国外计算机科学教材系列

Pearson

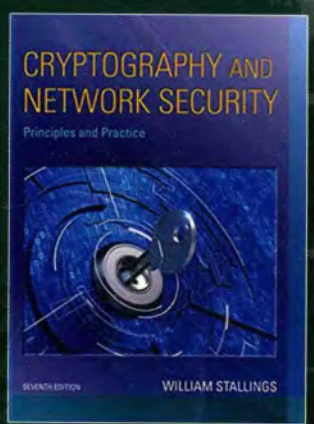
★William Stallings

密码编码学与网络安全

——原理与实践（第七版）

Cryptography and Network Security

Principles and Practice, Seventh Edition



[美] William Stallings 著

王后珍 李 莉 杜瑞颖 等译

张焕国 审校

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

权威经典

密码编码学与网络安全
——原理与实践（第七版）

Cryptography and Network Security
Principles and Practice, Seventh Edition

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。纸质教材分为六部分：背景知识部分介绍计算机与网络安全的概念、数论；对称密码部分讨论古典加密技术、分组加密和数据加密标准、有限域、高级加密标准、分组加密操作、随机位生成和流密码；非对称密码部分讨论公钥加密和RSA、其他公钥加密体制；密码编码数据完整性算法部分讨论哈希函数、消息认证编码、数字签名；互信部分讨论密钥管理与分发、用户鉴别；网络和互联网安全部分讨论网络访问控制和云安全、传输层安全、无线网络安全、电子邮件安全和IP安全。联机内容分为两部分：系统安全部分讨论恶意软件、入侵者、防火墙；法律和道德问题部分讨论与计算机和网络安全相关的法律与道德问题。与第六版相比，书的章节组织基本不变，但增加了许多新内容，如数论、格式保留加密、真随机数生成器、云安全、传输层安全、移动设备安全等。

本书可作为高校计算机、网络安全、信息安全、软件工程等专业研究生和高年级本科生的教材，也可供从事网络空间安全、计算机、通信、电子工程等领域的科技人员参考。

作者简介

William Stallings：美国圣母大学电子工程学士，麻省理工学院计算机科学博士，《密码术》期刊编委。美国多家高科技公司执行总监，政府机构、计算机软/硬件供应商顾问；曾为各种计算机和操作系统设计并实现TCP/IP和OSI协议套件；创建并维护了计算机科学专业学生资源网站ComputerScienceStudent.com。出版图书18种，内容涉及计算机安全、计算机网络和计算机系统结构等；在《IEEE进展》、《ACM计算评论》和《密码术》等杂志上发表文章多篇；先后13次获美国年度最佳计算机科学教科书奖和学术作者协会奖。

Pearson
ALWAYS LEARNING

Pearson
www.pearson.com

ISBN 978-7-121-32921-0
9 787121 329210 >
定价：95.00 元

策划编辑：谭海平
责任编辑：李秦华
责任美编：孙焱津

华信 欢迎登录 免费获取本书教学资源 http://www.hxedu.com.cn

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(第七版)

Cryptography and Network Security

Principles and Practice, Seventh Edition

[美] William Stallings 著

王后珍 李 莉 杜瑞颖 等译

张焕国 审校

電子工業出版社

Publishing House of Electronics Industry

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。纸质教材分为六部分：背景知识部分介绍计算机与网络安全的概念、数论；对称密码部分讨论古典加密技术、分组加密和数据加密标准、有限域、高级加密标准、分组加密操作、随机位生成和流密码；非对称密码部分讨论公钥加密和 RSA、其他公钥加密体制；密码编码数据完整性算法部分讨论哈希函数、消息认证编码、数字签名；互信部分讨论密钥管理与分发、用户鉴别；网络和互联网安全部分讨论网络访问控制和云安全、传输层安全、无线网络安全、电子邮件安全和 IP 安全。在线内容分为两部分：系统安全部分讨论恶意软件、入侵者、防火墙；法律和道德问题部分讨论与计算机和网络安全相关的法律与道德问题。与第六版相比，书的章节组织基本不变，但增加了许多新内容，如数论、格式保留加密、真随机数生成器、云安全、传输层安全、移动设备安全等。

本书可作为高校计算机、网络安全、信息安全、软件工程等专业研究生和高年级本科生的教材，也可供从事网络空间安全、计算机、通信、电子工程等领域的科技人员参考。

Authorized translation from the English language edition, entitled Cryptography and Network Security: Principles and Practice, Seventh Edition, ISBN: 9780134444284 by William Stallings. Published by Pearson Education, Inc. Copyright © 2017 Pearson Education, Inc.

All rights Reserved. No part of this book may be reproduced or transmitted in any forms or by any means, electronic or mechanical, including photocopying recording or by any information storage retrieval systems, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD, and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY, Copyright © 2017.

本书中文简体字版专有出版权由 Pearson Education（培生教育出版集团）授予电子工业出版社，未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权贸易合同登记号　图字：01-2016-9453

图书在版编目（CIP）数据

密码编码学与网络安全：原理与实践：第七版 /（美）威廉·斯托林斯（William Stallings）著；王后珍等译.
北京：电子工业出版社，2017.12
（国外计算机科学教材系列）
书名原文：Cryptography and Network Security: Principles and Practice, Seventh Edition
ISBN 978-7-121-32921-0

I. ①密… II. ①威… ②王… III. ①电子计算机－密码术－高等学校－教材 ②计算机网络－安全技术－高等学校－教材 IV. ①TP309.7②TP393.08

中国版本图书馆 CIP 数据核字（2017）第 257831 号

策划编辑：谭海平
责任编辑：李秦华
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
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
开　　本：787×1092　1/16　印张：33.75　字数：864 千字
版　　次：2003 年 11 月第 1 版（原著第 3 版）
　　2017 年 12 月第 5 版（原著第 7 版）
印　　次：2017 年 12 月第 1 次印刷
定　　价：95.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

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。

本书咨询联系方式：（010）88254552，tan02@phei.com.cn。

译 者 序

随着信息科学技术的高速发展和广泛应用，社会实现了信息化，人类社会进入信息时代。人们生活和工作在由物理世界、人类社会和网络空间组成的三元世界中。网络空间是信息时代人们赖以生存的信息环境，是所有信息系统的集合。

哪里有信息，哪里就存在信息安全问题，信息安全是信息的影子。信息论的基本观点告诉我们：系统是载体，信息是内涵。网络空间是人类和信息的生存环境，因此网络空间安全是人类和信息对网络空间的基本要求。网络空间是所有信息系统的集合，是复杂的巨系统。因此，网络空间存在更加突出的信息安全问题。网络空间安全的核心内涵仍是信息安全，没有信息安全就没有网络空间安全。

当前，一方面是信息技术与产业的空前繁荣，另一方面是危害信息安全的事件不断发生。敌对势力的破坏、黑客攻击、利用计算机犯罪、网上有害内容泛滥、隐私泄露等，对信息安全构成了极大威胁。信息安全的形势是十分严峻的。因此，信息安全成为世人关注的社会问题，并成为信息科学技术领域中的研究热点。

我国已经成为信息产业大国，但仍不是信息产业强国。实现信息化和确保信息安全是建设中国特色社会主义强国的两个重要方面。两者相辅相成，缺一不可。没有信息化，就没有国家现代化；没有信息安全，就没有国家安全。显然，只有同时实现信息化并确保信息安全，才能把我国建成中国特色社会主义强国。

把我国建设成信息技术和产业强国，人才是关键。人才培养，教育是关键。目前，我国许多大专院校都开设了信息安全专业或开设了信息安全课程，迫切需要一本合适的教科书。为此，电子工业出版社组织我们于 2015 年翻译出版了《密码编码学与网络安全——原理与实践（第六版）》这本优秀的教科书。这本书翻译出版后得到广大读者的厚爱，许多著名大学都采用它作为教材，为我国信息安全人才培养和信息安全知识传播发挥了重要作用。

2017 年原书作者又出版了该书的第七版。与第六版相比，第七版大体上保持了相同的章节，但修正了许多内容并增加了一些新的内容。最主要的变化包括以下几个方面：

① 安全设计基本原则：第 1 章新增了一个小节，讨论美国国家信息保障/网络空间防御专业认证计划（由美国国家安全局和国土安全部联合举办）所列出的一些安全设计原则。

② 攻击面和攻击树：第 1 章新增了这两个概念，有助于安全威胁的评估和归类。

③ 数论知识：将第六版中所提到的数论知识统一写成一个独立的第 2 章。

④ 有限域：对有限域章节进行了修改和扩展。

⑤ 保留格式加密：第 7 章新增了一节来讨论这种新的加密模式。

⑥ 真随机数生成器的校正和性能测试：现在第 8 章包含了这些重要的研究方向。

⑦ 用户认证模型：第 15 章给出了一般用户认证模型的新描述方法。

⑧ 云安全：第 16 章有关云安全的内容做了更新和扩展。

⑨ 传输层安全（TLS）：第 17 章的传输层安全相关内容进行了更新，并且增加了新 TLS 版本 1.3 的讨论。

⑩ 邮件安全：完全重写了第 19 章。

其中，新增了以下内容：

- 对邮件的威胁及一种全面保证邮件安全的方法。
- 保障 SMTP 机密性的新方法 STARTTLS。
- DNSSEC 在保证邮件安全中所起的作用。
- 基于 DNS 的命名实体认证（DANE）以及使用这种方法加强 SMTP 和 S/MIME 中证书的安全性。
- 发送方策略框架（SPF）。SPF 是一种用于认证电子邮件发件人的标准方法。
- 允许发件人自己定义邮件的处理办法 DMARC。

其中，修改了以下内容：

- 有关域名密钥识别邮件（DKIM）标准的讨论。
- 将 S/MIME 更新至最新的 3.2 版本。

为了使广大读者能够读到新版书，电子工业出版社又组织我们翻译出版了本书第七版。

本书的作者 William Stallings 先后获得了 Notre Dame 电气工程学士学位和 MIT 计算机科学博士学位。他累计编写出版了 48 本计算机网络和计算机体系结构领域的书籍，在计算机网络和计算机体系结构的学术交流和教育方面做出了卓越的贡献。本书就是其中最成功的一本书籍。William Stallings 的著作不仅学术造诣很高，而且十分实用，先后 13 次获得美国“教材和著作家协会”（Textbook and Academic Authors Assiciation）颁发的年度最佳计算机科学教材奖。

本书系统地介绍了密码学与网络安全的基本原理和应用技术。全书主要包含以下八个部分。第一部分：概览。主要介绍计算机与网络安全概念和数论知识。第二部分：对称密码。主要介绍古典密码、数据加密标准（DES）、有限域知识、高级加密标准（AES）、分组密码工作模式、伪随机数和流密码。第三部分：公钥密码。介绍了公钥密码原理、RSA 密码、ElGamal 密码和椭圆曲线密码。第四部分：密码学中的数据完整性算法。介绍了密码学 Hash 函数、消息认证码和数字签名。第五部分：互信，介绍了密钥管理和用户认证。第六部分：网络与 Internet 安全。讨论了网络访问控制和云安全、传输层安全、无线网络安全、电子邮件安全和 IP 安全等内容。第七部分：系统安全。讨论了恶意软件、非法入侵、防火墙技术。第八部分：法律和道德，讨论了与计算机和网络安全相关的法律和道德问题。

为了使中文版读者能够读到原书的完整内容，我们特别翻译了原书的在线内容（第七部分：系统安全、第八部分：法律和道德、附录 C-Y）。从而给中文读者提供一本完整的中文《密码编码学与网络安全——原理与实践（第七版）》[①]。

本书内容丰富，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。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。

本书的第一部分和附录由李莉翻译。第二部分由唐明翻译。第三部分和前言由王后珍翻

① 有关本书在线部分的中文版已上载至华信教育资源网（http://www.hxedu.com.cn），有兴趣的读者可免费注册下载——编者注。

译。第四部分由王张宜翻译。第五部分由陈晶翻译。第六部分由杜瑞颖翻译。第七部分和第八部分由彭国军翻译。

全书由张焕国统稿和校审。

研究生刘金会、刘鳌、汪鹏程、陈施旅、罗华、郭崎、文皓冬、熊璐、胡岸琪、于慧、陈震杭参与了翻译和译稿整理工作。

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。

译者于武汉大学珞珈山

2017 年 6 月

前　言

第七版的新内容

在本书第六版出版后的 4 年中，该领域仍处于不断地发展变革之中。在新版中，我试图在继续广泛涵盖本领域的主要内容的同时，增加这些新的变化。进行本次修订之初，许多讲授该课程的教授和研究该领域的专业人员都已经阅读过本书的第六版。这使得许多地方的叙述变得更清晰、更紧凑，对插图也进行了改进。

除了这些为改进教学法和使用户易读所做的修改以外，还有一些实质性的变化贯穿在本书中。与第六版相比，本书大体上保持了相同的章节，但修正了许多内容并增加了一些新内容。最主要的变化包括以下几个方面：

- **基本的安全设计原则**　第 1 章包含了一个新的小节讨论美国国家信息保障/网络空间防御专业认证计划（由美国国家安全局和国土安全局联合举办）所列出的一些安全设计原则。
- **攻击面和攻击树**　第 1 章包含了一个新的小节描述这两个概念，有助于安全威胁的评估和归类。
- **数论相关知识介绍**　为了便于读者参考，把第六版中所提到的数论知识点统一到了一个独立章节，即第 2 章。第 2 章的相关部分是根据本书密码算法学习需要来安排的。
- **有限域**　已经对有限域章节进行了修改和扩展，主要是增加了一些文字和新的图片以使读者便于理解。
- **保留格式加密**　这种相对较新的加密模式获得了越来越多的商业成功。第 7 章新增了一节来讨论这种加密模式。
- **真随机数生成器的校正和性能测试**　现在第 8 章包含了这些重要的研究方向。
- **用户认证模型**　第 15 章给出了一般用户认证模型的新描述，以助于统一讨论各种不同用户认证模型。
- **云安全**　第 16 章有关云安全的内容已经做了更新和扩展，力求凸显它的重要性和最新的研究进展。
- **传输层安全（TLS）**　第 17 章的传输层安全相关内容已经更新而且重新组织以求更加清晰，并且增加了新 TLS 版本 1.3 的讨论。
- **邮件安全**　第 19 章已经完全重写以求提供一个全面的、最新的邮件安全讨论。它包括：
  - 新增：讨论邮件威胁及一种全面的保证邮件安全方法。
  - 新增：增加 STARTTLS 的讨论。STARTTLS 是提供保证 SMTP 机密性的一种新方法。
  - 修改：S/MIME 已经更新至最新的 3.2 版本。
  - 新增：新增加了 DNSSEC 的讨论及其在保证邮件安全中所起的作用。

- 新增：讨论了基于 DNS 的命名实体认证（DANE）以及使用这种方法加强 SMTP 和 S/MIME 中证书的安全性。
- 新增：新增加了发送方策略框架（SPF）的讨论。SPF 是一种用于认证电子邮件发件人的标准方法。
- 修改：修订了有关域名密钥识别邮件（DKIM）标准的讨论。
- 新增：增加了 DMARC 的讨论。DMARC 是一种允许发件人自己定义邮件的处理办法，用于接收反馈报告的类型以及这些返回报告的频率。

本书的目标

本书的目标是概述密码学与网络安全的原理和应用。本书的前一部分给出关于密码学和网络安全的指导性的概述。后一部分讨论网络安全的实际应用，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掌握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背景知识的动机，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。

ACM/IEEE 计算机科学课程 2013 的支持

本书适合于学术和专业人员使用。作为教科书，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学与网络安全方面课程的教材，学时为一学期。本版的修订是为了支持当前草案版本的 ACM/IEEE 计算机科学课程 2013（CS2013）。CS2013 在课程体系中增加了 IAS（Information Assurance and Security）内容的课程，并将其作为计算机科学知识体系中的一个知识领域。CS2013 认为把 IAS 纳入课程体系，是因为 IAS 对于计算机科学教育具有关键作用。CS2013 把将所有课程分为三类：核心课程-1（课程应包含所有的课题）；核心课程-2（应包含全部或几乎全部的课题）；选修课程（根据意愿适当地提供广度与深度）。在 IAS 领域，CS2013 推荐把网络安全的基本概念纳入核心课程-1 和核心课程-2 中，而把密码学部分作为选修。本书实际上涵盖了 CS2013 所列举的三类课程中的所有课题。

本书还可用作参考用书或作为自学教材。

本书的组织

本书由以下八个部分组成。

- 概览
- 对称密码
- 公钥密码

- 密码学中的数据完整性算法
- 互信
- 网络与 Internet 安全
- 系统安全
- 法律与道德

本书还针对教学的需要，提供了计算机代数系统 Sage 和大量图表使得表达更加清晰。每一章中都有关键术语、习题、思考题和推荐读物。本书还给出了术语表，常用的首字母缩略词表和参考文献。另外，对于教师还提供了试题库。

教学支持文档[①]

本学科十分有趣而且发展迅速，本书的主要目的是为讲授这一学科内容提供一个有效的教学工具。该目的反映在本书的结构及支持文档。对于教师，我们提供了下列补充材料：

- **答案手册**：对于每章末尾的思考题和习题的答案。
- **项目手册**：对于下面列出的所有项目的建议的任务分配方案。
- **PPT 幻灯片**：包含所有章节内容的幻灯片，适于讲课中使用。
- **PDF 文件**：本书中所有图和表的副本。
- **习题集**：按章的习题集和答案。
- **教学大纲样例**：本书包含多于在一学期讲授的内容，因此为教师提供了若干教学大纲样例，以指导在有限时间使用本书。这些样例基于使用本书第五版的教授的实践经验。

在教师资源中心 IRC（Instructor Resource Center）中提供了所有的这些支持文档，可以通过出版商的网站：www.personhighered.com/stallings 或点击本书网站 WilliamStallings.com/Cryptography 上的教师资源链接来获取。

在本书网站 WilliamStallings.com/Cryptography 上（单击教师资源链接）还包括以下资源：

- 链接到使用本书的其他课程的网站。
- 使用本书的教师的邮箱列表，他们通过邮件互相讨论问题或对作者提出建议。

项目和其他学生练习

对许多教师来说，密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项目使得学生有机会亲手实践，以加深对课本中所学知识的理解。本书在很大程度上对该课程提供全面的支持，包含了课程中一整套的项目组件。教师资源中心（IRC）不仅包含如何布置和构建项目，而且还包括一系列涵盖本书内容的推荐教学项目：

- **Sage 项目** 下一节中详细介绍。
- **黑客项目** 本项目的目的是阐明入侵检测和预防的关键问题。

① 教辅申请方式请联系 Te_service@phei.com.cn 获取（只向教师提供）——编者注。

- **分组密码项目** 本实验对 AES 加密算法的操作过程进行跟踪，手工进行一轮的计算，并使用不同的分组密码工作模式进行计算。实验也包括 DES 算法。每种情况下都由在线（或离线下载）Java 小程序来实现 AES 或 DES 的运算。
- **实验室练习** 一系列针对本书里的概念进行编程和做实验的项目。
- **研究项目** 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。
- **编程项目** 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。
- **安全评估实践** 用于检验已有组织机构的现有架构和实现的一系列活动。
- **防火墙项目** 一个简易的网络防火墙可视化模拟器课题，用以支持讲授防火墙基本原理的练习。
- **案例研究** 一系列的实际案例研究，包括学习目标，案例描述，系列的案例讨论问题。
- **书面作业** 每一章里推荐了一些书面作业。
- **课外阅读/报告作业** 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。

这些各种各样的项目和学生练习使得教师能够方便地使用本书，把它当作丰富多样的教学经验中的一个组件。从而可以方便地安排课程计划，以适应教师和学生的各种特殊需求。具体细节请参见附录 A。

Sage 计算机代数系统

本书的一个最重要的特色就是使用 Sage 实现密码算法示例和作业。Sage 是一个开源的、跨平台的免费软件包，它实现了一个强大的、灵活的、易学的数学和计算机代数系统。与 Mathematica，Maple 和 MATLAB 等系统不同，Sage 没有使用许可和使用费的限制。因此 Sage 可以在学校的计算机和网络上使用，学生也可以分别将其下载到他们自己的个人电脑上在家里使用。使用 Sage 的另外一个好处是学生可以掌握一个非常强大、灵活的工具来帮助计算解决几乎所有的数学问题，而不仅限于密码学。

在对密码算法数学基础的教学过程中，使用 Sage 能够显著增强教学效果。本书的附录 B 中提供了涵盖各种密码学概念的大量 Sage 示例。

附录 C 按照密码学概念的分类给出了习题集，它能够使学生得到关于密码算法的第一手经验。本书的教师资源中心 IRC 为教师提供了该附录。附录 C 中专门有一节介绍如何下载和使用 Sage，另一节介绍 Sage 编程基础，除此以外还包括为学生准备的以下分类习题：

- **第 2 章——数论和有限域** 欧几里得算法和扩展欧几里得算法、多项式算术、有限域 GF（2^4）、欧拉函数、Miller-Rabin 测试、因式分解、模幂运算、离散数学以及中国剩余定理。
- **第 3 章——传统加密技术** 仿射密码和 Hill 密码。
- **第 4 章——分组密码和数据加密标准** 基于 SDES 的练习。
- **第 6 章——高级加密标准 AES** 基于 SAES 的练习。
- **第 8 章——伪随机数发生器和流密码** BBS、线性同余和 ANSI X9.17 伪随机数发生器。

- **第 9 章——公钥密码与 RSA** RSA 加密/解密以及签名。
- **第 10 章——密钥管理和其他公钥密码体制** Diffie-Hellman 密钥交换，椭圆曲线密码。
- **第 11 章——密码学 Hash 函数** 基于数论的 Hash 函数。
- **第 13 章——数字签名** DSA。

提供给学生的在线文档[①]

在新版书中，两个 Web 站点为学生提供了大量在线原始支持材料。WilliamStallings.com/Cryptography 上的伙伴网站（点击学生资源链接）包含一系列按章组织的章节和本书的勘误表。

购买新一版的书可以获得 6 个月在线材料访问权限，包括以下内容：

- **在线章节** 为了减少本书（英文版）的篇幅和成本，书中的 4 个章节提供了 PDF 格式的电子文档，其中包括关于计算机安全的三章以及关于法律和道德的一章。在本书的目录中列出了这些章。
- **在线附录** 支持材料包含了本书正文中涉及的大量有趣的话题，但在本书（英文版）纸质印刷版中没有提供。我们为对此感兴趣的学生们提供了包含了这些话题的总计 20 个在线附录。在本书的目录中列出了这些附录。
- **家庭作业和答案** 为了帮助学生更好地学习和理解本书内容，我们单独提供了家庭习题和答案集。
- **关键论文** 我们从专业文献中选择了一定数量的论文，其中许多是很难找到的，提供给读者进一步地阅读。
- **支持文档** 本书引用的其他各种类型的有用文档同时在线提供。
- **Sage 代码** 附录 B 中给出了示例的 Sage 源代码。如果学生们想要实现这些示例，可以以此作为参考。

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：Hossein Beyzavi（Marymount University），Donald F. Costello（University of Nebraska-Lincoln），James Haralambides（Barry University），Anand seetharam（California State University at Monterey Bay），Marius C. Silaghi（Florida Institute of Technology），Shambhu Upadhyaya（University at Buffalo），Zhengping Wu（California State University at san Bernardino），Liangliang Xiao（Frostburg State University），Seong-Moo（Sam）Yoo（The University of Alabama in Huntsville），and Hong Zhang（Armstrong State University）。

我还要感谢那些详细审阅其中某一章或数章的人员：Dino M. Amaral, Chris Andrew, Prof.（Dr）. C. Annamalai, Andrew Bain, Ricccardo Bernardini, Olivier Blazy, Zervopoulou Christina, Maria Christofi, Dhananjoy Dey, Mario Emmanuel, Mike Fikuart, Alexander Fries, Pierpaolo

① 部分文件也可登录华信教育资源网（www.hxedu.com）注册下载——编者注。

Giacomin, Pedro R. M. Inácio, Daniela Tamy Iwassa, Krzysztof Janowski, Sergey Katsev, Adnan kilic, Rob Knox, Mina Pourdashty, Yuri Poeluev, Pritesh Prajapati, Venkatesh Ramamoorthy, Andrea Razzini, Rami Rosen, Javier Scodelaro, Jamshid Shokrollahi, Oscar So, and David Tillemans。

此外，我也有幸审阅了一些领域大师的研究成果，这些大师包括英特尔公司的 Jesse Walker（英特尔的数字随机数发生器）、Vigil Security 公司的 Russ Housley（密钥封装）、Joan Daemen（AES），Santa Clara 大学的 Edward F. Schaefer（简化的 AES）、前 RSA 实验室的 Tim Mathews（S/MIME）、Waterloo 大学的 Alfred Menezes（椭圆曲线密码学）、*The Cryotogram* 一书的编辑与发行人 William Sutton（古典密码）、Johns Hopkins 大学的 Avi Rubin（数论）、信息安全公司的 Michael Markowitz（SHA 和 DSS）、IBM 因特网安全系统部的 Don Davis（Kerberos）、BBN 科技公司的 Steve Kent（X.509）和 Phil Zimmerman（PGP）。

Nikhil Bhargava（IIT Delhi）开发了一系列的在线家庭作业和解答。Microsoft 和华盛顿大学的 Dan Shumow 开发了附录 B 和附录 C 中所有的 Sage 示例和作业。Dakata 州立大学的 Sreekanth Malladi 教授开发了黑客练习。Australian Defence Force Academy 的 Lawrie Brown 提供了 AES/DES 分组密码项目和安全评估练习。

Purdue 大学的 Sanjay Rao 和 Ruben Torres 为教师资源中心（IRC）的实验室练习做了很多工作。下列人员为教师资源中心的项目计划做了许多工作：Henning Schulzrinne（Columbia 大学），Cetin Kaya Koc（Oregon 州立大学）和 David Balenson（Trusted Information Systems and George Washington University）。Kim McLaughlin 提供了习题库。

最后，我要感谢负责本书出版的工作人员，他们都做得很优秀。包括 Pearson 出版社的工作人员，特别是责任编辑 Tracy Johnson，项目经理 Carole Snyder，以及产品部经理 Bob Engelhardt。还要感谢 Pearson 出版社的市场和销售工作人员，没有他们的努力，这本书就不可能摆在你的面前。

关于作者

William Stallings 编写出版了 18 部著作，经修订再版累计共出版了 40 多本关于计算机安全、计算机网络和计算机体系结构等领域的书籍。他的著作无数次出现在出版物中，包括 *Proceedings of the IEEE*、*ACM Computing Reviews and Cryptologia*。

他 13 次获得美国“教材和著作家协会”（Text and Academic Authors Association）颁发的“年度最佳计算机科学教材”奖。

在过去的 30 年中，他曾在该领域的数个高科技企业中担任技术骨干、技术管理者和技术执行领导。他设计和实现了适用于从微型机到大型机的各种类型的计算机和操作系统的基于 TCP/IP 和基于 OSI 的协议。目前，他作为独立顾问为政府机构、计算机硬件制造商、软件开发商以及广大用户提供包括设计、选择和使用网络软件和产品的咨询服务。

他建设并维护计算机科学专业学生资源网站 WilliamStallings.com/StudentSupport.html。该网站为计算机科学专业的学生（和专业人员）提供各种文档和链接。他是 *Cryptologia* 杂志的编委，该杂志是密码学的学术期刊。

William Stallings 博士先后获得了 Notre Dame 电气工程学士学位和 MIT 计算机科学博士学位。

符 号 表

符号	表达式	含义
D,K	D(K,Y)	用密钥 K 和对称密码算法解密密文 Y
D,PR_a	D(PR_a,Y)	用 A 的私钥 PR_a 和非对称密码算法解密密文 Y
D,PU_a	D(PU_a,Y)	用 A 的公钥 PU_a 和非对称密码算法解密密文 Y
E,K	E(K,X)	用密钥 K 和对称密码算法加密明文 X
E,PR_a	E(PR_a,X)	用 A 的公钥 PU_a 和非对称密码算法加密明文 X
E,PU_a	E(PU_a,X)	用 A 的公钥 PU_a 和非对称密码算法加密明文 X
K		密钥
PR_a		用户 A 的私钥
PU_a		用户 A 的公钥
MAC,K	MAC(K,X)	消息 X 的消息认证码，密钥为 K
GF(p)		阶为 p 的有限域，p 为素数。域定义为 Z_p 及其上的模 p 算术运算
GF(2^n)		阶为 2^n 的有限域
Z_n		小于 n 的非负整数集合
gcd	gcd(i,j)	最大公因子，整除 i 和 j 的最大正整数
mod	a mod m	a 除以 m 的余数
mod, $\equiv$	$a \equiv b \pmod m$	$a \bmod m = b \bmod m$
mod, $\neq$	$a \neq b \pmod m$	$a \bmod m \neq b \bmod m$
dlog	$\mathrm{dlog}_{a,p}(b)$	以 a 为底 b 的对数，模 p 运算
φ	$\varphi(n)$	欧拉函数，小于 n 且和 n 互素的正整数个数
$\sum$	$\sum_{i=1}^{n} a_i$	$a_1+a_2+\cdots+a_n$
$\prod$	$\prod_{i=1}^{n} a_i$	$a_1 \times a_2 \times \cdots \times a_n$
$\mid$	$i \mid j$	i 整除得尽 j，即 i 除 j 的余数为零
$\lvert,\rvert$	$\lvert a \rvert$	a 的绝对值
$\parallel$	$x \parallel y$	级联 x 和 y
$\approx$	$x \approx y$	x 约等于 y
$\oplus$	$x \oplus y$	单比特变量时是异或运算，多比特变量时是按位异或
$\lfloor,\rfloor$	$\lfloor x \rfloor$	小于等于 x 的最大整数
$\in$	$x \in S$	元素 x 包含于集合 S
$\longleftrightarrow$	$A \longleftrightarrow (a_1,a_2,\cdots,a_k)$	整数 A 和整数序列$(a_1,a_2,\cdots,a_k)$对应

目　录

第一部分　概　览

第 1 章　计算机与网络安全概念 …… 2
1.1　计算机安全的概念 …… 3
1.2　OSI 安全架构 …… 5
1.3　安全攻击 …… 6
1.4　安全服务 …… 7
1.5　安全机制 …… 9
1.6　基本安全设计准则 …… 10
1.7　攻击面与攻击树 …… 12
1.8　网络安全模型 …… 15
1.9　标准 …… 16
1.10　关键术语、思考题和习题 …… 17

第 2 章　数论基础 …… 19
2.1　整除性和带余除法 …… 19
2.2　欧几里得算法 …… 21
2.3　模运算 …… 23
2.4　素数 …… 30
2.5　费马定理和欧拉定理 …… 31
2.6　素性测试 …… 34
2.7　中国剩余定理 …… 36
2.8　离散对数 …… 38
2.9　关键术语、思考题和习题 …… 42
附录 2A　mod 的含义 …… 45

第二部分　对 称 密 码

第 3 章　传统加密技术 …… 48
3.1　对称密码模型 …… 48
3.2　代替技术 …… 52
3.3　置换技术 …… 63
3.4　转轮机 …… 64
3.5　隐写术 …… 65
3.6　关键术语、思考题和习题 …… 66

第 4 章　分组密码和数据加密标准 …… 71
4.1　传统分组密码结构 …… 71
4.2　数据加密标准 …… 78
4.3　DES 的一个例子 …… 80
4.4　DES 的强度 …… 81
4.5　分组密码的设计原理 …… 83
4.6　关键术语、思考题和习题 …… 84

第 5 章　有限域 …… 87
5.1　群 …… 87
5.2　环 …… 89
5.3　域 …… 89
5.4　有限域 GF(p) …… 90
5.5　多项式运算 …… 93
5.6　有限域 GF(2^n) …… 98
5.7　关键术语、思考题和习题 …… 105

第 6 章　高级加密标准 …… 108
6.1　有限域算术 …… 108
6.2　AES 的结构 …… 109
6.3　AES 的变换函数 …… 114
6.4　AES 的密钥扩展 …… 122
6.5　一个 AES 例子 …… 124
6.6　AES 的实现 …… 128
6.7　关键术语、思考题和习题 …… 131
附录 6A　系数在 GF(2^8)中的多项式 …… 133

第 7 章　分组加密的工作模式 …… 136
7.1　多重加密与三重 DES …… 136
7.2　电码本模式 …… 140
7.3　密文分组链接模式 …… 141
7.4　密文反馈模式 …… 143
7.5　输出反馈模式 …… 145

7.6 计数器模式……146
7.7 用于面向分组的存储设备的 XTS-AES 模式……149
7.8 格式保持加密……153
7.9 关键术语、思考题和习题……164

第 8 章 伪随机数的产生和流密码……168
8.1 随机数产生的原则……169
8.2 伪随机数发生器……172
8.3 使用分组密码的伪随机数产生……174
8.4 流密码……179
8.5 RC4 算法……180
8.6 真随机数发生器……182
8.7 关键术语、思考题和习题……188

第三部分 公 钥 密 码

第 9 章 公钥密码学与 RSA……192
9.1 公钥密码体制的基本原理……193
9.2 RSA 算法……199
9.3 关键术语、思考题和习题……209

第 10 章 密钥管理和其他公钥密码体制……214
10.1 Diffie-Hellman 密钥交换……214
10.2 EIGamal 密码体制……217
10.3 椭圆曲线算术……220
10.4 椭圆曲线密码学……226
10.5 基于非对称密码的伪随机数生成器……228
10.6 关键术语、思考题和习题……230

第四部分 密码学中的数据完整性算法

第 11 章 密码学 Hash 函数……234
11.1 密码学 Hash 函数的应用……235
11.2 两个简单的 Hash 函数……238
11.3 需求和安全性……240
11.4 基于分组密码链接的 Hash 函数……244
11.5 安全 Hash 算法……245
11.6 SHA-3……252
11.7 关键术语、思考题和习题……260

第 12 章 消息认证码……264
12.1 对消息认证的要求……264
12.2 消息认证函数……265
12.3 对消息认证码的要求……270
12.4 MAC 的安全性……271
12.5 基于 Hash 函数的 MAC：HMAC……272
12.6 基于分组密码的 MAC：DAA 和 CMAC……275
12.7 认证加密：CCM 和 GCM……277
12.8 密钥封装……282
12.9 使用 Hash 函数和 MAC 的伪随机数发生器……286
12.10 关键术语、思考题和习题……288

第 13 章 数字签名……290
13.1 数字签名概述……290
13.2 ElGamal 数字签名方案……293
13.3 Schnorr 数字签名方案……294
13.4 数字签名标准……295
13.5 椭圆曲线数字签名算法……297
13.6 RSA-PSS 数字签名算法……300
13.7 关键术语、思考题和习题……303

第五部分 互 信

第 14 章 密钥管理和分发……307
14.1 基于对称加密的对称密钥分发……307
14.2 基于非对称加密的对称密钥分发……313
14.3 公钥分发……316
14.4 X.509 证书……319
14.5 公钥基础设施……324
14.6 关键术语、思考题和习题……326

第 15 章 用户认证……330
15.1 远程用户认证原理……330
15.2 基于对称加密的远程用户

认证 ……333
15.3 Kerberos ……335
15.4 基于非对称加密的远程用户认证 ……347
15.5 联合身份管理 ……349
15.6 个人身份验证 ……353
15.7 关键术语、思考题和习题 ……357

第六部分 网络与 Internet 安全

第 16 章 网络访问控制和云安全 ……361
16.1 网络访问控制 ……361
16.2 可扩展认证协议 ……363
16.3 IEEE 802.1X 基于端口的网络访问控制 ……365
16.4 云计算 ……368
16.5 云安全所面临的威胁和对策 ……371
16.6 云中的数据保护 ……372
16.7 云安全即服务 ……375
16.8 云计算安全问题应对 ……377
16.9 关键术语、思考题和习题 ……377

第 17 章 传输层安全 ……379
17.1 Web 安全性思考 ……379
17.2 传输层安全 ……380
17.3 HTTPS ……392
17.4 SSH ……393
17.5 关键术语、思考题和习题 ……401

第 18 章 无线网络安全 ……403
18.1 无线安全 ……403
18.2 移动设备安全 ……405
18.3 IEEE 802.11 无线网络概述 ……408
18.4 IEEE 802.11I 无线局域网安全 ……412
18.5 关键术语、思考题和习题 ……422

第 19 章 电子邮件安全 ……424
19.1 因特网邮件结构 ……424
19.2 邮件格式 ……427
19.3 电子邮件威胁及综合安全 ……432
19.4 S/MIME ……434
19.5 PGP ……442
19.6 DNSSEC ……442
19.7 基于 DNS 的命名实体身份认证 ……445
19.8 发送方策略框架 ……447
19.9 DKIM ……449
19.10 基于域的消息认证、报告和一致性协议 ……452
19.11 关键术语、思考题和习题 ……456

第 20 章 IP 安全性 ……458
20.1 IP 安全性概述 ……458
20.2 IP 安全性策略 ……462
20.3 封装安全性有效载荷 ……465
20.4 结合安全性关联 ……470
20.5 因特网密钥交换 ……471
20.6 密码学套件 ……477
20.7 关键术语、思考题和习题 ……478

附录 A 用于密码学和网络安全教学的项目 ……480

附录 B Sage 示例 ……484

参考文献 ……517

在 线 部 分

第七部分 系 统 安 全

第 21 章 恶意软件
21.1 恶意软件类型
21.2 高级持续性威胁
21.3 传播-感染内容-病毒
21.4 传播-利用漏洞-蠕虫
21.5 传播-社会工程-垃圾邮件和特洛伊木马

21.6 有效载荷-系统破坏
21.7 有效载荷-攻击代理-僵尸程序
21.8 有效载荷-信息窃取-键盘记录器、网络钓鱼和间谍软件
21.9 有效载荷-隐秘行动，后门，ROOTKIT
21.10 防范措施
21.11 分布式拒绝服务攻击
21.12 参考文献
21.13 关键术语、思考题和习题

第 22 章 入侵者
22.1 入侵者
22.2 入侵检测
22.3 口令管理
22.4 参考文献
22.5 关键术语、思考题和习题

第 23 章 防火墙
23.1 防火墙的必要性
23.2 防火墙的特性与访问策略
23.3 防火墙的分类
23.4 防火墙基础
23.5 防火墙的位置与配置
23.6 参考文献
23.7 关键术语、复习题和习题

第八部分 法律与道德

第 24 章 法律与道德
24.1 网络犯罪和计算机犯罪
24.2 知识产权
24.3 隐私
23.4 道德问题
24.5 推荐阅读
24.6 参考文献
24.7 关键术语、思考题和习题

附录 C Sage 习题
附录 D 标准和标准化组织
附录 E 线性代数的基本概念
附录 F 保密和安全的测度
附录 G 简化 DES
附录 H AES 的评估准则
附录 I 简化 AES 的补充
附录 J 背包公钥算法
附录 K 数字签名算法的证明
附录 L TCP/IP 和 OSI
附录 M Java 密码函数 API
附录 N MD5 Hash 函数
附录 O 使用 ZIP 的数据压缩
附录 P PGP
附录 Q 国际参考字母表
附录 R RSA 算法的证明
附录 S 数据加密标准（DES）
附录 T Kerberos 加密技术
附录 U 生日攻击的数学基础
附录 V SHA-3 评估标准
附录 W 算法的复杂度
附录 X Radix-64 变换
附录 Y 基本率谬误

第一部分　概　　览

第 1 章　计算机与网络安全概念

第 2 章　数论基础

第1章　计算机与网络安全概念

学习目标

通过本章的学习应具备以下能力：

- 描述保密性、完整性和可用性的关键安全需求
- 描述 OSI 的 X.800 安全架构
- 讨论安全威胁和攻击的类型，举例说明作用于不同计算机和网络的安全威胁与攻击
- 解释安全设计的基本准则
- 讨论攻击面和攻击树的使用
- 列出和简要描述与密码标准相关的主要组织

本书重点讨论两大领域：一是广泛使用的密码算法与协议；二是大量依赖于密码技术的网络和 Internet 安全。

密码算法与协议又可分为 4 个主要领域：

- **对称加密**　加密任意大小的数据块或数据流的内容，包括消息、文件、加密密钥和口令。
- **非对称加密**　加密小数据块，如加密密钥或数字签名中使用的散列函数值。
- **数据完整性算法**　保护数据块（如用于告警的消息）的内容不被修改。
- **认证协议**　基于密码算法设计的认证方案，用来认证实体的身份。

网络和 Internet 安全领域涉及阻止、防止、检测和纠正信息传输中出现的安全违规行为的措施，其内容相当广泛。为使读者对本书中讨论的领域有所了解，下面先举几个有关安全违规行为的例子：

（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 无权读取该文件，但他能够监视传输过程并截获该文件。

（2）网络管理员 D 向其管辖的计算机 E 发送一条消息，指示计算机 E 更新一份权限文件，以允许一批新用户可访问 E。用户 F 截获并修改该消息，如增加或删除一些用户，然后再将消息转发给 E，而 E 误以为是管理员 D 发来的消息并按照消息的内容更新权限文件。

（3）上例中，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。人事经理向服务器系统发送消息以注销该雇员的账号。账号注销后，服务器将把通知发送到该雇员的文件中以确认注销。该雇员可以截获并延时发送人事经理所发的消息，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。

网络安全违规行为的例子无疑会有很多，但以上例子给出了网络安全所涉及的范围。

1.1　计算机安全的概念

1.1.1　计算机安全的定义

NIST 的《计算机安全手册》[NIST95] 针对计算机安全这一术语的定义如下：

计算机安全

对于一个自动化的信息系统，采取保护措施确保信息系统资源（包括硬件、软件、固件、信息/数据和通信）的完整性、可用性和保密性。

这个定义给出了计算机安全最核心的三个关键目标：

- **保密性**（Confidentiality）　该术语包含两个相关的概念：
  数据[①]保密性　确保隐私或秘密信息不向非授权者泄露，也不被非授权者使用。
  隐私性　确保个人能够控制或确定与其自身相关的哪些信息是可以被收集的、被保存的，这些信息可以由谁来公开以及向谁公开。
- **完整性**（Integrity）　该术语包含两个相关的概念：
  数据完整性　确保信息和程序只能以特定和授权的方式进行改变。
  系统完整性　确保系统以一种正常方式来执行预定的功能，免于有意或无意的非授权操纵。
- **可用性**（Availability）　确保系统能工作迅速，对授权用户不能拒绝服务。

这三个概念组成了我们常称的 CIA 三元组。这三个概念体现了数据、信息和计算服务的基本安全目标。例如，NIST 标准 FIPS 199（联邦信息和信息系统安全分类标准）将保密性、完整性和可用性作为信息和信息系统的三个安全目标。FIPS 199 从安全需求和安全缺失的角度详细说明了这三个目标：

- **保密性**　对信息的访问和公开进行授权限制，包括保护个人隐私和秘密信息。保密性缺失的定义是信息的非授权泄露。
- **完整性**　防止对信息的不恰当修改或破坏，包括确保信息的不可否认性和真实性。完整性缺失的定义是对信息的非授权修改和毁坏。
- **可用性**　确保及时和可靠地访问与使用信息。可用性的缺失是对信息和信息系统访问和使用的中断。

虽然 CIA 三元组对于安全目标的定义已很清晰，但在某些安全领域中，还需要额外的一些安全概念来呈现更完整的安全定义（参见图 1.1），其中两个常被提及的概念如下：

- **真实性**（Authenticity）　一个实体是真实性的、可被验证的和可被信任的特性；对传输信息来说，信息和信息的来源是正确的。也就是说，能够验证那个用户是否是他声称的那个人，以及系统的每个输入是否均来自可信任的信源。

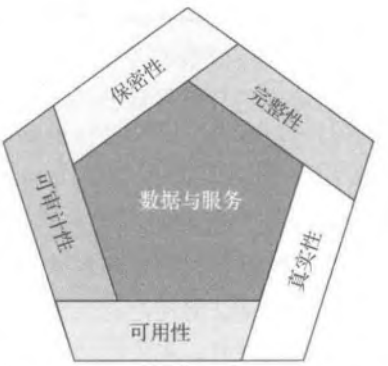


图 1.1　网络与计算机安全的基本需求

① RFC 4949 将信息定义为“事实和想法，可以用各种形式的数据进行表示”，而将数据定义为“用特定物理方式表示的信息，通常是一串有意义的符号序列，特别是可以由计算机处理和产生的信息的表示”。安全文献一般不做这种区分，本书也不做区分。

- **可追溯性（Accountability）** 这一安全目标要求实体的行为可以唯一追溯到该实体。这一属性支持不可否认性、阻止、故障隔离、入侵检测和预防、事后恢复，以及法律诉讼。因为无法得到真正安全的系统，我们必须能够追查到对安全泄露负有责任的一方。系统必须保留他们活动的记录，以允许事后的审计分析，进而跟踪安全事件或解决争执。

1.1.2 例子

下面提供一些应用例子来展示刚才列举的一些要求[①]。对于这些例子，如果发生了安全泄露事件（保密性、完整性或可用性的缺失），我们使用三个层次说明对组织和个人的影响。这些层次定义在 FIPS PUB 199 中。

- **低** 这种损失对组织的运行、组织的资产或个人的负面影响有限。有限的负面影响是指，例如，保密性、完整性或可用性的缺失可能：（1）导致执行使命的能力在一定程度上和时期内降级，期间仍能完成主要的功能，但效果稍有降低；（2）导致资产的较小损失；（3）导致很小的经济损失；（4）导致对个人的很小伤害。
- **中** 这种损失对组织的运行、组织的资产和个人有严重的负面影响。严重的负面影响是指，例如，这种损失可以：（1）导致执行使命的能力在一定程度上和时期内显著降级，期间仍能够完成主要功能，但效果会明显降低；（2）导致资产的显著损失；（3）导致显著的经济损失；（4）导致对个人的显著伤害，但不包括丧命或严重威胁生命安全的伤害。
- **高** 这种损失对组织的运行、资产和个人有严重或灾难性的负面影响。严重或灾难性的负面影响是指，例如，这种损失可以：（1）导致执行使命的能力在一定程度上和时期内严重降级，期间不能完成主要的一项或多项功能；（2）导致大部分资产的损失；（3）导致大部分经济损失；（4）导致对个人的严重或灾难性的伤害，包括丧命或严重威胁生命安全的伤害。

保密性 学生的分数信息是一种资产，学生认为分数的保密性非常重要。在美国，这种信息的发布受家庭教育权和隐私权法案（FERPA）管理。学生的分数仅能被学生自己、他们的父母以及需要这些信息来完成工作的学校雇员得到。学生的注册信息具有中等程度的保密等级。尽管注册信息仍然受 FERPA 管理，但这些信息可以以天为单位被更多人看到，与分数信息相比更少受到攻击，即使受到攻击，损失也较小。目录信息，如学生、老师、院系名单可列为低保密等级或无须保密。这些信息对公众自由开放，可以在学校网页上发布。

完整性 医院数据库中存储的病人过敏信息，可以说明完整性的几个方面。医生应能够信任这些信息是正确的且是最新的。现在假设一名有权查看和更新这些信息的雇员（如护士）有意篡改了数据并对医院造成了损失。这个数据库需要快速恢复到可以信任的状态，而且应能追溯到对这些错误负有责任的那个人。这个例子说明病人的过敏信息是对完整性要求很高的一种资产。不准确的信息会导致对病人的伤害，甚至造成病人死亡，进而使医院担负重大的责任。

对资产的完整性有中等要求的例子是 Web 站点，这些站点提供论坛供用户注册来讨论一些特定的话题。无论是注册用户还是黑客，都不能篡改某些网页或丑化网站。如果网站仅是为了娱乐，广告收入很少或没有，也不用于如科研等重大事项，那么潜在的危害就不会很严重。网站的主人可能会承受一些数据、经济和时间上的损失。

对完整性要求低的一个例子是匿名在线民意调查。许多 Web 站点，如新闻机构，为他们的用户提供几乎没有监管的这类民意调查。然而，这类民意调查的不准确性和非科学性早已为公众所了解。

① 这些例子来自普度大学信息技术安全与隐私办公室发表的安全政策文献。

可用性　一个部件或服务越关键，可用性的要求就越高。考虑一个为关键系统、应用和设备提供认证服务的系统，服务的瘫痪将导致顾客不能访问计算资源，员工不能访问他们执行重要任务所需要的资源。由于员工生产率的损失和顾客潜在的损失，使得服务的缺失转换为大量的经济损失。

对资产的可用性要求中等的一个例子是大学的公共网站；这样的网站为现有的和潜在的学生和捐助人提供信息。这种网站不能算是大学信息系统的关键部分，但其不可用仍然会给大学造成窘境。

在线电话目录查询应用可归类为可用性要求低的例子。尽管服务的临时缺失很恼人，但有其他办法获得这些信息，如纸质电话号码簿或接线员。

1.1.3　计算机安全的挑战

计算机和网络安全很吸引人，也很复杂。原因如下：

（1）安全对初学者而言并没有想象的那么简单。安全的要求看上去很直观；的确，对安全服务的大部分要求都可用含义不言自明的单词给出，如保密性、认证、不可否认或完整性。但是，满足这些要求的机制却非常复杂，理解它们需要缜密的推理。

（2）设计一个特别的安全机制或算法时，一定要考虑各种各样的潜在攻击。以与设计完全不同的方式来看问题往往可以使攻击成功，此时通常利用了设计机制中未预料到的弱点。

（3）由于第二点，提供一些特殊安全服务的方法并不直观。通常，安全机制比较复杂，从需求的陈述中并不能直观地领会到它的精心设计。只有考虑到了威胁的各个方面，精心设计的安全机制才有意义。

（4）设计好各种安全机制后，接下来是决定在哪里使用这些安全机制，包括物理位置（如网络的什么地方需要某一安全机制）和逻辑位置（如放在 TCP/IP 这种网络协议的哪一层或哪几层）。

（5）安全机制所用的算法或协议通常不止一个。这些算法和协议要求参与者拥有一些秘密信息（如加密密钥），这就带来了与秘密信息的产生、分发和保护等相关的问题。而且，通信协议本身的一些行为也可能会使安全机制的构建变得复杂。例如，若安全机制的正确执行对发送者与接收者之间的消息传送时间有限制要求，而任何协议和网络都存在不确定且不可预测的延迟，则会使得这个时间限制变得毫无意义。

（6）计算机和网络安全本质上是一场入侵者和设计者（或管理员）之间的智力战争。入侵者要努力找到漏洞，而设计者或管理员要努力封堵漏洞。入侵者的优势在于他或她仅需要找到一个弱点，而设计者必须找到并根除所有弱点来获得完全的安全。

（7）多数用户和系统管理员都有一种倾向，即只有发生了安全事件，才会意识到安全投资的收益。

（8）安全需要经常的甚至不断的监管，而这在当今短期、负荷过重的环境中难以做到。

（9）绝大多数情况下，安全仍然是一种事后措施。安全可能是在系统设计完成以后，才被考虑增加到系统中的，而不是从一开始就作为整个系统设计过程的组成部分。

（10）许多用户，甚至是安全管理员，认为强的安全不利于信息系统高效工作，有碍于用户友好操作或对信息的使用。

本书中，随着对各种安全威胁和机制讨论的展开，我们还会以多种方式遇到上面列举的各种困难。

1.2　OSI 安全架构

为了有效地评价一个机构的安全需求，以及对各种安全产品和政策进行评价和选择，负责安全的管理员需要以某种系统的方法来定义安全要求，并表征满足这些要求的措施。在集中式数据处理

环境下做到这一点已经非常困难，今天随着局域网和广域网的使用，这一问题变得更加复杂。

为此，ITU-T[①]推荐方案 X.800，即 OSI 安全框架，给出了一种系统化的定义方法[②]。对安全人员来说，OSI 安全框架是提供安全的一种组织方法。而且，因为这个框架是作为国际标准而开发的，所以许多计算机和通信的服务商已开发了与 OSI 安全框架的安全特性相适应的产品和服务。

对于我们来说，OSI 安全框架实际上对本书将要涉及的许多概念做了一个尽管抽象但非常有用的综述。OSI 安全架构主要关注安全攻击、安全机制和安全服务，它们简短地定义如下：

- **安全攻击** 任何危及信息系统安全的行为。
- **安全机制** 用来检测、阻止攻击或从攻击状态恢复到正常状态的过程（或实现该过程的设备）。
- **安全服务** 加强数据处理系统和信息传输的安全性的一种处理过程或通信服务，目的在于利用一种或多种安全机制进行反攻击。

在文献中，术语“威胁”和“攻击”的含义基本相同。表 1.1 给出了 RFC 4949（互联网安全术语表）中的定义。

表 1.1 威胁和攻击（RFC 4949）

威胁
破坏安全的潜在可能，在环境、能力、行为或事件允许的情况下，它们会破坏安全，造成危害。也就是说，威胁是脆弱性被利用而可能带来的危险。
攻击
对系统安全的攻击，它来源于一种具有智能的威胁，即有意违反安全服务和侵犯系统安全策略的（特别是方法或技术方面的）智能行为。

1.3 安全攻击

X.800 和 RFC 4949 都将攻击划分为被动攻击和主动攻击。被动攻击试图获取或利用系统的信息但并不影响系统资源，而主动攻击则试图改变系统资源或影响系统运行。

1.3.1 被动攻击

被动攻击［参见图 1.2(a)］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。信息内容的泄露和流量分析都属于被动攻击。

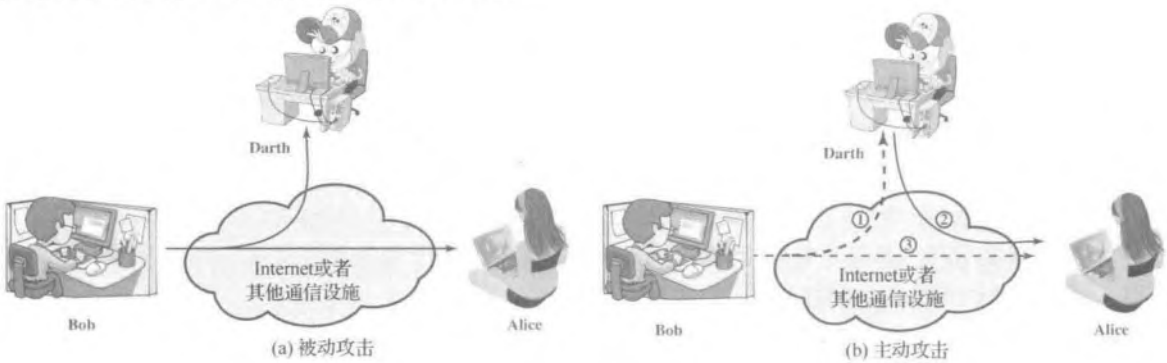


图 1.2 安全攻击

① 国际电信联盟（ITU）电信标准化组（ITU-T）是联合国资助的一个机构，主要开发与长途通信、开放系统互连（OSI）有关的各种标准，这些标准也称为建议。

② OSI 安全架构是在 OSI 协议框架范围内开发的，协议框架在附录 L 中给出。学习本章不需要理解 OSI 协议框架。

信息内容泄露攻击很容易理解。电话、电子邮件消息和传输的文件都可能含有敏感或秘密的信息，我们希望能阻止攻击者获取这些传输的内容。

第二种被动攻击是流量分析，它有些微妙。假设我们已使用方法隐藏了消息内容或其他信息流量，攻击者即使截获了消息也无法从消息中获得信息。加密是隐藏内容的常用技巧。但即使我们对消息进行了恰当的加密保护，攻击者仍具有可能获取这些消息的一些模式。攻击者可以确定通信主机的身份和位置，可以观察到传输消息的频率和长度。攻击者可以利用这些信息来判断通信的某些性质。

被动攻击由于不涉及对数据的更改，因此很难觉察。通常，信息流在被正常发送和接收时，收发双方都不会意识到有第三方已获取了消息或流量模式。但对于这样的被动攻击，通常可以使用加密方法来阻止攻击者的这种攻击。因此对于被动攻击，重点是预防而非检测。

1.3.2　主动攻击

主动攻击［参见图 1.2(b)］包括对数据流进行修改或伪造数据流，具体分为四类：伪装、重放、消息修改和拒绝服务。

伪装是指某实体假装成其他实体［参见图 1.2(b)中的路径 2］。伪装攻击通常还包含其他形式的主动攻击。例如，截获认证信息，并在认证信息完成合法验证之后进行重放，无权限的实体就可通过冒充有权限的实体获得额外的权限。

重放是指攻击者未经授权地将截获的信息再次发送（路径 1、2、3 有效）。

消息修改指未经授权地修改合法消息的一部分，或延迟消息的传输，或改变消息的顺序（路径 1、2 有效）。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对通信设施的正常使用或管理（路径 3 有效）。这种攻击可能针对的是具体的目标。比如，某实体可能会查禁所有发向某目的地（如安全审计服务）的消息。拒绝服务的另一种形式是破坏整个网络，或者使网络失效，或者使网络过载以降低其性能。

主动攻击与被动攻击相比具有一些不同的特性。被动攻击虽然难以被检测到，但可以预防。相比之下，由于物理通信设施、软件和网络自身潜在弱点的多样性，主动攻击难以绝对预防，所以对于主动攻击，重点在于检测并从攻击造成的破坏或延迟中恢复过来。如果对主动攻击的检测有威慑效果，那么其在某种程度上也可以阻止主动攻击。

1.4　安全服务

X.800 将安全服务定义为：在通信开放系统中，为系统或数据传输提供足够安全的协议层服务。RFC 4949 中给出的定义可能更清晰：安全服务是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 5 类共 14 个特定服务（参见表 1.2）。下面逐类进行讨论[①]。

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的多个方面。认证有时既用来指身份验证，又用来指本章中列出的各种数据的完整性功能。这里使用的术语与 X.800 和 RFC 4949 中的一致。

表 1.2 安全服务

认 证	数据完整性
保证通信的实体是它所声称的实体	保证收到的数据的确是授权实体发出的数据（即未修改、插入、删除或重播）
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、删除或重播，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
访问控制	**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访问资源，在什么条件下可以访问，这些访问的资源可用于做什么）	提供一次连接中传输的单个数据块内用户数据的指定部分的完整性，并判断指定部分是否有修改、插入、删除或重播
数据保密性	**无连接完整性**
保护数据免于非授权泄露	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，提供有限的重播检测
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指定域提供完整性保护；判断指定域是否被修改
无连接保密性	**不可否认性**
保护单个数据块中的所有用户数据	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	**源不可否认性**
对一次连接或单个数据块中指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

1.4.1 认证

认证服务用以保证通信的真实性。在单条消息的情况中，如一条警告或报警信号，认证服务功能向接收方保证消息确实来自其所声称的发送方。对于正在进行的通信，如终端和主机连接，会涉及发送方和接收方两个主体。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的干扰：这种干扰是指，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。

X.800 还定义了以下两个特殊的认证服务：

- **对等实体认证** 为连接中的对等实体提供身份确认。不同系统中的两个实体执行相同的协议时，考虑他们是对等的，如位于两个通信系统中的两个 TCP 模块。对等实体认证用于连接的建立或数据传输阶段。这个服务期望提供这样的保证：一个实体没有试图进行伪装或对以前的连接进行非授权重放。
- **数据源认证** 为数据的来源提供确认，对数据的复制或修改并不提供保护。这种服务支持电子邮件这样的应用，在这种应用的背景下，通信实体在通信前未预先进行交互。

1.4.2 访问控制

在网络安全中，访问控制是限制和控制那些通过通信连接对主机与应用进行访问的一种能力。为此，每个试图获得访问控制的实体必须被识别或认证后，才能获取相应的访问权限。

1.4.3 数据保密性

保密性是指防止传输的数据遭到被动攻击。关于数据传输，其保护可以分成不同的层级。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防止在 TCP 连接上传输的任何用户数据的泄露。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。与广泛的方法相比，这种细化用处要少，实现起来更复杂，成本更高。

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地址与目的地址、频率、长度或通信设施上的其他流量特征。

1.4.4　数据完整性

与保密性一样，完整性可应用于消息流、单条消息或消息的指定部分。同样，最有用也最直接的方法是对整个数据流提供保护。

用于处理消息流、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，保证消息未被复制、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需要处理消息流的修改和拒绝服务两个问题。另一方面，无连接的完整性服务仅处理单条消息，而不管大量的上下文信息，因此通常仅防止对消息的修改。

完整性服务还可分为可恢复的服务和无恢复的服务。因为完整性服务和主动攻击有关，我们更关心检测而非阻止攻击。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。或者，也可使用一些机制来恢复数据完整性，我们将在后文中看到。通常，自动恢复机制是一种更有吸引力的选择。

1.4.5　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，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，消息接收后，发送方能证明消息确实由声称的接收方收到。

1.4.6　可用性服务

X.800 和 RFC 4949 都将可用性定义为：根据系统的性能说明，系统资源可被授权实体请求访问或使用（即当用户请求服务时，若系统能够提供符合系统设计的这些服务，则系统是可用的）。很多攻击会导致可用性的损失或降级。对于某些攻击，可以通过一些自动防御措施，如认证、加密来防止。而针对其他的一些攻击，则需要使用一些物理措施来阻止或恢复分布式系统中被破坏了可用性的那部分功能。

X.800 将可用性视为与多种安全服务相关的性质。但是，单独说明可用性服务颇有意义。可用性服务确保系统的可用性。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，因此依赖于访问控制服务和其他安全服务。

1.5　安全机制

表 1.3 列出了 X.800 中定义的安全机制。由表可知，这些安全机制分为两类：一类在特定的协议层实现，如 TCP 或应用层协议；另一类不属于任何协议层或安全服务。本书将在后文中详述这些机制，在此暂时只讨论加密机制的定义。X.800 对可逆和不可逆加密机制进行了区分。可逆加密机制是一种单纯的加密算法，数据可以加密和解密。不可逆加密机制包括用于数字签名和消息认证应用中的散列算法与消息认证码。

根据 X.800 中的定义，表 1.4 给出了安全服务和安全机制的关系。

表 1.3 安全机制（X.800）

特定安全机制	普遍的安全机制
可以并入适当的协议层以提供一些 OSI 安全服务	不局限于任何特定 OSI 安全服务或协议层的机制
加密 运用数学算法将数据转换成不可知的形式。数据的变换和还原依赖于算法和零个或多个加密密钥	**可信功能** 据某些标准被认为是正确的功能（如根据安全策略所建立的标准）
数字签名 附加于数据单元之后的一种数据，它是对数据单元的密码变换，以使得（如接收方）可证明数据源和完整性，并防止伪造	**安全标签** 资源（可能是数据单元）的标志，命名或指定该资源的安全属性
访问控制 对资源行使访问控制的各种机制	**事件检测** 检测与安全相关的事件
数据完整性 用于保证数据单元或数据单元流的完整性的各种机制	**安全审计跟踪** 收集可用于安全审计的数据，它是对系统记录和行为的独立回顾与核查
认证交换 通过信息交换来保证实体身份的各种机制	**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
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	

表 1.4 安全服务与机制间的联系

	机 制							
服 务	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			

1.6 基本安全设计准则

虽然已经有了多年的研究和开发，但目前仍然没有一种安全设计或实现技术能够系统地摒除安全漏洞、阻止所有的非授权行为。在没有这种万无一失技术的情况下，能有一些可被广泛认可的设计准则来指导保护机制的开发也是有益的。美国国家信息保障/空间防御学术中心（National Centers of Academic Excellence in Information Assurance/Cyber Defense）与美国国家安全局、美国国土安全部联合倡议了如下安全设计准则[NCAE13]：

- 机制的经济性（Economy of mechanism）
- 故障安全默认（Fail-safe defaults）
- 完整的监察（Complete mediation）

- 开放的设计（Open design）
- 权限分离（Separation of privilege）
- 最小权限（Least privilege）
- 最小共同机制（Least common mechanism）
- 心理接受度（Psychological acceptability）
- 隔离（Isolation）
- 密封（Encapsulation）
- 模块化（Modularity）
- 分层（Layering）
- 最小意外（Least astonishment）

上面列出的前 8 个准则最早在文献[SALT75]中提出，并经过了时间的检验。本节逐一讨论这些准则。

机制的经济性准则是指嵌入在硬件和软件中的安全机制应设计得尽量简单、短小。因为相对简单、短小的设计更易被测试和验证。如果设计很复杂，攻击者就有更多机会发现可以利用的细微弱点，而这些弱点往往很难提前发现。机制越复杂，就越有可能存在瑕疵。简单的机制可以减少漏洞，降低维护的难度。进一步说，因为简单机制中的配置管理问题也得到简化，更新或替换安全机制变得不那么密集。但在实践中，这条准则可能是最难遵循的。在硬件和软件中，存在着对新特性的持续需求，这就使得安全设计任务复杂化。因此，最简单的是，在系统设计阶段，时刻提醒自己这条准则，避免不必要的复杂性。

故障安全默认准则意味着访问决策应基于允许（permission）而非拒绝（exclusion），也就是说，默认情况下是无访问权限，保护方案核实是否符合允许访问的条件。与默认情况下允许访问的故障机制相比，这种方式表现得更好，在安全机制的设计或实现中出现错误时，往往应拒绝访问许可，进而可以快速检测到错误的情况。相反，如果允许访问，在正常使用中可能很长时间都不会注意到这个失败。大多数文件访问系统及几乎所有客户端/服务器系统上受保护的服务都有故障安全默认值。

完整的监察准则意味着必须检查访问控制机制中的每个访问，系统不能依赖从缓存中检索到的访问决策。若系统被设计为支持长时间连续运行，则这条准则要求，访问决策结果被缓存下来以供将来使用时，需要仔细考虑权限的变化如何传播到缓存这种本地存储器中。文件访问系统就是一个符合该准则的系统的例子。但通常一旦用户打开了文件，系统就不会再检查用户对该文件的访问许可是否改变。为了完全实现完整的监察准则，每当用户读取文件中的字段或记录，或访问数据库中的数据项时，系统都须执行访问控制。这种资源密集监察型方法很少使用。

开放的设计准则指出安全机制的设计应是开放的，而不是保密的。例如，尽管密钥必须保密，但加密算法应该公开。这样，这个算法就可以受到很多专家的审查，从而使用户更加信任该算法的安全性。这也是 NIST 在标准化加密和散列算法时采用的方法，并使得 NIST 批准的算法被广泛采用。

权限分离准则在文献[SALT75]中定义为需要多个权限属性来访问一个受限资源时的准则。多因素用户认证就是这样的一个较好例子，它需要使用多种技术来验证用户，如口令和智能卡。权限分离准则现在也适用于任何需要将程序划分多个部分，每个部分仅被授予执行特定任务所需的特定特权，因此可用于减轻计算机受到攻击时的潜在损害。例如，日常的界面操作运行在较低的权限下，而将较高权限的操作放在另外一个进程中，并根据完成任务的需要以高权限来运行这个进程。

最小权限准则意味着系统的每个进程和用户应该以执行该任务所需的最小权限集来进行操作。一个很好的例子就是基于角色的访问控制，系统的安全策略应能识别和定义用户或进程的各种角色，每

个角色都只赋予完成其功能所需的最低权限，而每个权限只被允许访问特定的资源（如读写特定的文件或目录，连接特定的主机或端口）。除非显式地获得了权限，否则用户或进程都不能访问那些受保护的资源。更一般地说，任何访问控制系统应只允许授予用户其可被授予的权限。此外，最小权限准则还有时间方面的考虑。例如，拥有特别权限的系统程序或系统管理员应该只在需要时才拥有这些权限，而在执行普通任务时应收回这些权限，一直保留这些权限只会增大发生事故的概率。

最小共同机制准则意味着应把不同用户共享的功能设计得最小化，以便提供共同的安全性。这个准则有助于减少非预期的通信路径的数量，并减少所有用户依赖的硬件和软件数量，进而更容易验证是否存在任何不期望的安全隐患。

心理接受度准则说明安全机制在满足授权访问的用户需求的同时，不应过度干预用户的工作。如果安全机制妨碍了资源的使用或访问，用户很可能会倾向于把这些机制关闭。安全机制应尽可能地对系统用户透明，或尽量带来最小的妨碍。为了不造成侵扰或成为负担，安全程序必须反映用户的心理保护模式。如果保护程序对用户没有意义，或用户将其对保护的理解转换为不同的协议，就可能犯错。

隔离准则应用于以下三种情况。第一，公共访问系统应与关键资源（数据、进程等）隔离，以防止其泄露或被篡改。在信息的敏感性或关键性很高的情况下，组织可能希望限制存储该数据的系统的数量，并在物理上或逻辑上隔离它们。物理隔离可以包括确保组织中的公共访问信息资源和组织的关键信息之间不存在物理连接。实现逻辑隔离时，应在公共系统和负责保护关键资源的安全系统之间建立安全服务与机制层。第二，除非明确需要这么做，否则各个用户的进程和文件应彼此隔离。所有现代操作系统都提供了这种隔离，以便各个用户具有单独的、隔离的处理器空间、存储器空间和文件空间，防止未授权访问。最后，安全机制也应被隔离，以防止他人访问到这些机制。例如，逻辑访问控制可以提供将密码软件与主机系统的其他部分隔离，并用于保护密码软件免受篡改、密钥免于被更换或泄露的方法。

密封准则可视为基于面向对象的特定形式的隔离。通过将过程和数据对象的集合封装在其自身的作用域中来提供保护，使得数据对象的内部结构仅可由受保护子系统的过程访问，并且过程可仅在指定的作用域入口被调用。

模块化准则在安全领域中，既指将安全功能作为单独的模块开发，也指用于机制设计和实现的模块化架构。相比于使用独立的安全模块，这个设计准则是将公共的安全函数与服务，如密码服务，作为公共模块提供。例如，很多协议和应用都使用了密码功能，开发可被多个协议和应用调用的公共密码模块则是更安全的设计，而不是在每个协议或应用中都去实现这样的功能，这样就可将注意力集中在单个密码模块的安全设计和实现上，并在其中加入保护模块保护其免受篡改。使用模块化架构的机制设计与实现中，每个安全机制都能支持移植到新的技术或升级某些特性，而无须重新设计整个系统。安全机制的设计应该是模块化的，以便可以独立升级机制的各个部分，而不需要修改整个系统。

分层准则是指使用多个叠加的保护方法来保护信息系统的人员、技术和操作。通过使用多层叠加的保护方法，当其中的任何一层失败时，都不会使得系统是不受保护的。我们将在本书中看到，分层方法常用于在攻击者与受保护的信息或服务之间设置多个障碍。这种技术通常称为深度防御（defense of depth）。

最小意外准则是指程序或用户接口应让用户感到，对意外事故的处理响应最小化。例如，授权机制应该对用户足够透明，以便用户能直观地理解安全机制是如何满足安全目标的。

1.7 攻击面与攻击树

1.3 节简要介绍了计算机和网络系统面临的安全威胁与攻击，22.1 节将会详细介绍攻击的性质

和具有安全威胁的攻击者的类型。本节详细介绍在评估和分类安全威胁时很有用的两个概念：攻击面和攻击树。

1.7.1　攻击面

攻击面由系统中一系列可访问且可利用的漏洞组成[MANA11, HOWA03]，下面是常见的攻击面的例子：

- 向外部 Web 和其他服务器开放的端口，以及监听这些端口的代码。
- 防火墙内部可用的服务。
- 处理传入的数据、E-mail、XML 文件、Office 文档和企业特定的自定义数据交换格式的代码。
- 接口、SQL 和 Web 表单。
- 易受社会工程攻击的可访问敏感信息的员工。

攻击面可分为以下几类：

- **网络攻击面**　此类攻击面涉及的是企业网络、广域网或互联网。包含其中的是网络协议的漏洞，如用来进行拒绝服务攻击、中断通信链路、各种形式的入侵攻击的漏洞。
- **软件攻击面**　此类攻击面涉及的是应用程序、工具包或操作系统代码。其中的一个重点是 Web 服务器软件。
- **人类攻击面**　此类攻击面涉及的是由系统人员或外部人员造成的漏洞，如社会工程学、人为错误和被信任的内部人员。

评估安全威胁的规模和严重性时，攻击面分析是很有用的技术。对漏洞的系统分析能使开发人员和安全分析师知道系统中的哪些地方需要安全机制。一旦明确一个攻击面，设计者就可想方设法使这个攻击面变小，进而使攻击者的任务更加困难。攻击面也提供设置测试、加强安全措施和修改服务或应用的优先权的依据。

如图 1.3 所示，分层或深度防御与减少攻击面，可以在减少安全风险时相互补充。

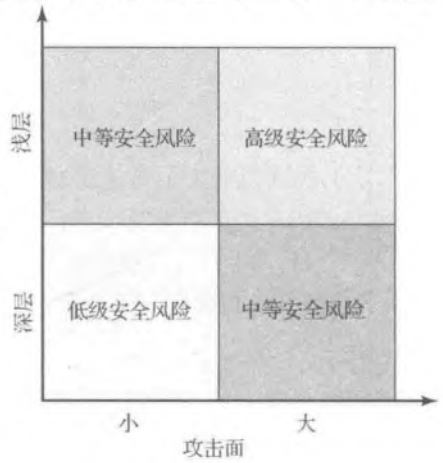


图 1.3　深度防御和攻击面

1.7.2　攻击树

攻击树是采用分支化、层次化表示来利用安全漏洞的可能技术集合的数据结构[MAUW05, MOOR01, SCHN99]。攻击的目的——造成安全事件，是攻击树的根节点，而那些攻击者为达到这一目的所采用的方法，被迭代增加地表示为攻击树上的分支和子节点。每个子节点定义一个子目标，而每个子目标又可能有其更细化的子目标。每个从根节点出发的路径上的末端节点，即叶子节点，表示发动攻击的不同方式。每个非叶子节点都是“与节点”（AND-node）或“或节点”（OR-node）。为了实现“与节点”的目标，该节点的所有子节点代表的子目标都必须要先完成；而为了实现“或节点”的目标，则需要至少完成一个子目标。每条分支可以标记不同的值来表示攻击的难度、成本或其他攻击属性，用以对不同的可选攻击方式进行比较。

使用攻击树的动机是为了更有效地利用关于攻击模式的可用信息。可以使用诸如 CERT 组织发布的安全建议，来构建一套关于一般攻击策略和特定攻击模式的知识体系。安全分析师可以使用攻

击树的这种结构化方式来归档安全攻击，进而揭示其中的关键漏洞。攻击树还可用来指导系统和应用的设计，以及对抗策略的选择和强度。

基于文献[DIMI07]的图 1.4 是一个网上银行认证程序的攻击树例子。树的根节点是攻击者的目标——攻击用户的账户。树中阴影框部分是树的叶子节点，代表组成攻击的事件。需要注意的是，在这棵攻击树中，除叶子节点外的节点都是“或节点”。生成此树的攻击分析考虑了认证中涉及的三个部分：

- **用户终端和用户（UT/U）** 这些攻击针对用户设备，包含可能涉及的令牌，如智能卡或其他密码生成器，以及用户的操作。
- **通信信道（CC）** 这种类型的攻击着重于攻击通信链路。
- **网上银行服务器（IBS）** 这些类型的攻击是对托管网上银行应用程序的服务器的离线攻击。



图 1.4 针对互联网银行认证的攻击树

由此可以确定五个总体攻击策略，每个策略利用三部分中的一部分或多部分。下面是这五种攻击策略：

- **攻击用户凭据** 这个策略可用来对付攻击面的很多元素。有很多程序化的攻击方式，例如监视用户的操作以获取用户的 PIN 或其他凭据，或窃取用户的令牌或手写便条。攻击者也可使用一系列令牌攻击工具来破坏令牌信息，如入侵智能卡或使用暴力攻击的方式猜测 PIN。

另一种可能的策略是，植入恶意软件来破坏用户的登录或获取密码。攻击者还可尝试通过信道（嗅探）来获取认证信息。最后，如图 1.4 所示，攻击者也可使用各种手段通过通信攻击目标用户。

- **命令注入**　在这种攻击中，攻击者可以监听 UT 和 IBS 间的通信，使用各种方案来模拟合法用户，从而获得对银行系统的访问。
- **用户凭据猜测**　据文献[HILT06]中的报道，通过发送随机的用户名和密码来暴力攻击某些网上银行的认证方案是可行的。这种攻击方法在分布式僵尸网络中的那些被控制的计算机上，执行自动计算用户名或用户密码的程序。
- **安全策略违规**　例如，银行员工违反银行安全策略，使用了弱的访问控制和日志机制，可能会导致内部安全事件，导致客户账户的泄露。
- **利用已被认证的会话**　这类攻击诱使或强迫用户使用预设的会话 ID 来连接 IBS，一旦服务器通过了用户的身份认证，攻击者就能利用这个已知的会话 ID，假冒该用户的身份给 IBS 发送数据包。

图 1.4 直观地给出了一个网上银行认证程序中的各种攻击方式。以这棵为起点，安全分析师可以评估每种攻击的风险，并利用前一节的设计准则，设计出一个周全的安全方案。文献[DIMO07]详细描述了这个设计的结果。

1.8　网络安全模型

图 1.5 给出了我们将要讨论的网络安全模型。消息通过 Internet 服务从一方传送到另一方，而作为消息交换的主体，双方都必须和对方合作以保证交换的可靠。双方信息通过路由从源地址到目的地址，并协作使用通信协议（如 TCP/IP），从而建立起一个逻辑信息通道。

需要保护传输信息，防止攻击者对信息机密性、真实性等进行攻击时，安全技术开始发挥作用。所有提供安全的技术都包含以下两部分：

- 对发送信息的安全变换。如加密消息，将消息打乱，使其变得不可读；基于消息内容，附加一段基于消息内容的编码，用来验证发送者的身份。
- 被两个主体共享且不被攻击者知道的一些机密信息。如在传输之前用于加密消息、收到消息后用于解密消息的密钥[①]。

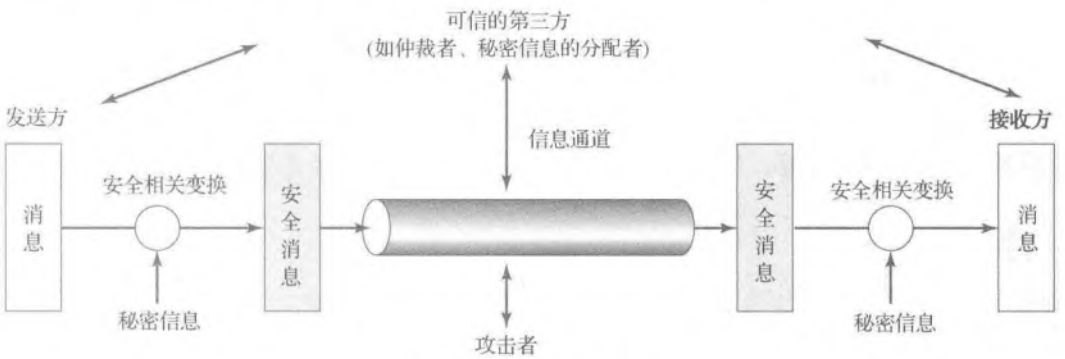


图 1.5　网络安全模型

① 第二部分讨论的对称加密的加密形式中，双方必须拥有相同的机密信息。

为了实现安全传输，可能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如下 4 个方面的内容：

1. 设计一个算法，它执行与安全相关的变换。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。
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。

本书的第一部分至第五部分讨论的是安全机制和服务，它们遵循图 1.5 中的模型，但存在不完全符合该模型的其他安全情形，本书也会讨论这些内容，它们的一般模型如图 1.6 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，黑客试图渗入到可通过网络访问的系统，他可能没有恶意，只是对闯入或进入计算机系统感到满足。入侵者可能是一名不如意的雇员，想进行破坏，或是一名罪犯，想利用计算机获利（如获取信用卡号或进行非法的资金转账）。

另一种有害访问是在计算机系统中加入程序，利用系统的弱点来影响应用程序和实用功能程序，如编辑程序和编译程序。这种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。



图 1.6 网络访问安全模型

病毒和蠕虫是两种软件攻击，如果磁盘上的应用软件中隐藏有害程序，那么这些攻击可通过磁盘进入系统。这些攻击也可通过网络进入系统。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（参见图 1.6）。第一类称为门卫功能，它包括基于口令的登录过程（该过程只允许授权用户的访问）和监控程序（负责检测和阻止蠕虫、病毒及其他类似的攻击）。一旦非法用户或软件获得了访问权，那么由各种内部控制程序组成的第二道防线就会监视其活动、分析存储的信息，以便检测非法入侵者。这些问题将会在第六部分讨论。

1.9 标准

本书中描述的许多安全技术和应用程序已经成为了标准。另外，目前人们还制定了涵盖管理实践、安全机制和服务的总体架构的标准。本书描述了正在使用的或正在开发的加密和网络安全各方面的重要标准。很多组织都参与了这些标准的制定和推广，其中（目前）最重要的组织如下：

- **美国国家标准与技术研究所（NIST）** NIST 是一个美国联邦机构，负责处理与美国政府使

用和促进美国私营部门创新有关的测量科学、标准和技术。除了在本国范围内的影响力外，NIST 联邦信息处理标准（FIPS）和特别出版物（SP）还具有全球影响力。

- **互联网协会（Internet Society，ISOC）** ISOC 是一个具有全球组织和个人会员资格的专业会员协会。它在解决未来互联网将面临的问题方面发挥领导作用，并是负责互联网基础设施标准的小组的组织机构[其中包括互联网工程任务组(IETF)和互联网架构委员会(IAB)]。这些组织开发了互联网标准及相关规范，所有这些都会作为征求意见（RFC）发布。
- **ITU-T** 国际电信联盟（ITU）是联合国系统（各国政府和私营部门协调全球电信网络和服务的一个组织）内的一个国际组织，国际电联电信标准化部门（ITU-T）是国际电联的三个部门之一，其任务是制定涵盖所有电信领域的技术标准，ITU-T 的标准称为建议书（Recommendation）。
- **ISO** 国际标准化组织（ISO[①]）是由 140 多个国家的国家标准机构组成的全球联盟（每个国家一个）。ISO 是一个非政府组织，致力于促进标准化和相关活动的发展，促进商品和服务的国际交流，并以此促进智力、科学、技术和经济活动领域的合作。ISO 的工作产生了很多已发布为国际标准的国际协议。

有关这些组织的详细讨论，请参考附录 D。

1.10　关键术语、思考题和习题

关键术语

访问控制	数据完整性	伪装	安全攻击
主动攻击	拒绝服务攻击	非否认性	安全机制
认证	加密	OSI 安全架构	安全服务
真实性	完整性	被动攻击	流量分析
数据机密性	入侵者	重放	

思考题

1.1　什么是 OSI 安全架构？

1.2　被动安全威胁与主动安全威胁有何区别？

1.3　列出并简要定义被动安全攻击与主动安全攻击的种类。

1.4　列出并简要定义安全服务的种类。

1.5　列出并简要定义安全机制的种类。

1.6　列出并简要定义基本的安全设计准则。

1.7　解释攻击面与攻击树的区别。

习题

1.1　考虑一台自动取款机（ATM），用户通过提供个人认证码（PIN）和银行卡来访问其账户。请给出该系统相关机密性、完整性等所需的安全需求，并针对每个安全需求给出重要性程度。

1.2　以电话交互系统为例重做习题 1.1。电话交互系统根据呼叫者呼叫的电话号码通过交互网络路由呼叫。

① ISO 不是缩写（若为缩写，应写为 IOS），而是一个从希腊语延伸过来的单词，意思是等于。

1.3 考虑一个为各种组织生成文档的桌面发布系统：

（a）请给出一种发布类型的例子，要求存储数据的机密性是最重要的。

（b）请给出一种发布类型的例子，要求存储数据的完整性是最重要的。

（c）请给出一个例子，此时对系统可用性的要求是最重要的。

1.4 对于下面的资产，发生保密性、可用性和完整性损失时，分别为其产生的影响划分低、中和高等级，并陈述理由。

（a）在自己的网络服务器上管理公开信息的组织。

（b）执法机构管理极端敏感的调查信息。

（c）金融机构管理例行的行政信息（非隐私信息）。

（d）合同签订机构的大单收购信息系统，包含敏感、预投标阶段的合同信息和例行的行政信息。请分别评价这两份信息资产的影响情况及整个信息系统的影响情况。

（e）发电厂包含有 SCADA（监管控制和数据获取）系统，它负责为军事设施提供电力分配。SCADA 系统中包含有实时传感器数据和例行的行政信息。请分别评价这两份信息资产的影响情况及整个信息系统的影响情况。

1.5 类似于表 1.4，画出一个矩阵来表明安全服务和安全攻击的关系。

1.6 类似于表 1.4，画出一个矩阵来表明安全机制和安全攻击的关系。

1.7 构建一个获得物理安全内容的访问权限的攻击树。

1.8 考虑这样一个驻于两栋大楼中的公司：公司总部在一栋楼中，网络和计算机机房在另一栋楼中。这两栋楼的外部有围墙，进入公司的唯一入口需要穿过围墙。除了围墙，公司大门还有保卫。公司的网络被划分为两个子网，一个是公司总部子网，另一个是网络服务子网。因特网用户通过防火墙可接入到公司的网站。拨号上网的用户可访问网络服务子网。设计一种攻击树，其中根节点表示公司专有秘密的泄露，攻击方式包括物理攻击、社会工程和技术攻击，攻击树包含“与节点”和“或节点”，且至少包含 15 个叶子节点。

1.9 阅读网站 WilliamStallings.com/Cryptography 上关于本章的“推荐读物”部分的所有经典论文。这些文章可以在 box.com/Crypto7e 上获得。写一篇 500～1000 字的论文（或 8～12 页的 PowerPoint 幻灯片），总结这些论文中出现的关键概念，特别是在大多数或全部论文中都共有的概念。

第2章 数论基础

学习目标

通过本章的学习应具备以下能力：

- 理解整除性和带余除法的概念
- 理解如何利用欧几里得算法找到最大公约数
- 了解模运算的概念
- 解释扩展欧几里得算法的用法
- 讨论素数的主要概念
- 理解费马定理
- 理解欧拉定理
- 掌握欧拉函数
- 理解素性测试的要点
- 解释中国剩余定理
- 掌握离散对数

在加密算法中，数论的应用非常广泛。为了让读者理解数论在密码学中的应用，本章提供覆盖广度和深度都足够的数论相关知识。熟悉这些内容的读者可略过本章。

前三节介绍数论中有限域的基本概念，包括整除、欧几里得算法和模运算。读者既可现在学习这些章节，也可等到第 5 章的有限域部分时学习。

2.4 节至 2.8 节讨论有关素数和离散对数的相关数论知识。这些知识是设计非对称（公钥）加密算法的基础。读者既可以现在学习这些内容，也可等到第三部分时学习。

数论中的概念和技术都很抽象，没有例子的话，很难直观地去理解它。因此，本章包含了大量的例子，每个例都用带阴影的方框突出显示。

2.1 整除性和带余除法

2.1.1 整除性

设 a, b, m 均为整数，若存在某个 m 使得 $a = mb$ 成立，则称非零数 b 整除 a。换言之，若 b 除 a 没有余数，则认为 b 整除 a。b 除 a 通常用 $b \mid a$ 来表示。同时，若 $b \mid a$，我们说 b 是 a 的一个**因子**。

24 的正因子有 1, 2, 3, 4, 6, 8, 12 和 24

$13 \mid 182$; $-5 \mid 30$; $17 \mid 289$; $-3 \mid 33$; $17 \mid 0$

随后，我们将需要用到一些简单的整数整除的性质，如下所示：

- 若 $a \mid 1$，则 $a = \pm 1$。

- 若 $a \mid b$ 且 $b \mid a$，则 $a = \pm b$。
- 任何不等于零的数整除 0。
- 若 $a \mid b$ 且 $b \mid c$，则 $a \mid c$。

11|66 且 66|198 ⇒ 11|198

- 对任意整数 m, n，若 $b \mid g$ 且 $b \mid h$，则可得出 $b \mid (mg + nh)$。

之所以得出最后一条结论，是因为

- 若 $b \mid g$，则存在 g_1，使得 g 可以表示为 $g = b \times g_1$。
- 右 $b \mid h$，则存在 h_1，使得 h 可以表示为 $h = b \times h_1$。

所以

$$mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$$

得出 b 整除 $mg + nh$。

$b = 7; g = 14; h = 63; m = 3; n = 2$ 7 | 14 且 7 | 63 要证明 7 | (3×14+2×63) 我们有(3×14 + 2×63)= 7(3×2 + 2×9) 显然有 7 | (7(3×2+2×9))

2.1.2 带余除法

对给定的任意一个正整数 n 和任意非负整数 a，若用 n 除 a，得到整数商 q 和整数余数 r，则满足以下关系式：

$$a = qn + r, \quad 0 \leqslant r < n; q = \lfloor a \mid n \rfloor \tag{2.1}$$

式中，$\lfloor x \rfloor$表示小于等于 x 的最大整数。式（2.1）称为带余除法[①]。

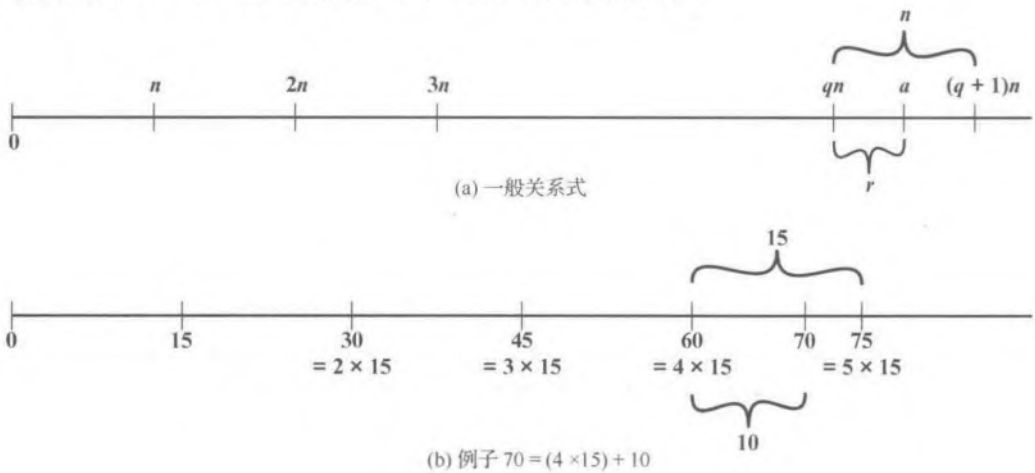


图 2.1 关系式 $a = qn + r$，$0 \leqslant r < n$

① 式(2.1)的表示更像一个定理而非算法，但依照传统，仍被称为带余除法。

图 2.1(a)表明，对于给定的任意 a 和正数 n，总可以找到满足上述关系的 q 和 r。在数轴上表示整数时，a 一定会落在轴上的某处（正数 a 的情况如图 2.1 所示，类似地，a 也可为负数）。从 0 开始，依次为 n, $2n$，直到 qn，使得 $qn \leq a$ 且 $(q+1)n > a$，其中 qn 到 a 的距离为 r。这样，就得到了唯一的 q 和 r。余数 r 通常称为剩余数。

$a = 11; n = 7; 11 = 1 \times 7 + 4; r = 4, q = 1$
$a = -11; n = 7; -11 = (-2) \times 7 + 3; r = 3, q = -2$
图 2.1(b)提供另外一个例子

2.2 欧几里得算法

欧几里得算法是数论中的一个最基本的技巧，它可以简单地求出两个正整数的最大公因子。首先，我们需要一个简单的定义：两个整数是互素的，当且仅当它们只有一个正整数公因子 1。

2.2.1 最大公因子

前面讲过，对于整数 a, b, m，若满足 $a = mb$，则称非零整数 b 是 a 的一个因子。我们用 $\gcd(a, b)$来表示 a 的最大公因子。a 和 b 的最大公因子是能同时整除 a 和 b 的最大整数。另外，定义 $\gcd(0, 0) = 0$。

更正式的描述是，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、b 的因子都是 c 的一个因子。

另一个等效的定义如下：

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

因为要求最大公因子必须是正数，所以有 $\gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$，一般来说，$\gcd(a, b) = \gcd(|a|, |b|)$。

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

同样，因为零可被所有非零整数整除，所以有 $\gcd(a, 0) = |a|$。

我们说整数 a, b 互素，当且仅当它们只有一个正整数公因子 1。也就是说，若 $\gcd(a, b) = 1$，则 a、b 互质。

8 和 15 互质，因为 8 的正数因子为 1, 2, 4 和 8，而 15 的正数因子为 1, 3, 5 和 15，所以 1 是它们唯一的公因子

2.2.2 求最大公因子

我们将描述一个由欧几里得发明的算法，它可以简单地求出两个整数的最大公因子（参见图 2.2）。这个算法在密码学中有着广泛的意义。该算法的阐述可分为以下几点：

（1）假设我们需要求出整数 a 和 b 的最大公因子 d；因为 $\gcd(|a|, |b|) = \gcd(a, b)$，所以这里可以假定 $a \geq b > 0$。

（2）使用带余除法，b 除 a 可以表示为

$$a = q_1 b + r_1, \qquad 0 \leqslant r_1 < b \tag{2.2}$$

（3）首先考虑到 $r_1 = 0$ 的情况。由此可知 b 整除 a，且 a 和 b 的公因子中不存在比 b 更大的数。所以有 $d = \gcd(a,b) = b$ 。

（4）式（2.2）的另外一种情况是 $r_1 \neq 0$ 。在这种情况下，可知一定有 $d \mid r_1$ 。因为由因子的基本性质可知：存在 $d \mid a$ 和 $d \mid b$ ，那么一定有 $d \mid (a - q_1 b)$ ，即 $d \mid r_1$ 。

（5）在开始应用欧几里得算法之前，需要知道：$\gcd(b, r_1)$的值是什么？我们知道 $d \mid b$ 和 $d \mid r_1$ 。那么，对于任意一个 b 和 r_1 的公因子 c，有 $c \mid (q_1 b + r_1) = a$ 。另外，因为 c 整除 a 和 b，所以一定有 $c \leqslant d$ ，其中 d 是 a 和 b 的最大公因子，因此有 $d = \gcd(b, r_1)$ 。

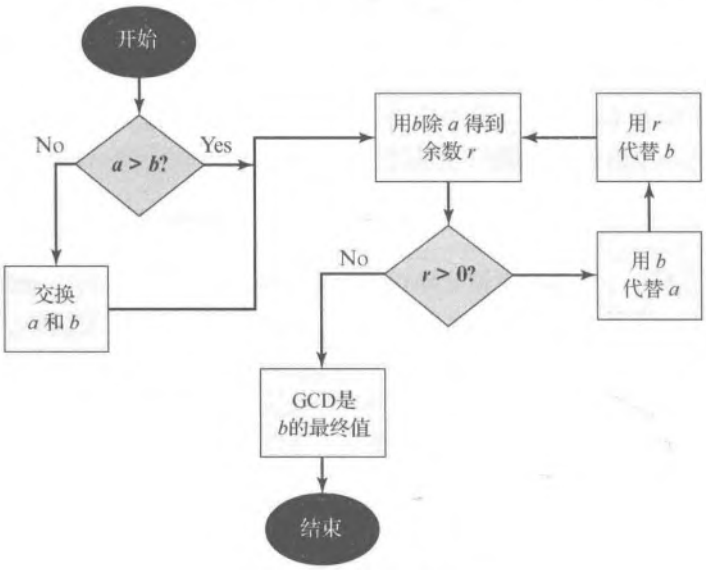


图 2.2 欧几里得算法

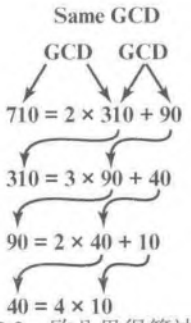


图 2.3 欧几里得算法示例 gcd(710, 310)

现在回到式（2.2）。假设 $r_1 \neq 0$，因为 $b > r_1$，可以用 r_1 除 b，然后运用带余除法得

$$b = q_2 r_1 + r_2, \qquad 0 \leqslant r_2 < r_1$$

如上，若 $r_2 = 0$，则 $d = r_1$，若 $r_2 \neq 0$，则 $d = \gcd(r_1, r_2)$ 。可以注意到，在计算的过程中，余数会形成递减的一系列非负值，并且余数为零时，算法一定会终止。例如，假设在第$(n+1)$阶段有 r_{n-1} 除以 r_n，结果如下：

$$\left.\begin{array}{ll} a = q_1 b + r_1 & 0 < r_1 < b \\ b = q_2 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_3 r_2 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = q_{n+1} r_n + 0 & \\ d = \gcd(a, b) = r_n & \end{array}\right\} \tag{2.3}$$

每次迭代，我们都进行一次 $d = \gcd(r_i, r_{i+1})$ 运算，直到 $d = \gcd(r_n, 0) = r_n$ 。因此，我们可以通过重复运用带余除法来找到两个整数的最大公因子。这个算法称为欧几里得算法。图 2.3 用一个简单的例子进行了阐述。

本质上，我们已经自顶到下地证明了最终结果就是 gcd(*a*, *b*)。我们也可以自底向上地证明。从自底向上的第一步可以看出，r_n 可以整除 *a* 和 *b*，而从上一个式子可以得出 r_n 整除 r_{n-1}。然后，又从上一个除法式子得出 r_n 整除 r_{n-2}（因为它可以整除等式右边的两项 $q_n r_{n-1}$ 和 r_n）。以此类推，我们发现 r_n 可以整除所有的 r_i，最后得到 r_n 整除 *a* 和 *b*。另外，还需要证明 r_n 是能够整除 *a* 和 *b* 的最大除数。如前所述，任何可以整除 *a* 和 *b* 的整数，一定可以整除 r_1。按照式（2.3）的顺序，可以证明 *c* 可以整除所有的 r_i。因此，$r_n = \gcd(a, b)$。

现在让我们通过一些较大的数字来看看这个算法的强大之处。

求 $d = \gcd(a, b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$
$r_2 = q_4 r_3 + r_4$	$104314826 = 31 \times 3313772 + 1587894$	$d = \gcd(3313772, 1587894)$
$r_3 = q_5 r_4 + r_5$	$3313772 = 2 \times 1587894 + 137984$	$d = \gcd(1587894, 137984)$
$r_4 = q_6 r_5 + r_6$	$1587894 = 11 \times 137984 + 70070$	$d = \gcd(137984, 70070)$
$r_5 = q_7 r_6 + r_7$	$137984 = 1 \times 70070 + 67914$	$d = \gcd(70070, 67914)$
$r_6 = q_8 r_7 + r_8$	$70070 = 1 \times 67914 + 2156$	$d = \gcd(67914, 2156)$
$r_7 = q_9 r_8 + r_9$	$67914 = 31 \times 2156 + 1078$	$d = \gcd(2156, 1078)$
$r_8 = q_{10} r_9 + r_{10}$	$2156 = 2 \times 1078 + 0$	$d = \gcd(1078, 0) = 1078$
所以，$d = \gcd(1160718174, 316258250) = 1078$		

在这个例子中，我们用 316258250 除 1160718174 作为开始，得到商为 3，余数为 211943424；接下来，我们用 211943424 除 316258250。继续这个过程，直到余数为 0 时停止，最终得到结果 1078。

将上述计算过程重新排列为表格的形式来看比较明了。对于循环的每一步，有 $r_{i-2} = q_i r_{i-1} + r_i$，其中 r_{i-2} 称为被除数，r_{i-1} 是除数，q_i 是商，r_i 是余数。表 2.1 总结了结果。

表 2.1 欧几里得算法实例

被除数	除 数	商	余 数
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943434$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$
$r_2 = 104314826$	$r_3 = 3313772$	$q_4 = 31$	$r_4 = 1587894$
$r_3 = 3313772$	$r_4 = 1587894$	$q_5 = 2$	$r_5 = 137984$
$r_4 = 1587894$	$r_5 = 137984$	$q_6 = 11$	$r_6 = 70070$
$r_5 = 137984$	$r_6 = 70070$	$q_7 = 1$	$r_7 = 67914$
$r_6 = 70070$	$r_7 = 67914$	$q_8 = 1$	$r_8 = 2156$
$r_7 = 67914$	$r_8 = 2156$	$q_9 = 31$	$r_9 = 1078$
$r_8 = 2156$	$r_9 = 1078$	$q_{10} = 2$	$r_{10} = 0$

2.3 模运算

2.3.1 模

如果 *a* 是一个整数，*n* 是正整数，则我们定义 *a* 除以 *n* 所得的余数为 ***a* 模 *n***。整数 *n* 称为模数。因此，对于任意整数 *a*，我们总可以把式（2.1）写成如下形式：

$$a = qn + r \qquad 0 \leq r < n; q = \lfloor a/n \rfloor$$
$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

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

若$(a \bmod n) = (b \bmod n)$，则称整数 a 和 b 是模 n 同余的，可以表示为 $a \equiv b \pmod n$ [①]。

$73 \equiv 4 \pmod{23} \qquad 21 \equiv -9 \pmod{10}$

注意到如果 $a \equiv 0 \pmod n$，那么 $n \mid a$。

2.3.2 同余的性质

同余有如下性质：

（1）若 $n \mid (a-b)$，则 $a \equiv b \pmod n$。

（2）若 $a \equiv b \pmod n$，则有 $b \equiv a \pmod n$。

（3）若 $a \equiv b \pmod n$，$b \equiv c \pmod n$，则有 $a \equiv c \pmod n$。

先证明第一条。若 $n \mid (a-b)$，则存在某个 k 使得 $(a-b) = kn$。于是可知 $a = b + kn$。因此$(a \bmod n) =$ $(b + kn$ 除以 n 的余数$) = (b$ 除以 n 的余数$) = (b \bmod n)$。

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

剩下的两个性质同样可以很容易地证明。

2.3.3 模算术运算

注意，由定义（参见图 2.1）可知，运算符（mod n）将所有整数映射到集合$\{0, 1, \cdots, (n-1)\}$中。于是出现了这样一个问题：能否限制在这个集合上进行算术运算？答案是可以，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）$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

下面证明第一个性质。令$(a \bmod n) = r_a$，$(b \bmod n) = r_b$，于是存在整数 j, k 使得 $a = r_a + jn$，$b = r_b + kn$。那么

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

剩下的性质同样可以简单地证明。下面是关于这三个性质的一些例子。

---

① 我们以两种方式使用了运算符 mod：首先是作为产生余数的二元操作符，如 $a \bmod b$；其次是作为一种同余关系，表明两个整数的等价关系，如 $a \equiv b \pmod n$。附录 2A 中有进一步的讨论。

11 mod 8 = 3; 15 mod 8 = 7
[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 = 2
(11+15) mod 8 = 26 mod 8 = 2
[(11 mod 8) − (15 mod 8)] mod 8 = −4 mod 8 = 4
(11−15) mod 8 = −4 mod 8 = 4
[(11 mod 8)×(15 mod 8)] mod 8 = 21 mod 8 = 5
(11×15) mod 8 = 165 mod 8 = 5

幂运算和普通算术一样，可以通过反复的乘法实现。

求 $11^7 \bmod 13$，我们可以如下进行：
$11^2 = 121 \equiv 4 \pmod{13}$
$11^4 = (11^2)^2 = 4^2 \equiv 3 \pmod{13}$
$11^7 = 11 \times 11^2 \times 11^4$
$11^7 \equiv 11 \times 4 \times 3 \equiv 123 \equiv 2 \pmod{13}$

因此，普通算术的加、减、乘运算法则可以平移到模算术中。

表 2.2 给出了模 8 的加法和乘法的例子。观察模 8 的加法例子，可以发现结果非常简单，并且矩阵有一定的规律。两个矩阵都关于主对角线对称，因为加法和乘法可以相互替换。与普通的加法运算一样，模运算中对于每个整数也存在加法的逆元素，或称为负数。在模运算中，整数 x 的负数 y 是使得$(x+y) \bmod 8 = 0$ 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；如$(2+6) \bmod 8 = 0$。同样，在乘法表中元素的意义更明确。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 $(x \times y) \bmod 8 = 1$ 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；如 $(3 \times 3) \bmod 8 = 1$。需要指出的是，并非所有整数模 8 都有乘法逆元；后面我们将会经常提到。

表 2.2 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 加法和乘法的逆

2.3.4 模运算的性质

定义比 n 小的非负整数集合为 Z_n

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

这个集合称为剩余类集，或模 n 的剩余类。更准确地说，Z_n 中的每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], …, [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] = {…, –16, –12, –8, –4, 0, 4, 8, 12, 16,…}
[1] = {…, –15, –11, –7, –3, 1, 5, 9, 13, 17,…}
[2] = {…, –14, –10, –6, –2, 2, 6, 10, 14, 18,…}
[3] = {…, –13, –9, –5, –1, 3, 7, 11, 15, 19…}

在剩余类的所有整数中，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。

如果在 Z_n 中进行模运算，那么表 2.3 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。

模运算有一个区别于普通运算的特性。首先，如普通算术中的运算一样，有

$$\text{若 } (a+b) \equiv (a+c)(\bmod n)\text{，则 } b \equiv c(\bmod n) \tag{2.4}$$

$$(5 + 23) \equiv (5 + 7) \pmod 8;\ 23 \equiv 7 \pmod 8$$

式（2.4）与加法逆元的存在性是一致的。将 a 的加法逆元加在式（2.4）的两边，可得

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

表 2.3 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	$[w \times (x+y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
单位元	$(0+w) \bmod n = w \bmod n$;　$(1 \times w) \bmod n = w \bmod n$
加法逆($-w$)	对于每个 $w \in Z_n$ 都存在一个 z 使得 $w + z \equiv 0 \bmod n$

然而，下面的陈述仅在有附加条件时才成立：

$$\text{若 } a \text{ 与 } n \text{ 互素：若 } (a \times b) \equiv (a \times c)(\bmod n)\text{，则 } b \equiv c(\bmod n) \tag{2.5}$$

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。同式（2.4）的情况类似，也可以说式（2.5）同乘法逆元的存在性是一致的。将 a 的乘法逆元施加在式（2.5）的两边得

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

要明白这一点，我们来看一个使式（2.5）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：

$$6\times 3 = 18 \equiv 2 \pmod 8$$
$$6\times 7 = 42 \equiv 2 \pmod 8$$

然而 $3 \neq 7 \pmod 8$。

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 的一个完整剩余类集。

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
剩余类	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类集，Z_8 中有多个整数映射到相同的剩余类。特别地，$6\times 0 \bmod 8 = 6\times 4 \bmod 8$; $6\times 1 \bmod 8 = 6\times 5 \bmod 8$；以此类推。因为这是一个多对一映射，所以乘法运算并没有唯一的逆运算。

然而，如果令 $a=5$，$n=8$，二者唯一的公因子是 1

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

剩余类这一行包含了 Z_8 所在行的所有整数，只是顺序不同。

一般来说，只有当一个整数与 n 互素时，它才会在 Z_n 中存在一个乘法逆元。从表 2.2(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 没有。

2.3.5 回顾欧几里得算法

欧几里得算法基于下面的定理：对任意整数 a, b，且 $a \geq b \geq 0$

$$\gcd(a,b) = \gcd(b, a \bmod b) \tag{2.6}$$

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

要知道式（2.6）的结果是如何得到的，我们定义 $d=\gcd(a, b)$。然后，根据 gcd 的定义，有 $d \mid a$、$d \mid b$ 成立。对于任何正整数 b，可以用整数 k, r 得到 a 的一个表示

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

因此，对于某些整数 k，存在$(a \bmod b) = a - kb$。由于 $d \mid b$，所以它也能整除 kb。同样，我们有 $d \mid a$，因此可得 $d \mid (a \bmod b)$。这表示 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，那么有 $d \mid kb$，并且可知 $d \mid [kb + (a \bmod b)]$，即 $d \mid a$。所以，a 和 b 的公因子的集合等于 b 和$(a \bmod b)$的公因子的集合。因此，两对数（a 和 b，b 和 $a \bmod b$）的 gcd 相同，定理得证。

为了求出最大公因子，可多次重复使用式（2.6）。

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

这与式（2.3）所示的方案相同，可重新改写如下：

欧几里得算法	
计　算	满 足 条 件
$r_1 = a \bmod b$	$a = q_1 b + r_1$
$r_2 = b \bmod r_1$	$b = q_2 r_1 + r_2$
$r_3 = r_1 \bmod r_2$	$r_1 = q_3 r_2 + r_3$
⋮	⋮
$r_n = r_{n-2} \bmod r_{n-1}$	$r_{n-2} = q_n r_{n-1} + r_n$
$r_{n+1} = r_{n-1} \bmod r_n = 0$	$r_{n-1} = q_{n+1} r_n + 0$ $d = \gcd(a, b) = r_n$

可用下面的递归函数更加精确地定义欧几里得算法：

```
Euclid(a,b)
        if (b=0) then return a;
        else return Euclid(b,a mod b);
```

2.3.6 扩展的欧几里得算法

下面介绍欧几里得算法的扩展，它对于后面有限域中的计算及 RSA 等密码算法非常重要。对于给定的整数 a 和 b，扩展的欧几里得算法不仅可以计算出最大公因子 d，而且可以得到两个整数 x 和 y，它们满足如下方程：

$$ax + by = d = \gcd(a, b) \tag{2.7}$$

首先应该明白的是，x 和 y 一定具有相反的正负号。在研究该算法之前，让我们先看一下 $a = 42$，$b = 30$ 时，x 和 y 的一些值。注意 gcd(42, 30) = 6。下面是 $42x + 30y$ 的部分值[①]。

y \ x	-3	-2	-1	0	1	2	3
-3	-216	-174	-132	-90	-48	-6	36
-2	-186	-144	-102	-60	-18	24	66
-1	-156	-114	-72	-30	12	54	96
0	-126	-84	-42	0	42	84	126
1	-96	-54	-12	30	72	114	156
2	-66	-24	18	60	102	144	186
3	-36	6	48	90	132	174	216

可以观察到，所有的项都可被 6 整除。这并不奇怪，因为 42 和 30 都可被 6 整除，所以每个形如 $42x + 30y = 6(7x + 5y)$ 的数都是 6 的倍数。还可发现，gcd(42, 30)= 6 也出现在表中。一般地，可以证明，对于两个给定的整数 a 和 b，可以表示为 $ax + by$ 的最小正整数等于 gcd(a, b)。

下面介绍对于给定的 a 和 b，如何扩展欧几里得算法去计算(x, y, d)。我们再次沿式（2.3）所示的除法顺序进行，并假设在每个步骤 i 都可找到 x_i 和 y_i 满足 $r_i = ax_i + by_i$。最终，得到下式：

$$\begin{aligned}
a &= q_1 b + r_1 & r_1 &= ax_1 + by_1 \\
b &= q_2 r_1 + r_2 & r_2 &= ax_2 + by_2 \\
r_1 &= q_3 r_2 + r_3 & r_3 &= ax_3 + by_3 \\
&\vdots & &\vdots \\
r_{n-2} &= q_n r_{n-1} + r_n & r_n &= ax_n + by_n \\
r_{n-1} &= q_{n+1} r_n + 0 & &
\end{aligned}$$

① 这个例子取自文献[SILV06]。

移项得

$$r_i = r_{i-2} - r_{i-1}q_i \tag{2.8}$$

同样，从 i–1 行和 i–2 行，可得到值

$$r_{i-2} = ax_{i-2} + by_{i-2} \quad 和 \quad r_{i-1} = ax_{i-1} + by_{i-1}$$

代入式（2.8）有

$$r_i = (ax_{i-2} + by_{i-2}) - (ax_{i-1} + by_{i-1})q_i = a(x_{i-2} - q_i x_{i-2}) + b(y_{i-2} - q_i y_{i-1})$$

然而，我们已假设 $r_i = ax_i + by_i$，因此

$$x_i = x_{i-2} - q_i x_{i-1} \quad 和 \quad y_i = y_{i-2} - q_i y_{i-1}$$

总结计算过程如下：

扩展的欧几里得算法

计　算	满　足	计　算	满　足
$r_{-1} = a$		$x_{-1} = 1; y_{-1} = 0$	$a = ax_{-1} + by_{-1}$
$r_0 = b$		$x_0 = 0; y_0 = 1$	$b = ax_0 + by_0$
$r_1 = a \bmod b$ $q_1 = \lfloor a / b \rfloor$	$a = q_1 b + r_1$	$x_1 = x_{-1} - q_1 x_0 = 1$ $y_1 = y_{-1} - q_1 y_0 = -q_1$	$r_1 = ax_1 + by_1$
$r_2 = b \bmod r_1$ $q_2 = \lfloor b / r_1 \rfloor$	$b = q_2 r_1 + r_2$	$x_2 = x_0 - q_2 x_1$ $y_2 = y_0 - q_2 y_1$	$r_2 = ax_2 + by_2$
$r_3 = r_1 \bmod r_2$ $q_3 = \lfloor r_1 / r_2 \rfloor$	$r_1 = q_3 r_2 + r_3$	$x_3 = x_1 - q_3 x_2$ $y_3 = y_1 - q_3 y_2$	$r_3 = ax_3 + by_3$
⋮	⋮	⋮	⋮
$r_n = r_{n-2} \bmod r_{n-1}$ $q_n = \lfloor r_{n-2} / r_{n-1} \rfloor$	$r_{n-2} = q_n r_{n-1} + r_n$	$x_n = x_{n-2} - q_n x_{n-1}$ $y_n = y_{n-2} - q_n y_{n-1}$	$r_n = ax_n + by_n$
$r_{n+1} = r_{n-1} \bmod r_n = 0$ $q_{n+1} = \lfloor r_{n-1} / r_n \rfloor$	$r_{n-1} = q_{n+1} r_n + 0$		$d = \gcd(a, b) = r_n$ $x = x_n; y = y_n$

这里还需要多做一些解释。在每一行，我们基于前两行的余数 r_{i-1} 和 r_{i-2} 来计算一个新余数 r_i。为了开始这个算法，需要 r_0 和 r_{-1} 有值，它们分别是 a 和 b。而 x_{-1}，y_{-i}，x_0 和 y_0 的取值则很直观。

从原始的欧几里得算法可知，该计算过程会在余数为 0 时结束，从而求得 a 和 b 的最大公因子为 $d = \gcd(a, b) = r_n$。而与此同时，我们也确定了 $d = r_n = ax_n + by_n$。因此在式（2.7）中，$x = x_n$，$y = y_n$。

下面举一个例子。令 $a = 1759$，$b = 550$，解等式 $1759x + 550y = \gcd(1759, 550)$。结果如表 2.4 所示。因此，我们有 $1759 \times (-111) + 550 \times 355 = -195249 + 195250 = 1$。

表 2.4　扩展欧几里得算法例子

i	r_i	q_i	x_i	y_i
–1	1759		1	0
0	550		0	1
1	109	3	1	–3
2	5	5	–5	16
3	4	21	106	–339
4	1	1	–111	355
5	0	4		

2.4 素数[①]

数论的核心是素数，实际上所有关于数论的书都是围绕这一主题来写的（如文献[CRAN01]和文献[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[②]± 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 2.5 列出了 2000 以内的素数。请注意素数的分布，尤其要注意每 100 个数中素数的个数。

任意整数 $a > 1$ 都可以唯一地因式分解为

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t} \tag{2.9}$$

式中 $p_1, p_2, \cdots, p_t$ 均是素数，$p_1 < p_2 < \cdots < p_t$，且所有 a_i 都是正整数。这就是算术基本定理。任何有关数论的教材都含有该定理的证明。

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$
$$11011 = 7 \times 11^2 \times 13$$

表 2.5 2000 以内的素数

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

① 若无特别说明，本节只讨论非负整数，使用负整数不会有本质上的不同。

② 2.1 节中说 a 是 b 的一个因子，如果 a 除 b 没有余数。同样，我们也可以说 a 整除 b。

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 a 可唯一地表示为

$$a=\prod_{p\in P}p^{a_p}\ ,\quad a_p\geqslant 0$$

上式右边是所有可能的素数 p 之积。对任意整数 a，其大多数指数 a^p 为 0。

我们可以通过列出上述公式中的所有非零指数来唯一地表示一个正整数。

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
整数 91 可用$\{a_7=1, a_{13}=1\}$来表示

两数相乘即指数对应相加。设 $a=\prod_{p\in P}p^{a_p}, b=\prod_{p\in P}p^{b_p}$，定义 $k=ab$。我们知道整数 k 可以表示为素数方幂的乘积，即 $k=\prod_{p\in P}p^{k_p}$。可以推出对于所有的 $p\in P$，有 $k_p=a_p+b_p$ 成立。

$k=12\times18=(2^2\times3)\times(2\times3^2)=216$
$k_2=2+1=3; k_3=1+2=3$
$216=2^3\times3^3=8\times27$

从素因子的角度看，a 整除 b，即$(a\mid b)$意味着什么呢？由任意整数 n 形成的 p^n 只能被小于等于相同幂次的同一素数即 p^j 整除（其中 $j\leqslant n$）。所以，有如下结论。

假设 $a=\prod_{p\in P}p^{a_p}, b=\prod_{p\in P}p^{b_p}$，如果 $a\mid b$，那么对任意的 $p\in P$ 有 $a_p\leqslant b_p$。

$a=12; b=36; 12|36$
$12=2^2\times3; 36=2^2\times3^2$
$a_2=2=b_2$
$a_3=1\leqslant 2=b_3$
因此不等式 $a_p\leqslant b_p$ 对所有素数成立

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。

$300=2^2\times3^1\times5^2$
$18=2^1\times3^2$
$\gcd(18, 300)=2^1\times3^1\times5^0=6$

下列关系式总成立：

如果 $k=\gcd(a, b)$，那么 $k_p=\min(a_p, b_p)$，对任意的 $p\in P$

确定一个大数的素因子并不简单，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。

2.5　费马定理和欧拉定理

费马定理和欧拉定理在公钥密码学中占有重要地位。

2.5.1 费马定理[①]

费马定理可如下描述：若 p 是素数，a 是正整数且不能被 p 整除，则

$$a^{p-1} \equiv 1 \pmod{p} \tag{2.10}$$

证明 考虑小于 p 的正整数集合$\{1, \cdots, p-1\}$，用 a 乘以集合中的所有元素并对 p 取模，得到集合 $X = \{a \bmod p, 2a \bmod p, \cdots, (p-1)\, a \bmod p\}$。因为 p 不能整除 a，所以 X 的元素都不等于 0，而且各元素互不相等。为了说明这一点，假设 $ja \equiv ka \pmod{p}$，其中 $1 \leqslant j < k \leqslant p-1$，因为 a 和 p 互素[②]，因此可将等式［参见式（2.3）］两边的 a 消去，推出 $j \equiv k \pmod{p}$。这个最后的等式不成立，因为 j 和 k 都是小于 p 的正整数。因此 X 的 $p-1$ 个元素都是正整数且互不相等。所以说 X 和$\{1, \cdots, p-1\}$的构成相同，只是元素顺序不同。将两个集合的所有元素分别相乘，并对结果模 p，有

$$a \times 2a \times \cdots \times (p-1)a \equiv [(1 \times 2 \times \cdots \times (p-1)] \pmod{p}$$

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}$$

因为$(p-1)!$和 p 互素，根据式（2.5），可以消去$(p-1)!$这一项，得到式（2.10）。证毕。

$a = 7, p = 19$
$7^2 = 49 \equiv 11 \pmod{19}$
$7^4 \equiv 121 \equiv 7 \pmod{19}$
$7^8 \equiv 49 \equiv 11 \pmod{19}$
$7^{16} \equiv 121 \equiv 7 \pmod{19}$
$a^{p-1} = 7^{18} \equiv 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 \pmod{19}$

费马定理的另一种有用表示形式是：若 p 是素数且 a 是任意正整数，则

$$a^p \equiv a \bmod p \tag{2.11}$$

请注意定理的第一种形式［参见式（2.10）］要求 a 与 p 互素，而后一种没有这一要求。

$p = 5, a = 3, a^p = 3^5 = 243 \equiv 3 \pmod{5} = a \pmod{p}$
$p = 5, a = 10, a^p = 10^5 = 100000 \equiv 10 \pmod{5} \equiv 0 \pmod{5} = a \pmod{p}$

2.5.2 欧拉函数

在给出欧拉定理之前，我们需要引入数论中的一个非常重要的概念，即**欧拉函数** $\phi(n)$，它指小于 n 且与 n 互素的正整数的个数。习惯上，$\phi(1) = 1$。

求$\phi(37)$和$\phi(35)$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，因此$\phi(37) = 36$
要计算$\phi(35)$，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$\phi(35) = 24$

① 这通常指的是费马小定理。

② 从 2.2 节可知，两个数若无公共素因子，即它们的唯一公因子为 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。

表 2.6 列出了$\phi(n)$前 30 项的值，虽然$\phi(1)$的值无意义，但我们仍将它定义为 1。

显然，对素数 p，有

$$\phi(p) = p-1$$

假设有两个素数 p 和 q，$p \neq q$，那么对于 $n = pq$，有

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

表 2.6　欧拉函数$\phi(n)$前 30 项的值

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

为证明$\phi(n) = \phi(p) \times \phi(q)$，考虑集合$\{1, \cdots, (pq-1)\}$，不与 n 互素的集合是$\{p, 2p, \cdots, (q-1)p\}$和$\{q, 2q, \cdots, (p-1)q\}$。之所以得出这个结果，是因为考虑到任何整数除 n 都必须除以 p 或 q。因此，与 n 互素的整数都不会包含 p 或 q；此外，这两个集合是非重叠的：因为 p 和 q 都是素数，我们发现第一个集合中不存在一个为 q 的整数倍的数，第二个集合中也不存在一个为 p 的整数倍的数。因此，两个集合中共有$(q-1) + (p-1)$个整数，所以

$$\begin{aligned}\phi(n) &= (pq-1) - [(q-1) + (p-1)] \\ &= pq - (p+q) + 1 \\ &= (p-1) \times (q-1) \\ &= \phi(p) \times \phi(q)\end{aligned}$$

$\phi(21) = \phi(3) \times \phi(7) = (3-1) \times (7-1) = 2 \times 6 = 12$
其中 12 个整数是$\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

2.5.3　欧拉定理

欧拉定理说明，对任意互素的 a 和 n，有

$$a^{\phi(n)} \equiv 1 (\mathrm{mod}\ n) \tag{2.12}$$

证明　若 n 是素数，根据$\phi(n) = (n-1)$和费马定理，式（2.12）成立，但实际上该式对任意整数 n 都成立。请注意$\phi(n)$是指小于 n 且与 n 互素的正整数的个数，考虑这些整数所组成的集合

$$R = \{x_1, x_2, \cdots, x_{\phi(n)}\}$$

即每个元素都有 $\gcd(x_i, n) = 1$。将 a 与 R 中的每个元素相乘，然后模 n

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \cdots, (ax_{\phi(n)} \bmod n)\}$$

S 是 R 的一个排列[①]，因为

（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 中所有元素均小于 n 且与 n 互素。

（2）S 中没有重复元素。参见式（2.5）。若 $ax_i \bmod n = ax_j \bmod n$，则 $x_i = x_j$。

所以集合 S 是 R 的一个置换。因此

$$\prod_{i=1}^{\phi(n)}(ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod n$$

$$a^{\phi(n)} \equiv 1 \pmod n$$

证毕。这里使用的推理和费马定理的证明一样。

$$a = 3; n = 10; \phi(10) = 4; a^{\phi(n)} = 3^4 = 81 \equiv 1 \bmod 10 = 1 \pmod n$$
$$a = 2; n = 11; \phi(11) = 10; a^{\phi(n)} = 2^{10} = 1024 \equiv 1 \pmod{11} = 1 \pmod n$$

类似费马定理，欧拉定理的另一种表述也很有用

$$a^{\phi(n)+1} \equiv a \pmod n \tag{2.13}$$

同样，和费马定理类似，欧拉定理的第一种形式［参见式（2.12）］要求 a 与 p 互素，而后一种没有这一要求。

2.6 素性测试

许多密码算法都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来解决该问题。

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。读者也许会对此感到惊讶，但该算法产生的数几乎可以肯定是素数。我们将很快解释这一点。我们也会讨论如何用决定性算法寻找素数。另外，本节的最后将讨论素数的分布。

2.6.1 Miller-Rabin 算法[②]

Miller 和 Rabin 提出的这一算法[MILL75, RABI80]是典型的大数素性测试算法。在阐述算法前，我们先给出一些背景。首先，$n \geq 3$ 的奇整数可表示如下：

$$n-1 = 2^k q, \quad k > 0, q \text{ 是奇数}$$

为了证明这一点，我们注意到$(n-1)$是一个偶数，接着用 2 去除$(n-1)$，直至所得结果为奇数，此处共做了 k 次除法。如果 n 是二进制数表示，则将该数向右移，直到最右边的位为 1 为止，这时可得到结果，此处移动了 k 次。现在给出我们要用到的素数的两个性质。

素数的两个性质 第一个性质叙述如下：若 p 是素数，a 是小于 p 的正整数，则 $a^2 \bmod p = 1$ 当且仅当 $a \bmod p = 1$ 或 $a \bmod p = -1 \bmod p = p - 1$。运用模算术运算规则，$(a \bmod p)(a \bmod p) = a^2$

① 有限集 S 的一个排列，是指 S 中某个给定顺序的元素，在该顺序中每个元素只出现一次。

② 文献中也称为 Rabin-Miller 算法、Rabin-Miller 测试或 Miller-Rabin 测试。

$\bmod p$。因此，若 $a \bmod p = 1$ 或 $a \bmod p = -1$，则有 $a^2 \bmod p = 1$。反之，若 $a^2 \bmod p = 1$，则$(a \bmod p)^2 = 1$ 成立，其中 $a \bmod p = 1$ 或 $a \bmod p = -1$ 之一成立。

第二个性质叙述如下：设 p 是大于 2 的素数，我们有 $p - 1 = 2^k q$，其中 $k > 0$，q 为奇数。设 a 是整数且 $1 < a < p - 1$，则有下面两个条件之一成立：

（1）a^q 模 p 和 1 同余，即 $a^q \bmod p = 1$，或等价地，$a^q \equiv 1 (\bmod p)$。

（2）整数 $a^q, a^{2q}, a^{4q}, \cdots, a^{2^{k-1}q}$ 中存在一个数，模 p 时和–1 同余。即存在一个 j（$1 \leqslant j \leqslant k$）满足 $a^{2^{j-1}q} \bmod p = -1 \bmod p = p-1$，或 $a^{2^{j-1}q} \equiv -1 (\bmod p)$。

证明 若 n 是素数，则由费马定理［参见式（2.10）］可知，$a^{n-1} \equiv 1 (\bmod n)$。由于 $p - 1 = 2^k q$，则 $a^{p-1} \bmod p = a^{2^k q} \bmod p = 1$。因此，如果看下述数列：

$$a^q \bmod p, a^{2q} \bmod p, \cdots, a^{2^{k-1}q} \bmod p, a^{2^k q} \bmod p \tag{2.14}$$

就知道最后的数为 1。而且，每个数都是前一个数的平方。因此，下面的两条必有一条是正确的：

（1）数列的第一个数为 1，那么其后的所有数都为 1。

（2）数列中有些数不为 1，但它们的平方模 p 后为 1。根据刚才给出的素数第一性质，我们知道满足这个条件的唯一整数为 $p - 1$。因此，此时数列中必有一个数为 $p - 1$。

证毕。

详细的算法 由上述讨论可知，若 n 为素数，则剩余类数列$(a^q, a^{2q}, \cdots, a^{2^{k-1}q}, a^{2^k q})$中，要么第一个数模 n 为 1，要么数列中的某个数模 n 为$(n - 1)$；否则 n 为合数。另一方面，如果条件满足，也不一定推出 n 为素数。例如，设 $n = 2047 = 23 \times 89$，则 $n - 1 = 2 \times 1023$。计算 $2^{1023} \bmod 2047 = 1$，所以 2047 满足条件，但不是素数。

我们可以利用上述性质做素性测试。过程 TEST 输入整数 n，若 n 不是素数，则返回合数，若 n 可能是素数，也可能不是素数时，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^k q);
2. 随机选取整数 a,1<a<n-1;
3. if a^q mod n=1, then 返回"不确定";
4. for j=0 to k-1 do
5. if a^(2^j q) mod 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$(n - 1) = 28 = 2^2(7) = 2^k q$。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”（即 29 可能为素数）。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 为素数是一致的。

对合数 $n = 13 \times 17 = 221$ 应用上述测试，有$(n - 1) = 220 = 2^2(55) = 2^k q$。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 行中的所有 $j(0, 1)$进行了测试，所以算法返回“合数”，这表明 221 肯定是合数。然而，假设我们选取 $a = 21$，则 $21^{55} \bmod 221 = 200$，$(21^{55})^2 \bmod 221 = 220$，则测试算法返回“不确定”，表明 221 可能是素数。事实上，在 2 到 219 之间的 218 个整数中，有 4 个整数会返回“不确定”，它们是 21, 47, 174 和 200。

重复使用 Miller-Rabin 算法 如何使用 Miller-Rabin 算法以更高的可信度来判定一个整数是否为素数？根据文献[KNUT98]，给定一个非素奇数 n 和一个随机选择的整数 a，$1 < a < n-1$，程序 TEST 返回“不确定”的概率小于 1/4（即不能确定 n 是不是素数）。因此，如果选择 t 个不同的 a，则它们都能通过测试（返回“不确定”）的概率小于$(1/4)^t$。例如，取 $t = 10$，则一个非素整数通过 10 次测试的概率小于 10^{-6}。因此，取足够大的 t，如果 Miller 测试总是返回“不确定”，则我们能以很大的把握说 n 是素数。

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”，这样当 t 足够大时，我们可以相信 n 是素数。

2.6.2 一个确定性的素性判定算法

在 2002 年以前，没有高效的方法能证明一个大数的素性。包括最为常用的 Miller-Rabin 算法在内，所有在用的算法给出的都是概率性结果。2002 年，Agrawal、Kayal 和 Saxena[AGRA04]给出了一个相对简单的可以有效判定一个大数是否为素数的确定性算法。这个称为 AKS 的算法并没有 Miller-Rabin 算法快，因此它未代替古老的概率算法。

2.6.3 素数的分布

值得注意的是，利用 Millelr-Rabin 测试算法或其他任何素数测试算法时，发现一个素数之前会测试多少个整数？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。因为所有偶数肯定不是素数，因此需测试的整数个数为 $0.5\times\ln(n)$。例如，若要找 2^{200} 左右的素数，则约需 $0.5\ln(2^{200})= 69$ 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 1001! + 2, 1001! + 3, ..., 1001! + 1000, 1001! + 1001 这 1000 个连续的整数均是合数。

2.7 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的一组剩余类数来重构，这组剩余类数是对该整数用一组两两互素的整数取模得到的。

> Z_{10}（0, 1, ⋯, 9）中的 10 个整数可通过它们对 2 和 5（10 的两个素因子）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，故唯一解为 $x = 8$。

CRT 有几种不同的表示形式，这里给出其中一种最有用的表示形式，习题 2.33 给出了另一种表示形式。令

$$M = \prod_{i=1}^{k} m_i$$

① 因为人们认为这个定理是公元 100 年由中国数学家孙子发现的，所以称为 CRT。

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_i} 中，这种对应关系即为

$$A\leftrightarrow(a_1, a_2, \cdots, a_k) \tag{2.15}$$

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）式（2.15）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，$0\leqslant A<M$，有唯一的 k 元组$(a_1, a_2, \cdots, a_k)$与之对应，其中 $0\leqslant a_i<m_i$，并且对任何这样的 k 元组$(a_1, a_2, \cdots, a_k)$，Z_M 中有唯一的 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，即在笛卡儿积的每个分量上独立地执行运算。

下面证明第一个断言。由 A 到$(a_1, a_2, \cdots, a_k)$的转换显然是唯一确定的。即只需取 $a_i=A \bmod m_i$。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 $j\neq i$，有 $M_i\equiv 0 \pmod{m_j}$。令

$$c_i=M_i\times(M_i^{-1} \bmod m_i), \qquad 1\leqslant i\leqslant k \tag{2.16}$$

根据 M_i 的定义，有 M_i 与 m_i 互素，所以 M_i 有唯一的模 m_i 的乘法逆元，因此式（2.16）是良定的，这样可得到唯一的 c_i。计算

$$A\equiv\left(\sum_{i=1}^{k}a_ic_i\right)(\bmod M)$$

要证明式（2.17）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 $a_i=A \bmod m_i$。

第二个断言与算术运算有关，可从模算术规则推出，即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

$$(A+B) \bmod M\leftrightarrow((a_1+b_1) \bmod m_1, \cdots, (a_k+b_k) \bmod m_k)$$
$$(A-B) \bmod M\leftrightarrow((a_1-b_1) \bmod m_1, \cdots, (a_k-b_k) \bmod m_k)$$
$$(A\times B) \bmod M\leftrightarrow((a_1\times b_1) \bmod m_1, \cdots, (a_k\times b_k) \bmod m_k)$$

中国剩余定理的用途之一是，它给出了一种方法，使得模 M 的大数运算转化到相对较小的数的运算，当 M 为 150 位或 150 位以上时，这种方法非常有效，但事先需要分解 M。

下面将 973 mod 1813 表示为模 37 和 49 的两个数。我们定义

$$m_1=37, m_2=49, M=1813, A=973$$

则 $M_1=49$ 且 $M_2=37$。利用扩展的欧几里得算法有 $M_1^{-1}=34 \bmod m_1$ 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)？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

$$
\begin{aligned}
(23,34) \leftrightarrow & a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} \bmod M \\
&= [(23)(49)(34) + (34)(37)(4)] \bmod 1813 \\
&= 43350 \bmod 1813 \\
&= 1651
\end{aligned}
$$

而(973+678) mod 1813 = 1651。请记住上述推导过程中，M_i^{-1} 是 M_1 模 m_1 的乘法逆，而 M_2^{-1} 是 M_2 模 m_2 的乘法逆。

假定要计算 73 乘 1651(mod 1813)。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

$$
\begin{aligned}
(14,32) \leftrightarrow & [(14)(49)(34) + (32)(37)(4)] \bmod 1813 \\
&= 856 \\
&= 1651 \times 73 \bmod 1813
\end{aligned}
$$

2.8 离散对数

离散对数是包括 Diffie-Hellman 密钥交换算法和数字签名算法（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，详细内容请读者参阅文献[ORE67]和[LEVE90]。

2.8.1 模 n 的整数幂

欧拉定理［参见式（2.12）］告诉我们，对任何互素的 a 和 n，有

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

其中欧拉函数$\phi(n)$是指小于 n 且与 n 互素的正整数的个数。下面考虑欧拉定理更一般的表示形式：

$$a^m \equiv 1 \pmod{n} \tag{2.18}$$

若 a 与 n 互素，则至少有一个整数 m 满足式（2.18），即 $m = \phi(n)$，我们称使得式（2.18）成立的最小正幂 m 为下列之一：

- a（mod 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，考虑下面 7 模 19 的各次幂：

$$
\begin{aligned}
7^1 &\equiv 7 \pmod{19} \\
7^2 = 49 = 2 \times 19 + 11 &\equiv 11 \pmod{19} \\
7^3 = 343 = 18 \times 19 + 1 &\equiv 1 \pmod{19} \\
7^4 = 2401 = 126 \times 19 + 7 &\equiv 7 \pmod{19} \\
7^5 = 16807 = 884 \times 19 + 11 &\equiv 11 \pmod{19}
\end{aligned}
$$

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，且其周期长是使 $7^m \equiv 1 \pmod{19}$成立的最小正幂 m。

表 2.7 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。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，表中的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）生成了模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。

并非所有整数都有本原根。事实上，只有形为2, 4, p^{α}和$2p^{\alpha}$的整数才有本原根，其中p是任何奇素数，α是正整数。证明并不简单，但在许多数论书中都有，如文献[ORE76]。

表2.7 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

2.8.2 模算术对数

对于普通正实数，对数函数是指数函数的逆函数。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

$$y = x^{\log_x(y)}$$

对数具有下列性质：

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z) \tag{2.19}$$

$$\log_x(y^r) = r \times \log_x(y) \tag{2.20}$$

对某素数 p（对非素数亦可）的本原根 a，a 的 1 到$(p-1)$的各次幂恰可一次产生 1 到$(p-1)$的每个整数，且仅一次。而对任何整数 b，根据模算术的定义，b 满足

$$b \equiv r \pmod{p}, \quad 0 \leq r \leq (p-1)$$

因此，对任何整数 b 和素数 p 的本原根 a，有唯一的幂 i 使得

$$b \equiv a^i \pmod{p}, \quad 0 \leq i \leq (p-1)$$

该指数 i 称为以 a 为底（模 p）的 b 的离散对数，记为 $\text{dlog}_{a,p}(b)$[①]。

请注意下列各式：

$$\text{dlog}_{a,p}(1) = 0, \quad \text{因为 } a^0 \bmod p = 1 \bmod p = 1 \tag{2.21}$$

$$\text{dlog}_{a,p}(a) = 1, \quad \text{因为 } a^1 \bmod p = a \tag{2.22}$$

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

$2^0 = 1$　　$2^4 \equiv 7 \pmod 9$

$2^1 = 2$　　$2^5 \equiv 5 \pmod 9$

$2^2 = 4$　　$2^6 \equiv 1 \pmod 9$

$2^3 = 8$

因此，对给定的本原根 $a = 2$（模 9）的离散对数，有下列数表：

对数	0	1	2	3	4	5
数	1	2	4	8	7	5

给定一个数，要得到其离散对数，我们可以重排上表，得

数	1	2	4	5	7	8
对数	0	1	2	5	4	3

考虑

$$x = a^{\text{dlog}_{a,p}(x)} \bmod p \qquad y = a^{\text{dlog}_{a,p}(y)} \bmod p$$

$$xy = a^{\text{dlog}_{a,p}(xy)} \bmod p$$

由模乘法的性质有

$$xy \bmod p = [(x \bmod p)(y \bmod p)] \bmod p$$

$$a^{\text{dlog}_{a,p}(xy)} \bmod p = [(a^{\text{dlog}_{a,p}(x)} \bmod p)(a^{\text{dlog}_{a,p}(y)} \bmod p)] \bmod p$$

$$= (a^{\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)}) \bmod p$$

根据欧拉定理，对任何互素的 a 和 n，有

$$a^{\phi(n)} \equiv 1 \pmod n$$

此外，任何正整数 z 可表示为 $z = q + k\phi(n)$，其中 $0 \leq q < \phi(n)$，所以由欧拉定理有

$$\text{若 } z \equiv q \bmod \phi(n), \text{ 则 } a^z \equiv a^q \pmod n$$

① 许多教科书把离散对数称为指标（index），这个概念没有通用的记号，更没有统一的名字。

将其代入前面的等式，则有

$$\mathrm{dlog}_{a,p}(xy) \equiv [\mathrm{dlog}_{a,p}(x) + \mathrm{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

由此可得

$$\mathrm{dlog}_{a,p}(y^r) \equiv [r \times \mathrm{dlog}_{a,p}(y)] \pmod{\phi(p)}$$

这说明了普通对数和离散对数之间的相似性。

注意，仅当 a 是 m 的本原根时，才存在有唯一的以 a 为底模 m 的离散对数。

表 2.8 可直接由表 2.7 推出，它列出了模 19 的离散对数集。

表 2.8 模 19 的离散对数表

(a)以 2 为底，模为 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
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b)以 3 为底，模为 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
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c)以 10 为底，模为 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
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d)以 13 为底，模为 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
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e)以 14 为底，模为 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
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f)以 15 为底，模为 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
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

2.8.3 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，而且存在计算 y 的有效算法（参见第 9 章）。

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的数量级相同。编写本书时，已知最快求模数为素数的离散对数的算法的难度阶为[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。

2.9 关键术语、思考题和习题

关键术语

双射	因子	最大公因子	阶
合数	欧几里得算法	单位元	素数
交换性	欧拉定理	指标	本原根
中国剩余定理	欧拉函数	模运算	互素
离散对数	费马定理	模量	余数

思考题

2.1 b 是 a 的因子是什么意思？

2.2 表达式 a 整除 b 的意义是什么？

2.3 模运算和普通运算的区别在哪？

2.4 什么是素数？

2.5 什么是欧拉函数？

2.6 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。如何用该算法测试素性？

2.7 一个数的本原根是什么？

2.8 指标和离散对数的区别是什么？

习题

2.1 去掉 a 为非负整数的限制，即 a 可以为任何整数，重新叙述式（2.1）。

2.2 对于 $a<0$，画一个与图 2.1 类似的图。

2.3 对于如下的每个方程，求相应的 x：

（a）$5x \equiv 4 \pmod 3$；（b）$7x \equiv 6 \pmod 5$；（c）$9x \equiv 8 \pmod 7$

2.4 本书假设模数为正整数。但是，如果模数 n 为负数，表达式 $a \bmod n$ 也是完全有意义的。求如下表达式的值：

（a）5 mod 3；（b）5 mod −3；（c）−5 mod 3；（d）−5 mod −3

2.5 模数为 0 并不适合上述定义，但习惯上如下定义：$a \bmod 0 = a$。按此定义，$a \equiv b \pmod 0$ 有何含义？

2.6 2.3 节将同余关系定义如下：整数 a 和 b 模 n 同余，当且仅当 $(a \bmod n) = (b \bmod n)$。然后证明了若 $n \mid (a-b)$，则 $a \equiv b \pmod n$。有些教材将后者作为同余关系的定义：若 $n \mid (a-b)$，则整数 a 和 b 称为模 n 同余。从后一个定义出发，证明若 $(a \bmod n) = (b \bmod n)$，则 n 整除 $(a-b)$。

2.7 对于 $1 \leqslant k \leqslant 6$，刚好有 k 个因子的最小正整数是多少？

2.8 证明以下论断：

（a）由 $a \equiv b \pmod n$ 可推出 $b \equiv a \pmod n$。

（b）由 $a \equiv b \pmod n$ 且 $b \equiv c \pmod n$ 可推出 $a \equiv c \pmod n$。

2.9 证明以下等式：

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$
$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

2.10 求 Z_5 中各非零元素的乘法逆元。

2.11　证明任意十进制整数 N 及其各位数字之和模 9 同余。例如，$475 \equiv 4+7+5 \equiv 16 \equiv 1+6 \equiv 7 (\text{mod } 9)$。这是在计算算术运算时，“舍去 9”的常见程序的基础。

2.12　（a）求 gcd(24140,16762)；（b）求 gcd(4655, 12075)。

2.13　本题的目的在于给欧几里得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 是欧几里得算法进行 i 次迭代后 A 的值，证明 $A_{i+2} < A_i/2$。

（c）证明若整数 m, n, N 满足 $1 \leqslant m, n \leqslant 2^N$，则欧几里得算法至多进行 2^N 次迭代就可求出 gcd(m, n)。

2.14　欧几里得算法已有 2000 多年的历史，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欧几里得算法竞争的算法。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 $A_{n+1} = |A_n - B_n|$，$B_{n+1} = \min(A_n, B_n)$，$C_{n+1} = C_n$，继续第 $n+1$ 步。

（a）为了获得一些感性认识，请分别用欧几里得算法和 Stein 算法计算 gcd(2152, 764)。

（b）Stein 算法相对于欧几里得算法的一个明显优势是什么？

2.15　（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)。因此，两个算法所需的计算步数大体相当。

（d）证明 Stein 算法的返回值确实是 gcd(A, B)。

2.16　用扩展欧几里得算法求下列乘法逆元：

（a）1234 mod 4321；　　（b）24140 mod 40902；　　（c）550 mod 1769

2.17　本题的目的是判定有多少素数。假定共有 n 个素数，依次为 $p_1 = 2 < p_2 = 3 < p_3 = 5 < \cdots < p_n$。

（a）定义 $X = 1 + p_1 p_2 \cdots p_n$，即所有素数的积加上 1，能够找到一个素数 p_m 整除 X 吗？

（b）m 有何特性？

（c）请推出素数的个数不可能是有限的。

（d）证明 $p_{n+1} \leqslant 1 + p_1 p_2 \cdots p_n$。

2.18　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 $\sum^{d \geqslant 1} \Pr[\gcd(a,b) = d] = 1$。利用上述等式确定 P 的值。提示：利用恒等式 $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$。

2.19　对两个连续整数 n 和 $n+1$，为什么 $\gcd(n, n+1) = 1$？

2.20　利用费马定理计算 $3^{201} \bmod 11$。

2.21 利用费马定理，找一个位于 0 到 72 之间的数 a，使得 a 模 73 与 9794 同余。

2.22 利用费马定理，找一个位于 0 到 28 之间的数 x，使得 x^{85} 模 29 与 6 同余（不能也不必用穷举搜索方法）。

2.23 利用欧拉定理，找一个位于 0 到 9 之间的整数 a，使得 7^{1000} 模 10 与 a 同余（注意这等同于 7^{1000} 的十进制展开的最后一位）。

2.24 利用欧拉定理，找一个位于 0 到 28 之间的整数 x，使得 x^{85} 模 35 与 6 同余（不能也不必用穷举搜索方法）。

2.25 在表 2.6 中，$n>2$ 时 $\phi(n)$是偶数。这一点对所有 $n>2$ 都成立。请证明之。

2.26 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。

2.27 可以证明（参见任何有关数论的书籍），若 $\gcd(m, n)=1$，则 $\phi(mn)=\phi(m)\phi(n)$。利用该性质和上题中的有关性质及对素数 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）$\phi(440)$

2.28 证明，对于任意正整数 a，有

$$\phi(a)=\prod_{i=1}^{t}[p_i^{a_i-1}(p_i-1)]$$

成立，其中 a 由式（2.9）给出，即 $a=p_1^{a_1}p_2^{a_2}\cdots p_t^{a_t}$。

2.29 考虑函数：$f(n)=$ 集合$\{a: 0 \leqslant a<n$，且 $\gcd(a, n)=1\}$中元素的个数。该函数是什么函数？

2.30 中国古代数学家做了很好的工作，提出了中国剩余定理，但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）给出一个奇数但非素数，它不满足上述条件。这个非素数可能非常大，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，中国古代数学家并未测试 $n=341$ 的情形。341 不是素数$(341=11\times31)$但能整除 $2^{341}-2$。证明 $2341\equiv2 \pmod{341}$（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}。

2.31 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”。

2.32 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。

2.33 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 $1\leqslant i, j\leqslant k, i\neq j$，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

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

模 M 有唯一解。证明定理的这种表示形式是正确的。

2.34 下面是孙子用来说明 CRT 的一个例子：

$$x\equiv2 \pmod 3;\ x\equiv3 \pmod 5;\ x\equiv2 \pmod 7$$

试求解 x。

2.35 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，这所大学禁止周日上课（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。

2.36　找出 25 的所有本原根。

2.37　给定 29 的本原根 2，构造离散对数表，并利用该表解下列同余方程：

a．$17x^2 \equiv 10 \pmod{29}$；　b．$x^2-4x-16 \equiv 0 \pmod{29}$；　c．$x^7 \equiv 17 \pmod{29}$

编程习题

2.1　写一个计算机程序实现模 n 的快速指数运算（连续平方）。

2.2　写一个计算机程序实现对用户指定的 n 进行 Miller-Rabin 测试。该程序有两种选择：（1）指定一个可能的证词(witness)a，用 Witness 算法进行测试；或者（2）指定一个随机证词 s，用 Miller-Rabin 测试算法进行检验。

附录 2A　mod 的含义

本书和文献中使用的 mod 运算符有两种使用方式：作为一个二元操作符和作为同余关系。本附录解释这种区别，并精确定义本书中关于括号的标志。这种标志很常见，但却不通用。

二元运算符 mod

如果 a 是整数，n 是非零整数，我们定义 $a \bmod n$ 是 a 被 n 除所得的余数。称整数 n 为模数，称余数为剩余。因此，对于任意整数 a，总有

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

正式地，mod 运算符定义为

$$a \bmod n = a - \lfloor a/n \rfloor \times n, \quad n \neq 0$$

作为一个二元运算符，mod 输入两个整数变量，返回一个余数。例如，7 mod 3 = 1。变量可以是整数、整数型变量或整数变量表达式。例如，如下例子都是正确的，具有明显的意义：

$$7 \bmod 3$$
$$7 \bmod m$$
$$x \bmod 3$$
$$x \bmod m$$
$$(x^2 + y + 1) \bmod (2m + n)$$

其中所有的变量都是整数。对每个例子，左侧被右侧除，所得结果是余数。注意左侧或右侧变量是表达式，表达式需要用括号括起来。mod 运算符不在括号内。

事实上，如果两个变量是任意的实数，mod 运算符也有效，而不仅仅对整数有效。本书中，我们仅关心整数运算。

同余关系 mod

作为一种同余关系，mod 表示两个变量关于一个给定的模数有相同的余数。例如，$7 \equiv 4 \pmod{3}$ 表示 7 和 4 在被 3 除的情况下余数都为 1。如下两个表达式是等价的：

$$a \equiv b \pmod{m} \Leftrightarrow a \bmod m = b \bmod m$$

上式的另一种表述是“说表达式 $a \equiv b \pmod{m}$”和“说 $a-b$ 是 m 的整数倍”是一样的。同样，所有的变量可以是整数、整型变量或整型变量表达式。例如，如下例子都是正确的，具有明显意义

$$7 \equiv 4(\text{mod } 3)$$
$$x \equiv y(\text{mod } m)$$
$$(x^2 + y + 1) \equiv (a + 1)(\text{mod}[m + n])$$

其中所有的变量是整数。此处有两个习惯：同余符号为≡，同余关系的模数放在括号内，在 mod 运算符之后。

同余关系用于定义剩余类。被 m 除后具有相同余数 r 的所有数，形成一个(mod m)的剩余类。有 m 个(mod m)的剩余类。对于一个给定的余数 r，r 所在的剩余类由如下数字组成：

$$r, r \pm m, r \pm 2m, \cdots$$

根据我们的定义，同余

$$a \equiv b(\text{mod } m)$$

表明 a 和 b 相差 m 的整数倍。结果就是，同余式也可以表示为 a 和 b 属于相同的(mod m)剩余类。

第二部分　对称密码

第3章　传统加密技术
第4章　分组密码和数据加密标准
第5章　有限域
第6章　高级加密标准
第7章　分组加密的工作模式
第8章　伪随机数的产生和流密码

第3章　传统加密技术

学 习 目 标

通过本章的学习应具备以下能力：

- 简要介绍对称密码主要概念
- 解释密码分析和穷举攻击的差异
- 理解单表代替密码的操作
- 理解多表代替密码的操作
- 简要介绍 Hill 密码
- 描述转轮机的操作

对称加密，也称传统加密或单钥加密，是 20 世纪 70 年代公钥密码产生之前唯一的加密类型。迄今为止，它仍是两种加密类型中使用最为广泛的加密类型。第一部分讨论了许多对称密码。本章中，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，我们讨论了计算机出现之前的许多算法。最后，简要地介绍了隐写术。第 4 章和第 6 章则探讨当今使用最广泛的两种加密算法：DES 和 AES。

首先，我们来定义一些术语。原始的消息为明文，而加密后的消息为密文。从明文到密文的变换过程被称为加密；从密文到明文的变换过程被称为解密。研究各种加密方案的领域被称为密码编码学。这样的加密方案被称为密码体制或密码。不知道任何加密细节的条件下解密消息的技术属于密码分析学的范畴。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。

3.1　对称密码模型

对称加密方案有五个基本成分（参见图 3.1）：

- **明文**　原始可理解的消息或数据，是算法的输入。
- **加密算法**　加密算法对明文进行各种代替和变换。
- **密钥**　密钥也是加密算法的输入。密钥独立于明文和算法。算法根据所用的特定密钥而产生不同的输出。算法所用的确切代替和变换也依靠密钥。
- **密文**　作为算法的输出，看起来完全随机而杂乱的消息，依赖于明文和密钥。对于给定的消息，不同的密钥产生不同的密文，密文看上去是随机的数据流，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运算。输入密文和密钥，输出原始明文。

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安全。如果有人发现该密钥，而且知道相应的算法，那么就能解读使用该密钥加密的所有通信。

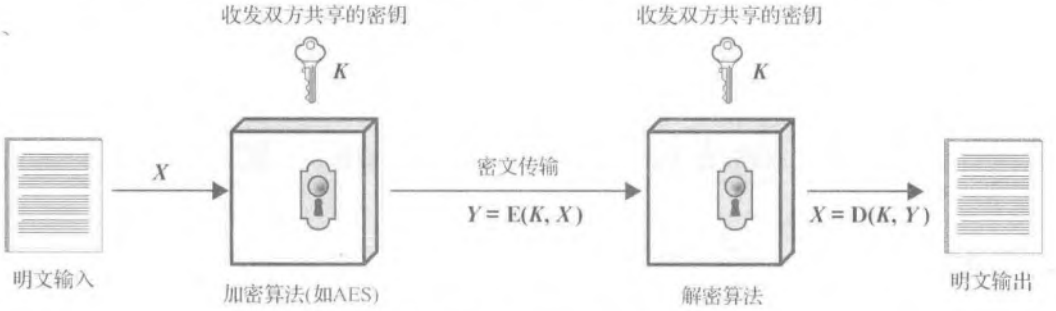


图 3.1　传统密码的简化模型

我们假设基于已知密文和已知加密/解密算法而破译消息是不实际的。换句话说，我们并不需要算法保密，仅需要密钥保密。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，许多产品里都有这种芯片。采用对称密码，首要的安全问题是密钥的保密性。

我们从图 3.2 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。传统上，字母表是由 26 个大写字母组成。而现在，最常用的是二进制字母表{0, 1}。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信道发送到接收方；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。

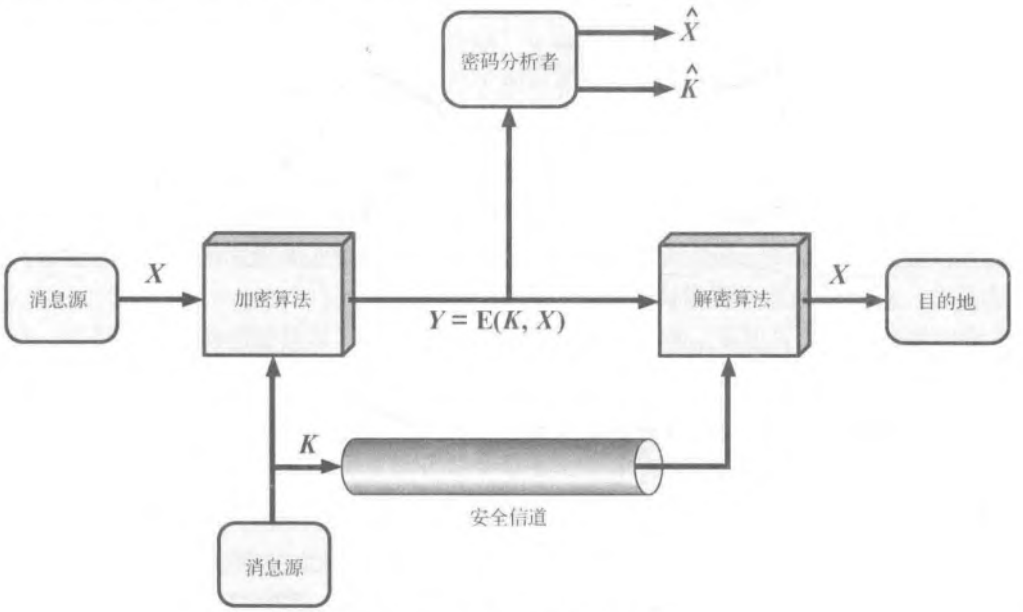


图 3.2　对称密码体制的模型

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 值决定。

拥有密钥 K 的预定接收者，可以进行以下变换：

$$X=\mathrm{D}(K, Y)$$

而得到明文。

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，如果他只是对这个特定信息感兴趣，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来恢复 K。

3.1.1 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：代替和置换，代替是将明文中的每个元素（如位、字母、位组或字母组等）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（即所有的运算是可逆的）。大多数密码体制，也称为乘积密码系统，都使用了多层代替和置换。

（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、秘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、双钥或公钥密码。

（3）**处理明文的方法** 分组密码每次处理输入的一组元素，相应地输出一组元素。流密码则是连续地处理输入元素，每次输出一个元素。

3.1.2 密码分析学和穷举攻击

攻击密码系统的典型目标是恢复使用的密钥而不是仅仅恢复出单个密文对应的明文。攻击传统的密码体制有两种通用的方法：

- **密码分析学** 密码分析学攻击依赖于算法的性质、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定的明文或使用的密钥。
- **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。

如果上述任意一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。

我们首先考虑密码分析学然后讨论穷举攻击。

基于密码分析者知道信息的多少，表 3.1 概括了密码攻击的几种类型。上表中唯密文攻击难度最大。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，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、法文文本，或可执行 EXE 文件、Java 源列表文件、会计文件等等。

唯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，还有，电子金融消息往往

有标准化的文件头或者标志。类似例子还有很多。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。

表 3.1　对加密信息的攻击类型

攻击类型	密码分析者已知的信息
唯密文攻击	● 加密算法 ● 密文
已知明文攻击	● 加密算法 ● 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 ● 密文 ● 分析者选择的明文，及对应的密文（与待解的密文使用同一密钥加密）
选择密文攻击	● 加密算法 ● 密文 ● 分析者选择的一些密文，及对应的明文（与待解的密文使用同一密钥解密）
选择文本攻击	● 加密算法 ● 密文 ● 分析者选择的明文，及对应的密文（与待解的密文使用同一密钥加密） ● 分析者选择的一些密文，及对应的明文（与待解的密文使用同一密钥解密）

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关键词。又比如，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获得信源系统，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，这在附录 S 中会有提及。一般说来，如果分析者有办法选择明文加密，那么他将特意选取那些最有可能恢复出密钥的数据。

表 3.1 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，但是仍然是两种可能的攻击方法。

只有相对较弱的算法才抵挡不住唯密文攻击。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。如果一个密码体制满足条件：无论有多少可使用的密文，都不足以唯一地确定密文所对应的明文，则称该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）所需的信息不在密文里。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。

如果加密体制满足了上述两条标准中的任意一条，则它是计算上安全的。然而，估计攻击者成功破译密文所需的工作量是非常困难的。

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，并能在密文中能找到一些蛛丝马迹。本章中，随着对各种对称密码体制讨论的深入，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。

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。平均地讲，要获得成功必须尝试所有可能密钥的一半。若存在 X 种不同密钥，攻击者平均需要尝试 $X/2$ 次才能找到实际的密钥。值得注意的是穷举攻击不只是简单的尝试所有可能的密钥，除非明文已知，否则分析者需要能够将明文识别为明文。若信息是英文明文则结果很容易得到，即使英文识别是自动化的，若文本信息加密前被压缩了则识别将更困难，若信息是一些更通用类型的数据（如数值文件）且被压缩了，自动化会变得更困难。因此，采用穷举攻击时需要一些关于预期明文的信息和自动的从混淆中区分明文的方法。

3.2 代替技术

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弄清楚今天所用的对称密码的一些基本方法，以及随之而来的密码攻击的类型等。

首先讨论一下所有加密技术都要用到的两个基本模块：代替和置换，我们将在接下来的两节里讨论这些内容。最后将讨论将两种技巧综合应用的密码系统。

代替技术是将明文字母替换成其他字母、数字或符号的方法[①]。如果把明文看成是二进制序列的话，那么代替就是用密文位串来代替明文位串。

3.2.1 Caesar 密码

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

```
明文：meet me after the toga party
密文：PHHW PH DIWHV 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。我们通过列出所有的字母来定义如下变换：

```
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，代替成密文字母 C[②]：

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

移位可以是任意整数 k，这样就得到了一般的 Caesar 算法

$$C = E(k, p) = (p + k) \bmod (26) \tag{3.1}$$

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

② 定义 $a \bmod n$ 为 a 除 n 的余数。例如，11 mod 7=4。关于模算术的更多讨论参考第 2 章。

这里 k 的取值范围从 1 到 25。解密算法是

$$p = \mathrm{D}(k, C) = (C - k) \bmod (26) \tag{3.2}$$

如果已知某给定的密文是 Caesar 密码，那么穷举攻击是很容易实现的：只要简单地测试所有的 25 种可能的密钥。图 3.3 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。

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 sqd 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 3.3　对 Caesar 密码的穷举密码学分析

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 7 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。

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，也就更不可识别了。例如图 3.4 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替密码来加密（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  Åä£èü3Å
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  Å◊¿″Zù-
Ω¨Õ¯6Œÿ{% „ΩÊó ¸ï π÷ÁÎ°úO2çSŷ´O-
2Äflßi /@^″∏K°ªPŒπ¸úé^´3∑˜ö˘ÔZÌ"Y¬ŸΩœY> Ω+eô/·<K£¿*÷˜"≤û˜
B ZøK˜Qßÿüƒ¸!ÒflÎzsS/]>ÈQ ü
```

图 3.4　经过压缩的文本样本

3.2.2　单表代替密码

Caesar 密码仅有 25 种可能的密钥，是远不够安全的。通过允许任意代替，密钥空间将会急剧增大。在继续之前，我们先定义术语置换。有限元素的集合 S 的置换是 S 的所有元素的有序排列，且每个元素只出现一次。例如，如果 S = {a,b,c}，则 S 有 6 个置换

abc,acb,bac,bca,cab,cba

一般地，具有 n 个元素的集合有 $n!$ 个置换，因为第一个元素有 n 种选择方式，第二个有 $n-1$ 种方式，以此下去得到结论。

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，好像可以抵挡穷举攻击了。这种方法被称为单表代替

```
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，参见图 3.5（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 3.5 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 a,h,i,n,o,r,s 中的某一个相对应。相对频率较低的字母（即 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。

此时我们可以从以下几种方法入手。可以尝试着做一些代替，填入明文，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。

统计双字母组合的频率是一个很有效的工具。由此可以得到一个类似于图 3.5 的双字母组合的相对频率图。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，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。

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，如果是这样，则 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
```

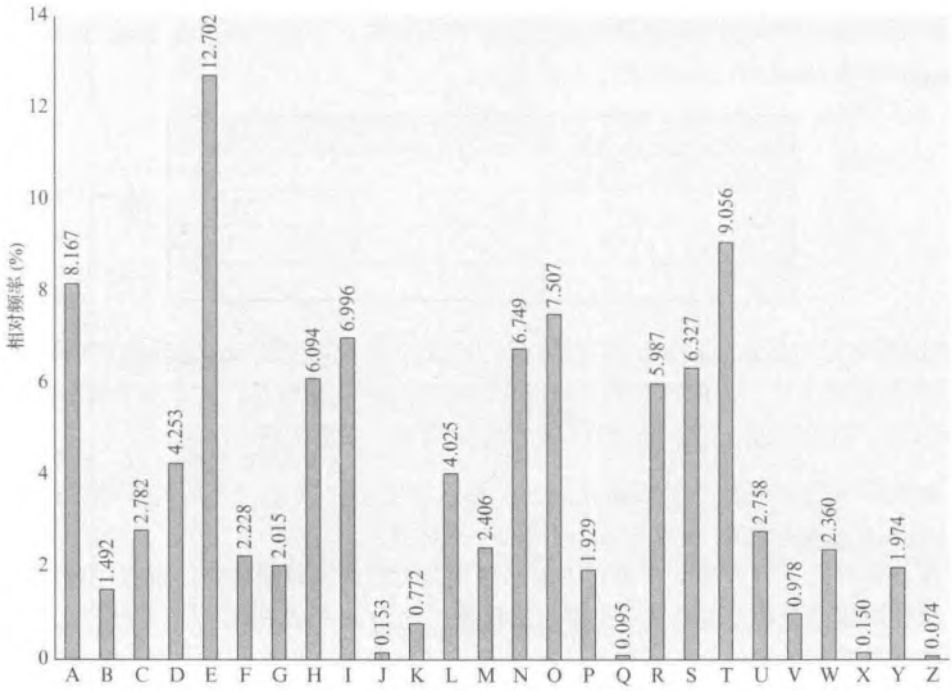


图 3.5　英文字母的相对频率

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
```

单表代替密码较容易被攻破，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替，称为同音词（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，循环或随机地选取其中一个同音词即可。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破坏了。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，从而使得密码分析相对简单。

有两种主要的方法可以减少代替密码里明文结构在密文中的残留度：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替密码。每一种我们都会简单地提及。

3.2.3　Playfair 密码

最著名的多字母代替密码是 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]。

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。

Playfair 算法是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 在 Dorothy Sayers 所著 *Have His Carcase*[①] 一书里给出。

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（去掉重复字母）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 暂且当成一个字母。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替，每行中最右边的一个字母就用该列中最左边的第一个字母来代替，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替，每行中最下面的一个字母就用该列中最上面的第一个字母来代替，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替，该字母所在行为密文所在行，另一字母所在列为密文所在列。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，尽管只有 26 个字母，但有 26×26 = 676 个字母对，因此识别出单个字母对要困难得多。而且，单个字母的相对频率比字母对的相对频率变化的幅度大，在统计规律上要好，这样利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战时加密体制，并且在第二次世界大战中，美军及其他一些盟军仍在大量使用。

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好地保留了明文语言的大部分结构特征。几百个字母的密文就足够我们分析出规律了。

图 3.6 显示了 Playfair 密码和其他一些密码加密的有效性。标有“明文”的曲线画出了 26 个字母的频率分布（不区分大小写）。这也是任意单表代替密码的频率分布，因为单字母的频率值是相同的，不同的是对原始字母用了不同的字母进行代替。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 9.056/12.702 ≈ 0.72，等等。水平轴上的点对应着按使用频率递降的字母。

图 3.6 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替密码是一种简单的方法。如果频率分布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，唯密文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。图中也展示了随后将讨论的 Vigenère 密码，图中 Hill 和 Vigenère 的曲线是基于文献[STMM93]中的结果。

① 本书对可能字攻击的描述引人入胜。

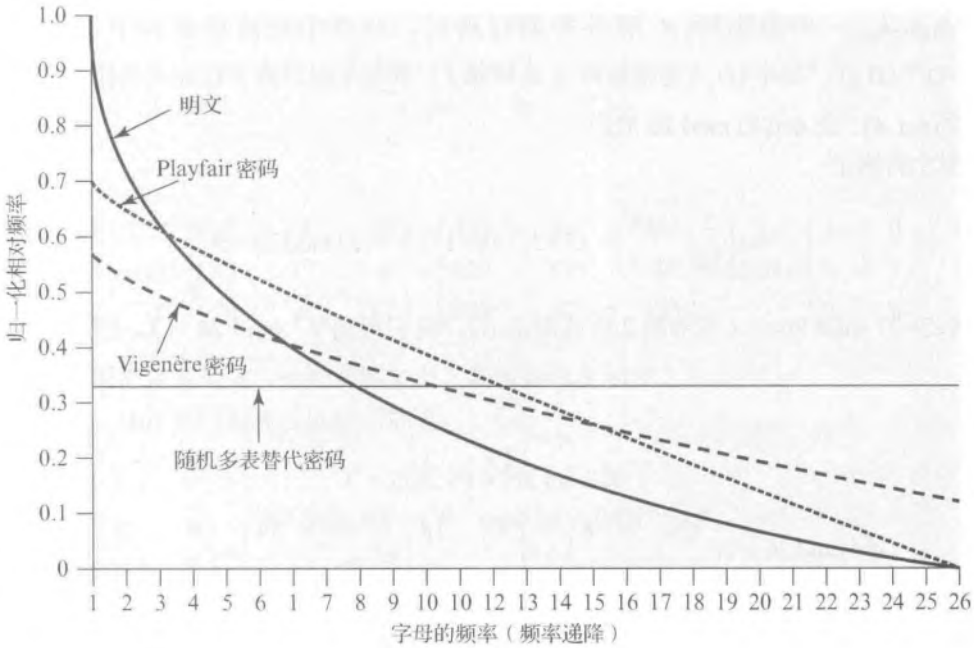


图 3.6　字母出现的相对频率

3.2.4　Hill 密码[①]

另一个有趣的多表代替密码是 Hill 密码，由数学家 Lester Hill 在 1929 年发明。

线性代数里的一些概念　在描述 Hill 密码前，让我们先简短地复习一下线性代数里的一些术语。现在我们关心模 26 的矩阵算术。对于需要补习矩阵乘法和逆运算的读者，请参考附录 E。

我们定义方阵 $\boldsymbol{M}$ 的逆矩阵 $\boldsymbol{M}^{-1}$ 应满足 $\boldsymbol{M}(\boldsymbol{M}^{-1}) = (\boldsymbol{M}^{-1})\boldsymbol{M} = \boldsymbol{I}$，其中 $\boldsymbol{I}$ 为单位矩阵。$\boldsymbol{I}$ 矩阵是除了主对角线上的元素为 1 外其余全为 0 的矩阵。一个矩阵的逆矩阵并不总是存在，但当存在的时候，它满足前述的方程。例如

$$\boldsymbol{A} = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \qquad \boldsymbol{A}^{-1} \bmod 26 = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \begin{pmatrix} (5\times 9)+(8\times 1) & (5\times 2)+(8\times 15) \\ (17\times 9)+(3\times 1) & (17\times 2)+(3\times 15) \end{pmatrix}$$

$$= \begin{pmatrix} 53 & 130 \\ 156 & 79 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

为了解释如何计算矩阵的逆矩阵，我们先介绍行列式的概念。对于任意一个方阵(m×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，部分项的前面有负号。对于 2×2 的矩阵

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

行列式是 $k_{11}k_{22} - k_{12}k_{21}$。对于 3×3 的矩阵，行列式的值为 $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} -$

① 该密码理解起来比本章的其他密码稍微困难一些，但它阐明了密码分析学里有用的一些要点。初学者可以跳过这一节。

$k_{21}k_{12}k_{23}-k_{11}k_{32}k_{23}$。如果矩阵 $\boldsymbol{A}$ 有非零的行列式，则这个矩阵的逆如下计算：$[\boldsymbol{A}^{-1}]_{ij}=(\det \boldsymbol{A})^{-1}(-1)^{i+j}(\mathrm{D}_{ji})$，其中 (D_{ji}) 是将矩阵 $\boldsymbol{A}$ 去掉第 j 行和第 i 列后的子行列式的值，det(**A**)是 **A** 的行列式，而$(\det \boldsymbol{A})^{-1}$是 $\det(\boldsymbol{A})$ mod 26 的逆。

继续刚才的例子

$$\det\begin{pmatrix}5 & 8\\ 17 & 3\end{pmatrix}=(5\times 3)-(8\times 17)=-121 \bmod 26=9$$

因为 9×3=27 mod 26=1（参考第 2 章或附录 E），我们得到 9^{-1} mod 26 = 3。因此，如下计算 $\boldsymbol{A}$ 的逆：

$$\boldsymbol{A}=\begin{pmatrix}5 & 8\\ 17 & 3\end{pmatrix}$$

$$\boldsymbol{A}^{-1} \bmod 26=3\begin{pmatrix}3 & -8\\ -17 & 5\end{pmatrix}=3\begin{pmatrix}3 & 18\\ 9 & 5\end{pmatrix}=\begin{pmatrix}9 & 54\\ 27 & 15\end{pmatrix}=\begin{pmatrix}9 & 2\\ 1 & 15\end{pmatrix}$$

Hill 算法 该加密算法将 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（$a=0$，$b=1$，…，$z=25$）。例如 $m=3$，系统可以描述为

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

用行向量和矩阵表示如下[①]：

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

或

$$\boldsymbol{C}=\boldsymbol{PK} \bmod 26$$

这里 $\boldsymbol{C}$ 和 $\boldsymbol{P}$ 是长度为 3 的行向量，分别代表密文和明文，$\boldsymbol{K}$ 是一个 3×3 矩阵，代表加密密钥。运算按模 26 执行。

例如，明文为“paymoremoney”，加密密钥为

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

明文的前面 3 个字母用向量(15 0 24)表示，那么(15 0 24) $\boldsymbol{K}$=(303 303 531) mod 26 = (17 17 11) = RRL，照此方式转换余下字母，可得整段明文对应的密文是 RRLLMWBKASPDH。

解密则需要用到矩阵 $\boldsymbol{K}$ 的逆。我们可以计算 det $\boldsymbol{K}$=23，所以$(\det \boldsymbol{K})^{-1}$ mod 26 =17。计算逆矩阵为：

① 一些密码书将明文和密文表达为列向量，以使得列向量放在矩阵的后面而不是像行向量放在矩阵的前面。Sage 使用了行向量，所以我们也采用这个习惯。

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

验证如下：

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

很显然，如果把逆矩阵 $\boldsymbol{K}^{-1}$ 应用到密文上，则可以恢复明文。

用一般术语，Hill 密码系统可以表示如下：

$$\boldsymbol{C}=\mathrm{E}(\boldsymbol{K},\boldsymbol{P})=\boldsymbol{PK}\bmod 26$$
$$\boldsymbol{P}=\mathrm{D}(\boldsymbol{K},\boldsymbol{C})=\boldsymbol{CK}^{-1}\bmod 26=\boldsymbol{PKK}^{-1}=\boldsymbol{P}$$

同 Playfair 密码相比，Hill 密码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 密码足以抗唯密文攻击，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，假如有 m 个明密文对，每个长度都是 m，定义 $\boldsymbol{P}_j=(p_{1j}\,p_{1j}\cdots p_{mj})$和 $\boldsymbol{C}_j=(c_{1j}\,c_{1j}\cdots c_{mj})$，使得对每个 $\boldsymbol{C}_j$和 $\boldsymbol{P}_j\boldsymbol{K}$（$1\leqslant j\leqslant m$）都有 $\boldsymbol{C}_j=\boldsymbol{P}_j\boldsymbol{K}$，其中 $\boldsymbol{K}$ 是未知的矩阵形密钥。现在定义两个 $m\times m$ 的矩阵 $\boldsymbol{X}=(p_{ij})$和 $\boldsymbol{Y}=(c_{ij})$。那么我们可以得出矩阵等式 $\boldsymbol{Y}=\boldsymbol{XK}$，若 $\boldsymbol{X}$ 可逆，则可得 $\boldsymbol{K}=\boldsymbol{X}^{-1}\boldsymbol{Y}$。若 $\boldsymbol{X}$ 不可逆，那么可以另找 $\boldsymbol{X}$ 和对应的 $\boldsymbol{Y}$，直至得到一个可逆的 $\boldsymbol{X}$。

考虑如下例子。假设明文“hillcipher”经过一个 2×2 的 Hill 密码加密生成密文 HCRZSSXNSP。因此我们知道(7　8)$\boldsymbol{K}$ mod 26 = (7　2); (11　11)$\boldsymbol{K}$ mod 26 = (17　25)；以此类推。那么由前两个明、密文对，可得

$$\begin{pmatrix}7 & 2\\ 17 & 25\end{pmatrix}=\begin{pmatrix}7 & 8\\ 11 & 11\end{pmatrix}\boldsymbol{K}\bmod 26$$

$\boldsymbol{X}$ 的逆是

$$\begin{pmatrix}7 & 8\\ 11 & 11\end{pmatrix}^{-1}=\begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}$$

因此

$$\boldsymbol{K}=\begin{pmatrix}25 & 22\\ 1 & 23\end{pmatrix}\begin{pmatrix}7 & 2\\ 17 & 25\end{pmatrix}=\begin{pmatrix}549 & 600\\ 398 & 577\end{pmatrix}\bmod 26=\begin{pmatrix}3 & 2\\ 8 & 5\end{pmatrix}$$

该结果可以由剩下的明密文对来验证。

3.2.5　多表代替加密

对简单单表代替的改进方法是在明文消息中采用不同的单表代替。这种方法一般称之为多表代替密码。所有这些方法都有以下的共同特征：

（1）采用相关的单表代替规则集。

（2）密钥决定给定变换的具体规则。

Vigenère 密码 多表代替密码中最著名的和最简单的是 Vigenère 密码。它的代替规则集由 26 个 Caesar 密码的代替表组成，其中每一个代替表是对明文字母表移位 0～25 次后得到的代替单表。每个密码由一个密钥字母来表示，这个密钥字母用来代替明文字母 a，故移位 3 次的 Caesar 密码由密钥值 d 来代表。

我们可用如下方式来表述 Vigenère 密码。假设明文序列为 $P=p_0,p_1,p_2,\cdots,p_{n-1}$，密钥由序列 $K=k_0,k_1,k_2,\cdots,k_{m-1}$ 构成，其中典型的是 $m<n$。密码序列 $C=C_0,C_1,C_2,\cdots,C_{n-1}$ 计算如下：

$$\begin{aligned}C=C_0,C_1,C_2,\cdots,C_{n-1}&=E(K,P)=E\left[(k_0,k_1,k_2,\cdots,k_{m-1}),(p_0,p_1,p_2,\cdots,p_{n-1})\right]\\&=(p_0+k_0)\bmod 26,(p_1+k_1)\bmod 26,\cdots,(p_{m-1}+k_{m-1})\bmod 26,\\&\quad(p_m+k_0)\bmod 26,(p_{m+1},k_1)\bmod 26,\cdots,(p_{2m-1},k_{m-1})\bmod 26,\cdots\end{aligned}$$

因此，密钥的第一个字母模 26 加到了明文的第一个字母，接着是第二个字母，以此类推，直到前 m 个明文处理完毕。对于第二组的 m 个明文，重复使用密钥字母。继续该过程直到所有的明文序列都被加密完。加密过程的一般方程是

$$C_i=(p_i+k_{i\bmod m})\bmod 26 \tag{3.3}$$

该式和 Caesar 密码的式（3.1）式类似。本质上，每一个明文字母根据相应的密钥字母用不同的 Caesar 密码加密。类似地，解密是式（3.2）的推广

$$p_i=(C_i-k_{i\bmod m})\bmod 26 \tag{3.4}$$

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

```
密钥:   deceptivedeceptivedeceptive
明文:   wearediscoveredsaveyourself
密文:   ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

用数字来表述，有如下结果。

密钥	3	4	2	4	15	19	8	21	4	3	4	2	4	15
明文	22	4	0	17	4	3	8	18	2	14	21	4	17	4
密文	25	8	2	21	19	22	16	13	6	17	25	6	21	19

密钥	19	8	21	4	3	4	2	4	15	19	8	21	4
明文	3	18	0	21	4	24	14	20	17	18	4	11	5
密文	22	0	21	25	7	2	16	24	6	11	12	6	9

这种密码的强度在于每个明文字母对应着多个密文字母，且每个密文字母使用唯一的密钥字母，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 3.6 给出了 9 个密钥词的 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。

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替或 Vigenère 密码来加密的。可以用一个简单的测试来做个区分。如果用单表代替，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 3.5，应该有一个密文字母的出现频率大约是 12.7%，另一个大约是 9.06%，等等。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刚好吻合。然而当它们的统计规律很接近时，就认为是用单表代替加密的。

另一方面，如果认为是用 Vigenère 密码加密的，破译能否取得进展将取决于能否判定密钥词的长度，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。能够导致答案的重要观察是：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”。我们对相关的密文字母做了下画线处理，对相关的密文数字用阴影处理以指示上述说明。

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）推断密钥词的长度是 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。然而如果信息长度足够大，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 m，那么密码实际上包含了 m 个单表代替。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”，它将密钥词和明文自身连接来生成不重复的密钥词。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 3.5 可以估计出其发生的概率为 $(0.127)^2 \approx 0.016$，而用 t 加密 t 发生的概率大概只有它的一半等。密码分析者利用这些规律能够成功地进行分析[①]。

Vernam 密码　最终的反破译措施也许只有选择一个和明文毫无统计关系，且和它一样长的密钥。1918 年 AT&T 公司的工程师 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（参见图 3.7）可以简明地表述为

$$c_i = p_i \oplus k_i$$

其中

p_i 是明文第 i 个二进制位。

k_i 是密钥第 i 个二进制位。

c_i 是密文第 i 个二进制位。

$\oplus$ 是异或运算符。

该式与 Vigenère 密码的式（3.3）类似。因此密文是通过对明文和密钥的逐位异或而成的。根据异或运算的性质，解密过程为

$$p_i = c_i \oplus k_i$$

该式和式（3.4）类似。

这种技术的本质在于构造密钥的方式。Vernam 提出使用连续的磁带，其最终也将循环。所以事实上该体制是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。

---

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解的。当对现代密码算法做出类似的论断时，这是值得汲取的教训。

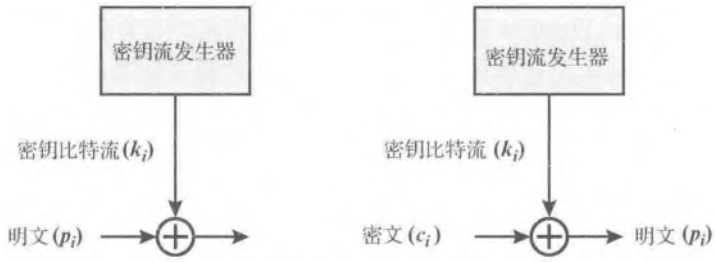


图 3.7 Vernam 密码

3.2.6 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，另外，密钥只对一个消息进行加解密，之后丢弃不用。每一条新消息都需要一个与其等长的新密钥。这就是著名的一次一密，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。假设使用的是 27 个字符（第 27 个字符是空格）的 Vigenère 密码，但是这里使用的一次性密钥和消息一样长。请看下面的密文：

```
ANKYODKYUREPFJBYOJDSPLREYIUDNOFDOIUERFPLUYTS
```

现在我们用两种不同的密钥解密：

```
密文：ANKYODKYUREPFJBYOJDSPLREYIUDNOFDOIUERFPLUYTS
密钥：pxlmvmsydoftyrvzwc tnlebnecvgdupahfzzlmnyih
明文：mr mustard with the candlestick in the hall

密文：ANKYODKYUREPFJBYOJDSPLREYIUDNOFDOIUERFPLUYTS
密钥：pftgpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
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呢（即正确的密钥）？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哪一种密钥更有可能。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，我们不再需要寻找密码了，一次一密提供了完全的安全性，但是在实际中，一次一密存在两个基本难点：

（1）产生大规模随机密钥有实际困难。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，一次一密实际很少使用，主要用于安全性要求很高的低带宽信道。

一次一密是唯一的具有完善保密的密码体制。附录 F 探讨了这一概念。

3.3 置换技术

到目前为止我们所讨论的都是将明文字母代替为密文字母。与之极不相同的一种加密方法是对明文进行置换，这种密码称为置换密码。

最简单的例子是栅栏技术，按照对角线的顺序写出明文，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。一个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:    TTNAAPTMTDUOAODWCDIXKNLYPETZ
```

因此在本例中，密钥是 4312567。为了加密，从标号为 1 的那列开始，本例中为第 3 列。写下那列的所有字母，接着是标号为 2 的列，即第 4 列，接着第 2 列，第 1 列，第 5，6，7 列。

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别。对于上述所示的列变换类型密码，密码分析很直观，可以从将密文排列成矩阵入手，再来处理列的位置。双字母音节和三字母音节频率表分析可以派上用场。

多次置换密码相对来讲要安全得多。这种复杂的置换是不容易重构的。前面那条消息用相同算法再加密一次

```
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替原始明文信息。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
```

这还是多少有些规律性，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。

3.4 转轮机

刚才的例子表明用多次置换得到的算法对密码分析来说有很大的难度。这对代替密码也适用。在介绍 DES 之前，先来介绍多层加密原理的一个重要应用例子：轮转机密码系统[①]。

转轮机的基本原理如图 3.8 所示。转轮机包括一组相互独立的旋转圆筒，电脉冲可以通过它。每个圆筒有 26 个输入引脚和 26 个输出引脚。内部连线使每一个输入仅同唯一一个输出连接，为简明起见，图中只画出了三条内部连接。

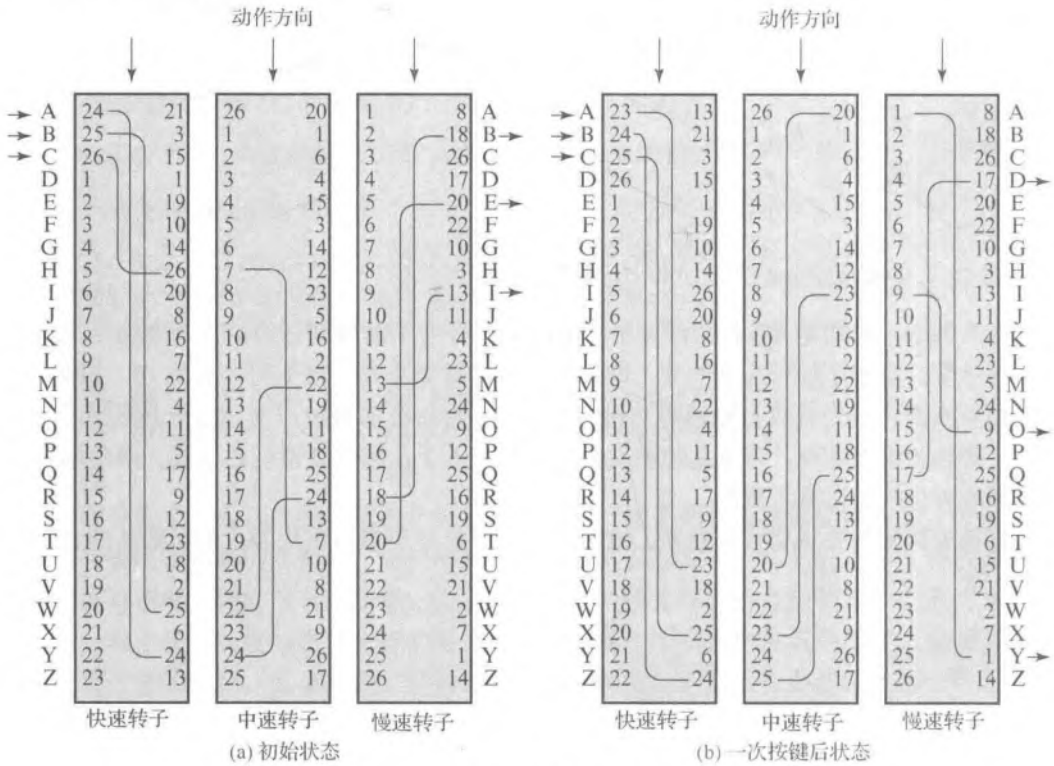


图 3.8 用编号表示内部引线的三转子机

如果把每个输入输出引脚当成字母表中的一个字母，那么一个圆筒就定义了一个单表替换。如图 3.7 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。

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

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。

相应改变了。因此就定义了不同的单表代替密码。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 的多表代替算法。

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 3.8 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，第一个圆筒就旋转一个引脚的位置。图 3.8 的右半部分表明了经过一次按键后的情况。内层的（第一个）圆筒旋转完一圈之后，中间的（第二个）圆筒就转一个引脚的位置。最后，当中间的圆筒转完一圈后，外层的圆筒旋转一个引脚的位置。它的原理就像里程表。所以整个系统重复使用 26×26×26＝17576 个不同的替换字母表，如果用 4 个或 5 个转轮则周期数将分别为 456 976 和 11 881 376。因此，一个五筒转轮机相当于密钥长度为 11 881 376 的 Vigenère 密码。

这样一个方案呈现出一个对于密码分析来说很难对付的挑战。例如，假如分析者尝试使用字母频率分析方法相当于面对超过 1.1 千万的多表密码，这使每个多表密码需要 50 个字母，也就是分析者需要掌握长度超过 10 亿个字母的密文。

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 4 章中对 DES 进行讨论。

3.5　隐写术

我们讨论一种技术来结束这一章，严格说来，它并不是加密，而是隐写术。

有两种办法可用来隐藏明文信息。隐写术，它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]。

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 3.9　Morse 侦探的疑惑

[引自于 Colin Dexter 的《沉默世界》的（Nicholas Quinn）]

一种简单却很耗时的隐写术可由一段明显无伤大雅的文本字词重新排列而成。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 3.9 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。这个并不难，看看你能否解密它。

① 隐写术是一个过时的说法，现在 David Kahn 将其修正并且给定了其含义，参见文献[KAHN96]。

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[WAYN09]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 3096×6144，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 130 kB 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，比如具体的加入方法由密钥来决定（参见习题 3.22）。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如下的情况：通信双方宁愿内容丢失，也不愿他们进行秘密通信的事实被人发现。加密标识信息也是重要和秘密的，通过它可以找出想进行消息隐藏的发送方或接收方。

3.6 关键术语、思考题和习题

关键术语

分组密码
穷举攻击
Caesar 密码
密码
密文
计算安全
传统加密
密码分析学
密码编码系统
密码编码学
密码学
解密过程
解密
双字母
加密过程
加密
Hill 密码
单表代替密码
一次一密
明文
Playfair 密码
多表代替密码
栅栏密码
单钥加密
隐写术
流密码
对称加密
置换密码
无条件安全
Vigenère 密码

思考题

3.1 什么是对称密码的本质成分？
3.2 密码算法中两个基本函数是什么？
3.3 用密码进行通信的两个人需要多少密钥？
3.4 分组密码和流密码的区别是什么？
3.5 攻击密码的两种一般方法是什么？
3.6 列出并简要的定义基于攻击者所知道信息的密码分析攻击类型。
3.7 无条件安全密码和计算上安全密码的区别是什么？
3.8 简要地定义 Caesar 密码。
3.9 简要地定义单表代替密码。
3.10 简要地定义 Playfair 密码。

3.11　单表代替密码和多表代替密码的区别是什么？

3.12　一次一密的两个问题是什么？

3.13　什么是置换密码？

3.14　什么是隐写术？

习题

3.1　仿射 Caesar 密码，即 Caesar 密码的一种推广，具有如下定义：对于每一个明文 p，用密文 C 代替，其中

$$C = E([a, b], p) = (ap + b) \bmod 26$$

对加密算法的一个基本要求是算法是单射的，即如果 $p \neq q$，则 $E(k, p) \neq E(k, q)$。否则，就会因为有很多的明文映射成相同的密文而不能解密。仿射 Caesar 密码并不是对所有的 a 都是一对一的映射。例如，当 $a = 2$，$b = 3$ 时，有 $E([a, b], 0) = E([a, b], 13) = 3$。

（a）对 b 的取值是否有限制？解释原因。

（b）判定 a 不能取哪些值？

（c）分析 a 可以取哪些值，不可以取哪些值？并给出理由。

3.2　有多少种仿射 Caesar 密码？

3.3　用仿射 Caesar 密码加密得到一份密文。频率最高的字母为“B”，次高的字母为“U”，请破译该密码。

3.4　已知下面的密文由简单的代替算法产生：

```
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）出现频率最高的字符应该代表 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。

3.5　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。这里就从一本涉及编码的富于悬念的侦探小说——Ruth Rendell 的《与陌生人的谈话》（Talking to Strange Men）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》（The Other Side Silence）（一本有关侦探 Kim Philby 的小说）一书的第一句话和单表代替方法产生的，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.

使用的是简单的代替密码。

（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？

3.6 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？

3.7 这是真实世界的一个例子，来自以前的美国特种兵手册（公开部分）。可从本书的网址下载。

（a）用记忆词 *cryptographic* 和 *network security* 作为两个密钥来加密如下消息：

Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends.

如何处理记忆词中的冗余字母，多余的字母，如何处理空格和标点符号？请做出合理的假设。

注：这条消息摘自 Sherlock Holmes 的小说 The Sign of Four。

（b）解密密文，说明你是如何做的。

（c）评议一下何时采用这种技术，有何优势？

3.8 普通的单表代替密码的一个缺点是：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

```
明文：  abcdefghijklmnopqrstuvwxyz
密文：  CIPHERABDFGJKLMNOQSTUVWXYZ
```

如果感觉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 3.2 节的例子（该例子以“it was disclosed yesterday”开头）中使用了这种方法。试确定密钥词。

3.9 当海军上尉 John F. Kennedy 管理的美国巡逻船 PT-109 被日本毁灭者击沉时，位于澳大利亚的一个无线站截获的一条用 Playfair 密码加密的消息：

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

密钥为 *royal new zealand navy*，请解密这条消息，将 TT 换为 tt。

3.10 （a）用密钥 *largest* 构造一个 Playfair 矩阵。

（b）用密钥 *occurrence* 构造一个 Playfair 矩阵。对密钥里的冗余字母的处理方法做出合理的假设。

3.11 （a）使用下述 Playfair 矩阵加密消息：Must see you over Cadogan West. Coming at once。

注：该消息摘自 Sherlock Holmes 的故事《Bruce-Partington 计划冒险记》。

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

（b）用习题 3.10(a)里的 Playfair 矩阵重做习题 3.11(a)。

（c）对这个习题的结果你如何解释？所得结论能做一般性推广吗？

3.12 （a）Playfair 密码有多少可能的密钥？那些会产生相同加密结果的密钥也计算在内。将结果用 2 的幂形式表示，取最佳逼近。

（b）除去那些会产生相同加密结果的密钥，那么 Playfair 密码有多少有效的唯一密钥？

3.13 在代替密码里，使用 25×1 的 Playfair 矩阵，结果会如何？

3.14 （a）用 Hill 密码加密消息“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）写出从密文恢复明文所做的解密计算。

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

3.16 已经证明了 Hill 密码里的矩阵 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ 需要满足（*ad–bc*）与 26 互素。也就是说（*ad–bc*）与 26 的唯一正整数因子是 1。因此，如果（*ad–bc*）为 13 或偶数，这样的矩阵不合格。运用下面的方法而不是逐个去数，给出 2×2 Hill 密码密钥的个数。

（a）判定如下矩阵的个数：判别式为偶数，且其一行或两行都是偶的（称一行为偶的是指该行的所有元素都为偶数）。

（b）判定如下矩阵的个数：判别式为偶数，且其一列或两列都是偶的。

（c）判定如下矩阵的个数：判别式为偶数，且其所有元素为奇数。

（d）考虑重叠情况，找出判别式为偶的矩阵总数。

（e）判定如下矩阵的个数：判别式为 13 的倍数，且其第一列为 13 的倍数。

（f）判定如下矩阵的个数：判别式为 13 的倍数，且其第一列不为 13 的倍数，但第二列在模 13 的意义下是第一列的倍数。

（g）计算判别式为 13 的倍数的所有矩阵个数。

（h）判定如下矩阵的个数：判别式为 26 的倍数，且满足情况（a）和（e），（b）和（e），（c）和（e），（a）和（f），等等。

（i）计算出判别式既不是 2 的倍数也不是 13 的倍数的矩阵总数。

3.17 计算模 26 行列式

（a） $\begin{pmatrix} 20 & 2 \\ 5 & 4 \end{pmatrix}$　　（b） $\begin{pmatrix} 1 & 7 & 22 \\ 4 & 9 & 2 \\ 1 & 2 & 5 \end{pmatrix}$

3.18 计算模 26 的逆

（a）$\begin{pmatrix} 2 & 3 \\ 1 & 22 \end{pmatrix}$　　（b）$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$

3.19 用 Vigenère 密码加密单词“explanation”，密钥为“leg”。

3.20 本题可显示出 Vigenère 密码的一次一密版本的用途。该方案中，密钥是位于 0～26 的随机数字。例如，如果密钥是 3 19 5 …，则明文的第一个字母使用 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，以此类推。

（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。

3.21 隐藏在图 3.9 中的消息是什么？

3.22 《女演讲者的神秘》一书中有一个情节，Lord Peter 面对图 3.10 的消息产生了困惑。他还发现加密消息的密钥是一个整数序列

```
787656543432112343456567878878765654
343211234345656787887876565443321 1234
```

（a）解密该消息，指出其中最大的整数。

（b）如果知道加密算法，但不知道密钥，这个加密方案还安全吗？

（c）如果知道密钥，但不知道加密算法，这个加密方案还安全吗？

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 3.10 Lord Peter 的困惑

编程习题

3.23 编写一个加密解密程序实现广义 Caesar 密码，该密码也称为加法密码。

3.24 编写一个加密解密程序实现习题 3.1 里的仿射密码。

3.25 写一个程序，无须人工干预，对加法密码实现字母频率攻击。软件按可能性大小的顺序给出可能的明文。如果用户界面允许用户定义“给出前 10 个可能明文”，则更好。

3.26 写一个程序，无须人工干预，对单表代替密码实现字母频率攻击。软件按可能性大小的顺序给出可能的明文。如果用户界面允许用户定义“给出前 10 个可能明文”，则更好。

3.27 写一个程序实现 2×2 Hill 密码的加解密算法。

3.28 写一个程序实现对 m 维 Hill 密码的已知明文攻击。算法的效率如何，用 m 的函数表示。

第 4 章　分组密码和数据加密标准

学 习 目 标

通过本章的学习应具备以下能力：

- 理解流密码和分组密码之间的差异
- 简要介绍 Feistel 密码，并解释解密为何是加密的逆过程
- 简要介绍数据加密标准（DES）
- 解释雪崩效应
- 讨论 DES 的密码学强度
- 总结主要的分组密码设计原理

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（the Data Encryption Standard，DES）。尽管 DES 之后涌现了大量的对称密码，尽管 DES 注定要被高级加密标准（Advanced Encryption Standard，AES）所取代，但它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所涉及的原理。

首先，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主要是指对称分组密码（除第 8 章的流密码外）。接下来我们研究完整的 DES。讨论过这个具体的算法后，再回到对分组密码设计问题的更一般性讨论上来。

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，不能像 RSA 或类似的算法给予简洁的描述。因此，读者可能会希望以简化的 DES 开始，简化 DES 在附录 G 里给出。对于这个简化版本，读者可以手工加解密，从而对算法的详细工作过程有较好的理解。课堂教学经验表明对简化版本的研究能提高对 DES 的理解[①]。

4.1　传统分组密码结构

事实上，现在使用的大多数对称分组加密算法都是基于 Feistel 分组密码结构的[FEIS73]。因为这个原因，研究 Feistel 密码的设计原理就很重要了。我们先比较流密码和分组密码，然后讨论 Feistel 结构分组密码的设计动机，最后来讨论它的一些含义。

4.1.1　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。理想情况下，可以使用一次一密版本的 Vernam 密码（参见图 3.7），其中密钥流(k_i)和明文位流(p_i)一样长。如果密钥流是随机的，除非是获得了密钥流，否则这个密码是不可破的。然

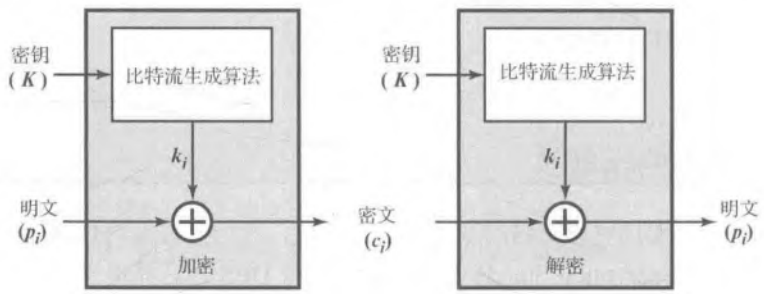
① 然而，你也可以跳过附录 G，至少初次阅读时可以这样做。如果因为陷入 DES 的技术细节而迷惘，你可以回过头来从简化的 DES 开始。

而，密钥流必须提前以某种独立、安全的信道提供给双方。如果待传递的数据流量很大，这就带来了一个不可逾越的障碍。

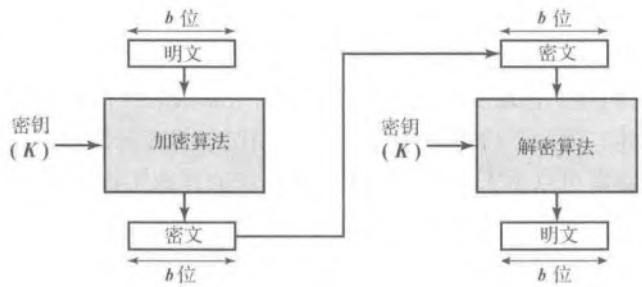
相应地，出于实用的原因，位流必须以算法程序的方式实现，从而双方都可以生产具有密码学意义的位流。在这种方法中［参见图 4.1(a)］，位流发生器是一个由密钥控制的算法，它必须产生在密码学意义上讲是强壮的位流。现在，两个用户只需要共享生成密钥，则各自可以生产密钥流。

分组密码是将一个明文分组作为整体加密并且通常得到的是与明文等长的密文分组。典型的分组大小是 64 位或 128 位。同流密码一样，两个用户要共享一个对称加密密钥［参见图 4.1(b)］。第 7 章中还要讲到，使用某些工作模式，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，本章及本书我们所讨论的对称密码将集中在分组密码的讨论上。



(a) 使用算法比特流发生器的流密码



(b) 分组密码

图 4.1 流密码和分组密码

4.1.2 Feistel 密码结构的设计动机

分组密码作用在 n 位明文分组上，而产生 n 位密文分组。共有 2^n 个不同的明文分组，且由于加密是可逆的（即可以解密），每一个明文分组将唯一对应一个密文分组。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。

在后面这个变换里，密文 01 可由两个明文分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[①]。

① 原因如下：对于第一个明文，我们可以选择 2^n 个密文分组中的任意一个。对于第二个明文，可以从剩下的 2^n-1 个里选择一个，以此类推。

可逆映射	
明文	密文
00	11
01	10
10	00
11	01

不可逆映射	
明文	密文
00	11
01	10
10	01
11	01

图 4.2 给出了 $n=4$ 时的一个普通代替密码的结构。4 位的输入有 16 种可能的输入状态，每一种被代替密码映射成 16 种可能输出状态中的唯一一个，每一个表示 4 位的密文输出。加密和解密映射可如表 4.1 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。Feistel 称这种密码为理想分组密码，因为它允许生成最大数量的加密映射来映射明文分组。

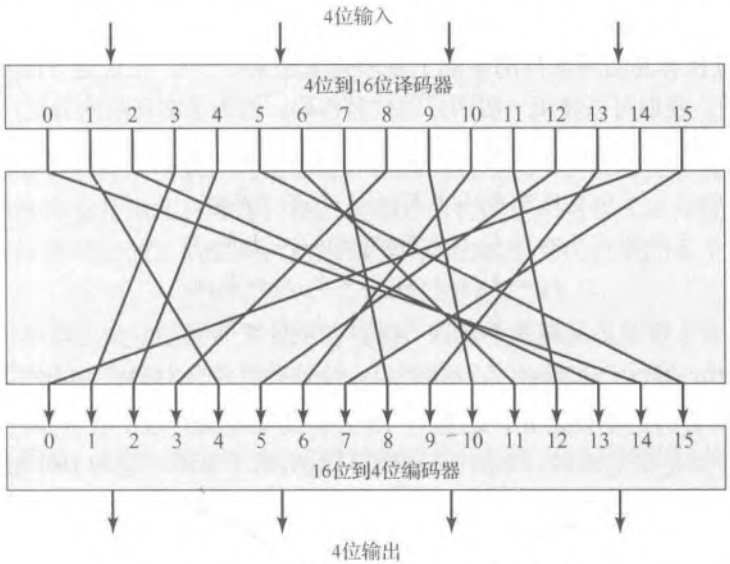


图 4.2　$n=4$ 时的一个 n 位到–n 位分组密码

表 4.1　图 4.2 所示代替密码的加/解表

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

密　文	明　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

但是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替密码。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替密码而是因为使用的分组规模太小。如果 n 充分大，并且允许明密文之间采用任意的可逆变换，那么明文的统计特征将被掩盖，从而不能用统计方法来攻击这种体制。

然而，从实现和运行的角度来看，采用大规模分组的任意可逆代替密码（即理想分组密码）是不可行的。因为对于这样的变换，映射本身就是密钥。再看一下表 4.1，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第二列来定义，它给出了每个明文对应的密文。本质上它就是决定所有可能映射中某一个映射的密钥。在这种情况下，运用这种直接的方法定义密钥，密钥长度为（4 位）×（16 行）=64 位。一般地，对于 n 位的代替分组密码，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，其密钥大小将要 $64 \times 2^{64} = 2^{70} \approx 10^{21}$ 位。

考虑到这些困难，Feistel 指出我们所需的是对理想分组密码体制（分组长度 n 较大）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。我们可以使用一般的分组代替密码，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，假设按照线性方程的集合来定义映射。在 $n=4$ 时，有

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

其中 x_i 是明文分组中的四位二进制数，y_i 是密文分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。如果算法为攻击者所知，这种公式表示的密码在密码分析攻击下将是很危险的。本例中，我们只是将第 3 章所讨论的 Hill 密码的知识运用到二进制数据而不是字符。正如第 3 章所见，这样一个简单的线性系统是较易攻破的。

4.1.3 Feistel 密码

Feistel 建议[FEIS73]使用乘积密码的概念来逼近理想分组密码。乘积密码是指依次使用两个或两个以上基本密码，所得结果的密码强度将强于所有单个密码的强度。这种方法的本质是开发一个分组密码，密钥长为 k 位，分组长为 n 位，采用 2^k 个变换，而不是理想分组密码的 $2^n!$ 个可用变换。

特别地，Feistel 建议使用这样的密码：该种密码交替地使用代替和置换。代替和置换的定义如下：

- **代替** 每个明文元素或元素组被唯一地替换为相应的密文元素或元素组。
- **置换** 明文元素的序列被替换为该序列的一个置换。也就是说，序列里没有元素被添加，删除或替换，但序列里元素出现的顺序改变了。

实际上，这是 Claude Shannon 提出的交替使用混淆和扩散的乘积密码的实际应用[SHAN49]①。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：基于 1945 年 Shannon 理论的 Feistel 密码结构，仍是当前使用的大多数重要对称分组密码的基本结构。

① 可以在本书的网站上找到该论文。这是 Shannon 于 1949 年发表的论文，它最初就是 1945 年给出的一个保密报告。Shannon 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，而且还是信息论的开创者。基于信息论的工作，他提出了数据通信信道的容量计算公式，至今仍在使用。此外，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。

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]。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，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 所指的强理想密码中，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替密码（参见图 4.2）就是这样一种密码，不过正如我们所见，它是不可能获得实际应用的①。

舍弃对理想系统的追求，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，这可以通过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_i 为字母。

$$y_n = \left(\sum_{i=1}^{k} m_{n+i} \right) \bmod 26$$

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；这来自于原始明文中不同位置的多个位对密文的某一位产生的影响②。

每一个分组密码都是明文分组到密文分组的变换，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的复杂性使得推导密钥极其困难。这可以用一些复杂的代替算法来实现，简单的线性代替函数几乎增加不了混淆。

正如文献[ROBS95b]指出，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 4.3 描述了 Feistel 提出的结构。加密算法的输入是长为 $2w$ 位的明文分组和密钥 K。明文分组被分为等长的两部分：LE_0 和 RE_0。这两半数据经过 n 轮迭代后组合成密文分组。第 i 轮迭代的输入 LE_{i-1} 和 RE_{i-1} 来自于上轮迭代的输出；而输入的子密钥 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。尽管可以使用任意轮数，但图 4.3 使用了 16 轮。

每轮迭代都有相同的结构。代替作用在数据的左半部分。它通过用轮函数 F 作用于数据的右半部分后，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。换一种说法，F 是 w 位长的右半分组以及 y 位长的子密钥的函数，输出 w 位的值：$F(RE_i, K_{i+1})$。代替之后，交换数据的左右两半完成置换③。这种结构是 Shannon 提出的代替置换网络（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。这

① 附录 F 扩展了 Shannon 关于密码算法保密和安全的度量概念。

② 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。

③ 在最后一轮迭代后有一个变换，以抵消在最后一轮迭代中的那个交换，可以从图中去掉者两个交换，但这样会使问题的表述不一致。我们将会看到，无论如何，去掉最后一轮中的交换将使解密更简单。

种安全性的增加来自更好的扩散性。传统上，64 位的分组长度比较合理，在分组密码设计里很常用。然而，高级加密标准使用的是 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，但会降低加密、解密速度。这种安全性的增加来自更好的抗穷尽攻击能力和更好的混淆性。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，密码分析就越困难。
- **轮函数** 同样，轮函数越复杂，抗攻击的能力就越强。

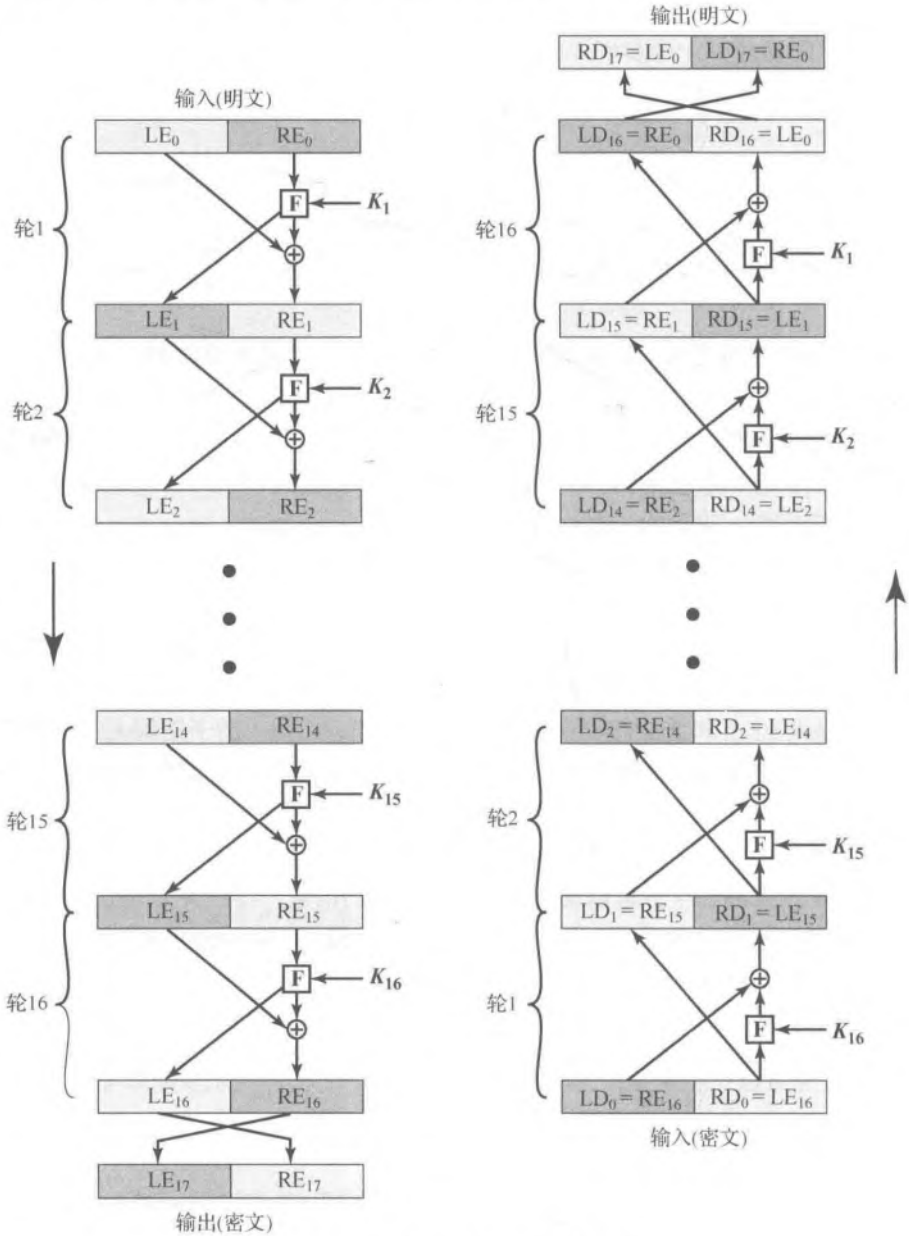


图 4.3 Feistel 加密和解密（16 轮）

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，加密算法被嵌入到应用程序中，以避免硬件实现的麻烦。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使密码分析尽可能困难，然而将算法设计得容易分析也有它的好处。也就是说，如果算法描述起来简洁清楚，那么分析其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。

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，因为我们不需要分别实现加密和解密两个算法。

图 4.3 表示了用逆序密钥和相同算法解密的过程。加密过程是在图的左边，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 $LD_{16-i}||LD_{16-i}$。

我们按照图 4.3 通过计算来证明上面说法的正确性。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

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

对于解密则有

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

XOR 运算有以下性质：

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

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 $LE_{15} \| RE_{15}$，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。我们把它表示成一般形式。对于第 i 轮加密算法

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

又可写为

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

因此我们已经描述了第 i 轮的输入是输出的函数，这些等式证实了图 4.3 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 $RE_0||LE_0$，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，比如取一种极端情况，假设它的输出为一个常数（如全为 1），等式依然成立。

为了帮助说明前面的概念，我们来看一个特别的例子（如图 4.4 所示）并且关注加密的第 15 轮，对应于解密的第 2 轮。假设每一阶段的分组长度为 32 位（分为 2 个 16 位），密钥长度为 24 位。假设第 14 轮加密后的中间分组是 DE7F03A6。那么 LE_{14}=DE7F，RE_{14}=03A6。同时假设 K_{15}=12DE52。15 轮后，我们有 LE_{15}=03A6，$RE_{15}=F$(03A6,12DE52)⊕DE7F。

现在来看解密过程。如图 4.3 所示，假设 $LD_1=RE_{15}$，$RD_1=LE_{15}$，我们想证明 $LD_2=RE_{14}$，$RD_2=LE_{14}$。所以从 LD_1= F(03A6,12DE52)⊕DE7F，RD_1 = 03A6 出发，接着从图 4.3 有 LD_2 = 03A6 = RE_{14}，RD_2= F(03A6,12DE52) ⊕[F(03A6,12DE52)⊕DE7F]=DE7F=LE_{14}。



图 4.4 Feistel 结构示例

4.2 数据加密标准

在 2001 年高级加密标准（AES）提出前，数据加密标准（DES）一直是使用最广泛的加密方案。DES 于 1977 年被美国国家标准局（National Bureau of Standards，NBS），即现在的国家标准和技术学会(National Institute of Standards and Technology，NIST)采纳为联邦信息处理标准 46(FIPS PUB 46)。这个算法本身指的是数据加密算法（Data Encryption Algorithm，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。

经过多年的发展，DES 成为主流的对称加密算法，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 颁布了标准的新版本（FIPS PUB 46-3），新版本规定了 DES 只能用于（历史）遗留系统以及 3DES 的使用。我们将会在第 7 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。有关这一内容的详细描述，有兴趣的读者可参阅附录 S。

① 术语有点混乱，直到现在还经常将 DES 和 DEA 互换使用。然而，最新版的 DES 的文件中包含了此处描述的 DEA，还有第 7 章的三重 DEA。它们都是数据加密标准的一部分。三重 DEA 算法一直被称为三重 DES，并写为 3DES，直到最近，官方术语采用了 TDEA。为了方便，我们采用 3DES。

4.2.1　DES 加密

图 4.5 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 轮相同函数的作用，每轮作用都有置换和代替。最后一轮迭代的输出有 64 位，它是输入明文和密钥的函数。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 互逆的置换 IP^{-1} 作用产生 64 位的密文。除了初始和末尾的置换，DES 的结构与图 4.3 所示的 Feistel 密码结构完全相同。

图 4.5 的右半部分给出了使用 56 位密钥的过程。首先，密钥经过一个置换后，再经过循环左移和一个置换分别得到各轮的子密钥 K_i 用于各轮的迭代。每轮的置换函数都一样，但是由于密钥的循环移位使得各轮子密钥互不相同。

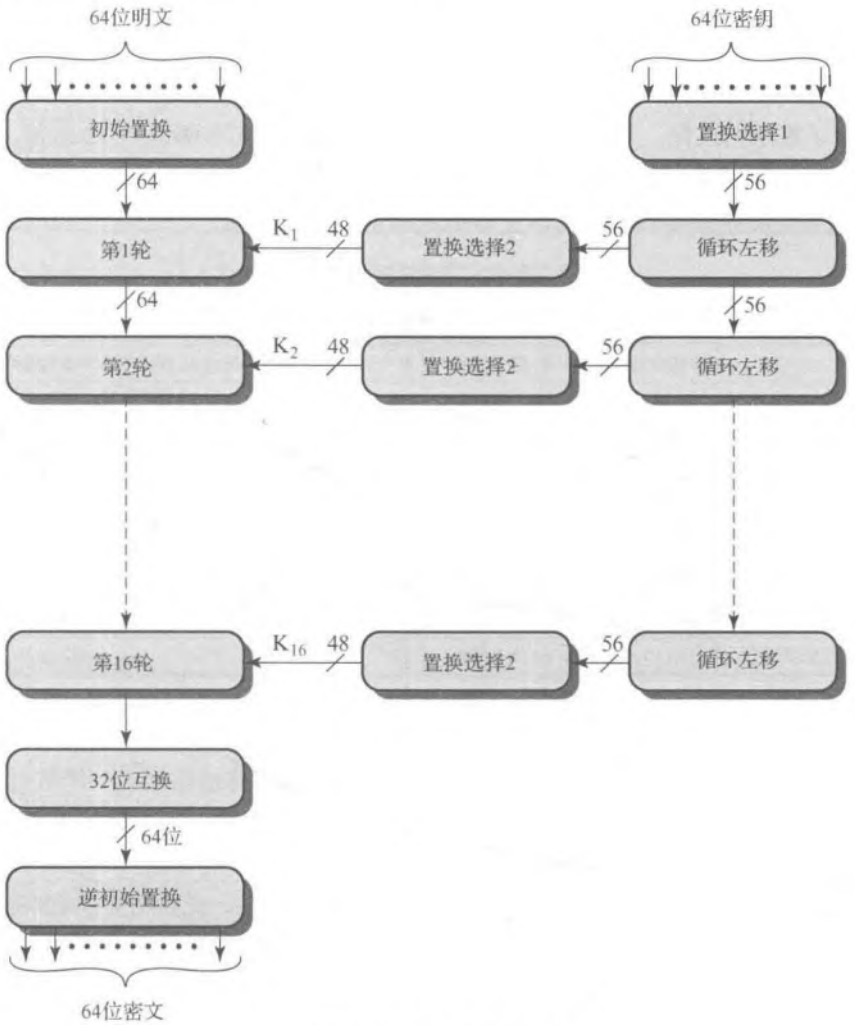


图 4.5　DES 加密算法概述

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。

4.2.2 DES 解密

Feistel 密码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。此外，初始置换和最终置换是相反的。

4.3 DES 的一个例子

我们现在跟踪一个例子并考虑它的含义。尽管不要求你去手工重复该例子，但是你将发现研究一下例子中十六进制模式的数据从一步到下一步的的变化是非常有用的。

本例中，明文是一个十六进制的回文，明文、密钥和所得密文如下：

明文	02468aceeca86420
密钥	0f1571c947d9e859
密文	da02ce3a89ecac3b

4.3.1 结果

表 4.2 显示了算法的过程。第一行是经过初始置换后数据的左右两边的 32 位值。下面的 16 行显示的是各轮运行后的结果。表中也显示了各轮的 48 位子密钥。注意 $L_i = R_{i-1}$。最后一行显示的是经过逆初始置换后左边和右边的值。这两个值组合起来形成密文。

表 4.2 DES 的例子

轮数	K_i	L_i	R_i	轮数	K_i	L_i	R_i
IP		5a005a00	3cf03c0f	9	04292a380c341f03	c11bfc09	887fbc6c
1	1e030f03080d2930	3cf03c0f	bad22845	10	2703212607280403	887fbc6c	600f7e8b
2	0a31293432242318	bad22845	99e9b723	11	2826390c31261504	600f7e8b	f596506e
3	23072318201d0c1d	99e9b723	0bae3b9e	12	12071c241a0a0f08	f596506e	738538b8
4	05261d3824311a20	0bae3b9e	42415649	13	300935393c0d100b	738538b8	c6a62c4e
5	3325340136002c25	42415649	18b3fa41	14	311e09231321182a	c6a62c4e	56b0bd75
6	123a2d0d04262a1c	18b3fa41	9616fe23	15	283d3e0227072528	56b0bd75	75e8fd8f
7	021f120b1c130611	9616fe23	67117cf2	16	2921080b13143025	75e8fd8f	25896490
8	1c10372a2832002b	67117cf2	c11bfc09	IP^{-1}		da02ce3a	89ecac3b

4.3.2 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需要的一个好性质。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，这被称为雪崩效应。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。

运用表 4.2 的例子，表 4.3 显示了当明文的第 4 位改变时的结果，此时明文是 12468aceeca86420。表的第二列显示的这两个明文在每轮结束后得到的 64 位值。第三列显示的是这两个值所差的位数量。该表表明，仅仅经过 3 轮，两组数据差别达到 18 位，当 16 轮全部结束时，两组密文差 32 位。

表 4.4 显示的是一个类似的测试，仍然使用原明文，使用的两个密钥仅在第 4 位有差异：初始密钥为 0f1571c947d9e859，改变后的密钥为 1f1571c947d9e859。同样，结果表明密文里大约有一半的位发生改变，且仅仅经过几轮的变换就出现了雪崩效应。

表 4.3　DES 的雪崩效应：改变明文

轮数		δ	轮数		δ
	02468aceeca86420 12468aceeca86420	1	9	c11bfc09887fbc6c 99f911532eed7d94	32
1	3cf03c0fbad22845 3cf03c0fbad32845	1	10	887fbc6c600f7e8b 2eed7d94d0f23094	34
2	bad2284599e9b723 bad3284539a9b7a3	5	11	600f7e8bf596506e d0f23094455da9c4	37
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18	12	f596506e738538b8 455da9c47f6e3cf3	31
4	0bae3b9e42415649 171cb8b3ccaca55e	34	13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
5	4241564918b3fa41 ccaca55ed16c3653	37	14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
6	18b3fa419616fe23 d16c3653cf402c68	33	15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
7	9616fe2367117cf2 cf402c682b2cefbc	32	16	75e8fd8f25896490 1ce2e6dc365e5f59	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33	IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

表 4.4　DES 的雪崩效应：改变密钥

轮数		δ	轮数		δ
	02468aceeca86420 02468aceeca86420	0	9	c11bfc09887fbc6c 548f1de471f64dfd	34
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3	10	887fbc6c600f7e8b 71f64dfd4279876c	36
2	bad2284599e9b723 9ad628c59939136b	11	11	600f7e8bf596506e 4279876c399fdc0d	32
3	99e9b7230bae3b9e 9939136b768067b7	25	12	f596506e738538b8 399fdc0d6d208dbb	28
4	0bae3b9e42415649 768067b75a8807c5	29	13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
5	4241564918b3fa41 5a8807c5488dbe94	26	14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
6	18b3fa419616fe23 488dbe94aba7fe53	26	15	56b0bd7575e8fd8f d2c3a56f2765c1fb	27
7	9616fe2367117cf2 aba7fe53177d21e4	27	16	75e8fd8f25896490 2765c1fb01263dc4	30
8	67117cf2c11bfc09 177d21e4548f1de4	32	IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

4.4　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长度和算法本身的性质。

4.4.1 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，要搜索一半的密钥空间。一台每毫秒执行一次 DES 加密的计算机需要用 1000 年才能破译出密文。

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 ms 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。

有了现在的技术，甚至没有必要使用特殊的、专用的硬件。相反地，商业化，现成的处理器的速度威胁着这个市场 DES 的安全。希捷科技最近的一篇文章[SEAG08]表明，对于今天的多核系统，每秒 10 亿（10^9）个按键组合的比率是合理的计算机。最近的产品证实了这一点。英特尔和 AMD 现在都提供基于硬件的指令来加速 AES 的使用。测试运行在现代多核英特尔机器的加密速度约为每秒 5 亿的加密率[BASU12]。另一个最近的分析表明，当代超级计算机技术为每秒 10^{13} 的加密率更为合理[AROR12]。

考虑到这些结果，表 4.5 显示了需要多少时间对各种密钥大小的蛮力攻击。可以看出，一台 PC 可以破坏 DES 的时间大约为一年，如果多台 PC 并行工作，时间将大大缩短。今天的超级计算机应该可以在一个小时内找到密钥。增大密钥大小是防止使用简单蛮力攻击的有效方法，128 位或更大的位是有效的。即使设法加快了 1 万亿的攻击系统（1012），使用 128 位密钥破解代码仍然需要 10 万年以上的时间才能攻破。

幸运的是有大量 DES 的替代算法，最重要的有 AES 和 3DES，我们将在第 6 章和第 7 章分别加以讨论。

4.4.2 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 个代替表，即 S 盒身上（参见附录 S）。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，进而攻击 DES。这种说法很诱人，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，仍然没有人发现 S 盒存在致命的弱点①。

表 4.5 穷尽密钥搜索所需的平均时间

密钥大小(位)	密码	密钥个数	每微秒执行一次解密所需时间	每微秒执行 1000 万次解密所需的时间
56	DES	$2^{56}=7.2\times10^{16}$	$2^{55}\mu s=1.125$ 年	1 小时
128	AES	$2^{128}=3.4\times10^{38}$	$2^{127}\mu s=5.3\times10^{21}$ 年	5.3×10^{17} 年
168	Triple DES	$2^{168}=3.7\times10^{50}$	$2^{167}\mu s=5.8\times10^{33}$ 年	5.8×10^{29} 年
192	AES	$2^{192}=6.3\times10^{57}$	$2^{191}\mu s=9.8\times10^{40}$ 年	9.8×10^{36} 年
256	AES	$2^{256}=1.2\times10^{77}$	$2^{255}\mu s=1.8\times10^{60}$ 年	1.8×10^{56} 年
26 个字符的排列组合	Monoalphabetic	$26!=4\times10^{26}$	$2\times10^{26}\mu s=6.3\times10^{9}$ 年	6.3×10^{6} 年

4.4.3 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有些关系。本质上计时攻击是通过观察算法的一个既定实现对多种密文解密的所需时间，来获得关于密钥或明

① 至少没有人公开宣布这样的发现。

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]给出了利用计时攻击获得密钥汉明权重（二进制串中 1 的位数）的一种方法。这离知道实际密钥还很遥远，但这的确是很让人感兴趣的一步。作者给出的结论是 DES 似乎能够很好地抵抗计时攻击，但是也给出了一些可能的研究建议。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，更不能攻击更强的对称密码，如 3DES 和 AES。

4.5　分组密码的设计原理

尽管在设计具有强密码学意义的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很大的改变。在本节我们重点讨论分组密码设计的三个主要问题：迭代轮数、函数 F 的设计和密钥的使用方案。

4.5.1　迭代轮数

Feistel 密码的强度来自于三个方面：迭代轮数、函数 F 和密钥使用算法。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，迭代轮数的选择标准是使密码分析的难度大于简单穷举攻击的难度。这个标准当然也用于了 DES 的设计中。Schneier[SCHN96]观察到，对 16 轮迭代的 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，而穷举攻击平均则需要 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效率就要高一些。

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

4.5.2　函数 F 的设计

Feistel 密码的核心是函数 F。函数 F 给 Feistel 密码注入了混淆的成分。因此难以破解由 F 实现的这个代替密码函数的作用。F 的明显标准是非线性，这一点在前面我们已经讨论过。F 的非线性成分越多，任何形式的分析就会越困难。非线性有多种度量方法，但这个话题超出本书的主题，在此不多赘述。粗略说来，越难将 F 近似表示为某些线性等式，F 的非线性度就越高。

设计 F 时还应考虑其他几个标准。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 SAC（Strict Avalanche Criterion），参见文献[WEBS86]，即对于所有的 i 和 j，它要求若 S 盒的输入的任意一位 i 发生变化，输出的任意一位 j 发生变化的可能性为 1/2（参见附录 S 中关于 S 盒的讨论）。尽管 SAC 是对 S 盒而言的，作为一个标准它同样适用于整个 F 函数。当 F 中不含 S 盒时，这条准则是很重要的。

文献[WEBS86]中建议的另一项标准是 BIC 标准（Bit Independent Criterion），即对任意的 i，j，k，当输入中的一位 i 发生变换时，输出中的位 j 和位 k 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。

4.5.3　密钥扩展算法

对于任何 Feistel 分组密码，密钥被用来为每轮迭代产生一个子密钥。一般说来，子密钥的选择

① 前面讲过，对 DES 进行密码分析需要 2^{47} 个选择明文。如果仅有已知明文，就需要大量的已知明密文对，并从中找到一个有用的。这样就使工作量上升到 $2^{55.1}$。

应该加大推导子密钥及密钥种子的难度。目前还没有这方面的一般原理见诸报道。

Adams 建议[ADAM94]，密钥扩展算法至少应保证密钥和密文符合严格雪崩标准和位独立标准。

4.6 关键术语、思考题和习题

关键术语

雪崩效应	分组密码	混淆	数据加密标准（DES）
扩散	Feistel 密码	不可逆映射	密钥
置换	乘积密码	轮	轮函数
子密钥	代替	可逆映射	

思考题

4.1 为什么说研究 Feistel 密码很重要？

4.2 分组密码和流密码的差别是什么？

4.3 为什么使用表 4.1 所示的任意可逆代替密码不实际？

4.4 什么是乘积密码？

4.5 混淆与扩散的差别是什么？

4.6 哪些参数与设计选择决定了实际的 Feistel 密码算法？

4.7 解释什么是雪崩效应。

习题

4.1 （a）在 4.1 节名为“Feistel 密码结构的设计动机”的小节里说到，对于 n 个位的分组长度，理想分组密码的不同可逆映射个数为 $2^n!$ 个。请证明。

（b）在同样的小节里说到，对于理想分组密码，若允许所有可逆映射，则密钥的长度为 $n\times 2^n$ 位。但是，如果有 $2^n!$ 个映射，则应该需要 $\log_2 2^n!$ 个位就可以区分这 $2^n!$ 个映射。所以密钥长度应为 $\log_2 2^n!$。然而，$\log_2 2^n! < n\times 2^n$。请解释这种差别。

4.2 考虑分组长度为 128 位，密钥长度为 128 位的 16 轮 Feistel 密码。假设对于给定的 k，头 8 个轮密钥 $k_1, k_2, \cdots, k_8$ 由密钥扩展算法决定，而设定 $k_9=k_8$，$k_{10}=k_7$，$\cdots$，$k_{16}=k_1$。假设有密文 c，请解释，如何只向加密 oracle 做一次提问，而解密 c 获得明文 m？这表明上述的密码易于被选择明文攻击（可以认为加密 oracle 就是一种装置，给定一个明文，返回相应的密文。装置的内部结构是未知的。当然你也不能打开装置来查看，你所能做的就是对其进行提问而观察相应的输出）。

4.3 设 π 表示整数 $0,1,\cdots,2^n-1$ 的一个置换。$\pi(m)$表示 m 的置换值，其中 $0\leqslant m<2^n$。换句话说，π 将 n 位的整数集合映射到其自身，并且没有两个整数映射到相同的整数。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。证明如下有点意外的结论：多于 60%的映射将至少有一个不动点。

4.4 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)，这里假设密钥 K 选择了 $N!$ 种映射中的一种。如果希望用穷举的方法找出 K。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。若成立就有理由说 $K = K'$。然而也有这样的情况，恰好 E(K, ·)和 E(K', ·)对这 t 个明密文对的结果一样，但其他明密文对不相同。

（a）$E(K,\cdot)$和 $E(K',\cdot)$确实表示不同映射的概率有多大？

（b）$E(K,\cdot)$和 $E(K',\cdot)$对另外 $t'(1\leqslant t'\leqslant N-t)$个明密文对正好都成立的概率有多大？

4.5　对于任意的分组密码，它的非线性对安全是至关重要的。为了证明这一点，假设有一个线性分组密码 EL，加密 128 位的明文分组为 128 位的密文分组。令 $EL(k, m)$是 128 位明文 m 在密钥为 k 时的加密结果。则对任意的 128 位的 m_1，m_2，有

$$EL(k, [m_1 \oplus m_2]) = EL(k, m_1) \oplus EL(k, m_2)$$

请说明给定 128 个选择密文，在不知密钥 k 的情况下，对手如何解密任何密文？(选择密文即对手可以选择密文，并能得到解密结果。此处，你有 128 个明密文对，且你可以选择密文的值)。

4.6　假设无论输入密钥 k 为何值，DES 的 F 函数总是将每一个 32 位的输入 R 映射为

（a）32 位的全 1 串；（b）R 的按位补。

问：（1）DES 算法起什么作用？

（2）解密看上去如何？

提示：运用异或运算的如下性质：

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

$$A \oplus A = \mathbf{0}$$

$$A \oplus 0 = A$$

$$A \oplus \mathbf{1} = A \text{ 的按位取反}$$

其中 A，B，C 是长为 n 位的字符串，$\mathbf{0}$ 是长为 n 位的全 0 串，$\mathbf{1}$ 是长为 n 位的全 1 串。

4.7　证明 DES 解密算法实际上是 DES 加密算法的逆。

4.8　为了是加密算法可逆，16 轮后 DES 算法需要交换 32 位。这样，仅仅将密文沿着原算法逆回去并将密钥逆序即可解密。习题 4.7 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）表示组合 $TD_1(IP(IP^{-1}(T_{17}(T_{16}(L_{15}\|R_{15})))))$和交换 32 位 L_{15} 和 R_{15} 的转换等价。也就是表示

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

（b）假设我们去掉了加密算法最后的 32 位的交换，请判断下式是否成立：

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

注意：下面的问题涉及 DES 的细节请参考附录 S。

4.9　考虑由表 S.2 中 S 盒 S_1 的第一行定义的代替。并给出对应这个代替，类似于图 4.2 的分组图。

4.10　假设密文和密钥的各位全为 1，计算 DES 解密时第一轮输出中的 1 位，16 位，33 位，48 位。

4.11　这个问题给出了用一轮 DES 加密的具体数值的例子。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 1100 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 得到 E[R_0]，其中 E[·]是表 4.2 的扩展函数。

（d）计算 A= E[R_0] ⊕ K_1。

（e）把（d）的 48 位结果分成 6 位（数据）一组的集合并求对应 S 盒代替的值。

（f）将（e）的结果连接起来获得一个 32 位的结果 B。

（g）应用置换获得 $P(B)$。

（h）计算 R_1=$P(B)$ ⊕ L_0。

（i）写出密文。

4.12 比较初始置换表［参见表 S.1(a)］和置换选择表一［参见表 S.3(b)］，它们的结构是否相似？如果是，请说明它们的相似之处。通过分析可以得出什么结论？

4.13 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 S.1 的右半部分对解密而言不再正确。请模仿表 S.3(d)为解密过程设计一个合适的密钥移位扩展方案。

4.14 （a）设 X'是对 X 按位取反的结果。证明对于 DES 加密算法，如果明文和密钥都取反，则密文也取反。即

$$\text{如果 } Y = \mathrm{E}(K, X)$$
$$\text{那么 } Y' = \mathrm{E}(K', X')$$

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？

4.15 证明 DES 中每个子密钥的前 24 位均来自于初始密钥的同一个子集，该子集有 28 位，而后 24 位来自于初始秘密钥的另外 28 位。

注意：下面的问题请参考附录 G 的简化 DES。

4.16 图 G.2 描述了 S-DES 密钥生成，参考该图回答：

（a）初始 P10 置换函数有什么重要性？

（b）两个 LS-1 移位函数有什么重要作用？

4.17 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。

4.18 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。把明文串的前 4 位译码为某个字母，后 4 位译码为另一个字母（这里，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)。

编程习题

4.19 编一个软件能够实现一般代替分组密码的加解密。

4.20 编一个软件能够实现 S-DES 分组密码的加解密。测试数据：请使用习题 4.18 里的明文、密文和密钥。

第5章 有 限 域

学 习 目 标

通过本章的学习应具备以下能力：

- 能够区分群、环和域
- 理解有限域 GF(p)的定义
- 能够解释普通多项式运算、系数在 Z_p 中的多项式运算和 GF(2^n)的多项式运算的差异
- 理解有限域 GF(2^n)的定义
- 能够解释模运算操作符的两种不同用法

有限域在密码学中的地位越来越重要。许多密码算法都依赖于有限域的性质，特别是高级加密标准（AES）和椭圆曲线加密算法。其他例子包括消息认证码 CMAC 以及认证加密方案 GMC。

本章为读者提供了足够有关有限域概念的背景知识，从而能够理解 AES 以及其他使用有限域知识的密码算法的设计。由于对抽象代数不熟悉的学生可能难以理解有限域的相关概念，我们以一种易于理解的方式进入这个主题。我们的计划如下所述。

（1）域是更大的一类代数结构——环的子集，而环又是更大的一类代数结构——群的子集。事实上，正如图 5.1 所示，群和环都有很大的不同。群被定义为拥有一些简单性质的集合，而且易于理解。而接下来的子集（交换群，环，交换环，等等）都添加了一些额外的性质，变得越来越复杂。我们将在 5.1 节到 5.3 节依次介绍群、环、域。

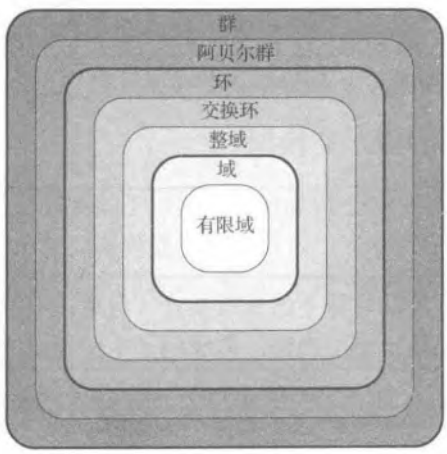


图 5.1 群、环、域

（2）有限域是只有有限个元素的域。这类域可以在一些密码算法中找到。有了域的相关概念，我们在 5.4 节转向一类特殊的有限域：这类域只有 p 个元素，p 是一个素数。一些非对称密码算法中用到了这类域。

（3）在密码学中，有一类更重要的有限域，它包含 2^n 个元素，记为 GF(2^n)。它被广泛地应用于各类密码算法中。在讨论这类域之前，需要先进入多项式运算这个主题，我们将在 5.5 节中讨论这些内容。

（4）当所有的这些准备工作做好之后，就可以在 5.6 节中讨论有限域 GF(2^n)。

在继续阅读之前，读者可以先复习一下在 2.1 节到 2.3 节中的数论相关知识。

5.1 群

群、环和域都是数学理论中的一个分支——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，可以通过很多种方法，使集合上的两个元素运算后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，必须指出，在抽象代数中，并不仅限于普通的算术操作。这一点通过后面章节的介绍很容易看明白。

5.1.1 群

群 G，有时记为$\{G,\cdot\}$，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，都有 $a\cdot(b\cdot c)=(a\cdot b)\cdot c$ 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'，使得下式成立 $a\cdot a'=a'\cdot a=e$。

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[②]。定义 S_n 为 n 个不同符号所形成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 π。可以很容易地验证 S_n 是一个群：

A1 如果 π，$\rho\in S_n$，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，如$\{3, 2, 1\}\cdot\{1, 3, 2\}=\{2, 3, 1\}$。显然，$\pi\cdot\rho\in S_n$。

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，单位元就是$\{1, 2, \cdots, n\}$。

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

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

5.1.2 交换群

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。

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。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$-a$；并且将减法用以下规则定义：$a-b=a+(-b)$。

5.1.3 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，定义 $a^0=\mathrm{e}$ 作为单位元；并且 $a^{-n}=(a')^n$，其中 a'是 a 在群内的逆元素。如果群中的每一个元素都是一个固定元素 $a(a\in G)$

① 操作符·具有一般性；可以指加法，乘法，或某些其他的数学运算。

② 这和第 2 章中置换的定义是等价的，那里说有限个元素的集合 S 的置换是 S 中元素的有序序列，其中每个元素只出现一次。

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。

5.2 环

环 R，有时记为$\{R, +, \times\}$，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，下面两个式子总成立：

$$a(b+c)=ab+ac$$
$$(a+b)c=ac+bc$$

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。

实数上所有 n 阶方阵的集合关于加法和乘法构成一个环。

环如果还满足以下条件，则被称为是交换环：

（M4）乘法的交换律 对于 R 中任意元素 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是交换环。前面一个例子中定义的所有 n 阶方阵的集合就不是交换环。

整数$\{0, 1, \cdots, n-1\}$的集合 Z_n 加上模 n 的算术运算构成一个交换环（参见表 4.3）。

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。

5.3 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。

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乘法。

（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 $a\,a^{-1} = (a^{-1})\,a = 1$ 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。

进一步理解之后，下面这些可替代的描述可能是有用的。域{F, +}是一个集合，定义了两个二元运算——加法和乘法，满足下面的条件：

（1）F 对加法组成一个交换群。

（2）除去零，F 对乘法组成一个交换群。

（3）分配律成立。也就是说，对于 F 中的任意元素 a、b、c，有

$$a(b+c)=ab+ac$$
$$(a+b)c=ac+bc$$

（4）图 5.2 总结了定义群、环和域的公理。



图 5.2 群、环和域的性质

5.4 有限域 GF(p)

在 5.3 节中，我们把域定义为一个满足图 5.2 中所有公理的集合，并给出了一些无限域的例子。无限域在密码学中没有特别的意义。然而，正如图 5.3 展示的那样，两种类型的有限域在许多密码算法中扮演着重要的角色。

可以看出，有限域的阶(元素的个数)必须是一个素数的幂 p^n，n 为正整数。阶为 p^n 的有限域一般记为 GF(p^n)，GF 代表 Galois 域，以第一位研究有限域的数学家的名字命名。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，将在本节对它进行研究。对于有限域 GF(p^n)来说，GF(2^n)具有特别重要的密码学意义。将在 5.6 节研究 GF(2^n)有限域。

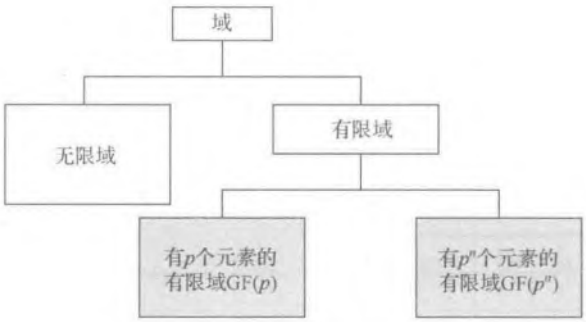


图 5.3 域的类型

5.4.1 阶为 p 的有限域

给定一个素数 p，元素个数为 p 的有限域 GF(p)被定义为整数$\{0,1,\cdots,p-1\}$的集合 Z_p，其运算为模 p 的算术运算。

曾在 5.2 节讲到，整数$\{0,1,\cdots,n-1\}$的集合 Z_n，在模 n 的算术运算下，构成一个交换环（参见表 2.5）。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［参见式（2.5）的讨论］[①]。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，可以在表 5.2 中列出的 Z_p 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，如果用 w 乘以 Z_p 的所有元素，得出的剩余类是 Z_p 中所有元素的另一种排列。因此，恰好只有一个剩余类值为 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，记为 w^{-1}，所以 Z_p 其实是一个有限域。而且，式（2.5）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如下：

$$\text{如果 } (a\times b)\equiv(a\times c)\ (\bmod p)\ ,\ \text{则 } b\equiv c\ (\bmod p)。 \tag{5.1}$$

将式（5.1）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)。它的算术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（XOR）运算，乘等价于逻辑与（AND）运算。

① 讨论式（2.5）时说过，如果两个整数的唯一正整数公因子是 1，则称其互素。

表 5.1 描述了 GF(7)的算术运算。这是一个阶为 7 的域，采用模 7 运算。将会看到它满足图 5.2 里所示域的所有性质，与表 5.1 比较，表 2.2 里集合 Z_8 用模 8 算术运算，不是域。本章的后面，我们将展示在 Z_8 上定义合适的加法和乘法使之成为一个域。

5.4.2 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。你只需构造一个乘法表，如表 5.1(e)，所要的结果可以从中直接读出。但是，当 p 值比较大时，这种方法是不切实际的。

表 5.1 模 7 和模 8 的算术运算

+	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 的加法和乘法逆

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

(c) 模 8 乘法

×	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

(d) 模 7 乘法

×	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

(e) 模 7 加法

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

(f) 模 7 的加法和乘法逆

如果 a 和 b 互素，则 b 有模 a 的乘法逆元。也就是说，如果 $\gcd(a,b) = 1$，那么 b 有模 a 的乘法逆元。即对于正整数 $b < a$，存在 $b^{-1} < a$ 使 $bb^{-1} = 1 \bmod a$。如果 a 是素数并且 $b < a$，则显然 a 和 b 互素，且其最大公因子为 1。运用扩展的 Euclid 算法容易计算 b^{-1}。

再次重复式（2.7），我们已经证明过该式可以用扩展 Euclid 算法来求解。

$$ax+by = d = \gcd(a,b)$$

现在，如果 $\gcd(a,b) = 1$，则有 $ax+by = 1$。运用 2.3 节定义的模算术的基本等式，有

$$[(ax \bmod a)+(by \bmod a)] \bmod a = 1 \bmod a$$
$$0+(by \bmod a) = 1$$

然而，如果 $by \bmod a = 1$，则 $y = b^{-1}$。因此，如果 $\gcd(a,b) = 1$，则通过对式（2.7）运用扩展 Euclid 算法可以获得 b 的乘法逆元。

考虑表 2.4 中的例子。此时有 a=1759，a 是一个素数，b=550。从方程 $1759x+550y=d$ 的解得到 y=355。因此 b^{-1}=355。容易验证，$550\times355 \bmod 1759 = 195250 \bmod 1759 = 1$。

更一般地，对于任意的 n，扩展 Euclid 算法可以用于求取 Z_n 内的乘法逆元。如果运用扩展 Euclid 算法于方程 $nx+by=d$，并且得到 d=1，则在 Z_n 内有 $y=b^{-1}$。

5.4.3 小结

在本节里我们展示了如何造阶为 p 的有限域，其中 p 为素数。特别地，我们用以下性质定义有限域 GF(p)：

（1）GF(p)由 p 个元素构成。

（2）在该集合中定义了二元运算+和×。加、减、乘、除可以在集合内实现。除 0 以外，集合内的其他元素都有乘法逆。

我们已经表明 GF(p)的元素是集合{0，1，…，$p-1$}中的元素，其算术运算是模 p 的加法和乘法。

5.5 多项式运算

在继续讨论有限域之前，需要介绍一个有趣的问题——多项式算术。我们只讨论单变元多项式，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 GF(p)中。
- 系数在 GF(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。

5.5.1 普通多项式运算

一个 n 次多项式（$n\geq0$）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，该数集称为系数集，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，仅仅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，一般不给多项式中的 x 赋一个特定值［如 $f(7)$］。为了强调这一点，变元 x 有时被称为不定元。

多项式运算包含加法、减法和乘法，这些运算是把变量 x 当成集合 S 中一个元素来定义的。除法也以类似方式定义，但这时要求 S 是域。这些域包括实数域、有理数域和素数域 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

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

那么加法运算定义为

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

乘法运算定义为

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

其中，$c_k=a_0b_k+a_1b_{k-1}+\cdots+a_{k-1}b_0+a_kb_0$。

在最后一个公式中，当 $i>n$ 时，令 $a_i=0$，当 $i>m$ 时，令 $b_i=0$。注意结果的次数等于两多项式次数之和。

例如，令 $f(x)=x^3+x^2+2$，$g(x)=x^2-x+1$，其中 S 是整数集，则

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

手工计算过程参见图 5.5(a)～5.5(c)。除法将在后面讨论。

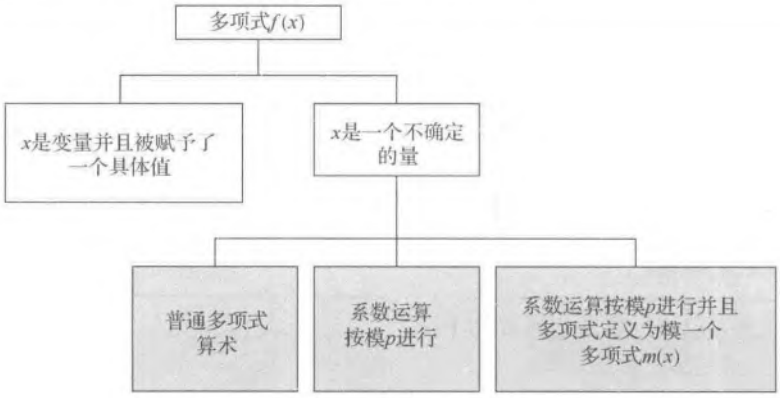


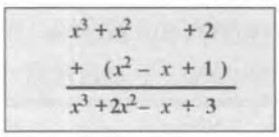
图 5.4 多项式概论

5.5.2 系数在 Z_p 中的多项式运算

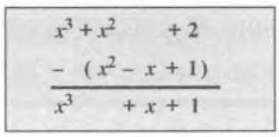
现在我们考虑这样的多项式，它的系数是域 F 的元素。称其为域 F 上的多项式。这种情况下，容易看出这样的多项式集合是一个环，称为多项式环。也就是说，如果把每个不同的多项式看成集合中的元素，这个集合是一个环[①]。

对域上的多项式进行多项式运算时，除法运算是可能的。注意这并不是说能进行整除。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。

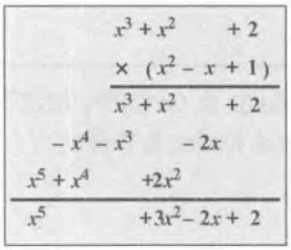
① 事实上，系数在交换环中的多项式集合构成多项式环，但是这个结论在这里并无多大的意义。



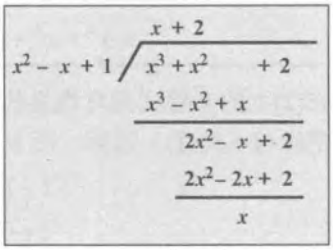
(a) 加法



(b) 减法



(c) 乘法



(d) 除法

图 5.5 多项式运算的例子

考虑集合 S 中的除法运算 5/3。如果 S 是有理数集合（是域），那么结果可以简单地表示成 5/3，这是 S 中的一个元素。现在假设 S 是域 Z_7，在这种情况下，用表 5.1(f)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。对于域上的多项式，式（2.1）的除法可以重述如下：给定 n 次多项式 $f(x)$ 和 m 次多项式 $g(x)$，（$m\leqslant n$），如果用 $g(x)$除 $f(x)$，可得到一个商 $q(x)$和一个余数 $r(x)$，满足如下关系式：

$$f(x) = q(x)g(x)+r(x) \tag{5.2}$$

各多项式的次数为

Degree $f(x)= n$

Degree $g(x)= m$

Degree $q(x)=n-m$

Degree $r(x)\leqslant m-1$

如果允许余数，我们说域上多项式除法是可以的。与整数的长除法相同，长除法也可应用到多项式除法上。后面会有一些例子。

与整数运算相似，可以在式（5.2）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 $r(x)=0$]，那么可以说 $g(x)$整除 $f(x)$，写为 $g(x)|f(x)$；等价地，可以说 $g(x)$是 $f(x)$的一个因式或除式。

对于上个例子[$f(x)=x^3+x^2+2$ 且 $g(x)=x^3-x+1$]，$f(x)/g(x)$产生一个商式 $q(x)=x+2$ 和一个余式 $r(x)=x$，如图 5.5(d)所示。这很容易证实，只要注意到

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

对我们而言，GF(2)上的多项式最有意义的。5.4 节中提过，在 GF(2)中，加法等价于异或（XOR）运算，乘法等价于逻辑与（AND）运算。而且，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 5.6 给出了 GF(2)上多项式运算的一个例子。对于 $f(x)=(x^7+x^5+x^4+x^3+x+1)$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，$g(x)|f(x)$

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

(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

$$\begin{array}{r} x^4 + 1 \\ x^3 + x + 1 \,\big/\, x^7 + x^5 + x^4 + x^3 + x + 1 \\ x^7 + x^5 + x^4 \\ \hline x^3 + x + 1 \\ x^3 + x + 1 \\ \hline \end{array}$$

(d) 除法

图 5.6 GF(2)上的多项式运算的例子

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被称为**素多项式**。

GF(2)上的多项式 $f(x)=x^4+1$ 是可约的，因为

$$x^4+1=(x+1)(x^3+x^2+x+1)$$

考虑多项式 $f(x)=x^3+x+1$，容易观察到 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

$$\begin{array}{r} x^2+x \\ x+1\overline{)\,x^3\qquad +x+1} \\ \underline{x^3+x^2} \\ x^2+x \\ \underline{x^2+x} \\ 1 \end{array}$$

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。

5.5.3 求最大公因式

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相似性。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。

可以改写 Euclid 算法来计算两个多项式的最大公因式。回忆第 2 章中的式(2.6)中的等式：$\gcd(a, b)=\gcd(a, a \bmod b)$，它是 Euclid 算法的基本等式。这个等式可以改写为

$$\gcd[a(x), b(x)]=\gcd[b(x), a(x) \bmod b(x)] \tag{5.3}$$

式（5.3）可以重复使用来决定最大公因子。将下列的方案和用于整数的 Euclid 算法的定义进行比较。

多项式 Euclid 算法	
计算	满足
$r_1(x)=a(x) \bmod b(x)$	$a(x)=q_1(x)b(x)+r_1(x)$
$r_2(x)=b(x) \bmod r_1(x)$	$b(x)=q_2(x)r_1(x)+r_2(x)$
$r_3(x)=r_1(x) \bmod r_2(x)$	$r_1(x)=q_3(x)r_2(x)+r_3(x)$
⋮	⋮
$r_n(x)=r_{n-2}(x) \bmod r_{n-1}(x)$	$r_{n-2}(x)=q_n(x)r_{n-1}(x)+r_n(x)$
$r_{n+1}(x)=r_{n-1}(x) \bmod r_n(x)=0$	$r_{n-1}(x)=q_{n+1}(x)r_n(x)+0$ $d(x)=\gcd(a(x), b(x))=r_n(x)$

每一循环里我们都有 $d(x)=\gcd(r_{i+1}(x), r_i(x))$，直至最后 $d(x)=\gcd(r_n(x), 0)=r_n(x)$。因此，通过重复应用除法我们得到了两个多项式[①]的最大公因子。这就是用于多项式的 Euclid 算法。算法假设 $a(x)$的次数大于 $b(x)$的次数。

求 $\gcd[a(x), b(x)]$，其中 $a(x)=x^6+x^5+x^4+x^3+x^2+x+1$，$b(x)=x^4+x^2+x+1$。
首先用 $b(x)$除 $a(x)$

① 此处原文为整数——编者注。

$$
\begin{array}{r}
x^2+x \\
x^4+x^2+x+1\overline{)\,x^6+x^5+x^4+x^3+x^2+x+1} \\
\underline{x^6\quad\;+x^4+x^3+x^2} \\
x^5\qquad\qquad+x+1 \\
\underline{x^5\quad\;+x^3+x^2+x} \\
x^3+x^2\quad+1
\end{array}
$$

从而获得 $r_1(x)=x^3+x^2+1$，$q_1(x)=x^2+x$。

接着用 $r_1(x)$除 $b(x)$。

$$
\begin{array}{r}
x+1 \\
x^3+x^2+1\overline{)\,x^4\quad\;+x^2+x+1} \\
\underline{x^4+x^3\quad\;+x} \\
x^3+x^2\quad+1 \\
\underline{x^3+x^2\quad+1}
\end{array}
$$

从而获得 $r_2(x)=0$，$q_2(x)=x+1$

因此，$\gcd[a(x), b(x)]=r_1(x)=x^3+x^2+1$。

5.5.4 小结

首先，我们讨论了一般多项式的算术。在一般多项式算术里，变量不予赋值，即不给多项式的变量赋值。相反，运用代数的一般规则对多项式进行算术操作（加、减、乘、除）。除非系数是域的元素，否则多项式除法是不行的。

接着，讨论了域 GF(p)上的多项式算术：加、减、乘、除。然而，除法不是整除，即通常除法的结果是商和余数。最后，说明了 Euclid 算法可以扩展用于求域上两个多项式的最大公因子。

本节论述的知识为下一节提供了一个基础。下一节中，将用多项式来构造阶为 p^n 的有限域。

5.6 有限域 GF(2^n)

在本章的前几节中，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 5.4 节中，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 5.2 所示）。当 $n>1$，p^n 上的多项式在模 p^n 运算时并不能产生一个域。在本节中，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)。

5.6.1 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，希望这个整数集中的数与给定的二进制位数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 0～255 之间的整数。但是由于 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，251～255 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 5.1 所示，以 8 为模的运算是定义合理的。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 5.2 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：

整数	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

目前，我们暂时不管表 5.2 中的矩阵是如何构造的，先做如下观察：

（1）加法和乘法表关于主对角线对称，这对应于加法和乘法的交换性。这个性质在表 5.1 里讲过，表 5.1 是模 8 算术。

（2）表 5.2 里的非零元素都有乘法逆，而表 5.1 不是。

（3）表 5.2 定义的集合满足有限域的所有要求，因此，我们可以称该域为 GF(2^3)。

（4）为方便，给出了 GF(2^3)元素的 3 位表示。

表 5.2 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

	×	000	001	010	011	100	101	110	111
		0	1	2	3	4	5	6	7
000	0	0	0	0	0	0	0	0	0
001	1	0	1	2	3	4	5	6	7
010	2	0	2	4	6	3	1	7	5
011	3	0	3	6	5	7	4	1	2
100	4	0	4	3	7	6	2	5	1
101	5	0	5	1	4	2	7	3	6
110	6	0	6	7	1	5	3	2	4
111	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，一个将整数集不均匀地映射到自身的算法用于加密时可能要弱于一个提供均匀映射的算法。正因为如此，有限域 GF(2^n)对加密算法是很有吸引力的。

总之，我们要寻找一个包含 2^n 个元素的集合，其上定义了加法和乘法使之成为一个域。我们给集合的每一个元素赋值为 0 到 2^n-1 之间的唯一整数。请记住我们不会用模算术，因为那样不能构成域。我们将会用多项式算术来构造需要的域。

5.6.2 多项式模运算

设集合 S 由域 Z_p 上次数小于等于 $n-1$ 的所有多项式组成。每一个多项式具有如下形式：

$$f(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0=\sum_{i=0}^{n-1}a_ix^i$$

其中，a_i 在集合$\{0, 1, \cdots, p-1\}$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

$$0，1，2，x，x+1，x+2，2x，2x+1，2x+2$$

当 $p=2$，$n=3$ 时，集合中共有 $2^3=8$ 个多项式，分别是

$$0，1，x，x+1，x^2，x^2+1，x^2+x，x^2+x+1$$

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）该运算遵循代数基本规则中的普通多项式运算规则及如下两条限制。

（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 $f(x)$，这个余数可表示为 $r(x)=f(x) \bmod m(x)$。

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 $f(x)=x^6+x^4+x^2+x+1$，$g(x)=x^7+x+1$，则

$$f(x)+g(x)=x^6+x^4+x^2+x+1+x^7+x+1=x^7+x^6+x^4+x^2$$

$$\begin{aligned}f(x)\times g(x)&=x^{13}+x^{11}+x^9+x^8+x^7+x^7+x^5+x^3+x^2+x\\&\quad+x^6+x^4+x^2+x+1\\&=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1\end{aligned}$$

$$\begin{array}{r}x^5+x^3\\ x^8+x^4+x^3+x+1\overline{)\,x^{13}+x^{11}+x^9+x^8\quad+x^6+x^5+x^4+x^3+1}\\ \underline{x^{13}\quad+x^9+x^8\quad+x^6+x^5}\\ x^{11}\quad+x^4+x^3\\ \underline{x^{11}\quad+x^7+x^6\quad+x^4+x^3}\\ +x^7+x^6\quad+1\end{array}$$

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类集合的概念。设 $m(x)$为 n 次多项式，则模 $m(x)$剩余类集合有 p^n 个元素，其中每个元素都可以表示成一个 m 次多项式（$m<n$）。

以 $m(x)$为模的剩余类$[x+1]$由所有满足 $a(x)\equiv(x+1) \pmod{m(x)}$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 5.2 的所有公理，于是可以形成一个有限域。还可以进一步看到，所有具有相同阶的有限域都是同构的，即任意两个具有相同阶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)，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 5.3 所示。由于该表和表 5.2 有相同的结构，所以我们已成功找到了定义阶为 2^3 的有限域的方法。

从表容易读出加法和乘法。例如，考虑式子 100+010=110。这等价于 x^2+x。再考虑 100×010=011，这等价于 $x^2 \times x=x^3$，约减后为 $x+1$，即 $x^3 \bmod (x^3+x+1)=x+1$ 等价于 011。

表 5.3 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
000	0	0	0	0	0	0	0	0	0
001	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

5.6.3 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，扩展 Euclid 算法则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 $a(x)$的次数且 gcd[$a(x)$, $b(x)$]=1，那么该算法能求出 $b(x)$以 $a(x)$为模的乘法逆元。若 $a(x)$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[$a(x)$, $b(x)$]=1。算法的描述方式和整数情形的扩展 Euclid 算法一样。给定两个多项式 $a(x)$和 $b(x)$，其中 $a(x)$的次数大于 $b(x)$的次数。我们希望解如下方程获得 $v(x)$,$w(x)$以及 $d(x)$，其中 $d(x)$=gcd[$a(x)$,$b(x)$]

$$a(x)v(x)+b(x)w(x)=d(x)$$

如果 $d(x)$=1，则 $w(x)$是 $b(x)$模 $a(x)$的乘法逆。计算过程如下：

用于多项式的扩展 Euclid 算法			
计　算	满　足	计　算	满　足
$r_{-1}(x)=a(x)$		$v_{-1}(x)=1; w_{-1}(x)=0$	$a(x)=a(x)v_{-1}(x)+b(x)w_{-1}(x)$
$r_0(x)=b(x)$		$v_0(x)=0; w_0(x)=1$	$b(x)=a(x)v_0(x)+b(x)w_0(x)$
$r_1(x)=a(x) \bmod b(x)$ $q1(x)=a(x)/b(x)$的商	$a(x)=q_1(x)b(x)+r_1(x)$	$v_1(x)=v_{-1}(x)-q_1(x)v_0(x)=1$ $w_1(x)=w_{-1}(x)-q_1(x)w_0(x)=-q_1(x)$	$r_1(x)=a(x)v_1(x)+b(x)w_1(x)$
$r_2(x)=b(x) \bmod r_1(x)$ $q_2(x)=b(x)/r_1(x)$的商	$b(x)=q_2(x)r_1(x)+r_2(x)$	$v_2(x)=v_0(x)-q_2(x)v_1(x)$ $w_2(x)=w_0(x)-q_2(x)w_1(x)$	$r_2(x)=a(x)v_2(x)+b(x)w_2(x)$
$r_3(x)=r_1(x) \bmod r_2(x)$ $q_3(x)=r_1(x)/r_2(x)$的商	$r_1(x)=q_3(x)r_2(x)+r_3(x)$	$v_3(x)=v_1(x)-q_3(x)v_2(x)$ $w_3(x)=w_1(x)-q_3(x)w_2(x)$	$r_3(x)=a(x)v_3(x)+b(x)w_3(x)$
⋮	⋮	⋮	⋮
$r_n(x)=r_{n-2}(x) \bmod r_{n-1}(x)$ $q_n(x)=r_{n-2}(x)/r_{n-3}(x)$的商	$r_{n-2}(x)=q_n(x)r_{n-1}(x)+r_n(x)$	$v_n(x)=v_{n-2}(x)-q_n(x)v_{n-1}(x)$ $w_n(x)=w_{n-2}(x)-q_n(x)w_{n-1}(x)$	$r_n(x)=a(x)v_n(x)+b(x)w_n(x)$
$r_{n+1}(x)=r_{n-1}(x) \bmod r_n(x)$ $q_{n+1}(x)=r_{n-1}(x)/r_{n-2}(x)$的商	$r_{n-1}(x)=q_{n+1}(x)r_n(x)+0$		$d(x)=\gcd(a(x), b(x))=r_n(x)$ $v(x)=v_n(x)$; $w(x)=w_n(x)$

表 5.4 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$(x^7+x+1)(x^7)\equiv 1 \pmod{(x^8+x^4+x^3+x+1)}$。

表 5.4　扩展 Euclid $[(x^8+x^4+x^3+x+1), (x^7+x+1)]$

初始化	$a(x)=x^8+x^4+x^3+x+1; v_{-1}(x)=1; w_{-1}(x)=0$ $b(x)=x^7+x+1; v_0(x)=0; w_0(x)=1$
循环 1	$q_1(x)=x; r_1(x)=x^4+x^3+x^2+1$ $v_1(x)=1; w_1(x)=x$
循环 2	$q_2(x)=x^3+x^2+1; r_2(x)=x$ $v_2(x)=x^3+x^2+1; w_2(x)=x^4+x^3+x+1$
循环 3	$q_3(x)=x^3+x^2+x; r_3(x)=1$ $v_3(x)=x^6+x^2+x+1; w_3(x)=x^7$
循环 4	$q_4(x)=x; r_4(x)=0$ $v_4(x)=x^7+x+1; w_4(x)=x^8+x^4+x^3+x+1$
结果	$d(x)=r_3(x)=\gcd(a(x), b(x))=1$ $w(x)=w_3(x)=(x^7+x+1)^{-1} \bmod (x^8+x^4+x^3+x+1)=x^7$

5.6.4　计算上的考虑

GF(2^n)中的多项式$f(x)$

$$f(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0=\sum_{i=0}^{n-1}a_ix^i$$

可以由它的 n 个二进制系数$(a_{n-1}, a_{n-2}, \cdots, a_0)$唯一地表示。因此，GF($2^n$)中的每个多项式都可以表示成一个 n 位的二进制数。

表 5.2 和表 5.3 给出了以 $m(x)=x^3+x+1$ 为模的域 GF(2^3)的加法和乘法表。其中，表 5.2 用二进制整数表示，而表 5.3 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相加，而对于 Z_2 上的多项式，加法其实就是异或（XOR）运算。所以，GF(2^n)中的两个多项式加法等同于按位异或运算。

考虑前面例子中 GF(2^8)上的两个多项式

$$f(x)=x^6+x^4+x^2+x+1 \text{ 和 } g(x)=x^7+x+1$$

$(x^6+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^6+x^4+x^2$		（多项式表示）
01010111 ⊕ 10000011	=11010100	（二进制表示）
{57}⊕{83}	={D4}	（十六进制表示）①

乘法 简单的异或运算不能完成 GF(2^n)上的乘法。但是可以使用一种相当直观且容易实现的技巧。以 $m(x)=x^8+x^4+x^3+x+1$ 为多项式的有限域 GF(2^8)是 AES 中用到的有限域，我们将参照该域来讨论该技巧。这个技巧容易推广到域 GF(2^n)。

这个技巧基于下面的等式

$$x^8 \bmod m(x)=[m(x)-x^8]=(x^4+x^3+x+1) \tag{5.4}$$

稍做思考就不难证明式（5.4）是正确的。如果不确信，可以除一下。一般地，在 GF(2^n)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(2^8)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

$$x\times f(x)=(b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x) \bmod m(x) \tag{5.5}$$

如果 $b_7=0$，那么结果就是一个次数小于 8 的多项式，已经是约化后的形式，不需要进一步计算。如果 $b_7=1$，那么可以通过式（5.4）进行模 $m(x)$的约化：

$$x\times f(x)=(b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x)+(x^4+x^3+x+1)$$

这表明乘以 x（即 00000010）的运算可以通过左移一位后再根据条件按位异或 00011011（其代表 x^4+x^3+x+1）来实现。总结如下：

$$x\times f(x)=\begin{cases} b_6b_5b_4b_3b_2b_1b_0\,0, & \text{如果} b_7=0 \\ b_6b_5b_4b_3b_2b_1b_0\,0\oplus 00011011, & \text{如果} b_7=1 \end{cases} \tag{5.6}$$

乘以 x 的更高次幂可以通过重复使用式（5.6）来实现。这样一来，GF(2^8)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，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 01010111 的中间结果

01010111×00000010=10101110
01010111×00000100=01011100 ⊕ 00011011=01000111
01010111×00001000=10001110
01010111×00010000=00011100 ⊕ 00011011=00000111
01010111×00100000=00001110
01010111×01000000=00011100
01010111×10000000=00111000

所以

01010111×10000011=01010011×(00000001 ⊕ 00000010 ⊕ 10000000)
=01010111 ⊕ 10101110 ⊕ 00111000=11000001

这等价于 x^7+x^6+1。

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 位二进制数，一个字节的二进制数用括号括起来的两个十六进制字符表示。

5.6.5 使用生成元

定义有限域 GF(2^n)的另一种等价方式有时更方便，它使用相同的不可约多项式。首先，需要两个定义：阶为 q 的有限域 F 的生成元是一个元素，记为 g，该元素的前 $q-1$ 个幂构成了 F 的所有非零元素，即域 F 的元素为 0，g^0，…，g^{q-2}。考虑由多项式 $f(x)$定义的域 F，如果 F 内的一个元素 b 满足 $f(b)=0$，则称 b 为多项式的 $f(x)$的根。最后，可以证明一个不可约多项式的根 g 是这个不可约多项式定义的有限域的生成元。

考虑前面讨论的由不可约多项式 x^3+x+1 定义的有限域 GF(2^3)。设生成元为 g，则 g 满足 $f(g)=g^3+g+1=0$。如前所述，请记住我们不要这个方程的数值解，只处理多项式算术（系数运算是模 2 的）。因此，方程的解满足 $g^3=-g-1=g+1$。可以证明事实上 g 产生了所有次数小于 3 的多项式，则有

$$g^4 = g(g^3) = g(g+1) = g^2+g$$
$$g^5 = g(g^4) = g(g^2+g) = g^3+g^2 = g^2+g+1$$
$$g^6 = g(g^5) = g(g^2+g+1) = g^3+g^2+g = g^2+g+g+1 = g^2+1$$
$$g^7 = g(g^6) = g(g^2+1) = g^3+g = g+g+1=1 = g^0$$

于是可看到，g 的幂产生了 GF(2^3)的所有非 0 多项式。同样应清楚，对所有 k，有 $g^k=g^{k \bmod 7}$。表 5.5 列出了幂表示，多项式表示及二进制表示。

这种幂表示使乘法更容易。计算幂表示的乘法，加法按模 7 进行。例如

$$g^4 \times g^6 = g^{(10 \bmod 7)} = g^6 = g+1.$$

用多项式运算可以得到同样的结果。我们有 $g^4 = g^2+g$ 和 $g^6 = g^2+1$。于是，$(g^2+g)\times(g^2+1) = g^4+g^3+g^2+g$。接下来通过除法来确定$(g^4+g^3+g^2+g) \bmod (g^3+g+1)$

$$\begin{array}{r} g+1 \\ g^3+g+1 \overline{)\, g^4+g^3+g^2+g} \\ \underline{g^4 \quad +g^2+g} \\ g^3 \\ \underline{g^3+ \quad g+1} \\ g+1 \end{array}$$

我们得到了结果 $g+1$，与用幂表示所得的结果相同。

表 5.6 给出了用幂表示的 GF(2^3)上的加法表和乘法表。注意，这与表 5.3 中的多项式表示的运算结果相同。

表 5.5 模 x^3+x+1 的域 GF(2^3)的生成元

幂表示	多项式表示	二进制表示	十（十六）进展表示
0	0	000	0
$g^0(=g^7)$	1	001	1
g^1	g	010	2
g^2	g^2	100	4
g^3	$g+1$	011	3
g^4	g^2+g	110	6
g^5	g^2+g+1	111	7
g^6	g^2+1	101	5

表 5.6 使用生成元的 GF(2^3)算术运算，其模多项式为(x^3+x+1)

	+	000	001	010	100	011	110	111	101
		0	1	G	g^2	g^3	g^4	g^5	g^6
000	0	0	1	G	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1
001	1	1	0	$g+1$	g^2+1	g	g^2+g+1	g^2+g	g^2
010	g	g	$g+1$	0	g^2+g	1	g^2	g^2+1	g^2+g+1
100	g^2	g^2	g^2+1	g^2+g	0	g^2+g+1	g	$g+1$	1
011	g^3	$g+1$	g	1	g^2+g+1	0	g^2+1	g^2	g^2+g
110	g^4	g^2+g	g^2+g+1	g^2	g	g^2+1	0	1	$g+1$
111	g^5	g^2+g+1	g^2+g	g^2+1	$g+1$	g^2	1	0	g
101	g^6	g^2+1	g^2	g^2+g+1	1	g^2+g	$g+1$	g	0

(a) 加法

	×	000	001	010	100	011	110	111	101
		0	1	G	g^2	g^3	g^4	g^5	g^6
000	0	0	0	0	0	0	0	0	0
001	1	0	1	g	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1
010	g	0	g	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1	1
100	g^2	0	g^2	$g+1$	g^2+g	g^2+g+1	g^2+1	1	g
011	g^3	0	$g+1$	g^2+g	g^2+g+1	g^2+1	1	g	g^2
110	g^4	0	g^2+g	g^2+g+1	g^2+1	1	g	g^2	$g+1$
111	g^5	0	g^2+g+1	g^2+1	1	g	g^2	$g+1$	g^2+g
101	g^6	0	g^2+1	1	g	g^2	$g+1$	g^2+g	g^2+g+1

(b) 乘法

通常，对于由不可约多项式 $f(x)$生成的域 GF(2^n)，有 $g^n=f(g)-g^n$。计算 g^{n+1} 到 g^{2^n-2} 的值。域的元素对应 g 的幂：g^0 到 g^{2^n-2}，另外再加上 0 元素。域元素的乘法用等式 $g^k = g^{k \bmod (2^n-1)}$ 进行，k 为任意整数。

5.6.7 小结

本节中，我们说明了如何构造阶为 2^n 的有限域。特别地，我们定义了具有如下性质的域 GF(2^n)：

1. GF(2^n)由 2^n 个元素组成。
2. 集合上定义了二元运算+和×。加、减、乘、除可以在集合内进行。除 0 元素外的其他元素都有乘法逆元。

我们说明了 GF(2^n)的元素可由二元域上所有次数不大于 n–1 的多项式集合定义。每一个多项式可由唯一的 n 位数来表示。其上的算术是模某个次数为 n 的不可约多项式的多项式算术。我们还介绍了 GF(2^n)的一种等价定义。该定义利用了生成元，其运算用生成元的幂来进行。

5.7 关键术语、思考题和习题

关键术语

阿贝尔群	最大公因子	阶	结合律
群	多项式	系数集合	单位元
多项式算术	交换律	无限域	多项式环
交换环	无限群	素数	循环群

整域	素多项式	因子	逆元素
互素	Euclid 算法	不可约多项式	剩余
域	模算术	环	有限域
模多项式算术	有限群	首 1 多项式	生成元

思考题

5.1 简短定义一个群。

5.2 简短定义一个环。

5.3 简短定义一个域。

5.4 列举三类多项式运算。

习题

5.1 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。

5.2 判断对于下列运算，模 3 剩余类组成的集合是否能构成一个群。

（a）模加

（b）模乘法

5.3 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。

5.4 类似于表 5.1，给出对应于 GF(5)的相应表。

5.5 证明以域元素为系数的多项式组成的集合是一个环。

5.6 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 的两个多项式乘积的次数为 $m+n$。

（c）次数分别为 m 和 n 的两个多项式和的次数为 $\max(m, n)$。

5.7 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）$(6x^2+x+3)\times(5x^2+2)$

5.8 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（仔细考虑）

5.9 求下列各对多项式的最大公因式。

（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。

5.10 类似于表 5.3，给出以 $m(x)=x^2+x+1$ 为模的域 GF(4)的对应表。

5.11 求 x^3+x+1 在 GF(2^4)里的乘法逆元，模 $m(x)=x^4+x+1$。

5.12 类似于表 5.5，给出以 $m(x)=x^4+x+1$ 为模的域 GF(2^4)的对应表。

编程习题

5.13 写一个简单的程序，能完成域 GF(2^4)的四种运算功能。可以用查表的方法来求乘法逆。

5.14 写一个简单的程序，能完成域 GF(2^8)的四种运算功能。求乘法逆应该一步完成。

第 6 章　高级加密标准

学 习 目 标

通过本章的学习应具备以下能力：

- 简要介绍高级加密标准（AES）的一般结构
- 理解 AES 的四种变换
- 理解 AES 的密钥扩展算法
- 理解有限域 GF(2^8)内多项式的使用

美国国家标准技术研究所（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，旨在取代 DES 成为广泛使用的标准。与公钥密码，如 RSA 相比，AES 以及大多数对称密码的结构都很复杂，描述起来不能像 RSA 以及许多其他密码学算法那样容易。因此，读者也许希望从简化版的 AES 开始，附录 I 为读者介绍了简化版本的 AES。读者能使用该版本手工实现加密和解密，从而对于算法工作的具体细节有更深的理解。教学实践也表明对于简化版本 AES 的学习有助于更深入地理解 AES[①]。一种可能的方法是先读本章，然后仔细阅读附录 I，最后再重新阅读本章的主体部分。

附录 H 介绍了 NIST 用来挑选 AES 算法的评估标准，以及挑选 Rijndael 作为胜出算法的合理性分析。这份材料不但对理解 AES 的设计有帮助，而且对理解评估其他对称加密算法的标准也有帮助。

6.1　有限域算术

AES 中的所有运算都是在 8 位的字节上进行的。特别地，加减乘除算术都是在有限域 GF(2^8)上进行的。5.6 节比较详细地讨论了这些运算。对于没有读过第 5 章的读者，本节总结一些重要的概念。对于读过第 5 章的读者，本节可以作为一个快速回顾。

本质上，一个域就是一个集合，在该集合内，可以进行加减乘除运算，结果不离开集合。除法遵循如下规则：$a/b = a(b^{-1})$。一个有限域（具有有限个元素）的例子是集合 Z_p={0，1，…，p–1}，其中 p 是一个素数，且其内的运算是模 p 进行的。

实际上所有的加密算法，包括对称的和非对称的，都用到了整数的算术运算。如果算法用到了除法，需要用到定义在域上的算术。这是因为除法需要每一个非零元素有一个乘法逆元。出于习惯和实现的效率考虑，给定整数的位数，我们希望使用具有该位长度的所有整数，而不浪费一些位模式。即我们希望使用 0 到 2^n–1 内的所有整数，其刚好是一个 n 位的字。遗憾的是，这些整数的集合 Z_{2^n} 在模算术下并不是一个域。例如，整数 2 在 Z_{2^n} 内没有乘法逆元。也就是说不存在 b 使得 $2b \bmod 2^n=1$。

① 然而读者在第一次阅读时仍可以跳过附录 I。如果读者对于 AES 的细节不能充分理解的话，则可以回过头从简化版本的 AES 开始。

有一种方法可以定义含有 2^n 个元素的有限域，该域被称为 GF(2^n)。考虑所有次数小于等于 $n-1$，系数为 0，1 的多项式集合 S。因此，每一个多项式具有如下形式：

$$f(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0=\sum_{i=0}^{n-1}a_ix^i$$

其中每个 a_i 取值 0 或 1，集合 S 内共有 2^n 个元素。对于 n=3，集合内的 8 个多项式为

0	x	x^2	x^2+x
1	$x+1$	x^2+1	x^2+x+1

定义合适的算术运算，则每一个这样的集合 S 都是一个有限域。定义由如下各项组成。

（1）该运算遵循代数基本规则中的普通多项式运算规则及如下两条限制。

（2）系数运算以 2 为模，这和 XOR 运算是一样的。

（3）如果乘法运算的结果是次数大于 $n-1$ 的多项式，那么必须用某个次数为 n 的不可约多项式 $m(x)$进行约化，即用 $m(x)$去除并取余式。对于多项式 $f(x)$，这个余数可表示为 $r(x)=f(x) \bmod m(x)$。一个多项式 $m(x)$称为是不可约多项式，当且仅当该多项式不能表示为次数小于 $m(x)$的次数的两个多项式的乘积。

例如，为了构造 GF(2^3)，需要选择次数为 3 的不可约多项式。仅仅有两个这样的多项式：(x^3+x^2+1)以及(x^3+x+1)。加法等价于各项的对位异或，因此，$(x+1)+x=1$。

GF(2^n)内的一个多项式可以由它的二元系数$(a_{n-1}a_{n-2}\cdots a_0)$唯一表示。因此，GF($2^n$)内的每个多项式可以由 n 位的数来表示。加法由两个 n 位的数进行对位异或来实现。GF(2^n)内的乘法没有简单的异或操作来实现，然而却有一种相当直观且容易实现的技巧。本质上，GF(2^n)内的数乘以 2 可以先左移，然后根据条件异或上一个常数。乘上一个大数可以重复运用该规则。

例如，AES 使用有限域 GF(2^8)内的算术，其中不可约多项式为 $m(x)=x^8+x^4+x^3+x+1$。考虑两个元素 $A=(a_7a_6\cdots a_1a_0)$和 $B=(b_7b_6\cdots b_1b_0)$。$A+B=(c_7c_6\cdots c_1c_0)$，其中 $c_i=a_i \oplus b_i$。当 $a_7=0$ 时乘积$\{02\}\cdot A$ 等于$(a_6\cdots a_1a_00)$，当 $a_7=1$ 时，乘积$\{02\}\cdot A$ 等于$(a_6\cdots a_1a_00) \oplus (00011011)$[①]。

总结一下，AES 在 8 位的字节上运算。两个字节的加定义为对位异或操作。两个字节的乘定义为有限域 GF(2^8)内的乘法，其中不可约多项式[②]为 $m(x)=x^8+x^4+x^3+x+1$。Rijndael 的开发者在谈到选择该多项式的动机时说这个多项式在文献[LIDL94]里是 30 个不可约多项式列表的第一个多项式，所以他们就选择了该多项式。

6.2　AES 的结构

6.2.1　总体结构

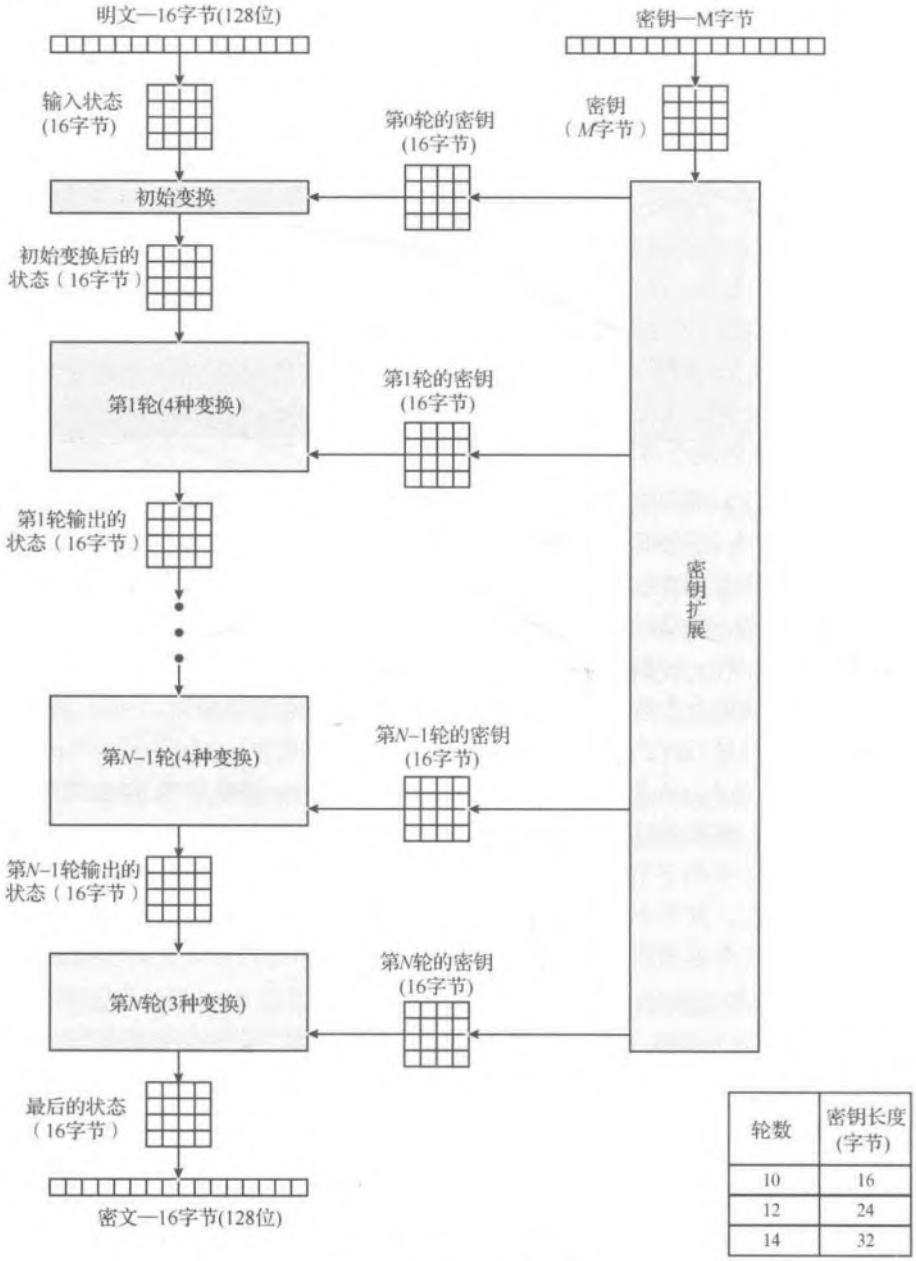
图 6.1 展示了 AES 加密过程的总体结构。明文分组的长度为 128 位即 16 字节，密钥长度可以为 16 字节，24 字节或 32 字节（128 位，192 位或 256 位）。根据密钥的长度，算法被称为 AES-128，AES-192 或 AES-256。

加密和解密算法的输入是一个 128 位分组。在 FIPS PUB 197 里，这个分组被描述为 4×4 的字节方阵。这个分组被复制到 **state** 数组，并在加密或解密的各个阶段被修改。图 6.2(a)描述了这些操作。同样地，密钥也被描述为字节的方阵。这个密钥接着被扩展为密钥字阵列。图 6.2(b)展示了 128 位密钥的扩展。每个字是 4 字节，128 位的密钥最终扩展为 44 字的序列。注意在矩阵中字节是按照

① 在 FIPS PUB 197 里，一个十六进制数字用大括号括起来表示。本章都使用这个规则。

② 本次讨论的其余部分，用到的域 GF(2^8)是指用这个多项式定义的有限域。

列进行排序的。所以，加密算法的 128 位明文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。相似地，扩展密钥的前四个字节(形成一个字)被放在 ***w*** 矩阵的第一列。



轮数	密钥长度(字节)
10	16
12	24
14	32

图 6.1 AES 的加密过程

密码由 *N* 轮组成，其中轮数依赖于密钥长度：16 字节密钥是 10 轮，24 字节密钥对应 12 轮，32 字节密钥对应 14 轮（参见表 6.1）。前 *N*–1 轮由 4 个不同的变换组成：字节代替，行移位，列混淆和轮密钥加，后面将逐一介绍。最后一轮仅包含三个变换，而在第一轮的前面有一个起始的单变换（轮密钥加），可以看成是 0 轮。每一个变换输入一个或多个 4×4 的矩阵，并输出一个 4×4 的矩阵。图 6.1 说明每轮的输出是一个 4×4 的矩阵，最后一轮的输出为密文。同样，密钥扩展函数产生 *N*+1 轮密钥，它们是互不相同的 4×4 矩阵。每一个轮密钥作为每轮的轮密钥加变换的一种输入。

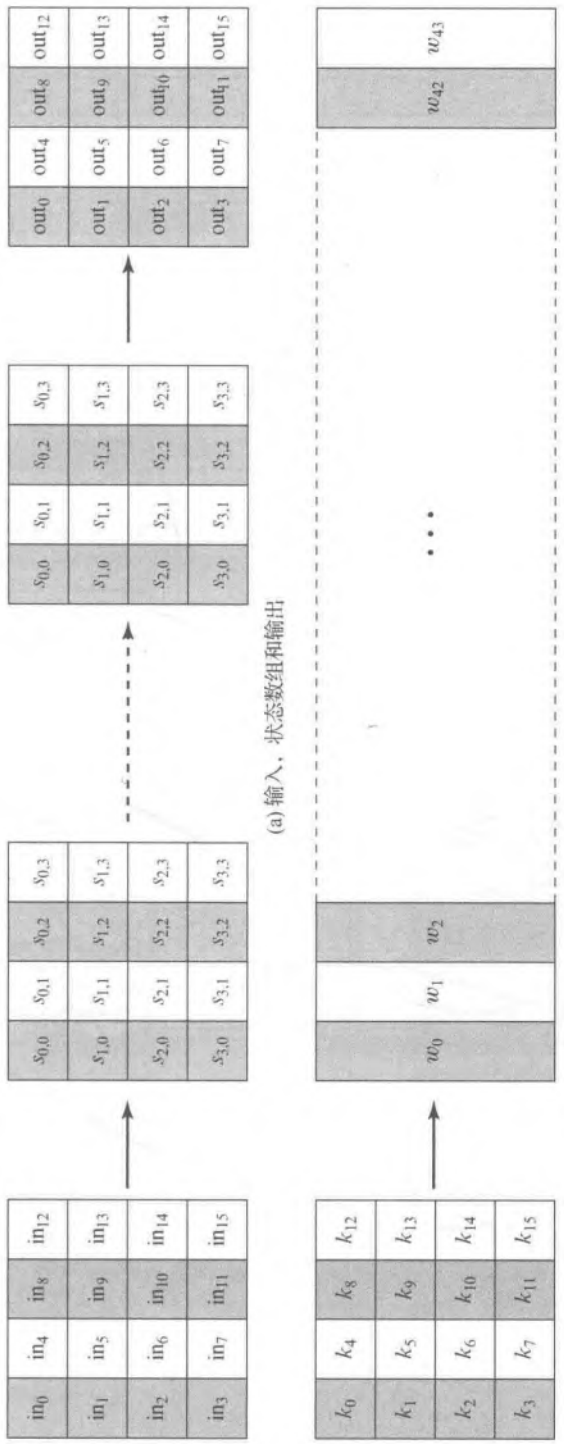


图 6.2　AES 的数据结构

表 6.1 AES 的参数

密钥长度（字/字节/位）	4/16/128	6/24/192	8/32/256
明文分组长度（字/字节/位）	4/16/128	4/16/128	4/16/128
轮数	10	12	14
每轮的密钥长度（字/字节/位）	4/16/128	4/16/128	4/16/128
扩展密钥长度（字/字节/位）	44/176	52/208	60/240

6.2.2 详细结构

图 6.3 更加详细地展示了 AES 密码，指明了每轮的变换顺序，并展示了相应的解密函数。同第 4 章的做法一样，图中加密的过程是沿着页面向下，而解密是沿着页面向上。

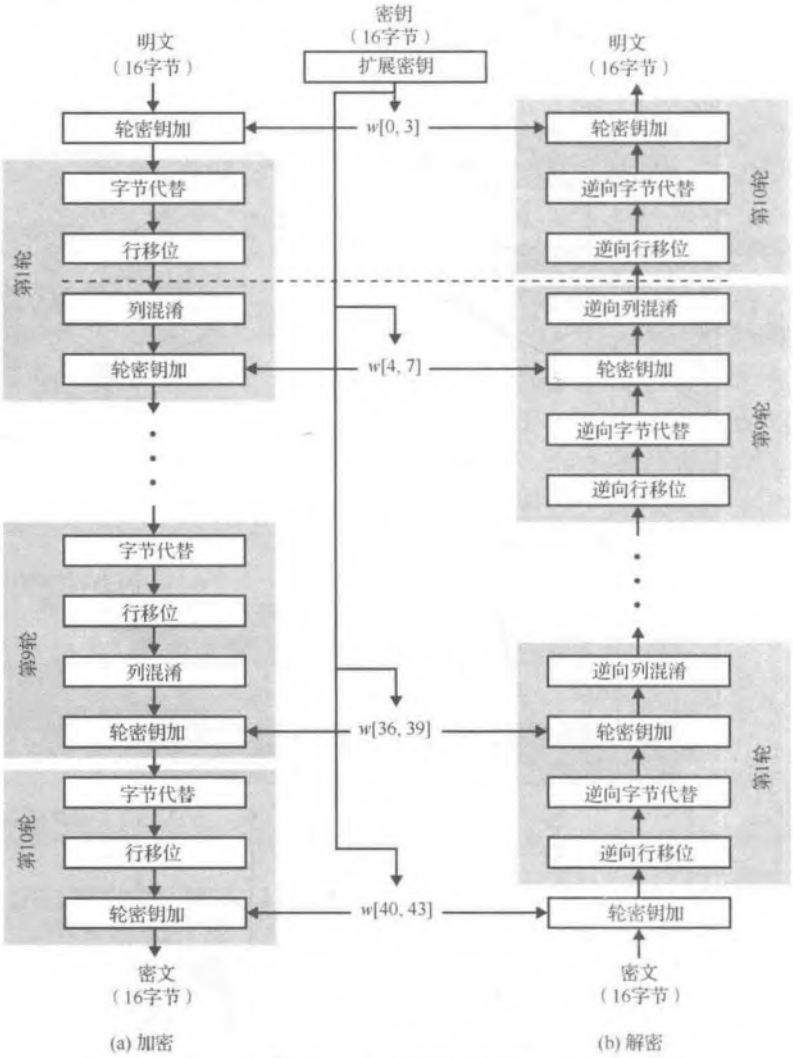


图 6.3 AES 加密和解密

在讨论 AES 的全部细节之前，我们对 AES 的总体结构做一些讨论。

（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。AES 算法没有使用 Feistel 结构，而是在每一轮都使用代替和混淆将整个数据分组作为一个单一的矩阵处理。

（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 6.3 中可以看出，每轮有四个不同的字（128 位）作为该轮的轮密钥。

（3）由 4 个不同的阶段组成，包括一个置换和 3 个代替：

- **字节代替**（Substitute Bytes） 用一个 S 盒完成分组的字节到字节的代替。
- **行移位**（ShiftRows） 一个简单的置换。
- **列混淆**（MixColumns） 利用域 GF(2^8)上的算术特性的一个代替。
- **轮密钥加**（AddRoundKey） 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 9 轮迭代运算，每轮都包含所有 4 个阶段的代替，接着是第 10 轮的 3 个阶段。图 6.4 描述了包含全部加密轮的结构。

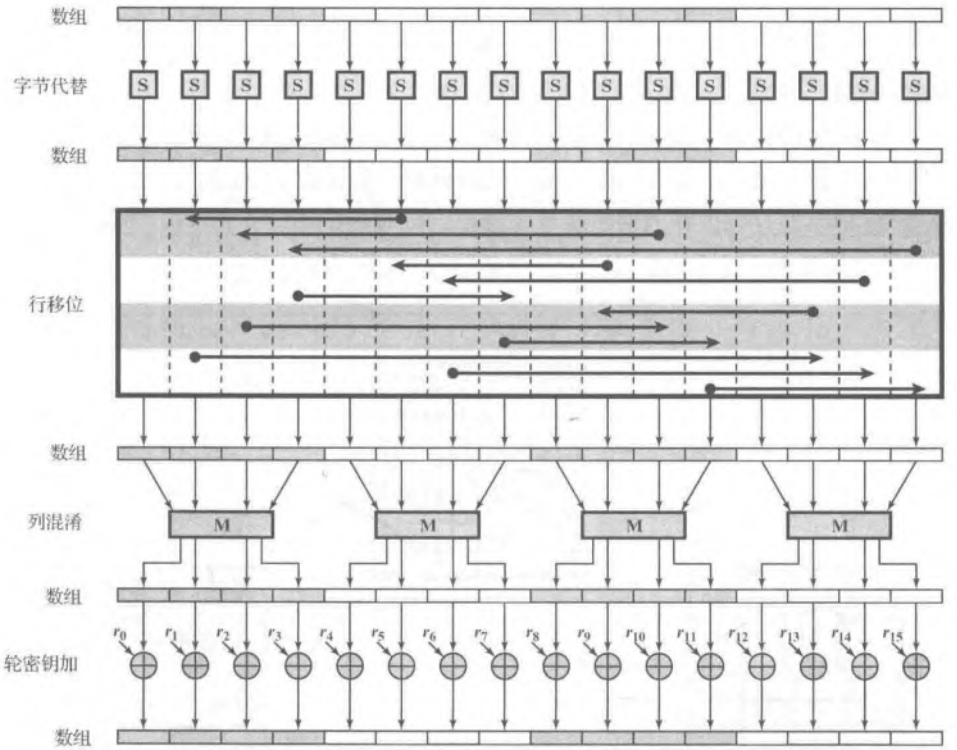


图 6.4　AES 的一轮加密过程

（5）仅仅在轮密钥加阶段中使用密钥。由于这个原因，该算法以轮密钥加开始，以轮密钥加结束。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一个分组的 XOR 加密（轮密钥加），接着对这个分组混淆（其他的三个阶段），再接着又是 XOR 加密，如此交替执行。这种方式非常有效且非常安全。

（7）每个阶段均可逆。对字节代替，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 $A \oplus B \oplus B = A$。

（8）同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 6.3 中加密和解密流程在纵向上是相反的。在每个水平点上（如图中由破折号组成的线），state 数组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。

6.3 AES 的变换函数

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），接着讨论该阶段的基本原理。

6.3.1 字节代替变换

正向和逆向变换 被称为字节代替的正向字节代替变换是一个简单的查表操作[参见图 6.5(a)]。AES 定义了一个 S 盒[参见表 6.2(a)]，它是由 16×16 个字节组成的矩阵，包含了 8 位所能表示的 256 个数的一个置换。状态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，以这些行列值作为索引从 S 盒的对应位置取出元素作为输出。例如，十六进制{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}。

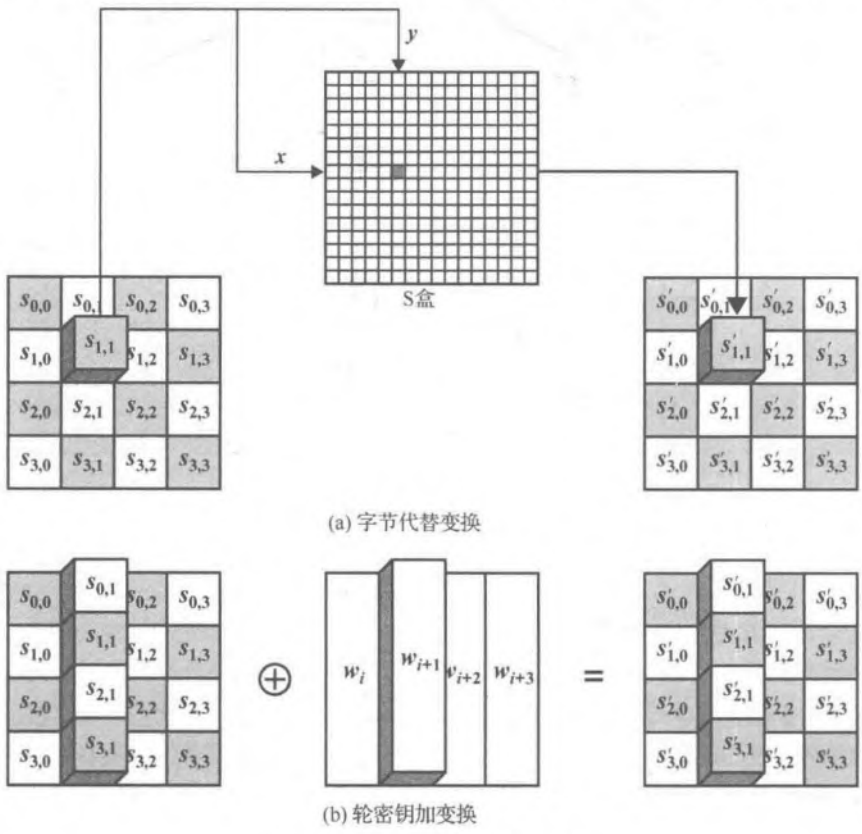


图 6.5 AES 的字节层操作

① 在 FIPS PUB 197 中，一个十六进制数用花括号来表示，本章我们也采用这种表示方法。

表 6.2　AES 的 S-盒

		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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

(a) S 盒

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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	24	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

(b) 逆 S 盒

下面是一个字节代替变换的例子：

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

S 盒按如下的方式构造[参见图 6.6(a)]：

（1）按字节值的升序逐行初始化 S 盒。第一行是{00}，{01}，{02}，…，{0F}；第二行是{10}，

{11}，…，{1F}等。因此，在行 y 列 x 的字节值是{yx}。

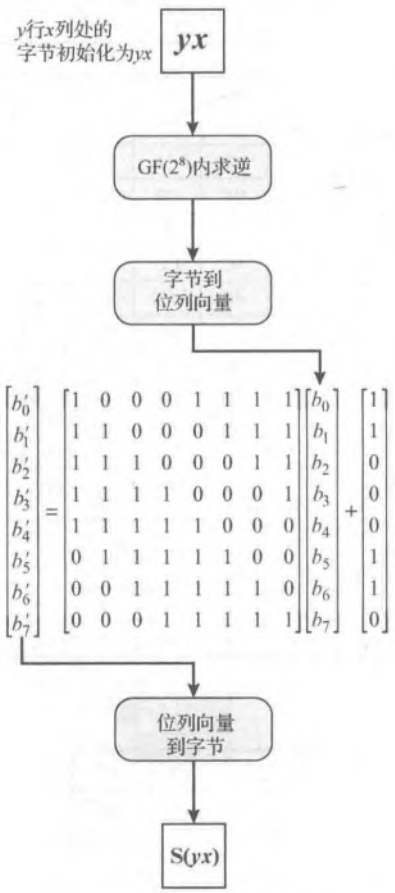
（2）把 S 盒中的每个字节映射为它在有限域 GF(2^8)中的逆；{00}被映射为它自身{00}。

（3）把 S 盒中的每个字节的 8 个构成位记为($b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$)。对 S 盒的每个字节的每个位做如下的变换：

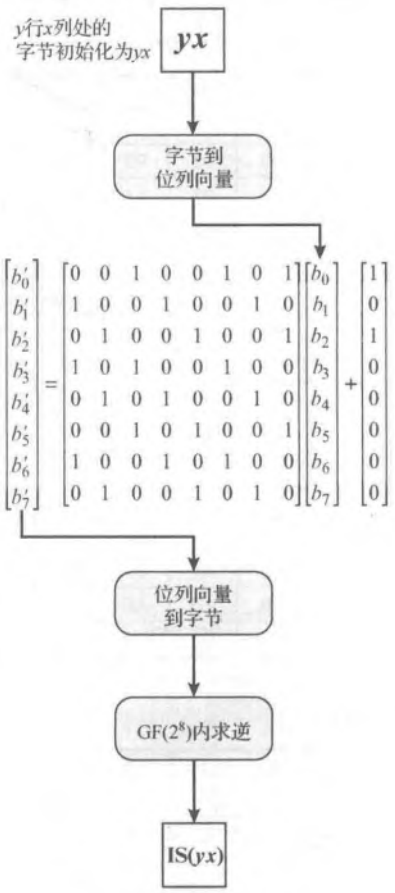
$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \tag{6.1}$$

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(′)表示变量的值要被等式右边的值更新。AES 标准用矩阵形式描述了这个变换

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{6.2}$$



(a) S盒y行x列处的字节计算



(b) 逆S盒y行x列处的字节计算

图 6.6 S 盒与逆 S 盒的结构

式（6.2）的解释需要仔细。在通常的矩阵乘法中[①]，乘积矩阵中的每个元素是一行和一列对应元素乘积的和。在这个例子中，乘积矩阵中每个元素是一行和一列对应元素乘积按位异或的值。进一步而言，式（6.2）中最终的加法是按位异或的。回忆 5.6 节知对位异或是 GF(2^8)内的加法。

例如，我们考虑输入值为{95}的情况。在 GF(2^8)中{95}$^{-1}$ 的乘法逆为{8A}，用二进制表示就是 10001010。用式（6.2）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 6.2(a)中得到证实。

逆字节代替变换利用了表 6.2(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[参见图 6.6(b)]是利用式（6.1）的逆变换，然后再求其在 GF（2^8）内的乘法逆。该逆变换是

$$b'_i = b_{(i+2)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus d_i$$

在这里，字节 d = {05}，或者 00000101。我们可以用如下的方式来描述这个转换：

$$\begin{bmatrix}b'_0\\b'_1\\b'_2\\b'_3\\b'_4\\b'_5\\b'_6\\b'_7\end{bmatrix}=\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}+\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}$$

为了理解逆字节代替变换是字节代替变换的逆，令字节代替变换和逆字节代替变换中的矩阵分别为 ***X*** 和 ***Y***，常量 c、d 的向量表示分别为 ***C*** 和 ***D***。对某个 8 位的向量 ***B***，式（6.2）变成了 ***B′ = XB⊕C***。我们需要证明 ***Y(XB⊕C)⊕D=B***。将括号里的内容乘出来，即为我们应该证明的 ***YXB⊕YC⊕D=B***。这变成了

$$\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus$$

① 对于矩阵和向量乘法规则的简要介绍，请参考附录 E。

$$\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=$$

$$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}\oplus\begin{bmatrix}1\\0\\1\\0\\0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7\end{bmatrix}$$

我们验证了 YX 等于单元矩阵，$YC=D$，于是 $YC\oplus D$ 等于零向量。

基本原理 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相关性很低的设计且输出值不能是输入的线性数学函数[DAEM01]。非线性度的产生是由于使用了乘法逆。另外，在式（6.1）中所选择的常量使得在 S 盒中没有不动点[S-box(a)=a]，也没有“反不动点”[S-box(a)=$\overline{a}$]，其中 $\overline{a}$ 表示 a 的逐位取反。

当然，S 盒必须是可逆的，即逆 S-box[S-box(a)]=a。然而，因 S-box(a)=逆 S-box(a)不成立，在这个意义上 S 盒不是自逆的。例如，S-box({95})={2A}，但逆 S-box({95})={AD}。

6.3.2 行移位变换

正向和逆向变换 图 6.7(a)描述了正向行移位变换。状态的第一行保持不变。把状态的第二行循环左移一个字节，状态的第三行循环左移两个字节，状态的第四行循环左移三个字节。行移位变换的一个例子如下：

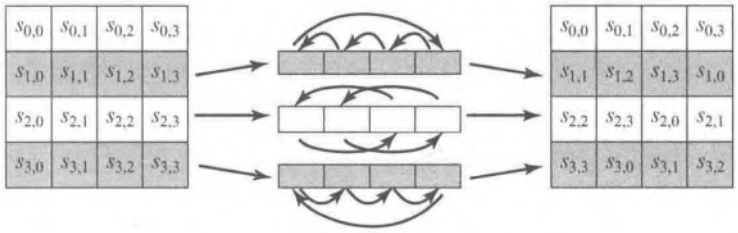
87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

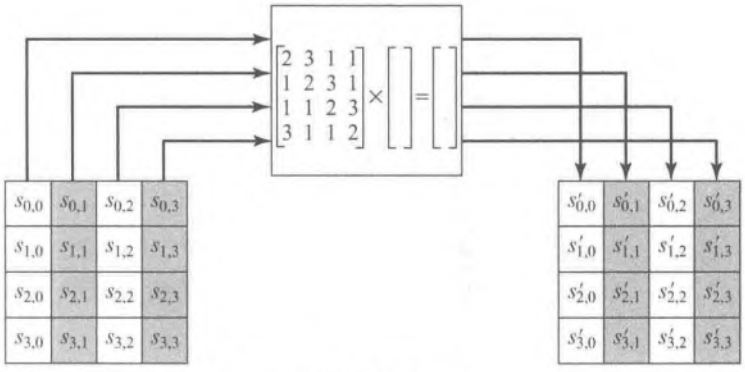
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

逆向行移位变换将状态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，其他行类似。

基本原理 行移位变换要比它看起来有用得多。这是因为状态和密码算法的输入输出数据一样，是一个由 4 列字节组成的数组，其中每列由 4 字节组成。因此在加密过程中，明文的前 4 个字节直接被复制到状态的第一列中，接着的四个字节被复制到状态的第二列中，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状态上。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。同时请注意这个转换确保了某列中的 4 字节被扩展到了 4 个不同的列。图 6.4 说明了这个效果。



(a) 行移位变换



(b) 列混淆变换

图 6.7　AES 的行与列操作

6.3.3　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。这个变换可由下面基于状态的矩阵乘法表示[参见图 6.7(b)]：

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \tag{6.3}$$

乘积矩阵中的每个元素均是一行和一列中的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。状态中单列的列混淆变换表示为

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned} \tag{6.4}$$

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

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。

让我们来验证一下该例子的第一列。在 5.6 节中提到在 GF(2^8)中，加法就是按位 XOR 操作，乘法的操作是按照式（4.14）所建立的规则进行的。特别是将某值乘上 x（即{02}）其结果就是将该值向左移一位，如果该值的最左边的位为 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：

$$
\begin{aligned}
(\{02\}\cdot\{87\})\oplus(\{03\}\cdot\{6E\})\oplus\{46\}\quad\oplus\{A6\}\quad&=\{47\}\\
\{87\}\quad\oplus(\{02\}\cdot\{6E\})\oplus(\{03\}\cdot\{46\})\oplus\{A6\}\quad&=\{37\}\\
\{87\}\quad\oplus\{6E\}\quad\oplus(\{02\}\cdot\{46\})\oplus(\{03\}\cdot\{A6\})&=\{94\}\\
(\{03\}\cdot\{87\})\oplus\{6E\}\quad\oplus\{46\}\quad\oplus(\{02\}\cdot\{A6\})&=\{ED\}
\end{aligned}
$$

对第一个方程，有{02}·{87} = (0000 1110) ⊕ (0001 1011)=(0001 0101)；{03}·{6E} = {6E}⊕({02}·{6E}) = (0110 1110) ⊕(1101 1100) =(1011 0010)。于是

$$
\begin{aligned}
\{02\}\cdot\{87\}&=00011100\\
\{03\}\cdot\{6E\}&=10010010\\
\{46\}\quad&=01000110\\
\{A6\}\quad&=\underline{10100110}\\
&\quad 01000111=\{47\}
\end{aligned}
$$

其他的方程也可以通过类似的方式得以验证。

逆向列混淆变换可由如下的矩阵乘法定义：

$$
\begin{bmatrix}0E&0B&0D&09\\09&0E&0B&0D\\0D&09&0E&0B\\0B&0D&09&0E\end{bmatrix}\begin{bmatrix}s_{0,0}&s_{0,1}&s_{0,2}&s_{0,3}\\s_{1,0}&s_{1,1}&s_{1,2}&s_{1,3}\\s_{2,0}&s_{2,1}&s_{2,2}&s_{2,3}\\s_{3,0}&s_{3,1}&s_{3,2}&s_{3,3}\end{bmatrix}=\begin{bmatrix}s'_{0,0}&s'_{0,1}&s'_{0,2}&s'_{0,3}\\s'_{1,0}&s'_{1,1}&s'_{1,2}&s'_{1,3}\\s'_{2,0}&s'_{2,1}&s'_{2,2}&s'_{2,3}\\s'_{3,0}&s'_{3,1}&s'_{3,2}&s'_{3,3}\end{bmatrix}\tag{6.5}
$$

从直观上，不容易看出式（6.5）是式（6.3）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

$$
\begin{bmatrix}0E&0B&0D&09\\09&0E&0B&0D\\0D&09&0E&0B\\0B&0D&09&0E\end{bmatrix}\begin{bmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\tag{6.6}
$$

即逆变换矩阵乘以正向变换矩阵结果为单位矩阵，为了验证式(6.6)的第一列，需要进行如下运算：

$$
\begin{aligned}
(\{0E\}\cdot\{02\})\oplus\{0B\}\oplus\{0D\}\oplus(\{09\}\cdot\{03\})&=\{01\}\\
(\{09\}\cdot\{02\})\oplus\{0E\}\oplus\{0B\}\oplus(\{0D\}\cdot\{03\})&=\{00\}\\
(\{0D\}\cdot\{02\})\oplus\{09\}\oplus\{0E\}\oplus(\{0B\}\cdot\{03\})&=\{00\}\\
(\{0B\}\cdot\{02\})\oplus\{0D\}\oplus\{09\}\oplus(\{0E\}\cdot\{03\})&=\{00\}
\end{aligned}
$$

对第一个等式，有{0E}·{02} = 00011100；而且{09}·{03}={09}⊕({09}·{02})= 00001001⊕

000010010 = 00011011。于是

$$\begin{aligned}\{0E\}\cdot\{02\} &= 00011100\\ \{0B\} &= 00001011\\ \{0D\} &= 00001101\\ \{09\}\cdot\{03\} &= \underline{00011011}\\ &\quad 00000001\end{aligned}$$

利用类似的方法可以验证其他的方程。

AES 文档描述了另一种描述列混淆变换的方式，即利用数学上的多项式来表示。在标准中，列混淆变换可以通过将状态的每列考虑为一个系数在 $GF(2^8)$ 上的四项多项式来定义。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：

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \tag{6.7}$$

附录 5A 论述了将状态的每列乘上 $a(x)$能写成式（6.3）中的矩阵乘法。相似地，能把式（6.5）中的变换的每列看成一个四项多项式且把该列乘上 $b(x)$。$b(x)$的定义如下：

$$b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} \tag{6.8}$$

显而易见 $b(x) = a^{-1}(x) \bmod (x^4+1)$。

基本原理　在式（6.3）中矩阵的系数是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文献[DAEM99]。

此外，列混淆变换的系数，即{01}、{02}、{03}是基于算法实现角度考虑的。正如上文所述，这些系数的乘法涉及至多一次移位和一次 XOR。逆向列混淆变换中的系数更加难以实现。然而加密被视为比解密更重要，原因如下：

（1）对于 CFB 和 OFB 密码模式（参见第 7 章中图 7.5 和图 7.6），仅使用加密算法。

（2）和任何其他的分组密码一样，AES 能用于构造消息验证码（参见第 13 章），这仅仅用到了加密过程。

6.3.4　轮密钥加变换

正向和逆向变换　在轮密钥加变换中，128 位的状态按位与 128 位的轮密钥 XOR。如图 6.5(b)所示，该操作可以看成是状态的一列中的四个字节与轮密钥的一个字进行列间的操作；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	D6

例子中第一个矩阵是状态，第二个矩阵是轮密钥。

逆向轮密钥加变换是和正向轮密钥加变换一样的，因为异或操作是其本身的逆。

基本原理　轮密钥加变换非常简单，却能影响状态中的每一位。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。

图 6.8 是描述单轮 AES 的另一种视角，强调各变换的机制和输入。

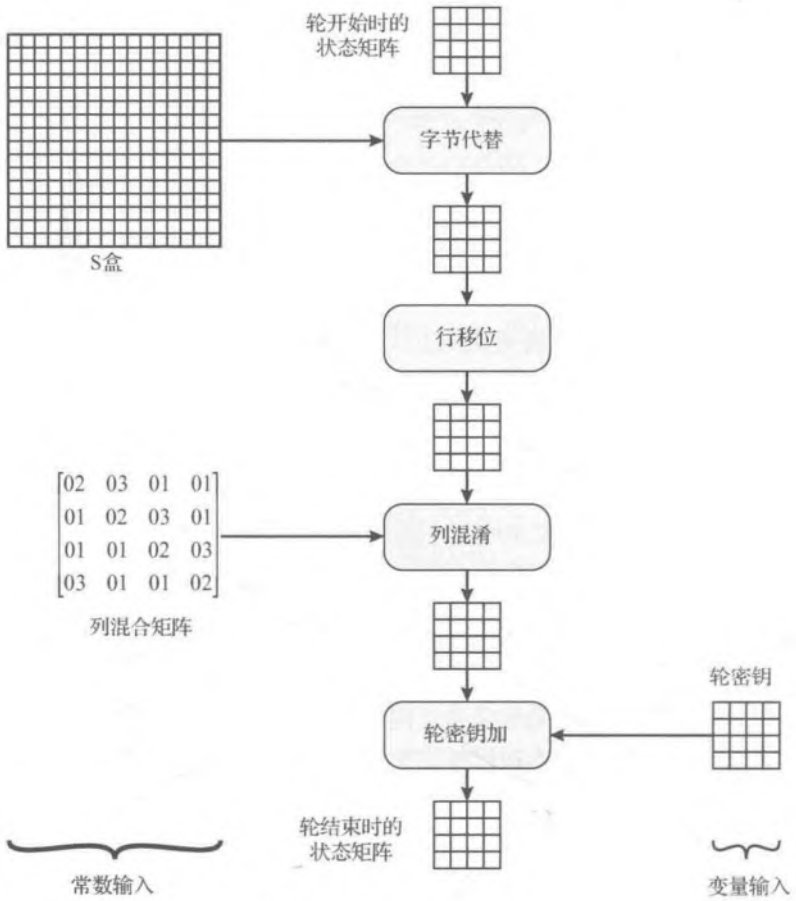


图 6.8 单轮 AES 的输入

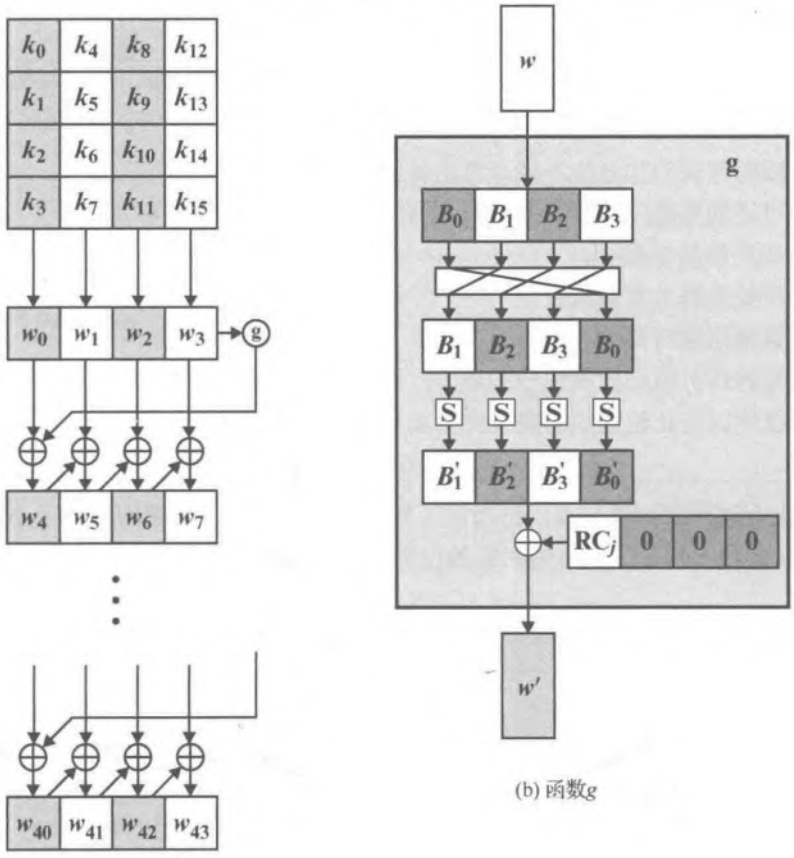
6.4 AES 的密钥扩展

6.4.1 密钥扩展算法

AES 密钥扩展算法的输入值是 4 个字（16 字节），输出值是一个 44 个字组成（176 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 4 个字。然后每次用 4 个字填充扩展密钥数组余下的部分。在扩展密钥数组中，每一个新增的字 $w[i]$的值依赖于 $w[i-1]$和 $w[i-4]$。在四种情形中，三个使用了异或。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 6.9 阐明了如何计算扩展密钥，其中使用符号 g 来表示这个复杂函数。函数 g 由下述的子功能组成：



(a)总体算法　　(b) 函数g

图 6.9　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$[B_0,B_1,B_2,B_3]$变换成$[B_1,B_2,B_3,B_0]$。

（2）字代替利用 S 盒对输入字中的每个字节进行字节代替[参见表 6.2(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。因此字与 Rcon 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][乘法是定义在域 GF(2^8)上的]。RC[j]的值按十六进制表示为

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 8 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 9 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	字循环后	字代替后	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

6.4.2 基本原理

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的轮密钥产生方式上的对称性或相似性。文献[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执行。
- 使用轮常量来消除对称性。
- 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许多位。
- 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密钥或轮密钥中少于 Nk 个连续字，那么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。

6.5 一个 AES 例子

现在我们研究一个例子并考虑它的含义。尽管不期待你手工来重做该例子，你还是会发现研究每一步间的十六进制模式将会带来很多的知识。

本例中，明文是十六进制表示的回文（顺读和倒读都一样）。明文、密钥和密文如下：

明文	0123456789abcdeffedcba9876543210
密钥	0f1571c947d9e8590cb7add6af7f6798
密文	ff0b844a0853bf7c6934ab4364148fb9

6.5.1 结果

表 6.3 展示了 16 字节密钥扩展为 10 轮的密钥。如前所述，这个过程是逐字进行的，每个四字节规模的字占用轮密钥字矩阵的一列。左手的列展示的是为每轮产生的四个轮密钥字。右手的列展示的是密钥扩展中用于产生辅助字的步骤。我们从密钥自身开始，其作为 0 轮的轮密钥。

接着，表 6.4 展示的是 AES 加密过程中状态的变换情况。第一列展示的是轮开始时状态的值。第一行里，状态仅仅是明文的重排矩阵。第二列，第三列和第四列分别展示的是该轮状态在经过字节代替，行移位以及列混淆变换后的值。第五列展示的是轮密钥。你可以验证这些轮密钥和表 6.3 里的值相等。第一列展示的是前轮列混淆后的状态和前轮的轮密钥进行逐位异或后的状态值。

6.5.2 雪崩效应

如果密钥或明文的小变化对密文的影响也是小变化，这有可能被用来有效减少明文（或密钥）的搜索空间。而我们期待的是雪崩效应，即对明文或密钥的小变化导致密文的大变化。

表 6.3　AES 例子的密钥扩展

密 钥 字	辅 助 函 数
w0=0f 15 71 c9	字循环 RotWord(w3)=7f 67 98 af=x1
w1=47 d9 e8 59	字代替 SubWord(x1)=d2 85 46 79=y1
w2=0c b7 ad d6	轮常数 Rcon(1)=01 00 00 00
w3=af 7f 67 98	y1⊕Rcon(1)=d3 85 46 79=z1
w4=w0⊕z1=dc 90 37 b0	字循环 RotWord(w7)=81 15 a7 38 =x2
w5=w4⊕w1=9b 49 df e9	字代替 SubWord(x4)=0c 59 5c 07=y2
w6=w5⊕w2=97 fe 72 3f	轮常数 Rcon(2)=02 00 00 00
w7=w6⊕w3=38 81 15 a7	y2⊕Rcon(2)=0e 59 5c 07=z2
w8=w4⊕z2=d2 c9 6b b7	字循环 RotWord(w11)=ff d3 c6 e6 =x3
w9=w8⊕w5=49 80 b4 5e	字代替 SubWord(x2)=16 66 b483=y3
w10=w9⊕w6=de 7e c6 61	轮常数 Rcon(3)=04 00 00 00
w11=w10⊕w7=e6 ff d3 c6	y3⊕Rcon(3)=12 66 b4 8e=z3
w12=w8⊕z3=c0 af df 39	字循环 RotWord(w15)=ae 7e c0 b1 =x4
w13=w12⊕w9=89 2f 6b 67	字代替 Subword(x3)=e4 f3 ba c8=y4
w14=w13⊕w10=57 51 ad 06	轮常数 Rcon(4)=08 00 00 00
w15=w14⊕w11=b1 ae 7e c0	y4⊕Rcon(4)=ec f3 ba c8=z4
w16=w12⊕z4=2c 5c 65 f1	字循环 RotWord(w19)=8c dd 50 43 =x5
w17=w16⊕w13=a5 73 0e 96	字代替 SubWord(x4)=64 c1 53 1a =y5
w18=w17⊕w14=f2 22 a3 90	轮常数 Rcon(5)=10 00 00 00
w19=w18⊕w15=43 8c dd 50	y5⊕Rcon(5)=74 c1 53 1a=z5
w20=w16⊕z5=58 9d 36 eb	字循环 RotWord(w23)= 40 46 bd 4c =x6
w21=w20⊕w17=fd ee 38 7d	字代替 SubWord(x5)=09 5a 7a 29=y6
w22=w21⊕w18=0f cc 9b ed	轮常数 Rcon(6)=20 00 00 00
w23=w22⊕w19=4c 40 46 bd	y6⊕Rcon(6)=29 5a 7a 29=z6
w24=w20⊕z6=71 c7 4c c2	字循环 RotWord(w27)= a5 a9 ef cf =x7
w25=w24⊕w21=8c 29 74 bf	字代替 SubWord(x6)=06 d3 bf 8a=y7
w26=w25⊕w22=83 e5 ef 52	轮常数 Rcon(7)=40 00 00 00
w27=w26⊕w23=cf a5 a9 ef	y7⊕Rcon(7)= 46 d3 bf 8a =z7
w28=w24⊕z7=37 14 93 48	字循环 RotWord(w31)= 7d a1 4a f7 =x8
w29=w28⊕w25=bb 3d e7 f7	字代替 SubWord(x7)=ff 32 d6 68=y8
w30=w29⊕w26=38 d8 08 a5	轮常数 Rcon(8)=80 00 00 00
w31=w30⊕w27=f7 7d a1 4a	y8⊕Rcon(8)= 7f 32 d6 68=z8
w32=w28⊕z8=48 26 45 20	字循环 RotWord(w35)= be 0b 38 3c =x9
w33=w32⊕w29=f3 1b a2 d7	字代替 SubWord(x8)=ae 2b 07 eb=y9
w34=w33⊕w30=cb c3 aa 72	轮常数 Rcon(9)=1b 00 00 00
w35=w34⊕w31=3c be 0b 38	y9⊕Rcon(9)= b5 2b 07 eb =z9
w36=w32⊕z9=fd 0d 42 cb	字循环 RotWord(w39)= 6b 41 56 f9 =x10
w37=w36⊕w33=0e 16 e0 1c	字代替 SubWord(x9)=7f 83 b1 99=y10
w38=w37⊕w34=c5 d5 4a 6e	轮常数 Rcon(10)=36 00 00 00
w39=w38⊕w35=f9 6b 41 56	y10⊕Rcon(10)=49 83 b1 99=z10
w40=w36⊕z10=b4 8e f3 52	
w41=w40⊕w37=ba 98 13 4e	
w42=w41⊕w38=7f 4d 59 20	
w43=w42⊕w39=86 26 18 76	

用表 6.4 的例子，表 6.5 展示的是当改变明文的第 8 位时的结果。表的第 2 列显示的是两个明

文在每轮结束时状态矩阵的值。注意到仅经过一轮后，状态向量里有 20 位发生了改变。两轮后，接近一半的位发生了改变。这种规模的差别一直扩散到后面的各轮里。一位的差别导致了密文大约一半的位置变了，这是很好的结果[①]。显然，如果几乎所有的位都改变了，这在逻辑上等价于几乎没有位发生改变。换一种说法，如果我们随机地选择两个明文，我们期望明文约有一半的位位置不同，密文也约有一半的位位置不同。

当明文相同，但加密的两个密钥在第 8 位不同时，表 6.6 显示了此时的状态矩阵的变化。也就是说，对于第二种情况，密钥是 0e1571c947d9e8590cb7add6af7f6798。同样，一轮就产生了很大的变化，后面各轮结束后的变化规模也大约是一半的位。因此，基于该例子，AES 展示了很强的雪崩效应。

在明文发生一位的变化或密钥发生一位变化的同等情况下，注意 AES 的这种雪崩效应比起 DES（参见表 4.2）来要强一些，DES 需要经过 3 轮后才达到大约一半位发生改变的效果。

表 6.4 AES 例子

轮开始	字节代替后	行移位后	列混淆后	轮密钥
01 89 fe 76				0f 47 0c af
23 ab dc 54				15 d9 b7 7f
45 cd ba 32				71 e8 ad 67
67 ef 98 10				c9 59 d6 98
0e ce f2 d9	ab 8b 89 35	ab 8b 89 35	b9 94 57 75	dc 9b 97 38
36 72 6b 2b	05 40 7f f1	40 7f f1 05	e4 8e 16 51	90 49 fe 81
34 25 17 55	18 3f f0 fc	f0 fc 18 3f	47 20 9a 3f	37 df 72 15
ae b6 4e 88	e4 4e 2f c4	c4 e4 4e 2f	c5 d6 f5 3b	b0 e9 3f a7
65 0f c0 4d	4d 76 ba e3	4d 76 ba e3	8e 22 db 12	d2 49 de e6
74 c7 e8 d0	92 c6 9b 70	c6 9b 70 92	b2 f2 dc 92	c9 80 7e ff
70 ff e8 2a	51 16 9b e5	9b e5 51 16	df 80 f7 c1	6b b4 c6 d3
75 3f ca 9c	9d 75 74 de	de 9d 75 74	2d c5 1e 52	b7 5e 61 c6
5c 6b 05 f4	4a 7f 6b bf	4a 7f 6b bf	b1 c1 0b cc	c0 89 57 b1
7b 72 a2 6d	21 40 3a 3c	40 3a 3c 21	ba f3 8b 07	af 2f 51 ae
b4 34 31 12	8d 18 c7 c9	c7 c9 8d 18	f9 1f 6a c3	df 6b ad 7e
9a 9b 7f 94	b8 14 d2 22	22 b8 14 d2	1d 19 24 5c	39 67 06 c0
71 48 5c 7d	a3 52 4a ff	a3 52 4a ff	d4 11 fe 0f	2c a5 f2 43
15 dc da a9	59 86 57 d3	86 57 d3 59	3b 44 06 73	5c 73 22 8c
26 74 c7 bd	f7 92 c6 7a	c6 7a f7 92	cb ab 62 37	65 0e a3 dd
24 7e 22 9c	36 f3 93 de	de 36 f3 93	19 b7 07 ec	f1 96 90 50
f8 b4 0c 4c	41 8d fe 29	41 8d fe 29	2a 47 c4 48	58 fd 0f 4c
67 37 24 ff	85 9a 36 16	9a 36 16 85	83 e8 18 ba	9d ee cc 40
ae a5 c1 ea	e4 06 78 87	78 87 e4 06	84 18 27 23	36 38 9b 46
e8 21 97 bc	9b fd 88 65	65 9b fd 88	eb 10 0a f3	eb 7d ed bd
72 ba cb 04	40 f4 1f f2	40 f4 1f f2	7b 05 42 4a	71 8c 83 cf
1e 06 d4 fa	72 6f 48 2d	6f 48 2d 72	1e d0 20 40	c7 29 e5 a5
b2 20 bc 65	37 b7 65 4d	65 4d 37 b7	94 83 18 52	4c 74 ef a9
00 6d e7 4e	63 3c 94 2f	2f 63 3c 94	94 c4 43 fb	c2 bf 52 ef
0a 89 c1 85	67 a7 78 97	67 a7 78 97	ec 1a c0 80	37 bb 38 f7
d9 f9 c5 e5	35 99 a6 d9	99 a6 d9 35	0c 50 53 c7	14 3d d8 7d
d8 f7 f7 fb	61 68 68 0f	68 0f 61 68	3b d7 00 ef	93 e7 08 a1
56 7b 11 14	b1 21 82 fa	fa b1 21 82	b7 22 72 e0	48 f7 a5 4a

① 原文有误，推断该如此——译者注。

（续表）

轮开始	字节代替后	行移位后	列混淆后	轮密钥
db a1 f8 77	b9 32 41 f5	b9 32 41 f5	b1 1a 44 17	48 f3 cb 3c
18 6d 8b ba	ad 3c 3d f4	3c 3d f4 ad	3d 2f ec b6	26 1b c3 be
a8 30 08 4e	c2 04 30 2f	30 2f c2 04	0a 6b 2f 42	45 a2 aa 0b
ff d5 d7 aa	16 03 0e ac	ac 16 03 0e	9f 68 f3 b1	20 d7 72 38
f9 e9 8f 2b	99 1e 73 f1	99 1e 73 f1	31 30 3a c2	fd 0e c5 f9
1b 34 2f 08	af 18 15 30	18 15 30 af	ac 71 8c c4	0d 16 d5 6b
4f c9 85 49	84 dd 97 3b	97 3b 84 dd	46 65 48 eb	42 e0 4a 41
bf bf 81 89	08 08 0c a7	a7 08 08 0c	6a 1c 31 62	cb 1c 6e 56
cc 3e ff 3b	4b b2 16 e2	4b b2 16 e2	4b 86 8a 36	b4 8e f3 52
a1 67 59 af	32 85 cb 79	85 cb 79 32	b1 cb 27 5a	ba 98 13 4e
04 85 02 aa	f2 97 77 ac	77 ac f2 97	fb f2 f2 af	7f 4d 59 20
a1 00 5f 34	32 63 cf 18	18 32 63 cf	cc 5a 5b cf	86 26 18 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

表 6.5　AES 的雪崩效应：明文的改变

轮数		不同的位数目
	0123456789abcdeffedbca9876543210 0023456789abcdeffedbca9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36ald891acl8la	64
7	dbl8a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

表 6.6 AES 的雪崩效应：密钥的改变

轮		不同的位数
	0123456789abcdeffedbca9876543210 0123456789abcdeffedbca9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8c8ea4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

6.6 AES 的实现

6.6.1 等价的逆算法

如上所述，AES 的解密算法和加密算法不同（参见图 6.3）。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。其缺点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。如图 6.3 所示，在加密过程中，其轮结构为字节代替，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，逆向字节代替，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，后两个阶段也需要交换。

交换逆向行移位和逆向字节代替 逆向移行影响在状态中字节的顺序，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。逆向字节代替影响状态中字节的内容，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。如对一个给定的状态 S_i

$$\text{逆向移行}[\text{逆向字节代替}(S_i)] = \text{逆向字节代替}[\text{逆向移行}(S_i)]$$

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状态中字节的顺序。如果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状态的一列进行操作。这两个操作对列输入是线性的。即对给定的状态 S_i 和给定的轮密钥 w_j

$$逆向列混淆(S_i \oplus w_j) = [逆向列混淆(S_i)] \oplus [逆向列混淆(w_j)]$$

为了说明这一点，假定状态 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)。然后有

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

我们来论证第一列。有

$$\begin{aligned} &[\{0E\} \cdot (y_0 \oplus k_0)] \oplus [\{0B\} \cdot (y_1 \oplus k_1)] \oplus [\{0D\} \cdot (y_2 \oplus k_2)] \oplus [\{09\} \cdot (y_3 \oplus k_3)] \\ &= [\{0E\} \cdot y_0] \oplus [\{0B\} \cdot y_1] \oplus [\{0D\} \cdot y_2] \oplus [\{09\} \cdot y_3] \oplus \\ &\quad [\{0E\} \cdot k_0] \oplus [\{0B\} \cdot k_1] \oplus [\{0D\} \cdot k_2] \oplus [\{09\} \cdot k_3] \end{aligned}$$

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，可以交换轮密钥加和逆向列混淆。注意，无须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（轮 10），因为这两个轮密钥加变换不能与逆向列混淆交换来产生等价的解密算法。

图 6.10 描述了等价的解密算法。

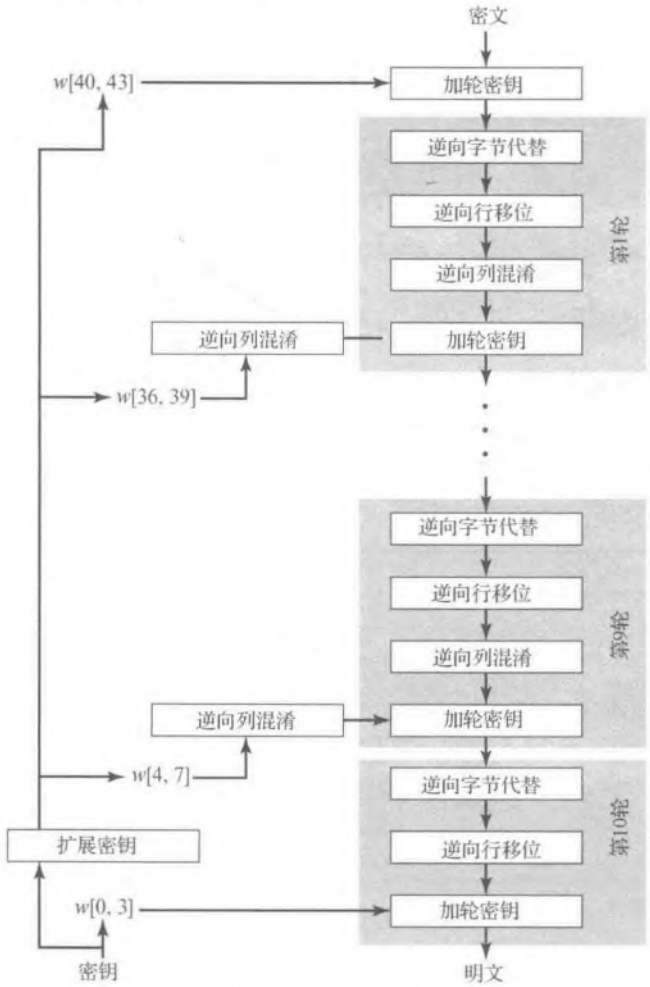


图 6.10　等价逆密码

6.6.2 实现方面

Rijndael 的提案[DAEM99]针对如何用像智能卡等 8 位处理器，像个人电脑等 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。轮密钥加是按字节异或操作。行移位是简单的移字节操作。字节代替是在字节级别上进行操作的，而且只要求一个 256 字节的表。

列混淆变换要求是在域 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。方程集合式(6.4)是说明在单列上进行列混淆变换的等式。利用性质{03}·x = ({02}·x)⊕x，能用如下的方式重写方程集合［参见式（6.4）］：

$$\begin{aligned}
\mathrm{Tmp} &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus \mathrm{Tmp} \oplus [2 \cdot (s_{0,j} \oplus s_{1,j})] \\
s'_{1,j} &= s_{1,j} \oplus \mathrm{Tmp} \oplus [2 \cdot (s_{1,j} \oplus s_{2,j})] \\
s'_{2,j} &= s_{2,j} \oplus \mathrm{Tmp} \oplus [2 \cdot (s_{2,j} \oplus s_{3,j})] \\
s'_{3j} &= s_{3,j} \oplus \mathrm{Tmp} \oplus [2 \cdot (s_{3,j} \oplus s_{0,j})]
\end{aligned} \tag{6.9}$$

方程集合式（6.9）能通过展开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 4.4 节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}·i。集合式（6.9）被描述为

$$\begin{aligned}
\mathrm{Tmp} &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus \mathrm{Tmp} \oplus X2[(s_{0,j} \oplus s_{1,j})] \\
s'_{1,j} &= s_{1,j} \oplus \mathrm{Tmp} \oplus X2[(s_{1,j} \oplus s_{2,j})] \\
s'_{2,j} &= s_{2,j} \oplus \mathrm{Tmp} \oplus X2[(s_{2,j} \oplus s_{3,j})] \\
s'_{3j} &= s_{3,j} \oplus \mathrm{Tmp} \oplus X2[(s_{3,j} \oplus s_{0,j})]
\end{aligned}$$

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，首先利用代数形式定义一轮的四个变换。假如利用 $a_{i,j}$ 表示状态矩阵，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：

字节代替	$b_{i,j} = \boldsymbol{S}[a_{i,j}]$
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

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}[a_{0,j}] \\ \boldsymbol{S}[a_{1,j-1}] \\ \boldsymbol{S}[a_{2,j-2}] \\ \boldsymbol{S}[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

$$= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot \boldsymbol{S}[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot \boldsymbol{S}[a_{1,j-1}] \right) \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot \boldsymbol{S}[a_{2,j-2}] \right) \oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot \boldsymbol{S}[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

在第二个方程中，我们用向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：

$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[x] \right)$	$T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot \boldsymbol{S}[x] \right)$	$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot \boldsymbol{S}[x] \right)$	$T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot \boldsymbol{S}[x] \right)$

因此，把每个表看成是输入一个字节值，产生一个列向量(一个 32 位的字)，该向量是对应该字节值的 S 盒输入的函数。这些表能被事先计算。

我们用如下的方式定义轮函数对一列的操作：

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{1,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

结果，基于上述等式的实现仅需要四张表，每轮每列的四次异或以及存储这些表所需的 4 KB 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 作为高级加密标准最重要因素之一。

6.7　关键术语、思考题和习题

关键术语

高级加密标准（AES）	有限域	美国国家标准技术研究所（NIST）	
雪崩效应	不可约	多项式	
Rjindael 算法	域	密钥扩展	S 盒

思考题

6.1　NIST 评估 AES 候选算法的最初标准是什么？

6.2　NIST 评估 AES 候选算法的最终标准是什么？

6.3　Rijndael 和 AES 有何不同？

6.4 使用状态数组的目的是什么？

6.5 如何构造 S 盒？

6.6 简述什么是字节代替。

6.7 简述什么是行移位变换。

6.8 行移位变换影响了状态中的多少字节？

6.9 简述什么是列混淆。

6.10 简述什么是轮密钥加变换。

6.11 简述密钥扩展算法。

6.12 字节代替和字代替有何不同？

6.13 行移位变换和字循环函数有何不同？

6.14 AES 解密算法和其等价逆算法之间有何不同？

习题

6.1 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。

6.2 （a）在 GF(2^8)上{01}的逆是什么？

（b）验证{01} 在 S 盒中的项。

6.3 当 128 位的密钥是全 0 时，给出密钥扩展数组的前 8 个字。

6.4 若明文是{000102030405060708090A0B0C0D0E0F}，密钥是{01010101010101010101 10101010101}，

（a）用 4×4 的矩阵来描述状态的最初内容。

（b）给出初始化轮密钥加后状态的值。

（c）给出字节代替后状态的值。

（d）给出行移位后状态的值。

（e）给出列混淆后状态的值。

6.5 验证式（6.11）。即验证 $x^j \bmod (x^4+1) = x^{j \bmod 4}$。

6.6 比较 AES 和 DES。对如下所述 DES 中的元素，指出 AES 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 函数的输出与分组左边的部分相异或。

（c）f 函数。

（d）置换 P。

（e）交换一个分组的两半部分。

6.7 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。

6.8 在关于实现这一节中，提出了仅用一个代数方程描述加密算法一个典型轮的四个阶段。请给出与第 10 轮等价的方程。

6.9 计算由输入字节序列“67 89 AB CD”经过列混淆后的输出结果。并对输出结果使用逆向列混淆变换来验证计算结果。将输入的第一个字节从“67”改为“77”，对新的输入重新执行列混淆变换，并判断输出有多少位发生了变化。注意：你可以通过手工计算或者编程来执行上述运算。如果是编程，

请独立编写所有代码；不要使用任何库或者公开的源代码。

6.10　使用密钥 1010 0111 0011 1011 加密表示为 ASCII 码的明文“ok”，即 0110 1111 0110 1011。S-AES 的设计者得出密文是 0000 0111 0011 1000，你的呢？

6.11　如下矩阵是 GF(2^4)上的矩阵，说明该矩阵是 S-AES 的列混淆使用的矩阵的逆。

$$\begin{pmatrix} x^3+1 & x \\ x & x^3+1 \end{pmatrix}$$

6.12　对于 S-AES 算法用密钥 1010 0111 0011 1011 解密密文 0000 0111 0011 1000。得出的明文应该与习题 6.10 所给明文相同。注意 S 盒的逆运算可以通过逆向查表来实现，而列混淆的逆运算可以通过习题 6.11 给出的矩阵来实现。

6.13　证明式（6.9）和式（6.4）等价。

编程习题

6.14　制作使用 S-AES 实现加密和解密的软件。测试数据：使用密钥 1010 0111 0011 1011 加密二进制明文 0110 1111 0110 1011，得出二进制密文 0000 0111 0011 1000。解密过程同上。

6.15　实现对于 1 轮 S-AES 的差分攻击。

附录 6A　系数在 GF(2^8)中的多项式

在 5.5 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是定义在域 Z_p 上的。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 5.4 节所指出的，如果把 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。于是有：

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{6.10}$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \tag{6.11}$$

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 5.6 节，GF(2^8)中的每个元素可以表示为系数为{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)中的每个元素都能被看成一个 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。对本节所定义的集合，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被看成一个四字节的字，该字中的每一个字节是 GF(2^8)内的元素，其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

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \tag{6.12}$$

其中

$c_0 = a_0 \cdot b_0$

$c_1 = (a_1 \cdot b_0) \oplus (a_0 \cdot b_1)$

$c_2 = (a_0 \cdot b_2) \oplus (a_1 \cdot b_1) \oplus (a_2 \cdot b_0)$

$c_3 = (a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3)$

$c_4 = (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$

$c_5 = (a_3 \cdot b_2) \oplus (a_2 \cdot b_3)$

$c_6 = a_3 \cdot b_3$

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际技巧基于如下观察：

$$x^j \bmod (x^4+1) = x^{j \bmod 4} \tag{6.13}$$

如果联立方程式（6.12）和方程式（6.13），可得到

$$\begin{aligned} d(x) = c(x) \bmod (x^4+1) &= [c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0] \bmod (x^4+1) \\ &= c_3x^3 + (c_2 \oplus c_6)x^2 + (c_1 \oplus c_5)x + (c_0 \oplus c_4) \end{aligned}$$

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：

$$d_0 = (a_0 \cdot b_0) \oplus (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$$

$$d_1 = (a_1 \cdot b_0) \oplus (a_0 \cdot b_1) \oplus (a_3 \cdot b_2) \oplus (a_2 \cdot b_3)$$

$$d_2 = (a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3)$$

$$d_3 = (a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3)$$

其矩阵表达方式为

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{6.14}$$

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。第一个是式（6.3）所示的矩阵乘法，即

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s_{3,3} \end{bmatrix}$$

第二种方式是将状态中的每列看成一个系数在 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

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

根据式（6.10），有 $a_3 = \{03\}$；$a_2 = \{01\}$；$a_1 = \{01\}$；$a_0 = \{02\}$。对状态中的第 j 列，我们有列多项式 $\mathrm{col}_j(x) = s_{3,j}x^3 + s_{2,j}x^2 + s_{1,j}x + s_{0,j}$。代入式（6.14），能将 $d(x) = a(x) \times \mathrm{col}_j(x)$表示为

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

这与式（6.3）等价。

乘以 x

考虑环中一个多项式乘上 x 的情况：$c(x) = x \oplus b(x)$。我们有

$$\begin{aligned} c(x) = x \oplus b(x) &= [x \times (b_3x^3 + b_2x^2 + b_1x + b_0)] \bmod (x^4+1) \\ &= (b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod (x^4+1) \\ &= b_2x^3 + b_1x^2 + b_0x + b_3 \end{aligned}$$

因此，多项式乘上 x 就相当于将这个多项式所对应的由 4 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 4 字节的列向量，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 7 章　分组加密的工作模式

学 习 目 标

通过本章的学习应具备以下能力：

- 分析多重加密的安全性
- 理解中间相遇攻击。三重 DES 在三个阶段使用 DES 算法，共用到两组或者三组密钥
- 比较 ECB、CBC、CFB、OFB 和其他工作模式
- 简要介绍 XTS-AES 工作模式

本章继续讨论对称密码。首先讨论多重加密，特别是目前广泛使用的一种多重加密方案：三重 DES。

接下来将介绍分组密码的工作模式。我们发现使用分组密码加密明文有许多种不同的方法，每一种方法都有各自的优点和特定应用环境。

7.1　多重加密与三重 DES

DES 在穷举攻击之下相对比较脆弱，因此很多人在想办法用某种算法替代它。一种方案是设计全新的算法，如 AES。还有一种方案，能够保护已有软件和硬件的投资，那就是用 DES 进行多次加密，且使用多个密钥。我们以这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。

7.1.1　双重 DES

多次加密的最简单形式是进行两次加密，使用两个密钥［参见图 7.1(a)］。给定明文 P 及密钥 K_1，K_2，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 56×2=112 位，密码强度增加了。不过我们还是要对它进行仔细分析。

约化为单次加密　考虑下述说法：对所有的 56 位密钥，给定 K_1 和 K_2，可能存在密钥 K_3 使得

$$\mathrm{E}(K_2, \mathrm{E}(K_1, P)) = \mathrm{E}(K_3, P) \tag{7.1}$$

若该说法成立，两层加密，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密钥进行一次 DES 加密的效果。

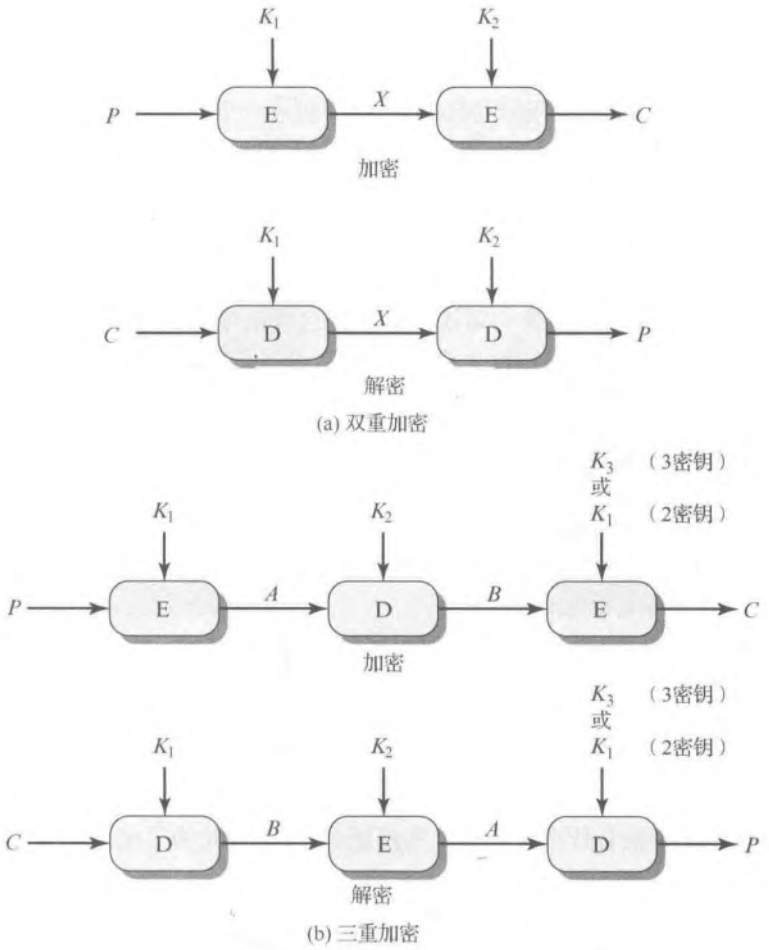


图 7.1　多重加密

表面看来，式（7.1）不太可能成立。DES 加密是 64 位分组之间的映射，实际上映射可以看成是一个置换。这就是说，对 2^{64} 个可能的明文分组，DES 用某特定密钥加密后都唯一对应一个 64 位的密文分组。否则，即是两个输入块映射为一个输出块，则解密以恢复原始的明文就不可能了。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

$$(2^{64})! = 10^{347,380,000,000,000,000,000} > (10^{10^{20}})$$

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 所对应的映射不能为单 DES 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，参见文献[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，这种方法不依赖于 DES 的任何特殊性质，对所有分组密码都有效。

该算法称为中间相遇攻击，文献[DIFF77]首次对它进行了描述。它基于如下观察：假设

$$C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$$

则有［参见图 7.1(a)］

$$X = \mathrm{E}(K_1, P) = \mathrm{D}(K2, C)$$

给定明密文对（P, C），攻击如下展开：首先，将 P 按所有可能的密钥 K_1 加密，得到的 2^{56} 个结果按 X 的值排序放在一个表内。然后将 C 用所有可能的密钥 K_2 解密，每解密一次就将解密结果与表中的值比较，如果有相等的，就用刚才测试的两个密钥对一个新的明密文对进行验证。如果两个密钥产生了正确的密文，就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 64 位的明密文对（P, C），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就可以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。其付出是 2^{56} 数量级的，比攻击单 DES 所需的 2^{55} 数量级多不了多少。

7.1.2 使用两个密钥的三重 DES

对付中间相遇攻击的一个明显办法是使用三个不同的密钥进行三次加密。这样，已知明文攻击的代价将升为 2^{112} 数量级，无论现在还是遥远的将来这都超出了实际可行性。然而，它需要长为 56×3=168 位的密钥，这多少有些笨拙。

Tuchman 建议仅使用两个密钥[TUCH79]进行三次加密［参见图 7.1(b)］。具体运算过程是加密-解密-加密（EDE），写成式子是

$$C = \mathrm{E}(K_1, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$
$$P = \mathrm{D}(K_1, \mathrm{E}(K_2, \mathrm{D}(K_1, C)))$$

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

$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}(K_1, P)$$
$$P = \mathrm{D}(K_1, \mathrm{E}(K_1, \mathrm{D}(K_1, C))) = \mathrm{D}(K_1, C)$$

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 ANSI X9.17 和 ISO 8732[①]。

当前，还没有对 3DES 的可行攻击方法。Copper Smith[COPP94]分析后认为对 3DES 的穷举攻击的代价是 $2^{112} \approx (5\times10^{33})$数据级的。且估计用差分密码分析的代价是按指数增长的，与单 DES 比较超过 10^{52}。

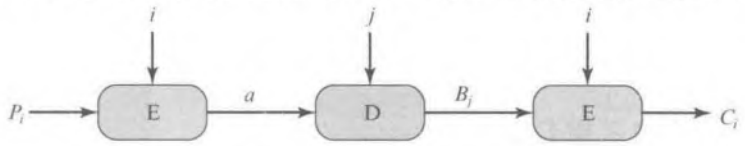
我们不妨看看几种对 3DES 的攻击方法，虽然都不实际，也许对以后产生好的攻击方法有所裨益。

第一个好的攻击建议来自 Merkle 和 Hellman[MERK81]。即挨个查看可能的明文，看哪一个明文的第一个中间值［参见图 7.1(b)］$A = 0$，然后再使用中间相遇攻击来得到两个密钥。这种攻击方法的代价是 2^{56} 数量级的，但是它需要 2^{56} 个选择明密文对，这是不可能的。

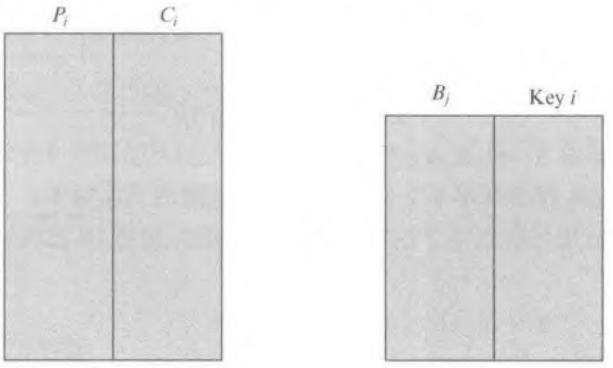
文献[VANO90]中概述了一种已知明文攻击，相对于选择明文攻击，这种方法有些进步，但付出的代价也增多了。攻击是基于观察到：若已知 A 和 C［参见图 7.1(b)］，那问题相当于对双重 DES 的攻击。当然，只要不知道密钥，即使知道了 P 和 C，攻击者还是得不到 A。然而，攻击者可选择 A 的一个可能值，再试着找出一个可产生 A 的已知（P，C）对。具体的攻击过程如下所述。

① 美国国家标准(ANSI)：金融机构密钥管理(批发)。从名字看，X9.17 显得并不出名，然而这个标准定义的大量技术已经被其他标准和应用采用，本书中我们将陆续看到。

（1）获取 n 个（P，C）对，将这些已知（P，C）对按 P 排序放在表（表 1）中［参见图 7.2(b)］。



(a) 用候选密钥对进行双密钥的三重加密



(b) n个已知（表1）明密文对的表　　(c) 中间值及（表2）候选密钥表

图 7.2　对 3DES 的已知明文攻击

（2）为 A 随意选择值 a，创建第 2 张表［参见图 7.2(c)］，表的项按下列方式定义。对每个可能的密钥 $K_1 = i$，计算可产生 a 的明文值 P_i

$$P_i = \mathrm{D}(i, a)$$

对所有能与图 7.2(b)中表 1 中的项匹配的 P_i，在表 2 中创建一个项，其中有 K_1 的值，以及由（P, C）对产生的 B 的值，即

$$B = \mathrm{D}(i, C)$$

最后将图 7.2(c)中的表 2 按 B 的值排序。

（3）现在表 2 中有了一批 K_1 的候选值，现在需要搜索 K_2。对任意可能的 2^{56} 个 K_2=j，对选定的 a 计算第 2 个中间值

$$B_j = \mathrm{D}(j, a)$$

每做一步都在表 2 中查找 B_j，若有相等者，则表 2 里的相应密钥 i 和 j 的值就是未知密钥（K_1，K_2）的可能值。因为我们找到一个密钥对（K_1, K_2）=（i, j），可以生成已知的（P，C）对［参见图 7.2(a)］。

（4）对这样的密钥对（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$(N+1)/(n+1)$次才可能摸一个红球，其中球不能重新放回。所以在 n 非常大的时候。尝试的 a 的个数为

$$\frac{2^{64}+1}{n+1} \approx \frac{2^{64}}{n}$$

则攻击的期望运算时间是以下数量级的：

$$(2^{56})\frac{2^{64}}{n} = 2^{120-\log_2 n}$$

7.1.3 使用三个密钥的三重 DES

尽管上述攻击方法并不实用，但是所有使用双密钥的三重 DES 算法的人还是感觉有点悬。因此，很多人觉得采用三个密钥的三重 DES 算法才是最好方案（如文献[KALI96a]）。三密钥的三重 DES 的密钥长度为 168 位，定义为

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

要想和单 DES 兼容，只需设 $K_3=K_2$ 或 $K_1=K_2$ 就可以了。人们可能期望 3TDEA 会提供 56×3=168 位的密钥强度，然而，3TDEA 将会耗尽 112 位的密钥，从而降低工作效率。

有些基于 Internet 的应用已经采纳了这种三密钥的 3DES，如第 19 章要讨论的 PGP 和 S/MIME。

7.2 电码本模式

分组密钥的输入为具有 b 位固定长度的明文分组和密钥，输出为 b 位的密文。明文长度若大于 b 位，则可简单将其分成 b 位一组的块。每次使用相同的密钥对多个分组加密，则会引发许多安全问题。为了将分组密码应用于各种各样的实际应用，NIST（SP 800-38A）定义了五种“工作模式”。从本质上讲，工作模式是一项增强密码算法或者使算法适应具体应用的技术，如将分组密码应用于数据块组成的序列或者数据流。这五种模式实际上覆盖了大量使用分组密码的应用。这些模式可用于包括三重 DES 和高级加密标准（AES）在内的任何分组密码。表 7.1 对这些模式做了一个总结，在余下的章节中将会对它们做简单的描述。

最简单的模式是电码本（ECB）模式，它一次处理一组明文分块，每次使用相同的密钥加密（参见图 7.3）。使用电码本这个词是因为对于给定的密钥，任何 b 位的明文组只有唯一密文与之对应，所以，可以想象存在一个很厚的密码本，根据任意 b 位明文都可以查到相应的密文。

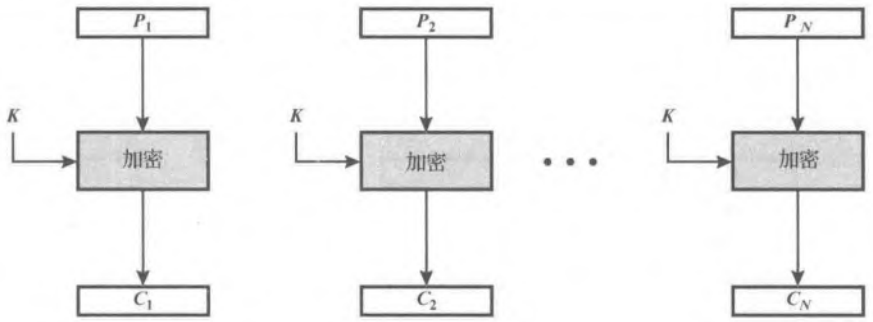
明文若长于 b 位，则可简单将其分成 b 位一组的块，有必要则可对最后一块进行填充。解密也是一次执行一块，且使用相同的密钥。图 7.3 中的明文（必要的时候进行填充）由一串 b 位的块组成，记为 P_1，P_2，…，P_N，相应的密文分组依次是 C_1，C_2，…，C_N。我们可以如下定义 ECB 模式。

ECB	$C_j = \mathrm{E}(K, P_j)$ $j=1,\cdots,N$	$P_j = \mathrm{D}(K, C_j)$ $j=1,\cdots,N$

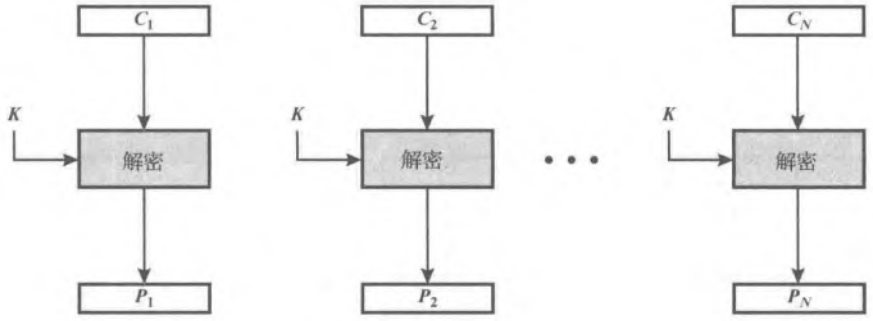
表 7.1 分组密码的工作模式

模　式	描　述	典型应用
电码本（ECB）	用相同的密钥分别对明文分组独立加密	● 单个数据的安全传输（如一个加密密钥）
密文分组链接（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	● 面向分组的通用传输 ● 认证
密文反馈（CFB）	一次处理 s 位，上一块密文作为加密算法的输入，产生的伪随机数输出与明文异或作为下一单元的密文	● 面向数据流的通用传输 ● 认证
输出反馈（OFB）	与 CFB 类似，只是加密算法的输入是上一次加密的输出，且使用整个分组	● 噪声信道上的数据流的传输（如卫星通信）
计数器（CTR）	每个明文分组都与一个经过加密的计数器相异或。对每个后续分组计数器递增	● 面向分组的通用传输 ● 用于高速需求

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



(a) 加密



(b) 解密

图 7.3　电码本(ECB)模式

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密文分组。

对于很长的消息，ECB 模型可能不安全。如果消息是非结构化的，密码分析者可能利用这些规律性特征来破译。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有大量已知明密文对以开展攻击。若消息有重复的成分，且重复的周期正好是 b 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块的方法进行攻击。

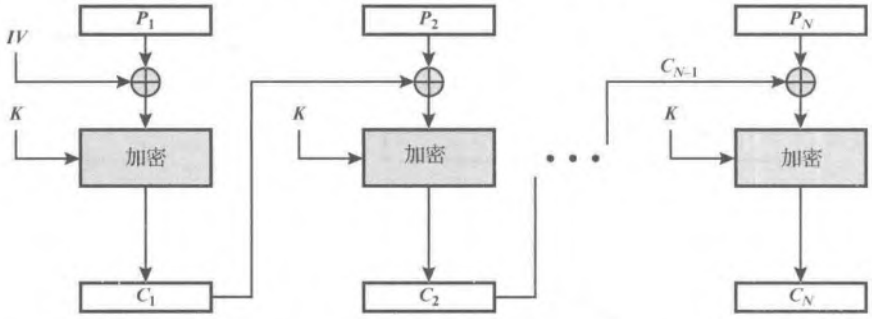
现在将考虑更复杂的工作模式，文献[KNUD00]列出了下面一些标准和性质，以便对那些优于 ECB 的分组密码工作模式进行评估和实施：

- **总体比较**　与 ECB 模式加解密相比较，加密和解密需要额外的操作。
- **错误恢复**　第 i 个密文分组的错误会只被模式同步后的一些明文分组继承。
- **错误传播**　第 i 个密文分组的错误会被第 i 组及其后所有明文分组继承，这意味着一位错误发生在密文分组传输的过程而不是明文分组加密的计算错误。
- **扩散**　熵值较小的密文分组不应该被映射到密文分组，粗略来说，熵值较小的密文分组等同于可预测性或者缺乏随机性（参见附录 F）。
- **安全性**　密文分组是否会泄露关于明文分组的信息。

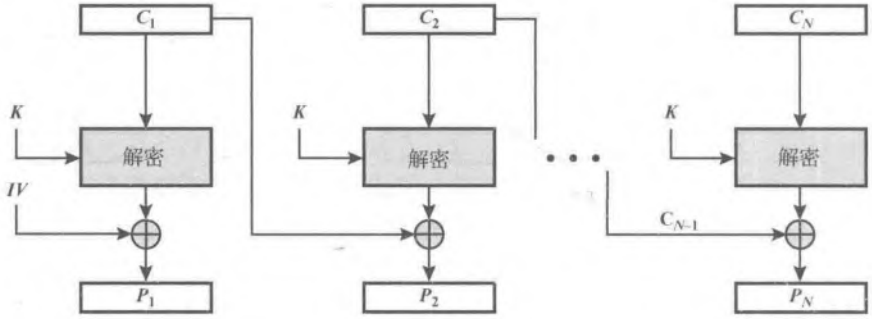
7.3　密文分组链接模式

为了克服 ECB 的这些弱点，需要将重复的明文分组加密成不同的密文分组。满足这一要求的简单方法是用 CBC 模式（参见图 7.4）。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与本明文组没有固定的关系。因此，若有重复的明文组，加密后并看不出来了。和 ECB 方式类似，CBC 方式也要求如果最后的分组不是完整的分组，则需要填充至 b 位的满分组。



(a) 加密



(b) 解密

图 7.4 密文分组链接（CBC）模式

解密时，每个密文组分别进行解密，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

$$\mathrm{D}(K, C_j) = \mathrm{D}(K, \mathrm{E}(K, [C_{j-1} \oplus P_j]))$$
$$\mathrm{D}(K, C_j) = C_{j-1} \oplus P_j$$
$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

第一块明文可以和一个初始向量（IV）异或后再加密，以产生第一个明文分组。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 是和密文具有相同的长度的数据分组。CBC 模式可以如下定义：

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])$ $j=2,\cdots,N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}$ $j=2,\cdots,N$

IV 必须为收发双方共享，但第三方不能预测。特别地，对于任意的给定的明文，在 IV 产生之前，不能预测和本明文关联的 IV。为了最大程度的安全，IV 不能不经授权而修改。对 IV 先用 ECB

加密后再发送的方式可以实现这一要求。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，接着攻击者就能够把明文的第一个分组的某些位取反了。为了弄明白这一点，请看

$$C_1 = \mathrm{E}(K, [\mathrm{IV} \oplus P_1])$$

$$P_1 = \mathrm{IV} \oplus \mathrm{D}(K, C_1)$$

用 $X[i]$表示 b 位 X 的第 i 位，则

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

使用 XOR 的性质，我们将上面重写为

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

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，则接收者收到的 P_1 相应也就改变了。

还有其他利用 IV 的一些攻击方法，参见文献[VOYD83]。

只要 IV 是不可预测的，那么具体选什么 IV 是不重要的。SP 800-3A 推荐了两种方法：第一个方法是用加密函数加密一个时变值[①]，所用密钥和明文加密所用密钥相同。这个时变值对每次加密运算来说必须唯一。例如，时变值可以是一个计数器，一个时间戳或者消息数目。第二个方法是用随机数发生器产生一个随机数分组。

总之，CBC 的链接机制使得它适合于加密长度大于 b 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 12 章描述。

7.4　密文反馈模式

对于 AES，DES 或任何的分组密码，加密是对一个 b 位的分组进行加密。对于 DES 的例子，$b = 64$，而在 AES 时，b=128。但是利用本节讨论的密文反馈模式（Cipher Feedback，CFB）和下面要讨论的输出反馈模式（OFB）及计数器（CTR）模式，亦可将分组密码当成流密码使用。流密码不需要将明文填充到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。所以要发送的是 8 位的字符，加密时也是用 8 位的。如果多于 8 位，传输能力就浪费了。

图 7.5 描述了 CFB 模式。假设传输单元是 s 位，s 通常为 8。同使用 CBC 模式一样，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，明文被分成 s 位的片段而不是 b 位的单元。

首先来考虑加密。加密函数的输入是 b 位的移位寄存器，它的值为初始向量 IV。加密函数输出最左边的 s 位与明文的第一个分段 P_1 异或得到密文的第一个单元 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。

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 $\mathrm{MSB}_s(X)$表示 X 的最左边 s 位。则

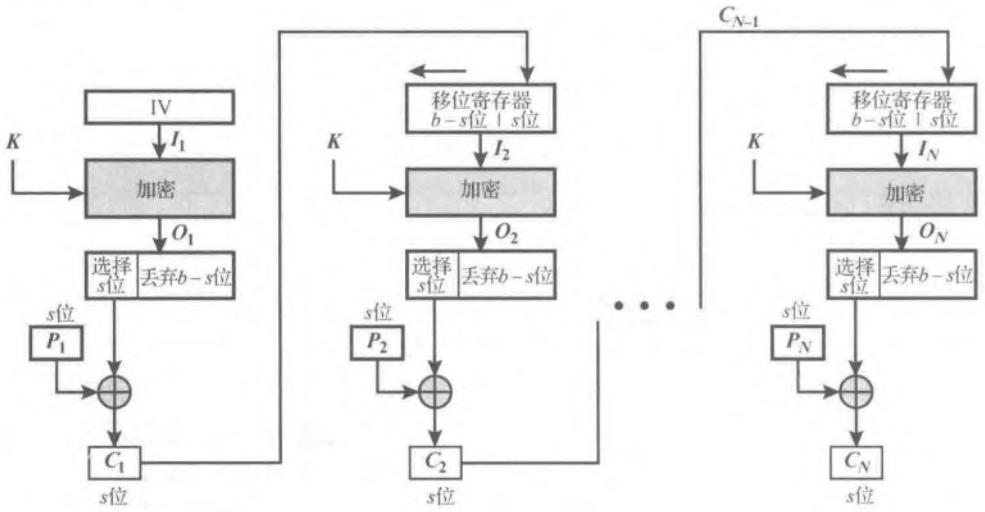
$$C_1 = P_1 \oplus \mathrm{MSB}_s[\mathrm{E}(K, \mathrm{IV})]$$

① NIST SP 800-90（使用决定性随机位发生器的随机数发生器建议）如下定义时变值：一个随时间变化的值，具有可忽略的重复概率，如每次使用都重新生成的随机值，时间戳，序列号或这些东西的某种组合。

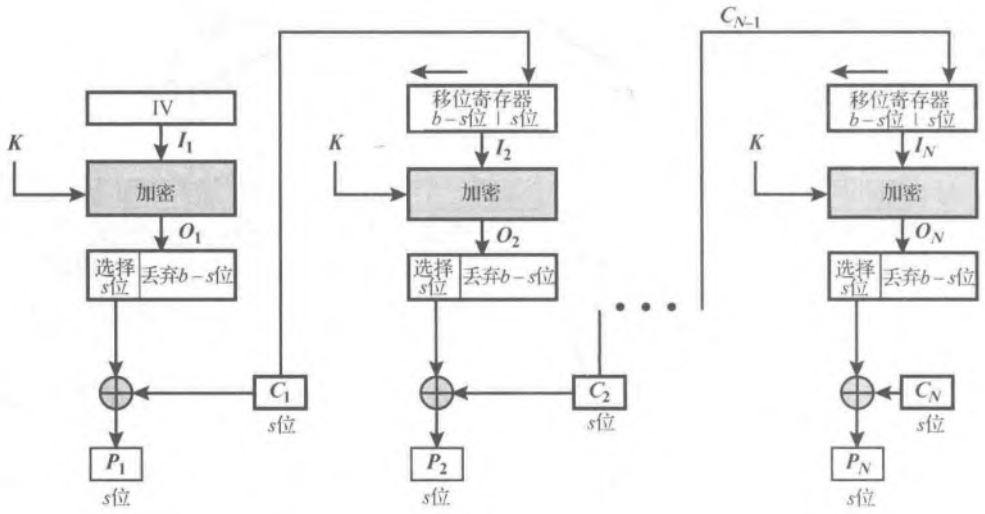
则

$$P_1 = C_1 \oplus \mathrm{MSB}_s\,[\mathrm{E}(K,\mathrm{IV})]$$

对后续单元亦同理可得。



(a) 加密



(b) 解密

图 7.5 s 位密文反馈（CFB）链接

CFB 模式可以如下定义：

CFB	I_1=IV		I_1 = IV	
	I_j=LSB$_{b-s}$(I_{j-1})||C_{j-1}	j=2,…,N	I_j = LSB$_{b-s}$(I_{j-1})||C_{j-1}	j =2,…,N
	O_j=E(K,I_j)	j=1,…,N	O_j = E(K,I_j)	j =1,…,N
	C_j=P_j ⊕ MSB$_s$(O_j)	j=1,…,N	P_j = C_j ⊕ MSB$_s$(O_j)	j =1,…,N

尽管 CFB 可以被看成是流密码，但是它和流密码的典型构造并不一致。典型的流密码输入某个初始值和密钥，输出位流，这个位流再和明文位进行异或运算（参见图 4.1）。而 CFB 模式里，与明文异或的位流是与明文相关的。

在 CFB 加密中，与 CBC 类似，每个密码函数（除了第一个）的输入块都依赖之前密码函数的结果输入到前端密码函数的数据块与前端密码函数以前的输出相关，因此，多重密码操作不能并行实现。在 CFB 解密中，如果输入块是首次构建于 IV 和密文，那么必需的密码操作是可以并行执行的。

7.5　输出反馈模式

输出反馈模式（Output Feedback，OFB）的结构和 CFB 很相似。如图 7.6 所示，它用加密函数的输出填充移位寄存器，而 CFB 是用密文单元来填充移位寄存器。其他不同是，OFB 模式对整个明文和密文分组进行运算，而不是仅对 s 位的子集运算。加密可表示如下：

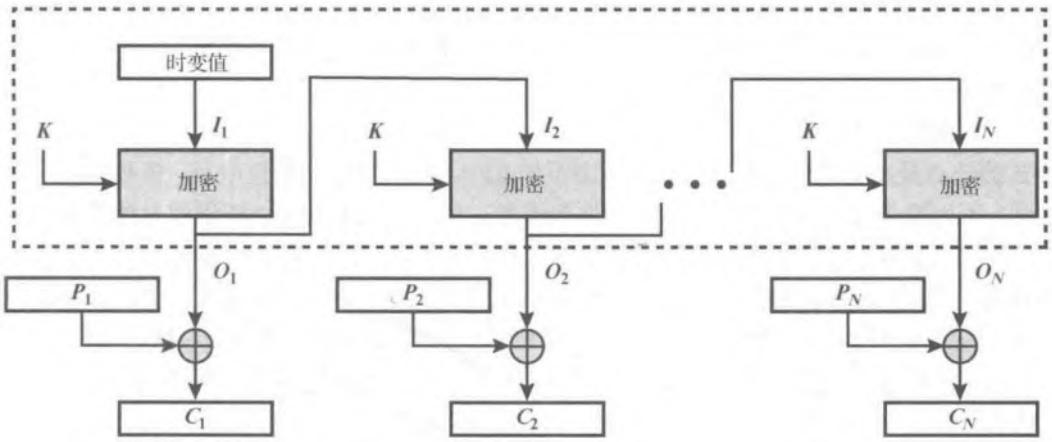
$$C_j = P_j \oplus \mathrm{E}(K, O_{j-1})$$

其中

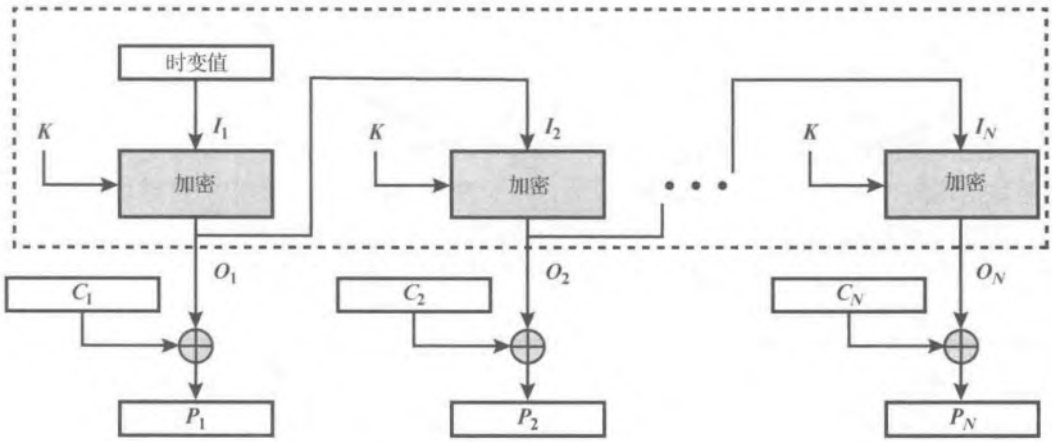
$$O_{j-1} = \mathrm{E}(K, O_{j-2})$$

于是，我们可以把加密表达式写成如下形式：

$$C_j = P_j \oplus \mathrm{E}(K, [C_{j-1} \oplus O_{j-1}])$$



(a) 加密



(b) 解密

图 7.6　输出反馈模式（OFB）

通过重排各项，解密可表示如下：

$$P_j = C_j \oplus \mathrm{E}(K, [C_{j-1} \oplus P_{j-1}])$$

我们可以如下定义 OFB 模式。

OFB	I_1=时变值 $I_j = O_{j-1}$ $j=2,\cdots,N$ $O_j = \mathrm{E}(K, I_j)$ $j=1,\cdots,N$ $C_j = P_j \oplus O_j$ $j=1,\cdots,N$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	I_1=时变值 $I_j = O_{j-1}$ $j=2,\cdots,N$ $O_j = E(K, I_j)$ $j=1,\cdots,N$ $P_j = C_j \oplus O_j$ $j=1,\cdots,N$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$

令分组的长度为 b。如果明文的最后一个分组包含 u 位（用*指示），$u<b$，那么最后的输出分组 O_N 的最左边的 u 位用来做异或运算。最后的输出分组的其余 $b-u$ 位丢弃不用。

如同 CBC 和 CFB 一样，OFB 模式需要一个初始化向量。在 OFB 模式时，IV 必须是一个时变值，即 IV 对每次加密运算都是唯一的。原因是加密输出的分组构成的序列 O_i 仅仅依赖密钥和 IV，而不依赖明文。因此，对于给定的密钥和 IV，用于和明文流进行异或运算的输出位流是固定的。如果两个不同的消息在相同的地方有一个相同的明文分组，那么攻击者就能够判断出那部分的 O_i 输出流。

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。读者若想做深入了解，请参阅文献[VOYD83]。

OFB 具有典型流密码的结构，因为密码产生的位流是初始值和密钥的函数，且产生出的位流和明文进行了异或（参见图 4.1）。产生的位流（即和明文进行异或运算的位流）本身是和明文独立的；图 7.6 用虚线圈起的盒子突出显示了这一点。与第 8 章将要讨论的流密码相比，差别是 OFB 一次加密一个明文分组，分组的典型长度是 64 位或 128 位。许多流密码一次加密一个字节。

7.6 计数器模式

随着计数器模式（CTR）在 ATM（异步传输模式）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，但是实际上这种模式在很早以前就已经提出来了（如文献[DIFF79]）。

图 7.7 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 对此的唯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。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（模 2^b，b 为分组长度）。加密时，计数器加密后与明文分组异或得到密文分组没有链接。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分组异或来恢复明文分组。因此，解密时必须知道初始计数器的值。给定计数器序列 T_1，T_2，…，T_N，我们可以如下定义 CTR 模式。

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j)$ $j=1,\cdots,N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j)$ $j=1,\cdots,N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

最后的明文分组可能不是完整的分组，只是长为 u 位的部分分组，最后输出分组的最左边的 u 位用来做异或运算；其余的 $b-u$ 位丢弃。与 ECB，CBC 及 CFB 方式不同的是，由于 CTR 的结构关系，我们不需要对明文进行填充。

如同 OFB 模式一样，初始计数器的值必须为时变值；也就是说，使用相同密钥加密的所有消息必须各自具有不同的 T_1。而且，所有消息的所有 T_i 值也必须唯一。如果违反了这个要求，一个计数器的值使用了多次，那么这个计数器值对应的明文分组的保密性就会泄露。特别地，如果用那个给定的计数器值加密的任何明文已知了，那么加密函数的输出很容易从关联的密文推导出来。对于所有使用该计数器值加密的明文来说，从这个输出很容易由密文推导出对应的明文。

保证计数器值唯一的一个办法是每一个消息计数器都增加 1。即每个消息的第一个计数器的值都要比前一条消息的最后一个计数器的值多 1。

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 模型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。算法的最大吞吐量受限于执行一次加密或解密所需要的交互时间。在 CRT 模型中，吞吐量仅受可并行度的限制。
- **软件效率**　类似地，因为 CTR 模式能够进行并行计算，因此可以充分利用能够支持如下并行特征的各类处理器，如提供流水线、每个时钟周期的多指令分派、大量的寄存器和 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，可以预处理加密盒的输出，这个输出又进一步作为 XOR 函数的输入，如图 7.7 所示。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。

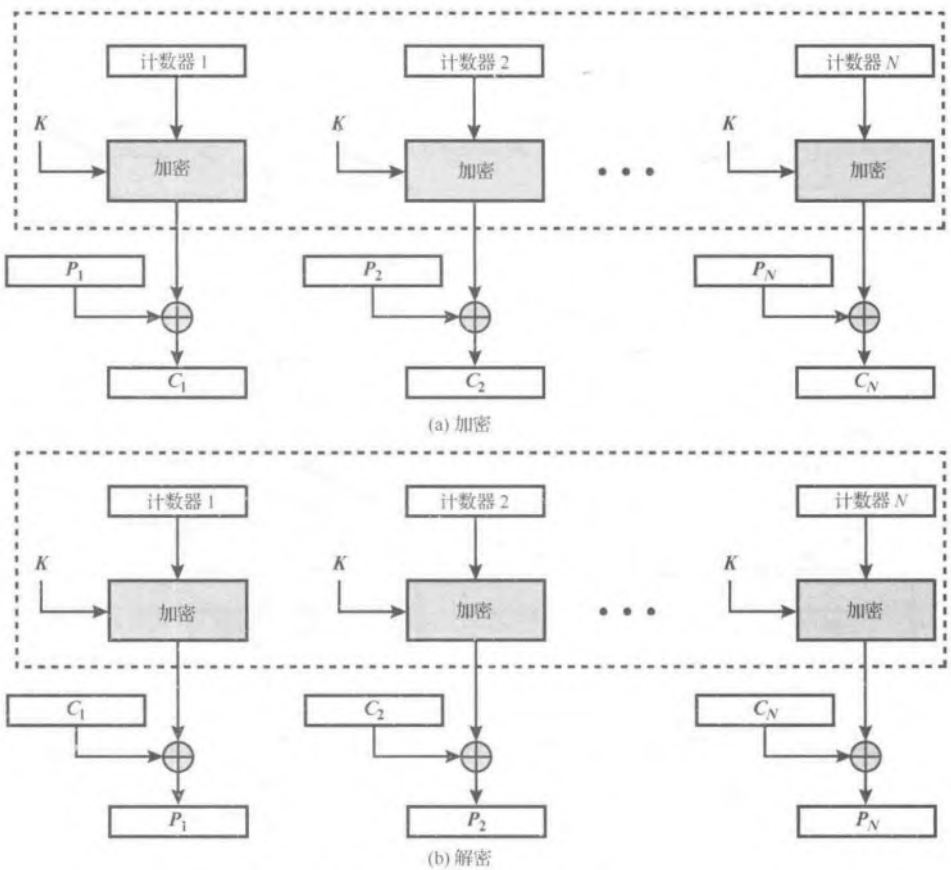


图 7.7　计数器（CTR）模式

- **随机访问** 明文或密文的第 i 个分组能够用一种随机访问的方式处理。在链接模式下知道前面的 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储存好了，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 模式只要求实现加密算法并不要求实现解密算法。当加密算法与解密算法有很大的不同时，这种模式就显得很重要，而高级加密标准就是这样。另外，也不用实现解密密钥扩展。

请注意，除了 ECB 模式外，NIST 批准的所有分组密码工作模式都含有反馈。这在图 7.8 里很明显。为了突出显示这个反馈机制，将加密函数想为从输入寄存器取输入，而把输出存在输出寄存器中，其中输入寄存器的长度等于加密分组长度。这个想法很有用。输入寄存器由反馈机制每次更新一个分组。每次更新后，加密算法执行一次，结果放在输出寄存器中。同时访问明文分组。注意 OFB 和 CTR 模式的输出独立于明文和密文。因此，它们是流密码的自然候选算法，通过 XOR 运算每次加密一个明文分组。

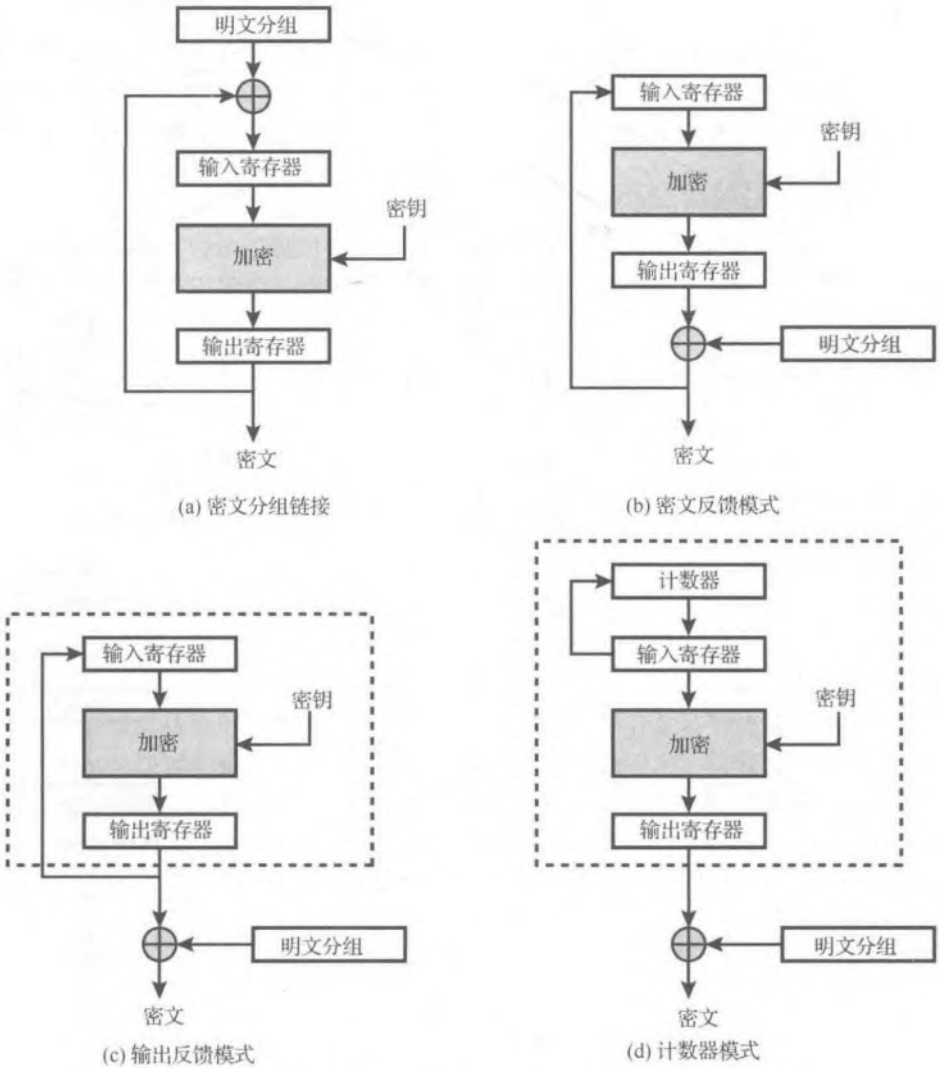


图 7.8 各工作模式的反馈特征

7.7　用于面向分组的存储设备的 XTS-AES 模式

2010 年，NIST 正在忙着批准另外一个分组密码工作模式：XTS-AES。这个模式也是 IEEE 的标准，即 IEEE Std 1619-2007，是由 IEEE 存储安全工作组（P1619）开发的标准。该标准描述了一种对基于扇区的设备上的数据进行加密的方法，此时的威胁模型包括对手对存储数据的访问。该标准已经受到工业界的广泛支持。

7.7.1　可调分组密码

XTS-AES 模式基于文献[LISK02]提出的可调分组密码的概念，其功能很像第 22 章讲述的给口令加盐的方法。文献[ROGA04]第一次描述了这个概念的形式。

在测试 XTS-AES 模式前先设想一下可调密码的通用结构。可调密码有三个输入：一个明文 P、一个对称密钥 K、一个微调 T，会产生一个密文输出 C。我们可写成这个形式 C=E(K, T, P)。微调不需要保密，然而密钥的使用需要保密。对相同的明文和密文使用不同的微调会产生不同的输出。可调分组密码的基本结构的实现如图 7.9 所示。加密可表示为

$$\mathrm{C} = \mathrm{H}(T) \oplus \mathrm{E}(K, \mathrm{H}(T) \oplus P)$$

其中 H 为哈希函数。解密使用相同的结构，输入明文，使用解密函数代替加密函数。把解密过程记为

$$\mathrm{H}(T) \oplus C = \mathrm{E}(K, \mathrm{H}(T) \oplus P)$$
$$\mathrm{D}[K, H(T) \oplus C] = \mathrm{H}(T) \oplus P$$
$$\mathrm{H}(T) \oplus \mathrm{D}(K, \mathrm{H}(T) \oplus C) = P$$

现在很容易对每个分组使用一个不同微调值进行分组密码模式操作。大体上使用 ECB 模式，但对于每个分组微调会改变，这克服了 ECB 的主要安全弱点，即相同明文产生相同密文。

7.7.2　存储加密的要求

加密存储的数据（也称为静止数据）的要求和加密传输数据的要求有些不同。P1619 标准在设计上有如下特征：

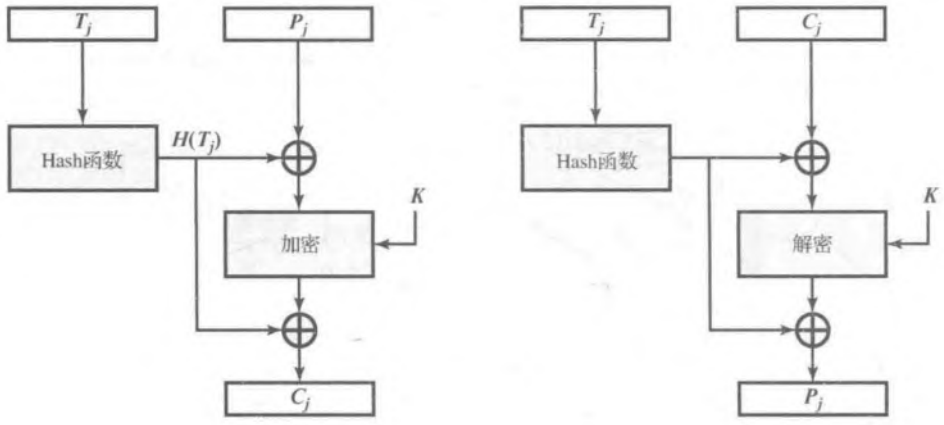
（1）攻击者可以随意获取密文。如下环境可以导致这种局面：
　（a）授权一组用户可以访问数据库。数据库内的一些记录经过加密，于是只有特定的用户可以成功读写它们。其他用户可以检索加密记录但没有密钥的话无法阅读。
　（b）一个非授权用户设法能访问加密记录。
　（c）数据磁盘或者便携电脑被偷了，让对手可以访问加密数据。
（2）数据版面在存储介质上或传输中不能改变。加密后的数据必须和明文数据有相同的大小。
（3）以规定长度的分组为单位访问数据，相互间独立。即授权用户可以以任何顺序访问一个或多个分组。
（4）加密以 16 字节的分组为单位，分组相互之间独立（如果某个扇区的大小不是 16 字节的倍数，那么该扇区的最后两个明文分组是例外）。
（5）除了数据分组在整个数据集里的位置外，没有使用其他元数据。
（6）不同地方相同的明文加密后得到不同的密文，但是当再次写到相同的位置时总是写为相同的密文。
（7）由一个同标准相容的设备加密的数据可以通过构造一个同标准相容的设备来解密。

P1619 工作组考虑了现存的一些工作模式用于存储数据的处理。对于 CTR 模式，具有对加密媒体写权限的对手可以改变明文的任何位，这仅仅通过修改密文的对应位就可以达到。

接着，我们考虑要求 6 和使用 CBC 模式。为了实现不同位置的相同明文加密为不同密文的要求，IV 可以从扇区号推导而来。每个扇区含有多个分组。从新位置读取扇区的应用仍然可以获得相同的明文扇区（也许第一个 128 位会例外）。例如，这意味着若一个敌手可以从第二个位置，而不是从第一个位置读取一个扇区，则该对手可以通过操作密文的方式发现第一个位置的扇区内容。另一个弱点是对手可以对明文的任何位取反，而这通过把相应的前一分组密文的相应位取反即可，当然副作用是把前一分组“随机化了”。

7.7.3 单分组的运算

图 7.10 显示了单个分组的加密和解密。运算包括具有两个密钥的 AES 算法的两个实例。如下参数和算法有关联。



(a) 加密　　(b) 解密

图 7.9　可调分组密码

密钥 Key　256 或 512 位的 XTS-AES 密钥；可以解析为两个长度相同的密钥 Key_1 和 Key_2 连接的结果，即 $Key=Key_1||Key_2$。

P_j　明文的第 j 个分组。除了最后一个分组外所有分组的长度为 128 位。明文数据单元，典型的是一个磁盘扇区，由明文分组序列 $P_1, P_2, \cdots, P_m$ 构成。

C_j　密文的第 j 个分组。除了最后一个分组外所有分组的长度为 128 位。

j　数据单元内 128 位分组的序列号。

i　128 位微调参数的值。每个数据单位（扇区）被赋予一个具有非负整数值的微调值。这个微调值按顺序赋值，可以从任意非负整数开始。

α　$GF(2^{128})$的一个本原元，对应于多项式 x（即 $000\cdots010_2$）。

α^j　$GF(2^{128})$内 α 自乘 j 次。

$\oplus$　位异或。

$\otimes$　两个二进制系数的多项式模 $x^{128}+x^7+x^2+x+1$ 乘法。因此，这是 $GF(2^{128})$内的乘法。

本质上，参数 j 的功能类似 CTR 模式里的计数器，以保证一个数据单元内不同位置的相同明文分组加密为不同的密文分组。参数 i 从数据单元的层面上说相当于时变值，以保证两个不同的数据

单元内相同位置相同明文分组加密为不同的密文分组。更一般地，它可以保证位于不同位置的相同数据单元加密为不同的密文数据单元。

单分组的加密和解密可以描述为

XTS-AES 分组运算	$T=\mathrm{E}(K_2, i)\otimes \alpha^j$ $PP=P\oplus T$ $CC=\mathrm{E}(K_1, PP)$ $C=CC\oplus T$	$T=\mathrm{E}(K_2, i)\otimes \alpha^j$ $CC=C\oplus T$ $PP=\mathrm{D}(K_1,\mathrm{CC})$ $P= PP \oplus T$

为了明白解密确实可以恢复明文，让我们把加密和解密的最后一行进行扩展。对于加密，有

$$C = CC \oplus T = \mathrm{E}(K_1, PP) \oplus T = \mathrm{E}(K_1, P\oplus T) \oplus T$$

对于解密有

$$P = PP \oplus T = \mathrm{D}(K_1, CC) \oplus T = \mathrm{D}(K_1, C \oplus T) \oplus T$$

现在，我们代替 C：

$$\begin{aligned} P &= D(K_1, C \oplus T) \oplus T \\ &= \mathrm{D}(K1, [\mathrm{E}(K_1, P\oplus T) \oplus T] \oplus T) \oplus T \\ &= \mathrm{D}(K1, [\mathrm{E}(K_1, P\oplus T)) \oplus T \\ &= (P \oplus T) \oplus T = P \end{aligned}$$

7.7.4　在扇区上的运算

扇区或者数据单元的明文组织为 128 位的分组，分组标记为 $P_0, P_1, \cdots, P_m$。最后的分组也许是空的，也许含有 1～127 个位。换句话说，XTS-AES 算法的输入是 m 个 128 位的分组，最后一个分组可能为部分分组。

对于加密和解密，每一个分组都独立处理，过程如图 7.10 所示。当最后一个分组不足 128 位时，唯一的例外就会发生。此时，最后的两个分组要使用密文窃取技术而不是填充技术来加密/解密。图 7.11 展示了这一方案。P_{m-1} 是最后一个满的明文分组，而 P_m 是最后的明文分组，其含有 s 位，其中 $1\leqslant s\leqslant 127$。$C_{m-1}$ 是最后一个满的密文分组，而 C_m 是最后一个密文分组，其含有 s 位。

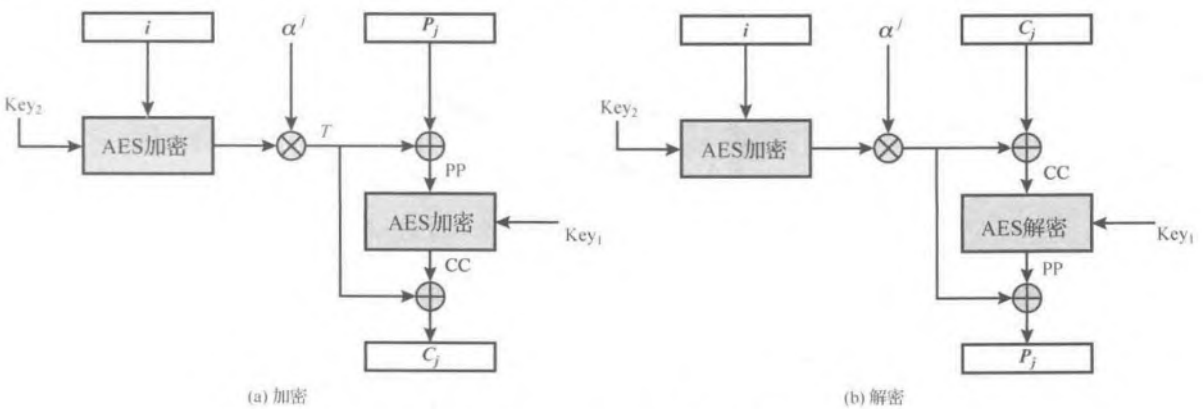


图 7.10　单个分组的 XTS-AES 运算

我们把图 7.10 的加密和解密算法标记为

分组加密：XTS-AES-blockEnc(K, P_j, i, j)

分组解密：XTS-AES-blockDec(K, C_j, i, j)

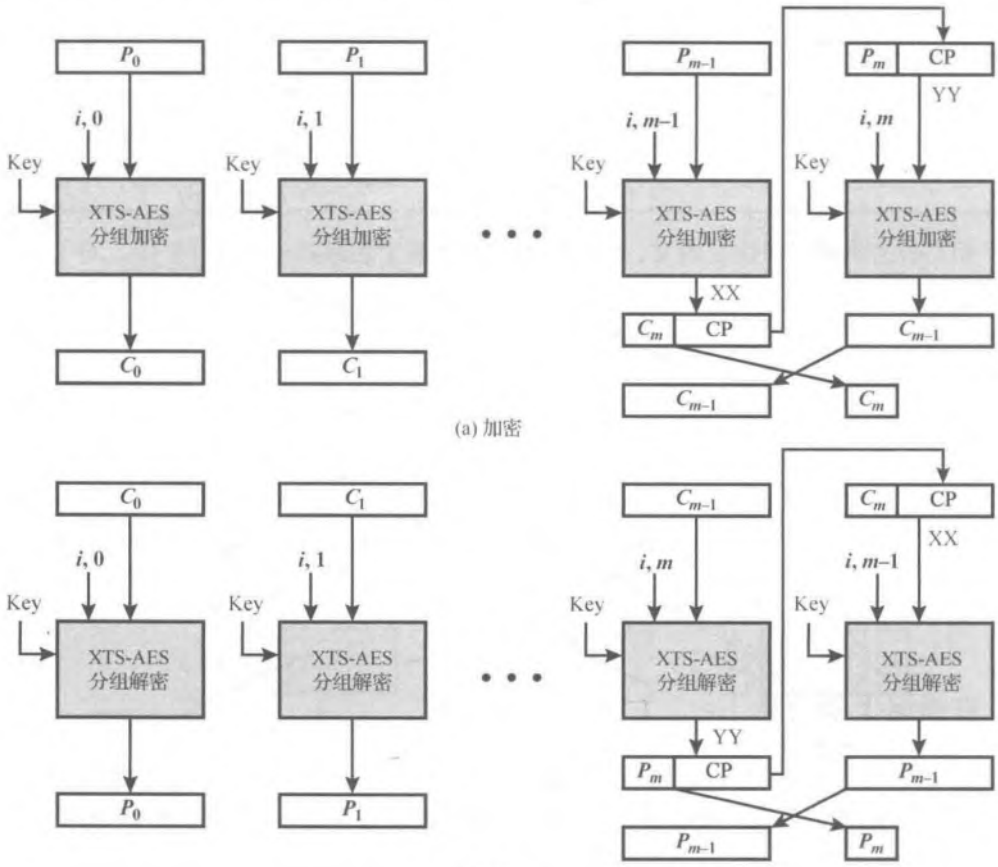


图 7.11 XTS-AES 模式

那么，如果最后的分组是空的，XTS-AES 模式如下定义：

XTS-AES 模式，最后分组为空	C_j= XTS-AES-blockEnc(K, P_j, i, j) j=0,⋯,m–1
	P_j= XTS-AES-blockEnc(K,C_j,i, j) j=0,⋯,m–1
XTS-AES 模式，最后分组含 s 位	C_j= XTS-AES-blockEnc(K, P_j, i, j) j=0,⋯,m–2 XX= XTS-AES-blockEnc(K, P_{m-1}, i, m–1) CP=LSB_{128-s}(XX) YY=P_m‖CP C_{m-1}= XTS-AES-blockEnc(K, YY, i, m) C_m=MSB_s(XX)
	P_j= XTS-AES-blockDec(K, C_j, i, j) j=0,⋯,m–2 YY= XTS-AES-blockDec(K,C_{m-1},i,m–1) CP=LSB_{128-s}(YY) XX=C_m‖CP P_{m-1}= XTS-AES-blockDec(K, XX, i, m) P_m=MSB_s(YY)

可以发现，如同 CTR 模式一样，XTS-AES 模式适合并行执行。因为没有链接，多个分组可以同时加密或解密。与 CTR 不同的是，XTS-AES 模式包含一个时变值（参数 i）以及一个计数器（参数 j）。

7.8　格式保持加密

FPE 是一种加密技术，这种加密技术将一种格式的明文作为输入，并输出相同格式的密文。例如，信用卡的明文由 16 个十进制数字组成。PFE 可以接受这种输入，并输出一个同样格式的密文，即 16 个密文组成的密文。值得注意的是，密文可能并不像一个正常的信用卡数字。但是它和明文的格式是相同的，和明文的存储方式也是一致的。

一个简单的加密算法并不能保证明文和密文的格式一致性，除了二进制的情况。例如表 7.2 就展示了 3 种不同的输入格式，它们都将被尝试用 FPE 加密。第三行展示了由 FPE 加密的结果。而第四行是由 AES 加密的结果。

表 7.2　FPE 与 AES 加密方法的比较

	信用卡号码	税号	银行账号
明文	8123 4512 3456 6780	219-09-9999	800N2982K-22
FPE	8123 4521 7292 6780	078-05-1120	709G9242H-35
AES(hex)	af411326466add24 c86abd8aa525db7a	7b9af4f3f218ab25 07c7376869313afa	9720ec7f793096ff d37141242e1c51bd

7.8.1　研究动机

FPE 使得针对传统密码应用的加密算法的改进更加便利，但是传统的加密模式却不能做到这一点，因为传统算法或多或少地会影响到数据所在的数据域或者路径。FPE 是一个有用的加密工具，它的应用领域包括商业信息安全、数据防护和在传统数据上的透明加密。

FPE 对一个数据库中之前的数据的兼容性强，在使用 FPE 之前的所有数据都不需要更改。不同于其他加密算法如 AES 或者 TDEA，如果使用 FPE，那么数据存储的策略和实现都不需要更改。唯一需要改变的是明文，因此使得对于一个系统的改变达到最小。

FPE 可以应用于以下传统领域：

- COBOL 数据处理应用：任何对于结构记录的改变都需要改变所有的相关代码。这些代码通常由许多模块组成，每一个平均有 5000～10000 行代码。
- 数据库应用：数据域是针对字符串的数据库不能存储传统加密算法产生的二进制流。如果不增加数据长度，Base64 对二进制流文件的解码是不够的，需要增加相关数据域的长度。
- FPE 加密的数据在经过压缩后，可以很容易地被用来告诉数据传输，但是 AES 加密产生的二进制流却做不到这一点。

7.8.2　设计 FPE 的难点

一个普通的标准 FPE 应该达到如下要求：

（1）密文的长度和格式要和明文相同。

（2）它应该适用于多数据格式的应用领域。如十进制数字、小写字母符号和完整的标准键盘或者国际键盘符号。

（3）它应该能处理变长的数据。

（4）安全性应该和 AES 的相当。

（5）对于短明文数据，它的安全性不会降低。

要达到第一个要求是曲折的。正如表 7.2 所展示的，直接用 AES 产生的 128 位数据块和要求的数据格式有不小差距。同时，一个标准的对称加密算法不能很好地适用于 FPE。

考虑一个简答的例子。假设我们需要一个算法可以加密最大长度为 32 的十进制数字串。用 16 位的空间来存储数据，其中每一个数字用 4 位的空间来表示（如 6 用 0101 来表示）。下面我们使用 AES 来加密 128 位的数据块：

（1）明文输入 X 可以表示成长度为 16 的 4 位数字串 $X[1]\cdots X[16]$。如果明文长度小于 16，缺少的部分用 0 来填充。

（2）把 X 当成长度为 128 位的二进制流用 AES 加密，密钥是 K，生成密文 $Y=\mathrm{AES}_K(X)$。

（3）将 Y 看成是由 16 个 4 位元素组成的。

（4）由于 Y 的某些元素可能大于 9（如 1100），因此为了产生目标格式的密文 Z，要进行如下的计算：

$$Z[i]=Y[i]\bmod 10,\quad 1\leqslant i\leqslant 16$$

如此就可以产生有 16 个十进制数字组成的密文，并且符合目标格式。但是上面的算法不符合算法可逆的加密算法的基本要求。对 Z 进行解密是不可能得到明文 X 的，因为上述算法是单向的。单向的例子有许多，如 $12 \bmod 10 = 2 \bmod 10 = 2$。所以，我们得设计一个可逆的函数并达到安全和格式符合这两个要求。

设计 FPE 的第二个难点是某些时候输入的长度比较短。例如，考虑 16 位的信用卡数字（CCN）。前 6 个数字是发行确认数字，这些数字用来确认信用卡发行方。最后一个数字是冗余位，用来纠错。剩下的 9 个数字是用户的账户数字。但是，有相当多的应用需要信用卡预留最后的 4 个数字（原来的纠错位加相邻的 3 个账户数字）。现在假设一个攻击者可以得到许多明文密文对。没个明文密文对应于不同的 CCN，但是这些 CCN 的中间 6 个数字是相同的。在一个大型信用卡数字数据库中，有许多信用卡数字的中间 6 个数字是相同的。攻击者可以收集大量的明文密文对。这可以用来攻击数据库中未知的密文。正如文献[BELL10a]中指出的，一个拥有 100000000 个条目的数据库中，对于一个给定的中间 6 位数字，平均有 100 个 CCN 共享它。因此，如果攻击者已经掌握了 k 个 CCN 并且可以进入相应的数据库，那么攻击者就可以成功地攻击 $100k$ 个 CCN。

解决第二个难题的方法是使用可以调整的数据块；这个概念将在 7.7 节中讲解。例如，一个 CCN 的调整可以用原来 CCN 的前两个数字和最后四个数字完成，我们称它们为 tweak。在加密之前，tweak 就会被逐个模 10 加到中间的 6 个数字上，之后再对得到的结果进行加密。不同的 CCN 即使有 6 个相同的中间数字，在经过上述的步骤后也会得到不同的结果，因此也就会有不同的加密结果。考虑下面的例子：

CCN	Tweak	明文	明文 + Tweak
4012 8812 3456 1884	401884	123456	524230
5105 1012 3456 6782	516782	123456	639138

两个 CCN 虽然有相同的中间 6 位数字，但是在模 10 加 tweak 后就会产生不同的 6 位数字。

7.8.3 Feistel 结构的格式保留加密算法

正如前面的部分所展示，FPE 的挑战在于抓取安全的数据、保留数据格式和可逆性。已经有许多的方法在近几年被提出[ROGA10,BELL09]。这些方法的绝大部分都用到了 Feistel 结构。虽然 IBM 在半

世纪前介绍 Lucifer 加密算法时就引入了 Feistel 结构，但是它仍是应用密码的坚实基础。

本节具体讲解为什么 Feistel 结构能被利用到 FPE 中，在下一节中，将介绍三个典型的基于 Feistel 结构的接收 NIST 许可证的算法。

加密和解密　图 7.12 显示了在所有 NIST 算法中使用的 Feistel 结构。图 7.12(a)显示加密，图 7.12(b)显示解密。图 7.12 中的结构与图 4.3 所示的结构相同，但是为了方便表示，我们省去了每一轮处理的最后一步。

加密算法的输入是有 $n=u+v$ 个字符的字符串。如果 n 是偶数，那么使 $u=v$，否则 u 和 v 相差 1。字符串的两个部分将经过偶数次处理以产生一个由 n 个字符组成的密文块，并且这个密文块的格式和明文一致。第 i 轮的输入是 A_i 和 B_i，由前一轮的运算产生。

每一轮处理都有相同的结构。在偶数轮处理中，将对 A_i 做替换处理。用 F_K 处理 B_i，然后用 F_K 得到的结果和 A_i 做模加运算。模加运算和模块的选择将在后面的部分介绍。在奇数轮处理中，将对 B_i 做替换处理。F_K 是单向函数，它将输入转换成二进制字符串，并在字符串上做抓取变换处理，然后将得到的结果在转换成和输入有相同格式和长度的字符串。这个函数将密钥 K、明文长度 n、tweak T 和轮数 i 作为输入参数。

注意在偶数轮，F_K 的输入是 v 个字符，并且模加运算将产生由 u 个字符组成的结果，但是在奇数轮，F_K 的输入是 u 个字符，并且模加运算将产生由 v 个字符组成的结果。总轮数是偶数，因此最后的结果由 A 和 B 组成，长度分别是 u 和 v 组成，和明文的格式一致。

解密的流程和加密的基本相同。不同之处如下：（1）原来的模加运算编程了减法运算，这个减法运算时模加运算的逆运算；（2）轮下标和加密过程相反。

为了展示解密运算产生了正确的结果，图 7.12(b)的左半边和右半边分别展示了加密过程和解密过程，流程方向分别为向下和向上。正如图标所展示的，在每一轮，解密过程的中间值和对应位置的加密过程的中间值相同。我们可以通过分析来证明上述过程，从图的底部开始。密文将在 $r-1$ 轮产生，它本质上就是 $A_r \| B_r$，其中 A_r 和 B_r 的长度分别是 u 和 v，加密过程的 $r-1$ 可用以下公式表示：

$$A_r = B_{r-1}$$
$$B_r = A_{r-1} + \mathrm{F}_K[B_{r-1}]$$

因此我们可以定义模加运算的逆运算，这些运算可由下面的公式表示：

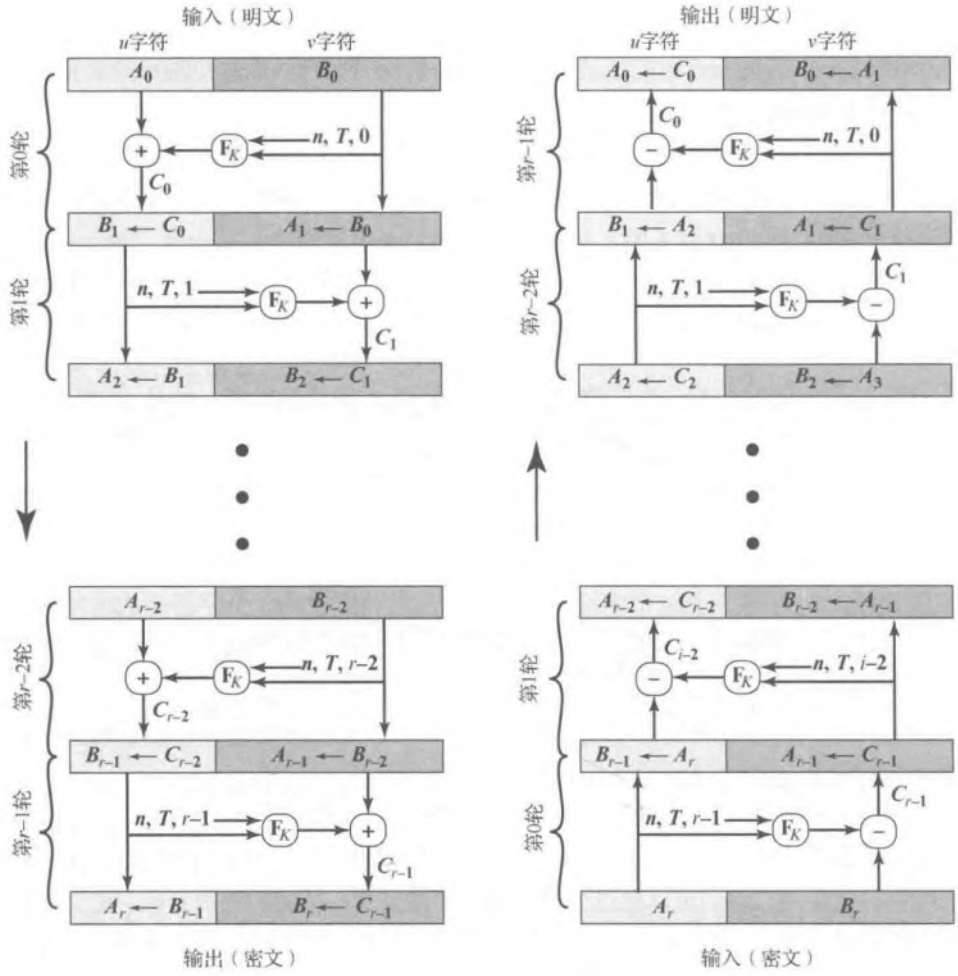
$$B_{r-1} = A_r$$
$$A_{r-1} = B_r - \mathrm{F}_K[B_{r-1}]$$

上面的两个公式展示了解密运算的 0 轮，所以 0 轮的输出和 $r-1$ 轮的输入相同。这样的对应关系在 r 轮运算中始终存在，正如上述公式所展示。

注意在该算法的衍生版本中 F 不一定是一个可逆函数。为了解释这个问题，考虑这样的限制，即不管输入是什么，F 的每个输出都是常量。上述的公式所揭示的关系仍然存在。

字符串　在一些 NIST 算法和一些 FPE 算法已经被提出了，它们和由一串元素组成的明文一起被使用，这些元素被称为字符。特别地，由 1 个或者 2 个符号组成的集合称为字母表，而字母表中的元素称为特征。一个字符串由字母表中的有限个元素组成。独立的字符可以在字符串中重复出现。字母表中不同字符的个数被称为这个字母表的基，也被称为这个字母表的基数。例如小写英文字母表 $a, b, c, \cdots$的基数是 26。为了加密和解密，明文字母表将会被转化为量，量是一个非负的整数，并且它的值比字母表的基小。例如，对于一个小写英文字母表，我们可以将 $a, b, c, \cdots, z$ 映射为 $0, 1, 2, \cdots, 25$。

这种方法的局限是明文中的所有元素对应的字母表必须有相同的基。所以一个由英文字母表和十进制数字组成的认证数字在保持格式不变的前提下不能被现有的 FPE 算法处理。



(a) 加密　　(b) 解密

图 7.12　FDE 加密的 Feistel 架构

为了表示这些转化，NIST 文档定义了一些标识［参见表 7.3(a)］。首先，它假设所有的字符串（元素属于英文字母表）由数字串表示。为了将数字串 X 转换成 x，$\mathrm{NUM}_{\mathrm{radix}}(X)$ 可以被使用来达到这个目的。将 X 视为 $X[1]\cdots X[m]$，其中 $X[1]\cdots X[m]$ 按高位到低位的顺序排列，函数定义如下：

$$\mathrm{NUM}_{\mathrm{radix}}(X)=\sum_{i=1}^{m}X[i]\mathrm{radix}^{m-i}=\sum_{i=0}^{m-1}X[m-i]\mathrm{radix}^{i}$$

有　$0\leqslant \mathrm{NUM}_{\mathrm{radix}}(X)<\mathrm{radix}^{m}$　和　$0\leqslant X[i]<\mathrm{radix}$

例如，在基 26 下考虑字符串 zaby，那么可以将 zaby 表示成 25 0 1 24。最后将得到数字 $x=(25\times 26^{3})+(1\times 26^{1})+24=439450$。考虑反方向的转换，将 $x<\mathrm{radix}^{m}$ 转换成长度为 m 的数字串 X，函数 $\mathrm{STR}^{m}_{\mathrm{radix}}(X)$ 将会被利用。

$$\mathrm{STR}^{m}_{\mathrm{radix}}(x)=X[1]...X[m]$$

$$X[i]=\left\lfloor\frac{x}{\mathrm{radix}^{m-i}}\right\rfloor \bmod \mathrm{radix},\quad i=1,\cdots,m$$

表 7.3　FPE 算法中的表示和参数

$[x]^s$	将一个整数转化为字节串；它是一个有 s 个字节组成并完成对数字 x 进行解码的字节串，$0 \leqslant x < 2^{8s}$。等价的标识是 $\mathrm{STR}_2^{8s}(x)$
$\mathrm{LEN}(X)$	字符串 X 的长度
$\mathrm{NUM}_{\mathrm{radix}}(X)$	将一个串转换为数字。也就是说，结果是用 radix 为基的数表示 X 的数字，并且结果的数据位按大端顺序排列
$\mathrm{PRF}_K(X)$	该函数将以 X 为输入，并利用密钥 K 产生一个 128 位的伪随机输出
$\mathrm{STR}_{\mathrm{radix}}^m(X)$	给定一个小于 radix^m 的整数 x，该函数将以 radix 作为 x 的基，生成一个由 m 个字符组成的字符串，结果按大端排列
$[i..j]$	i 到 j 的数字集合，包括 i 和 j
$X[i..j]$	该字符串是 X 的子字符串，包含 $X[i]$ 到 $X[j]$ 的字符、$X[i]$ 和 $X[j]$
$\mathrm{REV}(X)$	给定一个位串 X，返回的位串的顺序和 X 的正好相反

(a) 标识

radix	在给定字母表中基、或者字符的个数
tweak	秘密性没有受到保护的加密算法和解密算法的输入
tweakradix	tweak 字符串的基
minlen	多字符的最短信息长度
maxlen	多字符的最长信息长度
maxTlen	最大 tweak 长度

(b) 参数

通过字符到数字的映射和 NUM 函数的使用，一个明文字符可以被转换成一个数字，并作为一个无符号整数存储。我们更愿意将这个无符号整数存储为一个位串，应为如果这样我们就可以将它作为 F_K 中的位抓取算法的输入。但是，不同的平台对于无符号整数的存储方法也是不同的，一些以小端存储，一些以大端存储。所以需要多加一步。由 STR 函数的定义可知，$\mathrm{STR}_2^{8s}(x)$ 将产生一个长度为 $8s$ 的位串，即等价的长度为 s 的字节串，无论 x 被怎样存储，这个二进制整数始终是大端的。为了方便表示，使 $[x]^s = \mathrm{STR}_2^{8s}(x)$。因此，$\left[\mathrm{NUM}_{\mathrm{radix}}(X)\right]^s$ 将一个字符串 X 转换为一个无符号整数然后再转化为长度为 s 字节的大端存储的字节串。

那么，前面的例子应该很好地解释了包含的问题。

字　符　串	"zaby"
以数字串 X 表示字符串	25 0 1 24
将 X 转换为数字 $x = \mathrm{NUM}_{26}(X)$	十进制：439450 十六进制：6B49A 二进制：1101011010010011010
将 x 以 32 位的无符号整数存储在以大端顺序存储的机器上	十六进制：00 06 B4 9A 二进制：00000000000001101011010010011010
将 x 以 32 位的无符号整数存储在以小端顺序存储的机器上	十六进制：9A B4 06 00 二进制：10011010101101000000011000000000
在不考虑大小端存储的情况下，将 x 转化为 32 位的位串，表示为 $[x]^4$	00000000000001101011010010011010

F_K 函数　我们现在可以定义具有一般意义的函数 F_K。F_K 函数的核心是以位串作为输入和输出的随机函数。为了方便，位串应该是 8 的倍数，如此就可以很容易地将其转化为一个字节串。在偶数轮将 m 定义为 u，在奇数轮将 m 定义为 v；这将确保输出字符串的长度符合预期。用一个数字来表示一个长度为 m 字节的字符串，并将 b 定义为存储这个数字所需要的字节数。之后，包含 F_K 的每一轮操作都由下面的几步组成（在 i 轮，A 和 B 分别对应于 A_i 和 B_i）：

1. $Q \leftarrow [\text{NUM}_{\text{radix}}(B)]^b$	将数字串 X 转换为长度为 b 字节的字节串
2. $Y \leftarrow \text{RAN}[Q]$	伪随机函数将 Q 映射到伪随机字节串 Y
3. $y \leftarrow \text{NUM}_2(Y)$	将 Y 转换成无符号整数
4. $c \leftarrow (\text{NUM}_{\text{radix}}(A)+y) \bmod \text{radix}^m$	将 A 转换成一个整数，再模 radix^m 加 y
5. $C \leftarrow \text{STR}^m_{\text{radix}}(c)$	将 c 转换成长度为 m 的数字串 C
6. $A \leftarrow B$ $B \leftarrow C$	将前一轮循环的 B 赋给 A，将 C 赋给 B，一轮循环到此结束

步骤 1 至步骤 3 组成了轮函数 F_K。步骤 3 的输入是 Y，一个没有被组织的位串。因为不同硬件结构的字长和数据存储方式不同，因此有必要通过 $\text{NUM}_2(Y)$ 得到一个无符号整数 y。存储的 y 和 Y 可能相同，也可能不同。

正如前面提到的，步骤 2 的伪随机函数不一定可逆。它的目的是提供一个随机的、可被分割的位串。对于 DES，可以通过使用固定的 S 盒来达到这个目的，附录 S 对这个问题进行了介绍。显然所有基于 Feistel 架构的 FPE 策略都将 AES 作为自己的抓取函数来达到更好的安全性。

基数、信息长度和位长度的关系 考虑一个长度为 len 并以 radix 为基数的数字串 X。如果我们将它转化为 $x = \text{NUM}_{\text{radix}}(X)$，那么 x 的最大值就是 $\text{radix}^{\text{len}} - 1$。需要用来解码 x 的位数是

$$\text{bitlen} = \lceil \text{LOG}_2(\text{radix}^{\text{len}}) \rceil = \lceil \text{lenLOG}_2(\text{radix}) \rceil$$

通过观察可以发现，增加基数（radix）或者信息长度（len）都会增加位长度（bitlen）。通常，我们希望为位长度设定一个上限，例如 128 位（AES 的输入大小）。我们也希望 FPE 能处理不同的基数。对于现在讨论的 FPE 和那些后续讨论的算法，我们希望这种设计达到这样的要求：给定一个基数值的范围，之后再给出最大的字符串长度，我们可以得到拥有固定位长度的算法。假设基数的范围是 2 到 maxradix，最大的信息长度是 maxlen，那么有下面的关系式：

$$\text{maxlen} \leqslant \lfloor \text{bitlen} / \text{LOG}_2(\text{radix}) \rfloor, \text{ 或等效地}$$

$$\text{maxlen} \leqslant \lfloor \text{bitlen} \times \text{LOG}_{\text{radix}}(2) \rfloor$$

例如，当基数为 10 时，有 $\text{maxlen} \leqslant \lfloor \text{bitlen} \times 0.3 \rfloor$；当基数为 26 时，$\text{maxlen} \leqslant \lfloor \text{bitlen} \times 0.21 \rfloor$。当位长度固定时，基数越大，最大的信息长度（字符串长度）越小。

在 2013 年，NIST 提出了 SP 800-38G：*Recommendation for Block Cipher Modes of Operation*：格式一致加密的方法。这个建议包含了三种格式一致加密的方法，被分别称为 FF1，FF2 和 FF3。这三种方法都用到了 Feistel 架构，如图 7.12 所示。不过它们分别采用了不同的 F_K，而这三种 F_K 都通过 AES 算法构建。主要不同如下所示：

- FF1 支持的字符串明文和 tweak 的长度范围最大。为了达到这个目的，加密轮函数采用了 CBC(cipher-block-chaining)方式。而 FF2 和 FF3 则仅仅使用了简答的 ECB 加密。
- FF2 利用了由加密密钥和 tweak 产生的子密钥，而 FF1 和 FF3 则直接使用了加密密钥。这种方式可以防止原密钥受到侧信道攻击（测信道攻击从密码系统的硬件实现上得到信息），但不能防止暴力攻击和密码分析攻击。这种攻击通常会尝试分析能量消耗或者运行时间来推断密钥字节。
- FF3 用到的加密轮数最少，只有 8 轮，而 FF1 和 FF2 都只有 10 轮，但是它提供 tweak 的灵活性也是最低的。

FF1 算法 FF1 算法在被提交到 NIST 的时候，被命名为 FFX[Radix]，并被作为一种建议的 FPE 模式[BELL10a,BELL10b]。FF1 使用了随机函数 $\text{PRF}_K(X)$（参见图 7.13），这个函数的作用是用长度为 128

位和加密密钥 K 的数倍的输入 X 来产生 128 位的输出。本质上，$\mathrm{PRF}_K(X)$ 利用了 CBC 加密方法，只不过明文输入换成了 X，密钥换成了 K，初始向量（IV）为全零。输出是最后产生的密文块。这本质上和消息认证码（如第 12 章中介绍的 CBC-MAC 或 CMAC）是一样的。

Prerequisites:
Approved, 128 位 block cipher, CIPH;
Key, K, for the block cipher;

Input:
Nonempty bit string, X, such that LEN(X) is a multiple of 128.
Output:
128 位 block, Y

Steps:
1. Let m = LEN(X)/128.
2. Partition X into m 128 位 blocks $X_1, \cdots, X_m$, so that $X = X_1 \| \cdots \| X_m$
3. Let $Y_0 = [0]^{16}$
4. For j from 1 to m:
5. let $Y_j = \mathrm{CIPH}_K(Y_{j-1} \oplus X_j)$
6. Return Y_m

图 7.13　PRF(X)算法

FF1 加密算法如图 7.14 所示。加粗行对应的是函数 F_K。该算法有 10 轮迭代和以下参数［参见表 7.3(b)］：

Prerequisites:
Approved, 128 位 block cipher, CIPH;
Key, K, for the block cipher;
Base, *radix*, for the character alphabet;
Range of supported message lengths, [minlen .. maxlen];
Maximum byte length for tweaks, maxTeln.

Inputs:
Character string, X, in base radix of length n such that $n \in [minlen..maxlen]$;
Tweak T, a byte string of byte length t, such that $t \in [0..maxTlen]$.

Output:
Character string, Y, such that LEN(Y) = n

Steps:
1. Let $u = \lfloor n/2 \rfloor; v = n - u$.
2. Let $A = X[1..u]; B = X[u+1..n]$.
3. Let $b = \lceil \lceil v \mathrm{LOG}_2(radix) \rceil / 8 \rceil; d = 4\lceil b/4 \rceil + 4$.
4. Let $P = [1]^1 \| [2]^1 \| [1]^1 \| [radix]^3 \| [10]^1 \| [u \bmod 256]^1 \| [n]^4 \| [t]^4$.
5. For i from 0 to 9:

i. **Let** $Q = T \| [0]^{(-t-b-1) \bmod 16} \| [i]^1 \| [\mathrm{NUM}_{radix}(B)]^b$.
ii. **Let** $R = \mathrm{PRF}_K(P \| Q)$
iii. **Let** S **be the first** d **bytes of the following string of** $[d/16]$ **128 位 blocks:**
$R \| \mathrm{CIPH}_K(R \oplus [1]^{16}) \| \mathrm{CIPH}_K(R \oplus [2]^{16}) \| \ldots \| \mathrm{CIPH}_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.
iv. **Let** $y = \mathrm{NUM}_2(S)$.
v. **If** i **is even, let** $m = u$**; else, let** $m = v$.

vi. Let $c = (\mathrm{NUM}_{radix}(A) + y) \bmod radix^m$.
vii. Let $C = \mathrm{STR}^m_{radix}(c)$.
viii. Let $A = B$.
ix. Let $B = C$.
6. Return $Y = A \| B$.

图 7.14　FF1 算法（FFX[Radix]）

- radix $\in [2..2^{16}]$
- $\text{radix}^{\text{minlen}} \geqslant 100$
- minlen $\geqslant 2$
- maxlen $< 2^{32}$ 当最大的基数值是 2^{16} 时，存储整数值的最大位长度是 16×2^{32} 位。当最大的基数值是 2 时，存储整数值的最大的 bite 长度是 2^{32} 位。
- maxTlen $< 2^{32}$

加密函数的输入是一个长度是 n 的字符串 X 和一个长度为 t 的 tweak。tweak 的选择是灵活的，它甚至可以是一个空串。输出是长度为 n 的字符串。下面将会对函数的每一步进行详细的讲解。

（1），（2） 输入 X 将会被分割成两个子串 A 和 B 。如果 n 是偶数， A 和 B 的长度是相同的。否则 B 的长度比 A 的多一个字符的长度。

（3） 表达式 $\lceil v\text{LOG}_2(\text{radix})\rceil$ 的值等于解码 B 需要的位数， B 的长度是 v 个字符。将 B 作为一个字节串进行解码， b 是解码过程中的字节数。 d 的定义可以保证 Feistel 轮函数的输出至少比对 B 进行如上所述的解码而得到的结果多出 4 个字节。这将缩小（5.vi）中进行的模块缩小的权。

（4） P 是一个长度为 128 位（16 字节）的块，同时也是基数、 u 、 n 和 t 的函数。它将作为输入明文的第一块输入到（5.ii）中的 CBC 加密模式中，它的主要作用是增强安全性。

（5） 加密算法的 10 轮循环。

（5.i） tweak、 T 、子串、 B 和轮索引（ i ）都将被解码成一个二进制串（ Q ）它的长度是一个或者多个 128 位块的长度。为了理解这一步，首先需要注意 $\text{NUM}_{\text{radix}}(B)$ 产生了一个数字串，它其实是对 B 在基 radix 下进行的表示。这个数字串如何格式化和存储并不在这个标准的考虑范围之内。之后 $[\text{NUM}_{\text{radix}}(B)]^b$ 将 B 的数值转化为一个长度为 b 字节的二进制数字串。我们可以使 T 的长度是 t 字节，并将轮索引（ i ）存储到一个字节中。这将会产生一个长度为 $(t+b+1)$ 的串。之后再给这个串加上一个长度为 $z=(-t-b-1)\text{mod}16$ 的串。由取模运算的运算规则可知， $(z+t+b+1)\text{mod}16=0$ ，因此 Q 的长度和一个或者多个 128 位块的长度相等。

（5.ii） P 和 Q 的连接将被作为输入，输入到伪随机函数 PRF 中，该函数将产生一个长度为 128 位的输出 R 。这个函数是 Feistel 轮函数的伪随机的核心函数。它将抽取 B_i 的位（如图 7.12 所示）。

（5.iii） 这一步将压缩或者扩展 R ，使得 S 的长度为 d 字节。如果 $d \leqslant 16$ 字节，那么 R 将变为 R 的前 16 个字节。否则， R 将连接用逐次产生的常数异或 R 的结果进行逐次加密，之后再将这些结果连接到 R ，最后产生一个最短的由 16 字节块组成的字符串，这个字符串的长度将小于等于 d 字节。

（5.iv） 这一步将对对 B_i 进行抓取产生的结果进行转换，以使其的格式有利于和 A_i 进行结合。在这一步， d 字节的字符串将被转化成二进制的数字串。将转化后的结果记为二进制串 y 。

（5.v） 这一步将确定长度为 m 的字符串， m 并将其作为轮输出的 B 部分的长度一致。如果是偶数轮，那么长度是 u 个字符。如果是奇数轮，那么长度是 v 个字符。

（5.vi） 这一步将对 A 和 y 进行模 radix^m 加运算。这将压缩求和的结果，并可以用 m 个字节存储结果 c 。

（5.vii）这一步将 c 用 m 个字节的字符串表示，并将结果存到 C。

（5.viii,5.ix）这两步将前一轮的 B 赋给 A，再将 C 赋给 B，到此一轮循环结束。

（6）在最后一轮，返回 A 和 B 连接后的结果。

解释清楚 FF1 算法中的 NUM 函数是十分值得的。NUM 在给定基的情况下，将一个串转换成一系列整数。在步骤（5.i）中，B 是基 radix 下的字符串，所以 $\mathrm{NUM}_{\mathrm{radix}}(B)$ 将 B 转换成一个整数，并被存储为一个字节串，非常有利于步骤（5.ii）的加密运算。在步骤（5.iv）中，S 是加密函数输出的字节串，它可以被转换成位串，因此 $\mathrm{NUM}_2(S)$ 又将 S 变成了整数。

最后，例子很好地按顺序介绍了变量 d。使 radix = 26，并使 $v = 30$ 个字符。之后令 b 和 d 分别为 18 字节和 24 字节。步骤（5.ii）产生了一个 16 字节的输出 R。我们希望从输出中截取 b 字节以使其大小和输入相对，所以 R 需要连接一些字符。步骤（5.iii）将一些随机位连接到 R 而不是简单的全零位。所以，步骤（5.iv）的结果就是一个完全随机的大于 radix^m 的数字。随机的连接消除了始终连接造成的安全隐患。

FF2 算法　FF2 在被作为建议的 FPE 模型提交到 NIST 时被命名为 VAES3[VANC11]。加密算法如图 7.15 所示。加粗行对应于函数 F_K。函数具有下面的参数：

- $\mathrm{radix} \in [2..2^8]$
- $\mathrm{tweakradix} \in [2..2^8]$
- $\mathrm{radix}^{\mathrm{minlen}} \geqslant 100$
- $\mathrm{minlen} \geqslant 2$
- $\mathrm{maxlen} \leqslant 2\lfloor 120/\mathrm{LOG}_2(\mathrm{radix}) \rfloor$，如果 radix 是二的阶乘。对于最大的 radix 值 2^8，$\mathrm{maxlen} \leqslant 30$；对于最小的 radix 值 2，$\mathrm{maxlen} \leqslant 240$。在两种情况中，存储一个整数的最大位长度都是 240 位（30 字节）。
- $\mathrm{maxlen} \leqslant 2\lfloor 98/\mathrm{LOG}_2(\mathrm{radix}) \rfloor$，如果 radix 不是二的阶乘。对于最大的 radix 值 255，$\mathrm{maxlen} \leqslant 24$；对于最小的 radix 值，$\mathrm{maxlen} \leqslant 124$。
- $\mathrm{maxTlen} \leqslant \lfloor 104/\mathrm{LOG}_2(\mathrm{tweakradix}) \rfloor$。对于最大的 tweakradix 值 2^8，$\mathrm{maxTlen} \leqslant 13$。

对于 FF2 而言，明文字母表和 tweak 的字母表是不同的。FF2 的前两步和 FF1 相同，都对 v、u、A 和 B 的值进行了设置。

FF2 的处理流程包含以下步骤：

（3）P 是一个 128 位（16 字节）的块。如果给定一个 tweak，那么 P 就是基数、t、n 和 13 字节的数字 tweak 的函数。如果没有给定 tweak，即 $t = 0$，那么 P 就是基数和 n 的函数。P 的作用就是为第（4）步的加密生成加密密钥。

（4）J 是输入密钥为 K 明文为 P 时的加密结果。

（5）加密运算有 10 轮迭代。

（5.i）B 将被转换成一个 15 字节的数字，在它的前面连接轮数来产生 16 字节的块。

（5.ii）用密钥 J 加密 Q，产生的结果赋给 Y。

剩下的步骤和 FF1 基本一致。FF2 和 FF1 最大的不同在于参数如何被应用到块 F_K 中的加密算法。在两种情况中，加密算法都不是简单地利用密钥 K 对 B 进行加密。对于 FF1，B 将结合 tweak、轮数、t、n、u 和基数来产生多个 16 字节的块。之后在密钥为 K 的情况下使用 CBC 加密算法来产生 16 字节的输出。对于 FF2，除了 B 之外的所有参数都将被利用来产生一个 16 字节的块，并用密钥 K 对它进行加密来产生一个新的密钥 J。J 随后被用来对 B 进行 one-block 加密。

Approved, 128 位 block cipher, CIPH;
Key, K, for the block cipher;
Base, *tweakradix*, for the tweak character alphabet;
Range of supported message lengths, [*minlen* .. *maxlen*];
Maximum supported tweak length, *maxTeln*.

Inputs:
Numeral string, X, in base *radix*, of length n such that $n \in [minlen..maxlen]$;
Tweak numeral string, T, in base *tweakradix*, of length t, such that $t \in [0..maxTlen]$.

Output:
Numeral string, Y, such that $\text{LEN}(Y) = n$.

Steps:
1. Let $u = \lfloor n/2 \rfloor; v = n - u$.
2. Let $A = X[1..u]; B = X[u+1..n]$.
3. If $t > 0$, $P = [radix]^1 \| [t]^1 \| [n]^1 \| [\text{NUM}_{tweakradix}(T)]^{13}$;
   else $P = [radix]^1 \| [0]^1 \| [n]^1 \| [0]^{13}$.
4. Let $J = \text{CIPH}_K(p)$.
5. For i from 0 to 9:

   i. Let $Q \leftarrow [i]^1 \| [\text{NUM}_{radix}(B)]^{15}$
   ii. Let $Y \leftarrow \text{CIPH}_J(Q)$
   iii. Let $y \leftarrow \text{NUM}_2(Y)$
   iv. If i is even, let $m = u$; else, let $m = v$.

   v. Let $c = (\text{NUM}_{radix}(A) + y) \bmod radix^m$.
   vi. Let $C = \text{STR}^m_{radix}(c)$.
   vii. Let $A = B$.
   viii. Let $B = C$.
6. Return $Y = A \| B$.

图 7.15 FF2 算法（VAES3）

FF2 算法揭示了为什么限定最大长度的原因。在第 3 步中，P 将基数、tweak 长度、数字串长度和 tweak 加入了计算。正如 AES 的输入一样，P 的长度被限定在 16 字节。当最大基数值是 2^8 时，基数可以被存储在一个字节的空间中（字节值 0 对应于数值 256）。串长度 n 和 tweak 长度 t 都可以方便地存储到一个字节中。这就导致最多只有 13 个字节来存储 tweak 的值的情况发生。由于存储 tweak 需要的位数为 $\text{LOG}_2(\text{tweakradix}^{\text{Tlen}})$，因此这就导致 $\text{maxTlen} \geqslant \lfloor 104 / \text{LOG}_2(\text{tweakradix}) \rfloor$。同理，步骤（5i）将 B 和轮数作为输入输入到 AES，因此只剩下 15 字节来解码 B，所以长度必须要小于等于 $\lfloor 120 / \text{LOG}_2(\text{radix}) \rfloor$。参数 maxlen 对应于整个块，所以有 $\text{maxlen} \geqslant 2\lfloor 120 / \text{LOG}_2(\text{radix}) \rfloor$。

当基数的长度不是 2 的阶乘时，对 maxlen 会有更多的限制。正如在文献[VANC11]中解释的，当基数的长度不是 2 的阶乘时，模运算（$y \bmod \text{radix}^m$）的结果的格式将和预期的输出空间不一致，这将导致一个密码弱点产生。

FF3 算法 FF3 被作为推荐的 FPE 模式提交到 NIST 时被命名为 BPS-BC[BRIE10]。加密算法如图 7.16 所示。加粗的行对应于函数 F_K。该算法是有下列参数：

- $\text{radix} \in [2..2^{16}]$
- $\text{radix}^{\text{minlen}} \geqslant 100$
- $\text{minlen} \geqslant 2$

- maxlen $\leqslant 2\left\lfloor \mathrm{LOG}_{\mathrm{radix}}(2^{96}) \right\rfloor$。对于最大的基数值 2^{16}，有 maxlen $\leqslant 12$；对于最小的基数值 2，有 maxlen $\leqslant 192$。在两种情况中，存储整数值 X 的最大位长度均是 192 位。
- Tlen = 64

Approved, 128 位 block cipher, CIPH;
Key, K, for the block cipher;
Base, radix, for the character alphabet such that $radix \in [2..2^{16}]$;
Range of supported message lengths, [minlen .. maxlen], such that $minlen \geqslant 2$ and $maxlen \leqslant 2\left\lfloor \mathrm{LOG}_{radix}(2^{96}) \right\rfloor$.

Inputs:
Character string, X, in base radix of length n such that $n \in [minlen..maxlen]$;
Tweak bit string, T, such that LEN(T) = 64.

Output:
Character string, Y, such that LEN(Y) = n

Steps:
1. Let $u = \lceil n/2 \rceil; v = n - u$.
2. Let $A = X[1..u]; B = X[u+1..n]$.
3. Let $T_L = T[0..31]$ and $T_R[32..63]$.
4. For i from 0 to 7:
   i. If i is even, let $m = u$ and $W = T_R$, else let $m = v$ and $W = T_L$.
   ii. Let $P = \mathrm{REV}([\mathrm{NUM}_{radix}(\mathrm{REV}(B))]^{12}) \| [W \oplus \mathrm{REV}([i]^4)]$.
   iii. Let $Y = \mathrm{CIPH}_K(P)$.
   iv. Let $y = \mathrm{NUM}_2(\mathrm{REV}(Y))$.
   v. Let $c = (\mathrm{NUM}_{radix}(\mathrm{REV}(A)) + y) \bmod radix^m$.
   vi. Let $C = \mathrm{REV}(\mathrm{STR}^m_{radix}(c))$.
   vii. Let $A = B$.
   viii. Let $B = C$.
5. Return $A \| B$.

图 7.16　FF3 算法（BPS-BC）

FF3 的处理过程如下：

（1），（2）　输入 X 将会被分割成两个子串 A 和 B。如果 n 是偶数，A 和 B 的长度是相同的。否则 A 的长度比 B 的多一个字符的长度，而 FF1 和 FF2 算法则恰恰相等。

（3）　tweak 将被分割成左右两个个长度相等的部分，分别记为 T_L 和 T_R，长度都是 32 位。

（4）　加密循环将进行 8 轮迭代。

（4.i）　和 FF1 和 FF2 中的作用一样，这一步将确保字符串输出的长度 m 和轮输出的 B 部分的长度一致。这一步也将决定步骤（4.ii）中的 W 的值是 T_L 还是 T_R。

（4.ii）　颠倒 B 的位顺序，然后 $\mathrm{NUM}_{\mathrm{radix}}(\mathrm{REV}(B))$ 将生成一个以 radix 为基长度为 12 字节的数字串；结果的位顺序将再次被颠倒。轮数的 32 位解码结果将被存储在一个 4 字节的单元中，在被存储前将被倒置并且异或 W。通过连接上述得到的连个结果就可以得到一个长度为 16 字节的块 P。

（4.iii）　用密钥 K 来加密 P 并得到结果 Y。

（4.iv）　这一步和 FF1 的步骤（5.iv）相同，唯一的区别是在对 Y 进行变换之前，要颠倒它的位顺序。

（4.v）　将 A 倒置的值模 radix^m 加上 y。这将压缩加法运算的结果，使结果能被存储在 m 个字符中。

（4.vi）将 c 转换成数字串 C。

剩下的步骤和 FF1 相同。

7.9 关键术语、思考题和习题

关键术语

密文挪用	计数器模式（CTR）	输出反馈模式（OFB）	分组密码工作模式（CFB）
电码本模式（ECB）	三重 DES（3DES）	密文分组链接模式（CBC）	中间相遇攻击
可调分组密码	密文反馈模式（CFB）	时变值	XTS-AES 模式

思考题

7.1 什么是三重加密？

7.2 什么是中间相遇攻击？

7.3 在三重加密中用到多少个密钥？

7.4 为什么 3DES 的中间部分采用了解密而不是加密？

7.5 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？

习题

7.1 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 7.11 给出了两种可能，都是遵循 CBC 模式的定义。你将选择哪一个？

（a）从安全性角度考虑。

（b）从性能上考虑。

7.2 仅能够使用 3DES 芯片和一些异或 XOR 函数，你能对图 7.17 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。

7.3 对 3DES 的 Merkle-Hellman 攻击是这样的：首先令 A=0［参见图 7.1(b)］，然后，对 K_1 的所有 2^{56} 个取值，找出令 A=0 的明文。请描述算法的剩下部分。

7.4 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，则只有相应的明文分组会有影响。然而，在 CBC 模式中，这种错误具有扩散性。比如，图 7.4 中传输 C_1 时发生的错误将会影响明文分组 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。则这个错误要扩散至多少个密文分组？对接收者解密后的结果有什么影响？

7.5 在 CBC 模式里有可能对多个明文分组进行并行加密吗？解密呢？

7.6 CBC-Pad 是用于 RC5 密码的一种分组密码工作模式，其可以用于任何分组密码。CBC-Pad 能够处理任意长度的明文。密文至多比明文长一个分组的长度。填充的作用是保证输入明文是分组长度的整倍数。假设原始明文是整数个字节。明文在尾部填充 1 到 *bb* 字节，这里 *bb* 为分组的字节长度。填充字节值全都相同并将其值设为填充的字节数。例如如果有 8 字节填充，则每个字节为 00001000。为什么不允许 0 字节填充？即如果原始明文是分组大小的整数倍，为什么不省去填充？

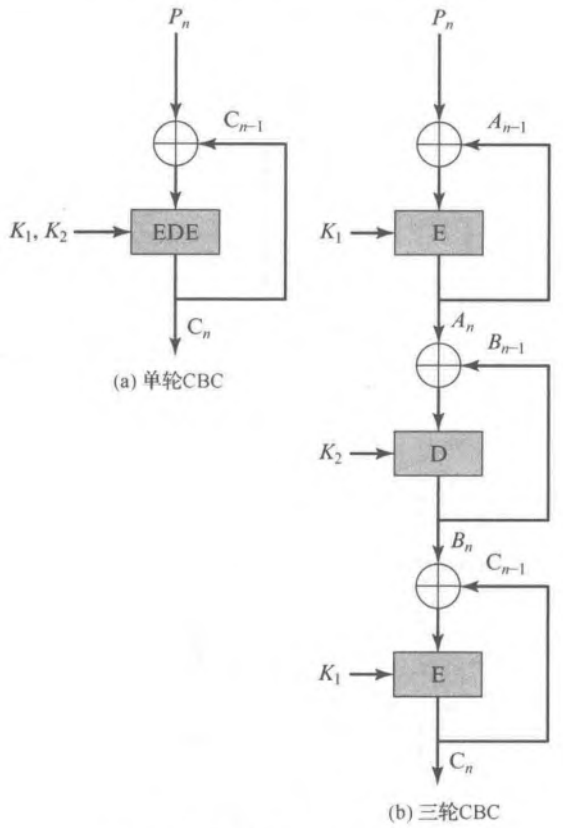


图 7.17　将 3DES 用于 CBC 模式

7.7　对于 ECB，CBC 和 CFB 来说，明文必须是一个或多个完整数据分组（或者对于 CFB 模式来说，数据段）组成的序列。即对于这三种模式，明文的总位数必须是分组（或段）长度的正整数倍。在需要的时候，一种常用的填充方法是先填 1，然后后面跟着填 0，也可能没有，直到填满最后一个分组。对于发送者来说填充每一个消息是好习惯，包括那些最后一个分组已经满的消息。不需要填充却还要包含一个填充分组的动机是什么？

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

7.9　在讨论 OFB 式，提到如果两个消息在相同的位置有相同的明文分组，有可能恢复出对应的 O_i。请给出计算过程。

7.10　在讨论 CTR 模式时，提到如果用给定的计数器值加密的明文分组内容已知，那么从关联的密文分组很容易求出加密函数的输出。请给出计算过程。

7.11　填充不总是合适的。例如，我们希望使用相同的内存缓冲区（明文起始时存于此处）来存储加过密的数据，这时密文必须和原始的明文长度相同。密文挪用模式（CTS）就是满足这样要求的一种工作模式。图 7.18(a)为该模式的实现过程。

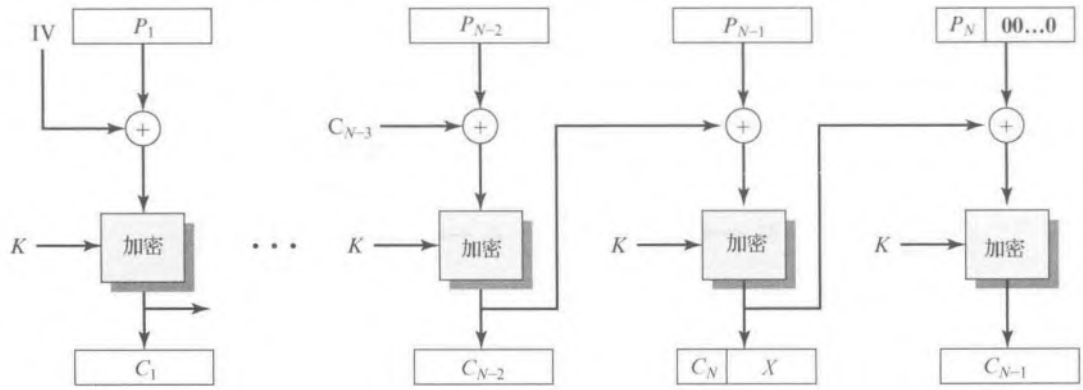
（a）解释 CTS 是如何工作的。

（b）描述如何解密 C_{n-1} 和 C_n。

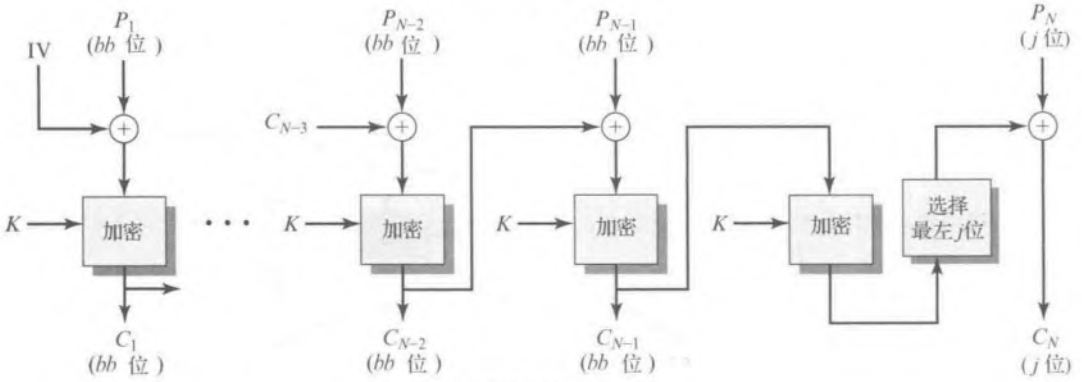
7.12　图 7.18(b)给出了 CTS 的一个替换方案，使得当明文不是分组长度的整数倍时产生的密文长度与明文长度相等。

（a）解释该算法。

（b）解释为何 CTS 比图 7.18(b)中的方法更可取。



(a) 密文窃取模式



(b) 替换方法

图 7.18 明文不是分组长度整数倍的分组密码工作模式

7.13 画一个和图 7.8 类似的 XTS-AES 模式图。

7.14 在不转换成二进制然后对位进行计数的情况下，解决下面的问题。然后和书中给出的将一个字符串编码成一个整数的公式相比，对特定的基数，反之亦然（提示：对每一个数字，考虑比它小的最接近的指数和比它大的最接近的指数）。

（a）编码下面的整数需要多少位（数字的下标表示对应整数的基数）？

（i）8191_{10}

（ii）8192_{10}

（iii）65535_{10}

（iv）65536_{10}

（v）65535_{16}

（vi）65536_{16}

（vii）$34F7_{16}$

（viii）13559_{10}

（b）上面的（a）中的数字需要多少字节来表示？

7.15 （a）在基数是 26 的情况下，写出下面字符串的数字编码 X，并跟着一些数字（即数字串的长度）。

（i）“key”

（ii）“random”

（iii）“or”

（iv）“digit”

（b）对（a）中的没一个问题，计算 $x = \mathrm{NUM}_{26}(X)$

（c）对（b）中计算出的 x，确定它的字节形式[x]。

（d）比（b）中的每一个数字字符串都大的 26 的最小指数是多少？

（e）在（a）中的每一种情况下的数字字符串的长度有联系吗？如果有，是什么？

7.16　参考算法 FF1 和 FF2。

（a）对每一算法中的步骤 1，$u \leftarrow \lfloor n/2 \rfloor$和 $v \leftarrow \lceil n-u \rceil$。结果表明对任意的 3 个整数 x，y 和 n：

（i）如果 $x = \lfloor n/2 \rfloor$和 $y = \lceil n-x \rceil$，则

（ii）要么 $x = n/2$，要么 $y = (n-1)/2$。

（iii）要么 $y = n/2$，要么 $x = (n+1)/2$。

（iv）$x \leqslant y$（在什么条件下有 $x = y$？）。

（b）在 u 和 v 分别是一个字符串左右一半的字符串的长度的情况下，上面子问题（iii）的结果有什么重大意义？

7.17　在算法 FF1 中的步骤 3 中，b 和 d 代表什么？这些数量的基本测量单元是（位，字节，数字，字符）？

7.18　在算法 FF1，FF2 和 FF3 的输入中，为什么给定基数的范围如此重要？例如，为什么在算法 FF2 中要求 $x \in [0..2^8]$或在算法 FF2 中要求 $x \in [2..2^{16}]$？

编程习题

7.19　使用如下的一种密码来编写密码分组链接模式的加密和解密软件：模 256 仿射，Hill 模 256，S-DES，DES。S-DES 的测试数据：使用二进制初始向量 1010 1010，使用二进制密钥 01111 11101 加密二进制明文 0000 0001 0010 0011，得出的结果应该是 1111 0100 0000 1011。能够相应地解密。

7.20　使用如下的一种密码来编写 4 位密文反馈模式的加密和解密软件：模 256 加，模 256 仿射，S-DES。或者 8 位密文反馈模式的加密和解密软件，使用如下密码：2×2 Hill 模 256。S-DES 的测试数据：使用二进制初始向量 1010 1011，使用二进制密钥 01111 11101 加密二进制明文 0001 0010 0011 0100，得出的结果应该是 1110 1100 1111 1010。能够相应地解密。

7.21　使用如下的一种密码来编写计数器模式的加密和解密软件：模 256 仿射，Hill 模 256，S-DES。S-DES 的测试数据：使用计数器从 0000 0000 开始，使用二进制密钥 01111 11101 加密二进制明文 0000 0001 0000 0010 0000 0100，得出的结果应该是 0011 1000 0100 1111 0011 0010。能够相应地解密。

7.22　实现对于 3 轮 S-DES 的差分攻击。

第 8 章　伪随机数的产生和流密码

学 习 目 标

通过本章的学习后应具备以下能力：

- 解释随机数的随机性和不可预测性
- 理解真随机数发生器、伪随机数发生器和伪随机函数之间的差异
- 简要介绍伪随机数发生器的需求
- 解释一个分组密码如何能构建一个伪随机数发生器
- 简要介绍流密码和 RC4
- 解释偏差的意义

一个重要的密码函数是具有强密码学意义的伪随机数发生器。伪随机数发生器（PRNG）在许多密码和安全应用里都有使用，包括密钥产生和加密。本质上，只有两种产生随机位或随机数的策略。第一种策略到现在还是密码学应用里面的主流方法，它通过一个算法确定地产生位。这种随机数发生器被称为伪随机数发生器（PRNG）或确定随机位发生器（DRBG）。另一种策略是利用一些可以产生随机输出的物理源来产生随机位串。这种随机数发生器被称为真随机数发生器（TRNG）或非确定随机位发生器（NRBG）。

本章先看一下 PRNG 的一些基本原则并把这些原则和真随机数发生器[①]进行比较。接下来研究一些常见的 PRNG，包括基于对称分组密码的 PRNG。然后，讨论了基于 PRNG 的对称流密码。接着本章研究了最为重要的流密码 RC4。

本章首先分析伪随机数发生器（PRNG）的基本原则。接下来研究一些常见的 PRNG，包括基于对称分组密码的 PRNG。本章还讨论了基于 PRNG 的对称流密码，包括最为重要的流密码 RC4。然后，介绍一种特殊的产品——英特尔数字随机发生器。

本章涉及 4 份重要的 NIST 文档。

- SP 800-90A（Recommendation for Radom Number Generation Using Deterministic Radom Bit Generators，January 2012）是 DRNG 的文档。
- SP 800-90B（Recommendation for Entropy Sources Used for Random Bit Generation，August 2012）中包括熵源（一种可以产生不可预测的随机性和 NRNG 的设备）的标准。
- SP 800-90C（Recommendation for Radom Bit Generator (RBG) Constructions, August 2012）讨论了怎样将 90B 中的熵源和 90A 中的 DRNG 结合起来，为密码学应用产生更高不可预测性的位串。
- SP 800-22（A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, April 2010）讨论了 NRBG 和 DRBG 的选择和测试。

① 术语的注解。一些标准，特别的是 NIST 以及 ANSI，认为 TRNG 是一种非确定性的随机数发生器（NRNG），而 PRNG 是一种确定定性的随机数发生器（DRNG）。

这些规范对美国乃至全世界的随机数发生器工业产生了重要的影响。

8.1　随机数产生的原则

在网络安全的各种应用里，随机数在加密算法里扮演重要的角色。本节里，我们简短回顾一下随机数在密码和网络安全里的使用情况，重点是伪随机数的产生原则。

8.1.1　随机数的使用

大量的基于密码学的网络安全算法和协议都使用了二进制随机数。例如：

- 密钥分发和相互认证方案，如第 14 章和第 15 章所讨论的。在那些方案里，通信的双方通过交换信息共同合作分发密钥和/或互相认证对方。许多时候需要使用时变值进行握手以防止重放攻击。时变值使用随机数可以防止攻击者判断或者猜测出时变值。
- 会话密钥的产生。本书内我们将看到大量的协议，需要为对称加密产生一个短时间内使用的秘密密钥。这个密钥通常称为会话密钥。
- RSA 公钥加密算法中密钥的产生（详见第 9 章）。
- 用于对称流密码加密的位流的产生。

这些应用对随机数序列产生提出了两个不同且未必兼容的要求：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：

- **分布均匀性**　序列中的位分布应是均匀的，即 0 和 1 出现的频率大约相等。
- **独立性**　序列中任何子序列不能由其他子序列推导出。

尽管有一些成熟的测试来判断位序列是否符合某个特定的分布，比如均匀性分布，然而还没有某种方法可以表明一个序列的独立性好。相反，却有很多测试可以证明一个序列不具有独立性。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。

密码学中算法的设计经常使用这种“似乎随机”的数序列。如对第 9 章将要讨论的 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 $\sqrt{N}$ 以内的奇数。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，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 $\sqrt{10^{150}}$ 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、简短、不确定性方法来取得某种可信程度的解。

不可预测性　在相互认证、会话密钥生成以及流密码之类的应用中，对随机数的要求不能仅仅停留在序列是统计随机的，而是要求随机数序列是不可预测的。所谓的“真随机数序列”是每个数都统计独立于其他数，因而不可预测。不过，马上就可以看到，真正的随机数序列很少用；一般看上去随机的随机数序列是由算法产生的。此时，必须要注意敌手不能够从先前的随机数推导出后面的随机数。

8.1.2　TRNG，PRNG 和 PRF

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。

你也许对这种把确定性算法生成的序列用于随机数的观念感到有点不安。但是，只要我们不追求所谓哲学上的完美性，这种方法的确有效。也就是说，在多数情况下，伪随机数发生器能如我们需要的那样工作。“如”很可惜是主观的，但伪随机数被广泛接受。在统计应用程序也会用相同的原理：统计员对人口采样并假设所得结果几乎和对全体人口进行测量的结果是一样的。

图 8.1 把一个真随机数发生器和两种伪随机数发生器进行了对比。TRNG 把一个很随机的源作为输入，这个源常称为熵源，将会在 8.6 节讨论这种随机源。本质上，熵源是从计算机的物理环境抽取的，可能包括键盘敲击时间模式，磁盘的电活动，鼠标移动，系统时间的瞬时值等。源或诸源的组合作为算法的输入，产生随机的二元输出。TRNG 也许仅仅是把模拟信号源转化为二元输出。TRNG 也可能会做额外的处理以消除源里的不平衡。这将在 8.6 节讨论。

相反，PRNG 取一个固定值，称为种子作为输入，用一个确定性的算法产生位输出序列。通常，如图显示有反馈路径，由此算法的部分结果反馈作为输入，而其他部分作为输出位。需要注意的是输出位流仅仅由输入值决定，所以知道算法和种子的敌手可以重现整个位流。

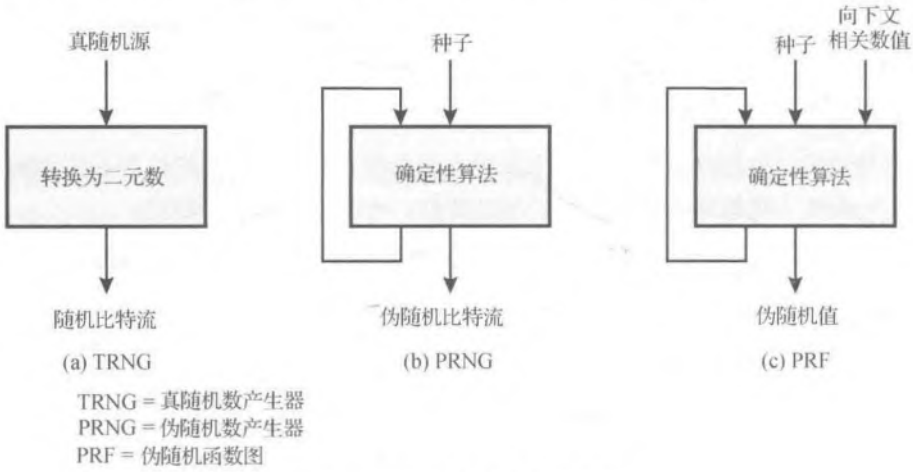


图 8.1 随机和伪随机数发生器

图 8.1 显示了两款基于应用的不同形式的 PRNG。

- **伪随机数发生器** 用于生产不限长位流的算法称为 PRNG。不限长位流的通常应用是作为对称流密码的输入，如同 8.4 节所讨论的那样，也可参见图 4.1(a)。
- **伪随机函数（PRF）** PRF 用于产生固定长度的伪随机位串。对称加密密钥和时变值就是例子。通常 PRF 的输入为种子加上一些上下文相关的特定值，如用户 ID 或应用 ID。本书中将看到许多 PRF 的例子，尤其是第 17 章和第 18 章。

除了产生位的数量外，PRNG 和 PRF 之间没有差别。这两个应用可以使用相同的算法。两者都需要种子，都必须具有随机性和不可预测性。而且，PRNG 应用可能也需要使用上下文相关的输入。接下来，我们对这两个应用不做区别。

8.1.3 对 PRNG 的要求

当 PRNG 或 PRF 用于密码学应用时，基本的要求是不知道种子的敌手不能决定伪随机串。例如，如果伪随机位流用于流密码，那么知道伪随机位流将导致敌手从密文恢复明文。类似地，我们希望保护 PRF 的输出值。对于后者，考虑如下场景。128 位的种子加上一些上下文相关的值用于产

生 128 位的密钥，该值接着又用于对称加密。在通常的环境下，128 位的密钥是能抗暴力攻击的。然而，如果 PRF 不能产生足够随机的 128 位输出值，敌手有可能降低穷举空间并成功进行暴力攻击。

对 PRNG 和 PRF 输出保密性的这个通用要求导致对随机性、不可预测性以及种子的特性的特定要求。我们依次来看一下。

随机性　就随机性而言，对 PRNG 的要求是生成的位流尽管是确定性的但要显示随机。没有单个的测试可以判定一个 PRNG 生成的数据具有随机的特性。所能做的就是对 PRNG 进行一系列的测试。如果 PRNG 在多次测试的基础上展现了随机性，那么可以认为它满足随机性的要求。NIST SP 800-22（密码学应用中的随机数和伪随机数发生器统计测试工具集）规定这些测试应该建立如下三个特征。

- **均匀性**　在产生随机或伪随机位序列的任何点，0 或 1 出现的量大约相等，即每个出现的概率恰好为 1/2。0（或 1）出现的次数为 $n/2$，其中 n 为序列的长度。
- **可伸缩性**　用于序列测试的任何测试也可以用于从序列中抽取的子序列的测试。如果序列是随机的，那么任何从中抽取的子序列也应该是随机的。因此，任何抽取的子序列都应该通过随机性测试。
- **一致性**　对于所有初始值（种子），发生器的行为必须具有一致性。基于单个种子产生的输出测试 PRNG，或者基于单个物理源的输出测试 TRNG 都是不充分的。

SP 800-22 列出了 15 种单独的随机性测试。对这些测试的理解需要统计分析的知识，所以此处并不准备做技术上的描述。相反，为了使测试不太枯燥，如下列出三个测试以及每个测试的目的。

- **频率测试**　这是最基本的测试，任何测试工具集都必须包含。这个测试的目的是判断序列中 0 和 1 的个数是否和真随机序列的期望值大约相同。
- **游程测试**　本测试的焦点是序列里游程的总量，所谓游程是相同位构成的不间断序列，其前后的位取与游程位不同的值。游程测试的目的是判断各种长度的 0 和 1 游程的数量是否符合随机序列的期望值。
- **Maurer 通用统计测试**　该测试的重点是匹配模式（与压缩序列长度相关的一种度量）间的位数。测试的目的是检测序列是否能大幅度压缩而不损失信息。一个能大幅度压缩的序列被认为非随机的。

不可预测性　随机数流应该具有两种形式的不可预测性：

- **前向不可预测性**　如果不知道种子，那么不管知道序列中前面的多少位都无法预测序列中的下一位。
- **后向不可预测性**　从产生的任何值都不能推断出种子值。很明显，种子和该种子产生的任意值之间应该没有相关性；序列的每个元素应该显得像独立随机事件的结果（其概率为 1/2）。

用于随机性测试的测试集也提供了不可预测性测试。如果产生的位流看上去随机，那么不可能从前面的任何位预测某一位或位序列。类似地，如果位序列看上去随机，那么也不可能从位序列推导出种子。即随机序列和固定值（种子）没有相关性。

种子的要求　在密码学应用时，作为 PRNG 输入的种子必须安全。因为 PRNG 是一个确定性算法，如果对手能推导出种子，那么就可以决定输出。因此，种子必须不可预测。事实上，种子本身必须是随机数或伪随机数。

典型地，如图 8.2 所示，种子是由 TRNG 产生的。这是 SP 800-90A 推荐的方案。读者也许会问如果可以用 TRNG，为何还需要用 PRNG 呢？如果应用是流密码，那么 TRNG 是不实际的。发送者需要产生和明文一样长的位密钥流，并把密钥流和密文安全传送给接收者。如果使用 PRNG，发送者仅需想法给接收方安全传送流密码的密钥，通常是 54 位或 128 位。

即使是 PRF 应用，即只需要产生有限数量的位，也希望用 TRNG 为 PRF 提供种子，并使用 PRF 的输出而不是直接使用 TRNG。如同 8.6 节解释的，TRNG 可能会产生不平衡的二元序列。PRF 具有使 TRNG 的输出更随机化的效果，从而消除不平衡性。

最后，用于产生真随机数的机制在速率方面也许不能跟得上需要随机位的应用。



图 8.2 输入到 PRNG 的种子的产生

8.1.4 算法设计

PRNG 多年来一直是密码学上的研究主题，为此产生了大量的算法。这些算法可以大体分为两类。

- **特意构造的算法** 这些算法是为了生产伪随机位流而特意或专门设计的。许多算法在许多 PRNG 应用里使用。其他算法则特意为了流密码而设计。后者最重要的例子是 8.5 节描述的 RC4。
- **基于现有密码算法的算法** 密码学算法具有随机化输入的效果。的确，这是此类算法的要求。例如，如果对称分组密码产生的密文具有某些规则性，这就有助于密码分析。因此，密码学算法在 PRNG 里起核心作用。

三种密码学算法常用来产生 PRNG：

- **对称分组密码** 这个方法将在 8.3 节讨论。
- **非对称密码** 用于非对称密码的数论概念也可以用于 PRNG；这种方法在第 10 章讨论。
- **Hash 函数和消息认证码** 这种方法将在第 12 章讨论。

这些方法的任意一种都可以产生密码学意义上强的 PRNG。特意构造的算法可以由操作系统提供用于通用目的。对于那些已经使用密码学算法用于加密或认证的应用来说，把这些代码重用于 PRNG 也是有意义的。因此，所有这些方法都是通用的。

8.2 伪随机数发生器

本节我们要研究用于 PRNG 的两种算法。

8.2.1 线性同余发生器

一个广泛使用的产生伪随机数的方法是由 Lehmer 首先提出的算法[LEHM51]，即线性同余方法。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	初始值或称种子	$0 \leqslant X_0 < m$

随机数序列$\{X_n\}$按下面的迭代式获得：

$$X_{n+1} = (aX_n + c) \bmod m$$

若 m, a, c 和 X_0 都是整数，那么这种方法将产生一个整数序列，且每个整数都满足 $0 \leqslant X_n < m$。

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 $X_0 = 1$。产生序列是{7, 17, 23, 1, 7, …}，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, 5, …}，周期为 8。

m 一般都很大，因此可产生长连串的不同随机数，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，那么，对于 32 位机，m 可以选择接近或等于 2^{31} 的值。

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，即函数在重复之前应该产生 0～m–1 之间的所有数。

T_2：产生的序列应显得随机。

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 p 是素数且 c=0，则 a 的某些取值可以使生成函数的周期为 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，这时的产生函数即为

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1).$$

a 的可能取值超过 20 亿个，但满足上述条件的只有其中很小一部分。a 取值为 7^5=16807 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[LEWI69]，这个发生器使用得最广泛，比起其他 PRNG 其经过了更为全面的测试，尤其适合于统计和仿真方面[JAIN91]。

若乘数和模选择恰当，用线性同余算法产生的随机数序列的统计特性几乎与从集合 1, 2, …, m–1 里随机抽取的序列相当（无放回抽取）。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（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，即使他只知道是采用了线性同余算法，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

$$X_1 = a(X_0 + c) \bmod m$$
$$X_2 = a(X_1 + c) \bmod m$$
$$X_3 = a(X_2 + c) \bmod m$$

由这三个等式可求解出 a、c 和 m。

因此，尽管这种办法用于伪随机数发生器有很多优点，但是最理想的还是要使产生的序列不可重新生产，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，一个方法是每隔 N 个数就以时钟值对 m 取模作为新的种子来产生新的序列。还有一个方法是直接将随机数加上时钟值再对 m 取模。

8.2.2 BBS 发生器

BBS 发生器（参见图 8.3）是产生安全伪随机数的普遍方法，BBS 是三个设计者名字的首字母：

Blum, Blum, Shub。它也许是特意构造算法里密码强度有最强公开证明的一个了。产生过程如下：首先，选择两个大素数 p 和 q，且要求

$$p \equiv q \equiv 3 (\text{mod } 4)$$

该符号在第 4 章中有完整解释，表示 $(p \bmod 4) = (q \bmod 4) = 3$。例如，素数 7 和 11 满足 $7 \equiv 11 \equiv 3 (\text{mod } 4)$。令 $n = p \times q$。接着，选择一个随机数 s，且要求 s 与 n 互素，即 p 或 q 都不是 s 的因子。然后 BBS 按下列算法产生位 B_i 序列：

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$X_i = (X_{i-1})^2 \bmod n$$
$$B_i = X_i \bmod 2$$

因此每个循环都是取的最低有效位。表 8.1 给出了 BBS 运算的一个例子。这里，n=192649=383×503，种子 s=101355。

BBS 被称为密码安全伪随机位发生器（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[①]，对于某输出序列的最初 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。

表 8.1 BBS 伪随机数发生器的一个例子

i	X_i	B_i	i	X_i	B_i
0	20749		11	137922	0
1	143135	1	12	123175	1
2	177671	1	13	8630	0
3	97048	0	14	114386	0
4	89992	0	15	14863	1
5	174051	1	16	133015	1
6	80649	1	17	106065	1
7	45663	1	18	45870	0
8	69442	0	19	137171	1
9	186894	0	20	48060	0
10	177046	0			

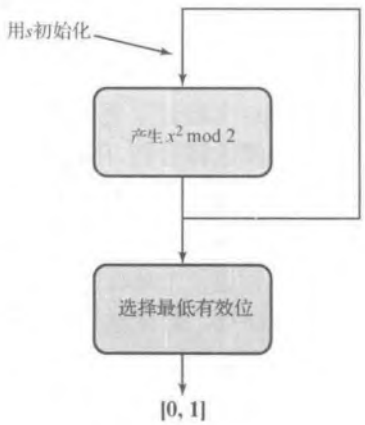


图 8.3 BBS 框图

8.3 使用分组密码的伪随机数产生

构造 PRNG 的常用方法是使用对称分组密码作为 PRNG 机制的核心。对于任意的明文分组，对称分组密码产生一个明显随机的分组输出。即密文里没有规则性或模式可用于推导明文。因此，对称分组密码很适合用来构造伪随机数发生器。

如果使用一个成熟的标准分组密码，如 DES 或 AES，那么：PRNG 的安全特性就得到保证了。而且，许多应用已经在使用 DES 或 AES，所以包含分组密码作为 PRNG 的一部分是简单直接的。

① 阶为 k 的多项式时间算法是指算法的运行时间由阶为 k 的多项式限定。

8.3.1　使用分组密码工作模式的 PRNG

两种使用分组密码构建 PRNG 的方法获得了广泛的接受：CTR 模式和 OFB 模式。SP 800-90A、ANSI 标准 X9.82 以及 RFC 4086 都推荐了 CTR 模式。X9.82 和 RFC 4086 推荐了 OFB 模式。

图 8.4 展示了两种方法。每一种情况里，种子由两部分组成：加密密钥值以及每产生一个随机数分组后都要有更新的 V 值。因此，对于 AES，种子由 128 位的密钥和 128 位的 V 值构成。当为 CTR 模式时，V 的值每加密一次后增加 1。当 OFB 模式时，V 的值更新为前一个 PRNG 分组的值。两种情况下，一次生产一个伪随机位分组（如对于 AES，一次生产 128 位的 PRNG 位）。

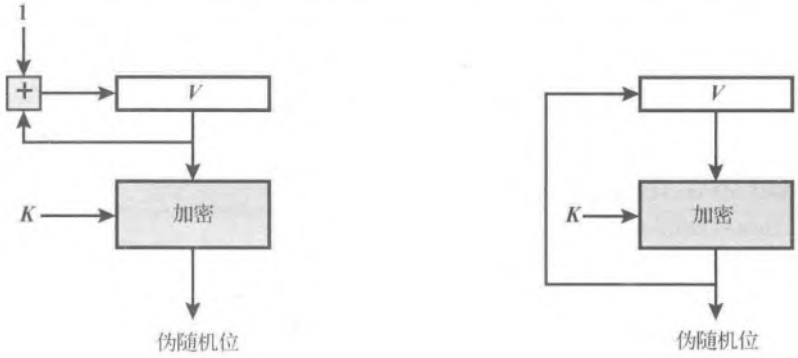


图 8.4　基于分组密码的 PRNG 机制

用于 PRNG 的 CTR 算法总结如下：

```
While (len(temp) <需要的位数量) do
        V=(V+1) mod 2^128
        output_block=E(Key,V)
        temp=temp ‖output_block
```

OFB 算法可以总结如下：

```
While (len(temp) <需要的位数量) do
        V=E(Key,V)
        temp=temp ‖V
```

为了了解这两个 PRNG 的性能，考虑如下的短实验。256 位的随机位序列是从 random.org 处获得的，通过使用三个相互之间可以调节的无线电来获取空气噪声。这 256 位形成种子，如下分配：

密钥:	Cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

256 位的种子里 1 的总数为 124，或者是 48%，很接近理想情况下的 50%。

对于 OFB 的 PRNG，表 8.2 显示了前 8 个输出分组（1024 位），带有两个粗略的安全度量指标。第二列显示了每个 128 位分组里 1 的分数。这对应于 NIST 的一个测试。结果表明输出为 0 和 1 的数量大约相等。第三列显示的是相邻分组匹配位的分数。如果这个数值偏离 0.5 比较大，这就表明分组间有相关，这是一个安全弱点。结果表明没有相关性。

表 8.3 显示了 CTR 模式下使用相同的密钥和 V 值的结果。结果再一次让人满意。

表 8.2 使用 OFB 的 PRNG 的例子结果

输出分组	位 1 所占的分数	和前一分组匹配位所占分数
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

表 8.3 使用 CTR 的 PRNG 的例子结果

输出分组	位 1 所占分数	和前一分组匹配位所占分数
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
f4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eebac658a3911515c3c66	0.47	0.45

8.3.2 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密码学意义上最强的伪随机数发生器之一。许多应用使用了这种方法，包括金融安全应用和 PGP（后者详见第 19 章）。

图 8.5 说明了算法流程，它使用了 3DES 来加密。构成成分如下：

- **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更新。另一个是 64 位的种子，可以被初始化为任意值，并在生成随机数的过程中被更新。
- **密钥** 发生器使用 3 个 3DES 加密模块。3 个加密模块都使用一个相同的 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。

定义以下变量：

DT_i：算法第 i 轮开始时的日期/时间值。

V_i：算法第 i 轮开始时的种子值。

R_i：算法第 i 轮所产生的伪随机数。

K_1, K_2：各阶段算法所用的 DES 密钥。

有

$$R_i = \text{EDE}([K_1, K_2], [V_i \oplus \text{EDE}([K_1, K_2], DT_i)])$$

$$V_{i+1} = \text{EDE}([K_1, K_2], [R_i \oplus \text{EDE}([K_1, K_2], DT_i)])$$

其中，$\text{EDE}([K_1, K_2], X)$代表加密-解密-加密序列加密 X，使用的算法为两个密钥的 3DES。

该方法的密码强度来自几个方面，包括 112 位密钥和 3 个 3DES 共计 9 次 DES 加密。整个方案的输入是两个伪随机数，日期和时间值，一个由发生器产生的种子，且其区别于发生器产生的伪随机数。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 3DES 运算。

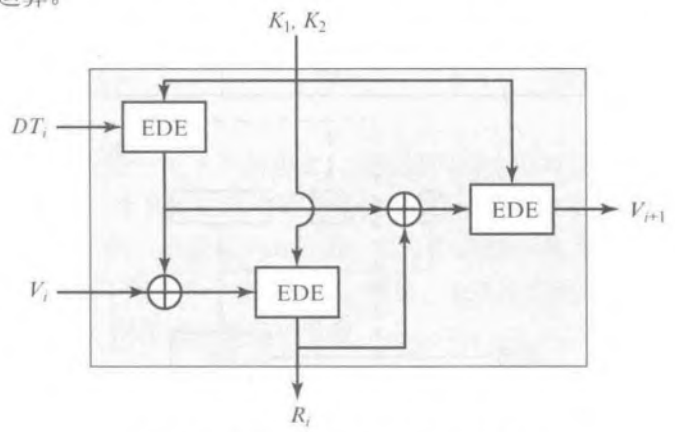


图 8.5　ANSI X9.17 伪随机数发生器

8.3.3　NIST CTR_DRBG

现在我们考虑定义于 NIST SP 800-90 中的基于 CTR 工作模式的 PRNG。这种 RPNG 被称为 CTR_DRBG(counter mode-deterministic random bit generator)。CTR_DRBG 被广泛地实现并且是最近的英特尔处理器芯片的（在 8.6 节讨论）硬件随机数发生器的一部分。

DRBG 假设熵源可提供随机位，熵源通常是基于一些物理源的 TRNG。其他源也是可能的，如果他们能被应用程序测量所需的熵。熵是信息论的概念用来度量不可预测性或随机性，详情参见附录 F。DRBG 使用的加密算法可能是带 3 个密钥的 3DES 或者密钥长度为 128 位、192 位或 256 位的 AES。

与该算法有关的 4 个参数为

- **输出分组长度**（outlen）　加密算法的输出分组的长度。
- **密钥长度**（keylen）　加密密钥的长度。
- **种子长度**（seedlen）　种子是一个位串作为 DRBG 机制的输入。种子确定了 DRBG 的一部分内部状态，它的熵必须能保证 DRBG 安全性。seedlin = outlen + keylen。
- **补种间隔**（reseed_interval）　加密密钥的长度。它是用新种子更新算法前的输出分组的最大数量。

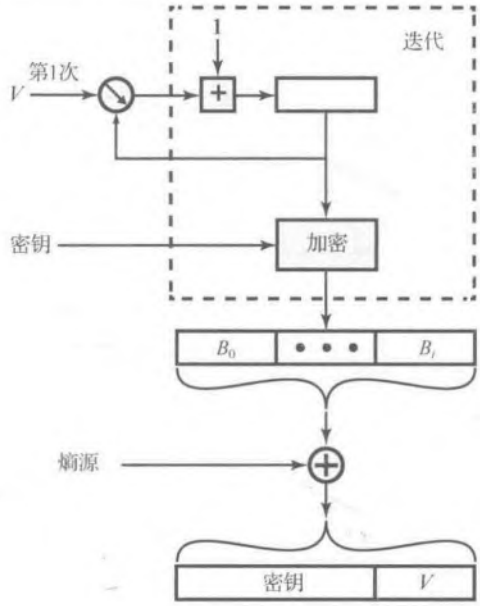
表 8.4 列出了 SP 800-90A 的这些参数的值。

初始化　如图 8.6 所示，CTR_DRBG 由两个主要函数组成。我们首先考虑 CTR_DRBG 如何使用初始化和更新函数初始化［参见图 8.6(a)］。CTR 分组模式需要一个密钥 K 和一个 SP 800-90A 的初试计数器值 V。K 和 V 的组合称为种子。开始 DRGB 操作时需要任意的 K 和 V。如 8.6 节介绍的英特尔数字随机数发生器使用 K=0 和 V=0。这些值是 CTR 工作模式用来产生 seedlen 个位的参数。此外，全部的 seedlen 个位必须是熵源提供的，熵源通常是某种形式的 TRNG。

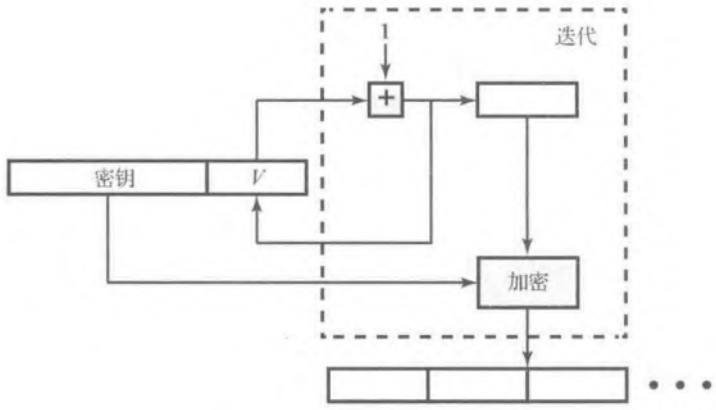
在这些输入下迭代 CTR 模式产生一个输出分组序列，每次加密后 V 加 1。这个过程持续到产生了最后一个 seedlen 位。最左边的 seedlen 输出位异或 seedlen 位从而产生一个新的种子。依次地，种子的最左边 keylen 位产生新密钥，种子的最右边 outlen 位产生新的计数器值 V。

表 8.4 CTR_DRBG 参数

	3DES	AES-128	AES-192	AES-256
outlen	64	128	128	128
keylen	168	128	192	256
seedlen	232	256	320	384
reseed_interval	$\leqslant 2^{32}$	$\leqslant 2^{48}$	$\leqslant 2^{48}$	$\leqslant 2^{48}$



(a) 初始化和更新函数



(b) 生成函数

图 8.6 CTR_DRBG 函数

产生 一旦获取 Key 和 V 的值后 DRBG 就进入产生阶段且能生成伪随机位，每一次一个分组[参见图 8.6(b)]。迭代加密函数生成期望的伪随机位，每次迭代使用相同的密钥，计数器值 V 每次迭代加 1。

更新 为了提高安全性，PRNG 产生的位数应该受限。CTR_DRGB 使用参数 reseed_interval 来

设置那个限制。在产生阶段，reseed counter 初始化为 1 且每次迭代都增加。当 reseed counter 达到 reseed_interval 时引入更新函数［参见图 8.6(a)］。更新函数和初始化函数相同。在更新阶段，最后被生成函数使用的 Key 和 V 值作为更新函数的输入参数。更新函数使用熵源的 seedlen 新位生成一个新种子（Key, V）。生成函数能保留伪随机位。注意到更新函数的结果会改变生成函数使用 Key 和 V 值。

8.4　流密码

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位或者大于一字节的单元。图 8.7 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(位)发生器，该伪随机数发生器产生一串随机的 8 位数。发生器的输出称为**密钥流**，密钥流和明文流的每一个字节进行按位异或运算，得到一个密文字节。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
```

解密需要使用相同的伪随机序列

```
  10100000    密文
⊕ 01101100    密钥流
  11001100    明文
```

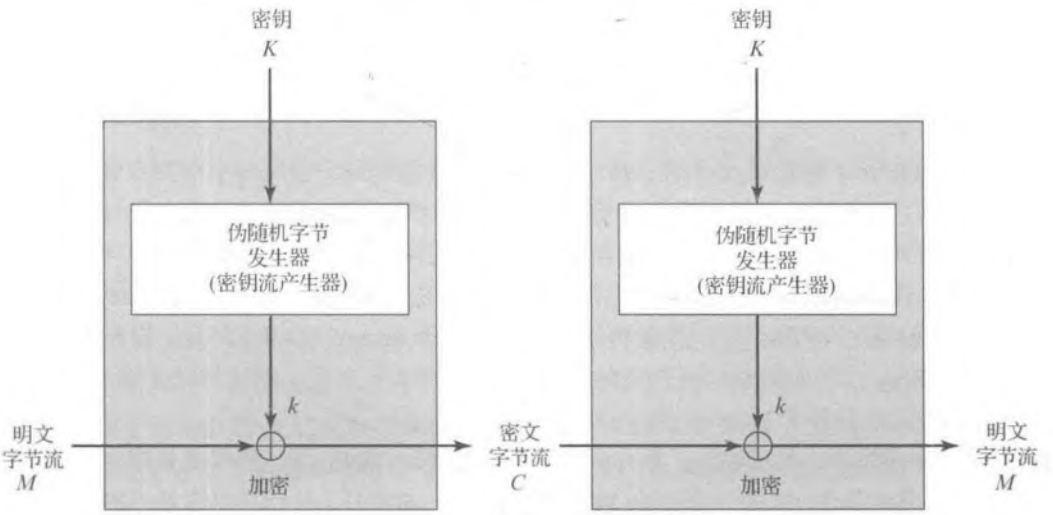


图 8.7　流密码结构图

流密码类似于第 3 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。

文献[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。伪随机数发生器使用的函数产生确定性的位流，该位流最终将出现重复。重复的周期越长，密码分析的难度就越大。这与讨论 Vigenère 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真正随机数流的特征。如 1 和 0 的个数应近似相等。若密钥流为字节流，则所有 256 种可能的字节的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 8.7 中伪随机数发生器的输出受输入密钥 K 的调节。为了防止穷举攻击，密钥应该足够长，即用于分组密码的考虑在此同样适用。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，当密钥长度相当时，流密码可以提供和分组密码一样的安全性。流密码的一个潜在的优点是没有使用分组密码作为构建模块的流密码通常都比分组密码更快、使用的代码更少。例如本章介绍的 RC4，仅仅数行代码就可实现。最近这些年，这一优势随着可通过软件高效实现的 AES 的引入已经消失了。进一步，硬件加速技术可以用于一轮加密运算或解密运算以及密钥扩展算法。与纯粹软件实现相比，用硬件指令实现，速度可以提高约一个数量级[XU10]。

分组密码的一个优点是可以重复使用密钥。而对于流密码，如果用流密码对两个明文加密且使用相同的密钥，则密码分析就会相当容易[DAWS96]。如果对两个密文流进行异或，那么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。

对于需要对数据流进行加密/解密的应用，比如在数据通信信道或者网页浏览连接上，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，e-mail 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使用于各种应用。

流密码可以用任何具有强密码学意义的 PRNG 来构造，如 8.2 节和 8.3 节讨论的例子。下一节中，将研究使用 PRNG 的流密码，该 PRNG 是专门设计用于流密码的。

8.5 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。该算法以随机置换作为基础。分析显示该密码的周期很可能大于 10^{100}[ROBS95a]。每输出一字节的结果仅需要 8～16 条机器操作指令，软件实现的该密码运行很快。RC4 应用很广，例如，它用于作为 IEEE 802.11 无线局域网标准一部分的 WEP（Wired Equivalent Privacy）协议和新的 WiFi 受保护访问协议（WPA）中。作为可选，它也被用于 Secure Shell（SSH）和 Kerberos 中。在当时 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 1～256 个字节（8～2048 位）的可变长度密钥初始化一个 256 个字节的状态向量 $\boldsymbol{S}$，$\boldsymbol{S}$ 的元素记为 $\boldsymbol{S}[0]$，S[1]，…，$\boldsymbol{S}[255]$，从始至终置换后的 $\boldsymbol{S}$ 包含从 0～255 所有的 8 位数。对于加密和解密，字节 k（参见图 8.5）是从 $\boldsymbol{S}$ 的 255 个元素中按一种系统化的方式选出的一个元素生成的。每生成一个 k 值，$\boldsymbol{S}$ 中的元素个体就被重新置换一次。

8.5.1 初始化 S

开始时，$\boldsymbol{S}$ 中元素的值按升序被置为从 0～255，即 $\boldsymbol{S}[0]=0$，$\boldsymbol{S}[1]=1$，…，$\boldsymbol{S}[255]=255$。同时建立一个临时向量 $\boldsymbol{T}$。如果密钥 K 的长度为 256 字节，则将 K 赋给 $\boldsymbol{T}$。否则若密钥长度为 keylen 个字节（keylen<256），则将 K 的值赋给 $\boldsymbol{T}$ 的前 keylen 个元素，并循环重复用 K 的值赋给 $\boldsymbol{T}$ 剩下的元素，直到 $\boldsymbol{T}$ 的所有元素都被赋值。这些预操作可被概括为如下：

```
/*  初始化  */
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

```
/*  S 的初始置换  */
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，所以唯一的改变就是置换。S 仍然包含所有值为 0～255 的元素。

8.5.2　密钥流的产生

向量 S 一旦完成初始化，输入密钥就不再被使用。密钥流的生成过程是，从 $S[0]$到 $S[255]$，对每个 $S[i]$，根据 S 的当前配置，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

```
/*  密钥流的生成  */
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 的值与明文的下一字节异或；解密中，将 k 的值与密文的下一字节异或。

图 8.8 总结了 RC4 的逻辑结构。

8.5.3　RC4 的强度

关于分析 RC4 的攻击方法有许多公开发表的文献（如文献[KNUD98]，[FLUH00]，[MANT01]）。但是当密钥长度很大时，比如 128 位，没有哪种攻击方法有效。值得注意的是文献[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 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修复这种攻击。这个问题恰恰说明设计一个安全系统的困难性不仅包括密码函数，还包括协议如何正确地使用这些密码函数。

最近，文献[PAUL07]揭示了 RC4 密钥安排算法的更根本的脆弱性，这减少了恢复密钥所需的工作量。最近的密码分析者探索了 RC4 密钥流反复加密明文所具有的偏差。作为漏洞发现的结果，尤其是在文献[ALFA13]报道的，IETF 发布 RFC7465，禁止 RC4 在 TLS 中的使用（Prohibiting RC4 Cipher Suites，February 2015）。在最近的 TLS 指南中，NIST 也禁止了 RC4 在政府中的使用（SP 800-52 ,Guidelines for the Selection, Configuration , and Use of Transport Layer Security(TLS) Implementations, Septermber 2013）。

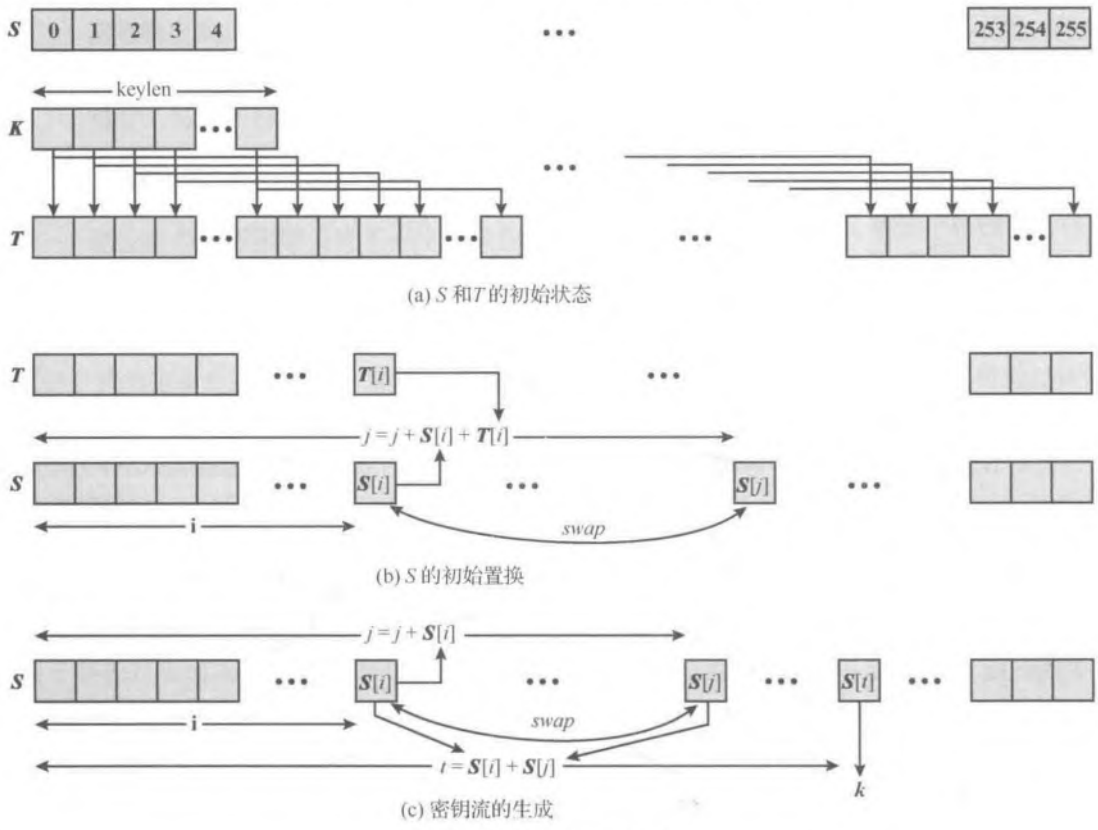


图 8.8 RC4 的逻辑结构

8.6 真随机数发生器

8.6.1 熵源

真随机数发生器（TRNG）使用不可预测源来产生随机数。大多数是通过测量不可预测的自然过程来实现的，如电离辐射的脉冲检测、气体放电管、漏电电容等。Intel 开发出一种芯片通过放大无驱动电阻的电压对热噪声进行采样。LavaRnd 是一项开放的项目，目标是通过廉价的相机、开源的代码和廉价的硬件来产生真随机数。该系统使用密封的饱和式 CCD 作为混沌源来产生种子，然后用软件将种子加工成各种形式的真随机数。

RFC 4086 列出了如下可能的随机源，这些都很容易用于计算机来产生真随机序列。

- **声音/图像输入** 许多计算机具有对现实世界模拟信号进行数据化的输入设备，这些信号有来自麦克风的声音或来自照相机的声像输入。如果声音数字化设备没有音源插入或者照相机没打开镜头盖那么来自这些设备的输入本质上是热噪声。如果系统具有足够的增益来检测任何信号，这种输入可以提供高质量的随机位。
- **磁盘驱动** 由于紊乱的空气波动，磁盘驱动在转动时有很小的随机波动[JAKO98]。加上一个底层磁盘时间寻找仪器就能产生一系列包含随机性的度量值。这样的数据通常都是高度相关的，所以需要有效的处理过程。然而，十年前的实验就表明，做过处理后，即使是当时低速计算机上的低速磁盘驱动器也能容易地每分钟产生 100 位或更多优质的随机数据。

在线服务网（random.org）可以通过互联网安全地传送随机序列。

8.6.2　PRNG 和 TRNG 的比较

表 8.5 总结了 PRNG 和 TRNG 的主要差异。PRNG 很有效，意味着它们能在短时间内生成许多数字；PRNG 是确定性的，意味着对于一个给定的数字序列若序列的开始点已知，一段时间后会再生。若你的应用程序需要很多数字，有效性是一个很好的特性；若你在后面的阶段会重复相同的数字序列，确定性是很易实现的。PRNG 通常是周期性的，这意味着序列最终会复制它自己。

TRNG 需要相当长时间生成数字，通常没有 PRNG 效率高。这在许多应用中会遇到困难。例如，银行或国家安全部门的密码系统每秒需要产生数百万的随机位。TRNG 也没有确定性，意味着即使相同的序列出现若干次，一个给定的数字序列也不能复制。TRNG 没有周期性。

8.6.3　调整[①]

一个真随机数发生器的输出应该有时会产生偏差，比如 1 的个数比 0 多，或者反之。更一般地，NIST SP 800-90B 定义的随机数产生离散输出（也就是可能的输出值）是有偏的，如果考虑到有些输出值出现的概率比其他的高。例如，一个物理源，如电子噪声，可能包含规则结构的叠加，如波形或其他周期现象。它们看起来是随机的，但是通过统计测试却不是随机的。

表 8.5　对比 PRNG 和 TRNG

	伪随机数发生器	真随机数发生器
效率	较高	较低
确定性	确定性的	非确定性的
周期性	周期性的	非周期性的

除了偏差，另一个在 SP 800-98B 中使用的概念是熵率。SP 800-98B 将熵率定义为数字化噪声源（或熵源）提供熵的比率。它通过将熵的估计值（通过从熵源得到位串得到）除以位串的总的位数得到（得到每一输出位提供的熵值的估计值）。这个值介于 0（没有不确定性）到 1（完全不确定）之间。熵率是一个位串随机性和不可预测性的一种度量。另一种计算给熵源熵率的方法是：熵率等于 k/n，其中 n 是位串的长度，k 是最小熵。最小熵是随机位串的随机性的一种度量，在附录 F 中有解释。本质上，一块位或位流是无偏的，即每一位或一位组和其他位或位组相互独立，熵率将等于 1。

对于随机位的硬件源，一种推荐的方法是假设熵源是有偏的或熵率小于 1，然后通过其他技术进一步随机化这些位。有很多用于这一目的，调整位流的方法已被发展。这有一些调整算法和偏差矫正算法也是可用的。

典型地，调整就是通过一种密码算法打乱位，减少偏差增加熵。有两种通用的方法使用了 Hash 函数和对称分组密码算法。

Hash 函数　正如第 11 章所描述的，Hash 函数将任意长度的位串映射为长度为 n 的位串。一种用 Hash 函数进行调整的方法如下。长度为 m（$m>n$）的位块通过 Hash 函数产生长度为 n 的位串输出，作为随机位。为了产生随机位流，连续的输入块通过 Hash 函数产生连续的输出块。

操作系统通常会提供一个内嵌的机制来生产随机数。例如，Linux 利用 4 个熵源：鼠标和键盘的活动，磁盘的 I/O 操作以及特定的中断。从这些源来的位在缓冲池内组合。当需要随机位时，从中读取适量的位，传给 SHA-1 函数[GUTT06]。

一种更复杂的方法是 SP800-90A 中的 Hash 派生算法（Hash_df）。Hash_df 定义如下：

① 在进一步阅读之前，对熵和最小熵不熟悉的读者应该先阅读附录 F。

参数

input_string：将被 hash 的串。

outlen：输出长度

no_of_bits_to_return：Hash_df 返回的位数。最大长度（max_number_of_bits）和实现有关，要小于等于 255×outlen。no_of_bits_to_return 用一个 32 位整型表示。

requested_bits：Hash_df 的执行结果。

Hash_df 过程

（1）temp = 空串

（2）len = $\left\lceil \frac{\text{no_of_bits_to_return}}{\text{outlen}} \right\rceil$

（3）counter = 0x01 注释：一个 8 位的二进制数表示整数 1

（4）For i=1 to len do 注释：（4.1）中 no_of_bits_to_return 被用作是 32 位串

（4.1）tem = temp||**Hash**(counter||no_of_bits_to_return||input_string)

（4.2）counter = counter+1

（5）requested_bits = **leftmost**(temp , no_of_bits_to_return)

（6）Return (**SUCCESS**, requsted_bits)

这个算法将任意长度的位串作为输入，输出要求长度的位串，它可以长达 Hash 函数输出长度的 255 倍。

读者可能觉得有点不理解，Hash 函数的输入包含了相同的输入串，仅仅是 counter 不同。然而，密码学性质强的 Hash 函数，如 SHA 系列，有非常好的扩散性（定义参见第 4 章），所以改变 counter 的值将导致结果产生巨大的不同。

分组密码 作为 Hash 函数的一种代替，分组密码，如 AES，可以用来打乱 TRNG 位。一种简单的方法是用任意密钥加密 128 位的 TRNG 块。SP 800-90B 有一种和前面提到的 hash_df 函数相似的方法。在后面的 Intel 实现中，提供了一个使用 AES 进行调整的例子。

8.6.4 随机性测试

图 8.9 为一个非确定的随机位发生器提供了一个通用的模型。一个硬件噪声源产生真正的随机输出。这个输出经过数字化处理后就是真正的非确定位源了。这个位源再经过调节模块，进一步缓和偏差，最大化熵。

图 8.9 还展示了一个随机性测试模块，用于测试数字转换器和调节器的输出。本质上，随机性测试就是要证实噪声源像我们期望的那样工作，以及调节模块能够产生具有我们想要特征的输出。SP 800-90B 推荐了两种随机性测试。

噪声源的随机性测试 噪声源的随机性测试很强地依赖于产生噪声的技术。通常，我们假设噪声源的数字化输出具有一定的偏差。这样，传统的统计测试，如在 SP 800-22 中定义的并在 8.1 节中讨论的，监视噪声源，都是没用的。因为噪声源很可能总是失效。进一步，测试需要根据正确工作的噪声源（我们所期望的）的统计特性进行定制。我们的目标不是确定噪声源是否有偏差，而是确定它是否按照我们所期望的那样工作。

SP 80-90B 详细给出了从噪声源获得的数字化样本（图 8.9 中的 *A* 点）的测试。目的是测试可变性。更具体的是确定噪声源是否产生了我们所期望的熵率。SP 800-909B 授权了两种测试的使用：重复计数测试和适应比例测试。

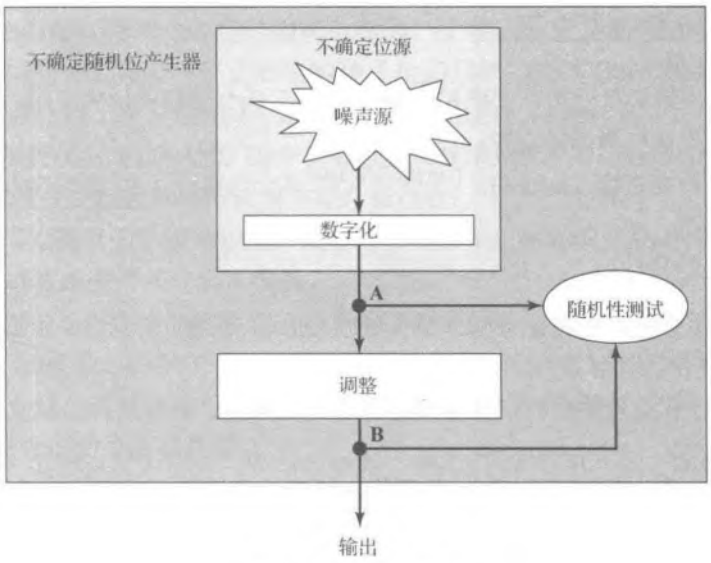


图 8.9　NRBG 模型

重复计数测试用来探测引起噪声源长时间停留在一个值上的灾难性故障。对于这个测试，我们假设对于给定的噪声源有一个最小熵的估计值 H。这个熵表示每一个样本不确定性的大小，一个样本可以是一位或是 n 位。有了这个估计的熵值，我们可以直接计算连续的 C 个样本都取相同值的概率。例如，每一样本最小熵为一位的噪声源，在一列位中有连续 2 位相同的概率不超过 1/2，有连续 3 位相同的概率不超过 1/4。一般地，在一列位串中有连续 C 位相同的概率不超过$(1/2)^{C-1}$ 。一般地，对于一个样本最小熵为 H 位的噪声源，有

$$\Pr[\text{在一列位中，连续 C 位相同}] \leqslant (2^{-H})^{(C-1)}$$

重复计数测试涉及寻找连续相同的样本。一旦计数达到了一个阈值 C，就会报错。为了确定在这个测试中用到的 C，测试必需设定一个参数 W，它是和一个值 C 次重复出现相关的假阳性概率。为了避免假阳性，W 的值要非常接近于 0 。给定 W，现在来确定 C 值。特别地，我们希望 C 是满足 $W \leqslant (2^{-H})^{(C-1)}$的最小值。进一步，有

$$C = \left\lceil 1 + \frac{-\log(W)}{H} \right\rceil$$

例如：给定 $W=2^{-30}$，样本熵 $H=7.3$ 的熵源，我们有阈值 $C = \left\lceil 1 + \frac{30}{7.3} \right\rceil = 6$ 。

重复计数测试算法先记录一个样本值，然后对同一个值连续重复出现的次数进行计数。如果计数器达到了阈值 C，就报错。如果遇到了一个与之前不同的值，就将计数器重置 1，算法从新开始。

适应比例测试设计用来侦测熵的巨大损失，如由物理故障或环境改变影响了噪声源引起的。这种测试通过连续地测量噪声源的一列样本值中某些样本出现的局部频率来决定样本是否出现得太过频繁了。

这个测试先记录一个样本值，然后观测 N 个连续样本值。如果初始的样本值出现了至少 C 次，就报错。SP 800-90B 推荐在这个测试中假阳性概率 $W=2^{-30}$，同时为 N 和 C 的选择提供了指导。

调节函数的随机性测试　SP 800-90B 规定随机性测试也要用到调节模块的输出（参见图 8.9 的 B 点）上，但是没有指示用那种测试。调节函数的随机性测试的目的是确定调节函数的输出表现和真正的随机位流一样。这样，就可以在 SP 800-22 中定义的和在 8.1 节中描述的随机性测试。

8.6.5 Intel 数字随机数发生器

如上所述，TRNG 通常只适用于密钥扩展和其他只需要少量随机位的应用。这是因为 TRNG 通常是低效的且只能产生低位率的随机位。

英特尔数字随机数发生器（DRNG）[TAYL11, MECH14]第一次实现了和 PRNG 相当的位生成率，它提供新的多核芯片。

DRNG 两个显著的特性：

（1）完全用硬件实现。这比包含软件组件的设备具有更高的安全性。仅硬件实现也能达到比软件组件更快的计算速度。

（2）整个 DRNG 和处理器在同一个多核芯片上，这减少了硬件随机数发生器之间的 I/O 时延。

DRNG 的硬件架构 图 8.10 展示了 DRNG 的整体结构。在技术发展的第一阶段，DRNG 从电路的热噪声生成随机数。电路的核心由两个反相器（非门）组成，每个反相器的输出连接到另一个的输入。这样的结构具有两个稳定状态，一个反相器输出逻辑 1，则另一个就输出逻辑 0。这种电路配置使两个反相器在时钟脉冲作用下具有相同的不稳定状态（所有的输入和输出都为逻辑 1）。反相器内的随机热噪推动两个反相器至相互稳定的状态。辅助电路用来补偿上述电路的偏差。在这一阶段，用硬件电路产生随机数的速率可达 4 Gbps。

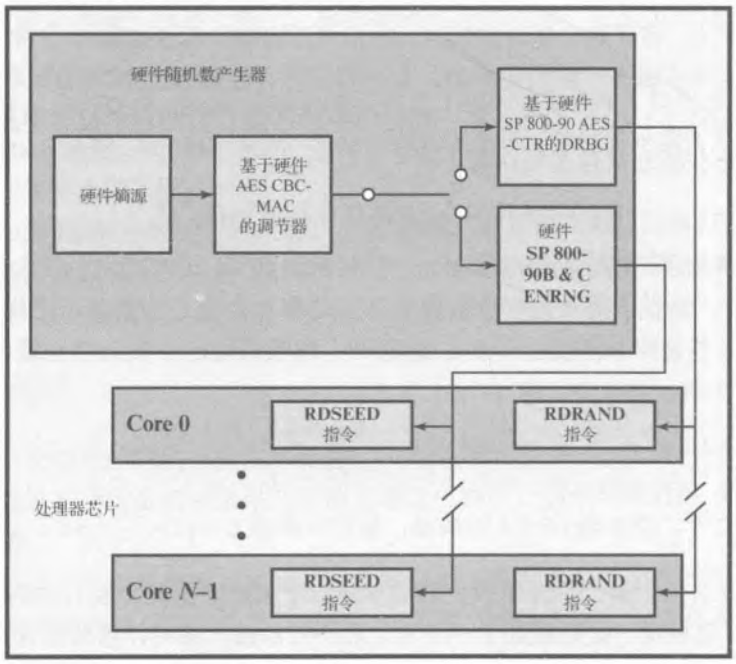


图 8.10 具有随机数发生器的 Intel 微处理器芯片

在技术发展的第一个阶段，电路的输出是每一时刻产生 512 位随机数。为了确保位流没有随机性偏差，在技术发展的第二阶段，人们采用一个密码函数对原有的输出进行随机化处理。在这种情况下，按照 NIST SP 800-38B 中的说明，这个函数称为 CBC-MAC 或 CMAC。本质上，CMAC 就是采用密文分组链接（CBC）方式（参见图 8.4）加密其输入，并把其输出作为最终的输出。我们将在第 12 章详细讨论 CMAC。在这一阶段，电路的输出是一次产生 256 位随机数，并且随机性不表现出有偏差。

虽然基于硬件电路热噪能够很快地产生随机数，但这仍满足不了现在的一些计算需求。为了使 DRNG 能和软件 PRNG 一样快地产生随机数并保持随机数的高质量，于是产生了技术发展

的第三阶段。在这个阶段，使用 256 位的随机数作为一个密码学安全的 PRNG 的种子，来产生一个 128 位的随机数。PRNG 能从一个 256 位的种子以超过熵源的 3 Gbps 速率的速度输出许多伪随机数，能产生上界为 511 128 位的样本。这个阶段所使用的算法是在 8.3 节介绍的 CTR_DRBG。

芯片的每个核都能通过 RDRAND 指令获取使用 DRNG 的输出。RDRAND 检索一个 16 位，32 位或 64 位的随机值使它能被软件可访问的寄存器获取。

Intel Core 第三代处理器的系统从预生产的样本产生初始数据时的性能[INTE12]：每秒 7 千万次 RDRAND 调用，随机数据生产率达到 4 Gbps。

调节器的输出也可用于另一个被称为加强非确定随机数发生器（ENRNG）模块，它提供用于各种密码算法种子的随机数。ENRGN 符合 SP 800-90B 和 SP 900-90C 中的规范。ENRGN 的输出可以通过 RDSEED 指令用于芯片上的每一个核。RDSEED 指令从 ENRNG 获得一个硬件产生的随机数种子，然后存入指令中的目的寄存器。

DRNG 的逻辑结构　图 8.11 描述了一个 Intel DRNG 的简化逻辑流示意图。硬件熵源的核心是一对互相反馈的反相器。两个由相同时钟驱动的晶体管使反相器的输入、输出都为逻辑 1 态。由于这是不稳定状态，热噪声会引起配置随机生成一个节点 A 为逻辑 1、节点 B 为逻辑 0 或相反的稳定态。因此，这个模块以时钟频率生成随机位。

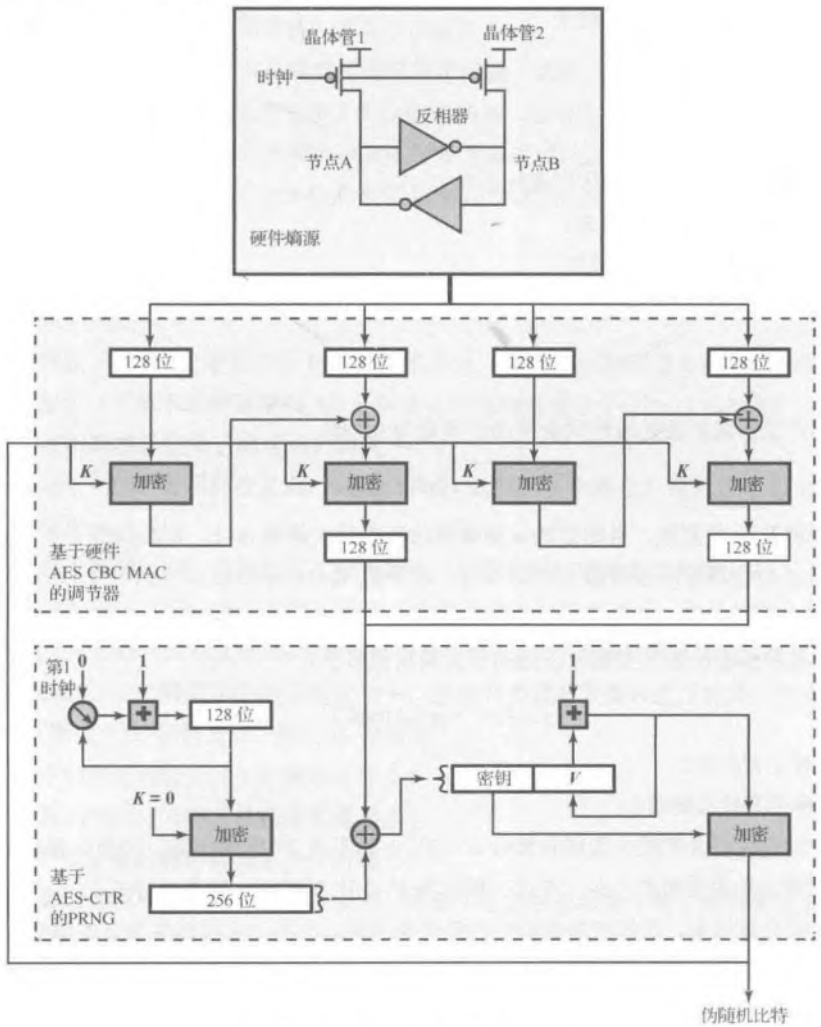


图 8.11　Intel DRNG 逻辑结构

熵源的输出每次采集 512 位反馈给两个用来进行 AES 加密的硬件实现的 CBC。每个实现需两个 128 位分组的明文，使用 CBC 模式加密。第二次加密的结果被保留了。CBC 模块最初需要使用全零的密钥。其后，PRNG 阶段的输出反馈给调节器阶段作为密钥。

调节器的输出由 256 位组成。这个分组作为 PRNG 阶段更新函数的输入。更新函数用全零密钥和零值计数器初始化。这个函数迭代两次产生一个 256 位分组，再异或调节器的输入，结果作为生成函数的 128 位密钥和 128 位种子。生成函数生成 128 位的伪随机位。

8.7 关键术语、思考题和习题

关键术语

后向不可预测性
Blum, Blum, Shub 发生器
消偏
熵源
前向不可预测性
线性同余发生器
伪随机函数（PRF）
伪随机数发生器(PRNG)
随机性
RC4
种子
流密码
偏差
真随机数发生器（TRNG）
不可预测性
密钥流

思考题

8.1 统计随机性和不可预测性的区别是什么？

8.2 请列出设计流密码的重要考虑？

8.3 为什么流密码重用而密钥不好？

8.4 RC4 用到了什么原语操作？

习题

8.1 如果将产生伪随机数的线性同余的加法项设为 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 $m-1$ 和 k 互素，则 a^k 也使得算法获得最大周期 $m-1$。以参数 $X_0=1$，$m=31$，$a^k=3$，3^2，3^3，3^4 为例生产序列来说明上述论断。

8.2 （a）下述的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？

8.3 你也许对线性同余算法中选用的模数 $m = 2^{31}-1$ 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？

8.4 在线性同余算法中，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发生器

$$X_{n+1} = (6X_n) \bmod 13$$

$$X_{n+1} = (7X_n) \bmod 13$$

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？

8.5 对伪随机数的任何使用，比如加密、仿真或是统计设计，盲目地相信计算机系统中的伪随机数发生器是危险的。文献[PARK88]发现了当代许多教材和软件包中使用的伪随机数发生器有缺陷。下面这道题可使你对你的系统进行测试。

测试基于 Ernesto Cesaro 的一个定理（文献[KNUT98]有证明），他证明了两个随机选择的整数的最大公因子是 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：一个是从系统中找到一些伪随机数发生器，并用它们产生一些伪随机数；一个是计算两个整数的最大公因子的 Euclid 算法；一个是计算平方根。如果后面两个子程序没有现成的，需要自己编写。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，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？

8.6 哪个 RC4 的密钥可以使 $\boldsymbol{S}$ 在初始化后不变？即在 $\boldsymbol{S}$ 的初始化置换以后，S 的项依次等于 0～255。

8.7 RC4 有保密的内部状态，即 $\boldsymbol{S}$ 的所有值的一个置换以及两个索引 i 和 j。

（a）用直接的方案来存储内部状态，需要多少位？

（b）假设我们从状态表示了多少信息的角度考虑问题。此时，我们需要判决有多少种不同的状态，然后取以 2 为底的对数求出这个表示了多少位的信息。运用这个方法，需要多少位来表示状态？

8.8 Alice 和 Bob 同意用基于 RC4 加密的电邮进行私有通信，但他们不想每次传输都需要一个新的密钥。Alice 和 Bob 秘密协商了一个 128 位的密钥 k。为了加密由位构成的消息 m，遵循如下程序：

（1）选择随机的 80 位值 v

（2）产生密文 $c = \text{RC4}(v||k) \oplus m$

（3）发送位串$(v||c)$

（a）假设 Alice 用上述程序给 Bob 发送消息 m。描述 Bob 如何用 k 从$(v||c)$恢复出消息 m。

（b）如果一个敌手能够观察到 Alice 和 Bob 间传输的若干个$(v1||c1), (v2||c2)$，…值，他/她如何判定两条消息使用了相同的密钥流？

（c）在一个密钥使用两次之前，Alice 可以大约发送多少消息？使用附录 U 给出的生日悖论的结果。

（d）关于密钥 k 的生命周期这意味着什么（即用 k 可加密消息的数量）？

8.9 假设有真随机数发生器，产生位串中的每一位为 0 或 1 的概率相同，并且各位之间互不相关。也就是说这些位来自相同的独立分布。然而该位串是有偏的。取 1 的概率为 $0.5+\partial$，取 0 的概率为 $0.5-\partial$，这里 $0<\partial<0.5$。一种简单的消偏算法如下：将序列当成非重叠对进行检查，丢弃所有的 00 和 11 组合。并将每个 01 替换为 0，每个 10 替换为 1。

（a）在原序列中每对组合产生的概率是多少？

（b）修改后的序列中 0 和 1 的概率各是多少？

（c）为了产生 x 位的输出需要多少位的输入？

（d）假设算法使用重叠对而不是非重叠对，即第 1 位的输出取决于第 1 位和第 2 位的输入。

8.10 另一种消偏算法是将位串看成连续非重叠的 n 位分组，然后输出每分组的奇偶性，即如果一组包括奇数个 1，则输出为 1；否则输出为 0。

（a）将该操作表示为布尔函数的形式。

（b）假设同上题取 1 的概率为 $0.5+\partial$，如果每组包括 2 位，那么输出为 1 的概率是多少？

（c）如果每组包括 4 位，那么输出为 1 的概率是多少？

（d）总结出如果每组包括 n 位时输出为 1 的概率。

8.11 注意到在 8.6 节中描述的重复计数测试不是非常有效的随机性测试是重要的。它可以检测到熵源灾难性的错误。例如，评估每一样本 8 位的最小熵的噪声源有一个阈值为 5 的重复数，可以确保发错误的比率接近 40 亿次中只出现 1 次。如果噪声源每一样本只能提供 4 位的最小熵，则需要多少样本才能通过重复计数测试检测出这个错误？

第三部分　公钥密码

第 9 章　公钥密码学与 RSA

第 10 章　密钥管理和其他公钥密码体制

第9章　公钥密码学与RSA

学习目标

通过本章的学习应具备以下能力：

- 概述公钥密码的基本原理
- 阐述公钥密码的两个不同的应用
- 列举和解释对公钥密码的要求
- 概述RSA算法
- 理解计时攻击
- 总结关于算法复杂性的相关问题

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，也许可以说是唯一的一次革命。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，但是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，公钥算法是基于数学函数而不是基于替换和置换，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。

下面先讨论有关公钥密码常见的几种误解。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。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会话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（参见文献[NEED78]的分析）。

本章和下一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章包含一些其他的重要公钥密码算法。

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，那么就可以不必严格地

去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 2 章中概要介绍了与公钥密码有关的数论知识。

表 9.1 定义了一些公钥密码重要术语。

表 9.1　公钥密码术语

非对称密钥 两个密钥：公钥和私钥，用来实现互补运算，即加密和解密，或者生成签名与验证签名
公钥证书 认证机构将用户的姓名和公钥绑定在一起，用户用自己的私钥对数字文件签名后，可以通过证书识别签名者，因为签名者是唯一拥有与证书上对应的私钥的用户
公钥密码（非对称密码）算法 含有两个密钥：公钥和私钥。从公钥中推出私钥在计算上不可行的
公钥基础设施 由一系列的协议、服务平台、软件和工作站组成，用于管理证书和公钥-私钥对，并产生、维护和废除公钥证书

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 14 章中经详细讨论过的密钥分配问题。

如第 14 章所讨论的，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；（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领域，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[DIFF76a,DIFFb]。

在下一小节中，先讨论公钥密码学的基本框架，然后讨论加密/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：

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。Hellam 认为 Merkle 也在同一时间独立发现了这一概念。虽然 Merkle 直到 1978 年才发表[MERK78]。实际上，描述公钥分配和公钥加密的第一个非保密文档是 1994 年由 Merkle（http://merkle.com/1974）提供的一个项目提案。但这并不是真正的开始。美国国家安全局（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 Jame 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 Communication-Electronics Security Group)，Ellis 称这种技术为非秘密的加密方法，并在文献[ELLI99]中描述了这一发现。

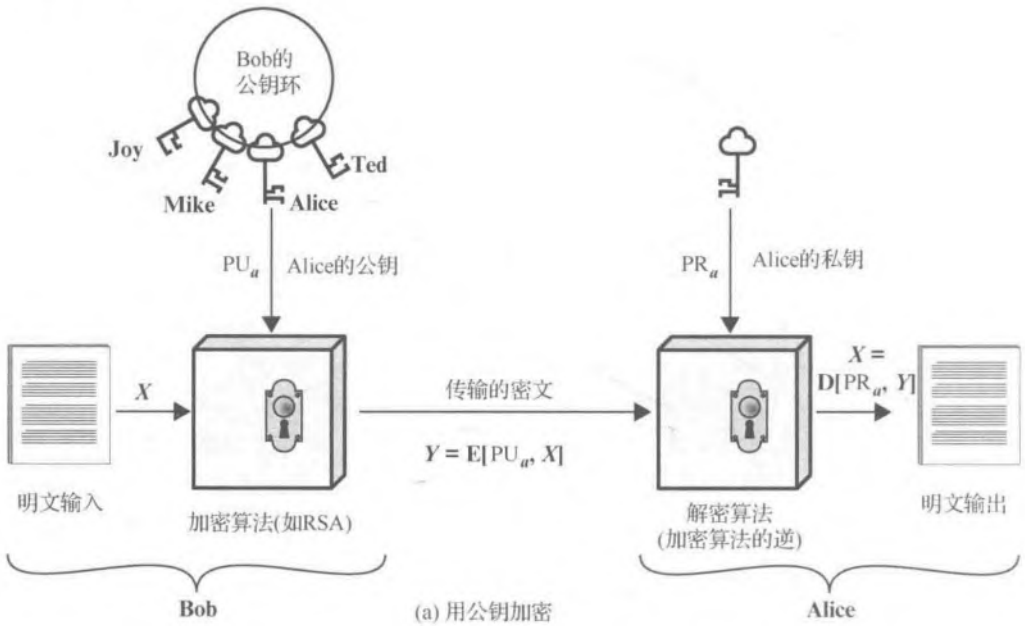
- 仅根据密码算法和加密密钥来确定解密密钥在计算上是不可行的。

另外有些算法（如 RSA），还有以下特点：

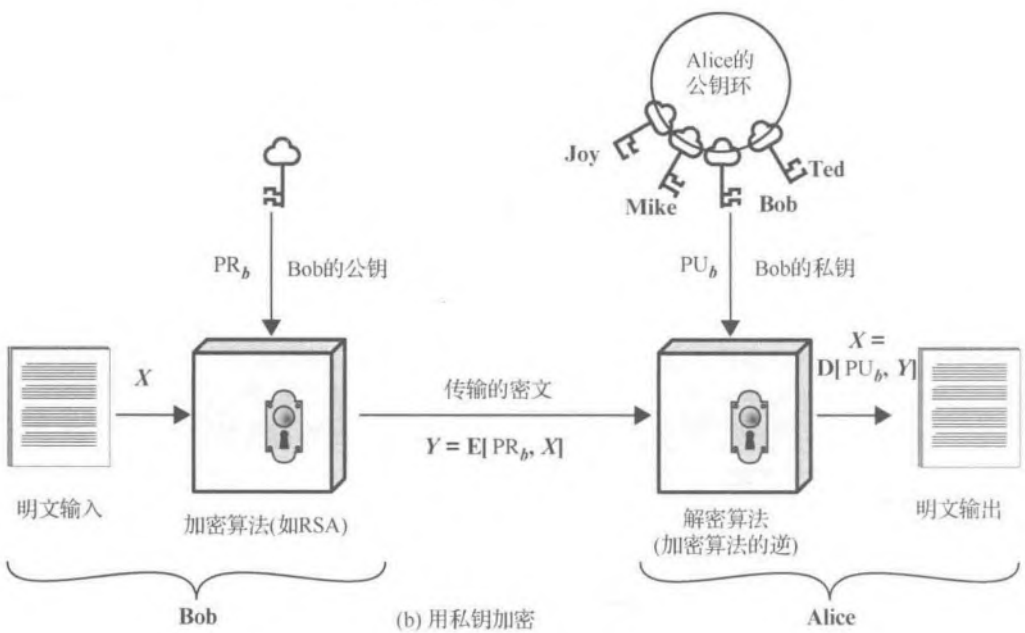
- 两个密钥中的任何一个都可用来加密，另一个用来解密。

公钥密码体制有 6 个组成部分［参见图 9.1(a)；并对照图 3.1］：

- **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，一个用于解密。加密算法执行的变换依赖于公钥或者私钥。



(a) 用公钥加密



(b) 用私钥加密

图 9.1 公钥密码体制

- **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）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。

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。只要用户的私钥受到保护，保持秘密性，那么通信就是安全的。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 9.2 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，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 9.2　传统密码和公钥密码

传 统 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和相同的算法	（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 3.2）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 PU_b 和私钥 PR_b，其中只有 B 知道 PR_b，而 PU_b 则是公开可访问的，所以 A 也可访问 PU_b。

对作为输入的消息 X 和加密密钥 PU_b，A 生成密文 $Y=[Y_1, Y_2, \cdots, Y_N]$

$$Y=E(PU_b, X)$$

期望的接收方因为拥有相应的私钥，所以可进行逆变换：

$$X=D(PR_b, Y)$$

攻击者可观察到 Y 并且可访问 PU_b，但是他不能访问 PR_b 或者 X，所以攻击者肯定会想方设法恢复 X 和/或 PR_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 X 来恢复 $\hat{X}$。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 PR_b 来试图恢复 $P\hat{R}_b$。

---

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_a 表示 A 的秘密密钥；用户 A 的公钥用 PU_a 表示，其相应的私钥用 PR_a 表示。分别用 $E[K_a, X]$、$E[PU_a, X]$和 $E[PR_a, X]$表示用秘密钥、公钥和私钥对明文 P 加密，同样，分别用 $D[K_a, Y]$、$D[PU_a, Y]$和 $D[PR_a, Y]$表示用秘密钥、公钥和私钥对密文 Y 解密。

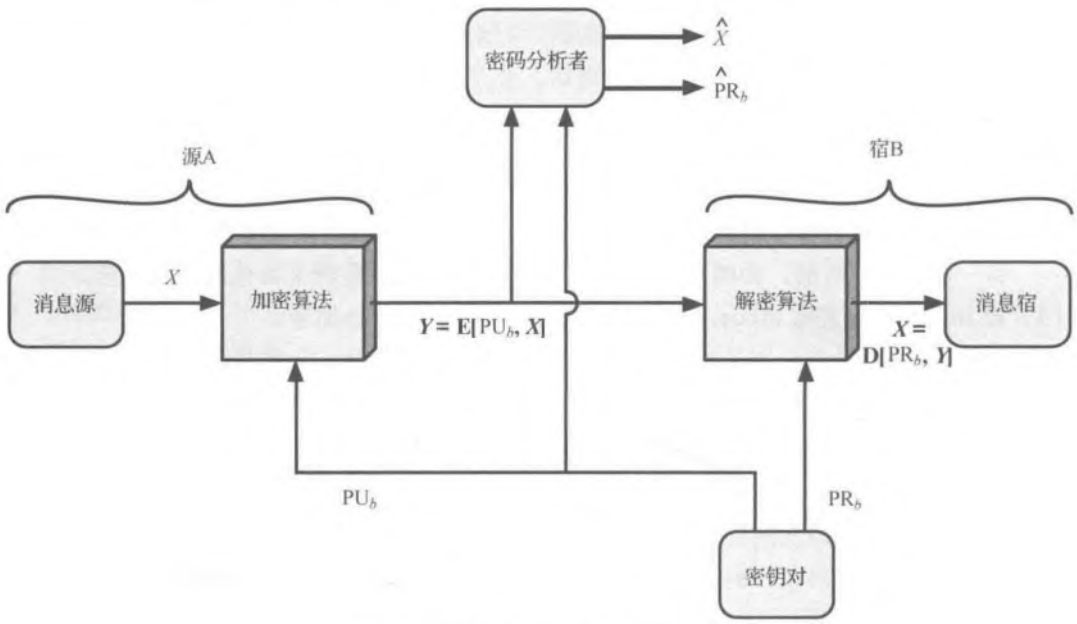


图 9.2 公钥密码体制：保密性

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，这样就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

$$Y = E(PR_a, X)$$
$$X = D(PU_a, Y)$$

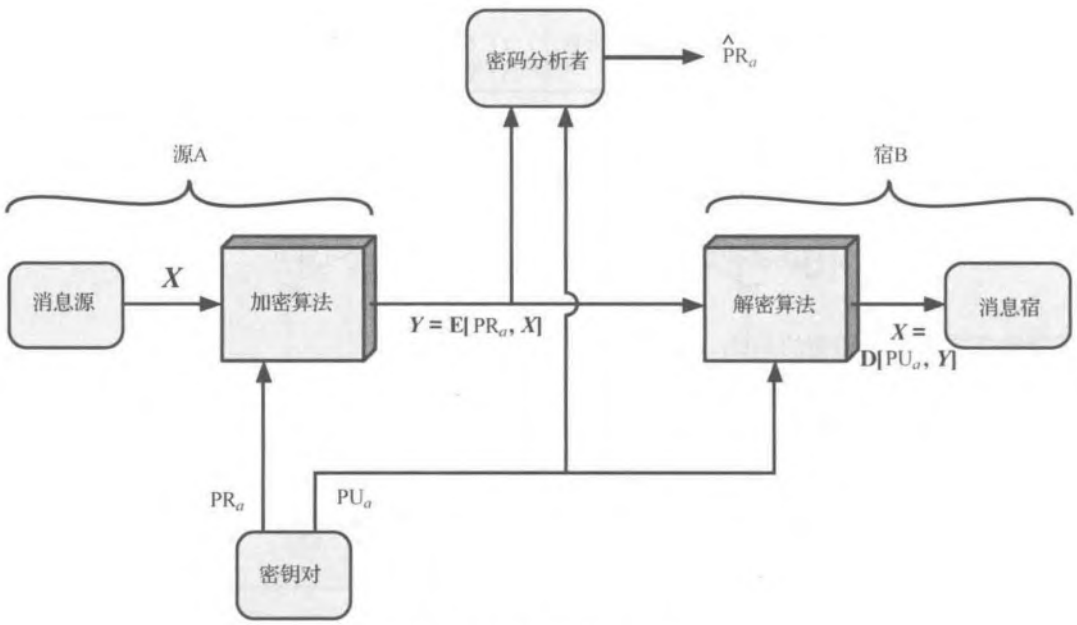


图 9.3 公钥密码体制：认证

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内容的有效性，但却需要大量的存储空间。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，在图 9.1(b)和图 9.3 中描述的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，但不能防止搭线窃听。在基于对部分消息签名的方法中，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。

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（参见图 9.4）

$$Z = \mathrm{E}(\mathrm{PU}_b, \mathrm{E}(\mathrm{PR}_a, X))$$
$$X = \mathrm{D}(\mathrm{PU}_a, \mathrm{D}(\mathrm{PR}_b, Z))$$

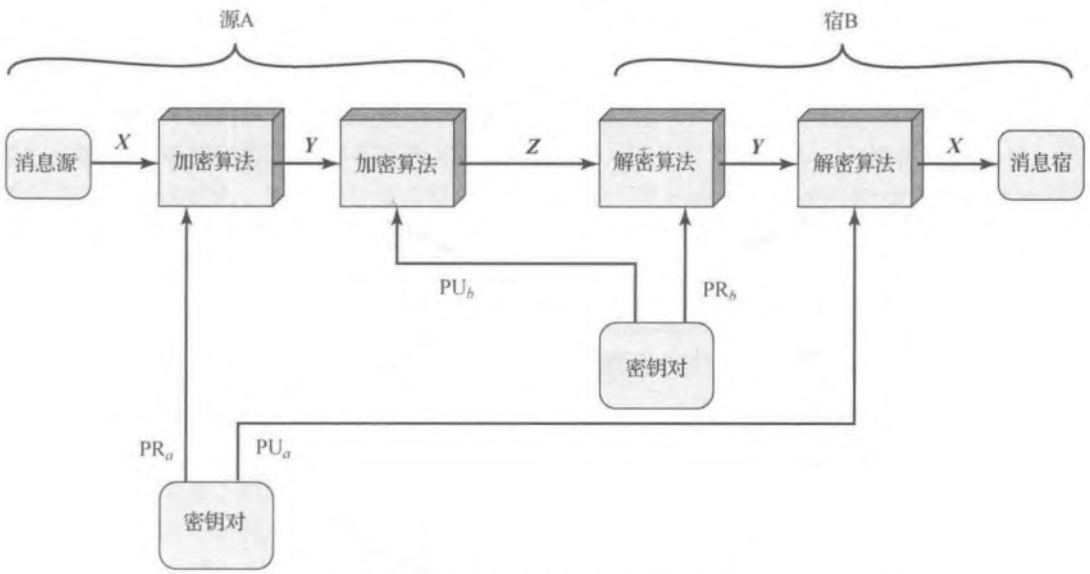


图 9.4　公钥密码体制：保密性和认证

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。

9.1.2　公钥密码体制的应用

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 9.3 列出了本书中所讨论的算法及其所支持的应用。

表 9.3 公钥密码体制的应用

算 法	加/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 PU_b，私钥 PR_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

$$C = E(PU_b, M)$$

（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

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

（4）已知公钥 PU_b 时，攻击者要确定私钥 PR_b 在计算上是不可行的。

（5）已知公钥 PU_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

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

在公钥密码学概念提出后的几十年中，只有几个满足这些条件的算法（RSA，椭圆曲线密码体制，Diffe-Hellman, DSS）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：每个函数值都存在唯一的逆，并且计算函数值是容易的，但求逆却是不可行的

$$Y = f(X) \quad \text{容易}$$
$$X = f^{-1}(Y) \quad \text{不可行}$$

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（参见第 11 章）。

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，这样的算法称为是 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 成正比，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 W 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 $Y = f_k(X)$。

若 k 和 Y 已知，则容易计算 $X = f_k^{-1}(Y)$。

若 Y 已知但 k 未知，则计算出 $X = f_k^{-1}(Y)$ 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

和对称密码一样，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/解密速度变慢，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。之后很多算法被提出，其中有一些刚提出时似乎很有前途，但后来都被攻破①。

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]②，

① 最著名的算法是由 Ralph Merkle 提出的背包公钥密码，这部分内容参见附录 J。

② 第一个可执行的公钥加密/解密系统是由英国 CESG 的 Clifford Cocks 于 1973 年提出的[COCK73]；Cock 的方法本质上与 RSA 是相同的。

可以说是最早提出的满足有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的公钥加密方法。

RSA 体制是一种分组密码，其明文和密文均是 0 至某 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，也就是说 n 小于 2^{1024}。本节我们将详细讨论 RSA 算法。首先给出该算法的描述，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

RSA 算法使用乘方运算，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 $\log_2(n)+1$ 位，在实际应用中，分组的大小是 i 位，其中 $2^i < n \leq 2^{i+1}$。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 PU={e,n}，私钥为 PR={d, n}。该算法要能用于公钥加密，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。

（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：

$$M^{ed} \bmod n = M$$

当 e 和 d 互为模 $\phi(n)$ 的乘法逆时，上述的关系式成立，其中 $\phi(n)$ 为欧拉函数。在第 2 章中已经证明，对素数 p 和 q，有 $\phi(pq) = (p-1)(q-1)$。e 和 d 的关系如下：

$$ed \bmod \phi(n) \equiv 1 \tag{9.1}$$

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 $\phi(n)$ 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。式（9.1）满足 RSA 要求的证明参见附录 R。

下面我们介绍 RSA 算法，该算法用到下列元素：

两个素数 p，q	（保密的，选定的）
$n=pq$	（公开的，计算得出的）
e，满足 $\gcd(\phi(n), e) = 1; 1<e<\phi(n)$	（公开的，选定的）
$d \equiv e^{-1} \pmod{\phi(n)}$	（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 $C = M^e \bmod n$，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。

图 9.5 归纳总结了 RSA 算法。与图 9.1(a)描述的一样，Alice 产生一对公钥/私钥，Bob 用 Alice 的公钥加密，Alice 用自己的私钥解密。图 9.6 所示的是文献[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，$p=17$，$q=11$。

（2）计算 $n=pq=17\times11=187$。

（3）计算 $\phi(n)=(p-1)(q-1)=16\times10=160$。

（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 $de \equiv 1 \bmod 160$ 且 $d<160$。因为 $23\times7=161=1\times160+1$，所以 $d=23$。d 可利用扩展的 Euclid 算法来计算（参见第 2 章）。

所得的公钥 PU = {7,187}，私钥 PR = {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：

$88^7 \bmod 187 = [(88^4 \bmod 187)\times(88^2 \bmod 187)\times(88^1 \bmod 187)] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$

$88^7 \bmod 187 = (88\times77\times132) \bmod 187 = 894{,}432 \bmod 187 = 11$

密钥产生	
选择 p, q	p 和 q 都是素数，$p \neq q$
计算 $n = p \times q$	
计算 $\phi(n) = (p-1)(q-1)$	
选择整数 e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
计算 d	$d \equiv e^{-1} \pmod{\phi(n)}$
公钥	PU = $\{e,n\}$
私钥	PR = $\{d,n\}$

加密	
明文	$M < n$
密文	$C = M^e \bmod n$

解密	
密文	C
明文	$M = C^d \bmod n$

图 9.5　RSA 算法

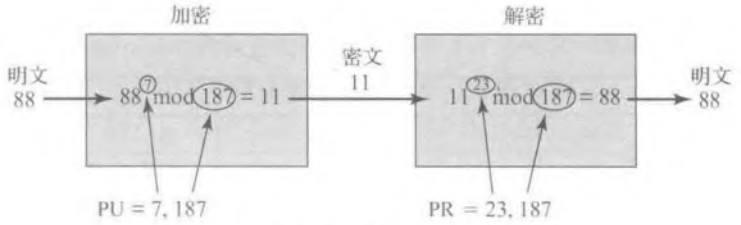


图 9.6　RSA 算法举例

解密时，我们计算 $M=11^{23} \bmod 187$

$11^{23} \bmod 187 = [(11^1 \bmod 187)\times(11^2 \bmod 187)\times(11^4 \bmod 187)$
$\times(11^8 \bmod 187)\times(11^8 \bmod 187)] \bmod 187$

$11^1 \bmod 187 = 11$

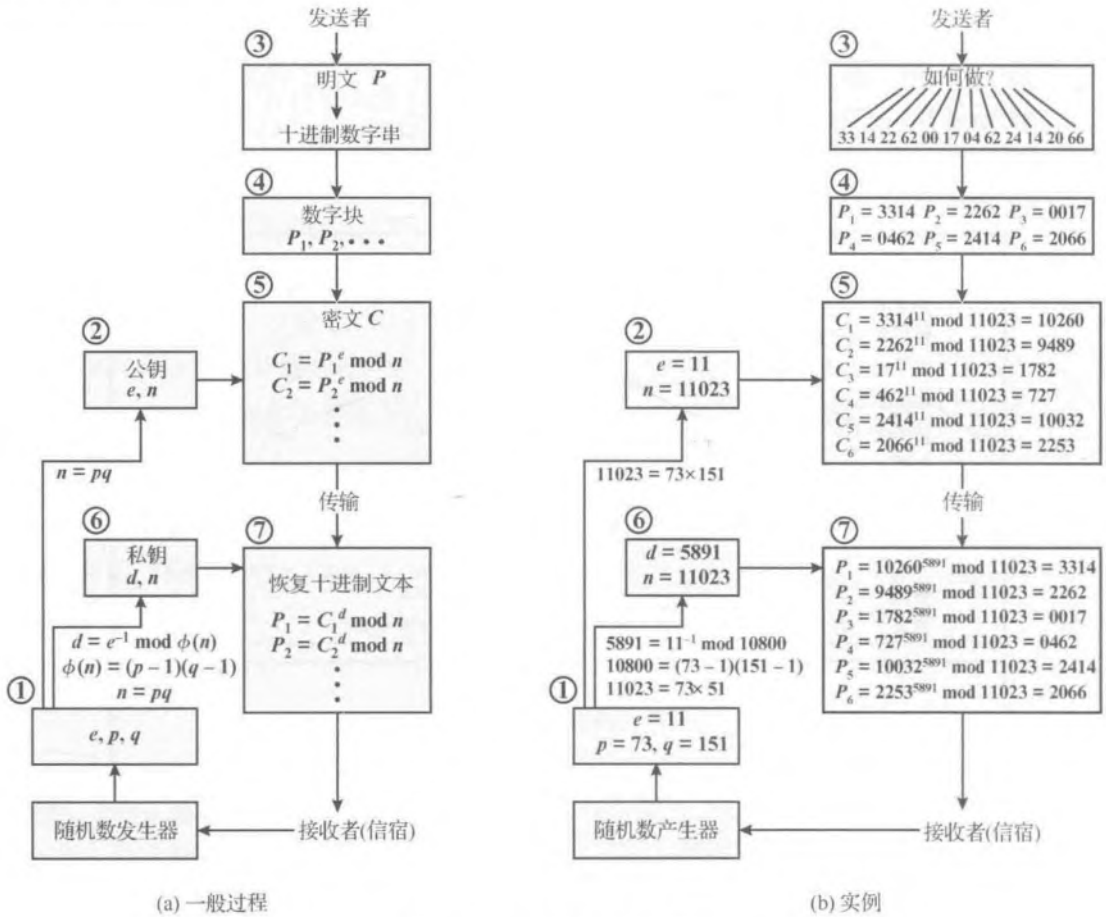
$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$

$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$

$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$

在文献[HELL79]中给出了一个使用 RSA 对不同的分组块进行加解密的例子，这个例子中，明文是一串字母，每个字母与一个两位的十进制数字对应（如 a=00，A=26）[①]，明文的每一分组块由 4 个十进制数字组成，即两个字母。图 9.7(a)给出了加密各分组块的过程，图 9.7(b)给出了一个实例。图中的数字表示运算的执行顺序。



(a) 一般过程　　(b) 实例

图 9.7　RSA 的加解密过程

9.2.2　计算方面的问题

下面讨论 RSA 的计算复杂性问题，它实际上包括两方面的问题：加密/解密和密钥产生。我们先讨论加密和解密过程，然后再讨论密钥的产生问题。

模算术里的求幂运算　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计算模幂运算：

① 完整的字母与十进制数字的对应可在本书提供的网站中（box.com/Crypto7e）下载，文件名是 RSAexample.pdf。

$$[(a \bmod n)\times(b \bmod n)] \bmod n = (a\times b) \bmod n$$

这样我们将中间结果对 n 取模，这使得计算切实可行。

因为 RSA 中所用到的指数很大，所以还应考虑幂运算的效率问题。为说明如何增加效率，以计算 x^{16} 为例。若直接计算则需进行 15 次乘法：

$$x^{16} = x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x$$

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}。又比如，对于整数 x 和 n，计算 $x^{11} \bmod n$。由于 $x^{11}=x^{1+2+8}=(x)(x^2)(x^8)$，我们先计算 $x \bmod n$，$x^2 \bmod n$，$x^4 \bmod n$，$x^8 \bmod n$，再计算$[(x \bmod n)\times(x^2 \bmod n)\times(x^8 \bmod n)] \bmod n$。

更一般地，假定要计算 a^b，其中 a 和 b 是正整数。若将 b 表示为二进制数 $b_k b_{k-1}\cdots b_0$，则

$$b = \sum_{b_i \neq 0} 2^i$$

所以

$$a^b = a^{\left(\sum_{b_i\neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^b \bmod n = \left[\prod_{b_i\neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i\neq 0}\left[a^{(2^i)} \bmod n\right]\right) \bmod n$$

下面我们讨论计算 $a^b \bmod n$ 的算法[①]，如图 9.8 所示。表 9.4 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 的终值即是指数。

```
c←0; f←1
for i←k downto 0
    do c←2×c
       f←(f×f) mod n
    if bi=1
                then c←c+1
                     f←(f×a) mod n
return f
```

注：整数 b 表示为二进制 $b_k b_{k-1}\cdots b_0$

图 9.8　计算 $a^b \bmod n$ 的算法

表 9.4　计算 $a^b \bmod n$ 的快速模幂算法，其中 a=7, b=560=1000110000, n=561

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

用公钥进行有效运算　为了加快 RSA 算法在使用公钥时的运算速度，通常都会选择一个特定的 e，大多数情况下选 e 为 65537（2^{16}+1），另外两个常用的选择是 3 和 17。这些指数里都只有 2 位，所以求幂运算时需要的乘法次数应极小化。

然而，若指数太小，如 e=3，RSA 甚至会遭受一些简单的攻击。假设有三个不同的 RSA 用户，

① 该算法已经提出了很长时间；这里给出的伪码取自文献[CORM09]。

他们都使用指数 $e=3$，但他们的模数 n 却各不相同，即 n_1，n_2，n_3。现在有一个用户 A 以加密方式给他们三个发送了相同的消息 M，则三个密文为：$C_1=M^3 \bmod n_1$，$C_2=M^3 \bmod n_2$，$C_3=M^3 \bmod n_3$。很有可能 n_1，n_2，n_3 是两两互素的，所以运用中国剩余定理(CRT)可以计算出 $M^3 \bmod (n_1n_2n_3)$。根据 RSA 的规则，M 应该比模数要小，所以有 $M^3<n_1n_2n_3$，从而攻击者只需要计算 M^3 的三次立方根就可以了。给每一个待加密的消息 M 填充一个唯一的伪随机位串可以阻止这种攻击，随后会讨论这个问题。

读者也许会注意到 RSA 定义（参见图 9.5）里，在密钥产生阶段要求用户选择的 e 要与 $\phi(n)$ 互素。因此，如果用户选择了 e，接着生成了素数 p 和 q，很有可能 $\gcd(\phi(n),e)\neq 1$。因此，用户应该去掉 p 和 q 的值并生成新的 p 和 q。

用私钥进行有效运算 我们不能为了计算的效率而简单地选择一个小数值的 d。d 的值太小容易遭受穷举攻击和其他形式的密码分析[WIEN90]。然而，运用中国剩余定理可以加快运算速度。我们希望计算 $M=C^d \bmod n$。先定义一些中间结果

$$V_p=C^d \bmod p \qquad V_q=C^d \bmod q$$

运用 CRT，式（8.8），定义

$$X_p=q\times(q^{-1} \bmod p) \qquad X_q=p\times(p^{-1} \bmod q)$$

用式（8.9），由 CRT 可得

$$M=(V_pX_p+V_qX_q) \bmod n$$

进一步地，可以用费马定理来简化 V_p 和 V_q 的计算，即根据：如果 p 和 a 互素，则 $a^{p-1}\equiv 1 \pmod p$。不难看出如下式子是正确的：

$$V_p=C^d \bmod p=C^{d \bmod (p-1)} \bmod p \qquad V_q=C^d \bmod q=C^{d \bmod (q-1)} \bmod q$$

$d \bmod (p-1)$ 和 $d \bmod (q-1)$ 可以预先计算出来。与直接计算 $M=C^d \bmod n$ 相比，上述的计算速度大约快 4 倍[BONE02]。

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。

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。

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（如文献[KNUT98]中介绍了若干素性测试方法），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 2 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的 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（如利用伪随机数发生器）。

（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。

这个过程有些烦琐，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（PU，PR）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 $d = e^{-1} \bmod \phi(n)$。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 2 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（参见习题 2.18）。

9.2.3　RSA 的安全性

对 RSA 算法的攻击可能有如下 5 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。
- **基于硬件故障的攻击**　这种方法应用产生签名过程中处理器发生的故障。
- **选择密文攻击**　这种攻击利用了 RSA 算法的性质。

如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。

在本小节里，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：

（1）分解 n 为两个素因子。这样就可以计算出$\phi(n) = (p-1)(q-1)$，从而可以确定 $d \equiv e^{-1} \pmod{\phi(n)}$。

（2）直接确定$\phi(n)$而不先确定 p 和 q。这同样也可以确定 $d \equiv e^{-1} \pmod{\phi(n)}$。

（3）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 Internet 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。和对 DES 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-768，其密钥长度为 232 的十进制位或 768 位位。表 9.5 给出了迄今为止得出的一些结果。

值得注意的是表 9.5 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法（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%。

表 9.5 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	十进制位数	二进制位数（近似值）	完成日期
100	332	1991 年 4 月	155	512	1999 年 8 月
110	365	1992 年 4 月	160	530	2003 年 4 月
120	398	1993 年 6 月	174	576	2003 年 12 月
129	428	1994 年 4 月	200	663	2005 年 5 月
130	431	1996 年 4 月	193	640	2005 年 11 月
140	465	1999 年 2 月	232	768	2009 年 12 月

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。我们已经看到，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，有可能设计出更好的算法。事实上，对某种特殊形式的数，用特殊数域筛法（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。分解 768 位整数的团队[KLEI10]观察到：分解一个 RSA-1024 大约比 RSA-768 难 1000 倍，并且分解一个 RSA-768 比 RSA-512 难几千倍。根据成果分解 512 位到 768 位的时间跨度，通过学术界的努力、完全有理由期望在今后十年内攻破 RSA-1024。因此，谨慎起见，该团队建议在未来几年内（从 2010 年算起）应逐渐淘汰 RSA-1024 的使用。

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 个十进制位）的密钥而言，p 和 q 都应约在 $10^{75}\sim10^{100}$ 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]。

计时攻击 如果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。密码学顾问 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，而且也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密文，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 9.8 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不固定的算法。图 9.8

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，则还需再执行一次模乘运算。

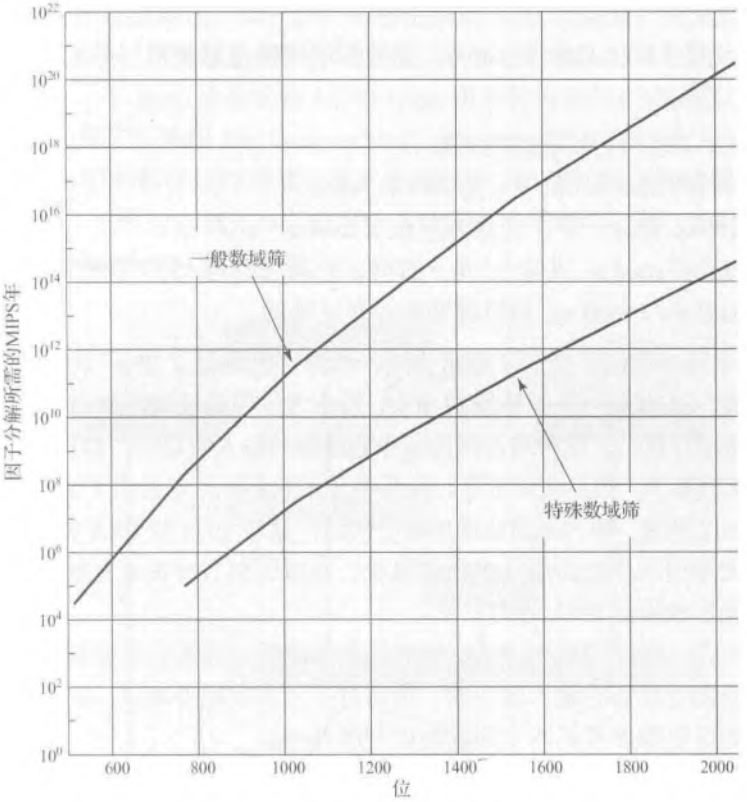


图 9.9　因子分解所需的 MIPS 年

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，使得一次迭代的执行时间会超过整个算法的平均执行时间，但存在有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文献[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 数据安全公司（RSA Data Security）在他们的产品里就使用了隐蔽方法，它用私钥实现操作 $M = C^d \bmod n$ 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 的模 n 的乘法逆元，关于乘法逆元的讨论参见第 2 章。根据 $r^{ed} \bmod n = r \bmod n$，可以证明结论是正确的。

RSA 数据安全公司声称由于使用了隐蔽方法，运算性能降低了 2%～10%。

基于故障的攻击 还有另一种非传统的 RSA 攻击方法见诸于文献[PELL10]。这种方法是对正在产生签名的处理器进行攻击。这种攻击通过减少处理器的输入电功率，以在签名计算中引入故障。故障导致软件产生无效签名，然后攻击者通过分析恢复出私钥。作者展示了如何来完成这样的分析工作且用具体的实例加以阐述，即利用商业通用微处理器，提取 1024 位 RSA 私钥大约需要 100 小时。

这种攻击方法采用引入单位错误并观察其结果。具体的细节可参见文献[PELL10]，文中也参考了一些其他的基于硬件故障的 RSA 攻击方法。

这种攻击值得关注，但还没有对 RSA 构成严重的威胁。因为它需要攻击者能够物理接触目标机器并且能够直接控制处理器的输入电功耗。虽然对于大多数硬件来说，控制输入电功耗要比控制交流电更容易，但是这也要涉及芯片上的供电控制硬件。

选择密文攻击和最佳非对称加密填充 基本的 RSA 算法易受选择密文攻击（CCA）。进行 CCA 攻击时，攻击者选择一些密文，并获得相应的明文，这些明文是利用目标对象的私钥解密获得的。因此，攻击者可以选择一个明文，运用目标对象的公钥加密，然后再用目标对象的私钥解密而取回明文。显然，这么做并没有给攻击者任何新的信息。可是攻击者可以利用 RSA 的性质，选择数据块使得当用目标对象的私钥处理时，产生密码分析所需要的信息。

利用 CCA 攻击 RSA 的一个简单例子利用了 RSA 如下的性质：

$$\mathrm{E}(\mathrm{PU}, M_1) \times \mathrm{E}(\mathrm{PU}, M_2) = \mathrm{E}(\mathrm{PU}, [M_1 \times M_2]) \tag{9.2}$$

利用 CCA 攻击，我们可以如下方式解密 $C = M^e \bmod n$。

1. 计算 $X = (C \times 2^e) \bmod n$。
2. 将 X 作为选择明文提交，并收到 $Y = X^d \bmod n$。

现在请注意

$$\begin{aligned} X &= (C \bmod n) \times (2^e \bmod n) \\ &= (M^e \bmod n) \times (2^e \bmod n) \\ &= (2M)^e \bmod n \end{aligned}$$

因此，$Y = (2M) \bmod n$，由此，我们可以得到 M。为了防止这种简单攻击，实用的基于 RSA 的密码体制在加密之前都会对明文进行随机填充。这使得密文随机化了，从而式（9.2）不再成立。然而，复杂一些的 CCA 攻击仍然是可能的，简单的随机填充已经证明是不足以提高期望的安全性。为了防止这种攻击，RSA 安全公司，也是 RSA 的主要厂商，RSA 专利的持有人，推荐使用一种称为最优非对称加密填充（Optimal Asymmetric Encryption Padding，OAEP）的程序对明文进行修改。

对安全威胁和 OAEP 的全面讨论超出了本书的范围，文献[POIN02]有一些介绍，而文献[BELL94a]则做了彻底的分析。这里，我们对 OAEP 做一些概括性介绍。

图 9.10 显示了 OAEP 加密算法。第一步，待加密的消息 M 被填充。可选参数集 P 作为 Hash 函数 H[①]的输入，输出用 0 加以填充以获得期望的长度，从而可以放入整体数据块 DB 内。接着，随机选择一个种子，并作为一个 Hash 函数的输入，该 Hash 函数称为掩码生成函数（MGF）。输出 Hash 值和 DB 进行按位异或运算产生掩码 DB（maskedDB）。该 maskedDB 反过来又作为 MGF 的输入产生一个 Hash 值，该 Hash 值和种子进行异或运算产生掩码种子。掩码种子和掩码 DB 连接起来构成编码后的消息 EM。注意，EM 包含填充过的消息，该消息由种子进行掩蔽，而种子又由 maskedDB 进行掩蔽。最后用 RSA 对整个 EM 进行加密。

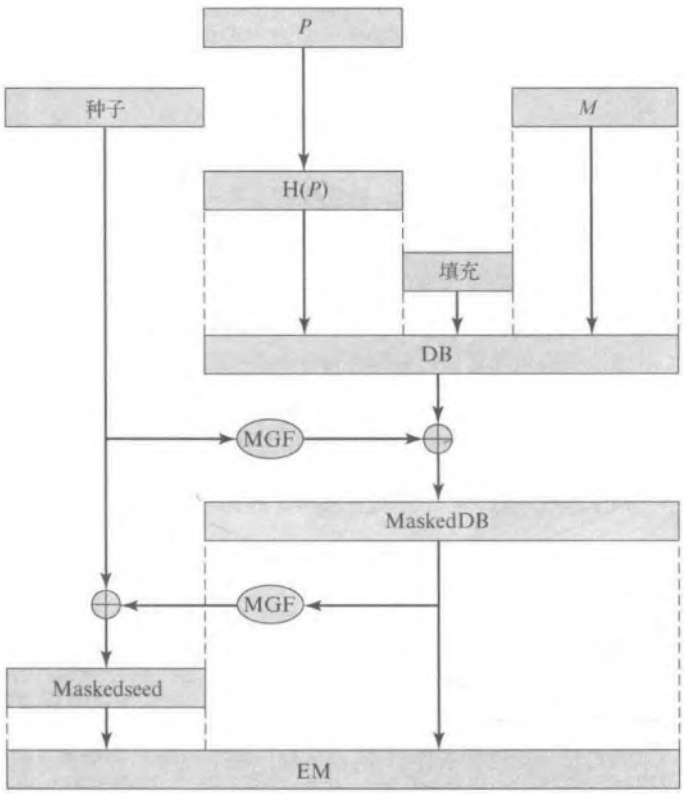


图 9.10　用最优非对称加密填充（OAEP）的加密算法

9.3　关键术语、思考题和习题

关键术语

选择密文攻击（CCA）　　数字签名　　密钥交换　　单向函数

① Hash 函数将变长的数据块或消息映射为定长的值，称为 Hash 码。第 11 章将对 Hash 函数进行深入的讨论。

最优非对称加密填充（OAEP）	私钥	公钥	公钥密码学
公钥密码体制	公钥加密	RSA	计时攻击
单向陷门函数			

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 用一般术语描述一个挑选素数的有效过程。

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 的向量 M_1 表示，其中 M_1 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 $\boldsymbol{M}_2$ 和 $\boldsymbol{M}_3$ 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 $\boldsymbol{M}_1$ 为 1 到 N 上的一个随机置换，即 1 到 N 之间的每个整数在 $\boldsymbol{M}_1$ 中恰好出现一次；$\boldsymbol{M}_2$ 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 $\boldsymbol{M}_3$：

$$f_3(f_2(f_1(k),p),k)=p \qquad \text{对任意的 } k \text{ 和 } p, \text{ 且 } 1 \leqslant k, p \leqslant N$$

也就是说，

（1）$\boldsymbol{M}_1$ 的输入为 k，输出为 x。

（2）$\boldsymbol{M}_2$ 的输入为 x 和 p，输出为 z。

（3）$\boldsymbol{M}_3$ 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 $\boldsymbol{M}_3$。例如，在下述简单的情况下可填充表 $\boldsymbol{M}_3$：

$\boldsymbol{M}_1=$

5
4
3
2
1

$\boldsymbol{M}_2=$

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

$\boldsymbol{M}_3=$

约定：$\boldsymbol{M}_1$ 的第 i 分量对应 $k=i$；$\boldsymbol{M}_2$ 的第 i 行对应 $x=i$，$\boldsymbol{M}_2$ 的第 j 列对应 $p=j$；$\boldsymbol{M}_3$ 的第 i 行对应 $z=i$，$\boldsymbol{M}_3$ 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 9.5 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：解密并不像你想象的那么难，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 是多少？

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？提示：首先用试探法决定 p 和 q，然后用扩展 Euclid 算法寻找 31 模 $\phi(n)$的乘法逆。

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　假设 Bob 使用 RSA 密码体制，其中模数非常大以使得因子分解是不可行的。假设 Alice 给 Bob 发消息，其中的字母表示为 0～25（$A\rightarrow0,\cdots,Z\rightarrow25$）。然后对每个字母用 RSA 算法单独加密，参数 e 和 n 都很大。这种方法安全吗？如果不安全，请给出最有效的攻击方法。

9.9　用电子制表软件（如 Excel）或计算器，做如下指定的运算。记录下所有模乘的中间结果。对每一个主要的变换（如加密，解密，素性测试等）判断其中的模乘运算。

（a）以 2 为底，用 Miller-Rabin 测试算法检测位于 233～241 之间的所有奇数的素性。

（b）用参数为 e=23，n=233×241 的 RSA 算法加密消息分组 M=2。

（c）根据上述给出的公钥(e, n)，计算相应的私钥(d, p, q)。

（d）用如下两种不同的方法解密上述得到的密文：

（1）不用中国剩余定理。

（2）用中国剩余定理。

9.10　假设你按如下方式生成了一个经过认证且加密的消息：首先用私钥及 RSA 算法进行变换，然后用接受方的公钥及 RSA 算法对消息加密（注意，在变换之前没有使用 Hash 函数）。这个方案能正确工作吗［即对于发送方的模数 n_S 和接受方的模数 n_R 之间的所有可能关系($n_S>n_R$，$n_S<n_R$，$n_S=n_R$)，给出接受方可以重构原来的消息的可能性］?对你的回答给出解释。如果你的答案是“否”，那么如何修改呢？

9.11　“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？”福尔摩斯从睡意蒙眬中醒过来。“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，我就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，…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 $2^K \bmod N$, $3^K \bmod N$, $4^K \bmod N$, …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可行的。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：$2^K \bmod N$, $3^K \bmod N$，就很容易计算出后续元素……”

请说明这是如何做到的？

9.12 说明如何通过习题 9.1 中的矩阵 $\boldsymbol{M}_1$，$\boldsymbol{M}_2$ 和 $\boldsymbol{M}_3$ 来描述 RSA。

9.13 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。

9.14 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 $PP' \equiv 1 \pmod{Q-1}$ 且 $QQ' \equiv 1 \pmod{P-1}$；

（4）作为对消息 M 的加密，B 计算 $C=M^N \pmod N$；

（5）A 求解 $M \equiv C^{P'} \pmod Q$ 和 $M \equiv C^{Q'} \pmod P$ 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 $\boldsymbol{M}_1$，$\boldsymbol{M}_2$ 和 $\boldsymbol{M}_3$ 描述这种方法。

9.15 “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以公钥密码体制为基础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 $(A, E(PU_b, [M, A]), B)$ 发送给 B；

（2）B 发送应答 $(B, E(PU_a, [M, B]), A)$ 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 $(A, E(PU_b, M), B)$ 发送给 B；

（2）B 发送应答 $(B, E(PU_a, M), A)$ 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，……”请完成上述的过程。

9.16　运用图 9.8 所示的快速求幂算法，计算 $5^{596} \bmod 1234$。给出计算中的步骤。

9.17　下面是快速求幂算法的另一种实现方法。证明它与图 9.8 中的方法是等价的：

```
1. f←1; T←a; E←b;
2. if odd(E), then f←f×T;
3. E←[E/2]
4. T←T×T;
5. if E>0, then goto 2;
6. output f
```

9.18　这个习题说明了选择密文攻击的简单应用。Bob 截获了一份发给 Alice 的密文 C，该密文是用 Alice 的公钥 e 加密的。Bob 想获得原始消息 $M=C^d \bmod n$。Bob 选择一个小于 n 的随机数 r，并计算

$$Z = r^e \bmod n$$

$$X = ZC \bmod n$$

$$t = r^{-1} \bmod n$$

接着，Bob 让 Alice 用她的私钥对 X 进行认证（如图 9.3 所示），从而对 X 进行解密。Alice 返回 $Y=X^d \bmod n$。请说明 Bob 如何利用获得的信息去求取 M。

9.19　图 9.10 给出了 OAEP 编码算法，请给出对应的 OAEP 解码算法用于解密。

9.20　改进附录 W 中的算法 P1。

（a）设计一个需要 $2n$ 次乘法和 $n+1$ 次加法的算法。提示：$x^{i+1}=x^i \times x$。

（b）设计一个仅需要 $n+1$ 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。

关于背包公钥密码算法详见附录 J

9.21　图 F.1 的背包体制里有哪些项？

9.22　运用背包体制对下列参数做加密和解密运算：

（a）$\boldsymbol{a}'=(1,3,5,10)$; $w=7$; $m=20$; $\boldsymbol{x}=1101$

（b）$\boldsymbol{a}'=(1,3,5,11,23,46,136,263)$; $w=203$; $m=491$; $\boldsymbol{x}=11101000$

（c）$\boldsymbol{a}'=(2,3,6,12,25)$; $w=46$; $m=53$; $\boldsymbol{x}=11101$

（d）$\boldsymbol{a}'=(15,92,108,279,563,1172,2243,4468)$; $w=2393$; $m=9291$; $\boldsymbol{x}=10110001$

9.23　为何要求 $m > \sum_{i=1}^{n} a_i'$？

第 10 章　密钥管理和其他公钥密码体制

学 习 目 标

通过本章的学习应具备以下能力：

- 掌握 Diffie-Hellman 密钥交换协议
- 理解中间人攻击
- 给出离散对数密系统的概述
- 理解椭圆曲线算法
- 给出椭圆曲线密码系统的概述
- 介绍用非对称密码产生伪随机数的两种技术

本章首先描述了一种最早最简单的 PKCS：Diffie-Hellman 密钥交换，同时关注 ElGamal PKCS。接下来还有日益重要的椭圆曲线密码。最后检测伪随机数生成器的公开密钥算法的应用。

10.1　Diffie-Hellman 密钥交换

Diffie 和 Hellman 在一篇具有独创意义的论文[DIFF76b]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 算法的有效性是建立在计算离散对数很困难的这一基础上的。简单地说，我们可如下定义离散对数。首先定义素数 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 $p-1$ 之间所有整数，也就是说若 a 是素数 p 的本原根，则

$$a \bmod p, a^2 \bmod p, \cdots, a^{p-1} \bmod p$$

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 a，我们可以找到唯一的指数使得

$$b \equiv a^i \bmod p, \quad \text{其中 } 0 \leqslant i \leqslant (p-1)$$

指数 i 称为 b 的以 a 为底的模 p 离散对数，记为 $\mathrm{dlog}_{a,p}(b)$。有关离散对数的进一步讨论请见第 2 章。

10.1.1　算法

图 10.1 概述了 Diffie-Hellman 密钥交换算法。在这种方法中，素数 q 和其本原根 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

① 英国 CSEG 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，请参见文献[ELLI99]的有关讨论。

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

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \qquad \text{根据模算术的运算规律} \\
&= (\alpha^{X_B X_A}) \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

至此双方完成了密钥的交换。通常这个秘密值被用于对称密码的密钥。现在考虑一个敌手能够观察到密钥交换的全过程并且期望得到这个秘密 K。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

$$X_B = \mathrm{dlog}_{\alpha,q}(Y_B)$$

然后他就可以像用户 B 那样计算出密钥 K。

$$K = (Y_A)^{X_B} \bmod q$$

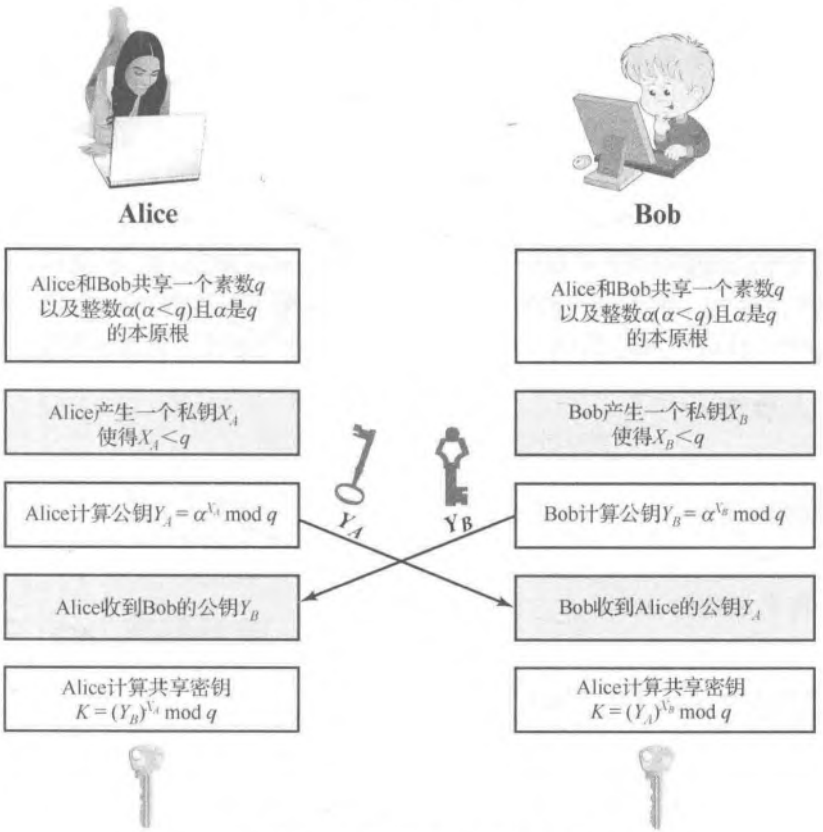


图 10.1　Diffie-Hellman 密钥交换

Diffie-Hellman 密钥交换的安全性建立在下述事实之上：求关于素数的模素数幂运算相对容易，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 分别选择密钥 $X_A = 97$ 和 $X_B = 233$，并计算相应的公钥

A 计算 $Y_A = 3^{97} \bmod 353 = 40$

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

A 计算 $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$

B 计算 $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

我们假定攻击者能够得到下列信息：

$$q = 353; \quad \alpha = 3; \quad Y_A = 40; \quad Y_B = 248$$

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 $3^{97} \bmod 353 = 40$，所以指数为 97 时可得到期望的结果。

对于较大的数，上述方法实际是不可行的。

10.1.2 密钥交换协议

图 10.1 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。

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秘钥 X_i（用户 i），并计算公开的 Y_i。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 j 都可以访问用户 i 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 i 和 j 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 i 知道只有用户 j 能用该密钥产生消息（真实性）。但是这种方法不能抗重播攻击。

10.1.3 中间人攻击

图 10.1 所示的协议不能抵抗所谓的中间人攻击。假定 Alice 和 Bob 希望交换密钥，而 Darth 是攻击者。攻击过程如下：

（1）为了进行攻击，Darth 先生成两个随机的私钥 X_{D1} 和 X_{D2}，然后计算相应的公钥 Y_{D1} 和 Y_{D2}。

（2）Alice 将 Y_A 传递给 Bob。

（3）Darth 截获了 Y_A，将 Y_{D1} 传给 Bob。Darth 同时计算 $K2 = (Y_A)^{X_{D2}} \bmod q$。

（4）Bob 收到 Y_{D1}，计算 $K_1 = (Y_{D1})^{X_B} \bmod q$。

（5）Bob 将 Y_B 传给 Alice。

（6）Darth 截获了 Y_B，将 Y_{D2} 传给 Alice。Darth 计算 $K1 = (Y_B)^{X_{D1}} \bmod q$。

（7）Alice 收到 Y_{D2}，计算 $K_2 = (Y_{D2})^{X_A} \bmod q$。

此时，Bob 和 Alice 想，他们已共享了密钥，但实际上，Bob 和 Darth 共享密钥 K_1，而 Alice 和 Darth 共享密钥 K_2。接下来，Bob 和 Alice 之间的通信以下列方式泄密：

（1）Alice 发了一份加了密的消息 M：E($K2$，M)。

（2）Darth 截获了该密消息，解密，恢复出 M。

（3）Darth 将 E(K_1，M)或 E(K_1，M')发给 Bob，其中 M'是任意的消息。第一种情况，Darth 只是简单地窃听通信，而不是改变它。第二种情况，Darth 想修改给 Bob 的消息。

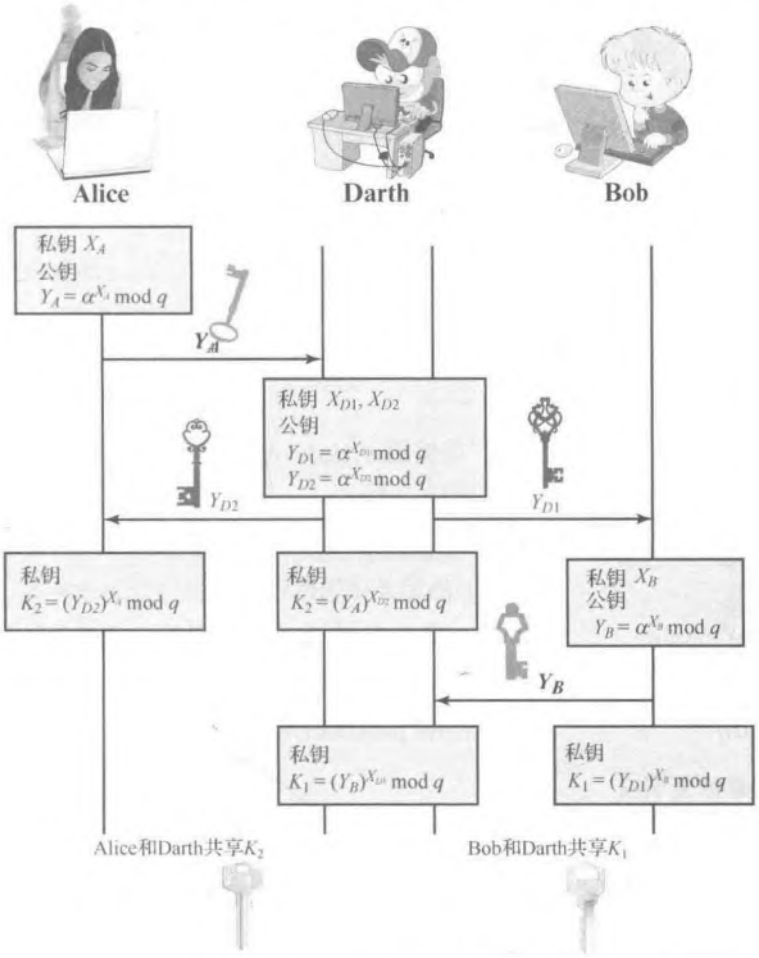


图 10.2　Diffie-Hellman 密钥交换

密钥交换协议不能抵抗上述的攻击，因为它没有对通信的参与方进行认证。这些缺陷可以通过使用数字签名和公钥证书来克服。这些主题将会在第 13 章和第 14 章讨论。

10.2　ElGamal 密码体制

1984 年，T. Elgamal 提出了一种基于离散对数的公开密钥体制，一种与 Diffie-Hellman 密钥分配体制[ELGA84,ELGA85]密切相关。ElGamal 密码体系应用于一些技术标准中，如数字签名标准（DSS）和 S/MIME 电子邮件标准（参见第 19 章），数字签名标准将在第 13 章中论述。

与 Diffie-Hellman 一样，ElGamal 的系统用户也是共同选择一个素数 q，α 是 q 的素根。用户 A 生成的密钥对如下：

（1）随机生成整数 X_A，使得$1 < X_A < q-1$。

（2）计算 $Y_A = \alpha^{X_A} \bmod q$。

（3）A 的私钥为 X_A，公开密钥为 $\{q, \alpha, Y_A\}$。

其他任何用户 B 通过 A 的公开密钥可以加密信息：

（1）将信息表示为一个整数 M，其中 $1 \leqslant M \leqslant q-1$，以分组密码序列的方式来发送信息，其中每个分块的长度不小于整数 q。

（2）选择任意整数 k，使得 $1 \leqslant k \leqslant q-1$。

（3）计算一次密钥 $K = (Y_A)^k \bmod q$。

（4）将 M 加密成明文对 (C_1, C_2)，其中

$$C_1 = \alpha^k \bmod q; \quad C_2 = KM \bmod q$$

用户 A 恢复明文：

（1）通过计算 $K = (C_1)^{X_A} \bmod q$ 恢复密钥。

（2）计算 $M = (C_2 K^{-1}) \bmod q$。

这些步骤在表 10.3 中有具体的总结。根据图 9.1(a)：Alice 生成公开和保密的密钥对；Bob 用 Alice 的公开密钥加密，然后 Alicce 用自己的私钥解密。

接下来具体阐述 ElGamal 体系的工作原理。首先，我们来看 K 被解密恢复的过程：

$K = (Y_A)^k \bmod q$	K 定义在加密过程中
$K = (\alpha^{Y_A} \bmod q)^k \bmod q$	用 $Y_A = \alpha^{X_A} \bmod q$ 来替换
$K = \alpha^{kX_A} \bmod q$	利用同余定理
$K = (C_1)^{X_A} \bmod q$	用 $C_1 = \alpha^k \bmod q$ 来替换

接下来，用 K 来恢复明文

$$C_2 = KM \bmod q$$
$$(C_2 K^{-1}) \bmod q = KMK^{-1} \bmod q = M \bmod q = M$$

我们能够用下面的图 10.3 来重述 ElGamal 过程。

（1）Bob 生成任意整数 k。

（2）Bob 用 Alice 的公开密钥 $\{q, k, Y_A\}$ 生成一个一次密钥 K。

（3）Bob 用 α 加密 k，得到 (C_1, C_2)，这给 Alice 足够的信息恢复 K。

（4）Bob 用 K 加密明文 M。

（5）Alice 用自己的私钥从 C_1 恢复出 K。

（6）Alice 用 K^{-1} 通过 C_2 恢复明文信息。

因此，K 将作为一次性密钥，用于加密和解密信息。

例如，对于素域 GF(19)，即 q=19。素根有{2，3，10，13，14，15}。如表 2.7 所示，选择 $\alpha = 10$。Alice 生成如下的密钥对：

（1）Alice 选择 $X_A = 5$。

（2）计算 $Y_A = \alpha^{X_A} \bmod q = \alpha^5 \bmod 19 = 3$（参见表 2.7）。

（3）Alice 的私钥为 5；公钥为 $\{q, \alpha, Y_A\}$={19,10,3}。

公开全局量	
q	素数
α	$\alpha < q$且α是q的素根

Alice 生成的密钥	
选择私钥 X_A	$X_A < q-1$
计算 Y_A	$Y_A = \alpha^{X_A} \bmod q$
公开密钥	$\{q, \alpha, Y_A\}$
私钥	X_A

Bob 用 Alice 的公钥加密	
明文	$M < q$
随机选择整数 k	$k < q$
计算 K	$K = (Y_A)^k \bmod q$
计算 C_1	$C_1 = \alpha^k \bmod q$
计算 C_2	$C_2 = KM \bmod q$
密文	(C_1, C_2)

用 Alicce 的私钥解密	
密文	(C_1, C_2)
计算 K	$K = (C_1)^{X_A} \bmod q$
明文	$M = (C_2 K^{-1}) \bmod q$

图 10.3　ElGamal 密码体制

假如 Bob 想将值 M=17 发送，则

（1）Bob 选择 k=6。

（2）计算 $K = (Y_A)^k \bmod q = 3^6 \bmod 19 = 729 \bmod 19 = 7$。

（3）因此

$$C_1 = \alpha^k \bmod q = \alpha^6 \bmod 19 = 11$$
$$C_2 = KM \bmod q = 7 \times 17 \bmod 19 = 119 \bmod 19 = 5$$

（4）Bob 发送密文(11, 5)。

解密：

（1）Alice 计算 $K = (C_1)^{X_A} \bmod q = 11^5 \bmod 19 = 161051 \bmod 19 = 7$。

（2）在 GF(19)中 K^{-1} 为 $7^{-1} \bmod 19 = 11$。

（3）最终 $M = (C_2 K^{-1}) \bmod q = 5 \times 11 \bmod 19 = 55 \bmod 19 = 17$。

如果信息必须分组然后以加密的密钥块序列发送，那么每个分块要有唯一的 k。如果 k 用于多个分块。利用信息的分块 M_1，攻击者会计算出其他块。

$$C_{1,1} = \alpha^k \bmod q; C_{2,1} = KM_1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q; C_{2,2} = KM_2 \bmod q$$

于是

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

如果 M_1 已知，则很容易计算出 M_2

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

ElGamal 的安全性是基于计算离散对数的困难性之上。为了恢复 Alice 的私钥，攻击者将会计算 $X_A = \mathrm{d}\log_{\alpha,q}(Y_A)$。或者，为了恢复一次性密钥 K，攻击者将会选择随机数 k，然后计算离散对数 $k = \mathrm{d}\log_{\alpha,q}(C_1)$。文献[STIN06]指出，当 $p \geqslant 300$，$q-1$ 至少有一个大的素因子时，这种算法是不可行的。

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。

ECC 比 RSA 或 Diffie-Hellman 更难阐述，关于 ECC 的完整的数学描述已超出本书范围。本节和下一节中，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，然后讨论定义在实数域上的椭圆曲线，紧接着讨论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，读者可回顾第 5 章中关于有限域的内容。

10.3.1 Abel 群

由第 5 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 中元素的序偶 (a, b) 与 G 中元素 $(a \cdot b)$ 对应，使得下述公理[①]成立：

（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，$e \cdot a = a \cdot e = a$。

（A4）逆元 对 G 中任何 a，存在 G 中元素 a'，使得 $a' \cdot a = a \cdot a' = \mathrm{e}$。

（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

$$a^k \bmod q = \underbrace{(a \times a \times \cdots \times a)}_{k\text{次}} \bmod q$$

要攻击 Diffie-Hellman，攻击者必须对给定 a 和 a^k 确定 k，这就是求离散对数问题。

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。

椭圆曲线密码学使用椭圆曲线上称之为加法的运算，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及系数的方程。对于密码学而言，变元和系数均限制于有限域上，这导致了有限 Abel 群的定义。在讨论这些之前，先讨论变元和系数均是实数的椭圆曲线，这种情况也许更容易想象。

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，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，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 10.4 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。我们考虑满足式（10.1）的所有的点（x, y）和元素 O 所组成的点集 E(a, b)。偶对(a, b)的值不同，则相应集合 E(a, b)也不同。使用上述术语，图 10.4 中的两条曲线可分别用集合 E(–1, 0)和 E(1, 1)表示。

加法的几何描述　式（10.1）里的参数 a 和 b，如果满足条件

$$4a^3+27b^2\neq 0 \tag{10.2}$$

则可基于集合 E(a, b)定义一个群。为了在 E(a, b)上定义一个群，必须定义一个运算，称为加法，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 坐标和相反的 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 间画一条直线并找出第三个交点 R，显然存在有唯一的交点 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 10.4 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

① 注意：x 和 y 真正的变元，它们具有值。这与第 5 章中讨论的多项式环和域不同，那里 x 被看成不定元。

连接这两点，这也可看成是在无穷远点处与曲线相交，因此有 $P+(-P)=O$，与上述步骤（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。

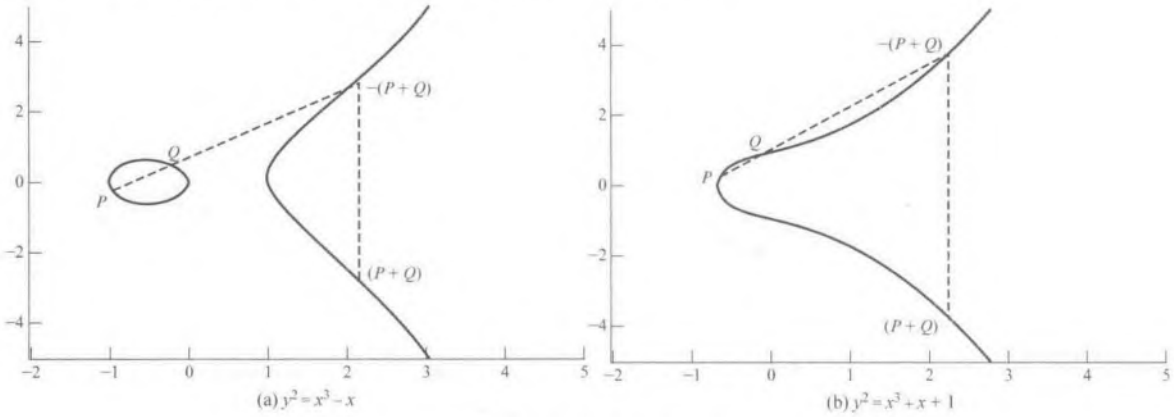


图 10.4 椭圆曲线举例

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$\Delta=(y_Q-y_P)/(x_Q-x_P)$。$l$ 恰与椭圆曲线相交与另一点，即 P 与 Q 之和的负元。利用某些代数运算，可如下表示和 $R=P+Q$：

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

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 curve）和在 GF(2^m)上构造的二元曲线。对于 Z_p 上的素曲线，使用三次方程，其中的变量和系数自集合{0,1,⋯,p−1}取值，运算为模 p 运算。对于 GF(2^m) 上的二元曲线，变量和系数在 GF(2^m)内取值，且运算为 GF(2^m)里的运算。文献[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，它可以用非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，用于实数域上椭圆曲线算术的代数解释可以移植过来，我们将采用这种方法。

① 关于这些结论的推导请参阅文献[KOBL94]或其他关于椭圆曲线的数学资料。

对 Z_p 上的椭圆曲线，如同实数情形一样，仅限于讨论形如式（10.1）的方程。此时，变元和系数均在 Z_p 中取值

$$y^2 \bmod p = (x^3+ax+b) \bmod p \tag{10.5}$$

例如，$a=1, b=1, x=9, y=7, p=23$ 时可满足式（10.5）

$$7^2 \bmod 23 = (9^3+9+1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3=3$$

下面考虑由所有满足式（10.5）的整数对(x, y)和无穷远点 O 组成的集合 $E_p(a, b)$。系数 a 和 b，变量 x 和 y 都是 Z_p 的元素。

例如，取 $p=23$。考虑椭圆曲线方程 $y^2=x^3+x+1$，其中 $a=b=1$。注意，该方程与图 10.4(b)中的方程是相同的。图中显示了所有满足方程的实点。对 $E_{23}(1,1)$，我们只对如下非负整数感兴趣，它们位于从(0,0)到$(p-1, p-1)$的象限中，满足模 p 的方程。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 10.5 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。

表 10.1　椭圆曲线 $E_{23}(1,1)$上的点（除了 O）

(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：

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \tag{10.6}$$

注意式（10.6）和式（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，$Q \in E_p(a, b)$

（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 $4P=P+P+P+P$。

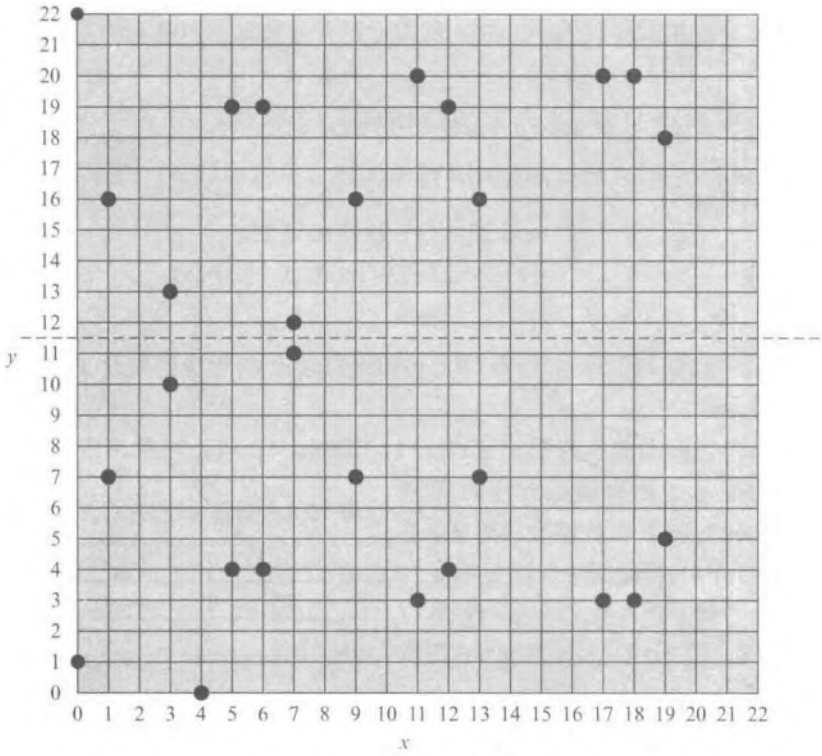


图 10.5 椭圆曲线 $E_{23}(1,1)$

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。这可以用 4.4 节定义的扩展 Euclid 算法实现。注意到(6×4) mod 23=24 mod 23=1。

$$x_R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 7$$

$$y_R = (6(3-7) - 10) \bmod 23 = (-34) \bmod 23 = 12$$

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

$$p + 1 - 2\sqrt{p} \leq N \leq p + 1 + 2\sqrt{p}$$

所以，$E_p(a, b)$上点的个数约等于 Z_p 中元素的个数，即 p 个元素。

10.3.4　GF(2^m)上的椭圆曲线

由第 5 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 $E_{2^m}(a, b)$。

例如，使用不可约多项式 $f(x)=x^4+x+1$ 定义的有限域 GF(2^4)，其生成元满足 $f(g)=0$，即 $g^4=g+1$，或者二进制 $g=0011$。我们可以如下给出 g 的各次方：

g^0=0001	g^4=0011	g^8=0101	g^{12}=1111
g^1=0010	g^5=0110	g^9=1010	g^{13}=1101
g^2=0100	g^6=1100	g^{10}=0111	g^{14}=1001
g^3=1000	g^7=1011	g^{11}=1110	g^{15}=0001

例如，$g^5=(g^4)(g)=(g+1)(g)=g^2+g=0110$。

现在考虑椭圆曲线 $y^2+xy=x^3+g^4x^2+1$。此时，$a=g^4$，$b=g^0=1$。满足该方程的一个点为(g^5,g^3)

$$(g^3)^2+(g^5)(g^3)=(g^5)^3+(g^4)(g^5)^2+1$$
$$g^6+g^8=g^{15}+g^{14}+1$$
$$1100+0101=0001+1001+0001$$
$$1001=1001$$

表 10.2 列出了 $E_{2^4}(g^4,1)$ 的部分点（没有点 O）。图 10.6 画出了这些点。

可以证明，只要 $b\neq0$，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

$$\lambda=\frac{y_Q+y_P}{x_Q+x_P}$$

4．若 $P=(x_P,y_P)$，则 $R=2P=(x_R,y_R)$由下列规则确定：

$$x_R=\lambda^2+\lambda+a$$
$$y_R=x_P^2+(\lambda+1)x_R$$

其中

$$\lambda=x_P+\frac{y_P}{x_P}$$

表 10.2 椭圆曲线 $E_{2^4}(g^4,1)$ 上的点（除了点 O）

(0,1)	(g^5,g^3)	(g^9,g^{13})
$(1,g^6)$	(g^5,g^{11})	(g^{10},g)
$(1,g^{13})$	(g^6,g^8)	(g^{10},g^8)
(g^3,g^8)	(g^6,g^{14})	$(g^{12},0)$
(g^3,g^{13})	(g^9,g^{10})	(g^{12},g^{12})

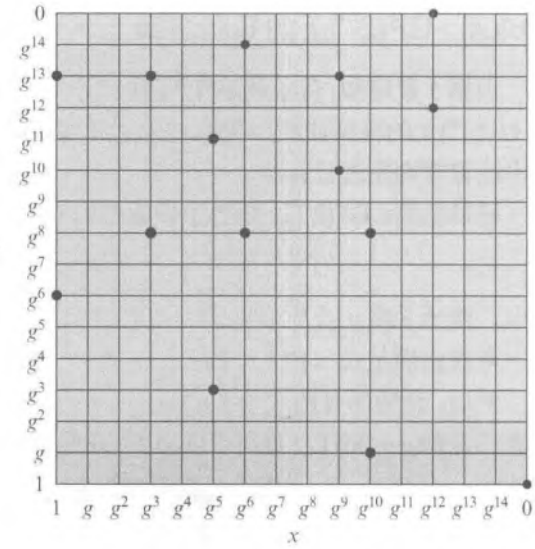


图 10.6 椭圆曲线 $E_{2^4}(g^4,1)$

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，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。这就是椭圆曲线的离散对数问题。

引用 Certicom 网站(www.certicom.com)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？穷举攻击方法通过多次计算 P 的倍数直至找到 Q 为止。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

利用椭圆曲线可如下实现密钥交换。首先，挑选一个大整数 q 以及式（10.5）或式（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。椭圆曲线上的点 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 之间完成密钥交换过程如下（参见图 10.7）：

（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 $121(130,203)=203(115,48)=(161,69)$。

全局公开量	
$E_q(a,b)$	参数为 a,b 和 q 的椭圆曲线，其中 q 是素数或形为 2^m 的整数
G	阶为 n 的椭圆曲线上的点，其中 n 是大整数

用户 A 的密钥产生	
选择私有的 n_A	$n_A<n$
计算公开的 P_A	$P_A=n_A\times G$

用户 B 的密钥产生	
选择私有的 n_B	$n_B<n$
计算公开的 P_B	$P_B=n_B\times G$

用户 A 产生秘密钥
$K=n_A\times P_B$

用户 B 产生秘密钥
$K=n_B\times P_A$

图 10.7　ECC Diffie-Hellman 密钥交换

请注意，这里的密钥是一对数字。若将它用于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，本小节介绍也许是其中最简单的一种方法。首先，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

① 本例由 Santa Clara 大学的 ED Schaefer 提供。

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

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。

让我们来考虑一个简单的实例。公钥全局量取 $q = 257$；$E_q(a,b)=E_{257}(0,-4)$，即其椭圆曲线方程为 $y^2 = x^3 - 4$，$G=(2, 2)$。Bob 的私钥为 $n_B = 101$，他的公钥为 $P_B = n_B G=101(2, 2)=(197,167)$。Alice 选取随即整数 $k = 41$ 并且计算 $kG = 41(2,2)=(136, 128)$，$kP_B=41(197, 167)=(68, 84)$以及 $P_m+kP_B=(112, 26)+(68, 84)=(246, 174)$。Alice 发送密文 $C_m=(C_1, C_2)=\{(136, 128), (246, 174)\}$给 Bob。Bob 收到密文并计算 $C_2-n_BC_1=(246, 174)-101(136, 128)=(246, 174)-(68, 84)=(112, 26)$。

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 方法是已知的求椭圆曲线对数的最快方法。根据 NIST SP800-57（密钥管理建议——第一部分：通用版，2015 年 9 月发布），表 10.3 从密码分析所需计算量的角度，通过给出可比较的密钥大小，比较了各种算法。由表可知，ECC 使用的密钥比 RSA 中使用的密钥要短得多。

基于这种分析，SP 800-57 建议至少到 2030 年，可接受的密钥长度对于 RSA 为 3072～14360 位、对于 ECC 则为 256～512 位。类似地，欧洲网络与信息安全管理局（ENISA）在其 2014 年报告（算法、密钥尺寸及参数报告–2014, 2014 年 11 月发布）中建议未来系统使用密钥长度为 3072 位的 RSA 或 256 位的 ECC。

分析表明在密钥长度相同时，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 10.3 可比较密钥大小（密码分析所需计算量的角度）（NIST SP-800-57）

对称方案（密钥位）	Diffie-Hellman 数字签名算法	RSA（模数位）	基于 ECC 方案（n 的位）
80	L=1024, N=160	1024	160～223
112	L=2048, N=224	2048	224～255
128	L=3072, N=256	3072	256～383
192	L=7680, N=384	7680	384～511
256	L=15360, N=512	15360	512+

注：L=公钥的大小，N=私钥的大小。

10.5 基于非对称密码的伪随机数生成器

在第 8 章中我们提到，因为对称分组密钥流产生的接近随机的输出，它可以作为伪随机数生成器（PRNG）的一组基。同样地，一个非对称的加密算法也能得到接近随机的输出以此来创建一个 PRNG。由于非对称算法的速度比对称算法的速度要明显慢一些，所以非对称算法不用于生成可扩展的 PRNG 位流。但是，对于生成短的伪随机序列，创建伪随机函数（PRF）非对称的方法还是很有用的。

在本节中，将检测两种基于伪随机函数的 PRNG 的设计。

10.5.1 基于 RSA 的 PRNG

对于足够长的密钥，RSA 算法被认为是安全的也是具有很好的条件来形成 PRNG 的一组基。这

种 PRNG，像已知的 Micali-Schnorr PRNG[MICA91]，是作为 ANSI 标准 X9.82（随机数生成）和 ISO 标准 18031（随机位生成）。

在图 10.8 中给出了 PRNG 的模式，从中可以看出，PRNG 在结构上与用于 PRNG 的反馈输出模式有很多相同之处。[参见图 8.4(b)和图 7.6(a)]。在这种情况下加密算法采用 RSA 要比对称分组密码好。与此同时，输出的一部分又反馈到下一个加密算法的循环中，剩下的输出则作为伪随机位。将输出分为这样的两种不同的部分的动机是因为，伪随机位从一个状态到另一个状态时不会泄露输入。这种分法一定会对其发展起到不可预测的贡献。

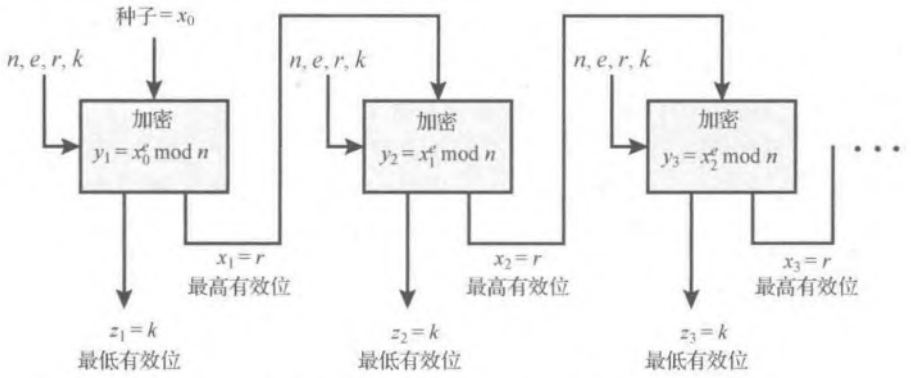


图 10.8　Mic*a*li-Schnorr 伪随机位发生器

PRNG 的定义如下：

创建　选择素数 p,q；$n = pq$；$\varphi(n) = (p-1)(q-1)$，选择 e 使得 $\gcd(e,\varphi(n)) = 1$。这就是标准的 RSA 的参数选择（参见图 9.5）。此外，$N = [\log_2 n] + 1$（n 的位长度）。选择 r,k，使得 $r + k = N$。

种子　选择位长度为 r 的随机种子 x_0。

生成　生成一个长为 $k \times n$ 的伪随机序列，将 i 从 1 到 m，计算

$$y_i = x_{i-1}^e \bmod n$$
$$x_i = r，\ y_i\text{的最高有效位}$$
$$z_i = k，\ y_i\text{的最低有效位}$$

输出　输出序列为 $z_1 \| z_2 \| \cdots \| z_m$。

参数 n，r，e，k 的选择要遵循下面 6 个要求：

（1）$n = pq$	n 是两个素数的乘积，在 RSA 中则是密码长度。
（2）$1 < e < \varphi(n)$；$\gcd(e,\varphi(n)) = 1$	确保映射 $s \to s^e \bmod n$ 是一对一的。
（3）$re \geqslant 2N$	确保求幂运算是有取模过程。
（4）$r \geqslant 2$	防止密码攻击。
（5）k,r 是 8 的倍数	应用方便。
（6）$k \geqslant 8; r + k = N$	所有位都有效。

在要求（4）中对于有用长度在 NIST SP 800-90 中定义为在攻破一个密码算法或者系统的工作相关量（即生成器的数量）；一个安全的长度是规定好的。同时也是从规定的集合（112，128，192，256）中的一个特定值。所需的总工作量是 2^{strength}。

明显可以看到 r 和 k 之间的平衡。因为与分组密码相比而言，RSA 的计算能力更强，我们希望

每个循环尽可能生成更多的伪随机位，同时又想 k 取较大值。尽管如此，对于密码的长度，我们还是希望 r 尽可能大一些。

例如，如果 e=3, N=1024。则有不等式 $3r$>1024，于是 r 只能取 683 个位。当 r 取这个大小时，k=341，即每个求幂运算生成的位（任何 RSA 编码）。在这种情况下，每个求幂运算只需一个模取 683 个位数的平方和一个模的倍数。即 $(x_i \times (x_i^2 \bmod n)) \bmod n$ 。

10.5.2 基于椭圆曲线密码的 PRNG

在本小节中，主要归纳由美国国家安全局（NSA）公布的对偶椭圆曲线 PRNG(DEC PRNG)技术。这种技术在 NIST SP800-90，ANST 标准 X9.82 和 ISO 标准 18031 中介绍过，与其他候选算法（如文献[SCHO06]，[BROW07]）相比，其安全性和效率要更好一些。

文献[SCHO06]总结的算法如下：让 P 和 Q 是给定椭圆曲线的两个已知点。这个 DEC PRNG 的种子为一个随机整数 $s_0 \in \{0,1,\cdots \#\mathrm{E}(\mathrm{GF}(p)-1)\}$，其中 $\#\mathrm{E}(\mathrm{GF}(p)-1)$ 表示椭圆曲线的解点数。让 x 表示曲线上的 x 轴上得到点的函数，$\mathrm{lsb}_i(s)$ 表示整数 s 的最小有意义位。DEC PRNG 将种子转变为长度为 240k，k>0 的伪随机序列的过程为

将 i=0 到 k
　将 s_ix←(s_{i-1} P)
　将 r_i←lsb_{240}(x(s_i Q))
结束
　返回 $r_1,\cdots,r_k$

如果这个 PRNG 收到安全性的影响。那么它唯一的动机就是用于与 ECC 互补但是不和用于建构 PRNG 中用到的对称的、非对称的或者 Hash 密码算法的系统中。

10.6 关键术语、思考题和习题

关键术语

Abel 群	二元曲线	三次方程	Diffie-Hellman 密钥交换
离散对数	椭圆曲线	椭圆曲线算术	椭圆曲线密码学
有限域	中间人攻击	Micali-Schnorr	素曲线
本原根	零点		

思考题

10.1 简要说明 Diffie-Hellman 密钥交换。

10.2 什么是椭圆曲线？

10.3 什么是椭圆曲线的零点？

10.4 椭圆曲线上同在一条直线上的三个点的和是什么？

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 α = 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，公用素数 $q=11$，本原根 $\alpha=2$。

（a）证明 2 是 11 的本原根。

（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 在 Diffie-Hellman 协议里，每一方都选择一个秘密参数 x，给对方发送 $\alpha^x \bmod q$，其中 α 是一个公开的数。如果通信参与方给对方发送的是 x^α，α 是一个公开的数，将会怎样？请给出至少一种方法使得 Alice 和 Bob 可以协商密钥。Eve 在不知道秘密参数的情况下能攻击你的系统吗？Eve 能找到秘密参数吗？

10.4 这个习题表明如果没有取模的步骤，则 Diffie-Hellman 协议是不安全的，即非离散对数不是难题！假设你是 Eve，你已捕捉了 Alice 和 Bob，并监控了他们。你窃听到如下对话：

Bob：哦，我们不要再为 Diffie-Hellman 协议里的素数费心了，那样将会更简单。

Alice：好吧，但我们仍需要一个基 α，那么 g=3 如何？

Bob：好啊，如此，我的结果是 27。

Alice：我的是 243。

Bob 的秘密参数 X_B，Alice 的秘密参数 X_A 是多少？他们的共享密钥是多少（别忘了写出你的过程）？

10.5 10.1 节介绍了针对 Diffie-Hellman 密钥交换协议的中间人攻击。攻击者生成了两组公-私钥对，请解释如果只生成一组公-私钥对可以完成攻击吗？

10.6 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？

10.7 根据实数域上椭圆曲线算术中的规则(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。但是若切线是垂直的，那么在这种情况下，$2Q$ 的值是多少？$3Q$ 的值是多少？

10.8 证明图 10.4 中两条实数域上的椭圆曲线均满足群的条件。

10.9 点（4, 7）在椭圆曲线 $y^2=x^3-5x+5$（定义在实数域上）上吗？

10.10 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 $2P$。

10.11 椭圆曲线方程 $y^2=x^3+10x+5$ 在 Z_{17} 上能定义一个群吗？

10.12 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。

10.13 下列 Z_{17} 上椭圆曲线的点的负数为几？P=(5, 8)；Q=(3, 0)；R= (0, 6)。

10.14 对 $E_{11}(1,6)$ 上的点 $G=(2,7)$，计算 $2G$ 到 $13G$ 的值。

10.15 利用 10.4 节中所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。

10.16 下面是椭圆曲线签名方案的一种尝试。我们有一个全局的椭圆曲线，素数 p 和“生成元” G。Alice 挑一个秘密的签名密钥 X_A，并产生公开的验证密钥 $Y_A=X_AG$。为了对消息 M 签名：

- Alice 挑一个值 k。
- Alice 将 M，k，以及签名 $S=M-kX_AG$ 发送给 Bob。
- Bob 验证 $M=S+kY_A$

（a）请说明该方案的工作原理。即说明验证过程可以说明签名是有效的。

（b）通过描述对任意消息的伪造签名来说明该方案是不可接受的。

10.17 下面是上述习题方案的一种改进。如前，我们有一个全局的椭圆曲线，素数 p 和“生成元” G。Alice 挑一个秘密的签名密钥 X_A 并产生公开的验证密钥 $Y_A=X_AG$。为了对消息 M 签名：

- Bob 挑一个值 k。
- Bob 将 $C_1=kG$ 发给 Alice。
- Alice 将 M，以及签名 $S=M-X_AC_1$ 发送给 Bob。
- Bob 验证 $M=S+kY_A$

（a）请说明该方案的工作原理。即说明验证过程可以说明签名是有效的。

（b）说明在这个方案里伪造签名的困难性和攻击（ElGamal）椭圆曲线密码一样难（或者找出一种简单的办法去伪造消息）。

（c）与我们看过的其他密码体制和签名方案比较，本方案有一个额外的“传递”。这有何缺点？

第四部分　密码学中的数据完整性算法

第 11 章　密码学 Hash 函数
第 12 章　消息认证码
第 13 章　数字签名

第 11 章　密码学 Hash 函数

学 习 目 标

通过本章的学习应具有以下能力：

- 总结密码学 Hash 函数的应用
- 解释 Hash 函数用于消息认证时为什么要保证其安全性
- 理解抗原像攻击、抗第二原像攻击和抗强碰撞攻击之间的区别
- 给出密码学 Hash 函数的基本结构
- 描述如何使用分组密码链接构造 Hash 函数
- 理解 SHA-512 的运算过程
- 理解生日悖论，给出生日攻击的过程

Hash 函数 H 将可变长度的数据块 M 作为输入，产生固定长度的 Hash 值 $h=H(M)$。一个“好”的 Hash 函数具有这样的特点：对于大的输入集合使用该函数，产生的输出结果均匀地分布且看起来随机。概括地说，Hash 函数的首要目标是保证数据完整性，对于 M 任何一位或几位的改变，都将极大可能改变其 Hash 码。

在安全应用中使用的 Hash 函数称为密码学 Hash 函数。密码学 Hash 函数要求如下两种情况在计算上不可行（即没有攻击方法比穷举攻击更有效）：(a) 对预先指定的 Hash 值找到对应的数据块（单向性）；(b) 找到两个不同的数据块对应相同的 Hash 值（抗碰撞性）。由于 Hash 函数具有上述特性，所以常被用于判断数据是否被篡改过。

图 11.1 描述了密码学 Hash 函数的操作过程。通常输入数据的长度首先被填充为某固定长度（如 1024 位）分组的整数倍，填充的内容包括原始消息的位长度信息。填充长度信息能够提高攻击者修改消息而保持 Hash 值不变的难度。

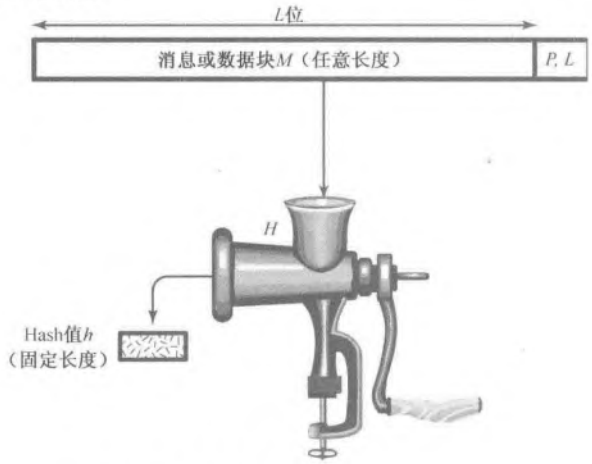


图 11.1　密码学 Hash 函数框图：$h=H(M)$

本章首先介绍密码学 Hash 函数的各种广泛应用；然后讨论 Hash 函数的安全性需求；接下来介绍如何通过分组密码链接构造密码学 Hash 函数；本章剩余的部分介绍最重要的也是应用最广泛的一类密码学 Hash 函数：安全 Hash 算法（SHA）系列算法。

附录 N 中介绍著名的密码学 Hash 函数 MD5 算法，其结构与 SHA-1 很相似。

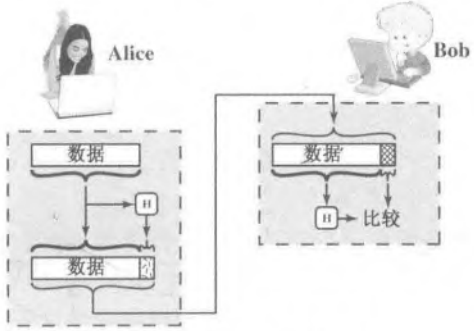
11.1　密码学 Hash 函数的应用

密码学 Hash 函数或许是用途最多的密码算法，它被广泛应用于各种不同的安全应用和网络协议中。为了更好地理解密码学 Hash 函数的安全性需求和含义，有必要首先考虑密码学 Hash 函数的应用范围。

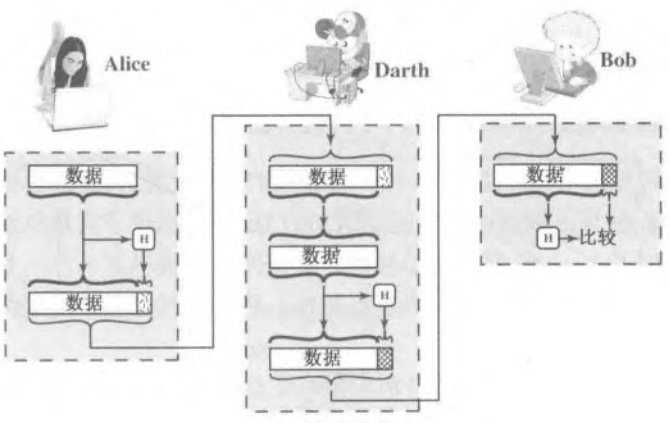
11.1.1　消息认证

消息认证是用来验证消息完整性的一种机制或服务。消息认证确保收到的数据确实和发送时的一样（即没有修改、插入、删除或重放）。此外，通常还要求消息认证机制确保发送方声称的身份是真实有效的。当 Hash 函数用于提供消息认证功能时，Hash 函数值通常称为消息摘要[①]。

消息认证中使用 Hash 函数的本质如下：发送者根据待发送的消息使用该函数计算一组 Hash 值，然后将 Hash 值和消息一起发送过去。接收者收到后对于消息执行同样的 Hash 计算，并将结果与收到的 Hash 值进行对比。如果不匹配，则接收者推断出消息（当然也可能是 Hash 值）遭受了篡改［参见图 11.2(a)］。



(a)使用Hash函数检查数据完整性



(b) 中间人攻击

图 11.2　对 Hash 函数的攻击

① 本节的主题总是围绕着消息认证讨论。但无论在概念上或技术上，最终都等同于对数据进行认证。例如，认证技术能用于存储中的文件，确保该文件没有被篡改。

Hash 函数的运算结果必须通过安全的方式传输。因为 Hash 函数必须得到保护，使得如果攻击者篡改或替换消息的话，对于攻击者不能够轻易地同时对 Hash 值进行修改以蒙骗接收者。这种类型的攻击如图 11.2(b)所示。该例中，Alice 传输消息时附上数据的 Hash 值。Darth 拦截了该消息，篡改或替换其中的数据，然后重新计算 Hash 值并附在后面。Bob 收到篡改后的数据块以及新的 Hash 值，未能发现消息已经被篡改。为了防止该类攻击，由 Alice 生成的 Hash 值必须得到保护。

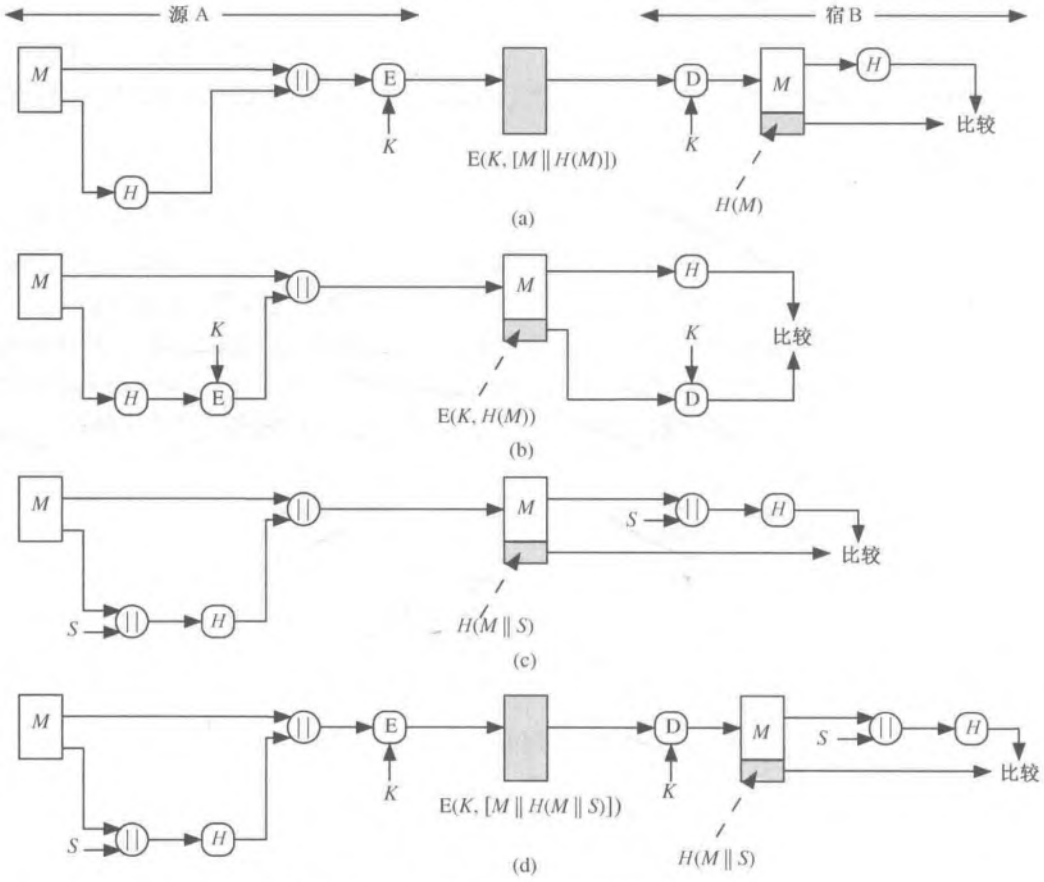


图 11.3 Hash 函数用于消息认证的简化示例

图 11.3 展示了 Hash 码能够通过如下各种不同的方法用于提供消息认证。

（a）使用对称密码算法 E 加密消息和 Hash 码。因为只有 A 和 B 共享密钥 K，所以消息必然是发自 A 处，并且未被更改过。这里附加的 Hash 码提供了实现认证功能的结构。因为对于整个消息以及 Hash 码都使用了加密，保密性也被提供。

（b）使用对称密码算法 E 只对 Hash 码进行加密。对于无须保密性的应用，这种方案减少了加解密操作的负担。

（c）不使用加密算法，仅使用 Hash 函数也能够实现消息认证。该方案假设通信双方共享相同的秘密值 S。发送方 A 将消息 M 和秘密值 S 串联后计算其 Hash 值，并将得到的 Hash 值附在消息 M 后发送。因为接收方 B 同时掌握 S，所以能够重新计算该 Hash 值进行验证。由于秘密值 S 本身并没有在信道传送，攻击者不能够对在信道上拦截的消息进行修改，进而也不能制作假消息。

（d）通过将整个消息和 Hash 值加密，能够在方案（c）的基础上提供保密性。

由于方案（b）中所需的计算较少，而方案（a）和方案（d）需要加密整个消息，所以如果不要求提供保密性，方案（b）比方案（a）和（d）更有优势。而且人们越来越对那些不含加密函数的方法感兴趣［参见图 11.3(c)］。文献[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，但是总有消息串需要通过加密系统输入或输出。
- 加密硬件成本不容忽视。尽管已有实现 DES 的低成本芯片，但是如果网络中所有节点都必须有该硬件，则总成本可能很大。
- 加密硬件的优化通常是针对大数据块的。对于小数据块，大比例的时间开销在初始化/调用上。
- 加密算法可能受专利保护，这也会增加成本。

更一般地，消息认证是通过使用消息认证码（MAC）实现的，即带密钥的 Hash 函数。通常情况下，通信双方基于共享的同一密钥来认证彼此之间交互的信息时，就会使用 MAC。MAC 函数将密钥和数据块作为输入，产生 Hash 值作为 MAC 码，然后将 MAC 码和受保护的消息一起传递或存储。需要检查消息的完整性时，使用 MAC 函数对消息重新计算，并将计算结果与存储的 MAC 码对比。攻击者能对消息进行篡改，但在不知道密钥的情况下不能够计算出与篡改后的消息相匹配的 MAC 值。注意，这里验证方也知道发送方是谁，因为除了通信双发之外其他人不知道密钥。

总体来看，MAC［参见图 11.3(b)］是 Hash 函数和加密函数操作的结合，即对于函数 $E(K, H(M))$，长度可变的消息 M 和密钥 K 是函数的输入，输出是固定长度的值。MAC 提供安全保护，用于抵抗不知道密钥的攻击者的攻击。在实现中，往往使用比加密算法效率更高的特殊设计的 MAC 算法。

我们将在第 12 章详细讨论消息认证码。

11.1.2　数字签名

同消息认证应用类似，对于 Hash 函数的另外一个重要应用就是数字签名。数字签名的操作与 MAC 相似，在进行数字签名过程中使用用户的私钥加密消息的 Hash 值，其他任何知道该用户公钥的人都能够通过数字签名来验证消息的完整性。在这种情况下，攻击者要想篡改消息，则需要知道用户的私钥。在第 14 章中我们将看到，数字签名的应用比消息认证更为广泛。

图 11.4 简要描述了 Hash 码用于提供数字签名的方案。

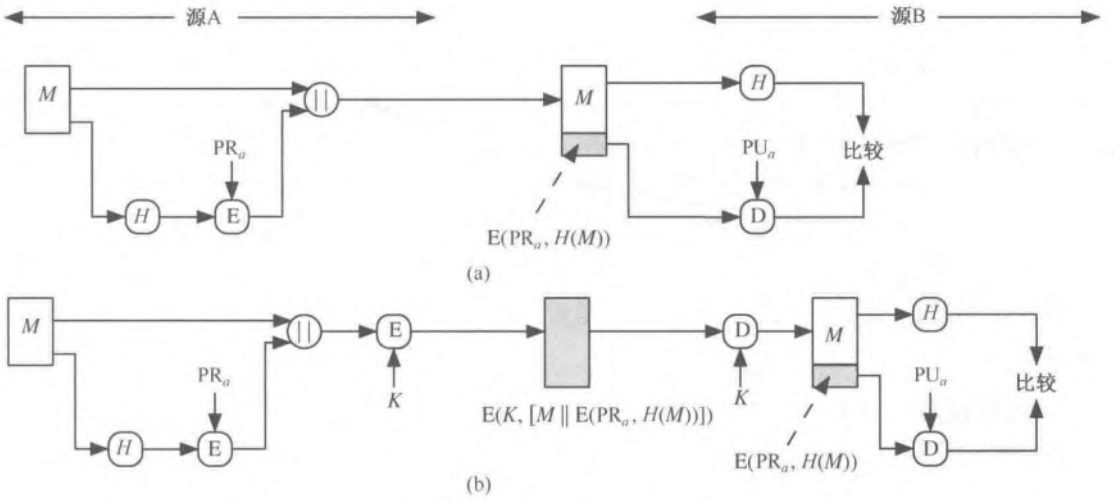


图 11.4　数字签名简要示例

（a）使用发送方的私钥，利用公钥密码算法仅对 Hash 码进行加密。与图 11.3(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。

（b）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公钥算法加密结果进行加密，这种技术比较常用。

11.1.3 其他应用

对于 Hash 函数，通常还被用于产生单向口令文件。第 21 章解释了操作系统为何存储口令的 Hash 值而不是口令本身的原因，即这样一来黑客即使能够访问口令文件，也不能够获取真正的口令。当用户输入口令时，操作系统将比对输入口令的 Hash 值和存储在口令文件中的 Hash 值。在大多数操作系统中都使用了这样的口令保护机制。

Hash 函数能用于入侵检测和病毒检测。将每个文件的 Hash 值 $H(F)$存储在安全系统中（如 CD-R 等保密系统中），随后就能够通过重新计算 $H(F)$来判断文件是否被修改过。入侵者只能够改变 F，而不能够改变 $H(F)$。

密码学 Hash 函数能够用于构建随机函数（PRF）或用做伪随机数发生器（PRNG）。基于 Hash 函数的 PRF 可用于对称密码中的密钥产生，我们将在第 12 章详细讨论。

11.2 两个简单的 Hash 函数

为了展示密码学 Hash 函数应考虑到的安全性，本节我们给出两个简单的、不够安全的 Hash 函数例子。所有的 Hash 函数都按下面的一般原理进行运算：输入（消息、文件等）都可视为一个 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，是将每个分组相应位异或（XOR），这个函数可如下描述为

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

其中，C_i= Hash 码的第 i 位，$1 \leq i \leq n$

$m = n$ 位输入分组的个数

b_{ij}= 第 j 个分组的第 i 位

$\oplus$= 异或运算

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，这种方法用于随机数的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，则对于这类数据，Hash 函数的有效性是 2^{-112} 而不是 2^{-128}。

一种简单的改进方法是，每处理完一个分组后，将 Hash 值平移一位或循环移位一次，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 值循环左移一位。

（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 11.5 给出了两种产生 16 位 Hash 值的 Hash 函数。

虽然第二种方法可以很好地保证数据完整性，但是如果使用图 11.3(b)和图 11.4(a)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）方法不能保证数据安全性，但是如果对消息和 Hash 码均加密，也许你会觉得那样一个简单函数还是有用的［参见图 11.3(a)］。但必须要小心，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，使用密文分组链接（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 $h = \mathrm{H}(M)$为所有分组的异或，并且将该 Hash 码作为最后一个分组

$$h = X_{N+1} = X_1 \oplus X_2 \oplus \cdots \oplus X_N$$

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（参见图 6.4），我们有

$$X_1 = IV \oplus \mathrm{D}(K, Y_1)$$
$$X_i = Y_{i-1} \oplus \mathrm{D}(K, Y_i)$$
$$X_{N+1} = Y_N \oplus \mathrm{D}(K, Y_{N+1})$$

且 Hash 码 X_{N+1} 为

$$\begin{aligned} X_{\mathrm{N+1}} &= X_1 \oplus X_2 \oplus \cdots \oplus X_N \\ &= [\mathrm{IV} \oplus \mathrm{D}(K, Y_1)] \oplus [Y_1 \oplus \mathrm{D}(K, Y_2)] \oplus \cdots \oplus [Y_{N-1} \oplus \mathrm{D}(K, Y_N)] \end{aligned}$$

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。

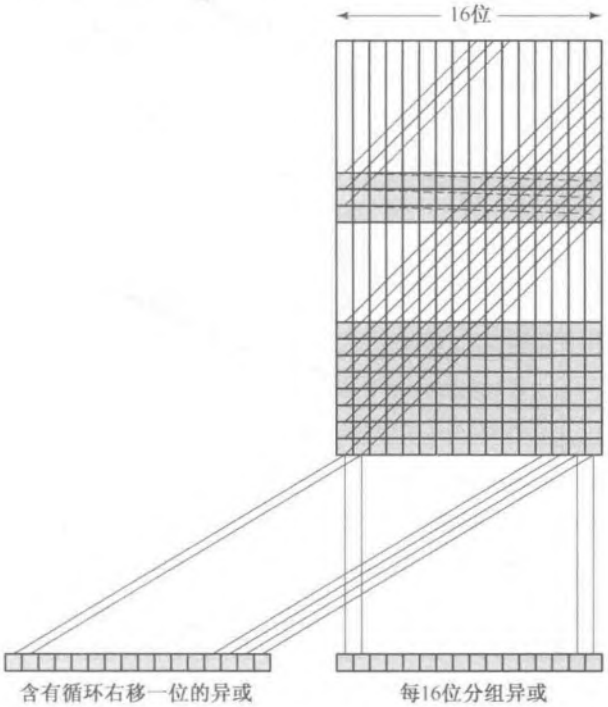


图 11.5　两个简单的 Hash 函数

11.3 需求和安全性

首先我们需要定义两个术语：对于 Hash 函数 $h=H(x)$，称 x 是 h 的原像。即把数据块 x 作为输入，使用 Hash 函数 H 得到 h。因为 H 是多对一映射，所以对于任意给定的 Hash 值 h，对应有多个原像。如果满足 $x\neq y$ 且 $H(x)=H(y)$，则称出现碰撞。因为我们使用 Hash 函数的目的是保证数据完整性，碰撞的出现显然是我们不希望的。

首先考虑对于给定的 Hash 值，有多少个对应的原像，这也是对于给定的 Hash 值的潜在碰撞个数的衡量。假设 Hash 码的长度是 n 位，函数 H 的输入消息或数据块长度是 b 位，且 $b>n$。那么消息的全部可能取值的个数为 2^b，Hash 值的全部可能取值的个数为 2^n，平均每个 Hash 值对应 2^{b-n} 个原像。如果 H 的 Hash 值趋于均匀分布，事实上每个 Hash 值将对应近似 2^{b-n} 个原像。如果我们允许输入是任意长度而不是固定长度，那么每个 Hash 值对应的原像个数将更大。然而，实际使用 Hash 函数的安全性风险并非如上面的分析所表现的一样。为了更好地理解密码学 Hash 函数的安全含义，需要精确地定义 Hash 函数的安全性需求。

11.3.1 密码学 Hash 函数的安全性需求

表 11.1 列出了被广泛认同的密码学 Hash 函数的安全性需求。前三个特性是 Hash 函数实际应用的需求。

第 4 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［参见图 11.3(c)］，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，则攻击者可以按如下方式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 $h=H(S\|M)$，然后求出 Hash 函数的逆，从而得出 $S\|M=H^{-1}(\mathrm{MD}_M)$。由于攻击者已知 M 和 $S_{AB}\|M$，所以可得出 S_{AB}。

第 5 个性质抗弱碰撞性可以保证，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［参见图 11.3(b)和图 11.4(a)］。如果第 5 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 码的替代消息。

表 11.1 密码学 Hash 函数 *H* 的安全性需求

需　求	描　述
输入长度可变	H 可应用于任意大小的数据块
输出长度固定	H 产生定长的输出
效率	对任意给定的 x，计算 $H(x)$ 比较容易，用硬件和软件均可实现
抗原像攻击（单向性）	对任意给定的 Hash 码 h，找到满足 $H(y)=h$ 的 y 在计算上是不可行的
抗第二原像攻击（抗弱碰撞性）	对任何给定的分块 x，找到满足 $y\neq x$ 且 $H(x)=H(y)$ 的 y 在计算上是不可行的
抗碰撞攻击（抗强碰撞性）	找到任何满足 $H(x)=H(y)$ 的偶对 (x,y) 在计算上是不可行的
伪随机性	H 的输出满足伪随机性测试标准

如果一个 Hash 函数满足表 11.1 中的前 5 个要求，就称其为弱 Hash 函数。如果第 6 个性质抗强碰撞性也满足，就称其为强 Hash 函数。强 Hash 函数能够保证免受如下攻击：通信双方中的一方生成消息，而另一方对消息进行签名。例如，假设 Bob 写一条 IOU（借据）消息并发送给 Alice，Alice 在借据上签名认可。Bob 如果能够找到两个消息具有同样的 Hash 值，其中一个借据消息要求 Alice 归还金额较小，另一个金额很大，那么让 Alice 签下第一个小额借据后，Bob 就能够声称第二个（大额）借据是真实的。

图 11.6 展示了三个安全特性（抗原像攻击、抗弱碰撞攻击、抗强碰撞攻击）之间的联系。一个函数如果是抗强碰撞的，那么也同时是抗弱碰撞的，但反之则不一定成立。一个函数可以是抗强碰撞的，但不一定是抗原像攻击的，反之亦然。一个函数可以是抗弱碰撞的，但不一定是抗原像攻击的，反之亦然。详细讨论请参见文献[MENE97]。

表 11.2 展示了在不同应用环境下 Hash 函数的安全性需求。

表 11.1 中的最后一个性质是伪随机性，在传统观念中并没有作为密码学 Hash 函数的安全性需求，但却在实际使用中或多或少有所要求。参考文献[JOHN05]指出密码学 Hash 函数通常用于密钥产生、伪随机数发生器以及消息完整性应用，上述三个安全特性都要求 Hash 函数的输出是随机的。因此需要 Hash 函数的输出具有伪随机性。

表 11.2　各种数据完整性应用中的 Hash 函数安全性需求

	抗原像攻击	抗弱碰撞攻击	抗强碰撞攻击
Hash+数字签名	是	是	是*
入侵检测和病毒检测		是	
Hash+对称加密			
单向口令文件	是		
MAC	是	是	是*

注：标*处要求攻击者能够实现选择消息攻击。

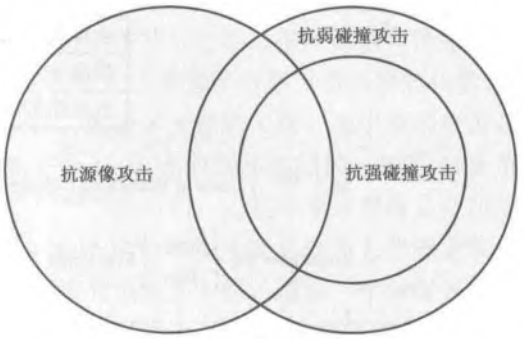


图 11.6　Hash 函数安全特性之间的联系

11.3.2　穷举攻击

同加密算法一样，对于 Hash 函数的攻击也分为两类：穷举攻击和密码分析。穷举攻击不依赖于任何算法的细节，仅与相应的长度相关。对于 Hash 函数，穷举攻击仅与算法所产生的 Hash 值的长度相关。与之相反，密码分析依赖于具体算法的设计缺点。我们首先介绍穷举攻击。

原像攻击和第二原像攻击　对于原像攻击和第二原像攻击，攻击者对给定的 Hash 值 h，试图找到满足 $H(y)=h$ 的 y。穷举攻击的方法是随机选择 y，尝试计算其 Hash 值直到碰撞出现。对于 m 位的 Hash 值，穷举规模大约是 2^m。对于攻击者平均尝试次数为 2^{m-1}，才能找到一个满足 $H(y)=h$ 的 y。该结论请参考附录 U 中的式（U.1）。

碰撞攻击　对于碰撞攻击，攻击者试图找到两个消息或数据块 x 和 y，满足 $H(x)=H(y)$。可想而知，与原像攻击和第二原像攻击相比，其穷举规模相对更小一些，这也通过数学上的生日悖论得到印证。本质上，如果我们在均匀分布的 0 到 $N-1$ 的范围内选择随机整数变量，那么在 $\sqrt{N}$ 次选择后发生重复的概率就会超过 0.5。因此，对于 m 位的 Hash 值，如果我们随机选择数据块，预计在 $\sqrt{2^m}=2^{m/2}$ 次尝试后就将找到两个具有相同 Hash 值的数据块。具体证明请参阅附录 U。

Yuval 提出了以下策略通过生日悖论进行碰撞攻击[YUVA79]：

（1）发送方 A 准备对文本消息 x 进行签名，其使用的方法是：用 A 的私钥对 m 位的 Hash 码加密并将加密后的 Hash 码附于消息之后［参见图 11.4(a)］。

（2）攻击者产生该（合法）消息 x 的 $2^{m/2}$ 种变式 x'，且每一种变式表达相同的意义，将这些消息以及对应的 Hash 值存储起来。

（3）攻击者准备伪造一条消息 y，并想获取 A 的签名。

（4）攻击者再产生该伪造（非法）消息 y 的消息变式 y'，每个变式 y'与 y 表达相同的意义。对于每个 y'，攻击者计算 $H(y')$，并与任意的 $H(x')$进行比对，重复这一过程直到碰撞出现。即这一过程直到找到一个 y'与某个 x'具有相同的 Hash 值。

（5）攻击者将该合法消息变式 x' 提供给 A 签名，将该签名附于伪造消息的有效变式 y' 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} [参见附录 U 中的式（U.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的某些词但不改变消息的意义，图 11.7 举例说明了这种方法。

总之，对长度为 m 的 Hash 码，对于穷举攻击所需的代价分别与下表中相应量成正比。

抗原像攻击	2^m
抗弱碰撞攻击	2^m
抗强碰撞攻击	$2^{m/2}$

As { the / — } Dean of Blakewell College, I have { had the pleasure of knowing / known } Cherise Rosetti for the { last / past } four years. She { has been / was } { a tremendous / an outstanding } { asset to / role model in } { our / the } school. I { would like to take this opportunity to / wholeheartedly } recommend Cherise for your { school's / — } graduate program. I { am / feel } { confident / certain } { that / — } { she / Cherise } will { continue to / — } succeed in her studies. { She / Cherise } is a dedicated student and { thus far her grades / her grades thus far } { have been / are } { exemplary / excellent }. In class, { she / Cherise } { has proven to be / has been } a take-charge { person / individual } { who is / — } able to successfully develop plans and implement them.

{ She / Cherise } has also assisted { us / — } in our admissions office. { She / Cherise } has { successfully / — } demonstrated leadership ability by counseling new and prospective students. { Her / Cherise's } advice has been { a great / of considerable } help to these students, many of whom have { taken time to share / shared } their comments with me regarding her pleasant and { encouraging / reassuring } attitude. { For these reasons / It is for these reasons that } I { highly recommend / offer high recommendations for } Cherise { without reservation / unreservedly }. Her { ambition / drive } and { abilities / potential } will { truly / surely } be an { asset to / plus for } your { establishment / school }.

图 11.7 具有 2^{38} 种变式的信

如果要求抗强碰撞能力（通用的安全 Hash 码通常期望具有该性质），那么值 $2^{m/2}$ 决定了该 Hash 码抗穷举攻击的强度。Van Oosrschot 和 Wiener[VANO94]耗资 1000 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 码视为一串 32 位的序列，那么以后将要使用 160 位的 Hash 码。对于 160 位 Hash 码，用相同的搜寻机器则需要 4000 年才能找到一个碰撞。然而，随着软硬件技术的进步，搜索的时间会大大缩短，所以对于穷举攻击现在看来 160 位也被认为比较弱。

11.3.3　密码分析

与对密码算法的攻击一样，对 Hash 函数的密码分析攻击，也是利用算法的某种性质而不是通过穷举来进行攻击。评价 Hash 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。

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 函数的总体结构，如图 11.8 所示。这种结构称为迭代 Hash 函数，它是由 Merkle[MERK79, MERK89]提出的。包括本章后面将会介绍的 SHA 在内的目前所使用的大多数 Hash 函数都是这种结构。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，最后一个分组包含输入的总长度。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。

Hash 函数中重复使用了压缩函数 f，它的输入包括两部分：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 $b>n$，因此称为压缩。Hash 函数可归纳如下：

$$CV_0 = IV = \text{初始 } n \text{ 位值}$$
$$CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \leqslant i \leqslant L$$
$$H(M) = CV_L$$

其中 Hash 函数的输入为消息 M，它由分组 $Y_0, Y_1, Y_2, \cdots, Y_{L-1}$ 组成。

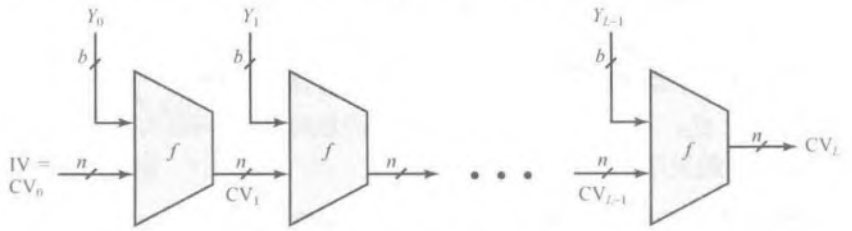


图 11.8　安全 Hash 码的一般结构

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构造安全 Hash 函数。由此可见，设计安全 Hash 函数可简化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。

① 其逆不一定为真。

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，并试图找到能由 f 的单步执行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。对 f 的攻击要依靠对其内部结构的研究。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位变化的模式。

值得注意的是：由于添加了一个长度域，我们将长度至少为两倍于分组大小 b 的消息映射为长度为 n 的 Hash 码，其中 $b \geq n$，所以任何 Hash 函数都存在有碰撞。因此能要求的仅是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文献[DOBB96]和文献[BELL97]。

11.4 基于分组密码链接的 Hash 函数

人们已经提出的 Hash 函数中，有许多基于分组密码链接方法，但没有使用密钥。例如 Rabin 所提出方法[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

$$H_0 = \text{初始值}$$
$$H_i = \mathrm{E}(M_i, H_{i-1})$$
$$G = H_N$$

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（想要伪造的）消息。

（3）对 $1 \leq i \leq (N-2)$，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 $\mathrm{E}(X, H_{N-2})$。再产生另外 $2^{m/2}$ 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 $\mathrm{E}(X, H_{N-2}) = \mathrm{D}(Y, G)$ 的概率较大。

（6）构造消息 Q_1，Q_2，…，Q_{N-2}，X，Y。因为该消息的 Hash 码也为 G，所以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：

$$H_i = \mathrm{E}(M_i, H_{i-1}) \oplus H_{i-1}$$

参考文献[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]。

11.5　安全 Hash 算法

近年来，安全 Hash 算法（SHA）是使用最广泛的 Hash 函数。事实上，由于其余的被广泛应用的 Hash 函数被发现存在安全性缺陷，从 2005 年以来 SHA 或许是这几年中仅存的 Hash 算法标准。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。随后该版本的 SHA（即 SHA-0）被发现存在缺陷，修订版于 1995 年发布（FIPS 180-1），通常称之为 SHA-1。实际的标准文件称其为“安全 Hash 标准”。SHA 算法建立在 MD4 算法之上，其基本框架与 MD4 类似。

SHA-1 产生 160 位的 Hash 值。2002 年，NIST 发布了修订版 FIPS 180-2，其中给出了三种新的 SHA 版本，Hash 值长度依次为 256 位、384 位和 512 位，分别称为 SHA-256、SHA-384 和 SHA-512。这些算法被统称为 SHA-2。SHA-2 同 SHA-1 类似，都使用了同样的迭代结构和同样的模算术运算与二元逻辑操作。在 2008 年发布的修订版 FIP PUB 180-3 中，增加了 224 位版本（参见表 11.3）。2015 年，NIST 颁布了 FIPS 180-4，增加了两个算法 SHA-512/224 和 SHA-512/256。SHA-1 和 SHA-2 在 RFC 6234 中也有描述，基本上也是复制 FIPS 180-3 中的内容，但增加了 C 代码实现。

2005 年，NIST 宣布了逐步废除 SHA-1 的意图，计划到 2010 年逐步转而依赖 SHA-2 的其他版本。此后不久，一个研究小组[WANG05]给出了一种攻击，用 2^{69} 次操作可以找到两个独立的消息使它们有相同的 SHA-1 值，而以前认为要找到一个 SHA-1 碰撞需要 2^{80} 次操作，所需操作大为减少。这一结果应该加速了向 SHA-2 版本的过渡。

本节，我们将介绍 SHA-512，其他版本与之类似。

表 11.3　SHA 参数比较

	消息长度	分组长度	字长度	消息摘要长度
SHA-1	$<2^{64}$	512	32	160
SHA-224	$<2^{64}$	512	32	224
SHA-256	$<2^{64}$	512	32	256
SHA-384	$<2^{128}$	1024	64	384
SHA-512	$<2^{128}$	1024	64	512
SHA-512/224	$<2^{128}$	1024	64	224
SHA-512/256	$<2^{128}$	1024	64	256

注：所有的长度以二进制位为单位。

11.5.1　SHA-512 逻辑

算法的输入是最大长度小于 2^{128} 位的消息，输出是 512 位的消息摘要，输入消息以 1024 位的分组为单位进行处理。图 11.9 显示了处理消息、输出摘要的总体过程，其迭代结构遵循图 11.8 所示的一般结构。这个过程包含下列步骤：

- **步骤 1：附加填充位**　填充消息使其长度模 1024 与 896 同余［即长度 ≡896 (mod 1024)］，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 1～1024 之间。填充由一个 1 和后续的 0 组成。
- **步骤 2：附加长度**　在消息后附加一个 128 位的块，将其视为 128 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。

  前两步的结果是产生了一个长度为 1024 整数倍的消息。图 11.9 中，扩展的消息被表示为一串长度为 1024 位的消息分组 $M_1, M_2, \cdots, M_N$，因此扩展消息的长度为 N×1024 位。

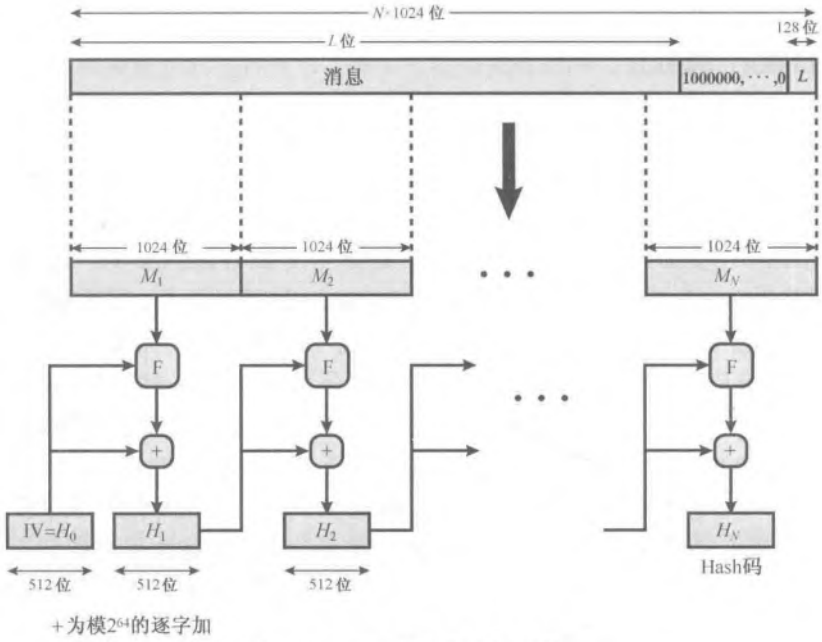


图 11.9 SHA-512 生成消息摘要

- **步骤 3：初始化 Hash 缓冲区** Hash 函数的中间结果和最终结果保存于 512 位的缓冲区中，缓冲区用 8 个 64 位的寄存器(a,b,c,d,e,f,g,h)表示，并将这些寄存器初始化为下列 64 位的整数（十六进制值）：

```
a = 6A09E667F3BCC908      e=510E527FADE682D1
b = BB67AE8584CAA73B      f=9B05688C2B3E6C1F
c = 3C6EF372FE94F82B      g=1F83D9ABFB41BD6B
d = A54FF53A5F1D36F1      h=5BE0CD19137E2179
```

这些值以高位在前格式存储，也就是说，字的最高有效字节存于低地址字节位置(最左边)。这些字的获取方式如下：前 8 个素数取平方根，取小数部分的前 64 位。

- **步骤 4：以 1024 位的分组（128 字节）为单位处理消息** 算法的核心是具有 80 轮运算的模块，图 11.9 中，该模块标记为 F。图 11.10 给出了它的逻辑原理。

每一轮都把 512 位缓冲区的值 *abcdefgh* 作为输入，并更新缓冲区的值。第一轮时，缓冲区里的值是中间值 H_{i-1}。每一轮，如第 t 轮，使用一个 64 位的值 W_t，该值由当前被处理的 1024 位消息分组 M_i 导出，导出算法是下面将要讨论的消息扩展算法。每一轮还将使用附加的常数 K_t，其中 $0 \leqslant t \leqslant 79$，用来使每轮的运算不同。这些常数的获得方法如下：对前 80 个素数开立方根，取小数部分的前 64 位。这些常数提供了 64 位随机串集合，可以初步消除输入数据里的统计规律。表 11.4 给出了轮常数的十六进制值（从左到右）。

第 80 轮的输出和第一轮的输入 H_{i-1} 相加产生 H_i。缓冲区中的 8 个字和 H_{i-1} 中对应的字分别进行模 2^{64} 的加法运算。

- **步骤 5：输出** 所有的 N 个 1024 位分组都处理完以后，从第 N 阶段输出的是 512 位的消息摘要。

总结 SHA-512 的运算如下：

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, abcdefgh_i)$$
$$\text{MD} = H_N$$

其中，IV=第三步里中定义的 *abcdefgh* 缓冲区的初始值；$abcdefgh_i$=第 i 个消息分组处理的最后一轮的输出；N=消息（包括填充和长度域）中的分组数；SUM_{64}=对输入对中的每个字进行独立的模 2^{64} 加；MD=最后的消息摘要值。

表 11.4　SHA-512 轮常数

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

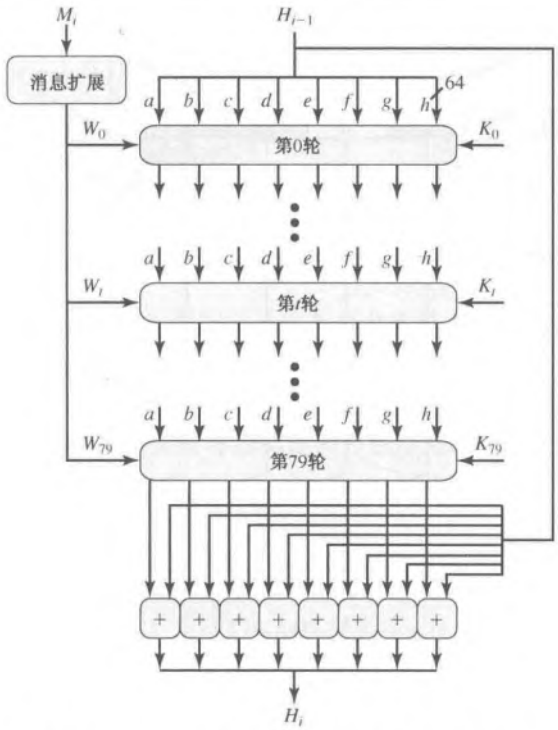


图 11.10　SHA-512 对单个 1024 位分组的处理

11.5.2 SHA-512 轮函数

下面我们仔细讨论 80 轮的每一轮的内部逻辑，每一轮处理 512 位的单分组（参见图 11.11）。每一轮由如下的方程定义：

$$T_1 = h + \mathrm{Ch}(e,f,g) + (\sum_1^{512} e) + W_t + K_t$$
$$T_2 = (\sum_0^{512} a) + \mathrm{Maj}(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

其中，t 为步骤数，$0 \leqslant t \leqslant 79$。

$\mathrm{Ch}(e,f,g) = (e \text{ AND } f) \oplus (\text{NOT } e \text{ AND } g)$条件函数：如果 e，则 f，否则 g。

$\mathrm{Maj}(a,b,c) = (a \text{ AND } b) \oplus (a \text{ AND } c) \oplus (b \text{ AND } c)$ 函数为真当且仅当变量的多数（2 个或 3 个）为真。

$(\sum_0^{512} a) = \mathrm{ROTR}^{28}(a) \oplus \mathrm{ROTR}^{34}(a) \oplus \mathrm{ROTR}^{39}(a)$

$(\sum_1^{512} e) = \mathrm{ROTR}^{14}(e) \oplus \mathrm{ROTR}^{18}(e) \oplus \mathrm{ROTR}^{41}(e)$

$\mathrm{ROTR}^n(x)$为对 64 位的变量 x 循环右移 n 位

W_t 为 64 位字，从当前的 1024 位输入分组导出

K_t 为 64 位附加常数

+为模 2^{64} 加

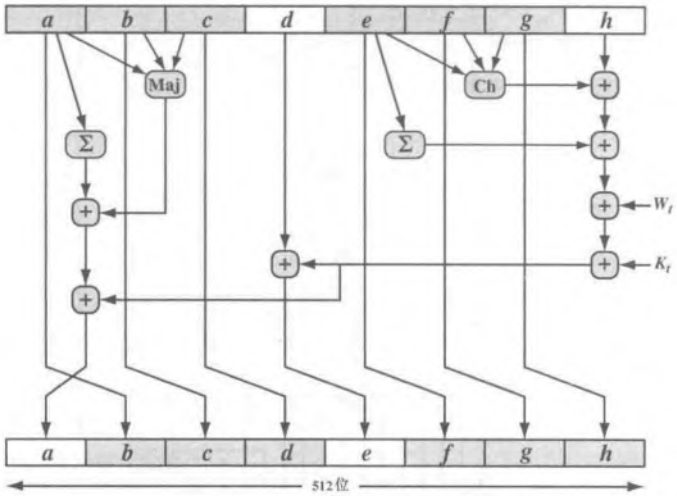


图 11.11 基本的 SHA-512 运算（单轮）

轮函数有如下两个特点：

（1）轮函数输出的 8 个字中的 6 个（b, c, d, f, g, h）是通过简单的轮转置换实现的。如图 11.11 中的阴影部分。

（2）输出中只有 2 个字（a, e）通过替代操作产生。字 e 是将输入变量（d, e, f, g, h）以及轮消息 W_t 和轮常数 K_t 作为输入的函数。字 a 是将除 d 之外的输入变量以及轮消息 W_t 和轮常数 K_t 作为输入的函数。

还需要指出的是 64 位的字 W_t 是如何从 1024 位消息里导出的。图 11.12 说明了该映射的结构。前 16 个 W_t 直接取自当前分组的 16 个字。余下的值按如下方式导出：

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

其中

$$\sigma_0^{512}(x) = \mathrm{ROTR}^1(x) + \mathrm{ROTR}^8(x) + \mathrm{SHR}^7(x)$$

$$\sigma_1^{512}(x) = \mathrm{ROTR}^{19}(x) + \mathrm{ROTR}^{61}(x) + \mathrm{SHR}^6(x)$$

$\mathrm{ROTR}^n(x)$为对 64 位的变量 x 循环右移 n 位。

$\mathrm{SHR}^n(x)$ 为对 64 位变量 x 向右移动 n 位，左边填充 0。

+为模 2^{64} 加。

因此，前 16 步处理过程中，W_t 的值等于消息分组中的相应字。对于余下的 64 步，W_t 的值由其前面的 4 个值推导得出，4 个值中的两个要进行算术移位和循环移位操作。这增加了被压缩消息分组的冗余性和相互依赖性，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。

图 11.13 总结了 SHA-512 的整个结构。

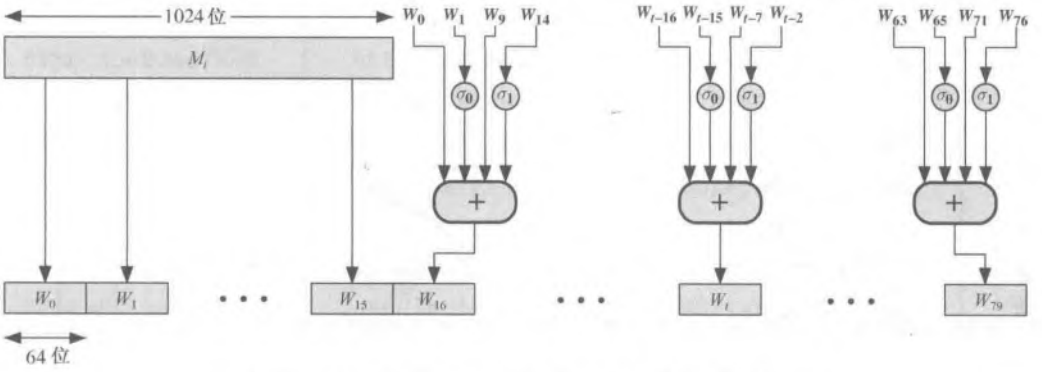


图 11.12　SHA-512 处理单个分组产生的 80 字的输入序列

SHA-512 算法具有如下特性：Hash 码的每一位都是全部输入位的函数。基本函数 F 多次复杂重复运算使得结果充分混淆，从而使得随机选择两个消息，甚至于这两个消息有相似特征，都不太可能产生相同的 Hash 码。除非 SHA-512 中存在目前未公开的隐藏缺陷，找到两个具有相同摘要的消息的复杂度需要 2^{256} 次操作，而给定摘要寻找消息的复杂度需要 2^{512} 次操作。

11.5.3　示例

我们这里给出 FIPS 180 中提供的一个示例。假设对由三个 ASCII 字符“abc”构成的消息块进行 Hash 处理，该消息块可写成如下 24 位的串：

```
01100001  01100010  01100011
```

从 SHA 算法的第一步开始，首先消息被填充后使其长度模 1024 与 896 同余。对于上面的消息块，填充长度为 896 – 24 = 872 位，填充内容包括 1 个“1”和后面紧跟的 871 个“0”。然后 128 位的长度信息附在填充消息的后面，保存的是消息的（填充前的）原始长度。本例中原始长度是 24 位，

即十六进制值 18。将上述内容连接在一起构成的完整 1024 位消息分块的十六进制表示为

```
6162638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018
```

在消息扩展过程中，该数据块的各字被分别赋值给 $W_0, \cdots, W_{15}$，如下所示：

$$
\begin{aligned}
W_0 &= 6162638000000000 & W_8 &= 0000000000000000\\
W_1 &= 0000000000000000 & W_9 &= 0000000000000000\\
W_2 &= 0000000000000000 & W_{10} &= 0000000000000000\\
W_3 &= 0000000000000000 & W_{11} &= 0000000000000000\\
W_4 &= 0000000000000000 & W_{12} &= 0000000000000000\\
W_5 &= 0000000000000000 & W_{13} &= 0000000000000000\\
W_6 &= 0000000000000000 & W_{14} &= 0000000000000000\\
W_7 &= 0000000000000000 & W_{15} &= 0000000000000018
\end{aligned}
$$

如图 11.13 所示，从 a 到 h 的 8 个 64 位变量，首先被赋值为从 $H_{0,0}$ 到 $H_{0,7}$。下表给出了这些变量的初始值和前两轮的值。

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

注意每一轮中有 6 个变量的取值是直接复制上一轮的变量。

该过程重复 80 轮，最后一轮的输出是

```
73a54f399fa4b1b2  10d9c4c4295599f6  d67806db8b148677  654ef9abec389ca9
d08446aa79693ed7  9bb4d39778c07f9e  25c96a7768fb2aa3  ceb9fc3691ce8326
```

其中的 Hash 值通过如下计算产生：

$H_{1,0}$ = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba

$H_{1,1}$ = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131

$H_{1,2}$ = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2

$H_{1,3}$ = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a

$H_{1,4}$ = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8

$H_{1,5}$ = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd

$H_{1,6}$ = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e

$H_{1,7}$ = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f

扩展后的消息包括分组 $M_1, M_2, \cdots, M_N$，每个消息分组 M_i 由 16 个 64 位字 $M_{i,0}, M_{i,1}, \cdots, M_{i,15}$ 构成。所有的加法运算都是指模 2^{64} 加法。

$H_{0,0}$ = 6A09E667F3BCC908　　$H_{0,4}$ = 510E527FADE682D1

$H_{0,1}$ = BB67AE8584CAA73B　　$H_{0,5}$ = 9B05688C2B3E6C1F

$H_{0,2}$ = 3C6EF372FE94F82B　　$H_{0,6}$ = 1F83D9ABFB41BD6B

$H_{0,3}$ = A54FF53A5F1D36F1　　$H_{0,7}$ = 5BE0CD19137E2179

for i=1 **to** N

1．预处理消息扩展

for t=0 **to** 15

$W_t = M_{i,t}$

for t=16 **to** 79

$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$

2．初始化工作变量

$a = H_{i-1,0}$　　$e = H_{i-1,4}$

$b = H_{i-1,1}$　　$f = H_{i-1,5}$

$c = H_{i-1,2}$　　$g = H_{i-1,6}$

$d = H_{i-1,3}$　　$h = H_{i-1,7}$

3．执行主 Hash 计算

for t=0 **to** 79

$T_1 = h + \mathrm{Ch}(e,f,g) + (\Sigma_1^{512} e) + W_t + K_t$

$T_2 = (\Sigma_0^{512} a) + \mathrm{Maj}(a,b,c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

4．计算 Hash 中间值

$H_{i,0} = a + H_{i-1,0}$　　$H_{i,4} = e + H_{i-1,4}$

$H_{i,1} = b + H_{i-1,1}$　　$H_{i,5} = f + H_{i-1,5}$

$H_{i,2} = c + H_{i-1,2}$　　$H_{i,6} = g + H_{i-1,6}$

$H_{i,3} = d + H_{i-1,3}$　　$H_{i,7} = h + H_{i-1,7}$

返回 $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$

图 11.13　SHA-512 逻辑

最终得到的 512 位消息摘要是

```
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f
```

假设我们将输入消息改变 1 位，从“abc”变成“cbc”，那么 1024 位消息块为

```
6362638000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000018
```

对应的 512 位消息摘要是

```
531668966ee79b70 0b8e593261101354 4273f7ef7b31f279 2a7ef68d53f93264
319c165ad96d9187 55e6a204c2607e27 6e05cdf993a64c85 ef9e1e125c0f925f
```

前后两个消息的Hash值有253位不同，几乎一半发生了改变，这说明SHA-512有良好的雪崩效应。

11.6 SHA-3

在本书编写过程中，SHA-1暂时还未被“攻破”，即还没有出现在实际运算能力之内的寻找碰撞的方法。但是SHA-1在设计上与MD5和SHA-0具有相似的结构和基本数学运算，而后两者已经被攻破。因此SHA-1被认为是不安全的，已经被SHA-2取代。

对于SHA-2，特别是512版本的安全强度是毋庸置疑的。但是SHA-2同样和它的前任一样在设计上有类似的结构和基本数学运算，这值得我们关注。或许在若干年后SHA-2的缺陷也会被发现而被取代，因此NIST决定开展新的Hash标准的制定工作。

随后NIST在2007年宣布公开征集新一代NIST的Hash函数标准，称其为SHA-3。NIST在2012年10月公布了SHA-3设计作品中的优胜者，并在2015年8月颁布了FIPS 202。SHA-3作为新一代密码学Hash函数的标准，将在各类应用中逐渐取代SHA-2。

附录V给出了NIST在众多AES候选算法中使用的评估准则，以及选择Keccak算法作为SHA-3候选算法中的优胜者的原因。这些材料不仅可以帮助大家理解SHA-3是如何设计的，还可以提供判断任何一个密码学Hash函数好坏的标准。

11.6.1 海绵结构

SHA-3的设计者将SHA-3使用的基本迭代结构称为海绵结构[BERT07, BERT11]方案。海绵结构与其他迭代Hash函数的结构相似（参见图11.8）。在海绵函数中，输入消息被分块为固定长度的分组。每个分组逐次作为每轮迭代的输入，同时上轮迭代的输出也反馈至下轮的迭代中，最终产生一组输出块。

海绵函数由三组参数定义：

f= 内部函数，用于处理每轮的输入分组①

r= 输入分组的位长度，称其为位速率（bitrate）

pad = 填充算法

海绵函数允许输入长度和输出长度都可变。由于具有这个灵活的结构特点，海绵函数能够用于设计Hash函数（固定输出长度）、伪随机数发生器（固定输入长度），以及其他密码函数。图11.14展示了这个特点。长度为n位的输入消息，被分为k个固定长度的分组，每个分组长度为r位。如果需要，消息将被填充以满足填充后的长度是r位的整数倍。分组后的结果是$P_0, P_1, \cdots, P_{k-1}$，其中$n = k \times r$。为了格式统一，任意消息都需要进行填充。因此如果$n \bmod r = 0$，那么将填充一个$r$位的完整块。填充算法实质上是作为海绵函数的一个参数。在海绵结构的实现建议[BERT11]中，给出了两个填充方案：

- **简单填充** 用pad10*表示，用一个1后面跟若干个0进行填充，0的个数是使得总长度为分组长度整倍数的最小值。
- **多重位速率填充** 用pad10*1表示，用一个1后面跟若干个0，再跟一个1进行填充，0的个数是使得总长度为分组长度整倍数的最小值。这是在安全地采用不同位速率r使用相同的f时，最简单的填充方法。

① Keccak文档中定义f为一个置换。接下来我们会看到，其包含置换和代替。因为f函数每轮迭代中执行一次，即处理每个消息分组时调用一次，所以将f称为迭代函数。

在对所有的分组操作完后，海绵函数产生一组输出块 $Z_0, Z_1, \cdots, Z_{j-1}$。产生的输出数据块的个数由需要的输出位数所决定。如果需要 ℓ 位输出，则产生 j 个输出块，其中 $(j-1)\times r < \ell < j\times r$。

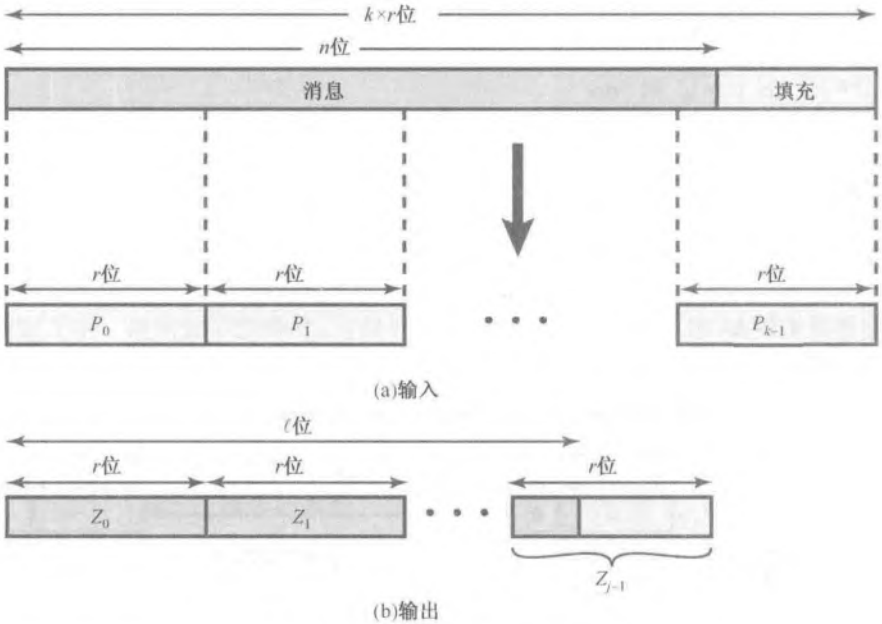


图 11.14　海绵函数的输入和输出

图 11.15 给出了海绵函数的迭代结构。海绵结构对于长度为 $b=r+c$ 位的状态变量 s 进行操作，状态变量的初值全部置为 0，其取值在每轮迭代中更新。数值 r 称为位速率，该值是输入消息的分组长度。位速率反映了每轮迭代中处理的位数：r 越大，海绵结构处理消息的速度就越快。数值 c 称为容量。关于容量的安全性含义已经超出我们讨论的范围。本质上容量是度量海绵结构能够达到的复杂程度以及安全度。在实际应用中可以通过降低速度来提高安全性，增大容量 c 的取值，减小位速率 r 的取值，反之亦然。Keccak 的默认选择是 $c=1024$ 位，$r=576$ 位，于是 $b=1600$ 位。

海绵结构包括两个阶段。吸水阶段过程如下：对于每轮迭代，通过填充若干个 0，将输入数据块的长度从 r 位扩展为 b 位；然后将扩展后的消息分组和 s 进行 XOR 异或运算得到 b 位的结果，并作为迭代函数 f 的输入；f 函数的输出作为下一轮迭代中 s 的取值。

如果需要的输出长度满足 $\ell \leqslant b$，那么在吸水阶段完成后，则返回 s 的前 ℓ 位，海绵结构的运行结束。否则，海绵结构进入挤压阶段。首先，s 的前 ℓ 位被保留作为输出分组 Z_0，然后在每轮迭代中通过重复执行 f 函数来更新 s 的值，s 的前 ℓ 位被依次保留作为输出分组 Z_i，并与前面已生成的分组连接起来。该处理过程共需要 $(j-1)$ 次迭代，直到满足 $(j-1)\times r < \ell \leqslant j\times r$。这时候将各输出分组连接 Z 的前 ℓ 位作为输出返回。

注意吸水阶段具有典型的 Hash 函数结构。一种常见的情况是需要的 Hash 长度小于或等于输入的分组长度，即 $\ell \leqslant r$。这种情况下海绵结构会在吸水阶段完成后直接结束。如果需要的输出长度超过 b 位，则需要执行挤压阶段的操作。因此海绵结构非常灵活。例如较短的长度为 r 的消息可以作为输入种子，海绵结构可以提供伪随机数发生器的功能。

总之，海绵结构是一种简单的迭代结构，基于固定长度的转换或对固定长度 b 位进行操作的置换 f，构造出来的整体函数 F 能够处理可变长度的输入，得到任意长度的输出。海绵结构在文献[BERT11]中的形式化定义如下：

算法： 海绵结构 SPONGE[f, pad, r]
要求： $r < b$
接口： Z=sponge (M, ℓ)，这里 $M \in Z_2^*$，整数 $l > 0$，$Y \in Z_2^l$

```
P = M || pad[r](|M|)
s = 0^b
for i = 0 to |P|_r - 1 do
    s = s ⊕ (P_i || 0^(b-r))
    s = f(s)
end for
Z = ⌊s⌋_r
while |Z|_r r < l do
    s = f(s)
    Z = Z || ⌊s⌋_r
end while
返回 ⌊Z⌋_ℓ
```

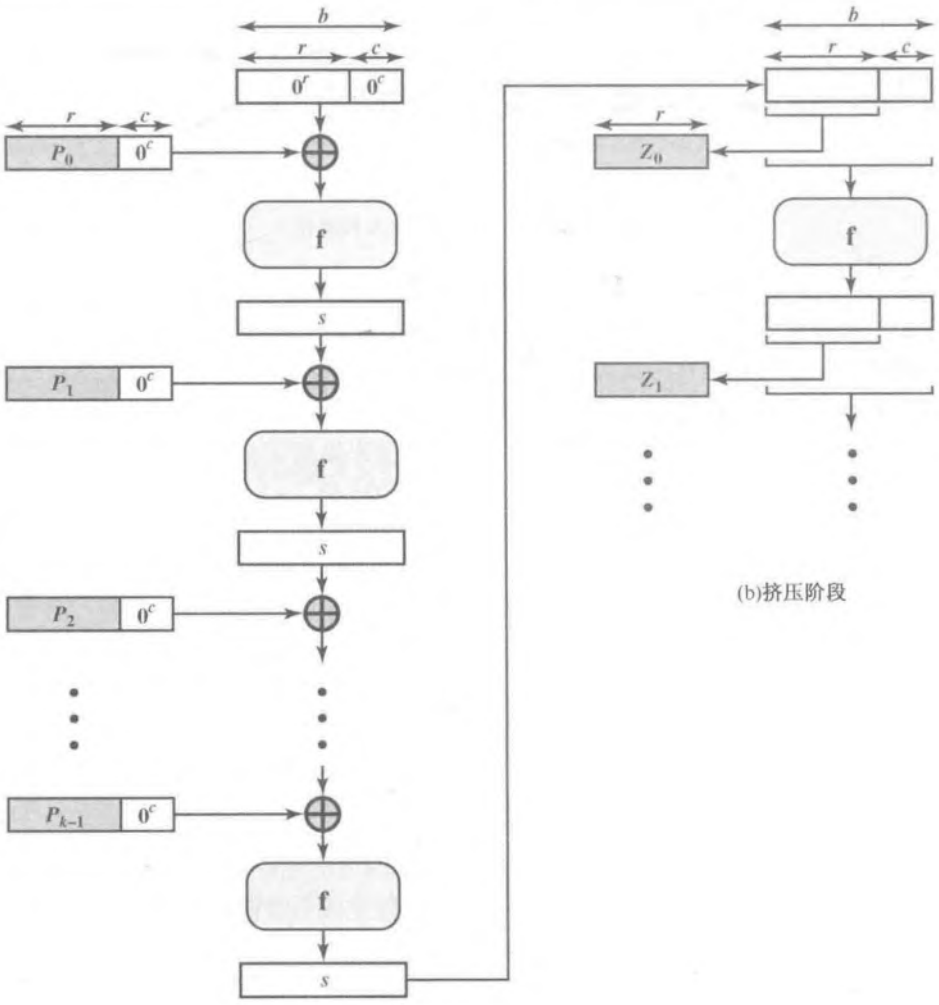


图 11.15 海绵函数的迭代结构

表 11.5　SHA-3 参数

消息摘要长度	224	256	384	512
消息长度	没有限制	没有限制	没有限制	没有限制
分组长度（位速率 r）	1152	1088	832	576
字长度	64	64	64	64
圈数	24	24	24	24
容量 c	448	512	768	1024
抗碰撞攻击	2^{112}	2^{128}	2^{192}	2^{256}
抗第二原像攻击	2^{224}	2^{256}	2^{384}	2^{512}

注：所有的长度和安全度量以二进制位为单位。

在算法的定义中，使用如下的符号：$|M|$是位串 M 的位长度。位串 M 可以被看成若干个长度固定为 x 的分组，当然最后一个分组长度可能是短块。M 的分组个数记为$|M|_x$。M 的各个分组记为 M_i，其下标从 0 到$|M|_x-1$索引。符号$\lfloor M \rfloor_\ell$表示对 M 截取其前 ℓ 位。

SHA-3 使用的迭代函数 f，记为 Keccak-f，将在下一节详细介绍。SHA-3 函数整体上是海绵函数结构，由于 SHA-3 有两个可操作的参数：消息分组长度 r 和容量 c，因此表示为 Keccak[r，c]，其默认值为 $r+c=1600$ 位。如表 11.5 所示，基于海绵结构的 Hash 函数的安全性的度量与容量 c 具有对应函数关系。

根据上述海绵结构的算法定义，Keccak[r，c]的定义如下：

$$\text{Keccak}[r,c] \triangleq \text{SPONGE}[\text{Keccak}-f[r+c], \text{pad10*1}, r]$$

接下来我们讨论迭代函数 Keccak-f。

11.6.2　SHA-3 迭代函数 f

下面我们介绍迭代函数 Keccak-f，其用于处理每个连续的输入消息分组。f函数的输入是 1600 位的状态变量 s，s 由与消息分组长度相等的 r 位和容量 c 位连接组成。在 f 的内部处理过程中，输入状态变量 s 被排列为 5×5×64 的矩阵 a。64 位单元被称为纵（lane）。为描述方便我们使用符号 $a[x,y,z]$ 表示状态矩阵中单独的一位。当更关心对于整个纵的操作时，我们将 5×5 矩阵记为 $L[x,y]$，这里 L 中的每个元素都是 64 位的纵。图 11.16 展示了矩阵元素下标的索引方法①。于是列号被索引从 x=0 到 x=4 标记，行号被索引从 $y=0$ 到 $y=4$ 标记，而纵中的各个位被索引从 $z=0$ 到 $z=63$ 标记。状态变量 s 中的位和符号 a 的对应关系如下：

$$s[64(5y+x)+z] = a[x,y,z]$$

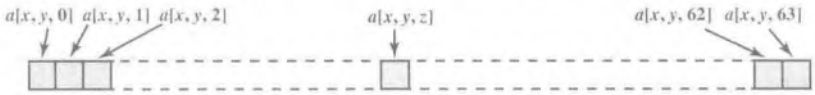
图 11.16 中的矩阵是对上述对应关系的图形化表示。当将状态看成由纵构成的矩阵时，第一个纵是左下角的 L[0，0]，对应着 s 中的前 64 位；第二个纵是最后一行第二列的 L[1，0]，对应着 s 中接下来的 64 位。以此类推，矩阵 a 从 y=0 行开始逐行被 s 的所有位填满。

f 的结构　函数 f 在每个输入消息块被 Hash 时执行一次。函数的输入是 1600 位的状态变量 s，将 s 转换为由 64 位纵构成的 5×5 矩阵。然后函数对矩阵执行 24 圈的操作，每一圈包括 5 个步骤，每个步骤通过置换或代替操作对状态矩阵进行更新。如图 11.17 所示，每圈的操作除了最后一步之外全都相同，最后一步通过使用不同的圈常数使得各圈操作互不相同。

① 注意第一个索引（x）表示列，第二个索引（y）表示行。这与数学类书籍里用第一个索引表示行、第二个索引表示列的传统习惯不同（例如，Knuth.D 的 *The Art of Computing Programming, Volume1, Fundamental Algorithms*；以及 Korn.G 的 *Mathematical Handbook for Scientists and Engineers* 等书籍）。

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=4$	$L[0,4]$	$L[1,4]$	$L[2,4]$	$L[3,4]$	$L[4,4]$
$y=3$	$L[0,3]$	$L[1,3]$	$L[2,3]$	$L[3,3]$	$L[4,3]$
$y=2$	$L[0,2]$	$L[1,2]$	$L[2,2]$	$L[3,2]$	$L[4,2]$
$y=1$	$L[0,1]$	$L[1,1]$	$L[2,1]$	$L[4,1]$	$L[4,1]$
$y=0$	$L[0,0]$	$L[1,0]$	$L[2,0]$	$L[3,0]$	$L[4,0]$

(a)状态变量表示为由64位字构成的5×5矩阵形式



(b)64位字的按位索引标记

图 11.16 SHA-3 状态矩阵

5 个步骤的操作可以表示成函数的复合形式[①]

$$R = i \circ \chi \circ \pi \circ \rho \circ \theta$$

表 11.16 总结了 5 个步骤的操作。因为这些步骤具有简单的描述，所以整个算法描述紧凑并且没有后门能被隐藏其中。算法中对于纵的操作仅限于位运算（XOR，AND，NOT）和循环移位。实现时不需要查表、算术运算或依赖于数据的移位操作。因此，SHA-3 能够简单、高效地通过软硬件实现。

我们依次来介绍函数中的各步骤运算。

θ 步函数 Keccak 的参考文档中将 θ 步函数描述如下：对于位于第 x 列、第 y 行的第 z 位

$$\theta: a[x,y,z] \leftarrow a[x,y,z] \oplus \sum_{y'=0}^{4} a[(x-1),y',z] \oplus \sum_{y'=0}^{4} a[(x+1),y',(z-1)] \qquad (11.1)$$

这里的求和运算均为 XOR 模二加操作。通过图 11.18(a)我们能够更清晰地理解操作的计算结果。首先，定义对于 x 列的各纵的 XOR 异或运算如下：

$$C[x] = L[x,0] \oplus L[x,1] \oplus L[x,2] \oplus L[x,3] \oplus L[x,4]$$

其中，以第 x 列，第 y 行的纵 $L[x, y]$为例。式（11.1）中的第一个求和项是对于第$(x-1)$ mod 4 列中的所有纵进行按位异或 XOR 运算，得到 64 位的纵 $C[x-1]$。第二个求和项是对于第$(x+1)$ mod 4 列中的所有纵进行按位异或 XOR 运算，然后在 64 位纵内部进行循环移位使得原位置第 z 位映射到第 $z+1$ mod 64 位，从而得到纵 ROT($C[x+1]$, 1)。上述两个纵与 $L[x, y]$一起通过按位异或 XOR 运算合并形成更新后的 $L[x, y]$。该过程可表示为

$$L[x,y] \leftarrow L[x,y] \oplus C[x-1] \oplus \mathrm{ROT}(C[x+1],1)$$

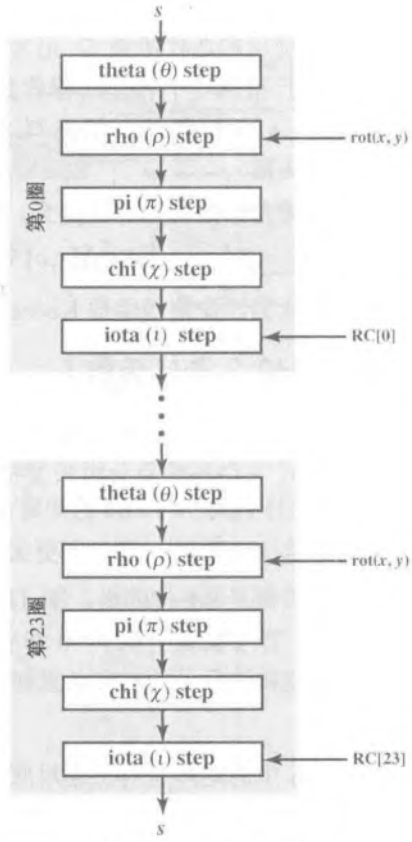


图 11.17 SHA-3 迭代函数 f

① 如果 f 和 g 是两个函数，那么满足等式 $y=F(x)=g[f(x)]$的函数 F 称为 f 和 g 的复合函数，记为 $F=g\circ f$。

表 11.6　SHA-3 的步函数

函数	类型	描　述
θ	代替	每个字中的每一位的新值，取决于其当前值、其前一列的每个字的相同位、后一列的每个字的邻接位
ρ	置换	对于每个字的内部使用循环移位进行置换操作，其中 $W[0,0]$不变
π	置换	字之间进行 5×5 矩阵的置换，其中 $W[0,0]$不变
χ	代替	每个字中的每一位的新值，取决于其当前值、相同行的下一个字的对应位的值、相同行的下下一个字的对应位的值
ι	代替	$W[0,0]$与圈常数进行 XOR 运算

图 11.18(a)展示了 $L[3,2]$的计算过程。在矩阵中所有的其他各纵执行同样的操作过程。

通过观察不难发现以下结论。纵中的每一位都进行了更新，其取值由自身的原值、其前一列的每一纵的相同位、后一列的每一纵的邻接位共同决定。因此每一位的更新值与原来的 11 位的取值相关，这提供了很好的混淆效果。同时，按照第 4 章的定义，θ 步函数也提供了很好的扩散效果。Keccak 的设计者指出 θ 步函数提供了高强度的扩散，如果没有 θ 步函数，轮函数的扩散效果将非常不明显。

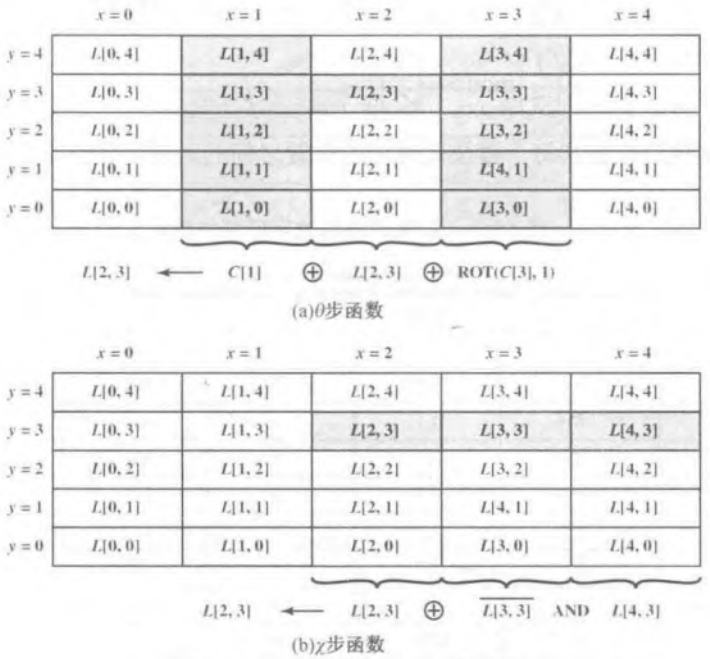


图 11.18　θ 步函数和 χ 步函数

ρ 步函数　ρ 步函数的定义如下：

$$\rho: a[x,y,z] \leftarrow a[x,y,z]\text{，如果 } x=y=0$$

否则

$$\rho: a[x,y,z] \leftarrow a[x,y,\left(z-\frac{(t+1)(t+2)}{2}\right)] \tag{11.2}$$

这里 t 满足 $0 \leqslant t < 24$ 且在 $\mathrm{GF}(5)^{2\times 2}$ 上有 $\begin{pmatrix}0 & 1\\ 2 & 3\end{pmatrix}^t\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}x\\y\end{pmatrix}$。

从上述公式很难直观地理解该步骤的操作过程，因此我们观察操作的细节：

（1）位置在（x，y）=（0，0）的纵，即 $L[0,0]$，不变。对于其他的字，将在字内执行循环移位。

（2）变量 t，$0 \leqslant t < 24$，用于计算循环移位的位数，以及决定哪个纵使用哪个移位值。

（3）24 个不同的移位方式，移位值分别是 $\frac{(t+1)(t+2)}{2} \bmod 64$。

（4）对于 5×5 矩阵上的各纵，位置在（x，y）的纵上的移位操作由 t 的值决定。具体地说，对于 t 的每一个取值，对应的矩阵位置由 $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 决定。例如，对于 t=3，有

$$\begin{aligned}
\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^3 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bmod 5 \\
&= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bmod 5 \\
&= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \bmod 5 \\
&= \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 6 \end{pmatrix} \bmod 5 = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \bmod 5 \\
&= \begin{pmatrix} 1 \\ 7 \end{pmatrix} \bmod 5 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}
\end{aligned}$$

表 11.7 给出了移位的位数和每个移位值对应的位置之间的所有计算结果。注意结果中所有的移位值都不相同。

表 11.7 SHA-3 使用的循环移位值

(a)移位值和位置的计算结果

t	$g(t)$	$g(t)$ mod 64	x, y	t	$g(t)$	$g(t)$ mod 64	x, y
0	1	1	1,0	12	91	27	4,0
1	3	3	0,2	13	105	41	0,3
2	6	6	2,1	14	120	56	3,4
3	10	10	1,2	15	136	8	4,3
4	15	15	2,3	16	153	25	3,2
5	21	21	3,3	17	171	43	2,2
6	28	28	3,0	18	190	62	2,0
7	36	36	0,1	19	210	18	0,4
8	45	45	1,3	20	231	39	4,2
9	55	55	3,1	21	253	61	2,4
10	66	2	1,4	22	276	20	4,1
11	78	14	4,4	23	300	44	1,1

注：$g(x) = (t+1)(t+2)/2$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} \bmod 5$$

(b) 矩阵中各字的循环移位值

	$x = 0$	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = 4$	18	2	61	56	14
$y = 3$	41	45	15	21	8
$y = 2$	3	10	43	25	39
$y = 1$	36	44	6	55	20
$y = 0$	0	1	62	28	27

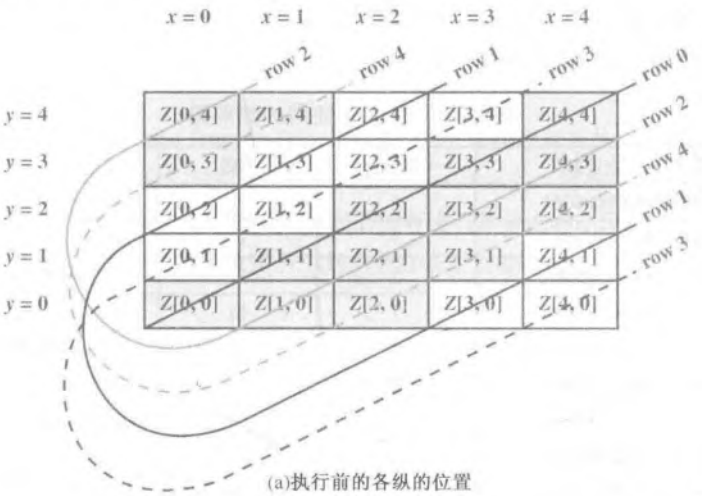
ρ 步函数的操作由各个纵内部的简单置换（循环移位）组成。其目的是提供每个纵内部的扩散。如果没有 ρ 步函数，纵之间的各个位之间的扩散将会非常缓慢。

π 步函数　π 步函数的定义如下：

$$\pi: a[x,y] \leftarrow a[x',y'], \text{ 这里} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{11.3}$$

上式可以写成 $(x,y)\times(y,(2x+3y))$。于是 5×5 矩阵上的各纵之间互相移位，新的 x 坐标等于原 y 坐标，新的 y 坐标等于 $(2x+3y) \bmod 5$。图 11.19 给出了该置换的图像化描述。执行 π 函数之前相同对角线（从左向右，增加 y 值）的纵，在执行 π 步函数之后被置换到同一行中。注意 $L[0, 0]$的位置不变。

因此 π 步函数是对于纵之间的置换：各纵在 5×5 矩阵内部移动位置。ρ 步函数是对于位的置换操作：纵内部的各位进行循环移位。注意 π 步函数的矩阵位置的计算方式与 ρ 步函数相同，移位常数的一维向量序列被映射到矩阵的各纵中。



(a)执行前的各纵的位置

	x = 0	x = 1	x = 2	x = 3	x = 4
y = 4	Z[2, 0]	Z[3, 1]	Z[4, 2]	Z[0, 3]	Z[1, 4]
y = 3	Z[4, 0]	Z[0, 1]	Z[1, 2]	Z[2, 3]	Z[3, 4]
y = 2	Z[1, 0]	Z[2, 1]	Z[3, 2]	Z[4, 3]	Z[0, 4]
y = 1	Z[3, 0]	Z[4, 1]	Z[0, 2]	Z[1, 3]	Z[2, 4]
y = 0	Z[0, 0]	Z[1, 1]	Z[2, 2]	Z[3, 3]	Z[4, 4]

(b)置换后的各纵的位置

图 11.19　π 步函数

χ 步函数　χ 步函数的定义如下：

$$\chi: a[x] \leftarrow a[x] \oplus ((a[x+1] \oplus 1) \text{AND } a[x+2]) \tag{11.4}$$

χ 步函数对每一位进行更新，其取值由自身的当前值、与当前位置相对应的同一行中接下来相邻两纵的取值共同决定。如果我们对单独的一位 $a[x, y, z]$进行讨论，该操作写成如下的布尔函数形式会更加明晰：

$$a[x,y,z] \leftarrow a[x,y,z] \oplus (\text{NOT}(a[x+1,y,z])) \text{AND} (a[x+2,y,z])$$

图 11.18(b)展示了 χ 步函数对于纵 $L[3, 2]$上各个位的运算。这是所有操作中唯一的非线性映射步骤。如果没有 χ 步函数，整个 SHA-3 圈函数将成为线性函数。

ι 步函数 ι 步函数的定义如下：

$$\iota: a \leftarrow a \oplus RC[i_r] \tag{11.5}$$

该函数通过将每圈个不相同的圈常数加入一组元素中。这打破了其他 4 个步骤函数中的对称性。事实上，式（11.5）看起来有些容易让人误解，圈常数仅被用于内部状态矩阵中的第一个纵。我们将其表示为如下形式：

$$L[0,0] \leftarrow L[0,0] \oplus RC[i_r], \quad 0 \leqslant i_r \leqslant 24$$

表 11.8 列出了 24 个 64 位圈常数。注意圈常数的汉明权重（即 1 的个数）从 1～6，大部分位的取值为 0，从而不改变 $L[0, 0]$中相应位的结果。如果将 24 个圈常数累积进行或 OR 运算，得到

$$RC[0] \text{ OR } RC[1] \text{ OR} \ldots \text{OR } RC[23] = 800000008000808B$$

因此，仅有 7 个位是活跃的，能够影响到 $L[0, 0]$的取值。当然，圈与圈之间通过置换和代替操作使得 ι 函数的效果传播到矩阵中所有的纵和所有的位。显而易见通过单圈操作 θ 和 χ 函数的作用，圈常数的扰乱充分扩散到状态中所有的纵中。

表 11.8 SHA-3 使用的圈常数

圈数	常数（十六进制）	1 的个数	圈数	常数（十六进制）	1 的个数
0	0000000000000001	1	12	000000008000808B	6
1	0000000000008082	3	13	800000000000008B	5
2	800000000000808A	5	14	8000000000008089	5
3	8000000080008000	3	15	8000000000008003	4
4	000000000000808B	5	16	8000000000008002	3
5	0000000080000001	2	17	8000000000000080	2
6	8000000080008081	5	18	000000000000800A	3
7	8000000000008009	4	19	800000008000000A	4
8	000000000000008A	3	20	8000000080008081	5
9	0000000000000088	2	21	8000000000008080	3
10	0000000080008009	4	22	0000000080000001	2
11	000000008000000A	3	23	8000000080008008	4

11.7 关键术语、思考题和习题

关键术语

吸水阶段	高位在前格式	生日攻击	生日悖论
位速率	容量	χ 步函数	抗碰撞攻击
压缩函数	密码学 Hash 函数	Hash 码	Hash 函数
Hash 值	ι 步函数	Keccak	带密钥 Hash 函数
纵	低位在前格式	MD4	MD5
消息认证码（MAC）	消息摘要	单向 Hash 函数	π 步函数
抗原像攻击	ρ 步函数	抗第二原像攻击	SHA-1
SHA-224	SHA-256	SHA-3	SHA-384

SHA-512　　海绵结构　　挤压阶段　　抗强碰撞攻击

θ 步函数　　抗弱碰撞攻击

思考题

11.1　安全 Hash 函数需要具有哪些特性？

11.2　抗弱碰撞和抗强碰撞之间的区别是什么？

11.3　Hash 函数中的压缩函数的作用是什么？

11.4　高位在前格式和低位在前格式的区别是什么？

11.5　SHA 中使用的基本算术和逻辑函数是什么？

11.6　描述 NIST 评估 SHA-3 候选算法的一系列标准。

11.7　给出海绵结构的定义。

11.8　简要描述迭代函数 f 的内部结构。

11.9　列出并简要描述构成迭代函数 f 的各步骤函数。

习题

11.1　高速传输协议 XTP（Xpress Transfer Protocol）使用 32 位校验和函数，它是两个 16 位函数的连接，两个函数为 XOR 和 RXOR（参见 11.4 节）图 11.5 中所示的“两种简单的 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，试分析其有效性。

11.2　（a）考虑 11.4 节中给出的 Davies 和 Price 提出的 Hash 码方案，假定使用 DES 作为加密函数

$$H_i = H_{i-1} \oplus \mathrm{E}(M_i, H_{i-1})$$

前面讲过 DES 的互补性（参见习题 3.14）：若 $Y = \mathrm{E}(K, X)$，则 $Y' = \mathrm{E}(K', X')$。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）证明存在有类似的攻击可成功地攻击文献[MEYE88]中提出的方法

$$H_i = M_i \oplus \mathrm{E}(H_{i-1}, M_i)$$

11.3　（a）考虑下列 Hash 函数。消息是 Z_n 中的整数序列，$M=(a_1,a_2,\cdots,a_t)$。对于某个预定义好的 n，计算 Hash 值 $h = \left(\sum_{i=1}^{t} a_i\right)$。这个 Hash 函数满足表 11.1 所列出的 Hash 函数的要求吗？给出解释。

（b）对于 Hash 函数 $h = \left(\sum_{i=1}^{t} (a_i)^2\right) \bmod n$，按（a）的要求做题。

（c）当 $M = (189, 632, 900, 722, 349)$, $n=989$ 时，计算（b）中的 Hash 函数值。

11.4　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，而分组密码是可逆的（为了能够解密），那么如何用 Hash 码构造分组密码呢？

11.5　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。通过解决下面问题，说明该方法是不安全的。给定两个分组消息 B1，B2，其 Hash 码为

$$\mathrm{RSAH(B1,B2)} = \mathrm{RSA(RSA(B_1)} \oplus \mathrm{B2)}$$

给定任一分组 C1，选择 C2 使得 RSAH(C1, C2) = RSAH(B1, B2)。因此该 Hash 函数不满足抗弱碰撞性。

11.6 设 H(m)是一个抗碰撞 Hash 函数，将任意长消息映射为定长的 n 位 Hash 值。对于所有的消息 x,x'，$x\neq x'$，都有 $H(x)\neq H(x')$。上述说法是否正确？给出解释。

11.7 图 11.12 里，假设有一个大小为 80 个字的数组（字长为 64 位）来存储 W_t的值，所以在开始处理一个分组时可以预计算它们的值。现在假设空间是短缺资源。作为替代方案，考虑使用 16 个字大小的循环缓冲区，用值 W_0到 W_{15}进行初始化。设计一个算法，对于每一步 t，计算需要的输入 W_t。

11.8 给出 SHA-512 中 $W_{16}, W_{17}, W_{18}, W_{19}$的值。

11.9 对于 SHA-512 如果消息分别为如下长度，给出填充区域的值。

（a）1919 位

（b）1920 位

（c）1921 位

11.10 对于 SHA-512 如果消息分别为如下长度，给出长度区域的值。

（a）1919 位

（b）1920 位

（c）1921 位

11.11 假定 $a_1a_2a_3a_4$是长为 32 位的字中的 4 个字节，将每个 a_i看成是 0～255 之间的二进制整数，在高位在前格式中，该字表示整数

$$a_1 2^{24}+a_2 2^{16}+a_3 2^8+a_4$$

在低位在前格式中，该字表示整数

$$a_4 2^{24}+a_3 2^{16}+a_2 2^8+a_1$$

（a）一些 Hash 函数如 MD5 使用的是低位在前格式。注意，消息摘要不依赖于低层结构这一点很重要，因此要在高位在前格式平台上执行 MD5 和 RIPEMD-160 的模 2 加法运算，必须进行调整。假定 $X=x_1x_2x_3x_4$且 $Y=y_1y_2y_3y_4$，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 使用高位在前格式，说明 SHA 的运算$(X+Y)$如何在低位在前格式的机器上执行。

11.12 本习题介绍了一个本质上和 SHA 类似的 Hash 函数，它对字母操作，而不是二元数据，称为趣味四字母 Hash（tth）[①]。给定一个由字母序列构成的消息，tth 产生一个四字母的 Hash 值。首先，tth 将消息划分为 16 个字母的分组，忽略空格，标点符号和大写。如果消息不能被 16 整除，用空白来填充。维持一个四个数的运行和，其初始值为(0,0,0,0)。将其输入压缩函数用于处理第一个分组。压缩函数由两轮组成。

第一轮 取下一文本分组，按行序排列成 4×4 的文本分组，并将文本转换为数字（A=0,B=1,等等）。例如，对于分组 ABCDEFGHIJKLMNOP，我们有

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	*P*

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

接着，将各列分别模 26 加，并将结果和运行和模 26 加。本例中，运行和为（24，2，6，10）。

① 感谢 *The Cryptogram* 杂志的 William K. Mason 提供了这个例子。

第二轮　运用第一轮的矩阵，将第一行循环左移 1 位，第二轮 2 位，第三轮 3 位，而第四轮则是将元素反序。本例中，有

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

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

现在，将各列分别模 26 相加，并将结果和运行和相加。新的运行和为（5，7，9，11）。该运行和作为压缩函数的输入，处理下一文本分组。最后一个分组处理完后，将其转换为字母。例如，如果消息是 ABCDEFGHIJKLMNOP，则 hash 值为 FHJL。

（a）画图表示 tth 的总体逻辑和压缩函数的逻辑，该图与图 11.9 和图 11.10 相仿。

（b）计算 48 个字母的消息“I leave twenty million dollars to my friendly cousin Bill.”的 hash 值。

（c）为了说明 tth 的弱点，请找出 48 个字母的分组也产生相同的 Hash 值。提示：使用多个 A。

11.13　对于表 11.5 中 SHA-3 所有可能的容量值，内部 55 状态矩阵中的哪些纵是全 0 的？

11.14　考虑分组长度为 1024 位的 SHA-3，假设在第一个消息分组（P_0）中的每一纵都至少有一个非零位。开始时，内部状态矩阵中对应容量部分的初始状态都是零。说明经过多少运算后所有的纵都至少有一个非零位。注：忽略掉置换。即跟踪原始的零纵，即便在矩阵中它们的位置已经改变。

11.15　考虑图 11.16(a)中的状态矩阵。现在重新排列矩阵的行和列，使得 $L[0, 0]$在正中间。例如将列从左到右按照（x=3，x=4，x=0，x=1，x=2）排列，将行从上到下按照（y=2，y=1，y=0，x=4，y=3）排列。考察该函数使用的置换算法以及函数中循环置换常数。使用该重新排列的矩阵，描述其置换算法。

11.16　ι 函数仅影响 $L[0, 0]$。11.6 节讲述了 $L[0, 0]$的改变通过单圈的 θ 函数扩散到状态的所有纵中。

（a）证明上述事实。

（b）矩阵中所有的位都受到 $L[0, 0]$改变的影响，需要经过多少次运算？

第 12 章　消息认证码

学 习 目 标

通过本章的学习应具有以下能力：

- 列举并解释对消息认证的可能攻击
- 掌握消息认证码的定义
- 列举并解释消息认证码的安全性需求
- 给出 HMAC 的概览
- 给出 CMAC 的概览
- 解释认证加密的概念
- 给出 CCM 的概览
- 给出 GCM 的概览
- 讨论密钥封装的概念和用途
- 理解 Hash 函数或消息认证码如何用于伪随机数产生

消息认证和数字签名是密码学中最吸引人也是最复杂的研究领域之一。只用少量的篇幅来介绍所有已提出或实现的用于消息认证与数字签名的密码函数和协议是不可能的。本章和下一章的目的是概要介绍上述内容，并给出对各种方法的系统描述。

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，然后归纳总结一些基本方法。本章其余部分讨论实现消息认证的基本方法，即消息认证码；在对 MAC 进行概述和安全性需求分析后，随后将讨论两类 MAC：使用密码学 Hash 函数的 MAC 和使用分组密码的 MAC。然后我们将讨论较新的认证加密。最后将介绍使用 Hash 函数和 MAC 的伪随机数发生器。

12.1　对消息认证的要求

在网络通信环境中，可能有下述攻击：

（1）消息透露给没有合法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**　对消息的延时和重播。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播，也可能是消息序列中的一条消息被延时或重播；在面向无连接的应用中，可能是一条消息（如数据报）被延时或重播。

（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）种攻击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，且是未被修改的消息，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。

12.2　消息认证函数

任何消息认证或数字签名机制在功能上基本都有上下两层。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：

- **Hash 函数**　它是将任意长的消息映射为定长的 Hash 值的函数，以该 Hash 值作为认证符。
- **消息加密**　对整个消息加密后的密文作为认证符。
- **消息认证码（MAC）**　它是消息和密钥的函数，它产生定长的值，以该值作为认证符。

我们已经在第 11 章讨论了 Hash 函数及其如何用于消息认证。本节将简要讨论剩余两类问题。本章剩余部分节将重点讨论关于 MAC 的话题。

12.2.1　消息加密

消息加密本身提供了一种认证手段。对称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［参见图 12.1(a)］。发送方 A 用 A 和 B 共享的密钥 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。

此外，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 能产生出可用 K 解密的密文，所以该消息一定来自于 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于 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。显然，若 M 可以是任意的位模式，那么不管 X 的值是什么，$Y = D(K, X)$ 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就只有 10^{-6}。

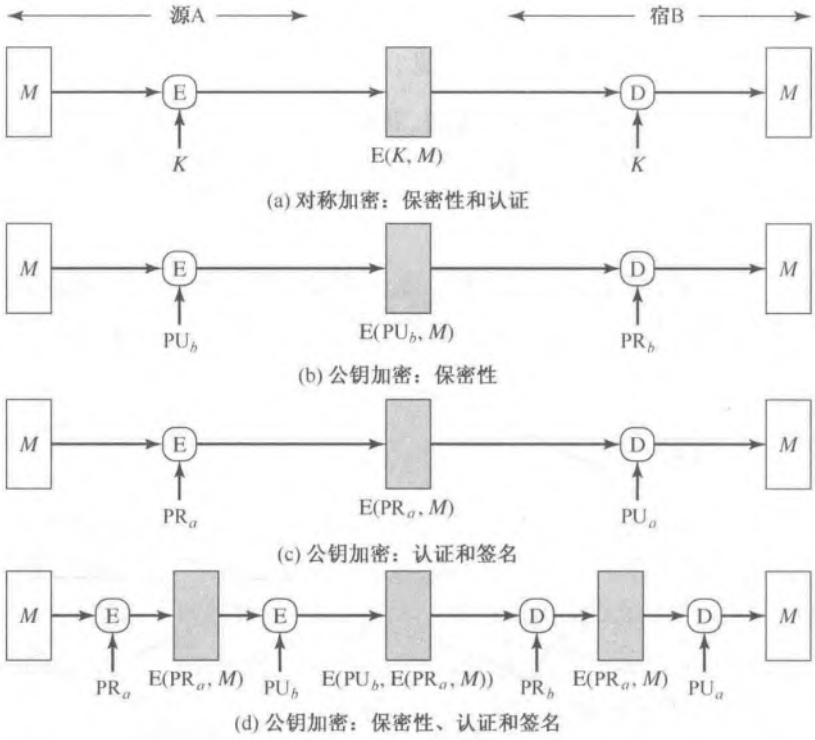


图 12.1 消息加密的基本使用

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：

```
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所得明文的可读性进行自动判别，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，即是真实的明文。因此，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 12.2(a)所示。A 准备发送明文消息 M，那么 A 将 M 作为 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，并将其视为消息和附加的 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。

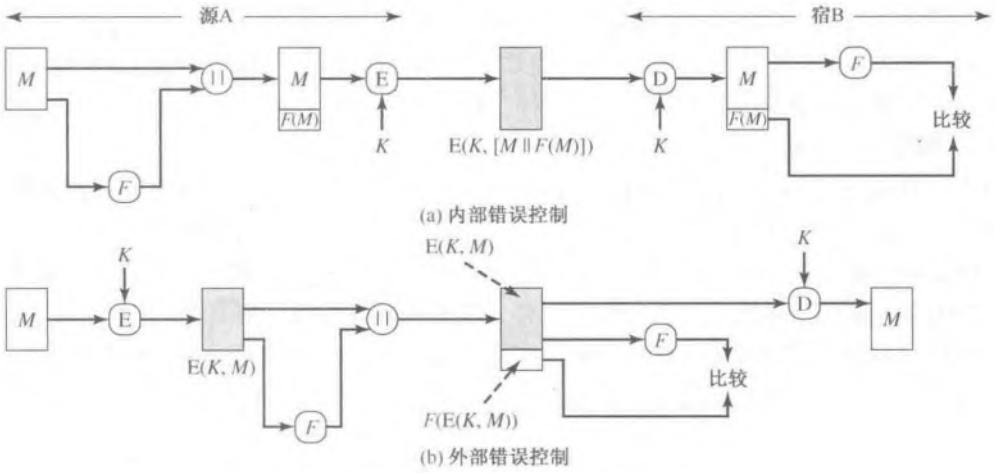


图 12.2　内部和外部错误控制

请注意，FCS 和加密函数执行的顺序很重要。文献[DIFF79]中将图 12.2(a)所示的这种序列称为内部错误控制，以与外部错误控制［参见图 12.2(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。

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 12.3 给出的 TCP 段的格式并说明了 TCP 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 头外的所有数据报加密，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会包含有意义的头信息。在这种方法中，头不仅包含校验和（校验整个头部），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。

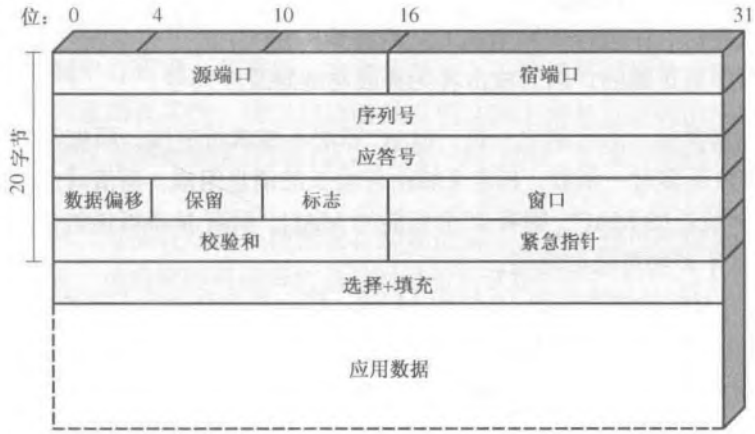


图 12.3　TCP 段

公钥加密　直接使用公钥加密［参见图 12.1(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 PU_b 对 M 加密，由于只有 B 拥有相应的私钥 PR_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 用自己的私钥对消息加密，而 B 用 A 的公钥对接收的消息解密［参见图 12.1(c)］。和对称密码情形的推理一样，这提供了认证功能：因为只有 A 拥有 PR_a，能产生用 PU_a 可解密的密文，所以该消息一定来自于 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。

假定明文具有这种结构，那么图 12.1(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 PR_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于 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［参见图 12.1(d)］。但这种方法的缺点是，一次通信中要执行四次而不是两次复杂的公钥算法。

12.2.2 消息认证码

消息认证码，又称密码校验和或者 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

$$\mathrm{MAC} = \mathrm{C}(K, M)$$

其中，M 为输入消息；C 为 MAC 函数；K 为共享的密钥；MAC 为消息认证码。

消息和 MAC 一起被发送给接收方。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［参见图 12.4(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

（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} 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 12.4(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［参见图 12.4(b)］或之前［参见图 12.4(c)］对消息加密，则可以获得保密性。

① 这不是构造数字签名的一般方法，但是我们将会看到，它们的基本原理是相同的。

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 12.4(b)中的方法。

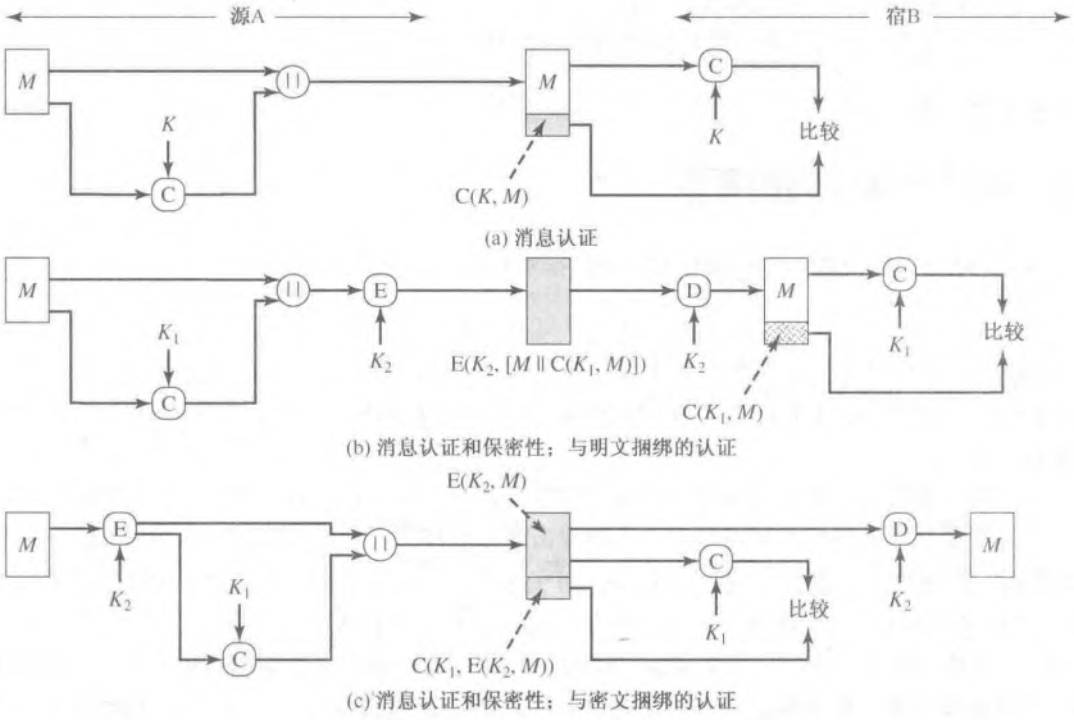


图 12.4　消息认证码（MAC）的基本用途

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重新设计的消息认证码呢？文献[DAVI89]中提出了三种必须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。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：

（1）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，但对这些应用不必加密 SNMP 传输。

（2）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，可能希望提供保密性。

（3）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。

最后，要注意的是，由于收发双方共享密钥，因此 MAC 不能提供数字签名。

12.3 对消息认证码的要求

MAC 也称为密码校验和，它由如下形式的函数产生：

$$T = \mathrm{MAC}(K, M)$$

其中 M 是一个变长消息，K 是收发双方共享的密钥，MAC(K, M)是定长的认证符，有时也称为标记（tag）。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。

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。除非算法中本身存在某些弱点，否则攻击者必须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，对于唯密文攻击，若给定密文 C，攻击者要对所有可能的 K_i 计算 $P_i = \mathrm{D}(K_i, C)$，直到产生的某 P_i 具有适当的明文结构为止。

对 MAC 情况则完全不同。一般 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 $T_1 = \mathrm{MAC}(K, M_1)$的 M_1 和 T_1，密码分析者要对所有可能的密钥值 K_i 计算 $T_i = \mathrm{MAC}(K_i, M_1)$，那么至少需要有一个密钥会使得 $T_i = T_1$。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 及 $T_1 = \mathrm{MAC}(K, M_1)$
  对所有 2^k 个密钥，计算 $T_i = \mathrm{MAC}(K_i, M_1)$
  匹配数 $\approx 2^{(k-n)}$
- **循环 2**
  给定 M_2 及 $T_2 = \mathrm{MAC}(K, M_2)$
  对余下的 $2^{(k-n)}$ 个密钥，计算 $T_i = \mathrm{MAC}(K_i, M_2)$
  匹配数 $\approx 2^{(k-2\times n)}$

以此类推。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，第三次循环则得到唯一一个密钥，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，标记）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。不过可能存在有无须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

$$\Delta(M)=X_1\oplus X_2\oplus\cdots\oplus X_m$$
$$\mathrm{MAC}(K,M)=\mathrm{E}(K,\Delta(M))$$

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||MAC(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 是按如下方式计算的：

$$Y_m=Y_1\oplus Y_2\oplus\cdots\oplus Y_{m-1}\oplus\Delta(M)$$

攻击者可以将 Y_1 至 Y_m 与原来的 MAC 连结成一个新的消息，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的长为 64×(m−1)位的消息。

因此，评价 MAC 函数的安全性时，应考虑对该函数的各种类型的攻击。下面将介绍 MAC 函数应满足的要求。假定攻击者知道 MAC 函数，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 MAC(K, M)，则他构造满足 MAC (K, M')= MAC (K, M)的消息 M' 在计算上是不可行的。

（2）MAC(K, M)应是均匀分布的，即对任何随机选择的消息 M 和 M'，MAC (K, M)= MAC (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 可能是将 M 的一位或多位取反，要求 Pr[MAC(K, M)= MAC(K, M')]= 2^{-n}。

前面已讲过，攻击者即使不知道密钥，他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 MAC (K, M)的攻击者可以对 M 的已知的“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。

12.4　MAC 的安全性

同加密算法和 Hash 函数一样，我们也可将对 MAC 的攻击分为两类：穷举攻击和密码分析。

12.4.1　穷举攻击

对 MAC 的穷举攻击由于需要知道〈消息–MAC〉对，所以这种攻击会比对 Hash 函数的攻击更加困难，下面来分析其原因。攻击者可以按下述方式对 Hash 码进行攻击：对给定的消息 x 及其 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$[x_i, \text{MAC}(K, x_i)]$，对任何新的输入$x \neq x_i$，计算〈消息–MAC〉对$[x, \text{MAC}(K, x)]$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。下面我们依次来讨论这些攻击。

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 k 位，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，此处所需的代价约为 2^k（即对 2^k 个可能的密钥中的每一个执行一次操作）。但是，如前所述，因为 MAC 是多对一映射的，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〉对或者密钥信息，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。

12.4.2 密码分析

与对加密算法和 Hash 函数的攻击一样，对 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的。评价 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 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。

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。文献[PREN96]中概述了对某些特定 MAC 的密码分析攻击。

12.5 基于 Hash 函数的 MAC：HMAC

在本章后面部分，将讨论使用对称分组密码的消息认证码（MAC），这一直是构造 MAC 的最常用方法。近年来，人们越来越感兴趣于利用密码学 Hash 函数来设计 MAC，因为：

（1）一般像 MD5 和 SHA 这样的密码学 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）有许多共享的密码学 Hash 函数代码库。

随着 AES 的开发以及密码算法代码的可用性日益广泛，上述考虑的意义将会削弱，但是基于 Hash 函数的 MAC 仍将持续广泛使用。

诸如 SHA 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，已经提出了许多方案将密钥加到现有的 Hash 函数中。HMAC（RFC 2104）是最受支持的方案[BELL96a,BELL96b]，它是 IP 安全里必须实现的 MAC 方案，并且其他 Internet 协议中（如 SSL）也使用了 HMAC。HMAC 也已作为 NIST 的标准发布（FIPS 198）。

12.5.1　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 函数视为“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-3 替代 SHA-2），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。

12.5.2　HMAC 算法

图 12.5 给出了 HMAC 的总体结构。定义下列符号：

H 为嵌入的 Hash 函数（如 MD5, SHA-1, RIPEMD-160）

IV 为 Hash 函数输入的初始值

M 为 HMAC 的消息输入（包括由嵌入 Hash 函数定义的填充位）

Y_i 为 M 的第 i 个分组，$0 \leqslant i \leqslant (L-1)$

L 为 M 中的分组数

b 为每一分组所含的位数

n 为嵌入的 Hash 函数所产生的 Hash 码长

K 为密钥；建议密钥长度 $\geqslant n$。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

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

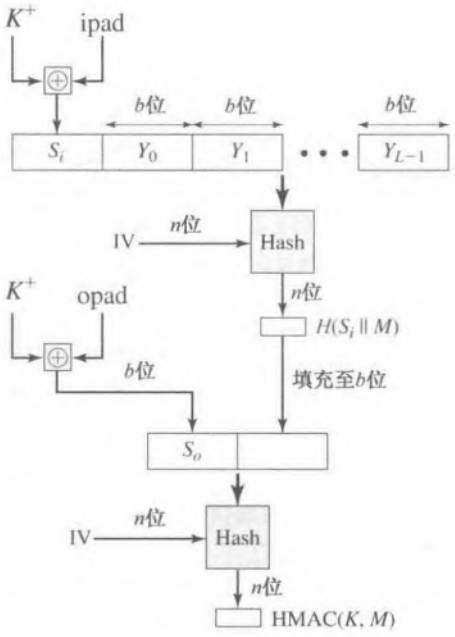


图 12.5　HMAC 的结构

HMAC 可描述如下：

$$\text{HMAC}(K, M) = H[(K^+ \oplus \text{opad}) \| H[(K^+ \oplus \text{ipad}) \| M]]$$

该算法描述如下：

（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 个字节的 0）。

（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_0。

（6）将步骤（4）中的 Hash 码附于 S_0 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 12.6 所示。我们可以预计算两个值：

$$f(\text{IV},(K^+ \oplus \text{ipad}))$$

$$f(\text{IV},(K^+ \oplus \text{opad}))$$

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要计算，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，在产生 MAC 的消息大多数都较短的情况下，这种实现特别有意义。

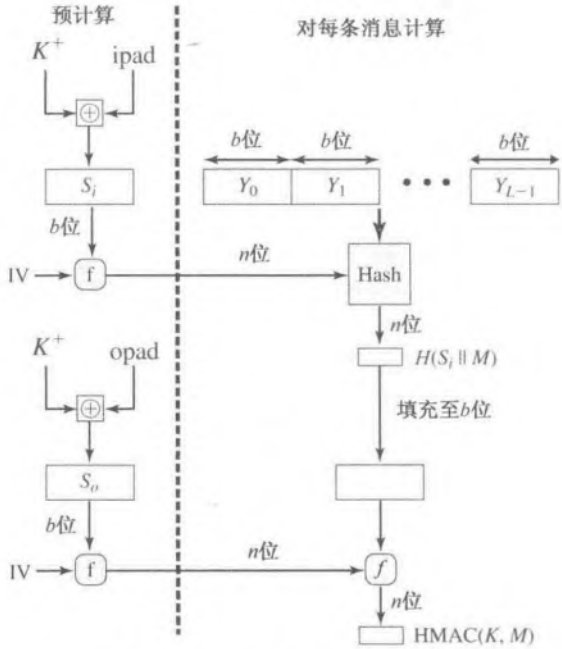


图 12.6 HMAC 的有效实现方案

12.5.3 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 Hash 函数的强度与 HMAC 的强度之间的联系。

在给定时间内，给定的一定数量的〈消息–MAC〉对（用相同密钥产生），伪造者伪造成功的概率可以用来描述 MAC 函数的安全性。文献[BELL96a]中已经证明，如果攻击者已知若干由合法用户生成的（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）对攻击者而言，即使 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，我们可将压缩函数视为将 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（其代价是 2^n 数量级的），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，我们已经证明当 Hash 码长为 n 位时其所需的代价是 $2^{n/2}$ 数量级的。根据现在的技术，若代价是 2^{64} 数量级的，则被认为是可计算的，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^{72} 位），对 1 Gb/s 连接，要想攻击成功，攻击者约需 150 000 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 作为 HMAC 的嵌入 Hash 函数，是完全可以接受的。

12.6　基于分组密码的 MAC：DAA 和 CMAC

本节我们讨论两个基于分组密码工作模式的 MAC 算法。首先将介绍数据认证算法（DAA），该算法较陈旧，目前已经被废止。然后介绍 CMAC 算法，该算法的设计克服了 DAA 算法的某些缺陷。

12.6.1　数据认证算法

数据认证算法（DAA）建立在 DES 之上，是多年以来使用最广泛的 MAC 算法之一，它是 FIPS 标准（FIPS PUB 113）和 ANSI 标准（X9.17）。然而我们随后会介绍，人们已经发现了这个算法的安全弱点，并正使用一个更新、更强的算法来代替它。

数据认证算法采用 DES 运算的密文块链接（CBC）方式（参见图 6.4），其初始向量为 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，…，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 12.7 所示。

$$
\begin{aligned}
O_1 &= E(K, D_1) \\
O_2 &= E(K, [D_2 \oplus O_1]) \\
O_3 &= E(K, [D_3 \oplus O_2]) \\
&\vdots \\
O_N &= E(K, [D_N \oplus O_{N-1}])
\end{aligned}
$$

其中，DAC 可以是整个块 O_N，也可以是其最左边的 M 位，其中 $16 \leqslant M \leqslant 64$。

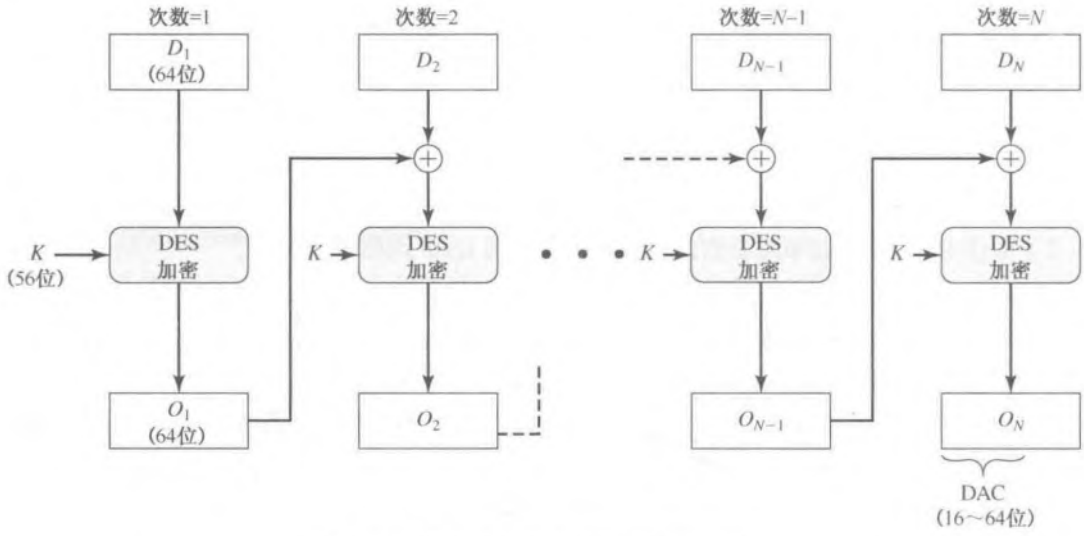


图 12.7 数据认证算法（FIPS PUB 113）

12.6.2 基于密码的消息认证码（CMAC）

我们已经提过 DAA 算法在政府和工业界被广泛采用。文献[BELL00]证明了在合理的安全准则下这种 MAC 是安全的，但有如下的限制：仅能处理固定长度为 mn 的消息，其中 n 是密文分组的长度，m 是一个固定的正整数。例如，给定一个消息分组 X 的 CBC MAC 码，如 T=MAC(K, X)，则攻击者马上就知道两个消息分组 $X\|(X\oplus T)$的 CBC MAC 码，因为这还是 T。

Black 和 Rogaway[BLAC00]证明了这种限制可以使用三个密钥来克服：一个长度为 k 的密钥 K，用在密文分组链接的每一步，两个长度为 n 的密钥，其中 k 是密钥长度，n 为密文分组长度。Iwata 和 Kurosawa 又优化了该构造，使得两个 n 位的密钥可以从加密密钥导出，而不是单独提供的[IWAT03]。这种优化已经被 NIST 采用作为基于密码的消息认证码（CMAC）的运算模式，对于 AES、3DES 适用。NIST 的专门出版物 800-38B 有该规范。

首先，当消息长度是分组长度 b 的 n 倍时，我们考虑 CMAC 的运算情况。对 AES，b=128，对于 3DES，b=64。这个消息被划分为 n 组（$M_1,M_2,\cdots,M_n$）。算法使用了 k 位的加密密钥 K 和 b 位的常数 K_1。对于 AES，密钥长度 k 为 128 位，192 位或 256 位，对于 3DES，密钥长度为 112 位或 168 位。CMAC 按如下方式计算（参见图 12.8）：

$$
\begin{aligned}
C_1 &= E(K, M_1)\\
C_2 &= E(K,[M_2 \oplus C_1])\\
C_3 &= E(K,[M_3 \oplus C_2])\\
&\vdots\\
C_n &= E(K,[M_n \oplus C_{n-1} \oplus K_1])\\
T &= \mathrm{MSB}_{\mathrm{Tlen}}(C_n)
\end{aligned}
$$

其中，T 为消息认证码，也称为 tag；Tlen 为 T 的位长度；$\mathrm{MSB}_s(X)$为位串 X 的最左边的 s 位。

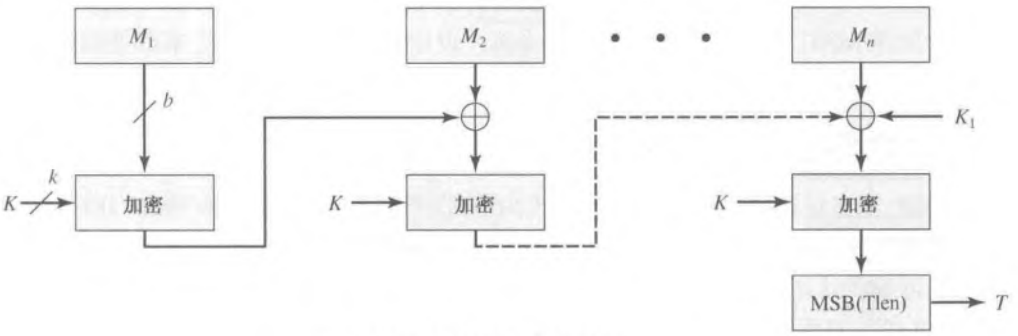
如果消息不是密文分组长度的整数倍，则最后分组的右边（低有效位）填充一个 1 和若干 0 使得最后的分组长度为 b。除了使用一个不同的 b 位密钥 K_2 代替 K_1 外，和前面所述的一样进行 CMAC 运算。

两个 b 位的密钥由 k 位的加密密钥按如下方式导出：

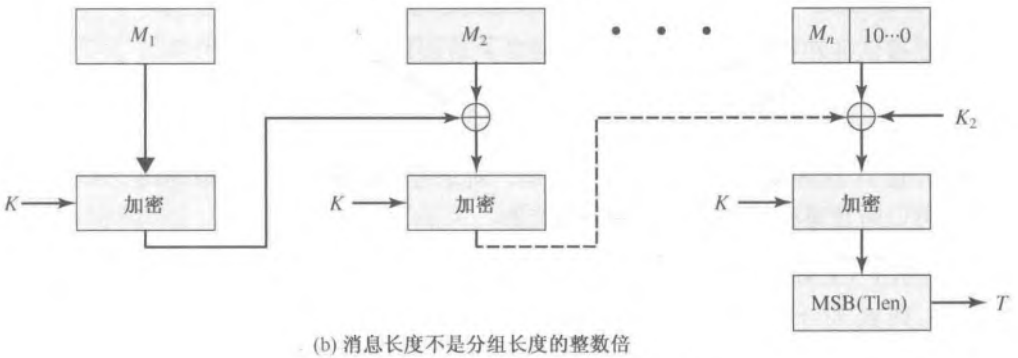
$$L = E(K, 0^b)$$
$$K_1 = L \cdot x$$
$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

其中，乘法（•）在域 GF(2^b)内进行，x 和 x^2 是域 GF(2^b)的 1 次和 2 次多项式。因此 x 的二元表示为 b–2 个 0，后跟 10，而 x^2 的二元表示是 b–3 个 0，后跟 100。有限域由不可约多项式定义，该多项式是那些具有极小非零项的多项式集合里按字典序排第一的那个多项式。对于已获批准的两个分组长度，多项式是 $x^{64}+x^4+x^3+x+1$ 以及 $x^{128}+x^7+x^2+x+1$。

为了生成 K_1 和 K_2，分组密码应用到一个全 0 分组上。第一个子密钥从所得密文导出，即先左移一位，并且根据条件和一个常数进行异或运算得到，其中常数依赖于分组的大小。第二个子密钥是采用相同的方式从第 1 个子密钥导出。形式为 GF(2^b)的有限域的这种性质在第 6 章讨论列混淆时已经解释过。



(a) 消息长度是分组长度的整数倍



(b) 消息长度不是分组长度的整数倍

图 12.8　基于密码的消息认证码（CMAC）

12.7　认证加密：CCM 和 GCM

认证加密（AE）是指在通信中同时提供保密性和认证（完整性）的加密系统。许多应用和协议中都同时需要这两种形式的安全性保证，但这两类安全系统一直都分开设计，直到近几年才合并在一起考虑。

对于消息 M 有四种同时提供认证和加密的通用方案：

- **先 Hash 再加密（H→E）**　对于 M 首先使用密码学 Hash 函数计算 $h = H(M)$，然后将消息和 Hash 值一起加密：E(K, (M||h))。

- **先认证再加密（A→E）** 使用两个密钥。首先通过计算 MAC 值 $T = \text{MAC}(K_1, M)$对明文进行认证，然后将消息和 MAC 一起加密：$\text{E}(K_2, [M\|T])$。SSL/TLS 协议（参见第 17 章）就是使用该方案的。
- **先加密再认证（E→A）** 使用两个密钥。首先加密消息得到密文 $C = \text{E}(K_2, M)$，然后通过计算 MAC 值 $T = \text{MAC}(K_1, C)$并考查（C, T)对明文进行认证。IPsec 协议（参见第 20 章）就是使用该方案的。
- **独立进行加密和认证（E+A）** 使用两个密钥。加密消息得到密文 $C = \text{E}(K_2, M)$；通过计算 MAC 值 $T = \text{MAC}(K_1, M)$并考查（C, T）对明文进行认证。这两个步骤的先后顺序可以交换。SSH 协议（参见第 17 章）就是使用该方案的。

对于每种方案，可直接进行解密和验证。对于 H→E、A→E 和 E+A，要先解密，后验证。对于 E→A，要先验证，后解密。这些方案都存在有安全缺陷。H→E 方案在无线加密协议（WEP）中用于保护 WiFi 网络，该方案存在根本性的缺陷以至于 WEP 协议被取代。文献[BLAC05]和文献[BELL00]指出了这三个加密/MAC 方案的安全性问题。当然，设计恰当的话，这些方案都能够提供高强度的安全性。这也是本节接下来将介绍的两个方案的目标，这两个方案都是 NIST 的标准。

12.7.1 分组密码链-消息认证码

分组密码链-消息认证码（CCM）工作模式由 NIST 作为标准提出，用于保护 IEEE 802.11 WiFi 无线局域网（参见第 18 章）的安全，当然也可用于任何需要认证和加密的网络应用。CCM 是 E+A 方案的改进，可提供认证加密。NIST SP800-38C 有该规范。

组成 CCM 的关键算法是 AES 加密算法（参见第 6 章）、CTR 工作模式（参见第 7 章）和 CMAC 认证算法（参见 12.6 节），在加密和 MAC 算法中共用一个密钥 K。CCM 加密过程的输入包括三部分：

（1）将要被认证和加密的数据，即明文消息 P 数据块。

（2）将要被认证但不需加密的相关数据 A，例如在协议进行时，协议头必须以明文传递，但需要认证保护。

（3）临时量 N 作为负载和相关数据的补充，对于每条消息在协议生命期内，N 的取值唯一，这可以防止重放攻击或其他相应的攻击。

图 12.9 展示了 CCM 的工作过程。对于认证，输入包括临时量、相关数据和明文。输入部分被格式化为从 B_0 到 B_r 的分组，第一块包括临时量以及存储 N、A 和 P 长度信息的位，后面跟着 0 个或者多个包含 A 的分组，然后是 0 个或者多个包含 P 的分组。整个分组作为 CMAC 算法的输入，产生长度为 Tlen 的 MAC 值，其中 Tlen 小于或等于分组长度［参见图 12.9(a)］。

对于加密，计数器 CTR 产生与临时量无关的序列。认证码使用计数器 Ctr_0 以 CTR 模式加密，输出的高 Tlen 位与 MAC 异或后产生加密后的 MAC。剩余的计数器用于以 CTR 模式加密明文（参见图 7.7）。加密后的明文和加密后的 MAC 一起形成密文输出［参见图 12.9(b)］。

SP800-38C 定义了如下认证/加密过程：

（1）使用格式函数将（N，A，P）格式化为分组 $B_0, B_1, \cdots, B_r$。

（2）令 $Y_0 = \text{E}(K, B_0)$。

（3）For $i = 1$ to r，do $Y_i = \text{E}(K, (B_i \oplus Y_{i-1}))$。

（4）令 $T = \text{MSB}_{\text{Tlen}}(Y_r)$。

（5）执行计数器生成算法，产生计数器分组 $\text{Ctr}_0, \text{Ctr}_1, \cdots, \text{Ctr}_m$，其中 $m = \lceil \text{Plen}/128 \rceil$。

（6）For $j=0$ to m，do $S_j=\mathrm{E}(K,\mathrm{Ctr}_j)$。

（7）令 $S=S_1\|S_2\|\cdots\|S_m$。

（8）返回 $C=(P\oplus\mathrm{MSB}_{\mathrm{Plen}}(S))\|(T\oplus\mathrm{MSB}_{\mathrm{Tlen}}(S_0))$。

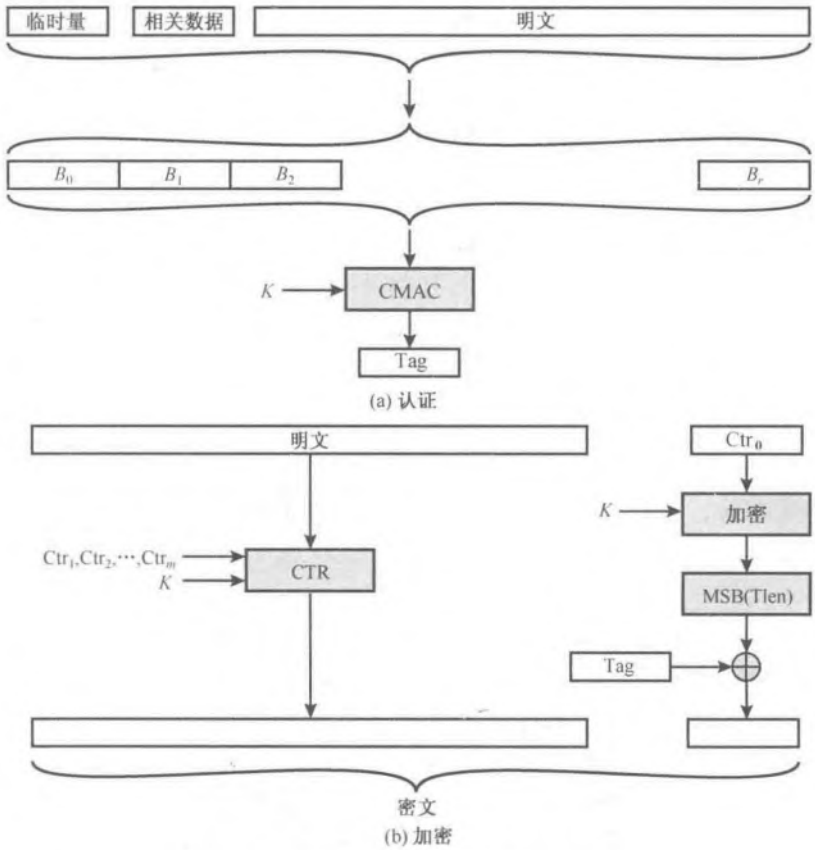


图 12.9　分组密码链-消息认证码（CCM）

对于解密和验证，接收方需要如下输入：密文 C，临时量 N，相关数据 A，密钥 K，以及初始计数器 Ctr_0。处理步骤如下：

（1）如果 Clen≤Tlen，则返回“无效”。

（2）执行计数器生成算法，产生计数器分组 $\mathrm{Ctr}_0,\mathrm{Ctr}_1,\cdots,\mathrm{Ctr}_m$，这里 $m=\lceil \mathrm{Clen}/128\rceil$

（3）For $j=0$ to m，do $S_j=\mathrm{E}(K,Ctr_j)$。

（4）令 $S=S_1\|S_2\|\cdots\|S_m$。

（5）令 $P=\mathrm{MSB}_{\mathrm{Clen\text{-}Tlen}}(C)\oplus\mathrm{MSB}_{\mathrm{Clen\text{-}Tlen}}(S)$。

（6）令 $T=\mathrm{LSB}_{\mathrm{Tlen}}(C)\oplus\mathrm{MSB}_{\mathrm{Tlen}}(S_0)$。

（7）使用格式函数将（N，A，P）格式化为分组 $B_0,B_1,\cdots,B_r$。

（8）令 $Y_0=\mathrm{E}(K,B_0)$。

（9）For $i=1$ to r，do $Y_i=\mathrm{E}(K,(B_i\oplus Y_{i-1})$。

（10）如果 $T\neq\mathrm{MSB}_{\mathrm{Tlen}}(Y_r)$，则返回“无效”，否则返回 P。

CCM 是一个相对复杂的算法。注意对于明文需要两次完全的处理：一次用来生成 MAC 值，一次用来加密。另外，该标准还要求临时量和 MAC 的长度进行折中选择，这项限制并不必要。同时

还要注意，加密密钥在 CTR 模式下使用了两遍：一次用来生成 MAC 值，一次用来加密明文和 MAC。这些增加的复杂操作是否增强了安全性还无定论。无论如何，关于算法的两个分析结果[JONS02].[ROGA03]表明 CCM 提供了高强度的安全性。

12.7.2 Galois/计数器模式

Galois/计数器（GCM）工作模式由 NIST 作为 NIST SP 800-38D 标准提出，基于并行化设计，因此可以提供高效的吞吐率和低成本、低延迟。其本质是消息在变型的 CTR 模式下加密，密文结果与密钥以及消息长度信息在 GF(2^{128})域上相乘。该标准同时还制定了仅支持 MAC 的工作模式，即 GMAC。

GCM 模式使用两个函数：带密钥的 Hash 函数 GHASH，以及计数器每次增 1 的 CTR 模式的 GCTR。

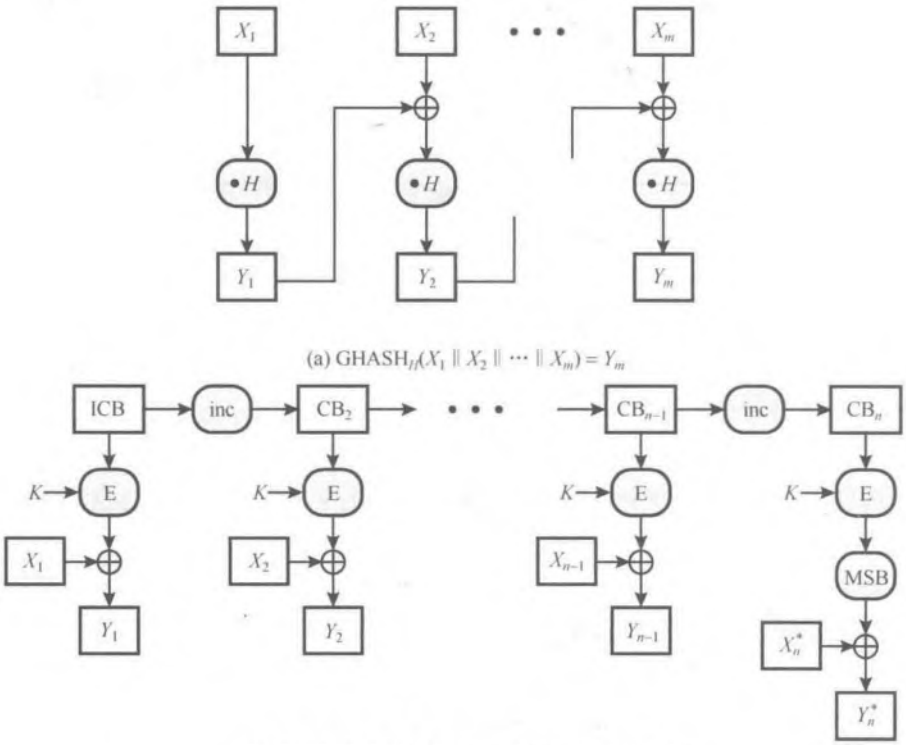
$\text{GHASH}_H(X)$将 Hash 密钥 H 和位串 X 作为输入，其中 len(X) = 128m 位，m 是正整数，输出是 128 位的 MAC 值。函数描述如下［参见图 12.10(a)］：

（1）令 $X_1, X_2, \cdots, X_{m-1}, X_m$ 表示输入序列，其中输入串 $X = X_1 \| X_2 \| \cdots \| X_{m-1} \| X_m$。

（2）令 Y_0 为 128 位的 0，表示为 0^{128}。

（3）For $i = 1, \cdots, m$，令 $Y_i = (Y_{i-1} \oplus X_i) \cdot H$。其中 · 表示 GF($2^{128}$)域上乘法。

（4）返回 Y_m。



(a) $\text{GHASH}_H(X_1 \| X_2 \| \cdots \| X_m) = Y_m$

(b) $\text{GCTR}_K(\text{ICB}, X_1 \| X_2 \| \cdots \| X_n^*) = Y_1 \| Y_2 \| \cdots Y_n^*$

图 12.10 GCM 认证和加密函数

$\text{GHASH}_H(X)$函数可表示为

$$(X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_{m-1} \cdot H^2) \oplus (X_m \cdot H)$$

上式非常适合快速实现。如果使用相同的 Hash 密钥认证多个消息，那么 H^2，H^3，…能够通过一次预计算来对所有消息进行认证操作，并且待认证的数据分组（$X_1, X_2, \cdots, X_m$）能够并行处理，因为每组运算都相互独立。

$\mathrm{GCTR}_K(\mathrm{ICB}, X)$将密钥 K 和任意长度的位串 X 作为输入，输出是长度与 X 相同的密文 Y。函数描述如下［参见图 12.10(b)］：

（1）如果 X 是空串，则返回空串 Y。

（2）令 $n=\lceil(\mathrm{len}(X)/128)\rceil$ 即 n 是大于等于 $\mathrm{len}(X)/128$ 的最小整数。

（3）令 $X_1, X_2, \cdots, X_{n-1}, X_n^*$表示输入序列，其中输入串

$$X=X_1\|X_2\|\cdots\|X_{n-1}\|X_n^*$$

其中，$X_1, X_2, \cdots, X_{n-1}$ 是 128 位完整分组。

（4）令 $\mathrm{CB}_1=\mathrm{ICB}$。

（5）For $I=2$ to n，令 $\mathrm{CB}_i=\mathrm{inc}_{32}(\mathrm{CB}_{i-1})$。其中 $\mathrm{inc}_{32}(S)$函数对于 S 的最右 32 位增 1 并取模 2^{32}，剩余位不变。

（6）For $I=1$ to $n-1$，do $Y_i=X_i\oplus\mathrm{E}(K,\mathrm{CB}_i)$。

（7）令 $Y_n^*=X_n^*\oplus\mathrm{MSB}_{\mathrm{len}(X_n^*)}(\mathrm{E}(K,\mathrm{CB}_n))$。

（8）令 $X=X_1\|X_2\|\cdots\|X_{n-1}\|Y_n^*$。

（9）返回 Y。

注意计数器值能够被快速生成，并且加密操作可以并行执行。

现在我们可以定义整个认证加密函数（参见图 12.11）。输入包括密钥 K，初始向量 IV，明文 P，以及附加认证信息 A。符号$[x]_s$表示非负整数 x 的二进制表示的第 s 位。函数步骤如下：

（1）令 $H=\mathrm{E}(\mathrm{K},0^{128})$。

（2）定义分组 J_0 如下：

如果 $\mathrm{len}(\mathrm{IV})=96$，则令 $J_0=\mathrm{IV}\|0^{31}\|1$。

如果$\mathrm{len}(\mathrm{IV})\neq 96$,则令 $s=128\lceil\mathrm{len}(\mathrm{IV})/128\rceil-\mathrm{len}(\mathrm{IV})$，并令 $J_0=\mathrm{GHASH}_H(\mathrm{IV}\|0^{s+64}\|[\mathrm{len}(\mathrm{IV})]_{64})$。

（3）令 $C=\mathrm{GCTR}_K(\mathrm{inc}_{32}(J_0),P)$。

（4）令 $u=128\lceil\mathrm{len}(C)/128\rceil-\mathrm{len}(C)$，并令 $v=128\lceil\mathrm{len}(A)/128\rceil-\mathrm{len}(A)$。

（5）定义分组 S 如下：

$$S=\mathrm{GHASH}_H(A\|0^v\|C\|0^u\|[\mathrm{len}(A)]_{64}\|[\mathrm{len}(C)]_{64})$$

（6）令 $T=\mathrm{MSB}_t(\mathrm{GCTR}_K(J_0,S))$，这里 t 是支持的 MAC 长度。

（7）返回（C,T）。

在第 1 步中，通过使用密钥 K 加密 128 位的全 0 产生 Hash 密钥。在第 2 步中，由 IV 预生成计数器分组（J_0），如果 IV 的长度是 96 位，则把 $0^{31}\|1$ 作为填充串附在 IV 后面生成 J_0，否则 IV 被最少的 0 填充，直到长度被填充为 128 位（分组）的整数倍，结果再填充 64 个 0，再附加 64 位的 IV 长度信息，GHASH 对整个填充后的串计算得出 J_0。

然后 GCM 基于 CTR 工作模式，仅需认证时的附加认证信息和消息加入 MAC。用于计算 Hash 的函数仅是 Galois 域上的乘法。之所以这样设计，是因为执行 Galois 域上的乘法运算很容易，便于硬件实现[MCGR05]。

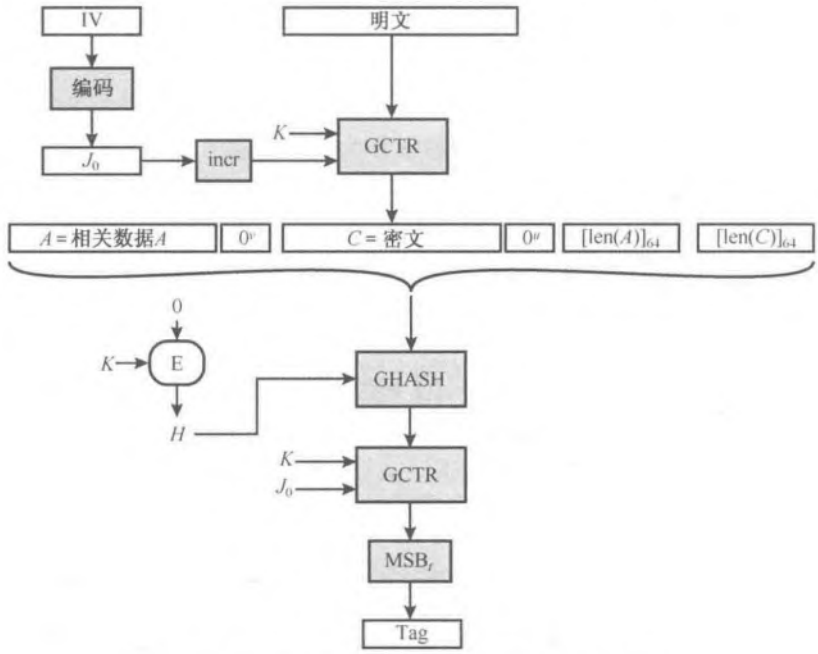


图 12.11 Galois/计数器-消息认证码（GCM）

参考文献[MCGR04]讨论了分组密码工作模式，结果显示基于 CTR 的认证加密方案在高速网络中是最高效的工作模式。该文还进一步说明了 GCM 满足高的安全性要求。

12.8 密钥封装

12.8.1 应用背景

密钥封装（KW）操作模式（SP 800-38F）是 NIST 制定的最新的一类分组密码工作模式。密钥封装模式使用 AES 或三重 DES 作为底层加密算法。在 RFC3394 中也有 AES 版本密钥封装模式的文档。

密钥封装的目的是为通信双方安全地交换对称密钥，而通信双方事先已共享有一组称为密钥加密密钥（KEK）的对称密钥。

这里有两个问题需要说明。第一，为什么需要使用一个通信双方已知的对称密钥去加密新的对称密钥？这类需求在本书介绍的许多协议中都是必要的，如 IEEE 802.11 和 IPsec 协议中的密钥管理部分。通常，一个协议需要调用不同层次的密钥，其中低级密钥使用频率高，为了抵抗攻击需要频繁更换。而高级密钥使用频率低，受到密码攻击的威胁相对较少。高级密钥通常用于加密新创建的低级密钥，以便在已经共享有高级密钥的通信双方之间进行低级密钥的交换。

第二个问题是，为什么需要新的工作模式？新工作模式的目的是为了用于密钥长度比分组长度要长的加密算法。如 AES 算法的分组长度是 128 位，但是密钥长度可以是 128 位，192 位或 256 位。对于后两种密钥长度，对密钥进行加密需要多个分组才能完成。注意密钥数据的重要性要比其他数据的重要性要高，因为密钥将被多次使用，密钥的泄露意味着所有使用该密钥进行加密的数据的泄露。因此 NIST 需要制定一个鲁棒的加密模式。密钥封装模式是鲁棒的，因为在密钥封装模式下，输出每一位的取值都无规律地受到输入每一位的取值影响。在前面介绍过的工作模式中许多模式都不满足上述要求。例如，对于前面所有的工作模式，明文的最后一个分组只影响密文的最后一个分组的取值；类似地，密文的第一个分组仅通过明文的第一个分组就可以被推导出来。

为了达到鲁棒性的要求，密钥封装模式的吞吐率与其他工作模式相比较低，在一些密钥管理应用中可能需要折中处理。另外，与消息加密或文件加密不同，密钥封装模式一般只用于对于少量数据的处理。

12.8.2　密钥封装算法

密钥封装算法对于 64 位分组进行操作。算法输入包括 64 位常数（该常数将在后续详述），以及被分为若干 64 位分组的明文密钥。我们使用如下符号：

$\mathrm{MSB}_{64}(W)$	W 的最高 64 位有效位
$\mathrm{LSB}_{64}(W)$	W 的最低 64 位有效位
W	临时变量；加密函数的输出
$\oplus$	按位异或 XOR
$\|\|$	字符串连接
K	密钥加密密钥
n	密钥数据的 64 位分组个数
s	封装过程的阶段数；$s=6n$
P_i	明文密钥数据的第 i 个分组；$1\leqslant i\leqslant n$
C_i	密文数据的第 i 个分组；$1\leqslant i\leqslant n$
$A(t)$	运行阶段 t 后的 64 位完整性检测寄存器；$1\leqslant t\leqslant s$
$A(0)$	初始化完整性检测寄存器（ICV）；十六进制表示：A6A6A6A6A6A6A6A6
$R(t, i)$	加密阶段 t 后的 64 位寄存器 i；$1\leqslant t\leqslant s$ ；$1\leqslant i\leqslant n$

密钥封装算法的描述如下：

输入　明文，n 个 64 位值（$P_1, P_2, \cdots, P_n$）
　　密钥加密密钥，K

输出　密文，（n+1）个 64 位值（$C_0, C_1, \cdots, C_n$）

（1）初始化变量

```
A(0)= A6A6A6A6A6A6A6A6
    for i =1 to n
      R(0, i)= pi
```

（2）计算中间值

```
for t = 1 to s
W = E(K, [A(t-1) ‖ R(t-1, 1)])
A(t) = t⊕ MSB64(W)
R(t, n) = LSB64(W)
for i = 1 to n-1
   R(t, i) = R(t-1, i+1)
```

（3）输出结果

```
C0 = A(s)
for i = 1 to n
   Ci = R(s, i)
```

注意，为了储存 ICV，密文比明文密钥要多一个分组长度。在密钥解封（解密）过程中，64 位

ICV 以及明文密钥都被恢复出来。如果解密出的 ICV 的值与初始输入值，即十六进制的 A6 A6 A6 A6 A6 A6 A6 A6 不一致，则检测出错误，解密出的明文密钥将被接收方拒绝。因此密钥封装算法不仅提供秘密性保护，还提供数据完整性保护。

图 12.12 演示了密钥封装算法对于 256 位密钥的加密过程。每个盒子表示一个加密阶段（t 的值）。注意 A 的输出反馈，作为下一阶段（t+1）的输入；而 R 的输出直接跨越 n 个阶段，至阶段（t+n），在本例中 n=4。这样的设计增强了雪崩效应和位之间的混淆。为了实现阶段的跨越，需要使用滑动缓冲，以便阶段 t 中 R 的输出能够在缓冲中每个阶段移一个位置，直到阶段 t+n 结束。如果我们把内层的 for 循环展开，则其结构更加清晰。例如对于本例中的 256 位密钥（n=4），其结构如下：

$$R(t,1) = R(t-1,2)$$
$$R(t,2) = R(t-1,3)$$
$$R(t,3) = R(t-1,4)$$

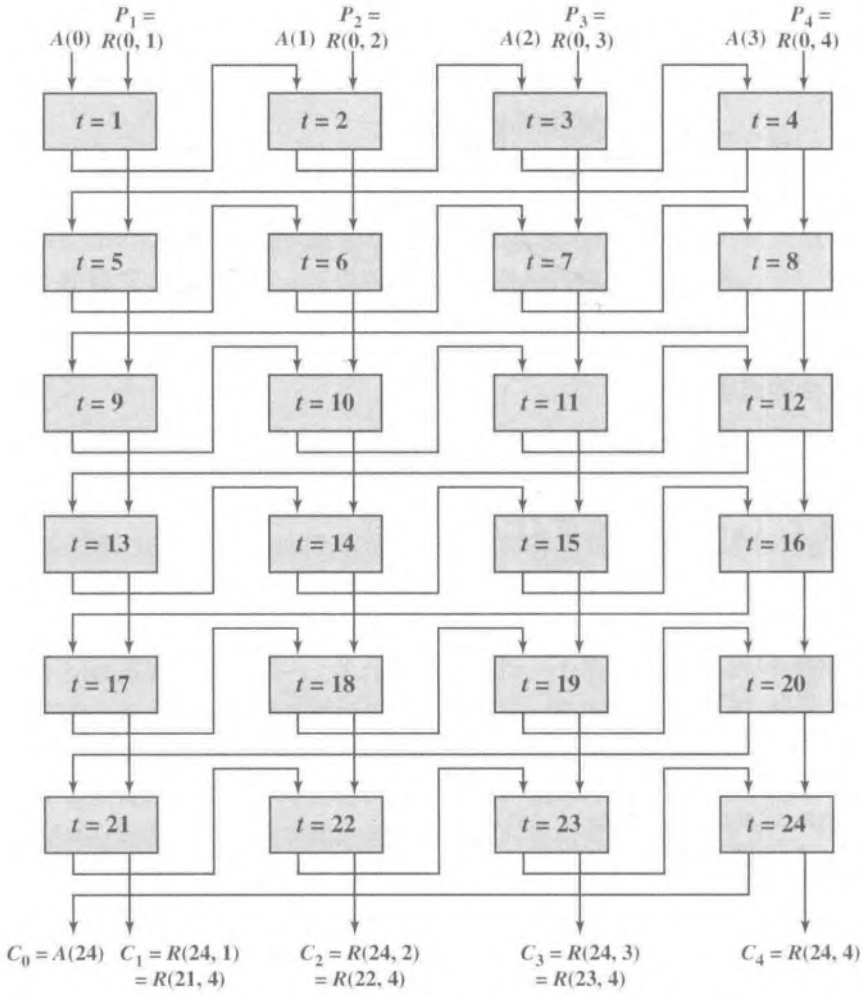


图 12.12　对 256 位密钥的密钥封装操作

例如，对于阶段 5，R 的输出为 $R(5,4) = x$。在阶段 6，计算 $R(6,3) = R(5,4) = x$。在阶段 7，计算 $R(7,2) = R(6,3) = x$。在阶段 8，计算 $R(8,1) = R(7,2) = x$。因此，在阶段 9，$R(t-1,1)$ 的输入是 $R(8,1) = x$。

图 12.13 描述了阶段 t 中对于 256 位密钥的操作。虚线部分的反馈表示对寄存器变量重新赋值。

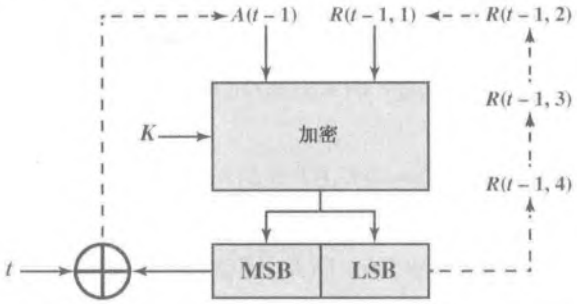


图 12.13　对 256 位密钥的密钥封装的阶段 t 的操作

12.8.3　密钥解封

密钥解封的算法定义如下：

输入　密文，（n+1）个 64 位值（$C_0, C_1, \cdots, C_n$）
　　密钥加密密钥，K

输出　明文，n 个 64 位值（$P_1, P_2, \cdots, P_n$），ICV

（1）初始化变量

```
A(s) = C0
for i = 1 to n
R(s, i) = Ci
```

（2）计算中间值

```
for t = s to 1
W = D(K, [(A(t)⊕ t)} R(t, n)])
A(t-1) = MSB64(W)
R(t-1, 1) = LSB64(W)
for i = 2 to n
   R(t-1, i) = R(t, i-1)
```

（3）输出结果

```
如果 A(0)=A6 A6 A6 A6 A6 A6 A6 A6
则
   for i =1 to n
      p(i) = R(0,i)
否则
   返回错误
```

注意，密钥解封算法使用解密函数。

接下来我们证明解封函数是封装函数的逆，即解封函数能够将明文密钥和 ICV 正确地恢复出来。首先，由于解封函数中的索引变量 t 的计数是从 s 到 1，解封函数中的阶段 t 恰好对应封装函数中的阶段 t。在封装函数的阶段 t 的输入变量索引序号为 t–1，解封函数的阶段 t 的输出变量索引序号为 t–1。因此，要想证明这两个函数互逆，只需要证明解封函数阶段 t 的输出变量等于封装函数阶段 t 的输入变量。

证明分为两个部分。首先我们证明 for 循环之前的 A 和 R 变量的计算是互逆的。为此简化符号，定义 128 位的数值 T 表示将 64 位的 t 后面补 64 个 0。因此密钥封装算法中步骤（2）的前 3 行可以

写成如下的单行形式：

$$A(t) \| R(t,n) = T \oplus \mathrm{E}(K,[A(t-1) \| R(t-1,1)]) \quad (12.1)$$

密钥解封算法中步骤（2）的前3行可以写成如下的单行形式：

$$A(t-1) \| R(t-1,1) = \mathrm{D}(K,([A(t) \| R(t,n)] \oplus T)) \quad (12.2)$$

将式（12.1）的右项展开，可得

$$\mathrm{D}(K,([A(t) \| R(t,n)] \oplus T)) = \mathrm{D}(K,([T \oplus E(K,[A(t-1) \| \ R(t-1,1)])] \ \oplus T))$$

由于$T \oplus T = 0$，对任意x有$x \oplus 0 = x$，所以

$$\begin{aligned} \mathrm{D}(K,([A(t) \| R(t,n)] \oplus T)) &= \mathrm{D}(K,([E(K,[A(t-1) \| R(t-1,1)])])) \\ &= A(t-1) \| R(t-1,1) \end{aligned}$$

第二部分证明的是密钥封装算法和解封算法步骤（2）的for循环中的操作是互逆的。对于密钥封装算法的阶段k，变量$R(t-1,1)$到$R(t-1,n)$是输入。$R(t-1,1)$用于加密计算；$R(t-1,2)$到$R(t-1,n)$映射到$R(t,1)$到$R(t,n-1)$；而$R(t,n)$是加密算法的输出。对于解封算法的阶段k，变量$R(t,1)$到$R(t,n)$是输入，$R(t,n)$是解密算法的输入并且解密输出结果是$R(t-1,1)$。剩余变量$R(t-1,2)$到$R(t-1,n)$是通过for循环计算得出的，与$R(t,1)$到$R(t,n-1)$一一对应。

因此，解封函数阶段k的输出变量等于封装函数阶段k的输入变量。

12.9 使用Hash函数和MAC的伪随机数发生器

所有伪随机数发生器（PRNG）的关键组成包括随机种子值和生成伪随机位流的确定性算法。如果算法使用伪随机函数（PRF）来产生诸如会话密钥之类的值，那么要求仅有PRF的使用者知道种子。如果该算法用于产生流密码的加密密钥流，那么种子的作用是仅为发送方和接收方知道的密钥。

我们在第8章和第10章中讨论过这个话题，是因为一个加密算法产生的输出是随机的，可以作为构造PRNG的基础。类似地，Hash函数和MAC的输出也是随机的，也能够用于构造PRNG。ISO标准18031（随机位生成）和NIST SP 800-90（使用确定性随机位生成器做随机数发生器的建议）都定义了使用密码学Hash函数进行随机数产生的方案。SP 800-90还定义了基于MAC的随机数发生器。我们接下来分别介绍这两种方案。

12.9.1 基于Hash函数的PRNG

图12.14(a)展示了SP 800-90和ISO 18031中基于Hash的PRNG的基本方案。算法步骤如下：

V为种子

seedlen为V的位长度，要求seedlen$\geq k+64$，其中k是需要的安全强度的长度。

N为需要的输出位数

算法使用产生outlen位长度Hash值的密码学Hash函数H。算法基本操作如下：

```
m = ⌈n / outlen⌉
data= V
W=空字符串
For i=1 to m
    w_i= H(data)
    W = ||w_i
    data=(data+1)mod 2^seedlen
返回W的左边n位
```

伪随机位流是 $w_1 \| w_2 \| \cdots \| w_m$，如果需要可将最后的块截断。

为了增加安全性，SP 800-90 标准还对 V 提供了周期性的更新。该标准还表示对于诸如 SHA-2 等强的密码学 Hash 算法，目前没有发现基于 Hash 方案的 PRNG 的已知或可疑缺陷。

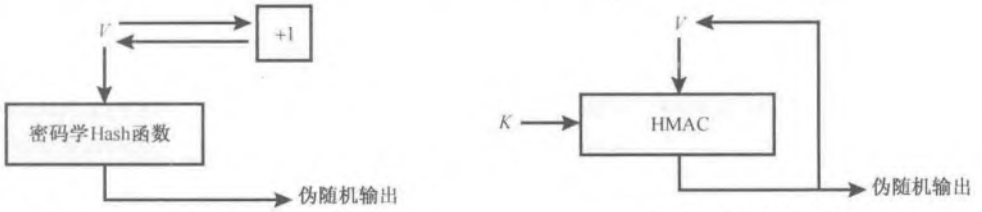


图 12.14　基于 Hash 的 PRNG 的基本结构（SP 800-90）

12.9.2　基于 MAC 的 PRNG

对于图 12.14(a)中描述的 PRNG，尽管没有发现基于密码学 Hash 算法的 PRNG 存在已知或可疑缺陷，但使用 MAC 可令我们更加放心。基于 MAC 的 PRNG 几乎都是使用 HMAC 构造的，这是因为 HMAC 是应用最广泛的标准 MAC 算法，在许多协议和应用中都已被实现。如 SP 800-90 中所述，与基于 Hash 的方案相比，其劣势是执行时间会增大一倍，这是因为 HMAC 对于每个输出块都要执行两次 Hash 函数运算。与基于 Hash 的方案相比，基于 HMAC 方案的优势是可以提供更高的安全性。

对于基于 MAC 的方案，需要两个输入：密钥 K 和种子 V。事实上，在 SP 800-90 中将 K 和 V 统称为种子。图 12.14(b)展示了 PRNG 机制的基本结构，图 12.15 的最左列说明了具体逻辑关系。注意，对于每个输出分组的密钥都是相同的，而每个分组的输入数据都等于前一分组的 MAC 值。为了增加安全性，SP 800-90 标准也提供了对 K 和 V 的周期性更新。

将一些应用中使用 HMAC 做 PRNG 的方法与 SP 800-90 标准相比较是有益的，如图 12.15 所示。对于 IEEE 802.11i 无线局域网安全标准（参见第 18 章），输入数据包括种子和计数器的连接，对于每个输出分组 w_i 计数器依次递增。该方案与 SP 800-90 标准相比，似乎安全性更高。对于 SP 800-90 标准，对于输出分组 w_i 的输入数据恰好是前一个 w_{i-1} 执行 HMAC 的输出。因此攻击者能够通过观察伪随机输出，就知道 HMAC 的输入和输出。尽管如此，假设 HMAC 是安全的，知道输入和输出并不能恢复出 K，也就不能猜测出接下来的随机位。

$m = \lceil n / \text{outlen} \rceil$ $w_0 = V$ $W =$ 空字符串 For i=1 to m $w_i = \text{MAC}(K, w_{i-1})$ $W = W \| w_i$ 返回 W 的左边 n 位	$m = \lceil n / \text{outlen} \rceil$ $W =$ 空字符串 For i=1 to m $w_i = \text{MAC}(K, (V \| i))$ $W = W \| w_i$ 返回 W 的左边 n 位	$m = \lceil n / \text{outlen} \rceil$ $A(0) = V$ $W =$ 空字符串 For i = 1 to m $A(i) = \text{MAC}(K, A(i-1))$ $w_i = \text{MAC}(K, (A(i) \| V))$ $W = W \| w_i$ 返回 W 的左边 n 位
NIST SP 800-90	**IEEE 802.11i**	**TLS/WTLS**

图 12.15　基于 HMAC 的三个 PRNG

传输层安全协议（第 17 章）和无线传输层安全协议（第 18 章）使用的方案包括每个输出分组 w_i 调用两次 HMAC。同 IEEE 802.11 一样，这样使得输出并不直接与输入相关。但双倍调用 HMAC 意味着操作负担也变为双倍，看起来有些过度了。

12.10 关键术语、思考题和习题

关键术语

认证符
分组密码链-消息认证码（CCM）
数据认证算法（DAA）
密钥封装工作模式
消息认证码（MAC）
基于密码的消息认证码（CMAC）
密码校验和
Galois/计数器模式（GCM）HMAC
密钥封装
CMAC
密码学 Hash 函数
密钥加密密钥
消息认证

思考题

12.1 消息认证是为了对付哪些类型的攻击?
12.2 消息认证或数字签名方法有哪两层功能?
12.3 产生消息认证有哪些方法?
12.4 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?
12.5 什么是消息认证码?
12.6 消息认证码和单向 Hash 函数之间的区别是什么?
12.7 为提供消息认证，应以何种方式保证 Hash 值的安全?
12.8 为了攻击 MAC 算法，必须要恢复密钥吗?
12.9 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?

习题

12.1 如果 F 是错误检测函数，那么无论是用做内部函数还是用做外部函数（参见图 12.2），它都具有错误检测能力。如果被传输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以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?

12.2 可以将 12.6 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，其初始向量为 0（参见图 12.7）。证明通过密文反馈方式可以得出同样的结果。

12.3 在 12.6 节的开头，当给定一个单分组消息 X 的 CBC MAC 值 $T = \text{MAC}(K, X)$ 时，攻击者立即就可以知道两分组消息 $X \| (X \oplus T)$ 的 CBC MAC 值，因为该值仍然是 T。请证明上述结论。

12.4 本习题我们证明对于 CMAC，当最后一轮加密完成后再和第二个密钥异或的变异方法没有效果。我们就消息长度是分组长度的整数倍的情况考虑这个问题。变异方法可以表示为 $\text{VMAC}(K, M) = \text{CBC}(K, M) \oplus K_1$。假设攻击者可以获得三条消息的 MAC：消息 $\mathbf{0} = 0^n$，其中 n 是分组的长度；消息 $\mathbf{1}=1^n$，以及消息 $\mathbf{1}\|\mathbf{0}$。结果，攻击者可以得到：$T_0 = \text{CBC}(K, \mathbf{0}) \oplus K_1$；$T_1 = \text{CBC}(K, \mathbf{1}) \oplus K_1$ 和 $T_2 = \text{CBC}(K, [\text{CBC}(K, \mathbf{1})]) \oplus K_1$。请证明攻击者可以计算出他没有提问过的消息 $\mathbf{0} \| (T_0 \oplus T_1)$ 的正确 MAC 值。

12.5 在讨论 CMAC 子密钥生成时，首先将分组密码应用到一个全 0 分组上。第一个子密钥从所得密文导出，即先左移一位，并且根据条件和一个常数进行异或运算得到，其中常数依赖于分组的大小。第二个子密钥是采用相同的方式从第一个子密钥导出的。

（a）对于长度为 64 位或 128 位的分组，需要何种常数?

（b）说明如何进行左移和异或运算以得到结果。

12.6　12.6 节给出了三种实现认证加密的基本途径：A→E、E→A 和 E+A。

（a）CCM 使用的是哪种途径？

（b）GCM 使用的是哪种途径？

12.7　给出 GHASH 函数的计算过程

$$(X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_{m-1} \cdot H^2) \oplus (X_m \cdot H)$$

12.8　对于认证解密过程给出类似图 12.11 的流程图。

12.9　Alice 想通过长度为 2 的字发送 1 位（单位）信息（是或否）给 Bob。Alice 和 Bob 有 4 种可能的密钥来进行认证。下表中给出了基于每个密钥所发送的 2 位字：

	消息	
密钥	0	1
1	00	01
2	10	00
3	01	11
4	11	10

（a）上表对于 Alice 的操作是必需的，请为 Bob 构造类似的表格用于认证。

（b）攻击者冒充 Alice 的成功概率是多少？

（c）攻击者对截获的消息进行替代的成功概率是多少？

12.10　对于密钥解封算法给出类似图 12.12 和图 12.13 的流程图。

12.11　对于如下的密钥封装算法：

（1）初始化变量

```
A = A6A6A6A6A6A6A6A6
for i = 1 to n
R(i) = P_i
```

（2）计算中间值

```
for j = 0 to 5
    for i = 1 to n
      B = E(K, [A ‖ R(i)])
      t = (n × j) + i
      A = t ⊕MSB_64(B)
      R(i) = LSB_64(B)
```

（3）输出结果

```
C_0 = A
for i = 1 to n
    C_i = R(i)
```

（a）该算法与 12.8 节中介绍的 SP 800-38F 算法，从功能上相比有什么区别？

（b）写出对应的密钥解封算法？

第13章 数字签名

学习目标

通过本章的学习应具有以下能力：

- 给出数字签名过程的概述
- 理解 Elgamal 数字签名方案
- 理解 Schnorr 数字签名方案
- 理解 NIST 数字签名方案
- 比较 NIST 数字签名方案和 Elgamal 数字签名方案以及 Schnorr 数字签名方案的异同
- 理解椭圆曲线数字签名方案
- 理解 RSA-PSS 数字签名方案

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。

图 13.1 是产生和使用数字签名过程的一般模型。本章介绍的所有数字签名方案都是基于此基本结构的。假设 Bob 想要发送消息给 Alice，该消息的秘密性不需要保证，但 Bob 想要 Alice 确认该消息确实是由 Bob 发出的。为了这个目标，Bob 使用安全 Hash 函数，例如 SHA-512，来产生该消息的 Hash 值。该 Hash 值同 Bob 的私钥一起作为数字签名产生算法的输入，算法的输出短块则作为数字签名。Bob 把该签名附在发送消息的后面，当 Alice 收到消息以及签名后，她（1）计算该消息的 Hash 值；（2）使用验证算法，输入 Hash 值以及 Bob 的公钥。如果算法返回结果表明签名是合法的，则 Alice 确认该消息是 Bob 发出的。由于其他任何人都没有 Bob 的私钥，也就意味着其他任何人都不能冒充一个签名能够使用 Bob 的公钥通过消息的验证。另外重要的一点是，不能够在不知道 Bob 的私钥的前提下篡改消息而不被发现，即消息在发送源和数据完整性两个方面都得到了认证。

本章先简要介绍数字签名的有关内容，然后介绍 Elgamal 数字签名方案以及 Schnorr 数字签名方案，理解这两个签名方案后很容易帮助我们理解数字签名算法（DSA）。本章接下来介绍其他两类已被制定为标准的数字签名方案：椭圆曲线数字签名算法（ECDSA）和 RSA 概率签名方案（RSA-PSS）。

13.1 数字签名概述

13.1.1 特征

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攻击，这种攻击可以有多种形式。

例如，假定 John 使用图 12.1 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，而这笔交易后来赔钱了，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。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。

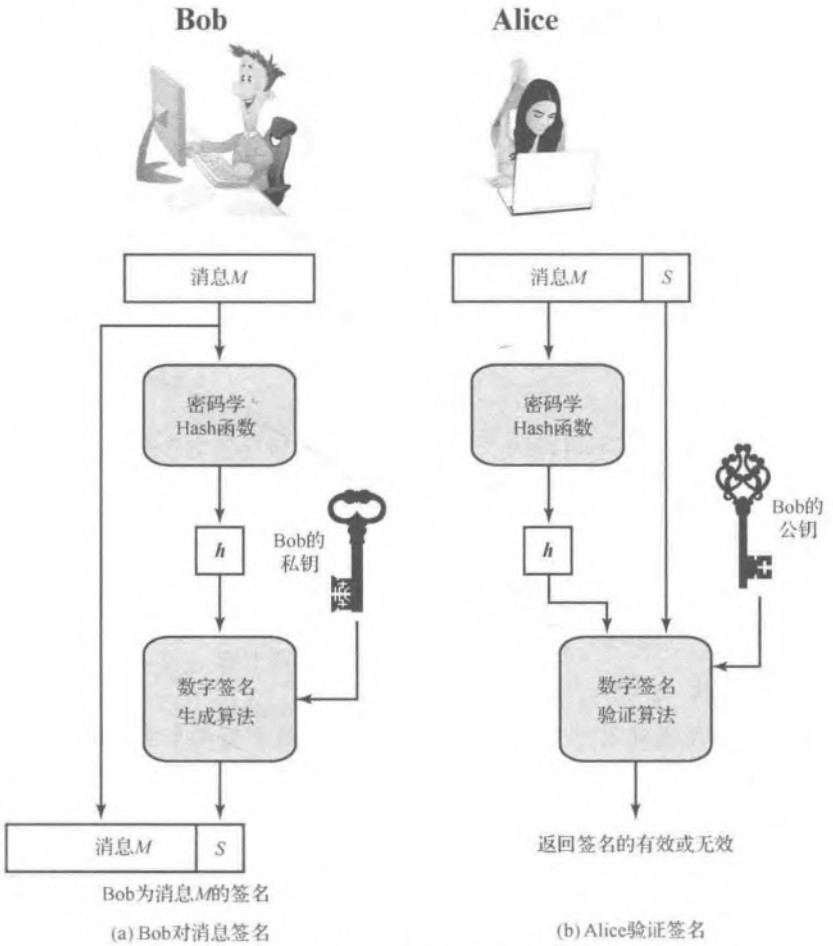


图 13.1　数字签名过程中关键部分的简单描述

13.1.2　攻击和伪造

文献[GOLD88]给出了危害程度从高到低排列的下列攻击类型。这里 A 代表其签名方案遭受攻击的用户，C 代表攻击者。

- **唯密钥攻击** C仅知道A的公钥。
- **已知消息攻击** C掌握一些消息以及对应的合法签名。
- **一般选择消息攻击** 在攻击A的签名方案之前，C首先选择一些消息，而无须知道A的公钥。然后C对于这些选择的消息从A处获得合法签名。该攻击是一般性的，因为与A的公钥无关，同样的攻击可用于其他用户。
- **定向选择消息攻击** 同一般选择消息攻击类似，不同的是攻击者选择消息的时间是在C掌握A的公钥之后，在签名生成之前。
- **适应性选择消息攻击** 允许C将A作为“oracle”进行轮询。这意味着A可以被要求对于特定的消息签名，而这些消息与C之前已经获得的<消息-签名>对相关。

文献[GOLD88]还定义了何谓成功的攻击签名方案，即C能够以一定的概率进行下列攻击：

- **完全破译** C判断出A的私钥。
- **通用伪造** C掌握一个有效的签名算法，使得对于任意消息都能够等价地构造出合法签名。
- **选择伪造** C对于所选特定消息能够伪造出合法签名。
- **存在性伪造** C至少可以伪造出一个消息的合法签名，但C不能控制该消息的选择。这种伪造对于A的危害最低。

13.1.3 数字签名需求

根据刚才讨论的基本特征和攻击，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。
- 保存数字签名的副本是可行的。

安全Hash函数在基于图13.1所示的方案中被使用，提供了满足上述条件的基础。然而具体的方案细节要仔细设计。

13.1.4 直接数字签名

直接数字签名指只涉及通信双方（发送方和接收方）的数字签名方案。假定接收方已知发送方的公钥。

用共享的密钥（对称密码）对整个消息和签名加密，则可以获得保密性。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。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管理和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。

被广泛接受的克服以上问题的方法是，使用数字证书的证书管理中心（CA）。我们将在第 14 章讨论这些话题，本章主要关注数字签名算法。

13.2　ElGamal 数字签名方案

在介绍 NIST 数字签名算法之前，首先介绍 ElGamal 和 Schnorr 签名方案。第 10 章中介绍的 ElGamal 加密方案能够使用用户的公钥进行加密，使用用户的私钥进行解密。ElGamal 数字签名方案则是使用私钥进行加密的，公钥进行解密的[ELGA84,ELGA85]。

在具体介绍之前，先复习数论的一个结论。回顾第 7 章，对于素数 q，如果 α 是 q 的原根，则有

$$\alpha,\alpha^2,\cdots,\alpha^{q-1}$$

取模（$\bmod q$）后各不相同。如果 α 是 q 的原根，则进一步有

（1）对于任意整数 m，$\alpha^m \equiv 1(\bmod q)$ 当且仅当 $m \equiv 0(\bmod q-1)$。

（2）对于任意整数 i，j，$\alpha^i \equiv \alpha^j(\bmod q)$ 当且仅当 $i \equiv j(\bmod q-1)$。

同 ElGamal 加密方案一样，ElGamal 数字签名方案的基本元素是素数 q 和 α，其中 α 是 q 的原根。用户 A 通过如下步骤产生公钥/私钥对：

（1）生成随机整数 X_A，使得 $1 < X_A < q-1$。

（2）计算 $Y_A = \alpha^{X_A} \bmod q$。

（3）A 的私钥是 X_A；A 的公钥是 $\{q,\alpha,Y_A\}$。

为了对消息 M 进行签名，用户 A 首先计算 Hash 值 $m = H(M)$，这里 m 是满足 $0 \leqslant m \leqslant q-1$ 的整数。然后 A 通过如下步骤产生数字签名：

（1）选择随机整数 K，使得满足 $1 \leqslant K \leqslant q-1$ 以及 $\gcd(K,q-1)=1$，即 K 与 $q-1$ 互素。

（2）计算 $S_1 = \alpha^K \bmod q$。注意，这同 ElGamal 加密方案中 C_1 的计算相同。

（3）计算 $K^{-1} \bmod (q-1)$，即计算 K 模 $q-1$ 的逆。

（4）计算 $S_2 = K^{-1}(m - X_A S_1) \bmod (q-1)$。

（5）签名包括 (S_1,S_2) 对。

任意用户 B 都能通过如下步骤验证签名：

（1）计算 $V_1 = \alpha^m \bmod q$。

（2）计算 $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$。

如果 $V_1 = V_2$ 则签名合法。下面我们给出证明：假设等式成立，那么有

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	假设 $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	代入 Y_A 和 S_1
$\alpha^{m-X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	等式左右移项
$m - X_A S_1 \equiv KS_2 \bmod (q-1)$	原根的性质

$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q-1)$ 代入 S_2

例如，对于素数域 GF(19)，即 $q=19$，如表 2.7 所示，其原根是{2,3,10,13,14,15}。我们选择 $\alpha=10$。Alice 通过如下步骤产生密钥对：

（1）Alice 选择 $X_A=16$。

（2）则 $Y_A=\alpha^{X_A} \bmod q=\alpha^{16} \bmod 19=4$。

（3）Alice 的私钥是 16；Alice 的公钥是 $\{q,\alpha,Y_A\}=\{19,10,4\}$。

假设 Alice 要对 Hash 值 $m=14$ 的消息进行签名：

（1）Alice 选择 $K=5$，其与 $q-1=18$ 互素。

（2）$S_1=\alpha^K \bmod q=10^5 \bmod 19=3$（参见表 2.7）

（3）$K^{-1} \bmod (q-1)=5^{-1} \bmod 18=11$。

（4）$S_2=K^{-1}(m-X_A S_1) \bmod (q-1)=11(14-(16)(3)) \bmod 18=-374 \bmod 18=4$。

Bob 能够通过如下步骤验证签名：

（1）$V_1=\alpha^m \bmod q=10^{14} \bmod 19=16$。

（2）$V_2=(Y_A)^{S_1}(S_1)^{S_2} \bmod q=(4^3)(3^4) \bmod 19=5184 \bmod 19=16$。

因为 $V_1=V_2$，所以签名是合法的。

13.3 Schnorr 数字签名方案

同 ElGamal 数字签名方案相同，Schnorr 数字签名方案也是基于离散对数的[SCHN89,SCHN91]。Schnorr 方案将生成签名所需的消息计算量最小化。生成签名的主要工作不依赖于消息，可以在处理器空闲时执行。生成签名过程与消息相关的部分需要进行 $2n$ 位长度的整数与 n 位长度的整数相乘。

该方案基于素数模 p，且 $p-1$ 包含大素数因子 q，即 $p-1\equiv 0(\bmod q)$。一般取 $p\approx 2^{1024}$ 和 $q\approx 2^{160}$，即 p 是 1024 位整数，q 是 160 位整数，也正好等于 SHA-1 中 Hash 值的长度。

该方案的第一部分是生成公钥/私钥对，包括以下步骤：

（1）选择素数 p 和 q，使得 q 是 $p-1$ 的素因子。

（2）选择整数 α，使得 $\alpha^q=1 \bmod p$。值 α、p 和 q 构成全局公钥参数，在用户组内的每个用户都可以取此值。

（3）选择随机整数 s，$0<s<q$，作为用户的私钥。

（4）计算 $v=\alpha^{-s} \bmod p$，作为用户的公钥。

对于私钥为 s、公钥为 v 的用户，通过如下步骤产生签名：

（1）选择随机整数 r，$0<r<q$，并计算 $x=\alpha^r \bmod p$。该过程与待签名消息 M 无关，可以在预处理过程计算。

（2）将 x 附在消息后面一起计算 Hash 值 e

$$e=H(M \| x)$$

（3）计算 $y=(r+se) \bmod q$。签名包括 (e,y) 对。

其他用户能够通过如下步骤验证签名：

（1）计算 $x' = \alpha^y v^e \bmod p$ 。

（2）验证是否 $e = H(M \| x')$ 。

对于该验证过程，有

$$x' \equiv \alpha^y v^e \equiv \alpha^y \alpha^{-se} \equiv \alpha^{y-se} \equiv \alpha^r \equiv x (\bmod p)$$

于是 $H(M \| x') = H(M \| x)$ 。

13.4　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，称为数字签名算法（DSA）。DSA 使用第 12 章中讨论的安全 Hash 算法（SHA）。DSA 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 FIPS 186-2，随后在 2009 年更新为 FIPS 186-3，在 2013 年更新为 FIPS 186-4。这个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 DSA 算法。

13.4.1　DSA 方法

DSA 使用的是只提供数字签名功能的算法。与 RSA 不同，DSA 虽然是一种公钥密码方案，但不能用于加密或密钥交换。

图 13.2 对用 DSA 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，用发送方的私钥将该 Hash 码加密形成签名，然后发送消息及其签名。接收方收到消息，计算 Hash 码。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，所以只有发送方能够产生有效的签名。

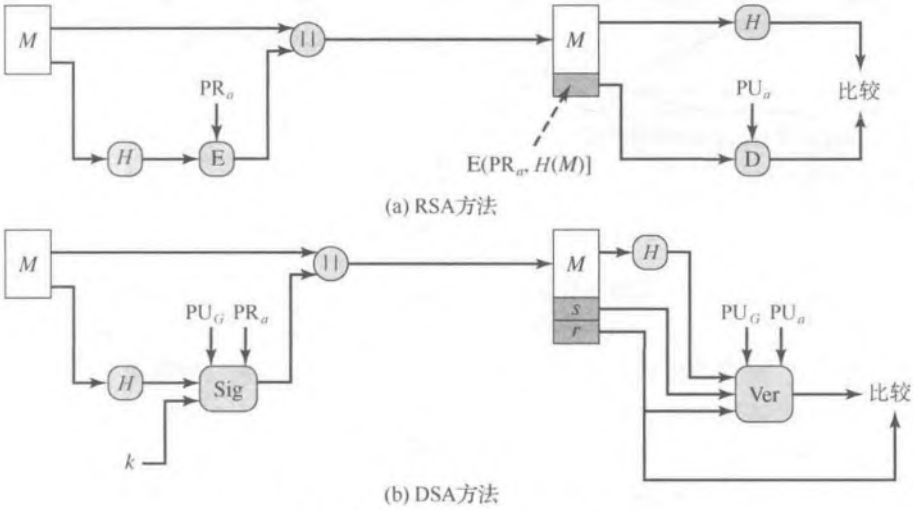


图 13.2　两种数字签名方法

DSA 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（PR_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（PU_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，验证函数依赖于全局公钥和发送方公钥（PU_a），该发送方公钥同发送方的私钥组成密钥对。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。

13.4.2 数字签名算法

DSA 是建立在求离散对数之困难性（参见第 2 章）以及 ElGamal[ELGA85]和 Schnorr[SCHN91]最初提出的方法之上的。

图 13.3 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 N 位的素数 q；然后选择一个长度在 512 ~ 1024 之间且满足 q 能整除（p–1）的素数 p；最后选择形为 $h^{(p-1)/q} \bmod p$ 的 g，其中 h 是 1 到 p–1 之间的整数使得 g 大于 1[②]。DSA 的公开参数的选择与 Schnorr 签名方案完全一样。

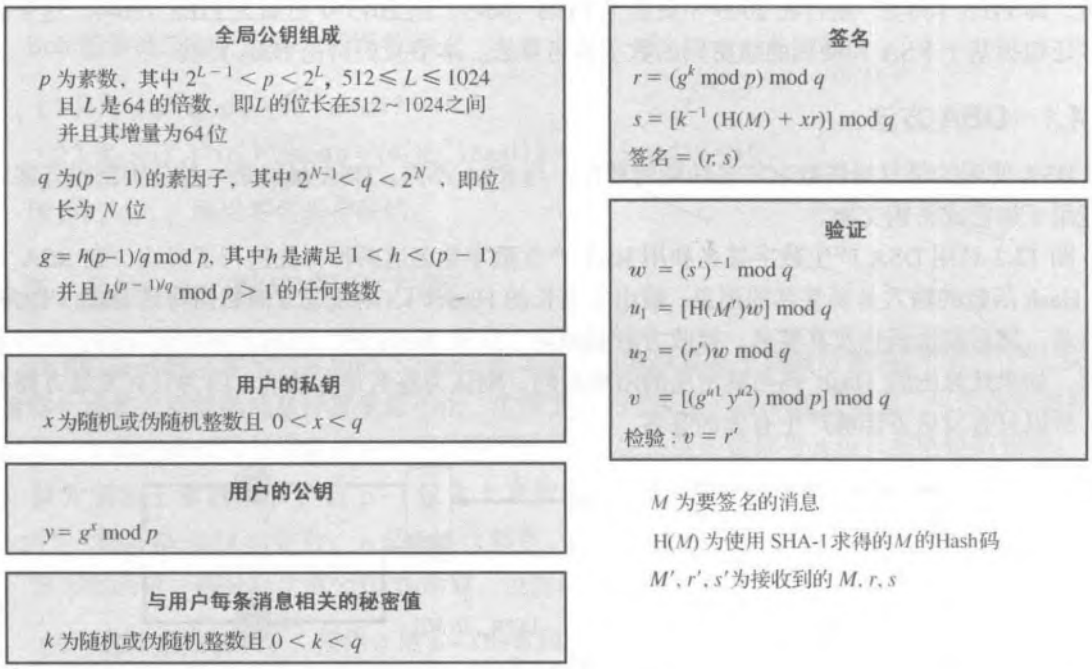


图 13.3 数字签名算法（DSA）

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、位于 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（参见第 2 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 对每次签名是唯一的。

令 M', r', s'为接收方收到的 M, r, s，接收方用图 13.3 所示的公式进行验证。接收方计算值 v，它

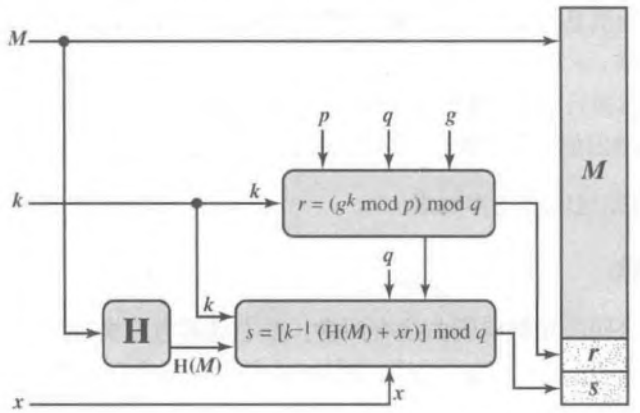
① 对不同的用户，这些附加参数可以不同，以便它们是用户公钥的一部分。在实际应用中，更可能的情形是将全局公钥与每个用户的公钥分开使用。

② 在数论术语中，g 是 q 模 p 的阶（参见第 2 章）。

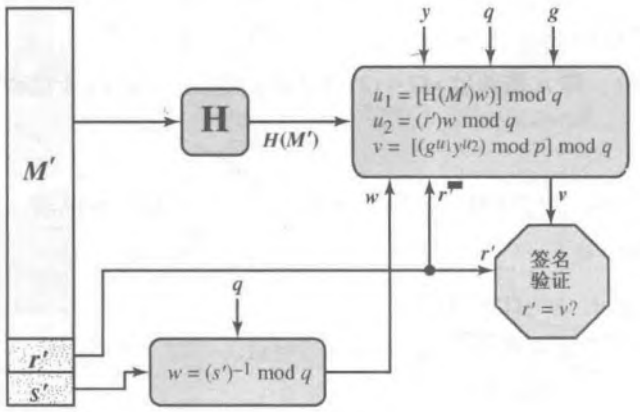
是公钥（p, q, g）、发送方公钥、接收到的消息的 Hash 码的函数。若 v 与签名中的 r 相同，则签名是有效的。

图 13.4 描述了 DSA 算法的签名和验证函数。

图 13.4 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、他的公钥以及全局公钥来恢复 r。从图 13.3 和图 13.4 不容易看出这种方法的正确性，本书附录 K 中给出了证明。



(a)签名



(b)验证

图 13.4　DSA 算法的签名和验证函数

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，产生签名对指数运算 $g^x \bmod p$ 的计算要求很高，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。其他要求比较高的任务是确定乘法逆元 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。

13.5　椭圆曲线数字签名算法

如前文所述，在 2009 年修订的 FIPS 186 加入了基于椭圆曲线密码的新数字签名方法，称其为

椭圆曲线数字签名算法（ECDSA）。由于椭圆曲线密码效率方面的优势，在需要使用短密钥长度的应用中，ECDSA 得到了日益增长的接受。

首先我们给出 ECDSA 处理过程的简要概述。本质上，包括四个基本元素。

（1）参与数字签名的所有通信方都使用相同的全局域参数，用于定义椭圆曲线以及曲线上的基点。

（2）签名者首先需要生成一对公钥、私钥。对于私钥，签名者选择一个随机数或者伪随机数作为签名者的私钥。使用随机数和基点，签名者计算出椭圆曲线上的另一解点，作为签名者的公钥。

（3）对于待签名的消息计算其 Hash 值。使用私钥、全局域参数、Hash 值，来产生签名。签名包括两个整数，r 和 s。

（4）如果要对签名进行验证，验证者使用签名者的公钥、全局域参数、整数 s 作为输入，并将计算得到的输出值 v 与收到的 r 进行比较。如果 $v=r$，则签名通过验证。

接下来我们依次介绍这四个基本元素。

13.5.1 全局域参数

第 10 章中可用于密码应用的椭圆曲线有两类：Z_p 上的素数域椭圆曲线和 $GF(2^m)$ 上的二元域椭圆曲线。对于 ECDSA 使用素数域椭圆曲线。ECDSA 的全局域参数如下：

q	一个素数
a，b	Z_q 上的整数，通过等式 $y^2 = x^3 + ax + b$ 定义椭圆曲线
G	满足椭圆曲线等式的基点，表示为 $G = (x_g, y_g)$
n	点 G 的阶，即 n 是满足 $nG = O$ 的最小正整数。这也等于曲线上点的个数

13.5.2 密钥产生

每个签名者需要产生一个密钥对，包括一个私钥和一个公钥。签名者，假设是 Bob，通过如下步骤产生上述两个密钥：

（1）选择随机整数 d，$d \in [1, n-1]$。

（2）计算 $Q = dG$。得到一个曲线 $E_q(a, b)$上的解点。

（3）Bob 的公钥是 Q，私钥是 d。

13.5.3 数字签名的产生与认证

通过使用公开的全局域参数和手中的私钥，Bob 对于消息 m 产生 320 字节数字签名的步骤如下：

（1）选择随机或伪随机整数 k，$k \in [1, n-1]$。

（2）计算曲线的解点 $P = (x, y) = kG$，以及 $r = x \bmod n$。如果 $r = 0$ 则跳至步骤（1）。

（3）计算 $t = k^{-1} \bmod n$。

（4）计算 $e = H(m)$，这里 H 是 Hash 函数 SHA-2，或 SHA-3。

（5）计算 $s = k^{-1}(e + dr) \bmod n$。如果 $s = O$，则跳至步骤（1）。

（6）消息 m 的签名是（r, s）对。

Alice 拥有公开的全局域参数以及 Bob 的公钥。Alice 收到 Bob 发送的消息以及数字签名后，对于签名的验证步骤如下：

（1）检验 r 和 s 是否为 1 到 n–1 之内的整数。

（2）使用 SHA，计算 160 位的 Hash 值 $e=\mathrm{H}(m)$。

（3）计算 $w=s^{-1} \bmod n$。

（4）计算 $u_1=ew$ 和 $u_2=rw$。

（5）计算解点 $X=(x_1, y_1)=u_1G+u_2Q$。

（6）如果 $X=O$，拒绝该签名；否则计算 $v=x_1 \bmod n$。

（7）当且仅当 $v=r$ 时，接受 Bob 的签名。

图 13.5 展示了签名的认证过程。我们可以给出该过程有效性的如下证明。如果 Alice 收到的消息确实是 Bob 签署的，那么

$$s=k^{-1}(e+dr) \bmod n$$

于是

$$k=s^{-1}(e+dr) \bmod n$$
$$k=(s^{-1}e+s^{-1}dr) \bmod n$$
$$k=(we+wdr) \bmod n$$
$$k=(u_1+u_2d) \bmod n$$

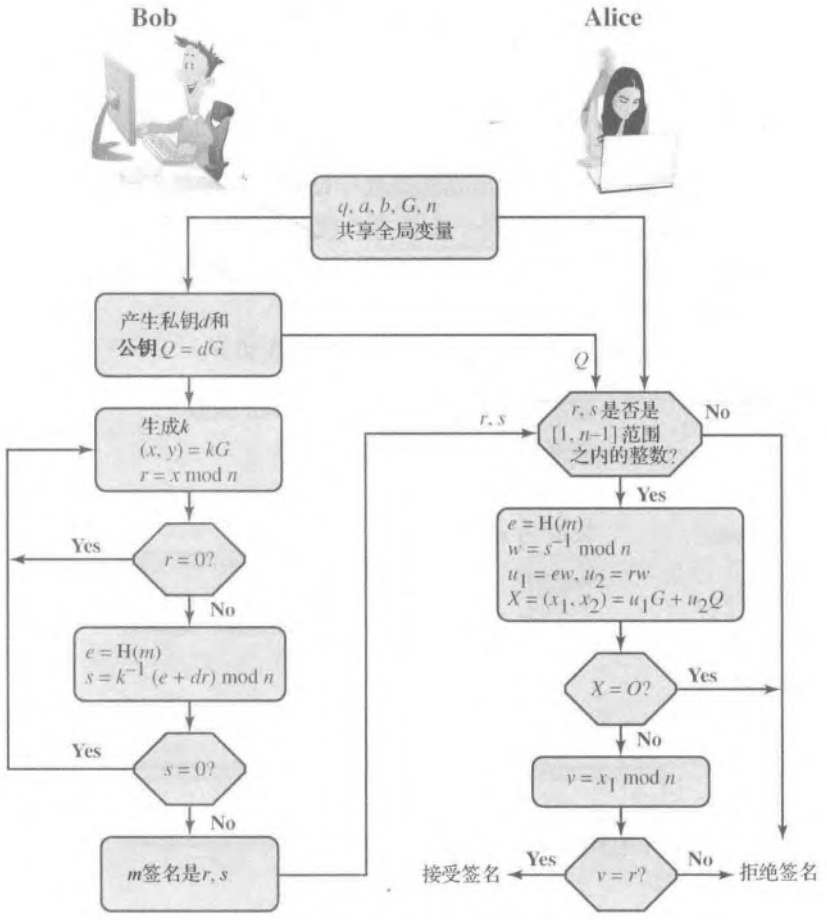


图 13.5　ECDSA 签名和验证

现在考虑

$$u_1G + u_2Q = u_1G + u_2dG = (u_1 + u_2d)G = kG$$

在验证过程的步骤（6）中，有 $v = x_1 \bmod n$，这里解点 $X = (x_1, y_1) = u_1G + u_2Q$。因为 $r = x \bmod n$ 且 x 是解点 kG 的 x 坐标，又因为我们已知 $u_1G + u_2Q = kG$，所以可得到 $v=r$。

13.6 RSA-PSS 数字签名算法

在 2009 年修订的 FIPS 186 中，除了 NIST 数字签名算法 DSA 和 ECDSA 之外，还包括几个由 RSA 实验室提出的、并且被广泛应用的基于 RSA 的签名方案。本书的网站中提供了使用 RSA 进行签名的范例。

本节我们将介绍 RSA 概率签名方案（RSA-PSS），该方案是基于 RSA 的最后一种方案，被 RSA 实验室推荐为 RSA 各类方案中最安全的一种。

因为基于 RSA 的方案广泛应用于包括金融在内的各类应用当中，所以证明这些方案的安全性非常引人关注。三类最主要的 RSA 签名方案的区别主要在于签名产生过程中将 Hash 值嵌入到消息中的填充格式，以及随后的签名验证过程中判断 Hash 值和消息的方法。对于在 PSS 之前提出的方案，目前为止都没有给出签名方案与 RSA 加解密方案安全性相等的数学证明[KALI01]。PSS 方案首先由 Bellare 和 Rogaway[BELL96c, BELL98]提出。与其他基于 RSA 的方案不同，该方案使用了随机化的处理过程，能够证明其安全性与 RSA 算法的安全性相关。这使得 RSA-PSS 在基于 RSA 的数字签名应用中更值得被选择。

13.6.1 掩码产生函数

在介绍 RSA-PSS 操作之前，需要先给出在构造块中使用的掩码产生函数（MGF）。MGF（X, maskLen）是一个伪随机函数，其输入参数是一个任意长度的位串以及需要的输出字节长度 L。MGF 通常基于安全 Hash 函数来构造，如 SHA-1。基于密码学 Hash 函数的 MGF 用于产生任意长度消息的摘要或者 Hash 值，输出固定长度的值。

目前 RSA-PSS 规范中使用的 MGF 函数是 MGF1，其参数如下：

选项	Hash	输出长度是 hLen 字节的 Hash 函数
输入	X	被掩码的字节串
	maskLen	掩码的字节长度
输出	mask	长度为 maskLen 字节的串

MGF1 的定义如下：

（1）初始化变量

```
T = 空字符串
k = ⌈maskLen / hLen⌉ - 1
```

（2）计算中间值

```
for counter = 0 to k
将 counter 表示为 32 位串 C 的形式
T = T‖Hash (X‖C)
```

（3）输出结果

```
mask = T 的前 maskLen 字节
```

本质上，MGF1 的操作过程如下。如果需要的输出长度等于 Hash 值的长度（maskLen=hLen），则将输入值 X 的 Hash 值连接取值为 0 的 32 位计数器，作为输出。如果 maskLen 比 hLen 大，MGF1 继续将当前处理的串 T 与计数器连接并对 X 计算 Hash，因此输出如下：

$$\text{Hash}(X \| 0) \| \text{Hash}(X \| 1) \| \cdots \| \text{Hash}(X \| k)$$

该过程直到 T 的长度大于或等于 maskLen 时结束，这时将 T 的前 maskLen 字节作为输出。

13.6.2　签名操作

消息编码　在对消息 M 生成 RSA-PSS 签名的第一阶段处理过程是由 M 生成固定长度的消息摘要，称为消息编码（EM）。图 13.6 展示了该过程。我们定义如下的参数和函数：

选项	Hash	输出长度是 hLen 字节的 Hash 函数。目前的选择是 SHA-1，该算法产生 20 字节的 Hash 值。
	MGF	掩码产生函数。目前规范中使用的是 MGF1。
	sLen	盐的字节长度。一般 sLen=hLen，当前版本为 20 字节。
输入	M	用于签名的待编码消息
	emBits	该数是比 RSA 模数 n 位长度小的值。
输出	EM	编码后的消息，是用于加密后形成消息签名的消息摘要。
参数	emLen	EM 的字节长度 $= \lceil \text{emBits}/8 \rceil$。
	填充 1	十六进制串 00 00 00 00 00 00 00 00；即 64 位的 0。
	填充 2	十六进制串若干个 00 后面跟着 01，其长度为（emLen-sLen-hLen-2）字节。
	盐（salt）	一组伪随机数。
	bc	BC 的十六进制值。

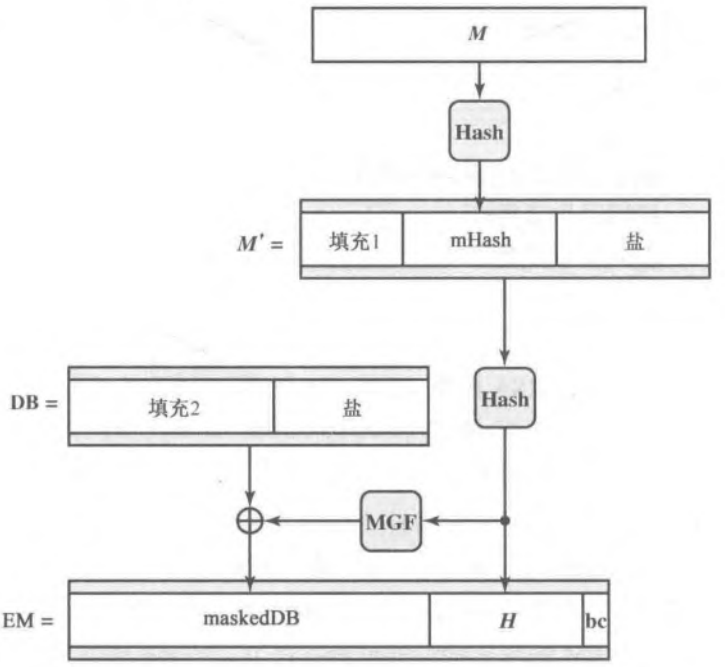


图 13.6　RSA-PSS 编码

编码过程包括如下步骤：

（1）生成消息 M 的 Hash 值：mHash = Hash(M)。

（2）生成伪随机字节串作为盐，得到 M' = padding_1||mHash||salt 的数据块。

（3）生成 M' 的 Hash 值：H = Hash(M')。

（4）构造数据块 DB = padding_2||salt。

（5）计算 H 的 MGF 值：dbMask = MGF(H, emLen-hLen-1)。

（6）计算 maskedDB = DB ⊕ dbMask。

（7）将 maskedDB 的最左字节中的最左 8emLen-emBits 位设为 0。

（8）EM = maskedDB||H||bc。

我们接下来对于该消息摘要算法给出几个地方的复杂度分析。在构造消息摘要的过程中，所有基于 RSA 的数字签名标准方案都使用了一个或多个消息扩展（如填充 1 和填充 2）。其目的是为了增大攻击者给定一个消息后找到另外一个消息映射到相同的摘要，或者寻找两个消息映射到相同摘要的难度。RSA-PSS 也使用了伪随机数，这里称为盐。因为在每次使用时盐的值都发生改变，所以使用相同的私钥对相同的消息进行两次签名，将得到两个不同的结果。这增加了签名方案的安全度。

形成签名 下面介绍拥有私钥{d, n}和公钥{e, n}的签名者如何形成签名（参见图 9.5）。将字节串 EM 作为无符号的非负二进制整数 m。通过如下方式加密 m 得到签名 s：

$$s = m^d \bmod n$$

令 k 是 RSA 模数 n 的字节长度。例如，RSA 的密钥长度是 2048 位，则 k=2048/8=256。然后将计算出的签名值 s 转换成 k 字节的串 S。

13.6.3 签名验证

解密 对于签名的验证，将签名 S 作为无符号的非负二进制整数 s。消息摘要 m 通过解密 s 得到

$$m = s^e \bmod n$$

然后，将消息转换为编码后的消息 EM 的形式，其长度为 emLen = $\lceil (\text{modBits} - 1)/8 \rceil$ 字节，这里 modBits 是 RSA 模数 n 的位长度。

EM 验证

选项	Hash	输出长度是 hLen 字节的 Hash 函数。
	MGF	掩码产生函数。
	sLen	盐的字节长度。
输入	M	用于验证的消息。
	EM	签名解密后的字节串，长度 emLen = $\lceil \text{emBits}/8 \rceil$。
	emBits	该数是比 RSA 模数 n 位长度小的值。
参数	填充 1	十六进制串 00 00 00 00 00 00 00 00；即 64 位的 0。
	填充 2	十六进制串若干个 00 后面跟着 01，其长度为（emLen-sLen-hLen-2）字节。

（1）生成 M 的 Hash 值：mHash = Hash(M)。

（2）如果 emLen<hLen+sLen+2，则输出“不一致”并停止。

（3）如果 EM 的最右字节不是十六进制值 BC，则输出“不一致”并停止。

（4）令 maskedDB 等于 EM 的最左 emLen-hLen-1 字节，H 为其接下来的 hLen 字节。

（5）如果 maskedDB 最左字节中的最左 8emLen-emBits 位不是全 0，则输出“不一致”并停止。

（6）计算 dbMask=MGF(H, emLen- hLen-1)。

（7）计算 DB = maskedDB ⊕ dbMask。

（8）设置 DB 的最左字节的最左 8emLen- emBits 位为 0。

（9）如果 DB 的最左（emLen-hLen-sLen -1）字节不等于填充 2，则输出“不一致”并停止。

（10）将 DB 的最后 sLen 字节设为盐值。

（11）构造 $M' = \text{padding}_1 \| \text{mHash} \| \text{salt}$ 的数据块。

（12）生成 M' 的 Hash 值：$H' = \text{Hash}(M')$。

（13）如果 $H = H'$，输出“一致”。否则，输出“不一致”。

图 13.7 展示了整个验证过程。阴影部分的方块标记为 H 和 H'，分别对应着解密后的签名中的值和直接从签名对应的消息中生成的值。剩下的三处阴影部分包括解密后的签名的值中与已知常数的比较部分。我们现在能够更清晰地理解签名者生成的 EM 中嵌入的常数和伪随机盐值的不同作用。验证者知道常数，因此计算出来的常数可以跟已知的常数值相比较，作为对于签名有效性的额外（除了检查 H 和 H' 之外的）检查。盐值使得给定相同的私钥对同一消息会产生不同的签名。验证者不知道盐值，不需要去进行盐值的比较。盐值起到的作用类似于 NIST DSA 和 ECDSA 中伪随机变量 k 的作用。在上述方案中，k 是签名者产生的伪随机数，使得对于同一消息使用相同的私钥进行多次签名时，能够产生不同的签名结果。验证者不知道也不需要知道 k 的具体值。

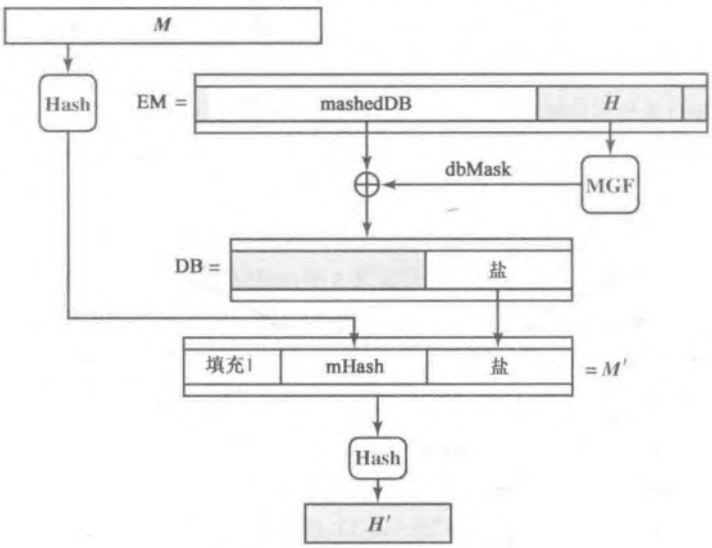


图 13.7　RSA-PSS EM 验证

13.7　关键术语、思考题和习题

关键术语

数字签名	数字签名算法（DSA）	直接数字签名	ElGamal 数字签名
椭圆曲线数字签名算法（ECDSA）	Schnorr 数字签名	时间戳	

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？

习题

13.1 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，特别是，对某些文件需要两个签名者来签名，使得

（1）第一签名者形成文件，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该文件确实是由两个签名者签过的文件，但只有第二签名者可验证步骤 1 中的签名。即接受方（或公众）仅可以验证具有两个签名的完整文件，而不是只有一个签名的中间文件。

而且，银行希望利用现有的支持 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）的唯一的签名者产生的。对这个过程稍加推广，就可解决 Hosgrave 先生的问题，也就是说……"

请完成上述叙述。

13.2 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？

13.3 如果用于产生 DSA 签名的 k 已被泄密，则会出现什么问题？

13.4 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 通过测试，可能是素数，转步骤（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，否则转步骤（2）。

（a）说明该算法的工作原理。

（b）证明该算法与第 2 章中给出的 Miller-Rabin 测试是等价的。

13.5 因为 DSA 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？

13.6　考虑 DSA 参数定义域生成问题。假设我们已经找到了参数 p 和 q，使得 $q|(p-1)$。现在我们需要寻找参数 $g \in Z_p$，使得 g 模 p 的阶为 q。考虑如下两种算法：

算法 1	算法 2
重复 　选择 $g \in Z_p$ 　$h \leftarrow g^q \bmod p$ 直至（$h=1$ 且 $g \neq 1$） 返回 g	重复 　选择 $h \in Z_p$ 　$g \leftarrow h^{(p-1)/p} \bmod p$ 直至（$g \neq 1$） 返回 g

（a）证明算法 1 返回的参数值的阶为 q。

（b）证明算法 2 返回的参数值的阶为 q。

（c）假设 $p = 40193$，$q=157$。算法 1 需要经过多少轮循环才可以找到一个生成元？

（d）如果 p 为 1024 位长，q 为 160 位长，你愿意推荐使用算法 1 来寻找 g 吗？为什么？

（e）假设 $p = 40193$，$q = 157$。算法 2 首次循环找到生成元的概率有多大（如果有用，你可以使用 $\sum_{d|n} \varphi(d) = n$ 这一事实来回答该问题）？

13.7　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 $h = H(M)$。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 $Z \times h \equiv X \pmod{q-1}$ 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）给出一种简单的方法可以对任意消息伪造用户签名，以证明这种体制是不可接受的。

13.8　运用对称密码实现数字签名的早期方案基于如下叙述的内容：为了对 n 位的消息签名，发送者事先随机产生 $2n$ 个 56 位的密钥

$$k_1,\ K_1,\ k_2,\ K_2,\ \cdots,\ k_n,\ K_n$$

这些参数是保密的。发送者再相应地事先准备好两套 64 位的验证参数

$$u_1,\ U_1,\ u_2,\ U_2,\ \cdots,\ u_n,\ U_n \text{ 和 } v_1,\ V_1,\ v_2,\ V_2,\ \cdots,\ v_n,\ V_n$$

这两套参数公开，其中

$$v_i = \mathrm{E}(k_i, u_i),\ V_i = \mathrm{E}(k_i, U_i)$$

消息 M 按如下方式签名。对于消息的第 i 位，依赖该消息位的值为 0 或 1，将 k_i 或 K_i 附在消息上。例如，如果消息的前三位为 011，则签名的前三个密钥为 k_1, K_2, K_3。

（a）接受方如何验证消息？

（b）该方法安全吗？

（c）一套保密密钥可以安全地为不同的消息签名多少次？

（d）该方案会带来何种实际问题（如果存在的话）？

第五部分　互　　信

第 14 章　密钥管理和分发
第 15 章　用户认证

第 14 章　密钥管理和分发

学 习 目 标

通过本章的学习应具备以下能力：

- 论述密钥分层的概念
- 理解利用非对称加密分配对称密钥的技术问题
- 概述公钥分配的方法，并分析其中的风险
- 列举并解释 X.509 证书中的主要组成元素
- 概述公钥基础设施的概念

密钥管理和密钥分发比较复杂，包括对密码、协议和管理的考虑。本章的目的是让读者对涉及的相关问题以及对密钥管理和分发的各个方面有一个大致的认识。若想了解更多的信息，请参阅 NIST SP 800-57 第 3 卷以及本章结束时推荐的阅读列表。

14.1　基于对称加密的对称密钥分发

对于对称加密来说，通信双方必须使用相同的密钥并且该密钥要对其他人保密。在限制攻击者攻陷密钥所需要的数据总数时，频繁的密钥交换是安全的。因此，任何密码系统的强度取决于密钥分发技术，即在想要交换数据的两者之间传递密钥且不被其他人知道的方法。对 A 和 B 来说，密钥分发能通过以下不同的方式实现：

（1）A 选择一个密钥后以物理的方式传递给 B。

（2）第三方选择密钥后物理地传递给 A 和 B。

（3）如果 A 和 B 先前或者最近使用过一个密钥，则一方可以将新密钥用旧密钥加密后发送给另一方。

（4）如果 A 和 B 到第三方 C 有加密连接，C 可以在加密连接上传送密钥给 A 和 B。

其中第（1）项和第（2）项需要手动交付一个密钥，对于链接加密来说是必需的，因为每个链接加密设备都只能与链接终端的用户交换数据。然而，手动交付对于网络中的端到端加密是不实用的。在分布式系统中，任何给定的主机或者终端可能需要同时与很多其他主机或终端交换数据，因此每个设备都需要动态提供大量的密钥。这个问题在大范围的分布式系统中更加明显。

这个问题的规模依赖于需支持的通信对的数目。如果端到端加密在网络中或者 IP 层执行，那么网络中每一对想要通信的主机都需要一个密钥，即若有 N 台主机则需要的密钥数目为 $[N(N-1)]/2$。如果加密在应用层执行的话，每一对需要通信的用户或者进程都需要一个密钥，然而一个网络可能有上百台主机，但却有上千个用户和进程。图 14.1 中举例说明了端到端加密的密钥分发任务的量级[①]。一个基于节点的网络有 1000 个节点就需要分发大概 50 万个密钥，如果相同的网络支持 10 000 个应用，则在应用层加密就需要 5000 万个密钥。

① 注意，该图使用 log-log 规模，于是线性图标代表指数增长。在 WilliamStallings.com/StudentSupport.html 的计算机科学学生资源网站中，对对数规模的一个基本的概述是数学中复习进修的文件。

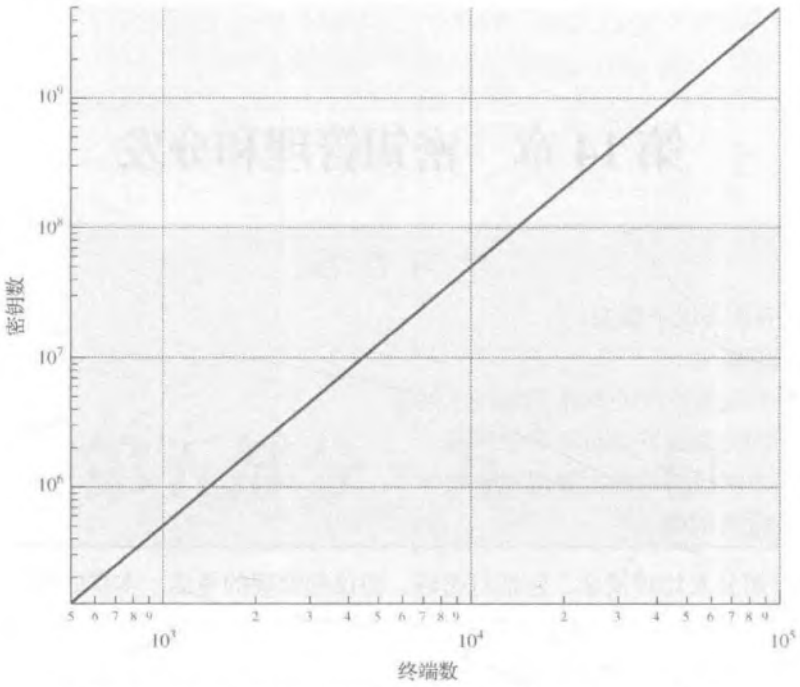


图 14.1 支持终点之间的任意连接所需要的密钥（key）数量

如第（3）项列出的方法可用于链接加密或者端到端加密，如果攻击者成功地获得一个密钥则随后的密钥都会泄露，那么，潜在的数百万密钥就必须进行重新分发。

对端到端的加密，很多关于第（4）项的变体已被广泛采用。这种策略中，负责为用户（主机、进程或者应用）分发密钥的密钥分发中心是必需的，且为了密钥的分发，每个用户都需要和密钥分发中心共享唯一的密钥。

密钥分发中心是基于密钥层次体系的，最少需要两个密钥层（参见图 14.2）。两个终端系统之间的通信使用临时密钥（通常指的是会话密钥）加密。会话密钥往往被用于逻辑连接中，如帧的转发或传输连接，然后随着连接的断开而丢弃。终端用户通信所使用的会话密钥从密钥分发中心得到，因此，会话密钥可以用密钥分发中心与终端系统或者用户共有的主密钥加密后进行传送。

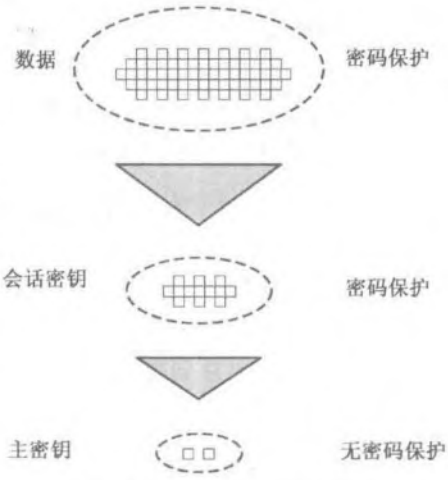


图 14.2 密钥层次的使用

每一个终端系统或者用户和密钥分发中心共有唯一的主密钥。如果有 N 个实体想要逐对通信，那么每次通信需要大概 $[N(N-1)]/2$ 个会话密钥，然而，这里只需要每个实体拥有一个主密钥，即共 N 个。因此，主密钥的分发可以通过一些不加密的方式完成，如物理传递。

14.1.1 密钥分发方案

密钥分发可以用不同的方式实现，一种典型的方案如图 14.3 所示[POPE79]。该方案假设每个用户和密钥分发中心（KDC）共享唯一的密钥。

假设用户 A 想要和 B 建立以逻辑连接，并且要求一次性的会话密钥来保护连接上传输的数据。A 有主密钥 K_a（只有它自己和 KDC 知道），B 有主密钥 K_b（和 KDC 共享），步骤如下：

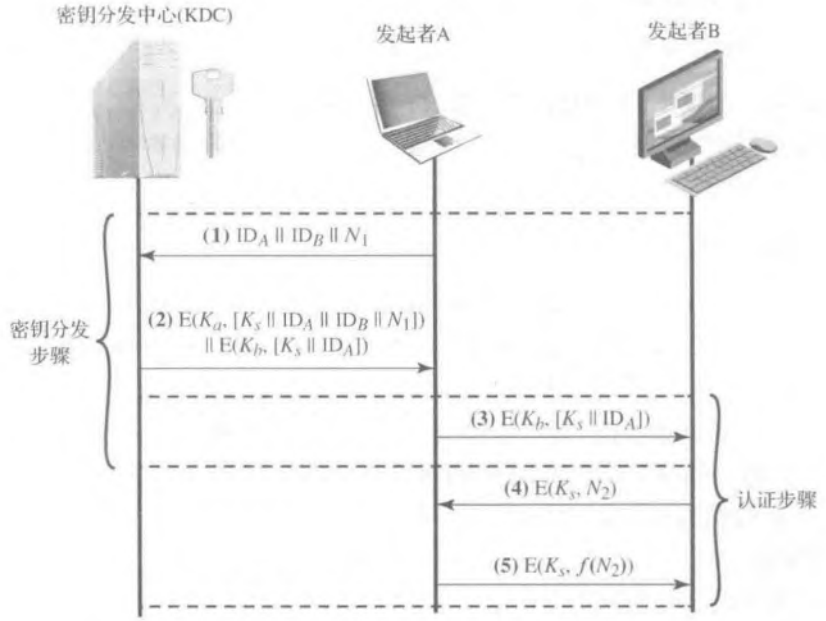


图 14.3　密钥分发方案

（1）A 向 KDC 发送包请求一个会话密钥来保护到 B 的逻辑连接的安全，其中包括 A、B 的身份以及该次传输的唯一标识 N_1，称为临时交互号（nonce）。这个临时交互号可以是一个时间戳、计数器或者一个随机数。最低要求是每次请求的临时交互号是不同的，而且为了防止伪造还要求敌手猜出该临时交互号是困难的，于是，随机数作为临时交互号是一个很好的选择。

（2）KDC 返回的信息是用 K_a 加密的，因此只有 A 能成功的读取该信息并且 A 知道是由 KDC 发来的。该信息中包含 A 想获取的两个部分：

- 用于会话的一次性会话密钥 K_s。
- 之前的请求信息，包括临时交互号，该信息使得 A 能够将这些返回信息和之前的请求相比较。

因此，A 可以知道它的原始信息在 KDC 收到之前是否被更改过，并且因为临时交互号的原因，可以知道先前的请求是否被重放。

另外，该信息中也包含 B 想获取的两个部分：

- 用于会话的一次性会话密钥 K_s。
- A 的标识（如 A 的网络地址）ID_A。

后面这两个部分由 K_b 加密，发送给 B，以达到建立连接并检验 A 的身份。

（3）A 存储将要使用的会话密钥，把来源于 KDC 的信息发送给 B，即 $E(K_b,[K_s \| ID_A])$。因为这个消息是用 K_b 加密的，所以可以防止窃听。B 现在知道会话密钥为 K_s、想建立连接的另一方是 A（ID_A）及该信息是由 KDC 加密的（因为是使用 K_b 加密的）。

此时，会话密钥已经安全的在 A 和 B 之间建立并且 A、B 可以使用其来保护通信，然而，还有另外两步要满足：

（4）B 使用新的会话密钥 K_s 加密临时交互号 N_2 并将结果发送给 A。

（5）同样，A 使用 K_s 加密 $f(N_2)$ 后发给 B，其中 f 是一个执行 N_2 转换的函数。

这一步能保证 B 在步骤（3）中收到的信息没有受到重放攻击。

注意：实际的密钥分发方案只包括步骤（1）至步骤（3），步骤（3）至步骤（5）执行认证功能。

14.1.2 层次密钥控制

对于单个的 KDC 没有必要限制其密钥分发功能，但是，对大型网络来说这样做是不实用的。为了改变这种状况，出现了 KDC 的层次体系，例如有本地 KDC，每一个本地 KDC 负责其所在的整个内部网络的一个域，如一个局域网（LAN）或者一栋建筑物。由于同一个本地域中的各个实体要相互通信，故由本地 KDC 负责密钥分发。如果两个实体在不同的域，则需要一个共享密钥，此共享密钥由两个相对应的本地 KDC 通过全局 KDC 协商产生，这种情况下，这三个 KDC 中的任意一个都可以选择密钥。层次的概念可以依据用户的规模以及内部网络的地理范围扩展到三个或者更多的层。

层次策略使得主密钥分发的开销最小化，因为大部分的主密钥是由本地 KDC 及其实体所共享的，而且，该策略将有缺陷的或者被破坏的 KDC 的危害限制在本地域中。

14.1.3 会话密钥生命周期

会话密钥交换得越频繁就越安全，因为对于任何给定的会话密钥，攻击者拥有的密文会比较少。另一方面，会话密钥分发会延迟交换的开始时间，增加网络负担。安全管理者在决定特定会话密钥的生命周期时，必须平衡这些竞争代价。

对于面向连接的协议，在会话的整个生命周期中使用同一个会话密钥，为每一次新的会话使用新的会话密钥。如果一个逻辑连接的生命周期很长，则它需要周期性的改变会话密钥，协议数据单元（PDU）的序列号随之重置。

对一个无连接协议，如面向事务（transaction）的协议，没有明确的连接初始和终止，因此不知道需要多长时间来更换一次密钥，最安全的方法是每次都使用新的会话密钥。然而，这也否定了无连接协议最主要的一个优点，即最小化每次执行的开销和延时。一个比较好的策略是为特定时期或者特定数量的事务分配不同的会话密钥。

14.1.4 透明的密钥控制策略

图 14.3 提出的方法有很多变体，这里将会描述其中一种（参见图 14.4），这种策略以一种对终端用户透明的方式，对网络或者传输层提供端到端的加密。该方法假定通信使用面向连接的端到端的协议如 TCP，该方法设定会话安全单元（SSM），SSM 包含协议层，执行端到端加密的功能以及代表主机、终端获取会话密钥的功能。

建立连接的步骤如图 14.4 所示，当一台主机想要与另一台主机建立连接时，会发送一个连接建立请求包（步骤 1）；SSM 缓冲请求包后向 KDC 申请建立连接（步骤 2），其中 SSM 和 KDC 之间的通信是由两者共享的主密钥加密的；如果 KDC 同意建立连接，则产生会话密钥并分发给 SSM，分发的过程中使用 SSM 的主密钥加密（步骤 3）；SSM 传输缓冲的请求包，连接将会在两个终端系统之间建立（步骤 4）。两个终端系统间所有用户数据的交互都需要各自的 SSM 使用会话密钥加密。

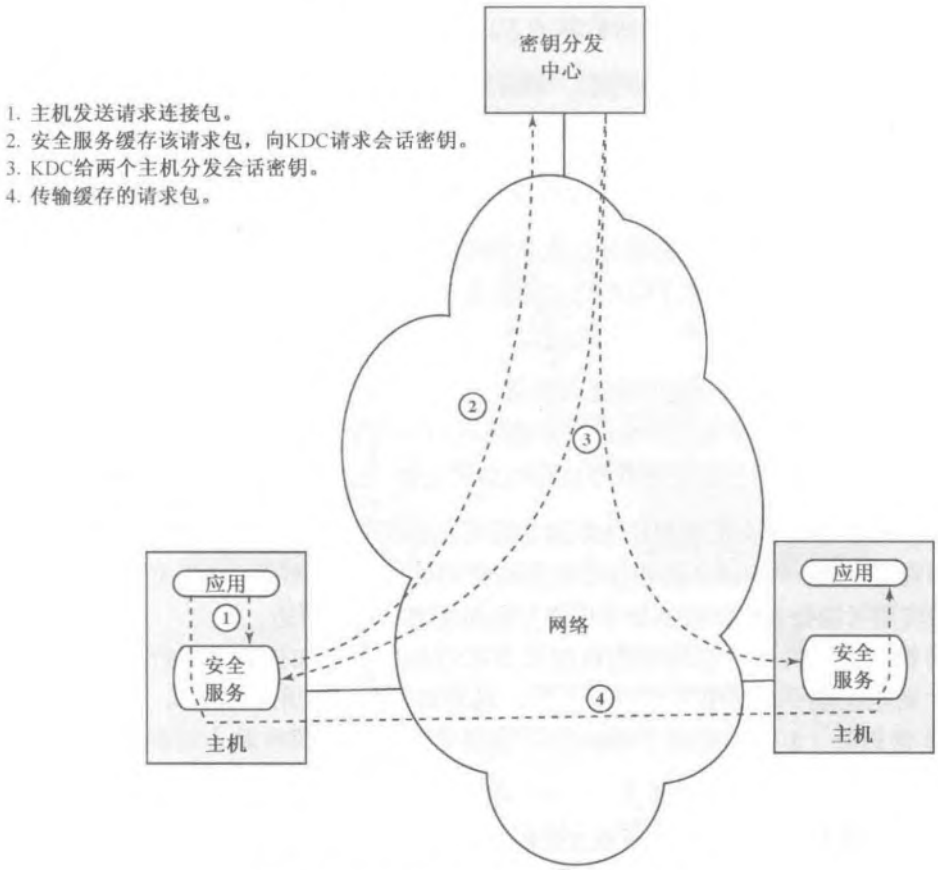


图 14.4　面向连接协议的自动密钥分发

自动密钥分发方法使得终端用户访问主机或者主机之间交换数据时更加灵活。

14.1.5　分布式密钥控制

密钥分发中心必须是可信的，而且是防破坏的，但如果密钥是完全分布式则无此要求。虽然对只使用对称加密的大型网络，完全分布式是不实用的，但是在局部环境下还是很有用的。

为了进行会话密钥的分发，分布式方法要求每个终端系统都能够以安全的方式与所有潜在的同伴或系统进行通信，因此，配置 n 个终端的分布式密钥分发中心需要大概 $[n(n-1)]/2$ 个主密钥。

设置一个会话密钥大致需要以下几步（参见图 14.5）：

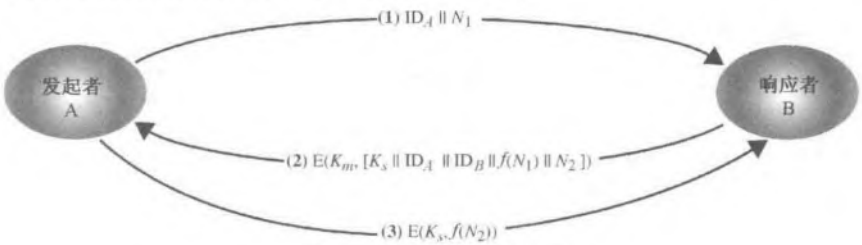


图 14.5　分布式密钥分发

（1）A 发送会话密钥请求给 B，包括一个临时交互号 N_1。

（2）B 用共享主密钥加密信息并回复给 A，该信息中包含 B 选择的会话密钥、B 的标识符、值 $f(N_1)$、临时交互号 N_2。

（3）使用新的会话密钥，A 返回信息 $f(N_2)$ 给 B。

每个节点至多拥有（n–1）个主密钥，和所需要的会话密钥一样多。信息传递使用主密钥的时间很短，因此密钥分析很困难。与之前一样，会话密钥用于在有限的时间内保护消息。

14.1.6 控制密钥的使用

密钥层次体系的概念和自动密钥分发技术的使用大大减少了必须手动管理和分发的密钥数量，而且满足自动密钥分发使用方式上强加控制的要求。如除了划分主密钥和会话密钥外，还要定义基于不同用途的各种会话密钥，如

- 数据加密密钥，用于网络中的通用通信
- PIN 加密密钥，用于电子资金过户和销售点应用的个人识别码（PIN）
- 文件加密密钥，用于存储公开可访问的加密文件

另外还需要考虑一个主密钥被作为数据加密密钥的风险。通常，主密钥在 KDC 和终端系统的硬件中是物理安全的，所以该主密钥加密的会话密钥是安全的。然而，如果主密钥曾作为会话密钥，则非授权的应用可能会获得以后通信中用该主密钥加密的会话密钥。

基于密钥关联的特征，在限制密钥使用方式的系统中添加控制是可行的。一个简单的设计是为每一个密钥关联一个标签[JONE82], [DAVI89]，这里提出的技术用于 DES，使用每个 64 位 DES 中的 8 个非密钥位［8 个非密钥（non-key）位通常预留为奇偶校验］以形成密钥标签。其中这些位有以下解释：

- 1 位表示该密钥是会话密钥还是主密钥
- 1 位表示该密钥能否用于加密
- 1 位表示该密钥能否用于解密
- 剩下的位预留为将来使用

因为标签嵌入在密钥中，密钥分发时标签也一起被加密。这种方案的缺陷是

（1）标签的长度被限制为 8 位，限制了它的灵活性。

（2）标签不能以清晰的格式传播，所以只能解密后使用，限制了对密钥使用方式的控制。

一个更加灵活的策略是控制向量，在文献[MATY91Za，MATY91Zb]中有描述。在这种策略中，每个会话密钥有一个关联控制向量，该向量包含很多域，详细说明了会话密钥的用法和限制。控制向量的长度各不相同。

会话密钥在 KDC 中产生时，控制向量和密钥一起被加密，耦合和去耦进程如图 14.6 所示。第一步，控制向量通过 Hash 函数（产生一个与加密密钥长度相同的值，Hash 函数已在第 11 章详细讨论）传递。大体上，Hash 函数用一种合理的方式将数值从大范围映射到小范围中，例如，范围是 1 ~ 100 的数字被映射到 1 ~ 10 之间的数字，大约 10%的原值被映射到一个目标值。

然后将 Hash 值和主密钥异或得到一个输出，该输出作为输入密钥来加密会话密钥，即

$$\text{Hash 值} = H = h(\text{CV})$$

$$\text{输入密钥} = K_m \oplus H$$

$$\text{密文} = E([K_m \oplus H], K_s)$$

其中，K_m 是主密钥，K_s 是会话密钥。通过逆操作可恢复出会话密钥

$$D([K_m \oplus H], E([K_m \oplus H], K_s))$$

KDC 将会话密钥传递给用户时，伴随着一个明文形式的控制向量。恢复会话密钥必须同时使用用户与 KDC 共享的主密钥以及控制向量，即会话密钥与其控制向量之间的链接是被保持的。

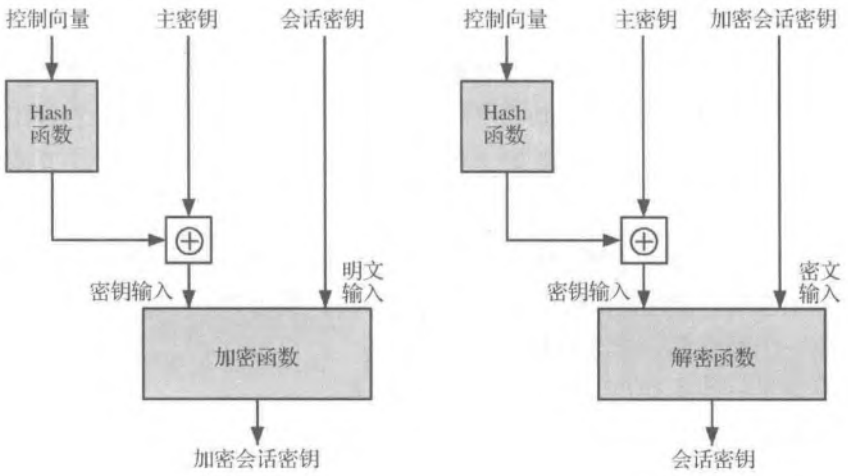


图 14.6　带控制向量的加密及解密

相比于 8 位标签，使用控制向量有以下两个优点：第一，控制向量的长度没有限制，可以给密钥的使用添加任意复杂的控制；第二，在所有的操作层面上，明文形式的控制向量都是有效的，即密钥应用的控制可以在多个位置使用。

14.2　基于非对称加密的对称密钥分发

因为公钥加密系统的效率比较低，所以几乎不会用于大数据块的直接加密，而是经常用于小块数据的加密。公钥密码系统最重要的应用之一是用于密钥的加密分发，在第五部分将会介绍很多例子。这里，我们讨论通用的原则和典型的方法。

14.2.1　简单密钥分发方案

Merkle 在文献[MERK79]中提出了一种非常简单的方案，参见图 14.7。如果，A 想要和 B 通信，需要以下几步：

（1）A 产生一个公私钥对 $\{PU_a, PR_a\}$，然后发送包含 PU_a 和 A 的标识符 ID_A 的消息给 B。

（2）B 产生密钥 K_s，用 A 的公钥加密后发送给 A。

（3）A 计算 $D(PR_a, E(PU_a, K_s))$ 从而恢复密钥，因为只有 A 能解密该信息，所以只有 A 和 B 知道 K_s。

（4）A 丢弃 PR_a、PU_a，B 丢弃 PU_a

A 和 B 可以使用会话密钥 K_s 加密以进行通信，在通信完成以后，A 和 B 都丢弃 K_s。尽管很简单，但也是一个很引人注目的协议。在通信开始之前和结束之后都不存在密钥，即密钥被攻破的风险也是最小的，同时通信是防窃听的。

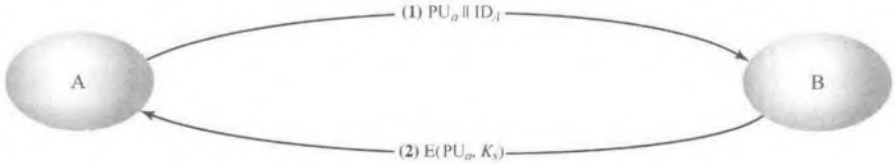


图 14.7 利用公钥加密建立会话密钥的简单方案

当有敌手截获消息然后再转发消息或者替换为其他消息［参见图 1.3(c)］时，图 14.7 描述的协议是不安全的，这样的攻击称为中间人攻击[RIVE84]。这种类型的攻击在第 10 章已经出现过（参见图 10.2）。这种情况下，如果敌手 D 已经控制了中间的通信信道，则可以通过以下方式在不被觉察的情况下控制通信（参见图 14.8）：

（1）A 产生一个公私钥对{PU_a,PR_a}，后发送包含 PU_a 和 A 的标识符 ID_A 的消息给 B。
（2）D 截获信息后，创建自己的公私钥对{PU_d,PR_d}，发送 $PU_d \| ID_A$ 给 B。
（3）B 产生一个密钥 K_s，发送 $E(PU_d, K_s)$。
（4）D 截获信息后计算 $D(PR_d, E(PU_d, K_s))$，获取 K_s。
（5）D 发送 $E(PU_a, K_s)$给 A。

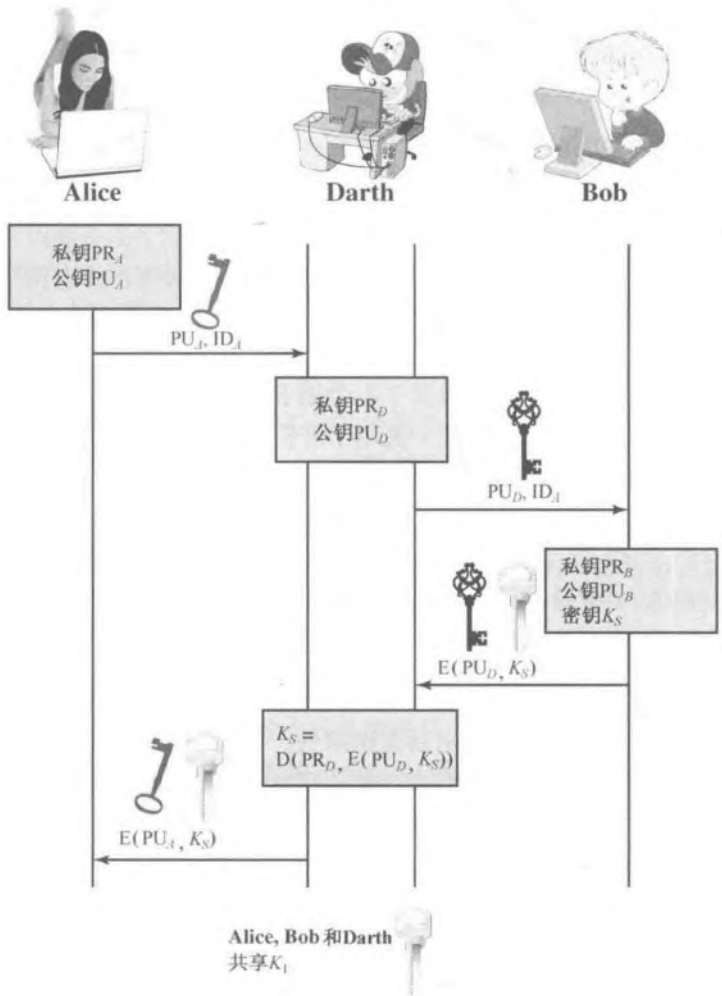


图 14.8 中间人攻击

结果就是 A、B 知道了密钥 K_s 但没有意识到 D 也已经获得了密钥 K_s。此时，若 A 和 B 使用 K_s 交换信息，D 不再需要主动的干扰通信信道，而只需要简单的窃听，因为知道 K_s，所以 D 可以解密所有的信息。因此，该协议存在中间人攻击的威胁。

14.2.2　确保保密性和身份验证的密钥分发方案

如图 14.9 所示，基于在文献[NEED78]中提出的方法，可以防止主动的和被动的攻击。我们假定 A 和 B 已经通过本章随后要介绍的方案中的一种来交换公钥，以下是接下来的步骤：

（1）A 用 B 的公钥去加密包含 A 的标识符 ID_A 和一个临时交互号 N_1 的消息并发给 B，其中临时交互号被用来唯一的标识该次消息传递。

（2）B 用 PU_a 加密包含 A 的临时交互号 N_1 及 B 产生的新的临时交互号 N_2 的消息发送给 A。因为只有 B 可以解密消息（1），故 N_1 在消息（2）中出现可以使 A 确定该消息来自于 B。

（3）A 使用的 B 的公钥加密后返回 N_2，使 B 确定消息来自于 A。

（4）A 选择密钥 K_s 后发送 $M = E(PU_b, E(PR_a, K_s))$ 给 B。用 B 的公钥加密确保只有 B 可以读取该消息，用 A 的私钥加密保证只有 A 可以发送该消息。

（5）B 计算 $D(PU_a, D(PR_b, M))$ 从而恢复密钥 K_s。

该方案可以保证交换密钥过程中的保密性和身份验证。

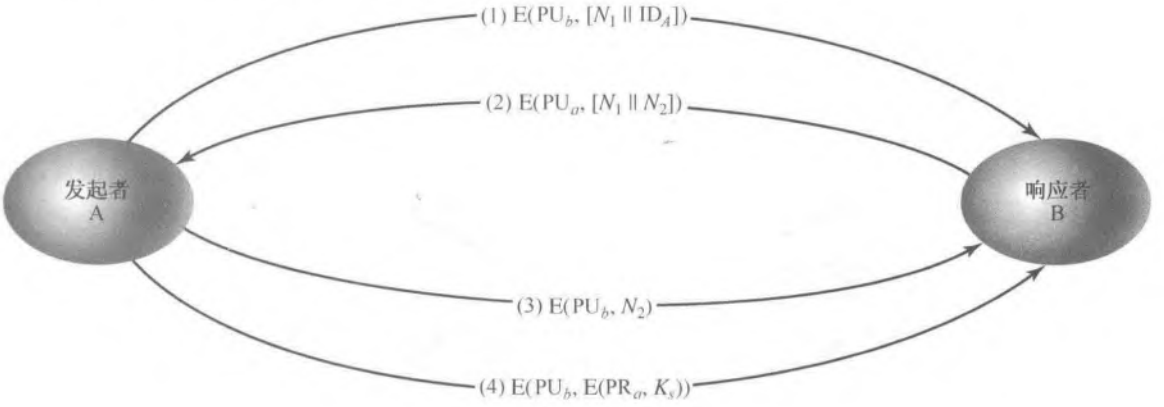


图 14.9　利用公钥分发密钥

14.2.3　混合方案

另一种使用公钥加密来分发密钥的方法是在 IBM 大型机[LE93]中使用的混合方法。该方案仍会用到密钥分发中心 KDC，KDC 和每个用户共享一个主密钥，用主密钥加密要分发的会话密钥，公钥方案被用于分发主密钥。以下是使用这个三层方案的理由：

- **性能**　应用广泛，尤其是在会话密钥变动频繁的面向事务的应用中。用公钥加密的会话密钥分发会降低整个系统的性能，因为公钥加密和解密需要相对较高的计算负荷。对一个三层体系，公钥加密只用于偶尔更新用户和 KDC 之间的主密钥。
- **后向兼容性**　混合方案可以仅用最小的破坏或者软件的更改覆盖已存在的 KDC 模式。

公钥层的加入可以提供一个安全有效的分发主密钥的方式，这在一个单个的 KDC 要为一个庞大的用户集分发密钥的机构中是一个优势。

14.3 公钥分发

人们已经提出了集中公钥分配方法，所有这些方法本质上都可以归结为以下几种方法：

- 公开发布
- 公开可访问目录
- 公钥授权
- 公钥证书

14.3.1 公钥的公开发布

表面上看，公钥密码的特点是公钥可以公开，因此如果有像 RSA 这样的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个通信方或广播给通信各方（参见图 14.10）。例如，越来越被人们广泛使用的 PGP（Pretty Good Privacy，将在第 19 章讨论）中使用了 RSA 算法，所以很多 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消息之后。

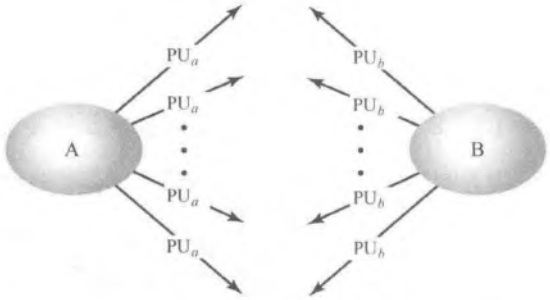


图 14.10 无控制的公钥分发

虽然这种发送比较简便，但是有一个较大的缺点，任何人都可以伪造这种公钥并公开发布，也就是说，某个用户可以假冒用户 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用户之前，该假冒者可以读取本应发送给 A 的加密后的消息，并且可以用伪造的密钥进行认证（参见图 9.3）。

14.3.2 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（参见图 14.11），这种方法包含下面几方面的内容：

（1）管理员通过为每一个通信方建立一个目录项{姓名，公钥}来维护该目录。
（2）每个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。用户希望更换公钥，可能是因为公钥已用于大量的数据，也可能是因为相应的私钥已经泄露。
（4）通信方也可以访问该目录，为实现这一目标，必须有从管理员到通信方安全的认证通信。

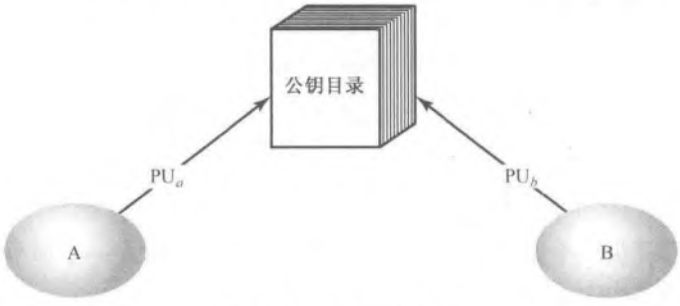


图 14.11 公钥的分发

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。

14.3.3　公钥授权

通过更加严格的控制目录中的公钥分配，可使公钥分配更加安全。图 14.12 举例说明了一个典型的公钥分配方案，它基于文献[POPE79]中给出的图形。像之前一样，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 14.12 中的序号相对应）：

（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 PR_{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。这条消息包含以下内容：

- B 的公钥 PU_b，A 用来加密发给 B 的消息。
- 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，以确保在管理员收到请求之前，其原始请求未被修改。
- 原先的时间戳，A 可以确定它收到的不是来自管理员的旧消息。

（3）存储 B 的公钥，并用它去加密包含 A 的身份标识符 ID_A 和临时交互号 N_1 的消息发送给 B，其中 N_1 唯一标识该次交互。

（4），（5）与 A 检索 B 的公钥一样，B 使用同样的方法从管理员处得到 A 的公钥。

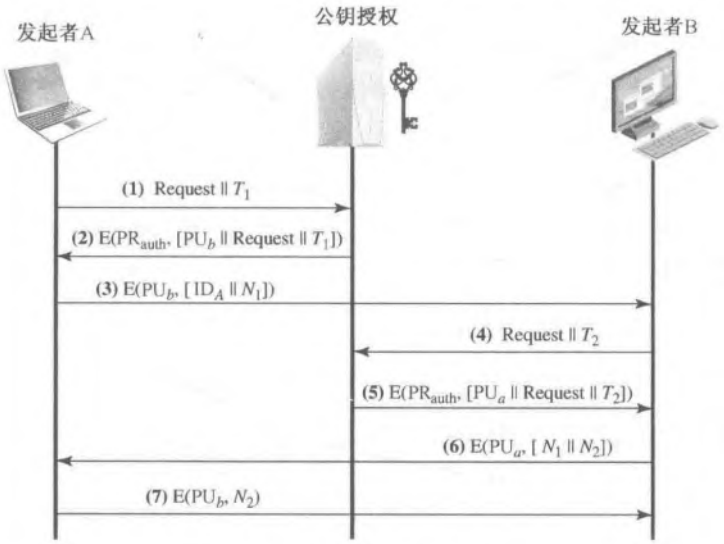


图 14.12　公钥分发方案

此时，公钥已被安全地传递给 A 和 B，A、B 之间的信息交换将会受到保护，尽管如此，最好还包含以下两步：

（6）B 用 A 的公钥 PU_a 加密包含 A 的临时交互号 N_1 和 B 新产生的临时交互号 N_2 的消息，并发送给 A。因为只有 B 能解密消息（3），所以消息（6）中的 N_1 可以使 A 确信该消息来自于 B。

（7）A 用 B 的公钥加密包含 N_2 的消息给 B，这样 B 就可以知道该消息来自于 A。

这样，总共需要 7 条消息，然而，前面的 4 条消息不会被频繁使用，因为 A、B 可以存储彼此的公钥以备将来之需。用户需要周期性地请求当前公钥信息，以保证通信中使用的是当前的公钥。

14.3.4 公钥证书

图 14.12 的方案虽然不错，但是它还有缺陷。因为用户要与其他用户通信，他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。

最早由 Kohnfelder 提出了使用证书的方法在文献[KOHN78]中。通信双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。证书包含公钥和公钥拥有者的标识，整个数据块由可信的第三方进行签名。通常，第三方是证书管理员，如政府机构或者金融机构，为用户群所信任。一个用户以一种安全的方式将他的公钥交给管理员，从而获得一个证书，接着用户就可以公开证书。任何需要该用户公钥的人都可以获得该证书，并通过查看附带的可信签名来验证证书的有效性。通信的一方也可以通过传递证书的方式将他的密钥信息传达给另一方。其他通信各方可以验证该证书确实是由证书管理员生成的。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）只有证书管理员可以产生并更新证书。

文献[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时效性。

图 14.13 举例说明了证书的使用方法。通信方向证书管理员提供一个公钥并请求证书。申请必须是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：

$$C_A = \mathrm{E}(\mathrm{PR}_{\mathrm{auth}}, [T \| \mathrm{ID}_A \| \mathrm{PU}_a])$$

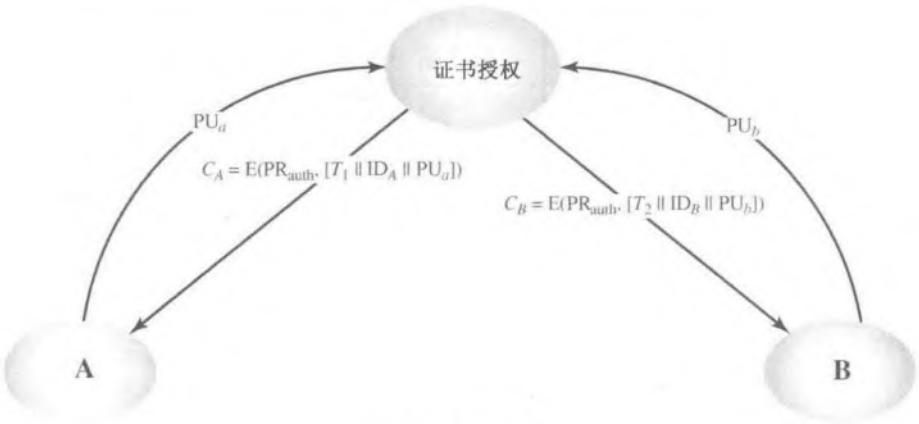
其中，$\mathrm{PR}_{\mathrm{auth}}$ 是证书管理员私钥，T 为时间戳。A 可以把该证书传递给任何其他参与者，后者通过以下方式读取和验证证书：

$$D(\mathrm{PU}_{\mathrm{auth}}, C_A) = \mathrm{D}(\mathrm{PU}_{\mathrm{auth}}, \mathrm{E}(\mathrm{PR}_{\mathrm{auth}}, [T \| \mathrm{ID}_A \| \mathrm{PU}_a])) = (T \| \mathrm{ID}_A \| \mathrm{PU}_a)$$

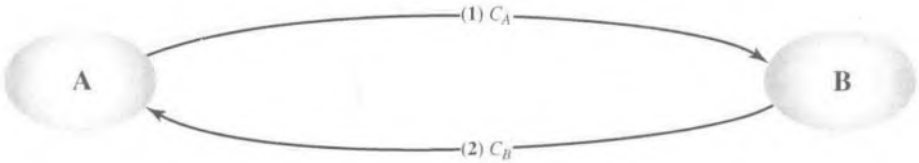
接受方使用证书管理员的公钥 $\mathrm{PU}_{\mathrm{auth}}$ 对证书解密，因为证书只有使用管理员的公钥才能读取证书，因此接收方可验证证书确实是来自于证书管理员的。ID_A 和 PU_a 向接收方提供证书持有者的名字和公钥，时间戳 T 验证证书的时效性。时间戳可以在攻击者已知 A 的私钥的情况下抗攻击，假设 A 产生新的公私钥对并向公钥管理员申请新的证书，同时，攻击者重放 A 的旧的证书给 B，若 B 用旧公钥加密数据的话，则攻击者就可以读取信息。

在这种情况下，私钥的泄密就如同信用卡的丢失一样，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的信用卡过时的时候，才能保证卡的持有者的安全，因此，时间戳有一些像截止日期。若一个证书超时，会被认为证书已失效。

X.509 标准是一个广为接受的方案，用来规范公钥证书的格式。X.509 证书在大部分网络安全应用里都有使用，包括 IP 安全、传输层安全（TLS）和 S/MIME，所有这些都在第五部分讨论。X.509 将在下一节进行详细介绍。



(a)从CA获得证书



(b)交换证书

图 14.13　公钥证书的交换

14.4　X.509 证书

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，用户信息包括用户名到网络地址的映射和用户的其他属性等。

X.509 定义了 X.500 用户目录的一个认证服务框架，这个目录可以作为在 14.3 节中讨论的公钥证书类型的存取库。每个证书包含该用户的公钥并由一个可信的签证机构用私钥签名。另外，X.509 还定义了基于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 证书格式被用于 S/MIME（参见第 19 章）、IP 安全（参见第 20 章）和 SSL/TLS（参见第 17 章）。

X.509 于 1988 年首次发布，随后为了解决文献[IANS90]和[MITC90]中提及的安全问题对其进行了修订，修订稿于 1993 年发布。标准的当前版本是第 7 版，于 2012 年发布。

X.509 是基于公钥密码体制和数字签名的标准。标准中并未规定使用某个特定的数字签名算法或某个特定的 Hash 函数。图 14.14 描述了 X.509 方案中一个公钥证书的产生过程。公钥的证书包括 Bob 唯一的身份标识、公钥、CA 的鉴别信息，以及一些其他信息。通过计算这些信息的 Hash 值，使用 CA 的私钥对 Hash 值生成数字签名，完成对这些信息的签名过程。X.509 表明签名是对 Hash 值进行加密而形成的，在 13.6 节提到的一种 RSA 方案被推荐使用。然而，目前版本的 X.509 并没有指明特定的数据签名算法。如果使用 NIST DSA（参见 13.4 节）或者 ECDSA（参见 13.5 节），则 Hash 值不会被加密，而是作为数字签名生成算法的输入。

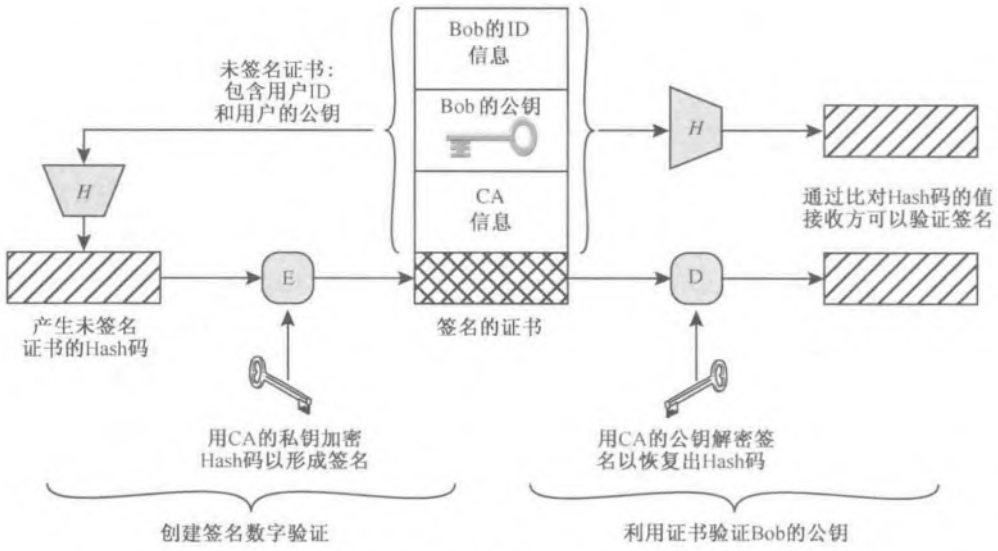


图 14.14 X.509 公钥证书的应用

14.4.1 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签证机构（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 14.15(a)表明证书常用格式包含以下要素：

- **版本号**（Version） 区分证书的不同版本，默认设置为 1。如果存在发行商唯一标识或主体唯一标识，则版本号为 2，如果存在一个或多个扩展，则版本号为 3。虽然 X.509 标准现在最新版本为 7，但是自版本 3 之后证书的此域就没有做过修改。
- **序列号**（Serial number） 一个整数，在 CA 中唯一标识证书。
- **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，这里很少含有信息。
- **发行商名字**（Issuer name） X.500 中创建、签名证书的签证机构 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。
- **发行商唯一标识**（Issuer unique identifier） 由于 X.500 的名字被许多不同实体引用，因此用可选位串唯一标识签证机构。
- **证书主体唯一标识**（Subject unique identifier） 由于 X.500 的名字被许多不同实体引用，因此用可选位域唯一标识证书主体。
- **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） 用 CA 私钥对证书的所有域及对这些域的 Hash 值一起加密。

唯一标识域是在版本 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：

$$CA\langle\langle A\rangle\rangle = CA\{V,SN,AI,CA,UCA,A,UA,Ap,T^A\}$$

其中

Y<<X>>：用户 X 的证书是签证机构 Y 发放的

Y{I}：Y 签名 I，包含 I 和 I 被加密后的 Hash 代码

V：证书的版本

SN：证书的序列号

AI：用于给证书签名的算法的标识

CA：证书机构的名称

UCA：CA 的可选择的唯一标识

A：用户 A 的名称

UA：A 的可选择的唯一标识

Ap：用户 A 的公钥

T^A：证书的有效周期

CA 用自己的私钥签署证书，如果用户知道相应的公钥，则用户就可以验证证书是 CA 签署的。在图 13.2 中举例说明了这种典型的数字签名方法。

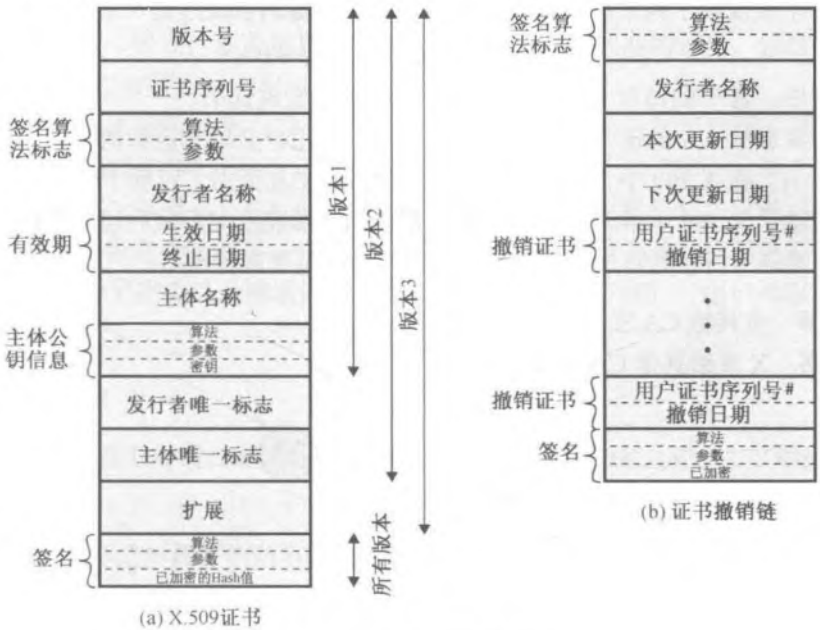


图 14.15　X.509 格式

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，则说明用户普遍信任 CA，所有用户的证书均被存放于同一个目录中，所有用户都可以进行存取。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个 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，使用户可以信任该证书。

现在，假设 A 获得了签证机构 X_1 的证书，而 B 获得了签证机构 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，从而安全地获得 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 可以逆向地获得 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来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 14.16 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：

- **前向证书** 由其他 CA 发给 X 的证书。
- **后向证书** X 发给其他 CA 的证书。

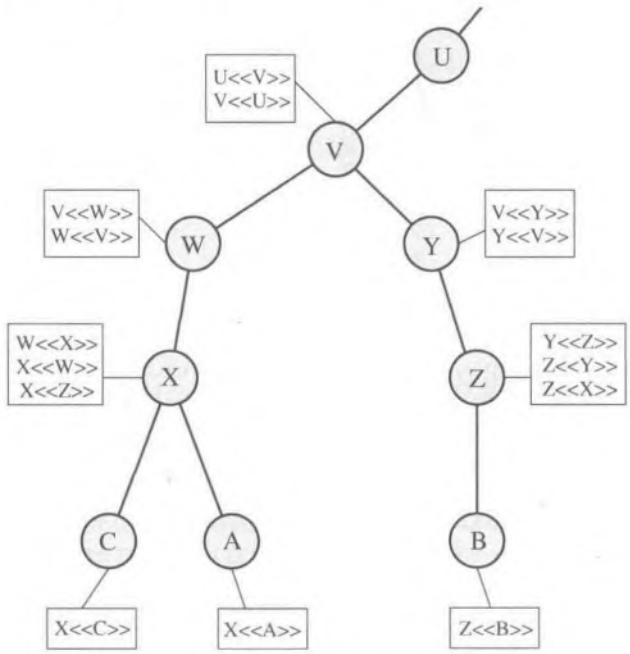


图 14.16 X.509 的层次结构：一个假设示例

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<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

当 A 获得相关证书后，可以通过顺序展开证书路径来获得 B 的公钥，用该公钥，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来获得 A 的公钥

Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

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。

证书撤销　回顾图 14.15，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回证书：

（1）用户私钥被认为不安全。

（2）用户不再信任该 CA。原因包括主体名已改变、证书已被废弃或者证书没有按照 CA 的规则发行。

（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［参见图 14.15(b)］。每个入口中包含该证书的序列号和撤销时间。由于一个 CA 中的序列号是唯一的，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，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（和可能的其他开销），用户可以维护一个本地的证书及 CRL 的缓存。

14.4.2　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。文献[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。

相比于向固定格式中继续增加字段，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，则其证书将会被当成非法的证书。

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已命名的规则集，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** 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（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，私钥的使用期一般比公钥短。
- **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，一些信息如邮件地址、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信息。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。

14.5 公钥基础设施

RFC 4949（互联网安全术语）定义了 PKI 系统是由硬件、软件、人、策略和程序构成的一整套体系。这些程序是用来创建、管理、存储、分发和撤销建立在非对称密码算法之上的数字证书。创建 PKI 的主要目的就是用来安全、便捷、高效地获得公钥。IETF（Internet Engineering Task Force）

的 PKIX（Public Key Infrastructure X.509）工作组在 X.509 的基础上建立了一个可以用来构建网络认证体系的基本模型。本节就简单介绍一下 PKIX 模型。

图 14.17 描述了在 PKIX 模型中各个关键元素之间的相互关系。这些元素是

- **端实体**　是必备的元素，它可以是一个终端用户、设备（如应用服务器、路由器等），或是其他可以在一个公钥数字证书作用范围中被认证的实体。终端实体支持 PKI 相关的设备。
- **签证机构（CA）**　是证书和证书撤销列表（CRL）的发行人，常常在其上运行着一个或多个注册机构（RA），同时它还承担一些管理任务。
- **注册机构（RA）**　是可选的元素，可以承担一些签证机构（CA）的管理任务。一般来说都是和端实体的注册进程相关的任务，也可以支持一些其他的管理任务。
- **证书撤销列表（CRL）发布点**　是可选的元素，签证机构可以通过它来发布证书撤销列表（CRL）。
- **证书存取库**　是必备的元素，它提供了存取数字证书和证书撤销列表（CRL）的方法，可以被终端用户检索。

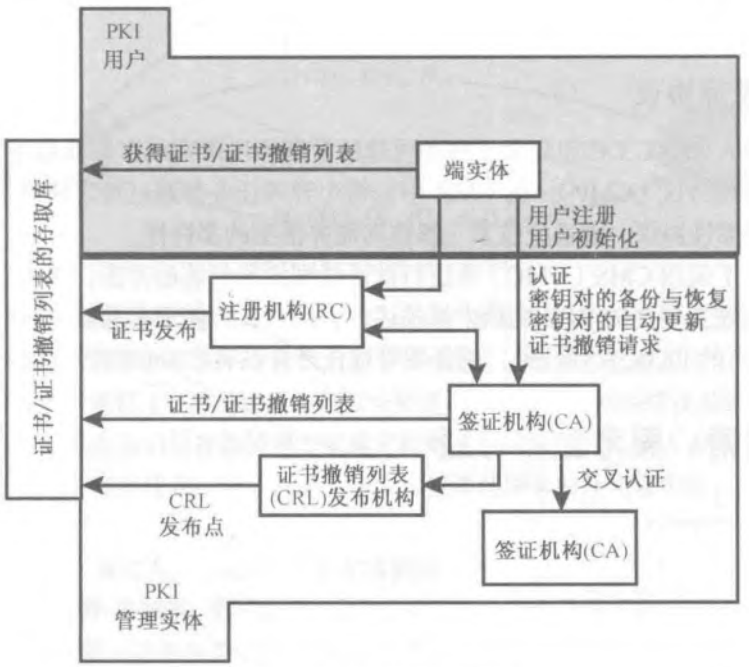


图 14.17　PKIX 结构模型

14.5.1　PKIX 管理任务

PKIX 体系中定义了一系列的管理任务，它们是在管理协议的支持下进行工作的。图 14.17 中描述了这些管理任务：

- **用户注册**　这是用户第一次进行认证之前进行的活动，它先于 CA 为用户颁布一个或多个证书。在 PKI 中注册从登记进程开始，注册通常包括一系列的在线和离线的交互认证，代表性地，端实体会在后续认证中发布一个或者多个共享密钥。
- **初始化**　在一个客户系统安全操作之前，有必要将和密钥存储相关的关键材料安装在基础设

施的别处，例如，客户需要用公钥和其他一些来自可信 CA 的确认信息安全的初始化，用于确认证书路径。

- **认证书** 这个进程中，签证机构通过用户的公钥向用户发放一个证书，返回证书给用户客户系统，并在证书存取库中进行保存。
- **密钥对的恢复** 密钥对可以用于数字签名的创建、验证和数据加解密。而对于数据加解密来说，当解密钥丢失时，必须提供机制来恢复解密密钥，这对恢复加密数据来说非常重要。密钥的丢失通常是由密钥遗忘、磁盘驱动崩溃、硬件令牌毁坏等原因造成的。密钥对的恢复使得终端实体重新存储源于授权的密钥反馈机制（如发布终端实体证书的 CA）的加解密密钥。
- **密钥对更新** 所有的密钥都需要定期更新（即替换为新的密钥对），及发布相应的证书。当证书的生命周期终止或者证书撤销时，需要更新。
- **证书撤销请求** 一个授权用户可以向签证机构提出要求撤销证书。当发生密钥泄露、从属关系变更或更名等时，需要提交这种请求。
- **交叉认证** 如果两个签证机构之间要交换数据，则可以通过交叉认证来建立信任关系。一个交叉认证证书是一个签证机构发布给另一个签证机构的，包含用来发布证书的数字签名。

14.5.2 PKIX 管理协议

在 PKIX 实体间，PKIX 工作组定义了两种可选的管理协议来完成在上节描述的管理任务。RFC 2510 定义了证书管理协议（CMP），在 CMP 中，每个管理任务都通过特定协议交换来明确识别。另外，CMP 是一种柔性协议，能适应技术、操作和商务模型的多样性。

RFC 2797 定义了采用 CMS（CMC）来进行认证管理消息封装的方法，其中 CMS 在 RFC 2630 中被描述，CMC 则在更早些的 RFC 文档中被描述，它们的设计使现有的技术发挥出更高的效能。另外，虽然支持所有的 PKIX 管理任务，但各项管理任务并没有和特定的协议交互一一对应。

14.6 关键术语、思考题和习题

关键术语

端到端加密	密钥分发	密钥分发中心（KDC）	密钥管理
中间人攻击	主密钥	临时交互号	公钥证书
公钥目录	X.509 证书		

思考题

14.1 列出在两个通信方之间分发密钥的方法。

14.2 会话密钥和主密钥之间有什么不同？

14.3 什么是临时交互号？

14.4 什么是密钥分发中心？

14.5 基于公钥加密的两种不同的密钥分发方法是什么？

14.6 列出四种公钥分发模式的类型。

14.7 公钥目录的必要要素是什么？

14.8　什么是公钥证书?

14.9　公钥证书的使用有哪些要求?

14.10　X.509 标准的用途是什么?

14.11　什么是证书链?

14.12　X.509 证书如何撤销?

习题

14.1　本地网络向量提供一个密钥分发方案，如图 14.18 所示。

（a）描述该方案。

（b）相比图 14.3 的方案，有哪些优点和缺点。

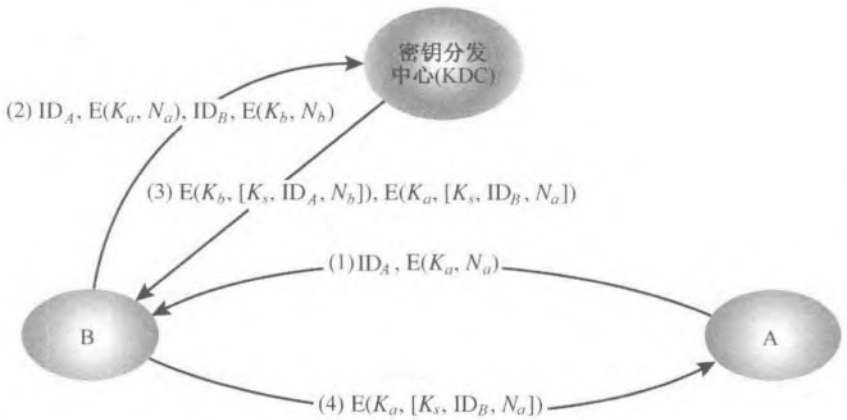


图 14.18　习题 14.1 的图

14.2　“福尔摩斯，我们有很大的压力。”侦探 Lestrade 看起来很紧张。“我们已经知道，一些敏感的政府文件复印件出现在伦敦一个外国大使馆的电脑上，通常，这些文件一般以电子形式存放在少数几台政府计算机上，满足最严格的安全要求。有时要通过网络连接发给所有的政府计算机，但是这些消息是由我们最好的加密专家鉴定的绝密算法加密的，NSA 和 KGB 都无法破解。但是现在这些文件却出现在一个小的甚至是无关紧要的国家的外交官手里，我想不明白为什么会发生这种事情。”

“但是你已经有了嫌疑人，不是吗？”福尔摩斯问。

“是的，我们做了一些调查，有一个男人合法地访问过一台政府计算机，并且和那个外交官联系很频繁。但是他访问的不是那些文件通常存放的计算机，他是嫌疑犯，但我们不知道他是如何获得文件复印件的。即便他得到加密的文件，也不能解密。”

“请描述下，该网络使用的通信协议。”福尔摩斯睁开眼睛，以告诉 Lestrade 他虽然一脸睡意但是他还是在认真听着。

“协议如下，网络中的每个节点 N 都有唯一的密钥 K_n，用于节点和可信服务器之间的安全通信，即所有的密钥也在服务器中存放。用户 A 想要发送秘密消息 M 给 B 时，使用以下协议：

（1）A 产生临时交互号 R，发送自己的名字 A、目的地 B 和 $E(K_a,R)$ 给服务器。

（2）服务器回复消息 $E(K_b,R)$ 给 A。

（3）A 发送 $E(R,M)$ 和 $E(K_b,R)$ 给 B。

（4）B 使用 K_b 解密 $E(K_b,R)$ 得到 R，随后应用 R 解密 $E(R,M)$ 得到消息 M。

每次有消息发送时产生一个随机密钥，我承认他可能在几个节点之间发送时截获消息，但是，他不可能解密消息。”

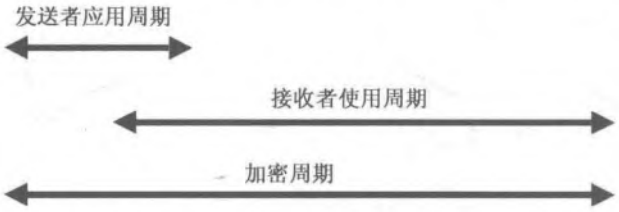
“我相信你有你的道理，Lestrade。这个协议是不安全，因为服务器不能鉴定谁发的请求。明显地，协议设计者相信发送 $E(K_x,R)$ 就可以鉴定发送者为用户 X，因为只有 X 知道 K_x，但是你也知道 $E(K_x,R)$ 可能被截获然后重放。只要知道漏洞在哪里，通过监控该男子对访问计算机的使用，可以得到更多的证据。他最有可能是这么做的：截获 $E(K_a,R)$ 和 $E(R,M)$ 后，我们称该男子为 Z，会假装 A，然后……”

完成福尔摩斯的话。

14.3 X.509 的 1988 版中列出了这样的内容：在现有的知识层面上，分解大整数是很困难的，故 RSA 一定是安全的。对公开的指数和模数 n 有如下限制：必须确保 $e > \log_2(n)$，才能阻止通过用第 e 个根来模 n 以得到明文的攻击。尽管，这个限制是对的，但是给出的原因却是错误的。错在哪里？正确的原因是什么？

14.4 在你的计算机上，找出至少一个中间的证书管理员的证书和一个根证书管理员的证书（举例来说，在浏览器）。显示每个证书大致的和详细的表的屏幕截图。

14.5 NIST 定义了加密周期（cryptoperiod）作为时间跨度，期间一个密钥被授权使用，或者，给定系统或应用的密钥仍旧有效。一个关于密钥管理的文件中，为共享密钥使用了如下的时间图表。



通过给出一个应用实例来解释重叠部分，在该实例中要求发送者对会话密钥的使用周期开始的比接收者应用周期早，且结束的也比较早。

14.6 考虑以下协议，为了 A 和 B 得到一个新的会话密钥 K'_{AB}，假定它们已经共享了一个长期密钥 K_{AB}。

（1）$A \rightarrow B: A, N_A$

（2）$B \rightarrow A: E(K_{AB}, [N_A, K'_{AB}])$

（3）$A \rightarrow B: E(K'_{AB}, N_A)$

（a）首先尝试去理解设计者的思路：

——在协议运行之后，A 和 B 为什么会相信共享的会话密钥会来自彼此？

——它们为什么会相信这个会话密钥是最新的？

在这种情况下，要解释 A 和 B 相信的原因，所以你的回答应完成下列句子：

A 相信和 B 共享 K'_{AB} 是因为……

B 相信和 A 共享 K'_{AB} 是因为……

A 相信 K'_{AB} 是最新的因为……

B 相信 K'_{AB} 是最新的因为……

（b）假定现在 A 和 B 开始运行该协议，然而，连接被 C 截获。展示 C 如何利用映像重新运行协议，使得 A 认为已经和 B 就新的会话密钥达成一致（尽管事实上是和 C 进行了通信）。从而，（a）中是错误的。

（c）为了阻止该攻击，提出一种修改方案。

14.7　PKI 的关键组成部分有哪些？简单描述各个部分。

14.8　解释密钥管理的问题以及它是如何影响对称加密的。

注意：剩下的习题涉及 IBM 开发的一个加密产品，在本书的网址（IBMCryto.pdf）上的一个文件做了简单的描述。在阅读完文件后，尝试以下习题。

14.9　添加指令 EMK_i（$\mathrm{EMK}_i : X \rightarrow E(\mathrm{KMH}_i, X) i = 0,1$）的作用有哪些？

14.10　假设 N 个不同的 IBM 加密子系统中有主机主密钥 $\mathrm{KMH}[i] (i = 1,2,\cdots,N)$，设计一种系统之间的通信方法，而各个系统既不用共享一个主机主密钥又不会泄露自己的主机主密钥。提示：每个系统都需要有自己主密钥的三个不同的变体。

14.11　IBM 密码子系统的主要目的是保护终端和处理系统之间的通信，设计一个程序，可能会添加指令，允许处理器产生一个会话密钥 K_S，并分发给终端 i 和终端 j，而不需要在主机中存储与密钥相关的变量。

第15章 用户认证

学习目标

通过本章的学习应具备以下能力：

- 理解身份鉴定和身份核实的区别
- 概述基于对称加密的远程用户认证技术
- 给出关于 Kerberos 的介绍
- 解释 Kerberos 版本 4 与版本 5 之间的不同
- 描述 Kerberos 在多个领域中的使用
- 概述基于非对称加密的远程用户认证技术
- 理解联合身份管理系统的必要性
- 解释作为用户认证系统一部分的 PIV 机制的使用

本章举例说明了一些能支持网络应用认证的认证功能。15.1 节介绍了一些在网络或者互联网上实现用户认证的概念和要素，15.2 节介绍了基于对称加密的用户认证协议，15.3 节介绍最早也是应用最广泛的用户认证服务之一：Kerberos，15.4 节是基于非对称加密的用户认证协议，之后是对 X.509 用户认证协议的讨论，最后介绍联合身份（federated identity）的概念。

15.1 远程用户认证原理

在大多数的计算机安全环境中，用户认证是最基本也是最重要的一道防线，用户认证是访问控制以及用户承担责任的基础。RFC 4949（互联网安全术语）对用户认证的定义如下：核实身份的方法是由或是对一个系统实体提出的。认证方法包括以下两步：

- **鉴定阶段** 给安全系统提供身份标识（身份标识要认真的分配，因为身份认证是其他安全服务的基础，如访问控制服务）。
- **核实阶段** 提供或者产生可以证实实体和标识之间对应关系的认证信息。

例如，用户 Alice Toklas 有用户标识 ABTOKLAS，该信息存储在 Alice 想要访问的服务器或者计算机系统中，并且要使系统管理员及其他用户知道。一个典型的和用户 ID 相连的认证信息项是口令（只有 Alice 和系统知道）。如果没有人得到或者猜到 Alice 的口令，则 ID 和口令的结合使得管理员可以建立 Alice 的访问许可及审查她的活动，因为 ID 是不保密的，故其他用户可以给她发送 e-mail，但由于密码保密所以不能冒充 Alice。

事实上，身份鉴定是指用户提供一个声明的身份给系统，用户认证是使得声明有效的一种方式。注意用户认证和消息认证是不一样的，就像第 12 章定义那样，消息认证允许通信双方验证接收到的消息是否被更改以及资源是否可信。本章只关注用户认证。

15.1.1　NIST 电子身份认证模型

NIST SP 800-63-2文档（Electronic Authentication Guideline，2013.8）定义了信息系统中为以电子形式存在的用户身份实现信任建立的过程——电子身份认证。系统能通过已验证的身份判断授权个体能否执行特定功能，如数据库事务或系统资源访问。在很多情况下，认证、事务处理和其他授权功能都在开放网络如Internet上进行；同样地，认证及授权也能在本地进行（如局域网中）。

NIST SP 800-63-2 定义了涉及多个实体和进程的电子身份认证通用模型。我们对应图 15.1 来对模型展开讨论。

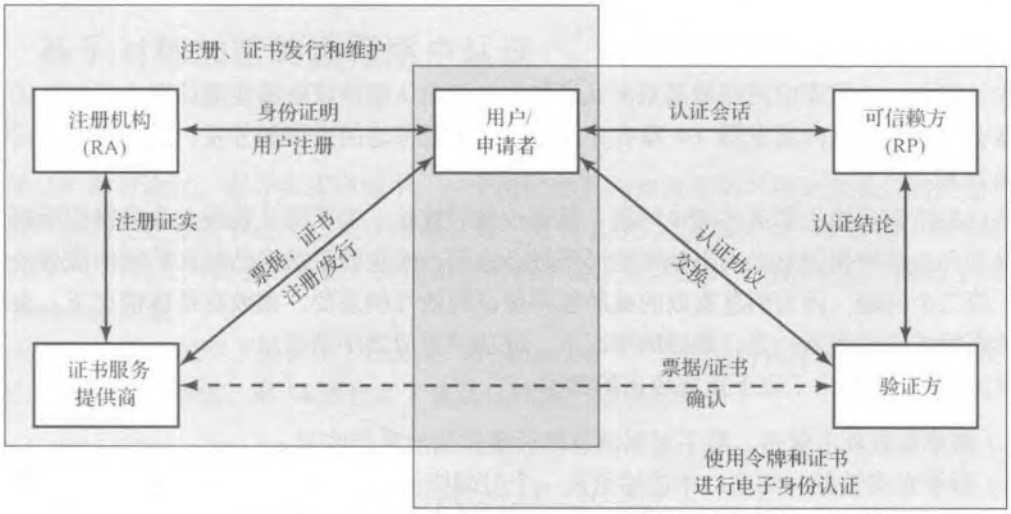


图 15.1　NIST SP 800-63-2 电子身份认证结构模型

身份认证的初始需求是用户必须在系统中注册，以下是用户注册的一个典型流程。申请者向注册机构（RA）申请成为证书服务提供商（CSP）的用户。在本模型中，RA 作为可信实体创建用户身份并向 CSP 为申请者提供担保。之后，CSP 会与用户建立信息交换。基于认证系统的不同实施细节，CSP 会给用户发行相应类型的电子票据。票据是一种将身份和附加属性与用户权标可信地绑定得到的数据结构，并在认证过程中可以被验证方所验证。权标可以是能够唯一标识用户的密钥或者加密口令，可以由 CSP 发行，由用户生成，或者由第三方提供。权标和票据会在随后的认证过程中用到。

申请者注册成为用户后，实际的认证过程会在用户和一个或多个执行认证、授权的系统之间进行。被认证的一方称为声称方，验证身份的一方称为验证方。当声称方在认证协议中向验证方成功证明了其对权标的所有权和控制权，后者将其鉴别为票据中标明的用户，并向可信赖方（RP）传递关于用户身份的声明。声明包括用户的身份信息，如用户名，注册时分配的标识符，或者其他已被证实的用户属性。RP 可以利用这些验证方提供的认证信息进行访问控制或授权决策。

实际中的认证系统各有不同，比起上述简化模型也复杂得多，但上述模型表明了一个安全的认证系统中所必需的关键角色和功能。

15.1.2　认证方式

认证一个用户的身份大致有 4 个常用工具，可以单独使用也可以一起使用：

- **知道什么**　如口令、个人身份号（PIN）或者之前准备的问题答案。
- **拥有什么**　如加密密钥、电子密钥卡、智能卡和物理密钥，这种类型的认证信息称为令牌。

- **静态生物特征** 如指纹、视网膜和脸。
- **动态生物特征** 如声音模式、手写特征和打字节奏。

所有这些方法通过适当的执行和应用都可以提供安全的用户认证。然而，每种方法都有缺陷，简单地说，攻击者可能伪造或者窃取令牌，用户可能会忘记口令或者丢失令牌。此外，管理系统的口令和令牌信息并确保这些信息的安全将会增加不少的系统开销。关于生物计量的认证信息，存在各种各样的问题，如处理假阳性和假阴性、用户满意度、成本和便利性。对于基于网络的用户认证，最重要的认证方法包括加密密钥和用户个人口令等。

15.1.3 双向认证

双向认证的一个重要应用领域是双向认证协议，双向认证协议能够使通信双方互相认证彼此身份并交换会话密钥。该内容在第 14 章有介绍，当时主要考虑的是密钥分发，在这里我们讨论认证更广泛的含义。

已认证的密钥交换主要关注两个问题：保密性和时效性。为了阻止伪装及会话密钥泄露，必要的身份鉴别和会话密钥信息必须以加密的形势进行通信，这也就要求有已存在的密钥或者公钥来进行加密。第二个问题，因为消息重放的威胁使得保证时效性很重要。重放在最坏情况下，会使敌手获得会话密钥或者扮演另一方；最好的情况下，成功的重放能中断通信。

文献[GONG93]列出了以下重放攻击的例子：

（1）简单重放攻击就是，敌手复制消息然后重放给消息接收者。
（2）敌手在有效的时间窗口中能够重放一个时间戳消息，如果原本的和重放的时间戳消息都在时间窗内收到，则这一事件能被日志记录下来。
（3）与第（2）项类似，敌手在有效的时间窗口中能够重放一个时间戳消息，而且敌手阻止了原消息，则这种重放不能被检测。
（4）还有一种称为无更改的反向重放。这个重放是给消息发送者的，这种攻击可能出现是因为对称加密的使用，发送者很难识别发送的消息与接收的消息在内容上的不同。

防止重放攻击的方法是为每一个用于认证交互的消息附上一个序列号，新的消息只有其序列号满足适当的顺序时才会被接收。这种方法的难点是要求每一方都跟踪与其交互的通信方最新的序列号。考虑到开销问题，序列号基本上不会用于认证和密钥交换，反而是以下两种方法更常用：

- **时间戳** 只有当消息中包含一个时间戳时，A 才接收该消息。该时间戳由 A 来判断，要接近于 A 所知的当前时间。该方法要求不同参与者之间的时钟是同步的。
- **挑战/应答** A 想要一个来自 B 的新消息，首先发给 B 一个临时交互号（询问），并要求后面从 B 收到的消息（回复）包含正确的临时交互号值。

一些学者认为（如文献[LAM92a]）由于技术本身的困难，时间戳方法不能用于面向连接的应用。首先，为了维持不同处理器的时钟同步需要协议必须容错（处理网络故障）和安全（应对敌手攻击）。其次，由于任何一方的时钟机制出现错误都会造成暂时的同步消失，使得攻击成功率增大。最后，网络延迟的多变性及不可预测性，使得分布式环境下的时钟不能维持精确同步。因此，任何基于时间戳的进程，都要求有一个足够大的时间窗口来适应网络延迟，或者足够小的时间窗口来最小化攻击机会。

另一方面，挑战/应答方法不适用于无连接类型的应用，因为在无连接传输之前的握手开销，实际上是否定了无连接传输的主要优点。对这些应用，最好的方法可能是依靠安全的时间服务器和各方的一致性要求来保持时钟同步[LAM92b]。

15.1.4　单向认证

加密越来越受欢迎的一个应用领域是电子邮件服务。电子邮件的本质及主要优势就是不需要发送者和接收者同时在线，电子邮件信息会直接发到接收者的电子邮箱，直到接收者有空闲去读。

“信封”即邮件的头部必须清楚，存储转发邮件协议才能处理该消息，。如简单邮件转发协议（SMTP）或者 X.400。通常我们希望邮件处理协议不要求访问消息的明文，否则就必须要有可信的邮件处理协议。因此，邮件消息需要加密，并且邮件处理系统不拥有解密密钥。

认证的作用通常是，接收者需要确保消息确实来自于所谓的发送者。

15.2　基于对称加密的远程用户认证

15.2.1　双向认证

在第 14 章讨论过，在分布式环境中，一个两层的对称加密密钥可用于为通信提供保密性。该方案需要可信的密钥分发中心（KDC）参与，每一方和 KDC 之间都共享一个密钥（称为主密钥）。KDC 负责产生两者之间的会话密钥，并使用主密钥来保证会话密钥分发的安全。例如 15.3 节主要讨论的 Kerberos 机制。

如图 14.3 所示，说明了最初由 Needham 和 Schroeder[NEED78]提出的认证协议。该协议使用 KDC 进行密钥分发。如前所述，第 14 章包含了该认证功能。协议总结如下[①]：

（1）$A \rightarrow KDC$：$ID_A \| ID_B \| N_1$

（2）$KDC \rightarrow A$：$E(K_a,[K_s \| ID_B \| N_1 \| E(K_b,[K_s \| ID_A])])$

（3）$A \rightarrow B$：$E(K_b,[K_s \| ID_A])$

（4）$B \rightarrow A$：$E(K_s,N_2)$

（5）$A \rightarrow B$：$E(K_s,f(N_2))$

密钥 K_a、K_b 分别是 A、B 与 KDC 所共享的主密钥，协议的目的是安全地将会话密钥 K_s 分发给 A 和 B。第（2）步 A 安全地收到会话密钥，第（3）步的消息只能由 B 解密，第（4）步反映了 B 的收到的 K_s，第（4）步使 B 明确了自己与 A 拥有相同的会话密钥，且临时交互号 N_2 保证 B 得到的消息是最新的。回顾第 14 章，第（4）步和第（5）步的目的是阻止特定类型的重放攻击。需要指出的是敌手捕获第（3）步的消息并重放它，将会在某些方式上打乱 B 的操作。

尽管有第（4）步和第（5）步，该协议还是很容易受到一种形式的重放攻击。假设敌手 X 已知之前的会话密钥，虽然这比敌手简单地观察和记录第（3）步更难发生，但这是一个安全隐患。除非 B 无限期地记得所有之前和 A 会话使用过的会话密钥，否则 B 就不能确定这是一个重放攻击。如果 X 能截获第 4 步的握手消息，他就能伪造第（5）步 A 的回复并将其发送给 B，而 B 却认为该消息来自于 A 且用已认证的会话密钥加密。

Denning[DENN81,DENN82]提议通过在第（2）步和第（3）步添加时间戳来修改 Needham/Schroeder 协议，克服以上弱点。假定主密钥 K_a 和 K_b 是安全的，包含以下步骤：

（1）$A \rightarrow KDC$：$ID_A \| ID_B$

（2）$KDC \rightarrow A$：$E(K_a,[K_s \| ID_B \| T \| E(K_b,[K_s \| ID_A \| T])])$

（3）$A \rightarrow B$：$E(K_b,[K_s \| ID_A \| T])$

① 冒号左边的部分表示发送者和接收者；冒号右边的内容表示消息的内容；符号‖表示连续。

（4）$B \rightarrow A$： $E(K_s, N_1)$

（5）$A \rightarrow B$： $E(K_s, f(N_1))$

T是时间戳，为A、B的会话密钥产生的时间，A和B时效性验证需要满足：

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

其中，Δt_1是估算的KDC时钟和本地时钟（A的或者B的）的正常时间差，Δt_2是网络时延期望值。每个节点都可以按照相关标准来设置自己的时钟，因为时间戳是用安全的主密钥加密的，攻击者即便知道旧的会话密钥也不能成功，因为重放第（3）步会被B察觉。

第（4）步和第（5）步在文献[DENN81]中没有，但在文献[DENN82]添加了，这两步能确保B收到了会话密钥。

与Needham/Schroeder的协议相比，Denning协议似乎提供了较高的安全度，然而，一个新的问题出现了：即新的方案要求在整个网络中时钟是同步的。文献[GONG92]指出威胁依然存在，主要是基于这样的事实：时钟或者同步机制的破坏或者错误都可能造成分布的时钟不同步[①]。当发送者的时钟快于接收者时钟时，攻击者从发送者处截获消息之后重放，消息中的时间戳刚好能满足接受者的时间。这个重放的后果是不可预料的，Gong将该攻击称为抑制重放攻击（Suppress Replay Attack）。

一种防止抑制重放攻击的方法是，强制要求各方定期检查自己的时钟是否和KDC同步。另一种方法不需要时钟同步，依赖使用临时交互号的握手协议，该方法不容易遭受抑制重放攻击，因为接收者选择的临时交互号对发送者来说是不可预测的。Needham/Schroeder协议只依赖临时交互号，但是就像我们看到的，有其他的缺陷。

文献[KEHN92]中想要找出一种方式，能防止抑制重放攻击，同时处理Needham/Schroeder协议的问题。但是协议中的不一致引起了大家的注意，在文献[NEUM93a][②]中给出了改进方法如下所述：

（1）$A \rightarrow B$： $ID_A \| N_a$

（2）$B \rightarrow KDC$： $ID_B \| N_b \| E(K_b, [ID_A \| N_a \| T_b])$

（3）$KDC \rightarrow A$： $E(K_a, [ID_B \| N_a \| K_s \| T_b]) \| E(K_b, [ID_A \| K_s \| T]) \| N_b$

（4）$A \rightarrow B$： $E(K_b, [ID_A \| K_s \| T]) \| E(K_s, N_b)$

协议的步骤如下：

（1）A产生临时交互号N_a和自己的标识一起以明文的形式发给B，该临时交互号会和会话密钥共同在加密消息中返回给A，确保A的时效性。

（2）B向KDC请求一个会话密钥，向KDC发送的消息包含自己的标识和临时交互号N_b，其中临时交互号和会话密钥共同在加密消息中返回给B，确保B的时效性。B发给KDC的消息还包含一个用B和KDC的共享主密钥加密的加密块，用于指示KDC需要发布证书给A；内容包括证书的接受人，证书的截止时间和来自于A的临时交互号。

（3）KDC发送B的临时交互号以及用KDC与B共享的主密钥加密的数据给A，该块作为“票据”用于A之后的认证。KDC也发送给A加密块，该块是用KDC与A共享的主密钥加密的，该块可用于明确B收到了A的原始消息ID_A且是一个没有重放（N_a）及时消息，也提供给A一个会话密钥K_s及其使用时间限制（T_b）。

① 这样的事情确实可能发生。近些年，有缺陷的芯片被用于大量的计算机和其他电子系统以跟踪时间和日期。这些芯片总一天会被更换掉[NEUM90]。

② 构造正确的协议的确很困难。

（4）A 把票据及 B 的临时交互号一起传递给 B，其中临时交互号使用会话密钥加密。票据提供给 B 解密 $E(K_s, N_b)$ 的密钥以恢复临时交互号。事实上，B 的临时交互号用会话密钥加密证明了该消息不是重放的。

该协议为 A、B 获得会话密钥提供了一个有效且安全的方法，此外，协议使 A 拥有认证 B 的密钥，且避免了重复的联系认证服务器。假设 A 和 B 利用协议建立和结束了一次会话，那么协议设置的限制时间内，A 想要和 B 再一次建立新的会话，便可以按以下协议执行：

（1）$A \rightarrow B$：$E(K_b, [ID_A \| K_s \| T_b]) \| N'_a$

（2）$B \rightarrow A$：$N'_b \| E(K_s, N'_a)$

（3）$A \rightarrow B$：$E(K_s, N'_b)$

当 B 接收到第一步的消息时，验证时间戳是否过期。新产生的临时交互号 N'_a 和 N'_b 确认双方都不会受到重放攻击。

T_b 所指定的时间只和 B 的时钟相关的，因为只有 B 检查该时间戳，因此不要求时钟同步。

15.2.2　单向认证

在如图 14.5 所示的分布式密钥分发环境中，使用对称加密是不切实际的。该方法要求发送者首先向接收者发起会话请求，等收到包含会话密钥的回复之后才能发送消息。

对加密的电子邮件来说，图 14.3 中描述的 KDC 策略是个不错的选择，但为了避免要求接收者（B）和发送者（A）同时在线，第（4）步和第（5）步需要删除。对内容为 M 的消息，步骤如下：

（1）$A \rightarrow KDC$：$ID_A \| ID_B \| N_1$

（2）$KDC \rightarrow A$：$E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$

（3）$A \rightarrow B$：$E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

该方法可以保证只有消息指定的接收者才能读取消息，同时提供了对发送者 A 的认证。但该协议不能抵抗重放攻击，已提供的防御策略中包括在消息中添加时间戳，但是由于电子邮件发送存在潜在延误，时间戳可能会限制邮件的发送。

15.3　Kerberos

Kerberos[①]是作为 MIT 大学的 Athena 计划的一部分而被开发的认证服务。Kerberos 阐述了这样一个问题：假设有一个开放的分布式环境，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，从该机上发送伪造的请求。
- 用户可以监听信息或使用重放攻击，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，

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。

Kerberos 通过提供一个集中的授权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 4120 和 RFC 4121）[①]。

本节首先简要介绍 Kerberos 的设计动机。由于 Kerberos 的复杂性，通过版本 4 中的认证协议来理解 Kerberos 策略的本质，而不用考虑掌握小的安全威胁细节。最后，阐述版本 5。

15.3.1 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都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，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）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，相信客户端系统提供的用户身份。
（3）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[②]。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，使用分布式服务器结构，即一个系统可以备份其他系统。
- **透明性** 理想状况下，用户除了要输入口令外，不需要知道认证发生的地点。
- **可伸缩性** 系统应能支持大量的客户端和服务器，这需要具有模块化、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。该认证基于 Needham 和 Schroeder 提出的协议[NEED78]（在 15.2 节中讨论过），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[③]。

---

① 版本 1 到版本 3 是内部开发版本，版本 4 才是“原始的” Kerbersos 版本。

② 然而，即使是封闭环境也面临怀有不满情绪的雇员的攻击。

③ 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。欧律斯透斯命令大力神捕捉 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。欧律斯透斯看见 Kerberos 的三个头流着口水，惊讶地发现大力神还活着。由于惧怕他的智慧，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。

15.3.2　版本 4

Kerberos 的版本 4 在协议中使用了 DES 提供认证服务，但从整体上很难看出使用 DES 的必要性。因此，使用 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。

检查完该协议，我们来看看版本 4 的一些其他方面。

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。服务器必须在每次客户/服务器交互中进行认证，但在一个开放式的环境中，将给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于一个集中式数据库，并且 AS 与每一个服务器共享一个唯一的密钥，密钥按物理方法或其他安全方法分发。考虑以下包含 3 个消息的会话：

$$(1)\mathrm{C}\rightarrow\mathrm{AS}: \quad \mathrm{ID}_C \| P_C \| \mathrm{ID}_V$$

$$(2)\mathrm{AS}\rightarrow\mathrm{C}: \quad \mathrm{Ticket}$$

$$(3)\mathrm{C}\rightarrow\mathrm{V}: \quad \mathrm{ID}_C \| \mathrm{Ticket}$$

$$\mathrm{Ticket} = \mathrm{E}(K_v, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_V])$$

其中，C 为客户端；AS 为认证服务器；V 为应用服务器；ID_C 为 C 上的用户标识；ID_V 为 V 的标识；P_C 为 C 上的用户口令；AD_C 为 C 的网络地址；K_v 为 AS 与 V 共享的加密密钥。

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的解密该票据；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，可以用 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，则从生成票据的工作站发出的票据才是合法的。

一个更加安全的认证会话　虽然上述会话解决了开放网络环境中认证的一些问题，但还存在两个突出问题。首先，我们可能希望能使用户输入口令的次数最小化。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，希望察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 想在一天中多次查询邮件，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) $C \rightarrow \mathrm{AS}: \mathrm{ID}_C \| \mathrm{ID}_{\mathrm{tgs}}$

(2) $\mathrm{AS} \rightarrow C: \mathrm{E}(K_c, \mathrm{Ticket}_{\mathrm{tgs}})$

每种服务类型仅一次：

(3) $C \rightarrow \mathrm{TGS}: \mathrm{ID}_C \| \mathrm{ID}_V \| \mathrm{Ticket}_{\mathrm{tgs}}$

(4) $\mathrm{TGS} \rightarrow C: \mathrm{Ticket}_v$

每次服务会话仅一次：

(5) $C \rightarrow V: \mathrm{ID}_C \| \mathrm{Ticket}_v$

$\mathrm{Ticket}_{\mathrm{tgs}} = \mathrm{E}(K_{\mathrm{tgs}}, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_{\mathrm{tgs}} \| \mathrm{TS}_1 \| \mathrm{Lifetime}_1])$

$\mathrm{Ticket}_v = \mathrm{E}(K_v, [\mathrm{ID}_C \| \mathrm{AD}_C \| \mathrm{ID}_v \| \mathrm{TS}_2 \| \mathrm{Lifetime}_2])$

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 $\mathrm{Ticket}_{\mathrm{tgs}}$，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（表明申请一个的 TGS 服务），申请得到票据授权票据。

（2）AS 利用对应用户口令的密钥加密票据，并将其返回给客户端（K_c），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（Lifetime），这样，客户端既可以重用票据，又不需要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）和步骤（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 和网络地址是否与消息来源一致。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_V 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（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，则可获得相应的服务。

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）必须能够证实票据的使用者与票据的所有者的一致性。

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。假冒的服务器即可伪装成真正的服务器捕获用户信息而对用户提供虚假服务。

我们将逐个讨论这些问题，表 15.1 描述了实际 Kerberos 协议。图 15.2 展示了简化了的概览。

表 15.1　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→AS:　$ID_C \| ID_{tgs} \| TS_1$

（2）AS→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

（1）C→TGS: $ID_V \| Ticket_{tgs} \| Authenticator_c$

（2）TGS→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

（1）C→K：$Ticket_v \| Authenticator_c$

（2）K→C：$E(K_{c,v}, [TS_5+1])$ (对所有认证)

$Ticket_V = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_V \| TS_4 \| Lifetime_4])$

$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$

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 15.1(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。

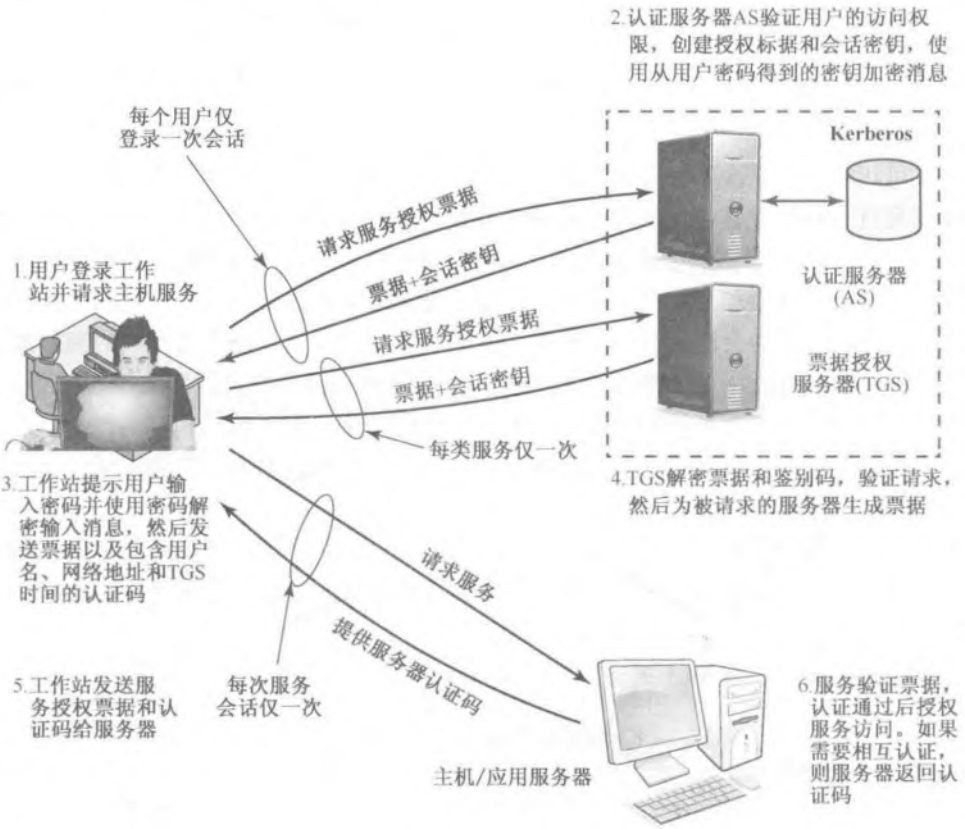


图 15.2 Kerberos 概览

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 知道该消息是新鲜的；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 15.1(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 K_{tgs} 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，因此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 15.1 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。

图 15.3 描述了 Kerberos 中各方的信息交互。表 15.2 总结了 Kerberos 协议各元素的合理性。

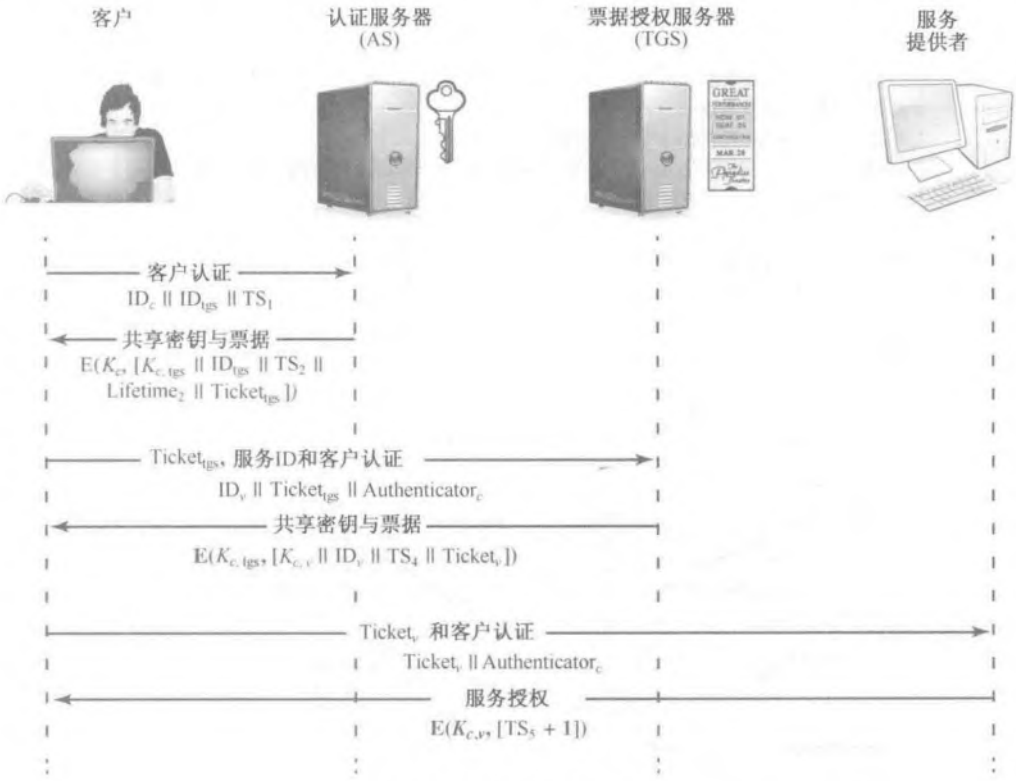


图 15.3　Kerberos 交换

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密钥，所有应用服务器必须在 Kerberos 服务器注册。

表 15.2　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

（续表）

(a) 认证服务交换	
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
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
$K_{c,tgs}$	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
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 确保此票据所有者与票据发放时的所有者相同，用短生命期防止重用
$K_{c,tgs}$	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_3	通知 TGS 认证消息的生成时间
(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：客户端认证服务器
$K_{c,v}$	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
K_v	用 TGS 与服务器共享的密钥加密的票据，防止伪造
$K_{c,v}$	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所有者与票据发放时的所有者相同，用短生命期防止重用
$K_{c,v}$	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

这种环境被称为一个 Kerberos 域。Kerberos 域是一组受管节点，它们共享同一 Kerberos 数据库。Kerberos 数据库驻留在 Kerberos 主控计算机系统上，该计算机系统应位于物理上安全的房间内。Kerberos 数据库的只读副本也可以驻留在其他 Kerberos 计算机系统上。但是，对数据库的所有更改都必须在主控计算机系统进行。更改或访问 Kerberos 数据库要求有 Kerberos 主控密码。这里还有一个概念就是 Kerberos 主体。Kerberos 主体是 Kerberos 系统知道的服务或用户。每个 Kerberos 主体通过主体名称进行标识。主体名称由三部分组成：服务或用户名称、实例名称以及域名。

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 服务器应和其他域的服务器共享一个密钥，两个的 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，可以用图 15.4 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。

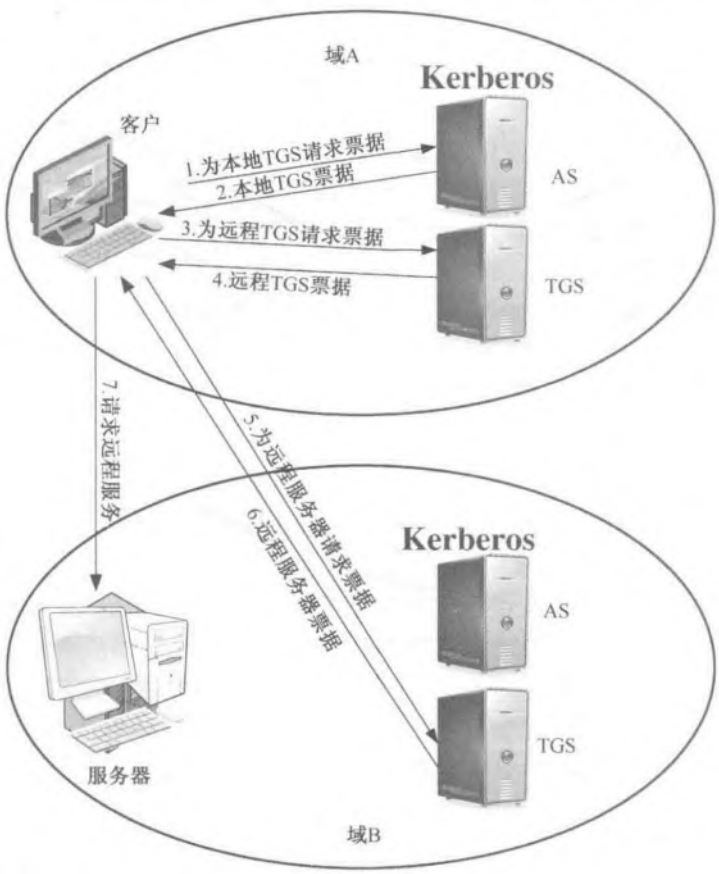


图 15.4　其他域的服务请求

图 15.4 中交换的消息细节如下所述（与表 15.1 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→C: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$

(7)C→V_{rem}: $Ticket_{vrem} \| Authenticator_c$

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。

15.3.3 Kerberos 版本 5

Kerberos 版本 5 在 RFC4120 中有详细描述，介绍了版本 4 的一些改进[KOHL94]。在此，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 提供了这项功能。

（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。

① 下面的讨论参照文献[KOHL94]。

除了这些环境限制，版本 4 中还存在一些技术不足。大多数不足在文献[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 15.1 中，提供给客户端的票据需要经过两次加密［消息（2）和消息（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，而第二次加密并不是必需的。

（2）**PCBC 加密**　版本 4 加密所使用 DES 的非标准模式 PCBC（Propagating Cipher Block Chaining）[①]，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，并能够用标准的 CBC 模式加密。特别地，一份消息的校验码或 Hash 特征码优先用 CBC 加密。

（3）**会话密钥**　每个票据包含一个会话密钥，用于客户端加密认证信息，并与票据一起送往服务器。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[②]，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 21 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 15.3 描述了版本 5 的基本会话，可以与版本 4 的表 15.1 进行对比。

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的过程。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。

表 15.3　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：$\text{Options} \| ID_C \| \text{Realm}_c \| ID_{tgs} \| \text{Times} \| \text{Nonce}_1$
（2）AS→C：$\text{Realm}_C \| ID_C \| \text{Ticket}_{tgs} \| E(K_{c,tgs}[K_{c,v} \| \text{Times} \| \text{Nonce}_1 \| \text{Realm}_{tgs} \| ID_{tgs}])$ $\text{Ticket}_{tgs}=E(K_{tgs},[\text{Flags} \| K_{c,tgs} \| \text{Realm}_c \| ID_C \| AD_C \| \text{Times}])$
（b）服务授权票据交换：获取服务授权票据
（3）C→TGS：$\text{Options} \| ID_V \| \text{Times} \| \text{Nonce}_2 \| \text{Ticket}_{tgs} \| \text{Authenticator}_c$
（4）TGS→C：$\text{Realm}_c \| ID_C \| \text{Ticket}_v \| E(K_{c,tgs},[K_{c,v} \| \text{Times} \| \text{Nonce}_2 \| \text{Realm}_v \| ID_V])$ $\text{Ticket}_{tgs}=E(K_{tgs},[\text{Flags} \| K_{c,tgs} \| \text{Realm}_c \| ID_C \| AD_C \| \text{Times}])$ $\text{Ticket}_v=E(K_v,[\text{Flags} \| K_{c,v} \| \text{Realm}_c \| ID_C \| AD_C \| \text{Times}])$ $\text{Authenticator}_c=E(K_{c,tgs},[ID_C \| \text{Realm}_c \| TS_1])$

① 这将在附录 T 中介绍。

② 附录 T 将讲述口令与加密密钥的映射。

（续表）

（c）客户/服务器认证交换：获取服务
（5）C→V：Options ‖ $Ticket_v$ ‖ $Authenticator_c$
（6）V→C：$E_{K_{C,V}}$[TS_2 ‖Subkey‖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#]

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。该数据块包含在客户端和 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：

- **子密钥** 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 $K_{c,v}$。
- **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者处得到的时间戳。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 15.4 总结了票据中可能包含的标志。

表 15.4 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	初始认证协议要求使用带名客户端独占的硬件资源
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是基于票据授权票据
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，这是唯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，他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（smart card）生成不断变化的口令，将其包含在预认证消息中。卡所生成的口令是基于用户口令，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，主要开销将在于获取新票据，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票据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]。

在有限条件下，代理是一个强大的转发程序。如果票据被设置了 FORWARDABLE 标志，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 都包含与其他域中的 Kerberos 服务器共享密钥的前提下，可以访问其他域的服务器。例如，各个域有层次结构时，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。

15.4　基于非对称加密的远程用户认证

15.4.1　双向认证

在 14 章中，介绍了将公钥加密应用于会话密钥分发的方案（如图 14.9 所示），该协议假定双方都拥有对方最新的公钥，但这个假定恐怕不能被满足。

文献[DENN81]提出使用时间戳的协议：

① 一些对代理可能应用的讨论参见文献[NEUM93b]。

（1）$A \rightarrow AS$： $ID_A || ID_B$

（2）$AS \rightarrow A$： $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T])$

（3）$A \rightarrow B$： $E(PR_{as}, [ID_A \| PU_a \| T]) \| E(PR_{as}, [ID_B \| PU_b \| T]) \| E(PU_b, E(PR_a, [K_s \| T]))$

这种情况下，涉及的中心系统作为一个认证服务器（AS），因为事实上它并不负责密钥分发，AS提供公钥证书。会话密钥由A选择并加密，因此，AS不会知道，时间戳能防止窃取密钥的重放攻击。

该协议很简洁但要求时钟同步。另一个是Woo和Lam[WOO92a]提出的使用临时交互号的协议，包括以下几步：

（1）$A \rightarrow KDC$： $ID_A \| ID_B$

（2）$KDC \rightarrow A$： $E(PR_{auth}, [ID_B \| PU_b])$

（3）$A \rightarrow B$： $E(PU_b, [N_a \| ID_A])$

（4）$B \rightarrow KDC$： $ID_A \| ID_B \| E(PU_{auth}, N_a)$

（5）$KDC \rightarrow B$： $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_B]))$

（6）$B \rightarrow A$： $E(PU_a, [E(PR_{auth}, [N_a \| K_s \| ID_B]) \| N_b])$

（7）$A \rightarrow B$： $E(K_s, N_b)$

A通知KDC它想要与B建立安全的连接［第（1）步］，KDC返回B的公钥证书给A［第（2）步］，使用B的公钥，A发送临时交互号N_a给B，表示想要和B建立通信［第（3）步］。在第（4）步中，B向KDC请求A的公钥证书以及会话密钥，因为B将A的临时交互号也发给了KDC（该临时交互号用KDC的公钥加密），所以KDC可以用该临时交互号来标记会话密钥。KDC将A的公钥证书和信息$\{N_a, K_s, ID_B\}$返回给B［第（5）步］，该信息指出K_s由KDC产生，能代表B并与N_a相关联，K_s和N_a的捆绑能使A确定K_s是最新的，用KDC的私钥加密允许B验证信息来自KDC，同时使用B的公钥加密是为了防止其他想和A建立不正当连接的实体使用。第（6）步，将用KDC的私钥加密的$\{N_a, K_s, ID_B\}$和B的临时交互号N_b一起转发给A。A得到会话密钥K_s后用其加密N_b返回给B，最后的消息能确保B知道A已经获知会话密钥。

这似乎是一个考虑了各种攻击的安全协议，然而，作者自己找出了缺陷，并在文献[WOO92b]中做了修正：

（1）$A \rightarrow KDC$： $ID_A \| ID_B$

（2）$KDC \rightarrow A$： $E(PR_{auth}, [ID_B \| PU_b])$

（3）$A \rightarrow B$： $E(PU_b, [N_a \| ID_A])$

（4）$B \rightarrow KDC$： $ID_A \| ID_B \| E(PU_{auth}, N_a)$

（5）$KDC \rightarrow B$： $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]))$

（6）$B \rightarrow A$： $E(PU_a, [E(PR_{auth}, [N_a \| K_s \| ID_A \| ID_B]) \| N_b])$

（7）$A \rightarrow B$： $E(K_s, N_b)$

在第（5）步和第（6）步中，A的标识ID_A被加入了用KDC加密的项目集，这将会话密钥K_s和双方标识捆绑在一起。ID_A的加入说明了临时交互号N_a在A产生的所有临时交互号中是唯一的，但并非在双方产生的临时交互号中都是唯一的，即$\{ID_A, N_a\}$才能唯一的标识连接请求来自于A。

该例和之前描述的协议可以说明，看起来安全的协议也需要不断的分析修正。这些例子强调了在认证领域中想要完美是很困难的。

15.4.2　单向认证

我们已经介绍了适合电子邮件的公钥加密方法，对全部信息的直接加密包括为了保密［如图 12.1(b)所示］、认证［如图 12.1(c)所示］或者两者都有［如图 12.1(d)所示］。这些方法要求或者发送者已知接收者的公钥（保密），或者接收者知道发送者的公钥（认证），或者两者皆有（保密和认证）。另外，对可能很长的消息来说，公钥算法必须执行一次或者两次。

如果保密性是首先要考虑的，以下方法可能更有效：

$$\mathrm{A} \rightarrow \mathrm{B}: \ \mathrm{E}(\mathrm{PU}_b, K_s) \| \mathrm{E}(K_s, M)$$

此时，消息用会话密钥加密，A 用 B 的公钥加密会话密钥，只有 B 能解密得到会话密钥然后使用该密钥解密消息，该方法比只用 B 的公钥简单地加密全部消息更有效。

如果主要考虑认证，则在图 13.2 中描述的数字签名就可以满足：

$$\mathrm{A} \rightarrow \mathrm{B}: \ M \| \mathrm{E}(\mathrm{PR}_a, H(M))$$

该方法可以保证 A 事后不能否认发送过消息，但该技术对其他敌手也是公开的。Bod 写了一封包含一个能节省公司钱财的办法的邮件给他的老板 Alice，附上自己的签名后发送进电子邮件系统，最后该消息会被传送到 Alice 的邮箱。但是，假定 Max 听说了 Bob 的想法，在邮件传送之前访问了邮件列表，找到 Bob 的邮件、去掉其签名并换成自己的，后将邮件传送给 Alice，最终 Max 会得到该想法的所有权。

为了防止这种情况，消息和签名都要用接收者公钥加密：

$$\mathrm{A} \rightarrow \mathrm{B}: \ \mathrm{E}(\mathrm{PU}_b, [M \| \mathrm{E}(\mathrm{PR}_a, H(M))])$$

后两种方案要求 B 知道 A 的公钥并且是新鲜的，用第 14 章介绍的数字证书来保证。则有

$$\mathrm{A} \rightarrow \mathrm{B}: \ M \| \mathrm{E}(\mathrm{PR}_a, H(M)) \| \mathrm{E}(\mathrm{PR}_{as}, [T \| \mathrm{ID}_A \| \mathrm{PU}_a])$$

A 发送给 B 的信息包括：消息 M、A 对消息的签名和用认证服务器私钥加密的 A 的证书。消息接收者首先使用证书得到 A 的公钥并验证其真实性，然后用该公钥来验证消息 M。如果同时要求保密，则全部的消息都要使用 B 的公钥加密，或者，用会话密钥加密全部消息，该会话密钥用 B 的公钥加密后传送给 A。这种方法将在第 19 章介绍。

15.5　联合身份管理

在多个企业以及大量应用的身份管理方案中，联合身份管理是一个相对较新的概念，支持数万甚至数十万的用户。首先介绍身份管理的概念，然后介绍联合身份管理。

15.5.1　身份管理

身份管理是一种集中式的、自动的方法，提供雇员或者其他授权的个人对资源拥有企业范围的访问。身份管理的重点是为用户（人或者进程）定义一个身份，为该身份关联属性，并完成用户的身份认证。身份管理系统的主要的概念是 SSO（Single Sign-On），SSO 能使用户在一次认证之后访问所有网络资源。

联合身份管理系统提供的典型服务如下：

- **接触点（Point of contact）**　包括用户身份的认证和用户/服务器会话的管理。

- **SSO 协议服务**（SSO protocol service） 提供了中立的供应商安全令牌服务，用以支持单点登录的联合服务。
- **信任服务**（Trust service） 联合关系要求商务伙伴之间有一个基于信任关系的联合。信任关系可以表示为：使用组合的安全令牌交换用户信息，其中，令牌中的信息被加密以保证安全，而且有选择地将身份映射规则应用到令牌所包含的信息中。
- **密钥服务**（Key service） 管理密钥和证书。
- **身份服务**（Identity service） 服务提供了本地存储数据的接口，包括用户登记系统和数据库，用于对户身份相关信息的管理。
- **授权**（Authorization） 基于认证的特定服务和资源的授权访问。
- **规定**（Provisioning） 包括在每一目标系统中为用户创建账户，登记或注册用户账户，建立确保账户数据的保密性和完整性的访问权限或证书。
- **管理**（Management） 与服务相关的运行时的配置和部署。

注意，Kerberos 包含许多身份管理系统的要素。

图 15.5 表明了通用身份管理结构中的实体和数据流。一个当事人（principal）就是一个身份持有者，也就是一个想要访问网络资源和服务的人（用户），用户设施、代理程序和服务器系统都可能作为当事人。当事人向身份提供者证明自己，身份提供者为当事人关联认证信息，如属性、一个或者多个身份标识符。

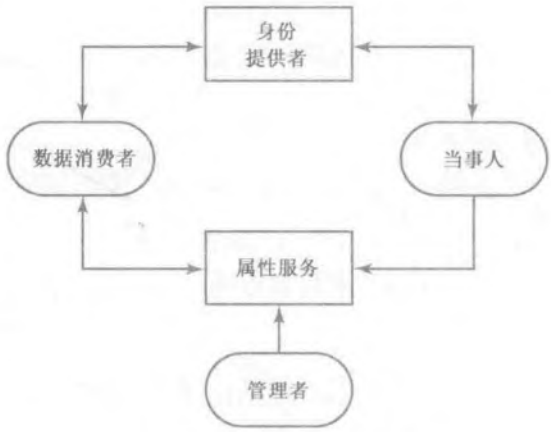


图 15.5 通用的身份管理架构

如今，数字身份包含的属性已不只是简单的标识符和认证信息（如口令和生物计量信息）了。属性服务管理这些属性的创建和维护，如网购用户每次下订单的时候都要提供一个运送地址给网店店主，若用户迁居时该信息需要修改。身份管理使得用户一次性提供这些信息并将其保存，在满足授权和隐私策略时发布给数据消费者。用户可以自己创建一些属性关联到其身份，如地址。管理者也可为用户指派属性，如角色、访问权限和雇员信息。

数据消费者是一些实体，能够得到并使用由身份和属性提供者提供的数据，数据消费者常被用于支持授权决定和收集审计信息，如数据库服务器或者文件服务器，需要客户的证书来决定为该客户提供什么样的访问权限。

15.5.2 身份联合

在本质上，身份联合是身份管理在多个安全域上的一个扩展，这些域包括自治的内部商业单元、外部的商业伙伴以及其他第三方的应用和服务。目的是提供数字身份共享，使得用户只需要一次认证就可以访问多个域的应用和资源。这些域是相对自治或者独立的，因此可以采用非集中化控制，当然，合作的组织之间必须形成一个基于协商和相互信任的标准来安全地共享数字身份。

联合身份管理涉及协议、标准和技术，支持数万用户的身份、身份属性和访问权限在多个企业以及大量的应用间移植。当多个组织执行联合身份管理时，一个组织的雇员能够使用 SSO 访问联盟中其他组织的服务，如一个雇员登录本公司的内部网，经过认证可以在内部网执行授权的功能、访问授权的服务，之后可以从外部网的卫生保健提供者处访问自己的服务，而不需要重新认证。

除了 SSO，联合身份管理也提供了其他的功能，如属性的标准化方式。渐渐地，数字身份包含的属性已经不只是简单的标识符和认证信息（如密码和生物计量信息）了，属性的例子包括账号、组织的角色、物理定位和文件所有权。一个用户可能有多个身份标识符，每个标识符可能关联唯一的角色，拥有自己的访问权限。

联合身份管理的另一个主要功能是身份映射，不同的安全域可能表示的身份和属性不同，而且，一个人在一个域中的信息总量可能多于其在另一个域的必需的信息总量。联合身份管理协议将一个域中用户的身份和属性映射到另一个域。

如图 15.6 所示，给出了通用联合身份管理结构中的实体和数据流。

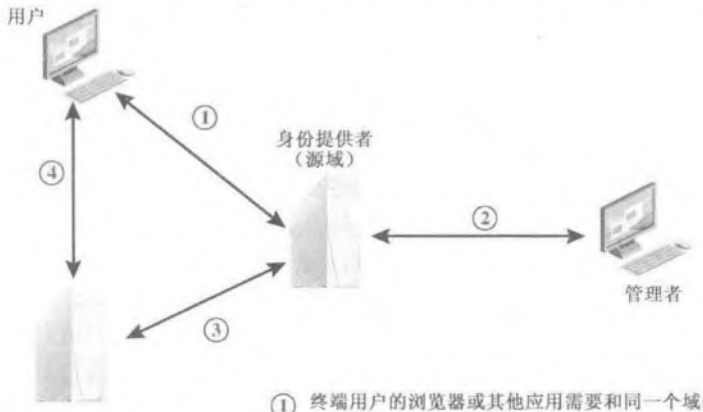


图 15.6　联合身份操作

身份提供者通过和用户以及管理者会话、协议交换来获得属性信息，如网购用户每次下订单的时候都要提供一个运送地址给网店店主，若用户迁居时需要修改该信息。身份管理使得用户一次性提供这些信息并将其保存，在满足授权和隐私策略时发布给数据消费者。

服务提供者是一些实体，能够得到和使用由身份和属性提供者维持和提供的数据，数据消费者常被用于支持授权决定和收集审计信息，如数据库服务器或者文件服务器，需要客户的证书来决定为该客户提供什么样的访问权限。一个服务提供者可以和用户以及身份提供者在同一个域，但联合身份管理中服务提供者和用户在不同的域。

标准　为了在不同域或多个系统之间进行安全的身份交换，联合身份管理使用很多标准来构建模块。事实上，组织可以为用户发布一些安全且可被合作伙伴处理的票据。身份联合标准关心的是如何定义这些票据包括内容和格式，如何为交换票据和执行管理任务提供协议。这些任务包括构建系统来执行属性传递和身份映射，以及执行登录和审计功能。下面列出了一些关键的标准：

- **可扩展标记语言（XML）**　一种标记语言，采用嵌入的标签集来描述文档中的文本元素以表明其外观、功能、含义或上下文。XML 文档类似 HTML 文档（一种可视化的网页语言），但是它提供了更强大的功能。XML 包含了对每个字段数据类型的严格定义，因此它支持数据库格式和语义。XML 还为传输和更新数据对象的命令提供了编码规则。
- **简单对象访问协议（SOAP）**　规定使用 HTTP 来发送 XML 格式的信息。应用程序之间的

服务请求和响应都采用基于 XML 的数据格式。其中，XML 定义了数据对象和结构，而 SOAP 则提供了一种交换数据对象和远程调用相关数据对象的方式，文献[ROS06]有其详细论述。

- **Web 服务安全（WS-Security）** 一组对 SOAP 扩展的协议，用于实现 Web 服务中消息的完整性与机密性。为了在应用程序间安全地交换 SOAP 消息，Web 服务安全为每一个消息分配了用于身份认证的令牌。
- **安全声明标记语言（SAML）** 一种基于 XML 标准的语言，用于在线合作伙伴之间安全信息的交换。SAML 以主题声明的形式传输认证信息，而声明是对认证实体发布的主题的描述。

联合身份管理的挑战是整合多种技术、标准和服务来提供一个安全的对用户友好的终端，关键是依靠一些被工业界广泛接受的成熟标准。联合身份管理似乎已经达到了成熟水平。

例子 为了对身份联合有更好的认识，我们来看三种应用场景[COMP06]。

第一种方案 [如图 15.7(a)所示]，Workplace.com 联系 Health.com 为雇员提供健康福利，一个雇员使用网站接口打开 Workplace.com，通过一个认证程序访问 Workplace.com 中授权的服务和资源，当雇员点击链接查看健康福利时，她的浏览器将会用安全的方式改变方向到 Health.com。这两个组织之间是合作交换用户的联合标识符，Health.com 维持 Workplace.com 中每个雇员的用户身份，并为其关联健康福利信息以及访问权限。在这个例子中，两个公司之间的链接是基于账户信息，用户参与是基于浏览器的。

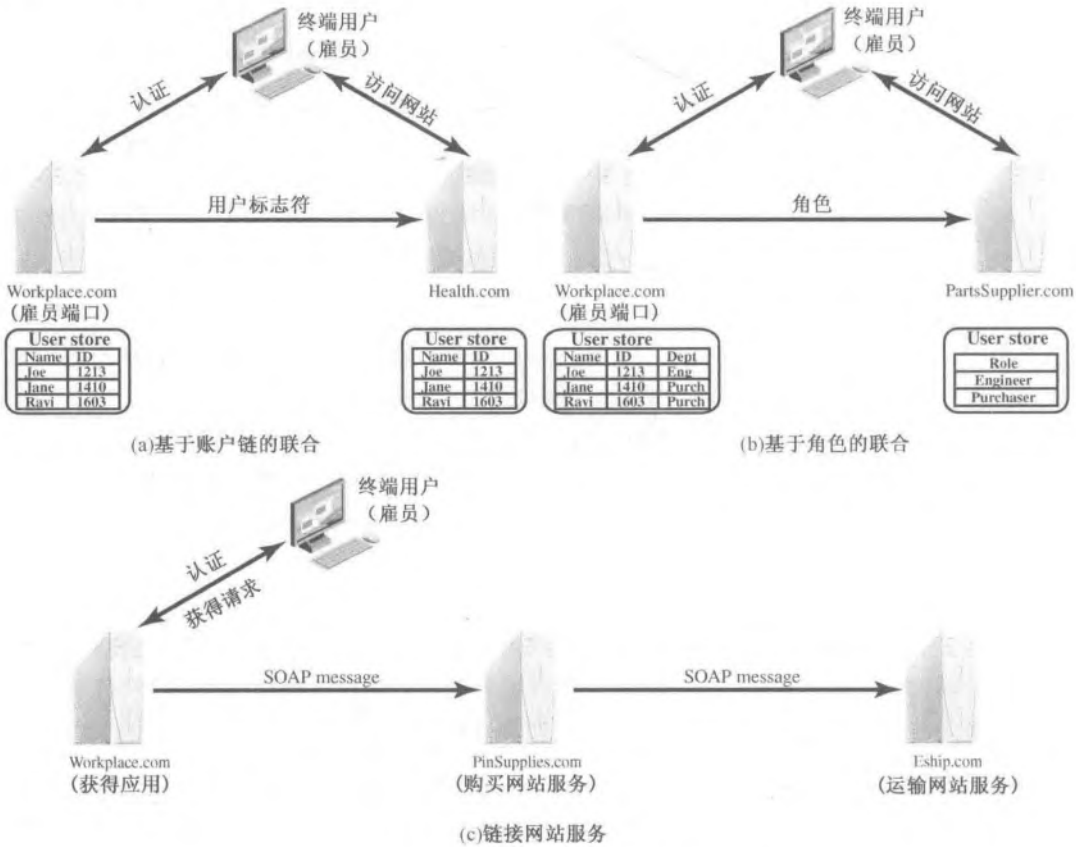


图 15.7 联合身份管理实例

如图 15.7(b)所示展示了基于浏览器的第二种方案。PartsSupplier.com 是 Workplace.com 的“零件”供应商，这种情况下，对信息的访问使用基于角色的访问控制（RBAC）。在 Workplace.com 的

员工入口处有一个工程师认证且点击链接可以访问 PartsSupplier.com 的信息，因为已认证用户是一个工程师，所以直接被带去 PartsSupplier.com 网页的技术文件和故障处理部分，而不需要再次认证；同样地，一个负责采购的雇员经认证登录 Workplace.com 后到 PartsSupplier.com 购买东西，也不再需要得到 PartsSupplier.com 的认证。这种情况下，PartsSupplier.com 没有 Workplace.com 单个雇员的身份信息，当然，联合双方的链接是基于角色的。

如图 15.7(c)所示的情景是基于文件的，而不是基于浏览器的。在第三个例子中，Workplace.com 和 PinSupplies.com 有购买协议，且后者和 E-Ship.com 有商业关系。一个 Workplace.com 的雇员登录并认证购买后，雇员可得到供应商名单和可订购的部分，用户点击 PinSupplies 按钮会出现一个订购网页（HTML 页面），然后填写表单并提交。采购程序会产生一个 XML/SOAP 文件，并将该文件插入一个基于 XML 的消息的封装体中，同时将用户的信誉以及 Workplace.com 组织的身份插入消息的封装头部，然后将该消息投递给 PinSupplies.com 的网站购买服务。该服务认证消息并处理请求，然后发送 SOAP 消息给他的运送伙伴来满足此次订购，该消息包括封装头部 PinSupplies.com 的安全令牌以及封装体中的发送项目列表和用户配送信息。运送网站认证请求并处理运送任务。

15.6　个人身份验证

基于拥有智能卡的用户认证变得越来越普遍。智能卡的外表类似信用卡，有着电子接口，可能还使用了各种认证协议。

智能卡中包含了一整块微处理器，它由中央处理器、存储器和 I/O 端口所构成。为了进行加密操作有些微处理还含有特殊的协处理电路，从而能加快完成编码与解码消息、生成数字签名验证传输信息这样的任务。在某些智能卡中，I/O 端口是通过裸露的电接触的方式被兼容的读卡器直接读取的。另外一些卡则依赖于嵌入式天线与读卡器进行无线通信。

一个典型的智能卡包括三种类型的存储器。只读存储器（ROM）存储了卡的生命周期内不改变的数据，例如卡号和持卡人的姓名；电可擦除可编程只读存储器（EEPROM）保存了应用数据和程序，如智能卡能执行的协议。它还可以保存随时间不断变化的数据。例如，在电话卡中，EEPROM 保存了剩余的通话时间；随机存储器（RAM）保存了应用程序在执行过程中产生的临时数据。

就智能卡认证的实际应用而言，众多的供应商必须遵守一定的标准，这些标准覆盖了智能卡协议，认证与访问控制的格式与协议，数据库条目，信息的格式，等等。在制定出的标准方面，最重要的一步就是 FIPS 201-2［Personal Identity Verification (PIV) of Federal Employees and Contractors, June 2012］。该标准定义了一个可靠的、政府范围内的 PIV 系统，此系统应用于对像联邦政府控制的设备和信息系统的访问。在 FIPS 201-2 标准描述的 PIV 系统中，通用的身份证件可以被创建，并在接下来利用它来验证用户的身份。该标准还确定了联邦政府对安全级别的广泛需求，这些安全级别依赖于正在受到保护设备和系统遇到的风险。该标准也适用于私营承包商，并可作为任何组织的一个准则。

15.6.1　PIV 系统模型

图 15.8 描述了 FIPS 201-2 兼容系统的主要组件。PIV 前端为请求访问设备的用户定义了物理接口，这些用户可能要物理访问一个受保护的物理区域，或者要逻辑访问一个信息系统。PIV 前端子系统支持三重认证；认证的重数依赖于安全级别的要求。前端使用了被称为 PIV 卡的智能卡，这种卡是一种双界面接触式或非接触式卡，它保存有 持卡人的照片、X.509 证书、加密密钥、生物特征数据和持卡人的唯一身份标识（CHUID）。某些持卡人信息可能是读保护的，需要输入个人身份证号才能被读卡器读取。在当前的标准中，生物特征读卡器就是指纹读卡器或者虹膜扫描器。

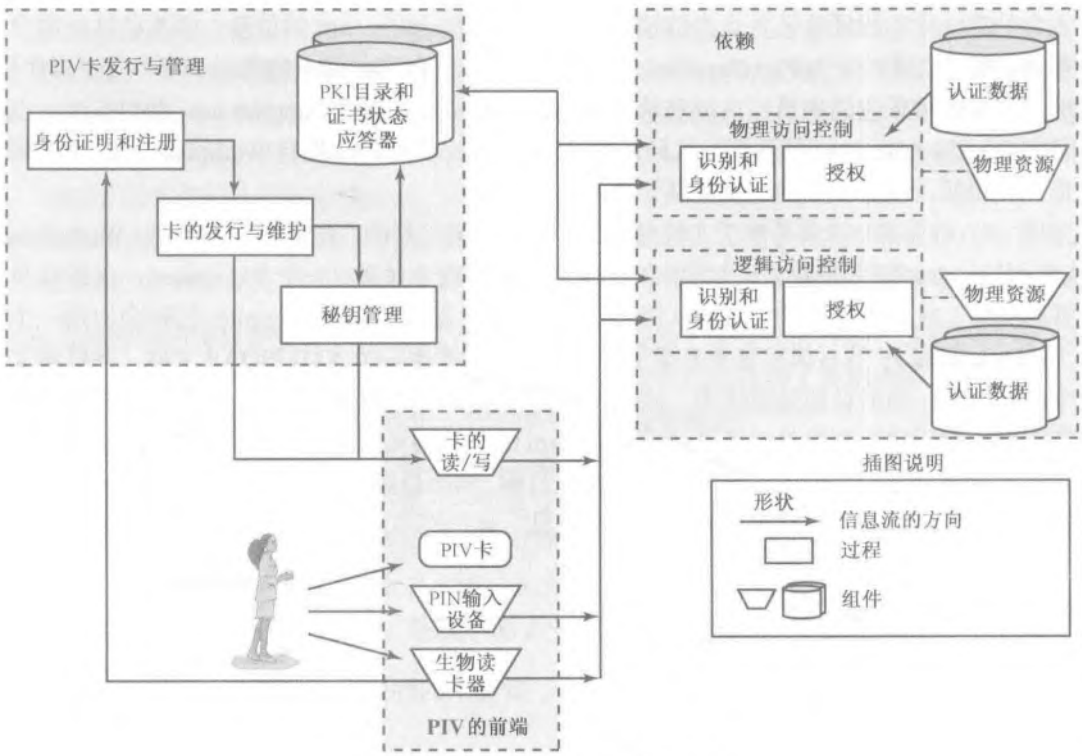


图 15.8 FIPS 201 PIV 系统模型

FIPS 201-2 标准定义了三个层次的保障级别来验证卡和存储在卡中编码的数据，这反过来又验证证件持有人的真实性。该标准定义的三个级别分别是：某些信任级别，该级别与读卡器和 PIN 的使用是相一致的；高度信任级别，该级别增加了生物特征的比较。比较的双方是在卡发行过程中获取并编码的指纹与物理访问点扫描到的指纹；非常高的信任级别，这个级别要求上述描述的过程是在控制点由正式的检查人员参与完成的。

PIV 系统的另外一个主要的组件是 PIV 卡发行与管理子系统。这个子系统包含的组件负责用户身份的证明与注册，卡和密钥的发行与管理，以及作为验证基础设施一部分所需的各种资源和服务［如公钥基础设施（PKI）目录，证书状态服务］的提供。

PIV 系统同依赖子系统进行交互，该子系统包含的组件负责确定特定的 PIV 持卡人对物理或逻辑资源的访问。FIPS 201-2 还对用于 PIV 系统和依赖系统之间的交互的数据格式和协议进行标准化。

与大多数访问控制卡中典型的卡号码/设备编码不同，FIPS 201 通过使用截止日期（持卡人身份唯一标识的数据字段）和可选择的数字签名，将认证级别提高到一个新的水平。数字签名用来确保卡中的 CHUID 记录是由一个可信的源数字签名的，同时它还保证卡被签名后，CHUID 数据是不会被修改的；截止日期用来验证卡未过期。这独立于任何截止日期。仅仅依靠读取和验证 CHUID，只提供了个人身份的某些保证，这是因为采用这种方式认证的是卡，而不是持卡人。PIN 和生物特征因素也提供了个人的身份验证。

15.6.2 PIV 文档

PIV 说明是很复杂的，NIST 发布了覆盖 PIV 广泛主题的许多文档。这些文档如下：

- **FIPS 201-2——联邦雇员和承包商的个人身份验证** 详细说明了物理卡的特征，存储媒体以及驻留在 PIV 卡中构成身份证书的数据元素。

- **SP 800-73-3——用于个人身份验证的接口**　详细说明了用于从智能卡中存储和检索身份证书的接口和卡的体系结构，并且提供了认证机制和协议的使用准则。
- **SP 800-76-2——用于个人身份验证的生物特征数据说明**　描述了 PIV 系统中生物特征证书的技术获取和形式化说明。
- **SP 800-78-3——用于个人身份验证的加密算法和密钥大小**　确定可接受的对称和非对称加密算法、数字签名算法和消息摘要算法，并且描述了用于确定与 PIV 密钥和数字签名有关算法的机制。
- **SP 800-104——用于 PIV 可视卡地形的方案**　提供了关于 PIV 卡颜色编码的附加推荐标准，用于标定员工的从属关系。
- **SP 800-116——物理访问控制系统中使用 PIV 证件的一个推荐标准**　描述了基于风险的方法，利用该方法可以选择一个合适的个人身份验证机制来管理对联邦政府设施和资产的访问。
- **SP 800-79-1——鉴别个人身份验证卡发行人是否合格的准则**　给出了鉴别个人身份验证卡的发行人是否可靠的准备，这些发行人收集、保存、传播个人身份证件，并且发行智能卡。
- **SP 800-96——PIV 卡和读卡器之间交互性准则**　给出了卡和读卡器之间便于互操作的要求。

除此之外，还有些文档涉及标识符的一致性检验和编码。

15.6.3　PIV 证书和密钥

PIV 卡包含了许多强制性和选择性的数据元素，它们可作为拥有不同强度和保障级别的身份证书。这些证书以单独或集合的形式被使用，用于认证 PIV 卡持卡人是否达到了特殊活动和交易所要求的保障级别。这些强制性数据元素如下：

- **个人身份证号（PIN）**　需要它来激活卡用于特权操作。
- **持卡人唯一身份标识（CHUID）**　包含了联邦机构智能证件号（FASC-N）和全球唯一的身份证号（GUID），这个号码唯一标识了卡和持卡人。
- **PIV 认证密钥**　一对非对称密钥和相应的证书，用于用户认证。
- **两个指纹模板**　用于生物特征的认证。
- **电子人脸图像**　用于生物特征的认证。
- **卡的非对称认证密钥**　一对非对称密钥和相应的证书，用于卡的认证。

选择性的数据元素如下：

- **数字签名密钥**　一对非对称密钥和相应的证书，支持文档签名和像 CHUID 这样数据元素的签名。
- **密钥管理密钥**　一对非对称密钥和相应的证书，支持密钥的建立和传输。
- **对称的卡的认证密钥**　支持物理访问的应用。
- **PIV 卡应用程序管理密钥**　与卡管理系统有关的对称密钥。
- **一幅或两幅虹膜图像**　用于生物特征认证。

表 15.5 列出了不同 PIV 密钥类型所要求的算法和密钥大小。

表 15.5　PIV 算法和密钥大小

PIV 密钥类型	算　法	密钥大小（位）	应　用
PIV 认证密钥	RSA	2048	支持交互式环境下卡和持卡人身份的认证
	ECDSA	256	

（续表）

PIV 密钥类型	算　法	密钥大小（位）	应　用
卡的认证密钥	3TDEA	168	支持物理访问中卡的认证
	AES	128,192 或 256	
	RSA	2048	支持交互式环境下卡和持卡人身份的认证
	ECDSA	256	
数字签名密钥	RSA	2048 或 3072	支持文档签名和临时签名
	ECDSA	256 或 384	
密钥管理密钥	RSA	2048	支持密钥的建立与传输
	ECDH	256 或 384	

15.6.4 认证

通过使用驻留在 PIV 卡中的电子证书，该卡支持下面的认证机制：

- CHUID　持卡人的认证使用了卡中签名的 CHUID 数据元素。PIN 是不要求的。这种机制在低级别的保证是可接受的、快速无接触的认证在必要情况下是很有用的。
- 卡认证密钥（Card Authentication Key）　PIV 卡的认证是在挑战/响应协议中使用认证密钥。个人身份证号（PIN）是不要求的。这种机制允许以接触（通过读卡器）或者非接触（通过无线电波）的方式来认证 PIV 卡，不需要持卡人的积极参与，并且提供了低级别的保证。
- BIO　持卡人的身份认证是在没有检查人员参与的环境下，比对他（她）的指纹样本和签名的生物数据元素。PIN 是需要的，用以激活该卡。这种认证机制达到了很高的保障级别，需要持卡人积极参与 PIN 和生物特征样本的提交。
- BIO-A　持卡人的身份认证是在有检测人员参与的环境下，比对他（她）的指纹样本和签名的生物数据元素。PIN 是需要的，用于激活该卡。这种认证机制达到了很高的保障级别，尤其是对卡中提取出生物特征模板的信任验证。这种机制需要持卡人积极参与 PIN 和生物特征样本的提交。
- PKI　持卡人的认证靠的是对挑战/响应协议中 PIV 认证私钥控制的展示。其中私钥可以使用 PIV 认证证书来验证。PIN 是需要的，用于激活该卡。持卡人这种认证机制对身份确定达到了很高的级别，它需要持卡人熟知 PIN。

在上述用例中，除了卡的对称认证密钥这个用例外，PIV 证书来源和完整性的有效性都是通过验证证书上的数字签名、并由可信的实体提供签名而得到的。

上面的每种认证类型都可以构造许多协议。SP-800-78-3 给出每种类型的例子。图 15.8 描绘了一个包含使用 PIV 认证密钥的认证会话。该会话提供了很高的保障级别，它很适合对拥有 PIV 卡、寻求访问计算机资源的用户进行认证。计算机（图中指定的本地系统）包含了 PIV 应用软件并且通过能够使用相对高级别过程调用的程序接口与卡进行通信。这些高级别的命令会转化 PIV 命令，然后通过物理接口经读卡器或无线接口下达到卡中。上述每一种情况，SP-800-73 都将卡的命令接口作为了 PIV 卡的边界。

认证过程是本地系统通过读卡器或无线连接的方式检测卡的时候开始的，接下来会在卡中选择一应用程序用于认证。本地系统为卡的 PIV 认证密钥请求公钥证书。如果证书有效（如具有有效的数字签名，没有过期或者被撤销），认证就会继续。否则，该卡就会被拒绝。下一步是本地系统请求持卡人输入卡的 PIN。如果用户提交的 PIN 和卡中存储的 PIN 相匹配，卡就会返回一个肯定的应答；否则卡就会返回一个失败的消息。根据不同的结果，本地系统要么继续，要么拒绝该卡。下一阶段是挑战/响应协议。本地系统发送一个需要被 PIV 签名的随机数，PIV 返回一个签名。本地系统使用 PIV 认证公钥来验证签名。如果签名有效，持卡人就被接受为确认身份人。否则本地系统就会拒绝该卡。

图 15.9 的会话完成了三种类型的认证。对持卡人的认证结合了卡和 PIN 服务知识。PIV 认证密钥证书确保了卡的证书的有效性。挑战/响应协议完成了对卡的认证。

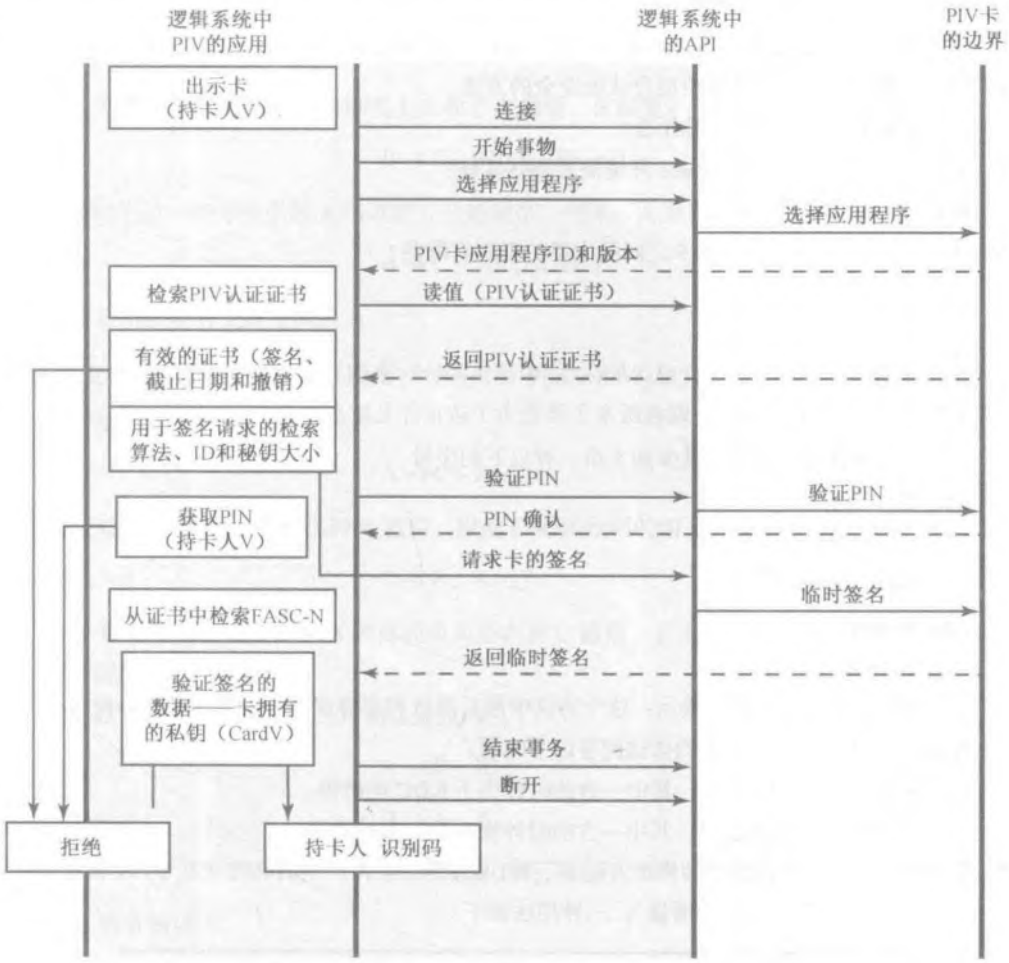


图 15.9　基于 PIV 认证密钥的认证

15.7　关键术语、思考题和习题

关键术语

认证	认证服务器	声称方	证书
证书服务提供商（CSP）	联合身份管理	身份管理	Kerberos
Kerberos 域	双向认证	临时交互号	单向认证
个人身份验证（PIV）	域	注册机构（RA）	可信赖方（RP）
重放攻击	用户	压制重放攻击	票据
票据授权服务器（TGS）	时间戳	验证方	

思考题

15.1　给出一个重放攻击的例子。

15.2 列出三个常用的防止重放攻击的方法。

15.3 什么是抑制重放攻击？

15.4 Kerberos 主要处理什么问题？

15.5 在网络或者 Internet 上，和用户认证相关联的三个威胁是什么？

15.6 列出三种确保分布式环境中用户认证安全的方法。

15.7 Kerberos 定义的 4 个要求是什么？

15.8 组成一个完全服务的 Kerberos 环境需要那些实体？

15.9 在 Kerberos 中，什么是域？

15.10 Kerberos 的版本 4 和版本 5 之间的主要的不同有哪些？

习题

15.1 在 15.4 节中，我们介绍了文献[WOO92a]中提出的针对密钥分发的公钥方案，其修正版在第（5）步及第（6）步中包含 ID_A。则该版本主要是为了防止什么攻击？

15.2 上题中的协议可以从 7 步减少到 5 步，有以下的序号：

（a）$A \rightarrow B$：

（b）$A \rightarrow KDC$：

（c）$KDC \rightarrow B$：

（d）$B \rightarrow A$：

（e）$A \rightarrow B$：

写出每一步传递的信息。提示：这个协议中最后消息和原协议中最后的消息是一样的。

15.3 参考 15.2 节中对压制攻击的描述回答以下问题：

（a）给出一个攻击的例子，其中一方的时钟快于 KDC 的时钟。

（b）给出一个攻击的例子，其中一方的时钟快于另一方的时钟。

15.4 使用临时交互号作为询问的典型方法有三种，假定 N_a 是 A 产生的临时交互号，A 和 B 共享密钥 K，$f()$ 是一个函数（如一个增量），三种用法如下：

用法 1	用法 2	用法 3
（1）$A \rightarrow B$：N_a	（1）$A \rightarrow B$：$E(K, N_a)$	（1）$A \rightarrow B$：$E(K, N_a)$
（2）$B \rightarrow A$：$E(K, N_a)$	（2）$B \rightarrow A$：N_a	（2）$B \rightarrow A$：$E(K, f(N_a))$

描述每种用法适用于什么情况。

15.5 在 PCBC 模型（参见附录 T 中图 T.2）中，说明为什么密文块中的一个随机错误会传播给随后的所有明文块？

15.6 假设在 PCBC 模型中，密文块 C_i 和 C_{i+1} 在传输中互换，说明为什么受影响是明文块 P_i 和 P_{i+1}，而不是随后的所有块。

15.7 除了为公钥证书格式提供了标准以外，X.509 也指定了一个认证协议，X.509 的之前版本中包含一个安全缺陷，该协议内容如下：

$$A \rightarrow B: A\{t_A, r_A, ID_B\}$$

$$B \rightarrow A: B\{t_B, r_B, ID_A, r_A\}$$

$$A \rightarrow B: A\{r_B\}$$

其中，t_A 和 t_B 为时间戳，r_A 和 r_B 为临时交互号，符号 $X\{Y\}$ 标识消息 Y 被 X 传输、加密及签名。

X.509 中的叙述表明检验时间戳 t_A 和 t_B 对认证是可选的，但考虑以下例子：假定 A 和 B 在之前的某些情况下使用了该协议，攻击者 C 截获以上三条消息，另外，假定时间戳没有用且全部被设为 0。最后，C 想要伪装为 A 与 B 通信，C 发送截获到的第一条消息给 B

$$C \rightarrow B: A\{0, r_A, ID_B\}$$

B 以为自己在和 A 通信，而事实上是和 C 在通信，B 回复：

$$B \rightarrow C: B\{0, r'_B, ID_A, r_A\}$$

C 同时通过一些方法引起 A 与其建立初始通信。结果，A 发送给 C 消息如下：

$$A \rightarrow C: A\{0, r'_A, ID_C\}$$

C 用 B 提供临时交互号回复 A

$$C \rightarrow A: B\{0, r'_B, ID_A, r'_A\}$$

A 回复

$$A \rightarrow C: A\{r'_B\}$$

当然 C 要向 B 证明 B 正在和 A 通信，因此 C 将接收到的如下消息给 B

$$C \rightarrow B: A\{r'_B\}$$

此时 B 会相信自己正在和 A 通信而事实是在和 C 通信。找出一种方法可以解决以上问题，且不使用时间戳。

15.8　考虑以下基于非对称加密的单向认证技术：

$$A \rightarrow B: ID_A$$

$$B \rightarrow A: R_1$$

$$A \rightarrow B: E(PR_a, R_1)$$

（a）解释该协议。

（a）该协议易受哪种类型的攻击。

15.9　考虑以下基于非对称加密的单向认证技术：

$$A \rightarrow B: ID_A$$

$$B \rightarrow A: E(PU_a, R_2)$$

$$A \rightarrow B: R_2$$

（a）解释该协议。

（b）该协议易受那种类型的攻击。

15.10　在 Kerberos，当 Bob 收到一个来自于 Alice 的票据，如何得知其是否真实？

15.11　在 Kerberos，当 Bob 收到一个来自于 Alice 的票据，如何得知其确实来自于 Alice？

15.12　在 Kerberos，若 Alice 收到一个回复，她如何得知该消息来自于 Bob（且是 Bob 最新的回复）？

15.13　在 Kerberos，票据包含哪些信息允许 Alice 和 Bob 安全通信？

第六部分　网络与 Internet 安全

第 16 章　网络访问控制和云安全
第 17 章　传输层安全
第 18 章　无线网络安全
第 19 章　电子邮件安全
第 20 章　IP 安全性

第 16 章　网络访问控制和云安全

学 习 目 标

通过本章的学习应具备以下能力：

- 说出一个网络访问控制系统的主要元素
- 说出主要的网络访问实施方法
- 描述可扩展认证协议的概况
- 理解 IEEE 802.1X 之基于端口的网络访问控制机制的操作和角色
- 描述云计算相关概念
- 理解涉及云计算独特的安全问题

本章我们从两个重要的方面讨论网络安全：网络访问控制和云安全。首先总体上介绍网络访问控制系统，总结它的主要元素和涉及的相关技术。接着我们会讨论可扩展的认证协议和 IEEE 802.1X，它们是两种广泛实施的标准，是许多网络访问控制系统的基础。

本章的后半部分讲述云安全相关内容。首先总体上介绍云计算相关知识，接着讨论云安全问题。

16.1　网络访问控制

网络访问控制（Network access control）是对网络访问管理的总称。NAC 在用户登录网络的时候进行认证，决定其可以获取哪些数据及可以执行哪些行为。NAC 同时还检测用户的计算机及移动设备（终端）的安全状况。

16.1.1　网络访问控制系统的组成元素

一个网络访问控制系统的组成元素包括：

- **访问请求者（AR）**　AR 是试图访问网络的节点。它可能是 NAC 系统管理的任何设备，包括工作站、服务器、打印设备、摄像机及其他具有 IP 的设备。AR 通常被认为是请求者或者简单说是客户端。
- **策略服务器**　根据 AR 的信息和企业制定的策略，策略服务器决定哪些访问是被允许的。策略服务器通常依赖后台系统，包括反病毒软件、补丁管理系统及用来帮助决定主机环境的用户字典。
- **网络接入服务器（NAS）**　NAS 是一个接入控制点，为用户远程连接企业的内部网络提供访问控制。通常它也被称为媒体网关、远程访问服务器或者策略服务器。一个 NAS 可能包含自己的认证服务器，或是依赖策略服务器提供的认证服务。

图 16.1 是一张常见的网络接入图。许多不同种类的 AR 通过向相应的 NAS 申请，来尝试访问

企业网络。第一步通常是对 AR 进行认证。认证通常包括一些安全协议和密钥的使用。NAS 可能执行这个认证过程，也可能只在认证过程中担当中介的作用。在之后的过程中，认证发生在请求者和认证服务器之间。认证服务器是策略服务器的一部分或者是通过认证服务器接入。

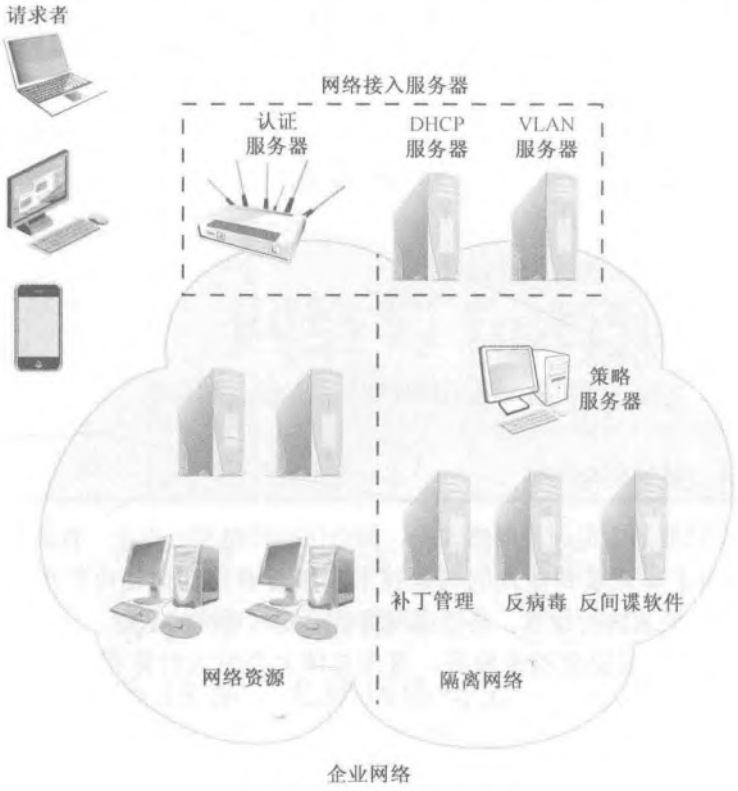


图 16.1 网络访问控制环境

认证过程提供多种目的的服务。它验证请求者所声明的身份，使得策略服务器能够据身份决定 AR 所拥有的权限。认证的过程可能会产生会话密钥，来确保在企业网络上用户和资源之间的安全通信。

通常，策略服务器或辅助服务器会对 AR 进行检查，决定 AR 是否可以进行远程交互连接的访问。这些检查（又称为健康、适配、筛选或评估检查）需要对用户系统上的软件进行认证，来确定其是否符合企业制定的安全配置的基本要求。比如，用户的反恶意软件必须是最新的，操作系统补丁要打满，远程主机必须是由企业拥有并控制的。这些检查必须在 AR 获得访问企业网络的授权之前进行。在这些检查结果的基础上，企业可以决定远程计算机是否可以进行交互远程访问。如果一个用户具有可接受的授权凭证但是远程计算机并没有通过安全检查，用户和远程计算机的网络访问会被拒绝，或者在隔离网络中进行受限的访问，这使得授权用户能够修复安全缺陷。图 16.1 展示了隔离网络的组成，通常包括策略服务器和 AR 相关的适用性服务器，也可能包含不需要常规安全门限的应用服务器。

一旦 AR 经过授权，明确了在企业网络访问中的级别，NAS 就会使 AR 能够与企业网络中的资源进行交互。NAS 可能会为了执行安全策略传递它的每次交换，也可能使用其他的方法限制 AR 的权限。

16.1.2 网络访问实施方法

实施方法是为了控制对企业网络的访问而实施在 AR 上的行为。许多供应商支持多种实施方法，允许用户使用一种或多种方法的组合来进行配置。下面将介绍常见的 NAC 实施方法：

- **IEEE 802.1X**　这是一个链路层协议，它在一个端口被分配 IP 之前执行授权。IEEE 802.1X 使用可扩展授权协议进行授权。16.2 节和 16.3 节分别讲述了可扩展授权协议和 IEEE 802.1X。
- **虚拟本地局域网（VLAN）**　企业网络是一系列有联系的 LAN 组成的集合。通过这种方式，它在逻辑上被分割为一系列虚拟的 LAN。NAC 系统根据设备是否需要安全修复、只需要网络访问或是需要访问企业资源决定哪些 VLAN 来管理相应的 AR。VLAN 可被动态创立，VLAN 中服务器和 AR 之间的成员关系可重叠。也就是说一个服务器或是一个 AR 可以属于一个或多个 VLAN。
- **防火墙**　一个防火墙提供一个表单。它允许或是拒绝企业主机和外部用户之间的网络流量。防火墙将在在线部分的第 23 章中进行讨论。
- **DHCP 管理**　动态主机配置协议是一个可以给主机动态分配 IP 的网络协议。一个 DHCP 服务器收到 DHCP 请求，然后进行 IP 分配。因此，NAC 的实施发生在以子网和 IP 分配为基础的 IP 层。一个 DHCP 服务器的安装和配置是简单的，但是它受限于多种形式的 IP 欺骗，提供了受限的安全性。

还有很多其他的实施方法被提供商支持。上面所介绍的是一些比较常见的方法，其中 IEEE 802.1X 是目前使用最多的实施方案。

16.2　可扩展认证协议

可扩展认证协议（EAP）在 RFC 3748 中定义。它是一个网络访问和授权协议的框架。EAP 提供了许多协议消息，它们可以封装多种认证方法，在客户端和认证服务器之间使用。EAP 可以在多种网络和链路层设备上进行操作，包括点到点链路、LAN 及其他网络，可以适应多种链路和网络的认证需求。图 16.2 展示了组成 EAP 的协议层次。

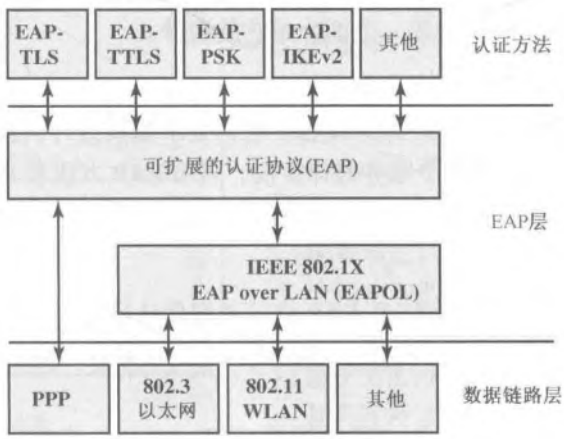


图 16.2　EAP 层环境

16.2.1　认证方法

EAP 支持多种认证方法，这就是 EAP 可扩展的含义所在。EAP 在客户端系统和认证服务器之间为认证信息交换提供了一个通用的传输服务。基本的 EAP 传输服务通过使用一个特殊的认证协议和方法进行扩展。这些协议会同时装在 EAP 客户端和认证服务器。

许多方法都规定可以在 EAP 上工作。下面是一些常见的 EAP 支持的方法：

- EAP-TLS（EAP Transport Layer Security） EAP-TLS（RFC 5216）定义了 TLS 协议（将在第 17 章介绍）如何封装在 EAP 消息里。EAP-TLS 使用 TLS 中的握手协议，而非它的加密方法。客户端和服务器使用其数字证书进行相互认证。客户端通过使用服务器的主密钥加密一个随机数产生一个预主密钥，并将其发往服务器。客户端和服务器都使用预主密钥产生相同的密钥。
- EAP-TTLS（EAP Tunneled TLS） EAP-TTLS 和 EAP-TLS 基本相同，不同之处在于服务器首先要通过证书向客户端证明自己。和 EAP-TLS 中相同，它使用密钥建立一个安全的连接（“隧道”），这个连接用来对客户进行认证，可能的话，服务器会再次使用 EAP 方法或是老方法如 PAP（口令认证协议）和 CHAP（挑战-握手认证协议）。EAP-TTLS 在 RFC 5281 中有详细的定义。
- EAP-GPSK（EAP Generalized Pre-Shared Key） EAP-GPSK 在 RFC 5433 中定义。它是一种适用交互认证的 EAP 方法，会话密钥由预共享密钥（PSK）产生。EAP-GPSK 指定了一个基于预共享密钥的 EAP 方法，采用了基于密钥的密码算法。因此，在消息流和计算开销方面十分有效，但是它要求客户端和服务器之间存在预共享密钥。它为成功进行交互认证提供了受保护的安全信道，它是为在如 IEEE 802.11 这种不安全网络上进行认证设计的。EAP-GPSK 不需要公钥密码算法。这种 EAP 方法协议交换最少可 4 步完成。
- **EAP-IKEv2** 它的基础是第 20 章会讲到的互联网密钥交换协议版本 2（IKEv2）。它支持交互认证，会话密钥建立使用了多种方法，它在 RFC 5106 中进行定义。

16.2.2 EAP 交换

无论何种方法使用在认证中，认证信息和认证协议信息都会包含在 EAP 消息中。

RFC 3748 定义了 EAP 消息交换的目标是成功的认证。在 RFC 3748 中，成功的认证是指，在一次 EAP 消息交换中，根据结果认证者决定是否允许客户访问及客户端决定是否进行访问。认证者的决定通常包括认证和授权。客户端可能通过了认证者的认证，但访问可能会因为策略原因被认证者拒绝。

图 16.3 展示了一个典型的 EAP 使用。下面介绍它的组件：

- EAP peer 尝试访问网络的客户计算机。
- EAP authenticator 一个接入点或是 NAS。它们要求提前进行认证接入到网络。
- Authenticator server 它是一个服务器计算机，利用 EAP 方法和 EAP 客户端进行交互，认证客户端的信息，授权其访问网络。典型地，授权服务器是一个提供远端用户拨入认证服务（RADIUS）的服务器。

认证服务器作为后台服务器可以为许多 EAP 认证者提供认证客户的服务。EAP 认证者进一步决定是否给客户端提供接入，也被称为 EAP 转移模式。有时认证者可能会代替 EAP 服务器，如此在 EAP 执行过程中就只包含两部分。

第一步，通常使用一个底层协议（如 PPP 协议或 IEEE 802.1X）来连接 EAP 认证者。EAP 客户端在这层操作的软件实体称为请求者。EAP 消息对于每种选定的 EAP 方法包含适当的信息。它会在 EAP 客户端和认证服务器之间进行交换。

EAP 消息应该包含以下几个部分：

- Code 指明 EAP 包的类型，共有 4 种：Request（1）、Response（2）、Success（3）和 Failure（4）。
- Identifier 用于匹配 Request 消息和 Response 消息。
- Length EAP 包的长度，包含了 Code、Identifier、Length 和 Data 域。

- Data　包含涉及认证的信息。通常 Data 域包含 Type 域和 Type-Data 域，其中 Type 域指明了所包含数据的类型。

Success 和 Failure 消息不包含 Data 域。

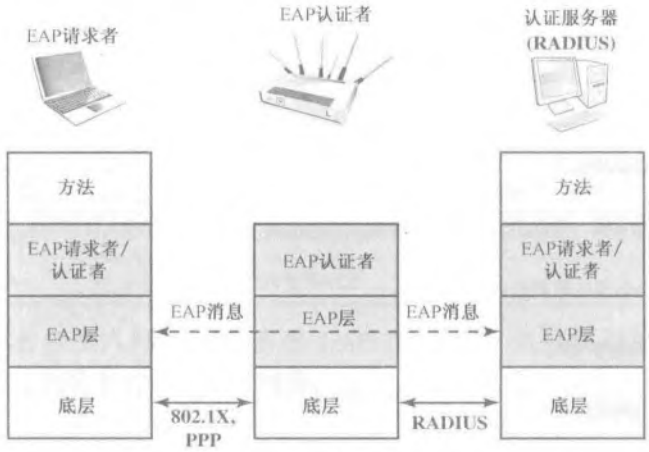


图 16.3　EAP 协议交换

EAP 认证交换过程如下所述。在底层交换建立了 EAP 所需连接后，认证者向客户端发送一个对于其身份的请求，客户端会返回一个带有其身份信息的回应。通信过程会遵从一系列流程，为了完成认证信息的交换，流程由一系列认证者的请求和客户端的回应组成。请求-应答交换的次数取决于验证方法。会话会在出现以下两种情况时终止：（1）认证者认为该客户端不能通过认证，发送 Failure 消息；（2）认证者认为认证成功，发送 Success 消息。

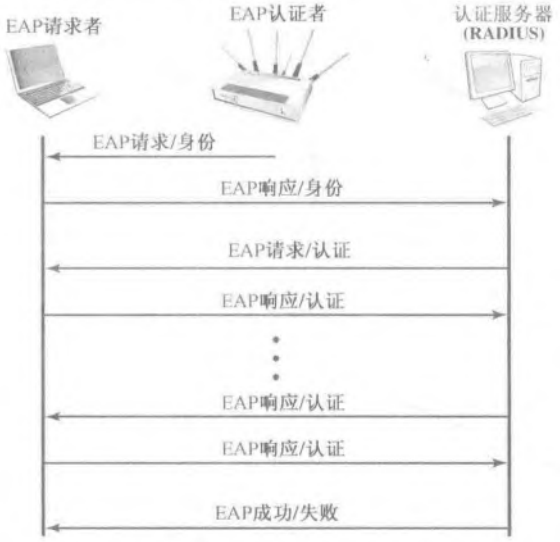


图 16.4　EAP 转移模式中的消息流

图 16.4 给出了一个 EAP 交换的例子。图中有一个从客户端发送到认证者的消息或信号并没有显示，它使用非 EAP 协议请求一个 EAP 交换来获得网络接入。一个例子就是下节讨论的 IEEE 802.1X。认证者和客户端交互中首次 Request 和 Response 消息是关于身份的，认证者请求客户端的身份，客户端在应答消息中声明自己的身份。应答通过认证者传递到认证服务器。其余的 EAP 交换在看客户端和认证服务器之间进行。

在收到身份 Response 消息后，服务器选择一种 EAP 方法，并发送第一个 EAP 消息，这个消息的 Type 域与这种 EAP 方法相关联。如果客户端支持这种 EAP 方法，就会回应一个这种方法的应答消息。如果不支持，客户端发送一个 NAK，服务器或者选择另一种 EAP 方法或者发送 Failure 消息结束这次连接。选择的 EAP 方法决定了请求-应答交换的次数。在交换中进行了包括密钥材料等适当认证信息的交换。当服务器认为认证成功或是认证失败时，交换结束。

16.3　IEEE 802.1X 基于端口的网络访问控制

IEEE 802.1X 基于端口的网络访问控制是为 LAN 提供访问控制设计的。表 16.1 简要定义了在

IEEE 802.11 标准使用的关键词。Supplicant、Network access point 和 Authentication 分别对应 EAP 中的 Peer、Authenticator 和 Authentication Server。

表 16.1 IEEE 802.1X 相关术语

Authenticator 一个点到点的 LAN 一端的实体，对链路另一端的实体进行认证
Authentication exchange 系统之间执行认证过程的一个双方会话
Authentication process 执行认证的密码操作和支持数据帧格式
Authentication server(AS) 一个为认证者提供认证服务的实体。根据请求者提供的身份，它决定是否授权请求者进入认证者所在的网络
Authentication transport 在两个系统之间传输认证交换信息的数据报会话
Bridge port 一个 IEEE 802.1D 或 802.1Q 网桥的端口
Edge port 一个只有一个网桥的网络中的网桥接口
Network access port 一个系统中 LAN 的一个连接点，可以是物理端口，如一个 LAN 中的一个 LAN MAC，也可以是一个逻辑端口，比如一个工作站和接入点的 IEEE 802.11 连接
Port access entity(PAE) 与一个端口相关的协议实体。它可以支持协议功能和认证者或是请求者或两者的连接
Supplicant 一个点到点的 LAN 一端的实体，被链路另一端的认证者进行认证

直到 AS（使用一个认证协议）认证一个请求者，认证者只是在 AS 和请求者之间传递控制和认证消息。此时 802.1X 控制信道是通畅的，802.11 数据信道是拥挤的。一旦请求者认证被通过，并被分配了密钥，认证者就能转发请求者的数据，但是其受限于预定义的请求者访问权限。在这种情况下，数据信道就顺畅了。

正如图 16.5 所示，802.1X 使用了受控端口和非受控端口的概念。端口是认证者和相关物理网络连接定义的逻辑实体。每个逻辑端口对应这两种物理端口中的一个。一个非受控端口允许协议数据单元（PDU）在 AS 和请求者之间交换，不需考虑请求者的认证状态。一个受控端口只有在请求者的状态允许这次交换时才允许 PDU 在网络上其他系统和请求者之间进行交换。

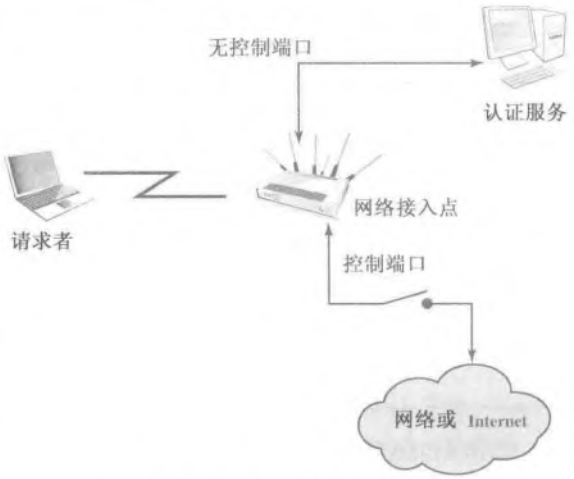


图 16.5 802.1X 访问控制

802.1X 定义了一个被称为 EAPOL（EAP over LAN）的基本元素。EAPOL 工作在网络层，在链路层使用 IEEE 802 LAN，比如以太网或 Wi-Fi。EAPOL 能使请求者连接到认证者，并支持 EAP 认证数据包的交换。

表 16.2 中展示了最常用的 EAPOL 数据包。当请求者最初连入 LAN，它不知道认证者的 MAC 地址。实际上它并不知道此刻是否存在一个认证者。请求者通过给为 IEEE 802.1X 认证者指定的组多播地址发送 EAPOL-Start 数据包来探测是否有认证者存在，并告知它请求者就位。在许多情况下，

认证者会通过硬件通知一个设备已经接入了。比如，一个集线器在一个电缆接入时就已经知道了，不需要它发送信息。这种情况下，认证者可能就用自己的信息抢占了开始信息。无论在哪种情况下，认证者都会发送一个封装在 EAPOL-EAP 信息中的 EAP-Request Identity 消息。EAPOL-EAP 是一种传输 EAP 数据包的 EAPOL 帧类型。

表 16.2　EAPOL 帧类型

帧 类 型	定　义
EAPOL-EAP	包含一个被封装的 EAP 包
EAPOL-Start	请求者能发送这个包，不用等待认证者的挑战
EAPOL-Logoff	当请求者结束使用网络时，返回端口的状态至未授权
EAPOL-Key	用来交换密钥信息

在认证者允许请求者接入网络后，使用 EAPOL-Key 数据包将相关密钥发送给它。发送 EAPOL-Logoff 数据包表明请求者希望离开网络。

EAPOL 数据包格式包括以下域：

- **协议版本（Protocol version）**　EAPOL 的版本。
- **包类型（Packet type）**　包括 start，EAP，Key，Logoff 等。
- **包体长度（Packet body length）**　如果数据包包括 body，就表明 body 的长度。
- **包体（Packet body）**　EAPOL 数据包的载荷，比如 EAP 包。

图 16.6 展示了一个使用 EAPOL 交换的例子。在第 18 章中我们会讨论在 IEEE 802.11 无线 LAN 中使用 EAP 和 EAPOL 的安全性。

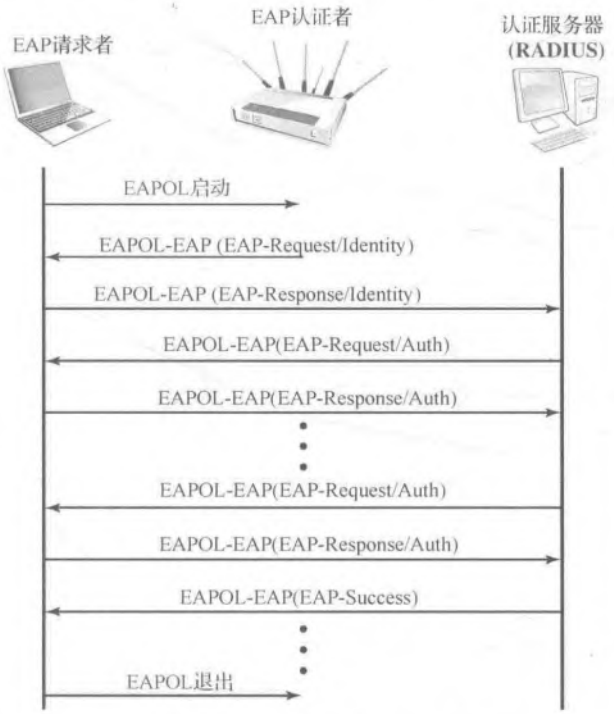


图 16.6　IEEE 802.1X 时序图举例

16.4 云计算

将大部分或是全部信息技术操作转移到被称为企业云计算的联网基础架构是一个日益凸显的趋势。本节就来对云计算进行综合介绍。关于云计算的更多细节请参见文献[STAL16]。

16.4.1 云计算元素

在 NIST SP-800-145（云计算的 NIST 定义）中，NIST 对云计算的定义如下所述。

> **云计算** 一种能够对共享的联合可配置计算资源（网络、服务器、存储器、应用程序、服务）进行普遍、方便、按需的网络访问的模式。这些资源可以在最小化的代价或是服务提供者的交互下迅速进行预分配、发布。这种云模式促进了可用性，有 5 个基本特性、3 种服务模式和 4 种部署模式。

在图 16.7 中展示了定义相关的模式及它们的特性。基本的云计算特性包括：

- **广域网络访问** 在网络和通过标准机制的访问都是可用的。其中标准机制可以促进异构稀疏或是密集客户平台（移动电话、笔记本、PDA）及其他传统或是基于云的软件服务的使用。
- **快速弹性** 云计算可以使你根据自己的特殊服务需求来扩展或是减少资源。比如，可能为了持续一个特殊的任务需要大量的资源，在完成任务后可以释放这些资源。
- **计量的服务** 云系统可以自动控制和优化资源，这是通过根据服务的类型（存储、处理、带宽、活跃的用户账号）在合适的时候进行抽象量化达到的。资源的使用可以被监控、控制、报告，并透明地提供给服务使用的双方。



图 16.7 云计算的要素

- **按需自助服务** 一个用户可根据需求单方面规定计算能力（服务时间、网络存储），不需要和服务提供者的交互。因为服务是按需的，资源不是 IT 结构的长久部分。
- **资源组合** 提供者的资源根据多租赁模式进行组合为多个用户提供服务。它根据用户的要求对物理和虚拟资源进行分配和重分配。这里存在位置独立的度的问题，用户可能不知道或

不能控制提供资源的明确位置，但可以在高的抽象层（国家、州、数据中心）指明其位置。资源可以是处理、内存、网络带宽、虚拟机。甚至在私有云中都有在一个组织中连接不同部分资源的趋势。

NIST 给定了三种服务模式，可以被视为嵌入服务的替代者：

- **软件即服务（SaaS）**　它使得用户可以使用在云上提供者提供的应用程序。这些应用程序通过像浏览器这种用户接口使得多种用户可以访问。企业可以在云服务获得通常的功能，同时不需要软件产品所需的桌面或是服务器的权限。SaaS 节省了软件安装、维护、升级、补丁的复杂和烦琐。这种服务的例子如 Gmail 和 Saleforce.com，其中 Gmail 用来提供谷歌的电子邮件服务，后者用来帮助商户保持与客户的联系。
- **平台即服务（PaaS）**　这种服务允许客户使用提供者支持的编程语言或工具创建自己需要的应用程序并部署在云平台上。PaaS 提供中间件模式服务，如软件需要的数据库、组件服务。实际上，PaaS 是云中的操作系统。
- **基础设施即服务（IaaS）**　这种服务可以使用户规定处理、存储、网络和其他基本的计算资源。用户可以部署、运行任意系统或应用软件。IaaS 使用户可以联合基本的计算资源（如数据计算、数据处理）来构建高适应性的计算系统。

NIST 给定了 4 种部署模式：

- **公有云**　这种基础结构来自一个出售云服务的组织，提供给一般大众或大的企业组织。云提供者为云基础架构及云内数据控制和操作负责。
- **私有云**　这种云基础架构只由一个组织操作。它由该组织或第三方进行管理，可以是内部部署或外部部署。云提供者（CP）只为云基础架构负责，不为控制负责。
- **社区云**　这种云基础架构由多个组织共享，支持一个特殊的社区，它们有共同的关注，比如任务、安全需求、策略、合规考虑。它由组织或第三方进行管理，可以是内部部署或外部部署。
- **混合云**　这种云基础架构是两个或多个云的组合（公共云、私有云、社区云），它们保持各自的特性，使用数据、应用程序可移植标准或是适当的技术聚合（如云爆发，它用来达到云之间的负载平衡）。

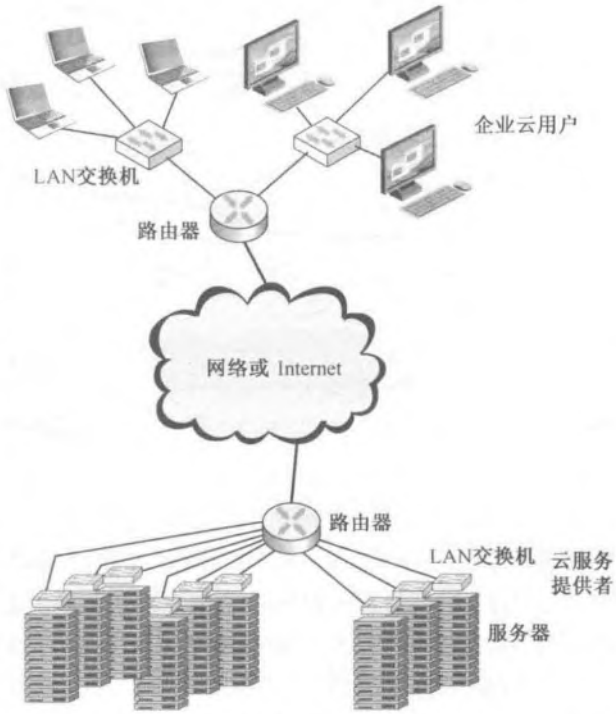


图 16.8　云计算环境

图 16.8 展示了典型的云服务环境。一个企业在企业 LAN 或 LAN 集合中维护工作站，通过路由器穿过网络或因特网连接到云提供者。云提供者维护大量的服务器，它使用网络管理、冗余、安全工具进行管理。在图中，云基础架构是大量的刀片服务器的组合，这是常见的架构。

16.4.2 云计算相关架构

NIST SP500-292（NIST 云计算参考架构）建立了相关架构，其描述如下所述。

> NIST 云计算参考架构关注一些需求，这些需求描述云服务提供什么，并非如何去设计解决方案和实施方法。参考架构试图去促进对于云计算中复杂操作的理解。它并没有提出一个具体云计算系统的架构，相应地，它是一个使用通用框架去描述、讨论、发展云计算系统架构的工具。

NIST 开发了参考架构以达到以下目的：

- 展示和理解在一个完整云计算概念模型环境中的多种云服务。
- 为用户理解、讨论、分类、比较云服务提供技术参考。
- 促进对候选标准安全性、互操作性、可移植性、参考实施的分析。

图 16.9 中，描述了参考架构，定义了 5 种角色和相关责任。

- **云用户** 和云提供商保持商业关系或是使用其服务的组织或个人。
- **云提供商** 负责为感兴趣的用户提供可用服务的个人、组织或实体。
- **云审计者** 一个可以管理云服务独立资产、信息系统操作、性能和云实施安全的实体。
- **云代理商** 一个管理云服务使用、性能、分发的实体，连接 CP 和用户。
- **云运营商** 一个在 CP 和云用户之间提供云服务连通性和传输的中间商。



图 16.9 NIST 的云计算参考结构

云用户和云提供商的角色已经讨论过了。总体来说，云提供商可以根据云用户的需求提供一种或多种云服务。对于三种模式（SaaS、PaaS、IaaS）中的每一种，CP 提供了支持相应服务模式所需的存储、处理设备和云用户所用的接口。对于 SaaS，CP 部署、配置、维护、更新云架构中的应用程序的操作，以保证服务能够达到用户期望达到的水平。SaaS 的用户可以是为他们的用户提供应用软件访问的组织（使用应用软件的终端用户），可以是为终端用户配置应用的应用软件管理者。

在 PaaS 中，CP 管理平台的计算架构，运行软件来提供平台的组件，比如运行时软件执行堆栈、数据库及其他的中间件组件。云用户可以使用 CP 提供的工具或资源去开发、测试、部署、管理在云环境中的应用程序。

在 IaaS 中，CP 需要提供物理基础设施，包括服务器、网络、存储和托管基础设施。用户根据它们的基本计算需求向虚拟计算机轮流使用这些计算资源。

云运营商是一个在 CP 和云用户之间提供云服务连通和传输的网络设备。典型地，CP 会和云运营商签订服务水平协议，来给用户提供持续的相应水平的服务。可能需要云运营商在用户和 CP 之间提供专用的安全连接。

在云服务太复杂以至于用户不能简单的管理时，云代理商十分好用。云代理商主要提供以下三方面的支持：

- **服务中介**　比如身份管理、性能报告、安全性增强等增值服务。
- **服务聚合**　代理组合多种云服务来满足用户不希望被一个单独 CP 处理的需求，或是进行性能优化或是代价最小化。
- **服务仲裁**　它和服务聚合很像，不同之处在于聚合的服务是不确定的。服务仲裁意味着一个代理可以灵活地在众多代理中选取服务。比如云代理商可使用信用评价服务来进行测量并从中选取一个评分最高的代理。

云审计者可以根据安全控制、隐私影响、性能等评估 CP 提供的服务。云审计者是一个独立的实体，可以保证 CP 遵从一系列标准。

16.5　云安全所面临的威胁和对策

一般来说，云计算中的安全控制和其他 IT 环境中的安全控制基本相同。但是由于为了能够提供云服务，其采取了相应的技术和操作模型，云计算可能面临云环境所独有的威胁。就这一点而言，它的基本概念是企业丢失了大量的对于资源、服务的控制，但是应用程序必须保持安全和隐私策略的有效性。

云安全联盟[CSA10]列出了下列云环境所面临的重要威胁，并给出了对策：

- **云计算的滥用**　对于许多 CP，注册和使用云服务是十分容易的，一些还提供无限制的试用期。这使得攻击者可以进入到内部引导许多攻击，比如垃圾邮件、恶意代码攻击和拒绝服务。PaaS 提供者经常受到这种类型的攻击，但是很多证据表明 IaaS 提供商开始成为攻击者的目标。这方面主要由 CP 进行攻击的防御，但是云服务客户必须监视其数据和资源方面的活动来检测恶意行为。

  对策包括：（1）更严格的注册和认证过程；（2）增强对信用卡伪造的监控和相关协调；（3）用户网络流量的综合分析；（4）在自己的网络块内关注公开的黑名单。
- **不安全的接口和 API**　CP 开发了一系列的接口和 API 供用户来使用，进行对云服务的管理和交互。一般云服务的安全性和可用性依赖于基本 API 的安全性。从认证、访问控制到加密、活动控制这些接口必须被良好设计。它应该避免被无意的或恶意的绕过策略。

  对策包括：（1）分析 CP 接口的安全模型；（2）确保强壮的认证和访问控制和加密传输的协同实施；（3）理解 API 和依赖链的联系。
- **恶意的内部成员**　在云计算范例中，一个组织放弃了许多安全方面的直接控制，给了 CP 一个很高级别的信任。一个重要的问题是恶意内部成员活动带来的威胁。云架构使得特定的位置有很大的威胁，比如 CP 系统管理员和管理安全服务的提供者。

  对策包括：（1）实施严格的供应链管理，实施综合的供应商评估；（2）将人力资源需求归

入法律合同；（3）要求所有信息安全、管理部分、报告的透明性；（4）决定安全分支通知过程。

- **共享技术问题** IaaS 提供商通过共享架构以一种可扩展的方式来实现它们的服务。通常组成架构的底层组件（CPU 缓存、GPU 等）并没有为多客户架构进行特殊的设计来保证独立性。CP 解决这类威胁通常采用的方法是为独立的客户采用相互独立的虚拟机。但这种解决方法仍然易受到攻击（包括内部和外部），所以只能是这个安全策略的一部分。
  对策包括：（1）对安装和配置进行安全最佳实践；（2）监视那些未授权的更改和活动；（3）对于管理访问和操作推行强认证和访问控制；（4）在打补丁和脆弱性补救时实施 SLA；（5）实施漏洞扫描和配置审计。
- **数据丢失或泄露** 对于多数用户来说，安全领域最大的影响的问题就是数据的丢失和泄露，我们将会在接下来的章节里讨论。
  对策包括：（1）实施强 API 访问控制；（2）在传输时加密并保护数据的完整性；（3）在设计和运行时分析数据的保护；（4）在打补丁和脆弱性补救时实施 SLA；（5）实施强壮的密钥产生、存储、管理、分配实践。
- **账户或服务劫持** 账户或服务劫持仍然是顶级的威胁，通常伴随着证书的窃取。通过窃取证书，攻击者可以访问到所部署云服务的关键区域，使得他们能够破坏这些服务的秘密性、完整性和可用性。
  对策包括：（1）禁止在用户之间和服务器之间共享证书；（2）采用双因子认证技术；（3）采用前瞻性监视来检测未授权活动；（4）理解 CP 安全策略和 SLA。
- **未知的安全威胁** 在使用云架构时，客户必须在很多问题上将控制权给 CP，这可能会导致很多安全问题。因此客户必须明确定义管理安全威胁的相关角色和责任。比如，雇员可能在 CP 部署应用程序和数据资源时并没有遵循正常的隐私、安全、监督策略和程序。
  对策包括：（1）审计可用的日志和数据；（2）对基础设施细节的部分或完全审计（补丁级别、防火墙）；（3）监控并更改必要的信息。

欧洲网络与信息安全署（ENISA）和美国国家标准技术研究所（NIST）给出了类似的清单[ENIS09,JANS11]。

16.6 云中的数据保护

从上节可以看到云安全中存在大量安全问题，并有大量途径提供了保证云安全的方法。进一步的例子可以参见 NIST 的云安全指导方针 SP-800-14，表 16.3 列出了相关内容。云安全在本章中进行了整体的介绍，本节将关注云安全中的一个特殊元素。

表 16.3 NIST 在安全隐私问题上的指导方针和建议

管理
扩展组织实践附属于一些策略、程序、标准，这些策略、程序、标准是在云中应用程序开发、服务供应时使用的。类似地，设计、实施、测试、使用、对于部署、使用服务的监控也是在应用程序的开发、服务供应时使用的
使用审计机制和工具来确保组织实践在系统生命周期中受控
承诺
理解多种相关法律、条款，主要是关于加强组织安全、隐私义务和有可能影响云计算初始化的部分，涉及数据位置、隐私和安全性控制、记录管理及电子举证需求
检查和评估云提供商是否保证了组织的要求，确保合同的条款确实满足要求

（续表）

确保云提供商电子举证过程不会威胁数据和应用程序的隐私和安全
信任
保证服务的安排有足够的手段来允许云服务提供者所提供的安全、隐私控制、过程和运行性能可见
在数据上建立清晰、唯一的控制权
建立一个威胁管理项目，保证它是足够灵活的来适应系统生命周期中一直进化、改变的威胁
持续的观察系统的安全状态来为正在进行的威胁管理决策提供支持
体系结构
理解云服务提供商提供服务在整个生命周期和所有系统组件中所使用的底层技术。包括系统安全及隐私相关的技术控制
识别和访问管理
确保在合适的地方有足够的安全保障来进行安全认证、授权和其他识别和访问管理功能，同时对于组织是合适的
软件隔离
理解虚拟化技术和其他逻辑隔离技术。云服务提供者在多客户软件结构中采用这种技术并评估该组织的相关风险
数据保护
评估云服务提供者的数据管理方法的灵活性，它主要相对于组织关心的数据和对于数据访问控制、闲时、传输时、使用时数据安全，以及审查数据的能力
考虑手机原始数据的威胁，尤其是那些数据欺骗成分高的或是数据结合在一起代表重大价值的
充分理解和权衡那些在云环境可用设备和云服务提供者建立过程中所涉及的密钥管理威胁
可用性
理解合同条款、程序在数据备份、恢复和灾难恢复时的可用性，确保它们满足组织连续性和突发性计划需求
确保在发生破坏、经受灾害或是非法操作时能够立刻恢复，所有的操作都能重新及时、有组织地重新建立
事件响应
理解事件响应相关的合同条款、程序，确保它们满足组织的需求
确保云服务提供者有透明的响应过程，并有足够的机制在经历和响应过后能够共享信息
确保组织能够在协同模式进行响应和云服务提供者在云环境中按照相关的角色进行协同

危害数据的方法有很多。在没有备份的情况下，删除和修改原始的上下文就是一个直接的例子。从一个大的上下文中分离出一个记录可能导致它不可恢复，因为存储介质可能不安全。丢失编码密钥会产生数据的毁坏。最后未授权的实体不允许对敏感数据获取访问权限。

危害数据安全的威胁在云中更加严重。这主要是因为面临威胁的数量和它们之间存在的交互。这些威胁或许是云中独特的，或是因为在云环境中架构或操作的特性使得其更加严重。

云计算中数据库环境的使用十分具有代表性。一些提供商支持多实例模式，为每个云用户提供一个单独的 DBMS 运行在虚拟机实例上。这给了用户对于角色定义、用户授权及其他与安全相关管理任务的完全控制。另一些提供商支持多用户模式。他们为每个用户提供与其他用户共享的预定义的环境，典型的方式是使用用户的标识来标记数据。标签让用户看起来好像在独占地使用这个实体，但它靠 CP 来建立维护一个健康、安全的数据库环境。

数据的安全应该在空闲时、传输时、使用时都得到保证，服务数据必须进行控制。客户在传输时可能采用加密来保护数据，但其中的涉及 CP 承担的密钥管理。客户执行访问控制技术，但是在一些依靠服务模式的方面会涉及 CP。

在数据空闲时，一个完美的方法是用户对数据进行加密，并只存储加密后的数据在云中，CP 无法拿到密钥。所以只要密钥是安全的，CP 就无法读取数据，不过损坏和其他拒绝服务攻击仍然有效。

一个直接的对于该安全问题的解决方法在上文提出，就是对这个数据库进行加密，并不对提供商提供密钥。但是这种解决方式本身是不灵活的。用户如果想根据关键词进行搜索个别条目就会受限，他必须下载这个表，然后进行解密，之后再进行相关操作。为了提高灵活性，在加密方式下数据库必须是可用的。

图 16.10 描述了一个例子，它在文献[DAMI05]和文献[DAMI03]中有相关描述，类似的途径还在文献[HACI02]中有描述。其中涉及 4 个实体：

- **数据所有者（Data owner）** 一个组织，它为组织内部或是外部的用户生产可用的数据。
- **用户（User）** 正在请求系统的人类实体。用户可以是雇员，他通过服务器获取了对数据的访问权限；用户可以是外部的，他通过认证获取了对数据的访问权限。
- **客户端（Client）** 前端，它将用户的查询转换成服务器上加密数据的查询。
- **云服务器（Server）** 一个组织，从 data owner 接收到加密数据，并使得它们可以分配到客户端。它可以是 data owner 所有，但通常是外部提供商提供并维护的设备，在我们的讨论中，它是云服务器。

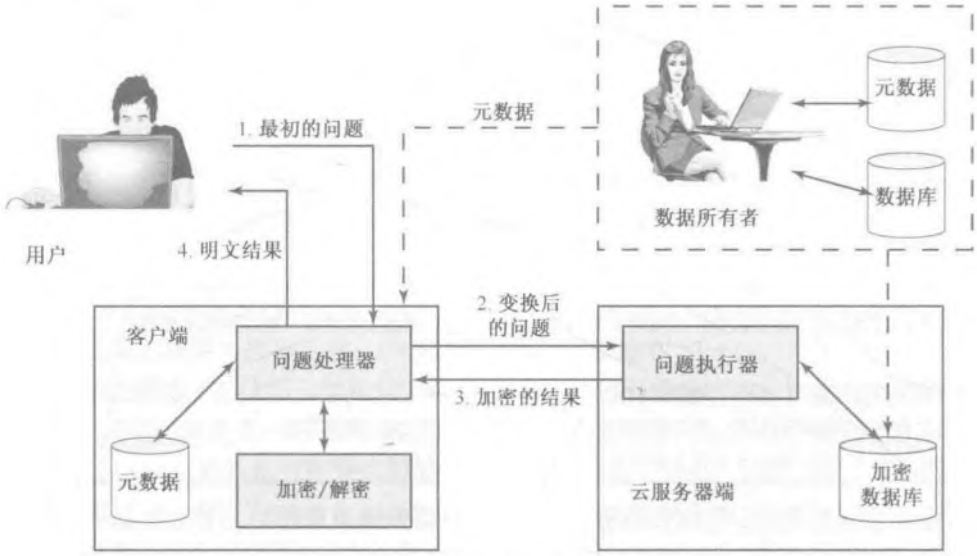


图 16.10 一种基于云的数据库加密方案

在继续讨论之前，需要定义一些数据库条目。在关系数据库用法中，基本建筑块是关系，它是一张二维表。行被称为元组，列被称为属性。一个主键被定义为一行的一部分，它能在这个表中唯一地标识这一行，一个主键由一个或多个属性组成。比如，在雇员表中，雇员 ID 足以在表中唯一标识这行。

让我们首先基于这种情况进行一个可能出现的简单测试。假设数据库中每个不同的条目都使用相同的密钥分别加密。加密的数据存储在服务器，但是服务器并无密钥。因此，数据在服务器是安全的。即使一些人可以攻击服务器，他所能获取的数据只是加密的数据。客户端有密钥的副本。一个客户端的用户可以通过以下的方式进行一条记录的检索：

（1）用户使用特殊的主键值来发送一个请求，在一个或多个记录中进行查询。
（2）客户端的请求处理器将主键值进行加密并修改相关的查询，并传输到服务器端。
（3）服务器使用加密后的主键值处理相关的查询，并返回相关的记录。
（4）请求处理器解密数据并返回。

这种方式相当直接但是却很受限。比如，如果这个表里有个工资字段，用户想检索所有工资小于 70×10^3 美元的记录。这种方法就没有直接的方式，因为每条记录中的工资值都是加密的。加密值的集合并没有保留原始属性值中的大小顺序关系。

有很多方式可扩展这种方法的功能性。比如，一个未加密的索引值可以和一个给定的属性联系起来，一个表就可以根据这个索引值进行划分，就使得用户可以查询特定的部分。具体的细节已超出了我们探讨的范围，感兴趣的读者可以参阅文献[STAL15]。

16.7　云安全即服务

“安全即服务”（SecaaS）的普遍含义是指服务提供商提供的一系列安全服务包。这使得企业卸下了安全方面的负担将其转给服务提供商。服务中一般提供认证、反病毒、反间谍软件、入侵监测和安全事件管理。在云计算环境下，云安全即服务（一种特定的 SecaaS）是 CP 提供的软件即服务的一部分。

云安全联盟对安全即服务定义为安全应用程序和服务中的条款，它们既可以是基于云的基础设施和软件，也可以是用户预定义系统的云[CSA11b]。云安全联盟定义了以下安全即服务类别：

- 身份和访问管理
- 数据丢失保护
- Web 安全
- 电子邮件安全
- 安全评估
- 侵入管理
- 安全信息和事件管理
- 加密
- 业务连续性和灾难恢复
- 网络安全

在本节中，我们将关注那些基于云的架构、服务的安全分类（参见图 16.1.1）。

身份和访问管理（IAM）　包括管理企业资源访问的人、过程和系统。它们确保一个实体的标识被确认，并基于这个确认的标识使其获取相应级别的访问权限。身份管理的一个方面就是身份提取，它要给标识用户提供访问，随后对于那些没有被企业指定访问企业云中数据权限的客户撤销或拒绝访问。对于云来说身份管理还可以采用纳入到联合身份管理方案中（参见第 15 章）的方式，该方案是客户使用的。其他方面，云服务提供商（CSP）必须能够进行身份标识和企业提供的身份之间的转换。

IAM 的访问管理包括认证和访问控制服务。比如，CSP 必须通过一种值得信任的方式认证用户。SPI 环境下的访问控制要求包括建立可信任的用户文件和策略信息，并使用它在云服务中进行访问控制，并在可审计的方式下进行。

数据丢失保护（DLP）　是在数据闲置、移动、使用时监控、保护、认证数据的安全性。许多 DLP 方法都可以在客户端实施，这在 16.6 节中进行了讨论。CSP 提供 DLP 服务，比如实施一些策略，确认数据在一些环境中哪些功能可以实施。

Web 安全　提供一种实时的防护。它不仅出现在用户端进行应用程序软件安装时，还出现在云中使用 CP 代理或重定向 Web 流量时。它在实体之上提供了一层保护，如反病毒软件阻止恶意软件通过浏览 Web 等方式进入企业。另外，为了防范恶意软件，一个基于云的安全服务应该保护策略实施、数据备份、流量控制和 Web 访问控制。

一个 CSP 可能提供一个基于 Web 的电子邮件服务，它需要相应的安全措施。电子邮件安全提供了对带内和带外电子邮件的控制，保护企业免受钓鱼、恶意附件的攻击，实施可接受使用、垃圾邮件保护等企业策略。CSP 在每个邮件客户端上放置了数字签名，并提供了加密选项。



图 16.11 云安全即服务的要素

安全评估 是云服务中的第三方审计。尽管这超出了 CSP 的范围，但是 CSP 提供了工具和访问点来满足各种评估活动。

侵入管理 包含入侵检测、保护、响应。它的核心是在云中实体和服务中实施入侵检测系统（IDS）和入侵防御系统（IPS）。一个 IDS 就是一系列的自动化工具来检测未授权用户访问一个主机系统。我们将在第 21 章中进行讨论。IPS 包含 IDS 的功能，同时还包含限制入侵者流量的机制。

安全信息和事件管理（SIEM） 从虚拟、真实的网络、应用程序、系统中聚合日志和数据事件（通过推和拉的模式）。这些信息被统计、分析来支持对那些可能需要响应或是干涉的信息、事件的实时报告和修改。CSP 通常提供一个整合的服务，它可以将来自多个源（包括云中和企业内部网络）的信息进行收集。

加密 是一种常见的服务，它可以为闲置时的数据、电子邮件流量、特殊用户网络管理信息和身份信息提供服务。CSP 提供的加密服务包括许多复杂的问题，如密钥管理、如何在云中实施 VPN，应用程序加密和数据环境访问。

业务连续性和灾难恢复 包含确保在任何服务中断事件发生时可恢复的方法和机制。这是一个 CSP 能从服务客户直接获取盈利的领域[WOOD10]。CSP 可以在多个地方提供备份，以提供可靠的失效转移和灾难恢复功能。这种服务必须包含一个灵活的数据基础设施、功能和硬件上的冗余、可监控的操作、地理上分散的数据中心和网络存活能力。

网络安全 包括指定访问、分配、监控、保护底层的资源服务。服务包含边界和服务器防火墙、拒绝服务保护。本节列出的许多其他服务都对网络安全服务有所帮助，包括身份和访问管理、数据丢失保护、Web 安全、侵入管理等。

16.8　云计算安全问题应对

迄今为止，已经有大量的文档为云计算相关的安全问题提供指导性意见。除了提供全局规范的 SP 800-144 外，NIST 还发布了 SP 800-146（云计算纲要和建议，2012）。NIST 的建议系统化考虑到了商用云服务的各种主要类型，包括基础设施即服务，平台即服务，软件即服务。由于根据云服务种类不同，安全问题也存在些许不同，NIST 发布了独立于云服务种类的多个建议。毫无疑问，NIST 建议选择满足如下条件的云服务提供商：使用高强度加密，采用合适的空间冗余机制，使用身份认证机制，让用户对保护他们的机制有足够清晰的认知。SP 800-146 还列出了云计算环境中分配给不同云中角色的全局安全控制。如表 16.4 所示。

由于云服务被越来越多的企业合并入它们的网络基础设施，云计算安全也随之成为一个重要问题。云计算安全事故的种种例子给追求商业利益的人们敲响了警钟，同时也促使云服务提供商慎重选用安全机制来减轻潜在用户的担忧。一些云服务提供商已经在数据层上采取行动来解决用户关于可用性和冗余度的担心。由于许多企业仍然不愿意大规模地接受云计算，云服务提供商还需要持续努力，使潜在客户相信其核心业务流程和关键任务应用的计算支持可以被安全地转移到云上。

表 16.4　安全控制功能和类别

技术方面	操作方面	管理方面
访问控制	意识和培训	认证、鉴别和安全评估
审计和问责	配置和管理	规划风险评估
身份识别和认证	应急计划	系统与服务获取
系统与通信保护	事件响应	
	日常维护	
	介质保护	
	物理和环境保护	
	人员安全系统和信息完整性	

16.9　关键术语、思考题和习题

关键术语

访问请求者（AR）
认证服务器
云
云审计者
云破坏者
云运营商
云计算
云消费者
云提供者
社区云
动态主机配置协议（DHCP）
EAP 认证者
EAP-IKEv2
EAP over LAN（EAPOL）
EAP 方法
EAP 传递模式
EAP 请求者
EAP-TLS
EAP-TTLS
可扩展认证协议（EAP）
防火墙
IEEE 802.1X
媒体网关
网络访问控制（NAC）
网络接入服务器（NAS）
平台即服务（PaaS）
策略服务器
私有云
公共云
远程访问服务器（RAS）
安全即服务（SecaaS）
软件即服务（SaaS）
提供商
虚拟局域网（VLAN）
EAP-GPSK

思考题

16.1 给网络访问控制提出一个简单的定义。

16.2 什么是 EAP?

16.3 列出并简单定义 4 种 EAP 认证方法。

16.4 什么是 EAPOL?

16.5 IEEE 802.1X 的功能是什么?

16.6 对云计算进行定义。

16.7 列出并简单定义三种云服务模式。

16.8 什么是云计算相关基础设施?

16.9 描述一些主要的对于云的安全威胁。

习题

16.1 调查你所在学校使用的网络访问控制方案。画出草图并指出其中的主要组件。

16.2 图 16.3 指出 EAP 可以在一个四层模型中进行描述。参考 RFC 3748 说明四层的格式和功能。

16.3 找出并观看关于云安全的相关视频。记录下你认为在这方面做得比较好的三个视频。如果你只能和同学分享一个，你会选哪个？为什么？使用（250 ~ 500 字）文章或 3 ~ 4 张 PPT 来总结你的推荐理由。

第 17 章　传输层安全

学 习 目 标

通过本章的学习应具备以下能力：

- 总结出 Web 安全威胁和 Web 流量安全方法
- 对传输层安全（SSL）协议进行概述
- 理解安全套接层和传输层安全的区别
- 将传输层安全中使用的伪随机函数和本书之前讨论的函数做比较
- 对 HTTPS（SSL 之上的 HTTP）进行概述
- 对 SSH（Secure Shell）进行概述

实际上，几乎所有的商业机构、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，大家越来越意识到这种现实，因此安全 Web 服务应运而生。

Web 安全性的话题非常广泛，本章首先讨论 Web 安全性的普遍需求，然后集中讨论三种在 Web 商务中越来越重要并且主要专注于传输层安全的标准模式：SSL/TLS，HTTPS 和 SSH。

17.1　Web 安全性思考

WWW 本质上是一种运行于互联网和 TCP/IP 的客户/服务器应用。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。但是，Web 使用情况的以下特点表明需要量身定制的安全工具：

- 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。
- 通常使用 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 安全威胁

表 17.1 总结了在使用 Web 时将要面临的一些安全威胁的类别。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，本书第六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 安全性威胁对照表

	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 • 修改传送中的消息	• 消息丢失 • 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 • 窃取服务器数据 • 窃取客户端数据 • 窃取网络配置信息 • 窃取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 • 用假消息使机器溢出 • 填满硬盘或内存 • 使用 DNS 攻击来孤立机器	• 中断 • 干扰 • 阻止正常工作	难以防止
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（IPsec）[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。

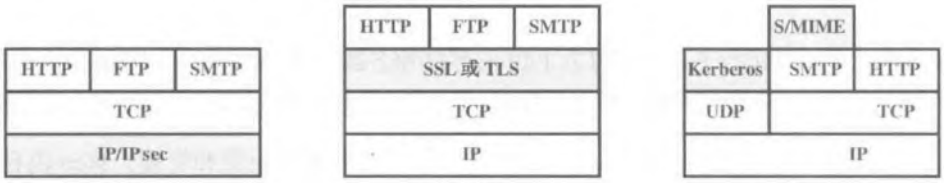


图 17.1 TCP/IP 协议栈中安全功能的相对位置

另一种解决方案就是在 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（Secure Sockets Layer，SSL）和被称为第二代互联网标准的传输层安全协议（Transport Layer Security，TLS）。对应的有两种实现方法，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)是该架构的示意图。这种方法的好处在于它是为给定应用定制的。

17.2 传输层安全

安全服务中应用最广泛的是被称为传输层安全（TLS）的 Internet 标准，其当前版本是 v1.2，在

RFC 5246 中定义。TLS 是在著名的商用协议安全套接层（SSL）上发展起来的 Internet 标准。尽管 SSL 仍广泛存在，但它被 IETF 组织和大部分提供 TLS 软件的公司所反对和弃用。TLS 是一个通用服务，由依赖于 TCP 的一组协议实现。在这一层，有两个实现选择。TLS 可以被作为下层协议套件而且对应用程序是透明的。或者，TLS 可以被嵌入在特定的包里。比如说，绝大部分浏览器开始配备了 TLS，而且绝大部分 Web 服务器实施了这个协议。

17.2.1　TLS 体系结构

TLS 为 TCP 提供了可靠的端到端安全服务。TLS 不是简单的单个协议而是两层协议，如图 17.2 所示。

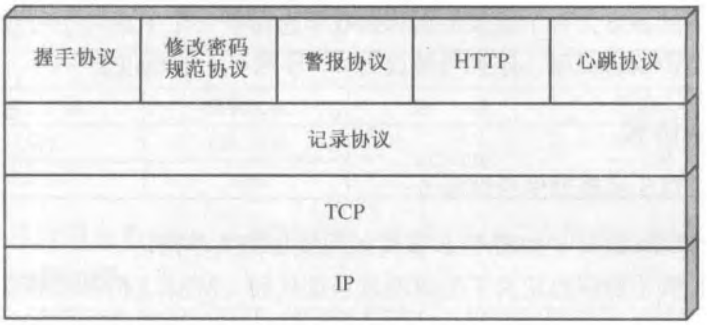


图 17.2　SSL 协议栈

TLS 记录协议（TLS 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 TLS。TLS 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 TLS 上层协议用于对 TLS 交换进行管理。第四个协议心跳协议，在独立的 RFC 中定义，将在本节后面进行讨论。

TLS 中包含两个重要概念：TLS 会话和 TLS 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 TLS 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**　TLS 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（如客户端和服务器间 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，会创建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：

- **会话标识**　服务器用于标识活动或恢复的会话状态所选的一个随机字节序列。
- **对等体证书**　对等体的 X.509.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。
- **主密钥**　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。

一个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。
- **初始化向量** 使用 CBC 加密模式时，需要为每个密钥维护一个初始化向量（IV）。该域首先被 TLS 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 TLS 记录协议

TLS 记录协议为 TLS 连接提供两种服务：

- **保密性** 握手协议定义了加密 TLS 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 例示了 TLS 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，将段分为块，压缩（可选）、加上 MAC、加密，再加上一个 TLS 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，再传递给高层用户。

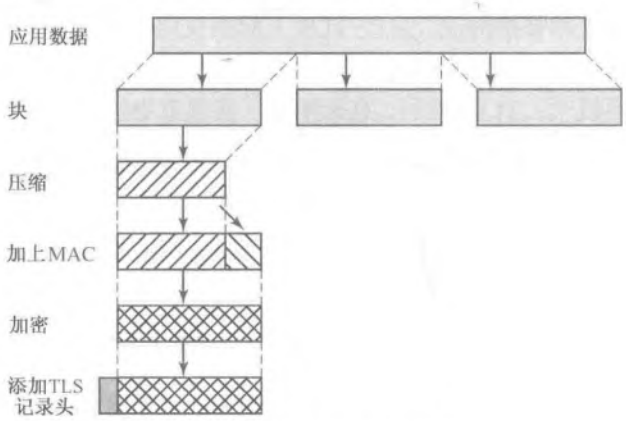


图 17.3 TLS 记录协议操作

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（16384 字节）的段；接着进行可选择压缩，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 TLSv2 中，没有指定压缩算法，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码 MAC。TLS 使用 RFC 2104 中定义的 HMAC 算法。第 12 章中提到，HMAC 计算方式如下：

$$\text{HMAC}_K(M) = H[(K^+ \oplus \text{opad}) \| H[K^+ \oplus \text{ipad}) \| M]]$$

其中，H 为嵌入的散列函数（对于 TLS 为 MD5 或 SHA-1）；M 为向 HMAC 输入的消息；K^+为左边填充全 0 的密钥，使得其长度与散列码中的块长度一致（MD5 或 SHA-1 中块长为 512 位）；ipad

① 当然，都希望压缩的结果是数据量变少而不是变多。但是对非常小的数据块有可能由于格式的转换使压缩算法的输出比输入更长。

为十六进制 0x36（0011 0110），重复 64 次（512 位）；opad 为十六进制 0x5C（0101 1100），重复 64 次（512 位）。

对于 TLS，MAC 计算包含以下表达式的各域：

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。

接下来，将压缩消息和 MAC 用对称加密方法加密。加密对内容增加长度不能超过 1024 字节，以便整个长度不能超过 2^{14}+2048。以下加密算法是允许的。

块　密　码		流　密　码	
算　法	密钥大小	算　法	密钥大小
AES	128，256	RC4-128	128
3DES	168		

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的整数倍，最大 255 字节。例如，如果分组长度为 16 字节（如 AES），明文（或如果使用了压缩则为压缩文本）加上 MAC 长度为 79 字节，则填充长度可以是 1，17，33 等，直到 161。当填充长度为 161 时，总长度为 79+161=240。使用可变的填充长度可以用来阻止那些基于分析消息长度的攻击。

TLS 记录协议的最后一步是加上一个由如下域组成的 SSL 头部：

- **内容类型（8 位）**　封装段使用的高层协议。
- **主版本号（8 位）**　表明 TLS 使用的主版本号，如 TLSv2 的值为 3。
- **从版本号（8 位）**　表明 TLS 使用的从版本号，如 TLSv2 的值为 1。
- **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 TLS 并没有什么限制，它们提供的数据内容对 TLS 来说是不透明的。

图 17.4 例示了 TLS 记录格式。

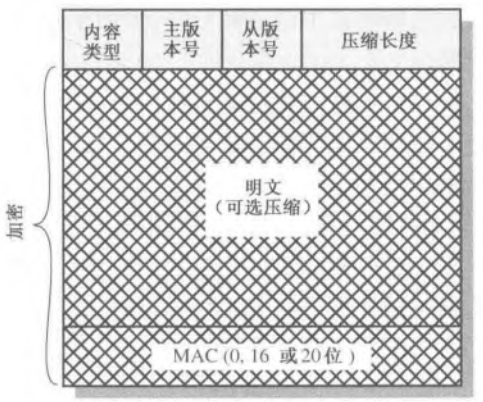


图 17.4　TLS 记录格式

17.2.3 修改密码规范协议

修改密码规范协议是 TLS 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复制到当前状态中，用于更新此连接使用的密码组。

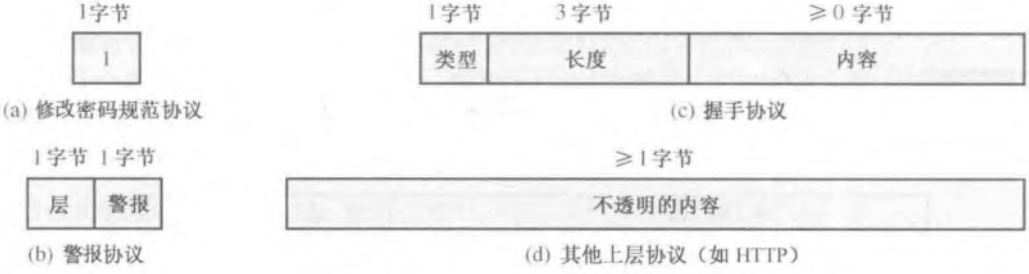


图 17.5 TLS 记录协议有效载荷

17.2.4 警报协议

警报协议用于向对等实体传递 TLS 相关的警报。和使用 TLS 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，传递消息出错的严重程度。如果级别为致命，则 TLS 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。通常导致致命错误的警报列举如下：

- **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数据长度）
- **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。
- **解密失败** 使用不正确的方法解密密文，或者长度不是块长度的整数倍或填充值不正确。
- **记录溢出** 接收的 TLS 记录中的载荷（密文）长度超过 2^{14}+2048 字节或者密文解密后长度超过 2^{14}+1024 字节。
- **未知的签证机构** 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签证机构匹配而不接收证书。
- **拒绝访问** 接收到合法证书，但发送者拒绝进行协商访问。
- **解码出错** 由于域超出了指定范围或消息长度不对使得消息不能解码。
- **输出限制** 密钥长度的输出限制不能达成一致。
- **协议版本** 客户端试图协商的协议版本可以识别但不能支持。
- **安全不足** 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** 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。

其余的警报如下：

- **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。

- **证书出错**　接收的证书被破坏（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。
- **解密出错**　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消息。
- **用户取消**　握手由于某些与协议错误的原因而被取消。
- **不再重新协商**　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。

17.2.5　握手协议

TLS 的复杂性主要来自于握手协议。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 TLS 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（≥0 字节）**　与消息相关的参数，如表 17.2 所示。

表 17.2　TLS 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 4 个阶段组成：

阶段 1：建立安全功能

此阶段用于建立初始的逻辑连接，并建立与之相关联的安全功能，客户端发起这个交换，发送具有如下参数的 **client_hello message** 消息：

- **版本**　客户端所支持的最高 TLS 版本。
- **随机数**　由客户端生成的随机数结构，用 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。这些值作为随机数，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。非零值意味着客户端想更新现存连接的参数或为此会话创建一个新的连接；零值意味着客户端想在新会话上创建一个新连接。
- **密码组**　按优先级的降序排列的、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 message 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。

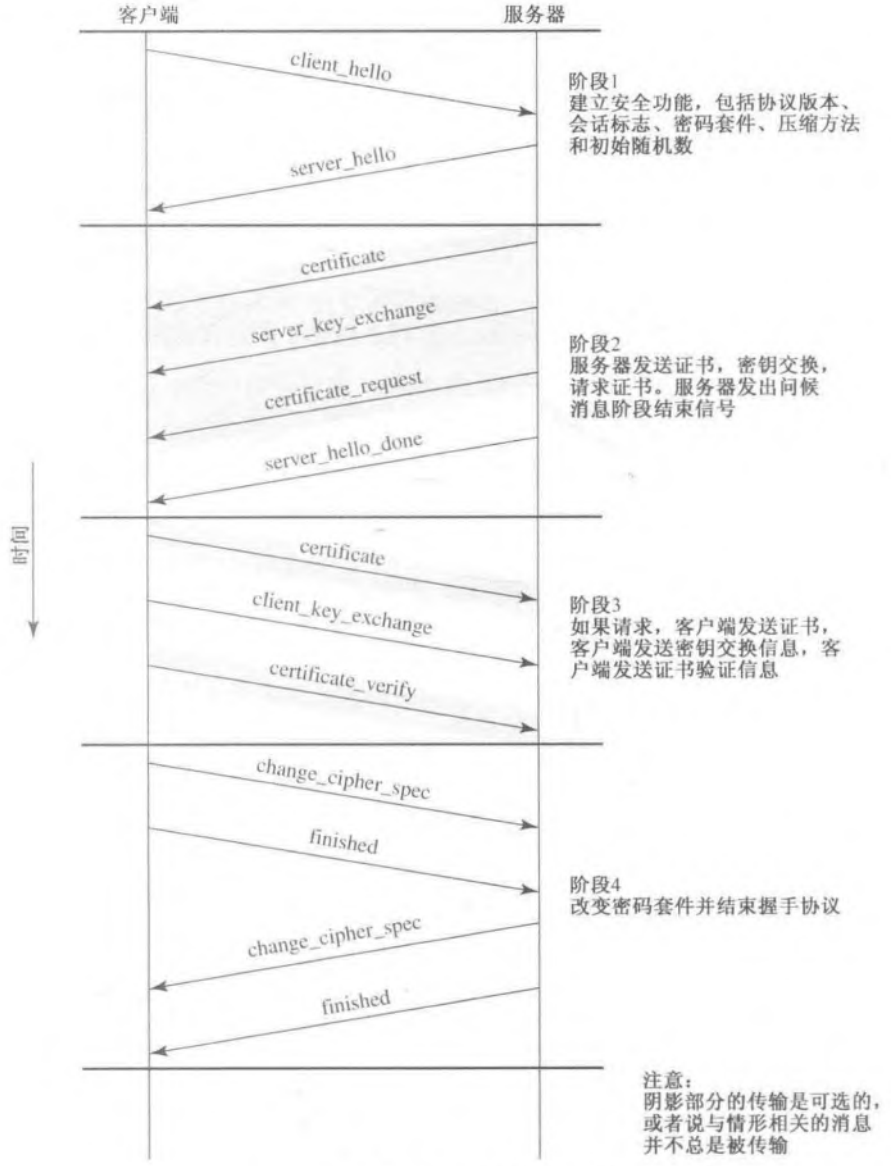


图 17.6 握手协议的处理过程

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，其中包含签证机构签发的 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。

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 或 IDEA
- **MAC 算法**　MD5 或 SHA-1
- **密码类型**　流或块
- **可否出口**　真或假
- **散列长度**　0，16（MD5）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 CBC 加密使用的初始向量的大小

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，此证书消息将由包含了服务器 Diffie-Hellman 公钥的参数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（primitive root）］和服务器 Diffie-Hellman 公钥（如图 10.1 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，参见图 9.5）和参数签名。

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 消息中的两个随机数，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：证书类型和签证机构。证书类型表明了公钥算法和它的用途：

（1）RSA：仅用于签名。
（2）DSS：仅用于签名。
（3）固定 Diffie-Hellman 的 RSA：此时，发送 RSA 签名证书，其签名仅用于认证。
（4）固定 Diffie-Hellman 的 DSS：仅用于认证。

证书请求消息中的第二个参数是一个可接受的签证机构名字表。

此阶段中的最后一个消息服务器完成消息 server_done message 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。它被用于生成稍后介绍的主密钥计算。
- **瞬时或匿名 Diffie-Hellman** 发送的客户端 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。

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的定义如下：

```
CertificateVerify.signature.md5_hash
        MD5 (handshake_messages);
Certificate.signature.sha_hash
        SHA (handshake_messages);
```

其中，握手消息 handshake_messages 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。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。客户端发送改变密码规范消息 change_cipher_spec message 并向当前 CipherSpec 中复制，以挂起 CipherSpec。注意，此消息不是握手协议的一部分，而是使用的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，完成消息的内容是

PRF(master_secret ,finished_label, MD5(handshake_messages) ||SHA-1(handshake_messages))

其中 finished_label 对于客户端是“client finished”，对于服务器端是“server finished”。

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 CipherSpec 中复制，以挂起 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 `pre_master_secret`，对于次密钥有两种可能：

- **RSA**　由客户端生成 48 字节的次密钥 `pre_master_secret`，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 `pre_master_secret`。
- **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 `pre_master_secret`。

然后，双方按如下方法计算主密钥 `master_secret`：

```
master_secret=
    PRF(pre_ master_secret,“master secret”,ClientHello.random||ServerHello.random)
```

其中，`ClientHello.random` 和 `ServerHello.random` 是在初始 hello 消息中交换的两个随机数。

算法执行直到生成 48 字节的伪随机输出。密钥分组数据（MAC 密钥，会话加密密钥，初始向量 IV）的计算定义为

```
key_block= PRF(SecurityParameters.master_secret, "key expansion",
    SecurityParameters.server_random || SecurityParameters.client_random)
```

直到生成足够的输出。

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、客户端写初始向量和服务器写初始向量，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，可以将主密钥看成该函数的伪随机种子值（seed value）；客户端和服务器随机数则可被视为是复杂密码分析方法的混淆值（salt value 的使用，参见第 21 章）。

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。其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 所示）：

```
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) = seed；A(i) = HMAC_hash(secret, A(i–1))。

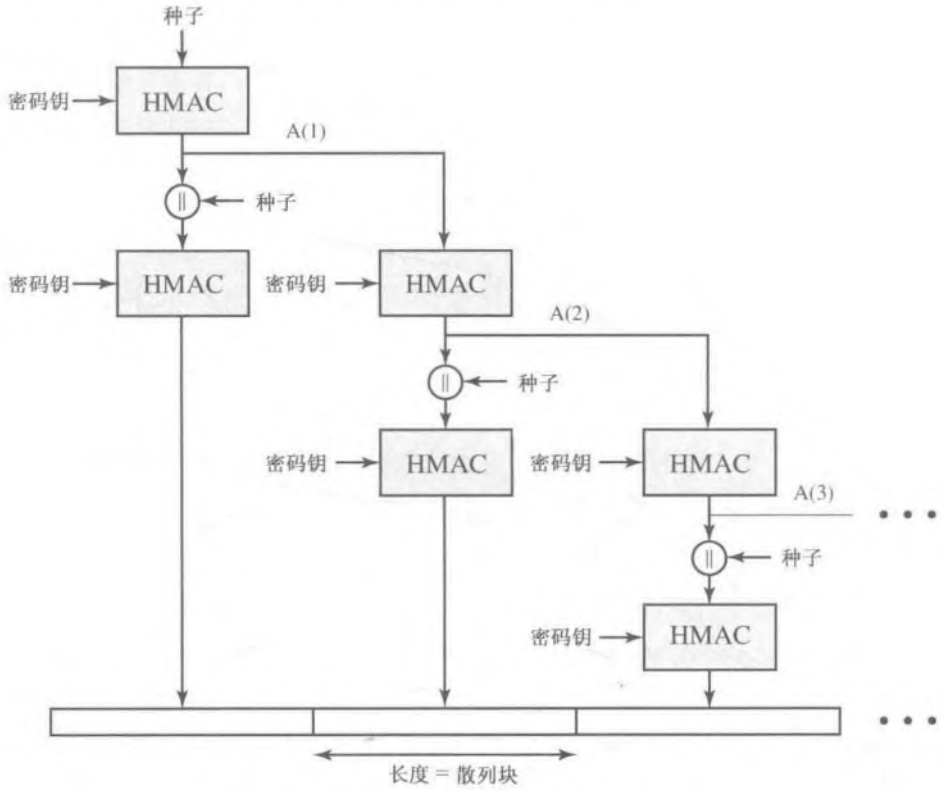


图 17.7 TLS 函数 P_hash(secret, seed)

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 256 生成 80 字节的数据，就需要迭代 3 次［通过 A(3)］，产生 96 字节的数据，略去最后的 16 字节。如果使用 P_MD5 则需要迭代 4 次，恰好产生 64 字节的数据。注意，每次迭代执行两次 HMAC，每次 HMAC 执行两次基本散列函数。

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 是安全的。PRF 定义如下：

PRF(secret, label, seed) = P_<hash>(secret, label || seed)

PRF 以密码值 secret、标识标志 label 和种子值 seed 为输入，产生任意长度的输出。

17.2.7 心跳协议

在计算机网络环境中，心跳是指由软件或硬件生成的周期性信号，用于指示常规操作或者同步系统内的不同部件。心跳协议可以典型地用于监督协议实体的运行。在 TLS 里，心跳协议于 2012 年在 RFC 6250（传输层安全和数据包传输层安全心跳扩展）中被定义。

心跳协议运行在 TLS 记录协议的上层，由两个消息类型组成：心跳请求和心跳回应。心跳协议的使用建立在握手协议的第一阶段中（如图 17.6 所示）。每个端表明其是否支持心跳协议。如果支持，此端表明其是否愿意接收心跳请求，回复心跳回应或者仅愿意发送心跳请求信息。

心跳请求消息可以随时发送。每当接收到一个心跳请求，节点应该立即回复相应的心跳回应消息。心跳请求消息包括载荷长度、载荷和填充域。其中载荷是 16B ~ 64 kB 大小的随机内容。对应的心跳回应消息里包括收到载荷的准确副本。填充的也是随机内容。填充域使得发送端能够通过不断增加发送请求的填充大小，直到一方主机无法处理消息而停止应答为止，以执行探测路径最大传输单元（MTU）的操作。

心跳有两个作用。第一，心跳能够使发送端确认接收端仍然活跃，即使一段时间内 TCP 连接上并没有任何活动。第二，处于空闲期时心跳能够在连接上生成活动，避免不接受空闲连接的防火墙将连接断开。

通过设计心跳协议来满足交换载荷的需求以支持数据包传输层安全协议（DTLS）的使用，DTLS 即无连接版本的 TLS。由于无连接服务存在丢包，载荷信息使得请求端能够将请求消息和响应消息匹配。简而言之，同样版本的心跳协议既适用于 TLS，也适用于 DTLS，心跳协议中的载荷 TLS 和 DTLS 都有需求。

17.2.8　SSL/TLS 攻击

自 1994 年发布 SSL 以及产生后续的 TLS 标准以来，这些协议已经被大量专门设计的攻击针对。每一次攻击的出现，都促使或是协议，或是加密工具，或是 SSL 和 TLS 的实施细节方面发生改变以应对这些威胁。

攻击种类　我们可以将这些攻击分为四类：

- **握手协议攻击**　早在 1998 年，一种基于利用数据格式和 RSA 加密方案实现漏洞攻击握手协议的方法就已经被提出[BLEI98]。针对防御它的对策，攻击方法做出了改良，结果不仅使防御对策失效还加速了攻击[BARD12]。
- **记录和应用数据协议攻击**　这些协议中的一些缺陷被发现，不得不通过补丁升级来应对相关的威胁。最近在 2011 年的一个例子中，研究人员 Thai Duong 和 Juliano Rizzo 演示了 BEAST（针对 SSL/TLS 的浏览器漏洞），把人们一直认为的理论缺陷转化为了实际攻击[GOOD11]。BEAST 利用了密码学里的一种称为选择明文攻击的攻击方法。攻击者选取与已知密文相联系的明文的猜测值来实施攻击。研究人员开发了实际的算法成功地执行了攻击，而后续的补丁也阻止了这种攻击。BEAST 攻击的作者同样是 2012 CRIME（压缩比信息泄露）攻击的作者，当数据压缩使用 TLS 标准时，这种攻击能够导致网站 cookies 的内容被恢复[GOOD12]。如果认证 cookies 的内容被这种攻击方法恢复，攻击者就能够在经过身份验证的 Web 会话中实现会话劫持。
- **PKI 攻击**　在 SSL/TLS 以及其他环境中，对 X.509 证书有效性的验证一直容易遭受各类攻击。例如，文献[GEOR12]表明用于 SSL/TLS 的常用库容易在证书验证过程中受到攻击。作者揭示了 OpenSSL，GnuTLS，JSSE，ApacheHttpClient，Weberknecht，cURL，PHP，Python 源代码的漏洞和建立在这些产品上的应用缺陷。
- **其他攻击**　文献[MEYE13]列出了一些不属于上面三种分类的攻击。其中一个例子是德国黑客组织 The Hackers Choice 在 2011 年提出的 DoS 攻击[KUMA11]。这种攻击通过超量的 SSL/TLS 握手请求给服务器带来了繁重的处理负载。建立新的连接和进行重新协商都会使系统负担增加。假定握手协议的计算工作主要在服务器端完成，这种攻击会造成服务器端比源设备多得多的系统负担，导致拒绝服务攻击。服务器会被迫持续重复计算随机数和密钥。

SSL/TLS 的攻防历史在基于互联网的协议中极具代表性。完美的协议和完美的实施策略从未出现。安全威胁和防御对策间的持续和反复推进了互联网协议的发展。

17.2.9 TLS v1.3

2014 年，IETF TLS 工作组开始了 TLS 1.3 版本的工作，首要目标是改进 TLS 的安全性。在撰写本文时，TLS v1.3 仍处在草案阶段，但其最终标准与目前的草案已经十分接近。TLS v1.3 区别于 1.2 版本的显著改变如下：

- TLSv1.3 移除了一些选项和功能的支持。移除了弃用功能的实现代码，减少了潜在的危险代码错误的几率，减少了受攻击面。移除项包括：
  - 压缩
  - 不提供认证加密功能的密码
  - 静态 RSA 和 DH 密钥交换
  - client_hello 中作为随机参数一部分的 32 位时间戳
  - 重新协商
  - 修改密码规范协议
  - RC4
  - MD5 和 SHA-224 用作签名
- TLSv1.3 使用 Diffie-Hellman 或者椭圆曲线 Diffie-Hellman 用来交换密钥，而禁止使用 RSA。使用 RSA 的危险在于一旦私钥被攻破，所有使用这组加密套件的握手都将被攻破。而使用 DH 或 ECDH 时，每一个握手对应一个新协商的密钥。
- TLSv1.3 允许一轮往返的握手，通过改变发送消息的顺序与建立一个安全的连接来实现。在加密套件协商完成前，客户端发送一条客户端密钥交换信息，这条信息包含密钥建立使用的密码参数。这使得服务器端可以在发送第一条响应消息前计算出供加密和认证使用的密钥。减少握手阶段中发送数据包的数量可以加快握手进程，同时减少了受攻击面。

这些改变将会改进 TLS 的效率和安全性。

17.3 HTTPS

HTTPS（SSL 之上的 HTTP）是指结合 HTTP 和 SSL 来实现 Web 浏览器和 Web 服务器之间的安全通信。HTTPS 已经融合在当今的 Web 浏览器中了，而其使用与否还取决于 Web 服务器是否支持 HTTPS 通信。例如一些搜索引擎就不支持 HTTPS。

Web 浏览器用户能看到的最基本的区别是在 URL（统一资源定位器）地址中是以 https://打头而不是 http://。一个普通的 HTTP 连接使用 80 端口，而当指定使用 HTTPS 时，用的却是 443 端口，这是用来唤醒 SSL 的。

当使用 HTTPS 时，如下的通信元素是加密处理的：

- 请求文档的 URL
- 文档的内容
- 浏览器格式的内容（格式是由浏览器用户决定的）
- 在浏览器和服务器之间传输的 cookies
- HTTP 报头的内容

HTTPS 的规范文档见 RFC 2818（HTTP over TLS），在使用 SSL 之上的 HTTP 和 TLS 之上的 HTTP 是没有根本性区别的，这两种方法的实现都成为 HTTPS。

17.3.1　连接初始化

对于 HTTPS 而言，用于 HTTP 客户端的代理和用于 TLS 客户端的代理是一致的。客户端在一个适当的端口初始化一个与服务器的连接然后发送 TLS ClientHello 来开始 TLS 握手。当 TLS 握手结束后，客户端会初始化第一个 HTTP 请求。所有的 HTTP 数据都将当作 TLS 应用数据发送。然后是包括保持连接在内的传统 HTTP 操作。

HTTPS 中有三层不同的含义。在 HTTP 层，一个 HTTP 客户端通过向下一层发送一个连接请求来请求与 HTTP 服务器建立连接。通常，下一层就是 TCP 层或者也可能是 TLS/SSL。在 TLS 层，TLS 客户端和 TLS 服务器之间会建立会话。该会话可以在任何时候支持多个连接。正如我们所看到的，一个 TLS 请求的建立是从建立客户端 TCP 实体和服务器端 TCP 实体的 TCP 连接开始的。

17.3.2　连接关闭

一个 HTTP 客户端或者服务器端可以通过在 HTTP 记录中加入如下行：Connection：close 来指示关闭连接。当该记录发送后，就意味着连接关闭。

HTTPS 连接的关闭要求 TLS 关闭与远程的对等 TLS 实体之间的连接，该过程要求关闭下一层的 TCP 连接。在 TLS 层，正确的关闭连接的方法是通信双方使用 TLS 警告协议发送 close_notify 警告。TLS 的实例必须在关闭连接之前初始化一个关闭警告的交换操作。一个 TLS 实例可能允许在一方发送了关闭警告而并未等待对方发送关闭警告而关闭连接，称之为“不完全关闭”。值得注意的是，如果一个用户这么做了，可能是为了之后再次使用该会话。而这也只能在应用层（通常是通过检查 HTTP 消息边界）知道了所有其关心的数据之后才行。

HTTP 客户端还必须处理这种情形，即下一层的 TCP 连接是在没有 close_notify 警告和 Connection: close 指示的情况下关闭的。这种情形可能是因为服务器端程序出错或者是通信出错导致的 TCP 连接的断开。而这种未声称而关闭的 TCP 可能会成为攻击者利用的对象。因此当这些情况发生时，HTTPS 客户端应该有一些安全的提醒。

17.4　SSH

SSH（Secure Shell）是一种保障网络通信安全的协议，并且简单易实现。最初的版本，SSH1 主要是提供一个安全的远程登录工具以取代 TELNET 和其他不安全的远程登录模式。SSH 也能提供 C/S 服务以及类似于文件传输和电子邮件的网络功能。新版本 SSH2 修补了在原版本中的漏洞，并在 RFC 4250 到 RFC 4256 之间的文档中定义。

SSH 客户端和服务器应用程序对大多数操作系统都可用。它可以作为远程登录或者 X 隧道技术的备选方法，同时也用于嵌入式系统之外的使用加密技术的应用程序中。

SSH 是按照三种构建在 TCP 之上的协议组织的，如图 17.8 所示。

- **传输层协议**　提供服务器认证、数据保密和带前向安全性的数据完整（例如，如果某个密钥在一次会话中泄密了，而该密钥不会影响到之前会话的安全性）。在传输层也提供可选的数据压缩。
- **用户认证协议**　为服务器认证用户。
- **连接协议**　在单一的低层 SSH 连接上提供多逻辑的通信信道。

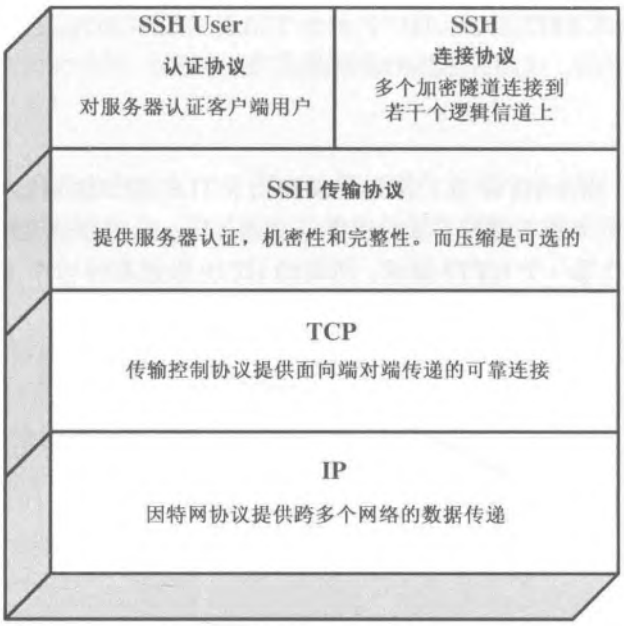


图 17.8 SSH 协议栈

17.4.1 传输层协议

主机密钥 服务器认证在传输层进行，是基于服务器持有一个公私密钥对的原理。一台服务器可能会有多个主机密钥用于多个不同的非对称加密算法。而多个主机又可能共享同一个主机密钥。在任何情况下，服务器的主机密钥都是在密钥交换阶段用来认证主机身份的。为了其可行性，客户端必须有服务器主机公钥的先验信息。RFC 4251 给出了两个可供选择的可信模型：

（1）在客户端拥有一个本地的数据库将每一个主机名（由用户输入的）与对应的主机公钥联系起来。该方法不需要集中的管理设施或第三方协助。不足是用户名和密钥对的数据库维护负担比较大。

（2）主机的用户名和密钥对的认证过程是由可信认证中心（CA）进行的。客户端只知道 CA 的根密钥并只能验证由 CA 认证的所有的主机密钥的合法性。这种可选择性降低了维护的难度，因为理论上只要在客户端安全的存储一个 CA 密钥。另一方面是，在认证成为可能之前每一个主机密钥必须有认证中心进行验证。

数据包交换 图 17.9 描述了 SSH 传输层协议中的事件序列。首先，客户端建立一个到服务器的 TCP 连接。这是通过 TCP 协议完成的，并且是不属于传输层协议的部分。当连接建立以后，客户端和服务器端就开始交换数据（也成为数据包报文，在 TCP 层的数据域）。每一个数据包都有如下的格式（如图 17.10 所示）。

- **数据包长度** 以字节为单位的数据包长度，不包含包长度域和 MAC 域。
- **填充长度** 随机填充域的长度。
- **有效载荷** 数据包的有用内容。在算法协商之前，该域是未压缩的。当压缩算法协商以后，下一个数据包的有效载荷域就会被压缩。
- **随机填充** 当协商成功一个加密算法时，则会附上该域。它包含了用于填充的随机字节，以使整个数据包（不包含 MAC 域）的长度是密码块大小的整数倍或者是 8 个字节的流密码。

- **消息认证码（MAC）**　当协商好消息认证，则该域就包含了 MAC 的值。MAC 是在这个数据包（不包含 MAC 域）和一个序列码上计算的。该序列码是一个 32 位的指示数据包序列的编码，并且将第一个数据包初始化为 0 而后续包的序列码则随之递增。序列码是不包含在数据包中在 TCP 连接链路上传输的。

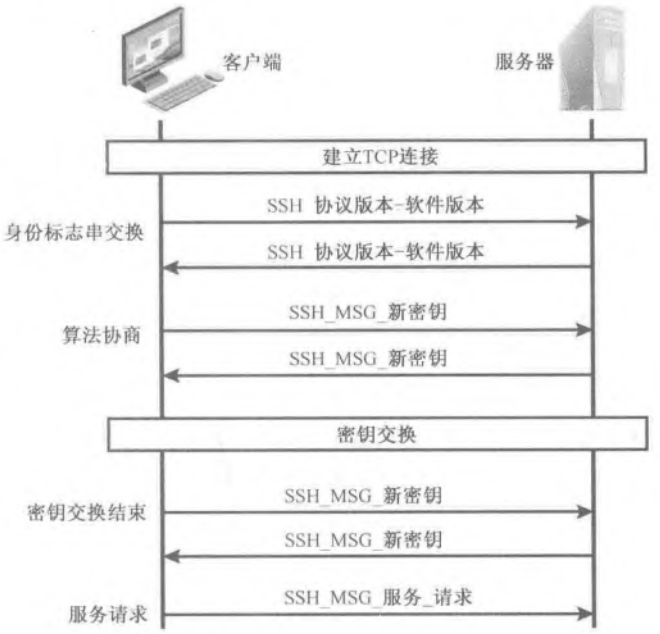


图 17.9　SSH 传输层包交换

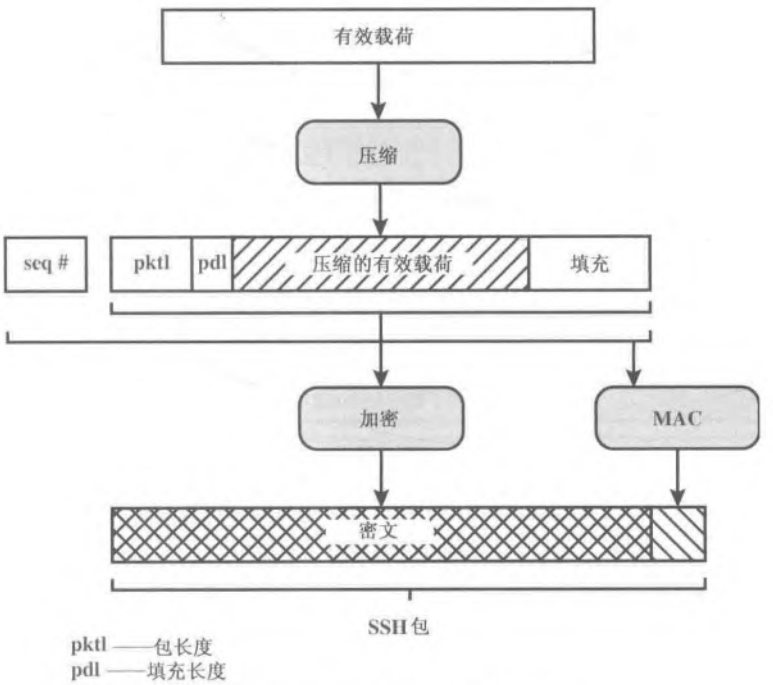


图 17.10　SSH 传输层包格式

当加密算法协商好并计算好 MAC 值之后，整个数据包（除 MAC 域之外）会被加密。

SSH 传输层数据包交换包含了一系列步骤（如图 17.9 所示）。第一步，标识串交换，客户端发送一个包含了标识串（形式如下）的数据包：

SSH-protoversion-softwareversion SP comments CR LF

其中，SP，CR 和 LF 分别表示空格键、回车符和换行符。例如一个合法的串是“SSH-2.0-billsSSH_3.6.3q3<CR><LF>”。然后服务器用自己的标识串响应。这些标识串用在 Diffie-Hellman 密钥交换中。

随后的是算法协商。通信双发各自发送一个 SSH_MSG_KEXINIT，其中包含了支持的算法的列表并以自己倾向程度进行排序。对每一种密码算法都有一个列表。这些算法包含密钥交换、加密、MAC 算法和压缩算法。表 17.3 显示了用于加密、MAC 和压缩的可选项。对每一类，所选择的算法是在客户端列表中的第一个算法并且服务器端也支持该算法。

下一步是密钥交换。规范文档允许有多种可选方法用来密钥交换，但如今，只有 Diffie-Hellman 密钥交换的版本可以使用。RFC 2409 定义了这两种版本并且对每个方向只需要一个包。在密钥交换中会涉及如下步骤。这里，定义 C 为客户端，S 为服务器端，p 是一个安全的大整数，g 是有限域 GF（p）的循环子群的生成元，q 是该循环子群的阶；V_S 是 S 的标识串，V_C 是 C 的标识串，K_S 是 S 的公开主机密钥，I_C 是 C 的 SSH_MSG_KEXINIT 消息，I_S 是 S 的 SSH_MSG_KEXINIT 消息（该消息在之前的部分已经交换了）。算法选择协商之后客户端和服务器端双方都知道 p，g 和 q 的值。Hash 函数 hash( )也在算法协商阶段被决定了。

表 17.3 SSH 传输层密码算法

密　码	
3des-cbc*	CBC 模式下的三密钥 3DES
blowfish-cbc	CBC 模式下的 blowfish
twofish256-cbc	CBC 模式下的 twofish，有 256 位密钥
twofish192-cbc	192 位密钥的 twofish
twofish128-cbc	129 位密钥的 twofish
aes256-cbc	CBC 模式下的 256 位密钥的 AES
aes192-cbc	192 位密钥的 AES
aes128-cbc**	128 位密钥的 AES
Serpent256-cbc	CBC 模式下 256 位密钥的 Serpent
Serpent192-cbc	192 位密钥的 Serpent
Serpent128-cbc	128 位密钥的 Serpent
arcfour	128 位密钥的 RC4
Cast128-cbc	CBC 模式下的 CAST-128

MAC 算法	
hmac-sha1*	HAMC-SHA1；摘要长度=密钥长度=20
hmac-sha1-96**	HAMC-SHA1 前 96 位；摘要长度=12；密钥长度=20
hmac-md5	HAMC-MD5；摘要长度=密钥长度=16
hmac-md5-96	HAMC-MD5 的前 96 位；摘要长度=12；密钥长度=16

压缩算法	
none*	无压缩
zlib	在 RFC 1950 和 RFC 1951 中定义

*为必须的　　**为建议的

- C 生成一个随机整数 x（$1<x<q$）并计算 $e=g^x \bmod p$。C 将 e 发送给 S。
- S 生成一个随机整数 y（$0<y<p$）并计算 $f=g^y \bmod p$。S 接收到 e，然后计算 $K=e^y \bmod p$，H=hash（V_C || V_S || I_C || I_S || K_S || e || f || K），然后用自己的主机私钥签名 H 形成 s。S 将（K_S || f || s）发送给 C。签名操作可能会涉及第二次散列运算。
- C 验证 K_S 确实是 S 的主机密钥（如使用证书或者本地数据库）。C 也允许在未进行验证情况下接受密钥，但是，这样做会使协议在应付主动攻击方面有脆弱性（但是可能在许多环境中短期内有实际意义）。然后 C 计算 $K=f^x \bmod p$，H=hash（V_C || V_S || I_C || I_S || K_S || e || f || K），然后验证 H 上的签名 s。

这些步骤结束后，通信双方就共享一个密钥 K。另外，服务器对客户端提供了认证，因为在 Diffie-Hellman 密钥交换中用自己的私钥进行了签名。最后，hash 值 H 就作为该连接的会话标识。计算完成之后，会话标识不变，即使当再次进行密钥交换来获取连接的新鲜密钥时。

密钥交换的结束阶段以交换 SSH_MSG_NEWKEYS 包为标志。此时，双方可能开始使用从 K 中生成的密钥，其过程随后阐述。

最后一步是服务请求。客户端发送一个 SSH_MSG_SERVICE_REQUEST 包来请求用户认证或者连接协议。这之后，所有的数据都将以 SSH 传输层包的负载进行交换，而数据包又由加密和 MAC 保护。

密钥生成　用于加密和 MAC（以及任何需要 IV 的时候）的密钥是从共享私密钥 K、从密钥交换 H 函数中获得的 Hash 值、会话标识生成的。其中会话标识和 H 是一致的，除非在初始密钥交换之后又有随后的密钥交换进行。其值的计算如下：

- 客户端到服务器的初始化值：HASH（K||H||"A"||session_id）
- 服务器到客户端的初始化值：HASH（K||H||"B"||session_id）
- 客户端到服务器的加密密钥：HASH（K||H||"C"||session_id）
- 服务器到客户端的加密密钥：HASH（K||H||"D"||session_id）
- 客户端到服务器的完整性密钥：HASH（K||H||"E"||session_id）
- 服务器到客户端的完整性密钥：HASH（K||H||"F"||session_id）

其中 HASH( )是在密钥协商阶段决定的散列函数。

17.4.2　用户认证协议

用户认证协议提供了用户向服务器证明自己的方法。

消息类型和格式　在用户认证协议中常使用三种类型的消息。从客户端发出的认证请求的格式：

byte	SSH_MSG_USERAUTH_REQUEST（50）
string	user name
string	service name
string	method name
…	method specific fields

其中 user name 是指用户名、客户端声称的认证标识，service name 是指客户端请求访问的设备（通常是 SSH 连接协议），method name 是请求中用来认证的方法。第一个 byte 的值为十进制的 50，表示该消息是 SSH_MSG_USERAUTH_REQUEST。

当服务器拒绝认证请求或者接受请求但是还需要更多的认证方法时，服务器会发出如下格式的消息：

byte	SSH_MSG_USERAUTH_FAILUER（51）
name-list	authentications that can continue
boolean	partial success

其中 name-list 是可能会继续对话的方法。当服务器接收认证，就发送一个 byte 消息：SSH_MSG_USERAUTH_SUCCESS（52）。

消息交换 消息交换涉及如下的步骤：

（1）客户端发送一个 SSH_MSG_USERAUTH_REQUEST。

（2）服务器查看并确认 user name 是否合法。若不合法，服务器返回 SSH_MSG_USERAUTH_FAILUER 和请求失败的部分成功值；若合法，服务器执行步骤 3。

（3）服务器返回 SSH_MSG_USERAUTH_FAILUER 和将会使用的认证方法列表。

（4）客户端选择一种可以接受的认证方法并发送 SSH_MSG_USERAUTH_REQUEST 以及该方法名和需要的方法指定域。这里会有一系列的信息交换来执行方法。

（5）若认证成功并需要更多的认证方法，则服务器转向执行步骤（3）并使用一个真实的部分成功值。若认证失败，则服务器执行步骤（3）并使用一个为假的部分成功值。

（6）当所有的认证方法都成功时，服务器发送一个 SSH_MSG_USERAUTH_SUCCESS 消息，并且认证协议结束。

认证方法 服务器可能会要求使用若干如下的认证方法：

- **公开密钥** 该方法的细节取决于采用的公钥密码算法。基本上是，客户端给服务器发送一个消息，其中包含了客户端的公钥，并用客户端的私钥签名该消息。当服务器接收到该消息就验证消息中提供的密钥是否能进行认证，如果可以就验证签名是否正确。
- **口令密码** 客户端发送一个消息，其中包含了明文形式的口令密码，而其又在传输层协议中被加密。
- **基于主机** 认证在客户端的主机上进行而不是客户端本身。因此，一个支持多客户端的主机可以提供该主机上所有客户端的认证。该方法中客户端发送一个由其所在的主机的私钥签名的消息。因此，SSH 服务器并不是直接验证用户的身份，而是认证了客户端主机的身份，并且当主机声称在客户机一方验证了该用户时，服务器就认为已经验证了用户。

17.4.3 连接协议

SSH 连接协议在 SSH 传输层协议的顶层运行，并假设使用了安全的认证连接[①]。安全的认证连接是指连接协议用一个通道虚拟出多条逻辑信道。

信道机制 使用 SSH 的所有类型的通信，例如终端会话，都支持使用独立信道。通信双方的任意一方都可以开启信道。对每一条信道，每一端都与一个唯一的信道序列码对应，即在同一条信道的两端不必使用同一个序列号。使用窗口机制对信道进行流量控制。只有当接收到表示窗口空间可用的消息时才能在信道中发送数据。

① RFC 4254，The Secure Shell (SSH) Connection Protocol，定义了连接协议运行在传输层协议和用于认证协议之上。RFC 4251，SSH Protocol Architecture，定义了连接协议运行在用于认证协议之上。事实上，连接协议是运行在传输层协议之上的，但是前提是假设用户认证协议已经在之前被唤醒了。

信道的生命周期有三个阶段：开启信道、数据传输和关闭信道。

当任何一端希望开启新的信道时，它就为信道动态分配一个本地序列号并发送如下格式的消息：

```
byte      SSH_MSG_CHANNEL_OPEN
string    channel type
uint32    sender channel
uint32    initial window size
uint32    maximum packet size
...       channel type specific data follows
```

其中 uint32 是指无符号 32 位整型。channel type 标识了该信道使用的应用程序（随后会介绍）。sender channel 是本地的信道序列号。initial window size 表示发送者在不调整窗口的情况下可以发送多少字节的数据。maximum packet size 指发送者发送的独立的数据包的最大尺寸。例如，一个用户可能希望使用小的数据包进行交互式的连接来获取在低速链路上的更好的交互回应。

当远程端可以开启信道时，它返回一个 SSH_MSG_CHANNEL_OPEN_CONFIRMATION 消息，其中包含了发送者的信道序列号、接收者的信道序列号、窗口数和数据包的大小。否则，远程端返回一个 SSH_MSG_CHANNEL_OPEN_FAILURE 消息和一个表示失败原因的报错码。

当一个信道开启时，就发送 SSH_MSG_CHANNEL_DATA 消息并执行数据传输。消息中包含接收者信道序列号和数据块。只要信道开启，这些消息可以任意方向传输。

当任意一方希望关闭信道时，就发送 SSH_MSG_CHANNEL_CLOSE 消息，其中包含了接收方信道序列号。

图 17.11 给出了连接协议消息交换的例子。

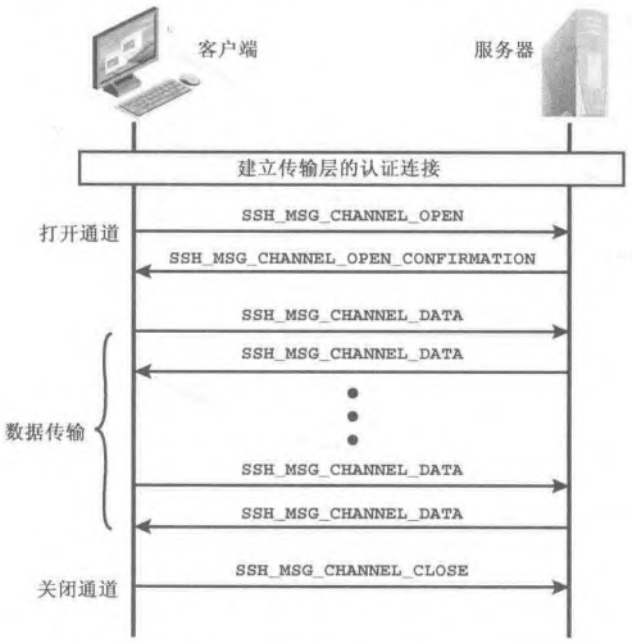


图 17.11　SSH 连接协议消息举例

信道类型　在 SSH 连接协议的规范文件中有 4 种信道类型：

- **会话**　程序的远程执行。这些程序可以能是 shell、诸如文件传输或者是电子邮件的应用程序、

系统命令和内置的子系统。当一个会话信道开启时，接下来的一些请求就用来开始运行远程程序。

- **x11** 这是指 X Window 系统，它是为网络计算机提供图形化用户界面（GUI）的一个计算机系统和网络协议。X 允许程序在网络服务器上运行并且在桌面机上显示。
- **前向 tcpip** 这是远程端口转发的，在下一节中会介绍。
- **直接 tcpip** 这是本地端口转发的，也在下一节中介绍。

端口转发 SSH 最有用的一个特征是端口转发。基本上，端口转发可以将任何的非安全的 TCP 连接转换成安全的 SSH 连接。这也被称为 SSH 的隧道技术。我们必须理解在该语境下的端口。端口是指一个 TCP 用户的标识。因此，任何在 TCP 顶层运行的应用程序都有一个端口号。TCP 链路就是在端口号的基础上向合适的应用程序发送数据的。一个应用程序可能会使用多个端口号。例如，简单邮件传输协议（SMTP），在服务器端是用 25 端口号监听的，于是一个 SMTP 请求使用 TCP 并发送数据到目的端 25 端口。TCP 识别这是 SMTP 服务器地址，于是就将数据路由至 SMTP 服务器应用程序。

图 17.12 描述了端口转发相关的基本概念。我们有一个用端口号 x 标识的客户端应用程序，并且服务器端的应用程序标识是端口号 y。在某时刻，客户端应用程序唤醒本地 TCP 实体并请求一个到端口号为 y 的远程服务器的连接。本地的 TCP 实体就与远程的 TCP 实体协商一个 TCP 连接，这样本地端口 x 就和远程端口 y 连接起来了。

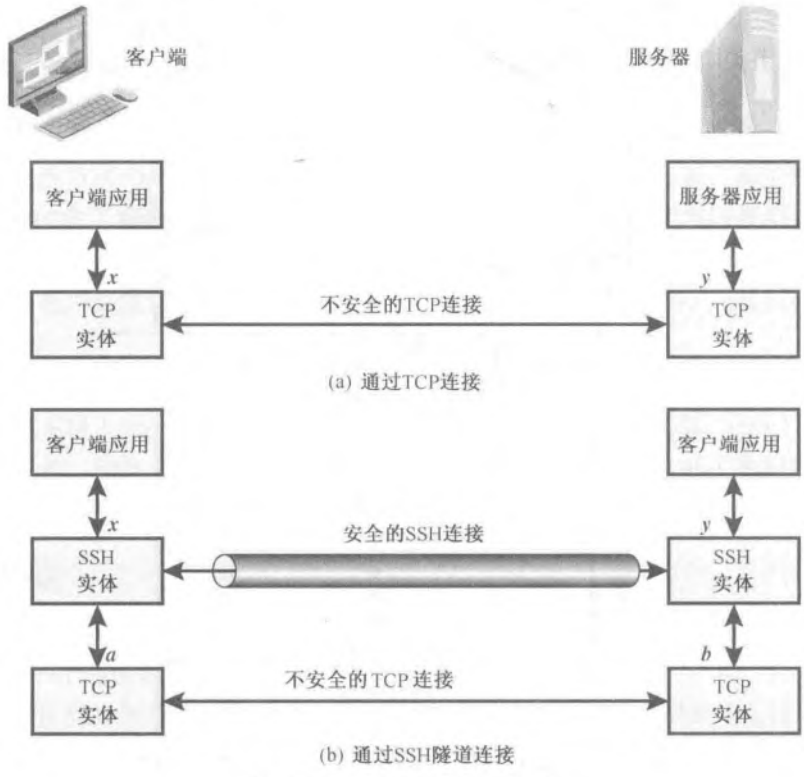


图 17.12 SSH 传输层包交换

为了使连接安全，需要配置 SSH 以使 SSH 传输层协议建立一个 SSH 客户端和服务器实体（分别用 TCP 端口号 a 和 b 表示）间的 TCP 连接。一个安全的 SSH 隧道就在 TCP 连接的基础上建立了。数据包从客户端 x 端口发出转到本地 SSH 实体，并通过隧道转发至远程 SSH 实体，再由远程 SSH 实体将数据转发到端口为 y 的服务器应用程序。相反方向的传输也类似。

SSH 支持两种类型的端口转发：本地转发和远程转发。本地转发允许客户端建立“hijacker”进程。该进程可以截获特定的应用层数据包，并将其从一个非安全的 TCP 连接转发到安全的 SSH 隧道。可以设置 SSH 监听特定的端口。SSH 可以抓取特定端口的所有数据包并通过 SSH 隧道将其发送。在另一端，SSH 服务器发送接收到的数据包到客户端应用程序指示的目的端口。

下面的例子可以帮助我们透彻理解本地转发。假若你在本地桌面机上有一个电子邮件的客户端程序，并使用它从你的邮件服务器通过邮政协议（POP）发送一个电子邮件。POP3 指定的端口号是 110。我们可以通过如下方法来使数据包安全的传输：

（1）SSH 客户端建立一个到远程服务器的连接。
（2）选定一个未使用的本地端口号，不妨设为 9999，配置 SSH 使其接收从该端口发出并且目标是服务器的 110 端口的数据包。
（3）SSH 客户端通知 SSH 服务器端创建一个到目标地址的连接，在这个例子中邮件服务器的端口号是 110。
（4）客户端发送任意位的数据到本地端口 9999 并让其将数据用 SSH 会话加密发送到服务器端。SSH 服务器解密收到的数据并将明文发送给端口 110。
（5）另一个方向，SSH 服务器获取任意位的从端口 110 接收到的数据，并将其在 SSH 会话中发送回客户端。客户端解密数据并将明文发送到与端口 9999 相连的进程。

使用远程转发，用户的 SSH 客户端代替服务器行动。客户端在一个给定的目标端口接收到数据包，将数据包放至正确的端口并将其发送到用户选择的目的地址。如下是一个典型的远程转发的例子。你希望在家里访问一台办公室的服务器。因为办公室的服务器在防火墙之后，它将不接受你家里计算机发出的 SSH 请求。但是，可以使用远程转发从办公室建立一个 SSH 隧道。如下是具体步骤：

（1）在办公室计算机上建立一个到家里计算机的 SSH 连接。防火墙是允许这样做的，因为这是向外的连接。
（2）配置 SSH 服务器使其监听本地端口，不妨设为 22，并通过 SSH 连接分发数据到远程的端口，设为 2222。
（3）你现在可以使用家中的计算机将 SSH 配置为可以接收端口 2222 的数据包。
（4）这样就有了一个 SSH 隧道可用于远程登录办公室服务器。

17.5　关键术语、思考题和习题

关键术语

警报协议	密码变换协议	握手协议	HTTPS（SSL 上的 HTTP）
主密钥	安全外壳（SSH）	安全套接层（SSL）	传输层安全（TLS）

思考题

17.1　指出图 17.1 中每种方法的优点。
17.2　TLS 包含哪些协议？
17.3　TLS 连接和 TLS 会话的区别是什么？
17.4　列举并简单定义 TLS 会话状态的参数。
17.5　列举并简单定义 TLS 会话状态的连接。

17.6 TLS 记录协议提供哪些服务？

17.7 TLS 记录协议传输有哪些步骤？

17.8 HTTPS 的目的是什么？

17.9 哪些应用程序可以使用 SSH？

17.10 列举并简单定义 SSH 协议。

习题

17.1 在 SSL 和 TLS 中，为什么有分离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 在 TLS 修改密码规范协议中的 MAC 服务的作用是什么？

17.3 考虑以下 Web 安全性威胁，并描述 TLS 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 TLS 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，攻击者代替一方的主机进行通信。

（h）SYN 洪：攻击者发送 TCP SYN 消息请求连接，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 SYN 消息可以阻塞 TCP 模块。

17.4 利用本章中学到的知识，回答：在 TLS 中接收者在接收到顺序紊乱的 TLS 记录块时能为它们排序吗？如果可以，解释它是如何做的；如果不可以，为什么？

17.5 对于 SSH，包含 MAC 之内的数据包加密方法有哪些优势？

第 18 章　无线网络安全

学 习 目 标

通过本章的学习应具体以下能力：

- 对无线网络的安全威胁和对策进行概述
- 理解由于企业网络移动设备的使用而形成的独特的安全威胁
- 描述移动设备安全策略中的主要因素
- 理解 IEEE 802.11 无线网络标准的必需元素
- 总结 IEEE 802.11i 无线网络安全体系结构的各种组成元素

首先，我们对无线安全问题做个总体的概览。随后关注移动设备安全中相对较新的领域，为在企业内部使用的移动设备检查威胁并制定对策。然后，我们着眼于无线网络安全的 IEEE 802.11i 标准。这个标准是 IEEE 802.11 的一部分，也与 Wi-Fi 相关。我们从 IEEE 802.11 的概述开始讨论，然后再看 IEEE 802.11i 的细节。

18.1　无线安全

无线网络和使用无线网络的无线设备，引起了许多有别于有线网络的安全问题。与有线网络相比，在无线网络中安全风险较高的关键因素主要包括[MA10]：

- **信道**　无线网络通常涉及广播通信，这个相比有线网络来说更容易受窃听和干扰影响。无线网络对恶意挖掘通信协议漏洞的主动攻击来说也更为脆弱。
- **移动性**　在理论和实际中，无线设备相对于有线设备来说都更轻便，有更好的移动性。但是，移动性导致了很多安全威胁，随后细述。
- **资源**　一些无线设备，如智能手机和平板电脑，有复杂的操作系统但是其有限的内存和处理资源会导致安全威胁，包括拒绝服务攻击和恶意软件。
- **易接近性**　一些无线设备，如传感器和机器人，可能被留在无人照管的边远区或是敌占区。这大大增加了它们对物理攻击的脆弱性。

简言之，无线环境由提供攻击点的三部分组成（如图 18.1 所示）。无线用户可以是一个手机，一个有 Wi-Fi 功能的便携式计算机或者掌上电脑，一个无线传感器，一个蓝牙设备等。接入点的例子可以是手机信号塔，Wi-Fi 热点和可以接入有线局域网或者广域网的无线接入点。传输媒介负责为数据传递传输无线电波，这也是脆弱性的来源。



图 18.1　无线联网组成部分

18.1.1 无线网络威胁

文献[CHOI08]列出了如下无线网络的安全威胁：

- **意外关联** 公司的无线 LAN 或是有线 LAN 的无线接入点很靠近（如在同一个或者相邻的建筑内）可能造成覆盖范围的重叠。一个试图连接到某一局域网的用户可能无意地关联了隔壁网络的一个无线接入点。尽管这个安全漏洞是意外的，然而这个漏洞却将局域网的资源暴露给了这个意外的用户。
- **恶意关联** 在这种情形下，一个无线设备被配置伪装成一个合法的接入点，以此让操作者从合法用户那里窃取密码，然后通过一个合法的无线接入点渗透到有线网络中。
- **自组网络** 它被定义为无线计算机之间的对等网络，在它们之间没有一个接入点。这种网络可能会由于缺少中央控制点而造成安全威胁。
- **非传统网络** 非传统网络和链接，如个人网络蓝牙设备、条形码阅读器、掌上电脑，都会因为窃听和欺骗而导致安全风险。
- **身份盗窃（MAC 欺骗）** 身份盗窃发生时，一个黑客能够监听网络数据流，并能使用网络权限确定一台计算机的 MAC 地址。
- **中间人攻击** 这类攻击在第 10 章中的 Diffie-Hellman 密钥交换协议中被描述过。更广泛的意义，这种攻击包括说服用户和接入点相信他们是在和对方通信，但事实上他们是在和一个中间攻击设备在通信。无线网络特别易受此类攻击。
- **拒绝服务攻击（DoS）** 这种攻击将在在线部分的第 21 章详细介绍。无线网络环境下，拒绝服务攻击发生时，攻击者不断地轰击目标的无线接入点或其他无线端口，发送各种请求系统资源的协议消息。无线环境适用于这种攻击，因为攻击者向目标发送多个无线消息是很简单的。
- **网络注入** 网络注入攻击目标无线接入点，而这类接入点通常是暴露无过滤能力的网络流中的，如路由协议消息和网络管理消息。这种攻击的一个例子是使用伪造的重置命令来影响路由器和交换机以降低网络的性能。

18.1.2 无线安全措施

按照文献[CHOI08]，我们把无线安全措施分为三类，分别处理无线传输、无线接入点和无线网络（包括无线路由和端点）。

安全无线传输 无线传输的主要威胁是窃听、消息的修改或插入以及破坏。为了解决窃听问题，有两种合适的对策：

- **信号隐藏技术** 组织机构可以采取很多方法使得攻击者更加难以定位他们的无线接入点，包括关闭设备通过无线接入点的设备标识集（SSID）的广播；给 SSID 分配神秘的名字；减少信号的强度到提供必备覆盖的最低水平；将无线接入点放置在建筑物内部，远离窗户和外墙。更有效的方法可以通过定向天线和信号屏蔽技术来实现。
- **加密** 对所有无线传输加密能够有效应对窃听，前提是加密密钥是安全的。

加密和认证协议的使用是应对消息的修改或插入攻击的标准方法。

在线部分的第 21 章中讨论的处理 DoS 的方法适用于无线传输。组织机构也可以减少无意 DoS 攻击的风险。现场勘测可以检测使用相同频率的其他设备，有助于决定在哪里放置无线接入点。信号强度可以被调整和屏蔽，从而使无线环境从与相邻传输争用中隔离出来。

安全无线接入点 设计无线接入点的主要威胁是对网络的未授权访问。组织这类接入的主要方

法是 IEEE 802.1X 标准基于端口的网络接入控制。这个标准为希望连接到局域网或无线网络的设备提供了认证机制。802.1X 的使用可以防止恶意接入点和其他未授权的设备成为不安全的后门。

16.3 节提供 802.1X 的介绍。

安全无线网络　文献[CHOI08]为无线网络安全推荐了如下技术：

（1）使用加密。大多数的无线路由都为从路由器到路由器传输内置了加密机制。

（2）使用防病毒软件和反间谍软件以及防火墙。这些设备应该配置到所有无线网络终端。

（3）关闭标识符广播。无线路由通常被配置成发送广播标识符信号，这样使得在它覆盖范围内的设备都能知道它的存在。如果网络要求被配置为只有授权的设备才能知道路由的标识符，那么这个功能就需要被关闭，以阻止攻击者。

（4）改变路由器上默认标识符。这个方法也是用来阻止试图通过默认路由标识符获得无线网络接入的攻击者。

（5）改变路由预设密码以便管理。这是另一个明智的步骤。

（6）只允许特定的计算机接入到你的无线网络。路由可以被配置为只与被认可的 MAC 地址通信。当然，MAC 地址可能被欺骗，因此这个仅仅是安全策略的元素之一。

18.2　移动设备安全

在智能手机广泛使用之前，在一个组织机构中，计算机和网络安全占主导地位的模式通常如下所述。各 IT 公司是严格控制的，用户设备通常仅限于 Windows PC。商业应用程序被 IT 控制，运行在终端或在数据中心的物理服务器上。网络安全是以清晰定义的从不可信网络中分离出可信网络的边缘为基础的。如今，所有这些假设都发生了巨大的变化。一个组织的网络必须容纳以下内容：

- **越来越多的新设备**　组织正在经历雇员移动设备使用的大量增长。在许多情况下，雇员被允许使用终端设备的组合来作为他们日常活动的一部分。
- **基于云的应用程序**　应用程序不再单独只在公司数据中心的物理服务器上运行。相反，应用程序可以在任何地方运行：在传统物理服务器上，在移动虚拟服务器上，或者是在云中。另外，终端用户现在可以利用一系列基于云的应用程序和 IT 服务来作为个人和专业使用。Facebook 可以用来做雇员个人资料，或者是作为一个企业的营销活动的组成部分。雇员依靠 Skype 来和国外的朋友对话或是合法的商业视频会议。Dropbox 和 Box 可以被用来在公司和个人设备间发放文档以增强移动性和用户工作效率。
- **边界模糊**　考虑到新设备增加、应用程序的移动性、基于云的客户和公司服务，静态网络的概念几乎已不存在。在设备、应用、用户和数据周围如今有大量的网络边界。由于要适应不同的环境条件，如用户角色、设备类型、服务器虚拟化移动性、网络选址和计时，因此这些网络边界也变得具有高度的动态性。
- **外部业务需求**　企业必须通过使用各个地点的各种设备为来为客户、第三方承包和商业伙伴网络提供网络接入。

所有这些变化的中心元素是移动计算设备。移动设备已经变成机构的必要元素，它是整个网络基础设施的一部分。移动设备，如智能手机、掌上电脑和记忆棒，为个人提供了便利，也为工作场所的效率提高增加了潜能。由于它们的广泛使用和独有的特性，移动设备的安全性是一个紧迫而复杂的事情。实质上，一个机构需要实施安全策略，它是通过建立在移动设备中的安全特性的组合，并且由控制移动设备使用的网络组件来提供辅助安全控制。

18.2.1 安全威胁

除了在其他终端装置实施的保护措施之外，例如在台式机和笔记本电脑，移动设备需要额外的、特殊的保护措施，这些设备只在机构设施内的网络上被使用。SP 800-14（企业中管理和保护移动设备指南，2012.07）列出了移动设备的 7 个主要安全关注点。我们依次调查每一种。

缺乏物理安全控制 移动设备通常在用户的完全控制下，使用和保存在各种组织的控制范围之外的地方，包括远程使用。即使设备被要求保持内部使用，使用者也可能在机构中将设备在安全区域和非安全区域间移动。因此，偷窃和干预是现实存在的威胁。

对于移动设备的安全策略必须基于移动设备可能被偷或者至少被恶意方访问的前提。威胁是双重的：恶意方可能试图去从设备本身得到敏感数据，或者使用该设备取得机构资源的访问权限。

使用不可信移动设备 除了公司发行和公司控制的移动设备，事实上所有的雇员都将有智能手机或掌上电脑。机构必须假设这些设备是不可信的。即设备可能没有使用加密，使用者或者第三方可能在安全构建限制操作系统使用中安装了旁路。

使用不可信网络 如果一个移动设备在内部部署使用，它可以通过组织所拥有的内部无线网络连接到组织的资源。但是，对于远程使用，使用者通常通过连接 Internet 和组织的 Wi-Fi 或者蜂窝来获得组织的资源。因此，包括远程控制部分的数据流对窃听和中间人之类的攻击是易受到潜在威胁的。因此，安全策略必须基于移动设备和组织之间的网络这一假设是不可信的。

使用未知方所创建的应用程序 通常，在移动设备上很容易找到并安装第三方应用程序。这显然地形成了安装恶意软件的风险。一个组织有几种选择来应对这种威胁，随后描述。

与其他系统交互 智能手机和掌上电脑的一个共同特征是与其他计算设备和云存储自动同步包括数据、app、联系人、照片等资源。除非组织控制了所有的同步设备，否则组织机构的数据将有很大的风险被存储在一个不安全地方，此外还会带来引入恶意软件的威胁。

使用不可信内容 移动设备可能会获取和使用其他计算设备没有存储的内容。一个例子是 QR（Quick Response）编码，它是个二维条码。QR 编码是用移动设备的摄像头捕捉并被移动设备使用。QR 编码转换为 URL，然后恶意的 QR 编码可以指引移动设备打开恶意的网址。

使用位置服务 移动设备上的 GPS 功能可以被用来获知这个设备的实际位置信息。尽管这个可能作为服务的一部分对组织是有用的，但是它也产生了安全风险。攻击者可以利用位置信息来判断设备和使用者的位置，这可能被攻击者利用。

18.2.2 移动设备安全策略

考虑到在之前讨论中列出的威胁，我们概述了移动设备安全策略的主要元素。它们分为三个种类：设备安全、客户/服务器通信安全和屏障安全（如图 18.2 所示）。

设备安全 一些组织会提供移动设备给雇员使用，并且将这些设备依照企业安全策略预配置。但是，一些组织会发现采用自带设备（BYOD）政策，即允许雇员的个人移动设备使用公司资源的会更方便或者更有必要。IT 管理人员应该能够在允许接入网络之前检查每个设备。IT 将希望为操作系统和应用程序建立配置指南。例如，root 和“越狱”的设备在网络上是不允许的，移动设备不能在本地存储公司联系信息。不论设备是归组织所有或 BYOD，组织都应该用如下安全控制来配置该设备：

- 启用自动锁定，即如果一定时间内设备没有使用就将其锁定，需要用户重新输入 4 位 PIN 码或是密码来重新激活该设备。
- 启用密码或 PIN 码保护。PIN 码或密码是用来解锁设备的。除此之外，它可以被配置为 E-mail 和设备上的数据都用 PIN 码或密码来加密并且只能用该 PIN 码或密码来恢复。

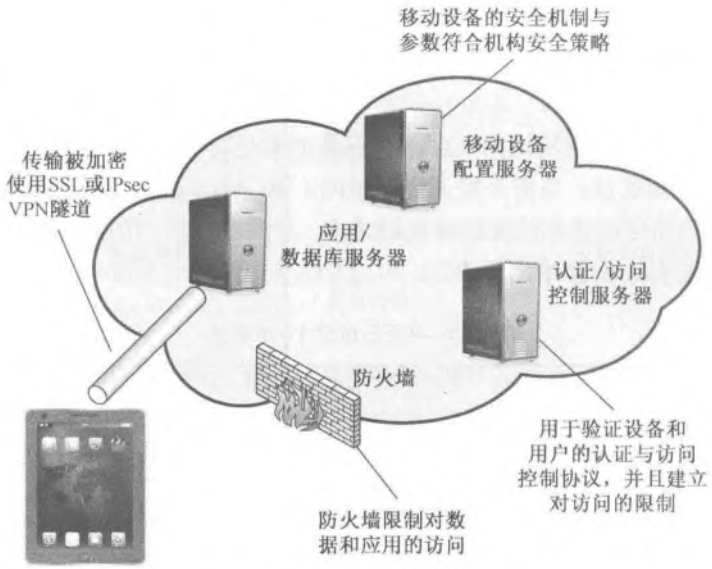


图 18.2　移动设备安全元素

- 避免使用记忆用户名或密码的自动完成功能。
- 启用远程清除。
- 如果可以，确保 SSL 保护启用。
- 确保软件，包括操作系统和应用程序是最新的。
- 安装反病毒软件。
- 敏感数据应该禁止存储在移动设备中或者应该被加密存储。
- IT 职员也可以远程接入设备，清除设备的所有数据，然后在设备丢失或被盗的情况下使其失效。
- 组织可能禁止所有第三方应用软件的安装，执行白名单来禁止未经证实的应用程序的安装，或者实施安全沙箱将组织的数据和应用程序从设备的其他所有数据和应用程序中隔离出来。任何在被认可列表中的应用程序都应该伴有数字签名和一个从认证中心获得的公钥证书。
- 组织可以实行限制决定什么设备可以同步和使用云存储。
- 为了解决不可信内容的威胁，安全响应可以包括人员在不可信内容内在风险的训练和关闭公司移动设备的摄像头使用。
- 为了应对位置服务恶意使用的威胁，安全政策可以命令此类设备在所有移动设备中不可使用。

数据流安全　数据流安全是基于加密和认证的常用机制。所有数据流都必须加密并通过安全方式传输，如 SSL 或 IPv6。虚拟专用网络（VPN）可以被配置成所有在移动设备和组织网络之间的通信是通过一个 VPN 的。

一个强健的认证协议可以被用来限制设备到组织资源的使用。通常，一个移动设备有一个单一的设备专用认证器，因为假定设备只有一个用户。一个更好的策略是有一个两层的认证机制，包括验证设备和验证设备的用户。

边界安全　组织应该有安全机制来保护网络不被非授权访问。安全策略也可以包括为移动设备通信特制的防火墙。防火墙可以限制所有移动设备的数据和应用程序的使用。同样地，入侵检测和入侵阻止系统可以被配置来使移动设备通信有更严格的规则。

18.3 IEEE 802.11 无线网络概述

IEEE 802 是一个开发了大量局域网（LAN）标准的委员会。在 1990 年，IEEE 802 委员会成立了新的工作组——IEEE 802.11，负责开发无线局域网（WLAN）的协议和传输规范。从那时起，开发了不同频段和不同数据传输速率的无线局域网需求。与需求同步，IEEE 802.11 工作组提出了一系列持续扩展的标准。表 18.1 简要给出了 IEEE 802.11 标准中使用的关键术语。

表 18.1 IEEE 802.11 术语表

接入点（AP）	任何具有基站功能并且提供通过无线媒体与已连接的基站访问分布系统的实体
基本服务集（BSS）	由一个单一的协调功能控制的基站的集合
协调功能	逻辑功能，决定基本服务集中运行的基站何时允许传输和可以接收 PDU
分布式系统（DS）	用来互联 BSS 并整合 LAN 以生成 ESS 的系统
扩展服务集（ESS）	若干个互联的 BSS 的集合，整合了 LAN 使任何与该 BSS 中之一相连的基站都觉得是单一的 BSS 到 LLC 层的连接
MAC 协议数据单元（MPDU）	使用物理层服务的对等 MAC 实体之间交换的数据单元
MAC 服务数据单元（MSDU）	在 MAC 用户之间传输的信息单元
基站（Station）	任何符合 IEEE 802.11 MAC 和物理层标准的设备

18.3.1 Wi-Fi 联盟

第一个广泛受到工业界支持的 802.11 标准是 802.11b。尽管 802.11b 的产品都是基于相同的标准生产的，但是仍然对不同生产商生产的产品的兼容性有质疑。为了解决该问题，无线以太网兼容性联盟（WECA）以一个工业团体的形式在 1999 年形成。该组织随后更名为无线保真（Wi-Fi）联盟并创建了一套用于验证 802.11b 产品兼容性的工具。验证 802.11b 产品的专业数据就是 Wi-Fi，Wi-Fi 的验证也成为了 802.11b 产品的外延。Wi-Fi 联盟还开发了 802.11a 产品的验证过程，称之为 Wi-Fi5。Wi-Fi 联盟关注与无线局域网相关的大部分领域，包括企业、家庭和热点。

最近，Wi-Fi 开发了 IEEE 802.11 安全标准的验证过程，称之为 Wi-Fi 受保护访问（WPA）。最新的 WPA 版本是 WPA2，整合了所有 IEEE 802.11i 无线局域网安全规范的特性。

18.3.2 IEEE 802 协议体系结构

在开始之前先简要介绍一下 IEEE 802 协议的体系结构。IEEE 802.11 标准是在层次化协议集的框架下定义的。用于 IEEE 802 标准的框架如图 18.3 所示。

物理层 IEEE 802 参考模型的最底层就是物理层，它包含了信号的编码/解码、位传输/接收等功能。另外，物理层还包括了传输介质的规范。在 IEEE 802.11 中还定义了频率带宽和天线的特性。

媒体访问控制层 所有的局域网都包含共享网络传输能力的设备集合。为了有序和高效的使用传输能力，对传输媒体的访问控制是必要的。这就是媒体访问控制（MAC）层的功能。MAC 层从上层协议[通常是逻辑链路控制（LLC）层]中接收数据块[也被称为 MAC 服务数据单元（MSDU）]。总之，MAC 层执行了如下的功能：

- 在传输方面，封装数据成帧［通常称为 MAC 协议数据单元（MPDU）］并带有地址和校验域。
- 在接收方面，拆封数据帧、执行地址辨别和错误检查。
- 管理对 LAN 传输媒体的访问。

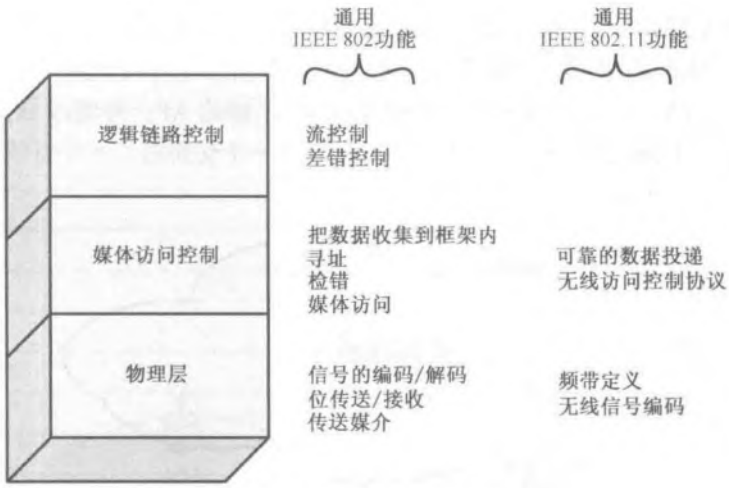


图 18.3　IEEE 802.11 协议栈

MPDU 的具体格式与所使用的 MAC 协议有关。但是基本上，所有的 MPDU 都有与图 18.4 相似的格式。这个框架有如下的域：

- **MAC 控制**　该域包含 MAC 协议功能需要的任何协议控制信息。例如，在这里会指示一个优先层。
- **目的 MAC 地址**　局域网中 MPDU 的目的物理地址。
- **源 MAC 地址**　局域网中 MPDU 的源物理地址。
- **MAC 服务数据单元**　上一层的数据。
- **CRC**　循环冗余校验域，也被称为帧校验序列（FCS）域。和用在其他数据链路层控制协议中一样，这是一种检错码。CRC 是在整个 MPDU 的位上计算的。发送者计算了 CRC 并将其附加在数据帧上。接收者在接收到的 MPDU 上执行相同的计算并对比该计算结果与接收到的 MPDU 中 CRC 域的值。若两个值不匹配，则传输中必然后若干位发生了改变。

MSDU 前面的域被称为 MAC 头部，MSDU 之后的域称为 MAC 尾部。MAC 头部和 MAC 尾部包含控制信息，这些控制信息和数据域一起在 MAC 协议中发挥作用。

逻辑链路控制　在大多数数据链路控制协议中，数据链路协议实体不仅负责使用 CRC 检错还要通过对失效帧的重发来纠正传输中的错误。在局域网协议框架中，这两个功能被分解到 MAC 层和 LLC 层。MAC 层负责检错并将出错的数据帧丢弃。LLC 层跟踪任意接收成功的帧并重发失效的帧。

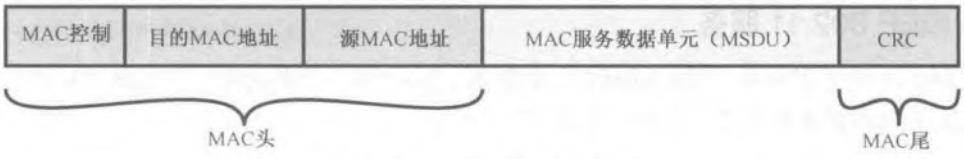


图 18.4　通用 IEEE 802 MPDU 格式

18.3.3　IEEE 802.11 网络构成和体系结构模型

图 18.5 描述了由 802.11 工作小组开发的模型。无线局域网最小的组成单元是基本服务集（BSS），它包含了执行相同 MAC 协议和竞争访问相同的共享无线媒体的无线基站。一个 BSS 可以是独立的也可以是通过接入点（AP）与主干分布式系统（DS）相连接。接入点的功能相当于网桥或中继点。

在 BSS 中，客户基站之间并不是直接相互通信的。假如 BSS 中的一个基站要和同一个 BSS 中的另一个基站通信，则源基站向 AP 发送 MAC 帧，然后 AP 将之转发至目的地基站。类似地，假如 BSS 中一个基站要与远程的基站通信，则从源基站中发送 MAC 帧给 AP，并通过 DS 中的 AP 中继，直到送至目的基站为止。BSS 相当于一个蜂窝。DS 可以是一个交换机、一个有线网络或者是无线网络。

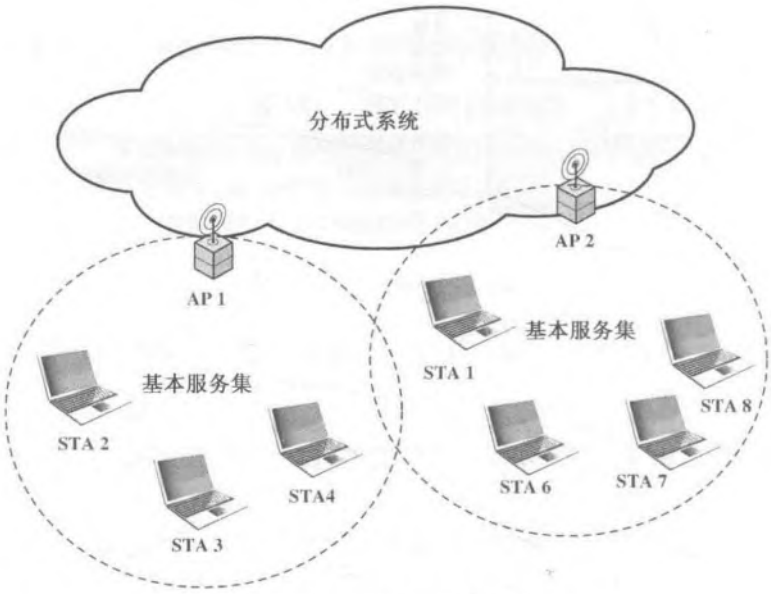


图 18.5 IEEE 802.11 扩展服务集

当 BSS 中所有的基站都是移动基站并且不使用 AP 而相互之间可以直接通信，则该 BSS 就称为独立 BSS（IBSS）。IBSS 的典型例子是 ad hoc 网络。在一个 IBSS 中，所有基站之间都直接通信，而不使用 AP。

图 18.5 只是给出了一种最简单的情形。途中每一个基站都属于一个 BSS，即每一个基站都只和同一个 BSS 中的基站享用同一无线网络空间。事实上也存在两个 BSS 在地理上交迭的情形，此时一个基站可能会参与到多个 BSS 中。此外，基站和 BSS 之间的连接也是动态的。基站可能关闭，可能进入或退出某个范围。

扩展服务集（ESS）包含两个或两个以上通过 DS 互联的基本服务集。在逻辑链路控制层，扩展服务集可以表示为一个逻辑上独立的局域网。

18.3.4 IEEE 802.11 服务

IEEE 802.11 定义了 9 种无线局域网提供的服务，这些服务用来达成与有线局域网相等的功能。表 18.2 列出了这些服务并指出了两种分类方法。

（1）服务可以分为是由基站提供的或是由 DS 提供的两种。基站服务在每一个 802.11 基站中实现，包括 AP 基站。DS 的服务是在 BSS 之间提供的，这些服务可能是在一个 AP 中实现，也可能是在 DS 中因特定目的加入的设备中实现。

（2）其中三种服务是用来控制 IEEE 802.11 局域网的访问和机密性的。另外 6 个服务是用来支持 MSDU 在基站间转发的。假如一个 MSDU 太大而不能在一个 MPDU 中传输，则可以将其分割然后在一系列 MPDU 中传输。

表 18.2　IEEE 802.11 服务

服　务	提 供 者	目　的
连接	分布式系统	MSDU 传输
认证	基站	LAN 的访问和安全
解除认证	基站	LAN 的访问和安全
关闭连接	分布式系统	MSDU 传输
分发	分布式系统	MSDU 传输
整合	分布式系统	MSDU 传输
MSDU 传输	基站	MSDU 传输
隐私	基站	LAN 的访问和安全
重新连接	分布式系统	MSDU 传输

根据 IEEE 802.11 文档，我们按序介绍这些服务以弄清 IEEE 802.11 ESS 网络的操作。作为基本的服务 MSDU 传输已经在之前提到了。与安全相关的服务参见 18.4 节。

DS 中的消息分发　涉及 DS 中的消息分发的两种服务是分发和整合。分发服务是当 MPDU 必须跨越从一个 BSS 中的基站到另一个 BSS 中的基站的 DS 时，用于基站间交互 MPDU 的最主要的服务方式。例如，假设有一个数据帧要从基站 2（STA 2）发送到基站 7（STA 7）（如图 18.5 所示）。数据帧从基站 2（STA 2）发送到在同一个 BSS 中的接入点（AP 1），然后 AP1 将数据帧交给 DS，而 DS 的功能是将帧发送到与目标 BSS 中的 STA 7 相连的接入点（AP2）上。AP2 接收到数据帧并将其转发到 STA7。消息是怎么在 DS 中传输的就不是 IEEE 802.11 标准管辖的范围了。

假如通信的两个基站在同一个 BSS 中，那么逻辑分发服务就只要通过该 BSS 中的 AP 就可以了。

整合服务使得数据能在 IEEE 802.11 局域网的基站与整合后的 IEEE 802.x 局域网中基站间传输。术语整合的（integrated）表示一个与 DS 物理连接的有线局域网，并且其基站通过整合服务可以与 IEEE 802.11 局域网在逻辑上连接。整合服务主要是进行数据交换中的地址翻译和媒体转换逻辑。

相关关联服务　MAC 层的主要功能是在 MAC 实体之间传输 MSDU，而该功能是由分发服务完成的。为了完成该功能，这需要相关关联服务提供 ESS 之内的基站的信息。在分发服务能够向基站发送数据或从基站接收数据之前，基站必须被关联。在讲述关联概念之前，我们先看一下移动性的概念。标准定义了三种基于移动性的转换类型：

- **无转换**　一个基站要么是固定的要么是移动的，但是这种移动仅仅是在一个孤立的 BSS 基站的直接通信范围内的移动。
- **BSS 转换**　定义了基站可以从一个 BSS 移动到同一 ESS 中的另一个 BSS。这种情况下，基站的数据传输要求地址功能以使其能识别基站的新位置。
- **ESS 转换**　定义了基站可以从一个 BSS 移动到不同 ESS 中的 BSS。这种情况下要求基站是能够移动的。802.11 支持的上层连接并不能得到保证。事实上，服务的中断会时常发生。

为了在 DS 中发送消息，分发服务必须知道目的基站的位置。另外，为了消息能到达目的基站，DS 还需要知道消息应该先发到哪一个 AP 上。为了满足这些要求，一个基站必须与其当前所在 BSS 中的 AP 保持连接。关于这些要求的服务有三种：

- **关联**　在基站和 AP 之间建立初始关联。在无线局域网中，一个基站传输或接收数据帧之前，

必须发布自己的标识和地址。为此，基站必须与同一 BSS 中的 AP 建立关联。然后 AP 可以将这些信息告知 ESS 中的其他 AP，以方便路由和传输地址帧。

- **重新关联** 使已建立的关联可以从一个 AP 向另一个 AP 转移，这样可以使移动基站从一个 BSS 向另一个 BSS 移动。
- **关闭关联** 从一个基站或者 AP 上发出公告表明当前的某个已存在的关联终止。一个基站必须在离开某个 ESS 或关闭之前公告该消息。当然，MAC 管理工具也能保护协议在基站未公告关闭关联消息就退出的情况下仍然正常运行。

18.4 IEEE 802.11i 无线局域网安全

在有线局域网中有两个特征与无线局域网是不同的。

- 为了能在有线局域网中传输信息，基站必须与该局域网物理连接。但是在无线局域网中，任何基站只要处在局域网中其他设备传输无线电波可达的范围之内即可。因此在有线局域网中就附带有某种程度的认证，因为它需要一些实际可见的行为来使基站与有线局域网相连。
- 相类似的是，为了接收一个从有线局域网中基站发出的消息，接收方基站也必须与该有线局域网相连。但是，在无线局域网中，任何在无线电波覆盖范围内的任何基站都可以接收。因此，有线局域网提供了一定程度的隐私性，限制数据的接收基站要与局域网相连。

无线局域网和有线局域网的差异性要求无线局域网有足够的安全服务和机制。在 802.11 规范文档中有一些关于隐私性和认证性的特征描述，但是这些特征都非常薄弱。在隐私性方面，802.11 定义了 WEP（Wired Equivalent Privacy）算法。802.11 标准中隐私性部分有致命的缺陷。在 WEP 文档之后，802.11i 工作组开发了一系列有关无线局域网（WLAN）安全问题的性能的研究。为了推进 WLAN 的强安全性的引入，Wi-Fi 联盟发布了 WPA（Wi-Fi Protected Access）并以之作为 Wi-Fi 标准。WPA 是一系列可以减少大多数 802.11 安全性问题的安全机制，并且也是基于当前的 802.11i 标准。802.11i 标准的最终形式是 RSN（Robust Security Network）。Wi-Fi 联盟在 WPA2 项目下确认生产商是否符合完整的 802.11i 标准。

RSN 规格书是十分复杂的，其内容占用了 2012 IEEE 802.11 标准的 145 页。在本节，我们仅提供一个概述。

18.4.1 IEEE 802.11i 服务

802.11i RSN 安全性规范定义了如下服务：

- **认证性** 一种协议，用来定义用户和认证服务器（AS）之间的交互，以提供相互认证，并生成用于客户端与 AP 之间通过无线连接进行通信的短期密钥。
- **访问控制**[①] 该功能增强了认证功能的使用，为消息提供路由，并为密钥交换提供支持。它能与多种认证协议协同工作。
- **带消息完整性的隐私性** MAC 层的数据（如一个 LLC PDU）与消息完整性码一起加密以保证数据不被改变。

图 18.6(a)给出了用于支持这些服务的安全协议，而图 18.6(b)列出了用于这些服务的密码学算法。

① 在此，我们将访问控制作为一个安全功能来讨论。如 18.3 节所述，它是一个与媒体访问控制（MAC）不同的功能。而在文献和标准中都使用访问控制这个术语。

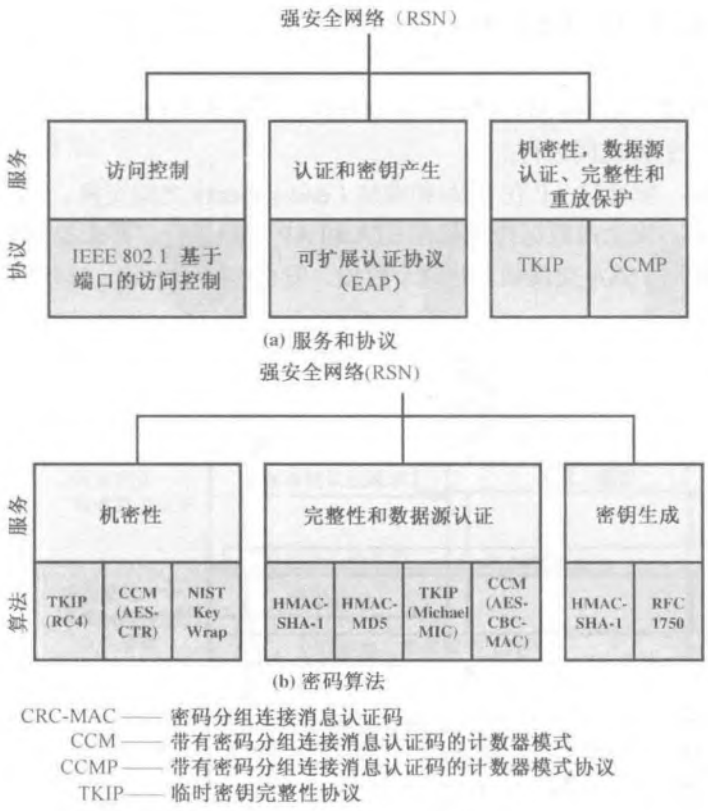


图 18.6　IEEE 802.11i 的元素

18.4.2　IEEE 802.11 操作阶段

IEEE 802.11i RSN 操作可以划分为 5 个相对独立的操作阶段。这些阶段的特性将依赖于配置和通信终端。可能性如下（详见图 18.5）：

- 在同一个 BSS 中的两个无线基站通过 BSS 中的 AP 进行通信。
- 在同一个 ad hoc IBSS 中的两个无线基站之间直接相互通信。
- 不同 BSS 中的两个无线基站通过各自的 AP 利用分布式系统进行通信。
- 一个无线基站与一个有线网络的终端基站通过 AP 和分布式系统进行通信。

IEEE 802.11i 的安全性仅关注基站（STA）和 AP 之间的安全通信。在上述列出的第 1 种情形下，安全通信通过每一个 STA 与 AP 之间建立安全通信来保证。第 2 种情形类似，AP 在功能上与 STA 在同一个域中。第 3 种情形，因为不在同一个 BSS 中，IEEE 802.11 层的分布式系统并没有提供安全性。如果需要的话，端到端的安全性必须由更高的层来提供。类似地，第 4 种情形也没有提供 STA 与 AP 之间的安全性。

考虑到这些，图 18.7 描述了一个 RSN 的 5 个操作阶段并将它们映射到对应的网络构成中。其中一个新的构件是认证服务器（AS）。矩形表示一系列 MPDU 的交换。这 5 个阶段如下定义：

- **发现**　一个 AP 使用称为信标（Beacon）和探测器响应（Probe Response）的对象来描述 IEEE 802.11i 的安全性策略。STA 用这些来表示 WLAN 中它想与之通信的 AP。STA 和 AP 建立连接时，AP 用信标和探测器响应选择加密工具和认证机制。
- **认证**　在该阶段，STA 和 AS 相互向对方证明自己的标识。认证交互成功之前，AP 会阻

塞 STA 和 AS 之间非认证的流量。除了发送 STA 和 AS 之间的通信，AP 并不参与认证交互过程。

- **密钥生成与分发** AP 和 STA 执行一系列操作，以生成加密密钥并置于 AP 和 STA。而帧交换只在 AP 和 STA 之间进行。
- **安全数据传输** 帧通过 AP 在 STA 和端站（end station）之间交换。如图 18.7 中阴影和加密模块图标所示，安全的数据传输只在 STA 和 AP 之间进行；并未提供端到端的安全性。
- **连接终止** AP 和 STA 交换帧。在这个阶段，安全连接被拆除，连接回复到初始状态。

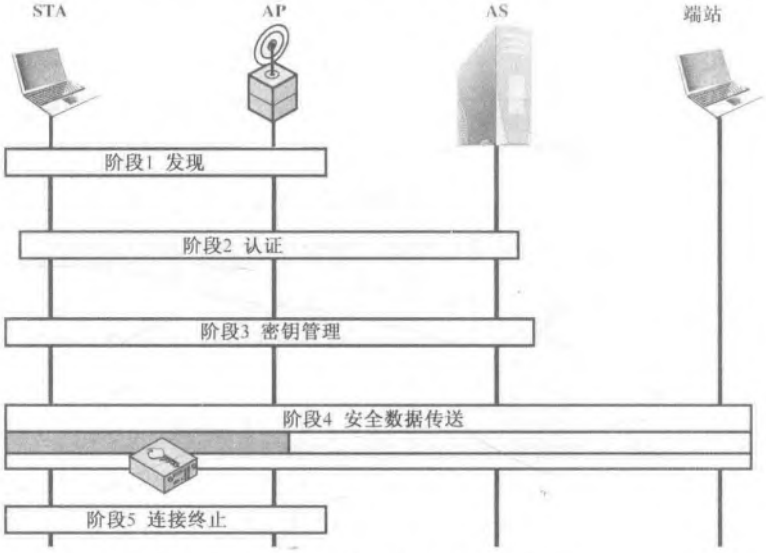


图 18.7 IEEE 802.11i 操作阶段

18.4.3 发现阶段

我们现在来看看 RSN 操作阶段的更多细节，最开始是发现阶段，在图 18.8 的上半部分有说明。这个阶段的目的是让 STA 和 AP 相互辨认，协商安全功能配置，并为将来使用这些安全功能建立关联。

安全功能 在此阶段，STA 和 AP 决定以下方面的具体技术：

- 用于保护单播流量（仅在该 STA 和 AP 之间）的机密性和 MPDU 完整性协议
- 认证方法
- 加密密钥管理方法

由于一个多播组中所有的 STA 必须使用相同的协议和加密，所以用于保护多播/广播流量的机密性和完整性协议由 AP 掌控。具体的协议和选定的密钥长度（如果是可变的）被称为密码套件（cipher suite）。机密性和完整性密码套件的选项有：

- WEP，有一个 40 位或 104 位的密钥，允许与过去的 IEEE 802.11 产品向后兼容
- TKIP
- CCMP
- 供应商的具体方法

另一个可协商套件是认证和密钥管理（AKM）套件，套件定义了：（1）AP 和 STA 进行相互认证的办法；（2）生成根密钥的办法，其他密钥可以从根密钥派生。可能的 AKM 套件是

- IEEE 802.1X
- 预共享密钥（没有进行明确的认证且只有当 STA 和 AP 共享一个独特的私有密钥时才会施行相互认证）
- 供应商的具体方法

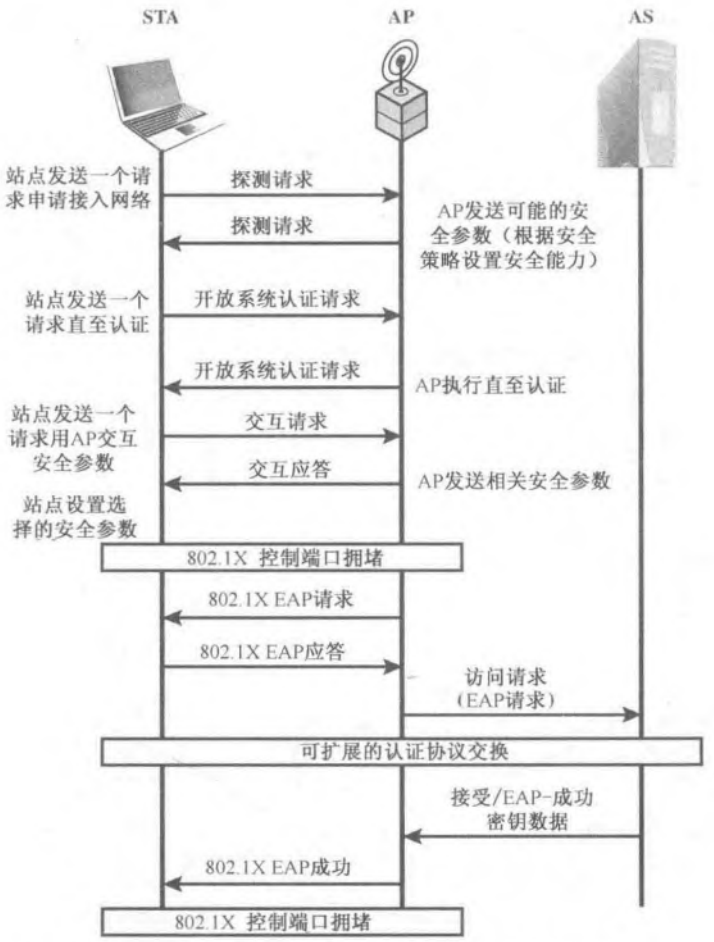


图 18.8　IEEE 802.11i 操作阶段：功能发现、认证和交互

MPDU 交换　发现阶段由三个交换组成：

- **网络和安全功能发现**　在该交换中，STA 发现与之通信的网络的存在。AP 要么定时广播它的安全功能（图中没有显示），广播按照 RSN IE 的指示，通过信标帧在一个特定的频道进行；要么通过探测响应帧对一个基站的探测请求做出回应。一个无线基站可以通过被动的监测信标帧，或者主动探测每个频道来发现可用的接入点以及相应的安全功能。
- **开放系统认证**　该帧序列的目的并不是提供安全性，而仅仅是为了与 IEEE 802.11 状态的机器保持向后兼容性。基本上，这两个设备（STA 和 AP）仅仅是交换了身份标识。
- **协商**　这一步的目的是协商一套将要使用的安全功能配置。STA 向 AP 发送一个协商请求帧。在这个帧里面，STA 从 AP 告知的能力中指定一套相匹配的能力（一个认证和密钥管理套件、一对密码套件和一个组密钥套件）。如果 AP 和 STA 之间没有匹配功能，AP 就拒绝协商请求。STA 也将其阻塞，以防止关联到一个恶意 AP 或者有人在它的频道非法插入帧。如图 18.8 所示，IEEE 802.1X 控制端口被阻塞，没有用户流量通过 AP。阻塞端口的概念随后解释。

18.4.4 认证阶段

正如前面提到的，认证阶段使得一个 STA 与分布式系统（DS）中的一个认证服务器（AS）能够相互认证。设计认证是为了只允许授权基站使用网络，并且向 STA 保证它连接的是一个合法的网络。

IEEE 802.1X 访问控制方法 IEEE 802.11i 利用了被设计用来为局域网提供访问控制功能的另一个标准，这个标准是 IEEE 802.1X，它是基于端口的网络访问控制。所使用的协议称之为扩展认证协议（EAP），在 IEEE 802.1X 标准中定义。IEEE 802.1X 使用了术语：请求者，认证者和认证服务器（AS）。在 802.11 无线局域网的上下文中，前两个术语对应无线基站和 AP。AS 通常是网络中有线连接的独立设备（如通过 DS 可访问的），但是也可以直接处于认证者一边。

在一个请求者被 AS 使用一个认证协议认证前，认证者只在请求者和 AS 之间传递控制和认证消息；802.1X 控制频道是非阻塞的，但是 802.11 数据频道是阻塞的并受预定义请求者对网络的访问控制的约束，一旦请求者被认证且授予其密钥，认证者可以发送来自请求者的数据，处于这些情况下，数据频道是非阻塞的。

如图 16.5 所示，802.1X 使用控制和非控制的概念。端口在认证者定义中是逻辑实体且指向物理网络连接。对于一个无线局域网，认证者（AP）可能只有两个物理端口：一个连接到 DS，另一个和它的 BSS 进行无线通信。每个逻辑端口都被映射到两个物理端口的其中之一。不管请求者的认证状态如何，一个非控制的端口允许请求者和其他 AS 之间的 PDU 交换。只有当请求者的当前状态授权这样一个交换时，一个控制端口才允许请求者和局域网内其他系统之间进行 PDU 交换。IEEE 802.1X 在第 16 章有更详细的讨论。

带有上层认证协议的 802.1X 框架与一个包含许多无线基站和一个 AP 的 BSS 架构非常适应。尽管如此，对于一个 IBSS 是没有 AP 的。对于一个 IBSS，802.11i 提供了一个更加复杂的解决方法，从本质上讲，涉及 IBSS 上基站间的成对认证。

MPDU 交换 图 18.8 的下半部分显示，对于认证阶段 IEEE 802.11 控制了 MPDU 交换。我们可以认为认证阶段由以下三个阶段组成。

- **连接到 AS** STA 向它的 AP（与该 STA 相关联）发送一个请求以连接到 AS。AP 识别这个请求并给 AS 发送一个访问请求。
- **EAP 交换** 这个交换让 STA 和 AS 相互授权。正如随后要解释的，许多可选的交换是可能的。
- **安全密钥分发** 一旦认证建立，AS 产生一个主会话密钥（MSK）并发送给 STA，此密钥也被称为认证（Authentication）、授权（Authorization）和审计（Accounting）（三个合称为 AAA）密钥。如随后解释的，STA 和它的 AP 进行安全通信所需的所有加密密钥都从这个 MSK 产生。IEEE 802.11i 没有规定 MSK 安全分发的方法而是说它依赖于 EAP。不管用的是什么方法，都涉及从 AS 发出，经过 AP，到达 AS 的一个包含加密过的 MSK 的 MPDU。

EAP 交换 如上所述，在认证期间有许多可能的 EAP 交换可以使用。通常地，STA 和 AP 之间流通的消息采用局域网上的 EAP（EAPOL）协议，而 AP 和 AS 之间流通的消息则使用用户服务中的远程认证拨号（RADIUS）协议，而且对于 STA 到 AP 和 AP 到 AS 则有其他可选协议。文献[FRAN07]对使用 EAPOL 和 RADIUS 的认证交换提供了以下总结。

（1）EAP 交换从 AP 发送一个 EAP-Request/Identity 帧给 STA 开始。

（2）STA 回应一个 EAP-应答/身份帧，AP 通过非控制端口接收。接着使用 RADIUS 封装数据包作为 RADIUS 访问请求数据包，通过 EAP 传送给 RADIUS 服务器。

（3）AAA 服务器回应一个 RADIUS-Access-Challenge 包，数据包作为 EAP-请求被送往 STA。该请求有适当的认证类型且包含有关挑战信息。

（4）STA 制定一个 EAP-Response 消息并发送给 AS。该应答被 AP 转换成 Radius-Access-Request，以将挑战的应答作为数据域。步骤（3）和步骤（4）根据使用的 EAP 方法可能重复多次。对于 TLS 隧道方法，验证通常要经过 10 ~ 20 轮的尝试。

（5）AAA 服务器利用 Radius-Access-Accept 包授权访问。AP 发送一个 EAP-Success 帧（为了验证真伪，一些协议在 TLS 隧道内部要求对 EAP 成功进行确认）。控制端口被授权，用户可以开始访问网络。

从图 18.6 可以注意到 AP 控制端口对于一般用户流量仍然是阻塞的。虽然验证成功，端口保持阻塞直到临时密钥安装到 STA 和 AP 上，密钥安装发生于 4 次（4-way）握手期间。

18.4.5　密钥管理阶段

在密钥管理阶段期间，各种加密密钥被生成并分发给各个 STA。有两种类型的密钥：用于 STA 和 AP 间通信的对密钥及用于多播通信的组密钥。基于文献[FRAN07]，图 18.9 显示了两个密钥层次结构（hierarchy），同时表 18.3 定义了独立密钥。

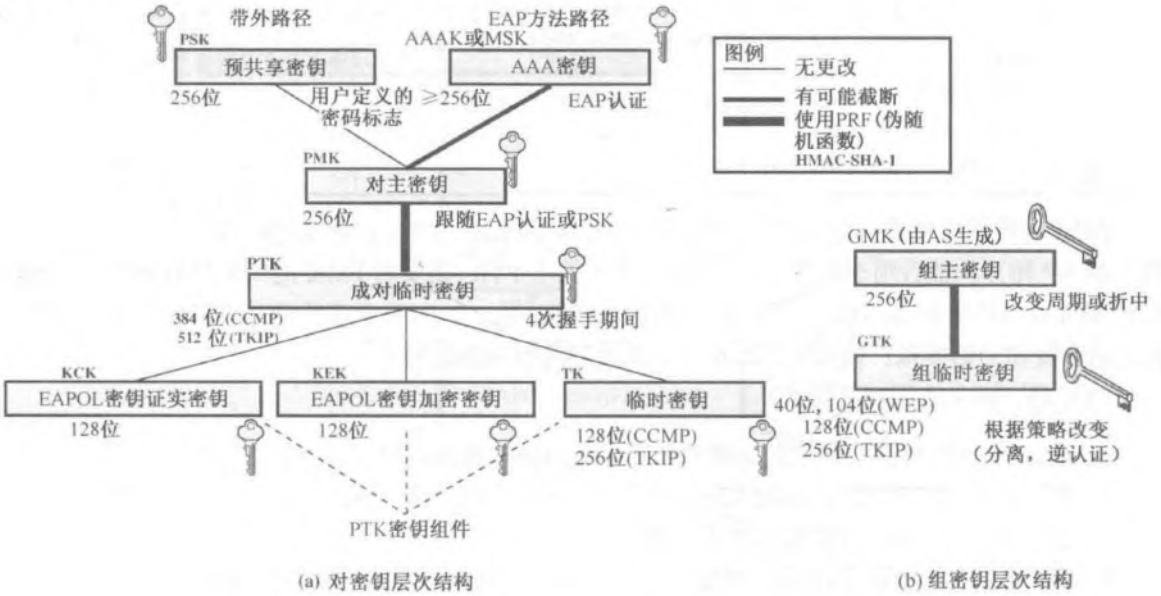


图 18.9　IEEE 802.11i 密钥层次结构

对密钥　对密钥用于一对设备之间的通信，通常是 STA 和 AP。这些密钥形成一个层次结构，开始于一个动态派生其他密钥的主密钥，并且在有限时间内使用。

层次结构的最上层有两种可能。预共享密钥（PSK）是一个被 AP 和 STA 共享的秘密信息，以及 IEEE 802.11i 之外的一些方式安装。另一个选择是主会话密钥（MSK），也被称为 AAAK，如上所述，它在认证期间使用 IEEE 802.1X 协议生成。密钥生成的实际方法取决于所使用的认证协议细节。不管是哪一种情况（PSK 或 MSK），都会有一个被 AP 及与 AP 通信的每一个 STA 共享的独一无二的密钥。由这个主密钥派生的所有其他密钥，在一个 AP 和一个 STA 之间都是独一无二的。因此，如图 18.9(a)层次结构描绘的那样，每一个 STA 在任何时候都有一套密钥，同时 AP 相对它的每个 STA 都有一套这样的密钥。

对主密钥（PMK）派生自主密钥。如果使用 PSK，那 PSK 被作为 PMK；如果使用 MSK，则

PMK 通过截断（truncation）（如果有必要）MSK 派生。在认证阶段的最后，根据 802.1X EAP 成功信息（如图 18.8 所示）所记，AP 和 STA 都有一份共享 PMK。

表 18.3 数据保密性和完整性协议的 IEEE 802.11i 密钥

缩写	名 称	描述/目的	大小（位）	类 型
AAA 密钥	认证，审计和授权密钥	用来派生 PMK 和 IEEE 802.1X 认证及密钥管理方法一起使用。和 MMSK 一样	≥256	密钥生成密钥，根密钥
PSK	预共享密钥	在预共享密钥环境成为 PMK	256	密钥生成密钥，根密钥
PMK	对主密钥	和其他输入一起使用派生 PTK	256	密钥生成密钥
GMK	组主密钥	和其他输入一起使用派生 GTK	128	密钥生成密钥
PTK	成对临时密钥	由 PMK 派生，包括 EAPOL-KCK，EAPOL-KEK 和 TK 及（对于 TKIP）MIC 密钥	512（TKIP） 384（CCMP）	组合密钥
TK	临时密钥	和 TKIP 或 CCMP 一起使用，为单播用户流量提供保密性和完整性保护	256（TKIP） 128(CCMP)	流量密钥
GTK	组临时密钥	从 GMK 派生，用来为多播/广播用户流量提供保密性和完整性保护	256（TKIP） 128(CCMP) 40,104(WEP)	流量密钥
MIC 密钥	消息完整性码密钥	TKIP 的 Michael 算法使用该密钥提供消息的完整性保护	64	消息完整性密钥
EAPOL-KCK	EAPOL-密钥确认密钥	为 4 次握手期间的密钥材料分发提供完整性保护	128	消息完整性密钥
EAPOL-KEK	EAPOL-密钥加密密钥	用来确保 GTK 和其他 4 次握手的密钥材料的保密性	128	流量密钥/密钥加密密钥
WEP 密钥	有线等价保密密钥	WEP 使用	40,104	流量密钥

PMK 被用来生成成对临时密钥（Pairwise Transient Key，PTK），PTK 实际由三个密钥组成，在 STA 和 AP 相互认证后用于两者之间的通信。为了派生 PTK，需要对 PMK 应用 HMAC-SHA-1 函数，需要 STA 和 AP 的 MAC 地址，还有必要时生成的随机数。在 PTK 生成中使用 STA 和 AP 地址以对抗会话劫持和会话模拟；使用随机数提供了额外的随机密钥材料。

PTK 的三部分密钥如下所示：

- **局域网上的 EAP（EAPOL）密钥确认密钥（EAPOL-KCK）** 支持 RSN 操作设置期间 STA 到 AP 控制帧的完整性和数据来源真实性。同时执行访问控制功能：拥有 PMK 的证明。拥有 PMK 的实体可以被授权使用链接。
- **EAPOL 密钥加密密钥（EAPOL-KEK）** 在一些 RSN 协商步骤期间保护密钥和其他数据的机密性。
- **临时密钥（TK）** 为用户流量提供实际保护。

组密钥 组密钥用于多播通信，一个 STA 可以发送 MPDU 给多个 STA。在组密钥层次结构的最上层是组主密钥（GMK）。GMK 是生成密钥的密钥，和其他输入一起使用派生组临时密钥（GTK）。不像 PTK 使用 AP 和 STA 的素材来生成，GTK 由 AP 生成并发送给其相关的 STA。究竟 GTK 是如何生成的还没有定义。不过，IEEE 802.11i 要求 GTK 的值和随机值在计算上是可区分的。利用已经建立的对密钥，GTK 被安全分发。每次有设备离开网络 GTK 都要改变。

对密钥分发 图 18.10 的上部显示了用于对密钥分发的 MPDU 交换。此交换被称为 4 次握手。STA 和 SP 使用这个握手来确认 PMK 的存在性，验证加密套件的选择，为接下来的数据会话派生一个新的 PTK。交换的 4 个部分如下所示：

- **AP→STA**　消息包含 AP 的 MAC 地址和一个随机数（Anonce）
- **STA→AP**　STA 生成自己的随机数（Snonce），使用随机数和 MAC 地址，加上 PMK，来生成一个 PTK。STA 接着发送一个包含 MAC 地址和 Snonce 的消息，使 AP 能够生成一个同样的 PTK。此消息包括使用 HMAC-MD5 或 HMAC-SHA-1-128 的消息完整性码（MIC）[①]。和 MIC 一起使用的密钥是 KCK。
- **AP→STA**　AP 现在能够生成 PTK。AP 接着发送一个消息到 STA，包含与第一个消息同样的信息，但是多了 MIC。
- **STA→AP**　这仅仅是一个确认消息，也受 MIC 保护。

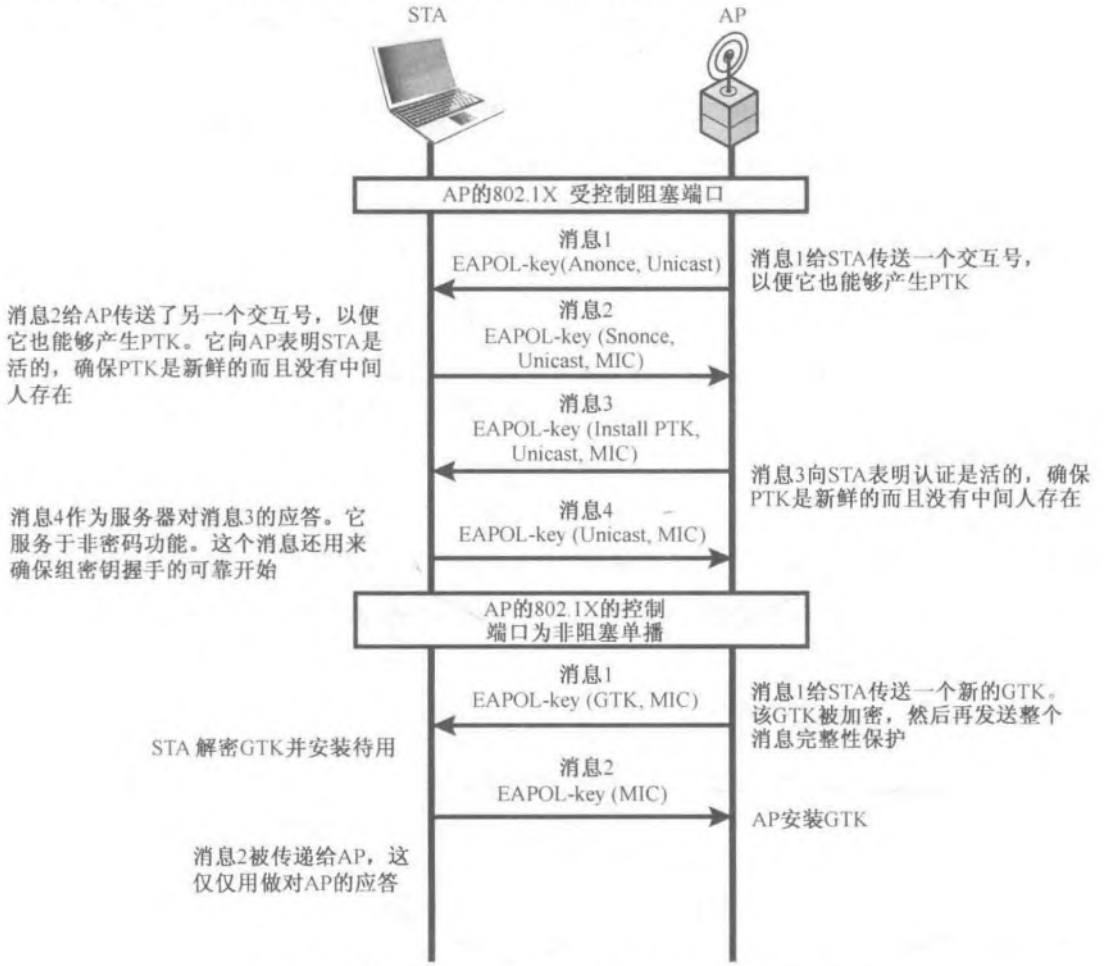


图 18.10　IEEE 802.11i 操作阶段：4 次握手和组密钥握手

组密钥分发　对于组密钥分发，AP 生成一个 GTK 并分发给多播组中的每个 STA。和每个 STA 交换的两个消息如下：

- **AP→STA**　这个消息包括 GTK，使用 RC4 或 AES 加密。用于加密的密钥是 KEK（参见第 12 章）。附加有一个 MIC 值。

① 尽管在密码学中通常用 MAC 来表示消息验证码，但是在 802.11i 中却是用 MIC，因为 MAC 在网络中有另外的标准意义，即媒体访问控制。

- **STA→AP** STA 确认对 GTK 的接收。消息包括一个 MIC 值。

18.4.6 安全数据传输阶段

IEEE 802.11i 定义了两个方案以保护 802.11MPDU 中的数据传输：临时密钥完整性协议（TKIP），和计数器模式 CBC-MAC（加密-阻塞-链接消息验证码）协议（CCMP）。

TKIP TKIP 的设计只要求对实施称为有线对等保密（WEP）的旧的无线 LAN 安全方法的设备进行软件改变。TKIP 提供两种服务：

- **消息完整性** TKIP 在 802.11MAC 帧的数据域后面加了一个消息完整性码（MIC）。MIC 由称为 Michael 的算法生成，算法使用源和目的 MAC 地址值和数据域再加上密钥材料作为输入，计算出一个 64 位值。
- **数据保密** 通过加密 MPDU 加上使用 RC4 的 MIC 值提供数据保密。

256 位 TK（参见图 18.9）使用如下。两个 64 位密钥通过使用 Michael 消息摘要算法产生一个消息完整性码。一个密钥用来保护 STA 到 AP 的消息，另一个密钥用来保护 AP 到 STA 的消息。剩下的 128 位截断生成 RC4 密钥用来加密传输数据。

对于额外保护，一个单调递增 TKIP 序列计数器（TSC）会被分配给每个帧。TSC 服务于两个目的。第一，TSC 包含于每个 MPDU，受 MIC 保护，以防止重放攻击。第二，TSC 和会话 TK 结合产生一个随每个传输 MPDU 改变的动态加密密钥，这样就使密码分析更加困难。

CCMP CCMP 适用于更新的 IEEE 802.11 设备，设备配置有支持这一方案的硬件。正如 TKIP，CCMP 提供两种服务：

- **消息完整性** CCMP 使用加密-阻塞-链接消息验证码（CBC-MAC）（在第 12 章有描述）。
- **数据保密** CCMP 使用 CTR 阻塞加密模块操作，用 AES 加密（CTR 在第 7 章有描述）。

同样的 128 位 AES 密钥用于完整性和保密性。方案使用一个 48 位的数据包序列号来构建一个随机数以防止重放攻击。

18.4.7 IEEE 802.11i 伪随机函数

在 IEEE 802.11i 方案的许多地方都会使用一个伪随机函数（PRF）。例如，用它来生成一个随机数来扩展对密钥，以及生成 GTK。最好的安全实践指出对于这些不同的目的要使用不同的伪随机数流。尽管如此，为了执行效率，我们会依靠单一的伪随机数生成函数。

PRF 是建立在使用 HMAC-SHA-1 来生成一个随机位流之上。回调那个 HMAC-SHA-1 需要一个消息（数据阻塞）和一个长度至少 160 位密钥，生成一个 160 位的 Hash 值。SHA-1 有一个特性，输入中哪怕一位的改变产生的新 Hash 值跟先前的 Hash 值也没有明显的联系。这个特性是伪随机数生成的基础。

IEEE 802.11i 中 PRF 需要 4 个输入参数，产生要求位数的随机位。函数形式 PRF(K, A, B, Len)，其中

K 为一个秘密密钥
A 为一个对应特定应用的文本字符串（如随机数生成或对密钥扩展）
B 为对应每个情况的一些数据
Len 为要求随机位的位数

例如，对于 CCMP 的成对临时密钥：

```
PTK=PRF(PMK, "Pairwise key expansion", min(AP-Addr, STA-Addr) ‖ max(AP-Addr,
    STA-Addr) ‖ min(Anonce, Snonce) ‖ max(Anonce, Snonce), 384)
```

所以，在这个例子中，参数就是

K 为 PMK
A 为文本字符串“Pairwise key expansion”
B 为两个 MAC 地址和两个随机数串联的字节序列
Len 为 384 位

类似地，一个随机数这样产生

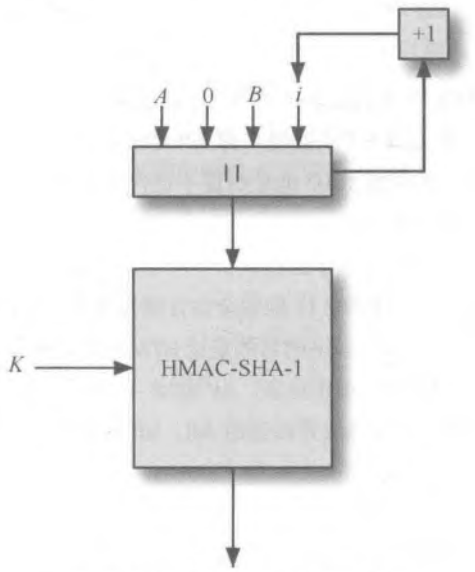
```
Nonce = PRF(随机数, "Init Counter", MAC ‖ 时间, 256)
```

这里时间是随机数产生者所知的网络时间的度量。组临时密钥这样产生

```
GTK = PRF(GMK, "Group key expansion", MAC ‖ Gnonce, 256)
```

图 18.11 说明了函数 PRF（*K*，*A*，*B*，Len）。参数 *K* 作为密钥输入 HMAC。消息由 4 个项目串联组成：参数 *A*，值为 0 的字节，参数 *B* 和一个计数器 *i*。计数器被初始化为 0。HMAC 算法执行一次，产生一个 160 位的 Hash 值。如果要求产生更多位，HMAC 使用相同的输入再运行一次，除了 *i* 每次都会增加直到产生需要数量的位。我们可以把逻辑表达为

```
PRF(K, A, B, Len)
    R ← null string
    for i ←0 to ((Len + 159)/160 - 1) do
    R ← R ‖ HMAC-SHA-1(K, A ‖ 0 ‖ B ‖ i)
    Return Truncate-to-Len (R, LEN)
```



$R = \text{HMAC-SHA-1}(K, A \| 0 \| B \| i)$

图 18.11　IEEE 802.11i 伪随机函数

18.5 关键术语、思考题和习题

关键术语

4 次握手
接入点（AP）
基本服务集（BSS）
计数器模式 CBC MAC 协议（CCMP）
分布式系统（DS）
扩展服务集（ESS）
组密钥
IEEE 802.1X
IEEE 802.11
IEEE 802.11i
独立 BSS（IBSS）
逻辑链路控制层（LLC）
介质访问控制层（MAC）
MAC 协议数据单元（MPDU）
MAC 服务数据单元（MSDU）
消息完整性码（MIC）
Michael 算法
对密钥
伪随机函数
强健安全网络（RSN）
临时密钥完整性协议(TKIP）
Wi-Fi
Wi-Fi 受保护访问（WPA）
有线等效保密协议（WEP）
无线局域网（WLAN）

思考题

18.1 什么是一个 802.11 WLAN 的基本构建块?
18.2 定义一个扩展服务集。
18.3 列出并简要定义 IEEE 802.11 服务。
18.4 分布式系统是一个无线网络吗?
18.5 互联性和流动性的观点是如何相联系的?
18.6 哪些安全区域是 IEEE 802.11i 提出的?
18.7 简要描述 IEEE 802.11i 操作的 4 个阶段。
18.8 TKIP 和 CCMP 的区别?

习题

18.1 在 IEEE 802.11 中，开放系统认证简单地由两个通信组成。一个认证是客户端的请求，包含基站 ID（通常是 MAC 地址）。随后的一个认证是来自 AP/路由器的回应，包含一个成功或失败消息。失败可能发生的一个例子是，客户端 MAC 地址明显不包含于 AP/路由器配置中。
（a）这个认证方案的好处是什么?
（b）这个认证方案的安全性弱点是什么?

18.2 在介绍 IEEE 802.11i 之前，IEEE 802.11 的安全性方案是有线对等保密（WEP）的。WEP 假设网络中的所有设备共享一个私有密钥。认证的目的是让 STA 证明它拥有该私有密钥。认证步骤如图 18.12 所示。STA 发送一个消息给 AP，请求认证。AP 发出一个挑战，这是一个以明文发送的 128 字节的随机序列。STA 用共享密钥加密挑战并回送给 AP。AP 解密接收的值并和自己发送的挑战比较。如果符合，AP 就确认认证已经成功。
（a）该认证方案的好处是什么?
（b）该认证方案是不完整的。缺了什么以及为什么它是重要的? 提示：增加一个或两个消息就能解决问题。
（c）该方案的加密弱点是什么?

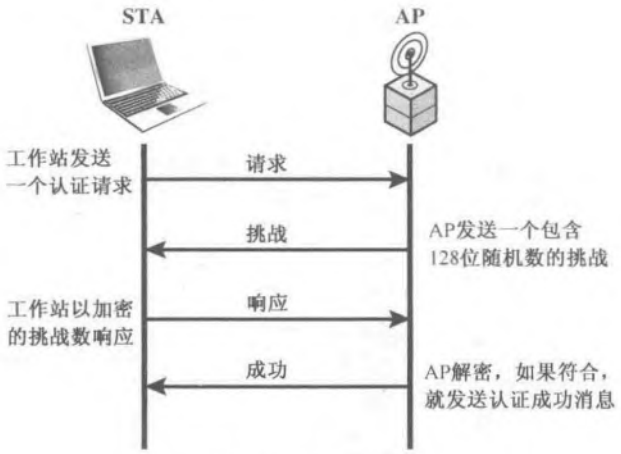


图 18.12　WEP 认证

18.3　对于 WEP，数据完整性和数据保密性使用 RC4 流加密算法实现。MPDU 发送者执行如下步骤，称为封装：

（1）发送者选择一个初始向量（IV）值。

（2）IV 值和被发送者与接收者共享的 WEP 密钥串联形成一个种子或密钥输入给 RC4。

（3）对 MAC 数据域计算一个 32 位循环冗余检验码（CRC）并附加到数据域。CRC 是常见的用于数据链路控制协议的错误检验码。在这种情况下，CRC 作为一个完整性检验值（ICV）。

（4）第（3）步的结果使用 RC4 加密形成密文块。

（5）明文 IV 被前置于加密块形成要传输的封装 MPDU。

（a）画出框图说明封装过程。

（b）描述接收端发现明文和执行完整性检验的步骤。

（c）画出框图说明（b）部分。

18.4　以 CRC 做完整性检验的潜在弱点是一个线性函数。这意味着如果消息中的单个改变，你可以预测 CRC 的哪个位会发生改变。进一步说，可以确定消息中可以被改变的位组合，以使 CRC 中的网络结果不会改变。因此，有许多明文消息的位翻转组合可以使 CRC 不改变，所以消息完整性就被攻破了。尽管如此，在 WEP 中，如果攻击者不知道加密密钥，攻击者就不能访问明文，只能访问密文块。这是否意味着 ICV 就可以不受位翻转攻击了？请解释。

第 19 章　电子邮件安全

学 习 目 标

通过本章的学习应具备以下能力：

- 总结 Internet 邮件体系结构的关键组成
- 阐述 SMTP、POP3 和 IMAP 的基本功能
- 阐述 MIME 对互联网邮件增强的意义
- 描述 MIME 的关键要素
- 理解 S/MIME（Secure/Multipurpose Internet Mail Extension）的功能和对它的安全威胁
- 理解 STARTTLS 的基本机制和在邮件安全中的作用
- 理解 DANE 的基本机制和在邮件安全中的作用
- 理解 SPF 的基本机制和在邮件安全中的作用
- 理解 DKIM 的基本机制和在邮件安全中的作用
- 理解 DMARC 的基本机制和在邮件安全中的作用

事实上，在所有的分布式环境中，电子邮件是运行最繁忙的网络应用。无论双方使用何种操作系统和通信软件，用户都希望能够给直接或间接与因特网相连的对方发送电子邮件。随着对电子邮件依附需求的爆炸式增长，认证性和保密性服务需求也在日益增长。两种被广泛应用的方法：PGP 和 S/MIME 脱颖而出。本章将阐述这两种方法，并在最后讨论 DKIM。

19.1　因特网邮件结构

为了理解本章主题，对因特网邮件结构有一个基本掌握是很有帮助的。该结构在 RFC 5598 中定义，本节将简要给出基本概念。

19.1.1　电子邮件组件

最基础的是，因特网邮件结构包含了以消息用户代理（MUA）形式的用户群，以消息处理服务（MHS）形式的传输群，其中 MHS 是由消息传输代理（MTA）组成的。MHS 从一个用户接收消息，然后将之分发给其他用户，形成一个虚拟的 MUA 到 MUA 的交换环境。该结构涉及三种类型的互用性。一个是直接处于用户之间的：消息必须由代表消息拥有者的 MUA 格式化以使目的端 MUA 可以将之向消息接收方展示。在 MUA 和 MHS 之间也有互用性要求，首先消息从 MUA 向 MHS 发出，然后经 MHS 向目的端 MUA 传输。在通过各个 MHS 的传输路径上的 MTA 成员之间也有互用性要求。

图 19.1 描述了因特网邮件结构的关键组件包含以下内容：

- **消息用户代理（MUA）**　代表用户行为和用户应用程序工作。这是他们在邮件服务中的代理。典型地，该功能是部署在用户机器上并被认为是客户端电子邮件程序或者是本地网络

电子邮件服务器。MUA 格式化消息并通过 MSA 向 MHS 初始提交。接收端 MUA 处理接收邮件用来存储或给接收方用户呈现。

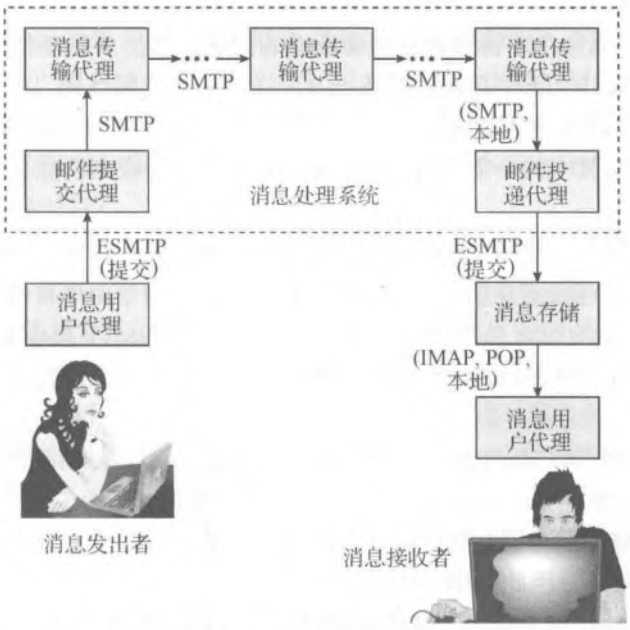


图 19.1　因特网邮件系统的功能模块和标准协议

- **邮件提交代理（MSA）**　接收由 MUA 提交的消息并增强源域的安全策略和因特网标准的要求。该功能可能部署在 MUA 或作为一个独立的功能模块。在后者情况下，简单邮件传输协议（SMTP）就运行在 MUA 与 MSA 之间。
- **消息传输代理（MTA）**　在应用层中继邮件。它就像包交换或 IP 路由一样使消息向接收方移动。由一系列的 MTA 执行中继直到消息到达目标地 MDA。MTA 还在消息头中加入路径信息。SMTP 就用在 MTA 之间和 MTA 与 MSA 或 MDA 之间。
- **邮件分发代理（MDA）**　负责将消息从 MHS 传输到 MS。
- **消息存储（MS）**　一个 MUA 可以使用一个长期的 MS。MS 可以部署在远程服务器上也可以与 MUA 部署在同一台机器上。典型地，MUA 从一个使用邮局协议（POP）或交互式邮件存取协议（IMAP）的远程服务器上提取消息。

还有两个概念需要定义。行政管理域（ADMD）是一个因特网邮件提供者。例子中包含一个执行一个本地邮件中继的部门（MTA），一个执行企业邮件中继的 IT 部门和提供公共电子邮件服务的 ISP。每一个 ADMD 会有不同的执行策略和基于可信度的决策。一个明显的例子是在组织内邮件交换和组织之间邮件交换的区别。处理这两种通信的法则是完全不同的。

域名系统（DNS）提供路径查找服务，该服务能将因特网上的主机名映射为 IP 地址，DNS 的内容将在后面讨论。

19.1.2　邮件协议

电子邮件传输用到两种类型的协议，第一种用来将信息从源端传送到目的端，即 SMTP，已历经数次扩展；第二种用来在邮件服务器之间传送信息，其中 IMAP 和 POP 使用得最为广泛。

简单邮件传输协议　SMTP 将邮件内容封装在信封中，在源端和目的端之间通过多重 MTA 来

转发。SMTP 最初于 1982 年在 RFC 821 中定义，历经数次修订后，最新的版本是 RFC 5321。修订增加了额外的指令，并引入了其他扩展，扩展 SMTP（ESMTP）多用于指后来版本的 SMTP。

SMTP 是一个基于文本的 CS 类型协议，协议中由客户端（邮件发送者）通知服务器端（下一跳接收者），通过发出一组命令来告知对方将要发送的消息，然后发送邮件消息本身。命令的大部分是由客户端发出的 ASCII 文本信息和由服务器端发送的结果返回代码。

消息从源端到目的端的传送可能发生在一个 TCP 连接上的一次 SMTP 客户端/服务器会话，也可能 SMTP 服务器仅仅是其中的一个中继节点，SMTP 客户端接收到消息后，沿着前往终点的路径向前提交给 SMTP 服务器。

SMTP 操作包括 SMTP 客户端和服务器之间交换的一系列命令和应答。首先由 SMTP 客户端发起并建立一个 TCP 连接，连接建立后，命令在此连接上传送。命令由单独的一行文本构成，前面是 4 个字母的命令代码，后面有时有参数字段。每一个命令都会在 SMTP 接收端生成一个确切的应答。大部分应答是单独的一行，而多行的情形也是允许的。应答前面是三位数字代码，后面可能会附上额外信息。

图 19.2 展示了 SMTP 客户端和服务器之间的一次通信实例。在通信开始时由客户端使用 TCP 端口 25 与服务器建立 TCP 连接（未显示在图中），使得服务器端激活 SMTP 协议并发送 220 应答。HELO 命令指明了发送域，服务器收到此命令后以 250 应答表示接受。SMTP 客户端将发送来源于用户 Smith@bar.com的邮件，因此使用 MAIL 命令标识了邮件的来源。这封邮件将发送给 foo.com 上的三位用户 Jones、Green 和 Brown。客户端使用单独的 RCPT 命令标明了每一位接收者。SMTP 服务器表示拥有 Jones 和 Brown 的信箱而没有查询到 Green 的相关信息。由于至少有一位接收者被确认，客户端准备开始发送文本消息，首先发送 DATA 命令来确保服务器准备好接收这些数据。在服务器接收完所有数据之后，发送一个 250 OK 消息告知客户端，于是客户端发送 QUIT 命令，服务器关闭这个连接。

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: HELO bar.com
S: 250 OK
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <crlf>.<crlf>
C: Blah blah blah . . .
C: . . . etc. etc. etc.
C: <crlf><crlf>
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

图 19.2 SMTP 邮件传输实例

SMTP 安全方面的一个重要扩展 STARTTLS，在 RFC 3207 中定义。STARTTLS 在 SMTP 代理

通信之间加入了保密和认证功能，使得 SMTP 代理具有防御监听者和攻击者的能力。如果客户端在建立连接时使用了支持 TLS 的端口（如端口 465 可用于 SSL 上的 SMTP 服务），服务器将提示 STARTTLS 选项可用。客户端可以在后续命令流中发送 STARTTLS 命令，由双方建立起安全 TLS 连接。使用 STARTTLS 的优点之一在于服务器无须申请额外的端口号，可以在一个端口上提供安全服务和明文操作。在 IMAP 和 POP 协议中运行 TLS 时，有着类似的机制。

一直以来，MUA/MSA 消息传送都使用 SMTP。现在优先选用的标准是 SUBMISSION，在 RFC 6409 中定义。尽管 SUBMISSION 是由 SMTP 发展而来，但 SUBMISSION 使用了独立的 TCP 端口，提出了不同的需求，如访问授权。

邮件访问协议（POP3,IMAP）　邮局协议 POP3 允许邮件客户端（用户代理 UA）从邮件服务器（MTA）上下载邮件。首先用户代理通过 TCP 连接与服务器通信（常用端口号 110），输入用户名和密码（已预先内部存储或者基于强安全策略由用户每次输入）。认证通过后，UA 可以发送 POP3 命令来取回和删除邮件。

与之相比，Internet 邮件访问协议 IMAP 也能实现邮件客户端对服务器上邮件的访问。IMAP 同样使用 TCP 连接，而使用的 TCP 端口号为 143。与 POP3 相比，IMAP 能提供更强的认证能力和其他 POP3 所无法支持的功能。

19.2　邮件格式

为了理解 S/MIME，我们首先得对其使用的电子邮件格式（即 MIME）有一个大概的理解。但是为了理解 MIME 的重要性，我们又得回到传统的电子邮件格式标准（RFC 822），而该标准仍然广泛使用。最新版本的格式规范是 RFC 5322（因特网消息格式）。因此，本节我们将首先介绍两个较早的标准然后再开始讨论 S/MIME。

19.2.1　RFC 5322

RFC 5322 定义了用电子邮件发送的文本消息的格式。这已经是基于因特网的文本邮件消息的标准并且仍然在广泛使用。在 RFC 5322 中，消息被看成是一个信封和内容的组合。信封包含了任何用来完成传输和发送功能的信息。内容则构成了发送给接收者的对象。RFC 5322 标准关注的是内容部分。但是，内容标准包含了一系列头域，这些头域是由邮件系统用来生成信封的，而该标准也致力于使程序获取头域信息更方便。

符合 RFC 5322 的消息的大体结构非常简单。一个消息包含了若干行的头部（the header）和随后的无限制的正文（the body）。头部和正文中间用一空行隔开。另外，消息是 ASCII 文本，第一个空行之上的所有行都是邮件系统中用户代理部分使用的头部行。

头部行通常包含关键字、冒号和关键字的参数，该格式允许将一个长行分割成若干短行。使用最频繁的关键字是 From，To，Subject 和 Date。例如：

```
Date: October 8, 2009 2:15:49 PM EDT
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 5322
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.
```

在 RFC 5322 中常见的另一个域是消息标识（Message-ID），该域包含一个与该消息关联的唯一标识符。

19.2.2 MIME

MIME 是 RFC 5322 的扩展，目的是解决一些因使用简单邮件传输协议 SMTP 或其他一些邮件传输协议而产生的限制和问题。RFC 2045 ~ RFC 2049 中给出了 MIME 的定义，之后又经历了数次更新文档。

之所以会使用 MIME，文献[PARZ06]列出了 SMTP/5322 方案的如下限制。

（1）SMTP 不能传输可执行文件和其他二进制对象。SMTP 邮件系统使用了很多用来将二进制文件转换为文本格式的方案，包括闻名的 UNIX UUencode/UUdecode 方案。但是，没有一个成为标准或者事实上的标准。

（2）SMTP 不能传输含有国际语言字符的文本数据，因为这些字符是由 8 位码表示的，其值可能是十进制 128 或者更大的数，而 SMTP 只能用 7 位的 ASCII 码。

（3）SMTP 服务器会拒绝超过一定尺寸的消息。

（4）ASCII 码和 EBCDIC 码之间转换的 SMTP 网关并未使用一致的映射集，因此常会出现转换错误。

（5）X.400 电子邮件网络的 SMTP 网关不能处理 X.400 消息中的非文本数据。

（6）一些 SMTP 的实现并没有严格遵守 RFC 821 文档中定义的 SMTP 标准，因此会导致一下常见的问题：

- 回车和换行的删除、添加和重排
- 截断或者是限制超过 76 个字符长度的行
- 去除白字符（制表符和空格符）
- 将消息中的行填充为等长
- 将制表符转换成多个空格符

MIME 期望能解决上述这些问题并与现存的 RFC 5322 实现兼容。

概述 MIME 规范包含以下要素：

（1）定义了 5 个新的头部域，这些域提供了消息正文的相关信息（在 RFC 5322 头中可能有定义）。

（2）定义了一些新的内容格式，标准化的表示支持了多媒体的电子邮件。

（3）定义了编码转换方式，以使邮件系统能将任何形式的内容统一的转换到一种形式。

在本小节中，我们将介绍这 5 个头部域，在随后的两节将讨论内容格式和编码转换方式。

MIME 中定义的 5 个头部域是：

- **MIME 版本（MIME-Version）** 参数的值必须为 1.0，该域表明消息遵循 RFC 2045 和 RFC 2046 的格式。
- **内容类型（Content-Type）** 详细的描述了正文中的数据以使接收方的代理能采用合适的代理或机制来为用户展示数据或者以一个合适方法来处理数据。
- **内容转换编码（Content-Transfer-Encoding）** 指示转换的类型以使其能将消息正文的展现形式可为邮件传输所用。
- **内容标识（Content-ID）** 在多重上下文中唯一标识 MIME 实体。
- **内容描述（Content-Description）** 正文对象的文本描述，这对于对象是不可读的情形是很有用的（如音频数据）。

通常在一个 RFC 5322 头中会出现一个或多个上述域。一个合适的实现必须支持 MIME-Version，Content-Type 和 Content-Transfer-Encoding 域，接收方的实现可以选择使用或忽略 Content-ID 和 Content-Description 域。

MIME 内容类型　MIME 规范文档中有很大的篇幅是关于各种内容类型的定义。这也反映了在多媒体环境下提供处理各种信息展现形式的需求。

表 19.1 列出了 RFC 2046 中规定的内容类型。主要有 7 个不同的大类和总共 15 个子类。一般而言，用内容类型表示数据的通用类型，用子类型来表示该数据类型的特定格式。

表 19.1　MIME 内容类型

类　型	子 类 型	描　述
Text	Plain	无格式文本，为 ASSCII 码或 ISO 8859 码
	Enriched	提供丰富的文档信息
Multipart	Mixed	各部分相互独立但被一起传送。接收方看到的形式与邮件消息中形式相同。
	Parallel	除了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	封装消息的正文为 RFC 822 一致
	Partial	允许按接收方透明的方式对大邮件分段
	External-body	包含一个指针，该指针是对存在于别处对象的引用
Image	jpeg	JFIF 编码的 JPEG 格式图像
	gif	图像为 GIF 格式
Video	mpeg	MPEG 格式
Audio	Basic	单通道 8 为 ISDN，8 kHz 编码
Application	PostScript	Adobe Postscript 格式
	octet-stream	一般的 8 位二进制数据

若正文为文本类型（text type），则除了要支持制定字符集外不需要特殊的软件来获取文字的全部意义，因此主要的子类型为纯文本（plain text），有简单的 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。

类型为多部分类型（multipart type）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。此分隔符不能随意出现，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。在每个部分，可以有一个通常的 MIME 头部。

以下是一个多部分消息的简单例子，包含由简单文本组成的两个部分（参见 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 4 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。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头部的 RFC 5322 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搜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 的重要特征。子类型 message/rfc822 表明该正文是一个完整的消息，包含了头和正文。除了子类型的名字外，封装的消息既可以是 RFC 5322 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，一个每段唯一的序列号 sequence number 和总的段数 total。

子类型 message/external-body 表明消息所要表达的数据并未包含在该正文中。而是在正文中包含了对该数据的访问。和其他消息类型一样，子类型 message/external-body 也有一个外头部和包含其头部的封装。外头部中唯一必须的域为内容类型（Content-Type）域，它指示了子类型 message/external-body。内头部是封装消息的头部。外头部中的内容类型（Content-Type）域必须包含一个访问类型（access-type）参数，它指示了访问方法，如 FTP（文件传输协议）。

类型 application 是指其他类型的数据，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 规范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传输方式。

MIME 标准定义了两种编码方式，但 Content-Transfer-Encoding 域可以取 6 个值，如表 19.2 所示。其中当值为 7 位，8 位和 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 位比较安全，而在其他邮件传输协议中可以使用 8 位和 binary。Content-Transfer-Encoding 域也可以取值 x-token，用来表示其他编码方式，并提供该方式的名字。这种方式可以是生产上规定的或应用程序规定的方式。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们阅读的形式而设计的。

表 19.2　MIME 转换编码

7 位	所有数据都是用短行的 ASCII 码字符表示
8 位	短行，但可以有非 ASCII 码字符（高位为 1 的 8 位）
binary	不仅可以有非 ASCII 码字符，而且可以存在 SMTP 无法传输的长行
quoted-printable	一种数据编码方式，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能被人所识别
Base64	将 6 位输入转为 8 位输出，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 转换编码使用数据由大量可打印的 ASCII 字符组成。其本质上是一种不安全的十六进制字符表示方法，并引入软回车来解决每行不得超过 76 字符的限制。

Base64 转换编码是一种基数 64 的编码方式，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，详见在线部分的附录 X。

一个多部分的例子　如图 19.3（详见 RFC 2045）描述了一个复杂的多部分消息。该消息有依次出现的 5 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。其中嵌入消息有两个需要并行播放的部分：图片和声音片段。

规范格式　规范格式是 MIME 和 S/MIME 的一个重要概念。规范格式指的是一种与内容类型相对应的格式，可以在系统中作为标准使用。规范格式与随系统而特定的本地格式相对应。RFC 2049 定义了如下这两种格式：

- **本地格式**　被传输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果标记。正文可以是 UNIX 格式的文本文件，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。本质上，这些数据是依据媒体类型按本地服务创建的。
- **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指明了所用本地格式的类型。将其转换为合适的本地格式将涉及字符集的转换、声音数据的转换、压缩或其他与媒体类型相关的操作。

```
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 ***richtext***. as defined in RFC 1896

       Isn't it cool?

     --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 19.3 MIME 消息格式举例

19.3 电子邮件威胁及综合安全

无论是对机构还是个人，电子邮件服务都应用广泛，而且容易面临各种安全威胁。一般来说，将电子邮件安全威胁分为下面几类：

- **真实性威胁**　可能导致对企业电子邮件系统的非授权访问。
- **完整性威胁**　可能导致对电子邮件内容的非授权修改。
- **机密性威胁**　可能导致敏感信息的非法泄露。
- **可用性威胁**　可能阻碍终端用户正常收发电子邮件。

对电子邮件威胁的详细说明和应对措施在 NIST SP 800-177（可信电子邮件，2015.9）中提供，如表 19.3 所示。

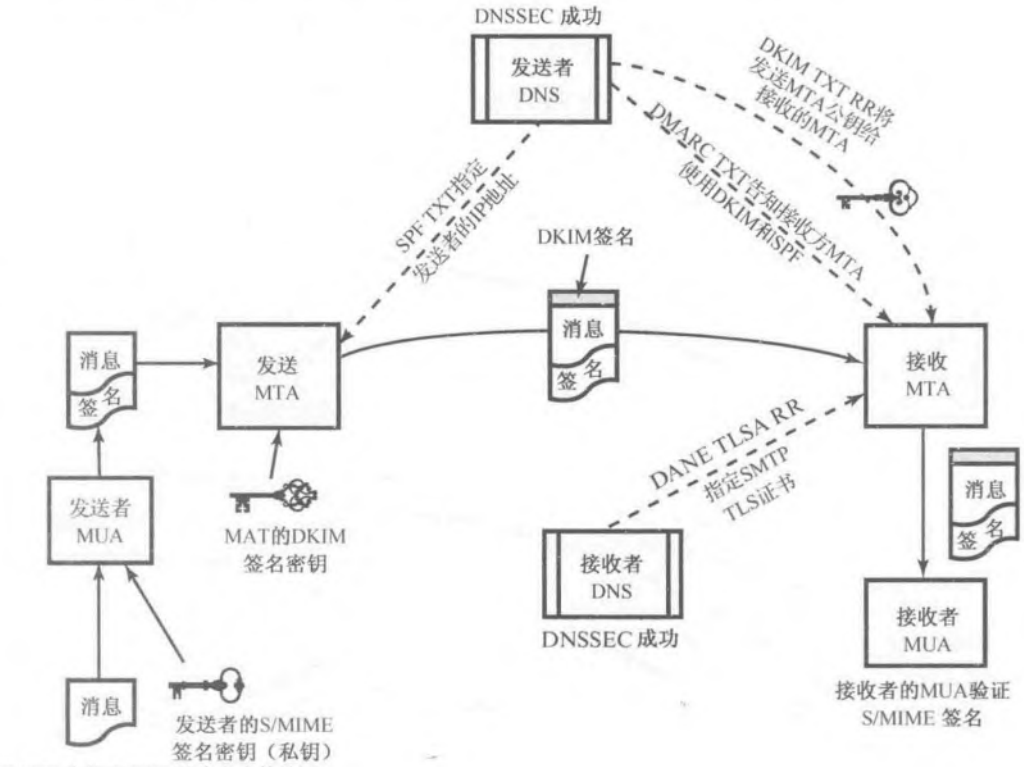
表 19.3　电子邮件威胁和应对

威　胁	对声称发送者的影响	对接收者的影响	应 对 措 施
由企业中未授权 MTA 发送的邮件（如恶意僵尸网络）	信誉受损，来自该企业的合法邮件可能因垃圾邮件/钓鱼攻击而被拦截	垃圾邮件和/或包含恶意链接的邮件可能被送达用户邮箱	部署基于域的认证技术。对邮件使用数字签名
使用欺诈的或未注册的域发送电子邮件消息	信誉受损，来自该企业的合法邮件可能因垃圾邮件/钓鱼攻击而被拦截	垃圾邮件和/或包含恶意链接的邮件可能被送达用户邮箱	部署基于域的认证技术。对邮件使用数字签名
使用伪造的发送地址或邮件地址发送电子邮件消息（如网络钓鱼，渔叉式网络钓鱼）	信誉受损，来自该企业的合法邮件可能因垃圾邮件/钓鱼攻击而被拦截	垃圾邮件和/或包含恶意链接的邮件可能被送达用户邮箱。用户可能无意间泄露敏感信息或个人身份信息	部署基于域的认证技术。对邮件使用数字签名
邮件传输途中被篡改	泄露敏感信息或个人身份信息	泄露敏感信息，替换内容可能包含恶意信息	使用 TLS 加密服务器间传输的邮件。使用端到端邮件加密技术
由监测和捕获电子邮件流量导致的敏感信息泄露	泄露敏感信息或个人身份信息	泄露敏感信息，替换内容可能包含恶意信息	使用 TLS 加密服务器间传输的邮件。使用端到端邮件加密技术
垃圾邮件（UBE）	无。除非声称的发送者被欺骗	垃圾邮件和/或包含恶意链接的邮件可能被送达用户邮箱	反垃圾邮件技术
对企业邮件服务器实施的 DoS/DDoS 攻击	无法发送邮件	无法接收邮件	使用多个邮件服务器，使用云端邮件服务提供商

SP 800-177 建议使用一系列标准化协议来应对这些威胁，包括：

- **STARTTLS**　SMTP 安全扩展协议，通过在 TLS 上运行 SMTP 协议为整个 SMTP 消息内容提供真实性、完整性、不可抵赖性（通过数字签名）和机密性服务（通过加密）。
- **S/MIME**　为 SMTP 数据中的消息字段提供真实性、完整性、不可抵赖性（通过数字签名）和机密性服务（通过加密）。
- **DNS 安全扩展（DNSSEC）**　为 DNS 数据提供真实性和完整性保护，是各类电子邮件安全协议的基础工具。
- **基于 DNS 的命名实体身份认证（DANE）**　通过为 DNSSEC 上的公钥验证提供额外的验证方式，解决证书授权中心（CA）系统中的问题。
- **发送方策略框架（SPF）**　利用域名系统，一个域名的管理者可以创建这样的 SPF 记录，记录中描述了对特定 IP 地址范围的消息发送授权。接收者可以通过检查 DNS 中的 SPF 记录轻易地确认声称的发送方是否允许使用源地址，拒绝接收来自于非授权 IP 地址的邮件。
- **域名密钥识别标准（DKIM）**　使报文传送代理可以对选择的头部和消息字段签名，通过数字签名可以验证发送者域名的合法性和消息字段的完整性。
- **基于域的消息认证、报告和一致性协议（DMARC）**　告知邮件发送者其 SPF 和 DKIM 策略的效用，告知邮件接收者在受到独立攻击和集中攻击时应采取的措施。

图 19.4 描述了在保护消息真实性和完整性时，这些组件是如何交互的。限于篇幅，图中没有展示 S/MIME 通过信息加密来保护消息机密性的部分。



DANE 为基于 DNS 的命名实体认证
DKIM 为域名密钥识别邮件标准
DMARC 为基于域名的消息认证、报告和一致性
SPF 为发送者政策框架
S/MIME 为安全多用途网际邮件扩充协议
TLSA RR 为传输层安全认证资源记录

图 19.4 保护消息真实性和完整性时 DNSSEC、SPF、DKIM、DMARC、DANE 和 S/MIME 的交互关系

19.4 S/MIME

S/MIME（Secure/Multipurpose Internet Mail Extension）是对基于 RSA 数据安全的 MIME 互联网电子邮件格式标准的安全性增强。S/MIME 的复杂性能是由一系列文档定义的，其中最重要的相关文档包括：

- **RFC 5750，S/MIME（版本 3.2）证书处理** 指定 S/MIME v3.2 中的 X.509 证书使用规则。
- **RFC 5751，S/MIME（版本 3.2）消息规范** 规范 S/MIME 消息生成及消息处理的主要定义文档。
- **RFC 4134，S/MIME 消息示例** 给出使用 S/MIME 协议的消息格式的示例。
- **RFC 2634，S/MIME 安全增强服务** 描述了为 S/MIME 协议设计的 4 个可选的安全服务扩展。
- **RFC 5652，加密消息语法（CMS）** 定义了加密消息语法（CMS）。它可以用来对任意的消息内容进行数字签名，摘要，认证或加密。

- RFC 3370，CMS 算法　描述了 CMS 中使用的几个常用的加密算法。
- RFC 5752，CMS 多重签名　描述了 CMS 中多重签名的使用。
- RFC 1847，MIME 安全多方框架-多方签名和多方加密　定义了应用于 MIME 字段的安全服务的框架。其中数字签名的使用与 S/MIME 相关，后文将提及。

19.4.1　操作描述

S/MIME 提供 4 类消息服务：认证、保密、压缩和电子邮件兼容性（如表 19.4 所示）。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这些功能。

表 19.4　S/MIME 服务概述

功　能	典型算法	典型行为
数字签名	RSA/SHA-256	用 SHA-256 生成消息的散列码，采用 SHA-256 算法，用发送者的私钥加密该消息摘要并将其加入消息中
消息加密	采用 CBC 模式的 AES-128	采用 CBC 模式下的 AES-128 算法，使用由发送者生成的一次性会话密钥加密消息。采用 RSA 算法，用接收者的公开密钥加密该会话密钥并将其加入到消息中
压缩	未指定	使用消息压缩以便存储或传输
电子邮件兼容性	基数 64 转换	为了给电子邮件应用程序提供透明性，对一个加密的消息可能会使用基数 64 转换将其转换成一个 ASCII 码字符串

认证　认证服务使用数字签名方式实现，通用的数字签名方案在第 13 章和图 13.1 中讨论过，通常会用到 RSA 算法和 SHA-256 算法。过程如下：

（1）发送方生成消息。
（2）用 SHA-256 算法生成消息的 256 位消息摘要。
（3）用发送方的私钥采用 RSA 算法加密该消息摘要并将结果添加到消息中。同时添加字段使接收方能匹配上签名者的公钥，也标识着签名者的身份。
（4）接收方用发送方的公钥按 RSA 算法解密消息并恢复消息摘要。
（5）接收方对该消息用相同的算法生成新的消息摘要，并比较该消息摘要与解密得到的散列码，如果两者吻合则该消息为可信的。

SHA-256 和 RSA 的结合提供了一个高效的数字签名方案。鉴于 RSA 的安全强度，接收者可以确信只有对应私钥的拥有者才能生成该签名。鉴于 SHA-256 的安全强度，接收者可以确信如果是不同的消息必定无法生成与之匹配的散列值。以此保证了原消息签名的有效。

尽管通常情况下签名都会附在消息或者是文件中，但也不尽然：签名也可以被分离出来。一个分离式签名可能会被同对应的消息分开存储或传输。这在一些情形下是有用的，例如某用户可能希望能为所有发送或接收到的消息的各分离式签名建立并保持一个日志。一个可执行程序的分离式签名可以用来甄别该程序是否被病毒感染。另外，分离式签名还可以用于多个成员对一份诸如合法契约等文件进行签名，每一个成员的签名都是独立的并且只用于该文件，否则签名可能会被嵌套，例如第二个成员会对文档和第一个成员生成的签名进行签名等。

保密　S/MIME 通过对消息加密来实现保密服务。其中最常使用的是 128 位密钥的 AES 算法，使用密文分组链接模式（CBC）。其密钥本身也常用 RSA 算法加密，下文将提及。

通常，涉及到加密时必须讨论密钥分发的问题。在 S/MIME 中，每一个对称密钥（指内容加密密钥）都只使用一次，这意味着对每一个消息都会随机生成一个数字用于新密钥。因为它只使用一

次，所以内容加密密钥会和消息附在一起传送，并用接收者的公开密钥加密以保护该密钥。这个流程描述如下：

（1）发送者生成消息和仅用于加密该消息的内容加密密钥，该密钥为 128 位随机自然数。
（2）使用该内容加密密钥加密消息。
（3）采用 RSA 算法，用接收者的公开密钥加密该内容加密密钥并附加在消息后。
（4）接收者采用 RSA 算法，用自己的私钥解密并恢复内容加密密钥。
（5）用内容加密密钥解密消息。

上述过程中我们会发现一些问题。首先，为了减少加密时间，对称加密和公钥加密的结合使用在效率上优于仅仅使用公钥加密直接对消息进行加密：在处理大批量信息时对称加密算法远远快于非对称加密算法。其次，公钥算法的使用解决了会话密钥分发的问题，因为只有接收者才能还原与消息绑定的会话密钥。值得注意的是，在这里我们不需要使用像在第 14 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，而是对每一个消息都使用独立的一次性密钥。此外，鉴于电子邮件存储转发的特性，使用握手协议来确保通信双方拥有相同的会话密钥也是不切实际的。最后，使用一次性对称密钥也使原本安全强度较高的对称加密方法更安全。每一个密钥都只加密少量的明文，并且密钥与密钥之间没有任何关联。因此，在某种意义上来说，只要公钥算法是安全的，那么整个方案就是安全的。

保密性和认证性 如图 19.5 所描述，保密和认证这两种服务可能会对同一个消息使用。在图中流程中，首先对消息明文生成签名，附加在消息后。然后用对称加密算法对消息明文和签名整体进行加密，其中对称加密密钥使用接收者的公钥进行加密。

S/MIME 中签名操作和消息加密操作以任意先后顺序进行均可。若首先进行签名操作，则签名者的身份信息会被加密操作所掩盖，而且这样更便于保存明文消息的签名。此外，在需要第三方验证时，假如签名在一开始就进行，这样第三方在不需要考虑对称密钥的情况下就可以进行签名的验证。

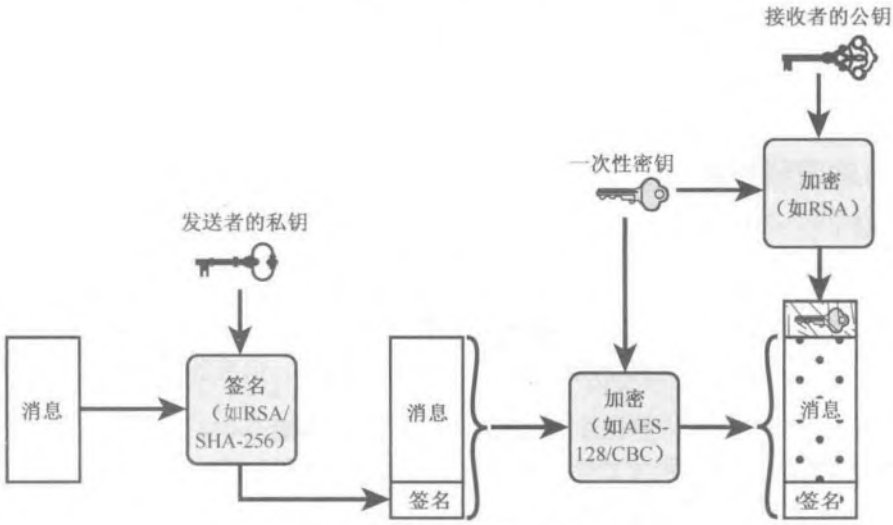
若首先进行加密操作，无需破译消息内容即可实现签名验证。由于验证签名过程无须私钥，在自动化签名验证场景下这无疑是有用的。然而，在这种情况下，接收者无法判断出签名者和原消息内容的关联性。

电子邮件兼容性 当使用 S/MIME 时，数据块中至少被传输的那部分会被加密。假如使用了数字签名服务，那么消息的摘要会用发送者的私钥加密。假如使用了保密服务，那么消息和签名（如果有签名的话）会用一次性对称密钥加密。因此，数据块的整体或部分会由任意的字节流组成。但是大多数的电子邮件系统都只允许使用由 ASCII 文本组成的数据块。为了适应该限制条件，S/MIME 提供了将原字节流转换成可打印的 ASCII 字符流的服务，即 7 位编码转换。

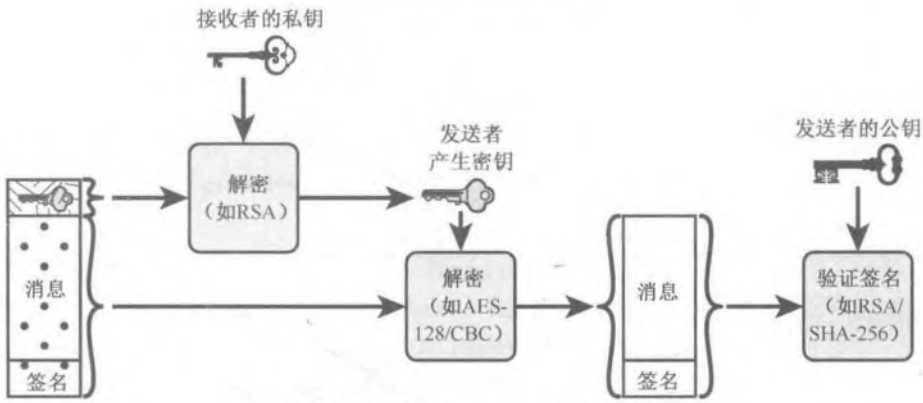
Base64 转换就是用于该目的的典型的解决方案。每 3 个字节组成的二进制数据可以映射成 4 个 ASCII 字符。详见附录 X。

Base64 算法有个值得一提的特性是它能将任意的输入流转换成 Base64 的格式，不管其内容是什么，即使输入流恰巧也是 ASCII 文本。因此，假如消息经过签名而未经加密，对整个数据块进行转换操作，那么对于一些偶然的侦听者而言输出流将是不可读的，这在一定程度上也提供了保密性。

RFC 5751 同时建议，即使外层数据没有使用 7 位编码，原始 MIME 内容也应该使用 7 位编码。原因在于这种做法允许 MIME 实体无须改动就可以被处理。例如，一个受信任的网关可能想要移除一段消息的加密而不改动签名，然后把被签名的消息发送给最终接收者使其能够直接验证签名。如果内部传输过程仍存在 8 位编码方式，如广域网上的单一邮件网关，那么除非原始 MIME 实体均为 7 位编码数据，否则签名验证是不能完成的。



(a) 发送者首先签名，然后加密消息



(b) 接收者先解密消息，然后验证签名

图 19.5　简化的 S/MIME 功能流程

压缩　S/MIME 也提供消息压缩功能，这对于电子邮件传输和文件存储的空间节省大有好处。压缩操作与签名和消息加密操作可以以任意顺序实施。RFC 5751 提出以下两条原则：

- 不建议压缩二进制编码加密数据，因为不会生成有意义的压缩结果。Base64 加密数据则非常适合被压缩。
- 如果伴随签名操作时使用有损压缩算法，则需要先进行压缩操作，再进行签名操作。

19.4.2　S/MIME 消息内容类型

根据 RFC 5652，加密消息语法中定义，S/MIME 使用以下消息内容类型：

- **数据**　指内部的 MIME 格式消息内容，之后将被封装在签名数据、封装数据或者压缩数据类型中。
- **签名数据**　用于将数字签名应用于消息。
- **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的密钥。
- **压缩数据**　用于将数据压缩应用于消息。

这些数据内容类型同时也被透明签名过程用到。透明签名，指为 MIME 编码的消息生成数字签名，消息和签名共同组成 MIME 消息。与签名数据用专用格式封装消息和签名不同，透明签名里的消息是直接可读的，签名可以被未实施 S/MIME 的电子邮件实体所验证。

19.4.3 认可的密码算法

表 19.5 总结了在 S/MIME 中使用的密码算法。S/MIME 中使用了如下术语（详见 RFC 2119 规范说明需求等级）：

- MUST 在规范说明书中表示一定要满足的需求，其实现必须与规范说明中的功能一致。
- SHOULD 如果在特定条件下有合理的理由也可以忽略，但推荐其实现包含该功能。

表 19.5 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-256 应该支持 SHA-1 接收方应该支持 MD5，以便向后兼容
利用消息摘要生成数字签名	必须支持使用 SHA-256 下的 RSA 算法 应该支持 • SHA-256 下的 DSA 算法 • SHA-256 下的 RSASSA-PSS 算法 • SHA-1 下的 RSA 算法 • SHA-1 下的 DSA 算法 • MD5 下的 RSA 算法
加密会话密钥与消息一起传输	必须支持 RSA 加密 应该支持 • RSAES-OAEP • 短暂静态模式的 Diffie-Hellman
加密消息与一次性会话密钥一起传输	必须支持 CBC 模式下的 AES-128 算法 应该支持 • AES-192 CBC 和 AES-256 CBC • 3DES CBC

S/MIME 文档规范包含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做如下两个选择：一是，发送发代理必须确定接收方代理是否能够使用该加密算法进行解密。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是可以接受的。为能达到上述要求，发送方代理可以在他发送消息之前宣布他的解密能力，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进行：

（1）如果发送方代理有一个接收方解密性能表，则他应该选择表中的第一个（即优先级最高）。

（2）如果发送方代理没有接收方代理的解密性能表，但曾经接收到一个或多个来自该接收方的消息，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）如果发送发代理没有接收方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，并且他们没有一个可以共同接受的加密算法，则发送发代理需要发送两条消息，此时，应该特别注意该消息的安全性将由于弱安全性的一份副本而受到威胁。

19.4.4　S/MIME 消息

S/MIME 使用了一系列新的 MIME 内容类型。所有的新类型都使用 PKCS（Public-Key Cryptography Specifications）设计，PKCS 是纪念为 S/MIME 做出贡献的 RSA 实验室及他们发布的一组公钥密码规范说明。

下面逐一介绍 S/MIME 消息处理过程。

保护 MIME 实体　S/MIME 用签名、加密或两者共同使用保护 MIME 实体。一个 MIME 实体可以是一个整体消息（除 RFC 5322 头部之外），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 头部）。后面我们将举例说明。

总之，将要发送的消息转换为规范形式。特别地，对给定的类型和子类型而言，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，使用安全算法会产生部分或全部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 消息中使用 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[①]标识，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（不包含 RFC 5322 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
```

① 这是一个 X.509 证书，在随后的小结中将讨论。

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的消息内容。

签名数据 smime-type 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内。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）计算带签名内容的消息摘要或散列函数。
（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，包含一个消息摘要算法标识符，被签名的消息和签名者信息块，签名数据实体可以包含一组公钥证书，该证书可以构成从上级认证机构或更高级的认证机构到该签名者的一条链路。最后将这些数据用 Base64 转换编码。例如（不包含 RFC 5322 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
```

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（Clear signing）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 能力的接收者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输过程中不改变，这意味着第一部分不能为 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 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　通常，一个应用或用户向认证机构申请公钥证书，S/MIME 实体 application/pkcs10 用于传递证书请求。证书请求包括证书的请求信息块，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题的名字（拥有待证实的公钥实体）和该用户公钥的标识位串。

仅含证书消息　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。其步骤除了没有消息内容及签名者信息块为空以外，其他过程与创建签名数据信息相同。

19.4.5　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（详见第 14 章）。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。

用户代理角色　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。每个密钥对必须是利用非确定的随机输入生成，并用安全的方式保护。用户代理应该能生成长度为 768 位到 1024 位的 RSA 密钥对，且禁止生成长度大于 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。该列表必须由用户或者代表用户的本地管理实体维护。

19.4.6　增强安全性服务

RFC 2634 中定义了 4 种 S/MIME 增强安全性服务：

- **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诉消息的发送方，已经收到消息，并通知第三方自己已经收到消息。本质上说，就是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。该标签既可用于访问控制，描述该对象能被哪些用户存取，还可描述优先级（秘密、机密和受限等）或角色（描述哪种人可以查看信息，如患者的医疗团队、医疗账单代理等）。
- **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，包括使用每一个接收方的公钥。用户可以通过使用 S/MIME 提供的邮件列表代理（Mail List Agent，MLA）来取代这一工作。邮件列表代理可以对一个输入消息为各个接收方进行相应的加密处理，而后自动发送消息。消息的发送者只需将用 MLA 的公钥加密过的消息发给 MLA 即可。

19.5 PGP

另一种电子邮件安全协议是优良保密协议 PGP，本质上与 S/MIME 具有相同的功能。PGP 最早由 Phil Zimmerman 提出，于 1991 年作为产品发布。PGP 可供大众免费使用，在个人使用中广为流行。最初的 PGP 协议作为专有技术，使用了一些知识产权限制的加密算法。1996 年，PGP 版本 5.x 在 IETF RFC 1991（PGP 消息交换格式）中问世。之后，基于 PGP 版本 5.x，OpenPGP 发展成为新的标准协议，OpenPGP 在 RFC 4880（OpenPGP 消息格式）和 RFC 3156（使用 OpenPGP 的 MIME 安全）中定义。

S/MIME 与 OpenPGP 之间主要的两个区别如下：

- **密钥认证** S/MIME 使用证书授权中心 CA（或者被 CA 授权颁发证书的本地代理）颁发的 X.509 证书。OpenPGP 中，用户生成自己的 OpenPGP 公私钥对，然后向他们已知的个人或组织征求对自己公钥的签名。如果存在指向可信根的有效 PKIX 链，则 X.509 证书是受信任的；如果一个 OpenPGP 公钥被另一个接收者信任的 OpenPGP 公钥所签名，则这个 OpenPGP 公钥是可信的。这被称为信任网。
- **密钥分发** OpenPGP 消息中不包含发送者的公钥信息，因而 OpenPGP 消息的接收者为了验证消息需要另外获得发送者的公钥。许多机构把公钥发布在受 TLS 保护的网站上：想要验证数字签名或者给机构发送加密电子邮件的人需要自行下载这些公钥并将其添加到本地 OpenPGP 客户端。密钥也可以通过 OpenPGP 公钥服务器注册，这些服务器维护了以电子邮件地址组织的 PGP 公钥信息数据库。任何人都可以把公钥发布在 OpenPGP 公钥服务器上，这个公钥可以关联任意的邮件地址。由于不存在对 OpenPGP 公钥信息的核查，用户需要根据信任网来决定是否信任一个给定的公钥。

NIST 800-177 标准推荐使用 S/MIME 而不是 PGP，因为 CA 系统在验证公钥上的表现更让人充满信心。

附录 P 给出了对 PGP 的概述。

19.6 DNSSEC

DNS 安全扩展（DNSSEC）被一些提供电子邮件安全服务的协议用到。本节首先给出域名系统 DNS 的简要概述，接着将讨论 DNSSEC。

19.6.1 域名系统

DNS 是一种目录查找服务，该服务能提供因特网主机名和其 IP 地址之间的映射。DNS 对因特网的正常运行非常重要。DNS 被邮件用户代理 MUA 和邮件传输代理 MTA 用来寻找邮件递送下一跳服务器的地址。发送端 MTA 向 DNS 查询接收者域名（@右边的部分）的邮件交换器资源记录（MX RR），以便找到接收端 MTA 并与之建立联系。

DNS 包含以下 4 个组件：

- **域名空间** DNS 使用树形结构命名空间来标识因特网上的资源。
- **DNS 数据库** 从概念上看，命名空间树形结构上的每一个节点和叶子节点都对包含在资源记录的信息集合（IP 地址，域名等）进行了命名。所有的资源记录 RR 被收集整理进入一个分布式的数据库。

- **域名服务程序**　一些保存有一部分域名树形结构和相关 RR 信息的服务器程序。
- **解析器**　从名服务程序中抽取相关信息以响应客户端请求的程序。典型的客户端请求诸如对一个给定的域名查询其对应的 IP 地址。

DNS 数据库　DNS 建立在层级数据库的基础上，数据库中包含名称、IP 地址和其他主机信息等资源记录 RR。数据库的关键特征如下：

- **可变的域名深度**　DNS 本质上允许无限层名称，以符号（.）作为名称中的分隔符。
- **分布式的数据库**　DNS 服务器下的数据库分散存在于因特网各处。
- **受数据库控制的分布**　DNS 数据库分散为成千上万个隔离的管理区间，被独立的机构管理。其分布和记录更新受数据库软件的控制。

通过这些数据库，DNS 服务器可以为需要定位特定服务器的网络应用提供域名-地址的名录服务。例如，每次发送出一份电子邮件，或者访问一个网页，就一定会发生一次 DNS 域名查找来获取邮件服务器或网站服务器的 IP 地址。

表 19.6 列出了各种类型的资源记录。

表 19.6　资源记录类型

类　型	描　述
A	主机地址。这种 RR 类型将系统名称映射到它的 IPv4 地址。一些系统（如路由器）可能有多个地址，则对应每个地址都有一个单独的 RR
AAAA	与 A 类型类似，但仅针对 IPv6 地址
CNAME	规范名称。指明一个主机的别名，并将其映射到规范（真实）名称
HINFO	主机信息。指明主机的中央处理器和操作系统
MINFO	邮箱或邮件列表信息。将一个邮箱或者邮件列表名称映射到主机名
MX	邮件交换记录。根据通往被查询域名的邮件的转发记录标识系统
NS	域的权威域名服务器
PTR	域名指针。指向域名空间的其他部分
SOA	起始授权机构记录（实现到了域名层级的那一部分）。包含了区域的相关参数
SRV	为给定的服务提供服务器名或域中能提供该服务的服务器
TXT	任意文本。提供向数据库中添加文本注释的途径
WKS	熟知服务。列出主机中可用的应用服务

DNS 操作　DNS 操作常见的包括以下步骤（参见图 19.6）：

（1）用户程序请求域名对应的 IP 地址。

（2）本地主机或本机 ISP 的解析模块向本地域名服务器查询，本地域名服务器在区域内也是作为解析器存在的。

（3）本地域名服务器检查查询的域名是否在本地数据库或缓存中，如果存在则直接返回 IP 地址。否则将向其他可用的域名服务器查询，如若需要还会向根服务器查询。

（4）当本地域名服务器收到响应后，会将查询到的域名/地址对应关系存储到本地缓存中，并根据检索到 RR 中的生存周期信息决定维护此条目的时间。

（5）用户程序收到返回的 IP 地址或错误信息。

随着因特网快速持续发展，支撑 DNS 功能的分布式 DNS 数据库必然需要频繁更新。此外，DNS 系统还需要处理 IP 地址的动态分配问题，如同 ISP 为 DSL 用户所做的一样。相应地，DNS 系统也定义有动态更新功能。条件允许时，DNS 域名服务器会自动向其他相关域名服务器发送更新信息。

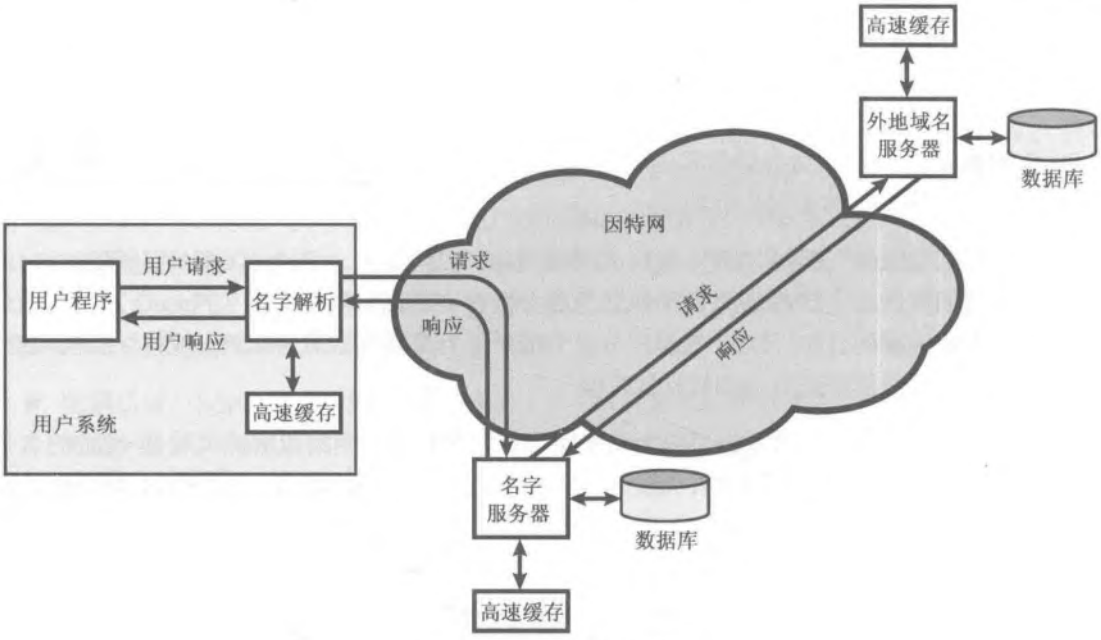


图 19.6 DNS 域名解析

19.6.2 DNS 安全扩展

DNSSEC 通过使用数字签名提供端到端的防护，数字签名由响应的区域管理机构生成，被接收者的解析软件验证。特别是，DNSSEC 避免了对直接域名服务器或解析器信任的需要，后者在源自于区域管理机构的 DNS 记录到达查询端之前会将 DNS 记录加入缓存。DNSSEC 包含一系列新的资源记录类型和对现有 DNS 协议的修改，定义在下文所述的文档中：

- **RFC 4033，DNS 安全介绍和需求** 介绍了 DNS 安全上的扩展，对其兼容性和局限性进行了讨论。文档还讨论了 DNS 安全扩展提供的服务和未提供的服务。
- **RFC 4034，DNS 安全扩展中的资源记录** 定义了 4 类为 DNS 提供安全性的新的资源记录。
- **RFC 4035，DNS 安全扩展中的协议改进** 定义了签名区间的概念，定义了使用 DNSSEC 时的服务需求和决策需求。这些技术使得一个有安全意识的询问者能够既认证 DNS 消息记录，也认证来自权威 DNS 的错误指示。

DNSSEC 操作 实质上，设计 DNSSEC 是用来保护 DNS 客户端，防止其接受伪造或是修改过的 DNS 资源记录。DNSSEC 通过使用数字签名来提供：

- **数据原始认证** 确保数据来自于正确的源端。
- **数据完整性验证** 确保 RR 的内容没有被修改过。

DNS 区域管理机构对区域类的每一个资源记录集合（RRset）进行数字签名，并在 DNS 系统中发布这些数字签名和自身的公钥。在 DNSSEC 中，对源端公钥（用于签名验证）的信任不是建立在某个第三方或一系列第三方（如公钥基础设施 PKI 链）基础之上，而是起始于一个可信区域（如根区域），由可信区域开始建立一个通往当前响应源端的信任链。在信任链上，父节点可以成功验证子节点的公钥的签名。可信区域的公钥被称为信任锚。

DNSSEC 中的资源记录 RFC 4034 定义了 4 类新的 DNS 资源记录：

- DNSKEY　包含一个公钥。
- RRSIG　一个资源记录的数字签名。
- NSEC　证实的对存在记录的否定。
- DS　授权签名者。

RRSIG 与每个 RRset 相关联，其中 RRset 是拥有相同标签，相同类型的资源记录的集合。每次客户端请求数据，都会返回一个 RRset 和 RRSIG 记录中与之相应的数字签名。客户端获得其 DNSKEY 公钥并对 RRset 的签名进行验证。

DNSSEC 依赖于导向被查询域名的 DNS 层级的真实性，因此其操作依赖于最开始对根区间的数字签名的使用。DS 资源记录为各 DNS 区间中的密钥签名和认证提供了便利，使得一条由 DNS 树根节点到特定域名的认证链，或者说对签名数据的可信序列得以构建。为保障所有 DNS 查询安全，包括对不存在域名和记录类型的查询，DNSSEC 使用 NSEC 资源记录类型来鉴别对查询的否定响应。NSEC 用来标识区间域名序列中的不存在的 DNS 名称或资源记录类型的范围。

19.7　基于 DNS 的命名实体身份认证

DANE 是一个将通常用在 TLS 中的 X.509 证书，使用 DNSSEC 与 DNS 域名相绑定的协议。该协议在 RFC 6698 中被提出，能够在无证书授权中心（CA）的情况下对 TLS 客户端和服务器实体进行认证。

使用 DANE 的基本依据，是因为在全球 PKI 系统中 CAs 的使用容易遭受攻击。每个浏览器开发商和操作系统供应商维护有一份 CA 根证书的列表作为信任锚，被称为软件根证书并存储在其根证书存储区内。PKIX 程序使得证书接收者能够根据某个证书溯源到根节点。只要根证书是值得信任的，认证过程是顺利的，客户端就可以建立连接。

然而，如果因特网上成百上千个 CA 中的任意一部分被妥协，影响就会广泛蔓延。攻击者能够获取 CA 的私钥，在非法域名下发布证书，或者在根证书存储区内引入新的伪造的根证书。妥协单一 CA 对整个 PKI 系统完整性的损害，和全球 PKI 系统的涉及范围一样，是无可估量的。此外，一些 CA 在安全实践上表现得极差。例如，一些 CA 会发布通配符证书，允许其持有者在世界各处对任意域或实体发布子证书。

DANE 的目的是用对 DNSSEC 提供安全性的依赖，取代对 CA 系统安全性的依赖。鉴于一个域名的 DNS 管理员是被授权给出关于这个区域的标识信息，那么允许管理员将域名和证书进行权威的组合绑定也是合理的，证书可供域名下的主机使用。

19.7.1　TLSA 记录

DANE 定义了新的 DNS 记录类型 TLSA，用来为认证 SSL/TLS 证书提供安全的方式。TLSA 提供：

- 指定哪个 CA 能为某一证书提供保证，或者哪个特定的 PKIX 终端实体证书是有效的。
- 指定一个服务证书或者 CA 能直接在 DNS 内被认证。

TLSA RR 使得证书发布和证书传输与给定的域紧密相关。一个服务器域名拥有者可以创建一条 TLSA 记录，记录标识着证书和其公钥信息。当客户端在 TLS 协商过程中收到一个 X.509 证书，它将按域查找 TLSA RR 记录，运行客户端证书验证程序将 TLSA 数据与证书相匹配。

图 19.7 展示了发送给请求方实体时的 TLSA RR 格式。它包括 4 个域，其中证书用法域为了适应用户对不同认证形式的需求，定义了 4 个不同的证书用法模型。这些模型描述如下：

- **PKIX-TA（CA 约束）** 在认证服务证书时指定哪个 CA 是应当信任的。这种用法模型约束了对于主机上的一个给定服务哪个 CA 有权发布证书。要求服务器证书链必须通过 PKIX 验证，验证到存储在客户端的可信根证书时验证终止。
- **PKIX-EE（服务证书约束）** 定义了服务中的何种终端实体服务证书是应当被信任的。这种用法模型约束了对于主机上的一个给定服务何种终端实体证书能被使用。要求服务器证书链必须通过 PKIX 验证，验证到存储在客户端的可信根证书时验证终止。
- **DANE-TA（信任锚断言）** 指定一个域操作 CA 作为信任锚。这种用法模型允许一个域名管理者指定一个新的信任锚——例如，当域在自己的 CA 下发布自己的证书，而此 CA 不在终端用户的信任锚集合中时。
- **DANE-EE（证书域发布）** 指定一个域操作 CA 作为信任锚。这种用法允许一个域名管理者为一个无第三方 CA 的域发布证书。这种服务器证书链是自发布的，不需要对存储在客户端的可信根证书进行验证。

前两种用法模型被设计共存使用，并增强了公共 CA 系统安全。后两种用法模型则在不使用公共 CA 的情况下也适用。

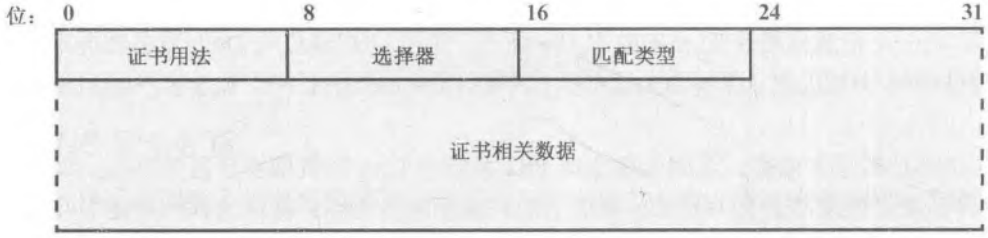


图 19.7 TLSA RR 传输格式

选择器域指示了整个证书是否匹配或者仅仅是公钥的值匹配。匹配在 TLS 协商中给出的证书和 TLSA RR 中存储的证书之间进行。匹配类型域指示了证书匹配是如何进行的。可选项有精确匹配，SHA-256 散列匹配和 SHA-512 散列匹配。证书相关数据是十六进制数据格式中的原始证书数据。

19.7.2 SMTP 中 DANE 的使用

根据 STARTTLS，DANE 能用来组合 SMTP 与 TLS，提供更全面的电子邮件传输安全。DANE 可以认证用户邮件客户端（MUA）所通信的 SMTP 提交服务器的证书，它也能够认证 SMTP 服务器（MTA）间的 TLS 连接。DANE 结合 SMTP 使用的文档在一份因特网草案（SMTP Security via Opportunistic DANE TLS，2015.5.29）中。

正如 19.1 节所述，SMTP 可以使用 STARTTLS 扩展在 TLS 上运行 SMTP，整个邮件消息和 SMTP 信封都是被加密的。这是机会性完成的，仅发生在两端都支持 STARTTLS 的情形。即使用 TLS 提供机密性保护，也会遭受以下攻击：

- 攻击者可以破坏对 TLS 性能的宣传，使连接降低级别转而不使用 TLS。
- TLS 连接有时是不可靠的（如自签名证书的使用和未匹配成功的证书都是常见的）。

DANE 能解决所有这些缺陷。一个域可以把 TLSA RR 的存在作为加密被确实执行的指示器，以此防范恶意降级。域也可以使用被 DNSSEC 签名的 TLSA RR 来认证 TLS 连接过程中使用的证书。

19.7.3　S/MIME 中 DNSSEC 的使用

DNSSEC 能用来与 S/MIME 组合提供更全面的电子邮件传输安全服务，与 DANE 作用相似。其组合使用的用法文档在因特网草案（Using Secure DNS to Associate Certificates with Domain Names for S/MIME，2015.8.27）中，草案提出了新的 SMIMEA DNS RR。SMIMEA RR 的目的是将证书与 DNS 域名相关联。

正如 19.4 节所讨论，S/MIME 消息通常包含可用来认证消息发送方的证书，此证书也可以用在加密应答时发送的消息。这个特点需要接收方 MUA 验证证书与声称发送方的联系，而 SMIMEA RR 提供了安全的验证方式。

本质上，SMIMEA RR 与 TLSA RR 的格式和内容相同，功能也相同。区别在于，SMIMEA RR 面向的是满足 MUA 处理消息字段中邮件地址指明的域名的需求，而不是处理外部 SMTP 信封中指明的域名的需求。

19.8　发送方策略框架

SPF 是一个发送方域标识和声明邮件发送者为一个给定域的标准方法。SPF 解决的问题如下：基于现有的电子邮件基础设施，任意主机可以使用任意域名作为自己邮件头部的标识符，而不仅仅局限于主机所处的区域域名。这种自由导致两个主要缺陷：

- 是减少垃圾邮件（UBE）的主要障碍，使邮件处理程序根据已知 UBE 资源过滤电子邮件变得困难。
- ADMD（参见 19.1 节）自然会关注其他实体利用其域名的便利程度，这些实体通常带着恶意企图。

SPF 定义在 RFC 7208 中，文档提供了 ADMD 授权主机在 MAIL FROM 或 HELO 身份中使用其域名的协议。遵从协议的 ADMD 在 DNS 系统中发布发送方策略框架（SPF）记录，指明了哪些主机允许使用他们的域名，遵从协议的邮件接收者在邮件事务中利用发布的 SPF 记录检查使用给定 MAIL FROM 或 HELO 身份的 MTA 的权威性。

SPF 通过检查发送方 IP 地址和发送域 SPF 记录中的策略来工作。发送域是指在 SMTP 连接中使用的域，而不是消息头部指明的域。这意味着 SPF 检查可以在消息内容被接收到之前进行。

图 19.8 给出了一个 SPF 发挥作用的例子。假设发送方 IP 地址为 192.168.0.1。来自 MTA 的消息域为 mta.example.net，发送方使用 MAIL FROM 标记，内容为 alice@example.org，表明消息源来自于 example.org 域。但是消息头部指定了 alice.sender@example.net。接收方使用 SPF 协议查询与 example.com 相关的 SPF RR，检查 IP 地址 192.168.0.1 是否是一个合法的发送方，基于 RR 检查结果进行合适的响应。

19.8.1　发送端 SPF

一个发送方域需要对给定的域名标识所有的发送者，作为独立的资源记录将信息添加到 DNS 系统中。接下来，发送方域使用 SPF 语法对每个发送者的适用策略编码。编码作为一系列机制和修饰符在 TXT DNS 资源记录中完成。机制用来定义待匹配的一个 IP 地址或地址范围，修饰符表明了对给定的匹配采用的策略。表 19.7 列出了 SPF 中最重要的机制和修饰符。

SPF 语法十分复杂，能表达复杂的发送者间的关系。更多细节详见 RFC 7208。

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: HELO mta.example.net
S: 250 OK
C: MAIL FROM:<alice@example.org>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <crlf>.<crlf>
C: To: bob@foo.com
C: From: alice.sender@example.net
C: Date: Today
C: Subject: Meeting Today
 . . .
```

图 19.8 SMTP 信封头部与消息头部不匹配的例子

表 19.7 通用 SPF 机制和修饰符

标 记	描 述
ip4	指定一个 IPv4 地址或地址段，代表一个域上的已授权发送者
ip6	指定一个 IPv6 地址或地址段，代表一个域上的已授权发送者
mx	声明邮件交换资源记录中列出的主机也是域上的合法发送者
include	列出了其他域，接收者可能对域中的发送者的 SPF RR 做进一步查询。这在拥有很多域和子域的大型组织中是有用的，这些域之间有一组共同的发送者。include 机制是递归的，即对记录进行的 SPF 检查是以整体全局进行的，并不是简单的检查的组合
all	对应还未被匹配到的所有 IP 地址

（a）SPF 机制

修 饰 符	描 述
+	给定的机制通过检查。作为默认操作不需要明确列出
−	给定的机制不允许代表该域发送电子邮件
~	给定的机制状态不明确，来自于该主机/IP 地址的邮件应当被接收并标记为仔细检查
?	SPF RR 没有对此机制给出任何陈述。这种情形下默认操作为接收邮件。（与“+”情形相同，除非导出了某种分散或汇总的邮件结论）

（b）SPF 机制修饰符

19.8.2 接收端 SPF

SPF 在接收端实施时，SPF 实体根据 SMTP 信封的 MAIL FROM 地址域和发送者的 IP 地址查询 SPF TXT RR。SPF 检查可以在电子邮件消息正文接收到之前开始，因此可能导致电子邮件内容的传送被阻止。或者，整个邮件可能被接收并缓存直到所有的检查完成。两种情形下，检查都必须在邮件消息进入终端用户邮箱之前完成。

SPF 检查涉及下列规则：

（1）如果未返回任何 SPF TXT RR，则默认操作为接收邮件消息。

（2）如果 SPF TXT RR 格式错误，则默认操作为接收邮件消息。

（3）否则用 RR 中的机制和修饰符来决定对邮件消息的处理。

图 19.9 描述了 SPF 操作。

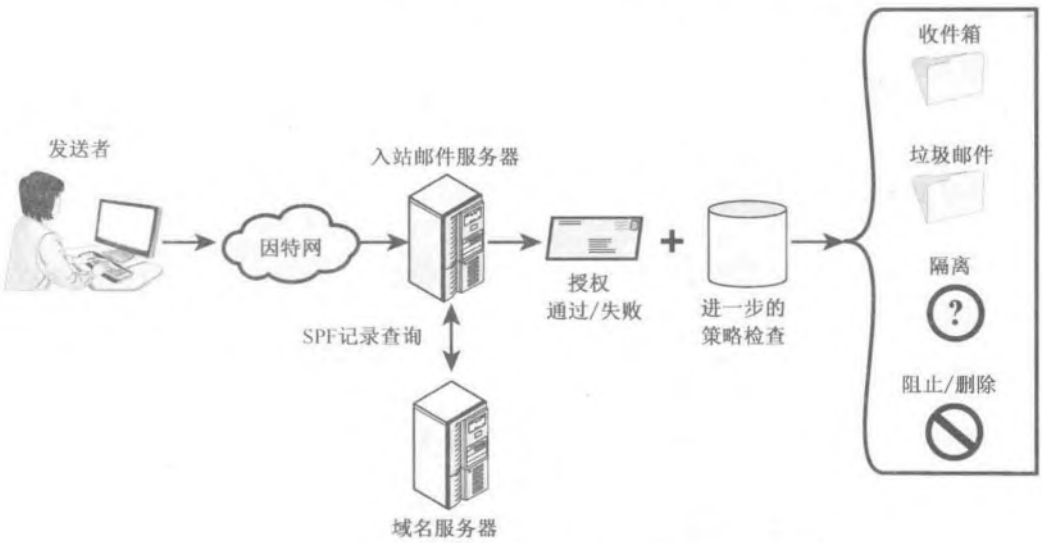


图 19.9　SPF 操作

19.9　DKIM

DKIM（DomainKeys Identified Mail）域名密钥识别邮件标准是一个用密码学方法签名电子邮件的规范，它支持一个签名域对邮件流中的某个邮件签名。消息接收方（或其代理）可以通过直接询问签名者域以获取合适公钥来验证消息是否是由拥有该签名域私钥的一方签名。DKIM 已成为因特网标准（RFC 6376：DomainKeys Identified Mail（DKIM）Signatures）。DKIM 已经被广泛采纳，组织包括公司，政府机构，gmail，Yahoo!和许多因特网服务提供商（ISP）。

19.1.1　电子邮件威胁

RFC 4686（DKIM 的威胁分析）描述了 DKIM 在特征，性能和本地潜在攻击方面带来的威胁。

特征　RFC 4686 刻画了攻击者在三个层次上的攻击范围。

（1）在较低层次上，攻击者仅仅是给那些不希望收到邮件的用户发送电子邮件，攻击者可以使用众多可用的商用工具以使发送者可以伪造消息的源地址。这使得接收者很难根据源地址或源域来过滤垃圾邮件。

（2）在中等层次上，大量垃圾邮件的专业发送者以商业组织的模式运转并代表第三方发送消息。他们使用更全面的攻击工具，包括邮件传输代理（MTA）和已注册的域名以及控制了的计算机（僵尸）网络，来发送消息并（在大多数情况下）获取更多发送地址。

（3）在最专业的层次上，攻击者技术娴熟并有充实的财力支持（如从基于电子邮件的诈骗中获取的商业利益）。他们可能使用各种上述提到的机制并可能攻击因特网基础设施本身，包括 DNS 挟持攻击和 IP 路由攻击。

性能　RFC 4686 列出了如下攻击者可能拥有的能力：

（1）在因特网上用多个地址给 MTA 和 MSA 提交消息。

（2）构建任意的消息头域，包含那些可以表示目标地址清单、中继者和其他邮件代理。

（3）在他们的控制下代表域对消息签名。

（4）生成大量貌似签名或根本未签名的消息用来进行拒绝服务（DOS）攻击。
（5）重发以前被合法域签名的消息。
（6）传输任何封装了必要信息的消息。
（7）假装成某个被控制的计算机并发送消息。
（8）操纵 IP 路由。目的是从某指定的 IP 地址或难以追踪的地址发送消息或者转发消息到某指定的目标域。
（9）使用如缓存挟持等方法影响部分域名服务器（DNS）。这样可以影响消息的路由或者伪造基于 DNS 的广告者的密钥和签名。
（10）控制大量的计算机资源，例如，通过“征召”感染蠕虫的“僵尸”计算机。这样可以让攻击者能进行各种蛮力攻击。
（11）监听一个正在进行的通信信道，如对象可能是无线网络。

定位 DKIM 主要关注那些管理单元之外的攻击者，而这些管理单元往往是声称了源发送者和接收者的。这些管理单元通常对应那些临近源发送者和接收者的受保护的网络隔离设备。这些部分正需要信任关系以使认证消息的提交不存在并且事实上也不适用于大规模的情形。相反，有了这些管理单元，就有其他更容易配置的方法（如提交认证消息），并且这些方法比 DKIM 更可用。外部的攻击者通常试图利用电子邮件的任意方到任意方的特性来使接收方 MTA 能接收来自任何地址的消息并将其发送至本地域。他们可能会生成不带签名的消息，或者是不正确的签名或者是无法追踪的域上的合法签名。他们也可能仿造邮件列表、贺卡或其他代理来为其他用户发送或转发邮件。

19.9.2 DKIM 策略

DKIM 是提供一个对终端用户透明的电子邮件认证技术的服务。基本上，用户的电子邮件都是由其所在的管理域的私钥签名的。这个签名包含消息的所有内容和 RFC 5322 消息头的一部分。在接收方终端，MDA 可以通过 DNS 访问相应的公钥并验证签名，以认证从声称拥有该消息的域认证该消息。于是，一些从某个地址发过来的消息声称是从其他域发出的就将无法通过认证测试并被拒绝接收。该方法和 S/MIME 或 PGP 都不同，后者是使用发送方的私钥签名消息的内容。DKIM 的产生源于如下因素[①]：

（1）S/MIME 要求接收方和发送方都使用 S/MIME。但是，对几乎所有的用户而言，大量的软件并不适用 S/MIME，并且大量给接收方发送的邮件也并不使用 S/MIME。
（2）S/MIME 仅仅对消息内容签名，因此，RFC 5322 头部中关于发送源的部分可能会受到安全性威胁。
（3）DKIM 并不是在客户端（MUA）实现的，因此对用户是透明的，用户不需要采取任何行动。
（4）DKIM 可应用于所有从协作域发出的邮件。
（5）DKIM 可以使合法的发送者证明其发送了某邮件并阻止仿冒者假冒合法用户。

图 19.10 是 DKIM 操作的一个简单例子。我们从一个用户生成消息并发送到 MHS 中让其发送给与该用户在同一个管理域中 MSA 的情形开始。一个电子邮件客户端生成一份电子邮件。电子邮件的提供者用自己的私钥签名了消息的内容和选择的 RFC 5322 头域。签名者是同某个域相关联的，而该域可能是一个合法的本地网络或者因特网服务提供商（ISP）或者类似于 gmail 的公共电子邮件工具。然后，签名消息就通过一系列的 MTA 向前传输。到目的地时，MDA 就索取来件签名所对应

① 在 S/MIME 中使用的因素，同样在 PGP 中也会应用到。

的公钥并在将该消息转发到目的地电子邮件客户端之前验证签名。默认的签名算法是结合了 SHA-256 的 RSA 算法，有时也用 SHA-1 与 RSA 的组合。

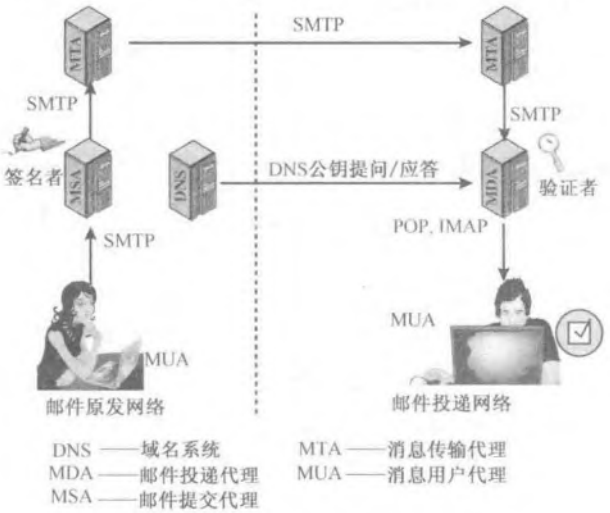


图 19.10　DKIM 部署的简单示例

19.9.3　DKIM 功能流程

图 19.11 提供了一个 DKIM 操作元素的详细示例。基本的消息处理过程是区分签名行政管理域（ADMD）和验证行政管理域。在最简单的情形下就是区分源 ADMD 和转发 ADMD，但是通常情况会涉及传输路径中的其他 ADMD。

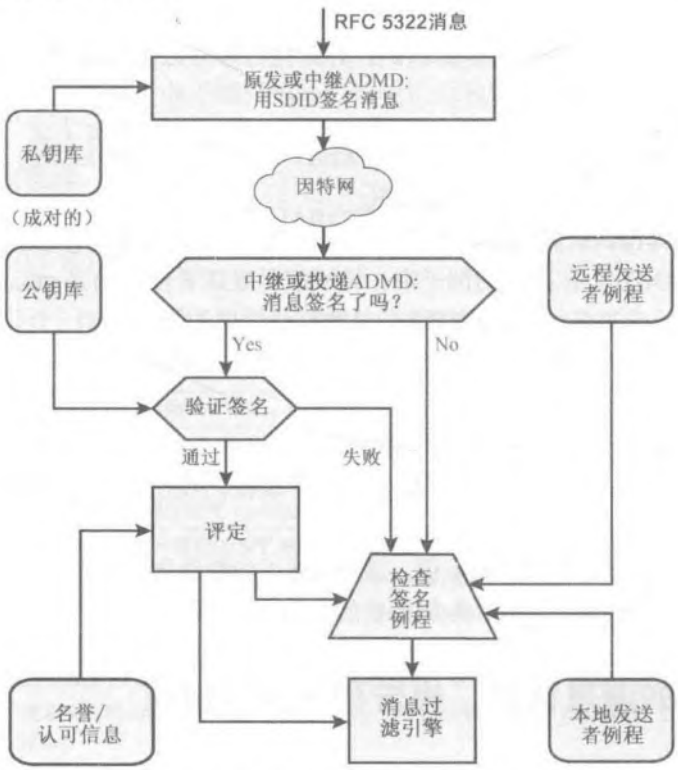


图 19.11　DKIM 功能流程

签名是由签名管理域中的认证模型使用密钥仓库中的私钥信息进行的。假如签名通过了，那么信誉信息就会用来评估签名者并且消息会通过邮件过滤系统。但若签名验证失败或没有用消息所有者的域签名，则关于签名的信息会与远程或本地的所有者联系，该信息也会通过邮件过滤系统。例如，假如发送者（如 gmail）使用了 DKIM，但是没有给出 DKIM 签名，那么邮件可能会被认为是编造的。

签名是以附加头的形式添加到 RFC 5322 消息中的，并以关键字 Dkim-Signature 开头。读者可以使用 View Long Headers（或者类似的字符）选项查看来件。如下是一个例子：

```
Dkim-Signature:      v=1; a=rsa-sha256; c=relaxed/relaxed;
                     d=gmail.com; s=gamma; h=domainkey-
                     signature:mime-version:received:date:
                     message-id:subject :from:to:content-type:
                     content-transfer-encoding;
                     bh=5mZvQDyCRuyLb1Y28K4zgS2MPOemFToDBgvbJ
                     7GO90s=;
                     b=PcUvPSDygb4ya5Dyj1rbZGp/VyRiScuaz7TTG
                     J5qW5slM+klzv6kcfYdGDHzEVJW+Z
                     FetuPfF1ETOVhELtwH0zjSccOyPkEiblOf6gILO
                     bm3DDRm3Ys1/FVrbhVOlA+/jH9Aei
                     uIIw/5iFnRbSH6qPDVv/beDQqAWQfA/wF7O5k=
```

在对消息签名之前，会对 RFC 5322 消息的头域和正文都进行标准化。标准化是用来处理消息中的微小变化，包括字符编码、消息行中的空格处理，以及头域中“可折叠”和“不可折叠”的行。标准化的目的是尽最大可能减小消息的传输（为了签名，消息本身没有改变，于是验证会重复进行一次标准化）以使接收端能最大可能地产生规范的值。DKIM 定义了两个头部标准化算法（simple 和 relaxed）和两个正文标准化算法（与前者相同）。其中 simple 算法几乎不容忍修改，而 relaxed 算法能容忍普通的修改。

签名包含了一些域。每个域都包含标识码，并后接等号且以分号结尾。这些域包含如下内容：

- **v** 为 DKIM 版本。
- **a** 为用来生成签名的算法，必须是 `rsa-sha1` 或者是 `rsa-sha256`。
- **c** 为作用在头部和正文上的标准化方法。
- **d** 为一个用于身份认证的域名，表示一个用户或一个组织的标识。在 DKIM 中，该标识称为签名域 ID（SDID）。在我们的例子中，该域表示发送者使用的是 gmail 地址。
- **s** 为由于在同一个签名域的不同环境下可能需要使用不同的密钥（允许密钥到期和分离式签名之类），DKIM 定义了一个选择器（与密钥相关），是验证方在签名验证阶段用来获取合适密钥的。
- **h** 为签名报头域。用冒号隔开的头域名称列表，这些头域名标识了签名算法的头域。在如上的例子中，签名给出了域密钥签名域。这与一个现在仍在使用的旧的算法（尽管已经被 DKIM 取代）有关。
- **bh** 为消息正文标准化部分的 Hash 值。这个提供了诊断消息签名验证失败时所需的附加信息。
- **b** 为 Base64 形式的签名数据，为加密后的散列码。

19.10 基于域的消息认证、报告和一致性协议

基于域的消息认证、报告和一致性协议（DMARC）允许邮件发送者指明以下方面的策略：他

们的邮件应当被如何处理，接收者能够返回的报告类型以及接收者返回报告的频率。协议定义在 RFC 7489 中（Domain-based Message Authentication,Reporting and Conformance，2015.3）。

DMARC 与 SPF 和 DKIM 配合使用。SPF 和 DKIM 通过 DNS 使发送者告知接收者，声称来自发送者的邮件是否是合法的，邮件是否应被传送，标记或丢弃。然而，无论 SPF 或 DKIM 都没有告知接收者是否 SPF 或 DKIM 协议处于使用状态的机制，也没有告知发送者反垃圾邮件技术使用效果的反馈机制。例如，当接收者收到没有 DKIM 签名的邮件消息，接收者并不知道到底邮件是真实可信但来自于未实施 DKIM 的发送者，还是消息是欺骗邮件。DMARC 通过标准化邮件接收者使用 SPF 和 DKIM 实施邮件认证的操作，以解决上述问题。

19.10.1　标识符匹配

DKIM，SPF 和 DMARC 分别认证一个消息的各个方面。DKIM 认证域上对消息的签名。SPF 关注于 SMTP 信封，如 RFC 5321 中定义。它既能认证 SMTP 信封上 MAIL FROM 位置出现的域，也能认证 HELO 域，或两者皆可。它们可能是不同的域，通常对终端用户不可见。

DMARC 认证的是消息头部的 From 域，如 RFC 5322 中定义。From 字段是 DMARC 机制中的核心身份，由于它是必填的消息头部字段，因此必然会在兼容消息中出现。大部分 MUA 把 RFC 5322 From 字段描述为消息的源端，并把一部分或所有的头部字段内容提交给终端用户。此字段的邮件地址是终端用户用来识别消息源头的地址，因此也是被滥用的首要目标。

DMARC 要求 From 字段地址与一个 DKIM 或 SPF 的已认证标识符相匹配。DKIM 情形下，匹配在 DKIM 签名域与 From 域间进行；SPF 情形下，匹配则在 SPF 已认证域与 From 域间进行。

19.10.2　发送端 DMARC

使用 DMARC 的电子邮件发送者必须使用 SPF 或 DKIM，或者两者均用。发送者在 DNS 系统中发布 DMARC 策略，告知接收者如何处理声称来自于发送者域的消息。策略以 DNS TXT 消息记录的形式存在。发送者还需要创建电子邮件地址以接收汇总和鉴定报告。由于这些邮件地址公开发布在 DNS TXT RR 中，易于被发现，发布者容易收到大量垃圾邮件。因此，DNS TXT RR 的发布者需要部署某种骚扰反制措施。

与 SPF 和 DKIM 类似，TXT RR 中的 DMARC 策略也是一系列以分号相隔的 `tag=value` 对编码描述的。表 19.8 描述了常用的标记。

表 19.8　DMARC 标记和取值描述

标记（名称）	描述
v=（版本）	版本字段必须作为第一个单元，默认情况下其值为 DMARC1
p=（策略）	强制填写的策略字段。取值可能为 none,quarantine 或 reject。允许渐进加强的策略，从发送方域对 DMARC 检查失败的邮件建议不采取措施（p=none），到将未通过邮件视为怀疑对象（p=quarantine），再到拒绝所有未通过邮件（p=reject）
aspf=（SPF 策略）	取值 r（默认）表示宽松的 SPF 域措施，取值 s 表示严格的 SPF 域措施。严格的一致性要求 From 地址的域间的精确匹配，SPF 检查必须精确匹配 MailFrom 地址（HELO 地址），宽松的一致性仅要求 From 和 MailFrom 地址的域对齐即可。例如，MailFrom 地址域 smtp.example.org 和 From 地址 announce@example.org 是对齐的，但是并不是严格匹配的
adkim=（DKIM 策略）	可选。取值 r（默认）表示宽松的 DKIM 域措施，取值 s 表示严格的 DKIM 域措施。严格的一致性要求消息头部的 From 域和（d=DKIM）中的 DKIM 域精确匹配，宽松情形下仅要求域部分是对齐的（同 aspf 中）

（续表）

标记（名称）	描述
fo=（报错选项）	可选。如果没有 ruf 内容则忽略。取值 0 表示如果所有的相关机制没有生成一致的通过结论，则产生一个 DMARC 失败报告。取值 1 表示如果任意一个相关机制生成了非通过结论的其他结论，则产生一个 DMARC 失败报告。其他的可选取值有 d（签名验证失败时生成一个 DKIM 出错报告），s（SPF 检查失败时生成一个 SPF 出错报告）。这些取值不是排他的，可能组合使用
ruf=	可选，但需要 fo 字段存在。列出了一系列 URI（目前仅有 mailto:<emailaddress>），指明了鉴定反馈报告应当发往何处。供特定消息失败产生的报告使用
rua=	URI 的可选列表（使用 mailto:URI），指明了发给发送方的汇总反馈报告应当发往何处。报告的发送间隔依据 ri 部分确定，若未列出则默认为 86400 秒
ri=（报告间隔）	可选，默认值为 86400 秒。此值代表发送方期望的报告间隔
pct=（百分比）	可选，默认值为 100。表明给定 DMARC 策略下发送方邮件的百分点数。使发送方能逐渐提高其策略执行的强度，防止在收到对现有策略的反馈之前执行太过严苛的策略
sp=（接收者策略）	可选，默认值为 none。其他取值与 p 标记的取值值域相同。此策略应用于所有来源于给定 DMARC RR 下已确认子域的电子邮件

当 DMARC RR 发布后，来自于发送者的消息通常按如下步骤处理：

（1）域所有者创建一条 SPF 策略并在其 DNS 数据库中发布。域所有者为其系统配置 DKIM 签名。最后，域所有者通过 DNS 发布一条 DMARC 消息处理策略。

（2）用户生成邮件消息并将消息交给域所有者的指定邮件提交服务。

（3）提交服务把相关细节传递给 DKIM 签名模块，以生成应用于消息的 DKIM 签名。

（4）提交服务把已签名消息转发给指定传输服务，递送给期望的接收方。

19.10.3 接收端 DMARC

来源于发送方的消息经其他中转节点最终到达接收者的传输服务单元，接收端通常的 DMARC 处理步骤如下：

（1）接收端执行标准验证测试，如核对 IP 黑名单，域信誉列表和执行特定来源的速率限制。

（2）接收端从消息中提取 RFC 5322 From 地址。如果地址不唯一或不合法，电子邮件将被报错拒绝。

（3）接收端根据发送域查询 DMARC DNS 记录。如果记录不存在，终止 DMARC 进程。

（4）接收端执行 DKIM 签名检查。如果消息中存在多于一个的 DKIM 签名，则必须验证一个。

（5）接收端查询发送域的 SPF 记录，执行 SPF 合法性检查。

（6）接收端执行标识符匹配操作，核对 RFC 5321 From 字段和 SPF 及 DKIM 记录（若存在）的地址结果。

（7）将上述步骤的检查结果连同邮件作者的域递交给 DMARC 模块。DMARC 模块尝试从 DNS 系统中获取此域的策略。如果未查找到，DMARC 模块决定组织结构的域，重复尝试从 DNS 系统中获取策略。

（8）如果查找到一个策略，将策略和邮件作者的域以及 SPF、DKIM 检查结果结合，得到一个 DMARC 策略结论（“通过”或“未通过”），据此有选择性地生成两种报告之一。

（9）接收端传输服务要么将邮件送达接收者的收信箱，要么基于 DMARC 结论采取其他的本地策略措施。

（10）当有需要时，接收端传输服务会从邮件传送阶段收集数据信息，供反馈使用。

图 19.12 根据一个 DMARC 实例总结了 DMARC 发送和接收功能流程。

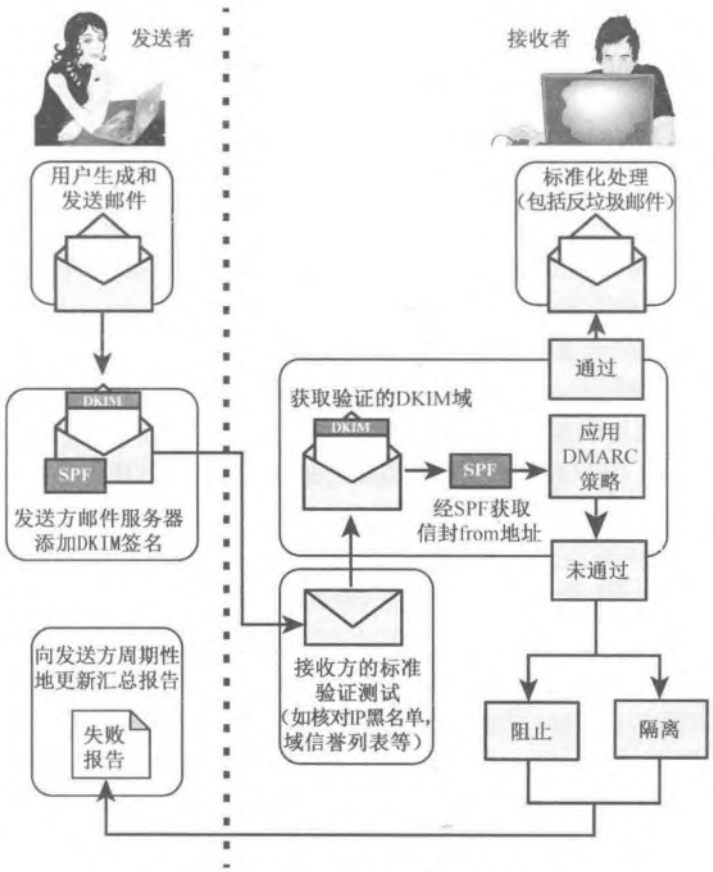


图 19.12　DMARC 功能流程

19.10.4　DMARC 报告

DMARC 报告提供关于发送方 SPF，DKIM，标识符匹配，消息处理策略等方面的反馈，使发送方能更有效率地实施这些策略。发送的报告有两种类型：汇总报告和鉴定报告。

汇总报告由接收方周期性地发送，包含成功消息认证和失败消息认证的汇总数据，包括：

- 此周期间隔的发送方 DMARC 策略。
- 接收方对消息的处置（送达，隔离，拒绝）。
- 对给定 SPF 标识符的 SPF 结果。
- 对给定 DKIM 标识符的 DKIM 结果。
- 标识符之间是否匹配。
- 按发送方子域分类的结果。
- 发送域与接收域对。
- 使用的策略，以及该策略是否与期望的策略相同。
- 消息成功认证的数量。
- 所有接收到的消息的总数。

这些信息使发送方能认识到邮件基础设施和邮件策略之间的差距。SP 800-177 标准建议发送方将 DMARC 策略设置为 p=none，这样如果一个消息未通过某些检查，对这个消息的最终处置由接收

方的本地策略决定。由于收集了 DMARC 汇总报告，发送方对外部接收方对自己邮件的认证程度有更优的定量的评估，将会把策略设置为 p=reject，表明任何未通过 SPF，DKIM 和标识符匹配检查的邮件消息都应当被拒绝。而根据自己的通信流量分析，接收者可以对发送者的 p=reject 策略是否充分可靠自行做出决策。

鉴定报告有助于发送方改良所用的 SPF 和 DKIM 机制，还能警告发送方他们的域名被卷入钓鱼网站/垃圾邮件活动。鉴定报告格式与汇总报告相似，两者的不同点如下：

- 包含足够的信息和消息头部，使发送方域能分析未通过原因。添加了与 DKIM 和 SPF DMARC 方式相适应的标识符匹配字段。
- 可选地添加了送达结果字段。
- 消息使用 DKIM 签名时，添加了 DKIM 域，DKIM 标识符，DKIM 选择器等字段。可选地添加了 DKIM 规范头部和正文字段。
- 添加了一个额外的 DMARC 认证失败类型，在认证机制之间未能产生一致的标识符时用到此类型。

19.11 关键术语、思考题和习题

关键术语

行政管理域（ADMD）	Base64 编码	加密消息语法（CMS）
分离式签名	基于 DNS 的命名实体身份认证（DANE）	DNS 安全扩展（DNSSEC）
基于域的消息认证、报告和一致性协议（DMARC）	域名系统（DNS）	域名密钥识别邮件（DKIM）
电子邮件	因特网邮件访问协议（IMAP）	邮件发送代理（MDA）
邮件提交代理（MSA）	消息处理服务（MHS）	消息存储
消息传输代理（MTA）	消息用户代理（MUA）	多用途因特网邮件扩充协议（MIME）
邮局协议版本 3（POP3）	优良保密协议（PGP）	发送方策略框架（SPF）
会话密钥	简单邮件传输协议（SMTP）	STARTTLS
SUBMISSION	S/MIME	信任

思考题

19.1 RFC 5321 与 RFC 5321 相比有什么区别？

19.2 SMTP 和 MIME 标准是什么？

19.3 MIME 内容类型与 MIME 传输编码之间有什么区别？

19.4 简要解释 Base64 编码。

19.5 为什么 Base64 转换在电子邮件应用中作用很大？

19.6 S/MIME 是什么？

19.7 S/MIME 提供的 4 个基本服务是什么？

19.8 分离式签名的作用是什么？

19.9 DKIM 是什么？

习题

19.1　对于一个 SMTP 服务器，字符序列“<CR><LF>.<CR><LF>”表示一封电子邮件的结束。如果邮件数据本身包含这串字符序列将会发生什么？

19.2　POP3 和 IMAP 分别是什么？

19.3　如果 S/MIME 中使用有损压缩算法，如 ZIP 压缩，为什么先生成签名后进行压缩比较好？

19.4　在部署 DNS 系统之前，通过在 SRI 网络信息中心维护一份简单文本文件（HOSTS.TXT）来完成对主机名称和地址的匹配。每个连接互联网的主机需要更新这份文件的本地版本以使用主机名称而不是直接复制 IP 地址。讨论与旧有的中心化 HOSTS.TXT 系统相比，DNS 系统的主要优点。

19.5　此题和接下来的几题，请参考附录 P。在图 P.2 中，公钥环的每一项包含一个 Owner Trust 域，指明了与公钥所有者相关的信任度。为什么这样做是不够的？即如果公钥所有者是受信任的并且确认是其公钥，为什么这份信任不足以允许 PGP 使用这个公钥？

19.6　在密钥层次和密钥信任方面，X.509 和 PGP 的基本区别是什么？

19.7　在 PGP 体制中，在预生成密钥产生之前，期望生成多少个会话密钥？

19.8　一个 PGP 用户可能拥有多重公钥。发送者是一个密钥 ID，由公共密钥的至少 64 位有效位组成，且随消息发出，这样接收者知道哪个公共密钥正在被发送者使用。一个拥有 N 个公钥的用户有多大几率拥有至少一个重复的密钥 ID？

19.9　PGP 签名的 128 位消息摘要中的前 16 位是以明文方式解释的。这样可以使接收者通过比较前两个字节的明文副本和解密摘要的前两个字节来确定是否使用了正确的公钥来解密消息摘要。

- 这对 Hash 算法的安全性有多大的威胁？
- 这对其设计功能有多少帮助？即帮助判断解密摘要的 RSA 密钥是否正确。

19.10　把 Base64 转换当成一种加密形式，它没有密钥。但假设对手仅知道是用来加密英文文本的某种替代算法，而没有猜到是 Base64，这种算法对抗密码分析的能力如何？

19.11　用如下技术为 plaintext 编码。假设字符存储在 8 位 ASCII 码中并使用偶校验。

（a）Base64

（b）Quoted-printable

19.12　使用一个 2×2 矩阵对 DANE 中的 4 个证书用法模型的性质进行分类。

第 20 章　IP 安全性

学 习 目 标

通过本章的学习应具备以下能力：

- 展示 IP 安全性（IPsec）的概貌
- 解释传输模式和隧道模式之间的区别
- 理解安全关联的概念
- 解释安全关联数据库和安全策略数据库之间的区别
- 总结 IPsec 在向外发数据包和向内发数据包时的流量处理功能
- 展示封装安全载荷的概貌
- 讨论混合安全关联的可选择性
- 展示因特网密钥交换协议（IKE）的概貌
- 总结 IPsec 中可选用的密码学套件

互联网社团开发了各种应用的安全机制，如电子邮件（S/MIME、PGP），客户端/服务器（Kerberos），Web 访问（SSL）等。然而，用户还有与协议层相关的安全需求。例如，企业为了确保内部 IP 网络的安全性和隐私性，它可能会不允许链接不信任站点，并对向外发出的数据包加密以及对进来的数据包认证。通过实现 IP 级安全性，企业不仅可以保护各种带有安全机制的应用，而且可以保护许多无安全机制的应用。

IP 级安全性包括三个方面的内容：认证、保密和密钥管理。认证机制不仅要确保收到的包是由包头部所标识的源端发出，还要确保该包在传输过程中未被篡改。保密性是将消息加密后传送，防止第三方窃听。密钥管理机制与密钥的安全交换有关。

本章介绍 IP 安全性（IPsec）的概要和 IPsec 的体系结构，然后再依次详细介绍其三类功能。附录 L 介绍了互联网的一些协议。

20.1　IP 安全性概述

1994 年因特网结构委员会（IAB）在报告中提出了“因特网结构中的安全性”（RFC1636）问题。该报告定义了安全机制中的关键领域。在这些领域中包含保障网络基础设施使网络通信免于在未认证情况下被监控，以及保证终端用户至终端用户之间使用认证和密码机制通信。

为了提供安全性，IAB 将认证和加密作为下一代 IP（即 IPv6）中必要的安全特性。幸运的是，这些安全特性可以同时在 IPv4 和 IPv6 中使用。这意味着生产商可以现在就提供这样的安全性能，而事实上很多厂商也已将 IPsec 的性能加入到产品中。IPsec 规范现在已经有了一系列的互联网标准。

20.1.1　IPsec 的应用

IPsec 提供了在 LAN、WAN 和互联网中安全通信的能力。其用途包括如下方面：

- **分支机构通过互联网安全互连**　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，减少了远程通信的费用。
- **与合作者建立外联网和内联网联系**　使用 IPsec 可以与其他组织进行安全通信，确保认证、保密并提供密钥交换机制。
- **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 通过在应用层附加一个安全层来保证任何由网络管理员指定的通信都是经过加密和认证的。

IPsec 能支持各种应用的关键在于它可以在 IP 层加密和（或）认证所有流量，这样就可以保护所有的分布应用，包括远程登录、客户/服务器，e-mail、文件传输、Web 访问等。图 20.1(a)是一个简化的 IPsec 隧道模式下的数据包格式。隧道模式利用了 IPsec 中一种称为封装安全有效载荷（ESP）的认证加密功能和一种密钥交换功能。对于 VPN 而言，认证和加密都不可缺少，因为这对（1）确保非授权用户无法渗透进入 VPN；（2）确保网络上的监听者不能破译 VPN 上传输的信息都至关重要。

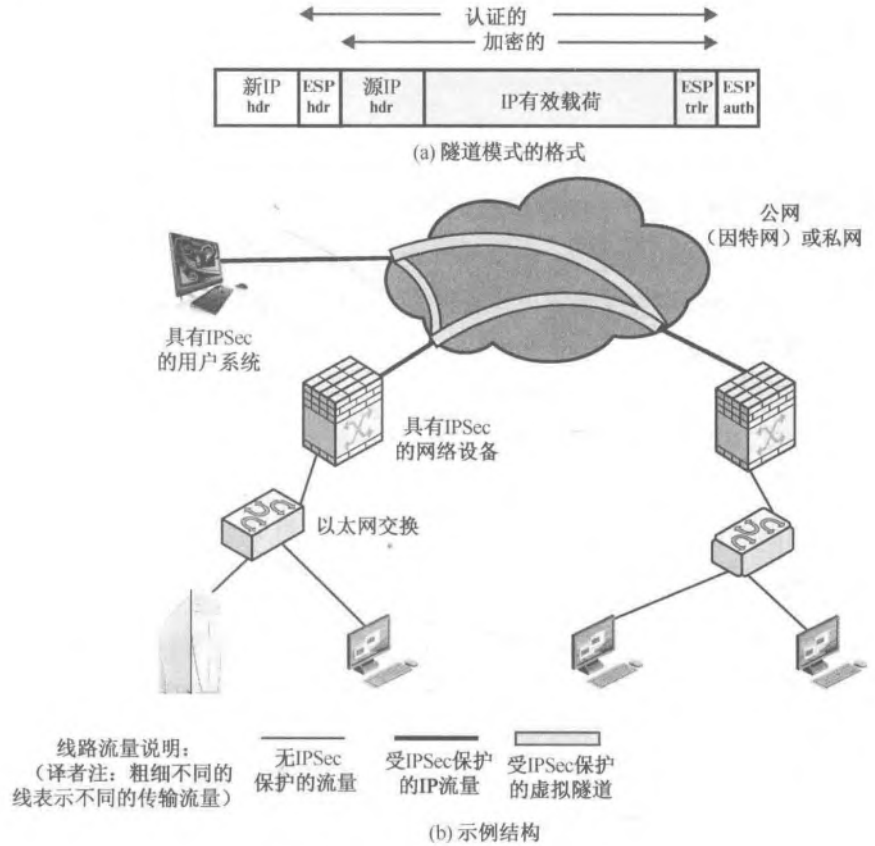


图 20.1　一种 IPSec VPN 方案

图 20.1(b)是 IPsec 的一个典型工作示意图。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，并解密来自 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进行安全传输。这些用户的工作站就必须使用 IPsec 协议提供安全保障。

20.1.2 IPsec 的优点

IPsec 的优点如下：

- 当防火墙或路由器使用 IPsec 时，它能对通过其边界的所有通信提供强安全保障。而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 能阻止所有外部流量的旁路，因为防火墙是从互联网进入组织内部的唯一通道。
- 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，即使在终端系统中使用 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，如对每个用户分配一个密钥，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。这对上网上班族非常有用，它可以在进行敏感应用程序时为用户和企业之间建立一个虚拟子网。

20.1.3 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。文献[HUIT98]列举了使用 IPsec 的例子，IPsec 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。
- 重定向报文来源于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，如开放式最短路径优先（OSPF）必须在用 IPsec 安全协议的路由器上运行。

20.1.4 IPsec 文档

IPsec 包含三个功能领域：认证、保密和密钥管理。这个 IPsec 规范是由数十个 RFC 文档和 IETF 草案文档组成的，这使得掌握所有的 IETF 规范显得非常复杂和困难。掌握 IPsec 的最好方法是查阅最新版本的 IPsec 文档目录，如 RFC 6071［IP Security（IPsec）and Internet Key Exchange（IKE）Document Roadmap, 2011.2］。所有的 IPsec 文档可以划分为如下几类：

- **体系结构** 包括 IPsec 的一般概念、安全需求、定义和机制。当前的规定是 RFC 4301，Security Architecture for the Internet Protocol。
- **认证头（AH）** AH 是一个用于提供消息认证的扩展头。当前的规范是 RFC 4302，IP Authentication Header。由于消息认证是由 ESP 提供的，所以 AH 的使用是透明的。这是在 IPsecv3 中用来保证后向兼容性的并且无法在新的应用中使用。本章将不具体介绍 AH 的内容。

- **封装安全性有效载荷（ESP）**　ESP 包含了一个封装的头和尾用来提供加密或机密和认证的结合。当前的规范是 RFC 4303，IP Encapsulating Security Payload（ESP）。
- **因特网密钥交换（IKE）**　这是描述用于 IPsec 中的密钥交换方案的文档集合。主要的规范是 RFC 7296，Internet Key Exchange（IKEv2）Protocol，而且还有大量相关的 RFC 文档。
- **密码算法**　这一类中是大量定义和描述用于加密、消息认证、伪随机函数（PRF）和密钥交换的密码算法文档。
- **其他**　是其他与 IPsec 相关的 RFC 文档，包含那些处理安全策略和管理信息库（MIB）内容。

20.1.5　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。RFC 4301 列出了如下服务：

- 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

20.1.6　传输和隧道模式

AH 和 ESP 均支持两种模式：传输模式和隧道模式，详细描述可以参阅 20.3 节，在此做简要介绍。

传输模式　传输模式主要是为上层协议提供保护的，同时增加了 IP 包[①]载荷的保护。例如，TCP 段或 UDP 段、ICMP 包均在主机协议栈的 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 可以加密和认证（可选）IP 载荷，但不包括 IP 头。传输模式的 AH 可以认证 IP 载荷和 IP 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成一个新 IP 包的载荷，并拥有一个新的外部 IP 包头。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 包头。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：一个网络中的主机 A 生成与另一个网络中主机 B 作为目的地址的一个 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向外发送的包，来决定是否需要 IPsec 的处理。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 头中封装包，外部 IP 包中的源 IP 地址为此防火墙的 IP 地址，

① 在本章中，术语 IP 包是指 IPv4 的报文或者是 IPv6 的数据包。

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 头；在 B 的防火墙处，除去外部 IP 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 包头。AH 在隧道模式中认证整个内部 IP 包和外部 IP 头中的选中部分。

表 20.1 总结了传输模式和隧道模式的功能。

表 20.1 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 头中的选中部分、IPv6 扩展头	认证整个内部 IP 包（内部包头和 IP 载荷）和部分选中的外部 IP 头、外部 IPv6 扩展头
ESP	加密 IP 载荷和跟在 ESP 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 头	加密内部 IP 包。认证内部 IP 包

20.2 IP 安全性策略

IPsec 操作的基本概念是应用于每一个从源端到目的端传输的 IP 包的安全策略。IPsec 策略主要由两个数据库的交互决定，这两个数据库是安全关联数据库（SAD）和安全策略数据库（SPD）。本节将概要介绍这两种数据库并总结其在 IPsec 操作中的使用。图 20.2 描述其相互关系。

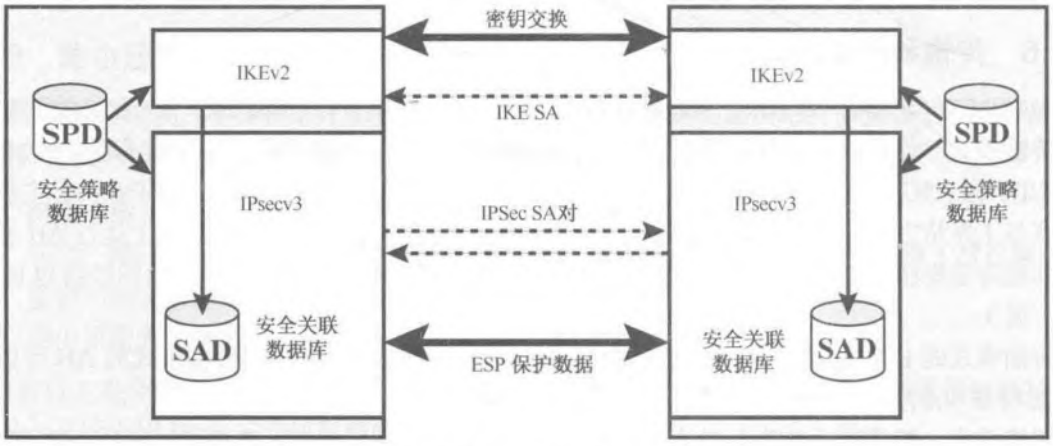


图 20.2 IPSec 结构

20.2.1 安全性关联

在 IP 的认证和保密机制中出现的一个核心概念是安全性关联（SA）。一个关联是一个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。

一个安全关联由三个参数唯一确定：

- **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** 这是 SA 的目的地址，可以是用户终端系统、防火墙或路由器。
- **安全协议标识** 来自外部 IP 报头并标识该关联是一个 AH 安全关联或 ESP 安全关联。

然而，在任何 IP 包中，安全关联由 IPv4 或 IPv6 头中的目的地址唯一标识，SPI 被封装于 AH 或 ESP 扩展头中。

20.2.2　安全性关联数据库

在任何 IPSec 实现中，一个名义上[①]称之为安全关联数据库（SAD）的对象，该对象定义了每个 SA 的相关参数。一个 SAD 中的安全关联通常用以下参数定义：

- **安全参数索引**　由一个 SA 的接收终端选择一个 32 位的值来唯一标识该 SA。在一个通向外部 SA 的 SAD 入口中，SPI 是用来创建 AH 包和 ESP 头部。在一个通向内部 SA 的 SAD 入口中 SPI 用来将通信映射到一个合适的 SA 上。
- **序列号计数器**　生成 AH 或 ESP 报头中序列号的 32 位值，详见 20.3 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）。
- **防止重放窗口**　用于判定一个内部 AH 或 ESP 数据包是否为重放，详见 20.3 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，并加上相应的操作指示（必须实现）。
- **IPSec 协议模式**　隧道、传送或通配符。
- **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（SPI）与认证和保密机制相联系。

IPSec 为用户提供多种方式在 IP 通信中实现 IPSec 服务。随后我们将看到，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。

20.2.3　安全性策略数据库

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。读者可以参见相关的 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，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 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：

- **远程 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（如位于防火墙之后）。

① 名义上在这里表示安全性关联数据库提供的功能必须在任何的 IPsec 实现中有所体现，但是该功能实现的具体形式却由生产商决定。

- **本地 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（如位于防火墙之后）。
- **下一层协议** IP 协议的报头（IPv4、IPv6 或者是扩展 IPv6）包含一个指定在 IP 之上运行协议的数据域（IPv4 的协议，IPv6 或扩展 IPv6 的下一个报头）。这是一个独立的协议码。假如使用 AH 或者是 ESP，IP 协议头将立即生成包中的 AH 和 ESP 头。
- **名称** 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。
- **本地和远程端口** 可以是单个 TCP 或 UDP 端口、一组端口和端口通配符。

表 20.2 主机 SPD 例子

协议	本地 IP	端口	远程 IP	端口	动 作	备 注
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT:ESP intransport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT:ESP intransport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS:avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

表 20.2 提供了在主机系统（与防火墙或路由器等网络系统相对）上的 SPD 的例子。上表反映了如下配置：本地网络配置包含两个网络。基本协作网络的配置的 IP 是 1.2.3.0/24。本地配置中也包含一个安全的 LAN，通常称为 DMZ，地址为 1.2.4.0/24。DMZ 通过防火墙可以免于外部和协作 LAN 中的其他成员的攻击。例子中主机的 IP 地址是 1.2.3.10，并且接受了认证可以连接到 DMZ 中的 1.2.4.10 服务器。

SPD 中的入口应该带自解释功能。例如，UDP 端口 500 是为 IKE 指点的端口号，任何为了 IKE 密钥交换目的从本地主机到远程主机传输的数据都会经过 IPsec 处理。

20.2.4 IP 通信过程

IPsec 是在逐个包上执行的。当实现了 IPsec 时，每一个发往外部的 IP 包都会在发送之前有 IPsec 逻辑进行处理，而每一个发往内部的 IP 包也会在接收到之后并且在传递给上一层（如 TCP 或 UDP）之前有 IPsec 处理。我们依次来看一下这两种情形的过程：

向外发包 图 20.3 给出了向外通信的 IPsec 处理过程的主要元素。例如将 TCP 层的上一层的数据块传递到 IP 层并形成 IP 包，IP 包是由 IP 头和 IP 正文组成的。然后执行如下步骤：

（1）IPsec 搜索与该包匹配的 SPD

（2）假如为找到匹配的 SPD，则丢弃该包并生成错误信息。

（3）假如找到匹配的 SPD，则由找到的第一个 SPD 入口决定往后的过程。假如对该包的策略是丢弃，则丢弃该包；假如对该包的策略是通过，则 IPsec 过程结束，IP 包就用于网络传输。

（4）假如对该包的策略是保护，则搜索匹配的 SAD。如果没有找到 SAD，则唤醒 IKE 用合适的私钥生成 SA 以及 SA 的入口。

（5）假如找到匹配的 SAD，则进一步的处理就由 SAD 决定。加密、认证或两者都执行，使用传输或隧道模式。然后 IP 包用于网络传输。

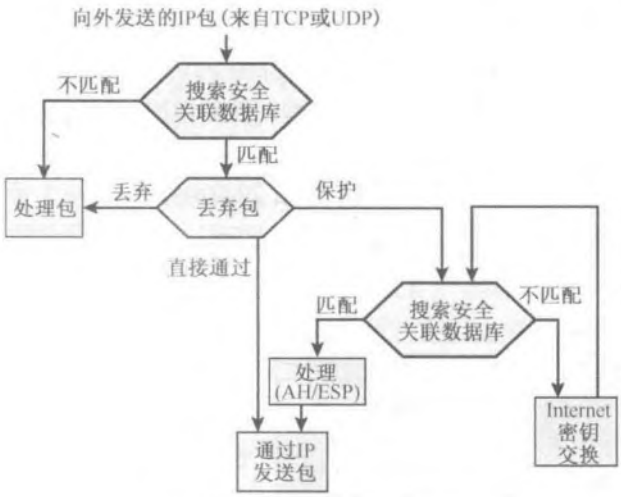


图 20.3　向外发包处理模型

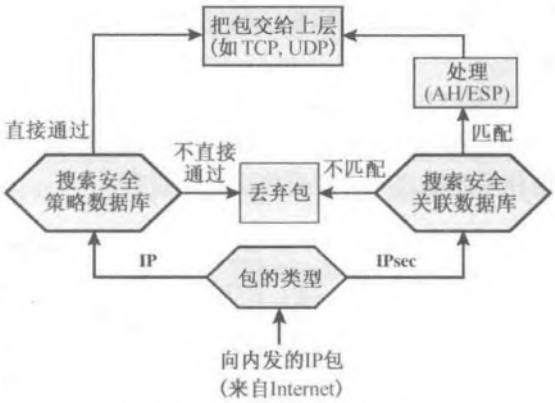


图 20.4　向内发包处理模型

向内发包　图 20.4 给出了向内通信的 IPsec 处理过程的主要元素。一个到来的 IP 包触发 IPsec 处理过程，然后执行如下步骤：

（1）IPsec 通过检查 IP 协议域（IPv4）或下一个头域（IPv6）来判断该包是一个非安全的 IP 包还是有 ESP 或 AH 头/尾的 IP 包。

（2）假如是非安全的包，IPsec 搜索匹配的 SPD。假如第一个匹配的入口的策略是通过，则处理 IP 头然后将包的正文传递给上一层，如 TCP 层。假如第一个匹配的入口的策略是保护或丢弃，或没有找到匹配的 SPD，则丢弃该包。

（3）如果是安全的包，IPsec 就搜索 SAD。假如没有找到匹配的 SAD 入口，则丢弃该包；否则 IPsec 执行合适的 ESP 或 AH 过程。然后处理 IP 报头并将包正文发送给上一层，例如 TCP 层。

20.3　封装安全性有效载荷

ESP 可以用来提供机密性、数据源认证性、无连接完整性和防止重放服务（一种部分序列完整性的形式）和（有限的）通信流机密性。这些服务的集合取决于建立安全性关联（SA）时选择的选项和其在网络拓扑中所处的位置。

ESP 可以和多种加密和认证算法协同工作，包括带认证的加密算法，如 GCM。

20.3.1　ESP 格式

图 20.5(a)是 ESP 包的格式，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，提供防止重放功能。
- **载荷数据（可变）**　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）。
- **完整性验证值（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。

当使用任何的结合型算法时，算法本身会被期望返回一个解密后的明文和指示完整性验证成功和失败的值。对于结合型算法，当选择检验完整性时，通常出现在 ESP 包尾部的 ICV 可以省略。当 ICV 省略并且要求检验完整性时，结合型算法就负责将一个与 ICV 相当的验证性的包加入到载荷数据中。

当 ESP 中有加密或认证的需要是，在载荷中会出现两个额外的域［如图 20.5(b)所示］，一个是初始值（IV）或随机数。假如使用隧道模式，IPsec 可能会假如通信流机密性（TFC）填充到载荷数据和填充域之间，随后将讲述其过程。



图 20.5 ESP 包格式

20.3.2 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向量（IV）这样的同步数据，则必须从载荷数据域报头取值，IV 通常作为密文的开头不被加密。

ICV 域是可选的，只有当选用了完整性服务时才存在，并且其要么是由单独的完整性算法提供，要么是由使用了 ICV 的结合型算法给出。ICV 的计算在对数据加密之后执行。按此顺序进行可以方便接收者在解密包之前快速检测并拒绝重放和伪造的包，因此在一定程度上降低了拒绝服务（DoS）攻击的影响。这也可以允许接收方可以对包并行处理，例如解密和完整性检测可以并行执行。值得注意的是，由于 ICV 没有用密码保护，因此必须使用一个基于密钥的完整性算法来计算 ICV。

20.3.3 填充

填充域功能如下：

- 如果加密算法需要原文满足一定的长度要求，则填充域可用于扩展原文长度（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，以及密文长度需要 32 位的整数倍，不足位用填充域来确保。
- 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。

20.3.4　防止重放服务

重放攻击是指攻击者在得到一个经过认证的包后，在以后的某个时刻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，将值写入序列号域，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），标记窗口的正确位置（如图 20.6 所示）。接收到包后的处理过程如下：

（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。

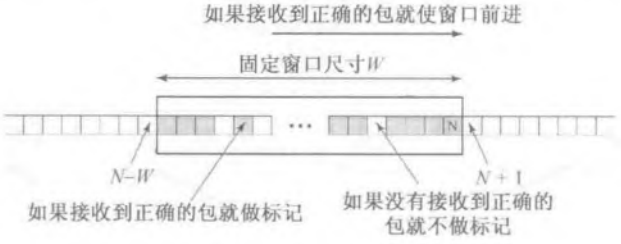


图 20.6　防重放机制

20.3.5　传输和隧道模式

图 20.7 给出了 IPsec ESP 服务所使用的两种方法。在图 20.7(a)，加密（和可选的认证）是直接提供在两个主机之间的。图 20.7(b)表明可以使用隧道模式操作来建立虚拟专用网（VPN）。在这个例子中，该组织有四个通过因特网互联的虚拟专用网。在内部网络的主机使用因特网来传输数据，但是并不与其他基于因特网的主机交互。通过在安全网关处终止到内部网络的隧道可以使主机避免执行安全性。前者的技术是通过传输模式的 SA 实现，后者的技术则是使用隧道模式的 SA 实现。

在本节中，我们大致了解 ESP 的两种模式。IPv4 和 IPv6 会有一些不同。开始我们使用图 20.8(a)的包格式。

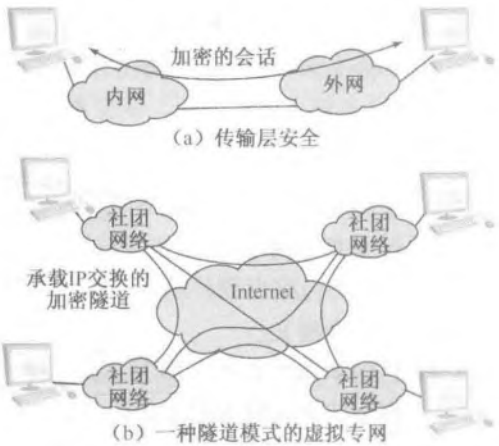


图 20.7　传输模式与隧道加密模式

传输模式的 ESP 传输模式的 ESP 使用加密和可选择的认证对由 IP（如 TCP 报文）携带的数据进行处理，如图 20.8(b)所示。

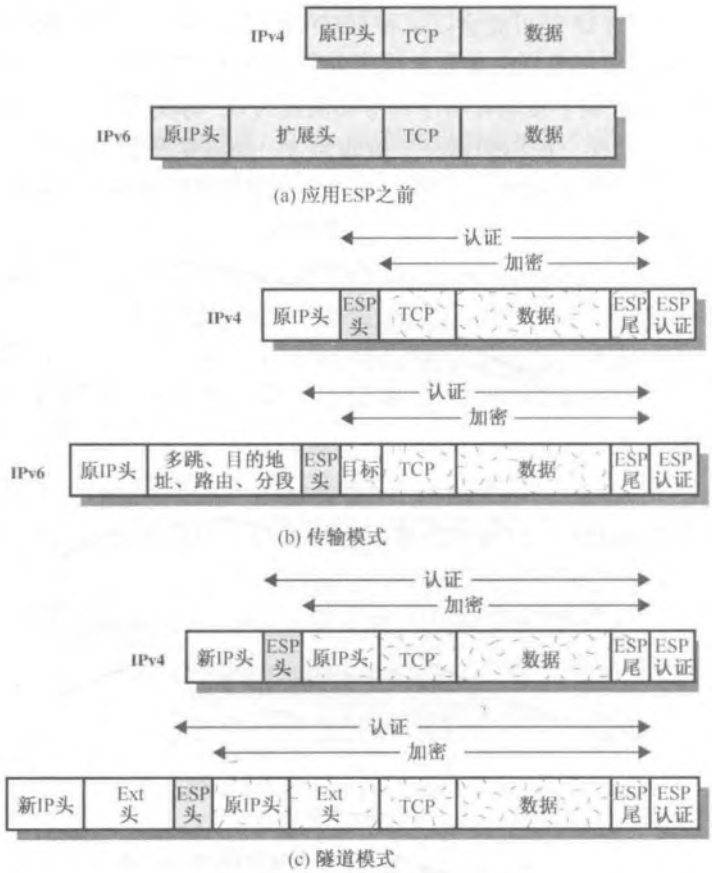


图 20.8 ESP 加密和认证范围

在传输模式下使用 IPv4，ESP 头被直接插入到传输层头（如 TCP、UDP、ICMP）之前，并将 ESP 尾（填充、填充长度和下个头域）放在 IP 包之后。假如选择了认证，ESP 认证数据域就添加在 ESP 尾之后。对整个传输层报文和 ESP 尾部进行加密。认证则覆盖了整个密文和 ESP 头部。

在 IPv6 的环境下，ESP 被看成是端到端的载荷。即它不会被中间路由检查或处理。因此，ESP 头部在 IPv6 基本头部和多跳、路由和扩展报文头部之后。根据语义要求目的选项扩展头可以在 ESP 头之前也可以在其之后。同样，认证覆盖了密文和 ESP 头。

传输模式操作可以归纳如下：

（1）在源端，由 ESP 尾部和整个传输层报文组成的数据块是被加密的，并且用密文代替明文形成 IP 包来传输。如果有认证则将其加上。

（2）然后将包路由到目的地。每一个中间路由都会检查和处理 IP 报头和明文 IP 扩展头，但是不检查密文。

（3）目的节点检查和处理 IP 头和任何明文 IP 扩展头。然后，根据 ESP 中的 SPI 解密包中的剩余部分恢复成明文传输层报文。

传输模式操作为任何使用它的应用程序提供机密性，这样可以避免在每个主机上的应用程序实现机密性。该模式的一个缺陷是可以根据传输的数据包进行通信量分析。

隧道模式的 ESP　隧道模式的 ESP 对整个 IP 包加密［如图 20.8(c)所示］。对该模式，ESP 头部预先放入了包中并加密了整个包和 ESP 尾部。该方法可以用来抵抗通信量分析。

因为 IP 头包含目的地址和源路由指示和跳转信息，所以用 ESP 头部传输加密的 IP 包比较困难。中间路由器无法处理该数据包。因此需要用新的 IP 头来封装整个数据块（ESP 头加密文加认证数据），该 IP 头中包含有足够的路由信息但不足以用于通信量分析。

传输模式适合于保护主机间的连接来支持 ESP 特性，而隧道模式则对于配置包含路由器和其他类型的安全网关来保护可信网络免于外部网络的侵袭是很有用的。在后者的情形下，加密仅仅在外部主机和安全网关之间以及两个安全网关之间进行。这减缓了内部网络主机加密的负担，同时也通过减少需要密钥的实体来降低密钥分发的任务量。另外，这还可以防止基于最终终端的通信量分析。

考虑到外部的主机可能希望和在防火墙之后的内部网络通信，在这种情况下就将 ESP 实现在外部主机和防火墙上。如下给出了从外部主机向内部主机发送传输层报文的步骤：

（1）源端准备了一个内部 IP 包，其目的地址是目标内部主机，该报文事先加入了 ESP 头部。然后该包和 ESP 会被加密并且可能会加上认证数据。处理后的数据块再用新的 IP 头部（基本头和可选扩展头，如 IPv6 中的路由和跳转选项）封装，新的 IP 报头的目标地址是防火墙。这就形成了外部 IP 包。

（2）外部包又被路由到目的地防火墙。每一个中间路由都会检查和处理外部 IP 头和任何外部 IP 扩展头，但是不会检查密文。

（3）目的防火墙检查和处理外部 IP 头和任何外部 IP 扩展头。然后根据 ESP 头中的 SPI，解密包中剩余的部分来恢复明文形成内部 IP 包。然后将该包发送给内部网路。

（4）内部包在经过若干路由传递到目的地主机。

图 20.9 给出了这两种方法的协议结构。

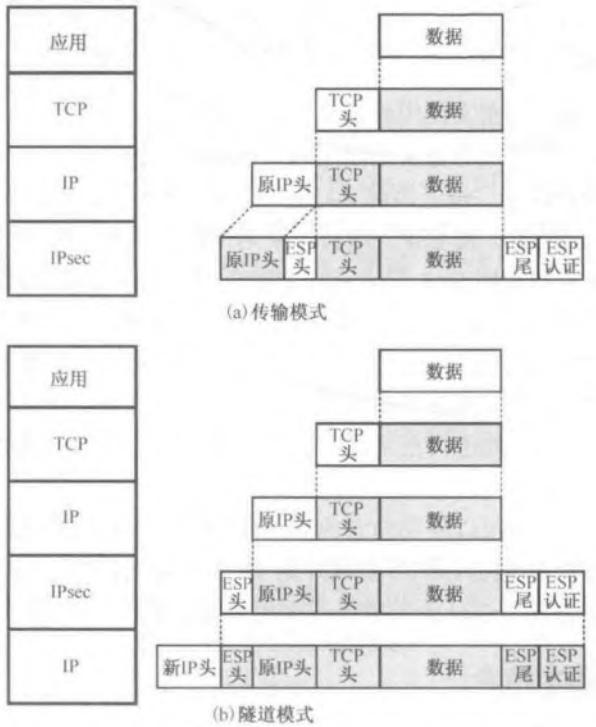


图 20.9　ESP 的协议操作

20.4 结合安全性关联

单个的 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来组合成束：

- **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（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，下面了解至少涉及一个隧道的 SA 组合。

20.4.1 认证性和机密性

在主机间要求传送经过加密和认证的 IP 包时加密和认证是可以组合的，以下是一些组合途径。

带认证的 ESP 该方法如图 20.8 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 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）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。

上述两种情况，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 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 头。与简单的使用 ESP 带认证的 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 的安全关联结束。此时，认证作用于 IP 载荷和除可变域外的 IP 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的密文和新的外部 IP 头。

20.4.2 安全关联的基本结合

IPSec 结构文档列举了 4 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 20.10 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。

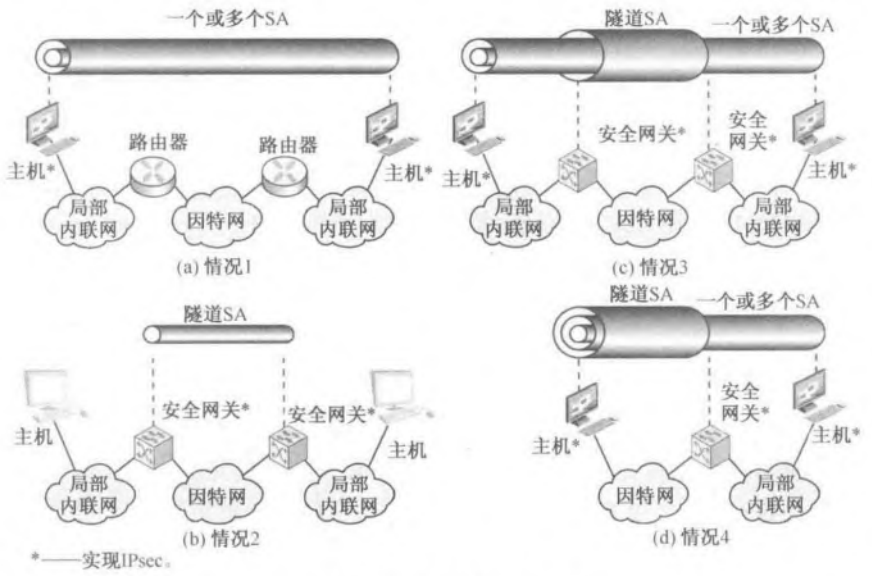


图 20.10　安全关联的基本组合

情况 1　为实施 IPSec 的终端系统间提供所需的安全机制，通过 SA 通信的任何两个终端系统，必须共享特定的密钥，可能的连接如下：

（a）传输模式的 AH
（b）传输模式的 ESP
（c）传输模式 AH 后紧跟 ESP（AH SA 内置于 ESP SA）
（d）在 AH 或 ESP 隧道模式中拥有（a）、（b）或（c）

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。

情况 2　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终端系统间的所有通信提供认证和/或保密。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。

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。

20.5　因特网密钥交换

IPSec 的密钥管理包括密钥的确定和分发。两个应用程序需要 4 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**　系统管理员手工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** 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，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。

尽管在基本功能上两者是相似的，但在 IKEv2 中就不再使用 Oakley 和 ISAKMP 了，因为这个在 IKEv1 中使用的 Oakley 和 ISAKMP 有显著差别。在本节中，我们将介绍 IKEv2 规范。

20.5.1 密钥确定协议

IKE 密钥确定协议是 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

$$K=(Y_B)^{X_A} \bmod q=(Y_A)^{X_B} \bmod q=a^{X_AX_B} \bmod q$$

Diffie-Hellman 算法有两个很好的特性：

- 仅在需要时创建密钥，不需要长时间的存放密钥，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。

然而文献[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（如图 10.2 所示）。
  （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（clogging attack）。受害者需要花费相当多的资源做无意义的乘方、取模运算。

IKE 密钥确定是一种保持了 Diffie-Hellman 优点而去掉了其缺点的一种算法。

IKE 密钥确定特性 IKE 密钥确定算法有如下 5 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。

IKE 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、目的端口和一个本地生成的秘密值能快速生成 Hash 值（如 MD5）。

IKE 密钥确定支持在 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 位模数的乘方取模

$$q = 2^{768} - 2^{704} - 1 + 2^{64} \times (\lfloor 2^{638} \times \pi \rfloor + 149686)$$
$$\alpha = 2$$

- 1024 位模数的乘方取模

$$q = 2^{1024} - 2^{960} - 1 + 2^{64} \times (\lfloor 2^{894} \times \pi \rfloor + 129093)$$
$$\alpha = 2$$

- 1536 位模数的乘方取模
  - 参数待定
- 2^{155} 的椭圆曲线
  - 生成器（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
  - 生成器（十六进制）：$X = 18$，$Y = D$
  - 椭圆曲线参数（十六进制）：$A = 0$，$Y = 1EE9$

前三种是使用乘方取模的传统Diffie-Hellman算法，后两种是使用椭圆曲线模拟Diffie- Hellman，参见第 10 章。

IKE 密钥确定使用随机数来防止重放攻击。每个随机数是一个本地生成的伪随机数，它在应答中出现，并在交换的特定部分加密以对它进行保护。

在 IKE 密钥确定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 在生成时使用重要参数如用户 ID、随机数等。
- **公钥加密** 使用发送方的私钥对参数如 ID、随机数加密来验证交换。
- **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。

IKEv2 交换 IKEv2 涉及消息的相互交换。交换的第一对称为初始交换［如图 20.11(a)所示］。在第一次交换中，对等实体间交换关于密码算法和其他安全参数的信息。这些安全参数是通信双方准备和随机数与 Diffie-Hellman（DH）共同使用的值。第一次交换的结果是建立一个称之为 IKE SA（如图 20.2 所示）的特殊 SA。该 SA 定义了对等实体间用于进行后续交换的安全信道的参数。因此，所有随后的 IKE 信息交换都有加密和消息认证保护。在第二次交换中，双方进行相互认证并建立第一个 IPsec SA 放入 SADB 中用于保护两者间的通信数据（如非 IKE）。因此为了建立用于通信的第一个 SA 需要 4 个消息。

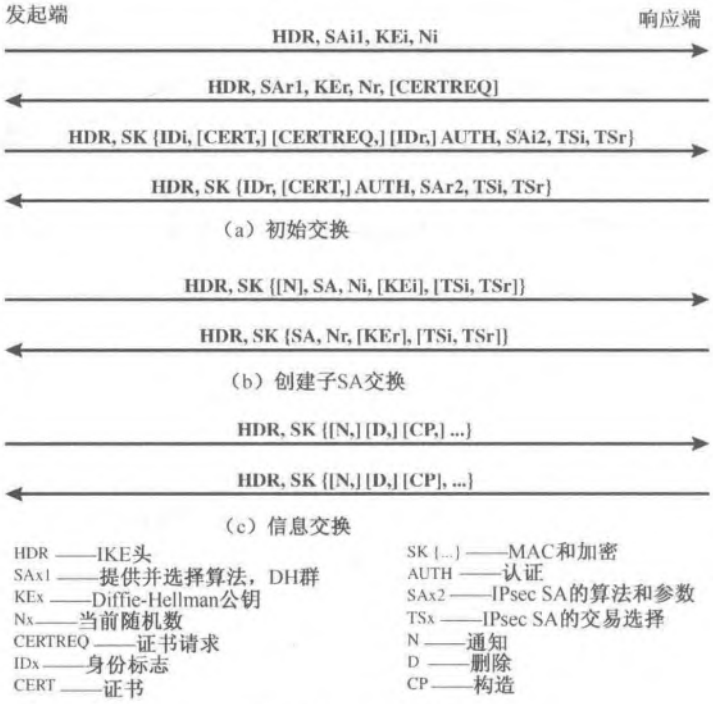


图 20.11 IKEv2 交换

CREATE_CHILD_SA 交换用于进一步建立为保护通信服务的 SA。信息交换用户交换管理信息、IKEv2 的错误信息和其他指示信息。

20.5.2 报头和有效载荷格式

IKE 定义了建立、协商、修改和删除安全关联的过程和包格式。IKE 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的密钥交换协议、加密算法和认证机制无关的一致性框架。

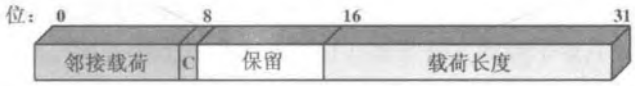
IKE 报头格式　IKE 报文由 IKE 头和一个或多个载荷组成，并包含在传输协议之中。规范指明了在实现时必须在传输协议中支持 UDP。

图 20.12(a)指明了 IKE 报文的头格式，它由以下域组成：

- **发起者 SPI（64 位）**　由发起者选择一个值，以确定一个独立的安全性关联（SA）。
- **响应者 SPI（64 位）**　由应答者选择一个值，以确定独立的 IKE SA。
- **邻接载荷（8 位）**　表明报文中第一个载荷的类型，载荷将在下一节中讨论。
- **主版本号（4 位）**　使用的 IKE 的主版本号。
- **从版本号（4 位）**　使用的从版本号。
- **交换类型（8 位）**　表明交换类型，在本节中稍后讨论。
- **标志（8 位）**　IKE 交换的可选项集合，目前定义了两位：当跟在 IKE 报头后面的所有载荷都使用了此 SA 的加密算法加密，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）**　报文的唯一标识。
- **长度（32 位）**　报文（头+所有载荷）总的字节长度。



(a) IKE头



(b) 一般载荷头

图 20.12　IKE 格式

IKE 载荷类型　所有 IKE 载荷开始于如图 20.12(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。

当接收者不理解前一个载荷中下一个载荷域的载荷类型码，并且发送者希望其跳过该载荷时，critical 位就置位 0。而当接收者不理解载荷类型，并且发送者希望其拒绝整个消息时，将 critical 置为 1。

表 20.3 总结了在 IKE 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。载荷是一个复杂的层次化结构，并包含有多个建议，每一个建议又包含多个协议，每个协议又包含多个转换，每个转换包含多个属性。载荷中的元素可以形式化为

- **建议**　该结构包含一个建议号，协议标识（AH、ESP 或 IKE），转换号的表示和一个转换子结构。当一个建议中包含多个协议时，在一个建议号下就会有一个建议的子结构。
- **转换**　不同的协议支持不同的转换类型。转换通常是用来定义使用在特定协议中的密码算法。
- **属性**　每一个转换会包含多个用于修改和完成转换规范的属性，如密钥长度。

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。

表 20.3 IKE 载荷类型

类 型	参 数
安全性关联	建议
密钥交换	DH 群#，密钥交换数据
标识	标识类型，标识数据
证书	证书编码，证书数据
证书请求	证书编码，证书中心
认证	认证方法，认证数据
随机数	随机数值
公告	协议标识，SPI 大小，公告消息类型，SPI，公告数据
删除	协议标识，SPI 大小，SPI 的#，SPI
生产商标识	生产商标识
通信选择器	通信选择器（TS）的值，通信选择器
加密	初始向量，加密的 IKE 载荷，填充，填充长度，ICV
配置	配置（CFG）类型，配置属性
扩展认证协议	扩展认证协议（EAP）消息

标识载荷用于确定通信方的标识和使用的认证信息。一般，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 限制的 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）
- 认证撤销表（ARL）
- SPKI 证书

在任何 IKE 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签证机构 CA。

在认证载荷中包含了用于消息认证目的的数据。当前定义了的认证方法类型有 RSA 数字签名、共享密钥消息完整性码和 DSS 数字签名。

随机数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，下表列出了 IKE 公告信息：

错 误 消 息	状 态 消 息
Unsupported Critical	Initial Contact
Payload	Set Window Size
Invalid IKE SPI	Additional TS Possible
Invalid Major Version	IPCOMP Supported
Invalid Syntax	NAT Detection Source IP
Invalid Payload Type	NAT Detection Destination IP
Invalid Message ID	Cookie

（续表）

错 误 消 息	状 态 消 息
Invalid SPI	Use Transport Mode
No Proposal Chosen	HTTP Cert Lookup Supported
Invalid KE Payload	Rekey SA
Authentication Failed	ESP TFC Padding Not Supported
Single Pair Required	Non First Fragments Also
No Additional SAS	
Internal Address Failure	
Failed CP Required	
TS Unacceptable	
Invalid Selectors	

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。

生产商标识载荷包含一个生产商定义的常量。生产上用该常量来标识并识别他们的设备。该机制可以使生产商可以在满足后向兼容性的同时尝试一些新的特性。

通信选择器载荷允许对等实体通过 IPsec 服务标识数据包的流。

加密的载荷包含其他载荷的加密形式。加密载荷的格式与 ESP 相似。当加密算法要求时会有初始向量（IV）并且当要求有认证时会有 ICV。

配置载荷用于 IKE 对等实体间交换配置信息。

扩展认证协议（EAP）载荷允许 IKE SA 用 EAP 进行认证，详见第 16 章。

20.6　密码学套件

IPsecv3 和 IKEv3 协议依赖于多种密码算法。如本书中所描述的，对于每种应用有多个密码算法，每种算法又有多个参数，如密钥长度。为了提升互操作性，有两个 RFC 文档定义了各种应用的推荐密码算法和参数。

RFC 4038 定义了两个密码套件用于建立虚拟专用网。在 2005 年提出 IKEv2 时，VPN-A 与在 IKEv1 中使用的 VPN 是匹配的。VPN-B 提供了更强的安全性并且推荐在 IPsecv3 和 IKEv2 等新的 VPN 中使用。

表 20.4(a)列出了这两个套件的算法和参数。关于这两个套件还有一些说明。值得注意的是，对于对称加密算法，VPN-A 使用了 3DES 和 HMAC，而 VPN-B 使用了 AES。使用了三种密钥算法：

- **加密**　对于加密使用了加密块链（CBC）模式。
- **消息认证**　对于消息认证，VPN-A 使用了 HMAC 和 SHA-1 并且输出为 96 位，而 VPN-b 使用了 CMAC 并且输出也为 96 位。
- **伪随机函数**　IKEv2 通过重复使用用于消息认证的 MAC 来生成伪随机数。

RFC 6379 定义了 4 类符合美国国家安全局（NSA）B 套件规范的可选密码算法套件。在 2005 年，NSA 在密码现代化工程[LATT09]中提出了 B 套件，其中定义了保护敏感信息的算法和强度。在 RFC 4869 中定义的 4 种套件为 ESP 和 IKE 提供了选择。这 4 个套件是根据密码算法强度和 ESP 中是同时提供机密性和完整性还是仅提供完整性来区分的。上述 4 种套件比 RFC 4308 中定义的两种 VPN 的安全强度更高些。

表 20.4(b)列出了这两个套件的算法和参数。正如 RFC 4308 所述，有如下三类密钥算法：

- **加密** 对于 ESP，通过使用 128 位或 256 位 AES 密钥的 GCM 模式提供认证的加密。IKE 加密使用了 CBC，如 VPN 套件。
- **消息认证** 对于 ESP，假如只要求认证，则使用 GMAC。如第 12 章中所讨论的，GMAC 是 GMC 的认证部分。对于 IKE，消息认证通过使用 HMAC 和 SHA-3 中的某个 Hash 函数来提供。
- **伪随机函数** 如 VPN 套件，IKEv2 通过重复使用用于消息认证的 MAC 来生成伪随机数。

对于 Diffie-Hellman 算法，规定使用椭圆曲线群上对素数求模。对于认证性，也有使用椭圆曲线上的数字签名。早期的 IKEv2 文档使用了基于 RSA 的数字签名。当使用 ECC 时可以使用更短的密钥来实现相当或更好的强度。

表 20.4 IPsec 的密码套件

	VPN-A	VPN-B
ESP 加密	3DES-CBC	AES-CBC（128 位密钥）
ESP 完整性	HMAC-SHA1-96	AES-XCBC-MAC-96
IKE 加密	3DES-CBC	AES-CBC（128 位密钥）
IKE PRF	HMAC-SHA1	AES-XCBC-PRF-128
IKE 认证性	HMAC-SHA1-96	AES-XCBC-MAC-96
IKE DH 群	1024-bit MODP	2048-bit MODP

（a）虚拟专用网（RFC 4308）

	GCM-128	GCM-256	GMAC-128	GMAC-256
ESP 加密/完整性	AES-GCM（128 位密钥）	AES-GCM（256 位密钥）	Null	Null
ESP 完整性	Null	Null	AES-GMAC（128 位长度）	AES-GMAC（256 位长度）
IKE 加密	AES-CBC（128 位密钥）	AES-CBC（256 位长度）	AES-CBC（128 位长度）	AES-CBC（256 位长度）
IKE PRF	HMAC-SHA-256	HMAC-SHA-384	HMAC-SHA-256	HMAC-SHA-384
IKE 完整性	HAMC-SHA-256-128	HMAC-SHA-384-192	HMAC-SHA-256-128	HMAC-SHA-384-192
IKE DH 群	256 位随机 ECP	384 位随机 ECP	256 位随机 ECP	384 位随机 ECP

（b）NSA B 套件（RFC 6379）

20.7 关键术语、思考题和习题

关键术语

防止重放服务
认证头部（AH）
封装安全有效载荷（ESP）
因特网密钥交换（IKE）
隧道模式
因特网安全性关联和密钥管理（ISAKMP）
IP 安全性（IPsec）
IPv4
IPv6
Oakley 密钥确定协议
重放攻击
安全性关联（SA）
传输模式

思考题

20.1 给出 IPSec 的一个应用实例。

20.2 IPSec 提供哪些服务？

20.3 哪些参数标识 SA，哪些参数刻画一个特定 SA 的本质？

20.4　指出传输模式和隧道模式的区别?

20.5　什么是重放攻击?

20.6　为什么 ESP 含填充域?

20.7　形成 SA 束有哪些基本方法?

20.8　IPSec 中 Oakley 密钥确定协议和 ISAKMP 的作用是什么?

习题

20.1　描述并解释表 20.2 中的每个实体。

20.2　为 AH 画一个与图 20.8 类似的示意图。

20.3　分别列出 AH 和 ESP 提供的主要安全性服务。

20.4　在讨论 AH 的处理中，曾经提到，并非 IP 头中的所有字段都参与 MAC 的计算。

（a）对于 IPv4 头的每个域，指明哪些是不变的、那些会变但可以预计的、哪些是随意的（0 优先级 ICV 计算）

（b）对 IPv6 头的每一个域，问题同（a）。

（c）对 IPv6 扩展头的每一个域，问题同（a）。

在各种情况下，判断你对各个域的理解。

20.5　假设当前的重放窗口从 120 ~ 530。

（a）假如下一个到来的认证包的序列号为 105，接收方如何处理该包，并且处理之后窗口的参数如何变化。

（b）假如下一个到来的认证包的序列号为 440，接收方如何处理该包，并且处理之后窗口的参数如何变化。

（c）假如下一个到来的认证包的序列号为 540，接收方如何处理该包，并且处理之后窗口的参数如何变化。

20.6　当使用隧道模式时，构造一个新的外部 IP 头。IPv4 和 IPv6 都指明了外部包的外部 IP 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。那么，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?

20.7　端对端的认证和加密应该在两台主机之间。参照图 20.8 中的表示：

（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。

20.8　IPSec 构架文档中说，当两个传输模式 SA 被绑定，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?

20.9　对于 IKE 的密钥交换，在每个消息中哪些参数是哪种 ISAKMP 载荷类型。

20.10　IPsec 在协议栈中处于什么位置?

附录 A　用于密码学和网络安全教学的项目

Analysis and observation, theory and experience must never disdain or exclude each other; on the contrary, they support each other.

— On War, Carl Von Clausewitz

很多教师都相信做项目对于加深对密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交互活动。项目可以加深学生对书本上概念的理解，使学生对加密算法和加密协议的工作原理有更深入的理解，同时，这也可以激励学生，让他们相信自己不仅能够掌握安全相关的技术细节而且能够实现它。

本书已经尽可能清楚地介绍了密码学和网络安全方面的概念，并且提供很多的课后作业来强化这些概念。然而，很多教师也希望能够通过做项目来辅助学习这些知识。本附录在这方面提供一些指导，并对于教师资源中心（Instructor's Resource Center，IRC）中的可用资源进行了介绍，这些教学资源都可以在 Prentice Hall 网站上获取。这些资料包括 12 类项目和一些其他的练习。

- Sage 计算机代数项目
- 黑客项目
- 分组密码项目
- 实验室练习
- 研究项目
- 编程项目
- 安全评估实践
- 书面作业
- 防火墙项目
- 案例研究(case studies)
- 书面作业
- 阅读/报告作业
- 课题讨论

A.1　Sage 计算机代数项目

本书新增的最重要的内容之一就是在密码算法示例和作业中使用了 Sage。Sage 是开源、跨平台的免费软件，能够在数学和计算机代数系统的学习过程中提供强大、灵活和易学的软件包。计算机代数系统（Computer Algebra System，CAS）是指不仅能够进行数值运算，还能够进行符号运算的软件。CAS 被用于教学已经有数十年的时间，目前已有大量相关的使用文献。在密码学相关的课程中，使用 CAS 作为实践的工具再合适不过了。

与 Mathematica，Maple 和 MATLAB 等系统不同，Sage 没有许可限制，也不需要支付相关的费用。因此 Sage 可以在学校的计算机和网络上使用，学生也可以在家中将其下载到他们的个人计算机上使用。使用 Sage 的另外一个好处是学生可以掌握一个非常强大、灵活的工具来帮助计算几乎所有的数学问题，而不仅限于密码学。Sage 网站（http://www.sagemath.org）提供了大量文档，介绍如何在各种类型的计算机平台上安装、设置和使用 Sage，以及如何通过 Web 在线使用 Sage。

在对密码算法中数学部分的教学过程中，使用 Sage 能够显著增强教学效果。附录 B 提供了 Sage 使用的大量示例，其中涵盖了很多密码学概念。在线附录 C 按照密码学概念的分类给出了习题集，使学生能够通过练习，在实践中加深对密码算法的理解。在线附录 C 都在网上提供了电子文档，因此学生不需要再把附录中的代码逐行地重复录入。

教师资源中心（IRC）提供了在线附录 C 中的所有练习答案。

附录 B 和在线附录 C 中的所有示例和作业都是由微软和华盛顿大学的 Dan Shumow 开发的。

A.2　黑客项目

本项目的目标是以黑客的身份通过一系列步骤入侵一个“公司”的网络。该公司的名字是 Extreme In Security 公司，从名字可以看出该公司有一些安全漏洞，聪明的黑客能够通过侵入其网络访问到重要的信息。IRC 中提供了部署这个公司网站所需要的内容。学生的目标是通过获取该公司下周关于一个政府项目的秘密询价信息，从而获得这个政府项目的合同信息。

学生首先应该从网站开始，逐步找到进入公司网络的方法。在每一步中，如果学生成功完成了本步骤，就会得到如何进行下一步骤的提示信息。

该项目可以通过尝试下面三种方式来完成：

（1）不提供任何形式的帮助。
（2）提供一些提示。
（3）提供全部的指导。

IRC 提供了本项目需要的所有文件，包括：

（1）网站安全项目。
（2）网站黑客练习（XSS 和脚本攻击），包括客户端和服务器端漏洞扫描。
（3）上述功能的安装和使用文档。
（4）一个网站入侵的教学幻灯片。该文件通过抓屏，清晰演示了如何使用这些练习材料。

该项目由达科他州立（Dakota State）大学的 Sreekanth Malladi 教授设计和实现。

A.3　分组密码项目

本实验对 AES 加密算法的执行过程进行跟踪，然后进行一轮手工的计算，来探索不同分组密码的使用。本实验也包括 DES 算法。在实验中，由一个在线的（或离线下载的）Java applet 程序来执行 AES 或 DES 的运算。

对于 AES 和 DES，实验都分为下面三部分：

- **分组密码内部组件**　该部分对明文加密，并考察每一轮的中间结果。对于 AES 和 DES 都有在线计算器来提供中间结果和最终密文的计算结果。
- **分组密码轮函数**　该部分对单轮运算进行手工计算，并可以将计算的结果与计算器提供的结果进行对比检查。
- **分组密码不同模式的使用**　使学生能够对 CBC 模式和 CFB 模式进行比较。

IRC 提供了在你自己的网站上实施本实验时所需的.html 和.jar 文件。这些材料在本书网站的“学生资源”部分提供，请点击相关链接。

本项目由澳大利亚国防研究院的 Lawrie Brown 提供。

A.4　实验室练习

作为教师资源中心的一部分，普度大学的 Sanjay Rao 教授和 Ruben Torres 教授准备了一套实验

室练习项目。这些项目需要在 Linux 系统上编程实现，但理论上也适用于任何类 UNIX 环境。这些实验室练习项目可以在实现安全功能和应用为学生提供了实际操作的经验。

A.5 研究项目

为了加深对课程中介绍的基本概念的理解，同时教给学生研究的技巧，最有效的方法就是给学生布置一个研究项目。这种项目可以是文献搜索，也可以是对供应商产品、研究室活动和标准化成果等的网上搜索。项目可以分配给一个组，比较小的项目也可以分配给个人。无论是何种形式，最好要求学生在本学期的早期提交项目提案，这样教师们就有时间来评估项目提案所给出的题目以及工作量是否合适。学生的科研报告应该包括以下几个部分：

- 提案的格式
- 最终报告的格式
- 中期和终期的进度安排表
- 可能的项目题目列表

学生们可以从 IRC 提供的题目列表中选择一个项目或者自己设计一个相当的项目。IRC 提供了一份推荐的提案和最终报告的格式，以及一份具有 15 个研究性项目的题目列表。

A.6 编程项目

进行编程项目是一种很有用的教学方式。像独立编写新的安全工具这样的编程项目具有好几个吸引人的特点：

（1）教师能够从多个密码学和网络安全概念中选择一个主题来布置项目。
（2）该项目可以是平台和语言无关的。学生可以在任何一台机器用任何语言进行编程。
（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。大一点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会因此被落在后面。大一点的项目通常会让好学生投入更多的时间和精力。而小一点的项目拥有更高的“概念转换代码”的效率，而且因为大多数项目都可以布置给学生完成，所以这样的项目涉及的领域可以更多样化。

和研究项目一样，学生们首先应该提交项目提案。学生们的实验报告同样应该包括上节中提到的那几个部分的内容。IRC 中提供了 12 个相应的编程项目。

下面这些人提供了 IRC 中的研究和编程项目，他们是：哥伦比亚大学的 Henning Schularinne，俄勒冈州立大学的 Cetin Kaya Koc，可信信息系统公司和乔治·华盛顿大学的 David M.Balenson。

A.7 安全评估实践

对一个现有组织的当前基础设施和实施方案进行考查，是提高安全评估能力的最好办法。IRC 包含了一系列这样的活动。学生可以独立或分组来选择合适的小型或中型组织，然后访问这些组织中的一些关键人员，来选择所要进行的与该组织的 IT 基础设施和实施方案相关的安全风险评估、审查任务。学生根据审查的结果，可以提供一些合适的改进建议，来提高该组织中的 IT 安全性。这些活动可以帮助学生更好地理解现有的安全实现机制，增强对安全性进行评估和改进的能力。

本项目由澳大利亚国防研究院的 Lawrie Brown 提供。

A.8　防火墙项目

网络防火墙的实现对于刚开始学习的学生来说，是一件比较难的事。IRC 提供了一个网络防火墙虚拟化工具来介绍和讲授网络安全及防火墙配置相关的知识。这个工具可以用来讲授和加深对防火墙的一些关键概念的理解，包括边界防火墙的使用和目的、隔离子网的使用、数据包过滤的目的以及简单包过滤防火墙的缺点。

IRC 提供了一个完全可以移植的.jar 文件以及一系列的练习。这个工具和相关练习由美国空军研究院提供。

A.9　案例研究

教师结合案例讨论可以使得学生更主动学习。IRC 提供了下列领域中的案例研究：

- 灾难恢复
- 防火墙
- 应急响应
- 物理安全
- 风险
- 安全策略
- 虚拟化

每个案例分析都包括学习目标、案例描述和一系列的案例讨论问题。每个案例分析都基于现实世界的情况，并提供了描述案例的论文或报告。

这些案例分析是由 North Carolina A&T 州立大学开发的。

A.10　书面作业

学习密码学和网络安全这样的技术类课程，书面作业可以起到事半功倍的效果。编写课程（WAC）活动（http://wac.colostate.edu）的忠实支持者报告了在促进学习方面书面作业的众多好处。书面作业可以使学生就某个专题做深入细致、全面的思考。另外，书面作业还可以克服学生的一种不好倾向：总是希望最快地学完一门课，却只了解事实和解决问题的技巧，而不是深入理解课程中的问题。

教师资源中心里按章推荐了很多的书面作业。教师最后也许会发现这是讲授本课时最重要的部分。如果你就这个问题有所反馈，并且能够推荐一些其他书面作业的话，我将非常感谢。

A.11　阅读/报告作业

另外一个可以加深学生对课堂知识理解和增加学生研究经验的有效方法是布置一些论文让学生阅读和分析，要求学生对于指定的论文给出简要的读书报告。教师资源中心提供了推荐性的论文列表，每章给出一到两篇。教师资源中心提供了每篇论文的 PDF 文件，同时，教师资源中心还提供了一份推荐的布置作业可使用的措辞。

A.12　课题讨论

增加合作经验的一个行之有效的方法就是，提供课题给学生讨论，IRC 中提供了很多这样的课题。每个课题都是和本书中的内容相关的。教师可以在课程教学中安排这些课题的讨论，使得学生可以在上课时间、在线的聊天室或者任何一个消息公告板上展开讨论。同样地，我将非常感谢任何对这个方面的反馈，以及任何提供其他讨论题目的建议。

附录 B　Sage 示例

By Dan Shumow

—— 华盛顿大学(University of Washington)

本附录包括大量的密码学概念的实现例子，并按照本书章节的组织进行了分类。所有的例子都是在 Sage 上实现的[①]。关于如何使用 Sage 以及 Sage 的语法和操作的简要介绍请参见在线附录 C。首先简要介绍一些基本的 Sage 矩阵和线性代数运算。

本节的例子应该都很容易理解，如果读者对于 Sage 代码的理解有困难，可以参考在线附录 C 的 C.2 节。

B.1　线性代数和矩阵函数

Sage 包含线性代数和矩阵函数的计算功能。下面展示可用于密码学的一些基本函数。

在 Sage 中将矩阵表示为数字组成的列表，传递给 `matrix` 函数生成。例如，向下面一样，给 `matrix` 函数传入一组由整数组成的列表：

```
sage: M = matrix([[1, 3],[7,9]]); M
[1 3]
[7 9]
```

另外，也可以像下面一样传递有理数组成的列表：

```
sage: M = matrix([[1/2, 2/3, 3/4],[5, 7, 8]]); M
[1/2 2/3 3/4]
[ 5 7 8]
```

还可以使用 `IntegerModRing` 函数（随后将介绍该函数）来对输入取模

```
Sage: R = IntegerModRing(100)
sage: M = matrix(R, [[1],[102],[1003]]); M
[1]
[2]
[3]
```

也可以指定输入是在有限域上的（随后也将介绍）

```
sage: F = GF(2);
sage: M = matrix(F, [[1, 2, 0, 3]]); M
[1 0 0 1]
```

Sage 还支持矩阵的乘法、加法和求逆矩阵。如下所示：

```
sage: M1 = matrix([[1, 2],[3,4]]);
sage: M2 = matrix([[1,-1],[1, 1]]);
sage: M1*M2
[3 1]
```

① 在本书的网站上有所有的 Sage 源代码，读者可以下载和执行任何想要的程序，访问信息请参见前言。

```
[7 1]

sage: M1 + M2
[2 1]
[4 5]

sage: M2^-1
[ 1/2 1/2]
[-1/2 1/2]
```

B.2 第 2 章 数论基础

例 1 中国剩余定理

```
def chinese_remainder_theorem(moduli, residues):
    r"""
    Function that implements the chinese remainder
    theorem.

    INPUT:

        moduli - list or positive integers.

        residues - list of remainders such that remainder
    at position j results when divided by the corresponding
    modulus at position j in moduli.

    OUTPUT:

        x - integer such that division by moduli[j] gives
    remainder residue[j].

    """

    if (len(moduli) != len(residues)):

    raise   ValueError,   "expected   len(moduli)   ==
    len(residues)"

    M = prod(moduli);

    x = 0;
for j in xrange(len(moduli)):
    Mj = moduli[j]
    Mpr = M/Mj

    (Mj_Mpr_gcd, Mpr_inv, Mj_inv) = xgcd(Mpr, Mj)

    Mpr_inv = Mpr_inv

    if (Mj_Mpr_gcd != 1):

    raise  ValueError,  "Expected  all  moduli  are
    coprime."

    x += residues[j]*Mpr*Mpr_inv;

return x;
```

例 2 Miller-Rabin 素性测试

```
r"""
EXAMPLES:
```

```
    sage: MILLER_RABIN_TEST(101)
    False

    sage: MILLER_RABIN_TEST(592701729979)
    True

"""

def MILLER_RABIN_TEST(n):
    r"""

    This function implements the Miller-Rabin Test.
    It either returns "inconclusive" or "composite."

    INPUT:

        n - positive integer to probabilistically
        determine the primality of.

    OUTPUT:

    If the function returns False, then the test was
    inconclusive.

        If the function returns True, then the test was
    conclusive and n is composite.

    """

    R = IntegerModRing(n); # object for integers mod n
    # (1) Find integers k, q w/ k > 0 and q odd so that
    (n-1) == 2^k * q
    q = n-1
    k = 0
while (1 == (q % 2)):
    k += 1
    q = q.quo_rem(2)[0] # q/2 but with result of type
    Integer

# (2) select random a in 1 < a < n-1

a = randint(1,n-1)

a = R(a) # makes it so modular exponentiation is done
fast

# if a^q mod n == 1 then return inconclusive
if (1 == a^q):
    return False

# (3) for j = 0 to k-1 do: if a^(2^j * q) mod n = n-1
return inconclusive

e = q

for j in xrange(k):
    if (n-1) == (a^e):
        return False
    e = 2*e

# (4) if you've made it here return composite.
return True
```

例 3 模幂运算（平方和乘法）

```
    def ModExp(x,e,N):
        r"""
```

```
        Calculates x^e mod N using square and multiply.

        INPUT:

            x - an integer.
            e - a nonnegative integer.
            N - a positive integer modulus.

        OUTPUT:

            y - x^e mod N

        """

            e_bits = e.bits()
            e_bitlen = len(e_bits)

        y = 1

        for j in xrange(e_bitlen):

            y = y^2 % N

        if (1 == e_bits[e_bitlen-1-j]):
            y = x*y % N

    return y
```

例 4 使用 Sage 内建的函数计算 CRT

Sage 已经内建了执行 CRT 的函数。Sage 中有好几个执行 CRT 计算的函数，其中最简单的函数可以执行两个模的 CRT，其中 CRT（或小写 crt）调用方式如下：

```
crt(a,b,m,n)
```

结果会返回同时满足与 a 模 m 以及 b 模 n 同余的整数。其中所有参数都是整数，并且参数 m 和 n 互素。该函数的示例如下：

```
sage: CRT(8, 16, 17, 49)
-3120

sage: CRT(1,2,5,7)
16

sage: CRT(50,64,101,127)
-62166
```

如果需要对更多的余数和模执行 CRT 计算，Sage 提供了 CRT_list 函数。

```
CRT_list(v, modulii)
```

其中，v 和 modulii 都是整数列表且列表长度相同，并且，modulii 中的元素必须互素。输出是与所有 v[i]模 modulii[i]（这里 i 取遍 len(v)）都同余的整数。例如，上面最后一个对 CRT 函数的调用可以通过如下另一种方法实现：

```
sage: CRT_list([50,64],[101,127])
1969
```

注意答案与上例不同，当然你可以对两个答案检查是否满足中国剩余定理。更大规模的例子如下：

```
sage: CRT_list([8, 20, 13], [49, 101, 127])
608343

sage: CRT_list([10,11,12,13,14],[29,31,37,41,43])
36657170
```

CRT_basis 函数可以用来预计算一组给定模的取值。如果 modulii 是一组互素的模，则

CRT_basis(modulii)返回一个列表 a。在列表 a 中，如果 x 是模 modulii 的剩余，则 CRT 的输出可通过如下求和得出：

```
a[0]*x[0] + a[1]*x[1] + ... + a[len(a)-1]*x[len(a)-1]
```

当使用 CRT_basic 函数计算上面 CRT_list 函数例子中的最后一个列表时，结果如下：

```
sage: CRT_basis([29,31,37,41,43])
[32354576, 20808689, 23774055, 17163708, 23184311]
```

Sage 提供的最后一个 CRT 函数是 CRT_vectors。该函数对数个不同的列表（每个列表的模相同）执行 CRT_list，并返回答案。这使得在使用相同的 CRT_basis 时更加高效，因为不需要对每个列表重复预计算。例如

```
sage:
CRT_vectors([[1,10],[2,11],[3,12],[4,13],[5,14]],
[29,31,37,41,43])
[36657161, 36657170]
```

例 5 使用 Sage 内建函数进行模幂运算。

Sage 能够使用高效的算法执行模幂运算（如平方和乘法），且不会增大中间运算量。该功能通过 IntegerModRing 对象实现。创建 IntegerModRing 对象意味着运算是模算术，然后将整数映射到环上，使得运算都是取模运算。这样环上的所有元素的幂运算都可以被快速执行。例如

```
sage: R = IntegerModRing(101)
sage: x = R(10)
sage: x^99
91

sage: R = IntegerModRing(1024)
sage: x = R(111)
sage: x^345
751

sage: x = R(100)
sage: x^200
0

sage: N = 127*101
sage: R = IntegerModRing(N)
sage: x = R(54)
sage: x^95
9177
```

创建 IntegerModRing 对象同使用 GF(...)创建 FiniteField 对象类似，唯一区别是模数可以是普通的合数。

例 6 使用 Sage 内建函数进行欧拉函数运算。

Sage 已经内建了执行欧拉函数计算的函数。因为通常使用希腊字母 phi 来表示这个函数，所以该函数称为 euler_phi 函数。这个函数的使用非常简单，只需要传入一个整数，然后它就可以计算出结果。因为这个函数需要将输入因式分解，所以它的计算复杂度是指数级的。

```
sage: euler_phi(101)
100

sage: euler_phi(1024)
512

sage: euler_phi(333)
216
```

```
sage: euler_phi(125)
100
sage: euler_phi(423)
276
```

B.3 第 3 章 传统加密技术

下面是古典密码的例子和习题：

```
en_alphabet = "abcdefghijklmnopqrstuvwxyz"

#
# This function returns true if and only if the character
c is an
# alphabetic character
#
def is_alphabetic_char(c):
    return (c.lower() in en_alphabet)

#
# This function converts a single character into its
numeric value
#
def char_to_num(c):
    return en_alphabet.index(c.lower())

#
# This function returns the character corresponding to x
mod 26
# in the English alphabet
#
def num_to_char(x):
    return en_alphabet[x % 26]
```

例 1 对于密钥（整数 0，1，2，…，25）和字符串，用 Sage 实现加密/解密函数。加解密函数只用对于字符'a'，'b'，…，'z'（都是大写或都是小写）操作，对其他字符不做改变。

解答：

```
def CaesarEncrypt(k, plaintext):
    ciphertext = ""
    for j in xrange(len(plaintext)):
        p = plaintext[j]
        if is_alphabetic_char(p):
            x = (k + char_to_num(p)) % 26
            c = num_to_char(x)
        else:
            c = p
        ciphertext += c
    return ciphertext

def CaesarDecrypt(k, ciphertext):
    plaintext = ""
    for j in xrange(len(ciphertext)):
        c = ciphertext[j]
```

```
        if is_alphabetic_char(c):
            x = (char_to_num(c) - k) % 26
            p = num_to_char(x)
        else:
            p = c
        plaintext += p
    return plaintext
```

例 2 实现对于密文进行穷举攻击的程序，要将密钥和对应的解密结果打印出来。而且要求能够提供一个可选参数，该可选参数将一个子串作为输入，如果提供了这个可选参数，那么程序将只打印出包含该输入串的解密结果。

解答：

```
def BruteForceAttack(ciphertext, keyword=None):
    for k in xrange(26):
        plaintext = CaesarDecrypt(k, ciphertext)
        if (None==keyword) or (keyword in plaintext):
            print "key", k, "decryption", plaintext
    return
```

例 3 根据如下输入（密钥，明文）对，给出例 1 中加密函数的输出。

- k = 16 明文= "Get me a vanilla ice cream, make it a double."
- k = 15 明文= "I don't much care for Leonard Cohen."
- k = 16 明文= "I like root beer floats."

解答：

```
sage: k = 6; plaintext = 'Get me a vanilla ice
cream, make it a double.'
sage: CaesarEncrypt(k, plaintext)
'mkz sk g bgtorrg oik ixkgs, sgqk oz g juahrk.'

sage: k = 15; plaintext = "I don't much care for
Leonard Cohen."
sage: CaesarEncrypt(k, plaintext)
"x sdc'i bjrw rpgt udg atdcpgs rdwtc."

sage: k = 16; plaintext = "I like root beer floats."
sage: CaesarEncrypt(k, plaintext)
'y byau heej ruuh vbeqji.'
```

例 4 根据如下输入（密钥，密文）对，给出例 1 中解密函数的输出。

- k = 12 密文= 'nduzs ftq buzq oazqe.'
- k = 3 密文= "fdhvdu qhhgv wr orvh zhljkw."
- k = 20 密文= "ufgihxm uly numnys."

解答：

```
sage: k = 12; ciphertext = "nduzs ftq buzq oazqe."
sage: CaesarDecrypt(k, ciphertext)
'bring the pine cones.'

sage: k = 3; ciphertext = "fdhvdu qhhgv wr orvh
zhljkw."
sage: CaesarDecrypt(k, ciphertext)
'caesar needs to lose weight.'
```

```
sage: k = 20; ciphertext = "ufgihxm uly numnys."
sage: CaesarDecrypt(k, ciphertext)
'almonds are tastey.'
```

例 5　对于如下密文，给出例 4 中穷举攻击程序的输出。如果指定了可选参数中的密钥字，就将其传递给攻击程序。

- 密文= 'gryy gurz gb tb gb nzoebfr puncry.' 密钥字 = 'chapel'
- 密文= 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.' 密钥字 = 'cymbal'
- 密文= 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.' 无密钥字

解答：

```
sage: ciphertext = 'gryy gurz gb tb gb nzoebfr puncry.'
sage: BruteForceAttack(ciphertext, 'chapel')
key 13 decryption tell them to go to ambrose chapel.

sage: ciphertext = 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.'
sage: BruteForceAttack(ciphertext, 'cymbal')
key 17 decryption fire the shot when the cymbals crash.

sage: ciphertext = 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.'
sage: BruteForceAttack(ciphertext)
key 0 decryption baeeq klwosjl osk s esf ozg cfwo lgg emuz.
key 1 decryption azddp jkvnrik nrj r dre nyf bevn kff dlty.
key 2 decryption zycco ijumqhj mqi q cqd mxe adum jee cksx.
key 3 decryption yxbbn hitlpgi lph p bpc lwd zctl idd bjrw.
key 4 decryption xwaam ghskofh kog o aob kvc ybsk hcc aiqv.
key 5 decryption wvzzl fgrjneg jnf n zna jub xarj gbb zhpu.
key 6 decryption vuyyk efqimdf ime m ymz ita wzqi faa ygot.
key 7 decryption utxxj dephlce hld l xly hsz vyph ezz xfns.
key 8 decryption tswwi cdogkbd gkc k wkx gry uxog dyy wemr.
key 9 decryption srvvh bcnfjac fjb j vjw fqx twnf cxx vdlq.
key 10 decryption rquug abmeizb eia i uiv epw svme bww uckp.
key 11 decryption qpttf zaldhya dhz h thu dov ruld avv tbjo.
key 12 decryption posse yzkcgxz cgy g sgt cnu qtkc zuu sain.
key 13 decryption onrrd xyjbfwy bfx f rfs bmt psjb ytt rzhm.
key 14 decryption nmqqc wxiaevx aew e qer als oria xss qygl.
key 15 decryption mlppb vwhzduw zdv d pdq zkr nqhz wrr pxfk.
key 16 decryption lkooa uvgyctv ycu c ocp yjq mpgy vqq owej.
key 17 decryption kjnnz tufxbsu xbt b nbo xip lofx upp nvdi.
key 18 decryption jimmy stewart was a man who knew too much.
key 19 decryption ihllx rsdvzqs vzr z lzm vgn jmdv snn ltbg.
key 20 decryption hgkkw qrcuypr uyq y kyl ufm ilcu rmm ksaf.
key 21 decryption gfjjv pqbtxoq txp x jxk tel hkbt qll jrze.
key 22 decryption feiiu opaswnp swo w iwj sdk gjas pkk iqyd.
key 23 decryption edhht nozrvmo rvn v hvi rcj fizr ojj hpxc.
key 24 decryption dcggs mnyquln qum u guh qbi ehyq nii gowb.
key 25 decryption cbffr lmxptkm ptl t ftg pah dgxp mhh fnva.
```

B.4　第 4 章　分组密码和数据加密标准

例 1　在本例中，需要实现在线附录 G 中提到的 DES 的简化版本 SDES。

```
#
# The Expansions/Permutations are stored as lists of
bit positions
#
```

```
P10_data = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6];
P8_data = [6, 3, 7, 4, 8, 5, 10, 9];
LS1_data = [2, 3, 4, 5, 1];
LS2_data = [3, 4, 5, 1, 2];
IP_data = [2, 6, 3, 1, 4, 8, 5, 7];
IPinv_data = [4, 1, 3, 5, 7, 2, 8, 6];
EP_data = [4, 1, 2, 3, 2, 3, 4, 1];
P4_data = [2, 4, 3, 1];
SW_data = [5, 6, 7, 8, 1, 2, 3, 4];
#
# SDES lookup tables
#
S0_data = [[1, 0, 3, 2],
           [3, 2, 1, 0],
           [0, 2, 1, 3],
           [3, 1, 3, 2]];
S1_data = [[0, 1, 2, 3],
           [2, 0, 1, 3],
           [3, 0, 1, 0],
           [2, 1, 0, 3]];
def ApplyPermutation(X, permutation):
    r"""
    This function takes a permutation list (list of
    bit positions.)
    And outputs a bit list with the bits taken from X.
    """
    # permute the list X
    l = len(permutation);
    return [X[permutation[j]-1] for j in xrange(l)];
def ApplySBox(X, SBox):
    r"""
    This function Applies the SDES SBox (by table
    look up
    """
    r = 2*X[0] + X[3];
    c = 2*X[1] + X[2];
    o = SBox[r][c];
    return [o & 2, o & 1];
#
# Each of these functions uses ApplyPermutation
# and a permutation list to perform an SDES
# Expansion/Permutation
#
def P10(X):
    return ApplyPermutation(X, P10_data);
def P8(X):
    return ApplyPermutation(X, P8_data);
```

```
def IP(X):
    return ApplyPermutation(X, IP_data);

def IPinv(X):
    return ApplyPermutation(X, IPinv_data);

def EP(X):
    return ApplyPermutation(X, EP_data);

def P4(X):
    return ApplyPermutation(X, P4_data);

def SW(X):
    return ApplyPermutation(X, SW_data);

def LS1(X):
    return ApplyPermutation(X, LS1_data);

def LS2(X):
    return ApplyPermutation(X, LS2_data);

#
# These two functions perform the SBox substitutions
#

def S0(X):
    return ApplySBox(X, S0_data);

def S1(X):
    return ApplySBox(X, S1_data);

def concatenate(left, right):
    r"""
    Joins to bit lists together.
    """
    ret = [left[j] for j in xrange(len(left))];
    ret.extend(right);
    return ret;

def LeftHalfBits(block):
    r"""
    Returns the left half bits from block.
    """
    l = len(block);
    return [block[j] for j in xrange(l/2)];

def RightHalfBits(block):
    r"""
    Returns the right half bits from block.
    """
    l = len(block);
    return [block[j] for j in xrange(l/2, l)];

def XorBlock(block1, block2):
    r"""
    Xors two blocks together.
    """
    l = len(block1);
    if (l != len(block2)):
        raise ValueError, "XorBlock arguments must
        be same length"
```

```
    return [(block1[j]+block2[j]) % 2 for j in
    xrange(l)];

def SDESKeySchedule(K):
    r"""
    Expands an SDES Key (bit list) into the two
    round keys.
    """
    temp_K = P10(K);

    left_temp_K = LeftHalfBits(temp_K);
    right_temp_K = RightHalfBits(temp_K);

    K1left = LS1(left_temp_K);
    K1right = LS1(right_temp_K);

    K1temp = concatenate(K1left, K1right);
    K1 = P8(K1temp);

    K2left = LS2(K1left);
    K2right = LS2(K1right);

    K2temp = concatenate(K2left, K2right);

    K2 = P8(K2temp);

    return (K1, K2);

def f_K(block, K):
    r"""
    Performs the f_K function supplied block and K.
    """
    left_block = LeftHalfBits(block);
    right_block = RightHalfBits(block);

    temp_block1 = EP(right_block);

    temp_block2 = XorBlock(temp_block1, K);

    left_temp_block2 = LeftHalfBits(temp_block2);
    right_temp_block2 = RightHalfBits(temp_block2);

    S0_out = S0(left_temp_block2);
    S1_out = S1(right_temp_block2);

    temp_block3 = concatenate(S0_out, S1_out);

    temp_block4 = P4(temp_block3)

    temp_block5 = XorBlock(temp_block4, left_block);

    output_block = concatenate(temp_block5, right_
    block)

    return output_block;

def SDESEncrypt(plaintext_block, K):
    r"""
    Performs a single SDES plaintext block encryption.
    (Given plaintext and key as bit lists.)
    """

    (K1, K2) = SDESKeySchedule(K);

    temp_block1 = IP(plaintext_block);
```

```
    temp_block2 = f_K(temp_block1, K1);
    temp_block3 = SW(temp_block2);
    temp_block4 = f_K(temp_block3, K2);
    output_block = IPinv(temp_block4);
    return output_block;
```

B.5 第 5 章 有限域

例 1 求最大公约数的欧几里得算法

```
def EUCLID(a,b):
    r"""
    The Euclidean algorithm for finding the gcd of a and b.
    This algorithm assumes that a > b => 0
    INPUT:
            a - positive integer
            b - nonnegative integer less than a
OUTPUT:
    g - greatest common divisor of a and b
    """
    if (b < 0) or ( a <= b):
        raise ValueError, "Expected 0 < a < b"
    (A, B) = (a,b);
    while (True):
        if (0 == B):
            return A;
        R = A % B;
        A = B;
        B = R;
```

例 2 求最大公约数的扩展欧几里得算法

```
def EXTENDED_EUCLID(m,b):
r"""
The extended Euclidean algorithm to find gcd(m,b).
The input is expected to be such that 0 <= b < m.
INPUT:
    m - positive integer
    b - nonnegative integer less than m
OUTPUT:
(g, b_inv) - g is the gcd of m and b, b_inv is
the multiplicative inverse of b mod m.
"""
if (m < b) or (b < 0):
    raise ValueError, "Expected input (0 < b < m)"
(A1,A2,A3) = (1,0,m);
```

```
    (B1,B2,B3) = (0,1,b);
    while (True):
        if (0 == B3):
            return (A3, None)
        if (1 == B3):
            return (B3, B2)
        Q = floor(A3/B3)
        (T1,T2,T3) = (A1-Q*B1, A2-Q*B2, A3-Q*B3)
    (A1, A2, A3) = (B1, B2, B3)
    (B1, B2, B3) = (T1, T2, T3)
```

例3 求多项式（系数为域上元素）gcd的欧几里得算法

```
def POLYNOMIAL_EUCLID(A, B):
    r"""
    Euclidian algorithm for polynomial GCD:
    Given two polynomials over the same base field,
    Assuming degree(A) => degree(B) => 0.
    INPUT:
        A - polynomial over a field.
        B - polynomial over the same field as A, and 0 <=
        degree(B) <= degree(A).
    OUTPUT:
        G - greatest common divisor of A and B.
    """
    degA = A.degree();
    degB = B.degree();
    if ((degB < 0) or (degA < degB)):
        raise ValueError, "Expected 0 <= degree(B) <=
        degree(A)"
    while(True):
        if (0 == B):
            return A;
        R = A % B;
        A = B;
        B = R;
```

例4 求两个多项式（系数为同一域上的元素）的gcd的扩展欧几里得算法

```
def POLYNOMIAL_EXTENDED_EUCLID(m, b):
    r"""
    Extended Euclidian algorithm for polynomial GCD:
    Given two polynomials over the same base field,
    Assuming degree(m) => degree(b) => 0
    INPUT:
        m - polynomial over a field.
    b - polynomial over the same field as A, and 0 <=
    degree(B) <= degree(M).
```

```
    OUTPUT:

        (g,b_inv) - the pair where:

        g - greatest common divisor of m and b.

        m_inv - is None if G is not of degree 0, and
                otherwise it is the polynomial such that
                b(X)*b_inv(X) = 1 mod m(X)

    """
    degm = m.degree();
    degb = b.degree();

    if(degb < 0) or (degm < degb):
        raise ValueError, "expected 0 <= degree(b) <=
        degree(m)"

    (A1, A2, A3) = (1, 0, m);
    (B1, B2, B3) = (0, 1, b);

    while (True):

        if (0 == B3):
            return (A3, None);

        if (0 == B3.degree()):
            return (B3/B3, B2/B3);

        Q = A3.quo_rem(B3)[0];

        (T1, T2, T3) = (A1 - Q*B1, A2 - Q*B2, A3 - Q*B3);
        (A1, A2, A3) = (B1, B2, B3);
        (B1, B2, B3) = (T1, T2, T3);
```

例 5 Sage 已经将 gcd 建成了函数库。通常的求最大公约数函数可直接调用

```
sage: gcd(15,100)
5

sage: gcd(90,65311)
1
```

对于整数对象还可以用对象方法的方式调用

```
sage: x = 10456890
sage: x.gcd(100)
10
```

Sage 的函数库还包括求最大公约数的扩展欧几里得算法。调用函数 xgcd(a, b)将返回一个三元组，其中，第一个元素是 gcd，第二个和第三个元素满足 gcd(a, b) = u*a + v*b 的系数 u，v。可如下所示进行调用：

```
sage: xgcd(17,31)
(1, 11, -6)
sage: xgcd(10, 115)
(5, -11, 1)
```

对于整数对象还可以用对象方法的方式调用

```
sage: x = 300
sage: x.xgcd(36)
(12, 1, -8)
```

例 6 Sage 提供对于有限域和有限域算术运算的鲁棒性支持。如果要初始化一个素数阶的有限域，可以使用 GF 命令把阶作为参数传递。

```
sage: F = GF(2)
sage: F
Finite Field of size 2

sage: F = GF(37)
sage: F
Finite Field of size 37

sage: p = 95131
sage: K = GF(p)
sage: K
Finite Field of size 95131
```

如果要初始化一个阶为素数的幂的有限域，可以使用 GF 命令及如下符号定义（扩域中单位元的迹）

```
sage: F.<a> = GF(128)
sage: F
Finite Field in a of size 2^7
```

有限域中的运算通过如下符号定义：

```
sage: K = GF(37)
sage: a = K(3)
sage: b = K(18)
sage: a - b
22
sage: a + b
21
sage: a * b
17
sage: a/b
31
sage: a^-1
25
sage: 1/a
25
```

在素数幂扩域中进行运算，可通过如下定义和操作有限域的元素：

```
sage: F.<a> = GF(128)
sage: b = a^2 + 1
sage: c = a^5 + a^3 + 1
sage: b - c
a^5 + a^3 + a^2
sage: b + c
a^5 + a^3 + a^2
sage: b*c
a^3 + a^2 + a
sage: b/c
a^5 + a^3 + a^2 + a
sage: b^-1
a^5 + a^3 + a
sage: 1/b
a^5 + a^3 + a
```

例7 在Sage中可以直接创建有限域上的多项式环。可通过如下方法创建有限域上的多项式环：

```
sage: R.<x> = GF(2)[]
sage: R
Univariate Polynomial Ring in x over Finite Field of
```

```
size 2 (using NTL)
sage: R.<x> = GF(101)[]

sage: R
sage: R.<x> = F[]
sage: R
Univariate Polynomial Ring in x over Finite Field in
a of size 2^7
```

初始化多项式环后，可按如下示例执行各种运算：

```
sage: R.<x> = GF(2)[]
sage: f = x^3 + x + 1
sage: g = x^5 + x
sage: f + g
x^5 + x^3 + 1
sage: f*g
x^8 + x^6 + x^5 + x^4 + x^2 + x
```

函数 quo_rem 执行除法运算

```
sage: g.quo_rem(f)
(x^2 + 1, x^2 + 1)
```

同样可以计算最大公因式

```
sage: f.gcd(g)
1

sage: g.gcd(g^2)
x^5 + x

sage: R.<x> = GF(17)[]
sage: f = 3*x^3 + 2*x^2 + x
sage: g = x^2 + 5
sage: f - g
3*x^3 + x^2 + x + 12
sage: f * g
3*x^5 + 2*x^4 + 16*x^3 + 10*x^2 + 5*x
sage: f.quo_rem(g)
(3*x + 2, 3*x + 7)
```

在该多项式环上计算 gcd

```
sage: f.gcd(g)
1

sage: f.gcd(x^2 + x)
x
```

当创建一个素数幂扩域时，Sage 将找出一个不可约多项式。例如

```
sage: F.<a> = GF(32)
a^5 + a^2 + 1
```

然而 GF(2)上的 5 次不可约多项式不止一个，例如 x^5+x^3+1。假设想要任选不可约多项式来建立二元域的 5 次扩域，可通过如下进行：

```
sage: R.<x> = GF(2)[]
sage: F = GF(2).extension(x^5 + x^3 + 1, 'a')
sage: a = F.gen()
```

这时需要在最后一步选择生成元。因为使用 GF 函数创建扩域时会自动选定生成元，但指定扩域时生成元不会自动选定。

B.6 第 6 章 高级加密标准

例 1 简化版本的 AES

```
#
# These structures are the underlying
# Galois Field and corresponding Vector Space
# of the field used in the SAES algorithm
# These structures allow us to easily compute with these fields.
#
F = GF(2);
L.<a> = GF(2^4);
V = L.vector_space();
VF8 = VectorSpace(F, 8);
#
# The MixColumns and its Inverse matrices are stored
# as 2x2 matrices with elements in GF(2^4) (as are state
matrices.)
# The MixColumns operation (and its inverse) are performed by
# matrix multiplication.
#
MixColumns_matrix = Matrix(L, [[1,a^2],[a^2,1]]);
InverseMixColumns_matrix = MixColumns_matrix.inverse();
SBox_matrix = Matrix(L,
  [
  [       1 + a^3,          a^2,          a + a^3,1 + a + a^3],
  [ 1 + a^2 + a^3,            1,              a^3,    1 + a^2],
  [       a + a^2,            0,                a,      1 + a],
  [      a^2 + a^3, a + a^2 + a^3, 1 + a + a^2 + a^3, 1 + a + a^2]
  ]);
InverseSBox_matrix = Matrix(L,
  [
  [   a + a^3,     1 + a^2,       1 + a^3,       1 + a + a^3],
  [         1, 1 + a + a^2,           a^3, 1 + a + a^2 + a^3],
  [   a + a^2,           0,             a,             1 + a],
  [ a^2 + a^3,         a^2, 1 + a^2 + a^3,     a + a^2 + a^3]
  ]);
RCON = [
  VF8([F(0), F(0), F(0), F(0), F(0), F(0), F(0), F(1)]),
  VF8([F(0), F(0), F(0), F(0), F(1), F(1), F(0), F(0)])
    ];
def SAES_ToStateMatrix(block):
    r"""
    Converts a bit list into an SAES state matrix.
    """
    B = block;
    # form the plaintext block into a matrix of GF(2^n)
    elements
    S00 = L(V([B[0], B[1], B[2], B[3]]));
```

```
    S01 = L(V([B[4], B[5], B[6], B[7]]));
    S10 = L(V([B[8], B[9], B[10], B[11]]));
    S11 = L(V([B[12], B[13], B[14], B[15]]));

    state_matrix = Matrix(L, [[S00,S01],[S10,S11]]);

    return state_matrix;

def SAES_FromStateMatrix(State Matrix):
    r"""
    Converts an SAES State Matrix to a bit list.
    """

    output = [];

    # convert State Matrix back into bit list
    for r in xrange(2):
        for c in xrange(2):
            v = V(State Matrix[r,c]);
            for j in xrange(4):
                output.append(Integer(v[j]));

    return output;

def SAES_AddRoundKey(state_matrix, K):
    r"""
    Adds a round key to an SAES state matrix.
    """

    K_matrix = SAES_ToStateMatrix(K);

    next_state_matrix = K_matrix + state_matrix;

    return next_state_matrix;

def SAES_MixColumns(state_matrix):
    r"""
    Performs the Mix Columns operation.
    """

    next_state_matrix = MixColumns_matrix*state_matrix;

    return next_state_matrix;
def SAES_InverseMixColumns(state_matrix):
    r"""
    Performs the Inverse Mix Columns operation.
    """

    next_state_matrix = InverseMixColumns_matrix*
    state_matrix;
    return next_state_matrix;

def SAES_ShiftRow(state_matrix):
    r"""
    Performs the Shift Row operation.
    """

    M = state_matrix;
    next_state_matrix = Matrix(L, [
                                   [M[0,0], M[0,1]],
                                   [M[1,1], M[1,0]]
                                   ]);
    return next_state_matrix;
```

```
def SAES_SBox(nibble):
    r"""
    Performs the SAES SBox look up in the SBox matrix
    (lookup table.)
    """

    v = nibble._vector_();
    c = Integer(v[0]) + 2*Integer(v[1]);
    r = Integer(v[2]) + 2*Integer(v[3]);
    return SBox_matrix[r,c];

def SAES_NibbleSubstitution(state_matrix):
    r"""
    Performs the SAES SBox on each element of an SAES
    state matrix.
    """

    M = state_matrix;
    next_state_matrix = Matrix(L,
            [ [ SAES_SBox(M[0,0]), SAES_SBox(M[0,1])],
              [ SAES_SBox(M[1,0]), SAES_SBox(M[1,1])] ]);
    return next_state_matrix;

def SAES_InvSBox(nibble):
    r"""
    Performs the SAES Inverse SBox look up in the SBox
    matrix (lookup table.)
    """

    v = nibble._vector_();
    c = Integer(v[0]) + 2*Integer(v[1]);
    r = Integer(v[2]) + 2*Integer(v[3]);
    return InverseSBox_matrix[r,c];

def SAES_InvNibbleSub(state_matrix):
    r"""
    Performs the SAES Inverse SBox on each element of an
    SAES state matrix.
    """

    M = state_matrix;
    next_state_matrix = Matrix(L,
    [ [ SAES_InvSBox(M[0,0]), SAES_InvSBox(M[0,1])],
      [ SAES_InvSBox(M[1,0]), SAES_InvSBox(M[1,1])] ]);
    return next_state_matrix;

def RotNib(w):
    r"""
    Splits an 8 bit list into two elements of GF(2^4)
    """
    N_0 = L(V([w[j] for j in xrange(4)]));
    N_1 = L(V([w[j] for j in xrange(4,8)]));
    return (N_1, N_0);

def SAES_g(w, i):
    r"""
    Performs the SAES g function on the 8 bit list w.
    """
    (N0, N1) = RotNib(w);
```

```
    N0 = V(SAES_SBox(N0));
    N1 = V(SAES_SBox(N1));
    temp1 = VF8( [ N0[0], N0[1], N0[2], N0[3],
                   N1[0], N1[1], N1[2], N1[3] ] );
    output = temp1 + RCON[i];
    return output;

def SAES_KeyExpansion(K):
    r"""
    Expands an SAES key into two round keys.
    """
    w0 = VF8([K[j] for j in xrange(8)]);
    w1 = VF8([K[j] for j in xrange(8,16)]);
    w2 = w0 + SAES_g(w1, 0);
    w3 = w1 + w2;
    w4 = w2 + SAES_g(w3, 1);
    w5 = w3 + w4;
    K0 = [w0[j] for j in xrange(8)];
    K0.extend([w1[j] for j in xrange(8)]);
    K1 = [w2[j] for j in xrange(8)];
    K1.extend([w3[j] for j in xrange(8)]);
    K2 = [w4[j] for j in xrange(8)];
    K2.extend([w4[j] for j in xrange(8)]);

    return (K0, K1, K2);

#
# Encrypts one plaintext block with key K
#

def SAES_Encrypt(plaintext, K):
    r"""
    Performs a SAES encryption on a single plaintext
    block.
    (Both block and key passed as bit lists.)
    """

    # get the key schedule
    (K0, K1, K2) = SAES_KeyExpansion(K);

    state_matrix0 = SAES_ToStateMatrix(plaintext);

    state_matrix1 = SAES_AddRoundKey(state_matrix0, K0);

    state_matrix2 = SAES_NibbleSubstitution
    (state_matrix1);

    state_matrix3 = SAES_ShiftRow(state_matrix2);

    state_matrix4 = SAES_MixColumns(state_matrix3);

    state_matrix5 = SAES_AddRoundKey(state_matrix4, K1);

    state_matrix6 = SAES_NibbleSubstitution
    (state_matrix5);

    state_matrix7 = SAES_ShiftRow(state_matrix6);

    state_matrix8 = SAES_AddRoundKey(state_matrix7, K2);

    output = SAES_FromStateMatrix(state_matrix8);
    return output;
```

```
#
# Decrypts one ciphertext block with key K
#
def SAES_Decrypt(ciphertext, K):
    r"""
    Performs a single SAES decryption operation on a -
    ciphertext block.
    (Both block and key passed as bit lists.)
    """

    # perform key expansion
    (K0, K1, K2) = SAES_KeyExpansion(K);
# form the ciphertext block into a matrix of GF(2^n)
elements
state_matrix0 = SAES_ToStateMatrix(ciphertext);
state_matrix1 = SAES_AddRoundKey(state_matrix0, K2);
state_matrix2 = SAES_ShiftRow(state_matrix1);
state_matrix3 = SAES_InvNibbleSub(state_matrix2);
state_matrix4 = SAES_AddRoundKey(state_matrix3, K1);
state_matrix5 = SAES_InverseMixColumns
(state_matrix4);
state_matrix6 = SAES_ShiftRow(state_matrix5);
state_matrix7 = SAES_InvNibbleSub(state_matrix6);
state_matrix8 = SAES_AddRoundKey(state_matrix7, K0);
output = SAES_FromStateMatrix(state_matrix8);
return output;
```

B.7 第 8 章 伪随机数的产生和流密码

例 1 BBS（Blum Blum Shub）伪随机数发生器

```
def BlumBlumShub_Initialize(bitlen, seed):
    r"""
    Initializes a Blum-Blum-Shub RNG State.
    A BBS-RNG State is a list with two elements:
    [N, X]
    N is a 2*bitlen modulus (product of two primes)
    X is the current state of the PRNG.
    INPUT:
        bitlen - the bit length of each of the prime
        factors of n
        seed - a large random integer to start out the
        prng
    OUTPUT:
        state - a BBS-RNG internal state
```

```
    """

    # note that this is not the most cryptographically
    secure
    # way to generate primes, because we do not know how the
    # internal sage random_prime function works.

    p = 3;
    while (p < 2^(bitlen-1)) or (3 != (p % 4)):
        p = random_prime(2^bitlen);

    q = 3;
    while (q < 2^(bitlen-1)) or (3 != (q % 4)):
        q = random_prime(2^bitlen);

    N = p*q;

    X = (seed^2 % N)

    state = [N, X]

    return state;

def BlumBlumShub_Generate(num_bits, state):
    r"""
    Blum-Blum-Shum random number generation function.

    INPUT:

        num_bits - the number of bits (iterations) to
        generate with this RNG.

        state - an internal state of the BBS-RNG (a list
        [N, X].)

    OUTPUT:

        random_bits - a num_bits length list of random
        bits.

    """

    random_bits = [];

    N = state[0]
    X = state[1]

    for j in xrange(num_bits):

        X = X^2 % N
        random_bits.append(X % 2)

    # update the internal state
    state[1] = X;

    return random_bits;
```

例 2　线性同余伪随机数发生器

```
def LinearCongruential_Initialize(a, c, m, X0):
    r"""

    This functional initializes a linear congruential
    RNG state.

    This state is a list of four integers: [a, c, m, X]
```

```
    a,c,m are the parameters of the linear congruential
    instantiation X is the current state of the PRNG.

    INPUT:

        a - The coefficient
        c - The offset
        m - The modulus
        X0 - The initial state

    OUTPUT:

        state - The initial internal state of the RNG

    """

    return [a,c,m,X0]

def LinearCongruential_Generate(state):
    r"""

    Generates a single linear congruential RNG output
    and updates the state.

    INPUT:

        state - an internal RNG state.

    OUTPUT:

        X - a single output of the linear congruential RNG.

    """
    a = state[0]
    c = state[1]
    m = state[2]
    X = state[3]
    X_next = (a*X + c) % m
    state[3] = X_next
    return X_next
```

B.8 第 9 章 公钥密码学与 RSA

例 1 使用 Sage 我们能够模拟执行 RSA 的加密和解密过程

```
sage: # randomly select some prime numbers
sage: p = random_prime(1000); p
191
sage: q = random_prime(1000); q
601
sage: # compute the modulus
sage: N = p*q
sage: R = IntegerModRing(N)
sage: phi_N = (p-1)*(q-1)
sage: # we can choose the encrypt key to be anything
sage: # relatively prime to phi_N
sage: e = 17
sage: gcd(d, phi_N)
1
sage: # the decrypt key is the multiplicative
inverse
```

```
sage: # of d mod phi_N
sage: d = xgcd(d, phi_N)[1] % phi_N
sage: d
60353
sage: # Now we will encrypt/decrypt some random 7
digit numbers
sage: P = randint(1,127); P
97
sage: # encrypt
sage: C = R(P)^e; C
46685
sage: # decrypt
sage: R(C)^d
97
sage: P = randint(1,127); P
46
sage: # encrypt
sage: C = R(P)^e; C
75843
sage: # decrypt
sage: R(C)^d
46
sage: P = randint(1,127); P
3
sage: # encrypt
sage: C = R(P)^e; C
288
sage: # decrypt
sage: R(C)^d
3
```

Sage 还能够容易地对更大的数进行运算

```
sage: p = random_prime(1000000000); p
114750751
sage: q = random_prime(1000000000); q
8916569
sage: N = p*q
sage: R = IntegerModRing(N)
sage: phi_N = (p-1)*(q-1)
sage: e = 2^16 + 1
sage: d = xgcd(e, phi_N)[1] % phi_N
sage: d
237150735093473
sage: P = randint(1,1000000); P
955802
sage: C = R(P)^e
sage: R(C)^d
955802
```

例 2 我们可以使用 Sage 进行 RSA 签名/验证

```
sage: p = random_prime(10000); p
1601
sage: q = random_prime(10000); q
4073
```

```
sage: N = p*q
sage: R = IntegerModRing(N)
sage: phi_N = (p-1)*(q-1)
sage: e = 47
sage: gcd(e, phi_N)
1
sage: d = xgcd(e,phi_N)[1] % phi_N
sage: # Now by exponentiating with the private key
sage: # we are effectively signing the data
sage: # a few examples of this
sage: to_sign = randint(2,2^10); to_sign
650
sage: # the signature is checked by exponentiating
sage: # and checking vs the to_sign value
sage: signed = R(to_sign)^d; signed
2910116
sage: to_sign == signed^e
True
sage: to_sign = randint(2,2^10); to_sign
362
sage: signed = R(to_sign)^d; signed
546132
sage: to_sign == signed^e
True
sage: # we can also see what happens if we try to
verify a bad signature
sage: to_sign = randint(2,2^10); to_sign
605
sage: signed = R(to_sign)^d; signed
1967793
sage: bad_signature = signed - randint(2,100)
sage: to_sign == bad_signature^e
False
```

B.9 第 10 章 密钥管理和其他公钥密码体制

例 1 下面是 Sage 中 Alice 和 Bob 执行 Diffie-Hellman 密钥交换的例子：

```
sage: # Alice and Bob agree on the domain
parameters:
sage: p = 619
sage: F = GF(p)
sage: g = F(2)
sage: # Alice picks a random value x in 1 . . . 618
sage: x = randint(1,618); x
571
sage: # Alice computes X = g^x and sends this to Bob
sage: X = g^571; X
591
sage: # Bob picks a random value y in 1 . . . 618
sage: y = randint(1,618);y
356
sage: # Bob computes Y = g^y and sends this to Alice
```

```
sage: Y = g^y; Y
199
sage: # Alice computes Y^x
sage: Y^x
563
sage: # Bob computes X^y
sage: X^y
563
sage: # Alice and Bob now share a secret value
```

例 2 为了防止小子群攻击，素数 p 的选择应满足 $p=2q+1$，其中 p 也是素数。

```
sage: q = 761
sage: p = 2*q + 1
sage: is_prime(q)
True
sage: is_prime(p)
True
sage: F = GF(p)
sage: g = F(3)
sage: g^q
1
sage: # note that g^q = 1 implies g is of order q
sage: # Alice picks a random value x in 2 . . . q-1
sage: x = randint(2,q-1); x
312
sage: # Alice computes X = g^x and sends it to Bob
sage: X = g^x; X
26
sage: # Bob computes a random value y in 2 . . . q-1
sage: y = randint(2,q-1); y
24
sage: # Bob computes Y = g^y and sends it to Alice
sage: Y = g^y; Y
1304
sage: # Alice computes Y^x
sage: Y^x
541
sage: # Bob computes X^y
sage: X^y
541
sage: # Alice and Bob now share the secret value 541
```

例 3 Sage 对于椭圆曲线的相关运算提供了大量的支持。因为这些功能可以使得计算过程简化，对于学习过程非常有用；如果直接手工计算这些例子则非常烦琐。首先通过指定所在的域，以及 Weierstrass 方程的系数，来选定椭圆曲线。为此我们把 Weierstrass 方程表示为

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Sage 函数 `EllipticCurve(R, [a1, a2, a3, a4, a6])`在环 `R` 上建立椭圆曲线

```
sage: E = EllipticCurve(GF(17), [1,2,3,4,5])
sage: E
Elliptic Curve defined by y^2 + x*y + 3*y = x^3 +
2*x^2 + 4*x + 5 over Finite Field of size 17
sage: E = EllipticCurve(GF(29), [0,0,0,1,1])
sage: E
```

```
Elliptic Curve defined by y^2 = x^3 + x + 1 over
Finite Field of size 29
sage: E = EllipticCurve(GF(127), [0,0,0,2,17])
sage: E
Elliptic Curve defined by y^2 = x^3 + 2*x + 17 over
Finite Field of size 127

sage: F.<theta> = GF(2^10)
sage: E = EllipticCurve(F, [1,0,0,1,0])
sage: E
Elliptic Curve defined by y^2 + x*y = x^3 + x over
Finite Field in theta of size 2^10
```

例 4 Koblitz 曲线。Koblitz 曲线是在二元域上由下式定义的椭圆曲线

$$y^2 + xy = x^3 + ax^2 + 1$$

其中 $a = 0$ 或 1。FIPS 186-3 推荐了一系列 Koblitz 曲线用于数字签名标准（DSS）。这里我们给出 Koblitz 曲线的例子

```
sage: F.<theta> = GF(2^17)
sage: E = EllipticCurve(F,[1,0,0,theta,1])
sage: E
Elliptic Curve defined by y^2 + y = x^3 + theta*
x^2 = 1 over Finite Field in theta of size 2^17
```

例 5 Sage 还能够指定曲线的密码学安全规模，如 FIPS 186-3 曲线之一 K163。

```
sage: F.<theta> = GF(2^163)
sage: E = EllipticCurve(F, [1,0,0,1,1])
sage: E
Elliptic Curve defined by y^2 + x*y = x^3 + x^2 + 1
over Finite Field in theta of size 2^163
```

但是，在实例化曲线的密码学安全规模时必须非常小心，因为实际运行时曲线对象的某些函数的运行需要指数级时间，所以无法正常工作。如果需要进行这些计算的实验时，最好在小规模的曲线上进行。

你可以计算出曲线的某些值，例如点的个数

```
sage: E = EllipticCurve(GF(107), [0,0,0,1,0])
sage: E.order()
108
```

还可以判断曲线的生成元

```
sage: E = EllipticCurve(GF(101), [0,0,0,1,0])
sage: E.gens()
((7 : 42 : 1), (36 : 38 : 1))
```

注意输出结果表示为(x : y : z)。这是出于快速实现的考虑，因为 Sage 使用“投影坐标”来存储曲线上的点。精确表示的意义不大，因为对于非无穷远点，其值 z 的取值总是 1，而前两个值同预期一样取 x 坐标和 y 坐标。该表示方法的优点是可以将无穷远点表示为 z 坐标为 0 的点

```
sage: E(0)
(0 : 1 : 0)
```

这样就可以识别一个点是否是无穷远点。如果想得到曲线上点的 x 坐标和 y 坐标，只需如下操作：

```
sage: P = E.random_point(); P
(62 : 38 : 1)
sage: (x,y) = P.xy(); (x,y)
(62, 38)
```

通过将有序的值对投射到曲线上，可得到一个点

```
sage: P = E((62,-38)); P
(62 : 63 : 1)
```

现在已经可以找到曲线的生成元并指定点，以及进行算术运算。继续使用前面例子中的曲线 E，可以设置 G1 和 G2 作为生成元

```
sage: (G1, G2) = E.gens()
sage: P = E.random_point(); P
(49 : 29 : 1)
```

对于两个点的加法运算示例如下：

```
sage: G1 + G2 + P
(69 : 96 : 1)
sage: G1 + P
(40 : 62 : 1)
sage: P + P + G2
(84 : 25 : 1)
```

使用单目运算符负号(–)可以计算点的逆

```
sage: -P
(49 : 72 : 1)
sage: -G1
(7 : 59 : 1)
```

使用*运算符可以进行点的自加运算（点重复与自身相加）

```
sage: 13*G1
(72 : 23 : 1)
sage: 2*G2
(9 : 58 : 1)
sage: 88*P
(87 : 75 : 1)
```

对于较小规模有限域上的曲线可以计算阶（对于无穷远点的离散对数）

```
sage: G1.order()
10

sage: G2.order()
10

sage: P.order()
10
```

例 6　使用 Sage 椭圆曲线函数执行椭圆曲线 Diffie-Hellman（ECDH）密钥交换。

```
sage: # calculate domain parameters
sage: F = GF(127)
sage: E = EllipticCurve(F, [0, 0, 0, 3, 4])
sage: G = E.gen(0); G
(94 : 6 : 1)
sage: q = E.order(); q
122

sage: # Alice computes a secret value x in
2 . . . q-1
sage: x = randint(2,q-1); x
33

sage: # Alice computes a public value X = x*G
sage: X = x*G; X
```

```
(55 : 89 : 1)

sage: # Bob computes a secret value y in 2 . . . q-1
sage: y = randint(2,q-1); y
55

sage: # Bob computes a public value Y = y*G
sage: Y = y*G; Y
(84 : 39 : 1)

sage: # Alice computes the shared value
sage: x*Y
(91 : 105 : 1)

sage: # Bob computes the shared value
sage: y*X
(91 : 105 : 1)
```

然而，在实践中，通常使用阶为素数的曲线

```
sage: # Calculate the domain parameters
sage: F = GF(101)
sage: E = EllipticCurve(F, [0, 0, 0, 25, 7])
sage: G = E((97,34))
sage: q = E.order()
sage: # Alice computes a secret values x in 2 . . . q-1
sage: x = randint(2,q-1)
sage: # Alice computes a public value X = x*G
sage: X = x*G
sage: # Bob computes a secret value y in 2 . . . q-1
sage: y = randint(2,q-1)
sage: # Bob computes a public value Y = y*G
sage: Y = y*G
sage: # Alice computes the shared secret value
sage: x*Y
(23 : 15 : 1)
sage: # Bob computes the shared secret value
sage: y*X
(23 : 15 : 1)
```

B.10 第 11 章 密码学 Hash 函数

例 1 下面是 Sage 中的 Hash 函数 MASH 的例子。MASH 是基于模算术的算法，使用了类似 RSA 的模 M，其中 M 的长度决定其安全强度。M 必须很难被分解因式，并且对于 M 的某些因子，它的安全性基于模运算下方根的计算困难性。M 同时还决定了待处理的消息的分组长度。本质上 MASH 定义为

$$H_i = ((x_i \oplus H_{i-1})^2 OR\, H_{i-1}) (\mathrm{mod}\, M)$$

其中，A = 0xFF00...00；H_{i-1} = 小于 M 的最大素数；x_i = 对于输入 n 按 M 的幂方展开，所得结果的第 i 位。即我们将 n 表示为 M 进制的形式

$$n = x_0 + x_1 M + x_2 M^2 + \cdots$$

接下来是 Sage 中的 Hash 函数 MASH 的例子：

```
#
# This function generates a mash modulus
# takes a bit length, and returns a Mash
# modulus l or l-1 bits long (if n is odd)
# returns p, q, and the product N
#
def generate_mash_modulus(l):
    m = l.quo_rem(2)[0]
    p = 1
    while (p < 2^(m-1)):
    p = random_prime(2^m)
    q = 1
    while (q < 2^(m-1)):
    q = random_prime(2^m)
    N = p*q
    return (N, p, q)
#
# Mash Hash
# the value n is the data to be hashed.
# the value N is the modulus
# Returns the hash value.
#
def MASH(n, N):
    H = previous_prime(N)
    q = n
    while (0 != q):
        (q, a) = q.quo_rem(N)
        H = ((H+a)^2 + H) % N
    return H
```

该函数的输出如下：

```
sage: data = ZZ(randint(1,2^1000))
sage: (N, p, q) = generate_mash_modulus(20)
sage: MASH(data, N)
220874
sage: (N, p, q) = generate_mash_modulus(50)
sage: MASH(data, N)
455794413217080
sage: (N, p, q) = generate_mash_modulus(100)
sage: MASH(data, N)
268864504538508517754648285037
sage: data = ZZ(randint(1,2^1000))
sage: MASH(data, N)
236862581074736881919296071248
sage: data = ZZ(randint(1,2^1000))
sage: MASH(data, N)
395463068716770866931052945515
```

B.11 第 13 章 数字签名

例 1 使用 Sage 能够执行 DSA 签名和验证

```
sage: # First we generate the domain parameters
sage: # Generate a 16 bit prime q
sage: q = 1;
sage: while (q < 2^15): q = random_prime(2^16)
. . . .:
sage: q
42697
sage: # Generate a 64 bit p, such that q divides
(p-1)
sage: p = 1
sage: while (not is_prime(p)):
. . . .: p = (2^48 + randint(1,2^46)*2)*q + 1
. . . .:
sage: p
12797003281321319017
sage: # Generate h and g
sage: h = randint(2,p-2)
sage: h
5751574539220326847
sage: F = GF(p)
sage: g = F(h)^((p-1)/q)
sage: g
9670562682258945855

sage: # Generate a user public / private key
sage: # private key
sage: x = randint(2,q-1)
sage: x
20499
sage: # public key
sage: y = F(g)^x
sage: y
7955052828197610751
sage: # Sign and verify a random value
sage: H = randint(2,p-1)
sage: # Signing
sage: # random blinding value
sage: k = randint(2,q-1)
sage: r = F(g)^k % q
sage: r = F(g)^k
sage: r = r.lift() % q
sage: r
6805
sage: kinv = xgcd(k,q)[1] % q
sage: s = kinv*(H + x*r) % q
sage: s
26026

sage: # Verifying
sage: w = xgcd(s,q)[1]; w
```

```
12250
sage: u1 = H*w % q; u1
6694
sage: u2 = r*w % q; u2
16706
sage: v = F(g)^u1 * F(y)^u2
sage: v = v.lift() % q
sage: v
6805
sage: v == r
True
sage: # Sign and verify another random value
sage: H = randint(2,p-1)
sage: k = randint(2,q-1)
sage: r = F(g)^k
sage: r = r.lift() % q
sage: r
3284
sage: kinv = xgcd(k,q)[1] % q
sage: s = kinv*(H + x*r) % q
sage: s
2330
sage: # Verifying
sage: w = xgcd(s,q)[1]; w
4343
sage: u1 = H*w % q; u1
32191
sage: u2 = r*w % q; u2
1614
sage: v = F(g)^u1 * F(y)^u2
sage: v = v.lift() % q
sage: v
3284
sage: v == r
True
```

例 2 下面的函数实现 DSA 的参数生成、密钥生成以及 DSA 签名

```
#
# Generates a 16 bit q and 64 bit p, both prime
# such that q divides p-1
#
def DSA_generate_domain_parameters():
    g = 1
    while (1 == g):
        # first find a q
        q = 1
        while (q < 2^15): q = random_prime(2^16)
        # next find a p
        p = 1
        while (not is_prime(p)):
            p = (2^47 + randint(1,2^45)*2)*q + 1
        F = GF(p)
        h = randint(2,p-1)
```

```
        g = (F(h)^((p-1)/q)).lift()
    return (p, q, g)

#
# Generates a users private and public key
# given domain parameters p, q, and g
#
def DSA_generate_keypair(p, q, g):
    x = randint(2,q-1)
    F = GF(p)
    y = F(g)^x
    y = y.lift()
    return (x,y)

#
# Given domain parameters p, q and g
# as well as a secret key x
# and a hash value H
# this performs the DSA signing algorithm
#
def DSA_sign(p, q, g, x, H):
    k = randint(2,q-1)
    F = GF(p)
    r = F(g)^k
    r = r.lift() % q
    kinv = xgcd(k,q)[1] % q
    s = kinv*(H + x*r) % q
    return (r, s)
```

参考文献

缩写词

ACM Association for Computing Machinery（计算机协会）

IBM International Business Machines Corporation（国际商用机器公司）

IEEE Institute of Electrical and Electronics Engineers（电气电子工程师学会）

NIST National Institute of Standards and Technology［（美国）国家标准和技术研究所］

ADAM94 Adams, C. "Simple and Effective Key Scheduling for Symmetric Ciphers." *Proceedings, Workshop on Selected Areas of Cryptography, SAC '94*, 1994.

AGRA04 Agrawal, M.; Kayal, N.; and Saxena, N. "PRIMES is in P." *IIT Kanpur, Annals of Mathematics*, September 2004.

AROR12 Arora, M. "How Secure is AES Against Brute-Force Attack?" *EE Times*, May 7, 2012.

ALFA13 Al Fardan, N., et al. "On the Security of RC4 in TLS and WPA." *USENIX Security Symposium*, July 2013.

BARD12 Bardou, R., et al. "Efficient Padding Oracle Attacks on Cryptographic Hardware," INRIA, Rapport de recherche RR-7944, Apr. 2012. http://hal.inria.fr/hal-00691958.

BASU12 Basu, A. *Intel AES-NI Performance Testing over Full Disk Encryption.* Intel Corp. May 2012.

BELL90 Bellovin, S., and Merritt, M. "Limitations of the Kerberos Authentication System." *Computer Communications Review*, October 1990.

BELL94 Bellare, M., and Rogaway, P. "Optimal Asymmetric Encryption—How to Encrypt with RSA." *Proceedings, Eurocrypt '94*, 1994.

BELL96a Bellare, M.; Canetti, R.; and Krawczyk, H. "Keying Hash Functions for Message Authentication." *Proceedings, CRYPTO '96*, August 1996; published by 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.

BELL96c Bellare, M., and Rogaway, P. "The Exact Security of Digital Signatures – How to Sign with RSA and Rabin." *Advances in Cryptology – Eurocrypt '96*, 1996.

BELL97 Bellare, M., and Rogaway, P. "Collision-Resistant Hashing: Towards Making UOWHF's Practical." *Proceedings, CRYPTO '97*, 1997; published by Springer-Verlag.

BELL98 Bellare, M., and Rogaway, P. "PSS: Provably Secure Encoding Method for Digital Signatures." *Submission to IEEE P1363*, August 1998. Available from http://grouper.ieee.org/groups/1363.

BELL00 Bellare, M.; Kilian, J.; and Rogaway, P. "The Security of the Cipher Block Chaining Message Authentication Code." *Journal of Computer and System Sciences*, December 2000.

BELL09 Bellare, M., et al. "Format Preserving Encryption." *Proceedings of SAC 2009 (Selected Areas in Cryptography)*, November 2009. Available at *Cryptology ePrint Archive* http://eprint.iacr.org/2009/.

BELL10a Bellare, M.; Rogaway, P.; and Spies, T. *The FFX Mode of Operation for Format-Preserving Encryption, Draft 1.1.* NIST, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf, February, 2010.

BELL10b Bellare, M.; Rogaway, P.; and Spies, T. *Addendum to the FFX Mode of Operation for Format-Preserving Encryption: A Parameter Collection for Enciphering Strings of Arbitrary Radix and Length.* NIST, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec2.pdf, September 2010.

BERT07 Bertoni, G., et al. "Sponge Functions." *Ecrypt Hash Workshop 2007*, May 2007.

BERT11 Bertoni, G., et al. "Cryptographic Sponge Functions." January 2011, http://sponge.noekeon.org/.

BETH91 Beth, T.; Frisch, M.; and Simmons, G., Eds. *Public-Key Cryptography: State of the Art and Future Directions.* New York: Springer-Verlag, 1991.

BLAC00 Black, J.; Rogaway, P.; and Shrimpton, T. "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions." *Advances in Cryptology – CRYPTO '00*, 2000.

BLAC05 Black, J. "Authenticated Encryption." *Encyclopedia of Cryptography and Security*, Springer, 2005.

BLEI98 Bleichenbacher, D. "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1," *CRYPTO '98*, 1998.

BLUM86 Blum, L.; Blum, M.; and Shub, M. "A Simple Unpredictable Pseudo-Random Number Generator." *SIAM Journal on Computing*, No. 2, 1986.

BONE02 Boneh, D., and Shacham, H. "Fast Variants of RSA." *CryptoBytes*, Winter/Spring 2002. http://www.rsasecurity.com/rsalabs.

BRIE10 Brier, E.; Peyrin, T.; and Stern, J. *BPS: A Format-Preserving Encryption Proposal.* NIST, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf, April 2010.

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.

BROW07 Brown, D., and Gjosteen, K. "A Security Analysis of the NIST SP 800-90 Elliptic Curve Random Number Generator." *Proceedings, Crypto '07*, 2007.

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.

CAMP92 Campbell, K., and Wiener, M. "Proof that DES is Not a Group." *Proceedings, Crypto '92*, 1992; published by Springer-Verlag.

CHOI08 Choi, M., et al. "Wireless Network Security: Vulnerabilities, Threats and Countermeasures." *International Journal of Multimedia and Ubiquitous Engineering*, July 2008.

COMP06 Computer Associates International. *The Business Value of Identity Federation.* White Paper, January 2006.

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.

CSA10 Cloud Security Alliance. *Top Threats to Cloud Computing V1.0.* CSA Report, March 2010.

CSA11a Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0.* CSA Report, 2011.

CSA11b Cloud Security Alliance. *Security as a Service (SecaaS).* CSA Report, 2011.

DAEM99 Daemen, J., and Rijmen, V. *AES Proposal: Rijndael, Version 2.* Submission to NIST, March 1999. http://csrc.nist.gov/archive/aes/index.html.

DAEM01 Daemen, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard." *Dr. Dobb's Journal*, March 2001.

DAMG89 Damgard, I. "A Design Principle for Hash Functions." *Proceedings, CRYPTO '89*, 1989; published by Springer-Verlag.

DAMI03 Damiani, E., et al. "Balancing Confidentiality and Efficiency in Untrusted Relational Databases." *Proceedings, Tenth ACM Conference on Computer and Communications Security*, 2003.

DAMI05 Damiani, E., et al. "Key Management for Multi-User Encrypted Databases." *Proceedings, 2005 ACM Workshop on Storage Security and Survivability*, 2005.

DAVI89 Davies, D., and Price, W. *Security for Computer Networks.* New York: Wiley, 1989.

DAWS96 Dawson, E., and Nielsen, L. "Automated Cryptoanalysis of XOR Plaintext Strings." *Cryptologia*, April 1996.

DENN81 Denning, D., and Sacco, G. "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.

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.

DIMI07 Dimitriadis, C. "Analyzing the Security of Internet Banking Authentication Mechanisms." *Information Systems Control Journal*, Vol. 3, 2007.

DOBB96 Dobbertin, H. "The Status of MD5 After a Recent Attack." *CryptoBytes*, Summer 1996.

ELGA84 Elgamal, T. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." *Proceedings, Crypto 84*, 1984.

ELGA85 Elgamal, T. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." *IEEE Transactions on Information Theory*, July 1985.

ENIS09 European Network and Information Security Agency. *Cloud Computing: Benefits, Risks and Recommendations for Information Security.* ENISA Report, November 2009.

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.

FRAN07 Frankel, S., et al. *Establishing Wireless Robust Security Networks: A Guide to IEEE 802.11i.* NIST Special Publication SP 800-97, February 2007.

GARD77 Gardner, M. "A New Kind of Cipher That Would Take Millions of Years to Break." *Scientific American*, August 1977.

GEOR12 Georgiev, M., et al. "The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software." *ACM Conference on Computer and Communications Security*, 2012.

GOLD88 Goldwasser, S.; Micali, S.; and Rivest, R. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks." *SIAM Journal on Computing*, April 1988.

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.

GOOD11 Goodin, D. "Hackers Break SSL Encryption Used by Millions of Sites." *The Register*, September 19, 2011.

GOOD12 Goodin, D. "Crack in Internet's Foundation of Trust Allows HTTPS Session Hijacking." *Ars Technica*, September 13, 2012.

GUTT06 Gutterman, Z.; Pinkas, B.; and Reinman, T. "Analysis of the Linux Random Number Generator." *Proceedings, 2006 IEEE Symposium on Security and Privacy*, 2006.

HACI02 Hacigumus, H., et al. "Executing SQL over Encrypted Data in the Database-Service-Provider Model." *Proceedings, 2002 ACM SIGMOD International Conference on Management of Data*, 2002.

HELL79 Hellman, M. "The Mathematics of Public-Key Cryptography." *Scientific American*, August 1970.

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.

HILT06 Hiltgen, A.; Kramp, T.; and Wiegold, T. "Secure Internet Banking Authentication." *IEEE Security and Privacy*, Vol. 4, No. 2, 2006.

HOWA03 Howard, M.; Pincus, J.; and Wing, J. "Measuring Relative Attack Surfaces." *Proceedings, Workshop on Advanced Developments in Software and Systems Security*, 2003.

HUIT98 Huitema, C. *IPv6: The New Internet Protocol.* Upper Saddle River, NJ: Prentice Hall, 1998.

IANS90 l'Anson, C., and Mitchell, C. "Security Defects in CCITT Recommendation X.509—The Directory Authentication Framework." *Computer Communications Review*, April 1990.

INTE12 Intel Corp. *Intel® Digital Random Number Generator (DRNG) Software Implementation Guide.* August 7, 2012.

IWAT03 Iwata, T., and Kurosawa, K. "OMAC: One-Key CBC MAC." *Proceedings, Fast Software Encryption*, FSE '03, 2003.

JAIN91 Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* New York: Wiley, 1991.

JAKO98 Jakobsson, M.; Shriver, E.; Hillyer, B.; and Juels, A. "A Practical Secure Physical Random Bit Generator." *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, November 1998.

JANS11 Jansen, W., and Grance, T. *Guidelines on Security and Privacy in Public Cloud Computing.* NIST Special Publication 800-144, January 2011.

JOHN05 Johnson, D. "Hash Functions and Pseudorandomness." *Proceedings, First NIST Cryptographic Hash Workshop*, 2005.

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 1958.

JONS02 Jonsson, J. "On the Security of CTR + CBC-MAC." *Proceedings of Selected Areas in Cryptography – SAC 2002*, 2002.

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.

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.

KALI01 Kaliski, B. "RSA Digital Signatures." *Dr. Dobb's Journal*, May 2001.

KEHN92 Kehne, A.; Schonwalder, J.; and Langendorfer, H. "A Nonce-Based Protocol for Multiple Authentications." *Operating Systems Review*, October 1992.

KLEI10 Kleinjung, T., et al. "Factorization of a 768-bit RSA modulus." Listing 2010/006, *Cryptology ePrint Archive*, February 18, 2010.

KNUD98 Knudsen, L., et al. "Analysis Method for Alleged RC4." *Proceedings, ASIACRYPT '98*, 1998.

KNUD00 Knudson, L. "Block Chaining Modes of Operation." *NIST First Modes of Operation Workshop*, October 2000. http://csrc.nist.gov/groups/ST/toolkit/BCM/workshops.html.

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Reading, MA: Addison-Wesley, 1998.

KOCH96 Kocher, P. "Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS, and Other Systems." *Proceedings, Crypto '96*, August 1996.

KOHL89 Kohl, J. "The Use of Encryption in Kerberos for Network Authentication." *Proceedings, Crypto '89*, 1989; published by 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.

KOHN78 Kohnfelder, L. *Towards a Practical Public Key Cryptosystem.* Bachelor's Thesis, M.I.T. 1978.

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

KUMA11 Kumar, M. "The Hacker's Choice Releases SSL DOS Tool." The *Hacker News*, October 24, 2011. http://thehackernews.com/2011/10/hackers-choice-releases-ssl-ddos-tool.html#.

LAM92a Lam, K., and Gollmann, D. "Freshness Assurance of Authentication Protocols." *Proceedings, ESORICS 92,* 1992; published by Springer-Verlag.

LAM92b Lam, K., and Beth, T. "Timely Authentication in Distributed Systems." *Proceedings, ESORICS 92,* 1992; published by Springer-Verlag.

LATT09 Lattin, B. "Upgrade to Suite B Security Algorithms." *Network World*, June 1, 2009.

LE93 Le, A., et al. "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.

LEUT94 Leutwyler, K. "Superhack." *Scientific American*, July 1994.

LEVE90 Leveque, W. *Elementary Theory of Numbers.* New York: Dover, 1990.

LEWA00 Lewand, R. *Cryptological Mathematics.* Washington, D.C.: 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/groups/ST/toolkit/BCM/workshops.html.

LISK02 Liskov, M.; Rivest, R.; and Wagner, D. "Tweakable Block Ciphers. *Advances in Cryptology – CRYPTO 2002*, 2002.

MA10 Ma, D., and Tsudik, G. "Security and Privacy in Emerging Wireless Networks." *IEEE Wireless Communications*, October 2010.

MANA11 Manadhata, P., and Wing, J. "An Attack Surface Metric." *IEEE Transactions on Software Engineering*, Vol. 37, No. 3, 2011.

MANT01 Mantin, I., Shamir, A. "A Practical Attack on Broadcast RC4." *Proceedings, Fast Software Encryption*, 2001.

MATY91a Matyas, S. "Key Handling with Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MATY91b Matyas, S.; Le, A.; and Abrahan, D. "A Key Management Scheme Based on Control Vectors." *IBM Systems Journal*, No. 2, 1991.

MAUW05 Mauw, S., and Oostdijk, M. "Foundations of Attack Trees." *International Conference on Information Security and Cryptology*, 2005.

MCGR04 McGrew, D., and Viega, J. "The Security and Performance of the Galois/Counter Mode (GCM) of Operation." *Proceedings, Indocrypt 2004.*

MCGR05 McGrew, D., and Viega, J. "Flexible and Efficient Message Authentication in Hardware and Software." 2005. Available at http://www.cryptobarn.com/gcm/gcm-paper.pdf.

MECH14 Mechalas, J. *Intel® Digital Random Number Generator (DRNG) Software Implementation Guide.* Intel Developer Zone, May 15, 2014. https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide.

MENE97 Menezes, A.; Oorshcot, P.; and Vanstone, S. *Handbook of Applied Cryptography.* Boca Raton, FL: CRC Press, 1997. Available at http://cacr.uwaterloo.ca/hac/index.html.

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; published by Springer-Verlag.

MEYE88 Meyer, C., and Schilling, M. "Secure Program Load with Modification Detection Code."

Proceedings, SECURICOM 88, 1988.

MEYE13 Meyer, C.; Schwenk, J.; and Gortz, H. "Lessons Learned From Previous SSL/TLS Attacks: A Brief Chronology of Attacks And Weaknesses." *Cryptology ePrint Archive*, 2013. http://eprint.iacr.org/2013/.

MICA91 Micali, S., and Schnorr, C. "Efficient, Perfect Polynomial Random Number Generators." *Journal of Cryptology*, January 1991.

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.

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].

MOOR01 Moore, A.; Ellison, R.; and Linger, R. "Attack Modeling for Information Security and Survivability." *Carnegie–Mellon University Technical Note CMU/SEI-2001-TN-001*, March 2001.

MYER91 Myers, L. *Spycomm: Covert Communication Techniques of the Underground.* Boulder, CO: Paladin Press, 1991.

NCAE13 National Centers of Academic Excellence in Information Assurance/Cyber Defense. *NCAE IA/CD Knowledge Units.* June 2013.

NEED78 Needham, R., and Schroeder, M. "Using Encryption for Authentication in Large Networks of Computers." *Communications of the ACM*, December 1978.

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.

NIST95 National Institute of Standards and Technology. *An Introduction to Computer Security: The NIST Handbook.* Special Publication 800-12. October 1995.

ODLY95 Odlyzko, A. "The Future of Integer Factorization." *CryptoBytes*, Summer 1995.

ORE67 Ore, O. *Invitation to Number Theory.* Washington, D.C.: 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.

PARZ06 Parziale, L., et al. *TCP/IP Tutorial and Technical Overview.* ibm.com/redbooks, 2006.

PAUL07 Paul, G., and Maitra, S. "Permutation after RC4 Key Scheduling Reveals the Secret Key", *Selected Areas of Cryptography: SAC 2007, Lecture Notes on Computer Science*, Vol. 4876, pp. 360–337, 2007.

PELL10 Pellegrini, A.; Bertacco, V.; and Austin, A. "Fault-Based Attack of RSA Authentication." *DATE '10 Proceedings of the Conference on Design, Automation, and Test in Europe*, March 2010.

POIN02 Pointcheval, D. "How to Encrypt Properly with RSA." *CryptoBytes*, Winter/Spring 2002. http://www.rsasecurity.com/rsalabs.

POPE79 Popek, G., and Kline, C. "Encryption and Secure Computer Networks." *ACM Computing Surveys*, December 1979.

PREN96 Preneel, B., and Oorschot, P. "On the Security of Two MAC Algorithms." *Lecture Notes in Computer Science 1561; Lectures on Data Security*, 1999; published by Springer-Verlag.

RABI78 Rabin, M. "Digitalized Signatures." *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.

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.

RIVE84 Rivest, R., and Shamir, A. "How to Expose an Eavesdropper." *Communications of the ACM*, April 1984.

ROBS95a Robshaw, M. *Stream Ciphers.* RSA Laboratories Technical Report TR-701, July 1995. http://www.rsasecurity.com/rsalabs.

ROBS95b Robshaw, M. *Block Ciphers.* RSA Laboratories Technical Report TR-601, August 1995. http://www.rsasecurity.com/rsalabs.

ROGA03 Rogaway, P., and Wagner, A. "A Critique of CCM." *Cryptology ePrint Archive: Report 2003/070*, April 2003.

ROGA04 Rogaway, P. "Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC." *Advances in Cryptology—Asiacrypt 2004. Lecture Notes in Computer Science*, Vol. 3329. Springer-Verlag, 2004.

ROGA10 Rogaway, P. "A Synopsis of Format-Preserving Encryption." *Unpublished Manuscript*, March 2010. http://web.cs.ucdavis.edu/~rogaway/papers.

ROS06 Ros, S. "Boosting the SOA with XML Networking." *The Internet Protocol Journal*, December 2006. cisco.com/ipj.

SALT75 Saltzer, J., and Schroeder, M. "The Protection of Information in Computer Systems." *Proceedings of the IEEE*, September 1975.

SCHN89 Schnorr, C. "Efficient Identification and Signatures for Smart Cards." *CRYPTO*, 1988.

SCHN91 Schnorr, C. "Efficient Signature Generation by Smart Cards." *Journal of Cryptology*, No. 3, 1991.

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

SCHN99 Schneier, B. "Attack Trees: Modeling Security Threats." *Dr. Dobb's Journal*, December 1999.

SCHO06 Schoenmakers, B., and Sidorenki, A. "Cryptanalysis of the Dual Elliptic Curve Pseudorandom Generator." *Cryptology ePrint Archive*, Report 2006/190, 2006. eprint .iacr.org.

SEAG08 Seagate Technology. *128-Bit Versus 256-Bit AES Encryption.* Seagate Technology Paper, 2008.

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, Fifteenth Edition*, 1993.

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

SINK09 Sinkov, A., and Feil, T. *Elementary Cryptanalysis: A Mathematical Approach.* Washington, D.C.: The Mathematical Association of America, 2009.

SMIT71 Smith, J. "The Design of Lucifer: A Cryptographic Device for Data Communications." *IBM Research Report RC 3326.* April 15, 1971.

STAL15 Stallings, W., and Brown, L. *Computer Security.* Upper Saddle River, NJ: Pearson, 2015.

STAL16 Stallings, W. *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.* Upper Saddle River, NJ: Pearson, 2016.

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.

STIN06 Stinson, D. *Cryptography: Theory and Practice.* Boca Raton, FL: CRC Press, 2006.

TAYL11 Taylor, G., and Cox, G. "Digital Randomness." *IEEE Spectrum*, September 2011.

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.

VANC11 Vance, J. *VAES3 Scheme for FFX.* NIST, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-ad-VAES3.pdf, May 2011.

VANO90 van Oorschot, P., and Wiener, M. "A Known-Plaintext Attack on Two-Key Triple Encryption." *Proceedings, EUROCRYPT '90*, 1990; published by Springer-Verlag.

VANO94 van 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.

VOYD83 Voydock, V., and Kent., S. "Security Mechanisms in High-Level Network Protocols." *Computing Surveys*, June 1983.

WANG05 Wang, X.; Yin, Y.; and Yu, H. "Finding Collisions in the Full SHA-1." *Proceedings, Crypto '05*, 2005; published by Springer-Verlag.

WAYN09 Wayner, P. *Disappearing Cryptography.* Boston: Burlington, MA: Morgan Kaufmann, 2009.

WEBS86 Webster, A., and Tavares, S. "On the Design of S-Boxes." *Proceedings, Crypto '85*, 1985; published by Springer-Verlag.

WIEN90 Wiener, M. "Cryptanalysis of Short RSA Secret Exponents." *IEEE Transactions on Information Theory*, Vol. 36, No. 3, 1990.

WOO92a Woo, T., and Lam, S. "Authentication for Distributed Systems." *Computer*, January 1992.

WOO92b Woo, T., and Lam, S. "'Authentication' Revisited." *Computer*, April 1992.

WOOD10 Wood, T., et al. "Disaster Recovery as a Cloud Service: Economic Benefits & Deployment Challenges." *Proceedings, USENIX HotCloud '10*, 2010.

YUVA79 Yuval, G. "How to Swindle Rabin." *Cryptologia*, July 1979.

XU10 Xu, L. *Securing the Enterprise with Intel AES-NI.* Intel White Paper, September 2010.