1　IETF 的领域

IETF 机构	任　务	工作组举例
常规	IETF 进度和流程	策略框架 互联网标准化机构的进程
应用	互联网应用	Web 相关协议(HTTP) EDI 与互联网集成 LDAP
互联网	互联网基础设施	IPv6 PPP 扩展
操作与管理	网络操作的标准与定义	SNMPv3 远程网络管理
路由	路由信息的协议和管理	组播路由 OSPF QoS 路由
安全	安全协议及技术	Kerberos IPSec X.509 S/MIME TLS
传输	传输层协议	区分服务 IP 电话 NFS RSVP
用户服务	提高信息质量的方法，用户可获取该信息	互联网的可靠使用 用户服务 FYI 文档

A.3.2　标准化过程

在 IETF 的推荐下，RFC 被 IESG 提升为标准。规范要成为标准，需满足以下几个条件：

- 可靠并且容易理解。
- 技术先进。
- 在实际应用中具有多重独立性和可操作性。
- 能够赢得广泛的支持。
- 对互联网的部分或所有领域具有强可用性。

这些要求与 ITU 国际标准的主要区别在于这里强调其实际可操作性。

图 A.1 左边给出的一组步骤称为标准轨迹，它是一个规范成为标准所经过的几个阶段。RFC 2026 定义了这个过程。这些步骤包括不断地详细检查与测试。在每个阶段，IETF 必须做出关于本阶段协议所做修改的报告，并且这份报告必须得到 IESG 的认可。整个过程从 IESG 批准以建议标准级 RFC 发布其互联网草案开始。

图中的白色方框代表临时状态，这些是一段时间内必须结束的。然而，一份文档必须以建议标准保留六个月，而草案标准则要保持四个月，以提供足够时间进行修改和评论。灰色方框代表一个长期的状态，这个状态可能延续几年。

一份文档被提升为草案标准，必须至少经过两个独立的互操作性实验，以通过足够的操作测试。

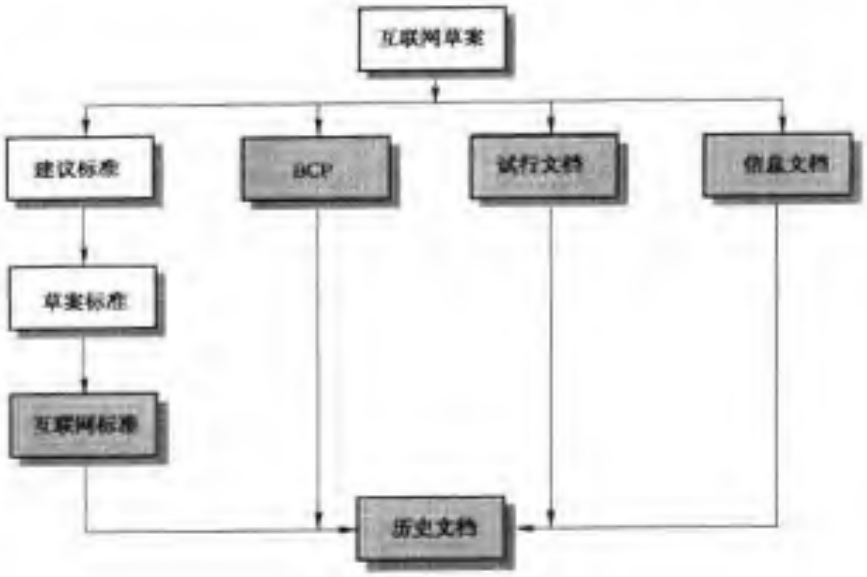


图 A.1 RFC 的发布过程

在经过重要的实现和互操作实验测试后，一份文档就被提升为标准协议。这时，该文档将分配一个类似于 RFC 编号的 STD 编号。

最后，当一份草案过于陈旧作废之后，它就被标注为历史文档。

A.3.3 互联网标准种类

所有的互联网标准被划分为两种：

- **技术规范(TS)**：技术规格定义了协议、服务、过程、会议或格式。大部分标准都属于技术规范。
- **适用性声明(AS)**：适用性声明规定了技术规范在什么条件下如何支持特定的互联网性能。AS 规定了相应性能所涉及到的 TS 或 TS 子集的一些参数的值和范围。

A.3.4 其他 RFC

很多 RFC 的原本目的并不是为了专门制定成互联网标准而开发的。许多 RFC 是一些团体对规则描述深思熟虑后的结果，或实施某些操作或 IETF 工作进程的最好方法。这种 RFC 称做 BCP。BCP 的批准与建议标准的批准流程是一样的，不像标准轨迹式文档，BCP 的提升不需要三个步骤。一个 BCP 从互联网草案到批准只需要一步。

一些没有条件进行标准化的协议或其他规范将作为试行 RFC 出版。经过一段修改工作之后，该规范可能被重新提交。如果该规范可靠，能够满足设计需求，并且容易理解，经过特定机构复查后能够得到特定机构的认可，该 RFC 就被提升为建议标准。

最后，汇总出版一份关于国际互联网协会一般资料的资料。

A.4 美国标准与技术研究所

美国标准与技术研究所(NIST),作为美国商务部的一部分,专门为美国政府部门和代理机构发布标准和指导方针。这些标准和指导方针作为联邦信息处理标准(FIPS)发布。在联邦政府有特别需求时,譬如安全和互操作性方面以及没有可以接受的工业标准或解决方案时,美国标准与技术研究所开发 FIPS。

- 为了便于公众审查和评论,NIST 在联邦注册局公布建议的 FIPS。同时,建议 FIPS 也会在 NIST 的网站公告,如果需要的话,建议 FIPS 的文字和有关说明也会在 NIST 的网站发布。
- 有 90 天时间用来审查和对提交给 NIST 的建议 FIPS 进行评论。联邦注册局和其他公告会指定注释提交到 NIST 的具体日期。
- NIST 会审查来自联邦注册局或其他的文档注释,以决定其是否有必要修改成建议 FIPS。
- 一份详细的原因说明文档将会用来说明文档、分析注释和解释是否需要修改,如果不需要修改,也要给出其原因。
- NIST 提交建议 FIPS、详细原因以及关于是否采纳或适合政府使用的推荐信给商务部长批准。
- 商务部长在联邦注册局和 NIST 的网站公布 FIPS 得到批准。

尽管 NIST 标准开发用于美国政府,但其中许多已在世界广泛使用。AES 和 DES 就是主要的示例。

A.5 本书引用的标准和说明

ANSI 标准

标准号	标题	日期
X.917	财政协会的密钥管理	1995

互联网 RFC

标准号	标题	日期
RFC 822	ARPA 互联网文本消息的格式标准	1982
RFC 1510	Kerberos 网络认证服务(V5)	1993
RFC 1636	互联网体系结构的安全	1994
RFC 1750	安全的自由建议	1994
RFC 1928	SOCKS 协议版本 5	1996
RFC 2026	互联网的标准过程	1996
RFC 2040	RC5, RC5-CBC, RC5-CBC-Pad, RC5-CTS 算法	1996
RFC 2045	MIME 第一部分:互联网消息体格式	1996
RFC 2046	MIME 第二部分:媒体类型	1996
RFC 2047	MIME 第三部分:非 ASCII 文本的消息头扩展	1996
RFC 2048	MIME 第四部分:注册过程	1996

（续表）

标准号	标题	日期
RFC 2049	MIME 第五部分：一致准则和例程	1996
RFC 2104	HMAC：消息认证的散列	1997
RFC 2119	RFC 中用于指示需求级别的关键词	1997
RFC 2246	TLS 协议	1999
RFC 2401	IP 的安全体系结构	1998
RFC 2402	IP 认证头（AH）	1998
RFC 2406	IP 封装安全载荷（ESP）	1998
RFC 2408	互联网安全关联和密钥管理协议（ISAKMP）	1998
RFC 2459	X.509 公钥基础设施证书和 CRL 轮廓	1999
RFC 2630	密码消息语法	1999
RFC 2632	S/MIME 版本 3 证书处理	1999
RFC 2633	S/MIME 版本 3 消息规范	1999
RFC 2828	互联网安全术语表	2000
RFC 3156	基于 OpenPGP 的 MIME 安全性	2001
RFC 3174	US SHA-1	2001

ITU-T 建议

标准号	标题	日期
X.509	目录：公钥和属性证书框架	2000
X.800	开放系统互联的安全体系结构	1991

NIST FIPS

标准号	标题	日期
FIPS 46-3	数据加密标准（DES）	1999
FIPS 81	DES 的操作模式	1980
FIPS 113	计算机数据认证	1985
FIPS 180-1	安全 hash 标准	1995
FIPS 180-2（草案）	安全 hash 标准	2001
FIPS 181	自动口令生成器	1993
FIPS 186-2	数字签名标准	2000
FIPS 197	高级加密标准（AES）	2001
SP 800-38A	分组密码模式的操作建议	2001

附录 B　用于密码编码学与网络安全教学的项目

很多教师都相信做项目对于加深对密码编码学和网络安全的理解是非常重要的。没有项目，学生就很难掌握一些基本的概念和弄清楚各个构件之间的相互关系。项目可以加深学生对书本介绍的概念的理解，使学生对加密算法和协议的工作原理有个非常直观的评价和了解，这可以激励学生并让他们相信自己不仅能够掌握而且能够实现与安全有关的技术细节。

在本书中，我已尽可能清楚地介绍了密码编码学和信息安全方面的原理，并且提供约 170 个课外问题，以加深对这些原理的理解。然而，很多教师或许希望用做项目的方法来深入这些知识。本附录在这方面提供了一些指导，并描述一些教师手册中有用的资料。这些有用的资料包括三个方面的项目：

- 研究项目。
- 编程项目。
- 阅读/报告项目。

B.1　研究项目

加深对课程中介绍的基本概念的理解和给学生教授研究技巧的最有效的方法就是实施一个研究项目。这种项目可以是文献研究，也可以是对供应商产品、研究室活动和标准化成果等进行网上搜索。项目可以分配给一个组，比较小的项目也可以分配给个人。无论如何，最好在一个学期的早期做好各种项目的建议，给教师们足够时间去评价相应的主题和其所取得的相应成绩。学生们的科研报告应该包括以下几个部分：

- 提议的格式。
- 最终报告的格式。
- 中期和最终期限的进度安排表。
- 可能项目的主题列表。

学生们可以从主题列表中选择一个或者自己设计一个相应的项目。教师的工作就是做好一份包括提议和最终报告的推荐格式和一份具有 15 个相应研究性项目的主题列表。

B.2　编程项目

编程项目是一个有用的教学工具。像独立编制新的安全工具这种编程项目具有好几个吸引点：

1. 教师能够从多个密码编码学和网络安全原理中选择布置项目。
2. 学生可以在任何一台机器上用任何语言进行编程；他们可以随意选择平台和语言。
3. 教师不需要为独立的项目下载、安装和配置特定的基础环境。

项目的大小具有灵活性。比较大的项目可以给学生们更多的成就感,但是能力不够或者组织性不强的学生可能被落在后面。大项目通常会让好学生抽出更多的时间和精力在上面。比较小的项目具有更高的代码效率,由于大多数项目都可以分配下去,学生们所涉及的领域更广。

再次,和研究项目一样,学生们会首次学会提交报告。学生们的实验报告同样应该包括B.1节所提到的那些要素。教师的工作就是准备12个相应的编程项目。

下面这些人提供了教师手册中的研究和编程项目:哥伦比亚大学的Henning Schulzrinne,俄勒冈大学的Cetin Kaya Koc,可信信息系统公司和华盛顿大学的David M. Balenson。

B.3 阅读/报告作业

另外一个可以加深学生对课堂知识理解和教授学生研究经验的有效方法是让学生们通过阅读和分析来提交论文。教师的工作就是做好建议论文列表,每章给出一到两份。所有准备的论文可以来自于互联网,也可以选自优秀大学的科技图书馆。教师的工作还包括做一份推荐的关键词。

术 语 表

术语表中用星号标识的术语摘自《计算机安全辞典》[NIST91]。

非对称加密(Asymmetric Encryption)

一种密码体制,其中加密和解密用不同的密钥,分别是公钥和私钥,非对称加密也称为公钥加密。

认证(Authentication *)

用于验证所传送数据,特别是消息完整性的过程。

认证符(Authenticator)

为使接收者能验证所收到消息的真实性而附加在消息后的额外信息。认证符可在功能上与消息本身的内容无关(如临时交互号或发方标识),也可以是消息内容的函数(如 hash 值或密码校验和)。

雪崩效应(Avalanche Effect)

加密算法的一种特征,它指明文或密钥的少量变化会引起密文很大的变化。对于 hash 码,雪崩效应是指少量消息位的变化会引起消息摘要的许多位发生变化。

细菌(Bacteria)

通过复制自身来消耗系统资源的程序。

分组链接(Block Chaining)

对称分组加密使用的一种方法,它使得输出分组不仅依赖于当前明文输入分组和密钥,而且还依赖于以前的输入及(或)输出。分组链接使得两个相同的明文输入分组可以产生不同的密文分组,从而增加密码分析的难度。

分组密码(Block Cipher)

将明文分组(通常为 64 位)转换为相同长度的密文分组的一种对称加密算法。

密码(Cipher)

加密和解密的算法。密码为了掩藏信息的内容,将明文信息替换为另一信息。一般地,这种转换规则受有关秘密钥控制。

密文(Ciphertext)

加密算法的输出;消息或数据的已加密形式。

编码,代码(Code)

将信息(如字母、单词、短语)替换为另一信息的一种规则,信息的顺序可以改变。一般地,编码并不是为了掩藏信息内容。例如,ASCII 字符编码(每个字符用 7 位表示)和频移键控(每个二进制值由一个特定的频率表示)。

计算上安全(Computationally Secure)

对一种密码的破译因为时间太长及(或)代价太高而不可行,则称该密码是计算上安全的。

混淆(Confusion)

使得密文的统计特性和密钥值之间具有复杂关系的一种密码编码技术。可使用依赖于密钥和输入的复杂的置乱算法以获得这种密码特性。

传统加密(Conventional Encryption)

对称加密。

隐信道(Covert Channel)

以一种违反通信系统设计者意愿的方式进行信息传输的通信信道。

密码分析学(Cryptanalysis)

密码学分支之一,涉及破译密码以获得信息或伪造加密信息以期获得认可。

密码校验和(Cryptographic Checksum)

一种认证符,它是用于认证的数据和秘密钥的密码函数,也称为消息认证码(MAC)。

密码编码学(Cryptography)

密码学分支之一,是为了保证消息的保密性及(或)真实性而设计的加密和解密算法。

密码学(Cryptology)

研究安全通信的科学,包括密码编码学和密码分析学。

解密(Decryption)

将加密的文本或数据(密文)转换成明文。

差分密码分析(Differential Cryptanalysis)

选择具有特定异或差分模式的明文进行加密,利用所得密文的差分模式来确定加密密钥的密码分析方法。

扩散(Diffusion)

为了消除明文的统计特征而使明文的每一位会影响密文的许多位的一种密码技术。

数字签名(Digital Signature)

一种认证方法。该方法使发送者可在原消息后附加一种编码作为其签名。这个签名对消息提供了认证并保证了其完整性。

双字母组合(Digram)

一个双字母序列。在英语和其他语言中,明文中不同的双字母组合的频率可以用于一些密码的密码分析中,也称二合字母。

自主访问控制(Discretionary Access Control *)

根据主体或主体所属的组的标识来决定对客体访问的控制方法。自主访问控制的意思是,有某种访问权的主体能够把这种访问权授予(可能是间接的)其他的主体(除非被强制访问控制限制)。

加密（Encryption）

用一种可逆的转换将明文或数据转换成不可懂的密文数据，该转换可以是一个转换表或一个算法。

Hash 函数（Hash Function）

将不同长度的分组数据或消息转换成定长的 hash 码的函数。在数据保护中，该函数可以提供对数据或消息的认证。也称为消息摘要。

蜜罐（Honeypot）

一种用来引诱潜在的攻击者远离至关重要的系统的诱骗系统，它是一种入侵检测技术。

初始矢量（Initialization Vector）

在使用分组链接加密技术时，在开始加密多个明文分组时使用的随机数据分组。初始矢量可以阻止已知明文攻击。

入侵者（Intruder）

获取或试图获取对一个计算机系统的非授权访问的人，或获取在该系统上的非授权权限的人。

Kerberos

Athena 项目的代码认证服务的名称。

密钥分配中心（Key Distribution Center）

建立临时会话时，为收发双方提供密钥的系统。每个会话密钥传送时，用一个密钥分配中心与接收方共享的密钥对其进行加密。

逻辑炸弹（Logic Bomb）

故意装入计算机系统中且在一定条件下才发作的程序。一旦检测到某个条件满足，则执行一些非授权的动作。

强制访问控制（Mandatory Access Control）

一种基于分配给用户、文件和其他客体的安全属性而对访问客体进行限制的方法。强制访问控制的意思是用户或他们的程序不能更改其安全属性。

主密钥（Master Key）

密钥分配中心与通信双方为保护会话密钥的传输而使用的长寿命密钥。一般地，主密钥的分配不是用密码学方法来进行的。主密钥也称为加密密钥的密钥。

消息认证码（Message Authentication Code(MAC)）

密码校验和。

消息摘要（Message Digest）

hash 函数值。

多级安全（Multilevel Security）

对访问不同安全级别数据进行访问控制的能力。

多次加密（Multiple Encryption）

反复使用一个加密函数，每次使用不同的密钥生成更复杂的密文。

临时交互号（Nonce）

只用一次的标识符或数字。

单向函数（One-Way Function）

在其定义域给定任何 x，可以很容易的算出 $f(x)$，但求逆在计算上是不可行的。

口令（Password*）

用来认证实体的字符序列。口令和相应的用户标识认证正确后，用户可以使用授予该用户标识的权限。

明文（Plaintext）

加密函数的输入或解密函数的输出。

私钥（Private Key）

非对称加密系统中的两个密钥之一。为了保证通信安全，私钥只有其拥有者知道。

伪随机数产生器（Pseudorandom Number Generator）

确定地产生一个具有明显统计随机性的数字序列的函数。

公钥（Public Key）

非对称加密系统中的两个密钥之一。公钥是公开的，与一个相应的私钥配合使用。

公钥加密（Public-Key Encryption）

非对称加密。

重放攻击（Replay Attacks）

如果密文中没有时间戳，则攻击者可通过重发截获的旧密文来冒充当前密文，收信者不能发现。这种攻击称为重播攻击或重放攻击。

RSA 算法（RSA Algorithm）

一种基于模幂运算的公钥加密算法。它是惟一一个被广泛接受且安全实用的公钥加密算法。

秘密钥（Secret Key）

对称加密系统中所用的密钥。加密双方必须共享同一个密钥，为了保护通信，这个密钥必须保密。

会话密钥（Session Key）

会话双方使用的临时加密密钥。

流密码（Stream Cipher）

一种对称加密算法，其中输出的密文由输入的明文逐位或逐字节地加密产生。

对称加密（Symmetric Encryption）

一种加密系统，其中加密和解密使用同一个密钥。也称为传统加密。

陷门（Trap-Door）

程序中暗藏的进入点，它可以绕过访问认证机制而获得访问权限。

单向陷门函数（Trap-Door One-Way Function）

计算函数值容易，但求逆函数很难，除非知道一定的特权信息。

特洛伊木马（Trojan Horse *）

一种暗藏的计算机程序，它既有明显而实用的有益功能，又有额外（暗藏）的功能，可秘密利用调用进程的合法权限对安全造成损害。

可信系统（Trusted System）

能够被验证可以实现特定的安全策略的计算机和操作系统。

绝对安全（Unconditionally Secure）

即使攻击者有无限的时间和无限的计算机资源，仍然能够确保安全。

病毒（Virus）

嵌在其他程序中的代码，将自身的一个副本插入一个或多个程序中。病毒不仅可以传播，而且可以实现一些有害的功能。

蠕虫（Worm）

可以复制自身的程序，在网络连接时将自身的副本在计算机之间传播。一旦到达一台计算机，蠕虫又开始复制并继续传播。蠕虫不仅可以传播，而且可以实现一些有害的功能。

参考文献

缩　写

ACM　Association for Computing Machinery

IEEE　Institute of Electrical and Electronics Engineers

NIST　National Institute of Standards and Technology

ADAM90 Adams, C., and Tavares, S. "Generating and Counting Binary Bent Sequences." *IEEE Transactions on Information Theory*, 1990.

ADAM94 Adams, C. "Simple and Effective Key Scheduling for Symmetric Ciphers." *Proceedings, Workshop in Selected Areas of Cryptography, SAC '94.* 1994.

AKL83 Akl, S. "Digital Signatures: A Tutorial Survey." *Computer*, February 1983.

ALVA90 Alvare, A. "How Crackers Crack Passwords or What Passwords to Avoid." *Proceedings, UNIX Security Workshop II*, August 1990.

ANDE80 Anderson, J. *Computer Security Threat Monitoring and Surveillance.* Fort Washington, PA: James P. Anderson Co., April 1980.

AXEL00 Axelsson, S. "The Base-Rate Fallacy and the Difficulty of Intrusion Detection." *ACM Transactions and Information and System Security*, August 2000.

BACE00 Bace, R. *Intrusion Detection.* Indianapolis, IN: Macmillan Technical Publishing, 2000.

BACE01 Bace, R., and Mell, P. *Intrusion Detection Systems.* NIST Special Publication SP 800-31, November 2000.

BARK91 Barker, W. *Introduction to the Analysis of the Data Encryption Standard (DES).* Laguna Hills, CA: Aegean Park Press, 1991.

BAUE88 Bauer, D., and Koblentz, M. "NIDX—An Expert System for Real-Time Network Intrusion Detection." *Proceedings, Computer Networking Symposium*, April 1988.

BELL90 Bellovin, S., and Merritt, M. "Limitations of the Kerberos Authentication System." *Computer Communications Review*, October 1990.

BELL92 Bellovin, S. "There Be Dragons." *Proceedings, UNIX Security Symposium III*, September 1992.

BELL93 Bellovin, S. "Packets Found on an Internet." *Computer Communications Review*, July 1993.

BELL94 Bellovin, S., and Cheswick, W. "Network Firewalls." *IEEE Communications Magazine*, September 1994.

BELL96a Bellare, M., Canetti, R., and Krawczyk, H. "Keying Hash Functions for Message Authentication." *Proceedings, CRYPTO '96*, August 1996; New York: Springer-Verlag. An expanded version is available at http://www.cse.ucsd. edu/users/mihir.

BELL96b Bellare, M., Canetti, R., and Krawczyk, H. "The HMAC Construction." *CryptoBytes*, Spring 1996.

BELL97 Bellare, M., and Rogaway, P. "Collision-Resistant Hashing: Towards Making UOWHF's Practical." *Proceedings, CRYPTO '97*, 1997; New York: Springer-Verlag.

BERL84 Berlekamp, E. *Algebraic Coding Theory*. Laguna Hills, CA: Aegean Park Press, 1984.

BERS92 Berson. T. "Differential Cryptanalysis Mod 2^{32} with Applications to MD5." *Proceedings, EUROCRYPT '92*, May 1992; New York: Springer-Verlag.

BETH91 Beth, T., Frisch, M., and Simmons, G. eds. *Public-Key Cryptography: State of the Art and Future Directions*. New York: Springer-Verlag, 1991.

BIHA93 Biham, E., and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard*. New York: Springer-Verlag, 1993.

BIHA00 Biham, E., and Shamir, A. "Power Analysis of the Key Scheduling of the AES Candidates" Proceedings, Second AES Candidate Conference, 24 October 2000. http://csrc.nist.gov/encryption/aes/round1/conf2/aes2conf.htm.

BLAK99 Blake, I., Seroussi, G., and Smart, N. *Elliptic Curves in Cryptography*. Cambridge: Cambridge University Press, 1999.

BLOO70 Bloom, B. "Space/time Trade-offs in Hash Coding with Allowable Errors." *Communications of the ACM*, July 1970.

BLUM86 Blum, L., Blum, M., and Shub, M. "A Simple Unpredictable Pseudo-Random Number Generator." *SIAM Journal on Computing*, No. 2, 1986.

BOER93 Boer, B., and Bosselaers, A. "Collisions for the Compression Function of MD5." *Proceedings, EUROCRYPT '93*, 1993; New York: Springer-Verlag.

BOSS96 Bosselaers, A., Govaerts, R., and Vandewlle, J. "Fast Hashing on the Pentium." *Proceedings, Crypto '96*, August 1996; New York: Springer-Verlag.

BOSS97 Bosselaers, A., Dobbertin, H., and Preneel, B. "The RIPEMD-160 Cryptographic Hash Function." *Dr. Dobb's Journal*, January 1997.

BRIG79 Bright, H., and Enison, R. "Quasi-Random Number Sequences from Long-Period TLP Generator with Remarks on Application to Cryptography." *Computing Surveys*, December 1979.

BRYA88 Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Project Athena document, February 1988. Available at http://web.mit.edu/kerberos/www/dialogue.html.

BURN97 Burn, R. *A Pathway to Number Theory*. Cambridge, England: Cambridge University Press, 1997.

CAMP92 Campbell, K., and Wiener, M. "Proof that DES is not a Group." *Proceedings, Crypto '92*, 1992; New York: Springer-Verlag.

CASS01 Cass, S. "Anatomy of Malice." *IEEE Spectrum*, November 2001.

CHAP95 Chapman, D., and Zwicky, E. *Building Internet Firewalls.* Sebastopol, CA: O'Reilly, 1995.

CHEN98 Cheng, P., et al. "A Security Architecture for the Internet Protocol." *IBM Systems Journal*, Number 1, 1998.

CHES97 Chess, D. "The Future of Viruses on the Internet." *Proceedings, Virus Bulletin International Conference*, October 1997.

CHES00 Cheswick, W., and Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker.* Reading, MA: Addison-Wesley, 2000.

COCK73 Cocks, C. *A Note on Non-Secret Encryption.* CESG Report, November 1973.

COHE94 Cohen, F. *A Short Course on Computer Viruses.* New York: Wiley, 1994.

COME00 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture.* Upper Saddle River, NJ: Prentice Hall, 2000.

COPP94 Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." *IBM Journal of Research and Development*, May 1994.

CRAN01 Crandall, R., and Pomerance, C. *Prime Numbers: A Computational Perspective.* New York: Springer-Verlag, 2001.

DAEM99 Daemen, J., and Rijmen, V. *AES Proposal: Rijndael, Version 2.* Submission to NIST, March 1999. http://csrc.nist.gov/encryption/aes.

DAEM01 Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard." *Dr. Dobb's Journal*, March 2001.

DAEM02 Daemen, J., and Rijmen, V. *The Design of Rijndael: The Wide Trail Strategy Explained.* New York, Springer-Verlag, 2000.

DAMG89 Damgard, I. "A Design Principle for Hash Functions." *Proceedings, CRYPTO '89*, 1989; New York: Springer-Verlag.

DAVI89 Davies, D., and Price, W. *Security for Computer Networks.* New York: Wiley, 1989.

DAVI93 Davies, C., and Ganesan, R. "BApasswd: A New Proactive Password Checker." *Proceedings, 16th National Computer Security Conference*, September 1993.

DAWS96 Dawson, E., and Nielsen, L. "Automated Cryptoanalysis of XOR Plaintext Strings." *Cryptologia*, April 1996.

DENN81 Denning, D. "Timestamps in Key Distribution Protocols." *Communications of the ACM*, August 1981.

DENN82 Denning, D. *Cryptography and Data Security.* Reading, MA: Addison-Wesley, 1982.

DENN83 Denning, D. "Protecting Public Keys and Signature Keys." *Computer*, February 1983.

DENN87 Denning, D. "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering*, February 1987.

DESK92 Deskins, W. *Abstract Algebra.* New York: Dover, 1992.

DIFF76a Diffie, W., and Hellman, M. "New Directions in Cryptography." *Proceedings of the AFIPS National Computer Conference*, June 1976.

DIFF76b Diffie, W., and Hellman, M. "Multiuser Cryptographic Techniques." *IEEE Transactions on Information Theory*, November 1976.

DIFF77 Diffie, W., and Hellman, M. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard." *Computer*, June 1977.

DIFF79 Diffie, W., and Hellman, M. "Privacy and Authentication: An Introduction to Cryptography." *Proceedings of the IEEE*, March 1979.

DIFF88 Diffie, W. "The First Ten Years of Public-Key Cryptography." *Proceedings of the IEEE*, May 1988. Reprinted in [SIMM92].

DOBB96a Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

DOBB96b Dobbertin, H., Bosselaers, A., and Preneel, B. "RIPEMD-160: A Strengthened Version of RIPEMD." *Proceedings, Third International Workshop on Fast Software Encryption*, 1996; New York: Springer-Verlag.

DORA99 Doraswamy, N., and Harkins, D. *IPSec.* Upper Saddle River, NJ: Prentice Hall, 1999.

DREW99 Drew, G. *Using SET for Secure Electronic Commerce.* Upper Saddle River, NJ: Prentice Hall, 1999.

EFF98 Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design.* Sebastopol, CA: O'Reilly, 1998.

ELGA85 ElGamal, T. "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." *IEEE Transactions on Information Theory*, July 1985.

ELLI70 Ellis, J. *The Possibility of Secure Non-Secret Digital Encryption.* CESG Report, January 1970.

ELLI99 Ellis, J. "The History of Non-Secret Encryption." *Cryptologia*, July 1999.

ENGE80 Enger, N., and Howerton, P. *Computer Security.* New York: Amacom, 1980.

ENGE99 Enge, A. *Elliptic Curves and Their Applications to Cryptography.* Norwell, MA: Kluwer Academic Publishers, 1999.

FEIS73 Feistel, H. "Cryptography and Computer Privacy." *Scientific American*, May 1973.

FEIS75 Feistel, H., Notz, W., and Smith, J. "Some Cryptographic Techniques for Machine-to-Machine Data Communications." *Proceedings of the IEEE*, November 1975.

FERN99 Fernandes, A. "Elliptic Curve Cryptography." *Dr. Dobb's Journal*, December 1999.

FLUH00 Fluhrer, S., and McGrew, D. "Statistical Analysis of the Alleged RC4 Key Stream Generator." *Proceedings, Fast Software Encryption 2000*, 2000.

FLUH01 Fluhrer, S., Mantin, I., and Shamir, A. "Weakness in the Key Scheduling Algorithm of RC4." *Proceedings, Workshop in Selected Areas of Cryptography*, 2001.

FORD95 Ford, W. "Advances in Public-Key Certificate Standards." *ACM SIGSAC Review*, July 1995.

FORR97 Forrest, S., Hofmeyr, S., and Somayaji, A. "Computer Immunology." *Communications of the ACM*, October 1997.

FRAN01 Frankel, S. *Demystifying the IPSec Puzzle.* Boston: Artech House, 2001.

FREE93 Freedman, D. "The Goods on Hacker Hoods." *Forbes ASAP*, 13 September 1993.

FUMY93 Fumy, S., and Landrock, P. "Principles of Key Management." *IEEE Journal on Selected Areas in Communications*, June 1993.

GARD72 Gardner, M. *Codes, Ciphers, and Secret Writing.* New York: Dover, 1972.

GARD77 Gardner, M. "A New Kind of Cipher That Would Take Millions of Years to Break." *Scientific American*, August 1977.

GARF97 Garfinkel, S., and Spafford, G. *Web Security & Commerce.* Cambridge, MA: O'Reilly and Associates, 1997.

GARR01 Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology.* Upper Saddle River, NJ: Prentice Hall, 2001.

GASS88 Gasser, M. *Building a Secure Computer System.* New York: Van Nostrand Reinhold, 1988.

GAUD00 Gaudin, S. "The Omega Files." *Network World*, June 26, 2000.

GOLL99 Gollmann, D. *Computer Security.* New York: Wiley, 1999.

GONG92 Gong, L. "A Security Risk of Depending on Synchronized Clocks." *Operating Systems Review*, January 1992.

GONG93 Gong, L. "Variations on the Themes of Message Freshness and Replay." *Proceedings, IEEE Computer Security Foundations Workshop*, June 1993.

GRAH94 Graham, R., Knuth, D., and Patashnik, O. *Concrete Mathematics: A Foundation for Computer Science.* Reading, MA: Addison-Wesley, 1994.

HAMM91 Hamming, R. *The Art of Probability for Scientists and Engineers.* Reading, MA: Addison-Wesley, 1991.

HARL01 Harley, D., Slade, R., and Gattiker, U. *Viruses Revealed.* New York: Osborne/McGraw-Hill, 2001.

HEBE92 Heberlein, L., Mukherjee, B., and Levitt, K. "Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks." *Proceedings, 15th National Computer Security Conference*, October 1992.

HELD96 Held, G. *Data and Image Compression: Tools and Techniques.* New York: Wiley, 1996.

HERS75 Herstein, I. *Topics in Algebra.* New York: Wiley, 1975.

HEVI99 Hevia, A., and Kiwi, M. "Strength of Two Data Encryption Standard Implementations Under Timing Attacks." *ACM Transactions on Information and System Security*, November 1999.

HEYS95 Heys, H., and Tavares, S. "Avalanche Characteristics of Substitution-Permutation Encryption Networks." *IEEE Transactions on Computers*, September 1995.

HONE01 The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community.* Reading, MA: Addison-Wesley, 2001.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol.* Upper Saddle River, NJ: Prentice Hall, 1998.

IANS90 I'Anson, C., and Mitchell, C. "Security Defects in CCITT Recommendation X.509—The Directory Authentication Framework." *Computer Communications Review*, April 1990.

ILGU93 Ilgun, K. "USTAT: A Real-Time Intrusion Detection System for UNIX." *Proceedings, 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1993.

JAIN91 Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York: Wiley, 1991.

JAVI91 Javitz, H., and Valdes, A. "The SRI IDES Statistical Anomaly Detector." *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

JONE82 Jones, R. "Some Techniques for Handling Encipherment Keys." *ICL Technical Journal*, November 1982.

JUEN85 Jueneman, R., Matyas, S., and Meyer, C. "Message Authentication." *IEEE Communications Magazine*, September 1988.

JUEN87 Jueneman, R. "Electronic Document Authentication." *IEEE Network Magazine*, April 1987.

JURI97 Jurisic, A., and Menezes, A. "Elliptic Curves and Cryptography." *Dr. Dobb's Journal*, April 1997.

KAHN96 Kahn, D. *The Codebreakers: The Story of Secret Writing.* New York: Scribner, 1996.

KALI95 Kaliski, B., and Robshaw, M. "The Secure Use of RSA." *CryptoBytes*, Autumn 1995.

KALI96a Kaliski, B., and Robshaw, M. "Multiple Encryption: Weighing Security and Performance." *Dr. Dobb's Journal*, January 1996.

KALI96b Kaliski, B. "Timing Attacks on Cryptosystems." *RSA Laboratories Bulletin*, January 1996. http://www.rsasecurity.com/rsalabs.

KATZ00 Katzenbeisser, S., ed. *Information Hiding Techniques for Steganography and Digital Watermarking.* Boston: Artech House, 2000.

KEHN92 Kehne, A., Schonwalder, J., and Langendorfer, H. "A Nonce-Based Protocol for Multiple Authentications" *Operating Systems Review*, October 1992.

KELS98 Kelsey, J., Schneier, B., and Hall, C. "Cryptanalytic Attacks on Pseudorandom Number Generators." *Proceedings, Fast Software Encryption*, 1998. http://www.counterpane.com/pseudorandom_number.html.

KENT00 Kent, S. "On the Trail of Intrusions into Information Systems." *IEEE Spectrum*, December 2000.

KEPH97a Kephart, J., Sorkin, G., Chess, D., and White, S. "Fighting Computer Viruses." *Scientific American*, November 1997.

KEPH97b Kephart, J., Sorkin, G., Swimmer, B., and White, S. "Blueprint for a Computer Immune System." *Proceedings, Virus Bulletin International Conference*, October 1997.

KLEI90 Klein, D. "Foiling the Cracker: A Survey of, and Improvements to, Password Security." *Proceedings, UNIX Security Workshop II*, August 1990.

KNUD98 Knudsen, L., et al. "Analysis Method for Alledged RC4." *Proceedings, ASIACRYPT '98*, 1998.

KNUT97 Knuth, D. *The Art of Computer Programming, Volume 1: Fundamental Algorithms.* Reading, MA: Addison-Wesley, 1997.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Reading, MA: Addison-Wesley, 1998.

KOBL92 Koblas, D., and Koblas, M. "SOCKS." *Proceedings, UNIX Security Symposium III*, September 1992.

KOBL94 Koblitz, N. *A Course in Number Theory and Cryptography.*

KOCH96 Kocher, P. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems." *Proceedings, Crypto '96*, August 1996.

KOCH98 Kocher, P., Jaffe, J., and Jun, B. Introduction to Differential Power Analysis and Related Attacks." http://www.cryptography.com/dpa/technical/index.html.

KOHN78 Kohnfelder, L. *Towards a Practical Public-Key Cryptosystem.* Bachelor's Thesis, M.I.T., May 1978.

KOHL89 Kohl, J. "The Use of Encryption in Kerberos for Network Authentication." *Proceedings, Crypto '89*, 1989; New York: Springer-Verlag.

KOHL94 Kohl, J., Neuman, B., and Ts'o, T. "The Evolution of the Kerberos Authentication Service." in Brazier, F., and Johansen, D. *Distributed Open Systems.* Los Alamitos, CA: IEEE Computer Society Press, 1994. Available at http://web.mit.edu/kerberos/www/papers.html.

KONH81 Konheim, A. *Cryptography: A Primer.* New York: Wiley, 1981.

KORN96 Korner, T. *The Pleasures of Counting.* Cambridge, England: Cambridge University Press, 1996.

KUMA97 Kumar, I. *Cryptology.* Laguna Hills, CA: Aegean Park Press, 1997.

KUMA98 Kumanduri, R., and Romero, C. *Number Theory with Computer Applications.* Upper Saddle River, NJ: Prentice Hall, 1998.

KUMA98 Kumanduri, R., and Romero, C. *Number Theory with Computer Applications.* Upper Saddle River, NJ: Prentice Hall, 1998.

LAM92a Lam, K., and Gollmann, D. "Freshness Assurance of Authentication Protocols" *Proceedings, ESORICS '92*, 1992; New York: Springer-Verlag.

LAM92b Lam, K., and Beth, T. "Timely Authentication in Distributed Systems." *Proceedings, ESORICS '92*, 1992; New York: Springer-Verlag.

LE93 Le, A., Matyas. S., Johnson, D., and Wilkins, J. "A Public Key Extension to the Common Cryptographic Architecture." *IBM Systems Journal*, No. 3, 1993.

LEHM51 Lehmer, D. "Mathematical Methods in Large-Scale Computing." *Proceedings, 2nd Symposium on Large-Scale Digital Calculating Machinery*, Cambridge: Harvard University Press, 1951.

LEVE90 Leveque, W. *Elementary Theory of Numbers.* New York: Dover, 1990.

LEWA00 Lewand, R. *Cryptological Mathematics.* Washington, DC: Mathematical Association of America, 2000.

LEWI69 Lewis, P., Goodman, A., and Miller, J. "A Pseudo-Random Number Generator for the System/360." *IBM Systems Journal*, No. 2, 1969.

LIDL94 Lidl, R., and Niederreiter, H. *Introduction to Finite Fields and Their Applications.* Cambridge: Cambridge University Press, 1994.

LIPM00 Lipmaa, H., Rogaway, P., and Wagner, D. "CTR Mode Encryption." *NIST First Modes of Operation Workshop*, October 2000. http://csrc.nist.gov/encryption/modes.

LODI98 Lodin, S., and Schuba, C. "Firewalls Fend Off Invasions from the Net." *IEEE Spectrum*, February 1998.

LUNT88 Lunt, T., and Jagannathan, R. "A Prototype Real-Time Intrusion-Detection Expert System." *Proceedings, 1988 IEEE Computer Society Symposium on Research in Security and Privacy*, April 1988.

MACG97 Macgregor, R., Ezvan, C., Liguori, L., and Han, J. *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice.* IBM RedBook SG24-4978-00, 1997. Available at www.redbooks.ibm.com.

MADS93 Madsen, J. "World Record in Password Checking." *Usenet, comp.security.misc newsgroup*, August 18, 1993.

MANT01 Mantin, I., Shamir, A. "A Practical Attack on Broadcast RC4." *Proceedings, Fast Software Encryption*, 2001.

MARK97 Markham, T. "Internet Security Protocol." *Dr. Dobb's Journal*, June 1997.

MATS93 Matsui, M. "Linear Cryptanalysis Method for DES Cipher." *Proceedings, EUROCRYPT '93*, 1993; New York: Springer-Verlag.

MATY91a Matyas, S. "Key Handling with Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MATY91b Matyas, S., Le, A., and Abraham, D. "A Key-Management Scheme Based on Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MCHU00 McHugh, J., Christie, A., and Allen, J. "The Role of Intrusion Detection Systems." *IEEE Software*, September/October 2000.

MEIN01 Meinel, C. "Code Red for the Web." *Scientific American*, October 2001.

MENE97 Menezes, A., Oorshcot, P., and Vanstone, S. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press, 1997.

MERK79 Merkle, R. *Secrecy, Authentication, and Public Key Systems.* Ph.D. Thesis, Stanford University, June 1979.

MERK81 Merkle, R., and Hellman, M. "On the Security of Multiple Encryption." *Communications of the ACM*, July 1981.

MERK89 Merkle, R. "One Way Hash Functions and DES." *Proceedings, CRYPTO '89*, 1989; New York: Springer-Verlag.

MEYE82 Meyer, C., and Matyas, S. *Cryptography: A New Dimension in Computer Data Security.* New York: Wiley, 1982.

MEYE88 Meyer, C., and Schilling, M. "Secure Program Load with Modification Detection Code." *Proceedings, SECURICOM 88*, 1988.

MILL75 Miller, G. "Riemann's Hypothesis and Tests for Primality." *Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing*, May 1975.

MILL88 Miller, S., Neuman, B., Schiller, J., and Saltzer, J. "Kerberos Authentication and Authorization System." *Section E.2.1, Project Athena Technical Plan*, M.I.T. Project Athena, Cambridge, MA. 27 October 1988.

MILL98 Miller, S. *IPv6: The New Internet Protocol.* Upper Saddle River, NJ: Prentice Hall, 1998.

MIST96 Mister, S., and Adams, C. "Practical S-Box Design." *Proceedings, Workshop in Selected Areas of Cryptography, SAC '96.* 1996.

MIST98 Mister, S., and Tavares, S. "Cryptanalysis of RC4-Like Ciphers." *Pro-*

ceedings, Workshop in Selected Areas of Cryptography, SAC '98. 1998.

MITC90 Mitchell, C., Walker, M., and Rush, D. "CCITT/ISO Standards for Secure Message Handling." *IEEE Journal on Selected Areas in Communications*, May 1989.

MITC92 Mitchell, C., Piper, F., and Wild, P. "Digital Signatures." In [SIMM92].

MIYA90 Miyaguchi, S., Ohta, K., and Iwata, M. "Confirmation that Some Hash Functions Are Not Collision Free." *Proceedings, EUROCRYPT '90*, 1990; New York: Springer-Verlag.

MUFT89 Muftic, S. *Security Mechanisms for Computer Networks.* New York: Ellis Horwood, 1989.

MURH98 Murhammer, M., et al. *TCP/IP: Tutorial and Technical Overview.* Upper Saddle River: NJ: Prentice Hall, 1998.

MURP90 Murphy, S. "The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts." *Journal of Cryptology*, No. 3, 1990.

MYER91 Myers, L. *Spycomm: Covert Communication Techniques of the Underground.* Boulder, CO: Paladin Press, 1991.

NACH97 Nachenberg, C. "Computer Virus-Antivirus Coevolution." *Communications of the ACM*, January 1997.

NECH92 Nechvatal, J. "Public Key Cryptography." In [SIMM92].

NECH00 Nechvatal, J., et al. *Report on the Development of the Advanced Encryption Standard.* National Institute of Standards and Technology. October 2, 2000.

NEED78 Needham, R., and Schroeder, M. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM*, December 1978.

NEUM90 Neumann, P. "Flawed Computer Chip Sold for Years." *RISKS-FORUM Digest*, Vol. 10, No. 54, October 18, 1990.

NEUM93a Neuman, B., and Stubblebine, S. "A Note on the Use of Timestamps as Nonces." *Operating Systems Review*, April 1993.

NEUM93b Neuman, B. "Proxy-Based Authorization and Accounting for Distributed Systems." *Proceedings of the 13th International Conference on Distributed Computing Systems*, May 1993.

NICH96 Nichols, R. *Classical Cryptography Course.* Laguna Hills, CA: Aegean Park Press, 1996.

NICH99 Nichols, R. ed. *ICSA Guide to Cryptography.* New York: McGraw-Hill, 1999.

NIST97 National Institute of Standards and Technology. "Request for Candidate Algorithm Nominations for the Advanced Encryption Standard." Federal Register, September 12, 1997.

ODLY95 Odlyzko, A. "The Future of Integer Factorization." *CryptoBytes*, Summer 1995.

OORS90 Oorschot, P., and Wiener, M. "A Known-Plaintext Attack on Two-Key Triple Encryption." *Proceedings, EUROCRYPT '90*, 1990; New York: Springer-Verlag.

OORS94 Oorschot, P., and Wiener, M. "Parallel Collision Search with Application to Hash Functions and Discrete Logarithms." *Proceedings, Second ACM Conference on Computer and Communications Security*, 1994.

OPPL97 Oppliger, R. "Internet Security: Firewalls and Beyond." *Communications of the ACM*, May 1997.

ORE67 Ore, O. *Invitation to Number Theory.* Washington, DC: The Mathematical Association of America, 1967.

PARK88 Park, S., and Miller, K. "Random Number Generators: Good Ones Are Hard to Find." *Communications of the ACM*, October 1988.

PFLE97 Pfleeger, C. *Security in Computing.* Upper Saddle River, NJ: Prentice Hall, 1997.

PIAT91 Piattelli-Palmarini, M. "Probability: Neither Rational nor Capricious." *Bostonia*, March 1991.

POHL81 Pohl, I., and Shaw, A. *The Nature of Computation: An Introduction to Computer Science.* Rockville, MD: Computer Science Press, 1981.

POPE79 Popek, G., and Kline, C. "Encryption and Secure Computer Networks." *ACM Computing Surveys*, December 1979.

PORR92 Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection.* Master's Thesis, University of California at Santa Barbara, July 1992.

PREN96 Preneel, B., and Oorschot, P. "On the Security of Two MAC Algorithms." *Lecture Notes in Computer Science 1561; Lectures on Data Security*, 1999; New York: Springer-Verlag.

PREN97 Preneel, B., Bosselaers, A., and Dobbertin, H. "The Cryptographic Hash Function RIPEMD-160." *CryptoBytes*, Autumn 1997.

PREN99 Preneel, B. "The State of Cryptographic Hash Functions." *Proceedings, EUROCRYPT '96*, 1996; New York: Springer-Verlag.

PROC01 Proctor, P. *The Practical Intrusion Detection Handbook.* Upper Saddle River, NJ: Prentice Hall, 2001.

RABI78 Rabin, M. "Digitalized Signatures." In *Foundations of Secure Computation*, DeMillo, R., Dobkin, D., Jones, A., and Lipton, R., eds. New York: Academic Press, 1978.

RABI80 Rabin, M. "Probabilistic Algorithms for Primality Testing." *Journal of Number Theory*, December 1980.

RAND55 Rand Corporation. *A Million Random Digits.* New York: The Free Press, 1955. http://www.rand.org/publications/classics/randomdigits.

RESC01 Rescorla, E. *SSL and TLS: Designing and Building Secure Systems.* Reading, MA: Addison-Wesley, 2001.

RIBE96 Ribenboim, P. *The New Book of Prime Number Records.* New York: Springer-Verlag, 1996.

RIVE78 Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM*, February 1978.

RIVE90 Rivest, R. "The MD4 Message Digest Algorithm." *Proceedings, Crypto '90*, August 1990; New York: Springer-Verlag.

RIVE94 Rivest, R. "The RC5 Encryption Algorithm." *Proceedings, Second International Workshop on Fast Software Encryption*, December 1994; New York: Springer-Verlag.

RIVE95 Rivest, R. "The RC5 Encryption Algorithm." *Dr. Dobb's Journal*, January 1995.

ROBS95a Robshaw, M. *Stream Ciphers.* RSA Laboratories Technical Report TR-701, July 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROBS95b Robshaw, M. *Block Ciphers.* RSA Laboratories Technical Report TR-601, August 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROBS95c Robshaw, M. *MD2, MD4, MD5, SHA and Other Hash Functions.* RSA Laboratories Technical Report TR-101, July 1995. http://www.rsasecurity.com/rsalabs/index.html.

ROSE00 Rosen, K. *Elementary Number Theory and its Applications.* Reading, MA: Addison-Wesley, 2000.

ROSI99 Rosing, M. *Implementing Ellipt Curve Cryptography.* Greeenwich, CT: Manning Publications, 1999.

RUBI97 Rubin, A. "An Experience Teaching a Graduate Course in Cryptography." *Cryptologia*, April 1997.

RUEP92 Rueppel, T. "Stream Ciphers." In [SIMM92].

SAFF93 Safford, D., Schales, D., and Hess, D. "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment." *Proceedings, UNIX Security Symposium IV*, October 1993.

SAUE81 Sauer, C., and Chandy, K. *Computer Systems Performance Modeling.* Englewood Cliffs, NJ: Prentice Hall, 1981.

SCHA96 Schaefer, E. "A Simplified Data Encryption Standard Algorithm." *Cryptologia*, January 1996.

SCHN91 Schnorr, C. "Efficient Signatures for Smart Card." *Journal of Cryptology*, No. 3, 1991.

SCHN93 Schneier, B. "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)." *Proceedings, Workshop on Fast Software Encryption*, December 1993; New York: Springer-Verlag.

SCHN94 Schneier, R. "The Blowfish Encryption Algorithm." *Dr. Dobb's Journal*, April 1994.

SCHN96 Schneier, B. *Applied Cryptography.* New York: Wiley, 1996.

SCHN00 Schneier, B. *Secrets and Lies: Digital Security in a Networked World.* New York: Wiley 2000.

SHAN49 Shannon, C. "Communication Theory of Secrecy Systems." *Bell Systems Technical Journal*, No. 4, 1949.

SIMM92 Simmons, G., ed. *Contemporary Cryptology: The Science of Information Integrity.* Piscataway, NJ: IEEE Press, 1992.

SIMM93 Simmons, G. "Cryptology." *Encyclopaedia Britannica,* 15th ed., 1993.

SIMO95 Simovits, M. *The DES: An Extensive Documentation and Evaluation.* Laguna Hills, CA: Aegean Park Press, 1995.

SING99 Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* New York: Anchor Books, 1999.

SINK66 Sinkov, A. *Elementary Cryptanalysis: A Mathematical Approach.* Washington, DC: The Mathematical Association of America, 1966.

SMIT97 Smith, R. *Internet Cryptography.* Reading, MA: Addison-Wesley, 1997.

SNAP91 Snapp, S., et al. "A System for Distributed Intrusion Detection." *Proceedings, COMPCON Spring '91*, 1991.

SPAF92a Spafford, E. "Observing Reusable Password Choices." *Proceedings, UNIX Security Symposium III*, September 1992.

SPAF92b Spafford, E. "OPUS: Preventing Weak Password Choices." *Computers and Security*, No. 3, 1992.

STAL00 Stallings, W. *Data and Computer Communications, Sixth Edition.* Upper Saddle River, NJ: Prentice Hall, 2000.

STAL02 Stallings, W. "The Advanced Encryption Standard." *Cryptologia*, to appear.

STEI88 Steiner, J., Neuman, C., and Schiller, J. "Kerberos: An Authentication Service for Open Networked Systems." *Proceedings of the Winter 1988 USENIX Conference*, February 1988.

STEP93 Stephenson, P. "Preventive Medicine." *LAN Magazine*, November 1993.

STER92 Sterling, B. *The Hacker Crackdown: Law and Disorder on the Electronic Frontier.* New York: Bantam, 1992.

STEV94 Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

STIN02 Stinson, D. *Cryptography: Theory and Practice.* Boca Raton, FL: CRC Press, 2002.

STOL88 Stoll, C. "Stalking the Wily Hacker." *Communications of the ACM*, May 1988.

STOL89 Stoll, C. *The Cuckoo's Egg.* New York: Doubleday, 1989.

THOM84 Thompson, K. "Reflections on Trusting Trust (Deliberate Software Bugs)." *Communications of the ACM*, August 1984.

TIME90 Time, Inc. *Computer Security, Understanding Computers Series.* Alexandria, VA: Time-Life Books, 1990.

TIPP27 Tippett, L. *Random Sampling Numbers.* Cambridge, England: Cambridge University Press, 1927.

TOUC95 Touch, J. "Performance Analysis of MD5." *Proceedings, SIGCOMM '95*, October 1995.

TSUD92 Tsudik, G. "Message Authentication with One-Way Hash Functions." *Proceedings, INFOCOM '92*, May 1992.

TUCH79 Tuchman, W. "Hellman Presents No Shortcut Solutions to DES." *IEEE Spectrum*, July 1979.

TUNG99 Tung, B. *Kerberos: A Network Authentication System.* Reading, MA: Addison-Wesley, 1999.

VACC89 Vaccaro, H., and Liepins, G. "Detection of Anomalous Computer Session Activity." *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1989.

VOYD83 Voydock, V., and Kent, S. "Security Mechanisms in High-Level Network Protocols." *Computing Surveys*, June 1983.

WACK02 Wack, J., Cutler, K., and Pole, J. *Guidelines on Firewalls and Firewall Policy.* NIST Special Publication SP 800-41, January 2002.

WAYN93 Wayner, P. "Should Encryption Be Regulated?" *Byte*, May 1993.

WAYN96 Wayner, P. *Disappearing Cryptography.* Boston: AP Professional Books, 1996.

WEBS86 Webster, A., and Tavares, S. "On the Design of S-Boxes." *Proceedings, Crypto '85*, 1985; New York: Springer-Verlag.

WIEN90 Wiener, M. "Cryptanalysis of Short RSA Secret Exponents." *IEEE Transactions on Information Theory*, vol. IT-36, 1990.

WEIS93 Weiss, J., and Schremp, D. "Putting Data on a Diet." *IEEE Spectrum*, August 1993.

WILL76 Williamson, M. *Thoughts on Cheaper Non-Secret Encryption.* CESG Report, August 1976.

WOO92a Woo, T., and Lam, S. "Authentication for Distributed Systems." *Computer*, January 1992.

WOO92b Woo, T., and Lam, S. " 'Authentication' Revisited." *Computer*, April 1992.

YIN97 Yin, Y. "The RC5 Encryption Algorithm: Two Years On." *CryptoBytes*, Winter 1997.

YUVA79 Yuval, G. "How to Swindle Rabin." *Cryptologia*, July 1979.

ZENG91 Zeng. K., Yang, C., Wei, D., and Rao, T. "Pseudorandom Bit Generators in Stream-Cipher Cryptography." *Computer*, February 1991.

ZIV77 Ziv, J., and Lempel, A. "A Universal Algorithm for Sequential Data Compression." *IEEE Transactions on Information Theory*, May 1977.