装备科技翻译图书基金资助专著



[美]乔纳森.卡茨
[以色列]耶胡达.林德尔 著
任 伟 译

现代密码学
——原理与协议

Introduction to Modern Cryptography:
Principles and Protocols

国防工业出版社
National Defense Industry Press

译 者 序

《现代密码学——原理与协议》一书的作者 Jonathan katz 和 Yehuda Lindell 是目前国际密码学领域的知名学者，分别师从于国际权威 Moti Yung 和 Oded Goldreich。该书自 2007 年 8 月首次出版以来，立即被多所美国顶尖大学选用为密码学课程研究生（或本科生）教程，其中包括斯坦福大学、普林斯顿大学、哈佛大学、加利福尼亚大学伯克利分校等。该书同时也在世界范围内被多所知名大学选用为现代密码学教材，其中包括以色列、德国、意大利、英国、比利时、加拿大、芬兰等国。其影响力正在继续扩大。

该书的内容涵盖了密码学的基本方面，如对称密钥密码学和公钥密码学，同时涵盖了其他教科书中不多见的主题，例如伪随机数发生器、伪随机函数（置换）、陷门置换、Paillier 加密以及随机预言模型等。并包括大量的习题和扩展阅读材料介绍。

该书的论述方式独具特色，全书使用统一的论述框架（形式化的安全定义、精确的假设、严格的安全证明）来介绍密码学知识，这不仅让初学者从一开始便接受到良好的数学训练、形成严谨的科学素养和现代密码学的思维习惯，而且对具备丰富密码学知识的读者而言，也非常具有启发性，能够从中学习到新的严格分析问题和解决问题的方法，这种方法是现代密码学研究中频繁使用的方法。

该书的另一个特点是内容组织独具匠心、逻辑上层层递进地阐述了基于计算复杂性的现代密码学的各个方面。从经典密码学讲起，引出现代密码学的基本原则和完善保密的局限。然后引出不可区分性、规约、可忽略概率等核心概念，利用现代密码学的统一方法，论述了伪随机发生器（流密码，语义安全）、伪随机置换（分组密码，CPA 安全），通过消息鉴别码构造 CCA 安全的加密方案，并进一步探讨了基于单向函数构造伪随机发生器。然后过渡到公钥密码学的数论问题困难性假设，并进而论述公钥加密（陷门置换）和公钥签名、随机预言模型。同时注重通用构造方案的原理性讲解，便于读者由浅入深地体会现代密码学的奥秘。

译者在美国从事信息安全研究的几年里，深深体会到国内外现代密码学研究和教学内容方面的差异，引进并翻译这本优秀的专著对提高我国在密码学和信息安全领域的整体科研和教学水平，了解国外密码学和信息安全研究和教学动向，是非常及时和必要的。

感谢北京邮电大学杨义先教授和武汉大学张焕国教授的支持。感谢新加坡管理大学 Robert H. Deng（邓慧杰）教授的帮助。感谢 Jonathan Katz 在百忙之中为中文版写序。另外，感谢海军工程大学吴晓平教授。感谢中国密码学会教育工作委员会。感谢北京邮电

大学信息安全中心以及灵创团队。感谢国防工业出版社的各位编辑辛勤的工作。

本书的出版得到了2010年中国人民解放军总装备部国防装备翻译基金资助和中央高校基本科研业务费专项资金资助(No.090109),在此一并表示感谢。

全书由我主持翻译,并最终统稿和定稿。参与翻译和校对的人员还有:赵俊阁、叶敏、刘宇靓,在此向他们表示感谢。由于水平有限,不足之处在所难免,敬请读者批评和指正。我的email是:weirencs@gmail.com。

任　伟

2010年7月于武汉南望山

前　言

本书讲述现代密码学的基本范例和原理，其编写目的是作为（计算机科学系或数学系）本科或研究生阶段密码学课程的教科书，同时也是适合（研究生新生）自学的全面介绍现代密码学的一本书籍，它也可作为高等院校学生、研究人员以及从事工程实践人员的参考书。

当前有大量密码学方面的书籍，可能读者会问：是否还有必要再写一本这方面的书。如果对这一问题的回答不是肯定的话，这本书将不会问世。该书的创新之处，或者说区别于目前其他所有类似书籍之处在于——它提供了对现代密码学的严格论述，同时又以一种容易被接受的方式进行介绍。

如前所述，我们重点讲述现代（1980 年后的）密码学。现代密码学与经典密码学的区别在于：它强调定义、精确的假设、严格的安全证明。我们简单地依次讨论这些原则（第 1 章将对它们详细论述）。

- **定义的核心地位**：现代密码学的核心价值是意识到形式化的（即正式的）安全定义是设计密码学原语和协议的第一个必需步骤。其原因非常简单，如果对应该达到的目标不清楚，如何知道什么时候可以达到目标？在本书中将会看到，对密码方案的安全性定义过强的话，初看上去好像是不可能达到的，但密码学一个令人着迷的地方是：基于某些被广泛接受的假设，有效的构造方案是可以满足这些强安全定义和要求的，且方案的存在性是可以证明的。

- **形式化和精确假设的重要性**：第 2 章和第 3 章将要解释，很多密码学构造方案在当前并不能被证明是无条件安全的。取而代之，安全性通常是依赖一些广泛相信（即使还未被证明）的假设。现代密码学方法认为：任何这些假设必须清晰地加以陈述，而且无歧义地进行定义。这不仅是因为考虑到对假设的客观评价，更重要的是为了后续的严格的安全证明。

- **严格安全证明的可能性**：前面两个观点很自然地引出当前这个，也就是说，实现密码学构造方案的可证明安全性，需要具备表述清晰的安全定义和相对于定义明确的密码学假设。这是现代密码学的根本，正是这一点将密码学从艺术转变为科学。

这一点非常重要。历史上很多密码学方案是以一种临时的方式设计的，如果设计者自己无法发现任何攻击，则被认为是安全的。相比而言，现代密码学崇尚在明确定义的模型中，通过对安全性进行形式化的数学证明来设计方案。这些方案可以保证是安全的，除非所基于的假设是错误的（或者对安全的定义不能恰当地反映现实世界中所关心的安全性）。基于长期成立的假设（例如“因子分解是困难的”假设），可以得到几乎不可能攻破的方案。

统一的方法。前述的现代密码学的贡献不仅仅只是与密码学理论研究者相关。在当前精确定义的重要性已经被广泛认同，并且被安全的研究者（即用密码学工具构造安全系统的研究人员）所称道，同时，严格的安全证明已经成为安全方案标准化的一种要求。

有鉴于此,本书不将“应用密码学”从“可证明安全”中分离出来;相反,在给出实用的、广泛使用的密码学构造方案的同时,也精确地陈述其安全定义所达到的结果(大部分情况下表现为证明过程)。

本书使用指南

本节旨在帮助使用本书作为授课教材的授课者,学生也能从中发现对本书内容的有用概述。

(1)**需要的基础知识**。本书使用到定义、证明以及数学概念,因而需要具备一定的数学能力,本书假设读者具备大学数学水平,如学过高等数学、离散数学、算法,或计算理论。尽管如此,本书还是尽量简化表述方式,便于读懂。因此本书不会比表述并不严格的相关教材更难。相反,一旦安全目标被清晰地定义,将更容易理解特定密码学构造方法中的设计选择。

本书的组织安排使得先修课程仅为算法课程和离散数学课程。仅依赖非常少量的数学:特别是我们假设读者熟悉基本的概率知识、大 O 表示法、模算术以及多项式高效算法。这些概念在附录 A 中复习,或在书中第一次使用时复习。

(2)**课程组织的建议**。本书的核心内容,或者说在任何密码学概论课程中强烈建议的授课内容,包括以下几个方面(带星号的内容未包括,将在后面讨论):

- 第 1 章 ~ 第 4 章(到 4.6 节),讨论经典密码学、现代密码学以及对称密钥密码学的基础(对称密钥加密和消息鉴别)。
- 第 5 章,列举基本的分组密码设计原则,包括广泛使用的分组密码 DES 和 AES。[①]
- 第 7 章,介绍被公认为困难的具体数学问题,提供了为了理解 RSA、Diffie-Hellman 以及 El Gamal 密码系统的数论背景。该章还给出了如何在密码学中使用数论假设的第一批示例。
- 第 9 章和第 10 章,介绍了公钥密码学的动机,讨论了公钥加密(包括基于 RSA 的方案和 El Gamal 加密)。
- 第 12 章,描述了数字签名方案。
- 13.1 节和 13.3 节,介绍了随机预言模型和 RSA - FDH 签名方案。

这些核心内容(若忽略一些深入的讨论和证明)可在 30 学时 ~ 35 学时的本科课程中讲授。有更多授课学时的讲授者可以放慢节奏,例如讲授所有证明的细节,以及在介绍相关群论基础和数论背景时放慢速度。或者,增加下面的一些额外主题。

对那些希望讲授额外的内容,有更长授课学时或者使用快节奏讲授研究生课程的授课者而言,如果时间允许(或者授课者感兴趣),会发现本书的组织允许灵活地包含其他的主题,特别是带星号(*)的章节。这部分章节不是不重要,只是不宜作为密码学概论课程的核心内容。正如刚才列出的课程大纲(未包含带星号的内容),带星号的章节可以跳过,或者在其出现的位置讲授,都不会影响后面的课程,特别是本书确保了不带星号的内容不会用到带星号的内容。就大部分内容而言,带星号的章节之间并不互相依赖(如果有的话,这种依赖将被指出)。

建议下面选自带星号内容的主题给那些希望给课程增色的授课者:

- 理论:在更倾向于讲授理论的课程中,包括内容 3.2.2 节(加密方案的语义安全定

① 虽然我们认为这是核心内容,但是它不在书中的其他部分使用,因此如果愿意,可以跳过该章。

义);4.8 和 4.9 节(讲授对称密钥加密的强安全概念);第 6 章(介绍单向函数和硬核比特,从任何单向置换构造伪随机数发生器和伪随机函数/置换;10.7 节从任何陷门置换构造公钥加密);第 11 章(描述 Goldwasser - Micali、Rabin 以及 Paillier 加密方案);12.6 节(给出一个不依赖随机预言模型的签名方案)。

- 应用:希望强调实践方面的讲授者强烈建议讲授 4.7 节(描述 HMAC)和第 13 章全部(给出基于随机预言模型的密码学构造方案)。
- 数学:如果学生有良好的数学基础,或者授课对象喜欢密码学的数学部分,可讲授第 7 章中更高级的数论内容(如中国剩余定理、椭圆曲线群);第 8 章的全部(大数分解和离散对数算法);第 11 章的部分(描述 Goldwasser - Mical、Rabin 和 Paillier 加密方案的数论背景)。

建议和勘误

本书的写作目的是使现代密码学可以被"理论计算机科学"研究者以外的广大读者所接受。希望读者告诉我们是否达到了这一目的。我们非常乐意收到关于此书的反馈、关于如何改进此书的建设性意见。我们希望书中没有错误或者笔误;但是如果你发现有并告诉我们,我们将十分感激(一个已知的勘误表将在 http://www.cs.umd.edu/~jkatz/imc.html 维护)。可以将您的建议和勘误发给 jkatz@cs.umd.edu 或 lindell@cs.biu.ac.il;主题栏请注明"Introduction to Modern Cryptography"。

鸣谢

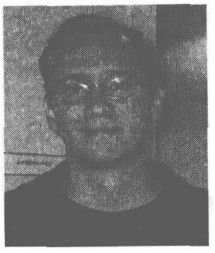
Jonathan Katz:感激 Zvi Galil, Moti Yung 和 Rafail Ostrovsky 在我职业生涯过程中的帮助、指导以及支持。没有他们的对我个人发展的贡献,本书不可能面世。同时感谢我的同事们,我曾经和他们多次讨论关于写一本密码学教材的"正确"方法。我的工作被国家自然科学基金委员会资助,项目号为#0627306、#0447075 以及#0310751。书中表达的所有意见、发现以及结论或者建议,都是我个人的,并不代表国家自然科学基金委员会的观点。

Yehuda Lindell:首先感谢 Oded Goldreich 和 Moni Naor 把我带入到密码学世界。他们影响我直至今日,并将毫无疑问地延续到将来。还有很多很多其他人多年来一直对我影响很大,这里不一一提及,只是说一声谢谢,他们知道我说的是谁。

感谢 Zoe Bermant 绘制了本书中的图片;David Wagner 回答了关于分组密码及其密码学分析方面的问题;Salil Vadhan 和 Alon Rosen 在哈佛大学的密码学概论课程中试用了该教材,并提供有价值的反馈意见。我们一起向阅读该书草稿并给出建议的人们,以及对预印版本提出修改意见的人们表示感谢:Adam Bender、Chiu - Yuen Koo、Yair Dombb、Michael Fuhr、William Glenn、S. Dov Gordon、Carmit Hazay、Eyal Kushilevitz、Avivit Levy、Matthew Mah、Ryan Murphy、Steve Myers、Martin Paraskevov、Eli Quiroz、Jason Rogers、Rui Xue、Dicky Yan、Arkady Yerukhimovich 和 Hila Zarosim。他们的建议极大地提高了书的质量并减少了错误。非常感谢那些鼓励我们写作本书的人们,他们和我们的感觉一致,那就是这样一本书是急需的。

最后,分别感谢在写作本书的许许多多日子里我们的妻子和孩子们的支持和理解。

Preface to Chinese Edition



The goal of this book is to provide an accessible introduction to modern cryptography, with a focus on definitions, assumptions, and proofs of security. In writing this book, we hoped to reach as many people as possible; I am therefore particularly thrilled that this book will now find an audience in China.

It was perhaps fortuitous that the translation of this book into Chinese was completed during the same week that I was visiting China for the first time. While there, I had a chance to learn about the tremendous variety of academic cryptography research taking place in China, and to see the large number of students interested in this subject. It is my sincere wish that this book will inspire the next generation of researchers, and help foster international cooperation and collaboration for the advancement of scientific knowledge.

Jonathan Katz

College Park, MD, USA

July 2010

中文版作者序

本书的目的是提供对现代密码学一种容易接受的介绍，重点在于定义、假设和安全性证明。在写作本书时，我们希望能够拥有尽可能广泛的读者群；因此现在我对该书能拥有中国读者而感到特别地激动。

也许是一种巧合，本书的翻译工作在我首次访问中国的同一周完成。在该次中国之行中，我有机会了解到中国正开展着数量巨大的多种多样的密码学学术研究，而且看到有大量的学生对这个主题感兴趣。我真诚地祝愿该书能够启发新一代研究者，并帮助增进国际间的交流与合作，以推动科学知识的进步。

乔纳森·卡茨

美国，马里兰大学

2010 年 7 月

目　录

第一部分　经典密码学介绍

第 1 章　概论 ····· 1

1.1　密码学和现代密码学 ····· 1
1.2　对称密钥加密的基本设置 ····· 2
1.3　古典加密术及其密码分析 ····· 5
1.4　现代密码学的基本原则 ····· 10
1.4.1　原则 1——形成精确的定义 ····· 11
1.4.2　原则 2——精确假设的依赖 ····· 14
1.4.3　原则 3——严格的安全证明 ····· 15
参考文献和扩展阅读材料介绍 ····· 15
练习 ····· 16

第 2 章　完善保密加密 ····· 17

2.1　定义和基本属性 ····· 17
2.2　一次一密（Vernam 加密） ····· 20
2.3　完善保密的局限 ····· 22
2.4　*香农定理 ····· 22
2.5　小结 ····· 24
参考文献和扩展阅读材料介绍 ····· 24
练习 ····· 25

第二部分　对称密钥（对称）密码学

第 3 章　对称密钥加密以及伪随机性 ····· 27

3.1　密码学的计算方法 ····· 27
3.1.1　计算安全的基本思想 ····· 28
3.1.2　有效的算法和可忽略的成功概率 ····· 31
3.1.3　规约证明 ····· 34
3.2　定义计算安全的加密 ····· 35
3.2.1　安全的基本定义 ····· 36
3.2.2　*定义的属性 ····· 38
3.3　伪随机性 ····· 41
3.4　构造安全加密方案 ····· 43
3.4.1　一个安全的定长加密方案 ····· 43
3.4.2　处理变长消息 ····· 46

3.4.3 流密码和多个加密 ……47
3.5 选择明文攻击（CPA）的安全性 ……50
3.6 CPA 安全的加密方案创建 ……52
3.6.1 伪随机函数 ……53
3.6.2 基于伪随机函数的 CPA 安全加密 ……55
3.6.3 伪随机置换和分组加密 ……58
3.6.4 加密操作模式 ……60
3.7 CCA 安全性 ……65
参考文献和扩展阅读材料介绍 ……66
练习 ……67
第 4 章 消息鉴别码和抗碰撞散列函数 ……70
4.1 安全通信与消息完整性 ……70
4.2 加密与消息鉴别的比较 ……70
4.3 消息鉴别码——定义 ……71
4.4 构造安全的消息鉴别码 ……74
4.5 CBC-MAC ……79
4.6 抗碰撞散列函数 ……81
4.6.1 抗碰撞的定义 ……81
4.6.2 散列函数安全性的一个较弱的定义 ……82
4.6.3 通用“生日”攻击 ……83
4.6.4 Merkle-Damgard 变换 ……84
4.6.5 实践中的抗碰撞散列函数 ……86
4.7 *NMAC 与 HMAC ……87
4.7.1 嵌套 MAC(NMAC) ……88
4.7.2 HMAC ……90
4.8 *构造 CCA 安全加密方案 ……91
4.9 *获得私密性和消息鉴别 ……95
参考文献和扩展阅读材料介绍 ……99
练习 ……99
第 5 章 伪随机置换（分组加密）的实际构建 ……102
5.1 代替—置换网络 ……104
5.2 Feistel 网络 ……109
5.3 DES——数据加密标准 ……110
5.3.1 DES 的设计 ……111
5.3.2 对减少轮数的 DES 变种的攻击 ……113
5.3.3 DES 的安全性 ……115
5.4 增加分组密码的密钥长度 ……116
5.5 AES——高级加密标准 ……118
5.6 差分密码分析和线性密码分析简介 ……120

参考文献和扩展阅读材料介绍 ····· 121
练习 ····· 121
第 6 章 *伪随机对象的理论构造 ····· 124
6.1 单向函数 ····· 124
6.1.1 定义 ····· 125
6.1.2 候选单向函数 ····· 127
6.1.3 硬核谓词 ····· 128
6.2 概述：从单向函数到伪随机置换 ····· 129
6.3 任意单向函数的硬核谓词 ····· 130
6.3.1 简单情形 ····· 130
6.3.2 复杂情形 ····· 131
6.3.3 完整证明 ····· 134
6.4 构造伪随机发生器 ····· 138
6.4.1 具有最小扩展的伪随机发生器 ····· 138
6.4.2 提高扩展系数 ····· 139
6.5 构造伪随机函数 ····· 143
6.6 构造（强）伪随机置换 ····· 146
6.7 对称密钥密码学的必要假设 ····· 148
6.8 偏离——计算不可区分性 ····· 151
6.8.1 伪随机性和伪随机发生器 ····· 151
6.8.2 多重抽样 ····· 152
参考文献及扩展阅读材料介绍 ····· 154
练习 ····· 154

第三部分 公钥（非对称密钥）密码学

第 7 章 数论和密码学困难性假设 ····· 156
7.1 预备知识和基本群论 ····· 157
7.1.1 素数与可除性 ····· 158
7.1.2 模算术 ····· 159
7.1.3 群 ····· 160
7.1.4 $\mathbb{Z}_N^*$群 ····· 163
7.1.5 *群同构和中国剩余定理 ····· 164
7.2 素数、大数分解和 RSA ····· 168
7.2.1 随机素数的产生 ····· 169
7.2.2 *素数判定 ····· 171
7.2.3 因子分解假设 ····· 175
7.2.4 RSA 假设 ····· 176
7.3 循环群中的假设 ····· 177
7.3.1 循环群和生成元 ····· 177

7.3.2 离散对数和 Diffie-Hellman 假设 …… 180
7.3.3 在 Zp*（的子群）中工作 …… 182
7.3.4 *椭圆曲线群 …… 183
7.4 数论假设的密码学应用 …… 186
7.4.1 单向函数和置换 …… 187
7.4.2 构造抗碰撞的散列函数 …… 189
参考文献和扩展阅读材料介绍 …… 191
练习 …… 191
第 8 章 因子分解和离散对数算法 …… 194
8.1 因子分解算法 …… 194
8.1.1 Pollard 的$p-1$方法 …… 195
8.1.2 Pollard 的 Rho 方法 …… 196
8.1.3 二次筛选算法 …… 198
8.2 计算离散对数的算法 …… 200
8.2.1 “小步大步”算法 …… 201
8.2.2 Pohlig-Hellman 算法 …… 202
8.2.3 Z_N中的离散对数问题 …… 203
8.2.4 索引演算方法 …… 204
参考文献和扩展阅读材料介绍 …… 205
练习 …… 205
第 9 章 对称密钥管理和公钥革命 …… 207
9.1 对称密钥加密的局限性 …… 207
9.2 局部解决方法——密钥分配中心 …… 208
9.3 公钥革命 …… 210
9.4 Diffe-Hellman 密钥交换 …… 212
参考文献和扩展阅读材料介绍 …… 216
练习 …… 217
第 10 章 公钥加密 …… 218
10.1 公钥加密简介 …… 218
10.2 定义 …… 220
10.2.1 选择明文攻击的安全性 …… 220
10.2.2 多重加密 …… 222
10.3 混合加密 …… 227
10.4 RSA 加密 …… 233
10.4.1 “教科书式 RSA”加密方案及其不安全性 …… 233
10.4.2 对“教科书式 RSA”加密方案的攻击 …… 235
10.4.3 填充 RSA …… 237

10.5 El Gamal 加密 ····· 239
10.6 选择密文攻击的安全性 ····· 242
10.7 *陷门置换 ····· 245
10.7.1 定义 ····· 245
10.7.2 来自陷门置换的公钥加密 ····· 246
参考文献和扩展阅读材料介绍 ····· 248
练习 ····· 249

第 11 章 *其他公钥加密方案 ····· 252

11.1 Goldwasser-Micali 加密方案 ····· 252
11.1.1 素数模二次剩余 ····· 252
11.1.2 合数模二次剩余 ····· 254
11.1.3 二次剩余假设 ····· 257
11.1.4 Goldwasser-Micali 加密方案 ····· 258
11.2 Rabin 加密方案 ····· 260
11.2.1 计算模平方根 ····· 260
11.2.2 基于分解的陷门置换 ····· 263
11.2.3 Rabin 加密方案 ····· 266
11.3 Paillier 加密方案 ····· 268
11.3.1 $\mathbb{Z}_{N^2}^*$结构 ····· 269
11.3.2 Paillier 加密方案 ····· 270
11.3.3 同态加密 ····· 273
参考文献和扩展阅读材料介绍 ····· 275
练习 ····· 275

第 12 章 数字签名 ····· 278

12.1 数字签名简介 ····· 278
12.2 定义 ····· 279
12.3 RSA 签名 ····· 281
12.3.1 “教科书式 RSA”签名方案及其不安全性 ····· 281
12.3.2 “散列后 RSA”签名方案 ····· 282
12.4 “Hash-and-Sign”范例 ····· 283
12.5 Lamport 的“一次性签名方案” ····· 285
12.6 *来自抗碰撞散列的签名 ····· 288
12.6.1 基于 Chain 的签名 ····· 288
12.6.2 基于 Tree 的签名 ····· 291
12.7 数字签名标准（DSS） ····· 295
12.8 数字证书和公钥基础设施 ····· 296
参考文献和扩展阅读材料介绍 ····· 300
练习 ····· 300

第 13 章　随机预言机模型中的公钥密码系统 ····· 303
13.1　随机预言机方法学 ····· 303
13.1.1　随机预言机模型 ····· 304
13.1.2　随机预言机方法是否合理 ····· 308
13.2　随机预言机模型中的公钥加密 ····· 310
13.2.1　选择明文攻击安全 ····· 310
13.2.2　选择密文攻击安全 ····· 313
13.2.3　OAEP ····· 318
13.3　随机预言机模型中的签名 ····· 319
参考文献和扩展阅读材料介绍 ····· 322
练习 ····· 322
常用符号索引 ····· 324
附录 ····· 327
附录 A　数学知识 ····· 327
A.1　恒等式和不等量 ····· 327
A.2　渐进表示法 ····· 327
A.3　概率论简介 ····· 328
A.4　“生日”问题 ····· 329
附录 B　算法数论补充知识 ····· 330
B.1　整数算术 ····· 332
B.2　模算术 ····· 334
B.3　*寻找一个循环群的生成元 ····· 339
参考文献和扩展阅读材料介绍 ····· 342
练习 ····· 342
参考文献 ····· 343

第一部分　经典密码学介绍

第1章　概　　论

1.1　密码学和现代密码学

《牛津简明词典》（2006 版）定义密码学为：编码或者解码的艺术。这个定义可能从历史角度看是准确的，但是它并没有抓住现代密码学的本质。首先，它仅仅关注秘密通信问题。理由是：定义中明确说明了“编码（code）”，将其定义为“一种预先安排好的信号系统，专用于确保传输消息时的保密性”。其次，该定义认为密码学是一种艺术。实际上，直到 20 世纪（或可认为直到 20 世纪末），密码学都是一种艺术。不论是构造好的编码，还是破解存在的编码，都依赖于创造性和个人技巧。只有非常少的理论可以依靠，甚至没有一个清晰的定义来说明什么构成好的编码。

到 20 世纪末，密码学的面貌从根本上得到了改变。出现了丰富的理论，于是可以将密码学作为一种科学来严格地学习。而且，当前密码学的领域拥有比秘密通信多得多的内容。它解决很多问题，如消息鉴别、数字签名、密钥交换协议、认证协议、电子拍卖和选举、数字现金等。实际上，现代密码学关注任何分布式计算中产生的，来源于内部和外部攻击的问题。这里不试图给出对现代密码学的完美定义，我们认为它是：对安全地进行数字信息和事务处理、分布式计算所需技术的科学研究。

另外，经典密码学（上世纪 80 年代以前）和现代密码学之间一个非常重要的区别是使用者。历史上，密码学的主要消费者是军事和智囊机构。但是现在，密码学无处不在，依靠密码学的安全机制几乎是计算机系统的一个集成部件。用户在访问一个安全的站点时（常常没有意识到）依靠了密码学。在多用户操作系统中，密码学方法确保了访问控制，并阻止窃贼从丢失的笔记本计算机中窃取商业机密。软件保护方法使用加密、认证和其他工具来阻止复制。我们可以继续举出很多这样的例子。

简言之，密码学从一种军事中处理秘密通信的艺术变成一种使全世界普通人的系统变得安全的科学。这意味着密码学成为计算机科学中一个越来越核心的主题。

本书关注的是“现代密码学”。但是将从密码学的上述变化之前开始学习，以便读者更容易进入主题，并了解密码学的来源，以及加深对改变程度的认识。经典密码学充满了临时的编码构造方法和相对简单的破解方式，这也是本书中采用严格方法的主要原因[①]。

① 这是我们编写该内容的主要目的，本章不应作为具有代表性的密码学历史报告。对密码学的历史感兴趣的读者请查阅章后所附参考文献。

1.2 对称密钥加密的基本设置

如上所述，在历史上密码学主要用于秘密通信。特别是使用密码学的加密术（现在叫做加密方案），使事先共享某种信息的两方能够秘密通信。该方式中，通信双方实现共享某种秘密信息，现在称为秘密密钥（private-key）（或对称密钥（symmetric-key））。在描述历史加密术之前，我们从通用术语的角度讨论秘密密钥情形和加密。

在对称密钥加密的情形中，双方共享一些密码信息，称为密钥（key），当他们希望安全地通信时，使用这个密钥。发送方使用密钥加密（或者“混合”）消息，接收方收到后使用相同的密钥解密（或者“去混合”）来恢复消息。消息本身称为明文（plaintext），从发送方传输到接收方的经加密的消息称为密文（ciphertext），如图 1.1 所示。共享的密钥用来从所有窃听者中区分通信方（假定通过公共信道传输）。

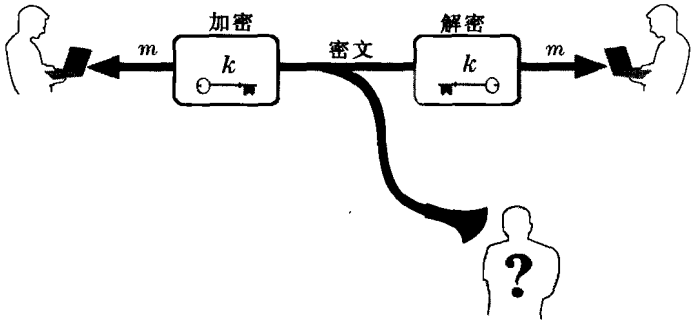


图 1.1 对称密钥加密的基本情形

在这一方式中，在明文和密文间变换时使用同一个密钥。这解释了为什么这一情形也被称为对称密钥（symmetric-key），这里对称是指双方拥有相同的密钥来加密和解密。这和非对称（asymmetric）加密（在第 9 章中介绍）相对应，后者的发送和接收方并不共享任何秘密并且有不同的密钥用于加密和解密。对称密钥是经典的，通过本章后面的内容将会了解到这一点。

在任何使用对称密钥加密的系统中，都隐含着一个假设：通信方可以使用某种秘密的方式建立初始的共享密钥（注意，如果一方通过公共信道简单地发送密钥给另一方，窃听者将也能获得密钥）。在军事领域中，这不是一个严重的问题，因为通信方能够在一个安全的位置实现物理上的相遇以认同一个密钥。但是当今在很多情况下，不可能安排通信方的碰面。如将在第 9 章中所述，这一点非常值得关注，并且实际上限制了仅依靠对称密钥的密码学系统的可应用性。尽管如此，对称密钥方法仍在很多情形下是够用的，且被广泛使用。例如：磁盘加密中同一个用户（在不同的时间点）使用固定的密钥来读写磁盘。正如将在第 10 章进一步讨论的，对称密钥加密也普遍和非对称密钥密码学联合使用。

加密的语法。对称密钥加密方案由三个算法组成：密钥产生，加密，解密。它们的功能如下：

(1) 密钥产生算法Gen是一个概率算法，能够根据方案定义的某种分布选择并输出一个密钥k。

(2) 加密算法Enc，输入为密钥k和明文m，输出为密文c。把使用密钥k加密明文m记为$\mathsf{Enc}_k(m)$。

(3) 解密算法Dec，输入为密钥k和密文c，输出为明文m。把使用密钥k解密密文记为$\mathsf{Dec}_k(c)$。

· 密钥产生函数输出的所有可能的密钥称为密钥空间，用$\mathcal{K}$表示。通常，Gen简单地从密钥空间中均匀随机地选择一个密钥（事实上，这种假设不失一般性）。所有“合法”消息（也就是那些被加密算法支持的消息）的集合记做$\mathcal{M}$，并称为明文（或消息）空间。因为任何密文是通过使用密钥加密明文来获得的，集合$\mathcal{K}$和$\mathcal{M}$一起定义了所有可能密文的集合$\mathcal{C}$。一个加密方案可通过明确三个算法（Gen，Enc，Dec）和明文空间$\mathcal{M}$来完全定义。

对任意加密方案的基本要求是：对于任意通过Gen输出的密钥k，每个明文消息$m \in \mathcal{M}$，满足

$$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$$

通过文字说明，解密一个密文（使用相应密钥）生成原来加密的原始消息。

回顾前面的讨论，一个加密方案可被通信双方以如下方式所使用：首先，Gen被运行获得各方共享的密钥k。当一方需要发送明文m到另一方，他首先计算$c := \mathsf{Enc}_k(m)$，然后通过公共信道发送密文c到另一方。另一方接收到c后，计算$m := \mathsf{Dec}_k(c)$恢复出原来的消息[②]。

密钥和 Kerckhoffs 原则。从前面所述可知，如果窃听敌手知道算法Dec以及通信双方共享的密钥k，则敌手可以解密他们间的所有通信。正因如此，通信双方必须安全地共享密钥k，并且保持k完全保密。但是，是否需要保持解密算法Dec为一个秘密？是不是所有组成加密方案的算法（即Gen和Enc）都需要保密？（注意通常认为明文空间$\mathcal{M}$是已知的，例如，可能由英文句子组成。）

在 19 世纪后期，Augueste Kerckhoffs 在一篇论文中对这一问题给出了意见，他发表了军用重要加密术的设计原则。其中最重要的（现在称作 Kerckhoffs 原则）是：加密方法不必保密，它可以被敌人轻易获得。换句话来说，加密方案本身不必保密，唯一需要保密的是通信双方共享的秘密密钥。

Kerckhoffs 的意图是：加密方案应该设计成只要敌手不知道使用的密钥，即使知道方案的组成算法细节，方案依然是安全的。换句话来说，Kerckhoff 原则要求安全性仅依靠于密钥的安全性。

接受 Kerckhoffs 原则有三个主要的理由。①与维护算法的保密性相比，通信各方非常容易维护短小密钥的保密性。共享一个短（如 100bit）串并安全地保存这个串，与共享和安全地保存一个长度大数千倍的程序相比，要容易得多。而且，算法的细节能够泄漏（可能被内部人员）或者通过逆向工程学习到；而当秘密信息是从随机产生的串中选取时，这些是不可能的。②万一密钥暴露了，诚实参与方将非常容易改变密钥而不是替换算法。实际上，良好的安全实践会要求：即使密钥没有暴露，也要经常刷新（替换）密钥。而替换软件程序则麻烦得多。③万一有多对人员（如同一个公司中）需要加密他们的通信，对所有参与方而言，使用同样的算法（或者程序）和不同的密钥，与使用不同的（取决于通信方的）程序相比要容易得多。

现在，Kerckhoffs 原则被理解为：提倡安全性不能建立在对算法的保密上，更提倡要公开

② 全书将使用“:=”表示赋值操作。常用符号的列表见书尾。

这些算法。这全然站在了“通过隐晦产生安全”的对立面，“隐晦产生安全”的思想是通过隐藏密码学算法来提高安全性。公开算法设计说明的“开放密码学设计”的优势在于：

(1) 公开的设计承受公开的钻研和分析，因此可能更加健壮。多年的经验表明：构造良好的密码学方案是非常困难的。因此，如果方案被广泛地（被方案的设计者之外的专家们）研究过，且没有发现弱点，就更相信方案的安全性。

(2) 安全缺陷，如果存在的话，被“正义黑客”所发现（希望能导致系统的修改），而不是被敌对方所发现会更好。

(3) 系统的安全取决于算法的保密性，代码的逆向（或者被工业间谍泄漏）导致严重的安全威胁。相比之下，安全密钥不是代码的一部分，所以不存在逆向工程的弱点。

(4) 公开设计使标准的建立成为可能。

听上去简单明了，但开放的密码学设计这一原则（即 Kerckhoffs 原则）被一次次忽视，导致了灾难性的后果。使用专利算法（即公司密码设计的非标准算法）是非常危险的，只有公开试用和测试的算法才能使用。值得庆幸的是，很多优秀的算法已经标准化，并不受专利保护，所以没有理由到现在还使用其他算法。

攻击场景。下面简要介绍一些基本的攻击类型来归纳加密方案。根据严重程度，包括以下几种攻击类型。

(1) 唯密文攻击（Ciphertext-only attack）：这是最基本的攻击方式，表示敌手只能观察到密文（或者多个密文），并且试图确定相应的明文（或者多个明文）。

(2) 已知明文攻击（Known-plaintext attack）：这里，敌手学习一个或者多个使用相同密钥加密的明文/密文对。目标是确定其他密文对应的明文（这里“其他”是指事先不知道明文的密文）。

(3) 选择明文攻击（Chosen-plaintext attack）：这种攻击中，敌手能够选择明文，并得到相应密文。试图确定其他密文对应的明文。

(4) 选择密文攻击（Chosen-ciphertext attack）：最后一种攻击类型是敌手甚至可以选择密文并得到相应的明文。敌手的目的依然是确定其他密文的明文（其明文不能直接得到）。

前两种攻击类型是被动的，敌手只是获得一些密文（有可能包括相应明文）并且发起攻击。相对而言，后两种攻击是主动的，敌手能够适应性地有选择地请求加密和解密。

上述前两种攻击明显是符合实际的。唯密文攻击在实践中最容易实现，敌手唯一需要做的事就是偷听传输密文的公共通信线路。在已知明文攻击中，假定敌手获得所见密文对应的明文。这是符合实际的，因为不是所有加密的消息都是保密的，至少不是无限期的保密。举一个简单的例子，当开始通信时，通信两方可能通常加密“hello”消息。一个更复杂的例子，加密可用来保证季度盈利的机密直到公开的那一天。该例中，任何偷听并获得密文的人，将在后来获得相应的明文。因此，任何合理的加密方案必须对发起已知明文攻击的敌手保持安全。

后两种主动攻击看上去有点奇怪并需要论证。（什么时候通信方会加密和解密敌手所需要的信息？）详细的讨论将放在对该类攻击进行形式化定义的时候：选择明文攻击在 3.5 节讨论，选择密文攻击在 3.7 节讨论。

不同的加密应用可能需要加密方案能够对付不同的攻击类型。通常并不要求必须使用可抵御“最强”攻击的加密方案，因为它们可能没有可抵御“弱一点”攻击的加密方案效率高。因此，如果后者足够满足应用的要求，则倾向于使用后者。

1.3 古典加密术及其密码分析

在学习“经典密码学”过程中，将回顾一些古典加密术并说明它们是完全不安全的。正如前面提及的，本节的主要目标是：①突出密码学中“临时”方法的弱点，并因而引出后续讨论的现代的严格方法；②演示企图用“简单的方法”达到安全加密是不可能成功的，并给出其原因。过程中将展示一些从这些弱点中学到的密码学的核心原则。

本节（且仅限本节），明文字符使用小写字母表示，密文字符使用大写字母表示。当描述攻击时，常使用 Kerckhoff 原则并假定方案被敌手所知（但是密钥不被敌手所知）。

恺撒（Caesar）加密。有记载的最古老的加密方法是恺撒加密，约公元 110 年在“De Vita Caesarum, Divus Iulius”（“The Lives of the Caesars, The Deified Julius”）中记载：

在他给 Cicero 的信中，以及给他的密友关于私人事务的信中，特别是后者，如果他有什么需要保密的话，他就用加密术来写，即通过改变字母表中字母的次序，使得没有一个词可以读得出来。如果任何人想解密，得到其中的涵义，他必须替换字母表中的第 4 个字母，即用 D 替换 A，依此类推。

也就是说，Julius Caesar 通过移动三个位置的字母进行加密：a 被 D 替换，b 被 E 替换，以此类推。当然，在字母表的最后，字母折返回来，即 x 被 A 替换，y 被 B 替换，z 被 C 替换。举例来说，短消息“begin the attack now”，去掉空格，将被加密成为：

EHJLQWKHDWWDFNQRZ

使得它不可读。

这种加密方法的一个问题是：方法是固定的。因而任何人学习到恺撒是如何加密的，便可以轻易地解密。如果用前面描述的加密语法来解释恺撒加密，密钥生成算法Gen过于简单（也就是什么都没做），即没有密钥。

有趣的是，这种加密方法的变种 ROT-13（移动 13 个位置，而不是 3 个）现在被广泛使用在很多在线论坛中。可以理解这并不提供任何密码学安全，ROT-13 仅仅用在保证文本（比如影评）不可读，除非读者有意识地试图解密。

移位加密以及充分密钥空间原则。恺撒加密的问题是加密方式总是一样的，且没有密钥。移位加密类似于恺撒加密，但是引入了密钥[③]。特别是在移位加密中，密钥k是一个 0 到 25 之间的数。那么加密时，字母移动k个位置，映射到加密语法中，意味着算法Gen输出一个在$\{0,\cdots,25\}$中的随机数k；算法Enc使用密钥k将英文字母组成的明文的每个字母向前移动k个位置（在 z 处折回到 a）；算法Dec使用k将密文的每个字母向后移动k个位置（在 a 处折回到 z）。明文消息空间$\mathcal{M}$定义为所有由英语字母表中字符组成的有限长度字符串（该方案中不含数字、标点，或者其他字符）。

若视字母表为数字$\{0,\cdots,25\}$（而不是英文字符），则可以得到一个更数学化的描述。首先，介绍符号：若a是一个整数且N是一个大于 1 的整数，定义$[a \mod N]$为a被N除后的余数。注意$[a \mod N]$是一个 0 到$N-1$闭区间的整数。我们称映射a到$[a \mod N]$为模N消减，将在第 7 章具体介绍。

使用这种符号，用密钥k来加密明文字符m_i，得到密文字符$[(m_i+k) \mod 26]$，解密密

③ 在有些书中，“移位加密”和“恺撒加密”可以互换。

文字符c_i，则为$[(c_i - k) \mod 26]$。从这一视角，消息空间$\mathcal{M}$定义为任何有限长度整数序列，整数的范围为$\{0, \cdots, 25\}$。

移位加密安全吗？在继续阅读之前，设法解密下面使用密钥k进行移位加密的密文（不破解密钥的值）：

OVDTHUFWVZZPISLRLFZHYLAOLYL

在不知道k的情况下能解密该消息吗？实际上，由于只有 26 个可能的密钥，因而非常容易尝试每个密钥，并观察哪个密钥解密密文后得到的明文“有意义”。这种对加密方案的攻击称为蛮力（brute-force）攻击或者穷举（exhausitive）攻击。很明显任何安全的加密方案必须能够抵御这种蛮力攻击；否则将被完全攻破，无论加密算法有多复杂。于是得到一个简单而重要的原则，叫做“密钥空间充分性原则”：

任何安全的加密方案必须拥有一个能够抵御穷举搜索的密钥空间④。

在当前这个年代，穷举搜索可能使用非常强大的计算机，或者使用分布在世界各地的上千台 PC 机。因而可能的密钥数量必须非常多（至少2^{60}或者2^{70}）。

强调上述原则给出的是安全的必要条件，不是充分条件。后面将看到加密方案虽有非常大的密钥空间，但仍不安全。

单字母替换（substitution）。移位加密映射每个明文字符到一个不同的密文字符，但是在每种情况下映射有相同的移位（该值由密钥决定）。单字母替换的思想是映射每个明文字符到一个不同的密文字符，以一种可任意确定的方式，只要映射是一一对应的以确保能够解密。如果使用英文字母表，密钥空间则由多个字母的置换组成，意味着密钥空间的规模为$26! = 26 \cdot 25 \cdot 24 \cdots 2 \cdot 1$（或大约为$2^{88}$）。例如，密钥

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

其中 a 映射到 X，等等，加密消息tellhimaboutme为GDOOKVCXEFLGCD。针对密钥空间的蛮力攻击，即使使用现在已知的最强大的计算机，也需要比人的一生长得多的时间。但是，这并不是说这种加密方法是安全的。事实上，将会看到，这种加密方法容易攻破，即使它拥有非常大的密钥空间。

假定被加密的是英语文本（即文本是语法正确的书面英语，不仅仅是由字母表的字符构成的文本）。一种可能的对单字母替换加密的攻击是使用英语的统计模式（当然，这种攻击适用于任何语言）。这种加密方法的两种特性将会在攻击中被利用：

(1) 这种加密方法中，每个字符的映射是固定的，因此如果 e 映射到 D，那么每次 e 在明文中出现的时候，都会导致 D 在密文中出现。

(2) 英语（或者其他语言）中单个字母的概率分布是已知的。也就是说，不同英文字母在不同文本中的平均出现频率通常一样。当然，文本越长，频率计算越接近平均值。但是，即使是相对较短的文本（仅有几十个字）已经足够接近平均值的分布。

该攻击将密文的概率分布进行列表，然后与已知的英语字母的概率分布（见图 1.2）进行比较。得到的概率分布是密文中每个字母的频率（即表格记录 A 出现 4 次，B 出现 11 次

④ 实际上该原则仅在消息空间大于密钥空间时正确（参见第 2 章的一个例子，它使用小的密钥空间获得安全性，只要消息空间更小）。实践中，当非常长的消息被同样的密钥加密时，密钥空间必须不怕穷举搜索。

等)。于是，可以根据频率计数来初步猜测映射关系。例如，既然 e 是英语中使用最频繁的字母，将猜测密文中最频繁的字母对应的明文为 e，如此类推。除非密文非常长，某些猜测可能错误。但是，即使对非常短的密文而言，猜测也可能快速解密（特别是使用英语语言的其他知识，如字母 t 和 e，可能出现的字母是 h，以及 u 通常跟在 q 后面)。

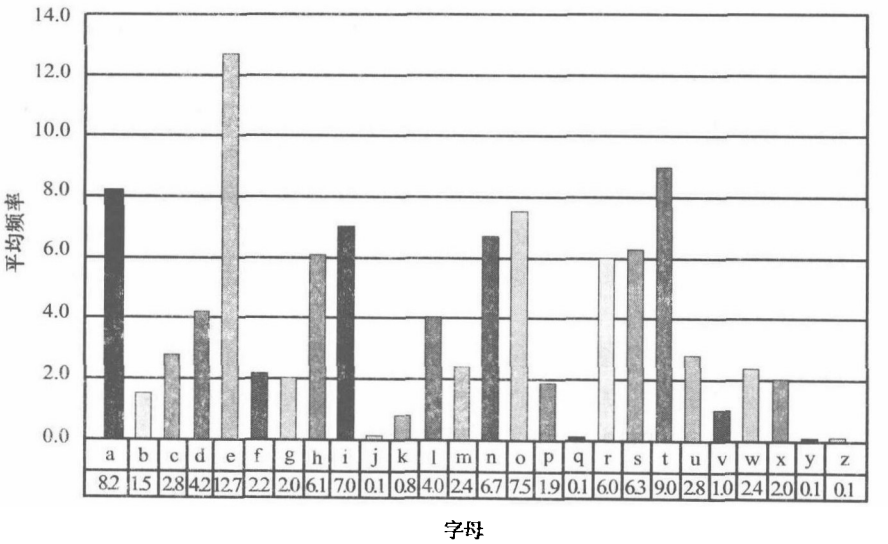


图 1.2　英文文本中字母的平均频率

实际上，单字母替换加密会很快被攻破，这并不令人惊讶，既然基于这种加密术的迷题出现在报纸上（且被人们在早餐咖啡前解决)。建议读者尝试解密下面的消息，这将帮助读者相信实施攻击是多么容易（当然，应该借助于图 1.2 的帮助)：

JGRMQOYGHMVBJWRWQFPWEGFFDQGFPFRKBEEBJIZQQ0CIBZKLFAFGQVFZF
WWEOGWOPFGFHWOLPHLRLOLFDMFGQWBLWBWQ0LKFWBYLBLYLFSFTJGR
MQBOLWJVFPFWQVHQWFFPQOQVFPQOCFPOGFWFJIGFQVHLHLROQVFGWJVF
PFOLFHGQVQVFILEOGQILHQFQGIQVVOSFAFGBWQVHQWIJVMJVFPFWHGFIW
H22RAGBABHZQOCGFHX

结论是：虽然单字母加密有非常大的密钥空间，但它仍然是完全不安全的。

一种对移位加密改进的攻击。可以使用字母频率表改进对移位加密的攻击。特别是前面的攻击需要使用每个可能的密钥解密密文，然后查看哪个密钥给出的明文结果是“有意义的”。这种方法的缺点是很难自动进行，因为对计算机而言查看明文是否“有意义”比较困难。(并不是说不可能，这个当然可以通过包含有效英文单词的字典来查看，这里只是说这种查看是不简单的。）并且，有些情况（后面将给出一种）明文字符符合英文文本的分布规律，但是明文本身不是有效的英文文本，使得问题变得更加困难。

同前所述，用0,…,25表示英文字母。令p_i（$0 < i \leqslant 25$）表示在普通英文文本中第i个字母的概率。对已知的值p_i容易计算

$$\sum_{i=0}^{25} p_i^2 \approx 0.065 \tag{1.1}$$

现在，给定一些密文并令q_i表示第i个字符在密文中的概率（q_i是第i个字符出现的次数除以密文的长度）。如果密钥是k，那么期望对于每个i，q_{i+k}约等于p_i。（使用$i+k$替代烦琐的$[i+k \mod 26]$。）对应地，如果对每个$j \in \{0,\cdots,25\}$计算

$$I_j \stackrel{\text{def}}{=} \sum_{i=0}^{25} p_i \cdot q_{i+j}$$

那么，希望发现$I_k \approx 0.065$，这里k是密钥（然而I_j，$j \neq k$则不为 0.065）。这导致密钥恢复（key-recovery）攻击非常容易自动进行：对于所有j计算I_j，并输出所有I_k接近 0.065 的k。

Vigenère（多字母移位）加密。正如前面描述的，对单字母替换密码进行统计攻击是因为每个字符的映射是固定的。因而，消除这种攻击可通过将相同的明文字符映射到不同的密文字符。这拥有将密文中字符的概率“均匀平滑化”的效果。例如，e 有时候映射到 G，有时候映射到 P，有时候到 Y。这样，密文字符 G、P 和 Y 将很可能没有突出的出现频率，因为其他出现频率较小的字符可能映射到这些字符。因而，计算字符的频率将不提供关于映射的信息。

Vigenere 加密通过依次运用多个移位加密。也就是说，一个短的密码字被选作密钥，明文的加密是通过每个明文字符“加”每个密钥的字符（如同移位加密），必要的时候折回。例如，使用密钥cafe，消息tellhimaboutme的密文如下：

明文: tellhimaboutme
密钥: cafecafecafeca
密文: WFRQKJSFEPAYPF

（密钥没必要为一个英文单词）。这其实和移位加密一样，加密第一个、第五个、第九个等字符和密钥$k=3$的移位加密情况是一样的。加密第二个、第六个、第十个等字符和密钥$k=1$时一样。加密第三个、第七个等和$k=6$一样，加密第四个、第八个等和$k=5$一样。因此，只是多次使用不同密钥的移位加密。注意在上例中l一次映射到R,一次映射到Q。而且，密文字符F有时候从e获得，有时候从a获得。因此，密文中字符的频率“平滑了”，正是需要的。

如果密钥是充分长的字（随机选择），则破解这种加密方法看上去很困难。其实，它被很多人认为是不可攻破的加密方法，虽然在 16 世纪发明，但对该方法的攻击却在几百年后才发现。

破解 Vigenère 加密。攻击的第一个观察是如果密钥的长度已知，则破解任务相对容易。特别地，如果长度为t（有时候叫周期），则密文能被分成t个部分，每个部分被视为使用单个移位加密。也就是说，令$k=k_1,\cdots,k_t$是密钥（每个k_i是字母表中的一个字母），令$c_1,c_2,\cdots$是密文字符。于是，对任意j（$1 \leqslant j \leqslant t$），字符集合

$$c_j, c_{j+t}, c_{j+2t}, \cdots$$

这些都是被密钥为k_j的移位加密方法来加密的。所以，剩下的工作就是对于每个j，决定26个可能的密钥中哪个是正确的密钥。这并不像移位加密的情况那样简单，因为不再可能通过解密“有意义”来猜测密钥的单个字符。而且，同时检查所有j将需要蛮力搜索26^t个可能的密钥（当t大于诸如15时，是计算不可行的）。尽管如此，还是可以使用前面描述的统计方法。即对于与给定密钥（即每个j值）相关联的密文字符的每个集合，可以将每个密文字符的

频率进行制表，然后检查26个可能的移位中哪一个产生“正确的”概率分布。既然这一方法对于每个密钥能够分别进行，所以攻击执行得非常快；所有要做的只是建立t频率表（一张表对应一个字符集合）以及与实际的概率分布进行比较。

一个某种程度上更简单的方法是使用改进的方法攻击移位加密。重提一下这种改进的攻击不依赖于检查明文是否“有意义”，而是仅仅依赖于明文中字符的概率分布。

当密钥长度已知时，上面的任一方法都能给出成功的攻击。剩下的问题是如何确定密钥的长度。

Kasiski 的方法，在 19 世纪中发现的一种解决这个问题的方法。第一步是发现密文中长度为 2 或者 3 的重复模式。这些可能是某种英文中出现非常频繁的 2 字词或者 3 字词。例如，考虑词“the”在英文文本中出现非常频繁。很明显，“the”将映射不同的密文字符，取决于在文本中的位置。但是，如果在同样的相对位置出现两次，则它将映射到密文中的同一个字符。例如，如果出现在位置$t+j$和$2t+i$（$i \neq j$），则每次它将映射到不同字符。但是，如果出现在位置$t+j$和$2t+j$，则它将映射到相同的密文字符。在一个足够长的文本中，“the”很可能映射到相同密文字符。

考虑下面的实例，用密钥beads（加空格是为了清楚）：

明文: the man and the woman retrieved the letter from the post office
密钥: bea dsb ead sbe adsbe adsbeadsb ean sdeads bead sbe adsb eadbea
密文: VMF QTP FOH MJJ XSFCS SIMTNFZXF YIS EIYUIK HWPQ MJJ QSLV TGJKGF

单词the有时映射到VMF，有时映射到MJJ，有时是YIS。但是，它两次映射到MJJ，在足够长的文本中，它可能多次映射到每种可能的结果。Kasiski 观察到这些多次出现的距离（除了巧合情况）是周期长度的倍数。（在上例中，周期长度是 5，而且两次MJJ出现的距离是40，它是周期长度的 8 倍。）因此，所有距离的最大公约数即为周期t或者是t的倍数。

另一种可选的方法称为巧合指数方法，更像是一种算法，所以更容易自动化。重提一下，如果密钥长度是t，则密文字符为：

$$c_1, c_{1+t}, c_{1+2t}, \cdots$$

是使用同样的移位加密。这意味着除了某个移位次序，在这个序列中字符的频率和标准的英文文本中字符的频率应该是一样的。更详细地说，令 q_i 表示第i个英文字符在上述序列中出现的频率（第i个字符的出现次数被序列中的总字符数除）。如果移动是k_1（即密钥中的第一个字符），则期望 q_{i+k_1} 约等于 p_i（对于所有的i），这里 p_i 是标准英文文本中的第i个字符的频率。但是，这意味着序列$p_0, \cdots, p_{25}$即为$q_0, \cdots, q_{25}$移动k_1个位置。因此，希望（见等式(1.1)）：

$$\sum_{i=0}^{25} q_i^2 = \sum_{i=0}^{25} p_i^2 \approx 0.0065$$

这导致一个更好的方法确定密钥的长度t。对于$\tau = 1, 2, \cdots$，观察密文字符序列$c_1, c_{1+\tau}, c_{1+2\tau}, \cdots$，并且对这个序列计算$q_0, \cdots, q_{25}$。于是计算

$$S_\tau \stackrel{\text{def}}{=} \sum_{i=0}^{25} q_i^2$$

当$\tau = t$时，我们希望如上所讨论的能看到$S_\tau \approx 0.065$。另一方面，对于$\tau \neq t$，希望（粗略地说）在序列$c_1, c_{1+\tau}, c_{1+2\tau}, \cdots$中所有字符将大约以等概率出现，所以希望对于所有的$i$，

$q_i \approx 1/26$。在这种情况下，得到

$$S_r \approx \sum_{i=0}^{25} \left(\frac{1}{26}\right)^2 \approx 0.038$$

这与 0.065 有足够的差异。

密文长度和密码分析攻击。在上面对 Vigenère 的攻击中，需要比前面方案更长的密文。例如，如果使用 Kasiski 方法，需要更长的密文来确定周期。而且，统计需要对t个不同的密文部分进行，消息的频率表随着其长度的增长收敛到一个平均值（而且，密文因此需要近似t倍长于单字母替换加密的情况）。类似地，攻击单字母替换加密方法与攻击移位密码的方法（可攻击仅仅由单个单词组成的消息）相比，需要更长的密文。这种现象不是偶然的，它和加密方案中密钥空间的大小相关。

唯密文攻击与已知明文攻击的比较。上面描述的攻击都是唯密文攻击（这是实践中最简单实施的攻击方式）。如果敌手能够执行已知明文攻击，则所有以上加密方法都会攻破；举例留作练习。

结论和讨论。我们仅仅表述了少量的历史上的加密方法。除了了解历史以外，我们的目的是列举一些重要的关于密码设计的经验教训。简要地说，这些经验是：

(1) 充分密钥空间原则：假设加密充分长的消息，一个安全的加密方案必须有不能在合理时间内被穷举的密钥空间。但是，一个大的密钥空间并不就意味着安全（例如，单字母替换密码有大的密钥空间但仍很容易破解）。因此，大的密钥空间只是必要条件，而不是充分条件。

(2) 设计安全的加密方法是一个困难的任务：Vigenère 加密保持很长一段时间未被攻破，部分归因于它假定的复杂性。更多的复杂方案被使用，例如德国的 Enigma。但是，这种复杂性并不意味着安全，而且所有历史上的加密术都能被完全攻破。通常，设计安全的加密方案是困难的，而且这种设计要留给专家。

经典加密方案是有趣的，无论是方法，还是对密码学和密码分析学对世界历史（例如，第一、二次世界大战）的影响。这里，仅仅试图给出一些基本方法的介绍，关注的是现代密码学能从这些方法中学到什么。

1.4 现代密码学的基本原则

前面一节给出古典密码学的一些介绍。公平地说，从历史角度看密码学更多的是一门艺术，而不是科学：方案以一种临时的方式设计，然后根据臆断的复杂性或者聪明才智来评价。不幸的是，正如已经看到的，所有这些方案（无论多么聪明）最终都被破解了。

现代密码学，基于更严格和更科学的基础，给出打破“构建方案，攻破方案”这一循环反复怪圈的希望。本节描述那些能够将现代密码学从经典密码学区分开来的主要原则和范例。这里明确提出三个主要原则：

(1) 原则 1: 解决任何密码学问题的第一步是公式化的表述严格且精确的安全定义。

(2) 原则 2: 当密码学构造方案的安全性依赖于某个未被证明的假设时，这种假设必须精确地陈述，而且，所假设的要尽可能地少。

(3) 原则 3: 密码学构造方案应当伴随有严格的安全证明，跟随符合原则 1 的安全定义，以及与原则 2 陈述的假设有关（如果假设是需要的）。

我们现在更深入地讨论上述每个原则。

1.4.1 原则 1——形成精确的定义

现代密码学的一个关键的贡献在于：形式化的安全定义是设计、使用或者研究任何密码学原语或者协议的基本先决条件。下面依次解释这三个方面。

(1) 对设计的重要性：假定对构造安全的加密方案感兴趣。如果没有完全清晰地理解想达到什么，怎么可能知道是否（或何时）达到。在头脑中拥有精确的定义使得更好地定位设计的努力方向，以及更好地评价构造方案的质量，从而改进最终的构造方案。特别需要指出的是，要首先定义需要什么，然后才开始设计阶段，而不是在设计结束的时候，事后定义达到了什么。后者的风险在于：当设计者的耐心没有了（而不是当目标达到的时候），设计阶段就结束了；或者可能导致构造的方案达到的多于需要的，从而效率上不如更好的方案。

(2) 对使用的重要性：假定想在某些更大的系统中使用加密方案。如何知道使用哪个加密方案？如果有多个候选方案，如何辨别哪个方案对应用已经足够？拥有给定方案的精确安全定义，以及与形式化的安全假设相关的安全证明（在原则 2 和原则 3 中讨论的），可以回答这些问题。特别是定义系统所需要的安全（见上面第(1)点），然后验证给定加密方案满足的安全是否达到所需要的安全。另外，明确说明需要的定义，并寻找满足这些定义的加密方案。注意，选择“最安全的”方案可能是不明智的，这是因为更弱的安全定义可能对应用来说已经足够，而且，也许能够使用更加高效的方案。

(3) 对研究的重要性：给定两种加密方案，如何比较它们？没有任何安全定义，唯一可比较的是效率，但是仅仅考虑效率是不够的，因为高效率的方案可能是完全不安全的，因而不可用。方案可以达到的安全级别的精确定义提供比较的另一点。如果两个方案是等效率的，但是第一个比第二个满足更强的安全定义，则选第一个[5]。在安全和效率之间可能有所权衡，但是，至少精确的定义能帮助理解权衡的细节。

当然，精确的定义同时使得严格的证明成为可能（将在原则 3 处讨论），但是即使不考虑这个，上述三个原因依然成立。

一个错误的想法是形式化的定义是不必要的，因为“我们有对什么是安全的直觉感受”。对初学者，不同的人对什么是安全有不同的直觉。甚至一个人对安全有多种直觉，取决于应用的环境。例如，第 3 章将研究四种不同的对称密钥加密的安全定义，每种方法应用在不同的场景中。在任何情况下，形式化的定义对传递所谓的“直觉想法”是必要的。

例子：安全加密。认为形式化定义是简单的也是一个错误。例如，如何形式化地描述对称密钥加密方案所需要的安全概念？（读者可能想停下来思考一下再继续读下去。）我们曾经多次问学生安全加密如何来定义，回答依次如下：

(1) 回答 1：如果没有敌手能够在给定密文的情况下发现秘密的密钥，则加密方案是安全的。这一定义不妥。加密的目标是保护加密的消息，而密钥仅仅是达到这一目的的手段。用一个看似荒唐的例子，考虑一种加密方案忽略密钥，直接输出明文。明显地，没有敌手能够发现密钥。但是，这种方案明显没有安全可言[6]。

(2) 回答 2：当没有敌手能够发现密文对应的明文，则加密方案是安全的。这个定义看上

---

⑤ 当然，事情很少有这么简单。

⑥ 你可能会说：“但是，这不是我想表达的意思！”是的，这就是问题的关键：通常形式化地描述想表达的想法不是一件容易的事情。

去比前面那个更好，甚至可以从某些密码学教科书中见到。但是，经过片刻思考，会发现它同样是不尽人意的。例如，根据这一定义，一个加密方案透露 90%的明文，将仍然被认为是安全的，只要很难发现剩下的 10%。但是，在大多数加密应用中，这明显不能接受。例如，雇佣合同中大部分为普通文本，仅仅薪水部分是机密的；如果薪水在被泄漏的明文的 90%部分中，则加密没有得到任何好处。

如果你发现上述反例很愚昧，请再次参见脚注⑥。关键点依旧是：如果陈述的定义不是想表达的，则方案所证明的安全没有真正提供需要的安全保护级别。（这个例子说明了为什么精确定义是重要的。）

(3) 回答 3：如果没有敌手能够确定密文所对应的明文中的任何字符，则加密方案是安全的。这看上去已经像是一个良好的定义。但是，其他微妙的地方出现了。回到雇佣合同的例子，不可能确定任何薪水值或者甚至是任何其中的数字。但是，如果泄露了加密的薪水是多于还是少于每年 10 万美元，加密方案还是安全的吗？很明显不是。这导致下一个回答。

(4) 回答 4：如果没有敌手能够从密文中确定关于明文的任何有意义的信息，则加密方案是安全的。这已经接近真正的定义。但是，它缺少一个方面：没有定义什么信息是“有意义的”。不同的信息可能在不同的应用中意义不同。这导致一个非常重要的原则，它关系到密码学原语的安全定义：安全定义应当对所有潜在的应用是充分的。这是基本要求，因为没有人知道将来会有什么应用。而且，其实现通常是通用密码库的一部分，可用于多个不同的场合和应用中。理想情况下安全应当在所有可能的应用中都得到保证。

(5) 最终回答：如果没有敌手能够从密文中计算任何关于明文的函数，则加密方案是安全的。这提供一个非常强的保证，而且进行了恰当的形式化陈述，被认为是“正确的”对加密的安全定义。甚至这里，还需要考虑攻击模型问题，以及应当如何定义。

虽然找到了安全加密的正确要求，从概念上来说，还需要将这些要求通过数学语言描述和进行形式化的描述，这不是一件简单的事情（将在第 2 章和第 3 章详细讨论）。

正如在“最终回答”中提到的，形式化的定义必须能明确地定义攻击模型：也就是说，是唯密文攻击还是选择明文攻击。这里列举的是通用的原则。特别地，为了完整地定义密码学任务的安全，有两个不同的问题必须明确。第一个是什么是“攻破”，第二个是敌手的能力是如何假定的。前面提过，加密方案被攻破是指：敌手从密文中学习到关于明文的某些函数。敌手的能力与其假定拥有的行为能力相关，包括计算能力。前者指敌手是否仅仅假定只能偷听加密的消息（即唯密文攻击），还是敌手可以主动请求任意感兴趣的明文的密文（即选择明文攻击）。后者是指敌手的计算能力。在本书中，除第 2 章外，都希望能抵御“有效”敌手的攻击，即任何多项式时间运行的敌手。（关于此问题的完整讨论见 3.2.1 节。此后，“有效的”即指能在一个生命期内完成。“可行的”是更精确的术语。）转变成实际的术语，即要求安全性达到任何敌手使用超级计算机需要计算几十年的时间。

总之，任何安全定义将使用如下的通用形式：

如果特定的敌手不能完成特定的攻破，则对给定任务的一个密码学方案是安全的。

需要强调这一定义不对敌手的策略进行任何假设。这是一个重要的区别：假设敌手的能力（例如，能够进行选择明文攻击而不是唯密文攻击），但是不对敌手如何使用其能力进行假设。这个称为“任意敌手原则”，安全必须确保抵御任意拥有特定计算能力的敌手。这一原则是重要的，因为敌手在攻击中将采用的策略是不可能预先知道的（且历史已经证明这种企图

注定失败）。

数学世界以及现实世界。安全定义本质上提供了对现实世界的问题数学描述。如果数学定义不能合适地对现实世界建模，则定义可能是无用的。例如，如果假定的敌手能力非常弱（但实际上敌手有更强的能力），或者密码方案的攻破来自不可预见的真实攻击（如同前面对加密方案的回答），那么还是没有得到“真正的安全”，即使是使用了“数学上安全的”构造方案。简言之，安全的定义必须准确地对现实世界建模，以便使用数学语言描述安全承诺。

事实上，通常被广泛接受的定义在某些新应用中是不合适的。作为一个明显的例子，有些加密方案被证明为安全的（相对一些前面讨论的定义而言），且在智能卡中实现了。归因于智能卡的物理属性，敌手可能监控到在加密方案运行时智能卡的能量使用情况（例如，使用的能量随时间的变化情况），而且，这些信息可以用来确定密钥。这既不是安全定义的错误，也不是满足安全定义的方案的安全证明的错误；问题出在安全定义和现实世界的实现（方案在智能卡上的实现）的不匹配。

这并不是说定义（或者证明）是无用的，定义以及满足定义的方案可能在某些情况下仍然是合适的，如当加密在终端主机上进行，它的能量使用情况不可能被敌手监控到。而且，达到智能卡安全加密的方案需要进一步提炼定义，以便于将能量分析包含在考虑中。或者，开发可能对抗能量分析的硬件对策，使得原来的定义（以及原来的方案）对智能卡来说还是合适的。关键点是：使用定义你至少知道你站在哪里，即使定义没有准确的建模应用该方案的特定场合。相比而言，没有定义时连什么错了都不清楚。

在数学模型和需要建模的现实之间存在不匹配的可能，它不是只出现在密码学中，而是出现在整个科学中。举一个来自计算机科学的例子，考虑数学证明的含义，存在一些定义清楚的问题但计算机不能解决⑦。这立刻得到一个问题：“计算机能解决的问题”是什么意思。特别地，只有当对“什么是计算机”的数学定义存在的时候，才能提供数学证明。问题是计算是一个真实世界的过程，有很多不同的计算方式。为了真正相信“不可解的问题”是真正不可解的，必须确信对计算的数学定义的确反映了计算的真实世界过程。我们并不知道何时能够达到这个目标？

这个内在的困难被阿兰图灵注意到，他研究了计算机能解决什么问题和不能解决什么问题。他的论文[140]中写到（方括号中的文本替换了原文使得读起来更清楚）：

> 没有企图表明[我们定义的可被计算机解决的问题]包括自然被认为可以计算的[那些问题]。所有可给出的论据其局限在于基本上求助于直觉。真正的问题是“什么是[计算]中可能执行的过程？”
>
> 我将使用三种论据。
>
> (1) 直觉。
>
> (2) 两种定义的等价证明（如果新定义更具直觉上的吸引力）。
>
> (3) 给出一大类[可使用给定计算定义来解决的]问题例子。

在某种情况下，图灵面对的困难和密码学家面对的是一样的。他发展了一种数学计算模型，但是需要确信模型是合适的。同样地，密码学家定义了安全概念，需要确信其定义意味着对现实世界的有意义的安全保证。和图灵一样，他们可能使用下面的工具来证实：

---

⑦ 学习过计算理论课程读者将熟悉这类问题是存在的这一事实（如停机问题）。

(1) 求助于直觉：这是第一个工具，当思考一个新安全定义的时候，观察这个定义是否意味着直觉上希望拥有的安全属性。这是最小要求，因为初始直觉通常导致非常弱的安全想法（正如在对加密的讨论中看到的）。

(2) 等价性证明：通常论证新的安全定义和一个旧的（或更强）的、更熟悉的或者直觉上更有吸引力的定义等价。

(3) 例子：一个证明安全定义是充分、有用的方法是：显示熟悉的、不同的现实世界攻击可被定义排除掉。

此外，依赖时间的检验以及随着时间的推移，研究者和实践者的钻研和调查将会证明定义的合理性。

1.4.2 原则 2——精确假设的依赖

大部分现代密码学的构造方案不可能证明为无条件安全。实际上，这类证明需要解决计算机复杂性理论中的难题，这不像是现在能办到的。这种不幸的状态导致一种有特色的结果：安全性依赖于某种假设。第二种现代密码学的原则是：假设必须被精确地陈述。这有三个原因：

(1) 假设的正确性：假设是没有证明的陈述，但是据推测是正确的。为了强调某些假设的可信程度，有必要对假设进行研究。假设检查得越仔细，且测试后未见反例，则假设的可信程度就越高。此外，对假设的研究能提供其正确性的支持证据，这些证据可从其他一些被广泛相信的假设推出。如果所依赖的假设没有精确的陈述和展现，它不可能被研究和（潜在性的）被驳斥。因此，提高假设的可信程度的一个先决条件是：拥有对被假设内容的精确陈述。

(2) 对多个方案的比较：通常在密码学中，可能面对两个方案，这两个方案都是可证明满足某些安全定义，但是基于不同的安全假设。假设两个方案是效率相等的，应该倾向于哪个方案？如果第一个方案基于的假设弱于第二个（即第二个假设推出第一个假设），那么第一个方案更可取，因为有可能发现第二个假设是错的，而第一个假设保持正确。如果被两个方案使用的假设是不可比的，那么通常的规则是选择那个其假设被更彻底地研究过的方案，或者假设更加简单的方案（原因在前面段落已经陈述）。

(3) 对安全证明的帮助：前面曾经提过，将对原则 3 进行更深入地讨论。现代密码学构造方法的介绍会伴随着安全证明。若方案的安全性不能被无条件地证明，即必须依赖某个假设，那么对“如果假设是正确的，则构造方案是安全的”的数学证明必须建立在精确的假设陈述之上。

通常有人假设构造方案本身是安全的。若安全性是明确定义的，则精确的假设也是明确定义的（构造方案的安全性证明是容易的）。不能这样做有几个原因：首先，如前所述，测试了多年的假设比一个新假设更可取，新的假设仅仅是为了证明给定构造方案的安全性而引入的。其次，通常倾向于陈述更简单的假设，因为这种假设更容易研究和反驳。所以比方说，某些数学问题难解决的假设是更容易学习和使用的，比假设一个加密方案满足复杂的（可能是不太自然的）安全定义要好。当简单的假设被长时间研究并且没有发现反驳意见，就会更自信地说它是正确的。另一个依靠“低层次”假设（而不是假设某个构造方案是安全的）的优势是：低层次假设的特点是能在多个构造方案中共享。如果特定的假设实例最终是错误的，（在任何基于该假设的高层次构造方案中）它可以简单地被不同的假设实例替换掉。

上述方法贯穿本书。例如，第 3 章和第 4 章显示如何（用多种方法）达到安全通信，假

定一个称为“伪随机函数”的原语存在。在这些章节中，完全没有说明如何构造这种原语。第 5 章讨论如何在实践中创建伪随机函数，并且在第 6 章中展示伪随机函数甚至可以从低层次原语构造。

1.4.3 原则 3——严格的安全证明

前面讨论的两个原则很自然地导致目前这个。现代密码学强调（对所建议方案的）严格安全证明的重要性。使用准确的定义和精确的假设，意味着这种安全证明是可能的。但是，需要证明的主要原因是构造方案和协议的安全性不可能用检查软件的方法来检查。例如，加密和解密“可行”，且密文看上去是乱码，并不意味着老练的敌手不能攻破方案。没有证明不存在特定能力的敌手可以攻破方案，则只好通过直觉。经验表明：密码学和计算机安全中依靠直觉将是灾难性的。有数不清的例子表明：没有证明的方案被攻破，有时候很快，有时候是方案提出或者部署后几年。

另一个安全证明如此重要的原因是与潜在的损害相关联的，如果使用不安全的系统，则导致损害。虽然软件缺陷有时候代价很高，但是当银行的加密方案或者认证机制被攻破后，其潜在的损害将是巨大的。最后，虽然软件中存在很多缺陷，但软件基本上还可用，因为通常用户不会试图让他们的软件崩溃。相比而言，攻击者使用相当复杂和诡异的方法（利用构造方案中的特殊属性）来攻击安全方案，有攻破系统的明确目标。所以，虽然正确性证明通常在计算机科学中是理想的，但是在密码学和计算机安全领域却是绝对必须的。需要强调上述观察不是猜测，而是多年来经验所证明的结论。

规约方法。大部分现代密码学中的证明使用所谓的规约方法。给定一个如下形式的定理

“若给定假设 X 是正确的，根据给定的定义，构造方案 Y 是安全的”。

证明通常展示：如何将假设 X 规约到攻破构造方案 Y。具体来说，证明中通常（通过构造的论据）展示：如何使用敌手攻破构造方案 Y 的方法导致一个与假设 X 的冲突。我们将在 3.1.3 节详细介绍。

总结——严格方法与临时方法的比较。以上三种原则组成了严格的现代密码学方法，与经典密码学中的临时方法不同。临时方法可能缺少上面三种原则中的某一个，也常常全部忽视三个。不幸的是，那些希望获得“快速和直接”解决方案的人（或者没有意识到严格方法的人），仍然在设计和应用临时方案。希望这本书帮助人们意识到严格方法的重要性，并且在开发新的、数学上安全的方案时取得成功。

参考文献和扩展阅读材料介绍

本章研究了少量已知的历史上的加密术。读者如果想了解更多历史和数学方面的知识，建议阅读 Stinson 的文献[138]，或者 Trappe 与 Washington 的文献[139]。这些方案在历史（特别是在战争史）中扮演着重要的角色（在 Kahn[81]的书中有介绍）。

讨论古典非严格方法（以古典加密术为例）与给予精确定义和证明的严格方法之间的区别。Shannon[127]是第一个采用后一种方法的人。现代密码学，基于（可计算的）假设和定义、证明，是从 Goldwasser 和 Micali[69]的开创性论文开始的。我们将在第 3 章学习。现代密码学方法的全面论述可参见 Goldreich 的著作《Foundations of Cryptography》[64,65]。本书的论

述在很多地方受到这本书的影响，特别是第 6 章。

练　习

1.1 请解密在“单字母替换”一节中最后部分提供的那段密文。

1.2 提供对一个Gen、Enc和Dec算法的形式化的定义，对单字母替换加密和 Vigenère 加密。

1.3 考虑 Vigenère 加密的改进版本，即在使用多个移位加密的地方，使用多个单字母替换加密。也就是说，密钥由t个字母的随机置换组成，明文字符在位置$i, t+i, 2t+i$等处使用第i个置换加密。说明如何攻破这个版本的加密方法。

1.4 若企图阻止对 Vigenère 加密方法的 Kasiski 攻击，提出了下面的修改建议。给定周期t，明文分解成多个块，每个块的大小为t。回顾一下：在每个块中，Vigenère 加密方法是通过使用第i个密钥（使用移位加密）来加密第i个字符。令密钥为$k_1, \cdots, k_t$，这意味着在每个块中的第i个字符是通过加上k_i模 26 来加密的。在这个建议的修改方案中：加密第j个块中的i个字符通过加上$k_i + j$模 26 来加密。

(1) 说明这种加密可以完成。

(2) 描述上面修改方案对 Kasiski 攻击的影响。

(3) 设计一种替代方法确定这个方案的周期。

1.5 说明移位加密、替换加密、Vigenère 加密都很容易被已知明文攻击攻破。对这些加密方法而言，需要多少明文就可以发现整个密钥？

1.6 说明移位加密、替换加密、Vigenère 加密都很容易被选择明文攻击攻破。为了完全发现密钥，需要多少明文被加密？和上一个问题进行比较。

第 2 章　完善保密加密

上一章介绍了一些古典加密方案以及如何用很少的计算量去完全破解这些方案。本章讨论另外一个极端，研究可证明安全的加密方案，甚至可抵御有无限计算能力的攻击者，这种方案称为完善保密加密。本章介绍在什么条件下才能达到完善保密加密，并解释其原因。

从某种意义上说，本章内容属于“经典密码学”多过“现代密码学”。除了这里介绍的内容是在密码学革命（20 世纪 70 年代中期和 80 年代）之前形成的之外，本章中研究的构造方法仅仅依靠 1.4 节中的第一个和第三个原则。也就是说，给出精确的数学定义和严格的证明，不必依赖任何未经证明的假设。这显然是有利的。但是，这种方法有其内在的局限性。因此，除了作为一个良好的基础来理解现代密码学的原则之外，本章的结论也证实了所采用的三个前述原则。

本章假定读者熟悉基本的概率知识，相关概念参见附录 A.3。

2.1　定义和基本属性

首先简要回顾一下上一章中介绍的一些语法。一个加密方案可由三个算法来定义，Gen、Enc和Dec，以及明文空间$\mathcal{M}$的说明，$|\mathcal{M}| > 1$[①]。密钥产生算法Gen是一种概率算法输出密钥$\mathcal{K}$，$\mathcal{K}$根据一定的概率分布来选取。用$\mathcal{K}$表示密钥空间，即由Gen输出的所有可能的密钥集合，而且$\mathcal{K}$必须是有穷的。加密算法Enc输入一个密钥$k \in \mathcal{K}$和一个明文$m \in \mathcal{M}$，然后输出一个密文c，我们定义它为$\mathsf{Enc}_k(m)$。加密算法可能是概率的，以至于$\mathsf{Enc}_k(m)$多次运行时输出不同的密文。为强调这一点，用$c \leftarrow \mathsf{Enc}_k(m)$来表示明文$m$用密钥$k$加密得到密文$c$的概率过程。（如果Enc是确定的，记为$c := \mathsf{Enc}_k(m)$加以强调。）对于所有可能的$k \in \mathcal{K}$和$m \in \mathcal{M}$，用$\mathcal{C}$表示所有可能的由$\mathsf{Enc}_k(m)$输出的密文集合。（以及所有随机选择的Enc，如果它是随机的话。）解密算法Dec通过输入密钥$k \in \mathcal{K}$和密文$c \in \mathcal{C}$，输出明文$m \in \mathcal{M}$。本书假设加密方案都是完全正确的，也就是说所有的$k \in \mathcal{K}$、$m \in \mathcal{M}$和任何由$\mathsf{Enc}_k(m)$输出的密文c，$\mathsf{Dec}_k(c) = m$的概率为 1。不失一般性，这意味着可以假设Dec是确定的（因为$\mathsf{Dec}_k(c)$每次运行时的输出必须相同）。用$m := \mathsf{Dec}_k(c)$来表示用密钥k解密密文c的过程。

在下面的定义和定理中，提及$\mathcal{K}$，$\mathcal{M}$和$\mathcal{C}$上的概率分布。$\mathcal{K}$的分布是由运行Gen并取输出而得。（正如前面提到的，Gen几乎总是从$\mathcal{K}$中均匀地选择一个密钥然后输出。通常这一假设不失一般性。）对于$k \in \mathcal{K}$，用$\Pr[K = k]$来表示Gen输出k的概率。（正式地，K是用来表示密钥值的一个随机变量。）类似地，对于$m \in \mathcal{M}$，用$\Pr[M = m]$来表示明文为m的概率。事实上，明文是根据一些分布进行选择的（而不是固定的），这一事实意味着，至少从敌手的观点上，不同的消息有不同的发送概率。（若敌手知道发送的是什么消息，则不必解密什么，通信双方也不必使用加密。）举一个例子，敌手可能知道加密的明文是“attack tomorrow”或者“don't attack”

① 若$|\mathcal{M}| = 1$，则只有一个消息，通信无意义，更不用说加密了。

中的一个。更进一步，敌手可能知道（通过其他方式）明文攻击命令的概率是 0.7，不攻击命令的概率是 0.3。此时，$\Pr[M = \text{attack tomorrow}] = 0.7$，$\Pr[M = \text{don't attack}] = 0.3$。

$\mathcal{K}$和$\mathcal{M}$的分布是独立的，也就是说密钥和明文是独立选择的。这是因为在知道明文前密钥已被选定（如通信双方共享）。此外，$\mathcal{K}$的分布是被加密方案本身确定（因为它由Gen确定），但是$\mathcal{M}$的分布可能随着使用加密方案的通信方而变化。

对于$c \in \mathcal{C}$，用$\Pr[C = c]$表示密文为c的概率。对于给定的加密算法Enc，$\mathcal{C}$上的分布完全取决于$\mathcal{K}$上和$\mathcal{M}$上的分布。

定义。现在定义完善保密加密这个概念。直观上，想象一个敌手知道$\mathcal{M}$的概率分布；也就是说，敌手知道将要发送的不同消息的可能性（如上面的例子）。敌手随后观察到一些已经由一方发送给另外一方的密文。理想情况下，观察这些密文应该对敌手的知识没有任何影响；换句话说，m已发送的后验（posteriori）概率（给定所观察到的密文）应该与m将被发送的先验（priori）概率相同。任何$m \in \mathcal{M}$都满足上述要求。此外，即使敌手有无限计算能力，上述要求依然满足。这意味着，密文没有泄漏任何明文信息，因此，敌手截获一个密文不能得到任何关于明文的信息。正式的描述：

定义 2.1 明文空间为$\mathcal{M}$的加密方案(Gen, Enc, Dec)是完善保密加密，若对$\mathcal{M}$上任意的概率分布，任何明文$m \in \mathcal{M}$、任何密文$c \in \mathcal{C}$且$\Pr[C = c] > 0$，有

$$\Pr[M = m | C = c] = \Pr[M = m]$$

（条件$\Pr[C = c] > 0$是从技术上避免出现零概率事件。）另外一种对定义 2.1 的解释是：当明文和密文上的分布是独立的，方案才是完善保密加密。

简化约定。本章只考虑$\mathcal{M}$和$\mathcal{C}$上的概率分布中，任何$m \in \mathcal{M}$和$c \in \mathcal{C}$的概率是非零的情况[②]，这大大简化了表述，因为常常需要被$\Pr[M = m]$或者$\Pr[C = c]$除，如果它们可能为零则会出现分母为零的问题。同样，在定义 2.1 中有时需要用到$C = c$和$M = m$的条件。如果这两个事件概率为零也将导致同样问题。得益于这种简化约定，从现在开始，不再特别声明$\Pr[M = m] > 0$或$\Pr[C = c] > 0$，即便确实需要如此。

值得强调的是，该约定仅仅是为了简化说明，并不是一种基本的限制。特别值得一提的是，所有证明的定理同样适用于$\mathcal{M}$和$\mathcal{C}$上任意概率分布的情形（可以给一些明文或密文的概率赋值为 0）。参见练习 2.6。

等价公式。下面的引理给出了定义 2.1 的一个等价公式。

引理 2.2 明文空间为$\mathcal{M}$的加密方案(Gen, Enc, Dec)是完善保密加密，当且仅当对于任意$\mathcal{M}$上的概率分布，任意一个明文$m \in \mathcal{M}$，和任意一个密文$c \in \mathcal{C}$，有

$$\Pr[C = c \mid M = m] = \Pr[C = c]$$

证明（充分条件）选定一个$\mathcal{M}$上的分布，对于任意的$m \in \mathcal{M}$和$c \in \mathcal{C}$，有

$$\Pr[C = c \mid M = m] = \Pr[C = c]$$

② 对$k \in \mathcal{K}$而言这通常是满足的，因为$\mathcal{K}$是由Gen输出的概率非零的密钥的集合。

在等式两边同时乘以$\Pr[M=m]/\Pr[C=c]$，得

$$\frac{\Pr[C=c \mid M=m]\cdot\Pr[M=m]}{\Pr[C=c]}=\Pr[M=m]$$

应用贝叶斯（Bayes）定理（参见定理 A.8），等式左边等于$\Pr[M=m \mid C=c]$。因此,$\Pr[M=m \mid C=c]=\Pr[M=m]$,该加密方案是完善保密加密方案。

必要条件的证明留做练习。 ■

注意：上面的证明中假设$m\in\mathcal{M}$和$c\in\mathcal{C}$出现的概率都为非零概率（根据前述简化约定），因此$\Pr[C=c]$和$\Pr[M=m]$都可以做分母。

完美不可区分性（Perfect indistinguishability）。现在应用引理 2.2 来得到一个完善保密加密的等价陈述：$\mathcal{C}$的概率分布独立于明文。也就是，用$\mathcal{C}(m)$表示加密$m\in\mathcal{M}$时的密文分布（密文分布取决于密钥的选择，以及加密算法的随机性，如果算法是概率的）。然后要求每一个$m_0,m_1\in\mathcal{M}$，$\mathcal{C}(m_0)$和$\mathcal{C}(m_1)$的分布是相同的。这刚好以另外一种方式说明：密文中不包含任何明文信息。我们称此为完美不可区分性，因为它意味着不可能区分m_1的密文和m_0的密文（因为在两种情况下，密文分布是一样的）。

引理 2.3 明文空间为$\mathcal{M}$的加密方案(Gen, Enc, Dec)是完善保密加密，当且仅当对于任意$\mathcal{M}$上的概率分布，每个$m_0,m_1\in\mathcal{M}$，以及每个$c\in\mathcal{C}$，有

$$\Pr[C=c \mid M=m_0]=\Pr[C=c \mid M=m_1]$$

证明 假设此加密方案是完善保密加密，并且选定明文$m_0,m_1\in\mathcal{M}$和密文$c\in\mathcal{C}$。

由引理 2.2 可得：

$$\Pr[C=c \mid M=m_0]=\Pr[C=c]=\Pr[C=c \mid M=m_1]$$

由此完成必要条件的证明。

下面证明充分性：假设对任意$\mathcal{M}$上的概率分布，每个$m_0,m_1\in\mathcal{M}$和每个$c\in\mathcal{C}$，满足$\Pr[C=c \mid M=m_0]=\Pr[C=c \mid M=m_1]$。选定某个$\mathcal{M}$上的分布，对任意的$m_0\in\mathcal{M}$与$c\in\mathcal{C}$，定义$p\stackrel{\text{def}}{=}\Pr[C=c \mid M=m_0]$，由于对所有$m$，有$\Pr[C=c \mid M=m]=\Pr[C=c \mid M=m_0]=p$，于是有

$$\begin{aligned}\Pr[C=c]&=\sum_{m\in\mathcal{M}}\Pr[C=c \mid M=m]\cdot\Pr[M=m]\\&=\sum_{m\in\mathcal{M}}p\cdot\Pr[M=m]\\&=p\cdot\sum_{m\in\mathcal{M}}\Pr[M=m]\\&=p\\&=\Pr[C=c \mid M=m_0]\end{aligned}$$

因为m_0是任意选择的，所以对所有$c\in\mathcal{C}$和$m\in\mathcal{M}$，都有$\Pr[C=c]=\Pr[C=c \mid M=m]$成立。由引理 2.2，可以得出此加密方案是完善保密的。

■

敌手不可区分性（Adversarial indistinguishability）：本节介绍完善保密加密的另一个等

价定义。这个定义基于一个涉及敌手$\mathcal{A}$的实验，并且形式化定义$\mathcal{A}$不可达到的能力，即$\mathcal{A}$不能区分出密文是来自哪个明文的加密。因而叫做**敌手不可区分性**。这个定义将作为起点引出下一章中计算安全的概念。纵观全书，将经常用实验来定义安全性。这个“实验”本质上是敌手和虚构的测试者之间的游戏，敌手试图破解一个密码方案，而测试者希望知道敌手是否成功。

定义一个实验$\mathsf{PrivK}^{\mathsf{eav}}$，因为实验只考虑加密的秘密密钥和窃听敌手（称敌手为窃听者是因其只能获得密文c并且试图从中确定一些有关明文的信息）。这个实验面向任意加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$定义，明文空间为$\mathcal{M}$，任意敌手为$\mathcal{A}$。用$\mathsf{PrivK}^{\mathsf{eav}}$表示一个给定$\Pi$和$\mathcal{A}$的实验。实验定义如下：

窃听不可区分实验 $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$:

(1) 敌手$\mathcal{A}$输出一对信息$m_0,m_1\in\mathcal{M}$。

(2) 由Gen产生一个随机密钥k，并且从$\{0,1\}$中随机选择一个比特b（$b\leftarrow\{0,1\}$）。（选择由虚构实体完成，该实体与$\mathcal{A}$一起运行实验。）然后，一个密文$c\leftarrow\mathsf{Enc}_k(m_b)$被计算出来并交给$\mathcal{A}$。

(3) $\mathcal{A}$输出一个比特b'。

(4) 如果$b'=b$，定义实验的输出为 1，否则为 0。如果输出为 1，用$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}=1$表示，此时称$\mathcal{A}$成功。

应该认为敌手$\mathcal{A}$会设法去猜测实验中选定的值b，当猜测值b'正确时$\mathcal{A}$就取得成功。可以观察到，$\mathcal{A}$在实验中仅仅通过随机猜测b'，便总是能以 1/2 的概率取得成功。问题是$\mathcal{A}$能否使猜测成功的概率更大？于是给出完善保密加密的另一种定义：如果没有敌手$\mathcal{A}$能以大于 1/2 的概率成功，则这种加密方案是完善保密加密。值得强调的是，本章中敌手$\mathcal{A}$的计算能力是没有限制的。

定义 2.4 明文空间为$\mathcal{M}$的加密方案$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$为完善保密加密，当对于所有敌手都满足

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}=1]=\frac{1}{2}$$

下面的命题说明定义 2.4 与定义 2.1 是等价的，命题的证明留做练习。

命题 2.5 明文空间为$\mathcal{M}$的加密方案$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$，当且仅当其为根据定义 2.1 的完善保密加密，则它是根据定义 2.4 的完善保密加密。

2.2 一次一密（Vernam 加密）

1917 年，Vernam 创造了一种完善保密加密的加密方法，现在称为“一次一密”（one-time pad），然而在当时并没有证明其是完善保密加密的（事实上，当时并没有完善保密加密的概念）。约 25 年以后，香农（Shannon）提出了完善保密加密的概念，并且证明“一次一密”能达到这一安全水平。

令$a\oplus b$表示把二进制串a和二进制串b按比特位异或（也就是说，如果$a=a_1,\cdots,a_l$，$b=b_1,\cdots,b_l$，则$a\oplus b=a_1\oplus b_1,\cdots,a_l\oplus b_l$）。“一次一密”方案定义如下：

(1) 令整数$l > 0$，设明文空间$\mathcal{M}$，密钥空间$\mathcal{K}$和密文空间$\mathcal{C}$都等于$\{0,1\}^l$(也就是说长度为l的二进制比特串的集合)。

(2) 密钥产生算法Gen从$\mathcal{K} = \{0,1\}^l$中依据均匀分布选择一个二进制比特串（2^l个串空间中任何一个被选中的概率都为2^{-l})。

(3) 加密算法Enc：给定一个密钥$k \in \{0,1\}^l$和一个明文$m \in \{0,1\}^l$，输出$c := k \oplus m$。

(4) 解密算法Dec：给定一个密钥$k \in \{0,1\}^l$和一个密文$c \in \{0,1\}^l$，输出$m := k \oplus m$。

在讨论“一次一密”的安全性前，我们注意到每个k和m都满足$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = k \oplus k \oplus m = m$，所以“一次一密”是一个正确的加密方案。

直观上，“一次一密”是完善保密加密，因为给定密文c，敌手没有办法知道来自哪个原始明文m。为弄清原因，注意到对所有可能m都存在一个密钥k满足$c = \mathsf{Enc}_k(m)$；也就是说，$k = m \oplus c$。此外，密钥根据均匀概率进行选择（不被敌手知道），所以没有一个密钥比其他密钥选中的可能性高。综上所述，由于所有明文被加密的机会均等，因而c没有泄露任何被加密明文m的信息。下面正式证明这一直觉：

定理 2.6 “一次一密”是完善保密加密。

证明 选定明文空间$\mathcal{M}$上的分布，选定任意一个$m \in \mathcal{M}, c \in \mathcal{C}$。“一次一密”有

$$\begin{aligned}\Pr[C = c \mid M = m] &= \Pr[M \oplus K = c \mid M = m] \\ &= \Pr[m \oplus K = c] = \Pr[K = m \oplus c] = \frac{1}{2^l}\end{aligned}$$

由于这对所有分布和所有m都成立，所以对于任意$\mathcal{M}$上的概率分布，每个$m_0, m_1 \in \mathcal{M}$和每个$c \in \mathcal{C}$，满足

$$\Pr[C = c \mid M = m_0] = \frac{1}{2^l} = \Pr[C = c \mid M = m_1]$$

根据引理 2.3，这意味着一次一密是完善保密加密。 ■

由此得出结论完善保密加密是可以实现的。令人遗憾的是，“一次一密”有很多缺陷，最突出的是需要密钥和明文有一样的长度。首先也是最重要的是，这意味着必须安全地存储一个很长的密钥，这在实践中问题较大且通常是不可能的。另外，其应用受到限制，例如当希望发送很长消息的时候（很难保证安全地存储一个很长的密钥），或者当事先并不知道明文消息的长度上限的时候（不可能共享一个没有长度上限的密钥）。并且，“一次一密”顾名思义，密钥只用一次的时候才是“安全”的。尽管并没有定义当多个消息被加密时的安全，但是很容易看出加密多于一次后会泄漏很多信息。特别地，如果明文信息m、m'用同样一个密钥k进行加密，敌手得到$c = m \oplus k$和$c' = m' \oplus k$从而可以计算出：

$$\begin{aligned}c \oplus c' &= (m \oplus k) \oplus (m' \oplus k) \\ &= m \oplus m'\end{aligned}$$

这样就知道了两条消息异或的结果。尽管这看上好像不很重要，但是已经足够说明其加密两条消息时不是完善保密加密。此外，如果明文消息对应英语文本，则给两个足够长消息的异或值，就可能通过频率分析（前一章有介绍，虽然更为复杂）得到明文消息。

2.3 完善保密的局限

本节介绍“一次一密”加密方案的上述限制是内在固有的。具体而言，证明所有完善保密加密方案的密钥空间至少要和明文空间一样大。如果密钥空间由固定长度的密钥组成，明文空间由固定长度的明文组成，则意味着密钥至少要和明文一样长。因此，长密钥的问题并不是“一次一密”专有，而是所有完善保密加密的内在问题。（另一个限制是密钥只能使用一次，也是完善保密加密的内在问题；参见练习 2.9。）

定理 2.7 设(Gen, Enc, Dec)是明文空间为$\mathcal{M}$的一个完善保密加密方案，$\mathcal{K}$的密钥空间由Gen决定，则$|\mathcal{K}| \geqslant |\mathcal{M}|$。

证明 下面证明如果$|\mathcal{K}| < |\mathcal{M}|$，则这个加密方案不是完善保密加密。假设$|\mathcal{K}| < |\mathcal{M}|$，考虑$\mathcal{M}$为均匀分布且设密文$c \in \mathcal{C}$有非零概率，$\mathcal{M}(c)$是所有密文$c$解密而得可能明文的集合，也就是

$$\mathcal{M}(c) \stackrel{\text{def}}{=} \{\hat{m} \mid \hat{m} = \mathsf{Dec}_{\hat{k}}(c)，\text{对于部分}\hat{k} \in \mathcal{K}\}$$

显然有$|\mathcal{M}(c)| \leqslant |\mathcal{K}|$，因为对于每个明文消息$\hat{m} \in \mathcal{M}(c)$，至少可以确定一个密钥$\hat{k} \in \mathcal{K}$使得$\hat{m} = \mathsf{Dec}_{\hat{k}}(c)$。（我们在前面曾假设Dec是确定的。）由于假设$|\mathcal{K}| < |\mathcal{M}|$，这就意味着有一些$m' \in \mathcal{M}$出现$m' \notin \mathcal{M}(c)$。但是

$$\Pr[M = m' \mid C = c] = 0 \neq \Pr[M = m']$$

所以这个方案不是完善保密加密。 ■

完善保密加密的更低代价？ 上面的定理显示出完善保密加密的内在局限性。即便如此，常有个人或者组织宣称自己开发了一种全新的无法破解的加密方案，并且达到“一次一密”那样的安全等级而不需使用长密钥。上面的证明说明这样的宣称不可能是事实；这些声称者要么根本不懂密码学，要么是在公开撒谎。

2.4 *香农定理

在香农开创性的关于完善保密加密的工作中，提出了完善保密加密方案的一种性质。这种特性为：假设$|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$，密钥产生算法Gen必须从所有可能密钥组成的集合中均匀地选择一个秘密密钥（就像“一次一密”），这样对于所有的明文和密文都存在唯一的密钥使明文映射到密文（还是就像“一次一密”那样）。进一步而言，这个定理是一个强大的验证（或者否定）完善保密加密方案的工具。证明后会进一步讨论这一点。

如前所述，假定$\mathcal{M}$和$\mathcal{C}$的概率分布满足：所有的$m \in \mathcal{M}$和$c \in \mathcal{C}$都为非零概率。这个定理考虑一种特殊情况：$|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$，即明文、密钥、密文集合的大小相等。由$|\mathcal{K}| \geqslant |\mathcal{M}|$，易知$|\mathcal{C}|$必须至少和$|\mathcal{M}|$一样大，否则一定存在两个明文映射到同一个密文（造成解密的不确定）。因此，从某种意义上说，$|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$的情形“效率最高”。现在可以开始陈述这个定理。

定理 2.8（香农定理） 设加密方案(Gen, Enc, Dec)的明文空间为$\mathcal{M}$，且$|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$，则当且仅当下列条件成立时，此方案是完善保密加密。

(1) 由Gen产生的任意密钥$k \in \mathcal{K}$的概率都是$1/|\mathcal{K}|$。

(2) 对任意明文$m \in \mathcal{M}$和任意密文$c \in \mathcal{C}$，只存在唯一的密钥$k \in \mathcal{K}$使得$\mathsf{Enc}_k(m)$输出c。

证明 证明背后的直觉逻辑如下。首先，如果加密方案满足条件(2)，则给定密文c，它可能从任意的明文m加密而来（这是由于对于任何明文m都存在一个密钥k映射到c）。同时，只有一个密钥映射每一个明文m到密文c，又由条件(1)知选取所有密钥的概率相同，完善保密加密可以用“一次一密”的方法来证明。以上为充分条件的证明，下面证明必要条件。如果$|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$则一定只有一个密钥映射每个明文$m$到每个密文$c$。（否则，要么有些明文$m$不能映射到某个给定的$c$，要么某些$m$通过多个密钥映射到$c$，导致另一个明文$m'$不能映射到$c$，这些都和完善保密加密相矛盾。）有鉴于此，密钥必须等概率选取，否则某些明文的可能性会高于其他明文，这与完善保密加密相悖。正式的证明如下。

设(Gen, Enc, Dec)为定理中的加密方案。简明起见，假设Enc为确定算法。首先证明如果(Gen, Enc, Dec)是完善保密加密，那么条件(1)和条件(2)成立。根据定理 2.7，不难发现对于每个$m \in \mathcal{M}$和$c \in \mathcal{C}$至少存在一个密钥$k \in \mathcal{K}$满足$\mathsf{Enc}_k(m) = c$。（否则，$\Pr[M = m \mid C = c] = 0 \neq \Pr[M = m]$。）选定$m$，考虑集合$\{\mathsf{Enc}_k(m)\}_{k\in\mathcal{K}}$，根据刚才的证明，$|\{\mathsf{Enc}_k(m)\}_{k\in\mathcal{K}}| \geqslant |\mathcal{C}|$。（因为任意$c \in \mathcal{C}$都存在一个$k \in \mathcal{K}$满足$\mathsf{Enc}_k(m) = c$。）另外，很明显有$|\{\mathsf{Enc}_k(m)\}_{k\in\mathcal{K}}| \leqslant |\mathcal{C}|$。得出结论

$$|\{\mathsf{Enc}_k(m)\}_{k\in\mathcal{K}}| = |\mathcal{C}|$$

既然$|\mathcal{K}| = |\mathcal{C}|$，于是$|\{\mathsf{Enc}_k(m)\}_{k\in\mathcal{K}}| = |\mathcal{K}|$。这意味着没有两个不同的密钥$k_1, k_2 \in \mathcal{K}$满足$\mathsf{Enc}_{k_1}(m) = \mathsf{Enc}_{k_2}(m)$。由于$m$是任意选择的，故对于任意$m$和$c$，都存在至少一个密钥$k \in \mathcal{K}$满足$\mathsf{Enc}_k(m) = c$。综上所述（即存在至少一个密钥，且至多一个密钥），得到条件(2)。

下面证明对于任意$k \in \mathcal{K}$有$\Pr[K = k] = 1/|\mathcal{K}|$。设$n = |\mathcal{K}|$，$\mathcal{M} = \{m_1, \cdots, m_n\}$（其中，$|\mathcal{M}| = |\mathcal{K}| = n$），并且选定密文$c$。标注密钥为$k_1, \cdots, k_n$以至于对于任意$i$（$1 \leqslant i \leqslant n$）都有$\mathsf{Enc}_{k_i}(m_i) = c$成立。这种标注可以实现是因为对任意密文$c$和$m_i$，都存在一个唯一的密钥$k_i$，满足$\mathsf{Enc}_{k_i}(m_i) = c$。此外对于不同的明文$m_i, m_j$，其密钥也不同（否则可能会出现解密结果不确定）。由完善保密加密,得到对任意i,有

$$\begin{aligned}\Pr[M = m_i] &= \Pr[M = m_i \mid C = c] \\ &= \frac{\Pr[C = c \mid M = m_i] \cdot \Pr[M = m_i]}{\Pr[C = c]} \\ &= \frac{\Pr[K = k_i] \cdot [M = m_i]}{\Pr[C = c]}\end{aligned}$$

其中第二个等式是由贝叶斯定理得来的，第三个等式由前面的标注得到的（即k_i是m_i映射到c的唯一密钥）。

综上所述，对于任意i，

$$\Pr[K = k_i] = \Pr[C = c]$$

因此，对于所有i和j，$\Pr[K = k_i] = \Pr[C = c] = \Pr[K = k_j]$，所有密钥等概率选取。得到结论：密钥选择服从均匀分布。也就是说，对任意密钥k，满足$\Pr[K = k_i] = 1/|\mathcal{K}|$，得证。

下面证明充分条件。假设所有密钥出现的概率满足$1/|\mathcal{K}|$，并且对于任意$m \in \mathcal{M}$和任意$c \in \mathcal{C}$都有唯一密钥$k \in \mathcal{K}$满足$\mathsf{Enc}_k(m) = c$。这意味着可立刻得出：对于所有m和c，有

$$\Pr[C = c \mid M = m] = \frac{1}{|\mathcal{K}|}$$

与$\mathcal{M}$上的概率分布无关。因此，对于任意$\mathcal{M}$上的概率分布，任意$m, m' \in \mathcal{M}$，任意$c \in \mathcal{C}$都满足

$$\Pr[C = c \mid M = m] = \frac{1}{|\mathcal{K}|} = \Pr[C = c \mid M = m']$$

所以根据引理 2.3 可知，这个加密方案是完善保密加密。 ■

香农定理的应用：定理 2.8 从本质上给出了完善保密加密的完整特征。另外，由于条件(1)和条件(2)与明文空间$\mathcal{M}$上的概率分布没有任何关系，这个定理意味着：如果存在一个加密方案对明文空间$\mathcal{M}$上的特定概率分布满足完善保密加密，则此加密方案在一般情况下也能满足完善保密加密（这里一般情况指明文$\mathcal{M}$上的所有概率分布）。最后，香农定理在证明一个给定方案是否是完善保密加密上有很大的用处。条件(1)很容易确认，条件(2)可以在不分析任何概率的情况下来证明或否定（这一点和定义 2.1 不同）。例如，“一次一密”的完善保密性（定理 2.6）用香农定理证明很就很简单。但是，定理 2.8 只在$|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$时满足，所以在应用定理时要特别小心。

2.5 小　结

这里将结束完善保密加密的介绍，本章主要思想是：完善保密加密是可以达到的，也就是说存在这样的加密方案使得密文完全不会泄漏明文的信息，即使攻击者有无限制的计算能力。但是，这些方案的局限是密钥必须至少和明文一样长。因此，在实践中完善保密加密很少用到。传言在冷战时期连接白宫和克里姆林宫的“红色电话”是用“一次一密”来保护的。当然，美国政府与苏联之间进行超长随机密钥的交换并不是很困难，所以“一次一密”实践上是可行的。然而，在大多数情况下（尤其是商业模式下），这种长密钥的局限令“一次一密”以及其他完善保密加密方案无法使用。

参考文献和扩展阅读材料介绍

“完善保密加密”最初是由香农提出并进行研究的[127]。此外，为了介绍这个定义，他证明了“一次一密”（最初由 Vernam[141]提出）是完善保密加密，并且也证明了完善保密加密的特征（及其局限性）。Stinson[138]给出了关于完善保密加密的更多讨论。

本章简要地介绍了完善保密的加密方案。还有其他一些密码学问题同样也可以用“完美”安全来解决。一个典型的例子就是消息鉴别问题，目标是防止敌手在传输数据时（以一种不能被检测到的方式）修改数据，这个问题将在第 4 章深入讨论。对完美安全的消息认证有兴趣的读者可以参考 Stinson[136]、Simmons[134]的综述，或者 Stinson 教课书的第一版[137]第 10 章了解更多信息。

练　习

2.1　证明定理 2.2 的必要条件。

2.2　给出证明或者提出反例：

所有完善保密加密方案都满足对任意明文空间$\mathcal{M}$，任意$m, m' \in \mathcal{M}$和任意$c \in \mathcal{C}$，都满足

$$\Pr[M = m \mid C = c] = \Pr[M = m' \mid C = c]$$

2.3　当使用密钥为$k = 0^\ell$的“一次一密”（Vernam's 密码）时，有$\mathsf{Enc}_k(m) = k \oplus m = m$，明文其实没有加密。因而建议在加密时使用$k \neq 0^\ell$来改善“一次一密”。（即随机均匀地从长为$\ell$的非零密钥集合中选取密钥。）这样是否可以改进“一次一密”？特别是它还是完善保密加密吗？证明你的回答。如果你的回答是肯定的，解释为什么“一次一密”不能用这种方式来描述。如果你的回答是否定的，用事实说明用0^ℓ加密并没有改变明文。

2.4　本题研究移位密码、单字母替换密码和 Vigerère 密码在何种条件下是完善保密加密。

(1) 证明只有给单个字符加密时，移位密码才是完善保密加密。

(2) 单字母替换密码是完善保密加密的最大明文空间$\mathcal{M}$是什么？（$\mathcal{M}$不必仅包括有效的英文单词。）

(3) 说明如何使用 Vigenère 加密方法给长为t的单词加密以达到完善保密加密（注意：密钥长度可以选择）。证明你的回答。

结合前面章节介绍的攻击案例。

2.5　证明或者提出反例：任意密钥空间大小与明文空间大小相等，且密钥从密钥空间均匀选择的加密方案是完善保密加密。

2.6　加密方案(Gen, Enc, Dec)对于任意$\mathcal{M}$上的分布，每一个非零概率$m \in \mathcal{M}$都满足定义 2.1（利用本章中的简化约定）。证明此方案在明文空间$\mathcal{M}$上所有分布都满足此定义（即包括可能出现零概率明文的明文空间）。推断，此方案对于任意明文空间$\mathcal{M}' \subset \mathcal{M}$仍然满足完善保密加密。

2.7　证明由定义 2.1 可以推出定义 2.4。

提示：利用习题 2.6，当明文分布为任何两个明文上的均匀分布，依旧是完善保密加密。（特别是这两个消息都是由实验中的$\mathcal{A}$输出的。）然后，利用引理 2.3。

2.8　证明命题 2.5 的必要条件。也就是证明定义 2.4 推出定义 2.1。

提示：如果加密方案Π根据定义 2.1 不是完善保密加密，则引理 2.3 表明存在明文$m_0, m_1 \in \mathcal{M}$和$c \in \mathcal{C}$，满足$\Pr[C = c|M = m_0] \neq \Pr[C = c|M = m_1]$。利用$m_0, m_1$构造的敌手$\mathcal{A}$，其成功的概率为$\Pr|\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1| > \frac{1}{2}$。

2.9　考虑下面对加密两个消息的完善保密加密定义。一个明文空间为$\mathcal{M}$的加密方案(Gen, Enc, Dec)是两消息完善保密加密，当对$\mathcal{M}$上的所有分布，任意$m, m' \in \mathcal{M}$，以及任意$c, c' \in C$满足$\Pr[C = c \wedge C' = c'] > 0$，有

$$\Pr[M = m \wedge M' = m' \mid C = c \wedge C' = c'] = \Pr[M = m \wedge M' = m']$$

其中m和m'是从相同$\mathcal{M}$上的分布中独立选择的。证明不存在加密方案满足这个定义。

提示：利用$m \neq m'$，但是$c = c'$。

2.10 考虑下面对两消息加密的完善保密加密定义。明文空间为$\mathcal{M}$的加密方案(Gen, Enc, Dec)是双消息完善保密加密，当对于明文空间$\mathcal{M}$上的所有分布，任意$m, m' \in \mathcal{M}$，以及任意$c, c' \in C$满足$c \neq c'$且$\Pr[C = c \wedge C' = c'] > 0$，有

$$\Pr[M = m \wedge M' = m' \mid C = c \wedge C' = c'] = \Pr[M = m \wedge M' = m' \mid M \neq M']$$

其中，m和m'是从$\mathcal{M}$上的同一分布中独立选择的。提出一个可证明满足这个定义的加密方案。密钥长度与明文长度相比如何？

提示：提出的加密方案可不要求效率。

2.11 证明可以假设密钥产生算法Gen是从密钥空间$\mathcal{K}$上均匀选择的。

提示：设$\mathcal{K}$表示随机算法Gen产生的所有随机带的集合。

2.12 假设只要求明文空间$\mathcal{M}$上的加密算法(Gen, Enc, Dec)满足下列条件：对所有$m \in \mathcal{M}$，$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$的概率至少是$2^{-t}$（这个概率是通过选择密钥$k$，以及加密解密算法的随机性决定的）。完善保密加密（如定义 2.1 所述）在当$|\mathcal{K}| < |\mathcal{M}|$，其中$t \geqslant 1$时可以达到。你能找到并证明$\mathcal{K}$的更小的界吗？

2.13 证明一个类似于定理 2.7 中的“几乎完美”安全的情况。设$\varepsilon < 1$是一个常量，仅要求对明文空间$\mathcal{M}$上的所有分布，任意$m \in \mathcal{M}$和任意$c \in C$，有

$$|\Pr[M = m \mid C = c] - \Pr[M = m]| < \varepsilon$$

找到并证明对任何满足这一定义的加密方案，相对于$\mathcal{M}$的$\mathcal{K}$的更小的界。

第二部分　对称密钥（对称）密码学

第3章　对称密钥加密以及伪随机性

本章将学习“伪随机性”的概念，即“看上去”是完全随机的（满足精确的定义），尽管它其实不是。理解如何用它来打破上一章给出的完善保密加密的约束条件。特别是本章会介绍一个使用短密钥（长度有几百比特）的加密方案用来安全地加密很长的消息（假设总共有几个吉字节）。这样的方案能够避开完善保密的内在限制，能够实现稍弱但已足够的计算安全。在开始讨论对称密钥加密之前，首先考察密码学中的一般计算方法。这种计算方法将在本书其他部分使用，是现代密码学的基础。

3.1　密码学的计算方法

在前两章中，已经学习了什么是经典密码学。从对历史上使用的密码方案的一个简短回顾开始，重点关注它们是如何被攻破的，以及从这些攻击中，可以学到什么。在第2章中，继续讲解密码学方案，这些密码学方案能够被证明是安全的（根据一些特殊的安全定义），甚至假设敌手拥有无限计算能力。这样的方案叫做“信息理论安全”，或者是“完美安全”，因为它们的安全性是基于敌手没有足够的“信息”来成功地完成攻击，而不管敌手的计算能力[①]。特别地，如前所述，在“完善保密”的加密方案中，密文不能含有任何关于明文的信息（假设密钥是未知的）。

信息理论安全与计算安全形成鲜明对比，后者是大多数现代密码学构造方法的目标。若限制在对称密钥加密的情形（尽管这里的讨论可以应用在更加通用的情况），现代加密方案的特点是，只要给足够的时间和计算能力，它们都能被攻破，所以它们不满足定义 2.1。尽管如此，在某种假设下，即使使用现有的最快的超级计算机，为了攻破这些加密方案需要的计算量所耗运行时间将会超过人的一生。对于所有实用目的，这个级别的安全就足够了。

计算安全比信息理论安全要稍弱一些。计算安全当前仍然依赖不能被证明的假设，然而实现信息理论安全不需要假设[②]。姑且认为计算安全可满足所有的实际用途，为什么要放弃实现完美安全的想法？2.3 节的结论给出了一个为什么现代密码学已经选择走这条路线的原因。该节说明完善保密的加密方案使得密钥的长度有严格的下界限制；也就是说，密钥的长度必须和所有被该密钥加密的消息的总长度相等。当需要信息理论安全时，其他的密码学任务也

① 术语“信息”有严格的数学含义，但是在这里以非正式的方式来使用它。

② 理论上，这些假设可能有一天都会被推翻。但是不幸的是，现在的知识状况要求做出一些假设，来证明任何密码学构造方案中的计算安全。那些熟悉$\mathcal{P}$和$\mathcal{NP}$问题的读者，值得留意：任何对密码学构造方法的计算安全的无条件证明，都需要证明$\mathcal{P} \neq \mathcal{NP}$。

拥有着类似的消极结论。因此，尽管数学上是吸引人的，但为了获得实用的密码学方案，有必要对完美安全进行妥协。需要强调的是，尽管不能获得完美安全，但这个并不意味着要抛开严格的数学方法。相反，定义和证明仍然是必须的，唯一的区别是，现在考虑的是稍弱一些，但是仍然有意义的安全定义。

3.1.1 计算安全的基本思想

Kerckhoffs 以他的“密码学设计应该是公开的”原则而闻名于世。但是，他实际上阐述了 6 条原则，下面的原则与本节的讨论非常相关：

一种[密码术]必须在实践上无法破译，如果不是数学上无法攻破的话。

尽管他在那个时候不太可能用现在的方式来陈述，这个 Kerckhoffs 原则从根本上说明了，使用一个完善保密的加密方案是不必要的，取而代之，使用一种不能在“合理的时间”内，以任何“合理的成功概率”攻破的方案就足够了（用 Kerckhoffs 的话说，这是一个“实践上不可破译”的方案）。用更具体的术语，使用这样一种加密方案就足够了：该方案能够（理论上）被攻破，但是在 200 年内，使用最快的可用的超级计算机，不能以大于10^{-30}的概率攻破。本节提出一个构架，来形式化地陈述密码学方案是“实践上不可破译”。

基于计算的方法包含了两种放宽的完美安全概念：

(1) 仅仅在对抗“有效的”敌手时，安全性存在，“有效的”是指在可行的时间里运行。

(2) 敌手潜在的成功概率是非常小的（小到以至于不关心它是否会真的发生）。

为了获得一个有意义的理论，需要精确地定义上面所提到的内容。有两个普遍的方法可以做到：具体方法和渐进方法。下面一一解释。

具体方法。具体方法是，通过明确限定任意敌手在最多某个特定时间内的最大成功概率，对一个给定密码学方案的安全性进行量化。也就是，令t, ε为正数常量，$\varepsilon \leqslant 1$。则一个对安全的具体定义，大体地说，将会是以下的形式：

一个方案为(t, ε)安全，如果每个运行时间最多为t的敌手以最多为ε的概率成功攻破该方案。

（当然，上面提供的仅仅只是一个通用模板，为了让上面的陈述有意义，需要准确地定义什么是“攻破”方案。）作为一个例子，一个人可能希望使用这样的方案，该方案能够保证在使用最快的可用的超级计算机，且最多运行 200 年的情况下，没有敌手能够以大于10^{-30}的概率成功攻破该方案。或者，以 CPU 周期作为单位来测量运行时间可能会更方便，使用这样一种密码学方案，在最多运行2^{80}个 CPU 周期的情况下，没有敌手能够以高于2^{-64}的概率攻破该方案。

感觉一下t, ε的值是有益的，这些值在现代密码方案中是有代表性的。

例 3.1 现代对称密钥加密方案一般被认为给予了几乎最优的安全性：当密钥的长度为n，一个敌手运行时间为t（假设用 CPU 周期测量），能够成功攻破该方案的概率最多为$t/2^n$。（稍后将会看到为什么这个确实是最优的。）对$t = 2^{60}$这个量级，计算在现在几乎不可能达到。在 1 兆赫的计算机（每秒执行10^9次周期）上运行2^{60}个 CPU 周期需要$2^{60}/10^9$秒，大约 35 年。并行地使用多台超级计算机可能会将这个数据降低到几年。

一个普遍的密钥长度值，可能为$n = 128$。在2^{60}和2^{128}之间的差别是一个包含了 21 个十

进制位的乘法因子2^{68}。为了对这个数有多大找点感觉，请注意，自从宇宙大爆炸到现在，根据物理学家的估计，秒数为2^{58}。

每百年发生一次的事件能够大致地被估计为事件在任意指定的一秒内发生的概率为2^{-30}。在任意指定的一秒内发生概率为2^{-60}的事件是前面提到概率的2^{30}分之一，可能被期待每 1000 亿年大概发生一次。因此，一个谨慎的参数选择应该为$t = 2^{80}$以及$\varepsilon = 2^{-48}$（意味着密钥必须至少有128比特长）。 ◇

具体方法在实践中是有用的，因为具体方法保证了上面这些有实际数据的安全，这些都是密码方案的用户感兴趣的。但是，在解释具体的安全保证时，必须要小心。举个例子，当声称没有敌手能够在 5 年的时间，以大于ε的概率攻破一个指定的方案，必须清楚：这里假定什么类型的计算能力（如台式个人计算机、超级计算机、联网的数百台计算机）？这里是否考虑了未来计算能力的提高（通过摩尔定理，计算能力每隔 18 个月便会增加一倍）？这里假定的是使用“现成的”算法，还是对攻击优化的专门软件？更进一步说，这样的一个保证几乎没有提及敌手运行 2 年的成功概率（除了概率最大为ε外），并且没有提及敌手运行 10 年的成功概率的相关信息。

当使用一个具体安全方法时，方案能够达到(t, ε)安全，但是永远不会只是安全（单独出现）。更加确切地说，t, ε的范围应该是多少，才能够认为一个(t, ε)安全方案是“安全”的？对此，没有明确的答案，因为一个安全保证可能能够满足普通用户，但是当用于加密机密的政府文件时，可能就无法满足了。

渐进方法。渐进方法是本书采用的方法。该方法来源于复杂性理论，将敌手的运行时间以及成功的概率视为某个参数的函数，而不是具体的数值。特别是，一个密码方案将包含一个安全参数，即整数n。当诚实的双方初始化方案（即生成密钥）时，它们选择某个n值作为安全参数；假设这个值被任何攻击该方案的敌手知道。那么敌手的运行时间（以及诚实方的运行时间）以及敌手的成功概率将都被看作n的函数。于是：

(1) 将“可行的策略”或者“有效的算法”的概念，和在以n为参数的多项式时间内运行的概率算法等同看待。（有时使用 PPT 来代表概率多项式时间。）这意味着，对于一些常量a, c，安全参数为n，该算法运行时间为$a \cdot n^c$。要求诚实方在多项式时间内运行，并且仅关心对抗多项式时间的敌手。需要强调的是，敌手尽管要求在多项式时间内运行，但可能比诚实方的计算能力更强（并且运行更长时间）。而且，一个需要超多项式时间的攻击策略不被认为是有现实意义的威胁（所以基本上被忽略）。

(2) 将“小的成功概率”的概念，和成功概率小于任何以n为参数的多项式倒数等同看待，这意味着对于每个常量c，当n的值足够大，敌手成功的概率小于n^{-c}（参见定义 3.4）。比任何多项式的倒数增长得都慢的函数被称为可忽略的（negligible）。

因此，对渐进安全的定义一般采用下面的形式：

如果每个 PPT 敌手以可忽略的概率成功攻破一个方案，那么该方案是安全的。

尽管从理论上看是很清晰的（因为能够确实地说出一个方案是否安全），但是要理解渐进方法仅仅只是在n足够大的情况下才“保证安全”，下面的例子会说明这一点。

例 3.2 假设我们有一个方案是安全的，则它可能是这样的，一个运行了n^3分钟的敌手能

够成功“攻破该方案”的概率为$2^{40} \cdot 2^{-n}$（这是一个可忽略的关于n的函数）。当$n \leqslant 40$，意味着一个运行40^3分钟（大约 6 周）的敌手能够攻破该方案的概率为 1，所以那样的n值不是很有用。甚至当$n = 50$，一个运行50^3分钟（大约 3 个月）的敌手能够攻破该方案的概率约为1/1000，这是不能被接受的。从另一个方面，当$n = 500$时，一个运行时间超过200年的敌手攻破该方案的概率大概仅仅为2^{-500}。 ◇

前面的例子指出，可以将一个更大的安全参数看作提供了“更高的”安全级别。就大部分情况而言，安全参数决定了诚实方使用的密钥长度，因此有个类似的概念，即密钥越长，安全程度越高。通过使用一个更大的安全参数值来“增强安全性”的能力有着很重要的实用意义，因为它能够使诚实方抵御敌手计算能力的增强以及算法的进步。下面的例子将充分反映这一点是如何在实际中体现的。

例 3.3 让我们来看一下，在实际中更快的计算机的可用性对安全的影响。假设有一个密码学方案，诚实方被要求运行$10^6 \cdot n^2$个时钟周期，同时，运行$10^8 \cdot n^4$个周期的敌手能够成功“攻破”该方案的概率为$2^{20} \cdot 2^{-n}$（该例子中的数字都是经过设计的，使得计算更简单，并不等同于任何一种存在的密码方案）。

假设所有的各方都是用 1 兆赫的计算机，并且$n = 50$。则诚实方运行$10^6 \cdot 2500$个周期，或2.5秒，同时，一个运行$10^8 \cdot (50)^4$个周期，或者大概一周的敌手，能够攻破该方案的概率只有2^{-30}。

现在假设一台 16 兆赫的计算机是可用的，并且所有各方都升级更新。诚实方能够将n增长到100（需要生成一个新的密钥），运行时间仍然被改进到0.625秒（即$10^6 \cdot 100^2$个周期，速度为$16 \cdot 10^9$周期/秒）。相比之下，敌手现在不得不运行10^7秒，或者超过16周，来实现成功概率2^{-80}。一个运行速度更快的计算机的影响就是使得敌手的工作更加困难。 ◇

渐进方法的优点就是不需要依赖于任何特定的假设，比方说，敌手使用机器的类型（这是计算复杂性理论中丘奇—图灵论题的一个结果，从根本上阐述了所有有足够能力的计算设备之间的相对速度是多项式相关的）。另一个方面，正如上面的例子证明的，在实际中，准确地理解一个特殊的渐进安全方案意味着什么水平的具体安全是有必要的。这是因为，诚实方必须选择一个具体的n值来使用，并且不能依赖于对“足够大的n值”的保证。决定安全参数值的任务很复杂，依赖于考虑之中的方案以及其他考虑。幸运的是，将渐进安全的保证转换为一个具体安全的保证往往相对简单。

从这里开始，将只使用渐进方法。尽管如此，正如上面的例子说明的那样，本书中所有的结论都能够转换为具体的安全结论。

一个技术注释。正如我们提到的，将敌手和诚实方的运行时间看作是n的函数。为了和算法以及复杂性理论中的标准约定保持一致，即一个算法的运行时间被衡量为一个输入长度的函数，必要的时候把安全参数表示为一进制的形式1^n（即n个1的字符串），提供给敌手和诚实方。

松弛的必要性。如前所述，计算安全在完美安全的概念中引入了两种松弛：首先，安全性仅仅抵抗有效的（即多项式时间的）敌手；其次，允许小的（即可忽略的）成功概率。这两种对概念的松弛对实现实践中的密码方案非常必要，特别是避开了完善保密加密的消极结

论。现在非正式地讨论其原因。假设有一个加密方案，密钥空间$\mathcal{K}$的大小比消息空间$\mathcal{M}$小得多（根据前一章所述，这意味着该方案不能够提供完善保密）。无论该加密方案是如何构造的，都有两种攻击可以应用，这两种攻击在两个相反的极端：

(1) 给定一份密文c，一个敌手通过使用所有的密钥$k \in \mathcal{K}$，能够破译c。这样将会给出一列c对应的所有可能消息的清单。因为该清单不可能含有所有的$\mathcal{M}$（因为$|\mathcal{K}| < |\mathcal{M}|$），这样会泄露已加密消息的一些信息。

再者，假设敌手执行了已知明文攻击，并且掌握了密文$c_1, \cdots, c_\ell$，分别对应着消息$m_1, \cdots, m_\ell$。该敌手能够反复地尝试使用所有可能的密钥，解密每个密文，直到找到一个密钥k，对于所有的i，满足$Dec_k(c_i) = m_i$。很大概率上，这个密钥是唯一的，在这种情况下，该敌手已经找到了诚实方正在使用的密钥[③]。因此，随后使用这个密钥就变得不安全了。

(2) 再次考虑这种情况，敌手掌握了密文$c_1, \cdots, c_\ell$，分别对应着消息$m_1, \cdots, m_\ell$。该敌手能够随机猜一个密钥$k \in \mathcal{K}$，然后检查看看对于所有的i，是否$Dec_k(c_i) = m_i$。如果满足，再次期望k有很大概率就是诚实方正在使用的密钥。

这里，敌手在必要的恒定时间内运行，成功的概率非零（尽管非常小），约为$1/|\mathcal{K}|$。

如果希望用单个短密钥来加密很多消息，要实现安全性，只有限制敌手的运行时间（以至于敌手没有时间来执行蛮力搜索），并且不认为非常小的成功概率是一个攻破（所以第二种“攻击”被排除）。因此，前面提到的两种放宽是必要的。上面的讨论得出的一个额外结果，就是在任何加密方案中，密钥空间必须足够大，以至于敌手不能够遍历。更正式地，密钥空间的规模必须为安全参数的超多项式。

3.1.2 有效的算法和可忽略的成功概率

上节概述了本书使用的渐进安全方法。那些未接触过算法或者复杂性理论的读者可能对“多项式时间”，“概率（或随机）算法”，或者“可忽略概率”这些概念不适应，对渐进方法感到困惑。本节将更加详细和更加正式地重述这种（渐进）方法。那些对上节内容已很了解的读者，建议直接跳到3.1.3节，必要时再回来查阅。

有效的计算。前面已经定义有效的计算是指能够在“概率多项式时间”（PPT）内执行的计算。一个算法A被认为在多项式时间内运行是指：如果存在一个多项式$p(\cdot)$，使得对于每个输入$x \in \{0,1\}^*$，$A(x)$的计算最多在$p(|x|)$个步骤内终止（这里$|x|$表示的是字符串x的长度）。一个概率算法具有“抛硬币”的能力，这是一种比喻，说明该算法可以获得一个随机源，该随机源产生无偏差的随机比特，其中每个比特都是相互独立的，为1的概率为1/2，为0的概率也是1/2。同样的，一个随机算法可视为除了输入，还赋予了一个随机带上的均匀分布的“足够长”的比特字符串。当考虑一个概率多项式时间算法，该算法运行时间为p，输入长度为n，一个长度为$p(n)$的随机字符串将是足够的，因为该算法在被分配的时间内，只能使用$p(n)$个随机比特。

那些对复杂性理论或者算法熟悉的读者将会意识到，将有效的计算和（概率）多项式时间计算等同起来的想法对密码学来说，并不是唯一的。（概率）多项式时间算法最主要

③ 技术上说，密钥可能实际上不唯一（举个例子，可能相同一个密钥被不止一个字符串表示）。尽管如此，如果一个等价的密钥将被找到，效果是相同的。

的优点就是给出了一类在组成上是封闭的算法，意思是说，一个多项式时间的算法A，将另外的多项式时间算法A'作为一个子函数来运行，则A也能够在多项式时间内运行。除了这个有用的事实，限制敌手在多项式时间内运行并没有其他内在的特殊原因，并且本书基本上所有的结论，都将公式化地表示为敌手的运行时间为$n^{\mathcal{O}(\log n)}$（诚实方仍然运行在多项式时间内）。

在继续讨论之前，要解决一个问题：为什么要考虑概率多项式时间算法而不是确定性多项式时间算法。对此，有两个主要的原因。首先，随机性对密码学来说是必要的（例如，为了选择随机的密钥等），所以诚实方必须是概率的。给定了这种情况，很自然地考虑到攻击方也是概率的。考虑概率算法的第二个原因是，“抛硬币”的能力可能提供了额外的力量。因为使用有效计算的概念来作为现实中的敌手模型，让这个类型尽可能地扩大是很重要的（在保证是现实的情况下）。

额外提醒一下：概率多项式时间的敌手是否比确定性多项式时间的敌手能力更强大，这个问题还没有解决。事实上，最近的复杂性理论中的结论指出随机性没有起到帮助作用。尽管如此，这个结论没有影响将敌手建模成概率算法的做法，并且这只能提供更强的保证，也就是说，任何安全方案若能抵抗概率多项式时间敌手，则也能抵抗确定性多项式时间的敌手。

生成随机性。前面已将各方建模成概率多项式时间算法，如前所述，密码系统只有在随机性可用时，才是可能的（如果秘密密钥不能够随机生成，则一个敌手就能够依照诚实方相同的步骤，来获得他们的“秘密密钥”。回想 Kerckhoffs 准则，假设这个过程是已知的）。有了这个前提，某些人可能怀疑是否有可能在计算机上实现“扔硬币”来得到概率计算。

实际中，很多方法能够获得“随机比特”。一个解决方案就是使用硬件随机数发生器，它能够根据特定的物理现象，比如热/电噪声或者是放射衰变，来生成随机比特流。另外一种可能性是，使用软件随机数发生器，它能够根据不可预料的行为，比如击键之间的间隔时间、鼠标的移动、硬盘访问时间等，来生成随机比特流。一些现代操作系统提供了这种类型的函数。在这两种情况下，依赖的不可预料事件（自然的或者是依赖用户的）不可能直接产生均匀分布的比特，所以对初始比特流的进一步处理是需要的。对此，技术上较复杂，一般很难理解，并且也超出了本书的范围。

必须非常小心地选择随机比特，因为使用设计拙劣或者不恰当的随机数发生器往往会让一个好的密码系统在攻击面前变得脆弱。特别地，要使用一个为“密码学用途”而设计的随机数发生器，而不是一个为“通用目的”而设计的随机数发生器，后者可能对某些应用来说可以，但是对密码学应用来说是不行的。作为一个具体的例子，在 C 编程语言中，使用`random()`函数是个很差的想法，因为它根本就不随机。同样地，当前时间（即使精确到毫秒）并不是非常随机，不能作为一个秘密密钥的基础。

可忽略的成功概率。现代密码学允许：能以非常小的概率被攻破的方案，仍被认为是“安全的”。考虑多项式运行时间是可行的，类似地，认为概率为多项式倒数时是显著的。因此，对某个（正）多项式p，如果一个敌手能够成功地攻破该方案的概率为$1/p(n)$，则该方案被认为是不安全的。但是，对任意多项式p，如果攻破该方案的概率渐进小于$1/p(n)$，则认为该方案是安全的。这是因为敌手成功的概率太小，被认为是无趣的。这样的概率叫做可忽略的，定义如下。

定义 3.4 函数f为可忽略的，如果对于每个多项式$p(\cdot)$，存在一个N，使得对于所有的整数$n > N$，都满足$f(n) < \frac{1}{p(n)}$。

一个和上面等价的公式化的表述是，对所有常量c，存在一个N，使得对于所有的$n > N$，满足$f(n) < n^{-c}$。为了方便记忆，上面的表述也可以陈述如下：对于任意多项式$p(\cdot)$以及所有足够大的n值，满足$f(n) < \frac{1}{p(n)}$。这当然是等价的陈述。通常将一个可忽略函数表示为negl。

例 3.5 函数2^{-n}，$2^{-\sqrt{n}}$以及$n^{-\log n}$都是可忽略的。但是，它们接近零的速度是非常不同的。为了说明此点，下面计算什么样的n值，会使得每一个函数的值都小于10^{-6}。

(1) $2^{20} = 1048576$，因此对于$n \geqslant 20$，有$2^{-n} < 10^{-6}$。

(2) $2^{\sqrt{400}} = 1048576$，因此对于$n \geqslant 400$，有$2^{-\sqrt{n}} < 10^{-6}$。

(3) $32^5 = 33554432$，因此对于$n \geqslant 32$，有$n^{-\log n} < 10^{-6}$。

从上面的示例可能产生一个印象，就是$n^{-\log n}$比$2^{-\sqrt{n}}$更快地接近零。但这并不正确；对于所有的$n > 65536$，满足$2^{-\sqrt{n}} < n^{-logn}$。尽管如此，这还是不能说明当$n$的值为成百上千时，敌手成功的概率$n^{-logn}$会比概率$2^{-\sqrt{n}}$更高。 ◇

可忽略的成功概率的一个技术上的优势是，它们也遵循某个闭包的特性。下面是个简单的练习。

命题 3.6 令negl_1和negl_2为可忽略函数。则

(1) 函数negl_3被定义为$\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n)$，则该函数是可忽略的。

(2) 对于任何正多项式p，函数$\mathsf{negl}_4(n)$被定义为$\mathsf{negl}_4(n) = p(n) \cdot \mathsf{negl}_1(n)$，则该函数是可忽略的。

上面命题的第(2)部分，意味着如果在某个实验中，一个特定的事件发生的概率可以忽略，那么该实验重复多项式次，该事件发生的概率还是可忽略的。举个例子，如果抛硬币n次，都是“正面”的概率可忽略。这个意味着，如果抛n个硬币多项式次，则任何一次实验中，n个“正面”的概率也是可以忽略的。（正式地，该结论已经被使用一致限来证明了，在命题中 A.7 陈述。）

以可忽略的概率发生的事件，对所有实践中的用途来说（至少对于足够大的n值），能够被安全地忽略，理解这一点是很重要的。其重要性值得反复强调：

以可忽略的概率发生的事件，因为几乎不可能发生，所以对所有实践用途，它们可被忽略。因此，一个攻破密码方案的事件若发生的概率是可忽略的，则是不重要的。

为了避免读者对敌手在某个微小（但是非零）概率下，能够攻破一个给定方案的事实感觉不适应，这里给出一个例子，在某个微小（但是非零）概率下，当执行该安全方案时，诚实方被小行星撞到。更温和地解释，但是意思相同：某个诚实方的硬盘以非零的概率失效，从而清除了秘密密钥。因此，担心发生概率足够小的事件是没有意义的。

对渐进安全的总结。任何安全定义由两部分组成：对该方案的“攻破”的定义，和对敌

手能力的详细说明。敌手的能力和很多问题（比如，在加密情况下，假设是唯密文攻击，还是选择明文攻击）相关。但是，考虑到敌手的计算能力，从现在开始，将敌手建模成有效的和概率多项式时间的（意味着只有“可行的”攻击策略被考虑）。定义往往被公式化，以至于以可忽略的概率发生攻破不被认为是重要的。因此，任何安全定义的一般框架如下：

如果每个概率多项式时间的敌手$\mathcal{A}$执行某个特定类型的攻击，在这次攻击（成功也被很好地定义过）中，$\mathcal{A}$成功的概率是可忽略的，那么这个方案是安全的。

这样的定义是渐进的，因为对于小的n值，敌手成功的概率很高是有可能的。为了说明更多细节，在上面的陈述中，将使用到上面陈述的对“可忽略”的完全定义：

如果每个概率多项式时间的敌手$\mathcal{A}$执行某个特定类型的攻击，对于每个多项式$p(\cdot)$，存在一个整数N，对$n > N$，$\mathcal{A}$在这次攻击中成功的概率小于$\frac{1}{p(n)}$，则这个方案是安全的。

注意，对$n \leqslant N$时，没有保证。

3.1.3 规约证明

如前所述，如果给予足够的时间，一个计算安全（而不是完美安全）的密码方案往往能够被攻破。为了无条件地证明某方案是计算安全的，需要证明攻破该方案所需要时间的下界。特别地，需要证明该方案不能被任何多项式时间算法攻破。不幸的是，当前的现状是还不能证明该类型的下界。事实上，对任何现代加密方案的无条件的证明，需要突破看起来遥不可及的复杂性理论中的结论[④]。这看上去别无选择，只能简单地假设给定的方案是安全的。正如在 1.4 节中讨论的那样，但是，这是个非常不理想的方法，历史证明这个方法很危险。

不用假设给定的密码学构造是安全的，取而代之的策略是，假设某个低层次难题很难解决，然后证明基于这个假设，构造方案是安全的。在 1.4.2 节中，已经很详细地解释了为什么这个方法更合适，所以在这里，不再重复这些论述了。

只要构造所依赖的问题是困难的，则给定的密码学构造方案就是安全的，对这一逻辑关系进行证明的一般方式是，陈述一个显式规约，说明如何将任何概率不可忽略的且成功“攻破”该构造方案的有效敌手$\mathcal{A}$，转换为成功解决该难题的有效算法$\mathcal{A}'$。（事实上，这是在本书中唯一使用的证明方法。）因为这个方法是如此重要，所以详细地浏览一遍证明步骤的概要。从假设开始，假设某个难题X不能通过任何多项式时间算法以不是可忽略的概率被解决（从某种准确定义的角度来说）。想要证明某个密码构造Π是安全的（同样，从某种准确定义的角度来说）。做法如下：

(1) 指定某个有效的（即概率多项式时间）敌手$\mathcal{A}$攻击Π。将敌手的成功概率表示成为$\varepsilon(n)$。

(2) 构造一个叫做“规约”的有效算法$\mathcal{A}'$，该算法将敌手$\mathcal{A}$作为子程序来使用，试图解决难题X（如图 3.1 所示）。这里很重要的一点是，$\mathcal{A}'$对$\mathcal{A}$是“如何”工作的一无所知；$\mathcal{A}'$知道的唯一的事情就是$\mathcal{A}$想要攻击Π。所以，指定难题X的某个输入实例x，算法$\mathcal{A}'$将会对$\mathcal{A}$模拟出一个Π的实例，满足：

①就$\mathcal{A}$而言，它和Π交互。更正式地说，当$\mathcal{A}$被作为子程序被$\mathcal{A}'$运行时，$\mathcal{A}$的分布应该和当$\mathcal{A}$自身和Π交互时的分布是相同的（或至少是很接近的）。

---

④ 对那些熟悉$\mathcal{P}$和$\mathcal{NP}$问题的读者，我们标注：对任意消息长度大于密钥长度的加密方案的安全性的无条件证明意味着要证明$\mathcal{P} \neq \mathcal{NP}$。

②如果$\mathcal{A}$成功“攻破了”由$\mathcal{A}'$模拟的Π的实例，则这将允许$\mathcal{A}'$解决给出的难题实例x，成功的概率至少为多项式的倒数，即$1/p(n)$。

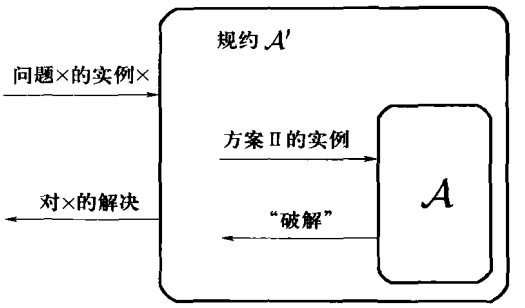


图 3.1　通过规约对安全性的证明的高层次概况

(3) 总起来考虑，(2)①和(2)②意味着，如果$\varepsilon(n)$不是可忽略的，则$\mathcal{A}'$解决难题X的概率为不可忽略的概率$\varepsilon(n)/p(n)$。因为$\mathcal{A}'$是有效的，将 PPT 敌手$\mathcal{A}$作为子程序来运行，意味着存在一个有效的算法能够以不可忽略的概率解决X，与原假设矛盾。

(4) 因而结论是，给定一个关于X的假设，不存在有效的敌手$\mathcal{A}$能够以不可忽略的概率成功攻破Π。换句话说，Π是计算上安全的。

当下一节给出这种证明的例子时，这一点将会更加清楚。

3.2　定义计算安全的加密

上一节介绍的背景已准备好定义对称密钥加密的计算安全。首先，重新定义对称密钥加密的语法；这个和在第 2 章中介绍的语法本质上是相同的，唯一不同是现在要明确考虑安全参数。现在，也令消息空间默认为所有（有限长度的）二进制字符串的集合$\{0,1\}^*$。

定义 3.7　一个对称密钥加密方案是概率多项式时间算法(Gen, Enc, Dec)的三元组：

(1) 密钥生成算法Gen的输入为安全参数1^n，输出密钥k；记为$k \leftarrow \mathsf{Gen}(1^n)$（要强调Gen是一个随机算法）。不失一般性，假设任何由$\mathsf{Gen}(1^n)$输出的密钥k，都满足$|k| \geqslant n$。

(2) 加密算法Enc将密钥k和明文消息$m \in \{0,1\}^*$作为输入，并且输出一个密文c[⑤]。因为Enc可能是随机化的，记为$c \leftarrow \mathsf{Enc}_k(m)$。

(3) 解密算法Dec将密钥k和密文c作为输入，输出一份消息m。假设Dec是确定性的，所以记为$m := \mathsf{Dec}_k(c)$。

需要满足的是：对每个n、每个由$\mathsf{Gen}(1^n)$输出的密钥k、每个$m \in \{0,1\}^*$，都满足$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$[⑥]。

---

⑤ 作为一个技术条件，Enc允许在以$|k|+|m|$（即输入的总长度）为参数的多项式时间内运行。如果仅仅包含$|m|$，那么加密单个比特将不会占用多项式时间，如果仅仅包含$|k|$，则不得不对要加密的消息长度设置一个先验限制。虽然有必要，从这里开始忽略该技术细节。

⑥ 给定这些，我们对Dec是确定的这一假设是不失一般性的。

如果(Gen, Enc, Dec)满足：对每个由$\mathsf{Gen}(1^n)$输出的密钥k，算法Enc_k只对消息$m \in \{0,1\}^{l(n)}$有定义，则假设(Gen, Enc, Dec)是一个消息长度为定长$\ell(n)$的对称密钥加密方案。

需要强调在本章和下一章，$\mathsf{Gen}(1^n)$都是均匀随机地选择$k \leftarrow \{0,1\}^*$。

3.2.1 安全的基本定义

定义对称密钥加密的安全性的标准方式有很多种，主要的不同在于：对攻击中敌手的能力的假设。本节从说明最基本的安全概念开始，即抵抗唯密文攻击这种弱安全，敌手只能观察到“单个”密文，在本章后面，会考虑更强的安全定义。

定义的动机。正如在第 1 章讨论的，任何对安全的定义由两个不同的部分组成：对假定的敌手能力的详细说明，对什么是“攻破”方案的描述。在定义开始之前，考虑一种情况，一个窃听敌手观察到单个消息，或者被给予希望“破解”的单个密文。这类敌手较弱，但是这正是前面章节所考虑的敌手类型（后面章节中会遇到更强的敌手）。当然，正如在前一节中解释的那样，目前只对那种有计算能力限制和运行时间被限制在多项式时间内的敌手感兴趣。

尽管对敌手能力做出了两个重要假设（即仅仅是窃听，在多项式时间内运行），但是这里对敌手的策略没有做出任何假设。这一点对获得有意义的安全概念是很关键的，因为预测所有可能的策略是不可能的。因此，要预防所有可被敌手执行的策略，这些敌手包含在已定义的敌手类型中。

定义对加密算法的“攻破”并不简单，已经在 1.4.1 节和前面一章中详细地讨论过这个问题。因此，这里只需要回忆一下，定义背后的思想是，敌手应该不能从密文中得到关于明文的任何信息。对“语义安全”定义的直接形式化正是这个概念，这是第一个被建议的加密安全的定义。不幸的是，语义安全的定义复杂并且困难。幸运的是，有个等价的定义，即“不可区分性”，它更简单。因为定义是等价的，所以可以从更简单的不可区分性的定义着手，确信获得的安全保证正是想从语义安全中所期望的。（参见 3.2.2 节对这一点的进一步讨论。）

这里给出的不可区分性的定义从语法上说，和在定义 2.4 中给出的完善保密的定义几乎是相同的。（这提供了进一步的定义不可区分性的动机。）回忆定义 2.4 中，考虑一个实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$，一个敌手$\mathcal{A}$输出两个消息$m_0, m_1$，然后敌手被给予其中一个消息加密后的密文，该消息是随机选择的并使用一个随机生成的密钥。定义陈述为：如果没有敌手$\mathcal{A}$能够以任何不同于$1/2$的概率判断出消息m_0, m_1中是哪一个被加密的($1/2$即随机猜测的概率)，则方案Π是安全的。

这里保持$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$基本上不变（除了某些下面讨论的技术区别以外），引入对定义的两个关键改变：

(1) 这里仅仅考虑在多项式时间内运行的敌手，尽管定义 2.4 中考虑无计算能力限制的敌手。

(2) 这里要求敌手可能判断出加密的消息的概率大于$1/2$的程度是可忽略的。

如上节的大量讨论，上面的条件放宽组成了计算安全的核心元素。

至于在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$中的不同，一个区别纯粹是语法上的，而另一个区别是因为技术问题。最突出的区别是，现在用一个安全参数n来将实验参数化。用n的函数来衡量敌手$\mathcal{A}$的运行时间和成功的概率。用$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$来表示给定安全参数值的实验，并且写为

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}=1] \tag{3.1}$$

来表示在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中，$\mathcal{A}$输出1的概率。值得注意的是当$\mathcal{A}$和$\Pi$固定时，等式(3.1)是$n$的函数。

实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$中的第二个区别是，这里需要敌手输出两个长度相等的消息m_0, m_1。从理论的角度讲，该限制是有必要的，因为这里要求加密方法应该能够用来加密任意长度的消息。如果愿意放弃这个要求，就可以不要这个限制，如同完善保密的情况中所做的那样（参见练习 3.3 以及 3.4）。该限制对在实际应用中的大部分加密方案都是适用的，这些加密方案中不同长度的消息导致了不同长度的密文，所以，如果允许输出不同长度的消息，则敌手能够很轻易地分辨出哪一个消息是经过加密的。

大部分实际中的加密方案都没有隐藏加密消息的长度。为了防止一个消息的长度可能会泄露一些敏感信息（比如，当消息指出一个雇员薪水是几数位时），必须在加密之前，填充输入使之达到某个固定长度。在后面将不再重申这一点。

窃听者存在情况下的不可区分性。现在给出正式的定义，从上面概述的实验开始。该实验的定义对任何对称密钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$，任何敌手$\mathcal{A}$，以及对任何安全参数$n$的值都适用：

窃听不可区分实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$：

(1) 给定输入1^n给敌手$\mathcal{A}$，$\mathcal{A}$输出一对长度相等的消息m_0, m_1。

(2) 运行$\mathsf{Gen}(1^n)$生成一个密钥k，选择一个随机比特b，$b\leftarrow\{0,1\}$。一个密文$c\leftarrow\mathsf{Enc}_k(m_b)$被计算并且给$\mathcal{A}$。把$c$叫做挑战密文。

(3) $\mathcal{A}$输出一个比特b'。

(4) 该实验的输出被定义为：如果$b=b'$，则为1，否则为0。如果$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1$，则称$\mathcal{A}$成功。

这里除了要求长度是相等的以外，加密消息m_0和m_1的长度是没有限制的。（当然，因为敌手被限制在多项式时间内运行，m_0和m_1的长度为以n为参数的多项式。）如果Π是一个定长方案，消息长度为$\ell(n)$，那么上面的实验要被修改，需要$m_0, m_1\in\{0,1\}^{\ell(n)}$。

不可区分性的定义说明了：如果在上面的实验中，任何 PPT 敌手的成功概率的最大值高于1/2的部分可忽略，则一个加密方案是安全的。（注意，可以1/2的概率成功是很容易的，因为可以通过输出一个随机比特b'，困难的地方是要比这个做到更好。）现在为下面的定义做好了准备。

定义 3.8 如果对于所有的概率多项式时间敌手$\mathcal{A}$，存在一个可忽略函数negl使得：

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]\leqslant\frac{1}{2}+\mathsf{negl}(n)$$

则一个对称密钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$具备在窃听者存在的情况下不可区分的加密，其中概率的来源是$\mathcal{A}$的随机性以及实验的随机性（选择密钥、随机比特b，以及在加密过程中使用到的任何随机性）。

该定义是对“所有的”概率多项式时间的敌手定量分析，意味着对所有“可行的”策略（可行的策略和那些能够在多项式时间执行的策略是等价的）而言，都是安全的。敌手的输入

被限制为（单个）密文，以及敌手和发送方或接收方没有更多的交互这一事实，隐含着敌手仅仅只有窃听的能力。（稍后会看到，允许更多的交互将会导致更强的敌手。）现在，该定义简短地说明任何敌手$\mathcal{A}$将成功猜出哪一个是加密消息的概率的最大值超出一个单纯的猜测（猜对的概率为1/2）的部分是可忽略的。

注意，敌手是被允许选择消息m_0和m_1。因此，即使知道c是这些明文消息中的某个消息加密的密文，仍然无法判断出是从哪一个加密来的。这是个非常强的保证，有很重要的实际意义。举个例子，考虑一个场景，敌手知道被加密的消息要么是“今天攻击”，或者是“不攻击”。很明显，不希望敌手知道是哪一个消息被加密，尽管已经知道密文是来自两种消息中的某一个。

一个等价的公式化表述。定义 3.8 阐述了一个窃听敌手不能做到：判断是哪一个明文被加密的概率明显大于随机猜测成功的概率。一个公式化该定义的等价方式是：陈述每个敌手，在观察到m_0的密文或者m_1的密文（对任何长度相等的m_0和m_1）时，其行为是相同的。因为$\mathcal{A}$只输出单个比特，“行为相同”意味着在每种情况下，输出1的概率几乎相等。为将其形式化，同上定义$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,b)$，唯一不同是使用一个固定比特$b$（而不是随机的选择）。此外，用$\mathsf{output}\left(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,b)\right)$来表示$\mathcal{A}$在$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,b)$中的输出比特$b'$。接下来的定义从根本上说明了$\mathcal{A}$无法判断是在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0)$中运行还是在$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1)$中运行。

定义 3.9 如果对所有的概率多项式时间的敌手$\mathcal{A}$，存在一个可忽略函数negl使得：

$$\left|\Pr\left[\mathsf{output}\left(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,0)\right)=1\right]-\Pr\left[\mathsf{output}\left(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n,1)\right)=1\right]\right|\leqslant\mathsf{negl}(n)$$

则对称密钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$具有窃听者存在的情况下不可区分性。

该定义和定义 3.8 等价的事实留作练习。

3.2.2 *定义的属性

通过要求敌手无法从密文中掌握任何关于明文的部分消息，给出了安全加密的定义。但是，不可区分性的实际定义看起来非常不同。如前所述，定义 3.8 实际上和“语义安全”是等价的，语义安全将部分消息不能被得到这一直觉概念进行了形式化地描述。这里将不证明等价性，但将证明两个声称（Claim）展示了不可区分性意味着语义安全的弱版本。为了比较，（从根本上）表述语义安全的完整定义。读者可以参考文献[65]的 5.2 章得到更深入的讨论以及完整的等价性证明。

不可区分性意味着一个随机选择的明文中没有单个比特能够被猜出的概率显著大于1/2。下面，我们用m^i来表示m中的第i个比特，如果$i>|m|$，令$m^i=0$。

声称 3.10 令$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$为一个对称密钥加密方案，在窃听者存在的情况下，它具备不可区分的加密，那么对于所有的概率多项式时间敌手$\mathcal{A}$和所有i，存在一个可忽略函数negl，使得

$$\Pr\left[\mathcal{A}\left(1^n,\mathsf{Enc}_k(m)\right)=m^i\right]\leqslant\frac{1}{2}+\mathsf{negl}(n)$$

其中m是从$\{0,1\}^n$中均匀随机选择的，并且概率来源于$\mathcal{A}$的随机性、m和k选择的随机性以及

在加密过程中使用到的任何随机性。

证明 证明背后的思想是给出$\mathsf{Enc}_k(m)$，如果猜出m的第i个比特是可能的，则区分明文消息m_0和m_1的密文也是可能的，其中m_0的第i个比特等于0，m_1的第i个比特等于1。特别地，给定一个密文c，尝试计算出对应明文的第i个比特。如果这个计算表明第i个比特为0，则猜测是m_0被加密；如果表明第i个比特为1，则猜测是m_1被加密。正式的描述是，给定$\mathsf{Enc}_k(m)$，对某个函数$\varepsilon(n)$，如果存在一个敌手$\mathcal{A}$能够在概率至少为$1/2+\varepsilon(n)$的情况下，猜出m的第i个比特，则存在一个敌手能够在不可区分性实验$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$中成功的概率为$1/2+\varepsilon(n)$。如果$\Pi$是不可区分的加密，则$\varepsilon(\cdot)$必须是可忽略的。

具体而言，令$\mathcal{A}$是一个概率多项式时间敌手，定义$\varepsilon(\cdot)$如下：

$$\varepsilon(n) \stackrel{\mathsf{def}}{=} \Pr\left[\mathcal{A}\left(1^n,\mathsf{Enc}_k(m)\right)=m^i\right]-\frac{1}{2}$$

其中m是从$\{0,1\}^n$中均匀随机选择的。从这里开始，为了视觉上更清晰，不再明确指出输入1^n给$\mathcal{A}$。令$n\geqslant i$，令I_0^n为所有长度为n，且第i个比特为0的字符串的集合，I_1^n为所有长度为n，且第i个比特为1的字符串的集合。有

$$\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m)\right)=m^i\right]=\frac{1}{2}\cdot\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m_0)\right)=0\right]+\frac{1}{2}\cdot\Pr[\mathcal{A}\left(\mathsf{Enc}_k(m_1)\right)=1]$$

其中，m_0是从I_0^n中随机选择的，m_1是从I_1^n中随机选择的。（上面的条件满足，因为I_0^n和I_1^n每一个都正好包含$\{0,1\}^n$的一半。因此，m在每个集合中的概率正好是$1/2$。）

现在，考虑下面的敌手$\mathcal{A}'$，窃听单个消息的加密：

敌手$\mathcal{A}'$：

(1) 输入1^n（$n\geqslant i$），从指定的集合中均匀随机地选择$m_0\leftarrow I_0^n$和$m_1\leftarrow I_1^n$。输出m_0,m_1。

(2) 当接收到密文c时，输入c给敌手$\mathcal{A}$。如果$\mathcal{A}$输出0，输出$b'=0$，如果$\mathcal{A}$输出1，输出$b'=1$。

因为$\mathcal{A}$在多项式时间运行，所以$\mathcal{A}'$在多项式时间内运行。

使用实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)$的定义（$n\geqslant i$），注意当且仅当$\mathcal{A}$收到$\mathsf{Enc}_k(m_b)$时，输出$b$，有$b'=b$。所以

$$\begin{aligned}\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)=1\right]&=\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m_b)\right)=b\right]\\&=\frac{1}{2}\cdot\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m_0)\right)=0\right]+\frac{1}{2}\cdot\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m_1)\right)=1\right]\\&=\Pr\left[\mathcal{A}\left(\mathsf{Enc}_k(m)\right)=m^i\right]\\&=\frac{1}{2}+\varepsilon(n)\end{aligned}$$

通过假设$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$在窃听者存在的情况下，具备不可区分加密，则$\varepsilon(\cdot)$必须是可忽略的。（注意，当$n<i$时，无论发生什么都不重要，因为这里只关心渐进行为。）得证。 ■

粗略地说，给定密文，PPT 敌手不能推测关于明文消息的任何函数，而且不管发送消息的先验分布如何，该条件都满足。形式化地定义这个要求并不简单。例如要计算明文消息m的第i个比特（如同上面所考虑的），若m从集合I_0^n中均匀地选取（而不是从$\{0,1\}^n$中均匀地

选取），则将非常容易计算。这里需要定义的是对某个函数f，如果一个接收到密文$c=\mathsf{Enc}_k(m)$的敌手能够计算出$f(m)$，则存在一个敌手能够在不给密文的（仅仅知道m的先验分布）情况下，以同样的概率正确地计算出$f(m)$。

在下一个声称中，当m均匀随机地从某个集合$S\subseteq\{0,1\}^n$中选择时，也满足上面的这些讨论。即如果明文是邮件信息，则令S是有正确邮件头的英语消息的集合。由于考虑渐进情形，故着手于无穷集合$S\subseteq\{0,1\}^*$。对于安全参数n，从$S_n\stackrel{\text{def}}{=}S\cap\{0,1\}^n$中均匀选出一个明文消息，该集合默认为是非空明文消息的集合（即S的子集，S中长度为n的字符串的集合）。作为一个技术上的条件，需要假设能够有效且均匀地从S_n中采样字符串；也就是说，存在某个概率多项式时间算法，对于输入1^n，输出S_n中一个均匀分布的元素。这样，认为集合S是“有效可采样的”。这里将函数f限制为能够在多项式时间内计算。

声称 3.11 令(Gen, Enc, Dec)为一个对称密钥加密方案，具备在窃听者存在的情况下不可区分加密，那么对任意概率多项式时间敌手$\mathcal{A}$，存在一个概率多项式时间算法$\mathcal{A}'$，使得对所有的多项式时间可计算函数f，以及所有有效可采样的集合S，存在一个可忽略函数negl，满足：

$$\left|\Pr\left[\mathcal{A}\left(1^n,\mathsf{Enc}_k(m)\right)=f(m)\right]-\Pr\left[\mathcal{A}'\left(1^n\right)=f(m)\right]\right|\leqslant\mathsf{negl}(n)$$

其中m是从$S_n\stackrel{\text{def}}{=}S\cap\{0,1\}^n$中均匀随机地选取，并且概率计算来源于对$m$和密钥$k$的选择、任何$\mathcal{A}$、$\mathcal{A}'$以及加密过程用到的随机性。

证明（概要） 仅仅陈述一个非正式的对该声称的证明概要。假设(Gen, Enc, Dec)具备不可区分性的加密。这就意味着，对任何$m\in\{0,1\}^n$，没有概率多项式时间的敌手$\mathcal{A}$能够区分$\mathsf{Enc}_k(m)$和$\mathsf{Enc}_k(1^n)$。那么给定$\mathsf{Enc}_k(m)$，$\mathcal{A}$成功计算出$f(m)$的概率，与给定$\mathsf{Enc}_k(1^n)$，$\mathcal{A}$成功计算出$f(m)$的概率，两者几乎是相等的。否则，$\mathcal{A}$就能被用来区分$\mathsf{Enc}_k(m)$和$\mathsf{Enc}_k(1^n)$。（区分器很容易构造：均匀随机地选择$m\leftarrow S_n$，这里假设S是有效可采样的，输出$m_0=m,m_1=1^n$。当给定一个密文c，c要么是m的密文，要么是1^n的密文，以c为输入来调用$\mathcal{A}$，当且仅当$\mathcal{A}$输出$f(m)$时输出0。当给定m的密文的概率与1^n的密文的概率存在非可忽略的差别时，$\mathcal{A}$输出$f(m)$，则描述的区分器违背了定义 3.9）。

上面的观察产生了下面的算法$\mathcal{A}'$，该算法不接收$c=\mathsf{Enc}_k(m)$，但是同样能计算$f(m)$：输入安全参数1^n，$\mathcal{A}'$选择随机密钥k，$c\leftarrow\mathsf{Enc}_k(1^n)$作为输入调用$\mathcal{A}$，$\mathcal{A}$的输出作为$\mathcal{A}'$的输出。以上，当$\mathcal{A}$作为一个子程序被$\mathcal{A}'$调用时，输出$f(m)$的概率几乎等于收到$\mathsf{Enc}_k(m)$的概率。因此，$\mathcal{A}'$实现了该声称需要的属性。 ■

语义安全。对语义安全的完整定义，比声称 3.11 证明的属性更一般。这里考虑了明文消息的任意分布，也考虑了任意“外部的”关于明文的信息，这些信息有可能通过其他的方式被“泄露”给敌手（比如，因为相同的消息m也被用作其他用途）。同上，用f来表示敌手试图计算出的明文函数。不考虑一个特定的集合S（以及一个在S的子集中的均匀分布），而是考虑一个任意分布$X=(X_1,X_2,\cdots)$，对安全参数n，明文是根据分布X_n来选择的。要求X是有效可采样的，这意味着存在一个 PPT 算法，对输入1^n，输出一个元素，该元素根据分布X_n来选择。另外，所有的n和所有在X_n中的字符串都有相同的长度。最后，除了m的加密之外，对某个任意函数h，通过给予敌手$h(m)$，来模拟“外部的”关于m信息被“泄露”给敌手。即：

定义 3.12 一个对称密钥加密方案(Gen, Enc, Dec)。如果对于每个概率多项式时间算法$\mathcal{A}$，存在一个概率多项式时间算法$\mathcal{A}'$，使得对所有的有效可采样的分布$X=(X_1,\cdots)$以及所有多项式时间可计算的函数f和h，存在一个可忽略函数negl满足：

$$\left|\Pr\left[\mathcal{A}\left(1^n,\mathsf{Enc}_k\left(m\right),h\left(m\right)\right)=f\left(m\right)\right]-\Pr\left[\mathcal{A}'\left(1^n,h\left(m\right)\right)=f\left(m\right)\right]\right|\leqslant \mathsf{negl}\left(n\right)$$

则称其为在窃听者存在的情况下是语义安全的。其中m是根据X_n分布来选择的，概率计算来源于对m和密钥k的选择、任何$\mathcal{A}$、$\mathcal{A}'$以及加密过程用到的随机性。

给予敌手$\mathcal{A}$密文$\mathsf{Enc}_k\left(m\right)$和历史函数$h\left(m\right)$，这里后一个函数表示的是敌手可能掌握的关于明文$m$的“外部的”知识。敌手$\mathcal{A}$试图去猜测$f\left(m\right)$的值。算法$\mathcal{A}'$也试图去猜测$f\left(m\right)$的值，但是仅仅只给予$h\left(m\right)$。安全需求陈述的是，当给予密文时，$\mathcal{A}$猜测$f\left(m\right)$的成功概率和某个不给予密文的算法$\mathcal{A}'$本质上是一样的。因此，密文$\mathsf{Enc}_k\left(m\right)$没有泄露任何关于$f\left(m\right)$的信息。

定义 3.12 组成了一个非常强且令人信服的对安全保证的形式化描述，这个安全保证由加密方案提供。可论证地，这个比不可区分性（仅仅考虑两个明文，没有提到外部知识或者任意分布）更令人信服。但是，从不可区分性定义着手更容易（比如，证明一个指定方案是安全的）。幸运的是，已证明两个定义是等价的：

定理 3.13 一个对称密钥加密方案具备窃听者存在情况下不可区分性加密，当且仅当该方案窃听者存在的情况下是语义安全的。

往后，类似的等价性适用于本章所有关于不可区分性的定义。因而通常使用不可区分性作为工作定义，而达到的安全保证是语义安全的。

3.3 伪 随 机 性

前面已经定义什么是加密方案的安全性，读者可能期望即刻开始构造一个安全加密方案。但是，在此之前，需要引入伪随机性的概念。该概念在密码学中扮演了一个很重要的角色，特别是在对称密钥加密中。不严格地说，一个伪随机的字符串（均为比特字符串）是看起来很像均匀分布的字符串，只要这个“看”的实体是在多项式时间内运行。如同不可区分性可视为完善保密在计算上的松弛，伪随机是真正的随机性在计算上的松弛。

从技术上来说，一个重要的概念是没有固定的字符串能够被认为是“伪随机的”（类似地，认为任何固定的字符串是“随机的”没有意义）。伪随机性实际上指的是字符串的分布，当声明一个长度为ℓ的字符串的分布$\mathcal{D}$是伪随机的，就意味着长度为ℓ的字符串的均匀分布和分布$\mathcal{D}$是不可区分的。（严格地说，在渐进的情形中，实际上是分布$\mathcal{D}=\{\mathcal{D}_n\}$的一个序列的伪随机性，这里$\mathcal{D}_n$和安全参数有关。在当前讨论中忽略这一点。）更精确地，对任何多项式时间算法来说，分辨出它是分布$\mathcal{D}$的字符串采样，还是一个均匀随机选择的ℓ比特长的字符串，是不可行的。

即使给出上面的讨论，这一概念经常滥用，把一个根据均匀分布采样的字符串叫做“随机字符串”，并且把一个根据伪随机分布$\mathcal{D}$采样的字符串叫做“伪随机字符串”。这里只是一种有用的速记法。

在继续之前，先提供一些直观的感受，为什么伪随机性能帮助构造安全的对称密钥加密方案。从一个简单的层面来讲，如果一个密文看起来随机，则很清楚的是，没有敌手能够从密文中推测任何关于明文的信息。在某种程度上，这是位于“一次一密”的完善保密性之后的准确直觉。如果是这样的话，该密文是均匀分布的（假设密钥未知），所以没有泄漏关于明文的任何信息。（当然，这样的陈述只是直觉上有吸引力，不能构成一个正式的论证。）“一次一密”是计算一个随机字符串（密钥）和明文的异或。如果一个伪随机字符串被使用，在一个多项式时间敌手看来，应该无法觉察到任何区别。因此，对多项式时间敌手来说，安全应该仍然满足。

如下所述，该想法可以实现。使用一个伪随机字符串而不是真正的随机字符串的好处在于，一个长的伪随机字符串能够通过一个相对短的随机种子（或者密钥）来生成。因此，一个短的密钥能用来加密一个长消息，这个完善保密不能办到的事情，现在可以办到了。

伪随机发生器。非正式地说，如上讨论，如果没有多项式时间的区分器能够区分一个根据D采样的字符串和一个均匀随机选择的字符串，则一个分布D是伪随机的。和定义 3.9 相似，可形式化为，当给予一个真正的随机字符串，以及一个伪随机字符串时，要求每个多项式时间算法输出1的概率相等（该输出比特被理解成为算法的“猜测”）。一个“伪随机发生器”是一个确定算法，接受一个短的真正随机的种子，然后将其扩展成为一个长的伪随机字符串。在下面的定义中，令n为种子的长度，要输入给发生器，并且令$\ell(n)$为输出长度。很明显，该发生器只有在$\ell(n) > n$时，才会有意义（否则，它不会产生任何新的“随机性”）。

定义 3.14 令$\ell(\cdot)$为多项式，令G为确定多项式时间算法，该算法满足：对于任何输入$s \in \{0,1\}^n$，算法G输出一个长度为$\ell(n)$的字符串。如果满足下面两个条件，则称G是一个伪随机发生器：

(1)（扩展性：）对每个n来说，满足$\ell(n) > n$。

(2)（伪随机性：）对所有的概率多项式时间的区分器D来说，存在一个可忽略函数negl，满足

$$|\Pr[D(r)=1] - \Pr[D(G(s))=1]| \leqslant \mathsf{negl}(n)$$

其中r是从$\{0,1\}^{\ell(n)}$中均匀随机选择的，种子s是从$\{0,1\}^n$中均匀随机选择的，并且概率来源为D的随机性和对r，s的选择。

函数$\ell(\cdot)$被称为G的扩展系数。

讨论。需要强调，一个伪随机发生器的输出实际上和随机输出相差很多。为了看清这一点，考虑当$\ell(n) = 2n$的情况，此时G的输入变成原来的两倍。对$\{0,1\}^{2n}$的均匀分布，2^{2n}中每个可能的字符串被选择的概率正好为2^{-2n}。相反，考虑由G生成的分布。因为G接收长度为n的输入，不同的可能字符串的个数的范围最多为2^n。因此，一个长度为$2n$的随机字符串在G的范围中的概率为$2^n/2^{2n} = 2^{-n}$（用所有在G的范围中的字符串的总数，除以长度为$2n$的字符串的数量）。即大部分长度为$2n$的字符串不会作为G的输出。

这特别意味着如果给定的时间无限制，区分随机字符串和伪随机字符串是很容易的。考虑下面指数时间的区分器D的如下工作：对某个输入字符串w，当且仅当存在一个$s \in \{0,1\}^n$，满足$G(s) = w$时，区分器D输出1。（通过搜索所有的$\{0,1\}^n$，对每个

$s \in \{0,1\}^n$计算$G(s)$来执行该计算。该计算能够执行是因为G的工作描述是已知的；只有它的种子是未知的。）现在，如果w是由G生成的，那么它满足D输出1的概率为1。相反地，如果w是均匀分布在$\{0,1\}^{2n}$中的，则存在一个s使得$G(s)=w$的概率最多为2^{-n}，在这种情况下，D输出1的概率最多为2^{-n}。因此我们有

$$|\Pr[D(r)=1]-\Pr[D(G(s)=1]| \geqslant 1-2^{-n}。$$

这个结果很大。该类型的攻击被叫做蛮力攻击，因为它试过了所有的可能种子。这类“攻击”的优势在于对所有的发生器都适用，和它们是如何工作的无关。

以上说明G产生的分布不是均匀的，但多项式时间的区分器没有时间来执行上面的步骤。如果G是一个伪随机发生器，则它保证了不存在任何多项式时间的步骤能够区分随机字符串和伪随机字符串。这意味着伪随机字符串和真正随机的字符串一样，只要种子是保密的，且只考虑多项式时间的观察者（敌手）。

种子和它的长度。一个伪随机发生器的种子必须被均匀地随机选取，且对区分器来说是完全保密的。从上面的蛮力攻击可看到另外一个关键点是，s必须足够的长，以至于没有“有效的算法”有时间去遍历所有可能的种子。技术上，所有算法假设都是在多项式时间内运行，因此当n足够大时，算法不可能搜索到所有2^n个可能的种子。但是实际上，种子必须被认为具有某个具体的长度。鉴于此，s至少必须足够长，以至于尝试所有的可能种子是不可能的。

伪随机发生器的存在性。一个首要问题应该是满足定义 3.14 的实体是否真的存在。不幸的是，目前不知道如何证明伪随机发生器的存在性。尽管如此，本书相信伪随机发生器真的存在，并且这种相信基于一个事实：伪随机发生器能够在相当弱的假设下被构造出来（从可证明的意义上），这个相当弱的假设是单向函数是存在的。这将在第 6 章更详细地讨论。就当前而言，有充分的理由说：存在某个已经研究了很久的问题，该问题还没有已知的有效算法，并且该问题被广泛地认为在多项式时间内是无法解决的。这类问题的一个例子是大数分解问题：即找到一个给定整数的素因子。单向函数，从而伪随机发生器能够在这些难题真的很“困难”的假设下被构造出来。

实际中，相信是伪随机发生器的各种构造方法是已知的。事实上，正如稍后将在本章和第 5 章看到的，这些构造方法是存在的，并且被认为满足甚至更强的要求。

3.4 构造安全加密方案

3.4.1 一个安全的定长加密方案

现在已准备好去构造一个定长的加密方案，该方案在有窃听者存在的情况下，不可区分加密。这里构造的加密方案和“一次一密”加密方案很相似（参见 2.2 节），除了这里使用的是一个“伪随机字符串”而不是一个随机字符串作为“填充”。由于一个伪随机字符串对任何多项式时间敌手来说“看起来随机”，故该加密方案能被证明是计算安全的。

加密方案。令G是一个扩展因子为ℓ的伪随机发生器（即$|G(s)|=|\ell(|s|)|$）。回想一个加密方案被三个算法定义：一个密钥生成算法Gen，一个加密算法Enc以及一个解密算法Dec。加密过程是，密钥（作为一个种子）应用到一个伪随机发生器来获得一个长的填充，然后和

明文消息作异或处理。该方案在构造方法 3.15 中正式地描述，如图 3.2 所示。

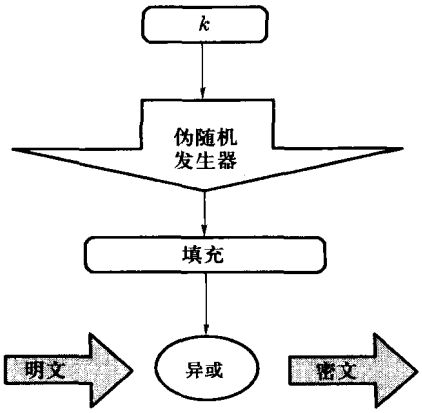


图 3.2　使用伪随机发生器的加密

构造方法 3.15

令G是一个带扩展因子ℓ的伪随机发生器。定义一个消息长度为ℓ的对称密钥加密方案如下：

- Gen：输入1^n，均匀随机地选择$k \leftarrow \{0,1\}^n$，并将其作为密钥输出。
- Enc：输入一个密钥$k \in \{0,1\}^n$，以及一个消息$m \in \{0,1\}^{\ell(n)}$，输出密文
$$c := G(k) \oplus m。$$
- Dec：输入一个密钥$k \in \{0,1\}^n$，以及一个密文$c \in \{0,1\}^{\ell(n)}$，输出明文消息
$$m := G(k) \oplus c。$$

一个使用任何伪随机发生器的对称密钥加密方案

现在证明在窃听者存在的情况下，若假设G是伪随机发生器，则给定的加密方案具备不可区分性。注意这里的声称不是无条件的，从而，将加密方案的安全规约到伪随机发生器G的属性。这是一个在 3.1.3 节中描述的非常重要的证明技术，证明后将进一步说明。

定理 3.16　若G是一个伪随机发生器，则构造方法 3.15 是一个在窃听者存在的情况下，具备不可区分加密的定长对称密钥加密方案。

证明　令Π表示构造方法 3.15。需要说明如果存在一个概率多项式时间的敌手$\mathcal{A}$，对其而言定义 3.8 不满足，则可以构造一个概率多项式时间算法将真正的随机字符串和G的输出区分开。这背后的直觉是，如果Π使用一个真正的随机字符串来代替伪随机字符串$G(k)$，则得到的方案将会和“一次一密”加密方案相同，$\mathcal{A}$将不能够以大于$1/2$的概率正确猜出是哪一个消息被加密。所以，如果定义 3.8 不满足，则$\mathcal{A}$必能（隐含地）区分真正的随机字符串和G的输出。下面的规约将使之更清晰。

令$\mathcal{A}$为一个概率多项式时间的敌手，定义ε为

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr\left[\mathsf{PrivK}^{\mathsf{eva}}_{\mathcal{A},\Pi}(n) = 1\right] - \frac{1}{2} \tag{3.2}$$

使用$\mathcal{A}$来构造一个伪随机发生器G的区分器D，使得D“成功”的概率为$\varepsilon(n)$。该区分器

被给定一个字符串w作为输入，并且它的目标是判断w是均匀随机的选出（即w是“随机字符串”），还是w通过选择一个随机k，然后通过$w=G(k)$计算出的（即w是“伪随机字符串”）。D用下面描述的方式来模拟$\mathcal{A}$的窃听实验，并且观察$\mathcal{A}$是否成功。如果$\mathcal{A}$成功，则D猜测w是一个伪随机字符串；如果$\mathcal{A}$没有成功，则D猜测w是一个随机字符串。具体如下：

区分器D：

D被指定了一个字符串$w\in\{0,1\}^{\ell(n)}$作为输入。（假设n能够被$\ell(n)$确定。）

(1) 运行$\mathcal{A}(1^n)$来获得一对消息$m_0,m_1\in\{0,1\}^{\ell(n)}$。

(2) 随机选择一个比特$b\leftarrow\{0,1\}$。令$c:=w\oplus m_b$。

(3) 把c给$\mathcal{A}$，得到输出b'。如果$b'=b$，则输出1，否则输出0。

在分析区分器D的行为之前，定义一个修改的加密方案$\widetilde{\Pi}=\left(\widetilde{\mathsf{Gen}},\widetilde{\mathsf{Enc}},\widetilde{\mathsf{Dec}}\right)$正好是“一次一密”加密方案，唯一不同的是现在包含一个安全参数，该安全参数确定了被加密消息的长度。即$\widetilde{\mathsf{Gen}}(1^n)$输出一个长度为$\ell(n)$的完全随机的密钥$k$，并且使用密钥$k\in\{0,1\}^{\ell(n)}$加密消息$m\in\{0,1\}^{\ell(n)}$，得到密文$c:=k\oplus m$。（解密照常，但对接下来的讨论不重要。）根据“一次一密”的完善保密性，有

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\pi}}(n)=1\right]=\frac{1}{2} \tag{3.3}$$

在对D的行为的分析中，主要的观察如下：

(1) 如果w均匀随机地从$\{0,1\}^{\ell(n)}$中选择，则当$\mathcal{A}$作为一个子程序被D运行的时候，$\mathcal{A}$所见内容（称为视图）的分布，与$\mathcal{A}$在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$中的所见内容的（视图）分布是相同的。这是因为$\mathcal{A}$被给定一个密文$c=w\oplus m_b$，其中$w\in\{0,1\}^{\ell(n)}$是一个完全随机的字符串。因此，结果就是均匀随机选择$w\leftarrow\{0,1\}^{\ell(n)}$，

$$\Pr[D(w)=1]=\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}=1\right]=\frac{1}{2}$$

其中第二个等式由式（3.3）而得。

(2) 如果w等于$G(k)$，$k\leftarrow\{0,1\}^n$是随机均匀地选择，则当$\mathcal{A}$作为一个子程序被D运行时的视图分布，和$\mathcal{A}$在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中的视图分布是相同的。这是因为$\mathcal{A}$被指定了一个密文$c=w\oplus m_b$，其中$w=G(k)$，这里$k\leftarrow\{0,1\}^n$是一个均匀分布的值。因此，当$w=G(k)$，$k\leftarrow\{0,1\}^n$是均匀随机地选择，有

$$\Pr[D(w)=1]=\Pr[D(G(k))=1]=\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]=\frac{1}{2}+\varepsilon(n)$$

其中第二个等式是依据ε的定义。

因此有

$$|\Pr[D(w)=1]-\Pr[D(G(k))=1]|=\varepsilon(n)$$

这里，w是从$\{0,1\}^{\ell(n)}$中均匀随机选择的，并且k是从$\{0,1\}^n$中均匀随机选择的。假设G是伪随机发生器，那么ε必须是可忽略的。通过ε的定义（参见式（3.2）），意味着Π在窃听者存在的情况下，具备不可区分的加密，证毕。 ■

很容易在证明的细节中迷失，也很容易感到疑惑，与“一次一密”相比，得到了什么；

毕竟，“一次一密”也是通过异或一个ℓ比特的消息和一个ℓ比特的字符串来加密。该构造方法的关键是，ℓ比特的字符串$G(k)$可比密钥k长很多。特别地，使用上面的加密方案加密一个几兆字节长的文件，仅使用 128 比特的密钥是可能的。这和定理 2.7 的陈述完全相反：完善保密的加密方案中，密钥必须至少和被加密的消息一样长。基于计算的方法得到的比完善保密更多。

规约——讨论。构造 3.15 不能无条件地证明是安全的。相反，在假设G是一个伪随机发生器的情况下，可证明是安全的。将构造的安全性规约到某个基础原语的方法是非常重要的，原因如下。首先，正如已经注意到的，目前还不知道如何证明满足定义 3.8 的加密方案的无条件存在性，并且这样的证明是目前不可能达到的。鉴于此，将高级构造方案的安全性规约到一个低级原语有很多好处（正如在 1.4.2 节讨论的那样）。好处之一就是，通常设计一个低级原语比高级构造方案要简单；相似地，与高级定义相比，满足低级定义更容易让人信服。这并不意味着构造一个伪随机发生器是“容易的”，构造发生器仅仅是比从最开始构造一个加密方案容易。（当然，在现在的情况下，加密方案几乎什么都不做，除了一个异或操作，该异或操作将伪随机发生器的输出和消息做异或，所以，这个说法并不是真正正确。但是，后面将看到更复杂的构造方法，在这些情况下，将这些任务规约到一个更简单任务的能力是非常重要的。）

3.4.2 处理变长消息

上一节的构造方法存在一个缺点，就是仅仅只对定长的消息加密。（即对每个给定的安全参数n，只有长度为$\ell(n)$的消息能被加密。）该缺陷能够通过使用一个在构造方法 3.15 中的输出长度可变的伪随机发生器来轻松解决。

输出长度可变的伪随机发生器。在某些应用中，不能事先知道需要多少比特的伪随机序列。因此，实际上需要这样一个伪随机发生器，该发生器能够输出一个任何希望长度的伪随机字符串。具体而言，希望G接受两个输入：种子s以及输出长度ℓ（该长度ℓ以一进制的形式给出，因为安全参数是一进制的）；G应该输出一个长度为ℓ的伪随机字符串。正式的定义如下：

定义 3.17 一个确定的多项式时间算法G，如果满足以下三个条件，则G是一个输出长度可变的伪随机发生器，如果满足以下条件：

(1) 令s为一个字符串，整数$\ell > 0$。则$G\left(s, 1^{\ell}\right)$输出一个长度为$\ell$的字符串。

(2) 对所有的s, ℓ, ℓ'，$\ell < \ell'$，字符串$G\left(s, 1^{\ell}\right)$是$G\left(s, 1^{\ell'}\right)$的前缀[7]。

(3) 定义$G_{\ell}(s) \stackrel{\text{def}}{=} G\left(s, 1^{\ell(|s|)}\right)$。则对于每个多项式$\ell(\cdot)$，有$G_{\ell}$是一个扩展因子为$\ell$的伪随机发生器。

任何标准的伪随机发生器（参见定义 3.14）能够被转化成为一个输出长度可变的伪随机发生器，参见 6.4.2 节。

---

[7] 需要该条件是基于技术理由，为了证明构造方法 3.15 在使用变长发生器时，对变长消息加密的安全性。

给出上面的定义后，很自然地修改构造方法 3.15：使用密钥k加密一个消息是通过计算密文$c := G\left(k, 1^{|m|}\right) \oplus m$来完成的；用密钥$k$解密一个消息是通过计算消息$m := G\left(k, 1^{|c|}\right) \oplus c$来完成的。将满足定义 3.8 的方案的证明留作练习。证明过程参照定理 3.16，除了某些技术上的细微差别。

3.4.3 流密码和多个加密

在密码学的文献资料中，在前两节陈述的加密方案的类型经常被称为“流密码”。这是因为加密的执行是通过首先生成一个伪随机比特流，然后将该比特流和明文做异或运算。不幸的是，这里有一点困惑，就是术语“流密码”指的是生成比特流的算法（即伪随机发生器G），还是整个加密方案。这是一个很关键的问题，因为使用伪随机发生器的方式直接决定了指定的加密方案是否安全。本书认为，术语流密码最好是指生成伪随机流的算法，因此，一个“安全的”流密码应该满足一个输出长度可变的伪随机发生器[⑧]的定义。使用这个术语，一个流密码本质上不是一个加密方案，而是一种构造加密方案的工具。在下面讨论多个加密问题时，这个讨论的重要性就会显而易见了。

实际应用中的流密码。实践中的流密码构造方法有很多种，并且这些一般都是非常快的。一个流行的例子就是流密码 RC4，当合理使用时（参见后面），RC4 被广泛认为是安全的。实际中，流密码的安全性不是很好理解，特别是和分组密码（稍后在本章进行介绍）进行比较时。事实证明：不存在一种标准的、流行的流密码被使用多年，且安全性没有出现问题。例如，“简单的”RC4（在某一点上被认为是安全的，仍然被广泛地部署）已知存在一些重要的弱点。其中之一，由 RC4 生成的输出流的前几个字节已经被证明是有偏离的。尽管看起来这不是很严重，但是已经被证实该缺陷能够被用来轻易地攻破在 802.11 无线网络中使用的 WEP 加密协议。（WEP 是保护无限通信的标准化协议。WEP 标准已经升级来修补该问题。）如果 RC4 被使用，输出的字符串最前面的 1024 比特应该被丢弃。

线性反馈移位寄存器（LFSRs）在历史上曾经是很流行的流密码。但是它们被证明非常不安全（在某种程度上，给定足够多字节的输出，密钥能够完全还原），所以应该永远不再使用。

一般主张使用分组密码来构造安全加密方案。对除了大部分资源受限的环境之外的所有场合，分组密码都是足够高效的，看起来比现有的流密码要安全得多。而且，流密码可从分组密码简单地构造出来，正如下面 3.6.4 节所描述的。该方法的缺陷是，和一个专门的流密码相比，其效率较低。

多个加密的安全性。定义 3.8 以及到目前为止所有的讨论，已经处理了敌手收到单个密文的情况。在现实中，通信双方发送多个密文，一个窃听者将会看到很多密文。因而很重要的一点是，即使在这样的情况下，也要确保加密方案的安全性。

首先给出一个安全定义。如定义 3.8，首先引入一个合适的实验，该实验为了加密方案Π而定义，一个敌手$\mathcal{A}$，以及一个安全参数n。

多消息窃听实验$\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$：

---

⑧ 很快将引入分组密码的概念。在那些内容中，该术语指的是工具本身，而不是如何使用以便实现安全加密。因此类似地，更愿意使用术语“流密码”。

(1) 敌手$\mathcal{A}$被给定输入1^n，输出一对消息向量$\vec{M}_0=(m_0^1,\cdots,m_0^t)$以及$\vec{M}_1=(m_1^1,\cdots,m_1^t)$，对于所有$i$，满足$|m_0^i|=|m_1^i|$。

(2) 通过运行$\mathsf{Gen}(1^n)$生成一个密钥k和选择一个随机比特$b\leftarrow\{0,1\}$。对于所有i，计算密文$c^i\leftarrow\mathsf{Enc}_k(m_b^i)$，并且将密文向量$\vec{C}=(c^1,\cdots,c^t)$给$\mathcal{A}$。

(3) $\mathcal{A}$输出一个比特b'。

(4) 如果$b'=b$，该实验的输出为1；否则输出0。

除了现在所指的是上面的实验，对多消息的安全定义和单个消息的安全定义其实是相同的。

定义 3.18 一个对称密钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$，如果对所有的概率多项式时间敌手$\mathcal{A}$，存在一个可忽略函数$\mathsf{negl}$，满足

$$\Pr\left[\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right]\leqslant\frac{1}{2}+\mathsf{negl}$$

则称其具备窃听者存在的情况下不可区分多次加密，其中概率来源于$\mathcal{A}$使用的随机性，以及在实验中使用的随机性（选择密钥、随机比特以及加密本身）。

一个关键的观察是：单个加密的安全（定义 3.8）并不意味着在多次加密下的安全。下面的命题给出正式的论述。

命题 3.19 存在这样的对称密钥加密方案，满足窃听者存在情况下不可区分的加密，但不满足窃听者存在情况下不可区分的多次加密。

证明 没必要寻找一个加密方案来满足该命题。特别地，在定理 3.16 中已经证明了构造方法 3.15 的单次加密的安全性，该构造用于多次加密时，是不安全的。对此不必觉得意外，因为前面已经看到只有在使用一次时，“一次一密”才是安全的，对构造方法 3.15 而言也是相同的道理。

具体而言，考虑下面的攻击加密方案（在某种意义上，通过$\mathsf{PrivK}^{\mathsf{mult}}$来定义）敌手$\mathcal{A}$：$\mathcal{A}$输出向量$\vec{M}_0=(0^n,0^n)$以及$\vec{M}_1=(0^n,1^n)$。也就是，第一个向量包含了两个明文，每个明文都是长度为n的全零字符串。相比之下，在第二个向量中，第一个明文是全零，第二个明文是全1。现在，令$\vec{C}=(c^1,c^2)$为$\mathcal{A}$接收到的密文向量。如果$c^1=c^2$，则$\mathcal{A}$输出0；否则，$\mathcal{A}$输出1。

现在分析$\mathcal{A}$成功猜测b的概率。关键之处是构造方法 3.15 是确定的，所以如果相同消息被多次加密，则会导致每次都是相同密文。现在，如果$b=0$，则每次都是相同消息被加密（因为$m_0^1=m_0^2$）；因此，$c^1=c^2$，$\mathcal{A}$在这种情况下，总是输出0。另一方面，如果$b=1$，则每次加密不同的消息（因为$m_0^1\neq m_0^2$），并且$c^1\neq c^2$；这里，$\mathcal{A}$总是输出1。因此，$\mathcal{A}$输出$b'=b$的概率为1，根据定义 3.18，该加密方案并不安全。 ■

命题 3.19 的证明可能看起来是人为的，但事实并非如此。相同的消息被重发能够提供重要的信息，在历史上对密码分析起到很大作用。

概率加密的必要性。在命题 3.19 的证明中，已经说明了构造方法 3.15 对多次加密并不安全。证明中用到的该构造方法的唯一特征是，加密一个消息总是产生相同的密文，所以实际上获得的是，任何确定方案对多次加密来说，都是不安全的。这足够重要以至于作为定理来

陈述。

定理 3.20 令$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$为一个加密方案，Enc是密钥和消息的一个确定函数，则Π不具备窃听者存在情况下不可区分多次加密。

这意味着根据定义 3.18 来构造一个安全的加密方案，至少必须保证当相同的消息被多次加密时，每次都产生不同的密文。初看上去这好像是一个不可能完成的任务，因为解密必须总能恢复消息。但是，稍后将会解释如何实现它。

使用流密码的多次加密中的一个普遍的错误。不幸的是，对密码学构造方法的不正确的实现是很频繁的。一个普遍的错误就是，使用一个流密码（在构造方法 3.15 中的初级形式）来加密多个明文。例如，这个错误出现在 Microsoft Word 以及 Excel 中执行加密的情况下；参见参考文献[147]。在实际中，这样的错误可能是毁灭性的。需要强调这不仅仅是“理论上的人工制品”，因为加密相同的消息两次产生相同消息。即使相同的消息从来不被加密两次，各种攻击也是可能的，正如 2.2 节中提到的。

使用流密码的安全多次加密。在实际中，一般有两种流密码/伪随机发生器的使用方式进行安全地加密多个明文（见如图 3.3）。

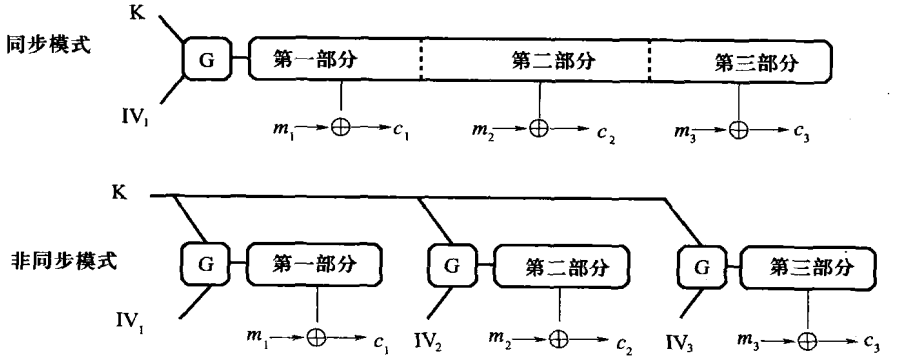


图 3.3 同步模式和非同步模式的比较

(1) 同步模式：在该模式中，通信双方使用流密码输出流的不同部分来加密每个消息。该模式是“同步的”，因为双方需要知道输出流的哪个部分已经被使用，以便于防止重用，因为（已经说明了）重用会不安全。

该模式对于通信双方在单个会话中是非常有用的。在该情形中，一方使用流的第一部分来发送它的第一个消息。另一方获得密文，解密，然后使用流的接下来的一部分来加密它的回复。因为流的每一部分仅仅使用一次，将所有双方发送的消息连接起来，看作单个长明文是可能的。该方案的安全性依据定理 3.16（变长消息情形）。

该模式并不适合所有的应用，因为在加密时，要求双方维护状态（要分清接下来使用流中哪一部分）。由于使用流的不同部分，尽管该方法是确定的，但其安全性和定理 3.20 并不冲突（因为它并不满足定义 3.7 中的语法要求）。

(2) 非同步模式：在这种模式下，加密是双方相互独立地执行的，并且双方不需要维护状态。但是，为了实现安全性，必须重点强调伪随机发生器的概念。这里一个伪随机发生器有两个输入：一个种子s和一个长度为n的初始向量IV。简言之，要求即使当IV已知时（但

是s保密），$G(s, IV)$依然是伪随机的。而且，对应于两个随机选择的初始向量IV_1和IV_2，流$G(s, IV_1)$以及$G(s, IV_2)$应该保持随机性，即使是它们连同各自的IV一起被观察到。

给出一个上面的发生器，加密能够被定义为

$$\mathsf{Enc}_k(m) := \langle IV, G(k, IV) \oplus m \rangle$$

其中$IV \leftarrow \{0,1\}^n$是随机选择的。（为了简化，这里主要关注对定长消息的加密。）对每次加密，IV是全新选出的（即独立且均匀随机地选择），并且因此每个流是伪随机的，即使前面的流已知。注意到IV是作为密文的一部分来发送的，用于接收方的解密；即给定(IV, c)，接收方能够计算出$m := c \oplus G(k, IV)$。

实际中的很多流密码被假定具备上面非正式地简要描述的“广义伪随机性”的特点，并且能够被应用在非同步模式。但是注意，一个标准的伪随机发生器可能不具备这个特点，该假设是一个强假设。事实上，这样一个发生器“几乎是”一个伪随机函数；参见 3.6.1 节以及练习 3.20。

3.5 选择明文攻击（CPA）的安全性

到目前为止，已经考虑了一个相对弱的敌手，仅仅只能在通信方之间被动地窃听。（其实该定义中$\mathsf{PrivK}^{\mathsf{eav}}$允许敌手选择要加密的明文。尽管如此，除了这个能力外，敌手是完全被动的。）在本节中，正式引入一个更强大的攻击类型，叫做选择明文攻击（CPA）。在 CPA 攻击下的安全定义和定义 3.8 中的相同，除了敌手的能力被加强了。

选择明文攻击背后的基本想法是，敌手$\mathcal{A}$被允许适应性地选择多消息来请求加密。这一点被形式化为：允许$\mathcal{A}$和“加密预言机”自由交互，加密预言机被视为一个“黑盒子”，将使用密钥k（$\mathcal{A}$不知道k）加密$\mathcal{A}$选择的消息。根据计算机科学中的标准表示方法，这里用$\mathcal{A}^{\mathcal{O}(\cdot)}$来表示$\mathcal{A}$访问预言机$\mathcal{O}$的计算过程，因此在这种情况下，用$\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$来表示$\mathcal{A}$访问使用密钥$k$的加密预言机$\mathcal{O}$的计算。当$\mathcal{A}$询问预言机并提供一个明文消息$m$作为输入时，预言机返回一个密文$c \leftarrow \mathsf{Enc}_k(m)$作为回复。当Enc是随机的，则预言机每次回答一个询问时，都会保证新的随机性。

安全性的定义要求：即使$\mathcal{A}$具备访问加密预言机能力，$\mathcal{A}$也不能区分两个任意消息的加密。这里先提出定义，然后将讨论该定义模拟真实世界的什么攻击。

首先定义一个对任意对称密钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，任意敌手$\mathcal{A}$，以及任意安全参数$n$的实验。

CPA 不可区分实验$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$：

(1) 一个密钥k是通过运行$\mathsf{Gen}(1^n)$来生成的。

(2) 输入1^n给敌手$\mathcal{A}$，敌手$\mathcal{A}$可以访问预言机$\mathsf{Enc}_k(\cdot)$，输出一对长度相等的消息m_0, m_1。

(3) 选择一个随机比特$b \leftarrow \{0,1\}$，计算出一个密文$c \leftarrow \mathsf{Enc}_k(m_b)$，交给$\mathcal{A}$。$c$叫做挑战密文。

(4) 敌手$\mathcal{A}$继续访问预言机$\mathsf{Enc}_k(\cdot)$，输出一个比特b'。

(5) 如果$b = b'$，该实验的输出被定义为1，否则定义为0。（若$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$，则认为

$\mathcal{A}$成功。）

定义 3.21 一个对称密钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，如果对所有的概率多项式敌手$\mathcal{A}$，存在一个可忽略函数$\mathsf{negl}$，使得

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

则是选择明文攻击（CPA）条件下的不可区分加密，这里概率来源于$\mathcal{A}$使用到的随机性，以及在实验中使用到的随机性。

任何方案如果是选择明文攻击条件下的不可区分加密，则也是窃听者存在的情况下不可区分加密。这是因为$\mathsf{PrivK}^{\mathsf{eav}}$是$\mathsf{PrivK}^{\mathsf{cpa}}$的一种特殊情况，即敌手没有使用预言机。

初看上去，可能会觉得定义 3.21 是不可能达到的，特别是考虑到敌手输出(m_0, m_1)，然后接收到挑战密文$c \leftarrow \mathsf{Enc}_k(m_b)$。因为敌手$\mathcal{A}$能够访问预言机$\mathsf{Enc}_k(\cdot)$，它能够要求该预言机加密消息$m_0$以及$m_1$，并且因此获得$c_0 \leftarrow \mathsf{Enc}_k(m_0)$和$c_1 \leftarrow \mathsf{Enc}_k(m_1)$。敌手$\mathcal{A}$能够比较$c_0$和$c_1$得到$c$，即如果$c = c_0$，则$\mathcal{A}$知道$b = 0$；如果$c = c_1$，则知道$b = 1$。问题是为什么该策略不能使$\mathcal{A}$以 1 的概率确定$b$？

答案就是，如果加密方案是确定的（因为这意味着每次同一个消息加密，会获得相同的密文），这样的攻击确实有效。因此，和多次加密的安全一样，任何确定加密方案都不能抵御选择明文攻击。换句话说，任何 CPA 安全的加密方案必须是概率性的。必须将随机性作为加密过程中的一部分，来确保相同消息的加密可能是不同的[9]。

现实世界中的选择明文攻击。定义 3.21 至少和早期的定义 3.8 一样强，所以从这个更新的定义着手，不会损失安全性。但是，通常使用一个太强的定义会付出一些代价，因为可能会导致使用效率更低的方案（即使存在满足“现实世界的应用”的更有效的方案）。因此，应该询问自己：选择明文攻击是否代表了一个真正值得关注的现实中的敌对威胁。

事实上在很多场景中，选择明文攻击（以某种形式）是一种现实的威胁。首先通过查看一些二战时期的军事历史可说明这一点。在 1942 年 5 月，美国海军密码专家已经发现，日本正计划对位于中太平洋的中途岛发动攻击。他们通过截获了一份通信消息才掌握到这个信息，该消息包含了一份密文片段“AF”，并且他们相信这对应着明文“中途岛”。他们尝试让华盛顿的指挥者相信的确如此，但不幸的是，尝试无效；通常认为中途岛不可能成为攻击目标。海军密码专家策划了接下来的计划。他们命令在中途岛的美军军队发送一个明文消息，说他们的淡水供给不足。日本截获了该消息，立刻报告给其上级：“AF”淡水不足。海军密码专家于是确信“AF”的确就是中途岛，美军迅速派遣了三架运输机飞抵该位置。结果就是中途岛被解救了，日军受到重创。据说，这一场战争就是美国和日本在太平洋战场的转折点。（参见文献[91]和[146]查阅更多信息。）

这里，海军密码专家完成了一次经典的选择明文攻击。他们能够要求日本（尽管是一种迂回方式）来加密“中途岛”这个词，来推测另外一份之前截获的密文的信息。如果日军的

---

⑨ 正如我们已经看到的，如果加密过程在连续加密之间保持状态（正如在流密码的同步模式中那样），则随机性（由抛硬币决定）可能不是必须的。按照定义 3.7，我们一般只考虑无状态的方案（更合意）。

加密方案能够抵御选择明文攻击，美军密码专家的策略将无法成功（并且历史可能变得非常不同）。需要强调，日本并没有打算成为美军的“加密预言机”，也没有分析 CPA 安全的必要性，他们不太可能意识到 CPA 安全是必要的[10]。

注意，不要认为选择明文攻击仅仅只是投机取巧的结果。在很多情况中，一个敌手能够影响诚实方加密的内容（对敌手来说，完全掌握加密的内容是不多见的）。考虑下面的例子：当前很多服务器以一种安全的方式（即使用加密）进行相互通信。但是，这些服务器发送给对方的消息是基于收到的内部和外部请求，这些请求可能是恶意用户选择的。这些敌手能够影响服务器加密的明文消息，且有的时候影响程度非常大。这样的系统必须通过使用一个可以抵御选择明文攻击的加密方案来保护，因此强烈建议总是使用这种类型的加密方案。

多次加密的 CPA 安全。把定义 3.21 扩展到多次加密的情况是很简单的，与定义 3.8 扩展到定义 3.18 相同。即定义一个实验，除了$\mathcal{A}$输出一对明文向量以外，该实验和$\mathsf{PrivK}^{\mathsf{cpa}}$完全相同。要求不存在多项式时间的敌手$\mathcal{A}$，这种敌手在该实验中成功的概率大于$1/2$的部分是不可忽略的。

重要的是，单次加密的 CPA 安全即意味着多次加密的 CPA 安全。（这和窃听敌手的情况形成对比；参见命题 3.19。）这里陈述该声称，而不给出证明，在 10.2.2 节公钥情形中会给出证明。

命题 3.22 任何在选择明文攻击条件下的不可区分加密的对称密钥加密方案，也是在选择明文攻击条件下的不可区分“多次”加密方案。

这是 CPA 安全的重要技术优势。它足够用来证明一个方案对单次加密是 CPA 安全的，然后我们“免费”获得了它对于多次加密也是 CPA 安全的。

固定长度和任意长度消息的比较。 从 CPA 安全的定义着手的另外一个优势是，它允许在处理定长的加密方案时，不会过多失去一般性。特别地，给定任何 CPA 安全的定长加密方案 $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，容易构造一个 CPA 安全的任意长度的加密方案 $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$。为了简化，假设$\Pi$加密1比特长的消息（当$\Pi$加密某个任意长度$\ell(n)$的消息，这里所述可以自然而然地进行相应扩展）。令$\mathsf{Gen}$和$\mathsf{Gen}'$是相同的。对任意消息$m$（有某个任意长度$\ell$），以下面的方式定义$\mathsf{Enc}'_k$：

$$\mathsf{Enc}'_k(m) = \mathsf{Enc}_k(m_1), \cdots, \mathsf{Enc}_k(m_\ell)$$

其中$m = m_1 \cdots m_\ell$，对所有i，$m_i \in \{0,1\}$。解密照常进行。于是，当且仅当Π是 CPA 安全时，Π'是 CPA 安全，证明可从命题 3.22 得出。

尽管如此，实际中可能存在比上面这种改变定长加密方案的方式更有效的方法来加密任意长度的消息。在 3.6.4 节中来介绍其他加密任意长度消息的方法。

3.6 CPA 安全的加密方案创建

本节将构造能够抵御选择明文攻击的加密方案，从引入伪随机函数这个重要概念开始。

---

⑩ 问问自己是否预先考虑到这样的攻击。

3.6.1 伪随机函数

如前所述，伪随机发生器能够被用来获得在窃听者存在情况下的安全性。伪随机性的概念在获得抵御选择明文攻击的安全性中，有重要的作用。但是，现在不考虑伪随机字符串，而是考虑伪随机函数。我们将特别关注将n比特字符串映射到n比特字符串的伪随机函数。正如早前对伪随机性的讨论，任何固定函数$f:\{0,1\}^n \to \{0,1\}^n$是伪随机的说法是没有意义的（同样的道理，一个固定的函数是随机的说法几乎没有意义）。因此，必须在技术上说明这是函数分布的伪随机性。一个简单的方式是考虑带密钥的函数，定义如下。

一个带密钥的函数F是双输入函数，$F:\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$，其中第一个输入叫做密钥，用$k$来表示，第二个输入就叫输入。一般地，密钥$k$被选择之后就固定了，于是关注单输入函数$F_k:\{0,1\}^* \to \{0,1\}^*$，用$F_k(x) \stackrel{\text{def}}{=} F(k,x)$来定义。为了简化，通常将假设$F$是保留长度的，意味着$F$中的密钥，输入以及输出的长度都是相等的。也就是说，假设函数F仅当密钥k和输入x有相同长度时才被定义，即$|F_k(x)| = |x| = |k|$。所以，通过固定密钥$k \in \{0,1\}^n$，可获得一个将n比特字符串映射到n比特字符串的函数$F_k(\cdot)$。如果存在一个确定的多项式时间算法能够在给定输入k和x的情况下，计算出$F(k,x)$，则称F是有效的（efficient）。这里将只关注有效的函数F。

假定选择一个随机密钥$k \leftarrow \{0,1\}^n$，并考虑产生的单输入函数F_k，一个带密钥的函数F导致一种函数的分布。从直觉上来说，如果函数F_k（对随机选择的密钥k而言），无法和一个从拥有同样定义域和值域的所有函数的集合中均匀随机选取的函数相区分；也就是说，如果没有多项式时间的敌手能够区分（从某种意义上来说，将马上非常严格的定义）是和F_k（随机选择的密钥k）交互，还是和f交互（这里f是从所有将n比特字符串映射到n比特字符串的函数中随机选择的），则称F是伪随机的。

因为随机选择一个函数的概念同随机选择一个字符串的概念比较起来，显得比较陌生，所以值得在这个概念上花更多的时间。从数学的观点来讲，考虑一个将n比特字符串映射到n比特字符串所有函数的集合Func_n。该集合是有限的（稍后将看到），因此，随机选择一个将n比特字符串映射到n比特字符串的函数，正好对应着均匀随机地从集合中选择一个元素。集合Func_n的规模有多大？通过对其每个定义域中的点给出值，一个函数f能够被完全确定下来。事实上，能够将任何函数（在有限定义域中）看作一个大的查找表，在该表中标签为x的那一行存储了$f(x)$。对于$f \in \mathsf{Func}_n$，f的查找表中有2^n行（每一行对应着定义域$\{0,1\}^n$中的一个点），并且每一行包含一个n比特字符串（因为f的值域是$\{0,1\}^n$）。任何这样的表能够正好用$n \cdot 2^n$比特来表示。此外，Func_n中的函数都是和这种形式的查找表一一对应的，也就是意味着它们和所有长度为$n \cdot 2^n$的字符串是一一对应的。既然长度为$n \cdot 2^n$的字符串有$2^{n \cdot 2^n}$个，这就是Func_n的大小。

将一个函数看成一个查找表提供了另外一种有用的方式来思考均匀随机地选择一个函数$f \in \mathsf{Func}_n$，这正好等价于在f的查找表中均匀随机地选择每一行。也就是说，$f(x)$和$f(y)$的值（$x \neq y$）是完全独立且均匀分布的。

回到关于伪随机函数的讨论，还记得目的是希望构造一个带密钥的函数F，使得能够区分F_k（$k \leftarrow \{0,1\}^n$是均匀随机选择的）和f（$f \leftarrow \mathsf{Func}_n$是均匀随机选择的）。前者是从（最多）$2^n$个不同的函数的分布中选择出来的，但是后者是从$\mathsf{Func}_n$中所有$2^{n \cdot 2^n}$个函数的分布中选择出来的。尽管如此，这些函数的“行为”对任意多项式时间区分器来说，必须看起来都是

相同的。

第一种试图将伪随机函数的概念进行形式化将会使用和定义 3.14 一样的方式进行。也就是说，可以要求每个多项式时间的区分器D，接收到一个伪随机函数F_k的描述，输出1的概率“几乎”和它接收到一个随机函数f的描述，输出 1 的概率相同。但是，该定义是不合理的，因为一个随机函数的描述具有指数长度（即查找表的长度，为$n \cdot 2^n$），然而D被限制在多项式内运行。所以，D甚至将没有足够的时间来检查它的整个输入。

因此，实际的定义允许D访问函数（F_k或者f）的预言机。对任何点x，D被允许问询预言机，并且对预言机返回的点x的函数值作出响应。在选择明文攻击的定义中，提供敌手访问加密预言机的能力，把该预言机当作黑盒子。但这里该预言机计算一个确定函数，因此，当对相同的输入询问两次时，总是返回相同的值。现在已经准备好给出正式的定义。（尽管该定义要求F是长度保留的，这仅仅是为了使假设简化，而不是必要的。）

定义 3.23 令$F:\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$是有效的、长度保留的、带密钥的函数。如果对所有多项式时间区分器D，存在一个可忽略函数negl，满足：

$$\left|\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]-\Pr\left[D^{f(\cdot)}(1^n)=1\right]\right| \leqslant \mathsf{negl}(n),$$

则称F是一个伪随机函数，其中$k \leftarrow \{0,1\}^n$是均匀随机选择的，并且f是从将n比特字符串映射到n比特字符串的函数集合中均匀随机选择出来的。

注意D是和预言机自由交互的，因此它能够适应性地问询，并根据前一个接收到的输出来选择下一个输入。但是，因为D在多项式时间内运行，所以它只能执行多项式时间数量的问询。同时注意，伪随机函数必须拥有随机函数固有的高效率的特点。例如，即使x和x'仅仅只有单个比特不同，$F_k(x)$以及$F_k(x')$的输出必须（概率计算主要来源于k的随机选择）看上去完全不相关。这预示伪随机函数对构造安全加密方案是有用的。

该定义中的关键点是，区分器D不知道密钥k。如果已知k，那么要求F_k是随机的将没有意义，因为给定k，区分F_k的预言机和f的预言机是很容易的。区分器向预言机问询点0^n来获得答案y，然后将其和结果$y'=F_k(0^n)$（通过使用已知的密码k计算出来）相比较。一个F_k的预言机将总是返回$y=y'$，然而随机函数的预言机返回$y=y'$的概率仅为2^{-n}。在实际中，这意味着一旦k被泄露，则F_k的伪随机性就不再满足。具体而言，考虑伪随机函数F，给定F_k的预言机供访问（这里k是随机的），找到一个输入x满足$F_k(x)=0^n$是困难的（因为对真正的随机函数f而言，找到这样的输入是很困难的）。但是如果k是已知的，则找到这样的一个输入可能很简单（现实中常常如此）。

伪随机函数的存在性。和伪随机发生器一样，很重要的一点是要弄清楚伪随机函数是否存在，如果存在，应该在怎样的假设下存在。实际中，一个非常有效的原语叫做分组密码，被使用且被广泛地认为可作为伪随机函数。在 3.6.3 节中将进一步讨论，并且第 5 章给出更深入地对分组密码的讨论。从理论角度看，已经知道，当且仅当伪随机发生器存在，则伪随机函数存在。因此，伪随机函数能够基于任何困难问题来构造，这些困难问题就是可以构造伪随机发生器的问题，在第 6 章中将详细讨论。基于困难问题的伪随机函数的存在代表了现代密码学的一个令人吃惊的真正迷人的贡献。

在密码学中使用伪随机函数。在很多不同的密码学构造方案中，伪随机函数成为一个非

常有用的构造模块。下面使用它来获得 CPA 安全的加密，并在第 4 章中，用它来构造消息鉴别码。伪随机函数用途很大的一个原因是，使用了伪随机函数的构造方案在分析时会很清晰和精巧。也就是说，给定一个基于伪随机函数的方案，一般的分析方法是：首先证明在真正随机函数的假设下，该方案是安全的。这一步骤依赖于一个概率分析，与计算的限制或困难性无关。接下来，原始方案的安全性是通过证明来获得，这个证明是：当伪随机函数被使用时，如果一个敌手能够攻破该方案，则它必须能够将伪随机函数与真正的随机函数区分开来。

3.6.2 基于伪随机函数的 CPA 安全加密

本节关注构造一个 CPA 安全的定长加密方案。根据 3.5 节末的陈述，这意味着存在一个 CPA 安全的任意长度消息的加密方案。3.6.4 节将考虑更多有效的方式来处理任意长度的消息。

从伪随机函数构造一个安全的加密方案的简单想法是定义$\mathsf{Enc}_k(m) = F_k(m)$，期待这“没有泄露任何关于$m$的信息”（因为随机函数$f$的$f(m)$只是一个简单的随机$n$比特字符串）。但是，该加密方法是确定性的，所以不是 CPA 安全的。具体而言，指定$c = \mathsf{Enc}_k(m_b)$，请求得到加密$\mathsf{Enc}_k(m_0)$和$\mathsf{Enc}_k(m_1)$是可能的；因为$\mathsf{Enc}_k(\cdot) = F_k(\cdot)$是一个确定函数，其中有一个肯定会等于$c$，因而就泄露了$b$的值。

实际中的构造是概率性的。特别是对随机值（而不是明文消息）应用伪随机函数，然后将这个结果和明文消息异或来进行加密。（参见构造方法 3.24，如图 3.4 所示。）这可以再一次被看作伪随机“填充”和明文消息进行异或的一个实例，主要的不同是每次使用一个独立的伪随机填充。只要伪随机函数每次的输入都不同，这一点就能办到。当然，一个随机值r可能重复，并多次使用，这个将在证明中明确地被考虑到。

构造方法 3.24

令F是伪随机函数，定义一个消息长度为n的对称密钥加密方案如下：

- Gen：输入1^n，均匀随机地选择$k \leftarrow \{0,1\}^n$，并将其作为密钥输出。
- Enc：输入一个密钥$k \in \{0,1\}^n$，以及一个消息$m \in \{0,1\}^n$，均匀随机地选择$r \leftarrow \{0,1\}^n$，并且输出密文
$$c := \langle r, F_k(r) \oplus m \rangle。$$
- Dec：输入一个密钥$k \in \{0,1\}^n$，以及一个密文$c = \langle r, s \rangle$，输出明文消息
$$m := F_k(r) \oplus s。$$

从任意伪随机函数构造 CPA 安全的加密方案

从直观上看安全性是满足的，因为$F_k(r)$对一个观察密文$\langle r, s \rangle$的敌手来说，看上去完全随机。因此，加密方案和“一次一密”是非常相似的，只要值r没有在之前的某次加密中使用（特别是当回答敌手的问询时，只要它没有被加密预言机使用）。此外，该“坏事件”（也就是值r的一次重复）发生的概率是可忽略的。

定理 3.25 如果F是伪随机函数，则构造方法 3.24 为消息长度为n的定长对称密钥加密方案，在选择明文攻击下，该构造方法具备不可区分加密。

证明 这里的证明遵照着一般处理伪随机函数的范例：首先是在理想的世界中分析方案

的安全性，一个真正的随机函数f取代了F_k，然后说明该方案在这种情况下是安全的。接下来声称，当F_k被使用时，如果该方案是不安全的，则意味着将F_k从一个真正随机函数中区分出来是可能的。

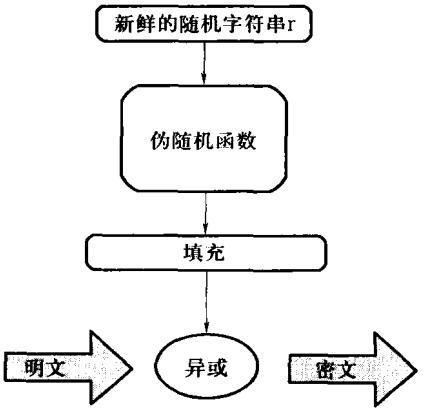


图 3.4　用伪随机函数的加密

令$\tilde{\Pi}=\left(\widetilde{\mathsf{Gen}},\widetilde{\mathsf{Enc}},\widetilde{\mathsf{Dec}}\right)$为加密方案，除了真正的随机函数$f$取代了$F_k$之外，完全和构造方法 3.24 中的$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$是相同的。也就是：$\widetilde{\mathsf{Gen}}\left(1^n\right)$选择了一个随机函数$f\leftarrow\mathsf{Func}_n$，除了$f$取代了$F_k$外，$\widetilde{\mathsf{Enc}}$加密和Enc是一样的。（这不是一个合法的加密方案，因为它不是有效的。尽管如此，这是一个为了证明而构造的脑力实验，为达到证明的目的而定义。）每个（甚至是无效的）敌手$\mathcal{A}$最多向它的加密预言机问询$q(n)$次，有

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)=1\right]\leqslant\frac{1}{2}+\frac{q(n)}{2^n} \tag{3.4}$$

为了说明这一点，回想每次一个消息m被加密（要么通过加密预言机，要么是在实验$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)$中的挑战密文被计算出来），选择一个随机$r\leftarrow\{0,1\}^n$，并且密文被设置为等于$\langle r,f(r)\oplus m\rangle$。令$r_c$表示生成挑战密文$c=\langle r_c,f(r_c)\oplus m_b\rangle$时用到的随机字符串。有两种分情况：

(1) 值r_c被加密预言机使用来回答至少一个$\mathcal{A}$的问询：在这种情况下，$\mathcal{A}$可能轻易地判断出哪一个消息被加密。这是因为无论何时加密预言机返回一个密文$\langle r.s\rangle$来回应加密消息m的请求，敌手掌握了值$f(r)$（因为$f(r)=s\oplus m$）。

因为$\mathcal{A}$最多向预言机询问$q(n)$次，并且每次预言机回答它的询问都是使用一个均匀随机选择的r，所以该事件发生的概率最多为$q(n)/2^n$（能够通过一致限来理解这一点）。

(2) 在加密预言机回答$\mathcal{A}$的询问时，值r_c从来没有被用到：在这种情况下，$\mathcal{A}$在它和加密预言机的交互过程中，没有掌握到任何关于$f(r_c)$值的信息（因为f是一个真正随机的函数）。这就意味着，据$\mathcal{A}$所知，$f(r_c)$和m_b作异或操作后的值是完全随机的，因此在这种情况下，$\mathcal{A}$输出$b'=b$的概率正好是$1/2$（和“一次一密”中的情况一样）。

令Repeat表示值r_c被加密预言机使用来回答至少一个$\mathcal{A}$的询问的事件。上面已经说明了Repeat发生的概率最多为$q(n)/2^n$，并且如果Repeat没有发生，则$\mathcal{A}$在$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)$中成功的概率刚好是$1/2$。因此，我们有：

$$
\begin{aligned}
&\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\right]\\
&\quad=\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\wedge\mathsf{Repeat}\right]+\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\wedge\overline{\mathsf{Repeat}}\right]\\
&\quad\leqslant\Pr\left[\mathsf{Repeat}\right]+\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\mid\overline{\mathsf{Repeat}}\right]\\
&\quad\leqslant\frac{q(n)}{2^n}+\frac{1}{2},
\end{aligned}
$$

得到等式(3.4)。

现在，固定某个 PPT 敌手$\mathcal{A}$，定义函数ε

$$
\varepsilon(n)\stackrel{\mathsf{def}}{=}\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right]-\frac{1}{2} \tag{3.5}
$$

由$\mathcal{A}$发出的预言机问询次数的上界值是它的运行时间。因为$\mathcal{A}$在多项式时间内运行，所以它发出的预言机问询的次数的上界是某个多项式$q(\cdot)$。关于这个$\mathcal{A}$，等式(3.4)也满足；因此

$$
\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\right]\leqslant\frac{1}{2}+\frac{q(n)}{2^n}
$$

并且通过ε的定义，

$$
\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right]=\frac{1}{2}+\varepsilon(n)
$$

如果ε不可忽略，则两者之间的差别也是不可忽略的。直觉上表明，这样一个“差距”（如果存在的话）将可用来区分伪随机函数和一个真正的随机函数。

下面正式用规约法来证明。使用$\mathcal{A}$来构造一个伪随机函数F的区分器D。区分器D可以使用某函数的预言机，其目标是判断出该函数是“伪随机的”（即F_k，随机选择的$k\leftarrow\{0,1\}^n$），还是“随机的”（f，随机选择的$f\leftarrow\mathsf{Func}_n$）。为了达到目的，$D$使用下面的方式，来模拟$\mathcal{A}$的 CPA 不可区分性实验，并且观察$\mathcal{A}$成功与否。如果$\mathcal{A}$成功了，则$D$猜测它的预言机一定是一个伪随机函数，然而，如果$\mathcal{A}$没有成功，则$D$猜测它的预言机一定是一个随机函数，具体如下：

区分器D：

D被指定了输入1^n以及可访问预言机$\mathcal{O}:\{0,1\}^n\to\{0,1\}^n$。

(1) 运行$\mathcal{A}(1^n)$。无论何时$\mathcal{A}$向它的加密预言机发出关于消息m的问询，用下面的方法来回答该询问：

① 均匀随机地选择$r\leftarrow\{0,1\}^n$。

② （向预言机）问询$\mathcal{O}(r)$，获得相应s'。

③ 返回密文$\langle r,s'\oplus m\rangle$给$\mathcal{A}$。

(2) 当$\mathcal{A}$输出消息$m_0,m_1\in\{0,1\}^n$时，随机选择一个比特$b\leftarrow\{0,1\}$，然后：

① 均匀随机地选择$r\leftarrow\{0,1\}^n$。

② （向预言机）问询$\mathcal{O}(r)$，获得相应s'。

③ 返回挑战密文$\langle r,s'\oplus m_b\rangle$给$\mathcal{A}$。

(3) 正如之前那样，继续回答任何$\mathcal{A}$对加密预言机的问询。最终，$\mathcal{A}$输出一个比特b'。如果$b'=b$，则输出1，否则输出0。

关键点如下：

(1) 如果D的预言机是一个伪随机函数，那么当$\mathcal{A}$被D作为一个子程序运行时的视图分布，与在实验$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$中$\mathcal{A}$的视图分布是相同的。因为密钥$k$是随机选择的，并且每次加密的完成都是通过选择一个随机的r，计算$s'=F_k(r)$，然后设定密文等于$\langle r, s'\oplus m\rangle$，与构造方法 3.24 一样。因此，

$$\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]=\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right]$$

其中，上面的$k\leftarrow\{0,1\}^n$是均匀随机选择的。

(2) 如果D的预言机是随机函数，那么当$\mathcal{A}$被D作为一个子程序运行时的视图分布，和在实验$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)$中的视图分布是相同的。这一点正好可以通过上面看出来，唯一的区别就是随机函数f代替了F_k。因此，

$$\Pr\left[D^{f(\cdot)}(1^n)=1\right]=\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)=1\right]$$

其中，上面的$f\leftarrow\mathsf{Func}_n$是均匀随机选择的。

联合等式(3.4)和(3.5)和上面的等式，有

$$\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]-\Pr\left[D^{f(\cdot)}(1^n)=1\right]\geqslant\varepsilon(n)-\frac{q(n)}{2^n}。$$

通过假设F是伪随机函数，那么$\varepsilon(n)-q(n)/2^n$必须是可忽略的。因为q是多项式，这反过来暗示着ε是可忽略的，所以Π是 CPA 安全的，证毕。 ■

正如在 3.5 节中讨论的，任何 CPA 安全的定长加密方案会自动产生一个任意消息长度的 CPA 安全加密方案。将这个方法应用到刚构造出来的定长方案上，有一个任意长度的消息$m=m_1,\cdots,m_\ell$，其中每一个m_i都是n比特的分块，安全的加密可通过计算

$$\langle r_1, F_k(r_1)\oplus m_1, r_2, F_k(r_2)\oplus m_2,\cdots, r_\ell, F_k(r_\ell)\oplus m_\ell\rangle$$

得到。（该方案能够通过删减来处理长度不是正好为n的倍数的消息，这里忽略细节。）于是有下面的推论。

推论 3.26 如果F是伪随机函数，则上面粗略描述的方案，是一个在选择明文攻击下，具备不可区分加密的任意长度消息的对称密钥加密方案。

构造方法 3.24 的有效性。在构造方法 3.24 中的 CPA 安全的加密方案以及在推论 3.26 中对任意长度消息的扩展，都存在一个缺陷，就是密文的长度（至少）是明文长度的两倍。这是因为每个大小为n的分组是使用一个n比特的字符串来加密的，且该字符串必须作为密文的一部分。3.6.4 节将给出密文长度是如何明显减小的方法。

3.6.3 伪随机置换和分组加密

令$F:\{0,1\}^*\times\{0,1\}^*\rightarrow\{0,1\}^*$是有效的、长度保留的、带密钥的函数。如果对每个$k$，函数$F_k(\cdot)$是单射（因为$F$是长度保留的，因而$F_k$实际上是双射）。把$F$称作一个带密钥的置换。如果给定$k$和$x$，存在一个多项式时间算法能够计算$F_k(x)$，并且如果给定$k$和$x$，也存在一个多项式时间算法能够计算$F_k^{-1}(x)$，则这个带密钥的置换是有效的。

这里约定密钥、输入和输出的长度全部是相等的，在实际中，这对构造方案来说并不是必要的。相反，输入和输出长度，一般叫做分块长度，是相同的（因为是置换，所以必须如此），但是密钥长度能够小于或者大于分块长度，取决于该构造方法。假设它们都是相等的，仅仅是为了简化概念。

使用和定义 3.23 类似的方法来定义：有效的带密钥置换F是伪随机置换。唯一的变化就是，现在要求无法区分F_k（k是随机选择的）是随机选择的置换，而不是随机选择的函数。实际上，这仅仅只是一个有美感的结论，因为使用多项式数量的问询，随机置换和（长度保留）随机函数是不可区分的。从直觉上说，这是因为一个随机函数f看起来和一个随机置换是相同的，除非能够找到一对不同的值x和y，使得$f(x)=f(y)$（因为在这种情况下，函数不可能是置换）。但是，用多项式数量的问询找到这样的点x和y的概率是可忽略的。下面命题的证明留作练习。

命题 3.27 如果F是一个伪随机置换，则也是一个伪随机函数。

如果F是一个有效的伪随机置换，那么基于F的密码学方案可能需要诚实方除了要计算置换F_k本身以外，还要计算逆F_k^{-1}。这潜在地引入了新的安全问题，这种问题没有被F是伪随机的这一事实所覆盖。在这样的情况下，可能需要更强的需求，即F_k和一个随机置换是不可区分的，即使区分器可以使用逆置换预言机。如果F有这个属性，称其为强伪随机置换。正式陈述如下：

定义 3.28 令$F:\{0,1\}^*\times\{0,1\}^*\to\{0,1\}^*$是有效的带密钥的置换。如果对于所有概率多项式时间区分器D，存在一个可忽略函数negl，使得

$$|\Pr\left[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n)=1\right]-\Pr\left[D^{f(\cdot),f^{-1}(\cdot)}(1^n)=1\right]|\leqslant\mathsf{negl}(n),$$

则称F是强伪随机置换，其中$k\leftarrow\{0,1\}^n$是随机均匀选择的，f是从n比特字符串置换的集合中均匀随机选择的。

当然，任何强伪随机置换同时也是一个伪随机置换。

前面已经注意到，一个流密码能够被建模成为一个伪随机发生器。类比在实际中强伪随机置换的情况，则是分组密码。不幸的是，在文献中并不是经常提到，一个分组密码实际上被认为是一个强伪随机置换。明确地用这种方法来建模分组密码，使得依赖于分组密码的许多实际构造方案的形式化分析成为可能。这些构造包括（本章学习的）加密方案、消息鉴别码（在第 4 章学习）、认证协议以及更多。此外，当证明一个构造的安全性时，声明分组密码是被建模成为伪随机置换还是强伪随机置换是很重要的。尽管现在使用的大部分分组密码是设计用来满足后者，更强的需求，但基于前者的方案能够被证明是安全的，弱的假设是更可取的（因为对分组密码的要求更容易满足）。

相对于流密码，分组密码自身是不安全的加密方案。但是，构造分组密码能够被用于构造安全的加密方案。例如，在构造方法 3.24 中使用分组密码产生了一个 CPA 安全的对称密钥加密方案。与此相反，只是计算$c:=F_k(m)$的加密方案不是 CPA 安全的，这里F_k是强伪随机置换。分组密码作为构造模块的加密方案和使用分组密码本身进行加密的加密方案间的区别

非常重要，也常常被忽视。

强伪随机置换在设计和分析有效密码学方案时非常有用，在本书的余下部分，将只使用伪随机置换（没必要强伪随机置换）。

3.6.4 加密操作模式

操作模式是使用分组密码（即伪随机置换）来加密任意长度消息的基本方法。在推论3.26中，已经看到加密模式的一个例子，尽管从密文的长度角度来说，不是很有效率。在这一节中，将看到许多具有较低密文扩展（定义为密文长度和消息长度的差别）的加密操作模式。

在继续之前，注意到通过在尾部添加一个1，然后使用足够多的0填充，可将任意长度的消息填充为总长度为分组大小的倍数。因此，假设明文消息的长度正好是分组大小的倍数。贯穿这一节，认为伪随机置换/分组密码F的分组长度为n，并且考虑由ℓ个分组（每个分组的长度为n）组成的消息的加密。这里将说明四种加密操作模式，并且讨论其安全性。

模式 1：电子密码本（ECB）模式。 这是可能的最简单的加密操作模式。给定一个明文消息$m = m_1, m_2, \cdots, m_\ell$，密文是通过分别“加密”每个分组获得的，其中这里的“加密”意味着直接对明文分组应用伪随机置换。也就是说，$c = \langle F_k(m_1), F_k(m_2), \cdots, F_k(m_\ell)\rangle$。如图3.5所示，其中$\ell = 3$。解密过程执行的方式很显然就是使用有效可计算的$F_k^{-1}$。

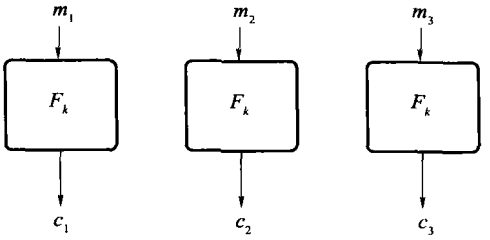


图3.5 电子密码本（ECB）模式

这里，加密过程是确定性的，因此，该操作模式不可能是CPA安全的（参见在定义3.21后的讨论）。更糟糕的是，在窃听者存在的情况下，ECB加密模式并不具备不可区分性。这是因为如果相同的分组在明文中重复了两次，那么在密文中就能够发觉有重复的分组。因此，非常容易区分：是两个相同分组组成的明文的加密，还是两个不同分组组成的明文的加密。注意这不仅仅是一个“理论问题”，通过观察这种方式产生的密文，能够掌握到很多信息。因此，ECB模式应该永远不被使用。（这里提到它只是因为它的历史重要性。）

模式 2：密码分组链接（CBC）模式。 在这种模式中，首先要选择一个长度为n的随机初始向量（IV）。然后，通过对当前明文和前一个密文的异或值做伪随机置换，来获得下一个密文分组。也就是说，令$c_0 := IV$，然后$i = 1$到ℓ，令$c_i := F_k(c_{i-1} \oplus m_i)$。最终的密文是$\langle c_0, c_1, \cdots, c_\ell\rangle$，如图3.6所示。注意$IV$不加密并作为密文的一部分发送，这对解密来说是非常关键的。

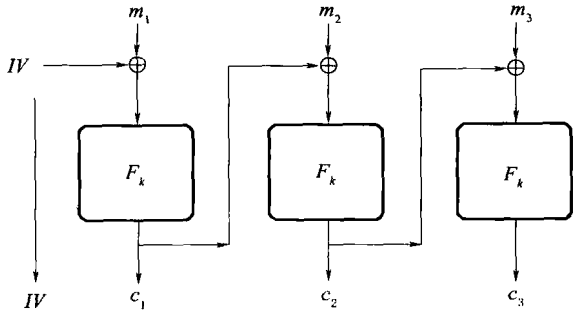


图 3.6　加密块链（CBC）模式

重要的是 CBC 模式的加密是概率的。已证明，如果F为伪随机置换，那么 CBC 加密模式是 CPA 安全的。该模式的主要缺陷是加密必须依次执行，因为要加密明文分组m_i，我们需要密文分组c_{i-1}。因此，如果并行处理是可行的，则 CBC 模式不是最有效率的选择。

有人可能会想在每次加密时使用不同的IV（而不是随机的IV）就可以了；比如，首先使用$IV = 1$，然后每次对IV增加 1。练习 3.16 要求说明这个 CBC 加密操作模式的变体是不安全的。

模式 3：输出反馈（OFB）模式。 在这里说明的第三种模式被叫做 OFB。本质上说，该模式是使用分组密码来生成伪随机流，然后和消息作异或操作。首先，选择一个随机$IV \leftarrow \{0,1\}^n$，然后通过IV用下面的方式生成一个流（独立于明文）：定义$r_0 := IV$，令流的第i个分组r_i为$r_i := F_k(r_{i-1})$。则每个明文分组通过和对应的流分组做异或操作来达到加密的目的；也就是$c_i := m_i \oplus r_i$。（参见图 3.7 的图形描述。）和 CBC 模式一样，为了解密，IV是密文的一部分；相对于 CBC 模式，这里不需要F是可逆的（事实上，F甚至不需要是个置换）。

该模式也是概率的，并且能够证明如果F是伪随机函数，则它是 CPA 安全的加密方案。这里，加密和解密都必须依次执行。从另一方面说，该模式有个优点是大块的计算（也就是伪随机流的计算）能够独立于实际加密消息进行。所以，在预处理之前，准备好一个流是可能的，此后加密明文（一旦已知）是非常快的。

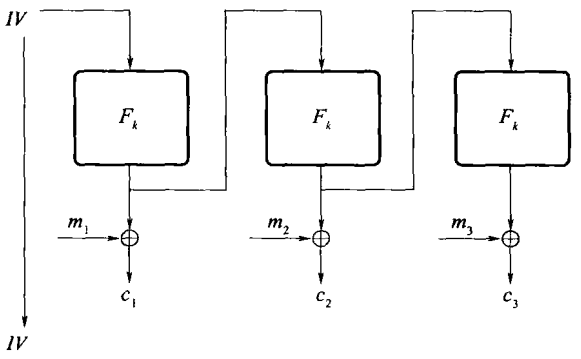


图 3.7　输出反馈（OFB）模式

模式 4：计数器（CTR）模式。有很多不同的 CTR 加密模式的变体，这里描述随机化的计数器模式。和 OFB 一样，计数器模式能够看成是一种用分组密码生成伪随机流的方法。首先，选择一个随机$IV \leftarrow \{0,1\}^n$，该IV经常用ctr表示。通过计算$r_i := F_k(\text{ctr}+i)$（其中ctr和i被看作是整数，加法是模2^n的加法）生成一个流。最后第i个密文分组计算为$c_i := r_i \oplus m_i$，并且IV再次被作为密文的一部分发送，如图 3.8 所示。需要强调的是，该解密不要求F是可逆的，甚至不要求是置换。

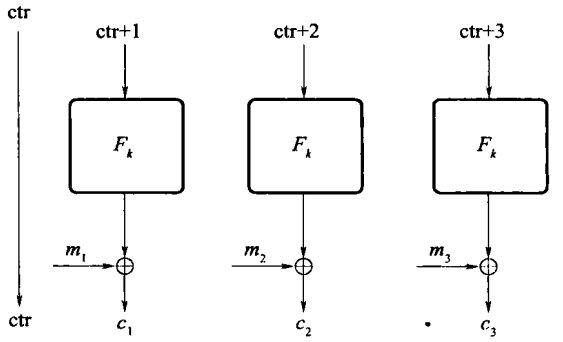


图 3.8　计数器（CTR）模式

计数器模式有很多重要的特点。首先也是最重要的，随机计数模式（即每次消息被加密，当ctr是随机均匀地选择的时候）是 CPA 安全的，这将在下面证明。第二，加密和解密都能完全并行，和 OFB 模式一样，在预处理之前生成独立于消息的伪随机流是可能的。最后，只解密第i个密文分组而不解密任何其他的密文是可能的，该特点被叫做随机访问。上述特点让计数模式看上去非常有吸引力。

定理 3.29　如果F是一个伪随机函数，那么在选择明文攻击下，随机计数模式（描述如上）具备不可区分加密。

证明　正如定义 3.25 的证明一样，对现在的定理，首先说明当使用一个真正随机的函数时，随机计数模式是 CPA 安全的。然后证明，用一个伪随机函数替换随机函数不会令该方案不安全。

当挑战密文在实验$\mathsf{PrivK}^{\mathsf{cpa}}$中被加密时，令ctr*表示使用的初始值ctr。从直觉上看，当在随机计数模式中使用一个随机函数f时，只要使用ctr^*+i来加密挑战密文的每个分组c_i，且ctr^*+i从未被加密预言机用于回答以前的问询，就可以达到安全性。这是因为：如果ctr^*+i从未被加密预言机用于回答以前的加密问询，那么$f(\text{ctr}^*+i)$就将是一个完全随机的值，那么用这个值和明文的一个分组作异或操作，和通过“一次一密”相比，具有相同的效果。当使用随机函数时，随机计数模式是 CPA 安全的证明转化为界定ctr^*+i以前被使用过的概率。

$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$表示随机计数模式的加密方案，然后令$\widetilde{\Pi}=\left(\widetilde{\mathsf{Gen}},\widetilde{\mathsf{Enc}},\widetilde{\mathsf{Dec}}\right)$是和$\Pi$相同的加密方案，唯一的区别是$\widetilde{\Pi}$使用的是一个真正的随机函数，而不是一个伪随机置换$F$。也就是说，$\widetilde{\mathsf{Gen}}(1^n)$选择一个随机函数$f \leftarrow \mathsf{Func}_n$，并且除了$f$代替了$F_k$外，$\widetilde{\mathsf{Enc}}$和Enc加密方式是一样的。（当然，$\widetilde{\mathsf{Gen}}$和$\widetilde{\mathsf{Enc}}$都不是有效的算法，但是这对定义含有$\widetilde{\Pi}$的实验这一目的来说没有影响。）现在说明对每个概率多项式时间的敌手$\mathcal{A}$，存在一个可忽略函数negl，使得

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1\right] \leqslant \frac{1}{2}+\mathsf{negl}(n) \tag{3.6}$$

实际上，不需要作出任何关于$\mathcal{A}$的运行时间（或者是计算能力）的假设；仅仅要求$\mathcal{A}$对其加密预言机（每一次问询都是多项式长度的消息）作出多项式数量的问询就足够了，此外，$\mathcal{A}$输出多项式长度的m_0, m_1。

令q是$\mathcal{A}$作出的预言机问询数量的多项式上界，也是任何这样问询的最大长度以及m_0, m_1的最大长度。固定安全参数为某个值n。令ctr^*表示当挑战密文被加密时，使用的初始值ctr，并且令ctr_i表示当加密预言机回答$\mathcal{A}$的第i个预言机询问时，使用的初始值ctr。当挑战密文被加密时，函数f被应用在值$\mathsf{ctr}^*+1,\cdots,\mathsf{ctr}^*+\ell^*$上，其中$\ell^* \leqslant q(n)$是$m_0, m_1$的长度。当第$i$个预言机问询被回答时，函数$f$被应用在值$\mathsf{ctr}^*+1,\cdots \mathsf{ctr}^*+\ell_i$上，其中$\ell_i \leqslant q(n)$是（以分组表示）被请求加密的消息的长度。这里有两种情况被考虑：

情况 1。 不存在任意的$i,j,j' \geqslant 1$（$j \leqslant \ell_i$，且$j' \leqslant \ell^*$）使得$\mathsf{ctr}_i+j=\mathsf{ctr}^*+j'$。在这种情况下，当加密挑战密文时，使用的值$f(\mathsf{ctr}^*+1),\cdots,f(\mathsf{ctr}^*+\ell^*)$是彼此独立的，并且是均匀分布的，因为当加密任何敌手的预言机问询时，f没有在这些输入上使用过。这意味着：挑战密文的计算是通过异或随机比特流和消息m_b来计算的，所以在这种情况下，$\mathcal{A}$输出$b'=b$的概率正好是$1/2$（同“一次一密”中的情况一样）。

情况 2。 存在任意的$i,j,j' \geqslant 1$（$j \leqslant \ell_i$，并且$j' \leqslant \ell^*$）使得$\mathsf{ctr}_i+j=\mathsf{ctr}^*+j'$。也就是，用来加密第$i$个加密预言机问询的第$j$分组的值和用来加密挑战密文的第$j'$分组的值是相同的。在这种情况下，$\mathcal{A}$很容易确定哪一个消息被加密且给出挑战密文（因为敌手能够从它的第i个预言机问询的答案中推断出$f(\mathsf{ctr}_i+j)=f(\mathsf{ctr}^*+j')$的值）。

现在，分析上述情况发生的概率。如果ℓ^*和每一个ℓ_i尽可能地大，概率能够达到最大，所以假设对所有i，都有$\ell^*=\ell_i=q(n)$。令$\mathsf{Overlap}_i$表示序列$\mathsf{ctr}_i+1\cdots\mathsf{ctr}_i+q(n)$重叠于序列$\mathsf{ctr}^*+1\cdots\mathsf{ctr}^*+q(n)$的事件，令$\mathsf{Overlap}$表示对某个$i$，$\mathsf{Overlap}_i$发生的事件。因为最多有$q(n)$个预言机问询，由一致限，得

$$\Pr[\mathsf{Overlap}] \leqslant \sum_{i=1}^{q(n)} \Pr[\mathsf{Overlap}_i] \tag{3.7}$$

固定ctr^*，事件$\mathsf{Overlap}_i$发生是当ctr_i满足

$$\mathsf{ctr}^*+1-q(n) \leqslant \mathsf{ctr}_i \leqslant \mathsf{ctr}^*+q(n)-1$$

因为ctr_i有$2q(n)-1$个值使得$\mathsf{Overlap}_i$发生，并且ctr_i是随机均匀地从$\{0,1\}^n$中选择的，于是

$$\Pr[\mathsf{Overlap}_i]=\frac{2q(n)-1}{2^n}$$

将其和等式(3.7)结合，得出$\Pr[\mathsf{Overlap}] \leqslant 2q(n)^2/2^n$。

于是能够轻易地界定$\mathcal{A}$成功的概率：

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)=1\right]=$$

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)=1 \wedge \mathsf{Overlap}\right]+\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)=1 \wedge \overline{\mathsf{Overlap}}\right] \leqslant \Pr[\mathsf{Overlap}]+\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(n)\right.$$

$$\left.=1|\overline{\mathsf{Overlap}}\right] \leqslant \frac{2q(n)^2}{2^n}+\frac{1}{2}$$

因为q是多项式，$2q(n)^2/2^n$是可忽略的，式(3.6)得证。也就是说，该（虚构的）方案$\tilde{\Pi}$是 CPA 安全的。

证明的下一步骤是说明Π（即感兴趣的方案）是 CPA 安全的；也就是说，对任何概率多项式时间$\mathcal{A}$，存在一个可忽略函数negl'，使得

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)=1\right]\leqslant\frac{1}{2}+\mathsf{negl}'(n)。$$

从直觉来说，这是因为只要是一个多项式时间敌手，用在Π中使用的伪随机函数F_k替换在$\tilde{\Pi}$中使用的随机函数f应该“没有影响”。当然，该直觉应该被严格地证明。该证明和在定理 3.25 的证明中的相应步骤是非常相似的，所以将证明留作练习。 ■

分组长度和安全性。 所有上面的模式（除了 ECB 是不安全的）都使用了随机IV。IV有随机化加密过程的效果，并且确保（概率很高）分组密码总是使用之前没有用过的输入。这一点很重要，正如定理 3.25 和定理 3.29 的证明所述，如果分组密码的一个输入被多次使用，那么安全性就将破坏。（比如，在计数模式的情况下，相同的伪随机字符串将和两个不同的明文分组进行异或。）有趣的是，这说明了在评价安全性中起重要作用的不仅仅是分组密码的密钥长度，还包括分组长度。例如，假设使用一个 64 比特分组长度的分组密码，定理 3.29 的证明中说明，在随机计数模式中，即使使用这一分组长度的完全随机函数（也就是说即使分组密码是“完美的”），敌手在选择明文攻击中，向加密预言机作出q次问询，并且每个问询都是q个分组长，则仍然能够以约为$\frac{1}{2}+\frac{q^2}{2^{63}}$的概率成功。尽管这是渐进可忽略的（当分组长度随着安全参数n增长时），但是从任何实践意义来说，当$q\approx 2^{30}$时（对这个特别的分组长度）安全性不再满足。应用中有人可能转换到分组长度更大的分组密码（2^{30}仅仅是 1 吉字节，没有考虑现在的存储需求）。

其他的操作模式。 在最近几年，引入了许多不同的操作模式，每一个都在某种情形中提供了独特的优势。尽管如此，普遍来说，CBC、OFB 以及 CTR 模式满足大部分需要 CPA 安全的应用。

加密模式和消息篡改。 在很多密码学文献中，操作模式也会在有效防止敌手篡改密文方面相互比较。这里，并不包含这样的比较，因为消息完整性或者消息鉴别的问题应该和加密分开处理。从定义的意义上来说，上面提到的模式都没有能够实现消息完整性。下一章将更深入地讨论。

流密码和分组密码。 正如这里的 OFB 和计数器模式，让分组密码工作在“流密码模式”是可能的（即生成一个伪随机比特流然后将其和明文异或）。更进一步，一个分组密码能够被用来生成多个（独立的）伪随机流，相比之下（一般的）一个流密码被限制只能生成单个这样的流。这提出了一个问题：分组密码和流密码，哪一个更合适？流密码的唯一优势就是它们相对来说更有效，尽管这个优势可能仅仅只是二分之一，除非正在使用资源受限的设备，比如 PDA 或者移动电话。[11]从另外一个方面看，流密码看起来远不如分组密码好理解（在实践中）。有很多非常出色的分组密码都是有效的，并且被认为是高度安全的（将在第 5 章学习两种）。相比之下，流密码看起来更经常被攻破，对它们的信心也比较低。更可能的是，流密码以某种方式被乱用，如相同的伪随机流被使用两次。因此，通常推荐使用分组密码，除非某些原因使得分组密码不可能被使用。

[11] 特别地，根据文献[42]中的评估：用比特/秒来测量，在一个典型的家用 PC 上，流密码 RC4 仅仅是分组密码 AES 的两倍快。

3.7 CCA 安全性

直到现在，已经定义了抵抗两种类型攻击的安全性：一种是被动攻击，只是窃听；另一种是主动攻击，执行选择明文攻击。第三种类型的攻击，叫做选择密文攻击，比前面两种更为强大。在选择密文攻击中，提供敌手的不仅仅是能够加密其选择的消息的能力（正如在选择明文攻击里面的那样），还提供了解密其选择的密文的能力（稍后讨论一个例外）。正式地说，除了加密预言机之外，也让敌手使用解密预言机。正式的定义以及深入的讨论推迟到后面。

考虑到下面对任意对称密钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，敌手$\mathcal{A}$，安全参数值$n$的实验。

CCA 不可区分实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$：

(1) 通过运行$\mathsf{Gen}(1^n)$生成一个密钥k。

(2) 敌手$\mathcal{A}$被指定输入1^n，可使用预言机$\mathsf{Enc}_k(\cdot)$和$\mathsf{Dec}_k(\cdot)$。它输出一对长度相等的消息m_0, m_1。

(3) 随机选择比特$b \leftarrow \{0,1\}$，则计算密文$c \leftarrow \mathsf{Enc}_k(m_b)$并将其提供给$\mathcal{A}$。我们把$c$叫做挑战密文。

(4) 敌手$\mathcal{A}$继续使用预言机$\mathsf{Enc}_k(\cdot)$和$\mathsf{Dec}_k(\cdot)$，但是不允许用挑战密文本身来问询$\mathsf{Dec}_k(\cdot)$。最终，$\mathcal{A}$输出一个比特b'。

(5) 如果$b' = b$，实验的输出为1，否则输出0。

定义 3.30 一个对称密钥加密方案Π满足如下条件：对所有的概率多项式时间敌手$\mathcal{A}$，存在一个可忽略的函数negl，使得

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1\right] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

则称其为选择密文攻击条件下不可区分加密（或者是 CCA 安全的）其中概率来源于实验中的所有随机因素。

在上面的实验中，敌手使用解密预言机是不被限制的，除了有一个限制：敌手不能要求解密挑战密文本身。该限制是很必要的，否则很明显，任何加密方案都不可能满足定义3.30。

在这一点上，读者可能想知道选择密文攻击是否能够真实地模拟任何真实世界中的攻击。正如在选择明文攻击中的情况那样，不期望诚实方来解密敌手选择的任意密文。无论如何，存在一种场景，敌手可能能够影响解密的内容，然后掌握一些关于结果的部分信息。举例如下：

(1) 在中途岛（参见 3.5 节）的情况中，可以想象美国密码分析专家可能也试图向日本发送已加密的消息，然后监视他们的动作。这样的行为（比如，类似军队的调动等）能够提供重要的关于潜在明文的信息。

(2) 想象一个用户和他们的银行通信，其中所有的通信都是经过加密的。如果该通信没有被认证，则敌手能够代表用户发送特定密文；银行将解密这些密文，敌手可能能够从结果中

掌握到某些信息。例如，如果一个密文对应着一个不合文法的明文（比如，一个莫名其妙的消息，或者一个格式不正确的简单消息），该敌手能够通过银行的反应（比如接下来的通信模式）推断出一些信息。

(3) 加密经常在高级别的协议中使用到；比如，一个加密方案可能被作为认证协议的一部分来使用，其中一方发送一份密文给对方，对方解密然后返回结果。在这种情况下，一个诚实方可能正好扮演了一个解密预言机的角色，所以该方案必须是 CCA 安全的。

前面研究方案的不安全性。 到目前为止，已给出的加密方案中，都不是 CCA 安全的。将在构造方法 3.24 中给出说明，其中加密方式为$\mathsf{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$。该方案不是 CCA 安全的事实能够很容易地说明如下。一个在 CCA 不可区分实验中的敌手$\mathcal{A}$能够选择$m_0 = 0^n$以及$m_1 = 1^n$。那么，接收到密文$c = \langle r, s \rangle$，敌手$\mathcal{A}$能够翻动s的第一个比特，然后要求解密这个改动过的密文c'。因为$c' \neq c$，该问询是允许的，解密预言机的回答要么是10^{n-1}（在这种情况下，显然$b = 0$）或者是01^{n-1}（在这种情况下，显然$b = 1$）。这个例子说明了为什么 CCA 安全性更严格。特别地，任何允许密文以“逻辑方式”被操纵的加密方案都不是 CCA 安全的。因此，CCA 安全实际上意味着一个非常重要的特点叫做“不可延展性”。粗略地说，一个不可延展的加密方案具备这个特点：如果敌手试图去修改一个指定的密文，结果要么是一个非法密文，要么是一个和原明文没有任何关系的明文的密文。目前给出的加密模式中，没有一种是 CCA 安全的，这一点的说明留作练习。

构造一个 CCA 安全加密方案。 4.8 节将说明如何构造一个 CCA 安全的加密方案。后面将说明其构造，这是因为构造 CCA 安全的加密方案需要使用第 4 章详述的工具。

参考文献和扩展阅读材料介绍

密码学中的现代计算方法最早出现在 Goldwasser 和 Micali 的论文[69]中。该论文引入了语义安全的概念，并且说明了语义安全能够在公钥加密的情形中实现（参见第 9 章和第 10 章）。该论文也提议了不可区分性的概念（即定义 3.8），可以看到不可区分性意味着语义安全。逆命题见文献[104]。

抵御选择明文攻击的安全性的正式定义在 Luby 的[96]和 Bellare 等的[9]中提到。选择密文攻击（在公钥加密的情形中）的正式定义首先被 Naor 和 Yung[107]，Rackoff 和 Simon[121]提出，在文献[50]和[9]中也被考虑到。如果要了解其他对称密钥加密安全的概念，可以参见文献[86]。

Blum 和 Micali 的文献[23]引入了伪随机发生器的概念，并且在基于特定数论假设基础上，证明了其存在性。在同一工作中，Blum 和 Micali 也指出了在构造方法 3.15 中，伪随机发生器和对称密钥加密之间的联系。由 Blum 和 Micali 给出的伪随机发生器的定义和本书中（定义 3.14）给出的定义不同，后者起源于 Yao 的文献[149]，Yao 说明了两种形式化的表述是等价的。Yao 也说明了基于一般假设的伪随机发生器的构造方法，我们将在第 6 章探讨这个结论。

伪随机函数在 Goldreich 等的文献[67]中被定义和构造，并且他们在接下来的工作[66]中说明了伪随机函数在加密上的应用。Luby 和 Rackoff 在文献[97]中研究了伪随机置换和强伪随机置换。当然，他们从这些工作开始对分组密码进行理论研究，但分组密码已经使用了很多年。

在文献[108]中，ECB、CBC 以及 OFB 操作模式（也包含 CFB 模式，这里没有包含其他模式）连同 DES 分组密码[108]一起被标准化。CBC 和 CTR 加密模式在文献[9]中已经证明是 CPA 安全的。其他近期的一些加密模式，参见 http://csrc.nist.gov/CryptoToolkit。一个不错的但稍有些过时的对实际中使用的流密码的概论能够在文献[99]的第 6 章中找到。RC4 流密码在文献[125]中被讨论，对其最近的攻击以及后果的讨论可在文献[56]中找到。

练　习

3.1 证明命题 3.6。

3.2 现在已知的，找到一个n比特素因数的最佳算法的运行时间为$2^{c\cdot n^{\frac{1}{3}}(\log n)^{\frac{2}{3}}}$。假设 4 兆赫的计算机以及$c=1$（给定表达式的单位为时钟周期），估算数字的大小，使得该数字在接下来的100年内不能被因数分解。

3.3 如果Π能够加密任意长度的消息，并且在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中敌手不被限制地输出等长消息，证明定义 3.8 不能被满足。
提示：当Π被用来加密单个比特时，令$q(n)$为密文长度的多项式上界。然后考虑一个输出$m_0\in\{0,1\}$的敌手以及一个随机$m_1\in\{0,1\}^{q(n)+2}$。

3.4 假设$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$，对$k\in\{0,1\}^n$，算法$\mathsf{Enc}_k$仅仅是为长度最多为$\ell(n)$（某个多项式$\ell$）的消息定义的。构造一个即使当敌手在实验$\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中，没有被限制要求输出等长的消息时仍然满足定义 3.8 的方案。

3.5 证明定义 3.8 和定义 3.9 是等价的。

3.6 令G为一个伪随机发生器，其中$|G(s)|>2\cdot|s|$。
① 定义$G'(s)\stackrel{\mathsf{def}}{=}G\left(s0^{|s|}\right)$。$G'$必然是伪随机发生器吗？
② 定义$G'(s)\stackrel{\mathsf{def}}{=}G\left(s_1\cdots s_{n/2}\right)$，其中$s=s_1\cdots s_n$。$G'$必然是伪随机发生器吗？

3.7 假设一个伪随机函数存在，证明在窃听者存在的情况下，存在一个加密方案，该加密方案具备不可区分的多次加密（根据定义 3.18），但不是 CPA 安全的（根据定义 3.21）。
提示：该方案不必“自然而然”。将需要用到一个事实：在选择明文攻击中，敌手能够适应性地选择对加密预言机的问询。

3.8 无条件证明一个有效伪随机函数$F:\{0,1\}^*\times\{0,1\}^*\to\{0,1\}^*$的存在性，其中输入长度是安全参数的对数（密钥的长度是安全参数的多项式）。
提示：实现一个随机函数，输入为对数输入长度。

3.9 从任意伪随机函数中，表述一个变输出长度的伪随机发生器的构造方法。证明这个构造方法满足定义 3.17。

3.10 令G为伪随机发生器，定义$G'(s)$为G的输出，并且被缩短到n比特（其中$|s|=n$）。证明函数$F_k(x)=G'(k)\oplus x$不是伪随机的。

3.11 证明命题 3.27（也就是说，证明任意伪随机置换也是一个伪随机函数）。
提示：说明在多项式时间内，一个随机置换不能从一个随机函数中区分开来（使用附录 A.4 的结论）。

3.12 通过自然而然地改写定义 3.12，定义抵抗选择明文攻击的完善保密的概念。说明该定义是不能被实现的。

3.13 假设F是伪随机置换。说明存在一个函数F'，该函数是伪随机置换但非强伪随机置换。
提示：构造一个F'，使得$F'(k)=0^{|k|}$。

3.14 令F为伪随机置换，定义一个定长加密方案(Gen, Enc, Dec)如下：输入$m\in\{0,1\}^{n/2}$，密钥$k\in\{0,1\}^n$，算法Enc选择一个长度为$n/2$随机字符串$r\leftarrow\{0,1\}^{n/2}$，并且计算$c:=F_k(r\|m)$。
说明如何解密，并证明该方案对长度为$n/2$的消息是CPA安全的。（如果正寻找一个真正的挑战，证明在F是强伪随机置换的条件下，该方案是CCA安全的。）和构造方法3.24相比，该方案的优势和缺陷分别是什么？

3.15 令F为伪随机函数，G是一个伪随机发生器，扩展因子$\ell(n)=n+1$。对下面每个加密方案，在窃听者存在的情况下，说明该方案是否具备不可区分加密，以及说明该方案是否是CPA安全的。在每一种情况下，共享密码都是随机的$k\in\{0,1\}^n$。
① 加密$m\in\{0,1\}^{2n+2}$，解析m为$m_1\|m_2$，其中$|m_1|=|m_2|$，然后发送$\langle G(k)\oplus m_1, G(k)\oplus m_2\rangle$。
② 加密$m\in\{0,1\}^{n+1}$，选择一个随机$r\leftarrow\{0,1\}^n$，然后发送$\langle r, G(r)\oplus m\rangle$。
③ 加密$m\in\{0,1\}^n$，发送$m\oplus F_k(0^n)$。
④ 加密$m\in\{0,1\}^{2n}$，解析m为$m_1\|m_2$，其中$|m_1|=|m_2|$，然后随机选择$r\leftarrow\{0,1\}^n$，最后发送$\langle r, m_1\oplus F_k(r), m_2\oplus F_k(r+1)\rangle$。

3.16 考虑一个CBC加密模式的变体：每次当一个消息被加密时，发送方只是简单地对IV增加1（而不是每次随机选择IV）。说明该方案是非CPA安全的。

3.17 陈述已知的所有不同加密模式的解密准则。哪些模式的解密可以并行运算？

3.18 完成定理3.29的证明。

3.19 令F为伪随机函数，满足：$k\in\{0,1\}^n$，函数F_k为n比特输入到n比特输出的映射。（该章已假设$\ell_{in}(n)=\ell_{out}(n)=n$。）
① 考虑使用该形式的一个F来实现计数模式。对于ℓ_{in}, ℓ_{out}中的哪个函数来说，得出来的加密方式是CPA安全的？
② 考虑使用上面的F来实现计数模式，但是只能处理固长$\ell(n)$（总是$\ell_{out}(n)$的整数倍）的消息。对于$\ell_{in}, \ell_{out}, \ell$中的哪个函数来说，得出来的加密方式是CPA安全的？在窃听者存在的情况下，对于$\ell_{in}, \ell_{out}, \ell$中的哪个函数来说，该方案具备不可区分加密？

3.20 函数$g:\{0,1\}^n\leftarrow\{0,1\}^n$，令$g^{\$}(\cdot)$为预言机：输入$1^n$，随机均匀地选择$r\leftarrow\{0,1\}^n$，并且返回$(r, g(r))$。假设一个带密钥的函数$F$是一个弱伪随机函数，如果对所有的PPT算法$D$，存在一个可忽略函数negl，使得：

$$|\Pr\left[D^{F_k^{\$}(\cdot)}(1^n)=1\right]-\Pr\left[D^{f^{\$}(\cdot)}(1^n)=1\right]|\leqslant \mathsf{negl}(n)$$

其中$k\leftarrow\{0,1\}^n$，$f\leftarrow\mathsf{Func}_n$是随机均匀地选择的。
① 证明：如果F是伪随机的，那么它是弱伪随机。
② 令F'为伪随机函数，然后定义

$$F_k(x)=\begin{cases}F'_k(x) & \text{若}x\text{是偶数}\\ F'_k(x+1) & \text{若}x\text{是奇数}\end{cases}$$

证明F是弱伪随机，但不是伪随机。

③ 使用弱伪随机函数F的计数模式加密是 CPA 安全的吗？在窃听者存在的情况下，它是否必然具备不可区分加密？证明给出的答案。

④ 基于弱伪随机函数，构造一个 CPA 安全的加密方案。

提示：在本章中存在这样一个构造。

3.21 令$\Pi_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$和$\Pi_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$为两个加密方案，已知至少有一个是 CPA 安全的。问题是不知道哪个是 CPA 安全的，哪个不是。说明如何构造一个加密方案Π使得：只要Π_1和Π_2中至少有一个是 CPA 安全的，则Π一定是 CPA 安全的。试着对答案给出一个完整的证明。

提示：从原始明文中，生成两份明文消息，这样掌握了两者之一，都不会泄露明文的内容，但是都掌握了两者，就能够产生出原始的明文。

3.22 说明 CBC、OFB 以及计数模式的加密不是 CCA 安全的加密方案（无论F是什么）。

第 4 章　消息鉴别码和抗碰撞散列函数

4.1　安全通信与消息完整性

加密的最基本目标之一是使通信各方在开放信道进行通信的安全得以保障，但什么是“安全通信”？第 3 章讨论了可以在开放信道实现秘密通信。也就是说，讲解了如何通过加密来阻止窃听者（甚至可能是更加主动的敌手）获知关于消息内容的任何东西，这些消息是经过未保护通信信道发送的。然而，并不是所有的安全问题都与敌手能否获知关于发送内容的信息有关。在很多案例里面，保证信息的完整性（消息鉴别）同样重要甚至更重要，即通信各方都能确保收到的消息的确是来自另一方。

例如，一供应商收到一家大型超市发来的要求订一万箱苏打的邮件，在收到这个邮件之时，供应商须考虑下面两个问题：

(1) 这个订单是否可信？也就是说，超市是否真的下了这个订单？还是竞争对手伪造超市邮件地址来行骗呢（这非常容易做到）。

(2) 即便能保证这个订单是超市发出的，供应商仍然会询问订单上的内容是否真的来自超市的要求，还是在发送途中被敌对方篡改（例如，敌对路由器）。

这个订单本身不是机密的，因此，私密性问题不会在这里出现。这纯粹是消息完整性问题。

一般情况下，如果没有任何特殊措施来保证，不能信赖通信的完整性。事实上，任何没有保护的在线购物订单、网上银行操作、电子邮件和短信，通常很难确信是来自于所声称的源头。不幸的是，人们通常会轻信，从而在很多情况下，诸如呼叫者 ID 或者邮件的回邮地址通常被认为是“可信的来源”（实际上，这些都比较容易伪造）。这些将给潜在的破坏性敌手攻击打开一扇门。

本章将会展示如何使用密码技术来阻止不可检测的消息破坏，这些消息通过公开的信道传输，以保证消息的完整性。注意，无法阻止敌手对消息的破坏，这只能在物理层面上完成。能够保证的是，任何对消息的修改都会被诚实方检测到。

4.2　加密与消息鉴别的比较

消息的私密性和消息的完整性有着不同的目标，因此实现的技术和工具也是不一样的。不幸的是，私密性和完整性常常令人混淆且不必要地纠缠在一起。需要首先明确：加密（通常）不提供完整性的保证，因此，加密应该不被用于消息鉴别。（当然，加密可能与其他技术结合起来达到消息鉴别的目的，见本章末的讨论。）

可能有人会错误地认为，加密就能解决消息鉴别问题（事实上，这是一个常见错误）。这是基于含糊的（且不正确的）推理：既然密文完全隐藏了消息的内容，敌手不可能修改加密的消息并保持消息有意义。尽管在直觉上像很有道理，但这个推理完全是错误的。这里举例

说明这一点，迄今为止，所有的加密方案都不能保证消息的完整性。

使用流密码加密。首先考虑下例：m是一个利用流密码加密的消息，$\mathsf{Enc}_k(m)$计算密文$c := G(k) \oplus m$。其中，G是一个伪随机数生成器。这个例子中的密文很容易被篡改。特别地，在密文c中任何比特位的翻转都会导致在解密后原始信息同样比特位的翻转。因此，给定一段加密m的密文c，可能会产生一个改动的密文c'，满足$m' = \mathsf{Dec}_k(c')$，m'与m相同，除了有一个（或者多个）比特位翻转。这个简单的攻击可能会产生严重的后果。举一个简单的例子，假设一个用户希望从银行账户中转帐一定的金额，金额数量进行了加密，假定金额是用二进制表示的。最低有效比特位的翻转仅能使金额数量改变 1 美元，但是在第 11 位的一个翻换则能使金额的改变超过 1 千美元。有趣的是，敌手在这个例子中并不知道是增加还是减少了初始金额（也就是说，不知道是从 0 翻转成 1 还是 1 翻转成 0）。此外，这种攻击的存在与加密方案的私密性并不矛盾（参见定义 3.8）。事实上，这个攻击同样适用于“一次一密”，也表明即使完善保密加密方案也并不足以保证最基本的消息完整性。

使用分组密码算法加密。上面提到的攻击利用了这样一个事实：对密文中某个比特位的翻转只会导致明文中相应比特位的翻转，而不会引起其他比特位的改变。该攻击也同样适用于 OFB 和计数模式加密方案。它们也是使用一个伪随机流的异或运算来进行加密（即使每次加密消息时都改变伪随机流）。可以看到，即使是使用 CPA 安全的加密也不能阻止消息被篡改。

有人可能会想，攻击 ECB 或 CBC 加密模式可能会更难，因为这些方案的解密需要反转强伪随机函数 F。即使x和x'只有一个比特的差别，也希望$F_k^{-1}(x)$和$F_k^{-1}(x')$完全不相关。（当然，ECB 模式不保证最基本的隐私，但是这不在目前的讨论范围内）。毕竟，密文中一个比特位的改变仍引起在明文中可预见的改变。举例来说，在使用 ECB 加密模式时，密文第i个分组中某位的翻转只影响第i组明文，其他分组并不会变化。虽然对第i组明文的影响可能无法预测，但是对该组的改变（而其他组不变）可能代表着一种有害的攻击。同样地，在使用 CBC 模式时，IV中第j比特位的翻转只影响第一个消息分组的第j位（因为$m_1 := F_k^{-1}(c_1) \oplus IV$，其中$IV'$是被修改的$IV$）；除了第一组，明文的其他分组保持不变（其中第$i$组的明文计算公式如下：$m_i := F_k^{-1}(c_i) \oplus c_{i-1}$，其中$c_i$和$c_{i-1}$分组未修改）。更糟的是，ECB 中分组的顺序可能会被改变（在不改动任何分组的情况下），而 CBC 加密消息的第一个分组可能会被对手使用流密码篡改。这种对 CBC 模式完整性的攻击可能会非常麻烦，因为消息的第一个分组通常包含非常重要的数据包的头信息。

最后，需要指出的是，迄今为止所有的加密方案都有一种属性：每种可能的密文（满足某个长度约束），都会对应某个有效消息。对敌手来说，非常容易伪装来自某个通信方的消息，即发送任意的密文，即使敌手不知道对应的明文消息是什么。在正式定义可鉴别的通信时将会看到这一点，这种类型的攻击应该排除。

4.3 消息鉴别码——定义

如前所述，加密并未解决消息鉴别的问题。取而代之，需要额外的机制使得通信各方了解消息是否被篡改。该机制被称作消息鉴别码（message authentication code）。

消息鉴别码的目的是：阻止敌手进行不被检测到的对消息的修改。在加密情况下，只有在通信各方都掌握了敌手不可知的秘密时才可能实现（否则，没有办法来阻止对手伪装某一

方来发送信息）。这里将继续考虑双方共享密钥情形。

消息鉴别码的语法。在正式定义消息鉴别码（MAC）的安全性之前，首先定义什么是消息鉴别码以及如何使用它。两个用户希望通过认证的方式进行交流，并在双方通信前共享一个密钥k。当一方发送消息m给另一方时，在消息以及共享密钥的基础上计算出 MAC 标记（或简言之标记）t，并将消息m与标记t一起发送给对方。标记是采用标记生成算法产生的，该算法用Mac表示；换句话说，发送方计算出消息m的标记$t \leftarrow \mathsf{Mac}_k(m)$，并将$(m,t)$发送给接收方。当接收方收到$(m,t)$，接收方验证$t$是否是消息$m$的有效标记（通过共享密钥来验证）。这是由一个验证算法Vrfy完成的，该算法的输入是共享的密钥、消息m以及标记t，并指出标记是否有效。正式的定义如下：

定义 4.1 消息鉴别码（MAC）是一个概率多项式时间算法的三元组$(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$，满足：

(1) 密钥产生算法Gen：输入参数1^n，输出密钥k，其中$|k| \geqslant n$。

(2) 标记生成算法Mac：输入密钥k和消息$m \in \{0,1\}^*$，输出标记t，该算法是随机的，记作$t \leftarrow \mathsf{Mac}_k(m)$。

(3) 校验算法Vrfy：输入密钥k、消息m和标记t，输出比特位b。$b=1$意味着有效，$b=0$则无效，不失一般性，假设Vrfy算法是确定的，有$b := \mathsf{Vrfy}_k(m,t)$。

对于每一个n，$\mathsf{Gen}(1^n)$输出的每一个k，以及每一个$m \in \{0,1\}^*$，都有$\mathsf{Vrfy}(m, \mathsf{Mac}_k(m)) = 1$。

如果$(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$满足：对$\mathsf{Gen}(1^n)$输出的每一个$k$，算法$\mathsf{Mac}_k$仅对消息$m \in \{0,1\}^{l(n)}$有定义（且$\mathsf{Vrfy}_k$在任意$m \notin \{0,1\}^{l(n)}$时输出 0），称$(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$是一个固定长度的消息鉴别码（MAC），其消息长度为$l(n)$。

对私钥加密而言，通常$\mathsf{Gen}(1^n)$是均匀随机地选择$k \leftarrow \{0,1\}^n$。

消息鉴别码的安全性。下面定义消息鉴别码的安全性。（与对称密钥加密相比，它只有一个被广泛接受的定义。）该安全定义来自一个直观的思想：没有多项式时间的敌手可以对任何“新”消息产生有效的标记，这些新消息是指没有被某一通信方发出过的（即曾经鉴别过的）。

在任何安全定义中，为了形式化地定义“安全性”，需要对敌手的能力以及什么是“攻破”进行定义。通常认为只存在概率多项式时间的敌手[①]，那么关于敌手能力的实际问题是：如何模拟敌手与各通信方的交互。在消息鉴别码的情形中，观察到诚实通信方的敌手可以看到所有通信双方的消息以及相应的 MAC 标记。敌手可能直接地或者间接地（如果敌手的外部行为影响了发送的消息）影响消息的内容。对于直接影响，有这样一个例子：假设敌手就是某一个通信方的私人助理，可以控制所发送的消息。

为了正式地建模上述可能性，允许敌手请求任意消息的 MAC 标记。也就是说，允许敌手访问一个 MAC 预言机$\mathsf{Mac}_k(\cdot)$；敌手可以提交任何消息m给预言机，得到一个标记$t \leftarrow \mathsf{Mac}_k(m)$。

下面考虑什么是“攻破”一个 MAC 方案。如果敌手能够输出任意消息m及其标记t，满

① 如在第 2 章的“参考文献和扩展阅读材料介绍”中指出的，完善安全消息鉴别的概念（这里对敌手没有计算能力的限制），同样可被考虑。但在完善加密中，完善安全的 MAC 有严重的效率限制，限制了其在实践中的可使用性。

足：①t是消息m的有效标记$\mathsf{Vrfy}(m,t)=1$；②敌手没有在之前获得m的 MAC 标记（从预言机），则认为是方案被“攻破”。敌手满足第一个条件意味着：在现实世界中，如果敌手发送(m,t)给通信的某个诚实方，这一方将错误地认为m是由正当的通信方所发出的（因为$\mathsf{Vrfy}(m,t)=1$），而需要满足第二个条件则是因为：对手常常有可能只是复制某一方之前发出的一条消息和 MAC 标记（当然，这是可以接受的）。这种攻击被称作重放攻击，它并不被认为是攻破了消息鉴别码。这并不意味着重放攻击不是安全问题；它们也是，这会在后面进一步讨论。

满足前面所定义的安全级别的 MAC 称作：在适应性选择消息攻击下的存在性不可伪造。“存在性不可伪造”指对手无法伪造任何消息的标记，而“适应性选择消息攻击”是指敌手可获得想要的任何消息的 MAC 标记，这些标记在攻击中会被适应性地选择。

为了形式化地定义，接下来考虑下面这个消息鉴别码的实验，这里$\Pi=(\mathsf{Gen},\mathsf{Mac},\mathsf{Vrfy})$是消息鉴别码，敌手$\mathcal{A}$，值$n$为安全参数：

消息鉴别码实验 $\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$产生一个随机密钥k；

(2) 敌手被给予输入值1^n，并能访问预言机$\mathsf{Mac}_k(\cdot)$。最后输出一对(m,t)的值，并用$\mathcal{Q}$表示所有$\mathcal{A}$对预言机的询问集合；

(3) 实验的输出结果为 1，当且仅当：$\mathsf{Vrfy}(m,t)=1$，$m\notin\mathcal{Q}$。

如果没有敌手能在上述实验中，以不可忽略的概率成功，则认为 MAC 是安全的。

定义 4.2 一个消息鉴别码$\Pi=(\mathsf{Gen},\mathsf{Mac},\mathsf{Vrfy})$，如果对所有的多项式时间敌手$\mathcal{A}$，存在一个可忽略函数$\mathsf{negl}$，满足

$$\Pr[\mathsf{Mac}-\mathsf{forge}_{\mathcal{A},\Pi}(n)=1]\leqslant \mathsf{negl}(n)$$

则它是适应性选择消息攻击下存在性不可伪造的，或者说消息鉴别码是安全的。

这个定义是否太严格？ 上文的定义是相当严格的，表现在两个方面：第一，敌手被允许查询任何消息的 MAC 标记；其次，只要敌手能够输出一个有效的标记，该标记是之前未经过鉴别的一个消息，就被认为是“攻破”了该方案。可能有人反对，该定义的两项要素不现实，而且太苛刻。在“现实世界”中，通过使用 MAC，通信方只能鉴别“有意义的”消息（敌手可能只有有限的控制能力），并且同样地，只有敌手能对“有意义的”信息伪造一个有效的标记，才会被认为是存在安全漏洞。为什么不调整定义来反映这一意见呢？

关键问题是，消息是否有意义通常依赖于应用程序。某些 MAC 应用只能鉴别英文消息，其他某些应用可能鉴别电子表格，数据库入口，或者是原始数据。协议需要鉴别所有情况。事实上，某些实体认证协议所做的工作正是如此。通过对 MAC 尽可能严格地加以定义，才能确保安全的 MAC 可广泛地用于多种目的，而不用担心 MAC 在应用场景中语义的兼容性。

重放攻击。 注意上述定义以及消息鉴别码，没有提供对于重放攻击的防御，重放攻击即再次发送前面发送过的消息（以及 MAC 标记）给某个诚实通信方。尽管如此，重放攻击仍是一个值得高度关注的问题。看看下面的这个例子：Alice 发送了一条指令将她的银行账户的

1000 美元转到 Bob 的账户中，为了达到目的，Alice 计算出一个 MAC 标记，附加到消息上并发送到银行，这样银行就能知道该消息是可信的。如果 MAC 是安全的，Bob 不可能拦截该消息并将金额改成 10000 美元（因为这将导致伪造对一个未鉴别过的消息的有效的标记）。但是，没办法阻止 Bob 拦截该消息，并将它重复发送给银行 10 次。如果银行接受这些消息，实际的效果就是 10000 美元而不是 1000 美元转移到 Bob 的账户中。

MAC 不能阻止重放攻击，因为 MAC 的定义（定义 4.1）中并没有将任何“状态”包括在验证算法中（因此每当有效的(m,t)对出现在验证算法中，总是输出 1）。然而，对重放攻击的防御（如果这种防御是必要的话），留给了应用层。MAC 进行这种结构化定义的原因是，不想假设任何与应用相关的语义；特别地，是否需要将重放的消息作为“有效的”消息，完全由应用程序决定。

有两种常用的防御重放攻击的技术涉及系列号或者时间戳。第一种方法的基本思想是将每个消息m都分配一个序列号i，则 MAC 通过连接后的消息$i||m$来计算。（实际上，不是这么简单，因为连接必须以一种特定的方式完成，这种方式中$i||m$唯一地确定i和m。但是这里略过这一细节。）假设发送者在每条消息附加上独一无二的序列号，接收者追踪所收到的序列号，这样，任意一次成功的重放消息m都需要在新的连接信息$i'||m$上伪造一个有效的 MAC 标记，这里i'是从未被使用过的。这被 MAC 的安全性所排除。

使用序列号的缺点是接收者需要存储所接收到的所有序列号（如果通信发生在一个专用会话中，发送者很容易每发送一个消息便增加一次序列号，接收者只需要存储接收到的最高序列号）。为了解决这个问题，可采用时间戳的方式。在这种方式下，发送者将消息发送的时间戳附加到消息上（精确到毫秒），而不是将序列号附上去。当接收者收到消息，它需要检查包含的时间戳是否在可接受的时间区间里。这种方法也有缺陷，需要消息的发送方和接收方有同步的钟表，且在这种情况下，重放攻击仍然可能会发生，只要它足够快。

对重放攻击的进一步讨论超出了本书的范围，读者可以查阅其他有关网络安全的书籍。

4.4 构造安全的消息鉴别码

伪随机函数是构建安全的消息鉴别码的常用工具。从直观上来看，如果 MAC 的标记t通过对m运行一个伪随机函数来得到，那么，对未经过鉴别的消息伪造一个标记就要求敌手猜测到随机函数上一个“新的”点的值（也就是说消息）。猜测伪随机函数在新点处的值的概率为2^{-n}（假定函数输出的长度为n）。因此，猜测伪随机函数值的概率可以忽略。

构造方法 4.3

假设F是一个伪随机函数。长度为n的消息的固定长度的 MAC 定义如下：

- Gen：输入参数1^n，均匀随机地选择$k \leftarrow \{0,1\}^n$。
- Mac：输入密钥$k \in \{0,1\}^n$和消息$m \in \{0,1\}^n$，输出标记$t := F_k(m)$。（如果$|m| \neq |k|$则没有输出。）
- Vrfy：输入密钥$k \in \{0,1\}^n$，消息$m \in \{0,1\}^n$，以及标记$t \in \{0,1\}^n$，当且仅当$t \stackrel{?}{=} F_k(m)$，输出 1。（如果$|m| \neq |k|$，输出 0。）

从任意伪随机函数中得到固定长度的 MAC

构造方法 4.3 是为了构造固定长度消息的安全 MAC。虽然它未能达到最终目的，但是仍然有用。本节将讨论如何将处理固定长度消息的 MAC 转换为能处理任意长度消息的 MAC。

定理 4.4 如果F是一个伪随机函数，那么，构造方法 4.3 是长度为n的消息的定长 MAC，这种 MAC 是在适应性选择消息攻击下存在性不可伪造的。

证明 该定理证明背后的直观解释在前文有所描述，接下来直接进入细节。在之前的伪随机函数的使用中，证明的方式是首先在真正的随机函数中分析安全性，接着将真正的随机函数替换为伪随机函数，考察替换后的结果。

假设$\mathcal{A}$是一个概率多项式时间敌手，对其定义如下：

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1]$$

假设一个消息鉴别码$\widetilde{\Pi}=(\widetilde{\mathsf{Gen}},\widetilde{\mathsf{Mac}},\widetilde{\mathsf{Vrfy}})$与构造方法 4.3 中定义的$\Pi=(\mathsf{Gen},\mathsf{Mac},\mathsf{Vrfy})$相同，除了随机函数$f$被伪随机方程$F_k$替代。即，$\widetilde{\mathsf{Gen}}(1^n)$选择一个随机函数$f \leftarrow \mathsf{Func}_n$，$\widetilde{\mathsf{Mac}}$与Mac一样计算出一个 MAC 标记，只是在$\widetilde{\mathsf{Mac}}$中使用了$f$，而不是$F_k$（从技术上讲，这个 MAC 并不合法，因为它没有效率，但可以达到证明的目的）。易知

$$\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1] \leqslant 2^{-n} \tag{4.1}$$

因为对任意$m \notin \mathcal{Q}$，在敌手$\mathcal{A}$看来，都有$t=f(m)$均匀分布在$\{0,1\}^n$上。

现在构造一个多项式时间的区分器D，它能够访问函数的随机预言机。目标是确定这个函数是伪随机的（即是否等于F_k，$k \leftarrow \{0,1\}^n$），还是真正随机的（即等于f，$f \leftarrow \mathsf{Func}_n$）。为了达到此目的，$D$以下面的方式仿真消息鉴别实验，来观察$\mathcal{A}$是否能对“新”消息输出有效的标记。如果可以，$D$就认为随机预言机是伪随机函数，否则，$D$认为它是随机函数。具体如下：

区分器D:

D被给定输入1^n，可以访问预言机$\mathcal{O}:\{0,1\}^n \leftarrow \{0,1\}^n$，按照如下方式工作：

(1) 运行$\mathcal{A}(1^n)$。$\mathcal{A}$询问 MAC 预言机，得到消息m的 MAC，按照下面的方式进行回答：

- 针对m向$\mathcal{O}$问询，并得到回应t；将t返回$\mathcal{A}$。

(2) 当$\mathcal{A}$在执行结束时输出(m,t)时：

① 针对m向$\mathcal{O}$查询，并得到回应$\widehat{t}$。

② 如果$\widehat{t}=t$且$\mathcal{A}$从未针对m查询 MAC 预言机，则输出 1，否则输出 0。

显然，D在多项式时间内运行。

注意，如果D的预言机是一个伪随机函数，$\mathcal{A}$作为D的一个子程序时所见内容（即视图）的分布与$\mathcal{A}$在$\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)$实验中视图的分布是一样的。而且，当$\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1$时输出 1。因此结论为

$$\Pr[D^{F_k(\cdot)}(1^n)=1]=\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1]=\varepsilon(n)$$

其中$k \leftarrow \{0,1\}^n$是均匀随机选取的。如果D的预言机是一个随机函数，那么$\mathcal{A}$作为D的一个子程序时视图的分布，与$\mathcal{A}$在$\mathsf{Mac}-\mathsf{forge}_{\mathcal{A},\tilde{\Pi}}(n)$实验中视图的分布是一样的。当$\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\tilde{\Pi}}(n)=1$时$\mathcal{A}$再次输出1。即

$$\Pr[D^{f(\cdot)}(1^n)=1]=\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\tilde{\Pi}}(n)=1]\leqslant\frac{1}{2^n}$$

其中$f \leftarrow \mathsf{Func}_n$是在上面均匀随机选取的。因此

$$|\Pr[D^{F_k(\cdot)}(1^n)=1]-\Pr[D^{f_k(\cdot)}(1^n)=1]|\geqslant\varepsilon(n)-\frac{1}{2^n}$$

既然F是一个伪随机函数，则存在一个可忽略的函数negl，有$\varepsilon(n)-2^{-n}\leqslant\mathsf{negl}(n)$。于是有$\varepsilon(n)\leqslant\frac{1}{2^n}+\mathsf{negl}(n)$，因此$\varepsilon$是可忽略的。于是构造方法 4.3 是在适应性选择消息攻击下是存在性不可伪造的。

扩展到变长信息。构造方法 4.3 的重要性体现在它显示了基于伪随机函数构造安全的消息鉴别码的通用范例。不幸的是，该构造仅能处理固定长度的消息，且相对较短。在许多（如果不是大部分的）应用中，这项限制是不可接受的[②]。下面说明如何通过定长消息的 MAC 来构建一个通用的（用于可变长度消息的）MAC。这个构造不是非常有效，且不太可能用于实践。事实上，更有效地构造安全的变长 MAC 已经提出，将在 4.5 中进行讨论。这里介绍这个构造方法是因为它的简洁和通用。

令$\Pi'=(\mathsf{Gen}',\mathsf{Mac}',\mathsf{Vrfy}')$是一个长度为$n$的消息的安全定长 MAC（为了简单起见，假设$\mathsf{Gen}'$选择一个随机$n$位密钥）。在介绍这个基于$\Pi'$的变长 MAC 的构造方法之前，明确几个简单的观点。在下文中，基本的观点是m分成块：$m_1,\cdots,m_d$，并通过Π'对分块进行认证。考虑下面的几个建议：

(1) 异或所有的分块，并鉴别得到的结果。即计算标记$t:=\mathsf{Mac}'_k(\bigoplus_i m_i)$，在这个例子中，敌手可以通过改变原始消息来伪造新消息的有效标记，使得所有分块的 XOR 值不会改变。这很容易做到。

(2) 分开鉴别每个分块。也就是说，计算$t_i:=\mathsf{Mac}'_k(m_i)$并输出$\langle t_1,\cdots,t_d\rangle$作为标记。防止敌手发送前面未鉴别过的分块（且不被觉察到）。但是，它不能阻止敌手改变分块的顺序，并计算出有效的标记，例如，消息$m_d,\cdots,m_1$（在定义 4.2 中是不允许的）。

(3) 对每个带有序列号的分块进行鉴别。即计算$t_i=\mathsf{Mac}'_k(i\|m_i)$，并输出$\langle t_1,\cdots,t_d\rangle$作为标记。阻止上面描述的改变顺序攻击。但是，无法阻止对手从末尾丢弃分块（因为$\langle t_1,\cdots,t_{d-1}\rangle$是信息$m_1,\cdots,m_{d-1}$的有效标记）。此外，敌手可以将不同消息的分块进行混合和匹配。即如果敌手分别获得消息$m_1,\cdots,m_d$和$m'_1,\cdots,m'_d$上的标记$\langle t_1,\cdots,t_d\rangle$和$\langle t'_1,\cdots,t'_d\rangle$，可以得到对消息$m_1,m'_2,m_3,m'_4,\cdots$的有效标记$\langle t_1,t'_2,t_3,t'_4,\cdots\rangle$。

上文所描述的所有攻击必须能被最终方案防御。最终方案在每个分块上附加信息。特别地，除了附加序列号，还可以附加随机的“消息标识符”来阻止来自不同消息的分块的组合。最后，每个分块都包含消息的长度，从而消息尾部的分块不可能被丢弃。方案如构造方

② 给定一个可处理变长输入的伪随机函数，构造方法 4.3 产生一个对任意长度消息的安全 MAC。同样地，伪随机函数有更长的，但是固定的定义域，将为更长的消息产生一个安全 MAC。但是，存在实用的伪随机函数（分组密码）仅仅对短的、定长输入有定义。

法 4.5 所示[③]。

构造方法 4.5

假设$\Pi'=(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$是针对长度为$n$的消息的定长 MAC。对 MAC 的定义如下：

- Gen：同Gen'。
- Mac：输入密钥$k\in\{0,1\}^n$和消息$m\in\{0,1\}^*$，消息长度$l<2^{\frac{n}{4}}$，将m分成d个分块$m_1,\cdots, m_d$，每个长度为$n/4$。（必要时用 0 填充最后一个分块。）然后，选择一个随机的标识码$r\leftarrow\{0,1\}^{n/4}$。
  对$i=1,\cdots,d$，计算标记$t_i\leftarrow \mathsf{Mac}'_k(r\|l\|i\|m_i)$，其中，$i,l$被唯一地编码成长度为$n/4$的字符串[④]。最后，输出标记$t:=\langle r,t_1,\cdots,t_d\rangle$。
- Vrfy：输入密钥$k\in\{0,1\}^n$和消息$m\in\{0,1\}^*$，消息的长度$l<2^{\frac{n}{4}}$，标记$t=\langle r,t_1,\cdots,t_{d'}\rangle$，将$m$分成$d$个分块$m_1,\ldots,m_d$，每个分块长度为$n/4$。（必要时用 0 填充最后一个分块）。当且仅当$d'=d$，且对于$1\leqslant i\leqslant d$时，$\mathsf{Vrfy}'_k(r\|l\|i\|m_i,t_i)=1$，输出 1。

从定长 MAC 到变长 MAC

定理 4.6 如果Π'是一个长度为n的消息的安全定长 MAC，那么构造方法 4.5 所得到的 MAC 是在适应性选择消息攻击下存在性不可伪造的。

证明 直观地认为，只要Π'是安全的，敌手就不能引入带有有效标记的新分块。此外，每个分块包含的额外信息阻止了各种各样的攻击（丢弃分块，调换分块顺序等）。需要强调，能够防御已知的攻击与证明其安全性完全不同，因此下面严格证明其安全性（并且说明上面的攻击是“仅仅”可能的攻击）。

假设Π表示构造方法 4.5 中给出的 MAC，$\mathcal{A}$代表概率多项式时间敌手，定义$\varepsilon(\cdot)$如下：

$$\varepsilon(n)\stackrel{\mathsf{def}}{=}\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1] \tag{4.2}$$

令Repeat表示这样一个事件：在$\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)$实验中，同样的消息标识码（$r$）出现在 MAC 预言机返回的两个标记中。如果$(m,t=\langle r,t_1,\cdots\rangle)$代表最后$\mathcal{A}$的输出，$l$代表$m$的长度，令Forge代表这样一个事件：至少$r\|l\|i\|m_i$中的一个分块未被 MAC 预言机鉴别过，但$\mathsf{Vrfy}'_k(r\|l\|i\|m_i,t_i)=1$。也就是说，Forge代表$\mathcal{A}$能对事先未被定长 MAC Π'鉴别过的“消息”输出有效的标记。

于是有

$$\begin{aligned}\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1]&=\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1\wedge\mathsf{Repeat}]\\&+\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1\wedge\overline{\mathsf{Repeat}}\wedge\overline{\mathsf{Forge}}]\\&+\Pr[\mathsf{Mac}\text{-}\mathsf{forge}_{\mathcal{A},\Pi}(n)=1\wedge\overline{\mathsf{Repeat}}\wedge\mathsf{Forge}]。\end{aligned}\tag{4.3}$$

---

③ 该构造方法仅仅对长度$l<2^{n/4}$的消息有定义。这个局限容易解决：改变Mac，当$l\geqslant 2^{n/4}$时输出$\perp$，改变Vrfy，使其当$l\geqslant 2^{n/4}$时输出 1。虽然这看上去“有点不太聪明”，但多项式时间敌手不能输出一个长度为$2^{n/4}$（当n足够大）的消息，所以安全性依然满足。从实践的角度看，不认为诚实方曾发送指数长度的消息，所以原构造方法是足够的。

④ 注意i和l可以编码为$n/4$比特，因为$i,l<2^{n/4}$。

下面说明$\Pr[\text{Mac - forge}_{\mathcal{A},\Pi}(n)=1 \wedge \text{Repeat}] \leqslant \Pr[\text{Repeat}]$是可忽略的，且$\Pr[\text{Mac - forge}_{\mathcal{A},\Pi}(n)=1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{Forge}}]=0$。

声称 4.7 存在一个可忽略函数negl，有$\Pr[\text{Repeat}] \leqslant \text{negl}(n)$。

假设$q(n)$是$\mathcal{A}$所作出的 MAC 随机预言机问询的数量。为了回应第i次随机预言机问询，随机预言机均匀随机选择$r_i \leftarrow \{0,1\}^{n/4}$。先看事件Repeat发生的概率，即$i \neq j$时$r_i = r_j$的概率。应用附录 A.4 中的“生日问题”（引理 A.9），有$\Pr[\text{Repeat}] \leqslant \frac{q(n)^2}{2^{n/4}}$。（标识码是从大小为$|\{0,1\}^{n/4}|=2^{n/4}$的集合中选出的。）既然$\mathcal{A}$只能作出多项式次数的问询，那么这个值是可忽略的，从而证明了该声称。

声称 4.8 $\Pr[\text{Mac - forge}_{\mathcal{A},\Pi}(n)=1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{Forge}}]=0$。

也就是说，如果Mac-forge$_{\mathcal{A},\Pi}(n)=1$且Repeat事件并未发生，就意味着Forge事件发生。下面证明这一点。还记得$\mathcal{A}$最后输出(m,t)，l表示m的长度。将m分成d个分块：$m_1,\cdots,m_d$，每个长度是$n/4$（必要时用 0 填充最后一个分块）。有$t=\langle r,t_1,\cdots,t_d\rangle$；既然Mac - forge$_{\mathcal{A},\Pi}(n)=1$，知道$t$包括$d+1$个部分。有如下几种情况：

(1) 标识码r与 MAC 预言机所使用的标识码不同。这意味着$r\|l\|1\|m_1$从未被 MAC 预言机鉴别过。因为$\text{Mac}-\text{forge}_{\mathcal{A},\Pi}(n)=1$，有$\text{Vrfy}'_k(r\|l\|1\|m_1,t_1)=1$，因此，Forge事件在这种情况下发生。

(2) 标识码r使用在$\mathcal{A}$从 MAC 随机预言机获得的多个 MAC 标记中的一个标记中。用m'代表$\mathcal{A}$查询随机预言机的消息，该消息的应答t'中包含标识码r。因为$m \notin \mathcal{Q}$，有$m' \neq m$。用l'表示m'的长度。有两种情况：

① 情况 1：$l' \neq l$。则依旧有$r\|l\|1\|m_1$从未被 MAC 预言机鉴别过。（MAC 预言机应答有两种情况：一个使用不同的标识码$r' \neq r$，以及使用相同标识码但长度不同$l' \neq l$）。因为Mac - forge$_{\mathcal{A},\Pi}(n)=1$，所以Forge事件在这种情况下发生。

②情况 2：$l'=l$。将m'分成d个分块$m'_1,\cdots,m'_d$（既然$l'=l$，那么m和m'中分块数量是相同的）。因为$m' \neq m$，肯定存在某些i使得$m'_i \neq m_i$。于是存在$r\|l\|i\|m_i$从未被 MAC 预言机鉴别过。（预言机应答分下述情况：标识码不同$r' \neq r$；使用相同标识码r，但序列号不同$i' \neq i$，阶段$m'_i \neq m_i$。）因为Mac - forge$_{\mathcal{A},\Pi}(n)=1$，Forge事件在这种情况下发生。

(3) 标识码r使用在多个 MAC 标记中，这些标记是$\mathcal{A}$从预言机得到的。这种情况是不可能发生的，因为Repeat事件不发生。

上面的讨论涵盖了所有可能发生的事件，从而证明了该项声称。这是证明的核心部分。

返回来看式 4.3 使用上面提到的两项声称，有：

$$\Pr[\text{Mac - forge}_{\mathcal{A},\Pi}(n)=1 \wedge \overline{\text{Repeat}} \wedge \text{Forge}] \geqslant \varepsilon(n) - \text{negl}(n)。$$

现在构造概率多项式时间的敌手$\mathcal{A}'$攻击固定长度的 MAC Π'，并且对事先未鉴别过的消息成功输出有效的伪造标记，敌手成功的概率至少要和以上的式子相同。$\mathcal{A}'$的构造方法十分明显，简单描述如下：$\mathcal{A}'$将$\mathcal{A}$作为子程序运行，并应答$\mathcal{A}$的 MAC 标记查询，当查询消息m的 MAC 标记时，选择$r \leftarrow \{0,1\}^{n/4}$，对$m$进行恰当解析，并查询自己的随机预言机。当$\mathcal{A}$输出$(m,t)$时，消息$m$的长度为$l$，敌手将$m$解析成$m_1,\cdots,m_d$，并将$t$看做$\langle r,t_1,\cdots,t_{d'}\rangle$，检查前

面未鉴别的分块$r\|l\|i\|m_i$（这很容易做到，因为$\mathcal{A}'$能追踪它对自身预言机所发出的所有查询）。如果存在这样的一个分块，$\mathcal{A}'$输出$(r\|l\|i\|m_i, t_i)$；如果不存在，则什么也不输出。

当Forge事件发生时，$\mathcal{A}'$能对未鉴别的消息伪造一个有效的标记（对Π'而言），因此，$\mathcal{A}$能对未鉴别的消息伪造一个有效的标记（对Π而言），易知，$\mathcal{A}$作为$\mathcal{A}'$的子程序运行得到的分布，与$\mathcal{A}$在Mac -forge$_{\mathcal{A},\Pi}(n)$实验中得到的分布是一样的。因此有

$$\begin{aligned}&\Pr[\text{Mac- forge}_{\mathcal{A}',\Pi'}(n)=1]=\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n)=1\wedge \text{Forge}]\\&\geqslant \Pr[\text{Mac}-\text{forge}_{\mathcal{A},\Pi}(n)=1\wedge \overline{\text{Repeat}}\wedge \text{Forge}]\\&\geqslant \varepsilon(n)-\text{negl}(n)\end{aligned}$$

因为Π'安全，所以$\Pr[\text{Mac-forge}_{\mathcal{A}',\Pi'}(n)=1]$可忽略，于是，$\varepsilon$是可忽略的。 ■

4.5 CBC-MAC

定理 4.4 和定理 4.6 一起表明：在仅给定一个定长输入的伪随机函数的情况下，可以构造（任意长度的消息）安全的消息鉴别码。这原则上说明 MAC 能通过分组加密构造。不幸的是，该构造方法非常低效：为了计算消息长度为$l\cdot n$的 MAC 标记，需要应用分组加密$4l$次，MAC 标记长度为$(4l+1)n$比特。幸运的是，还有更有效的解决办法。

CBC-MAC 的构造方式与 CBC 加密模式相似，且在实践中广泛使用。在构造方法 4.5 中，消息被分成许多块，然后对分块加密。然而为了计算消息长为$l\cdot n$的标记，这里n是消息分块的数量，分块加密需要被使用l次。更重要的是，MAC 标记现在的长度只有n位（即单个分块的长度）。这里给出基本的 CBC-MAC 构造方法，该方法给出定长的安全 MAC（对于事先设定好的任意长度的消息）。需要注意的是，这个基本方案在一般情况下是不安全的，这种一般情况是指当不同长度的消息需要鉴别。

构造方法 4.9

令F是一个伪随机函数，固定的长度函数l。基本的 CBC-MAC 构造如下：

- Gen：输入1^n，均匀随机选择$k\leftarrow\{0,1\}^n$；
- Mac：输入一个密钥$k\in\{0,1\}^n$和长度为$l(n)\cdot n$的消息m，完成如下（令$l=l(n)$）：

(1) 将m解析分块$m_1,\cdots,m_l$，每个m_i的长度为n，并令$t_0:=0^n$；

(2) 对$i=1,\cdots,l$，令$t_i=F_k(t_{i-1}\oplus m_i)$。

输出t_l作为标记。

- Vrfy：输入一个密钥$k\in\{0,1\}^n$，长度为$l(n)\cdot n$的消息m，和长度为n的标记t，当且仅当$t\stackrel{?}{=}\text{Mac}_k(m)$时，输出 1。

定长消息的 CBC-MAC

构造方法 4.9 的图示见图 4.1（修改成用来处理可变长度的消息，后面将会讨论）。这个构造方法用来描述长度为 n 的倍数的消息，但是它也可以通过填充来处理任意长度的消息。该构造方法的安全性见如下定理。

定理 4.10 令l是一个多项式，如果F是一个伪随机函数，构造方法 4.9 就是一个长度为$l(n)\cdot n$的消息的定长 MAC。该 MAC 是适应性选择消息攻击下存在性不可伪造的。

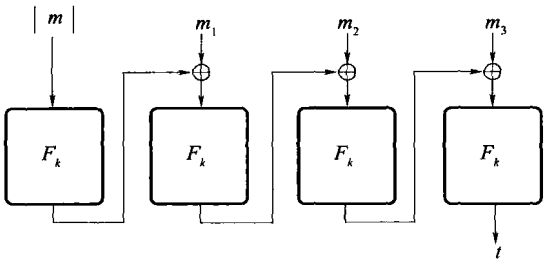


图 4.1　安全地鉴别变长消息的 CBC-MAC 变体

对定理 4.10 的证明十分复杂，因此从略。注意，尽管构造方法 4.9 可以被扩展以处理任意 n 的倍数长度的消息，但是只有在消息的长度是固定时，MAC 才是安全的。也就是说，如果对手能够获得可变长度消息的 MAC 标记，这个方案就不安全了（见练习 4.8）。该构造方法的优势超过了构造方法 4.3。构造方法 4.3 也给出了一个固定长度的 MAC，但是构造方法 4.9 能鉴别长得多的消息。

CBC-MAC 与 CBC 加密模式的比较。基本的 CBC-MAC 与 CBC 模式加密有两个不同：

(1) CBC 加密模式使用一个随机初始向量（IV），这对安全性来说非常重要。与此相反，CBC-MAC 不用 IV（或者说 IV 为固定值，$IV = 0^n$），这对其安全性来说是十分重要的。确切地说，CBC-MAC 使用随机的 IV 反而是不安全的。

(2) 在 CBC 模式加密中，所有的分块t_i（在 CBC 模式加密中被叫做c_i）都被加密算法作为密文的一部分输出，但是在 CBC-MAC 中只输出最后分块。这看起来是技术上的差别，基于这一事实：在加密中，所有的分块都要输出以进行解密，但是对 MAC 而言这是不必要的，而且 CBC-MAC 输出所有分块时反而是不安全的。

在练习 4.9 中，读者被要求验证上面的观点。这是一个很好的例证：看起来无害的对密码学构造的改变可能会导致不安全。因此，准确地执行密码学构造方案是十分重要的，不要轻易引入任何改变（除非能证明改变后的方案是安全的）。此外，理解这些构造方案也是十分必要的。举例来说，在很多案例中，密码库提供一个“CBC 函数”给程序员，但是没有区分该程序是用来进行加密还是进行消息鉴别。

变长消息的安全 CBC-MAC。为了得到可变长度消息 CBC-MAC 的安全版本，构造方法 4.9 需要修改。可以用很多种方法进行修改。下面有三种可供选择的办法能被证明是安全的：

(1) 对长度为l的输入消息应用伪随机函数（分组加密）来获得一个密钥k_l（令$k_l := F_k(l)$），然后使用密钥k_l计算基本 CBC-MAC。这确保了不同的（独立计算的）密钥用于不同长度的消息鉴别。

(2) 预先考虑长度为$|m|$的消息（记为n比特字符串），然后对该消息计算出基本 CBC-MAC（参见图 4.1）。注意将分块的长度附加到消息末尾是不安全的。

(3) 改变方案，这样生成两个不同的密钥，$k_1 \leftarrow \{0,1\}^n$和$k_2 \leftarrow \{0,1\}^n$。然后，鉴别消息m，首先使用k_1计算出m的基本 CBC-MAC，记为t，然后输出结果$\widehat{t} := F_{k_2}(t)$。

第三种方案有优势，因为在计算 MAC 之前不用知道消息的长度。它的劣势是需要两份密钥。然而，以伪随机函数的两个额外应用为代价，可以存储单一的密钥k，并在开始计算前推导出$k_1 = F_k(1)$和$k_2 = F_k(2)$。

4.6 抗碰撞散列函数

在本节中，先从消息鉴别码的讨论引出抗碰撞散列函数的概念，它在整个密码学中非常有用。在这里提出是因为在下一节中要使用它，以得到另一种变长消息的 MAC 有效构造方法。（不研究 4.7 节的读者可以延迟本节的阅读直到第 12 章。）

总的来说，散列函数是将任意长度的字符串压缩成短的字符串的函数。散列函数的经典运用是在数据结构中，它能用来获得$\mathcal{O}(1)$的插入和查找时间来保存集合中的元素。特别地，如果散列函数H的大小为N，需要初始化一个长度为N的表格。然后，元素x被存储在表格的$H(x)$单元中。为了检索x，它需要计算出$H(x)$，并取出该单元中的元素。为了达到这个目的，一个“好的”散列函数应该尽可能地避免碰撞，碰撞是指一对不同的数据x和x'，有$H(x)=H(x')$。注意，当碰撞发生时，两个元素被储存在同一个单元格。因此，多个碰撞导致不理想的检索复杂度。简言之，散列函数将元素良好地分布到表中时，是理想的，从而尽可能减少碰撞的数量。

抗碰撞散列函数大体上与数据结构中所使用的相类似。尤其是，都是用来将可变长度的字符串压缩成固定长度的字符串。在数据结构中，目的是为了避免碰撞，但是，标准散列函数与抗碰撞散列函数有根本性的差别。例如，在数据结构中减少碰撞是理想的需求，而在密码学中的是强制性的需求。此外，在数据结构的情形中，假设数据元素的选择是独立于散列函数的，并没有试图引发碰撞。但是在密码学中正好相反，敌手可能会依赖于散列函数来选择元素，具有制造碰撞的明确目标。这意味着密码学中散列函数的要求比数据结构中的要求更加严格，因此抗碰撞散列函数更难构造。

4.6.1 抗碰撞的定义

函数中的碰撞是指有一对不同的输入量x和x'满足$H(x)=H(x')$；这种情况也称作x和x'在H下发生碰撞。如果任何概率多项式时间的敌手在H中都不能发现碰撞，就称H是抗碰撞的。典型地，会对具有无穷定义域（例如，所有可能长度的字符串作为输入）和有限值域的函数H感兴趣。在这样的一个函数中，根据鸽巢原理，碰撞是必然存在的，因而要求仅仅是碰撞“很难”发现。有时候会对定义域和值域都是有限的函数H感兴趣，在这种情况下，仅对那些压缩输入量的函数感兴趣，这意味着输出量的长度小于输入量的长度。如果不要求压缩，抗碰撞就很容易，比如，标识函数就是抗碰撞的。

下面正式处理一系列散列函数，这些函数使用一个“密钥”s来索引。也就是说，H是一个双输入参数的函数，输入一个密钥s和一个字符串x，并输出一个字符串$H^s(x) \stackrel{\text{def}}{=} H(s,x)$。要求是：对随机产生的$s$，很难在$H^s(x)$找到碰撞。密钥$s$不是一个普通的密钥，在伪随机函数的情形中至少有两处与密钥函数不同。首先，不是所有的字符串都需要对应有效的密钥（即$H^s(x)$可能对某些s没有定义）。因此密钥s由Gen算法产生，而不是均匀随机选取的。其次，也许是更重要的，密钥s并不是保密的。它只是用在一个特定的函数H^s中。为了强调密钥s并不是保密的，这里将它标在右上方，记为H^s而不是H_s。

定义 4.11 散列函数是一对概率多项式时间算法(Gen, H)，满足如下条件：

(1) Gen是一个概率算法，输入参数1^n并输出密钥s。假设1^n隐含在s中。

(2) 存在一个多项式l满足：对H输入一个密钥s和一个字符串$x \in \{0,1\}^*$，并输出一个字

符串$H^s(x) \in \{0,1\}^{l(n)}$（$n$是隐含在$s$中的安全参数）。

如果当输入$x \in \{0,1\}^{l'(n)}$和$l'(n) > l(n)$$H^s$有定义，就称$(\mathsf{Gen}, H)$是输入长度为$l'(n)$的定长散列函数。

在定长的例子中，要求l'大于l。这保证了函数是经典的散列函数，即对输入进行了压缩。在一般的例子中，对l没有要求，因为函数的输入是所有（有穷）二进制字符串，特别是比$l(n)$长的字符串。因此，从定义上来说，它也对输入进行了压缩（虽然仅仅对长于$l(n)$的字符串进行了压缩）。

现在进入到安全的定义。通常，首先定义一个散列函数$\Pi = (\mathsf{Gen}, H)$，一个对手$\mathcal{A}$，和一个安全参数n的实验：

碰撞发现实验$\mathsf{Hash}-\mathsf{coll}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$得到密钥s。

(2) 敌手输入s并输出x和x'。（如果Π是输入长度为$l'(n)$的定长散列函数，我们要求$x, x' \in \{0,1\}^{l'(n)}$。）

(3) 当且仅当$x \neq x'$且$H^s(x) = H^s(x')$时，实验的输出被定义为 1。在这种情况下，称$\mathcal{A}$发现了一个碰撞。

抗碰撞的定义表明：敌手在上面的实验中发现碰撞的概率是可忽略的。

定义 4.12 散列函数$\Pi = (\mathsf{Gen}, H)$是抗碰撞的，如果对于所有的概率多项式时间敌手$\mathcal{A}$，存在着一个可忽略的函数negl，满足

$$\Pr[\mathsf{Hash}-\mathsf{coll}_{\mathcal{A},\Pi}(n) = 1] \leqslant \mathsf{negl}(n)。$$

术语：简单地说，认为H, H^s和$\Pi = (\mathsf{Gen}, H)$是“抗碰撞散列函数”，不会导致任何歧义。

4.6.2 散列函数安全性的一个较弱的定义

抗碰撞是一个较强的安全性要求，并且很难达到。然而，在某些应用程序里，依赖更松弛的要求已经足够。当说到密码学散列函数，有三种典型的安全水平：

(1) 抗碰撞。这是最强的安全定义，是我们目前为止讨论过的。

(2) 抗第二原像。非正式地讲，一个散列函数是抗第二原像的，如果给定输入s和x，对概率多项式时间的敌手来说，找到$x \neq x'$满足$H^s(x') = H^s(x)$是不可行的。

(3) 抗原像。非正式地讲，一个散列函数是抗原像的，如果给定输入s和$y = H^s(x)$（不是x本身），对概率多项式时间的敌手来说，找到一个x'使得$H^s(x') = y$是不可行的。（本书后面章节提到这实质上意味着H^s是单向的）。

没有正式地定义后两个名词，因为它们在本书中未被使用，把这个任务留作练习。

任何一个抗碰撞的散列函数都是抗第二原像和抗原像的。直观地说，这是因为如果给定的x，敌手能发现$x \neq x'$有$H^s(x') = H^s(x)$，那么就很容易发现一对碰撞的x和x'。同样地，任何一个抗第二原像的散列函数都是前像抵抗的。这是因为：如果能反转y得到一个x'使得

$H^s(x') = y$，那么就可能：对于给定的x，计算出$y := H^s(x)$，然后将y反转得到x'，且$x \neq x'$的概率很高（依赖于H是压缩函数这一事实，且很多不同的输入映射到相同的输出），这样，第二原像就找到了。（练习 4.10 中要求形式化上述论述。）结论是：上述三种安全要求形成一种层次，每个定义蕴涵着下面一个定义。

4.6.3 通用“生日”攻击

在显示如何构造抗碰撞散列函数之前，介绍在任意散列函数中发现碰撞的通用攻击（即使时间与散列函数输出长度成指数关系）。这个攻击意味着对散列函数来说，需要一个最小输出长度，以保证对运行一定时间的敌手来说是安全的。下面进行解释。

假设给定一个散列函数$H : \{0,1\}^* \to \{0,1\}^n$。（为了简单起见，假定散列函数的输入为任意长度，尽管攻击略微改动后对定长散列函数也适用。既然攻击独立于任何密钥，在这里省略密钥s。）攻击照下面的方式进行：任意选择q个不同的输入$x_1, \cdots, x_q \in \{0,1\}^{2l}$，计算$y_i := H_{x_i}$，并检查是否有两个$y_i$相等。

这种算法发现一个碰撞的概率是多少？显然，如果$q > 2^l$，该事件的概率为 1。然而，感兴趣是更小的q。分析它的确切概率有一定难度，所以转而分析理想情况：视H为一个随机函数[5]。也就是说，对于每一个i，假设$y_i = H(x_i)$的值均匀分布在$\{0,1\}^l$中，并且独立于任何先前输出的值$\{y_i\}_{j<i}$（别忘了假设所有$\{x_i\}$是不同的）。于是把问题精简如下：如果均匀随机选择$y_1, \cdots, y_q \leftarrow \{0,1\}^l$，那么存在不同的$i, j$，使得$y_i = y_j$的概率是多少？

这个问题已被深入研究过，且与附录 A.4 中讨论的所谓“生日问题”有关。鉴于此，所描述的碰撞寻找算法通常称为“生日攻击”。对生日问题的描述如下：如果有q个人在同一个房间里，那么其中有两个人生日是同一天的概率是多少？（假设生日是均匀且独立地分布在非闰年的 365 天中。）这与要探讨的问题完全类似：如果y_i代表第i个人的生日，那么有$y_1, \cdots, y_q \leftarrow \{1, \cdots, 365\}$是均匀随机选择，生日匹配就是挑选$i, j$使得$y_i = y_j$（即生日匹配意味着碰撞）。

当$q = \theta(2^{l/2})$时，这种碰撞的概率大约是 1/2。在生日攻击的案例中，如果有 23 个人在房子里，那么两个人生日相同的概率就大于 1/2。这在附录 A.4 中得到证明。

散列函数的生日攻击的总结。如果一个散列函数的输出长度是l比特，则生日攻击通过$\mathcal{O}(q) = \mathcal{O}(2^{l/2})$次散列函数求值，以一个较高概率找到一个碰撞（通过对输出结果排序，如果存在碰撞，可在$\mathcal{O}(l \cdot 2^{l/2})$时间发现；简言之，假设对$H$的求值能在恒定的时间内进行）。因此能得到结论：若散列函数能抵御在时间T内运行的碰撞寻找攻击，那么散列函数的输出长度至少应该是 2log$T$ 比特。当考虑到安全性的渐进界限，一个简单的尝试$2^l + 1$个元素进行的攻击方法，与一个使用$2^{l/2}$个元素的生日攻击相比，没什么区别：如果$l(n) = \mathcal{O}(logn)$，那么两个攻击都是在多项式时间内运行的，但是如果$l(n)$是超对数的，那么两个攻击都不会在多项式时间内运行。然而，实践中生日攻击有很大的不同。举例来说，假设一个散列函数的输出长度为 128 比特，显然不可能运行2^{128}步来寻找一个碰撞。但是，运行2^{64}次是可行的（虽然仍很困难）。因此，通用生日攻击的存在要求实践中的任何抗碰撞散列函数的输出长度必须大于 128 比特。有一个足够长的输出只是满足定义 4.12 的必要条件，但是它远不是充分的。故生日攻击仅仅针对抗碰撞；没有通用的攻击针对抗第二原像以及抗原像，且运行时间少于2^l。

---

⑤ 实际上，可以看到这（本质上）是最坏情况，算法发现碰撞的概率更高，如果H明显偏离随机情况。

改进的生日攻击。上面提到的生日攻击有两个缺陷。首先，它需要一个大量的存储。其次，攻击几乎不能控制碰撞的值。构造更好的生日攻击来克服这两个缺陷是有可能的。

基本的生日攻击要求攻击者存储所有的q值$\{y_i\}$，因为攻击者事先并不知道哪一对值会造成碰撞。这有一个明显的缺陷，因为存储相对于时间来说通常是稀缺资源。举例来说（一个临时的例子），将2^{60}比特与2^{60}CPU 指令进行比较：2^{60}比特是大约 10 亿千兆字节。使用目前最大的商用存储设备也仅仅只能装下大约 1000 个千兆字节，这意味着需要 100 万个这样的存储设备。相比而言，2^{60}个指令需要大约 2 年时间完成（假设 CPU 每秒能处理 250 个千兆指令，这已是目前高端 PC 的配置）。这需要等待很长时间，然而是一种可行的计算。此外，进行这种复杂度的运算实际上不需要借助大范围分布式网络即可完成。

如果空间需求降低，生日攻击会变得可行。事实上，有可能完成生日攻击所使用的时间复杂性和成功率，与使用一个固定数量的存储相似。即采用一个随机的初始值x_0，并且对于$i>0$，计算出$x_i := H(x_{i-1})$和$x_{2i} := H(H(x_{2(i-1)}))$。在每一步中，对$x_i$和$x_{2i}$进行比较；如果相等，那么$x_{i-1}$和$H(x_{2(i-1)})$是一个碰撞（除非刚好相等，但是如果定义$H$是随机函数，这个事件发生的概率可忽略）。关键问题是这里需要的存储仅仅是存储x_i和x_{2i}。这个方法给出碰撞的概率在$q=\theta(2^{l/2})$步中大约为 1/2；8.1.2 节在不同的情形下分析了类似的办法。

第二个缺陷是缺乏对碰撞的控制。尽管没必要为了发现“有意义”的碰撞而去破坏抗碰撞的形式化定义，但是生日攻击能发现“有意义”的碰撞是件很好的事情。假设一个攻击者 Alice 试图找到两个消息x和x'使得$H(x)=H(x')$，此外，第一条消息x应该是她的雇主解释为什么解雇她的一封信，第二条消息x'应该是一个赞美的推荐信。生日攻击仅仅依靠这一事实：$x_1,\cdots,x_q$是不同的，但是这些消息不需要是随机的。因此，可以执行生日攻击，这种攻击通过产生$q=\theta(2^{l/2})$个第一种类型的消息和产生q个第二种类型的消息，然后在这两种类型的消息中寻找碰撞。这看起来似乎不可能；但是，很容易把同一个句子用不同的方式写出来。例如，看看下面的句子：

> It is *hard/difficult/challenging/impossible to imagine/believe* that we will *find/locate/hire another employee/person* having similar *abilities/skills/character* as Alice. She has done *a great/super* job.

需要注意的地方是任何斜体字的组合都是可能的。这个句子能用$4\cdot 2\cdot 3\cdot 2\cdot 3\cdot 2=288$种不同的方式写出来。这只是一个句子，实际上很容易写一封能用2^{64}种方法来写的信（你只需要 64 个同义词）。使用这种方法，攻击者有可能准备$2^{l/2}$封信来解释为什么攻击者被解雇，准备$2^{l/2}$封推荐信；两种不同类型的信中发现碰撞的概率很高。这种攻击需要大量的存储，上文提到的低存储的方法在这里不能用。

4.6.4 Merkle-Damgard 变换

本节介绍一个重要的方法，称为 Merkle-Damgard 变换，在实践中它被广泛使用在抗碰撞散列函数的构造中。这个方法使得固定长度的散列函数转换为成熟的散列函数（即能处理任意长度的输入），同时保持了前者的抗碰撞特性（如果有的话）。这意味着在设计抗碰撞散列函数的时候，可以将注意力限制在固定长度的情况上。这反过来就使得设计实用的抗碰撞散列函数的工作变得更加容易。除了在实践中的大量使用，Merkle-Damgard 变换从理论观点看是有趣的，因为它意味着压缩一个比特与压缩任意的数量是同样容易（或困难）的。

具体而言，考虑这样一种情况：定长输入的抗碰撞散列函数，它对输入量压缩了一半；也就是说，输入的长度为$l'(n) = 2l(n)$，输出的长度为$l(n)$。在练习 4.14 中，读者被要求将构造方法推广到任意$l' > l$的情形。用(Gen, h)表示定长抗碰撞散列函数，并使用它来构造一个抗碰撞散列函数(Gen, H)，这个抗碰撞散列函数的输入为任意长度的量，其输出的长度为$l(n)$。（Gen保持不变。）在许多文献中，定长抗碰撞散列函数叫做压缩函数。Merkle-Damgard 变换的定义见构造方法 4.13，如图 4.2 所示。

构造方法 4.13

令(Gen, h)是一个定长抗碰撞散列函数，其输入量长度为$2l(n)$，输出量长度为$l(n)$。构造变长散列函数(Gen, H)的方法如下：

- Gen：保持不变。
- H：输入密钥s和一个串$x \in \{0,1\}^*$，其长度$L < 2^{l(n)}$。按照下面步骤来完成（假设后面$l = l(n)$）：

  ①设$B := \lceil \frac{L}{l} \rceil$（即$x$中分块的数量）。用 0 把$x$的长度填充成$l$的倍数，将填充后的$x$按长度为$l$比特的分块序列进行解析$x_1, \cdots, x_B$，令$X_{B+1} := L$，其中$L$（长度）用$l$比特表示。

  ②设$z_0 := 0^l$。

  ③对于$i = 1, \cdots, B+1$，计算$z_i := h^s(z_{i-1} \| x_i)$。

  ④输出z_{B+1}。

Merkle-Damgard 变换

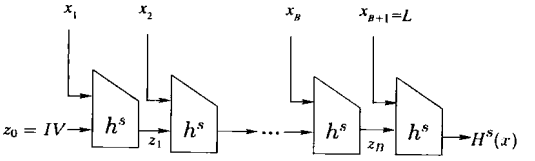


图 4.2　Merkle-Damgard 变换

将x的长度限制在最大值为$2^{l(n)} - 1$上，这样它的长度L能够看作是$l(n)$的一个整数倍。参见脚注③的解释，脚注③说明了为什么在实践中或者理论上这都不是一个限制。

初始矢量值。构造方法 4.13 的第二步中的z_0是任意的，它可以被任何常量代替。这个值通常被称为IV，或初始矢量。

Merkle-Damgard 变换的安全性。Merkle-Damgard 变换的安全性的直观解释是有两个不同的串x和x'，它们在H^s中发生碰撞，在对$H^s(x)$和$H^s(x')$的计算中，必有不同的中间值$z_{i-1} \| x_i$和$z'_{i-1} \| x'_i$，满足$h^s(z_{i-1} \| x_i) = h^s(z'_{i-1} \| x'_i)$。换句话说，若以$h^s$为压缩函数的碰撞存在时，$H^s$中的碰撞才发生。在下文的证明中显示，如果$h^s$中没有碰撞，那么就会有$x = x'$（与$x$和$x'$在$H^s$中碰撞的假设相违背）。现在进入到形式化的证明。

定理 4.14 如果(Gen, h)是一个定长抗碰撞散列函数，那么(Gen, H)是一个抗碰撞散列函数。

证明　证明对于任意s，H^s中的碰撞导致了h^s中的碰撞。假设x和x'是两个不同的串，它们的长度分别为L和L'，满足$H^s(x)$=$H^s(x')$。假设$x_1, \cdots, x_B$是x经过填充后的B个分块，再假设$x'_1, \cdots, x'_{B'}$是x'经过填充后的B'个分块。$x_{B+1} = L$，$x'_{B+1} = L'$。下面有两种情况需要考虑。

(1) 情况一： $L \neq L'$。在这种情况下，计算$H^s(x)$的最后一步是$z_{B+1} := h^s(z_B \| L)$，计算$H^s(x')$的最后一步是$z'_{B'+1} := h^s(z'_{B'} \| L')$。既然有$H^s(x) = H^s(x')$，那么就有$h^s(z_B \| L) = H^s(z'_{B'} \| L')$。然而，$L \neq L'$，所以$z_B \| L$和$z'_{B'} \| L'$就是在$h^s$中发生碰撞的两个串。

(2) 情况二： $L = L'$，这意味着$B = B'$且$x_{B+1} = x'_{B+1}$。假设z_i和z'_i是计算$H^s(x)$和$H^s(x')$的中间值。因为$x \neq x'$，但$|x| = |x'|$，这就至少存在一个索引$i(1 \leqslant i \leqslant B)$，使得$x_i \neq x'_i$。假设$i^* \leqslant B+1$是满足$z_{i^*-1} \| x_{i^*} \neq z'_{i^*-1} \| x'_{i^*}$的最大索引，如果$i^* = B+1$，那么$z_B \| x_{B+1}$和$z'_B \| x'_{B+1}$就是在$h^s$中发生碰撞的两个不同串，因为

$$h^s(z_B \| x_{B+1}) = z_{B+1} = H^s(x) = H^s(x') = z'_{B+1} = h^s(z'_B \| x'_{B+1})。$$

如果$i^* \leqslant B$，那么i^*的最大值就意味着$z_{i^*} = z'_{i^*}$。因此，又一次有$z_{i^*-1} \| x_{i^*}$和$z'_{i^*-1} \| x'_{i^*}$，这就是在h^s中发生碰撞的两个不同的串。

接下来要说的是：在散列函数H^s中的任意碰撞导致了定长散列函数h^s中的碰撞。很容易将其转变为形式化的安全规约，这个留作练习。

4.6.5 实践中的抗碰撞散列函数

在伪随机函数/置换中，构造抗碰撞散列函数有两种方法：基于某些数论假设的可证明安全构造方法[⑥]或者是启发式的高效构造方法。前者将在 7.4.2 节中解释。本节将解释后一种方法，包括仅用在实践中的散列函数。

实践中的抗碰撞散列函数和这里说明的概念有一个重要的不同：运用于实践的散列函数通常是没有密钥的。这意味着一个固定的散列函数H被定义，这样就不存在任何关于Gen算法生成H的密钥的概念（事实上，H不需要密钥）。从一个纯理论角度来看，在任何关于抗碰撞散列函数的讨论中都需要包含有密钥，因为对没有密钥的散列函数定义抗碰撞十分困难[⑦]。关于无密钥的散列函数讨论最多的是：它不可能在合理时间内（如 75 年）发现一个H中的碰撞。实践中，这种类型的安全保证已经足够了。

即使从实用角度来看，有密钥的散列函数也有它的优势：首先，如果在无密钥的散列函数H中发现碰撞（通过使用耗时数年的穷举搜索），那么H就不是抗碰撞的，而且应该被替换掉。如果H是一个有密钥的函数，那么H^s的碰撞被蛮力搜索得到，并不会使在使用新密钥s'的$H^{s'}$中发现碰撞也变得容易；因而，只要密钥进行了更新，H可继续使用。

尽管不可能希望证明实践中的散列函数是抗碰撞的（特别是因为它们没有密钥），这并不意味着在一些大的构造方案中使用这种散列函数时可以不需要证明。抗碰撞的安全证明都表明：如果构造方案能被某个多项式时间敌手“攻破”，那么就能在多项式时间发现该散列函数中的一个碰撞（前面在证明 Merkle-Damgard 变换的时候已有一个这样的例子）。在考虑到无密钥的函数时，有下面的陈述：“如果敌手在一定的时间内攻破了给定的构造方案，那么就有

---

⑥ 有趣的是，构造抗碰撞散列函数需要的假设比那些构造伪随机置换需要的假设更强，有迹象表明这可能是固有的。

⑦ 为得到该技术问题的感觉，令x, x'为散列函数 H 的碰撞（若 H 处理比输出更长的输入，则这样的x, x'的确存在）。现在，考虑（固定时间）算法，简单输出x和x'。这样一个算法以概率 1 发现 H 中的碰撞。注意输出在 H^s 中（s 是随机选择的）碰撞的类似算法不存在。

可能用类似的时间[8]在散列函数中找到碰撞”。如果相信在任意合理时间内从给定的散列函数中很难找到碰撞，那么这就为更大的构造方案的安全性提供了合理的保障。

回到实践中的构造方案中来，之前提到的“生日攻击”给出散列函数的输出长度的下限，该下限可达到一定水平的安全性：如果散列函数对运行2^l时间的敌手是抗碰撞的，那么散列函数的输出长度至少是$2l$比特。好的抗碰撞散列函数在实际中的输出长度最少为 160 比特，这就意味着生日攻击的运行时间为2^{80}，这在目前是不可能达到的。

两个比较流行的散列函数是 MD5 和 SHA-1。（下文将会讨论，鉴于目前对 MD5 的攻击，MD5 已经不再安全，且不能在要求抗碰撞的应用中使用。这里仍然对它进行讨论是因为它在历史遗留代码中仍有使用。）MD5 和 SHA-1 首先都定义一个压缩函数，对固定长度的输入量进行相对少量地压缩（在概念中，这种压缩函数是一个定长输入抗碰撞散列函数）。然后 Merkle-Damgard 变换（或者其他类似的）被应用在压缩函数中以得到一个针对任意输入长度的抗碰撞散列函数。MD5 的输出长度为 128 比特，SHA-1 的输出长度为 160 比特。SHA-1 的输出长度更长，对它进行通用“生日攻击”难度更大：对 MD5 来说，一个生日攻击要求$2^{128/2} = 2^{64}$次散列运算，但对 SHA-1 来说，同样的攻击则要求$2^{160/2} = 2^{80}$次散列运算。

在 2004 年，一个中国密码专家小组在对 MD5 以及相关散列函数的攻击上取得了突破。其发现碰撞的技术对碰撞点几乎不能控制；尽管如此，他们所采用的方法（实际上，任何能发现“随机碰撞”的方法）都可以用来发现碰撞，比如，两个可读的 postscript 文件。一年之后，这个中国小组展示了（理论上）对 SHA-1 的攻击，能够找到碰撞，且使用的时间比通用生日攻击使用的时间少。对 SHA-1 的攻击要求进行2^{69}次运算，这超出了目前可行的范围；在 SHA-1 中依旧没有发现明显的碰撞。（与 MD5 的攻击相比而言，后者在几分钟之内就发现了碰撞。）

这些攻击导致寻找更强的散列函数，即拥有更大的输出长度，且不易受到已知的针对 MD5 和 SHA-1 的攻击的影响。值得注意的是 SHA-2 家族，扩展了 SHA-1，且包含了输出长度为 256 和 512 比特的散列函数。这些攻击的分支促使设计新散列函数并发展新散列函数标准。

4.7 *NMAC 与 HMAC

前面解释了基于伪随机函数（或分组密码）的消息鉴别码的构造方法。NMAC 与 HMAC 的构造方法中使用了不同的技术：通过在压缩函数（即定长抗碰撞散列函数）上应用 Merkle-Damagard 变换来构造抗碰撞散列函数。大部分已知的散列函数都是用这种方式构造的，因而 NMAC 和 HMAC 有广泛的应用。不严格地说，NMAC 和 HMAC 的安全基于以下的假设：压缩函数除了抗碰撞外，还具有某些伪随机的特性。这个假设对大多数实践中使用的压缩函数来说是正确的。下面将讨论压缩函数所需要的更精确的假设。

下面描述 NMAC，并简单讨论其安全性。然后介绍 HMAC，它可以看作是 NMAC 的一个特例。通常我们对构造法的描述并不完整，特别是我们没有描述在计算前如何填充输入的消息。

---

⑧ 实际上，并不总是这样：在某些证明中，一个运行时间为t的敌手“攻破”构造方案可转变为一个运行时间长得多的敌手（比如说t^2）来发现一个碰撞。在这种情况下，安全性保证可能不是十分有用。

定义 Merkle-Damgård 中的IV。本节会对构造方法 4.13 中的 Merkle-Damgård 变换所使用的IV进行明确介绍；回忆一下，它是赋给z_0的固定值。在标准的 Merkle-Damgård 构造方法中，IV固定为一个任意常量。这里对它有所改变，用$H^s_{IV}(x)$表示构造方法 4.13 中带有密钥s的对输入量x的计算，并将z_0的值设定为$IV \in \{0,1\}^l$。

4.7.1 嵌套 MAC(NMAC)

假设H是一个对压缩函数h使用 Merkle-Damgård 变换而构造的散列函数。为了简单起见，用n表示H和h的输出长度（而不是之前用到的$l(n)$），并仍假设h将它的输入长度压缩一半。构造 NMAC 的第一步是定义一个有保密密钥的压缩（散列）函数。假设s是一个固定的不保密的密钥，定义一个有保密密钥的h^s如下：$h^s_k(x) \stackrel{?}{=} h^s(k\|x)$。也就是说，有保密密钥的压缩函数使用无密钥的压缩函数，该函数的输入为密钥k和消息的连接。定义H^s_k为散列函数，通过在 Merkle-Damgard 构造方法中设定$IV = k$得到。（这和上面引入的$H^s_{IV}(x)$有同样的定义。）注意第一个 Merkle-Damgard 变换的迭代中，$z_1 := h^s(IV\|x_1)$。这里$IV = k$，因此得到$z_1 = h^s(k\|x_1) = h^s_k(x_1)$。

现在定义NMAC。用文字描述，NMAC首先用密钥保密的抗碰撞散列函数$H^s_{k_2}$来计算输入m，然后将结果输入到一个密钥保密的压缩函数$h^s_{k_1}$。（见构造方法 4.15 和图 4.3）称$H^s_{k_2}$为内部函数，称$h^s_{k_1}$为外部函数。

构造方法 4.15

令$(\widetilde{\mathsf{Gen}}, h)$是一个定长抗碰撞散列函数，并令$(\widetilde{\mathsf{Gen}}, H)$是对$(\widetilde{\mathsf{Gen}}, h)$应用 Merkle-Damgård 变换的结果。NMAC 的定义如下：

- Gen：输入1^n，运行$\widetilde{\mathsf{Gen}}(1^n)$得到一个密钥$s$。同样随机选择$k_1, k_2 \leftarrow \{0,1\}^n$，输出密钥$(s, k_1, k_2)$。
- Mac：对输入的密钥(s, k_1, k_2)和消息$m \leftarrow \{0,1\}^*$，输出标记$t := h^s_{k_1}(H^s_{k_2}(m))$。
- Vrfy：对输入的密钥(s, k_1, k_2)和消息$m \leftarrow \{0,1\}^*$以及标记t，当且仅当$t \stackrel{?}{=} \mathsf{Mac}_{s,k_1,k_2}(m)$时，输出 1。

嵌套 MAC(NMAC)

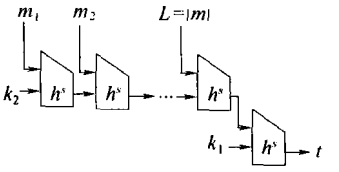


图 4.3 NMAC

NMAC 的安全性依赖于一个假设：含有密钥k的h^s_k是安全的 MAC。正式地讲，给定一个定长散列函数$(\widetilde{\mathsf{Gen}}, h)$，定义定长的消息鉴别码$\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$，鉴别长度为$n$的消息，定义如下：$\mathsf{Gen}(1^n)$运行$(\widetilde{\mathsf{Gen}}(1^n)$获得$s$，并均匀随机选取一个密钥$k \leftarrow \{0,1\}^n$；密钥为$(s,k)$。然后对$m \in \{0,1\}^n$设定$\mathsf{Mac}_{s,k}(m) \stackrel{\mathrm{def}}{=} h^s_k(m)$，对Vrfy的定义照常。如果这种方法定义的$\Pi$是一个安全的定长 MAC，那么$(\widetilde{\mathsf{Gen}}, h)$产生的是一个安全的 MAC。

定理 4.16 假设$(\widetilde{\mathsf{Gen}}, H)$表示对$(\widetilde{\mathsf{Gen}}, h)$的 Merkle-Damgård 变换，如果$(\widetilde{\mathsf{Gen}}, h)$是抗碰撞的，并产生了一个安全的 MAC（正如上面定义的），那么（对任意长度的消息而言）NMAC 在适应性选择消息攻击下是存在性不可伪造的。

从本质上说，NMAC 构造方法首先将准备鉴别的消息m散列（使用抗碰撞散列函数），然后对得到的结果使用定长消息鉴别码（这个 MAC 基于同一个散列函数）。这与在 12.4 节中大量讨论的“hash-and-sign”方法相似，因此在这里仅就 NMAC 的安全性证明背后的思想做粗略的描述。假设导致一个矛盾：存在一个概率多项式时间的敌手$\mathcal{A}$攻击 NMAC，敌手以不可忽略的概率对一个新消息伪造有效的标记。回忆一下，$\mathcal{A}$被赋予一个 MAC 预言机，该预言机可用来问询任意选择的消息的标记。令m^*代表$\mathcal{A}$准备伪造标记的消息，令$\mathcal{Q}$代表$\mathcal{A}$对它的 MAC 预言机所发出的问询（也就是它获得的 MAC 标记的消息集合）。不失一般性，假设$m^* \notin \mathcal{Q}$（因为$\mathcal{A}$不能成功）。有两种可能：

(1) 情况一：存在一个消息$m \in \mathcal{Q}$，使得$H^s_{k_2}(m^*) = H^s_{k_2}(m)$。在这种情况下，$m$的 MAC 标记与$m^*$的 MAC 标记相等。显然，$\mathcal{A}$能成功地对$m^*$伪造一个标记。然而，这个例子与$H$是抗碰撞的假设相矛盾，因为$\mathcal{A}$发现了不同的$m$和$m^*$有$H^s_{k_2}(m^*) = H^s_{k_2}(m)$。（在定理 4.14 中，$(\widetilde{\mathsf{Gen}}, h)$的抗碰撞意味着对任意的$k_2$，$(\widetilde{\mathsf{Gen}}, H_{k2})$是抗碰撞的。）

(2)情况二：对每一个消息$m \in \mathcal{Q}$，有$H^s_{k_2}(m^*) \neq H^s_{k_2}(m)$，定义$\mathcal{Q}' = \{H^s_{k_2}(m) | m \in \mathcal{Q}\}$。这里一个重要的观察是：$m^*$满足$H^s_{k_2}(m^*) \notin \mathcal{Q}'$。在这个例子中，$\mathcal{A}$在“新消息”上伪造了一个定长消息鉴别码 II 的有效的标记$H^s_{k_2}(m^*)$，定长消息鉴别码Π将会在接下来的定理 4.18 前面部分进行描述，这与假设Π是一个安全 MAC 矛盾。

对上面的正式证明请读者参考定理 12.5，那里对数字签名进行了几乎相同的证明。很容易将那个证明转换到这里使用。

安全假设。注意假设是抗碰撞的，不意味着$(\widetilde{\mathsf{Gen}}, h)$给出安全的 MAC。毕竟，存在实用的压缩函数（就像在 SHA-1 中使用的那个），假设它满足附加的要求（给出安全的 MAC）。

为保证 NMAC 的安全性，构造方法 4.15 中的密钥s不需保密。于是，在系统范围内，s一旦选择即可被所有人使用，其实这正是实践中所做的，使用一个无密钥的散列函数（可视为使用固定密钥s的散列函数）。

一旦假设$(\widetilde{\mathsf{Gen}}, h)$抗碰撞，就可以不需要构造方法 4.15 中的密钥$k_2$。（也就是说，可以在 Merkle-Damgard 变换的原始描述中简单地固定$k_2 = 0^n$。）其实上面证明中并不依靠k_2，在 NMAC 中引入k_2的原因是因为它可证明 NMAC 的安全，这一证明是基于一个潜在更弱的安全性假设。特别地，考虑下面修改的抗碰撞定义：密钥s是由$\widetilde{\mathsf{Gen}}$生成，随机选择$k_2 \leftarrow \{0,1\}^n$，那么敌手允许和“散列函数预言机”交互，该预言机返回对x的问询$H^s_{k_2}(x)$。（一个变化方案是给敌手一个非机密的密钥s。）如果能输出不同的输入x，x'，满足$H^s_{k_2}(x) = H^s_{k_2}(x')$，则敌手成功。如果每个概率多项式时间的敌手$\mathcal{A}$在实验中以可忽略的概率取得成功，称$(\widetilde{\mathsf{Gen}}, H)$是弱抗碰撞的。如果$(\widetilde{\mathsf{Gen}}, H)$是抗碰撞的，那么它也是弱抗碰撞的；但后者是更容易满足的较弱的条件。

弱抗碰撞是证明 NMAC 是安全 MAC 的充分条件，对前面给出的证明概要进行仔细检查可以体会到这一点。注意，如果s是一个系统范围的参数（当无密钥的散列函数被使用时），

针对某个IV在H^s_{IV}中发现碰撞的蛮力攻击，对随机选择的k_2对应的$H^s_{k_2}$中发现碰撞并无帮助。如果k_2增加了保密性则该论断更加正确。尽管使用q次调用散列函数（例如使用生日攻击$q=2^{n/2}$）可能发现$H^s_{k_2}$中的碰撞，但是敌手从诚实方获得q个合法标记是困难的。为了使攻击更加困难，敌手甚至得不到$H^s_{k_2}(x)$；相反，只能得到$h^s_{k_1}(H^s_{k_2}(x))$，这至少隐藏了$H^s_{k_2}(x)$的大部分信息。

4.7.2 HMAC

NMAC 的一个缺点是：散列函数的IV需要修改。在实践中，这可能导致复杂化，因为在 SHA-1 中，IV要求是固定的，现有的密码学函数库不允许输入外部的IV。HMAC 通过对压缩函数和散列函数使用不同的密钥来解决这个问题。另一个不同是：HMAC 使用的是一个密钥，而不是两个。

H和h与在上一节所表示的一样。继续假设H和h的输出长度为n比特，且h对它的输入压缩一半。令初始矢量IV为一个固定值，且假设它超出了诚实方的控制，假设当$x\in\{0,1\}^n$时（即x为单一分块时），对$H^s_{IV}(x)$的计算仅仅涉及对压缩函数h^s的调用。这在技术上和对 Merkle-Damgård 变换的描述不同，这是因为长度常作为添加的分块。但是，在实践中当输入量足够小的时候，这个长度被包含在第一个分块中。为了简化这个问题，当x的长度为n时，忽略两个长度的连接，并设$H^s(IV)\|x)=h^s_{IV}(x)$。

构造方法 4.17

令$(\widetilde{\mathsf{Gen}},h)$是一个定长抗碰撞散列函数，$(\widetilde{\mathsf{Gen}},H)$是对$(\widetilde{\mathsf{Gen}},h)$应用 Merkle-Damgård 变换的结果。$IV$，opad，ipad的长度为$n$的常量。HMAC 的定义如下：

- Gen：输入参数1^n，运行$\widetilde{\mathsf{Gen}}(1^n)$得到密钥$s$。同样随机选择$k\leftarrow\{0,1\}$，输出密钥$(s,k)$。
- Mac：对输入的密钥(s,k)和长度为L的消息$m\in\{0,1\}^*$，输出标记$t=H^s_{IV}((k\oplus\mathsf{opad})\|H^s_{IV}((k\oplus\mathsf{ipad})\|m))$
- Vrfy：对输入的密钥(s,k)和信息$m\in\{0,1\}^*$以及标记t，当且仅当 $t=\mathsf{Mac}_{s,k}(m)$时，输出 1。

HMAC

HMAC 使用了两个常量opad和ipad。这两个串的长度为n（即，与输入到H的单个分块的长度相等），对它们的定义如下：opad由字节“0x36”根据需要不断重复组成，ipad则由“0x5C”以同样的方法组成。

HMAC 由构造方法 4.17 给出，如图 4.4 所示。初看它与构造方法 4.15 非常不同。但是，可以将 HMAC 看作是 NMAC 的一个变种。注意，在计算内部散列函数$H^s_{IV}((k\oplus\mathsf{ipad})\|m)$时，第一步是计算$k_2\stackrel{\mathsf{def}}{=}z_1:=h^s((k\oplus\mathsf{ipad}))$。因此有：

$$H^s_{IV}((k\oplus\mathsf{ipad})\|m)=H^s_{k_2}(m)$$

同样地，在计算外部散列函数时，第一步是计算$k_1\stackrel{\mathsf{def}}{=}z'_1:=h^s(IV\|(k\oplus\mathsf{opad}))$。因此有：

$$H^s_{IV}((k \oplus \text{opad}) \| H^s_{IV}((k \oplus \text{ipad}) \| m)) = H^s_{k_1}(H^s_{k_2}(m))。$$

既然$H^s_{k_2}(m)$的长度只有一个分块那么长，那么反过来它与$H^s_{k_1}(H^s_{k_2}(m))$相等（从上面对H^s的假设可看出）。因此，这在 NMAC 中相当于使用两个相互依赖的密钥k_1, k_2。（这些密钥均是从k中使用确定的方法衍生出来的，因而不是独立的）。

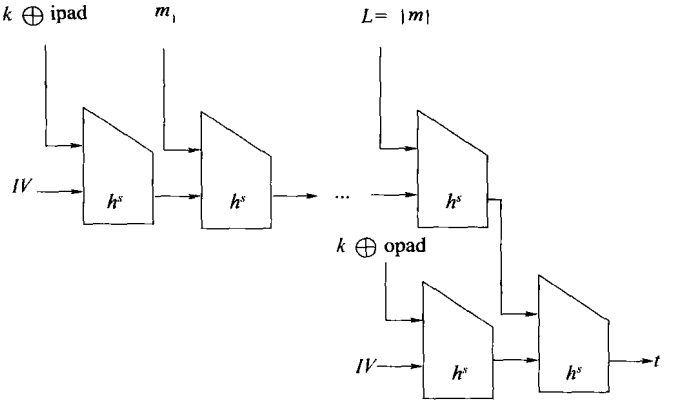


图 4.4 HMAC

定义

$$G(k) \stackrel{\text{def}}{=} h^s(IV \| (k \oplus \text{opad})) \| h^s(IV \| (k \oplus \text{ipad})) = k_1 \| k_2。 \tag{4.4}$$

如果G是一个伪随机发生器，并且$k \leftarrow \{0,1\}^n$是均匀随机选择的，那么相互依赖的密钥k_1，k_2“看起来”是独立的，均匀选择的密钥也可以这么认为。因此，如果假设G是一个伪随机发生器，那么 HNAC 的安全归约到 NMAC 的安全。有下面的定理：

定理 4.18 假设$(\widetilde{\text{Gen}}, h)$满足定理 4.16 中的条件，如果$G$如式 4.4 中定义，是一个伪随机发生器，那么 HMAC 是适应性选择消息攻击下存在性不可伪造的（对任意消息而言）。

实际应用中的 HMAC。 HMAC 是工业标准，且广泛应用于实践中。它很高效而且很容易实现，且可以进行安全证明（根据假设实践中的抗碰撞散列函数是抗碰撞的）。HMAC 的重要性部分归功于它出现的及时性。在引入 HMAC 之前，许多实践者拒绝使用 CBC-MAC（声称它“太慢”），并坚持使用启发式的消息鉴别码，但是那些通常不安全。比如说，一个常见的错误是将 MAC 定义为$H^s(k \| x)$（其中k是一个密钥）。不难发现，当H通过使用 Merkle-Damgård 变换时，这完全不是一个安全的 MAC。

4.8 *构造 CCA 安全加密方案

3.7 节介绍了对称密钥加密方案的 CCA 安全（这里就不回顾概念了，读者可以自行参阅前面的章节）。在本节中，使用消息鉴别码与 CPA 安全加密方案一起，构造对称密钥加密方案来满足这种安全性。

构造 CCA 安全加密方案。 这种构造方法按下面的方式工作：发送方和接收方共享两个密钥，一个是 CPA 安全加密方案的，一个是消息鉴别码的。为了加密消息m，发送者先使用

CPA 安全的方案加密，然后对得到的密文计算 MAC 标记t；现在整个密文是$\langle c,t\rangle$。对于给定的密文$\langle c,t\rangle$，接收方在解密c之前验证标记的有效性。

如果t是关于c的有效 MAC 标记，称密文$\langle c,t\rangle$是有效的。该构造方法的作用是：敌手将不能生成任何有效的由诚实方发出的密文（从技术上讲，需要 MAC 这一附加条件，但这只是一种直觉）。它会使解密预言机无用，正如将从安全性证明中看到的那样。

构造方法 4.19

令$\Pi_E=(\mathsf{Gen}_E,\mathsf{Enc},\mathsf{Dec})$是一个对称密钥加密方案，并令$\Pi_M=(\mathsf{Gen}_M,\mathsf{Mac},\mathsf{Vrfy})$是一个消息鉴别码。定义一个加密方案如下$(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$：

- (Gen')：输入参数1^n，运行$\mathsf{Gen}_E(1^n)$和$\mathsf{Gen}_M(1^n)$，分别得到密钥k_1,k_2。
- (Enc')：输入密钥(k_1,k_2)和明文消息m，计算$c\leftarrow\mathsf{Enc}_{k_1}(m)$和$t\leftarrow\mathsf{Mac}_{k_2}(c)$，并输出密文$\langle c,t\rangle$。
- (Dec')：输入密钥(k_1,k_2)和密文$\langle c,t\rangle$，先检查$\mathsf{Vrfy}_k(c,t)\stackrel{?}{=}1$是否成立，如果是，则输出$\mathsf{Dec}_{k_1}(c)$；否则输出$\perp$。

CCA 安全对称密钥加密方法

在构造方法 4.19 中，扩展了加密方案的语法，为了让加密算法Dec'输出一个特别的标记$\perp$来表示“失败”。正确则意味着没有Enc'的密文输出导致解密算法输出$\perp$。

在证明该构造方案的安全性之前，先介绍对 MAC 方案的额外要求，$(\mathsf{Gen}_M,\mathsf{Mac},\mathsf{Vrfy})$对每一个$k$都有唯一的标记，而对每一个$m$，都有唯一的值$t$满足$\mathsf{Vrfy}_k(m,t)=1$。（这暗示了$\mathsf{Mac}$可能是确定性的。）对唯一标记的要求并不难达到；事实上，构造方法 4.5 中变长 MAC 是目前所见唯一不满足这个性质的。

定理 4.20 如果Π_E是一个 CPA 安全的对称密钥加密方案，且Π_M是一个具有唯一标记的消息鉴别码，那么构造方案 4.19 是一个 CCA 安全的对称密钥加密方案。

证明 该定理的证明过程如下：与前面定义的一样，若$\mathsf{Vrfy}_{k_2}(c,t)=1$则称密文$\langle c,t\rangle$是有效的（对于密钥$(k_1,k_2)$而言）。敌手对解密预言机的问询有两种方式：一种是敌手从其加密预言机得到的密文，另一种不是。第一种解密问询并不是很有用，因为敌手已经知道密文对应的明文。（特别地，如果敌手接收到对消息m的加密预言机问询的结果$\langle c,t\rangle$，那么敌手就会知道$\langle c,t\rangle$的解密结果就是m。）对于第二种类型的密文而言，既然Π_M是一个安全的消息鉴别码，可以认为，（除了可忽略的概率外）所有这种密文是无效的，所以解密预言机会简单地返回$\perp$。因此，第二种问询也是没有用处的。既然解密预言机是无用的，那么$\Pi'=(\mathbf{Gen}',\mathbf{Enc}',\mathbf{Dec}')$的安全性就转化到$\Pi_E$的 CPA 安全性。下面给出正式证明。

令$\mathcal{A}$是一个概率多项式时间的敌手，对构造方法 4.19 进行选择密文攻击（参阅定义 3.30）。设$\mathsf{ValidQuery}$是一个事件，$\mathcal{A}$提交一个问询$\langle c,t\rangle$给解密预言机，该问询未曾从加密预言机获得过，但又满足$\mathsf{Vrfy}_{k_2}(m,t)=1$，有：

$$\begin{aligned}&\Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1]\\&\leqslant\Pr[\mathsf{ValidQuery}]+\Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1\wedge\overline{\mathsf{ValidQuery}}]。\end{aligned}\tag{4.5}$$

将会看到，$\Pr[\mathsf{ValidQuery}]$是可以忽略的，并存在可忽略函数negl，满足$\Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1\wedge\overline{\mathsf{ValidQuery}}]\leqslant\frac{1}{2}+\mathsf{negl}(n)$。这就证明了上面的定理。

声称 4.21 $\Pr[\text{ValidQuery}]$是可忽略的。

直观地，这是由于如果事件ValidQuery发生，那么敌手伪造了新消息c的有效标记t。正式地，假设$q(\cdot)$是一个多项式，该多项式的上界是$\mathcal{A}$对解密预言机查询的次数，考虑到敌手$\mathcal{A}_M$攻击消息鉴别码Π_M（也就是说，运行实验$\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)$）：

敌手$\mathcal{A}_M$：

$\mathcal{A}_M$输入参数为1^n，并能访问预言机$\text{Mac}_{k_2}(\cdot)$。

(1) 均匀随机地选择$k_1 \leftarrow \{0,1\}^n$。

(2) 均匀随机地选择$i \leftarrow \{1,\cdots, q(n)\}$。

(3) 输入参数1^n运行$\mathcal{A}$，当$\mathcal{A}$对消息m做加密预言机问询时，按下面的方式进行回答：

① 计算$c \leftarrow Enc_{k_1}(m)$。

② 对 MAC 预言机做问询c，并接收响应值t，返回$\langle c,t\rangle$到$\mathcal{A}$。

挑战密文也按照同样的方式进行准备（随机选择一个比特位$b \leftarrow \{0,1\}$，对消息m_b进行加密）。

当$\mathcal{A}$对密文$\langle c,t\rangle$做出一个解密预言机问询时，按照下面的方式进行回答：

①如果$\langle c,t\rangle$是之前对消息m所做的加密预言机问询的回应，那么返回m。

②如果这是第i次使用“新”的c值对解密预言机的问询，输出$\langle c,t\rangle$并停止。

③否则，输出$\perp$。

本质上说，$\mathcal{A}_M$希望它对$\mathcal{A}$的第i条解密预言机问询将会是第二种有效问询的第一次。在这种情况下，$\mathcal{A}_M$输出一个对消息c的有效伪造，且c之前未曾提交给 MAC 预言机。

显然$\mathcal{A}_M$在概率多项式时间内运行。现在分析$\mathcal{A}_M$成功的一次伪造即在$\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)$实验中成功生成一个好的标记的概率。关键因素是$\mathcal{A}$作为$\mathcal{A}_M$的子例程运行所见内容（视图）与在实验$\text{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)$中运行的视图，具有完全相同的概率分布，一直到事件ValidQuery发生。为表明这一点，注意到$\mathcal{A}$的加密预言机问询（和计算挑战密文一样）由$\mathcal{A}_M$来完美地模拟。对于$\mathcal{A}$的解密预言机问询，直到ValidQuery发生，都是正常地模拟：在情况①中，这很明显；在情况②或③中，如果ValidQuery没有发生，那么对解密预言机问询回答实际上是$\perp$。结论是：实验$\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)$中ValidQuery事件发生的概率，与在实验$\text{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)$中该事件的概率相等。

如果$\mathcal{A}_M$正确地猜中指数i，该指数表示当ValidQuery事件第一次发生时的解密问询，那么$\mathcal{A}_M$就在实验$\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)$中成功。$\mathcal{A}_M$正确猜中$i$的概率是$1/q(n)$，也就是说：

$$\Pr[\text{Mac-forge}_{\mathcal{A}_M,\Pi_M}(n)=1] \geqslant \Pr[\text{ValidQuery}]/q(n)$$

因为Π_M是安全的 MAC，且q是多项式，结论是：$\Pr[\text{ValidQuery}]$是可忽略的。这就证明了声称 4.21。

声称 4.22 存在一个可忽略的函数negl，满足

$\Pr[\text{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1 \wedge \overline{\text{ValidQuery}}] \leqslant \frac{1}{2}+\text{negl}(n)$。

现在使用Π_E的 CPA 安全。令$\mathcal{A}$的假设与之前一样，考虑下面的敌手$\mathcal{A}_E$使用选择明文攻

击的方法对Π_E进行攻击：

敌手$\mathcal{A}_E$:

$\mathcal{A}_E$的输入为1^n，访问预言机$\mathsf{Enc}_k(\cdot)$。

(1) 均匀随机选择$k_2 \leftarrow \{0,1\}^n$。

(2) 以输入参数1^n运行$\mathcal{A}$，当$\mathcal{A}$对消息m做出一个加密预言机问询时，按照下面的方式进行回答：

① 对加密预言机做问询m，并得到应答c；

② 计算$\mathrm{t} \leftarrow \mathsf{Mac}_{k_2}(c)$，并返回$\langle c,t\rangle$给$\mathcal{A}$；

当$\mathcal{A}$对密文$\langle c,t\rangle$做出一个解密预言机问询时，按照下面的方式进行回答：

如果$\langle c,t\rangle$是之前对m加密预言机问询的应答，那么返回m；否则，输出$\perp$。

(3) 当$\mathcal{A}$输出消息(m_0,m_1)时，输出同样的消息接收一个挑战密文c作为应答。计算$t \leftarrow \mathsf{Mac}_{k_2}(c)$，并将$\langle c,t\rangle$作为挑战密文返回给$\mathcal{A}$。如上所述继续回答$\mathcal{A}$的预言机问询。

(4) 输出与$\mathcal{A}$的输出一样的比特b'。

注意到$\mathcal{A}_E$并不需要一个解密预言机，因为它简单地假设任何$\mathcal{A}$使用“新的”密文$\langle c,t\rangle$来做解密问询是无效的。

显然，$\mathcal{A}_E$在概率多项式时间内运行。而且，只要ValidQuery事件不发生，$\mathcal{A}$作为$\mathcal{A}_E$的子例程的视图概率分布与$\mathcal{A}$在实验$\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)$中视图的概率分布是相同的。（这是因为如果ValidQuery事件不发生，则$\mathcal{A}$做出的任何解密问询的正确应答是$\perp$，这里$\mathcal{A}$使用c的新值。）因此，当ValidQuery事件不发生的情况下，$\mathcal{A}_E$成功的概率与$\mathcal{A}$在ValidQuery事件不发生的情况下成功的概率相等。也就是：

$$\Pr[\mathsf{PrivK}^{cpa}_{\mathcal{A}_E,\Pi_E}(n)=1 \wedge \overline{\mathsf{ValidQuery}}] = \Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1 \wedge \overline{\mathsf{ValidQuery}}]$$

这意味着

$$\begin{aligned}\Pr[\mathsf{PrivK}^{cpa}_{\mathcal{A}_E,\Pi_E}(n)=1] &\geqslant \Pr[\mathsf{PrivK}^{cpa}_{\mathcal{A}_E,\Pi_E}(n)=1 \wedge \overline{\mathsf{ValidQuery}}] \\ &= \Pr[\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi'}(n)=1 \wedge \overline{\mathsf{ValidQuery}}]。\end{aligned}$$

因为Π_E是 CPA 安全的，那么就存在一个可忽略函数negl，满足$\Pr[\mathsf{PrivK}^{cpa}_{\mathcal{A}_E,\Pi_E}(n)=1] \leqslant \frac{1}{2}+\mathsf{negl}(n)$，这意味着声称 4.22 成立。这个定理通过对声称 4.21 和声称 4.22 与等式（4.5）的组合派生。

唯一标记的角色。如果 MAC 方案Π_M没有唯一的标记，那么构造方法 4.19 就不是 CCA 安全的。特别地，可能会很容易修改关于值c的有效标记t，改为一个不同的（但仍然有效的）值t'。如果挑战密文是$\langle c,t\rangle$，那么敌手可以向它的解密预言机问询$\langle c,t'\rangle$，从而得到明文m_b。

比唯一标记特性更弱的条件，也是前面定理的充分条件是：敌手不能发现对未鉴别消息m不同的标记（虽然可能存在不同的标记）。拥有这种特性的安全消息鉴别码被称为强安全的。通常存在大量标记唯一的有效 MAC，所以这一要求并不是局限。

CCA 安全和实用意义。构造方法 4.19 看起来似乎并不令人满意，因为它通过简单地确保解密随机预言机对敌手是无用的来获得 CCA 安全。其实这应被看作是该构造方法的优势而不是缺陷。特别地，虽然有其他的方法也可达到 CCA 安全（例如练习 3.14），这里敌手不能生成任何有效的未曾被诚实方创建过的密文。正如下一节所述，这意味着构造方法 4.19 同时保证了私密性和消息鉴别。

4.9 *获得私密性和消息鉴别

第 3 章学习了如何加密消息以及如何获得私密性，本章讲解消息鉴别码如何保证数据的认证和完整性。然而通常对消息的私密性和完整性都有要求，这就会使得人们这样想：一个安全的加密方案和一个安全的消息鉴别码的组合能够保证这两方面的要求。不幸的是，并不是所有的组合都是这样的。通常，即便是优秀的密码学工具的组合也可能不安全。因此，除非一个特定的组合能被证明是安全的，否则，在使用它的时候仍需小心。

有三种常见的方法对加密和消息鉴别进行组合。假设k_1代表一个加密密钥，k_2代表一个 MAC 密钥[9]。三种方法如下：

(1) 加密和鉴别：在这种方法中，加密和消息鉴别是互相独立计算的。也就是说，给定一个明文m，发送者发出$\langle c, t\rangle$。其中

$$c \leftarrow \mathsf{Enc}_{k_1}(m)，\text{而且} t \leftarrow \mathsf{Mac}_{k_2}(m)。$$

接收方对c进行解密得到消息m，然后验证标记t。如果$\mathsf{Vrfy}_{k_2}(m, t) = 1$，接收方输出$m$；否则输出$\perp$。

(2) 鉴别后加密：这里首先需要计算一个 MAC 标记t，然后消息和标记在一起加密。也就是说，接收方发送c，通过：

$$t \leftarrow \mathsf{Mac}_{k_2}(m)，\ c \leftarrow \mathsf{Enc}_{k_1}(m\|t)。$$

鉴别标记t并不是“以明文”发送，而是放在明文中一起加密的。接收方解密出c，然后验证对消息m的标记t。如前，如果$\mathsf{Vrfy}_{k_2}(m, t) = 1$，接收方输出$m$；否则输出$\perp$。

(3) 加密后鉴别：在这个情形中，消息m首先被加密，然后再在加密信息上计算出一个标记t。也就是说，消息是$\langle c, t\rangle$，其中

$$c \leftarrow \mathsf{Enc}_{k_1}(m)，\ t \leftarrow \mathsf{Mac}_{k_2}(c)。$$

接收方在解密c之前验证t，这就是前面一节构造 4.19 所做的。

本节分析以上每个方法，使用任意 CPA 安全加密方案和任意的安全 MAC（有唯一标记）。这里分析会按照“全都或者全都不”的方法：也就是说，只对既保证私密性又保证完整性的组合表示满意，这种组合使用任何 CPA 安全的加密方案和任何安全的消息鉴别码。换句话说，将拒绝任何即使是只存在单个反例（加密或者 MAC）的组合。例如，会拒绝“加密和鉴别”方法，因为它不安全。这并不是说对每个安全加密和 MAC 来说，这种“加密和鉴别”方法都不安全，只是说存在一个安全的加密方案和 MAC，其组合是不安全的。

使用“全都或者全都不”的方法是因为它可以减少实现中错误的可能性。这是由于有可

---

⑨ 一个常见的错误是使用相同的密钥来完成加密和鉴别。应该永远不要这样做，因为独立的密钥应该用在独立的应用中（除非当使用相同密钥时，特定的安全证明已经知道）。这一点的进一步讨论见本节末。

能替换任何安全加密方案为另一个方案而不影响应用的安全性（对 MAC 来说也一样）。这种替换在实践中很常见，当密码学函数库更新或者是修改标准的时候。

是仅仅私密性，还是私密性和消息完整性。大部分在线任务，以及任何在线购物或者银行交易，都同时需要加密和鉴别，但是通常不很明确何时在私密性之外还需要鉴别。例如，当加密磁盘上的文件时，是否也需要进行鉴别？初看上去可能会认为，既然磁盘加密是用来防止攻击者读取秘密文件，就没有必要进行鉴别。然而，财务报告等被修改后，敌手可能会造成很大的损失（比如，因此导致公司错误地发布假的报告）。最佳实践通常使用加密和鉴别；通常不单独仅仅使用加密，除非有非要这么做的理由（比如在严重资源受限的设备中实现该方法），并且，即便是这样，也只在确保对数据的修改不会引起损害的情况下使用。注意到，缺乏完整性有时会导致违背私密性，正如从选择密文攻击中已看到的。

安全性要求。为了分析哪种加密方法和鉴别方法的组合是安全的，必须首先定义什么是“安全的组合”。最好的方法是对什么是安全信道建立一个通用模型，并证明给定的组合满足这一定义。不幸的是，提供安全信道的正式定义超出了本书的范围，因此这里提供一个更“直白的”定义，它只是简单地包括私密性和完整性。定义和结果分析对理解手头的关键问题已经足够。

假设$\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$是一个任意加密方案，且令$\Pi_M = (\mathsf{Gen}_M, \mathsf{Enc}, \mathsf{Dec})$是一个消息鉴别码。一个消息传输方案$\Pi' = (\mathsf{Gen}', \mathsf{EncMac}', \mathsf{Dec}')$起源于$\Pi_E$和$\Pi_M$的组合，是一个算法元组，执行如下：

(1) 密钥生成算法Gen'输入参数为1^n，运行$\mathsf{Gen}_E(1^n)$和$\mathsf{Gen}_M(1^n)$来分别获得密钥k_1和k_2。密钥为(k_1, k_2)。

(2) 消息传输算法EncMac'，输入为密钥(k_1, k_2)以及消息m，输出为c，该值源于$\mathsf{Enc}_{k_1}(\cdot)$和$\mathsf{Mac}_{k_2}(\cdot)$的组合。

(3) 解密算法Dec'，输入为密钥(k_1, k_2)和传输值c，并应用$\mathsf{Dec}_{k_1}(\cdot)$和$\mathsf{Vrfy}_{k_2}(\cdot)$的某些组合。$\mathsf{Dec}'$的输出要么是明文$m$，要么是特殊符号$\perp$表示错误。

正确性要求是对每一个n，每一对由$\mathsf{Gen}'(1^n)$输出的密钥(k_1, k_2)，以及每一个$m \in \{0,1\}^*$都有

$$\mathsf{Dec}'_{k_1,k_2}(\mathsf{EncMac}'_{k_1,k_2}(m)) = m$$

Π'实际上满足加密方案的语法。可认为它是一个“消息传播方案”，当定义安全的时候要求在私密性上附加消息完整性。

如前所述，通过定义分离的私密性和鉴别的概念来定义一个消息传播方案Π'的安全性。私密性的定义如之前所述：Π'（现在视为一个加密方案）是 CCA 安全的。（它有“推动”私密性需求的效果，因为开始时加密方案Π_E仅仅是 CPA 安全的。）对消息完整性的定义对在选择密文攻击条件下的存在不可伪造性是十分必要的。既然Π'不满足消息鉴别码语法的要求，这里引入一个特殊的定义。下面定义一个消息传输方案Π'，敌手$\mathcal{A}$和安全参数为n的实验。

安全消息传输实验$\mathsf{Auth}_{\mathcal{A},\Pi'}(n)$如下：

(1) 运行$\mathsf{Gen}'(1^n)$产生随机密钥$k = (k_1, k_2)$。

(2) 敌手$\mathcal{A}$被给定输入参数1^n和预言机消息传输算法$\mathsf{EncMac}'_{k_1,k_2}(\cdot)$。敌手最终输出$c$。设$\mathcal{Q}$表示所有$\mathcal{A}$对其预言机的问询集合。

(3) 令 $m := \mathsf{Dec}'_k(c)$。当且仅当$m \neq \perp$且$m \notin \mathcal{Q}$时，实验的输出为 1。

定理 4.23 如果对所有概率多项式时间的敌手$\mathcal{A}$，存在一个可忽略的函数negl，满足

$$\Pr[\mathsf{Auth}_{\mathcal{A},\Pi'}(n) = 1] \leqslant \mathsf{negl}(n)$$

则消息传输方案Π'达到可鉴别的通信。

将上面的实验与定义 4.2 前的消息鉴别实验进行比较，可看到这里敌手的工作更容易，敌手不需要知道输出c对应的消息m。这意味着满足上面定义的构造方法（在某种程度上）比仅仅满足定义 4.2 的构造方法更加安全。

基于上面的描述，现在能给出安全消息传输方案。

定义 4.24 如果一个消息传输方案$(\mathsf{Gen}', \mathsf{EncMac}', \mathsf{Dec}')$既有 CCA 安全的加密方案，且达到了认证的通信，那么它就是安全的。

现在分析之前提到的三种加密和认证组合的方法。

加密和鉴别。正如已经所描述的，在这种方法中，加密和认证是分开计算的。给定一个信息m，传输值为$\langle c, t\rangle$，其中

$$c \leftarrow \mathsf{Enc}_{k_1}(m)，且\ t \leftarrow \mathsf{Mac}_{k_2}(m)。$$

这种组合并不（必须）是安全的，因为它可能违反私密性的要求。因为一个安全 MAC 并不必保证任何私密性，确切地说消息的标记可能会泄露整个消息的内容（特别地，如果$(\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$是一个安全消息鉴别码，那么它是由$\mathsf{Mac}'_K = (m, \mathsf{Mac}_k(m))$定义的方案。）因而，加密和鉴别的组合可能会产生一个方案，该方案甚至在窃听敌手存在的情况下不能达到不可区分的加密，这是最基本的私密安全级别了。

上面所举的例子看起来是一个不自然的反例，但是它足以说明问题，因为要求安全应该对构造方法的所有实现实例都得到保证。不难发现，即便是使用普通的 MAC，加密与鉴别的组合方式也不安全（不管加密方案是什么）。练习 4.19 要求说明如果 CBC-MAC 被当做Π_M使用，那么不能抵御选择明文攻击。

先鉴别后加密。在这里，首先计算一个 MAC 标记$t \leftarrow \mathsf{Mac}_{k_2}(m)$；然后对$m \| t$进行加密，结果$\mathsf{Enc}_{k_1}(m \| \mathsf{Mac}_{k_2}(m))$被传输。我们通过下面的加密方法来说明这种组合也不一定是安全的。

令$\mathsf{Transform}(m)$如下：消息m中的 0 被转化成 00，消息中的 1 被转化为 01 或者 10 中任意一个[10]。这个变换编码的逆过程是将消息划分成比特对，然后映射 00 到 0，01 到 10 到 1。如果遇到一个 11，那么结果就是$\perp$。（也就是说，$\mathsf{Transform}^{-1}(0110) = 11$，但是$\mathsf{Transform}^{-1}(0111) = \perp$。）

定义$\mathsf{Enc}_k(m) = \mathsf{Enc}'_k(\mathsf{Transform}(m))$，其中$\mathsf{Enc}'$表示使用伪随机函数的计数模式。（关键是$\mathsf{Enc}'$对每一个要加密消息产生一个新的伪随机流，然后将伪随机流和消息异或。）注意Enc

[10] 这个编码是人为杜撰的，但是，对输入的编码是常见的，如果密码学方案依赖于使用的编码方法，则显然是不理想的。

是 CPA 安全的。

下面说明使用鉴别后加密的方法，加密方案和任何 MAC 的组合都不是选择密文攻击安全的。只要敌手能够知道一个密文是否有效，即使不能获得全部密文的解密信息，描述的攻击也是可以工作的。因此，这个攻击很容易实施。敌手所要做的事情就是将密文传送给某一方，并观察其反应是否与收到无效密文时一致（例如，如果要求重传）。

考虑下面的选择密文攻击：给定一个挑战密文$c = \mathsf{Enc}'_{k_1}(\mathsf{Transform}(m\|Mac_{k_2}(m)))$，敌手简单地在$c$的第二个分块上翻转前两个比特（回想一下，$c$的第一个分块是初始计数值$ctr$），并验证结果密文是否是有效的。（这可以通过对加密预言机的询问确定。敌手只需要知道密文是否有效，但是，一个更弱的预言机也足够。）如果消息m的第一个比特是 1，那么对修改的密文就是有效的。这是因为如果m的第一位是 1，那么$\mathsf{Transform}(m)$的头两位是 01 或者 10。这两个比特的翻转会导致另一个对m有效的变换编码。（对Enc'的选择确保了对密文前两个比特的翻转导致被编码消息的前两个比特的翻转。）此外，标记仍然有效，因为它适用于m而不是对m的编码。另一方面，如果m的第一个比特为 0，那么对密文的改变就无效，因为$\mathsf{Transform}(m)$的前两位是 00，会转变成 11。

这个攻击能被对m的每个比特一个个使用，导致消息m的完整复原。

这个反例演示了鉴别后加密的组合并不是安全的。然而，有些使用该办法的特殊例子是安全的；一个例子是在 SSL 中使用的那个组合。此外，如同前面所提到的，安全性依赖于特殊的实现方法，这个方法是糟糕的。

先加密后认证。在该方法中，消息首先被加密，然后在密文上计算 MAC。也就是说，消息是一对$\langle c, t\rangle$，其中

$$c \leftarrow \mathsf{Enc}_{k_1}(m) \text{ 而且} t \leftarrow \mathsf{Mac}_{k_2}(c)。$$

加密方法与构造方法 4.19 中相同。有如下定理：

定理 4.25 令Π_E是一个 CPA 安全的对称密钥加密方案，令Π_M是一个具有唯一标记的安全消息鉴别码。那么组合$(\mathsf{Gen}', \mathsf{EncMac}', \mathsf{Dec}')$通过对$\Pi_E$以及$\Pi_M$使用先加密后认证的方法得到。它是一个安全的消息传输方法。

我们已经证明了上面的组合在定理 4.20 中是 CCA 安全的，并将证明它能提供可鉴别的通信留作练习。

安全消息传输与 CCA 安全的比较。尽管使用同样的构造方法达到 CCA 安全和安全消息传输，每种情况中的安全目标是不同的。在 CCA 安全的设置中，对获得消息鉴别不感兴趣；而是，希望即使是在面对一个能够发出解密问询的强大敌手时，也能保证私密性。当考虑安全消息传输时，与此相反，对双目标即 CCA 安全和消息的完整性感兴趣。很明显，正如所定义的，安全消息传输意味着 CCA 安全，而反过来则未必。

对独立密钥的需要。结论是要强调一个基本的安全加密原则：不同的安全目标通常需要使用不同的密钥。也就是说，如果一个加密方案和消息鉴别码同时需要，那么就需要使用独立的密钥。为了说明这一点，试想一下在“先加密后鉴别”方法中，当加密和鉴别使用同一个密钥k时会发生什么。设F是一个强伪随机置换，那么F和F^{-1}都是强伪随机置换。定义$\mathsf{Enc}_k(m) = F_K(m\|r)$，$m \in \{0,1\}^{n/2}$且随机数$r \leftarrow \{0,1\}^{n/2}$，定义$\mathsf{Mac}_k(c) = F_K^{-1}(c)$。可以

看到，给定的加密方法是 CPA 安全的（事实上甚至是 CCA 安全的），给定的消息鉴别码是一个安全 MAC。然而，先加密后鉴别的组合应用在消息m上，使用相同的密钥k会产生

$$\mathsf{Enc}_k(m), Mac_k(Enc_k(m)) = F_k(m\|r), F_k^{-1}(F_k(m\|r)) = F_k(m\|r), m\|r$$

那么消息m就被泄露了。

参考文献和扩展阅读材料介绍

消息鉴别码的安全定义来自 Bellare[12]对 Goldwasser 等[70]给出的数字签名安全性定义的改编（见 12 章）。将伪随机函数用于消息鉴别的范例（如构造方法 4.3）由 Goldreich 等[66]引入，构造方法 4.5 中固定长度的 MAC 扩展到可变长度的 MAC，是 Goldreich[65]提出的。另一种固定长度 MAC 扩展到可变长度 MAC 的可选择方法是使用抗碰撞散列函数，在 12.4 节中数字签名的情形中给出（就像所提到的，NMAC 也是使用同样的思想）。

CBC-MAC 在 20 世纪 80 年代早期被标准化[148, 80]，接下来由 Bellare 等进行了形式化分析和安全证明[12]（证明包括固定长度的情况和可变长度的情况）。NMAC 和 HMAC 由 Bellare 等[8]引入，在文献[110]中给出标准化。

抗碰撞散列函数的首次形式化的定义由 Damgard[43]提出，Merkle-Damgard 变换由 Damgard 和 Merkle 独立地提出[44, 102]。更多关于除了抗碰撞散列函数外的安全名称和概念见文献[99，123]。关于 SHA-1 和 MD5 的更多的信息可见 Kaufman 等的著作[87]。但是，请注意，这些论述先于最近 Wang 等[143, 144]发现的攻击。其他抗碰撞散列函数在计算机安全上的有趣的应用见 Kaufman 等的文献[87]（但是注意，有很多情况下给出的安全性讨论都是完全启发式的）。文献中出现了很多散列函数，有一些被攻破了，有些没有。最新信息在“散列函数长廊（Hash Function Lounge）”[7]中可以找到。

Dolev 等在文献[50]中论述了先加密后应用 MAC 达到 CCA 安全的办法。Bellare 和 Namprempre[13]，以及 Krawczyk[90]分析了同时达到保密和鉴别的不同方法，Krawczyk 也分析了在 SSL 中所使用的先鉴别后加密的方法。其他的包含完整性的安全加密概念在文献[85，13]中讨论。练习 4.5 的解答见文献[11]。

练　习

4.1 设$\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$，是一个安全的 MAC，对于$k \in \{0,1\}^n$，标记生成算法$\mathsf{Mac}_k$通常输出长度为$t(n)$的标记。证明$t$必须是超对数的，或者，如果$t(n) = \mathcal{O}(\log n)$，那么$\Pi$就不是一个安全的 MAC。

提示：考虑随机猜测一个有效标记的概率。

4.2 考虑下面一个消息长为$l(n) = 2n - 2$的定长 MAC，使用伪随机函数F，输入消息$m_0\|m_1$，其中$|m_0| = |m_1| = n-1$，且$k \in \{0,1\}^n$，算法 Mac 输出$t = F_k(0\|m_0)\|F_k(1\|m_1)$，算法Vrfy遵照一般的定义。那么(Gen, Mac, Vrfy)在选择消息攻击条件下是否是存在不可伪造的呢？证明你的答案。

4.3 令F是一个伪随机函数，证明下面消息长为$2n$的 MAC 是不安全的：共享的密钥k是一个随机$k \in \{0,1\}^n$，为了鉴别一个消息$m_1\|m_2$有$|m_1| = |m_2| = n$，计算标记$\langle F_k(m_1), F_k(F_k(m_2))\rangle$。

4.4 令F是一个伪随机函数。论证下面的每个消息鉴别码是不安全的（共享的密钥是随机数$k \in \{0,1\}^n$）。

①为了鉴别一个消息$m = m_1 \| ... \| m_l$，其中$m_i \in \{0,1\}^n$，计算$t := F_k(r) \oplus F_k(m_1) \oplus \cdots \oplus F_k(m_l)$。

②为了鉴别一个消息$m = m_1 \| ... \| m_l$，其中$m_i \in \{0,1\}^n$，随机选择$r \leftarrow \{0,1\}^n$，计算$t := F_k(r) \oplus F_k(m_1) \oplus \cdots \oplus F_k(m_l)$，并发送$(r,t)$。

③为了鉴别一个消息$m = m_1 \| ... \| m_l$，其中$m_i \in \{0,1\}^{n/2}$，随机选择$r \leftarrow \{0,1\}^n$，计算$t := F_k(r) \oplus F_k(\langle 1 \rangle || m_1) \oplus \cdots \oplus F_k(\langle l \rangle || m_l)$。（其中$\langle i \rangle$是一个对整数$i$的编码，长度为$n/2$比特），并发送$\langle r,t \rangle$。

4.5 当敌手同时拥有Mac和Vrfy随机预言机时，考虑安全消息鉴别码的扩展定义。

①为这种情况提供一个形式化的安全定义，并解释现实世界中敌手的什么行为由Vrfy随机预言机来建模。

②表明如果$\prod$有唯一的标记（见 4.8 节），那么如果它满足定义 4.2，那么它就满足你的定义。

③如果$\prod$没有唯一的标记，那么它可能满足定义 4.2，而无法满足你的定义。

4.6 构造方法 4.3 在使用弱伪随机函数（见练习 3.20）时是否安全？解释原因。

4.7 证明构造方法 4.5 在作出如下修改时是安全的：不将l包含在每一个分块中，设定$t_i := F_k(r \| b \| i \| m_i)$，其中$b$是单个比特，满足在所有分块中（除了最后一个模块）为$b = 0$。最后一个模块中为$b = 1$。该修改的优势是什么？（这里要求所有消息的长度必须为$n/2 - 1$的整数倍。）

4.8 说明基本的 CBC-MAC 构造方法用于鉴别不同长度的消息时是不安全的。

4.9 证明下面对 CBC-MAC 的修改不会产生安全的定长 MAC。

①修改 CBC-MAC，使得随机的IV用每次计算标记（IV与标记t_l一同输出）。也就是说，$t_0 \leftarrow \{0,1\}^n$是随机选择的，而不是固定为0^n，且标记为t_0, t_l。

②修改 CBC-MAC，这样所有的分块$t_0, \cdots, t_l$都被输出，而不是只输出t_l。

4.10 提供抗第二原像碰撞和抗原像碰撞的形式化定义。证明任何抗碰撞散列函数都是抗第二原像碰撞的，且任何抗第二原像碰撞的函数都是抗原像碰撞的。

4.11 令(Gen_1, H_1)和(Gen_2, H_2)是两个散列函数。定义(Gen, H)，这样Gen运行Gen_1和Gen_2，分别得到密钥s_1和s_2，定义$H^{s_1,s_2}(x) = H_1^{s_1}(x) \| H_2^{s_2}(x)$。

①证明如果至少有一个(Gen_1, H_1)和(Gen_2, H_2)是抗碰撞的，那么(Gen, H)是抗碰撞的。

②分别确定对抗第二原像碰撞和抗原像碰撞有类似的结论。在每种情况下证明你的回答。

4.12 令(Gen, H)是一个抗碰撞散列函数，那么由$\widehat{H}^s(x) \stackrel{\mathsf{def}}{=} H^s(H^s(x))$定义的$(\mathsf{Gen}, \widehat{H})$一定抗碰撞吗？

4.13 给出定理 4.14 的形式化证明（即描述形式化的规约）。

4.14 给出一般化的方法构造 Merkle-Damgard 变换，使用任意至少压缩 1 比特的压缩函数。你应该涉及通用输入长度l'和通用输出长度l（$l' > l$）。

4.15 对下面每一种对 Merkle-Damgard 变换的修改，确定结果是否是抗碰撞的。如果是，证明之；如果不是，则展示一种攻击。

①修改该构造方法使得不包括输入长度（即输出z_B，但是不输出$z_{B+1} = h^s(z_B \| L)$）。

②修改该构造方法使得不输出$z = h^s(z_B \| L)$，算法输出$z_B \| L$。

③替换掉IV，从x_1开始计算。就是说，定义$z_1 := x_1$，然后对$i = 2, \cdots, B+1$计算$z_i = h^s(z_{i-1} \| x_i)$，并输出$z_{B+1}$。

④不使用IV，设$z_0 := L$，然后对$i = 1, \cdots, B$，计算$z_i = h^s(z_{i-1} \| x_i)$，并输出$z_B$。

4.16 当依赖的函数的输入长度为l'和输出长度为l（有$l' > l$)时，给出 HMAC 的详细描叙。描述使用 SHA-1 的 HMAC 实例。

4.17 在 HMAC 发明之前，通常使用$\mathsf{Mac}_k(m) = H^s(k \| m)$来定义 MAC，其中$H$是一个抗碰撞散列函数。说明当$H$通过 Merkle-Damgard 变换构造时，不是一个安全的 MAC。

4.18 说明即使只使用构造方法 4.15 的情况下，构造方法 4.19 仍然是 CCA 安全的（这个 MAC 没有唯一标记)。

4.19 说明如果任何消息鉴别码有唯一的标记，在使用先加密后鉴别的方法，该组合的结果不是 CPA 安全的。

4.20 说明一个 CCA 安全的加密方案不是一个安全的消息传输方案。

4.21 说明一个可以鉴别通信的消息传输方案不是一个安全的消息传输方案。

4.22 证明定理 4.25。

第5章　伪随机置换（分组加密）的实际构建

前几章学习了伪随机置换如何用来构造安全的加密方案和消息鉴别码。但是，一个至关重要的问题还没有论及，那就是伪随机置换起初是怎样构造的，甚至它是否存在。下一章将从理论的角度研究这些问题，并表明当基于十分弱的假设时，伪随机置换的构造方法是可证明安全的。相比而言，本章的关注焦点是启发式的，但是效率高得多的伪随机置换的构造方法，称为分组密码，并在实践中使用。

本章将仔细探讨的分组密码的构造方法（大部分）是启发式的，至少它们没有基于任何更弱的假设的安全性证明。尽管如此，大量用于实践的分组密码仍经受住了多年的公开推敲和密码分析。鉴于此，假设这些分组密码确实为（强）伪随机置换是十分合理的。这一假设受到下面要讨论技术问题的影响。

当然，从某种意义上说，假设因子分解是困难的和假设 DES（一个随后在本章中仔细研究的分组密码）是一个伪随机置换，这两个假设之间没有本质的区别。然而，这些假设之间有一个明显的性质上的区别[①]。最基本的区别是前一个假设是较弱的。也就是说，要求某个问题（如因子分解问题）很难求解，和要求区分一个给定密钥的函数和一个随机函数，前者看上去“更容易满足”。另一个重要程度较低的但依旧相关的区别是：研究因子分解问题比研究区分 DES 和随机函数问题的时间要长得多，而且认为因子分解是一个困难的数学问题，比基于它的密码学方案的出现还要早。进一步强调：大多数的针对 DES 和其他分组密码的密码学分析主要集中在密钥恢复攻击，目的是在给定多对$(x, DES_k(x))$的情况下恢复密钥k；相对少的研究是针对（潜在的更容易的）问题，即从随机函数中区分 DES。（这也许对较新的分组密码而言正确程度较低，见后面的论述。）

总之，一个被充分研究过的分组密码（例如 DES）是伪随机的，这一假设的确是合理的，而且在实践中依赖这个假设会感觉很方便。不过，最好还是将现代分组密码的安全建立在以更弱和更长时间满足的假设的基础之上。第 6 章中将说明这（原则上）是可能的；不幸的是，其构造方法比本章用到的分组密码效率低几个数量级。

作为强伪随机置换的分组密码。分组密码是一个有效的加密置换$F: \{0,1\}^n \times \{0,1\}^l \to \{0,1\}^l$。回忆一下 3.6.3 节的内容，意味着函数$F_k$由$F_k(k) = F(k,x)$定义，是双射（即置换），此外，在给定$k$时，$F_k$和它的逆$F_k^{-1}$能有效地计算。选择$n$为密码长度，$l$为$F$分块的长度。不用按照第 3 章的定义，不再要求$n = l$（虽然通常是这样的）。一个更重要的差别是：这里$n$和$l$是固定的常量，而在第 3 章中，它们是一个安全参数n的函数。这将处于一个有具体数值的安全中，而不是一个渐进数值的安全中。下文将进一步讨论[②]。

尽管如此称呼，分组密码应该视为（强）伪随机置换而不是加密方案。换句话说，分组

① 必须清楚一点，本段的讨论是非正式的，如果我们不能首先证明分解问题是困难的，我们不能正式地声称任何内容。

② 虽然定长密钥的分组加密没有“安全参数”，仍然认为安全依赖密钥的长度，并用n表示这个值。需要强调：当比较拥有不同密钥长度的分组加密时，或者使用支持不同长度密钥的分组加密时，将密钥的长度视为参数是有意义的。

密码应被视为加密或其他方案的构造组成部分，而不是加密方案本身。就像在第 3 章中讨论的一样，将分组密码建模成（强）伪随机置换使得可以对基于分组密码的构造方案进行严格的安全证明，并使对分组密码的需求显现出来。此外，对分组加密希望达到的目标（如定义 3.28）要有充分的理解。分组密码应被塑造成伪随机置换这一观点，至少是在最近，被作为设计中的主要影响因素。例如，近期的高级加密标准（AES，将在本章后面提到）的征集提案陈述了如下评价标准：

> 由算法提供的安全是最重要的因素……算法将由以下因素来判断……
> - 算法的输出结果与对输入分块的随机置换不可区分的程度。

本质上，这表明了分组密码应该是一个伪随机置换。（提交的建议方案在什么程度上被评价为一个强伪随机置换是不清楚的。然而，如果一个被证明的攻击表明某个建议方案不满足这个标准，这个建议方案将不可能被采用。）因此，如前所述，现代分组密码被期望成为伪随机置换。这适用于本书已讨论过的所有依赖伪随机置换（或者是伪随机函数）的构造方法。

分组密码和定义 3.28 虽然将分组密码视为伪随机置换，一个技术困难是任何给定的分组密码往往定义成具有固定的密码长度和模块长度。于是，定义 3.28 中渐进安全的定义不适用。当考虑诸如 SHA-1 之类的实用的抗碰撞性散列函数时，出现了类似的问题（虽然在那里没有给出仔细的讨论）。在这两种情况下，处理此问题的恰当方法是：要理解只是期望对“在任何合理的时间之内”（也就是说，100 年的时间中使用可用的最强大的计算机。）运行的攻击者来说，分组密码和随机函数是难以区分的。这可在具体安全的框架中作形式化的论述，在 3.1.1 节简要地提到。

攻击分组密码。尽管事实是分组密码不应该和加密方案混淆，攻击分组密码F的（这里将采用的）标准术语参见：

(1) 唯密文攻击：攻击者仅仅被给予一系列不知道输入$\{x_i\}$的输出$\{F_k(x_i)\}$。

(2) 已知明文攻击：攻击者被给予输入和输出对$\{x_i, F_k(x_i)\}$。

(3) 选择明文攻击：攻击者被给予对一系列攻击者自己随机选择的输入$\{x_i\}$的$\{x_i, F_k(x_i)\}$。

(4) 选择密文攻击：攻击者被给予攻击者自己选择的$\{x_i\}$和$\{y_i\}$的$\{x_i, F_k(x_i)\}$以及$\{(F_k^{-1}(y_i), y_i\}$。

上述观点认为，伪随机置换在选择明文攻击条件下是安全的，而一个强伪随机置换需要能够抵御选择密文攻击。根据渐进的含义，攻击的运行时间为多项式时间。然而如前所述，分组密码往往有某些固定的分块长度和密码长度，因此渐进的方法就没有用了。相反，对于分组密码来说，它的目的是在任何合理的时间范围内要牢不可破。这一点在分组密码的上下文中严格地进行了解释，如果已知的最有效攻击具有的时间复杂性大致相当于蛮力搜寻密钥，则分组密码才被认为是“好的”。因此，如果一个具有密钥长度$n = 112$的分组密码可以在2^{56}时间内被破解（后面将给出一个这样的例子），那么这个分组密码（通常）被认为是不安全的，即使2^{56}仍然是一个相对很大的数字。注意在渐进设定中，复杂度为$2^{n/2}$的攻击是不可行的，因为这需要指数时间（因而即使这种攻击是可行的，分组密码可能仍然满足伪随机

置换的定义)。但是在非渐进近似的情形中，必须关心攻击的实际时间复杂度（而不是它的渐进近似行为)。此外，这种攻击的存在可能预示着密码设计中更多的根本缺陷。

本章的目的：

为了避免任何混淆的发生，本章的主要目的是说明在现代分组密码的构造中用到的一些设计原则，次重要的目的是向读者介绍流行的分组密码 DES 和 AES。提醒读者注意：

(1) 本章目的不是说明 DES 或 AES 的底层细节，而且对这些分组密码的描述不依赖于具体实现方法。这里对这些分组密码的描述通常（有意的）不太精确，当某些细节与本章想强调的要点不相关时，将忽略这些技术细节。

(2) 本章的目的不是讲授如何构造安全的分组密码。相反，我们坚信：既然非常多的优秀分组密码很容易得到，那么新的（且专属的）分组密码，既不应该被构造出来，也不应该投入使用。

建议那些对构造分组密码的专业技术发展有兴趣的人们从本章后的参考文献开始入手。

5.1 代替—置换网络

分组密码所要求的主要特征是它的行为应该像一个随机置换。当然，一个真正的随机置换将是完美的。拥有分块长度（即输入和输出长度）为n比特的随机置换将需要$log(2^n!) \approx n \cdot 2^n$比特来表示，当$n > 20$时，这是不切实际的，当$n > 50$时，这完全不可行。(将来，实践中$n \approx 64$的分块长度在某些情况下已经太小，因此现代分组密码具有 $n \geqslant 128$ 的分块长度。）因此，需要以某种方式构造一个简明函数，该函数表现得像随机函数一样。

混淆—扩散范例。除了对完美加密的研究，香农（Shannon）还介绍了一个基本范例，用来构造简明的看起来随机的置换。基本的思想是从许多较小的随机的（或看上去随机的）置换$\{f_i\}$（拥有短分块长度），来构造拥有长分块长度的看起来随机的置换F。下面讨论其在最基本的层面上是如何工作的。假设想让F拥有 128 比特的分块长度。定义F：F的密钥k将表示 16 个随机置换$f_1, \cdots, f_{16}$，每个都拥有 8 比特的长度[③]。给定输入$x \in \{0,1\}^{128}$，将其解析为 16 个连续的 8 比特的分块，$x_1, \cdots, x_{16}$，然后设定

$$F_k(x) = f_1(x_1) \cdots f_{16}(x_{16}) \tag{5.1}$$

可以（非正式地）说这些$\{f_i\}$给F带来了混淆。

然而，很明显，如上定义的F并不是伪随机的。具体来说，如果x和x'仅仅第一个比特不同，那么$F_k(x)$和$F_k(x')$仅仅第一个字节不同（不用考虑k值)。相反，如果F是一个真正的随机置换，那么改变输入的第一位比特将会影响输出的所有字节。

基于这个原因，两个额外的修改被引入。首先，“分散”步骤被引入，在这一步骤里，输出的比特是一种置换[④]或者“混合”。其次，混淆/分散步骤，称为一轮（round)，被重复多次。例如，一个两轮分组密码可以操作如下：首先，$x' := F_k(x)$会在式（5.1）中计算，然后x'的比特值会被重新排列次序得到输出值x_2。这里强调用于每一轮操作的函数$\{f_i\}$以及随机置

③ 既然对 8 比特的随机置换能使用$\approx 8 \cdot 2^8$比特表示，则F密钥的长度约为$16 \cdot 8 \cdot 2^8$比特，或者 32K 比特。这比针对 128 比特的随机置换需要的比特数（$\approx 128 \cdot 2^{128}$比特）小得多。

④ 在这个情形中，“置换”表示重新排列比特位的次序。

换不必是相同的。虽然也可引入对密钥的依赖关系，但通常假设用于每一轮的混合置换是固定的（即密钥独立的）。

重复使用混淆与分散确保输入中任何微小的改变都将混合以及传递到所有的输出比特。其效果是输入上的细小变化将会对输出有显著的影响，正如随机置换中所期望的那样。

代替置换（SP）网络。代替置换网络可以视为对混淆分散范例的直接实现。这里主要的区别是每一轮中使用的函数$\{f_i\}$视为固定的（而不是依赖于密钥k），而且密钥用于不同的目的（将会简短地解释这些目的）。现在将$\{f_i\}$看作是 S 盒，因为它们充当固定的“替代函数”（会继续要求它们也作为置换）。

代替置换网络本质上遵循前面所概述的混淆分散范例的步骤。然而，因为 S 盒不再依赖于密钥，需要用其他方法引入依赖（根据 Kerckhoffs 原则，假设 S 盒和混合置换的确切结构是公开的，唯一机密的是密钥）。有很多方法，这里重点说明其中一种，即简单地异或带密钥的函数和中间结果，并且中间结果作为 SP 网络中每一轮的输入。分组密码的密钥有时被看作是主密钥，而每一轮中与中间结果进行异或运算的子密钥是根据密钥编排（Key Schedule），从主密钥衍生而来的。虽然可以定义更复杂的编排方式，但通常密钥编排很简单，而且只需选取密钥比特的子集（例如，一个双轮 SP 网络会在第一轮中使用主密钥的前半部分，在第二轮中使用主密钥的后半部分）。图 5.1 所示是代替置换网络的全局架构，图 5.2 所示是对此类网络中单轮的更细致描绘。

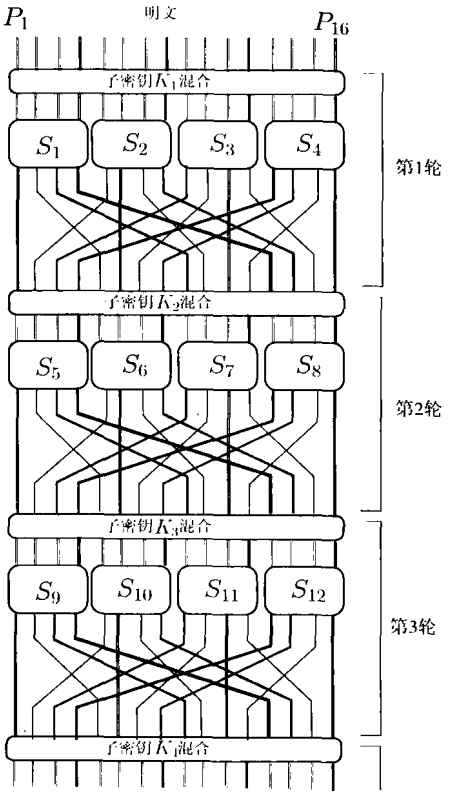


图 5.1　代替置换网络

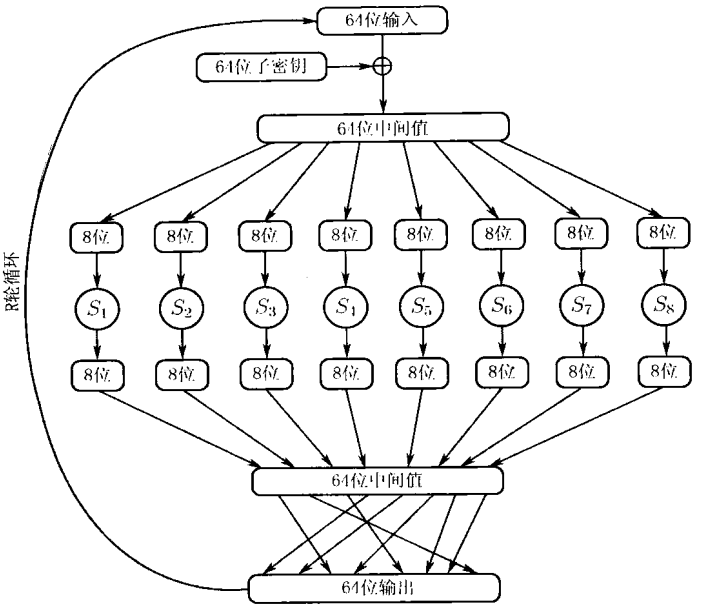


图 5.2　替代置换网络的单轮构造

S 盒，混合置换和密钥编排的选择从根本上决定了一个给定的分组密码是容易被攻破还是高度安全的。这里在一个较粗略的水平上讨论设计背后的基本原则。第一个原则仅仅是功能需求，而第二个则与安全更加相关。

设计原则 1——S 盒的可逆性。在代替置换网络中，S 盒必须是可逆的；也就是说，必须是单射和满射函数。否则分组函数将不能成为一个置换。为了理解使 S 盒成为单射和满射是充分条件，这里说明当该充分条件满足时，由输出和密钥可以完全确定输入。具体而言，需要说明每一轮循环是可逆的（意味着通过从最后运行到最前，整个网络都是可逆的）。一轮操作由三个阶段组成：前一轮输出与子密钥进行异或，把结果传递给 S 盒（如式（5.1）所示），最后使用混合置换重新整理这个结果中的比特。混合置换很容易被逆转，因为它仅仅是比特的一个重新排序。如果 S 盒是单射和满射，那么也同样可以逆转。然后结果可以与相应的子密钥来异或，从而获得原始输入。因此有：

命题 5.1　假设F是一个由替代置换网络定义的有密钥函数，该网络中所有 S 盒都是单射和满射。不管密钥编排方法以及轮的次数，对任意选择的k，F_k都是置换。

设计原则 2——雪崩效应。任何分组密码的一个重要特性就是输入的微小改变必然会导致输出的巨大改变。否则，两个相似的输入所生成的分组密码的输出看起来就不是独立的了（然而，在随机置换中，任何不等的输入所形成的输出都是独立分布的）。为了确保这样，分组密码被设计成要展现出雪崩效应，意味着输入的单个比特的变化会影响输出的每一个比特。（这并不意味着改变输入的一个比特就会改变输出的每一个比特，仅仅是能影响输出的所有比特。注意，即使是完全随机的函数，平均而言，改变输入的一个比特也只能期望改变输出一半的比特数。）

很容易证明在代替置换网络中起作用的雪崩效应提供了以下两个特性（以及使用充分多

的轮数）：

(1) S 盒的设计要使得改变 S 盒输入的单个比特就能改变 S 盒输出的至少两个比特。

(2) 混合置换的设计要使得任何给定的 S 盒的输出比特都被传递到下一轮中不同的 S 盒中。

为了理解这是怎样产生雪崩效应的，假设 S 盒都是那种只要改变输入的一个比特，就会导致输出的恰好两个比特的变化，而且混合置换也满足这个要求。具体而言，假设 S 盒有 4 个比特的输入/输出大小，并且假设分组密码的分块长度为 128 比特。现在考虑当分组密码的两个输入只有一个比特不同时，将会发生什么：

(1) 第一轮操作之后，中间值恰有两个比特不同。这是因为和当前的子密钥异或，会保持中间值一个比特的差别，所以除一个以外，所有 S 盒的输入都是相同的。在输入值不同的 S 盒中，S 盒的输出导致一个 2 比特的差别。结果中用混合置换改变了这些差别的位置，但是仍会保持 2 比特的差别。

(2) 之前所提到的第二个特性，在第一轮操作最后用到的混合置换将中间结果不同的两个比特位置扩展到了第二轮操作中的两个不同的 S 盒中。（即使子密钥和前一轮的结果异或后，仍然是这样。）因此，在第二轮操作中，有两个 S 盒接收有单个比特不同的输入。与之前一样，于是第二轮的最后中间值有 4 比特的不同。

(3) 继续同样的论证，可以预见 8 比特的中间值在第三轮操作之后受到影响，16 比特的中间值在第四轮之后受到影响，而在第 7 轮最后输出的 128 比特全部受到影响。

最后一点不是十分精确，并且可能（取决于确切的输入，以及 S 盒和混合置换的具体选择）在某轮的最后会有少于所期望的改变。由于这个原因，往往使用多于 7 轮。这里的关键之处在于它给出了轮数的下限：如果少于 7 轮，那么必然有某些输出的比特没有受到影响，意味着可能会将分组密码与随机置换区别开来。

有人可能会期望“最好的”设计 S 盒的方法是随机地选择它们（满足是单射和满射这一限制条件）。有趣的是，其实并不是这样[5]。例如，考虑这样一种情况，运行 4 比特输入的 S 盒，并令x和x'为两个不同的输入值。令$y = S(x)$，现在考虑随机选择$y \neq y'$作为$S(x')$的值。则有 4 个字符串与y相比仅有 1 个比特差别，因此以概率$4/15(> 1/4)$选择y'，它与y没有在两个或更多的比特上有差别。当考虑所有输入时，并且当需要多个 S 盒时则会变得更糟。基于此例，得到的结论是：作为一个通用的规则，最好是小心地设计拥有某种理想的属性（除了上面给出的一些之外）的 S 盒，而不是盲目地随机选择。

除了上述之外，我们同样注意到随机选择的 S 盒不是防御后面 5.6 节中攻击的最好办法。

代替置换网络的安全性。根据经验以及多年密码分析的努力，说明只要在选择 S 盒、混合置换和密钥编排的时候很谨慎，那么代替置换网络就是构造伪随机置换的一个良好选择。在 5.5 节中描述的 AES，结构上类似于上面所提到的代替置换网络，并且普遍被认为是很强的伪随机置换。

---

⑤ 这一节与前面对混淆扩散范例的讨论不同。轮函数（S 盒）依赖密钥而且因此对敌手保密。这里，函数是固定的，且是公开的，问题是找到什么函数是最佳的。

然而，很重要的是：此种方法构造的分组密码的长度非常依赖于轮的次数。为了更加深刻地理解代替置换网络，这里演示在轮数较少的分组密码类型上的攻击。这些攻击是明显的，但是值得提醒，因为它们结论性地证实了为什么轮数要多。

对轮数减少的代替置换网络的攻击。根据对伪随机置换的定义（见定义 3.23），敌手会被给予一个随机预言机，它要么是随机置换，要么是给定的分组密码（伴随随机选择的密钥）。敌手想要确定是哪种情况。很明显，如果敌手可以获得分组密码的秘密密钥，那么它就可以将其和随机置换区别开来。这样的攻击被称为完全攻破，因为一旦知道秘密密钥，安全便荡然无存。

对单轮代替置换网络的攻击：令F为一个单轮代替置换网络。下面证明一个攻击，在此攻击中，针对随机选择的输入x，敌手得到一个输入/输出对(x,y)，很容易知道作为$y=F_k(x)$的秘密密钥k。敌手从输出值y开始，然后逆转混合置换和 S 盒。这样做是因为混合置换和 S 盒的说明是公开的。敌手从这些逆转中计算出的中间值是$x\oplus k$（不失一般性，假设主密钥在只有一轮的网络中被用作子密钥）。由于敌手还拥有输入x，它将可以立即获取秘密密钥k，因此完全破解。

对两轮代替置换网络的攻击：再次展示了一个恢复秘密密钥的攻击，虽然这个攻击需要更多的时间。考虑下面的具体参数，令分组密码的分块长度为 64 比特，并且令每一个S盒拥有 4 比特的输入/输出长度。此外，让密钥k的长度为 128 比特，第一轮中使用密钥的前半部分$k^a\in\{0,1\}^{64}$，在第二个轮中使用密钥的后半部分$k^b\in\{0,1\}^{64}$。这里使用独立的密钥来简化对下面攻击的描述，这一处理只会使该攻击变得更加困难。

假设给了敌手密码的一个输入值x和输出$y=F_k(x)$。敌手从“反向工作”开始，逆转分组密码的第二轮中的混合置换和 S 盒（就像之前的攻击一样）。用ω_1来表示前 4 比特的结果。用α_1表示第一轮输出的前 4 个比特，有$\omega_1=\alpha_1\oplus k_1^b$，这里$k_1^b$表示$k^b$的前 4 比特（敌手不知道$\alpha_1$或$\omega_1$）。这里要重点观察的是：当“前向工作”是从输入$x$开始时，$\alpha_1$的值会至多被 4 个不同的 S 盒影响（因为，在最坏情况下，α_1的每一比特都来自第一轮中不同的 S 盒）。此外，因为已知第一轮中的混合置换，所以敌手确切知道 S 盒中的哪一个影响了α_1。这反过来意味着密钥k^a最多有 16 比特（在已知的位置）影响这 4 个 S 盒的计算结果。随后敌手就能猜测出相应的k^a的 16 比特和k_1^b的 4 比特值，然后使用已知的输入/输出对(x,y)来验证这个猜测的正确性。这个验证通过如下步骤完成：通过异或k^a的相关 16 比特和输入x的相关 16 比特，计算出结果α_1，然后对比ω_1和$\alpha_1\oplus k_1^b$（此处k_1^b也是猜测的一部分）。如果不相等，那么猜测不正确；如果相等，那么猜测可能是正确的。如此继续，敌手就可以毫无遗漏地找到这个密钥的所有 20 比特的值，而这个值和已给的(x,y)相符。尝试每一种可能性要用2^{20}次。

如果简化假设，一个不正确的猜测（即和密钥k的比特不相符的猜测）产生了$\alpha_1\oplus k_1^b$的随机值，那么一个不正确的猜测能通过上面的验证的概率为$1/2^{|\omega_1|}=1/16$。这意味着可粗略地认为有$2^{20}/16=2^{16}$的可能性通过这个测试（包括正确的可能性）。如果使用另一个输入/输出对(x',y')来重复上述的步骤，可期望消除约 15/16 的错误概率，然后看到如果使用许多不同的输入/输出对来重复此举，可期望一次猜测密钥的 20 比特，而这个猜测是和所有已给的输入/输出对相符的。于是这个猜测是正确的。

具体而言，假设 8 个输入/输出对被用来将可能性缩减到一种，然后敌手在$8\cdot 2^{20}=2^{23}$的次数内知道了k_1^b的 4 个比特。可以重复用在k^b的所有 16 个部分中，导致知道k^b的 64 比特的攻击拥有的总复杂度为$16\cdot 2^{23}=2^{27}$。沿着这个方法，敌手同样会知道k^a的所有 64 比特。这

实际高估了攻击的时间复杂度，因为k^a的某些部分会再次被利用，而且先前已经决定的k^a的部分不需重新猜测。在任何情况下，拥有时间复杂度为2^{27}的攻击都是实际所能达到的，远小于复杂度2^{128}，复杂度2^{128}是穷举搜索 128 比特的密钥空间所需要的。

以上说明很重要的一点，这个攻击是可能的，因为密钥的不同部分可与其他部分隔离开来（比起单个攻击时间 $2^{16\times4}=2^{64}$，16 次2^{23}攻击会快得多，这里，2^{23}为执行每次揭示k^b的 4 个比特的时间复杂度）。因此，需要进一步的分散来确保密钥的所有比特影响输出的所有比特。两轮操作不足以使其发生。

攻击三轮代替—置换网络。介绍一个较弱的攻击；不用知道密钥，只需表明将攻击三轮代替置换网络和伪随机置换区别开来是很容易的。这个攻击是基于前面提到过的观察，在三轮操作之后雪崩效应还没有完成（当然，这取决于分组密码的分块长度和 S 盒的输入/输出长度，但是如果是合理的参数就会是这样）。因此，敌手只需要计算两个有单比特差异的字符串。三轮分组密码会有这样的特性：输出的许多比特是相同的，使得很容易和随机置换区分开。

5.2 Feistel 网络

Feistel 网络是构造分组密码的一个替代方法。低级的构造模块（S 盒，混合置换和密钥编排）是一样的，区别是在高级设计中。Feistel 网络优于代替置换网络的优势在于 Feistel 网络剔除了 S 盒必须可逆这一要求。这很重要，因为一个好的分组密码应该拥有“非结构化的”行为（以便于它看起来是随机的）；然而，要求构造的所有部件都是可逆的，这本身就引入了结构性。因此，Feistel 网络是从非可逆组成部分构造出可逆函数的一种方法。这似乎是矛盾的术语，因为如果你不能逆转组成部分，似乎就不能逆转整个结构，但是 Feistel 网络设计巧妙地实现了这一点。

Feistel 网络，就像在代替置换网络中的情况一样，运行多轮操作。在每一轮操作中使用特定的轮函数，描述如下。在 Feistel 网络中，轮函数不需要是可逆的。轮函数往往含有像 S 盒和混合置换之类的组成部件，但是 Feistel 网络能处理任何轮函数，不用考虑轮函数的具体设计。当通过 S 盒构造轮函数时，设计者有更多自由，因为这里 S 盒不需要是可逆的。

Feistel 网络的第i轮是这样运行的。该轮的输入被分成两半，分别由L_{i-1}和R_{i-1}表示（分别由L_{i-1}表示“左半部分”，R_{i-1}表示“右半部分”）。如果分组密码的分块长度是n比特，那么L_{i-1}和R_{i-1}分别有$n/2$的长度，而第i个循环函数f_i会有$n/2$比特长的输入，并且产生$n/2$比特长的输出。（再次强调虽然f_i的输入和输出长度是一样的，它也没有必要是单射和满射。）此轮的输出(L_i, R_i)，这里L_i和R_i再次分别表示左半部分和右半部分，是由下式给出：

$$L_i := R_{i-1}\text{和}R_i := L_{i-1} \bigoplus f_i(R_{i-1}) \tag{5.2}$$

在t轮 Feistel 网络中，网络的n比特输入被解析为(L_0, R_0)，而输出是在应用了t轮操作之后得到的n比特大小的值(L_t, R_t)。如图 5.3 所示，展示了 4 轮 Feistel 网络。

下面讨论对密钥的依赖是怎么被引入的。就像在代替置换网络中的情况一样，主密钥k被用来推导在每一个轮操作中都要用到的子密钥；第i轮函数f_i依赖于第i个子密钥，用k_i表示。正式地，Feistel 网络的设计确定了一个与每轮操作i相关联的公开的“变形函数”$\widehat{f}_i$。这个函数$\widehat{f}_i$将一个子密钥和$n/2$比特的字符串作为输入，并且输出一个$n/2$比特的字符串。当主密钥是确定的，因此每一个子密钥k_i也确定时，第i轮函数f_i通过$f_i(R) \stackrel{\text{def}}{=\!=} \widehat{f}_i(k_i, R)$来定义。

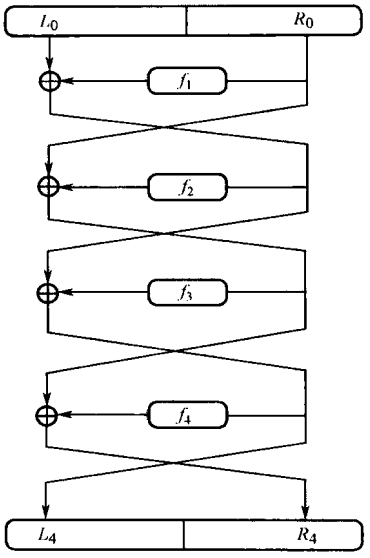


图 5.3　四轮 Feistel 网络

逆转 Feistel 网络。无论轮函数$\{f_i\}$是如何定义的，Feistel 网络都是可逆的（从而不用考虑变形函数$\{\widehat{f}_i\}$是什么）。为了证实这一点，只需表明任何给定轮数的 Feistel 网络都是可逆的。给定第i轮的输出(L_i, R_i)，可以按照如下办法计算出(L_{i-1}, R_{i-1})：首先设$R_{i-1} := L_i$，然后计算：

$$L_{i-1} := R_i \oplus f_i(R_{i-1})$$

（如果知道了主密钥，就可以从$\widehat{f}_i$推导出函数f_i。）可以验证出结果是正确的，(L_{i-1}, R_{i-1})正是这一轮的输入（即它计算出式（5.2）的逆）。注意f_i只在正向中计算，就像需要的那样。

因此有：

命题 5.2　令 F 为由 Feistel 网络定义的加密函数。不考虑变形函数$\{\widehat{f}_i\}$和轮数，对任意选择的k，F_k是一个置换。

同代替置换网络中的情况一样，当轮数太低时，对 Feistel 网络的攻击是可能的。在下一节讨论 DES 时，会看到此类的攻击。在 6.6 节会讨论关于 Feistel 网络的安全性的理论结果。

5.3　DES——数据加密标准

数据加密标准，或者说是 DES，是 20 世纪 70 年代由 IBM 开发的（在美国国家安全局的帮助下），并作为美国联邦信息处理标准（FIPS）被采用。在它的基本形式中，DES 已不再被认为是安全的，因为它的密钥长度较短，只有 56 比特。尽管如此，它的加强形式即 3DES 仍然在今天广泛使用（如 5.4 节所描述的）。

DES 有重要的历史意义，并且承受住了密码学家们的深入推敲，这种推敲可能比其他任何密码算法都要多。普遍一致的看法是，与其密钥长度有关，DES 是极其安全的。的确，即使经过多年，已知的实践中对 DES 的最好攻击是对所有2^{56}个可能密钥的蛮力搜索。后面将会说明，重要的对 DES 的理论攻击，需要比蛮力攻击更少的计算量；但是，这些攻击假设了某

些似乎不可能在实践中满足的条件。

本节提供 DES 的主要组成部分的一个高层次概述。这里不会提供每一个细节的完整说明，设计的某些部分在描述时被忽略。因为目的是介绍构造 DES 方案的基本思想，而不是所有的低层细节；对此类细节有兴趣的读者可以查阅本章后的参考文献。

5.3.1 DES 的设计

DES 分组密码是一个拥有 16 轮的 Feistel 网络，且分块长度为 64 比特，密钥长度为 56 比特。回顾在 Feistel 网络中，每一轮中使用的内部f函数一次只操作半个分块。因此，DES 轮函数的输入/输出长度就是 32 比特。在 DES 的 16 轮中每轮所用到的轮函数都是由同一个变形函数$\widehat{f}_i = \widehat{f}$推导而来的。DES 的密码编排被用来从 56 比特的主密钥k中获得每一轮的 48 比特的子密钥k_i。如在前一节所讨论的，第i个轮函数f_i就被定义为$f_i(R) \stackrel{\text{def}}{=\!=} \widehat{f}(k_i, R)$。正如从 DES 使用 Feistel 结构这一事实所期望的，轮函数是不可逆的。

DES 的密钥编排相对比较简单，每一个子密钥k_i都是由主密钥估算出的 48 比特的子集。这里不精确地描述密钥编排。主密钥的 56 比特被分成两部分，即“左半部分”和“右半部分”，其中每一部分含有 28 比特（事实上，这种分离发生在密钥的初始置换之后，但是这里的描述中忽略这些）。在每一轮中，子密钥最左边的 24 比特被当作是主密钥左半部分 28 比特的某个子集，而子密钥最右边的 24 比特被当作主密钥右半部分 28 比特的某个子集。整个密钥编排（包括比特被分为左半部分和右半部分，以及哪些比特被用来形成子密钥k_i）都是确定和公开的，唯一机密的是主密钥本身。

DES 变形函数$\widehat{f}$。前面曾经提到：变形函数$\widehat{f}$和第i个子密钥k_i共同决定了第i轮函数f_i。DES 中的变形函数是根据之前分析的范例构造出来的：它（本质上）只是一个单轮代替置换网络。更详细地说，使用$K_i \in \{0,1\}^{48}$和$R \in \{0,1\}^{32}$计算$\widehat{f}(k_i, R)$，方法如下：首先R被扩展成一个 48 比特的值R'。这是通过复制R一半的比特来完成的；用$R' := E(R)$来表示，这里E代表扩展函数。然后，完全按照之前讨论的代替置换网络中的方法进行计算：展开值R'和k_i进行异或运算，结果被分解为 8 块，每一块长度为 6 比特。每一个分块都经过（不同的）S 盒，S 盒的输入为 6 比特且输出为 4 比特；把 8 个 S 盒的输出连接起来就得到了一个 32 比特的结果。最后一步，对这个结果使用混合置换，从而获得$\widehat{f}$的最终输出。构造过程图如图 5.4 所示。

与最初讨论的代替置换网络相比，这里的一个不同是上面所提到的 S 盒是不可逆的；实际上，它们不可能是可逆的，因为它们的输入比输出长。更多关于 S 盒的结构细节在后面给出。

再次强调上面所描述的一切（包括 S 盒本身以及混合置换）都是公开的。唯一保密的是用来获得所有子密钥的主密钥。

S 盒。构成$\widehat{f}$“核心”的 8 个 S 盒是 DES 构造方法中重要的组成部分，而且是精心设计的（据报道是在国家安全局的帮助下）。对 DES 的研究表明：如果对 S 盒引入一些小变化，或者 S 盒是随机选择的，DES 将会更容易受到攻击。这应该作为对所有想设计分组密码的人一个警告：看上去随意的选择实际上完全不是随意的，如果做得不正确，则会使整个结构不安全。

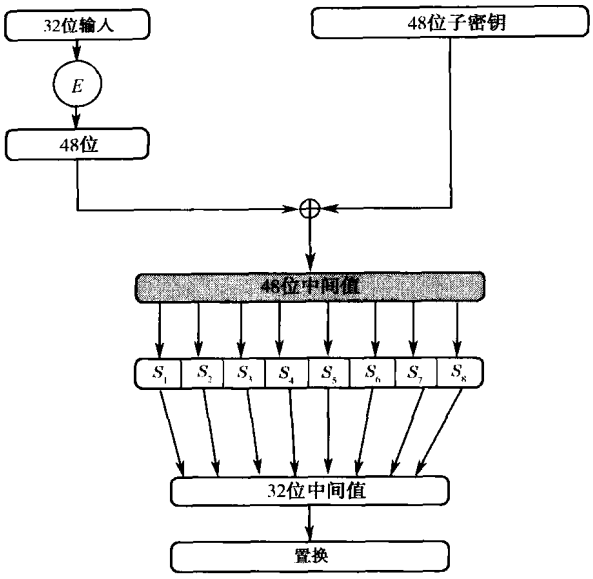


图 5.4 DES 变形函数

回想一下，每一个 S 盒都将 6 比特的字符串映射到 4 比特的字符串。每一个 S 盒都可以被看作是一个 4 行 16 列的表格，每一个单元格都含有一个 4 比特的记录。一个 6 比特的输入可被看作是搜索表中$2^6 = 64$个单元格的索引：输入的第一个和最后一个比特用来选择表中行数，而 2 到 5 比特被用来选择表中列数。找到的单元格中的 4 比特记录表示输出值。

由于其重要性，将描述 DES 中 S 盒的基本属性：

(1) 每一个 S 盒都是一个 4 对 1 的函数。（也就是说，有且只有 4 个输入被映射到每一个可能的输出上。）这是下面的属性所产生的。

(2) 表格的每一行恰好包含 16 种可能的 4 比特字符串中的一个。（就是说，每一行是 16 种可能的 4 比特字符串的一个置换。）

(3) 改变输入的 1 比特通常改变至少 2 个输出比特。

以上的属性将用于分析后面轮数减少的 DES。

DES 雪崩效应。如前所述，雪崩效应是任何安全分组密码的一个重要属性。上面描述的 DES 的 S 盒的第三个属性，和在变形函数中使用的混合置换，确保了 DES 展现出很强的雪崩效应。为了理解这一点，（当 DES 计算两个单比特不同的输入时）将追溯中间值的差别。用(L_0, R_0)和(L'_0, R'_0)来表示这两个输入，这里假设$R_0 = R'_0$，且因而唯一的一个不同的比特出现在输入的左半部分，参见式（5.2）。第一轮之后中间值(L_1, R_1)和(L'_1, R'_1)仍然只有 1 比特的不同，现在这种不同是在右半部分。在 DES 的第二轮中，输入的右半部分通过$\widehat{f}$运行。假设R_1和R'_1不同的那一个比特在展开步骤中没有被复制，在使用 S 盒之前中间值仍然是只有 1 比特不同。由第三个属性，在 S 盒计算之后，中间值至少有两比特不同。结果，中间值(L_2, R_2)和(L'_2, R'_2)有 3 个比特不同：在L_2和L'_2之间有 1 个比特不同（延续了R_1和R'_1之间的不同）以及在R_2和R'_2之间有 2 个比特不同。

混合置换将R_2和R'_2之间的 2 比特的不同扩展到这些字符串的不同区域。效果是，在下一轮中，两个不同的比特值都被输入到不同的 S 盒，结果是在中间值的右半部分有 4 个比特不同。（在左半部分现在也有 2 比特的不同。）与替代置换网络一样，有一个指数效应，因而经

过了 7 轮之后期望右半部分的所有 32 比特都受到影响（而且 8 轮之后左半部分的所有 32 个比特也同样受到影响）。

DES 有 16 轮操作，所以雪崩效应在计算中早早就结束了。这确保了 DES 对相似的输入生成完全不同和看起来独立的输出。DES 中的雪崩效应同样是由于对混合置换的慎重选择，事实上前面曾经表明，随机混合置换产生非常弱的雪崩效应。

5.3.2 对减少轮数的 DES 变种的攻击

理解更多关于 DES 构造方法和安全性的一个有用的做法是：观察只有轮数较少的 DES 的行为。这里将展现对 1，2 或 3 轮操作的 DES 的变种的攻击（回顾一下，真正的 DES 是有 16 轮操作的）。很明显拥有 3 个或更少轮数的 DES 不可能是一个伪随机置换，因为在仅仅 3 轮操作之后，雪崩效应还没有完成。因此，将有兴趣演示更难的（并且更有破坏性的）密钥恢复攻击，这个攻击只需使用相对少量的输入/输出对来计算密钥k。一些攻击和在代替置换网络的情形中看到的类似；然而，这里将看到它们是如何攻击具体的分组密码，而不是抽象的设计。

下述所有攻击都是已知明文攻击，敌手拥有明文/暗文对$\{(x_i, y_i)\}$，这里$y_i = DES_k(x_i)$，密钥为k。当描述这些攻击时，关注一个特定的输入/输出对(x, y)，并且描述敌手可以从这一输入/输出对上推出的密钥相关信息。继续使用之前的标记方法，分别用L_0和R_0表示输入x的左半部分和右半部分，用L_i, R_i表示第i轮之后的左半部分和右半部分。继续用E表示 DES 的扩展函数，用f_i表示在第i轮中用到的函数，并且用k_i表示在第i轮中用到的子密钥。

单轮 DES。在单轮 DES 中，有$y = (L_1, R_1)$，这里$L_1 = R_0$且$R_1 = L_0 \oplus f_1(R_0)$。因此可知$f_1$的一个输入/输出对；特别是可知道$f_1(R_0) = R_1 \oplus L_0$（注意，所有这些值都是已知的）。通过对输出结果$R_1 \oplus L_0$使用混合置换的逆操作，获得含有所有来自 S 盒的输出的中间值，这里前 4 比特是来自第一个 S 盒的输出，接下来的 4 比特是来自第二个 S 盒的输出，以此类推。这就意味着有了来自每一个 S 盒的输出值。

考虑（已知的）第一个 S 盒的 4 比特输出。回想每一个 S 盒都是一个 4 对 1 函数，这就意味着对 S 盒而言，的确有 4 种可能的输入值，会产生出给定的输出，所有其他的 S 盒也是类似的；每一个此类的输入都是 6 比特长。S 盒的输入仅是对本轮中用到的密钥k_1与$E(R_0)$的异或。（实际上，对于单轮 DES 来说，k_1是唯一的密钥。）因为R_0是已知的，于是结论是：对于k_1的每个 6 比特的部分都有四种可能值（而且可以计算它们）。这就意味着可能的密钥k_1的数量从2^{48}减到$2^{48/6} = 4^8 = 2^{16}$（因为k_1的 8 个 6 比特部分中的每一个都有 4 种可能）。这已经是一个小数目了，所以只要对每一个不同的输入/输出对(x', y')尝试每一种可能，就能找到正确的那一个。这样就获得了完整的密钥，只用了两个已知的明文且用时约为2^{16}。

两轮 DES。在两轮 DES 中，输出y和(L_2, R_2)相等，这里

$$
\begin{aligned}
L_1 &= R_0 \\
R_1 &= L_0 \oplus f_1(R_0) \\
L_2 &= R_1 = L_0 \oplus f_1(R_0) \\
R_2 &= L_1 \oplus f_2(R_1)。
\end{aligned}
$$

注意，L_0, R_0, L_2, R_2是由给定的输入/输出对(x, y)得知的，因此有$L_1 = R_0$和$R_1 = L_2$。这意味着已经知道f_1和f_2的输入/输出，所以攻击单轮 DES 的方法同样可以用在这里来确定k_1和k_2，用时约为$2 \cdot 2^{16}$。即使k_1和k_2是完全独立的密钥，这个攻击也同样奏效。事实上，DES

的密钥编排确保了k_1和k_2的许多比特是相等的，这可以被用来使攻击的速度变得更快。

三轮 DES。如图 5.5 所示，输出值y等于(L_3, R_3)。因为$L_1 = R_0$和$R_1 = L_2$，所以图中的未知值只有R_1和L_2（它们是相等的）。

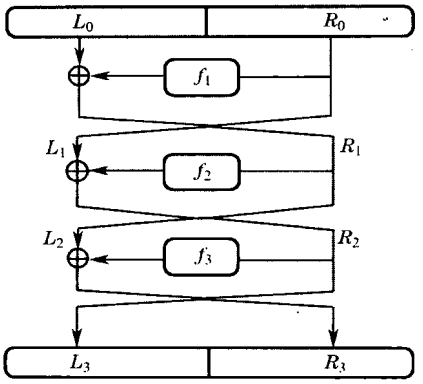


图 5.5　三循环 DES

现在不再有任何函数f_i的输入/输出了。例如，考虑f_2。在这种情况下，输出值等于$L_1 \oplus R_2$，这里两个值都是已知的。然而，不知道f_2的输入值R_1。的确知道$R_1 = L_0 \oplus f_1(R_0)$以及$R_1 = R_3 \oplus f_3(R_2)$，但是$f_1$和$f_3$的输出值都是未知的。类似的实践表明：对于$f_1$和$f_3$只能确定输入，而不能确定输出。因此，破解单轮和两轮 DES 时所用的攻击在这里就不起作用了。

不依赖某轮函数的输入和输出的全部信息，我们将利用f_1和f_3的输入和输出之间的某种关系。观察到f_1的输出等于$L_0 \oplus R_1 = L_0 \oplus L_2$，$f_3$的输出等于$L_2 \oplus R_3$。这意味着$f_1$的输出和$f_3$的输出的异或等于

$$(L_0 \oplus L_2) \oplus (L_2 \oplus R_3) = L_0 \oplus R_3$$

也就是说，f_1和f_3输出的异或是已知的。此外，f_1的输入是R_0并且f_3的输入是L_3，两者都是已知的。因此结论是可以确定f_1和f_3的输入，以及它们输出的异或。现在描述一种基于这些信息的能发现密钥的攻击。

回顾一下，DES 的密钥编排有个特性，就是主密钥被分成“左半部分”，用k_L表示，和“右半部分”，用k_R表示，每一个包含 28 比特。此外，每轮中用到的子密钥的最左比特是从k_L取得，并且每一个子密钥的最右比特从k_R取得。这意味着在任何一轮中主密钥的左半部分只影响前 4 个 S 盒的输入，而主密钥的右半部分只影响后 4 个 S 盒的输入。因为混合置换是已知的，所以同样知道每轮函数输出的哪些比特是来自每个 S 盒的。

这个攻击背后的思想是分别穿越主密钥每一半的密钥空间，攻击的复杂度（约）为$2 \cdot 2^{28}$而不是2^{56}。这样的攻击是可行的，如果可以确定对半个主密钥的猜测，现在说明这是怎么完成的。令k_L是对主密钥左半部分的猜测。知道f_1的输入为R_0，所以使用对k_L的猜测可以估算出前 4 个 S 盒的输入。这意味着可以估算f_1的一半的输出（混合置换展开了已知的比特，但是因为混合置换是已知的，所以可清楚地知道具体是哪些比特）。同样地，可以使用已知的函数f_3的输入L_3，和同样的对k_L的猜测，来计算f_3在相同位置的输出。最后，可以计算这些输出值的异或，并且检测它们是否与已知的f_1和f_3的输出值的异或相匹配。如果它们不相等，那么猜测的k_L就是不正确的。正确的半个密钥k_L总是会通过测试的，而不正确的半个密钥通过测试的可能性约为2^{-16}（因为是在两个计算值上检验 16 比特的相等性）。这里有2^{28}种可能的

半密钥可以试，所以期望在以上所说之后，对k_L来说剩下$2^{28}/2^{16}=2^{12}$的可能性。

通过对主密钥的每一半使用以上措施，在$2\cdot 2^{28}$的时间内近似地获得了左半部分的2^{12}个候选和右半部分的2^{12}个候选。因为左半部分和右半部分的每一个结合都是可能的，所以总共有2^{24}个候选密钥，并且可以使用另外一个输入/输出对(x',y')来对2^{24}个候选密钥进行强力搜索。这个攻击的复杂度大约为$2\cdot 2^{28}+2^{24}<2^{30}$，而它的空间复杂为$2\cdot 2^{12}$。这样的复杂度的攻击可以在标准的个人计算机上进行。

5.3.3 DES 的安全性

在几乎 30 年的大量研究之后，已知对 DES 的最好的实用攻击仍然只是对密钥空间的穷举搜索。（下面讨论一些重要的理论攻击。这些攻击需要大量的输入/输出对，这些数对在任何使用 DES 的现实系统中是很难获得的。）不幸的是，DES 的 56 比特的密钥长度较短，这样对所有2^{56}个可能密钥的穷举搜索现在就变得可行了（虽然仍然是不简单的）。在 70 年代末，已经有一些对 DES 选择如此短的密钥的强烈反对。在那之前，这些反对是理论上的，因为搜索如此多的密钥所需计算能力还不存在[6]。然而，现今对 DES 的蛮力攻击的实践可能性，在 1997 年被证实，当时由 RSA 公司组织的 DES 挑战被解决了（挑战的形式是给出输入/输出对，对第一个发现密钥的人或组织给予奖励）。第一个挑战是在 1997 年由 DESCHALL 项目使用 Internet 协调的几千台计算机来破解的，整个计算用了 96 天的时间。第二个挑战是在第二年中由 distributed.net 项目仅用 41 天破解的。随后，在 1998 年一个重要的突破性进展到来，第三个挑战只在 56 小时内被解决。这个壮举是由一个被称为 Deep Crack 的专用 DES 破解机完成的，它是由电子前沿基金会花了 25 万美元建造的。最近的一次挑战只用了 22 个多小时就解决了（由 Deep Crack 和 distributed.net 的联合努力）。DES 的密钥太短，在当今的任何应用中都不能被认为是安全的。

与 DES 较短的密钥长度相比重要性次之，但是仍然值得关注的是，相对较短的分块长度。短的分块长度有问题的原因是许多基于分组密码的构造方法的安全性都是依赖于分块的长度（即使使用的加密是“完美的”，并且不考虑密钥长度）。例如，对计数模式加密的安全性的证明（如定理 3.29）表明了即使使用完全随机函数，如果获得了q个明文/密文对，攻击者也会以$\Theta(q^2/2^n)$的可能性攻破这个加密方案，这里n表示分块长度。在 DES 的$n=64$时，意味着如果攻击者只获得了$q=2^{27}$的明文/密文对，攻破安全的概率也很高。如果敌手窃听到加密的消息包括已知的标题、冗余等，那么获得明文/密文对就相对容易（虽然获得2^{27}对这样的数是无法办到的）。

需要强调 DES 的不安全与它的内部结构和设计是无关的，而只是由于其较短的密钥长度（并且在某种程度上，是由于其较短的分块长度）。这是对 DES 设计者的一个大赞美，DES 看上去成功地构造了一个几乎“完美”的分组加密（除了密钥太短以外）[7]。因为 DES 本身似乎没有明显的结构缺陷，使用 DES 作为构造攻击构造拥有更长密钥的分组加密是有意义的。5.4 节中讨论这种方法。

向前看一点，注意到高级加密标准 AES（DES 的替代标准）明确的设计目的是解决短密钥和短分块长度问题。AES 支持长度为 128 比特（以及更长）的密钥，分块长度为 128 比特。

---

⑥ 在 1977 年，一台可以在一天攻破 DES 的机器需要耗费 2 千万美元。

⑦ 实际上，DES 的设计者几乎肯定意识到密钥太短。NSA 建议密钥要足够短，这样他们才能破解。

对 DES 的高级密码分析攻击。上述成功的强力攻击并没有利用任何 DES 的内部弱点。的确，许多年来没有人知道存在这样的弱点。这一点上的第一次突破是在 80 年代末来自 Biham 和 Shamir 的工作，他们提出了一种叫做差分密码分析的技术，并利用该技术设计了比蛮力搜索用时更少的一种对 DES 的攻击。他们的攻击要用2^{37}次（并使用可忽略不计的存储），但是需要攻击者分析来自2^{47}个已选明文集合的2^{36}个密文。虽然这种攻击的出现是来自理论观点的一个突破，但它并没有表现出很大的实用意义，因为很难想象现实中敌手可以获得这么多明文进行选择明文攻击。

有趣的是，Biham 和 Shamir 的研究表明 DES 的 S 盒是被设计专门用来对抗差分密码分析（在某种程度上）的，表明差分密码分析技术是被 DES 的设计者已知的（但没有公开透露）。在 Biham 和 Shamir 公开结果之后，DES 的设计者声称他们确实考虑了差分密码分析，而且已设计的 DES 阻止了这种类型的攻击（但是被 NSA 要求在涉及国家安全时保持沉默）。

伴随着 Biham 和 Shamir 的突破，线性密码分析在 90 年代早期由 Matsui 提出，且成功地应用于 DES。Matsui 的攻击优势在于虽然它仍然需要大量的输出值（确切的说是2^{43}），但它们可以是任意的，且不需要被攻击者选择。（也就是说，它利用了已知明文攻击而不是选择明文攻击。）尽管如此，仍然很难想象现实中可能获得如此大量的输入/输出（或者明文/密文）对。

5.6 节中简要描述差分密码分析和线性密码分析背后的基本思想。然而，虽然使用复杂的密码分析技术，比蛮力搜索攻破 DES 所用时间少，穷举密钥搜索仍然是实践中最有效的攻击。

5.4 增加分组密码的密钥长度

唯一已知的 DES 在实践中的弱点是它相对较短的密钥。因此尝试使用“基本的”DES 作为构造工具来设计一个拥有更长密钥的分组密码就显得有意义了。这一部分讨论一些这么做的方法。虽然在整个讨论中都提及 DES，DES 仍然是应用这些技术的最突出的例子，这里所说的一切通常适用于任何分组密码。

内部干扰和黑盒构造方法。有两种通用的方法可用来构造另一个基于 DES 的分组密码。第一个方法可能会在某种程度上改变 DES 的内部结构，同时增加密钥长度。举个例子，其中一个不改变变形函数，仅仅使用一个 128 比特主密钥，且密钥编排不同（每一轮中仍然选择 48 比特大小的子密钥）。或者可以改变 S 盒本身并且在每一轮中使用更长的子密钥。这种方法的劣势是要改变 DES，即使最小的改动也会失去对修改的 DES 的信心，尽管 DES 多年以来一直保持安全。更关键的是，密码学结构是非常敏感的，因而即使是轻微的，似乎不重要的改变也会致使原始的构造方法变得彻底不安全[⑧]。因此不建议改变分组密码的内部结构。（如果确实认为应该对分组密码引入一些改变，通常更好的是从草案中构造出一个新的分组密码。当然假设工作是由专家完成的。不建议设计一个新分组密码，尤其是有像 AES 之类的现代分组密码的时候。）

一个没有上述问题的替代方法是将 DES 当成一个“黑盒子”来使用。也就是说，在这种方法中完全忽略 DES 的内部结构，并且把它视为实现了密钥长为 56 比特的“完美的”分组

⑧ 事实上，这种效果的各种结果已经在 DES 中显现；例如，轻微地改变 S 盒使得 DES 更容易被攻击。

密码。然后，通过调用原始的没有修改的 DES 来构造新的分组密码。因为 DES 本身一点都没改变，这个方法非常可能形成一个安全的分组密码（虽然它也可能导致效率较低），这个方法将在这里探索。

双重加密。令F为一个分组密码。拥有两倍原始长度的密钥的分组密码F'可以这样定义：

$$F'_{k_1,k_2}(x) \stackrel{\text{def}}{=} F_{k_2}(F_{k_1}(x))$$

这里k_1和k_2是相互独立的密钥。如果F是 DES，那么结果就是一个密钥大小为 112 比特的分组密码F'。如果穷举密钥搜索是可用的对F'的最佳攻击，那么 112 比特的密钥长度就足够了，因为需要2^{112}时间的攻击完全是不可能的。不幸的是，现在介绍一个对F'的攻击，这个攻击用时约为2^n，在原始密钥k_1和k_2的长度为n的情况下（分块长度至少是n）；这明显比期望的穷举搜索$2n$比特大小的密钥所需时间少。这意味着新的分组密码本质上并不比旧的好，即使它拥有两倍的长度⑨。

这个攻击被称作“中间相遇攻击”，下面解释其原因。假定给敌手单个的输入/输出对(x,y)，这里$y = F'_{k_1,k_2}(x) = F_{k_2}(F_{k_1}(x))$。敌手将会通过以下方式缩小可能密钥集合的范围：

(1) 设$S := \phi$。

(2) 对于每一个$k_1 \in \{0,1\}^n$，计算$z := F_{k_1}(x)$，并将(z,k_1)储存在L中。

(3) 对于每一个$k_2 \in \{0,1\}^n$，计算$z = F^{-1}_{K_2}(y)$，并将(z,k_2)储存在L'中。

(4) 根据第一个组成部分，分别排序L和L'。

(5) 如果$z_1 = z_2$，假设L的一个条目$(z_1,\widetilde{k_1})$以及L'的一个条目$(z_2,\widetilde{k_2})$是匹配的。对于每一个匹配，将$(\widetilde{k_1},\widetilde{k_2})$添加到$S$中。

这种算法产生的集合S的确包含那些$y = F'_{k_1,k_2}(x)$的值(k_1,k_2)。满足条件的原因是因为它输出满足以下式子的(k_1,k_2)：

$$F_{K_1}(x) = F^{-1}_{K_2}(y) \tag{5.3}$$

这个式子在当且仅当$y = F'_{k_1,k_2}(x)$时成立。这种攻击的描述如图 5.6 所示。

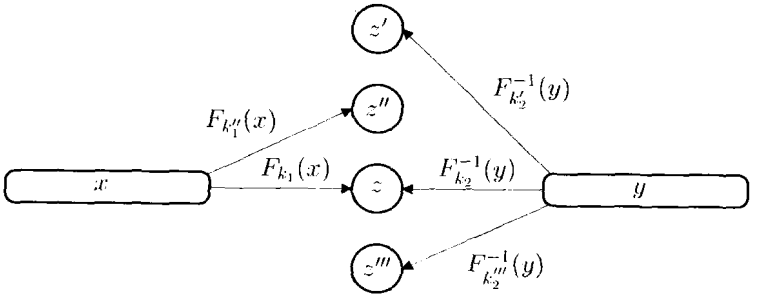


图 5.6　中间相遇攻击

如果n也是F的分块长度，那么随机组合对(k_1,k_2)就会被期望以约2^{-n}的概率满足等式（5.3），并且S中的元素数目接近$2^{2n}/2^n = 2^n$。给定另外两个输入/输出对，并尝试S的所有2^n个元素，这些组合被期望以非常高的概率标识正确的(k_1,k_2)。

复杂度。目录L和L'在约2^n时间内被产生和排序（例如，使用计数排序）。一旦目录被排

⑨ 这不是非常正确，因为对F的蛮力攻击能够在2^n时间和固定存储中完成，然而，对F'的攻击需要2^n时间以及2^n存储（存储比时间更昂贵）。尽管如此，攻击表明F'并没有达到足够的安全水平。

序，所有的匹配能在$\mathcal{O}(|S|+|L|+|L'|)=\mathcal{O}(2^n)$时间内被找到（见练习 5.13）。使用额外的组合$(x',y')$来确定正确的密钥要用$\mathcal{O}(|S|)=2^n$时间。因而上述算法时间复杂度为$\mathcal{O}(2^n)$。

三重加密。对前面方法的明显一般化是连续使用分组密码三次。这种方法有两种变形很普遍：

(1) 变形 1：三个独立的密钥：选择三个独立的密钥k_1,k_2,k_3，并定义$F'_{k_1,k_2,k_3}(x) \stackrel{\text{def}}{=} F_{k_3}(F_{K_2}^{-1}(F_{k_1}(x)))$。

(2) 变形 2：两个独立密钥：选择两个独立密钥k_1,k_2，并定义$F'_{k_1,k_2}(x) \stackrel{\text{def}}{=} F_{k_1}(F_{K_2}^{-1}(F_{k_1}(x)))$。

在比较这两个替代方法的安全性之前，注意原始分组密码的中间那次调用实际上是逆向的。如果F是一个足够完善的分组密码，那么安全性就没什么区别了，因为如果F是一个强伪随机置换，那么F^{-1}也必然是。F、F^{-1}、F这种奇怪的更迭变化的原因是：如果有人选择了$k_1=k_2=k_3$，那么结果就相当于调用一次带有k_1的F。这确保了向后兼容性（即为了回到调用F一次，把所有的密钥设置成相等的就足够了）。

第一种变形的安全性。这个变形的密钥长度是$3n$（这里，与之前一样，原始分组密码F的密钥长度是n），因此希望对此分组密码的最好攻击需要用时2^{3n}。然而，就像在双重加密中的情况一样，这个密码可能遭受中间相遇攻击的影响，虽然现在攻击只要时间2^{2n}。这是最著名的攻击。因此，虽然这个变形没有希望的那样安全，但如果$n=56$，用于实践有足够的安全性（当然，假设原始分组密码F没有缺陷）。

第二种变形的安全性。这个变形的密钥长度是$2n$，所以所能期望的最好的情况是安全性可以经得起运行时间为2^{2n}的攻击。当敌手只给予单个输入/输出对时，已知的攻击都没有更好的时间复杂度。但是，使用2^n个选择的输入/输出对，在2^n时间内找到密钥的已知的选择明文攻击是存在的（见练习 5.12）。不考虑这个，它仍然是实践中合理的选择。

三重 DES（3DES）。三重 DES 的基础是使用 2 个或 3 个密钥对 DES 进行三重调用，就像上面描述的。它被广泛地认为是高度安全的，并在 1999 年作为一个官方标准取代了 DES。三重 DES 至今仍被广泛地使用，并且被认为是非常强的分组密码。它唯一的缺点是其相对较小的分块长度，以及由于它需要 3 个完整的分组密码运算而非常慢。这些缺点导致 DES/三重 DES 被 AES 所取代，见下一节的介绍。

5.5 AES——高级加密标准

在 1997 年 1 月，美国国家标准和技术协会（NIST）宣布会举办一个比赛来选出一个新的分组密码（称为高级加密标准或 AES），去代替 DES。这个比赛从公开号召团队提交候选分组密码方案开始，全世界总共提交了 15 种不同的算法，这些所提方案中包含了许多当今最优秀的密码学家和密码分析学专家的作品。每一个团队的候选方案都由 NIST 成员、公众和（尤其是）其他团队进行精心地分析。1998 年和 1999 年分别又举办了两次研讨会，研讨会上展示了对各种提交方案的密码学分析攻击。在 1998 年研讨会中，NIST 将范围缩减到 5 个“决赛者”，竞赛的第二个回合开始。在 1999 年研讨会中邀请 NIST 对这 5 个决赛入围者进行额外的审核。在 2000 年 10 月，NIST 宣布获胜的算法是 Rijndael（一个来自比利时的由 John Daemen 和 Vincent Rijmen 设计的分组密码），虽然 NIST 承认 5 个决赛者中的任何一个都是

很好的选择。特别是，在 5 个决赛者中的任何一个中都没有发现严重的安全漏洞，而且对获胜者的选择有一部分是基于效率、灵活性等特性。

选拔 AES 的过程非常具有创意，因为任何提交算法且希望他们的算法被采用的团队，都有强烈的动机去寻找对其他算法的攻击方法[10]。在这种方式下，实质上所有世界上最优秀的密码学家都在集中精力寻找竞赛中候选方案的即使是最微小的缺陷。在过去的几年中，每一个候选算法都已经经历了大量的研究，因此增加了对胜出算法的安全性的信心。当然，算法在不被攻破的情况下使用的时间越长，信心则增长得越多。今天，AES 已经被广泛使用并且没有发现明显的安全漏洞。

AES 的构造方法。本节将要从高层面介绍 Rijndael/AES 的结构。（从技术上讲，Rijndael 和 AES 不是相同的，其区别对这里的讨论来说并不重要。）关于 DES，将不介绍完整的说明，而且描述将作为实现的基础。目的只是提供一个算法如何工作的总体思想。

AES 分组密码拥有 128 比特的分块长度，而且可以使用 128、192 或者 256 比特大小的密钥。密钥的长度影响着密钥编排（即在每一轮中使用的子密钥）和轮的次数，但是不影响每一轮中的高级结构。

与使用 Feistel 结构的 DES 相比，AES 本质上是一个代替置换网络。在 AES 的计算中，有一个被称为状态的 4 乘 4 字节数组，是通过多轮操作来修改的。这个状态的最初设置和分组密码的输入是相等的（注意输入是 128 比特，确切的是 16 字节）。下面的操作会在每一轮的 4 个阶段中操纵这个状态。

(1) 阶段 1——密钥加（AddRoundKey）：在 AES 的每一轮中，128 比特的子密钥都是从主密钥处推得的，并解释成 4 乘 4 的字节数组。状态数组与子密钥异或，实现更新。

(2) 阶段 2——字节代替（SubBytes）：在这一步中，状态数组中的每个字节都被另一个字节代替，这个代替的字节是由一个固定的查询表S决定的。这个替换表（或S盒）是一个在$\{0,1\}^8$上的双射。只有一个S盒，且它在每一轮中被用来代替状态数组的所有字节。

(3) 阶段 3——行移位（ShiftRows）：在这一步中，状态数组每一行的字节按如下方式循环左移：第一行不动，第二行左移一位，第三行左移两位，第四行左移三位。所有的移位都是循环的，例如第二行中第一个字节变成了第四个字节。

(4) 阶段 4——列混合（MixColumns）：在这一步中，对每一列进行可逆的线性转换。可以视为矩阵乘法（在某些适合的域上的运算）。

通过将第 3 个和第 4 个阶段一起视为“混合”阶段，可看到 AES 的每一轮都有代替置换网络的结构：某轮的子密钥首先和当前这一轮的输入进行异或；接下来，一个小的可逆函数被用在“分块”上；最后，结果的比特被混合起来以得到扩散。唯一的不同是，不像对代替置换网络的总体描述，这里的混合步骤不是由比特的简单置换组成的，而是使用比特的可逆线性转换来完成的。（将事情简化一点，看一个简单的 3 比特的例子，$x = x_1\|x_2\|_3$的比特置换，可能将x映射到$x' = x_2\|x_1\|x_3$。一个可逆线性转换可将x映射到 $x_1 \oplus x_2\|x_2 \oplus x_3\|x_1 \oplus x_2 \oplus x_3$。）

AES 中的轮数取决于密钥的长度。128 比特的密钥对应 10 轮，192 比特的密钥对应 12 轮，而 256 比特则对应 14 轮。在 AES 的最后轮中，“列混合”步骤被额外的“密钥加”步骤取代（这可以阻止敌手简单地逆转最后三个阶段，这是不依赖于密钥的）。

AES 的安全性。如前所述，AES 加密在选择过程中承受了大量的审查，这种审查直到现

[10] 动机不是来自于财务方面，因为胜出的方案是不能申请专利的。尽管如此，它能带来的是极大的荣誉。

在还在继续。至今，发现的唯一非平凡的密码分析攻击就是针对轮数减少的 AES。通常很难比较密码分析攻击，因为趋向于更好表现的攻击与参数相关。这里描述一组攻击的复杂度简要介绍一些已知的情况。有一些已知攻击针对密钥为 128 比特的 6 轮 AES（使用约2^{72}数量级的加密），有的攻击针对密钥为 192 比特的 8 轮 AES（使用约2^{188}数量级的加密），有的攻击针对密钥为 256 比特的 8 轮 AES（使用大约2^{204}数量级的加密）。上述攻击是针对轮数减少的 AES，而且至今没有发现比穷举密钥搜索更好的对完整 AES 构造方案的攻击。（此外，即使是对轮数减少的 AES 的攻击的复杂度也是很高的。）

因此至今为止，对几乎所有依赖伪随机置换的密码学实现而言，AES 是一个极好的选择对象。它是免费的、标准化的、高效的和高度安全的。

5.6 差分密码分析和线性密码分析简介

典型的分组密码有相对复杂的结构，因而很难去分析和破译。然而，人们不应该愚蠢地认为一个复杂的分组密码是很难攻破的。相反，构造一个安全的分组密码是很难的，构造一个不安全的分组密码却出奇地容易（无论它看起来有多复杂）。这应该被视作一个警告，那就是非专家不应该尝试着去构造一个新的分组密码，除非有非常好的理由。如果三重 DES 和 AES 可用，那么在大多数应用中很难去论证其他选择的安全性。

本节将简要提到当前一个密码分析学家的工具箱中标准的两个工具。这种工具的出现应该被视为强化了上面这个警告，那就是构造一个好的分组密码是非常困难的。

差分密码分析。这个技术第一次出现是在 20 世纪 80 年代末期，由 Biham 和 Shamir 提出，他们使用它在 1993 年攻击 DES。攻击背后的基本思想是：将输入中特定的不同制成表格，这些不同会导致输出中特定的不同。具体而言，假定分组密码有分块长度n并且令$\Delta_x, \Delta_y \in \{0,1\}^n$。差别$\Delta_x, \Delta_y$以概率$p$出现，如果随机输入$x_1$和$x_2$满足$x_1 \oplus x_2 = \Delta_x$，以及随机选择的密钥$k$，$F_k(x_1) \oplus F_k(x_2) = \Delta_y$的概率是$p$。很明显，对于一个随机函数来说，应该没有以比$2^{-n}$高得多的概率出现。然而，在一个弱的分组密码中，这种差别可能会以明显的更高概率出现。

人们可以找出以高概率出现的差别，如通过蛮力搜索（对分组密码进行彻底地搜索，独立于任何特定的密钥），或者通过对分组密钥本身进行仔细地分析。如果一个差别以$p \gg 2^{-n}$的概率出现，那么分组密码就不再有成为伪随机置换的资格了。差分密码分析的关键是利用许多个差别，每一个差别可能只比2^{-n}稍微大一点，并使用选择明文攻击来恢复秘密密钥（因此就完全地攻破了分组密码）。这里不再讨论细节，虽然对拥有给定差别的随机输入组合使用分组密码会使得密码分析人员能够分离出部分秘密密钥，并对这一部分的进行证实猜测。正如前面讨论的对两轮代替置换网络的攻击，隔离出一部分秘密密钥使攻击者获得密钥的时间少于蛮力搜索所用时间。然而，需要注意的是，选择需要的明文这一事实使得攻击在现实中有一定局限。

虽然差分密码分析并没有表现出对 DES 和 AES 产生任何实际的攻击（因为实行攻击所需选择的明文数量是巨大的），但是差分分析曾经成功地被用来攻击其他的分组密码。一个重要的例子就是 FEAL-8，在使用差分分析的情况下，FEAL-8 被完全地破解了。

线性密码分析。线性密码分析是在 90 年代早期由 Matsui 所创造的。这个方法考虑到了输入和输出之间的线性关系。假定如果比特位$i_1, \cdots, i_l$和$i'_1, \cdots, i'_l$有偏移 ，对于随机选择的输

入x和密钥k，有

$$\Pr[x_{i_1} \oplus \cdots \oplus x_{i_l} \oplus y_{i'_1} \oplus \cdots \oplus y_{i'_l} = 0] = p,$$

这里$y = F_k(x)$，x_i, y_i代表x和y的比特，对于真正的随机函数，期望偏差接近 0.5。Matsui 介绍了如何在给定的分组加密F中使用足够大的偏差，通过寻找秘密密钥来完全攻破加密方案。这个攻击的一个重要特征是：它不需要选择明文，已知明文就足够。即使是这样，如果需要非常大数量的明文/密文对，那么这个攻击在大多数情况下是不切实际的。

参考文献和扩展阅读材料介绍

混淆扩散范例和代替置换网络由 Shannon[127]和 Feistel[53]引入。如果想了解更多的代替置换网络设计信息请参考 Heys 的论文[78]。Luby 和 Rackoff 在文献[97]给出了对它们的理论分析，将在第 6 章讨论。

DES 标准在文献[109]中可以找到，并且在由 Kaufman 等写的书[87]中有对其更便于阅读的描述。关于选择 Rijndael 作为 AES 的竞赛的细节见 http://csrc.nist.gov/CryptoToolkit/aes/index.html。对 AES 的综合性描述可以在文献[87]和它的设计者 Daemen 和 Rijmen 的书[41]中找到。Cid 等在文献[33]中介绍了一种可能导致对 AES 的密码分析学攻击的方法，还有大量其他的好的（和稍逊的）分组密码。关于其他密码的广泛使用的但有些旧的概述，见文献[99]的第 7 章。

对双重加密的中间相遇攻击是由 Diffie 和 Hellman[48]提出的。本书提到的双密钥三重加密的攻击（在练习 5.12 中探索）是由 Merkle 和 Hellman[103]提出的。双重和三重加密的安全性的理论分析可以在文献[3，17]中找到。

DESX 是增加 DES 的有效密钥长度的另一个技术，不需要对 DES 的额外调用。秘密密钥由值$k_i, k_o \in \{0,1\}^n$和$k \in \{0,1\}^{56}$组成，而且分组密码由下式定义：

$$DESX_{k_i,k,k_o}(x) = k_o \oplus DES_k(x \oplus k_i)。$$

这个方法首先是被 Even 和 Mansour[52]研究的，它在 DES 上的实际应用是由 Rivest 提出的，而它的安全性是随后由 Kilian 和 Rogaway 分析的[88]。

差分密码分析是由 Biham 和 Shamir[18]引入的，它在 DES 上的应用在文献[19]中论述。Coppersmith[34]在考虑到差分密码分析的情况下介绍了 DES 的设计。线性密码分析是由 Matsui[98]提出的。Langford 的论文[93]包含了对差分和线性密码分析的进一步改善，并且考察了到 1995 年为止对 DES（和轮数减少的 DES）的已知的攻击。这些高级密码分析技术的更多信息，我们建议读者阅读由 Heys 写的关于差分和线性密码分析的优秀指导书[77]。更简明的介绍可以在 Stinson 写的教科书[138]中找到。

练　习

5.1 假定分组密码F有以下性质：对于随机选择的密钥k，给定可访问的随机预言机F_k，只需要对随机预言机发出 100 次问询，就可以确定k（以及最小计算时间）。形式化地说明F_k不是一个伪随机置换。

5.2 在对 2 轮代替置换网络的攻击中，思考一个分块长度为 64 比特，有 16 个 S 盒，且每个 S 盒都有 4 比特输入的网络。对 8 个 S 盒，每个 S 盒的输入为 8 比特的情况，重新分析

攻击的复杂度是什么？对分块长度是 128 比特，以及拥有 8 比特输入的 16 个 S 盒的情况，再分析一次。分块长度会造成任何不同吗？

5.3 对 3 轮代替置换网络的攻击并没有恢复密钥，而只是介绍了如何将分组密码从随机置换区分开，因此它不是“完全破解”。尽管如此，也表明在计数模式加密方案（见 3.6.4 节）中使用 3 轮代替置换网络可对加密的安全性产生灾难性的影响。

5.4 考虑一个修改过的代替－置换网络，这里不是以交替方式在r轮中执行密钥混合、代换、置换这些步骤，而是先执行r轮密钥混合，然后执行r轮的代换，最后执行r轮置换。分析这个构造方法的安全性。

5.5 在以下两种情况下，当输入是(L_0, R_0)时，r轮 Feistel 网络的输出是什么：

① 每一轮函数输出所有的 0，忽略输出。

② 每一轮函数都是恒等函数。

5.6 说明 DES 有这样的特性，就是对于每一个密钥k和输入x（这里$\overline{z}$表示z的按位求补）来说都有$DES_k(x) = \overline{DES_{\overline{k}}(\overline{x})}$。这被称为 DES 的互补特性。（这章所给的关于 DES 的描述可以足够完成这个练习。）

5.7 应用上一个练习说明如何才可能在 DES 中（以概率为 1）于时间2^{55}里找到秘密密钥。

提示：使用选择明文攻击，使用两个仔细选择的明文。

5.8 在 DES 的实际构造方法中，Feistel 网络的最后一个轮操作输出的两半部分被交换了。也就是说，如果最后一个轮的输出为(L_{16}, R_{16})，那么分组密码的输出实际上是(R_{16}, L_{16})。说明对DES_k和DES_k^{-1}（给出两半部分的交换）的计算之间的唯一区别就是子密钥的顺序。

5.9 （本练习采用上一个练习的结果。）

① 对于$k = 0^{56}$，有$DES_k(DES_K(x)) = x$。为什么使用该密钥会面临安全威胁？

② 找到另外三个具有同样性质的 DES 密钥，它们被认为是 DES 的弱密钥。

③ 存在这样的 4 个弱密钥是否说明了 DES 有严重的弱点？解释你的观点。

5.10 描述对下面 DES 修改方法的攻击。

① 每轮的子密钥都是 32 比特长，并且变形函数只将该轮的子密钥和该轮的输入异或（即$\widetilde{f}(k, R) = k_i \oplus R$））。对于这个例子，密钥编排不重要，你可以将$k_i$看成是独立的密钥。

② 在每一轮中使用相同的 48 比特子密钥而不是不同的子密钥。说明如何可以不用2^{48}数量级的时间进行蛮力搜索将分组密码和随机置换区分开来。

提示：练习 5.8 和 5.9 可能有帮助……

5.11 说明对三轮 DES 攻击的改进就是使用两个输入/输出对，在$2 \cdot 2^{28} + 2 \cdot 2^{12}$时间内恢复密钥。

5.12 这个问题阐明了对双密钥三重加密的攻击。令F为一个拥有n比特分块长度和密钥长度的分组密码，并设$F'_{k_1,k_2}(x) = F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$。

① 假设给定一对(m_1, m_2)，就可能在恒定的时间内找到所有密钥k_2，满足$m_2 = F_{k_2}^{-1}(m_1)$。说明如何使用 3 个已知的输入/输出对，在大概2^n的时间内（以很高的概率）恢复F'的整个密钥。

② 通常，不可能像上边那样在恒定时间内找到k_2。但是，说明通过使用2^n时间的预先处理步骤，就可能在给定m_2的情况下，在（基本上）恒定的时间内找到所有的密钥k_2满足$m_2 = F_{k_2}^{-1}(0^n)$。

③ 假设k_1是已知的，并且上面的预处理步骤已经执行过了。说明如何使用一对(x, y)，对选择的输入值x，在恒定的时间内确定k_2。

④ 将上面的组成部分放在一起来设计一个攻击，这个攻击可以通过运行大约2^n时间，并需要对大约2^n的选择输入进行加密，来复原整个密钥。

5.13 说明在对双重加密的中间相遇攻击中，在与匹配数和两个排序列表的长度呈线性关系的时间里，如何发现所有的匹配。

5.14 DES 的密钥编排修改如下：主密钥的左半部分被用来推导 1 到 8 轮中的所有子密钥，而主密钥的右半部分被用来推导 9 到 16 轮中的所有子密钥。演示一个对这个修改方案的攻击，这个攻击可以在大约2^{28}的时间里恢复整个密钥。

5.15 考虑把 DES 作为定长抗碰撞散列函数：定义$h \cdot \{0,1\}^{112} \rightarrow \{0,1\}^{64}$为

$h(x_1 \| x_2) \stackrel{\text{def}}{=\!=} DES_{x_1}(DES_{x_2}(0^{64}))$，其中$|x_1| = |x_2| = 56$。

① 写出h中明确的碰撞。

提示：参考练习 5.9。

② 描述对于给定的值y（对于x_1, x_2有$h(x_1 \| x_2) = y$）如何在2^{56}时间内找到原像；

③ 描述一个更加高明的原像攻击，该攻击在2^{32}时间内运行，且成功率更高。

提示：依靠附录 A.4 中的结果。

第 6 章　*伪随机对象的理论构造

第 3 章介绍了伪随机性的概念，定义了基本密码学原语，如伪随机发生器、伪随机函数和伪随机置换。在第 3 章和第 4 章中，说明了这些原语就是对称密钥密码学的构造单元。鉴于此，从理论的角度理解这些原语就显得非常重要。本章正式地介绍单向函数的概念，非正式地讲就是这样一种函数，正向计算容易但是反向求逆十分困难，同时介绍仅仅基于单向函数存在的假设如何构建伪随机发生器和伪随机置换[①]。此外，应该理解：对于任何非平凡的对称密钥体制的密码学任务来说，单向函数是必需的假设。综合起来考虑，这就意味着单向函数的存在性等价于所有（非平凡的）对称密钥方案的安全性。这个结论是现代密码学主要的贡献之一。

在本章中介绍的基于单向函数的伪随机置换的构造方法可以视为前一章给出的分组加密构造方法的补充。前一章关注伪随机置换是如何在当前的实践中实现的，其目的是介绍一些在构造中用到的基本方法和设计原理。然而，有点令人失望的是在介绍的构造方法中，没有一个是在任何更弱的（即更合理的）假设的基础上可证明安全的。相比之下，本章中将证明：基于非常轻微的假设，即假设单向函数是存在的，构造伪随机置换是可能的。这个假设比起假设 DES 是一个伪随机置换要更加合理，这是因为有很多被研究了多年的候选单向函数，甚至在密码学出现之前就在研究。（第 5 章开头对这一点做了更深入的讨论。）虽然，缺点是这里讨论的构造方法的效率远不如第 5 章中的构造方法，因而并不在实际中应用。这给密码学家留下了一个重大的挑战来“弥补这一鸿沟”，以及发展可证明安全的伪随机发生器、伪随机函数、伪随机置换的构造方法，而且这些构造方法的效率与可用的序列密码和分组密码的最好情形相当。

关于本章的注解。本章的内容比本书其他章节在某种程度上更深一些。正如本章被标记为“带星号的”，这里介绍的内容并没有在其他章节中使用，所以如果愿意的话可以跳过本章。虽然如此，我们试着通过一种让本科高年级学生或者低年级研究生（通过努力后）能够理解的方式来介绍本章的内容。强烈鼓励每个人都阅读一下 6.1 节和 6.2 节，这两部分介绍了单向函数，同时还给出了本章其余部分描述的各种构造方法的一个综述。可以确信，对本章某些主题的熟悉，其重要性对于所付出的努力来说是足够的。

6.1　单 向 函 数

本节正式定义单向函数，然后简单地讨论一些被广泛认为满足该定义的候选单向函数。（第 7 章和第 11 章给出更多的猜想的单向函数的例子。）通过介绍“硬核谓词（hard-core predicate）”的概念结束本节；这些可以看成是（在某种意义上讲）对单向函数求逆的困难性的“封装”，将会在接下来的几节里的各种构造方法中被广泛地应用。

① 实际上，这不是非常正确，因为我们本章中大部分内容基于单向置换。尽管如此，我们知道单向函数已经足够。

6.1.1 定义

一个单向函数f有正向计算容易而反向求逆十分困难的特性。第一个条件很容易形式化：只需简单地要求函数f在多项式运算时间内可计算。对于第二个条件，因为最终感兴趣的是构造密码学方案，该方案对于概率多项式时间的敌手来说是很难攻破的，即攻破的概率是可忽略的。本节将对f求逆的困难性进行形式化描述，即要求：任何概率多项式时间的算法对f求逆是不可行的。这里f求逆就是说，给定y找其原像的概率是可忽略的。（仅通过猜测就找到原像总是可能的，其概率是可忽略的。类似地，通过蛮力搜索f的定义域在指数时间内找到原像总是可能的。）一个重要的技术要点是y的生成是通过在定义域中随机地选择元素x，然后令$y := f(x)$（而不是在值域中随机地选择y值）。这一点会在下面的形式化定义中变得清楚。

令$f : \{0,1\}^* \to \{0,1\}^*$为一个函数。考虑下面的实验，其定义对于任意算法$\mathcal{A}$和任意安全参数值$n$：

求逆实验$\mathsf{Invert}_{\mathcal{A},f}(n)$

(1) 选择输入$x \leftarrow \{0,1\}^n$，计算$y := f(x)$。

(2) 1^n和y值作为$\mathcal{A}$的输入，输出为x'。

(3) 如果$f(x') = y$，那么定义该实验的输出为 1，否则为 0。

这里强调$\mathcal{A}$没必要获得x本身，$\mathcal{A}$获取任何满足$f(x') = y = f(x)$的x'就足够。在第二步中赋予$\mathcal{A}$安全参数1^n，是因为计算原因：想让$\mathcal{A}$在以安全参数n表示的多项式时间内运行，而不考虑y的长度。

现在可以定义一个函数f是单向的是什么意思了。

定义 6.1 如果一个函数$f : \{0,1\}^* \to \{0,1\}^*$满足下述两个条件，那么它就是单向函数：

(1)（易于计算：）存在一个多项式时间算法M_f来计算f；也就是说，对所有x，有$M_f(x) = f(x)$。

(2)（反向求逆十分困难：）对任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leqslant \mathsf{negl}(n)$$

符号表示法。本章常常通过在概率表示中加上脚注来使概率空间（更加）明确。运用这种表示法，能非常简洁地将上述定义中的第二个条件表述如下：对于每一个概率多项式算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(f(x)) \in f^{-1}(f(x))] \leqslant \mathsf{negl}(n)$$

单向函数成功求逆。一个不是单向函数的函数不必要在所有时候（或者甚至是“经常”）都容易求逆。相反，定义 6.1 的第二个条件的反面说法就是存在概率多项式时间算法$\mathcal{A}$和不可忽略函数ε，满足求$f(x)$的逆的概率至少为$\varepsilon(n)$（这里概率来源于随机选取的$x \leftarrow \{0,1\}^n$）。这意味着反过来，存在一个正多项式$q(\cdot)$，满足对于无穷多个n值，算法$\mathcal{A}$求出f的逆的概率至少为$1/q(n)$。因此，如果存在算法$\mathcal{A}$求f的逆的概率为n^{-10}，这里为任意的偶数n（但是当n为奇数时，常不能求出f的逆），那么函数f就不是单向的。即使$\mathcal{A}$仅仅对一半的n能成功，和仅

以概率n^{-10}（对可成功的n）成功地求出f的逆，f仍不是单向的。

指数时间求逆。只要有足够长的时间，任何单向函数都能够求逆。具体地说，给定一个y值和安全参数1^n，可以简单地尝试所有$x \in \{0,1\}^n$，直到发现一个x值满足$f(x)=y$。这个算法在指数时间内运行，且总能成功。因而，单向函数的存在天生是关于“计算复杂性”和“计算困难性”的一种假设。这些理论考虑这样的问题：可以原则上解决，但是假设高效率地解决是困难的。

单向置换。通常对拥有额外的结构上的属性的单向函数感兴趣。如果对于所有的x值都有$|f(x)|=|x|$，就称该函数是“长度保留的”。如果一个单向函数既是长度保留的又是单射，就称它为单向置换。正式地：

定义 6.2 设函数$f:\{0,1\}^* \leftarrow \{0,1\}^*$为长度保留的，设$f_n$是定义域为$\{0,1\}^n$的$f$函数（也就是，$f_n$仅仅对$x \in \{0,1\}^n$有定义，这种情况下$f_n(x)=f(x)$）。如果对于每一个$n$，$f_n$为双射，则单向函数$f$被称为“单向置换”。

单向置换一个有趣的性质就是任意的y值都唯一确定其原象$x=f^{-1}(y)$。即使y完全确定x，也很难在多项式时间内找到x。

单向函数簇和单向置换簇。上面的对单向函数和单向置换的定义是非常方便的，这在于只考虑在一个有穷定义域和值域上定义的单一函数。然而，我们知道的大部分候选单向函数和置换并不是自然而然地适合这个定义框架。而是，典型的一个算法生成一些安全参数I，由I定义一些函数f_I；要求对于除了可忽略概率以外的所有I值，f_I必须具有单向性。因为每一个I值定义了一个不同的函数，现在引出单向函数簇（相应地，单向置换簇）的概念。在这里给出定义，在下一节中给出具体的实例（同时参见 7.4.1 节）。

定义 6.3 一个概率多项式时间算法的元组$\Pi=(\mathsf{Gen},\mathsf{Samp},f)$，如果满足下面的条件，那它就是函数簇：

(1) 参数生成算法Gen，输入为1^n，输出参数I，$|I| \geqslant n$。每个输出的I值定义两个集合D_I和R_I，分别构成如下定义的函数f_I的定义域和值域。

(2) 抽样算法Samp，输入I，输出一个均匀分布在集合$\mathcal{D}_I$中的一个元素（例外情况的概率为用$|I|$表示的可忽略概率）。

(3) 确定的评价算法f，输入I和$x \in \mathcal{D}_I$，输出元素$y \in \mathcal{R}_I$。记为$y:=f_I(x)$。

Π是一个置换簇，如果对每一个由$\mathsf{Gen}(1^n)$输出的I值，满足$\mathcal{D}_I=\mathcal{R}_I$，且函数$f_I:D_I \leftarrow D_I$是双射函数。

令Π是一个函数簇。接下来是前面介绍的实验的一种明显的类比。

求逆实验$\mathsf{Invert}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$得到I，然后计算$\mathsf{Samp}(I)$得到一个随机数$x \leftarrow \mathcal{D}_I$，最后计算$y:=f_I(x)$。

(2) $\mathcal{A}$的输入为I和y，输出为x'。

(3) 如果$f_I(x')=y$，定义本次实验的结果为 1，否则为 0。

如果上述实验以可忽略的概率成功，那么该函数簇是单向的。

定义 6.4 函数或置换簇$\Pi = (\mathsf{Gen}, \mathsf{Samp}, f)$是单向的，如果对于每一个概率多项式时间算法$\mathcal{A}$，存在可忽略函数negl，满足

$$\Pr[\mathsf{Invert}_{\mathcal{A},\Pi}(n) = 1] \leqslant \mathsf{negl}(n)$$

这里全章探讨遵照定义 6.1 和定义 6.2 的单向函数和置换，而不是单向函数簇。这主要是为了方便起见，且不会明显地影响任何结果。参见练习 6.6。

6.1.2 候选单向函数

仅当单向函数存在时，它才是让人感兴趣的。既然不知如何证明它无条件地存在（因为这意味着复杂性理论方面有一个大的突破），于是猜想或者假设它存在。这种猜想（或假设）是基于一些非常自然的计算问题，这些问题受到很多注意，且尚无多项式时间算法。可能其中最著名的就是整数分解，也就是寻找一个大整数的素数因子。这使得定义函数$f_{mult}(x, y) = x \cdot y$。（形式化地表述，输入长度为$n$的一个串，令$x$为前$\lfloor n/2 \rfloor$比特，令$y$为后$\lceil n/2 \rceil$比特；输出等于$xy$。）但是注意，如果对$x$和$y$的长度没有任何限制，那函数$f_{mult}$就容易求逆：设$x \cdot y$为偶数的概率较高，这种情况下$(2, xy/2)$两个因子就能够被返回作为函数的逆。有两种方法修改$f_{mult}$来解决这个问题。比较直接的方法就是输出结果时包括$x$和$y$（当视为整数时，忽略前面的 0）的长度；即定义

$$f_{mult}(x, y) = (xy, ||x||, ||y||)$$

另一个办法，如上所述，我们要求x和y的长度均恰为$n/2$。第二种方法本质上是限制函数f_{mult}的定义域为等长的素数x和y。第 7 章将讨论这种方法。

另一个候选单向函数不是直接依赖于数论，而是基于子集求和问题，定义为

$$f(x_1, \cdots, x_n, J) = (x_1, \cdots, x_n, \sum_{j \in J} x_j),$$

其中每个x_j为n比特串，解释为整数，J为n比特串，解释为集合$\{1, \cdots, n\}$的子集。给定该函数的一个输出$(x_1, \cdots, x_n, y)$，求逆的任务就是要找到一个子集$J' \subseteq \{1, \cdots, n\}$满足$\sum_{j \in J'} x_j = y$。学过$\mathcal{NP}$完全理论的读者可能熟悉一个事实：这个问题是$\mathcal{NP}$完全的。需要强调，这并不是说$\mathcal{P} \neq \mathcal{NP}$就意味着单向函数的存在：$\mathcal{P} \neq \mathcal{NP}$只是意味着在最坏情况下的子集求和问题在多项式时间内不能被解决，而对单向函数的要求求逆几乎总是困难的。因而，可以相信前述函数是单向的，这一信念是基于目前没有解决这个问题的算法的事实，不仅仅基于一般问题是$\mathcal{NP}$完全的这个事实。

通过给出一个被认为是“单向的”置换簇来总结。设Gen是一个概率多项式时间算法，输入1^n，输出一个n比特素数p，以及一个特殊元素$g \in \{2, \cdots, p-1\}$。（第 7 章中将看到这样的算法是存在的。现在不定义元素g是“特殊的”意味着什么。）设Samp是一个算法，输入p和g，输出一个$\{1, \cdots, p-1\}$中的随机整数。最后，定义

$$f_{p,g}(x) = g^x \bmod p$$

$f_{p,g}$能够高效地被计算出来的事实来自于附录 B.2.3 中的结果。能够证明该函数是单射，因而是一个置换。对该函数求逆的困难性假设是基于对离散对数问题困难性的猜测，第 7 章将进一步详细阐述。

6.1.3 硬核谓词

由定义可知，单向函数求逆是困难的。换句话说，给定一个值$y = f(x)$，x不能由任何多项式时间的算法确定。这看起来好像是说在多项式时间内不能得到关于x的任何信息。其实不是这样。实际上，函数$f(x)$可能“泄露”了许多关于x的信息，虽然函数仍然很难求逆。举个简单的例子，设f是单向函数，定义$g(x_1, x_2) = (x_1, f(x_2))$，其中$|x_1| = |x_2|$。很容易知道$g$也是一个单向函数（其证明留作练习），它“泄露”了输入的一半。

对于实际应用来说，需要找到一些被$f(x)$隐藏的关于x的信息。这就引出了“硬核谓词”的概念。大体上说，函数f的硬核谓词$hc : \{0,1\}^* \to \{0,1\}$有如下的性质：给定$f(x)$，对于任何多项式时间算法，能够以明显高于$1/2$的概率正确确定$hc(x)$是不可行的。（通过猜测能以$1/2$的概率正确地计算$hc(x)$，这总是可能的。）

定义 6.5 如果函数$hc : \{0,1\}^* \to \{0,1\}$满足下面的两个条件，那它就是函数$f$的硬核谓词：①给定$x$，$hc(x)$能够在多项式时间计算出来；②对于每一个概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}(f(x)) = hc(x)] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

其中概率来源于在$\{0,1\}^n$均匀选择x，以及$\mathcal{A}$的随机性。

需要强调，给定x，$hc(x)$是可高效率计算的（因为函数hc能够在多项式时间内被计算出来）；定义还要求给定$f(x)$要计算$hc(x)$是困难的。上面的定义没要求f是单向的，虽然我们只对单向函数的硬核谓词感兴趣。

简单的想法不可行。考虑一下，一个候选硬核谓词定义为$hc(x) = \bigoplus_{i=1}^{n} x_i$，其中$x_1, \cdots, x_n$表示$x$的比特位。直觉上认为该函数“应该”是一个硬核谓词是因为：如果函数f不能求逆，那么$f(x)$至少隐藏了其原象的某个比特位x_i。于是所有x的比特位的异或一定是难以计算的（因为x_i本身就很难计算）。不管多么有吸引力，这种想法是错误的。具体来说，给定一个单向函数f，定义函数$g(x) = (f(x), \bigoplus_{i=1}^{n} x_i)$。不难证明函数$g$是单向的。然而，显然$g(x)$没有隐藏$hc(x) = \bigoplus_{i=1}^{n} x_i$的值，因为它是输出的一部分；因此，$hc(x)$不总是硬核谓词。（扩展这个观点，对于任意给定的谓词hc，能够证明总是存在某个单向函数f，使得hc不是该函数的硬核谓词。）

平凡的硬核谓词。部分函数有“平凡的”硬核谓词。例如，令f为简单地去掉输入的最后一个比特位的函数（也就是说，$f(x_1, \cdots, x_n) = x_1 \cdots x_{n-1}$）。于是，既然$x_n$独立于输出，给定$f(x) = x_1 \cdots x_{n-1}$不能预测$x_n$的值。然而，上述函数$f$却不是单向的。在运用硬核谓词构造伪随机发生器的时候，就会清楚：为什么这种平凡的硬核谓词在密码学领域一点用都没有。

相比之下，有硬核谓词的单射函数f必定是单向的（参见练习 6.10）。在直觉上，这是因为当函数是单射时，$f(x)$完全确定了x（从信息论意义上讲）。因此，不能从$f(x)$计算$hc(x)$必然是因为从$f(x)$来确定x的计算限制。

6.2 概述：从单向函数到伪随机置换

本章的目的是介绍如何基于任意单向置换，构造伪随机发生器、伪随机函数以及伪随机置换。本节将对这些构造方法给出一个概述，在后面几节中将给出详细介绍。

任意单向函数的硬核谓词。第一步要证明对于任意的单向函数都存在一个硬核谓词。实际上，该陈述是否正确目前仍是一个公开问题；我们将证明稍弱的对达到目的已经足够的命题。即，我们将证明给定任意单向函数f，能构造一个“不同的”单向函数g以及函数g的一个硬核谓词。也就是说：

定理 6.6 设函数f为单向函数。则必定（构造性地）存在一个单向函数g以及函数g的一个硬核谓词gl。另外，如果f是单向置换，那么g也是单向置换。

（在 Goldreich 和 Levin 两人证明了定理 6.6 之后，硬核谓词就用gl来表示。）函数g及其硬核谓词gl构建如下：令$g(x,r) \stackrel{\mathsf{def}}{=} (f(x),r)$，其中$|x| = |r|$。定义：

$$\mathsf{gl}(x,r) \stackrel{\mathsf{def}}{=} \bigoplus_{i=1}^{n} x_i \cdot r_i$$

其中$x = x_1 \cdots x_n$（r也类似）。注意，函数$\mathtt{gl}(x,\cdot)$输出的是x的比特位的随机子集的异或值。这是因为r可以视为从$\{1,\cdots,n\}$中选择一个随机的子集（即，当$r_i = 1$时，x_i包含于XOR中，否则就不包含），且r是均匀分布的。因此，定理 6.6 实质上就是陈述如果函数f是任意单向函数，那么$f(x)$隐藏了x的比特位的一个随机子集的异或值。

从单向置换到伪随机发生器。下一步就是要阐述如何用单向置换的硬核谓词来构造伪随机发生器。（单向函数足够用来构造伪随机发生器，但是证明极其复杂，超出本书讨论的范围。）具体阐述如下：

定理 6.7 f是单向置换，$\mathtt{hc}$是它的一个硬核谓词，则$G(s) \stackrel{\mathsf{def}}{=} (f(s),\mathtt{hc}(s))$构成了伪随机发生器，其扩展系数$l(n) = n+1$。

之所以定理中定义的G为伪随机发生器，首先注意当s是均匀随机选择的，则$G(s)$的输出的前n个比特位（即$f(s)$的比特位）必定是“真正”随机的，这是因为f是单向置换。其次，hc是一个硬核谓词意味着$\mathsf{hc}(s)$“看起来是随机的”——也就是伪随机——即使给定$f(s)$（再次强调s是随机选择的）。综合上述观察，我们就知道G的整个输出都是伪随机的。

拥有任意扩展系数的伪随机发生器。仅仅扩展了种子的一个比特（正如刚才看到的）的伪随机发生器的存在已经是一个了不起的事。但是为了实际应用（例如，高效地加密像 3.4 节中那样的大消息），需要一个大得多的扩展系数。幸运的是，我们能够得到（基本上）想要的扩展系数：

定理 6.8 假设有一个伪随机发生器，其扩展系数为$l(n) = n+1$。则对于任意多项式

$p(\cdot)$，存在一个拥有扩展系数为$l(n)=p(n)$的伪随机发生器。

能够得出结论，拥有（基本上）任意扩展系数的伪随机发生器可以从任意单向置换构造。

从伪随机发生器到伪随机函数和伪随机置换。对于得到窃听者存在条件下的不可区分加密的对称密钥加密方案，伪随机发生器已经足够。然而，为了获得CPA安全的对称密钥加密（更不用说消息鉴别码），要依靠伪随机函数。下面的结果阐述了可以用伪随机发生器构造伪随机函数：

定理 6.9 假定存在扩展系数为$l(n)=2n$伪随机发生器，那么存在伪随机函数。

实际上，甚至能够得到更多：

定理 6.10 假定存在伪随机函数，那么存在强伪随机置换。

综合考虑上面的定理以及第 3 章、第 4 章的结论，得出如下推论：

推论 6.11 假定存在单向置换，那么存在拥有任意多项式扩展系数的伪随机发生器、伪随机函数以及强伪随机置换。

推论 6.12 假定存在单向置换，那么存在CCA安全的对称密钥加密方案，以及适应性选择消息攻击条件下的存在性不可伪造的消息鉴别码。

正如之前提到的，实际上可以仅仅基于单向函数的存在性得到所有这些结果。

6.3 任意单向函数的硬核谓词

本节我们来证明定理 6.6。

定理 6.13 f是单向函数，定义函数g为$g(x,r)\stackrel{\text{def}}{=}(f(x),r)$，其中$|x|=|r|$；定义$\mathsf{gl}(x,r)\stackrel{\text{def}}{=}\bigoplus_{i=0}^{n}x_i\cdot r_i$，其中$x=x_1\cdots x_n$，$r=r_1\cdots r_n$，那么gl就是函数$g$的一个硬核谓词。

6.3.1 简单情形

首先证明如果存在一个多项式时间敌手$\mathcal{A}$，总是能够在给定$g(x,r)=(f(x),r)$的情况下正确地计算出$\mathsf{gl}(x,r)$，那么就有可能在多项式时间内求函数f的逆。既然假定函数f是单向函数，则这样的$\mathcal{A}$不存在。

命题 6.14 设函数f和gl如定理 6.13 所定义。如果存在一个概率多项式时间算法$\mathcal{A}$，对于无穷多的n值，都有

$$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathsf{gl}(x,r)]=1$$

那么存在另外一个概率多项式时间算法$\mathcal{A}'$，对于无穷多的n值，都满足

$$\Pr_{x\leftarrow\{0,1\}^n}[\mathcal{A}'(f(x))\in f^{-1}(f(x))]=1。$$

证明 设$\mathcal{A}$为命题中所定义的。构造$\mathcal{A}'$如下。输入一个y值，$|y|=n$，敌手$\mathcal{A}'$计算$x_i := \mathcal{A}(y, e^i)$, $i=1,\cdots,n$，其中e^i表示在第i位上为 1，其他位上为 0 的n比特串。然后$\mathcal{A}'$输出$x = x_1 \cdots x_n$。显然$\mathcal{A}'$运行在多项式时间内。

为了分析$\mathcal{A}'$成功计算f的逆的概率。在等式

$$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r) = \mathsf{gl}(x,r)] = 1 \tag{6.1}$$

中，固定n值，考虑执行$A'(y)$。令$y=f(\hat{x})$。那么由$\mathcal{A}'$计算出的x_i满足

$$x_i = \mathcal{A}(f(\hat{x}), e^i) = \bigoplus_{j=1}^{n} \hat{x}_j \cdot e_j = \hat{x}_i$$

对第二个等式应用$\mathsf{gl}(x,r)$的定义，以及事实：第三个等式来自对$j\neq i$，有$e^i_j = 0$。因此，对于所有的i都有$x_i = \hat{x}_i$，因而$\mathcal{A}'$以 1 的概率输出正确的逆$x=\hat{x}$。 ■

根据f是单向函数的假设，任何概率多项式时间的算法以不可忽略的概率求f的逆是不可能的。因此得出结论：不存在这样的概率多项式时间算法，它能够对无穷多的n值，从$(f(x),r)$正确计算出$\mathsf{gl}(x,r)$。这是一个相对较弱的结果，其与证明确定$\mathsf{gl}(x,r)$的概率不会明显高于$1/2$的最终目标相去甚远。

6.3.2 复杂情形

现在证明任何多项式时间算法$\mathcal{A}$以明显高于$3/4$的概率计算出$\mathsf{gl}(x,r)$是困难的。假设有这样的$\mathcal{A}$存在，将再次证明这意味着存在一个多项式时间的$\mathcal{A}'$能以不可忽略的概率求出f的逆。注意，在对命题 6.14 的证明中的策略在这里完全不能用，因为当$r=e^i$时有可能$\mathcal{A}$根本就不能成功（虽然可能对所有其他的r值成功）。此外，在当前的情形中，$\mathcal{A}'$不知道一个特殊的被$\mathcal{A}$输出的比特作为对$\mathsf{gl}(x,r)$的猜测是否正确。$\mathcal{A}'$唯一知道的是，敌手$\mathcal{A}$能以高出$3/4$的概率正确，且高出$3/4$的部分是不可忽略的。这些问题使证明更加复杂。

命题 6.15 设f和gl如定理 6.13 中所定义。如果存在一个概率多项式时间算法$\mathcal{A}$和一个多项式$p(\cdot)$，对于无穷多的n值，满足

$$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r) = \mathsf{gl}(x,r)] \geqslant \frac{3}{4} + \frac{1}{p(n)}$$

那么存在一个概率多项式时间算法A'，对于无穷多的n值，满足

$$\Pr_{x\leftarrow\{0,1\}^n}[\mathcal{A}'(f(x)\in f^{-1}(f(x))] \geqslant \frac{1}{4\cdot p(n)}$$

证明 对该命题的证明过程的观察是：对于每一个$r\in\{0,1\}^n$，结合$\mathsf{gl}(x,r\oplus e^i)$和$\mathsf{gl}(x,r)$的值可以推出$x$的第$i$个比特位。（还记得前面曾经提到，$e^i$表示除了第$i$位外，其余位全是 0 的$n$比特串。）这源于下面的计算。

$$\begin{aligned}&\mathsf{gl}(x,r)\oplus\mathsf{gl}(x,r\oplus e^i)\\ &= (\bigoplus_{j=1}^{n} x_j\cdot r_j)\oplus(\bigoplus_{j=1}^{n} x_j\cdot(r_j\oplus e^i_j)) = x_i\cdot r_i\oplus(x_i\cdot(r_i\oplus 1)) = x_i,\end{aligned}$$

其中，第二个等式成立是因为当$j\neq i$时，$x_j\cdot r_j$的值同时出现在两个和中，故相互抵消了。

上面的证明表明，如果$\mathcal{A}$能同时正确地对$(f(x),r)$和$(f(x),r\oplus e^i)$进行回答，那么$\mathcal{A}'$就能

正确地计算出x_i。可惜的是$\mathcal{A}'$并不知道$\mathcal{A}$何时能正确地回答，何时不能；仅仅知道$\mathcal{A}$能以“高”概率正确地回答。正是这个原因，$\mathcal{A}'$将使用多个随机的r值，使用每一个r值去猜测一个x_i，然后选择出现次数最多的x_i作为最终的x_i值。作为一个基本步骤，证明对于很多的x，只要r是均匀随机选择的，那么$\mathcal{A}$能同时正确地对$(f(x),r)$和$(f(x),r\oplus e^i)$进行回答的概率是足够大的。这一点将在下面的声称中加以证明。这些声称将允许我们固定x值，仅仅关注均匀选择的r，这样做使分析变得更容易。

声称 6.16 令n满足

$$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]\geqslant\frac{3}{4}+\frac{1}{p(n)}$$

那么存在一个集合$S_n\subseteq\{0,1\}^n$，其大小至少为$\dfrac{1}{2p(n)}\cdot 2^n$，并且满足：对于任意的$x\in S_n$都有

$$\Pr_{r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]\geqslant\frac{3}{4}+\frac{1}{2p(n)}\tag{6.2}$$

证明 令$\varepsilon(n)=1/p(n)$，对于任意的$x\in\{0,1\}^n$，设

$$s(x)\stackrel{\mathrm{def}}{=\!=}\Pr_{r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]。$$

令S_n为所有x的集合，有$s(x)\geqslant 3/4+\varepsilon(n)/2$（即式（6.2）满足）。如果$S_n=\{0,1\}^n$，已经得到证明。否则，使用一个简单的平均参数来说明$|S_n|\geqslant\frac{\varepsilon(n)}{2}\cdot 2^n$。有

$$\begin{aligned}&\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]\\&=\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)|x\in S_n]\cdot\Pr_x[x\in S_n]\\&\quad+\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)|\,x\notin S_n]\cdot\Pr_x[x\notin S_n]\\&\leqslant\Pr_x[x\in S_n]+\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)|\,x\notin S_n]\end{aligned}$$

其中非下标变量为定值时，下标变量（即x与/或r）就表示是从$\{0,1\}^n$随机选择。因此：

$$\Pr_x[x\in S_n]\geqslant\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]-\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)|\,x\notin S_n]$$

通过定义S_n，对$x\notin S_n$时，$\Pr[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]<3/4+\varepsilon(n)/2$。也就是说，$\Pr_{x,r}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)|x\notin S_n]<3/4+\varepsilon(n)/2$，从而，

$$\Pr_x[x\in S_n]\geqslant\frac{3}{4}+\varepsilon(n)-(\frac{3}{4}+\frac{\varepsilon(n)}{2})=\frac{\varepsilon(n)}{2}$$

这就意味着S_n的大小至少为$\frac{\varepsilon(n)}{2}\cdot 2^n$（因为$x$是均匀分布在$\{0,1\}^n$中的），证明完毕。 ■

下面就是需要的结果，现在作为一个简单的推论。

声称 6.17 设n满足

$$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]\geqslant\frac{3}{4}+\frac{1}{p(n)}$$

那么存在一个集合$S_n \subseteq \{0,1\}^n$，其大小至少为$\frac{1}{2p(n)}\,2^n$，满足：对于每一个$x \in S_n$和每一个i都有

$$\Pr_{r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)\wedge\mathcal{A}(f(x),r\oplus e^i)=\mathrm{gl}(x,r\oplus e^i)]\geqslant\frac{1}{2}+\frac{1}{p(n)}$$

证明 设$\varepsilon(n)=1/p(n)$，选取集合S_n为声称6.17中所保证的集合。对于任意的$x\in S_n$都有

$$\Pr_{r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)\neq\mathrm{gl}(x,r)]\leqslant\frac{1}{4}-\varepsilon(n)/2$$

固定任意的$i\in\{1,\cdots,n\}$。如果r是均匀分布的，那么$r\oplus e^i$也是均匀分布的；这就意味着

$$\Pr_{r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r\oplus e^i)\neq\mathrm{gl}(x,r\oplus e^i)]\leqslant\frac{1}{4}-\varepsilon(n)/2$$

我们对$\mathcal{A}$对$\mathrm{gl}(x,r)$和$\mathrm{gl}(x,r\oplus e^i)$输出正确回答的概率下界感兴趣；换句话说，想知道$\mathcal{A}$对这两种情况都不能输出正确回答的概率上界。注意，$r$和$r\oplus e^i$不是独立的，所以不能简单地把失败的概率相乘。然而，可以应用一致限（参见附录A的命题A.7），并将失败的概率加在一起。也就是说，$\mathcal{A}$不能正确回答$\mathrm{gl}(x,r)$或者$\mathrm{gl}(x,r\oplus e^i)$的概率最大为

$$(\frac{1}{4}-\frac{\varepsilon(n)}{2})+(\frac{1}{4}-\frac{\varepsilon(n)}{2})=\frac{1}{2}-\varepsilon(n)$$

也就是$\mathcal{A}$能对$\mathrm{gl}(x,r)$和$\mathrm{gl}(x,r\oplus e^i)$正确回答的概率至少为$1/2+\varepsilon(n)$，得证。 ■

余下部分，令$\varepsilon(n)=1/p(n)$，且只考虑那些能让$\mathcal{A}$以至少为$3/4+\varepsilon(n)$的概率成功的n值。上面的声称阐述了：对于输入的x值中的$\varepsilon(n)/2$部分，敌手$\mathcal{A}$对$(f(x),r)$和$(f(x),r\oplus e^i)$以至少$1/2+\varepsilon(n)$的概率（对随机选择的r）成功地回答，从现在开始仅仅关注这些x值。构造一个概率多项式时间算法$\mathcal{A}'$，当$x\in S_n$时，$\mathcal{A}'$能以至少为$1/2$的概率求出$f(x)$的逆。这足够证明命题，因为

$$\begin{aligned}&\Pr_x[\mathcal{A}'(f(x))\in f^{-1}(f(x))]\\&\quad\geqslant\Pr_x[\mathcal{A}'(f(x))\in f^{-1}(f(x))|\ x\in S_n]\cdot\Pr_x[x\in S_n]\\&\quad\geqslant\frac{1}{2}\cdot\frac{1}{2p(n)}=\frac{1}{4p(n)}\end{aligned}$$

给定一个元素y作为输入，算法$\mathcal{A}'$工作如下：

(1) For i=1,⋯,n do:

(a) 随机地选择一个$r\leftarrow\{0,1\}^n$，然后计算一个“猜测”，其中$\overline{x}_i:=\mathcal{A}(y,r)\oplus\mathcal{A}(y,r\oplus e_i)$是$y$的原象的第$i$个比特位。

(b) 重复上一步充分多次（参见下面），令x_i为猜测中的多数。

(2) 输出$x=x_1,\cdots,x_n$。

勾勒一下$\mathcal{A}'$正确求出给定输入值y的逆的概率的分析（允许更简洁一点，因为对最困难情况的完整证明在接下来的小节中给出）。固定一个n值（无穷多的值中的一个），满足$\Pr_{x,r\leftarrow\{0,1\}^n}[\mathcal{A}(f(x),r)=\mathrm{gl}(x,r)]\geqslant\frac{3}{4}+\frac{1}{p(n)}$，假定算法$\mathcal{A}'$的输入$y=f(\hat{x})$，其中$\hat{x}\in S_n$（前面曾经提到过后者发生的概率至少为$\varepsilon(n)/2$）。固定某个$i$值。前面的声称意味着猜测$\overline{x}_i$等于$\mathrm{gl}(\hat{x},e^i)$的概率至少为$1/2+\varepsilon(n)$。通过重复充分多次，且让出现次数最多的$\overline{x}_i$作为最终的$x_i$，算法$\mathcal{A}'$能确保$x_i$等于$\mathrm{gl}(\hat{x},e^i)$的概率至少为$1-\frac{1}{2n}$。需要确保这能够通过取多项式次猜测

的多数情况来完成；因为对某个多项式p，有$\epsilon(n)=1/p(n)$，这的确可以使用$Chernoff$界（一个来自于概率论的标准界），以及在每次迭代中选择独立的值r这一事实。使用$Chernoff$界进行完整的证明留作练习。

总结当前的结果，对于每一个i，$\mathcal{A}'$计算出的x_i是错误的概率最多为$\frac{1}{2n}$。一致限表明对于某些i，$\mathcal{A}'$错误$\mathsf{gl}(\hat{x},e^i)$的概率最多为$n\cdot\frac{1}{2n}=\frac{1}{2}$。也就是说，对于所有的$i$，$\mathcal{A}'$正确的概率（即正确求出$y$的逆的概率）至少为$\frac{1}{2}$。命题得证。 ■

命题 6.15 的一个推论是如果f是一个单向函数，那么当给定$(f(x),r)$时，正确猜测出$\mathsf{gl}(x,r)$的概率的最大值大于3/4的部分是可忽略的。因此，比特$\mathsf{gl}(x,r)$有很大的不确定性（当考虑到多项式时间的观察者时）。

6.3.3 完整证明

这一节比书中其他部分的内容要更深一些，需更为复杂的概率论知识。为了内容的完整，和高年级学生课程的需要，这里给出完整的证明过程。

预备知识：抽样概率。这里先快速回顾一下概率论，一些概率论中标准的结论将用到。一个0/1随机变量X_i就是从$\{0,1\}$取的一个值。0/1随机变量$X_1,\cdots,X_m$是两两独立的，如果对于任意的$i\neq j$，每一对$b_i,b_j\in\{0,1\}$都有

$$\Pr[X_i=b_i\wedge X_j=b_j]=\Pr[X_i=b_i]\cdot\Pr[X_j=b_j]$$

依赖下面这个命题：

命题 6.18 设$\{X_i\}$两两独立，这些0/1随机变量有如下性质：存在$b\in\{0,1\}$和$\varepsilon>0$，满足对于所有的$1\leqslant i\leqslant n$都有

$$\Pr[X_i=b_i]=\frac{1}{2}+\varepsilon$$

考虑一个过程：过程中m个值$X_1,\cdots,X_m$被记录，令X为其中出现次数最多的那个值，那么

$$\Pr[X\neq b]\leqslant\frac{1}{4\cdot\varepsilon^2\cdot m}$$

该命题是一个标准结果。如果读者愿意接受该命题，可直接继续学习后面的部分；考虑到完整性，这里提供一个对该命题的证明。

设$\mathsf{Exp}[X]$表示随机变量X的期望。有：

（1）马尔可夫不等式：设X是非负的随机变量，且$v>0$，则

$$\Pr[X\geqslant v]\leqslant\mathsf{Exp}[X]/v$$

证明 有

$$\begin{aligned}\mathsf{Exp}[X]&=\sum_{x\geqslant 0}\Pr[X=x]\cdot x\\&\geqslant\sum_{0\leqslant x\leqslant v}\Pr[X=x]\cdot 0+\sum_{x\geqslant v}\Pr[X=x]\cdot v\\&=\Pr[X\geqslant v]\cdot v\end{aligned}$$

■

当关于X的信息知道得非常少时，马尔可夫不等式是非常有用的。当知道了X的方差的上界时，存在更好的界。将用到下面来自概率论的基本事实：$\mathsf{Var}[X] \stackrel{\text{def}}{=} \mathsf{Exp}[(X-\mathsf{Exp}[x])^2]$，$\mathsf{Var}[X]=\mathsf{Exp}[X^2]-\mathsf{Exp}[X]^2$以及$\mathsf{Var}[aX+b]=a^2\mathsf{Var}[X]$。

（2）切比雪夫不等式：设X是随机变量，且$\delta>0$，那么：

$$\Pr[|X-\mathsf{Exp}[X]|\geqslant\delta]\leqslant\frac{\mathsf{Var}[X]}{\delta^2}$$

证明 定义非负随机变量$Y\stackrel{\text{def}}{=}(X-\mathsf{Exp}[X])^2$，然后运用马尔可夫不等式。即

$$\begin{aligned}\Pr[|X-\mathsf{Exp}[X]|\geqslant\delta]&\leqslant\Pr[|(X-\mathsf{Exp}[X])^2\geqslant\delta^2]\\&\leqslant\frac{\mathsf{Exp}[(X-\mathsf{Exp}[X])^2]}{\delta^2}\\&=\frac{\mathsf{Var}[X]}{\delta^2}\end{aligned}$$

■

如果$X_1,\cdots,X_m$是两两独立的，那么$\mathsf{Var}[\sum_{i=1}^m X_i]=\sum_{i=1}^m\mathsf{Var}[X_i]$（这是因为当$i\neq j$时，由于两两独立，有$\mathsf{Exp}[X_i\cdot X_j]=\mathsf{Exp}[X_i]\cdot\mathsf{Exp}[X_j]$）。切比雪夫不等式的一个重要推论如下。

推论 6.19 设$X_1,\cdots,X_m$是两两独立的随机变量，并有相同的期望μ和相同的方差σ^2。对每一个$\varepsilon>0$，有，

$$\Pr[|\frac{\sum_{i=1}^m X_i}{m}-u|\geqslant\varepsilon]\leqslant\frac{\sigma^2}{\varepsilon^2 m}$$

证明 由期望的线性性质，$\mathsf{Exp}[\sum_{i=1}^m X_i/m]=\mu$。对随机变量$\sum_{i=1}^m X_i/m$运用切比雪夫不等式，可得到

$$\Pr[|\frac{\sum_{i=1}^m X_i}{m}-u|\geqslant\varepsilon]\leqslant\frac{\mathsf{Var}[\frac{1}{m}\cdot\sum_{i=1}^m X_i]}{\varepsilon^2}$$

运用两两独立性，有

$$\mathsf{Var}[\frac{1}{m}\cdot\sum_{i=1}^m X_i]=\frac{1}{m_2}\sum_{i=1}^m\mathsf{Var}|\ X_i|=\frac{1}{m_2}\sum_{i=1}^m\sigma_2=\frac{\sigma_2}{m}$$

综合以上两式，不等式得证。 ■

现在证明命题 6.18，令$b=1$（由对称性可知，这个选择是不相关的）；这就意味着$\mathsf{Exp}[X_i]=\frac{1}{2}+\varepsilon$。设$X$表示命题中$\{X_i\}$的多数值，注意$X\neq 1$当且仅当$\sum_{i=1}^m X_i\leqslant m/2$。所以

$$\begin{aligned}\Pr[X\neq 1]&\leqslant\Pr[\sum_{i=1}^m X_i\leqslant m/2]\\&=\Pr[\frac{\sum_{i=1}^m X_i}{m}-\frac{1}{2}\leqslant 0]\\&=\Pr[\frac{\sum_{i=1}^m X_i}{m}-(\frac{1}{2}+\varepsilon)\leqslant-\varepsilon]\\&\leqslant\Pr[|\frac{\sum_{i=1}^m X_i}{m}-(\frac{1}{2}+\varepsilon)|\geqslant\varepsilon]。\end{aligned}$$

对于0/1随机变量X_i，有$\sigma^2 = \mathsf{Var}[X_i] \leqslant 1/4$（这是因为在这种情形下$\mathsf{Exp}[X_i] = \mathsf{Exp}[X_i^2]$，所以$\mathsf{Var}[X_i] = \mathsf{Exp}[X_i](1 - \mathsf{Exp}[X_i])$，当$\mathsf{Exp}[X_i] = \frac{1}{2}$时，取最大值）。运用前面的推论，就能得出结论

$$\Pr[X \neq 1] \leqslant \frac{1}{4\varepsilon^2 m}$$

得证。

定理 6.13 的证明。假设熟悉前面简化的证明，并基于那里介绍的思想。证明如下命题，该命题推出定理 6.13：

命题 6.20 设f和gl如定理 6.13 所述。存在一个概率多项式时间算法$\mathcal{A}$和一个多项式$p(\cdot)$满足：对于无穷多的n，有

$$\Pr_{x,r \leftarrow \{0,1\}^n}[\mathcal{A}(f(x), r) = \mathsf{gl}(x, r)] \geqslant \frac{1}{2} + \frac{1}{p(n)}$$

那么一定存在一个概率多项式时间敌手A'和一个多项式$p'(\cdot)$，对于无穷多的n值都有

$$\Pr_{x \leftarrow \{0,1\}^n}[\mathcal{A}'(f(x)) \in f^{-1}(f(x))] \geqslant \frac{1}{p'(n)}$$

证明 正如在命题 6.15 中，令$\varepsilon(n) = 1/p(n)$，只考虑那些能让$\mathcal{A}$成功的概率为$1/2 + \varepsilon(n)$的n值。下面的证明和声称 6.16 相似，证明过程相同。

声称 6.21 设n满足

$$\Pr_{x,r \leftarrow \{0,1\}^n}[\mathcal{A}(f(x), r) = \mathsf{gl}(x, r)] \geqslant \frac{1}{2} + \varepsilon(n)$$

那么存在大小至少为$\frac{1}{2p(n)}\, 2^n$的集合$S_n \subseteq \{0,1\}^n$，满足：对于每一个$x \in S_n$，有

$$s(x) \stackrel{\mathsf{def}}{=} \Pr_{r \leftarrow \{0,1\}^n}[\mathcal{A}(f(x), r) = \mathsf{gl}(x, r)] \geqslant \frac{1}{2} + \varepsilon(n) \tag{6.3}$$

如果想试着跟命题 6.15 的证明过程完全一样地进行下去，将会遇到困难，因为与声称 6.17 的类比不会满足。具体地说，这里能够声称的最佳结果是：当$x \in S_n$时，对于任意的i，有

$$\Pr_{r \leftarrow \{0,1\}^n}[\mathcal{A}(f(x), r) = \mathsf{gl}(x, r) \wedge \mathcal{A}(f(x), r \oplus e^i) = \mathsf{gl}(x, r \oplus e^i)] \geqslant \frac{1}{p(n)}$$

这意味着如果试着去构造一个算法$\mathcal{A}'$，通过计算$\mathcal{A}(f(x), r) \oplus \mathcal{A}(f(x), r \oplus e^i)$来猜测$x_i$的值，那么所有能声称的是：这个猜测将以至少是$1/p(n)$的概率正确，这甚至还不如直接随机地猜测。（而且，也不能声称：取结果的相反值能以高概率得到一个好的猜测。）

取而代之，设计$\mathcal{A}'$，通过只调用一次$\mathcal{A}$来计算$\mathsf{gl}(x, r)$和$\mathsf{gl}(x, r \oplus e^i)$。为达到该目的，让$\mathcal{A}'$运行$\mathcal{A}(x, r \oplus e^i)$，并简单地“猜测”$\mathsf{gl}(x, r)$的值。一种作这个的简单方法是独立地选择$r$的值，如前所述，对每一个$r$值独立地猜测一次$gl(x, r)$的值。但是，如果$\mathcal{A}'$将要输出正确的回答，正如将看到的，则所有的猜测值都要正确，这种情况的概率是可忽略的，因为有多项式个不同的r被使用。

对当前证明的一个关键的观察就是：$\mathcal{A}'$能用两两独立的方式生成r，以一种特别的方式进行猜测，使得所有猜测都正确的概率为不可忽略的概率。具体地说，为了生成m个不同的r

值，$\mathcal{A}'$独立地选择$l=\lceil log(m+1)\rceil$和均匀分布的串$s^1,\cdots,s^l\in\{0,1\}^n$。对于每一个非空子集$I\in\{1,\cdots,l\}$，算法$\mathcal{A}'$令$r^I=\oplus_{i\in I}s^i$。既然存在$2^l-1$个非空子集，这就定义了$2^{\lceil log(m+1)\rceil}-1\geqslant m$个不同的串。孤立地考虑时，每一个这样的串都是均匀分布的。而且，这些串是两两独立的。为了说明这一点，注意：对于任意的两个子集$I\neq J$，存在指数$j\in I\cup J$且$j\notin I\cap J$。不失一般性，加设$j\in J$。则即使已知一部分r^I的值，任何关于值s^j的信息都不会被泄露，因而s^j仍然是均匀分布的。此外，因为s^j包含于定义r^J的XOR中，所以即使知道了部分r^I的值，r^J也是均匀分布的。

于是有下面两个重要的观察结果：

(1) 给定正确的$\mathsf{gl}(x,s^1),\cdots,\mathsf{gl}(x,s^l)$，对于任何非空子集$I\subseteq\{1,\cdots,l\}$，可能正确地计算出$\mathsf{gl}(x,r^I)$。这是因为

$$\mathsf{gl}(x,r^I)=\mathsf{gl}(x,\oplus_{i\in I}s^i)=\oplus_{i\in I}\mathsf{gl}(x,s^i)。$$

(2) $\mathsf{gl}(x,s^1),\cdots,\mathsf{gl}(x,s^l)$的值都被猜对的概率为$1/2^l$。这是因为每个比特位$\mathsf{gl}(x,s^i)$被猜对的概率为$1/2$，而一共有$l$个比特位。如果$m$是关于安全参数$n$的多项式，那么$2^l$也是关于$n$的多项式。因此，能够正确地猜测出$\mathsf{gl}(x,s^1),\cdots,\mathsf{gl}(x,s^l)$的值的概率是不可忽略的。

结合上面的两个结果，就产生了一个方法，该方法能获得$m=\mathsf{poly}(n)$个两两独立的串$\{r^I\}$，以及对于所有的I，获取正确的$\{\mathsf{gl}(x,r^I)\}$的值的概率是不可忽略的。这些值又可以用来计算x_i，跟证明命题 6.15 的方法相同。具体如下：

求逆算法$\mathcal{A}'$。现在给出算法$\mathcal{A}'$的完整描述，该算法的输入为y，并试图求出其逆。算法过程如下：

(1) 设$n:=|y|$，$l:=\lceil\log(2n/\varepsilon(n)^2+1)\rceil$。

(2) 均匀随机地选择$s^1,\cdots,s^l\in\{0,1\}^n$和$\sigma^1,\cdots,\sigma^l\in\{0,1\}^n$。

(3) 对每一个非空子集$I\in\{1,\cdots,l\}$，设$r^I:=\oplus_{i\in I}s^i$，计算$\rho^I:=\oplus_{i\in I}\sigma^i$。

(4) 对$i=1,\cdots,n$：

① 对于每一个非空子集$I\subseteq\{1,\cdots,l\}$，设$x_i^I:=\rho^I\oplus\mathcal{A}(y,r^I\oplus e^i)$。

② 令$x_i:=\mathsf{majority}_I\{x_i^I\}$（即，取前面步骤中出现次数最多的比特）。

(5) 输出$x=x_1\cdots x_n$。

分析算法$\mathcal{A}'$的成功概率。剩下的工作是计算$\mathcal{A}'$正确输出$x\in f^{-1}(y)$的概率。跟命题 6.15 的证明过程类似，集中考虑$y=f(\hat{x})$的情形，其中$\hat{x}\in S_n$。每个σ^i都可以看成是$\mathsf{gl}(\hat{x},s^i)$的一个“猜测值”。正如之前所说，能正确地猜测所有这些值的概率是不可忽略的；下面说明如果这样的话，$\mathcal{A}'$输出$x=\hat{x}$的概率至少为$1/2$。这样就可以完成证明。

对于所有i，假设$\sigma^i=\mathsf{gl}(\hat{x},s^i)$，每个$\rho^i$等于$\mathsf{gl}(\hat{x},r^I)$。固定指数$i\in\{1,\cdots,n\}$，并且考虑$\mathcal{A}'$得到正确$x_i$值的概率。因为$\mathcal{A}(y,r^I\oplus e^i)=\mathsf{gl}(\hat{x},r^I\oplus e^i)$的概率至少为$\frac{1}{2}+\varepsilon(n)/2$(这是因为$\hat{x}\in S_n$，以及孤立地考虑，$r^I\oplus e^i$是均匀分布的串)，易知，对于所有$I$，

$$\Pr[x_i^I=\hat{x}_i]=\Pr[\mathcal{A}(f(\hat{x}),r^I\oplus e^i)=\mathsf{gl}(\hat{x},r^I\oplus e^i)]\geqslant\frac{1}{2}+\frac{\varepsilon(n)}{2}$$

而且，$\{x_i^I\}_{I\subseteq\{1,\cdots,l\}}$是两两独立的，因为$\{r^I\}_{I\subseteq\{1,\cdots,l\}}$（从而$\{r^I\oplus e^i\}_{I\subseteq\{1,\cdots,l\}}$）是两两独立的。因为$x_i$定义为$\{\hat{x}_i^I\}$中发生次数最多的值，现在可以应用命题 6.18。设$m=2^l-1$，有

$$\Pr[x_i \neq \hat{x}_i] \leqslant \frac{1}{4 \cdot (\varepsilon(n)/2)^2 \cdot (2^l - 1)}$$
$$\leqslant \frac{1}{4 \cdot (\varepsilon(n)/2)^2 \cdot (2n/\varepsilon(n)^2)}$$
$$= \frac{1}{2n}$$

对所有i，上式均成立，因此通过应用一致限，证明对于某些i，$x_i \neq \hat{x}_i$的概率至多是$1/2$。也就是说，对于所有i，$x_i = \hat{x}_i$（因此$x = \hat{x}$）的概率至少是$1/2$。

综上所述：$y = f(\hat{x})$，$\hat{x} \in S_n$成立的概率至少为$\varepsilon(n)/2$。独立于这一事件，所有的猜测σ_i是正确的概率至少为

$$\frac{1}{2^l} \geqslant \frac{1}{2 \cdot (2n/\varepsilon(n)^2 + 1)} > \frac{\varepsilon(n)^2}{5n}$$

（n足够大时最后的不等式成立）。在以上满足的条件下，$\mathcal{A}'$输出一个y的逆的概率至少为$1/2$。因此对无穷多个n值，$\mathcal{A}'$对其输入y求逆的总概率至少为$\varepsilon(n)^3/20n = 1/20 \cdot n \cdot p(n)^3$。因为$\mathcal{A}'$是多项式时间运行，这与$f$的单向性矛盾。 ■

6.4 构造伪随机发生器

首先指出在假设单向置换存在的条件下，怎样构造伪随机发生器来把输入延长一个比特位。然后指出怎样扩充该方法来获得任意多项式（大小）的扩展系数。

6.4.1 具有最小扩展的伪随机发生器

令f是一个单向置换，hc是f的核心谓词（该核心谓词在定理 6.13 中出现）。构造的出发点是给定一个随机数s的$f(s)$，使得很难以大于$1/2$的概率（即大于$1/2$的部分是不可忽略的概率）猜出$\mathsf{hc}(s)$的值。因而从直觉上感觉$\mathsf{hc}(s)$是一个伪随机比特。此外，因为f是一个置换，$f(s)$是均匀分布的（把一个置换应用到一个均匀分布的值产生一个均匀分布的值）。因此，得出结论：串$(f(s), \mathsf{hc}(s))$是伪随机的，所以算法$G(s) = (f(s), \mathsf{hc}(s))$构成一个伪随机发生器。

定理 6.22 令f是一个单向函数，令hc是f的硬核谓词。那么，算法$G(s) = (f(s), \mathsf{hc}(s))$是$l(n) = 1 + n$的伪随机发生器。

证明 令D是一个概率多项式区分器，并设

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr_{s \leftarrow \{0,1\}^n}[D(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+1}}[D(r) = 1]$$
$$= \Pr_{s \leftarrow \{0,1\}^n}[D(f(s), \mathsf{hc}(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{n+1}}[D(r) = 1]$$

注意到

$$\Pr_{r \leftarrow \{0,1\}^{n+1}}[D(r) = 1] = \Pr_{r \leftarrow \{0,1\}^n, r' \leftarrow \{0,1\}}[D(r, r') = 1]$$
$$= \Pr_{s \leftarrow \{0,1\}^n, r' \leftarrow \{0,1\}}[D(f(s), r') = 1]$$
$$= \frac{1}{2} \cdot \Pr_{s \leftarrow \{0,1\}^n}[D(f(s), \mathsf{hc}(s)) = 1] +$$
$$\frac{1}{2} \cdot \Pr_{s \leftarrow \{0,1\}^n}[D(f(s), \overline{\mathsf{hc}}(s)) = 1]$$

利用f是一个置换的事实得到第一个等式，由于随机比特位r'与$hc(s)$相等的概率恰为$1/2$，得到第二个等式。因此，

$$\varepsilon(n)=\frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\mathsf{hc}(s))=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\overline{hc}(s))=1]\right)$$

考虑下面的算法$\mathcal{A}$，输入一个值$y=f(s)$，并且尝试预测$\mathsf{hc}(s)$的值：

(1) 均匀随机地选取$r'\leftarrow\{0,1\}$。

(2) 运行$D(y,r')$，如果D输出 1，则输出r'；否则，输出$1-r'$。

根据$\mathcal{A}$的定义有

$$\begin{aligned}
&\Pr_{s\leftarrow\{0,1\}^n}[\mathcal{A}(f(s))=\mathsf{hc}(s)]\\
&\quad=\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[\mathcal{A}(f(s))=\mathsf{hc}(s)|r'=\mathsf{hc}(s)]\\
&\qquad+\frac{1}{2}\cdot\Pr_{s\leftarrow\{0,1\}^n}[\mathcal{A}(f(s))=\mathsf{hc}(s)|\ r'\neq\mathsf{hc}(s)]\\
&\quad=\frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\mathsf{hc}(s))=1]+\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\overline{\mathsf{hc}}(s))=0]\right)\\
&\quad=\frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\mathsf{hc}(s))=1]+\left(1-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\overline{\mathsf{hc}}(s))=1]\right)\right)\\
&\quad=\frac{1}{2}+\frac{1}{2}\cdot\left(\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\mathsf{hc}(s))=1]-\Pr_{s\leftarrow\{0,1\}^n}[D(f(s),\overline{\mathsf{hc}}(s))=1]\right)\\
&\quad=\frac{1}{2}+\varepsilon(n)
\end{aligned}$$

显然算法$\mathcal{A}$为多项式运行时间。因为hc是f的硬核谓词，所以存在可忽略函数negl，满足$\varepsilon(n)\leqslant\mathsf{negl}(n)$。

类似地，$-\varepsilon(n)\leqslant\mathsf{negl}(n)$。综上所述，有

$$\left|\Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1]-\Pr_{r\leftarrow\{0,1\}^{n+1}}[D(r)=1]\right|\leqslant\mathsf{negl}(n),$$

G为伪随机发生器，得证。 ■

6.4.2 提高扩展系数

现在说明伪随机发生器的扩展系数能够提高到任意多项式数量。这表明前面的构造方法（扩展系数为$l(n)=n+1$）足够用来构造一个具有任意多项式扩展系数的伪随机发生器。

定理 6.23 如果存在扩展系数为$l(n)=n+1$的伪随机发生器$\hat{G}$，那么对于任意多项式$p(n)>n$，存在扩展系数为$l(n)=p(n)$的伪随机发生器G。

证明 用$\hat{G}$构造G的思想如下。给定一个长度为n的初始种子密钥s，发生器$\hat{G}$用于得到$n+1$个伪随机位。输出$n+1$中的某一位，余下的n位再次用作$\hat{G}$的种子。余下的n位能够用作种子的原因是因为它们是伪随机的，所以实质上如同真正的随机种子。重复迭代这个过程可用于输出如期望那么多的比特位，如图 6.1 所示。

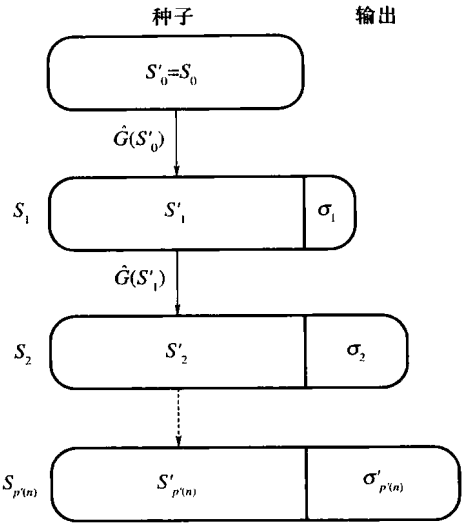


图 6.1　提高伪随机发生器的扩展系数

现在正式描述G的构造方法。输入$s \in \{0,1\}^n$：

(1) 令$p'(n) = p(n) - n$。注意这是所期望的G对输入长度的扩展量。

(2) 设$s_0 := s$。对于$i = 1, \cdots, p'(n)$，则：

①令s'_{i-1}表示s_{i-1}的开始n比特，令σ_{i-1}表示剩下的$n-1$比特。（当$i=1$时，σ_0为空串。）

②设$s_i := (\hat{G}(s'_{i-1}), \sigma_{i-1})$。

(3) 输出$s_{p'(n)}$。

步骤开始前，要注意当$i=1$时，s'_0是初始种子，在步骤（2）②中有$s_1 = \hat{G}(s'_0)$。接着当$i=2$时，长度为$n+1$比特的串s_1分成前n位记为s'_1，和后一位记为σ_1。串s'_1再次用作$\hat{G}$的种子，最后生成长度为$n+2$比特的串s_2（也就是$(\hat{G}(s'_1), \sigma_1)$）。观察迭代的结果，串$s_2$的最后两个比特变成$\sigma_2$（$\sigma_2$的第一位是$\hat{G}(s'_1)$的最后一位，$\sigma_2$的第二位是$\sigma_1$）。因此，每次迭代仅增加一个比特，加入到“$\sigma$部分”。由于这个原因，$\sigma_i$在每次迭代后长度增加一位。

证明$G(s)$是长度为$p(n)$的伪随机串，从证明$p(n) = n+2$的简单情况开始。

定义三个分布序列$\{H_n^0\}$，$\{H_n^1\}$，$\{H_n^2\}$，每一个H_n^0，H_n^1，H_n^2都是长度为$n+2$的串的分布。在分布H_n^0中，串$s_0 \leftarrow \{0,1\}^n$是均匀随机选取的，且输出是$G(s_0)$。在分布H_n^1中，串$s_1 \leftarrow \{0,1\}^{n+1}$是均匀随机选取的，然后$G$如上述方法运行，但从$i=2$开始迭代。也就是说，解析$s_1$为$(s'_1, \sigma_1)$，$|s'_1| = n$，然后输出$(\hat{G}(s'_1), \sigma_1)$。在分布$H_n^2$中，均匀随机地选取串$s_2 \leftarrow \{0,1\}^{n+2}$并输出。用$s_2 \leftarrow H_n^i$表示$(n+2)$比特的串$s_2$的选择是根据$H_n^i$的分布。

首先声称：对于任意概率多项式时间的区分器D，存在一个可忽略的函数negl，满足

$$\left| \Pr_{s_2 \leftarrow H_n^0}[D(s_2) = 1] - \Pr_{s_2 \leftarrow H_n^1}[D(s_2) = 1] \right| \leqslant \mathrm{negl}(n) \tag{6.4}$$

为理解上式，确定某个D并考虑多项式时间的区分器D'，对输入$w \in \{0,1\}^{n+1}$，设$s_1 := w$，然后如上述方法运行G，但从$i=2$开始迭代。这样产生一个串s_2，于是D'输出$D(s_2)$。下述观察结果直接来源于H_n^0和H_n^1的定义：

(1) 如果均匀随机地选取w，由D'生成的s_2的分布正好是H_n^1的分布。因此，

$$\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1]=\Pr_{s_2\leftarrow H_n^1}[D(s_2)=1]。$$

(2) 如果$s\leftarrow\{0,1\}^n$是均匀随机选取的，$w=\hat{G}(s)$，由D'生成的s_2的分布正好是H_n^0的分布。也就是说，

$$\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1]=\Pr_{s_2\leftarrow H_n^0}[D(s_2)=1]$$

$\hat{G}$的伪随机性表明存在一个可忽略函数negl，使得

$$|\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1]-\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1]|\leqslant \mathsf{negl}(n)。$$

得到式(6.4)。

我们然后声称：对任意概率多项式时间的区分器D，存在一个可忽略函数negl，满足

$$|\Pr_{s_2\leftarrow H_n^1}[D(s_2)=1]-\Pr_{s_2\leftarrow H_n^2}[D(s_2)=1]|\leqslant \mathsf{negl}(n)。\tag{6.5}$$

证明是类似的。考虑多项式时间区分器D'，对输入$w\in\{0,1\}^{n+1}$，均匀随机选取$\sigma_1\leftarrow\{0,1\}$，设$s_2:=(w,\sigma_1)$，并输出$D(s_2)$。注意如果$w$是均匀随机选取的，那么$s_2$是均匀分布的，因而正好与$H_n^2$的分布一致。于是，

$$\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1]=\Pr_{s_2\leftarrow H_n^2}[D(s_2)=1]$$

另一方面，如果$s\leftarrow\{0,1\}^n$是均匀随机选取的，$w=\hat{G}(s)$，那么s_2的分布和H_n^1完全一样，所以，

$$\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1]=\Pr_{s_2\leftarrow H_n^1}[D(s_2)=1]$$

如前所述，$\hat{G}$的伪随机性推出式(6.5)。

固定某个概率多项式时间区分器D'，有

$$\begin{aligned}&|\Pr_{s\leftarrow\{0,1\}^n}[D'(G(s))=1]-\Pr_{r\leftarrow\{0,1\}^{n+2}}[D(r)=1]|\\&=|\Pr_{s_2\leftarrow H_n^0}[D(s_2)=1]-\Pr_{s_2\leftarrow H_n^2}[D(s_2)=1]|\\&\leqslant|\Pr_{s_2\leftarrow H_n^0}[D(s_2)=1]-\Pr_{s_2\leftarrow H_n^1}[D(s_2)=1]|\\&\quad+|\Pr_{s_2\leftarrow H_n^1}[D(s_2)=1]-\Pr_{s_2\leftarrow H_n^2}[D(s_2)=1]|\end{aligned}\tag{6.6}$$

利用式(6.4)和式(6.5)推出式(6.6)是可忽略的。

完全的证明。对于任意的$p(n)$现在给出证明。这里主要的区别是技术上的：对$p(n)=n+2$的情况，可以定义并且显式地工作在三个分布序列$\{H_n^0\}$，$\{H_n^1\}$和$\{H_n^2\}$上。相比之下，这里将（在某种意义上）需要处理无穷多个分布序列。与这里显式的处理方式不同，用被称为“混合参数”的一般技术来隐式地处理它们。（实际上，即使在$p(n)=2$的情况下，也使用简单的混合参数。）

令$p'(n)=p(n)-n$。对于任意的n和$0\leqslant j\leqslant p'(n)$，令$H_n^j$是在长度为$p(n)$的串上分布，定义如下：均匀随机地选取$s_j\leftarrow\{0,1\}^{n+j}$，然后从迭代$i=j+1$开始运行$G$并输出$s_{p'(n)}$。（当$j=p'(n)$时简单地随机均匀选取$s_{p'(n)}\leftarrow\{0,1\}^{p(n)}$并输出。）这里关键要注意的是$H_n^0$对应于输出$G(s)$，$s\leftarrow\{0,1\}^n$是均匀随机选取的，而$H_n^{p'(n)}$对应于输出一个均匀随机选取的$p(n)$比特串。固定任意多项式时间的区分器$D$，这意味着

$$\varepsilon(n) \stackrel{\text{def}}{=} \left| \Pr_{s\leftarrow\{0,1\}^n}[D(G(s))=1] - \Pr_{r\leftarrow\{0,1\}^{p(n)}}[D(r)=1] \right|$$

$$= \left| \Pr_{s_{p'(n)}\leftarrow H_n^0}[D(s_{p'(n)})=1] - \Pr_{s_{p'(n)}\leftarrow H_n^{p'(n)}}[D(s_{p'(n)})=1] \right| \tag{6.7}$$

的目标是证明ε是可忽略的，表明G是伪随机发生器。

如上所述固定D，当给定一个串$w\in\{0,1\}^{n+1}$作为输入时，考虑区分器D'执行如下操作：

(1) 均匀随机选取$j\leftarrow\{1,\cdots,p'(n)\}$。

(2) 均匀随机选取$\sigma_j\leftarrow\{0,1\}^{j-1}$。

(3) 设$s_j:=(w,\sigma_j)$。然后从迭代$i=j+1$开始运行G，且计算$s_{p'(n)}$，输出$D(s_{p'(n)})$。

分析D'的行为比前面的更加复杂，虽然基本思想是相同的。固定n，同时假定D'选取$j=J$。如果$w\leftarrow\{0,1\}^{n+1}$是均匀随机选取的，那么s_J是均匀分布的，因此$s_{p'(n)}$的分布此时正好是分布H_n^J。即

$$\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1 \mid j=J] = \Pr_{s_{p'(n)}\leftarrow H_n^J}[D(s_{p'(n)})=1]$$

因为每一个j的值是以相同概率选取，则

$$\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1] = \frac{1}{p'(n)}\cdot\sum_{J=1}^{p'(n)}\Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1 \mid j=J]$$

$$= \frac{1}{p'(n)}\cdot\sum_{J=1}^{p'(n)}\Pr_{s_{p'(n)}\leftarrow H_n^J}[D(s_{p'(n)})=1] \tag{6.8}$$

另一方面，假定D'选取$j=J$，且$w=\hat{G}(s)$，$s\leftarrow\{0,1\}^n$是均匀随机选取的。设$s_{J-1}=(s,\sigma_J)$，可以看到s_{J-1}是均匀分布的，因此$s_{p'(n)}$的分布此时正好是H_n^{J-1}的分布，即

$$\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1 \mid j=J] = \Pr_{s_{p'(n)}\leftarrow H_n^{J-1}}[D(s_{p'(n)})=1]$$

于是

$$\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1] = \frac{1}{p'(n)}\cdot\sum_{J=1}^{p'(n)}\Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1 \mid j=J]$$

$$= \frac{1}{p'(n)}\cdot\sum_{J=1}^{p'(n)}\Pr_{s_{p'(n)}\leftarrow H_n^{J-1}}[D(s_{p'(n)})=1]$$

$$= \frac{1}{p'(n)}\cdot\sum_{J=0}^{p'(n)-1}\Pr_{s_{p'(n)}\leftarrow H_n^{J}}[D(s_{p'(n)})=1] \tag{6.9}$$

（注意在最后一步求和的索引被跳过。）现在分析D'从随机数中区分$\hat{G}$的输出程度：

$$\left| \Pr_{s\leftarrow\{0,1\}^n}[D'(\hat{G}(s))=1] - \Pr_{w\leftarrow\{0,1\}^{n+1}}[D'(w)=1] \right|$$

$$= \frac{1}{p'(n)}\cdot\left| \sum_{J=0}^{p'(n)-1}\Pr_{s_{p'(n)}\leftarrow H_n^J}[D(s_{p'(n)})=1] - \sum_{J=1}^{p'(n)}\Pr_{s_{p'(n)}\leftarrow H_n^J}[D(s_{p'(n)})=1] \right|$$

$$= \frac{1}{p'(n)}\cdot\left| \Pr_{s_{p'(n)}\leftarrow H_n^0}[D(s_{p'(n)})=1] - \Pr_{s_{p'(n)}\leftarrow H_n^{p'(n)}}[D(s_{p'(n)})=1] \right|$$

$$= \frac{\varepsilon(n)}{p'(n)}$$

根据式(6.8)和式(6.9)，得到第一个等式，根据式(6.7)得到最后一个等式。（第二个等式是因为在每个求和中包含相同的项，除了左边总和的第一项和右边总和的最后一项。）因为$\hat{G}$是伪随机发生器，所以D'在多项式时间运行，并且p'是多项式。从而得出结论ε是可忽略的。 ■

混合技术。混合技术用于许多安全证明中，是当一个基本原语被应用多次时证明不可区分性的基本工具。这个技术通过定义一系列连接两个“极限分布”的混合分布来起作用，这些分布是期望证明不可区分的分布（在前面的证明中，这分别对应于G的输出和一个随机串）。为应用这个证明方法，应具备三个条件。首先，极限混合应与原始感兴趣的情况匹配（在前面的证明中，这意味着H_n^0与由G产生的分布相同，而$H_n^{p'(n)}$与均匀分布相同）。第二，必须有可能把区分相邻混合分布的能力转化为打破某个基本假设的能力（上述区分H_n^i和H_n^{i+1}与区分$\hat{G}$的输出和随机数本质上是相同的）。最后，混合数应该是多项式的，因此“区分成功”仅仅减少了（乘以）多项式系数（的程度）。

任意扩展系数的显式发生器。令f是具有硬核谓词hc的单向置换。通过结合定理 6.22 的构造方法（叙述了$G'(s) = (f(s), hc(s))$是伪随机发生器）和定理 6.23 的证明，得到

$$G(s) = (f^{p'(n)}(s), \mathsf{hc}(f^{p'(n)-1}(s)), \cdots, \mathsf{hc}(s))$$

是扩展系数为$p(n) = n + p'(n)$的伪随机发生器。这个发生器称为 Blum-Micali 发生器。

现代流密码设计。许多现代流密码工作是通过保持伪随机的内部状态，大多与以上描述的G的构造方法相同。发生器的每一次迭代，都会输出一些伪随机比特并且更新内部状态。定理 6.23（和图 6.1）所描述的构造方法就是作这样工作的，除了对流密码而言只输出σ_i（而且特别是$s'_{p'(n)}$不输出），以便能够达到无上限的扩展。证明过程验证了流密码的这一设计原则是正确的，其在某些条件下可用于得到一种可证明安全的构造方法。

6.5 构造伪随机函数

在说明怎样由单向置换构造伪随机发生器之后，继续说明怎样由伪随机发生器构造伪随机函数。如同 3.6.1 节的定义，伪随机函数是可高效计算的密钥函数，对任意多项式区分器而言是看似随机的；还记得这个区分器可访问真正随机函数或者伪随机函数的预言机。

通过下述简短的例子作为完整构造方法的出发点。令G是扩展系数为$l(n) = 2n$的伪随机发生器（也就是G是双倍长度），同时表示成$G(s) = (G_0(s), G_1(s))$，其中$s = |G_0(s)| = |G_1(s)| = n$。也就是说，种子长度为$n$，$G_0(s)$表示$G(s)$输出的前半部分，而$G_1(s)$表示输出的后半部分。我们此时用$G$构造密钥函数，使用一个$n$比特的密钥$k$，取单独一个比特作为输入，输出长度为$n$的串。对于密钥$k$，定义：

$$F_k(0) \stackrel{\text{def}}{=} G_0(k) \quad \text{和} \quad F_k(1) \stackrel{\text{def}}{=} G_1(k)。$$

我们说这个函数是伪随机的[②]。这是因为G是伪随机发生器：一个随机函数通过定义表将 1 比

② 我们对伪随机函数的形式化定义（定义 3.23）假设一个定义域和值域为$\{0,1\}^*$的长度保留函数。定义明显可以扩展到有任意定义域和值域的伪随机函数上。

特映射成n比特，该表有两个n比特值，每个都是随机选取的。这里定义一个密钥函数，基本上通过一个有两个n比特值的表，每个值都是伪随机的。因此，F_k（其中k是随机选取的）不能由任意多项式时间算法从随机函数中区分出来。

进行进一步的构造，定义一个输入为 2 比特的伪随机函数。对于密钥k，定义：

$$F_k(00) \stackrel{\text{def}}{=} G_0(G_0(k)) \quad F_k(10) \stackrel{\text{def}}{=} G_0(G_1(k))$$

$$F_k(01) \stackrel{\text{def}}{=} G_1(G_0(k)) \quad F_k(11) \stackrel{\text{def}}{=} G_1(G_1(k))$$

要求四个串$G_0(G_0(k))$、$G_0(G_1(k))$、$G_1(G_0(k))$和$G_1(G_1(k))$是伪随机的。（如上所述，这足够证明函数F_k是伪随机的。）实际上，考虑下面的混合分布：

$$G_0(k_0),\ G_1(k_0),\ G_0(k_1),\ G_1(k_1)$$

其中$k_0, k_1 \leftarrow \{0,1\}^n$是独立的、均匀分布的串。在这个混合分布中，随机串$k_0$代替$G_0(k)$，随机串$k_1$代替$G_1(k)$。现在，如果可能从基本分布中区分出混合分布，那么就能够区分伪随机串$G(k) = (G_0(k), G_1(k))$和真正的随机串(k_1, k_2)，与G的伪随机性相反。同样地，如果可能从长度为$4n$的真正随机串中区分出混合分布，那么就能从长度为$2n$的真正随机串中区分出$G(k_0) = (G_0(k_0), G_1(k_0))$，或者从长度为$2n$的真正随机串中区分出$G(k_1) = (G_0(k_1), G_1(k_1))$。这再次与$G$的伪随机性相反。综上所述，得到这四个串是伪随机的，因此所定义的函数也是伪随机的。这个事实的形式化证明留作练习。

将上述构造方法一般化，得到输入为n比特的伪随机函数，定义如下

$$F_k(x) = G_{x_n}(\cdots G_{x_1}(k)\cdots)$$

其中$x = x_1,\cdots,x_n$，参见构造方法 6.24。这个函数是伪随机的这种直觉与前面相同，但是此时正式的证明是复杂的，因为要考虑指数多个值（即，从$F_k(0\cdots0)$到$F_k(1\cdots1)$）。

构造方法 6.24

令G是扩展系数为$l(n) = 2n$的伪随机发生器。用$G_0(k)$表示G的输出的前半部分，用$G_1(k)$表示G的输出的后半部分。对于每一个$k \in \{0,1\}^n$，定义函数$F_k : \{0,1\}^n \to \{0,1\}^n$为：

$$F_k(x_1x_2\cdots x_n) = G_{x_n}(\cdots(G_{x_2}(G_{x_1}(k)))\cdots)。$$

一个由伪随机发生器产生的伪随机函数

这个构造方法可以看作是深度为n的完全二叉树，定义如下（如图 6.2 所示）。根的值等于密钥k。然后，对于任意节点k'的值，k'的左边孩子节点的值为$G_0(k')$，右边孩子节点的值为$G_1(k')$。因此，$x = x_1\cdots x_n$时，$F_k(x)$的值与叶子节点的值相等，叶子节点的位置是根据x的值来遍历整个树得到的（也就是说，$x_i = 0$表明“向左走”，而$x_i = 1$表明“向右走”）。强调这个函数只对长度为n的输入有定义，因此只有叶子节点的值会被输出。树的大小是用n表示的指数函数，特别是有2^n个叶子节点。然而，计算函数$F_k(x)$从不需要构造和存储整个树，而只需计算从根到有用叶子节点路径上的值。

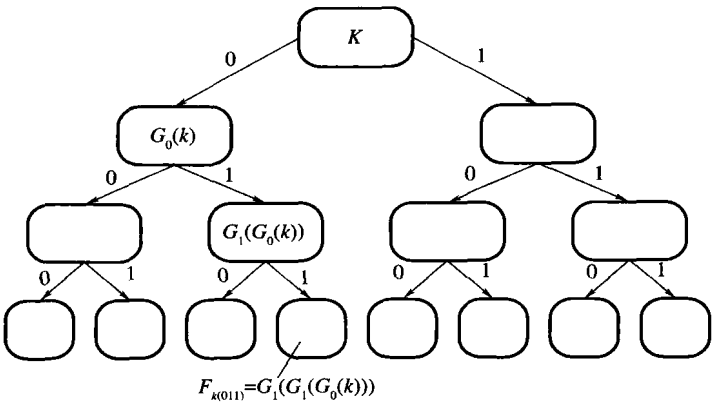


图 6.2　构造伪随机函数

定理 6.25　如果G是扩展系数为$l(n)=2n$的伪随机发生器，那么构造方法 6.24 是伪随机函数。

证明（概要）　这个证明可用作混合技术的另一个例子。令D是概率多项式时间区分器，给定1^n，同时可以访问一个预言器，该预言器要么是来自随机选择的函数f（f将n比特串映射成n比特串），或者来自函数F_k（k为一个随机选取的密钥）。

在二叉树上定义一个分布序列。通过关联任意给定的深度为n的二叉树上叶子节点的值与长度为n的串（见图 6.2），等同地认为这些是函数上的分布。令H_n^i，$0 \leqslant i \leqslant n$表示深度为$n$的二叉树上的分布：第$i$层节点的值是从$\{0,1\}^n$中独立地均匀随机选取的；在$j \geqslant i+1$层任意节点的值由这个节点双亲的值$k'$确定，而且若这个节点是左边孩子，则设其值等于$G_0(k')$；若这个节点是右边孩子，则设其值等于$G_1(k')$。（注意把这看作一个函数，从0层到$i-1$层节点的值是不相关的。）

注意H_n^n能将n比特串映射成n比特串，是真正的随机函数，因为所有叶子节点（即深度为n的节点）的值都是均匀地独立地随机选取的。另一方面，H_n^0正好是有均匀选取的密钥的构造方法 6.24，因为只有根节点（在第 0 层）是随机选取的，而且树中每一个其他节点的值是关于根节点值的确定函数。

利用证明定理 6.23 中的混合参数技术，如果多项式时间区分器D能够从真正的随机函数中以不可忽略的概率区分出构造 6.24，那么必有值i，有不可忽略的概率从H_n^{i+1}中区分出H_n^i。可以用这个方法从随机发生器中区分伪随机发生器。从直觉上感到是这样的，因为相邻的混合分布H_n^i和H_n^{i+1}的区别仅仅是：在分布H_n^{i+1}上伪随机发生器G用于树从根节点到叶子节点路径上的额外层。实际证明技巧性更强，因为不能保存树的整个第$(i+1)$层（大小可能是指数级）。令$t(n)$是能够区分构造方法 6.24 和随机函数的区分器D的最大运行时间。D使用最多进行$t(n)$次预言机问询。现在，令D'是G的区分器，得到$2n \cdot t(n)$长的输入，D'或者是真正随机的或者是$t(n)$次的$G(s)$调用，每次的s都是独立随机的。（虽然这里不作证明，但不难说明所有采样一起构成长度为$2n \cdot t(n)$的伪随机串。）于是，D'随机选取$i \leftarrow \{0,\cdots,n-1\}$，并且回答$D'$的预言机问询如下，开始保存一个空的二叉树。接收到来自D的问询$x=x_1\cdots x_n$，区分器D'使用$x_1\cdots x_i$到达某个第i层的节点。于是，D'以其一个输入进行采样（长度为$2n$），用采样的前半部分标记到达节点的左边孩子，用采样的后半部分标记到达节点的右边孩子。D'按照构造方法 6.24 继续计算输出。注意在以后的问询中，如果输入x使D'到达一个已经赋值

的节点，那么D'的回答始终是已存在的那个值。否则，D'从输入中使用一个新的采样。（注意D'根据其问询同态地填充树。这是因为整个树太大不能完全存储。）

得到一个重要的观察结果如下：

(1) 如果D'得到一个长度为$2n \cdot t(n)$的真正的随机串，那么会根据分布H_n^{i+1}回答D'。这是因为D'（动态）构造的树中$i+1$层所有的值都是随机的。

(2) 如果D'得到伪随机输入（即$t(n)$次对$G(s)$的调用中，s每次都是独立的值），那么根据分布H_n^i回答D'。因为$i+1$层的值是伪随机的且是由G产生的，完全根据定义所述。（D'不知道这些伪随机值的种子，但这对结果没有影响。）

利用混合分析证明定理 6.23。我们看到如果D以概率$\varepsilon(n)$区分构造方法 6.24 和真正的随机函数，那么D'则以概率$\varepsilon(n)/n$区分$G(s)$的$t(n)$次调用和长为$2n \cdot t(n)$的真正随机串。如果$\varepsilon(n)$不可忽略，则这与假设G是伪随机发生器矛盾。 ■

6.6 构造（强）伪随机置换

本节说明如此从伪随机函数构造伪随机置换和强伪随机置换。回顾 3.6.3 节，伪随机置换是一个可高效地求逆的伪随机函数F（这表明对于任意密钥k，F_k是一个置换），然而从随机置换中区分一个“强”伪随机置换困难更大，即使敌手可以访问置换及其逆的随机预言机。

Feistel网络回顾。 Feistel网络在 5.2 节有过介绍，是一种由不可逆操作来构造可逆函数的方法。在某种意义上，这正是这里期望做的。Feistel网络是一系列（几轮）的操作。把第i轮的输入看作一个长度为$2n$的串，然后等分为两个n比特的部分L_{i-1}和R_{i-1}（分别是“左半部分”和“右半部分”）。第i轮的输出是$2n$比特的串(L_i, R_i)，其中

$$L_i := R_{i-1} \quad \text{和} \quad R_i := L_{i-1} \oplus f_i(R_{i-1})$$

这里f_i是某个可高效计算的（但不必是可逆的）轮函数，将n比特输入映射成n比特输出。

用$\mathsf{Feistel}_{f_1,\cdots,f_r}$表示使用轮函数$f_1,\cdots,f_r$的$r$轮Feistel网络。（即$\mathsf{Feistel}_{f_1,\cdots,f_r}(L_0,R_0)$输出$2n$的串$(L_r,R_r)$。）在 5.2 节中对于每个$r$值，不管轮函数$\{f_i\}$，$\mathsf{Feistel}_{f_1,\cdots,f_r}$都是一个高效可逆置换。特别地，这表明$F$是一个伪随机函数，于是$F^{(1)}$定义

$$F_k^{(1)}(x) \stackrel{\text{def}}{=} \mathsf{Feistel}_{F_k}(x)$$

是有密钥的置换。（注意$k \in \{0,1\}^n$时$F_k^{(1)}$是$\{0,1\}^{2n}$上的置换。）然而，$F^{(1)}$是伪随机的吗？

稍微注意一下可以确定$F^{(1)}$不是伪随机的。对任意的密钥$k \in \{0,1\}^n$，$F_k^{(1)}$的输出的前n比特（即L_1）等于输入的后n比特（即R_0），但对随机函数而言，发生这种情况的概率是可忽略的概率。继续这个方法，我们能够定义一个有密钥的随机置换$F^{(2)}: \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}^{2n}$如下：

$$F_{k_1,k_2}^{(2)}(x) \stackrel{\text{def}}{=} \mathsf{Feistel}_{F_{k_1},F_{k_2}}(x) \tag{6.10}$$

（注意是k_1和k_2独立的密钥。）不幸的是，$F^{(2)}$也不是伪随机的。这一点在练习 6.18 中被要求给出证明。

综上所述，得到三轮Feistel网络是伪随机的，这一结果可能有点令人惊讶。也就是说，定义有密钥的函数$F^{(3)}$，其密钥长度为$3n$，且能将$2n$比特的输入映射成$2n$比特的输出，如下所示：

$$F^{(3)}_{k_1,k_2,k_3}(x) \stackrel{\text{def}}{=} \text{Feistel}_{F_{k_1},F_{k_2},F_{k_3}}(x) \tag{6.11}$$

其中，k_1, k_2和k_3依旧是独立选取的。可证明如下结果：

定理 6.26 如果F是保留长度的伪随机函数，那么$F^{(3)}$是能够将$2n$比特串映射成$2n$比特串的伪随机置换（利用一个长度为$3n$比特的密钥）。

$F^{(3)}$不是强伪随机的（练习 6.19 中要求给出其证明）。幸亏增加一次第四轮迭代就能够产生一个强伪随机置换。详细描述见构造方法 6.27，同时参见图 6.3。

构造方法 6.27

令F为长度可保留的带密钥的函数。定义带密钥的置换$F^{(4)}$如下：

- 输入：密钥$k \in \{0,1\}^{4n}$解析为$k = (k_1, k_2, k_3, k_4)$，其中$|k_i| = n$，输入的$x \in \{0,1\}^{2n}$解析为(L_0, R_0)，其中$|L_0| = |R_0| = n$。
- 计算：

（1）计算 $L_1 := R_0$和$R_1 := L_0 \oplus F_{k_1}(R_0)$。

（2）计算 $L_2 := R_1$和$R_2 := L_1 \oplus F_{k_2}(R_1)$。

（3）计算 $L_3 := R_2$和$R_3 := L_2 \oplus F_{k_3}(R_2)$。

（4）计算 $L_4 := R_3$ 和 $R_4 := L_3 \oplus F_{k_4}(R_3)$。

（5）输出 (L_4, R_4)。

由任意伪随机函数产生的强伪随机置换

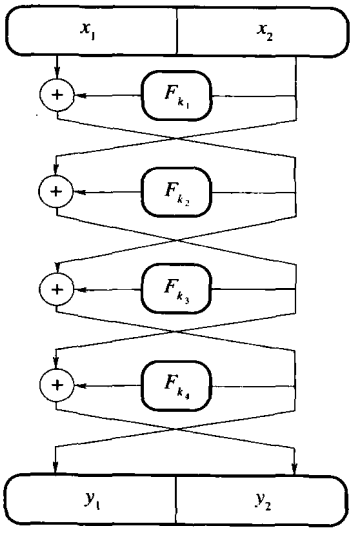


图 6.3 四轮Feistel网络，用于从伪随机函数构造强伪随机置换

定理 6.28 如果F是长度可保留的伪随机函数，那么构造方法 6.27 是能够将$2n$比特串映射成$2n$比特串的强伪随机置换（利用一个长度为$4n$比特的密钥）。

定理 6.26 和定理 6.28 的证明是技术性的，这里省略，感兴趣的读者可参考文献[64]。

6.7 对称密钥密码学的必要假设

总结目前为止在这一章得到了什么：

(1) 如果存在单向置换，那么就存在伪随机发生器。

(2) 如果存在伪随机发生器，那么就存在伪随机函数。

(3) 如果存在伪随机函数，那么就存在（强）伪随机置换。

因此，假设存在单向置换就能得到伪随机发生器和伪随机置换。实际上，由任意单向函数构造伪随机发生器是可能的，虽然这里我们不作证明。不管怎样，我们得到如下基本定理：

定理 6.29 如果存在单向函数，那么就存在伪随机发生器、伪随机函数和强伪随机置换。

在第 3 章和第 4 章研究过的所有对称密钥方案都能由伪随机发生器和伪随机函数构造。因此有

定理 6.30 如果存在单向函数，则存在加密方案具有选择密文攻击下不可区分加密，以及存在消息鉴别码具有选择消息攻击下存在性不可伪造。

非正式地说，单向函数是所有对称密钥密码学的充分条件。对于此，可能疑惑是否单向函数也是必要条件。在本节余下部分，指出情况确实如此。

伪随机性意味着单向函数。 首先指出伪随机发生器的存在性表明单向函数的存在性：

命题 6.31 如果存在伪随机发生器，则存在单向函数。

证明 令G是扩展系数为$l(n)=2n$的伪随机发生器。（根据定理 6.23，知道伪随机发生器的存在表明这种扩展系数的伪随机发生器存在。）G本身是单向的。其高效计算性是显而易见的（因为G能够在多项式时间内计算）。求G的逆的能力可以转化为区分G的输出和随机数的能力。直觉上说，这是成立的，因为求G的逆的能力意味着发现了发生器的种子。

令$\mathcal{A}$是概率多项式时间算法，并且定义

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Invert}_{\mathcal{A},G}(n)=1]$$

（比较定义 6.1。）构造下列运行在多项式时间的区分器D：输入串$w\in\{0,1\}^{2n}$，运行$\mathcal{A}(w)$得到输出x。如果$G(x)=w$，则输出 1；否则，输出 0。

现在分析D的行为。首先考虑当输入的串w是随机选取时，D输出 1 的概率。因为G的范围至多有2^n个值（也就是说，值$\{G(s)\}_{s\in\{0,1\}^n}$），串w在G的范围内的概率至多为2^{-n}。当不是这种情况时，$\mathcal{A}$不可能计算出w的逆，因而D不可能输出 1。总结得到

$$\Pr_{w\leftarrow\{0,1\}^{2n}}[D(w)=1]\leqslant 2^{-n}$$

另一方面，如果对于种子$s\in\{0,1\}^n$，$w=G(x)$是均匀随机选取的，那么根据定义，$\mathcal{A}$正确计算出w的逆的概率恰好是$\varepsilon(n)$，因此得

$$\left|\Pr_{w\leftarrow\{0,1\}^{2n}}[D(w)=1]-\Pr_{s\leftarrow\{0,1\}^{n}}[D(G(s))=1]\right|\geqslant\varepsilon(n)-\frac{1}{2^n}$$

因为G是伪随机发生器，则存在一个可忽略函数negl，其中$\varepsilon(n)-\frac{1}{2^n}\leqslant \mathsf{negl}(n)$。从而结论是：$\varepsilon(n)$是可忽略的，证明了$G$是单向函数。 ■

对称密钥加密方案意味着单向函数。命题 6.31 表明如果想要构造伪随机发生器或伪随机函数，则需要假设单向函数是存在的。这不意味着单向函数对于构造安全的对称密钥加密方案是必要的，因为不依赖这些原语也可能构造安全的加密方案。此外，只要明文不长于密钥，就有可能构造完美安全的加密方案（参见第 2 章）。因此，安全的对称密钥加密方案意味着单向函数的证明更加微妙。

现在证明一种加密方案满足所考虑的最弱的安全定义（即存在窃听者的条件下不可区分加密）意味着单向函数的存在性。

命题 6.32 如果在窃听者的条件下不可区分加密的对称密钥方案（如定义 3.8）存在，则存在单向函数。

证明 这依据对一个事实的证明：定义 3.8 要求安全能够对加密任意长度的消息都满足。实际上，所需要的是：支持消息明文长于密钥的加密方案。重要的是，这个定理不支持那些加密与密钥等长的明文的加密方案（例如完美安全的“一次一密”方案）。

令$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$是窃听者存在条件下不可区分加密的对称密钥加密方案。假设当使用一个n比特的密钥，Enc使用最多$l(n)$比特的随机位来加密长度为$2n$的明文消息。用$\mathsf{Enc}_k(m;r)$表示对消息m的加密，同时使用密钥k和随机值r。

定义函数f

$$f(k,m,r)\stackrel{\text{def}}{=}(\mathsf{Enc}_k(m;r),m)$$

其中$|k|=n$，$|m|=2n$，$|r|=l(n)$，称函数f是单向函数。立刻可知它可以高效地计算，但很难求逆。令$\mathcal{A}$是概率多项式时间算法，并设

$$\varepsilon(n)\stackrel{\text{def}}{=}\Pr[\mathsf{Invert}_{\mathcal{A},f}(n)=1]$$

（比较定义 6.1）。证明$\varepsilon(n)$是可忽略的，将得到函数f是单向函数的证明。

考虑下列概率多项式时间的敌手$\mathcal{A}'$，在实验$\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)$中运行：

敌手 $\mathcal{A}'(1^n)$

(1) 随机选取$m_0,m_1\leftarrow\{0,1\}^n$并且输出两个明文消息，得到一个挑战密文$c$。

(2) 运行$\mathcal{A}(c,m_0)$得到(k',m',r')。如果$f(k',m',r')=(c,m_0)$，输出 0；否则，输出一个随机比特。

分析当$b=0$时，$\mathcal{A}'$输出 0 的概率。（回顾$b=0$表示挑战密文是m_0的加密。）令$\mathsf{invert}_{\mathcal{A}}$表示$\mathcal{A}$输出$(k',m',r')$的事件，这里$f(k',m',r')=(c,m_0)$。（当$\mathsf{invert}_{\mathcal{A}}$发生时，由$\mathcal{A}$输出的密钥$k'$可能不是“正确的密钥”，即它可能不等于实验中计算挑战密文的密钥k，但这与这里的目的无关。）观察到$b=0$时，事件$\mathsf{invert}_{\mathcal{A}}$发生的概率恰好为$\varepsilon(n)$。这是正确的，因为用于计算$c$的密钥$k$是均匀随机选取的，正如消息$m_0$和用于计算$c$的随机值。

当$\mathsf{invert}_{\mathcal{A}}$发生时，敌手$\mathcal{A}'$输出 0。当$\mathsf{invert}_{\mathcal{A}}$不发生时，$\mathcal{A}'$输出一个随机比特。所以，$b=0$时$\mathcal{A}'$成功（即输出正确的回答）的概率由下式给出

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1|b=0]\\
&\quad=\Pr[\mathsf{invert}_{\mathcal{A}}|\ b=0]+\frac{1}{2}\cdot(1-\Pr[\mathsf{invert}_{\mathcal{A}}|\ b=0])\\
&\quad=\varepsilon(n)+\frac{1}{2}\cdot(1-\varepsilon(n))\\
&\quad=\frac{1}{2}+\frac{\varepsilon(n)}{2}
\end{aligned}$$

进而分析当$b=1$时$\mathcal{A}'$输出 1 的概率。如前所述，从确定事件$\mathsf{invert}_{\mathcal{A}}$发生的概率开始。初看之下，当$b=1$时$\mathsf{invert}_{\mathcal{A}}$可能不会发生，因为$c$是$m_1$的密文，从而$\mathcal{A}$不可能找到满足$f(k',m',r')=(c,m_0)$的$(k',m',r')$。可是这是不正确的，因为对于某些$c=\mathsf{Enc}_k(m_1)$，可能存在不同的密钥$k'$，满足$m_0=Dec_{k'}(c)$；其实对完美安全的加密方案来说，这个属性对于每个$m_0$和$m_1$都有满足的。

然而，事件$\mathsf{invert}_{\mathcal{A}}$的发生概率最大也是可忽略的。为了说明这一点，固定一个挑战密文$c=\mathsf{Enc}_k(m_1)$，同时注意当$b=1$时密文独立于m_0。现在至多有2^n个可能的消息明文，对每个可能的密钥值都有一个明文与密文c对应。如果m_0恰好是这种可能性，那么不能得到事件$\mathsf{invert}_{\mathcal{A}}$发生的概率界限。另一方面，如果$m_0$不在这种可能性中，那么事件$\mathsf{invert}_{\mathcal{A}}$不可能发生（因为既然这样，$(c,m_0)$就不在$f$的范围内）。因为至多有$2^n$个可能的消息明文与$c$相关，而且$m_0$是从$\{0,1\}^n$中均匀随机选取的，事件$\mathsf{invert}_{\mathcal{A}}$发生的概率至多为$2^n/2^{2n}=2^{-n}$。这反过来表明当$b=1$时$\mathcal{A}'$成功的概率由下式给出

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1|\ b=1]\\
&\quad=\frac{1}{2}\cdot(1-\Pr[\mathsf{invert}_{\mathcal{A}}|\ b=1])\geqslant\frac{1}{2}\cdot(1-2^{-n})\\
&\quad=\frac{1}{2}-2^{-(n+1)}
\end{aligned}$$

综上所述，加上b是随机选取，有

$$\begin{aligned}
&\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1]\\
&=\frac{1}{2}\cdot\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1|\ b=0]+\frac{1}{2}\cdot\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1|\ b=1]\\
&\geqslant\frac{1}{2}\cdot(\frac{1}{2}+\frac{\varepsilon(n)}{2})+\frac{1}{2}\cdot(\frac{1}{2}-2^{-(n+1)})\\
&=\frac{1}{2}+\frac{\varepsilon(n)}{4}-\frac{1}{2^{n+2}}
\end{aligned}$$

Π的安全表明对于可忽略的函数negl有$\Pr[\mathsf{PrivK}^{eav}_{\Pi,\mathcal{A}'}(n)=1]\leqslant\frac{1}{2}+\mathsf{negl}(n)$。这反过来表明$\varepsilon(n)$是可忽略的，完成了函数$f$是单向的证明。 ■

消息鉴别码意味着单向函数。 满足定义 4.2 的消息鉴别码意味着单向函数的存在性，这同样是正确的。至于对称密钥加密情形，这个事实的证明稍微有些微妙，因为当需要鉴别的消息的数量有“先念（预先）”的范围界限时，则不存在无条件安全的消息鉴别码。因此，证明需要用到一个事实：即使当敌手看到任意（多项式）个消息的鉴别标签时，定义 4.2 要求安全性仍然满足。证明相当复杂，我们这里不再给出。

讨论。总结如下：为得到所有非平凡的对称密钥密码学体制，存在单向函数是必要的也是充分的假设。换句话说，单向函数的存在性假设是对称密钥密码学所关心的底线。这似乎与后面将研究的公钥密码学不一样。尽管单向函数对公钥加密也是必要的，但似乎不是充分条件。（除了不知道如何由单向函数构造公钥加密，有证据表明：这样的构造方法在某种意义上“不可能存在”。）

6.8 偏离——计算不可区分性

计算不可区分性是密码学理论的核心。它基于本章介绍的大部分内容，并且值得明确地论述。非正式地说，如果没有有效的算法可以分离它们（或者区分它们），则两个概率分布是计算不可区分的。形式化如下。令D为某个概率多项式时间算法，或者区分器，则提供给D一个来自第一个分布或第二个分布的抽样。当得到一个来自第一个或第二个分布的抽样时，如果每一个这样的区分器D几乎以同样的概率输出 1，则说这两个分布是“计算不可区分的”。这听起来很熟悉，实际上和如何定义伪随机发生器是一样的。事实上，一个伪随机发生器是产生一个分布的算法，该分布与关于一定长度的串上的均匀分布是计算不可区分的。下面将以这种方法正式地重新定义伪随机发生器的概念。

计算的不可区分性的实际定义与“概率总体”有关。这些概率总体是概率分布的无穷序列（而不是单一分布）。这种形式是渐近逼近的必然结果，因为利用穷举搜索很“容易”区分两个确定的有穷分布。

定义 6.33 党员令I是可数集。一个由I**索引的概率总体**是随机变量$\{X_i\}_{i\in I}$的集合。

大多数情况下，I是自然数$\mathbb{N}$或者是$\{0,1\}^*$的一个高效可计算的子集。$I=\mathbb{N}$时，一个分布总体仅仅是随机变量$X_1, X_2, \cdots$的序列，而且安全参数设为n时，随机变量X_n可能对应于某些密码学方案的输出。这种情况下，对于某些多项式p，X_n将在$\{0,1\}^{\leqslant p(n)}$（即，长度至多为$p(n)$的比特串）上取值。

根据这个表示方法，现在可以形式化地定义两个分布计算不可区分的含义。

定义 6.34 两个概率总体$X=\{X_n\}_{n\in\mathbb{N}}$和$Y=\{Y_n\}_{n\in\mathbb{N}}$是计算不可区分的，表示为$X \overset{c}{\equiv} Y$，如果对于每一个概率多项式时间的区分器$D$，都存在一个可忽略函数$\mathsf{negl}$，使得：

$$|\Pr[D(1^n, X_n)=1]-\Pr[D(1^n, Y_n)=1]| \leqslant \mathsf{negl}(n)$$

其中标记$D(1^n, X_n)$表明x的选取符合分布X_n，然后运行$D(1^n, x)$。

给出区分器D的一进制表示的输入1^n，使其能在以n表示的多项式时间运行。这在当X_n和Y_n的输出可能很短时是重要的。

6.8.1 伪随机性和伪随机发生器

伪随机性只是计算不可区分性的一个特例。令$U_{l(n)}$表示$\{0,1\}^{l(n)}$上的均匀分布，则有下列定义：

定义 6.35 如果对于某个多项式l，全体X和全体$U=\{U_{l(n)}\}_{n\in\mathbb{N}}$是计算不可区分的，则

全体$X=\{X_n\}_{n\in\mathbb{N}}$是伪随机的。

这能够用于重新定义伪随机发生器的概念（比较定义 3.14）：

定义 6.36 令$l(\cdot)$是多项式，同时令G为（确定的）多项式时间算法，其中对于所有s有$|G(s)|=l(|s|)$成立。如果下列两个条件成立，就称G是伪随机发生器：

(1) （可扩展性：）对于每一个n有$l(n)>n$成立。

(2) （伪随机性：）全体$\{G(U_n)\}_{n\in\mathbb{N}}$是伪随机的。

本书的许多其他定义和假设也能够用于不可区分性的特例。尽管前面包含这部分的跳过了，但我们给出一个例子：7.3.2 节的判定性 Diffie-Hellman（DDH）假设能够用这种方法形式化，即规定$(\mathbb{G},q,g,g^x,g^{xy})$的元组总体和$(\mathbb{G},q,g,g^x,g^z)$的元组总体是计算不可区分的，其中$(\mathbb{G},q,g)$是由某些算法$\mathcal{G}(1^n)$输出的，而$x,y,z$是从$\mathbb{Z}_q$中随机选取的。

6.8.2 多重抽样

关于计算不可区分性的一个重要的一般定理是：全部计算不可区分的多重抽样也是计算不可区分的。例如，假定有一个扩展系数为l的伪随机发生器D。那么，独立于应用G的$p(n)$的输出是长度为$p(n)\cdot l(n)$伪随机串。也就是说，$\{(G(s_1),\cdots,G(s_{p(n)}))\}$和$\{0,1\}^{p(n)l(n)}$上的均匀分布是计算不可区分的，其中$s_1,\cdots,s_{p(n)}$是独立均匀选取的长度为$n$的串。证明这个定理是因为其经常用于密码学的证明（例如，根据它证明定理 6.25），同时也是因为其是另一个混合参数的实例（见于定理 6.23 的证明）。

如果存在一个概率多项式时间的算法S使得对于每一个n，随机变量$S(1^n)$和X_n是完全相同的分布，则说明总体$X=\{X_n\}_{n\in\mathbb{N}}$是有效的抽样。也就是说，算法$S$是抽样$X$的有效方法。明显地，由伪随机发生器产生的总体是有效的抽样：算法S随机选取一个长度为n的串s然后输出$G(s)$。现在证明如果两个有效抽样总体X和Y是有效不可区分的，那么多项式次数（独立的）X抽样和多项式次数（独立的）Y抽样的是计算不可区分的。（如果X和Y不是有效抽样，则定理不成立。）我们用$\overline{X}=\{(X_n^{(1)},\cdots,X_n^{p(n)})\}_{n\in\mathbb{N}}$表示独立于抽样$X_n$的由$p(n)$产生的总体，类似地定义$\overline{Y}$。为了描叙清晰，我们不明确给出区分器的输入$1^n$，但假定知道安全参数的值，并且能够在以$n$表示的多项式时间内运行。

定理 6.37 令X和Y是计算不可区分的有效可抽样总体。那么，对于每个多项式$p(\cdot)$，总体$\overline{X}=\{(X_n^{(1)},\cdots,X_n^{p(n)})\}_{n\in\mathbb{N}}$和总体$\overline{Y}=\{(Y_n^{(1)},\cdots,Y_n^{p(n)})\}_{n\in\mathbb{N}}$是计算不可区分的。

证明 采用归纳法证明。我们表明如果存在概率多项式时间的区分器D以不可忽略的概率成功区分$\overline{X}$和$\overline{Y}$，那么存在一个概率多项式时间区分器D'以不可忽略的概率成功区分X的一次抽样和Y的一次抽样。形式化地，确定某些概率多项式时间区分器D，并定义

$$\varepsilon(n)\stackrel{\text{def}}{=}|\Pr[D(X_n^{(1)},\cdots,X_n^{p(n)})=1]-\Pr[D(Y_n^{(1)},\cdots,Y_n^{p(n)})=1]|$$

对于$0\leqslant i\leqslant p(n)$，定义一个混合随机变元$H_n^i$作为一个序列，它包含$i$个独立于$X_n$的拷贝，紧接着是$p(n)-i$个独立于$Y_n$的拷贝。即

$$H_n^i \stackrel{\text{def}}{=} (X_n^{(1)}, \cdots, X_n^{(i)}, Y_n^{(i+1)}, \cdots, Y_n^{p(n)})$$

注意$H_n^0 = \overline{Y}_n$和$H_n^{p(n)} = \overline{X}_n$。混合参数的主要思想是如果$D$能区分这些极限混合，那么也能区分相邻的混合（即使没有“设计来”这样做）。为了看到这个结果，在进行形式化讨论之前，提供基本的混合分析。有

$$\begin{aligned}&|\Pr[D(\overline{X_n}) = 1] - \Pr[D(\overline{Y_n}) = 1]| \\ &\quad = |\sum_{i=0}^{p(n)-1} \Pr[D(H_n^i) = 1] - \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i+1}) = 1]|\end{aligned}$$

这是根据叠加总和剩余的唯一项是$\Pr[D(H_n^0) = 1]$和$\Pr[D(H_n^{p(n)}) = 1]$。因此，

$$\begin{aligned}\varepsilon(n) &= |\Pr[D(\overline{X_n}) = 1] - \Pr[D(\overline{Y_n}) = 1]| \\ &= |\sum_{i=0}^{p(n)-1} \Pr[D(H_n^i) = 1] - \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i+1}) = 1]| \\ &\leqslant \sum_{i=0}^{p(n)-1} |\Pr[D(H_n^i) = 1] - \Pr[D(H_n^{i+1}) = 1]|\end{aligned}$$

因此，存在一个i，满足

$$|\Pr[D(H_n^i) = 1] - \Pr[D(H_n^{i+1}) = 1]| \geqslant \varepsilon(n)/p(n)$$

H_n^i和H_n^{i+1}唯一的不同发生在第$(i+1)$次抽样（两个分布中，前i次抽样来自X_n，而后$n-i-1$次抽样来自Y_n）。所以，D能够区分H_n^i和H_n^{i+1}的事实可用于构造一个区分X的一次抽样和Y的一次抽样的区分器D'，与假设$X \stackrel{c}{\equiv} Y$矛盾。下述正式证明中，D'随机选取i（因为不知道对哪个i值，D能够良好地区分）。

正式地说，对于X_n和Y_n的一次抽样，构造一个概率多项式时间的区分器D'。当输入一次抽样α时，D'选取一个随机数$i \leftarrow \{0, \cdots, p(n)-1\}$，生成向量$\overline{H}_n = (X_n^{(1)}, \cdots, X_n^{(i)}, \alpha, Y_n^{(i+2)}, \cdots, Y_n^{(p(n))})$，针对向量$\overline{H}_n$调用$D$，并且输出$D$的结果[③]。如果$\alpha$的分布符合$X_n$，那么$\overline{H}_n$的分布正好类似$H_n^{i+1}$（因为前$i+1$次抽样来自$X_n$，而后$n-i-1$次抽样来自$Y_n$）。相反，如果$\alpha$的分布符合$Y_n$，那么$\overline{H}_n$的分布正好类似$H_n^i$（因为前$i$次抽样来自$X_n$，而后$n-i$次抽样来自$Y_n$）。这个讨论成立是因为抽样是独立的，所以使得由谁产生抽样以及什么次序产生没有区别。现在，每一个i正好是以$1/p(n)$的概率选取的。因此，

$$\Pr[D'(X_n) = 1] = \frac{1}{p(n)} \cdot \sum_{i=0}^{p(n)-1} \Pr[D(H_n^{i+1}) = 1]$$

而且

$$\Pr[D'(Y_n) = 1] = \frac{1}{p(n)} \cdot \sum_{i=0}^{p(n)-1} \Pr[D(H_n^i) = 1]$$

因而接下来有：

---

③ 对X和Y的高效采样对于构造向量$\overline{H}_n$是需要的。

$$
\begin{aligned}
&|\Pr[D'(X_n)=1]-\Pr[D'(Y_n)=1]| \\
&=\frac{1}{p(n)}\cdot\Big|\sum_{i=0}^{p(n)-1}\Pr[D(H_n^{i+1})=1]-\sum_{i=0}^{p(n)-1}\Pr[D(H_n^{i})=1]\Big| \\
&=\frac{1}{p(n)}\cdot|\Pr[D(H_n^{p(n)})=1]-\Pr[D(H_n^{0})=1]| \\
&=\frac{1}{p(n)}\cdot|\Pr[D(\overline{X_n})=1]-\Pr[D(\overline{Y_n})=1]|\geqslant\frac{\varepsilon(n)}{p(n)}
\end{aligned}
$$

因为$X\overset{c}{\equiv}Y$，故存在一个可忽略函数negl，满足$|\Pr[D'(X_n)=1]-\Pr[D'(Y_n)=1]|\leqslant \mathsf{negl}(n)$。因$p$是多项式，这表明$\varepsilon$是可忽略的。■

参考文献及扩展阅读材料介绍

单向函数的概念是由 Diffie 和 Hellman 首次提出的[47]，后来由 Yao 形式化并研究[149]。硬核比特的概念是由 Blum 和 Micali 提出[23]的，Goldreich 和 Levin 证明了对于每个单向函数都存在一个硬核比特[68]。

伪随机性的概念是 Yao 首次提出的[149]，Blum 和 Micali 给出了第一个伪随机发生器的构造（在特定的数论困难性假设下）[23]。Yao 给出了由单向置换构造伪随机发生器的构造方法[149]，Hastad 等证明了伪随机发生器可以由单向函数构造的事实[74]。Goldreich、Goldwasser 和 Micali 定义并构造了伪随机函数[67]，其扩展到伪随机置换是 Luby 和 Rackoff 完成的[97]。Goldreich 的专著[64]中对定理 6.28 进行了非常清楚和简洁的证明。文献[79]证明了单向函数对于多数对称密钥密码学是必要的。

本章的多数内容参照了 Goldreich 的专著[64]，强烈建议有兴趣深入理解密码学基本原理的读者阅读此书。本章仅仅是密码学所提供的丰富理论的一次浅尝辄止。

练　习

6.1 证明加法函数$f(x,y)=x+y$（其中$|x|=|y|$，x,y为自然数）不是单向函数。类似地，证明$f(x)=x^2$也不是单向函数。

6.2 证明如果存在单向函数，则存在单向函数f满足对每个n有$f(0^n)=0^n$。对回答给出完整（形式化）的证明。注意这个说明：对无穷多个x的值，函数f是容易求逆的。说明为什么这和单向性不矛盾。

6.3 证明如果存在单向函数，则存在长度保留的单向函数。对你的回答给出完整的证明。
提示：令f是单向函数，令$p(\cdot)$是多项式使得$|f(x)|\leqslant p(|x|)$（证明这样的函数p是存在的）。定义$f'(x)=f(x)||1^{p(|x|)-|f(x)|}$。证明$f'$是长度保留的且是单向的。

6.4 证明：如果f是单向函数，则当$|x_1|=|x_2|$时，$g(x_1,x_2)=f(f(x_1),x_2)$也是单向函数。观察到：g泄漏其输入比特的一半，但仍然是单向的。

6.5 设f是一个长度保留的单向函数，另hc是函数f的硬核谓词。定义$G(x)=(f(x),\mathsf{hc}(x))$，则$G$是一个伪随机生成器吗？证明你的回答。

6.6 证明：当且仅当存在单向函数簇时，存在单向函数。讨论为什么你的证明对单向置换继续存在。

6.7 设f是一个单向函数，那么$g(x)=f(f(x))$一定需要是单向函数吗？$g(x)=(f(x),f(f(x)))$呢？证明你的回答。

6.8 本习题适合学过复杂性理论课程或者熟悉$\mathcal{NP}$完全问题的学生。

（1）证明单向函数的存在性意味着$\mathcal{P}\neq\mathcal{NP}$。

（2）假设$\mathcal{P}\neq\mathcal{NP}$。证明存在一个函数$f$有：① 可以在多项式时间内计算；② 在最坏情况下（即对所有可能的多项式时间$\mathcal{A}$，$\Pr_{r\leftarrow\{0,1\}}[f(A(f(x)))=f(x)]\neq 1$），函数很难求逆；③不是单向的。

6.9 设$x\in\{0,1\}^n$，且$x=x_1\cdots x_n$。证明如果存在单向函数，则存在一个单向函数f满足：对每个i，都存在一个算法A_i，使得

$$\Pr_{x\leftarrow\{0,1\}^n}[A_i(f(x))=x_i]\geqslant\frac{1}{2}+\frac{1}{2n}$$

（这个练习说明，不必声称每个单向函数都隐藏输入的至少一个特定比特位。）

6.10 证明如果一个单射函数有一个硬核谓词，那么它是单向的。

6.11 通过找寻 Chernoff 界来完成命题 6.15 的证明，并将其用于提高$\mathcal{A}'$推测x_i的过程。

6.12 证明声称 6.21。

6.13 设G是伪随机发生器。证明：

$$G'(x_1\|\cdots\|x_n)=(G(x_1)\|\,G(x_2)\|\cdots\|G(x_n))$$

是一个伪随机发生器，其中$|x_1|=\cdots=|x_n|=n$。

提示：使用混合参数。可以不使用定理 6.37。

6.14 证明：定义为

$$G'(s)=G_0(G_0(s)),G_0(G_1(s)),G_1(G_0(s)),G_1(G_1(s))$$

的函数G'是扩展系数为$l(n)=4n$的伪随机发生器。

6.15 证明如果修改构造方法 6.24 使得敌手能够问询任意串$x\in\{0,1\}^{\leqslant n}\ \{\epsilon\}$（即任意长度最多为$n$的非空串）的$F_k(x)$，则这个构造方法不再是伪随机函数。

6.16 证明：如果存在一个伪随机函数F，其密钥长度为n，且将$p(n)$比特的输入映射到单一比特的输出，则存在一个伪随机函数可以将$p(n)$比特输入映射到n比特输出。（这里的n同样表示安全参数。）要求不依赖 6.7 节的结论直接给出构造方法。

提示：使用长度为n^2的密钥，并使用混合参数证明构造方法的安全性。

6.17 假设存在伪随机置换。证明存在一个有密钥的置换F是伪随机的，但不是强伪随机的。

提示：虽然这可由练习 6.19 得出，但直接给出证明也是可能的。

6.18 证明使用伪随机轮函数（如式（6.10））的二轮Feistel网络不是伪随机的。

6.19 证明使用伪随机轮函数（如式（6.11））的三轮Feistel网络不是强伪随机的。

提示：这明显比前面一题更难。利用一个区分器，进行两次置换问询和一次求逆问询。

6.20 设G是扩展系数为$l(n)=n+1$的伪随机发生器。证明G是单向函数。

6.21 设$X=\{X_n\}_{n\in\mathbb{N}}$和$Y=\{Y_n\}_{n\in\mathbb{N}}$是计算不可区分的概率总体。

（1）证明对任意概率多项式时间算法$\mathcal{A}$，有$\{\mathcal{A}(X_n)\}_{n\in\mathbb{N}}$和$\{\mathcal{A}(Y_n)\}_{n\in\mathbb{N}}$是计算不可区分的。

（2）证明如果$\mathcal{A}$不能在多项式时间运行，则上述结论可能不再成立。

第三部分　公钥（非对称密钥）密码学

第7章　数论和密码学困难性假设

如前所述，现代密码学几乎总是基于一个假设，即某些难题不能在多项式时间内解决。（参见 1.4.2 节关于这一点的讨论。）例如第 3 章和第 4 章提及高效的对称密钥加密方案，以及消息鉴别，都基于伪随机置换是存在的这一假设。曾经提到，这意味着存在某些带密钥的置换F，区分与F_k（对于一个随机选择的密钥k）交互，还是与真正的随机置换交互，不能在多项式时间内完成。

表面上，存在伪随机序列的假设似乎是一个非常强的假设，而且这一假设并不明显，所以追问这种假设是否为真或者是否存在证据，也是合理的。第 5 章已经探究了如何在实践中构造伪随机置换（分组密码）。这些构造方法的抗攻击性至少预示着伪随机置换的存在似乎是有道理的。然而，依然很难想象某些F从直观上就可以确信是伪随机置换。当前的理论状况是：并不知道如何证明实践中构造方法的伪随机性，这些构造方法与任何“更合理的”假设相关。总的来说，这是一个不完全满意的状态。

相反，就像第 3 章中（在第 6 章具体研究）提到的，可以证明：基于弱很多的假设（即存在单向函数），可证明伪随机置换是存在的。（非正式地说，如果函数容易计算，但很难求逆，就是单向函数。参考 7.4.1 节。）然而，除了 6.1.2 节中的简短介绍，并没有发现任何被认为是单向函数的实例。

本章的目的之一是介绍各种被认为是“困难的”问题，以及介绍可以基于这类问题[①]推测出的单向函数。本章的第二个目标是建立学习公钥密码学所需要的基础知识（本书的另一个主题）。

这里探讨的所有例子本质上是基于数论，所以先简短介绍一些数论与群论知识。因为额外感兴趣的是可以高效率解决的问题（甚至是一个容易在单个方向进行计算的单向函数，一个必须允许诚实方进行高效算法的加密方案），所以也开始研究“算法”数论。因此，即使是熟悉数论或者群论的读者，也建议阅读此章，因为在关于这些主题的纯数学探讨中算法方面通常被忽略。

在算法数论中，有一个方法说明何为“多项式时间”。算法的运行时间通常用函数的输入长度来度量。（如果算法被给予一个安全参数1^n作为额外的输入，那么整个输入长度增长了n。）这意味着，若以整数N作为输入的算法的运行时间，是通过$||N||$来度量（也就是N的二进制表示形式的长度），而不是通过N本身来度量。一个输入为N的算法的运行时间为

① 重提一下，目前还不知道如何证明存在单向函数，最多能做的是使单向函数基于某些问题是困难的假设。

$\Theta(N)$，当用输入长度$||N|| = \Theta(\log N)$计算时，实际上为指数时间算法。

本章内容不是为了对数论进行全面介绍，而只是为了介绍最少量的，本书后面讨论密码学应用中所要用到的数论知识。因此，对数论知识的讨论分为两个部分：本章介绍的材料对理解第 8 章～第 10 章、第 12 章、第 13 章是足够的。第 11 章介绍了附加的只供该章使用的数论知识。

读者或许会疑惑为什么在此之前没有数论知识的讨论，而现在又突然需要用到。将数论知识放在此处讨论有两个原因：

(1) 本章可视为在第 3 章～第 6 章中介绍的对称密钥密码学的“自顶向下”方法的一个顶点。也就是说，第 3 章和第 4 章中介绍的所有对称密钥密码学都基于伪随机函数和伪随机置换。后者可以使用分组密码进行实际应用中的实例化，第 5 章介绍了分组密码。后者也可以用严格证明的方式从任何单向函数进行构造，第 6 章介绍了单向函数。在此，再往前走一步，讨论单向函数如何基于某些数学难题。图 7.1 对这种自顶向下的方法进行了总结。

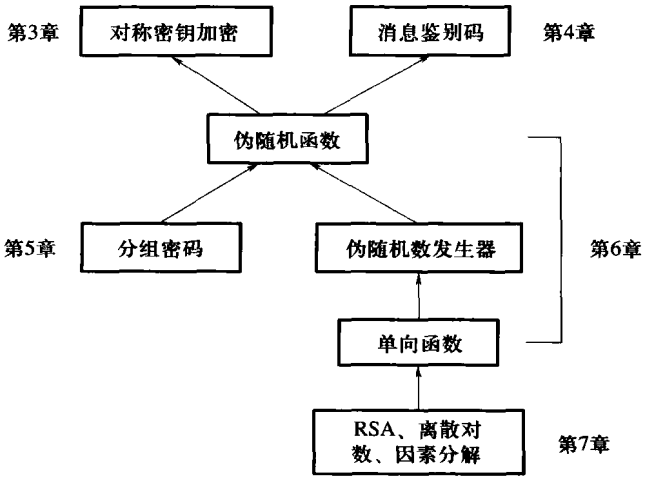


图 7.1　对称密钥加密的世界：自顶向下方法（箭头表示蕴涵）

(2) 学习这些资料的第二个目的是说明：到目前为止所关注的各种对称密钥情形与在接下来将要关注的各种公钥情形的区别。（公钥情形将在第 9 章介绍。）也就是说，在对称密钥情形中存在着合适的原语（即 Hash 函数和伪随机数发生器，伪随机函数和伪随机置换）用于构造方案，并且这些原语能够高效地被构造，至少以一种启发式的方式，没有应用任何数论知识。但是，在公钥情形中，所有已知的高效构造方法依赖于来自算法数论的数学难题。（本书也会研究一些并没有直接依靠数论知识的构造方法。不幸的是，这些构造方法的效率都低得多。）

本章的资料既作为到目前为止在对称密钥加密方面所学内容的一个顶点，也作为公钥密码学的基础。

7.1　预备知识和基本群论

本节首先回顾一些素数与基本模算术知识。即使以前学习过这些知识的读者也应该略读一下接下来的两节，因为其中有些内容可能是新的，并且提供了对大部分描述结果的证明。

（任何省略的证明可以在标准的代数课本中找到，参见本章后面的参考文献。）

7.1.1 素数与可除性

整数集合用$\mathbb{Z}$表示，当$a,b\in\mathbb{Z}$时，如果存在一个整数c，使得$ac=b$成立，则称a整除b，记作$a|b$。如果a不能整除b，记为$a\nmid b$。（虽然定义在a、b、c为负数或零时也有意义，但是主要考虑a、b、c都为正数的情形。）一个简单的观察是：如果$a|b$,且$a|c$,则对于任意$X,Y\in\mathbb{Z}$，都有$a|(Xb+Yc)$。

如果$a|b$并且a是正数，则称a是b的一个除数。如果附加条件$a\notin\{1,b\}$，则称a是b的一个非平凡除数，或者因子。正整数$p>1$如果没有因子，则为素数；也就是说，它只有 1 和自身两个除数。一个大于 1 的正整数如果不是素数则为合数。约定 1 既不是素数也不是合数。

算术基本定理为，任何大于 1 的整数可以唯一表示为素数的乘积（升序排列）。也就是说任何正整数$N>1$可以写为$N=\Pi_i p_i^{e_i}$，其中$\{p_i\}$为不同的素数，且对所有i，都有$e_i\geqslant 1$；此外$\{p_i\}$与$\{e_i\}$由排列顺序唯一确定。

小学阶段就已经熟悉了带余除法。下面的命题对这个概念进行正式介绍。

命题 7.1 令a为整数，b为正整数，则存在唯一的整数q,r满足$a=qb+r$，其中$0\leqslant r<b$。

此外，给出命题中的整数a和b，可以在多项式时间内计算出q,r，参见附录 B.1。

两个非负整数a,b的最大公约数记为$\gcd(a,b)$，它是满足$c|a$，$c|b$的最大整数c。（$\gcd(0,0)$无定义。）当a,b其中任意一个或两个为负时，最大公约数的定义仍有意义，但是不需要用到这一点；因此，当写$\gcd(a,b)$时，通常假设$a,b\geqslant 0$。注意$\gcd(b,0)=\gcd(0,b)=b$；同时，如果p是素数，则$\gcd(a,p)$既不等于 1 也不等于p。如果$\gcd(a,b)=1$,则称a,b互素。

下面是一个非常有用的结论：

命题 7.2 设a,b为正整数，则存在整数X,Y满足$Xa+Yb=\gcd(a,b)$。此外，$\gcd(a,b)$是可用这种方式表示的最小正整数。

证明 考虑集合$I\stackrel{\text{def}}{=}\{\hat{X}a+\hat{Y}b\mid\hat{X},\hat{Y}\in\mathbb{Z}\}$。注意到$a,b\in I$，所以$I$肯定包含某些正整数。设$d$是$I$中的最小正整数，下面只需证明$d=\gcd(a,b)$。（因为$d\in I$，故有某些$X,Y\in\mathbb{Z}$，$d$可以写为$d=Xa+Yb$，这样就证明了定理。）

为了证明$d=\gcd(a,b)$，必须证明：$d|a,d|b$，且d是拥有该性质的最大整数。事实上，可以证明d整除I中的所有元素。为了说明这一点，选取一个任意值$c\in I$并且记为$c=X'a+Y'b$，其中$X',Y'\in\mathbb{Z}$。利用带余除法（命题 7.1），得到$c=qd+r$，其中q,r为整数且$0\leqslant r<d$，则有

$$r=c-qd=X'a+Y'b-q(Xa+Yb)=(X'-qX)a+(Y'-aY)b\in I。$$

如果$r\neq 0$，这与选择d是I中的最小正整数矛盾（因为$r<d$）。所以，$r=0$，$d|c$。到此为止证明了d整除I中的每一个元素。

由于$a\in I,b\in I$，上面的证明表明$d|a,d|b$，所以d是a与b的公约数。剩下的是需要证明它是最大的公约数。假设存在一个整数$d'>d$，满足$d'|a,d'|b$，则通过前面的一个观察结论可

得$d'|Xa+Yb$。由于后者等于d，意味着$d'|d$，这与d'大于d矛盾。因此，d是整除a与b的最大整数，因此$d=\gcd(a,b)$。 ■

给定a和b，欧几里得算法可以在多项式时间内计算出$\gcd(a,b)$。扩展欧几里得算法可以用来在多项式时间内计算出X,Y（就像上面命题中的介绍）。参见附录 B.1.2 可得到更多细节。

前面的命题在证明关于可除性的结论时非常有用。下面举两个例子。

命题 7.3 如果$c|ab$，且$\gcd(a,c)=1$，则$c|b$。特别地，如果p为素数且$p|ab$，则可得$p|a$，或$p|b$。

证明 由于$c \mid ab$，对于某些整数γ可以写为$\gamma c=ab$。如果$\gcd(a,c)=1$，应用前面的命题，可知存在两个整数X,Y满足$1=Xa+Yc$。两边同时乘以b，可以得到

$$b=Xab+Ycb=X\gamma c+Ycb=c\cdot(X\gamma+Yb)$$

由于$X\gamma+Yb$是一个整数，可以得到$c \mid b$。

如果$p \nmid a$，则$\gcd(a,p)=1$，由此可以证明命题的第二部分。 ■

命题 7.4 如果$p \mid N$，$q \mid N$，且$\gcd(p,q)=1$，则$pq \mid N$。

证明 设$pa=N,qb=N$，并且$1=Xp+Yq$（利用命题 7.2），其中a,b,X,Y都是整数。在最后一个方程两边同时乘以N，得到

$$N=XpN+YqN=Xpqb+Yqpa=pq(Xb+Ya)$$

故$pq \mid N$得证。 ■

7.1.2 模算术

设$a,b,c\in\mathbb{Z}$且$N>1$，用$[a \bmod N]$来表示a除以N的余数。更详细地说：由命题 7.1 可知存在唯一的q,r满足$a=qN+r$，其中 $0\leqslant r<N$。定义$[a \bmod N]$等于r。注意到 $0\leqslant[a \bmod N]<N$。把a映射到$[a \bmod N]$的过程称为模N的“消减”运算。

如果$[a \bmod N]=[b \bmod N]$，则称a,b模N同余，记作$a=b \bmod N$。也就是说a被N除的余数与b被N除的余数相等。注意到当且仅当$N \mid (a-b)$时$a=b \bmod N$。使用这种表示法，表达式

$$a=b=c=\cdots=z \bmod N$$

序列中的每一个等号（并不只是最后一个）表示对N同余。

注意到$a=[b \bmod N]$意味着$a=b \bmod N$，但是反之不成立。（另一方面，当且仅当$a=b \bmod N$时$[a \bmod N]=[b \bmod N]$。）例如，$36=21 \bmod 15$，但是$36\neq[21 \bmod 15]=6$。

对N同余的关系是一个等价关系：也就是说，它具有自反性（对所有a，都有$a=a \bmod N$），对称性（$a=b \bmod N$意味着$b=a \bmod N$），传递性（如果$a=b \bmod N$，$b=c \bmod N$，则$a=c \bmod N$）。对N的同余关系也服从标准的加、减、乘法算术规则；所以，比方说，如果$a=a' \bmod N$且$b=b' \bmod N$，则有$(a+b)=(a'+b') \bmod N$，且$ab=a'b' \bmod N$。由此得出一个结论，同余计算中，可以“先做消减然后加或乘法”，代替了“先做加法或乘法然后消减”。这个性质通常用来简化计算。

例 7.5 计算$[1093028\cdot 190301 \bmod 100]$。因为$1093028=28 \bmod 100$，$190301=1 \bmod 100$，可以得到

$$1093028 \cdot 190301 = [1093028 \bmod 100] \cdot [190301 \bmod 100] \bmod 100$$
$$= 28 \cdot 1 = 28 \bmod 100$$

另一种计算方法（即先计算$1093028 \cdot 190301$的乘积，然后对100取模）耗时会多很多。

对N同余（一般情况下）并不满足除法运算。也就是说，如果$a = a' \bmod N$，$b = b' \bmod N$则$a/b = a'/b' \bmod N$并不一定正确。事实上，表达式“$a/b \bmod N$”并不总是有意义。举一个容易引起混淆的特例，$ab = cb \bmod N$并不一定意味着$a = c \bmod N$。

例 7.6 取$N = 24$，则$3 \cdot 2 = 6 = 15 \cdot 2 \bmod 24$，但是$3 \neq 15 \bmod 24$。

但是，在某些情况下，可以定义一个关于除法的非常有用的符号。如果给定一个整数b，存在一个整数b^{-1}满足$bb^{-1} = 1 \bmod N$，则称b^{-1}为b关于模N的(乘法)逆元，并称b模N可逆。显然，0 不可逆。不难证明如果β是b的模N乘法逆元，则$[\beta \bmod N]$也是。此外，如果β'是b模N的另一个逆元，则有$[\beta \bmod N] = [\beta' \bmod N]$。当$b$可逆时，可以简单地用$b^{-1}$表示$b$在$\{1,\cdots, N-1\}$范围内的唯一乘法逆元。

当b模N可逆时，把除以b模N定义为乘以b^{-1}模N（也就是说，定义$a/b \stackrel{\text{def}}{=} ab^{-1} \bmod N$）。强调只有$b$可逆时才能定义被$b$除。如果$ab = cb \bmod N$并且$b$可逆，在方程两边同时除以$b$（或者两边同时乘以$b^{-1}$）可以得到

$$(ab) \cdot b^{-1} = (cb) \cdot b^{-1} \bmod N \quad \Rightarrow \quad a = c \bmod N。$$

从此例可以看到除法如期望的那样完成。在某种意义上，可逆整数更容易处理。

一个很容易想到的问题：哪些整数关于模N可逆？运用命题 7.2 可回答这个问题。

命题 7.7 设a, N为整数，$N > 1$，则当且仅当$\gcd(a, N) = 1$时a关于模N可逆。

证明 假设a关于模N可逆，用b表示其逆元。注意到：因为$0 \cdot b = 0 \bmod N$无论b为何值都成立，所以$a \neq 0$。因为$ab = 1 \bmod N$，对N同余意味着对某些$c \in \mathbb{Z}$，有$ab - 1 = cN$，相当于$ba - cN = 1$。运用命题 7.2，因为$\gcd(a, N)$是可以用这种方式表达的最小正整数，而此时并没有比 1 小的整数，这就意味着$\gcd(a, N) = 1$。

反过来，如果$\gcd(a, N) = 1$，则运用命题 7.2 可知：存在整数X, Y满足$Xa + YN = 1$。方程两边同时对N取模得到$Xa = 1 \bmod N$，并且发现$[X \bmod N]$是a的一个乘法逆元素。

例 7.8 设$a = 11, b = 17$。由$(-3) \cdot 11 + 2 \cdot 17 = 1$可得$14 = [-3 \bmod 17]$为11的逆元。可以验证$14 \cdot 11 = 1 \bmod 17$。

加、减、乘以及关于模N求逆（如果存在的话）可以在多项式时间内完成，参见附录 B.2。求幂运算（也就是计算$[a^b \bmod N]$，$b > 0$且为整数）也可在多项式时间内完成，参见附录 B.2.3。

7.1.3 群

令$\mathbb{G}$为一个集合，$\mathbb{G}$上的二元运算$\circ$是简单地为一个函数$\circ(\cdot, \cdot)$，并以$\mathbb{G}$中的两个元素作为输入。如果$g, h \in \mathbb{G}$，用$g \circ h$取代复杂的表示方法$\circ(g, h)$。

接下来开始介绍群论中的重要概念。

定义 7.9 群是一个集合$\mathbb{G}$和其上的二元运算$\circ$，满足下列条件：

(1)（封闭性：）对任意$g,h\in\mathbb{G}$，$g\circ h\in\mathbb{G}$。

(2)（存在单位元:）存在一个单位元$e\in\mathbb{G}$，对所有$g\in\mathbb{G}$，满足$e\circ g=g=g\circ e$。

(3)（存在逆元：）对任意$g\in\mathbb{G}$，存在一个元素$h\in\mathbb{G}$满足$g\circ h=e=h\circ g$。这样的元素h称为g的逆元。

(4)（结合律：）对任意的$g_1,g_2,g_3\in\mathbb{G}$，恒有$(g_1\circ g_2)\circ g_3=g_1\circ(g_2\circ g_3)$。

当$\mathbb{G}$中元素有限时，称$\mathbb{G}$为有限群，并且用$|\mathbb{G}|$表示群的阶，也就是$\mathbb{G}$中元素的个数。

群$\mathbb{G}$在$\circ$运算上如果满足如下条件则是阿贝尔群：

(5)（交换律：）对任意$g,h\in\mathbb{G}$，$g\circ h=h\circ g$。

当二元运算为已知，简单地称集合$\mathbb{G}$为群。

通常处理有限阿贝尔群。但是，当结论需要更多假设时，会谨慎地声明。

结合律意味着在书写长表达式时不需要圆括号；也就是说，表示法$g_1\circ g_2\circ\cdots\circ g_n$是确定的，因为运算$\circ$和计算次序没有关系。

可以证明群$\mathbb{G}$中的单位元是唯一的，称为群的单位元。同样也可以证明群中的每一个元素有唯一的逆元。参见练习 7.1。

如果$\mathbb{G}$是一个群，集合$\mathbb{H}\subseteq\mathbb{G}$，且$\mathbb{H}$关于$\mathbb{G}$中的运算构成群，则$\mathbb{H}$是$\mathbb{G}$的一个子群。为了检验$\mathbb{H}$是一个子群，需要验证定义 7.9 中的封闭性、存在单位元和逆元、结合律。（实际上，结合律会从$\mathbb{G}$中自动得到。如果$\mathbb{G}$是阿贝尔群，交换律也同样如此）。任意群$\mathbb{G}$通常都有平凡子群$\mathbb{G}$和$\{1\}$。如果$\mathbb{H}\neq\mathbb{G}$则称$\mathbb{H}$为$\mathbb{G}$的一个严格子群。

通常情况下，并不用$\circ$来表示群运算。取而代之，根据所讨论的群的情况用“加号”或者“乘号”来表示。当用加法符号时，群运算应用到两个元素g,h时表示为$g+h$；单位元用 0 表示，元素g的逆元素用$-g$表示。当用乘法符号时，群运算应用到两个元素g,h时表示为$g\cdot h$或者简写为gh；单位元用 1 表示，元素g的逆元素用g^{-1}表示。当在乘法运算中对N取模时，除以g可定义为乘以g^{-1}（也就是说，h/g意味着hg^{-1}）。当陈述一般情况的结论时，通常用乘法符号。这并不意味着群运算对应于整数加法和整数乘法，这仅仅作为一种有用的表示方法。

下面几个例子对理解这一点很有用处。

例7.10 一个集合在某种运算上可能是一个群，但在其他运算上可能不是。例如，整数集合$\mathbb{Z}$在加法运算上是阿贝尔群：单位元为 0，每一个整数g都有逆元$-g$。另一方面，$\mathbb{Z}$在乘法运算下不能构成群，如整数 2 在整数范围内没有乘法逆元。

例 7.11 实数集合$\mathbb{R}$在乘法运算上不能构成群，因为 0 没有乘法逆元。但是非零实数集合在乘法运算下是阿贝尔群，其单位元为 1。

下面的例子介绍将会经常用到的群$\mathbb{Z}_N$。

例 7.12 设整数 $N\geqslant 2$。如果集合$\{0,\cdots,N-1\}$满足加法模N运算（也就是说$a+b\stackrel{\text{def}}{=}[a+b \bmod N]$），则此群是阶为$N$的阿贝尔群：封闭性是显然的，结合律和交换律可以从整数满足这两条性质得到，单位元为 0，因为$a+(N-a)=0 \bmod N$，可得元素a的逆元素为$[(N-a) \bmod N]$。用$\mathbb{Z}_N$表示这个群。（也用$\mathbb{Z}_N$表示没有声明任何特殊群运算的集合$\{0,\cdots,N-1\}$。）

本节以一个简单的引理结尾：群的消去律。

引理 7.13 设$\mathbb{G}$为群，且$a,b,c\in\mathbb{G}$。如果$ac=bc$，则$a=b$。特别地，如果$ac=c$，则a是$\mathbb{G}$的单位元。

证明 由于$ac=bc$，在其两边同时乘以c的唯一逆元c^{-1}，可以得到$a=b$。详细过程为

$$\begin{aligned} ac=bc \quad &\Rightarrow \quad (ac)c^{-1}=(bc)\cdot c^{-1} \quad \Rightarrow \quad a(cc^{-1})=b(cc^{-1}) \quad \Rightarrow \quad a\cdot 1=b\cdot 1 \\ &\Rightarrow \quad a=b \end{aligned}$$

■

比较上面的证明和（命题 7.7 之前的）关于模N除法中消去律的讨论，由相似性表明，关于模N的可逆元素形成了一个乘法模N的群。下面将会讨论这个例子的更多细节。

群的幂运算。能够描述对群中一个固定元素g作m次群运算是非常有用的，这里m是一个正整数。当群运算为加法符号时，表示为$m\cdot g$或者mg；也就是，

$$mg=m\cdot g\stackrel{\text{def}}{=}\underbrace{g+\cdots+g}_{m\ times}$$

注意到m是一个整数，g是群中的元素。所以mg并不代表对m和g进行群运算（事实上，这个定义只作用在群运算用加号表示的群）。但幸运的是，这个表示法“像它应该做的那样运算”；所以，如果$g\in\mathbb{G}$，且m,m'为整数，则$(mg)+(m'g)=(m+m')g$，$m(m'g)=(mm')g$，$1\cdot g=g$。在阿贝尔群$\mathbb{G}$中，$g,h\in\mathbb{G}$，则有$(mg)+(mh)=m(g+h)$。

当用乘法符号时，对元素g进行m次群运算表示为g^m。也就是，

$$g^m\stackrel{\text{def}}{=}\underbrace{g\cdots g}_{m\ times}$$

类似幂运算的性质有：$g^m\cdot g^{m'}=g^{m+m'}$，$(g^m)^{m'}=g^{mm'}$，$g^1=g$。如果$\mathbb{G}$是阿贝尔群，且$g,h\in\mathbb{G}$，则$g^m\cdot h^m=(gh)^m$。注意到所有这些结果都只是简单地“转换”前面段落介绍的内容，只是将加法改为乘法。

将上面的表示方法以自然而然的方法扩展到m为 0 或负数的情况。（一般情况下，当m不是整数时，g^m无定义。）当用加法符号时，可得$0\cdot g\stackrel{\text{def}}{=}0$，$(-m)\cdot g\stackrel{\text{def}}{=}m\cdot(-g)$，其中$m$为正整数。（注意到方程‘$0\cdot g=0$’左边的‘0’是一个整数，然而右边的‘0’是群单位元。）正如期望的，可以证明$(-m)\cdot g=-(mg)$。当用乘法符号时$g^0\stackrel{\text{def}}{=}1$，$g^{-m}\stackrel{\text{def}}{=}(g^{-1})^m$。仍然如所期望的，可以证明$g^{-m}=(g^m)^{-1}$。

设$g\in\mathbb{G}$，$b\geqslant 0$且为整数，则幂运算g^b通过多项式次数的群$\mathbb{G}$的运算来计算。因此，如果群运算可以在多项式时间内计算则幂运算也可以在多项式时间内计算。附录 B.2.3 中将进行讨论。

现在有足够的知识证明下面这个值得注意的结论：

定理 7.14 设$\mathbb{G}$是一个有限群，其阶$m=|\mathbb{G}|$，则对任意元素$g\in\mathbb{G}$，有$g^m=1$。

证明 只对$\mathbb{G}$为阿贝尔群的情形进行证明（虽然定理对任意有限群都成立）。任意选定$g\in\mathbb{G}$，令$g_1,\cdots,g_m$为$\mathbb{G}$中元素，可得

$$g_1\cdot g_2\cdots g_m=(gg_1)\cdot(gg_2)\cdots(gg_m)$$

由引理 7.13 知$gg_i=gg_j$意味着$g_i=g_j$，所以等式右边括号内的m个元素都各不相同。因为$\mathbb{G}$中有m个元素，这m个元素在等式右边相乘就是$\mathbb{G}$中的所有元素以某个排列相乘。

又由于$\mathbb{G}$是阿贝尔群，可以“拿出”右边括号中所有出现的g，可得到

$$g_1 \cdot g_2 \cdots g_m = (gg_1) \cdot (gg_2) \cdots (gg_m) = g^m \cdot (g_1 \cdot g_2 \cdots g_m)$$

再次应用引理 7.13，可得$g^m = 1$。 ■

据此可以得出一个很重要的推论：“在幂运算中可以通过对群的阶取模”来计算。

推论 7.15 令$\mathbb{G}$是一个有限群，其阶$m = |\mathbb{G}| > 1$。则对任意$g \in \mathbb{G}$及任意整数i，可以得出$g^i = g^{[i \bmod m]}$。

证明 $i = qm + r$，其中q, r为整数，并且$r = [i \bmod m]$。运用定理 7.14 可得

$$g^i = g^{qm+r} = g^{qm} \cdot g^r = (g^m)^q \cdot g^r = 1^q \cdot g^r = g^r$$

证毕。 ■

例 7.16 若群运算为加法，上面的推论可变为：如果g是阶为m群的一个元素，则有$i \cdot g = [i \bmod m] \cdot g$。例如群$\mathbb{Z}_{15}$阶为$m = 15$，取元素$g = 11$。根据推论可得

$$152 \cdot 11 = [152 \bmod 15] \cdot 11 = 2 \cdot 11 = 11 + 11 = 22 = 7 \bmod 15$$

上面的介绍恰好证明了（参见例 7.5）可以“先消减再做乘法”而不是“先做乘法后消减”。

另一个推论在密码学应用中非常有用：

推论 7.17 令$\mathbb{G}$是一个有限群，$m = |\mathbb{G}| > 1$。令$e > 0$是一个整数，并且定义函数f_e：$\mathbb{G} \to \mathbb{G}$通过$f_e(g) = g^e$。如果$\gcd(e, m) = 1$，则$f_e$是一个置换（也就是双射）。此外，如果$d = [e^{-1} \bmod m]$则$f_d$是$f_e$的逆元素。

证明 应用命题 7.7，$\gcd(e, m) = 1$意味着e是关于模m的逆元。此时d是e关于模m的乘法逆元。推论的第二部分隐含着第一部分，所以只需要证明f_d是f_e的逆元。这很显然是正确的，因为对任意$g \in \mathbb{G}$，有

$$f_d(f_e(g)) = f_d(g^e) = (g^e)^d = g^e d = g^{[ed \bmod m]} = g^1 = g$$

其中第四个等式由推论 7.15 得来。

7.1.4 $\mathbb{Z}_N^*$群

在例 7.12 中已讨论集合$\mathbb{Z}_N = \{0, \cdots, N-1\}$是一个关于模$N$加法运算的群。可以定义一个群，其集合为$\{0, \cdots, N-1\}$，群运算为关于模$N$乘法吗？为满足条件，必须先去掉集合中不可逆的元素；例如，必须去掉 0，因为 0 没有乘法逆元。这不是唯一可能的元素：如果$N = 6$，通过逐一试探集合中所有元素，发现 3 不可逆。

哪些$a \in \{0, \cdots, N-1\}$关于模N是可逆的？命题 7.7 中表明这些元素恰好就是满足$\gcd(a, N) = 1$的a。从 7.1.2 节中也可以发现当a是可逆的，则其逆元在范围$\{0, \cdots, N-1\}$中。所以，可以定义对于任意$N > 1$，集合

$$\mathbb{Z}_N^* \stackrel{\text{def}}{=} \{a \in \{1, \cdots, N-1\} \mid \gcd(a, N) = 1\}$$

也就是说，$\mathbb{Z}_N^*$由集合$\{0, \cdots, N-1\}$中与N互素的整数组成。群运算为关于模N乘，即$ab \stackrel{\text{def}}{=} [ab \bmod N]$。

可以声称$\mathbb{Z}_N^*$是一个遵从此运算的阿贝尔群，因为元素 1 一直属于$\mathbb{Z}_N^*$，此集合明显包含

一个单位元。上面的讨论表明$\mathbb{Z}_N^*$中每一个元素都有一个乘法逆元，并且属于同一个集合。交换律和结合律可以从整数具有此性质的事实得到。为了证明其满足封闭性，令$a, b \in \mathbb{Z}_N^*$，$c = [ab \bmod N]$，且$c \notin \mathbb{Z}_N^*$。这意味着$\gcd(c, N) \neq 1$，所以存在一个素数p可以被N和c整除。因为对于某些整数q，$ab = qN + c$，可以得到$p \mid ab$。由命题 7.3，意味着$p \mid a$或者$p \mid b$；但是$\gcd(a, N) \neq 1$或者$\gcd(b, N) \neq 1$都与假设$a, b \in \mathbb{Z}_N^*$矛盾。

总结：

命题 7.18 令$N > 1$且为整数，则$\mathbb{Z}_N^*$在关于模N的模乘运算下为阿贝尔群。

定义$\phi(N) \stackrel{\text{def}}{=} |\mathbb{Z}_N^*|$为群$\mathbb{Z}_N^*$的阶（$\phi$称为欧拉函数）。$\phi(N)$的值为多少？首先考虑$N = p$是素数的情形。此时，$\{1, \cdots, p-1\}$中所有元素都与$p$互素，并且$\phi(p) = |\mathbb{Z}_p^*| = p - 1$。然后，考虑$N = pq$的情形，其中$p, q$为不同的素数。如果一个整数$a \in \{1, \cdots, N-1\}$并不与$N$互素，则$p \mid a$或$q \mid a$（$a$不能被$p$和$q$整除，因为这意味着$pq \mid a$，但$a < N = pq$）。$\{1, \cdots, N-1\}$中的元素可被$p$整除的元素有$(q-1)$个：$p, 2p, 3p, \cdots, (q-1)p$，另外，集合中可被$q$整除的元素有$(p-1)$个：$q, 2q, 3q, \cdots, (p-1)q$。剩下的元素（既不能被$p$，也不能被$q$整除）的数量为

$$(N-1) - (q-1) - (p-1) = pq - p - q + 1 = (p-1)(q-1)。$$

于是证明了当N是两个不同素数p, q的乘积时，$\phi(N) = (p-1)(q-1)$。

读者将被要求在练习 7.4 中证明下面的一般性结论（在本书后面的内容中很少用到）：

定理 7.19 设$N = \Pi_i p_i^{e_i}$，其中$\{p_i\}$为互不相同的素数，且$e_i \geqslant 1$。则$\phi(N) = \Pi_i p_i^{e_i - 1}(p_i - 1)$。

例 7.20 设$N = 15 = 5 \cdot 3$，则$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$，$|\mathbb{Z}_{15}^*| = 8 = 4 \cdot 2 = \phi(15)$。8 在$\mathbb{Z}_N^*$中的逆元为 2，因为$8 \cdot 2 = 16 = 1 \bmod 15$。

前面已经证明$\mathbb{Z}_N^*$的阶为$\phi(N)$。下面是定理 7.14 和推论 7.17 的简单推论：

推论 7.21 任意选取$N > 1$，$a \in \mathbb{Z}_N^*$，则

$$a^{\phi(N)} = 1 \bmod N$$

对于$N = p$是素数并且$a \in \{0, \cdots, p-1\}$的具体情况，可得

$$a^{p-1} = 1 \bmod p$$

推论 7.22 选取$N > 1$。对于整数$e > 1$，通过$f_e(x) = [x^e \bmod N]$定义f_e：$\mathbb{Z}_N^* \to \mathbb{Z}_N^*$。如果$e$与$\phi(N)$互素则$f_e$是一个置换。此外，如果$d = [e^{-1} \bmod \phi(N)]$，则$f_d$是$f_e$的逆。

7.1.5 *群同构和中国剩余定理

群$\mathbb{G}$的同构为$\mathbb{G}$提供了一种可以等价的替换思考。

定义 7.23 令群$\mathbb{G}$、$\mathbb{H}$的群运算分别为$\circ_{\mathbb{G}}$，$\circ_{\mathbb{H}}$。则函数f：$\mathbb{G} \to \mathbb{H}$是从$\mathbb{G}$到$\mathbb{H}$的同构，如果满足：

(1) f是为双射；

(2) 对任意$g_1, g_2 \in \mathbb{G}$，可得$f(g_1 \circ_{\mathbb{G}} g_2) = f(g_1) \circ_{\mathbb{H}} f(g_2)$。

如果存在一个从$\mathbb{G}$到$\mathbb{H}$的同构，则称这两个群同构，记作$\mathbb{G} \simeq \mathbb{H}$。

本质上，从$\mathbb{G}$到$\mathbb{H}$的同构仅仅是将$\mathbb{G}$中的元素改名为$\mathbb{H}$中的元素。注意到如果$\mathbb{G}$是有限群，且$\mathbb{G} \simeq \mathbb{H}$，则$\mathbb{H}$必须为有限群且与$\mathbb{G}$的大小（群中元素的个数）相同。同样，如果存在一个函数f是群$\mathbb{G}$到$\mathbb{H}$的同构，则函数f^{-1}是群$\mathbb{H}$到$\mathbb{G}$的同构。然而，计算f效率可能很高，但是计算f^{-1}效率可能很低（反之亦然）。

本节的目的是用群同构语言来更好地理解$\mathbb{Z}_N$和$\mathbb{Z}_N^*$的群结构，其中$N = pq$是两个互不相同素数的乘积。首先，需要介绍群“叉乘”的定义。给定群$\mathbb{G}$，$\mathbb{H}$，其群运算分别为$\circ_{\mathbb{G}}$，$\circ_{\mathbb{H}}$，现在定义一个新群$\mathbb{G} \times \mathbb{H}$（群$\mathbb{G}$与$\mathbb{H}$的“叉乘”）如下：$\mathbb{G} \times \mathbb{H}$的元素是有序对$(g, h)$，其中$g \in \mathbb{G}$，$h \in \mathbb{H}$；因此，如果$\mathbb{G}$有$n$个元素且$\mathbb{H}$有$n'$个元素，则$\mathbb{G} \times \mathbb{H}$有$n \cdot n'$个元素。群$\mathbb{G} \times \mathbb{H}$上的运算$\circ$是组成部分分别应用运算；也就是：

$$(g, h) \circ (g', h') \stackrel{\text{def}}{=} (g \circ_{\mathbb{G}} g', h \circ_{\mathbb{H}} h')$$

$\mathbb{G} \times \mathbb{H}$确实是一个群的证明留作练习 7.7。尽管在后面并不需要，但是有必要提出前面关于叉乘的定义可以很自然地扩展到超过两个群的情形。

定理 7.24（中国剩余定理） 设$N = pq$，其中p与q互素。则

$$\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q, \quad \mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

此外，设f为映射元素$x \in \{0, \cdots, N-1\}$到二元组(x_p, x_q)的函数，其中$x_p \in \{0, \cdots, p-1\}$且$x_q \in \{0, \cdots, q-1\}$。将此函数定义为

$$f(x) \stackrel{\text{def}}{=} ([x \bmod p], [x \bmod q])$$

则函数f是从$\mathbb{Z}_N$到$\mathbb{Z}_p \times \mathbb{Z}_q$的同构，以及从$\mathbb{Z}_N^*$到$\mathbb{Z}_p^* \times \mathbb{Z}_q^*$的同构。

证明 很明显对任意$x \in \mathbb{Z}_N$，$f(x)$的输出是一对元素(x_p, x_q)，其中$x_p \in \mathbb{Z}_p$且$x_q \in \mathbb{Z}_q$。此外，可以声明如果$x \in \mathbb{Z}_N^*$，则$(x_p, x_q) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$。事实上，如果$x_p \notin \mathbb{Z}_p^*$，则意味着$\gcd([x \bmod p], p) \neq 1$。但此时$\gcd(x, p) \neq 1$。这意味着$\gcd(x, N) \neq 1$，与假设$x \in \mathbb{Z}_N^*$矛盾。（如果$x_q \notin \mathbb{Z}_q^*$可以得到相似的结论。）

现在证明函数f是从$\mathbb{Z}_N$到$\mathbb{Z}_p \times \mathbb{Z}_q$的同构。（类似可证明其为从$\mathbb{Z}_N^*$到$\mathbb{Z}_p^* \times \mathbb{Z}_q^*$的同构。）首先证明$f$是单射。$f(x) = (x_p, x_q) = f(x')$，则$x = x_p = x' \bmod q$，$x = x_q = x' \bmod q$。这些反过来意味着$(x - x')$可以同时被$p$和$q$整除。因为$\gcd(p, q) = 1$，根据命题 7.4 的描述可得$pq = N$可以整除$(x - x')$。但是此时$x = x' \bmod N$。由于$x, x' \in \mathbb{Z}_N$，这意味着$x = x'$且$f$是单射。因为$|\mathbb{Z}_N| = N = p \cdot q = |\mathbb{Z}_p| \cdot |\mathbb{Z}_q|$，$\mathbb{Z}_N$和$\mathbb{Z}_p \times \mathbb{Z}_q$的大小相同。加上$f$是单射，可知$f$是双射。

下面的段落中，用$+_N$表示关于模N的模加运算，用$\boxplus$表示$\mathbb{Z}_p \times \mathbb{Z}_q$的群运算（也就是说，前部分做关于模$p$的加运算，后面部分做关于模$q$的加运算）。总结以上证明，$f$是从$\mathbb{Z}_N$到$\mathbb{Z}_p \times \mathbb{Z}_q$的同构，需要证明对于所有$a, b \in \mathbb{Z}_N$都有$f(a +_N b) = f(a) \boxplus f(b)$。

为了说明其正确性，注意到

$$f(a +_N b) = \Big([(a +_N b) \bmod p], [(a +_N b) \bmod q]\Big)$$
$$= \Big((a + b) \bmod p], [(a + b) \bmod q]\Big)$$
$$= \Big([a \bmod p], [(a \bmod q]\Big) \boxplus \Big([b \bmod p], [(b \bmod q]\Big) = f(a) \boxplus f(b)$$

（上面第二个等式是因为当$p \mid N$时，$[[X \bmod N] \bmod p] = [[X \bmod p] \bmod p]$，参见练习 7.8。）■

定理并不需要p或者q为素数。扩展的中国剩余定理为：如果$p_1, p_2, \cdots, p_l$两两互素（也就是说对所有$i \neq j$，都有$\gcd(p_i, q_j) = 1$），且$N \stackrel{\text{def}}{=} \Pi_{i=1}^{l} p_i$，则

$$\mathbb{Z}_N \simeq \mathbb{Z}_{p1} \times \cdots \times \mathbb{Z}_{pl}, \quad \mathbb{Z}_N^* \simeq \mathbb{Z}_{p1}^* \times \cdots \times \mathbb{Z}_{pl}^*$$

可以用上面介绍定理的扩展形式分别求出这两种情况下的同构。

通过符号定义当N已知，且$x \in \{0, \cdots, N-1\}$时，对于$x_p = [x \bmod p]$和$x_p = [x \bmod p]$，我们写为$x \leftrightarrow (x_p, x_q)$。也就是说，当且仅当$f(x) = (x_p, x_q)$时$x \leftrightarrow (x_p, x_q)$，其中$f$是前面定理中的函数。一种理解这个符号的方式为：它意味着“$\mathbb{Z}_N$中的x对应于$\mathbb{Z}_p \times \mathbb{Z}_q$中的$(x_p, x_q)$”。类似的符号定义在处理$x \in \mathbb{Z}_N^*$时将会用到。

例 7.25 取$15 = 5 \cdot 3$，考虑$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$。根据中国剩余定理知：此群和$\mathbb{Z}_5^* \times \mathbb{Z}_3^*$同构。事实上，可以计算

$$1 \leftrightarrow (1,1) \quad 2 \leftrightarrow (2,2) \quad 4 \leftrightarrow (4,1) \quad 7 \leftrightarrow (2,1)$$
$$8 \leftrightarrow (3,2) \quad 11 \leftrightarrow (1,2) \quad 13 \leftrightarrow (3,1) \quad 14 \leftrightarrow (4,2)$$

这里每一个可能序列(a, b)只出现一次，其中$a \in \mathbb{Z}_5^*$，$b \in \mathbb{Z}_3^*$。

中国剩余定理的应用。 如果两个群同构，则代表它们有相同的“代数结构”。然而，选择不同表示方法会影响群运算的计算效率。首先抽象地介绍这一点，然后在特定的群$\mathbb{Z}_N$和$\mathbb{Z}_N^*$中介绍。

设$\mathbb{G}$、$\mathbb{H}$都为群，且其群运算分别为$\circ_{\mathbb{G}}$和$\circ_{\mathbb{H}}$，并且称函数f为从$\mathbb{G}$到$\mathbb{H}$的同构，其中f和f^{-1}均可以高效计算（通常这是不必的）。对于$g_1, g_2 \in \mathbb{G}$，可以用两种方式计算$g = g_1 \circ_{\mathbb{G}} g_2$：第一种是直接用$\mathbb{G}$的群运算计算，第二种按下面的步骤计算：

(1) 计算$h_1 = f(g_1)$和$h_2 = f(g_2)$；

(2) 用$\mathbb{H}$的群运算计算$h = h_1 \circ_{\mathbb{H}} h_2$；

(3) 计算$g = g^{-1}(h)$。

哪一种方式更好取决于具体的群，以及计算f和f^{-1}的效率。

现在转到具体的例子，计算关于模N的乘法，其中$N = pq$是两个不同素数的乘积。中国剩余定理表明关于模N的模加和模乘运算，可以转换为类似的p和q取模运算。（此外，关于中国剩余定理的一个简单推论表明这些性质对幂运算也成立。）利用练习 7.25，可以给出当$N = 15$时的一些简单例子。

例 7.26 如果希望计算乘积$14 \cdot 13$对15取模（也就是说在$\mathbb{Z}_{15}^*$内）。练习 7.25 得到$14 \leftrightarrow (4,2)$，$13 \leftrightarrow (3,1)$。则

$$[14 \cdot 13 \bmod 15] \leftrightarrow (4,2) \cdot (3,1) = \big([4 \cdot 3 \bmod 5], [2 \cdot 1 \bmod 3]\big) = (2,2)$$

但是$(2,2) \leftrightarrow 2$是正确的答案是因为$14 \cdot 13 = 2 \bmod 15$。

例 7.27 如果要计算$11^2 \bmod 15$。例 7.25 给出$11 \leftrightarrow (1,2)$，所以$[11^2 \bmod 15] \leftrightarrow (1,2)^2$，其中右边幂运算在群$\mathbb{Z}_5^* \times \mathbb{Z}_3^*$内，因此

$$[11^2 \bmod 15] \leftrightarrow (1,2)^2 = (1^2 \bmod 5, 2^2 \bmod 3) = (1,1) \leftrightarrow 1$$

事实上，$11^2 = 1 \bmod 15$。

前面没有讨论如何在元素对N取模的表示法和元素对p或q取模的表示法之间来回转换的算法。如果知道N的分解形式，可以证明这种转换可以在多项式时间内实现。

很容易将一个元素x关于N取模映射到对应的表示法关于p或q取模；元素x对应$([x \bmod p], [x \bmod q])$。因为两种必要的模消减运算都可以高效地实现（参见附录 B.2），其过程可以在多项式时间内完成。

对另一方向的证明，应用下面的观察：一个元素用表示法(x_p, x_q)可以写为

$$(x_p, x_q) = x_p \cdot (1,0) + x_q \cdot (0,1)$$

所以，如果可以找到元素$1_p, 1_q \in \{0, \cdots, N-1\}$满足$1_p \leftrightarrow (1,0)$和$1_q \leftrightarrow (0,1)$。则（运用中国剩余定理）得知

$$(x_p, x_q) \leftrightarrow [(x_p \cdot 1_p + x_q \cdot 1_q) \bmod N]$$

因为p，q是互不相同的素数，$\gcd(p,q) = 1$。运用扩展欧几里得算法（参见附录 B.1.2），可得X, Y满足

$$Xp + Yq = 1$$

认为$1_p = [Yq \bmod N]$，这是因为

$$\Big[[Yq \bmod N] \bmod p\Big] = [Yq \bmod p] = [(1 - Xp) \bmod p] = 1$$

同时，

$$[Yq \bmod N] \bmod q = [Yq \bmod q] = 0$$

所以$Yq \bmod N \leftrightarrow (1,0)$。用相同的方式可证明$1_q = [Xp \bmod N]$。

综上所述，可以用下面的方式将一个元素的表示法(x_p, x_q)转化为关于模N的表示法（假设p，q都已知）：

(1) 计算X, Y，它们满足$Xp + Yq = 1$。

(2) 设$1_p = [Yq \bmod N]$，和$1_q = [Yp \bmod N]$。

(3) 计算$x = [(x_p \cdot 1_p + x_q \cdot 1_q) \bmod N]$。

注意到如果这个转换可以实现，则$1_p, 1_q$在预处理阶段计算一次，并多次利用。

例 7.28 设$p = 5, q = 7$，$N = 5 \cdot 7 = 35$。给出表示法$(4,3)$并希望将其转换为对应$\mathbb{Z}_{35}$中的元素。利用扩展欧几里得算法，计算

$$3 \cdot 5 - 2 \cdot 7 = 1$$

因此，$1_p = [(-2 \cdot 7) \bmod 35] = 21$且$1_q = [(3 \cdot 5) \bmod 35] = 15$。所以

$$\begin{aligned}(4,3) &= 4 \cdot (1,0) + 3 \cdot (0,1) \\ &\leftrightarrow [4 \cdot 1_p + 3 \cdot 1_q \bmod 35] \\ &= [4 \cdot 21 + 3 \cdot 15 \bmod 35] = 24\end{aligned}$$

因为$24 = 4 \bmod 5$，$24 = 3 \bmod 7$，所以计算正确。

例 7.29 计算$[29^{100} \bmod 35]$。首先计算对应式$29 \leftrightarrow ([29 \bmod 5], [29 \bmod 7]) = (-1, 1)$。利用中国剩余定理，可以得出

$$[29^{100} \bmod 35] \leftrightarrow (1, -1)^{100} = (1^{100} \bmod 5, (-1)^{100} \bmod 7) = (1, 1)$$

且很快可以得出$(1, 1) \leftrightarrow 1$。故$1 = [29^{100} \bmod 35]$。

例 7.30 计算$[18^{25} \bmod 35]$。由于$18 \leftrightarrow (3, 4)$，所以

$$18^{25} \bmod 35 \leftrightarrow (3, 4)^{25} = ([3^{25} \bmod 5], [4^{25} \bmod 7])$$

因为$\mathbb{Z}_5^*$是阶为 4 的群，可以“对幂指数关于 4 取模”（参见推论 7.15）得

$$3^{25} = 3^{25 \bmod 4} = 3^1 = 3 \bmod 5$$

类似地，

$$4^{25} = 4^{25 \bmod 6} = 4^1 = 4 \bmod 7$$

因此，$([3^{25} \bmod 5], [4^{25} \bmod 7]) = (3, 4) \leftrightarrow 18$，故$[18^{25} \bmod 35] = 18$。

7.2 素数、大数分解和 RSA

本节介绍据推测是“困难的”数论问题的第一个例子。从讨论一个很古老的问题开始：“大数分解”或“分解”。

给定一个合数N，大数分解问题是找到正整数p，q满足$pq = N$。大数分解是一个很经典的困难问题，既是因为此问题描述起来很简单，也是因为很长时间以来被认为是困难的计算问题（甚至其被用到密码学以前）。这个问题可以在指数时间$\mathcal{O}(\sqrt{N} \cdot \mathrm{polylog}(N))$内用试除法解决：也就是说，对所有$p = 2, \cdots, \lfloor \sqrt{N} \rfloor$穷举检查是否$p$整除$N$。（这个方法需要做$\sqrt{N}$次除法，每次需要时间$\mathrm{polylog}(N) = (\log N)^c$，其中$c$为常量。）这种方法通常可以成功，因为尽管$N$的最大的因数像$N/2$那么大，但是$N$最小的素因子最大为$\lfloor \sqrt{N} \rfloor$。尽管经过多年努力，已经知道了更好的运行时间的算法（参考第 8 章），但不存在多项式时间的算法可以解决这个分解问题。

考虑下面的实验，给定算法$\mathcal{A}$和参数n：

弱因子分解实验$\mathsf{w-Factor}_{\mathcal{A}}(n)$：

(1) 随机选择两个n比特整数x_1, x_2。

(2) 计算$N := x_1 \cdot x_2$。

(3) 给定N给$\mathcal{A}$，输出x_1', x_2'。

(4) 如果$x_1' \cdot x_2' = N$，实验输出定义为 1，其他情况输出 0。

刚才介绍因子分解问题被认为是困难的。但是这是否意味着：任何 PPT 算法$\mathcal{A}$，对于可忽视的函数negl，

$$\Pr[\mathsf{w-Factor}_{\mathcal{A}}(n) = 1] \leqslant \mathsf{negl}(n)$$

成立？完全不是。首先，上面实验中N为偶数的概率为 3/4（因为当x_1或者x_2为偶数时，N都为偶数），显然，在这种情况下$\mathcal{A}$分解N很容易。然而我们可以让$\mathcal{A}$的工作困难一点，要求$\mathcal{A}$的输出整数x_1', x_2'的长度为n（正如第 6 章中建议的），这样剩下的情况是x_1或x_2（从而N）含有小素因子的情况，这些小因子可被$\mathcal{A}$轻易地发现。在密码学领域，应该注意防止出现这种情况。

这个讨论意味着：“最难”分解的数是那些看上去只有大的素因子的数。这导致重新定义

上面的实验：x_1, x_2为n位随机素数，而不是n位随机整数。事实上，这个实验将在 7.2.3 节中正式定义因子分解假设时用到。这个实验在密码学中非常有用，但是，必须高效地产生n位随机素数。这是下一节的内容。

7.2.1 随机素数的产生

附录 B.2.4 中讨论了在确定范围内选择随机整数的一般方法，这一方法可以用于产生n位随机素数。（附录 B.2.4 中的讨论对下面的介绍很有用处，但并不是必不可少的。）特别地，可以重复产生n位随机整数直至找到第一个素数；重复至少t次。参考算法 7.31，它对这个过程进行了整体描述。

算法 7.31
产生一个随机素数的算法概要
输入：长度n，参数t
输出：一个n位随机素数
for $i = 1$ to t : {
  $p' \leftarrow \{0,1\}^{n-1}$
  $p := 1||p'$
if p 是素数，return p
}
return fail

注意到算法通过固定p的高位为 1 来迫使输出整数的位数恰为n（而不是最多为n）。本书约定整数长度为n，意味着其二进制表示形式的长度为n位，且最高有效位为 1。

假设有一种方法可以正确地确定一个给定的整数p是否为素数，则上面的算法可以随机输出一个n位素数，一旦输出不是fail的话。算法输出错误的概率由t决定，而这里的目的是希望设置t使错误的概率（以n表示）可忽略。为了证明这个方法可以高效（也就是说n的多项式时间内）地产生素数，需要很好地理解下面两个问题：①随机选择n位整数为素数的概率；②如何有效地测试一个给定的整数p为素数。接下来简要讨论这两个问题，第二个问题的深层次讨论将在 7.2.2 节中进行。

素数的分布。素数定理是数学中一个重要的结论。素数定理给出指定长度的整数是素数的比例的一个相当精确的界。对这里的目的而言，只需要很弱的版本：

定理 7.32 存在常量c，满足对任意的$n > 1$，n位素数的个数至少为$c \cdot 2^{n-1}/n$。

尽管已有初等证明方法，但此处并不给出这个定理的证明（参见本章后面的文献）。此定理意味着一个随机n位整数为素数的概率至少为

$$\frac{c \cdot 2^{n-1}/n}{2^{n-1}} = \frac{c}{n}$$

回到前面介绍的关于产生素数的方法，这意味着如果设置$t = n^2/c$，则在算法的t次迭代中仍没有某个素数被选中的概率至少为

$$\left(1-\frac{c}{n}\right)^t=\left(\left(1-\frac{c}{n}\right)^{n/c}\right)^n\leqslant\left(e^{-1}\right)^n=e^{-n}$$

（应用不等式 A.2），该值是（以n表示）可忽略的。因此，利用poly(n)次迭代得到错误的概率（以n表示）可忽略。

素性测试。有效判定一个给定整数p是否为素数的问题，有很长的历史。19 世纪 70 年代出现了第一个有效地用于素性测试的概率算法，这个算法有下列性质：如果给定输入p是一个素数，则输出通常是“素数”。另一方面，如果p是一个合数，则输出为“合数”，除了以p的长度表示的可忽略概率外。换句话说，这意味着如果输出为“合数”，则p一定是合数，但是如果输出是“素数”，则p很可能是素数，但也有可能p是合数[②]。

当把这种类型的素性测试方法应用到算法 7.31（前面介绍的素数产生算法）时，只要算法没有输出fail并且随机素数测试总是正确的，则算法的输出是一个有期望长度的随机素数。这意味着引入了另外一种错误源（除了输出fail的可能性外），算法现在可能错误地输出一个合数。因为可以确保这种情况发生的概率是可忽略的，这种概率没有实际意义，从而可安全地被忽略。

素数测试的多项式时间算法被认为是一个突破性的结果，出现在 2002 年。尽管这个算法可以多项式时间运行，但是比前面介绍的概率测试算法要慢。所以，概率素性测试算法仍然大量应用在实际中来产生大素数。

7.2.2 节中描述并且分析了最广泛应用之一的概率素数测试算法：Miller-Rabin算法。这个算法选择两个输入：一个用于素性测试的整数N和一个确定错误概率的参数t。Miller-Rabin算法运行时间为$||N||$和t的多项式时间，并且满足以下定理：

定理 7.33 工科如果N为素数，则Miller-Rabin测试总是输出“素数”。如果N为合数，则算法输出“素数”的概率至多为2^{-t}（并且输出正确答案“合数”的概率为$1-2^{-t}$）。

综合考虑。根据前面的讨论，现在可以描述一个多项式时间的素数产生算法：输入n，输出一个随机的n位素数（除了以n表示可忽略的概率外）。（在算法中，c是来自定理 7.32 的没有确定的常量。）过程的完整描述在算法 7.34 中介绍。

算法 7.34
产生一个随机素数
输入：长度参数n；
输出：一个随机n位素数
for $i=1$ **to** n^2/c : {
  $p'\leftarrow\{0,1\}^{n-1}$
  $p:=1||p'$
run Miller-Rabin 测试（输入为p和n）
if 输出是prime，**return** p
}
return fail

② 存在反向的概率素性测试：它们总是正确地判定合数，有时候错误地把素数当成合数。这里不考虑这种算法。

产生特定形式的素数。通常希望产生有特定形式的n比特随机素数p，例如满足$p = 3 \bmod 4$或$p = 2q + 1$，其中q也是素数（后面形式中的p称为强素数）。在这种情况下，将前面介绍的素数产生算法适当修改即可（例如，为了得到一个$p = 2q + 1$形式的素数，产生一个随机素数q，计算$p = 2q + 1$，如果p也是素数则输出p）。虽然这些修改算法在实际中能很好地应用，但是要严格证明它们的运行时间为多项式时间，且失败的概率可忽略，是非常复杂的（某些情况下，依靠未证明的关于具有特定形式的素数的密度的数论猜想）。关于这些问题的详细探究超出了本书范围，本书只简单地假定在需要的时候适当的素数产生算法是存在的。

7.2.2 *素数判定

现在描述Miller-Rabin素数判定算法，并且证明定理 7.33。这些资料在本书接下来的内容中并没有直接用到。

Miller-Rabin算法的关键是找到一个能区分素数与合数的属性。首先考虑下面的观察：如果N是素数，则$|\mathbb{Z}_N^*| = N - 1$，并由定理 7.14 对任意$a \in \{1, \cdots, N-1\}$可得$a^{N-1} = 1 \bmod N$。这意味着通过选择一个随机元素a并检查$a^{N-1} = 1 \bmod N$是否成立，可以测试给定整数N是否为素数。如果$a^{N-1} \neq 1 \bmod N$，则N不可能是素数。反之，如果N不是素数，则存在合理的概率选择a使其满足$a^{N-1} \neq 1 \bmod N$，所以，通过重复判定多次可以确定N是否为素数，或者以很高的确信度确定。上面的方法在算法 7.35 中给予说明。（回忆一下，关于模N的幂运算以及计算最大公约数，它们都可以在多项式时间内完成。选择$\{1, \cdots, N-1\}$的一个随机元素，也可以在多项式时间内完成。参考附录 B.2。）

算法 7.35
素性测试：初步方法
输入：整数N和参数t
输出：确定N是素数或合数
for $i = 1$ to t:
  $a \leftarrow \{1, \cdots, N-1\}$
  if $gcd(a, N) \neq 1$ **return** "composite"
  if $a^{N-1} \neq 1 \ mod \ N$ **return** "composite"
return "prime"

如果N是素数，则上面的讨论意味着此算法输出为"素数"。如果N为合数，算法在某个迭代中发现$a \in \mathbb{Z}_N^*$满足$a^{N-1} \neq 1 \bmod N$，算法输出"合数"。（即使$a \notin \mathbb{Z}_N^*$，算法也输出合数，将在后面介绍这种情况。）称具有这种性质的$a \in \mathbb{Z}_N^*$为"N是合数的证据"，或简称为"证据"。通常希望当N是合数时有很多证据，从而算法以"高"概率发现一个证据。这一直觉是正确的：只要至少有一个证据，就会有很多证据。在证明这一观点之前，需要用到两个群论的引理。

命题 7.36 设$\mathbb{G}$为有限群，且$\mathbb{H} \subseteq \mathbb{G}$。假设$\mathbb{H}$包含$\mathbb{G}$的单位元，并且对于任意$a, b \in \mathbb{H}$，

有$ab \in \mathbb{H}$。则$\mathbb{H}$是$\mathbb{G}$的一个子群。

证明 首先需要确认$\mathbb{H}$满足定义 7.9 的所有条件。$\mathbb{H}$的结合律可以自动从$\mathbb{G}$继承而来。通过假设，$\mathbb{H}$在群运算上存在单位元并且是封闭的。唯一需要验证的是，$\mathbb{H}$中的每个元素的逆元素仍然属于$\mathbb{H}$。设$\mathbb{G}$的阶为m（这里用到$\mathbb{G}$是有限群），考虑任意元素$a \in \mathbb{H}$。因为$a \in \mathbb{G}$，可以得到$1 = a^m = a \cdot a^{m-1}$。这意味着$a^{m-1}$为$a$的逆元素。因为$a \in \mathbb{H}$，$a^{m-1} \in \mathbb{H}$确保$\mathbb{H}$具有封闭性。 ■

引理 7.37 设$\mathbb{H}$为有限群$\mathbb{G}$的严格子群（也就是说$\mathbb{G} \neq \mathbb{H}$）。则$|\mathbb{H}| \leqslant |\mathbb{G}|/2$。

证明 设$\bar{h}$为$\mathbb{G}$的元素，且不属于$\mathbb{H}$；因为$\mathbb{H} \neq \mathbb{G}$，可知存在$\bar{h}$。考虑集合$\bar{\mathbb{H}} \stackrel{\text{def}}{=} \{\bar{h}h \mid h \in \mathbb{H}\}$。如果证明①$|\bar{\mathbb{H}}| = |\mathbb{H}|$；②$|\bar{\mathbb{H}}|$中的每个元素都不属于$\mathbb{H}$,即$\mathbb{H}$和$\bar{\mathbb{H}}$的交集为空集；则得证。这是因为$\mathbb{H}$和$\bar{\mathbb{H}}$都为$\mathbb{G}$的子集，意味着$|\mathbb{G}| \geqslant |\mathbb{H}| + |\bar{\mathbb{H}}| = 2|\mathbb{H}|$。

下面分别证明①和②。对于任意$h_1, h_2 \in \mathbb{H}$，如果$\bar{h}h_1 = \bar{h}h_2$，则两边同时乘以$\bar{h}^{-1}$，可得到$h_1 = h_2$。这就证明每个不同元素$h \in \mathbb{H}$对应一个不同的元素$\bar{h}h \in \bar{\mathbb{H}}$，从而证明①。

假设对于某个h，有$\bar{h}h \in \mathbb{H}$。这意味着对某个$h' \in \mathbb{H}$，有$\bar{h}h = h'$，所以$\bar{h} = h'h^{-1}$。因为$\mathbb{H}$是一个子群并且$h', h^{-1} \in \mathbb{H}$，因而有$h'h^{-1} \in \mathbb{H}$。但是这意味着$\bar{h} \in \mathbb{H}$，与$\bar{h}$的选择相矛盾。这样也就证明了②，从而引理得证。 ■

下面的定理可用来分析前面介绍的算法。

定理 7.38 选定N，存在一个N为合数的证据，则$\mathbb{Z}_N^*$中至少一半的元素是N为合数的证据。

证明：设Bad为集合$\mathbb{Z}_N^*$中不是证据的元素的集合；也就是说，$a \in$ Bad意味着$a^{N-1} = 1 \bmod N$。很显然$1 \in$ Bad，如果$a, b \in$ Bad，则$(ab)^{N-1} = a^{N-1} \cdot b^{N-1} = 1 \cdot 1 = 1 \bmod N$，因此$ab \in$ Bad。运用引理 7.36 可得Bad为$\mathbb{Z}_N^*$的一个子群。因为（根据假设）至少存在一个证据，Bad是$\mathbb{Z}_N^*$的一个严格子群。引理 7.37 表明$|\text{Bad}| \leqslant |\mathbb{Z}_N^*|/2$，所以$\mathbb{Z}_N^*$中至少一半元素不属于Bad（从而作为证据）。 ■

设N为合数。如果存在一个N为合数的证据，则至少存在$|\mathbb{Z}_N^*|/2$个证据。在算法的任何给定迭代中，找到一个证据或者一个元素不属于$\mathbb{Z}_N^*$的概率至少为

$$\frac{\frac{|\mathbb{Z}_N^*|}{2} + \left((N-1) - |\mathbb{Z}_N^*|\right)}{N-1} = 1 - \frac{|\mathbb{Z}_N^*|/2}{(N-1)} \geqslant 1 - \frac{|\mathbb{Z}_N^*|/2}{|\mathbb{Z}_N^*|} = \frac{1}{2}$$

所以，算法在t次迭代中都没有找到证据的概率（即算法错误地输出“素数”的概率）至多为2^{-t}。

不幸的是上面的论述并没有给出完整的答案，因为没有任何证据的合数N有无穷多个！具有此特性的N称为 Carmichael 数；更多细节方面的讨论超出了本书的范围。

令人高兴的是，对上面测试的改进对任何N都有效。设$N - 1 = 2^r u$，其中u为奇数且$r \geqslant 1$（给定N，很容易计算r, u。同时限制$r \geqslant 1$意味着N为奇数，但是当N为偶数时，素性测试是简单的。）前面给出的算法仅仅测试是否有$a^{N-1} = a^{2^r u} = 1 \bmod N$。一个更加精细的

算法考察$r+1$项的序列$(a^u, a^{2u}, \cdots, a^{2^r u})$（全都要对$N$取模）。序列中的每一项是前面项的平方；因此，如果其中某个值等于± 1，则序列中所有接下来的值都等于1。

如果①$a^u \neq \pm 1 \bmod N$，且②对于所有$i \in \{1, \cdots, r-1\}$，有$a^{2^i u} \neq -1 \bmod N$，则称$a \in \mathbb{Z}_N^*$是“$N$为合数的强证据”（或简称强证据）。如果$a$不是强证据，则$a^{2^{r-1}u} = \pm 1 \bmod N$，有

$$a^{N-1} = a^{2^r u} = \left(a^{2^{r-1}u}\right)^2 = 1 \bmod N$$

即a也不是N为合数的证据。换句话说，如果a是证据，则也是强证据，所以仅有的可能是强证据比证据多。也注意到如果元素a不是强证据，则序列$(a^u, a^{2u}, \cdots, a^{2^r u})$（全部都要对$N$取模）选择下面两种形式之一：

$$(\pm 1, 1, \cdots, 1) \text{或} (\star, \cdots, \star, -1, 1, \cdots, 1)$$

其中$\star$表示任意项。

首先介绍如果N为素数时，则不存在N为合数的强证据。为了达到此目的，依赖下面一个简单的引理（它是第11章中证明的命题11.1的特殊情形）：

引理 7.39 如果$x^2 = 1 \bmod N$，则称$x \in \mathbb{Z}_N^*$为“1模N的一个平方根”。如果N为奇素数，则1模N的仅有的平方根为$[\pm 1 \bmod N]$。

证明 很明显$(\pm 1)^2 = 1 \bmod N$。看到N是奇素数且$x^2 = 1 \bmod N$，其中$x \in \{1, \cdots, N-1\}$。则$0 = x^2 - 1 = (x+1)(x-1) \bmod N$，运用命题7.3可得$N \mid (x+1)$或$N \mid (x-1)$。只有当$x = [\pm 1 \bmod N]$时才可能出现这种情况。 ■

看到N是一个奇素数，选定任意$a \in \mathbb{Z}_N^*$。设$i \geqslant 0$为$a^{2^i u} = 1 \bmod N$的最小值；因为$a^{2^r u} = a^{N-1} = 1 \bmod N$，可知存在$i \leqslant r$。如果$i = 0$，则$a^u = 1 \bmod N$，$a$不是一个强证据。否则，

$$(a^{2^{i-1}u})^2 = a^{2^i u} = 1 \bmod N$$

$a^{2^{i-1}u}$为1的一个平方根。如果N为奇素数，1的平方根仅可能为± 1；选择的i使$a^{2^{i-1}u} \neq 1 \bmod N$，所以$a^{2^{i-1}u} = -1 \bmod N$，从而$a$不是强证据。于是当$N$为奇素数时不存在$N$的强证据。

如果对于某个素数$\hat{p}$和整数$e \geqslant 2$，有$N = \hat{p}^e$，则合数N是一个素数幂。现在说明每一个不是素数幂的奇合数N，存在“很多”强证据。

定理 7.40 设N为不是素数幂的奇合数，则$\mathbb{Z}_N^*$中至少一半的元素为N是合数的强证据。

证明 设$\mathsf{Bad} \subseteq \mathbb{Z}_N^*$表示不是强证据的元素的集合。定义集合$\mathsf{Bad}'$并指出：①$\mathsf{Bad}$为$\mathsf{Bad}'$的子集，而且②$\mathsf{Bad}'$为$\mathbb{Z}_N^*$的严格子群。这些能够成立是因为结合②和引理7.37可以得出$|\mathsf{Bad}'| \leqslant |\mathbb{Z}_N^*|/2$。此外，由①有$\mathsf{Bad} \subseteq \mathsf{Bad}'$，所以类似定理7.38可以得出$|\mathsf{Bad}| \leqslant |\mathsf{Bad}|' \leqslant |\mathbb{Z}_N^*|/2$。因此，$\mathbb{Z}_N^*$中至少一半的元素为强证据。（需要强调的是这里并没有声明Bad为$\mathbb{Z}_N^*$的子群。）

首先注意到$-1 \in \mathsf{Bad}$，因为$(-1)^u = -1 \bmod N$（还记得u是奇数）。设$i \in \{0, \cdots, r-1\}$是满足$a^{2^i u} = -1 \bmod N$的最大的整数（这里$a \in \mathsf{Bad}$）；或者，i满足

$$(a^u, a^{2u}, \cdots, a^{2^r u}) = (\underbrace{\star, \cdots, \star, -1}_{i+1 \text{ terms}}, 1, \cdots, 1)$$

的最大整数（这里$a \in \mathsf{Bad}$）。因为$-1 \in \mathsf{Bad}$且$(-1)^{2^0 u} = -1 \bmod N$，其中$i$已经定义。

就像前面那样选定i，并且定义

$$\mathsf{Bad}' \stackrel{\text{def}}{=} \{a \mid a^{2^i u} = \pm 1 \bmod N\}$$

现在开始证明前面的声明。

声称 7.41 $\mathsf{Bad} \subseteq \mathsf{Bad}'$。

设$a \in \mathsf{Bad}$对与某些$j \in \{0, \cdots, r-1\}$，有$a^u = 1 \bmod N$或$a^{2^j u} = -1 \bmod N$。第一种情况中，$a^{2^i u} = (a^u)^{2^i} = 1 \bmod N$且得出$a \in \mathsf{Bad}'$。第二种情况中，通过选择$i$有$j \leqslant i$。如果$j = i$，则很明显$a \in \mathsf{Bad}'$。如果$j < i$，则$a^{2^u} = (a^{2^i u})^{2^{i-j}} = 1 \bmod N$，从而$a \in \mathsf{Bad}'$。因为$a$是任意选择的，所以证明了$\mathsf{Bad} \subseteq \mathsf{Bad}'$。

声称 7.42 Bad'为$\mathbb{Z}_N^*$的子群。

很明显$1 \in \mathsf{Bad}'$。此外，如果$a, b \in \mathsf{Bad}'$，则

$$(ab)^{2^i u} = a^{2^i u} b^{2^i u} = (\pm 1)(\pm 1) = \pm 1 \bmod N$$

所以$ab \in \mathsf{Bad}'$。运用引理 7.36，可知Bad'是一个子群。

声称 7.43 Bad'为$\mathbb{Z}_N^*$的严格子群。

如果N是一个合数，且不是一个素数幂，则N可以写为$N = N_1 N_2$，其中$\gcd(N_1, N_2) = 1$。应用中国剩余定理，设符号$a \leftrightarrow (a_1, a_2)$表示将$a \in \mathbb{Z}_N^*$表示为$\mathbb{Z}_{N_1}^* \times \mathbb{Z}_{N_2}^*$；也就是，$a_1 = [a \bmod N_1]$且$a_2 = [a \bmod N_2]$。选择$a \in \mathsf{Bad}'$满足$a^{2^i u} = -1 \bmod N$，且$a \leftrightarrow (a_1, a_2)$。知道

$$(a_1^{2^i u}, a_2^{2^i u}) = (a_1, a_2)^{2^i u} \leftrightarrow a^{2^i u} = -1 \leftrightarrow (-1, -1)$$

所以

$$a_1^{2^i u} = -1 \bmod N_1 \text{且} a_2^{2^i u} = -1 \bmod N_2$$

考虑元素$b \in \mathbb{Z}_N^*$满足$b \leftrightarrow (a_1, 1)$。则

$$b^{2^i u} \leftrightarrow (a_1, 1)^{2^i u} = (a_1^{2^i u}, 1) = (-1, 1) \nleftrightarrow \pm 1$$

也就是说，$b^{2^i u} \neq \pm 1 \bmod N$。所以找到一个元素$b \notin \mathsf{Bad}'$。就像前面提到的，这就证明$\mathsf{Bad}'$是$\mathbb{Z}_N^*$的一个严格子群，所以运用引理 7.37，可得$\mathsf{Bad}'$的大小（以及Bad的大小）最多是$\mathbb{Z}_N^*$的 1/2，得证。 ■

如果$\hat{N}$为整数，$e \geqslant 2$，且有$N = \hat{N}^e$，则N为“完全幂”（这里不需要$\hat{N}$为素数，虽然任何素数幂也是完全幂）。现在可以完整描述一个素性测试算法。

算法 7.44
Miller-Rabin**素性测试**
输入：整数$N > 2$和参数t
输出：判定N是素数还是合数
if N是偶数，**return** “composite”
if N是完全幂次，**return** “composite”
computer $r \geqslant 1$和奇数u，满足$N-1=2^r u$
for $j = 1$ *to* t :
$a \leftarrow \{1, \cdots, N-1\}$
if $gcd(a, N) \neq 1$ **return** “composite”
if a是一个强证据，**return** “composite”
return “prime”

练习 7.11 和 7.22 要求读者说明测试N是否为完全幂，并且测试一个特定数a是否为强证据，且可以在多项式时间内完成。给出这些结果，算法运行时间很明显为关于$||N||$和t的多项式时间。现在可以完成定理 7.33 的证明。

证明 如果N为素数，则没有强证据，所以Miller-Rabin总是输出素数。如果N为合数，则存在两种情况：如果N是素数幂，算法通常输出合数。否则，由定理 7.40 发现，在任何迭代中找到一个强证据或者元素不属于$\mathbb{Z}_N^*$的概率至少为

$$\frac{|\mathbb{Z}_N^*|/2 + \left((N-1) - |\mathbb{Z}_N^*|\right)}{N-1} = 1 - \frac{|\mathbb{Z}_N^*|/2}{(N-1)} \geqslant 1 - \frac{|\mathbb{Z}_N^*|/2}{|\mathbb{Z}_N^*|} = \frac{1}{2}$$

所以，算法在t次迭代中都没有找到一个证据的概率（也就是算法错误地输出素数的概率）至多为2^{-t}。

7.2.3 因子分解假设

现在开始讨论如何产生随机素数，并形式化定义因子分解假设。设GenModulus为多项式时间算法，满足输入1^n，输出(N, p, q)，其中$N = pq$，除了（用n表示的）可忽略概率外，p, q都为n比特素数。考虑下面的实验：给定算法$\mathcal{A}$和参数n：

因子分解实验$\mathsf{Factor}_{\mathcal{A},\mathsf{GenModulus}}(n)$:
(1) 运行$\mathsf{GenModulus}(1^n)$得到(N, p, q)。
(2) $\mathcal{A}$被给定N，输出$p', q' > 1$。
(3) 如果$p' \cdot q' = N$则定义实验输出为 1，其他情况定义为 0。

当然，除可忽略的概率外，如果实验输出为 1，则$\{p', q'\} = \{p, q\}$。

定义 7.45 如果对于任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl满足

$$\Pr\left|\mathsf{Factor}_{\mathcal{A},\mathsf{GenModulus}}(n) = 1\right| \leqslant \mathsf{negl}(n)。$$

则称分解问题与GenModulus相关是困难的。

因子分解假设只是简单地假设：存在GenModulus，与之相关因子分解问题是困难的。一种构造恰当的GenModulus算法的自然方式是产生两个随机的且长度为n的素数p, q，并设N为它们的乘积；因子分解问题与这种形式的GenModulus有关，被认为是困难的。

7.2.4 RSA 假设

因子分解问题的研究已经有数百年的历史，但是一直没有发现效率很高的算法。所以似乎很合理地认为这个问题是困难的。令人遗憾的是，尽管因子分解假设产生单向函数（参见7.4.1 节），但是并不知如何从所描述的因子分解假设来产生实际的密码学构造方法。（但是在11.2.2 节中，将介绍一个非常有用的难题，其困难程度等价于因子分解。）因此就有了一种去寻找其他问题的动机，这些问题的困难程度与因子分解相关。目前所知最好的难题由 Rivest、Shamir 和 Adleman 提出，现在称为 RSA 问题。

$\mathbb{Z}_N^*$是阶为$\phi(N) = (p-1)(q-1)$的群。如果N的分解方法是已知的，则很容易计算出群的阶$\phi(N)$，并且关于模N的计算可以简化为在“指数对$\phi(N)$取模”（参考推论 7.15）。另一方面，如果不知道N的分解形式，则计算$\phi(N)$是困难的（事实上，计算$\phi(N)$和分解N难度相当。参见练习 7.13）。因此“指数对$\phi(N)$取模”不是一个可用的选择，至少不是一个明显的方式。RSA 问题利用这种不对称性：如果已知$\phi(N)$则 RSA 问题容易解决，但是如果不知道$\phi(N)$，则看起来很难解决。本节关注在不知道模N分解形式的情况下解决 RSA 问题的困难程度：如果已知N的分解形式则 RSA 问题变得很简单，这一事实对密码学应用非常有用，后面将说明这一点。

给定模N和一个与$\phi(N)$互素的整数$e > 0$，推论 7.22 说明e次幂模N是一个置换。从而（对任意$y \in \mathbb{Z}_N^*$）定义$y^{1/e} \bmod N$为$\mathbb{Z}_N^*$中满足$(y^{1/e})^e = y \bmod N$的唯一元素是有意义的。

下面正式描述 RSA 问题：给定N，一个与$\phi(N)$互素的整数$e > 0$，和一个元素$y \in \mathbb{Z}_N^*$，计算$y^{1/e} \bmod N$；也就是说，给定N, e, y找到x满足$x^e = y \bmod N$。正式地，设GenRSA为一个概率多项式算法，输入1^n，输出模N，即两个n比特素数的乘积，以及满足$\gcd(e, \phi(N)) = 1$的整数$e > 0$和满足$ed = 1 \bmod \phi(N)$的整数d。（因为e关于模$\phi(N)$可逆，所以存在d。）该算法失败的概率是（以n表示）可忽略的。给定算法$\mathcal{A}$和参数n，考虑下面的实验：

RSA 实验$\mathsf{RSA\text{-}inv}_{\mathcal{A},\mathsf{GenRSA}}(n)$：

(1) 运行$\mathsf{GenRSA}(1^n)$得到(N, e, d)。

(2) 选择$y \leftarrow \mathbb{Z}_N^*$。

(3) $\mathcal{A}$被给定N, e, y，输出$x \in \mathbb{Z}_N^*$。

(4) 如果$x^e = y \bmod N$，实验的输出定义为 1，否则定义为 0。

定义 7.46 如果对于任意概率多项式时间算法$\mathcal{A}$，存在一个忽略函数negl满足

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A},\mathsf{GenRSA}}(n) = 1] \leqslant \mathsf{negl}(n)。$$

则RSA问题与GenRSA相关是困难的。

RSA假设为：存在一个GenRSA，与之相关RSA问题是困难的。一个合适的算法GenRSA能够以任何算法GenModulus为基础来构造，GenModulus产生一个合数模连同其因子分解。下面介绍一个总体轮廓，唯一没有确定的事情为如何选择e。事实上有很多选择e的方式（使得RSA问题仍是困难的）；10.4.1 节将具体介绍一些选择e的特定方法。

算法 7.47
GenRSA算法（总体轮廓）
输入： 安全参数1^n
输出： 如前所述的N, e, d
$(N, p, q) \leftarrow \mathsf{GenModulus}(1^n)$
$\phi(N) = (p-1)(q-1)$
find e 满足 $\gcd(e, \phi(N)) = 1$
compute $d = [e^{-1} \bmod \phi(N)]$
return N, e, d

当GenRSA以上面方式构造时，哪些GenModulus算法可能使 RSA 问题是困难的？如果知道N的分解形式，则 RSA 问题很容易解决：首先计算$\phi(N)$，然后计算$d = [e^{-1} \bmod \phi(N)]$，最后计算$[y^d \bmod N]$。根据推论 7.22 可以得到正确的答案。若 RSA 问题是困难的，那么对GenModulus输出的N进行因子分解是不可行的。因此，如果 RSA 问题与GenRSA相关是困难的，那么因子分解问题和GenModulus相关是困难的。也就是说，RSA 问题不会比因子分解问题更难。

反过来情况会如何？若N是两个素数的乘积，N的分解可以用$\phi(N)$高效地计算出来（参见练习 7.13）。所以分解N与计算$\phi(N)$是等难度的。事实上，可以得到更多：给定满足$ed = 1 \bmod \phi(N)$的N, e, d，可以在概率多项式时间内计算分解N；练习 7.14 是这个结论的一个简单例子。目前没有证明：必须明确计算出$\phi(N)$或者d的才能解决 RSA 问题。因而根据当前的知识，不能认为 RSA 问题和分解问题一样难，所以，假设 RSA 问题是困难的和假设因子分解问题是困难的，前者看起来是更强的假设。（也就是说，即使因子分解问题不可能在多项式时间解决，但 RSA 问题仍可能在多项式时间解决。）

尽管如此，当GenRSA以模产生算法GenModulus为基础来构造时，即算法 7.47（也就是说，通过选择N，其为两个n比特随机素数的乘积），当分解问题与GenModulus相关是困难的，则 RSA 问题与GenRSA相关是困难的。

7.3 循环群中的假设

这一节将介绍一类关于循环群中的密码学困难性假设。首先讨论一些必要的背景知识。

7.3.1 循环群和生成元

设$\mathbb{G}$为阶为m的有限群。对于任意元素$g \in \mathbb{G}$，考虑集合

$$\langle g \rangle \stackrel{\text{def}}{=} \{g^0, g^1, \cdots, \}。$$

由定理 7.14，可得$g^m = 1$。设$i \leqslant m$为满足$g^i = 1$的最小整数。上面序列重复i项之后（也就是说，$g^i = g^0, g^{i+1} = g^1$，等等），可得

$$\langle g \rangle = \{g^0, \cdots, g^{i-1}\}。$$

可知$\langle g \rangle$至少包含i个元素。事实上，也恰好包含i个元素，因为如果$g^j = g^k$，其中$0 \leqslant j < k < i$，则$g^{k-j} = 1$和$0 < k - j < i$，与选择的i相矛盾。

不难验证：对任意g，$\langle g \rangle$是$\mathbb{G}$的一个子群，（参见练习 7.3），称$\langle g \rangle$为由g产生的子群。如果子群$\langle g \rangle$的阶为i，则i称作g的阶。也就是说：

定义 7.48 设$\mathbb{G}$为有限群，$g \in \mathbb{G}$。g的阶为满足$g^i = 1$的最小整数i。

下面介绍的是推论 7.15 的一个类比（证明相同）：

命题 7.49 设$\mathbb{G}$为一个有限群，$g \in \mathbb{G}$且其阶为i。则对任意整数x，有$g^x = g^{[x \bmod i]}$。

实际上可以证明更强的结果。

命题 7.50 设$\mathbb{G}$为有限群，$g \in \mathbb{G}$是阶为i的元素。则当且仅当$x = y \bmod i$时，有$g^x = g^y$。

证明 如果$x = y \bmod i$，则$[x \bmod i] = [y \bmod i]$，并且根据前面的命题有

$$g^x = g^{[x \bmod i]} = g^{[y \bmod i]} = g^y。$$

反方向的证明更有趣。即若$g^x = g^y$，设$x' = [x \bmod i]$和$y' = [y \bmod i]$；由前面的命题有$g^{x'} = g^{y'}$，即$g^{x'}(g^{y'})^{-1} = 1$。不失一般性，如果$x' \neq y'$，可以假设$x' > y'$。由于x'、y'都要小于i、$x' - y'$的差是一个比i小的非零整数。但是

$$1 = g^{x'} \cdot \left(g^{y'}\right)^{-1} = g^{x'-y'}$$

与i为g的阶相矛盾。 ■

任意群$\mathbb{G}$的单位元的阶为 1，生成群$\langle 1 \rangle = \{1\}$，且是唯一阶为 1 的元素。另外一个极端，如果存在一个元素$g \in \mathbb{G}$，且阶为m（m为$\mathbb{G}$的阶），则$\langle g \rangle = \mathbb{G}$。此时，称$\mathbb{G}$为循环群，并且称$g$为$\mathbb{G}$的一个生成元。（注意到一个循环群可能有多个生成元，所以不能称$\mathbb{G}$的生成元为g。）如果g是$\mathbb{G}$的一个生成元，则通过定义，每一个元素$h \in \mathbb{G}$，对于某个$x \in \{0, \cdots, m-1\}$等于g^x，下一节中将讨论这一点。

群$\mathbb{G}$中不同的元素可能有不同的阶。然而，可对这些可能的阶进行限制。

命题 7.51 设$\mathbb{G}$是阶为m的有限群，若元素$g \in \mathbb{G}$的阶为i，则$i \mid m$。

证明 通过定理 7.14，可以发现$g^m = 1$。因为g的阶为i，通过命题 7.49 可以得到$g^m = g^{[m \bmod i]}$。如果i不能整除m，则$i' \stackrel{\text{def}}{=} [m \bmod i]$为一个比$i$小的正整数，其中$g^{i'} = 1$。因为$i$为$g$的阶，故这是不可能的。 ■

接下来的推论说明了这个结论的影响：

推论 7.52 如果$\mathbb{G}$是一个阶为素数p的群，则$\mathbb{G}$为循环群。此外，$\mathbb{G}$中的元素除单位元外

都是$\mathbb{G}$的生成元。

证明 由命题 7.51 可知，群$\mathbb{G}$中元素可能的阶只能为 1 或p。只有单位元阶为 1，所以所有其他元素的阶都为p，且可以生成$\mathbb{G}$。 ■

以素数为阶的群形成一类循环群。加法群$\mathbb{Z}_N$，对于$N > 1$是另外一个关于循环群的例子（元素 1 通常为生成元）。下一个定理介绍了一类重要的加法循环群；尽管证明超出了本书的范围，但是可以在任何标准的抽象代数课本中找到。

定理 7.53 如果p为素数，则$\mathbb{Z}_p^*$为循环群。

$p > 3$的素数，$\mathbb{Z}_p^*$没有素数阶，所以上面内容并不遵循前面的推论。

举一些例子可以帮助说明前面的讨论。

例 7.54 考虑（加法）群$\mathbb{Z}_{15}$。已经注意到，$\mathbb{Z}_{15}$是一个循环群，且元素 1 是一个生成元：因为$15 \cdot 1 = 0 \bmod 15$且对任意$0 < i' < 15$，$i' \cdot 1 = i' \neq 0 \bmod 15$（回忆一下，此群的单位元为 0）。

$\mathbb{Z}_{15}$还有其他生成元。例如$\langle 2 \rangle = \{0, 2, 4, \cdots, 14, 1, 3, 5, \cdots, 13\}$，所以 2 也是一个生成元。

并不是所有元素都可以生成$\mathbb{Z}_{15}$。例如，因为$5 \cdot 3 = 0 \bmod 15$，元素3的阶为5，所以3不是$\mathbb{Z}_{15}$的一个生成元。子群$\langle 3 \rangle$由5个元素$\{0, 3, 6, 9, 12\}$组成，这事实上是一个模15的加法子群。因为$3 \cdot 10 = 0 \bmod 15$，元素10的阶为3，且子群$\langle 10 \rangle$有3个元素$\{0, 5, 10\}$组成。注意到就像命题7.51要求的5和3都可以整除$|\mathbb{Z}_{15}| = 15$。

例 7.55 考虑（乘法）群$\mathbb{Z}_{15}^*$，其阶为$(5-1) \times (3-1) = 8$。因为$\langle 2 \rangle = \{1, 2, 4, 8\}$，所以 2 的阶为 4。就像命题 7.51 要求的，4 整除 8。

例 7.56 考虑阶为素数p的群$\mathbb{Z}_p$。已经知道此群为循环群，但推论 7.52 给出了更多性质：也就是说，除了 0 之外的每一个元素都为生成元。事实上，当且仅当$p \mid ih$时，对任何元素$h \in \{1, \cdots, p-1\}$和整数$i > 0$，可得$ih = 0 \bmod p$。但由命题 7.3，有$p \mid h$或$p \mid i$。前者不可能发生（因为$h < p$），后者发生的最小正整数为$i = p$。根据推论 7.52，得出任何非零元素h的阶为p（因而生成$\mathbb{Z}_p$）。

例 7.57 考虑群$\mathbb{Z}_7^*$，由定理 7.53 可知其为循环群。因为$\langle 2 \rangle = \{1, 2, 4\}$，故 2 不是一个生成元。然而

$$\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\} = \mathbb{Z}_7^*$$

所以 3 是$\mathbb{Z}_7^*$的一个生成元。

下面的例子要用 7.1.5 节中介绍的内容。

例 7.58 设$\mathbb{G}$是阶为n的循环群，且g为$\mathbb{G}$的一个生成元。则映射$f: \mathbb{Z}_n \to \mathbb{G}$，定义为$f(a) = g^a$是$\mathbb{Z}_n$和$\mathbb{G}$之间的同构。事实上，对于$a, a' \in \mathbb{Z}_n$，可得

$$f(a + a') = g^{[a+a' \mod n]} = g^{a+a'} = g^a \cdot g^{a'} = f(a) \cdot f(a')$$

运用n是g的阶可以证明f为双射。

上面的例子说明所有等阶的循环群在“代数”意义上是相同的。在“计算”意义上并不相同，特别在同构函数$f^{-1}: \mathbb{G} \to \mathbb{Z}_n$（必须存在）并不需要高效地计算。此外，即使$\mathbb{Z}_p^*$（$p$为素数）与群$\mathbb{Z}_{p-1}$同构，这两个群运算的计算复杂度可能区别很大。在第 8 章中将继续介绍这一点。

7.3.2 离散对数和 Diffie-Hellman 假设

现在介绍一些可以定义在任意循环群上的计算问题。本节中对这些问题进行抽象讨论，在 7.3.3 节和 7.3.4 节中将使用群的具体例子介绍被认为是困难的问题。

如果$\mathbb{G}$是阶为q的循环群，则存在一个生成元$g \in \mathbb{G}$满足$\{g^0, g^1, \cdots, g^{q-1}\} = \mathbb{G}$。等价的，对任意$h \in \mathbb{G}$，存在唯一$x \in \mathbb{Z}_q$满足$g^x = h$。当群$\mathbb{G}$已知，称$x$为关于$g$的$h$的离散对数，记作$x = \log_g h$。注意，对任意整数$x'$，如果$g^{x'} = h$，则$\log_g h = [x' \bmod q]$。强调这种情况下的对数称为“离散”，因为值在有限范围内，相反“标准”对数是因为值的范围为无穷集合。

离散对数服从很多“标准”对数的规则。例如，$\log_g 1 = 0$（其中 1 是$\mathbb{G}$的单位元）并且$\log_g(h_1 \cdot h_2) = [(\log_g h_1 + \log_g h_2) \bmod q]$。

给定一个生成元为g的循环群$\mathbb{G}$，其离散对数问题为：给定随机元素$h \in \mathbb{G}$作为输入，计算$\log_g h$。正式的说，设$\mathcal{G}$为多项式时间算法，输入1^n，输出一个循环群$\mathbb{G}$（的描述），其阶为q（$||q|| = n$），生成元$g \in \mathbb{G}$。同时，也需要群运算可以高效计算（也就是说，以n表示的多项式时间）。给定群生成算法$\mathcal{G}$，算法$\mathcal{A}$，和参数n，考虑下面实验：

离散对数实验$\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$：

(1) 运行$\mathcal{G}(1^n)$而得到$(\mathbb{G}, q, g)$，其中$\mathbb{G}$是一个阶为q的循环群（$||q|| = n$），g为$\mathbb{G}$的一个生成元。

(2) 选择$h \leftarrow \mathbb{G}$（可以通过选择$x' \leftarrow \mathbb{Z}_q$，并且设置$h := g^{x'}$来完成）。

(3) $\mathcal{A}$给定$\mathbb{G}, q, g, h$，输出$x \in \mathbb{Z}_q$。

(4) 如果$g^x = h$，定义实验输出为 1，其他情况下输出为 0。

定义 7.59 如果对于任意概率多项式算法$\mathcal{A}$，存在一个可忽略函数negl满足

$$\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leqslant \mathsf{negl}(n)$$

则离散对数问题与$\mathcal{G}$相关是困难的。

离散对数假设是简单地假设存在$\mathcal{G}$，与之相关离散对数问题是困难的。下面两节讨论一些满足此种情况下的候选的群生成算法$\mathcal{G}$。

有些非常有用的与计算离散对数相关的问题，即所谓Diffie-Hellman问题。这种问题有两种变化形式：计算Diffie-Hellman问题（CDH）和判定Diffie-Hellman问题（DDH），尽管CDH问题在本书后面没有用到，但是介绍这些知识将非常有教育意义，至少是在讨论DDH问题之前。

选定循环群$\mathbb{G}$，以及一个生成元$g \in \mathbb{G}$。给定两个元素h_1和h_2，定义$\mathsf{DH}_g(h_1, h_2) \stackrel{\mathrm{def}}{=} g^{\log_g h_1 \cdot \log_g h_2}$。也就是说，如果$h_1 = g^x$且$h_2 = g^y$，则

$$\mathsf{DH}_g(h_1, h_2) = g^{x \cdot y} = h_1^y = h_2^x。$$

CDH问题为：当随机选定h_1和h_2时计算出$\mathsf{DH}_g(h_1, h_2)$。

如果离散对数问题关于某些$\mathcal{G}$是容易的，则CDH问题也是如此，因为对给定h_1和h_2，可以首先计算出$x = \log_g h_1$，然后输出结果h_2^x。相反，并不知道离散对数问题的困难性是否意味着CDH问题的困难性。

简单而言，DDH问题就是对一个群中随机选择的元素h_1和h_2，把一个随机群元素和$\mathsf{DH}_g(h_1, h_2)$相区分。也就是说，给定随机选择的h_1、h_2和候选答案h'，确定是$h' = \mathsf{DH}_g(h_1, h_2)$，还是$h'$从$\mathbb{G}$中随机选择。正式地，设$\mathcal{G}$如上所述，有：

定义 7.60 如果对任意概率多项式时间算法$\mathcal{A}$存在一个可忽略函数negl，满足

$$\left|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]\right| \leqslant \mathsf{negl}(n)$$

则DDH与$\mathcal{G}$相关是困难的。概率来源于实验中$\mathcal{G}(1^n)$的输出$(\mathbb{G}, q, g)$，以及随机选择的$x, y, z \in \mathbb{Z}_q$。

注意到当z从$\mathbb{Z}_q$中随机选择时，元素g^z在$\mathbb{G}$中是独立均匀分布的。

已经发现如果离散对数问题对某些$\mathcal{G}$是简单的，则CDH问题也同样如此。类似的，如果CDH问题对某些$\mathcal{G}$是简单的，则DDH问题也是如此；练习 7.16 要求读者给出证明。但是反过来并不正确，存在某些群，即使该群中DDH问题是简单的，但离散对数问题和CDH问题仍然是困难的，参见练习 11.10。

素数阶群的使用。存在大量类别的循环群，在这些群中离散对数问题与Diffie-Hellman问题被认为是困难的。虽然非素数阶循环群也被用于某些密码学应用中，但是一般优先使用素数阶的循环群。这是因为下面解释的多种原因。

相比起来使用素数阶群的第一个原因是，某种意义上说，这些群中的离散对数问题是最困难的。特别是第 8 章将要介绍的Pohlig-Hellman算法，一个阶为$q = q_1 \cdot q_2$的群的离散对数问题规约为两个阶分别为q_1和q_2的群的离散对数问题。（假设q的分解形式已知，如果q有较小的素数因子时，找到q的非平凡因子比较简单。）强调这并不意味着非素数阶群的离散对数问题是简单的（可以在多项式时间内解决）；这仅仅意味着问题变得相对简单（至少对当前已知的算法而言）。这解释了为什么素数阶群更加理想。

使用素数阶群的第二个原因是，非常容易找到这种群的生成元，正如测试给定元素是否为生成元一样。根据推论 7.52，素数阶群的每个元素（单位元除外）都是生成元。即使可以在多项式时间内找到任意循环群一个生成元（参考附录 B.3），使用一个素数阶群可能产生一个更高效的算法$\mathcal{G}$（前面提到过该算法需要计算群$\mathbb{G}$的生成元g）。

对于一些密码学构造方法，安全性证明需要计算某些指数的乘法逆元（7.4.2 节中将举一个例子）。当群的阶为素数q时，则任何非零指数对模q可逆，从而使计算成为可能。

最后一个使用素数阶群的原因针对当判定Diffie-Hellman问题为困难问题时的情形。选定一个阶为q的群$\mathbb{G}$，则DDH问题归结为区别$(h_1, h_2, \mathsf{DH}_g(h_1, h_2))$的三元组形式（$h_1$、$h_2$为随机选择）和三元组$(h_1, h_2, y)$（其中$h_1$、$h_2$、$y$为随机选择）。DDH问题是困难的，因为一个

必需条件是$\mathsf{DH}_g(h_1,h_2)$自身应该与一个随机群元素不可区分。当h_1,h_2随机选择时，如果$\mathsf{DH}_g(h_1,h_2)$为一个随机群元素则最为理想[3]。当群的阶q为素数时，这（几乎）是正确的。为了说明这一点，首先证明：

命题 7.61 设$\mathbb{G}$是群，其阶为素数q且有生成元g。如果x_1和x_2均匀从$\mathbb{Z}_q$中选择，则

$$\Pr[\mathsf{DH}_g(g^{x_1},g^{x_2})=1]=1-\left(1-\frac{1}{q}\right)^2=\frac{2}{q}-\frac{1}{q^2}$$

而且对任何其他值$y\in\mathbb{G},y\neq 1$：

$$\Pr[\mathsf{DH}_g(g^{x_1},g^{x_2})=y]=\frac{1}{q}\cdot\left(1-\frac{1}{q}\right)=\frac{1}{q}-\frac{1}{q^2}$$

证明 首先，运用$\mathsf{DH}_g(g^{x_1},g^{x_2})=g^{[x_1\cdot x_2 \bmod q]}$。因为$q$为素数，当且仅当$x_1=0$或$x_2=0$时，$[x_1\cdot x_2 \bmod q]=0$。因为$x_1$和$x_2$从$\mathbb{Z}_q$中均匀选择，

$$\begin{aligned}\Pr[\mathsf{DH}_g(g^{x_1},g^{x_2})=1]&=\Pr[x_1=0\vee x_2=0]\\&=1-\Pr[x_1\neq 0]\cdot\Pr[x_2\neq 0]=1-\left(1-\frac{1}{q}\right)^2\end{aligned}$$

选定任意$y\in\mathbb{G},y\neq 1$，并且设$x=\log_g y\neq 0$。注意到当且仅当$x_1x_2=x \bmod q$时$\mathsf{DH}_g(g^{x_1},g^{x_2})=y$。因为$q$为素数，所有$\mathbb{Z}_q$中非零元素有模$q$的乘法逆元，并且当且仅当$x_1$不为 0 且$x_2=x\cdot(x_1)^{-1} \bmod q$时$x_1x_2=x \bmod q$。所以：

$$\begin{aligned}\Pr[\mathsf{DH}_g(g^{x_1},g^{x_2})=y]&=\Pr[x_1x_2=x \bmod q]\\&=\Pr[x_2=x\cdot(x_1)^{-1} \bmod q\mid x_1\neq 0]\cdot\Pr[x_1\neq 0]\\&=\frac{1}{q}\cdot\left(1-\frac{1}{q}\right)\end{aligned}$$

得证。 ■

现将其与$\mathbb{G}$上的均匀分布相比较。一个均匀分布的元素y'对任意$y\in\mathbb{G}$（即包括当$y=1$的情况）有$\Pr[y'=y]=1/q$。当$||q||=n$（所以$q=\Theta(2^n)$）时上面命题表明：对于均匀分布的h_1和h_2，有

$$\Pr[\mathsf{DH}_g(h_1,h_2)=y]=\frac{1}{q}\pm\mathsf{negl}(n)$$

从这个意义上说，$\mathsf{DH}_g(h_1,h_2)$近似于$\mathbb{G}$中的均匀分布。尽管如此，使用素数阶群对 DDH 问题是困难的这一断言来说即不是必要的，也不是充分的（事实上，DDH 问题对于加法群$\mathbb{Z}_p$很简单，其中p为素数）。相反，这些仅仅是一些附加的、启发式的原因说明为什么素数阶群受欢迎。

7.3.3 在 Z_p^*（的子群）中工作

p为素数的群$\mathbb{Z}_p^*$，是一类循环群，在该群中离散对数问题被认为是困难的。具体而言，设$\mathcal{G}_1$为一个算法，其输入为1^n，输出随机选择的一个n位素数p，群$\mathbb{Z}_p^*$的阶$q=p-1$，其生成元为g。（7.2.1 节中讨论了一个选择随机素数的高效算法，附录 B.3 证明了如何高效地发现群

[3] 牢记$\mathsf{DH}_g(h_1,h_2)$的分布和给定h_1,h_2的$\mathsf{DH}_g(h_1,h_2)$的条件分布之间的区别是很重要的。因为$\mathsf{DH}_g(h_1,h_2)$是一个关于h_1和h_2的确定函数，后者的分布使正确回答$\mathsf{DH}_g(h_1,h_2)$的概率为 1,因而远不是均匀分布。这里感兴趣的是当h_1,h_2是随机且未知时$\mathsf{DH}_g(h_1,h_2)$的分布。

$\mathbb{Z}_p^*$的生成元。）于是离散对数问题与$\mathcal{G}_1$相关是困难的。

循环群$\mathbb{Z}_p^*$（对$p > 3$的素数而言）没有素数阶。（优先考虑素数阶群的原因在上一节中已介绍。）更大的问题是，判定Diffie – Hellman问题在这些群中并不是困难的（参考第 11 章的练习 11.10），因此不能用于在第 9 章和第 10 章中讨论的密码学应用。

幸运的是，这些问题可以通过使用$\mathbb{Z}_p^*$中适当的“子群”来相对简单地解决。如果存在$x \in \mathbb{Z}_p^*$满足$x^2 = y \bmod p$，则元素$y \in \mathbb{Z}_p^*$为模p平方剩余。不难证明模p平方剩余的集合形成了一个$\mathbb{Z}_p^*$的子群。更多地，当p为素数时，可以证明模p平方是一个二对一函数，意味着事实上$\mathbb{Z}_p^*$中一半的元素为平方剩余。11.1.1 节给出对此事实的证明，该节给出关于模素数p的二次剩余的进一步讨论。

如果p是一个强素数，也就是说，$p = 2q + 1$，q为素数，则关于模p二次剩余的子群恰有$(p-1)/2 = q$个元素。因为q为素数，推论 7.52 表明这个子群为循环群，并且此群中所有元素（单位元除外）都为生成元。为了完整地给出介绍，下面粗略介绍一个满足上面讨论的容易理解的多项式时间算法$\mathcal{G}_2$。

算法 7.62
群生成算法$\mathcal{G}_2$
输入： 安全参数1^n
输出： 循环群$\mathbb{G}$，其阶为q，生成元g
产生一个随机$n+1$比特强素数p
$q := (p-1)/2$
选择一个任意满足$x \neq \pm 1 \bmod p$的$x \in \mathbb{Z}_p^*$，且设$g := x^2 \bmod p$
返回p, q, g

注意到p的作用是表明群$\mathbb{G}$为对p取模的二次剩余。对上面介绍的$\mathcal{G}_2$产生的群而言，DDH问题是困难的。

强素数p可以按照 7.2.1 节中介绍的方法来产生。注意到g的计算方式取保了其为模p的二次剩余且$g \neq 1$，同时g是模p二次剩余子群的生成元。

7.3.4 *椭圆曲线群

到目前为止已见过的群都基于模算术。另一类感兴趣的群是由椭圆曲线上的点组成的。这些群可用于密码学应用是因为和$\mathbb{Z}_p^*$相反，目前没有亚指数时间算法来解决该群中的离散对数问题（参考第 8 章进一步的讨论）。尽管椭圆曲线群在密码学实际应用中非常重要，但是本书对这种群的介绍很少，这是因为：

(1) 对椭圆曲线群深层次的理解所需要的数学知识，比起本书对读者要求的要多。因此本书关于椭圆曲线的介绍相当少，省去一般性换来简洁性。对这个话题深层次的探讨感兴趣的读者建议查阅本章后面的参考文献。

(2) 大部分基于椭圆曲线群的密码学方案（以及所有本书中的方案），都能以完全通用的方式来对待相应的群进行分析和理解，而不用考虑具体实现方案的特定的群。例如，将在后

面的几章中介绍的密码学方案，基于任意的循环群（可能是素数阶）；只要对应的群中的某些计算问题是困难的，则这些方案是安全的。由可证明安全的观点，群如何实例化不会产生区别（只要群中相关的计算问题被认为是困难的）。当然，在实践中实现方案时，群的具体选择是很重要的。

现在开始简要介绍椭圆曲线。令$p \geqslant 5$为素数[④]。考虑有变量x和y的方程E，形式为

$$y^2 = x^3 + Ax + B \bmod p \tag{7.1}$$

其中$A, B \in \mathbb{Z}_p$为满足$4A^3 + 27B^2 \neq 0 \bmod p$的常数（后面这个条件确保了方程$x^3 + Ax + B = 0 \bmod p$没有重根）。令$\hat{E}(\mathbb{Z}_p)$表示满足上面方程的$(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$数对的集合，也就是说

$$\hat{E}(\mathbb{Z}_p) \stackrel{\text{def}}{=} \{(x, y) \mid x, y \in \mathbb{Z}_p \text{且} y^2 = x^3 + Ax + B \bmod p\}$$

定义$E(\mathbb{Z}_p) \stackrel{\text{def}}{=} \hat{E}(\mathbb{Z}_p) \cup \{\mathcal{O}\}$，其中，$\mathcal{O}$是一个特殊值，其目的将简要介绍。集合$E(\mathbb{Z}_p)$的元素称为在由等式（7.1）定义的“椭圆曲线$E$”上的点，$\mathcal{O}$称为“无穷远处的点”。

例 7.63 前面曾经提到过，如果存在$x \in \mathbb{Z}_p^*$满足$x^2 = y \bmod p$，则元素$y \in \mathbb{Z}_p^*$是模p的二次剩余；此时，称x为y的平方根。此外，当$p > 2$且为素数时，每个模p二次剩余都恰有两个平方根。（参见 11.1.1 节更深入的讨论。）

令$f(x) \stackrel{\text{def}}{=} x^3 + 3x + 3$并且考虑曲线$E : y^2 = f(x) \bmod 7$。每个使$f(x)$为模 7 二次剩余的$x$值产生曲线上的两个点，使$f(x)$是非二次剩余的$x$值则不在曲线上，使$f(x) = 0 \bmod 7$的$x$给出曲线上的一个点。这样，可以决定曲线上的点：

- $f(0) = 3 \bmod 7$，一个模 7 二次非剩余。
- $f(1) = 0 \bmod 7$，所以得到点$(1, 0) \in E(\mathbb{Z}_7)$。
- $f(2) = 3 \bmod 7$，一个模 7 的非二次剩余。
- $f(3) = 4 \bmod 7$，一个模 7 的二次剩余，两个平方根为 2 和 5。因此产生两个点$(3, 2), (3, 5) \in E(\mathbb{Z}_7)$。
- $f(4) = 2 \bmod 7$，一个模 7 的二次剩余，两个平方根为 3 和 4。因此产生两个点$(4, 3), (4, 4) \in E(\mathbb{Z}_7)$。
- $f(5) = 3 \bmod 7$，一个模 7 的非二次剩余。
- $f(6) = 6 \bmod 7$，一个模 7 的非二次剩余。

包括无穷远的点，$E(\mathbb{Z}_7)$中有6个点。

一个有用的概念化$E(\mathbb{Z}_p)$的方法为看方程（7.1）在实数域上的图形（也就是说，$y^2 = x^3 + Ax + B$方程没有对p取模），如图 7.2 所示。该图没有和$E(\mathbb{Z}_p)$精确对应，这是因为，比方说$E(\mathbb{Z}_p)$有有限个点（$\mathbb{Z}_p$是一个有穷集合），然而如果允许x和y的范围为所有实数，则对于同样的方程有无穷个解。虽然如此，该图提供了有用的直观感受。可以认为图中“无穷远处的点”$\mathcal{O}$在坐标中的位置为在y轴的顶部，且位于每条垂直线旁。

---

④ 理论上可以处理$p = 2$或者 3 的情况，但是这将引入更多复杂性。对高级读者而言，知道椭圆曲线能够在任何（有限或者无限）域上定义，这里讨论特征不为 2 和 3 的域。

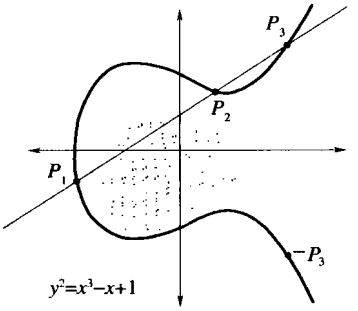


图 7.2　实数域上的椭圆曲线

可以证明每条和曲线E相交的直线与E有三个交点：①如果直线和曲线在P点相切，则点P计算两次。②无穷远处的点也要计算在内（当直线垂直时）。据此可以定义一个二元运算，称为“加”并用符号“+”表示，对$E(\mathbb{Z}_p)$上的点用下面的方式操作[⑤]：

- 点$\mathcal{O}$定义为（加法）单位元。也就是说，对于所有$P \in E(\mathbb{Z}_p)$，定义$P+\mathcal{O}=\mathcal{O}+P=P$。
- 如果P_1, P_2, P_3是E上线性相关的点，则有

$$P_1 + P_2 + P_3 = \mathcal{O} \tag{7.2}$$

（这里不考虑P_1, P_2, P_3的顺序，意味着加法对所有的点是满足交换律，对线性相关的点满足结合律。）

取反以及任意点的加法运算如下。

取反。给定点P，$-P$为（根据取反的定义）点，满足条件$P+(-P)=\mathcal{O}$。如果$P=\mathcal{O}$则$-P=\mathcal{O}$。否则，因为$P+(-P)+\mathcal{O}=(P+(-P))+\mathcal{O}=\mathcal{O}+\mathcal{O}=\mathcal{O}$，可以看到运用方程（7.2），$-P$对应直线穿过$P$和$\mathcal{O}$的第三个点，或者相当于通过$P$的垂线。可以看到在图7.2 中，这意味着$-P$只是简单的$P$关于$x$轴的对称点；也就是说，如果$P=(x,y)$，则$-P=(x,-y)$。

加法。对于E上任意两个点$P_1, P_2 \neq \mathcal{O}$，可以通过画直线经过$P_1, P_2$的方式计算$P_1+P_2$的和（如果$P_1=P_2$，则画的直线在$P_1$点与$E$相切），找到这条线与$E$的第三个交点$P_3$；如果直线为垂线，则第三个交叉点可能为$P_3=\mathcal{O}$。方程（7.2）意味着$P_1+P_2+P_3=\mathcal{O}$，或者$P_1+P_2=-P_3$。如果$P_3=\mathcal{O}$，则$P_1+P_2=-\mathcal{O}=\mathcal{O}$。否则，如果通过$P_1$和$P_2$的交线的第三个交点$P_3=(x,y)\neq\mathcal{O}$，则

$$P_1 + P_2 = -P_3 = (x, -y)$$

P_1+P_2可以用画图法确定：找过P_1和P_2的直线和与曲线E的第三个交点，然后找到此点关于x轴的对称点。

推导具体的加法法则并不困难，但很繁琐。令$P_1=(x_1,y_1)$，$P_2=(x_2,y_2)$为$E(\mathbb{Z}_p)$上的两个点，其中$P_1, P_2 \neq \mathcal{O}$且$E$为方程（7.1）的形式。为了简化，假设$x_1 \neq x_2$（当$x_1=x_2$情况是容易推导的，但冗长乏味）。通过这些点的直线的斜率为

$$m = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$$

⑤ 该方法是非正式的，为了保持定义一致，这里不作论证。

假设$x_1 \neq x_2$，意味着$x_1 - x_2 \neq 0 \bmod p$且$(x_2 - x_1)$关于模p的逆元存在。直线经过P_1和P_2，有

$$y = m \cdot (x - x_1) + y_1 \bmod p$$

为了找到直线与E的第三个交点，把上面的方程带入E中可得

$$\left(m \cdot (x - x_1) + y_1\right)^2 = x^3 + Ax + B \bmod p$$

满足方程的x的值为x_1, x_2和$[m^2 - x_1 - x_2 \bmod p]$。头两个解对应于原始点$P_1$和$P_2$，然而第三个解对应于第三个交点$P_3$的$x$坐标。第三个$x$对应的$y$值为：$y = [m \cdot (x - x_1) + y_1 \bmod p]$，也就是说，$P_3 = (x_3, y_3)$有

$$x_3 = [m^2 - x_1 - x_2 \bmod p]，且 y_3 = [m \cdot (x_3 - x_1) + y_1 \bmod p]$$

为获得理想的答案$P_1 + P_2$，剩下的只是对P_3取反（即取P_3对x轴的对称点）有：

命题 7.64 设$p \geqslant 5$为素数，且$P_1 = (x_1, y_1), P_2 = (x_2, y_2)$为椭圆曲线$y^2 = x^3 + Ax + B \bmod p$上的点，且$P_1, P_2 \neq \mathcal{O}$，$x_1 \neq x_2$。则$P_1 + P_2 = (x_3, y_3)$满足

$$x_3 = [m^2 - x_1 - x_2 \bmod p]和 y_3 = [m \cdot (x_1 - x_3) - y_1 \bmod p]$$

其中$m = \left[\frac{y_2 - y_1}{x_2 - x_1} \bmod p\right]$

为了保证完整性，下面描述前面命题中没有包括的点的加法规则。

命题 7.65 设$p \geqslant 5$为素数，并且$P_1 = (x_1, y_1), P_2 = (x_2, y_2)$为椭圆曲线$y^2 = x^3 + Ax + B \bmod p$上的点，且$P_1, P_2 \neq \mathcal{O}$。

(1) 如果$x_1 = x_2$，但是$y_1 \neq y_2$，则$P_1 = -P_2$且$P_1 + P_2 = \mathcal{O}$。

(2) 如果$P_1 = P_2$且$y_1 = 0$，则$P_1 + P_2 = 2P_1 = \mathcal{O}$。

(3) 如果$P_1 = P_2$且$y_1 \neq 0$，则$P_1 + P_2 = (x_3, y_3)$满足

$$x_3 = [m^2 - 2x_2 \bmod p]和 y_3 = [m \cdot (x_1 - x_3) - y_1 \bmod p]$$

其中$m = \left[\frac{3x_1^2 + A}{2y_1} \bmod p\right]$。

令人惊叹的是，可以证明$E(\mathbb{Z}_p)$点的集合与前面定义的加法运算规则形成了一个阿贝尔群。事实上，已经得到几乎所有成为阿贝尔群所需要的属性：因为任何直线和E相交有三个交点，加法下的封闭性可以据此得到（这里没有证明）；$\mathcal{O}$为单位元；$E(\mathbb{Z}_p)$中的每个点在$E(\mathbb{Z}_p)$中有一个逆元；加法交换律可以由方程（7.2）得来。唯一困难的是证明结合律，不相信的读者可以通过冗长的计算来检查其正确性。依据代数几何知识可以得到一个不需要通过计算的启发式证明。

典型的密码学应用中，通过素数阶群$E(\mathbb{Z}_p)$（或者其子群）来选择椭圆曲线的参数，得到一个循环群。至于高效的参数选择方法超出了本书的范围。

7.4 数论假设的密码学应用

前面已经花了相当多的笔墨讨论数论和群论，并且介绍了被广泛认为是计算困难的假

设。这些假设的应用将占据本书后面的内容，但这里只简要给出几个例子。

7.4.1 单向函数和置换

单向函数是最小的密码学原语，对于第 3 章和第 4 章中所有对称加密构造方法，它是必要的和充分的。单向函数在密码学中扮演的角色在第 6 章中给出了完整的介绍；这里只给出单向函数的定义，并且论证它们的存在是基于本章中所介绍的数论问题困难性的假设。

非正式地说，函数f是单向函数，计算很容易，但求逆很困难。下面的实验和定义是关于此点的正式描述（也是对定义 6.1 的重新陈述）：

求逆实验$\mathsf{Invert}_{\mathcal{A},f}(n)$：

(1) 选择输入$x \leftarrow \{0,1\}^n$，计算$y := f(x)$。

(2) $\mathcal{A}$给定1^n和输入y，输出x'。

(3) 当且仅当$f(x') = y$时，实验的输出定义为 1。

定义 7.66 如果满足下面两个条件，则函数$f : \{0,1\}^* \to \{0,1\}^*$是“单向函数”

(1)（容易计算：）存在一个多项式时间算法，输入x，输出$f(x)$。

(2)（求逆困难：）对任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Invert}_{\mathcal{A},f}(n) = 1] \leqslant \mathsf{negl}(n)$$

现在正式证明因子分解假设意味着单向函数的存在性。设Gen为一个多项式时间算法，输入1^n，输出(N,p,q)，其中$N = pq$且p和q为两个n比特素数（例外的情况为以n表示的可忽略概率）。（此处用Gen而不用GenModulus单纯是因为符号表示方便。）因为Gen运行时间为多项式时间，所以存在一个多项式p满足：算法对输入1^n使用的随机比特的位数最大为$p(n)$。为了简洁，假设算法通常对输入1^n恰好使用$p(n)$比特位。并且，$p(n)$是严格单调递增的。在算法 7.67 中，定义函数f_{Gen}，其输入作为Gen的一个随机带。因为单向函数求逆的困难性要求输入是随机的，这里将输入解释为随机带，则生成函数将拥有需要的属性。因此，尽管f_{Gen}运行Gen作为子程序，但是实际上它是确定算法。（基于输入x的长度来计算n，是一个技术性处理，用来确保Gen接受到一个有合适长度的随机带。）

算法 7.67
计算f_{Gen}的算法
输入： 字符串x
输出： 字符串N

compute n，满足$p(n) \leqslant |x| < p(n+1)$
compute $(N,p,q) := \mathsf{Gen}(1^n; x)$
/*也就是说，以x作为随机带运行$\mathsf{Gen}(1^n)$ */

return N

如果因子分解问题与Gen相关是困难的，则直观上，f_{Gen}是一个单向函数。当然，f_{Gen}要

容易计算。至于函数求逆的困难性，对任意n'，下面的分布是相同的：

(1) 模N由$f_{\mathsf{Gen}}(x)$输出，其中$x \in \{0,1\}^{n'}$是根据均匀分布选择的。

(2)模N由随机过程输出，在随机过程中，运行$\mathsf{Gen}(1^n)$得到N，这里n满足

$$p(n) \leqslant n' < p(n+1)$$

因为根据第二种分布选择的模N很难分解，故根据第一种分布选择的模N也很难分解。此外，给定任意x，满足$f_{\mathsf{Gen}}(x) = N$，很容易恢复N的一个因子（通过运行$\mathsf{Gen}(1^n, x)$而得到(N, p, q)。并且输出因数p和q)。因此，求f_{Gen}的逆并且找到满足条件的x与分解因子问题一样困难。因此可以得到下面的定理（正式的证明可以相当简单地从描述中得到）：

定理 7.68 如果分解问题与Gen相关是困难的，则f_{Gen}是一个单向函数。

由此得出一个推论：RSA问题的困难性意味着存在一个单向函数（至少当GenRSA通过算法 7.47 来构造时，RSA问题的困难性意味着因子分解问题是困难的)。

***单向置换。**已经提到如果RSA问题是困难的，则存在单向函数。然而，RSA问题实际上给出更强的结论：单向置换簇。(从此处开始到本节结束是 10.7 节需要的，并且对已经学习了第 6 章的读者有意义；否则，可以略过。）首先，重新描述定义 6.3 和 6.4：

定义 7.69 一个概率多项式时间算法的三元组$\Pi = (\mathsf{Gen}, \mathsf{Samp}, f)$，如果满足下列条件，则其为一个函数簇：

(1) “参数产生算法Gen”，输入为1^n，输出参数I，满足$|I| \geqslant n$。每一个由Gen输出的I值定义了集合$\mathcal{D}_I$和$\mathcal{R}_I$，分别构成函数f_I的定义域和值域。

(2) “取样算法Samp”，输入为I，输出一个均匀分布的集合$\mathcal{D}_I$中的元素（例外情况的概率是以$|I|$表示的可忽略概率)。

(3) 确定的“评估算法f”，输入为I，$x \in \mathcal{D}_I$，输出一个元素$y \in \mathcal{R}_I$。记为$y = f_I(x)$。

Π是一个置换簇：如果对$\mathsf{Gen}(1^n)$输出的I的每个值，满足$\mathcal{D}_I = \mathcal{R}_I$且函数$f_I : \mathcal{D}_I \to \mathcal{D}_I$为双射。

由最后一个条件，当Π是置换簇时，均匀随机选择$x \leftarrow \mathcal{D}_I$并且令$y := f_I(x)$得到$y$值，该值为$\mathcal{D}_I$上的均匀分布。

给定置换簇Π，对于任意算法$\mathcal{A}$和参数n，考虑下面的实验：

求逆实验$\mathsf{Invert}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$得到I，且运行$\mathsf{Samp}(I)$得到一个随机$x \leftarrow \mathcal{D}_I$。最后，计算$y := f_I(x)$。

(2) $\mathcal{A}$被给定I和y为输入，输出x'。

(3) 如果$f_I(x') = y$，则实验的输出被定义为 1，否则为 0。

定义 7.70 一个函数/置换簇$\Pi = (\mathsf{Gen}, \mathsf{Samp}, f)$是单向函数，如果对于所有概率多项式时间算法$\mathcal{A}$存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Invert}_{\mathcal{A},\Pi}(n) = 1] \leqslant \mathsf{negl}(n)$$

给定如 7.2.4 节中的GenRSA，构造方法 7.71 定义了置换簇。于是，如果RSA问题对

GenRSA而言是困难的，则此函数簇实际上为单向函数。可以类似得到，对于$\mathbb{Z}_p^*$中（p为素数）的离散对数问题的困难性，意味着存在一个单向置换簇（参考练习 7.20）。

构造方法 7.71

设GenRSA为 7.2.4 节中介绍的算法。以下面的方式定义一个置换簇：

- Gen：输入1^n，运行GenRSA(1^n)得到(N,e,d)，且输出$I=\langle N,e\rangle$。集合$\mathcal{D}_I=\mathbb{Z}_N^*$。
- Samp：输入$I=\langle N,e\rangle$，从$\mathbb{Z}_N^*$中选择一个随机元素。
- f：输入$I=\langle N,e\rangle$和$x\in\mathbb{Z}_N^*$，输出$[x^e \bmod N]$。

一个单向置换簇（假设RSA问题与GenRSA相关是困难的。）

7.4.2 构造抗碰撞的散列函数

抗碰撞的散列函数在 4.6 节中进行了介绍。就像在本节中介绍的，虽然存在一些启发式的抗碰撞散列函数构造方法，而且这些方法广泛应用于实践中，但是并没有发现任何这种散列函数构造方法可以在一些基础假设情况下证明是安全的。（因此，第 6 章中没有介绍这样的构造方法。事实上，有证据显示用任意单向函数或单向置换构造抗碰撞散列函数是不可能的。）

现在介绍一个基于素数阶群离散对数假设构造的抗碰撞散列函数。基于RSA问题的第二个构造方法在练习 7.21 中描述。虽然这些构造方法的效率比实践中使用的散列函数要低，但是它们是重要的，因为它们例证了基于标准的和经过充分研究的密码学假设获得抗碰撞的可行性。

就像在 7.3.2 节中介绍的，设$\mathcal{G}$为多项式时间算法：输入1^n，输出一个循环群$\mathbb{G}$（的描述），其阶为q（$||q||=n$），和一个生成元g。通常，假设群$\mathbb{G}$上的运算可以高效地计算。最后，也需要q为素数，例外的情况为可忽略的概率。（通常倾向于使用素数阶群，就像 7.3.2 节中介绍的。）基于$\mathcal{G}$的定长散列函数在构造方法 7.72 中给出。

构造方法 7.72

设$\mathcal{G}$为如前所述的算法。定义一个定长散列函数如下：

- Gen：输入为1^n，运行$\mathcal{G}(1^n)$得到$(\mathbb{G},q,g)$，然后选择$h\leftarrow\mathbb{G}$。输出$s:=\langle\mathbb{G},q,g,h\rangle$作为密钥。
- H：给定一个密钥$s=\langle\mathbb{G},q,g,h\rangle$且输入$(x_1,x_2)\in\mathbb{Z}_q\times\mathbb{Z}_q$，输出$H^s(x_1,x_2):=g^{x_1}h^{x_2}$。

一个定长散列函数

注意到Gen和H可以在多项式时间内计算。在继续分析这个构造方法前，给出一些技术备注：

(1) 对于一个给定$s=\langle\mathbb{G},q,g,h\rangle$，其中$n=||q||$，函数$H^s$以$\mathbb{Z}_q\times\mathbb{Z}_q$的元素作为输入。然而$H^s$可以视为将长度为 $2(n-1)$的比特串作为输入。原因是输入$x\in\{0,1\}^{2(n-1)}$可拆分成两个串x_1,x_2，且每个长度为$n-1$，此时可将x_1,x_2视为$\mathbb{Z}_q$的元素。

(2) 类似地，H^s的输出可视为$\mathbb{G}$中的某个元素，如果固定$\mathbb{G}$的某些表示方式，也可以将其

视为一个比特串。为了满足定义 4.12 的要求（即输出的长度固定为n的函数），可以填充输出为所需形式。

(3) 根据如上所述，此构造方法只对某些群$\mathbb{G}$能够压缩输入（具体而言，当$\mathbb{G}$中元素可以用少于$2n-2$位表示的时候）。就像练习 7.19 中说明的，当使用 7.33 节中讨论的$\mathbb{Z}_p^*$子群时，可达到压缩目的。通用的构造方法 7.72 可从任何离散对数问题是困难的算法$\mathcal{G}$中获得压缩，而不管表示群中元素所需要的比特数（参见练习 7.22）。

定理 7.73 如果离散对数问题与$\mathcal{G}$相关是困难的，则构造方法 7.72 是一个定长抗碰撞散列函数（满足上面介绍的关于压缩的讨论）。

证明 设$\Pi=(\mathsf{Gen},H)$如构造方法 7.72 中所述，设$\mathcal{A}$是一个概率多项式时间算法，满足

$$\varepsilon(n)\stackrel{\mathsf{def}}{=}\Pr[\mathsf{Hash}-\mathsf{coll}_{\mathcal{A},\Pi}(n)=1]$$

（参考定义 4.12）。下面介绍$\mathcal{A}$如何被$\mathcal{A}'$使用来解决离散对数问题，且成功的概率为$\varepsilon(n)$：

算法$\mathcal{A}'$：

算法给定$\mathcal{G},q,g,h$作为输入。

(1) 设$s:=\langle\mathbb{G},q,g,h\rangle$。运行$\mathcal{A}(s)$得到输出$x,x'$。

(2) 如果$x\neq x'$且$H^s(x)=H^s(x')$，则：

① 如果$h=1$，则返回 0。

② 否则（$h\neq 1$），把x拆分为(x_1,x_2)，把x'拆分为(x_1',x_2')。返回$[(x_1-x_1')\cdot(x_2-x_2')^{-1} \bmod q]$。

显然，$\mathcal{A}'$运行时间为多项式时间。此外，把s输入到$\mathcal{A}$，当$\mathcal{A}$作为一个子例程被$\mathcal{A}'$运行时，其分布和一个有相同安全参数n的实验$\mathsf{Hash}-\mathsf{coll}_{\mathcal{A},\Pi}$是一样的。（$\mathcal{A}'$的输入由$\mathcal{G}(1^n)$生成，得到$\mathbb{G},q,g$，然后随机均匀地选择$h\in\mathbb{G}$。这其实就是$\mathsf{Gen}(1^n)$产生$s$的方式。）所以，存在碰撞（$x\neq x'$且$H^s(x)=H^s(x')$）的概率为$\varepsilon(n)$。

下面证明每当碰撞发生时，$\mathcal{A}'$返回正确答案$\log_g h$。如果$h=1$，这显然是正确的（因为此时$\log_g h=0$）。否则，存在一个碰撞意味着

$$\begin{aligned}H^s(x_1,x_2)=H^s(x_1',x_2')&\Rightarrow g^{x_1}h^{x_2}=g^{x_1'}h^{x_2'}\\&\Rightarrow g^{x_1-x_1'}=h^{x_2'-x_2}\end{aligned}\tag{7.3}$$

设$\Delta\stackrel{\mathsf{def}}{=}x_2'-x_2$。注意到$\Delta\neq 0 \bmod q$，因为这意味着$[(x_1-x_1') \bmod q]=0$，有$x=(x_1,x_2)=(x_1',x_2')=x'$与假设$x\neq x'$矛盾。因为$q$是素数且$\Delta\neq 0 \bmod q$，存在逆元$[\Delta^{-1} \bmod q]$。把式（7.3）两边取这个幂次，有

$$g^{(x_1-x_1')\cdot\Delta^{-1}}=\left(h^{x_2'-x_2}\right)^{[\Delta^{-1} \bmod q]}=h^{[\Delta\cdot\Delta^{-1} \bmod q]}=h^1=h$$

这样

$$\log_g h=[(x_1-x_1')\ \Delta^{-1} \bmod q]=[(x_1-x_1')\cdot(x_2'-x_2)^{-1} \bmod q]$$

输出由$\mathcal{A}'$返回。

于是$\mathcal{A}'$正确解决了离散对数问题，其概率为$\varepsilon(n)$。因为由假设离散对数问题与$\mathcal{G}$相关是困难的，得出$\varepsilon(n)$为可忽略概率。 ■

利用练习 7.22 并且结合Merkle-Damgard变换（参见 4.6.4 节），可以得到下面的定理：

定理 7.74 如果存在一个多项式时间算法$\mathcal{G}$，并且与之相关的离散对数问题是困难的，则存在一个抗碰撞散列函数。

参考文献和扩展阅读材料介绍

Childs[32]的书极好地涵盖了本章讨论的群论，且更加深入但阐述的深度类似。Shoup[131]给出了更加深入的介绍，但对于大部分知识而言还是容易理解的。对于抽象代数和群论这些远远超出本书介绍范围的内容在 Fraleign[57]和 Herstein[76]的书中给出了相对平实的介绍；如果愿意的话，有兴趣的读者容易找到一些更高级的代数课本。

第一个高效的素性测试由 Solovay 和 Strassen[135]提出。Miller-Rabin 测试归功于 Miller[105]和 Rabin[120]。一个确定的素性测试最近由 Agrawal 等[2]发现。参见 Dietzfelbinger[46]对素性测试作了可理解的全面地介绍。

RSA 置换由 Rivest、Shamir 和 Adleman[122]提出。Boneh[25]对 1999 年前的这个问题状态做出了总结。离散对数问题和 Diffe-Hellman 问题第一次被 Diffie 和 Hellman[47]考虑到，至少是隐含地被考虑。关于这些问题及其应用的综述由 Odlyzko[112]和 Boneh[24]给出。

椭圆曲线虽然有很多参考资料，但几乎没有一本不需要以高等数学为知识背景。由 Silverman 和 Tate[132]写的一本书或许是个例外，不像大多数书中对这个主题的介绍，这本书较少介绍有限域上的椭圆曲线，而这些知识大多和密码学有关。由 Washington[145]写的课本尽管更加高深，但处理了大量的（虽然不仅仅是）有限域的情况。与密码学椭圆曲线相关的实现问题在 Hankerson 等的著作[72]中有所介绍。

以离散对数问题为基础构造抗碰撞散列函数归功于文献[31]，且一个更早的基于因子分解的困难性的抗碰撞散列函数在文献[70]给出（参见练习 7.21）。

练　习

7.1 设$\mathbb{G}$为阿贝尔群。证明在$\mathbb{G}$中存在一个唯一的单位元，且对于每个元素$g \in \mathbb{G}$只有唯一一个逆元。

7.2 证明当$\mathbb{G}$为无穷的时，命题 7.36 并不必满足。

提示：考虑集合$\{1\} \cup \{2,4,6,8,\cdots\} \subset \mathbb{R}$。

7.3 设$\mathbb{G}$为有限群，$g \in \mathbb{G}$。证明$\langle g \rangle$是$\mathbb{G}$的一个子群。当$\mathbb{G}$为无穷的，集合$\{g^0, g^1, \cdots\}$需要是$\mathbb{G}$的子群吗？

7.4 这个问题涉及到欧拉函数。

（1）设p为素数，$e \geqslant 1$为整数。证明$\phi(p^e) = p^{e-1}(p-1)$。

（2）设p、q互素。证明$\phi(pq) = (p-1)(q-1)$。（可以不用中国剩余定理。）

（3）证明定理 7.19。

7.5 计算3^{1000}的最后两个（十进制）数字（手工计算）。

提示：答案为$[3^{1000} \bmod 100]$。

7.6 计算$[101^{4\,800\,000\,023} \bmod 35]$（手工计算）。

7.7 证明如果$\mathbb{G}$，$\mathbb{H}$为群，则$\mathbb{G} \times \mathbb{H}$也是群。

7.8 设p, N是满足$p \mid N$的整数。证明对于任何整数x，有

$$[[X \bmod N] \bmod p] = [X \bmod p]$$

证明，相反地，$[[X \bmod p] \bmod N]$并不需要等于$[X \bmod N]$。

7.9 完成中国剩余定理细节的证明，并且证明$\mathbb{Z}_N^*$是$\mathbb{Z}_p^* \times \mathbb{Z}_q^*$的同构。

7.10 推论 7.21 说明如果$N = pq$和$ed = 1 \bmod \phi(N)$。则对于所有$x \in \mathbb{Z}_N^*$，可得$(x^e)^d = x \bmod N$。证明对于所有$x \in \mathbb{Z}_N$都满足这个等式。

提示：利用中国剩余定理。

7.11 这个练习开发一个高效的算法来测试一个整数是否为完全幂。

（1）证明：如果$N = \hat{N}^e$对于一些整数$\hat{N}, e > 1$。则$e \leqslant ||N|| + 1$。

（2）给定N和e满足$2 \leqslant e \leqslant ||N|| + 1$，给出如何在poly($||N||$)时间内确定是否存在一个整数$\hat{N}$满足$\hat{N}^e = N$。

提示：利用二元搜索。

（3）给定N，给出如何在poly($||N||$)时间内测试N是否为完全幂。

7.12 给定N和$a \in \mathbb{Z}_N^*$，证明如何在多项式时间内测试a是否为N为合数的强证据。

7.13 设$N = pq$是两个不同素数的乘积。证明如果已知$\phi(N)$和N，则可以在多项式时间内计算出p和q。

提示：推导一个二次方程（整数域上），未知数为p。

7.14 设$N = pq$是两个不同素数的乘积。证明如果已知N和整数d，满足$3 \cdot d = 1 \bmod \phi(N)$，$d \in \{1, \cdots, \phi(N)\}$则能在多项式时间内计算$p$、$q$。

提示：得到一个$\phi(N)$概率的小段列表，然后利用前面的练习。

7.15 正式地证明与$\mathcal{G}$相关的CDH问题的困难性意味着与$\mathcal{G}$相关的离散对数问题的困难性。

7.16 正式地证明与$\mathcal{G}$相关的DDH问题的困难性意味着与$\mathcal{G}$相关的CDH问题的困难性。

7.17 证明命题 7.65 的第三个陈述。

7.18 确定下面的问题是否为困难的。设p为素数，选定$x \in \mathbb{Z}_{p-1}^*$，给出p，x和$y := [g^x \bmod p]$（其中g为 1 到$p-1$之间的一个随机值），计算g的值。也就是说，计算$y^{1/x} \bmod p$。如果你认为这个问题是困难的，说明可以规约到用本章中介绍的某个假设。如果你认为这个问题是简单的，提出一个算法，证实其正确性，并分析其复杂度。

7.19 设$\mathcal{G}$为一个算法，输入1^n，输出p, q, g，其中$p = 2q + 1$是一个强素数，g是模p二次剩余子群的生成元。（参见 7.3.3 节。）说明如何得到一个构造方法 7.72 中介绍的压缩，其中p足够大。

7.20 设$\mathcal{G}_1$如 7.3.3 节中介绍。证明与$\mathcal{G}_1$相关的离散对数问题的困难性，意味着存在单向置换簇。

提示：对$\mathbb{Z}_p^*$的元素定义一个置换。

7.21 设GenRSA为 7.2.4 节中介绍的情形。证明如果与GenRSA相关的RSA问题是困难的，则下面的构造方法 7.75 是一个定长抗碰撞散列函数。

构造 7.75

定义(Gen, H)如下：

- Gen：输入1^n，运行GenRSA(1^n)得到N, e, d，且选择$y \leftarrow \mathbb{Z}_N^*$。密钥为$s := \langle N, e, y \rangle$。
- H：如果$s = \langle N, e, y \rangle$，则$H^s$映射$\{0,1\}^{3n}$的输入到$\mathbb{Z}_N^*$的输出。设$f_0^s(x) \stackrel{\text{def}}{=} [x^e \bmod N]$和$f_1^s(s) \stackrel{\text{def}}{=} [y \cdot x^e \bmod N]$。对一个$3n$比特长的字符串$x = x_1 \cdots x_{3n}$，定义

$$H^s(x) \stackrel{\text{def}}{=} f_{x_1}^s\left(f_{x_2}^s\left(\cdots\left(1\right)\cdots\right)\right)。$$

提示：证明y的e次根可以从任何碰撞计算出来。

7.22 考虑下面显示的构造 7.72 的一般化形式。

构造 7.76

定义一个定长散列函数(Gen, H)如下：

- Gen：输入1^n，运行$\mathcal{G}(1^n)$得到$(\mathbb{G}, q, h_1)$，然后选择$h_2, \cdots, h_t \leftarrow \mathbb{G}$。输出$s := \langle \mathbb{G}, q, (h_1, \cdots, h_t) \rangle$作为密钥。
- H：给定一个密钥$s := \langle \mathbb{G}, q, (h_1, \cdots, h_t) \rangle$且输入$(x_1, \cdots, x_t)$满足$x_i \in \mathbb{Z}_q$，输出$H^s(x_1, \cdots, x_i) := \Pi_i h_i^{x_i}$。

（1）证明如果离散对数问题关于$\mathcal{G}$是困难的，且q为素数，则对于任意$t = \text{poly(n)}$，这个构造方法是一个定长抗碰撞散列函数。

（2）不考虑表示$\mathbb{G}$中元素需要的位数，讨论如何使用该构造方法得到压缩（只要在n的多项式时间内）。

第 8 章　因子分解和离散对数算法

正如第 7 章已经讨论过的，因子分解或计算某些群中的离散对数，目前还没有已知的多项式时间算法，但是这并不意味着蛮力搜索是攻击这些问题的最佳可用途径。本章综述对这些问题的有效算法。这些算法本身是有趣的，并可作为已学过的数论知识的良好应用。此外，理解这些算法的效率对于在实际应用中选择密码学参数是至关重要的。如果一个基于因子分解的密码学方案被认为可以经受住至少 15 年的攻击，那么该方案中的模N要足够长，以至于即使是最高效的因子分解算法，也至少要用 15 年时间才能成功分解N的因子。

关于解决其他难题的算法。本章重点是因子分解和计算离散对数的算法，而不是解决 RSA 问题或判定 Diffie-Hellman 问题的算法。这可能有些误导，因为在构造密码学方案时后者比前者使用得更加频繁，如前面章节所提到的，后者比前者潜在地更容易解决。（也就是说，解决 RSA 问题潜在地比解决因子分解问题容易，解决判定 Diffie-Hellman 问题可能比计算离散对数问题容易。）如此选择关注重点是基于以下事实：当前已知的解决 RSA 问题的最好算法是先分解模的因子，并且（第 7.3.3 节和 7.3.4 节讨论过在适当的群中）已知的解决判定 Diffie-Hellman 问题最好的算法是计算离散对数。

8.1　因子分解算法

本节假设$N=pq$是两个不同素数的乘积。最感兴趣的情况是，p和q有相同的（已知的）长度n，此时有$n=\Theta(\log N)$。还有其他为不同形式的N专门设计的算法（例如，当$N=p^rq$，其中p,q为素数，整数$r>1$，或当p和q的长度相差很大），但是在此不作讨论。

这里将会经常用到中国剩余定理以及 7.1.5 节中给出的表示符号。中国剩余定理规定：

$$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

对$x \in Z_N^*$满足同构$f(x) \stackrel{\text{def}}{=} ([x \mod p], [x \mod q])$。$f$是同构意味着$f$建立了从$x \in Z_N^*$到$(x_p, x_q) \in Z_p^* \times \mathbb{Z}_q^*$间的一个双射，记为$x \leftrightarrow (x_p, x_q)$，其中$x_p = [x \mod p]$且$x_q = [x \mod q]$。回想 7.2 节中的试除法，一种简单的蛮力搜索因子分解方法，在$\mathcal{O}(\sqrt{N} \cdot \text{polylog}(N))$时间内找到一个给定的数$N$的因子的概率为 1。因此一种更复杂的分解方法只有在运行时间复杂度比上一方法更优下才会引起人们的兴趣。下面将描述改进运行时间的三种不同分解算法：

(1) Pollard 的$p-1$方法，当$p-1$有“小”素数因子时它是一种有效的方法。

(2) Pollard 的 rho 方法，这种方法适用于任意的N（因此，这种方法还被称为“通用”因子分解算法。）对于本节开始部分讨论的那种类型的整数N来说，它的时间复杂度是$\mathcal{O}(N^{1/4} \cdot \text{polylog}(N))$。因为$N=2^{\Theta(n)}$，所以这仍是关于$n$（$N$的长度）的指数函数。

(3) 二次筛算法，这是一种通用因子分解算法，运行时间的复杂度为（关于N的长度）亚指数。[①]只给出这种算法如何工作的简要概述，该算法的细节比较复杂，超出本书的讨论

① 如果$f(n)=2^{\Omega(n)}$，那么f是关于n的指数。如果$f(n)=2^{o(n)}$，则f是关于n的亚指数。一个关于n的多项式有形式$f(n)=2^{\mathcal{O}(\log n)}=n^{\mathcal{O}(1)}$。

范围。

当前，已知最好的通用因子分解算法（考虑其渐进运行时间复杂度）是“通用数域筛法”。从启发式分析的角度来说，当分解长度为$\mathcal{O}(n)$的数N时，该算法的运行时间复杂度平均为$2^{\mathcal{O}(n^{1/3}\cdot(\log n)^{2/3})}$。②

8.1.1 Pollard 的$p-1$方法

整数$N=pq$且$p-1$只有“小”因子，在这种情况下，Pollard 设计了一个算法能有效地进行因子分解（算法的描述要用到附录 B.3.1 中的结果）。这种方法的关键在于以下的观察：

能够找到一个元素$y\in Z_N^*$且$y\leftrightarrow(1,y_q)$，且$y_q\neq 1$。即

$$y=1 \bmod p，但 y\neq 1 \bmod q \tag{8.1}$$

或者等价地写为

$$y-1=0 \bmod p，但 y-1\neq 0 \bmod q$$

以上所述表明$p|(y-1)$，但$q\nmid(y-1)$，由此可推出$\gcd(y-1,N)=p$，因此，一个简单的gcd计算（正如附录 B.1.2 中描述的可以高效地计算）生成了一个N的非平凡因子。

从而N的因子分解问题就规约为寻找一个具有所陈述属性的y值。现在来描述怎样寻找这样一个满足条件的y。假设一个整数B满足

$$(p-1)|B，但(q-1)\nmid B \tag{8.2}$$

（在后文将叙述这样的B值是如何确定的。）对整数γ，$B=\gamma\cdot(p-1)$。这种算法（见下面的伪代码）选择一个随机元素$x\leftarrow Z_N^*$且设定$y:=[x^B \bmod N]$；注意y值可以用附录 B.2.3 中的高效求幂算法计算。

算法 8.1
Pollard 的$p-1$因式分解方法
输入：整数N
输出：整数N的一个非平凡因子
$x\leftarrow Z_N^*$
$y:=[x^B \bmod N]$
$p:=\gcd(y-1,N)$
if $p\notin\{1,N\}$ **return** p

现在说明：y满足式（8.1），从而用以上算法就能以较高的概率找到N的一个非平凡因子。有

$$\begin{aligned} y=[x^B \bmod N]\leftrightarrow(x_p,x_q)^B&=(x_p^B \bmod p,x_q^B \bmod q)\\ &=((x_p^{p-1})^\gamma \bmod p,x_q^B \bmod q)=(1,x_q^B \bmod q)\end{aligned}$$

最后一个等式利用了定理 7.14 以及Z_p^*的阶为$p-1$的事实，故有$y\leftrightarrow(1,x_q^B \bmod q)$。从而若能证明$x_q^B\neq 1 \bmod q$，则$y$满足式（8.1）。下面证明这一点。

因为q是一个素数，Z_q^*是$q-1$阶循环群，该群恰好包含$\phi(q-1)$个生成元，且每个生成元的阶都是$q-1$（见定理 B.18）。可以认为如果x_q是Z_q^*的一个生成元，且$(q-1)\nmid B$，则

② 严格地分析这个算法的运行时间仍是一个公开问题。

$x_q^B \neq 1 \mod q$。为了说明这一点，使用“带余除法”（命题 7.1），记$B = \alpha \cdot (q-1) + \beta$（$1 \leqslant \beta < (q-1)$），利用定理 7.14，有

$$x_q^B = x_q^{\alpha \cdot (q-1)+\beta} = (x_q^{q-1})^\alpha x_q^\beta = x_q^\beta \mod q$$

但是$x_q^\beta \neq 1 \mod q$，因为x_q的阶比β大。剩下就是分析x_q是生成元的可能性。

假定$(q-1) \nmid B$，如果x都从Z_N^*中均匀随机选择，则$x_q \stackrel{\text{def}}{=} [x \mod q]$均匀分布在$Z_q^*$中（这是由中国剩余定理给出的$Z_N^*$和$Z_p^* \times Z_q^*$之间的双射得到的结果。）因为$Z_q^*$有$\phi(q-1)$个生成元，所以选择的$x$满足$x_q^B \neq 1 \mod q$的概率至少为$\dfrac{\phi(q-1)}{q-1}$，使用定理 B.16，此概率是$\Omega(1/\log q)$。

上述讨论表明在满足式（8.2）的B已知的情况下，Pollard 的$p-1$算法成功地找到N的一个非平凡因子的概率为$\Omega(1/\log q) = \Omega(1/n)$。或者，运行算法$\mathcal{O}(\log^2 q) = \text{poly}(n)$次循环，找到一个$N$的非平凡因子，不成功的概率是可忽略的。剩下的工作就是选择B的值。

一种方法是选择B满足

$$B = \prod_{i=1}^{k} p_i^{\lfloor n/\log p_i \rfloor}$$

式中p_i代表第i个素数（即$p_1 = 2, p_2 = 3, p_3 = 5, \cdots$）；$k$代表一个范围，它影响时间复杂度和算法成功的概率。注意：$p_i^{\lfloor n/\log p_i \rfloor}$是能整除$p-1$的最大$p_i$幂，$p-1$的最大长度为$n$。因此只要$p-1$可以被写成形式$\prod_{i=1}^{k} p_i^{e_i} (e_i \geqslant 0)$（即只要$p-1$没有比$p_k$大的素数因子），就可以得出$(p-1) | B$。相比之下，只要$q-1$一个素数因子比$p_k$大，那么$(q-1) \nmid B$。

选择一个较大的数k，以增大B的值会增加算法的运行时间（进行模幂运算直到B次幂）k的值越大，$(p-1)|B$成立的可能性就越大，但同时$(q-1) \nmid B$成立的可能性就越小。当然，也可以选择不同的k值多次运行算法，或考虑一些其他选择B的方法。

当$p-1$与$q-1$只有大素数因子时，Pollard 的$p-1$方法就会有麻烦。（更精确地说，除了在B大到算法不再是高效的情况下，该算法都能继续工作。）正是这个原因，在产生用于密码学应用的模$N = pq$时，有时候选取强素数p和q（前面曾经提到，如果$(p-1)/2$是素数则p是强素数）。用这种方法选择p和q，显然其效率比简单地随机选择素数p和q的效率要低。因为更好的因子分解算法总是可用的，正如下文中会看到的，所以当前的共识是：产生强素数p和q的计算代价并没有被任何安全收益所弥补。但是，由于与Z_N^*群结构相关的技术原因，某些密码学体制（未出现在本书中）需要p和q必须是强素数。

8.1.2 Pollard 的 Rho 方法

与上节所讲述的算法不同，Pollard 的 rho 方法可以找到任意整数N的非平凡因子，不用对p和q值有任何假设；也就是说，这是一种“通用”因子分解算法。确定该算法的运行时间和成功概率的严格的界限仍是一个公开问题。然而利用启发式分析方法，这种算法可以在$\mathcal{O}(N^{1/4} \cdot \text{ploylog}(N)) \approx 2^{\|N\|/4}$步之内以恒定的概率将$N$因子分解，比试除法效率高。

Rho 方法的思想是寻找两个不同的数值$x, x' \in \mathbb{Z}_N^*$，这两个数关于模p同余（即$x = x' \mod p$）；将这样的一对值称为“good”。正如上一节所述，观察到$x - x' = 0 \mod p$但是$x - x' \neq 0 \mod N$，从而$p|(x-x')$但$N \nmid (x-x')$。这意味着$\gcd(x-x', N) = p$，它是N的一个非平凡因子。

怎样找到一个“good”整数对呢？假设均匀的从Z_N^*中分别独立地均匀随机选择数值

$x_1, \cdots, x_k$，这里$k = 2^{n/2} = \mathcal{O}(\sqrt{p})$。根据附录 A.4 中所证明的生日问题的界，有

(1) 存在不同的i, j使得$x_i = x_j$的概率最大为

$$\frac{k^2}{2 \cdot \phi(N)} = \frac{2^n}{2 \cdot \phi(N)} = \mathcal{O}(2^{-n})$$

该数值是用n表示的可忽略量（参见引理 A.9。）

(2) 由中国剩余定理，作为Z_N^*与$Z_p^* \times Z_q^*$之间的双射，$\{[x_m \bmod p]\}_{m=1}^k$独立均匀地分布在$Z_p^*$中。根据引理 A.10，存在$i$和$j$满足$[x_i \bmod p] = [x_j \bmod p]$，即 $x_i = x_j \bmod p$ 的概率约为 1/4。

综上，可得约 1/4 的概率存在i和j，使得x_i, x_j为“good”对；即

$$x_i = x_j \mod p, \text{ 但} x_i \neq x_j \mod N$$

如前所述，这一对值将被用于寻找N的非平凡因子。

可以在$\mathcal{O}(\sqrt{p}) = \mathcal{O}(N^{1/4})$时间内生成$Z_N^*$中的$k = \mathcal{O}(\sqrt{p})$个随机元素，然后测试所有的整数对，来获得“good”整数对，这样会花费$\binom{k}{2} = \mathcal{O}(k^2) = \mathcal{O}(p) = \mathcal{O}(N^{1/2})$时间。（注意因为$p$是未知的，所以不能简单地计算序列$\hat{x}_i \stackrel{\text{def}}{=} [x_i \mod p]$，然后再通过排序$\hat{x}_i$来获得“good”数对。取而代之，对所有的$i, j$必须计算出$\gcd(x_i - x_j, N)$以确定该整数对是否能得出$N$的非平凡因子）。如果不进一步优化，这种方法并不比试除法好。

Pollard 的想法是：通过递归的方法选择 $x_i, \cdots, x_k, \cdots, x_{2k}$，即随机选择$x_1 \in \mathbb{Z}_N^*$，然后使用合适的$F$函数计算出$x_m := F(x_{m-1}) \mod N$。（下文将讨论选择$F$的方法。）正如下面声称将要证明的，为寻找“good”数对，现在只需要针对所有$i \leqslant k$来测试x_i与x_{2i}就足够，而不需要测试每个整数对x_i, x_j $(i, j \leqslant k)$。

声称 8.2 设$x_1, \cdots$是满足$x_m = F(x_{m-1}) \mod N$的一个序列。如果$x_i = x_j \mod p$ $(i < j)$，那么存在$i' < j$使得$x_{i'} = x_{2i'} \mod p$。

证明： 如果$x_i = x_j \mod p$，则序列$[x_i \mod p], [x_{i+1} \mod p], \cdots$以周期$j - i$重复。（即对于所有的$i' \geqslant i$以及整数$\delta \geqslant 0$都有$x_{i'} = x_{i'+\delta(j-i)} \mod p$）。取$i'$是$j - i$的最小倍数且$i'$不小于$i$；也就是说，$i' \stackrel{\text{def}}{=} (j - i) \cdot \lceil i/(j - i) \rceil$。因为序列$i, i + 1, \cdots i + (j - i - 1)$包含$j - i$的倍数，故有$i' < j$。又因为$2i' - i' = i'$是周期的倍数，且$i' \geqslant i$，因此$x_{i'} = x_{2i'} \mod p$。 ■

根据上述声称可知，如果序列 $x_1, \cdots, x_k$ 中存在这样一对“good”数x_i, x_j，那么在序列$x_1, \cdots, x_{2k}$ 中一定存在另一对这样的适当的数$x_{i'}, x_{2i'}$。这样，需要验证的数就从$\binom{k}{2}$降到$k = \mathcal{O}(\sqrt{p}) = \mathcal{O}(N^{1/4})$。下面是整个算法的描述：

算法 8.3
Pollard 的 rho 因子分解算法
输入： 整数N，两个n比特素数的乘积
输出： 整数N的一个非平凡因子

```
x_0 ← Z_N^*
for  i = 1 to 2^{n/2}
     x_i := [F(x_{i-1}) mod N]
     x_{2i} := [F(F(x_{2i-2})) mod N]
     p := gcd(x_{2i} - x_i, N)
     if  p ∉ {1, N}  return  p
```

Pollard 对 $x_1, \cdots$的选择方法使得该算法在运行时间方面得到很大改进。但是，因为序列中的数值不再是随机独立选择的，前面的分析（说明了一对“good”数对存在的概率大约是 1/4）不再适用。但是从启发式分析的角度说，如果这个序列是“行为上随机”，希望得到“good”数对的概率仍是 1/4（从第 3 章的角度来说，这个序列肯定不是伪随机的，但是密码学伪随机性在并不是 Pollard 的 rho 算法成功的必要条件。）若F具有$F(x) = x^2 + b$的形式，其中$b \neq 0, -2 \mod N$，这给了F一种能高效地计算且在实践中似乎也能良好工作的形式。（参见文献[142]的 10.2 节中阐述的选择这种F的机理。）对任意具体的F函数给出对 Pollard 算法的精确分析，仍是一个有趣的公开问题。

8.1.3 二次筛选算法

Pollard 的 rho 算法的工作时间是待分解的数N的长度的指数函数。二次筛选算法运行时间为亚指数函数。直到 20 世纪 90 年代初，此算法是所知最快的因子分解算法，而且在分解最多约 300 比特长的数据时仍都采用此算法。描述二次筛选算法的一般准则，但是读者请注意，很多重要的细节将省略。

回忆前面曾经提到过，如果存在元素$x \in \mathbb{Z}_N^*$满足条件$x^2 = z \mod N$，则元素$z \in \mathbb{Z}_N^*$是模N的一个二次剩余；这种情况下称x为z的平方根。根据下面这些观察，也将在第 11 章用到，将作为开始：

(1) 如果$N = pq$是两个不同素数的乘积，那么N的每个二次平方剩余都恰好有四个平方根（参见 11.1.2 节的证明）。

(2) 给定x，y满足$x^2 = y^2 \mod N$且$x \neq \pm y \mod N$，则在时间多项式内计算N的一个非平凡因子是可能的。这是因为事实$x^2 = y^2 \mod N$意味着

$$0 = x^2 - y^2 = (x - y)(x + y) \mod N$$

进而$N \mid (x - y)(x + y)$。但是$N \nmid (x - y)$且$N \nmid (x + y)$，又因为$x \neq \pm y \mod N$。所以只有一种情况：$\gcd(x - y, N)$与N的一个素数因子相同。同时参见引理 11.21。

二次筛选算法试图产生一对数据x, y，它们的平方模N相等，其希望是能以固定的概率保持$x \neq \pm y \mod N$。该算法通过以下两个步骤搜寻x, y：③

第一步。规定一个小素数组成的集合$B = \{p_1, \cdots, p_k\}$，寻找不同的数$x_1, \cdots, x_l \in \mathbb{Z}_N^*$（$l > k$）使得$q_i \stackrel{\text{def}}{=} [x_i^2 \mod N]$足够“小”，以便能被分解（例如使用试除法），而且使q_i的所有素数因子都在B中。（$x_i > \sqrt{n}$也是需要满足，这能确保$x_i^2 > n$，从而在对x_i^2进行取模运算时不是平凡的）。这里省略如何寻找$\{x_i\}$的细节。

根据这一步，可以得到下面等式

$$\begin{aligned} x_1^2 &= \prod_{i=1}^{k} p_i^{e_{1,i}} \mod N \\ &\vdots \\ x_l^2 &= \prod_{i=1}^{k} p_i^{e_{l,i}} \mod N \end{aligned} \tag{8.3}$$

③ 为了简化陈述，一些细节被改变。描述仅仅表达了算法的主要思想。

通过对每个p_i的指数模2，降低指数，能够得到矩阵$\boldsymbol{\Gamma}$，定义为

$$\begin{pmatrix}\gamma_{1,1}\gamma_{1,2}\cdots\gamma_{1,k}\\ \vdots\\ \gamma_{l,1}\gamma_{l,2}\cdots\gamma_{l,k}\end{pmatrix}\stackrel{\text{def}}{=}\begin{pmatrix}[e_{1,1} \mod 2][e_{1,2} \mod 2]\cdots[e_{1,k} \mod 2]\\ \vdots\\ [e_{l,1} \mod 2][e_{l,2} \mod 2]\cdots[e_{l,k} \mod 2]\end{pmatrix}$$

目标是有$l>k$，且$\boldsymbol{\Gamma}$中没有全为0的行。一旦这个目标达到，即可进入下一步。

第二步。上一步中建立的矩阵$\boldsymbol{\Gamma}$，其行数比列数要多。因此，部分行叠加到某一行，使该行结果对2取模全为0。（此外，创建的时候，$\boldsymbol{\Gamma}$没有全0的行）。利用线性代数可以高效的选择这些符合条件的行。为了更清楚的示例，假设行l_1,l_2,l_3相加为全0行，也就是说

$$\begin{array}{c}\gamma_{l_1,1}\cdots\gamma_{l_1,k}\\ \gamma_{l_2,1}\cdots\gamma_{l_2,k}\\ +\ \gamma_{l_3,1}\cdots\gamma_{l_3,k}\\ \hline 0\cdots\ 0\end{array}$$

这里的加法是模2加。从式（8.3）中选择恰当的等式，得

$$X\stackrel{\text{def}}{=}x_{l_1}^2\cdot x_{l_2}^2\cdot x_{l_3}^2=\prod_{i=1}^{k}p_i^{e_{l_1,i}+e_{l_2,i}+e_{l_3,i}} \mod N$$

并且，通过对l_1,l_2,l_3的选择，对于所有的i，$e_{l_1,i}+e_{l_2,i}+e_{l_3,i}$的值是偶数，这就意味着可将其写为

$$X=(x_{l_1}\cdot x_{l_2}\cdot x_{l_3})^2=(\prod_{i=1}^{k}p_i^{(e_{l_1,i}+e_{l_2,i}+e_{l_3,i})/2})^2 \mod N$$

而且已经找到两个元素，其平方模N是相等的。虽然不能保证

$x_{l_1}\cdot x_{l_2}\cdot x_{l_3}\neq\pm\prod_{i=1}^{k}p_i^{(e_{l_1,i}+e_{l_2,i}+e_{l_3,i})/2} \mod N$，但至少能通过启发式的方法期望这种情况发生的概率约为1/2（因为X有四个平方根）。

例 8.4 取$N=377753$。有$6647=[620^2 \mod N]$，而且能够分解6647为（通过整数分解，而不需要任何模消减）

$$6647=17^2\times 23$$

因此，$620^2=17^2\cdot 23 \mod N$。类似的有

$$621^2=2^4\cdot 17\cdot 29 \mod N$$
$$645^2=2^7\cdot 13\cdot 23 \mod N$$
$$655^2=2^3\cdot 13\cdot 17\cdot 29 \mod N$$

所以

$$620^2\cdot 621^2\cdot 645^2\cdot 655^2=2^{14}\cdot 13^2\cdot 17^2\cdot 23^2\cdot 29^2 \mod N$$
$$\Longrightarrow [620\cdot 621\cdot 645\cdot 655 \mod N]^2=[2^7\cdot 13\cdot 17^4\cdot 23\cdot 29 \mod N]^2 \mod N$$
$$\Longrightarrow 127194^2=45335^2 \mod N$$

且$127194\neq\pm 45335 \mod N$。在计算$\gcd(127194-45335,377753)=751$，产生一个$N$的一个非平凡因子。

运行时间。在上述算法的讨论中已经省略了很多细节。但是，可以看出通过适当的优化，二次筛选算法能在$2^{\mathcal{O}(\sqrt{n\cdot\log n})}$时间内分解长度为$\mathcal{O}(n)$的数$N$。重要的是这种方法的运行时间是关于$N$的长度的亚指数时间。

8.2 计算离散对数的算法

假设$\mathbb{G}$是一个群，群上的运算可以高效执行；由附录 B.2.3 得结果可知，$\mathbb{G}$群中的幂运算也可以被高效执行。一个离散对数问题的实例可用下面的形式（参见 7.3.2 节）：给定$g \in \mathbb{G}$且$y \in \langle g \rangle$，找到x，使得$g^x = y$。（回忆一下，$\langle g \rangle$是由g生成的循环子群，$\{g^0, g^1, \cdots\} \subseteq G$。若$\langle g \rangle = \mathbb{G}$，则$g$是$\mathbb{G}$群的一个生成元，且$\mathbb{G}$是循环群。）这个回答$\log_g y$表示，且模$g$的阶是唯一的。有时在离散对数问题中把$g$称作“底数”。

攻击离散对数问题的算法分为两类：第一类用于计算“任意”群（这类算法有时称为“通用的”），第二类是用于某些特定的群。前一类算法中，通常选取$\langle g \rangle$群自身（当g不是群$\mathbb{G}$的生成元时，忽略$\mathbb{G} \setminus \langle g \rangle$中的元素）。在这种情况下，用$q$表示$\langle g \rangle$群的阶，并假设$q$是已知的。注意蛮力搜索离散对数$\log_g y$的时间复杂度为$\mathcal{O}(q)$。因此，只对运算时间优于此值的算法感兴趣。

下面讨论可用于任意群的算法：

(1) 由Shanks提出的“小步/大步”算法，对阶为q的群计算离散对数的时间复杂度为$\mathcal{O}(\sqrt{q} \cdot \mathrm{polylog}(q))$。

(2) 当对群的阶q的因子分解已知时，可用Pohlig-Hellman算法。当q含有小因子，这种方法将给定的离散对数问题的实例规约为具有较小阶的离散对数问题的多个实例。后者中每个实例的解答可以联合起来得到原问题的解答。

采取这种方法，关注在某些特定群中计算离散对数问题。举一个简单的例子，8.2.3 节提到了（加法）群$\mathbb{Z}_N$中的离散对数问题，并指出这种情形下离散对数可以在多项式时间内计算。

这个例子的重点在于说明了即使阶为q的任意循环群与$\mathbb{Z}_q$同构（回顾例 7.58），所有具有相同阶的循环群在代数上都是相同的，而离散对数问题的困难性主要依赖于所使用的群的特定表示。

实际上，加法群$\mathbb{Z}_N$中计算离散对数的算法依赖于一个事实：模N的乘法也被定义了。对任意不涉及模算术运算的群而言，上述论述没有任何意义。

将注意力转到在密码学方面有重要意义的群，8.2.4 节简要讨论了循环群$\mathbb{Z}_p^*$（p为素数）中离散对数的计算问题。给出一个“索引演算方法”的概要，这种方法能在亚指数时间内解决该群上的离散对数问题。此方法的详细细节超出本书范围。

就渐进运行时间方面，“小步大步”算法被认为是优化的，且达到了通用算法所能达到的（但是，已知道有些通用算法有相同的运行时间和更高效率的空间利用）。已经证明的一般群上求离散对数的复杂度下限，没有论及在任何特定群上求离散对数的困难性，正如示例中计算$\mathbb{Z}_N$上的离散对数那样容易。

当前，计算$\mathbb{Z}_p^*$（p为素数）上离散对数的已知最好算法是“通用数域筛法”。[④]从启发式的角度分析，这种方法在计算$\mathbb{Z}_p^*$（p的长度为$\| p \| = \mathcal{O}(n)$）上的离散对数的平均时间复杂度是$2^{\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})}$。重要的是，目前计算某个专门构造的椭圆曲线群（参见 7.3.4 节）的离散对数时，基本上没有非通用算法。这就意味着针对这种群，只要群的阶是素数（使用 Pohlig-

④ 这个名字被因子分解算法和计算离散对数的算法所共同拥有，这是个巧合，因为这些算法有很多相同的步骤。

Hellman 算法预先排除），目前只有指数时间的算法计算离散对数。

为理解后半部分所述内容在实践中的重要性，比较每一种类型的群的大小，使离散对数问题有相同的难度（这只是一种粗略的比较，因为更细致的比较，对初学者而言，需要考虑上面给出的大$\mathcal{O}$标号中的常量）。对于一个 1024 bit 的素数p，通用数域筛法计算$\mathbb{Z}_p^*$上离散对数，大约需要 $e^{1024^{1/3}} \times 10^{2/3} \approx e^{10 \times 4.6} = e^{46} \approx 2^{66}$ 步。这与用另一种计算阶为q的椭圆曲线群上离散对数的最优通用算法的时间复杂度相似，在那里q是一个 132 bit 的素数，$\sqrt{q} \approx 2^{132/2} = 2^{66}$。一个明显更小的椭圆曲线群，同时这些群上的操作更快，从而在不减少离散对数问题难度（至少对目前已知最好的技术而言）的条件下使用这些群。粗略地说，通过使用$\mathbb{Z}_p^*$上的椭圆曲线群得到的密码学方案，对诚实方而言效率更高，对攻击敌手来说困难程度相同。这解释了为什么它们能如此流行，特别是当其在较弱的硬件系统上实现时。

8.2.1 “小步大步”算法

Shanks 提出的“小步大步”算法是可以在时间$\mathcal{O}(\sqrt{q} \cdot \text{ploylog}(q))$计算阶为$q$的群中的离散对数。算法思想很简单。给定输入$g$和$y \in \langle g \rangle$，想象$\langle g \rangle$的所有元素都排列成下面的一个循环

$$1 = g^0, g^1, g^2, \cdots, g^{q-2}, g^{q-1}, g^q = 1$$

元素y必定在这个循环中的某个地方。计算并写下所有这个循环中的点需要花费时间$\Omega(q)$。另外，以大小为$t \stackrel{\text{def}}{=} \lfloor \sqrt{q} \rfloor$的间隔“标记”这个循环；也就是说，计算和记录$\lfloor q/t \rfloor + 1 = \mathcal{O}(\sqrt{q})$个元素

$$g^0, g^t, g^{2t}, \cdots, g^{\lfloor q/t \rfloor \cdot t}$$

（这就是“大步”。）注意，任意连续的“标记”之间的“间隔”最大是t。此外，易知$y = g^x$位于某个间隔中，因此t个元素

$$y \cdot g^0 = g^x, y \cdot g^1 = g^{x+1}, \cdots, y \cdot g^t = g^{x+t}$$

中的某一个必然等于刚才标记出来的某个点（这就是“小步”）。有$y \cdot g^i = g^{k \cdot t}$，因而很容易得到$y = g^{kt-i}$或者$\log_g y = [kt - i \mod q]$。算法的伪代码如下：

算法 8.5
小步大步算法
输入：$g \in \mathbb{G}$、$y \in \langle g \rangle$；g的阶q
输出：$\log_g y$
$t := \lfloor \sqrt{q} \rfloor$
for $i = 0$ to $\lfloor q/t \rfloor$:
计算 $g_i := y \cdot g^{i \cdot t}$
根据第 2 项对 (i, g_i) 排序
for $i = 0$ to t :
计算 $y_i := y \cdot g^i$
if 存在k，使得$y_i = g_k$，**return** $[kt - i \mod q]$

这个算法要求在$\mathbb{G}$中进行$\mathcal{O}(\sqrt{q})$次求幂和乘法运算，并且每次求幂可使用高效求幂算法在时间$\mathcal{O}(\text{polylog}(q))$完成。（事实上，除了第一个变量$g_1 = g^t$，其他每个变量$g_i$都能够使用一次乘法如$g_i := g_{i-1} \cdot g_1$来计算。同样地，每一个$y_i$也能够通过$y_i := y_{i-1} \cdot g$来计算。）在时间

$\mathcal{O}(\sqrt{q}\cdot\log q)$内能够将$\mathcal{O}(\sqrt{q})$个“数对”$(i,g_i)$进行排序，于是利用二分查找算法在时间$\mathcal{O}(\log q)$内检查$y_i$是否等于某个$g_k$，从而整个算法运行时间为$\mathcal{O}(\sqrt{q}\cdot\mathrm{polylog}(q))$。

例 8.6 现在演示这个算法在阶为$q=29-1=28$的循环群$\mathbb{Z}_{29}^*$中的运用，取$g=2$和$y=17$，设置$t=5$然后开始计算$2^0=1$，$2^5=3$，$2^{10}=9$，$2^{15}=27$，$2^{20}=23$，$2^{25}=11$。（注意这些运算都在$\mathbb{Z}_{29}^*$中进行。）然后计算$17\cdot 2^0=17$，$17\cdot 2^1=5$，$17\cdot 2^2=10$，$17\cdot 2^3=20$，$17\cdot 2^4=11$，$17\cdot 2^5=22$，发现$2^{25}=11=17\cdot 2^4$，因此可得$\log_2 17=25-4=21$。

8.2.2 Pohlig-Hellman 算法

当群的阶q的任何非平凡因子已经知道的情况下，Pohlig-Hellman 算法可用来加速对离散对数的计算。前面曾经提到，任何元素g的阶（记为$\mathrm{ord}(g)$）是最小的正整数i使得$g^i=1$。需要下面的引理：

引理 8.7 令 $\mathrm{ord}(g)=q$，并且有$p|q$，则可得$\mathrm{ord}(g^p)=q/p$。

证明 由于$(g^p)^{q/p}=g^q=1$，必有g^p的阶最大为q/p。若令$i>0$，满足$(g^p)^i=1$，于是$g^{pi}=1$，并且，因为q是g的阶，它必定满足$pi\geqslant q$，即有$i\geqslant q/p$。因此，g^p的阶恰为q/p。 ■

也将会使用通用的中国剩余定理：如果$q=\prod_{i=1}^{k}q_i$以及$\{q_i\}$是两两互素的素数（即对所有$i\neq j$，有$\gcd(q_i,q_j)=1$），那么

$$\mathbb{Z}_q\backsimeq\mathbb{Z}_{q1}\times\cdots\mathbb{Z}_{qk}\ \text{且}\ \mathbb{Z}_q^*\backsimeq\mathbb{Z}_{q1}^*\times\cdots\mathbb{Z}_{qk}^*$$

（这能够通过引进 k 来证明，使用中国剩余定理作为基本情况。）更进一步地说，通过扩展 7.1.5 节中的算法，能够高效地在一个$\mathbb{Z}_q$的元素的表示方法和一个$\mathbb{Z}_{q1}\times\cdots\mathbb{Z}_{qk}$的元素的表示方法之间转换。

现在描述 Pohlig-Hellman 的方法。给定g和y，希望找到一个x满足$g^x=y$。令$\mathrm{ord}(g)=q$，其因式分解

$$q=\prod_{i=1}^{k}q_i$$

已知，且$\{q_i\}$两两互素。（注意这没必要是q的完全素数因子分解。）知道

$$(g^{q/q_i})^x=(g^x)^{q/q_i}=y^{q/q_i}\qquad(i=1,\cdots,k)\tag{8.4}$$

令$g_i\overset{\mathrm{def}}{=}g^{q/q_i}$，因此得到在$k$个“更小的”群中的离散对数问题的$k$个实例，每一个大小为$\mathrm{ord}(g_i)=q_i$（由引理 8.7 可得）。

能够通过任何其他解决离散对数问题的算法来解决产生的k个实例中的每一个；具体地说，假设使用前一节提到的“小步大步”算法。解决这些实例得到了一个结果集合$\{x_i\}_{i=1}^k$，有$g_i^{x_i}=g^{q/q_i}=g_i^x$。（第二个等式由式 8.4 得到。）命题 7.50 意味着：对所有i有$x=x_i \mod q_i$。通过前面提到的通用的中国剩余定理，约束条件

$$x = x_1 \mod q_1$$
$$\vdots$$
$$x = x_k \mod q_k$$

唯一地确定x模q。因此，x可以从$x_1, \cdots, x_k$有效地重新生成。

例 8.8 利用上述思想再来计算$\mathbb{Z}_p^*$中的离散对数。这里，取$p = 31$，$\mathbb{Z}_{31}^*$的阶为$q = 31 - 1 = 30 = 5\times3\times2$。可以得到$g = 3$和$y = 26 = g^x$。有

$$(g^{30/5})^x = y^{30/5} \Longrightarrow (3^6)^x = 26^6 \Longrightarrow 16^x = 1$$
$$(g^{30/3})^x = y^{30/3} \Longrightarrow (3^{10})^x = 26^{10} \Longrightarrow 25^x = 5$$
$$(g^{30/2})^x = y^{30/2} \Longrightarrow (3^{15})^x = 26^{15} \Longrightarrow 30^x = 30$$

（重申一下，这里省略了“mod 31”。）解每个等式，得

$$x = 0 \mod 5, \quad x = 2 \mod 3, \quad x = 1 \mod 2$$

所以$x = 5 \mod 30$，事实上$3^5 = 26 \mod 31$。

假设q的因子分解如上，并且假定使用“小步大步”算法来解决每个“更小的”离散对数问题的实例，整个算法的运行时间为$\mathcal{O}(\text{polylog}(q) \cdot \sum_{i=1}^{k} \sqrt{q_i})$。由于$q$可有最多$\log q$个因子，这简化为$\mathcal{O}(\text{polylog}(q) \cdot \max_i\{\sqrt{q_i}\})$。取决于$q$的最大已知因子的大小，这个结果是前一节提供的$\mathcal{O}(\sqrt{q})$算法的明显改进。特别是当$q$有许多小的因数时，则在阶为$q$的群中离散对数问题能通过这个方法相对容易地解决。正如 7.3.2 节所讨论的，这导致在密码学应用中选择q为素数。

假如q有素数因子分解$q = \prod_{i=1}^{k} p_i^{e_i}$，前面描述的 Pohlig-Hellman 算法在$\mathcal{O}(\text{polylog}(q) \cdot \max_i\{\sqrt{p_i^{e_i}}\})$时间解决阶为$q$的群中的离散对数问题。使用更多的想法，这一结果还能改进到$\mathcal{O}(\text{polylog}(q) \cdot \max_i\{\sqrt{p_i}\})$（参见练习 8.3）。

8.2.3 $\mathbb{Z}_N$中的离散对数问题

前面两节的算法都是通用的，在某种意义上，这些算法并不关注所使用的群（除了知道群的阶以外）。本小节简要介绍的目的只是强调非通用的算法，即那些利用群（及其表示）的特殊性的算法，这些算法效率更高。

现在考虑计算加法群$\mathbb{Z}_N$（N是任意的）的离散对数。这个问题对于底数$g = 1$的情况是平凡的：由于$y \cdot 1 = y \mod N$，对于任意$y \in \mathbb{Z}_N$的离散对数就是整数y本身。注意，等式左边的y表示整数y，然而等式右边的y表示元素$y \in \mathbb{Z}_N$。然而，群$\mathbb{Z}_N$的特殊性质允许从本质上把这两个y视为可互相交换。

当使用生成元$g \neq 1$时，事情变得有点复杂（练习 8.4 处理了这种情况，即g不是$\mathbb{Z}_N$的生成元。）令$g \in \mathbb{Z}_N$为生成元，给定某个y，想要计算的x满足$x \cdot g = y \mod N$。运用定理 B.18（还有 1 是生成元的事实），得到$\gcd(g, N) = 1$。于是g有模N的乘法逆元g^{-1}（该逆元可以通过附录 B.2.2 讨论的高效算法来计算）。需要的结果就是$x = y \cdot g^{-1} \mod N$。

有趣的是这里再次使用到$\mathbb{Z}_N$的非通用属性。在这个例子中，算法隐含着使用了一个事实：除了群运算（N的模加运算）外，群的元素上还定义了N的模乘运算。

8.2.4 索引演算方法

索引演算方法能够以p的长度表示的亚指数时间内，解决循环群$\mathbb{Z}_p^*$（p为素数）上的离散对数问题。细心的读者可能发现这里描述的算法类似于 8.1.3 节介绍的二次筛选算法。在该算法中，仅讨论索引演算方法的主要思想，细节超出了本书的范围。同样地，引入简化使得叙述清晰。

索引演算方法包括两步。重要的是第一步需要的知识仅仅只是模p以及底数g，因此该步骤可以作为一个“预先处理步骤”在给定y之前完成，这里的目标是计算y的离散对数。同样的原因，在计算多个离散对数问题的实例时，第一步只需要运行一次，只要这些实例有共同的p和g。

步骤 1。令$q=p-1$，是$\mathbb{Z}_p^*$的阶。固定小素数的集合$B=\{p_1,\cdots,p_k\}$。在这个步骤中，发现$l\geqslant k$个不同的非零值$x_1,\cdots,x_l\in\mathbb{Z}_q$，满足$g_i\overset{\text{def}}{=}[g^{x_i} \mod p]$是“小”的，以便$g_i$的因子分解（比如使用试除法），并且$g_i$的所有素数因子都在集合$B$中。这里不讨论如何发现这些$\{x_i\}$。

此后，得出具有以下形式的l个方程：

$$\begin{aligned} g^{x_1} &= \prod_{i=1}^{k} p_i^{e_{1,i}} \mod p \\ &\vdots \\ g^{x_l} &= \prod_{i=1}^{k} p_i^{e_{l,i}} \mod p \end{aligned}$$

两边取离散对数，将其转换为线性方程：

$$\begin{aligned} x_1 &= \sum_{i=1}^{k} e_{1,i}\cdot\log_g p_i \mod (p-1) \\ &\vdots \\ x_l &= \sum_{i=1}^{k} e_{l,i}\cdot\log_g p_i \mod (p-1) \end{aligned} \tag{8.5}$$

注意其中$\{x_i\}$和$\{e_{j,i}\}$（其中，$0\leqslant j\leqslant l$，$1\leqslant i\leqslant k$）是已知量，但$\{\log_g p_i\}$是未知的。

步骤 2。现在给定一个元素y，来计算$\log_g y$。找到一个数$x^*\in\mathbb{Z}_q$，满足条件$g^*\overset{\text{def}}{=}[g^{x^*}\cdot y \mod p]$是“小”的，使得$g^*$能够进行因子分解并且$g^*$的所有素数因子都在集合$B$中。这里不讨论如何找到$x^*$。

于是

$$\begin{aligned} & g^{x^*}\cdot y = \prod_{i=1}^{k} p_i^{e_i^*} \mod p \\ \Longrightarrow & x^*+\log_g y = \sum_{i=1}^{k} e_l^*\cdot\log_g p_i \mod (p-1) \end{aligned}$$

其中x^*和$\{e_i^*\}$都是已知的。将上式与式（8.5）联立，发现在$l+1 \geqslant k+1$个线性方程中包含了$\{\log_g p_i\}_{i=1}^k$和$\log_g y$的$k+1$个变量。使用线性代数[⑤]的方法（假设方程组不是不定的），能够计算出所有的未知数特别是能够计算出所期望的$\log_g y$。

例 8.9 令$p=101$，$g=3$，$y=87$。有$3^{10}=65 \mod 101$，并且$65=5\cdot 13$（在整数域上操作），类似地，$3^{12}=80=2^4\cdot 5 \mod 101$。以及$3^{14}=13 \mod 101$。因此可得线性方程：

$$\begin{aligned} 10 &= \log_3 5 + \log_3 13 \mod 100 \\ 12 &= 4\cdot \log_3 2 + \log_3 5 \mod 100 \\ 14 &= \log_3 13 \mod 100 \end{aligned}$$

此外还能得到$3^5\cdot 87 = 32 = 2^5 \mod 101$，或者是

$$5 + \log_3 87 = 5\cdot \log_3 2 \mod 100 \tag{8.6}$$

将第 2 个等式和第 3 个等式相加减去第 1 个等式，得到$4\cdot \log_3 2 = 16 \mod 100$。这并不唯一确定$\log_3 2$，但是可以得出$\log_3 2 =$4、29、54 或者79（见练习 8.4）。尝试所有的可能，表明$\log_3 2 = 29$。代入式（8.6）得到结果$\log_3 87 = 40$。

运行时间。可以发现，通过适当的优化，索引演算算法来计算$\mathbb{Z}_p^*$域上的离散对数（这里p是长度为n的素数）的时间复杂度为$2^{\mathcal{O}(\sqrt{n\cdot \log n})}$。关键是其为关于$||p||$的亚指数。注意该方法在运行时间复杂度上与二次筛选法相同。

参考文献和扩展阅读材料介绍

利用$p-1$方法解决因子分解的问题由 Pollard 在 1974 年提出[115]，他的 rho 算法在 1975 年提出[116]。（他发表了一个计算离散对数的 rho 算法。）二次筛选法是 Pomerance 基于早期 Dixon 的观点提出的[117]。

计算离散对数问题的“小步大步”算法归功于 Shanks[126]，而 Pohlig-Hellman 算法是在 1978 年提出的[114]。对于索引演算算法以及计算离散对数方法，读者可参考 Odlyzko 的综述[112]。

由 Wagstaff[142]、Shoup[131]、Crandall、Pomerance[40]和 Bressoud[29]编写的教科书都提供了关于因子分解和计算离散对数的更详细的信息。

计算离散对数的所谓通用算法（即不论群的表示方法如何，算法应用于任意群，）的更低的界是由 Nechaev[111]和 Shoup[128]提出的。

Lenstra 和 Verheul[94]提供了综合的讨论，即现有的因子分解和计算离散对数的算法如何影响实践中对密码学参数的选择。

练　习

8.1 为了提高 RSA 密钥生成算法的速度，有人建议生成很多小随机素数，然后将它们相乘并加上 1（当然，要检测结果是否为素数）来产生一个的素数。忽略结果的确是素数的概

⑤ 这个名字被因子分解算法和计算离散对数的算法所共同拥有，这是个巧合，因为这些算法有很多相同的步骤。

率这个问题，你认为这个方法如何？

8.2 这里展示如何在时间$\mathcal{O}(\text{polylog}(q)\cdot\sqrt{p})$解决一个阶为$q=p^e$的循环群的离散对数问题。给定一个阶为$p^e$的生成元$g$，以及值$y$，想计算$x=\log_g y$。注意，$p$可以很容易从$q$计算出来（见练习 7.11）。

（1）说明如何在时间$\mathcal{O}(\text{polylog}(q)\cdot\sqrt{p})$计算$[x \mod p]$。

提示：求解等式：

$$(g^{p^{e-1}})^{x_0}=y^{p^{e-1}}$$

并在 Pohlig-Hellman 算法中使用同样的思想。

（2）令$x=x_0+x_1\cdot p+\cdots+x_{e-1}\cdot p^{e-1}$（$0\leqslant x_i<p$）。上一步已经确定出$x_0$。说明如何在$\text{polylog}(q)$时间计算$(g^p)^{x_1+x_2\cdot p+\cdots+x_{e-1}\cdot p^{e-2}}=y_1$。

（3）使用递归得到针对原问题所声称的运行时间。（注意$e=\text{polylog}(q)$。）

8.3 令q的素数因子分解$q=\prod_{i=1}^{k}p_i^{c_i}$。利用前面问题的结果，说明一种修改 Pohlig-Hellman 算法的方法，能够解决阶为q的群中的离散对数问题，且在$\mathcal{O}(\text{polylog}(q)\cdot\sum_{i=1}^{k}c_i\sqrt{p_i})=\mathcal{O}(\text{polylog}(q)\cdot\max\{\sqrt{p_i}\})$时间内。

8.4 （1）如果$ab=c \mod N$，且$\gcd(b,N)=d$，那么：

① $d|c$；

② $a\cdot(b/d)=(c/d) \mod (N/d)$；

③ $\gcd(b/d,N/d)=1$。

（2）描述如何运用上述方法在$\mathbb{Z}_N$中高效地计算离散对数，即使底数g不是$\mathbb{Z}_N$的生成元。

8.5 使用声称 8.2 的变体，完整描述和分析对密码学散列函数的固定空间生日攻击，这个攻击的简短描述在 4.6.3 节中。

第9章　对称密钥管理和公钥革命

9.1　对称密钥加密的局限性

前面已经说明，对称密钥加密可用来确保在不安全的信道中进行安全的通信。因而它看上去似乎完全解决了密码学的主要问题，然而事实并非如此，接下来要讨论一些原因。

密钥分配的问题。对称密钥加密需要通信双方有共享和保密的密钥。仍未讨论这些共享的密钥最初是如何获得的。很明显，由于通信途中密钥可能被敌手窃听到，这些密钥不能简单地在不安全的通信信道中传送。

使用可实现的安全通道，可以初步共享一对密钥，例如，使用可信的信使服务。尽管政府、军队、智囊组织和其他一些类似的实体有条件使用这种方法共享密钥，但是对于普通人来说却不大可行。（事实上，据说连接莫斯科和华盛顿的红线电话是用“一次一密”来加密，其中的密钥是通过外交官共享的，他们随身携带着装满填充码的公文包从一个国家飞到另一个国家。）另一个让通信双方共享密钥的更加实用的方法是组织一次见面会议，在此会议上产生一个随机的密钥，将密钥的备份发给各方。尽管可能会设想两个使用者在一个黑暗的角落进行这样的一次见面，但更常见的是见面发生在标准的工作场所。例如，一个经理可能会在每个员工上班的第一天给他们一个共享的密钥。

虽然，在一名经理和每个员工共享密钥的情况下，这种方法可能可行，但是若所有的员工相互之间都需要共享密钥，这种办法的扩展性则较差。上述方法意味着，每次有一名新的员工到来，所有其他的员工将不得不和她共享一个新的密钥。这样产生很大的麻烦，特别是当公司较大且在很多不同的地区都有分公司时。

一种部分解决这种情况的办法是：利用一个指定的“控制者”（即公司的 IT 经理），帮助所有的员工两两之间建立共享的密钥。特别地，当一个新的员工进入公司，这个控制者产生一些随机的密钥k_1，…，之后把这些密钥亲自交给新来的员工，然后把k_i发给第i个进入公司的员工，这个密钥是用控制者和第i个员工共享的密钥加密过的。（这里假设控制者是一位员工，因此所有的员工均与之共享一对密钥。）这是一个非常麻烦的方法。更重要的是，由于密钥不完全是安全的，这种方案没有给出一个完整的解决方案。一个不诚实的控制者可能解密员工间的所有通信，而且如果一个攻击者侵入了控制者的电脑，那么该系统中的所有的密钥将均被泄露。

密钥的存储和保密。继续考虑前面提到的各个员工之间共享密钥的工作环境。当有U个员工，系统里密钥的数量是$\binom{U}{2} = \Theta(U^2)$。更重要的是，这就意味着每个员工持有$U-1$个密钥。事实上，情况可能会更加糟糕，因为员工还可能需要其他的密钥以保证和远程资源的安全通信，例如数据库、远程服务等。当组织变得很大时，这种方法会在很多层面产生大量的问题。首先，密钥的增长是一个很大的管理问题。其次，所有这些密钥必须安全存储。然而密钥数越多，保护就越困难，敌手窃取密钥的概率就越大。计算机系统经常会被病毒、

蠕虫和其他各种恶意软件感染。这些恶意程序可能用来窃取密钥，然后悄悄地将密钥通过网络传送到攻击者手里；这些程序在过去曾经出现过，因此它们的存在并不只是一个理论上的威胁。既然如此，将密钥存储在个人计算机上通常不是一个合理的解决方案。

不论各方持有多少密钥，潜在的密钥丢失通常是一个值得关心的问题。当只有少量的密钥需要存储时，有一些可用的好办法来解决这种威胁。一个典型的解决办法是将密钥存储在智能卡上，它是一个高度保护的硬件设备。这种智能卡能运用存储的密钥完成密码学计算，确保这些密钥不会留在用户的个人电脑上。由于智能卡比个人电脑更难攻击，如它们不会被感染病毒，这就提供了一种很好的保护用户密钥的方法。不幸的是，智能卡在内存上很受限制，因此不能存储成百（或上千）的密钥。

当然，理论上安全存储任意多的密钥是可能的，而且安全存储的问题可以通过组织来解决，如政府，他们可投入大量的资源，尽管通常并不是很方便。本书第二作者在多年前曾对工作于西欧的美国大使馆的一个人聊过。那个人的工作是解密所有收到的消息，系统大致如下：无论什么时候当一个加密的信息到达时，他将信息带到一个锁着的有人守卫的房间里，这个房间中存放着所有的密钥。然后他找到恰当的密钥，解密那条消息。这件事的关键在于：政府和一些大型的组织可以仅仅使用对称密钥加密来确保通信安全。然而，这些解决办法费用昂贵，扩展性不好，并且不适用于工业或个人使用。

开放系统。正如上述所讨论的，对称密钥加密很难部署和维护，需要管理和对安全地存储大量的密钥。至少在理论上，它可以用来解决在“封闭”系统中的安全通信，在这种系统中可以使用物理方式分配密钥。不幸的是，在“开放”环境下，通信方无法安全地分配密钥，对称密钥加密是不充分的。例如，在网上购物时需要加密，或者要给国外的同事发邮件（发送者可能从来没有见过此人），仅仅使用对称密钥加密无法解决。但是由于对称密钥加密的重要性，重申一点：

基于对称密钥加密的方法不足够解决开放系统中的安全通信问题，在这个开放系统中通信方无法见面，也无法有短暂的交互。

这意味着我们必须进一步寻找适当的解决办法。

9.2 局部解决方法——密钥分配中心

如前所述，使用对称密钥加密存在着三个独特的问题：①密钥分配问题；②大量密钥的管理问题；③在开放系统中的不适用性。虽然完全解决第一个问题是不可能，但是有一个方法可以缓和前两个问题，并且可让对称密钥加密在大型组织中使用。一种扩展的想法（不进一步讨论）允许在“半开放”系统中使用对称密钥加密，这种半“开放”系统由多个组织组成，他们在一定范围内相互信任。

密钥分配中心。仍然考虑大型组织中所有员工能够安全通信的情况。每两个员工共享一对密钥的解决办法导致了密钥的大量增长。一个不同的方法是基于所有的员工可能会信任某个实体这样一个事实，如组织中的 IT 经理，至少关于与工作有关的信息的安全性是值得信任的。因此，IT 经理可以建立一个服务器，名叫密钥分配中心（Key Distributed Center，KDC），它可以充当希望通信的双方的一个中介。一个密钥分配中心可以这样工作：首先，所有的员工均和密钥分配中心共享一个密钥，这个密钥可以在员工第一天工作的时候产生和共享。然后，当员工中的 Alice 想和 Bob 进行安全的通信，她就向密钥分配中心发送一条消息

“Alice 想和 Bob 通信”（这条消息要使用 Alice 和 KDC 共享的密钥进行认证）。接着，KDC 选择一个新的随机的密钥，叫做会话密钥，将这个密钥传给 Alice 和 Bob，并且这个会话密钥分别用 Alice 和 Bob 拥有的密钥进行加密。一旦他们解密出会话密钥，他们就可以使用它来安全地进行通信了。当通信结束后，他们能够（也必须）毁掉这个会话密钥，因为当他们后来想再次通信时，他们可以再次联系密钥分配中心。注意这只是解决办法的一个轮廓，不足够提供需要的安全。（分析这些方法需要的严格定义和证明超出了本书的范围。）然而，它足以在关于如何让对称密钥加密工作这个问题上给一点直观感受。

考虑一下这种方法的优点：

(1) 每个员工只需要存储一个密钥，因此可以使用智能卡。密钥分配中心需要存储大量的密钥，但是 KDC 在一个安全的地方并且给予最高等级的保护措施来防止网络攻击。

(2) 当一个新的员工进入公司，所有要做的只是在这个员工和 KDC 之间产生一对密钥。其他的员工不用更新他们持有的密钥。当一个员工离开时也是这样。

这种方法缓和了对称密钥加密中的两个问题：密钥分配简单化了（当某方加入组织时只需要共享一对新密钥），而且密钥的存储问题也解决了（除 KDC 外，其他各方只需存储一个密钥）。这种方法让对称密钥加密在单一组织中变得可用，组织中存在被所有人共同信任的实体。

话虽如此，这种方法也有一些缺点：

(1) 如果成功地攻击了 KDC，将导致服务所有各方的系统被完全攻破。这样，攻破 KDC 的动机将会变得很大，这就增加了安全风险。另外，组织内部可以访问 KDC 的攻击者（例如，IT 经理）将可以解密各方之间的所有通信。

(2) 密钥分配中心是单一失效点：如果 KDC 崩溃了，安全通信将不可行。由于所有的员工都不断地与 KDC 联系，它的负荷可能会很大，继而增加了 KDC 崩溃和响应缓慢的可能性。

一个简单的解决办法是复制这个 KDC。这种方法可行（在实践中可完成），但是更多 KDC 的存在意味着对于系统有了更多的攻击点。而且，这将使增加和减少员工变得更加困难，因为更新的信息需要安全地传输给所有的 KDC。

上面基于 KDC 的解决方案，类似于之前提供的通过一个指定的“控制者”设置所有员工之间共享密钥的方法。前者中，控制者相当于一个离线的 KDC。由于控制者仅在初始阶段介入，所有的员工仍然要持有很多密钥。与在线的 KDC 方案相比，KDC 可以在需要的时候用来在双方之间交换密钥。

使用 KDC 的密钥分配协议。在关于使用 KDC 的安全密钥分配的文献中可以找到大量相关的协议，其中一个是 Needham-Schroeder 协议。这里将不讨论协议的细节，取而代之，建议读者查阅本章后的参考文献获得更多细节。应注意该协议的工程特点，当 Alice 和 KDC 联系，要求和 Bob 通信时，KDC 并不是将加密的会话密钥发给 Alice 和 Bob，而是使用 Alice 和 Bob 的密钥加密会话密钥后发送给 Alice，然后 Alice 使用 Bob 的密钥加密会话密钥后发送给 Bob；如图 9.1 所示。该协议如此设计是考虑到 Bob 可能不在线；这会导致一个问题，即 KDC 可能无休止地等待 Bob 的响应。通过将两个密钥加密的会话密钥发送给 Alice，KDC 避免了维护一个开放的会话。KDC 发给 Alice 的，用 Bob 的密钥加密的会话密钥叫做票据（ticket），它可以被看作是允许 Alice 和 Bob 通话的一个凭证。

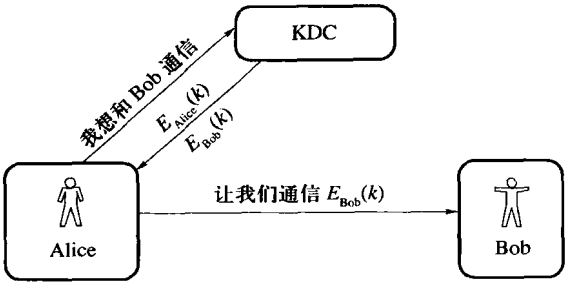


图 9.1　通用密钥分配协议

需要强调，使用 CPA 安全的加密机制来加密会话密钥使得该协议很容易受到攻击；有必要使用一个如 4.9 节所介绍的安全消息传输机制。

在实践中类似于 Needham-Schroeder 这样的协议被广泛地使用。很多情况下，Alice 和 Bob 可能不都是用户，可能仅仅 Alice 是一个用户，而 Bob 是一个资源。比如，Alice 希望阅读某个服务器上的被保护磁盘上的文件。Alice 向 KDC 请求许可，KDC 发给 Alice 一个票据作为允许 Alice 在被保护磁盘上读数据的一个凭证。这个票据包含了一个会话密钥（正如上面所述），这样 Alice 和服务器之间的通信就可以被保护。一个使用非常广泛的，可以通过 KDC 进行用户认证和安全通信的协议是 Kerberos 协议，它是由 MIT 开发的。Kerberos 协议有一些重要的特点，它被微软的 Windows（包括 Windows 2000 和之后的版本）操作系统采用，以保证内网安全。

留意在实践中 Alice 和 KDC 共享的密钥通常是较短的且容易记忆的口令（因为通常用户可能没有用来存储长密钥的智能卡）。这种情况下，很多要考虑和处理的安全问题就会增加。和前面一样，更多相关信息的文献列在本章后供感兴趣的读者阅读。

9.3　公 钥 革 命

KDC 和类似 Kerberos 的协议非常有用，并且实践中广泛使用。但是，它们仍然没有解决像在互联网这样的开放系统中的密钥分配问题，这些开放系统中没有隐蔽的信道。（在 KDC 的情况中，隐含假设存在一个隐蔽的信道，可用来产生员工和 KDC 之间的初始密钥。）为了能在没有隐蔽信道的情况下进行秘密通信，必须使用一些完全不同的方法。1976 年，Diffie 和 Hellman 发表了一篇看似天真的名为“密码学的新方向”的文章[47]。这篇文章的影响是巨大的。除了引入一种完全不同的看待密码学的方式外，它迈出了第一步，密码学走出秘密领域，走入公开领域。在描述 Diffie 和 Hellman 的基本思想之前，引用一下他们论文的前两段：

我们今天站在密码学革命的边缘。廉价的数字硬件的发展已经摆脱了机械计算的设计限制，并降低了高端加密设备的成本，使得可以在像远程自动取款机和计算机终端这样的商业应用中使用。

反过来，这些应用需要新的加密系统，使得安全密钥分配的必要性降到最小，并且提供和书面签名等同的功能。同时，信息理论和计算机科学的理论发展展现了提供可证明安全密码系统的前景，把这一古老的艺术转变成科学。

Diffie 和 Hellman 并不是夸大其词，他们谈到的革命很大程度上归功于其工作。在 1976 年以前，普遍认为若首先不共享密钥则进行加密是不可行的，但是，Diffie 和 Hellman 观察到

世界上存在一种天然的不对称：存在着某些很容易完成但是反过来却不容易完成的行为。例如，没有钥匙时，可以（容易地）关锁，但是无法再解锁。更明显的一个例子，很容易打碎一个玻璃花瓶，但是想要碎片再还原却是十分困难的。从算法角度更确切地说，计算两个大素数的乘积很简单，但是从一个积中找出这两个素数却很难。（这正是前面章节所提到的因子分解问题。）这些现象的存在意味着可以构造一种加密方案，它不依赖于共享密钥，而是依赖于一种加密很容易，但除了指定的接受者以外其他人解密都很困难。

更详细地说，设想这样一个密码方案，方案中存在两个而不是一个密钥：一个是加密密钥，被发送者用来加密消息；另一个是解密密钥，被消息的接收者用来从密文中恢复明文。此外，它的存在是令人惊讶的；即使攻击者知道加密密钥（而不是解密密钥），加密信息的保密性仍然受到保护。具有这种属性的加密方案叫做不对称加密方案或者公钥加密方案，它和目前了解的对称加密或者对称密钥加密是相反的。在公钥加密方案中，加密密钥叫做公钥，由于它是由接收者公开的，任何人想发送加密消息也可以这样做；解密密钥叫做私钥，因为它是由接收者维持完全保密的。

公钥加密的发明的确是密码学的一次革命。这并不是巧合，直到 20 世纪 70 年代末，80 年代早期，加密和密码学大都属于智囊和军事组织。直到公钥技术的到来，才使密码学传播到大众。

公钥加密机制使秘密通信不再依赖于保密信道[①]。接收者可以在报纸或者其网页上公开其公钥，这样可使每个人都能发送加密的信息给她。接收者也可以简单地通过邮件将公钥发送给感兴趣的发送方，不用担心窃听所有通信的攻击者也观察到这个公钥[②]。

下面来总结一下公钥加密时怎样解决在 9.1 节中提到的对称密钥加密的局限性的：

(1) 公钥加密可以让密钥分配在公开的信道中完成。这可以简化系统的首次部署，也可以减轻通信方加入或离开时的系统维护。

(2) 公钥加密大量减少了需要存储的密钥。即使所有的通信双方都希望能安全地通信，各方仅仅需要安全地存储自己的私钥。其他通信方的公钥，要么是当需要的时候去获取，要么以非安全的方式存储（即可公开读取的方式）。

(3) 公钥加密（更）适合于公开环境，通信各方没有事先交互，但是却希望进行安全地通信。例如，一个商人可以在线发布他的公钥；任何购物者可获得这个公钥，用来加密自己的信用卡信息。

公平起见，应该强调上述讨论掩盖了一些问题，最重要的是公钥可认证分配的必要性。对于这一点的详细讨论，需等到 12.8 节。

公钥原语。 Diffie 和 Hellman 引入了三个公钥（非对称密钥）原语。第一个是上面提到的公钥加密；将在第 10 章和第 11 章学习。第二个是类似于消息鉴别码，叫数字签名，将会在第 12 章介绍。和消息鉴别码一样，数字签名机制是用来阻止对消息不可检测的篡改。然而与 MACs 不一样的是，消息的真实性能被知道发送者公钥的任何人验证。这由此产生了深远的影响。具体而言，可将由 Alice 数字签名的文件呈给第三方，比如法官，来证明确实是 Alice 签署了这份文件。因为只有 Alice 知道相应的私钥，这可以作为是 Alice 签署这份文件的证据。这个特性叫做不可否认性，且广泛地应用于电子商务中。例如，可用数字签名签署一份

① 但是假定是认证的通道，各方“知道”谁在信道的另一端。在 12.8 节，当讨论 CA 和公钥基础设施时会重新考虑这一假设。

② 假设敌手不能替换其公钥，即假定存在一个公开且认证的信道；见上一个页末注。

合同，发送签名的电子购物单或者承诺付款等。数字签名也用于公钥基础设施中帮助分发公钥。详细内容将在第 12 章中讨论。

第三个由 Diffie 和 Hellman 引入的原语是交互式密钥交换。交互式密钥交换协议是一种方法，该方法中没有共享任何秘密信息的通信双方，可在公开的信道中产生一对共享的安全密钥。这里确保的一个主要属性是：一个可以窃听到通信中所有信息的敌手无法知道密钥的任何信息。注意，安全密钥交换的存在是很令人惊叹的，它意味着从原则上讲，如果你和你的朋友站在一个房间的两端，你们可以相互喊叫，产生一对共享的密钥，并且其他人（听到你所有说的话）无法知道关于密钥的任何事。

密钥交换和加密的主要区别是：前者是一种交互协议，所以要求通信双方同时在线。相反，加密（通常）不是一个交互的过程，因此更适合于某些应用程序。例如，安全邮件需要加密，因为当邮件发送时接收者不必在线。

尽管 Diffie 和 Hellman 引入了以上三种原语（公钥加密、数字签名和密钥交换），他们只给出了密钥交换的解释。下一节将描述 Diffie-Hellman 密钥交换协议。一年以后，Ron Rivest、Adi Shamir 以及 Len Adleman 提出了 RSA 问题，并且基于这个问题的困难性第一次提出了公钥加密和数字签名方案。这种方案的变种是广泛使用的密码学方案。有趣的是，在 1985 年，El Gamal 提出的加密方案仅仅是对 Diffie-Hellman 密钥交换协议的轻微的改动。尽管 Diffie 和 Hellman 没有成功地构造（非交互）公钥加密方案，但是已非常接近了。

历史。阅读由 Diffie 和 Hellman 开创的公钥革命的历史是令人着迷的。当时其他一些人也正研究一些类似的想法。另外一个从事类似研究的独立工作是 Ralphe Merkle，他被大多数人认为是公钥密码学的共同发明人之一。（尽管他是在 Diffie 和 Hellman 之后发表论文的。）也应该提到 Michael Rabin 的研究，在 Ron Rivest、Adi Shamir 和 Len Adleman 的方案提出约一年以后，他构造了基于因子分解困难性的签名机制和公钥加密机制。早在 Diffie-Hellman 的论文发表之前（20 世纪 10 年代早期），智囊机构（英国的 GCHQ）就已经知道公钥加密方案。尽管公钥加密中的数学基础在 1976 年之前就已经被发现了，但是事实上，这项新科技的广泛传播是在 Diffie 和 Hellman 的出现以后。

随着工作的展开，Diffie 和 Hellman（还有密码学方面的其他人）受到被起诉的威胁。这是因为在国际武器交易条例中，密码学方面的科技文献被认为是战争工具。尽管密码学的出版物不久之后被接受并且广泛传播，但当时这些出版物被认为是极其敏感的。Hellman 讲了一个故事，他曾经在一次会议上亲自陈述论文，该论文是与 Ralph Merkle 和 Steven Pohlig 的共同研究成果（他们俩那时是研究生），这是因为斯坦福大学的总顾问建议学生不要陈述论文以免被起诉。幸运的是，美国政府没有这样，而且在密码学方面发表论文也被继续允许。（美国仍然限制出口密码实现设备。但从 2000 年开始，这些限制被放松且现在几乎感觉不到。）

9.4 Diffe-Hellman 密钥交换

本节将介绍 Diffie-Hellman 密钥交换协议，并在窃听敌手存在条件下证明它的安全性。防御被动的窃听敌手的安全相对较弱，实践中的安全必须能够对抗主动敌手，他们可能会劫持并修改通信方之间的消息。这里不给出针对这类敌手的安全定义，这超出了本书的范围。（而且，这里感兴趣的是通信双方之前没有共享密钥，在这种情况下为防止敌手伪装成其中一个通信方能做的不多。）

基本状况以及安全定义。考虑这样一种情形，通信双方 Alice 和 Bob 运行协议产生一对共享的保密密钥；把该协议记为Π（这样，Π可以看作是协议中给 Alice 和 Bob 的一组指令）。首先 Alice 和 Bob 持有安全参数1^n，然后（相互独立地）随机选择并执行协议Π。在协议的最后，Alice 和 Bob 分别输出密钥 k_A，$k_B \in \{0,1\}^n$。对Π协议最基本的要求是 $k_A=k_B$（对所有 Alice 和 Bob 的随机选择）。由于只需要让协议满足这个要求，简单地称：诚实地执行Π协议产生了密钥 $k=k_A=k_B$。

现在定义安全。直观的说，如果 Alice 和 Bob 输出的密钥完全不被窃听敌手所知晓，那么密钥交换协议是安全的。形式化的定义则要求：窃听协议执行的敌手无法区分协议执行后产生的（即 Alice 和 Bob 共享的）密钥k和长度为n的随机密钥。这一安全定义比仅要求敌手无法准确地计算k要强得多，如果通信方想用k来完成一些密码学的任务，则这个定义是必要的（例如，在一个对称密钥加密机制中使用 k）。

对以上的内容进行形式化描述，令Π为密钥交换协议，$\mathcal{A}$是攻击者，n是安全参数。实验如下：

密钥交换实验$\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$：

(1) 两个持有1^n的通信方执行Π协议。执行协议后将得到通信双方发送的所有消息的一个副本trans，以及通信双方输出的密钥k。

(2) 随机选择一个比特$b \leftarrow \{0,1\}$。如果$b=0$，则均匀随机地选择一个二进制数$\hat{k} \leftarrow \{0,1\}^n$，如果$b=1$，则令$\hat{k} := k$。

(3) 敌手得到了消息副本trans和$\hat{k}$，输出一个比特b'。

(4) 如果$b'=b$，则实验输出为 1，否则为 0。（在$\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1$时，称$\mathcal{A}$成功。）

敌手$\mathcal{A}$得到副本trans表示$\mathcal{A}$窃听整个协议执行的过程，即敌手可看到通信双方之间的所有通信。当然，在现实世界中，$\mathcal{A}$不会得到任何密钥；上面实验中的敌手得到$\hat{k}$只是作为一个手段来定义敌手$\mathcal{A}$攻破Π。也就是说，如果敌手$\mathcal{A}$可以正确地区分$\hat{k}$是执行协议得到的密钥，还是一个独立于副本的完全随机的密钥，那么敌手就成功地攻破了Π。正如预期的一样，如果敌手成功的可能性上限大于1/2的部分可忽略，则认为Π是安全的。即：

定义 9.1 则密钥交换协议Π在存在窃听者的情况下是安全的。

$$\Pr[\mathsf{KE}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] \leqslant \frac{1}{2} + \mathrm{negl}(n)$$

如果对每个概率多项式时间的敌手$\mathcal{A}$，都存在一个可忽略函数negl，满足

密钥交换协议的目的在于总是能产生一个共享的密钥k，它可以用于通信双方之间进一步的保密通信，例如，分别使用对称密钥加密和消息鉴别码来加密和认证随后的通信。直觉上，由于密钥k“看起来”像一个随机密钥，可期望这个方法是安全的。然而，上述定义没有指出密钥的使用是否能获得预期的安全特性，并且仅仅依赖直觉往往是危险的。幸运的是，可以证明上述定义对于这个应用来说是足够的（参见练习 9.1）。

Diffie-Hellman 的密钥交换协议。接下来描述一下出现在 Diffie 和 Hellman 原论文中的密钥交换协议（尽管它们没有这里这么正式）。取$\mathcal{G}$为概率多项式时间算法，其中输入为1^n，输出一个循环群$\mathbb{G}$，群的阶为q（$||q|| = n$），生成元为g。例如，7.3.3 节中的$\mathcal{G}$。描述 Alice 和

Bob 的 Diffie-Hellman 密钥交换协议在构造方案 9.2 中给出，见图 9.2。

构造方法 9.2

- 输入：安全参数1^n
- 协议：

① Alice 执行$\mathcal{G}(1^n)$来获得$(\mathbb{G}, q, g)$。

② Alice 从$\mathbb{Z}_q$均匀随机选择x（$x \leftarrow \mathbb{Z}_q$），计算出$h_1 := g^x$。

③ Alice 将$(\mathbb{G}, q, g, h_1)$发送给 Bob。

④ Bob 接受到$(\mathbb{G}, q, g, h_1)$。从$\mathbb{Z}_q$中均匀随机地选择y（$y \leftarrow \mathbb{Z}_q$），并且计算出$h_2 := g^y$。Bob 将$h_2$发送给 Alice，并且输出密钥$k_B := h_1^y$。

⑤ Alice 接受到h_2，并且输出密钥$k_A := h_2^x$。

Diffie-Hellman 密钥交换协议

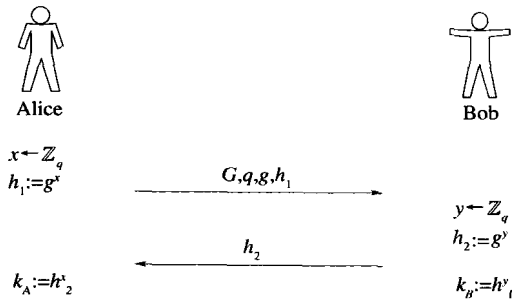


图 9.2　Diffie-Hellman 密钥交换协议

在描述中，我们假设 Alice 产生$(\mathbb{G}, q, g)$，并且在第一个消息中把这些发给 Bob。在特定的情况下，如当 Alice 和 Bob 是同一个公司的员工，或者他们共同信任某个认证中心，可能通信方不需要事先约定好$(\mathbb{G}, q, g)$，Alice 只需要发送h_1，而且 Bob 不必等待接收 Alice 的消息以计算和发送h_2。

不难看出这个协议的正确性，Bob 计算出密钥：

$$k_B := h_1^y = (g^x)^y = g^{xy}$$

Alice 计算出密钥：

$$k_A := h_2^x = (g^y)^x = g^{xy}$$

因此$k_A = k_B$。（善于观察的读者会注意到共享的密钥是群中的元素，不是比特串。将在后面讨论这一点。）现在重点放在该协议的安全性上。

Diffie-Hellman 没有证明他们协议的安全性；的确，相关的概念（定义框架以及形式化精确的假设）还没有准备好。下面说明要保证协议的安全性需要哪些假设。Diffie-Hellman 首先发现，对于该协议的安全性要求至少是与$\mathcal{G}$相关离散对数问题是困难的。否则，得到副本的敌手可以计算出某一通信方的保密值，既而轻而易举地计算出密钥，即给定g和h_1，假设离散对数问题是容易的，攻击者可以计算出$x := \log_g h_1$（这就是 Alice 的保密值），然后像 Alice 那样计算出$k_A := h_2^x$。因此离散对数问题的困难性对该协议的安全性而言是必要的，然而这并不是充分的，因为即使没有明确地找到x或者y，也可能通过其他的方式计算出密钥$k_A = k_B$。CDH 假设，只能保证g^{xy}很难作为一个整体进行计算，也不是充分的。定义 9.1 中

所要求的是：给定敌手g、g^x和g^y，g^{xy}与随机数是不可区分的。这正是 7.3 节中介绍的 DDH 假设。

下面将说明，协议安全性的证明几乎可从假设立刻得到。这不足为奇，因为 Diffie-Hellman 假设，在 Diffie 和 Hellman 发表论文之后，是 Diffie-Hellman 密钥交换协议的安全基础。考虑到这一点，应该先想想是否能从定义和安全证明中收获什么。讲到这里，但愿你已经相信答案是肯定的：通过精确定义安全的密钥交换迫使我们去思考哪些安全属性是需要的（因而去研究 Diffie-Hellman 密钥交换达到的安全属性）；通过详细说明一个精确的假设，可以研究这种假设，并独立于任何特殊的应用；最后，一旦开始确信 CDH 假设是正确的，就可以构造其他依赖于这个假设的协议。

为了完整性，现在证明了基于 CDH 假设的 Diffie-Hellman 密钥交换协议的安全性。实际上，对于定义 9.1 来说，准备去“欺骗”而不是证明它的安全性。在构造方案 9.2 中描述的协议放在这个定义里是不安全的。问题的关键在于定义 9.1 中要求通信方输出的密钥和一个随机的字符串是不可区分的，然而只能证明构造方案 9.2 中，通信方的输出和$\mathbb{G}$中的随机元素是不可区分的。（通常，群中一个随机元素和一个随机字符串看起来是很不一样的。）如果该协议用于实际应用中，应该慎重看待这个差异，毕竟，群元素作为密码学中的密钥不是非常有用，下面简单讨论一个标准的方式。令$\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$为一个改进的实验，其中如果$b=0$，一个均匀随机地从$\mathbb{G}$中取一个值$\hat{k}$给攻击者，而不是将一个随机串给攻击者。

定理 9.3 如果与$\mathcal{G}$相关判定性 Diffie-Hellman 问题是困难的，则 Diffie-Hellman 密钥交换协议Π在窃听者存在情况下是安全的（根据修改的实验$\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$）。

证明 证明所依据的直觉已经在上面给出，下面给出详细说明。取$\mathcal{A}$作为一个 PPT 攻击者。由于$\Pr[b=0]=\Pr[b=1]=1/2$，有

$$\begin{aligned}&\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]\\&=\tfrac{1}{2}\cdot\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1|b=1]+\tfrac{1}{2}\cdot\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1|b=0]\end{aligned}\tag{9.1}$$

在实验$\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中敌手收到$(\mathbb{G},q,g,h_1,h_2,\hat{k})$，其中$(\mathbb{G},q,g,h_1,h_2)$表示执行协议的副本，$\hat{k}$要么是通信方计算出来的真实密钥$g^{xy}$（如果$b=1$），要么是一个随机的群元素（如果$b=0$）。区分这两种情况和解决判定性 Diffie-Hellman 问题是等价的。即使用式（9.1）：

$$\begin{aligned}&\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]\\&=\tfrac{1}{2}\cdot\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1|b=1]+\tfrac{1}{2}\cdot\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1|b=0]\\&=\tfrac{1}{2}\cdot\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^{xy})=1]+\tfrac{1}{2}\cdot\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^z)=0]\\&=\tfrac{1}{2}\cdot\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^{xy})=1]+\tfrac{1}{2}\cdot(1-\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^z)=1])\\&=\tfrac{1}{2}+\tfrac{1}{2}\cdot(\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^{xy})=1]-\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^z)=1])\\&\leqslant\tfrac{1}{2}+\tfrac{1}{2}\cdot|\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^{xy})=1]-\Pr[\mathcal{A}(\mathbb{G},g,q,g^x,g^y,g^z)=1]|\end{aligned}$$

其中最后三行的概率由$\mathcal{G}(1^n)$从$(\mathbb{G},q,g)$上输出，$x,y,z\leftarrow\mathbb{Z}_q$。（注意，由于$g$是生成元，当$z$在$\mathbb{Z}_q$均匀分布时，$g^z$是$\mathbb{G}$上均匀分布的一个元素。）如果判定性 Diffie-Hellman 假设相对于$\mathcal{G}$是

困难的，这意味着存在一个可忽略函数negl满足

$$|\Pr[\mathcal{A}(\mathbb{G}, g, q, g^x, g^{xy}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leqslant \text{negl}(n)$$

因此，

$$\Pr[\hat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leqslant \frac{1}{2} + \frac{1}{2} \cdot \text{negl}(n)$$

证毕。

¥

随机群元素与随机串的比较。前面定理表明由 Alice 和 Bob 输出的密钥和一个随机的群元素（对于多项式时间的窃听者）在计算上是不可区分的。但是，由于这个密钥用在随后的加密操作中，它应该和某一长度的随机串是不可区分的。这可以通过将群元素映射到串来实现，这种映射“保留了均匀性”。完成这一映射的有效方法是存在的，称为“随机数抽取器”。目前这些在计算机科学中是相对标准化的，这超出了本书的范围。

主动攻击者。目前为止，只考虑了窃听敌手的情况。虽然窃听攻击是最常见的（因为很容易执行），但是它们绝不是唯一可能的攻击。主动攻击是攻击者将自己的消息发给单个或者通信双方，任何实践中的协议必须抵御主动攻击。当考虑到主动攻击，区分伪装攻击和中间人攻击是很有用的，其中伪装攻击是一个诚实的通信方和敌手伪装的另一方执行协议，中间人攻击是两个诚实的通信方都和敌手执行协议，但是敌手在双方通信途中截获并且修改消息。

这里不定义抵御这类攻击的安全，因为这样的定义相当复杂，而且这样的安全需要通信方事先共享信息。值得注意的是：Diffie-Hellman 协议对于中间人攻击是完全不安全的。事实上，在中间人攻击中，Alice 和 Bob 通信结束时的密钥k_A和k_B是不同的，而且敌手知道这两个密钥，但是 Alice 和 Bob 都不能察觉到受到了攻击。这个攻击的详细情况留作练习。

虽然 Diffie-Hellman 协议无法抵抗中间人攻击，但这一点没有减弱该协议的重要性。Diffie-Hellman 协议是作为非对称技术（以及数论问题）用于密码学中缓和密钥分配问题的第一个实证。而且，Diffie-Hellman 协议的扩展协议可以抵抗中间人攻击，并且这些协议如今正广泛使用。

在实践中的 Diffie-Hellman 密钥交换协议。由于 Diffie-Hellman 协议的基本形式不能抵抗中间人攻击，因此通常不在实践中应用。然而，它是其他可以抵抗中间人攻击并且广泛使用的密钥交换协议的核心。其中一个基于 Diffie-Hellman 密钥交换的著名的标准化协议的例子是 IPsec。

参考文献和扩展阅读材料介绍

总的来说，仅简单地讨论了密钥分配问题和密钥管理问题。关于更多的信息，推荐读者阅读网络安全教材。我们最喜欢的是 Kaufman 等写的[87]，书中对安全密钥分配的不同协议有精彩的介绍，包括要达到什么目标，如何到达。

强烈推荐阅读 Diffie-Hellman 的[47]原文。公钥密码发展史也是很有趣的；另外，Levy 的书[95]重点在于对公钥革命的政策和历史方面介绍。

练　习

9.1 考虑下面的交互协议Π'加密消息：首先，发送者和接收者运行一个密钥交换协议Π产生共享密钥k。然后，发送者计算$c \leftarrow \mathsf{Enc}_k(m)$并发送$c$给另一方，这一方使用$k$来解密得到$m$。

（1）形式化定义：存在窃听者（见定义 3.8）的不可区分加密，适用于这种交换情形。

（2）证明如果Π在窃听者存在的情况下是安全的，而且$(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$是一个对称密钥加密方案具有加密不可区分性，那么$\Pi'$满足（1）中所给的定义。

9.2 详细描述对 Diffie-Hellman 密钥交换协议的中间人攻击，该攻击中敌手和 Alice 共享密钥k_A，和 Bob 共享密钥k_B，Alice 和 Bob 不能觉察到有何不妥。如果 Alice 和 Bob 设法检测是否存在中间人攻击的敌手，他们互相发送（加密的）问题且只有另一方知道如何回答，则将会发送什么结果。

9.3 考虑下面的密钥交换协议：

（1）Alice 随机选择$k, r \leftarrow \{0,1\}^n$，发送$s := k \oplus r$给 Bob。

（2）Bob 随机选择$t \leftarrow \{0,1\}^n$，发送$u := s \oplus t$给 Alice。

（3）Alice 计算$w := u \oplus r$，发送w给 Bob。

（4）Alice 输出k，Bob 计算$w \oplus t$。

说明 Alice 和 Bob 输出同样的密钥。分析方案的安全性（即要么证明其安全，要么指出一个实质性的攻击）。

第10章 公钥加密

10.1 公钥加密简介

正如上一章所讨论的，公钥加密的出现标志着密码学领域的一场革命。在此之前，密码学家们完全依靠共享的秘密密钥来实现秘密通信。与之相反，公钥技术使得通信双方可以在没有事先共享任何秘密信息的情况下进行通信。正如前面提到过的（在稍微不同的情况下），这样的情况相当吸引人并且不直观：两个人在一间屋子的两端，只能通过大声喊叫与对方交流，没有初始密钥，但屋子里没有人知道他们说什么。

在对称密钥加密情况下，两方同意共享一个可同时用于（任何一方）加密和解密的秘密密钥k，公钥加密在这方面是非对称的。具体来说，一方（接收方）产生一对密钥(pk, sk)，被分别称为公钥和私钥。发送方用公钥将消息加密，然后发送给接收方。接收方就用私钥将最终收到的密文进行解密。

既然目的是避免双方必须事先同意某个信息，那发送方又如何获得pk？从一个抽象的层面上来说，主要有两种情况：

第一种情况。假定接收方为Alice，发送方为Bob。如果Alice知道Bob想与她通信，她会产生一组密钥(pk, sk)（假设她前面没做这一步），然后将pk以明文形式发给Bob；Bob就能用pk去加密他的消息。需要强调：在Alice与Bob间的通信信道可能是公开的，但假设是认证的，即敌手是无法修改Alice发给Bob的公钥（特别的是敌手不能用自己产生的密钥去替换）。可以用数字签名来解决这个问题，参见12.8节所讨论的如何在没有认证的信道中发布公钥。

第二种情况。先让Alice提前产生她的密钥对(pk, sk)，独立于任何特定的发送者。（事实上，在密钥产生的时候，Alice不必知道Bob想和她进行通话，甚至也不用知道存在Bob。）然后Alice广泛地传播它的公钥pk，比方说通过她的个人网页，将公钥放置在她的名片中，公布在报纸上面，或者是放在某个公共目录下。现在任何想与Alice秘密通信的人可以查到她的公钥，然后通过上述的方法处理。注意，在所有通信中，多个发送方可以利用同样的pk与Alice通信多次。

重要的一点，公钥pk本来就是公开的，而且，在上述两种情况中都很容易被敌方获得。在第一种情况下，一个敌手窃听Alice与Bob的通信，从Bob发给Alice的第一个消息中容易获得pk；而在第二种情况下，一个敌手能自己查到Alice提供的公钥。因此，公钥加密的安全不依赖于公钥的安全性，而是依赖于私钥的安全性。所以，重要的是Alice没有将其私钥泄露给其他人，包括Bob在内。

公钥加密与对称密钥加密的比较。可能两者最明显的不同点在于：前者假设了所有密钥的完全保密性，然后后者只要求了密钥对(pk, sk)一半的保密性。虽然这看似很小的区别，其实有着很大的不同：在对称密钥加密中，通信的双方必须能够共享密钥，并且不允许第三方获得此密钥；而在公钥环境中，一方可以在不损失安全的情况下通过公共信道将公钥传给第一方。对在一个房间里互相大声喊叫的各方来说，或者更现实点，在一个完全在公共网络

（类似电话线或者互联网等）中通信的各方，公钥加密是唯一的选择。

另外一个重要的区别在于：对称密钥加密方案加密和解密时使用的密钥是同一个，而公钥加密方案使用不同的密钥。因为这个原因，公钥加密方案有时候被称作非对称的。这个非对称在公钥情况中意味着发送方和接收方不能像私钥情况下互换身份：一个给定的公钥加密方案只允许单向通信（这可以用很多方法来解释，但是关键点在于：单方发起的加密迫使发送者和接收者之间的存在区别。）另外，公钥加密方案可使多个发送者与单个接受者秘密通信，与之不同的是，依靠双方共享密钥的对称密钥加密只能让两方进行秘密通信。

总结和详细阐述前面的讨论，可见，相对于对称密钥加密来说，公钥加密有着如下的优势（在前一章中也讨论过。）：

(1) 最重要的优势是公钥加密（在某种程度上）解决了密钥分配问题，因为通信双方不需事先共享秘密密钥。公钥加密可以让双方秘密通信，甚至它们之间所有的通信都被监听。

(2) 在一个接受者与U个发送者通信的情况下（例如，一个在线商家处理多方购买者的信用卡订单的过程），接受者存储一个单独的私钥sk，比共享、存储以及管理U个不同的密钥（例如，每个发送方都有一个密钥）要方便得多。事实上，当利用公钥加密时候，潜在发送方的数量和标识不需要在密钥产生时被知道。这就有了极大的灵活性，并且对于一个在线商家来说非常重要。

公钥加密最主要的缺点是：比对称密钥加密要慢至少 2～3 个数量级[①]。例如，对一些类似于智能卡或者RFID标签等资源限制的设备来说是一个挑战。甚至当一个强大的电脑进行一些密码操作的时候，执行数百次加密操作可能会行不通（这个实际上很常见，好比一个服务器为在线商家处理信用卡交易）。我们可能会得出这样的结论：如果对称密钥加密是一种选择（换而言之，如果双方能够事先安全地共享一个密钥），它就应当被利用。

事实上，对称密钥加密在公钥情况下被用于提高（公钥）加密长消息时的效率；进一步的讨论在 10.3 节。所以，透彻理解对称密钥加密，才会对公钥加密如何在实践中实现有感触。

公钥的安全分配。到目前为止，我们隐含地假定敌手是被动的。换言之，敌手仅仅偷听接受者和发送者的通信，但不主动干扰通信。如果敌手能够篡改诚实通信方全部的通信，并且该通信方没有共享密钥，则不能实现私密性。例如，一个接受者Alice将她的公钥pk发送给Bob，但是敌方却用自己的公钥pk'替换了该公钥（与私钥sk'配对），然后尽管Bob用pk'将消息加密，敌手将容易（利用sk'）解密这个消息。一个类似的攻击是，一个敌手改变Alice存储在某些公共目录下的公钥，或者敌手在公钥从目录到Bob的传输过程中篡改了公钥。如果Alice和Bob不提前共享任何信息，而且也没有依赖某些相互信任的第三方，则Alice和Bob没办法阻止这类主动攻击，甚至不能辨别此类攻击正在发生[②]。展望一下，12.8 节将讨论如何稍稍依赖可信第三方来抵御此类攻击。

本章以及下一章对公钥加密的讨论，都简单地假设发送方有一个接收方公钥的合法拷贝（这将隐含在安全定义中。）换言之，假设密钥已经安全分配，这个假设不是因为上面提到的主动攻击不值得关注（事实上，它们代表一种严重的威胁，必须在任何利用公钥加密的实际

① 因为效率取决于实际场景和多方面的实现细节，所以很难给出一个精确的比较结论。

② 在“隔着房间呼喊”方案中，当敌手妨碍交流时，Alice 和 Bob 可以察觉。但是这仅仅是因为：敌手不能阻止从 Alice 发出的消息到达 Bob，且 Alice 和 Bob 事先共享着某些信息（例如他们的声音或查看的方式），从而准许他们认证所交流的信息。

中加以处理），而是因为存在着其他阻止这些主动攻击的方案（参见诸如 12.8 节），而且很方便的（且有用的）将安全公钥加密的研究与安全密钥分配的研究分离开来。

10.2 定　义

首先从公钥加密的语法开始。除了定义的加密密钥和解密密钥是不同的，而不是单一的密钥这个区别以外，这里的定义和定义 3.7 非常类似。

定义 10.1 公钥加密方案是如下一个概率多项式时间算法(Gen, Enc, Dec)的元组：

(1) 生成密钥算法Gen用安全参数1^n作为输入，输出一对密钥(pk, sk)。把pk称为公钥，把sk称为私钥。为了方便，假设pk和sk的各自长度至少为n，而且n可由pk和sk确定。

(2) 加密算法Enc把公钥pk和来自某个明文空间的一个消息m作为输入。输出密文c，记为$c \leftarrow \mathsf{Enc}_{pk}(m)$。

(3) 解密算法Dec把私钥sk和密文c作为输入，输出一个消息m或一个定义为失败的特殊符号$\perp$。不失一般性，假设Dec是确定的，记为$m := \mathsf{Dec}_{sk}(c)$。

满足

$$\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m))=m]$$

例外的概率是可忽略的，概率的计算来源于由$\mathsf{Gen}(1^n)$输出的(pk, sk)和Enc使用的任何随机性。

考虑到定义的语法，和对称密钥加密的一个重要的区别是：密钥生成算法Gen现在输出两个密钥，而不是一个。（而且，不能再简单地假设pk仅仅是一个n位的随机串。）公钥pk用来加密，同时私钥sk用来解密。重申之前的讨论，即假设pk广泛地被分发，任何人都可以加密给生成这个密钥的通信方的消息，但是sk必须由接收方保密来保证安全性。

注意，这里允许一个可忽略的解密错误。将要陈述的具体方案中可以有一个可忽略的错误（比如，如果需要一个素数，但以一个可忽略的概率得到一个合数）。尽管如此，从本节开始将忽略这个问题。

在公钥加密的实际应用中，希望明文是$\{0,1\}^n$或者$\{0,1\}^*$（并且，特别是要独立于公钥）。有些描述的加密方案使用的消息空间M未包含所有的定长比特串（长度可能依赖于公钥），将说明在这些情况中如何把比特串编码成M中的元素。这种编码方法以及解码方法必须高效，从而接收者可恢复加密的比特串。

例 10.2 比如一个加密方案有消息空间$\mathbb{Z}_N$，其中N是公钥中的一个n位长的整数。通过把任何一个这样的串视为严格小于N的整数，可将长度为$n-1$的字符串编码为$\mathbb{Z}_N$中的元素。这样的编码和解码都是高效的。

10.2.1 选择明文攻击的安全性

通过在公钥情形中引入和定义 3.8 中的对应部分来开始对安全问题的论述。引入这个定义的动机（连同其他相关内容）在第 3 章中已给出，这里的讨论将相对简单，而且重点是介绍对称密钥加密和公钥加密的区别。

给定一个公钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，敌手$\mathcal{A}$，进行下面的实验：

窃听不可区分实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$：

(1) 执行$\mathsf{Gen}(1^n)$来获得密钥(pk, sk)。

(2) 敌手$\mathcal{A}$被给予pk，输出一对相同长度的消息m_0, m_1（消息必须在与pk相关的明文空间中）。

(3) 随机选择$b \leftarrow \{0,1\}$，然后计算出$c \leftarrow \mathsf{Enc}_{pk}(m_b)$并且给$\mathcal{A}$。称$c$是挑战密文。

(4) $\mathcal{A}$输出一位b'。

(5) 如果$b = b'$实验输出 1，否则输出 0。

定义 10.3 如果对于所有概率多项式时间的敌手$\mathcal{A}$存在可忽略函数negl，满足

$$\Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

则公钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$在窃听者存在情况下具有不可区分加密。

定义 10.3 与定义 3.8 的主要区别是，这里的$\mathcal{A}$给予公钥pk，并且，允许$\mathcal{A}$基于这个公钥选择它的消息m_0, m_1。在定义公钥加密的安全性时这十分重要，如前所述，必须假设在公钥加密情形中窃听通信的敌手知道接收方的公钥。

这看上去是“少量”的修改，即赋予敌手$\mathcal{A}$用于加密消息的公钥pk，但有一个巨大的影响：它让$\mathcal{A}$可以自由地访问加密预言机。（加密预言机的概念在 3.5 节中已经解释。）因为敌手被给予pk，就可以加密任何想加密的消息m，即通过使用正常产生的随机源计算$\mathsf{Enc}_{pk}(m)$。（通常假设$\mathcal{A}$知道算法Enc。）因此，与定义 3.21 使用的方法类似，定义 10.3 等价于抗选择明文攻击安全。特别的，考虑到下面定义公钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$和敌手$\mathcal{A}$的实验：

CPA**不可区分实验**$\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$：

(1) 执行$\mathsf{Gen}(1^n)$来获得密钥(pk, sk)。

(2)敌手$\mathcal{A}$被给予pk，并且敌手能够访问$\mathsf{Enc}_{pk}(\cdot)$预言机，敌手输出一对相同长度的消息m_0, m_1（消息必须在与pk相关的明文空间中）。

(3) 随机选择$b \leftarrow \{0,1\}$，然后计算出$c \leftarrow \mathsf{Enc}_{pk}(m_b)$并且给$\mathcal{A}$。称$c$是挑战密文。

(4) $\mathcal{A}$继续使用$\mathsf{Enc}_{pk}(\cdot)$，输出一位b'。

(5) 如果$b = b'$实验输出 1，否则输出 0。

定义 10.4 如果对于所有概率多项式时间的敌手$\mathcal{A}$存在如下一个可忽略函数negl，满足

$$\Pr\left[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

则公钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$在选择明文攻击下具有不可区分加密。

总结上面的内容，由于$\mathcal{A}$可以自己使用pk加密消息，所以加密预言机是不必要的。因此：

命题 10.5 如果一个公钥加密方案Π在窃听者存在情况下具有不可区分加密，则Π在选择明文攻击下也具有不可区分加密。

这和对称密钥加密情形是不同的。在对称密钥加密方案中，窃听者存在情况下具有不可区分加密，在选择明文攻击时是不安全的（参见命题 3.19 和命题 3.22）。由此立即可得更多和对称密钥加密情形的区别，将在下面讨论。

完美安全的公钥加密方案的不可能性。完美安全的公钥加密可以从窃听者的视角，类似于定义 2.1 一样定义（即包括公钥）。同样地，它可以通过扩展定义 10.3 来定义，即要求所有的敌手$\mathcal{A}$满足

$$\Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1\right]=\frac{1}{2}$$

与对称密钥加密的情形相反，无论多长的密钥和多短的消息，完美安全的公钥加密是不可能的。事实上，给定公钥pk，密文c可以通过$c \leftarrow \mathsf{Enc}_{pk}(m)$计算出来，对于一个无限强大的敌手，能以概率 1 确定消息m。对其证明留作练习。

确定性公钥加密的不安全性。正如对称密钥加密中提到的，任何一个确定性的加密方案都不是CPA安全的。在公钥加密情形中有窃听者存在情况下，由于 CPA 安全和不可区分加密的等价性，可得出结论：

定理 10.6 在窃听者存在情况下，没有一种确定性的公钥加密方案具有不可区分加密。

由于定理 10.6 十分重要，因此值得进一步讨论。这个定理不仅仅是安全定义中的一个“人工制品”，它也预示着定义 10.3 太强。确定性的公钥加密方案在现实情况下容易受到攻击，因此不应使用。因为确定性的方案不仅让敌手确定是否是相同的消息发送了两次（如同在对称密钥加密情形中），也允许敌手以概率 1 的可能性恢复消息，只要可能的被加密消息的集合很小（见练习 10.2）。比如，假如一个教授加密一个学生的期末成绩。这时，一个窃听者知道学生的分数肯定是$\{A,B,C,D,F\}$中的一个。如果教授使用确定性的公钥加密方案，窃听者通过加密所有可能的分数，并且比较所给的密文，可以快速的确定学生的真实分数。

尽管上述的定理看起来似乎很简单，但是有很长一段时间，许多现实世界的系统都是用确定性的公钥加密来设计的。当引入公钥加密时，概率性加密的重要性并没有完全被认识到。在定义 10.3 中提到的，并在定理 10.6 中证明的，由Goldwasser和Micali提出的开创性工作，是密码学领域中的一个转折点。通过形式化的定义来刻画直觉，第一次用正确的方法看待问题，即使现在回忆起来看似简单，其重要性应不被低估。

10.2.2 多重加密

正如第 3 章，理解使用相同密钥（这里是相同的公钥）加密多重消息的影响是很重要的。通过扩展窃听安全性（定义 10.3），来定义在这种情形下的安全性。虽然从前几节的讨论中很清楚地了解到：该定义等价于选择明文攻击条件下的定义。于是证明了：在窃听者存在的情况下，任何不可区分加密用来加密多重消息时都是安全的。这意味着，可以通过基于前一个定义来证明安全性，这种方法更简单，同时，这种证明了安全性的方案也满足后一种定义，这种定义更加精确地建模了敌手的攻击。

类似的结果在对称密钥加密的情形中已经陈述，但是没有证明，参见命题 3.22。它陈述了选择明文攻击下的安全性，但是如前所述在这个情形下等价于窃听情况下的安全性。

考虑下面这个实验，定义公钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$和敌手$\mathcal{A}$：

多重加密窃听实验$\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$：

(1) 执行$\mathsf{Gen}(1^n)$来获得密钥(pk,sk)。

(2) 把pk给敌手$\mathcal{A}$，敌手输出一对消息向量$\vec{M}_0=(m_0^1,\cdots,m_0^t)$和$\vec{M}_1=(m_1^1,\cdots,m_1^t)$，满足对所有的$i$有$|m_0^i|=|m_1^i|$。（所有的消息必须在与$pk$相关的明文空间中。）

(3) 随机选择$b\leftarrow\{0,1\}$，对于所有的i，然后计算出$c^i\leftarrow\mathsf{Enc}_{pk}(m_b^i)$，并且把密文向量$\vec{C}=(c^1,\cdots,c^t)$给$\mathcal{A}$。敌手$\mathcal{A}$输出一位$b'$。

(4) 如果$b=b'$，实验输出 1，否则输出 0。

定义 10.7 如果对于所有概率多项式时间的敌手$\mathcal{A}$存在一个可忽略函数negl，满足

$$\Pr\left[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right]\leqslant\frac{1}{2}+\mathsf{negl}(n)$$

则公钥加密方案$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$在窃听者存在的情况下具有不可区分的多重加密。

对上述定义的证明等价于定义 10.3 的证明，尽管有一点技巧性，但是不难。因此在给出正式的证明前，提供一些直观感受。讨论中，令$\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$中$t=2$。（一般的，$t$可取任意值，甚至可以依赖于$n$或$pk$。）给定任意的一个PPT敌手$\mathcal{A}$和一个窃听者存在情况下不可区分加密的公钥加密方案$\Pi$，考虑 t=2 的实验$\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$。在这个实验中，如果$b=0$，将$\vec{C}=(\mathsf{Enc}_{pk}(m_0^1),\mathsf{Enc}_{pk}(m_0^2))$给敌手；如果$b=1$，将$\vec{C}=(\mathsf{Enc}_{pk}(m_1^1),\mathsf{Enc}_{pk}(m_1^2))$给敌手。

存在一个可忽略函数negl，满足

$$\Pr\left[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right]\leqslant\frac{1}{2}+\mathsf{negl}(n)$$

使用事实

$$\begin{aligned}\Pr\left[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right]&=\frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1),\mathsf{Enc}_{pk}(m_0^2))=0]\\&\quad+\frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_1^1),\mathsf{Enc}_{pk}(m_1^2))=1]\end{aligned}$$

这里等式来自于b有两个可能的取值的条件概率。

考虑如果把$(\mathsf{Enc}_{pk}(m_0^1),\mathsf{Enc}_{pk}(m_1^2))$给$\mathcal{A}$；即一对密文中第一个来自$\overrightarrow{M_0}$的密文，第二个来自$\overrightarrow{M_1}$的密文。尽管这并不对应任何实验$\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$中可能发生的情况，但是当给出两个以这种方式计算的密文时，$\mathcal{A}$输出 0，其概率仍然是有定义的。于是声称：

声称 10.8 存在一个可忽略函数negl，满足

$$\begin{aligned}\frac{1}{2}+\mathsf{negl}(n)\geqslant\frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1),\mathsf{Enc}_{pk}(m_0^2))=0]\\+\frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1),\mathsf{Enc}_{pk}(m_1^2))=1]\end{aligned}$$

证明 为了证明，建立下面这个PPT敌手$\mathcal{A}'$，其窃听了加密的“单个”消息。

敌手$\mathcal{A}'$：

(1) $\mathcal{A}'$得到pk，执行$\mathcal{A}(pk)$来获得两个消息向量$\vec{M}_0=(m_0^1,m_0^2)$和$\vec{M}_1=(m_1^1,m_1^2)$。

(2) $\mathcal{A}'$输出一对消息(m_0^2, m_1^2)，把密文c^2给$\mathcal{A}'$。

(3) $\mathcal{A}'$计算出$c^1 \leftarrow \mathsf{Enc}_{pk}(m_0^1)$，执行$\mathcal{A}(c^1, c^2)$，输出$\mathcal{A}$的输出，即比特位$b'$。

如果观察实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)$，可发现当$b=0$时，敌手$\mathcal{A}'$得到$\mathsf{Enc}_{pk}(m_0^2)$。并且

$$\Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(m_0^2))=0] = \Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_0^2))=0]。$$

相反，在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)$中当$b=1$时，敌手$\mathcal{A}'$得到的是$\mathsf{Enc}_{pk}(m_1^2)$。并且

$$\Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(m_1^2))=1] = \Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_1^2))=1]。$$

由于Π的安全性（单消息不可区分性），存在一个可忽略函数negl，满足

$$\begin{aligned}
\frac{1}{2}+\mathsf{negl}(n) &\geqslant \Pr\left[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)=1\right] \\
&= \frac{1}{2}\cdot\Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(m_0^2))=0] + \frac{1}{2}\cdot\Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(m_1^2))=1] \\
&= \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_0^2))=0] + \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_1^2))=1]
\end{aligned}$$

证毕。 ■

类似的，可以证明：

声称 10.9 存在一个可忽略函数negl，满足

$$\begin{aligned}
\frac{1}{2}+\mathsf{negl}(n) \geqslant & \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_1^2))=0] \\
& + \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_1^1), \mathsf{Enc}_{pk}(m_1^2))=1]
\end{aligned}$$

将上面两个声称得到的不等式求和，并且使用两个可忽略函数的和仍是可忽略的这个事实，可得，存在一个可忽略函数negl，满足

$$\begin{aligned}
1+\mathsf{negl}(n) \geqslant & \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_0^2))=0] \\
& + \frac{1}{2}\cdot\Big(\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_1^2))=1] \\
& \quad + \Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_1^2))=0]\Big) \\
& + \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_1^1), \mathsf{Enc}_{pk}(m_1^2))=1] \\
= & \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_0^1), \mathsf{Enc}_{pk}(m_0^2))=0] + \frac{1}{2} \\
& + \frac{1}{2}\cdot\Pr[\mathcal{A}(\mathsf{Enc}_{pk}(m_1^1), \mathsf{Enc}_{pk}(m_1^2))=1] \\
= & \frac{1}{2} + \Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1]
\end{aligned}$$

即

$$\Pr\left[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right] \leqslant \frac{1}{2}+\mathsf{negl}(n)$$

得证。完成了$t=2$情况下的证明。

在一般情况下的主要复杂之处是t不再是一个固定的值，而是可能依赖于n。虽然对首次阅读的读者来说，可能想跳过证明，但是后面的对定理的形式化证明可作为一个很好的“混合参数”的例子，这种混合参数广泛应用于分析更复杂的密码方案。

定理 10.10 如果在窃听者存在情况下，公钥加密方案Π是不可区分加密，则Π在窃听者存在情况下是不可区分多重加密。

证明 固定任意一个PPT敌手$\mathcal{A}$，考虑实验$\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$。令$t=t(n)$为由$\mathcal{A}$输出的两个向量中消息个数的上界，不失一般性，假设$\mathcal{A}$输出的向量总是刚好这么多消息。（对不失一般性的证明留作练习。）对于一个给定的公钥pk和由$\mathcal{A}$输出的向量$\vec{M}_0(m_0,\cdots,m_0^t)$和$\vec{M}_1(m_1,\cdots,m_1^t)$，定义

$$\vec{C}^{(i)} \stackrel{\mathsf{def}}{=} (\underbrace{\mathsf{Enc}_{pk}(m_0^1),\ \cdots,\mathsf{Enc}(m_0^i)}_{i\ \text{terms}},\underbrace{\mathsf{Enc}_{pk}(m_1^{i+1}),\ \cdots,\mathsf{Enc}(m_1^t)}_{t-i\ \text{terms}})$$

其中$0\leqslant i\leqslant t$。要强调的是上述的加密总是使用独立的随机因素，并且它实际上代表着包含t个密文的向量的一种分布。利用这一表示法，有

$$\Pr\left[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1\right]=\frac{1}{2}\cdot\Pr[\mathcal{A}(\vec{C}^{(t)})=0]+\frac{1}{2}\cdot\Pr[\mathcal{A}(\vec{C}^{(0)})=1] \tag{10.1}$$

考虑下面窃听单个消息的密文的PPT敌手$\mathcal{A}'$。

敌手$\mathcal{A}'$：

(1) 把pk给$\mathcal{A}'$，执行$\mathcal{A}(pk)$来获得两个消息向量$\vec{M}_0(m_0^1,\cdots,m_0^t)$和$\vec{M}_1(m_1^1,\cdots,m_1^t)$，有$t=t(n)$。

(2) $\mathcal{A}'$选择一个随机索引$i\leftarrow\{1,\cdots,t\}$，并且输出一对消息$m_0^i,m_1^i$，并且把密文$c^i$给$\mathcal{A}'$。

(3) 对于$j<i$，$\mathcal{A}'$计算出$c^j\leftarrow\mathsf{Enc}_{pk}(m_0^j)$。对于$j>i$，$\mathcal{A}'$计算出$c^j\leftarrow\mathsf{Enc}_{pk}(m_1^j)$。然后$\mathcal{A}'$执行$\mathcal{A}(c^1,\cdots,c^t)$，输出$\mathcal{A}$输出的比特$b'$。

从直觉上，视$\mathcal{A}'$为“猜”出索引$i\in\{1,\cdots,t\}$，对这个i，$\mathcal{A}$能区分出$\vec{C}^{(i)}$和$\vec{C}^{(i-1)}$。这是因为给$\mathcal{A}$的$(c^1,\cdots,c^t)$中的第i个元素是c^i，是$\mathcal{A}'$的挑战密文。因而，如果$\mathcal{A}$能区分$\vec{C}^{(i)}$和$\vec{C}^{(i-1)}$，就意味着能区分是m_0^i的加密还是m_1^i的加密（因为这是$\vec{C}^{(i)}$和$\vec{C}^{(i-1)}$的唯一区别）。这是下面证明的核心思想。

在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)$中，当$b=0$且$i=i^*$，把$\mathsf{Enc}_{pk}(m_0^{i^*})$给敌手$\mathcal{A}'$，$\mathcal{A}$的输入是根据$\vec{C}^{(i^*)}$分布的，运行$\mathcal{A}$，这样

$$\begin{aligned}\Pr[\mathcal{A}'\ \text{outputs}\ 0\mid b=0]&=\sum_{i^*=1}^{t}\Pr[\mathcal{A}'\ \text{outputs}\ 0\mid b=0\wedge i=i^*]\cdot\Pr[i=i^*]\\&=\sum_{i^*=1}^{t}\frac{1}{t}\cdot\Pr[\mathcal{A}(\vec{C}^{(i^*)})=0]\end{aligned}$$

另外，在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)$中，当$b=1$且$i=i^*$，把$\mathsf{Enc}_{pk}(m_1^{i^*})$给敌手$\mathcal{A}'$，$\mathcal{A}$的输入根据$\vec{C}^{(i^*-1)}$分布的，运行$\mathcal{A}$，这样

$$\begin{aligned}\Pr[\mathcal{A}' \text{ outputs } 1 \mid b=1] &= \sum_{i^*=1}^{t} \Pr[\mathcal{A}' \text{ outputs } 1 \mid b=1 \wedge i=i^*] \cdot \Pr[i=i^*] \\ &= \sum_{i^*=1}^{t} \frac{1}{t} \cdot \Pr[\mathcal{A}(\vec{C}^{(i^*-1)})=1] \\ &= \sum_{i^*=0}^{t-1} \frac{1}{t} \cdot \Pr[\mathcal{A}(\vec{C}^{(i^*)})=1]\end{aligned}$$

通过假设，Π在有窃听者存在的情况下具有不可区分加密，因此存在一个可忽略函数negl，满足

$$\begin{aligned}\frac{1}{2}+\mathsf{negl}(n) &\geqslant \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi}(n)=1] \\ &= \frac{1}{2}\Pr[\mathcal{A}' \text{ outputs } 0 \mid b=0] + \frac{1}{2}\Pr[\mathcal{A}' \text{ outputs } 1 \mid b=1] \\ &= \sum_{i^*=1}^{t} \frac{1}{2t} \cdot \Pr[\mathcal{A}(\vec{C}^{(i^*)})=0] + \sum_{i^*=0}^{t-1} \frac{1}{2t} \cdot \Pr[\mathcal{A}(\vec{C}^{(i^*)})=1] \\ &= \frac{1}{2t} \cdot \sum_{i^*=1}^{t-1} \Big(\Pr[\mathcal{A}(\vec{C}^{(i^*)})=0] + \Pr[\mathcal{A}(\vec{C}^{(i^*)})=1] \Big) \\ &\quad + \frac{1}{2t} \cdot \Big(\Pr[\mathcal{A}(\vec{C}^{(t)})=0] + \Pr[\mathcal{A}(\vec{C}^{(0)})=1] \Big) \\ &= \frac{t-1}{2t} + \frac{1}{t} \cdot \Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A}',\Pi}(n)=1]\end{aligned}$$

其中最后一步用到了式（10.1）。简单的代数知识表明

$$\frac{1}{2}+t(n)\cdot\mathsf{negl}(n) \geqslant \Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)=1]$$

因为t是多项式，函数$t(n)\cdot\mathsf{negl}(n)$也是可忽略的。由于$\mathcal{A}$是一个任意的PPT敌手，在有窃听者存在情况下$\Pi$具有不可区分的多重加密，得证。

■

加密任意长度的消息。上述结论的直接结果是：给定任意一个输入为定长消息的公钥加密方案，它是窃听者存在情况下的不可区分加密，那么可以得到输入为任意长度消息的公钥加密方案，满足同样的安全性。举一个极端的例子，原始的方案加密 1 比特的消息。例如$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$是一个加密方案，其中明文空间是$\{0,1\}$。可以构造一个新的方案$\Pi'=(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$，且明文空间是$\{0,1\}^*$。定义$\mathsf{Enc}'$如下：

$$\mathsf{Enc}'_{pk}(m)=\mathsf{Enc}_{pk}(m_1),\cdots,\mathsf{Enc}_{pk}(m_t) \tag{10.2}$$

其中$m = m_1 \cdots m_t$。解密算法Dec′也作类似修改，有：

命题 10.11 令Π和Π′如上所述。如果Π在窃听者存在的情况下是不可区分的加密，那么Π′也是。

当Π在窃听者存在情况下是不可区分的加密时（而且，可扩展到当Π是 CPA 安全时），命题 10.11 是正确的，但是在选择明文攻击情况下，类似的结论是不正确的。见 10.6 节和练习 10.13。

术语说明。已经说明，在公钥加密情形下，任何在有窃听者存在情况下是不可区分加密的方案也是 CPA 安全的。这意味着一旦证明了某个方案对于前一个（相对较弱）的定义是安全的，那么将会直接获得关于后一个（更加现实）的定义的安全性。因此在讨论和定理陈述中，将倾向于“CPA 安全”，但是在证明中，大多使用这一定义：窃听者存在情况下的不可区分加密。

10.3 混 合 加 密

命题 10.11 表明：任何 1 比特消息的 CPA 安全的公钥加密方案可用来获得对于任意消息长度的 CPA 安全的加密方案。使用这种方法加密t比特的消息需要调用原始加密方案t次，意味着计算和密文长度都在原有基础上增加了t的倍数。（当然，原始方案可以加密长度为n的字符串，则只需要$\lceil t/n \rceil$次调用。然而，对于长消息而言，这仍然是不合理的。）

对于充分长的消息，通过联合使用对称密钥加密和公钥加密，明显可以做得更好。因为对称密钥加密比公钥加密明显更高效，所以这样提高了效率。这种组合叫做“混合加密”，广泛应用在实际中。其基本思想是把加密分成两步。为了加密一个消息m，做如下步骤（见图 10.1）：

(1) 发送方首先选择一个随机的对称密钥k，然后用接收方的公钥加密k，密文为c_1。接收方将能通过解密c_1恢复出k，然而对于窃听者来说k是不可知的（通过公钥加密方案的安全性），因此，这样就可以在发送方和接收方之间建立一个共享的密钥。

(2) 然后发送方使用对称密钥加密方案(Gen′, Enc′, Dec′)和刚刚共享的密钥k加密消息m。这个加密的结果叫密文c_2，可以被接受者用k解密。

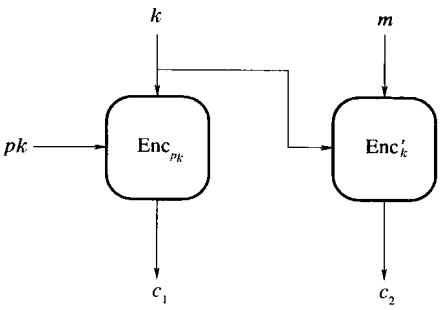


图 10.1 混合加密

以上两步可以通过让发送方发送密文$\langle c_1, c_2\rangle$来“一次性完成”。这里强调：尽管把对称密钥加密作为一个组成部分来使用，但是以上所构成的公钥加密方案实质上发送和接收双方事先没有共享任何秘密密钥。

构造方法 10.12 给出了混合加密方案Π^{hy}的一个正式的描述，它是基于任何公钥加密方案Π和对称密钥加密方案Π'（前面曾经提到假设Π'的密钥产生算法Gen'输出一个长度为n的均匀分布的密钥）。构造方法假设Π明文空间中包含$\{0,1\}^n$，因此秘密密钥$k \in \{0,1\}^n$可以被加密。Π^{hy}的明文空间和Π'的明文空间是相同的，为了简洁，设它们为$\{0,1\}^*$。

构造方法 10.12

令$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$为一个公钥加密方案，$\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$为一个对称密钥加密方案，构造一个公钥加密方案$\Pi^{\mathsf{hy}} = (\mathsf{Gen}^{\mathsf{hy}}, \mathsf{Enc}^{\mathsf{hy}}, \mathsf{Dec}^{\mathsf{hy}})$如下：

- $\mathsf{Gen}^{\mathsf{hy}}$。输入$1^n$执行$\mathsf{Gen}(1^n)$，并且输出公钥和私钥$(pk, sk)$。
- $\mathsf{Enc}^{\mathsf{hy}}$。输入公钥$pk$和消息$m \in \{0,1\}^*$，执行如下操作：
  （1）随机选择$k \leftarrow \{0,1\}^n$（如前所述，n可被pk确定）。
  （2）计算出$c_1 \leftarrow \mathsf{Enc}_{pk}(k)$和$c_2 \leftarrow \mathsf{Enc}'_k(m)$。
  （3）输出密文$\langle c_1, c_2\rangle$。
- $\mathsf{Dec}^{\mathsf{hy}}$。输入私钥$sk$和密文$\langle c_1, c_2\rangle$，执行以下：

（1）计算$k := \mathsf{Dec}_{sk}(c_1)$。
（2）输出消息$m := \mathsf{Dec}'_k(c_2)$。

混合加密

相对于Enc的方法像式（10.2）那样一比特一比特或一个分组一分组地加密，混合加密的效率如何？首先观察到混合加密当$|m| > n$时有性能改进；否则，不用Enc加密k，只需加密m本身。当$|m| \gg n$，假设Enc′比Enc更加高效（就加密每个比特而言），混合加密有实质上的效率提高。详细地说，对于n的某个固定值，令α表示使用Enc加密n比特密钥的代价，β表示使用Enc′加密（每比特明文）的代价，则使用Π^{hy}加密t比特明文的单个比特的代价为

$$\frac{\alpha + \beta \cdot t}{t} = \frac{\alpha}{t} + \beta \tag{10.3}$$

其中当t逐渐变大时，该值将接近于β。对于非常长的消息，公钥加密方案Π^{hy}加密单比特产生的代价和对称密钥加密方案Π'加密单比特产生的代价是一样的，因此，混合加密能以对称密钥加密的高效率获得公钥加密的功能，至少对足够长的消息是这样。

类似方法可计算混合加密对密文长度的影响。对一个固定值n，令l表示用Enc加密n比特密钥的密文长度，假设用Enc′加密消息m的密文长度为$n + |m|$（使用 3.6.4 节中的某个加密模式便可得到该值；实际上密文长度甚至可能是$|m|$，因为Π'不必具有 CPA 安全），则在Π^{hy}方案中密文的总长度为

$$l + n + |m| \tag{10.4}$$

当使用式（10.2）的方法加密n比特的消息分组，则密文长度为$l \cdot \lceil |m|/n \rceil$，因此上述方法对足够长的$m$是一个提高。

可以使用一些粗略估算来感受一下上述结论在实践中的意义。（这些数据只是想让读者对改进有一个感觉，然而实际的数据将依赖于各种参数。）（对称）密钥k的典型长度n为$n \approx 100$。此外公钥加密方案在单次调用中可能加密多达 1000 比特，生成的密文长度至少为 2 倍长。（更精确地说，明文长度为t的密文长度为$2000 \times \lceil t/1000 \rceil \approx 2t$。）像前面一样令$\alpha$表示公钥加密 100 比特密钥的开销，那么按式（10.2）的方法计算出，将花费$\approx \alpha \times 10^6/10^3 = 10^3\alpha$的代价加密 1 兆比特（$= 10^6$比特）的消息，而且密文长度将是 2 兆比特。将其与混合加密的效率相比，像前面一样令β表示对称密钥加密中加密单个比特的计算代价，一个合理的估算是$\beta \approx \alpha/10^6$。使用式（10.3），知道混合加密 1 兆比特的消息的计算代价将为

$$10^6 \times \left(\frac{\alpha}{10^6} + \frac{\alpha}{10^6} \right) = 2\alpha$$

并且密文的长度将略长于 1 兆比特。从而在这种情况下混合加密提高计算效率达 500 倍，并且密文的长度改善 2 倍。

下面剩下的是分析Π^{hy}的安全性。随后的定理说明：若窃听者存在情况下，原始公钥加密方案Π具有 CPA 安全，且对称密钥加密方案Π'具有不可区分加密，则合成的方案Π^{hy}具有 CPA 安全。注意，Π'满足不可区分加密这个较弱的安全性已经足够让混合加密方案Π^{hy}满足 CPA 安全。前面曾提到，对称密钥加密情形下不可区分加密不意味着 CPA 安全。直觉上的原因是在加密过程中，每加密一个新消息时，使用的密钥k是重新选择的，并且是完全随机的。由于每个密钥k只使用一次，在Π'中的单次加密中的不可区分性是混合加密方案Π^{hy}的安全性的充分条件。（从而混合加密是一种可使用流密码的应用，但是仍达到强安全保证。）

在正式证明Π^{hy}的安全性之前，强调一下总体上的直觉。令“$X \stackrel{c}{\equiv} Y$”表示没有多项式时间的敌手可以区分 X 和 Y。（这个概念在 6.8 节中有正式的介绍，虽然这里并不依赖该节中的内容。）比如，Π是 CPA 安全的，这意味着对于任何一对由一个PPT敌手输出的消息m_0, m_1，有

$$\left(pk, \mathsf{Enc}_{pk}(m_0) \right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(m_1) \right)$$

其中pk是由$\mathsf{Gen}(1^n)$产生的。这就是说，即使给出pk，m_0的加密和m_1的加密也是（在多项式时间内）不可区分的。类似地，在窃听者存在情况下Π'有不可区分的加密，这意味着任意由$\mathcal{A}$输出的m_0, m_1，有

$$\mathsf{Enc}'_k(m_0) \stackrel{c}{\equiv} \mathsf{Enc}'_k(m_1)$$

其中k是均匀随机选择的。现在，为了证明Π^{hy}的 CPA 安全性，需要说明

$$\left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0) \right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1) \right) \tag{10.5}$$

m_0, m_1是由PPT敌手$\mathcal{A}$输出的，其中pk是由$\mathsf{Gen}(1^n)$产生的，并且密钥k是在$\{0,1\}^n$中随机选择的。（式(10.5)足够证实Π^{hy}在窃听者存在情况下是不可区分加密，通过定理 10.10 证明Π^{hy}是 CPA 安全的。）

证明的过程有三步，如图 10.2 所示。首先根据Π的 CPA 安全性证明

$$\left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0) \right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0) \right) \tag{10.6}$$

确实，等式左右两边的唯一区别是加密k或加密相同长度的全 0 串，在这两种情况中均使用加密方案Π，但是Π的安全性意味着PPT敌手不会觉察到这一变化。并且，即使敌手知道k也是如此，k加密m_0的事实不会引入任何复杂性。（相比而言，如果基于Π'的安全性来证明$\left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)\right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)\right)$将陷入困境，因为只有在敌手没有$k$的信息的情况下，$\mathsf{Enc}'_k(m_0)$和$\mathsf{Enc}'_k(m_1)$的不可区分性才存在。然而，如果使用$\Pi$来加密$k$，则不再成立。当然，直觉上$\mathsf{Enc}_{pk}(k)$没有泄露$k$，但是这正是需要证明的。）

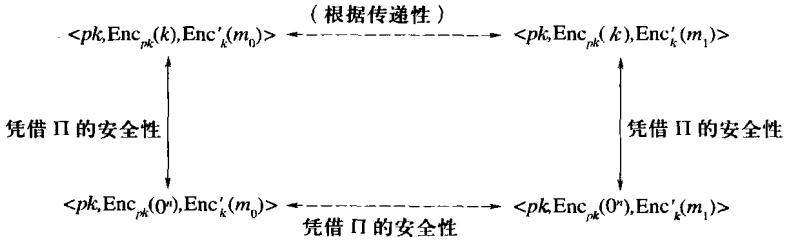


图 10.2　证明定理 10.13 的高层结构（箭头表示不可区分性）

下面基于在窃听者存在情况下Π'具有不可区分加密，证明

$$\left(pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)\right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)\right) \tag{10.7}$$

这里等式两边的不同是加密m_0和m_1，在两种情况下都是使用Π'而且密钥k均是随机选择的。并且，没有任何其他的信息泄露给敌手，特别是没有任何信息是通过$\mathsf{Enc}_{pk}(0^n)$泄露的，因为这只是加密一个全 0 的字符串（而不是k本身）。

同式（10.6）的情况一样，也要基于Π的 CPA 安全来说明

$$\left(pk, \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)\right) \stackrel{c}{\equiv} \left(pk, \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)\right) \tag{10.8}$$

式（10.6）～式（10.8）意味着，通过传递性可得到式（10.5）中期望的结果。（这里没有证明传递性，它将隐含在下面给出的证明中。）

定理 10.13　如果在窃听者存在的情况下，Π是 CPA 安全的公钥加密方案，Π'是不可区分加密的对称密钥加密方案，那么构造方法 10.12 中的Π^{hy}是 CPA 安全的公钥加密方案。

证明　固定一个PPT敌手$\mathcal{A}^{\mathsf{hy}}$，并且考虑实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}^{\mathsf{hy}},\Pi^{\mathsf{hy}}}(n)$。（由通过定理 10.10，只要说明在窃听者存在的情况下Π^{hy}具有不可区分加密，则它是 CPA 安全的。）目标是证明存在一个可忽略函数negl，满足

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}^{\mathsf{hy}},\Pi^{\mathsf{hy}}}(n) = 1] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

像前面一样，要用到

$$\begin{aligned}\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}^{\mathsf{hy}},\Pi^{\mathsf{hy}}}(n) = 1] = {} & \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)) = 0] \\ & + \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)) = 1]\end{aligned} \tag{10.9}$$

注意在每种情况下，概率来源于随机生成的pk和均匀选择的k。并且，$\mathcal{A}^{\mathsf{hy}}$也拥有公钥

pk，但是为了具有更好的可读性，这一点是隐含的。

考虑如下的 PPT 敌手$\mathcal{A}_1$，窃听使用公钥加密方案Π加密的消息。

敌手$\mathcal{A}_1$：

(1) 把pk给$\mathcal{A}_1$，$\mathcal{A}_1$随机选择$k \leftarrow \{0,1\}^n$（还记得按照惯例，n可以从pk确定），输出一对消息$k, 0^n$，并且把密文c_1给$\mathcal{A}_1$。

(2) $\mathcal{A}_1$执行$\mathcal{A}^{\mathsf{hy}}(pk)$获得两个消息$m_0, m_1$。

(3) $\mathcal{A}_1$计算$c_2 \leftarrow \mathsf{Enc}'_k(m_0)$，然后执行$\mathcal{A}^{\mathsf{hy}}(c_1, c_2)$，输出由$\mathcal{A}^{\mathsf{hy}}$输出的一个比特$b'$。

实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n)$中当$b=0$时，把密文$\mathsf{Enc}_{pk}(k)$给敌手$\mathcal{A}_1$，其中$k$是由$\mathcal{A}_1$在第一步中均匀随机选择的。这意味着给$\mathcal{A}^{\mathsf{hy}}$的密文形式是$\langle c_1, c_2\rangle = \langle \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)\rangle$，其中$pk$是随机生成的，$k$是均匀选择的。因此，

$$\Pr[\mathcal{A}_1 \text{ outputs } 0 \mid b=0] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)) = 0]$$

另外，在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n)$中，当$b=1$时，把密文$\mathsf{Enc}_{pk}(0^n)$给敌手$\mathcal{A}_1$。这意味着给$\mathcal{A}^{\mathsf{hy}}$的密文是$\langle \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)\rangle$，因此

$$\Pr[\mathcal{A}_1 \text{ outputs } 1 \mid b=1] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 1]$$

通过假设在窃听者存在情况下Π有不可区分的加密，存在一个可忽略函数negl_1，满足

$$\begin{aligned}
&\frac{1}{2} + \mathsf{negl}_1(n) \geqslant \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_1,\Pi}(n) = 1] \qquad (10.10)\\
&= \frac{1}{2}\cdot \Pr[\mathcal{A}_1 \text{ outputs } 0 \mid b=0] + \frac{1}{2}\cdot \Pr[\mathcal{A}_1 \text{ outputs } 1 \mid b=1]\\
&= \frac{1}{2}\cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)) = 0]\\
&\quad + \frac{1}{2}\cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 1]
\end{aligned}$$

接着，考虑如下 PPT 敌手$\mathcal{A}'$窃听使用对称密钥加密方案Π'加密的消息。

敌手$\mathcal{A}'$：

(1) $\mathcal{A}'(1^n)$执行$\mathsf{Gen}(1^n)$产生密钥(pk, sk)。

(2) $\mathcal{A}'$执行$\mathcal{A}^{\mathsf{hy}}(pk)$来获得消息$m_0, m_1$。这些相同的消息由$\mathcal{A}'$输出，并且把密文$c_2$给$\mathcal{A}'$。

(3) $\mathcal{A}'$计算$c_1 \leftarrow \mathsf{Enc}_k(0^n)$，然后执行$\mathcal{A}^{\mathsf{hy}}(c_1, c_2)$，输出由$\mathcal{A}^{\mathsf{hy}}$输出的一个比特$b'$。

实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi'}(n)$中当$b=0$时，把密文以$\mathsf{Enc}'_k(m_0)$的形式给敌手$\mathcal{A}'$，其中$k$是随机选择的（并对$\mathcal{A}'$来说是未知的）。这意味着给$\mathcal{A}^{\mathsf{hy}}$的密文是$\langle \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)\rangle$，其中$pk$是随机生成的，$k$也是随机选择的。因此有

$$\Pr[\mathcal{A}_1 \text{ outputs } 0 \mid b=0] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 0]$$

另外，实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi'}(n)$中，当$b=1$时，密文$\mathsf{Enc}'_k(m_1)$给敌手$\mathcal{A}'$，并且$k$仍然是随机选择的。这意味着给$\mathcal{A}^{\mathsf{hy}}$的密文是$\langle \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)\rangle$，因此

$$\Pr[\mathcal{A}_1 \text{ outputs } 1 \mid b=1] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 1]$$

通过假设窃听者存在情况下Π'具有不可区分加密，存在可忽略函数negl'，满足

$$\begin{aligned}
&\frac{1}{2} + \mathsf{negl}'(n) \geqslant \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}',\Pi'}(n) = 1] \qquad (10.11)\\
&= \frac{1}{2} \cdot \Pr[\mathcal{A}_1 \text{ outputs } 0 \mid b=0] + \frac{1}{2} \cdot \Pr[\mathcal{A}_1 \text{ outputs } 1 \mid b=1]\\
&= \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 0]\\
&\quad + \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 1]
\end{aligned}$$

接下来说明完全和式（10.10）类似的一些内容（因此欢迎读者直接跳到式（10.12））。考虑如下PPT敌手$\mathcal{A}_2$，窃听使用公钥加密方案Π加密的消息。

敌手$\mathcal{A}_2$：

(1) 把pk给$\mathcal{A}_2$，$\mathcal{A}_2$随机选择$k \leftarrow \{0,1\}^n$，输出一对消息$0^n, k$（这些消息的次序相对于敌手$\mathcal{A}_1$被交换了）。把密文c_1给$\mathcal{A}_2$。

(2) $\mathcal{A}_2$执行$\mathcal{A}^{\mathsf{hy}}(pk)$来获得消息$m_0, m_1$。$\mathcal{A}_2$计算$c_2 \leftarrow \mathsf{Enc}'_k(m_1)$，然后$\mathcal{A}_2$执行$\mathcal{A}^{\mathsf{hy}}(c_1, c_2)$，输出由$\mathcal{A}^{\mathsf{hy}}$输出的一个比特$b'$。

实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi}(n)$中当$b=0$时，给$\mathcal{A}^{\mathsf{hy}}$的密文形式是$\langle c_1, c_2\rangle = \langle \mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)\rangle$，其中$pk$是随机生成的，$k$是均匀分布的。因此，

$$\Pr[\mathcal{A}_2 \text{ outputs } 0 \mid b=0] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 0]。$$

另外，实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi}(n)$中，当$b=1$时，给$\mathcal{A}^{\mathsf{hy}}$的密文形式是$\langle \mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)\rangle$，因此

$$\Pr[\mathcal{A}_2 \text{ outputs } 1 \mid b=1] = \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)) = 1]。$$

由于在窃听者存在情况下Π具有不可区分的加密，存在如下一个可忽略函数negl_2，满足

$$\begin{aligned}
&\frac{1}{2} + \mathsf{negl}_2(n) \geqslant \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A}_2,\Pi}(n) = 1] \qquad (10.12)\\
&= \frac{1}{2} \cdot \Pr[\mathcal{A}_2 \text{ outputs } 0 \mid b=0] + \frac{1}{2} \cdot \Pr[\mathcal{A}_2 \text{ outputs } 1 \mid b=1]\\
&= \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 0]\\
&\quad + \frac{1}{2} \cdot \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)) = 1]
\end{aligned}$$

最后，证明结论。对式（10.10）～式（10.12）求和，由于三个可忽略函数的和仍然是可忽略函数，故存在如下一个可忽略函数negl，满足

$$\begin{aligned}
&\frac{3}{2} + \mathsf{negl}(n) \geqslant\\
&\frac{1}{2} \cdot \Big(\Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)) = 0] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 1]\\
&\quad + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 0] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 1]\\
&\quad + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 0] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)) = 1] \Big)
\end{aligned}$$

注意

$$\Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 1] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_0)) = 0] = 1$$

由于互补事件的概率之和总是为 1。类似地

$$\Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 1] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(0^n), \mathsf{Enc}'_k(m_1)) = 0] = 1$$

因此

$$\frac{1}{2} + \mathsf{negl}(n) \geqslant \frac{1}{2} \cdot \Big(\Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_0)) = 0] + \Pr[\mathcal{A}^{\mathsf{hy}}(\mathsf{Enc}_{pk}(k), \mathsf{Enc}'_k(m_1)) = 1] \Big)$$

式（10.9）得证。

■

上述定理的证明焦点放在可加密长度（安全参数）为n的公钥加密方案上。当然，理论上构造仅仅加密单比特消息的方案是足够的（因为它们可以通过使用 10.2.2 节中介绍的方法扩展到n比特消息）。然而实际上，需要有比分别加密单比特更高效的方案。关键点在于：即使考虑到效率问题，只要能加密长度为n的消息，就能加密更长的消息。

10.4 RSA 加密

前面关于公钥加密的讨论一直是抽象的：知道了怎样使用公钥加密方案加密任意长度的消息，但是仍然没有任何具体例子。在本节中，集中介绍基于RSA假设的一类方案。

10.4.1 “教科书式 RSA”加密方案及其不安全性

令GenRSA为一个PPT算法，输入为1^n，输出为两个n比特的素数之积的模N，同时还有满足$ed = 1 \bmod \phi(N)$的e, d。（和前面一样，算法可能会以一个可忽略概率失败，但是这里忽略它。）回顾 7.2.4 节，这样一个算法很容易通过任何输出一个合数模N及其因子分解的算法GenModulus来构造：

算法 10.14

RSA密钥生成算法GenRSA

输入：安全参数1^n

输出：N, e, d如前所述

$(N, p, q) \leftarrow \mathsf{GenModulus}(1^n)$

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

选择满足$\gcd(e, \phi(N)) = 1$的e

计算$d := [e^{-1} \bmod \phi(N)]$

返回N, e, d

这里给出所谓的“教科书式RSA”加密方案，如构造方法 10.15 所述。“教科书式RSA”是确定性的，因此是不安全的，这已在 10.2.1 节中详细讨论。虽然它不安全，在这里介绍是

因为：它提供了一个便捷的例证，说明为什么RSA假设对建立公钥加密方案很有用，它也是将在 10.4.3 节中给出的安全构造方法的一个铺垫。（介绍“教科书式 RSA”加密方案同样是一个发出警告的机会。）在本章的后面部分和第 13 章，将讨论RSA假设是如何用于构造“安全的”加密方案。

构造方法 10.15

GenRSA如前面所定义，定义一个公钥加密方案如下：

- Gen：输入1^n执行GenRSA(1^n)，获得(N, e, d)。其中公钥是$\langle N, e\rangle$，私钥是$\langle N, d\rangle$。
- Enc：输入公钥$pk = \langle N, e\rangle$和消息$m \in \mathbb{Z}_N^*$，计算密文$c = [m^e \bmod N]$。

  Dec：输入私钥$sk = \langle N, d\rangle$和密文$c \in \mathbb{Z}_N^*$，计算消息$m = [c^d \bmod N]$。

“教科书式RSA”加密方案

由推论 7.22，立刻知解密总是能成功地得到明文消息。对于方案的安全性，可以认为：计算私钥的困难性和分解由GenRSA输出的模一样困难。原因是：正如 7.2.4 节中简单提到的，给定$N = pq$和满足$ed = 1 \bmod \phi(N)$的e, d，在多项式时间内可以计算出 N 的因子。注意，这个结果没有谈到是否能用其他的方式（不涉及私钥的明确计算）从密文中恢复出明文，也没有谈到加密方案本身是安全的。

尽管“教科书式RSA”对于在本章中建议的任何安全定义来说都是不安全的，但是如果RSA假设对于GenRSA（见定义 7.46）是成立的，则有可能证明这个方案的一个非常弱的安全性。换言之，如果一个消息m是均匀随机地从$\mathbb{Z}_N^*$中选择的，则没有一个PPT敌手，在给予公钥$\langle N, e\rangle$和密文$c = [m^e \bmod N]$的情况下，能完全恢复出明文消息m。这确实是一个很弱的保证：m必须随机地选取（因此，不太清楚当m表示英文文本时能得到什么结论），并且唯一能说明的是：敌手不会知道关于m的全部信息（但是可能知道m的许多部分信息）。因此对大多数应用，它不能构成一个合理的安全性概念。

RSA 的实现问题。以RSA加密在实践方面的讨论来结束本小节。这里的讨论不仅应用到教科书式RSA方案，也可应用于基于RSA假设的其它方案。

加密$\mathbb{Z}_N^*$中的二进制串元素。令$l = ||N||$。任何长度为$l-1$的二进制串m可以视为$\mathbb{Z}_N$中的元素。也可以通过使用某些无歧义的填充方案把各种长度的二进制串编码成$\mathbb{Z}_N$中的元素。

其中理论上的关注点是（编码的）消息m可能不在$\mathbb{Z}_N^*$中（比如，可能是$\gcd(m, N) \neq 1$的情况）。即使这种情况发生，解密仍然可以成功，练习 7.10 有所讨论。如果m是随机选择的，这种情况发生的可能性很小（根据命题 B.17）。而且，即使一个发送方尝试去发现不在$\mathbb{Z}_N^*$中的m，若不分解N是很难做到的，因为给定$m \in \mathbb{Z}_N \backslash \mathbb{Z}_N^*$，$\gcd(m, N)$的值是$N$的一个非平凡因子。

e的选择。对于不同的e和采用不同的选择e的方法，RSA问题的困难性都没有表现出任何差异。一个常见的选择是取$e = 3$，然后计算e次幂模N（当使用教科书式 RSA 加密时），因这个计算只需要两次乘法（参见附录 B.2.3）。如果e设为 3，则选择的p和q必须满足$p, q \neq 1 \bmod 3$使得$\gcd(e, \phi(N)) = 1$。

选择$e = 3$使得“教科书式RSA”易于受到某些攻击，下一节中将给出例子。这说明$e = 3$是一个糟糕的选择，同时更多地说明了构造方法 10.15 是不适当的。

注意，选择小的d值用来加速解密（即改变GenRSA以便先选择一个小的d，然后计算e）是

一个不好的建议。如果在一个较小范围（如$d < 2^{16}$）中选择d，则容易用蛮力搜索到d。即使选择d满足$d \approx N^{1/4}$，蛮力攻击失效了，但是可以使用其他潜在的攻击方法来从公钥中恢复d。

使用中国剩余定理。接收方持有私钥，因此可因子分解N，能利用中国剩余定理（7.1.5 节）来加速计算模N的d次根，这是解密时必须的。特别是由于$[c^d \bmod N] \leftrightarrow ([c^d \bmod p], [c^d \bmod q])$，接收方可以计算部分结果

$$m_p := c^d \bmod p = \left[c^{[d \bmod (p-1)]} \bmod p\right]$$

和

$$m_q := c^d \bmod q = \left[c^{[d \bmod (q-1)]} \bmod q\right]$$

然后结合这些，正如 7.1.5 节中所讨论的，得到$m \leftrightarrow (m_p, m_q)$。注意到$[d \bmod (p-1)]$和$[d \bmod (q-1)]$可以在密钥产生的时候计算和存储，所以每次解密时不用再次计算。假设乘方模v比特的整数需要v^3次操作，故解密RSA需要$8n^3$步（因为$||N|| = 2n$），然而使用中国剩余定理只需要$2n^3$步，或者1/4的时间。这种加速解密的优化十分重要，因为一个服务器经常同时需要执行许多解密操作。

例 10.16 若$p = 11, q = 23, e = 3$，且$N = 253, \phi(N) = 220, d = 147$。

为了使用“教科书式 RSA”加密二进制消息$m = 0111001$，公钥$pk = \langle N = 253, e = 3\rangle$，转换成十进制数$m$为 57。然后计算

$$250 := [57^3 \bmod 253]$$

为了解密，计算$57 := [250^{147} \bmod 253]$。使用中国剩余定理，接收方可以计算

$$250^{[147 \bmod 10]} \bmod 11 = 8^7 \bmod 11 = 2$$

并且

$$250^{[147 \bmod 22]} \bmod 23 = 20^{15} \bmod 23 = 11$$

事实上，$57 \leftrightarrow (2, 11)$，因此解密成功（期望的结果可以通过如 7.1.5 节中描述的表示方法$(2, 11)$恢复）。

10.4.2 对“教科书式 RSA”加密方案的攻击

为了对RSA问题有更多的感觉，也为了举例说明其他的关于“教科书式 RSA”的问题，现在描述各种各样的攻击，其中一些只用于构造方法 10.15，有的应用面更广。注意，对于任何基于RSA假设的可证明安全方案，如果这些方案合理使用的话，这些攻击将不奏效，例如在下一节中见到的一个例子。

使用小e加密短消息。 如果e很小，则使用“教科书式RSA”加密短消息时是不安全的。比如，$e = 3$，且消息m满足$m < N^{1/3}$，但是攻击者不知道m。（假设在这个情况下，m的左边填充多个 0，然后视为$\mathbb{Z}_N$中的元素。）在这个情况下，加密m不涉及任何模减运算，因为整数m^3小于N。这意味着给予密文$c = [m^3 \bmod N]$，攻击者可以通过计算整数$m := c^{1/3}$得到m，这个计算容易完成。

注意一个现实的攻击场景：如果比特串编码成$\mathbb{Z}_N^*$中的元素，则$m < N^{1/3}$对应于加密一个长度小于$||N||/3$的短消息（假设通过在左边填充多个 0 来编码消息）。考虑使用一个混合加密的情形：1024 比特RSA加密一个长度为 128 比特的私钥；在这个情况下密钥容易提取。

当使用小e时的一般攻击。[3]上述攻击说明了如果“教科书式RSA”使用小e，则短消息很容易从密文恢复出来。把这种攻击扩展到任意长度的消息，只要相同的消息被发送给多个接收方。和前面一样，令$e=3$，相同的消息发送给三个不同的通信方，分别持有公钥$pk_1=\langle N_1,3\rangle$，$pk_1=\langle N_2,3\rangle$，$pk_1=\langle N_3,3\rangle$。于是窃听者观察到$c_1=[m^3 \bmod N_1]$，$c_2=[m^3 \bmod N_2]$，$c_3=[m^3 \bmod N_3]$。假设对于所有的$i,j$有$\gcd(N_i,N_j)=1$（如果不是，至少一个模可能立刻被因子分解，且消息$m$很容易被恢复）。令$N^*=N_1N_2N_3$。由扩展的中国剩余定理可得，存在唯一的非负值$\hat{c}<N^*$，满足

$$\begin{aligned}\hat{c}&=c_1 \bmod N_1\\ \hat{c}&=c_2 \bmod N_2\\ \hat{c}&=c_3 \bmod N_3\end{aligned}$$

并且，使用同 7.1.5 节类似的方法，在给出公钥和以上密文的情况下能高效地计算出$\hat{c}$。注意$\hat{c}\equiv m^3 \bmod N^*$。但是由于$m<\min\{N_1,N_2,N_3\}$，有$m^3<N^*$。正如前面所述的攻击，这意味着在整数范围内$\hat{c}=m^3$（即没有模减法发生），并且可以通过计算$\hat{c}$的整数立方根来获得$m$。

恢复明文m的平方改进。由于“教科书式RSA”加密是确定的，于是如果消息m是从一个小的可能值的列表中选择的，则可通过简单地尝试m的每个可能值，从密文$c=[m^e \bmod N]$中确定m，10.2.1 节中曾经讨论过。如果知道$1\leqslant m<\mathcal{L}$（当把m解释成整数时），则完成攻击的时间是$\mathcal{L}$的线性关系。这样，可能有人希望“教科书式RSA”加密能使用在当$\mathcal{L}$很大时的情形中，即消息是从一个大的集合（值）中选择的。一个可能发生这种情况的场景是在混合加密的情形中（见 10.3 节）：通过公钥部分直接加密的“消息”，是长度为l的随机对称密钥，此时$\mathcal{L}=2^l$。

不幸的是，在这种情况下，有一个更聪明的（以高概率）在时间约$\sqrt{\mathcal{L}}$内恢复m的攻击方法。这是实践中一个明显的改进：如果一个 80 比特的（随机）消息被加密，则执行2^{80}步的蛮力攻击是不可行的，但执行2^{40}步的攻击相对容易。

下面算法 10.17 是对攻击的描述。在算法的描述中，假设$m<2^l$（即$\mathcal{L}=2^l$），并且攻击者知道l。α的值是一个常量，且$\frac{1}{2}<\alpha<1$。

算法 10.17
对“教科书式RSA”加密的攻击
输入：公钥$\langle N,e\rangle$；密文c；参数l
输出：$m<2^l$满足$m^e=c \bmod N$
令$T:=2^{\alpha l}$
for $r=1$ to T：
　　$x_r:=[c/r^e \bmod N]$
通过第二项将$\{(r,x_r)\}_{r=1}^{T}$排序
for $s=1$ to T：
　　if 对某些r，有$s^e \bmod N\stackrel{?}{=}x_r$
return $[r\cdot s \bmod N]$

③ 此攻击依赖于 7.1.5 节中介绍的中国剩余定理。

在这个算法中，如果某个r模N是不可逆的，则因子分解 N 容易被计算出（因为这意味着$\gcd(r, N) \neq 1$）。在算法的描述中没有另花笔墨明确写出这些。

算法的时间复杂性取决于对$2^{\alpha l}$个配对(i, x_i)排序的时间，它可以在时间$\mathcal{O}(l \cdot 2^{\alpha l})$内完成。算法描述的倒数第二行使用二分查找，检查是否存在r满足$x_r = [s^e \bmod N]$。

现在大致解释为什么攻击能以高概率恢复m。令$c = m^e \bmod N$同时$m < 2^l$。如果m是随机选择的l比特整数，可以证明有较高概率存在满足$1 < r, s \leqslant 2^{\alpha l}$的$r, s$且$m = rs$。（参见本章最后的参考文献。）无论什么时候发生了这种情况，上述算法都发现m。事实上，在这种情况中

$$c = m^e = (r \cdot s)^e = r^e \cdot s^e \bmod N$$

因此$c/r^e = s^e \bmod N$且$r, s < T$。很容易验证这个算法在该情况下能够发现r, s。如果你还没有信服，那么这个攻击将告诉你永远不要使用确定性加密，即使你是加密大的随机密钥。

公共模攻击 I。这是一个典型的误用RSA的例子。设想一个公司想给每个员工使用相同的模N。由于不希望一个员工加密的消息能被任何其他员工读到，公司分发给每个员工不同的(e_i, d_i)。换言之，第i个员工的公钥是$pk_i = \langle N, e_i \rangle$并且他们的私钥是$sk_i = \langle N, d_i \rangle$，其中对于所有的$i$有$e_i \cdot d_i = 1 \bmod \phi(N)$。

这种方法是不安全的，它允许任何员工都可以阅读给其他员工的加密消息。原因是，正如 7.2.4 节中提到的一样，给定N和满足$e_i \cdot d_i = 1 \bmod \phi(N)$的$e_i, d_i$，可以有效地计算$N$的因子分解。当然，若$N$的因子分解已知，对于任何$j$都能计算出$d_j := e_j^{-1} \bmod \phi(N)$。

公共模攻击 II。刚讨论的攻击允许任何员工可以解密发送给其他员工的消息。这就使得存在这样一种可能性，只要所有的员工相互信任，共享模N是可以的（或者，只要对外部人员保留保密性，而不是对公司内部人员保留保密性）。这里给出一个场景说明：模的共享仍然是一个不理想的主意，至少在使用“教科书式RSA”的时候是这样。

比如加密相同的消息m，并且发送公钥(N, e_1)与(N, e_2)给两个不同的（已知）员工，其中$e_1 \neq e_2$。假设$\gcd(e_1, e_2) = 1$。则窃听者看到两个密文

$$c_1 = m^{e_1} \bmod N \text{ 和 } c_2 = m^{e_2} \bmod N$$

由于$\gcd(e_1, e_2) = 1$，则根据命题 7.2，存在整数X, Y，满足$Xe_1 + Ye_2 = 1$。并且，给予公开的指数e_1, e_2，使用扩展的欧几里得算法可以高效的计算出X, Y（参见附录 B.1.2）。很容易计算出$m = [c_1^X \cdot c_2^Y \bmod N]$。这是因为

$$c_1^X \cdot c_2^Y = m^{Xe_1} m^{Ye_2} = m^{Xe_1 + Ye_2} = m^1 = m \bmod N。$$

因此共享整个密钥比共享部分密钥要好。

这个例子和先前的一些例子可以作为使用RSA（和其他一些加密方案）的一个警告：即使是较小的或者是看上去无害的改动都有可能给攻击打开一扇门。

10.4.3 填充 RSA

“教科书式RSA”加密方案的不安全性，不仅有前两节中描述的各种各样的攻击，也有它不能满足定义 10.13 的安全性，意味着需要考虑基于RSA的其它加密方法。一个简单的想法是在加密之前随机填充消息。这种方法的一个通用范例见于构造方法 10.18。这个构造方法是基于参数l来定义的，该参数决定了可加密消息的长度。

构造方法 10.18

如前面一样定义GenRSA，并且令l为一个对所有n满足$l(n) \leqslant 2n-2$的函数。定义如一个公钥加密方案如下：

- Gen：输入1^n，执行GenRSA(1^n)来获得(N,e,d)。输出公钥是$pk=\langle N,e\rangle$，私钥是$sk=\langle N,d\rangle$。
- Enc：输入公钥$pk=\langle N,e\rangle$和消息$m \in \{0,1\}^{l(n)}$，选择随机串$r \leftarrow \{0,1\}^{||N||-l(n)-1}$，并解释$r||m$为$\mathbb{Z}_N$中的元素，输出密文$c := [(r||m)^e \bmod N]$。
- Dec：输入私钥$sk=\langle N,d\rangle$和密文$c \in \mathbb{Z}_N^*$，计算消息$\hat{m} := [c^d \bmod N]$，并且输出$\hat{m}$的低$l(n)$位。

填充的RSA加密方案

根据方案的描述，明显表示解密总能成功。（当$r||m \in \mathbb{Z}_N^*$这是显而易见的，就是$r||m \notin \mathbb{Z}_N^*$时，它也是正确的。后者出现的概率是可忽略的概率。）填充RSA加密方案的安全性取决于l。如果l太大，那么$l(n)=2n-\mathcal{O}(\log n)$，则对所有随机填充的$r$值的蛮力搜索可以在时间$2^{\mathcal{O}(\log n)}=\mathrm{poly}(n)$中完成，这个方案不是 CPA 安全的。这不“仅仅”只是一个理论上的观点，因为它也预示着如果消息m是从一个小概率空间中选择的，那么窃听者可以使用蛮力搜索确定m。

对于$c<2$的某个常量当$l(n)=c \cdot n$时，可以合理地推测填充RSA是安全的，但是还没有基于第 7 章介绍的标准RSA假设的安全性证明。（然而，可以基于非标准假设证明其安全性，见练习 10.9。）

当$l(n)$很小时，将有如下证明：

定理 10.19 如果RSA问题与GenRSA相关是困难的，则构造方法 10.18，其中$l(n)=\mathcal{O}(\log n)$，在选择明文攻击下具有不可区分加密。

该定理的完整证明超出了本书的范围；然而，其中证明$l(n)=1$这步作为练习 10.10 的一部分[④]。对于一种不同的但是相关的，基于RSA的困难性构造的安全公钥加密方案，见 10.7 节。

PKCS#1v1.5。它是一个广泛使用的标准的加密方案，RSA实验室公钥加密标准(PKCS) #1 1.5 版本，利用了填充加密的原理。对于一个常见形式的公钥$pk=\langle N,e\rangle$，令k为N的比特长度，即k是一个满足$2^{8(k-1)} \leqslant N < 2^{8k}$的整数。假设要加密的消息$m$是一个 8 比特长的倍数，并且长度可以长达$k-11$字节。加密一个$D$比特的消息计算如下

$$[(00000000||00000010||r||00000000||m)^e \bmod N]$$

其中r是随机生成的字符串，$(k-D-3)$字节，这些字节均不等于 0（后一个关于r的条件使得消息在解密的时候没有歧义）。注意，m的最大允许长度能保证r的长度至少为 8 字节。

尽管至今还没有基于RSA假设的证明，PKCS #1 v1.5仍被认为是 CPA 安全的，但是一种对该方案的选择密文攻击已被证实。从而导致对标准的修订，得到一个新的方案叫做OAEP（最优非对称加密填充），它被证明是可以对抗这种攻击（基于所谓的随机预言模型，参见 13 章）。尽管旧版本由于兼容性仍然在广泛使用，但是新的实现系统倾向于这个更新的版本。在 13.2.3 节中将给出这种方案的一个高层次描述。

④ 对于已经阅读过第 6 章的读者，此处标注这个定理依靠一个事实：在 RSA 中，最低有效位是一个硬核谓词。

10.5 El Gamal 加密

El Gamal加密方案是另一个常见的加密方案，它的安全性是基于判定性Diffie-Hellman问题的困难性。在 7.3.2 节中已详细讨论了DDH问题。

下面从陈述和证明一个简单的引理开始，该引理是El Gamal加密方案的基础，它也将对第 11 章的内容有用。令$\mathbb{G}$为一个有限群，令m为$\mathbb{G}$中任意一个元素。该引理陈述了一个随机群元素g乘以m得到一个随机的群元素g'。因为g'的分布独立于m，这意味着g'不包含m的任何信息。

引理 10.20 令$\mathbb{G}$为一个有限群，并令m为$\mathbb{G}$中任意一个元素。选择一个随机的$g \leftarrow \mathbb{G}$，令$g' := m \cdot g$，所得g'的分布，与随机从$\mathbb{G}$中选择的$g' \leftarrow \mathbb{G}$，是相同的。即对于任意$\hat{g} \in \mathbb{G}$，有

$$\Pr[m \cdot g = \hat{g}] = 1/|\mathbb{G}|$$

其中概率来源于随机选择的g。

证明 令$\hat{g} \in \mathbb{G}$是任意的。则

$$\Pr[m \cdot g = \hat{g}] = \Pr[g = m^{-1} \cdot \hat{g}]$$

因为g均是随机选择的，g等于固定元素$m^{-1} \cdot \hat{g}$的概率是$1/|\mathbb{G}|$。

■

上述引理引出了一种构造完美安全的对称密钥加密方案的方法，该方法加密$\mathbb{G}$中的消息。发送方和接收方共享一个随机的元素$g \leftarrow \mathbb{G}$。然后，加密消息$m \in \mathbb{G}$，发送方计算密文$g' := m \cdot g$。接收方可以根据密文g'计算$m := g'/g$来恢复明文。这个方案的完美安全性由引理可得。事实上，已经在另一种情形下见过这种方案，即“一次一密”加密方案正是上述的方法的一个例子，它的群是定长串的集合以及群运算是逐位异或。

正如所描述的，在这个方案中，g确实是随机的，并且因为发送方和接收方提前共享g，因此解密是可能的。在公钥情形中，需要一个不同的方法来让接收者解密消息。一个关键的思路是使用“伪随机”元素g，而不是一个真正随机的g。更详细地说，g可以这样定义：接收者将能使用其私钥计算出g，然而g对于任何窃听者来说“看上去是随机的”。现在说明如何使用DDH假设来实现这个想法，并且一旦理解了定理 10.22 的证明，这个讨论将变得更加清晰。

同 7.3.2 节一样，令$\mathcal{G}$为一个多项式时间算法，输入为1^n，输出一个循环群$\mathbb{G}$，其阶为q（其中$||q|| = n$），生成元为g。（通常要求$\mathbb{G}$中的运算可以在关于n的多项式时间内计算出来，并且允许$\mathcal{G}$以可忽略概率失败。）El Gamal加密的一种情况是：$\mathbb{G}$是一个椭圆曲线群，这类群在 7.3.4 节中简短介绍过，它们的优势是可使用短很多的密钥。

El Gamal加密方案的定义如下：

构造方法 10.21

令$\mathcal{G}$如文中的定义。如下定义一个公钥加密方案：

- Gen：输入1^n，执行$\mathcal{G}(1^n)$来获得$(\mathbb{G}, q, g)$。然后选择随机的$x \leftarrow \mathbb{Z}_q$并且计算$h := g^x$。公钥是$(\mathbb{G}, q, g, h)$，私钥是$(\mathbb{G}, q, g, x)$。
- Enc：输入公钥$pk = (\mathbb{G}, q, g, h)$和消息$m \in \mathbb{G}$，选择随机的$y \leftarrow \mathbb{Z}_q$，并且输出密文$\langle g^y, h^y \cdot m \rangle$。

Dec：输入私钥$sk = (\mathbb{G}, q, g, x)$和密文$\langle c_1, c_2 \rangle$，输出$m = c_2/c_1^x$。

El Gamal加密方案

为了解密成功，令$\langle c_1, c_2\rangle = \langle g^y, h^y \cdot m\rangle$，这里$h = g^x$。则

$$\frac{c_2}{c_1^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{(g^x)^y \cdot m}{g^{xy}} = \frac{g^{xy} \cdot m}{g^{xy}} = m$$

为了完全描述这个方案，需要说明如何把二进制串编码成为$\mathbb{G}$中的元素。这将在本节的最后讨论。

现在证明El Gamal加密方案的安全性。读者可能希望比较下面定理的证明和定理 3.16 和 9.3 的证明。

定理 10.22 如果DDH问题相对于$\mathcal{G}$来说是困难的，则El Gamal加密方案在选择明文攻击情况下有不可区分的加密。

证明 令Π表示El Gamal加密方案。证明在窃听者存在的情况下Π有不可区分加密。由命题 10.5 可知它是 CPA 安全的。

令$\mathcal{A}$为概率多项式时间的敌手，并且定义

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1]$$

考虑到修改的“加密方案”$\tilde{\Pi}$，其中Gen和Π中的一样，但是用公钥$(\mathbb{G}, q, g, h)$加密一个消息m可以通过选择随机的$y \leftarrow \mathbb{Z}_q$和$z \leftarrow \mathbb{Z}_q$来完成，并且输出密文$\langle g^y, g^z \cdot m\rangle$。尽管$\tilde{\Pi}$实际上不是一个加密方案（因为接收者不可能解密），实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$是有良好定义的，因为实验仅依赖于密钥产生和加密算法。

引理 10.20 和它下面的讨论意味着方案$\tilde{\Pi}$中的密文第二个元素是均匀分布的群元素，特别是，独立与要加密的消息m。（记住当z是从$\mathbb{Z}_q$中随机选择的，g^z是$\mathbb{G}$中的一个随机元素。）密文中的第一个元素和m是明显独立的。总的来说，这意味着整个密文不包含m的任何信息。因此，

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n) = 1] = \frac{1}{2}$$

现在来考虑下面的PPT算法D，试图解决与$\mathcal{G}$相关的DDH问题（回顾D接收到$(\mathbb{G}, q, g, g_1, g_2, g_3)$其中对于随机的$x, y, z$，有$g_1 = g^x$，$g_2 = g^y$，而$g_3$要么等于$g^{xy}$要么等于$g^z$）：

算法D：

算法以$\mathbb{G}, q, g, g_1, g_2, g_3$为输入。

(1) 设$pk = (\mathbb{G}, q, g, g_1)$，运行$\mathcal{A}(pk)$来得到消息$m_0, m_1$。

(2) 选择一个随机比特b，并且设$c_1 := g_2$，$c_2 := g_3 \cdot m_b$。

(3) 把密文$\langle c_1, c_2\rangle$给$\mathcal{A}$，并且输出一位b'。如果$b' = b$，则输出 1；否则输出 0 。

分析D的行为。有两种情况要考虑：

情况 1：D的输入是由执行$\mathcal{G}(1^n)$来获得$(\mathbb{G}, q, g)$，随机选择$x, y, z \leftarrow \mathbb{Z}_q$，并且设$g_1 := g^x$，$g_2 := g^y$，$g_3 := g^z$。然后$D$执行$\mathcal{A}$，$\mathcal{A}$有如下公钥：

$$pk = \langle \mathbb{G}, q, g, g^x\rangle$$

和如下密文：

$$\langle c_1, c_2\rangle = \langle g^y, g^z \cdot m_b\rangle$$

易知，在这种情况下当$\mathcal{A}$作为D的子程序运行，与$\mathcal{A}$在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n)$中，其视图分布是一样的。因为$D$输出 1 的时候也是$\mathcal{A}$的输出$b'$等于$b$的时候，有

$$\Pr[D(\mathbb{G},q,g,g^x,g^y,g^z)=1]=\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\tilde{\Pi}}(n)=1]=\frac{1}{2}$$

情况 2： D的输入是运行$\mathcal{G}(1^n)$获得的$(\mathbb{G},q,g)$，随机选择$x,y\leftarrow\mathbb{Z}_q$，并且设$g_1:=g^x$，$g_2:=g^y$，$g_3:=g^{xy}$。然后$D$执行$\mathcal{A}$，有如下公钥：

$$pk=\langle\mathbb{G},q,g,g^x\rangle$$

和如下密文：

$$\langle c_1,c_2\rangle=\langle g^y,g^{xy}\cdot m_b\rangle=\langle g^y,(g^x)^y\cdot m_b\rangle$$

易知，在这种情况下，$\mathcal{A}$作为D的子程序运行，与$\mathcal{A}$在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中，其视图分布是一样的。因为$D$输出 1 的时候也是$\mathcal{A}$的输出$b'$等于$b$的时候，有

$$\Pr[D(\mathbb{G},q,g,g^x,g^y,g^{xy})=1]=\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]=\varepsilon(n)$$

由于与$\mathcal{G}$相关DDH的问题是困难的，必然存在一个可忽略函数negl，满足

$$\begin{aligned}\left|\mathsf{negl}(n)\geqslant\Pr[D(\mathbb{G},q,g,g^x,g^y,g^z)=1]-\Pr[D(\mathbb{G},q,g,g^x,g^y,g^{xy})=1]\right|\\ =\left|\frac{1}{2}-\varepsilon(n)\right|\end{aligned}$$

这意味着$\varepsilon(n)\leqslant\frac{1}{2}+\mathsf{negl}(n)$，证毕。

■

El Gamal的实现问题。简单讨论一下关于El Gamal加密的一些实现方面的问题。

编码二进制串。 正如前面所提到的，为了完全描述可用的加密方案，需要说明怎样把二进制串编码成$\mathbb{G}$中的元素。这当然取决于要考虑的特殊类型的群。下面大致描述了一种可能的编码方法，即当$\mathbb{G}$是二次剩余模一个强素数的子群的时候，同 7.3.3 节中讨论的一样。这里给出的编码方法是为了简单起见而选择的，更高效的编码方法是可能的。

令p是一个强素数，即$q=(p-1)/2$也是一个素数。二次剩余模p的集合构成一个模p乘法群$\mathbb{G}$，其阶为q。可以通过平方把整数$\{1,\cdots,(p-1)/2\}$映射到二次剩余模p的集合：即整数$\tilde{m}$映射到二次剩余$m=[\tilde{m}^2 \bmod p]$。这种编码是单射，并且求逆计算是高效率的。它是单射是因为二次剩余$[\tilde{m}^2 \bmod p]$有两个平方根$[\pm\tilde{m} \bmod p]$，只有一个值在$\{1,\cdots,(p-1)/2\}$范围内。（这是因为当且仅当$[-\tilde{m} \bmod p]=p-[\tilde{m} \bmod p]>(p-1)/2$，有$[\tilde{m} \bmod p]\leqslant(p-1)/2$。）这种编码是可高效求逆的，因为平方根模$p$容易计算（感兴趣的读者参见 11.1.1 节和 11.2.1 节）。

根据上述内容，用如下的方式将长度为$n-1$的串$\hat{m}$映射到$\mathbb{G}$中的元素：给予一个串$\hat{m}\in\{0,1\}^{n-1}$，把它解释为一个整数，并且加 1 得到整数$\tilde{m}$，满足$1\leqslant\tilde{m}\leqslant q$（$q=(p-1)/2$），取$m=[\tilde{m}^2 \bmod p]$。

例 10.23 令$q=83,p=2q+1=167$，并且令$\mathbb{G}$为二次剩余模p的群。因为q是素数，$\mathbb{G}$中除了 1 以外的任何元素都是生成元。取$g=2^2=4 \bmod 167$。例如，接收者选取私钥$37\in\mathbb{Z}_{83}$，从而公钥是

$$pk = \langle 167, 83, 4, [4^{37} \bmod 167] \rangle = \langle 167, 83, 4, 76 \rangle$$

为了加密 6 比特的消息$\hat{m} = 011101$，把它看作整数 29，然后加 1 得$\tilde{m} = 30$。将它平方，有$m = [30^2 \bmod 167] = 65$。这就是对消息的编码。加密的时候选择$y = 71$，得到密文：

$$\langle [4^{71} \bmod 167], [76^{71} \cdot 65 \bmod 167] \rangle = \langle 132, 44 \rangle$$

为了解密，接收方首先计算$124 = [132^{37} \bmod 167]$。然后，由于$66 = [124^{-1} \bmod 167]$，接收方能接收到$m = 65 = [44 \cdot 66 \bmod 167]$。这个$m$有两个平方根 30 和 137，但是后者比$q$大。所以$\tilde{m} = 30$并且$\hat{m}$是 29 的二进制表示。

共享公共参数。构造方法 10.21 中的El Gamal加密方案需要接收方运行$\mathcal{G}$来生成$\mathbb{G}, q, g$。然而，实际上，通常这些参数是“一次产生，多次使用”，并且可被多个接收者使用。（例如，一个系统管理者可以将这些参数固定在一个特别的安全参数n上，系统中的每个人都可以共享这些值。）当然，每个接收方必须选择它们自己的保密值x，然后公布自己的包含$h = g^x$的公钥。

选择公共的参数并不认为是对加密方案安全性的妥协。假设DDH问题与$\mathcal{G}$相关是困难的，则使用共享的公共参数是没有问题的，因为DDH问题至今仍然被认为是困难的，即使是对于运行$\mathcal{G}$生成$\mathbb{G}, q, g$的通信方来说，也是如此。与RSA相比不同的是，RSA中执行GenRSA的通信方，（至少按本书描述的方式）是知道输出的模N的因子分解。当参数共享时的其他攻击已在 10.4.2 节中讲述。结论是：在这种情况下，RSA参数是不能共享的。

10.6 选择密文攻击的安全性

同对称密钥加密情形一样，选择密文攻击在公钥加密情形中也是一个需要关注的问题。在选择密文攻击中，允许敌手选择获得任意密文的解密。事实上，公钥加密情形中这更值得关注，因为在公钥加密中接收者希望能从多个发送者那里接到密文，那些发送方可能事先并不知道，然而在对称密钥加密情形中，接收方是试图和单个已知的发送者通信，并使用特定的秘密密钥。

存在很多可能的选择密文攻击的现实场景。假设一个窃听者$\mathcal{A}$观察到一个从发送方$\mathcal{S}$发给接收方$\mathcal{R}$的密文c。在公钥情形中，可以设想两类可能发生的选择密文攻击：

(1) $\mathcal{A}$可能把密文c'发给$\mathcal{R}$，但是宣称这个密文是发送方$\mathcal{S}$发送的。（比如，在加密的e-mail的情形中，$\mathcal{A}$可能编写一个加密的电子邮件消息c'，并且伪装“From”字段，以便看起来好像是由$\mathcal{S}$发送的。）在这种情况下，尽管$\mathcal{A}$不太可能完全得到c'的解密m'，但是$\mathcal{A}$可能基于$\mathcal{R}$随后的行为推断出关于m'的信息。通过c产生出c'，解密c'得到m'，可能推断出关于m的信息。

(2) $\mathcal{A}$以自己的名字把密文c'发送给$\mathcal{R}$。在这种情况下，$\mathcal{A}$可能更容易获得c'的整个解密m'，因为$\mathcal{R}$会直接给$\mathcal{A}$回复。还有一种可能性是$\mathcal{A}$不能完全获得c'的解密，但是m'的内容有利于$\mathcal{A}$（参见下面第三个情景的例子）。

注意第二类的攻击只适用公钥加密的情形，对于对称密钥加密没有任何意义。下面给出一些上述攻击类型的例子。在每个例子中，会讨论重放c（即令$c' = c$）的影响：为什么定义中不允许这种类型的问询，但定义仍然是有实践意义的安全概念。

情景 1。例如一个用户$\mathcal{S}$通过发送一个加密的口令pw给银行来登录她的银行账户。（这种登录方式存在其他问题，但是这里忽略。）假设在登录失败后，银行发送了两种类型的错误消息：接收到假设拥有银行帐号的用户$\mathcal{S}$发送来的加密的pw，如果pw中包含任何非字母字符，则银行发送“无效口令”；如果pw是正确的口令，但是与存储$\mathcal{S}$的密钥不匹配，则银行返回“口令不正确”。

如果敌手获得了$\mathcal{S}$发送给银行的密文c，则能以$\mathcal{S}$的名义通过发送密文c'给银行，并观察导致的错误信息来发起（部分）选择密文攻击。这个信息可能足够使敌手来确定用户的口令。注意，敌手发送c给银行，并不能获得关于用户密码的任何信息，因为已知在这种情况下将不会产生任何错误消息[5]。

情景 2。比如$\mathcal{S}$发送一封加密的邮件c给$\mathcal{R}$，并且这封电子邮件被$\mathcal{A}$看到了。如果$\mathcal{A}$发送（用自己的名字）加密的电子邮件c'给$\mathcal{R}$，则$\mathcal{R}$可能回复这封邮件并且引用c'的解密文本m'。在这种情况下，$\mathcal{R}$恰好扮演着$\mathcal{A}$的解密预言机，可能解密$\mathcal{A}$发送的任何密文。另外，如果$\mathcal{A}$把c本身发送给$\mathcal{R}$，则$\mathcal{R}$可能会怀疑并且拒绝回复，这取决于消息m的内容。

情景 3。选择密文攻击安全的一个密切的相关问题是密文的可延展性。因为正式的定义很繁琐，这里只给出一个直观上的解释。即一个加密方案拥有这样一个属性：给定未知消息m的密文c，可得到未知消息m'的密文c'，其中m'和m具有某种已知的关联。例如，给定m的密文c，可以构造$2m$的密文c'。（本节后面将看到拥有这种类似属性的例子，另外参见 11.3.3 节。）

现在设想$\mathcal{R}$正在进行拍卖，其中$\mathcal{S}$和$\mathcal{A}$双方使用$\mathcal{R}$的公钥来加密竞价并提交。如果 CPA 安全的加密方案有上述使用的属性，则对敌手$\mathcal{A}$来说可能通过完成以下攻击来使自己总是能给出最高的竞价（并不须出最大价）：等待$\mathcal{S}$发送其出价m（对$\mathcal{A}$来说是未知的）的密文c；然后，$\mathcal{A}$发送出价$m' = 2m$的密文c'。注意m和m'对于$\mathcal{A}$均是未知的（直到$\mathcal{R}$宣布结果），因此这种攻击的概率与该加密方案是 CPA 安全的这个事实并不矛盾。相比而言，可以证明 CCA 安全的方案非延展的，即意味着不会受到这类攻击。

定义。现在给出一个抗选择密文攻击安全性的正式定义，完全类似于定义 3.30。（区别是没有给敌手访问加密预言机的权利；和前面讨论的一样，在公钥情形中加密预言机不能给予敌手额外的能力。）对于一个公钥加密方案Π和敌手$\mathcal{A}$，考虑如下实验：

选择密文攻击不可区分实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$：

(1) 执行$\mathsf{Gen}(1^n)$来获得密钥(pk, sk)。

(2) 给敌手pk，并且可以使用解密预言机$\mathsf{Dec}(\cdot)$。输出一对有相同长度的消息m_0, m_1。（消息来自于与pk相关联的明文空间。）

(3) 选择随机的一个比特$b \leftarrow \{0,1\}$，然后计算出密文$c \leftarrow \mathsf{Enc}_{pk}(m_b)$，并把密文给$\mathcal{A}$。

(4) $\mathcal{A}$继续使用解密预言机，但是不能问询c本身的解密。最后，$\mathcal{A}$输出一个比特b'。

---

⑤ 重新发送 c 是一个“重放攻击”的例子，此例可欺骗银行使其认为 $\mathcal{S}$ 已经登录而产生负面影响。这一点可以通过用户一起加密时间戳和密码来修复，但是增加时间戳并没有改变所描述的攻击的可行性。（事实上，此时攻击可能变得更容易，因为现在存在第三种错误信息：“无用的时间戳”。）

(5) 如果$b' = b$则实验的输出定义为 1，否则为 0。

定义 10.24 公钥加密方案$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$是选择密文攻击下不可区分加密（或是CCA 安全的加密），如果对于所有的概率多项式时间的敌手$\mathcal{A}$，存在可忽略的函数negl，满足

$$\Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

本书将不会对 CCA 安全的加密方案举例，因为对所有存在的构造方法的安全性证明都相当复杂。然而，要指出的是：已经知道有基于DDH假设的并且效率大约是El Gamal加密方案的两倍的 CCA 安全的加密方案（参见本章末的参考文献）。第 13 章将讨论 CCA 安全的加密方案，它是高效的但是仅在理想模型（在该章详细介绍）下是可证明安全的。构造一个更简单的或者更加高效的 CPA 安全的公钥加密方案，特别是基于RSA假设的方案，是一个重要的公开问题。

选择密文攻击的例子。加密方案容易受到选择密文攻击，这并不仅仅是一个理论上的可能性。这里说明目前所给出的所有方案在这类攻击下都是不安全的。

教科书式 RSA 加密。在之前的讨论中，曾提到教科书式RSA加密方案至少是满足以下安全属性（假设与GenRSA相关RSA问题是困难的）：如果消息m均是随机地从$\mathbb{Z}^*_N$中选择，并且使用公钥$\langle N, e\rangle$加密，则窃听敌手不能完全恢复出m。不难得出如果允许敌手发动选择密文攻击，即使是这样的弱属性也不再有了。比如敌手$\mathcal{A}$拦截了密文$c = [m^e \bmod N]$。接着敌手可以选择一个随机的$r \leftarrow \mathbb{Z}^*_N$，并且计算出密文$c' = [r^e \cdot c \bmod N]$。给予这个密文的解密明文$m'$，$\mathcal{A}$能恢复$m = [m' \cdot r^{-1} \bmod N]$。因为

$$m' \cdot r^{-1} = (c')^d r^{-1} = (r^e \cdot m^e)^d r^{-1} = r^{ed} m^{ed} r^{-1} = rmr^{-1} = m \bmod N$$

其中d是接收者私钥中包含的值，用来解密c'（因此$ed = 1 \bmod \phi(N)$）。

教科书式RSA加密同样易受到之前讨论的在拍卖场景中的攻击。比方说敌手观察到使用公钥$\langle N, e\rangle$加密的密文$c = [m^e \bmod N]$。有密文$c' = [2^e c \bmod N]$解密为$[2m \bmod N]$，这是因为

$$(c')^d = (2^e m^e)^d = 2^{ed} m^{ed} = 2m \bmod N$$

其中d和上述一样。

PKCS#1 v1.5。前面曾经提到，作为PKCS #1 v1.5标准的一部分的公钥加密方案，使用了各种填充RSA加密，其中填充部分是以特别的方式完成（且不由任意比特组成）。如果解密密文没有正确的格式，将返回一个错误消息。它表明了这些错误信息的存在足以发起选择密文攻击。换言之，给定一个正确生成的密文c，攻击者可以恢复出明文m，方法是通过提交多个密文并且观察哪个密文能成功解密，哪个会产生错误。由于这类信息很容易获得（例如，来源于当接收到不正确格式的消息时，发出错误信息的服务器），这个攻击是实践中可行的（尽管这个攻击需要大量的密文）。

这种实践中的选择密文攻击的存在某种程度上是有益的，它促使标准化一个改进加密方案的工作，该方案具有选择密文攻击下可证明安全性。这些工作在 13.2.3 节中讨论的一个（一类）方案中达到高潮。

El Gamal 加密。El Gamal加密方案和教科书式RSA加密一样，容易受到选择密文攻

击。这可能让人有点惊讶，因为曾经证明了El Gamal加密方案在DDH假设下是 CPA 安全的，但是再一次强调，这没有矛盾，因为现在讨论的是一个更强的攻击模型。

比如敌手$\mathcal{A}$拦截了使用公钥$pk = \langle \mathbb{G}, q, g, h\rangle$加密消息$m$的密文$c = \langle c_1, c_2\rangle$，这意味着

$$c_1 = g^y, c_2 = h^y \cdot m$$

一些$y \in \mathbb{Z}_q$对$\mathcal{A}$来说是未知的。然而，如果敌手计算$c_2' := c_2 \cdot m'$，则很容易知道密文$c' = \langle c_1, c_2'\rangle$是消息$m \cdot m'$的密文。这个观察结果容易引起选择密文攻击，并且也表明了El Gamal加密在前面提到的拍卖场景中是易受到攻击的。

有人可能会反对：如果接收者收到两个密文c, c'，并且密文的前一个元素相同，将产生怀疑。（确实，对于正常生成密文，这种情况发生的概率可忽略。）然而，敌手很容易避免其发生，令c_1, c_2, m, m'如上所述，$\mathcal{A}$可以选择随机的$y'' \leftarrow \mathbb{Z}_q$并且设$c_1'' := c_1 \cdot g^{y''}$，$c_2'' := c_2 \cdot h^{y''} \cdot m'$。则

$$c_1'' = g^y \cdot g^{y''} = g^{y+y''} 并且 c_2'' = h^y \cdot m \cdot h^{y''} m' = h^{y+y''} mm'$$

因此密文$c'' = \langle c_1'', c_2''\rangle$是$m \cdot m'$的密文，且它的第一个元素是完全随机的。

10.7 *陷门置换

（本节是以 7.4.1 节中介绍的内容为依据的。）

在 10.4.3 节中看到怎样建立一个基于RSA假设的 CPA 安全的公钥加密方案。通过提取在构造中使用的RSA的那些特性，并定义一个封装这些属性的抽象概念，能获得一个构造安全加密方案的通用“模板”，这个模板是基于任何满足相同属性集的原语来构造的。“陷门置换”是单向置换的一个特例，作为一个这种用途的抽象。

下一节将定义陷门置换簇，并且观察到RSA的单向置换簇（构造方法 7.71）满足成为陷门置换簇的附加条件。10.7.2 节给出从任何陷门置换中构造公钥加密方案的方法。除 10.7.2 节外，10.7.1 节中的内容仅在 11.2 节中直接用到，该节给出了陷门置换的第二个例子。陷门置换虽然在第 13 章中提到，但是对理解该章的内容来说不是很重要。10.7.2 节的内容没有在本书剩下的部分中使用到。

10.7.1 定义

回忆一下 7.4.1 节中的函数簇的定义和单向置换簇的定义，该节说明了RSA假设产生单向置换簇。细心的读者可能已注意到所给的构造方法（构造方法 7.71）没有提到一个特殊属性：即参数生成算法Gen输出一些额外信息，包括能使f_I高效可逆的I。称这些额外信息为“陷门”，并且称这些有额外信息的单向置换簇为“陷门置换簇”。正式的定义如下：

定义 10.25 一个概率多项式时间算法 4 元组$(\mathsf{Gen}, \mathsf{Samp}, f, \mathsf{Inv})$是一个陷门置换簇（或者一个陷门置换），如果有：

(1) 参数生成的概率算法Gen，输入为1^n，输出为(I, td)，其中$|I| \geqslant n$。每个由Gen输出的(I, td)定义一个集合$\mathcal{D}_I = \mathcal{D}_{\mathsf{td}}$。

(2) 令Gen_1表示通过运行Gen产生的算法，只输出I。即$(\mathsf{Gen}_1, \mathsf{Samp}, f)$是一个单向置换簇。

(3) 确定性求逆算法Inv，输入为td和$y \in \mathcal{D}_{\mathsf{td}}$，输出为$x \in \mathcal{D}_{\mathsf{td}}$。写为$x := \mathsf{Inv}_{\mathsf{td}}(y)$。对于$\mathsf{Gen}(1^n)$产生的所有的$(I, \mathsf{td})$和所有的$x \in \mathcal{D}_I$，要求有

$$\mathsf{Inv}_{\mathsf{td}}(f_I(x)) = x$$

最后一个条件意味着f_I可以通过td而求逆。第二个条件意味着没有td时f_I是不能高效求逆的。那么有：只要RSA问题与GenRSA相关是困难的，构造方法 7.71 可以修改以得到一个陷门置换簇，这里称之为RSA陷门置换。基于因式分解假设的另一个陷门置换将在 11.2.2 节中给出。

10.7.2 来自陷门置换的公钥加密

现在粗略地看一下怎样从一个任意的陷门置换中构造一个公钥加密方案。尽管读者可能在阅读了第 6 章后更能体会本节的内容，但是第 6 章不是理解本节的前提。为了保持本节的独立性（自包含性），内容会有重复。

在继续解释之前，先引入一些速记表示法是有益处的。4 元组$(\mathsf{Gen}, \mathsf{Samp}, f, \mathsf{Inv})$是陷门置换簇，并且$(I, \mathsf{td})$是Gen生成的一对值，简单地用“$x \leftarrow \mathcal{D}_I$”表示从$\mathcal{D}_I$中随机地选取一个元素（不再明确提及算法Samp）。用f_I^{-1}代替$\mathsf{Inv}_{\mathsf{td}}$，并且$f_I^{-1}$只能在td已知的情况下高效地计算出来。因而简单用$(\mathsf{Gen}, f)$表示陷门置换簇，尽管正式的表示是$(\mathsf{Gen}, \mathsf{Samp}, f, \mathsf{Inv})$。

仅仅给出I和$f_I(x)$，其中I是随机生成，且x是随机选择的，(Gen_1, f)是一个单向置换，这意味着不能期待高效地计算出x。然而，这不意味着关于x的某些信息（比方说x的最小比特位）很难计算出来。在构造公钥加密方案中的第一步将是：通过发现很难计算的x的单比特信息，来提炼陷门置换簇中的困难性。该想法其实即为硬核谓词的概念。（对于已经学习过第 6 章的读者，下面根据当前的应用情形改编定义 6.5。）

定义 10.26 令$\Pi = (\mathsf{Gen}, f)$为陷门置换簇。令hc为一个确定的多项式时间算法，其中输入为I和$x \in \mathcal{D}_I$，输出为单一比特$\mathsf{hc}_I(x)$。如果对于每个概率多项式时间算法$\mathcal{A}$，存在可忽略函数negl，满足

$$\Pr[\mathcal{A}(I, f_I(x)) = \mathsf{hc}_I(x)] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

则hc是Π的硬核谓词，其中概率来源于由$\mathsf{Gen}(1^n)$生成的(I, td)和x从$\mathcal{D}_I$中均匀选择的随机性。

给定一个陷门置换簇$\hat{\Pi}$和对这个簇的硬核谓词的hc，可以用如下方法加密 1 个比特。接收者运行$\hat{\mathsf{Gen}}(1^n)$来生成(I, td)，并且设其公钥为I，私钥为td。给定公钥I，发送方加密 1 比特m的步骤如下：①选择随机的$x \leftarrow \mathcal{D}_I$；②计算$y := f_I(x)$；③发送密文$\langle y, \mathsf{hc}_I(x) \oplus m \rangle$。为了解密密文$\langle y, m' \rangle$，接收者首先使用私钥td计算$x := f_I^{-1}(y)$，然后输出消息$m := \mathsf{hc}_I(x) \oplus m'$（见构造方法 10.27）。容易看到恢复原始消息。

构造方法 10.27

令$\hat{\Pi} = (\hat{\mathsf{Gen}}, f)$为陷门置换族，并且令hc为对于$\hat{\Pi}$的硬核谓词。构造如下公钥加密方案：

- Gen：输入为1^n，执行$\hat{\mathsf{Gen}}(1^n)$来获得(I, td)。输出公钥I和私钥td。
- Enc：输入公钥I和消息$m \in \{0,1\}$，随机选择$x \leftarrow \mathcal{D}_I$并且输出密文$\langle f_I(x), \mathsf{hc}_I(x) \oplus m \rangle$。
- Dec：输入私钥td和密文$\langle y, m' \rangle$，其中$y \in \mathcal{D}_{\mathsf{td}}$，计算$x := f_I^{-1}(y)$并且输出消息$\mathsf{hc}_I(x) \oplus m'$。

从陷门置换簇中产生的公钥加密方案

直觉上来说，它是安全的，因为hc是$\hat{\Pi}$的硬核谓词这个事实意味着：以窃听者的观点来看，$\mathsf{hc}_I(x)$是伪随机的，窃听者可以看到公钥I和$f_I(x)$，其中x是随机的。和熟悉的私钥情形中的方式是类似，可得到安全性。下面正式证明。

定理 10.28 如果$\hat{\Pi}$是陷门置换簇并且hc是$\hat{\Pi}$的硬核谓词。则构造方法 10.27 在选择明文攻击下是不可区分加密。

证明 令Π表示由构造方法 10.27 构造的公钥加密方案。先证明Π在窃听者存在情况下具有不可区分的加密，然后运用定理 10.10 得到陈述的结果。

令$\mathcal{A}$为一个概率多项式时间敌手，并且定义

$$\varepsilon(n) \stackrel{\mathsf{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]$$

不失一般性，假设由$\mathcal{A}$输出的消息m_0, m_1总是不同的。（读者应证明这的确不失一般性。）

考虑下面的PPT算法$\mathcal{A}_{\mathsf{hc}}$，在给定$I$和$y=f_I(x)$作为输入时，计算$\mathsf{hc}_I(x)$：

算法$\mathcal{A}_{\mathsf{hc}}$：

算法的输入为I和$y \in \mathcal{D}_I$。

(1) 设$pk=I$并且执行$\mathcal{A}(pk)$以得到$m_0, m_1 \in \{0,1\}$。

(2) 独立地选择随机比特z和b，设$m' := m_b \oplus z$。

(3) 把密文$\langle y, m'\rangle$给$\mathcal{A}$并且得到 1 比特输出b'。如果$b'=b$则输出z，否则输出$\bar{z}$（z的补）。

分析$\mathcal{A}_{\mathsf{hc}}$的行为。令$x$满足$y=f_I(x)$（注意由于$f$是置换，$x$是明确定义的），可以把$z$看作$\mathcal{A}_{\mathsf{hc}}$对于$\mathsf{hc}_I(x)$的初始“猜想”。这个猜想正确的概率是1/2，错误的概率也是1/2。有

$$\begin{aligned}&\Pr[\mathcal{A}_{\mathsf{hc}}(I, f_I(x)) = \mathsf{hc}_I(x)]\\ &= \frac{1}{2}\cdot\Big(\Pr[b'=b \mid z=\mathsf{hc}_I(x)] + \Pr[b'\neq b \mid z\neq \mathsf{hc}_I(x)]\Big) \qquad (10.13)\end{aligned}$$

当$z=\mathsf{hc}_I(x)$时，$\mathcal{A}$作为$\mathcal{A}_{\mathsf{hc}}$的子程序运行，和$\mathcal{A}$在使用比特$b$的实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中，两者视图的分布是相同的。这是因为对于随机选择的值x有m'满足$m'=m_b\oplus \mathsf{hc}_I(x)$，因此给$\mathcal{A}$的密文$\langle y, m'\rangle$确实是$m_b$的随机加密。它满足

$$\Pr[b'=b \mid z=\mathsf{hc}_I(x)] = \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1] = \varepsilon(n)。$$

另外，当$z\neq \mathsf{hc}_I(x)$时，$\mathcal{A}$作为$\mathcal{A}_{\mathsf{hc}}$的子程序运行，和$\mathcal{A}$在使用比特$\bar{b}$的实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中，两者视图的分布是相同的。这是因为$m'$满足

$$m' = m_b \oplus \bar{\mathsf{hc}}_I(x) = m_{\bar{b}} \oplus \mathsf{hc}_I(x)$$

（还记得假设$m_0\neq m_1$），因此给$\mathcal{A}$的密文$\langle y, m'\rangle$是对$m_{\bar{b}}$的随机加密。因此

$$\Pr[b'=b \mid z\neq \mathsf{hc}_I(x)] = \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=0] = 1-\varepsilon(n)$$

则$\Pr[b'\neq b \mid z\neq \mathsf{hc}_I(x)] = \varepsilon(n)$。

结合上述和式(10.13)，有

$$\Pr[\mathcal{A}_{\mathsf{hc}}(I, f_I(x)) = \mathsf{hc}_I(x)] = \frac{1}{2}\cdot(\varepsilon(n)+\varepsilon(n)) = \varepsilon(n)。$$

由于hc是Π的硬核谓词，有$\varepsilon(n) \leqslant \frac{1}{2} + \mathsf{negl}(n)$。证毕。

■

现在只剩下说明陷门置换簇有硬核谓词。对于一些正常的族，比如前面讨论的基于RSA假设的簇，特定的硬核谓词是已经知道的。（举个例子，如果RSA假设成立，则对于RSA陷门置换簇，最不重要的比特是硬核谓词。）通常来说，依赖下面的结果，可通过对定理 6.6 进行适当修改来证明这个结果。

定理 10.29 如果存在陷门置换簇Π，则存在陷门置换簇$\hat{\Pi}$，以及$\hat{\Pi}$的硬核谓词hc。

加密更长的消息。利用命题 10.11，可扩展构造方法 10.27 来加密l比特的消息，l为任意多项式。这样，使用公钥I，对应l比特消息$m = m_1 \cdots m_l$的密文的形式为

$$\langle f_I(x_1), \mathsf{hc}_I(x_1) \oplus m_1 \rangle, \cdots, \langle f_I(x_l), \mathsf{hc}_I(x_l) \oplus m_l \rangle$$

其中$x_1, \cdots, x_l$的选择是独立随机地从$\mathcal{D}_I$中选择的。

可以减少密文的长度，通过让发送者如下进行：选择随机的$x_1 \leftarrow \mathcal{D}_I$并且计算$x_{i+1} := f_I(x_i)$，其中$i = 1, \cdots, l$。然后输出密文

$$\langle x_{l+1}, \mathsf{hc}_I(x_1) \oplus m_1, \cdots, \mathsf{hc}_I(x_l) \oplus m_l \rangle。$$

其安全性证明要使用 6.4 节的思路，留作高级练习。

参考文献和扩展阅读材料介绍

公钥加密的思想是由Diffie和Hellman[47]首次提出的（首次在公开的文献中提出）。但是让人吃惊的是，尽管它被看作是 1976 年Diffie和Hellman引入的Diffie-Hellman密钥交换协议的直接变形，直到 1984 年才提出El Gamal加密方案[59]。Rivest、Shamir和Adleman[122]引入了RSA假设并基于该假设提出公钥加密方案。

定义 10.3 来源于Goldwasser和Micali[69]的工作，他们也是首先认识到满足该定义的概率加密的必要性。对于混合加密的特例的安全性证明是由Blum和Goldwasser[22]首先给出的。

由Rivest、Shamir和Adleman[122]给出的公钥加密方案和这里给出的教科书式RSA方案是对应的。10.4.2 节中描述的攻击在参考文献[73, 45, 133, 27]有说明，有参见参考文献[99]的第 8 章和参考文献[25]以了解更多攻击和解释。PKCS #1 RSA加密标准（最新版本和前面的版本）可以在http://ww.rsa.com/rsalabs下载。定理 10.19 的证明可以来源于参考文献[5,75]。

正如在第 4 章中提到的，选择密文攻击首先是由Naor和Yung[107]、Rackoff和Simon[121]定义的。对“教科书式RSA”和El Gamal加密方案的选择密文攻击是明显的；对PKCS #1 v1.5的攻击源于Bleichenbacher[20]。对于最新定义的更强攻击下的公钥加密方案，包括选择密文攻击，读者可参考Dolev等[50]和Bellare等[10]的著作。第一个选择密文攻击安全的高效公钥加密方案是由Cramer和Shoup[39]给出的。Shoup[129]的说明性论文讨论了选择密文攻击安全的重要性。

基于任意陷门置换的公钥加密的存在性是由Yao[149]给出的，还有 10.7.2 节中讨论的效率改进的问题应归功于Blum和Goldwasser[22]。如果读者对RSA陷门置换簇的硬核谓词感兴趣，请阅读参考文献[5，75，4]以及这些书的参考文献。

在实践中使用任何加密方案，就会出现密钥的长度问题。这个问题不应该掉以轻心，建议读者阅读参考文献[94]对这个问题的论述。

练　习

10.1 一个针对单比特消息的公钥加密方案，说明：给定pk和密文c，计算$c \leftarrow \mathsf{Enc}_{pk}(m)$，对于一个无限制的敌手，以 1 的概率确定$m$是可能的。这表明完美安全的公钥加密是不可能的。

10.2 假定一个确定性公钥加密方案用来加密消息m，该消息来自于一个具有$\mathcal{L}$个可能值的小集合。说明如何在关于$\mathcal{L}$的线性时间里确定m（假设加密一个元素耗费一个单位时间）。

10.3 说明对任意 CPA 安全的公钥方案，在加密一个单比特后，其密文的大小是关于安全参数的超离散对数。（即对$(pk, sk) \leftarrow \mathsf{Gen}(1^n)$，任意$b \in \{0,1\}$，满足$|\mathsf{Enc}_p k(b)| = \omega(log\ n)$。）

提示：否则，可能密文范围的大小仅是多项式的。

10.4 说明任何 2 轮密钥交换协议（即每方发送一个消息的情形）满足定义 9.1，可以转换成 CPA 安全的公钥加密方案。

10.5 说明在定义 10.7 中，可不失一般性地假设$\mathcal{A}$总是输出两个向量，其中每个恰好包含$t(n)$消息。换而言之，说明对任意方案Π和任意敌手$\mathcal{A}$，如何构造一个敌手$\mathcal{A}'$，总是输出同样的$t(n)$长度的向量，对于每个固定的n值满足$\Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) = 1] = \Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A}',\Pi}(n) = 1]$。

10.6 证明声称 10.9。

10.7 固定N，假设存在一个敌手$\mathcal{A}$，在时间t运行，满足$\Pr[\mathcal{A}(x^e \mod N) = x] = 0.01$，这里概率来源于随机选择的$x \rightarrow \mathbb{Z}^*_N$。说明可以构造一个敌手$\mathcal{A}'$满足$\Pr[\mathcal{A}'(x^e \mod N) = x] = 0.99$，$\mathcal{A}'$的运行时间$t'$满足$t' = \mathsf{poly}(||N||, t)$。

提示：使用事实$y^{1/e} \cdot r = (y \cdot r^e)^{1/e} \mod N$。

10.8 RSA中的公开指数e是任意选择的，满足$\gcd(e, \phi(N)) = 1$。常选的e包括$e = 3$，和$e = 2^{16} + 1$。解释为什么这样的e比起同样长度的随机值更好。

提示：参见附录 B. 2. 3 的模乘方算法。

10.9 令GenRSA的含义如前所述。考虑下面的对于算法$\mathcal{A}$和函数l的实验，对所有的n，有$l(n) \leqslant 2n - 2$。

填充RSA实验$\mathsf{PAD}_{\mathcal{A},\mathsf{GenRSA},l}(n)$：

（1）运行$\mathsf{GenRSA}(1^n)$得到输出(N, e, d)。

（2）把N, e给$\mathcal{A}$，输出一个串$m \in \{0,1\}^{l(n)}$。

（3）随机选择$y_0 \leftarrow \mathbb{Z}^*_N$。随机选择$r \leftarrow \{0,1\}^{||N||-l(n)-1}$和集合

$y_1 := [(r||m)^e \mod N]$。

（4）随机选择$b \leftarrow \{0,1\}$。敌手$\mathcal{A}$被给予y_b，输出一个比特b'。

（5）若$b' = b$，则实验的输出定义为 1，否则为 0。

如果对于所有概率多项式时间算法$\mathcal{A}$，存在可忽略函数negl满足$\Pr[\mathsf{PAD}_{\mathcal{A},\mathsf{GenRSA},l}(n) = 1] \leqslant \frac{1}{2} + \mathsf{negl}(n)$，则称$l$填充的RSA问题与GenRSA相关是困难的。

证明如果l填充的RSA问题是与GenRSA相关是困难的，则使用l填充的RSA加密方案

（构造方法 10.18）是 CPA 安全的。

10.10 如果对所有PPT算法$\mathcal{A}$，存在一个可忽略函数negl满足

$$|\Pr[\mathcal{A}(N,e,y,f(x))=1]-\Pr[\mathcal{A}(N,e,y,f(r))=1]|\leqslant \mathsf{negl}(n)$$

其中每种情况下概率来源于实验中的$\mathsf{GenRSA}(1^n)$输出(N,e,d)，随机的x，随机选择$r\leftarrow\mathbb{Z}_N^*$，$y$是集合$[x^e \bmod N]$。于是，称函数$f$是GenRSA的硬核。

对于$x\in\mathbb{Z}_N^*$，当把x写为恰有$||N||$比特的整数的时候，令$\mathsf{lsb}(x)$（$\mathsf{msb}(x)$）表示x的最小（最大）影响比特。定义$f(x)\stackrel{\text{def}}{=}\mathsf{msb}(x)||\mathsf{lsb}(x)$。可以证明如果RSA问题与GenRSA相关是困难的，则f是GenRSA的硬核[4]。依靠这个结果证明定理 10.19。

提示：注意在构造方法 10.18 中，$\mathsf{msb}(r||m)$总等于 0。

10.11 考虑下面的公钥加密体制。完全如同El Gamal加密方案一样，公钥是$(\mathbb{G},q,g,h)$，以及私钥x。为了加密一个比特b，发送者操作如下：

（1）如果$b=0$，则选择随机$y\leftarrow\mathbb{Z}_q$，然后计算$c_1=g^y$和$c_2=h^y$。密文是$\langle c_1,c_2\rangle$。

（2）如果$b=1$，则独立随机地选择$y,z\leftarrow\mathbb{Z}_q$，计算$c_1=g^y$和$c_2=g^z$，且设定密文等于$\langle c_1,c_2\rangle$。

说明用给定x的知识高效解密是可能的。证明：如果判定性 Diffie-Hellman 问题与$\mathcal{G}$相关是困难的，则这个加密方案是 CPA 安全的。

10.12 把混合加密应用在El Gamal加密方案的一种自然方式如下。同El Gamal加密方案一样，公钥是$pk=(\mathbb{G},q,g,h)$，为加密消息m，发送方选择随机的$k\leftarrow\{0,1\}^n$，且发送$\langle g^r,h^r\cdot k,\mathsf{Enc}_k(m)\rangle$，其中$r\leftarrow\mathbb{Z}_q$是随机选择的，且Enc表示一个私钥加密方案。请改进方法使密文更短，仅包含单一的群元素，接一个用私钥加密的m。

10.13 说明在 CCA 安全性的情形下，命题 10.11 不成立；相对而言，在 CCA 安全性的情形下，定理 10.10 成立。解释为什么在 CPA 安全情形中，多个消息和单一长消息是没有区别的，然而在 CCA 安全的情形中，这是有区别的。

10.14 考虑后面填充的RSA加密。假设将要加密的消息m有长度$||N||/2$（即约为模长度的一半）。为了加密，首先用一个字节的 0 填充m的左边，然后 10 个随机字节，然后全 0，结果表示为$\bar{m}$（即$\bar{m}=(0^k||r||00000000||m)$，其中$k$是使得$\bar{m}$具有合适长度的 0 的数量）。最后，计算 $c=[\bar{m}^e \bmod N]$。设计这个方案的一个选择密文攻击，并且说明为什么这个方案比起 PKCS#1 v1.5 更容易构造一个选择密文攻击。

10.15 令Π是 CCA 安全的公钥加密方案，且Π'是一个 CCA 安全的对称密钥加密方案。使用构造方法 10.12 实例化的Π和Π'是 CCA 安全的吗？证明你的回答。

10.16 根据下面的构造方法 10.27 的变体，该变体给出另一个使用任何陷门置换簇进行加密的方法。

构造方法 10.30

令$\widehat{\Pi}=(\widehat{\mathsf{Gen}},f)$和hc如构造方法 10.27。

- Gen：同构造方法 10.27。
- Enc：对输入的公钥I和一个消息$m\in\{0,1\}$，选择随机的$x\leftarrow\mathcal{D}_I$，满足$\mathsf{hc}_I(x)=m$，输出密文$f_I(x)$。
- Enc：输入私钥td和密文y，有$y\in\mathcal{D}_{td}$，计算$x:=f_I^{-1}(y)$，并输出消息$\mathsf{hc}_I(x)$。

（1）辨别：加密能够在多项式时间完成。

（2）证明如果$\widehat{\Pi}$是一个陷门置换，且hc是$\widehat{\Pi}$的硬核谓词，则这个构造方法是 CPA 安全的。

10.17 考虑两方A和B公平投硬币的协议（更复杂的版本可能用于互联网赌博）：①一个可信第三方T，发布其公钥pk；②选择一个随机比特b_A，使用pk加密，公开密文c_A给B和T；③然后，B同样的公开密文 $c_B \neq c_A$；④T解密c_A和c_B，然后各方XOR结果得到了投币值。

（1）辨别：即使A是不诚实的（但是B是诚实的），最终的投币结果是均匀分布的。

（2）假设使用El Gamal加密（其中比特b被编码为群中的元素g^b）。说明不诚实的B如何能够使投币值偏向其喜欢的那一个。

（3）说明什么类型的加密方案在此使用会是合适的。请定义一个合适的安全概念，并且证明你的建议达到了这个定义。

第 11 章 *其他公钥加密方案

前一章介绍了一些基于RSA和判定性Diffie-Hellman问题的公钥加密方案。本章将探讨更多的加密方案，这些方案基于其他与因子分解困难性相关的数论问题。虽然RSA与El Gamal加密方案的应用比本章讨论的加密方案都要广泛，但是本章介绍的加密方案和前一章所介绍的同样重要，这些加密方案放在本章讨论是因为它们需要更多的数论知识。这里介绍的每个加密方案至少从理论学习的角度来看值得认真理解：

(1) Goldwasser-Micali加密方案是第一个证明为CPA安全的方案，它基于从合数模的二次非剩余中区分出二次剩余的困难性。此方案所依赖的密码学假设在其他内容中也用到。

(2) Rabin加密方案与RSA加密方案非常相似，但是有一个关键的区别是：基于分解困难假设，Rabin加密方案可证明是CPA安全的。但是前面曾经提到，目前不知道RSA问题的困难性（以及任何基于RSA问题的加密方案的安全性）是否可由分解假设决定。

(3) Paillier加密方案以与分解问题相关的密码学假设为基础（但并不知道是否等同于分解假设），与Goldwasser-Micali加密方案和可证明安全的Rabin变体方案相比，在Paillier加密方案中消息不是按位加密，即Paillier加密的效率可与EI Gamal加密相媲美。Paillier加密方案还有一些其他优势，将在 11.3 节中讨论。

本章中，令p和q为奇素数，N表示两个不同奇素数的乘积，并且将用到所熟悉的 7.1.5 节中的中国剩余定理。

11.1 Goldwasser-Micali 加密方案

首先讨论Goldwasser-Micali加密方案。在给出这个方案之前，需要对二次剩余进行更好的理解。首先探讨素数模二次剩余这一简单情形，然后讨论稍微复杂的合数模二次剩余。

11.1.1 素数模二次剩余

给定群$\mathbb{G}$，如果存在一个$x \in \mathbb{G}$满足$x^2 = y$，则元素$y \in \mathbb{G}$是一个二次剩余。此时，称x为y的一个平方根。一个元素不是二次剩余则称为二次非剩余。在阿贝尔群中，二次剩余的集合形成一个子群。

在$\mathbb{Z}_p^*$的特例中，如果存在x满足$x^2 = y \mod p$，则称y是一个二次剩余。下面从一个简单的命题开始。

命题 11.1 设$p > 2$为素数，$\mathbb{Z}_p^*$中的每个二次剩余都有两个平方根。

证明 设$y \in \mathbb{Z}_p^*$是一个二次剩余。则存在$x \in \mathbb{Z}_p^*$满足$x^2 = y \bmod p$。显然，$(-x)^2 = x^2 = y \bmod p$。此外，$-x \neq x \bmod p$：如果$-x = x \bmod p$，则$2x = 0 \bmod p$，意味着$p \mid 2x$。因为p为素数，这意味着$p \mid 2$（因为$p > 2$，所以不可能成立）或$p \mid x$（因为$0 < x < p$，所以也不可能成立）。所以，$[x \bmod p]$和$[-x \bmod p]$是$\mathbb{Z}_p^*$中互不相同的元素，即y至少有两个平方根。

设$x' \in \mathbb{Z}_p^*$是y的一个平方根。则$x^2 = y = (x')^2 \bmod p$意味着$x^2 - (x')^2 = 0 \bmod p$。等式左边因子分解，得

$$(x - x')(x + x') = 0 \bmod p$$

所以$p \mid (x - x')$或者$p \mid (x + x')$。在前一种情形中，$x' = x \bmod p$，后一种情形中，$x' = -x \bmod p$，这说明事实上只有$[\pm x \bmod p]$为y平方根。 ■

设sq_p：$\mathbb{Z}_p^* \to \mathbb{Z}_p^*$为函数$\mathsf{sq}_p(x) \stackrel{\text{def}}{=} [x^2 \bmod p]$。上面的命题说明当$p > 2$为素数时，$\mathsf{sq}_p$是一个“二对一”函数。这也意味着$\mathbb{Z}_p^*$中恰好一半的元素为二次剩余。模$p$二次剩余的集合用符号$\mathcal{QR}_p$表示，二次非剩余用$\mathcal{QNR}_p$表示。当$p > 2$为素数时，$|\mathcal{QR}_p| = |\mathcal{QNR}_p| = \dfrac{|\mathbb{Z}_p^*|}{2} = \dfrac{p-1}{2}$。

定义$\mathcal{J}_p(x)$为x模p的雅可比符号[①]如下。设$p > 2$为素数且$x \in \mathbb{Z}_p^*$，则

$$\mathcal{J}_p(x) \stackrel{\text{def}}{=} \begin{cases} +1 & x\text{是一个关于}p\text{的二次剩余} \\ -1 & x\text{是一个关于}p\text{的二次非剩余} \end{cases}$$

则此表示法可以通过定义$\mathcal{J}_p(x) = \mathcal{J}_p([x \bmod p])$很自然地扩展为对于任意$x$与$p$互素的情形。

能否描述$p > 2$为素数时$\mathbb{Z}_p^*$中二次剩余的特征？首先$\mathbb{Z}_p^*$是一个阶为$p-1$的循环群（参考定理 7.53）。设g是$\mathbb{Z}_p^*$的一个生成元。这意味着

$$\mathbb{Z}_p^* = \{g^0, g^1, g^2, \cdots, g^{\frac{p-1}{2}-1}, g^{\frac{p-1}{2}}, g^{\frac{p-1}{2}+1}, \cdots, g^{p-2}\}$$

（前面已经提到p是奇数，所以$p-1$为偶数）。将上面列出的元素逐个平方，再将幂关于$p-1$取模（参考推论 7.15），产生一个$\mathbb{Z}_p^*$中所有二次剩余的列表：

$$\mathcal{QR}_p = \{g^0, g^2, g^4, \cdots, g^{p-3}, g^0, g^2, \cdots, g^{p-3}\}。$$

（注意到每个二次剩余都在列表中出现了两次）。可发现$\mathbb{Z}_p^*$中的二次剩余恰好是一些可以写作g^i的元素，其中$i \in \{0, \cdots, p-2\}$为偶数。

上面的特点引出了一个简单的计算雅可比符号的方式，也就是判断一个给定元素$x \in \mathbb{Z}_p^*$是否为二次剩余。

命题 11.2 设$p > 2$为素数，则$\mathcal{J}_p(x) = x^{\frac{p-1}{2}} \bmod p$。

证明 设g为$\mathbb{Z}_p^*$中的任意一个生成元。如果x是模p的二次剩余，先前的讨论说明对于某些偶数i，有$x = g^i$。记$i = 2j$，其中j为整数，可得

$$x^{\frac{p-1}{2}} = (g^{2j})^{\frac{p-1}{2}} = g^{(p-1)j} = (g^{p-1})^j = 1^j = 1 \bmod p$$

所以$x^{\frac{p-1}{2}} = +1 = \mathcal{J}_p(x) \bmod p$，得证。

另外，如果x不是一个二次剩余，则对于某些奇数i，有$x = g^i$。设$x = 2j+1$，其中j为整数，得到

$$x^{\frac{p-1}{2}} = (g^{2j+1})^{\frac{p-1}{2}} = (g^{2j})^{\frac{p-1}{2}} \cdot g^{\frac{p-1}{2}} = 1 \cdot g^{\frac{p-1}{2}} = g^{\frac{p-1}{2}} \bmod p$$

此时有

$$\left(g^{\frac{p-1}{2}}\right)^2 = g^{p-1} = 1 \bmod p$$

因为$[\pm 1 \bmod p]$为 1 的两个平方根（命题 11.1），所以$g^{\frac{p-1}{2}} = \pm 1 \bmod p$。因为$g$是一个生成元，其阶为$p-1$，所以$g^{\frac{p-1}{2}} \neq 1 \bmod p$。于是$x^{\frac{p-1}{2}} = -1 = \mathcal{J}_p(x) \bmod p$。

■

① $\mathcal{J}_p(x)$有时也称为x的勒让德符号，表示为$\mathcal{L}_p(x)$；后面选择的符号与前面保持一致。

命题 11.2 直接给出一个多项式时间算法，用于测试一个给定元素$x \in \mathbb{Z}_p^*$是否为二次剩余。

算法 11.3
判定素数模二次剩余
输入：素数p；元素$x \in \mathbb{Z}_p^*$
输出：$\mathcal{J}_p(x)$（或者输出x是二次剩余或二次非剩余）
$b = [x^{\frac{p-1}{2}} \bmod p]$
if $b = 1$ **return** “二次剩余”
else return “二次非剩余”

本节最后介绍下面的命题：模p的二次剩余以及二次非剩余有良好的乘法属性。

命题 11.4 设$p > 2$为素数，且$x, y \in \mathbb{Z}_p^*$，则

$$\mathcal{J}_p(xy) = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y)。$$

证明 应用前面的命题：

$$\mathcal{J}_p(xy) = (xy)^{\frac{p-1}{2}} = x^{\frac{p-1}{2}} \cdot y^{\frac{p-1}{2}} = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y) \bmod p。$$

因为$\mathcal{J}_p(xy), \mathcal{J}_p(x), \mathcal{J}_p(y) = \pm 1$，此等式在整数域上也满足。 ■

推论 11.5 设$p > 2$为素数，$x, x' \in \mathcal{QR}_p$，且$y, y' \in \mathcal{QNR}_p$。则：
（1）$[xx' \bmod p] \in \mathcal{QR}_p$。
（2）$[yy' \bmod p] \in \mathcal{QR}_p$。
（3）$[xy \bmod p] \in \mathcal{QNR}_p$。

11.1.2 合数模二次剩余

现在将注意力转到群$\mathbb{Z}_N^*$中的二次剩余。如果结合前面一节的结论和中国剩余定理，容易描述关于模N的二次剩余的特征。回忆中国剩余定理中提到$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$，设$y \leftrightarrow (y_p, y_q)$表示定理中对应的$y_p = [y \bmod p]$和$y_q = [y \bmod q]$。主要观察是：

命题 11.6 设$N = pq$，其中p, q为不同的素数，$y \in \mathbb{Z}_N^*$且满足$y \leftrightarrow (y_p, y_q)$。则当且仅当$y_p$是一个关于模$p$的二次剩余，且$y_q$是一个关于模$q$的二次剩余时，$y$是一个关于模$N$的二次剩余。

证明 如果y是关于模N的二次剩余，由定义可知存在一个$x \in \mathbb{Z}_N^*$，满足$x^2 = y \bmod N$。设$x \leftrightarrow (x_p, x_q)$。则

$$(y_p, y_q) \leftrightarrow y = x^2 \leftrightarrow (x_p, x_q)^2 = ([x_p^2 \bmod p], [x_q^2 \bmod q]),$$

其中$(x_p, x_q)^2$是$\mathbb{Z}_p^* \times \mathbb{Z}_q^*$中元素$(x_p, x_q)$的平方。因而得到

$$y_p = x_p^2 \bmod p, \quad y_q = x_q^2 \bmod q \tag{11.1}$$

且y_p, y_q为二次剩余（对相应的模而言）。

反过来，如果$y \leftrightarrow (y_p, y_q)$且$y_p, y_q$分别为模$p$和$q$的平方剩余，则存在满足方程（11.1）的$x_p \in \mathbb{Z}_p^*$和$x_q \in \mathbb{Z}_q^*$。设$x \in \mathbb{Z}_N^*$满足$x \leftrightarrow (x_p, x_q)$。前面的步骤反过来，可得$x$是$y$关于模$N$的平方根。 ■

前面的命题描述关于模N的二次剩余。仔细检查证明可以得出另外一个重要的观察结论：每个二次剩余$y \in \mathbb{Z}_N^*$恰好有 4 个平方根。为了说明这一点，设$y \leftrightarrow (y_p, y_q)$为一个关于模$N$的二次剩余，设$y_p, y_q$分别为关于模$p$和$q$的二次剩余。则$y$的 4 个平方根在$\mathbb{Z}_N^*$中对应于：

$$(x_p, x_q),\ (-x_p, x_q),\ (x_p, -x_q),\ (-x_p, -x_q)。 \tag{11.2}$$

上面每一个都是y的一个平方根，因为

$$\begin{aligned}(\pm x_p, \pm x_q)^2 &= \Big([(\pm x_p)^2 \bmod p], [(\pm x_q)^2 \bmod q]\Big) \\ &= ([x_p^2 \bmod p], [x_q^2 \bmod q]) = (y_p, y_q) \leftrightarrow y\end{aligned}$$

其中，符号$(\cdot,\cdot)^2$表示群$\mathbb{Z}_p \times \mathbb{Z}_q$中元素的平方。因为$x_p$和$-x_p$是模$p$唯一（类似的$x_q$和$-x_q$模$q$唯一），则中国剩余定理确保了方程（11.2）中 4 个元素每个都对应到$\mathbb{Z}_N^*$中不同的元素。

例 11.7 考虑$\mathbb{Z}_{15}^*$（具体内容由中国剩余定理一节例 7.25 中列表给出）。元素 4 关于模 15 是一个二次剩余，且有一个平方根为 2。因为$2 \leftrightarrow (2, 2)$，由此可得 4 的其他平方根为

(1) $(2, [-2 \bmod 3]) = (2, 1) \leftrightarrow 7$；

(2) $([-2 \bmod 5], 2) = (3, 2) \leftrightarrow 8$；

(3) $([-2 \bmod 5]. [-2 \bmod 3]) = (3, 1) \leftrightarrow 13$。

读者可以验证$7^2 = 8^2 = 13^2 = 4 \bmod 15$。

设$\mathcal{QR}_N$表示模N二次剩余的集合。因为模N平方是一个 4 对 1 的函数，因此立即可以得出$\mathbb{Z}_N^*$中$1/4$的元素为二次剩余。可能注意到当且仅当y_p, y_q为二次剩余时，$y \in \mathbb{Z}_N^*$是一个二次剩余，所以$\mathcal{QR}_N$和$\mathcal{QR}_p \times \mathcal{QR}_q$之间有一一对应的关系。因此，关于模$N$的二次剩余的比例为

$$\frac{|\mathcal{QR}_N|}{|\mathbb{Z}_N^*|} = \frac{|\mathcal{QR}_p| \cdot |\mathcal{QR}_q|}{\mathbb{Z}_N^*} = \frac{\frac{p-1}{2} \cdot \frac{q-1}{2}}{(p-1)(q-1)} = \frac{1}{4}$$

如前所述。

上一节中定义了$p > 2$为素数时的雅可比符号$\mathcal{J}_p(x)$。下面将定义扩展到$N = pq$为两个不同奇素数乘积时的情形。对于任何与$N = pq$互素的x有

$$\begin{aligned}\mathcal{J}_N(x) &\stackrel{\text{def}}{=} \mathcal{J}_p(x) \cdot \mathcal{J}_q(x) \\ &= \mathcal{J}_p([x \bmod p]) \cdot \mathcal{J}_q([x \bmod q])\end{aligned}$$

定义$\mathcal{J}_N^{+1}$为$\mathbb{Z}_N^*$中的雅可比符号为$+1$元素的集合，类似地对$\mathcal{J}_N^{-1}$进行定义。

从命题 11.6 可知，如果x是一个关于模N的二次剩余，则$[x \bmod p]$和$[x \bmod q]$分别为模p和模q的二次剩余；换言之，$\mathcal{J}_p(x) = \mathcal{J}_q(x) = +1$。所以$\mathcal{J}_N(x) = +1$，并且可得：如果$x$是一个关于模$N$的二次剩余，则$\mathcal{J}_N(x) = +1$。尽管如此，当$\mathcal{J}_p(x) = \mathcal{J}_q(x) = -1$时也可出现$\mathcal{J}_N(x) = +1$；也就是说，此时$[x \bmod p]$和$[x \bmod q]$都不是模$p$和$q$的二次剩余（所以$x$不是一个关于模$N$的二次剩余）。此结果对Goldwasser-Micali加密方案非常有用，因此引出符号$\mathcal{QNR}_N^{+1}$表示这类元素的集合。也就是

$$\mathcal{QNR}_N^{+1} \stackrel{\text{def}}{=} \left\{ x \in \mathbb{Z}_N^* \middle| x \text{ 不是模 } N \text{ 的二次剩余，但 } \mathcal{J}_N(x) = +1 \right\}$$

很容易证明下面的命题（参见图 11.1）。

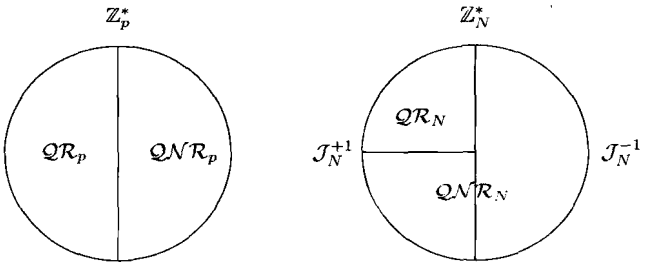


图 11.1 $\mathbb{Z}_p^*$和$\mathbb{Z}_N^*$的结构

命题 11.8 设$N=pq$，p,q为不相同的奇素数，则：

(1) $\mathbb{Z}_N^*$中恰好一半元素属于$\mathcal{J}_N^{+1}$。

(2) $\mathcal{QR}_N$包含于$\mathcal{J}_N^{+1}$。

(3) $\mathcal{J}_N^{+1}$中恰好一半元素属于$\mathcal{QR}_N$（另一半属于$\mathcal{QNR}_N^{+1}$）。

证明：$\mathcal{J}_p(x)=\mathcal{J}_q(x)=+1$或$\mathcal{J}_p(x)=\mathcal{J}_q(x)=-1$时，有$\mathcal{J}_N(x)=+1$。同时(根据前面一节)，$\mathbb{Z}_p^*$中恰好一半元素雅可比符号为+1，另一半元素雅可比符号为-1（$\mathbb{Z}_q^*$中元素也是如此)。定义$\mathcal{J}_p^{+1}$，$\mathcal{J}_p^{-1}$，$\mathcal{J}_q^{+1}$和$\mathcal{J}_q^{-1}$，因而可得

$$\begin{aligned}|\mathcal{J}_N^{+1}| &= |\mathcal{J}_p^{+1}\times\mathcal{J}_q^{+1}|+|\mathcal{J}_p^{-1}\times\mathcal{J}_q^{-1}|\\ &= |\mathcal{J}_p^{+1}|\cdot|\mathcal{J}_q^{+1}|+|\mathcal{J}_p^{-1}|\cdot|\mathcal{J}_q^{-1}|\\ &= \frac{(p-1)}{2}\frac{(q-1)}{2}+\frac{(p-1)}{2}\frac{(q-1)}{2}=\frac{\phi(N)}{2}\end{aligned}$$

所以$|\mathcal{J}_N^{+1}|=|\mathbb{Z}_N^*|/2$，这就证明了$\mathbb{Z}_N^*$中一半元素属于$\mathcal{J}_N^{+1}$。

前面已经注意到关于模N的所有二次剩余的雅可比符号都为+1，这就证明了$\mathcal{QR}_N\subseteq\mathcal{J}_N^{+1}$。

因为当且仅当$\mathcal{J}_p(x)=\mathcal{J}_q(x)=+1$时，$x\in\mathcal{QR}_N$。可以得出

$$|\mathcal{QR}_N|=|\mathcal{J}_p^{+1}\times\mathcal{J}_q^{+1}|=\frac{(p-1)}{2}\frac{(q-1)}{2}=\frac{\phi(N)}{4}$$

所以$|\mathcal{QR}_N|=|\mathcal{J}_N^{+1}|/2$。因为$\mathcal{QR}_N$是$\mathcal{J}_N^{+1}$的一个子集，这就证明了$\mathcal{J}_N^{+1}$中一半的元素属于$\mathcal{QR}_N$。■

下面的两个结论和命题 11.4 与推论 11.5 相似。

命题 11.9 设$N=pq$为两个不相等奇素数的乘积，且$x,y\in\mathbb{Z}_N^*$，则有$\mathcal{J}_N(xy)=\mathcal{J}_N(x)\cdot\mathcal{J}_N(y)$。

证明 利用定义$\mathcal{J}_N(\cdot)$和命题 11.4，可得

$$\begin{aligned}\mathcal{J}_N(xy)&=\mathcal{J}_p(xy)\cdot\mathcal{J}_q(xy)=\mathcal{J}_p(x)\cdot\mathcal{J}_p(y)\cdot\mathcal{J}_q(x)\cdot\mathcal{J}_q(y)\\ &=\mathcal{J}_p(x)\cdot\mathcal{J}_q(x)\cdot\mathcal{J}_p(y)\cdot\mathcal{J}_q(y)=\mathcal{J}_N(x)\cdot\mathcal{J}_N(y)\end{aligned}$$

■

推论 11.10 设$N=pq$为两个不同奇素数的乘积，且$x,x'\in\mathcal{QR}_N$，$y,y'\in\mathcal{QNR}_N^{+1}$。则有：

(1) $[xx' \bmod N]\in\mathcal{QR}_N$。

(2) $[yy' \bmod N] \in \mathcal{QR}_N$。
(3) $[xy \bmod N] \in \mathcal{QNR}_N^{+1}$。

证明 只证明最后一个，其他的证明与其类似。因为$x \in \mathcal{QR}_N$，可得$\mathcal{J}_p(x) = \mathcal{J}_q(x) = +1$。因为$y \in \mathcal{QNR}_N^{+1}$，可得$\mathcal{J}_p(y) = \mathcal{J}_q(y) = -1$。利用定义 11.4。

$$\mathcal{J}_p(xy) = \mathcal{J}_p(x) \cdot \mathcal{J}_p(y) = -1 \text{和} \mathcal{J}_q(xy) = \mathcal{J}_q(x) \cdot \mathcal{J}_q(y) = -1$$

所以$\mathcal{J}_N(xy) = +1$。但是xy不是一个关于模N的二次剩余，因为$\mathcal{J}_p(xy) = -1$，所以$[xy \bmod p]$不是模p的二次剩余。所以可以得出$xy \in \mathcal{QNR}_N^{+1}$。 ■

与推论 11.5 相比，$y, y' \in \mathcal{QNR}_N$并不意味着$yy' \in \mathcal{QR}_N$。（反而，如推论中所示的，只有$y, y' \in \mathcal{QNR}_N^{+1}$时才成立。）例如，可能有$\mathcal{J}_p(y) = +1$，$\mathcal{J}_q(y) = -1$和$\mathcal{J}_p(y') = -1$，$\mathcal{J}_q(y') = +1$，所以$\mathcal{J}_p(yy') = \mathcal{J}_q(yy') = -1$，此时即使$\mathcal{J}_N(yy') = +1$，$yy'$也不是一个二次剩余。

11.1.3 二次剩余假设

在 11.1.1 节中，介绍了判定一个给定输入x是否为素数模p的二次剩余的一个高效算法。怎样改变此算法使其在模N为合数时也可以工作？命题 11.6 给出一个简单的解决方案，用于当N的分解形式已知的情况。这个方案以算法的形式在下面算法 11.11 中给出。

算法 11.11
已知合数的分解形式，判定某数是否为该合数模的二次剩余。
输入：合数$N = pq$；因数p和q；元素$x \in \mathbb{Z}_N^*$
输出：一个判定结果，即是否$x \in \mathcal{QR}_N$
计算 $\mathcal{J}_p(x)$和$\mathcal{J}_q(x)$
if $\mathcal{J}_p(x) = \mathcal{J}_q(x) = +1$，**return** “二次剩余”
else return “二次非剩余”

通常情况下，假定N的分解形式为不同的奇素数。对上面的算法作一个简单的修正，可以在已知N的分解形式时计算$\mathcal{J}_N(x)$。

当N的分解形式未知时，现在还没有多项式时间算法可判定x是否为模N的二次剩余。有点吃惊的是，已知有一个多项式时间算法在未知N分解形式的情况下可以计算$\mathcal{J}_N(x)$。（尽管算法本身不是很复杂，但是算法的正确性证明超出了本书的范围，所以本书不介绍这个算法，有兴趣的读者可以查阅本章后的参考文献。）这导致二次剩余的部分测试：对于给定输入x，如果$\mathcal{J}_N(x) = -1$，则x不可能是二次剩余。（参见命题 11.8。）这个测试没有提及$\mathcal{J}_N(x) = +1$的情况，且普遍认为不存在多项式时间算法在此种情况下能够判定二次剩余（这个方法比随机猜测算法效果更好）。

现在正式提出这个假设。设GenModulus是一个多项式时间算法，输入为1^n，输出为(N, p, q)，其中$N = pq$且p, q为n比特素数，例外的情况为关于n的可忽略概率。

定义 11.12 如果对于所有概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\left|\Pr[\mathcal{A}(N,\mathsf{qr})=1]-\Pr[\mathcal{A}(N,\mathsf{qnr})=1]\right|\leqslant \mathsf{negl}(n)$$

则判定性二次剩余问题与GenModulus相关是困难的。其中每种情况下的概率来源于实验：运行GenModulus(1^n)得到(N,p,q)，qr从$\mathcal{QR}_N$中随机选择，qnr从$\mathcal{QNR}_N^{+1}$中随机选择。

上面介绍中至关重要的一点是：qnr是从$\mathcal{QNR}_N^{+1}$中选择，而不是$\mathcal{QNR}_N$中选择。如果qnr从$\mathcal{QNR}_N$中选择，则2/3的概率可能出现$\mathcal{J}_N(x)=-1$，导致从随机二次剩余中区分qnr会很简单。（前面曾经提到，即使不知道N分解的形式，也可以高效计算$\mathcal{J}_N(x)$。）

二次剩余假设可以简单描述为：存在一个GenModulus相关的判定性二次剩余问题为困难问题。易知，如果判定性二次剩余问题与GenModulus相关是困难的，则因子分解问题与GenModulus相关也是困难的。

11.1.4 Goldwasser-Micali 加密方案

根据前面一节的结果，可立刻得到一个基于二次剩余假设的用于单比特信息加密的公钥加密算法：

(1) 公钥为模N，私钥为N的分解形式。

(2) 位“0”的加密是一个随机二次剩余，位“1”的 0 加密是一个二次非剩余且雅可比符号为+1。

(3) 接收者可以凭借自己的私钥，利用N的分解形式判定c是否为二次剩余，从而解密一个密文c。

从定义 10.3 上来说，这个方案的安全性来自于定义 11.12 中正式陈述的二次剩余问题的困难性。

上面的描述中缺少发送者在不知道N分解形式的情况下，如何随机选择一个$\mathcal{QR}_N$中的元素（当想要对“0”进行加密时）或者随机选择一个$\mathcal{QNR}_N^{+1}$中的元素（当想要对“1”进行加密时）。前者很容易实现，但是后者的实现需要一些创造性。

选择随机二次剩余。选择一个随机元素$y\in\mathcal{QR}_N$很简单：简单挑选一个随机$x\leftarrow\mathbb{Z}_N^*$（参考附录B.2.4）且设置$y:=x^2 \bmod N$。很显然$y\in\mathcal{QR}_N$。事实上$y$是$\mathcal{QR}_N$中的一个均匀分布：因为平方模$N$是一个 4 对 1 函数（参见 11.1.2 节），且x从$\mathbb{Z}_N^*$中随机选择。更详细地，任意选定$\hat{y}\in\mathcal{QR}_N$，且在上面的过程后计算$y=\hat{y}$的概率。定义$\hat{y}$的四个平方根为$\pm\hat{x}$，$\pm\hat{x}'$，则

$$\begin{aligned}\Pr[y=\hat{y}]&=\Pr[x\text{ 是 }\hat{y}\text{ 的平方根}]\\&=\Pr[x\in\{\pm\hat{x},\pm\hat{x}'\}]\\&=\frac{4}{|\mathbb{Z}_N^*|}=\frac{1}{|\mathcal{QR}_N|}\end{aligned}$$

因为对任意$\hat{y}\in\mathcal{QR}_N$时，上面等式都成立，所以y是$\mathcal{QR}_N$中的一个均匀分布。

选择$\mathcal{QNR}_N^{+1}$中的随机元素。一般情况下，在不知道N的分解形式时，并不知道如何选择一个$\mathcal{QNR}_N^{+1}$中的随机元素。产生密钥的接收者可以有办法。具体地说，修改上面描述的方案以至于接受者另外选择一个随机数$z\leftarrow\mathcal{QNR}_N^{+1}$，且将$z$作为自己公钥的一部分。（因为已知$N$的分解形式，所以接受者很容易做到，参考练习 11.3。）发送者可以通过随机选择$x\leftarrow\mathbb{Z}_N^*$（像前面一样）和设定$y=[z\cdot x^2 \bmod N]$来实现随机选择一个元素$y\leftarrow\mathcal{QNR}_N^{+1}$。理由是根据

推论 11.10，有$y \in \mathcal{QNR}_N^{+1}$，$y$是均匀分布在$\mathcal{QNR}_N^{+1}$中的证明留着练习，下面安全性的证明中并不会直接用到。

构造方法 11.13 给出Goldwasser-Micali加密方案的一个完整描述，实现了上面的思路。

构造方法 11.13

设GenModulus是一个多项式时间算法，输入为1^n，输出为(N,p,q)，其中$N=pq$且p,q为n比特素数，异常的概率是关于n的可忽视概率。下面构造一个公钥加密方案：

- Gen：输入1^n，运行GenModulus(1^n)得到(N,p,q)，选择一个随机$z \leftarrow \mathcal{QNR}_N^{+1}$。公钥为$pk=\langle N,z\rangle$，私钥为$sk=\langle p,q\rangle$。
- Enc：输入一个公钥$pk=\langle N,z\rangle$和一个信息$m \in \{0,1\}$，选择一个随机$x \leftarrow \mathbb{Z}_N^*$，最后输出密文$c:=[z^m \cdot x^2 \bmod N]$。
- Dec：输入一个私钥$sk=\langle p,q\rangle$和一个密文c，判断c是否为模N的一个二次剩余，例如利用算法 11.11。如果c是一个二次剩余，输出 0；否则，输出 1。

Goldwasser-Micali加密方案

定理 11.14 如果二次剩余问题与GenModulus相关是困难的，则Goldwasser-Micali加密方案在选择明文攻击时为不可区分加密。

证明 用Π表示Goldwasser-Micali加密方案。现在证明Π在窃听者面前具有不可区分加密性。由定理 10.10 可知，这意味着其为CPA安全。

设$\mathcal{A}$是一个概率多项式时间敌手，并且定义

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]$$

考虑下面的PPT敌手D，其试图解决和GenModulus相关的二次剩余问题。

算法D：

算法给定N和z作为输入，目的是确定$z \in \mathcal{QR}_N$或$z \in \mathcal{QNR}_N^{+1}$。

(1) 设定$pk=\langle N,z\rangle$且运行$\mathcal{A}(pk)$得到两个单比特信息m_0和m_1。

(2) 选择一个随机位b和一个随机$x \leftarrow \mathbb{Z}_N^*$，然后设定$c=[z^{m_b} \cdot x^2 \bmod N]$。

(3) 将密文c给$\mathcal{A}$，得到一个输出位b'。如果$b'=b$，输出为 1；否则，输出为 0。

现在分析算法D的性能。有两种情况需要考虑：

情形 1。 D的输入由运行GenModulus(1^n)得到(N,p,q)，然后选择一个随机$z \leftarrow \mathcal{QNR}_N^{+1}$。则$D$运行$\mathcal{A}$,使用的公钥与$\Pi$相同，这种情况下，当$\mathcal{A}$作为$D$的一个子程序运行时的视图分布，与$\mathcal{A}$在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中运行时的视图分布是相同的。因为$D$输出 1 恰好是$\mathcal{A}$输出$b'=b$，可以得到

$$\Pr[D(N,\mathsf{qnr})=1]=\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]=\varepsilon(n)$$

其中qnr代表定义 11.12 中属于$\mathcal{QNR}_N^{+1}$的一个随机元素。

情形 2。 D的输入由运行GenModulus(1^n)得到(N,p,q)而产生，然后选择一个随机$z \leftarrow \mathcal{QNR}_N$。下面证明此种情况下的$\mathcal{A}$的分布与比特位$b$独立。注意：不管是加密 0 还是 1，交给$\mathcal{A}$的密文$c$是一个随机二次剩余：

（1）当加密 0 时，对随机$x \leftarrow \mathbb{Z}_N^*$，$c = [x^2 \bmod N]$，且$c$是一个随机二次剩余。

（2）当加密 1 时，对随机$x \leftarrow \mathbb{Z}_N^*$，$c = [z \cdot x^2 \bmod N]$。设$\hat{x} \stackrel{\text{def}}{=} [x^2 \bmod N]$，并且注意到$\hat{x}$是群$\mathcal{QR}_N$中一个均匀分布的元素。因为$z \in \mathcal{QR}_N$，可以应用引理 10.20 得出$c$也是均匀分布在$\mathcal{QR}_N$中。

因为$\mathcal{A}$的分布与比特b独立，这种情况下$b' = b$的概率恰好为$\frac{1}{2}$，也就是

$$\Pr[D(N, \mathsf{qr}) = 1] = \frac{1}{2}$$

其中qr表示$\mathcal{QR}_N$的一个随机元素，如定义 11.12。因此，

$$\left|\Pr[D(N, \mathsf{qr}) = 1] - \Pr[D(N, \mathsf{qnr}) = 1]\right| = \left|\varepsilon(n) - \frac{1}{2}\right|$$

由于假设二次剩余问题关于GenModulus是困难的，则存在一个可忽略函数negl，满足

$$\left|\varepsilon(n) - \frac{1}{2}\right| \leqslant \mathsf{negl}(n)$$

所以$\varepsilon(n) \leqslant \frac{1}{2} + \mathsf{negl}(n)$，得证。 ■

11.2 Rabin 加密方案

Rabin加密方案的安全性基于这一事实：在已知N的分解形式的情况下，计算关于合数模N的平方根是简单的，但是不知道N的分解形式时，计算关于合数模N的平方根是困难的。事实上，正如所看到的，计算关于模N的平方根等同于（也就是说困难性等同于）分解N。换句话说，分解假设蕴涵着计算合数模（适当生成的）平方根的困难性。由于这个等价关系，此版本的Rabin加密方案仅基于分解问题是困难的假设就可以证明其为CPA安全。所以至少从理论的观点上看，Rabin加密方案很有吸引力。对RSA加密而言，目前并不知道是否有类似结论，且RSA问题有潜在的可能性比分解问题要简单。Goldwasser-Micali加密方案也是如此，在不用分解N的情况下有可能判定模N的二次剩余。

有趣的是，Rabin加密方案与RSA加密方案（至少从表面上看）非常相似，但是其优势是基于一个更弱的假设。事实上RSA加密方案比前者应用更加广泛，似乎是由于历史因素而不是技术因素，这一点在本节末进行讨论。

11.2.1 计算模平方根

Rabin加密方案需要接收者计算模平方根，所以在这一节将探索这个问题的算法复杂性。首先介绍一个计算素数模p的平方根的高效算法，然后扩展这个算法到能计算合数模N的平方根的情形，这里已知合数模N的分解形式。读者如果接受这些算法的存在，可以直接跳到下一节，那里介绍在不知道合数模N的分解形式的情况下，求模N的平方根等价于分解N。

设p是一个奇素数。当$p = 3 \bmod 4$时计算模p平方根是相对简单的，但是当$p = 1 \bmod 4$时比较复杂。（简单的情况正是在Rabin加密算法中所需要的，在 11.2.3 节中将要介绍。为了完整性，这里也包括第二种情况的介绍。）两种情况下，将说明如何计算二次剩余$a \in \mathbb{Z}_p^*$的一个平方根。注意如果x是a的一个平方根，则$[-x \bmod p]$是另外一个。

首先处理简单的情形。$p = 3 \bmod 4$意味着对于某些整数i可以写作$p = 4i + 3$。因为$a \in \mathbb{Z}_p^*$是一个二次剩余，所以$\mathcal{J}_p(a) = 1 = a^{\frac{p-1}{2}} \bmod p$（参考命题 11.2）。在两边同时乘以$a$

可得

$$a = a^{\frac{p-1}{2}+1} = a^{2i+2} = (a^{i+1})^2 \bmod p$$

所以$a^{i+1} = a^{\frac{p+1}{4}} \bmod p$为$a$的一个平方根。换言之，计算$a$模$p$的一个平方根的方法是$x := [a^{\frac{p+1}{4}} \bmod p]$。

上面的介绍中关键点是$(p+1)/2$为偶数，因为这样确保了$(p+1)/4$是一个整数（为了让$a^{\frac{p+1}{4}} \bmod p$定义有意义，这是必要的；前面曾经提过指数必须为整数）。当$p = 1 \bmod 4$时这个方法并不适用，此时$p+1$是一个不能被4整除的整数。

当$p = 1 \bmod 4$时和上面的处理稍微有点不同。受上面方法的启发，设法寻找一个满足$a^r = 1 \bmod p$的奇数r。然后，按照上面的方法，$a^{r+1} = a \bmod p$且$a^{\frac{r+1}{2}} \bmod p$是当$(r+1)/2$为整数时$a$的一个平方根。尽管无法做到这一点，但可以做一些等同的事情：找到一个奇数r以及一个元素$b \in \mathbb{Z}_p^*$和一个偶数r'，它们满足

$$a^r \cdot b^{r'} = 1 \bmod p。$$

则$a^{r+1} \cdot b^{r'} = a \bmod p$，且$a^{\frac{r+1}{2}} \cdot b^{\frac{r'}{2}} \bmod p$为$a$的一个平方根（指数$(r+1)/2$和$r'/2$都是整数）。

例 11.15 取$p = 29$和$a = 22$，则$22^7 \cdot 2^{14} = 1 \bmod 29$，所以$15 = 22^{(7+1)/2} \cdot 2^{14/2} = 22^4 \cdot 2^7 \bmod 29$为22关于29取模的一个平方根。当然，另外一个平方根为$-15 = 14 \bmod 29$。

现在描述一般方法来寻找满足规定属性的r, b和r'。设$\frac{p-1}{2} = 2^l \cdot m$其中$l, m$为整数，$l \geqslant 1$且$m$为奇数[②]。因为$a$是一个二次剩余，可知

$$a^{2^l m} = a^{\frac{p-1}{2}} = 1 \bmod p \tag{11.3}$$

这意味着$a^{2^l m/2} = a^{2^{l-1} m} \bmod p$是1的一个平方根。1模$p$的平方根为$\pm 1 \bmod p$，所以可知$a^{2^{l-1} m} = \pm 1 \bmod p$。如果$a^{2^{l-1} m} = 1 \bmod p$，则出现了方程（11.3）中相同的情况，除了此时$a$的幂指数可以被一个更小的2的幂整除之外。这朝着正确的方向在推进：如果可以知道何时a的幂不能被任何2的幂整除（如出现$l = 1$的情况），则a的幂指数是一个奇数并且可以按照前面介绍的方式计算一个平方根。此处先给一个例子，然后马上回来讨论如何处理当$a^{2^{l-1} m} = -1 \bmod p$的情况。

例 11.16 取$p = 29$和$a = 7$。因为7是模29的一个二次剩余，有$7^{14} \bmod 29 = 1$并且$7^7 \bmod 29$为1的一个平方根。事实上，$7^7 = 1 \bmod 29$，幂指数7为奇数。所以$7^{(7+1)/2} = 7^4 = 23 \bmod 29$是7关于29取模的一个平方根。

总之，从$a^{2^l m} = 1 \bmod p$开始，并且去掉a的幂指数中2的因子，直到下面两种情况之一发生：对于某些$l' < l$，有$a^m = 1 \bmod p$或$a^{2^{l'} m} = -1 \bmod p$。在第一种情况中，因为$m$为奇数，可以按照例11.16那样很快计算出$a$的一个平方根。在第二种情况中，将方程两边同时乘以$-1 \bmod p$，使方程右边“恢复”为+1。然而，根据开始讨论这个问题时的动机，希望通过在方程左边乘以某些元素b而使其成为一个偶数幂来达到此目的。如果有一个可用的二次非剩余$b \in \mathbb{Z}_p^*$将很简单：因为$b^{2^l m} = b^{\frac{p-1}{2}} = -1 \bmod p$可得

② 整数l和m可以很容易通过取出$(p-1)/2$中因子2来计算得到。

$$a^{2^{l'}m} \cdot b^{2^l m} = (-1)(-1) = +1 \bmod p$$

现在可以按照前面的方式来进行，整个左边取平方根，从a的指数中约减 2 的最大幂，再乘以$b^{2^l m}$（需要的），所以式子右边一直为+1。观察到：b的指数中可被 2 的幂整除，且幂次总是比a的指数整除 2 的幂的幂次大。（所以事实上可以对两边指数中除以 2 得到平方根）。继续这个过程直到a的指数为奇数，可以按照前面描述的方式计算a的平方根。这个算法的伪代码在 11.17 中介绍。可以验证：给定一个二次非剩余b时，算法运行时间为多项式时间。

算法 11.17
计算素数模平方根
输入：素数p；平方剩余$a \in \mathbb{Z}_p^*$
输出：a的一个平方根
case $p = 3 \bmod 4$:
return $[a^{\frac{p+1}{4}} \bmod p]$
case $p = 1 \bmod 4$:
设b为一个模p的平方非剩余
compute 满足$2^l \cdot m = \frac{p-1}{2}$的$l$和$m$
$r := 2^l \cdot m, r' := 0$
for $i = l$ to 1{
/ * maintain the invariant $a^r \cdot b^{r'} = 1 \bmod p$ * /
$r := r/2, r' := r'/2$
if $a^r \cdot b^{r'} = -1 \bmod p$
$r' := r' + 2^l \cdot m$
}
/ * now $r = m, r'$ is even, and $a^r \cdot b^{r'} = 1 \bmod p$ * /
return $\left[a^{\frac{r+1}{2}} \cdot b^{\frac{r'}{2}} \bmod p\right]$

前面还没有提到是如何找到b。事实上，还没有确定的多项式时间算法可以找到模p的二次非剩余。幸运的是，很容易以一定的概率找到一个二次非剩余：简单地从$\mathbb{Z}_p^*$中随机选择元素直到找到一个二次非剩余。这样做起作用是因为$\mathbb{Z}_p^*$中恰好一半的元素为二次非剩余，并且已经知道判定素数模p二次剩余的多项式时间算法（参考 11.1.1 节中对这些陈述的证明）。这意味着已给出的算法当$p = 1 \bmod 4$时实际上为随机算法；这种情况下计算平方根的确定的多项式时间算法目前还没有。

例 11.18 此处考虑“最坏的情况”，也就是取平方根总得到-1。设要计算$a \in \mathbb{Z}_p^*$的平方根；设$b \in \mathbb{Z}_p^*$是一个二次非剩余；设$\frac{p-1}{2} = 2^3 \cdot m$，其中$m$为奇数。

第一步，有$a^{2^3 m} = 1 \bmod p$。因为$a^{2^3 m} = \left(a^{2^2 m}\right)^2$且 1 的平方根为$\pm 1$，这意味着$a^{2^2 m} = \pm 1 \bmod p$；假设最坏的情况，$a^{2^2 m} = -1 \bmod p$，等式乘以$b^{\frac{p-1}{2}} = b^{2^3 m} = -1 \bmod p$，可得

$$a^{2^2 m} \cdot b^{2^3 m} = 1 \bmod p$$

第二步，观察到$a^{2m} \cdot b^{2^2 m}$为 1 的一个平方根；仍然假设最坏情况，因而可得$a^{2m} \cdot b^{2^2 m} = -1 \bmod p$。等式乘以$b^{2^3 m}$来“纠正”这个结果

$$a^{2m} \cdot b^{2^2 m} \cdot b^{2^3 m} = 1 \bmod p。$$

第三步，选取平方根且假设为最坏情况（和上面类似），得到$a^m \cdot b^{2m} \cdot b^{2^2 m} = -1 \bmod p$。等式乘以“纠正因子”$b^{2^3 m}$可以得到

$$a^m \cdot b^{2m} \cdot b^{2^2 m} \cdot b^{2^3 m} = 1 \bmod p$$

此时正是所希望的情形。为了得到最后的算法，在等式两边同时乘以a得到

$$a^{m+1} \cdot b^{2m+2^2 m+2^3 m} = a \bmod p$$

因为m为奇数，所以$(m+1)/2$为整数且$a^{\frac{m+1}{2}} \cdot b^{m+2m+2^2 m} \bmod p$为$a$的一个平方根。

例 11.19 下面解决一个具体的例子。设$p = 17, a = 4$且$b = 3$。注意到此时$(p-1)/2 = 2^3$且$m = 1$。

首先有$4^{2^3} = 1 \bmod 17$，所以4^{2^2}应该等于$\pm 1 \bmod 17$。通过计算，发现$4^{2^2} = 1 \bmod 17$且在这一步中并不需要纠正项。

接下来，知道4^2是 1 的一个平方根，所以一定等于$\pm 1 \bmod 17$。计算得到$4^2 = -1 \bmod 17$。等式乘以3^{2^3}得到$4^2 \cdot 3^{2^3} = 1 \bmod 17$。

最后，考虑$4 \cdot 3^{2^2} = 1 \bmod 17$。此时几乎已经完成：在等式两边同时乘以 4 得到$4^2 \cdot 3^{2^2} = 4 \bmod 17$，所以$4 \cdot 3^2 = 2 \bmod 17$为 4 的一个平方根。

计算模N平方根。不难发现上面介绍的计算素数模平方根的算法，容易扩展到计算合数模N平方根的情形，这里合数$N = pq$的分解形式是已知的。具体地，由中国剩余定理，设$a \in \mathbb{Z}_N^*$是一个二次剩余且满足$a \leftrightarrow (a_p, a_q)$。分别计算$a_p, a_q$模$p, q$的平方根$x_p, x_q$，得到$a$的一个平方根$(x_p, x_q)$（参考 11.1.2）。给定$x_p$和$x_q$，对应于$(x_p, x_q)$的$x$可以按照 7.1.5 节中讨论的方法得到。明确地写出这些步骤：计算a关于模N的一个平方根，其中$N = pq$的分解形式已知：

(1) 计算$a_p := [a \bmod p]$和$a_q := [a \bmod q]$。

(2) 利用算法 11.17，计算a_p关于模p的一个平方根x_p和a_q关于模q的一个平方根x_q。

(3) 反过来将表示法$(x_p, x_q) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$转化为$x \in \mathbb{Z}_N^*$满足$x \leftrightarrow (x_p, x_q)$。输出$x$，其为$a$对$N$取模的一个平方根。

很容易修改算法使其返回a的所有 4 个平方根。

11.2.2 基于分解的陷门置换

前面已经看到在已知N的分解形式时，计算模N的平方根可以在多项式时间内完成。不同的是，此处介绍在N的分解形式未知时，计算合数模N的平方根与分解N一样困难。

正式地，设GenModulus是一个多项式时间算法，输入为1^n，输出为(N, p, q)，其中$N = pq$且p, q为n比特素数，例外的情况为关于n的可忽略概率。给定算法$\mathcal{A}$和参数n，考虑下面的实验：

平方根计算实验$\mathsf{SQR}_{\mathcal{A},\mathsf{GenModulus}}(n)$：

(1) 运行GenModulus(1^n)得到输出N, p, q。

(2) 选择$y \leftarrow \mathcal{QR}_N$。

(3) 给定(N, y)，$\mathcal{A}$输出$x \in \mathbb{Z}_N^*$。

(4) 如果$x^2 = y \bmod N$，实验的输出定义为1，否则为0。

定义 11.20 如果对于任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl,满足

$$\Pr[\mathsf{SQR}_{\mathcal{A},\mathsf{GenModulus}}(n) = 1] \leqslant \mathsf{negl}(n)$$

则计算平方根与GenModulus相关是困难的。

很容易发现：如果计算平方根与GenModulus相关是困难的，则因子分解必然与GenModulus相关是困难的：如果GenModulus输出的模N很容易分解，则很容易计算模N平方根。即首先分解N，然后应用前面节中介绍的算法。现在的目的是要说明逆命题：如果因子分解与GenModulus相关是困难的，则计算平方根问题也是如此。再次强调：对于RSA问题或者判定性二次剩余问题并不知道有如此结果。

下面的引理是关键，它介绍$\mathbb{Z}_N^*$中任意元素的两个“不相关”的平方根可以用于分解N。

引理 11.21 设$N = pq$，p, q为不相等的奇素数。给定$x, \hat{x}$，其满足$x^2 = y = \hat{x}^2 \bmod N$，但是$x \neq \pm\hat{x} \bmod N$，可以在以$||N||$表示的多项式时间内分解$N$。

证明 这里声称要么$\gcd(N, x + \hat{x})$或者$\gcd(N, x - \hat{x})$等于N的一个素数因子[③]。因为gcd可以在多项式时间内计算（参考附录B.1.2），从而证明了引理。

如果$x^2 = \hat{x}^2 \bmod N$，则$0 = x^2 - \hat{x}^2 = (x - \hat{x}) \cdot (x + \hat{x}) \bmod N$，所以$N \mid (x - \hat{x})(x + \hat{x})$。则$p \mid (x - \hat{x})(x + \hat{x})$，所以$p$整除其中一项。不妨令$p \mid (x + \hat{x})$（$p \mid (x - \hat{x})$的证明与其相似）。如果$q \mid (x + \hat{x})$，则$N \mid (x + \hat{x})$，但是又因为$x \neq -\hat{x} \bmod N$，这种情况不可能发生。所以$q \nmid x + \hat{x}$且$\gcd(N, x + \hat{x}) = p$。 ■

上面引理的另一种证明的方式着眼于中国剩余定理表示法。假定$x \leftrightarrow (x_p, x_q)$。因为$x$和$\hat{x}$为相同值$y$的平方根，易知$\hat{x}$对应于$(-x_p, x_q)$或$(x_p, -x_q)$。（它不可能对应于$(x_p, x_q)$或$(-x_p, -x_q)$，因为前者对应于$x$，而后者对应于$[-x \bmod N]$，两者的可能性被引理的假设排除。）不妨设$\hat{x} \leftrightarrow (-x_p, x_q)$，则

$$[x + \hat{x} \bmod N] \leftrightarrow (x_p, x_q) + (-x_p, x_q) = (0, [2x_q \bmod q])$$

并且$x + \hat{x} = 0 \bmod p$，而$x + \hat{x} \neq 0 \bmod q$。由此得到$\gcd(N, x + \hat{x}) = p$，为$N$的一个因子。

现在，证明本节的主要结果。

定理 11.22 如果因子分解与GenModulus相关是困难的，则计算模平方根与GenModulus相关是困难的。

证明 设$\mathcal{A}$为一个概率多项式时间算法，并且定义

$$\varepsilon(n) \stackrel{\mathsf{def}}{=} \Pr[\mathsf{SQR}_{\mathcal{A},\mathsf{GenModulus}}(n) = 1]$$

考虑下面的概率多项式时间算法$\mathcal{A}_{\mathsf{fact}}$，其试图分解由GenModulus输出的模：

---

③ 事实上，两者都等于N的一个素因子，很容易证明这里所声称的。

算法$\mathcal{A}_{\mathsf{fact}}$：

算法给定一个模N作为输入。

(1) 随机选择$x \leftarrow \mathbb{Z}_N^*$并且计算$y = [x^2 \bmod N]$。

(2) 运行$\mathcal{A}(N, y)$得到输出$\hat{x}$。

(3) 如果$\hat{x}^2 = y \bmod N$并且$\hat{x} \neq \pm x \bmod N$，则利用引理 11.21 分解$N$。

通过引理 11.21，可以知道当$\hat{x} \neq \pm x \bmod N$并且$\hat{x}^2 = y \bmod N$时，$\mathcal{A}_{\mathsf{fact}}$可以成功分解$N$。即

$$\begin{aligned}&\Pr[\mathsf{Factor}_{\mathcal{A}_{\mathsf{fact}},\mathsf{GenModuls}}(n) = 1]\\&= \Pr[\hat{x} \neq \pm x \bmod N \wedge \hat{x}^2 = y \bmod N]\\&= \Pr[\hat{x} \neq \pm x \bmod N \mid \hat{x}^2 = y \bmod N] \cdot \Pr[\hat{x}^2 = y \bmod N] \qquad (11.4)\end{aligned}$$

上面所有的概率均针对实验$\mathsf{Factor}_{\mathcal{A}_{\mathsf{fact}},\mathsf{GenModuls}}(n)$（参考 7.2.3 节中对此实验的描述）。在实验中，模N作为实验$\mathcal{A}_{\mathsf{fact}}$的输入由$\mathsf{GenModulus}(1^n)$产生，因为$x$随机均匀地从$\mathbb{Z}_N^*$中选择（参考 11.1.4 节），$y$是一个关于模$N$的随机二次剩余，则$\mathcal{A}$作为$\mathcal{A}_{\mathsf{fact}}$的一个子程序时的分布，与$\mathcal{A}$在实验$\mathsf{SQR}_{\mathcal{A},\mathsf{GenModulus}}(n)$中的分布完全相同。因此，

$$\Pr[\hat{x}^2 = y \bmod N] = \Pr[\mathsf{SQR}_{\mathcal{A},\mathsf{GenModulus}}(n) = 1] = \varepsilon(n) \qquad (11.5)$$

满足实验$\mathsf{Factor}_{\mathcal{A}_{\mathsf{fact}},\mathsf{GenModuls}}(n)$中的$y$的二次剩余前提条件，$x$的值等概率地等于$y$的四个可能的平方根中的任何一个。这意味着从算法$\mathcal{A}$的角度（$\mathcal{A}$正作为$\mathcal{A}_{\mathsf{fact}}$的一个子程序），$x$等概率地等于$y$的 4 个可能平方根中的任何一个。反过来意味着，满足$\mathcal{A}$输出某些y的平方根$\hat{x}$的前提条件，$\hat{x} = \pm x \bmod N$的概率恰好为$1/2$。（此处强调并没有做任何关于$\hat{x}$在y的平方根上如何分布的假设，特别的，并没有假设$\mathcal{A}$输出y的一个随机平方根，而用到的事实是：x均匀分布在y的平方根中。）也就是说，

$$\Pr[\hat{x} \neq \pm x \bmod N \mid \hat{x}^2 = y \bmod N] = \frac{1}{2} \qquad (11.6)$$

结合式（11.4）～式（11.6），可以看到

$$\Pr[\mathsf{Factor}_{\mathcal{A}_{\mathsf{fact}},\mathsf{GenModuls}}(n) = 1] = \frac{1}{2} \cdot \varepsilon(n)$$

因为因子分解与GenModulus相关是困难的，则存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Factor}_{\mathcal{A}_{\mathsf{fact}},\mathsf{GenModuls}}(n) = 1] \leqslant \mathsf{negl}(n)$$

这意味着$\varepsilon(n)/2 \leqslant \mathsf{negl}(n)$，即$\varepsilon \leqslant 2 \cdot \mathsf{negl}(n)$。从而完成了证明。 ■

前面的定理直接导致一个单向函数簇（参见定义 7.70），基于任何GenModulus，与之相关分解问题是困难的：

(1) 算法Gen，输入1^n，运行$\mathsf{GenModulus}(1^n)$得到(N, p, q)并且输出$I = N$。定义域$\mathcal{D}_I$为$\mathbb{Z}_N^*$并且值域$\mathcal{R}_I$为$\mathcal{QR}_N$。

(2) 算法Samp，输入N，选择一个随机元素$x \leftarrow \mathbb{Z}_N^*$。

(3) 算法f，输入N和$x \in \mathbb{Z}_N^*$，输出$[x^2 \bmod N]$。

此函数簇为单向函数这个事实来自于假设分解问题是困难的，以及来自于f求逆和因子分解是等价的这个事实。

使用特殊形式的模N，并且令$\mathcal{D}_I$为$\mathbb{Z}_N^*$的一个子集，可将这个函数簇转变成一个单向置换簇。（参见练习中介绍了另外一种使其成为置换的方式。）设$N=pq$是一个Blum整数：如果p和q为不相等的素数，且满足$p=q=3 \bmod 4$。构造一个置换的关键是下面的命题。

命题 11.23 设N是一个Blum整数。此时任何模N二次剩余都恰好有一个平方根也是二次剩余。

证明 假设$N=pq$满足$p=q=3 \bmod 4$。利用命题 11.2，可以发现-1不是一个模p或模q的二次剩余。这是因为$p=3 \bmod 4$对于某些整数i满足$p=4i+3$，所以

$$(-1)^{\frac{p-1}{2}}=(-1)^{2i+1}=-1 \bmod p$$

（因为$2i+1$是奇数）。现在，设$y \leftrightarrow (y_p, y_q)$是任意一个关于模$N$的二次剩余，且其有四个平方根为$(x_p, x_q)$，$(-x_p, x_q)$，$(x_p, -x_q)$，$(-x_p, -x_q)$。

下面声称恰好其中有一个为模N的二次剩余。为了说明这一点，假设$\mathcal{J}_p(x_p)=+1$和$\mathcal{J}_q(x_q)=-1$（其他情形的证明过程与其相似）。利用命题 11.4，可得

$$\mathcal{J}_q(-x_q)=\mathcal{J}_q(-1)\cdot\mathcal{J}_q(x_q)=+1,$$

所以$(x_p, -x_q)$对应到一个关于模N的二次剩余（利用命题 11.6）。同样地，$\mathcal{J}_p(-x_p)=-1$且y的任何其他平方根都不是模N的二次剩余。 ■

换一种方法表述，上面的命题表明：当N是一个Blum整数，由$f_N(x)=[x^2 \bmod N]$给出的函数$f_N: \mathcal{QR}_N \to \mathcal{QR}_N$是一个$\mathcal{QR}_N$上的置换。修改前面的抽样算法Samp，选择一个随机$x \leftarrow \mathcal{QR}_N$（正如已看到的，这可以通过选择随机$r \leftarrow \mathbb{Z}_N^*$和设定$x:=[r^2 \bmod N]$来容易地完成。）给出一个单向置换簇。最后，因为给定N的分解形式，模N平方根可以在多项式时间内计算，所以通过简单修改便可产生一个陷门置换簇，其是基于任何GenModulus，只要与之相关分解问题是困难的。这就称为Rabin陷门置换簇。总之：

定理 11.24 设GenModulus是一个输入为1^n，输出为(N,p,q)的算法，其中$N=pq$且p,q为不相等的素数（例外情况的概率是可忽略的），满足$p=q=3 \bmod 4$。如果分解问题与GenModulus相关是困难的，则存在一个陷门置换簇。

11.2.3 Rabin 加密方案

Rabin陷门置换意味着一种“教科书式Rabin”加密方案，与“教科书式RSA”加密方案相似。此加密方案为确定的，所以不是CPA安全的。一个更好的方法是依赖 10.7.2 节中的结果，此处介绍了任何陷门置换都可以用于构造一个CPA安全的公钥加密方案。把那里介绍的变换方法应用到前面一节中介绍的Rabin陷门置换簇，需要一个该簇的硬核谓词（参见定义10.26）。可以证明最低位$\mathsf{lsb}(\cdot)$组成了一个硬核谓词。也就是说，设GenModulus为一个输出Blum整数的算法，且与之相关分解问题是困难的。则对于任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr[\mathcal{A}(N,[x^2 \bmod N])=\mathsf{lsb}(x)] \leqslant \frac{1}{2}+\mathsf{negl}(n)$$

此处概率取自实验GenModulus(1^n)，其输出为(N,p,q)，并且x从$\mathcal{QR}_N$中随机选择。将此加入

构造方法 10.27 中可以得到一个公钥加密方案的具体例子，且此方案的安全性可以基于因子分解假设。构造方法 11.25 中将对此进行正式描述。

构造方法 11.25

设GenModulus是一个多项式时间算法，输入为1^n，输出为(N,p,q)，其中$N=pq$且p,q为n比特素数（出现例外的概率是关于n的可忽略概率），满足$p=q=3 \bmod 4$。下面构造一个公钥加密方案：

- Gen：输入1^n，运行GenModulus(1^n)得到(N,p,q)。公钥为N，并且私钥为$\langle p,q\rangle$。
- Enc：输入一个公钥N和信息$m\in\{0,1\}$，选择一个随机$x\leftarrow\mathcal{QR}_N$，最后输出密文$c:=\langle[x^2 \bmod N],\mathsf{lsb}(x)\oplus m\rangle$。
- Dec：输入一个私钥$\langle p,q\rangle$，和一个密文$\langle c,c'\rangle$，计算满足$x^2=c \bmod N$的唯一$x\in\mathcal{QR}_N$，并且输出$\mathsf{lsb}(x)\oplus c'$。

Rabin 加密方案

本节并不对下面的定理进行证明，参见第 10 章（及参考文献）的练习 10.10，可以得出一个如何进行证明的思路。

定理 11.26 如果分解困难与GenModulus相关是困难的，则构造方法 11.25 在选择明文攻击条件下是不可区分加密。

读者也可能会考虑“填充的Rabin”的加密方案，类似于 10.4.3 节中构造的填充RSA加密方案，其效率比构造方法 11.25 要高。但是在这种情况下，基于可证明分解问题的安全性并不知道是成立的。参见练习 11.14，介绍了一个实例说明此构造方法是如何完成的。

Rabin 加密方案与 RSA 加密方案的比较。讨论Rabin和RSA密码系统的相似点与不同点是很有意义的。（此处的讨论可以应用于任何以Rabin或RSA陷门置换为基础的密码学构造方法，而不一定非要是公钥加密方案。）

简而言之，RSA和Rabin陷门置换似乎相当相似，在Rabin中取平方的情况相当于在RSA中选择$e=2$。（当然，2 不与$\phi(N)$互素，所以Rabin不是RSA的一个特例。）在各自构造方法提供的安全性方面，注意到计算模平方根的难度等价于因子分解问题的难度，然而并不知道因子分解问题的困难性是否就意味着解决RSA问题的困难性。即Rabin陷门置换基于一个可能更弱的假设。开发出一个高效的算法能解决RSA问题但计算平方根仍然是困难的，这种情况在理论上是有可能的。似乎很合理的是，提出一个算法用比因子分解要短的时间解决RSA问题（但仍然是超多项式时间）。然而引理 11.21 确保计算模N平方根不可能比分解N的最佳可用算法还快。

在效率方面，RSA和Rabin置换基本上是相同的。事实上，如果一个大指数e应用到RSA中，则计算第e次幂（如在RSA中）要比求平方（如在Rabin中）稍微慢些。另外，使用Rabin置换时需要更多的注意，因为其仅仅是一个在$\mathbb{Z}_N^*$子集上的置换，而RSA的置换是在所有$\mathbb{Z}_N^*$上。

“教科书式Rabin”加密方案的构造方式和“教科书式RSA”的加密非常相似，易受到选择密文攻击，选择密文攻击使得敌手学习到接收者的整个私钥（参见练习 11.12）。尽管“教科书式RSA”加密也易受到选择明文攻击，从而导致泄漏整个信息，但是目前还不知道对“教科书式RSA”的选择密文攻击会导致整个私钥泄漏。可能存在对“教科书式Rabin”的此类攻击，在早些时候影响了密码设计者，以至于完全拒绝使用Rabin加密。

总之，RSA置换在实际中的应用要比Rabin置换广泛得多，但是根据前面的介绍，这看上去更多的是由于历史的偶然，而不是任何强迫性的技术上的考虑。

11.3 Paillier 加密方案

Paillier加密方案与RSA、Goldwasser-Micali、Rabin加密方案一样，是基于合数N分解的困难性，这里合数N为两个素数的乘积。（除了Rabin加密方案以外，其他这些加密方案的安全性并不知道是否与分解的困难性等价。）Paillier加密方案比Goldwasser-Micali密码系统要更加高效，对在定理 10.19 和定理 11.26 中各自介绍的证明安全的RSA和Rabin方案也是如此。可能更重要的是，Paillier加密方案拥有一些很好的同态属性，在 11.3.3 节中将进一步讨论。但是要注意的是，其依赖的是一个更新的且更少研究的困难性假设。

Paillier加密方案使用群$\mathbb{Z}_{N^2}^*$。下面的命题将给出该群一个很有用的特征，该命题陈述了：对于具有特定形式[④]的N，$\mathbb{Z}_{N^2}^*$与$\mathbb{Z}_N \times \mathbb{Z}_N^*$同态（参见定义 7.23）。本章将在下一节证明这个命题。

命题 11.27 设$N = pq$，其中p, q为值不相等，但长度相等的素数。则：

(1) $\gcd(N, \phi(N)) = 1$。

(2) 对于任意整数$a \geqslant 0$，可得$(1+N)^a = (1+aN) \bmod N^2$。因此，$\mathbb{Z}_{N^2}^*$中$(1+N)$的阶为$N$。也就是说，$(1+N)^N = 1 \bmod N^2$，且对于任何$1 \leqslant a \leqslant N$，$(1+N)^a \neq 1 \bmod N^2$。

(3) $\mathbb{Z}_N \times \mathbb{Z}_N^*$与$\mathbb{Z}_{N^2}^*$同态，同态$f : \mathbb{Z}_N \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^2}^*$由下面函数给出：

$$f(a,b) = [(1+N)^a \cdot b^N \bmod N^2]$$

根据上面命题的最后一点，介绍一些方便的速记法。给出熟悉的N和$x \in \mathbb{Z}_{N^2}^*$，$a \in \mathbb{Z}_N$，$b \in \mathbb{Z}_N^*$。如果$f(a,b) = x$，写作$x \leftrightarrow (a,b)$，其中f是上面命题中的一个同态。思考这个符号的一个方法为“x属于$\mathbb{Z}_{N^2}^*$对应于(a,b)属于$\mathbb{Z}_N \times \mathbb{Z}_N^*$”。在本书中由中国剩余定理给出同态$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$的内容中也使用了相同的符号，此处继续用这个符号是因为这两种情况都涉及到同态群。然而，因为群$\mathbb{Z}_{N^2}^*$和上面的命题只在本节中使用（并且本节没有用到中国剩余定理），所以应该不会造成混淆。注意此处同态（而不是其逆）即使在不知道N的分解形式的情况下仍可以高效计算。

11.3.1 节专门证明命题 11.27。如果读者愿意不加怀疑地接受这个命题可以直接跳至 11.3.2 节。

④ 前面曾经提到$\mathbb{Z}_N$为模N加法群，然而$\mathbb{Z}_N^*$为模N乘法群。

11.3.1 $\mathbb{Z}_{N^2}^*$结构

本节通过多个声称证明命题 11.27。设N, p, q和命题中保持一致。

声称 11.28 对于命题 11.27 中的N, p, q，$\gcd(N, \phi(N)) = 1$。

证明 前面有$\phi(N) = (p-1)(q-1)$。不失一般性，假设$p > q$，因为p是一个素数，且$p > p-1 > q-1$，很明显$\gcd(p, \phi(N)) = 1$。类似地，$\gcd(q, q-1) = 1$。此时，因为q是素数，如果$\gcd(q, p-1) \neq 1$则$\gcd(q, p-1) = q$。但是此时$(p-1)/q \geqslant 2$，与假设p和q的长度相同矛盾。 ■

声称 11.29 $a \geqslant 0$且是一个整数，可得$(1+N)^a = 1 + aN \bmod N^2$。因而$\mathbb{Z}_{N^2}^*$中$(1+N)$的阶为$N$。

证明 利用二项展开定理（定理A.1）：

$$(1+N)^a = \sum_{i=0}^{a} \binom{a}{i} N^i$$

对右边关于N^2取模进行化简，所有$i \geqslant 2$ 的项都变为 0，所以$(1+N)^a = 1 + aN \bmod N^2$。满足$(1+N)^a = 1 \bmod N^2$的最小非零$a$故为$a = N$。 ■

声称 11.30 $\mathbb{Z}_N \times \mathbb{Z}_N^*$与$\mathbb{Z}_{N^2}^*$同态，同态$f : \mathbb{Z}_N \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^2}^*$由$f(a,b) = [(1+N)^a \cdot b^N \bmod N^2]$给出。

证明 注意到：因为$\gcd((1+N), N^2) = 1$且$\gcd(b, N^2) = 1$（因为$b \in \mathbb{Z}_N^*$），则$(1+N)^a \cdot b^N$和N^2没有一个共同的因数，所以$[(1+N)^a \cdot b^N \bmod N^2]$属于$\mathbb{Z}_{N^2}^*$。现在证明$f$是一个同态。

首先说明f是一个双射。因为

$$\begin{aligned} |\mathbb{Z}_{N^2}^*| &= \phi(N^2) = p \cdot (p-1) \cdot q \cdot (q-1) \\ &= pq \cdot (p-1)(q-1) = |\mathbb{Z}_N| \cdot |\mathbb{Z}_N^*| = |\mathbb{Z}_N \times \mathbb{Z}_N^*| \end{aligned}$$

（第二个等式参见定理 7.19），以此足够证明f是一个单射。假设$a_1, a_2 \in \mathbb{Z}_N$和$b_1, b_2 \in \mathbb{Z}_N^*$满足$f(a_1, b_1) = f(a_2, b_2)$。则

$$(1+N)^{a_1 - a_2} \cdot (b_1/b_2)^N = 1 \bmod N^2 \tag{11.7}$$

（注意到$b_2 \in \mathbb{Z}_N^*$，因而$b_2 \in \mathbb{Z}_{N^2}^*$，所以$b_2$关于模$N^2$有一个乘法逆元。）在等式两边同时取$\phi(N)$幂次，并且利用$\mathbb{Z}_{N^2}^*$的阶为$\phi(N^2) = N \cdot \phi(N)$，可得

$$\begin{aligned} (1+N)^{(a_1-a_2)\cdot\phi(N)} \cdot (b_1/b_2)^{N\cdot\phi(N)} &= 1 \bmod N^2 \\ \Rightarrow (1+N)^{(a_1-a_2)\cdot\phi(N)} &= 1 \bmod N^2 \end{aligned}$$

由声称 11.29 可知，$(1+N)$关于模N^2的阶为N。应用命题 7.50 可知，$(a_1 - a_2) \cdot \phi(N) = 0 \bmod N$，所以$N$整除$(a_1 - a_2) \cdot \phi(N)$。因为由声称 11.28 可知，$\gcd(N, \phi(N)) = 1$，得到$N \mid (a_1 - a_2)$。因为$a_1, a_2 \in \mathbb{Z}_N$，只有当$a_1 = a_2$时才会出现这种情况。

回到等式（11.7），设定$a_1 = a_2$，因而有$b_1^N = b_2^N \bmod N^2$，这意味着$b_1^N = b_2^N \bmod N$。因为N与$\phi(N)$互素，$\mathbb{Z}_N^*$的阶的N次幂是$\mathbb{Z}_N^*$中的一个双射（参见推论 7.17），这意味着

$b_1 = b_2 \bmod N$。因为$b_1, b_2 \in \mathbb{Z}_N^*$，可得$b_1 = b_2$。因而我们断定$f$是一个单射，也为一个双射。

为了说明f是一个同态，证明$f(a_1, b_1) \cdot f(a_2, b_2) = f(a_1 + a_2, b_1 \cdot b_2)$。（注意到等式左边乘法关于$N^2$取模，但是等式右边加法或乘法关于$N$取模。）得到

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= ((1+N)^{a_1} \cdot b_1^N) \cdot ((1+N)^{a_2} \cdot b_2^N) \bmod N^2 \\ &= (1+N)^{a_1+a_2} \cdot (b_1 b_2)^N \bmod N^2 \end{aligned}$$

因为$(1+N)$关于模N^2的阶为N（由声称 11.29 可知），应用命题 7.49 可以得到

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= (1+N)^{a_1+a_2} \cdot (b_1 b_2)^N \bmod N^2 \\ &= (1+N)^{a_1+a_2 \bmod N} \cdot (b_1 b_2)^N \bmod N^2 \end{aligned} \tag{11.8}$$

此时还没有完成证明，因为$b_1 b_2$在等式（11.8）中代表模N^2乘，然而希望其为模N。设$b_1 b_2 = r + \gamma N$，其中γ, r为满足 $1 \leqslant r < N$的整数（因为$b_1, b_2 \in \mathbb{Z}_N^*$，$r$不能为 0，所以它们的乘积不能被$N$整除）。注意到$r = b_1 b_2 \bmod N$。也可以得到

$$\begin{aligned} (b_1 b_2)^N &= (r + \gamma N)^N \bmod N^2 \\ &= \sum_{k=0}^{N} \binom{N}{k} r^{N-k} (\gamma N)^k \bmod N^2 \\ &= r^N + N \cdot r^{N-1} \cdot (\gamma N) = r^N = (b_1 b_2 \bmod N)^N \bmod N^2 \end{aligned}$$

利用声称 11.29 中的二项式展开定理，将此加入到等式（11.8）中可以得到想要的结果：

$$\begin{aligned} f(a_1, b_1) \cdot f(a_2, b_2) &= (1+N)^{a_1+a_2 \bmod N} \cdot (b_1 b_2 \bmod N)^N \bmod N^2 \\ &= f(a_1 + a_2, b_1 b_2) \end{aligned}$$

从而证明了f是从$\mathbb{Z}_N \times \mathbb{Z}_N^*$到$\mathbb{Z}_{N^2}^*$的一个同态。 ■

11.3.2 Paillier 加密方案

设$N = pq$是两个值不相等但长度相等的素数的乘积。命题 11.27 说明$\mathbb{Z}_N \times \mathbb{Z}_N^*$是到$\mathbb{Z}_{N^2}^*$的同态，且同态函数为$f(a, b) = [(1+N)^a \cdot b^N \bmod N^2]$。得到一个结论为：随机元素$y \in \mathbb{Z}_{N^2}^*$对应于一个随机元素$(a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$。或者换句话说，随机选取$a \in \mathbb{Z}_N$，随机选取$b \in \mathbb{Z}_N^*$的元素$(a, b)$。

如果y是N次幂，则$y \in \mathbb{Z}_{N^2}^*$是模N^2的“N次方剩余”。也就是说，如果存在一个$x \in \mathbb{Z}_{N^2}^*$满足$y = x^N \bmod N^2$。将模N^2的N次方剩余的集合用$\mathsf{Res}(N^2)$表示。现在来描述$\mathbb{Z}_{N^2}^*$的N次方剩余的特征。选取满足$x \leftrightarrow (a, b)$的任何$x \in \mathbb{Z}_{N^2}^*$，并且求N次幂，得到

$$[x^N \bmod N^2] \leftrightarrow (a, b)^N = (N \cdot a \bmod N, b^N \bmod N) = (0, b^N \bmod N)$$

（前面曾经提到，$\mathbb{Z}_N \times \mathbb{Z}_N^*$中的群运算在第一部分为模$N$加，第二部分为模$N$乘）。此外，声称任何元素$y$满足$y \leftrightarrow (0, b)$为一个$N$次剩余。为了弄明白这一点，我们先回顾前面提到过，存在$\gcd(N, \phi(N)) = 1$和$d \stackrel{\text{def}}{=} [N^{-1} \bmod \phi(N)]$。所以对于任何$a \in \mathbb{Z}_N$，有

$$(a, |b^d \bmod N|)^N = (Na \bmod N, [b^{dN} \bmod N]) = (0, b) \leftrightarrow y$$

因此说明了$\mathsf{Res}(N^2)$对应于集合：

$$\{(0, b) \mid b \in \mathbb{Z}_N^*\}$$

（将其与$\mathbb{Z}_{N^2}^*$相比，在$\mathbb{Z}_{N^2}^*$中与$\{(a, b) \mid a \in \mathbb{Z}_N, b \in \mathbb{Z}_N^*\}$对应。）上面同时也证明了对任意$y \in \mathsf{Res}(N^2)$，其 N 次根的个数恰为 N，故计算 N 次幂是一个N对 1 函数。因此，如果$r \leftarrow \mathbb{Z}_{N^2}^*$随机均匀选择，则$[r^N \bmod N^2]$是$\mathsf{Res}(N^2)$中元素的一个均匀分布的元素。

判定合数剩余问题，粗略地说，就是从$\mathsf{Res}(N^2)$的随机元素中区分$\mathbb{Z}^*_{N^2}$中的一个随机元素。注意到并不像二次剩余假设，那里二次剩余中$\mathcal{QR}_N$和$\mathcal{QNR}_N^{+1}$为不相交的集合，此处$\mathsf{Res}(N^2) \subseteq \mathbb{Z}^*_{N^2}$。然而，$\mathsf{Res}(N^2)$仅占$\mathbb{Z}^*_{N^2}$中一个可忽略的部分。现在，设GenModulus是一多项式时间算法，输入为1^n，输出为(N,p,q)，其中$N=pq$且p,q为n比特素数（例外的概率为以n表示可忽视的概率）。那么有：

定义 11.31 如果对于任意概率多项式时间算法$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\left|\Pr[\mathcal{A}(N,[r^N \bmod N^2])=1]-\Pr[\mathcal{A}(N,r)=1]\right| \leqslant \mathsf{negl}(n)$$

则称判定性合数剩余问题与GenModulus相关是困难的。其中每种情况下的概率来自$\mathsf{GenModulus}(1^n)$输出(N,p,q)的实验，然后选择一个随机$r \leftarrow \mathbb{Z}^*_{N^2}$的随机性。（注意到$[r^N \bmod N^2]$是$\mathsf{Res}(N^2)$中的一个随机元素。）

判定性合数剩余（DCR）假设为，存在一个GenModulus，与之相关判定性合数剩余问题是困难的。这个假设可以视为前面已看到的$\mathbb{Z}^*_N$中的二次剩余假设的一般化。

正如前面讨论的，$\mathbb{Z}^*_{N^2}$中的随机元素有(r',r)的形式，其中r'和r是随机选择的（在合适的群中），然而随机N次剩余有$(0,r)$的形式，其中r是随机的。DCR假设是：从第二种类型的随机元素中区分第一种类型的随机元素是困难的。这预示着下面这个用公钥N加密消息$m \in \mathbb{Z}_N$的抽象方式：选择一个随机的N次剩余$(0,r)$，且设密文等于$c \leftrightarrow (m,1)\cdot(0,r)=(m,r)$（其中，“$\cdot$”代表$\mathbb{Z}_N \times \mathbb{Z}^*_N$中的群运算。）现在不用担心发送者如何高效实现这些，或者接收者如何解密，先简单地相信此方案是安全的。因为一个随机N次剩余$(0,r)$不能从随机元素(r',r)中区分出来，用上面方法构造的密文与以$c' \leftrightarrow (m,1)\cdot(r',r)=([m+r' \bmod N],r)$方法构造的密文是不可区分的（从窃听者的角度讨论，窃听者不知道N的分解方法，因而假定不能计算f^{-1}），其中随机选取$r' \in \mathbb{Z}_N$。引理 10.20 说明$[m+r' \bmod N]$是$\mathbb{Z}_N$中的均匀分布，所以，此密文r'与消息m独立。在偷听者在场的情况下不可区分加密满足。

下面的形式化证明遵循以上论述中的思路。先看看如何高效地加密和解密，然后给出加密方案的形式化描述。

加密。上面已经描述的加密好像发生在$\mathbb{Z}_N \times \mathbb{Z}^*_N$中。事实上它发生在同态群$\mathbb{Z}^*_{N^2}$中。也就是说，发送者通过选择一个随机$r \in \mathbb{Z}^*_N$并计算$c := [(1+N)^m \cdot r^N \bmod N^2]$来得到一个密文$c \in \mathbb{Z}^*_{N^2}$。

观察到

$$c=((1+N)^m \cdot 1^N)\cdot((1+N)^0 \cdot r^N) \bmod N^2 \leftrightarrow (m,1)\cdot(0,r)$$

所以$c \leftrightarrow (m,r)$,得证。

注意到：发送者选择随机的$r \leftarrow \mathbb{Z}^*_N$或随机地$r \leftarrow \mathbb{Z}^*_{N^2}$并没有任何不同，因为任意两种情况下$[r^N \bmod N^2]$的分布是相同的（可以通过观察同态群$\mathbb{Z}_N \times \mathbb{Z}^*_N$中发生了什么而得到验证）。

解密。现在描述给定N的分解形式，如何可以高效地实现解密。密文c按照上面介绍的方法生成，声称通过下面的步骤来解密得到m：

(1) 设置$\hat{c} := [c^{\phi(N)} \bmod N^2]$。

(2) 设置$\hat{m} := (\hat{c}-1)/N$。（注意此步是在整数范围执行。）

(3) 设置$m := [\hat{m} \cdot \phi(N)^{-1} \bmod N]$。

下面介绍此方法为何可以实现解密，对任意$r \in \mathbb{Z}_N^*$，令$c \leftrightarrow (m, r)$。于是

$$\begin{aligned}\hat{c} &\stackrel{\text{def}}{=} [c^{\phi(N)} \bmod N^2] \\ &\leftrightarrow (m, r)^{\phi(N)} \\ &= \left([m \cdot \phi(N) \bmod N], [r^{\phi(N)} \bmod N]\right) \\ &= ([m \cdot \phi(N) \bmod N], 1)\end{aligned}$$

运用命题 11.27（3），这意味着$\hat{c} = (1+N)^{[m \cdot \phi(N) \bmod N]} \bmod N^2$。利用命题 11.27（2），可以知道

$$\hat{c} = (1+N)^{[m \cdot \phi(N) \bmod N]} = (1 + [m \cdot \phi(N) \bmod N] \cdot N) \bmod N^2$$

因为$1 + [m \cdot \phi(N) \bmod N] \cdot N$通常比$N^2$小，我们可以去掉后面的模$N^2$，并且可将上面的等式视为整数运算的等式。因此，$\hat{m} \stackrel{\text{def}}{=} (\hat{c} - 1)/N = [m \cdot \phi(N) \bmod N]$，最终

$$m = [\hat{m} \cdot \phi(N)^{-1} \bmod N]$$

得到所需要的。（注意到，因为$\gcd(N, \phi(N)) = 1$，所以$\phi(N)$是模N的逆。）

根据上面计算的例子，描述一个完整的Paillier加密方案。

构造方法 11.32

设GenModulus是一多项式时间算法，输入为1^n，输出为(N, p, q)，其中$N = pq$且p, q为n比特素数（例外的概率为以n表示的可忽略的概率）。定义一个公钥加密方案如下：

- Gen：输入1^n，运行GenModulus(1^n)得到(N, p, q)。公钥为N，并且私钥为$\langle N, \phi(N)\rangle$。
- Enc：输入一个公钥N和消息$m \in \mathbb{Z}_N$，选择一个随机$r \leftarrow \mathbb{Z}_N^*$，最后输出密文
$$c := [(1+N)^m \cdot r^N \bmod N^2]$$
- Dec：输入一个私钥$\langle N, \phi(N)\rangle$，和一个密文$c$，计算
$$m := \left[\frac{[c^{\phi(N)} \bmod N^2] - 1}{N} \cdot \phi(N)^{-1} \bmod N\right]$$

Paillier 加密方案

例 11.33 设$N = 11 \cdot 17 = 187$（所以$N^2 = 34969$），考虑将信息$m = 175$加密，然后解密对应密文。选择$r = 83 \in \mathbb{Z}_{187}^*$，计算密文

$$c := [(1+187)^{175} \cdot 83^{187} \bmod 34969] = 23911$$

对应于$(175, 83)$。为了解密，注意到$\phi(N) = 160$。所以首先计算$\hat{c} := [23911^{160} \bmod 34969] = 25620$。将其减去 1 并且除以 187 得到$\hat{m} := (25620 - 1)/187 = 137$；因为$90 = [160^{-1} \bmod 187]$，消息恢复为$m := [137 \cdot 90 \bmod 187] = 175$。此处正确性可以根据前面的讨论得到。现在证明安全性。

命题 11.34 如果判定性合数剩余问题与GenModulus相关是困难的，则Paillier加密方案在选择明文攻击条件下是不可区分加密。

证明 设Π表示Paillier加密方案。证明Π在窃听者存在条件下为不可区分加密。由定理

10.10 可知，Π为CPA安全。

设$\mathcal{A}$是一个概率多项式时间敌手，并且定义

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1]$$

考虑下面的PPT敌手D，其试图解决与GenModulus相关的判定性合数剩余问题：

算法D：

算法给定N, y作为输入。

(1) 设置$pk = \langle N\rangle$且运行$\mathcal{A}(pk)$得到两个消息m_0, m_1。

(2) 选择一个随机比特位b且设置$c := [(1+N)^{m_b} \cdot y \bmod N^2]$。

(3) 将密文c给$\mathcal{A}$并且得到输出b'。如果$b' = b$，输出 1；否则输出 0。

现在分析算法D的行为。有两种情形需要考虑：

情形 1。 D的输入通过运行$\mathsf{GenModulus}(1^n)$得到(N,p,q)来产生，选择一个随机$r \leftarrow \mathbb{Z}^*_{N^2}$，并且设$y := [r^N \bmod N^2]$。（也就是说，$y$是$\mathsf{Res}(N^2)$中的一个随机元素。）在这种情况下，对于随机$r \in \mathbb{Z}^*_{N^2}$，密文$c$的创建通过$c = [(1+N)^{m_b} \cdot r^N \bmod N^2]$。

前面曾经提到：不管r是从$\mathbb{Z}^*_N$中随机选择还是从$\mathbb{Z}^*_{N^2}$中随机选择，$[r^N \bmod N^2]$的分布是一样的。当$\mathcal{A}$作为D的一个子程序运行时的分布，与$\mathcal{A}$在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中的分布，完全是一样的。因为恰好当$\mathcal{A}$的输出$b'$=$b$时，$D$的输出为 1，有

$$\Pr[D(N,[r^N \bmod N^2]) = 1] = \Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \varepsilon(n)$$

其中最左边的概率来自定义 11.31 中的实验。

情形 2。 D的输入通过运行$\mathsf{GenModulus}(1^n)$得到(N,p,q)而产生，然后选择一个随机$y \leftarrow \mathbb{Z}^*_{N^2}$。可以声称此种情况下的$\mathcal{A}$的分布与比特$b$独立。为了看到这一点，注意到因为$y$是群$\mathbb{Z}^*_{N^2}$中的一个随机元素，密文$c$随机分布在$\mathbb{Z}^*_{N^2}$中（参见引理 10.20），特别是，它与$m$独立。这意味着$b' = b$的概率恰好为$\frac{1}{2}$。也就是说，

$$\Pr[D(N,r) = 1] = \frac{1}{2}$$

其中的概率来自定义 11.31 中的实验。

结合前面内容，可以得到

$$\left|\Pr[D(N,[r^N \bmod N^2]) = 1] - \Pr[D(N,r) = 1]\right| \leqslant \left|\varepsilon(n) - \frac{1}{2}\right|$$

由于假设判定性合数剩余问题与GenModulus相关是困难的，则存在一个可忽略函数negl满足

$$\left|\varepsilon(n) - \frac{1}{2}\right| \leqslant \mathsf{negl}(n),$$

所以$\varepsilon(n) \leqslant \frac{1}{2} + \mathsf{negl}(n)$，得证。 ■

11.3.3 同态加密

因为Paillier加密方案是一个加法群上同态加密的例子，所以此加密方案在大量情形中都适用。也就是说，如果用$\mathsf{Enc}_N(m)$表示（随机化的）的对一个消息$m \in \mathbb{Z}_N$的Paillier加密，公钥为N，对于所有$m_1, m_2 \in \mathbb{Z}_N$，有

$$\mathsf{Enc}_N(m_1)\cdot\mathsf{Enc}_N(m_2)=\mathsf{Enc}_N([m_1+m_2 \bmod N])$$

为了证明这一点，读者可以先确认

$$\begin{aligned}&((1+N)^{m_1}\cdot r_1^N)\cdot((1+N)^{m_2}\cdot r_2^N)\\&=(1+N)^{[m_1+m_2 \bmod N]}\cdot(r_1r_2)^N \bmod N^2\end{aligned}$$

上式是对消息$[m_1+m_2 \bmod N]$的一个有效加密。

定义 11.35 如果对由$\mathsf{Gen}(1^n)$输出的所有n和所有(pk,sk)分别组成的群$\mathbb{M}$和$\mathbb{C}$满足下面两点，则公钥加密方案$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$是一个同态加密。

(1) 明文空间为$\mathbb{M}$，且所有由Enc_{pk}输出的密文属于$\mathbb{C}$。

(2) 对满足$m_1=\mathsf{Dec}_{sk}(c_1)$和$m_2=\mathsf{Dec}_{sk}(c_2)$的任意$m_1,m_2\in\mathbb{M}$和$c_1,c_2\in\mathbb{C}$，有

$$\mathsf{Dec}_{sk}(c_1\cdot c_2)=m_1\cdot m_2$$

其中群运算分别在$\mathbb{M}$和$\mathbb{C}$中执行。

重申前面所说的内容， Paillier加密方案是选取$\mathbb{M}=\mathbb{Z}_N$和$\mathbb{C}=\mathbb{Z}_{N^2}^*$，公钥$pk=N$的同态加密。

Paillier加密方案并不是本书讲述的第一个同态加密方案。EI Gamal加密方案也是同态的：如果$\mathsf{Gen}(1^n)$输出$pk=(\mathbb{G},q,g,h)$，则明文消息为$\mathbb{G}$中的元素，密文为$\mathbb{G}\times\mathbb{G}$中的元素。此外

$$\langle g^{y_1},h^{y_1}\cdot m_1\rangle\cdot\langle g^{y_2},h^{y_2}\cdot m_2\rangle=\langle g^{y_1+y_2},h^{y_1+y_2}\cdot m_1m_2\rangle$$

是明文消息$m_1m_2\in\mathbb{G}$的一个有效加密。Goldwasser-Micali加密方案也为同态（参见练习11.17）。

Paillier加密方案的一个有利的特征是，此方案是一个大的加法群（也就是$\mathbb{Z}_N$）上的同态。为什么它是个有利的特征？下面我们用基于Paillier加密的密码投票方案为例来说明这一点。

(1) 权威机构产生Paillier加密方案的一个公钥N，并且公布N。

(2) 设 0 代表“no”，1 代表“yes”。l个投票人的每一个都可以通过加密选票进行投票。也就是说，投票人i通过计算$c_i:=[(1+N)^{v_i}\cdot r^N \bmod N^2]$投递自己的选票，其中$r\leftarrow\mathbb{Z}_N^*$是随机选择的。

(3) 每一个投票人公开自己的选票c_i。然后这些选票用计算式

$$c^*:=\left[\prod_{i=1}^{l}c_i \bmod N^2\right]$$

公开统计。任何外部观察者都可以验证此步骤是依照此要求完成的。

(4) 权威机构给予密文c^*。（假定权威机构不能观察到现在为止发生了什么。）通过解密c^*，权威机构得到投票总数：

$$v^*=\sum_{i=1}^{l}v_i \bmod N。$$

如果l较小（以至于$v^*\ll N$），则$v^*=\sum_{i=1}^{l}v_i$。

上面方案的关键是权威机构在不知道任何个人投票的情况下，可以得到正确的投票总数。此外，没有任何投票选手知道他人的投票，并且计算的投票总数可以公开验证的。我们假设所有参与方都是诚实的（仅仅试图通过他们已经知道的信息去推断其他人的选票）。当这个假设不成立时，会出现对此协议的攻击，一个完整的密码学研究方向致力于形式化地定义各种情况（如上述情况）中的合适的安全概念，并且设计安全协议。

参考文献和扩展阅读材料介绍

Childs[32]和Shoup[131]更深入地探讨了本章中用到的（计算）数论知识。参考文献[46]中详细地描述了一个在不知道合数分解形式的情况下，计算该合数模的雅可比符号的算法，并且给了准确的证明。如果想了解，在不知道合数分解形式的情况下，判定关于合数模二次剩余问题，可以阅读Gauss的文献[60]，文中不涉及到其他（猜想的）困难的数论知识。Goldwasser-Micali加密方案在参考文献[69]提到，它是第一个具有严格安全性证明的公钥加密方案。

Rabin的文献[119]证明了计算合数模平方根问题等同于分解问题。文献[28]中指出解决RSA问题并不等同于解决分解问题。本书 11.2.2 节中介绍的基于合数模平方得到一个置换簇的方法是由Blum[21]提出的。Rabin陷门置换的硬核谓词在参考文献[5,75,4]及其参考文献中有介绍。

Shoup[131]中的 10.2 节有关于对于任意整数N, e（而不是此处的$N = pq, e = 2$）的$\mathbb{Z}^*_{N^e}$的介绍。Paillier加密方案在参考文献[113]中有介绍。

练　　习

11.1 设$\mathbb{G}$是一个阿贝尔群。证明$\mathbb{G}$的二次剩余的集合形成一个子群。

11.2 这个问题是关于加法群$\mathbb{Z}_N$中的二次剩余问题。（当且仅当存在一个满足$2x = y \bmod N$的$x \in \mathbb{Z}_N$时，元素$y \in \mathbb{Z}_N$是一个二次剩余。）

（1）当p是奇素数时，$\mathbb{Z}_p$中哪些元素是二次剩余？

（2）设$N = pq$是两个奇素数p和q的乘积。$\mathbb{Z}_N$中哪些元素是二次剩余呢？

（3）设N是一个偶数，$\mathbb{Z}_N$中哪些元素是二次剩余？

11.3 设$N = pq$是两个不相等的奇素数p和q的乘积。指出一个PPT算法，在已知N的分解形式时，选择$\mathcal{QNR}^{+1}_N$中一个随机元素。（你的算法失败的概率是以$||N||$表示可忽略的概率。）

11.4 设$N = pq$是两个不相等奇素数p和q的乘积。证明如果$x \in \mathcal{QR}_N$，则$[x^{-1} \bmod N] \in \mathcal{QR}_N$，并且如果$x \in \mathcal{QNR}^{+1}_N$，则$[x^{-1} \bmod N] \in \mathcal{QNR}^{+1}_N$。

11.5 设$N = pq$是两个不相等奇素数p和q的乘积，并且选定$z \in \mathcal{QNR}^{+1}_N$。说明选择随机$x \leftarrow \mathcal{QR}_N$，设$y := [z \cdot x \bmod N]$得到的$y$是均匀分布在$\mathcal{QNR}^{+1}_N$中。也就是说，对于任意$\hat{y} \in \mathcal{QNR}^{+1}_N$有$\Pr[z \cdot x = \hat{y} \bmod N] = 1/|\mathcal{QNR}^{+1}_N|$，此处概率取自随机选择$x \leftarrow \mathcal{QR}_N$。

提示：利用前面的练习。

11.6 设N是 5 个不相等奇素数的乘积。如果$y \in \mathbb{Z}_N^*$是一个二次剩余，此时方程$x^2 = y \bmod N$有多少解？

11.7 考虑下面Goldwasser-Micali加密方案的变体：运行GenModulus(1^n)得到(N, p, q)，其中$N = pq$，且$p = q = 3 \bmod 4$。（即N为Blum整数。）其公钥为N，私钥为$\langle p, q\rangle$。为了加密消息$m \in \{0,1\}$，发送者随机选择$x \in \mathbb{Z}_N$并且计算密文$c := [(-1)^m \cdot x^2 \bmod N]$。

（1）证明对于上面所描述的N，$[-1 \bmod N] \in \mathcal{QNR}_N^{+1}$。

（2）如果判定性二次剩余与GenModulus相关是困难的，证明上面描述的方案在选择明文攻击条件下是不可区分加密。

11.8 假设判定性二次剩余与GenModulus相关是困难的。说明这意味着从$\mathcal{J}_N^{+1}$的随机元素中区分$\mathcal{QR}_N$中的随机元素是困难的。

11.9 考虑下面Goldwasser-Micali加密方案的变体：运行GenModulus(1^n)得到(N, p, q)。其公钥为N，私钥为$\langle p, q\rangle$。为了加密 0，发送者选择n个随机元素$c_1, \cdots, c_n \leftarrow \mathcal{QR}_N$。为了加密 1，发送者选择$n$个随机元素$c_1, \cdots, c_n \leftarrow \mathcal{J}_N^{+1}$。在每种情况下，得到的密文为$c^* = \langle c_1, \cdots, c_n\rangle$。

（1）说明发送者如何在多项式时间内产生$\mathcal{J}_N^{+1}$中的一个随机元素（失败的概率是以n表示可忽略的）。

（2）提出一个接收者高效解密的方法，虽然出现错误的概率是用n表示可忽略的。

（3）证明如果判定性二次剩余与GenModulus相关是困难的，上面描述的方案为CPA安全的。

提示：利用前面的练习。

11.10 设$\mathcal{G}$是一个多项式时间算法，输入为1^n，输出为一个素数p，$||p|| = n$，和$\mathbb{Z}_p^*$的一个生成元g，证明DDH问题与$\mathcal{G}$相关不是困难的。

提示：用到一个事实：素数模的二次剩余能够高效地被确定。

11.11 当使用前面练习中的$\mathcal{G}$时，离散对数问题被认为是困难的。这意味着函数（簇）$f_{p,q}$中，$f_{p,q}(x) \stackrel{\text{def}}{=} [g^x \bmod p]$是单向函数。设lsb$(x)$表示$x$的最低位。说明lsb不是$f_{p,q}$的一个硬核谓词。

11.12 考虑一个“教科书式Rabin”加密方案，其中消息$m \in \mathcal{QR}_N$被加密，加密使用公钥N（此处N是一个Blum整数），计算的密文为$c := [m^2 \bmod N]$。说明针对此方案的选择密文攻击可以恢复整个私钥。

11.13 设N是一个Blum整数。

（1）定义集合$S \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N^* \mid x < N/2 \text{ and } \mathcal{J}_N(x) = 1\}$。定义函数$f_N : S \leftarrow \mathbb{Z}_N^*$如下：

$$f_N(x) = \begin{cases} [x^2 \bmod N] & if [x^2 \bmod N] < N/2 \\ [-x^2 \bmod N] & if [x^2 \bmod N] > N/2 \end{cases}$$

说明f_N是S上的一个置换。

（2）基于上面定义的f_N的分解问题，定义一个陷门置换簇。

11.14 （1）设N是一个Blum整数。定义函数$\text{half}_N : \mathbb{Z}_N^* \leftarrow \{0,1\}$为

$$\text{half}_N(x) = \begin{cases} 0 & \text{if } x < N/2 \\ 1 & \text{if } x < N/2 \end{cases}$$

说明函数$f : \mathbb{Z}_N^* \to \mathcal{QR}_N \times \{-1, +1\} \times \{0,1\}$定义为

$$f(x) = [x^2 \bmod N], \mathcal{J}_N(x), \mathsf{half}_N(x)$$

是一个单射。

（2）提出一个“填充Rabin”加密方案，来加密长度为n的消息。（提出的方案中所有的算法运行时间都为多项式时间，并且方案必须能正确地解密。尽管安全性证明不大可能，但是提出的方案应该不易受到任何明显的攻击。）

11.15 当N的分解形式已知时，证明命题11.27中的同态可以高效地求逆。

11.16 设$\Phi(N^2)$表示集合$\{(a,1) \mid a \in \mathbb{Z}_N\} \subset \mathbb{Z}^*_{N^2}$。证明判定一个给定的元素$y \in \mathbb{Z}^*_{N^2}$是否属于$\Phi(N^2)$不是困难的。

11.17 当明文空间$\{0,1\}$被视为群$\mathbb{Z}_2$时，证明Goldwasser-Micali加密方案是同态的。

第12章　数字签名

12.1　数字签名简介

到目前为止，仅处理了在公钥环境下实现秘密通信的方法。本章将问题转到保护消息的完整性。在公钥情形下与消息鉴别码相对应的是数字签名，两者也存在一些重要的区别。

数字签名方案允许已确立公钥pk的签名者S以如下方式对消息进行签名：任意知道pk（并且知道公钥是由S确立的）的一方都可以验证消息来源于S且没有经过任何方式的修改。下面介绍一个数字签名方案的典型示例，考虑一个软件公司希望以可认证的方式传播软件补丁：也就是说当公司发布一个软件补丁时，需要所有用户都认为补丁是真实的，并且恶意第三方永远都不可能通过欺骗用户的方式，使其接受一个实际上不是由公司发布的补丁。为了达到此目的，公司可以生成一个公钥pk且伴随一个私钥sk，然后以某种可信的方式将pk分发给客户，并将sk保密。（就像在公钥加密方案中，假设初始的公钥分发已正确实施，以至于所有客户都有一个正确的pk副本。）当发布一个软件补丁m时，软件公司可以用私钥sk计算出m的数字签名σ，然后将消息对(m,σ)发送给所有客户。每一个客户都可以利用公钥pk检查σ是否为m的合法签名，从而核实m的真实性。因为所有用户使用的是同一个公钥pk，所以软件公司仅仅需要计算一个签名，并将其发送给所有用户。

恶意方可能发布一个伪造补丁，而将(m',σ')发送给客户，其中m'代表一个公司没有发布过的补丁。m'可能由一些以前的补丁版本m改造而来，或者是与前面补丁没有任何关系的全新补丁。如果签名方案是“安全的”（将给出更细致的定义），则当客户尝试核实σ'时，将发现其为m'关于pk的无效签名，因此拒绝此签名。即使m'只是将真实补丁m作很小的改动，客户也应该拒绝。

上面例子并不只是数字签名理论上的一个应用，也在当前有着非常广泛的使用。

与消息鉴别码的比较。消息鉴别码和数字签名方案都是用来确保传送消息的完整性（或真实性）。虽然第 9 章讨论了公钥和私钥环境时的加密情况，但很容易发现讨论中也应用到了消息完整性。尤其当发送者需要与多个接收方通信时，使用数字签名，而不是消息鉴别码，可以简化密钥管理。特别是，使用数字签名方案时，发送者可以避免为每一个潜在的接收者确定一个不同的秘密密钥，从而避免了必须分别为每一个密钥计算出MAC标签，发送者仅仅需要计算一个签名，就可以被所有接收者验证。

数字签名和消息鉴别码相比较，一个定性的优势为：数字签名是可公开验证的。这意味着如果一个接收者验证一个给定消息的签名为合法时，可以确保其他接收此签名消息的接收方对其验证也是合法的。然而消息鉴别码不具有此特征，因为此时签名者与每一个接收者共享一个单独密钥：在此情形下，一个恶意发送者可以计算出一个接收方A共享密钥下的正确MAC标签，但是对另一用户B的密钥，其为一个错误的MAC标签。此时，A确认自己接收到一个来自发送方的真实消息，但是不能保证其他接收方的也是如此。

公开验证意味着签名是可传递的：一个由特定签名者S发出的m的签名σ可以被第三方验

证，且该第三方可以用S的公钥验证σ是m一个合法签名（此处，假定第三方也知道S的公钥）。通过复制一个签名，此时第三方可以将此签名显示给另外一方，并且使他们相信m已经通过S的认证，等等。传递性和公开验证是将数字签名应用到证书和公钥基础设施的基础，12.8 节将讨论更多的细节。

数字签名方案也提供了非常重要的属性：不可抵赖性。也就是说，首先假定签名者S广泛公布自己的公钥，一旦S对一个消息进行签名，则以后都不能对此否认。数字签名这个方面的性质对下面的情况非常重要，一个接收者需要向第三方（比如说法官）证明签名者确实对一个特定消息进行了签名（例如一个合同）：假设法官知道S的公钥，或者公钥是公开可得到的，一个有效的消息签名足够使法官相信S确实对此消息进行了签名。消息鉴别码不能提供此功能。理由如下：假设用户S和R共享一个密钥k_{SR}，S将消息m和与之对应的一个（有效的）MAC标签tag发送给R，tag由k_{SR}计算得到。因为法官不知道k_{SR}（事实上，这个密钥由S和R保密），所以法官没法决定tag是否有效。如果R将密钥k_{SR}透露给法官，法官仍然没有办法知道此密钥是否就是R和S共享的“真正”密钥，或者是否为R“伪造”的某些密钥。即使假设给法官的是真正的密钥k_{SR}，但R并没有办法证明tag是由S生成的，因为消息鉴别码是对称的（这意味着任何S能做的，R也能做），这意味着R可能自己生成tag，所以对于法官来说没有办法辨别双方的行为。

就像私钥和公钥加密的对比，消息鉴别码比数字签名的效率要高出约 2 个～3 个数量级。因而，当不需要公开验证、传递性、不可抵赖的情况下，且发送者主要与一个接收者（可以与其共享秘密密钥的一方）通信时，消息鉴别码要更好一些。

与公钥加密的关系。数字签名经常被误认为是公钥加密的“逆向”操作，将发送者和接收者互换位置。事实上数字签名曾经一度被建议可以通过“逆向操作”公钥加密而得到，也就是说对消息m进行签名是通过对其加密（利用私钥）得到σ，并且验证签名就是对σ解密（利用对应的公钥），最后检查结果是否为m。[①] 这种构造数字签名的方法是错误的：大多数情况下并不可用，并且当其使用时，导致数字签名方案是完全不安全的。

12.2 定　义

就像我们所注意到的，数字签名即为消息鉴别码在公钥密码学中的对应。发送者针对消息m的算法用符号Sign表示（而不是Mac），并且算法的输出称为签名（而不是标签）。接收者针对消息和签名的算法仍然用Vrfy表示。下面正式定义数字签名方案。

定义 12.1　一个数字签名方案是一个由概率多项式时间算法组成的三元组(Gen, Sign, Vrfy)，满足下列条件：

(1) 密钥生成算法Gen以安全参数1^n为输入，并且输出一对密钥(pk, sk)，分别称为公钥和私钥。为了方便，假设pk和sk长度至少为n，并且n可以由pk，sk确定。

(2) 签名算法Sign以一个私钥sk和消息[②]$m \in \{0,1\}^*$作为输入。输出一个签名σ，表示为

---

① 这个观点的有些部分确实不令人怀疑，因为，正如将在 12.3.1 节中介绍的，“教科书式RSA”签名事实上为“教科书式 RSA”加密的逆向操作。然而，不管是“教科书式 RSA”签名方案，还是“教科书式 RSA”加密，都不满足最小的安全性定义。

② 类似在公钥加密中的情形，假设其相应消息空间可能依赖于pk。除“教科书式 RSA”签名方案外（不安全的），本书所有的方案都将签名比特串。

$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$。

(3) 确定的验证算法Vrfy以一个公钥pk、消息m和一个签名σ作为输入。输出一位b，当$b = 1$时意味着签名有效，当$b = 0$时为签名无效。将其表示为$b := \mathsf{Vrfy}_{pk}(m, \sigma)$。

此方案需要对于任意n，所有由$\mathsf{Gen}(1^n)$输出的(pk, sk)，以及任意$m \in \{0,1\}^*$，必须满足

$$\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{pk}(m)) = 1$$

如果$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$对于所有由$\mathsf{Gen}(1^n)$输出的$(pk, sk)$，算法$\mathsf{Sign}_{sk}$只对消息$m \in \{0,1\}^{l(n)}$有定义（并且$\mathsf{Vrfy}_{pk}$对于任意$m \notin \{0,1\}^{l(n)}$输出都为0），此时称$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$是一个消息长度为$l(n)$的数字签名方案。

当$\mathsf{Vrfy}_{pk}(m, \sigma) = 1$时，称$\sigma$是消息$m$的一个有效签名（关于某个公钥$pk$）。需要注意的是此处定义的正确性可以放宽到一定程度：错误的出现概率是可忽略的（类似定义 10.1）。虽然在具体方案中这有时是需要的，但在表述中省略这一点。

数字签名方案的一种用法如下。S作为发送方，运行$\mathsf{Gen}(1^n)$得到密钥(pk, sk)。然后将属于S的公钥pk公开。例如S可以将公钥放在网页上，或者放在某些共用目录上面。类似公钥加密，假定任何其它方都可以得到S的合法公钥副本（参考下面的讨论）。当S希望传输信息m时，计算签名$\sigma \leftarrow \mathsf{Sign}_{sk}(m)$并且发送$(m, \sigma)$。一旦接收$(m, \sigma)$，知道公钥$pk$的接收方可以计算$\mathsf{Vrfy}_{pk}(m, \sigma) \stackrel{?}{=} 1$，从而验证$m$的真实性。这说明：S发送了$m$，并且$m$在传送过程中没有任何修改。然而在消息鉴别码中，并没有描述任何关于m何时被发送的情况，所以存在重放攻击的可能性（参考 4.3 节）。

假设参与方可以得到S公钥的一个合法副本，这意味着S至少能够以一种可靠和可验证的方式传送至少一个消息（也就是说pk）。既然如此，读者可能会产生疑问——为什么还需要数字签名。关键点是：可信任密钥pk的分发是一个困难的任务，但是使用数字签名方案意味着密钥分发仅仅需要执行一次，此后不限数量的消息都能以可信赖的方式发送。此外，就像 12.8 节中将要讨论的，签名方案本身被用来确保其它公钥的可信赖分发。这将是“公钥基础设施”中解决密钥分发问题的一个重要工具。

数字签名的安全性。给定一个由签名者S生成的公钥pk，如果敌手输出一个消息m和有效签名σ，并且先前S并没有给m签名，称敌手输出一个“伪造”签名。类似在消息鉴别中，数字签名的安全性意味着，即使敌手允许得到很多自己选择消息的签名，也不能输出一个伪造的签名。这也就是直接类比了消息鉴别码安全性的定义，并且希望读者参阅 4.3 节思考该定义动机和进行深入的讨论。

设$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$是一个数字签名方案，对于敌手$\mathcal{A}$和参数$n$，考虑下面的实验：

数字签名实验$\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$得到密钥(pk, sk)。

(2) 将pk和访问“签名预言机”$\mathsf{Sign}(\cdot)$的权利给敌手$\mathcal{A}$。（此预言机对于敌手选定的任意消息m返回一个签名$\mathsf{Sign}_{sk}(m)$。）然后敌手输出(m, σ)。设$\mathcal{Q}$表示$\mathcal{A}$在执行期间问询签名的消息的集合。

(3) 当且仅当$\mathsf{Vrfy}_{pk}(m, \sigma) = 1$且$m \notin \mathcal{Q}$时，实验输出定义为1。

现在介绍安全性的定义，其与定义 4.2 节中消息鉴别码的安全性定义基本相同。

定义 12.2 一个签名方案$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$如果对任意概率多项式时间敌手$\mathcal{A}$存在一个可忽略函数$\mathsf{negl}$，满足

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] \leqslant \mathsf{negl}(n)$$

则此签名方案在适应性选择消息攻击下是存在性不可伪造的。

12.3 RSA 签名

以基于RSA假设的各种方案的讨论开始考虑具体的签名方案。提醒读者本节介绍的方案都是不安全的（尽管本节所讨论的一个方案的变体，在后面第 13 章中详细讨论的模型中证明是安全的）。介绍这些方案主要是提供一些数字签名方案攻击的例子，同时也让读者直观地感受一下构造安全签名是非常不简单的。

12.3.1 “教科书式 RSA”签名方案及其不安全性

首先介绍“教科书式RSA”签名方案，这个名字是因为很多教科书描述RSA签名时，完全与之一样且没有进一步的警告。但是，下面将要展示的“教科书式RSA”签名是不安全的。

设GenRSA是一个PPT算法，输入为1^n，输出为一个模 N，其为两个n位素数的乘积（例外的概率是可忽略的），以及两个满足$ed = 1 \bmod \phi(N)$的整数e，d。“教科书式RSA”方案简单的运行GenRSA而产生密钥，输出$\langle N, e\rangle$作为公钥，输出$\langle N, d\rangle$作为私钥。为签名消息$m \in \mathbb{Z}_N^*$，签名者计算$\sigma := [m^d \bmod N]$。为验证m在公钥$\langle N, e\rangle$下的签名σ，检查是否有$m \stackrel{?}{=} \sigma^e \bmod N$。参考构造方案 12.3。

构造方案 12.3

设GenRSA如前所述。下面定义一个数字签名方案：

- Gen：输入1^n，运行$\mathsf{Gen}(1^n)$得到(N, e, d)。其中，公钥为$\langle N, e\rangle$，私钥为$\langle N, d\rangle$。
- Sign：输入私钥$sk = \langle N, d\rangle$和消息$m \in \mathbb{Z}_N^*$，计算签名$\sigma := [m^d \bmod N]$。
- Vrfy：输入公钥$pk = \langle N, e\rangle$，消息$m \in \mathbb{Z}_N^*$，和签名$\sigma \in \mathbb{Z}_N^*$。当且仅当$m \stackrel{?}{=} [\sigma^e \bmod N]$时输出为 1。

“教科书式RSA”签名方案

很容易发现验证生成签名的合法总是可以成功的，这是因为

$$\sigma^e = (m^d)^e = m^{[ed \bmod \phi(N)]} = m^1 = m \bmod N$$

然而，教科书式RSA签名方案是不安全的，下面的例子中将进行介绍。

无消息攻击。即使没有得到合法签名者的任何签名，只是基于公钥，就可以伪造一个“教科书式RSA”签名方案的签名。攻击者按照下面的方式进行：给定一个公钥$pk = \langle N, e\rangle$，选择任意一个$\sigma \in \mathbb{Z}_N^*$并且计算$m := [\sigma^e \bmod N]$。然后输出一个伪造$(m, \sigma)$。很快可以发现$\sigma$是$m$的一个有效签名，但是这明显是一个伪造，因为公钥的拥有者并没有生成m的签名（事实上根本没有对此签名）。因而，“教科书式RSA”签名方案并不满足定义 12.2。

读者可能认为这并没有构成一个“现实的”攻击，因为其伪造的一个有效签名并不在m的控制范围内。当然，这一点与定义 12.2 所关注的无关，并且在第 4 章已经讨论，对任何将要利用密码学方案鉴别的消息进行语义假设是很危险的。此外，敌手并没有对m进行控制。例如，通过选择多个随机σ的值可以（以高概率）得到m，m的某些位在理想的范围内。或者通过某种结构化的方式选择σ，而不是随机选择，同样也可以影响伪造者可以输出的消息。

对任意消息伪造签名。一个更具破坏性的“教科书式RSA”签名方案攻击需要敌手得到两条同一签名者的签名，此时允许敌手输出一个任选消息的伪造。假设敌手希望伪造信息$m \in \mathbb{Z}_N^*$关于公钥$pk = \langle N, e\rangle$的签名。敌手选择一个随机$m_1 \in \mathbb{Z}_N^*$，设置$m_2 := [m/m_1 \bmod N]$，然后分别得到m_1和m_2的签名σ_1和σ_2。声明$\sigma := [\sigma_1 \cdot \sigma_2 \bmod N]$也是$m$的一个有效签名。这是因为

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 m_2 = m \bmod N,$$

这里利用了σ_1和σ_2为m_1和m_2的有效签名的事实。因为m不等于m_1或者m_2（例外的概率是可忽略概率），所以构成了一个伪造。

对任意的消息都可以伪造其签名，这很明显是灾难性的。然而，读者可能会争论此攻击是不现实的，因为敌手永远不可能取得签名者的信任，然后恰好得到上面攻击所需要的消息m_1和m_2的签名。同样，这与定义 12.2 所关注的无关。此外，假设签名者将要签名或者不愿意签名什么样的消息是很危险的。例如，客户通过对服务器发送的一个随机值签名而证明其身份，签名可以作为一种认证方法。此时恶意的服务器可以得到其选择任意消息的签名。为了让m_2是一个“合法的”消息让签名者签名，敌手也可以通过特别的方式选择m_1（而不是随机的选择）。同时注意，此攻击可以一般化：如果敌手得到关于某些q个任意消息$M = \{m_1, \cdots, m_q\}$的有效签名，那么敌手可通过选择M子集（集合大小大于 1）的乘积得到的任意$2^q - q - 1$个消息的伪造。

12.3.2 “散列后 RSA”签名方案

为了防止前一节中描述的攻击，目前提出了很多改进的“教科书式RSA”数字签名方案。大多数提议并没有安全性证明，故不应被使用。尽管如此，这里还是给出一个这样的例子（另一个例子在练习 12.4 中给出）。该例子作为更一般性的范例（在下一节中进行介绍）的示例，它在“理想”模型下可证明是安全的，此理想模型将在第 13 章中进行详细地介绍。

改进的“教科书式RSA”签名方案的基本思想是在签名之前对消息应用某个函数 H。即公钥和私钥不变，另外，某个函数$H : \{0,1\}^* \to \mathbb{Z}_N^*$此时包含为公钥的一部分。消息$m \in \{0,1\}^*$的签名为

$$\sigma := [H(m)^d \bmod N]$$

（也就是说首先计算$\hat{m} := H(m)$，然后计算$\sigma := [\hat{m}^d \bmod N]$。）验证数对$(m, \sigma)$是通过检查是否有

$$\sigma^e \stackrel{?}{=} H(m) \bmod N$$

很明显，合法签名的验证总是可以取得成功。

立即可观察到：上面的方案要想成为安全的最小条件是 H 必须为抗碰撞函数（参考 4.6 节）。否则，敌手可以找到满足$H(m_1) = H(m_2)$的两个不同的消息m_1, m_2，此时伪造是容易的。（注意H不需要是压缩的。）因为 H 必须为抗碰撞散列函数，此改进方案有时称为“散列后RSA”签名方案。

从定义 12.2 的意义上来说，不存在已知的函数 H，使“散列后RSA”签名方案为安全方案。然而至少可以描述一种直觉：为什么前一节介绍的“教科书式RSA”签名方案的攻击方法，攻击“散列后RSA”签名方案更加困难？

无消息攻击。前面介绍的试图进行无消息攻击的一种很自然的方法：任选一个$\sigma \in \mathbb{Z}_N^*$，计算$\hat{m} := [\sigma^e \bmod N]$，然后试图找到满足$H(m) = \hat{m}$的某些$m \in \{0,1\}^*$。如果求函数 H 逆的效率很低，则实现此攻击看上去是困难的。

对任意消息伪造签名。前面介绍的选择消息攻击方法需要敌手找到满足$H(m) = [H(m_1) \cdot H(m_2) \bmod N]$的三个消息$m, m_1, m_2$。同样的，如果求函数$H$逆的效率很低，则实现此攻击似乎很困难。

“散列后RSA”的安全性。“散列后RSA”的变体在实践中有所应用。可以证明“散列后RSA”在理想模型下是安全的，此理想模型中 H 为真正的随机函数。很明显 H 不是真正随机的（因为 H 是一个具体的、确定的函数），但当 H 是一个“看起来随机的”密码学散列函数时（例如SHA-1），它提供了一个具有启发意义的验证。参见第 13 章中关于刚提到的理想模型的深入介绍，以及基于此模型的安全性证明。

12.4 “Hash-and-Sign”范例

“散列后RSA”签名方案可以视作一种阻止某些对“教科书式RSA”签名方案攻击的尝试。此方案能够取得成功是因为其为混合结构，如上面的讨论。可能会注意到，“散列后RSA”相对于“教科书式RSA”方案相比具有另外一个优势：它可以用于对任意长度的比特串进行签名，而不仅仅是$\mathbb{Z}_N^*$中的元素。通常这个特征非常有用，先对一个消息进行散列，然后对结果进行签名的方法（利用一个安全的签名方案）是达到此目的的一个标准方式。现在研究此方法的安全性。

设$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$是一个消息长度为$l(n)$的标准方案。（满足这一点的所有方案可以改进为适用于任意长度l，只要它是n的超对数。）设$\Pi_H = (\mathsf{Gen}_H, H)$为定义 4.11 中介绍的散列函数，其中 H 的输出的长度为n，与安全参数1^n有关。对任意长度的消息，构造一个签名方法$\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$如下：$\mathsf{Gen}'(1^n)$计算$(pk, sk) \leftarrow \mathsf{Gen}_S(1^n)$和$s \leftarrow \mathsf{Gen}_H(1^n)$，并且设公钥等于$\langle pk, s\rangle$，私钥等于$\langle sk, s\rangle$。为了签名消息$m \in \{0,1\}^*$，签名者计算$\sigma \leftarrow \mathsf{Sign}_{sk}(H^s(m))$。验证检查$\mathsf{Vrfy}_{pk}(H^s(m), \sigma) \stackrel{?}{=} 1$。参考构造方法 12.4。注意“散列后RSA”签名方案事实上是由“教科书式RSA”签名方案利用这个方法构造而来（当然下面的定理不适用于这种情况，因为“教科书式RSA”签名方案是不安全的）。

构造方案 12.4

设$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$与$\Pi_H = (\mathsf{Gen}_H, H)$为前面文中介绍的方案。下面构造一个对任意长度消息的签名方案Π'：

- Gen'：输入1^n，运行$\mathsf{Gen}_S(1^n)$得到(pk, sk)，并且运行$\mathsf{Gen}_H(1^n)$得到s。其中，公钥为$pk' = \langle pk, s\rangle$，私钥为$sk' = \langle sk, s\rangle$。
- Sign'：输入私钥$\langle sk, s\rangle$和消息$m \in \{0,1\}^*$，计算$\sigma' \leftarrow \mathsf{Sign}_{sk}(H^s(m))$。
- Vrfy'：输入公钥$\langle pk, s\rangle$，消息$m \in \{0,1\}^*$和一个签名σ，当且仅当$\mathsf{Vrfy}_{pk}(H^s(m), \sigma) \stackrel{?}{=} 1$时输出 1。

Hash-and-Sign范例

定理 12.5 如果Π在适应性选择消息攻击下是存在性不可伪造的，且Π_H是抗碰撞的，则构造方案 12.4 在适应性选择消息攻击下是存在性不可伪造的。

证明 证明的基本思路为：如果存在一个伪造，则必须找到H的碰撞或者伪造一个根据定长签名方案Π的签名。这一思路来自于 4.7 节中的定理 4.16，所以此处直接给出正式的证明。

设$\mathcal{A}'$为一个概率多项式时间敌手攻击Π'。在执行实验$\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)$时，设$pk'=\langle pk,s\rangle$表示用到的公钥，令$\mathcal{Q}$表示$\mathcal{A}'$所请求的签名消息的集合，令$(m,\sigma)$为$\mathcal{A}'$的最后输出。不失一般性，假设$m\notin\mathcal{Q}$。定义$\mathsf{coll}_{\mathcal{A}',\Pi'}(n)$为下面的事件：在实验$\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)$中，存在一个$m'\in\mathcal{Q}$对于$H^s(m')=H^s(m)$。有

$$\begin{aligned}&\Pr[\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)=1]\\&=\Pr[\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)=1\wedge\mathsf{coll}_{\mathcal{A}',\Pi'}(n)]\\&\quad+\Pr[\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)=1\wedge\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)]\\&\leqslant\Pr[\mathsf{coll}_{\mathcal{A}',\Pi'}(n)]+\Pr[\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)=1\wedge\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)]\end{aligned}\tag{12.1}$$

下面证明方程（12.1）中的两项都是可忽略的，从而来完成证明。就直觉而言，第一项是可忽略的，因为Π_H的抗碰撞性。第二项也是可忽略的，因为Π的安全性。（Π的安全性意味着第二项是可忽略的，这是因为：当$\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)$发生时，对任意$m'\in\mathcal{Q}$，有$H^s(m)\neq H^s(m')$，这意味着$\mathcal{A}'$针对方案$\Pi'$成功伪造的$(m,\sigma)$产生了一个针对方案$\Pi$的成功伪造$(H^s(m),\sigma)$。由于$\Pi$的安全性，这发生的概率必然是可忽略的。）

下面从证明第一项开始。考虑下面找出Π_H的一个碰撞的PPT算法$\mathcal{C}$。

算法$\mathcal{C}$：

算法给定s作为输入（隐含地包括n）。

(1) 计算$\mathsf{Gen}_S(1^n)$得到(pk,sk)。设$pk'=\langle pk,s\rangle$。

(2) 以pk'为输入，运行$\mathcal{A}'$。当$\mathcal{A}'$问询第i个签名（某个消息$m_i\in\{0,1\}^*$的签名）时，计算$\sigma_i\leftarrow\mathsf{Sign}_{sk}(H^s(m_i))$，并且将$\sigma_i$给$\mathcal{A}'$。

(3) 最终，$\mathcal{A}'$输出(m,σ)。如果存在i，满足$H^s(m)=H^s(m_i)$，则输出(m,m_i)。

现在分析$\mathcal{C}$的行为。当$\mathcal{C}$的输入由运行$\mathsf{Gen}_H(1^n)$得到s时，$\mathcal{A}'$作为$\mathcal{C}$的子程序运行时视图的分布与$\mathcal{A}'$在实验$\mathsf{Sig-forge}_{\mathcal{A}',\Pi'}(n)$中运行时视图的分布是相同的。因为恰好当$\mathsf{coll}_{\mathcal{A}',\Pi'}(n)$发生时，$\mathcal{C}$输出一个碰撞，得到

$$\Pr[\mathsf{Hash\text{-}coll}_{\mathcal{C},\Pi_H}(n)=1]=\Pr[\mathsf{coll}_{\mathcal{A}',\Pi'}(n)]$$

因为Π_H抗碰撞，得到$\Pr[\mathsf{coll}_{\mathcal{A}',\Pi'}(n)]$是可忽略的。

现在证明式（12.1）中的第二项是可忽略的。考虑下面攻击签名方案Π的PPT敌手$\mathcal{A}$。

敌手$\mathcal{A}$：

给定一个公钥pk作为敌手的输入（也隐含地包括n在内），并且可访问签名预言机$\mathsf{Sign}(\cdot)$。

(1) 计算$\mathsf{Gen}_H(1^n)$得到s，设$pk'=\langle pk,s\rangle$。

(2) 以pk'为输入，运行$\mathcal{A}'$。当$\mathcal{A}'$问询第i个签名（某个消息$m_i \in \{0,1\}^*$的签名）时，回答如下：计算$\hat{m}_i = H^s(m_i)$，然后由签名预言机得到$\hat{m}_i$的签名σ_i，并且将σ_i给$\mathcal{A}'$。

(3) 最终，$\mathcal{A}'$输出(m,σ)。输出$(H^s(m),\sigma)$。

实验$\mathsf{Sig}-\mathsf{forge}_{\mathcal{A},\Pi}(n)$中，$\mathcal{A}'$作为$\mathcal{A}$的子程序运行时视图的分布与$\mathcal{A}'$在实验$\mathsf{Sig}-\mathsf{forge}_{\mathcal{A},\Pi}(n)$中运行时视图的分布完全一样。此外，很容易验证当$\mathsf{Sig}-\mathsf{forge}_{\mathcal{A}',\Pi'}(n)=1$和$\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)$同时发生时，$\mathcal{A}$输出一个伪造。（很明显，$\sigma$是对$H^s(m)$的一个有效签名。事实上，$\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)$的发生意味着$\mathcal{A}$没有向自己的签名预言机问询过$\hat{m}$（$\hat{m}=H^s(m)$），所以$\hat{m}\notin\mathcal{Q}$。）因此

$$\Pr[\mathsf{Sign}-\mathsf{forge}_{\mathcal{A},\Pi}(n)=1]=\Pr[\mathsf{Sign}-\mathsf{forge}_{\mathcal{A}',\Pi'}(n)\wedge\overline{\mathsf{coll}}_{\mathcal{A}',\Pi'}(n)],$$

并且由Π的安全性意味着后者的概率可忽略。因此定理得证。

■

消息鉴别码也具有与定理 12.5 相似的性质，并给定理 4.6 提供一个选择，从定长MAC构造变长MAC（在附加的假设条件下，即假设抗碰撞散列函数存在，这对定理 4.6 来说是不必要的）。就像在 4.7 节中注意到的，“hash-and-MAC”方法是NMAC和HMAC安全性的基础。

12.5 Lamport的“一次性签名方案”

定义 12.2 为数字签名安全性的标准定义，这里也考虑一个较弱的定义。满足这些较弱定义的签名方案可能适合某些受限制的应用，也可作为满足较强安全性定义的签名方案的有用的“构造工具”，将在下一节中介绍。

本节给出“一次性签名方案”的正式定义。当此方案只用于对一个消息进行签名时，是安全的。然后介绍一个Lamport提出的构造方案，它是基于任何的单向函数的“一次性签名方案”。定义如下，设$\Pi=(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})$为一个数字签名方案，对于敌手$\mathcal{A}$和参数$n$，考虑下面的实验。

一次性签名实验$\mathsf{Sig}-\mathsf{forge}^{1-\mathsf{time}}_{\mathcal{A},\Pi}(n)$：

(1) 运行$\mathsf{Gen}(1^n)$得到密钥(pk,sk)。

(2) 将pk给敌手$\mathcal{A}$，并且向预言机$\mathsf{Sign}_{sk}(\cdot)$问询$m'$。然后，$\mathcal{A}$输出$(m,\sigma)$，其中$m\neq m'$。

(3) 当且仅当$\mathsf{Vrfy}_{\mathsf{pk}}(m,\sigma)=1$时，实验的输出定义为 1。

定义 12.6 如果对于任意概率多项式时间敌手$\mathcal{A}$，存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Sig}-\mathsf{forge}^{1-\mathsf{time}}_{\mathcal{A},\Pi}(n)=1]\leqslant\mathsf{negl}(n)$$

则签名方案$\Pi=(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})$在单消息攻击下是存在性不可伪造的，或者为“一次性签名方案”。

Lamport签名方案的基本思想很简单，利用消息为 3 比特的情况进行描述。设f是一个单向函数。还记得单向函数意味着f很容易计算，但是很难求逆，参见定义 7.66。在f的值域内，公钥由 6 个元素$y_{1,0},y_{1,1},y_{2,0},y_{2,1},y_{3,0},y_{3,1}$组成。私钥包含相应的原像

$x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}$。这些密钥可以视为二维数组：

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix} \qquad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix}$$

对一个消息$m = m_1 \cdot m_2 \cdot m_3$签名时，其中每一个$m_i$是一个比特位，签名者公布相应的前像$x_{i,m}$，$1 \leqslant i \leqslant 3$；签名$\sigma$简单的由三个值$(x_{1,m_1}, x_{2,m_2}, x_{3,m_3})$组成。签名验证是显然的：给出消息$m = m_1 \cdot m_2 \cdot m_3$的候选签名$(x_1, x_2, x_3)$，对于$1 \leqslant i \leqslant 3$时，当且仅当$f(x_i) \stackrel{?}{=} y_{i,m}$时予以接受。此过程如图 12.1 所示。对于一般情况下任意长度l的签名将在构造方案 12.7 中正式描述。

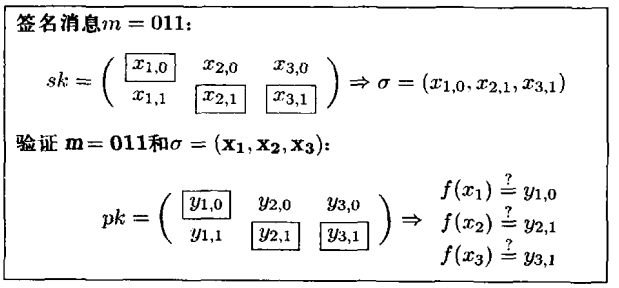


图 12.1　用于给消息$m = 011$签名的Lamport方案

构造方案 12.7

设f为一个单向函数。下面构造一个长度为$l = l(n)$的签名方案：

- Gen：输入1^n，当$i \in \{1, \cdots, l\}$时，按下面的方式进行。

（1）随机选择$x_{i,0}, x_{i,1} \leftarrow \{0,1\}^n$。

（2）计算$y_{i,0} = f(x_{i,0})$和$y_{i,1} = f(x_{i,1})$。

公钥pk和私钥sk为

$$pk := \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{l,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{l,1} \end{pmatrix} \qquad sk := \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{l,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{l,1} \end{pmatrix}$$

- Sign：输入上面的私钥和消息$m \in \{0,1\}^l$，其中$m = m_1 \cdots m_l$，输出签名$(x_{1,m_1}, \cdots, x_{l,m_l})$。
- Vrfy：输入上面的公钥和消息$m \in \{0,1\}^l$，其中$m = m_1 \cdots m_l$，和签名$\sigma = (x_1, \cdots, x_l)$，对于$1 \leqslant i \leqslant l$，当且仅当$f(x_i) = y_{i,m_i}$时，输出为 1。

Lamport签名方案

定理 12.8　设l为任意多项式。如果f为单向函数，则构造方案 12.7 为一个消息长度为l的“一次性签名方案”。

证明　在剩下的证明中，设$l = l(n)$。作为一种对方案安全性的直觉，注意到对于一个给定公钥$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{l,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{l,1} \end{pmatrix}$的敌手，对于任意$(i^*, b^*)$，找到满足$f(x) = y_{i^*,b^*}$的$x$相当于对$f$求逆。所以，对于任意消息$m$，只给定公钥时，计算其签名必然是困难的。当给出一个消息m'的签名后，计算某个不同消息m的签名的难度又如何呢？如果$m' \neq m$，必至少在一个位置i^*上两个消息是不同的。假定$m_{i^*} = b^* \neq m'_{i^*}$，想伪造一个$m$的签名需要找到一个满足

$f(x)=y_{i^*,b^*}$的x。即使给定对于所有$(i,b)\neq(i^*,b^*)$的$\{x_{i,b}\}$，寻找满足条件的x不会变得更加容易（因为$\{x_{i,b}\}_{(i,b)\neq(i^*,b^*)}$的选择独立于$x_{i^*,b^*}$），$m'$的签名对$x$的值泄漏甚至更少。

现在将这个直观分析转化为一个正式的证明。设Π表示Lamport方案。设$\mathcal{A}$为一个概率多项式时间敌手，并且定义

$$\varepsilon(n)\stackrel{\text{def}}{=}\Pr[\mathsf{Sig\text{-}forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)=1]。$$

在一个特定的情况下执行$\mathsf{Sig-forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)$，设$m'$表示$\mathcal{A}$所请求其签名的消息（不失一般性，假设$\mathcal{A}$总是请求一个消息的签名），并且设$(m,\sigma)$表示$\mathcal{A}$的最终输出。如果$\mathsf{Vrfy}_{\mathsf{pk}}(m,\sigma)=1$，且$m_i\neq m'_i$（也就是说$m$和$m'$在第$i$个位置上不同），以及$m_i=b\neq m'_i$，称$\mathcal{A}$输出了一个在$(i,b)$的伪造。注意每当$\mathcal{A}$输出一个伪造时，也输出了一个在$(i,b)$的伪造。

考虑下面的PPT算法$\mathcal{I}$，其试图对单向函数f求逆。

算法$\mathcal{I}$：

算法给定y和1^n作为输入。

(1) 随机选择$i^*\leftarrow\{1,\cdots,l\}$和$b^*\leftarrow\{0,1\}$。设$y_{i^*,b^*}:=y$。

(2) 对于任意$i^*\leftarrow\{1,\cdots,l\}$和$b\in\{0,1\}$且$(i,b)\neq(i^*,b^*)$：

- 选择$x_{i,b}\leftarrow\{0,1\}^n$且设$y_{i,b}:=f(x_{i,b})$。

(3) 输入$pk:=\begin{pmatrix}y_{1,0} & y_{2,0} & \cdots & y_{l,0}\\ y_{1,1} & y_{2,1} & \cdots & y_{l,1}\end{pmatrix}$，运行$\mathcal{A}$。

(4) 当$\mathcal{A}$请求一个消息m'的签名时：

- 如果$m'_{i^*}=b^*$，停止。
- 否则，返回正确的签名$\sigma=(x_{1,m'_1},\cdots,x_{l,m'_l})$。

(5) 当$\mathcal{A}$输出(m,σ)且$\sigma=(x_1,\cdots,x_p)$：

- 如果$\mathcal{A}$输出一个在(i^*,b^*)的伪造，则输出x_{i^*}。

当$\mathcal{A}$输出一个在(i^*,b^*)的伪造时，算法$\mathcal{I}$对其输入y成功地求逆。当$\mathcal{I}$的输入由选择随机的$x\leftarrow\{0,1\}^n$和设$y:=f(x)$产生时（参考定义 7.66），这种情况发生的概率是我们感兴趣的。想象一个“脑力实验”，在开始时将x给$\mathcal{I}$，设$x_{i^*,b^*}:=x$，此时在第 4 步总是返回一个签名给$\mathcal{A}$（也就是说，即使$m'_{i^*}=b^*$）。不难发现$\mathcal{A}$在此理想试验中作为$\mathcal{I}$的子程序运行时，其视图分布与$\mathcal{A}$在实验$\mathsf{Sig-forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)$中相同。因此，$\mathcal{A}$在第 5 步中输出一个伪造的概率恰好为$\varepsilon(n)$。因为在实验开始时，$(i^*,b^*)$是随机选择的，可以理解为$\mathcal{A}$独立于与此选择，$\mathcal{A}$在$(i^*,b^*)$输出一个伪造的概率至少为$1/2l(n)$。（这是因为一次签名伪造意味着至少存在一个点$(i,b)$的伪造。由于有$2l(n)$个点，在$(i^*,b^*)$点存在一个伪造的概率至少为$1/2l(n)$。）因此得出在此“脑力实验”中，$\mathcal{A}$输出一个$(i^*,b^*)$的伪造的概率至少为$\varepsilon(n)/2l(n)$。

回到刚开始描述的包含$\mathcal{I}$的“真实实验”，观察$\mathcal{A}$输出一个在(i^*,b^*)处伪造的概率并没有改变。这是因为如果$\mathcal{A}$输出一个在(i^*,b^*)点的伪造时，“脑力实验”和“真实实验”一致。也就是说如果$\mathcal{A}$请求m'（$m'_{i^*}=b^*$）的签名时，实验不同；但在这种情况下，（根据定义）$\mathcal{A}$随后不能输出一个在(i^*,b^*)点的伪造。所以在“真实试验”中，$\mathcal{A}$输出一个在(i^*,b^*)点伪造的概率仍然至少为$\varepsilon(n)/2l(n)$。因而得到

$$\Pr[\mathsf{Invert}_{\mathcal{I},f}(n)=1]\geqslant\varepsilon(n)/2l(n)$$

因为f为一个单向函数，所以存在一个可忽略函数negl，使得

$$\Pr[\mathsf{Invert}_{\mathcal{I},f}(n)=1]\leqslant\mathsf{negl}(n)$$

这意味着$\varepsilon(n)/2l(n) \leqslant \mathsf{negl}(n)$，又因为$l$是一个多项式，则$\varepsilon(n)$本身是可忽略的。得证。

■

推论 12.9 如果存在单向函数，则对于任意多项式l，都存在一个针对消息长度为l的“一次性签名方案”。

12.6 *来自抗碰撞散列的签名

到目前为止还没有介绍任何适应性选择消息攻击下存在性不可伪造的签名方案（参考定义 12.2）。这里介绍一个效率较低的构造方法，其实质是基于目前为止所介绍的最简单的一个密码学假设。此构造方法依赖抗碰撞散列函数的存在性，并且主要用于证明实现定义 12.2 的可行性。

值得提出的是满足定义 12.2 的签名方案，一般而言很难构造，即使是到目前为止，已知的方案中可证明满足此定义的有效方案很少。第 13 章将讨论一个非常高效的签名方案，在某个“理想”模型下可证明是安全的，并且第 13 章将对该模型进行介绍和深入讨论。发展其他高效的基于标准模型（现在正使用该模型）的可证明安全签名方案，是当今一个活跃的研究领域。

下面分阶段逐步构造最终的构造方案。在 12.6.1 节中，将定义有状态签名方案的概念，其中签名者每次签名后都更新自己的私钥。这里还将介绍如何构造一个满足定义 12.2 的有状态的签名方案。在 12.6.2 节中，将介绍这个方案的一个高效的（仍然有状态的）变体，并且证明其在适应性选择消息攻击下仍然具有存在性不可伪造。然后，描述此构造方案如何变成无状态的，从而得到一个起初定义的签名方案。

12.6.1 基于 Chain 的签名

首先定义一个签名方案，其允许签名者维护一些状态，这些状态在每次签名后进行更新。

定义 12.10 一个有状态签名方案是概率多项式时间算法的 3 元组$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$，满足下列条件：

(1) 密钥生成算法Gen以一个安全参数1^n为输入，并且输出(pk, sk, s_0)。它们分别称为公钥、私钥和初始状态。假设pk和sk的长度至少为n，并且n可由pk，sk来确定。

(2) 签名算法Sign以私钥sk、值s_{i-1}和消息$m \in \{0,1\}^*$为输入，输出签名σ和值s_i。

(3) 具有确定性的验证算法Vrfy以公钥pk、消息m和签名σ为输入，输出一个比特b。

要求对任意n，每个由$\mathsf{Gen}(1^n)$输出的(pk, sk, s_0)，任意的消息$m_1, \cdots, m_l \in \{0,1\}^*$，如果递归地计算$(\sigma_i, s_i) \leftarrow \mathsf{Sign}_{sk, s_{i-1}}(m_i)$ $(i \in \{1, \cdots, l\})$，则对于任意$i \in \{1, \cdots, l\}$，有

$$\mathsf{Vrfy}_{pk}(m_i, \sigma_i) = 1$$

如果$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$对所有由$\mathsf{Gen}(1^n)$输出的$(pk, sk)$都满足，算法$\mathsf{Sign}_{sk}$只对消息$m \in \{0,1\}^{l(n)}$有定义（且对于$m \notin \{0,1\}^{l(n)}$，$\mathsf{Vrfy}_{pk}$输出为 0），此时称$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$是一

个消息长度为$l(n)$的有状态签名方案。

需要强调状态对于验证签名并不是必须的（其实取决于方案，为了确保安全，状态可能需要保密）。不维护状态的签名方案称为无状态签名，区别于有状态方案。很明显，无状态方案更受人欢迎（虽然有状态方案也有其用途）。然而，这里的目的是：有状态签名方案的介绍是作为完全无状态构造方案的铺垫。

有状态签名方案的适应性选择消息攻击下的存在性不可伪造，其定义方式类似于定义12.2，微小的差别是签名预言机仅返回签名（不返回状态），且每次被调用时，签名预言机相应地更新状态。

可以简单地构造一个“l次”状态签名方案，此方案可以签名$l = l(n)$个消息，其中l为任意多项式。（此处安全性定义可能和前面给出“一次性签名方案”定义相似；由于这里只是进行非正式的讨论，所以没有给出正式的定义。）此构造方法是可行的，简单地让l个独立产生的公钥（针对任何“一次性签名方案”），私钥也由此方法组成。换言之，$pk = (pk_1, \cdots, pk_l)$且$sk = (sk_1, \cdots, sk_l)$，其中每一对$(pk_i, sk_i)$都是针对某个“一次性签名方案”独立生成的密钥对。其状态是一个初始值为 1 的计数器。对消息m进行签名要使用私钥sk和现在的状态$s \leqslant l$，然后简单输出$\sigma \leftarrow \mathsf{Sign}_{sk_s}(m)$（也就是说利用私钥$sk_s$产生消息$m$的一次性签名），且将状态更新为$s + 1$。因为初始状态从 1 开始，这意味着第$i$个消息用$sk_i$签名。验证$m$的签名$\sigma$可以通过检查$\sigma$是否为消息$m$关于公钥$\{pk_i\}$的有效签名。

从直觉上讲，因为每一个私钥仅用于对一个消息进行签名，所以此方案用于对l个消息进行签名时是安全的。因为l可能是一个任意的多项式，但是为什么这没有给出所要找的答案呢？主要缺点是此方案在生成密钥时，需要预先设定好可签名消息的数量上限。（特别是此方案不满足定义 12.2。）因为一旦达到上限，需要生成一个新的公钥并进行分发，所以这是一个潜在的很严重的局限。取而代之的是，需要一个可以支持不限数量消息的签名方案。此方案的另外一个缺陷是，因为公钥和私钥的长度与可签名消息的数量成线性关系，所以此方案的效率并不高。

设$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$为一个“一次性签名方案”。在刚刚描述的方案中，签名者要调用Gen算法l次，得到公钥$pk_1, \cdots, pk_l$，把每个公钥包含到实际的公钥pk。签名者此时被限制为只能最多签名l个消息。“基于Chain的签名方案”改进了这一情况，签名者可以按照自己需要的方式“实时”生成和验证额外的公钥。

在基于Chain的方案中，公钥仅由Gen产生的单个公钥pk_1组成，私钥包含与此公钥关联的私钥sk_1。对第一个消息m_1进行签名时，签名者首先用Gen产生一个新密钥对(pk_2, sk_2)，然后利用sk_1对m_1和pk_2进行签名，得到$\sigma_1 \leftarrow \mathsf{Sign}_{sk_1}(m_1||pk_2)$。此时输出的签名包括$pk_2$和$\sigma_1$，签名者添加$(m_1, pk_2, sk_2, \sigma_1)$作为当前状态。通常，当要对第$i$个消息进行签名时，签名者存储$\{(m_j, pk_{j+1}, sk_{j+1}, \sigma_j)\}_{j=1}^{i-1}$作为状态的一部分。为了对第$i$个消息$m_i$进行签名，签名者首先利用Gen产生一个新的密钥对$(pk_{i+1}, sk_{i+1})$，然后使用$sk_i$对$m_i$和$pk_{i+1}$进行签名，得到一个签名$\sigma_i \leftarrow \mathsf{Sign}_{sk_i}(m_i||pk_{i+1})$。实际输出的签名包括$pk_{i+1}$、$\sigma_i$以及值$\{(m_j, pk_{j+1}, \sigma_j)\}_{j=1}^{i-1}$。然后签名者添加$(m_i, pk_{i+1}, sk_{i+1}, \sigma_i)$作为自身状态。参见图 12.2 对此过程做了一个图形化的描述。

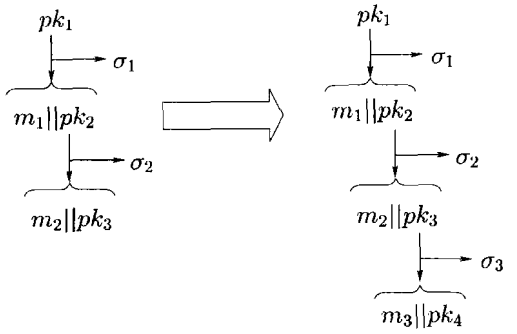


图 12.2　基于Chain的签名：对第三个消息m_3签名前后的状态

为了验证消息$m = m_i$关于公钥pk_1的签名$(pk_{i+1}, \sigma_i, \{(m_j, pk_{j+1}, \sigma_j)\}_{j=1}^{i-1})$，接收者验证公钥$pk_j$和下一个公钥$pk_{j+1}$之间链上的每一个环节，以及验证最后一个公钥$pk_{i+1}$和$m$之间的环节。也就是说，对于任意$j \in \{1, \cdots, i\}$，当且仅当$\mathsf{Vrfy}_{pk}(m_j||pk_{j+1}, \sigma_j) \stackrel{?}{=} 1$时验证程序输出为 1。（参考图 12.2）观察可得：此验证程序从初始分发的公钥pk_1开始验证。

至少从直观上来说，并不难相信这样构造的签名方案在适应性选择消息攻击下是存在不可伪造的（不考虑有多少签名消息被签名）。正式地说，这同样是因为每个密钥对(pk_i, sk_i)仅被用于签名一个“消息”（此处消息实际上是一个消息/公钥对$m_i||pk_{i+1}$）。由于在下一节中将证明一个更加高效签名方案的安全性，这里并不给出基于Chain方案的正式安全性证明。

在基于Chain的方案中，每一个公钥pk_i被用于对一个消息和另外一个公钥进行签名。因而，一个基本的要求是：“一次性签名方案”Π能够对长于公钥的消息进行签名。12.5 节中介绍的Lamport方案不具备此性质。然而，如果应用 12.4 节中介绍的Hash-and-Sign范例到Lamport方案，可以得到一个对任意长度消息进行签名的“一次性签名方案”。（尽管定理 12.5 仅陈述关于满足定义 12.2 的签名方案，并不难发现同样的证明对“一次性签名方案”也成立。）因为这个结论对下一节的内容很重要，现给出正式的叙述。

引理 12.11　如果存在抗碰撞散列函数，则存在“一次性签名方案”（对任意长度的消息）。

证明　如前所述，简单利用 12.5 节中介绍的Hash-and-Sign范例与Lamport签名方案的结合。注意到抗碰撞散列函数的存在性意味着单向函数的存在性（参考练习 12.10），所以此假设是充分的。

■

基于Chain的签名方案是一个有状态的签名方案，此方案在适应性选择消息攻击下是存在性不可伪造的。尽管如此，此方案也有一些缺陷：①没有直接的方式消除状态（回顾最终目标是一个满足定义 12.2 的无状态方案）。②此方案并不高效，因为签名的长度、状态的大小以及验证时间都与已经签名消息的数量成线性关系。③每一个签名都揭露了所有前面已经签名的消息。这些在技术上并没有破坏任何签名所需的安全需求，但在某些情形下可能是不理想的。

12.6.2 基于 Tree 的签名

前面一节介绍的基于Chain的签名方案中，可以视为在签名者维护一棵“树”，根为公钥pk_1，其度数为 1，且其深度等于目前为止已经加密过的消息数量（参考图 12.2）。提高此方法效率的一个自然的方式为使用二叉树，其中每个节点度数为 2。和前面类似，一个签名对应于树中“验证的”路径，从叶子到根节点。注意，只要树为多项式深度（即使是有指数的大小），验证仍然可以在多项式时间内完成。

具体地说，为了对一个长度为n的消息签名，将在一个深度为n并且有2^n个叶子节点的二叉树中工作。和前面一样，签名者可以按照需要“实时”往树中添加节点。与基于Chain的方案相比不同的是，只有叶子节点（并非内部节点）可以用于验证消息。树的每一个叶子节点对应一个长度为n的可能的消息。

更详细地，想象一个深度为n的二叉树，根节点用ε标记（即空字符串），一个用二进制串w（串长度小于n）标记的节点，其左孩子节点标记为$w0$，右孩子节点标记为$w1$。此树不会构造完毕（即达到指数级的大数目），相反可按照签名者的需要构造。

对每个节点w，关联某个“一次性签名方案”Π的一对密钥pk_w，sk_w。根节点的公钥pk_ε实际上是签名者的公钥。签名一个消息$m \in \{0,1\}^n$时，签名者按照下面的步骤完成：

(1) 首先，对从根到标记为m的叶子节点的路径上的所有需要的节点，都生成密钥。（某些公钥在对前面消息的签名过程中可能已经生成了，不用再次生成。）

(2) 然后，用私钥sk_w计算$pk_{w0}||pk_{w1}$的签名，“认证”从根到标记为m的叶子节点的路径，其中每一个字符串w为m的一个前缀。

(3) 最后，用私钥sk_m计算m的签名来“认证”m本身。

m最终的签名由下面两部分组成：m在pk_m下的签名，以及验证从标记为m的叶子到根节点路径所需要的全部信息，如图 12.3 所示。另外，签名者通过存储签名过程产生的所有密钥对来更新自身状态。构造方案 12.12 中给出此方案的正式描述。

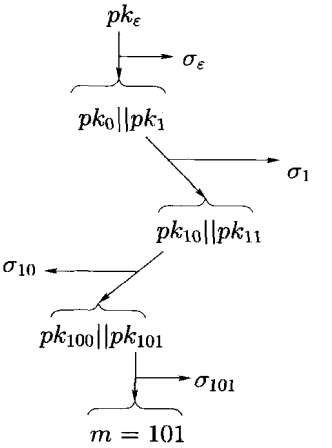


图 12.3 基于 Tree 的签名：签名消息 m=101

构造方案 12.12

设$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$为一个签名方案。对于任意字符串$m$，设$m|_i \stackrel{\text{def}}{=} m_1 \cdots m_i$表示$m$的$i$比特前缀（且空字符串表示为$m|_0 \stackrel{\text{def}}{=} \varepsilon$）。按照下面的方式构造签名方案$\Pi^* = (\mathsf{Gen}^*, \mathsf{Sign}^*, \mathsf{Vrfy}^*)$：

- Gen^*：输入1^n，计算$(pk_\varepsilon, sk_\varepsilon) \leftarrow \mathsf{Gen}(1^n)$，并且输出公钥$pk_\varepsilon$。私钥和初始状态为$sk_\varepsilon$。
- Sign^*：输入一个消息$m \in \{0,1\}^n$，执行下面的步骤。

① 对$i = 0$到$(n-1)$，做如下工作：

如果$pk_{m|_i 0}$，$pk_{m|_i 1}$和$\sigma_{m|_i}$不在状态里面，则计算$(pk_{m|_i 0}, sk_{m|_i 0}) \leftarrow \mathsf{Gen}(1^n)$，$(pk_{m|_i 1}, sk_{m|_i 1}) \leftarrow \mathsf{Gen}(1^n)$和$\sigma_{m|_i} \leftarrow \mathsf{Sign}_{sk_{m|_i}}(pk_{m|_i 0} || pk_{m|_i 1})$。另外，将这些值都添加到状态里面。

② 如果σ_m还不在状态里面，计算$\sigma_m \leftarrow \mathsf{Sign}_{sk_m}(m)$并且将其作为状态的一部分存储。

③ 输出签名$\left(\{\sigma_{m|_i}, pk_{m|_i 0}, pk_{m|_i 1}\}_{i=0}^{n-1}, \sigma_m\right)$。

- Vrfy^*：输入一个公钥pk_ε，消息m和签名$\left(\{\sigma_{m|_i}, pk_{m|_i 0}, pk_{m|_i 1}\}_{i=0}^{n-1}, \sigma_m\right)$，当且仅当满足下面两个条件时，输出 1。

① 对于所有$i \in \{0, \cdots, n-1\}$，$\mathsf{Vrfy}_{pk_{m|_i}}(pk_{m|_i 0} || pk_{m|_i 1}, \sigma_{m|_i}) \stackrel{?}{=} 1$。

②$\mathsf{Vrfy}_{pk_m}(m, \sigma_m) \stackrel{?}{=} 1$。

基于 Tree 的签名方案

注意到此方案中每一个对应的私钥仅用于对一个消息签名。关联到内部节点的每一个私钥都对一对公钥进行签名，在叶子节点上私钥只用于对一个消息进行签名（一次）。注意，密钥用于给其它一对密钥签名，从而需要“一次性签名方案”Π能对比公钥还长的消息进行签名。引理 12.11 介绍这种方案可以基于抗碰撞散列函数构造。

在证明基于Tree的方法前，注意到此方案比基于Chain的方案在很多方面都有改进。它仍允许对没有不限数量的消息进行签名。（尽管只有2^n个叶子，消息空间仅包含2^n个消息。在任何一种情况，2^n最终都将比任何关于n的多项式函数要大。）在效率方面，签名的长度与验证时间与消息长度n成比例，但是独立于已签名的消息数量。此方案仍然是有状态的，但是在证明下面的结论后，将看到如何避免使用状态。

定理 12.13 设Π是一个“一次性签名方案”。构造方案 12.12 在适应性选择消息攻击下是存在性不可伪造的。

证明 设Π^*表示构造方案 12.12。设$\mathcal{A}^*$表示一个概率多项式时间敌手，$l^* = l^*(n)$为$\mathcal{A}^*$进行签名问询数量的（多项式）上限，并且设$l(n) \stackrel{\text{def}}{=} 2nl^*(n) + 1$；为了便于表示，用$l$代替$l(n)$。注意$l(n)$为$\Pi$中公钥数量的上限，是$\Pi^*$生成$l^*(n)$个签名所需要的。这是因为$\Pi^*$中每一个签名至多需要$2n$个来自$\Pi$的新密钥（在最坏的情况下），以及一个来自$\Pi$的额外密钥用于实际公钥$pk_\varepsilon$。定义

$$\delta(n) \stackrel{\text{def}}{=} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}^*, \Pi^*}(n) = 1]$$

考虑下面攻击“一次性签名方案”Π的PPT敌手$\mathcal{A}$。

敌手$\mathcal{A}$：

给定公钥pk作为$\mathcal{A}$的输入（安全参数n是隐含的）。

(1) 选择一个随机指数$i^* \leftarrow \{1,\cdots,l\}$。按照下面的方式构造一个公钥列表$pk^1,\cdots,pk^l$：

设$pk^{i^*} := pk$。

如果$i \neq i^*$，计算$(pk^i, sk^i) \leftarrow \mathsf{Gen}(1^n)$。

(2) 输入公钥$pk_\varepsilon = pk^1$，运行$\mathcal{A}^*$。当$\mathcal{A}^*$需要消息m的签名时：

① 对$i = 0$到$(n-1)$，做如下工作：

如果$pk_{m|_i 0}$，$pk_{m|_i 1}$和$\sigma_{m|_i}$都没有定义，则设$pk_{m|_i 0}$，$pk_{m|_i 1}$等于下面两个没有用到的密钥pk^j和pk^{j+1}，并且计算$pk_{m|_i 0}||pk_{m|_i 1}$关于公钥$pk_{m|_i}$[③]的签名$\sigma_{m|_i}$。

② 如果σ_m还没有定义，则计算m关于密钥pk_m的签名σ_m（参考页末注）。

③ 将$\left(\{\sigma_{m|_i}, pk_{m|_i 0}, pk_{m|_i 1}\}_{i=0}^{n-1}, \sigma_m\right)$给$\mathcal{A}^*$。

(3) $\mathcal{A}^*$输出消息m和签名$\left(\{\sigma'_{m|_i}, pk'_{m|_i 0}, pk'_{m|_i 1}\}_{i=0}^{n-1}, \sigma'_m\right)$。如果其为$m$的一个有效签名，则：

情形 1。假定对于$pk'_{m|_j 0} \neq pk_{m|_j 0}$或$pk'_{m|_j 1} \neq pk_{m|_j 1}$，存在一个$j \in \{0,\cdots,n-1\}$，这也包括当$pk_{m|_j 0}$，$pk_{m|_j 1}$从没有被$\mathcal{A}$定义的情形。选择此时最小的$j$，设$i$满足$pk^i = pk_{m|_j} = pk'_{m|_j}$（由于$j$有极小值，所以存在这样的$i$）。如果$i = i^*$，输出$(pk_{m|_j 0}||pk_{m|_j 1}, \sigma'_{m|_j})$。

情形 2。如果不满足情形 1，则$pk'_m = pk_m$。设i满足$pk^i = pk_m$。如果$i = i^*$，输出(m, σ'_m)。

至此完成了$\mathcal{A}$的描述。

在实验$\mathsf{Sig\text{-}forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)$中，$\mathcal{A}^*$作为$\mathcal{A}$的一个子程序运行时视图的分布，与$\mathcal{A}^*$在实验$\mathsf{Sig\text{-}forge}_{\mathcal{A}^*,\Pi^*}(n)$中是相同的。[④]因此，当$\mathcal{A}^*$在实验中作为$\mathcal{A}$的一个子程序运行时，$\mathcal{A}^*$输出一个伪造的概率恰为$\delta(n)$。假定$\mathcal{A}^*$输出一个伪造，考虑上面描述的两种可能情况：

情形 1。因为i^*为均匀随机选择的，并且其分布与$\mathcal{A}^*$独立，$i = i^*$的概率恰好为$1/l$。如果$i = i^*$，此时$\mathcal{A}$请求对消息$pk_{m|_j 0}||pk_{m|_j 1}$的签名（对应给定的公钥$pk = pk^{i^*} = pk_{m|_j}$，并不需要其它的签名）。此外，

$$pk'_{m|_j 0}||pk'_{m|_j 1} \neq pk_{m|_j 0}||pk_{m|_j 1}$$

和$\sigma'_{m|_j}$已经是消息$pk'_{m|_j 0}||pk'_{m|_j 1}$关于$pk$的有效签名。因此，$\mathcal{A}$在此种情形下输出一个伪造。

情形 2。同样的，因为i^*为均匀随机选择的，并且其分布与$\mathcal{A}^*$独立，$i = i^*$的概率恰好为$1/l$。如果$i = i^*$，此时$\mathcal{A}$不需要问询任何关于公钥$pk = pk^i = pk_m$的签名，并且σ'_m已经是m关于pk的一个有效签名。

---

③ 如果$i \neq i^*$，此时$\mathcal{A}$可以自己计算一个关于pk^i的签名。$\mathcal{A}$也可以通过对签名预言机适当地问询得到（单个）关于pk^{i^*}的签名。这些就是此处的意思。

④ 正如已提到的，$\mathcal{A}$绝不会“用完”公钥。$\mathcal{A}^*$的一个签名问询要用到$2n$个公钥。因此，即使回答$\mathcal{A}^*$的每次签名问询都需要新的密钥（通常情况不是这样），除了“根”公钥pk_ε外，$\mathcal{A}$只需要$2nl^*(n)$个公钥。

如果$\mathcal{A}^*$输出一个伪造（不管上面哪种情形发生），$\mathcal{A}'$的一个伪造的概率恰好为$1/l$。即

$$\Pr[\mathsf{Sig\text{-}forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)=1]=\delta(n)/l(n)$$

因为Π是一个“一次性签名方案”，故存在一个可忽略函数negl，有

$$\Pr[\mathsf{Sig\text{-}forge}^{1\text{-}\mathsf{time}}_{\mathcal{A},\Pi}(n)=1]\leqslant \mathsf{negl}(n)$$

因为l为多项式，可得$\delta(n)$是可忽略的。

■

无状态解决方案。Π^*中签名者的状态取决于已签名的消息。但可以想象，在生成密钥时签名者必须预先产生整棵“树”中所有节点需要的信息。（也就是在生成密钥时，签名者需要生成针对长度至多为n的所有二进制串w的密钥$\{(pk_w, sk_w)\}$和签名$\{\sigma_w\}$。）如果按照这种方式产生密钥，则签名者不必更新自己的状态。这些值可作为一个（大的）私钥的一部分来存储，因而可以得到一个无状态的方案。当然这种方法的问题是，产生所有这些值将需要指数时间。

另一种可选方法为，按照需要存储一些可以用于生成$\{(pk_w, sk_w)\}$和$\{\sigma_w\}$的随机值，而不是存储值本身。也就是说，签名者对于每个w，存储一个随机字符串r_w，每当需要pk_w，sk_w时，签名者计算$(pk_w, sk_w) := \mathsf{Gen}(1^n; r_w)$，这里表示使用随机值$r_w$产生一个长度为$n$的密钥。同样的，如果签名过程是概率的，签名者可以存储r'_w，然后设$\sigma_w := \mathsf{Sign}_{sk_w}(pk_{w0}||pk_{w1}; r'_w)$（这里假设$|w| < n$）。然而产生和存储足够多的随机串，仍然需要指数时间和空间。

上面替换方案的一个简单修改可以得到一个多项式时间解决方案。并不像上面提出的存储随机r_w和r'_w，签名者存储一个伪随机函数F的两个密钥k，k'。当需要时，pk_w，sk_w的值可以按照下面两步产生：

(1) 计算$r_w := F_k(w)$。[5]

(2) 计算$(pk_w, sk_w) := \mathsf{Gen}(1^n; r_w)$（和前面一样）。

另外，r'_w是用于生成签名σ_w，而密钥k'用于生成r'_w。从而得到了一个无状态的签名方案，在此方案中，密钥生成（以及签名和验证）可以在多项式时间内完成。从直观上讲，这是因为存储一个随机函数相当于存储所有需要的r_w和r'_w，而存储一个伪随机函数“正好可以达到此目的”。此修改过的方案在适应性选择消息攻击下保持存在性不可伪造，对此的证明留着练习。

因为抗碰撞散列函数的存在性意味着单向函数的存在性（参见练习 12.10），而单向函数的存在性意味着伪随机函数的存在性（参见第 6 章），可得定理 12.14。

定理 12.14 如果存在抗碰撞散列函数，则存在一个（无状态）签名方案在适应性选择消息攻击下具有存在性不可伪造。

通过单向函数是存在的这一（最小）假设，强调构造一个满足定义 12.2 的签名方案是可能的。对这一结果的证明超出了本书的范围。

---

⑤ 假设F的输出的长度足够长，并且w以单射方式填充成某个定长串。此处略去这些技术细节。

12.7 数字签名标准(DSS)

12.3.2 节中描述的散列后RSA签名方案，以及关于此方案的变体给出了一个被广泛应用于实践的签名方案的例子。数字签名标准是另外一个很重要的例子（DSS），有时也称为数字签名算法（DSA）。此方案由美国国家标准技术研究所（NIST）于 1991 年提出，此后成为美国政府标准。DSS的安全性依赖离散对数问题的困难性，已经应用了很多年，并没有发现任何严重的攻击。但是就像散列后的RSA签名，现在还没有基于离散对数（或其它假设）的对DSS的安全性证明。前面注意到散列后RSA签名在某个理想模型下是可证明安全的，此理想模型将在下一章中介绍。不幸的是，即使在此理想模型下，也没有对DSS的安全性证明。由于这些原因，这里仅仅给出此方案的描述。

设$\mathcal{G}$是一个概率多项式时间算法，输入为1^n，输出为(p,q,g)（除去可忽略的概率外）：①p与q都为素数，且$||q|| = n$；②$q \mid (p-1)$但$q^2 \nmid (p-1)$；③g为$\mathbb{Z}_p^*$子群的一个生成元，此子群阶为q。（例如可用 7.3.3 节中的算法。）DSS在构造方案 12.15 中给出。

构造方案 12.15

设$\mathcal{G}$为上面介绍的算法。定义一个签名方案如下：

- Gen：输入1^n，运行算法$\mathcal{G}(1^n)$得到(p,q,g)。设$H: \{0,1\}^* \to \mathbb{Z}_q$为一个函数。均匀随机地选择$x \leftarrow \mathbb{Z}_q$，并且设置$y := [g^x \bmod p]$。公钥为$\langle H,p,q,g,y\rangle$，私钥为$\langle H,p,q,g,x\rangle$。
- Sign：输入私钥$\langle H,p,q,g,x\rangle$和消息$m \in \{0,1\}^*$，均匀随机地选择$k \leftarrow \mathbb{Z}_q^*$，并且设置$r := [[g^k \bmod p] \bmod q]$。计算$s := [(H(m) + xr) \cdot k^{-1} \bmod q]$，并且输出签名$(r,s)$。
- Vrfy：输入公钥$\langle H,p,q,g,y\rangle$，消息$m \in \{0,1\}^*$和签名(r,s)，其中$r \in \mathbb{Z}_q$，$s \in \mathbb{Z}_q^*$。计算$u_1 := [H(m) \cdot s^{-1} \bmod q]$和$u_2 := [r \cdot s^{-1} \bmod q]$。当且仅当$r \stackrel{?}{=} [[g^{u_1} y^{u_2} \bmod p] \bmod q]$时输出1。

数字签名标准（DSS）

下面来验证此方案的正确性。设$\hat{m} = H(m)$，由签名者输出的签名(r,s)满足$r = [[g^k \bmod p] \bmod q]$和$s = [(\hat{m} + xr) \cdot k^{-1} \bmod q]$。

假设$s \neq 0$（出现的概率是可忽略的）。利用事实$y = g^x$，以及还记得前面曾经提过可以在“指数”模q中工作，有

$$\begin{aligned} g^{\hat{m}s^{-1}} y^{rs^{-1}} &= g^{\hat{m}\cdot(\hat{m}+xr)^{-1}k} g^{xr\cdot(\hat{m}+xr)^{-1}k} \bmod p \\ &= g^{(\hat{m}+xr)\cdot(\hat{m}+xr)^{-1}k} \bmod p \\ &= g^k \bmod p \end{aligned}$$

因此，$[[g^{\hat{m}s^{-1}} y^{rs^{-1}} \bmod p] \bmod q] = [[g^k \bmod p] \bmod q] = r$，验证成功。

12.8 数字证书和公钥基础设施

通过简要讨论数字签名的一个主要的应用来结束本章，即介绍安全的公钥分发。这造成了公钥的讨论一个循环。在本章以及前三章中，已经说明了当公钥安全地分发后，如何使用公钥进行加密。现在介绍如何使用公钥加密进行安全的公钥分发。这可能听起来像一个循环，但并非如此。本质上来说，一旦单个公钥（属于某个可信方）安全地分发，则此密钥可以用于“引导”安全的分发其它任意多的公钥。因此，安全密钥分发的问题仅需要解决一次（至少从理论上来说是如此）。

此处关键的思想是数字证书的概念。数字证书为一个签名，它将某个实体和某个公钥绑定在一起。具体地说，假设参与方Charlie已经生成了一个安全数字签名方案的（在本节仅关注满足定义 12.2 的签名方案）的一个密钥对(pk_C, sk_C)。进一步假设另外一个参与方Bob也生成一个密钥对(pk_B, sk_B)（在目前的讨论中，可能是一个签名方案的密钥对，或者是公钥加密方案的密钥对。），并且Charlie知道pk_B是Bob的公钥。Charlie可以计算签名

$$\mathsf{cert}_{C\to B} \stackrel{\mathsf{def}}{=} \mathsf{Sign}_{sk_C}(\text{' Bob's key is } pk_B \text{'})$$

并且将此签名发送给Bob。此签名$\mathsf{cert}_{C\to B}$称为一个由Charlie发布的关于Bob的密钥的数字证书。在实践中，一个证书应该无歧义地标识参与方持有的特定公钥，所以一个比“Bob”更具唯一性地描叙词汇应被使用，例如，Bob的全名或email地址。

现在，假设Bob希望与某个参与方Alice通信，并且Alice已经知道pk_C。Bob现在可以做的就是将$(pk_B, \mathsf{cert}_{C\to B})$发送给Alice。然后，Alice可以验证$\mathsf{cert}_{C\to B}$的确是消息“Bob's key is pk_B”对应于pk_C的有效签名。假设验证成功，Alice现在知道Charlie已经签名了所指的消息。如果Alice信任Charlie，此时她可能会接收pk_B就是Bob的合法公钥。

注意Bob与Alice之间的所有通信都是在一个不安全、不可靠的信道上进行的。如果一个主动敌手干扰了从Bob到Alice之间的通信$(pk_B, \mathsf{cert}_{C\to B})$，除非Charlie前面已经签名了其它一些关联Bob与pk'_B的证书，否则敌手不能生成一个关联Bob与其它公钥pk'_B的有效证书（这种情况并不能构成攻击）。这里均假设Charlie是诚实的，并且其签名私钥没有泄漏。

在上面的描述中，已经省去了很多细节。尤其是，没有讨论Alice起初如何得知pk_C；Charlie如何可以确信pk_B就是Bob的公钥；以及起初Alice如何决定是否相信Charlie。完整说明这些细节（和其它一些）可以给出一个公钥基础设施（PKI），其可以广泛地分发公钥。很多不同的PKI模型已经被提出，在此只提到少量较流行的PKI。此处的论述将保持一个概述性的层次，对深入的细节感兴趣的读者可以查阅本章后的参考文献。

单一认证机构。最简单的PKI假设一个单一认证机构（CA）被每个人完全信任，并且分发每个人的公钥证书。一个证书认证机构通常不是个人，更有可能是从事公钥认证业务的公司、政府部门或者组织里面的一个部门（虽然后者CA可能只被组织内的人员使用）。任何人想依赖CA的服务，都必须得到一个合法的CA公钥副本pk_{CA}。很明显，如果某个参与方得到一个不正确的pk_{CA}，则此参与方可能无法获得任何人的真实公钥副本，所以获得合法的CA公钥副本pk_{CA}必须在一个安全的方式下进行。这意味着pk_{CA}必须在一个可靠的渠道下分发。最早的解决方案为通过物理方法：例如，如果CA在一个组织内部，则任何雇员都可以在其工作的第一天从CA处直接得到一个真实的pk_{CA}副本。如果CA是一个公司，则其他用户必须亲自到该公司，取一个包含CA公钥的CD-ROM。需要再次强调的是，这个相对繁琐的步骤只

需要执行一次。

在实践中，CA分发自己公钥常用的一个方式为将这个公钥和某些其它软件“捆绑在一起”。事实上，此方法应用在当今最流行的Web浏览器上：将CA的公钥与Web浏览器一起提供，称为“证书存储”。并且在证书到达的时候，Web浏览器可以利用存储的公钥，实行程序化地自动验证。（实际上，Web浏览器通常有多个CA的公钥，它们固定在代码中，所以更准确地属于下面将讨论的“多个CA”的情况。）举另外一个例子：当购买一个新电脑时，公钥可能包含为操作系统的一部分。

参与方Bob从CA得到一个证书的机制必须非常细致地控制。例如，Bob可能必须亲自到场，在CA面前提交一个CD-ROM，其中包含Bob公钥pk_B，以及可证明其姓名（或email地址）是真实的标识。只有满足此条件时，CA才发放Bob公钥的一个有效证书。

在单 CA 模型中，仅当适当的时候，参与方完全信任CA来发布证书。这就是为什么在证书发布之前，仔细的验证过程是至关重要的。因此，如果Alice收到一个证书$\mathsf{cert}_{CA\to B}$认证pk_B就是Bob的公钥，Alice将认为此证书有效，并使用pk_B作为Bob的公钥。

多个证书认证机构。尽管单CA模型非常简单且很有吸引力，但并不实用。首先，在单一组织范围外，要每个人都相信同一个CA的可能性非常小。这并不意味着每个人都认为CA被攻破。可能只是一个简单情况：某些人发现CA的认证过程是不充分的（比如产生一个证书时，CA仅仅需要一个ID，但是Alice希望使用两个ID，而不是一个）。此外，CA是整个系统的单一失效点。如果CA被攻破，或者可以被收买，或者CA仅仅松懈了签名私钥的保护，则CA所发布公钥的合法性都会受到质疑。即使是在没有敌手的环境中，仅依靠单个CA也是一个问题：如果CA遥不可及，则不能发布新证书；如果多个参与方希望马上能得到证书，则CA可能超负荷（尽管这仍比KDC模型好很多，因为一旦证书发布了，CA便不需要在线）。

缓解此问题的一个方案为依靠多个CA。参与方Bob如果希望得到一个自己公钥的证书，可以选择任意一个CA(s)来发布证书，并且参与方Alice，呈递一个证书，或者由多个CA发布的证书，可选择一个自己信任的CA。Bob从哪个CA得到证书并没有影响（除去Bob的不便或花费因素），但是Alice必须非常小心，因为其通信的安全性最终仅仅取决于其信任的CA的最小安全性。也就是说，如果Alice信任两个CA，分别为CA_1和CA_2，并且CA_2被敌手攻破。此时尽管敌手不能伪造由CA_1发布的证书，但是此敌手可对自己选择的任何身份/公钥对生成CA_2发布的证书。在目前的Web浏览器中，这实际上是一个很现实的问题。前面已经提到，Web 浏览器通常在证书存储中预先保存一些CA公钥，并且默认设置是对所有CA同等信任的。基本上任何愿意付费的公司，都可以作为一个CA加入。所以，预先设置的CA列表中包括一些有着很好信誉、制度完善的公司，以及其他信任度并不容易建立的新公司。让用户自己手工设置浏览器的设定，可以让用户仅接收自己信任的CA签名的证书。

授权和证书链。另外一种缓解某些单CA负担的方法（但不涉及单一失效点的安全担忧）是使用“证书链”。此处介绍的证书链长度为 2，但是很容易推广到任意长度的证书链形式。

假设Charlie充当一个CA，按照原讨论的方式发布一个Bob的证书。进一步假设Bob的密钥pk_B是签名方案的公钥。Bob反过来可以为其它参与方发布证书。例如，Bob用下面的形式发布一个Alice的证书$\mathsf{cert}_{B\to A} \stackrel{\mathsf{def}}{=} \mathsf{Sign}_{sk_B}(\text{' Alice's key is } pk_A \text{'})$。

此时，如果Alice想与第四方Dave通信，Dave知道Charlie的公钥（但是不知道Bob的公

钥），然后Alice发送“pk_A，$\text{cert}_{B\to A}$，pk_B，$\text{cert}_{C\to B}$”给Dave。Dave从这里面可以辨认出：因为拥有Charlie的公钥并相信Charlie，所以首先验证Charlie签名的证书$\text{cert}_{C\to B}$，表明pk_B确实属于Bob。Dave也可以验证Bob签名的证书$\text{cert}_{B\to A}$，表明pk_A确实属于Alice。如果Dave相信Charlie仅对值得信任的人发布证书，则Dave接受pk_A为Alice的真实密钥。

在这个例子中，证书$\text{cert}_{C\to B}$关联了很强的语义。在前面的讨论中，此种形式的证书仅是一个断言：Bob持有公钥pk_B。此时，证书的断言是：Bob持有公钥pk_B，且Bob发布的其它证书应得到信任。并不是由Charlie发布的所有证书都拥有这个语义。Charlie可以做的是：比方说在发布证书时准备两种不同“类型”的证书。

当Charlie签名一个Bob的证书时，若具有前面讨论的强语义，Charlie将自己发布证书的能力授权给Bob。从效果上讲，Bob现在可以作为Charlie的一个代理，代表Charlie发布证书。回到基于CA的PKI，可以想象一个“根”CA和“二级”CA：$CA_1,\cdots,CA_n$。根CA发布证书给每一个二级CA，此时二级CA可以给其它持有公钥的在主体发布证书。从而减轻了根CA的负担，并且参与方可以更方便地获得证书（因为比方说他们现在可以联系最近的二级CA）。另外，管理这些二级CA可能是困难的，并且它们的存在意味着系统中现在有更多的攻击点。

“信任Web（Web of Trust）”模型。最后一个PKI的例子将讨论一个完全分布式的模型，其中没有中央信任点，称为“信任Web”模型。此模型的一个变体被用于PGP电子邮件加密程序中的公钥分发。

在信任Web模型中，任何人都可以发布证书给其他任何人，并且每个用户都必须自己作出决定：其他用户发布的证书有多少是值得信任的。下面介绍一个例子说明这些是如何做到的：假设用户Alice已经拥有某些用户C_1，C_2，C_3的公钥pk_1，pk_2，pk_3。（下面将讨论起初Alice如何获得这些公钥。）另外一个想要与Alice交流的用户Bob可能有证书$\text{cert}_{C_1\to B}$，$\text{cert}_{C_3\to B}$和$\text{cert}_{C_4\to B}$，并且将这些证书（以及自己的公钥pk_B）发送给Alice。Alice无法验证$\text{cert}_{C_4\to B}$（因为她没有C_4的公钥），但是她可以验证其它的两个证书。现在，Alice必须决定对C_1和C_3的信任程度。如果Alice信任C_1，或者是信任C_1和C_3，她就会决定接受pk_B。（例如，Alice可能想到C_1或者C_3被敌手攻破，但是她考虑不太可能两者都被攻破。）

就像已经描述过的，在此模型中，用户希望收集其他用户的公钥以及自己公钥的证书（由其他参与方发布）。在PGP情形中，这些通常在PGP用户聚在一起的时候，例如，在一个会议上，由“密钥签名聚会”完成，参加者互换自己公钥的真实副本，并且互相签署证书。通常情况下，作为密钥签名方的用户可能互不知道对方，但是在接收某人公钥前，可以通过检查驾驶执照来确认。

公钥和证书也可以存储在中央数据库中，PGP中正是如此（参见http：//pgp.mit.edu）。当Alice希望发送一个加密消息给Bob时，她可以在此数据库中查询Bob的公钥。伴随Bob的公钥，数据库也将返回一个已发布的所有Bob公钥的证书的列表。同时，也可能从数据库中发现多个Bob的公钥，以及每一个公钥可能被不同的参与方发布的证书所认证。同样，Alice在使用它们之前，需要决定对每一个公钥的信任程度。

信任Web模型是有吸引力的，因为它是在“草根”层次工作，不需要信任任何中央机构。另外，它可能在普通用户对自己邮件加密方面很有用，但在关键安全领域或对于机构的公钥分发是不合适的。例如，如果一个用户想和银行通信，该用户不大可能信任在会议上遇到的人来认证银行的公钥，银行代表也不太可能去参加“密钥签名聚会”来得到银行密钥的

认证。

吊销证书。前面有一个很重要的问题还没有提及，就是证书不可能无限期有效。一个雇员可能离开了公司，此种情况下，该雇员将不再允许接收来自公司内部的加密通信；一个用户的私钥也有可能被偷（假设用户知道被偷这件事），此时该用户希望产生一个新密钥对，并且将旧公钥删除。在任何场景中，我们都需要一种方法使前面问题中的证书无效。

处理此问题的方法各种各样且复杂，在此我们仅提及两个相对简单的想法，从某种意义上说，代表了两个极端。（改进这些方法在如今是一个活跃的研究领域，读者可以参见本章后对这一领域文献的介绍。）

过期。一种防止证书被无限期地使用的方法是将过期日期作为证书的一部分。CA Charlie为Bob公钥发布的证书可能是下面的形式

$$\mathsf{cert}_{C\to B} \stackrel{\mathsf{def}}{=} \mathsf{Sign}_{sk_C}(\text{' Bob's key is } pk_B \text{ ', date})$$

其中date是某个未来的日期，到那一天证书将无效。（例如，可能是证书发布之日后的一年。）当另一个用户验证此证书时，不仅需要知道pk_B，还要知道截止日期。并且现在不仅需要检查签名的有效性，还需要检查过期日期是否已过。当证书过期时，持有证书的用户必须联系CA得到一个新发布的证书。此时，在发布另一个证书前，CA需要再次验证用户的身份或者凭证。

过期日期提供了前面问题一个非常粗略的解决办法。如果雇员在得到证书的后一天离开公司，并且证书过期日期为一年，则此雇员可以不合法地使用自己的公钥一整年的时间，直到过期日期。由于这个原因，此方法通常与其它方法相结合使用，下面给出其中的一个。

撤销。当雇员离开公司，或者用户私钥被偷，通常希望已经颁发给的相应证书立即失效，或者至少越快越好。这通过CA明确的撤销证书来解决。当然，如果针对用户有多个CA颁发的证书情形来讨论可能更通用一些。但为了简便，这里假设只有单个CA。

有很多不同的方法可以处理撤销问题。一种可行方案（也是仅讨论的一种方案）为：CA在颁发的每一个证书中都包含一个序列号。也就是说，现在证书的形式为

$$\mathsf{cert}_{C\to B} \stackrel{\mathsf{def}}{=} \mathsf{Sign}_{sk_C}(\text{' Bob's key is } pk_B \text{ ', \#\#\#})$$

其中，"###"代表证书的序列号。每一个证书都应该有一个唯一的序列号，并且CA将存储每个证书生成的信息(Bob, pk_B, ###)。

如果用户Bob对应于公钥pk_B的私钥被偷，Bob可以报警给CA密钥被偷。（注意，此处CA一定要验证Bob的身份，防止另外一个用户虚假撤销给Bob颁发的证书。另一种可选方案参见练习 12.13。）然后CA在自己的数据库中查找与证书（关于Bob和pk_B）相关联的序列号。在每一天结束时，CA将产生一个包含所有撤销证书的序列号的"证书撤销列表"（CRL），并且对整个列表和当前的日期进行签名。然后将签名的列表广泛分发，可能是公布到CA的公共网页上。

为了验证按照上面方法颁发的证书，另一个用户需要pk_B以及证书的序列号（可由Bob与其它信息一起传送）。验证需要检查签名的有效性，以及检查序列号并没有出现在最近的撤销列表中，并且验证撤销列表中的CA签名。

在刚刚描述的这种方法中，在证书撤销变成无效之前，最多有一天的间隔。此方法的灵活性比基于过期日期的方法要更强。

参考文献和扩展阅读材料介绍

早期在数字签名方面取得显著成果的有：Diffie and Hellman[47]，Rivest，Shamir和Adleman[122]，以及Goldwasser，Micali和Yao[71]。Lamport“一次性签名方案”发表于1979年[92]，之前已经在参考文献[47]中有所讲述。

Goldwasser，Micali和Rivest[70]定义了在适应性选择消息攻击下存在性不可伪造的概率，并且也第一次构造了满足此定义的有状态签名方案（有趣的是，就像论文中所解释的，在他们作此项工作之前，有些人认为满足定义 12.2 的方案是不可能存在的。）Goldreich的文献[63]中提出了一个使Goldwasser-Micali-Rivest成为无状态方案的方法，并且实质上在12.6.2 节中已经采用了Goldreich的思想。

因为现已知的签名构造方案都较难分析，并且需要的密码学假设超出了本书的介绍，所以没有介绍任何高效的签名构造方案。有兴趣的读者可查阅相关文献，如[61, 38, 55]。

基于Tree的构造方案在思想上与Merkle[100,101]提出的构造方案 12.12 相似，虽然基于Tree的方法在参考文献[70]中也有应用。Naor和Yung[106]提出单向置换是构造一次性签名的充分条件，该一次性签名可对任意长度的消息签名。Rompel[124,84]对此结果进行改进，说明了单向函数是构造一次性签名的充分条件。就像 12.6.2 节中看到的，这种类型的一次性签名可用于构造适应性选择消息攻击下存在性不可伪造的签名方案。

Goldreich[65]的第 6 章和Katz[83]提供了对签名方案比本章所述更广泛更深入的论述。

证书的概念由Kohnfelder[89]在他的本科论文中第一次提出。参考文献[87]的第 15 章和参考文献[1]对公钥基础设施细节进行了更深入的讨论。Ellison和Schneier的文献[51]讨论了为什么PKI不是身份管理的万灵药的一些原因。

类似在私钥情形中（参考 4.9 节），也考虑在公钥情形中同时获得私密性和完整性的机制，这有时会称为“签密”。An、Dodis和Rabin的文献[6]对此领域进行了很好的介绍。

练　习

12.1　证明 1 比特消息“一次性签名方案”的存在性意味着单向函数的存在性。

12.2　对于下列对签名的安全性定义的变体，说明“教科书式RSA”签名方案是否安全的，并且证明你的答案：

(1)在第一个变体中，实验如下：将公钥pk和一个随机消息给敌手。然后允许敌手询问一次签名预言机，得到一个不等于m的消息的签名。其后，敌手输出一个签名σ，并且如果$\mathsf{Vrfy}_{pk}(m,\sigma)=1$则成功。如果敌手在此实验中最多以可忽略概率取得成功，则称方案具有安全性。

(2)第二个变体和上面类似，唯一不同的是根本不允许敌手询问签名预言机。

12.3　教科书式Rabin签名方案与教科书式RSA签名方案类似，不同的是前者应用了Rabin陷门置换（参见 11.2 节）。说明教科书式Rabin签名具有下面的属性：敌手可以应用适应性选择消息攻击得到私钥。

12.4　另外一种试图构造安全的RSA签名方法（除了散列的RSA）为编码的RSA。这里公钥

和私钥与教科书式RSA一样。一个公共的编码函数enc是固定的，并且对m的签名计算为$\sigma := [\mathsf{enc}(m)^d \bmod N]$。

(1) 在编码的RSA中如何实现验证？

(2) 讨论为什么合适选择的编码函数可以阻止在 12.3.1 节中描述的"无消息攻击"。

(3) 说明编码的RSA应用编码函数$\mathsf{enc}(m) = 0||m||0^{l/10}$是不安全的（其中$l = ||N||$，$|m| = 9l/10 - 1$，并且$m$不是全 0 消息）。

(4) 说明编码的RSA应用编码函数$\mathsf{enc}(m) = 0||m||0||m$是不安全的（其中$|m| = (||N|| - 1)/2$，并且$m$不是全 0 消息）。

12.5 说明构造方案 4.5 中以任何定长MAC构造变长MAC的方法也可以用于（通过适当改造）以任何消息长度为定长$l(n)$的签名方案构造一个消息长度为任意长的签名方案。$l(n)$的最小长度可能是多少？此构造方案与 Hash-and-Sign 相比有哪些优缺点？

12.6 设f为一个单向置换（类似定义 6.2 中介绍的）。考虑下面的签名方案，其消息属于集合$\{1, \cdots, n\}$：

(1) 为了生成密钥，随机选择$x \leftarrow \{0,1\}^n$和设置$y := f^n(x)$。然后公钥为y，私钥为x。

(2) 签名消息$i \in \{1, \cdots, n\}$，输出$f^{n-i}(x)$（其中$f^0(x) \stackrel{\mathsf{def}}{=} x$）。

(3) 通过检查是否有$y \stackrel{?}{=} f^i(\sigma)$，验证消息$i$关于公钥$y$的签名$\sigma$。

① 说明上面的方案不是"一次性签名方案"。给定一个对消息i的签名，对什么消息j敌手可以输出一个伪造签名？

② 证明不存在PPT敌手在给定i的签名时可以输出一个消息$j > i$的伪造，其中例外的概率是可以忽略的。

③ 提出如何改造上面的方案，从而得到一个"一次性签名方案"。

提示：公钥中包括两个值y，y'。

12.7 考虑Lamport"一次性签名方案"。描述一个敌手如果得到自己选择的两个消息的签名后，可以伪造任何消息的签名。

12.8 Lamport"一次性签名方案"在公钥中使用$2l$个值签名长度为l的消息。考虑下面的变体：私钥由$2l$个值$x_1, \cdots, x_{2l}$组成，公钥包含值$y_1, \cdots, y_{2l}$，其中$y_i = f(x_i)$。消息$m \in \{0,1\}^{l'}$以单射的方式映射到一个大小为l的子集$S_m \subset \{1, \cdots, 2l\}$。为了签名$m$，签名者透露$\{x_i\}_{i \in S_m}$。证明此方法给出了一个"一次性签名方案"。此方案支持的最大消息长度l'是多少？

12.9 一个强"一次性签名方案"满足下列条件（非正式的）：给出一个消息m的签名σ，σ'为m'的一个有效签名，输出$(m', \sigma') \neq (m, \sigma)$是不可行的（注意，允许$m = m'$）。

(1) 给出强一次签名的正式定义。

(2) 假设存在单向函数，说明一个单向函数，对于它的Lamport方案不是一个强"一次性签名方案"。

(3) 基于本书使用的任何假设构造一个强"一次性签名方案"。

提示：在 Lamport 方案中使用一个特殊的单向函数。

12.10 设(Gen, H)是一个抗碰撞散列函数，其中 H 将长度为$2n$的串映射到长度为n的串。证明函数簇$(\mathsf{Gen}, \mathsf{Samp}, H)$为单向的（参考定义 7.70），其中Samp是对一个长为$2n$的随机串取样的平凡算法。

12.11 在 12.6.2 节末尾，说明了一个伪随机函数可以用于使构造方案 12.12 无状态。同样的方法在 12.6.1 节中描述的基于路径的方案中也可用吗？如果适用，简要描述一个构造方案，并且给出证明。如果不适用，解释为什么，并且修改此方案得到一个无状态的变体。

12.12 证明定理 12.14。

12.13 假设撤销证书按照下面的方式处理：当用户Bob声称对应于公钥pk_B的私钥被偷时，用户将此事实的描述进行签名（对应于pk_B），然后发送给CA。一旦接受这样一个签名的消息，CA将撤销原来的证书。

解释在此种情况中，为什么CA不必检查Bob的身份。特别是，解释为什么不用关心偷得Bob私钥的敌手可以伪造签名（对应于pk_B）。

第 13 章　随机预言机模型中的公钥密码系统

在前三章中，讲述了公钥加密方案以及基于多种假设的数字签名的构造方法。但是，已讨论并分析过的大部分可证明安全的方案并不特别有效。具体来说：

(1) 10.4.3 节给出一个基于 RSA 假设，可证明安全的加密方案（参见定理 10.19），但是该方案的效率和在 10.4.1 节中描述的“教科书式 RSA 加密”方案的效率相比，有一定差别。事实上，能够和“教科书式 RSA”加密方案的效率近似，且基于 RSA 问题的安全加密方案目前是未知的。（满足$\ell = \Theta(n)$的填充 RSA 加密方案是有效的，但是还不知道其安全性可从 RSA 问题是困难的这个假设得出。）

(2) 没有说明任何在选择密文攻击下安全的公钥加密方案。尽管基于判定性 Diffie-Hellman 及其他假设的有效方案已经存在，但是，基于 RSA 假设的实用方案还不存在。

(3) 前面介绍中仅仅只有一个数字签名方案在适应性选择消息攻击下是存在性不可伪造的。该构造方法，（在 12.6.2 节中有说明）并不是非常实用。基于本书介绍的任何假设，能够被证明是安全的，并且其效率能够和“教科书式 RSA 签名”方案近似，这样的方案到目前为止还没发现。

以上三点均成立，即使假设有效率的伪随机函数（能通过分组加密，比如 DES 或是 AES 来实例化）和/或有效率的抗碰撞散列函数（能通过密码学散列函数，比如 SHA-1 来实例化）。总之，非常少的公钥密码系统同时具备：①足够高的效率，应用于实践；②基于“标准的”密码学假设（例如 RSA、因数分解，或者是 DDH 假设）是可证明安全的。[①]

这种情况对密码学家来说是一个挑战。密码学家们继续研究提高现有方案的效率，提出新的假设，并且探索使用现有的假设，能够达到的最高效率上界。与此同时，在实际中留下一个问题：应该使用什么样的方案。有人建议，简单地选择当今可用的可证明安全方案中最有效率的。现实上，通常是人们宁愿什么都不使用，也不会去使用一个无效率的方案。另外，当前某些解决方案甚至在将来也不可能是实用的。[②]

当然，另外一种可能性就是使用一个高效率的但是完全临时的密码系统，并且没有验证其安全性，只给出一个事实：设计者尝试攻击该方案，但是没有成功。它违背了目前为止所论述的严格的现代密码学方法的重要性，所以应清楚这不能被接受。使用一个仅仅看起来是“难以攻破”的方案，就是将自己暴露在更聪明、能够攻破该方案的敌手面前。所以必须寻找更好的选择。

13.1　随机预言机方法学

在实际中获得很大成功的一种方法，是在“完全严格的安全性证明”和“没有证明”之间

① 一个例外就是 El Gamal 加密方案，不仅仅是有效的，并且能够基于 DDH 假设证明是安全的。

② 基于上面的讨论，证实了一个疑虑，为什么 El Gamal 加密方案没有在实践中得到广泛采用，对此的确没有好的解释。

提供一个“中间地带”。该方法就是在证明密码学方案的安全性中，引入一个理想模型。尽管该理想模型可能并不能准确地反映现实，但是至少能够从理想模型下的证明中，推出对一个方案设计合理性的可信度度量。只要该模型是合理的，这样的证明当然比没有任何证明要好。

该方法最流行的例子就是“随机预言机模型”，该模型假定了一个公共的、随机选择的函数H的存在，对该函数的评估只能通过“问询”一个预言机，它可以被认为是一个“魔术盒”，当给出输入x时，返回值$H(x)$。下一节中将准确地讨论其含义。为了区分，到目前为止使用的模型，其中不存在随机预言机，通常被称作“标准模型”。

没有人严肃地声称随机预言机的存在（尽管有建议说，在实际中能够通过使用一个可信方来实现随机预言机）。但是，通过下面的两步随机预言机提供了一个正式的方法学，该方法能被用来设计和证明密码学方案：

(1) 一个方案被设计，并且在随机预言机模型中被证明是安全的。也就是说，假设真实世界中包含随机预言机，并且基于这个假设，来构造和分析密码学方案。标准的密码学假设（前面已经介绍的那些类型）也将在的安全性证明中被使用到。

(2) 当希望在现实世界中实现该方案时，没有随机预言机，取而代之，方案中的随机预言机H，是用一个密码学散列函数$\hat{H}$（比如 SHA-1）作适当修改后的实例。即：当要求一方问询预言机来获得值$H(x)$的每个时刻，该方就自己计算$\hat{H}(x)$来代替。

所希望的是：在第二步中使用的密码学的散列函数在仿真随机预言机方面是“足够优秀的”，从而在第一步中给出的安全性证明会在该方案的现实世界实例化后依旧成立。虽然现今还不存在理论上的验证支持这个希望，但是事实上现有设计的方案能够在随机预言机模型下被证明是安全的，但在第二步中“无论随机预言机是如何被实例化的”都是不安全的。而且，作为一个实际的问题，并不清楚散列函数“很优秀”的仿真一个随机预言机准确地意味着什么，也不清楚这是否是一个可达到的目标。基于这些原因，在随机预言机模型下对方案的安全性证明应该被看作是提供了一种证据：该方案没有“内在的设计缺陷”，但是不应该被认为严格地证明了任意现实世界中对方案的实例化都是安全的。在 13.1.2 节中，将对如何解释随机预言模型中的证明进行深入的讨论。

13.1.1 随机预言机模型

首先我们来确定随机预言模型需要什么。理解随机预言模型的一种好方法如下：把“预言机”比喻成一个简单的盒子，接受一个二进制串作为输入，并且返回一个二进制串作为输出。这个盒子的内部的工作是未知的，且是高深莫测的。每个人（包括诚实方以及敌手）能够和这个盒子交互。这样的交互包括输入一个二进制串x作为输入，并且接受到一个二进制串y作为输出。称此交互过程为“向预言机问询x”，并且把x称作向预言机的“问询”。向预言机的问询都被假设是秘密的，这样，如果某方向预言机问询输入x，那么没有其他的人知道x，甚至不知道这一方曾询问过预言机。这一点是很有意义的，因为对预言机的调用，在现实中对应于本地秘密地对密码学散列函数求值。

可以保证这个盒子的输出是一致的，也就是说，如果盒子对一个特定的输入x输出y，那么当再次给定相同的输入x时，它仍然输出相同的回答y。这意味着能够把这个盒子看作是一个函数H。即依据该盒子的输入/输出的特点，可简单地定义函数H。为了方便，因此说“问询H”而不是问询盒子。没有人“知道”整个函数H（除了盒子本身），顶多已知：到目前为止，已经明确问询过H的输入串的返回值。

现在，讨论随机选择函数H的意义。任何将n比特输入映射到$\ell(n)$比特输出的函数H能够被看作是一张表格，这张表格指示出了每个可能的输入$x \in \{0,1\}^n$，对应的输出值$H(x) \in \{0,1\}^{\ell(n)}$。使用词典的顺序来排列输入，意味着任意这样的函数能够用一个长度为$2^n \cdot \ell(n)$比特的字符串来表示，而且反过来说，每个这样长度的字符串都能够被看作是将n比特输入映射到$\ell(n)$比特输出的函数。一个直接的推论是：存在着$U \stackrel{\text{def}}{=} 2^{\ell(n)\cdot 2^n}$个不同的函数，具备这样规定的输入和输出长度。随机均匀地挑选这个类型的函数H，意味着从这些U个可能性中均匀地选择H。在已经描绘的随机预言机模型中，这相当于通过选择一个H来初始化预言机，并且依据H来做出预言机回答。注意，在任意的物理设备中存储代表H的字符串或者表格将需要一个（和输入长度）成指数关系的比特数，所以就算是中等大小的输入，也是我们所不希望的。

一个等价的，但通常更方便的假想“均匀随机地选择一个函数H”的方法是：想象在需要的时候，“实时”地生成随机输出。具体而言，设想该函数是通过具有$\{(x_i, y_i)\}$数对的表格来定义的，表格初始化为空。当预言机接收到一个对x的问询，它首先检查在表格中是否存在某对(x_i, y_i)，满足$x = x_i$，如果存在，那么返回对应的y_i。否则，选择一个随机的字符串$y \in \{0,1\}^{\ell(n)}$，返回y作为回答，并且预言机在表格中存储(x, y)，这样如果相同的输入再次被问询，就能够返回相同的输出。如果有人设想在实践中如此操作，那么这就远离了起初“固定”函数H的构想，即在开始运行某个密码学方案之前，“一次生成，多次使用”函数H。但是，从各方和预言机交互的角度来看，这是完全等价的。注意到，当H被定义在例如$\{0,1\}^*$的无限域上时，将H看作“实时”选择输出值也让事情（从技术上和概念上）变得简单得多。

在 3.6.1 节中定义伪随机函数时，也提到问询随机函数预言机的算法。以免混淆，注意 3.6.1 节中对随机函数的使用和本节对随机函数的使用是完全不同的。在 3.6.1 节中，随机函数是一种定义方法，用来定义一个具体的有密钥函数是伪随机的。在随机预言机模型中，随机函数被用作“构造原语的一部分”，并且因此必须以某种方法在实际操作中实例化，以得到具体的原语实现。

随机预言机模型下的安全证明。（读者可以在首次阅读时跳过这一部分内容，在阅读了 13.2 和 13.3 节中的证明之后，应该会对这一部分内容理解得更清楚。）

在随机预言机模型下的定义和安全性证明和目前熟悉的标准模型下的相对应的部分有一点不同。在标准模型中，有一个具体的方案Π，一个为Π定义的实验$\mathsf{Expt}_{\mathcal{A},\Pi}$，任意敌手$\mathcal{A}$，以及值$\gamma$代表希望得到的某个“坏”事件的最大概率（比如，对于加密来说，$\gamma = \frac{1}{2}$，对于签名来说，$\gamma = 0$）。对Π的安全性的定义，则采用下面的一般形式：方案Π是安全的，如果对任意概率多项式时间的敌手$\mathcal{A}$，有

$$\Pr[\mathsf{Expt}_{\mathcal{A},\Pi}(n) = 1] \leqslant \gamma + \mathsf{negl}(n)$$

其中，概率计算来源于运行Π的各方以及敌手$\mathcal{A}$的随机选择。使用Π的各方（在实际操作中）将做出随机的选择，因而上式保证了实际操作中使用该方案的安全性。

在随机预言机模型中，一个方案Π将会用依赖预言机H的方式来定义。一个具体的方案仅仅是通过选定H而获得，这里用Π^H表示用这种方式获得的方案。（在后面的几节中，将不再明确说明这一点。）对Π的安全性的定义则采用下面的一般形式：如果对任意概率多项式时间敌手$\mathcal{A}$，有

$$\Pr[\mathsf{Expt}_{\mathcal{A}^H,\Pi^H}(n) = 1] \leqslant \gamma + \mathsf{negl}(n)$$

则Π是安全的。（同样也给予$\mathcal{A}$访问H预言机的能力，因为对$\mathcal{A}$来说这是唯一计算H的方式。

同样在后面的几节中，将不再明确说明这一点。）但是，现在上面的概率计算来源于H的随机选择，以及运行Π的各方的随机选择和敌手$\mathcal{A}$的随机选择。正是因为现在概率也来源于H的“随机”选择，认为H是“随机”预言机。

为了在现实世界中使用Π，某个H必须被指定。但不幸的是，对任意特定的函数H，Π的安全性将不能得到保证，Π的安全性仅在随机选择H时有保证。（这和另一事实形成类比：一个方案在标准模型下是安全的，并不保证对诚实方随机选择的任意特定集合也是安全的，其实，对这些随机选择是以高概率安全的。）这就是很难证明以下论断的原因：使用预言机H的任何具体的实例（通过确定性函数如 SHA-1），得到Π在实际操作中的实现，这一实现是安全的。（SHA-1 可能是使得方案碰巧不安全的函数之一。）一个额外的困难就是，一旦一个具体的函数H被指定了，敌手$\mathcal{A}$不再被限制仅能问询作为预言机的H，而是能够去查看H的代码，并且将这些作为额外的信息用到攻击训练中。

随机预言机模型的简单实例。这里给出的例子是相当简单的，没有使用随机预言机模型能够提供的全部能力，并且没有真正的说明任何随机预言机方法论的限制。举这些例子的意图仅仅是为了使读者更容易理解该模型。

接下来，假设一个随机预言机，其将n_1比特的输入映射为n_2比特的输出，其中$n_1, n_2 \geqslant n$，n为安全参数。（因而从技术上来说，n_1, n_2是关于n的函数。）

随机预言机作为单向函数。首先说明一个随机预言机像一个单向函数那样工作。注意，并没有说一个随机预言机是单向函数，因为（正如在前面讨论的那样）随机预言机并不是一个固定的函数。相反，这里声称的是：任意多项式时间敌手$\mathcal{A}$在下面的实验中，成功的概率是可忽略的：

(1) 选择一个随机函数H。

(2) 选择一个随机输入$x \in \{0,1\}^{n_1}$，计算$y := H(x)$。

(3) $\mathcal{A}$得到y，如果$\mathcal{A}$输出一个值x'，满足$H(x') = y$，则成功。

需要强调：在实验中$\mathcal{A}$能够询问预言机H任意多次数。（假设H对于任何人来说都是可访问的。）这一点是有意义的，因为在真实情况中，预言机被一个具体的函数代替，该函数对每个人来说都是已知的，包括敌手。

现在证明为什么任意多项式时间$\mathcal{A}$在上面的实验中成功的概率是可忽略的。不失一般性地假设：$\mathcal{A}$从来没有向H做出两次相同的问询，并且$\mathcal{A}$输出的值x'被$\mathcal{A}$用来向预言机问询过。进一步假设，如果$\mathcal{A}$曾问询过满足$H(x_i) = y$的x_i，那么$\mathcal{A}$就成功了。（这仅仅意味着$\mathcal{A}$没有表现得太愚蠢，即没有在找到正确答案的时候不输出该正确答案。）那么上面实验中$\mathcal{A}$成功的概率和下面实验中$\mathcal{A}$成功的概率正好是相等的（为了理解其原因，回想前一节中关于随机函数的“实时”选择）：

(1) 选定一个随机的$x \in \{0,1\}^{n_1}$，将一个随机值$y \in \{0,1\}^{n_2}$指定给$\mathcal{A}$。

(2) 每次$\mathcal{A}$向随机预言机作问询x_i，做以下工作：

① 如果$x_i = x$，那么$\mathcal{A}$成功。

② 否则，选择一个随机$y_i \in \{0,1\}^{n_2}$。如果$y_i = y$，那么$\mathcal{A}$成功；否则，将y_i作为对问询的回答返回给$\mathcal{A}$，并且继续实验。

令q为$\mathcal{A}$向预言机问询的次数，因为$\mathcal{A}$在多项式时间内运行，故满足$q = \mathsf{poly}(n)$。因为y是完全独立于x的，对于某个i，$\mathcal{A}$因为问询$x_i = x$而成功的概率最多为$q/2^{n_1}$。更进一步说明，当问询$x_i \neq x$时，因为回答y_i是随机选择的，所以对于某个i，$\mathcal{A}$因为$y_i = y$而成功的概率最多为

$q/2^{n_2}$。因为$n_1, n_2 \geqslant n$，因此$\mathcal{A}$成功的概率最多为$2q/2^n = \mathsf{poly}(n)/2^n$，该概率是可忽略的。

随机预言机作为抗碰撞散列函数。要看到随机预言机表现得很像抗碰撞散列函数并不是一件非常困难的事情。也就是说，在下面的游戏中，任意多项式时间的敌手$\mathcal{A}$成功的概率是可忽略的：

(1) 选择一个随机函数H。

(2) 如果$\mathcal{A}$输出x, x'，满足$H(x) = H(x')$，但是$x \neq x'$，则$\mathcal{A}$成功。

为了说明这些，先不失一般性地假设，$\mathcal{A}$仅仅输出其之前问询过预言机的值x, x'，并且$\mathcal{A}$从来不向预言机做出两次相同的问询。令$\mathcal{A}$向预言机的问询为$x_1, \cdots, x_q$，满足$q = \mathsf{poly}(n)$，很清楚的是，$\mathcal{A}$成功的概率上界是由$H(x_i) = H(x_j)$的概率决定的，其中$i \neq j$。将一个随机H的选择看成是被“实时”计算的，这正好等于独立地、均匀随机地选择ℓ个字符串$y_1, \cdots, y_\ell \in \{0,1\}^{n_2}$，对某个$i \neq j$，满足$y_i = y_j$的概率。该问题现在已经转化成为生日问题的一个例子。使用附录 A.4 的结论，得到$\mathcal{A}$成功的概率为$\mathcal{O}(q^2/2^{n_2})$，这是可忽略的。

用随机预言机构造伪随机函数。在随机预言机模型下构造一个伪随机函数是相当容易的（尽管证明过程没有上面的例子那样简单）。令$n_1 = 2n$，以及$n_2 = n$。定义

$$F_k(x) \stackrel{\mathsf{def}}{=} H(k \| x)$$

其中$|k| = |x| = n$。可声称这是一个伪随机函数。也就是说，对任意多项式时间$\mathcal{A}$，在接下来的实验中，$\mathcal{A}$成功的概率最大时超出1/2的部分也是可忽略的：

(1) 随机选择函数H，随机值$k \in \{0,1\}^n$，以及随机比特b。

(2) 如果$b = 0$，敌手被指定访问预言机$F_k(\cdot)$。如果$b = 1$，则$\mathcal{A}$被指定访问一个将n位输入映射到n位输出的随机函数。（该随机函数独立于H。）

(3) $\mathcal{A}$输出一个比特b'，如果$b = b'$，则$\mathcal{A}$成功。

在第二步中，除了实验提供给$\mathcal{A}$的函数预言机，$\mathcal{A}$还能访问H。（考虑到这个原因，H自身没有密钥，不是一个伪随机函数。换句话说，即使可以访问H，在随机预言机模型下的伪随机函数和随机函数必须是不可区分的。）练习 13.1 要求证明上面的构造方法确实能给出一个伪随机函数。

有趣的一点是，甚至对于“计算能力无限制”的敌手，上面的所有证明仍然成立，只要该敌手被限制只能向预言机作多项式次数的问询。在现实世界中并不存在其对应部分：现实世界中任何函数都能够被一个敌手在无限制的时间内求逆，此外，没有办法定义“评估函数”多项式次数是什么意思（因为可能在多个点确定函数的输出，而不必明确地计算函数。）

随机预言机模型中的高级证明机制。前面的例子可能没有阐述清楚，就是随机预言机模型使得某些在标准模型中没有的技术变得可行，这关键是因为敌手仅能访问H的预言机（不能自行对H求值）。这里概述这些证明技术，但是要提醒读者的是，要获得充分的理解，可能要等到看完本章的后面内容，即当这些技术被用来证明某个具体方案的时候。

首先回顾，在前面已经使用过的随机预言机模型的一个明显的特征是：

如果一个敌手$\mathcal{A}$没有确切地向预言机问询某个x，那么$H(x)$的值是完全随机的（至少就$\mathcal{A}$而言是这样）。

这个可能从表面上看起来和伪随机发生器提供的保证非常相似，但是实际上是非常不同的。如果G是一个伪随机发生器，并假定x是均匀随机选择的，且对观察者而言是完全未知的，那么对观察者来说$G(x)$就是伪随机的。但是对H这个随机预言机，只要敌手没有问询过x，那么$H(x)$是真正随机的。即使是出现以下情况这个说法也成立：x是已知的，或者如果x

不是均匀随机选择的，但在选择时还是存在足够的不确定性使得猜测x是困难的。（例如，如果x是一个n比特字符串，其中x的前半部分是已知的，而后半部分是随机的，则$G(x)$可能很容易和随机数区分开来，但是$H(x)$就不会。）

假设要在随机预言机模型下，证明某个方案的安全性。下面将构造一个归约，来说明攻破该方案安全性的敌手$\mathcal{A}$（在随机预言机模型中）是如何被用来破坏某个密码学假设的[③]。作为归约的一部分，必须模仿与$\mathcal{A}$交互的随机预言机。也就是：$\mathcal{A}$将向其认定的预言机提交问询，以及接受回答，但是现在归约本身必须回答这些问询。结果这提供了许多能力。对发起者而言：

归约根据其偏好来选择H的输出值（只要这些值是正确分布的，即均匀随机的分布）。

有时称其为“可编程性（Programmability）”。尽管似乎可编程性没有带来任何优势，但事实是的确有，定理 13.11 的证明说明了这一点。

归约能够模仿随机预言机的事实的另外一个优势是：

归约“知道”所有$\mathcal{A}$向随机预言机所作的问询。

（这和事实并不矛盾，前面提到的，向随机预言机作的问询是“秘密”的。在形式化模型中，这是正确的，这里在归约中将$\mathcal{A}$作为子程序使用。）这一点十分有用，正如定理 13.2 和定理 13.6 的证明过程将说明的那样。

一旦预言机H用一个具体的函数来实例化，那么以上两点都将不存在对应部分。一旦H在某次实验的开始被固定，归约不再能够按照其偏好来设置H的输出值，并且也不再“知道”$\mathcal{A}$对H求值的输入。在下面对这一点将作更详细的讨论。

13.1.2 随机预言机方法是否合理

回想在随机预言机模型下证明安全的方案，在现实世界中是通过使用某个具体函数实例化H来实现的。有了随机预言机模型机制，我们来转向更基础的问题，例如：

(1) 随机预言机模型下的证明在现实世界中保证了什么？

(2) 随机预言机模型下的证明是否和标准模型下的证明有根本上的不同？

在一开始就强调，这些问题当前还没有任何确定性的答案：如今关于随机预言机所扮演的角色，在密码学团体中存在着很多争论，一个活跃的研究领域就是判定在随机预言机模型下的安全性证明究竟在现实中保证了什么。下面我们只能给出争论双方的一些观点。

反对使用随机预言机模型的观点。

抵制使用随机预言机的观点的出发点很简单：正如已经注意到的，没有正式的或者严格的证明：在随机预言机模型下，对某个方案Π的安全性证明意味着任何现实中Π（即一旦随机预言机H用任意特定的散列函数$\hat{H}$实例化）的安全性。这些不仅仅只是理论上的顾虑。一个更基本的问题是：没有具体的散列函数能够表现得像一个“真正的”随机预言机。举个例子，在随机预言机模型下，如果x没有被明确地询问，则值$H(x)$是“完全随机”的，其对应部分是，如果$\hat{H}$没有明确地对x求值，将要求$\hat{H}(x)$是随机的（或者是伪随机的）。这个要求如何在现实中解释？对于发起者，“明确求$\hat{H}$的值”的含义不清楚：要是敌手知道某个计算$\hat{H}$的捷径，而不需要运行$\hat{H}$的实际代码，将会怎样？进一步说明：$\hat{H}(x)$不可能是随机的（或甚至是伪随机的），因为一旦敌手掌握到$\hat{H}$的描述，那么对所有输入的函数值立即就被定义了。

---

③ 相比之下，在前面一节中，单向性，抗碰撞性以及伪随机的证明是信息论意义上的，并没有基于任何密码学假设。

一旦检查在 13.1.1 节中引入的证明技术，随机预言机的局限就更清楚了。作为一个例子，回想一下，证明中使用了这个事实：一个归约算法能够“知道”敌手向随机预言机做出的每个问询。但如果用一个特定的散列函数$\hat{H}$来代替随机预言机，意味着必须在实验的开始向敌手提供$\hat{H}$的描述，但这样$\mathcal{A}$就能够自己求$\hat{H}$的值，不用做出任何明确的问询，因此，该归约将不再具有“知道”$\mathcal{A}$所作的任何问询的能力。（事实上，正如前一段提到的，$\mathcal{A}$计算不同$\hat{H}$值的概念可能不是正确的，当然不可能被正式地定义。）同样的，该归约算法能够按照它的希望来选择H的输出，当使用一个具体的函数时，有些内容显然就不正确了。

即使愿意忽略上面理论性的担心，也有一个实际的难题目前还没有清楚地理解：在实例化一个随机预言机时，一个具体的散列函数“足够优秀”，到底意味着什么？具体地说，假设希望使用（某个经过合适修改的）SHA-1 来实例化随机预言机。对某个特定的方案Π，当使用 SHA-1 来实例化时，假设Π安全可能是合理的，但如果假设在有随机预言机模型的每个方案中，SHA-1 都能代替随机预言机，则该假设就不太合理了。正如早前说的，SHA-1 不是一个随机预言机。设计一个基于随机预言机模型可证明安全方案并不困难，但是当随机预言机被 SHA-1 代替时，该方案可能是完全不安全的。（参见练习 13.2。）

需要强调的是，“SHA-1 表现得像随机预言机”的假设和“SHA-1 是抗碰撞的”或者“AES 是伪随机函数”的假设是明显不同的。（对前者没有一个令人满意的定义，但是对后者有这样的定义）特别是，一个随机预言机和伪随机函数是不同的：后者是带密钥的函数，只有当密钥已知时才能够被求值，当密钥未知时仅仅是“看起来随机”的。比较而言，一个随机预言机是不带密钥的函数，能被任何人求值，且在某种不清楚的定义下，仍然要保持“看起来随机”。

鉴于此，使用随机预言机来证明方案的安全性定性地看是不同于标准模型，后者为了证明方案的安全性，引入新的密码学假设。因此，和在标准模型下的证明比较起来，随机预言机模型下安全性的证明不够令人满意。本书中对各章的分配可被看作是对这一取向的认同。

支持使用随机预言机模型一方的观点。随机预言机模型存在如此多的问题，为什么还要使用它呢？更加确切地说：为什么在现代密码学（特别是当今密码学的实践应用）的发展中，随机预言机模型如此有影响力，而且为什么它仍然被如此广泛地应用？正如将看到的，当前使用随机预言机模型能设计更高效的方案，这些方案比那些已知的在标准模型下构造的方案更加高效。同样的，非常少的（如果有的话）当前使用的公钥密码系统在标准模型下有安全性证明，然而有许多被广泛部署的方案在随机预言机模型下有安全性证明。除此之外，对被考虑的方案来说，在随机预言机下的证明几乎被普遍地认为和在标准模型下的证明同样重要。随机预言机模型已经增强了对使用某些有效方案的信心，并且对扩大密码学算法的应用范围起到了主要的作用。

根本共识是：

在随机预言机模型下的安全性证明比起没有任何证明要好得多。

下面给出这一共识的支持观点：

(1) 在随机预言机模型下，对给定方案的安全性证明表示了该方案的设计是“合理的”，在现实世界对方案的实例化是唯一可能的弱点，因为使用散列函数代替了随机预言机。换而

言之，在随机预言机模型下的证明意味着，在现实中唯一“攻破”该方案的就是“攻破”散列函数本身（用某种方法）。因此，如果散列函数是“足够好”的，就有信心相信该方案的安全性。更进步一说，如果给定的该方案的实例成功的被攻击了，那么只要简单地用一个“更好”的函数来取代这个散列函数即可。

(2) 重要的是，随机预言机模型中证明是安全的方案的真实攻击是很少的。（这里，不包含对练习 13.2 中那样“设计”的方案的攻击，但是要注意，正如在练习 13.3 中给出的曾被广泛使用的方案所指示的，在实例化随机预言机时，要非常地注意。）在设计实践方案时，这给出了证据证明随机预言机是的有用的。

尽管如此，以上最终仅表示直观的推测：在随机预言机模型中的证明是有用的，其他条件都相同时，仍更偏向于在标准模型下的证明。在随机预言机模型下所证明的在现实世界中能保证什么，对其准确的理解仍然是当今密码学所面临的一个重要的研究问题。

总结和推荐。使用一个在随机预言机模型下证明安全的方案明显好于使用一个没有任何安全性证明的方案。但是，当一个合理有效的构造方案在标准模型下的证明是已知时（即使它的效率略低于另一个依赖随机预言机的方案），还是推荐使用这个。

实例化随机预言机。本章已经多次提到，在实际操作中，通过使用“一个经过合适修改的密码学散列函数”，可将随机预言机实例化。事实上，事情是很复杂的，包括许多问题：

(1) 现存的密码学散列函数几乎都是遵循着 Merkle-Damgard 的范式（参见 4.6.4 节），因此当允许变长的输入时，能够相对容易地将散列函数和随机预言机区分开来。（参见练习 13.3）。

(2) 对随机预言机的输出来说，有一个特定的形式是有必要的。例如，预言机应该输出$\mathbb{Z}_N^*$的元素，而不是比特字符串。但密码学的散列函数仅仅输出比特字符串。

如何在实践中处理这些问题的详细讨论超出了本书的范围，这里的目标仅仅是提醒读者注意其中的微妙之处。

13.2 随机预言机模型中的公钥加密

在这一节中，说明多种随机预言机模型下的公钥加密方案。这些构造方法是基于 RSA 难题的，这是因为方便，也是因为这些构造方案在实践中，通常使用 RSA 来实例化。但是应该注意到，所有这些构造方案都能够使用任意的陷门置换来实例化（参加 10.7.1 节）。

13.2.1 选择明文攻击安全

之前已经介绍基于 RSA 的安全公钥加密方案（参见定理 10.19）是没有效率且很难被证明是安全的。但在随机预言机模型中，证明就变得非常简单了。考虑到下面的方案，在构造方案 13.1 中正式地描述。和其他基于 RSA 的方案一样，公钥为$\langle N, e\rangle$，私钥为$\langle N, d\rangle$。为了加密一个消息$m \in \{0,1\}^{\ell(n)}$，发送者选择一个随机$r \leftarrow \mathbb{Z}_N^*$，发送密文：

$$\langle [r^e \mod N], H(r) \oplus m \rangle$$

其中H是作为随机预言机而被模拟的函数，将$\mathbb{Z}_N^*$中的元素映射到长度为$\ell(n)$的串。

构造方案 13.1

令GenRSA如前所述，$\ell(n)$为一个任意多项式。令H为一个函数，对任意N，其定义域为$\mathbb{Z}_N^*$，对任意n，其值被定义在$\{0,1\}^{\ell(n)}$中。构造一个公钥加密方案如下：

- Gen：输入1^n，运行GenRSA(1^n)，计算$\langle N,e,d\rangle$。公钥为$\langle N,e\rangle$，私钥为$\langle N,d\rangle$。
- Enc：输入一个公钥$\langle N,e\rangle$，以及一个消息$m\in\{0,1\}^{\ell(n)}$，选择一个随机$r\leftarrow\mathbb{Z}_N^*$，输出密文$\langle[r^e \bmod N], H(r)\oplus m\rangle$。
- Dec：输入一个私钥$\langle N,d\rangle$，以及一个密文$\langle c_1,c_2\rangle$，计算$r:=[c_1^d \bmod N]$，然后输出消息$H(r)\oplus c_2$。

在随机预言机模型下 CPA 安全的 RSA 加密

假设 RSA 难题与GenRSA相关是困难的，能够凭直觉认为在随机预言机模型下，该方案是 CPA 安全的：因为r是随机选择的，对一个窃听敌手来说，从$c_1=[r^e \bmod N]$中恢复r是不可行的。因此，该敌手将不会向预言机询问r，所以从敌手的角度来看，$H(r)$的值是完全随机的。但是c_2正好是“一次一密”类型的加密，用随机值$H(r)$来对m进行加密，所以敌手不会得到关于m的信息。下面，把该直觉拓展成一个正式的证明。

上面的直觉指出来的，证明过程很大程度依赖于H是随机预言机这个事实，如果H被替换成，比如说一个伪随机发生器G，那么证明是无效的。原因在于，RSA 假设仅仅意味着（粗略地说）一个敌手无法从$[r^e \bmod N]$中恢复出r，但没有提到敌手可能恢复出关于r的部分信息。举个例子，情况可能是这样的：敌手能够计算出r的一半比特，在此情况下，不再声称$G(r)$是伪随机的（因为$G(r)$的伪随机性要求r是完全随机的）。但是，当H是随机预言机时，关于r的部分信息泄露是无关紧要的。只要r没有被直接地向预言机询问过，则$H(r)$是随机的。

定理 13.2 如果 RSA 难题与GenRSA相关是困难的，并且H为一个随机预言机。构造方案 13.1 具有在选择明文攻击下不可区分加密。

证明 令Π表示构造方案 13.1。先证明Π在窃听者存在的情况下，具备不可区分加密。通过命题 10.5 可知，Π是 CPA 安全的。

令$\mathcal{A}$为概率多项式时间敌手，定义

$$\varepsilon(n)\stackrel{\text{def}}{=}\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1]$$

为了方便读者阅读，下面给出实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$的步骤。注意，$H$是随机选择的，并作为该实验的一部分。

(1) 随机选择一个函数H。

(2) 运行GenRSA(1^n)来生成$\langle N,e,d\rangle$。$\mathcal{A}$被指定$pk=\langle N,e\rangle$，可问询$H(\cdot)$。最终，$\mathcal{A}$输出两个消息$m_0,m_1\in\{0,1\}^{\ell(n)}$。

(3) 选择一个随机比特$b\leftarrow\{0,1\}$以及一个随机$r\leftarrow\mathbb{Z}_N^*$，$\mathcal{A}$被指定密文$\langle[r^e \bmod N], H(r)\oplus m_b\rangle$。敌手可继续问询$H(\cdot)$。

(4) 则$\mathcal{A}$输出一个比特b'。如果$b'=b$，该实验的输出被定义为1，否则为0。

在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$执行中，令Query表示事件：在执行期间，$\mathcal{A}$向随机预言机$H$问询过$r$（其中$r$是用来生成挑战密文的值）。使用Success表示事件$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)=1$。那么

$$\begin{aligned}\Pr[\mathsf{Success}] &= \Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] + \Pr[\mathsf{Success} \wedge \mathsf{Query}] \\ &\leqslant \Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] + \Pr[\mathsf{Query}]\end{aligned}$$

其中所有概率都来源于在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$使用到的随机性。下面说明$\Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] \leqslant \frac{1}{2}$，以及$\Pr[\mathsf{Query}]$是可忽略的。定理如下。

声称 13.3 如果H是作为一个随机预言机而被模拟的，那么

$$\Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] \leqslant \frac{1}{2}$$

如果$\Pr[\overline{\mathsf{Query}}]=0$，那么该声明是很显然的。否则，有

$$\begin{aligned}\Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] &= \Pr[\mathsf{Success}|\overline{\mathsf{Query}}] \cdot \Pr[\overline{\mathsf{Query}}] \\ &\leqslant \Pr[\mathsf{Success}|\overline{\mathsf{Query}}]\end{aligned}$$

易知，$\Pr[\mathsf{Success}|\overline{\mathsf{Query}}]=\frac{1}{2}$。这是一个很直接的早前提到过的结果：如果$\mathcal{A}$没有向预言机直接问询$r$，那么从$\mathcal{A}$的角度来看，$H(r)$是完全随机的，因此，$\mathcal{A}$不知道到底是$m_0$或者是$m_1$被加密。（这正是“一次一密”加密方案的情况。）因此，当Query没有发生时，$b=b'$的概率正好是$\frac{1}{2}$。读者应该要让这个直觉变成一个正式的证明。

声称 13.4 如果 RSA 难题与GenRSA相关是困难的，并且H为一个随机预言机，那么$\Pr[\mathsf{Query}]$是可忽略的。

这里的直觉为，如果Query发生的概率不是可忽略的，那么能够以不可忽略的概率，利用$\mathcal{A}$来解决 RSA 难题：指定输入N,e，以及$c_1 \in \mathbb{Z}^*_N$，将公钥$\langle N,e\rangle$以及密文$\langle c_1,c_2\rangle$提供给$\mathcal{A}$，其中$c_2 \leftarrow \{0,1\}^{\ell(n)}$是一个随机的字符串。然后监视$\mathcal{A}$向预言机作的所有问询。（参见 13.1.1 节的讨论。）如果Query发生了，那么其中$\mathcal{A}$的一个问询r满足$r^e = c_1 \mod N$，因此将r作为答案输出，从而以$\Pr[\mathsf{Query}]$的概率解决了 RSA 难题。因为 RSA 难题与GenRSA相关是困难的，所以$\Pr[\mathsf{Query}]$必须是可忽略的。

正式地，考虑下面的算法：

算法$\mathcal{A}'$：

该算法被指定$\langle N,e,\hat{c_1}\rangle$作为输入。

(1) 选择一个随机$\hat{k} \leftarrow \{0,1\}^{\ell(n)}$。$\mathcal{A}'$隐含地有关系$H(\hat{r})=\hat{k}$，其中$\hat{r} \stackrel{\mathsf{def}}{=} [\hat{c_1}^{1/e} \mod N]$，但是注意$\mathcal{A}'$不知道$\hat{r}$。

(2) 对输入公钥$pk=\langle N,e\rangle$运行$\mathcal{A}$。在表格中存储字符串$(\cdot,\cdot)$，初始为空。当$\mathcal{A}$向随机预言机H问询x，按如下方式回答：

① 如果表格中存在条目(x,k)，返回k。

② 如果$x^e = \hat{c_1} \mod N$，返回$\hat{k}$，在表格中存储$(x,\hat{k})$。（在这种情况下，有$x=\hat{r}$，$\hat{r}$如上定义。）

③ 否则，随机选择$k \leftarrow \{0,1\}^{\ell(n)}$，将$k$返回给$\mathcal{A}$，在表中存储$(x,k)$。

(3) 在某个时刻，$\mathcal{A}$输出消息$m_0,m_1 \in \{0,1\}^{\ell(n)}$。

(4) 随机选择$b \leftarrow \{0,1\}$，令$c_2 := \hat{k} \oplus m_b$。将密文$\langle \hat{c_1},c_2\rangle$给予$\mathcal{A}$。如前所述继续使用随

机预言机。

(5) 在$\mathcal{A}$执行的最后（输出猜测比特b'之后），令$x_1, \cdots, x_p$为$\mathcal{A}$做出的所有预言机问询的列表。如果存在一个i满足$x_i^e = \hat{c}_1 \mod N$，输出x_i。

$\mathcal{A}'$在多项式时间内运行。假设$\mathcal{A}'$的输入是通过运行$\mathsf{GenRSA}(1^n)$生成的，获得了(N, e, d)，然后随机选择$\hat{c}_1 \leftarrow \mathbb{Z}_N^*$（参见定义 7.46）。那么当$\mathcal{A}$作为$\mathcal{A}'$的子程序运行时，$\mathcal{A}$所见视图的分布和在实验$\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$中$\mathcal{A}$所见视图的分布相同。（在两种情况下，生成$\langle N, e\rangle$的方式都是相同的。对于一个随机$r \leftarrow \mathbb{Z}_N^*$而言，$\hat{c}_1$等于$[r^e \mod N]$，且$\mathcal{A}$的随机预言机问询用随机串来回答。）因此，事件Query的概率是不变的。更进一步说，每次Query发生，$\mathcal{A}'$就能正确解决指定的 RSA 实例。也就是说，

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A}',\mathsf{GenRSA}}(n) = 1] = \Pr[\mathsf{Query}]$$

因为 RSA 难题与GenRSA相关是困难的，所以$\Pr[\mathsf{Query}]$必须是可忽略的。得证。

■

上面证明中一个关键的地方是，归约算法$\mathcal{A}'$能够知道$\mathcal{A}$向预言机做出的所有问询。就是这个特点，使得每次$\mathcal{A}$做出“正确”的问询（即每次事件Query发生时），$\mathcal{A}'$就能够正确地解决指定的 RSA 难题实例。注意到该证明也使用了一个事实：归约能够隐含地设定$H(\hat{r}) = \hat{k}$，甚至在不知道$\hat{r}$的情况下，这用到了随机预言机证明中的“可编程性”的特点。

13.2.2 选择密文攻击安全

现在还没有看到任何在选择密文攻击下安全的公钥加密方案。虽然这样的方案是存在的，但是比较复杂。更进一步说，基于 RSA 或者是因数分解假设的实用方案还是未知的（相比于 DDH 假设，在 DDH 中这样的方案是存在的）。再次强调，在随机预言机模型中，情况变得非常简单了。在这里说明一个基于 RSA 假设的构造方法。

令GenRSA和前面一节一样，令$\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$为对称密钥加密方案，消息长度为$\ell(n)$。考虑下面在构造方案 13.5 正式地描述了的公钥加密方案。和往常一样，公钥和私钥分别为$\langle N, e\rangle$和$\langle N, d\rangle$。为了加密一个消息$m \in \{0,1\}^{\ell(n)}$，发送方选择随机$r \leftarrow \mathbb{Z}_N^*$，发送密文：

$$\langle [r^e \mod N], \mathsf{Enc}'_{H(r)}(m)\rangle$$

其中H是一个函数，作为随机预言机而被模拟的，将$\mathbb{Z}_N^*$中的元素映射为长度为n的串。

构造方案 13.5

令GenRSA像前面一节中那样，$\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$为对称密钥加密方案，其消息长度为$\ell(n)$。令$H$为一个函数，对任意$N$，其定义域被定义在$\mathbb{Z}_N^*$中，对任意$n$，其值域被定义在$\{0,1\}^n$中。构造一个公钥加密方案如下：

- Gen：输入1^n，运行$\mathsf{GenRSA}(1^n)$来计算$\langle N, e, d\rangle$。公钥为$\langle N, e\rangle$，私钥为$\langle N, d\rangle$。
- Enc：输入一个公钥$\langle N, e\rangle$，以及一个消息$m \in \{0,1\}^{\ell(n)}$，选择一个随机$r \leftarrow \mathbb{Z}_N^*$，计算$k := H(r)$。输出密文$\langle [r^e \mod N], \mathsf{Enc}'_k(m)\rangle$。
- Dec：输入一个私钥$\langle N, d\rangle$，以及一个密文$\langle c_1, c_2\rangle$，计算$r := [c_1^d \mod N]$，令$k := H(r)$。然后输出消息$\mathsf{Dec}'_k(c_2)$。

在随机预言机模型下的 CCA 安全的 RSA 加密

构造方案 13.1 能够被看作是上面的一种特殊情况：使用“一次一密”作为对称密钥加密方案$(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$。事实上，在窃听者存在的情况下，只要$\Pi'$具备不可区分性加密，就可能推广定理 13.2，并证明构造方案 13.5 生成了一个 CPA 安全的公钥加密方案（参见练习 13.4）。这里感兴趣的是，说明只要Π'本身是 CCA 安全的，那么该构造就提供了一个 CCA 安全的公钥加密方案。正如 4.8 节说明的那样，满足安全性概念的有效率的对称密钥加密方案构造起来相对容易。

当H为一个随机预言机时，对 CCA 安全性证明的直觉大致和上一节一样。令r是提供给敌手的用于加密挑战密文的值，将再次区分两种情况：一种情况是敌手没有向随机预言机H问询过r，另一种情况则是问询过r。在第一种情况中，敌手没有掌握到任何关于密钥$k = H(r)$的信息，所以能够把该构造的安全性归约到对称密钥加密方案Π'的安全性上。对于第二种情况，如果 RSA 难题与GenRSA相关是困难的，则该情况发生的概率是可忽略的。对此的证明比前一节要复杂得多，因为这里必须说明：在不知道私钥的情况下，如何去模仿敌手的解密预言机问询。需要说明归约是如何通过合适的方法“编程”随机预言机来做到这些的。

定理 13.6 如果 RSA 难题与GenRSA相关是困难的，在选择密文攻击下，对称密钥加密方案Π'具备不可区分性加密，并且H为一个随机预言机，那么在选择密文攻击下，构造方案 13.5 是不可区分性加密的公钥加密方案。

证明 令Π表示构造方案 13.5，令$\mathcal{A}$为一个概率多项式时间攻击者。定义

$$\varepsilon(n) \stackrel{\text{def}}{=} \Pr[\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1]$$

为了方便，下面介绍实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$的步骤：

(1) 选择一个随机函数H。

(2) 运行$\mathsf{GenRSA}(1^n)$来获得(N, e, d)。$\mathcal{A}$被指定$pk = \langle N, e\rangle$，能问询$H(\cdot)$以及解密预言机$\mathsf{Dec}_{\langle N,d\rangle}(\cdot)$。

(3) 最终，$\mathcal{A}$输出两个消息$m_0, m_1 \in \{0,1\}^{\ell(n)}$。选择一个随机$b \leftarrow \{0,1\}$以及$r \leftarrow \mathbb{Z}^*_N$，并且挑战密文$\langle [r^e \mod N], \mathsf{Enc}'_{H(r)}(m_b)\rangle$给$\mathcal{A}$。

(4) 敌手$\mathcal{A}$可继续问询$H(\cdot)$以及解密预言机，尽管它不能对解密预言机问询给它的挑战密文。

(5) $\mathcal{A}$输出一个比特b'。如果$b' = b$，该实验的输出被定义为1，否则为0。

在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$的执行中，令$\mathsf{Query}$表示事件：在执行期间，$\mathcal{A}$向预言机$H$问询过$r$。使用$\mathsf{Success}$表示事件$b' = b$。则

$$\begin{aligned}\Pr[\mathsf{Success}] &= \Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] + \Pr[\mathsf{Success} \wedge \mathsf{Query}] \\ &\leqslant \Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] + \Pr[\mathsf{Query}]\end{aligned}$$

其中所有的概率都来源于在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中使用到的随机性。需要证明，存在一个可忽略函数$\mathsf{negl}$，满足

$$\Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

并且$\Pr[\mathsf{Query}]$是可忽略的，定理即可得证。

声称 13.7 在选择密文攻击下，如果对称加密方案Π'具备不可区分性加密，并且H为随机预言机，那么存在一个可忽略函数negl，满足

$$\Pr[\mathsf{Success} \wedge \overline{\mathsf{Query}}] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

现在，这个证明比起在前一节中对应声称的证明要复杂得多。正如在该定理前面讨论的直觉，这一部分是因为构造方案 13.1 使用了完美保密的“一次一密”作为其“对称密钥加密组件”，而构造方法 13.5 使用了一个计算安全的对称密钥加密方案Π'。

考虑下面敌手$\mathcal{A}'$对Π'执行选择密文攻击（参阅定义 3.30）：

敌手$\mathcal{A}'(1^n)$：

对某个（未知的）密钥$\hat{k}$，$\mathcal{A}'$可访问一个解密预言机$\mathsf{Dec}_{\hat{k}}(\cdot)$。

(1) 运行$\mathsf{GenRSA}(1^n)$来计算(N, e, d)。选择$r \leftarrow \mathbb{Z}_N^*$，以及令$\hat{c}_1 := [\hat{r}^e \mod N]$。

（$\mathcal{A}'$隐含地有$H(\hat{r}) = \hat{k}$。）

(2) 对输入$pk := \langle N, e\rangle$，运行$\mathcal{A}$。解析串对$(\cdot,\cdot)$并在表格中存储，初始化为空。当$\mathcal{A}$向它的解密预言机问询$\langle c_1, c_2\rangle$时，$\mathcal{A}'$按如下方式回答：

①如果$c_1 = \hat{c}_1$，向解密预言机问询c_2，然后将结果返回给$\mathcal{A}$。

②如果$c_1 \neq \hat{c}_1$，计算$r := [c_1^d \mod N]$。则使用下面讨论的过程来计算$k := H(r)$，将结果$\mathsf{Dec}'_k(c_2)$返回给$\mathcal{A}$。

当需要值$H(r)$时，要么是为了回应$\mathcal{A}$向随机预言机做出的问询，或者是在回应由$\mathcal{A}$向解密预言机做出的问询，如下计算$H(r)$：

①如果在表格中，存在条目(r, k)，返回k。

②否则，选择一个随机的$k \leftarrow \{0,1\}^n$，返回它，并且将(r, k)存储在表格中。

(3) 在某个时刻，$\mathcal{A}$输出$m_0, m_1 \in \{0,1\}^{\ell(n)}$。输出这些相同的消息，接收一个挑战密文$\hat{c}_2$。将挑战密文$\langle \hat{c}_1, \hat{c}_2\rangle$提供给$\mathcal{A}$，如前述那样继续回答$\mathcal{A}$的问询。

(4) 当$\mathcal{A}$输出猜测比特b'时，这个值由$\mathcal{A}'$输出。

很显然，$\mathcal{A}'$在多项式时间内运行。更进一步说，在步骤(3)之后$\mathcal{A}'$从未提交自己的挑战密文$\hat{c}_2$给它自己的解密预言机。这发生的唯一可能是：$\mathcal{A}$向自己的解密预言机提交了挑战密文$\langle \hat{c}_1, \hat{c}_2\rangle$，但这是不允许的。

令$\Pr'[\cdot]$指代在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}',\Pi'}(n)$中一个事件的概率，同样地令$\Pr[\cdot]$指代在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中一个事件的概率。如上定义$\mathsf{Success}$以及$\mathsf{Query}$。也就是说，$\mathsf{Success}$是$b' = b$的事件，$\mathsf{Query}$是$\mathcal{A}$向随机预言机问询$\hat{r}$的事件。关键的观察就是，当$\mathcal{A}$被$\mathcal{A}'$作为一个子程序运行时（在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}',\Pi'}(n)$中），其视图的分布和在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中视图的分布是相同的，直到事件$\mathsf{Query}$发生。如下能看到：

(1) 在每种情况中，指定给$\mathcal{A}$的公钥显然是相同分布的。

(2) 在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}',\Pi'}(n)$中，$\mathcal{A}$的所有随机预言机问询都被一个随机串所回答，和在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中的情况是一样的。

(3) 在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}',\Pi'}(n)$中，$\mathcal{A}$的形式为$\langle c_1, c_2\rangle$的解密问询，其中$c_1 \neq \hat{c}_1$，和在实验中$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$的回答一样。

(4) 只要Query没有发生，在实验$\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A}',\Pi'}(n)$和实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中，$\mathcal{A}$的形式为$\langle \hat{c}_1, c_2\rangle$的解密问询的回答是相同的。能够得出，通过假设$H(\hat{r})$的值为$\hat{k}$，其中$\hat{k}$是一个随机选择的值。（从另一个方面，如果$\mathcal{A}$问询$H(\hat{r})$，即如果Query发生了，那么为了回应这个问询，$\mathcal{A}'$返回一个随机值$k$给$\mathcal{A}$，$k$可能不等于$\hat{k}$。）

但是重点在于，如果Query没有发生，那么$\mathcal{A}$视图的分布在两个实验中是相同的。更进一步说，$\Pr'[\mathsf{Query}] = \Pr[\mathsf{Query}]$（这是遵循一个事实：实验是相同的，直到Query发生），因此$\Pr'[\overline{\mathsf{Query}}] = \Pr[\overline{\mathsf{Query}}]$。所以

$$\begin{aligned}\mathsf{Pr}'[\mathsf{Success}] \geqslant \mathsf{Pr}'[\mathsf{Success} \wedge \overline{\mathsf{Query}}] &= \mathsf{Pr}'[\mathsf{Success}|\overline{\mathsf{Query}}] \cdot \mathsf{Pr}'[\overline{\mathsf{Query}}] \\ &= \mathsf{Pr}[\mathsf{Success}|\overline{\mathsf{Query}}] \cdot \mathsf{Pr}[\overline{\mathsf{Query}}] \\ &= \mathsf{Pr}[\mathsf{Success} \wedge \overline{\mathsf{Query}}]\end{aligned}$$

（上面假设$\Pr[\overline{\mathsf{Query}}] \neq 0$，否则即有$\mathsf{Pr}'[\mathsf{Success} \wedge \overline{\mathsf{Query}}] = 0 = \mathsf{Pr}[\mathsf{Success} \wedge \overline{\mathsf{Query}}]$。）因为$\Pi'$是 CCA 安全的，存在一个可忽略函数$\mathsf{negl}$，满足

$$\mathsf{Pr}'[\mathsf{Success}] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

因此

$$\mathsf{Pr}[\mathsf{Success} \wedge \overline{\mathsf{Query}}] \leqslant \mathsf{Pr}'[\mathsf{Success}] \leqslant \frac{1}{2} + \mathsf{negl}(n)$$

证毕。

■

声称 13.8 如果 RSA 难题与GenRSA相关是困难的，且H作为一个随机预言机，则$\Pr[\mathsf{Query}]$是可忽略的。

直觉上讲，$\Pr[\mathsf{Query}]$是可忽略的，其原因和定理 13.2 的证明过程中的原因一样。特别是，如果敌手向H问询$\hat{r}$，其中挑战密文是$\langle [\hat{r}^e \mod N], \mathsf{Enc}'_{H(\hat{r})}(m_b)\rangle$，那么对一个随机选择的值$\hat{r} \in \mathbb{Z}_N^*$，敌手已经从$[\hat{r}^e \mod N]$计算出$\hat{r}$。但是，为了正式地证明，需要构造一个归约算法，该算法使用这样的一个敌手来解决 RSA 难题，因为该归约算法必须在不知道私钥（解密密钥）的情况下，能够回答$\mathcal{A}$的解密问询，所以是困难的。幸运的是，随机预言机模型能够解决：解密一份密文$\langle c_1, c_2\rangle$（先前的解密问询都没有使用相同的初始分量c_1），归约算法生成一个随机密钥k，返回消息$\mathsf{Dec}'_k(c_2)$。它隐含地令$H(r) = k$，其中$r \stackrel{\mathsf{def}}{=} [c_1^{1/e} \mod N]$。注意在这个时候，$r$可能是未知的，并且一般来说，如果不对$N$分解因子，那么归约算法不能够计算出它。因此，该归约必须保证$\mathcal{A}$向随机预言机提出的先前的和后续的问询的一致性（以防将来向预言机问询r）。做以下工作是相对简单：

(1) 当解密一个密文$\langle c_1, c_2\rangle$时，归约首先检查任意先前的随机预言机问询$H(r)$，是否满足$c_1 = r^e \mod N$。如果找到了，那么先前返回$H(r)$的值被用来解密c_2。

(2) 当回答一个随机预言机问询$H(r)$时，该归约计算$c_1 := [r^e \mod N]$，检查任意先前的解密问询是否使用过c_1作为密文的第一分量。如果是，那么之前用来回答解密问询的k值，现在作为$H(r)$的值返回。

一个简单的数据结构处理这两种情况：归约将维护一个表格，存储所有的随机预言机问询和答案，就和定理 13.2 中的一样（和先前声称的证明过程一样），除了现在该表格将包含三元组而不是二元组。两种类型的条目将出现在表格中：

(1) 第一个类型的条目的形式为(r, c_1, k)，满足$c_1 = [r^e \mod N]$。该条目意味着归约已经定义了$H(r) = k$。

(2) 第二个类型的条目的形式为$(*, c_1, k)$，意味着值$r \stackrel{\text{def}}{=} [c_1^{1/e} \mod N]$是未知的。（再次提醒，如果不对$N$分解因子，那么归约算法不能够计算出这个值。）这种条目表明归约隐式地令$H(r) = k$。所以当回答$\mathcal{A}$的一个解密问询$\langle c_1, c_2 \rangle$时，归约将返回消息$\mathsf{Dec}'_k(c_2)$。如果$\mathcal{A}$曾向随机预言机问询$H(r)$，那么该归约算法将返回正确的答案k，因为它将检查表格，寻找任意满足$c_1 = [r^e \mod N]$，且c_1为第二分量的条目。

通过下面的归约算法$\mathcal{A}'$，我们来实现上面的设想。

算法$\mathcal{A}'$：

该算法的输入为$(N, e, \hat{c_1})$。

(1) 选择随机的$\hat{k} \leftarrow \{0,1\}^n$，三元组$(\cdot, \cdot, \cdot)$在表格中存储，初始的时候仅仅包含三元组$(*, \hat{c_1}, \hat{k})$。令$\hat{r} \stackrel{\text{def}}{=} [\hat{c_1}^{1/e} \mod N]$（正是$\mathcal{A}'$要寻找的答案），$\mathcal{A}'$隐式地令$H(\hat{r}) = \hat{k}$。

(2) 输入$pk := \langle N, e \rangle$，运行$\mathcal{A}$。

当$\mathcal{A}$向解密预言机问询$\langle c_1, c_2 \rangle$时，回答如下：

①如果在表格中存在一个条目，其第二个分量为c_1（即该条目要么是(r, c_1, k)这种形式，满足$r^e = c_1 \mod N$，要么是$(*, c_1, k)$这样的形式），令k为该条目的第三个分量。返回$\mathsf{Dec}'_k(c_2)$。

②否则，选择一个随机的$k \leftarrow \{0,1\}^n$，返回$\mathsf{Dec}'_k(c_2)$，在表格中存储$(*, c_1, k)$。

当$\mathcal{A}$向随机预言机询问r时，计算$c_1 := [r^e \mod N]$，如下回答该询问：

①如果在表格中存在一个形式为(r, c_1, k)的条目，返回k。

②如果在表格中存在一个形式为$(*, c_1, k)$的条目，返回k并且在表格中存储(r, c_1, k)。

(3) 在某个点，$\mathcal{A}$输出消息$m_0, m_1 \in \{0,1\}^{\ell(n)}$。选择一个随机比特$b \leftarrow \{0,1\}$并且令$\hat{c_2} \leftarrow \mathsf{Enc}'_{\hat{k}}(m_b)$。把密文$\langle \hat{c_1}, \hat{c_2} \rangle$提供给$\mathcal{A}$，继续和之前一样，回答$\mathcal{A}$的预言机问询。

(4) 在$\mathcal{A}$的执行的最后，如果在表格中存在一个条目，形式为$(\hat{r}, \hat{c_1}, \hat{k})$，那么输出$\hat{r}$。

算法$\mathcal{A}'$执行了早期概述的策略，唯一增加的是：在实验开始的时候，选择一个随机的密钥$\hat{k}$，并且$\mathcal{A}'$隐式地令$H([\hat{c_1}^{1/e} \mod N]) = \hat{k}$。

显然的$\mathcal{A}'$在多项式时间内运行。假设$\mathcal{A}'$的输入是通过运行$\mathsf{GenRSA}(1^n)$来生成的，获得了(N, e, d)，然后随机选择$\hat{c_1} \leftarrow \mathbb{Z}_N^*$（参见定义 7.46）。那么当$\mathcal{A}$作为$\mathcal{A}'$的子程序被运行时，$\mathcal{A}$的分布和在实验$\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$中$\mathcal{A}$的分布是相同的。（让自己确信这一点可能要花些功夫，但是根据观察就容易得知这一点，这个观察是$\mathcal{A}'$提供的预言机回答从未出现过任何不一致。）因而事件Query的概率在两个实验中保持不变。更进一步说，只要Query发生了，那么$\mathcal{A}'$就能够正确地解决给定的 RSA 实例。即

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A}',\mathsf{GenRSA}}(n) = 1] = \Pr[\mathsf{Query}]$$

因为 RSA 难题与GenRSA相关是困难的，所以$\Pr[\mathsf{Query}]$必须是可忽略的。这总结了对声称的证明，定理得证。

■

13.2.3 OAEP

在 13.2.2 节中指定的公钥加密方案提供了一个基于 RSA 假设的相当高效的方法来实现随机预言机模型下的选择密文攻击安全。(如前所述，给出的范例能够通过使用任意陷门置换来实例化，所以能构造基于分解因子困难性的一个有相似安全性的方案。) 虽然对某些应用来说，更有效的方案是理想的。先前方案的主要缺陷为，密文比$\mathbb{Z}_N^*$中的单个元素要长，即使是在加密一个短消息时。

最优非对称加密填充（OAEP）技术消除了这个缺陷，使得当加密短消息时，密文仅仅由$\mathbb{Z}_N^*$中的单个元素组成。(为了加密更长的消息，正如在 10.3 节讨论的，将会使用混合加密。) 技术上，OAEP 是一个填充方法，而不是加密方案，虽然使用该填充技术的加密方案经常被简单地叫做 OAEP。用 RSA-OAEP 来表示 OAEP 填充和教科书式 RSA 加密的结合（从下面的讨论以及构造方案 13.9 中，这一点会变得清晰)。

OAEP 是一种可逆的，随机化的方式，将一个长度为$n/2$的明文消息编码成一个长度为$2n$的串$\hat{m}$。(这和构造方案 13.9 是匹配的，但是也能够推广到其他的消息/编码长度。) 如果令$\mathsf{OAEP}(m,r)$表示使用随机数r对消息m进行编码，则使用 RSA-OAEP 加密消息$m \in \{0,1\}^{n/2}$，公钥为$\langle N,e\rangle$。选择一个随机$r \leftarrow \{0,1\}^n$，然后计算密文：

$$[\mathsf{OAEP}(m,r)^e \mod N]$$

(假设$\|N\| > 2n$。) 要解密密文c，接收者计算$\hat{m} := [c^d \mod N]$，然后尝试去计算$(m,r) := \mathsf{OAEP}^{-1}(\hat{m})$。如果没有这样的逆存在，那么接受者知道该密文为无效的（并且输出一个特殊的错误符号$\perp$)。否则，唯一的逆(m,r)存在，并且接收者输出消息m。对 OAEP 编码的详细说明，在构造方案 13.9 中给出。

上面的内容实际上是一个简化，某些细节被忽略了，其他参数的选择也是可能的。读者如果对 RSA-OAEP 的实现感兴趣的话，可以参考在本章末的参考文献。

RSA-OAEP 使用了两个函数G和H，在分析中，这两个函数建模为相互独立的随机预言机。虽然在 13.1.1 节介绍随机预言机模型时，没有讨论多于一个随机预言机的存在性，但是这可以用很自然的方式来解释。事实上，使用单个随机预言机$\bar{H}$来实现两个相互独立的随机预言机G,H是相当容易的，如可令$G(x) \stackrel{\mathsf{def}}{=} \bar{H}(0x)$以及$H(x) \stackrel{\mathsf{def}}{=} \bar{H}(1x)$。

仅仅利用事实：编码函数 OAEP 是一个单射函数，将$3n/2$比特（即一个$n/2$比特的消息加上一个n比特随机串）映射为$2n$比特，可看到$\mathbb{Z}_N^*$中一个随机元素在 OAEP 原像中的概率为$2^{-n/2}$，是可忽略的。事实上，①编码函数 OAEP 的设计方法，(从直觉上说) 就是要满足找到一个在 OAEP 的像的唯一方法是选择 *m* 和 *r*，然后计算 OAEP(*m*,*r*)。将这个直觉转变为对上面构造方案的正式安全性证明已经超出了本书的范围。这里仅仅提到，如果 RSA 难题与 GenRSA相关是困难的，并且G和H为相互独立的随机预言机，那么对于某些具有公开指数e(包括当$e=3$这种普通情况) 的类型，RSA-OAEP 能够被证明是 CCA 安全的。使用任意 RSA 公开指数，或者更一般的说，其他陷门置换的各种 OAEP 的变体，也已经提出；参见本章末的参考文献。

构造方案 13.9

GenRSA同前节描述，令$G:\{0,1\}^n \to \{0,1\}^n$以及$H:\{0,1\}^n \to \{0,1\}^n$。构造一个公钥加密方案：

- Gen：输入1^n，运行GenRSA(1^{n+1})来计算(N,e,d)，其中$\|N\| > 2n$。[④]公钥为$\langle N,e\rangle$，私钥为$\langle N,d\rangle$。
- Enc：输入一个公钥$\langle N,e\rangle$，以及一个消息$m \in \{0,1\}^{n/2}$，首先选择一个随机的$r \leftarrow \{0,1\}^n$。然后令$m' := m\|0^{n/2}$，计算$\hat{m}_1 := G(r) \oplus m'$。定义$\hat{m} := \hat{m}_1\|(r \oplus H(\hat{m}_1))$。解释$\hat{m}$为$Z_N^*$中的元素，并输出密文$c := [\hat{m}^e \mod N]$。
- Dec：输入一个私钥$\langle N,d\rangle$，以及一个密文c，先计算$\hat{m} := [c^d \mod N]$，然后解析$\hat{m}$为$\hat{m}_1\|\hat{m}_2$，有$|\hat{m}_1| = |\hat{m}_2| = n$。再计算$r := H(\hat{m}_1) \oplus \hat{m}_2$，$m' := \hat{m}_1 \oplus G(r)$。如果$m'$的最后$n/2$比特不是$0^{n/2}$，则输出$\perp$。否则输出$m'$的前$n/2$比特。

RSA-OAEP 加密方案

13.3 随机预言机模型中的签名

已经完成了在随机预言机下对公钥加密的讨论，把注意力转移到全域散列函数（FDH）签名方案上来。尽管全域散列函数同样可能用任意的陷门置换实例化，这里还是描述一个基于 RSA 的方案，叫做 RSA-FDH。

实际上在 12.3.2 节中，已经提到了 RSA-FDH 方案，在那里我们把它叫做散列后 RSA。散列后 RSA 是通过对一个消息的散列值，而不是对消息本身，应用教科书式 RSA 来获得的。复习一下：在教科书式 RSA 签名方案中，一个消息$m \in \mathbb{Z}_N^*$通过计算$\sigma := [m^d \mod N]$来签名。教科书式 RSA 是完全不安全的（尤其是在 12.3.1 节中说明的），面对下面攻击时会非常脆弱：

(1) 敌手能够选择任意的σ，计算$m := [\sigma^e \mod N]$，然后输出(m,σ)作为一个伪造的签名。

(2) 指定（合法生成的）分别为消息m_1和m_2的签名σ_1和σ_2，敌手能够计算出一个对消息$m := [m_1 \cdot m_2 \mod N]$的有效签名$\sigma := [\sigma_1 \cdot \sigma_2 \mod N]$。

在 RSA-FDH（即散列后 RSA），签名者在签名之前，先对m求散列值。也就是，对消息m的签名是$\sigma := [H(m)^d \mod N]$。（参见构造方案 13.10。）在 12.3.2 节中，非正式地讨论过，当H为密码学散列函数时，为什么这样的修改能够预防上面的攻击。现在，易知如果H建模为随机预言机，为什么这些攻击就失效了。

(1) 当H为随机预言机时，则对于任意给定的σ，找到一个m，满足$H(m) = [\sigma^e \mod N]$是困难的。（参见 13.1.1 节的关于为什么随机预言机表现得像单向函数的讨论。）

(2) 如果σ_1和σ_2分别是对消息m_1以及m_2的签名，这意味着$H(m_1) = \sigma_1^e \mod N$以及$H(m_2) = \sigma_2^e \mod N$。但不太可能的是：$\sigma = [\sigma_1 \cdot \sigma_2 \mod N]$为消息$m = [m_1 \cdot m_2]$的一个

④ 这解释了我们不寻常的选择，运行GenRSA时，输入为1^{n+1}而不是1^n。

有效签名，因为没有理由相信$H(m_1 \cdot m_2) = H(m_1) \cdot H(m_2) \mod N$。（如果$H$是一个伪随机预言机，后者发生的概率是可忽略的。）

构造方案 13.10

令GenRSA如前所述，令H为函数，其定义域为$\{0,1\}^*$，值域为$\mathbb{Z}_N^*$（对任意的N）。构造一个签名方案：

- Gen：输入1^n，运行GenRSA(1^n)来计算(N,e,d)，令H的值域为$\mathbb{Z}_N^*$。公钥为$\langle N,e\rangle$，私钥为$\langle N,d\rangle$。
- Sign：输入一个私钥$\langle N,d\rangle$，以及一个消息$m \in \{0,1\}^*$，计算$\sigma := [H(m)^d \mod N]$。
- Vrfy：输入一个公钥$\langle N,e\rangle$，以及一个消息m，以及一个签名σ，当且仅当$\sigma^e \stackrel{?}{=} H(m) \mod N$时，输出1。

RSA-FDH 签名方案。

注意，前述内容仅仅是直觉，事实上，可证明 RSA-FDH 是抵抗上面这些攻击的，作为定理 13.11 的结论，将在下文证明。强调上文非正式的讨论仅当H作为随机预言机时可被证明，如果H“仅仅”是抗碰撞的，目前还不知道如何去证明。

下面证明，如果 RSA 难题与GenRSA相关是困难的，那么在随机预言机模型中，RSA-FDH 在选择消息攻击下是存在性不可仿造的。关于这个结论，我们首先可以考虑到在无消息攻击下（即当敌手不能要求任意签名时）的存在不可伪造性。这里，敌手被限制问询随机预言机，可以不失一般性地假设，如果敌手输出一个声称的伪造签名(m,σ)，那么该敌手已经在之前询问过$H(m)$。令$y_1,\cdots,y_q$表示敌手接收到的，对其q个向随机预言机问询的回答，每个y_i是完全随机的。进一步说，对某个消息伪造一个有效的签名，需要计算这些值的e次方根。因而不难看到，在 RSA 的假设下，敌手输出一个有效的伪造签名的概率是可忽略的（因为计算这些随机元素的e次方根正好是 RSA 难题）。

更正式地说，在无消息攻击中，从一个敌手$\mathcal{A}$伪造一个有效签名开始，构造一个解决 RSA 难题的算法$\mathcal{A}'$。指定输入(N,e,y)，公钥为$pk=(N,e)$，算法$\mathcal{A}'$首先运行$\mathcal{A}$。除了有一个从$\mathcal{A}$的q个问询中随机选择的是用y来回答之外，$\mathcal{A}'$用$\mathbb{Z}_N^*$中的随机元素来回答$\mathcal{A}$的随机预言机问询。假设$\mathcal{A}$输出(m,σ)，满足$\sigma^e = H(m) \mod N$（即$\mathcal{A}$输出一个伪造签名）。如果$\mathcal{A}'$的输入是通过在$\mathbb{Z}_N^*$中随机选择y来生成的，那么当$\mathcal{A}$作为子程序被$\mathcal{A}'$运行时，与在攻击原始签名方案时$\mathcal{A}$所见内容（视图）的分布是相同的。更进一步说，$\mathcal{A}$并不知道，哪一个预言机问询是用y来回答的。所以，问询$H(m)$是用y来回答的概率为$1/q$，在这种情况下，$\mathcal{A}'$可以通过输出$\sigma = y^{1/e} \mod N$，来解决 RSA 难题的指定实例。看到如果$\mathcal{A}$成功的概率为ε，那么$\mathcal{A}'$解决 RSA 难题的概率为ε/q。因为q是多项式，所以结论是：如果 RSA 假设满足，则ε必然是可忽略的。

当敌手被允许自己选择消息的签名，处理这种情况更困难。复杂性来自于上面的归约$\mathcal{A}'$不知道解密指数d，但是现在不得不对$\mathcal{A}$向签名预言机提交的消息计算有效的签名。这个看起来不可能（甚至可能矛盾），直到我们意识到$\mathcal{A}'$能够正确地对消息m计算出一个签名，只要对一个已知的值σ，让$H(m)$等于$[\sigma^e \mod N]$。如果σ是均匀随机选择的，那么$[\sigma^e \mod N]$也是均匀分布的，因此$\mathcal{A}'$仍然“恰当地”模仿随机预言机。这里，用到了一种基本的方法：随

机预言机事实上是“可编程的”。

以上的直觉构成了下面证明过程的基础：

定理 13.11 如果 RSA 与GenRSA相关是困难的，并且H为随机预言机，那么构造方案 13.10 是适应性选择消息攻击下存在性不可伪造的。

证明 令$\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$表示构造方案 13.10，令$\mathcal{A}$为一个概率多项式时间敌手。定义

$$\varepsilon(n) \stackrel{\mathsf{def}}{=} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1]。$$

先描述实验$\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n)$的步骤：

(1) 选择一个随机函数H。

(2) 运行$\mathsf{GenRSA}(1^n)$来获得(N, e, d)。

(3) 敌手$\mathcal{A}$被指定$pk = \langle N, e\rangle$，可能问询$H(\cdot)$以及“签名预言机” $\mathsf{Sign}_{\langle N,d\rangle}(\cdot)$。（当$\mathcal{A}$请求一个消息$m$的签名时，它被回复$\sigma := [H(m)^d \mod N]$。）

(4) 最终，$\mathcal{A}$输出一对值(m, σ)，其中$\mathcal{A}$之前没有请求m的签名。如果$\sigma^e = H(m)^d \mod N$，该实验的输出为1，否则为0。

因为已经讨论过上面过程来自的直觉，直接讨论正式的证明。为了简化而又不失一般性假设：①$\mathcal{A}$从未向随机预言机提出两次相同的问询；②如果$\mathcal{A}$请求消息m的签名，那么它在之前已经问询过$H(m)$；③如果$\mathcal{A}$输出(m, σ)，那么它之前已问询过$H(m)$。

令$q = q(n)$为$\mathcal{A}$向随机预言机提出问询次数的（多项式）上界。考虑下面的算法$\mathcal{A}'$：

算法$\mathcal{A}'$：

该算法的输入指定为(N, e, y^*)。

(1) 选择$j \leftarrow \{1, \cdots, q\}$。

(2) 指定$\mathcal{A}$公钥$pk = \langle N, e\rangle$。在表格中存储三元组$(\cdot, \cdot, \cdot)$，初始值为空。一个条目(m_i, σ_i, y_i)指示出：$\mathcal{A}'$已经令$H(m_i) = y_i$，以及$\sigma_i^e = y_i \mod N$。

(3) 当$\mathcal{A}$向其随机预言机问询$H(m_i)$时，如下作答：

①如果$i = j$，返回y^*。

②否则，选择一个随机值$\sigma_i \leftarrow \mathbb{Z}_N^*$，计算$y_i := [\sigma_i^e \mod N]$，将$y_i$作为答案返回，并且在表格中存储$(m_i, \sigma_i, y_i)$。

当$\mathcal{A}$请求消息m的签名时，令i满足$m = m_i$，如下回答问询[5]：

①如果$i \neq j$，那么在表格中存在一个条目(m_i, σ_i, y_i)。返回σ_i。

②如果$i = j$，那么中止该实验。

(4) 在对$\mathcal{A}$的执行的最后输出(m, σ)。如果$m = m_j$，$\sigma^e = y^* \mod N$，那么输出σ。

很显然，$\mathcal{A}'$是在多项式时间内运行的。假设$\mathcal{A}'$的输入通过运行$\mathsf{GenRSA}(1^n)$来生成，获得(N, e, d)，然后均匀随机地选择$y^* \leftarrow \mathbb{Z}_N^*$。在第一步中，由$\mathcal{A}'$选择出来的标志$j$代表着一个猜测：$\mathcal{A}$的哪一个预言机问询将会对应着$\mathcal{A}$最后输出的伪造签名。当该猜测是正确的时候，$\mathcal{A}$作为一个子程序被$\mathcal{A}'$在实验$\mathsf{RSA\text{-}inv}_{\mathcal{A}',\mathsf{GenRSA}}(n)$中运行时，以及$\mathcal{A}$在实验$\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n)$中的

---

⑤ 这里m_i表示的是向随机预言机提出的第i个询问。回顾前面的假设，如果$\mathcal{A}$请求一个消息的签名，那么它之前已经向随机预言机询问过该消息了。

分布是相同的。这是因为：当作为一个子程序被$\mathcal{A}'$运行时，$\mathcal{A}$的q个随机预言机问询中的每一个，都确实是用随机值来回答的：

①问询$H(m_j)$是用y^*来回答的，y^*是从$\mathbb{Z}_N^*$中均匀随机选择出来的。

②问询$H(m_i)$（$i \neq j$），是用$y_i = [\sigma_i^e \mod N]$来回答的，其中$\sigma_i$是从$\mathbb{Z}_N^*$中均匀随机选择的。因为 RSA 是一个置换，这意味着y_i在$\mathbb{Z}_N^*$也是均匀分布的。

此外，j和$\mathcal{A}$的分布是相互独立的，除非$\mathcal{A}$正好请求了消息m_j的签名。但是，在这样的情况下，$\mathcal{A}'$的猜测是错误的（因为一旦$\mathcal{A}$请求了m_j的签名，它就不能够输出一个消息m_j的伪造签名）。

当$\mathcal{A}'$猜测正确，且$\mathcal{A}$输出了一个伪造签名，那么$\mathcal{A}'$就解决了 RSA 难题的指定实例（因为$\sigma^e = y^* \mod N$，因此σ是需要的y^*的逆）。因为$\mathcal{A}'$正确猜中的概率是$1/q$，有

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A}',\mathsf{GenRSA}}(n) = 1] = \varepsilon(n)/q(n)$$

因为 RSA 与GenRSA相关是困难的，存在着一个可忽略的函数negl，满足

$$\Pr[\mathsf{RSA\text{-}inv}_{\mathcal{A}',\mathsf{GenRSA}}(n) = 1] \leqslant \mathsf{negl}(n)$$

因为q是多项式，于是$\varepsilon(n)$也是可忽略的，证毕。 ■

参考文献和扩展阅读材料介绍

尽管在密码学应用中，使用一个“看上去随机”的函数的想法在之前已经被提出，最著名的是 Fiat 和 Shamir[54]，但是第一个对随机预言机模型的正式论述是由 Bellare 和 Rogaway 提出的[14]。基于具体密码学散列函数的随机预言机的实例化在参考文献[14, 15, 16, 62, 37]中进行了讨论。

关于随机预言机模型的消极结论最早由 Canetti 等在文献[30]中提出：（设计的）在随机预言机模型下安全的方案，在对随机预言机的任何具体的实例化后，是不安全的。

OAEP 是由 Bellare 和 Rogaway 引入的[15]，后来被 PKCS#1 v2.1 标准化（参见 http://www.rsa.com/rsalabs）。对 OAEP 最初的证明后来被发现是有缺陷的。感兴趣的读者可以参考文献[26, 58, 130]找到更多细节内容。

全域散列函数签名方案是由 Bellare 和 Rogaway 在他们最初的关于随机预言机模型的论文中提出的。后来的改进包含在参考文献[16, 35, 36, 62]中，其中第一个已经作为 PKCS#1 v2.1 的一部分被标准化。

练　习

13.1　证明在 13.1.1 节中的伪随机函数构造方案在随机预言机模型下确实是安全的。

13.2　在本练习中，说明在随机预言机模型下被证明是安全的方案，当随机预言机被用 SHA-1 实例化后，该方案就不安全了。（该练习有一点不正式，因为 SHA-1 被定义为有固定的输出长度。尽管如此，它能说明主要想法。）令Π为在标准模型下的安全签名方案，构造一个签名方案Π_y，其中对签名的执行如下：如果$H(0) = y$，那么输出私钥；如果$H(0) \neq y$，那么返回使用Π计算的签名。

(1)证明对任意值y方案Π_y在随机预言机模型下是安全的。

(2)说明存在一个特殊的y，使得当随机预言机用 SHA-1 实例化后，Π_y是不安全的。

13.3 引入一个消息鉴别码$\Pi=(\mathsf{Gen},\mathsf{Sign},\mathsf{Vrfy})$，其中$\mathsf{Mac}_k(m):=H(k\|m)$，函数$H:\{0,1\}^*\to\{0,1\}^n$（注意$k\leftarrow\{0,1\}^n$，并且验证过程是用通常方式执行）。说明如果$H$为随机预言机，那么$\Pi$是一个安全的消息鉴别码。说明如果$H$是通过 Merkle-Damgard 变换而构造的任意具体的散列函数，那么Π不是一个安全的消息鉴别码。

13.4 如构造方案 13.5 用一个对称密钥加密方案Π'来实例化，该方案在窃听者存在情况下，具备不可区分性加密。证明：如果 RSA 与GenRSA相关是困难的，且H为随机预言机，那么这就是一个 CPA 安全的公钥加密方案。

13.5 假设一个公钥加密方案$(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$是单向的，如果任意 PPT 敌手$\mathcal{A}$在下面的实验中，成功的概率是可忽略的：

①运行$\mathsf{Gen}(1^n)$来获得密码对(pk,sk)。

②随机均匀地选择消息$m\leftarrow\{0,1\}^n$，计算密文$c\leftarrow\mathsf{Enc}_{pk}(m)$。

③$\mathcal{A}$被指定pk以及c，输出一个消息m'。如果$m'=m$，则称$\mathcal{A}$成功。

(1)说明一个在随机预言机模型下，基于任何单向公钥加密方案的 CPA 安全的公钥加密方案的构造方法。

(2)一个加密是确定性的公钥加密方案可以是单向的吗？如果不行，给出证明；如果可以，说明一个基于本书中介绍的任意一个假设的构造方法。

13.6 假设三个用户拥有 RSA 公钥$\langle N_1,3\rangle$，$\langle N_2,3\rangle$和$\langle N_3,3\rangle$（即都使用$e=3$）满足$N_1<N_2<N_3$。向每一方发送相同的消息m：选择随机的$r\leftarrow\mathsf{Z}_N^*$，发送每个人相同的密文$\langle[r^3 \mod N_1],[r^3 \mod N_2],[r^3 \mod N_3],H(r)\oplus m\rangle$。

(1) 说明以上方案是不安全的，即使H是建模为随机预言机，敌手也能恢复出m。

提示：参见 10.4.2 节。

(2) 设计一种简单的方法来修改该方案，获得一个 CPA 安全的方案，其中密文长度为$3|m|+\mathcal{O}(n)$。

(3) 设计一个更好的方法，使得密文长度为$|m|+\mathcal{O}(n)$。

13.7 令$\Pi=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$为公钥加密方案，在选择明文攻击下具备不可区分加密，令$\Pi'=(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$为对称密钥加密方案，在选择密文攻击下具备不可区分加密。考虑下面的公钥加密方案Π^*的构造方案。

构造方案 13.12

令$H:\{0,1\}^n\to\{0,1\}^n$为一个函数。构造一个公钥加密方案如下：

- Gen^*：输入1^n，运行$\mathsf{Gen}(1^n)$来计算(pk,sk)，将其分别作为公钥和私钥输出。
- Enc^*：输入一个公钥pk，以及一个消息$m\in\{0,1\}^n$，选择一个随机$r\leftarrow\{0,1\}^n$，输出密文$\langle\mathsf{Enc}_{pk}(r),\mathsf{Enc}'_{H(r)}(m)\rangle$。
- Dec^*：输入一个私钥sk，以及一个密文$\langle c_1,c_2\rangle$，计算$r:=\mathsf{Dec}_{sk}(c_1)$，令$k:=H(r)$。然后输出$\mathsf{Dec}'_k(c_2)$。

如果H为随机预言机，在选择密文攻击下，上面的构造方案是否具备不可区分加密？如果是的，给出证明。如果不是的，那么用来证明定理 13.6 的方法中有哪一点不成立？

常用符号索引

通用符号

- 如果A是一个随机算法，那么$y \leftarrow A(x)$表示输入为x，通过均匀选择的随机带运行A，并且将输出赋给y。用$A(x;r)$表示输入为x，使用随机带r的确定性计算A。
- 如果S是个集合，则$x \leftarrow S$表示x是从S中均匀随机地选择。
- $\wedge$表示的是布尔合取（与操作）。
- $\vee$表示的是布尔析取（或操作）。
- $\oplus$表示的是异或（XOR）操作。该操作能被应用到单个比特或者整个字符串（后者中XOR是按位计算的）。
- $\{0,1\}^n$表示的是所有长度为n的比特串的集合。
- $\{0,1\}^{\leqslant n}$表示的是所有长度大于0且最大为n的比特串的集合。
- $\{0,1\}^*$表示的是所有长度大于0的有穷比特串的集合。
- 0^n和1^n分别表示的是包含n个0和2的比特串。
- $\|x\|$表示（正）整数x表示成二进制形式时的长度，首位的比特为1。注意$\log x < \|x\| \leqslant \log x + 1$。
- $|x|$表示的是二进制字符串x（可能首部有多位为零）的长度，或者表示实数x的绝对值。
- $\mathcal{O}(\cdot), \Theta(\cdot), \Omega(\cdot), \omega(\cdot)$参见附录A.2。
- $x\|y$和(x,y)等价，表示连接字符串x和y。
- $\Pr[X]$表示的是事件X的概率。
- $x := y$意味着将值y赋值给x。
- $\log x$表示的是底数为2的x的对数。

密码学特定的符号

- n是安全参数。
- PPT代表“概率多项式时间”。
- $\mathcal{A}^{O(\cdot)}$表示的是访问预言机$\mathcal{O}$的算法$\mathcal{A}$。
- k通常表示一个密钥（在对称密钥加密和MAC中）。
- (pk, sk)表示的是公钥/私钥密钥对（用于公钥加密和数字签名）。
- negl表示可忽略函数，即函数$\mu(n)$，对任意多项式$p(n)$，存在着一个整数N使得，$n > N$时，满足$\mu(n) < 1/p(n)$。
- poly(n)表示任意一个多项式。
- polylog(n)表示poly$(\log(n))$。
- Func_n表示函数的集合，这些函数将n比特串映射到n比特串。

- IV表示一个初始向量（分组密码操作以及抗碰撞散列函数中用到）。

算法和过程

- G表示的是一个伪随机数产生器。
- F表示的是一个带密钥的函数，该函数通常是伪随机函数或者伪随机置换。
- (Gen, Enc, Dec)分别表示密钥生成、加密、解密的过程，在公钥加密和对称密钥加密中使用。对对称密钥加密的情况，若Gen没有被特别指定，假设 Gen(1^n)的输出$k \leftarrow \{0,1\}^n$是均匀随机选择的。
- (Gen, Mac, Vrfy)分别表示的是在消息鉴别码中，密钥生成过程、鉴别标签生成过程和验证过程。当Gen没有被特别指定，假设Gen (1^n)的输出$k \leftarrow \{0,1\}^n$是均匀随机选择的。
- (Gen, Sign, Vrfy)分别表示的是在数字签名方案中，密钥生成过程、签名生成过程和签名验证过程。
- GenPrime表示一个概率多项式时间算法，输入为1^n，以不可忽略的概率输出一个n比特素数。
- GenModulus表示一个概率多项式时间算法，输入为1^n，输出(N,p,q)，其中$N=pq$（以不可忽略的概率），p和q是n的素数。
- GenRSA表示一个概率多项式时间算法，输入为1^n，输出一个模数N，一个整数$e>0$，满足$\gcd(e,\phi(N))=1$，以及一个整数d满足$ed=1 \mod \phi(N)$。N是两个n比特素数的乘积，出现例外的概率是可忽略的。
- $\mathcal{G}$表示的是概率多项式时间算法，输入为1^n，输出一个循环群$\mathbb{G}$（群$\mathbb{G}$中的操作能够在poly (n)时间内完成计算）的描述，例外的概率是很小的，群的阶为q（$\|q\|=n$），以及生成元$g \in \mathbb{G}$。

数论

- $\mathbb{Z}$表示整数集合。
- $a|b$表示a整除b。
- $a \nmid b$表示a不能整除b。
- $\gcd(a,b)$表示a和b的最大公约数。
- $[a \mod b]$表示的是b作为除数时，a被b除后的余数。注意到$0 \leqslant [a \mod b] < b$。
- $x_1 = x_2 = \cdots = x_n \mod N$意味着$x_1, \cdots, x_n$都是模$N$同余的。

注意：$x = y \mod N$意味着a和b模N同余，而$x = [y \mod N]$意味着当N作为除数时，x等于y被N除后的余数。

- $\mathbb{Z}_N$表示的是模N的加法群，有时也表示集合$\{0, \cdots, N-1\}$。
- $\mathbb{Z}_N^*$表示的是模N的可逆整数乘法群（也就是说，这些整数与N是互素的）。
- $\phi(N)$表示的是$\mathbb{Z}_N^*$中元素的个数。
- $\mathbb{G}$和$\mathbb{H}$都表示的是群。
- $\mathbb{G}_1 \simeq \mathbb{G}_2$表示的是群$\mathbb{G}_1$和群$\mathbb{G}_2$是同构的。如果该同构映射指定为$f$，$f(x_1)=x_2$，则写成$x_1 \leftrightarrow x_2$。
- g通常是群的一个生成元。

- $\log_g h$表示底数为g的h的离散对数。
- $\langle g \rangle$表示的是由生成元g生成的群。
- p和q通常表示素数。
- N通常表示两个有相同长度的不同素数p和q的乘积。
- $\mathcal{QR}_{\mathcal{P}}$是模$p$的二次剩余的集合。
- $\mathcal{QNR}_{\mathcal{P}}$是模$p$的二次非剩余的集合。
- $\mathcal{J}_{\mathcal{P}}(x)$是$x$模$p$的雅可比符号。
- $\mathcal{J}_N^{+1}$是模N雅可比符号为$+1$的元素的集合。
- $\mathcal{J}_N^{-1}$是模N雅可比符号为-1的元素的集合。
- $\mathcal{QNR}_N^{+1}$是模N雅可比符号为$+1$的二次非剩余的集合。

附　录

附录 A　数学知识

A.1　恒等式和不等量

列出一些在本书中多次使用的标准等式和不等量。

定理 A.1（二项式展开式定理）令x，y为实数，n是一个正整数。则

$$(x+y)^n=\sum_{i=0}^{n}\binom{n}{i}x^iy^{n-i}$$

命题 A.2　任意$x\geqslant 1$，有$(1-1/x)^x\leqslant e^{-1}$。

命题 A.3　任意x，有$1-x\leqslant e^{-x}$。

命题 A.4　任意$0\leqslant x\leqslant 1$，有

$$e^{-x}\leqslant 1-\left(1-\frac{1}{e}\right)\cdot x\leqslant 1-\frac{x}{2}$$

A.2　渐进表示法

遵照标准的符号来表达函数的渐进行为。

定义 A.5

$f(n)$、$g(n)$是从非负整数到非负实数的函数，则

- $f(n)=\mathcal{O}(g(n))$意味着存在一个正整数c和n'，使得对于所有的$n>n'$，满足$f(n)\leqslant c\cdot g(n)$。
- $f(n)=\Omega(g(n))$意味着存在一个正整数c和n'，使得对于所有的$n>n'$，满足$f(n)\geqslant c\cdot g(n)$。
- $f(n)=\Theta(g(n))$意味着$f(n)=\mathcal{O}(g(n))$，并且$f(n)=\Omega(g(n))$。
- $f(n)=o(g(n))$意味着$\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$。
- $f(n)=\omega(g(n))$意味着$\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=\infty$。

例 A.6

$f(n)=n^4+3n+500$，则

- $f(n)=\mathcal{O}(n^4)$。

- $f(n)=\mathcal{O}(n^5)$。实际上，$f(n)=o(n^5)$。
- $f(n)=\Omega(n^3\log n)$。实际上，$f(n)=\omega(n^3\log n)$。
- $f(n)=\Theta(n^4)$。

A.3 概率论简介

假设读者熟悉基本的概率理论，具备典型的关于离散数学的本科课程中覆盖的知识。这里简单回顾一些符号和基本事实。

如果E是个事件，$\overline{E}$是事件E的逆事件，也就是说，$\overline{E}$是E不发生的事件。根据定义，$\Pr[E]=1-\Pr[\overline{E}]$。如果$E_1$和$E_2$都是事件，那么$E_1\wedge E_2$表示$E_1$与$E_2$的合取，也就是说，$E_1\wedge E_2$是事件$E_1$和事件$E_2$同时发生的事件。根据定义，$\Pr[E_1\wedge E_2]\leqslant\Pr[E_1]$。如果满足$\Pr[E_1\wedge E_2]=\Pr[E_1]\cdot\Pr[E_2]$，则事件$E_1$和$E_2$是相互独立的。

如果E_1和E_2都是事件，那么$E_1\vee E_2$表示E_1和E_2的析取，也就是，$E_1\vee E_2$表示要么E_1发生、要么E_2发生。根据定义，$\Pr[E_1\vee E_2]\geqslant\Pr[E_1]$。因此，一致限（Union Bound）往往是一个非常有用的上界。

命题 A.7 （一致限）

$$\Pr[E_1\vee E_2]\leqslant\Pr[E_1]+\Pr[E_2]$$

对任意事件$E_1,\cdots,E_k$重复应用一致限，有

$$\Pr\left[\bigvee_{i=0}^{k}E_i\right]\leqslant\sum_{i=0}^{k}\Pr[E_i]$$

令$F,E_1,\cdots,E_n$都是事件，对任意$i\neq j$，满足$\Pr[E_1\vee\cdots\vee E_n]=1$和$\Pr[E_i\wedge E_j]=0$。也就是说，$E_i$隔开了所有可能事件的空间，以至于某一个事件$E_i$发生的概率为1。于是

$$\Pr[F]=\sum_{i=1}^{n}\Pr[F\wedge E_i]$$

举个特例，当$n=2$，$E_2=\overline{E_1}$，则

$$\Pr[F]=\Pr[F\wedge E_1]+\Pr[F\wedge\overline{E_1}]$$

在E_2事件发生的条件下，E_1事件发生的条件概率记作$\Pr[E_1|E_2]$，被定义为

$$\Pr[E_1|E_2]\stackrel{\text{def}}{=}\frac{\Pr[E_1\wedge E_2]}{\Pr[E_2]}$$

只要$\Pr[E_2]\neq0$。（如果$\Pr[E_2]=0$，则$\Pr[E_1|E_2]$无意义。）这个表示E_2发生的情况下，事件E_1发生的概率。根据定义，立即有

$$\Pr[E_1\wedge E_2]=\Pr[E_1|E_2]\cdot\Pr[E_2]$$

即使$\Pr[E_2]=0$，等式依然成立，因为等式的右边乘了零。

现在可以轻松地推导出贝叶斯定理（Bayes' Theorem）。

定理 A.8（贝叶斯定理）如果$\Pr[E_2]\neq0$，则

$$\Pr[E_1|E_2]=\frac{\Pr[E_1]\cdot\Pr[E_2|E_1]}{\Pr[E_2]}$$

证明

$$\begin{aligned}\Pr[E_1|E_2] &= \frac{\Pr[E_1 \wedge E_2]}{\Pr[E_2]} \\ &= \frac{\Pr[E_2 \wedge E_1]}{\Pr[E_2]} \\ &= \frac{\Pr[E_2|E_1] \cdot \Pr[E_1]}{\Pr[E_2]}\end{aligned}$$

A.4 “生日”问题

如果从有N个元素的集合中，均匀地随机选择q个元素，$y_1, \cdots, y_q$，那么存在两个不同的i, j，使得$y_i = y_j$的概率有多大？上述事件称作一个碰撞，并且将该事件的概率表示成$\mathsf{coll}(q, N)$。这个问题和所谓的生日问题相关。生日问题描述的是，需要一个多少人的小组，使得该小组中，某两个人有相同生日的概率达到1/2。为看清关系，用y_i表示小组中第i个人的生日。如果小组中有q个人，就有q个值$y_1, \cdots, y_q$，这q个值均匀地从$\{1, \cdots, 365\}$取值。为简化假设，认为生日是独立地均匀分布在非闰年的365天中。进一步说，匹配生日相当于一个碰撞，也就是说，存在不相等的i和j满足$y_i = y_j$。因此，生日问题的解答应该是，给出一个最小的整数值q，使得$\mathsf{coll}(q, 365) \geqslant 1/2$。（答案可能让你很吃惊，$q = 23$就已足够。）

在本节中，要证明$\mathsf{coll}(q, N)$的近似上界和下界值。总的来说，如果$q < \sqrt{N}$，则一个碰撞的概率是$\Theta\left(q^2/N\right)$。或者，对于$q = \Theta\left(\sqrt{N}\right)$，那么碰撞的概率是一个常数。

一个碰撞概率的上界是很容易得到的。

引理 A.9 N是一个正整数常量，假设从规模为N的一个集合中均匀地、相互独立地随机选取q个元素$y_1, \cdots, y_q$。则存在不同的i, j，使得$y_i = y_j$的概率最多为$\frac{q^2}{2N}$，即

$$\mathsf{coll}(q, N) \leqslant \frac{q^2}{2N}$$

证明 证明的过程是对一致限的简单应用（命题 A.7）。一个碰撞意味着存在不同的i, j，使得$y_i = y_j$。将Coll表示为一个碰撞事件，将$\mathsf{Coll}_{i,j}$记为$y_i = y_j$的事件。很快得出，对任意不相等的i, j，$\Pr[\mathsf{Coll}_{i,j}] = 1/N$。更进一步说，$\mathsf{Coll} = \bigvee_{i \neq j} \mathsf{Coll}_{i,j}$，并且重复应用一致限，意味着

$$\begin{aligned}\Pr[\mathsf{Coll}] &= \Pr\left[\bigvee_{i \neq j} \mathsf{Coll}_{i,j}\right] \\ &\leqslant \sum_{i \neq j} \Pr[\mathsf{Coll}_{i,j}] = \binom{q}{2} \cdot \frac{1}{N} \leqslant \frac{q^2}{2N}\end{aligned}$$

引理 A.10 N是一个正整数常量，假设从规模为N的集合中均匀地相互独立地随机地选出q个元素$y_1, \cdots, y_q$，并且$q \leqslant \sqrt{2N}$。则存在不同的i, j使得$y_i = y_j$的概率至少为$\frac{q(q-1)}{4N}$。也就是说，

$$\mathsf{coll}(q, N) \geqslant \frac{q(q-1)}{4N}$$

证明 一个碰撞意味着存在不同的i, j，使得$y_i = y_j$，用Coll来表示这个事件。用NoColl_i

来表示另外一个事件，该事件的含义是在$y_1,\cdots,y_i$中不存在任何碰撞。也就是说，对于所有的$j<k\leqslant i$，都有$y_j\neq y_k$。$\mathsf{NoColl}_q=\overline{\mathsf{Coll}}$表示的是不存在碰撞的事件。

如果NoColl_q发生了，那么对于所有的$i\leqslant q$来说，NoColl_i必须也已经发生。因此，

$$\Pr[\mathsf{NoColl}_q]=\Pr[\mathsf{NoColl}_1]\cdot\Pr[\mathsf{NoColl}_2|\mathsf{NoColl}_1]\cdots\Pr[\mathsf{NoColl}_q|\mathsf{NoColl}_{q-1}]$$

现在，因为y_1不可能和自己发生碰撞，所以$\Pr[\mathsf{NoColl}_1]=1$。更进一步说，如果事件NoColl_i发生了，那么$\{y_1,\cdots,y_i\}$就包含了i个不同的值。所以，y_{i+1}和其中一个值发生碰撞的概率是$\frac{i}{N}$，因此，y_{i+1}不和这些值发生碰撞的概率是$1-\frac{i}{N}$。这个意味着

$$\Pr[\mathsf{NoColl}_{i+1}|\mathsf{NoColl}_i]=1-\frac{i}{N}$$

所以

$$\Pr[\mathsf{NoColl}_q]=\prod_{i=1}^{q-1}\left(1-\frac{i}{N}\right)$$

对于所有的i，因为$i/N<1$，有$1-\frac{i}{N}\leqslant e^{-i/N}$（不等式 A.3），所以

$$\Pr[\mathsf{NoColl}_q]\leqslant\prod_{i=1}^{q-1}\left(e^{-i/N}\right)=e^{-\sum_{i=1}^{q-1}(i/N)}=e^{-q(q-1)/2N}$$

于是

$$\Pr[\mathsf{Coll}]=1-\Pr[\mathsf{NoColl}_q]\geqslant 1-e^{-q(q-1)/2N}\geqslant\frac{q(q-1)}{4N}$$

在最后一步使用不等式 A.4（$q(q-1)/2N<1$）。

附录 B　算法数论补充知识

在这本书中给出的密码学创建都是有效的（也就是说，运行时间是输入长度的多项式），对于这些构造方法来说，使用有效的（也就是多项式时间）算法来执行基本的数论操作是很有必要的。尽管在某些情况下，这些数论算法是简单的，但是思考其效率仍然是值得的，因为对于密码学应用来说，使用数千个比特长度的整数是很普遍的。在另外一些情况下，算法本身是很巧妙的，对它们执行性能的分析可能需要高深的群论中的结论。

在附录 B.1 中，描述整数算术的基本算法。这里涵盖了加法，减法等常见算法，还有计算最大公因数的欧几里得（Euclidean）算法。也讨论扩展欧几里得算法，对于这个讨论，假设读者已经掌握了 7.1.1 节中的内容。

在附录 B.2 中，说明模算术的不同算法。除了对基本模操作（也就是模加、模减、模乘和求模逆）作了简短的讨论，也讨论针对一些密码学领域之外的很少碰到的难题的算法：模N的乘幂运算（在任意群里）以及在$\mathbb{Z}_N$或者$\mathbb{Z}_N^*$中选择一个随机元素（或者在一个任意群里）。这一节假设读者熟知 7.1 节中所涵盖的基本群论内容。

以上列举的这些内容在书中后半部分隐含地被使用，但阅读这些内容并不是必须的。（特别是那些接受附录中的结论而不关注证明的读者，很容易读懂对定理结论的总结。）附录 B.3 包含着几乎很少用到的资料，如讨论在循环群中如何找到生成元的问题，并且假定 7.3.1 节中的结论成立。（包含这个内容是为了参考和完整性。）

因为这里的目的只是确立哪些问题能够在多项式时间内解决，所以在选择算法和算法描述

的时候，选择简单的而不是高效率的（只要算法能够在多项式时间内运行）。基于这个原因，通常对算法的精确运行时间不感兴趣，而是关注它们是否在多项式时间内运行。那些对实现算法非常感兴趣的读者可以查看其他关于实现效率的资料，如各种对必要计算的加速技术。

附录 B 中的结论总结为以下一些定理。

在本附录中自始至终假定：任何输入整数a，长度为$\|a\|$比特，也就是说，高位比特是 1。在附录 B.1 中，有：

定理 B.1（整数操作） 给定两个整数a, b，在关于$\|a\|$和$\|b\|$的多项式时间内执行以下的操作：

(1) 计算$a+b$以及$a-b$；

(2) 计算ab；

(3) 计算出正整数q和r（$r<b$），满足$a=qb+r$（也就是带余数除法）；

(4) 计算a和b的最大公因数，$\gcd(a,b)$；

(5) 计算整数X, Y，使得$Xa+Yb=\gcd(a,b)$。

接下来的结论在附录 B.2 中证明。

定理 B.2（模操作） 给定整数$N>1$，a以及b，在关于$\|a\|$，$\|b\|$以及$\|N\|$的多项式时间内执行以下操作：

(1) 计算模减法$[a \mod N]$；

(2) 计算出$[(a+b) \mod N]$，$[(a-b) \mod N]$，以及$[ab \mod N]$；

(3) 判断出a在模N的情况下，是否可逆；

(4) 假设a是在模N的情况下，是可逆的，计算出乘法逆元$[a^{-1} \mod N]$；

(5) 求幂运算$[a^b \mod N]$；

下面将定理 B.2(4)一般化到任意群：

定理 B.3（群的求幂运算） 群$\mathbb{G}$是一个乘法群。g是群里的一个元素，b是一个非负整数。g^b能够使用$\text{poly}(\|b\|)$次群操作计算出来。

定理 B.4（选择随机元素） 存在一个随机算法，输入为N，具有以下特征：

(1) 该算法在关于$\|N\|$的多项式时间内运行；

(2) 该算法输出fail的概率可忽略；

(3) 在没有输出fail的条件下，该算法输出一个均匀分布在$\mathbb{Z}_N$的元素。

对$\mathbb{Z}_N^*$存在类似特征的算法。

用$x \leftarrow \mathbb{Z}_N$表示从$\mathbb{Z}_N$中随机选择一个元素$x$，使用上述定理所保证的算法（$\mathbb{Z}_N^*$依此类推）。因为输出fail的概率是可以忽略的，所以忽略它（或视为隐含的概率）。在附录 B.2 中，也会讨论从有限群中选择随机元素的通用情况。

下面定理在附录 B.3 中证明。

定理 B.5（测试并找到生成元） $\mathbb{G}$是一个阶为q的循环群，假定群操作和选择一个随机群元素的操作能够在单元时间内完成。

(1) 存在一个算法，输入q，q的素数分解和给定一个元素$g \in \mathbb{G}$，能够在poly $(\|q\|)$的时间内运行，并且正确地判定g是否是$\mathbb{G}$的生成元。

(2) 存在一种随机算法，输入q和q的素数分解，能够在poly $(\|q\|)$的时间内运行，并且输出$\mathbb{G}$的一个生成元，错误的概率是可忽略的（概率是关于$\|q\|$的表示）。若输出一个生成元，则该生成元是均匀分布在$\mathbb{G}$的生成元中的。

B.1 整数算术

B.1.1 基本操作

从整数的加法/减法、乘法和带余除法开始来探讨算法数论。简单思考一下可知，这些操作可以在以输入数的长度为参数的多项式时间完成，使用标准的小学算法。例如，两个正整数a, b，其中$a > b$，计算a和b相加的和的操作能够在与$\|a\|$成线性关系的时间内完成，通过从最低比特开始，逐步地处理a和b的每个比特，每步都计算出相应的输出比特和进位比特（细节省略）。举另外一个例子，两个有n比特的整数a和b相乘，首先生成一列n个整数，其中长度最大为$2n$（其中每一个都等于$a \cdot 2^{i-1} \cdot b_i$，其中$b_i$是$b$的第$i$个比特），然后将这$n$个整数加起来，得到最后的结果。

尽管小学算法已经足够证明前面提到的问题能够在多项式时间内解决，但有趣的是，这些算法在某些情况下并不是最好的。例如，上面计算两个n比特整数乘法的简单算法其时间复杂度为$\mathcal{O}\left(n^2\right)$，但是有的算法时间复杂度为$\mathcal{O}\left(n^{\log_2^3}\right)$（甚至这也可能不是最好的）。对于日常碰到的数，其中的差别是无关紧要的，但当数很大的时候，这些差别就值得注意了。在密码学应用中，使用比特位数多达数千的整数（例如，$n > 1000$）是很平常的，明智地选择算法是很关键的。

B.1.2 欧几里得和扩展的欧几里得算法

在 7.1 节中的$\gcd(a; b)$，整数a, b的最大公因数就是能够同时整除a和b最大的整数d。陈述一个关于最大公因数的简单命题，然后说明它是如何导致计算gcd的有效算法。

命题 B.6 $a, b > 1$且$b \times a$，则

$$\gcd(a, b) = \gcd(b, |a \mod b|)$$

证明 如果$b > a$，命题得证。所以假设$a > b$，记为$a = qb + r$，q, r都是正整数，并且$r < b$（对照命题 7.1）。注意，因为$b \times a$，所以$r > 0$。因为$r = |a \mod b|$，通过证明$\gcd(a, b) = \gcd(b, r)$来证明该命题。

让$d = \gcd(a, b)$，d能够整除a和b，所以d也能够整除$r = a - qb$。根据对最大公因数的定义，有$\gcd(b, r) \geqslant d = \gcd(a, b)$。

令$d' = \gcd(b, r)$，d'能够整除b和r，所以d'也能够整除$a = qb + r$。根据对最大公因数的定义，有$\gcd(a, b) \geqslant d' = \gcd(b, r)$。

因为$d \geqslant d'$并且$d' \geqslant d$，所以得出结论$d = d'$。

以上的命题预示着递归的欧几里得算法（算法 B.7）来计算两个整数a和b的最大公因数$\gcd(a,b)$。

算法的正确性是从命题 B.6 轻易地得出的。至于它的运行时间，下面说明对于输入为(a,b)，该算法的递归调用次数小于$2\cdot\|b\|$。因为检查b是否整除a以及计算$|a \mod b|$都能够在关于$\|a\|$和$\|b\|$的多项式时间内完成，意味着整个算法能够在多项式时间内完成。

算法 B.7
欧几里得算法GCD
输入：整数a,b，并且$a \geqslant b>0$
输出：a和b的最大公因数
if b整除a
  return b
else return GCD $(b,[a \mod b])$

命题 B.8 考虑到GCD (a_0,b_0)的执行，令a_i,b_i（for $i=1,\cdots,l$）表示对GCD的第i次调用的参数。则对于$0\leqslant i\leqslant l-2$，有$b_{i+2}\leqslant b_i/2$。

证明 因为$b_{i+1}=[a_i \mod b_i]$，经常有$b_{i+1}\leqslant b_i$，所以$\{b_i\}$随着i的递增而递减。固定任意一个i，$0\leqslant i\leqslant l-2$，要证明$b_{i+2}\leqslant b_i/2$。如果$b_{i+1}\leqslant b_i/2$，得证（因为$b_{i+2}<b_{i+1}$），否则，有$b_{i+1}>b_i/2$。因为$a_{i+1}=b_i$，第$i+1$次递归调用是

$$\text{GCD}\,(a_{i+1},b_{i+1})=\text{GCD}\,(b_i,b_{i+1})$$

并且

$$b_{i+2}=[a_{i+1} \mod b_{i+1}]=[b_i \mod b_{i+1}]=b_i-b_{i+1}<b_i/2$$

对第三个等式和最后一个不等式，使用$b_i>b_{i+1}>b_i/2$。

推论 B.9 在算法GCD (a,b)的执行中，对GCD的递归调用最多有$2\|b\|-2$次。

证明 a_i,b_i（$i=1,\cdots,l$）表示第i次对GCD的递归调用的参数。$\{b_i\}$中的元素总是大于0，并且如果$b_i=1$（意味着$b_i|a_i$），该算法就不会再继续递归调用了。前面的命题指出每经过两次循环，$\{b_i\}$减少的倍数至少为2（即至少除以2）。因而GCD的递归调用次数最多为$2\cdot(\|b\|-1)$。

扩展欧几里得算法。 由命题 7.2 可知，对于整数a和b，存在整数X,Y，使得$Xa+Yb=\gcd(a,b)$。简单修改欧几里得算法，称作扩展欧几里得算法，除了计算$\gcd(a,b)$，还可用来找到X,Y。参见算法 B.10。练习 B.1 要求给出扩展的欧几里得算法的正确性证明，在练习 B.2 中，要求证明该算法能够在多项式时间内运行。

算法 B.10
扩展的欧几里得算法 eGCD
输入：整数a,b，并且$a\geqslant b>0$
输出：(d,X,Y)，$d=\gcd(a,b)$并且$Xa+Yb=d$
if b整除a

return $(b, 0, 1)$
else 计算整数q和r，$a = qb + r$并且$0 \leqslant r < b$
$(d, X, Y) :=$ eGCD (b, r) /*注意$Xb + Yr = d$*/
return

B.2 模算术

现在，将注意力转到一些基本的模N算术运算操作上。在本节，除非额外声明，否则认为$N > 1$。使用$\mathbb{Z}_N$指代集合$\{0, \cdots, N-1\}$和该集合中的元素进行模N相加之后得到的群。$\mathbb{Z}_N^*$亦同。

B.2.1 基本操作

针对整数基本算术运算操作的有效算法直接意味着存在相应的模N算术运算操作有效算法。例如，利用对整数进行简单的带余除法，计算模消减$[a \mod N]$能够在关于$\|a\|$和$\|N\|$的多项式时间内完成。接下来假设$\|N\| = n$，考虑a和b的模操作，$a, b \in \mathbb{Z}_N$（注意a, b的长度最大为n。实际上，要求$\mathbb{Z}_N$中的所有元素长度都为n会很方便，必要的时候可在左边填充0）。a和b模N相加，可首先计算出$a + b$，该结果的长度最多为$n + 1$，然后对该值模N。类似地，模N的乘法可以首先计算出ab，该结果的长度最多为$2n$，然后对该值模N。因为加法和乘法以及带余除法都能够在多项式时间内处理，使得模N的加法和乘法存在多项式时间的算法。

B.2.2 计算模的逆元

到目前为止，已经说明了如何去做模N的加法、减法和乘法。除法操作，或者说是计算模N乘法的逆还没有介绍。回想 7.1.2 节中，a（$a \in \mathbb{Z}_N$）的乘法逆元（模N）为元素a^{-1}，$a^{-1} \in \mathbb{Z}_N$，满足$a \cdot a^{-1} = 1 \mod N$。定理 7.7 表明元素$a$有一个逆元当且仅当$\gcd(a, N) = 1$，即当且仅当$a \in \mathbb{Z}_N^*$。因此，使用欧几里得算法，可以在模$N$的情况下，容易地判断出一个给定的元素$a$是否有乘法逆元。

给定N和$a \in \mathbb{Z}_N$，满足$\gcd(a, N) = 1$，由命题 7.2 知存在着整数X, Y，使得$Xa + YN = 1$。回忆$[X \mod N]$是a的乘法逆元；这是因为

$$[X \mod N] \cdot a = Xa = 1 - YN = 1 \mod N$$

满足$Xa + YN = 1$的整数X, Y能够通过 B.1.2 中介绍的扩展欧几里得算法eGCD有效地找出。这样，在多项式时间内计算乘法逆元的算法如下：

算法 B.11
计算模逆元
输入：模N，元素a
输出：$[a^{-1} \mod N]$（如果存在）
$(d, X, Y) :=$ eGCD (a, N)/*注意$Xa + YN = \gcd(a, N)$*/
if $d \neq 1$ return “模N情况下，a不可逆”
else return $[X \mod N]$

B.2.3 模幂运算

一个更有挑战性的任务就是计算模N的幂运算，即计算$[a^b \mod N]$，这里底数$a \in \mathbb{Z}_N$，整数指数$b > 0$（当$b = 0$的时候，这个问题就很简单了。当$b < 0$，且$a \in \mathbb{Z}_N^*$，则$a^b = \left(a^{-1}\right)^{-b} \mod N$，该问题就转化成一个正指数幂运算的情况，只要能用前面章节讨论的办法算出元素的逆元）。注意，在模加和模乘的情况中用到的基本方法（即计算整数a^b，然后对这个中间结果模N消减）在这里不起作用：因为整数a^b的长度有$\|a^b\| = \Theta\left(\log a^b\right) = \Theta\left(b \cdot \|a\|\right)$，所以即使保存中间结果，计算$a^b$仍将需要关于$\|b\| = \Theta(\log b)$的指数时间。

解决这个问题，可以通过在计算的整个过程中反复地做模N消减，而不是等到最后才做模N消减。这样做的效果是在整个计算过程中，保持中间结果的值“小”。即使观察到这个重要过程，设计一种多项式时间的算法来计算模幂仍然是不简单的。考虑下面初步的算法：

算法 B.12
模幂的直接（无技巧）计算算法
输入：模N，底数$a \in \mathbb{Z}_N$，整数指数$b > 0$
输出：$[a^b \mod N]$

```
x = 1
for i = 1 to b:
        x := x · a  mod N
return x
```

因为该算法使用了b次内循环，所以该算法的运行时间仍然是关于$\|b\|$的指数时间。

上面给出的这个简单算法看作是依赖于下面的重复：

$$[a^b \mod N] = [a \cdot a^{b-1} \mod N] = [a \cdot a \cdot a^{b-2} \mod N] = \cdots$$

并且能够很容易写成递归的形式使得对应关系更清楚。从上面的等式可以看到任何依赖于该重复的算法都需要$\Theta(b)$的时间。利用下面的重复来做将更好：

$$[a^b \mod N] = \begin{cases} \left[\left(a^{\frac{b}{2}}\right)^2 \mod N\right] & \text{（当}b\text{为偶数）} \\ \left[a \cdot \left(a^{\frac{b-1}{2}}\right)^2 \mod N\right] & \text{（当}b\text{为奇数）} \end{cases}$$

根据这个方法产生了一种递归算法，叫做“平方乘”（或者叫做“重复平方”），原因很明显，仅需$\mathcal{O}(\log b) = \mathcal{O}(\|b\|)$次模平方/乘法。参见算法 B.13。在该算法中，每次递归之后，b的长度减1。递归调用的次数最多为$\|b\|$。更进一步说，在每次递归调用中执行的操作都能够在关于$\|a\|$和$\|N\|$的多项式时间完成。因此，该算法的整个运行时间为关于$\|a\|$，$\|b\|$和$\|N\|$的多项式时间。仔细研究该算法，看到它最多执行$2 \cdot \|b\|$次模N乘法、加法、消减。为了便于理解，算法 B.13 采取递归的方式来描述。为了提高效率，在实际中更倾向于迭代的算法。参见练习 B.3。

指定a和N，考虑模幂函数$f_{a,N}(b) = [a^b \mod N]$。可以看到计算$f_{a,N}$是容易的。相比而言，计算该函数的逆，也就是说给出合理选择的a，N以及$[a^b \mod N]$，计算出b，普遍认为是困难的。求模幂函数的逆被认为是要解决离散对数问题，相关的细节在 7.3.2 节中讨论。

算法 B.13
有效的模幂算法 ModExp
输入：模N，底数$a \in \mathbb{Z}_N$，整数指数$b > 0$
输出：$[a^b \mod N]$

if $b = 1$ return a
else
  if b是偶数
    $t := \mathsf{ModExp}(N, a, b/2)$
  return $[t^2 \mod N]$
  if b是奇数
    $t := \mathsf{ModExp}(N, a, (b-1)/2)$
  return $[a \cdot t^2 \mod N]$

任意群中的幂运算。上面给出的有效模幂运算的算法是很直接的，使得在任何群中有效幂运算成为可能，只要依赖的群操作能够有效地执行。特别是，如果$\mathbb{G}$是一个群，g是群$\mathbb{G}$中的一个元素，则g^b可以通过最多$2 \cdot \|b\|$次群操作计算出来。

如果$\mathbb{G}$的阶q是已知的，则$a^b = a^{[b \mod q]}$（参考命题 7.49），通过首先对b做模q消减，能够进一步提高计算速度。该结论也能应用在前面描述的模幂算法上。

$$[b \cdot g \mod N] \stackrel{\text{def}}{=} [\underbrace{g + \ldots + g}_{\text{b 次}} \mod N]$$

考虑（加法）群$\mathbb{Z}_N$，有。这给出了一个计算“幂运算”的方法，该方法和前面讨论的方法不同，前面方法依赖于标准的整数乘法然后模消减。比较这两种解决同一个问题的方法，注意到原始算法利用了$\mathbb{Z}_N$特有的信息，特别是它实际上将“指数”b作为$\mathbb{Z}_N$中的一个元素（可能首先对b做了模N消减）。相比而言，“平方乘”算法将$\mathbb{Z}_N$作为一个抽象群。（当然，群操作中模N加法用到$\mathbb{Z}_N$的特性）。

这里的讨论仅仅只是说明有些算法是通用的，也就是说可应用到所有的群中，然而有些群算法依赖于一个特定群或者一类群的特有属性，第 8 章已经给出一些这样的例子。

B.2.4 选择一个随机的群元素

在密码学的应用中，经常要求在一个给定群$\mathbb{G}$中选择一个随机元素（在术语中，“随机”意味着“均匀分布”）。首先在抽象群里处理该问题，然后关注$\mathbb{Z}_N$和$\mathbb{Z}_N^*$的情况。

一个群$\mathbb{G}$中的元素必须通过某种表示方法来指定，如比特串，不失一般性，假设群中的所有元素都用相等长度的串来表述。（注意，每个群元素有唯一的串表示）。例如，如果$\|N\| = n$，则$\mathbb{Z}_N$的元素都能够表示成长度为n的串，这里整数$a \in \mathbb{Z}_N$，在$\|a\| < n$的情况下，在左边填充0。

表示方式不是重点，因为对本书中所有提及的群来说，表示方式可简单地采用一种“自然”方式（类似于上面$\mathbb{Z}_N$的情形）。但是，对相同的群，不同的表示方式会影响执行各种计算的复杂度，所以，为给定的群选择“正确”的表示方式在实践中是很重要的。因为这里的目标只是为了说明需要的操作有多项式时间算法（而不是为了说明已知的最有效的算法），对

这个目的来说，使用准确的表示方式并不是很重要。此外，大部分展示的“高级”算法以“黑盒子”的方式使用群操作，只要群操作能在多项式时间内执行（用某些参数），则算法也能够在多项式时间内运行。

给定一个群$\mathbb{G}$，其中所有的元素都用长度为ℓ的字符串来表示，选择一个随机元素可以通过选择随机ℓ比特的字符串，直到发现一个群$\mathbb{G}$的元素为止。为了获得一个运行时间有界的算法，引入一个参数t来限制该过程的最大重复次数。如果t次循环后，没有成功选择出$\mathbb{G}$中的一个元素，则该算法输出“fail”（另一种可能是输出$\mathbb{G}$中一个任意元素）。即：

算法 B.14
选择一个随机的群元素
输入：一个群$\mathbb{G}$，长度参数ℓ，参数t
输出：$\mathbb{G}$中的一个随机元素

```
for i = 1 to t;
    x ← {0,1}^ℓ
if x ∈ 𝔾 return x
return "fail"
```

很明显，只要该算法不输出“fail”，就能输出一个均匀分布在$\mathbb{G}$中的元素。原因很简单，因为$\mathbb{G}$中的每一个元素等概率地在某个循环中被选择。正式地表述，如果让Fail来表示算法输出“fail”的事件，则对于任何元素$g \in \mathbb{G}$，有

$$\Pr\left[\text{算法输出 } g | \overline{\text{Fail}}\right] = \frac{1}{\mathbb{G}}$$

该算法输出“fail”的概率是多少？在任何给定的循环中，$x \in \mathbb{G}$的概率是$|\mathbb{G}|/2^{\ell}$，因此，t次循环中x均不在$\mathbb{G}$中的概率为

$$\left(1 - \frac{|\mathbb{G}|}{2^{\ell}}\right)^{t} \tag{B.1}$$

在密码学的设置中，常会有一个秘密的参数n，群$\mathbb{G}$（以及ℓ）将依赖于这个参数 n 而不是固定的。正式地表述，固定某一类群（而不是一个群），用$\mathcal{C}$表示，在该类群中，给每个群关联一个值n，然后问：是否可能在以n为参数的多项式时间内，从群$\mathbb{G} \in \mathcal{C}$中选取一个随机元素。也就是说，是否可能从$\mathcal{C}$这一类群中，以多项式时间选取一个随机元素。（技术上的注释：类型$\mathcal{C}$中的每个群都有特定的表示方式。需要$\ell = \text{poly}(n)$，并假设凡是有相同n值的群，拥有相同的长度参数$\ell = \ell(n)$。）

例如，考虑一类群$\mathcal{C} = \{\mathbb{Z}_N | N \in \mathbb{Z}\}$，在该类型的群中，和群$\mathbb{Z}_N$关联的参数为$n = \|N\|$。问：能否在poly (n) $(\|N\|)$时间内，从群$\mathbb{Z}_N$选取出一个随机元素。

算法 B.14 的运行时间和算法输出“fail”的概率之间存在一个权衡，因为增加t将会减少失败的概率，但是同时也会增加最坏情况下的运行时间。对密码学应用来说，需要的算法是，在最坏情况下的运行时间是关于n的多项式且失败的概率可忽略（概率是关于n的表达式）。为了实现这些，每个带有参数n的群$\mathbb{G}$必须满足两个条件：

(1) 在多项式时间poly (n)内判断一个$\ell(n)$位比特串是否为$\mathbb{G}$的元素；

(2) 一个随机的$\ell(n)$位比特串是$\mathbb{G}$中一个元素的概率应该至少为$1/\text{poly}(n)$。

第一个条件的必要性是显而易见的，因为算法 B.14 需要检查是否$x \in \mathbb{G}$。假设第二个条件满足，也就是说，存在一个多项式p，对每个带参数n并且长度参数为$\ell = \ell(n)$的群$\mathbb{G}$（在指定的某个类群$\mathcal{C}$中），一个随机的$\ell(n)$位比特串是$\mathbb{G}$中一个元素的概率至少为$1/p(n)$。设置$t(n) = \lceil p(n) \cdot n \rceil$。该算法输出“fail”的概率为（参考式 B.1）：

$$\begin{aligned}\left(1 - \frac{|\mathbb{G}}{2^{\ell}}\right)^{t} &\leqslant \left(1 - \frac{1}{p(n)}\right)^{|p(n)\cdot n|} \\ &\leqslant \left(\left(1 - \frac{1}{p(n)}\right)^{p(n)}\right)^{n} \\ &\leqslant \left(e^{-1}\right)^{n} = e^{-n}\end{aligned}$$

对第三个不等式使用命题 A.2。因此可以看到当第二个条件满足的时候，有可能获得一个算法，使得$t = \text{poly}(n)$，并且输出失败的概率可忽略。

注意任何类型的群未必就能满足这两个条件。作为替代，对每个特定的类型，这两个条件必须可验证。

$\mathbb{Z}_N$ 的情况。考虑形式为$\mathbb{Z}_N$的群，$n = \|N\|$。很容易验证上面概括的两个条件。检查一个n比特串x（解释为长度最多为n的一个正整数）是否是$\mathbb{Z}_N$的一个元素，只需要简单地检查是否$x < N$，这明显能够在poly (n)时间内完成。并且，一个随机的n位比特串在$\mathbb{Z}_N$的概率为

$$\frac{N}{2^n} \geqslant \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

具体而言，这里给出得到的算法：

算法 B.15
选择一个$\mathbb{Z}_N$中随机元素
输入：模N，其中N长度为n
输出：$\mathbb{Z}_N$中的一个随机元素
for $i = 1$ to $2n$;
　　$x \leftarrow \{0,1\}^n$
　　if $x < N$ return x
return “fail”

这个算法在poly (n)时间内运行，并且输出“fail”的概率可忽略。

$\mathbb{Z}_N^*$的情况。考虑形式为$\mathbb{Z}_N^*$的群，同样$n = \|N\|$。将“如何判断一个n比特串x是否是$\mathbb{Z}_N^*$中的一个元素”留作练习。为了证明第二个条件，需要证明$\frac{\phi(N)}{2^n} \geqslant 1/\text{poly}(n)$。因为

$$\frac{\phi(N)}{2^n} = \frac{N}{2^n} \cdot \frac{\phi(N)}{N}$$

并且已经看到$\frac{N}{2^n} \geqslant \frac{1}{2}$，需要的界来自下面的定理。

定理 B.16 对于长度为n的N，$N \geqslant 3$，有$\frac{\phi(N)}{N} \geqslant 1/2n$。（有更大的下界，但是上面的已足够满足目的。）这里不证明定理，从两种情况考察$\phi(N)/N$的下界：N是素数，以及N是两个长度接近的不同素数的乘积。

（1）当N是奇素数的时候，分析过程是容易的。这里$\phi(N) = N - 1$，当从$\mathbb{Z}_N$中选择一个随机元素的算法时，

$$\frac{\phi(N)}{2^n} = \frac{N-1}{2^n} \geqslant \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

（2）当$N = pq$的情况时，其中p和q是个两个不同的素数，每一个长度约为$n/2$。

命题 B.17 $N = pq$，其中p和q是个两个不同的素数，每一个长度至少为$n/2$。则

$$\frac{\phi(N)}{N} = 1 - \mathsf{negl}(n)$$

证明 有

$$\frac{\phi(N)}{N} = \frac{(p-1)(q-1)}{pq} = 1 - \frac{1}{q} - \frac{1}{p} + \frac{1}{pq} > 1 - \frac{1}{q} - \frac{1}{p} \geqslant 1 - 2 \cdot 2^{-\left(\frac{n}{2}-1\right)}$$

得证。

因此，当N是素数的时候或者是两个不同的大素数的乘积的时候，存在着一个算法，可以产生$\mathbb{Z}_N^*$中的一个随机元素并且在关于$n = \|N\|$的多项式时间内运行，输出“fail”的概率可忽略。

在本书中，用“$x \leftarrow \mathbb{Z}_N$”或者“$x \leftarrow \mathbb{Z}_N^*$”表示从$\mathbb{Z}_N$或者$\mathbb{Z}_N^*$中随机选择的一个元素$x$，使用本节中的某一个算法。通常假设$x$在希望的范围内，以及选择$x$的算法可能输出“fail”的概率是很小的。

B.3 *寻找一个循环群的生成元

本节将关注在阶为q的循环群$\mathbb{G}$中找一个生成元。这里，q不一定非要是素数，实际上，当q是素数时，寻找一个生成元是平凡的，见推论 7.52。

寻找生成元的方法类似于 B.2.4 中找一个随机的生成元。即将重复对$\mathbb{G}$中的元素进行随机采样，直到找到的元素是生成元为止。正如 B.2.4，分析该方法需要理解两件事情：

(1) 如何有效地测试一个指定的元素是不是生成元；

(2) 生成元占的比例。

为了解释这些问题，首先介绍一点群论基础知识。

B.3.1 群论基础知识

一个元素h的阶是使得$h^i = 1$的最小正整数i。g是一个群$\mathbb{G}$的生成元，阶为$q > 1$。注意到这意味着g的阶是q。考虑到任何元素$h \in \mathbb{G}$，h不是单位元（单位元不可能是$\mathbb{G}$的生成元），问：这个元素是否可能是$\mathbb{G}$的一个生成元。因为g生成了$\mathbb{G}$，故$h = g^x$，$x \in \{1, \cdots, q-1\}$（注意$x \neq 0$，因为h不是单位元）。考虑以下两种情况：

情况 1。$\gcd(x,q)=r>1$。$x=\alpha\cdot r$并且$q=\beta\cdot r$，α和β是非零的小于q的整数。则

$$h^{\beta}=(g^{x})^{\beta}=g^{\alpha r\beta}=(g^{q})^{\alpha}=1$$

所以h的阶最多为$\beta<q$，h不可能是$\mathbb{G}$的生成元。

情况 2。$\gcd(x,q)=1$。h的阶$i\leqslant q$。则

$$g^{0}=1=h^{i}=(g^{x})^{i}=g^{xi}$$

命题 7.50 暗示了$xi=0 \mod q$。这就意味着$q|xi$。因为$\gcd(x,q)=1$，但是，命题 7.3 说明了$q|i$，于是$i=q$。所以h是$\mathbb{G}$的生成元。

综上所述，当$x\in\{1,\cdots,q-1\}$，仅当$\gcd(x,q)=1$的时候，$h=g^{x}$才是$\mathbb{G}$的生成元。前面已有集合$\{x\in\{1,\cdots,q-1\}|\gcd(x,q)=1\}$，这恰好就是$\mathbb{Z}_q^{*}$。因此可证明下面的定理：

定理 B.18 $\mathbb{G}$是一个循环群，阶$q>1$，生成元为g。则群$\mathbb{G}$中有$\phi(q)$个生成元，这些就是集合$\{g^{x}|x\in\mathbb{Z}_q^{*}\}$。

特别的是若$\mathbb{G}$是一个群有素数阶q，则它有$\phi(q)=q-1$个生成元，这和推论 7.52 一致。

下一个问题是判断一个指定的元素h是否是$\mathbb{G}$的生成元。当然，一个检查h是否生成$\mathbb{G}$的方法是简单的列举$\{h^{0},h^{1},\cdots,h^{q-1}\}$，看这个列表是否包含了$\mathbb{G}$中的每个元素。这需要的时间与$q$成线性关系（$\|q\|$的指数关系），与目的相悖，因此不能接受。另外一种方法，如果已经知道了一个生成元g，要去计算出离散对数$x=\log_g h$，并且应用先前的定理。但是，通常可能没有这样一个g，而且计算离散对数本身就是个困难问题。

如果知道q的分解因式，有更好的办法。

命题 B.19 $\mathbb{G}$是一个阶为q的群，$q=\prod_{i=1}^{k}p_i^{e_i}$是$q$的素因子分解，其中$\{p_i\}$是不同的素数，并且$e_i\geqslant 1$。令$q_i=q/p_i$。则$h\in\mathbb{G}$是$\mathbb{G}$的一个生成元，当且仅当

$$h^{q_i}\neq 1 \text{ for } i=1,\cdots,k$$

证明 必要性的证明很容易。假设某个i，令$h^{q_i}=1$，则h的阶最多为$q_i<q$，所以h不可能是生成元。

下面证明其充分性，假设h不是一个生成元，其阶$q'<q$。由命题 7.51 可知$q'|q$。这意味着q'可写做$q'=\prod_{i=1}^{k}p_i^{e_i'}$，$e_i'\geqslant 0$，且至少存在一个$j$，使得$e_j'<e_j$。于是$q'$整除$q_j=p_j^{e_j-1}\prod_{i\neq j}p_i^{e_i}$，故（由命题 7.50）$h^{q_j}=h^{[q_j \bmod q']}=h^{0}=1$。

该命题不要求群$\mathbb{G}$是循环群。如果群$\mathbb{G}$不是循环群，则对每个元素$h\in\mathbb{G}$将存在某个i，使得$h^{q_i}=1$，所以没有生成元（如果$\mathbb{G}$不是循环群必然如此）。

B.3.2 有效算法

现在说明如何有效地测试一个给定的元素是不是生成元，同时说明如何在一个任意群里面有效地找到一个生成元。

测试一个元素是否是生成元。从命题 B.19 立刻得到一种有效的算法判定一个指定的元素h是否为生成元。

算法 B.20
测试一个元素是否为生成元
输入：群的阶q，q的素因数$\{p_i\}_{i=1}^{k}$，素$h \in \mathbb{G}$
输出：判定h是否为$\mathbb{G}$的生成元
for $i = 1$ to k;
　　if $h^{q/p_i} = 1$ return “h不是一个生成元”
return “h是一个生成元”

根据命题 B.19，算法的正确性很明显。现在说明该算法在关于$\|q\|$的多项式内结束。因为，在每个循环中，h^{q/p_i}都能够在多项式时间内被计算，于是仅需说明循环的次数k是多项式的。实际情况正是如此，因为一个整数q只能有不多于$\log_2 q = \mathcal{O}(\|q\|)$个素因数。这是因为

$$q = \prod_{i=1}^{k} p_i^{e_i} \geqslant \prod_{i=1}^{k} p_i \geqslant \prod_{i=1}^{k} 2 = 2^k$$

于是$k \leqslant \log_2 q$。

算法 B.20 需要群的阶q的素因数作为输入。有趣的是，当不知道群的阶的因数时，目前还没有有效的算法测试一个元素是否是一个任意群的生成元。

群的生成元所占比例。由定理 B.18 可知，阶为q的群的生成元的比例为$\phi(q)/q$。定理 B.16 表明$\phi(q)/q = \Omega(1/\|q\|)$。生成元所占比例足够高，将确保从群中选取多项式个元素来产生一个生成元的概率是不可忽略的。（分析的过程和 B.2.4 节相同。）

$\mathbb{Z}_p^*$的实例。综上所述，若已知群的阶的分解因式，找到群$\mathbb{G}$的一个生成元的有效概率方法是存在的。因此，为密码学应用选择一个群时，很重要的一点是该群要满足这个条件。对于形式为$\mathbb{Z}_p^*$的群，p为素数，一些可能性如下：

(1) 正如 7.3.2 中已经大量讨论的，在$\mathbb{Z}_p^*$的素数阶子群中操作，将能够消除当q不是素数而引起的困难。回忆一下，获得这样一个子群的方法是选择p作为强素数（也就是$p = 2q + 1$，q也是一个素数），于是在模p的二次剩余子群中操作（一个素数阶为q的子群）。在这种情况下，所有的元素（除了单位元外）都是生成元。

(2) 或者，如果p是上面说的那种强素数，则循环群$\mathbb{Z}_p^*$的阶是$2q$并且该群的阶的因数分解是已知的。容易找到该群的一个生成元，尽管该群没有素数阶。

(3) 另一种可能性是生成一个随机的素数p，其中$p-1$的因数分解是已知的。这个方法是可能的，但具体的细节已超出本书的范围。

参考文献和扩展阅读材料介绍

强烈推荐那些想更深入探索本章论题的细节的读者，阅读 Shoup 的文献[131]，特别是$\phi(N)/N$的范围（定理 B.16 的渐近（收敛）版本）能够在该文献的第 5 章中找到。

Kalai 的文献[82]给出了一个简单的方法来生成随机数以及它们的因数分解。

练　习

B.1　证明扩展的欧几里得算法的正确性。

B.2　证明扩展欧几里得算法能够在以输入长度为参数的多项式时间内完成。

提示：首先证明一个类似命题 B.8 的命题。

B.3　提出一个迭代算法来有效地（也就是多项式时间）计算$[a^b \mod N]$。（一个迭代算法不会递归调用自己。）

提示：使用辅助变量x（初始值为a）和t（初始值为 1），保持不变量$t \cdot x^b = a^b \mod N$。该算法在$x = 1$时结束，t为最后的结果。

B.4　说明如何在多项式时间内，判断一个n比特串是否是$\mathbb{Z}_N^*$的元素。

参 考 文 献

[1] C. Adams and S. Lloyd. *Understanding PKI: Concepts. Standards, and Deployment Considerations.* Addison Wesley, 2nd edition, 2002.

[2] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781 - 793,2004.

[3] W. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan. Security amplification by composition: The case of doubly-iterated, ideal ciphers. In *Advances in Cryptology - Crypto' 98*, *volume* 1462 of *Lecture Notes in Computer Science*, pages 390 - 407. Springer, 1998.

[4] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. In *Proc. 44 th Annual Symposium on Foundations of Computer Science*, pages 146 - 157. IEEE, 2003.

[5] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194 - 209,1988.

[6] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Advances in Cryptology - Eurocrypt 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83 - 107. Springer, 2002.

[7] P. Barreto. The hashing function lounge. Online link can be found at http://paginas. terra. com. br/informatica/paulobarreto.

[8] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - Crypto'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1 - 15. Springer, 1996.

[9] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proc. 38 th Annual Symposium on Foundations of Computer Science*, pages 394 - 403. IEEE, 1997.

[10] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - Crypto' 98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26 - 45. Springer, 1998.

[11] M. Bellare, O. Goldreich, and A. Mityagin. The power of verification queries in message authentication and authenticated encryption. A vailable at http://eprint. iacr. org/2004/309.

[12] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362 - 399,2000.

[13] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531 - 545. Springer, 2000.

[14] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *1st ACM Conf. on Computer and Communications Security*, pages 62 - 73. ACM, 1993.

[15] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - Eurocrypt' 94*, volume 950 of *Lecture Notes in Computer Science*, pages 92 - 111. Springer, 1995.

[16] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In *Advances in Cryptology - Eurocrypt' 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399 - 416. Springer, 1996.

[17] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology - Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pa-

ges 409 – 426. Springer, 2006. A full version of the paper is available at http://eprint. iacr. org.

[18] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4 (1):3 – 72,1991.

[19] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard.* Springer, 1993.

[20] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology – Crypto'98* , volume 1462 of *Lecture Notes in Computer Science*, pages 1 – 12. Springer, 1998.

[21] M. Blum. Coin flipping by telephone. In *Proc. IEEE COMPCOM*, pages 133 – 137,1982.

[22] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Advances in Cryptology – Crypto'84* , volume 196 of *Lecture Notes in Computer Science*, pages 289 – 302. Springer, 1985.

[23] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850 – 864, 1984.

[24] D. Boneh. The decision Diffie-Hellman problem. In *Algorithmic Number Theory, 3rd Intl. Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48 – 63. Springer, 1998.

[25] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2):203 – 213,1999.

[26] D. Boneh. Simplified OAEP for the RSA and Rabin functions. In *Advances in Cryptology – Crypto 2001* , volume 2139 of *Lecture Notes in Computer Science*, pages 275 – 291. Springer, 2001.

[27] D. Boneh, A. Joux, and P. Nguyen. Why textbook ElGamal and RSA encryption are insecure. In *Advances in Cryptology – Asiacrypt 2000* , volume 1976 of *Lecture Notes in Computer Science*, pages 30 – 44. Springer, 2000.

[28] D. Boneh and R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology – Eurocrypt'98* , volume 1403 of *Lecture Notes in Computer Science*, pages 59 – 71. Springer, 1998.

[29] D. Bressoud. *Factorization and Primality Testing.* Springer, 1989.

[30] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557 – 594,2004.

[31] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology – Crypto' 91* , volume 576 of *Lecture Notes in Computer Science*, pages 470 – 484. Springer, 1992.

[32] L. N. Childs. *A Concrete Introduction to Higher Algebra.* Undergraduate Texts in Mathematics. Springer, 2nd edition, 2000.

[33] C. Cid, S. Murphy, and M. Robshaw. *Algebraic Aspects of the Advanced Encryption Standard.* Springer, 2006.

[34] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38(3):243 – 250, 1994. Available for download from http://researchweb. watson. ibm. com/journal/rd/.

[35] J. – S. Coron. On the exact security of full-domain hash. In *Advances in Cryptology – Crypto 2000* , volume 1880 of *Lecture Notes in Computer Science*, pages 229 – 235. Springer, 2000.

[36] J. – S. Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology – Eurocrypt 2002* , volume 2332 of *Lecture Notes in Computer Science*, pages 272 – 287. Springer, 2002.

[37] J. – S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In *Advances in Cryptology – Crypto 2005* , volume 3621 of *Lecture Notes in Computer Science*, pages 430 – 448. Springer, 2005.

[38] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161 - 185,2000.

[39] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167 - 226,2003.

[40] R. Crandall and C. Pomerance. *Prime Numbers: A Computational Perspective*. Springer, 2nd edition, 2005.

[41] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[42] W. Dai. Crypto ++ 5.2.1 benchmarks. Available for download from http://www.cryptopp.com/benchmarks.html.

[43] I. Damgård. Collision free hash functions and public key signature schemes. In *Advances in Cryptology - Eurocrypt'87*, volume 304 of *Lecture Notes in Computer Science*, pages 1 - 20. Springer, 1988.

[44] I. Damgård. A design principle for hash functions. In *Advances in Cryptology - Crypto'89* , volume 435 of *Lecture Notes in Computer Science*, pages 416 - 427. Springer, 1990.

[45] J. DeLaurentis. A further weakness in the common modulus protocol for the RSA cryptoalgorithm. *Cryptologia*, 8:253 - 259,1984.

[46] M. Dietzfelbinger. *Primality Testing in Polynomial Time*. Springer, 2004.

[47] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644 - 654,1976.

[48] W. Diffie and M. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, pages 74 - 84, June 1977.

[49] J. D. Dixon. Asymptotically fast factorization of integers. *Mathematics of Computation*, 36:255 - 260,1981.

[50] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography, *SIAM Journal on Computing*, 30(2): 391 - 437,2000.

[51] C. Ellison and B. Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1):1 - 7,2000.

[52] S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. In *ASIACRYPT*, volume 739 of *Lecture Notes in Computer Science*, pages 210 - 224. Springer, 1993.

[53] H. Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15 - 23,1973.

[54] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - Crypto' 86* , volume 263 of *Lecture Notes in Computer Science*, pages 186 - 194. Springer, 1987.

[55] M. Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In *Public Key Cryptography - PKC 2003* , volume 2567 of *Lecture Notes in Computer Science*, pages 116 - 129. Springer, 2003.

[56] S. Fluhrer, I. Mantin, and A. Shamir. Attacks on RC4 and WEP. *CryptoBytes*, 5(2):26 - 34,2002.

[57] J. B. Fraleigh. *A First Course in Abstract Algebra*. Addison Wesley, 7th edition, 2002.

[58] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA - OAEP is secure under the RSA assumption. In *Advances in Cryptology - Crypto 2001* , volume 2139 of *Lecture Notes in Computer Science*, pages 260 - 274. Springer, 2001.

[59] T. El Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theory*, 31(4):469 - 472, 1985.

[60] C. F. Gauss. *Disquisitiones Arithmeticae*. Springer, 1986. (English edition).

[61] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology - Eurocrypt' 99* , volume 1592 of *Lecture Notes in Computer Science*, pages 123 - 139. Springer, 1999.

[62] E. -J. Goh, S. Jarecki, J. Katz, and N. Wang. Efficient signature schemes with tight security reductions to the Diffie-Hellman problems. *J. Cryptology*, to appear.

[63] O. Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In *Advances in Cryptology – Crypto'86*, volume 263 of *Lecture Notes in Computer Science*, pages 104 – 110. Springer, 1987.

[64] O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools.* Cambridge University Press, 2001.

[65] O. Goldreich. *Foundations of Cryptography, vol. 2: Basic Applications.* Cambridge University Press, 2004.

[66] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In *Advances in Cryptology – Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 276 – 288. Springer, 1985.

[67] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33 (4): 792 – 807, 1986.

[68] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on Theory of Computing*, pages 25 – 32. ACM, 1989.

[69] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2): 270 – 299, 1984.

[70] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2): 281 – 308, 1988.

[71] S. Goldwasser, S. Micali, and A. C. – C. Yao. Strong signature schemes. In *Proc. 15th Annual ACM Symposium on Theory of Computing*, pages 431 – 439. ACM, 1983.

[72] D. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer, 2004..

[73] J. Håstad. Solving simultaneous modular equations of low degree. *SIAM Journal on Computing*, 17(2): 336 – 341, 1988.

[74] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4): 1364 – 1396, 1999.

[75] J. Håstad and M. Näslund. The security of all RSA and discrete log bits. *Journal of the ACM*, 51(2): 187 – 230, 2004..

[76] I. N. Herstein. *Abstract Algebra.* Wiley, 3rd edition, 1996.

[77] H. Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3): 189 – 221, 2002.

[78] H. M. Heys. *The Design of Substitution-Permutation Network Ciphers Resistant to Cryptanalysis.* PhD thesis, Queen's University, 1994.

[79] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proc. 30th Annual Symposium on Foundations of Computer Science*, pages 230 – 235. IEEE, 1989.

[80] ISO/IEC 9797. Data cryptographic techniques – data integrity mechanism using a cryptographic check function employing a block cipher algorithm, 1989.

[81] D. Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet.* Scribner, 1996.

[82] A. Kalai. Generating random factored numbers, easily. *Journal of Cryptology*, 16(4): 287 – 289, 2003.

[83] J. Katz. *Digital Signatures.* Springer, 2007.

[84] J. Katz and C. – Y. Koo. On constructing universal one-way hash functions from arbitrary one-way functions. *J. Cryptology*, to appear. Available at http://epring. iacr. org/2005/328.

[85] J. Katz and M. Yung. Unforgeable encryption and chosen-ciphertext secure modes of operation. In *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 284 – 299. Springer, 2001.

[86] J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of*

Cryptology, 19(1):67 - 96,2006.

[87] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World.* Prentice Hall, 2nd edition, 2002.

[88] J. Kilian and P. Rogaway. How to protect DES against exhauxtive key search(an analysis of DESX). *Journal of Cryptology*, 14(1):17 - 35,2001.

[89] L. Kohnfelder. Towards a practical public-key cryptosystem, 1978. Undergraduate thesis, MIT.

[90] H. Krawczyk. The order of encryption and authentication for protecting communication(or: How secure is SSL?). In *Advances in Cryptology - Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310 - 331. Springer, 2001.

[91] H. Kugel. America's code breaker. Available for download from: http://militaryhistory. about. com/.

[92] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL - 98, SRI International, 1978.

[93] S. K. Langford. *Differential-Linear Cryptanalysis and Threshold Signatures.* PhD thesis, Stanford University, 1995.

[94] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255 - 293,2001.

[95] S. Levy. *Crypto: How the Code Rebels Beat the Government - Saving Privacy in the Digital Age.* Viking, 2001.

[96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton University Press, 1996.

[97] M. Luby and C. Rackoff. How to construct pseudorandom permutationsfrom pseudorandom functions. *SIAM Journal on Computing*, 17(2):373 - 386,1988.

[98] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - Eurocrypt' 93*, volume 765 of *Lecture Notes in Computer Science*. Springer, 1994.

[99] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRS Press, 1997.

[100] R. Merkle. A digital signature scheme based on a conventional encryption function. In *Advances in Cryptology - Crypto' 87*, volume 293 of *Lecture Notes in Computer Science*, pages 369 - 378. Springer, 1988.

[101] R. Merkle. A certified digital signature. In *Advances in Cryptology - Crypto' 89*, volume 435 of *Lecture Notes in Computer Science*, pages 218 - 238. Springer, 1990.

[102] R. Merkle. One way hash functions and DES. In *Advances in Cryptology - Crypto' 89*, volume 435 of *Lecture Notes in Computer Science*, pages 428 - 446. Springer, 1990.

[103] R. Merkle and M. Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7): 465 - 467,1981.

[104] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. Computing*, 17(2):412 - 426,1988.

[105] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13 (3):300 - 317, 1976..

[106] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21 st Annual ACM Symposium on Theory of Computing*, pages 33 - 43. ACM, 1989.

[107] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 427 - 437. ACM, 1990.

[108] National Bureau of Standards. DES modes of operation, 1980. Federal Information Processing Standard (FIPS), publication 81.

[109] National Bureau of Standards. Data encryption standard(DES), 1977. Federal Information Processing

Standard(FIPS), publication 46.

[110] National Institute of Standards and Technology. The keyed-hash message authentication code (HMAC), 2002. Federal Information Processing Standard(FIPS), publication 198.

[111] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55 (2):165 - 172,1994.

[112] A. M. Odlyzko. Discrete logarithms: The past and the future. *Designs, Codes, and Cryptography*, 19(2/3):129 - 145,2000.

[113] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt' 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223 - 238. Springer,1999.

[114] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Information Theory*, 24(1):106 - 110,1978.

[115] J. M. Pollard. Theorems of factorization and primality testing. *Proc. Cambridge Philosophical Society*, 76: 521 - 528, 1974.

[116] J. M. Pollard. A Monte Carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331 - 334, 1975.

[117] C. Pomerance. The quadratic sieve factoring algorithm. In *Advances in Cryptology - Eurocrypt' 84*, volume 209 of *Lecture Notes in Computer Science*, pages 169 - 182. Springer, 1985.

[118] M. O. Rabin. Digitalized signatures. In R. A. Demillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Security Computation*, pages 155 - 168, Academic Press, 1978.

[119] M. O. Rabin. Digitalized signatures as intractable as factorization. Technical Report TR - 212, MIT/LCS, 1979.

[120] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*,12(1):128 - 138, 1980.

[121] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - Crypto' 91*, volume 576 of *Lecture Notes in Computer Science*, pages 433 - 444. Springer, 1992.

[122] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120 - 126, 1978.

[123] P. Rogaway and T. Shrimpton. Cryptographic hash function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption - FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371 - 388. Springer, 2004.

[124] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 387 - 394. ACM, 1990.

[125] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, 2nd edition, 1995.

[126] D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symposia in Pure Mathematics 20*, pages 415 - 440, 1971.

[127] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656 - 715,1949.

[128] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - Eurocrypt' 97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256 - 266. Springer, 1997.

[129] V. Shoup. Why chosen ciphertext security matters. Technical Report RZ 3076, IBM Zurich, November 1998. Available at http://shoup.net/papers/expo.pdf.

[130] V. Shoup. OAEP reconsidered. In *Advances in Cryptology - Crypto, 2001*, volume 2139 of *Lecture Notes in*

Computer Science, pages 239 – 259. Springer, 2001.

[131] V. Shoup. A *Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005. Also available at http://www. shoup. net/ntb.

[132] J. H. Silverman and J. Tate. *Rational Points on Elliptic Curves*. Undergraduate Texts in Mathematics. Springer, 1994.

[133] G. Simmons. A 'weak' privacy protocol using the RSA crypto algorithm. *Cryptologia*, 7:180 – 182, 1983.

[134] G. Simmons. A survey of information authentication. In G. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, pages 379 – 419. IEEE Press, 1992.

[135] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM. Journal on Computing*, 6(1): 84 – 85, 1977.

[136] D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes, and Cryptography*, 4(4): 369 – 380, 1994.

[137] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 1st edition, 1995.

[138] D. R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 3rd edition, 2005.

[139] W. Trappe and L. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, 2nd edition, 2005.

[140] A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230 – 265, 1936.

[141] G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of the American Institute for Electrical Engineers*, 55:109 – 115, 1926.

[142] S. S. Wagstaff, Jr. *Cryptanalysis of Number Theoretic Ciphers*. Chapman & Hall/CRC Press, 2003.

[143] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA – 1. In *Advances in Cryptology – Crypto 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17 – 36. Springer, 2005.

[144] X. Wang and H. Yu. How to break MD5 and other hash functions. In *Advances in Cryptology – Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19 – 35. Springer, 2005.

[145] L. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC Press, 2003.

[146] P. Weadon. The battle of Midway: AF is short of water, 2000. NSA Historical Publications. Available at: http://www. nsa. gov under Historical Publications.

[147] H. Wu. The misuse of RC4 in Microsoft Word and Excel. Available at http://eprint. iacr. org/2005/007.

[148] ANSI X9. 9. American national standard for financial institution message authentication (wholesale), 1981.

[149] A. C. – C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd Annual Symposium on Foundations of Computer Science*, pages 80 – 91. IEEE, 1982.

内 容 简 介

密码学在确保数据的私密性和完整性，以及计算机网络的安全性方面扮演了关键角色。《现代密码学：原理与协议》一书提供了对现代密码学严格而又容易理解的介绍，关注形式化的定义、精确的假设以及严格的证明。

本书介绍了现代密码学的核心原则，包括现代的、基于计算复杂性的安全，以克服完美安全的局限性。对对称密钥加密和消息鉴别也做了较大篇幅的介绍。同时，举例说明了分组密码，如 DES 和 AES 的设计原则，并且在低级原语层面展示了分组密码的可证明安全构造方法。该书下半部分介绍公钥密码学。首先对数论知识进行了介绍，这些知识是理解 RSA、Diffie - Hellman、El Gamal 以及其他密码学系统所需要的。在探讨了公钥加密和数字签名之后，介绍了随机预言模型及其应用。

无论作为教材、参考书或者自学用书，本书呈献给读者透彻理解现代密码学这一迷人主题所需要的工具。