国外计算机科学教材系列

密码学原理与实践

（第二版）

Cryptography
Theory and Practice
(Second Edition)

Douglas R. Stinson 著

冯登国 译

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是密码学领域的经典著作，被世界上的多所大学用做指定教科书。本书在第一版的基础上进行了细致和严谨的修改，将重点集中放在密码学研究的核心问题上，从应用离散数学的角度对密码学进行了系统阐述。全书共分 7 章，从古典密码学开始，继而介绍了 Shannon 的信息论在密码学中的应用，然后进入现代密码学部分，先后介绍了加密技术、数据加密标准(DES)、高级加密标准(ADES)、公钥密码学、单向 Hash 函数、数字签名等。在内容的选择上，全书既突出了广泛性，又注重对要点的深入探讨。书中每一章后都附有大量的练习题，这既利于读者对书中内容的总结和应用，又是对兴趣、思维和智力的挑战。

本书适合于作为计算机科学、数学等相关学科的密码学课程的教材或教学参考书，同时也是密码学研究的必备参考书。

图书在版编目(CIP)数据

密码学原理与实践:第二版/(加)斯廷森(Stinson, D. R.)著;冯登国译. —北京:电子工业出版社,2003.2
(国外计算机科学教材系列)
书名原文:Cryptography: Theory and Practice, Second Edition
ISBN 7-5053-8465-1

Ⅰ.密…　Ⅱ.①斯…②冯…　Ⅲ.密码—理论—教材　Ⅳ.TN918.1

中国版本图书馆 CIP 数据核字(2003)第 004999 号

责任编辑:张　毅　zhangyi@phei.com.cn
印 刷 者:北京市增富印刷有限责任公司
出版发行:电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编:100036
经　　销:各地新华书店
开　　本:787×1092　1/16　　印张:18.75　字数:480 千字
版　　次:2003 年 2 月第 1 版　2003 年 2 月第 1 次印刷
定　　价:34.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。
联系电话:(010)68279077

出 版 说 明

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

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

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

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

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

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

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

电子工业出版社

教材出版委员会

译 者 序

密码学的研究与应用已有几千年的历史,但作为一门科学是20世纪50年代才开始的。不可否认,互联网的广泛应用大大推动了密码学的研究与发展。大多数国家和地区都已经成立了密码学学会,这些学会定期举办学术会议进行学术交流,促进了密码学的研究与应用。

国内外已出版了大量有关密码学的书籍,其理论研究也相对比较成熟,在很多观点上已达成共识。Douglas R. Stinson所著的《密码学原理与实践(第二版)》一书是一本很有特色的教科书,具体表现在以下几个方面:

1. 表述清晰。书中的语言描述浅显易懂,如分组密码的差分分析和线性分析本是很难描述的问题,本书中却表述得非常清楚。

2. 论证严谨。书中对很多密码问题,如惟一解距离、Hash函数的延拓准则等进行了严格的数学证明,具有一种逻辑上的美感。

3. 内容新颖。书中从可证明安全的角度对很多密码学问题特别是公钥密码问题进行了清楚的论述,使用了预言(Oracle)这一术语,通过阅读本书可使读者能够精确掌握这一概念的灵魂。书中顺便对一些密码学领域的最新进展也做了相应的介绍。

4. 选材精良。书中选择了大量典型的、相对成熟的素材,特别适合于教学使用。

5. 习题丰富。通过演练每章后面精心安排的习题,可以迅速并熟练掌握密码学的基本技巧。

要掌握好一门课程,必须选择一两本好书。我认为本书是非常值得精读的一本好书,这也是我花费大量时间翻译本书的初衷。

本书在翻译过程中,得到了一大批博士的协助,他们是徐涛博士、薛锐博士、孙中伟博士、黄寄洪博士、张斌博士、王鹏博士和杨海波博士,特别是徐涛博士协助完成了全书的统稿和审校工作,没有他们的鼎力相助,本书的翻译和审稿工作绝不会如此顺利,在此对他们表示衷心的感谢。

本书的翻译工作得到了国家973项目(编号:G1999035802)和国家杰出青年科学基金(编号:60025205)的支持,在此表示感谢。

冯登国

2003年元旦于北京

序　言

本书的第一版于1995年3月出版。那时，我的目标是写一本通用的教材，包含密码学方面所有的基本核心领域，并选择一些前沿的主题。在写作过程中，我试图使内容具有足够的弹性并容纳这一学科的众多方向，以便它可在数学、计算机科学和工程等专业的本科生和研究生的密码学课程中使用。

下面列举了本书第一版中的一些特色，并在本书中保留了下来。

- 数学背景知识在需要的时候“及时(just-in-time)”地提供出来。
- 密码体制的描述由更精确的伪代码给出。
- 提供例子以说明密码体制的工作过程。
- 对算法和密码体制的数学基础做了仔细、严格的解释。
- 包含了大量的练习，其中一些很具有挑战性。

本书的第一版包含13章。当两年前开始修订时，我发现有丰富的新材料可以加入到第二版中。为了防止篇幅过于庞大，并能在合理的时间内完成，我决定把第二版的内容更紧密地集中在密码学的核心领域上，这就更像是一门课程里所覆盖的内容。其结果就使得本书包含了对第一版的前7章内容的更新、修改、扩展和重新组织。我计划不久再写一本“姊妹篇”，包含对第一版其余内容的更新，并加入新的内容。

下面对本书的7章内容做一概述：

- 第1章对于简单的“经典”密码体制进行了基本的介绍。一些主题被更新或改进，例如，给出了基于Dan Velleman建议的对于维吉尼亚密码的简化的密码分析。
- 第2章覆盖了密码学中Shannon理论的主要内容，包括完善保密性的概念以及信息论在密码学中的应用。这些内容并没有显著的改动；然而，相对于第一版，本书包含了对于基础概率论的更细致的阐述。
- 第3章是几乎完全重写的。在第一版对应的章节中基本上只讲述了数据加密标准(DES)，现在看来已很陈旧。我决定使用代换-置换网络(substitution-permutation networks)作为数学模型来引入现代分组密码设计和分析的许多概念，包括差分分析和线性分析。这里比以前更加强调一般的原则，并讨论了特定的密码体制(DES和新的AES)以说明这些一般原则。
- 第4章对第一版的第7章做了较大改动。这一章现在介绍了对于带密钥的Hash函数和不带密钥的Hash函数的统一处理，以及它们在构造消息认证码中的应用。这里强调了数学分析和安全性证明。本章还包含了安全Hash算法(SHA)的描述。
- 第5章讨论了RSA密码体制，并给出了相当多的数论背景知识，例如素性检验(primality testing)和因子分解。本章已经被扩展了，包含了几个新的小节，例如Pollard ρ算法、Wiener的低解密指数攻击和基于RSA的密码体制的语义安全。
- 第6章讨论了基于离散对数(Discrete Logarithm)问题的公钥密码体制，例如ElGamal密码体制。这一章也包含了很多新内容，例如Pollard ρ算法、通用算法复杂性更低的下

界、椭圆曲线的扩展讨论、离散对数密码体制的语义安全性，以及 Diffie-Hellman 问题。本章中不再对背包(knapsack)密码体制和 McEliece 密码体制进行讨论。

- 第 7 章讨论了签名方案问题。像以前一样，介绍了 DSA 方案，也包含了对特殊类型的签名方案的阐述，例如不可否认签名和 fail-stop 签名。新材料重点讨论了安全性定义、ElGamal 签名方案的变种(例如 Schnorr 签名方案和椭圆曲线 DSA 方案)，和可证明的安全签名方案，例如全域 Hash。

写一本密码学方面的著作的一个最大的困难就是确定应包含多少数学背景知识。密码学是一个涉及广泛的学科，它需要多个数学领域的知识，包括数论、群论、环论、域论、线性代数、概率论以及信息论。同样地，熟悉计算复杂性、算法和 NP 完全性理论也是很有用的。在我看来，正是需要广泛的数学背景知识导致了学生们在开始学习密码学时感到很困难。

我试图假定不需要太多的数学背景，因此在大多数情况下，只有需要时才引入相应的数学工具。当然，如果读者熟悉基本线性代数和模算术是会很有帮助的。另一方面，对于更专门的主题，例如信息论中的熵概念，仅给出了白描似的介绍。

我要对那些不同意在书名中使用“原理与实践”的人说声抱歉。我承认本书中理论多于实践。使用这个词组的含义是指，我试图使选择到本书中的材料既能重理论，又能体现在实践中的重要性。因此，我选用了一些不在实践中出现的密码体制，因为它们或者从数学角度来看是很优雅的，或者说明了一个很重要的概念或技巧。但另一方面，我也提供了在实践中使用的最重要的密码体制，包括几个美国密码学标准。

在我写本书的过程中，很多人给予了鼓励，指出了本书草稿中的书写和排版中的错误，对于怎样选取新材料和如何处理不同的主题也提供了有益的建议。我要在此特别感谢 Howard Heys、Alfred Menezes 和 Edlyn Teske。

Douglas R. Stinson

于加拿大安大略省 Waterloo

目　　录

第1章　古典密码学

本章主要对密码学和密码分析学做一简要介绍，并给出一些简单的古典密码体制以及对这些体制的破译方法。同时，本章对本书中要用到的各种数学知识也进行了介绍。

1.1　几个简单的密码体制

密码学的基本目的是使得两个在不安全信道中通信的人，通常称为 Alice 和 Bob，以一种使他们的敌手 Oscar 不能明白和理解通信内容的方式进行通信。这样的不安全信道在实际中是普遍存在的，比如电话线或计算机网络。Alice 发送给 Bob 的信息，通常称为明文(plaintext)，例如英文单词、数据或符号。Alice 使用预先商量好的密钥(key)对明文进行加密，加密过的明文称为密文(ciphertext)，Alice 将密文通过信道发送给 Bob。对于敌手 Oscar 来说，他可以窃听到信道中 Alice 发送的密文，但是却无法知道其所对应的明文；而对于接收者 Bob，由于知道密钥，可以对密文进行解密，从而获得明文。

上述观点可用数学方式描述为定义 1.1。

定义 1.1　一个密码体制是满足以下条件的五元组$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$：

1. $\mathcal{P}$表示所有可能的明文组成的有限集。

2. $\mathcal{C}$表示所有可能的密文组成的有限集。

3. $\mathcal{K}$代表密钥空间，是由所有可能的密钥组成的有限集。

4. 对任意的 $K\in\mathcal{K}$，都存在一个加密法则 $e_K\in\mathcal{E}$和相应的解密法则 $d_K\in\mathcal{D}$。并且对每一 $e_K:\mathcal{P}\to\mathcal{C}$和 $d_K:\mathcal{C}\to\mathcal{P}$，对任意的明文 $x\in\mathcal{P}$，均有 $d_K(e_K(x))=x$。

定义 1.1 中，最关键的是性质 4。它主要保证了如果使用 e_K 对明文 x 进行加密，则可使用相应的 d_K 对密文进行解密，从而得到明文 x。

Alice 和 Bob 使用某一特定的密码体制来执行以上协议。首先，他们随机选择一个密钥 $K\in\mathcal{K}$，这一步必须在安全的环境下进行，不能被敌手 Oscar 知道。例如，两人可在同一地点协商密钥，或者使用安全信道传输密钥。完成密钥协商后，假如 Alice 想通过不安全信道发送消息串给 Bob，不妨设此消息串为

$$x=x_1x_2\cdots x_n$$

n 为正整数，$x_i\in\mathcal{P}$，$i=1,2,\cdots,n$。对每一 x_i，使用加密规则 e_K 对其进行加密，K 是预先协商好的密钥。Alice 计算 $y_i=e_K(x_i)$，$1\leqslant i\leqslant n$。然后将密文串

$$\boldsymbol{y} = y_1 y_2 \cdots y_n$$

通过信道发送给 Bob。当 Bob 接收到密文串 $y_1 y_2 \cdots y_n$ 时,他使用解密规则 d_K 对其进行解密,就可得到明文串 $x_1 x_2 \cdots x_n$。图 1.1 描述出了具体的加解密过程。

显然,用来加密的加密函数 e_K 必须是一个单射函数(例如,一对一映射),否则将给解密工作带来麻烦,例如,如果:

$$y = e_K(x_1) = e_K(x_2)$$

且 $x_1 \neq x_2$,则 Bob 就无法判断 y 究竟该对应于 x_1 还是 x_2。如果$\mathcal{P} = \mathcal{C}$,即明文空间等于密文空间,则具体的加密函数就是一个置换。这就是说,如果明文空间等于密文空间,则每个加密函数仅仅是对明文空间的元素的一个重新排列(置换)。

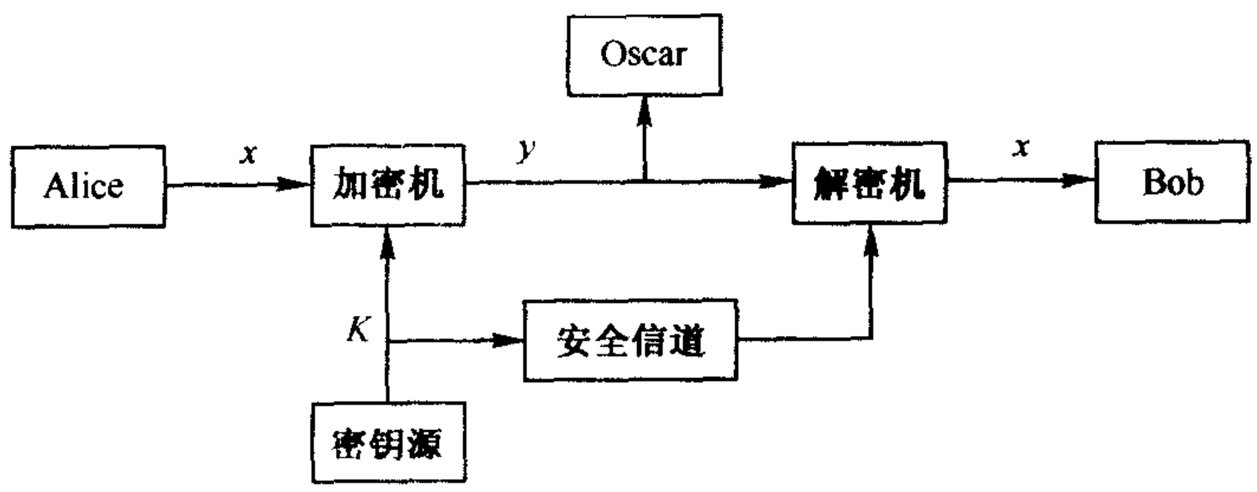

图 1.1 通信信道

1.1.1 移位密码

本小节介绍的移位密码(Shift Cipher),其基础是数论中的模运算。这里首先给出一些模运算的基本定义。

定义 1.2 假设 a 和 b 均为整数,m 是一正整数。若 m 整除 $b - a$,则可将其表示为 $a \equiv b \pmod{m}$。式 $a \equiv b \pmod{m}$读做"a 与 b 模 m 同余",正整数 m 称为模数。

假如用 m 分别去除 a 和 b,可得相应的商和余数,余数是在 0 与 $m-1$ 之间。即可将 a 与 b 分别表示为 $a = q_1 m + r_1$, $b = q_2 m + r_2$, $0 \leqslant r_1 \leqslant m-1$, $0 \leqslant r_2 \leqslant m-1$。这样,可以看出 $a \equiv b \pmod{m}$当且仅当 $r_1 = r_2$。上述的余数 r_1 可用记号 $a \bmod m$ 来表示。因此,$a \equiv b \pmod{m}$当且仅当 $a \bmod m = b \bmod m$。

我们给出两个例子。计算 $101 \bmod 7$, $101 = 7 \times 14 + 3$。因为 $0 \leqslant 3 \leqslant 6$,故 $101 \bmod 7 = 3$。再如计算 $(-101) \bmod 7$,因为 $-101 = 7 \times (-15) + 4$,故 $(-101) \bmod 7 = 4$。

注 许多计算机编程语言定义 $a \bmod m$ 的取值在 $-m+1, \cdots, m-1$ 之间,并要求取值与 a 的正负号相同。在此定义下,$(-101) \bmod 7$ 应为 -3。但在这里,为了方便起见,我们要求 $a \bmod m$ 恒为一非负值。

下面定义模 m 上的算术运算:令 $\mathbb{Z}_m$ 表示集合 $\{0, 1, \cdots, m-1\}$,在其上定义加法和乘法,

其运算类似于普通的实数域上的加法和乘法,所不同的只是所得的值是取模以后的余数。

例如,在$\mathbb{Z}_{16}$上计算 11×13,因为 $11 \times 13 = 143 = 8 \times 16 + 15$,故在$\mathbb{Z}_{16}$上 $11 \times 13 = 15$。

以上定义的$\mathbb{Z}_m$ 上的加法和乘法满足很多我们熟知的运算法则,在此不加证明地列出这些法则:

1. 对加法运算封闭:对任意的 $a, b \in \mathbb{Z}_m$,有 $a + b \in \mathbb{Z}_m$。
2. 加法运算满足交换律:对任意的 $a, b \in \mathbb{Z}_m$,有 $a + b = b + a$。
3. 加法运算满足结合律:对任意的 $a, b, c \in \mathbb{Z}_m$,有$(a + b) + c = a + (b + c)$。
4. 0 是加法单位元:对任意的 $a \in \mathbb{Z}_m$,有 $a + 0 = 0 + a = a$。
5. 任意 $a \in \mathbb{Z}_m$ 的加法逆元为 $m - a$,因为 $a + (m - a) = (m - a) + a = 0$。
6. 对乘法运算封闭:对任意的 $a, b \in \mathbb{Z}_m$,有 $ab \in \mathbb{Z}_m$。
7. 乘法运算满足交换律:对任意的 $a, b \in \mathbb{Z}_m$,有 $ab = ba$。
8. 乘法运算满足结合律:对任意的 $a, b, c = \mathbb{Z}_m$,有$(ab)c = a(bc)$。
9. 1 是乘法单位元:对任意的 $a \in \mathbb{Z}_m$,有 $a \times 1 = 1 \times a = a$。
10. 乘法和加法之间存在分配律:对任意的 $a, b, c \in \mathbb{Z}_m$,有$(a + b)c = (ac) + (bc)$,$a(b + c) = (ab) + (ac)$。

性质 1、3 ~ 5,说明$\mathbb{Z}_m$ 关于其上定义的加法运算构成一个群;若再加上性质 2,则构成一个交换群(阿贝尔群)。

性质 1 ~ 10,说明$\mathbb{Z}_m$ 是一个环。本书后面将碰到许多其他的群和环,常见的环有定义了普通加法和乘法的全体整数$\mathbb{Z}$、全体实数$\mathbb{R}$、全体复数$\mathbb{C}$ 等,这些环均是无限环。但本书中我们关心的环都是有限环。

由于在$\mathbb{Z}_m$ 中存在加法逆,我们可以在$\mathbb{Z}_m$ 中减去一个元素。我们定义$\mathbb{Z}_m$ 中 $a - b$ 为$(a - b) \bmod m$。即,我们计算整数 $a - b$,然后对它进行模 m 约化。例如,为了在$\mathbb{Z}_{31}$中计算 $11 - 18$,我们首先用 11 减去 18,得到 -7,然后计算$(-7) \bmod 31 = 24$。

密码体制 1.1 给出了移位密码。因为英文有 26 个字母,故其一般定义在$\mathbb{Z}_{26}$上。很容易验证移位密码满足前面所定义的密码体制的条件,对任意的 $x \in \mathbb{Z}_{26}$,有 $d_K(e_K(x)) = x$。

密码体制 1.1　移位密码

令$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$。对 $0 \leqslant K \leqslant 25$,任意 $x, y \in \mathbb{Z}_{26}$,定义

$$e_K(x) = (x + K) \bmod 26$$

以及

$$d_K(y) = (y - K) \bmod 26$$

注　若取 $K = 3$,则此密码体制通常叫做凯撒密码(Caesar Cipher),因为它首先为儒勒·凯撒(Julius Caesar)所使用。

使用移位密码可以用来加密普通的英文句子,但是首先必须建立英文字母和模 26 剩余之间的一一对应关系:如 A↔0,B↔1,…,Z↔25。将其列表如下:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

下面给出一个实例。

例 1.1 假设移位密码的密钥为 $K = 11$,明文为:

wewillmeetatmidnight

首先,将明文中的字母对应于其相应的整数,得到如下数字串:

22 4 22 8 11 11 12 4 4 19

0 19 12 8 3 13 8 6 7 19

然后,将每一数都与 11 相加,再对其和取模 26 运算,可得:

7 15 7 19 22 22 23 15 15 4

11 4 23 19 14 24 19 17 18 4

最后,再将其转换为相应的字母串,即得密文:

HPHTWWXPPELEXTOYTRSE

要对密文进行解密,只需执行相应的逆过程即可。Bob 首先将密文转换为数字,再用每个数字减去 11 后取模 26 运算,最后将相应的数字再转换为字母可得明文。

注 以上例子中,我们使用小写字母来表示明文,而使用大写字母来表示密文,为了后面讨论的方便,仍然使用这种规则。

一个实用的加密体制,应该满足某些特性才行,显然以下两点必须满足:

1. 加密函数 e_K 和解密函数 d_K 都应该易于计算。
2. 对任何敌手来说,即使他获得了密文 y,也不可能由此确定出密钥 K 或明文 x。

第二点关于"安全"的要求,在这里有些模糊不清。在已知密文 y 的情形下,试图得到密钥 K 的过程,我们称其为密码分析(后面还有详细介绍)。要注意,如果 Oscar 能获得密钥 K,则他解密密文 y 即可得明文 x。因此,通过密文 y 计算密钥 K,至少要和通过密文 y 计算明文一样困难。

移位密码(模 26)是不安全的,显然可用密钥穷尽搜索方法来破译。因为密钥空间太小,只有 26 个可能的情况,可以穷举所有的可能密钥,得到我们所希望的有意义的明文来。我们给出一个例子。

例 1.2 设有如下密文串:

JBCRCLQRWCRVNBJENBWRWN

依次试验所有可能的解密密钥 $d_0, d_1 \cdots$，可得如下不同字母串：

jbcrclqrwcrvnbjenbwrwn
iabqbkpqvbqumaidmavqvm
hzapajopuaptlzhclzupul
gyzozinotzoskygbkytotk
fxynyhmnsynrjxfajxsnsj
ewxmxglmrxmqiweziwrmri
dvwlwfklqwlphvdyhvqlqh
cuvkvejkpvkogucxgupkpg
btujudijoujnftbwftojof
astitchintimesavesnine

至此，已可以得出有意义的明文来，相应的密钥 $K = 9$（明文串为“a stitch in time saves nine”，中文意思是“小洞不补，大洞吃苦”）。

平均来看，使用上述方法计算明文只需试验 26/2 = 13 次即可。

上面的例子表明，一个密码体制安全的必要条件是能抵抗穷尽密钥搜索攻击，普通的做法是密钥空间必须足够大。但是，很大的密钥空间并不是保证密码体制安全的充分条件。

1.1.2　代换密码

另一个比较有名的古典密码体制是代换密码（Substitution Cipher）。这种密码体制已经使用了数百年。报纸上的数字猜谜游戏就是代换密码的一个典型例子。

密码体制 1.2　代换密码

令 $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$。$\mathcal{K}$ 由 26 个数字 $0, 1, \cdots, 25$ 的所有可能置换组成。对任意的置换 $\pi \in \mathcal{K}$，定义：

$$e_\pi(x) = \pi(x)$$

再定义：

$$d_\pi(y) = \pi^{-1}(y)$$

这里 π^{-1} 代表置换 π 的逆置换。

事实上，在代换密码的情形下，我们也可以认为 $\mathcal{P}$ 和 $\mathcal{C}$ 是 26 个英文字母。在移位密码中使用 $\mathbb{Z}_{26}$ 是因为加密和解密都是代数运算。但是在代换密码的情形下，可更简单地将加密和解密过程直接看做是一个字母表上的置换。

任取一置换 π,便可得到一加密函数,见下表(同前,小写字母表示明文,大写字母表示密文):

a	b	c	d	e	f	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	s	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

按照上表应有 $e_\pi(a)=X, e_\pi(b)=N$,等等。解密函数是相应的逆置换,由下表给出:

A	B	C	D	E	F	G	H	I	J	K	L	M
d	l	r	y	v	o	h	E	z	x	w	p	t

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	g	f	j	q	n	m	U	s	k	a	c	i

因此,$d_\pi(A)=d, d_\pi(B)=l$,等等。

作为一个练习,读者可以使用解密函数来解密下面的密文:

MGZVYZLGHCMHJMYXSSFMNHAHYCDLMHA

代换密码的一个密钥刚好对应于 26 个英文字母的一种置换。所有可能的置换有 26! 种,这个数值超过了 4.0×10^{26},是一个很大的数字。因此,采用穷尽密钥搜索的攻击方法,即使使用计算机,在计算上也是不可行的。但是,后面我们将看到,采用别的密码分析方法,代换密码可以很容易地被攻破。

1.1.3 仿射密码

从前面我们看到,移位密码实际上是代换密码的一种特殊情况,其只包含了 26! 种代换密码置换中的 26 种。另一个代换密码的特殊情形是所谓的仿射密码(Affine Cipher)。在仿射密码中,加密函数定义为:

$$e(x)=(ax+b) \bmod 26$$

$a, b\in\mathbb{Z}_{26}$。因为这样的函数被称为仿射函数,所以也将这样的密码体制称为仿射密码(可以看出,当 $a=1$ 时,其对应的正是移位密码)。

为了能对密文进行解密,必须保证所选用的仿射函数是一个单射函数。换句话说,对任意 $y\in\mathbb{Z}_{26}$,如下同余方程有惟一解 x:

$$ax+b\equiv y \pmod{26}$$

上述同余方程等价于:

$$ax\equiv y-b \pmod{26}$$

当 y 跑遍$\mathbb{Z}_{26}$时,显然 $y-b$ 亦跑遍$\mathbb{Z}_{26}$。故我们只需研究同余方程 $ax\equiv y \pmod{26}$($y\in$

$\mathbb{Z}_{26}$)即可。

以上同余方程有惟一解的充分必要条件是 $\gcd(a,26)=1$(这里 gcd 表示最大公约数)。分析如下。首先,假设 $\gcd(a,26)=d>1$,则同余方程 $ax\equiv 0 \pmod{26}$至少有两个解,分别为 $x=0$ 和 $x=26/d$。在这种情况下,$e(x)=(ax+b) \bmod 26$ 不是一个单射函数,因此不能用来作为一个有效的加密函数。

例如,$\gcd(4,26)=2$,显然 $4x+7$ 不是一个有效的加密函数:因为对任意 $x\in\mathbb{Z}_{26}$,x 和 $x+13$ 将被加密成相同的密文。

若 $\gcd(a,26)=1$,存在 x_1,x_2 使得:

$$ax_1\equiv ax_2 \pmod{26}$$

则有:

$$a(x_1-x_2)\equiv 0 \pmod{26}$$

应有:

$$26 \mid a(x_1-x_2)$$

现在我们利用整除的基本性质:如果 $\gcd(a,b)=1$ 且 $a\mid bc$,那么 $a\mid c$。由于 $26\mid a(x_1-x_2)$,且 $\gcd(a,26)=1$,因此有:

$$26 \mid (x_1-x_2)$$

这就意味着 $x_1\equiv x_2 \pmod{26}$。

由上面的分析可以看出,若 $\gcd(a,26)=1$,则同余方程 $ax\equiv y \pmod{26}$至多只有一个解。因此,如果 x 跑遍$\mathbb{Z}_{26}$,则 ax 也相应地取遍$\mathbb{Z}_{26}$的所有值,不会重复。这说明,$ax\equiv y \pmod{26}$有惟一的解 x。

定理 1.1　设 $a\in\mathbb{Z}_m$,对任意的 $b\in\mathbb{Z}_m$,同余方程 $ax\equiv b \pmod{m}$有惟一解 $x\in\mathbb{Z}_m$ 的充分必要条件是 $\gcd(a,m)=1$。

因为 $26=2\times 13$,故所有的与 26 互素的数为 $a=1,3,5,7,9,11,15,17,19,21,23$ 和 25。b 的取值可为$\mathbb{Z}_{26}$中的任何数。因此仿射密码的密钥空间为 $12\times 26=312$(当然,这个密钥量太小,是很不安全的)。

下面我们考虑模是 m 的一般情形,首先给出数论中的一个概念。

定义 1.3　设 $a\geqslant 1, m\geqslant 2$ 且均为整数。如果 $\gcd(a,m)=1$,则称 a 与 m 互素。$\mathbb{Z}_m$ 中所有与 m 互素的数的个数使用 $\phi(m)$来表示(这个函数称为欧拉函数)。

数论中的一个著名结论用 m 的素数幂分解的形式给出了 $\phi(m)$的值。(一个整数 p 称为

素数,如果它没有除 1 和 p 之外的素因子。任一整数 $m>1$ 可以用惟一的方式分解成素数幂的乘积。例如,$60=2^2\times3\times5$ 和 $98=2\times7^2$。)

如下的定理给出了值的公式。

定理 1.2 假定:

$$m=\prod_{i=1}^{n}p_i^{e_i}$$

这里 p_i 均为素数且互不相同,$e_i>0$,$1\leqslant i\leqslant n$。则:

$$\phi(m)=\prod_{i=1}^{n}(p_i^{e_i}-p_i^{e_i-1})$$

由上面关于欧拉函数 $\phi(m)$的公式,可推出在仿射密码中密钥空间的大小为 $m\phi(m)$。例如,如果 $m=60$,$\phi(60)=2\times2\times4=16$,那么此仿射密码的密钥空间的大小为 $60\times16=960$。

下面分析仿射密码在模为 26 情形下的解密过程。已知有 $\gcd(a,26)=1$,解密的过程就是解同余方程 $y\equiv ax+b\pmod{26}$。由前面的讨论可知,此同余方程在$\mathbb{Z}_{26}$上只有惟一解,但是还需要有效的方法来具体找出这个解。下面给出具体的方法。

在此首先给出乘法逆的概念。

定义 1.4 设 $a\in\mathbb{Z}_m$,若存在 $a'\in\mathbb{Z}_m$,使得 $aa'\equiv a'a\equiv1\pmod{m}$,则 a'称为 a 在$\mathbb{Z}_m$ 上的乘法逆,将其记为 $a^{-1}\bmod m$。在 m 是固定的情形下,一般也可将其简记为 a^{-1}。

同样由前面的讨论可知,当且仅当 $\gcd(a,m)=1$ 时,a 在$\mathbb{Z}_m$ 上存在乘法逆,并且其逆如果存在,则必惟一(模 m)。还可发现,如果 $b=a^{-1}$,那么 $a=b^{-1}$。如果 p 为素数,则$\mathbb{Z}_p$ 上任一非零元均有乘法逆存在,一个环如果满足这条性质,也把它称为域。

在本书的后面章节,将详细给出求乘法逆的具体算法。但是,在$\mathbb{Z}_{26}$的情形下,可以很容易地找出与 26 互素的元的乘法逆来:

$$1^{-1}=1$$
$$3^{-1}=9$$
$$5^{-1}=21$$
$$7^{-1}=15$$
$$11^{-1}=19$$
$$17^{-1}=23$$
$$25^{-1}=25$$

以上式子可以很容易验证,例如,$7\times15=105\equiv1\pmod{26}$,因此 $7^{-1}=15$ 并且 $15^{-1}=7$。

考虑同余方程 $y\equiv ax+b\pmod{26}$。其等价于如下同余方程:

$$ax \equiv y - b \pmod{26}$$

因为 $\gcd(a,26)=1$，故 a 在 $\mathbb{Z}_{26}$ 上存在乘法逆元 a^{-1}，在上式两边同乘以 a^{-1}，有：

$$a^{-1}(ax) \equiv a^{-1}(y-b) \pmod{26}$$

使用乘法结合律，有：

$$a^{-1}(ax) \equiv (a^{-1}a)x \equiv 1x \equiv x \pmod{26}$$

故有，$x = a^{-1}(y-b) \pmod{26}$。因此相应的解密变换为：

$$d(y) = a^{-1}(y-b) \bmod 26$$

最后，我们给出完整的仿射密码体制。

密码体制 1.3　仿射密码

令 $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ 且

$$\mathcal{K} = \{(a,b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} : \gcd(a,26) = 1\}$$

对任意的 $K=(a,b) \in \mathcal{K}$，$x, y \in \mathbb{Z}_{26}$，定义加密变换为：

$$e_K(x) = (ax + b) \bmod 26$$

相应的解密变换为：

$$d_K(y) = a^{-1}(y-b) \bmod 26$$

我们给出一个小例子。

例 1.3　设密钥 $K=(7,3)$。由前所得，有 $7^{-1} \bmod 26 = 15$。加密变换为：

$$e_K(x) = 7x + 3$$

相应的解密变换为：

$$d_K(y) = 15(y-3) = 15y - 19$$

以上运算均是在 $\mathbb{Z}_{26}$ 上完成的。下面来验证对任意的 $x \in \mathbb{Z}_{26}$，都有 $d_K(e_K(x)) = x$：

$$\begin{aligned} d_K(e_K(x)) &= d_K(7x+3) \\ &= 15(7x+3) - 19 \\ &= x + 45 - 19 \\ &= x \end{aligned}$$

使用上面的密钥,我们来加密明文"hot"。首先转换字母"h"、"o"、"t"为对应的模 26 下的数,分别为 7、14、19。将其分别加密如下:

$$(7 \times 7 + 3) \bmod 26 = 52 \bmod 26 = 0$$
$$(7 \times 14 + 3) \bmod 26 = 101 \bmod 26 = 23$$
$$(7 \times 19 + 3) \bmod 26 = 136 \bmod 26 = 6$$

所以三个密文字符为 0、23、6,相应的密文应为"AXG"。具体的解密变换留给读者完成。

1.1.4 维吉尼亚密码

在前面介绍的移位密码和代换密码中,一旦密钥被选定,则每个字母对应的数字都被加密变换成对应的惟一数字。这种密码体制我们一般称为单表代换密码。下面介绍的是有名的维吉尼亚密码(Vigenère Cipher),这是一种多表代换密码。其发明者为 16 世纪的法国人 Blaise de Vigenère。

密码体制 1.4 维吉尼亚密码

设 m 是一个正整数。定义$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$。对任意的密钥 $K = (k_1, k_2, \cdots, k_m)$,定义:

$$e_K(x_1, x_2, \cdots, x_m) = (x_1 + k_1, x_2 + k_2, \cdots, x_m + k_m)$$

和

$$d_K(y_1, y_2, \cdots, y_m) = (y_1 - k_1, y_2 - k_2, \cdots, y_m - k_m)$$

以上所有的运算都是在$\mathbb{Z}_{26}$上进行。

使用前面所述的方法,对应 A↔0, B↔1, ⋯, Z↔25,则每个密钥 K 相当于一个长度为 m 的字母串,称为密钥字。维吉尼亚密码一次加密 m 个明文字母,下面给出一个小例子。

例 1.4 假设 $m = 6$,密钥字为"CIPHER",其对应于如下的数字串 $K = (2, 8, 15, 7, 4, 17)$。要加密的明文为:

thiscryptosystemisnotsecure

将明文串转化为对应的数字,每 6 个为一组,使用密钥字进行模 26 下的加密运算,如下所示:

19	7	8	18	2	17	24	15	19	14	18	24	18	19
2	8	15	7	4	17	2	8	15	7	4	17	2	8
21	15	23	25	6	8	0	23	8	21	22	15	20	1

4	12	8	18	13	14	19	18	4	2	20	17	4
15	7	4	17	2	8	15	7	4	17	2	8	15
19	19	12	9	15	22	8	25	8	19	22	25	19

则相应的密文应该为：

VPXZGIAXIVWPUBTTMJPWIZITWZT

解密时，使用相同的密钥字，进行逆运算即可，这里不再给出。

可看出，维吉尼亚密码的密钥空间大小为 26^m，所以即使 m 的值很小，使用穷尽密钥搜索方法也需要很长的时间。例如，当 $m=5$ 时，密钥空间大小超过 1.1×10^7，这样的密钥量已经超出了使用手算进行穷尽搜索的能力范围(当然使用计算机另当别论)。

在一个具有密钥字长度为 m 的维吉尼亚密码中，一个字母可以被映射为 m 个字母中的某一个(假定密钥字包含 m 个不同的字母)。这样的一个密码体制称为多表代换密码体制。一般来说，多表代换密码比单表代换密码更为安全一些。

1.1.5　希尔密码

本小节介绍另外一种多表代换密码——希尔密码(Hill Cipher)。这种密码体制是 Lester S. Hill 在 1929 年提出的。设 m 是一正整数，定义 $\mathcal{P}=\mathcal{C}=(\mathbb{Z}_{26})^m$。希尔密码的主要思想是利用了我们熟知的线性变换的方法，不同的是这种变换是在 $\mathbb{Z}_{26}$ 上进行的。

例如，设 $m=2$，每一个明文单元使用 $x=(x_1,x_2)$ 来表示，同样密文单元使用 $y=(y_1,y_2)$ 来表示。具体加密中，y_1,y_2 将被表示为 x_1,x_2 的线性组合。例如：

$$y_1=(11x_1+3x_2)\bmod 26$$
$$y_2=(8x_1+7x_2)\bmod 26$$

使用矩阵，可将上式简写为：

$$(y_1,y_2)=(x_1,x_2)\begin{pmatrix}11 & 8\\ 3 & 7\end{pmatrix}$$

以上的运算都是在 $\mathbb{Z}_{26}$ 上进行的。密钥 $\boldsymbol{K}$ 一般取为一个 $m\times m$ 的矩阵，记为 $\boldsymbol{K}=(k_{i,j})$。对明文 $x=(x_1,x_2,\cdots,x_m)\in\mathcal{P}$ 以及 $\boldsymbol{K}\in\mathcal{K}$，按照如下方法来计算 $y=e_K(x)=(y_1,\cdots,y_m)$：

$$(y_1,y_2,\cdots,y_m)=(x_1,x_2,\cdots,x_m)\begin{pmatrix}k_{1,1} & k_{1,2} & \cdots & k_{1,m}\\ k_{2,1} & k_{2,2} & \cdots & k_{2,m}\\ \vdots & \vdots & \ddots & \vdots\\ k_{m,1} & k_{m,2} & \cdots & k_{m,m}\end{pmatrix}$$

或者也可以使用矩阵形式，直接表示为 $y=x\boldsymbol{K}$。

从上面的加密变换可以看出,密文是通过对明文进行线性变换得出的。下面来考虑解密过程,也就是如何从 y 算出 x。熟悉线性代数的读者可能很容易地想到使用 $\boldsymbol{K}$ 的逆矩阵 $\boldsymbol{K}^{-1}$ 来进行解密变换。相应的明文应该为 $x = y\boldsymbol{K}^{-1}$。

首先给出一些基本的线性代数的知识。设 $\boldsymbol{A} = (a_{i,j})$ 为一个 $l \times m$ 矩阵,$\boldsymbol{B} = (b_{j,k})$ 为一个 $m \times n$ 矩阵,则我们定义矩阵乘法 $\boldsymbol{AB} = (c_{i,k})(1 \leqslant i \leqslant l, 1 \leqslant k \leqslant n)$ 为如下形式:

$$c_{i,k} = \sum_{j=1}^{m} a_{i,j} b_{j,k}$$

矩阵乘法满足结合律,即有 $(\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC})$,但是,一般不满足交换律,这就是说,在一般情形下,$\boldsymbol{AB} = \boldsymbol{BA}$ 并不成立。

$m \times m$ 矩阵中,有一个特殊矩阵,其主对角线的值为 1,其余均为 0,我们将其称为单位矩阵,记为 $\boldsymbol{I}_m$。如 2×2 单位矩阵为如下形式:

$$\boldsymbol{I}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

单位矩阵 $\boldsymbol{I}_m$ 有如下性质:对任意的 $l \times m$ 矩阵 $\boldsymbol{A}$,有 $\boldsymbol{AI}_m = \boldsymbol{A}$;对任意的 $m \times n$ 矩阵 $\boldsymbol{B}$,有 $\boldsymbol{I}_m\boldsymbol{B} = \boldsymbol{B}$。这也正是将其称为单位矩阵的原因。有了单位矩阵 $\boldsymbol{I}_m$,$m \times m$ 矩阵 $\boldsymbol{A}$ 的逆矩阵 $\boldsymbol{A}^{-1}$(如果存在)应满足 $\boldsymbol{AA}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_m$,并且如果 $\boldsymbol{A}^{-1}$ 存在,其一定是惟一的。

有了上面的事实,我们很容易得到相应的解密函数。首先有 $y = x\boldsymbol{K}$,两边同乘以 $\boldsymbol{K}^{-1}$,有:

$$y\boldsymbol{K}^{-1} = (x\boldsymbol{K})\boldsymbol{K}^{-1} = x(\boldsymbol{K}\boldsymbol{K}^{-1}) = x\boldsymbol{I}_m = x$$

对前面给出的密钥矩阵 $\boldsymbol{K}$,容易得出其在 $\mathbb{Z}_{26}$ 上的逆矩阵,如下所示:

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

这是因为

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = \begin{pmatrix} 11 \times 7 + 8 \times 23 & 11 \times 18 + 8 \times 11 \\ 3 \times 7 + 7 \times 23 & 3 \times 18 + 7 \times 11 \end{pmatrix}$$

$$= \begin{pmatrix} 261 & 286 \\ 182 & 131 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(注意,其中的算术运算都是模 26 的。)

下面给出一个具体的应用实例。

例 1.5 假设密钥为:

$$\boldsymbol{K} = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$$

由前所述，其逆矩阵为：

$$\boldsymbol{K}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

设要加密的明文为“july”，则可将明文化为如下的两个加密单元：(9,20)(对应于“ju”)和(11,24)(对应于“ly”)。分别对其进行加密变换如下：

$$(9,20)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (99+60, 72+140) = (3,4)$$

$$(11,24)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = (121+72, 88+168) = (11,22)$$

因此，密文为 DELW。要解密密文，Bob 做如下的计算：

$$(3,4)\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (9,20)$$

$$(11,22)\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = (11,24)$$

这样即可得所需的明文。

由前面的分析可以看出，如果密钥 $\boldsymbol{K}$ 可逆，则加解密可轻松完成。事实上，$\boldsymbol{K}$ 可逆是完成解密的必要条件(此结论来自于基本的线性代数的知识，在此我们不给出证明)。因此，我们关心的主要问题就是矩阵 $\boldsymbol{K}$ 是否可逆。

定义 1.5　设 $\boldsymbol{A} = (a_{i,j})$ 是一个 $m \times m$ 矩阵。对 $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant m$，定义 $\boldsymbol{A}_{i,j}$ 为从矩阵 $\boldsymbol{A}$ 中删除第 i 行第 j 列后的新矩阵。$\boldsymbol{A}$ 的行列式的值，一般记为 $\det \boldsymbol{A}$，可如下递归计算：当 $m = 1$ 时，$\det \boldsymbol{A} = a_{1,1}$；当 $m > 1$ 时，

$$\det \boldsymbol{A} = \sum_{j=1}^{m} (-1)^{i+j} a_{i,j} \det \boldsymbol{A}_{i,j}$$

上式中 i 是任意的一个介于 1 与 m 之间的固定整数。

上面计算 $\det \boldsymbol{A}$ 的值与选取哪一个具体的 i 值无关，要看出这一点不是很容易，但是我们可以证明此结论是正确的。为了后面使用方便，这里给出 2×2 和 3×3 矩阵的行列式值的计算方法：

如果 $\boldsymbol{A} = (a_{i,j})$ 是一个 2×2 矩阵，则：

$$\det \boldsymbol{A} = a_{1,1} a_{2,2} - a_{1,2} a_{2,1}$$

如果 $\boldsymbol{A} = (a_{i,j})$ 是一个 3×3 矩阵，则：

$$\det \boldsymbol{A} = a_{1,1}a_{2,2}a_{3,3} + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{2,1}a_{3,2} - (a_{1,1}a_{2,3}a_{3,2} + a_{1,2}a_{2,1}a_{3,3} + a_{1,3}a_{2,2}a_{3,1})$$

显然,对于较大的数 m,使用上面介绍的递归的方法来计算行列式的值比较麻烦。一般计算行列式的值,我们可以使用"初等行变换"这一手段来进行,这方面可参看相应的线性代数教科书。

关于行列式的值有两个重要的结论这里需要给出:首先,$\det \boldsymbol{I}_m = 1$;其次,$\det(\boldsymbol{AB}) = \det \boldsymbol{A} \times \det \boldsymbol{B}$。

一个实矩阵 $\boldsymbol{K}$ 可逆当且仅当其所对应的行列式的值非零。但是,这里需要强调指出的是,解密工作需要的运算都是在 $\mathbb{Z}_{26}$ 上进行。此时的结论变为:矩阵 $\boldsymbol{K}$ 在模 26 情形下存在可逆矩阵的充分必要条件是 $\gcd(\det \boldsymbol{K}, 26) = 1$。先证必要性,假设 $\boldsymbol{K}$ 可逆,其逆矩阵记为 $\boldsymbol{K}^{-1}$,则

$$1 = \det \boldsymbol{I} = \det(\boldsymbol{K}\boldsymbol{K}^{-1}) = \det \boldsymbol{K} \det \boldsymbol{K}^{-1}$$

显然,有 $\gcd(\det \boldsymbol{K}, 26) = 1$。

充分性的证明稍微麻烦一点,首先给出一个利用伴随矩阵求逆的方法。定义新矩阵 $\boldsymbol{K}^*$,其第 i 行第 j 列的取值为 $(-1)^{i+j}\det \boldsymbol{K}_{ji}$($\boldsymbol{K}_{ji}$ 是通过删除 $\boldsymbol{K}$ 的第 j 行和第 i 列后形成的矩阵)。$\boldsymbol{K}^*$ 称为矩阵 $\boldsymbol{K}$ 的伴随矩阵。关于伴随矩阵,我们不加证明地给出下面的定理:

定理 1.3 设 $\boldsymbol{K} = (k_{i,j})$ 为一个定义在 $\mathbb{Z}_n$ 上的 $m \times m$ 矩阵。若 $\boldsymbol{K}$ 在 $\mathbb{Z}_n$ 上可逆,则有 $\boldsymbol{K}^{-1} = (\det \boldsymbol{K})^{-1}\boldsymbol{K}^*$,这里 $\boldsymbol{K}^*$ 为矩阵 $\boldsymbol{K}$ 的伴随矩阵。

注 利用伴随矩阵可以计算矩阵的逆,但当矩阵维数很大时,实际计算是很麻烦的。一般还是使用矩阵的初等行变换来计算其逆。

对 2×2 矩阵,很容易从定理 1.3 得到如下推论:

推论 1.4 设矩阵

$$\boldsymbol{K} = \begin{pmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{pmatrix}$$

为一个定义在 $\mathbb{Z}_n$ 上的矩阵。$\det \boldsymbol{K} = (k_{1,1}k_{2,2} - k_{1,2}k_{2,1}) \bmod n$ 是可逆的,则有:

$$\boldsymbol{K}^{-1} = (\det \boldsymbol{K})^{-1} \begin{pmatrix} k_{2,2} & -k_{1,2} \\ -k_{2,1} & k_{1,1} \end{pmatrix}$$

我们考虑前面提到的例子。首先有:

$$\det \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = 11 \times (7 - 8 \times 3) \bmod 26$$
$$= (77 - 24) \bmod 26$$

$$= 53 \bmod 26$$
$$= 1$$

显然，$1^{-1} \bmod 26 = 1$，故其逆矩阵为：

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

这和前面的验证是一致的。

再给出一个 3×3 矩阵的例子。

例 1.6　设矩阵

$$\boldsymbol{K} = \begin{pmatrix} 10 & 5 & 12 \\ 3 & 14 & 21 \\ 8 & 9 & 11 \end{pmatrix}$$

为定义在$\mathbb{Z}_{26}$上的矩阵。易验证 $\det K = 7$，并且有 $7^{-1} \bmod 26 = 15$。相应的伴随矩阵为：

$$\boldsymbol{K}^* = \begin{pmatrix} 17 & 1 & 15 \\ 5 & 14 & 8 \\ 19 & 2 & 21 \end{pmatrix}$$

故 $\boldsymbol{K}$ 的逆矩阵 $\boldsymbol{K}^{-1}$ 存在，且为：

$$\boldsymbol{K}^{-1} = 15\boldsymbol{K}^* = \begin{pmatrix} 21 & 15 & 17 \\ 23 & 2 & 16 \\ 25 & 4 & 3 \end{pmatrix}$$

下面给出$\mathbb{Z}_{26}$上希尔密码的具体描述。

密码体制 1.5　希尔密码

设 $m \geqslant 2$ 为正整数，$\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$，且

$$\mathcal{K} = \{\text{定义在}\mathbb{Z}_{26}\text{上的 } m \times m \text{ 可逆矩阵}\}$$

对任意的密钥 $\boldsymbol{K}$，定义加密变换：

$$e_K(x) = x\boldsymbol{K}$$

解密变换：

$$d_K(y) = y\boldsymbol{K}^{-1}$$

以上运算都是在$\mathbb{Z}_{26}$上进行的。

1.1.6 置换密码

前面讨论的密码体制都是代换密码:明文字母被不同的密文字母所代替。下面讨论的置换密码(Permutation Cipher)的特点是保持明文的所有字母不变,只是利用置换打乱了明文字母的位置和次序。

定义在有限集 X 上的置换为一双射函数 $\pi: X \to X$。换句话说,π 既是单射,又是满射。即对任意的 $x \in X$,存在惟一的 $x' \in X$ 使得 $\pi(x') = x$。这样,可以定义置换 π 的逆置换 $\pi^{-1}: X \to X$ 为:

$$\pi^{-1}(x) = x' \quad \text{当且仅当} \quad \pi(x') = x$$

则 π^{-1} 也是 X 上的一个置换。

密码体制 1.6 给出了置换密码的具体定义。置换密码的使用已经有数百年的历史,最早在 1563 年就由 Giovanni Porta 给出了置换密码和代换密码的具体区别。

密码体制 1.6 置换密码

令 m 为一正整数。$\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$,$\mathcal{K}$ 由所有定义在集合 $\{1, 2, \cdots, m\}$ 上的置换组成。对任意的密钥(置换)π,定义加密变换:

$$e_\pi(x_1, \cdots, x_m) = (x_{\pi(1)}, \cdots, x_{\pi(m)})$$

相应的解密变换为:

$$d_\pi(y_1, \cdots, y_m) = (y_{\pi^{-1}(1)}, \cdots, y_{\pi^{-1}(m)})$$

上式中 π^{-1} 为置换 π 的逆置换。

对于代换密码,由于在加密和解密过程中没有执行代数运算,使用字母比使用模 26 剩余更方便一些。

下面给出一个具体的例子。

例 1.7 设 $m = 6$,密钥为如下的置换 π:

x	1	2	3	4	5	6
$\pi(x)$	3	5	1	6	4	2

注意,上表的第一行是关于 $x(1 \leqslant x \leqslant 6)$ 值的列表,第二行是其相应的置换 $\pi(x)$。逆置换 π^{-1} 可重新安排第二行的次序得出:

x	1	2	3	4	5	6
$\pi^{-1}(x)$	3	6	1	6	2	4

假设我们要加密的明文为：

shesellsseashellsbytheseashore

首先，将明文字母分为每六个一组：

shesel | lsseas | hellsb | ythese | ashore

对每组的六个字母使用加密变换 π，则可得：

EESLSH | SALSES | LSHBLE | HSYEET | HRAEOS

因此，最后的密文如下：

EESLSHSALSESLSHBLEHSYEETHRAEOS

解密过程同加密过程一样，只不过使用的是逆置换 π^{-1}，在此不再给出。

事实上，置换密码是希尔密码的一种特殊情形，下面主要对这一点予以说明。给定集合 $\{1,\cdots,m\}$ 的一个置换 π，可按如下方法定义一个置换 π 的关联置换矩阵 $\boldsymbol{K}_\pi=(k_{i,j})_{m\times m}$：

$$k_{i,j}=\begin{cases}1 & \text{若 } i=\pi(j)\\ 0 & \text{其他}\end{cases}$$

（一个置换矩阵是指每行每列都恰好只有一个 1，其他都是 0 的矩阵。一个置换矩阵可以通过对单位矩阵进行行置换和列置换而得到。）

容易看出使用矩阵 $\boldsymbol{K}_\pi$ 为密钥的希尔密码事实上等价于使用密钥 π 进行加密的置换密码，并且还有 $\boldsymbol{K}_\pi^{-1}=\boldsymbol{K}_{\pi^{-1}}$，即 $\boldsymbol{K}_\pi$ 的逆矩阵是置换 π^{-1} 的关联置换矩阵。这说明二者的解密变换也是等价的。

对前面例子中的置换 π，其关联置换矩阵为：

$$\boldsymbol{K}_\pi=\begin{pmatrix}0&0&1&0&0&0\\0&0&0&0&0&1\\1&0&0&0&0&0\\0&0&0&0&1&0\\0&1&0&0&0&0\\0&0&0&1&0&0\end{pmatrix}$$

并且还有：

$$\boldsymbol{K}_\pi^{-1}=\begin{pmatrix}0&0&1&0&0&0\\0&0&0&0&1&0\\1&0&0&0&0&0\\0&0&0&0&0&1\\0&0&0&1&0&0\\0&1&0&0&0&0\end{pmatrix}$$

很容易验证以上两矩阵的乘积恰为单位矩阵。

1.1.7 流密码

前面研究的密码体制中,连续的明文元素是使用相同的密钥 K 来加密的,即密文串使用如下方法得到:

$$y = y_1 y_2 \cdots = e_K(x_1) e_K(x_2) \cdots$$

满足这种特点的密码体制我们通常称为分组密码。

另外一种被广泛使用的密码体制称为流密码(Stream Cipher),其基本思想是产生一个密钥流 $z = z_1 z_2 \cdots$,然后使用它根据下述规则来加密明文串 $x = x_1 x_2 \cdots$:

$$y = y_1 y_2 \cdots = e_{z_1}(x_1) e_{z_2}(x_2) \cdots$$

最简单的流密码是其密钥流直接由初始密钥使用某种算法变换得来的,密钥流和明文串是相互独立的。这种类型的流密码称为"同步"流密码,正式定义如下:

定义 1.6 同步流密码为一六元组$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{L},\mathcal{E},\mathcal{D})$和函数 g,并且满足如下条件:

1.$\mathcal{P}$是由所有可能明文构成的有限集。

2.$\mathcal{C}$是由所有可能密文构成的有限集。

3.密钥空间$\mathcal{K}$为一有限集,由所有可能密钥构成。

4.$\mathcal{L}$是一个称为密钥流字母表的有限集。

5.g 是一个密钥流生成器。g 使用密钥 K 作为输入,产生无限的密钥流 $z = z_1 z_2 \cdots$, $z_i \in \mathcal{L}$, $i \geqslant 1$。

6.对任意的 $z \in \mathcal{L}$,都有一加密规则 $e_z \in \mathcal{E}$和相应的解密规则 $d_z \in \mathcal{D}$。并且对每一明文 $x \in \mathcal{P}$, $e_z: \mathcal{P} \to \mathcal{C}$和 $d_z: \mathcal{C} \to \mathcal{P}$是满足 $d_z(e_z(x)) = x$ 的函数。

我们利用前面提到的维吉尼亚密码给同步流密码定义一个解释。假设 m 为维吉尼密码的密钥长度,定义$\mathcal{K} = (\mathbb{Z}_{26})^m$, $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathbb{Z}_{26}$;定义 $e_z(x) = (x + z) \bmod 26$, $d_z(y) = (y - z) \bmod 26$。再定义密钥流 $z_1 z_2 \cdots$,如下所示:

$$z_i = \begin{cases} k_i & \text{若 } 1 \leqslant i \leqslant m \\ z_{i-m} & \text{若 } i \geqslant m+1 \end{cases}$$

上式中 $K = (k_1, k_2, \cdots, k_m)$,这样利用 K 可产生的密钥流如下:

$$k_1 k_2 \cdots k_m k_1 k_2 \cdots k_m k_1 k_2 \cdots$$

注 分组密码可以认为是流密码的特殊情况,即对所有 $i \geqslant 1$,密钥流为一常数 $z_i = K$。

如果对所有 $i \geqslant 1$ 的整数有 $z_{i+d} = z_i$，则称该流密码为具有周期 d 的周期流密码。如上面分析的密钥字长为 m 的维吉尼亚密码可看做是周期为 m 的流密码。

流密码通常以二元字符来表示，即 $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathbb{Z}_2$，此时加密解密刚好都可看做模 2 的加法：

$$e_z(x) = (x + z) \bmod 2$$

和

$$d_z(y) = (y + z) \bmod 2$$

如果认为“0”代表布尔值为“假”，“1”代表布尔值为“真”，那么模2加法对应于异或运算。这样，加密解密都可用硬件方便地实现。

下面给出另一个产生(同步)密钥流的方法。假设以 $(k_1, k_2, \cdots, k_m)$ 开始，并且 $z_i = k_i, 1 \leqslant i \leqslant m$。利用次数为 m 的线性递归关系来产生密钥流：

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2$$

这里 $c_0, c_1, \cdots, c_{m-1} \in \mathbb{Z}_2$ 是确定的常数。

注　这个递归关系的次数为 m，是因为每一个项都依赖于前面 m 个项；又因为 z_{i+m} 是前面项的线性组合，故称其为线性的。注意，不失一般性，我们取 $c_0 = 1$，否则递归关系的次数将为 $m - 1$。

这里密钥 K 由 $2m$ 个值 $k_1, \cdots, k_m, c_0, \cdots, c_{m-1}$ 组成。如果 $(k_1, \cdots, k_m) = (0, \cdots, 0)$，则生成的密钥流全为零，当然这种情况是需要避免的，否则明文将与密文相同。另一方面，如果常数 $c_0, \cdots, c_{m-1}$ 选择适当的话，则任意非零初始向量 $(k_1, \cdots, k_m)$ 都将产生周期为 $2^m - 1$ 的密钥流。这种利用“短”的密钥来产生较长的密钥流的方法，正是我们希望看到的，后面将用实例来说明具有短周期密钥流的维吉尼亚密码是很容易被攻破的。

下面给出一个具体例子。

例 1.8　设 $m = 4$，密钥流按如下线性递归关系产生：

$$z_{i+4} = (z_i + z_{i+1}) \bmod 2, \quad i \geqslant 1$$

如果密钥流的初始向量不为零，则我们将获得周期为 $2^4 - 1 = 15$ 的密钥流。例如，若初始向量为(1,0,0,0)，则可产生密钥流如下：

$$1\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\cdots$$

任何一个非零的初始向量都将产生具有相同周期的密钥流序列。

这种密钥流产生方法的另外一个诱人之处在于密钥流能使用线性反馈移位寄存器(LFSR)以硬件的方式来有效地实现。使用具有 m 个状态的移位寄存器，向量 $(k_1, \cdots, k_m)$ 用来初始化移位寄存器，在每一个时间单元，自动完成下列运算：

1. k_1 被抽出作为下一个密钥流比特。

2. $k_2, \cdots, k_m$ 分别左移一个状态位。

3. 新的 k_m 值由下式"线性反馈" 给出:

$$\sum_{j=0}^{m-1} c_j k_{j+1}$$

我们可以看出,线性反馈是通过抽取寄存器的某级状态并计算模 2 加法来进行的,如图 1.2 所示,其对应的 LFSR 将产生例 1.8 中的密钥流。

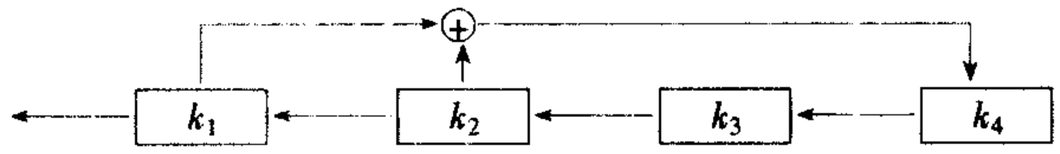

图 1.2 线性反馈移位寄存器

在流密码中,还有这样一种情况,密钥流 z_i 的产生不但与密钥 K 有关,而且还与明文元素 $(x_1, \cdots, x_{i-1})$ 或密文元素 $(y_1, \cdots y_{i-1})$ 有关,这类流密钥我们称为异步流密码。下面给出一个来源于维吉尼亚密码的异步流密码,称做自动密钥密码。称为"自动密钥" 的原因是因为它使用明文来构造密钥流(除了最初的"原始密钥" 外)。当然,由于仅有 26 个可能的密钥,自动密钥密码是不安全的。

密码体制 1.7 自动密钥密码

设 $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$, $x_1 = K$,定义 $z_i = x_{i-1}$, $i \geqslant 2$。对任意的 $0 \leqslant z \leqslant 25$, $x, y \in \mathbb{Z}_{26}$,定义

$$e_z(x) = (x + z) \bmod 26$$

和

$$d_z(y) = (y - z) \bmod 26$$

下面给出一个例子。

例 1.9 假设 $K = 8$,明文为:

rendezvous

首先将明文转换为整数序列:

17 4 13 3 4 25 21 14 20 18

相应的密钥流是:

8 17 4 13 3 4 25 21 14 20

将对应的元素相加,并通过模 26 约简,得:

25　21　17　16　7　3　20　9　8　12

以字母形式的密文就是：

ZVRQHDUJIM

解密时，Alice 首先转换密文字母为相应的数字串：

25　21　17　16　7　3　20　9　8　12

然后计算：

$$x_1 = d_8(25) = (25 - 8) \bmod 26 = 17$$

再计算：

$$x_2 = d_{17}(21) = (21 - 17) \bmod 26 = 4$$

这样一直做下去，每次获得下一个明文字母，用它作为下一个密钥流元素。

下一节，主要讨论密码分析，并针对我们前面提出的各种密码体制进行分析破译。

1.2　密码分析

本节讨论密码分析技术。一般情况下，我们都假设敌手 Oscar 知道正在使用的密码体制，这个假设通常称为 Kerckhoff 假设。当然，如果 Oscar 不知道具体的密码体制，那么完成密码分析将更加困难。但我们不能将密码体制的安全性建立在敌手 Oscar 不知道具体的密码体制这样一种（可能不确定的）假设上。因此，我们的目标是设计在 Kerchhoff 假设下安全的密码体制。

首先，我们想找出对于密码体制的不同攻击模型之间的区别。一个攻击模型就确定了敌手进行攻击时可得到的信息。最常见的攻击类型有：

惟密文攻击(ciphertext only attack)：敌手只有密文串 y。

已知明文攻击(known plaintext attack)：敌手掌握明文串 x 和对应的密文串 y。

选择明文攻击(chosen plaintext attack)：敌手可获得对加密机的访问权限，这样他能获得任意的明文串 x 对应的密文串 y。

选择密文攻击(chosen ciphertext attack)：敌手可获得对解密机的访问权限，这样他能获得任意的密文串 y 对应的明文串 x。

在以上任何一种情况下，敌手的目标都是为了确定正在使用的密钥。显然，这四种类型的攻击强度依次增大。我们将会看到，选择密文攻击主要与后面几章将要介绍的公钥密码体制相关。

首先考虑最弱类型的攻击，即惟密文攻击，这里假设明文串是不包括标点符号及空格的普通英文文本(当然，如果待加密明文中有标点符号和空格，那么具体分析起来将更加困难)。

许多密码分析方法都利用了英文语言的统计特性。目前已经有许多从各种小说、杂志和报纸上统计的26个英文字母出现的频率,表1.1的统计数据由Beker和Piper给出。

表1.1 26个英文字母出现的概率

字母	概率	字母	概率
A	0.082	N	0.067
B	0.015	O	0.075
C	0.028	P	0.019
D	0.043	Q	0.001
E	0.127	R	0.060
F	0.022	S	0.063
G	0.020	T	0.091
H	0.061	U	0.028
I	0.070	V	0.010
J	0.002	W	0.023
K	0.008	X	0.001
L	0.040	Y	0.020
M	0.024	Z	0.001

在表1.1的基础上,Beker和Piper把26个英文字母划分成如下5组:

1. E的概率大约为0.120。

2. T、A、O、I、N、S、H、R的概率为0.06 ~ 0.09。

3. D、L的概率大约为0.04。

4. C、U、M、W、F、G、Y、P、B的概率为0.015 ~ 0.023。

5. V、K、J、X、Q、Z的概率小于0.01。

另外,考虑两字母组或三字母组组成的固定序列也是很有用的。以下是30个最常见的两字母组(按出现次数递减排序):TH、HE、IN、ER、AN、RE、DE、ON、ES、ST、EN、AT、TO、NT、HA、ND、OU、EA、NG、AS、OR、TI、IS、ET、IT、AR、TE、SE、HI和OF。12个最常见的三字母组(按出现次数递减排序)为:THE、ING、AND、HER、ERE、ENT、THA、NTH、WAS、ETH、FOR和DTH。

1.2.1 仿射密码的密码分析

可利用统计数据来破译仿射密码。我们看一个简单例子,假设Oscar已经截获到了下列密文。

例1.10 从仿射密码中获得如下密文:

FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKAPRKDLYEVLRHHRH

这些密文的频率分析参见表 1.2。

表 1.2　密文出现字母频率统计

字母	频率	字母	频率
A	2	N	1
B	1	O	1
C	0	P	2
D	7	Q	0
E	5	R	8
F	4	S	3
G	0	T	0
H	5	U	2
I	0	V	4
J	0	W	0
K	5	X	2
L	2	Y	1
M	2	Z	0

这里虽然只有 57 个字母，但它足以分析仿射密码，最大频率的密文字母是：R(8 次)，D(7 次)，E、H、K(每个 5 次) 和 S、F、V(各 4 次)。首先，我们可以猜想 R 是 e 的加密而 D 是 t 的加密，因为 e 和 t(分别) 是两个出现频率最高的字母。以数字表达即为 $e_K(4) = 17$ 和 $e_K(19) = 3$，因为 $e_K(x) = ax + b$，这里 a 和 b 是未知的，所以我们有如下的关于两个未知数的线性方程组：

$$4a + b = 17$$
$$19a + b = 13$$

这个方程组有惟一解 $a = 6, b = 19$(在 $\mathbb{Z}_{26}$)，但这是一个不合法的密钥，因为 $\gcd(a,26) = 2 > 1$。所以我们的猜想肯定是不正确的。

我们再猜测 R 是 e 的加密，而 E 是 t 的加密，继续使用上述的方法，得到 $a = 13$，这也是一个不合法的密钥。再试一种可能性：R 是 e 的加密，H 是 t 的加密，则有 $a = 8$，这也是不合法的。继续进行，我们猜测 R 是 e 的加密，K 是 t 的加密，这样可得 $a = 3, b = 5$，首先它至少是一个合法的密钥，下一步工作就是检验密钥 $K = (3,5)$ 的正确性。如果我们能得到有意义的英文字母串，则可证实是有效的。

解密函数为 $d_K(y) = 9y - 19$，对密文进行解密有：

algorithmasarequitegeneraldefinitionsofarithmeticprocesses

显然所得的密钥是正确的。

1.2.2 代换密码的密码分析

下面来看一个更为复杂的情况:代换密码。

例 1.11 考虑如下利用代换密码加密的密文

YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

此密文的频率分析由表 1.3 给出

表 1.3 出现字母的频率统计

字母	频率	字母	频率
A	0	N	9
B	1	O	0
C	15	P	1
D	13	Q	4
E	7	R	10
F	11	S	3
G	1	T	2
H	4	U	5
I	5	V	5
J	11	W	8
K	1	X	6
L	0	Y	10
M	16	Z	20

因为 Z 出现的次数高于任何其他密文字母,所以我们可以猜测 $d_K(Z) = e$,出现 10 次以上的其余的密文字母是 C、D、F、J、M、R、Y。我们希望这些字母对应的是 t、a、o、i、n、s、h、r(子集合)的加密,但实际的频率变化很难看出它们之间的对应关系。

注意一下形如 - Z 或 Z - 的两字母组,因为已经假设 Z 解密成 e。我们发现,出现这种类型的最一般的两字母是 DZ 和 ZW(每个出现 4 次);NZ 和 ZU(每个出现 3 次);RZ、HZ、XZ、FZ、ZR、ZV、ZC、ZD 和 ZJ(每个出现 2 次),因为 ZW 出现 4 次而 WZ 一次也未出现,同时 W 比许多其他字母出现的次数少,所以我们可以假定 $d_K(W) = d$。又因为 DZ 出现 4 次而 ZD 出现 2 次,故可猜测 $D_K(D) \in \{r,s,t\}$,但具体是哪一个还不太清楚。

如前面的猜测,假设 $d_K(Z) = e, d_K(W) = d$,回过头再看看密文并注意到 ZRW 和 RZW 出现在密文的开始部分,RW 在后面也出现过。因为 R 在密文中频繁地出现,而 nd 是一个常见的两字母组,所以我们可以视 $d_K(R) = n$ 为可能的情况。

这样,我们就有如下的形式:

```
------end---------e----ned---e-----------
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

--------e----e----------n--d---en----e----e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

-e---n------n------ed---e---e -ne-nd-e-e--
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-----n----------e----ed-------d---e--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

下一步我们可以试试 $d_K(N) = h$,因为 NZ 是一个常见的两字母组而 ZN 不是常见的两字母组。如果这个猜测是正确的,则明文 ne – ndhe 很可能说明 $d_K(C) = a$。结合这个假设,我们进一步又有:

```
------end-----a---e-a--nedh--e------a-----
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h-------ea---e-a---a---nhad---en--a-e-h--e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-a-n------n------ed---e---e--neandhe-e--
NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-a---nh---ha---a-e----ed-----a-d--he--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

现在我们考虑出现次数较高的密文字母 M,由前面分析,密文段 RNM 解密成 nh –,这说明 h – 是一个词的开头,所以 M 很可能是一个元音。因为已经使用了 a 和 e,所以猜测 $d_K(M) = i$ 或 o。因为 ai 是一个比 ao 出现次数更多的明文组,所以首先猜得 $d_K(M) = i$,这样我们有:

```
-----iend-----a-i-e-a-inedhi-e-------a---i-
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h-----i-ea-i-e-a---a-i-nhad-a-en--a-e-hi-e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-a-n-----in-i----ed---e---e-ineandhe-e--
ZNUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-a--inhi--hai--a-e-i--ed-----a-d--he--n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

下面需要确定明文 o 对应的密文。因为 o 是一个经常出现的字母,所以我们猜测相应的密文字母是 D、F、J、Y 中的一个。Y 似乎最有可能,否则将得到长串的元音字母,即从 CFM 或 CJM 中得到 aoi。因此,我们假设 $d_K(Y) = o$。

剩下密文字母中三个最高频率的字母是 D、F、J,我们猜测它们以某种次序解密成 r、s、t,三

字母 NMD 两次出现说明很可能 $d_K(Y) = o$,对应的明文三字母组为 his(这与前面假设 $d_K(D) \in \{r,s,t\}$ 是一致的)。HNCMF 可能是 chair 的加密,它说明 $d_K(F) = r$(同时 $d_K(H) = c$),这样通过排除法有 $d_K(J) = t$,现在我们有:

```
o-r-riend-ro--arise-a-inedhise--t---ass-it
YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

hs-r-riseasi-e-a-orationhadta-en--ace-hi-e
NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he-asnt-oo-in-i-o-redso-e-ore-ineandhesett
ZNUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

-ed-ac-inhischair-aceti-ted--to-ardsthes-n
XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR
```

有了上面的提示,我们很容易确定出明文和例 1.11 中的密钥,解密明文如下:

Our friend from Paris examined his empty glass with surprise, as if evaporation had taken place while he wasn't looking. I poured some more wine and he settled back in his chair, face tilted up towards the sun. (选自 P.Mayle, A Year in Provence, A. Knopf, Inc., 1989.)

1.2.3 维吉尼亚密码的密码分析

本小节给出分析维吉尼亚密码的一些方法。首先必须确定密钥字的长度 m,这里我们介绍两种方法:Kasiski 测试法和重合指数法。

Kasiski 测试由 Friedrich Kasiski 在 1863 年给出描述,但是,更早在 1854 年这一方法就被 Charles Babbage 首先发现。它主要是基于这样一个事实:两个相同的明文段将加密成相同的密文段,它们的位置间距假设为 δ,则 $\delta = 0 \pmod m$。反过来,如果在密文中观察到两个相同的长度至少为 3 的密文段,那么将给破译者带来很大方便,因为它们实际上对应了相同的明文串。

Kasiski 测试过程如下:搜索长度至少为 3 的相同的密文段,记录这些相同密文段到起始点之间的距离;假如得到如下几个距离 $\delta_1, \delta_2, \cdots$,那么,可以猜测 m 为这些 δ_i 的最大公因子的因子。

求 m 值的进一步的方法是使用所谓的重合指数法,这种方法由 Wolfe Friedman 在 1920 年首先提出,具体过程如下。

定义 1.7 设 $\boldsymbol{x} = x_1 x_2 \cdots x_n$ 是一个含有 n 个字符的字符串,$\boldsymbol{x}$ 的重合指数记为 $I_c(\boldsymbol{x})$,定义为 $\boldsymbol{x}$ 中两个随机元素相同的概率。

假设 $f_0, f_1, \cdots, f_{25}$ 分别表示 A,B,…,Z 在 $\boldsymbol{x}$ 中出现的频率,共有 $\binom{n}{2}$ 种方法来选择 $\boldsymbol{x}$ 中的任两个元素(二项式系数 $\binom{n}{k} = n!/(k!(n-k)!)$ 表示从 n 个物体中取出 k 个物体的方式的个

数)。对每一个 i,$0 \leqslant i \leqslant 25$,共有$\binom{f_i}{2}$种方法使得所选的两个元素皆为 i。因此,有如下公式:

$$I_c(\boldsymbol{x}) = \frac{\sum_{i=0}^{25}\binom{f_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}$$

假设 $\boldsymbol{x}$ 是英语文本串,记表 1.1 中字母 A,B,…,Z 出现的期望概率为 $p_0, p_1, \cdots, p_{25}$,那么我们期望:

$$I_c(\boldsymbol{x}) \approx \sum_{i=0}^{25} p_i^2 = 0.065$$

上式成立主要是因为两个随机元素都是 A 的概率为 p_0^2,两个随机元素都为 B 的概率为 p_1^2,等等。如果 $\boldsymbol{x}$ 是通过单表代换密码而得来的,其分析原理相同。此时,各个概率将被置换,但量 $\sum_{i=0}^{25} p_i^2$ 将不会改变。

假设我们使用维吉尼亚密码加密的密文串为 $\boldsymbol{y} = y_1 y_2 \cdots y_n$。将串 $\boldsymbol{y}$ 分割为 m 个长度相等的子串,分别为 $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m$,这样可以以列的形式写出密文,组成一个 $m \times (n/m)$ 矩阵。矩阵的每一行对应于子串 $\boldsymbol{y}_i$,$1 \leqslant i \leqslant m$。换言之,我们有如下形式:

$$\begin{aligned}
\boldsymbol{y}_1 &= y_1 y_{m+1} y_{2m+1} \cdots \\
\boldsymbol{y}_2 &= y_2 y_{m+2} y_{2m+2} \cdots \\
&\vdots \quad \vdots \quad \vdots \\
\boldsymbol{y}_m &= y_m y_{2m} y_{3m} \cdots
\end{aligned}$$

如果 $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_m$ 按如上方法构造,则 m 实际上就是密钥字的长度,每一个 $I_c(\boldsymbol{y}_i)$ 的值大约为 0.065。另一方面,如果 m 不是密钥字的长度,那么子串 $\boldsymbol{y}_i$ 看起来更为随机,因为它们是通过不同密钥以移位加密方式获得的。易知,对一个完全的随机串,其重合指数为:

$$I_c \approx 26(1/26)^2 = \frac{1}{26} = 0.038$$

值 0.065 和 0.038 的差别是比较大的,按这种方法通常可以确定密钥字的长度(或确信一个利用 Kasiski 方法猜测的密钥字长)。

这里我们用一个例子来说明这两种技术。

例 1.12　已知使用维吉尼亚密码加密获得如下密文:

```
CHREEVOAHMAERATBIAXXWTNXBEEOPHBSBQMQEQERBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK
LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX
VRVPRTULHDNQWTWDTYGBPHXTFALJHASVBFXNGLLCHR
ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT
AMRVLCRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBBI
PEEWEVKAKOEWADREMXMTBJJCHRTKDNVRZCHRCLQOHP
WQAIIWXNRMGWOIIFKEE
```

首先使用 Kasiski 测试法。在密文中密文串 CHR 共出现在 5 个位置,开始位置分别为 1、166、236、276 和 286,其距离分别为 165、235、275 和 285。这三个整数的最大公约数为 5,故我们猜测密钥字的长度很可能为 5。

我们再使用重合指数法确认这一猜测。当 $m = 1$ 时,重合指数为 0.045;当 $m = 2$ 时,两个重合指数分别为 0.046 和 0.041;当 $m = 3$ 时,为 0.043、0.050 和 0.047;当 $m = 4$ 时,为 0.042、0.039、0.046 和 0.040;当 $m = 5$ 时,可获得的值分别为 0.063、0.068、0.069、0.061 和 0.072。这些值为密钥字的长度为 5 提供了强有力的证据。

现在假设 $m = 5$,怎样来确定具体的密钥 $K = (k_1, k_2, \cdots, k_m)$ 呢?下面给出一个简单而有效的方法。令 $1 \leqslant i \leqslant m$,$f_0, f_1, \cdots, f_{25}$ 分别表示串 $\boldsymbol{y}_i$ 中字母 A,B,…,Z 出现的频率。再令 $n' = n/m$ 表示串 $\boldsymbol{y}_i$ 的长度,则 26 个字母在 $\boldsymbol{y}_i$ 中出现的概率为:

$$\frac{f_0}{n'}, \frac{f_1}{n'}, \cdots, \frac{f_{25}}{n'}$$

考虑到子串 $\boldsymbol{y}_i$ 是由对应的待加密的明文子集中的字母移动 k_i 个位置所得的。因此,移位后的概率分布

$$\frac{f_{k_i}}{n'}, \cdots, \frac{f_{k_i+25}}{n'}$$

应该近似于表 1.1 中统计得出的概率分布 $p_0, p_1, \cdots, p_{25}$。

假设 $0 \leqslant g \leqslant 25$,定义数值:

$$M_g = \sum_{i=0}^{25} \frac{p_i f_{i+g}}{n'} \tag{1.1}$$

如果 $g = k_i$,类似于前面重合指数的讨论,应该有:

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

如果 $g \neq k_i$,则 M_g 一般应该小于 0.065(证明参见练习)。对任意的 i,$1 \leqslant i \leqslant m$,使用这种方法就可以具体确定每一个 k_i 的值。

我们再回头考虑例 1.12。

例 1.12(续) 前面已经假设密钥字长为 5。下面对任意 $1 \leqslant i \leqslant 5$,计算其对应的 M_g 的值。结果在表 1.4 中列出。对每一个 i,找寻其 M_g 值接近于 0.065 的那一个。这些具体的 g 决定了相应的移位 $k_1, \cdots, k_5$。

由表 1.4 中的数据可知,密钥很可能为 $K = (9,0,13,4,19)$,对应的字母为 JANET。容易验证它是正确的,解密后的明文为:

The almond tree was in tentative blossom. The days were longer, often ending with magnificent evenings of corrugated pink skies. The hunting season was over, with hounds and guns put away for six months. The vineyards were busy again as well-organized farmers treated their vines and the more

lackadaisical neighbors hurried to do the pruning they should have done in November.　（选自 P.Mayle, A Year in Provence, A. Knopf, Inc., 1989.）

表 1.4　M_g 的值

i	$M_g(y_i)$ 的值								
1	.035	.031	.036	.037	.035	.039	.028	.028	.048
	.061	.039	.032	.040	.038	.038	.044	.036	.030
	.042	.043	.036	.033	.049	.043	.041	.036	
2	.069	.044	.032	.035	.044	.034	.036	.033	.030
	.031	.042	.045	.040	.045	.046	.042	.037	.032
	.034	.037	.032	.034	.043	.032	.026	.047	
3	.048	.029	.042	.043	.044	.034	.038	.035	.032
	.049	.035	.031	.035	.065	.035	.038	.036	.045
	.027	.035	.034	.034	.037	.035	.046	.040	
4	.045	.032	.033	.038	.060	.034	.034	.034	.050
	.033	.033	.043	.040	.033	.028	.036	.040	.044
	.037	.050	.034	.034	.039	.044	.038	.035	
5	.034	.031	.035	.044	.047	.037	.043	.038	.042
	.037	.033	.032	.035	.037	.036	.045	.032	.029
	.044	.072	.036	.027	.030	.048	.036	.037	

1.2.4　希尔密码的密码分析

采用惟密文攻击希尔密码是很难攻破的，但是如果采用已知明文攻击，则很容易破译希尔密码。假定敌手已经确定了正在使用的 m 值，至少有 m 个不同的明-密文对，设为：

$$x_j = (x_{1,j}, x_{2,j}, \cdots, x_{m,j})$$
$$y_j = (y_{1,j}, y_{2,j}, \cdots, y_{m,j})$$

对任意的 $1 \leqslant j \leqslant m$，有 $y_j = e_K(x_j)$。如果我们定义两个 $m \times m$ 矩阵 $\boldsymbol{X} = (x_{i,j})$ 和 $\boldsymbol{Y} = (y_{i,j})$，则有矩阵方程 $\boldsymbol{Y} = \boldsymbol{XK}$，其中 $m \times m$ 矩阵 $\boldsymbol{K}$ 是未知密钥。假如矩阵 $\boldsymbol{X}$ 刚好是可逆的，则敌手 Oscar 可轻松计算出 $\boldsymbol{K} = \boldsymbol{X}^{-1}\boldsymbol{Y}$，从而破译希尔密码（如果 $\boldsymbol{X}$ 不可逆，则必须重新选择 m 个明-密文对）。

下面给出一个小例子。

例 1.13　假设明文“friday” 利用 $m = 2$ 的希尔密码加密，得到密文为“PQCFKU”。

首先我们有 $e_K(5,17) = (15,16)$，$e_K(8,2) = (2,5)$，$e_K(0,24) = (10,20)$。使用头两个明-密文对，可得矩阵方程：

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

利用引理 1.4，容易计算

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

因此,

$$\boldsymbol{K} = \begin{pmatrix} 9 & 1 \\ 2 & 15 \end{pmatrix}\begin{pmatrix} 15 & 16 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 7 & 19 \\ 8 & 3 \end{pmatrix}$$

这个结果可以使用第三个明-密文对进行验证。

假如敌手不知道 m 的具体值,该如何攻击呢?假定 m 不是太大,他将简单地试 $m = 2, 3, \cdots$,直到发现密钥为止。如果假定 m 的值不正确,则可利用其余的明-密文对进行验证查出。使用这种方法,即使在不知道 m 值的情况下,也可以破译希尔密码。

1.2.5 基于 LFSR 流密码的密码分析

在前面介绍的流密码中,密文是明文和密钥流的模2加,即 $y_i = (x_i + z_i) \bmod 2$。利用下列线性递归关系从初态 $(z_1, z_2, \cdots, z_m) = (k_1, k_2, \cdots, k_m)$ 产生密钥流:

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2,\ i \geqslant 1$$

这里 $c_0, c_1, \cdots, c_{m-1} \in \mathbb{Z}_2$。

因为这个密码体制中所有运算都是线性的,同前面的希尔密码一样,它容易受到已知明文攻击。假定 Oscar 有了明文串 $x_1 x_2 \cdots x_n$ 和相应的密文串 $y_1 y_2 \cdots y_n$,那么他能计算密钥流比特 $z_i = (x_i + y_i) \bmod 2, 1 \leqslant i \leqslant n$。若 Oscar 再知道 m 的值,那么 Oscar 仅需要计算 $c_0, c_1, \cdots, c_{m-1}$ 的值就能重构整个密钥流。换句话说,他只需要确定 m 个未知的值就够了。

现在已知,对任何 $i \geqslant 1$,我们有

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2$$

它是 m 个未知数的线性方程。如果 $n \geqslant 2m$,就有 m 个未知数的 m 个线性方程,利用它就可以解出这 m 个未知数。

m 个线性方程能以矩阵形式表示为:

$$(z_{m+1}, z_{m+2}, \cdots, z_{2m}) = (c_0, c_1, \cdots, c_{m-1})\begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{bmatrix}$$

如果系数矩阵有逆(模2),则可解得:

$$(c_0, c_1, \cdots, c_{m-1}) = (z_{m+1}, z_{m+2}, \cdots, z_{2m})\begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{bmatrix}^{-1}$$

事实上,如果 m 是产生密钥流的递归次数,那么这个矩阵一定是可逆的(证明参见练习)。

这里我们仍然给出一个例子。

例 1.14　假设 Oscar 得到密文串:

$$1\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,0\,1\,0$$

和相应的明文串:

$$0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0$$

那么他能计算出密钥流比特是:

$$1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0$$

假定 Oscar 也知道密钥流是使用 5 级 LFSR 产生的,那么他利用前面 10 个比特就可得到如下方程组:

$$(0,1,0,0,0) = (c_0, c_1, c_2, c_3, c_4)\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Oscar 易求得:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

这样可解得

$$(c_0, c_1, c_2, c_3, c_4) = (0,1,0,0,0)\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$= (1,0,0,1,0)$$

由此,可知用来产生密钥流的递归公式为:

$$z_{i+5} = (z_i + z_{i+3}) \bmod 2$$

1.3　注释与参考文献

有关经典密码学的许多资料在一些教科书中都已经谈到,例如 Bauer[5] 所著的《Decrypted Secrets, Methods and Maxims of Cryptology》;Beker 和 Piper[8] 的《Cipher System, The Protection of Communications》;Beutelspacher[20] 的《Cryptology》;Denning[63] 的《Cryptography and Data

Security》;Kippenhahn[116] 的《Code Breaking, A History and Exploration》;Konheim[123] 的《Cryptography, A Primer》;van der Lubbe[136] 的《Basic Methods of Cryptography》。

前面关于 26 个字母的使用频率的统计数据引自 Beker 和 Piper[8]。

关于基础数论方面的知识可参考 Rosen[182] 的《Elementary Number Theory and its Application》一书。关于线性代数的知识可参考 Anton[3] 的《Elementary Linear Algebra》一书。

Kahn[109] 的《The Codebreakers》是一本关于密码学史的趣味性、知识性很强的读物;另外一本有关这方面的书可参见 Singh[197] 所著的《The Code Book》。

练习

1.1 计算下列数值:

(a) 7503 mod 81

(b) (- 7503) mod 81

(c) 81 mod 7503

(d) (- 81) mod 7503

1.2 设 $a, m > 0$,且 $a \not\equiv 0 \pmod{m}$。证明:

$$(-a) \bmod m = m - (a \bmod m)$$

1.3 证明 $a \bmod m = b \bmod m$ 当且仅当 $a \equiv b \pmod{m}$。

1.4 证明 $a \bmod m = a - \left\lfloor \frac{a}{m} \right\rfloor m$,这里 $\lfloor x \rfloor = \max\{y \in \mathbb{Z} : y \leqslant x\}$。

1.5 使用穷尽密钥搜索法,破译如下利用移位密码加密的密文:

BEEAKFYDJXUQYHYJIQRYHTYJIQFBQDUYJIIKFUHCQD

1.6 在一个密码体制中,如果一个加密函数 e_K 和一个解密函数 d_K 相同,我们将这样的密钥 K 称为对合密钥。试找出定义在 $\mathbb{Z}_{26}$ 上的移位密码体制中的所有对合密钥。

1.7 计算 $m = 30, 100$ 和 1225 定义在 $\mathbb{Z}_m$ 上的仿射密码密钥量大小。

1.8 找出 $m = 28, 33$ 和 35 在 $\mathbb{Z}_m$ 上的所有可逆元。

1.9 设 $1 \leqslant a \leqslant 28$,利用反复试验的方法求出 $a^{-1} \bmod 29$ 的值。

1.10 已知 $K = (5, 21)$ 为定义在 $\mathbb{Z}_{29}$ 上的仿射密码的密钥。

(a) 以 $d_K(y) = a'y + b'$ 的形式给出解密函数,这里 $a', b' \in \mathbb{Z}_{29}$。

(b) 证明对任意的 $x \in \mathbb{Z}_{29}$,都有 $d_K(e_K(x)) = x$。

1.11 (a) 假设 $K = (a, b)$ 为定义在 $\mathbb{Z}_n$ 上的仿射密码的密钥。证明 K 是对合密钥当且仅当 $a^{-1} \bmod n = a$ 且 $b(a + 1) \equiv 0 \pmod{n}$。

(b) 试求出 $\mathbb{Z}_{15}$ 上的仿射密码的所有对合密钥。

(c) 设 $n = pq$,这里 p, q 为不同的奇素数,证明定义在 $\mathbb{Z}_n$ 上的所有仿射密码的对合密钥量为 $n + p + q + 1$。

1.12 (a) 设 p 为一素数,证明在 $\mathbb{Z}_p$ 上 2×2 可逆矩阵的数目为 $(p^2 - 1)(p^2 - p)$。

提示:因为 p 是素数,所以 $\mathbb{Z}_p$ 是一个域,则在一个域中矩阵是可逆的当且仅当它的行向量是线性无关的。

(b) 设 p 是素数,$m \geqslant 2$ 是一个整数,试求出 $\mathbb{Z}_p$ 上可逆的 $m \times m$ 矩阵的计算公式。

1.13　设 $n = 6, 9$ 和 26,问定义在 $\mathbb{Z}_n$ 上的 2×2 可逆矩阵有多少?

1.14　(a) 设 $\boldsymbol{A}$ 为 $\mathbb{Z}_{26}$ 上的矩阵且 $\boldsymbol{A} = \boldsymbol{A}^{-1}$,证明 $\det \boldsymbol{A} \equiv \pm 1 \pmod{26}$。

(b) 利用推论 1.4 求出当 $m = 2$ 时定义在 $\mathbb{Z}_{26}$ 上的希尔密码的所有对合密钥。

1.15　求以下定义在 $\mathbb{Z}_{26}$ 上的矩阵的逆:

(a) $\begin{pmatrix} 2 & 5 \\ 9 & 5 \end{pmatrix}$

(b) $\begin{pmatrix} 1 & 11 & 12 \\ 4 & 23 & 2 \\ 17 & 15 & 9 \end{pmatrix}$

1.16　(a) 设 π 为集合 $\{1, \cdots, 8\}$ 上的置换:

x	1	2	3	4	5	6	7	8
$\pi(x)$	4	1	6	2	7	3	8	5

求出逆置换 π^{-1}。

(b) 解密如下使用 $m = 8$ 置换密码加密的密文,密钥为(a) 中的 π^{-1}。

ETEGENLMDNTNEOORDAHATECOESAHLRMI

1.17　(a) 证明在置换密码中,置换 π 是对合密钥,当且仅当对任意的 $i, j \in \{1, \cdots, m\}$,若 $\pi(i) = j$,则必有 $\pi(j) = i$。

(b) 在置换密码中分别令 $m = 2, 3, 4, 5, 6$,试求出其对合密钥。

1.18　考虑如下定义在 $\mathbb{Z}_2$ 上的线性递归序列:

$$z_{i+4} = (z_i + z_{i+1} + z_{i+2} + z_{i+3}) \bmod 2$$

$i \geqslant 0$。对其 16 种可能的初始向量 $(z_0, z_1, z_2, z_3) \in (\mathbb{Z}_2)^4$,分别求其生成密钥流的周期。

1.19　令递归关系式为:

$$z_{i+4} = (z_i + z_{i+3}) \bmod 2$$

$i \geqslant 0$。重新完成练习 1.18。

1.20　假设我们使用下列方法产生一个同步流密码。令 $\boldsymbol{K} \in \mathcal{K}$ 为密钥,$\mathcal{L}$ 为密钥流字母表,Σ 表示所有状态的有限集。首先,初始状态 $\sigma_0 \in \Sigma$ 由 $\boldsymbol{K}$ 按某种规则产生。对任意的 $i \geqslant 1$,状态 σ_i 由状态 σ_{i-1} 通过如下规则决定:

$$\sigma_i = f(\sigma_{i-1}, K)$$

这里 $f: \Sigma \times \mathcal{K} \to \Sigma$。另外,对任意的 $i \geqslant 1$,密钥流元素 z_i 按如下规则计算:

$$z_i = g(\sigma_i, K)$$

这里 $g: \Sigma \times \mathcal{K} \to \mathcal{L}$。证明使用这种方法产生的密钥流的周期至多为 $|\Sigma|$。

1.21 以下给出的四段密文,第一个由代换密码加密而成,第二个由维吉尼亚密码加密而成,第三个由仿射密码加密而成,最后一个不知其具体的密码体制,试从密文确定其明文。要求给出清晰的分析过程,包括统计分析和进行的计算。

(a) 代换密码:

EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOXYSIPJCK
QPKUGKMGOLICGINCGACKSNISACYKZSCKXECJCKSHYSXCG
OIDPKZCNKSHICGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZU
GFZCCNDGYYSFUSZCNXEOJNCGYEOWEUPXEZGACGNFGLKNS
ACIGOIYCKXCJUCIUZCFZCCNDGYYSFEUEKUZCSOCFZCCNC
IACZEJNCSHFZEJZEGMXCYHCJUMGKUCY

提示: F 解密到 w。

(b) 维吉尼亚密码:

KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFDETDGILTXRGUD
DKOTFMBPVGEGLTGCKQRACQCWDNAWCRXIZAKFTLEWRPTYC
QKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYHJVDAHCTRL
SVSKCGCZQQDZXGSFRLSWCWSJTBHAFSIASPRJAHKJRJUMV
GKMITZHFPDISPZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFS
PEZQNRWXCVYCGAONWDDKACKAWBBIKFTIOVKCGGHJVLNHI
FFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQIY
CWHJVLNHIQIBTKHJVNPIST

(c) 仿射密码:

KQEREJEBCPPCJCRKIEACUZBKRVPKRBCIBQCARBJCVFCUP
KRIOFKPACUZQEPBKRXPEIIEABDKPBCPFCDCCAFIEABDKP
BCPFEQPKAZBKRHAIBKAPCCIBURCCDKDCCJCIDFUIXPAFF
ERBICZDFKABICBBENEFCUPJCVKABPCYDCCDPKBCOCPERK
IVKSCPICBRKIJPKABI

(d) 未知密码:

BNVSNSIHQCEELSSKKYERIFJKXUMBGYKAMQLJTYAVFBKVT
DVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUUALRWXM
MASAZLGLEDFJBZAVVPXWICGJXASCBYEHOSNMULKCEAHTQ
OKMFLEBKFXLRRFDTZXCIWBJSICBGAWDVYDHAVFJXZIBKC
GJIWEAHTTOEWTUHKRQVVRGZBXYIREMMASCSPBHLHJMBLR
FFJELHWEYLWISTFVVYEJCMHYUYRUFSFMGESIGRLWALSWM
NUHSIMYYITCCQPZSICEHBCCMZFEGVJYOCDEMMPGHVAAUM
ELCMOEHVLTIPSUYILVGFLMVWDVYDBTHFRAYISYSGKVSUU
HYHGGCKTMBLRX

1.22　(a) 假设 $p_1,\cdots,p_n$ 和 $q_1,\cdots,q_n$ 均为概率分布且有 $p_1 \geqslant \cdots \geqslant p_n$。令 $q'_1,\cdots,q'_n$ 为 $q_1,\cdots,q_n$ 的任意置换，证明值

$$\sum_{i=1}^{n} p_i q'_i$$

当 $q'_1 \geqslant \cdots \geqslant q'_n$ 时取得最大值。

(b) 试用此结论解释式(1.1)。

1.23　假设明文

breathtaking

使用 Hill 密码被加密为

RUPOTENTOIFV

试分析出加密密钥矩阵(矩阵维数 m 未知)。

1.24　以下仿射密码体制由希尔密码修改得来：设 m 为一正整数，定义 $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$，密钥由 $(\boldsymbol{L}, b)$ 组成，这里 $\boldsymbol{L}$ 为定义在 $\mathbb{Z}_{26}$ 上的可逆矩阵，$b \in (\mathbb{Z}_{26})^m$。对任意的 $x = (x_1,\cdots,x_m) \in \mathcal{P}$ 和 $\boldsymbol{K} = (\boldsymbol{L}, b) \in \mathcal{K}$，定义加密变换 $y = e_K(x) = (y_1,\cdots,y_m)$ 为 $y = x\boldsymbol{L} + b$。因此，若 $\boldsymbol{L} = (l_{i,j})$，$b = (b_1,\cdots,b_m)$，则

$$(y_1,\cdots,y_m) = (x_1,\cdots,x_m)\begin{pmatrix} l_{1,1} & l_{1,2} & \cdots & l_{1,m} \\ l_{2,1} & l_{2,2} & \cdots & l_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ l_{m,1} & l_{m,2} & \cdots & l_{m,m} \end{pmatrix} + (b_1,\cdots,b_m)$$

假设敌手 Oscar 知道如下明文

adisplayedequation

被加密成如下密文

DSRMSIOPLXLJBZULLM

并且 Oscar 知道 $m = 3$。试求出密钥，要求给出详细过程。

1.25　这里描述一下如何利用惟密文攻击来分析希尔密码。假定我们已知 $m = 2$。把密文分成两字母组的块。每一块是由明文块利用未知的加密矩阵加密所得。找出出现次数最多的密文块，假定它是在表 1.1 后面所列出的一个常见两字母组(例如，TH 或者 ST)。对于每一个这样的猜测，使用在已知明文攻击中的计算过程，直到找出正确的加密矩阵为止。利用这个方法来解密如下的密文例子：

LMQETXYEAGTXCTUIEWNCTXLZEWUAISPZYVAPEWLMGQWYA
XFTCJMSQCADAGTXLMDXNXSNPJQSYVAPRIQSMHNOCVAXFV

1.26　下面给出一个特殊的置换密码。设 m,n 为正整数，将明文写成一个 $m \times n$ 矩阵的形式，然后依次取矩阵的各列构成密文。例如，设 $m = 4, n = 3$，加密明文“cryptography”：

cryp

togr

aphy

对应的密文应该为“CTAROPYGHPRY”。

(a) 在已知 m 和 n 的情形下,Bob 如何来解密密文。

(b) 试解密通过上述方法获得的下列密文:

MYAMRARUYIQTENCTORAHROYWDSOYEOUARRGDERNOGW

1.27 本练习的目的是为了证明 1.2.5 小节中的 $m \times m$ 的相关系数矩阵的可逆性。此命题等价于在$\mathbb{Z}_2$ 上矩阵的行向量线性无关。

同前所述,假设递归关系具有如下形式:

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2$$

初始向量为$(z_1, \cdots, z_m)$。对任意的 $i \geqslant 1$,定义

$$\boldsymbol{v}_i = (z_i, \cdots, z_{i+m-1})$$

注意,系数矩阵的各行就是向量 $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m$,因此我们的目标就是这 m 个向量是线性无关的。

试证明如下结论:

(a) 对任意的 $i \geqslant 1$,

$$\boldsymbol{v}_{m+i} = \sum_{j=0}^{m-1} c_j \boldsymbol{v}_{i+j} \bmod 2$$

(b) 令 h 为使得在$\mathbb{Z}_2$ 上向量 $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_h$ 线性相关的最小的值,则

$$\boldsymbol{v}_h = \sum_{j=0}^{h-2} \alpha_j \boldsymbol{v}_{j+1} \bmod 2$$

α_j 不全为零,显然 $h \leqslant m+1$,因为在 m 维向量空间中任意 $m+1$ 个向量必线性相关。

(c) 证明,对任意的 $i \geqslant 1$,密钥流必满足:

$$z_{h-1+i} = \sum_{j=0}^{h-2} \alpha_j z_{j+i} \bmod 2$$

(d) 可发现,如果 $h \leqslant m$,那么满足线性递归关系式的密钥流的级数小于 m,这一点与它的级数为 m 相矛盾。故 $h = m+1$,从而矩阵必为可逆的。

1.28 利用穷举密钥搜索法解密下列从自动密钥密码中获得的密文:

MALVVMAFBHBUQPTSOXALTGVWWRG

1.29 下面给出一个由维吉尼亚密码改进的流密码。给定长度为 m 的密钥字$(K_1, \cdots, K_m)$通过规则 $z_i = K_i (1 \leqslant i \leqslant m)$, $z_{i+m} = (z_i + 1) \bmod 26 \ (i \geqslant 1)$ 来构造密钥流序列,

换句话说，每次我们使用的密钥字都使用字母后续者的模 26 来替代。例如，如果“SUMMER”是密钥字，我们首先用“SUMMER”来加密第一个六字母组，对下一个六字母组，使用“TVNNFS”，等等。

(a) 描述一下怎样利用重合指数法来确定首次用来加密的密钥字的长度，然后找到它。

(b) 通过分析下列密文来测试你的方法：

```
IYMYSILONRFNCQXQJEDSHBUIBCJUZBOLFQYSCHATPEQGQ
JEJNGNXZWHHGWFSUKULJQACZKKJOAAHGKEMTAFGMKVRDO
PXNEHEKZNKFSKIFRQVHHOVXINPHMRTJPYWQGJWPUUVKFP
OAWPMRKKQZWLQDYAZDRMLPBJKJOBWIWPSEPVVQMBCRYVC
RUZAAOUMBCHDAGDIEMSZFZHALIGKEMJJJPCIWKRMLMPIN
AYOFIREAOLDTHITDVRMSE
```

1.30　下面再来给出一个流密码，它融入了二战中德国使用的“Enigma”密码体制的一种设计思想。假设 π 是一个定义在 $\mathbb{Z}_{26}$ 上的置换，密钥 $\boldsymbol{K} \in \mathbb{Z}_{26}$。对任意的 $i \geqslant 1$，密钥流元素 $z_i \in \mathbb{Z}_{26}$ 按 $z_i = (K + i - 1) \bmod 26$ 产生。加密解密使用置换 π 和对应的逆置换 π^{-1}：

$$e_K(x) = \pi(x) + z \bmod 26$$

$$d_K(y) = \pi^{-1}(y - z \bmod 26)$$

这里 $z \in \mathbb{Z}_{26}$。

假设 π 为定义在 $\mathbb{Z}_{26}$ 上的如下置换：

x	0	1	2	3	4	5	6	7	8	9	10	11	12
$\pi(x)$	23	13	24	0	7	15	14	6	25	16	22	1	19

x	13	14	15	16	17	18	19	20	21	22	23	24	25
$\pi(x)$	18	5	11	17	2	21	12	20	4	10	9	3	8

试用穷尽密钥搜索法破译使用该密码体制加密的如下密文：

```
WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDO
KLXRMCBKGRCCURR
```

第 2 章　Shannon 理论

2.1　引言

1949 年，Claude Shannon 在《Bell Systems Technical Journal》上发表了题为“Communication Theory of Secrecy Systems”的论文。这篇论文对密码学的研究产生了巨大的影响。在本章中，我们讨论了若干 Shannon 的思想。首先让我们看看几个评价密码体制安全性的不同途径，下面定义了几个有用的准则。

计算安全性(computational security)

这种度量涉及到攻破密码体制所做的计算上的努力。如果使用最好的算法攻破一个密码体制需要至少 N 次操作，这里的 N 是一个特定的非常大的数字，我们可以定义这个密码体制是计算安全的。问题是没有一个已知的实际的密码体制在这个定义下可以被证明是安全的。实际中，人们经常通过几种特定的攻击类型来研究计算上的安全性，例如穷尽密钥搜索攻击。当然，对一种类型的攻击是安全的，并不表示对其他类型的攻击是安全的。

可证明安全性(provable security)

另外一种途径是将密码体制的安全性归结为某个经过深入研究的数学难题。例如，可以证明这样一类命题：如果给定的整数是不可分解的，那么给定的密码体制是不可破解的。我们称这种类型的密码体制是可证明安全的。但是必须注意，这种途径只是说明了安全和另一个问题是相关的，并没有完全证明是安全的。这和证明一个问题是 NP 完全的(NP-complete)有些类似：证明给定的问题和任何其他的 NP 完全问题的难度是一样的，但是并没有完全证明这个问题的计算难度。

无条件安全性(unconditional security)

这种度量考虑的是对攻击者 Oscar 的计算量没有限制的时候的安全性。即使提供了无穷的计算资源，也是无法被攻破的，我们定义这种密码体制是无条件安全的。

在讨论密码体制的安全性的时候，我们同时规定了正在考虑的攻击类型。例如，在第 1 章中，我们说移位密码、代换密码和维吉尼亚密码对惟密文攻击都不是计算上安全的(如果给了足够的密文)。

我们将在 2.3 节研究一个对惟密文攻击是无条件安全的密码体制。这个理论从数学上证明了：如果给出的密文足够少，某些密码体制是安全的。例如，可以证明，如果只有单个的明文用给定的密钥加密，移位密码和代换密码都将是无条件安全的。类似地，使用密钥字长度为 m 的维吉尼亚密码是无条件安全的，如果密钥用于加密一个单位的明文（由 m 个字母组成)。

2.2 概率论基础

密码体制的无条件安全性当然不能用计算复杂性的观点来研究,因为我们允许计算时间是无限的。研究无条件安全的合适框架是概率论。我们只需要概率论的基本知识,现在复习一下其主要的内容。首先定义随机变量的概念。

定义 2.1 一个离散的随机变量,比方说 **X**,由有限集合 X 和定义在 X 上的概率分布组成。我们用 $\mathbf{Pr}[\mathbf{X}=x]$表示随机变量 **X** 取 x 时的概率。如果随机变量是固定的,我们有时缩写成 $\mathbf{Pr}[\mathbf{X}]$。对任意的 $x\in X$,有 $0\leqslant \Pr[x]\leqslant 1$,并且

$$\sum_{x\in X}\mathbf{Pr}[x]=1$$

举一个例子,我们可以把抛硬币看成定义在集合 $\{heads, tails\}$ 上的随机变量。相关的概率分布可以是 $\mathbf{Pr}[heads]=\mathbf{Pr}[tails]=1/2$。

假设我们有定义在集合 X 上的随机变量 **X**,且 $E\subseteq X$。**X** 在子集 E 上取值的概率计算如下:

$$\mathbf{Pr}[x\in E]=\sum_{x\in E}\mathbf{Pr}[x] \tag{2.1}$$

子集 E 通常称为事件。

例 2.1 假设随机抛一对骰子。我们可以用一个随机变量 **Z** 来刻画这个过程。这个随机变量定义在集合

$$Z=\{1,2,3,4,5,6\}\times\{1,2,3,4,5,6\}$$

上,并且对任意的 $(i,j)\in Z$,$\mathbf{Pr}[(i,j)]=1/36$。考虑两个骰子的和。每一个可能的和都定义了一个事件,这些事件的概率可以通过公式(2.1)计算出来。例如,要计算和为 4 的概率,相应的事件是:

$$S_4=\{(1,3),(2,2),(3,1)\}$$

因此,$\mathbf{Pr}[S_4]=3/36=1/12$。

所有和的概率都可以用类似的公式计算。如果用 S_j 表示和为 j 的事件,可以得到如下结果:$\mathbf{Pr}[S_2]=\mathbf{Pr}[S_{12}]=1/36$,$\mathbf{Pr}[S_3]=\mathbf{Pr}[S_{11}]=1/18$,$\mathbf{Pr}[S_4]=\mathbf{Pr}[S_{10}]=1/12$,$\mathbf{Pr}[S_5]=\mathbf{Pr}[S_9]=1/9$,$\mathbf{Pr}[S_6]=\mathbf{Pr}[S_8]=5/36$,$\mathbf{Pr}[S_7]=1/6$。

既然事件 $S_2,\cdots,S_{12}$ 是集合 S 的一个划分,我们可以把这对骰子和的值当成一个随机变量,概率分布是例 2.1 计算的结果。

下面考虑联合概率和条件概率的概念。

定义 2.2　假设 **X** 和 **Y** 是分别定义在有限集合 X 和 Y 上的随机变量。联合概率 $\mathbf{Pr}[x, y]$是 **X** 取 x 并且 **Y** 取 y 的概率。条件概率 $\mathbf{Pr}[x \mid y]$表示 **Y** 取 y 时 **X** 取 x 的概率。如果对任意的 $x \in X$ 和 $y \in Y$,有 $\mathbf{Pr}[x, y] = \mathbf{Pr}[x]\mathbf{Pr}[y]$,则称随机变量 **X** 和 **Y** 是统计独立的。

联合概率和条件概率可以通过下面的公式联系起来:

$$\mathbf{Pr}[x, y] = \mathbf{Pr}[x \mid y]\mathbf{Pr}[y]$$

交换 x 和 y,我们有:

$$\mathbf{Pr}[x, y] = \mathbf{Pr}[y \mid x]\mathbf{Pr}[x]$$

从这两个表达式,立即得到下面的结果,也就是 Bayes 定理。

定理 2.1(Bayes 定理)　如果 $\mathbf{Pr}[y] > 0$,那么:

$$\mathbf{Pr}[x \mid y] = \frac{\mathbf{Pr}[x]\mathbf{Pr}[y \mid x]}{\mathbf{Pr}[y]}$$

推论 2.2　**X** 和 **Y** 是统计独立的随机变量,当且仅当对所有的 $x \in X$ 和 $y \in Y$,有 $\mathbf{Pr}[x \mid y] = \mathbf{Pr}[x]$。

例 2.2　假设我们随机地抛一对骰子。同例 2.1,考虑两个骰子的和,得到一个定义在集合 $X = \{2, \cdots, 12\}$ 上的随机变量 **X**。进一步,假设随机变量 **Y**,当骰子的值相等时,取 D;不等时,取 N。我们有 $\mathbf{Pr}[D] = 1/6, \mathbf{Pr}[E] = 5/6$。

可以直接算出这些随机变量的联合概率和条件概率。例如,$\mathbf{Pr}[D \mid 4] = 1/3, \mathbf{Pr}[4 \mid D] = 1/6$,所以

$$\mathbf{Pr}[D \mid 4]\mathbf{Pr}[4] = \mathbf{Pr}[D]\mathbf{Pr}[4 \mid D]$$

符合 Bayes 定理。

2.3　完善保密性

这一节中,假设$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$是一个特定的密码体制,密钥 $K \in \mathcal{K}$ 只用于一次加密。假设明文空间$\mathcal{P}$存在一个概率分布。因此明文元素定义了一个随机变量,用 **x** 表示。$\mathbf{Pr}[\mathbf{x} = x]$表示明文 x 发生的先验概率。还假设(Alice 和 Bob)以固定的概率分布选取密钥(通常密钥是随机选取的,因此所有的密钥都是等概率的,但是这里不需要这样)。所以密钥也定义了一个

随机变量,用 **K** 表示。$\mathbf{Pr}[\mathbf{K}=K]$表示密钥 K 发生的概率。回忆一下,密钥是在 Alice 知道明文之前选取的。因此,我们可以合理地假设密钥和明文是统计独立的随机变量。

$\mathcal{P}$和$\mathcal{K}$的概率分布导出了$\mathcal{C}$的概率分布。因此,同样可以把密文元素看成随机变量,用 **y** 表示。不难计算出密文 y 的概率 $\mathbf{Pr}[\mathbf{y}=y]$。对于密钥 $K\in\mathcal{K}$,定义:

$$C(K)=\{e_K(x):x\in\mathcal{P}\}$$

也就是说 $C(K)$代表密钥是 K 时所有可能的密文。对于任意的 $y\in\mathcal{C}$,我们有:

$$\mathbf{Pr}[\mathbf{y}=y]=\sum_{\{K:y\in C(K)\}}\mathbf{Pr}[\mathbf{K}=K]\mathbf{Pr}[\mathbf{x}=d_K(y)]$$

同样可观察到,对于任意的 $y\in\mathcal{C}$和 $x\in\mathcal{P}$,可如下计算条件概率 $\mathbf{Pr}[\mathbf{y}=y\mid\mathbf{x}=x]$(也就是给定明文 x,密文 y 的概率):

$$\mathbf{Pr}[\mathbf{y}=y\mid\mathbf{x}=x]=\sum_{\{K:x=d_K(y)\}}\mathbf{Pr}[\mathbf{K}=K]$$

现在可以用 Bayes 定理计算条件概率 $\mathbf{Pr}[\mathbf{x}=x\mid\mathbf{y}=y]$(也就是给定密文 y,明文 x 的概率)。计算如下:

$$\mathbf{Pr}[\mathbf{x}=x\mid\mathbf{y}=y]=\frac{\mathbf{Pr}[\mathbf{x}=x]\times\sum_{\{K:x=d_K(y)\}}\mathbf{Pr}[\mathbf{K}=K]}{\sum_{\{K:y\in C(K)\}}\mathbf{Pr}[\mathbf{K}=K]\mathbf{Pr}[\mathbf{x}=d_K(y)]}$$

可见,只要知道了概率分布任何人都可以完成这些计算。

我们给出一个例子,看看具体如何计算这些概率。

例2.3 假设 $\mathcal{P}=\{a,b\}$ 满足 $\mathbf{Pr}[a]=1/4$, $\mathbf{Pr}[b]=3/4$。设 $\mathcal{K}=\{K_1,K_2,K_3\}$, $\mathbf{Pr}[K_1]=1/2$, $\mathbf{Pr}[K_2]=\mathbf{Pr}[K_3]=1/4$。设 $\mathcal{C}=\{1,2,3,4\}$,加密函数定义为 $e_{K_1}(a)=1$, $e_{K_2}(b)=2$, $e_{K_2}(a)=2$, $e_{K_2}(b)=3$, $e_{K_3}(a)=3$, $e_{K_3}(b)=4$。这个密码体制可以用如下加密矩阵表示:

	a	b
K_1	1	2
K_2	2	3
K_3	3	4

计算 e 上的概率分布如下:

$$\mathbf{Pr}[1]=\frac{1}{8}$$

$$\mathbf{Pr}[2]=\frac{3}{8}+\frac{1}{16}=\frac{7}{16}$$

$$\mathbf{Pr}[3]=\frac{3}{16}+\frac{1}{16}=\frac{1}{4}$$

$$\mathbf{Pr}[4]=\frac{3}{16}$$

现在计算给定密文后，明文空间上的条件概率分布为：

$$\mathbf{Pr}[a|1]=1 \quad \mathbf{Pr}[b|1]=0$$

$$\mathbf{Pr}[a|2]=\frac{1}{7} \quad \mathbf{Pr}[b|2]=\frac{6}{7}$$

$$\mathbf{Pr}[a|3]=\frac{1}{4} \quad \mathbf{Pr}[b|3]=\frac{3}{4}$$

$$\mathbf{Pr}[a|4]=0 \quad \mathbf{Pr}[b|4]=1$$

我们现在来定义完善保密性的概念。通俗地讲，完善保密性就是说 Oscar 不能通过观察密文得到明文的任何信息。用概率分布的术语，精确的定义如下。

定义 2.3　一个密码体制具有完善保密性，即，如果对于任意的 $x\in\mathcal{P}$ 和 $y\in\mathcal{C}$，有 $\mathbf{Pr}[x|y]=\mathbf{Pr}[x]$。也就是说，给定密文 y，明文为 x 的后验概率等于明文 x 的先验概率。

在例 2.3 中，对于密文 $y=3$ 满足完善保密性的定义，但是对于其他的密文不满足。

现在证明移位密码的完善保密性。从直觉上看这是很显然的。因为对于任意给定的密文 $y\in\mathbb{Z}_{26}$，任何 $x\in\mathbb{Z}_{26}$ 都可能是对应的明文。下面的定理用概率分布给出了正式的陈述和证明。

定理 2.3　假设移位密码的 26 个密钥都是以相同的概率 1/26 使用的，则对于任意的明文概率分布，移位密码具有完善保密性。

证明　这里 $\mathcal{P}=\mathcal{C}=\mathcal{K}=\mathbb{Z}_{26}$，对于 $0\leqslant K\leqslant 25$，加密函数 e_K 定义为 $e_K(x)=(x+K) \bmod 26$ $(x\in\mathbb{Z}_{26})$。首先计算 $\mathcal{C}$ 上的概率分布。假设 $y\in\mathbb{Z}_{26}$，则

$$\begin{aligned}\mathbf{Pr}[\mathbf{y}=y] &= \sum_{K\in\mathbb{Z}_{26}}\mathbf{Pr}[\mathbf{K}=K]\mathbf{Pr}[\mathbf{x}=d_K(y)]\\ &= \sum_{K\in\mathbb{Z}_{26}}\frac{1}{26}\mathbf{Pr}[\mathbf{x}=y-K]\\ &= \frac{1}{26}\sum_{K\in\mathbb{Z}_{26}}\mathbf{Pr}[\mathbf{x}=y-K]\end{aligned}$$

现在固定 y，值 $(y-K) \bmod 26$ 构成 $\mathbb{Z}_{26}$ 的一个置换。因此有：

$$\sum_{K\in\mathbb{Z}_{26}}\mathbf{Pr}[\mathbf{x}=y-K]=\sum_{K\in\mathbb{Z}_{26}}\mathbf{Pr}[\mathbf{x}=x]=1$$

得到对于任意的 $y\in\mathbb{Z}_{26}$，

$$\mathbf{Pr}[y]=\frac{1}{26}$$

接下来,对于任意的 x, y,我们有:

$$\begin{aligned}\mathbf{Pr}[y \mid x] &= \mathbf{Pr}[\mathbf{K} = (y - x) \bmod 26] \\ &= \frac{1}{26}\end{aligned}$$

(这是因为对于任意的 x, y,满足 $e_K(x) = y$ 的惟一的密钥 $K = (y - x) \bmod 26$。)现在应用 Bayes 定理,很容易计算出:

$$\begin{aligned}\mathbf{Pr}[x \mid y] &= \frac{\mathbf{Pr}[x]\mathbf{Pr}[y \mid x]}{\mathbf{Pr}[y]} \\ &= \frac{\mathbf{Pr}[x]\frac{1}{26}}{\frac{1}{26}} \\ &= \mathbf{Pr}[x]\end{aligned}$$

所以这个密码体制是完善保密的。

因此,如果新的随机密钥用来加密每个明文字母,则移位密码是“不可攻破的”。

下面考察一般的完善保密性。首先,我们可以发现,使用 Bayes 定理,对于所有的 $x \in \mathcal{P}$ 和 $y \in \mathcal{C}$,$\mathbf{Pr}[x \mid y] = \mathbf{Pr}[x]$,等价于对于所有的 $x \in \mathcal{P}$ 和 $y \in \mathcal{C}$,$\mathbf{Pr}[y \mid x] = \mathbf{Pr}[y]$。我们可以合理地假设对于所有的 $y \in \mathcal{C}$,$\mathbf{Pr}[y] > 0$(如果 $\mathbf{Pr}[y] = 0$,则密文 y 从不会使用到,可以从 $\mathcal{C}$ 中去掉)。固定任意的 $x \in \mathcal{P}$。对于任意 $y \in \mathcal{C}$,我们有 $\mathbf{Pr}[y \mid x] = \mathbf{Pr}[y] > 0$,。因此,对于任意 $y \in \mathcal{C}$,一定至少存在一个密钥 K 满足 $e_K(x) = y$。这样有 $|\mathcal{K}| \geqslant |\mathcal{C}|$。在任意一个密码体制中,因为加密函数是单射的,一定有 $|\mathcal{C}| \geqslant |\mathcal{P}|$。当 $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ 时,我们有一个关于什么时候取得完善保密性的很好的性质。这个性质最早是由 Shannon 提出来的。

定理 2.4 假设密码体制 $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ 满足 $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$。这个密码体制是完善保密的,当且仅当每个密钥被使用的概率都是 $1/|\mathcal{K}|$,并且对于任意的 $x \in \mathcal{P}$ 和 $y \in \mathcal{C}$,存在惟一的密钥 K 使得 $e_K(x) = y$。

证明 假设这个密码体制是完善保密的。由上面可知,对于任意的 $x \in \mathcal{P}$ 和 $y \in \mathcal{C}$,一定至少存在一个密钥 K 满足 $e_K(x) = y$。因此有不等式:

$$\begin{aligned}|\mathcal{C}| &= |\{e_K(x) : K \in \mathcal{K}\}| \\ &\leqslant |\mathcal{K}|\end{aligned}$$

但是我们假设 $|\mathcal{C}| = |\mathcal{K}|$,因此一定有:

$$|\{e_K(x) : K \in \mathcal{K}\}| = |\mathcal{K}|$$

也就是说,不存在两个不同的密钥 K_1 和 K_2 使得 $e_{K_1}(x) = e_{K_2}(x) = y$。因此对于 $x \in \mathcal{P}$ 和 $y \in \mathcal{C}$,刚好存在一个密钥 K 使得 $e_K(x) = y$。

记 $n = |\mathcal{K}|$。设 $\mathcal{P} = \{x_i : 1 \leqslant i \leqslant n\}$ 并且固定一个密文 $y \in \mathcal{C}$。设密钥为 $K_1, K_2, \cdots, K_n$,并且

$e_{K_1}(x_i)=y, 1\leqslant i\leqslant n$。使用 Bayes 定理，我们有：

$$\mathbf{Pr}[x_i \mid y]=\frac{\mathbf{Pr}[y \mid x_i]\mathbf{Pr}[x_i]}{\mathbf{Pr}[y]}$$
$$=\frac{\mathbf{Pr}[\mathbf{K}=K_i]\mathbf{Pr}[x_i]}{\mathbf{Pr}[y]}$$

考虑完善保密的条件 $\mathbf{Pr}[x_i \mid y]=\mathbf{Pr}[x_i]$。从这里，我们有 $\mathbf{Pr}[K_i]=\mathbf{Pr}[y], 1\leqslant i\leqslant n$。也就是说，所有的密钥都是等概率使用的(概率为 $\mathbf{Pr}[y]$)。但是密钥的数目为$\mathcal{K}$，我们得到对任意的 $K\in\mathcal{K}, \mathbf{Pr}[K]=1/|\mathcal{K}|$。

相反地，如果两个假设的条件是成立的，类似于定理 2.3 的证明，我们很容易得到密码体制是完善保密的。证明留给读者自己完成。

一个著名的具有完善保密性的密码体制是“一次一密”(One-time Pad)密码体制。这个体制首先由 Gilbert Vernam 于 1917 年用于报文消息的自动加密和解密。有意思的是，“一次一密”很多年来被认为是不可破的，但是一直都没有一个数学的证明，直到 30 年后 Shonnon 提出了完善保密性的概念。“一次一密”的描述如下。

密码体制 2.1　一次一密

假设 $n\geqslant 1$ 是正整数，$\mathcal{P}=\mathcal{C}=\mathcal{K}=(\mathbb{Z}_2)^n$。对于 $K\in(\mathbb{Z}_2)^n$，定义 $e_K(x)$为 K 和 x 的模 2 的向量和(或者说是两个相关比特串的异或)。因此，如果 $x=(x_1,\cdots,x_n)$并且 $K=(K_1,\cdots,K_n)$，则：

$$e_K(x)=(x_1+K_1,\cdots,x_n+K_n) \bmod 2$$

解密和加密是一样的。如果 $y=(y_1,\cdots,y_n)$，则

$$d_K(y)=(y_1+K_1,\cdots,y_n+K_n) \bmod 2$$

由定理 2.4，容易看出“一次一密”提供了完善保密性。这个密码体制的加密和解密也很容易，有一定的吸引力。Vernam 对此还申请了专利，希望有广泛的商业用途。不幸的是，“一次一密”存在一个较大的不利因素。$|\mathcal{K}|\geqslant|\mathcal{P}|$ 意味着秘密使用的密钥数量必须至少和明文的数量一样多。例如，在“一次一密”的密码体制中，我们要求用 n 比特的密钥加密 n 比特的明文。相同的密钥可以用于加密不同的消息将不是一个重要的问题，但是密码体制的无条件安全性是基于每个密钥仅用一次的事实。

例如，“一次一密”对已知明文攻击将是脆弱的，因为 K 可以通过异或比特串 x 和 $e_K(x)$得到。因此，新生成的密钥需要为将要发送的明文在安全的通道上传输。这就带来了一个严峻的密钥管理问题，因此限制了“一次一密”在商业上的应用。但是“一次一密”在要求无条件安全的军事和外交环境中有着很重要的应用。

在密码学的发展历史中，人们试图设计出密钥可以加密相对长的明文(也就是一个密钥可

以加密许多消息),并且仍然可以保持一定的计算安全性。一个这样的例子就是数据加密标准(Data Encryption Standard),我们将在下一章讨论。

2.4 熵

上一节,我们讨论了完善保密性的概念。我们将注意力集中在密钥只能用于一次加密的特殊情况下。现在看看用一个密钥加密多个消息会发生什么,并且还要看看给密码分析员足够的时间,进行一次成功的惟密文攻击有多大的可能性。

研究这个问题的基本工具是熵(Entropy)。这个概念来自于 Shannon 于 1948 年创建的信息论。熵可以看做是对信息或不确定性的数学度量,是通过一个概率分布的函数来计算的。

假设离散的随机变量 **X** 根据特定的概率分布从有限集合 X 中取值。我们可以从按照这个概率分布的实验结果中得到什么信息呢? 或者说,在实验发生之前,结果的不确定性是什么呢? 这个性质称为 **X** 的熵,用 $H(\mathbf{X})$表示。

这些描述看起来还相当抽象,让我们看看更具体的例子。假设随机变量 **X** 代表掷硬币。如前面提到的,相关的概率分布是 $\mathbf{Pr}[\mathit{heads}] = \mathbf{Pr}[\mathit{tails}] = 1/2$。即然我们可以将"*heads*"编码为 1,将"*tails*"编码成 0,可以合理地说一次掷硬币的信息或者熵是 1 比特。用同样的方式,n 次独立的掷硬币的熵是 n 比特,因为 n 次投掷可以用长为 n 比特的比特串进行编码。

再看一个稍微复杂一点的例子。假设有一个随机变量 **X**,取三种可能的值 x_1, x_2, x_3,概率分别为 1/2,1/4,1/4。对这三种结果最有效的编码是将 x_1 表示成 0,x_2 表示成 10,x_3 表示成 11。那么编码的平均比特长度是

$$\frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = \frac{3}{2}$$

上面的例子说明以概率 2^{-n} 发生的事件可以编码成长度为 n 的比特串。更一般地,我们可以想像以概率 p 发生的结果可以编码成长大约为 $-\log_2 p$ 的比特串。对于任意的概率分布为 $p_1, \cdots, p_n$ 的随机变量 **X**,将 $-\log_2 p_i$ 的加权平均值作为对信息的度量。正式的定义如下。

定义 2.4 假设随机变量 **X** 在有限集合 X 上取值,则随机变量 **X** 的熵定义为:

$$H(\mathbf{X}) = -\sum_{x \in X} \mathbf{Pr}[x] \log_2 \mathbf{Pr}[x]$$

注 可发现在 $y = 0$ 时 $\log_2 y$ 没有定义。因此熵有时定义为在所有非零的概率上的相关和。但是因为 $\lim_{y \to 0} y \log_2 y = 0$,实际上对于某些 x 可以令 $\mathbf{Pr}[x] = 0$。

同样地,我们注意到选择 2 作为指数的底也是任意的,选取其他的底仅仅改变了熵的常数因子。

如果 $|X| = n$ 并且对于所有的 $x \in X$, $\mathbf{Pr}[x] = 1/n$, 那么 $H(\mathbf{X}) = \log_2 n$。同样, 容易知道对于任意的随机变量 $\mathbf{X}$, $H(\mathbf{X}) \geqslant 0$。$H(\mathbf{X}) = 0$ 当且仅当对于某一个 $x_0 \in X$, $\mathbf{Pr}[x_0] = 1$, 对于所有的 $x \neq x_0$, $\mathbf{Pr}[x] = 0$。

让我们看看密码体制不同部分的熵。我们可以把密钥看成在 $\mathcal{K}$ 上取值的随机变量 $\mathbf{K}$, 因此可以计算熵 $H(\mathbf{K})$。同样地, 可以分别计算同明文和密文相关的熵 $H(\mathbf{P})$ 和 $H(\mathbf{C})$。

作为示例, 我们计算例 2.3 的密码体制的熵。

例 2.3(续)　计算如下:

$$\begin{aligned} H(\mathbf{P}) &= -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} \\ &= -\frac{1}{4}(-2) - \frac{3}{4}(\log_2 3 - 2) \\ &= 2 - \frac{3}{4}(\log_2 3) \\ &\approx 0.81 \end{aligned}$$

同样, 可得出 $H(\mathbf{K}) = 1.5$, $H(\mathbf{C}) \approx 1.85$。

2.4.1　Huffman 编码

这一小节, 我们简要讨论熵和 Huffman 编码之间的联系。这一小节与熵在密码学上的应用没有什么关系, 可以跳过去。但是本小节有助于理解熵的概念。

我们是通过根据特定的概率分布对随机事件进行编码来介绍熵的。首先让这个想法更精确些。和以前一样, $\mathbf{X}$ 是一个在有限集合 X 上取值的随机变量, p 是相关的概率分布。

对 $\mathbf{X}$ 的编码是任何映射

$$f: X \to \{0,1\}^*$$

其中, $\{0,1\}^*$ 代表所有的 0 和 1 的有限串的集合。对一列有限的事件 $x_1 \cdots x_n$, $x_i \in X$, 我们用一个明显的方式扩展编码 f, 定义如下:

$$f(x_1 \cdots x_n) = f(x_1) \| \cdots \| f(x_n)$$

其中, $\|$ 表示串的连接。这样, 我们可以将 f 看成映射

$$f: X^* \to \{0,1\}^*$$

现在假设串 $x_1 \cdots x_n$ 是由无记忆源生成的, 串中的每一个 x_i 都按照 X 上特定的概率分布发生。也就是说, 任意串 $x_1 \cdots x_n$ 的概率可以计算如下:

$$\mathbf{Pr}[x_1 \cdots x_n] = \mathbf{Pr}[x_1] \times \cdots \times \mathbf{Pr}[x_n]$$

注　因为源是无记忆的, 串 $x_1 \cdots x_n$ 没有必要由不同的值组成。举一个简单的例子, 考虑 n 次公平地掷硬币: 如果我们用 1 表示正面, 用 0 表示反面, 每一个长为 n 的二进制串对应于 n 次掷硬币的序列。

现在,假设我们要使用映射 f 对串进行编码,以一种明确的方式编码是很重要的。因此要求 f 是单射的。

例 2.4 假设 $X=\{a,b,c,d\}$,考虑以下三种编码:

$$f(a)=1 \quad f(b)=10 \quad f(c)=100 \quad f(d)=1000$$
$$g(a)=0 \quad g(b)=10 \quad g(c)=110 \quad g(d)=111$$
$$h(a)=0 \quad h(b)=01 \quad h(c)=10 \quad h(d)=11$$

可以看出 f 和 g 是单射,但 h 不是。任何使用 f 的编码都可以从字母编码的尾部开始向后解码:每遇到一个 1,就预示一个当前字母的开始。

使用 g 的编码可以从头开始依次进行解码。一看到子串是 a,b,c 或者 d 的编码,就马上解码,并且去掉这个子串。例如,给定串 10101110,我们将 10 解码成 b,然后 10 解码成 b,接下来 111 解码成 d,最后 0 解码成 a。所以解码得到的字符串是 $bbda$。

h 不是单射,一个例子就可以说明:

$$h(ac)=h(ba)=010$$

考虑到解码的简单性,我们更喜欢编码 g 而不是 f。这是因为 g 的解码可以从开头到结尾依次完成,不需要记忆。这种允许简单的按次序的解码性质称为无前缀特性。(我们说 g 的解码是无前缀的,如果不存在两个元素 $x,y\in X$ 和一个串 $z\in\{0,1\}^*$,满足 $g(x)=g(y)\|z$。)

讨论到目前为止还没有涉及熵。不用奇怪,熵和解码的效率有关。我们将同以前一样度量 f 的解码效率:**X** 的元素的编码的加权平均长度(用 $l(f)$ 表示)。因此我们有如下定义:

$$l(f)=\sum_{x\in X}\mathbf{Pr}[x]\,|f(x)|$$

其中 $|y|$ 表示串 y 的长度。

现在,我们的基本问题是找到一个单射的编码 l,使得 $l(f)$ 最小。有名的 Huffman 算法达到了这个目标。另外,由 Huffman 算法得到的 f 解码是无前缀的,并且有:

$$H(\mathbf{X})\leqslant l(f)<H(\mathbf{X})+1$$

因此,**X** 的熵值是对最佳单射编码的平均长度的精确估计。

我们不证明上述结果,但是会给出 Huffman 算法的简短的非正式的描述。Huffman 算法从集合 X 上的概率分布开始,每个元素的码字最初是空的。在每一个步骤中,两个概率最小的元素组合成一个元素,并以这两个概率的和作为新元素的概率。概率最小的元素赋值为"0",另一个元素赋值为"1"。当只剩下一个元素时,每个 $x\in X$ 的编码可以通过从最后一个元素到最初的元素 x"回溯"记录下得到的 01 序列而构造出来。

举个例子很容易说明这个算法。

例 2.5 假设 $X=\{a,b,c,d,e\}$ 有如下概率分布:$\mathbf{Pr}[a]=0.05$,$\mathbf{Pr}[b]=0.10$,$\mathbf{Pr}[c]=0.12$,$\mathbf{Pr}[d]=0.13$,$\mathbf{Pr}[e]=0.60$。Huffman 算法按如下表格进行:

a	b	c	d	e
0.05	0.10	0.12	0.13	0.60
0	1			
0.15		0.12	0.13	0.60
		0	1	
0.15		0.25		0.60
0		1		
0.40				0.60
0				1
1.0				

得到如下编码：

x	$f(x)$
a	000
b	001
c	010
d	011
e	1

因此，编码的平均程度为：

$$\begin{aligned} l(f) &= 0.05 \times 3 + 0.10 \times 3 + 0.12 \times 3 + 0.13 \times 3 + 0.60 \times 1 \\ &= 1.8 \end{aligned}$$

和熵比较：

$$\begin{aligned} H(\mathbf{X}) &= 0.2161 + 0.3322 + 0.3671 + 0.3842 + 0.4422 \\ &= 1.7402 \end{aligned}$$

可以看出，编码的平均长度和熵十分接近。

2.5　熵的性质

在这一节中，我们证明一些和熵有关的基本结论。首先，我们描述一个基本结论，称做 Jensen 不等式。这个不等式对我们很有用。Jensen 不等式涉及凸函数，下面给出其定义。

定义 2.5　区间 I 上的实值函数 f 是凸函数，如果对任意的 $x, y \in I$ 满足

$$f\left(\frac{x+y}{2}\right) \geqslant \frac{f(x)+f(y)}{2}$$

f 是区间 I 上严格的凸函数，如果对任意的 $x, y \in I, x \neq y$ 满足

$$f\left(\frac{x+y}{2}\right) > \frac{f(x)+f(y)}{2}$$

下面是 Jensen 不等式,我们不给出证明。

定理 2.5 (Jensen 不等式) 假设 f 是区间 I 上连续的严格的凸函数,

$$\sum_{i=1}^{n} a_i = 1$$

其中 $a_i > 0, 1 \leqslant i \leqslant n$。那么,

$$\sum_{i=1}^{n} a_i f(x_i) \leqslant f\left(\sum_{i=1}^{n} a_i x_i\right)$$

其中 $x_i \in I, 1 \leqslant i \leqslant n$。当且仅当 $x_1 = \cdots = x_n$,等号成立。

我们现在继续给出几个熵的结论。下面的定理利用了函数 $\log_2 x$ 在区间 $(0, \infty)$ 上是严格凸的这一事实(事实上,对数函数的二阶导数在 $(0, \infty)$ 上是负的,很容易就得到这个结论)。

定理 2.6 假设 $\mathbf{X}$ 是一个随机变量,概率分布为 $p_1, p_2, \cdots, p_n$,其中 $p_i > 0, 1 \leqslant i \leqslant n$。那么 $H(\mathbf{X}) \leqslant \log_2 n$,当且仅当 $p_i = 1/n, 1 \leqslant i \leqslant n$ 时等号成立。

证明 应用 Jensen 不等式,我们有如下结果:

$$\begin{aligned} H(\mathbf{X}) &= -\sum_{i=1}^{n} p_i \log_2 p_i \\ &= \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \\ &\leqslant \log_2 \sum_{i=1}^{n} \left(p_i \times \frac{1}{p_i}\right) \\ &= \log_2 n \end{aligned}$$

当且仅当 $p_i = 1/n, 1 \leqslant i \leqslant n$ 时,等号成立。

定理 2.7 $H(\mathbf{X}, \mathbf{Y}) \leqslant H(\mathbf{X}) + H(\mathbf{Y})$,当且仅当 $\mathbf{X}$ 和 $\mathbf{Y}$ 统计独立时等号成立。

证明 假设 $\mathbf{X}$ 取值 $x_i, 1 \leqslant i \leqslant m$,$\mathbf{Y}$ 取值 $y_j, 1 \leqslant j \leqslant n$。记 $p_i = \mathbf{Pr}[\mathbf{X} = x_i], 1 \leqslant i \leqslant m$;$q_j = \mathbf{Pr}[\mathbf{Y} = y_j], 1 \leqslant j \leqslant n$。记 $r_{ij} = \mathbf{Pr}[\mathbf{X} = x_i, \mathbf{Y} = y_j], 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$(这是联合概率分布)。

可发现:

$$p_i = \sum_{j=1}^{n} r_{ij}$$

$(1 \leqslant i \leqslant m)$,并且,

$$q_j = \sum_{i=1}^{n} r_{ij}$$

$(1 \leqslant j \leqslant n)$。计算如下:

$$H(\mathbf{X})+H(\mathbf{Y})=-\left(\sum_{i=1}^{m}p_i\log_2 p_i+\sum_{j=1}^{n}q_j\log_2 q_j\right)$$
$$=-\left(\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2 p_i+\sum_{j=1}^{n}\sum_{i=1}^{m}r_{ij}\log_2 q_j\right)$$
$$=-\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2 p_iq_j$$

另一方面，

$$H(\mathbf{X},\mathbf{Y})=-\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2 r_{ij}$$

于是有：

$$H(\mathbf{X},\mathbf{Y})-H(\mathbf{X})-H(\mathbf{Y})=\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2\frac{1}{r_{ij}}+\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2 p_iq_j$$
$$=\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2\frac{p_iq_j}{r_{ij}}$$
$$\leqslant\log_2\sum_{i=1}^{m}\sum_{j=1}^{n}p_iq_j$$
$$=\log_2 1$$
$$=0$$

（以上的计算使用了 Jensen 不等式，用到了 r_{ij} 是正实数并且和是 1 这一事实）。

我们同样可以得出什么时候取等号：此时存在常数 c 对于所有的 i,j 使得 $p_iq_j/r_{ij}=c$。利用

$$\sum_{j=1}^{n}\sum_{i=1}^{m}r_{ij}=\sum_{j=1}^{n}\sum_{i=1}^{m}p_iq_j=1$$

于是有 $c=1$。因此，当且仅当 $r_{ij}=p_iq_j$ 时等号成立，也就是当且仅当

$$\mathbf{Pr}[\mathbf{X}=x_i,\mathbf{Y}=y_j]=\mathbf{Pr}[\mathbf{X}=x_i]\mathbf{Pr}[\mathbf{Y}=y_j]$$

$1\leqslant i\leqslant m,1\leqslant j\leqslant m$。这说明 **X** 和 **Y** 是统计独立的。

接下来定义条件熵。

定义 2.6　假设 **X** 和 **Y** 是两个随机变量。对于 **Y** 的任何固定值 y，得到一个 **X** 上的（条件）概率分布；记相应的随机变量为 $\mathbf{X}|y$。显然，

$$H(\mathbf{X}|y)=-\sum_{x}\mathbf{Pr}[x\mid y]\log_2\mathbf{Pr}[x\mid y]$$

定义条件熵 $H(\mathbf{X}|\mathbf{Y})$ 为熵 $H(\mathbf{X}|y)$ 取遍所有的 y 的加权平均值：

$$H(\mathbf{X}|\mathbf{Y})=-\sum_{y}\sum_{x}\mathbf{Pr}[y]\mathbf{Pr}[x\mid y]\log_2\mathbf{Pr}[x\mid y]$$

条件熵度量了 **Y** 揭示的 **X** 的平均信息量。
接下来的两个结论很容易得到,我们将其证明留做练习。

定理 2.8 $H(\mathbf{X},\mathbf{Y}) = H(\mathbf{Y}) + H(\mathbf{X}|\mathbf{Y})$

推论 2.9 $H(\mathbf{X},\mathbf{Y}) \leqslant H(\mathbf{X})$,当且仅当 **X** 和 **Y** 统计独立时,等号成立。

2.6 伪密钥和惟一解距离

这一节中,我们来应用证明过的有关密码体制的熵的结果。首先,我们给出密码体制组成部分的熵的基本关系。条件熵 $H(\mathbf{K}|\mathbf{C})$称为密钥含糊度,度量了给定密文下密钥的不确定性。

定理 2.10 设$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$是一个密码体制,那么

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C})$$

证明 首先,有 $H(\mathbf{K},\mathbf{P},\mathbf{C}) = H(\mathbf{C}|\mathbf{K},\mathbf{P}) + H(\mathbf{K},\mathbf{P})$。因为,$y = e_K(x)$,所以密钥和明文惟一决定密文。这说明 $H(\mathbf{C}|\mathbf{K},\mathbf{P}) = 0$,因此,$H(\mathbf{K},\mathbf{P},\mathbf{C}) = H(\mathbf{K},\mathbf{P})$。但是 **K** 和 **P** 是统计独立的,所以 $H(\mathbf{K},\mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P})$。因此,

$$H(\mathbf{K},\mathbf{P},\mathbf{C}) = H(\mathbf{K},\mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P})$$

同样,既然密钥和密文惟一决定明文(即 $x = d_K(y)$),我们有 $H(\mathbf{P}|\mathbf{K},\mathbf{C}) = 0$,因此有 $H(\mathbf{K},\mathbf{P},\mathbf{C}) = H(\mathbf{K},\mathbf{C})$。

所以,

$$\begin{aligned} H(\mathbf{K}|\mathbf{C}) &= H(\mathbf{K},\mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K},\mathbf{P},\mathbf{C}) - H(\mathbf{C}) \\ &= H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}) \end{aligned}$$

让我们回到例 2.3 来解释这个结论。

例 2.3(续) 我们已经计算出 $H(\mathbf{P}) \approx 0.81$, $H(\mathbf{K}) \approx 1.5$, $H(\mathbf{C}) \approx 1.85$。由定理 2.10 知,$H(\mathbf{K}|\mathbf{C}) \approx 1.5 + 0.81 - 1.85 \approx 0.46$。可以通过条件熵的定义直接验证。首先我们需要计算概率 $\mathbf{Pr}[\mathbf{K} = K_i | \mathbf{y} = j]$, $1 \leqslant i \leqslant 3$, $1 \leqslant j \leqslant 4$。这可以使用 Bayes 定理完成,计算结果如下:

$$\mathbf{Pr}[K_1|1] = 1 \quad \mathbf{Pr}[K_2|1] = 0 \quad \mathbf{Pr}[K_3|1] = 0$$

$$\mathbf{Pr}[K_1|2] = \frac{6}{7} \quad \mathbf{Pr}[K_2|2] = \frac{1}{7} \quad \mathbf{Pr}[K_3|2] = 0$$

$$\mathbf{Pr}[K_1|3] = 0 \quad \mathbf{Pr}[K_2|3] = \frac{3}{4} \quad \mathbf{Pr}[K_3|3] = \frac{1}{4}$$

$$\mathbf{Pr}[K_1|4] = 0 \quad \mathbf{Pr}[K_2|4] = 0 \quad \mathbf{Pr}[K_3|4] = 1$$

所以，

$$H(\mathbf{K}|\mathbf{C}) = \frac{1}{8} \times 0 + \frac{7}{16} \times 0.59 + \frac{1}{4} + 0.81 + \frac{3}{16} \times 0 = 0.46$$

与由定理 2.10 得到的结果相符合。

设$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$是正在使用的密码体制，明文串

$$x_1 x_2 \cdots x_n$$

用一个密钥加密，得到密文串：

$$y_1 y_2 \cdots y_n$$

密码分析的基本目的是确定密钥。我们考虑惟密文攻击，并且假设分析者 Oscar 有无限的计算资源。同时假定 Oscar 知道明文是某一自然语言，如英语。一般来说，Oscar 能排除某些密钥，但仍存在许多可能的密钥，这其中只有一个密钥是正确的。那些可能的但不正确的密钥称为伪密钥。

例如，假设 Oscar 得到密文“WNAJW”，这个密文是使用移位密码得到的。容易知道，只有两个有意义的明文串，即“river”和“arena”，分别对应于可能的加密密钥 $F(=5)$和 $W(=22)$。这两个密钥，一个是正确的密钥，一个是伪密钥(实际上，对长为 5 的移位密码的密文中找出两个有意义的解密是有些困难的；读者可以试着寻找其他一些例子)。

我们的目的是推导伪密钥的期望数的下界。首先，定义自然语言 L 的熵(每字母)的含义，记为 H_L。H_L 应该是对有意义的明文串中的每个字母平均信息的度量(注意，一个随机的字母串具有的熵(每字母)是 $\log_2 26 \approx 4.70$)。作为 H_L 的“一阶”近似值，我们使用 $H(\mathbf{P})$。L 是英语的时候，我们使用表 1.1 给出的概率分布得到 $H(\mathbf{P}) \approx 4.19$。

当然，语言中相继的字母不是统计独立的，相继字母的相关性减少了熵。例如在英语中，字母“Q”后面总是跟着字母“U”。作为“二阶”近似值，我们计算所有字母报的概率分布的熵，然后除以 2。一般地，定义 $\mathbf{P}^n$ 为构成所有 n 字母报的概率分布的随机变量。我们使用以下定义。

定义 2.7　假设 L 是自然语言，语言的熵定义为：

$$H_L = \lim_{n \to \infty} \frac{H(\mathbf{P}^n)}{n}$$

语言 L 的冗余度(redundacy)定义为：

$$R_L = 1 - \frac{H_L}{\log_2 |\mathcal{P}|}$$

注　H_L 中度量了语言 L 中每个字母的平均熵。一个随机的语言具有熵 $\log_2 |\mathcal{P}|$。因此

R_L 度量了“多余字母”的比例,即冗余度。

在英语中,可以通过大量字母报的列表和它们的频率给出 $H(\mathbf{P}^2)$的估计。$H(\mathbf{P}^2)/2 \approx 3.90$ 就是这样得出的。可以继续通过字母报列表,得到对 H_L 的估计。事实上,各种实验已经得出了一个经验性的结果:$1.0 \leqslant H_L \leqslant 1.5$。也就是说,英语的平均信息内容大概是每字母 1~1.5 比特。

使用 1.25 作为 H_L 的估计值,冗余度为 0.75。这意味着英语有 75% 的冗余度!(这不是说任意地从英语文本中去掉四分之三的字母,还可以阅读。而是说,对一个充分大的 n,可以找到一个 n 字母报的 Huffman 编码,将英语原文压缩到原来长度的四分之一。)

给定$\mathcal{K}$和$\mathcal{P}^n$ 的概率分布,我们可以定义导出的$\mathcal{C}^n$ 上的概率分布,其中$\mathcal{C}^n$ 是密文的 n 字母报(已经讨论过 $n=1$ 时的情况)。定义 $\mathbf{P}^n$ 为代表明文 n 字母报的随机变量。类似的,定义 $\mathbf{C}^n$ 为代表密文 n 字母报的随机变量。

给定 $\mathbf{y} \in \mathbf{C}^n$,定义:

$$K(\mathbf{y}) = \{K \in \mathcal{K} : \exists \mathbf{x} \in \mathcal{P}^n, \text{使得 } \mathbf{Pr}[\mathbf{x}] > 0, e_K(\mathbf{x}) = \mathbf{y}\}$$

也就是说,$K(\mathbf{y})$是一个密钥的集合,在这些密钥下,$\mathbf{y}$ 是长为 n 的有意义的明文串的密文;或者说,$K(\mathbf{y})$是密文为 y 的可能密钥的集合。如果 y 是被观察的密文串,那么伪密钥的个数是 $|K(\mathbf{y})| - 1$,因为只有一个可能的密钥是正确的密钥。伪密钥的平均数目(在所有可能的长为 n 的密文串上)记为 $\overline{s}_n$,其值的计算如下:

$$\begin{aligned}\overline{s}_n &= \sum_{\mathbf{y} \in \mathcal{C}^n} \mathbf{Pr}[\mathbf{y}](|K(\mathbf{y})| - 1) \\ &= \sum_{\mathbf{y} \in \mathcal{C}^n} \mathbf{Pr}[\mathbf{y}]|K(\mathbf{y})| - \sum_{\mathbf{y} \in \mathcal{C}^n} \mathbf{Pr}[\mathbf{y}] \\ &= \sum_{\mathbf{y} \in \mathcal{C}^n} \mathbf{Pr}[\mathbf{y}]|K(\mathbf{y})| - 1\end{aligned}$$

根据定理 2.10,我们有:

$$H(\mathbf{K}|\mathbf{C}^n) = H(\mathbf{K}) + H(\mathbf{P}^n) - H(\mathbf{C}^n)$$

我们也用到了估计

$$H(\mathbf{P}^n) \approx nH_L = n(1 - R_L)\log_2 |\mathcal{P}|$$

其中 n 充分大。当然

$$H(\mathbf{C}^n) \leqslant n \log_2 |\mathcal{C}|$$

如果$|\mathcal{C}| = |\mathcal{P}|$,则有:

$$H(\mathbf{K}|\mathbf{C}^n) \geqslant H(\mathbf{K}) - nR_L \log_2 |\mathcal{P}| \tag{2.2}$$

接下来,我们将 $H(\mathbf{K}|\mathbf{C}^n)$和伪密钥的个数 $\overline{s}_n$ 关联起来。计算如下:

$$H(\mathbf{K}|\mathbf{C}^n) = \sum_{\mathbf{y} \in \mathcal{C}^n} \mathbf{Pr}[\mathbf{y}] H(\mathbf{K} \mid \mathbf{y})$$

$$\leqslant \sum_{\mathbf{y}\in\mathcal{C}^n} \mathbf{Pr}[\mathbf{y}]\log_2 |K(\mathbf{y})|$$
$$= \log_2 \sum_{\mathbf{y}\in\mathcal{C}^n} \mathbf{Pr}[\mathbf{y}]\,|K(\mathbf{y})|$$
$$= \log_2(\bar{s}_n + 1)$$

其中，我们对函数 $f(x)=\log_2 x$ 用到了 Jensen 不等式(定理 2.5)。因此得到不等式

$$H(\mathbf{K}|\mathbf{C}^n) \leqslant \log_2(\bar{s}_n + 1) \tag{2.3}$$

将(2.2)和(2.3)结合起来，我们有：

$$\log_2(\bar{s}_n + 1) \geqslant H(\mathbf{K}) - nR_L \log_2 |\mathcal{P}|$$

考虑等概率选取密钥的情况(此时 $H(\mathbf{K})$取最大值)，我们得到如下结论。

定理 2.11　假设$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$是一个密码体制，$|\mathcal{C}| = |\mathcal{P}|$并且密钥是等概率选取的。设 R_L 表示明文的自然语言的冗余度，那么给定一个充分长(长为 n)的密文串，伪密钥的期望数满足：

$$\bar{s}_n \geqslant \frac{|\mathcal{K}|}{|\mathcal{P}|^{nR_L}} - 1$$

当 n 增加时，$\frac{|\mathcal{K}|}{|\mathcal{P}|^{nR_L}} - 1$ 以指数速度趋近 0。同时可出现，如果 n 很小，$H(\mathbf{P}^n)/n$ 可能对 H_L 的估计不够好，此时上面的估计可能不大精确。

还有一个概念需要定义。

定义 2.8　一个密码体制的惟一解距离，定义为使得伪密钥的期望数等于零的 n 的值，记为 n_0，即在给定足够的计算时间下，分析者能惟一计算出密钥所需密文的平均量。

如果在定理 2.11 中令 $\bar{s}_n = 0$，解出 n，我们可以得到惟一解距离的一个近似估计，即

$$n_0 \approx \frac{\log_2 |\mathcal{K}|}{R_L \log_2 |\mathcal{P}|}$$

作为一个例子，考虑代换密码。在这种密码体制中，$|\mathcal{P}| = 26$，$|\mathcal{K}| = 26!$。如果取 $R_L = 0.75$，就得到惟一解距离的估计：

$$n_0 \approx 88.4/(0.75\times 4.7) \approx 25$$

这意味着，给定的密文串的长至少是 25 时，通常解密才是惟一的。

2.7 乘积密码体制

Shannon 在其 1949 年的论文中介绍的另一个新思想是通过“乘积”来组合密码体制。这种思想在现代密码体制的设计中非常重要，比如 AES 的设计，我们将在下一章进行研究。

为简单起见，这一节中我们将注意力集中在$\mathcal{C}=\mathcal{P}$的密码体制上：这种类型的密码体制称为内包的密码体制(endomorphic cryptosystem)。设 $\mathbf{S}_1=(\mathcal{P},\mathcal{P},\mathcal{K}_1,\mathcal{E}_1,\mathcal{D})$，$\mathbf{S}_2=(\mathcal{P},\mathcal{P},\mathcal{K}_2,\mathcal{E}_2,\mathcal{D}_2)$是两个具有相同明文空间(密文空间)的内包的密码体制。那么 $\mathbf{S}_1$ 和 $\mathbf{S}_2$ 的乘积密码体制 $\mathbf{S}_1\times\mathbf{S}_2$ 定义为：

$$(\mathcal{P},\mathcal{P},\mathcal{K}_1\times\mathcal{K}_2,\mathcal{E},\mathcal{D})$$

乘积密码体制的密钥形式是 $K=(K_1,K_2)$，其中 $K_1\in\mathcal{K}_1$，$K_2\in\mathcal{K}_2$。加密和解密的规则定义如下。

对于任意一个 $K=(K_1,K_2)$，加密 e_K 定义为：

$$e_{(K_1,K_2)}(x)=e_{K_2}(e_{K_1}(x))$$

解密 d_K 定义为：

$$d_{(K_1,K_2)}(y)=d_{K_1}(d_{K_2}(y))$$

也就是，我们首先用 e_{K_1} 加密 x，然后用 e_{K_2} 再加密一次所得到的密文。解密是类似的，但是必须以相反的次序进行：

$$\begin{aligned}d_{(K_1,K_2)}(e_{(K_1,K_2)}(x)) &= d_{(K_1,K_2)}(e_{k_2}(e_{K_1}(x)))\\ &= d_{K_1}(d_{K_2}(e_{K_2}(e_{K_1}(x))))\\ &= d_{K_1}(e_{K_1}(x))\\ &= x\end{aligned}$$

回忆一下，密码体制有和密钥空间相关的概率分布，因此需要定义密钥空间$\mathcal{K}$的概率分布。我们很自然地定义：

$$\mathbf{Pr}[(K_1,K_2)]=\mathbf{Pr}[K_1]\times\mathbf{Pr}[K_2]$$

换句话说，分别根据定义在$\mathcal{K}_1$ 和$\mathcal{K}_2$ 上的概率分布，独立地选取 K_1 和 K_2。

下面用一个简单的例子来说明一个乘积的密码体制的定义。定义乘法密码(Multiplicative Cipher)如下。

密码体制 2.2　乘法密码

设$\mathcal{P}=\mathcal{C}=\mathbb{Z}_{26}$,并且

$$\mathcal{K}=\{a\in\mathbb{Z}_{26}:\gcd(a,26)=1\}$$

对于 $a\in\mathcal{K}$,定义

$$e_a(x)=ax \bmod 26$$

并且,

$$d_a(y)=a^{-1}y \bmod 26$$

$(x,y\in\mathbb{Z}_{26})$。

假设 **M** 是乘法密码(密钥等概率选取),**S** 是移位密码(密钥等概率选取)。很容易看出 **M** × **S** 也是仿射密码(同样,密钥等概率选取)。要说明 **S** × **M** 是密钥等概率选取的仿射密码要略微困难一些。

现在我们证明这个论断。移位密码的密钥是 $K\in\mathbb{Z}_{26}$;相应的加密规则为 $e_K(x)=(x+K) \bmod 26$。乘法密码的密钥是 $a\in\mathbb{Z}_{26}$,$\gcd(a,26)=1$;相应的加密规则是 $e_a(x)=ax \bmod 26$。因此,乘积密码 **M** × **S** 的密钥具有(a,K)的形式,并且

$$e_{(a,K)}(x)=(ax+K) \bmod 26$$

但是,这就是仿射密码的定义。另外,仿射密码的密钥的概率是密钥 a 和 K 的概率乘积:$1/312=1/12\times 1/26$。因此 **M** × **S** 是仿射密码。

现在考虑 **S** × **M**。这个密码的密钥具有形式(K,a),并且

$$e_{(K,a)}(x)=a(x+K) \bmod 26=(ax+aK) \bmod 26$$

这样,乘积密码 **S** × **M** 的密钥(K,a)等同于仿射密码的密钥(a,aK)。剩下要证明的是密钥在仿射密码 **S** × **M** 中具有和在乘积密码中相同的概率 1/312。注意到,$aK=K_1$ 当且仅当 $K=a^{-1}K_1$($\gcd(a,26)=1$,故 a 对模 26 有乘积逆)。换句话说,仿射密码中的密钥(a,K_1)等同于乘积密码 **S** × **M** 中的密钥$(a^{-1}K_1,a)$。这样,我们在两个密钥空间之间建立了一个双射。因为每个密钥都是等概率的,**S** × **M** 确实是仿射密码。

我们已经证明了 **M** × **S** = **S** × **M**,因此可以说密码体制 **M** 和 **S** 是可交换的。但是,不是所有的密码体制都是可交换的,很容易找出反例。另一方面,乘积运算是可结合的:

$$(\mathbf{S}_1\times\mathbf{S}_2)\times\mathbf{S}_3=\mathbf{S}_1\times(\mathbf{S}_2\times\mathbf{S}_3)$$

如果将(内包的)密码体制和自己做乘积,我们得到密码体制 **S** × **S**,记为 $\mathbf{S}^2$。如果做 n 重乘积,得到的密码体制记为 $\mathbf{S}^n$。

如果 $\mathbf{S}^2=\mathbf{S}$,一个密码体制是幂等的。我们在第 1 章中研究的许多密码体制都是幂等的。

例如移位密码、代换密码、仿射密码、希尔密码、维吉尼亚密码和置换密码都是幂等的。当然，如果一个密码体制是幂等的，使用乘积系统 $\mathbf{S}^2$ 就毫无意义，因为这需要多余的密钥，但并没有提供更多的安全性。

如果密码体制不是幂等的，那么多次迭代有可能提高安全性。这个思想在 DES 中使用过了，其中包括了 16 轮的迭代。但是，这种途径显然要求以非幂等的密码体制开始。一种构造简单的非幂等的密码体制的方法，是对两个不同的(简单的)密码体制做乘积。

注 不难证明如果 $\mathbf{S}_1$ 和 $\mathbf{S}_2$ 都是幂等的，并且是可交换的，则 $\mathbf{S}_1 \times \mathbf{S}_2$ 也是幂等的。证明如下：

$$\begin{aligned}(\mathbf{S}_1 \times \mathbf{S}_2) \times (\mathbf{S}_1 \times \mathbf{S}_2) &= \mathbf{S}_1 \times (\mathbf{S}_2 \times \mathbf{S}_1) \times \mathbf{S}_2 \\ &= \mathbf{S}_1 \times (\mathbf{S}_1 \times \mathbf{S}_2) \times \mathbf{S}_2 \\ &= (\mathbf{S}_1 \times \mathbf{S}_1) \times (\mathbf{S}_2 \times \mathbf{S}_2) \\ &= \mathbf{S}_1 \times \mathbf{S}_2\end{aligned}$$

(注意，证明中用到了结合律。)

因此，如果 $\mathbf{S}_1$ 和 $\mathbf{S}_2$ 都是幂等的，我们要求 $\mathbf{S}_1 \times \mathbf{S}_2$ 不是幂等的，则 $\mathbf{S}_1$ 和 $\mathbf{S}_2$ 是不可交换的。

幸运的是，有许多简单的密码体制适合这种类型的构造。通常使用的技术是将代换密码和置换密码做乘积。我们将在下一章中看到几个这样的实现。

2.8 注释与参考文献

完善的保密系统的思想和熵在密码学中的使用最先出现于 Shannon[191]。这篇论文也讨论了乘积密码体制。Shannon[190]定义了熵的概念。熵、Huffman 编码和相关的论题可以在 Welsh[213]、Goldie 和 Pinch[93]的书中找到很好的介绍。

2.6 节中的结果来自于 Beauchemin 和 Brassard[6]，他们对 Shannnon 的早期结论做了一般化处理。

练习

2.1 参见例 2.2，计算所有的联合概率和条件概率 $\mathbf{Pr}[x, y]$、$\mathbf{Pr}[x \mid y]$和 $\mathbf{Pr}[y \mid x]$，其中 $x \in \{2, \cdots, 12\}$，$y \in \{D, N\}$。

2.2 设 n 是正整数。阶为 n 的拉丁方(Latin square)是 $n \times n$ 矩阵 $\boldsymbol{L}$，n 个整数 $1, \cdots, n$ 中的每一个在 $\boldsymbol{L}$ 的每一行和每一列中恰好出现一次。下面是一个 3 阶拉丁方的例子：

1	2	3
3	1	2
2	3	1

给定一个阶为 n 的拉丁方 $\boldsymbol{L}$，我们可以定义一个相关的密码体制。取 $\mathcal{P}=\mathcal{C}=\mathcal{K}=\{1,\cdots,n\}$。对于 $1\leqslant i\leqslant n$，加密规则 e_i 定义为 $e_i(j)=\boldsymbol{L}(i,j)$（这样每一行都给出一个加密规则）。

直接证明，如果密钥是等概率的，拉丁方密码体制具有完善保密性。

2.3　(a)证明仿射密码是完善保密的，如果每个密钥的概率都是 1/312。

(b)更一般地，假设在集合

$$\{a\in\mathbb{Z}_{26}:\gcd(a,26)=1\}$$

上给定一个概率分布，且仿射密码的每个密钥 (a,b) 的概率是 $\mathbf{Pr}[a]/26$。证明当这个概率分布定义在密钥空间上时，仿射密码具有完善保密性。

2.4　假设一个密码体制对一个特定的明文的概率分布是完善保密的，证明对任意的明文概率分布，这个密码体制仍然是完善保密的。

2.5　证明：如果一个密码体制是完善保密的，并且 $|\mathcal{K}|=|\mathcal{C}|=|\mathcal{P}|$，则每个密文都是等概率的。

2.6　假设在“一次一密”密码中，密文 y 和 y'（两个二进制的 n 元数组）是使用同一个密钥 K，分别加密明文 x 和 x' 得到的。证明 $x+x'\equiv y+y'\pmod 2$。

2.7　(a)对 $n=3$ 的“一次一密”密码，构造一个加密矩阵（如例 2.3 中定义的那样）。

(b)设 n 是任一正整数，直接证明“一次一密”密码的加密矩阵是定义在 $(\mathbb{Z}_2)^n$ 上的阶为 2^n 的拉丁方。

2.8　假设 X 是个数为 n 的集合，其中 $2^k\leqslant n<2^{k+1}$，对于任意的 $x\in X$，$\mathbf{Pr}[x]=1/n$。

(a) 找出一个 X 的无前缀的编码，设为 f，使得 $l(f)=k+2-2^{k+1}/n$。

提示：将 X 中的 $2^{k+1}-n$ 个元素，编码成长为 k 的串，剩下的编码成长为 $k+1$ 的串。

(b) 对 $n=6$ 说明你的构造，并计算 $l(f)$ 和 $H(\mathbf{X})$。

2.9　假设 $X=\{\mathrm{a},\mathrm{b},\mathrm{c},\mathrm{d},\mathrm{e}\}$ 有如下概率分布：$\mathbf{Pr}[\mathrm{a}]=0.32$，$\mathbf{Pr}[\mathrm{b}]=0.23$，$\mathbf{Pr}[\mathrm{c}]=0.20$，$\mathbf{Pr}[\mathrm{d}]=0.15$，$\mathbf{Pr}[\mathrm{e}]=0.10$。使用 Huffman 编码找出无前缀的最佳编码。将这个编码的长度和 $H(\mathbf{X})$ 进行比较。

2.10　证明 $H(\mathbf{X},\mathbf{Y})=H(\mathbf{Y})+H(\mathbf{X}|\mathbf{Y})$。然后得到推论 $H(\mathbf{X}|\mathbf{Y})\leqslant H(\mathbf{X})$，当且仅当 $\mathbf{X}$ 和 $\mathbf{Y}$ 统计独立时等号才成立。

2.11　证明当且仅当 $H(\mathbf{P}|\mathbf{C})=H(\mathbf{P})$ 时，密码体制是完善保密的。

2.12　证明在任何密码体制中，$H(\mathbf{K}|\mathbf{C})\geqslant H(\mathbf{P}|\mathbf{C})$。（直观上讲，给定密文，分析者对密钥的不确定性至少和对明文的不确定性一样大。）

2.13　考虑一个密码体制，其中 $\mathcal{P}=\{a,b,c\}$，$\mathcal{K}=\{K_1,K_2,K_3\}$，$\mathcal{C}=\{1,2,3,4\}$。假设加密矩阵如下：

	a	b	c
K_1	1	2	3
K_2	2	3	4
K_3	3	4	1

假设密钥是等概率选取的,明文的概率分布是 $\mathbf{Pr}[a]=1/2, \mathbf{Pr}[b]=1/3, \mathbf{Pr}[c]=1/6$,计算 $H(\mathbf{P})$、$H(\mathbf{C})$、$H(\mathbf{K})$、$H(\mathbf{K}|\mathbf{C})$和 $H(\mathbf{P}|\mathbf{C})$。

2.14 对仿射密码计算 $H(\mathbf{K}|\mathbf{C})$和 $H(\mathbf{K}|\mathbf{P},\mathbf{C})$。

2.15 考虑密钥字长为 m 的维吉尼亚密码。证明惟一解距离是 $1/R_L$,其中 R_L 是自然语言的冗余度。(这个结论可以解释如下。如果 n_0 表示加密的字母数,因为每个明文由 m 个字母组成,所以明文的"长度"是 n_0/m。因此惟一解距离 $1/R_L$ 对应于由 m/R_L 个字母组成的明文。)

2.16 证明具有 $m \times m$ 加密矩阵的希尔密码的惟一解距离不小于 m/R_L。(注意,这个长度对应的明文中的字符个数是 m^2/R_L。)

2.17 明文空间大小为 n 的代换密码的密钥量为 $|\mathcal{K}| = n!$。Stirling 公式给出了 $n!$ 的如下估计:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

(a)使用 Stirling 公式得出代换密码的惟一解距离的估计。

(b)设 $m \geqslant 1$ 是整数,m 字母报代换密码是明文(密文)空间由所有的 26^m 个 m 字母报构成的代换密码。对 $R_L = 0.75$,估计 m 字母报代换密码的惟一解距离。

2.18 证明密钥等概率选取的移位密码是幂等的。

2.19 假设 $\mathbf{S}_1$ 是移位密码(密钥等概率),$\mathbf{S}_2$ 是密钥满足概率分布 $P_{\mathcal{K}}$(不必是等概率的)的移位密码。证明 $\mathbf{S}_1 \times \mathbf{S}_2 = \mathbf{S}_1$。

2.20 假设 $\mathbf{S}_1$ 和 $\mathbf{S}_2$ 都是维吉尼亚密码,密钥都是等概率的,并且长分别是 m_1 和 m_2,$m_1 > m_2$。

(a)如果 $m_1 \mid m_2$,则 $\mathbf{S}_2 \times \mathbf{S}_1 = \mathbf{S}_1$。

(b)有人可能试图证明 $\mathbf{S}_2 \times \mathbf{S}_1 = \mathbf{S}_3$,其中 $\mathbf{S}_3$ 是密钥长为 $\mathrm{lcm}(m_1, m_2)$的维吉尼亚密码来得到上面的结果。证明这个猜想是错误的。

提示:如果 $m_1 \not\equiv 0 \pmod{m_2}$,则乘积密码 $\mathbf{S}_2 \times \mathbf{S}_1$ 的密钥个数小于 $\mathbf{S}_3$ 的密钥个数。

第3章　分组密码与高级加密标准

3.1　引言

当今大多数的分组密码都是乘积密码（乘积密码的介绍参见2.7节）。乘积密码通常伴随一系列置换与代换操作，常见的乘积密码是迭代密码。下面是一个典型的迭代密码的定义：这种密码明确定义了一个轮函数和一个密钥编排方案，一个明文的加密将经过Nr轮类似的过程。

设 K 是一个确定长度的随机二元密钥，用 K 来生成Nr个轮密钥（也叫子密钥）。$K^1,\cdots,K^{Nr}$ 轮密钥的列表（$K^1,\cdots,K^{Nr}$）就是密钥编排方案。密钥编排方案通过一个固定的、公开的算法生成。

轮函数 g 以轮密钥（K^r）和当前状态 w^{r-1} 作为它的两个输入。下一个状态定义为 $w^r = g(w^{r-1},K^r)$。初态 w^0 被定义成明文 x，密文 y 定义为经过所有Nr轮后的状态。整个加密操作过程如下：

$$
\begin{aligned}
w^0 &\leftarrow x\\
w^1 &\leftarrow g(w^0,K^1)\\
w^2 &\leftarrow g(w^1,K^2)\\
&\vdots\\
w^{Nr-1} &\leftarrow g(w^{Nr-2},K^{Nr-1})\\
w^{Nr} &\leftarrow g(w^{Nr-1},K^{Nr})\\
y &\leftarrow w^{Nr}
\end{aligned}
$$

为了能够解密，轮函数 g 在其第二个自变量固定的条件下必须是单射函数，这等价于存在函数 g^{-1}，对所有的 w 和 y，有 $g^{-1}(g(w,y),y)=w$。解密过程如下：

$$
\begin{aligned}
w^{Nr} &\leftarrow y\\
w^{Nr-1} &\leftarrow g^{-1}(w^{Nr},K^{Nr})\\
&\vdots\\
w^1 &\leftarrow g^{-1}(w^2,K^2)\\
w^0 &\leftarrow g^{-1}(w^1,K^1)\\
x &\leftarrow w^0
\end{aligned}
$$

在3.2节中，我们将用一类简单的迭代密码，代换-置换网络（substitution-permutation network），来说明实际的分组密码设计中应用的主要原则。3.3和3.4节分别介绍了线性密码分析和差分密码分析。3.5节讨论了Feistel型密码与数据加密标准（DES）。我们在3.6节中给

出了高级加密标准(AES)。最后,在 3.7 节中介绍了分组密码的工作模式。

3.2 代换-置换网络

我们首先给出代换-置换网络(SPN)的定义。可以看到,一个 SPN 就是一类特殊的迭代密码,只是在某些地方有一点小变化,后面我们将指出这些变化。设 l 和 m 都是正整数,明密文都是长为 lm 的二元向量(即 lm 是该密码的分组长度)。一个 SPN 包括两个变换,分别记为 π_S 和 π_P。

$$\pi_S : \{0,1\}^l \rightarrow \{0,1\}^l$$

与

$$\pi_P : \{1,\cdots,lm\} \rightarrow \{1,\cdots,lm\}$$

都是置换,置换 π_S 叫做 S 盒(字母"S"表示"substitution(代换)"),它用一个 l 比特向量来替代另一个 l 比特向量;置换 π_P 用来置换 lm 个比特。

给定一个 lm 比特的二元串 $x = (x_1,\cdots,x_{lm})$,可以将其看做 m 个 l 比特子串 $x_{(1)}\cdots,x_m$ 的并联。因此,

$$x = x_{(1)} \| \cdots \| x_{(m)}$$

且对于 $1 \leqslant i \leqslant m$,我们有

$$x_{(i)} = (x_{(i-1)l+1},\cdots,x_{il})$$

将要给出的 SPN 由 Nr 轮组成,在每一轮(除了最后一轮稍有不同外),我们将先用异或操作混入该轮密钥,再用 π_S 进行 m 次代换,然后用 π_P 进行一次置换。基于 π_S 和 π_P,我们现在给出一个 SPN,参见密码体制。

密码体制 3.1 代换-置换网络

设 l,m 和 Nr 都是正整数,$\pi_S : \{0,1\}^l \rightarrow \{0,1\}^l \rightarrow$ 与 $\pi_P : \{1,\cdots,lm\} \rightarrow \{1,\cdots,lm\}$ 均为置换。

设 $\mathcal{P} = \mathcal{C} = \{0,1\}^{lm}$,$\mathcal{K} \subseteq (\{0,1\}^{lm})^{Nr+1}$ 是由初始密钥 K 用密钥编排算法生成的所有可能密钥编排方案之集。对一个密钥编排方案 $(K^1,\cdots,K^{Nr+1})$,我们使用算法 3.1 来加密明文 x。

算法 3.1 $\text{SPN}(x,\pi_S,\pi_P,(K^1,\cdots,K^{Nr+1}))$

$w^0 \leftarrow x$

for $r \leftarrow 1$ **to** $Nr-1$

$$\textbf{do}\begin{cases} u^r \leftarrow w^{r-1} \oplus K^r \\ \textbf{for} \quad i \leftarrow 1 \ \textbf{to} \ m \\ \quad \textbf{do} \ v^r_{(i)} \leftarrow \pi_S(u^r_{(i)}) \\ w^r \leftarrow (v^r_{\pi_P(1)}, \cdots, v^r_{\pi_P(lm)}) \end{cases}$$

$u^{Nr} \leftarrow w^{Nr-1} \oplus K^{Nr}$

for $i \leftarrow 1$ **to** m

　do $v^{Nr}_{(i)} \leftarrow \pi_S(u^{Nr}_{(i)})$

$y \leftarrow v^{Nr} \oplus K^{Nr+1}$

output(y)

在算法 3.1 中，u^r 是第 r 轮对 S 盒的输入，v^r 是第 r 轮对 S 盒的输出。w^r 由 v^r 应用置换 π_P 得到，然后 u^{r+1} 由轮密钥 K^{r+1} 异或 w^r 得到(这叫做轮密钥混合)，最后一轮没有用置换 π_P。如果对密钥编排方案做适当修改并用 S 盒的逆来取代 S 盒，那么该加密算法也能用来解密(参见练习)。

值得注意的是，该 SPN 的第一个和最后一个操作都是异或轮密钥，这叫做白化(whitening)。如果一个攻击者不知道密钥的话，他将无法开始进行一个加密或解密操作。

下面用一个特定的 SPN 来说明上面的一般性描述。

例 3.1　设 $l = m = N_r = 4$，π_S 如下定义，这里输入 z 和输出 $\pi_S(z)$都以十六进制表示(0↔(0,0,0,0)，1↔(0,0,0,1)，…，9↔(1,0,0,1)，A↔(1,0,1,0)，…，F↔(1,1,1,1))：

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

π_P 如下定义：

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

图 3.1 给出了该 SPN，为后面叙述的方便，我们命名了 16 个 S 盒，$S^r_i(1 \leqslant i \leqslant 4, 1 \leqslant r \leqslant 4)$，它们都基于同样的置换 π_S。下面给出该 SPN 的密钥编排算法。我们以一个 32 比特的密钥 $K = (k_1, \cdots, k_{32}) \in \{0,1\}^{32}$开始。对轮数 $1 \leqslant r \leqslant 5$，定义 K^r是由 K 中从 k_{4r-3}开始的 16 个相邻比特组成的(选择这样的简单密钥编排方案只是为了说明 SPN，并非是一种很安全的方式)。

下面我们使用该 SPN 来进行加密。所有数据都用二元形式表示。设密钥是：

$$K = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$$

则轮密钥如下：

$$K^1 = 0011\ 1010\ 1001\ 0100$$

$$K^2 = 1010\ 1001\ 0100\ 1101$$
$$K^3 = 1001\ 0100\ 1101\ 0110$$
$$K^4 = 0100\ 1101\ 0110\ 0011$$
$$K^5 = 1101\ 0110\ 0011\ 1111$$

设明文为:

$$x = 0010\ 0110\ 1011\ 0111$$

则加密 x 的过程如下:

$$w^0 = 0010\ 0110\ 1011\ 0111$$
$$K^1 = 0011\ 1010\ 1001\ 0100$$
$$u^1 = 0001\ 1100\ 0010\ 0011$$
$$v^1 = 0100\ 0101\ 1101\ 0001$$
$$w^1 = 0010\ 1110\ 0000\ 0111$$
$$K^2 = 1010\ 1001\ 0100\ 1101$$
$$u^2 = 1000\ 0111\ 0100\ 1010$$
$$v^2 = 0011\ 1000\ 0010\ 0110$$
$$w^2 = 0100\ 0001\ 1011\ 1000$$
$$K^3 = 1001\ 0100\ 1101\ 0110$$
$$u^3 = 1101\ 0101\ 0110\ 1110$$
$$v^3 = 1001\ 1111\ 1011\ 0000$$
$$w^3 = 1110\ 0100\ 0110\ 1110$$
$$K^4 = 0100\ 1101\ 0110\ 0011$$
$$u^4 = 1010\ 1001\ 0000\ 1101$$
$$v^4 = 0110\ 1010\ 1110\ 1001$$
$$K^5 = 1101\ 0110\ 0011\ 1111$$

密文为:

$$y = 1011\ 1100\ 1101\ 0110$$

SPN 有一些颇具吸引力的特色。首先,无论从硬件还是软件的角度来看,这种设计均简单、有效。在软件方面,一个 S 盒通常以查表的形式来实现。注意,S 盒 $\pi_S: \{0,1\}^l \rightarrow \{0,1\}^l$ 所需的存储量是 $l2^l$ 比特,这是因为我们必须存储 2^l 个值,而每个值占 l 比特。特别地,硬件实现必须使用相对小的 S 盒。

在例 3.1 中,每一轮都应用了相同的 S 盒,S 盒的存储需求是 2^6 比特。如果我们应用由 16 比特映射到 16 比特的 S 盒,则存储需求会增加到 2^{20} 比特,这对某些应用来说是过高了。在高级加密标准(3.6 节将会讨论)中应用的就是 8 比特映射到 8 比特的 S 盒。

例 3.1 中的 SPN 并不安全,即使没有其他原因,32 比特的密钥长度对穷尽密钥搜索攻击来说也是太短了。然而,"更大些"的 SPN 可设计成能抵抗所有已知的攻击。一个实际的安全的 SPN 比例 3.1 中的 SPN 会具有更长的密钥长度和分组长度,应用更大些的 S 盒,并有更多的

轮数。已被选做高级加密标准的 Rijndael 就是 SPN 的一个例子，它在许多方面与例 3.1 中的 SPN 相似。Rijndael 的最小密钥长度是 128 比特，分组长度是 128 比特，最小轮数是 10 轮，并且它的 S 盒将 8 比特映射到 8 比特(参见 3.6 节)。

SPN 的许多变体也是可能的。常见的一个改动就是不止应用一个 S 盒。在例 3.1 中，如果需要，完全可以在每一轮应用不同的 S 盒，来代替四轮中使用的相同的 S 盒。这一特点可以在数据加密标准中找到，它在 8 轮中使用了 8 个不同的 S 盒(参见 3.5.1 小节)。另一个常见的设计策略是在每一轮中包含一个可逆的线性变换，该线性变换要么作为置换操作的替代，要么作为其补充，这可以在高级加密标准中看到(参见 3.6.1 小节)。

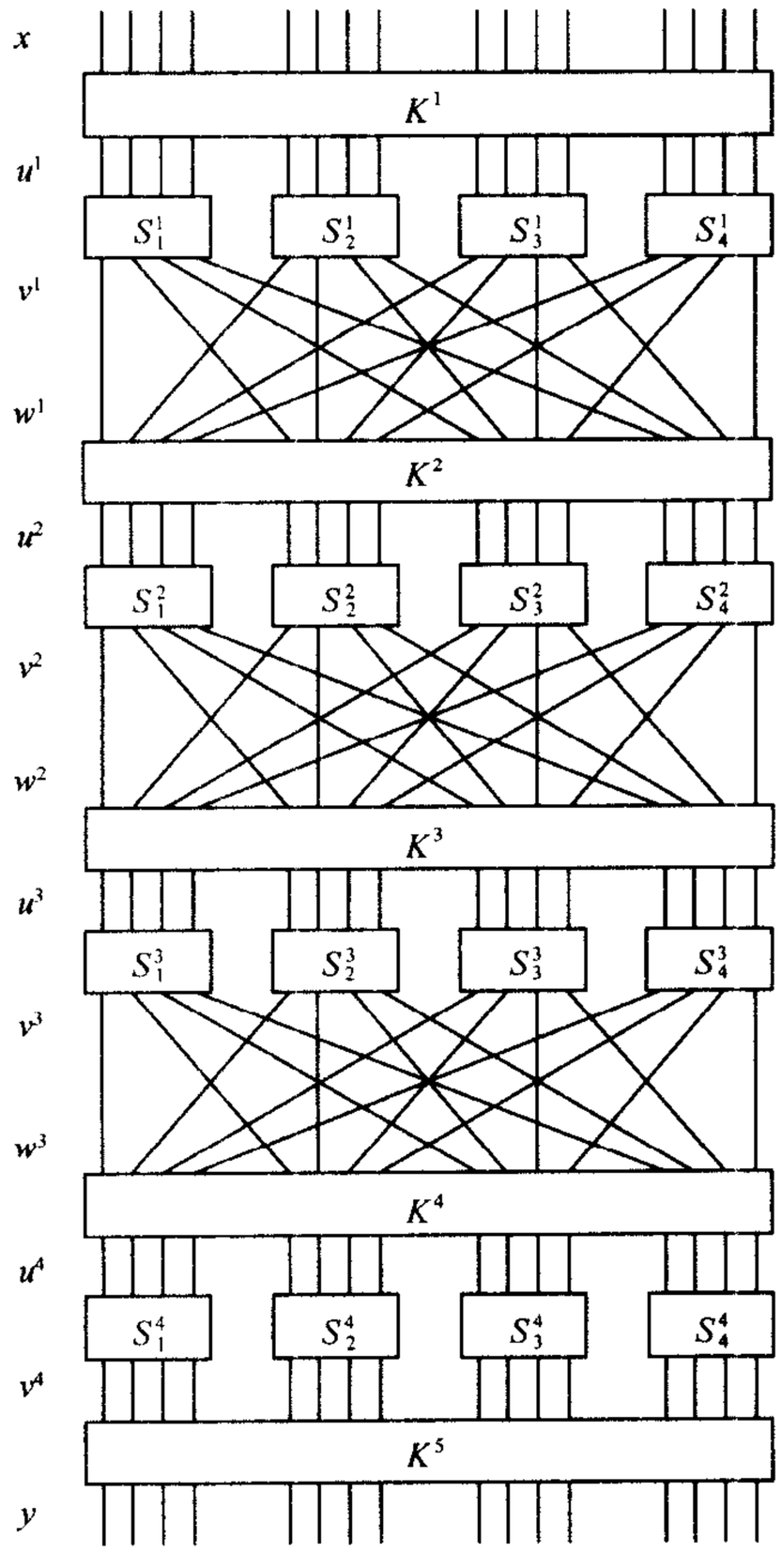

图 3.1　一个代换-置换网络

3.3　线性密码分析

我们首先非正式地叙述一下线性密码分析的思想，原则上该思想可以应用于任何迭代密码。假使能够在一个明文比特子集和最后一轮即将进行代换的输入状态比特子集之间找到一个概率线性关系，换句话说，即存在一个比特子集使得其中元素的异或表现出非随机的分布

(比如,该异或值以偏离 1/2 的概率取 0 值)。现在假设一个攻击者拥有大量的用同一未知密钥加密的明-密文对(也就是说我们使用一个已知明文攻击)。对每一个明-密文对,我们将用所有可能的候选密钥来对最后一轮解密密文。对每一个候选密钥,我们计算包含在线性关系式中的相关状态比特的异或值,然后确定上述的线性关系是否成立。如果成立,就在对应于特定候选密钥的计数器上加 1。在这个过程的最后,我们希望计数频率离明-密文对数的一半最远的候选密钥含有那些密钥比特的正确值。

后面我们将以一个详细的例子来说明上述思想。但首先,我们需要根据概率论的一些结论来为包含在这个攻击中的技术提供一个非严格的说明。

3.3.1 堆积引理

我们沿用 2.2 节中引入的术语和概念。设 $\mathbf{X}_1, \mathbf{X}_2$ 是取值于集合 $\{0,1\}$ 的独立随机变量。设 $p_1, p_2, \cdots$ 都是实数,且对所有的 i,有 $0 \leqslant p_i \leqslant 1$,再设

$$\mathbf{Pr}[\mathbf{X}_i = 0] = p_i$$

$i = 1,2,\cdots$,则:

$$\mathbf{Pr}[\mathbf{X}_i = 1] = 1 - p_i$$

$i = 1,2,\cdots$。假设 $i \neq j$,则 $\mathbf{X}_i$ 和 $\mathbf{X}_j$ 的独立性意味着:

$$\mathbf{Pr}[\mathbf{X}_i = 0, \mathbf{X}_j = 0] = p_i p_j$$
$$\mathbf{Pr}[\mathbf{X}_i = 0, \mathbf{X}_j = 1] = p_i(1 - p_j)$$
$$\mathbf{Pr}[\mathbf{X}_i = 1, \mathbf{X}_j = 0] = (1 - p_i)p_j$$
$$\mathbf{Pr}[\mathbf{X}_i = 1, \mathbf{X}_j = 1] = (1 - p_i)(1 - p_j)$$

现在考虑离散随机变量 $\mathbf{X}_i \oplus \mathbf{X}_j$(即 $\mathbf{X}_i + \mathbf{X}_j \bmod 2$)。容易看出 $\mathbf{X}_i \oplus \mathbf{X}_j$ 具有以下概率分布:

$$\mathbf{Pr}[\mathbf{X}_i \oplus \mathbf{X}_j = 0] = p_i p_j + (1 - p_i)(1 - p_j)$$
$$\mathbf{Pr}[\mathbf{X}_i \oplus \mathbf{X}_j = 1] = p_i(1 - p_j) + (1 - p_i)p_j$$

对取值于(0,1)的随机变量,用分布偏差来表示它的概率分布常常是很方便的。$\mathbf{X}_i$ 的偏差被定义为:

$$\epsilon_i = p_i - \frac{1}{2}$$

注意下列事实:对 $i = 1,2,\cdots$

$$-\frac{1}{2} \leqslant \epsilon_i \leqslant \frac{1}{2}$$

$$\mathbf{Pr}[\mathbf{X}_i = 0] = \frac{1}{2} + \epsilon_i$$

$$\mathbf{Pr}[\mathbf{X}_i = 1] = \frac{1}{2} - \epsilon_i$$

对 $i_1 < i_2 < \cdots < i_k$，用 $\epsilon_{i_1, i_2, \cdots, i_k}$ 来表示随机变量 $\mathbf{X}_{i_1} \oplus \mathbf{X}_{i_2} \oplus \cdots \oplus \mathbf{X}_{i_k}$ 的偏差，不难验证。$\epsilon_{i_1, i_2} = 2\epsilon_{i_1}\epsilon_{i_2}$。一般说来，下列事实成立，并被称做“堆积引理”(piling-up lemma)。

引理 3.1(堆积引理)　设 $\mathbf{X}_{i_1}, \cdots, \mathbf{X}_{i_k}$ 是独立随机变量，$\epsilon_{i_1, i_2, \cdots, i_k}$ 表示随机变量 $\mathbf{X}_{i_1} \oplus \mathbf{X}_{i_2} \oplus \cdots \oplus \mathbf{X}_{i_k}$ 的偏差，则

$$\epsilon_{i_1, i_2, \cdots, i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}$$

证明　对 k 应用归纳法。显然，$k = 1$ 时结论成立。假设 $k = 1$ 时结论成立，正整数 $l \geqslant 1$。则当 $k = l + 1$ 时，我们需要确定 $\mathbf{X}_{i_1} \oplus \cdots \oplus \mathbf{X}_{i_{l+1}}$ 的偏差。由归纳假设，$\mathbf{X}_{i_1} \oplus \cdots \oplus \mathbf{X}_{i_l}$ 的偏差为 $2^{l-1} \prod_{j=1}^{l} \epsilon_{i_j}$，因此，

$$\mathbf{Pr}[\mathbf{X}_{i_1} \oplus \cdots \oplus \mathbf{X}_{i_l} = 0] = \frac{1}{2} + 2^{l-1} \prod_{j=1}^{l} \epsilon_{i_j}$$

$$\mathbf{Pr}[\mathbf{X}_{i_1} \oplus \cdots \oplus \mathbf{X}_{i_l} = 1] = \frac{1}{2} - 2^{l-1} \prod_{j=1}^{l} \epsilon_{i_j}$$

显见，

$$\begin{aligned}
&\mathbf{Pr}[\mathbf{X}_{i_1} \oplus \cdots \oplus \mathbf{X}_{i_{l+1}} = 0] \\
&= \left(\frac{1}{2} + 2^{l-1} \prod_{j=1}^{l} \epsilon_{i_j}\right)\left(\frac{1}{2} + \epsilon_{i_{l+1}}\right) + \left(\frac{1}{2} - 2^{l-1} \prod_{j=1}^{1} \epsilon_{i_j}\right)\left(\frac{1}{2} - \epsilon_{i_{l+1}}\right) \\
&= \frac{1}{2} + 2^{l} \prod_{j=1}^{l+1} \epsilon_{i_j}
\end{aligned}$$

由归纳假设，结论成立。

推论 3.2　设 $\mathbf{X}_{i_1}, \cdots, \mathbf{X}_{i_k}$ 是独立随机变量，$\epsilon_{i_1, i_2, \cdots, i_k}$ 表示随机变量 $\mathbf{X}_{i_1} \oplus \mathbf{X}_{i_2} \oplus \cdots \oplus \mathbf{X}_{i_k}$ 的偏差，若对某 j，有 $\epsilon_{i_j = 0}$，则 $\epsilon_{i_1, i_2, \cdots, i_k} = 0$。

注意，引理 3.1 一般来说只在相关随机变量是统计独立的情况下才成立。我们通过下面的例子可以说明这一点。假设 $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1/4$，由引理 3.1，$\epsilon_{1,2} = \epsilon_{1,3} = \epsilon_{2,3} = 1/8$，现在考虑随机变量 $\mathbf{X}_1 \oplus \mathbf{X}_3$，显然，

$$\mathbf{X}_1 \oplus \mathbf{X}_3 = (\mathbf{X}_1 \oplus \mathbf{X}_2) \oplus (\mathbf{X}_2 \oplus \mathbf{X}_3)$$

如果随机变量 $\mathbf{X}_1 \oplus \mathbf{X}_2$ 和 $\mathbf{X}_2 \oplus \mathbf{X}_3$ 相互独立，则由引理 3.1 可知，$\epsilon_{1,3} = 2(1/8)^2 = 1/32$。然而，我们知道事实并非如此：$\epsilon_{1,3} = 1/8$。由于 $\mathbf{X}_1 \oplus \mathbf{X}_2$ 和 $\mathbf{X}_2 \oplus \mathbf{X}_3$ 并不相互独立，所以引理 3.1 并未给出正确答案。

3.3.2 S盒的线性逼近

考虑如下一个S盒 $\pi_S:\{0,1\}^m \to \{0,1\}^n$(我们并未假定 π_S 是一个置换,甚至也未假定 $m=n$)。m重输入 $X=(x_1,\cdots,x_m)$均匀随机地从集合$\{0,1\}^m$ 中选取,这就是说每一个坐标 x_i 定义了一个随机变量 $\mathbf{X}_i$,$\mathbf{X}_i$ 取值于$\{0,1\}$,并且其偏差 $\varepsilon_i=0$。更进一步,这 m 个随机变量相互独立。n 重输出 $Y=(y_1,\cdots,y_n)$中,每一个坐标 y_i 定义了一个随机变量 $\mathbf{Y}_i$,$\mathbf{Y}_i$ 取值于$\{0,1\}$。这 n 个随机变量一般说来并不相互独立,与 $\mathbf{X}_i$ 也不相互独立。事实上,不难验证:如果 $(y_1,\cdots,y_n)\neq\pi_S(x_1,\cdots,x_m)$,则

$$\mathbf{Pr}[\mathbf{X}_1=x_1,\cdots,\mathbf{X}_m=x_m,\mathbf{Y}_1=y_1,\cdots,\mathbf{Y}_n=y_n]=0$$

如果$(y_1,\cdots,y_n)=\pi_S(x_1,\cdots,x_m)$,则

$$\mathbf{Pr}[\mathbf{X}_1=x_1,\cdots,\mathbf{X}_m=x_m,\mathbf{Y}_1=y_1,\cdots,\mathbf{Y}_n=y_n]=2^{-m}$$

后一公式成立是因为

$$\mathbf{Pr}[\mathbf{X}_1=x_1,\cdots,\mathbf{X}_m=x_m]=2^{-m}$$

并且,如果$(y_1,\cdots,y_n)=\pi_S(x_1,\cdots,x_m)$,则

$$\mathbf{Pr}[\mathbf{Y}_1=y_1,\cdots,\mathbf{Y}_n=y_n\mid\mathbf{X}_1=x_1,\cdots,\mathbf{X}_m=x_m]=1$$

现在应用上述公式,计算如下形式的随机变量的偏差就相对直接多了:

$$\mathbf{X}_{i_1}\oplus\cdots\oplus\mathbf{X}_{i_k}\oplus\mathbf{Y}_{j_1}\oplus\cdots\oplus\mathbf{Y}_{j_l}$$

如果一个这种形式的随机变量具有偏离0的偏差值,那么线性密码分析就成为可能。

下面看一个例子。

例3.2 我们仍然应用例3.1中的S盒,$\pi_S:\{0,1\}^4\to\{0,1\}^4$。在下表中我们记下了8个随机变量 $\mathbf{X}_1,\cdots,\mathbf{X}_4,\mathbf{Y}_1,\cdots,\mathbf{Y}_4$ 所有可能的取值:

$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{Y}_1$	$\mathbf{Y}_2$	$\mathbf{Y}_3$	$\mathbf{Y}_4$
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1

现在考虑随机变量 $\mathbf{X}_1 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_2$。通过计算上述表中 $\mathbf{X}_1 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_2 = 0$ 的行数，我们可以得到该随机变量取值为 0 的概率：

$$\mathbf{Pr}[\mathbf{X}_1 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_2 = 0] = \frac{1}{2}$$

因此，

$$\mathbf{Pr}[\mathbf{X}_1 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_2 = 1] = \frac{1}{2}$$

可见，该随机变量的偏差为 0。

如果我们再分析一下随机变量 $\mathbf{X}_3 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_1 \oplus \mathbf{Y}_4$，就会看到这个随机变量偏差为 -3/8。（我们建议读者验证一下这个计算。）事实上并不难确定所有 $2^8 = 256$ 个这种形式的随机变量的偏差。我们用下述记号来记录这个信息。把每一个相关的随机变量表示成下列形式：

$$\left(\bigoplus_{i=1}^{4} a_i \mathbf{X}_i\right) \oplus \left(\bigoplus_{i=1}^{4} b_i \mathbf{Y}_i\right)$$

其中 $a_i \in \{0,1\}$，$b_i \in \{0,1\}$，$i = 1,2,3,4$。为了记号的紧凑，我们把每一个二元向量 (a_1, a_2, a_3, a_4) 和 (b_1, b_2, b_3, b_4) 看做一个十六进制数字（这些数字分别叫做“输入和”与“输出和”）。这样，这 256 个随机变量里的每一个就用一对十六进制数字来表示。

作为一个例子，考虑随机变量 $\mathbf{X}_1 \oplus \mathbf{X}_4 \oplus \mathbf{Y}_2$。输入和为 (1,0,0,1)，即十六进制的 9；输出和为 (0,1,0,0)，即十六进制的 4。对于一个具有（十六进制）输入和 a 与输出和 b 的随机变量（这里 $a = (a_1, a_2, a_3, a_4)$，$b = (b_1, b_2, b_3, b_4)$，二进制表示），设 $N_L(a,b)$ 表示满足如下条件的二元 8 重组 $(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4)$ 的个数：

$$(y_1, y_2, y_3, y_4) = \pi_S(x_1, x_2, x_3, x_4)$$

及

$$\left(\bigoplus_{i=1}^{4} a_i \mathbf{X}_i\right) \oplus \left(\bigoplus_{i=1}^{4} b_i \mathbf{Y}_i\right) = 0$$

该随机变量的偏差计算公式为：$\varepsilon(a,b) = (N_L(a,b) - 8)/16$。

我们已经知道 $N_L(9,4) = 8$，因此在例 3.2 中，$\varepsilon(9,4) = 0$。包含所有 N_L 值的表叫做线性逼近表，参见图 3.2。

3.3.3　SPN 的线性密码分析

线性密码分析要求找出一组 S 盒的线性逼近，这组线性逼近能够用来导出一个完整 SPN（除最后一轮外）的线性逼近。我们仍然用例 3.1 中的 SPN 来说明整个过程。图 3.3 说明了我们将要应用的逼近的结构。图 3.3 中，带箭头的线条对应于包含在线性逼近中的随机变量。带标号的 S 盒就是在这些逼近运算中使用的 S 盒（称做逼近中的活动 S 盒（active S-boxes））。

a \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	2	6	6	10	10	6	6
4	8	10	8	6	6	4	6	8	8	6	8	10	10	4	10	8
5	8	6	6	8	6	8	12	10	6	8	4	10	8	6	6	8
6	8	10	6	12	10	8	8	10	8	6	10	12	6	8	8	6
7	8	6	8	10	10	4	10	8	6	8	10	8	12	10	8	10
8	8	8	8	8	8	8	8	8	6	10	10	6	10	6	6	2
9	8	8	6	6	8	8	6	6	4	8	6	10	8	12	10	6
A	8	12	6	10	4	8	10	6	10	10	8	8	10	10	8	8
B	8	12	8	4	12	8	12	8	8	8	8	8	8	8	8	8
C	8	6	12	6	6	8	10	8	10	8	10	12	8	10	8	6
D	8	10	10	8	6	12	8	10	4	6	10	8	10	8	8	10
E	8	10	10	8	6	4	8	10	6	8	8	6	4	10	6	8
F	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

图 3.2 线性逼近表:$N_L(a,b)$的值

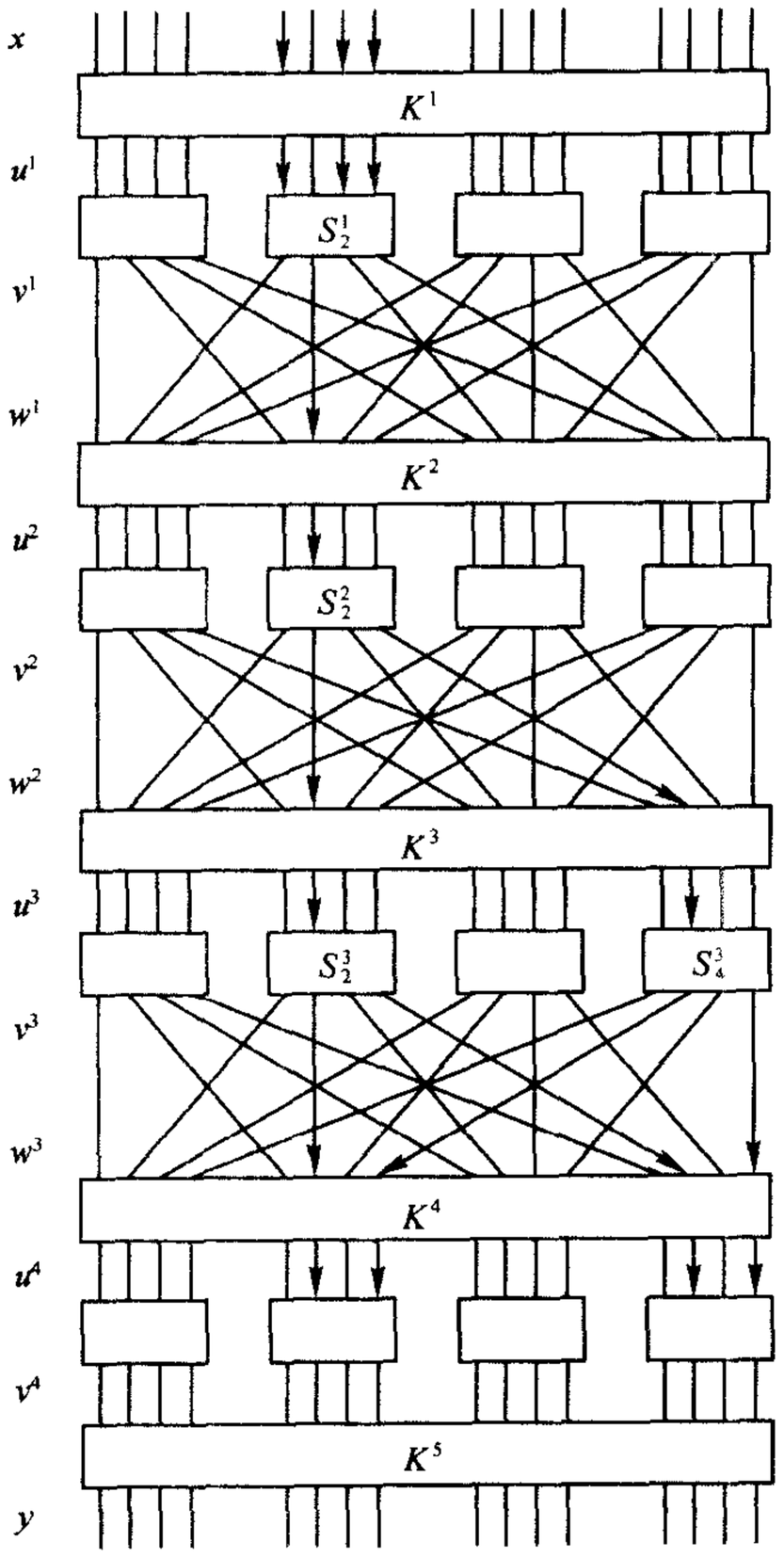

图 3.3 一个代换-置换网络的线性逼近

图中的逼近包含如下四个活动 S 盒：

- 在 S_2^1 中，随机变量 $\mathbf{T}_1 = \mathbf{U}_5^1 \oplus \mathbf{U}_7^1 \oplus \mathbf{U}_8^1 \oplus \mathbf{V}_6^1$ 具有偏差 1/4
- 在 S_2^2 中，随机变量 $\mathbf{T}_2 = \mathbf{U}_6^2 \oplus \mathbf{V}_6^2 \oplus \mathbf{V}_8^2$ 具有偏差 - 1/4
- 在 S_2^3 中，随机变量 $\mathbf{T}_3 = \mathbf{U}_6^3 \oplus \mathbf{V}_6^3 \oplus \mathbf{V}_8^3$ 具有偏差 - 1/4
- 在 S_4^3 中，随机变量 $\mathbf{T}_4 = \mathbf{U}_{14}^3 \oplus \mathbf{V}_{14}^3 \oplus \mathbf{V}_{16}^3$ 具有偏差 - 1/4

$\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4$ 这四个随机变量都具有较高的偏差绝对值，而且我们将会看到，它们的异或会消去中间变量。

如果我们假设这四个随机变量相互独立，那么我们就可用堆积引理（引理 3.1）来计算它们的异或的偏差（事实上这些随机变量并不相互独立，这意味着我们未能给这个逼近提供一个数学上的证明。然而，正如我们将要看到的，这个逼近在实际中很有效）。因此，我们假定随机变量

$$\mathbf{T}_1 \oplus \mathbf{T}_2 \oplus \mathbf{T}_3 \oplus \mathbf{T}_4$$

具有偏差 $2^3(1/4)(-1/4)^3 = -1/32$。

随机变量 $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4$ 具有以下性质：它们的异或可用明文比特、u^4 的比特（S 盒最后一轮的输入）以及密钥比特表示出来。这一点可从以下事实看出：首先，由图 3.3 易验证以下关系成立：

$$\begin{aligned}
\mathbf{T}_1 &= \mathbf{U}_5^1 \oplus \mathbf{U}_7^1 \oplus \mathbf{U}_8^1 \oplus \mathbf{V}_6^1 &&= \mathbf{X}_5 \oplus \mathbf{K}_5^1 \oplus \mathbf{X}_7 \oplus \mathbf{K}_7^1 \oplus \mathbf{X}_8 \oplus \mathbf{K}_8^1 \oplus \mathbf{V}_6^1 \\
\mathbf{T}_2 &= \mathbf{U}_6^2 \oplus \mathbf{V}_6^2 \oplus \mathbf{V}_8^2 &&= \mathbf{V}_6^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{V}_6^2 \oplus \mathbf{V}_8^2 \\
\mathbf{T}_3 &= \mathbf{U}_6^3 \oplus \mathbf{V}_6^3 \oplus \mathbf{V}_8^3 &&= \mathbf{V}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{V}_6^3 \oplus \mathbf{V}_8^3 \\
\mathbf{T}_4 &= \mathbf{U}_{14}^3 \oplus \mathbf{V}_{14}^3 \oplus \mathbf{V}_{16}^3 &&= \mathbf{V}_8^2 \oplus \mathbf{K}_{14}^3 \oplus \mathbf{V}_{14}^3 \oplus \mathbf{V}_{16}^3
\end{aligned}$$

将上述等式的右端相异或得到：

$$\mathbf{X}_5 \oplus \mathbf{X}_7 \oplus \mathbf{X}_8 \oplus \mathbf{V}_6^3 \oplus \mathbf{V}_8^3 \oplus \mathbf{V}_{14}^3 \oplus \mathbf{V}_{16}^3 \oplus \mathbf{K}_5^1 \oplus \mathbf{K}_7^1 \oplus \mathbf{K}_8^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{K}_{14}^3 \tag{3.1}$$

具有偏差 - 1/32。下一步是把上式中的 $\mathbf{V}_i^3$ 用包含 $\mathbf{U}_i^4$ 与下轮密钥比特的表达式来代替：

$$\begin{aligned}
\mathbf{V}_6^3 &= \mathbf{U}_6^4 \oplus \mathbf{K}_6^4 \\
\mathbf{V}_8^3 &= \mathbf{U}_{14}^4 \oplus \mathbf{K}_{14}^4 \\
\mathbf{V}_{14}^3 &= \mathbf{U}_8^4 \oplus \mathbf{K}_8^4 \\
\mathbf{V}_{16}^3 &= \mathbf{U}_{16}^4 \oplus \mathbf{K}_{16}^4
\end{aligned}$$

现在我们把这些式子代入（3.1）式，可得：

$$\begin{aligned}
&\mathbf{X}_5 \oplus \mathbf{X}_7 \oplus \mathbf{X}_8 \oplus \mathbf{U}_6^4 \oplus \mathbf{U}_8^4 \oplus \mathbf{U}_{14}^4 \oplus \mathbf{U}_{16}^4 \\
&\qquad \oplus \mathbf{K}_5^1 \oplus \mathbf{K}_7^1 \oplus \mathbf{K}_8^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{K}_{14}^3 \oplus \mathbf{K}_6^4 \oplus \mathbf{K}_8^4 \oplus \mathbf{K}_{14}^4 \oplus \mathbf{K}_{16}^4
\end{aligned} \tag{3.2}$$

（3.2）式仅包含明文比特、u^4 的比特以及密钥比特。假设式（3.2）中的密钥比特固定，则随机变量

$$\mathbf{K}_5^1 \oplus \mathbf{K}_7^1 \oplus \mathbf{K}_8^1 \oplus \mathbf{K}_6^2 \oplus \mathbf{K}_6^3 \oplus \mathbf{K}_{14}^3 \oplus \mathbf{K}_6^4 \oplus \mathbf{K}_8^4 \oplus \mathbf{K}_{14}^4 \oplus \mathbf{K}_{16}^4$$

具有固定的值 0 或 1。因此,随机变量

$$\mathbf{X}_5 \oplus \mathbf{X}_7 \oplus \mathbf{X}_8 \oplus \mathbf{U}_6^4 \oplus \mathbf{U}_8^4 \oplus \mathbf{U}_{14}^4 \oplus \mathbf{U}_{16}^4 \tag{3.3}$$

具有偏差 ± 1/32,这里偏差的符号取决于未知密钥比特的值。注意,随机变量(3.3)式仅包含明文比特及 u^4 的比特。(3.3)式具有偏离 0 的偏差这一事实允许我们进行 3.3 节开始时提到的线性密码攻击。

假设我们拥有用同一未知密钥 K 加密的 T 对明-密文(后面将会看到,为使攻击成功约需要 $T \approx 8\,000$ 对明-密文)。用$\mathcal{T}$来表示 T 对明-密文的集合。线性攻击将使我们获得 $K_{(2)}^5$ 和 $K_{(4)}^5$ 的 8 比特密钥,即,

$$K_5^5, K_6^5, K_7^5, K_8^5, K_{13}^5, K_{14}^5, K_{15}^5, K_{16}^5$$

这些正是与 S 盒 S_2^4 和 S_4^4 的输出相异或的 8 比特密钥。注意对这 8 比特密钥来说,共有 $2^8 = 256$ 种可能,我们把由这 8 比特密钥组成的一个二进制 8 元组叫做一个候选子密钥。

对每一个$(x, y) \in \mathcal{T}$及每一个候选子密钥,计算 y 的一个部分解密并获得 $u_{(2)}^4$ 和 $u_{(4)}^4$。然后计算

$$x_5 \oplus x_7 \oplus x_8 \oplus u_6^4 \oplus u_8^4 \oplus u_{14}^4 \oplus u_{16}^4 \tag{3.4}$$

的值。我们保持对应于这 256 个候选子密钥的 256 个计数器,每当(3.4)式取值为 0 时,就将对应于该子密钥的记数器加 1(这些计数器初始化全为 0)。

在计数过程的最后,我们希望大多数的计数器值接近于 $T/2$,而真正的候选子密钥对应的计数器具有接近于 $T/2 \pm T/32$ 之值,这有助于我们确定正确的 8 个子密钥比特。算法 3.2 给出了这个特殊的线性攻击算法。集合$\mathcal{T}$表示 T 对明-密文的集合,变量 L_1 和 L_2 取十六进制的值,置换 π_S^{-1} 对应于 S 盒的逆置换,π_S^{-1} 被用来部分地解密密文;输出 *maxkey* 包含了该攻击确定出的具有最大可能的 8 个子密钥比特。

一般来说,一个基于偏差为 ε 的线性逼近的线性攻击要想获得成功,所需要的明-密文对数目 T 要接近于 $c\varepsilon^{-2}$,c 是某个"小"的常数。将算法 3.2 实现一下就会发现:如果我们取 $T = 8000$,这个攻击通常会成功。注意,$T = 8000$ 对应于 $c \approx 8$,这是因为 $\varepsilon^{-2} = 1024$。

算法 3.2 线性攻击 LinearAttack($\mathcal{T}, T, \pi_S^{-1}$)

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y) \in \mathcal{T}$
  for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
    do
      $v_{(2)}^4 \leftarrow L_1 \oplus y_{(2)}$
      $v_{(4)}^2 \leftarrow L_2 \oplus y_{(4)}$
      $u_{(2)}^4 \leftarrow \pi_S^{-1}(v_{(12)}^4)$
      do $u_{(4)}^4 \leftarrow \pi_S^{-1}(v_{(4)}^4)$
      $z \leftarrow x_5 \oplus x_7 \oplus x_8 \oplus u_6^4 \oplus u_8^4 \oplus u_{14}^4 \oplus u_{16}^4$
      if $z = 0$
        then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$

$max \leftarrow -1$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

do { $Count[L_1, L_2] \leftarrow |Count[L_1, L_2] - T/2|$

if $Count[L_1, L_2] > max$

then { $max \leftarrow Count[L_1, L_2]$

$maxkey \leftarrow (L_1, L_2)$

output(*maxkey*)

3.4　差分密码分析

差分密码分析在许多方面与线性密码分析相似，它与线性密码分析的主要区别在于差分密码分析包含了将两个输入的异或与其相对应的两个输出的异或相比较。一般来说，我们将要考察两个二元串 x 与 x^*，它们具有固定的异或值 $x' = x \oplus x^*$。在本节中，我们将用(′)来表示两个比特串的异或。

差分密码分析是一个选择明文攻击。我们假设一个攻击者拥有大量的 4 重组 (x, x^*, y, y^*)，其中异或值 $x' = x \oplus x^*$ 是固定的。明文 x 与 x^* 用同一个密钥 K 加密分别得到密文 y 与 y^*。对这些 4 重组中的每一个，我们将应用所有可能的候选密钥来对该密码的最后一轮进行解密，对每一个候选密钥，我们计算某些状态比特的值，并确定它们的异或是否有一个确定的值(即对给定输入异或值的最可能取的值)。每当上述叙述成立时，我们就把对应于特定候选密钥的计数器加 1。在这个过程的最后，我们希望具有最高频率的候选密钥含有真正密钥这些比特的取值(同线性密码分析部分一样，我们将用一个特殊的例子来说明这个攻击)。

定义 3.1　设 $\pi_S: \{0,1\}^m \rightarrow \{0,1\}^n$ 为一个 S 盒。考虑长为 m 的有序比特串对 (x, x^*)，我们称 S 盒的输入异或为 $x \oplus x^*$，输出异或为 $\pi_S(x) \oplus \pi_S(x^*)$。注意，输出异或是一个长为 n 的比特串。

对任何 $x' \in \{0,1\}^m$，定义集合 $\Delta(x')$ 为包含所有具有输入异或值 x' 的有序对 (x, x^*)。

易见，集合 $\Delta(x')$ 包含 2^m 对，并且

$$\Delta(x') = \{(x, x \oplus x') : x \in \{0,1\}^m\}$$

对集合 $\Delta(x')$ 中的每一对，我们都能计算它们关于 S 盒的输出异或，然后我们能将所有输出异或的值列成一张结果分布表，一共有 2^m 个输出异或，它们的值分布在 2^n 个可能值之上。一个非均匀的输出分布将会是一个成功的差分攻击的基础。

例 3.3 我们仍然应用例中的 S 盒。设输入异或 $x' = 1011$,则

$$\Delta(1011) = \{(0000,1011),(0001,1010),\cdots,(1111,0100)\}$$

对 $\Delta(1011)$中的每一有序对,我们可计算 π_S 的输出异或。在下表中的每一行,均有 $x \oplus x^* = 1011$, $y = \pi_S(x)$, $y^* = \pi_S(x^*)$及 $y' = y \oplus y^*$:

x	x^*	y	y^*	y'
0000	1011	1110	1100	0010
0001	1010	0100	0110	0010
0010	1001	1101	1010	01111
0011	1000	0001	0011	0010
0100	1111	0010	0111	0101
0101	1110	1111	0000	1111
0110	1101	1011	1001	0010
0111	1100	1000	0101	1101
1000	0011	0011	0001	0010
1001	0010	1010	1101	0111
1010	0001	0110	0100	0111
1011	0000	1100	1110	0010
1100	0111	0101	1000	1101
1101	0110	1001	1011	0010
1110	0101	0000	1111	1111
1111	0100	0111	0010	0101

观察上表的最后一列,我们就获得如下的输出异或分布:

0000	0001	0010	0011	0100	0101	0110	0111
0	0	8	0	0	2	0	2

1000	1001	1010	1011	1100	1101	1110	1111
0	0	0	0	0	2	0	2

在例 3.3 中,16 个可能的输出异或中实际上只有 5 个出现,这个特殊的例子具有一个非常不均匀的分布。我们能像例 3.3 中那样,对任何可能的输入异或做这些计算。为了更方便地描述这些输出异或分布,引入如下定义。对 m 长的比特串 x'和 n 长的比特串 y',定义:

$$N_D(x',y') = |\{(x,x^*) \in \Delta(x') : \pi_S(x) \oplus \pi_S(x^*) = y'\}|$$

换言之,$N_D(x',y')$记下了输入异或等于 x',输出异或等于 y'的对数(对某一给定的 S 盒)。图 3.4 列出了针对例 3.1 中的 S 盒,所有可能的 $N_D(a',b')$值(a'和 b'分别是输入异或与输出异或的十六进制表示)。可发现,例 3.3 中计算的分布对应于图 3.4 中的 B 行。

a'	b'															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

图 3.4　差分分布表：$N_D(a', b')$的值

回想一下例 3.1 里的 SPN，其中第 r 轮第 i 个 S 盒的输入是 $u'_{(i)}$，并且 $u'_{(i)} = w'^{-1}_{(1)} \oplus K'_{(i)}$。一个输入异或可以这样计算：

$$u'_{(i)} \oplus (u'_{(i)})^* = (w'^{-1}_{(i)} \oplus K'_{(i)}) \oplus ((w'^{-1}_{(i)})^* \oplus K'_{(i)}) = w'^{-1}_{(i)} \oplus (w'^{-1}_{(i)})^*$$

因此，该输入异或并不依赖于第 r 轮的子密钥，它仅等于第 $r-1$ 轮置换的输出异或（然而，第 r 轮的输出异或当然与第 r 轮的子密钥相关）。

设 a' 表示一个输入异或，b' 表示一个输出异或。(a', b')对叫做一个差分。差分分布表中的每一项均导致了一个异或扩散率（简称扩散）。对应于差分(a', b')的扩散率 $R_p(a', b')$ 由如下式子定义：

$$R_p(a', b') = \frac{N_D(a', b')}{2^m}$$

$R_p(a', b')$也可被理解为一个条件概率：

$$\Pr[\text{输出异或} = b' \mid \text{输入异或} = a'] = R_p(a', b')$$

假设我们能在连续的轮中找到扩散率，亦即某一轮的一个差分输入异或实际上就是前一轮差分的置换输出异或，这样这些差分就组成了一个差分链。我们假设差分链中不同的扩散率相互独立（这有可能不是一个数学上有效的假设）。由这个假设，我们就能把一个差分链里不同的扩散率相乘来获得整个差分链的扩散率。

我们还是回到例 3.1 中的 SPN 来说明整个过程。图 3.5 表示了一个特殊的差分链，其中箭头用来表示输入和输出异或里的“1”比特。图 3.5 中的差分攻击使用了下述差分的扩散率，所有这些比率都可由图 3.4 推出。

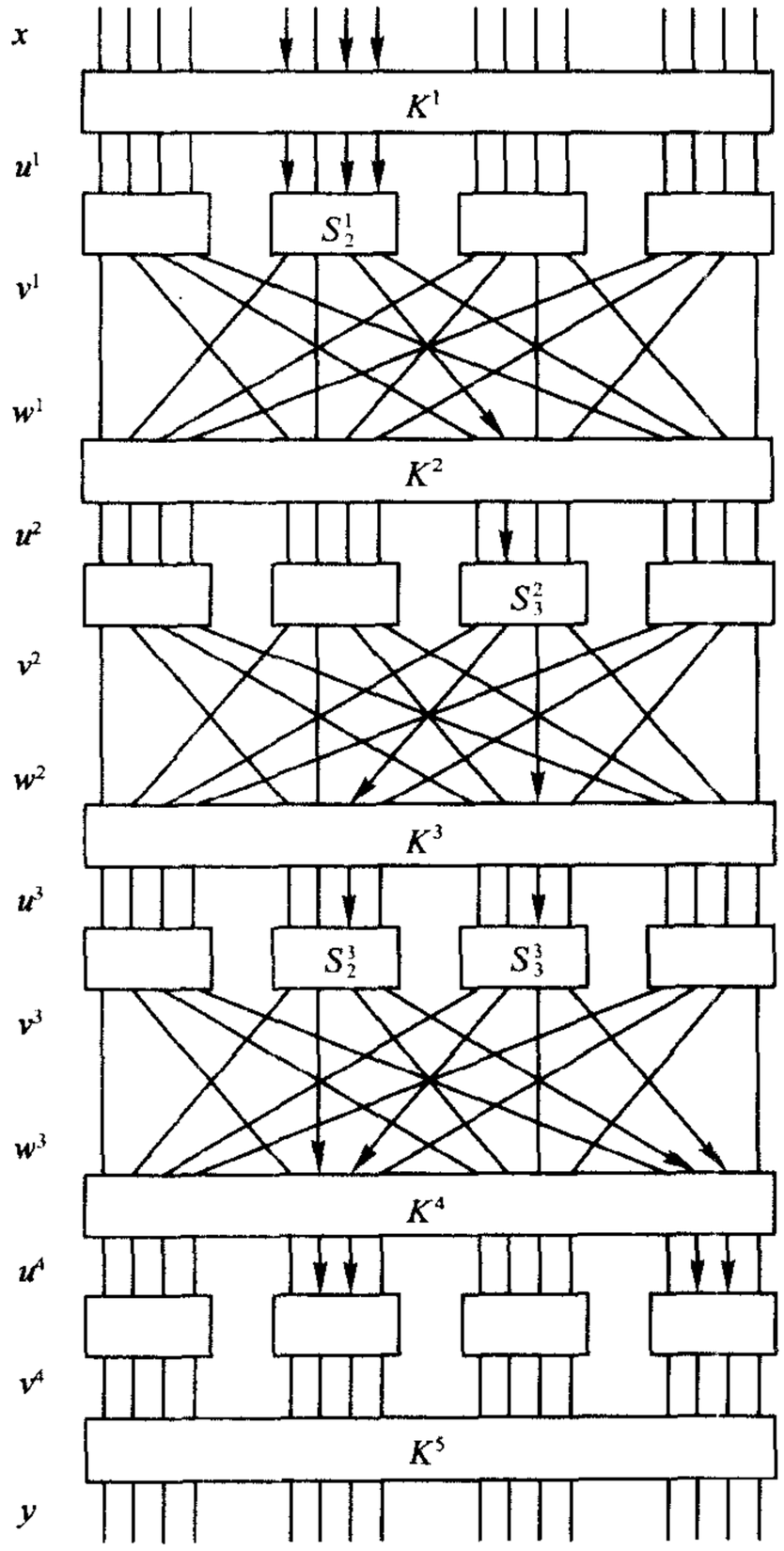

图 3.5 一个代换-置换网络的差分链

- 在 S_2^1 中，$R_p(1011,0010)=1/2$
- 在 S_3^2 中，$R_p(0100,0110)=3/8$
- 在 S_2^3 中，$R_p(0010,0101)=3/8$
- 在 S_3^3 中，$R_p(0010,0101)=3/8$

这些差分能组合成一个差分链，这样我们就获得了该 SPN 前三轮的差分链的一个扩散率：

$$R_p(0000\ 1011\ 0000\ 0000,0000\ 0101\ 0101\ 0000)=\frac{1}{2}\times\left(\frac{3}{8}\right)^3=\frac{27}{1024}$$

换言之，

$$x'=0000\ 1011\ 0000\ 0000\Rightarrow(v^3)'=0000\ 0101\ 0101\ 0000$$

以概率 27/1024 成立。然而，

$$(v^3)'=0000\ 0101\ 0101\ 0000\Leftrightarrow(u^4)'=0000\ 0110\ 0000\ 0110$$

因此，

$$x' = 0000\ 1011\ 0000\ 0000 \Leftrightarrow (u^4)' = 0000\ 0110\ 0000\ 0110$$

以概率 27/1024 成立。注意$(u^4)'$就是最后一轮 S 盒两个输入的异或。

现在,基于本节开始部分的非正式描述,我们可以对这个特殊的例子给出一个算法,参见算法 3.3。这个算法的输入和输出与线性攻击的算法相似。主要区别是:在差分攻击中,$\mathcal{T}$是 4 重组(x, x^*, y, y^*)组成的集合,其中差分值 x'是固定的。

算法 3.3 利用了一种叫做过滤的操作。使得差分成立的 4 重组(x, x^*, y, y^*)叫做一个正确对(right pairs),正是这些正确对使得我们能够确定相关的密钥比特(那些非正确对基本上产生"随机噪声",而不能提供任何有用信息)。一个正确对满足

$$(u^4_{(1)})' = (u^4_{(3)})' = 0000$$

因此,一个正确对必须满足 $y_{(1)} = (y_{(1)})^*$ 和 $y_{(3)} = (y_{(3)})^*$。如果 4 重组(x, x^*, y, y^*)不满足这些条件,那么它就不是一个正确对,就必须丢弃它。这个过滤操作能提高该攻击的效率。

当 4 重组(x, x^*, y, y^*)的数量 T 接近$c\varepsilon^{-1}$时,一个基于扩散率为 ε 的差分链的差分攻击一般会成功,其中 c 是一个"小"的常数。将算法 3.3 实现一下就会发现:如果我们取 T 的值在 50 ~ 100 之间,该攻击通常就会成功。在这个例子中,$\varepsilon^{-1} \approx 38$。

算法 3.3　差分攻击 Differential Attack$(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y, x^*, y^*) \in \mathcal{T}$
  do **if** $(y_{(1)} = (y_{(1)})^*)$ **and** $(y_{(3)} = (y_{(3)})^*)$
    then **for** $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
      do $v^4_{(2)} \leftarrow L_1 \oplus y_{(2)}$
        $v^4_{(4)} \leftarrow L_2 \oplus y_{(4)}$
        $u^4_{(2)} \leftarrow \pi_S^{-1}(v^4_{(2)})$
        $u^4_{(4)} \leftarrow \pi_S^{-1}(v^4_{(4)})$
        $(v^4_{(2)})^* \leftarrow L_1 \oplus (y_{(2)})^*$
        $(v^4_{(4)})^* \leftarrow L_2 \oplus (y_{(4)})^*$
        $(u^2_{(2)})^* \leftarrow \pi_S^{-1}((v^4_{(2)})^*)$
        $(u^4_{(4)})^* \leftarrow \pi_S^{-1}((v^4_{(4)})^*)$
        $(u^4_{(2)})' \leftarrow u^4_{(2)} \oplus (u^{\prime}_{(\cdot)})^*$
        $(u^4_{(4)})' \leftarrow u^4_{(4)} \oplus (u^4_{(4)})^*$
        if $((u^4_{(2)})' = 0110)$ **and** $((u^4_{(4)})' = 0110)$
          then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)

$$\textbf{do}\begin{cases}\textbf{if}\ Count[L_1, L_2] > max \\ \quad \textbf{then}\begin{cases} max \leftarrow Count[L_1, L_2] \\ maxkey \leftarrow (L_1, L_2) \end{cases}\end{cases}$$

output(*maxkey*)

3.5 数据加密标准

1973 年 5 月 15 日,美国国家标准局(现在是美国国家标准技术研究所,即 NIST)在联邦记录中公开征集密码体制,这一举措最终导致了数据加密标准(DES)的出现,它曾经成为世界上最广泛使用的密码体制。DES 由 IBM 开发,它是对早期被称为 Lucifer 体制的改进。DES 在 1975 年 3 月 17 日首次在联邦记录中公布,在经过大量的公开讨论后,1977 年 2 月 15 日 DES 被采纳作为"非密级"应用的一个标准。最初预期 DES 作为一个标准只能使用 10 ~ 15 年;然而,事实证明 DES 要长寿得多。在其被采用后,大约每隔 5 年被评审一次。DES 的最后一次评审在 1999 年 1 月;在当时,一个 DES 的替代品,高级加密标准(Advanced Encryption Standard),已经开始使用了(参见 3.6 节)。

3.5.1 DES 的描述

1977 年 1 月 15 日的联邦信息处理标准版 46 中(FIPS PUB46)给出了 DES 的完整描述。DES 是一种特殊类型的迭代密码,叫做 Feistel 型密码。现在我们描述一下 Feistel 型密码的基本形式,我们仍然使用 3.1 节中的术语。在一个 Feistel 型密码中,每一个状态 u^i 被分成相同长度的两部分,L^i 和 R^i。轮函数 g 具有以下形式:

$$g(L^{i-1}, R^{i-1}, K^i) = (L^i, R^i)$$

其中,

$$L^i = R^{i-1}$$
$$R^i = L^{i-1} \oplus f(R^{i-1}, K^i)$$

我们注意到,函数 f 并不需要满足任何单射条件,这是因为一个 Feistel 型轮函数肯定是可逆的,给定轮密钥,就有:

$$L^{i-1} = R^i \oplus f(L^i, K^i)$$
$$R^{i-1} = L^i$$

DES 是一个 16 轮的 Feistel 型密码,它的分组长度为 64,用一个 56 比特的密钥来加密一个 64 比特的明文串,并获得一个 64 比特的密文串。在进行 16 轮加密之前,先对明文做一个固定的初始置换 IP(initial permutation),写做 $IP(x) = L^0 R^0$。在 16 轮加密之后,对比特串 $R^{16} L^{16}$ 做逆置换 IP^{-1} 来给出密文 y,即 $y = IP^{-1}(R^{16} L^{16})$。(注意,在使用 IP^{-1} 之前,要交换 L^{16} 和 R^{16}。)IP

和 IP^{-1} 的使用并没有任何密码学上的意义，所以在讨论 DES 的安全性时常常忽略它们。DES 的一轮加密如图 3.6 所示。

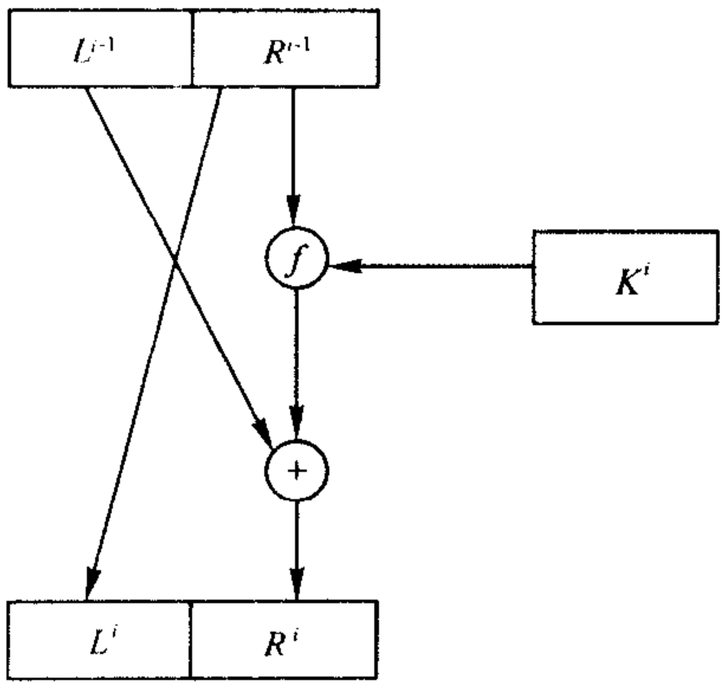

图 3.6　一轮 DES 加密

每一个 L^i 和 R^i 都是 32 比特长。函数

$$f:\{0,1\}^{32}\times\{0,1\}^{48}\rightarrow\{0,1\}^{32}$$

的输入是一个 32 比特的串（当前状态的右半部）和轮密钥。密钥编排方案 $(K^1,K^2,\cdots,K^{16})$ 由 16 个 48 比特的轮密钥组成，这些轮密钥由 56 比特的种子密钥 K 导出。每一个 K^i 都是由 K 置换选择而来。图 3.7 给出了函数 f。它主要包含一个应用 S 盒的代换以及其后跟随的一个固定置换 P。

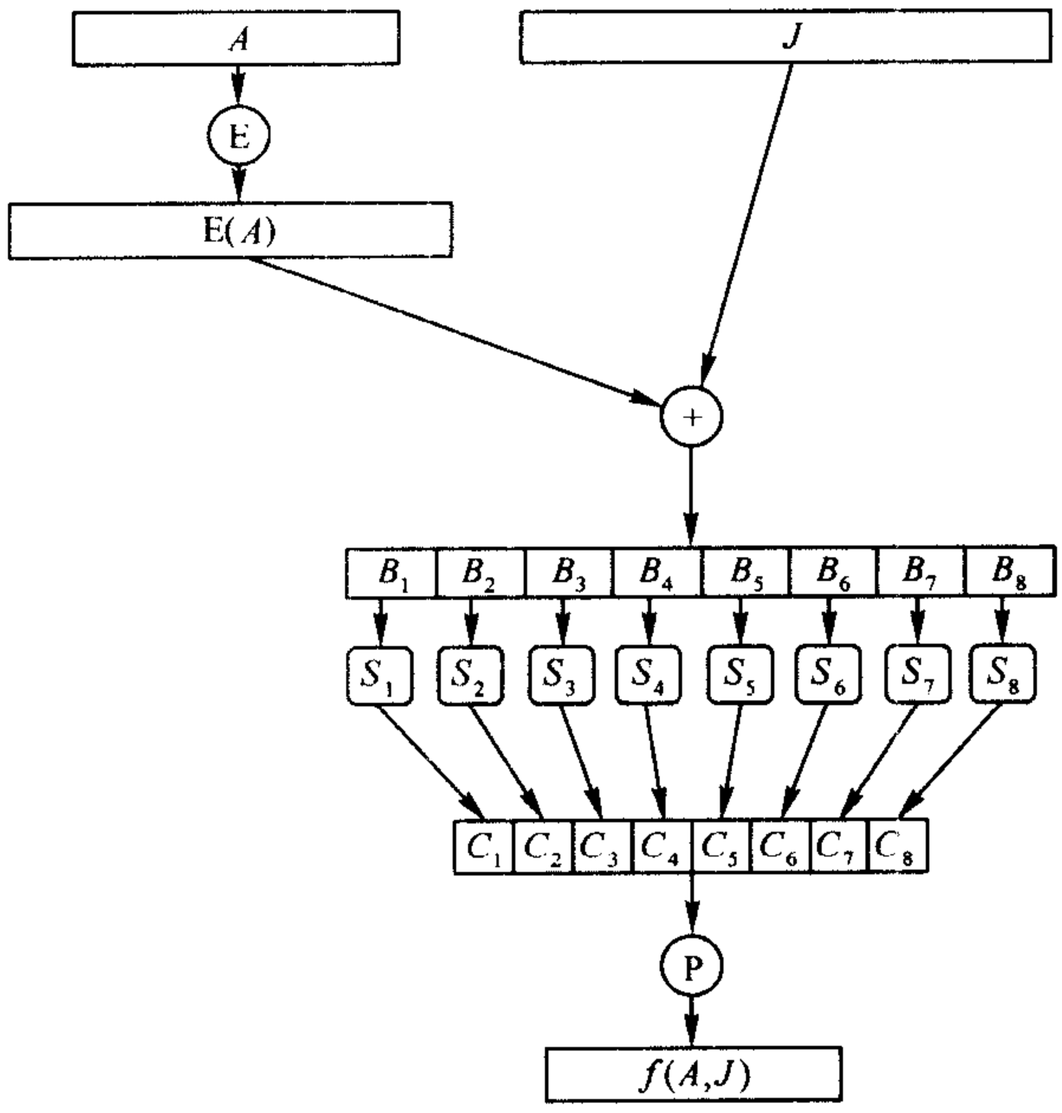

图 3.7　DES 的 f 函数

设 f 的第一个自变量是 A，第二个自变量是 J，计算 $f(A,J)$ 的过程如下。

1. 首先根据一个固定的扩展函数 E,将 A 扩展成一个长度为 48 比特的串。E(A)包含经过适当置换后的 A 的 32 比特,其中有 16 比特出现两次。

2. 计算 E(A)$\oplus J$,并且将结果写做 8 个 6 比特串的并联 $B = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$。

3. 使用 8 个 S 盒 $S_1, \cdots, S_8$。每一个 S 盒

$$S_i : \{0,1\}^6 \rightarrow (0,1)^4$$

把 6 比特映射为 4 比特,一般用一个 4×16 的矩阵来描述,它的元素来自整数 $0, \cdots, 15$。给定一个长度为 6 的比特串 $B_j = b_1 b_2 b_3 b_4 b_5 b_6$,可通过如下步骤计算 $S_j(B_j)$:用 $b_1 b_6$ 两比特决定 S_j 某一行 $r(0 \leqslant r \leqslant 3)$的二进制表示,用 $b_2 b_3 b_4 b_5$ 四比特决定 S_j 某一列 $c(0 \leqslant c \leqslant 15)$的二进制表示,则 $S_j(B_j)$被定义为写做二进制的 4 比特串 $S_j(r,c)$,这样对 $1 \leqslant j \leqslant 8$,我们可以计算 $C_j = S_j(B_j)$。

4. 根据置换 P,对 32 比特的串 $C = C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8$ 做置换,所得结果 P(C)就是 $f(A, J)$。

为了参考方便,下面列出了 8 个 S 盒。

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

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

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	1	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

例 3.4　我们来说明应用上述一般表述如何来计算一个 S 盒的输出。考虑 S 盒 S_1，并设其输入为 6 重 101000，第一个和最后一个比特是 10，它代表整数 2。中间的 4 个比特是 0100，它代表整数 4。S_1 的标号为 2 的行是其第三行(这是因为行标号从 0 开始)；同样，标号为 4 的列是其第五列。可见 S_1 的标号为行 2、列 4 的项是 13，二进制表示为 1101，因此，1101 就是 S 盒 S_1 在输入为 101000 时的输出。

DES 的 S 盒当然不是置换，这是因为可能的输入总数(64)超过了可能的输出总数(16)。然而可以证明这 8 个 S 盒中的每一个的每一行都是整数 0，…，15 的一个置换。这一性质正是为了防止某些类型的密码攻击而采取的 S 盒设计的若干原则之一。

下面的表给出了扩展去数 E。

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

给定一个长为32的比特串 $A=(a_1,a_2,\cdots,a_{32})$,E($A$)即为下列的长为48的比特串:

$$\mathrm{E}(A)=(a_{32},a_1,a_2,a_3,a_4,a_5,a_4,\cdots,a_{31},a_{32},a_1)$$

置换P如下:

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

将比特串记为 $C=(c_1,c_2,\cdots,c_{32})$,则置换后输出的比特串P($C$)如下:

$$\mathrm{P}(C)=(c_{16},c_7,c_{20},c_{21},c_{29},\cdots,c_{11},c_4,c_{25})$$

3.5.2 DES的分析

在DES作为一个标准被提出时,曾出现过许多的批评,其中之一就是针对S盒的。DES里的所有计算,除去S盒,全是线性的,也就是说,计算两个输出的异或与先将两个对应输入异或再计算其输出是相同的。作为非线性部件,S盒对密码体制的安全性至关重要(在第1章里我们都看到了线性密码体制,如希尔密码是如何被一个已知明文攻击简单攻破的)。在DES刚提出时,就有人怀疑S盒里隐藏了"陷门(trapdoors)",而美国国家安全局能够轻易地解密消息,同时还虚假地宣称DES是"安全"的。当然,无法否定这样一个猜测,然而到目前为止,并没有任何证据能证明DES里的确存在陷门。

事实上,后来表明DES里的S盒是被设计成能够防止某些类型的攻击的。在20世纪90年代初,Biham与Shamir发现差分密码分析(在3.4节已经讨论过)时,美国国家安全局就已承认某些未公布的S盒设计原则正是为了使得差分密码分析变得不可行。事实上,差分密码分析在DES最初被研发时就已为IBM的研究者所知,但这种方法却被保密了将近20年,直到

Biham 与 Shamir又独立地发现了这种攻击。

对 DES 最中肯的批评是，密钥空间的规模 2^{56} 对实际安全而言确实是太小了。DES 的前身，IBM 的 Lucifer 密码体制具有 128 比特的密钥长度。DES 的最初提案也有 64 比特的密钥长度，但后来被减少到 56 比特。IBM 声称，这个减少的原因是必须在密钥中包含 8 位奇偶校验位，这就意味着 64 比特的存储只能包含一个 56 比特的密钥。

早在 20 世纪 70 年代，就有人建议建造一台特殊目的的机器来实施已知明文攻击。这台机器本质上是对密钥进行穷尽搜索，即给定一个 64 比特的明文 x 和相应的密文 y，实验每一个可能的密钥，直到找到一个密钥 K 使得 $e_K(x) = y$（注意，可能有不止一个这样的密钥 K）。在 1977 年，Diffie 与 Hellman 就建议制造一个每秒能实验 10^6 个密钥的 VLSI 芯片，一个具有 10^6 个这样的芯片的机器能在大约一天的时间里搜索完整个密钥空间。他们估计在当时建造这样一台机器需要 20 000 000 美元。

后来，在 CRYPTO'93 的自由演讲会上，Michael Wiener 给出了一个非常详细的 DES 密钥搜索机的设计方案。这台机器基于一个串行的密钥搜索芯片，它能同时完成 16 次加密。这个芯片一秒钟能测试 5×10^7 个密钥，用 1993 年的技术制造这样一个芯片要 10.50 美元。制造一个包含 5760 个芯片的主机需要 100 000 美元。平均起来，这样一台机器能在 1.5 天内找到一个 DES 密钥。一台使用十个主机的机器将需要 1 000 000 美元，但能把平均搜索时间缩短到 3.5 小时。

Wiener 的设计从未被附诸实施，但 1998 年电子先驱者基金会（Electronic Frontier Foundation）制造了一台耗资 250 000 美元的密钥搜索机。这台叫做“DES 破译者”的计算机包含 1536 个芯片，并能每秒搜索 880 亿个密钥。在 1998 年 7 月，它成功地在 56 个小时里找到了 DES 密钥，从而赢得了 RSA 实验室“DES Challenge Ⅱ-2”挑战赛的胜利。在 1999 年 1 月，在遍布全世界的 100 000 台计算机（被称做分布式网络 distributed，net）的协同工作下，“DES 破译者”又获得了 RSA 实验室“DES Challenge Ⅲ”的优胜。这次的协同工作在 22 小时 15 分钟里找到了 DES 密钥，每秒实验超过 2450 亿个密钥。

除去穷尽密钥搜索，DES 的另外两种最重要的密码攻击是差分密码分析和线性密码分析（对于 SPN，这两个攻击分别在 3.3 节和 3.4 节做了描述）。对 DES 而言，线性攻击更有效。在 1994 年，一个实际的线性密码分析由其发明者 Matsui 提出。这是一个使用 2^{43} 对明-密文的已知明文攻击，所有这些明-密文对都用同一个未知密钥加密。他用了 40 天来产生这 2^{43} 对明-密文，又用了 10 天来找到密钥。这个密码分析并未对 DES 的安全性产生实际影响，由于这个攻击需要数目极端庞大的明-密文对，在现实世界中一个敌手很难积攒下用同一密钥加密的如此众多的明-密文对。

3.6　高级加密标准

1997 年 1 月 2 日，NIST（National Institute of Standards and Technology）开始了遴选 DES 替代者的工作。该替代者称为高级加密标准，即 AES。1997 年 9 月 12 日发布了征集算法的正式公告，要求 AES 具有 128 比特的分组长度，并支持 128、192 和 256 比特的密钥长度，而且要求 AES 要能在全世界范围内免费得到。

在至1998年6月15日提交的21个算法里,有15个满足所有的必备条件并被接纳为AES的候选算法。NIST在1998年8月20日的"第一次AES候选大会"上宣布了15个AES的候选算法。1999年3月举行了"第二次AES候选大会",之后,在1999年8月,5个候选算法入围了最后决赛:MARS、RC6、Rijndael、Serpent和Twofish。

2000年4月举行了"第三次AES候选大会"。2000年10月2日,Rijndael被选择为高级加密标准。在2001年2月28日,NIST宣布关于AES的联邦信息处理标准的草案可供公众讨论。2001年11月26日,AES被采纳为一个标准,并在2001年12月4日的联邦记录中作为FIPS 197公布。

AES的遴选过程以其公开性和国际性闻名。三次候选算法大会和官方请求公众评审为候选算法意见的反馈、公众讨论与分析提供了足够的机会,而且这一过程为置身其中的每一个人所称道。15个AES候选算法的作者代表着不同国家:澳大利亚、比利时、加拿大、哥斯达黎加、法国、德国、以色列、日本、韩国、挪威、英国及美国,这正表明了AES的国际性。最终选做AES的Rijndael就是由两位比利时研究者Daemen和Rijmen提出的。另一个有趣的变化是"第二次AES候选大会"在美国之外的意大利罗马举行。

AES的候选算法根据以下三条主要原则进行评判:

- 安全性
- 代价
- 算法与实现特性

其中,算法的"安全性"无疑是最重要的,如果一个算法被发现不安全就不会再被考虑。"代价"指的是各种实现的计算效率(速度和存储需求),包括软件实现、硬件实现和智能卡实现。"算法与实现特性"包括算法的灵活性、简洁性及其他因素。最后,五个入围最终决赛的算法都被认为是安全的。Rijndael之所以最后当选是由于它集安全性、性能、效率、可实现性及灵活性于一体,被认为优于其他四个决赛者。

3.6.1 AES的描述

如上所述,AES具有128比特的分组长度,三种可选的密钥长度,即128比特、192比特和256比特。AES是一个迭代型密码;轮数Nr依赖于密钥长度。如果密钥长度为128比特,则Nr = 10;如果密钥长度为192比特,则Nr = 12;如果密钥长度为256比特,则Nr = 14。

首先,我们给出一个AES的总体描述。该算法的执行过程如下:

1. 给定一个明文 x,将State初始化为 x,并进行AddRoundKey操作,将RoundKey与State异或。

2. 对前Nr - 1轮中的每一轮,用S盒对进行一次代换操作,称为SubBytes;对State做一置换ShiftRows;再对State做一次操作MixColumns;然后进行AddRoundKey操作。

3. 依次进行SubBytes、ShiftRows和AddRoundKey操作。

4. 将State定义为密文 y。

从上述总体描述中,我们可以看到AES与3.2节中讨论的SPN在许多方面相似。在这两个密码体制的每一轮中,都要进行轮密钥混合、代换和置换。这两个密码都包括白化过程。AES更甚,它还在每一轮中包括一个额外的线性变换(MixColumns)。

我们现在给出 AES 中用到的所有操作的详细描述。我们将描述 State 的结构，讨论密钥编排方案的构造。所有 AES 中的操作都是以字节为基础的，所有用到的变量都由适当数量的字节组成。明文 x 包括 16 个字节 x_0，…，x_{15}。State 用如下的 4×4 字节矩阵表示：

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

首先，State 被定义为由明文 x 的 16 个字节组成，即：

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

←

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

我们将用十六进制来代表一个字节的内容，这样每一个字节将含有两个十六进制数字。SubBytes操作使用一个 S 盒 π_S 对State的每一个字节都进行一个独立的代换，其中 π_S 是 $\{0,1\}^8$ 的一个置换。为了给出这个 π_S，我们用十六进制来表示字节。π_S 被描述为一个 16 × 16 的矩阵，其中行号与列号都用十六进制数字表示，行标号为 X、列标号为 Y 的项是 $\pi_S(XY)$，图 3.8 给出了 π_S 的矩阵表示。

	Y															
X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	OC	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

图 3.8　AES 的 S 盒

与 DES 的 S 盒相比,AES 的 S 盒能进行代数上的定义,而不像 DES 的 S 盒那样是明显的"随机"代换。AES 的 S 盒的代数公式包含了有限域上的操作(有限域在 6.4 节中有详细的讨论)。我们下面的描述假定读者已经熟悉有限域的知识(其他读者可以首先阅读 6.4 节,或跳过这段描述)。置换 π_S 包含有限域

$$\mathbb{F}_{2^8} = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$$

中的操作。设 FieldInv 表示求一个域元素的乘法逆;BinaryToField 把一个字节变换成一个域元素;FieldToBinary 进行相反的变换。这个变换以一种明显的方式进行:域元素

$$\sum_{i=0}^{7} a_i x^i$$

对应于字节

$$a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$$

这里 $a_i \in \mathbb{Z}_2, 0 \leqslant i \leqslant 7$。置换 π_S 根据算法 3.4 定义,在这个算法里,8 个输入比特 $a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$ 被 8 个输出比特 $b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$ 所代替。

算法 3.4 SubBytes($a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$)

external FieldInv, BinaryToField, FieldToBinary

$z \leftarrow$ BinaryToField($a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$)

if $z \neq 0$

then $z \leftarrow$ FieldInv(z)

($a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$) $\leftarrow$ FieldToBinary(z)

($c_7\, c_6\, c_5\, c_4\, c_3\, c_2\, c_1\, c_0$) $\leftarrow$ (01100011)

comment:在下面的循环中,所有下标都要经过模 8 约简

for $i \leftarrow 0$ **to** 7

do $b_i \leftarrow (a_i + a_{i+4} + a_{i+5} + a_{i+6} + a_{i+7} + c_i) \bmod 2$

return($b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$)

例 3.5 我们用一个小例子来说明算法 3.4,这里还包含了到十六进制的变换。假设我们以十六进制的 53 开始。在二进制下就是 01010011,它表示域元素为:

$$x^6 + x^4 + x + 1$$

它的乘法逆元素(在有限域 $\mathbf{F}_{2^8}$ 中)为:

$$x^7 + x^6 + x^3 + x$$

因此,在二进制下,我们有:

$$(a_7\, a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0) = (11001010)$$

下面我们计算

$$\begin{aligned} b_0 &= a_0 + a_4 + a_5 + a_6 + a_7 + c_0 \bmod 2 \\ &= 0 + 0 + 0 + 1 + 1 + 1 \bmod 2 \\ &= 1 \\ b_1 &= a_1 + a_5 + a_6 + a_7 + a_0 + c_1 \bmod 2 \\ &= 1 + 0 + 1 + 1 + 0 + 1 \bmod 2 \\ &= 0 \end{aligned}$$

等等。结果是：

$$(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) = (11101101)$$

以十六进制表示就是 ED。

上述计算可由图 3.8 验证，在行标号为 5 列标号为 3 的项正是 ED。

对 State 的 ShiftRows 操作如下面所示：

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

←

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

MixColumns 操作对 State 四列中的每一列进行操作，算法 3.5 给出了它的过程。State 的每一列都被一个新列替代，这个新列由原列乘上域$\mathbb{F}_{2^8}$中的元素组成的矩阵而来（这里的“乘”是指域$\mathbb{F}_{2^8}$中的乘法。我们假设外部进程 FieldMult 以两个域元素为输入，并能计算它们在域中的乘积）。域加法就是分量的模 2 加（即对应比特串的异或）。在算法 3.5 中，这项操作用“$\oplus$”表示。

算法 3.5　MixColumn(c)

external FieldMult, BinaryToField, FieldToBinary

for $i \leftarrow 0$ **to** 3

　do $t_i \leftarrow$ BinaryToField($S_{i,c}$)

$u_0 \leftarrow$ FieldMult(x, t_0) $\oplus$ FieldMult($x+1$, t_1) $\oplus t_2 \oplus t_3$

$u_1 \leftarrow$ FieldMult(x, t_1) $\oplus$ FieldMult($x+1$, t_2) $\oplus t_3 \oplus t_0$

$u_2 \leftarrow$ FieldMult(x, t_2) $\oplus$ FieldMult($x+1$, t_3) $\oplus t_0 \oplus t_1$

$u_3 \leftarrow$ FieldMult(x, t_3) $\oplus$ FieldMult($x+1$, t_0) $\oplus t_1 \oplus t_2$

for $i \leftarrow 0$ **to** 3

　do $S_{i,c} \leftarrow$ FieldToBinary(u_i)

下面我们来讨论 AES 的密钥编排方案。我们将描述如何用 128 比特的种子密钥为 10 轮版本的 AES 构造密钥编排方案（12 轮和 14 轮版本的密钥编排方案同 10 轮版本的类似，但在密钥编排算法上稍有不同)。对 10 轮版本的 AES，我们需要 11 个轮密钥，每个轮密钥由 16 个字节组成。密钥编排算法是以字为基础的（一个字由 4 个字节组成，即 32 比特)。因此，每一个轮密钥由 4 个字组成。轮密钥的并联叫做扩展密钥，共包含 44 个字，表示为 $w[0],\cdots,w[43]$,这里每一个 $w[i]$都是一个字。扩展密钥用 KeyExpansion 操作，算法 3.6 给出了该操作。

算法 3.6 KeyExpansion(*key*)

external RotWord, SubWord

$RCon[1] \leftarrow 01000000$

$RCon[2] \leftarrow 02000000$

$RCon[3] \leftarrow 04000000$

$RCon[4] \leftarrow 08000000$

$RCon[5] \leftarrow 10000000$

$RCon[6] \leftarrow 20000000$

$RCon[7] \leftarrow 40000000$

$RCon[8] \leftarrow 80000000$

$RCon[9] \leftarrow 1B000000$

$RCon[10] \leftarrow 36000000$

for $i \leftarrow 0$ **to** 3

  do $w[i] \leftarrow (key[4i], key[4i+1], key[4i+2], key[4i+3])$

for $i \leftarrow 4$ **to** 43

do
- $temp \leftarrow w[i-1]$
- **if** $i \equiv 0 \pmod 4$
  - **then** $temp \leftarrow \text{SubWord}(\text{RotWord}(temp)) \oplus RCon[i/4]$
- $w[i] \leftarrow w[i-4] \oplus temp$

return $(w[0],\cdots,w[43])$

算法 3.6 的输入是 128 比特的密钥 *key*,它被处理成一个字节的数组:$key[0],\cdots,key[15]$;输出是字的数组 w,如上所述。KeyExpansion包括其他两个操作RotWord和SubWord。RotWord(B_0,B_1,B_2,B_3)对四个字节 B_0,B_1,B_2,B_3 进行循环移位,即

$$\text{RotWord}(B_0,B_1,B_2,B_3)=(B_1,B_2,B_3,B_0)$$

SubWord(B_0,B_1,B_2,B_3)对四个字节 B_0,B_1,B_2,B_3 使用 AES 的 S 盒,即

$$\text{SubWord}(B_0,B_1,B_2,B_3)=(B_0',B_1',B_2',B_3')$$

其中 $B_i' = \text{SubBytes}(B_i), i = 0,1,2,3$。$RCon$ 是一个 10 个字的数组 $RCon[1], \cdots, RCon[10]$。这些都是定义在算法 3.6 中的以十六进制表示的常数。

至此,我们已把 AES 的加密所需的所有操作描述完毕。为了解密,只需将所有操作逆序进行,并逆序使用密钥编排方案即可。另外,操作 ShiftRows、SubBytes 及 MixColumns 均需用它们的逆操作来代替(操作 AddRoundkey 的逆操作就是它自己)。构造一个 AES 的"等价逆密码"是可能的,这个"等价逆密码"能通过一系列的逆操作来实现 AES 的解密,这些逆操作将以与 AES 加密相同的顺序进行,这样做据说可以提高实现效率。

3.6.2　AES 的分析

显然,对所有已知攻击而言,AES 是安全的。它的设计的各个方面融合了各种特色,从而为抵抗各种攻击提供了安全性。例如,S 盒构造中有限域逆操作的使用导致了线性逼近和差分分布表中的各项趋近于均匀分布。这就为抵御差分和线性攻击提供了安全性。类似地,线性变换 MixColumns 使得找到包含"较少"活动 S 盒的差分和线性攻击成为不可能事件(设计者将这一特色称为宽轨道策略)。显然,对 AES 现在还不存在快于穷尽密钥搜索的攻击。即使是对 AES 减少迭代轮数的各种变体而言"最好"的攻击,也对 10 轮的 AES 无效。

3.7　工作模式

1980 年 12 月,FIPS 81 标准化了为 DES 开发的四种工作模式。这些工作模式可对任何分组密码。现在,AES 的工作模式正在研发,这些 AES 的工作模式可能会包括以前 DES 的工作模式,还有可能包括新加的工作模式。

现在我们简单地讨论一下 DES 的四种工作模式。只要将分组长度由 64 改为 128,即可把这几种模式应用于 AES。下面是 DES 的四种工作模式:

- 电码本模式(ECB 模式)
- 密码反馈模式(CFB 模式)
- 密码分组链接模式(CBC 模式)
- 输出反馈模式(OFB 模式)

ECB 模式就是一个分组密码的直接使用:给定一个 64 比特的明文分组序列 $x_1 x_2 \cdots$,每一个 x_i 都用同一密钥 K 来加密,产生密文分组序列 $y_1 y_2 \cdots$。

在 CBC 模式中,每一个密文分组 y_i 在用密钥 K 加密之前,都要先跟下一个明文分组 x_{i+1} 相异或。严格地说,我们从一个 64 比特的初始向量 IV 开始,定义 $y_0 = \text{IV}$,然后用下述公式构造 $y_1, y_2, \cdots$

$$y_i = e_K(y_{i-1} \oplus x_i)$$

$i \geqslant 1$。图 3.9 给出了 CBC 模式的描述。

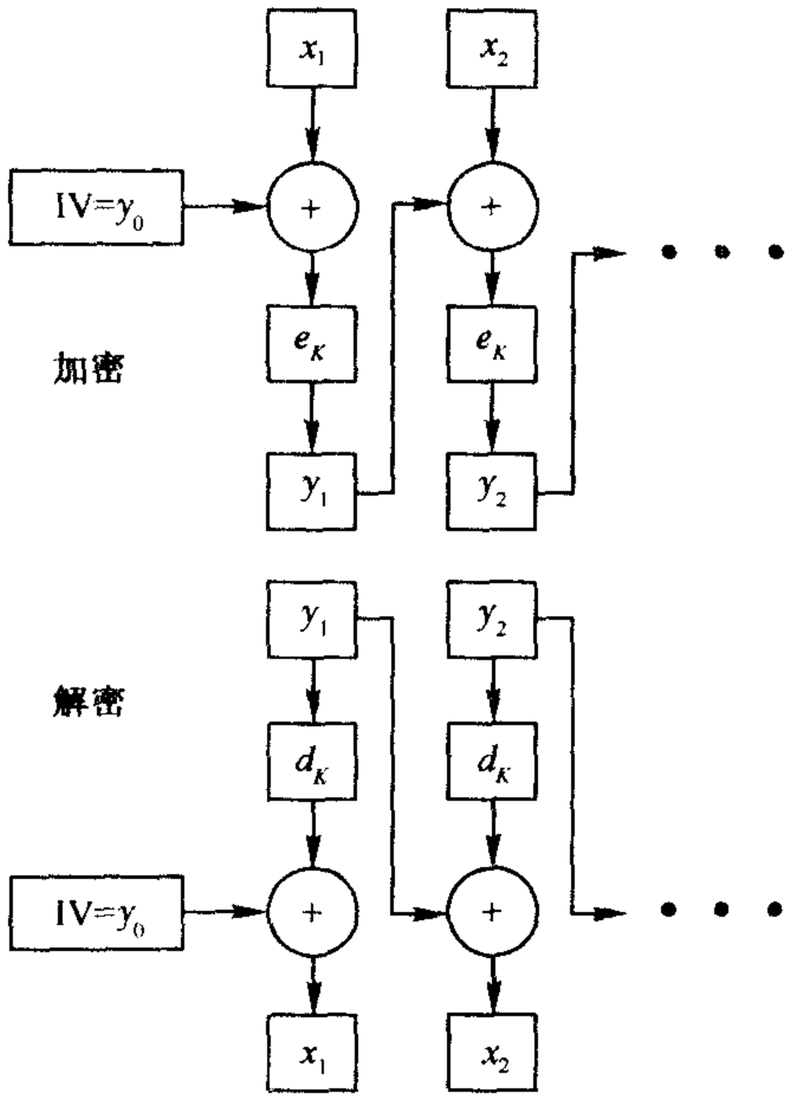

图 3.9 CBC 模式

在 OFB 和 CFB 模式中都要产生一个密钥流,然后将其与明文相异或(即像流密码一样工作,参见 1.1.7 小节)。OFB 模式实际上就是一个同步流密码:密钥流由反复加密一个 64 比特的初始向量 IV 而得。我们定义 $z_0 = \text{IV}$,然后用下述公式计算密钥流 $z_1 z_2 \cdots$:

$$z_i = e_K(z_{i-1})$$

$i \geqslant 1$。明文分组序列 $x_1 x_2 \cdots$,通过计算

$$y_i = x_i \oplus z_i$$

$i \geqslant 1$,来加密。

在 CFB 模式中,我们由 $y_0 = \text{IV}$(一个 64 比特的初始向量)开始,然后通过加密以前的密文分组来产生密钥流元素 z_i,即

$$z_i = e_K(y_{i-1})$$

$i \geqslant 1$,同 OFB 模式里相同,定义

$$y_i = x_i \oplus z_i$$

$i \geqslant 1$。图 3.10 给出了 CFB 模式的描述(注意,对于 OFB 和 CFB 模式,加密函数 e_K 既用于加密也用于解密)。

其他模式还有称做 k 比特反馈模式的 OFB 和 CFB 模式的变体($1 \leqslant k \leqslant 64$)。上面我们已经描述了 64 比特的反馈模式。在实际中,如需一次加密一个比特或一个字节,则经常使用 1 比特和 8 比特的反馈模式。

这四种工作模式各有其优缺点。ECB 模式的一个明显缺点是加密相同的明文分组将产生相同的密文分组。在消息组选自一个“低熵”明文空间的情况下,这尤其是一个严重的问题。

例如,如果一个明文分组由 64 个 0 或 64 个 1 组成,则 ECB 加密模式基本上没什么用。

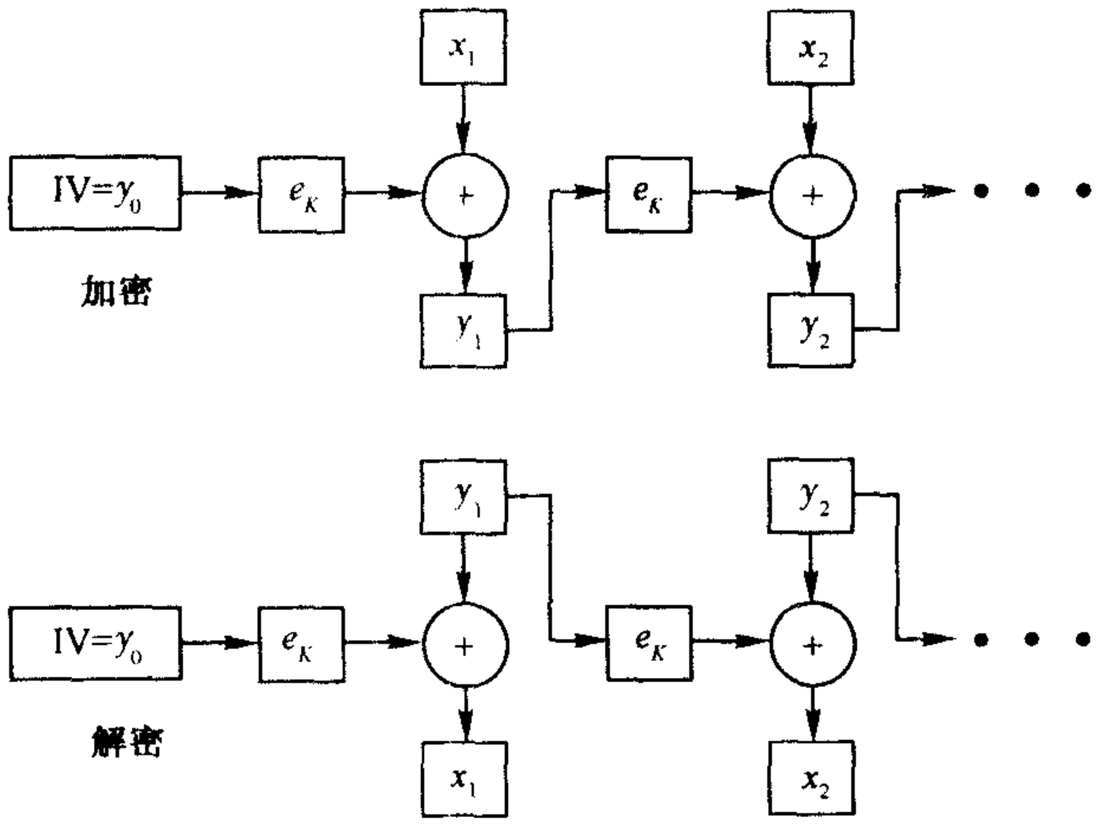

图 3.10　CFB 模式

在 ECB 和 OFB 模式中,改变一个 64 比特的明文分组 x_i 仅仅引起相对应的密文分组 y_i 的改变,而其他密文分组则不受影响。在有些环境下,这是一个所期待的性质,例如,通信信道不十分安全。OFB 模式常常用来加密卫星通信。

另一方面,在 CBC 和 CFB 模式中,改变一个明文分组 x_i,则 y_i 与其后所有密文分组都会受到影响。这一性质说明 CBC 和 CFB 模式对于认证是有用的。更明确地说,这些模式能被用来产生一个消息认证码,即 MAC。这个 MAC 附着在一系列明文分组的后面,它能使 Bob 相信给定的明文序列的确来自 Alice,而且没有被 Oscar 篡改。这样,这个 MAC 保障了消息的完整性(或认证性,但没有提供机密性)。我们将在第 4 章里更多地讨论 MAC。

3.8　注释与参考文献

Smid 与 Branstad[199]撰写了一篇很好的关于 DES 历史的文章。在文献[83]中能找到一个 Lucifer 的描述。Coppersmith[51]里讨论了几个 DES 设计中与抵抗某些攻击密切相关的方面。Wiener的 DES 密钥搜索机在 CRYPTO '93[215]中描述。Landau 写了几篇关于 DES[128]和 AES[129]的文章。Knudsen[117]是近期发表的一篇关于分组密码的优秀综述。

联邦信息处理标准(FIPS)公布了关于 DES 的以下材料:DES 的描述[72]、DES 的使用和实现[74],DES 的工作模式[73]和使用 DES 进行认证[75]。

针对 DES 的一个时间-存储折中攻击由 Hellman[102]发现。我们在练习里给出了一个相关的方法。

在 FIPS 出版物[81]里可找到一个 AES 的描述。Nechvatal 等的文献[153]是一篇关于 AES 发展的详细报告。文献[54]中介绍了 Rijndael;文献[53]给出了 Rijndael 的前身 Square。Daemon 和 Rijmen也写了一本专著来解释 Rijndael 及融入在设计中的各种策略。关于对减少轮数的 Rijndael 变体的攻击可以在 Ferguson 等的文献[84]中找到。Ferguson、Schroeppel 和 Whiting

[85]中给出了一个相对简单的 Rijndael 的代数表示。

差分密码分析技术由 Biham 和 Shamir[22]开发出来(还可见[24]及他们关于 DES 差分密码分析的专著[23])。线性密码分析由 Matsui[137,138]开发出来。关于发展了这些攻击理论基础的著作有 Lai、Massey 及 Murphy[126]和 Nyberg[156]。关于 DES 的线性密码分析有效性的最新的实验结果能在 Junod[108]中找到。

我们对于差分和线性密码分析的处理方式是基于 Heys[104]一书;我们同样采用了[104]中描述的针对 SPN 的差分和线性密码分析。SPN 设计中抵抗线性和差分密码分析的一般原则由 Heys 及 Tavares[105]给出。关于 Rijndael 抵抗线性密码分析的安全性的最新结果由 Keliher、Meijer 及 Tavares[112,113]给出。

Nyberg[155]中建议使用域上的逆运算来设计 S 盒(这一技术后来在 Rijndael 中得到采用)。Chabaud 及 Vaudenay[45]同样也研究了能抵抗差分和线性密码分析的 S 盒的设计。

练习

3.1 设 y 是算法 3.1 在输入为 x 时的输出,π_S 与 π_P 同例 3.1 中的定义相同。换言之,

$$y = \mathrm{SPN}(x, \pi_S, \pi_P, (K^1, \cdots, K^{Nr+1}))$$

这里$(K^1, \cdots, K^{Nr+1})$是密钥编排方案。试找出一个代换 $\pi_S{}^*$ 和一个置换 $\pi_P{}^*$,满足

$$x = \mathrm{SPN}(y, \pi_S{}^*, \pi_P{}^*, (K^{Nr+1}, \cdots, K^1))$$

3.2 证明解密一个 Feistel 密码相当于对密文使用加密算法,但密钥编排方案要逆序使用。

3.3 设 DES(x, K)表示使用 DES 在密钥 K 下对明文 x 进行加密,假定 $y = \mathrm{DES}(x, K)$, $y' = \mathrm{DES}(c(x), c(K))$,这里 $c(\cdot)$表示对其自变量按比特位取反。试证明 $y' = c(y)$(即如果把明文和密钥都按比特位取反,则密文同样是按比特位取反)。注意,证明这一点只需使用 DES 的"高层"描述,S 盒的实际结构和系统的其他组件与此无关。

3.4 在 AES 研发之前,有人曾建议通过使用乘积密码 DES × DES(见 2.7 节)来增加 DES 的安全性。该乘积密码使用两个 56 比特的密钥。

本练习考察对这种乘积密码的已知明文攻击。一般来说,假设我们取任何一个自同态密码 $\mathbf{S} = (\mathcal{P}, \mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$与它自身的乘积,更进一步假设$\mathcal{K} = \{0,1\}^n$ 及$\mathcal{P} = \{0,1\}^m$。

现在假设我们有乘积密码 $\mathbf{S}^2$ 的明-密文对$(x_1, y_1), \cdots, (x_l, y_l)$这些明-密文对都是用同一未知密钥$(K_1, K_2)$加密得来。

(a) 证明对所有 $i, 1 \leqslant i \leqslant l$,有 $e_{K_1}(x_i) = d_{K_2}(y_i)$。这表明满足对所有 $i, 1 \leqslant i \leqslant l, e_{K_1}(x_i) = d_{K_2}(y_i)$的密钥$(K_1, K_2)$的期望数大约为 2^{2n-lm}。

(b) 假设 $l \geqslant 2n/m$,可以应用一个时间-存储折中攻击来求出未知密钥(K_1, K_2)。我们计算两个表,每一个表都含有 2^n 项,这里的每一项都是一个含有$\mathcal{P}$元素和一个$\mathcal{K}$元素的 l 重。如果已将这两个表存储好,则一个普通的 l 重能通过对每一个表进行线性查找辨别出。证明这个算法需要 $2^{n+m+1}l + 2^{2n+1}$ 比特的存储及 $l2^{n+1}$ 次加密或解密

操作。

(c) 证明如果加密的总数增加到 2^t 倍，则这个攻击所需的存储要求将减少到原来的 $1/2^t$。

提示：把问题拆分成 2^{2t} 种子情况，这里每一种子情况由同时定下 K_1 的 t 比特和 K_2 的 t 比特来确定。

3.5　假设我们有 128 比特的 AES 密钥，用十六进制表示为：

```
2B7E151628AED2A6ABF7158809CF4F3C
```

由上述种子密钥构造一个完整的密钥编排方案。

3.6　使用上题中的 128 比特密钥，在 10 轮 AES 下计算下列明文(以十六进制表示)的加密结果：

```
3243F6A8885A308D313198A2E0370734
```

3.7　设明文分组序列 $x_1 \cdots x_n$ 产生的密文分组序列为 $y_1 \cdots y_n$。假设一个密文分组 y_i 在传输时出现了错误(即某些 1 变成了 0，或者相反)。证明不能正确解密的明文分组数目在应用 ECB 或 OFB 模式时为 1，在应用 CBC 或 CFB 模式时为 2。

3.8　本练习的目的是考察一下对特定类型密码在选择明文攻击下的时间-存储折中攻击。假设我们有一个完善保密密码体制满足 $\mathcal{P} = \mathcal{C} = \mathcal{K}$，则 $e_K(x) = e_{K_1}(x)$ 意味着 $K = K_1$。记 $\mathcal{P} = Y = \{y_1, \cdots, y_N\}$。设 x 是一个固定的明文。定义函数 $g: Y \to Y$ 为 $g(y) = e_y(x)$。定义一个有向图 G 具有顶点集 Y，边集包括所有的有向边 $(y_i, g(y_i))$，$1 \leqslant i \leqslant N$。

算法 3.7　时间-存储折中(x)

$y_0 \leftarrow y$

$backup \leftarrow$ **false**

while $g(y) \neq y_0$

do { **if** 对某 j 有 $y = z_j$ **and not** $backup$

then { $y \leftarrow g^{-T}(z_j)$; $backup \leftarrow$ **true** }

else { $y \leftarrow g(y)$; $K \leftarrow y$ }

(a) 证明有向图 G 由不相交的有向圈连接而成。

(b) 设 T 为一个适当的时间参数。假设我们有一个集合 $Z = \{z_1, \cdots, z_m\} \subseteq Y$ 满足对每一个元素 $y_i \in Y$，要么 y_i 包含在一个长度至多为 T 的圈中，要么存在一个 $z_j \neq y_i$，满足在 G 中从 y_i 到 z_j 的距离至多为 T。证明存在这样一个集合 Z，在满足

$$|Z| \leqslant \frac{2N}{T}$$

时，$|Z| = O(N/T)$。

(c) 对每一个 $z_j \in Z$，定义 $g^{-T}(z_j)$ 为满足 $g^T(y_i) = z_j$ 的元素 y_i，这里，g^T 是将 g 迭代 T 次后的函数。构造一张表 X 包含所有有序对 $(z_j, g^{-T}(z_j))$，并按它们的第一个坐标排序存储。

给定 $y = e_K(x)$，就给出查找密钥 K 的算法的伪码描述。证明该算法在至多 T 步内找到密钥 K(因此，时间-存储折中为 $O(N)$)。

(d) 描述一个伪编码算法，在时间 $O(NT)$ 内构造所期望的集合 Z，要求不使用规模为 N 的数组。

3.9 设 $\mathbf{X}_1$、$\mathbf{X}_2$ 及 $\mathbf{X}_3$ 是定义在集 $\{0,1\}$ 上的独立离散随机变量。用 ϵ_i 表示 $\mathbf{X}_i$ 的偏差，$i = 1,2,3$。证明 $\mathbf{X}_1 \oplus \mathbf{X}_2$ 与 $\mathbf{X}_2 \oplus \mathbf{X}_3$ 相互独立当且仅当 $\epsilon_1 = 0, \epsilon_3 = 0$ 或 $\epsilon_2 = \pm 1/2$。

3.10 对 DES 的每一个 S 盒，计算下列随机变量的偏差(应注意到这些偏差在绝对值上都相对较大)。

$$\mathbf{X}_2 \oplus \mathbf{Y}_1 \oplus \mathbf{Y}_2 \oplus \mathbf{Y}_3 \oplus \mathbf{Y}_4$$

3.11 DES 的 S 盒 S_4 具有一些不寻常的性质：

(a) 证明 S_4 的第二行能由第一行通过下列映射获得：

$$(y_1, y_2, y_3, y_4) \mapsto (y_2, y_1, y_4, y_3) \oplus (0,1,1,0)$$

这里的项都用二进制串来表示。

(b) 证明 S_4 的任何一行都能通过一个类似的操作变换成另一行。

3.12 设 $\pi_S : \{0,1\}^m \to \{0,1\}^n$ 是一个 S 盒。证明下列关于函数 N_L 的事实。

(a) $N_L(0,0) = 2^m$。

(b) 对任何满足 $0 < a \leqslant 2^{m-1}$ 的整数 a，有 $N_L(a,0) = 2^m - 1$。

(c) 对任何满足 $0 \leqslant b \leqslant 2^n - 1$ 的整数 b，有

$$\sum_{a=0}^{2^m-1} N_L(a,b) = 2^{2m-1} \pm 2^{m-1}$$

(d) 下述关系式成立：

$$\sum_{a=0}^{2^m-1} \sum_{b=0}^{2^n-1} N_L(a,b) \in \{2^{n+2m-1}, 2^{n+2m-1} + 2^{n+m-1}\}$$

3.13 我们说一个 S 盒 $\pi_S : \{0,1\}^m \to \{0,1\}^n$ 是平衡的，如果对所有 $y \in \{0,1\}^n$，有

$$|\pi_S^{-1}(y)| = 2^{n-m}$$

证明下列关于平衡 S 盒的 N_L 函数满足的事实。

(a) 对所有满足 $0 < b \leqslant 2^n - 1$ 的整数 b，有 $N_L(0,b) = 2^{m-1}$。

(b) 对任何满足 $0 \leqslant a \leqslant 2^m - 1$ 的整数 a，下述关系式成立：

$$\sum_{b=0}^{2^n-1} N_L(a,b) = 2^{m+n-1} - 2^{m-1} + i2^n$$

这里的整数 i 满足 $0 \leqslant i \leqslant 2^{m-n}$。

3.14　假设例 3.1 中的 S 盒被下述由代换 π_S 定义的 S 盒取代：

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_{S'}(z)$	8	4	2	1	C	6	3	D	A	5	E	7	F	B	9	0

(a) 对该 S 盒计算 N_L 值表。

(b) 找出一个使用 3 个活动 S 盒的线性逼近，并用堆积引理来估计下列随机变量的偏差：

$$\mathbf{X}_{16} \oplus \mathbf{U}_1^4 \oplus \mathbf{U}_9^4$$

(c) 类似于算法 3.2，描述一个线性攻击来找到最后一轮的 8 比特子密钥。

(d) 实现你的攻击算法，并测试为了找到正确的子密钥比特需要多少明文(大约 1000 ~ 1500 个明文就足够了；这个算法比算法 3.2 更有效，这是因为使用的偏差是后者中的 2 倍，这就意味着所需明文数目会减到原来的 1/4)。

3.15　假设例 3.1 中的 S 盒被下述由代换 $\pi_{S''}$ 定义的 S 盒取代：

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_{S''}(z)$	E	2	1	3	D	9	0	6	F	4	5	A	8	C	7	B

(a) 对该 S 盒计算 N_D 值表。

(b) 找出一个应用四个活动 S 盒的差分链，即，S_1^1、S_4^1、S_4^2 和 S_4^3 具有扩散率 27/2048。

(c) 类似于算法 3.3，描述一个差分攻击来找到最后一轮的 8 个子密钥比特。

(d) 实现你的攻击，并测试一下它需要多少明文来找到正确的子密钥比特(大约 100 ~ 200 个就足够了；这个算法不如算法 3.3 有效，因为扩散率缩小了一半)。

3.16　假设我们应用例 3.1 中给出的 SPN，但 S 盒被一个不是置换的函数 π_T 取代。这意味着 π_T 不是一个满射。应用这一事实来导出一个惟密文攻击，在给定足够多的用同一密钥加密的密文的情况下，来确定最后一轮的密钥比特。

第 4 章　Hash 函数

4.1　Hash 函数与数据完整性

密码学上的 Hash 函数能保障数据的完整性。Hash 函数通常用来构造数据的短“指纹”；一旦数据改变，指纹就不再正确。即使数据被存储在不安全的地方，通过重新计算数据的指纹并验证指纹是否改变，就能够检测数据的完整性。

设 h 是一个 Hash 函数，x 是数据。作为一个直观的例子，不妨设 x 是任意长度的二元串，相应的指纹定义为 $y = h(x)$。通常指纹也被称为消息摘要(message digest)。一个典型的消息摘要是相当短的二元串，通常是 160 比特。

我们假定 y 被存储在一个安全的地方，而对 x 没有这个要求。如果 x 改变为 x'，则我们希望“旧”的消息摘要 y 并不是 x' 的消息摘要。如果真是这样的话，则通过计算消息摘要 $y' = h(x')$ 并验证 $y' \neq y$ 就很容易发现 x 被改变这个事实。

Hash 函数在数字签名方案中有着特别重要的应用，我们将在第 7 章中进行讨论。

上面所讨论的例子假定存在一个单一固定的 Hash 函数，这也有助于研究带密钥的 Hash 函数族。一个带密钥的 Hash 函数通常用来作为消息认证码，即 MAC。假定 Alice 和 Bob 共享密钥 K，该密钥确定了一个 Hash 函数 h_k。对于消息 x，Alice 和 Bob 都能够计算出相应的认证标签 $y = h_k(x)$。二元组 (x, y) 可以在不安全信道上从 Alice 传送给 Bob。当 Bob 接收到 (x, y) 后，他能够验证是否有 $y = h_k(x)$。如果这个条件满足并且所用到的 Hash 族是“安全”的，那么他就可以确信 x 和 y 都没有被篡改。

注意，由不带密钥的 Hash 函数和带密钥的 Hash 函数各自提供的数据完整性保证是有区别的。用不带密钥的 Hash 函数时，消息摘要必须被安全地存放，不能被篡改。另一方面，如果 Alice 和 Bob 用密钥 K 来确定所用到的 Hash 函数，他们可以在不安全的信道中同时传送数据和认证标签。

在本章的剩余部分，我们将研究 Hash 函数和带密钥的 Hash 族。我们首先给出带密钥的 Hash 族的定义。

定义 4.1　一个 Hash 族是满足下列条件的四元组 $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$：

1. $\mathcal{X}$ 是所有消息的集合。
2. $\mathcal{Y}$ 是由所有的消息摘要或认证标签组成的有限集。
3. $\mathcal{K}$ 是密钥空间，是所有密钥的有限集。
4. 对于每个 $K \in \mathcal{K}$，存在一个 Hash 函数 $h_k \in \mathcal{H}$，$h_K : \mathcal{X} \to \mathcal{Y}$。

在上面的定义中,$\mathcal{X}$可以是有限或无限集;$\mathcal{Y}$总是有限集。如果$\mathcal{X}$是有限集,则 *Hash* 函数常常称为压缩函数。这时,我们总是假定$|\mathcal{X}| \geqslant |\mathcal{Y}|$,并且还经常假定更强的条件$|\mathcal{X}| \geqslant 2|\mathcal{Y}|$。

如果 $h_K(x) = y$,则二元组$(x, y) \in \mathcal{X} \times \mathcal{Y}$称为在密钥 K 下是有效的。本章所讨论的很多内容是关于防止对手构造某种类型的有效二元组的方法。

令$\mathcal{F}^{\mathcal{X},\mathcal{Y}}$为所有从$\mathcal{X}$到$\mathcal{Y}$的函数集合。假定$|\mathcal{X}| = N$, $|\mathcal{Y}| = M$。则显然$\mathcal{F}^{\mathcal{X},\mathcal{Y}} = M^N$。任何 Hash 族$\mathcal{F} \subseteq \mathcal{F}^{\mathcal{X},\mathcal{Y}}$被称为一个$(N, M)$-Hash 族。

一个不带密钥的 Hash 函数是一个函数 $h: \mathcal{X} \to \mathcal{Y}$,其中$\mathcal{X}$和$\mathcal{Y}$与定义 4.1 一致。我们可以把不带密钥的 Hash 函数简单地当做仅仅只有一个密钥的 Hash 族,也就是说,$|\mathcal{K}| = 1$。

本章中剩下的部分按以下方式来组织。在 4.2 节中,我们介绍了 Hash 函数的安全性概念,特别是碰撞稳固(collision resistance)的观点。我们还在这节中研究了随机预言模型中理想的 Hash 函数的严格安全性,并讨论了生日悖论,以对在任意一个 Hash 函数中发现碰撞的难度进行估计。4.3 节介绍了迭代 Hash 函数重要的设计技术。我们讨论了怎样把这个方法用于实用 Hash 函数的设计,以及用于从安全压缩函数构造出可证明安全的 Hash 函数。在 4.4 节中提供了对消息认证码的分析,我们也给出了一些一般性的结构和安全性证明。在 4.5 节中考虑了无条件安全 MAC 以及利用强通用 Hash 族构造无条件安全 MAC 的方法。

4.2 Hash 函数的安全性

假定 $h: \mathcal{X} \to \mathcal{Y}$是一个不带密钥的 Hash 函数。设 $x \in \mathcal{X}$,定义 $y = h(x)$。很多应用在密码学中的 Hash 函数都希望仅有一种方法产生出一个有效的二元组(x, y),就是首先选择 x,再把函数 h 作用于x,计算出 $y = h(x)$。Hash 函数在特定协议中的应用会出现其他的安全需求,就像在签名方案中那样(参见第 7 章)。我们现在定义三个问题,如果一个 Hash 函数被认为是安全的,就应该出现对这三个问题都是难解的情况。

问题 4.1 原像(Preimage)

条件: Hash 函数 $h: \mathcal{X} \to \mathcal{Y}$和 $y \in \mathcal{Y}$。

找出: $x \in \mathcal{X}$使得 $h(x) = y$。

给定一个(可能)的消息摘要 y,原像问题是问是否可找到 x 使得 $h(x) = y$。如果对某个给定的 $y \in \mathcal{Y}$,原像问题能够解决,则(x, y)是有效的。不能有效解决原像问题的 Hash 函数通常称为单向(one-way)的或者原像稳固(preimage resistant)的。

问题 4.2 第二原像(Second Preimage)

条件: Hash 函数 $h: \mathcal{X} \to \mathcal{Y}$和 $x \in \mathcal{X}$。

找出: $x' \in \mathcal{X}$使得 $x' \neq x$,并且 $h(x') = h(x)$。

给定一个消息 x,第二原像问题是问是否能找到 $x' \neq x$,使得 $h(x') = h(x)$。要注意的是,如果问题能够解决,则$(x', h(x))$是有效的二元组。不能有效解决第二原像问题的 Hash 函数通常称为第二原像稳固(second preimage resistant)。

问题 4.3　碰撞(Collision)

条件：　Hash 函数 $h: \mathcal{X} \to \mathcal{Y}$。

找出：　$x, x' \in \mathcal{X}$使得 $x' \neq x$,并且 $h(x') = h(x)$。

碰撞问题是问是否可找到 $x' \neq x$,使得 $h(x') = h(x)$。对这个问题的解答并不能直接产生有效的二元组。可是,如果(x, y)是有效的二元组,并且 x, x'是碰撞问题的解,则(x', y)也是一个有效的二元组。已有各种各样的方案来避免这种情况的出现。不能有效解决碰撞问题的 Hash 函数通常称为碰撞稳固。

在后面,我们要处理的一部分难题就是关于这三个问题各自的难度,以及这三个问题相对的难度。

4.2.1　随机预言模型

在本小节中,我们描述了 Hash 函数的某种理想化的模型,试图得到一个“理想的”Hash 函数。如果 Hash 函数 h 设计得好,对给定的 x,求出函数 h 在点 x 的值应该是得到 $h(x)$的惟一有效的方法。甚至当其他的值 $h(x_1), h(x_2), \cdots$已经计算出来,这仍然应该是正确的。

下面举一个不能保持上述性质的例子。假定 Hash 函数 $h: \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ 是一个线性函数,令

$$h(x, y) = ax + by \bmod n$$

$a, b \in \mathbb{Z}_n$ 且 $n \geqslant 2$ 是正整数。假定已得到

$$h(x_1, y_1) = z_1$$

和

$$h(x_2, y_2) = z_2$$

令 $r, s \in \mathbb{Z}_n$;则有

$$\begin{aligned} h(rx_1 + sx_2 \bmod n, ry_1 + sy_2 \bmod n) &= a(rx_1 + sx_2) + b(ry_1 + sy_2) \bmod n \\ &= r(ax_1 + by_1) + s(ax_2 + by_2) \bmod n \\ &= r\,h(x_1, y_1) + s\,h(x_2, y_2) \bmod n \end{aligned}$$

因此,给定函数 h 在(x_1, y_1)和(x_2, y_2)两点的值,我们就可以知道其他各点的值而无需实际计算 h 在这些点的值(为了应用上述技术,甚至不需要知道常数 a 和 b 的值)。

由 Bellare 和 Rogaway 引入的随机预言模型(random oracle model)提供了一个“理想的”Hash 函数的数学模型。在这个模型中,随机从$\mathcal{F}^{\mathcal{X},\mathcal{Y}}$中选出一个 Hash 函数 $h: \mathcal{X} \to \mathcal{Y}$,我们仅允许预言器访问函数 h。这意味着不会给出一个公式或者算法来计算函数 h 的值。因此,计算 $h(x)$的

惟一方法是询问预言器。这可以想像为在一本巨大的关于随机数的书中查询 $h(x)$的值,对于每个 x,有一个完全随机的值 $h(x)$与之对应。

作为在随机预言模型假定下的结果,下面的独立性质是显然成立的:

定理 4.1 假定 $h \in \mathcal{F}^{\mathcal{X},\mathcal{Y}}$是随机选择的,令$\mathcal{X}_0 \subseteq \mathcal{X}$。假定当且仅当 $x \in \mathcal{X}_0$ 时,$h(x)$(通过查询 h 的预言器)被确定。则对所有的 $x \in \mathcal{X} \setminus \mathcal{X}_0$ 和 $y \in \mathcal{Y}$,有 $\mathbf{Pr}[h(x) = y] = 1/M$。

在上述定理当中,概率 $\mathbf{Pr}[h(x) = y]$实际上是对任意的 $x \in \mathcal{X}_0$ 计算所有的函数 h,得出指定值的条件概率。定理 4.1 是随机预言模型中关于问题复杂性证明中所要用到的关键性质。我们将在下一小节中进一步讨论。

4.2.2 随机预言模型中的算法

在本小节中,我们考虑随机预言模型下,4.2 节定义的三个问题的复杂性。因为不需要知道关于 Hash 函数的任何信息(除了必须具体说明的,对任意 x 值要计算出 Hash 函数值的方法),所以在随机预言模型中的算法可应用于任何 Hash 函数。

我们介绍和分析的算法都是随机算法;它们能在执行的过程中做出随机的选择。Las Vegas 算法是一个不一定给出答案的随机算法(也就是说,会随着消息"失败"而终止),但是一旦该算法返回一个答案,那么这个答案就是正确的。

假定 $0 \leqslant \epsilon < 1$ 是一个实数。如果对每个问题实例,一个随机算法能返回一个正确答案的概率是 ϵ,那么该算法具有最差情况成功率 ϵ(worst-case success probability)。如果对规定范围里的问题实例,一个随机算法平均能返回一个正确答案的概率至少是 ϵ,那么该算法有平均情况成功率 ϵ(average-case success probability)。要注意的是,在后一种情况中,对给定的问题实例,算法返回正确答案的概率可能高于,也可能低于 ϵ。

在本小节中,我们用术语(ϵ, q)-算法来表示一个具有平均情况成功率 ϵ 的 Las Vegas 算法,该算法向预言器查询(也就是求 h 的值)的次数最多为 q。如果 x 或者 y 被确定作为问题实例的一部分,成功率 ϵ 就是对所有 $h \in \mathcal{F}^{\mathcal{X},\mathcal{Y}}$,以及所有 $x \in \mathcal{X}$或 $y \in \mathcal{Y}$的可能出现的随机选择的平均值。

我们分析一下那些在随机预言模型中计算 q 个 $x \in \mathcal{X}$的 $h(x)$值的一般算法。实际上,因为是对所有的 $h \in \mathcal{F}^{\mathcal{X},\mathcal{Y}}$取平均值,这就说明了这个算法的复杂性不依赖于 q 个 x 值的选择。

首先考虑一个通过计算 q 个点的 h 值来解决原像问题的算法。

算法 4.1 FidPreimage(h, y, q)

选择任意的$\mathcal{X}_0 \subseteq \mathcal{X}$, $|\mathcal{X}_0| = q$

for each $x \in \mathcal{X}_0$

do { **if** $h(x) = y$ **then return**(x)

return(failure)

定理 4.2　对任意的$\mathcal{X}_0 \subseteq \mathcal{X}$，且$|\mathcal{X}_0| = q$，算法 4.1 中的平均情况成功率为 $\varepsilon = 1-(1-1/M)^q$。

证明　给定 $y \in \mathcal{Y}$，令$\mathcal{X}_0 = \{x_1, \cdots\cdots, x_q\}$。对于 $1 \leqslant i \leqslant q$，令 E_i 表示事件“$h(x_i) = y$”。由定理 4.1 可知，E_i 是独立事件，并且对所有的 $1 \leqslant i \leqslant q$，$\mathbf{Pr}[E_i] = 1/M$。因此，下面的等式成立：

$$\mathbf{Pr}[E_1 \vee E_2 \vee \cdots \vee E_q] = 1 - \left(1 - \frac{1}{M}\right)^q$$

对任何选定的 y，算法 4.1 的成功率都是常数。因此，所有 $y \in \mathcal{Y}$的平均成功率也是相同的。

注意，q 远小于 M，所以上面的成功率大约是 q/M。

现在介绍和分析一个类似的企图解决第二原像问题的算法。

算法 4.2　FindSecondPreimage(h, x, q)

$y \leftarrow h(x)$

选择$\mathcal{X}_0 \subseteq \mathcal{X} \setminus \{x\}$, $|\mathcal{X}_0| = q-1$

for each $x_0 \in \mathcal{X}_0$

do { **if** $h(x_0) = y$
　　then return(x_0)

return(failure)

对算法 4.2 的分析与以前的算法分析类似。惟一不同之处是额外使用 h，来对输入值 x 计算 $y = h(x)$。

定理 4.3　对任意$\mathcal{X}_0 \subseteq \mathcal{X} \setminus \{x\}$，且$|\mathcal{X}_0| = q-1$，算法 4.2 中的成功率为 $\varepsilon = 1-(1-1/M)^{q-1}$。

下面是针对碰撞问题的基本算法。

算法 4.3　FindCollision(h, q)

选择任意的$\mathcal{X}_0 \subseteq \mathcal{X}$, $|\mathcal{X}_0| = q$

for each $x \in \mathcal{X}_0$

　do $y_x \leftarrow h(x)$

if 对某一 $x' \neq x$，有 $y_x = y_{x'}$

　then return(x, x')

　else return (failure)

在算法 4.3 中，对某一 $x' \neq x$，可以有效地检验是否有 $y_x = y_{x'}$，例如，可以对 y_x 排序。这

个算法可以利用类似于标准的"生日悖论"的概率观点来分析。生日悖论是说在一个随机选择的 23 个成员的组里面,至少有两人生日相同的概率至少为 1/2(当然这不是一个悖论,但它与直觉相反)。这可能表现不出与 Hash 函数有关,但如果我们重新用公式表示有关问题,则两者之间的联系就清楚了。假定函数 h 的定义域是整个人类的集合,对于所有的 x,$h(x)$表示某个人 x 的生日,则 h 的范围是一年的 365 天(如果考虑 2 月 29 号的话是 366 天)。发现两个人是否为同一天生日与在这个 Hash 函数中发现一个碰撞是同一回事。按照这个设定,生日悖论说明了当 $q = 23$ 和 $M = 365$ 时,算法 4.3 的成功率至少为 1/2。

下面我们在随机预言模型下分析算法 4.3。这个算法类似于往 M 个箱子中随机投 q 个球,然后检查是否有某一箱子装有至少两个球(这 q 个球对应于 q 个随机的 x_i,而 M 个箱子对应于$\mathcal{Y}$中 M 个可能的元素)。

定理 4.4 对任意$\mathcal{X}_0 \subseteq \mathcal{X}$,且$|\mathcal{X}_0| = q$,算法 4.3 的成功率为:

$$\varepsilon = 1 - \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-q+1}{M}\right)$$

证明 令$\mathcal{X}_0 = \{x_1, \ldots, x_q\}$。对于 $1 \leqslant i \leqslant q$,令 E_i 表示事件:

$$"h(x_i) \notin \{h(x_1), \cdots, h(x_{i-1})\}"$$

利用归纳法,由定理 4.1 可知 $\mathbf{Pr}[E_i] = 1$,并且对于 $2 \leqslant i \leqslant q$,有:

$$\mathbf{Pr}[E_i \mid E_1 \wedge E_2 \wedge \cdots \wedge E_{i-1}] = \frac{M-i+1}{M}$$

由此可得:

$$\mathbf{Pr}[E_1 \wedge E_2 \wedge \cdots \wedge E_q] = \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-q+1}{M}\right)$$

至此定理成立。

上面的定理说明了无碰撞的概率是:

$$\left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-q+1}{M}\right) = \prod_{i=1}^{q-1}\left(1-\frac{i}{M}\right)$$

如果 x 是一个小实数,则 $1 - x \approx e^{-x}$。这个估计是通过取下面的级数展开的前面两项得来:

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!}\cdots$$

利用这个估计,无碰撞的概率大约是:

$$\prod_{i=1}^{q-1}\left(1-\frac{i}{M}\right) \approx \prod_{i=1}^{q-1} e^{-\frac{i}{M}} = e^{-\sum_{i=1}^{q-1}\frac{i}{M}}$$

$$= e^{\frac{-q(q-1)}{2M}}$$

因此,可以估计产生至少一个碰撞的概率为:

$$1 - e^{\frac{-q(q-1)}{2M}}$$

如果把这个概率表示为 ε,则可以把 q 作为 M 和 ε 的函数:

$$e^{\frac{-q(q-1)}{2M}} \approx 1 - \varepsilon$$

$$\frac{-q(q-1)}{2M} \approx \ln(1-\varepsilon)$$

$$q^2 - q \approx 2M \ln\left(\frac{1}{1-\varepsilon}\right)$$

如果忽略 $-q$ 项,则可以估计出:

$$q \approx \sqrt{2M \ln\left(\frac{1}{1-\varepsilon}\right)}$$

如果取 $\varepsilon = 0.5$,那么

$$q \approx 1.17\sqrt{M}$$

这就说明了在$\mathcal{X}$中对超过$\sqrt{M}$个随机元素计算出的 Hash 函数值里面有 50%的概率出现一个碰撞。注意,对 ε 不同的选择会导致不同的常数因子,但 q 总是与$\sqrt{M}$成一定的比例。这是一个 $(1/2, O(\sqrt{M}))$-算法。

我们回到前面举的例子。令 $M = 365$,可得出 $q \approx 22.3$。所以,像在前面提到的,随机选择 23 个人中至少有两人生日相同的概率至少为 1/2。

生日攻击意味着安全消息摘要的长度有一个下界。40 比特的消息摘要将会非常不安全,因为仅仅在 2^{20}(大约为一百万)个随机 Hash 值中就有 50%的概率发现一个碰撞。通常建议一个消息摘要可接受的最小长度为 128 比特(生日攻击此时需要超过 2^{64}个 Hash 值)。事实上,通常建议用 160 比特或更长的消息摘要。

4.2.3　安全标准的比较

在随机预言模型中,我们已经看到解决碰撞问题比解决原像问题和第二原像问题要容易。一个相关的问题是,在三个问题中是否存在能应用到任意 Hash 函数上的变形。利用算法 4.4 可以相当容易地把碰撞问题变为第二原像问题。

算法 4.4　CollisionToSecondPreimage(h)

external Oracle2ndPreimage

均匀地随机选择 $x \in \mathcal{X}$

if (Oracle2ndPreimage(h, x) = x')

　then return(x, x')

else return(failure)

假定 Oracle2ndPreimage 是对一个特定的 Hash 函数解决了第二原像问题的(ε, q)-算法。显然 CollisionToSecondPreimage 就是对同样的 Hash 函数 h 解决了碰撞问题的$(\varepsilon, q+2)$-算法。这个转化并不要求对 Hash 函数 h 做任何假定。作为这个转化的结果,可以说碰撞稳固性质意味着第二原像稳固性质。

下面将研究更有趣的问题:是否碰撞问题可以转化为原像问题。也就是说,是否碰撞稳固意味着原像稳固?我们将证明至少在一些特殊情况下,这是对的。更明确的是,我们将证明任何能解决原像问题且概率为 1 的算法也能够解决碰撞问题。

这个转化仅仅要求对 Hash 函数的定义域和值域做相当弱的假定就能够完成。设 Hash 函数 $h:\mathcal{X}\to\mathcal{Y}$,其中$\mathcal{X}$和$\mathcal{Y}$都是有限集,并且$|x|\geqslant 2|y|$。假定 OraclePreimage 对原像问题是一个$(1, q)$-算法。OraclePreimage 作为输入接收一个消息摘要 $y\in\mathcal{Y}$,并且总可以发现一个元素 OraclePreimage$(y)\in x$,使得 h(OraclePreimage$(y)) = y$(这意味着 h 是满射)。我们将分析算法 4.5 描述的 CollisionToPreimage 算法。

算法 4.5 CollisionToPreimage(h)

external OraclePreimage

均匀地随机选择 $x\in\mathcal{X}$

$y\leftarrow h(x)$

if (OraclePreimage$(h, y) = x'$)且$(x\neq x')$

then return(x, x')

else return(failure)

定理 4.5 假定 $h:\mathcal{X}\to\mathcal{Y}$是一个 Hash 函数,$|\mathcal{X}|$ 和 $|\mathcal{Y}|$ 是有限的,并且$|\mathcal{X}|\geqslant 2|\mathcal{Y}|$。假定 OraclePreimage对固定的Hash函数 h 是原像问题的一个$(1, q)$-算法。则CollisionToPreimage对固定的 Hash 函数 h 是碰撞问题的一个$(1/2, q+1)$-算法。

证明 显然 CollisionToPreimage 是一个 Las Vegas 类型的概率算法,因为它或者发现一个碰撞,或者返回"失败"。这样,我们的主要任务是计算平均成功率。对任意 $x\in\mathcal{X}$,如果 $h(x) = h(x_1)$,定义 $x\sim x_1$。容易看出"~"是一个等价关系。定义:

$$[x] = \{x_1\in\mathcal{X}: x\sim x_1\}$$

每个等价类$[x]$由$\mathcal{Y}$中某个元的逆像组成。所以等价类的个数等于$|\mathcal{Y}|$。用$\mathcal{C}$表示等价类的集合。

现在,设 x 是由算法 CollisionToPreimage 中选出的$\mathcal{X}$中的随机数。对于这个 x,有$|[x]|$种可能的 x_1 可以作为 OraclePreimage 的输出被返回。有$|[x]|-1$个 x_1 与 x 不同,因此可产生碰撞(注意,算法 OraclePreimage 并不知道由算法 CollisionToPreimage 初始选择的等价类$[x]$的代

表元)。所以,给定元素 $x\in\mathcal{X}$,成功率为$(|[x]|-1)/|[x]|$。

算法 CollisionToPreimage 的成功率是通过对所有可能选择的 x 做平均计算得到:

$$\begin{aligned}\mathbf{Pr}[\text{success}] &= \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\frac{|[x]|-1}{|[x]|}\\ &= \frac{1}{|\mathcal{X}|}\sum_{C\in\mathcal{C}}\sum_{x\in C}\frac{|C|-1}{|C|}\\ &= \frac{1}{|\mathcal{X}|}\sum_{C\in\mathcal{C}}(|C|-1)\\ &= \frac{1}{|\mathcal{X}|}\left(\sum_{C\in\mathcal{C}}|C|-\sum_{C\in\mathcal{C}}1\right)\\ &= \frac{|\mathcal{X}|-|\mathcal{Y}|}{|\mathcal{X}|}\\ &\geqslant \frac{|\mathcal{X}|-|\mathcal{X}|/2}{|\mathcal{X}|}\\ &= \frac{1}{2}\end{aligned}$$

因此,我们构造了一个平均情况成功率至少为 1/2 的 Las Vegas 算法。

4.3　迭代 Hash 函数

迄今为止,考虑的是有限定义域上的 Hash 函数(也就是压缩函数)。现在我们研究一种特殊的技术,将一个记为 compress 的压缩函数,延拓为具有无限定义域的 Hash 函数 h。通过这种方法构造的 Hash 函数称为迭代 Hash 函数。

在这一节中,我们仅关注那些输入和输出都是比特串的 Hash 函数(也就是由 0 和 1 组成的串)。我们把比特串 x 的长度记为$|x|$,并且把比特串 x 和 y 的串联记为 $x\|y$。

设 compress: $\{0,1\}^{m+t}\rightarrow\{0,1\}^m$ 是一个压缩函数,其中 $t\geqslant 1$。基于 compress 的迭代 Hash 函数的求值主要由三步组成。

预处理

给定一个输入比特串 x,其中 $|x|\geqslant m+t+1$,用一个公开的算法构造一个串 y,使得 $|y|\equiv 0(\bmod\ t)$。令

$$y=y_1\|y_2\|\cdots\|y_r$$

其中对 $1\leqslant i\leqslant r$,有 $|y_i|=t$。

处理

设 IV 是一个长度为 m 的作为公开的初始值的比特串。则计算:

$$z_0\leftarrow \mathrm{IV}$$

$$z_1 \leftarrow \text{compress}(z_0 \| y_1)$$
$$z_2 \leftarrow \text{compress}(z_1 \| y_2)$$
$$\vdots \quad \vdots \quad \vdots$$
$$z_r \leftarrow \text{compress}(z_{r-1} \| y_r)$$

可选的输出变换

设 $g: \{0,1\}^m \rightarrow \{0,1\}^l$ 是一个公开函数。定义 $h(x) = g(z_r)$。上面构造的迭代 Hash 函数为:

$$h: \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \rightarrow \{0,1\}^l$$

预处理这一步通常用以下方式构造串 y:

$$y = x \| \text{pad}(x)$$

其中 pad(x)是由填充函数对 x 作用后得到的。一个典型的填充函数是填入 $|x|$ 的值,并填充一些额外的比特(比如说 0)使得所得到的比特串 y 变成 t 倍长。

在预处理阶段必须保证映射 $x \mapsto y$ 是单射(如果映射 $x \mapsto y$ 不是一一对应的,就可能找到 $x \neq x'$ 使得 $y = y'$。则 $h(x) = h(x')$,并且 h 将不是碰撞稳固的)。注意,因为上述映射具有所需要的单射性质,所以有 $|y| = rt \geqslant |x|$。

大多数实用 Hash 函数实际上都是迭代 Hash 函数,并可视做上述通用结构的特殊实例。在 4.3.2 小节中,将介绍一个这样的 Hash 函数——安全 Hash 算法(就是我们熟知的 SHA-1)。我们在下一小节中讨论的 Merkle-Damgård 结构就是一个能够给出正式的安全证明的迭代 Hash 函数。

4.3.1 Merkle-Damgård 结构

在本小节中,我们提出一个用压缩函数构造 Hash 函数的特定方法。用这种方法构造的 Hash 函数能满足所期望的安全性质,如碰撞稳固,只要压缩函数满足此性质。该技术称为 Merkle-Damgård 结构。

假定 compress: $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$ 是一个碰撞稳固压缩函数,其中 $t \geqslant 1$。我们将利用 compress 来构造一个碰撞稳固 Hash 函数 $h: \mathcal{X} \rightarrow \{0,1\}^m$,其中

$$\mathcal{X} = \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i$$

我们首先考虑 $t \geqslant 2$ 时的情况。

我们把元素 $x \in \mathcal{X}$ 视做比特串。假定 $|x| = n \geqslant m + t + 1$。我们把 x 表示成串联的形式:

$$x = x_1 \| x_2 \| \cdots \| x_k,$$

其中,

$$|x_1| = |x_2| = \cdots = |x_{k-1}| = t - 1$$

并且,

$$|x_k| = t - 1 - d$$

其中 $0 \leqslant d \leqslant t-2$。因此，我们得到

$$k = \left\lceil \frac{n}{t-1} \right\rceil$$

我们定义算法 4.6 的输出结果为 $h(x)$。

算法 4.6　Merkle-Damgård(x)

external compress

comment: compress: $\{0,1\}^{m+t} \to \{0,1\}^m$，其中 $t \geqslant 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n \leftarrow k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

　do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \| 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \| y_1$

$g_1 \leftarrow$ compress(z_1)

for $i \leftarrow 1$ **to** k

　do { $z_{i+1} \leftarrow g_i \| 1 \| y_{i+1}$; $g_{i+1} \leftarrow$ compress(z_{i+1}) }

$h(x) \leftarrow g_{k+1}$

return($h(x)$)

令

$$y(x) = y_1 \| y_2 \| \cdots \| y_{k+1}$$

可发现 y_k 是在 x_k 的右边添加 d 个零构成的，因此所有的分组 $y_i (1 \leqslant i \leqslant k)$ 的长度都是 $t-1$。而且 y_{k+1} 应该在左边添加零，使得 $|y_{k+1}| = t-1$。

按照在 4.3 节中所描述的通用结构的做法，我们通过首先构造 $y(x)$，然后用一种特定的方式来处理分组 $y_1, y_2 \cdots y_{k+1}$ 从而得到 x 的 Hash 值。y_{k+1} 是定义在映射 $x \mapsto y(x)$ 为单射的情况下的，这十分有利于观察迭代 Hash 函数是否是碰撞稳固的。

下面的定理表明，如果可找到 h 的一个碰撞，则可找到 compress 的一个碰撞。换言之，如果 compress 是碰撞稳固的，那么 h 也是碰撞稳固的。

定理 4.6　假定 compress: $\{0,1\}^{m+t} \to \{0,1\}^m$ 是碰撞稳固的压缩函数，其中 $t \geqslant 2$。则按

照算法 4.6 构造的函数

$$h: \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \to \{0,1\}^m$$

是一个碰撞稳固的 Hash 函数。

证明 假定找到 $x \neq x'$ 使得 $h(x) = h(x')$。我们将说明如何在多项式时间内找到 compress 的碰撞。

令

$$y(x) = y_1 \| y_2 \| \cdots \| y_{k+1}$$

和

$$y(x') = y'_1 \| y'_2 \| \cdots \| y'_{l+1}$$

其中 x 和 x' 被分别填充了 d 和 d' 个 0。用 $g_1, \cdots, g_{k+1}$ 和 $g'_1 \cdots, g'_{k+1}$ 分别表示算法中计算出的 g 值。

我们根据是否满足 $|x| \equiv |x'| \pmod{t-1}$ 这一条件,区分出两种情况。

情况 1: $|x| \not\equiv |x'| \pmod{t-1}$

这里,$d \neq d'$ 且 $y_{k+1} \neq y'_{l+1}$,可得:

$$\begin{aligned} \text{compress}(g_k \| 1 \| y_{k+1}) &= g_{k+1} \\ &= h(x) \\ &= h(x') \\ &= g'_{l+1} \\ &= \text{compress}(g'_l \| 1 \| y'_{l+1}) \end{aligned}$$

因为,$y_{k+1} \neq y'_{l+1}$,所以找到了 compress 的一个碰撞。

情况 2: $|x| \equiv |x'| \pmod{t-1}$

为了便于讨论,分成两种更细的情况。

情况 2a: $|x| = |x'|$

此时有 $k = l$ 和 $y_{k+1} = y'_{k+1}$,像情况 1 中一样,我们有:

$$\begin{aligned} \text{compress}(g_k \| 1 \| y_{k+1}) &= g_{k+1} \\ &= h(x) \\ &= h(x') \\ &= g'_{k+1} \\ &= \text{compress}(g'_k \| 1 \| y'_{k+1}) \end{aligned}$$

如果 $g_k \neq g'_k$,则找到了 compress 的碰撞,所以可假定 $g_k = g'_k$。则有:

$$\begin{aligned} \text{compress}(g_{k-1} \| 1 \| y_k) &= g_k \\ &= g'_k \\ &= \text{compress}(g'_{k-1} \| 1 \| y'_k) \end{aligned}$$

或者我们找到 compress 的一个碰撞，或者 $g_{k-1} = g'_{k-1}$ 并且 $y_k = y'_k$。假定没有找到碰撞，重复上述过程，最后得到：

$$\begin{aligned}\text{compress}(0^{m+1} \| y_1) &= g_1 \\ &= g'_1 \\ &= \text{compress}(0^{m+1} \| y'_1)\end{aligned}$$

如果 $y_1 \neq y'_1$，则找到了 compress 的一个碰撞，因此可假定 $y_1 = y'_1$。但是对 $1 \leqslant i \leqslant k+1$，都有 $y_i \neq y'_i$，所以 $y(x) = y(x')$。但因为映射是单射 $x \mapsto y(x)$，这意味着 $x = x'$。而我们假定了 $x \neq x'$，这就产生了矛盾。

情况 2b： $|x| \neq |x'|$

不失一般性，设 $|x'| > |x|$，因此 $l > k$。按照情况 2a 类似的过程，假定没有找到 compress 的碰撞，最后总有：

$$\begin{aligned}\text{compress}(0^{m+1} \| y_1) &= g_1 \\ &= g'_{l-k+1} \\ &= \text{compress}(g'_{l-k} \| 1 \| y'_{l-k+1})\end{aligned}$$

但是，$0^{m+1} \| y_1$ 的第 $(m+1)$ 比特是 0，而 $g'_{l-k} \| 1 \| y'_{l-k+1}$ 的第 $(m+1)$ 比特是 1。因此必然会找到 compress 的一个碰撞。

上面已讨论了所有的情况，证明了定理 4.6。

在算法 4.6 中提出的结构仅适用于 $t \geqslant 2$ 的情况。现在看看 $t = 1$ 时的情况。我们需要对 h 使用一个不同的结构。假定 $|x| = n \geqslant m+2$。我们用一种特殊方式对 x 编码。这将用到下面所定义的函数 f：

$$f(0) = 0$$
$$f(1) = 01$$

算法 4.7 说明了 $h(x)$ 的结构。

算法 4.7　Merkle-Damgård2(x)

external compress

comment: compress: $\{0,1\}^{m+1} \to \{0,1\}^m$

$n \leftarrow |x|$

$y \leftarrow 11 \| f(x_1) \| f(x_2) \| \cdots \| f(x_n)$

令 $y = y_1 \| y_2 \| \cdots \| y_k$，其中 $y_i \in \{0,1\}$，$1 \leqslant i \leqslant k$

$g_1 \leftarrow \text{compress}(0^m \| y_1)$

for $i \leftarrow 1$ **to** $k-1$

　do $g_{i+1} \leftarrow \text{compress}(g_i \| y_{i+1})$

return(g_k)

按照算法 4.7 中的定义,编码 $x \mapsto y = y(x)$满足两个重要的性质:

1. 如果 $x \neq x'$,则 $y(x) \neq y(x')$(即 $x \mapsto y(x)$是单射)。

2. 不存在两个串 $x \neq x'$和一个串 z 使得 $y(x) = z \| y(x')$(即没有任何编码是别的编码的后缀。这很显然,因为串 $y(x)$由 11 开始,而在串的剩余部分不存在连续的两个 1)。

定理 4.7 假定 compress: $\{0,1\}^{m+1} \to \{0,1\}^m$ 是碰撞稳固的压缩函数。则按照算法 4.7 构造的函数

$$h: \bigcup_{i=m+2}^{\infty} \{0,1\}^i \to \{0,1\}^m$$

是一个碰撞稳固的 Hash 函数。

证明 假定找到 $x \neq x'$使得 $h(x) \neq h(x')$。令

$$y(x) = y_1 y_2 \cdots y_k$$

和

$$y(x') = y'_1 y'_2 \cdots y'_l$$

考虑两种情况。

情况 1: $k = l$

类似于定理 4.6,我们或者找到 compress 的一个碰撞,或者得到 $y = y'$。但这意味着 $x = x'$,矛盾。

情况 2: $k \neq l$

不失一般性,设 $l > k$。可类似于前面的处理方式。假定没有找到 compress 的碰撞,则有下列的等式序列:

$$\begin{aligned} y_k &= y'_l \\ y_{k-1} &= y'_{l-1} \\ &\vdots \quad \vdots \\ y_1 &= y'_{l-k+1} \end{aligned}$$

但这跟前面的“后缀无关”性质相矛盾。

我们证明了 h 是碰撞稳定的。

在本小节中,我们总结了 Hash 函数的两种结构,而且在下面的定理中,得到了在计算 h 时,需要应用 compress 的次数。

定理 4.8 假定 compress: $\{0,1\}^{m+t} \to \{0,1\}^m$ 是碰撞稳固的压缩函数,其中 $t \geqslant 1$。则存在一个碰撞稳固 Hash 函数:

$$h: \bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \to \{0,1\}^m$$

在求 h 的值的时候,compress 最多被计算的次数为:

$$1+\left\lceil \frac{n}{t-1} \right\rceil, \text{如果 } t \geqslant 2$$

$$2n+2, \text{如果 } t=1$$

其中 $|x|=n$。

4.3.2　安全 Hash 算法

在本小节中，介绍 SHA-1(安全 Hash 算法)，这是一个有 160 比特消息摘要的迭代 Hash 函数。SHA-1 建立在对比特串的面向字的操作上，每一个字由 32 比特(或者由 8 个 16 进制数)组成。SHA-1 所用到的操作如下：

$X \wedge Y$	X 和 Y 的逻辑“和”
$X \vee Y$	X 和 Y 的逻辑“或”
$X \oplus Y$	X 和 Y 的逻辑“异或”
$\neg X$	X 的逻辑“补”
$X+Y$	模 2^{32} 整数加
$\mathrm{ROTL}^s(X)$	X 循环左移 s 个位置($0 \leqslant s \leqslant 31$)

我们首先描述 SHA-1 所用的填充算法，参见算法 4.8。注意，SHA-1 要求 $|x| \leqslant 2^{64}-1$。因此在算法 4.8 中用 l 表示 $|x|$ 的二进制，其长度最多为 64 比特。如果 $|x|<64$，则在左边填充零，使得其长度刚好为 64 比特。

算法 4.8　SHA-1-PAD(x)

comment: $|x| \leqslant 2^{64}-1$

$d \leftarrow (447-|x|) \bmod 512$

$l \leftarrow |x|$ 的二进制表示，其中 $|l|=64$

$y \leftarrow x \| 1 \| 0^d \| l$

在 y 的构造当中，我们添加了一个单独的 1 给 x，然后串联足够的 0 使得其长度模 512 同余 448，最后我们串联包括 x 的(原先的)二进制长度在内的 64 比特。所得的串 y 的长度能被 512 整除。所以，y 可以写成由每个分组为 512 比特，共 n 个分组组成的串联：

$$y = M_1 \| M_2 \| \cdots \| M_n$$

按如下方式定义 $f_0, \cdots, f_{79}$：

$$f_t(B,C,D)=\begin{cases}(B \wedge C) \vee ((\neg B) \wedge D) & 0 \leqslant t \leqslant 19 \\ B \oplus C \oplus D & 20 \leqslant t \leqslant 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leqslant t \leqslant 59 \\ B \oplus C \oplus D & 60 \leqslant t \leqslant 79\end{cases}$$

每个函数 f_t 有 B、C、D 三个字作为输入,并产生一个字作为输出。

按如下方式定义字常数 $K_0, \cdots, K_{79}$,用于 SHA-1(x)的计算:

$$K_t = \begin{cases} 5A827999 & 0 \leqslant t \leqslant 19 \\ 6ED9EBA1 & 20 \leqslant t \leqslant 39 \\ 8F1BBCDC & 40 \leqslant t \leqslant 59 \\ CA62C1D6 & 60 \leqslant t \leqslant 79 \end{cases}$$

现在 SHA-1 可描述为密码体制 4.1。

密码体制 4.1 SHA-1(x)

external SHA-1-PAD
global $K_0, \cdots, K_{79}$
$y \leftarrow$ SHA-1-PAD(x)
令 $y = M_1 \| M_2 \| \cdots \| M_n$,其中每个 M_i 是一个 512 比特的分组
$H_0 \leftarrow$ 67452301
$H_1 \leftarrow$ EFCDAB89
$H_2 \leftarrow$ 98BADCFE
$H_3 \leftarrow$ 10325476
$H_4 \leftarrow$ C3D2E1F0
for $i \leftarrow 1$ **to** n
do
- 令 $M_i = W_0 \| W_1 \| \cdots \| W_{15}$,其中每个 W_i 是一个字
- **for** $t \leftarrow 16$ **to** 79
  - **do** $W_t \leftarrow \mathrm{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$
- $A \leftarrow H_0$
- $B \leftarrow H_1$
- $C \leftarrow H_2$
- $D \leftarrow H_3$
- $E \leftarrow H_4$
- **for** $t \leftarrow 0$ **to** 79
  - **do**
    - $temp \leftarrow \mathrm{ROTL}^5(A) + f_t(B, C, D) + E + W_t + K_t$
    - $E \leftarrow D$
    - $D \leftarrow C$
    - $C \leftarrow \mathrm{ROTL}^{30}(B)$
    - $B \leftarrow A$
    - $A \leftarrow temp$
- $H_0 \leftarrow H_0 + A$
- $H_1 \leftarrow H_1 + B$
- $H_2 \leftarrow H_2 + C$
- $H_3 \leftarrow H_3 + D$
- $H_4 \leftarrow H_4 + E$

return($H_0 \| H_1 \| H_2 \| H_3 \| H_4$)

可以看到 SHA-1 严格遵循迭代 Hash 函数的一般模型。填充方案最多对输入 x 扩充一个额外的 512 比特分组。压缩函数把 160 + 512 比特映射为 160 比特，其中的 512 比特组成一个消息分组。

SHA-1 在一系列相关的迭代 Hash 函数中是最新的。第一个这种 Hash 函数，MD4，由 Rivest 在 1990 年提出。Rivest 随后在 1992 年把 MD4 改进为 MD5。SHA 在 1993 年由 NIST 提议作为标准，并且被采纳为 FIPS 180。SHA-1 是 SHA 的小的变形，为 FIPS 180-1。

Hash 函数的这些进步合并了各种各样的修正来增强后面版本的安全性，抵抗以前版本中已经发现的攻击。例如，在 MD4 中“完全”Hash 函数以及 MD5 中压缩函数的碰撞被揭示出来。原始的 SHA 有一个弱点，允许在大约 2^{61} 步中发现一个碰撞。这种可能性在现有技术条件下并不是可行的，但它比生日攻击有效，生日攻击大约需要 2^{80} 步。这个 SHA 的弱点在 SHA-1 中得到纠正。

2001 年 5 月 30 日，NIST 宣布 FIPS 180-2 的草案接受公众的评审。这次提议的标准包括 SHA-1 和三个新的 Hash 函数，它们分别命名为 SHA-256 和 SHA-384 和 SHA-512（后缀“256”、“384”、“512”表示消息摘要的长度）。这三个新的 Hash 函数也是迭代 Hash 函数，但它们比 SHA-1 具有更复杂的描述。

4.4　消息认证码

我们现在把注意力转向消息认证码，也就是满足某种安全性质的带密钥的 Hash 函数。我们将看到，MAC 所需要的安全性质与（不带密钥的）Hash 函数所需要的安全性质是截然不同的。

构造 MAC 的一个常用方法是通过把一个密钥作为要被 Hash 函数处理的消息的一部分，从而在一个不带密钥的 Hash 函数中介入一个密钥。可是，为了防止某种攻击，这样做必须很小心。我们会在几个例子中展示一些可能的缺陷。

作为第一个例子，假定我们通过一个记为 h 的不带密钥的迭代 Hash 函数，定义 IV = K，并使它的值保密，来构造一个新的带密钥的 Hash 函数 h_K。为了简单起见，假定 h 没有预处理步骤或者输出变换。这样的 Hash 函数需要每个输入消息 x 的长度是 t 的倍数，其中，compress：$\{0,1\}^{m+t} \rightarrow \{0,1\}^m$ 是用于建立 h 的压缩函数。而且，密钥 K 的长度是 m 比特。

我们将说明攻击者怎样对给定的任何消息 x 及相应的 MAC，$h_K(x)$ 下，无需知道密钥 K，就可以构造某个消息的有效 MAC。设 x' 为任意长度为 t 的比特串，考虑消息 $x \| x'$。

对这个消息的 MAC，$h_K(x \| x')$ 计算为：

$$h_k(x \| x') = \text{compress}(h_K(x) \| x')$$

因为 $h_K(x)$ 和 x' 都是已知的，对攻击者来说，即使 K 是保密的，计算 $h_K(x \| x')$ 也是一件简单的事情。这显然是一个安全问题。

即使消息被填充，上述攻击经过修改仍然可以实现。例如，假定在预处理步骤中 $y = x \| \text{pad}(x)$。可发现，对某一整数 r，有 $|y| = r\,t$。设 w 为长度为 t 的任意比特串，定义

$$x' = x \parallel \mathsf{pad}(x) \parallel w$$

在预处理步骤中,我们将计算

$$y' = x' \parallel \mathsf{pad}(x') = x' \parallel \mathsf{pad}(x) \parallel w \parallel \mathsf{pad}(x')$$

其中对某一整数 $r' > r$,有 $|y| = r't$。

考虑 $h_K(x')$的计算(这与 $\mathrm{IV} = K$ 时计算 $h(x')$一样)。在处理步骤中,明显有 $z_r = h_K(x)$。因此攻击者可以做如下计算:

$$\begin{aligned} z_{r+1} &\leftarrow \mathsf{compress}(h_K(x) \parallel y_{r+1}) \\ z_{r+2} &\leftarrow \mathsf{compress}(z_{r+1} \parallel y_{r+2}) \\ &\vdots \quad\quad \vdots \quad\quad \vdots \\ z_{r'} &\leftarrow \mathsf{compress}(z_{r'-1} \parallel y_{r'}) \end{aligned}$$

则

$$h_K(x') = z_{r'}$$

因此攻击者即使不知道密钥 K,也可以计算出 $h_K(x')$(可以看到这种攻击对填充的长度并无要求)。

记住上述例子,我们要明确表达出怎样才意味着 MAC 算法是安全的。就像所看到的,攻击者的目标是试图在一个未知但是固定的密钥 K 下,产生一对有效的(x, y)。攻击者允许请求(直到)q 个他自己选择的消息 $x_1, x_2, \cdots$的有效 MAC。为了方便起见,假定为了回答攻击者的请求,存在一个黑盒(或预言器)。并且我们为了方便起见经常会用到这一术语。

因此,攻击者通过向预言器提出消息 $x_1, \cdots, x_q$ 请求,获得了一列有效的二元对(在未知密钥 K 的情况下):

$$(x_1, y_1), (x_2, y_2), \cdots, (x_q, y_q)$$

后来,当攻击者输出二元对(x, y)时,要求 $x \notin \{x_1, \cdots, x_q\}$。另外,如果$(x, y)$是有效的,就称这个二元对假冒(forgery)。如果攻击者输出一个假冒的概率至少为 ϵ,则攻击者对给定的 MAC,称为一个(ϵ, q)-假冒。概率 ϵ 可能是对所有可能的密钥的平均情况概率,或者是最差情况概率。为了具体化,在后面,我们一般认为概率 ϵ 是最差情况概率。这意味着攻击者不管密钥是否被使用,都能至少以概率 ϵ 产生一个假冒。

用这些术语来讲的话,上面描述的攻击就是(1,1)-假冒。

4.4.1 嵌套 MAC 和 HMAC

一个嵌套 MAC 是指合成两个(带密钥的)Hash 族来建立一个 MAC 算法。假定$(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{G})$和$(\mathcal{Y}, \mathcal{Z}, \mathcal{L}, \mathcal{H})$是 Hash 族。这些 Hash 族的复合是指 Hash 族$(\mathcal{X}, \mathcal{Z}, \mathcal{M}, \mathcal{G} \circ \mathcal{H})$,其中$\mathcal{M} = \mathcal{K} \times \mathcal{L}$并且

$$\mathcal{G} \circ \mathcal{H} = \{goh : g \in \mathcal{G}, g \in \mathcal{H}\}$$

其中对所有的 $x\in\mathcal{X}$,有$(goh)_{(K,L)}(x)=h_L(g_K(x))$。在这个结构中,$\mathcal{Y}$和$\mathcal{Z}$是使得$|\mathcal{Y}|\geqslant|\mathcal{Z}|$的有限集;$\mathcal{X}$是有限的或无限的集合。如果$\mathcal{X}$是有限的,则$|\mathcal{X}|>|\mathcal{Y}|$。

假定这两个 Hash 族是在满足合适的安全要求的情况下构造的,我们的兴趣是发现能保证嵌套 MAC 安全的条件。粗略地讲,证明如果满足以下两个条件,则嵌套 MAC 是安全的:

1. 给定一个固定的(未知的)密钥,作为 MAC,$(\mathcal{Y},\mathcal{Z},\mathcal{L},\mathcal{H})$是安全的;

2. 给定固定的(未知的)密钥,$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{G})$是碰撞稳定的。

直观地讲,是通过一个安全的"小 MAC"(即$(\mathcal{Y},\mathcal{Z},\mathcal{L},\mathcal{H})$)和一个碰撞稳固的带密钥的 Hash 族(即$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{G})$)的复合来构建一个安全的"大 MAC"(即嵌套 MAC)。现在,让我们进一步明确上述条件,然后予以精确的安全性证明。

安全性实际上是基于对三种 Hash 族的某种类型攻击的相对难度的比较。我们将考虑以下三种攻击者:

- 对嵌套 MAC 的假冒者("大 MAC 攻击")
- 对小 MAC 的假冒者("小 MAC 攻击")
- 当密钥是保密的,对 Hash 族的碰撞-探测者("未知密钥碰撞攻击")

下面是对这三种攻击者的更详细的描述。

在"大 MAC 攻击"中,选择并保密一对密钥(K,L)。攻击者允许选择 x 的值,并查询大 MAC 预言器关于 $h_L(g_K(x))$的值。然后攻击者试图产生二元对(x',z)使得 $z=h_L(g_K(x'))$,其中 x'从未进行过查询。

在"小 MAC 攻击"中,选择并保密密钥 L。攻击者允许选择 y 的值,并查询小 MAC 预言器关于 $h_L(y)$的值。然后攻击者试图产生二元对(y',z)使得 $z=h_L(y')$,其中 y'从未进行过查询。

在"未知密钥碰撞攻击"中,选择并保密密钥 K,攻击者允许选择 x 的值,并查询 Hash 预言器关于 $g_K(x)$的值。然后攻击者试图产生 x',x''使得 $x'\neq x''$,并且 $g_K(x')=g_K(x'')$。

我们假定对随机选择的 $g_K\in\mathcal{G}$不存在$(\varepsilon_1,q+1)$-未知密钥碰撞攻击(如果密钥 K 不是保密的,这将符合碰撞稳固的概念。因为假定 K 是保密的,攻击者所面临的问题更加困难,因此我们要求比碰撞稳固更弱的安全假定)。我们也假定对随机选择的 $h_L\in\mathcal{H}$不存在(ε_2,q)-小 MAC 攻击,其中 L 是保密的。最后,假定对随机选择的$(goh)_{(K,L)}\in\mathcal{G}\circ\mathcal{H}$,存在$(\varepsilon,q)$-大 MAC 攻击,其中$(K,L)$是保密的。

由于概率至少为 ε,大 MAC 攻击在向大 MAC 预言器最多查询 q 次后,能输出有效的二元对(x,z)。设 $x_1,\cdots,x_q$ 表示攻击者的查询,又设 $z_1,\cdots,z_q$ 是预言器做出的相应回答。在攻击者执行完查询后,可以得到一列有效的二元对$(x_1,z_1),\cdots,(x_q,z_q)$以及可能的有效二元对$(x,z)$。

假定现在取出 $x_1,\cdots,x_q$ 和 x,向 Hash 预言器作 $q+1$ 次查询。可以获得一列值 $y_1=g_K(x_1),\cdots,y_q=g_K(x_q)$和 $y=g_K(x)$。假定恰巧 $y\in\{y_1,\cdots,y_q\}$,比如说 $y=y_i$,则可以输出一对 x,x_i 作为碰撞的解。这是一次成功的未知密钥碰撞攻击。另一方面,如果 $y\notin\{y_1,\cdots,y_q\}$,则可以输出二元对$(y,z)$,这(可能)是小 MAC 的有效对。从 q 个小 MAC 查询中(间接)得到 q 个答案后,也就是$(y_1,z_1),\cdots,(y_q,z_q)$,可构成一个假冒。

由我们所做的假定,任何未知密钥的碰撞攻击最多有 ε_1 的成功率,而我们假定大 MAC 攻

击至少有 ε 的成功率,因此,(x,z)是有效二元对,并且 $y \notin \{y_1,\cdots,y_q\}$ 的概率至少是 $\varepsilon-\varepsilon_1$。任何小 MAC 攻击的成功率最多是 ε_2,而上述的小 MAC 攻击的成功率至少为 $\varepsilon-\varepsilon_1$,因此,有 $\varepsilon \leqslant \varepsilon_1+\varepsilon_2$。

定理 4.9 假定$(\mathcal{X},\mathcal{Z},\mathcal{M},\mathcal{G}\circ\mathcal{H})$是一个嵌套 MAC。当密钥 K 是保密的,假定对随机选择的 $g_K \in \mathcal{G}$不存在$(\varepsilon_1,q+1)$-碰撞攻击。而且,假定对随机选择的 $h_L \in \mathcal{H}$不存在(ε_2,q)-假冒,其中 L 是保密的。最后,假定对随机选择的$(goh)_{(K,L)} \in \mathcal{G}\circ\mathcal{H}$,对嵌套 MAC 存在$(\varepsilon,q)$-假冒。则 $\varepsilon \leqslant \varepsilon_1+\varepsilon_2$。

HMAC 是一个被提议作为 FIPS 标准的嵌套 MAC 算法。它通过(不带密钥的)Hash 函数来构造 MAC;我们基于 SHA-1 来描述 HAMC。这个版本的 HMAC 用 512 比特的密钥,记为 K。x 是需要认证的消息,ipad 和 opad 是按如下定义的十六进制的 512 比特常数:

$$\text{ipad} = 3636\cdots36$$
$$\text{opad} = 5C5C\cdots5C$$

则 160 比特 MAC 定义如下:

$$\text{HMAC}_K(x) = \text{SHA-1}(K \oplus \text{opad}) \| \text{SHA-1}(((K \oplus \text{ipad}) \| x))$$

注意,HMAC 在 SHA-1 中用 $K\oplus$ipad 添加上 x 作为密钥。SHA-1 的这个应用是假定对未知密钥碰撞攻击是安全的。然后密钥 $K\oplus$opad 再添加上次构造的消息摘要,然后再次应用 SHA-1。两次 SHA-1 计算仅需要应用一次压缩函数,并且我们假定作为 MAC,这样使用 SHA-1 是安全的。如果这两个假定是有效的,则定理 4.9 说明了 HMAC 作为 MAC 是安全的。

4.4.2 CBC-MAC

构造一个 MAC 的最常用的方法之一是基于一个固定的(公开的)初始化向量的 CBC 模式。在 CBC 模式中,每个密文分组 y_i 在用安全密钥 K 加密之前,与下一个明文分组 x_{i+1} 一起异或(x-or)。更正式地,我们由一个记做 IV 的初始化向量开始,定义 $y_0 = \text{IV}$。然后用如下规则构造 $y_1,y_2,\cdots$:

$$y_i = e_K(y_{i-1} \oplus x_i)$$

$i \geqslant 1$。

假定$(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$是一个满同态密码体制,其中$\mathcal{P}=\mathcal{C}=\{0,1\}^t$。设 IV 是由 t 个 0 组成的比特串,又设 $K \in \mathcal{K}$为密钥。最后,令 $x = x_1 \| \cdots \| x_n$ 是长度为 tn 的比特串(对某个正整数 n),其中每个 x_i 都是长度为 t 的比特串。我们按如下方式计算 CBC-MAC(x,K):

密码体制 4.2 CBC-MAC(x,K)

令 $x = x_1 \| \cdots \| x_n$

IV← 00…0
y_0← IV
for i←1 **to** n
　do $y_i \leftarrow e_K(y_{i-1} \oplus x_i)$
return(y_n)

对 CBC-MAC 已知的最好的通用攻击是生日(碰撞)攻击。下面描述这种攻击。基本上,我们假定攻击者能要求得到大量消息的 MAC。如果发现一个重复的 MAC,则攻击者可构造一个另外的消息并且要求得到它的 MAC。最后,攻击者能产生新的消息及其相应的 MAC(也就是假冒),即使他不知道密钥。这种攻击对任何预先确定的,有固定大小的消息都起作用。

下面是攻击过程的细节。令 $n \geqslant 2$ 是整数。令 $x_3, \cdots, x_n$ 是长度为 t 的固定比特串。对于 $1 \leqslant i \leqslant q$ 和 $3 \leqslant h \leqslant n$,定义 $x_h^i = x_h$。令 $q \approx 1.17 \times 2^{t/2}$ 为整数,选择任意 q 个独立的、长度为 t 的比特串,记为 $x_1^1, \cdots, x_1^q$。又令 $x_2^1, \cdots, x_2^q$ 为随机选择的长度为 t 的比特串,对 $1 \leqslant i \leqslant q$,定义

$$x^i = x_1^i \| \ldots \| x_n^i$$

注意,如果 $i \neq j$,因为 $x_1^i \neq x_1^j$,则 $x^i \neq x^j$。

现在攻击者请求得到 $x^1, x^2, \cdots, x^q$ 的 MAC 值。注意,$h_K(x^i) = h_K(x^j)$当且仅当 $y_2^i = y_2^j$,这时当且仅当

$$y_1^i \oplus x_2^i = y_1^j \oplus x_2^j$$

设 x_δ 为长度为 t 的任意比特串。定义

$$v = x_1^i \| (x_2^i \oplus x_\delta) \| \cdots \| x_n^i$$

和

$$w = x_1^j \| (x_2^j \oplus x_\delta) \| \cdots \| x_n^j$$

然后攻击者要求得到 v 的 MAC。不难看出 v 和 w 有相同的 MAC,所以攻击者能成功地构造 w 的 MAC,即使他不知道密钥 K。这次攻击产生了一个$(1/2, O(2^{t/2}))$-假冒。

已经知道,当基本加密算法满足适当的安全性质时,CBC-MAC 是安全的。也就是说,如果某些似乎合理且未证明的关于加密方案随机性的假定是对的,那么 CBC-MAC 将是安全的。

4.5　无条件安全消息认证码

在这一节中,我们将定义泛(universal)Hash 函数族,探讨它们在无条件安全 MAC 构造中的应用。在对无条件安全 MAC 的研究中,我们假定一个密钥只产生一个认证标签。这样,一个攻击者在发动一个可能的假冒攻击之前,他至多只能进行一次查询。也就是说,我们将构造这样的 MAC:对于某些 ε 值,即使攻击者拥有无限的计算能力,我们也能证明不存在$(\varepsilon, 0)$-假冒和$(\varepsilon, 1)$-假冒。

对于 $q=0,1$,我们定义欺骗概率 Pd_q 是使得存在一个(ε,q)-假冒的ε 的最大值。这个最大值是通过对所有可能的密钥值 K 进行计算得来的,K 是从$\mathcal{K}$中随机选取的。通常,我们要构造 Pd_0 和 Pd_1 值小的 MAC。这意味着无论用哪一个密钥,攻击者进行成功攻击的可能性都很小。有时我们把一个$(\varepsilon,0)$-假冒攻击称做模仿(impersonation),把一个$(\varepsilon,1)$-假冒攻击称做代替(substitution)。

下面我们通过无条件安全 MAC 的一个小例子来解释上面的概念。

例 4.1 假定

$$\mathcal{X}=\mathcal{Y}=\mathbb{Z}_3$$

且

$$\mathcal{K}=\mathbb{Z}_3\times\mathbb{Z}_3$$

对每一个 $K=(a,b)\in\mathcal{K}$和每一个 $x\in\mathcal{X}$,定义

$$h_{(a,b)}(x)=ax+b \bmod 3$$

再定义

$$\mathcal{H}=\{h_{(a,b)}:(a,b)\in\mathbb{Z}_3\times\mathbb{Z}_3\}$$

研究 Hash 函数族$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$的认证矩阵会有助于理解。所有 $h_{(a,b)}(x)$的值以如下方式组成一张表:对每一个密钥$(a,b)\in\mathcal{K}$和每一个 $x\in\mathcal{X}$,其认证标签 $h_{(a,b)}(x)$位于一个$|\mathcal{K}|\times|\mathcal{X}|$矩阵 $\boldsymbol{M}$ 的(a,b)行 x 列。矩阵 $\boldsymbol{M}$ 如表 4.1 所示。

表 4.1 认证矩阵

密　钥	0	1	2
(0,0)	0	0	0
(0,1)	1	1	1
(0,2)	2	2	2
(1,0)	0	1	2
(1,1)	1	2	0
(1,2)	2	0	1
(2,0)	0	2	1
(2,1)	1	0	2
(2,2)	2	1	0

先考虑模仿攻击。Oscar 选取一条消息 x,试图猜测出“正确的”认证标签。实际被使用的密钥记为 K_0(Oscar 并不知道)。如果 Oscar 猜出标签 $y_0=h_{K_0}(x)$,他将成功地构造一个假冒。但是,对任一个 $x\in\mathcal{X}$和 $y\in\mathcal{Y}$,很容易证明只有 3 个(总共 9 个)密钥 $K\in\mathcal{K}$,使得 $h_K(x)=y$(也就是说,每个符号在认证矩阵的每一列只出现 3 次)。这样,任何一对(x,y)是有效的概率为

1/3,因此,$Pd_0 = 1/3$。

代替的分析要稍微复杂一些。有一种特殊情况,假定 Oscar 查询 $x = 0$ 时的标签,得到的回答是 $y = 0$,$(x, y) = (0,0)$就是一个有效对。这就给了 Oscar 一些关于密钥的有效信息:他知道

$$K_0 \in \{(0,0),(1,0),(2,0)\}$$

现在假定 Oscar 产生二元组(1,1)作为一个可能的假冒,二元组(1,1)是一个假冒当且仅当 $K_0 = (1,0)$。在给定(0,0)是一个有效的二元组的情况下,因为 K_0 是在集合$\{(0,0),(1,0),(2,0)\}$中,所以 K_0 是密钥的概率为 1/3。

对于 Oscar 查询标签 y 得到的任一 x 值,以及 Oscar 产生的可用来作为假冒的二元组$(x', y')(x' \neq x)$,均可做类似的分析。一般地,任一有效二元组(x, y)的知识把密钥限定于三种可能性之一。这样,对选择的每个$(x', y')(x' \neq x)$,能够证明有这样一个密钥(三个可能的密钥之一):x'是 y'的正确的认证标签。因此,证明了 $Pd_1 = 1/3$。

现在我们讨论一下,对任一消息认证码,如何通过其认证矩阵来计算欺骗概率(the deception probabilities)。首先,考虑 Pd_0。同上,令 K_0 为 Alice 和 Bob 选择的密钥。对任一 $x \in \mathcal{X}$和 $y \in \mathcal{Y}$,定义 payoff(x, y)为二元组(x, y)有效的可能性。不难看出,

$$\text{payoff}(x,y) = \Pr[y = h_{K_0}(x)] = \frac{|\{K \in \mathcal{K} : h_K(x) = y\}|}{|\mathcal{K}|}$$

即,payoff(x, y)是通过计算认证矩阵的 x 列中具有元素 y 的行数,再除以可能的密钥数得出的。

为了增加成功的概率,Oscar 将选择会使 payoff(x, y)值最大的(x, y)。这样,我们就有下面的公式:

$$Pd_0 = \max\{\text{payoff}(x,y) : x \in \mathcal{X}, y \in \mathcal{Y}\} \tag{4.1}$$

现在,考虑代替。选定 $x \in \mathcal{X}$,$y \in \mathcal{Y}$使得(x, y)是一个有效二元组(x 是 Oscar 的一个可能查询,而 y 是他将得到的预言回答)。令 $x' \in \mathcal{X}$,$x' \neq x$,定义 payoff$(x', y'; x, y)$为(x', y')是有效二元组的概率,假定(x, y)为一有效二元组。按照惯例,令 K_0 为 Alice 和 Bob 选择的密钥。那么我们可以按如下方式计算:

$$\begin{aligned}\text{payoff}(x', y'; x, y) &= \Pr[y' = h_{K_0}(x') \mid y = h_{K_0}(x)] \\ &= \frac{\Pr[y' = h_{K_0}(x') \wedge y = h_{K_0}(x)]}{\Pr[y = h_{K_0}(x)]} \\ &= \frac{|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|}{|\{K \in \mathcal{K}, h_K(x) = y\}|}\end{aligned}$$

这个分数的分子是认证矩阵的第 x 列中具有值 y 且第 x'列中具有值 y'的行数;分母为 x 列中具有值 y 的行数。注意,分母为一非零数,因为我们假定(x, y)至少在一个密钥下为一有效二元组。

给定有效二元组(x,y),Oscar将会选择使**payoff**(x,y)值最大的(x',y')。我们要减小Oscar通过所有可能的有效对(x,y)成功实施代替攻击的概率。因此,我们定义:

$$\mathcal{V}=\{(x,y):|K\in\mathcal{K}:h_K(x)=y|\geqslant 1\}$$

$\mathcal{V}$是至少在一个密钥下有效的所有二元组的集合。

我们有如下公式:

$$Pd_1=\max\{\textbf{payoff}(x',y';x,y):x,x'\in\mathcal{X},y,y'\in\mathcal{Y},(x,y)\in\mathcal{V},x\neq x'\} \tag{4.2}$$

4.5.1 强泛Hash函数族

强泛Hash函数族用在密码学的几个领域中。先看看这些重要对象的定义。

定义4.2 假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash函数族。如果对任意的$x,x'\in\mathcal{X},x\neq x',y,y'\in\mathcal{Y}$,满足

$$|\{K\in\mathcal{K}:h_K(x)=y,h_K(x')=y'\}|=\frac{|\mathcal{K}|}{M^2}$$

则称这个Hash函数族是强泛的。

读者可以证明例4.1中的Hash函数族是一个强泛(3,3)-Hash函数族。

强泛Hash函数族产生的认证码中,Pd_0和Pd_1很容易被计算出来。下面先陈述并证明一个关于强泛Hash函数族的简单引理,然后给出一个有关这些欺骗概率的定理的证明。

引理4.10 假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个强泛$(N,M)$-Hash函数族。则对任意$x\in\mathcal{X}$和$y\in\mathcal{Y}$,有:

$$|\{K\in\mathcal{K}:h_K(x)=y\}|=\frac{|\mathcal{K}|}{M}$$

证明 对给定的$x,x'\in\mathcal{X},x\neq x',y,y'\in\mathcal{Y}$,我们有如下结论:

$$\begin{aligned}|\{K\in\mathcal{K}:h_K(x)=y\}|&=\sum_{y'\in\mathcal{Y}}|\{K\in\mathcal{K}:h_K(x)=y,h_K(x')=y'\}|\\&=\sum_{y'\in\mathcal{Y}}\frac{|\mathcal{K}|}{M^2}\\&=\frac{|\mathcal{K}|}{M}\end{aligned}$$

定理4.11 假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个强泛$(N,M)$-Hash函数族。则$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是$Pd_0=Pd_1=1/M$的认证码。

证明 对任意的$x\in\mathcal{X}$和$y\in\mathcal{Y}$,我们在引理4.1中已证明:

$$|\{K\in\mathcal{K}: h_K(x)=y\}| = \frac{|\mathcal{K}|}{M}$$

因此,对任意的 $x\in\mathcal{X}$和 $y\in\mathcal{Y}$,$\text{payoff}(x,y)=1/M$,从而 $Pd_0=1/M$。

令 $x,x'\in\mathcal{X}, x\neq x'; y,y'\in\mathcal{Y}, (x,y)\in\mathcal{V}$,则有如下结论:

$$\begin{aligned}\text{payoff}(x',y';x,y) &= \frac{|\{K\in\mathcal{K}: h_K(x')=y', h_K(x)=y\}|}{|\{K\in\mathcal{K}: h_K(x)=y\}|}\\ &= \frac{|\mathcal{K}|/M^2}{|\mathcal{K}|/M}\\ &= \frac{1}{M}\end{aligned}$$

因此,$Pd_1=1/M$。

现在,我们给出强泛 Hash 函数族的几个构造实例。第一个构造实例是例 4.1 的一般化。

定理 4.12　设 p 为素数,对任意的 $a,b\in\mathbb{Z}_p$,用规则

$$f_{(a,b)}(x) = ax+b \bmod p$$

定义 $f(a,b):\mathbb{Z}_p\to\mathbb{Z}_p$,则$(\mathbb{Z}_p,\mathbb{Z}_p,\mathbb{Z}_p\times\mathbb{Z}_p,\{f_{(a,b)}: a,b\in\mathbb{Z}_p\})$是一个强泛$(p,p)$-Hash 函数族。

证明　假定 $x,x',y,y'\in\mathbb{Z}_p, x\neq x'$,我们将说明有一个惟一的密钥$(a,b)\in\mathbb{Z}_p\times\mathbb{Z}_p$ 使得 $ax+b\equiv y \pmod p$以及 $ax'+b\equiv y' \pmod p$。这并不难做到,因为(a,b)是$\mathbb{Z}_p$ 上关于两个未知数的两个二元二次线性方程的解。特别地,

$$\begin{aligned}a &= (y'-y)(x'-x)^{-1} \bmod p\\ b &= y-x(y'-y)(x'-x)^{-1} \bmod p\end{aligned}$$

(注意,因为 $x\neq x' \pmod p$且 p 是素数,所以$(x'-x)^{-1} \bmod p$ 存在。)

这是一个强泛 Hash 函数族类的构造,其域的基数可比值域更大。

定理 4.13　设 l 为正整数,p 为素数。

对任意的 $\vec{r}\in(\mathbb{Z}_p)^l$,定义$\mathcal{X}=\{0,1\}^l\setminus\{(0,\cdots,0)\}$。用规则

$$f_{\vec{r}}(\vec{x}) = \vec{r}\cdot\vec{x} \bmod p$$

定义 $f_{\vec{r}}:\mathcal{X}\to\mathbb{Z}_p$,其中 $\vec{x}\in\mathcal{X}$,并且

$$\vec{r}\cdot\vec{x} = \sum_{i=1}^{l} r_i x_i$$

是向量的通常内积。那么$(\mathcal{X},\mathbb{Z}_p,(\mathbb{Z}_p)^l,\{f_{\vec{r}}:\vec{r}\in(\mathbb{Z}_p)^l\})$是一个强泛$(2^l-1,p)$-Hash 函数族。

证明 令 $\vec{x},\vec{x}'\in\mathcal{X},\vec{x}\neq\vec{x}',y,y'\in\mathbb{Z}_p$。我们想说明使 $\vec{r}\cdot\vec{x}\equiv y \pmod p$和 $\vec{r}\cdot\vec{x}\equiv y' \pmod p$的向量 $\vec{r}\in(\mathbb{Z}_p)^l$ 的个数是一个常数。向量 $\vec{r}$ 是$\mathbb{Z}_p$ 上两个 l 元线性方程的解。这两个方程是线性无关的,因此线性方程解的个数为常数 p^{l-2}。

4.5.2 欺骗概率的优化

在这一小节中,我们将证明无条件安全认证码欺骗概率的几个下界(lower bounds),它们将表明来源于强泛 Hash 函数族的认证码具有最小的欺骗概率。

假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash 函数族。指定消息 $x\in\mathcal{X}$,我们可计算:

$$\begin{aligned}\sum_{y\in\mathcal{Y}}\text{payoff}(x,y) &= \sum_{y\in\mathcal{Y}}\frac{|\{K\in\mathcal{K}:h_K(x)=y\}|}{|\mathcal{K}|}\\ &= \frac{|\mathcal{K}|}{|\mathcal{K}|}\\ &= 1\end{aligned}$$

因此,对任意的 $x\in\mathcal{X}$,存在一个消息认证标签 y(依赖于 x),使得

$$\text{payoff}(x,y)\geqslant\frac{1}{M}$$

下面的定理是上述计算过程的一个简单推论。

定理 4.14 假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash 函数族,则 $Pd_0\geqslant 1/M$。进一步,对任意的 $x\in\mathcal{X}$和 $y\in\mathcal{Y}$, $Pd_0=1/M$ 当且仅当

$$|\{K\in\mathcal{K}:h_K(x)=y\}|=\frac{|\mathcal{K}|}{M} \tag{4.3}$$

现在,我们考虑替代攻击的欺骗概率下界。假定给定 $x,x'\in\mathcal{X},x\neq x';y,y'\in\mathcal{Y},(x,y)\in\mathcal{V}$。我们有:

$$\begin{aligned}\sum_{y'\in\mathcal{Y}}\text{payoff}(x',y';x,y) &= \sum_{y'\in\mathcal{Y}}\frac{|\{K\in\mathcal{K}:h_K(x')=y',h_K(x)=y\}|}{|\{K\in\mathcal{K}:h_K(x)=y\}|}\\ &= \frac{|\{K\in\mathcal{K}:h_K(x)=y\}|}{|\{K\in\mathcal{K}:h_K(x)=y\}|}\\ &= 1\end{aligned}$$

因此,对每一个$(x,y)\in\mathcal{V}$和 $x'(x\neq x')$,存在一个认证标签 y'使得:

$$\text{payoff}(x',y';x,y)\geqslant\frac{1}{M}$$

我们已证明了如下定理。

定理 4.15　假定$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash 函数族,则 $Pd_1 \geqslant 1/M$。

稍加证明,我们就可确定 $Pd_1 = 1/M$ 的充分必要条件。

定理 4.16　假设$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash 函数族,则 $Pd_1 = 1/M$ 当且仅当该 Hash 函数族是强泛的。

证明　如果 Hash 族是强泛的,我们已经在定理 4.11 中证明了 $Pd_1 = 1/M$。现在要证明反过来也是成立的;所以,我们假定 $Pd_1 = 1/M$。

首先证明$\mathcal{V} = \mathcal{X} \times \mathcal{Y}$。令$(x',y') \in \mathcal{X} \times \mathcal{Y}$;下面证明$(x',y') \in \mathcal{V}$。令 $x \in \mathcal{X}, x \neq x'$。选择任意的 $y \in \mathcal{Y}$使得$(x,y) \in \mathcal{V}$。由定理 4.15 的讨论过程可知,对任意的 $x,x' \in \mathcal{X}, x' \neq x, y,y' \in \mathcal{Y}$且$(x,y) \in \mathcal{V}$,显然有:

$$\frac{|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|}{|\{K \in \mathcal{K} : h_K(x) = y\}|} = \frac{1}{M} \tag{4.4}$$

因此,

$$|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}| > 0$$

于是有

$$|\{K \in \mathcal{K} : h_K(x') = y'\}| > 0$$

这就证明了$(x',y') \in \mathcal{V}$,因此$\mathcal{V} = \mathcal{X} \times \mathcal{Y}$。

现在再回顾一下公式(4.4)。设 $x,x' \in \mathcal{X}, x' \neq x$ 以及 $y,y' \in \mathcal{Y}$。我们有$(x,y) \in \mathcal{V}$和$(x',y') \in \mathcal{V}$,因此可以在公式(4.4)中把(x,y)和(x',y')相交换。对所有这样的 x,x',y,y',就得到

$$|\{K \in \mathcal{K} : h_K(x) = y\}| = |\{K \in \mathcal{K} : h_K(x') = y'\}|$$

因此,

$$|\{K \in \mathcal{K} : h_K(x) = y\}|$$

的个数是一个常数(换句话说,在认证矩阵的任意的 n 列中,符号 y 出现的个数是常数)。再由公式(4.4)可知

$$|\{K \in \mathcal{K} : h_K(x') = y', h_K(x) = y\}|$$

也是一个常数。因此该 Hash 族是强泛的。

下面的推论说明了只要 $Pd_1 = 1/M$ 就有 $Pd_0 = 1/M$。

推论 4.17　假设$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是一个$(N,M)$-Hash 函数族,且 $Pd_1 = 1/M$,则 $Pd_0 = 1/M$。

证明　在假定条件下,由定理 4.16 可知,$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$是强泛的。再由定理 4.11 可知,$Pd_0 = 1/M$。

4.6 注释与参考文献

有关 Hash 函数的最新综述,可参见 Preneel[173]。原像稳固、碰撞稳固等概念已经探讨了一段时间,进一步的详细资料可参见[173]。

随机预言模型是 Bellare 和 Rogaway 在[14]中提出的;4.2.2 小节中的分析取材于 Stinson[207]。

4.3 节的材料取材于 Damgård[56],类似的方法在 Merkle[146]中也有论述。

Rivest 的 MD4 和 MD5 Hash 算法在[179]和[180]中分别有详述。SHA 在 FIPS 180[76]中叙述;它已被 SHA-1 取代,SHA-1 在 FIPS 的 180-1[77]中有详述。FIPS 的 180-2[78]中已有一个草案,它说明的是 Hash 算法 SHA-256、SHA-384 和 SHA-512。

den Boer 和 Bosselaers 在[61]中给出了一个针对三轮 MD4 算法的其中两轮的攻击,Dobbertin 也给出了这样的攻击(参见 CryptoBytes, volume 2, number 2, Summer 1996, pages 1-6)。后来,Dobbertin[69]发现了 MD4 的碰撞。MD5 压缩函数的碰撞是 den Boer 和 Bossalaers[62]发现的。

Chabaud 和 Joux 在 1998 年发现了一种能用大约 2^{61} 次运算来发现 SHA 中的碰撞的算法[44]。那时,SHA 已经升级为 SHA-1 了,这也许正是由于早些时候被 NSA 发现的类似的分类攻击(classified attacks)。

将密码分组链模式用于消息认证的详细情况在 1985 年 FIPS 的 113[75]中有描述。用这种方法构造 MAC 的安全性证明出现要晚得多:参见 Bellare、Kilian 和 Rogaway 的论文[13]。Preneel 和 van Oorschot[175]介绍了迭代(iterated)消息认证码的最新研究进展。构造 MAC 的方法中可证明安全的有 Bellare、Canetti 和 Krawczyk[11]的 HMAC(我们在 4.4.1 小节中已描述),以及 Bellare、Guerin 和 Rogaway[12]的 XOR-MAC。

无条件安全认证码是 Gilbert、Mac Williams 和 Sloane[92]在 1974 年提出的,Simmons 发展了无条件安全消息认证码的很多理论,他证明了这一领域的许多基础性结论,可参见 Simmons[195]。我们的处理方式与 Simmons 介绍的模式不同,因为我们考虑的是主动攻击。在这种情况下,攻击者在产生一个可能的假冒之前,会做一个预言器查询以获得一个认证标签。而在 Simmons 考虑的模型中,攻击是被动的:攻击者用一个相应的认证标签来观测一条消息,但是这条消息不是攻击者选择的。

泛 Hash 函数族的概念是由 Carter 和 Wegman[42,212]引入的。论文[212]中首次将强泛 Hash 函数族用于认证;它同时也引入了几乎强泛 Hash 函数族(almost strongly universal hash families)的概念,这使得无条件安全 MAC 的密钥长度大大减少。更多相关的文章可在 Stinson[206]中查到。最后,需要注意的是,泛 Hash 函数族也被用于构造计算上有效的安全 MAC;这样的 MAC 被称为 UMAC,在 Black[25]中有详述。

练习

4.1 假定 $h:\mathcal{X}\to\mathcal{Y}$是一个$(N,M)$-Hash 函数,对任意的 $y\in\mathcal{Y}$,令

$$h^{-1}(y) = \{x : h(x) = y\}$$

记 $s_y = |h^{-1}(y)|$，定义

$$S = |\{\{x_1, x_2\} : h(x_1) = h(x_2)\}|$$

注意，S 表示$\mathcal{X}$中在 h 下碰撞的无序对的个数。

(a)证明：

$$\sum_{y \in \mathcal{Y}} s_y = N$$

这样 s_y 的平均值就是$\bar{s} = \frac{N}{M}$。

(b)证明：

$$S = \sum_{y \in \mathcal{Y}} \binom{s_y}{2} = \frac{1}{2} \sum_{y \in \mathcal{Y}} s_y^2 - \frac{N}{2}$$

(c)证明：

$$\sum_{y \in \mathcal{Y}} (s_y - \bar{s})^2 = 2S + N - \frac{N^2}{M}$$

(d)利用(c)中证明的结果，证明：

$$S \geqslant \frac{1}{2}\left(\frac{N^2}{M} - N\right)$$

进一步证明，等式成立当且仅当对任意的 $y \in \mathcal{Y}$有 $s_y = \frac{N}{M}$。

4.2　同练习 4.1，假定 $h: \mathcal{X} \to \mathcal{Y}$是一个$(N, M)$-Hash 函数，对任意的 $y \in \mathcal{Y}$，令

$$h^{-1}(y) = \{x : h(x) = y\}$$

用 ϵ 表示$h(x_1) = h(x_2)$的概率，其中 x_1 和 x_2 是$\mathcal{X}$中的任意元素(可相同)。证明：

$$\epsilon \geqslant \frac{1}{M}$$

等式成立当且仅当对任意的 $y \in \mathcal{Y}$，有

$$|h^{-1}(y)| = \frac{N}{M}$$

4.3　假定 $h: \mathcal{X} \to \mathcal{Y}$是一个$(N, M)$-Hash 函数，对任意的 $y \in \mathcal{Y}$，令

$$h^{-1}(y) = \{x : h(x) = y\}$$

且令 $s_y = |h^{-1}(y)|$。假定我们要利用算法 4.1 来解决函数 h 的原像问题，条件是我们只有 h 的预言器访问权限。对一个给定的 $y \in \mathcal{Y}$，设$\mathcal{X}_0$ 为个数为 q 的$\mathcal{X}$的任一子集。

(a) 证明：给定 y，算法 4.1 的成功概率为：

$$1-\frac{\binom{N-s_y}{q}}{\binom{N}{q}}$$

(b) 证明:对于所有的 $y\in\mathcal{Y}$,算法 4.1 的平均成功概率为:

$$1-\frac{1}{M}\sum_{y\in\mathcal{Y}}\frac{\binom{N-s_y}{q}}{\binom{N}{q}}$$

(c) 当 $q=1$ 时,说明(b)中的成功可能性为 1/M。

4.4 假定 $h:\mathcal{X}\rightarrow\mathcal{Y}$是一个$(N,M)$-Hash 函数,对任意的 $y\in\mathcal{Y}$,令

$$h^{-1}(y)=\{x:h(x)=y\}$$

且令 $s_y=|h^{-1}(y)|$。假定我们要利用算法 4.2 来解决函数 h 的第二原像问题,条件是我们只有 h 的预言器访问权限。对一个给定的 $x\in\mathcal{Y}$,$\mathcal{X}_0$ 为个数为 $q-1$ 的$\mathcal{X}\setminus\{x\}$的任一子集。

(a)证明:给定 x,算法 4.2 的成功概率为:

$$1-\frac{\binom{N-s_y}{q-1}}{\binom{N-1}{q-1}}$$

(b) 证明:对于所有的 $x\in\mathcal{X}$,算法 4.2 的平均成功概率为:

$$1-\frac{1}{N}\sum_{y\in\mathcal{Y}}\frac{s_y\binom{N-s_y}{q-1}}{\binom{N-1}{q-1}}$$

(c) 当 $q=2$ 时,说明(b)中的成功概率为:

$$\frac{\sum_{y\in\mathcal{Y}}s_y^2}{N(N-1)}-\frac{1}{N-1}$$

4.5 如果我们定义一个 Hash 函数(或压缩函数)h,它把一个 n 比特的二元串压缩成 m 比特的二元串。我们可以把 h 看做一个从$\mathbb{Z}_{2^n}$到$\mathbb{Z}_{2^m}$的函数。它试图以模 2^m 的整数运算来定义 h。我们在本练习中说明这种类型的一些简单构造是不安全的,应该避免。

(a) 假定 $n=m$,$m>1$,$h:\mathbb{Z}_{2^m}\rightarrow\mathbb{Z}_{2^m}$被定义为:

$$h(x)=x^2+ax+b \bmod 2^m$$

证明:对任意的 $x\in\mathbb{Z}_{2^m}$,无需解二次方程式,就很容易解决第二原像问题。

(b) 假定 $n>m$ 且 $h:\mathbb{Z}_{2^n}\rightarrow\mathbb{Z}_{2^m}$被定义为一个 d 次多项式:

$$h(x) = \sum_{i=0}^{d} a_i x^i \bmod 2^m$$

其中 $a_i \in \mathbb{Z}, 0 \leqslant i \leqslant d$。证明：对任意的 $x \in \mathbb{Z}_{2^n}$，无需解二次方程式，就很容易解决第二原像问题。

4.6　假定 $f: \{0,1\}^m \to \{0,1\}^m$ 是一个原像稳固的双射。定义 $h: \{0,1\}^{2m} \to \{0,1\}^m$ 如下：给定 $x \in \{0,1\}^{2m}$，记做：

$$x = x' \parallel x''$$

其中 $x', x'' \in \{0,1\}^m$，然后定义

$$h(x) = f(x' \oplus x'')$$

证明：h 不是第二原像稳固的。

4.7　对 $M = 365, 15 \leqslant q \leqslant 30$，比较定理 4.4 公式中给出的 ϵ 的准确值和定理证明中对 ϵ 的估计值。

4.8　假定 $h: \mathcal{X} \to \mathcal{Y}$ 是一个 Hash 函数，其中 $|\mathcal{X}|$ 和 $|\mathcal{Y}|$ 是有限值且 $|\mathcal{X}| \geqslant 2|\mathcal{Y}|$。假定 H 是平衡的，也就是说，对所有 $y \in \mathcal{Y}$，有

$$|h^{-1}(y)| = \frac{|\mathcal{X}|}{|\mathcal{Y}|}$$

最后，假定对给定的 Hash 函数 h，OraclePreimage 是一个针对原像的 (ϵ, q)-算法。证明：对给定的 Hash 函数 h，CollisionToPreimage 是一个针对碰撞的 $(\epsilon/2, q+1)$-算法。

4.9　假定 $h_1: \{0,1\}^{2m} \to \{0,1\}^m$ 是一个碰撞稳固的 Hash 函数。

(a) 定义 $h_2: \{0,1\}^{4m} \to \{0,1\}^m$ 如下：

1. 将 $x \in \{0,1\}^{4m}$ 写做 $x = x_1 \parallel x_2$，其中 $x_1, x_2 \in \{0,1\}^{2m}$。

2. 定义 $h_2(x) = h_1(h_1(x_1) \parallel h_1(x_2))$。

证明：h_2 是碰撞稳固的。

(b) 对整数 $i \geqslant 2$，从 h_{i-1} 递归定义 Hash 函数 $h_i: \{0,1\}^{2^i m} \to \{0,1\}^m$ 如下：

1. 将 $x \in \{0,1\}^{2^i m}$ 写做 $x = x_1 \parallel x_2$，其中 $x_1, x_2 \in \{0,1\}^{2^{i-1} m}$。

2. 定义 $h_i(x) = h_1(h_{i-1}(x_1) \parallel h_{i-1}(x_2))$。

证明：h_i 是碰撞稳固的。

4.10　在本练习中，我们考虑 Merkle-Damgård 结构的一个简化版本。假定

$$\text{compress}: \{0,1\}^{m+t} \to \{0,1\}^m$$

其中 $t \geqslant 1$，并且假定

$$x = x_1 \parallel x_2 \parallel \cdots \parallel x_k$$

其中

$$|x_1| = |x_2| = \cdots = |x_k| = t$$

我们研究下面的迭代 Hash 函数：

算法 4.9 简化的 Merkle-Damgård(x, k, t)

external compress

$z_1 \leftarrow 0^m \| x_1$

$g_1 \leftarrow$ compress(z_1)

for $i \leftarrow 1$ **to** $k-1$

do { $z_{i+1} \leftarrow g_i \| x_{i+1}$; $g_{i+1} \leftarrow$ compress(z_{i+1}) }

$h(x) \leftarrow g_k$

return($h(x)$)

假定 compress 是碰撞稳固的,进一步假定 compress 是零原像稳固的,也就是说,难以找到满足 compress(z) $= 0^m$ 的 $z \in \{0,1\}^{m+t}$。在这些假定条件下,证明 h 是碰撞稳固的。

4.11 可以不用 CBC 模式,而用分组密码的 CFB 模式来产生一个消息认证码。给定系列明文分组 $x_1, \cdots, x_n$,假定我们定义初始向量 IV 为 x_1。在 CFB 模式中用密钥 K 加密序列 $x_2, \cdots, x_n$,得到密文序列 $y_1, \cdots, y_{n-1}$(注意,只有 $n-1$ 个密文分组)。最后,定义 MAC 为 $e_K(y_{n-1})$。证明:该 MAC 与 3.7 节中用 CBC 模式产生的 MAC 相同。

4.12 假定$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$是一个满同态加密体制,其中$\mathcal{P} = \mathcal{C} = \{0,1\}^m$。令 $n \geqslant 2$ 为一个整数,定义 Hash 函数族$(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$,$\mathcal{X} = (\{0,1\}^m)^n$,$\mathcal{Y} = \{0,1\}^m$ 如下:

$$h_K(y_1, \cdots, y_n) = e_K(y_1) \oplus \cdots \oplus e_K(y_n)$$

按如下方式证明$(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$不是一个安全的消息认证码:

(a) 证明该 Hash 函数族存在一个(1,1)-假冒。

(b) 证明该 Hash 函数族存在一个(1,2)-假冒:对任意的消息$(y_1, \cdots, y_n)$,均可伪造 MAC(这种伪造被称为选择假冒(selective forgery);这些假冒在前面被看做存在假冒(existential forgery)的例子)。注意,当 $y_1 = \cdots = y_n$ 时情况较复杂。

4.13 假定$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$是一个满同态密码体制,其中$\mathcal{P} = \mathcal{C} = \{0,1\}^m$。令 $n \geqslant 2$ 为一个整数,定义 Hash 函数族$(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$,$\mathcal{X} = (\{0,1\}^m)^n$,$\mathcal{Y} = \{0,1\}^m$ 如下:

$$h_K(y_1, \cdots, y_n) = e_K(y_1) + 3e_K(y_2) + \cdots + (2n-1)e_K(y_n) \bmod 2^m$$

(a) 当 n 为奇数时,证明该 Hash 函数族存在 (1,2)-假冒。

(b) 当 $n = 2$ 时,证明该 Hash 函数族存在如下(1/8,2)-假冒:

1. 求(x, y)和(y, x)的 MAC。假定 $a = h_K(x, y)$,$b = h_K(y, x)$。
2. 证明只有 8 个满足:$x' = e_K(x)$,$y' = e_K(y)$的有序对(x', y')与给定的 MAC 的 a, b 值相一致。
3. 为 x'随机选择 8 个值,求出可能的假冒$(x, x), x'$。证明该假冒有效的概率为 1/8。

(c) 证明该 Hash 函数族存在 (1,3)-假冒:对任意的消息$(y_1,\cdots,y_n)$,均可假冒 MAC。

4.14　假定$(\mathcal{X},\mathcal{Y},\mathcal{C},\mathcal{H})$是一个强泛$(N,M)$-Hash 函数族。

(a) 如果$|\mathcal{K}|=M^2$,证明该 Hash 函数族存在(1,2)-假冒(即 $Pd_2=1$)。

(b) (下面推广(a)中证明的结果。)记 $\lambda=|\mathcal{K}|/M^2$。证明该 Hash 函数族存在$(1/\lambda,2)$-假冒(即 $Pd_2\geqslant 1/\lambda$)。

4.15　计算以下矩阵所表示的消息认证码的 Pd_0 和 Pd_1。

密　　钥	1	2	3	4
1	1	1	2	3
2	1	2	3	1
3	2	1	3	1
4	2	3	1	2
5	3	2	1	3
6	3	3	2	1

4.16　设 p 为奇素数。对于 $a,b\in\mathbb{Z}_p$,按规则

$$f_{(a,b)}(x)=(x+a)^2+b \bmod p$$

定义 $f_{(a,b)}:\mathbb{Z}_p\to\mathbb{Z}_p$。证明$(\mathbb{Z}_p,\mathbb{Z}_p,\mathbb{Z}_p\times\mathbb{Z}_p,\{f_{(a,b)}:a,b\in\mathbb{Z}_p\})$是一个强泛的$(p,p)$-Hash 族。

4.17　设 $k\geqslant 1$ 为整数。一个(N,M)-Hash 族$(\mathcal{X},\mathcal{Y},\mathcal{K},\mathcal{H})$称为 k 强泛的,如果对任意 k 个不同的$x_1,x_2,\cdots,x_k\in\mathcal{X}$以及对 k 个独立的$y_1,y_2,\cdots,y_k\in\mathcal{Y}$,满足以下条件:

$$\left|\{K\in\mathcal{K}:h_K(x_i)=y_i,1\leqslant i\leqslant k\}\right|=\frac{|\mathcal{K}|}{M^k}$$

(a) 证明对所有满足 $1\leqslant l\leqslant k$ 的 l,一个 k 强泛的 Hash 族是一个 l 强泛的。

(b) 设 p 是素数,$k\geqslant 1$ 是整数。对所有的 k 元组$(a_0,\cdots,a_{k-1})\in(\mathbb{Z}_p)^k$,按规则

$$f_{(a_0,\cdots,a_{k-1})}(x)=\sum_{i=0}^{k-1}a_ix^i \bmod p$$

定义 $f_{(a_0,\cdots,a_{k-1})}:\mathbb{Z}_p\to\mathbb{Z}_p$。证明:$(\mathbb{Z}_p,\mathbb{Z}_p,(\mathbb{Z}_p)^k,\{f_{(a_0,\cdots,a_{k-1})}:(a_0,\cdots,a_{k-1})\in(\mathbb{Z}_p)^k\})$是 k 强泛的(p,p)-Hash 族。

提示:使用域上的 d 次多项式最多有 d 个根这一事实。

第 5 章　RSA 密码体制和整数因子分解

5.1　公钥密码学简介

到目前为止，在我们研究的经典密码学模型中，Alice 和 Bob 秘密地选择密钥 K。K 于是给出了一条加密规则 e_K 和一条解密规则 d_K。在这些密码体制中，d_K 或者跟 e_K 相同，或者可以容易地从 e_K 导出(例如，DES 解密等同于加密，但是密钥方案是相反的)。由于 e_K 或者 d_K 的泄露会导致系统不安全，这种类型的密码体制称为对称密钥密码体制。

对称密钥密码体制的一个缺点是，它需要在 Alice 和 Bob 传输密文之前使用一个安全的通道交换密钥。实际上，这可能很难达到。例如，假定 Alice 和 Bob 相距遥远，他们决定用 E-mail 通信，在这种情况下，Alice 和 Bob 可能无法获得一个相当安全的通道。

在公钥密码体制中的一个想法就是：可以找到一个密码体制，使得由给定的 e_K 来求 d_K 是计算不可行的。如果这样的话，加密规则 e_K 是一个公钥，可以在一个目录中发布(这也就是公钥体制名称的由来)。公钥体制的优点就是 Alice(或者其他任何人)可以利用公钥加密规则 e_K 发出一条加密的消息给 Bob，而不需要预先的共享密钥的通信。Bob 将是惟一能够利用解密规则 d_K(称为私钥)对密文进行解密的人。

考虑如下的类比：Alice 在一个金属盒子里放入一件东西，利用号码锁锁住留给 Bob。由于只有 Bob 知道号码，他是惟一能打开盒子的人。

公钥密码体制的思想是在 1976 年由 Diffie 和 Hellman 提出的。然后在 1977 年由 Rivest、Shamir 和 Adleman 发明了著名的 RSA 密码体制，这将在本章中介绍。此后，几个公钥密码体制被提出，其安全性依赖于不同的计算问题。其中，最著名的是 RSA 密码体制(及其变种)，其安全性基于分解大整数的困难性；ElGamal 密码体制(及其变种，例如，椭圆曲线密码体制)，其安全性基于离散对数问题。我们在本章中讨论 RSA 密码体制和它的变种，ElGamal 密码体制将在第 6 章中介绍。

在 Diffie 和 Hellman 之前，公钥密码学的思想已经由 James Ellis 在 1970 年 1 月的一篇题为“非秘密加密的可能性”的文章中(短语“非秘密加密”即是公钥密码学)提出。James Ellis 是电子通讯安全小组(CESG)的成员，这个小组是英国政府通信总部(GCHQ)的一个特别部门。这篇文章没有在公共文献中发表，而是在 1997 年 12 月由 GCHQ 正式解密的五篇文章中的一篇。在这五篇文章中，还有一篇是 Clifford Cocks 在 1973 年发表的题为“关于非秘密加密的注记”的文章，其中描述的公钥密码体制跟 RSA 密码体制基本一致。

一个重要的事实就是公钥密码体制无法提供无条件安全性。这是因为一个敌手可通过观察密文 y，利用公钥规则 e_K 来加密每一条可能的明文，直到他发现惟一的 x 使得 $y = e_K(x)$ 为止，这个 x 就是 y 的解密。所以，我们仅研究公钥体制的计算安全性。

考虑一种称为陷门单向函数(trapdoor one-way function)的抽象,对于研究公钥密码体制是很有帮助的。我们现在给出它的非正式定义。

Bob 的公开加密函数应该是容易计算的。我们注意到计算其逆函数(即解密函数)应该是困难的(对于除 Bob 以外的人)。回顾 4.2 节,一个函数容易计算但难于求逆,通常称为单向的。在加密过程中,我们希望加密函数 e_K 为一个单向的单射函数,以便可以解密。不幸的是,尽管有很多单射函数被认为是单向的,但是还没有一个函数能被证明是单向的。

例如,有如下一个函数被认为是单向的,假定 n 为两个大素数 p 和 q 的乘积,b 为一个正整数,那么定义 $f:\mathbb{Z}_n \rightarrow \mathbb{Z}_n$ 为:

$$f(x) = x^b \bmod n$$

(如果 $\gcd(b, \phi(n)) = 1$,那么事实上这是 RSA 加密函数;我们将在后面给出详细描述)。

如果我们要构造一个公钥密码体制,仅给出一个单向的单射函数是不够的。从 Bob 的观点来看,并不需要 e_K 是单向的,因为他需要用有效的方式解密所收到的信息。因此,Bob 应该拥有一个陷门,其中包含容易求出 e_K 的逆函数的秘密信息。也就是说,Bob 可以有效地解密,因为他有额外的秘密知识,即 K,能够提供给他解密函数 d_K。因此,我们称一个函数为一个陷门单向函数,如果它是一个单向函数,并在具有特定陷门的知识后容易求出其逆。

考虑上面的函数 $f(x) = x^b \bmod n$。我们将在 5.3 节中看到其逆函数 f^{-1} 有类似的形式:$f(x) = x^a \bmod n$,对于合适的取值 a。陷门就是利用 n 的因子分解,有效地算出正确的指数 a(对于给定的 b)。

为方便起见,我们把特定的某类陷门单向函数记为$\mathcal{F}$。那么随机选取一个函数 $f \in \mathcal{F}$,作为公开加密函数;其逆函数 f^{-1} 是秘密解密函数。这类似于在对称密钥密码体制中,从特定的密钥空间随机选取一个密钥。

本章的其余内容按如下组织。5.2 节介绍了几个重要的数论结果。5.3 节开始讨论 RSA 密码体制。5.4 节提供了一些重要的素性检验的方法。5.5 节简要讨论了关于模 n 的平方根的存在性,然后在 5.6 节中提供了一些分解因子的方法。5.7 节考虑了其他一些对 RSA 密码体制的攻击,并在 5.8 节中描述了 Rabin 密码体制。最后,在 5.9 节中讨论了 RSA 类密码体制的语义安全性。

5.2 更多的数论知识

在描述 RSA 密码体制如何工作之前,我们需要讨论一些关于模算法和数论的知识。我们需要两个重要的工具:Euclidean 算法和中国剩余定理。

5.2.1 Euclidean 算法

在第 1 章中我们已经知道对于任一正整数 n,$\mathbb{Z}_n$ 是一个环。我们也证明了 $b \in \mathbb{Z}_n$ 有一个乘法逆当且仅当 $\gcd(b, n) = 1$;小于 n 且与 n 互素的正整数的个数为 $\phi(n)$。

模 n 的余数与 n 互素的全体记为$\mathbb{Z}_n^*$。容易看到,$\mathbb{Z}_n^*$ 在乘法下形成一个阿贝尔群。我们已

经证明模 n 乘法是可结合的和可交换的，1 为乘法单位元。$\mathbb{Z}_n^*$ 中任一元素有一个乘法逆(也在 $\mathbb{Z}_n^*$ 中)。最后，$\mathbb{Z}_n^*$ 是乘法封闭的，由于当 x 和 y 与 n 互素时，xy 与 n 互素(试证明这一点)。

在这种观点下，我们知道任一 $b \in \mathbb{Z}_n^*$ 都有一个乘法逆 b^{-1}，但还没给出有效的算法求出 b^{-1}。存在这样的一个算法，称为扩展 Euclidean 算法。我们首先描述 Euclidean 算法的基本形式，它可以给出两个正整数 a 和 b 的最大公因子。Euclidean 算法首先令 r_0 为 a，令 r_1 为 b，然后执行如下除法运算：

$$\begin{aligned}
r_0 &= q_1 r_1 + r_2, && 0 < r_2 < r_1 \\
r_1 &= q_2 r_2 + r_3, && 0 < r_3 < r_2 \\
&\vdots \quad \vdots \quad \vdots && \vdots \\
r_{m-2} &= q_{m-1} r_{m-1} + r_m, && 0 < r_2 < r_1 \\
r_{m-1} &= q_m r_m
\end{aligned}$$

Euclidean 算法的伪代码在算法 5.1 中给出。

算法 5.1　Euclidean Algorithm(a, b)

$r_0 \leftarrow a$

$r_1 \leftarrow b$

$m \leftarrow 1$

while　$r_m \neq 0$

do
- $q_m \leftarrow \left\lfloor \dfrac{r_{m-1}}{r_m} \right\rfloor$
- $r_{m+1} \leftarrow r_{m-1} - q_m r_m$
- $m \leftarrow m + 1$

$m \leftarrow m - 1$

return$(q_1, \cdots, q_m; r_m)$

comment: $r_m = \gcd(a, b)$

注　我们将在本章后面用表$(q_1, \cdots, q_m)$来表示算法 5.1 计算过程中出现的数值。

在算法 5.1 中，容易看到

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots \gcd(r_{m-1}, r_m) = r_m$$

因此可以得出 $\gcd(a, b) = r_m$。

由于 Euclidean 算法能计算出最大公因子，它可以用来判断一个正整数 $b < n$ 是否有模 n 的乘法逆，通过调用 Euclidean 算法(n, b)来看一下是否有 $r_m = 1$。但是，这并没有计算出其值 $b^{-1} \bmod n$(如果逆存在)。

现在，假定我们按下面的构造定义了两个数列(其中 q_j 按算法 5.1 定义)：$t_0, t_1, \cdots, t_m$

和 $s_0, s_1, \cdots, s_m$,其中

$$t_j = \begin{cases} 0 & j = 0 \\ 1 & j = 1 \\ t_{j-2} - q_{j-1} t_{j-1} & j \geqslant 2 \end{cases}$$

和

$$s_j = \begin{cases} 1 & j = 0 \\ 0 & j = 1 \\ s_{j-2} - q_{j-1} s_{j-1} & j \geqslant 2 \end{cases}$$

那么我们有如下有用的结论。

定理 5.1 对于 $0 \leqslant j \leqslant m$,有 $r_j = s_j r_0 + t_j r_1$,其中 r_j 按算法5.1定义,s_j 和 t_j 按上面定义。

证明 对 j 用归纳法。对于 $j = 0$ 和 $j = 1$,命题是平凡的。假定命题对于 $j = i - 1$ 和 $j = i - 2$ 成立,其中 $i \geqslant 2$;我们将证明命题对于 $j = i$ 也是成立的。由归纳假定,我们有:

$$r_{i-2} = s_{i-2} r_0 + t_{i-2} r_1$$

和

$$r_{i-1} = s_{i-1} r_0 + t_{i-1} r_1$$

现在我们计算

$$\begin{aligned} r_i &= r_{i-2} - q_{i-1} r_{i-1} \\ &= s_{i-2} r_0 + t_{i-2} r_1 - q_{i-1} (s_{i-1} r_0 + t_{i-1} r_1) \\ &= (s_{i-2} - q_{i-1} s_{i-1}) r_0 + (t_{i-2} q_{i-1} t_{i-1}) r_1 \\ &= s_i r_0 + t_i r_1 \end{aligned}$$

因此,由归纳法,命题对于所有整数 $j \geqslant 0$ 是正确的。

算法 5.2 Extended Euclidean Algorithm(a, b)

$a_0 \leftarrow a$

$b_0 \leftarrow b$

$t_0 \leftarrow 0$

$t \leftarrow 1$

$s_0 \leftarrow 1$

$s \leftarrow 0$

$q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$

$r \leftarrow a_0 - q b_0$

while $r > 0$

do
- $temp \leftarrow t_0 - qt$
- $t_0 \leftarrow t$
- $t \leftarrow temp$
- $temp \leftarrow s_0 - qs$
- $s_0 \leftarrow s$
- $s \leftarrow temp$
- $a_0 \leftarrow b_0$
- $b_0 \leftarrow r$
- $q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
- $r \leftarrow a_0 - qb_0$

$r \leftarrow b_0$

return (r, s, t)

comment: $r = \gcd(a, b)$且 $sa + tb = r$

在算法 5.2 中，我们给出扩展 Euclidean 算法，它以两个整数 a 和 b 作为输入，计算出整数 r、s 和 t，使得 $r = \gcd(a, b)$且 $sa + tb = r$。在这个算法的版本中，我们不必记录所有的 q_j、r_j、s_j 和 t_j；在算法的任一点上，记录每个数列的最后两项就足够了。

下面推论可以由定理 5.1 直接推出。

推论 5.2　假定 $\gcd(r_0, r_1) = 1$，那么 $r_1^{-1} \bmod r_0 = t_m \bmod r_0$。

证明　由定理 5.1，我们有：

$$1 = \gcd(r_0, r_1) = s_m r_0 + t_m r_1$$

两边模 r_0 约化，我们得到：

$$t_m r_1 \equiv 1 \pmod{r_0}$$

即得所证。

我们提供了一个例子来说明算法，其中给出了所有 s_j、t_j、q_j 和 r_j 的值。

例 5.1　假定我们要计算 $28^{-1} \bmod 75$。那么我们如下计算：

i	r_i	q_i	s_i	t_i
0	75		1	0
1	28	2	0	1
2	19	1	1	-2
3	9	2	-1	3
4	1	9	3	-8

因此,我们发现

$$3 \times 75 - 8 \times 28 = 1$$

应用推论 5.2,我们得到

$$28^{-1} \bmod 75 = -8 \bmod 75 = 67$$

扩展 Euclidean 算法立即得出数值 $b^{-1} \bmod a$(如果存在)。事实上,乘法逆 $b^{-1} \bmod a = t \bmod a$;这可由推论 5.2 直接导出。然而,一个更有效的算法是把所有 s_j 的计算从算法 5.2 中去掉,并在主循环中每一次求 t 时都进行模 a 运算。于是我们得到算法 5.3。

算法 5.3 Multiplicative Inverse(a,b)

$a_0 \leftarrow a$

$b_0 \leftarrow b$

$t_0 \leftarrow 0$

$t \leftarrow 1$

$q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$

$r \leftarrow a_0 - qb_0$

while $r > 0$

do
- $temp \leftarrow (t_0 - qt) \bmod a$
- $t_0 \leftarrow t$
- $t \leftarrow temp$
- $a_0 \leftarrow b_0$
- $b_0 \leftarrow r$
- $q \leftarrow \left\lfloor \frac{a_0}{b_0} \right\rfloor$
- $r \leftarrow a_0 - qb_0$

if $b_0 \neq 1$

then b 没有模 a 的逆

else return (t)

5.2.2　中国剩余定理

中国剩余定理是求解某类特定同余方程组的一个好办法。假定 $m_1,\cdots,m_r$ 为两两互素的正整数(即当 $i\neq j$ 时 $\gcd(m_i,m_j)=1$)。假定 $a_1,\cdots,a_r$ 为整数,考虑如下的同余方程组:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \quad \vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

中国剩余定理断言这个方程组有模 $M=m_1\times m_2\times\cdots\times m_r$ 的惟一解。我们将在本节中证明这个结论,并给出求解这种类型的同余方程组的有效算法。

为方便起见,我们研究函数 $\chi:\mathbb{Z}_M\rightarrow\mathbb{Z}_{m_1}\times\cdots\times\mathbb{Z}_{m_r}$ 按如下定义:

$$\chi(x)=(x \bmod m_1,\cdots,x \bmod m_r)$$

例 5.2　假定 $r=2,m_1=5$ 且 $m_2=3$,那么 $M=15$。于是函数取 χ 值如下:

$$\begin{aligned} &\chi(0)=(0,0) \quad \chi(1)=(1,1) \quad \chi(2)=(2,2) \\ &\chi(3)=(3,0) \quad \chi(4)=(4,1) \quad \chi(5)=(0,2) \\ &\chi(6)=(1,0) \quad \chi(7)=(2,1) \quad \chi(8)=(3,2) \\ &\chi(9)=(4,0) \quad \chi(10)=(0,1) \quad \chi(11)=(1,2) \\ &\chi(12)=(2,0) \quad \chi(13)=(3,1) \quad \chi(14)=(4,2) \end{aligned}$$

证明中国剩余定理就等于证明函数 χ 是一个双射。在例 5.2 中容易看出是一个双射。事实上,我们可以给出逆函数 χ^{-1} 的显式公式。

对于 $1\leqslant i\leqslant r$,定义

$$M_i=\frac{M}{m_i}$$

那么容易看到

$$\gcd(M_i,m_i)=1$$

对于 $1\leqslant i\leqslant r$ 成立。下一步,对于 $1\leqslant i\leqslant r$,定义

$$y_i=M_i^{-1} \bmod m_i$$

(逆存在是因为 $\gcd(M_i,m_i)=1$,且可以用算法 5.3 找到。)注意,

$$M_iy_i=1 \pmod{m_i}$$

对于 $1\leqslant i\leqslant r$ 成立。

现在,定义一个函数 $\rho:\mathbb{Z}_{m_1}\times\cdots\times\mathbb{Z}_{m_r}\rightarrow\mathbb{Z}_M$ 如下:

$$\rho(a_1,\cdots,a_r)=\sum_{i=1}^{r}a_iM_iy_i \mod M$$

我们将证明函数 $\rho=\chi^{-1}$,即,它提供了一个求解原来的同余方程组的显式公式。

记 $X=\rho(a_1,\cdots a_r)$,令 $1\leqslant j\leqslant r$。考虑上面和式中的项 $a_iM_iy_i$,模 m_j 约化:如果 $i=j$,那么

$$a_iM_iy_i=a_i(\mathrm{mod}\ \ m_i)$$

因为,

$$M_iy_i=1\ (\mathrm{mod}\ \ m_i)$$

另一方面,如果 $i\neq j$,那么,

$$a_iM_iy_i=0\ (\mathrm{mod}\ \ m_j)$$

因为在这种情形下 $m_j\mid M_i$。因此我们有:

$$\begin{aligned}X&\equiv\sum_{i=1}^{r}a_iM_iy_i(\mathrm{mod}\ m_j)\\&\equiv a_j(\mathrm{mod}\ m_j)\end{aligned}$$

由于上式对所有的 $j,1\leqslant j\leqslant r$ 成立,所以 X 是同余方程组的一个解。

到现在为止,我们还需证明 X 模 M 是惟一的。这可以通过简单的计数来做到。函数 χ 是从基数为 M 的定义域到基数为 M 的值域的映射。我们已经证明 χ 是一个满射。因此,χ 也必须是单射,由于定义域和值域有相同的基数。于是 χ 是一个双射,且 $\chi^{-1}=\rho$。也可看出 χ^{-1}是它的参数 $a_1,\cdots a_r$ 的一个线性函数。

下面用一个大一点的例子来说明上述过程。

例 5.3 假定 $r=3,m_1=7,m_2=11$ 且 $m_3=13$,那么 $M=1001$。我们计算 $M_1=143,M_2=91$ 且 $M_3=77$,于是 $y_1=5,y_2=4$ 且 $y_3=12$。那么函数 $\chi^{-1}:\mathbb{Z}_7\times\mathbb{Z}_{11}\times\mathbb{Z}_{13}\rightarrow\mathbb{Z}_{1001}$ 如下:

$$\chi^{-1}(a_1,a_2,a_3)=(715a_1+364a_2+924a_{3_4})\ \mathrm{mod}\ 1001$$

例如,如果 $x\equiv5\ (\mathrm{mod}\ 7),x\equiv3\ (\mathrm{mod}\ 11)$且 $x\equiv10\ (\mathrm{mod}\ 13)$,那么这个公式告诉我们,

$$\begin{aligned}x&=(715\times5+364\times3+924\times10)\ \mathrm{mod}\ 1001\\&=13\ 907\ \mathrm{mod}\ 1001\\&=894\end{aligned}$$

这可以通过用 894 模 7、11 和 13 约化来验证。

为方便后面引用,我们将本小节的结论记为定理 5.3。

定理 5.3(中国剩余定理) 假定 $m_1,\cdots,m_r$ 为两两互素的正整数,又假定 $a_1,\cdots,a_r$ 为整数。那么同余方程组 $x\equiv a_i(\mathrm{mod}\ m_i)(1\leqslant i\leqslant r)$有模 $M=m_1\times m_2\times\cdots\times m_r$ 的惟一解,由下式给出:

$$x = \sum_{i=1}^{r} a_i M_i y_i \mod M$$

其中 $M_i = \frac{M}{m_i}$，且 $y_i = M_i^{-1} \bmod m_i$，$1 \leqslant i \leqslant r$。

5.2.3 其他结论

下面我们将提到来自基本群理论的另一结果，称为 Lagrange 定理，这对处理 RSA 密码体制有很大关系。对于一个(有限)乘法群 G，定义元素 $g \in G$ 的阶(order)为使得 $g^m = 1$ 的最小的正整数 m。下面的结论是相当简单的，我们就不再给出证明。

定理 5.4 (Lagrange)　假定 G 是一个阶为 n 的乘法群，且 $g \in G$。那么 g 的阶整除 n。

基于我们的意图，如下的推论是基本的。

推论 5.5　如果 $b \in \mathbb{Z}_n^*$，那么 $b^{\phi(n)} \equiv 1 \pmod n$。

证明　$\mathbb{Z}_n^*$ 是阶为 $\phi(n)$ 的乘法群。

推论 5.6 (Fermat)　假定 p 是一个素数，且 $b \in \mathbb{Z}_p$，那么 $b^p \equiv b \pmod p$。

证明　$\mathbb{Z}_n^*$ 是阶为 $\phi(n)$ 的乘法群。如果 p 是一个素数，那么 $\phi(p) = p - 1$。所以，对于 $b \neq 0 \pmod p$，结果可由推论 5.5 得到。对于 $b \equiv 0 \pmod p$，结果仍成立，因为 $0^p \equiv 0 \pmod p$。

到此为止，我们知道如果 p 是一个素数，那么 $\mathbb{Z}_p^*$ 是阶为 $p-1$ 的乘法群，且 $\mathbb{Z}_p^*$ 中任一元素的阶整除 $p-1$。事实上，如果 p 是一个素数，那么群 $\mathbb{Z}_p^*$ 是一个循环群(cyclic group)：存在一个元素 $\alpha \in \mathbb{Z}_p^*$ 其阶等于 $p-1$。我们不去证明这一重要事实，但记录下来以供后面参考。

定理 5.7　如果 p 是一个素数，那么 $\mathbb{Z}_p^*$ 群是一个循环群。

一个元素 α 具有模 p 的阶 $p-1$，称为一个模 p 的本原元素。易知 α 是一个模 p 的本原元素当且仅当

$$\{\alpha^i : 0 \leqslant i \leqslant p-2\} = \mathbb{Z}_p^*$$

现在假定 p 是一个素数且 α 是一个模 p 的本原元素。任一元素 $\beta \in \mathbb{Z}_p^*$ 可以写成惟一的形式 $\beta = \alpha^i$，其中 $0 \leqslant i \leqslant p-2$。容易证明 $\beta = \alpha^i$ 的阶为：

$$\frac{p-1}{\gcd(p-1, i)}$$

于是 β 本身是一个模 p 的本原元素当且仅当 $\gcd(p-1, i) = 1$。于是模 p 的本原元素的个数为 $\phi(p-1)$。

我们用一个小例子来说明一下。

例 5.4 假定 $p = 13$。根据上面的结果可知有4个模 13 的本原元素。首先,通过计算 2 的连续幂次,我们可以验证 2 是一个模 13 的本原元素:

$$2^0 \bmod 13 = 1$$
$$2^1 \bmod 13 = 2$$
$$2^2 \bmod 13 = 4$$
$$2^2 \bmod 13 = 8$$
$$2^4 \bmod 13 = 3$$
$$2^5 \bmod 13 = 6$$
$$2^6 \bmod 13 = 12$$
$$2^7 \bmod 13 = 11$$
$$2^8 \bmod 13 = 9$$
$$2^9 \bmod 13 = 5$$
$$2^{10} \bmod 13 = 10$$
$$2^{11} \bmod 13 = 7$$

元素 2^i 是本原的当且仅当 $\gcd(i,12) = 1$,即,当且仅当 $i = 1,5,7,11$。因此,模 13 的本原元素为 2、6、7 和 11。

在上面的例子中,我们为了验证 2 是一个模 13 的本原元素要计算 2 的所有幂次。如果 p 是一个很大的素数,那将会花很长的时间来计算一个元素 $\alpha \in \mathbb{Z}_p^*$ 的 $p-1$ 个幂次。幸运的是,如果 $p-1$ 的因子分解已知,我们可以利用下面结果更快地判断 $\alpha \in \mathbb{Z}_p^*$ 是否为一个本原元素。

定理 5.8 假定 $p > 2$ 是一个素数,且 $\alpha \in \mathbb{Z}_p^*$。那么 α 是一个模 p 的本原元素当且仅当 $\alpha^{(p-1)/q} \neq 1 \pmod p$ 对于所有满足 $q \mid (p-1)$ 的素数 q 成立。

证明 如果 α 是一个模 p 的本原元素,那么对于所有的 $1 \leqslant i \leqslant p-2$,有 $\alpha^i \neq 1 \pmod p$,所以结果成立。

反过来,假定 $\alpha \in \mathbb{Z}_p^*$ 不是模 p 的本原元素。令 d 为 α 的阶。那么由 Lagrange 定理,有 $d \mid (p-1)$。因为 α 不是本原的,所以 $d < p-1$。那么 $(p-1)/d$ 是一个大于 1 的整数。令 q 为 $(p-1)/d$ 的素因子。那么 d 是 $(p-1)/q$ 的一个因子。由于 $\alpha^d \equiv 1 \pmod p$ 且 $d \mid (p-1)/q$,于是有 $\alpha^{(p-1)/q} \equiv 1 \pmod p$。

12 的因子分解为 $12 = 2^2 \times 3$。因此,在前面的例中,我们可以通过计算 $2^6 \neq 1 \pmod{13}$ 和 $2^4 \neq 1 \pmod{13}$ 来验证 2 是一个模 13 的本原元素。

5.3　RSA 密码体制

我们现在可以描述 RSA 密码体制。这个密码体制利用$\mathbb{Z}_n$ 中的计算，其中 n 是两个不同的奇素数p 和q 的乘积。对于这样一个整数 n，注意到 $\phi(n)=(p-1)(q-1)$。这个密码体制的正式描述如下。

密码体制 5.1　RSA 密码体制

设 $n=pq$。其中 p 和q 为素数。设$\mathcal{P}=\mathcal{C}=\mathbb{Z}_n$，且定义

$$\mathcal{K}=\{(n,p,q,a,b):ab\equiv 1 \pmod{\phi(n)}\}$$

对于 $K=(n,p,q,a,b)$，定义

$$e_K(x)=x^b \bmod n$$

和

$$d_K(y)=y^a \bmod n$$

$(x,y\in\mathbb{Z}_n)$。值 n 和b 组成了公钥，且值 p,q 和a 组成了私钥。

让我们验证加密和解密是逆运算。由于

$$ab\equiv 1 \pmod{\phi(n)}$$

我们有：

$$ab=t\phi(n)+1$$

对于某个整数 $t\geqslant 1$。假定 $x\in\mathbb{Z}_n^*$；那么我们有：

$$\begin{aligned}(x^b)^a &\equiv x^{t\phi(n)+1} \pmod{n}\\ &\equiv (x^{\phi(n)})^t x \pmod{n}\\ &\equiv 1^t x \pmod{n}\\ &\equiv x \pmod{n}\end{aligned}$$

这正是我们所期望的。我们把证明如下结论作为练习：如果 $x\in\mathbb{Z}_n\setminus\mathbb{Z}_n^*$，仍有$(x^b)^a\equiv x \pmod{n}$。

下面是一个描述 RSA 密码体制的小例子(不安全的)。

例 5.5　假定 Bob 选取 $p=101, q=113$。那么 $n=1143$ 和 $\phi(n)=100\times 112=11\,200$。由于 $11\,200=2^6 5^2 7$，一个整数 b 可以选为加密指数当且仅当b 不能被 2、5 或 7 整除(实际上，Bob 不会分解 $\phi(n)$。他将会利用算法 5.3 来验证 $\gcd(\phi(n),b)=1$，并同时计算出 b^{-1})。假定 Bob 选取 $b=3533$。那么，

$$b^{-1} \bmod \quad 11\,200 = 6597$$

因此,Bob 的秘密解密指数为 a = 6597。

Bob 在一个目录中发布 $n = 11\,413$ 和 $b = 3533$。现在,假定 Alice 想加密明文 9726 并发给 Bob。她将计算:

$$9726^{3533} \bmod 11\,413 = 5761$$

然后把密文 5761 通过信道发出。当 Bob 收到了密文 5761,他将用秘密解密指数来计算:

$$5761^{6597} \bmod \quad 11\,413 = 9726$$

(到目前为止,加密和解密过程可能看起来非常复杂,但我们将会在下一节中讨论这些运算的有效算法。)

RSA 密码体制的安全性是基于相信加密函数 $e_K(x) = x^b \bmod n$ 是一个单向函数,所以对于一个敌手来说,试图解密密文将是计算上不可行的。允许 Bob 解密密文的陷门是分解 $n = pq$ 的知识。由于 Bob 知道这个分解,他可以计算 $\phi(n) = (p-1)(q-1)$,然后用扩展 Euclidean 算法来计算解密指数 a。我们将在后面给出更多关于 RSA 密码体制安全性的论述。

5.3.1 实现 RSA

RSA 密码体制有很多方面可以讨论,包括实现细节、加密和解密的效率,以及安全问题。为了建立这个体制,Bob 使用了 RSA 参数生成算法,由下面的算法 5.4 非正式地给出。Bob 如何实现这个算法的细节问题,将在本章的后面讨论。

算法 5.4 RSA 参数生成算法

1. 生成两个大素数,p 和 q
2. $n \leftarrow pq$,且 $\phi(n) \leftarrow (p-1)(q-1)$
3. 选择一个随机数 $b(1 < b < \phi(n))$,使得 $\gcd(b, \phi(n)) = 1$
4. $a \leftarrow b^{-1} \bmod \phi(n)$
5. 公钥为(n, b),私钥为(p, q, a)

对于 RSA 密码体制的一个明显的攻击就是密码分析者试图分解 n。如果这一点做到了,那么很简单地可以计算 $\phi(n) = (p-1)(q-1)$,然后可以跟 Bob 一样地从 b 精确地算出解密指数 a(这里隐含假定了破解 RSA 密码体制是与分解 n 多项式等价的,但这一点仍未得到证明)。

如果一个 RSA 密码体制要成为安全的,显然要求 $n = pq$ 必须充分大,使得分解它是计算上不可行的。现在的分解算法能够分解由 512 个二进位表示的整数(关于分解因子的更多信息,可参看 5.6 节)。为了安全着想,一般推荐取 p, q 均为 512 比特的素数;那么 n 就是 1024 比特的模数。分解这样长度的整数就大大超出了现有的分解因子算法的能力。

先把如何寻找 512 比特的素数的问题放到一边，我们首先考察一下加密和解密的算术运算。一个加密(或解密)包含了模 n 的指数运算。由于 n 是很大的，我们必须用多精度运算来执行$\mathbb{Z}_n$ 上的运算，所需的时间将依赖于 n 的二进制表示位数。

假定 x 和 y 是 k 和 l 位二进制表示的正整数；即 $k = \lfloor \log_2 x \rfloor + 1, l = \lfloor \log_2 y + 1 \rfloor$。假定 $k \geqslant l$。用标准的初等算术技巧，可以容易地得到对 x 和 y 执行各种运算所需时间的上界估计。我们现在概要列出这些结果(我们并没有说这些是最好的上界估计)。

- $x + y$ 的计算时间复杂度为 $O(k)$
- $x - y$ 的计算时间复杂度为 $O(k)$
- xy 的计算时间复杂度为 $O(kl)$
- $\lfloor x/y \rfloor$ 的计算时间复杂度为 $O(l(k-l)), O(kl)$是一个弱估计
- $\gcd(x, y)$的计算时间复杂度为 $O(k^3)$

我们看一下最后一项，GCD 可以用算法 5.1 计算。易见 Euclidean 算法的迭代次数为$O(k)$(参见练习)。每一次迭代执行一次长除需要时间 $O(k^2)$；所以，GCD 算法的计算复杂度看起来为 $O(k^3)$(实际上，采用更为细致的分析可知 GCD 算法的复杂度为 $O(k^2)$)。

现在我们返回模算术，即$\mathbb{Z}_n$ 中的运算。假定 n 为一个 k 比特整数，且 $0 \leqslant m_1, m_2 \leqslant n-1$。设 c 为一个正整数。我们有如下的结果：

- $(m_1 + m_2) \bmod n$ 的计算时间复杂度为 $O(k)$
- $(m_1 - m_2) \bmod n$ 的计算时间复杂度为 $O(k)$
- $(m_1 m_2) \bmod n$ 的计算时间复杂度为 $O(k^2)$
- $(m_1)^{-1} \bmod n$ 的计算时间复杂度为 $O(k^3)$
- $(m_1)^c \bmod n$ 的计算时间复杂度为 $O(\log c) \times k^2)$

上面大多数结果并不难于证明。前三个运算(模加、模减、模乘)可有相应的整数运算完成，然后再执行一次模 n 求余即可。模逆(即计算乘法逆)可以用算法 5.1 完成。复杂度分析也类似于 GCD 运算。

我们现在考虑模指数，即计算形如 $x^c \bmod n$ 的函数。在 RSA 密码体制中，加密和解密运算都是模指数运算。计算 $x^c \bmod n$ 可以由 $c-1$ 次模乘来实现；然而，如果 c 非常大，其效率会很低。注意，c 可能跟 $\phi(n)-1$ 一样大，$\phi(n)-1$ 又几乎跟 n 一样大，且相对于 k 是指数阶大的。

著名的平方-乘算法可以把计算 $x^c \bmod n$ 所需模乘的次数降低为最多 $2l$ 次，其中 l 是 c 的二进制表示的位数。于是 $x^c \bmod n$ 可以在时间 $O(lk^2)$内算出。如果我们假定 $c < n$(如在 RSA 密码体制中所定义)，那么我们可以看到 RSA 加密和解密都可以在时间 $O(\log n)^3)$内完成，这是一个关于明文(或密文)中字符的比特数的一个多项式函数。

平方-乘算法假定指数 c 用二进制表示，即

$$c = \sum_{i=0}^{l-1} c_i 2^i$$

其中 $c_i = 0$ 或 $1, 0 \leqslant i \leqslant l-1$。计算 $z = x^c \bmod n$ 的算法如算法 5.5 所述。

算法 5.5 平方-乘 Square-and-Multiply(x, c, n)

$z \leftarrow 1$

for $i \leftarrow l-1$ **downto** 0

do { $z \leftarrow z^2 \bmod n$

if $c_i = 1$

then $z \leftarrow (z \times x) \bmod n$

return (z)

这个算法的正确性证明留做练习。容易得到算法中模乘的计算次数。总要执行 l 次平方运算。形如 $z \leftarrow (z \times x) \bmod n$ 的模乘的次数等于 c 的二进制表示中"1"的个数,这是一个介于 0 和 l 之间的整数。因此,如上所述,模乘的执行次数至少为 l,至多为 $2l$。

我们仍使用例 5.5 来描述平方-乘算法的作用。

例 5.5 (续) 这里 $n = 11\,413$,公开的加密指数为 $b = 3533$。Alice 利用平方-乘算法,通过计算 $9726^{3533} \bmod 11\,413$ 来加密明文 9726,过程如下:

i	b_i	z
11	1	$1^2 \times 9726 = 9726$
10	1	$9726^2 \times 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \times 9726 = 9167$
7	1	$9167^2 \times 9726 = 4958$
6	1	$4958^2 \times 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \times 9726 = 10\,185$
2	1	$10\,185^2 \times 9726 = 105$
1	0	$105^2 = 11\,025$
0	1	$11\,025^2 \times 9726 = 5761$

因此,如前所述,密文是 5761。

到此为止,我们已经讨论了 RSA 的加密和解密运算。对于 RSA 参数生成,构造素数 p 和 q (第一步)的方法将在下一节中讨论。第二步是直接的,可以在时间 $O((\log n)^2)$ 中完成。第三步和第四步利用算法 5.3,其复杂度为 $O((\log n)^2)$ 中。

5.4　素性检测

在建立 RSA 密码体制的过程中，生成大的“随机素数”是必要的。实际应用中，一般做法是先生成大的随机整数，然后利用随机多项式时间 Monte Carlo 算法（例如 Solovay-Strassen 算法或者 Miller-Rabin 算法）来检测它们的素性，本节中将介绍这两个算法。这些算法是很快的（就是说，一个整数 n 可以在 $\log_2 n$ 的多项式时间内检测其素性，$\log_2 n$ 即是 n 的二进制表示的位数），但有时算法有可能将一个合数 n 断言为一个素数。然而，通过充分多次运行算法，错误概率可以降低到任何所期望的值以下（我们将在后面详细讨论这一点）。

另一个相关的问题是需要检测多少个随机整数（特定长度）才能找到一个素数。假定我们定义 $\pi(N)$ 为小于等于 N 的素数的个数。数论中一个著名的结论，称为素数个数定理，声称 $\pi(N)$ 约等于 $N/\ln N$。因此，如果在 $1 \sim N$ 之间随机选取一个整数，其为素数的概率大约是 $1/\ln N$。对于 1024 比特的模数 $n = pq$，p 和 q 将选取为 512 比特的素数。一个随机 512 比特整数为素数的概率大约为 $1/\ln 2^{512} \approx 1/355$。就是说，一般地，给定 355 个随机 512 比特整数 p，其中一个会是素数（当然，如果我们把范围限定为奇数，概率就加倍，大约为 2/355）。所以我们能有效地生成“可能为素数”的大整数，因此 RSA 密码体制的参数生成是实际可行的。下面我们将详细描述如何来完成。

一个判定问题（decision problem）是指只能回答“是（yes）”或者“否（no）”的问题。一个随机算法是指任一使用了随机数的算法（反过来，一个算法没有使用随机数，称为确定的算法。下面的定义属于对判定问题的随机算法。

定义 5.1　对于一个判定问题的一个偏是（yes-biased）Monte Carlo 算法是具有下列性质的一个概率算法：一个“是”回答总是正确的，但一个“否”回答也许是不正确的。类似地，可定义一个偏否的（no-biased）Monte Carlo 算法。我们说一个偏是 Monte Carlo 算法具有错误概率 ε，如果算法对任何回答应该为“是”的实例（instance）至多以 ε 的概率给出一个不正确的回答“否”（这个概率是对于给定的输入，通过算法产生所有可能的随机选择进行计算而得出的）。

注　一个 Las Vegas 算法也许不给一个回答，但任何回答总是正确的。反过来，一个 Monte Carlo 算法总是给出一个回答，但回答也许是不正确的。

问题 5.1 提供的判定问题称为合数问题。

问题 5.1　合数（Composites）

条件：　一个正整数 $n \geqslant 2$。

问题：　n 是一个合数吗？

注意,对于一个判定问题的算法只需回答“是”或者“否”。特别地,对于合数问题而言,在 n 为合数的情形下我们并不需要给出其因子分解。

我们首先来描述 Solovay-Strassen 算法,这个算法是对于合数问题的一个偏是 Monte Carlo 算法,该算法具有的错误概率为 1/2。因此,如果算法回答“是”,那么 n 是合数;反之,如果 n 是合数,那么算法至少以 1/2 的概率回答“是”。

尽管 Miller-Rabin 算法(稍后将讨论)比 Solovay-Strassen 算法快,但我们首先考察 Solovay-Strassen 算法,因为它从概念上容易理解,并且引入一些有用的数论知识,这些知识将在后面章节中用到。在描述算法之前,我们先深入探讨一些数论的背景知识。

定义 5.2 假定 p 为一个奇素数,a 为一个整数。a 定义为模 p 的二次剩余(quadratic residue),如果 $a \not\equiv 0 \pmod p$,且同余方程 $y^2 \equiv a \pmod p$ 有一个解 $y \in \mathbb{Z}_p$。a 定义为模 p 的二次非剩余(quadratic non-residue),如果 $a \not\equiv 0 \pmod p$,且 a 不是模 p 的二次剩余。

例 5.6 在 $\mathbb{Z}_{11}$ 中,我们有 $1^2 = 1, 2^2 = 4, 3^2 = 9, 4^2 = 5, 5^2 = 3, 6^2 = 3, 7^2 = 5, 8^2 = 9, 9^2 = 4$ 和 $(10)^2 = 1$。因此模 11 的二次剩余为 1,3,4,5 和 9,模 11 的二次非剩余为 2,6,7,8 和 10。

假定 p 是一个奇素数,a 为一个模 p 的二次剩余。那么存在 $y \in \mathbb{Z}_p^*$,使得 $y^2 \equiv a \pmod p$。显然,$(-y)^2 \equiv a \pmod p$,又因为 p 为奇数,$y \not\equiv -y \pmod p$。现在考虑同余方程 $x^2 - a \equiv 0 \pmod p$。这个同余方程可以分解因子为 $(x-y)(x+y) \equiv 0 \pmod p$,这等价于 $p \mid (x-y)(x+y)$。现在,因为 p 为素数,因此必有 $p \mid (x-y)$ 或者 $p \mid (x+y)$。也就是说,$x \equiv \pm y \pmod p$,由此可知同余方程 $x^2 - a \equiv 0 \pmod p$ 恰好有两个解(模 p)。进一步,这两个解(模 p)互为相反数。

现在我们研究判定一个整数 a 是否为模 p 二次剩余的问题。二次剩余的判定问题如问题 5.2 所定义。注意,这个问题仅是寻求“是”或者“否”的答案:在 a 为模 p 二次剩余的情形时,我们并不需要计算平方根。

问题 5.2 二次剩余(Quadratic Residues)

条件: 一个奇素数 p,和一个整数 a。

问题: a 是一个模 p 二次剩余吗?

我们证明一个结论,称为 Euler 准则,可以为二次剩余问题提供一个多项式时间的判定算法。

定理 5.9 (Euler 准则) 设 p 为一个奇素数,a 为一个正整数。那么 a 是一个模 p 二次剩余当且仅当

$$a^{(p-1)/2} \equiv 1 (\text{mod } p)$$

证明　首先，假定 $a \equiv y^2 (\text{mod } p)$。从推论 5.6 可知，如果 p 是素数，那么 $a^{p-1} \equiv 1 (\text{mod } p)$ 对于任一 $a \not\equiv 0 (\text{mod } p)$成立。于是我们有

$$\begin{aligned} a^{(p-1)/2} &\equiv (y^2)^{(p-1)/2} (\text{mod } p) \\ &\equiv y^{p-1} (\text{mod } p) \\ &\equiv 1 (\text{mod } p) \end{aligned}$$

反过来，假定 $a^{p-1} \equiv 1 (\text{mod } p)$。设 b 为一个模 p 的本原元素。那么 $a \equiv b^i (\text{mod } p)$对于某个正整数 i，我们有

$$\begin{aligned} a^{(p-1)/2} &\equiv (b^i)^{(p-1)/2} (\text{mod } p) \\ &\equiv b^{i(p-1)/2} (\text{mod } p) \end{aligned}$$

由于 b 的阶为 $p-1$，因此必有 $p-1$ 整除 $i(p-1)/2$。因此，i 是偶数，于是 a 的平方根为 $\pm b^{i/2} \text{mod } p$。

利用模 p 指数的平方-乘算法，定理 5.9 给出了二次剩余问题的多项式时间算法。算法的复杂性为 $O((\log p)^3)$。

我们现在给出数论中更多的定义。

定义 5.3　假定 p 是一个奇素数。对于任一整数 a，定义 Legendre 符号$\left(\frac{a}{p}\right)$如下：

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & a \equiv 0 (\text{mod } p) \\ 1 & a \text{ 是一个模 } p \text{ 二次剩余} \\ -1 & a \text{ 是一个模 } p \text{ 二次非剩余} \end{cases}$$

我们已经知道 $a^{(p-1)/2} \equiv 1 (\text{mod } p)$当且仅当 a 是一个模 p 二次剩余。如果 a 是 p 的倍数，那么显然有 $a^{(p-1)/2} \equiv 0 (\text{mod } p)$。最后，如果 a 是一个模 p 二次非剩余，那么 $a^{(p-1)/2} \equiv -1 (\text{mod } p)$，因为

$$(a^{(p-1)/2})^2 \equiv a^{p-1} \equiv 1 (\text{mod } p)$$

且 $a^{(p-1)/2} \not\equiv 1 (\text{mod } p)$。因此我们有如下结论，提供了求 Legendre 符号值的有效算法。

定理 5.10　假定 p 是一个奇素数，那么

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} (\text{mod } p)$$

下面我们定义 Legendre 符号的一般形式。

定义 5.4 假定 n 是一个奇的正整数,且 n 的素数幂因子分解为:

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

设 a 为一个整数,那么 Jacobi 符号$\left(\frac{a}{n}\right)$定义为:

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k}\left(\frac{a}{p_i}\right)^{e_i}$$

例 5.7 考虑 Jacobi 符号$\left(\frac{6278}{9975}\right)$。9975 的素数幂因子分解为 $9975 = 3 \times 5^2 \times 7 \times 19$。因此我们有:

$$\begin{aligned}\left(\frac{6278}{9975}\right) &= \left(\frac{6278}{3}\right)\left(\frac{6278}{5}\right)^2\left(\frac{6278}{7}\right)\left(\frac{6278}{19}\right)\\ &= \left(\frac{2}{3}\right)\left(\frac{3}{5}\right)^2\left(\frac{6}{7}\right)\left(\frac{8}{19}\right)\\ &= (-1)(-1)^2(-1)(-1)\\ &= -1\end{aligned}$$

假定 $n > 1$ 为奇数。如果 n 是素数,那么对于任一整数 a 有$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod n$。另一方面,如果 n 是合数,$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod n$可能成立,也可能不成立。如果同余式成立,则称 n 为对于基底 a 的 Euler 伪素数。例如,91 是一个对于基底 10 的 Euler 伪素数,因为,

$$\left(\frac{10}{91}\right) = -1 \equiv 10^{45} \pmod{91}$$

可以证明,对于任一奇合数 n,n 是对于基底 a 的 Euler 伪素数至多对一半的整数 $a \in \mathbb{Z}_n^*$ 成立(参见练习)。容易看出$\left(\frac{a}{n}\right) = 0$,当且仅当 $\gcd(a, n) > 1$(因此,如果 $1 \leqslant a \leqslant n-1$ 且 $\left(\frac{a}{n}\right) = 0$,必定是 n 为合数的情形)。这两个事实说明,Solovay-Strassen 算法(算法 5.6)是一个偏是 Monte Carlo 算法,并具有 1/2 的错误概率。

算法 5.6 Solovay-Strassen(n)

选取随机整数 a,使得 $1 \leqslant a \leqslant n-1$

$x \leftarrow \left(\frac{a}{n}\right)$

if $x = 0$

then return ("n is composite")

$y \leftarrow a^{(n-1)/2} \pmod{n}$
if $x \equiv y \pmod{n}$
　then return ("n is prime")
　else return ("n is composite")

到目前为止,还不清楚算法 5.6 是否为一个多项式时间算法。我们已经知道如何在时间 $O((\log n)^3)$内求值 $a^{(n-1)/2} \pmod{n}$,但如何有效地计算 Jacobi 符号呢? 从表面上看,好像要先对 n 进行因子分解,由于 Jacobi 符号$\left(\frac{a}{n}\right)$的定义就是n 的因子分解组成的项。但是,如果我们已经知道 n 的因子分解,我们便知道了 n 是否为素数;所以这种方式就陷入了错误的循环中。

幸运的是,我们可以利用一些数论的结果而无需分解 n 就可以求出 Jacobi 符号的值,其中最重要的一条就是二次互反律的一般形式(如下性质 4)。我们仅列出这些性质,不给证明:

1. 如果 n 是一个正奇数,且 $m_1 \equiv m_2 \pmod{n}$,那么:

$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

2. 如果 n 是一个正奇数,那么:

$$\left(\frac{2}{n}\right) = \begin{cases} 1 & n \equiv \pm 1 \pmod 8 \\ -1 & n \equiv \pm 3 \pmod 8 \end{cases}$$

3. 如果 n 是一个正奇数,那么:

$$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

特别地,如果 $m = 2^k t$ 且 t 为一个奇数,那么:

$$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$$

4. 如果 m 和 n 是正奇数,那么:

$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & m \equiv n \equiv 3 \pmod 4 \\ \left(\frac{n}{m}\right) & \text{其他情况} \end{cases}$$

例 5.8　作为上面这些性质应用的描述,我们计算 Jacobi 符号$\left(\frac{7411}{9283}\right)$的值如下:

$$\left(\frac{7411}{9283}\right) = -\left(\frac{9283}{7411}\right) \quad \text{由性质 4}$$

$$= -\left(\frac{1872}{7411}\right) \quad \text{由性质 1}$$

$$= -\left(\frac{2}{7411}\right)^4\left(\frac{117}{7411}\right) \quad \text{由性质 3}$$

$$= -\left(\frac{117}{7411}\right) \quad \text{由性质 2}$$

$$= -\left(\frac{7411}{117}\right) \quad \text{由性质 4}$$

$$= -\left(\frac{40}{117}\right) \quad \text{由性质 1}$$

$$= -\left(\frac{2}{117}\right)^3\left(\frac{5}{117}\right) \quad \text{由性质 3}$$

$$= \left(\frac{5}{117}\right) \quad \text{由性质 2}$$

$$= \left(\frac{117}{5}\right) \quad \text{由性质 4}$$

$$= \left(\frac{2}{5}\right) \quad \text{由性质 1}$$

$$= -1 \quad \text{由性质 3}$$

注意,我们在计算过程中连续地使用了性质 4、1、3 和 2。

一般地,如上面例子那样用相同的方式使用这四个性质,在多项式时间内算出 Jacobi 符号是可行的。所需的算术运算仅是模约化运算和 2 的幂次分解运算。注意,如果整数采用二进制表示,那么 2 的幂次分解就等于数一下二进制表示中末尾零的个数。因此,算法的复杂性取决于执行的模约化的次数。容易看出至多执行 $O(\log n)$次模约化,每次所需时间为 $O((\log n)^2)$。这说明算法的复杂性为 $O((\log n)^3)$,这是关于 $\log n$ 的多项式时间(事实上,通过更为精细的分析可知其复杂性为 $O((\log n)^2)$)。

假定我们已经生成了随机数 n,且用 Solovay-Strassen 算法已检测其素性。如果我们运行了算法 m 次,我们能在多大程度上相信 n 是一个素数呢? 有结论说 n 为素数的概率为 $1-2^{-m}$。这个结果在很多教科书和技术文章中出现,但它并不能从给定数据中导出。

在使用概率时我们必须特别小心。假定我们定义了如下的随机变量:a 表示事件

"一个特定长度的随机奇整数 n 是一个合数"

b 表示事件

"算法连续回答了 m 次'n 是一个素数'"

显然有条件概率 $\mathbf{Pr}[b \mid a] \leqslant 2^{-m}$。然而,我们真正感兴趣的概率是 $\mathbf{Pr}[a \mid b]$,通常与 $\mathbf{Pr}[b \mid a]$ 并不相同。

我们可以利用 Bayes 定理(定理 2.1)来计算 $\mathbf{Pr}[b \mid a]$。为了做到这点,我们需要知道 $\mathbf{Pr}[a]$。假定 $N \leqslant n \leqslant 2N$。应用素数定理,在 N 和 $2N$ 之间的(奇)素数的个数大约是:

$$\frac{2N}{\ln 2N} - \frac{N}{\ln N} \approx \frac{N}{\ln N}$$

$$\approx \frac{n}{\ln n}$$

由于在 N 和 $2N$ 之间的奇数的个数为 $N/2 \approx n/2$,我们使用估计

$$\mathbf{Pr}[a] \approx 1 - \frac{2}{\ln n}$$

那么我们可以进行如下计算:

$$\begin{aligned}
\mathbf{Pr}[b \mid a] &= \frac{\mathbf{Pr}[b \mid a]\mathbf{Pr}[a]}{\mathbf{Pr}[b]} \\
&= \frac{\mathbf{Pr}[b \mid a]\mathbf{Pr}[a]}{\mathbf{Pr}[b \mid a]\mathbf{Pr}[a] + \mathbf{Pr}[b \mid \overline{a}]\mathbf{Pr}[\overline{a}]} \\
&\approx \frac{\mathbf{Pr}[b \mid a]\left(1 - \frac{2}{\ln n}\right)}{\mathbf{Pr}[b \mid a]\left(1 - \frac{2}{\ln n}\right) + \frac{2}{\ln n}} \\
&= \frac{\mathbf{Pr}[b \mid a](\ln n - 2)}{\mathbf{Pr}[b \mid a](\ln n - 2) + 2} \\
&\leqslant \frac{2^{-m}(\ln n - 2)}{2^{-m}(\ln n - 2) + 2} \\
&= \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}}
\end{aligned}$$

注意,在计算过程中,$\overline{a}$ 表示事件:

"一个随机奇数 n 是素数"

把 $(\ln n - 2)/(\ln n - 2 + 2^{m+1})$ 与 2^{-m} 两个量作为 m 的函数做一个对比是很有趣的。假定 $n \approx 2^{512} \approx e^{355}$,这就是用来构建 RSA 模数的 p 和 q 的大小。那么第一个函数大约为 $353/(353 + 2^{m+1})$。我们把两个函数对某些 m 的取值列在表 5.1 中。

表 5.1　Solovay-Strassen 算法的错误概率

m	2^{-m}	错误概率的界
1	.500	.980
2	.250	.978
5	$.312 \times 10^{-1}$	.847

续表

m	2^{-m}	错误概率的界
10	$.977 \times 10^{-3}$	.147
20	$.954 \times 10^{-6}$	$.168 \times 10^{-3}$
30	$.931 \times 10^{-9}$	$.164 \times 10^{-6}$
50	$.888 \times 10^{-15}$	$.157 \times 10^{-12}$
100	$.789 \times 10^{-30}$	$.139 \times 10^{-27}$

尽管 $353/(353+2^{m+1})$ 以指数速度快速接近零,但是还不如 2^{-m} 快。然而实际上,可以选取 m 为 50 或 100,使得错误概率是非常小的量。

我们导出另一个对于合数问题的 Monte Carlo 算法,称为 Miller-Rabin 算法(也称为"强伪素数检测(strong pseudo-prime test)")。这个算法在算法 5.7 中描述。

算法 5.7 Miller-Rabin (n)

把 $n-1$ 写成 $n-1=2^k m$,其中 m 是一个奇数

选取随机整数 a,使得 $1 \leqslant a \leqslant n-1$

$b \leftarrow a^m \pmod n$

if $b \equiv 1 \pmod n$

  then return ("n is prime")

for $i \leftarrow 0$ **to** $k-1$

do { **if** $b \equiv -1 \pmod n$

  then return ("n is prime")

  else $b \leftarrow b^2 \pmod n$

return ("n is composite")

算法 5.7 显然是一个多项式时间算法,初步分析可知其时间复杂度为 $O((\log n)^3)$,跟 Solovay-Strassen 算法相同。但在实际运行中,Miller-Rabin 算法要比 Solovay-Strassen 算法好。

我们现在证明 Miller-Rabin 算法在"n 为素数"的情形下,不会回答"n 为合数",也就是说,这个算法是一个偏是的 Monte Carlo 算法。

定理 5.11 Miller-Rabin 算法对于合数问题是一个偏是的 Monte Carlo 算法。

证明 我们用反证法。先假定算法 5.7 对于某个素数 n 回答了"n 为合数",然后推出矛盾。由于算法回答"n 为合数",必有 $a^m \not\equiv 1 \pmod n$。现在考虑在算法中检测的 b 的序列。由于 b 在 **for** 循环的每一步中都做平方运算,我们测试的值为 $a^m, a^{2m}, \cdots, a^{2^{k-1}m}$。由于算法回答"$n$ 为合数",我们可知对于 $0 \leqslant i \leqslant k-1$,有:

$$a^{2^t m} \not\equiv -1 (\mathrm{mod}\ n)$$

现在，利用 n 为素数的假定，由于 $n-1=2^k m$，由 Fermat 定理(参见推论 5.6)知：

$$a^{2^k m} \equiv 1 (\mathrm{mod}\ n)$$

那么 $a^{2^{k-1} m}$ 是模 n 的 1 的平方根。由于 n 为素数，仅有两个模 n 的 1 的平方根，即 $\pm 1 \bmod n$。我们有：

$$a^{2^{k-1} m} \not\equiv -1 (\mathrm{mod}\ n)$$

由此得出

$$a^{2^{k-1} m} \equiv 1 (\mathrm{mod}\ n)$$

那么 $a^{2^{k-2} m}$ 一定是模 n 的 1 的平方根。基于相同的理由，

$$a^{2^{k-2} m} \equiv 1 (\mathrm{mod}\ n)$$

重复上述过程，我们最后得到：

$$a^m \equiv 1 (\mathrm{mod}\ n)$$

但是在这种情形下，算法会回答“n 为素数”，推出矛盾。

仍需考虑 Miller-Rabin 算法的错误概率。可以证明 Miller-Rabin 算法的错误概率至多为 1/4，在这里就不再给出证明过程。

5.5　模 n 的平方根

在这一节中，我们简要地讨论与模 n 的平方根存在性相关的几个结论。贯穿本节的始终，我们假定 n 为一个奇数，且 $\gcd(n, a)=1$。第一个问题是同余方程 $y^2 \equiv a (\mathrm{mod}\ n)$ 具有 $y \in \mathbb{Z}_n$ 的根的个数。从 5.4 节中我们已经知道，当 n 是素数的时候，同余方程或者有零个或者有两个解，依赖于 $\left(\frac{a}{n}\right) = -1$ 或者 $\left(\frac{a}{n}\right) = 1$。

我们的下一个定理把这种特征扩展到了(奇)素数幂的情形。证明的概要过程在练习中给出。

定理 5.12　假定 p 为一个奇素数，e 为一个正整数，且 $\gcd(a, p)=1$，那么同余方程 $y^2 \equiv a (\mathrm{mod}\ p^e)$ 当 $\left(\frac{a}{p}\right) = -1$ 时没有解，当 $\left(\frac{a}{p}\right) = 1$ 时有两个解(模 p^e)。

注意，定理 5.12 告诉我们，模 p^e 的平方根的存在性可以由计算 Legendre 符号来决定。

还可以容易地把定理 5.12 推广到任意奇整数 n 的情形中。如下结论是中国剩余定理的一个基本应用。

定理 5.13 假定 $n>1$ 为一个奇数有如下分解:

$$n = \prod_{i=1}^{l} p_i^{e_i}$$

其中 p_i 为不同的素数,且 e_i 为正整数。进一步假定 $\gcd(a,n)=1$。那么同余方程 $y^2 \equiv a(\bmod n)$ 当 $\left(\frac{a}{p_i}\right)=1$ 对于所有的 $i \in \{1,\cdots,l\}$ 成立时有 2^l 个模 n 的解,其他情形下没有解。

证明 容易知道 $y^2 \equiv a(\bmod n)$ 当且仅当 $y^2 \equiv a(\bmod p_i^{e_i})$ 对于所有 $i \in \{1,\cdots,l\}$ 成立。如果对于某个 i 有 $\left(\frac{a}{p_i}\right)=-1$,那么同余方程 $y^2 \equiv a(\bmod p_i^{e_i})$ 没有解,因此 $y^2 \equiv a(\bmod n)$ 没有解。

现在假定对于所有的 $i \in \{1,\cdots,l\}$ 有 $\left(\frac{a}{p_i}\right)=1$。由定理 5.12 可知每一个同余方程 $y^2 \equiv a(\bmod p_i^{e_i})$ 有两个模 $p_i^{e_i}$ 的解,设为 $y \equiv b_{i,1}$ 或 $b_{i,2}(\bmod p_i^{e_i})$。对于 $1 \leqslant i \leqslant l$,令 $b_i \in \{b_{i,1}, b_{i,2}\}$。那么方程 $y \equiv b_i(\bmod p_i^{e_i})$ $(1 \leqslant i \leqslant l)$ 有惟一的模 n 解,这可以利用中国剩余定理求出。存在 2^l 种方式选择 l 组 $(b_1,\cdots,b_l)$,因此对于同余方程 $y^2 \equiv a(\bmod n)$ 有 2^l 个模 n 的解。

假定 $x^2 \equiv y^2 \equiv a \pmod n$,其中 $\gcd(a,n)=1$。设 $z = xy^{-1} \bmod n$。于是可得出 $z^2 \equiv 1(\bmod n)$。反过来,如果 $z^2 \equiv 1(\bmod n)$,那么 $(xz)^2 \equiv x^2(\bmod n)$ 对于任意 x 成立。因此,把 $a \in \mathbb{Z}_n^*$ 的某个给定平方根与 1 的 2^l 个平方根做出 2^l 个乘积,就得到 a 的 2^l 个平方根。我们将在本章后面利用这个观察的结果。

5.6 分解因子算法

攻击 RSA 密码体制的最明显方式就是试图分解公开模数。关于分解因子的算法有大量的文章讨论,要做一个完整的介绍将会花去本书更多的篇幅。我们这里仅试着做一个概览,包括对当今最好的算法的非正式讨论,以及在现实中的应用。对于大整数最有效的三种算法是二次筛法(quadratic sieve)、椭圆曲线分解算法(elliptic curve factoring)和数域筛法(number field sieve)。其他作为先驱的著名算法包括 Pollard 的 ρ 方法和 $p-1$ 算法、William 的 $p+1$ 算法、连分式算法(continued fraction),当然还有试除法(trial division)。

贯穿本节始终,我们假定要分解的整数 n 为奇数。如果 n 是合数,容易看出 n 有一个素因子 $p \leqslant \lfloor \sqrt{n} \rfloor$。因此,作为最简单方法的试除法,就是用直到 $\lfloor \sqrt{n} \rfloor$ 的每个奇数去除 n,这足以判断 n 是素数还是合数。如果 $n < 10^{12}$,它还算是个不错的算法,但对于非常大的 n,我们还需要更多复杂的技巧才行。

当我们说要分解 n,我们或者是要完全分解为素因子之积,或者只是找到任一非平凡因子(non-trivial factor)即可。在我们研究的大多数算法中,我们仅是寻找一个任意的非平凡因子。一般地,我们得到形如 $n = n_1 n_2$ 的分解,其中 $1 < n_1 < n$,且 $1 < n_1 < n$。如果我们需要把 n 分解为素数积,可以先用随机素性检测算法进行测试,然后再对不是素数的因子进一步分解。

5.6.1　Pollard $p-1$ 算法

作为有时可用于大整数分解的简单算法的一个例子，我们介绍 Pollard 的 $p-1$ 算法，它于 1974 年被提出。这个算法如算法 5.8 所述，有两个输入：要分解的(奇)整数 n，和一个预先指定的“界”B。

算法 5.8　Pollard $p-1$ Factoring Algorithm(n, B)

$a \leftarrow 2$

for $j \leftarrow 2$ **to** B

　do $a \leftarrow a^j \bmod n$

$d \leftarrow \gcd(a-1, n)$

if $1 < d < n$

　then return (d)

　else return(“failure”)

下面对上述算法的合理性稍做解释。假定 p 是 n 的一个素因子，又假定对每一个素数幂 $q \mid (p-1)$，有 $q \leqslant B$。那么在这种情形下必有：

$$(p-1) \mid B!$$

在 **for** 循环结束时，我们有：

$$a \equiv 2^{B!} \pmod{n}$$

由于 $p \mid n$，一定有：

$$a \equiv 2^{B!} \pmod{p}$$

现在，由 Fermat 定理可知：

$$2^{p-1} \equiv 1 \pmod{p}$$

由于 $(p-1) \mid B$，于是有：

$$a \equiv 1 \pmod{p}$$

因此有 $p \mid (a-1)$。由于我们已经有 $p \mid n$，我们可以看到 $p \mid d$，其中 $d = \gcd(a-1, n)$。整数 d 就是 n 的一个非平凡因子(除非 $a = 1$)。一旦找到 n 的一个非平凡因子 d，我们可以对 d 和 n/d 继续分解(如果 d 和 n/d 还是合数)。

下面用一个例子来说明上述过程。

例 5.9　假定 $n = 15\,770\,708\,441$。如果我们选取 $B = 180$ 应用算法 5.8，我们可以发现 $a = 11\,620\,221\,425$，d 计算的结果为 135 979。事实上，n 的完全素因子分解为：

$$15\,770\,708\,441 = 135\,979 \times 115\,979$$

在这个例中,分解成功是因为 135 978 仅有"小"的素因子:

$$135\,978 = 2 \times 3 \times 131 \times 173$$

因此,通过选取 $B \geqslant 173$,会有 $135\,978 \mid B!$,正是我们所需要的。

在 $p-1$ 算法中,有 $B-1$ 个模指数,利用"平方-乘"算法计算每一个模指数需要至多 $2\log_2 B$ 个模乘法。gcd 的计算可用 Euclidean 算法在时间 $O((\log n)^3)$ 内完成。因此,该算法的时间复杂度是 $O(B \log B(\log n)^2 + (\log n)^3)$。如果 B 是 $O((\log n)^i)$,i 是某一固定整数,那么该算法的确是多项式时间的算法($\log n$ 的一个函数)。然而,对 B 的这样的一个选择,算法成功的概率将是很小的。另一方面,如果猛地增加 B 的大小,比如说$\sqrt{n}$,那么该算法成功的概率将是很高的,但是它并不比试除法快。

可见,这个算法的缺陷是它要求 n 有一个素因子 p 使得 $p-1$ 只有小的素因子。这就要求我们在构造 RSA 模 $n = pq$ 时选择的 p 和 q 应使 $p-1$ 和 $q-1$ 含有大的素因子,使 $p-1$ 分解算法失败。一般的方法是选择两个大素数 p_1 和 q_1,使得 $p = 2p_1 + 1$ 和 $q = 2q_1 + 1$ 也是素数(形如 $p = 2p_1 + 1$ 的素数通常称为安全素数,其中 p_1 为素数),此时,RSA 模 $n = pq$ 将能抵抗 $p-1$ 方法的分解。

20 世纪 80 年代中期,由 Lenstra 提出的更有效的椭圆曲线算法实际上是 $p-1$ 算法的一般化。椭圆曲线算法的成功率依赖于接近于 p 的一个整数只有小的素因子。$p-1$ 算法依赖于群$\mathbb{Z}_p$ 中的一个关系,椭圆曲线算法引入了定义在模 p 椭圆曲线上的群。

5.6.2 Pollard ρ 方法

设 p 为 n 的最小的素因子。假定存在两个整数 $x, x' \in \mathbb{Z}_n$,使得 $x \neq x'$ 且 $x \equiv x' \pmod{p}$。那么 $p \leqslant \gcd(x - x', n) < n$,所以我们通过计算最大公因子得到 n 的一个非平凡素因子(注意到为了使这个算法工作,并不需要事先知道 p 的值)。

假定我们通过先选择一个随机子集 $X \subseteq \mathbb{Z}_n$ 来分解 n,然后对所有不同的 $x, x' \in X$ 计算 $\gcd(x - x', n)$。这个方法能够成功当且仅当映射 $x \mapsto x \bmod p$ 对于 $x \in X$ 得到至少一个碰撞。利用 4.2.2 小节中描述的生日悖论可以分析这种情形:如果 $|X| \approx 1.17\sqrt{p}$,那么至少存在一个碰撞的概率为 50%,因此能找到一个 n 的非平凡因子。然而,为了寻找形如 $x \bmod p = x' \bmod p$ 的碰撞,我们需要计算 $\gcd(x - x', n)$(因为 p 值未知,我们不能像 4.2.2 节中那样对于 $x \in X$ 显式地计算 $x \bmod p$,然后对结果排序)。这意味着我们在找到 n 的一个因子前,需要计算超过$\binom{|X|}{2} > p/2$ 次 gcd。

Pollard 的 ρ 方法集成了这个技巧的一个变形,仅需要较少的 gcd 计算,占用较少的存储空间。假定 f 为一个具有整数系数的多项式,例如 $f(x) = x^2 + a$,其中 a 为一个小常数(通常取 $a = 1$)。假定映射 f 类似于一个随机映射。(当然不是"随机的",我们提出的是一个启发式的分析,而不是一个严格的证明。)设 $x_1 \in \mathbb{Z}_n$,考虑序列 $x_1, x_2, \cdots$,其中:

$$x_j = f(x_{j-1}) \bmod n$$

对于所有的 $j \geqslant 2$。令 m 为一个整数，且定义 $X = \{x_1, \cdots, x_m\}$。为简化讨论，假定 X 包含 m 个不同的模 n 剩余。希望 X 为 $\mathbb{Z}_n$ 的 m 个元素的随机子集。

我们寻找两个不同的值 $x_i, x_j \in X$ 使得 $\gcd(x_j - x_i, n) > 1$。每次我们计算序列中的一个新项 x_j，要对所有的 $i < j$ 计算 $\gcd(x_j - x_i, n)$。然而，我们可以大大地减少 gcd 计算的次数。下面我们描述如何做到这一点。

假定 $x_i \equiv x_j \pmod{p}$。使用 f 为整数系数多项式这一事实，我们有 $f(x_i) \equiv f(x_j) \pmod{p}$。回想 $x_{i+1} = f(x_i) \bmod n$ 且 $x_{j+1} = f(x_j) \bmod n$。那么：

$$x_{i+1} \bmod p = (f(x_i) \bmod n) \bmod p = f(x_i) \bmod p$$

因为 $p \mid n$。类似地，

$$x_{j+1} \bmod p = f(x_j) \bmod p$$

因此 $x_{i+1} \equiv x_{j+1} \pmod{p}$。重复上述过程，我们得到如下重要结论：

如果 $x_i \equiv x_j \pmod{p}$，那么对于所有的整数 $\delta \geqslant 0$，$x_{i+\delta} \equiv x_{j+\delta} \pmod{p}$。

记 $l = j - i$，可知 $x_{i'} \equiv x_{j'} \pmod{p}$ 如果 $j' > i' \geqslant i$ 且 $j' - i' \equiv 0 \pmod{l}$。

假定我们利用顶点集 $\mathbb{Z}_p$ 构造一个图 G，其中对于 $i \geqslant 1$，我们从点 $x_i \bmod p$ 向点 $x_{i+1} \bmod p$ 做一条有向边。存在第一个点对 x_i, x_j 且 $i < j$，使得 $x_i \equiv x_j \pmod{p}$。通过上面的观察，容易看到图 G 包括一条“尾巴”：

$$x_1 \bmod p \rightarrow x_2 \bmod p \rightarrow \cdots \rightarrow x_i \bmod p$$

和一个长为 l 的无穷重复的环，具有顶点：

$$x_i \bmod p \rightarrow x_{i+1} \bmod p \rightarrow \cdots \rightarrow x_j \bmod p = x_i \bmod p$$

因此 G 看起来像希腊字母 ρ，这就是 ρ 算法名字的由来。

我们看一个说明上面过程的例子。

例 5.10　假定 $n = 7171 = 71 \times 101$，$f(x) = x^2 + 1$ 和 $x_1 = 1$。那么序列 x_i 前面的部分如下：

1	2	5	26	667	6557	4105
6347	4903	2218	219	4936	4210	4560
4872	375	4377	4389	2016	5471	88

上面的值，模 71 后的结果如下：

1	2	5	26	38	25	58
28	4	17	6	27	21	16
44	20	46	58	28	4	17

上面表中第一个碰撞为：

$$x_7 \bmod 71 = x_{18} \bmod 71 = 58$$

因此图 G 中包含一条长为 7 的尾巴和一个长为 11 的圈。

我们已经提到我们的目标是通过求最大公因子找出两个项 $x_i \equiv x_j \pmod{p}$, $i < j$。不一定要找出这种类型的第一个碰撞。为了简化和改进算法,我们通过取 $j = 2i$ 来寻找碰撞。得到的算法如算法 5.9 所述。

算法 5.9 Pollard ρ Factoring Algorithm(n, x_1)

external f
$x \leftarrow x_1$
$x' \leftarrow f(x) \bmod n$
$p \leftarrow \gcd(x - x', n)$
while $p = 1$
  do { **comment**: 在第 i 次反复中, $x = x_i$ 且 $x' = x_{2i}$
    $x \leftarrow f(x) \bmod n$
    $x' \leftarrow f(x') \bmod n$
    $x' \leftarrow f(x') \bmod n$
    $p \leftarrow \gcd(x - x', n)$
if $p = n$
  then return ("failure")
  else return (p)

这个算法并不难于分析。如果 $x_i \equiv x_j \pmod{p}$,那么就有如下情形:对于所有满足 $i' \equiv 0 \pmod{l}$, $i' \geqslant i$ 的 i' 有 $x_{i'} = x_{2i'} \pmod{p}$。在 l 个连续的整数 $i, \cdots, j-1$ 中,一定存在一个数可以被 l 整除。因此满足上面两个条件的最小的 i' 值不超过 $j-1$。因此,需要找到一个因子 p 的循环次数最多为 j,而 j 的值最大为 $\sqrt{p}$。

在例 5.10 中,模 71 的碰撞最先出现在 $i = 7, j = 18$。大于等于 7 且被 11 整除的最小整数为 $i' = 11$。因此算法 5.9 在计算出 $\gcd(x_{11} - x_{22}, n) = 71$ 时,就找到了 n 的因子 71。

一般地,由于 $p < \sqrt{n}$,算法的期望复杂度为 $O(n^{1/4})$(忽略其对数因子)。然而,我们再强调一下,这是一个启发式的分析,并非一个数学证明。另一方面,算法实际上的执行类似于这个估计。

算法 5.9 也可能没有找到 n 的一个非平凡因子而导致失败。这种情形发生在当且仅当第一对满足 $x \equiv x' \pmod{p}$ 的 x 和 x' 也满足 $x \equiv x' \pmod{n}$(这等价于 $x = x'$,因为 x 和 x' 是模 n 约化的)。我们可以启发式地估计这种情形发生的概率大致为 p/n,当 n 很大时这是一个很小的值,因为 $p < \sqrt{n}$。如果算法以这种方式失败,我们可以取一个不同的初值重新运行算法,或者选择一个不同的函数 f。

读者可能希望对一个较大的 n 来运行算法 5.9。当 $n = 15\,770\,708\,441$ 时(与例 5.9 中所取

n 值相同)，$x_1 = 1$ 和 $f(x) = x^2 + 1$，可以验证 $x_{422} = 2\,261\,992\,698$，$x_{211} = 7\,149\,213\,937$ 且

$$\gcd(x_{422} - x_{211}, n) = 135\,979$$

5.6.3　Dixon 的随机平方算法

许多分解因子算法的理论基于下列的简单事实。假定我们可以找到 $x \not\equiv \pm y \pmod{n}$使得 $x^2 \equiv y^2 \pmod{n}$，那么

$$n \mid (x - y)(x + y)$$

但 $x + y$ 或者 $x - y$ 都不能被 n 整除。因此 $\gcd(x + y, n)$是 n 的一个非平凡因子(类似地，$\gcd(x - y, n)$也是 n 的一个非平凡因子)。

作为一个例子，容易验证 $10^2 \equiv 32^2 \pmod{77}$。通过计算 $\gcd(10 + 32, 77) = 7$，我们发现了 77 的一个因子 7。

随机平方算法使用一个因子基(factor base)，$\mathcal{B}$因子基是 b 个最小素数的集合(适当选取 b)。我们首先得到几个整数 z，使得 $z^2 \bmod n$ 的所有素因子都在因子基$\mathcal{B}$中(如何做到这一点将在稍后讨论)。然后，将某些 z 相乘使得每一个在因子基中的素数出现偶数次。这样我们就建立起了一个所期望的类型的同余方程 $x^2 \equiv y^2 \pmod{n}$，该方程可能导出 n 的一个分解。

我们用一个仔细设计的例子来说明 Dixson 算法的基本观点。

例 5.11　假定 $n = 15\,770\,708\,441$(这与例 5.9 中所取 n 值相同)。令 $b = 6$;那么$\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$。考虑如下三个同余方程:

$$8\,340\,934\,156^2 \equiv 3 \times 7 \pmod{n}$$
$$12\,044\,942\,944^2 \equiv 2 \times 7 \times 13 \pmod{n}$$
$$2\,773\,700\,011^2 \equiv 2 \times 3 \times 13 \pmod{n}$$

如果我们取上面同余方程两边的乘积，那么有:

$$(8\,340\,934\,156 \times 12\,044\,942\,944 \times 2\,773\,700\,011)^2 \equiv (2 \times 3 \times 7 \times 13)^2 \pmod{n}$$

在两边表达式的括号中模 n 约化，我们有:

$$9\,503\,435\,785^2 \equiv 546^2 \pmod{n}$$

然后，利用 Euclidean 算法，我们计算

$$\gcd(9\,503\,435\,785 - 546, 15\,770\,708\,441) = 115\,759$$

找到 n 的因子 115 759。

假定$\mathcal{B} = \{p_1, \cdots, p_b\}$为因子基。设 c 为稍大于 b 的整数(比如 $c = b + 4$)，且假定我们已经得到 c 个同余方程:

$$z_j^2 \equiv p_1^{a_{1j}} \times p_2^{a_{2j}} \times \cdots \times p_b^{a_{bj}} \pmod{n}$$

其中 $1 \leqslant j \leqslant c$。对于每一个 j，考虑向量

$$\boldsymbol{a}_j = (\alpha_{1j} \bmod 2, \cdots, \alpha_{bj} \bmod 2) \in (\mathbb{Z}_2)^b$$

如果我们可以找到$\{\boldsymbol{a}_j\}$的子集使得其模2的和为向量$(0,\cdots,0)$,那么对应的z_j的乘积将会使用$\mathcal{B}$中的每个因子偶数次。

我们回到例5.11中来说明,其中存在一个线性相关关系。

例5.11(续) 三个向量$\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$如下:

$$\boldsymbol{a}_1 = (0,1,0,1,0,0)$$
$$\boldsymbol{a}_2 = (1,0,0,1,0,1)$$
$$\boldsymbol{a}_3 = (1,1,0,0,0,1)$$

容易看出,

$$\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3 = (0,0,0,0,0,0) \bmod 2$$

由此我们可以得到前面的同余方程,能成功地分解n。

可以看到,寻找c个向量$\boldsymbol{a}_1, \cdots, \boldsymbol{a}_c$的一个子集使得其和模2为零向量,等价于寻找这些向量(在$\mathbb{Z}_2$上)的一个线性相关。给出$c > b$,这样的一个线性相关一定存在,且可以用Gaussian消去法容易地找到。我们选取$c > b+1$的原因是,不能保证任一给定的同余方程$x^2 \equiv y^2 \pmod{n}$一定能得到n的分解。然而,我们可以如下估计$x \equiv \pm y \pmod{n}$的概率最多为50%。假定$x^2 \equiv y^2 \pmod{n}$,其中$\gcd(a,n)=1$。定理5.13告诉我们a有2^l个模n的平方根,其中l为n的素因子的个数。如果$l \geqslant 2$,那么a至少有四个平方根。因此,如果我们假定x和y"随机"选取,可以推出$x \equiv \pm y \pmod{n}$出现的概率为$2/2^l \leqslant 1/2$。

现在,如果$c > b+1$,我们可以得到几个形如$x^2 \equiv y^2 \pmod{n}$的同余方程(由a_j的不同线性相关关系得出)。可以设想,所得到的同余方程中至少有一个可以得到形如$x^2 \equiv y^2 \pmod{n}$的同余方程且$x \not\equiv \pm y \pmod{n}$,由此可以得到$n$的一个非平凡因子。

我们现在讨论如何得到整数z,使得值$z^2 \bmod n$可以在给定因子基$\mathcal{B}$上完全分解。有几个方法可以做到这一点。一种方法是简单地随机选择z;这也是随机平方算法名字的由来。然而,试用形如$j + \lceil \sqrt{kn} \rceil, j = 0,1,2,\cdots, k = 1,2,\cdots$的整数是特别有用的。这些整数在平方以及模$n$约化后,一般会比较小,因此它们比随机选择得到的结果在$\mathcal{B}$上完全分解的可能性要高一些。另一个有用的技巧是试用形如$z = \lfloor \sqrt{kn} \rfloor$的整数。当平方和模$n$约化后,这些整数比$n$小一点。这意味着$-z^2 \bmod n$是很小的,有可能在$\mathcal{B}$上完全分解。因此,如果我们把$-1$加到$\mathcal{B}$中,就可以在$\mathcal{B}$上分解$z^2 \bmod n$。

我们用一个小例子来描述这些技巧。

例5.12 假定$n = 1829, \mathcal{B} = \{-1,2,3,5,7,11,13\}$。我们计算$\sqrt{n} = 42.77$,$\sqrt{2n} = 60.48$,$\sqrt{3n} = 74.04$且$\sqrt{4n} = 85.53$。假定我们取$z = 42,43,64,65,74,75,85,86$。我们得到$z^2 \bmod n$在$\mathcal{B}$上的分解。在下面的表中,所有同余式都是模$n$的:

$$
\begin{aligned}
z_1^2 &\equiv 42^2 \equiv -65 \equiv (-1)\times 5\times 13\\
z_2^2 &\equiv 43^2 \equiv 20 \equiv 2^2\times 5\\
z_3^2 &\equiv 61^2 \equiv 63 \equiv 3^2\times 7\\
z_4^2 &\equiv 74^2 \equiv -11 \equiv (-1)\times 11\\
z_5^2 &\equiv 85^2 \equiv -91 \equiv (-1)\times 7\times 13\\
z_6^2 &\equiv 86^2 \equiv 80 \equiv 2^4\times 5
\end{aligned}
$$

因此我们得到 6 个分解式，从而得到$(\mathbb{Z}_2)^7$ 中的 6 个向量。这并不能足以保证有一个相关关系，但是在这种特殊情形下已经足够了。这 6 个向量如下：

$$
\begin{aligned}
\boldsymbol{a}_1 &= (1,0,0,1,0,0,1)\\
\boldsymbol{a}_2 &= (0,0,0,1,0,0,0)\\
\boldsymbol{a}_3 &= (0,0,0,0,1,0,0)\\
\boldsymbol{a}_4 &= (1,0,0,0,0,1,0)\\
\boldsymbol{a}_5 &= (1,0,0,0,1,0,1)\\
\boldsymbol{a}_6 &= (0,0,0,1,0,0,0)
\end{aligned}
$$

显然 $\boldsymbol{a}_2 + \boldsymbol{a}_6 = (0,0,0,0,0,0,0)$；然而，你可以检查出这个相关关系并不能得到一个 n 的分解。可以起作用的相关关系是：

$$\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3 + \boldsymbol{a}_5 = (0,0,0,0,0,0,0)$$

我们得到的同余式是：

$$(42\times 43\times 61\times 85)^2 \equiv (2\times 3\times 5\times 7\times 13)^2 \pmod{1829}$$

通过化简，可以得到：

$$1459^2 \equiv 901^2 \pmod{1829}$$

然后直接计算

$$\gcd(1459 + 901, 1829) = 59$$

于是我们得到 n 的一个非平凡因子。

一个重要的一般问题是因子基应该取多大(作为要分解的整数 n 的函数)，以及算法的复杂性如何。通常，有如下认识：如果 $b = |\mathcal{B}|$ 很大，那么整数 $z^2 \bmod n$ 似乎可以在$\mathcal{B}$上分解。但是 b 越大，为了找到一个相关关系我们就要堆积越多的同余式。数论中的一些结论可以帮助我们更好地选取 b。我们现在讨论一些主要的想法。这将是一个启发式的分析，并假定整数 z 随机选取。

假定 n 和 m 为正整数。我们称 n 是 m 光滑的(m-smooth)，如果 n 的任一素因子都小于等于 m。$\psi(n,m)$定义为小于等于 n 且是 m 光滑的正整数的个数。数论中一个重要的结论是，如果 $n \gg m$，那么，

$$\frac{\psi(n,m)}{n} \approx \frac{1}{u^u}$$

其中 $u = \log n / \log m$。注意，$\psi(n, m)/n$ 表示从集合 $\{1, \cdots, n\}$ 中随机选取一个整数是 m 光滑的概率。

假定 $n \approx 2^r$ 和 $m \approx 2^s$。那么，

$$u = \frac{\log n}{\log m} \approx \frac{r}{s}$$

一个 r 比特的整数除以一个 s 比特的整数执行时间为 $O(rs)$。如果我们假定 $r < m$，我们可以在时间 $O(rsm)$ 内判断集合 $\{1, \cdots, n\}$ 中一个整数是否为 m 光滑的(参见练习)。

因子基 $\mathcal{B}$ 包含所有小于等于 m 的素数。因此，应用素数定理，我们有：

$$|\mathcal{B}| = b = \pi(m) \approx \frac{m}{\ln m}$$

为使算法成功，我们需要找出略多于 b 个的 m 光滑的模 n 平方数。我们期望测试 bu^u 个整数就可以得到 b 个 m 光滑的整数。因此，找到所需 m 光滑的平方数的期望时间为 $O(bu^u \times rsm)$。我们已经有 b 为 $O(m/s)$，所以算法的第一部分的运行时间为 $O(rm^2 u^u)$。

在算法的第二部分，我们需要化简模 2 的关联矩阵，构造形如 $x^2 \equiv y^2 \pmod{n}$ 的同余式，再应用 Euclidean 算法。可以容易地检验出完成这几步所需时间是 r 和 m 的多项式，即 $O(r^i m^j)$，其中 i 和 j 为正整数(一般来说，算法的第二部分最多做两次就完成了，因为一个同余式不能提供一个 n 的分解的概率最多为 1/2。这样导致了一个不大于 2 的因子，但它可以由高次项吸收进去)。

到此为止，我们知道算法的全部运行时间可写成 $O(rm^2 u^u + r^i m^j)$ 形式。回忆一下给定的 $n \approx 2^r$，我们可以试着选择 $m \approx 2^s$ 来优化运行时间。m 的一个不错的选择是取 $s \approx \sqrt{r \log_2 r}$。那么，

$$u \approx \frac{r}{s} \approx \sqrt{\frac{r}{\log_2 r}}$$

现在我们计算

$$\begin{aligned}
\log_2 u^u &= u \log_2 u \\
&\approx \sqrt{\frac{r}{\log_2 r}} \log_2 \left(\sqrt{\frac{r}{\log_2 r}} \right) \\
&< \sqrt{\frac{r}{\log_2 r}} \log_2 \sqrt{r} \\
&= \sqrt{\frac{r}{\log_2 r}} \frac{\log_2 r}{2} \\
&= \frac{\sqrt{r \log_2 r}}{2}
\end{aligned}$$

于是有

$$u^u \leqslant 2^{0.5\sqrt{r \log_2 r}}$$

我们又有 $m \approx 2^{\sqrt{r \log_2 r}}$ 和 $r = 2^{\log_2 r}$。因此，所有运行的时间可以写成如下形式：

$$O(2^{\log_2 r+2\sqrt{r\log_2 r}+0.5\sqrt{r\log_2 r}}+2^{i\log_2 r+j\sqrt{r\log_2 r}})$$

容易看出可以写成

$$O(2^{c\sqrt{r\log_2 r}})$$

其中 c 为某一常数。利用 $r\approx\log_2 n$,我们得到运行时间为

$$O(2^{c\sqrt{\log_2 n\log_2\log_2 n}})$$

通常运行时间会表示为相对于基 e 的对数和指数的项。利用 m 的最优选择,更精确的分析可以导出如下通常的期望运行时间:

$$O(e^{(1+o(1))\sqrt{\ln n\ln\ln n}})$$

5.6.4　实际中的分解算法

一个非常著名的,并在实际中广泛应用的算法是由 Pomerance 提出的二次筛法。"二次筛法"名字来源于判定 $z^2 \bmod n$ 在$\mathcal{B}$上分解的一个筛法过程(我们在这里不给出描述)。数域筛法是 20 世纪 80 年代后期发展起来的一个分解算法。它也是通过构造同余式 $x^2\equiv y^2 \pmod n$来分解 n,但它是在代数整数环中进行计算。最近,数域筛法成为分解大整数所选择的算法。

二次筛法、椭圆曲线算法和数域筛法的渐进运行时间分别为:

二次筛法	$O(e^{(1+o(1))\sqrt{\ln n\ln\ln n}})$
椭圆曲线算法	$O(e^{(1+o(1))\sqrt{2\ln p\ln\ln p}})$
数域筛法	$O(e^{(1.92+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}})$

记号 $o(1)$表示一个当 $n\to\infty$时趋向于 0 的函数,p 表示 n 的最小素因子。

在最坏的情况下,$p\approx\sqrt{n}$,二次筛法和椭圆曲线算法的渐进运行时间本质上一样。但在这种情况下,二次筛法一般快于椭圆曲线算法。如果 n 的素因子具有不同的长度,那么椭圆曲线算法是更有效的。Brent 在 1988 年使用椭圆曲线方法分解了一个很大的 Fermat 数 $2^{2^{11}}+1$。

直到 20 世纪 90 年代中期,二次筛法是分解 RSA 模(所谓 RSA 模是指 $n=pq$,p 和 q 是两个不同的素数,p 和 q 的长度大致相等)的最常用的算法。数域筛法是这三个算法中最近发展起来的算法。该算法跟其他算法相比具有的优点是,它的渐进运行时间比二次筛法和椭圆曲线算法的渐进运行时间都少。已证明数域筛法对大于 125 130 比特的十进制数是较快的算法。数域筛法的一个早期应用是在 1990 年,Lenstra、Manasse 和 Pollard 利用这个算法把 $2^{2^9}-1$ 分解成三个素数,这三个素数分别是 7 位、49 位和 99 位的十进制数。

在因子分解中一些著名的里程碑包括下面的事例。1983 年,用二次筛法成功地分解了一个 69 位的十进制整数,该数是 $2^{251}-1$ 的一个(合数)因子(由 Davis、Holdridge 和 Simmons 完成运算)。整个 20 世纪 80 年代又取得不断的进步。1989 年,Lenstra 和 Manasse 利用二次筛法并且通过把计算分配给成百上千个工作站的办法成功地分解了一个 106 位的十进制整数(他们称

这种方式为“通过电子邮件分解”)。

1994 年 4 月,Atkins、Graff、Lenstra 和 Leyland 使用二次筛法分解了一个称作 RSA-129 的 129 位的十进制整数(整数 RSA-100,RSA-110,...,RSA-500 是 RSA 模的一个列表,发布在因特网上为分解算法提供“挑战”数字。每一个数字 RSA-d 是一个 d 位的十进制整数,该数是大约等长的两个素数的乘积)。RSA-129 的分解耗去了由世界各地 600 多个研究者贡献的 5000MIPS 年的计算时间。

RSA-130 在 1996 年被分解;RSA-140 在 1999 年 2 月被分解;RSA-155 的分解在 1999 年 8 月 22 日完成。这三个分解都是利用数域筛法完成的。

这些分解中最后一个,RSA-155 包括一个 512 比特的二进制模(注意,155 位的十进制数大体上是 512 比特)。它需要在 6 个国家的大约 300 台 PC 和工作站的 8400MIPS 年的计算时间。这个分解提供的可信的证据说明,作为现在商业上广泛使用的 RSA 实现,512 比特 RSA 模,不应该认为是安全的。

在求解 Simon Singh 的《The Code Book》一书中第十个“密码挑战”中,另一个 155 位的 RSA 十进制模数的分解也已完成。这个分解由五个瑞典研究者利用数域筛法于 2000 年 10 月完成。

推测在未来的分解因子的发展趋势,有人提出 768 比特的模数将在 2010 年可以分解,1024 比特的模数到 2018 年可以分解。

5.7 对 RSA 的其他攻击

在这一节中,我们专注于如下问题:对 RSA 密码体制除了分解 n 之外是否还有其他可能攻击?例如,可以设想存在一种不用分解模数 n 就可解密 RSA 密文的方法。

5.7.1 计算 $\phi(n)$

我们首先看到计算 $\phi(n)$并不比分解 n 简单。例如,假设 n 和 $\phi(n)$已知,n 为两个素数 p 和 q 的乘积,那么 n 可以容易地分解,通过求解如下两个方程:

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

得到两个“未知数”p 和 q。按如下方法,这可容易地完成。如果我们用 $q = n/p$ 代入第二个方程中,我们可以得到一个关于未知数 p 的二次方程:

$$p^2 - (n - \phi(n) + 1)p + n = 0 \tag{5.1}$$

方程(5.1)的两个根就是 p 和 q,即 n 的因子。因此,如果一个密码分析者能够求出 $\phi(n)$的值,他就能分解 n,进而攻破系统。换句话说,计算 $\phi(n)$并不比分解 n 容易。

下面给出一个例子。

例 5.13 假定 $n = 84\,773\,093$,且敌手已经得到 $\phi(n) = 84\,754\,668$。这些信息给出了如下

的二次方程：

$$p^2 - 18\,426p + 84\,773\,093 = 0$$

这可以用二次方程求根公式求解，得到两个根 9539 和 88 887。这就是 n 的两个因子。

5.7.2　解密指数

我们现在证明一个有趣的结论，如果解密指数 a 已知，那么 n 可以通过一个随机算法在多项式时间内分解。因此我们可以说计算 a（本质上）并不比分解 n 容易（然而，这并不能排除不用计算 a 就可攻破 RSA 密码体制的可能性）。注意，这一点并不仅仅出于理论兴趣。它告诉我们如果 a 被泄漏（由于意外或其他原因），那么 Bob 重新选一个加密指数是不够的，他必须选择一个新的模数 n。

我们要描述的算法是一个 Las Vegas 形式（参见 4.2.2 小节中的定义）的随机算法。这里，我们考虑 Las Vegas 算法具有最坏情形成功的概率最低为 $1-\varepsilon$。因此，对于任一问题实例，算法不能给出一个答案的概率最多为 ε。

如果我们有了一个这样的 Las Vegas 算法，那么我们只需一次又一次地运行它，直到找到一个答案为止。算法连续 m 次返回“没有答案”（no answer）的概率为 ε^m。为了得到一个答案，必须运行算法的平均次数（即期望值）是 $1/(1-\varepsilon)$（参见练习）。

我们将描述一个对于给定的值 a，b 和 n 作为输入，以至少 1/2 的概率分解 n 的算法。因此，如果算法运行 m 次，那么 n 被分解的概率至少为 $1-1/2^m$。

算法基于一些与 1 的平方根（模 n）相关的事实，这里 $n=pq$ 是两个不同奇素数的乘积若 $x^2 \equiv 1 \pmod{p}$，定理 5.13 告诉我们存在 1 模 n 的四个根。其中两个根为 $\pm 1 \bmod n$，称为 1 模 n 的平凡平方根；另两个根称为非平凡的，它们也是模 n 的互为相反数。

这里给出一个小例子。

例 5.14　假定 $n = 403 = 13 \times 31$。1 模 403 的四个平方根为 1，92，311 和 402。平方根 92 是通过使用中国剩余定理求解如下方程组得到的：

$$x \equiv 1 \pmod{13}$$
$$x \equiv -1 \pmod{31}$$

另外一个非平凡平方根是 $403 - 92 = 311$。它是如下方程组的解：

$$x \equiv -1 \pmod{13}$$
$$x \equiv 1 \pmod{31}$$

假定 x 为一个模 n 的非平凡平方根。那么，

$$x^2 \equiv 1^2 \pmod{n}$$

但

$$x \not\equiv \pm 1 \pmod{n}$$

那么,在随机平方分解算法中,我们可以通过计算 $\gcd(x+1,n)$和 $\gcd(x-1,n)$来分解 n。在例 5.14 中,

$$\gcd(93,403)=31$$

且

$$\gcd(312,403)=13$$

算法 5.10 试图通过寻找 1 模 n 的非平凡平方根来分解 n。在分析算法之前,我们先给出一个例子描述其应用。

例 5.15 假定 $n=89\,855\,713$, $b=34\,986\,517$,且 $a=82\,330\,933$,取随机数 $w=5$。我们有

$$ab-1=2^3\times 360\,059\,073\,378\,795$$

那么,

$$w^r \bmod n=85\,877\,701$$

恰好有:

$$85\,877\,701^2\equiv 1(\bmod n)$$

因此算法将返回:

$$x=\gcd(85\,877\,702,n)=9103$$

这是 n 的一个因子;另一个因子是 $n/9103=9871$。

算法 5.10 RSA-FACTOR(n,a,b)

comment: 我们假定 $ab\equiv 1(\bmod \phi(n))$

记 $ab-1=2^s r$, r 为奇数

随机选择 w 使得 $1\leqslant w\leqslant n-1$

$x\leftarrow \gcd(w,n)$

if $1<x<n$

  then return (x)

comment: x 是 n 的一个因子

$v\leftarrow w^r \bmod n$

if $v\equiv 1(\bmod n)$

  then return ("failure")

while $v\not\equiv 1(\bmod n)$

  do $\begin{cases} v_0\leftarrow v \\ v\leftarrow v^2 \bmod n \end{cases}$

if $v_0\equiv -1(\bmod n)$

then return ("failure")

else { $x \leftarrow \gcd(v_0 + 1, n)$; **return**(x) }

comment: x 是 n 的一个因子

让我们来分析一下算法 5.10。首先,如果我们幸运地选到 w 为 p 或 q 的倍数,那么我们可以直接分解 n。如果 w 与 n 互素,那么我们通过连续平方计算 $w^r, w^{2r}, w^{4r}, \cdots$,直到对于某个 t,有

$$w^{2^t r} \equiv 1 (\bmod\ n)$$

由于

$$ab - 1 = 2^s r \equiv 0 (\bmod\ \phi(n))$$

我们知道 $w^{2^t r} \equiv 1 (\bmod\ n)$。因此,**while** 循环至多运行 s 次就会终止。**while** 循环结束后,我们找到一个值 v_0 使得 $(v_0)^2 \equiv 1 (\bmod\ n)$,但是 $v_0 \not\equiv 1 (\bmod\ n)$。如果 $v_0 \equiv -1 (\bmod\ n)$,那么算法失败;否则,v_0 是 1 模 n 的一个非平凡平方根,且我们能够分解 n。

现在我们面临的主要任务是证明算法的成功概率至少为 1/2。有两种方式使得算法分解 n 失败:

1. $w^r \equiv 1 (\bmod\ n)$

2. $w^{2^t r} \equiv -1 (\bmod\ n)$,对于某个 $t, 0 \leqslant t \leqslant s - 1$

我们有 $s + 1$ 个同余方程要考虑。如果 w 是这 $s + 1$ 个同余方程中至少一个的解,那么它是一个"坏"选择,算法失败。所以我们下面讨论其中每一个同余方程的解的个数。

首先,考虑同余方程 $w^r \equiv 1 (\bmod\ n)$。分析这个方程的解的方法是分别考虑模 p 和模 q 的解,然后用中国剩余定理把它们组合起来。可以看出 $x \equiv 1 (\bmod\ n)$ 当且仅当 $x \equiv 1 (\bmod\ p)$ 且 $x \equiv 1 (\bmod\ q)$。

因此,我们先考虑 $w^r \equiv 1 (\bmod\ p)$。由于 p 是一个素数,由定理 5.7,$\mathbb{Z}_p^*$ 是一个循环群。设 g 为模 p 的一个本原元素。我们可以写成 $w = g^u$,对于某一惟一的整数 $u, 0 \leqslant u \leqslant p - 2$。于是我们有:

$$w^r \equiv 1 (\bmod\ p)$$
$$g^{ur} \equiv 1 (\bmod\ p)$$

因此,$(p - 1) \mid ur$。我们记 $p - 1 = 2^i p_1$,其中 p_1 是一个奇数;$q - 1 = 2^j q_1$,其中 q_1 是一个奇数。由于

$$\phi(n) = (p - 1)(q - 1) \mid (ab - 1) = 2^s r$$

因此 $i + j \leqslant s$,且 $p_1 q_1 \mid r$。现在,条件 $(p - 1) \mid ur$ 变成了 $2^i p_1 \mid ur$。由于 $p_1 \mid r$ 且 r 为奇数,因此充要条件是 $2^i \mid u$。因此,$u = k2^i, 0 \leqslant k \leqslant p_1 - 1$,且同余方程 $w^r \equiv 1 (\bmod\ p)$ 的解的个数为 p_1。

经过相同的过程,同余方程 $w^r \equiv 1 (\bmod\ q)$ 恰好有 q_1 个解。我们把任一模 p 的解和模 q

的解组合起来,利用中国剩余定理,就可得到模 n 的解。因此,同余方程 $w^r \equiv 1 \pmod{n}$ 的解的个数为 $p_1 q_1$。

下一步是对于固定的 $t(0 \leqslant t \leqslant s-1)$,考虑同余方程 $w^{2^t r} \equiv -1 \pmod{n}$ 的解。再一次,我们首先考虑模 p 和模 q 的解(注意,$w^{2^t r} \equiv -1 \pmod{n}$ 当且仅当 $w^{2^t r} \equiv -1 \pmod{p}$ 且 $w^{2^t r} \equiv -1 \pmod{q}$)。先考虑 $w^{2^t r} \equiv -1 \pmod{p}$。像上面一样,记 $w = g^u$,我们有:

$$g^{u2^t r} \equiv -1 \pmod{p}$$

由于 $g^{(p-1)/2} \equiv -1 \pmod{p}$,我们有:

$$u2^t r \equiv \frac{p-1}{2} \pmod{p-1}$$

$$(p-1) \mid \left(u2^t r - \frac{p-1}{2} \right)$$

$$2(p-1) \mid (u2^{t+1} r - (p-1))$$

由于 $p - 1 = 2^i p_1$,我们得到:

$$2^{i+1} p_1 \mid (u2^{t+1} r - 2^i p_1)$$

取出公因子 p_1,上式变为:

$$2^{i+1} \mid \left(\frac{u2^{t+1} r}{p_1} - 2^i \right)$$

现在,如果 $t \geqslant i$,无解,由于此时 $2^{i+1} \mid 2^{t+1}$,但 $2^{i+1} \nmid 2^i$。另一方面,如果 $t \leqslant i-1$,那么 u 是一个解当且仅当 u 是 2^{i-t-1} 的奇数倍(注意,r/p_1 是奇数)。所以这种情形下解的个数为:

$$\frac{p-1}{2^{i-t-1}} \times \frac{1}{2} = 2^t p_1$$

通过类似的推理,同余方程 $w^{2^t r} \equiv -1 \pmod{q}$ 当 $t \geqslant j$ 时无解,当 $t \leqslant j-1$ 时有 $2^t q_1$ 个解。利用中国剩余定理,我们知道 $w^{2^t r} \equiv -1 \pmod{n}$ 的解的个数为:

$$\begin{cases} 0 & t \geqslant \min\{i, j\} \\ 2^{2t} p_1 q_1 & t \leqslant \min\{i, j\} - 1 \end{cases}$$

现在,t 可以从 0 到 $s-1$ 取值。不失一般性,假定 $i \leqslant j$,那么当 $t \geqslant i$ 时解的个数为零。对于 w 的“坏”选择的总数最多为:

$$p_1 q_1 + p_1 q_1 (1 + 2^2 + 2^4 + \cdots + 2^{2i-2}) = p_1 q_1 \left(1 + \frac{2^{2i} - 1}{3} \right)$$

$$= p_1 q_1 \left(\frac{2}{3} + \frac{2^{2i}}{3} \right)$$

前面已知 $p - 1 = 2^i p_1$,且 $q - 1 = 2^j q_1$。现在,$j \geqslant i \geqslant 1$,所以 $p_1 q_1 < n/4$。我们又有

$$2^{2i} p_1 q_1 \leqslant 2^{i+j} p_1 q_1 = (p-1)(q-1) < n$$

因此,我们得到:

$$p_1 q_1\left(\frac{2}{3}+\frac{2^{2i}}{3}\right)<\frac{n}{6}+\frac{n}{3}$$
$$=\frac{n}{2}$$

由于至多$(n-1)/2$个 w 的选择是“坏”的,容易知道至少有$(n-1)/2$个选择是“好”的,因此算法成功的概率至少为 1/2。

5.7.3　Wiener 的低解密指数攻击

像前面一样,我们假定 $n=pq$,其中 p 和 q 为素数,那么 $\phi(n)=(p-1)(q-1)$。在本小节中,我们介绍由 M.Wiener 提出的一种攻击,可以成功地计算秘密的解密指数 a,前提是满足如下条件:

$$3a < n^{1/4} \quad 且 \quad q < p < 2q \tag{5.2}$$

如果 n 的二进制表示有 l 比特,那么当 a 的二进制表示的位数小于 $l/4-1$,且 p 和 q 相距不太远时攻击有效。

注意,Bob 可能试图选择较小的解密指数来加快解密过程。如果他使用算法 5.5 来计算 $y^a \bmod n$,那么当选择一个满足(5.2)式的 a 值时,解密的时间大致可以减少 75%。本小节中我们证明的结果说明,应该避免采用这种办法来减少解密时间。

由于 $ab\equiv 1(\bmod \phi(n))$,可知存在一个整数 t 使得:

$$ab-t\phi(n)=1$$

由于 $n=pq>q^2$,我们有 $q<\sqrt{n}$。因此,

$$0<n-\phi(n)=p+q-1<2q+q-1<3q<3\sqrt{n}$$

现在,我们看到:

$$\left|\frac{b}{n}-\frac{t}{a}\right|=\left|\frac{ba-tn}{an}\right|$$
$$=\left|\frac{1+t(\phi(n)-n)}{an}\right|$$
$$<\frac{3t\sqrt{n}}{an}$$
$$=\frac{3t}{a\sqrt{n}}$$

由于 $t<a$,我们有 $3t<3a<n^{1/4}$,因此,

$$\left|\frac{b}{n}-\frac{t}{a}\right|<\frac{1}{an^{1/4}}$$

最后,由于 $3a < n^{1/4}$,我们有:

$$\left| \frac{b}{n} - \frac{t}{a} \right| < \frac{1}{3a^2}$$

因此,分数 t/a 是分数 b/n 的一个很近的近似。从连分数的理论可知,这样接近的近似值是 b/n的连分数展开的一个收敛子(参看定理 5.14)。这种扩展可以像下面描述的这样,由 Euclidean 算法得到。

一个(有限)连分数是一个非负整数的 m 组,即

$$[q_1, \cdots, q_m]$$

它是下面表达式的简写形式:

$$q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cdots + \cfrac{1}{q_m}}}$$

假定 a 和 b 为满足 $\gcd(a,b) = 1$ 的正整数,且假定算法 5.1 的输出为 m 组 $(q_1, \cdots, q_m)$。那么容易看出 $a/b = [q_1, \cdots, q_m]$。我们称 $[q_1, \cdots, q_m]$ 是 a/b 在这种情形下的连分数展开(continued fraction expansion)。现在,对于 $1 \leqslant j \leqslant m$,定义 $C_j = [q_1, \cdots, q_j]$。C_j 称为 $[q_1, \cdots, q_m]$ 的第 j 个收敛子。每一 C_j 可以写成有理数形式 c_j/d_j,其中 c_j 和 d_j 满足如下的递推关系:

$$c_j = \begin{cases} 1 & j = 0 \\ q_1 & j = 1 \\ q_j c_{j-1} + c_{j-2} & j \geqslant 2 \end{cases}$$

且

$$d_j = \begin{cases} 0 & j = 0 \\ 1 & j = 1 \\ q_j d_{j-1} + d_{j-2} & j \geqslant 2 \end{cases}$$

例 5.16 我们计算 34/99 的连续分数展开。用 Euclidean 算法进行如下计算:

$$\begin{aligned} 34 &= 0 \times 99 + 34 \\ 99 &= 2 \times 34 + 31 \\ 34 &= 1 \times 31 + 3 \\ 31 &= 10 \times 3 + 1 \\ 1 &= 3 \times 1 \end{aligned}$$

因此,34/99 的连续分数展开为[0,2,1,10,3],即

$$\frac{34}{99} = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{10 + \cfrac{1}{3}}}}$$

这个连分数的收敛子如下：

$$[0] = 0$$
$$[0,2] = 1/2$$
$$[0,2,1] = 1/3$$
$$[0,2,1,10] = 11/32$$
$$[0,2,1,10,3] = 34/99$$

读者可以用前面的递推关系式来计算并验证这些收敛子。

一个有理数的连分数展开的收敛子满足很多有趣的性质。基于我们的目的，最重要的性质如下。

定理 5.14　假定 $\gcd(a,b) = \gcd(c,d) = 1$ 且

$$\left| \frac{a}{b} - \frac{c}{d} \right| < \frac{1}{2d^2}$$

那么，c/d 是 a/b 连分数展开的一个收敛子。

现在我们可以把这个结论应用到 RSA 密码体制上。我们已经知道，如果条件(5.2)成立，那么未知分数 t/a 是 b/n 的一个很接近的近似。定理 5.14 告诉我们，t/a 一定是 b/n 的连分数展开的一个收敛子。既然 b/n 是公开信息，那么很容易计算它的收敛子。我们所需要的仅是一个测试它们中间哪一个是“正确”的方法。

但这也并不难做到。如果 t/a 是 b/n 的一个收敛子，那么我们能够计算 $\phi(n)$ 的值为 $\phi(n) = (ab-1)/t$。一旦 n 和 $\phi(n)$ 已知，我们可以通过求解关于 p 的二次方程(5.1)来分解 n。我们事先并不知道 b/n 的哪个收敛子能得到 n 的分解，所以我们依次试用它们，直到找到 n 的分解为止。如果我们不能用这种方法分解 n，那必然是假设(5.2)不成立的情形。

Wiener 算法的伪代码描述参见算法 5.11。

算法 5.11　Wiener Algorithm(n, b)

$(q_1, \cdots, q_m; r_m) \leftarrow$ Euclidean Algorithm(n, b)

$c_0 \leftarrow 1$

$c_1 \leftarrow q_1$

$d_0 \leftarrow 0$

$d_1 \leftarrow 1$

```
j ← 1
while j ≤ m
    do { n' ← (d_j b − 1)/c_j
         comment: n' = φ(n),如果 c_j/d_j 是正确的收敛子
         if n' 是一个整数
            then { 设 p 和 q 为方程 x^2 − (n − n' + 1)x + n = 0 的根
                   if p 和 q 为小于 n 的整数
                      then return(p, q) }
         j ← j + 1
         c_j ← q_j c_{j−1} + c_{j−2}
         d_j ← q_j d_{j−1} + d_{j−2} }
return("failure")
```

我们提供一个描述上述算法的例子。

例 5.17 假定 $n = 160\,523\,347$,且 $b = 60\,728\,973$。b/n 的连分数展开为:

$$[0,2,1,1,1,4,12,102,1,1,2,3,2,2,36]$$

最初的几个收敛子为:

$$0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{14}{37}$$

读者可以验证前五个收敛子并不能产生 n 的分解。然而,收敛子 14/37 得到

$$n' = \frac{37 \times 60\,728\,973 - 1}{14} = 160\,498\,000$$

现在,我们求解方程

$$x^2 - 25\,348x + 160\,523\,347 = 0$$

得到根 $x = 12\,347, 13\,001$。因此我们发现了分解:

$$60\,523\,347 = 12\,347 \times 13\,001$$

注意,对于 $n = 160\,523\,347$,Wiener 算法适用条件:

$$a < \frac{n^{1/4}}{3} \approx 37.52$$

5.8 Rabin 密码体制

在本节中,我们描述 Rabin 密码体制,假定模数 $n = pq$ 不能被分解,则该类体制对于选择明

文攻击是计算安全的。因此,Rabin 密码体制提供了一个可证明安全的密码体制的例子:假定分解整数问题是计算上不可行的,Rabin 密码体制是安全的。我们现在描述 Rabin 密码体制。

密码体制 5.2　Rabin 密码体制

设 $n = pq$,其中 p 和 q 为素数,且 $p, q \equiv 3 \pmod 4$。设 $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n^*$,且定义:

$$\mathcal{K} = \{(n, p, q)\}$$

对于 $K = (n, p, q)$,定义

$$e_K(x) = x^2 \bmod n$$

和

$$d_K(y) = \sqrt{y} \bmod n$$

n 的值为公钥,p 和 q 为私钥。

注　条件 $p, q \equiv 3 \pmod 4$ 可以省去。又,如果我们取 $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ 而非 $\mathbb{Z}_n^*$,密码体制仍能"工作"。然而,我们使用了更多的限制性描述,简化了许多方面的计算和密码体制的分析。

Rabin 密码体制的一个缺点是加密函数并不是一个单射,所以解密不能以一种明显的方式完成。我们证明如下。假定 y 是一个有效的密文,这意味着 $y = x^2 \bmod n$,对于某一 $x \in \mathbb{Z}_n^*$。定理 5.13 证明了存在 y 模 n 的四个解,是对应于密文 y 的四个可能的解。一般地,Bob 并不能区别这四个可能的明文中哪一个是"正确"的,除非明文中包含足够的冗余信息来消去四个可能值中的三个。

让我们从 Bob 的观点来看解密问题。他得到一个密文 y,且想找出 x 使得

$$x^2 \equiv y \pmod n$$

这是一个关于 $\mathbb{Z}_n$ 中未知元 x 的二次方程,解密需要求出模 n 的平方根。这等价于求解两个同余方程:

$$z^2 \equiv y \pmod p$$

和

$$z^2 \equiv y \pmod q$$

我们可以利用 Euler 准则来判断 y 是否为一个模 p(或模 q)的二次剩余。事实上,如果加密正确地执行,则 y 是一个模 p 和模 q 的二次剩余。不幸的是,Euler 准则并不能帮我们找到 y 的平方根;它仅能得到一个"是"或"否"的答案。

当 $p \equiv 3 \pmod 4$ 时,有一个简单公式来计算模 p 二次剩余的平方根。假定 y 是一个模 p 二次剩余,且 $p \equiv 3 \pmod 4$,那么我们有:

$$\begin{aligned}(\pm y^{(p+1)/4})^2 &\equiv y^{(p+1)/2} \pmod p \\ &\equiv y^{(p-1)/2} y \pmod p\end{aligned}$$

$$\equiv y(\mathrm{mod}\ p)$$

这里我们又一次使用了 Euler 准则,即如果 y 是一个模 p 二次剩余,那么 $y^{(p-1)/2} \equiv 1(\mathrm{mod}\ p)$。因此,$y$ 模 p 的两个平方根为 $\pm y^{(p+1)/4} \bmod p$。同样的讨论可知,y 模 q 的两个平方根为 $\pm y^{(q+1)/4} \bmod q$。然后可以直接用中国剩余定理来得到 y 模 n 的四个平方根。

注 对于 $p \equiv 1(\mathrm{mod}\ 4)$,还不知道是否存在多项式时间的确定算法来计算模 p 二次剩余的平方根(尽管已有一个多项式时间的 Las Vegas 算法)。这就是我们在 Rabin 密码体制的定义中规定 $p, q \equiv 3(\mathrm{mod}\ 4)$的原因。

例 5.18 我们用一个小例子来说明 Rabin 密码体制的加密和解密过程。假定 $n = 77 = 7 \times 11$。那么加密函数为:

$$e_K(x) = x^2 \bmod 77$$

且解密函数为:

$$d_K(y) = \sqrt{y} \bmod 77$$

假定 Bob 需要解密密文 $y = 23$。首先需要找到 23 模 7 和模 11 的平方根。由于 7 和 11 都是模 4 余 3,我们利用前面的公式:

$$23^{(7+1)/4} \equiv 2^2 \equiv 4(\mathrm{mod}\ 7)$$

$$23^{(11+1)/4} \equiv 1^3 \equiv 1(\mathrm{mod}\ 11)$$

利用中国剩余定理,我们计算 23 模 77 的四个平方根为 $\pm 10, \pm 32 \bmod 77$。因此,四个可能的明文为 $x = 10, 32, 45, 67$。可以验证这四个明文平方后模 77 约化得到的值为 23。这证明了 23 是一个有效的密文。

5.8.1 Rabin 密码体制的安全

我们现在讨论 Rabin 密码体制的(可证明)安全性。安全性证明使用了一个 Turing 约化,其定义如下。

定义 5.5 假定 **G** 和 **H** 为问题(problem)。一个从 **G** 到 **H** 的 Turing 约化是一个算法 SolveG,具有如下性质:

1. SolveG 假定了存在任意算法 SolveH 求解问题 **H**;

2. SolveG 可以调用 SolveH 并使用它的任一输出值,但 SolveG 不能对于 SolveH 执行的实际运算做任何限定(也就是说,SolveH 被看做一个"黑盒子",称为一个预言(oracle));

3. SolveG 是一个多项式时间算法;

4. SolveG 正确地求解问题 **G**。

如果存在一个从 **G** 到 **H** 的 Turing 约化,我们记为 $\mathbf{G} \propto_T \mathbf{H}$。

一个 Turing 约化 $\mathbf{G} \propto_T \mathbf{H}$ 并不一定得到一个求解问题 **G** 的多项式时间算法。它实际上证明了如下隐含的事实：

如果存在一个多项式时间算法求解问题 **H**，*那么存在一个多项式时间算法求解问题* **G**。

这是因为任一求解 **H** 的算法 SolveH 可以“插入”到算法 SolveG 中，从而产生一个求解 **G** 的算法。显然，如果 SolveH 是一个多项式时间的算法，那么得到的算法是多项式时间的算法，反之亦然。

我们将提供 Turing 约化的一个清晰的例子：我们将证明，一个解密预言（decryption oracle）Rabin Decrypt 可以并入到一个分解模数 n 的 Las Vegas 算法中，具有至少 1/2 的概率。也就是说，我们可得到 **Factoring** $\propto_T$ **Rabin decryption**，其中 Turing 约化本身是一个随机算法。在下面的算法中，我们假定 n 是两个不同素数 p 和 q 的乘积；Rabin Decrypt 是一个执行 Rabin 解密过程的预言，对一给定密文返回对应的四个可能明文中的一个。

算法 5.12　Rabin Oracle Factoring(n)

external　Rabin Decrypt

选择一个随机整数，$r \in \mathbb{Z}_n^*$

$y \leftarrow r^2 \bmod n$

$x \leftarrow$ Rabin Decrypt(y)

if $x \equiv \pm r \pmod n$

　then return(“failure”)

　else { $p \leftarrow \gcd(x + r, n)$; $q \leftarrow n/p$; **return**(“$n = p \times q$”) }

这里需要做几点说明。首先，观察到 y 是一个有效的密文，且 Rabin Decrypt(y)将返回四个可能明文中的一个作为 x 的值。实际上，有下式成立：$x \equiv \pm r \pmod n$ 或者 $x \equiv \pm \omega r \pmod n$，其中 ω 是 1 模 n 的一个非平凡的平方根。在第二种情形下，我们有 $x^2 \equiv r^2 \pmod n$，但是 $x \not\equiv \pm r \pmod n$。因此，计算 $\gcd(x + r, n)$ 一定能够得出 p 或者 q，这就完成了 n 的分解。

下面计算算法在所有可能选择的随机值 $r \in \mathbb{Z}_n^*$ 的成功概率。对于一个剩余 $r \in \mathbb{Z}_n^*$，定义：

$$[r] = \{\pm r \bmod n, \pm \omega r \bmod n\}$$

显然，在$[r]$中分别有两个剩余用算法 5.12 得到相同的 y 值，且由预言 Rabin Decrypt 给出的输出 x 值也在$[r]$中。我们已经观察到算法 5.12 能够成功当且仅当 $x \equiv \pm \omega r \pmod n$。这个预言并不知道用四个可能的 r 值中哪一个来构造 y，且 r 是在调用预言 Rabin Decrypt 之前随机选择的。因此，$x \equiv \pm \omega r \pmod n$ 的概率是 1/2。我们得出结论算法 5.12 成功的概率为 1/2。

我们已经证明了 Rabin 密码体制对选择明文攻击是可证明安全的。然而，这个体制对于

选择密文攻击是完全不安全的。事实上,算法 5.12 可以用来在选择密文攻击中攻破 Rabin 密码体制! 在选择密文攻击中,(假想的)预言 Rabin Decrypt 被一个实际解密算法取代。(非正式地,安全性证明中说解密预言可以用来分解 n;且被选择的密文攻击假定一个解密预言存在。合起来,这就攻破了密码体制!)

5.9 RSA 的语义安全

到现在为止,我们假定敌手试图攻破密码体制以找出秘密密钥(对称密码体制下的情形)或者私钥(公钥密码体制下的情形)。如果 Oscar 能够做到这一点,那么密码体制被完全攻破。然而,可能敌手的目标并没有这么大的野心。即使 Oscar 不能找到秘密密钥或公钥,他仍可以获得比我们所希望的更多的信息。如果我们要确保一个密码体制是"安全的",我们应该更准确地估计敌手的目的。

下面列出了潜在的敌手的目的。

完全攻破(total break)

敌手能够找出 Bob 的秘密密钥(对称密码体制下的情形)或者私钥(公钥密码体制下的情形)。因此他能解密利用给定密钥加密的任意密文。

部分攻破(partial break)

敌手能以某一不可忽略的(non-negligible)概率解密以前没有见过的密文(无需知道密钥)。或者,敌手能够对于给定的密文,得出明文的一些特定信息。

密文识别(distinguishability of ciphertexts)

敌手能够以超过 1/2 的概率识别两个不同明文对应的密文,或者识别出给定明文的密文和随机字符串。

在下面的内容中,我们考虑一些针对 RSA 类密码体制达到上面某种类型目的的可能攻击。我们也介绍在一定的计算假设成立的情形下,如何构造一个公钥密码体制使得敌手不能(在多项式时间内)识别密文。这样的密码体制称为达到了语义安全(semantic security)。达到语义安全是非常困难的,因为我们是在对抗敌手非常弱的、容易达到目的的攻击。

5.9.1 与明文比特相关的部分信息

一些密码体制的弱点就是明文的部分信息可以通过密文"泄漏"出去。这表示是对系统的一种部分攻破,事实上,它在 RSA 密码体制中发生了。假定我们给定密文,$y = x^b \bmod n$,其中 x 表示明文。由于 $\gcd(b, \phi(n)) = 1$,必然是 b 为奇数的情形。因此 Jacobi 符号

$$\left(\frac{y}{n}\right) = \left(\frac{x}{n}\right)^b = \left(\frac{x}{n}\right)$$

所以,给定密文 y,任何人无需解密密文就可以有效地计算$\left(\frac{x}{n}\right)$。也就是说,一个 RSA 加密

“泄漏”了一些有关明文的信息，即 Jacobi 符号$\left(\frac{x}{n}\right)$的值。

在本小节中，我们考虑一些由密码体制泄漏的其他特定类型的部分信息：

1. 给定 $y = e_K(x)$，计算 $parity(y)$，其中 $parity(y)$表示 x 的二进制表示的最低位数(即，当 x 为偶数时 $parity(y) = 0$；x 为奇数时 $parity(y) = 1$)。

2. 给定 $y = e_K(x)$，计算 $half(y)$，其中当 $0 \leqslant x < n/2$ 时，$half(y) = 0$；当 $n/2 < x \leqslant n - 1$ 时，$half(y) = 1$。

我们将证明假定 RSA 加密是安全的，RSA 密码体制不会泄漏这种类型的信息。更精确地说，我们将证明 RSA 解密问题可以 Turing 约化为计算 $half(y)$的问题。这意味着，如果存在一个多项式算法计算 $half(y)$，那么存在 RSA 解密的多项式时间算法。也就是说，计算关于明文的特定信息，即 $half(y)$，不会比解密密文得到整个明文来得容易。

我们现在讨论，在给定计算 $half(y)$的假设算法(预言)的前提下，如何计算 $y = d_K(x)$。算法如算法 5.13 所描述。

算法 5.13　Oracle RSA Decryption(n, b, y)

```
external  Half
k ← ⌊log2 n⌋
for i ← 0  to  k
    do { h_i ← Half(n, b, y)
       { y ← (y × 2^b) mod n
l_o ← 0
hi ← n
for i ← 0 to k
    do { mid ← (hi + l_o)/2
       { if h_i = 1
         then l_o ← mid
         else hi ← mid
return(⌊hi⌋)
```

我们对算法的原理做一下解释。首先，我们注意到 RSA 加密函数满足在$\mathbb{Z}_n$ 中的如下乘法性质：

$$e_K(x_1)e_K(x_2) = e_K(x_1x_2)$$

现在，利用如下事实：

$$y = e_K(x) = x^b \bmod n$$

容易看到，对于 $0 \leqslant i \leqslant \lfloor \log_2 n \rfloor$，第一个 **for** 循环运行第 i 次时有：

$$h_i = half(y \times (e_K(2))^i) = half(e_K(x \times 2^i))$$

我们观察到

$$half(e_K(x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{2}\right)$$

$$half(e_K(2x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{4}\right) \cup \left[\frac{n}{2}, \frac{3n}{4}\right)$$

$$half(e_K(4x)) = 0 \Leftrightarrow x \in \left[0, \frac{n}{8}\right) \cup \left[\frac{n}{4}, \frac{3n}{8}\right) \cup \left[\frac{n}{2}, \frac{5n}{8}\right) \cup \left[\frac{3n}{4}, \frac{7n}{8}\right)$$

等等。因此,我们可以利用二分检索技巧来找到 x,这就是在第二个 **for** 循环中完成的。下面用一个小例子来说明。

例 5.19 假定 $n = 1457, b = 779$,且我们有一个密文 $y = 722$。然后假定,利用预言 Half,我们得到如下 h_i 值:

i	0	1	2	3	4	5	6	7	8	9	10
h_i	1	0	1	0	1	1	1	1	1	0	0

然后我们利用二分检索,过程如表 5.2 所示。因此,明文为 $x = \lfloor 999.55 \rfloor = 999$。

表 5.2 对于 RSA 加密的二分检索

i	lo	mid	hi
0	0.00	728.50	1457.00
1	728.50	1092.75	1457.00
2	728.50	910.62	1092.75
3	910.62	1001.69	1092.75
4	910.62	956.16	1001.69
5	956.16	978.92	1001.69
6	978.92	990.30	1001.69
7	990.30	996.00	1001.69
8	996.00	998.84	1001.69
9	998.84	1000.26	1001.69
10	998.84	999.55	1000.26
	998.84	999.55	999.55

不难看出算法 5.13 的算法复杂度是:

$$O((\log n)^3) + O(\log n) \times \text{Half 的复杂度}$$

因此,如果 Half 是一个多项式时间算法,我们就会得到一个 RSA 解密的多项式算法。

容易看到,计算 $parity(y)$是多项式等价于计算 $half(y)$。这是由于在 RSA 加密过程中涉及到如下两个等式(参见练习):

$$half(y) = parity((y \times e_K(2)) \bmod n) \tag{5.3}$$

$$parity(y) = half((y \times e_K(2^{-1})) \bmod n) \tag{5.4}$$

和前面提到的乘法规则 $e_K(x_1)e_K(x_2) = e_K(x_1x_2)$。因此，从上面证明的结果可知，如果存在计算 *parity*(y)的多项式时间算法，那么存在 RSA 解密的多项式时间算法。

我们已经提供证据说明计算 *parity* 或者 *half* 是很困难的，即 RSA 解密是很困难的。然而，我们提出的证明并没有排除如下的可能性：可能找到能以 75% 准确率计算 *parity* 或者 *half* 的有效算法。还有许多其他形式的可能泄漏的明文信息值得考虑，我们当然不能对所有可能的信息形式提供单独的证明。因此，本节只是对于特定形式的攻击提供一些安全证据。

5.9.2　最优非对称加密填充

我们真正想要做的是找到一个设计这样的密码体制的方法，这个密码体制允许我们证明（在一些似是而非的计算假定下）不可能在多项式时间内通过检查密文的手段找到任何有关明文的信息。可以证明这等价于证明如下问题，敌手不能区别密文。因此，我们考虑密文识别问题（Ciphertext Distinguishability），其定义如下：

问题 5.3　密文识别（Ciphertext Distinguishability）

条件：　一个加密函数 $f: X \to X$；两个明文 $x_1, x_2 \in X$；和一个密文 $y = f(x_i)$，其中 $i \in \{1,2\}$。

问题：　是否 $i = 1$?

如果加密函数 f 是确定性的（deterministic），问题 5.3 就当然是平凡的，由于此时能够计算 $f(x_1)$和 $f(x_2)$，然后看哪一个得到了密文 y。因此，如果要使密文识别是计算上不可行的，就必须要求加密过程是随机的。我们现在提出具体的方法来达到这个目标。我们描述密码体制 5.3，该体制基于一个任意的陷门单向置换（trapdoor one-way permutation），即从集合 X 到自身的（双射）陷门单向函数。如果 $f: X \to X$ 是一个陷门单向置换，那么其逆置换通常记为 f^{-1}。f 是加密函数，f^{-1}是公钥密码体制的解密函数。

密码体制 5.3　语义安全的公钥密码体制

设 m, k 为正整数；设 $\mathcal{F}$ 为一族陷门单向置换，对于任意 $f \in \mathcal{F}$，有 $f: \{0,1\}^k \to \{0,1\}^k$；且设 $G: \{0,1\}^k \to \{0,1\}^m$ 为一个随机预言。令 $\mathcal{P} = \{0,1\}^m$，且 $\mathcal{C} = \{0,1\}^k \times \{0,1\}^m$，并定义：

$$\mathcal{K} = \{(f, f^{-1}, G): f \in \mathcal{F}\}$$

对于 $K = (f, f^{-1}, G)$，随机选取 $r \in \{0,1\}^k$，且定义：

$$e_K(x) = (y_1, y_2) = (f(r), G(r) \oplus x)$$

其中 $x, y_1 \in \{0,1\}^k$, $y_2 \in \{0,1\}^m$。进一步,定义:

$$d_K(y_1, y_2) = G(f^{-1}(y_1)) \oplus y_2$$

($y_1 \in \{0,1\}^k$, $y_2 \in \{0,1\}^m$)。函数 f 和 G 为公钥,函数 f^{-1} 为私钥。

在 RSA 密码体制下,我们取 $n = pq$, $X = \mathbb{Z}_n$, $f(x) = x^b \bmod n$,且 $f^{-1}(x) = x^a \bmod n$,其中 $ab \equiv 1 (\bmod \phi(n))$。密码体制 5.3 又引入了一个特定的随机函数 G。实际上,G 由一个随机预言模型化,其定义参见 4.2.1 小节。

我们观察到密码体制 5.3 是非常有效的:相对于底层的基于 f 的公钥密码体制而言,它只需要添加很少的运算。实际中,函数 G 可以由安全 Hash 函数用很有效的方式给出,例如 SHA-1。密码体制 5.3 的主要缺点是数据扩张(data expansion):m 比特的明文加密成 $k + m$ 比特的密文。如果 f 基于 RSA 加密函数,那么为了使系统安全,需要取 $k \geqslant 1024$。

容易看出,在语义安全的公钥密码体制中必须有一定的数据扩张,因为要做到加密是随机的。然而,存在更有效的方案,仍可证明是安全的。其中最重要的一种是最优非对称加密填充(Optimal Asymmetric Encryption Padding),这将在本小节后面讨论(见密码体制 5.4)。我们从密码体制 5.3 开始讨论,是因为它概念简单、容易分析。

密码体制 5.3 在随机预言模型中是语义安全的一个直观的论据如下:为了判定关于明文 x 的任一信息,我们需要知道关于 $G(r)$ 的信息。假定 G 是一个随机预言,确定关于 $G(r)$ 值的任一信息的惟一方式是首先计算 $r = f^{-1}(y_1)$(仅计算关于 r 的部分信息是不够的;为了得到关于 $G(r)$ 值的任一信息,必须得到关于 r 的全部信息)。然而,如果 f 是单向的,那么对于不知道陷门 f^{-1} 的敌手而言,不能在合理的时间内算出 r。

前面的论据可能是相当令人信服的,但它并不是一个证明。如果我们要把这个论据转换为一个证明,需要描述一个从求函数 f 的逆问题到密文识别问题的一个约化。当 f 是随机的,像在密码体制 5.3 中那样,即使存在对给定密文的足够多的加密,求解问题 5.3 也是不可行的。

我们现在考虑一种比前面的 Turing 约化更一般的约化。我们假定存在一个算法 Distinguish 可以对两个明文 x_1, x_2 求解密文识别问题,那么我们如下修改这个算法就可以得到求 f 的逆的算法。算法 Distinguish 必是一个“完善”的算法;我们仅需要它以 $1/2 + \varepsilon$ 的概率给出正确的答案即可,其中 $\varepsilon > 0$(即,它比随机猜“1”或“2”更精确)。Distinguish 允许询问一个随机预言,因此它可以计算明文的加密。也就是说,我们假定它是一个选择明文攻击。

像上面提到的那样,我们将证明密码体制 5.3 在随机预言模型中是语义安全的。这个模型(在 4.2.1 小节中引入)的主要特性和约化,我们描述如下:

1. 假定 G 是一个随机预言,所以判断关于值 $G(r)$ 的任何信息的惟一方式就是用输入 r 调用函数 G。

2. 我们通过修改算法 Distinguish 来构造一个新算法 Invert,可以以不为 0 的概率对随机选择的元素 y 求逆(即,给定一个值 $y = f(x)$,其中 x 随机选择,那么算法 Invert 能够以特定概率找出 x)。

3. 算法 Invert 将用我们描述的一个特定函数 SimG 替换随机预言，它的所有输出为随机数。SimG 是随机预言的一个完善的模拟。

算法 Invert 如算法 5.14 所述。

算法 5.14　Invert(y)

external　f

global *RList*, *GList*, l

procedure SimG(r)

　$i \leftarrow 1$

　$found \leftarrow$ **false**

　while $i \leqslant l$ **and not** *found*

　　do { **if** $RList[i] = r$ **then** $found \leftarrow$ **true** **else** $i \leftarrow i + 1$

　　if *found*

　　then return ($GList[i]$)

　if $f(r) = y$

　　then { 随机选择 $j \in \{1,2\}$; $g \leftarrow y_2 \oplus x_j$

　　else 随机选择 g

　$l \leftarrow l + 1$

　$RList[l] \leftarrow r$

　$GList[l] \leftarrow g$

　return(g)

main

　$y_1 \leftarrow y$

　随机选择 y_2

　$l \leftarrow 0$

　在 Distinguish(x_1, x_2, (y_1, y_2))中插入代码

　for $i \leftarrow 1$ **to** l

　　do { **if** $f(RList[i] = y)$ **then return** ($RList[i]$)

　return("failure")

给定两个明文 x_1 和 x_2，Distinguish 以 $1/2 + \varepsilon$ 的概率求解密文识别问题。Invert 的输入是要求逆的 y；目的是输出 $f^{-1}(y)$。Invert 开始时构造密文(y_1, y_2)，其中 $y_1 = y$，y_2 随机选取。

Invert 对于密文(y_1, y_2)运行算法 Distinguish,试图判断它是 x_1 或者 x_2 的密文。Distinguish 将会在执行过程中不同的地方询问 SimG。SimG 的操作概括如下:

1. SimG 包含一个列表,记为 RList,记录了在 Distinguish 执行过程中询问的所有输入 r;所做响应的列表,记为 GList,记录了 SimG 的所有输出。

2. 如果一个输入 r 满足$f(r) = y$,那么 SimG 定义为使得(y_1, y_2)是 x_1 和 x_2 其中一个(随机选取)的有效加密。

3. 如果预言前面已经用输入 r 询问过,那么 SimG(r)已定义。

4. 其他情况下,SimG(r)的值随机选取。

可以看到,对于任一明文 x_0,(y_1, y_2)是 x_0 的一个有效加密,当且仅当

$$\text{SimG}(f^{-1}(y_1)) = y_2 \oplus x_0$$

特别地,SimG$(f^{-1}(y_1))$适当地定义,(y_1, y_2)可以是 x_1 或者 x_2 的有效加密。算法 SimG 的描述保证了(y_1, y_2)是 x_1 或者 x 的有效加密。

最后,算法 Distinguish 将会以结果"1"或者"2"终止,结果可能正确也可能不正确。到此,算法 Invert 检查列表 *RList* 看是否对其询问的某一 r 有 $y = f(r)$。一旦发现有这样的 r,我们可以立即结束算法 Invert,返回 r 值作为输出。没必要一直运行算法 Distinguish 以得到它的结论。然而,算法 5.14 的成功概率分析,将比理解算法更容易一些(读者可能会去验证对上面提到的算法 Invert 的修改并不会改变它的成功概率)。

我们现在计算算法 Invert 成功的概率的一个下界。我们通过检查算法 Distinguish 的成功概率来做到这一点。我们假定 Distinguish 在与一个随机预言相互作用的成功概率至少为 1/2 + ε。在算法 Invert 中,Distinguish 与模拟预言 SimG 相互作用。显然 SimG 对于任何输入与一个随机预言的输出是完全没有区别的,除了可能对于输入 $r = f^{-1}(y)$例外。然而,如果 $f(r) = y$ 且(y_1, y_2)是 x_1 或者 x_2 的有效加密,那么必然有 SimG(r) = $y_2 \oplus x_1$ 或者 SimG(r) = $y_2 \oplus x_2$。SimG 从这两个选择中随机选取。因此,它对于输入 $r = f^{-1}(y)$的输出也是与真正的随机预言没有区别的。于是,Distinguish 当与模拟预言 SimG 相互作用时,其成功概率为至少 1/2 + ε。

我们现在计算 Distinguish 的成功概率,以是否 $f^{-1}(y) \in RList$ 来考虑:

$$\begin{aligned}&\mathbf{Pr}[\text{Distinguish succeeds}] = \\ &\quad \mathbf{Pr}[\text{Distinguish succeeds} \mid f^{-1}(y) \in RList]\mathbf{Pr}[f^{-1}(y) \in RList] + \\ &\quad \mathbf{Pr}[\text{Distinguish succeeds} \mid f^{-1}(y) \notin RList]\mathbf{Pr}[f^{-1}(y) \notin RList]\end{aligned}$$

显然有

$$\mathbf{Pr}[\text{Distinguish succeeds} \mid f^{-1}(y) \notin RList] = 1/2$$

因为如果不知道 SimG$(f^{-1}(y))$的值,就无法区别由 x_1 还是由 x_2 加密的密文。现在,利用如下事实:

$$\mathbf{Pr}[\text{Distinguish succeeds} \mid f^{-1}(y) \in RList] \leqslant 1$$

我们得到如下关系:

$$\frac{1}{2} + \varepsilon \leqslant \mathbf{Pr}[\text{Distinguish succeeds}]$$
$$\leqslant \mathbf{Pr}[f^{-1}(y) \in RList] + \frac{1}{2}\mathbf{Pr}[f^{-1}(y) \notin RList]$$
$$\leqslant \mathbf{Pr}[f^{-1}(y) \in RList] + \frac{1}{2}$$

因此,可以得到

$$\mathbf{Pr}[f^{-1}(y) \in RList] \geqslant \varepsilon$$

由于

$$\mathbf{Pr}[\text{Invert succeeds}] = \mathbf{Pr}[f^{-1}(y) \in RList]$$

就可以推出

$$\mathbf{Pr}[\text{Invert succeeds}] \geqslant \varepsilon$$

可以直接考虑 Invert 相对于 Distinguish 的运行时间。假定 t_1 是 Distinguish 的运行时间,t_2 是对函数 f 求值需要的时间,用 q 记 Distinguish 所做的询问的次数,那么容易看到 Invert 的运行时间为 $t_1 + O(q^2 + qt_2)$。

我们以一个更有效的可证安全密码体制作为本小节结束。

密码体制 5.4　Optimal Asymmetric Encryption Padding

设 m, k 为正整数,且 $m < k$。令 $k_0 = k - m$。设 $\mathcal{F}$ 为一族陷门单向置换,使得对于所有 $f \in \mathcal{F}$,有 $f: \{0,1\}^k \to \{0,1\}^k$。设 $G: \{0,1\}^{k_0} \to \{0,1\}^m$,且设 $H: \{0,1\}^m \to \{0,1\}^{k_0}$ 为"随机"函数。定义 $\mathcal{P} = \{0,1\}^m$,$\mathcal{C} = \{0,1\}^k$,且定义:

$$\mathcal{K} = \{(f, f^{-1}, G, H): f \in \mathcal{F}\}$$

对于 $K = (f, f^{-1}, G, H)$,设 $r_0 \in \{0,1\}^m$ 随机选择,并定义:

$$e_K(x) = f(y_1 \| y_2)$$

其中,

$$y_1 = x \oplus G(r)$$
$$y_2 = r \oplus H(x \oplus G(r))$$

$x, y_1 \in \{0,1\}^m$, $y_2 \in \{0,1\}^{k_0}$,"‖"表示向量的串联。进一步,定义:

$$f^{-1}(y) = x_1 \| x_2$$

其中 $x_1 \in \{0,1\}^m$,且 $x_2 \in \{0,1\}^{k_0}$。那么定义:

$$r = x_2 \oplus H(x_1)$$

且

$$d_K(y) = G(r) \oplus x_1$$

函数 f, G 和 H 为公钥函数，f^{-1} 为私钥。

在密码体制 5.4 中，取 k_0 足够大就可以使得 2^{k_0} 是一个不可行的大运行时间；对大多数应用而言，取 $k_0 = 128$ 应该足够了。在密码体制 5.4 中一个密文的长度比明文长出 k_0 比特，所以与密码体制 5.3 相比数据扩张是相当少了。然而，密码体制 5.4 的安全性证明更加复杂。

密码体制 5.4 中的形容词“最优”是指信息的扩张。可发现任一明文有 2^{k_0} 种可能的加密。解决密文识别问题的一种方式就是直接计算两个明文中的一个(如 x_1)的所有可能的密文，然后看是否得到给定的密文。这个算法的复杂性为 2^{k_0}。因此容易看到，密码体制的信息扩张必定至少与求解密文识别问题算法的计算时间的以 2 为底的对数一样大。

5.10 注释与参考文献

公钥密码学的思想是由 Diffie 和 Hellman 在 1976 年的公开文献中引入的。尽管[68]是引用最多的文献，实际上会议论文[67]出现得稍早一点。RSA 密码体制是由 Rivest、Shamir 和 Adleman[181]发现的。对于公钥密码学的一个一般的综述，我们推荐 Diffie 的文章[66](现在看起来有点过时，但仍值得一读)。关于 RSA 的一个特别概览，参见 Boneh[31]。

Solovy-Strassen 检测首先在[202]中描述。Miller-Rabin 检测在[147]和[178]中给出。我们关于错误概率的讨论是由 Brassarg 和 Bratley[37]的观察所激发的(又见于[7])。现在关于 Miller-Rabin 算法的错误概率上界的最好估计可以在[58]中找到。

关于分解因子算法有许多来源。Lenstra[130]是关于分解算法的一个好的概览，Lenstra[132]是关于一般数论算法的一篇好文章。Bressoud[39]是一本关于分解因子和素性检测的基础教科书。推荐一本强调数论知识的密码学书是 Kobritz[120](本书中例 5.12 就是从这本书中找来的)。推荐对密码学研究非常有用的数论书是 Bach 和 Shallit[4]，von zur Gathen 和 Gerhard[90]，以及 Yan[219]。Lenstra[131]是关于数域的一篇较好的论文。RSA-155 的分解因子在[43]中有介绍。

5.7.2 小节和 5.9.1 小节中的材料是基于 Salomaa[184]中的处理(给定解密指数分解 n 是在[60]中证明的；有关 RSA 密文泄露部分信息的结论来自[96])。Wiener 攻击可以在[214]中找到；Boneh 和 Durfee 对攻击的加强，发表于[33]。

Rabin 密码体制在 Rabin[177]中描述。加密无歧义的可证明安全体制可以从 Williams[217]和 Kurosawa、Ito、Takeuhi[125]中找到。

由 RSA 密文泄露的部分信息在 Alexi、Chor、Goldreich、Schnorr[1]中研究。语义安全的概念由 Goldwasser 和 Micali[95]提出；Blum-Goldwasser 密码体制[29]是早期的概率公钥密码体制，是可证明语义安全的。

对于可证明安全密码体制的近期介绍，参见 Bellare[9]。随机预言模型首先由 Bellare 和 Rogaway 在[14]中描述；密码体制 5.3 就是在这篇文章中提出的。最优非对称加密填充在[15]

中提出,它被并入关于公钥密码学的 IEEE P1363 标准规范中。最近有一些讨论关于最优非对称加密填充和防范选择密文攻击密码体制的著作:Shoup[193]、Fujisaki、Okamoto、Poiatcheval、Stern[87]和 Boneh[32]。

练习 5.15 ~ 5.17 给出了一些协议失败的例子。这方面的一篇好文章是 Moore[151]。

练习

5.1　在算法 5.1 中,证明:

$$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$$

且有 $r_m = \gcd(a, b)$。

5.2　假定在算法 5.1 中 $a > b$,

(a) 证明对于所有的 $0 \leqslant i \leqslant m-2$,有 $r_i \geqslant 2r_{i+2}$。

(b) 证明 m 是 $O((\log a)^2)$。

(c) 证明 m 是 $O((\log b)^2)$。

5.3　利用扩展 Euclidean 算法计算如下乘法逆:

(a) $17^{-1} \bmod 101$

(b) $357^{-1} \bmod 1234$

(c) $3125^{-1} \bmod 9987$。

5.4　计算 gcd(57,93),并找出整数 s 和 t,使得 $57s + 93t = \gcd(57,93)$。

5.5　假定 $\chi: \mathbb{Z}_{105} \rightarrow \mathbb{Z}_3 \times \mathbb{Z}_5 \times \mathbb{Z}_7$ 定义为:

$$\chi(x) = (x \bmod 3, x \bmod 5, x \bmod 7)$$

求出函数 χ^{-1} 的显式公式,并用它计算 $\chi^{-1}(2,2,3)$。

5.6　求解如下的同余方程组:

$$x \equiv 12 (\bmod 25)$$
$$x \equiv 9 (\bmod 26)$$
$$x \equiv 23 (\bmod 27)$$

5.7　求解如下的同余方程组:

$$13x \equiv 4 (\bmod 99)$$
$$15x \equiv 56 (\bmod 101)$$

提示: 首先使用扩展 Euclidean 算法,然后应用中国剩余定理。

5.8　利用定理 5.8 找出模 97 的最小本原元素。

5.9　假定 $p = 2q + 1$,其中 p 和 q 为奇素数。进一步假定 $\alpha \in \mathbb{Z}_p^*$, $\alpha \not\equiv \pm 1 (\bmod p)$。证明 α 是一个模 p 的本原元素当且仅当 $\alpha^q \equiv -1 (\bmod p)$。

5.10 假定 $n = pq$,其中 p 和 q 为不同的奇素数,且 $ab \equiv 1(\mathrm{mod}\ (p-1)(q-1))$。RSA 加密运算是 $e(x) = x^b \bmod n$ 且解密运算为 $d(y) = y^a \bmod n$。我们已证明 $d(e(x)) = x$,对于 $x \in \mathbb{Z}_n^*$ 成立。现在证明这个断言对于任一 $x \in \mathbb{Z}_n$ 都成立。

提示: 利用 $x_1 \equiv x_2 (\mathrm{mod}\ pq)$ 当且仅当 $x_1 \equiv x_2 (\mathrm{mod}\ p)$ 且 $x_1 \equiv x_2 (\mathrm{mod}\ q)$。这可以由中国剩余定理得出。

5.11 对于 $n = pq$,其中 p 和 q 为不同的奇素数,定义

$$\lambda(n) = \frac{(p-1)(q-1)}{\gcd(p-1, q-1)}$$

假定我们对 RSA 密码体制做如下修改,限定 $ab \equiv 1(\mathrm{mod}\ \lambda(n))$:

(a) 证明加密和解密在修改后的密码体制中仍是逆运算。

(b) 如果 $p = 37, q = 79, b = 7$。计算在修改后的密码体制中以及原来的 RSA 密码体制中 a 的值。

5.12 表 5.3 和表 5.4 中给出了 RSA 的两个例子。你的任务是解密密文。系统的公开参数为 $n = 18\,923, b = 1261$(对于表 5.3)和 $n = 31\,313, b = 4913$(对于表 5.4)。这可以按如下步骤完成。首先,分解 n(因为 n 较小,所以容易做到)。然后,利用 $\phi(n)$ 计算指数 a,最后,解密密文。利用平方乘算法来计算模 n 指数。

为了将明文变为通常的英文文字,你需要知道英文字母是如何在 $\mathbb{Z}_n$ 中"编码"的。$\mathbb{Z}_n$ 中每一元素表示三个英文字母,参见如下例子:

$$\mathrm{DOG} \rightarrow 3 \times 26^2 + 14 \times 26 + 6 = 2398$$
$$\mathrm{CAT} \rightarrow 2 \times 26^2 + 0 \times 26 + 19 = 1371$$
$$\mathrm{ZZZ} \rightarrow 25 \times 26^2 + 25 \times 26 + 25 = 17\,575$$

你需要在你的程序中最后一步完成这个过程的逆。

第一段明文取自 Robertson Davies 的"The Diary of Samuel Marchbanks"(1947 年),第二段明文取自 Garrison Keillor 的"Lake Wobegon Days"(1985 年)。

表 5.3 RSA 密文

12423	11524	7243	7459	14303	6127	10964	16399
9272	13629	14407	18817	18830	13556	3159	16647
5300	13951	81	8986	8007	13167	10022	17213
2264	961	17459	4101	2999	14569	17183	15827
12963	9553	18194	3830	2664	13998	12501	18873
12161	13071	16900	7233	8270	17086	9792	14266
13236	5300	13951	8850	12129	6091	18110	3332
15061	12347	7817	7946	11675	13924	13892	18031
2620	6276	8500	201	8850	11178	16477	10161
3533	13842	7537	12259	18110	44	2364	15570
3460	9886	8687	4481	11231	7574	11383	17910
12867	13203	5102	4742	5053	15407	2976	9330
12192	56	2471	15334	841	13995	17592	13297
2430	9741	11675	424	6686	738	13874	8168
7913	6246	14301	1144	9056	15967	7328	13203
796	195	9872	16979	15404	14130	9105	2001

9792	14251	1498	11296	1105	4502	16979	1105
56	4118	11302	5988	3363	15827	6928	4191
4277	10617	874	13211	11821	3090	18110	44
2364	15570	3460	9886	9988	3798	1158	9872
16979	15404	6127	9872	3652	14838	7437	2540
1367	2512	14407	5053	1521	297	10935	17137
2186	9433	13293	7555	13618	13000	6490	5310
18676	4782	11374	446	4165	11634	3846	14611
2364	6789	11634	4493	4063	4576	17955	7965
11748	14616	11453	17666	925	56	4118	18031
9522	14838	7437	3880	11476	8305	5102	2999
18628	14326	9175	9061	650	18110	8720	15404
2951	722	15334	841	15610	2443	11056	2186

表 5.4　RSA 密文

6340	8309	14010	8936	27358	25023	16481	25809
23614	7135	24996	30590	27570	26486	30388	9395
27584	14999	4517	12146	29421	26439	1606	17881
25774	7647	23901	7372	25774	18436	12056	13547
7908	8635	2149	1908	22076	7372	8686	1304
4082	11803	5314	107	7359	22470	7372	22827
15698	30317	4685	14696	30388	8671	29956	15705
1417	26905	25809	28347	26277	7897	20240	21519
12437	1108	27106	18743	24144	10685	25234	30155
23005	8267	9917	7994	9694	2149	10042	27705
15930	29748	8635	23645	11738	24591	20240	27212
27486	9741	2149	29329	2149	5501	14015	30155
18154	22319	27705	20321	23254	13624	3249	5443
2149	16975	16087	14600	27705	19386	7325	26277
19554	23614	7553	4734	8091	23973	14015	107
3183	17347	25234	4595	21498	6360	19837	8463
6000	31280	29413	2066	369	23204	8425	7792
25973	4477	30989					

5.13　加速 RSA 解密过程的通常方式是采用中国剩余定理。假定 $d_K(y) = y^a \bmod n$ 和 $n = pq$。定义 $d_p = d \bmod (p-1)$和 $d_q = d \bmod (q-1)$;又令 $M_p = q^{-1} \bmod p$ 和 $M_q = p^{-1} \bmod q$。那么考虑如下算法:

算法 5.15　CRT-Optimized RSA Decryption(n, d_p, d_q, M_p, M_q, y)

$x_p \leftarrow y^{d_p} \bmod p$

$x_q \leftarrow y^{d_q} \bmod q$

$x \leftarrow M_p q x_p + M_q p x_q \bmod n$

reture (x)

算法 5.15 用模 p 和 q 的模指数运算代替了模 n 指数。如果 p 和 q 是 l 比特的整数,且模 l 比特整数的指数运算需要时间 cl^3,那么执行指数运算所需时间就由 c

$(2l)^3$ 减少为 $2cl^3$,节省了 75%。最后一步,引入中国剩余定理,如果 d_p, d_q, M_p, M_q 已经预先算好,需要时间$\mathcal{O}(l^2)$。

(a) 证明由算法 5.15 返回的值 x,实际上是 $y^d \bmod n$。

(b) 给定 $d = 1\,234\,577$, $p = 1511$ 和 $q = 2003$,计算 d_p, d_q, M_p 和 M_q。

(c) 给定如上的 d, p 和 q,用算法 5.15 解密密文 $y = 152\,702$。

5.14 证明 *RSA* 密码体制对于选择密文攻击是不安全的。特别地,给定密文 y,描述如何选择密文 $\hat{y} \neq y$,使得根据明文 $\hat{x} = d_K(\hat{y})$可以计算出 $x = d_K(y)$。

提示:使用 *RSA* 密码体制的乘法性质,即 $e_K(x_1)e(x_2) \bmod n = e_K(x_1 x_2 \bmod n)$。

5.15 本练习展示了什么是协议失败。这里提供一个例子描述了密码体制在粗心的方式下,敌手可以不用知道密钥就可以解密密文。为了保证"安全"通信,仅使用一个"安全"密码体制是不足够的。

假定 Bob 使用的 RSA 密码体制有一个很大的模数 n,不能在合理的时间内分解。假定 Alice 通过用 0 到 25 的整数表示字母的方式给 Bob 发消息(即,$A \leftrightarrow 0$, $B \leftrightarrow 1$,等等),然后加密模 26 的余数作为明文字母。

(a) 描述 Oscar 如何容易地解密用这种方式加密的信息。

(b) 通过解密如下的密文来描述这种攻击(这是用 *RSA* 密码体制加密的,$n = 18\,721$, $b = 25$),而不用分解 n:

365,0,4845,14 930,2608,2608,0

5.16 本练习描述了另一个在 *RSA* 密码体制中协议失败的例子(由 Simmons 提出),它称为"公模协议失败(common modulus protocol failure)"。假定 Bob 有一个模数为 n 的 RSA 密码体制,加密指数为 b_1,而 Charlie 有(相同的)模数为 n 的 RSA 密码体制,加密指数为 b_2。又假定 $\gcd(b_1, b_2) = 1$。现在考虑如下情形:Alice 要把同一密文 x 加密后发给 Bob 和 Charlie。因此她计算 $y_1 = x^{b_1} \bmod n$ 和 $y_2 = x^{b_2} \bmod n$,然后把 y_1 发送给 Bob,把 y_2 发送给 Charlie。假定 Oscar 截获了 y_1 和 y_2,然后执行算法 5.16 所述的运算。

算法 5.16 RSA Common Modulus Decryption(n, b_1, b_2, y_1, y_2)

$c_1 \leftarrow b_1^{-1} \bmod b_2$

$c_2 \leftarrow (c_1 b_1 - 1)/b_2$

$x_1 \leftarrow y_1^{c_1} (y_2^{c_2})^{-1} \bmod n$

return(x_1)

(a) 证明在算法 5.16 中计算得到的值实际上就是 Alice 的明文 x。因此,Oscar 可以解密出 Alice 发出的消息,即使所用的密码体制是"安全的"。

(b) 通过这个方法计算 x 来描述这种攻击，如果取 $n = 18\,721, b_1 = 43, b_2 = 7717, y_1 = 12\,677$ 和 $y_2 = 14\,702$。

5.17　我们再给出另一个 RSA 密码体制中协议失败的例子。假定网络中有三个使用者，比如 Bob、Bart 和 Bert，都有公开加密指数 $b = 3$。设他们的模数分别为 n_1, n_2, n_3，假定 n_1, n_2, n_3 两两互素。现在假定 Alice 加密了同一明文 x 发给 Bob、Bart 和 Bert。即是说，Alice 计算 $y_i \equiv x^3 \bmod n_i, 1 \leqslant i \leqslant 3$。描述 Oscar 如何由给定的 y_1, y_2, y_3 计算出 x，而无需分解任何一个模数。

5.18　如果 $e_K(x) = x$，则明文 x 称为是不动的(fixed)。证明，对于 RSA 密码体制，不动明文 $x \in \mathbb{Z}_n^*$ 的个数等于

$$\gcd(b-1, p-1) \times \gcd(b-1, q-1)$$

提示：考虑如下的同余方程组：

$$e_K(x) \equiv x \pmod{p}$$
$$e_K(x) \equiv x \pmod{q}$$

5.19　假定 **A** 是确定性算法，以 RSA 模数 n、加密指数 b 和一个密文 y 作为输入。**A** 或者解密 y，或者失败。假定 $\varepsilon(n-1)$ 是 **A** 可以成功地解密的非零密文的数目。描述如何利用 **A** 作为预言器来构造一个具有成功率 ε 的 Las Vegas 算法。

5.20　写一个程序，利用 5.4 节中所描述的四个性质来计算 Jacobi 符号。程序中除了除以 2 的幂次外，不做任何因子分解。计算如下 Jacobi 符号来测试你的程序：

$$\left(\frac{610}{987}\right), \left(\frac{20\,964}{1987}\right), \left(\frac{1\,234\,567}{11\,111\,111}\right)$$

5.21　对于 $n = 837, 851, 1189$，找出基 b 的个数，使得 n 是相对于 b 的 Euler 伪素数。

5.22　本练习的目的是要证明 Solovay-Strassen 素性检测算法的错误概率至多为 1/2。设 $\mathbb{Z}_n^*$ 表示模 n 可逆元组成的群。定义：

$$G(n) = \left\{a : a \in \mathbb{Z}_n^*, \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}\right\}$$

(a) 证明 $G(n)$ 是 $\mathbb{Z}_n^*$ 的一个子群。因此，由 Lagrange 定理，如果 $G(n) \neq \mathbb{Z}_n^*$，那么

$$|G(n)| \leqslant \frac{|\mathbb{Z}_n^*|}{2} \leqslant \frac{n-1}{2}$$

(b) 假定 $n = p^k q$，其中 p 和 q 为奇数，p 是素数，$k \geqslant 2$，且 $\gcd(p, q) = 1$。设 $a = 1 + p^{k-1} q$。证明：

$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n}$$

提示：利用二项式定理来计算 $a^{(n-1)/2}$。

(c) 假定 $n = p_1 \cdots p_s$，其中 p_i 为互不相同的奇素数。假定 $a \equiv u \pmod{p_1}$ 且 $a \equiv 1 \pmod{p_2 p_3 \cdots p_s}$，其中 u 是一个模 p_1 的二次非剩余(注意，由中国剩余定理，这

样的 u 一定存在)。证明:

$$\left(\frac{a}{n}\right) \equiv -1 \pmod{n}$$

但

$$a^{(n-1)/2} \equiv 1 \pmod{p_2 p_3 \cdots p_s}$$

所以

$$a^{(n-1)/2} \not\equiv -1 \pmod{n}$$

(d) 如果 n 是奇数且为合数,证明 $|G(n)| \leqslant (n-1)/2$。

(e) 综合上面结果,证明 Solovay-Strassen 素性检测算法的错误概率至多为 1/2。

5.23 假定我们有一个失败概率为 ϵ 的 Las Vegas 算法。

(a) 证明在第 n 次尝试时首次成功的概率为 $p_n = \epsilon^{n-1}(1-\epsilon)$。

(b) 达到成功的平均(期望)次数是:

$$\sum_{n=1}^{\infty} (n \times p_n)$$

证明这个平均值等于 $1/(1-\epsilon)$。

(c) 设 δ 为一个小于 1 的正实数。证明为了使失败的概率减少为至多 δ,所需要的迭代次数为:

$$\left\lfloor \frac{\log_2 \delta}{\log_2 \epsilon} \right\rfloor$$

5.24 假定在这个题目中 p 是一个奇素数,且 $\gcd(a,p)=1$。

(a) 假定 $i \geqslant 2$ 且 $b^2 \equiv a \pmod{p^{i-1}}$。证明存在惟一的 $x \in \mathbb{Z}_{p^i}$,使得 $x^2 \equiv a \pmod{p^i}$,且 $x \equiv b \pmod{p^{i-1}}$。描述如何有效地计算出这个 x。

(b) 对如下情况举例说明你的方法:从同余方程 $6^2 = 17 \pmod{19}$ 开始,找出 17 模 19^2 和模 19^3 的平方根。

(c) 对于所有的 $i \geqslant 1$,证明同余方程 $x^2 \equiv a \pmod{p^i}$ 的解的数目为 0 或者 2。

5.25 选择不同的界 B,利用 $p-1$ 方法尝试分解 262 063 和 9 420 457。在每种情形下需要多大的界才能成功?

5.26 用 Pollard ρ 算法分解 262 063,9 420 457 和 181 937 053,函数 f 定义为 $f(x) = x^2 + 1$。分解每一个整数需要多少次迭代?

5.27 假定我们要用随机平方算法来分解整数 $n = 256\,961$。使用因子基

$$\{-1,2,3,5,7,11,13,17,19,23,29,31\}$$

对于 $z = 500,501,\cdots$ 测试整数 $z^2 \bmod n$,直到找到一个形如 $x^2 \equiv y^2 \pmod{n}$ 的同余方程,从而可以得到 n 的分解。

5.28 在随机平方算法中,需要测试一个正整数 $w \leqslant n-1$,看它能否在因子基 $\mathcal{B} = \{p_1, \cdots, p_B\}$ 上完全分解,其中 $\mathcal{B}$ 是最小的 B 个素数的集合。已有 $p_B = m \approx 2^s$,且 $n \approx 2^r$。

(a) 证明这可以通过对至多有 r 比特的整数至多 $B+r$ 次除以至多有 s 比特的整数

来完成。

(b) 假定 $r < m$。证明这个测试的复杂度为$\mathcal{O}(rsm)$。

5.29　在本练习中,我们描述在 RSA 密码体制的参数生成过程中,应注意保证 $q - p$ 不是很小,其中 $n = pq$,且 $q > p$。

(a) 假定 $q - p = 2d > 0$,且 $n = pq$。证明 $n + d^2$ 是一个完全平方数。

(b) 给定一个整数 n 是两个奇素数的乘积,且给定一个小的正整数 d 使得 $n + d^2$ 是一个完全平方数。描述如何利用这些信息来分解 n。

(c) 使用这个技巧来分解整数 $n = 2\,189\,284\,635\,403\,183$。

5.30　假定 Bob 由于粗心泄露了他的加密指数 $a = 14\,039$,在这个 RSA 密码体制中公开密钥为 $n = 36\,581$, $b = 4679$。对于给定的信息,实现一个随机算法来分解 n。用“随机”选择 $w = 9983$ 和 $w = 13\,461$ 来测试你的算法。写出所有的计算。

5.31　如果 $q_1, \cdots, q_m$ 是在用 Euclidean 算法求 $\gcd(r_0, r_1)$过程中得到的商的序列,证明连分数$[q_1, \cdots, q_m] = r_0/r_1$。

5.32　假定在 RSA 密码体制中公开密钥为 $n = 317\,940\,011$, $b = 77\,537\,081$。使用 Wiener 算法,试分解 n。

5.33　考虑 Rabin 密码体制的修改, $e_K(x) = x(x + B) \bmod n$,其中 $B \in \mathbb{Z}_n$ 是公钥的一部分。假定 $p = 199$, $q = 211$, $n = pq$ 且 $B = 1357$,执行如下计算。

(a) 计算加密 $y = e_K(32\,767)$。

(b) 求出这个给定密文 y 的四个可能解密。

5.34　证明与函数 *half* 和 *parity* 相关的等式(5.3)和(5.4)。

5.35　证明密码体制 5.3 对于选择密文攻击不是语义安全的。给定 x_1, x_2,一个密文(y_1, y_2),它是 x_i ($i = 1$ 或 2)的加密,给定对于密码体制 5.3 的一个解密预言 Decrypt,描述一个算法判断 $i = 1$ 或者 $i = 2$。你可以对于除密文(y_1, y_2)外的任一输入调用算法 Decrypt,且它会输出对应的明文。

第6章　基于离散对数问题的公钥密码体制

本章主要讨论基于离散对数问题的公钥密码体制。第一个同时也是最著名的公钥密码体制是 ElGamal 密码体制。我们后面将要讨论的密码协议，多数以离散对数为基础，我们将花大量的时间讨论这个重要问题。本章的后几节给出了另外几类基于有限域和椭圆曲线的 ElGamal 型的密码体制。

6.1　ElGamal 密码体制

ElGamal 密码体制基于离散对数问题。我们首先在有限乘法群$(G,\cdot)$中描述这个问题。对于一个 n 阶元素$\alpha\in G$，定义：

$$\langle\alpha\rangle=\{\alpha^i:0\leqslant i\leqslant n-1\}$$

容易看出，$\langle\alpha\rangle$是 G 的一个子群，$\langle\alpha\rangle$是一个 n 阶循环群。

经常使用的一个例子是，取 G 为有限域$\mathbb{Z}_p$（p 为素数）的乘法群，α 为模 p 的本原元。这时有 $n=|\langle\alpha\rangle|=p-1$。另一个经常用到的情况是，取 α 为乘法群$\mathbb{Z}_p^*$ 的一个素数 q 阶的元素（其中 p 为素数，并且 $p-1\equiv 0(\bmod\ q)$）。在$\mathbb{Z}_p^*$ 中，这种元素 α 可以由本原元的$(p-1)/q$ 次幂得到。

我们在群$(G,\cdot)$的子群$\langle\alpha\rangle$中定义离散对数问题。

问题 6.1　离散对数(Discrete Logarithm)

实例： 乘法群$(G,\cdot)$，一个 n 阶元素$\alpha\in G$ 和元素$\beta\in\langle\alpha\rangle$

问题： 找到惟一的整数 a，$0\leqslant a\leqslant n-1$，满足：

$$\alpha^a=\beta$$

我们将这个整数 a 记为 $\log_\alpha\beta$。

在密码中主要应用离散对数问题的如下性质：求解离散对数是困难的，而其逆运算指数运算可以应用平方-乘的方法(算法 5.5)有效地计算出来。换句话说，在相应的群 G 中，指数函数是单向函数。

ElGamal 提出了一个基于$(\mathbb{Z}_p^*,\cdot)$上离散对数问题的公钥密码体制。这个体制表述为密码体制 6.1。

密码体制 6.1 $\mathbb{Z}_p^*$ 上的 ElGamal 公钥密码体制

设 p 是一个素数,使得$(\mathbb{Z}_p^*,\cdot)$上的离散对数问题是难处理的,令 $\alpha\in\mathbb{Z}_p^*$ 是一个本原元。令$\mathcal{P}=\mathbb{Z}_p^*$,$\mathcal{C}=\mathbb{Z}_p^*\times\mathbb{Z}_p^*$,定义

$$\mathcal{K}=\{(p,\alpha,a,\beta):\beta\equiv\alpha^a(\bmod\ p)\}$$

p,α,β 是公钥,a 是私钥。

对于 $K=(p,\alpha,a,\beta)$,以及一个(秘密)的随机数 $k\in\mathbb{Z}_{p-1}$,定义:

$$e_K(x,k)=(y_1,y_2)$$

其中

$$y_1=\alpha^k\bmod\ p$$

而

$$y_2=x\,\beta^k\bmod\ p$$

对 $y_1,y_2\in\mathbb{Z}_p^*$,定义

$$d_K(y_1,y_2)=y_2(y_1^a)^{-1}\bmod\ p$$

在 ElGamal 密码体制中,加密运算是随机的,因为密文既依赖于明文 x,又依赖于 Alice 选择的随机数 k。所以,对于同一个明文,会有许多($p-1$ 个)可能的密文。

ElGamal 密码体制的工作方式可以非正式地描述如下:明文 x 通过乘以β^k"伪装"起来,产生 y_2。值 α^k 也作为密文的一部分传送。Bob 知道密钥 a,可以从 α^k 计算出β^k。最后用 y_2 除以 β^k除去伪装,得到 x。

下面的这个简单例子能够说明在 ElGamal 密码体制中所进行的计算。

例 6.1 设 $p=2579,\alpha=2$。α 是模p 的本原元。令 $a=765$,所以

$$\beta=2^{765}\bmod 2579=949$$

假设 Alice 现在想要传送消息 $x=1299$ 给 Bob。比如 $k=853$ 是她选择的随机数,那么她计算:

$$\begin{aligned}y_1&=2^{853}\bmod 2579\\&=435\end{aligned}$$

和

$$\begin{aligned}y_2&=1299\times 949^{853}\bmod 2579\\&=2396\end{aligned}$$

当 Bob 收到密文 $y=(435,2396)$后,计算:

$$\begin{aligned}x&=2396\times(435^{765})^{-1}\bmod 2579\\&=1299\end{aligned}$$

这正是 Alice 加密过的明文。

很显然，如果 Oscar 可以计算 $a = \log_\alpha \beta$，那么 ElGamal 密码体制就是不安全的，因为那时 Oscar 可以像 Bob 一样解密密文。因此，ElGamal 密码体制安全的一个必要条件就是，$\mathbb{Z}_p^*$ 上的离散对数问题是难处理的。一般正是这么认为的，当然 p 要仔细地选取，且 α 是模 p 的本原元。特别是对于这种形式的离散对数问题，不存在已知的多项式时间算法。为了防止已知的攻击，p 应该至少有 300 个十进制数位，$p-1$ 应该具有至少一个较“大”的素数因子。

6.2　离散对数问题的算法

本节中，我们假设$(G,\cdot)$是一个乘法群，$\alpha \in G$ 是一个 n 阶元素。因而离散对数问题可以表达成下面的形式：给定 $\beta \in \langle \alpha \rangle$，找出惟一的元素 a，$0 \leqslant a \leqslant n-1$，使得 $\alpha^a = \beta$。

我们从分析一些基本的算法开始，这些算法可以用于解离散对数问题。我们在分析中假定，群 G 中两个元素的乘积需要常数(即 $O(1)$)时间。

首先，我们注意到离散对数问题可以通过 $O(n)$时间和 $O(1)$空间穷举搜索解决。只要计算 $\alpha, \alpha^2, \alpha^3, \cdots$，直到发现 $\beta = \alpha^a$(上述序列中每一项 α^i 通过前一项 α^{i-1} 乘以 α 得到，因此总的时间需要 $O(n)$)。

另外一种方法是，预先计算出所有可能的值 α^i，并对有序对(i,α^i)以第二个坐标排序列表，然后，给定 β，我们对存储的列表施行一个二分搜索，直到找到 a 使得 $\alpha^a = \beta$。这需要 $O(n)$时间预先计算 α 的 n 次幂，$O(n\log n)$时间对 n 个元素排序(如果使用有效的排序算法，如 Quicksort 算法，可以用 $O(n\log n)$步完成对 n 个元素的排序)。如果我们像通常分析算法那样，忽略对数因子，预先计算的时间就是 $O(n)$。n 个有序元素列表的二分检索时间为 $O(\log n)$。如果我们(再次)忽略对数因子项，我们看到，离散对数问题可以用 $O(1)$时间、$O(n)$步预先计算和 $O(n)$存储空间解决。

6.2.1　Shanks 算法

我们描述的第一个非平凡的时间-存储平衡算法是 Shanks 算法。Shanks 算法表述为算法 6.1。

算法 6.1　Shanks(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m-1$
   do 计算 α^{mj}
3. 对 m 个有序对(j, α^{mj})关于第二个坐标排序，得到一个列表 L_1
4. **for** $i \leftarrow 0$ **to** $m-1$

do 计算 $\beta\alpha^{-i}$

5.对 m 个有序对$(i,\beta\alpha^{-i})$关于第二个坐标排序,得到一个列表 L_2

6.找到对$(j,y)\in L_1$ 和$(i,y)\in L_2$(即,找到两个具有相同第二坐标的对)

7.$\log_\alpha \beta \leftarrow (mj+i) \bmod n$

这里需要做些解释。首先,如果需要的话,第 2、3 步可以预先计算(然而,这并不影响渐近的运行时间)。其次,可以看到如果$(j,y)\in L_1$ 和$(i,y)\in L_2$,则

$$\alpha^{mj}=y=\beta\alpha^{-i}$$

因此,

$$\alpha^{mj+i}=\beta$$

反过来,对任意的 $\beta\in\langle\alpha\rangle$,有 $0\leqslant\log_\alpha\beta\leqslant n-1$。我们用 m 去除 $\log_\alpha\beta$,就可以将 $\log_\alpha\beta$ 表示为形式

$$\log_\alpha\beta=mj+i$$

其中 $0\leqslant j,i\leqslant m-1$。$j\leqslant m-1$ 可以从下面得出:

$$\log_\alpha\beta\leqslant n-1\leqslant m^2-1=m(m-1)+m-1$$

因而第 6 步将会成功(但是,如果恰巧 $\beta\notin\langle\alpha\rangle$,第 6 步就不会成功)。

很容易实现这个算法,使其运行时间为 $O(m)$,存储空间为 $O(m)$(忽略对数因子)。这里是几个细节:步骤 2 可以先计算 α^m,然后依次乘以 α^m。这步总的花费时间为 $O(m)$。同样地,步骤 4 花费的时间为 $O(m)$。步骤 3 和 5 利用有效的排序算法,花费时间为 $O(m\log m)$。最后,做一个对两个表 L_1 和 L_2 同时进行的遍历,完成步骤 6,这需要的时间为 $O(m)$。

这里举一个小例子说明 Shanks 算法。

例 6.2 假设我们要在$(\mathbb{Z}_{809}^*,\cdot)$中求出 $\log_3 525$。注意,809 是素数,α 是$\mathbb{Z}_{809}^*$中本原元,这时,$\alpha=3$,$n=808$,$\beta=525$ 和 $m=\lceil\sqrt{808}\rceil=29$ 。则

$$\alpha^{29} \bmod 809=99$$

首先,对于 $0\leqslant j\leqslant 28$ 计算有序对$(j,99^j \bmod 809)$。得到下面的列表:

(0,1)	(1,99)	(2,93)	(3,308)	(4,559)
(5,329)	(6,211)	(7,664)	(8,207)	(9,268)
(10,644)	(11,654)	(12,26)	(13,147)	(14,800)
(15,727)	(16,781)	(17,464)	(18,632)	(19,275)
(20,528)	(21,496)	(22,564)	(23,15)	(24,676)
(25,586)	(26,575)	(27,295)	(28,81)	

这些序对排序后产生 L_1。

第二个列表包括序对$(i,525\times(3^i)^{-1}\bmod 809)$,$0\leqslant i\leqslant 28$。如下所示:

$$\begin{array}{cccc}(0,525) & (1,175) & (2,328) & (3,379)\\ (5,132) & (6,44) & (7,554) & (8,724)\\ (10,440) & (11,686) & (12,768) & (13,256)\\ (15,388) & (16,399) & (17,133) & (18,314)\\ (20,754) & (21,521) & (22,713) & (23,777)\\ (25,356) & (26,658) & (27,489) & (28,163)\end{array}$$

排序后得到 L_2。

现在同时遍历两个列表,发现(10,644)在 L_1 中,(19,644)在 L_2 中。所以可以进行计算

$$\begin{aligned}\log_3 525 &= (29\times 10+19)\bmod 808\\ &= 309\end{aligned}$$

这个结果可以通过验证 $3^{309}\equiv 525(\bmod 809)$得到检验。

6.2.2　Pollard ρ 离散对数算法

前面 5.6.2 小节中我们曾介绍了因子分解的 Pollard ρ 算法。现在介绍对应的解离散对数问题的算法。与前面一样,假设$(G,\cdot)$是一个群,$\alpha\in G$ 是一个 n 阶元素,我们要计算元素 $\beta\in\langle\alpha\rangle$的离散对数。由于$\langle\alpha\rangle$是 n 阶循环群,可以把 $\log_\alpha\beta$ 看做$\mathbb{Z}_n$ 中的元素。

与因子分解的 ρ 算法一样,通过迭代一个貌似随机的函数 f,我们构造一个序列 $x_1,x_2,\cdots$。一旦在序列中得到两个元素 x_i 和 x_j,满足 $x_i=x_j$,这里 $i<j$,我们就有希望计算出 $\log_\alpha\beta$。为了能够节省时间和空间,我们寻求一种与因子分解算法一样的碰撞 $x_i=x_{2i}$。

假设 $S_1\cup S_2\cup S_3$ 是群 G 的一个划分,它们的元素个数大致相同。定义函数 $f:\langle\alpha\rangle\times\mathbb{Z}_n\times\mathbb{Z}_n\to\langle\alpha\rangle\times\mathbb{Z}_n\times\mathbb{Z}_n$ 如下:

$$f(x,a,b)=\begin{cases}(\beta x,a,b+1) & x\in S_1\\ (x^2,2a,2b) & x\in S_2\\ (\alpha x,a+1,b) & x\in S_3\end{cases}$$

而且,我们构造的每个三元组(x,a,b)要满足 $x=\alpha^a\beta^b$。选择初始三元组满足这个性质,比如(1,0,0)。

可以看出,如果(x,a,b)满足这个性质,$f(x,a,b)$也满足这个性质。因此我们定义:

$$(x_i,a_i,b_i)=\begin{cases}(1,0,0) & i=0\\ f(x_{i-1},a_{i-1},b_{i-1}) & i\geqslant 1\end{cases}$$

比较三元组(x_{2i},a_{2i},b_{2i})和(x_i,a_i,b_i),直到发现 $x_{2i}=x_i$,$i\geqslant 1$。这时有:

$$\alpha^{a_{2i}}\beta^{b_{2i}}=\alpha^{a_i}\beta^{b_i}$$

如果记 $c=\log_\alpha\beta$,则下面等式成立:

$$\alpha^{a_{2i}+cb_{2i}} = \alpha^{a_i+cb_i}$$

由于 α 是 n 阶元素,就有:

$$a_{2i} + cb_{2i} \equiv a_i + cb_i (\text{mod } n)$$

改写后有:

$$c(b_{2i} - b_i) \equiv a_i - a_{2i} (\text{mod } n)$$

如果 $\gcd(b_{2i} - b_i, n) = 1$,就可以通过如下式子解出 c:

$$c = (a_i - a_{2i})(b_{2i} - b_i)^{-1} \text{mod } n$$

我们用一个例子说明上述算法的应用。注意,我们必须保证 $1 \notin S_2$(因为 $1 \in S_2$ 时,对任意的 $i \geqslant 0$ 都有 $x_i = (1,0,0)$)。

例 6.3 整数 $p = 809$ 是素数,可以验证 $\alpha = 89$ 在 $\mathbb{Z}_{809}^*$ 中是 $n = 101$ 阶的元素。元素 $\beta = 618$ 在子群 $\langle\alpha\rangle$ 中;我们计算 $\log_\alpha \beta$。

如下定义 S_1, S_2, S_3:

$$S_1 = \{x \in \mathbb{Z}_{809} : x \equiv 1 (\text{mod } 3)\}$$
$$S_2 = \{x \in \mathbb{Z}_{809} : x \equiv 0 (\text{mod } 3)\}$$
$$S_3 = \{x \in \mathbb{Z}_{809} : x \equiv 2 (\text{mod } 3)\}$$

对于 $i = 1,2,\ldots$,得到三元组 (x_{2i}, a_{2i}, b_{2i}) 和 (x_i, a_i, b_i) 的值如下:

i	(x_i, a_i, b_i)	(x_{2i}, a_{2i}, b_{2i})
1	(618,0,1)	(76,0,2)
2	(76,0,2)	(113,0,4)
3	(46,0,3)	(488,1,5)
4	(113,0,4)	(605,4,10)
5	(349,1,4)	(422,5,11)
6	(488,1,5)	(683,7,11)
7	(555,2,5)	(451,8,12)
8	(605,4,10)	(344,9,13)
9	(451,5,10)	(112,11,13)
10	(422,5,11)	(422,11,15)

上表中第一个碰撞是 $x_{10} = x_{20} = 422$。要解的方程是:

$$c = (11 - 5)(11 - 15)^{-1} \text{mod } 101 = (6 \times 25) \text{mod } 101 = 49$$

所以,在 $\mathbb{Z}_{809}^*$ 中 $\log_{89} 618 = 49$。

离散对数的 Pollard ρ 算法由算法 6.2 给出。该算法中继续假设 $\alpha \in G$ 具有阶数 n，并且 $\beta \in \langle \alpha \rangle$。

算法 6.2　离散对数算法 Pollard ρ (G, n, α, β)

Procedure $f(x, a, b)$
if $x \in S_1$
　then $f \leftarrow (\beta \cdot x, a, (b+1) \bmod n)$
　else if $x \in S_2$
　then $f \leftarrow (x^2, 2a \bmod n, 2b \bmod n)$
　else $f \leftarrow (\alpha \cdot x, (a+1) \bmod n, b)$
return(f)
main
　定义划分 $G = S_1 \cup S_2 \cup S_3$
　$(x, a, b) \leftarrow f(1,0,0)$
　$(x', a', b') \leftarrow f(x, a, b)$
　while $x \neq x'$

$$\textbf{do}\begin{cases}(x, a, b) \leftarrow f(x, a, b)\\ (x', a', b') \leftarrow f(x', a', b')\\ (x', a', b') \leftarrow f(x', a', b')\end{cases}$$

if $\gcd(b' - b, n) \neq 1$
　then return("failure")
　else return$((a - a')(b' - b)^{-1} \bmod n)$

当 $\gcd(b' - b, n) > 1$ 时，算法 6.2 会停止并输出"failure"，这种情形并不悲观，如果 $\gcd(b' - b, n) = d$，容易证明同余方程方程 $c(b' - b) \equiv a - a' \pmod n$恰有 d 个解。假如 d 不是很大的话，可以相对直接地算出同余方程的 d 个解检验哪个解是正确的。

分析算法 6.2 的方式与 Pollard ρ 的分解算法类似。在函数 f 的随机性的合理假设下，可以期望在 n 阶循环群中用算法的 $O(\sqrt{n})$次迭代计算离散对数。

6.2.3　Pohlig-Hellman 算法

我们要研究的下一个算法是 Pohlig-Hellman 算法。假设

$$n = \prod_{i=1}^{k} p_i^{c_i}$$

其中 p_i 是不同的素数。值 $a = \log_\alpha \beta$ 是模 n(惟一)确定的。首先我们知道，如果能够对于每个 $i, 1 \leqslant i \leqslant k$，计算出 $a \bmod p_i^{c_i}$，就可以利用中国剩余定理计算出 $a \bmod n$。所以假设 q 是素数，

$$n \equiv 0 (\mathrm{mod}\ q^c)$$

且

$$n \not\equiv 0 (\mathrm{mod}\ q^{c+1})$$

我们说明如何计算

$$x = a \mod q^c$$

其中 $0 \leqslant x \leqslant q^c - 1$。我们把 x 以 q 的幂表示为:

$$x = \sum_{i=0}^{c-1} a_i q^i$$

其中 $0 \leqslant i \leqslant c-1, 0 \leqslant a_i \leqslant q-1$。还有,我们可以把 a 表示为:

$$a = x + sq^c$$

s 是某个整数。因而就有:

$$a = \sum_{i=0}^{c-1} a_i q^i + sq^c$$

算法的第一步是计算 a_0。算法中主要利用了等式(6.1):

$$\beta^{n/q} = \alpha^{a_0 n/q} \tag{6.1}$$

其证明如下:

$$\begin{aligned}
\beta^{n/q} &= (\alpha^a)^{n/q} \\
&= (\alpha^{a_0 + a_1 q + \cdots + a_{c-1} q^{c-1} + sq^c})^{n/q} \\
&= (\alpha^{a_0 + Kq})^{n/q} \qquad \text{其中 } K \text{ 是整数} \\
&= \alpha^{a_0 n/q} \alpha^{Kn} \\
&= \alpha^{a_0 n/q}
\end{aligned}$$

有了等式(6.1),确定 a_0 就很简单了。比如,可以计算:

$$\gamma = \alpha^{n/q}, \gamma^2, \cdots$$

直到对于某个 $i \leqslant q-1$,使得

$$\gamma^i = \beta^{n/q}$$

这时,$a_0 = i$。

如果 $c = 1$,事情已经解决。否则 $c > 1$,继续确定 $a_1, \cdots, a_{c-1}$,这可以与计算 a_0 类似的方式进行。记 $\beta_0 = \beta$,对于 $1 \leqslant j \leqslant c-1$,定义:

$$\beta_j = \beta \alpha^{-(a_0 + a_1 q + \cdots + a_{j-1} q^{j-1})}$$

把等式(6.1)推广为:

$$\beta_j^{n/q^{j+1}} = \alpha^{a_j n/q} \tag{6.2}$$

可以看到，在 $j=0$ 时，等式(6.2)归结为等式(6.1)。

等式(6.2)的证明类似于等式(6.1)的证明：

$$\begin{aligned}\beta_j^{n/q^{j+1}} &= \left(\alpha^{a-(a_0+a_1q+\cdots+a_{j-1}q^{j-1})}\right)^{n/q^{j+1}}\\ &= \left(\alpha^{a_jq^j+\cdots+a_{c-1}q^{c-1}+sq^c}\right)^{n/q^{j+1}}\\ &= \left(\alpha^{a_jq^j+K_jq^{j+1}}\right)^{n/q^{j+1}}\\ &= \alpha^{a_jn/q}\alpha^{K_jn}\\ &= \alpha^{a_jn/q}\end{aligned}$$

其中 K_j 是整数。所以，给定 β_j，能够从(6.2)直接计算 a_j。

为了使算法完整，我们可以看到，当 a_j 知道的情况下，β_{j+1}能够由 β_j 通过简单的递归关系计算出：

$$\beta_{j+1}=\beta_j\alpha^{-a_jq^j} \tag{6.3}$$

所以，交替利用等式(6.2)和(6.3)，可以计算 $a_0,\beta_1,a_1,\beta_2,\cdots,\beta_{c-1},a_{c-1}$。

Pohlig-Helman 算法用伪码的形式表述为算法 6.3。我们总结一下这个算法的运算，a 是乘法群 G 的一个 n 阶元素，q 是素数，

$$n\equiv 0(\bmod\ q^c)$$

且

$$n\not\equiv 0(\bmod\ q^{c+1})$$

算法计算出了 $a_0,\cdots,a_{c-1}$，其中，

$$\log_a\beta \bmod q^c = \sum_{i=0}^{c-1}a_iq^i$$

算法 6.3　Pohlig-Hellman(G,n,α,β,q,c)

$j\leftarrow 0$

$\beta_j\leftarrow\beta$

while $j\leqslant c-1$

do
- $\delta\leftarrow\beta_j^{n/q^{j+1}}$
- 找到满足 $\delta=\alpha^{in/q}$ 的 i
- $a_j\leftarrow i$
- $\beta_{j+1}\leftarrow\beta_j\alpha^{-a_jq^i}$
- $j\leftarrow j+1$

return$(a_0,\cdots,a_{c-1})$

下面用一个小例子对 Pohlig-Hellman 算法加以说明。

例 6.4 设 $p=29,\alpha=2,p$ 是素数,α 是模 p 的本原元素,我们有:

$$u=p-1=28=2^2 7^1$$

设 $\beta=18$,我们要计算 $a=\log_2 18$。我们首先计算 $a \bmod 4$,随后计算 $a \bmod 7$。

应用算法 6.3,先选择 $q=2$ 和 $c=2$。得到 $a_0=1$ 和 $a_1=1$。所以,$a\equiv 3(\bmod 4)$。

其次对于 $q=7,c=1$,应用算法 6.3。算出 $a_0=4$,所以,$a\equiv 4(\bmod 7)$。

最终应用中国剩余定理解方程组:

$$a\equiv 3(\bmod 4)$$
$$a\equiv 4(\bmod 7)$$

得到 $a\equiv 11(\bmod 28)$。即我们算得在 $\mathbb{Z}_{29}^*$ 中 $\log_2 18=11$。

我们考察算法 6.3 的复杂性。不难看出,直接实现算法的时间为 $O(cq)$。这可以改进。注意,每次计算满足 $\delta=\alpha^{in/q}$ 的值 i,可以视为解一个特殊的离散对数问题。特别地,$\delta=\alpha^{in/q}$ 当且仅当

$$i=\log_{\alpha^{n/q}} \delta$$

元素 $\alpha^{n/q}$ 的阶数是 q,所以每个 i 可以用 $O(\sqrt{q})$ 时间(利用 Shanks 算法)计算。这样,算法 6.3 的复杂性可以降到 $O(c\sqrt{q})$。

6.2.4 指数演算法

前面三个小节介绍的算法可以应用到任何群中。这一小节我们介绍指数演算法。这种方法非常特殊:它用于计算 $\mathbb{Z}_p^*$ 的离散对数这种特殊的情形中,其中 p 是素数,α 是模 p 的本原元素。在这种特殊的情形中,指数演算法比前面考虑的算法要快。

指数演算法计算离散对数,主要是模仿了许多因子分解的算法。这个方法使用了一个因子基。因子基是由一些"小"素数组成的集合 $\mathcal{B}$。假设 $\mathcal{B}=\{p_1,p_2,\cdots,p_B\}$。第一步(预处理步)是计算因子基中 B 个素数的离散对数。第二步,利用这些离散对数,计算所要求的离散对数。

假设 C 比 B 稍大,比如 $C=B+10$。预计算阶段,构造 C 个模 p 的同余方程,它们具有下述形式:

$$\alpha^{x_j}\equiv p_1^{a_{1j}} p_2^{a_{2j}}\cdots p_B^{a_{Bj}}(\bmod\ p)$$

$1\leqslant j\leqslant C$。它们等价于:

$$x_j\equiv a_{1j}\log_\alpha p_1+\cdots+a_{Bj}\log_\alpha p_B(\bmod\ p-1)$$

$1\leqslant j\leqslant C$。给定 B 个"未知量"$\log_\alpha p_i(1\leqslant i\leqslant B)$的 C 个同余方程,希望存在模 $p-1$ 下的惟一

解。如果是这样的话,就可以算出分解基元素的离散对数。

如何产生 C 个期望的同余方程呢?一个基本的方法是,随机地取一个数 x,计算 $\alpha^x \bmod p$,确定是否 $\alpha^x \bmod p$ 的所有因子在 $\mathcal{B}$ 中(例如,可以利用试除法)。

假设预计算步已经顺利实现。利用 Las Vegas 型的随机算法计算所求的离散对数。选择一个随机数 $s(1 \leqslant s \leqslant p-2)$,计算

$$\gamma = \beta \alpha^s \bmod p$$

选择试图在因子基 $\mathcal{B}$ 上分解 γ。如果成功,就得到如下的同余方程:

$$\beta \alpha^s \equiv p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B} (\bmod\ p)$$

等价于

$$\log_\alpha \beta + s \equiv c_1 \log_\alpha p_1 + \cdots + c_B \log_\alpha p_B (\bmod\ p-1)$$

由于上式中除了 $\log_\alpha \beta$ 外,其余的项都已知,容易解出 $\log_\alpha \beta$。

这里有一个特制的小例子,用来说明算法的两个步骤。

例 6.5　整数 $p = 10\,007$ 是素数。$\alpha = 5$ 是本原元素用做模 p 的离散对数的基。假设取 $\mathcal{B} = \{2,3,5,7\}$ 作为因子基。当然 $\log_5 5 = 1$,因此有三个因子基元素的对数要确定。4063,5136 和 9865 属于我们需要的"幸运"指数。

当 $x = 4063$,计算

$$5^{4063} \bmod\ \ 10\,007 = 42 = 2 \times 3 \times 7$$

产生同余方程

$$\log_5 2 + \log_5 3 + \log_5 7 \equiv 4063 (\bmod\ 10\,006)$$

类似地,由于

$$5^{5136} \bmod\ \ 10\,007 = 54 = 2 \times 3^3$$

和

$$5^{9865} \bmod\ \ 10\,007 = 189 = 3^3 \times 7$$

我们进一步得到两个同余方程:

$$\log_5 2 + 3\log_5 3 \equiv 5136 (\bmod\ 10\,006)$$

和

$$3\log_5 3 + \log_5 7 \equiv 9865 (\bmod\ 10\,006)$$

我们有了具有三个未知量的三个同余方程,并且模 10 006 具有惟一的解。即 $\log_5 2 = 6578$,$\log_5 3 = 6190$ 以及 $\log_5 7 = 1301$。

现在,假设要求 $\log_5 9451$。选择"随机"指数 $s = 7736$,计算

$$9451 \times 5^{7736} \bmod\ \ 10\,007 = 8400$$

因为 $8400 = 2^4 3^1 5^2 7^1$ 在$\mathcal{B}$上完全分解,得到

$$\begin{aligned}\log_5 9451 &= (4\log_5 2 + \log_5 3 + 2\log_5 5 + \log_5 7 - s) \bmod 10\,006 \\ &= (4 \times 6578 + 6190 + 2 \times 1 + 1301 - 7736) \bmod 10\,006 \\ &= 6057\end{aligned}$$

检查 $5^{6057} \equiv 9451 (\bmod 10\,007)$,得以验证。

人们对于各种版本的指数演算算法进行过启发式分析。在合理的假设下,例如在 5.6.3 小节,分析 Dixon 算法中考虑的那样,算法的预计算阶段花费的渐近运行时间为:

$$O(e^{(1+O(1))\sqrt{\ln p \ln\ln p}})$$

计算特定的离散对数所需时间为:

$$O(e^{(1/2+O(1))\sqrt{\ln p \ln\ln p}})$$

6.3 通用算法的复杂性下界

这一节,我们把注意力转到离散对数问题复杂性的一个有趣的下界上。我们描述的离散对数问题的算法,有几个可以应用到任何的群中。这种算法称为通用算法(generic algorithm),因为它不依赖于群的任何特别的性质。离散对数问题的通用算法包括 Shanks 算法,Pollard ρ 算法和 Pohlig-Hellman 算法。另一方面,上节介绍的指数演算算法不是通用的。这个算法将 $\mathbb{Z}_p^*$ 的元素视为整数,把它们分解成素数因子的乘积。很显然,这一点在一般群中难以做到。

另外一个关于特殊群的非通用算法的例子是,研究加法群$(\mathbb{Z}_n, +)$的离散对数算法(我们以前在乘法群中定义离散对数问题,目的只是为了对算法表示概念上的相容性)。假设 $\gcd(\alpha, n) = 1$,因此 α 是$\mathbb{Z}_n$ 的生成元。由于群的运算是模 n 的加法,"指数"运算 α^a 就对应于被模 n 的 a 相乘。因此,在这种方式下,离散对数问题就是找满足下列式子的 a:

$$\alpha a \equiv \beta (\bmod\ n)$$

根据 $\gcd(\alpha, n) = 1$,α 有一个模 n 的乘法逆元,可以利用扩展的欧几里德算法,容易算出 $\alpha^{-1} \bmod n$。然后解出 a,得到

$$\log_\alpha \beta = \beta\, \alpha^{-1} \bmod n$$

这个算法的运行当然很快;它的复杂性是 $\log n$ 的多项式。

当 $\alpha = 1$ 时,有一个平凡的算法计算$(\mathbb{Z}_n, +)$中的离散对数问题。这时,有 $\log_1 \beta = \beta$ 对于所有的$\beta \in \mathbb{Z}_n$。

根据定义,离散对数问题发生在 n 阶循环(子)群中。所有的 n 阶循环群是同构的,这是众所周知的事实,也是几乎显然可证的。通过上面的讨论,我们知道如何快速地在加法群$(\mathbb{Z}_n, +)$中计算离散对数。这意味着,我们或许能够把任何群 G 的 n 阶子群$\langle\alpha\rangle$中的离散对数问题,归约到在$(\mathbb{Z}_n, +)$中容易求解的形式。

我们考虑如何(至少在理论上)做到这一点。$\langle\alpha\rangle$同构于$(\mathbb{Z}_n, +)$,表明存在一个双射

$$\phi:\langle\alpha\rangle\rightarrow\mathbb{Z}_n$$

满足

$$\phi(xy)=(\phi(x)+\phi(y))\bmod n$$

对于所有 $x, y\in\langle\alpha\rangle$。易得

$$\phi(\alpha^a)=a\phi(\alpha)\bmod n$$

故而有

$$\beta=\alpha^a\Leftrightarrow a\phi(\alpha)\equiv\phi(\beta)(\bmod n)$$

所以,解出上述的 a(利用扩展的欧几里德算法),得

$$\log_\alpha\beta=\phi(\beta)(\phi(\alpha))^{-1}\bmod n$$

结论是,如果有一个有效的方法计算同构 ϕ,我们就具有有效的方法计算$\langle\alpha\rangle$中的离散对数。关键的问题是,即使我们知道两个群是同构的,对于任意群 G 的任何子群$\langle\alpha\rangle$,没有已知的有效方法,计算同构 ϕ。事实上,不难看出,计算$\langle\alpha\rangle$中的离散对数,等价于显式地给出$\langle\alpha\rangle$和$\mathbb{Z}_n$ 之间的同构。所以,这个方法看来走不通。

$(\mathbb{Z}_n, +)$中的离散对数问题有一个非常有效的算法,这个事实使得通用算法的复杂性看起来似乎不会存在有趣的下界。然而,事实并非如此。Shoup 给出了离散对数问题通用算法的一个下界。我们记得,Shanks 和 ρ 算法的复杂性(运行算法时需要的群运算数)大致是$\sqrt{n}$,这里的 n 是定义离散对数问题的(子)群的阶数。Shoup 的结论表明,在通用算法中,这些算法实质上是最优的。

我们首先精确地表述所谓的通用算法。考虑 n 阶循环(子)群,它同构于$(\mathbb{Z}_n, +)$。我们研究$(\mathbb{Z}_n, +)$中的离散对数问题的通用算法(我们将会看到,通用算法与这种特殊群的选取是无关的;$(\mathbb{Z}_n, +)$的选取是任意的)。

$(\mathbb{Z}_n, +)$的一个编码是一个单射 $\sigma:\mathbb{Z}_n\rightarrow S$,其中 S 是一个有限集。编码函数确定了群元素的表示方式。任意基数为 n 的(子)群中的离散对数问题,都可以通过定义适当的编码函数确定。例如,考虑乘法群$(\mathbb{Z}_p^*, \cdot)$,α 为$\mathbb{Z}_p^*$ 中一个本原元素。设 $n=p-1$,定义编码函数为 $\sigma(i)=\alpha^i\bmod p, 0\leqslant i\leqslant n-1$。那么,很清楚,解$(\mathbb{Z}_p^*, \cdot)$中关于本原元 α 的离散对数问题,在 σ 下等价于,在$(\mathbb{Z}_n, +)$中,解关于生成元的离散对数问题。

通用算法是适用于任何编码的算法。特别地,通用算法必须对于随机单射函数 σ 是正确的;例如,当 $S=\mathbb{Z}_n$,且 σ 是$\mathbb{Z}_n$ 的一个随机置换。这一点类似于随机预言模型,其中一个 Hash 函数作为随机函数,以定义进行形式安全证明的理想化模型。

假设对于群$(\mathbb{Z}_n, +)$,有一个随机编码 σ,该群中任意元素 a 对于基数 1 的离散对数当然就是 a。给定编码函数 σ,生成元的编码 $\sigma(1)$,以及群的一个任意元素的编码 $\sigma(a)$,通用算法企图计算元素 a。当群中的元素用函数 σ 编码后,为了在该群中施行群运算,我们假设存在一个预言器(或者一个子程序)完成这个任务。

给定两个群元素的编码,如 $\sigma(i)$和 $\sigma(j)$,应该能够计算 $\sigma((i+j)\bmod n)$和 $\sigma((i-j)\bmod$

n)。这对我们进行群元素的加法和减法是十分必要的,假设我们的预言器可以完成这个功能。通过组合上述类型的运算,可以计算形为 $\sigma((ci \pm dj) \bmod n)$的线性组合,其中 $c, d \in \mathbb{Z}_n$。而且,通过 $-j \equiv n - j (\bmod n)$,我们发现只需能够计算 $\sigma((ci + dj) \bmod n)$形式的线性组合。我们将假设预言器能在一个单位时间内直接计算出这种形式的线性组合。

通用算法中仅允许上述类型的群运算。即,我们假设可以有方法计算编码元素的群运算,并且仅此而已。我们考虑一个通用算法,比如称做 Genlog,是如何计算离散对数的。Genlog 的输入有 $\sigma_1 = \sigma(1)$和 $\sigma_2 = \sigma(a)$,其中 $a \in \mathbb{Z}_n$ 是随机选取的。当且仅当 Genlog 输出 a 的值时,它就成功了(为了简化分析,我们假设 n 是素数)。

Genlog 利用预言器产生一个 1 和 a 线性组合的编码序列,比如长度为 m。Genlog 的运行依照有序对$(c_i, d_i) \in \mathbb{Z}_n \times \mathbb{Z}_n, i \leqslant m$ 进行(我们假设这 m 个有序对是互不同的)。对每个有序对(c_i, d_i),预言器计算编码 $\sigma_i = \sigma((c_i + d_i a) \bmod n)$。我们可以定义$(c_1, d_1) = (1,0)$和$(c_2, d_2) = (0,1)$,因此这个概念与算法的输入相一致。

以这种方式,算法 Genlog 得到编码群元素的一个列$(\sigma_1, \cdots, \sigma_m)$。由于编码函数 σ 是单射,立即可以得出 $c_i + d_i a \equiv c_j + d_j a (\bmod n)$当且仅当 $\sigma_i = \sigma_j$。这给出了计算未知值 a 的一种可能的方法:假设对两个整数 $i \neq j$,有 $\sigma_i = \sigma_j$,如果 $d_i = d_j$,则 $c_i = c_j$,有序对(c_i, d_i)和(c_j, d_j)是相同的。因为我们假设了有序对是互不相同的,因而 $d_i \neq d_j$。由 n 是素数,可以计算 a 如下:

$$a = (c_i - c_j)(d_j - d_i)^{-1} \bmod n$$

(记得在 Pollard ρ 算法中我们使用了类似的方法计算离散对数的值)。

首先,假设算法 Genlog 在算法开始就选择了 m 个不同有序对的集合:

$$C = \{(c_i, d_i) : 1 \leqslant i \leqslant m\} \subseteq \mathbb{Z}_n \times \mathbb{Z}_n$$

这种算法称为非适应性算法(Shanks 算法属于非适应性算法)。然后可以由预言器得到 m 个对应的编码。定义 $\mathbf{Good}(\mathcal{C})$为所有方程 $a = (c_i - c_j)(d_j - d_i)^{-1} \bmod n$ 的解 $a \in \mathbb{Z}_n$ 组成的集合,这里 $i \neq j, i, j \in \{1, \ldots, m\}$。根据上述理由,$a$ 可以由 Genlog 计算,当 $a \in \mathbf{Good}(\mathcal{C})$且仅当。显然$|\mathbf{Good}(\mathcal{C})| \leqslant \binom{m}{2}$,因此当对应于$\mathcal{C}$中有序对的 m 个编码群元素序列得到后,Genlog 至多能够计算$\binom{m}{2}$个元素的离散对数值。$a \in \mathbf{Good}(\mathcal{C})$的概率至多是$\binom{m}{2} \Big/ n$。如果 $a \notin \mathbf{Good}(\mathcal{C})$,算法 Genlog 的最好策略是在$\mathbb{Z}_n / \mathbf{Good}(\mathcal{C})$中随机地猜一个值作为 a。记 $g = \mathbf{Good}(\mathcal{C})$。通过是否 $a \in \mathbf{Good}(e)$这个条件,我们可以计算算法成功概率的一个界。假设我们计算下面的随机变量:**a** 为事件 $a \in \mathbf{Good}(\mathcal{C})$;**b** 为事件"算法输出 a 的正确值",则有

$$\begin{aligned}\mathbf{Pr}[\mathbf{b}] &= \mathbf{Pr}[\mathbf{b}|\mathbf{a}]\mathbf{Pr}[\mathbf{a}] + \mathbf{Pr}[\mathbf{b}|\bar{\mathbf{a}}]\mathbf{Pr}[\bar{\mathbf{a}}] \\ &= 1 \times \frac{g}{n} + \frac{1}{n-g} \times \frac{n-g}{n} \\ &= \frac{g+1}{n}\end{aligned}$$

$$\leqslant \frac{\binom{m}{2}+1}{n}$$

当然，通用离散对数算法并不需要在算法一开始就选定$\mathcal{C}$中所有的序对。可以在看到前面的有序对是什么样子之后，再选择后面的有序对(即，我们允许算法是一个适应性算法)。然而，利用归纳方法，我们会看到，这并不能够改进算法成功的概率。

设 Genlog 是一个离散对数问题的适应性算法。对于 $1 \leqslant i \leqslant m$，令$\mathcal{C}_i$ 由前 i 个序对组成，预言器计算了对应$\mathcal{C}_i$ 的编码 $\sigma_1, \ldots, \sigma_i$。集合$\mathcal{C}_i$ 和列 $\sigma_1, \ldots, \sigma_i$ 是算法 Genlog 在运行时间 i 内可以得到的所有信息。

我们断言，如果 $a \in \mathbf{Good}(\mathcal{C}_i)$，那么 a 可以在时间计算 i 得到；并且如果 $a \notin \mathbf{Good}(\mathcal{C}_i)$，那么 a 在集合$\mathbb{Z}_n/\mathbf{Good}(\mathcal{C}_i)$中具有任意给定值的概率恰为 $1/(n - |\mathbf{Good}(\mathcal{C})|)$(这个概率是 a 在时间 i 的条件概率，假定 a 在$\mathbb{Z}_n$ 中随机选择)。

这个结论在 $i = 1$ 时是正确的(这是归纳的基础)，这是因为由定义，$\mathcal{C}_1 = (1,0)$，并且 $\mathbf{Good}(\mathcal{C}_1) = \emptyset$。如果论断对 $i = j - 1$ 正确，容易看出它对 $i = j$ 也是正确的。那么，结论由归纳法保证正确。当 $i = m$ 时，论断表明算法的成功概率至多为$\left(\binom{m}{2}+1\right)\Big/ n$，与非适应性算法情况完全相同。

如果 Genlog 要保证计算期望的离散对数，那么它的成功概率等于 1，上述结果就是$\left(\binom{m}{n}+1\right)\Big/ n \geqslant 1$，或 $m^2 + m \geqslant 2n$。因此，m 是 $O(\sqrt{n})$。这是素数 n 阶(子)群中离散对数问题的任何通用算法的一个复杂性下界。

6.4　有限域

ElGamal 密码体制可以在任何离散对数问题难处理的群中实现。在密码体制 6.1 的表述中我们使用了乘法群$\mathbb{Z}_n^*$，但是还有别的群也是合适的候选者。下面的两类群就属于这类群：

1. 有限域$\mathbb{F}_{p^n}$ 的乘法群
2. 定义在有限域上的椭圆曲线的群

我们将在后面介绍这两类群。

我们已经讨论过，当 p 是素数时，$\mathbb{Z}_p$ 是一个域。然而，还有其他形式的域。事实上，如果 $q = p^n$，p 是素数，$n \geqslant 1$ 是整数，就存在一个具有 q 个元素的域。我们将简要介绍这类域的构造方法。首先，我们需要几个定义。

定义 6.1　假设 p 是一个素数。定义$\mathbb{Z}_p[x]$是变元 x 的所有多项式的集合。按照通常多项式的乘法和加法定义(并且模 p 归约系数)，它构成一个环。

对于 $f(x), g(x) \in \mathbb{Z}_p[x]$，如果存在 $q \in \mathbb{Z}_p[x]$满足

$$g(x) = q(x)f(x)$$

则说 $f(x)$整除 $g(x)$(记做:$f(x) \mid g(x)$)。对 $f(x) \in \mathbb{Z}_p[x]$,f 的阶数 $\deg(f)$定义为 f 的项中最高次数。

假设 $f(x), g(x), h(x) \in \mathbb{Z}_p[x]$,且 $\deg(f) = n \geqslant 1$。如果

$$f(x) \mid (g(x) - h(x))$$

则定义:

$$g(x) \equiv h(x) (\mathrm{mod}\ f(x))$$

我们注意到,多项式的同余与整数的同余有着非常相似之处。

我们要定义一个"模 $f(x)$"的多项式环,记为$\mathbb{Z}_p[x]/(f(x))$。从$\mathbb{Z}_p[x]$构造$\mathbb{Z}_p[x]/(f(x))$是基于模 $f(x)$同余的观点,这类似于由$\mathbb{Z}$构造$\mathbb{Z}_m$。

假设 $\deg(f) = n$。用 $f(x)$去除 $g(x)$,得到(惟一)商 $q(x)$和余式 $r(x)$。其中:

$$g(x) = q(x)f(x) + r(x)$$

并且

$$\deg(r) < n$$

这可以由多项式的长除法实现。因此$\mathbb{Z}_p[x]$中任何多项式模 $f(x)$同余于惟一的次数至多为 $n-1$ 的多项式。

定义$\mathbb{Z}_p[x]/(f(x))$的元素是$\mathbb{Z}_p[x]$中所有 p^n 个次数不超过 $n-1$ 的多项式。加法和乘法与$\mathbb{Z}_p[x]$中相同,并且模 $f(x)$归约。有了这两个运算,$\mathbb{Z}_p[x]/(f(x))$是一个环。

我们知道,$\mathbb{Z}_m$ 是一个域当且仅当 m 是素数,乘法逆元可以通过欧几里德算法求得。对于$\mathbb{Z}_p[x]/(f(x))$有类似的情形。多项式中类似于素性的概念是不可约性,定义如下:

定义 6.2 一个多项式 $f(x) \in \mathbb{Z}_p[x]$称做是不可约的,如果不存在多项式 $f_1(x), f_2(x) \in \mathbb{Z}_p[x]$,满足

$$f(x) = f_1(x)f_2(x)$$

其中 $\deg(f_1) > 0$ 和 $\deg(f_2) > 0$。

一个重要的事实是,$\mathbb{Z}_p[x]/(f(x))$是域当且仅当 $f(x)$是不可约的。$\mathbb{Z}_p[x]/(f(x))$中元素的乘法逆元可以直接通过改进(扩展的)欧几里德算法计算。

下面是一个例子,可以说明上述概念。

例 6.6 我们试图构造具有八个元素的域。这可以通过在$\mathbb{Z}_2[x]$中找一个次数为 3 的不

可约多项式做到。因为任何常数项为 0 的多项式都可以被 x 整除，因而是可约的，我们只需考虑具有常数项为 1 的多项式。有四个这样的多项式。

$$f_1(x) = x^3 + 1$$
$$f_2(x) = x^3 + x + 1$$
$$f_3(x) = x^3 + x^2 + 1$$
$$f_4(x) = x^3 + x^2 + x + 1$$

$f_1(x)$是可约的，因为

$$x^3 + 1 = (x + 1)(x^2 + x + 1)$$

（记住，所有的系数要模 2 约化。）还有，$f_4(x)$是可约的，因为

$$x^3 + x^2 + x + 1 = (x + 1)(x^2 + 1)$$

但是，$f_2(x)$和 $f_3(x)$都是不可约的，任何一个都可以用于构造八个元素的域。

我们使用 $f_2(x)$，构造域$\mathbb{Z}_2[x]/(x^3 + x + 1)$。八个域元素是八个多项式 $0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x$ 和 $x^2 + x + 1$。

要计算两个域元素的乘积，我们进行多项式的相乘，并且模 $x^3 + x + 1$ 约化（即用 $x^3 + x + 1$ 去除，找出余式）。由于是用一个三次多项式去除，余式的次数至多是二，因此是该域中的元素。

例如，要计算$\mathbb{Z}_2[x]/(x^3 + x + 1)$中的$(x^2 + 1)(x^2 + x + 1)$，首先在$\mathbb{Z}_2[x]$中计算乘积，得到 $x^4 + x^3 + x + 1$。随后用 $x^3 + x + 1$ 去除，得到表达式

$$x^4 + x^3 + x + 1 = (x + 1)(x^3 + x + 1) + x^2 + x$$

因此，在域$\mathbb{Z}_2[x]/(x^3 + x + 1)$中，有

$$(x^2 + 1)(x^2 + x + 1) = x^2 + x$$

下面我们给出一个域中非零元素的乘法表。为了节省空间，将多项式 $a_2x^2 + a_1x + a_0$ 写做三元组 $a_2a_1a_0$。

	001	010	011	100	101	110	111
001	001	010	011	100	101	110	111
010	010	100	110	011	001	111	101
011	011	110	101	111	100	001	010
100	100	011	111	110	010	101	001
101	101	001	100	010	111	011	110
110	110	111	001	101	011	010	100
111	111	101	010	001	110	100	011

通过直接应用扩展的欧几理德算法，可以计算域元素的逆元。

最终，域中非零多项式是一个七阶乘法群，由于 7 是素数，所以域中除 0 和 1 外，任何元素都是本原元。

例如，计算 x 的幂，可以得到：

$$x^1 = x$$
$$x^2 = x^2$$
$$x^3 = x + 1$$
$$x^4 = x^2 + x$$
$$x^5 = x^2 + x + 1$$
$$x^6 = x^2 + 1$$
$$x^7 = 1$$

这囊括了域中的所有非零元素。

余下要讨论这类域的存在性和惟一性。可以证明,在$\mathbb{Z}_p[x]$中存在任意给定次数 $n \geqslant 1$ 的不可约多项式。通常,$\mathbb{Z}_p[x]$中有许多次数为 $n \geqslant 1$ 的不可约 n 次多项式。但是可以证明,由任何两个不可约 n 次多项式构造的域是同构的。因此,存在惟一的 p^n(n 是素数,$n \geqslant 1$)个元素的域,记做$\mathbb{F}_{p^n}$。$n = 1$ 时,$\mathbb{F}_p$ 与 $\mathbb{Z}_p$ 相同。最后,可以证明,如果存在 r 阶的域,那么一定存在某个素数 p 以及某个整数 $n \geqslant 1$,使得 $r = p^n$。

我们已经注意到,乘法群$\mathbb{Z}_p^*$(p 是素数)是一个阶数为 $p - 1$ 的循环群。事实上,任何有限域的乘法群都是循环群:$\mathbb{F}_{p^n}/\{0\}$是一个 $p^n - 1$ 阶的循环群。这进一步给出了可供研究离散对数问题的循环群。

在实际中,研究最多的是有限域$\mathbb{F}_{2^n}$。任何通用算法自然适用于域$\mathbb{F}_{2^n}$。然而,更重要的是,指数演算法可以直接改进,应用于这些域中。我们记得,指数演算法的主要步骤是在一个由小素数组成的分解基上分解$\mathbb{Z}_p$ 中的元素。类似的分解基在$\mathbb{Z}_2[x]$中是一组低次数的不可约多项式。想法是在给定的分解基上分解$\mathbb{F}_{2^n}$ 中的元素。读者可以容易地给出详细步骤。

做适当的改进后,在$\mathbb{F}_{2^n}$ 中指数演算法地预计算时间为:

$$O\left(e^{(1.405+o(1))n^{1/3}(\ln n)^{2/3}}\right)$$

计算一个离散对数的时间为:

$$O\left(e^{(1.098+o(1))n^{1/3}(\ln n)^{2/3}}\right)$$

对于大数 n(比如,$n > 1024$),只要 $2^n - 1$ 至少有一个“大”素因子(为了抵抗 Pohlig-Hellman 攻击),$\mathbb{F}_{2^n}$ 上离散对数问题当前被认为是计算上不可行的。

6.5 椭圆曲线

椭圆曲线被描述为一个二元方程解的集合。模 p 定义的椭圆曲线在公钥密码中非常重要。我们先看定义在实数上的椭圆曲线,因为这种情形中的许多基本概念更易理解。

6.5.1　实数上的椭圆曲线

定义 6.3　设 $a, b \in \mathbb{R}$ 是满足 $4a^2 + 27b^3 \neq 0$ 的实数。方程

$$y^2 = x^3 + ax + b$$

的所有解 $(x, y) \in \mathbb{R} \times \mathbb{R}$ 的集合 E，加上一个无穷远点 $\mathcal{O}$，组成了一个非奇异椭圆曲线。

可以证明，条件 $4a^3 + 27b^2 \neq 0$ 是保证方程 $y^2 = x^3 + ax + b$ 有三个不同解（实数或复数）的充要条件。如果 $4a^3 + 27b^2 = 0$，则对应的椭圆曲线称为奇异椭圆曲线。

图 6.1 画出了椭圆曲线 $y^2 = x^3 - 4x$。

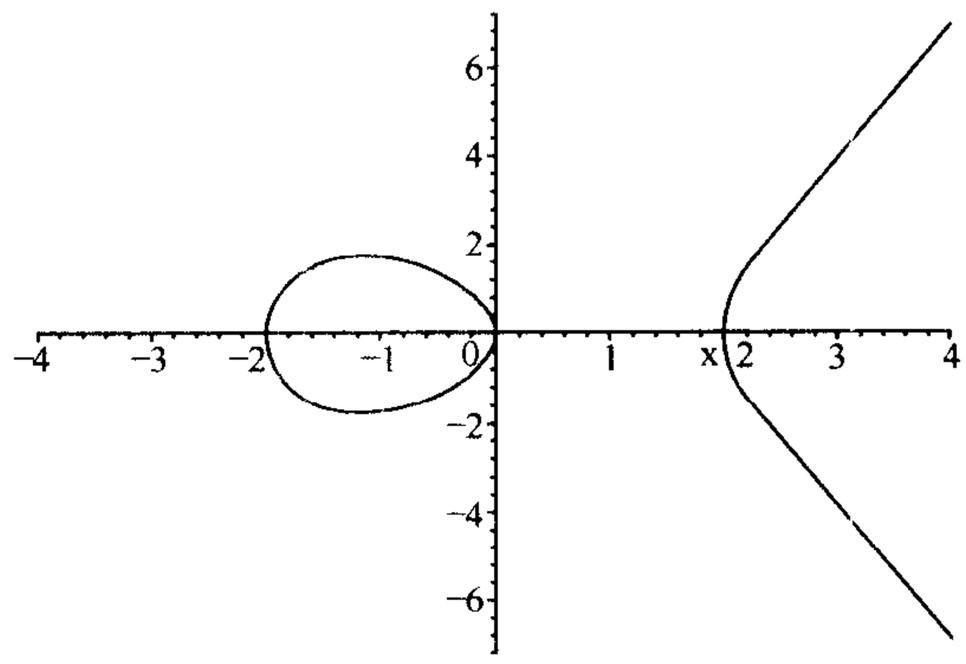

图 6.1　实数域上的椭圆曲线

假设 E 是一个非奇异椭圆曲线。我们在 E 上定义一个二元运算，使其成为一个阿贝尔群。这个二元运算通常用加法表示。无穷远点 $\mathcal{O}$ 将是单位元。因此有 $P + \mathcal{O} = \mathcal{O} + P = P$，对于所有 $P \in E$。

假设 $P, Q \in E$，其中 $P = (x_1, y_1)$，$Q = (x_2, y_2)$。我们分三种情形讨论：

1. $x_1 \neq x_2$

2. $x_1 = x_2$，且 $y_1 = -y_2$

3. $x_1 = x_2$，且 $y_1 = y_2$

情形 1　定义 L 是通过 P 和 Q 的直线。L 交 E 于 P 和 Q，容易看出，L 还交 E 于第三点，记作 R'。对 x 轴反射 R'，得到一点 R。定义 $P + Q = R$。

我们给出一个计算 R 的代数公式。首先，使方程 L 为 $y = \lambda x + v$，其中 L 的斜率是：

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

并且

$$v = y_1 - \lambda x_1 = y_2 - \lambda x_2$$

为了算出 $E \cap L$ 中的点,将 $y = \lambda x + v$ 代入到 E 的方程中,得到

$$(\lambda x + v)^2 = x^3 + ax + b$$

等价于

$$x^3 - \lambda^2 x^2 + (a - 2\lambda v)x + b - v^2 = 0 \tag{6.5}$$

方程(6.5)的根是 $E \cap L$ 中点的 x 坐标。我们已经知道 $E \cap L$ 中的两个点,即 P 和 Q,因此 x_1 和 x_2 是方程(6.5)的两个根。

方程(6.5)是实数域上的三次方程,具有两个实根,那么第三个根也应该是实根,记为 x_3。三根之和是二次项系数 λ^2 的相反数。所以

$$x_3 = \lambda^2 - x_1 - x_2$$

x_3 是点 R' 的 x 坐标。R' 的 y 坐标记作 $-y_3$,则 R 的 y 坐标就是 y_3。求 y_3 的一个简单的办法是利用 L 的斜率是由 L 上的两点确定这个事实。用点 (x_1, y_1) 和 $(x_3, -y_3)$ 计算这个斜率,得到:

$$\lambda = \frac{-y_3 - y_1}{x_3 - x_1}$$

即

$$y_3 = \lambda(x_1 - x_3) - y_1$$

所以对情形 1,我们导出 $P + Q$ 的一个计算公式:如果 $x_1 \neq x_2$,那么 $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$,其中:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

情形 2 当 $x_1 = x_2, y_1 = -y_2$ 时,我们定义 $(x, y) + (x, -y) = O, (x, y) \in E$。因此,$(x, y)$ 和 $(x, -y)$ 是关于椭圆曲线加法运算互逆的。

情形 3 我们要把一个点 $P = (x_1, y_1)$ 与自己相加。假设 $y_1 \neq 0$,否则就是情形 2。情形 3 与情形 1 非常类似,只是我们定义 L 是 E 在 P 点的切线。运用微积分的知识可以使计算简单一些。计算 L 的斜率要利用对 E 的方程隐微分:

$$2y \frac{dy}{dx} = 3x^2 + a$$

替换 $x = x_1, y = y_1$,得到切线的斜率为:

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

余下的分析与情形 1 完全相同。得到的公式也是一样的，只是斜率的计算不同。

诚如上述定义所示，加法运算的下列性质应该是明确的：

1. 加法在集合 E 上是封闭的。

2. 加法是可交换的。

3. $\mathcal{O}$是加法的单位元。

4. E 上每个点有关于加法的逆元。

要证明$(E,+)$是阿贝尔群，还须证明加法是结合的。用代数方法证明这一点比较繁杂。利用一些几何结果可以使证明得到简化；但是，我们不在这里讨论这个证明。

6.5.2　模素数的椭圆曲线

设 $p>3$ 是素数。$\mathbb{Z}_p$ 上的椭圆曲线可以与实数上的一样定义(加法定义的方式也相同)，只是$\mathbb{R}$上的运算用$\mathbb{Z}_p$ 中的类似运算代替。

定义 6.4　$p>3$ 是素数。$\mathbb{Z}_p$ 上的同余方程

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{6.6}$$

的所有解$(x,y)\in \mathbb{Z}_p \times \mathbb{Z}_p$，连同一个特殊的点$\mathcal{O}$即无穷远点，共同构成$\mathbb{Z}_p$ 上的椭圆曲线 $y^2 = x^3 + ax + b$，其中 $a,b\in \mathbb{Z}_p$ 是满足 $4a^2 + 27b^3 \not\equiv 0$ 的常量。

E 上的加法定义如下（这里的所有运算都在$\mathbb{Z}_p$ 中）：假设

$$P(x_1, y_1)$$

以及

$$Q = (x_2, y_2)$$

是 E 上的点。如果 $x_1 = x_2$ 且 $y_2 = -y_1$，则 $P + Q = \mathcal{O}$；否则 $P + Q = (x_3, y_3)$，其中

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

且

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1} & P = Q \end{cases}$$

最后，对于所有的 $P\in E$，定义

$$P + \mathcal{O} = \mathcal{O} + P = P$$

注意，$\mathbb{Z}_p$ 上的椭圆曲线没有实数上椭圆曲线的直观的几何解释。然而，同样的公式可以

用来定义加法运算,$(E, +)$仍然是一个阿贝尔群。

我们看一个简单例子。

例 6.7 设 E 是$\mathbb{Z}_{11}$上的椭圆曲线 $y^2 = x^3 + x + 6$。我们首先确定 E 的点。这可以通过对每个 $x \in \mathbb{Z}_{11}$,计算 $x^3 + x + 6 \bmod 11$,试着解方程(6.6)求 y。对于给定的 x,可以利用 Euler 判别法来测试是否 $z = x^3 + x + 6 \bmod 11$ 是一个二次剩余。我们知道,对素数 $p \equiv 3 \pmod 4$,有一个现成的公式计算模 p 的剩余。利用这个公式,二次剩余 z 的平方根是:

$$\pm z^{(11+1)/4} \bmod 11 = \pm z^3 \bmod 11$$

这些计算结果列在表 6.1 中。

表 6.1 $\mathbb{Z}_{11}$上椭圆曲线 $y^2 = x^3 + x + 6$ 的点

x	$x^3 + x + 6 \bmod 11$	是否为二次剩余	y
0	6	否	
1	8	否	
2	5	是	4,7
3	3	是	5,6
4	8	否	
5	4	是	2,9
6	8	否	
7	4	是	2,9
8	9	是	3,8
9	7	否	
10	4	是	2,9

E 有 13 个点。因为任意素数阶的群是循环群,因此 E 同构于$\mathbb{Z}_{13}$,且任何非无穷远点都是 E 的生成元。假设取生成元 $\alpha = (2,7)$。可以计算 α 的"幂"(因为群的运算是加法,可以写成 α 的乘积)。要计算 $2\alpha = (2,7) + (2,7)$,首先计算

$$\begin{aligned}\lambda &= (3 \times 2^2 + 1)(2 \times 7) \bmod 11 \\ &= 2 \times 3^{-1} \bmod 11 \\ &= 2 \times 4 \bmod 11 \\ &= 8\end{aligned}$$

所以有:

$$\begin{aligned}x_3 &= 8^2 - 2 - 2 \bmod 11 \\ &= 5\end{aligned}$$

和

$$\begin{aligned}y_3 &= 8(2 - 5) - 7 \bmod 11 \\ &= 2\end{aligned}$$

因此 $2\alpha = (5,2)$。

下一个乘积是 $3\alpha = 2\alpha + \alpha = (5,2) + (2,7)$。再次计算 λ，这时可以计算如下：

$$\begin{aligned}\lambda &= (7-2)(2-5)^{-1} \bmod 11\\ &= 5 \times 8^{-1} \bmod 11\\ &= 5 \times 7 \bmod 11\\ &= 2\end{aligned}$$

那么有

$$\begin{aligned}x_3 &= 2^2 - 5 - 2 \bmod 11\\ &= 8\end{aligned}$$

以及

$$\begin{aligned}y_3 &= 2(5-8) - 2 \bmod 11\\ &= 3\end{aligned}$$

因此 $3\alpha = (8,3)$。

如此继续下去，其余的乘积计算结果如下：

$$\begin{array}{lll}\alpha = (2,7) & 2\alpha = (5,2) & 3\alpha = (8,3)\\ 4\alpha = (10,2) & 5\alpha = (3,6) & 6\alpha = (7,9)\\ 7\alpha = (7,2) & 8\alpha = (3,5) & 9\alpha = (10,9)\\ 10\alpha = (8,8) & 11\alpha = (5,9) & 12\alpha = (2,4)\end{array}$$

所以，诚如所料，$\alpha = (2,7)$的确是本原元。

现在看一个利用例 6.7 中椭圆曲线 ElGamal 加密解密的例子。椭圆曲线的群运算是加法，所以我们将密码体制 6.1 中的运算翻译成加法。

例 6.8　假设 $\alpha = (2,7)$，Bob 的私钥是 7，有

$$\beta = 7\alpha = (7,2)$$

这样，加密运算是：

$$e_K(x,k) = (k(2,7), x + k(7,2))$$

其中 $x \in E$ 及 $0 \leqslant k \leqslant 12$，解密运算是：

$$d_K(y_1, y_2) = y_2 - 7y_1$$

假设 Alice 要加密明文 $x = (10,9)$（这是 E 上的一个点）。如果随机选择了 $k = 3$，那么她要计算

$$\begin{aligned}y_1 &= 3(2,7)\\ &= (8,3)\end{aligned}$$

和

$$\begin{aligned} y_2 &= (10,9) + 3(7,2) \\ &= (10,9) + (3,5) \\ &= (10,2) \end{aligned}$$

所以,$y = ((8,3),(10,2))$。现在,如果 Bob 收到密文 y,解密如下:

$$\begin{aligned} x &= (10,2) - 7(8,3) \\ &= (10,2) - (3,5) \\ &= (10,2) + (3,6) \\ &= (10,9) \end{aligned}$$

因此,解密得到正确的明文。

6.5.3 椭圆曲线的性质

定义在$\mathbb{Z}_p$(p 是素数,$p > 3$)上的椭圆曲线 E 大致有 p 个元素。更精确地,有一个由 Hasse 提出的著名定理断言,E 上的点数如果记做 $\#E$,那么它满足下面的不等式:

$$p + 1 - 2\sqrt{p} \leqslant \#E \leqslant p + 1 + 2\sqrt{P}$$

计算 $\#E$ 的准确值较为困难,但有一个有效的算法计算它,这个算法由 Schoof 发现(这里"有效"的意思是,算法的运行时间是 $\log p$ 的多项式。Schoof 的算法运行时间为 $O((\log p)^8)$多个位运算,即 $O((\log p)^6)$多个$\mathbb{Z}_p$ 中域运算)。对于数百位的素数 p 来说,这个算法比较实用。

如果可以计算 $\#E$,进一步要找到 E 的一个循环子群,在其中离散对数问题是难解的。所以要对群 E 的结构有所了解。下述定理给出了有关 E 的结构的许多信息。

定理 6.1 E 是定义在$\mathbb{Z}_p$ 上的一个椭圆曲线,其中 p 是素数,$p > 3$。则存在正整数 n_1 和 n_2,使得$(E, +)$同构于$\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$,并且有 $n_2 \mid n_1$ 和 $n_2 \mid (p-1)$。

注意,在上述定理中 $n_2 = 1$ 是可能的。事实上,当且仅当 E 是循环群时,$n_2 = 1$。还有,如果 $\#E$ 或者是一个素数,或者是两个不同素数的乘积,E 也必定是循环群。

无论如何,一旦整数 n_1 和 n_2 算定,我们就知道 E 具有一个循环子群同构于$\mathbb{Z}_{n_1}$,这意味着它可以用于 ElGamal 密码体制的情形。

通用算法适用于椭圆曲线的离散对数问题,而指数演算对椭圆曲线情形的适用性不得而知。但有一个方法明白地揭示了椭圆曲线与有限域的同构,这对一些椭圆曲线类给出了一个有效的算法。这个技巧由 Menezes、Okamoto 和 Vanstone 开发出来,它适用于所谓超奇异曲线,这是一类特殊椭圆曲线,它们被建议用于密码体制中。

另外一类弱椭圆曲线,就是所谓的"迹一(trace one)"曲线。这些曲线定义在 $\mathbb{Z}_p$(p 是素数)上,具有 p 个元素。这些曲线上的椭圆曲线问题是易解的。

看来,如果避免上面这类曲线,具有一个大约 2^{160} 个元素循环子群的椭圆曲线对于密码系

统来说,是一种安全的情形,条件是子群的阶数至少有一个大素数因子(也是为了避免 Pohlig-Hellman 攻击)。

6.5.4　点压缩与 ECIES

在实际中,实现椭圆曲线上的 ElGamal 密码体制存在着一些困难。在 $\mathbb{Z}_p$ 中实现 ElGamal 密码体制有一个二倍的消息扩张因子。椭圆曲线的实现有一个大约四倍的消息扩张因子。之所以如此,因为大致有 p 个明文,但每个密文有四个域元素组成。然而,更为严重的问题是,密文空间由 E 上的点组成,没有一个方便的方法能够确定地生成 E 上的点。

一个较为有效的 ElGamal 型体制用于所谓的 ECIES (椭圆曲线集成加密方案)。ECIES 不仅结合了消息认证码,而且结合了对称密钥加密,叙述起来十分复杂。我们给出一个简化版,主要实现用于 ECIES 的基于 ElGaml 公钥加密方案的椭圆曲线。在这个简化中,椭圆曲线的 x 坐标只是为"掩人耳目",一个密文可以是任意的(非零)域元素(即,不要求是 E 上的点)。

我们还要用到另外一个标准的窍门,称为点压缩,它可以降低椭圆曲线上点的存储空间。椭圆曲线 E 上的一个(非无穷)点是一个对 (x, y),其中 $y^2 \equiv x^3 + ax + b \pmod p$。给定一个 x 的值,对 y 有两个可能的值(除非 $x^3 + ax + b \equiv 0 \pmod p$)。这两个可能的 y 值模 p 互为相反数。因为 p 是奇数,两个可能的 $y \bmod p$ 值中,有一个是奇数,另一个是偶数。因此可以通过指定 x 的值,连同 $y \bmod 2$ 一个比特来确定 E 上惟一点 $P = (x, y)$。这减少了大约 50% 的存储空间,代价是需要额外的计算来重构 P 点的 y 坐标。

点压缩运算可以表示为函数:

$$\text{PointCompress}: E \setminus \{\mathcal{O}\} \to \mathbb{Z}_p \times \mathbb{Z}_2$$

具体定义为:

$$\text{PointCompress}(P) = (x, y \bmod 2), \text{其中 } P = (x, y) \in E$$

逆运算 PointDecompress 从 $(x, y \bmod 2)$ 重构 E 上的点 $P = (x, y)$。可以通过如下算法实现。

算法 6.4　PointDecompress(x, i)

$z \leftarrow x^3 + ax + b \bmod p$

if z 是模 p 的非二次剩余类

then return("failure")

$$\textbf{else}\begin{cases} y \leftarrow \sqrt{z} \bmod p \\ \textbf{if } y \equiv i \pmod 2 \\ \quad \textbf{then return}(x, y) \\ \quad \textbf{else return}(x, p - y) \end{cases}$$

如前所述,对于 $p \equiv 3 \pmod 4$ 以及 z 是模 p 的二次剩余(或 $z = 0$),$\sqrt{z}$ 可按照公式 $z^{(p+1)/4}$

mod p 计算。

我们所谓的简化的 ECIES 表述为密码体制 6.2。

密码体制 6.2 简化的 ECIES

令 E 是定义在$\mathbb{Z}_p$($p>3$ 素数)上的一个椭圆曲线,E 包含一个素数阶 n 的循环子群 $H=\langle P\rangle$,在其上离散对数问题是难解的。

设$\mathcal{P}=\mathbb{Z}_p^*$,$\mathcal{C}=(\mathbb{Z}_p\times\mathbb{Z}_2)\times\mathbb{Z}_p^*$,定义

$$\mathcal{K}=\{(E,P,m,Q,n):Q=mP\}$$

值 P,Q 和 n 是公钥,$m\in\mathbb{Z}_p^*$ 是私钥。

对于 $K=(E,P,m,Q,n)$,一个(秘密)随机数 $k\in\mathbb{Z}_n^*$,以及 $x\in\mathbb{Z}_p^*$,定义

$$e_K(x,k)=(\text{PointCompress}(kP),xx_0 \bmod p)$$

其中 $kQ=(x_0,y_0)$且 $x_0\neq 0$。

对密文 $y=(y_1,y_2)$,这里 $y_1\in\mathbb{Z}_p\times\mathbb{Z}_2$,$y_2\in\mathbb{Z}_p^*$,定义

$$d_K(y)=y_2(x_0)^{-1}\bmod p$$

其中

$$(x_0,y_0)=m\ \text{PointDecompress}(y_1)$$

简化的 ECIES 有一个(近似)等于 2 的消息扩张。这类似于$\mathbb{Z}_p^*$ 上的 ElGamal 密码体制。我们用定义在$\mathbb{Z}_{11}$上的椭圆曲线 $y=x^3+x+6$ 来说明简化的 ECIES 的加密和解密过程。

例 6.9 与上例一样,假设 $P=(2,7)$,Bob 的私钥是 7,则有:

$$Q=7P=(7,2)$$

如果 Alice 要加密明文 $x=9$,选择随机数 $k=6$,首先,她要计算:

$$kP=6(2,7)=(7,9)$$

和

$$kQ=6(7,2)=(8,3)$$

所以 $x_0=8$

其次,算出

$$y_1=\text{PointCompress}(7,9)=(7,1)$$

和

$$y_2=8\times 9 \bmod 11=6$$

送给 Bob 的密文是：

$$y = (y_1, y_2) = ((7,1),6)$$

一旦 Bob 收到密文 y，他通过计算：

$$\text{PointDecompress}(7,1) = (7,9)$$
$$7(7,9) = (8,3)$$
$$6 \times 8^{-1} \bmod 11 = 9$$

解密得到正确的明文 9。

6.5.5　计算椭圆曲线上点的乘积

我们可以利用平方-乘算法(参见算法 5.5)在乘法群中有效地计算幂 α^a。椭圆曲线中群运算写为加法，我们可以用一个类似的称为“倍数-和”算法计算椭圆曲线点 P 的倍数 aP(平方运算 $\alpha \mapsto \alpha^2$ 由倍数运算 $P \mapsto 2P$ 代替，两个群元素的乘积换为椭圆曲线上两点的和)。

椭圆曲线上的加法运算有这样的性质，加法的逆非常容易计算。这个事实可以通过倍数-和算法的扩展来体现，这个扩展称为“倍数-和差”算法。我们现在描述这个技巧。

设 c 是一个整数。c 的一个带符号的二进制表示是一个如下形式的方程：

$$c = \sum_{i=0}^{l-1} c_i 2^i$$

其中对所有的 i，$c_i \in \{-1,0,1\}$。一般来讲，一个整数有多个二进制表示，例如：

$$11 = 8 + 2 + 1 = 16 - 4 - 1$$

所以

$$(c_4, c_3, c_2, c_1, c_0) = (0,1,0,1,1) \text{ 或者 } (1,0,-1,0,-1)$$

两者都是 11 的带符号的二进制表示。

设 P 是一个 n 阶椭圆曲线的点。给定整数 c 的一个带符号的二进制表示，其中 $0 \leqslant c \leqslant n-1$，运用下述算法，通过一系列的倍运算、加减法计算出椭圆曲线的点 P 的乘积 cP。

算法 6.5　倍数-和差算法 DOUBLE-AND-(ADD OR SUBTRACT)$(P,(c_{l-1},\cdots,c_0))$

$Q \leftarrow \mathcal{O}$

for $i \leftarrow l-1$ **downto** 0

do
- $Q \leftarrow 2Q$
- **if** $c_i = 1$
  - **then** $Q \leftarrow Q + P$
  - **else if** $c_i = -1$
  - **then** $Q \leftarrow Q - P$

return(Q)

算法 6.5 中的减法 $Q - P$,应该先计算 P 的加法逆 $-P$,然后将结果与 Q 相加。

整数 c 的一个带符号的二进制表示$(c_{l-1},\cdots,c_0)$,如果其中没有两个连续的 c_i 是非零的,则说它是非相邻的形式。这样的表示称作 NFA 表示。我们可以很容易地将整数 c 的一个带符号的二进制表示转换为 NFA 表示。转换的基础是,在二进制表示中,用(1,0,...,0,-1)替换(0,1,...,1)。这种类型的替换并不改变 c 的值,因为有等式:

$$2^i + 2^{i-1} + \cdots + 2^j = 2^{i+1} - 2^j$$

这里 $i > j$。这个过程从最右面(即低位)的位开始向左,按照需要重复进行。

我们用一个例子说明上述过程:

$$\begin{array}{cccccccccccc}
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & & & & & & \downarrow & & & & & \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \\
 & & & & & & \downarrow & & & & & \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 \\
 & & & & & & \downarrow & & & & & \\
1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1
\end{array}$$

因此,

$$(1,1,1,1,0,0,1,1,0,1,1,1)$$

的 NAF 表示是:

$$(1,0,0,0,-1,0,1,0,0,-1,0,0,-1)$$

这个讨论说明,任何一个非负整数有一个 NAF 表示。一个整数的 NAF 表示可以证明是惟一的(见练习)。所以我们可以毫无疑义地说一个整数的 NAF 表示。

一个 NAF 表示没有两个相邻的非零系数。我们可以期望,一般情况下,NAF 表示比通常的二进制表示中有更多的零。事实的确如此:可以证明,平均来说,一个 l 比特整数的二进制表示中含有 $l/2$ 个零,它的 NAF 表示中含有 $2l/3$ 个零。

这些结果使得我们非常容易地比较下面两者的平均有效性:使用二进制表示的倍数-和算法和使用 NAF 表示的倍数-和差算法。每个算法需要 l 次倍运算,但在第一个算法中需要 $l/2$ 次加法(或减法)运算,而在第二个算法中需要 $l/3$ 次。如果假设倍运算与加法(或减法)花费大致相同的时间,那么,两个算法花费时间的比率近似为:

$$\frac{l + \frac{l}{2}}{l + \frac{l}{3}} = \frac{9}{8}$$

利用这个简单的技巧,我们可以平均提高(大致)11% 的速度。

6.6　实际中的离散对数算法

在密码应用中最为重要的离散对数问题(G,α),包括下列诸情形:

1. $G=(\mathbb{Z}_p^*,\cdot)$,p 是素数,α 是模 p 的一个本原元。
2. $G=(\mathbb{Z}_p^*,\cdot)$,p,q 是素数,$p\equiv 1 \bmod q$,α 是$\mathbb{Z}_p$ 中的一个 q 阶的元素。
3. $G=(\mathbb{F}_{2^n}^*,\cdot)$,$\alpha$ 是$\mathbb{F}_{2^n}^*$ 中的一个本原元素。
4. $G=(E,+)$,其中 E 是模素数 p 的一个椭圆曲线,$\alpha\in E$ 是一个具有素数 $q=\#E/h$ 阶的点,这里(典型的)$h=1,2$ 或 4。
5. $G=(E,+)$,其中 E 是有限域$\mathbb{F}_{2^n}$ 上的椭圆曲线,$\alpha\in E$ 是一个具有素数 $q=\#E/h$ 阶的点,这里(典型的)$h=2$ 或 4 .

对情形 1、2 和 3,利用$(\mathbb{Z}_p^*,\cdot)$或$(\mathbb{F}_{2^n}^*,\cdot)$上 Index-Calculus 算法的适当形式可以形成攻击。对于情形 2、4 和 5,利用 q 阶子群中 Pollard ρ 算法可以形成攻击。

我们简单地报告一些由 Lenstra 和 Verheul 估计的相对安全性结果。要使基于椭圆曲线离散对数的密码体制保持安全到 2020 年,对情形 4,建议应该取 $p\approx 2^{160}$(或者在情形 5 中取 $n\approx 160$)。而对于情形 1、2,要达到相同程度(预计)安全水平,p 至少应为 2^{1880}。导致如此大差异的原因是,对椭圆曲线离散对数没有已知的指数演算攻击。其结果,椭圆曲线密码在实际应用中越来越流行,在诸如无线装置和智能卡这些受限平台上的应用尤其如此。这类平台的可用存储空间非常小,例如$(\mathbb{Z}_p^*,\cdot)$上,基于离散对数的安全实现需要太多的空间,因而是不切实际的。基于椭圆曲线的密码需要较小的空间,正如所愿。

上述估计是一种猜测,它是在对未来若干年算法设计和计算速度发展的合理假设下,基于目前最好的算法做出的。观察一下目前的离散对数问题算法的发展水平也是有趣的。由于椭圆曲线密码在实际应用中的重要性,椭圆曲线上的离散对数问题近年来受到极大的关注。为了激励实现有效的离散对数算法,Certicom 公司发出了一系列的“挑战”。最近解决的比较困难的挑战是定义在$\mathbb{F}_{2^{109}}$ 上的椭圆曲线的离散对数算法。这个挑战称为 ECC2K-108,它是联合了 40 个国家的 9500 台计算机于 2000 年 4 月解决的。所花费的时间大约是解决名为 RSA-512 的 RSA 挑战所花费的 50 倍。

6.7　ElGamal 体制的安全性

本节我们研究 ElGamal 型密码体制安全性的几个方面。首先我们看离散对数的比特安全性,其次考虑 ElGamal 型密码体制的语义安全性,最后介绍 Diffie-Hellman 问题。

6.7.1　离散对数的比特安全性

这一小节我们考虑离散对数单个位的计算的难易。确切地说,就是考虑问题 6.2,我们称之为离散对数的第 i 位问题(本小节考虑的是$(\mathbb{Z}_p^*,\cdot)$上的离散对数,p 是素数)。

问题 6.2 离散对数第 i 比特问题

条件： $I=(p,\alpha,\beta,i)$，其中 p 是素数，$\alpha\in\mathbb{Z}_p^*$ 是一个本原元，$\beta\in\mathbb{Z}_p^*$，i 是一个整数，满足 $1\leqslant i\leqslant\lceil\log_2(p-1)\rceil$

问题： 计算 $L_i(\beta)$，这是(对于确定的 α 和 p)$\log_\alpha\beta$ 二进制表示的第 i 个最低比特。

我们先证明，计算一个离散对数的最低比特是容易的。换句话说，如果 $i=1$，那么离散对数的第 i 比特问题可以有效地计算。主要是利用 Euler 的关于模 p 二次剩余的判定法则，p 是素数。

映射 $f:\mathbb{Z}_p^*\rightarrow\mathbb{Z}_p^*$ 定义为：

$$f(x)=x^2 \bmod p$$

记 QR(p)为模 p 的二次剩余的集合；因此有：

$$\mathrm{QR}(p)=\{x^2 \bmod p : x\in\mathbb{Z}_p^*\}$$

首先，有 $f(x)=f(p-x)$。其次，注意到

$$w^2\equiv x^2 \pmod p$$

当且仅当

$$p\mid(w-x)(w+x)$$

后者成立当且仅当

$$w\equiv\pm x \pmod p$$

所以对每个 $y\in\mathrm{QR}(p)$，

$$|f^{-1}(y)|=2$$

因此，

$$|\mathrm{QR}(p)|=\frac{p-1}{2}$$

即，$\mathbb{Z}_p^*$ 中恰有一半的元素是二次剩余，一半不是。

现在，假设 α 是 $\mathbb{Z}_p$ 的一个本原元。那么如果 a 是偶数，则 $\alpha^a\in\mathrm{QR}(p)$。因为 $(p-1)/2$ 个元素 $\alpha^0 \bmod p,\alpha^2 \bmod p,\ldots,\alpha^{p-3} \bmod p$ 互不相同，因此，

$$\mathrm{QR}(p)=\{\alpha^{2i} \bmod p : 0\leqslant i\leqslant(p-3)/2\}$$

所以，β 是二次剩余当且仅当 $\log_\alpha\beta$ 是偶数，也就是说，当且仅当 $L_1(\beta)=0$。但由 Euler 判别法可知，β 是一个二次剩余当且仅当

$$\beta^{(p-1)/2}\equiv 1 \pmod p$$

所以,有下列有效的公式计算 $L_1(\beta)$:

$$L_1(\beta)=\begin{cases}0 & \beta^{(p-1)/2}\equiv 1(\bmod\ p)\\ 1 & \text{其他情况}\end{cases}$$

现在考虑如何计算 i 大于 1 时 $L_i(\beta)$的值。假设 $p-1=2^s t$, t 是奇数。可以证明对 $i\leqslant s$,容易计算 $L_i(\beta)$。另一方面,计算 $L_{s+1}(\beta)$ (或许)是困难的,因为任何计算 $L_{s+1}(\beta)$的假设的算法(或预言器)都可以用于计算$\mathbb{Z}_p$ 上的离散对数。

我们对于 $s=1$ 证明这个结果。更准确地说,如果 $p\equiv 3(\bmod\ 4)$是一个素数,我们将说明,用任何一个计算 $L_{s+1}(\beta)$的预言器如何解$\mathbb{Z}_p$ 上的离散对数。

我们曾讲过,如果 β 是$\mathbb{Z}_p$ 中一个二次剩余,并且 $p\equiv 3(\bmod\ 4)$,那么 β 模 p 的两个平方根是 $\pm\beta^{(p+1)/4}\bmod p$。更重要的是,如果 $\beta\neq 0$,对任何 $p\equiv 3(\bmod\ 4)$,

$$L_1(\beta)\neq L_1(p-\beta)$$

原因如下。假设:

$$\alpha^a\equiv\beta(\bmod\ p)$$

则

$$\alpha^{a+(p-1)/2}\equiv-\beta(\bmod\ p)$$

因为 $p\equiv 3(\bmod\ 4)$,整数$(p-1)/2$ 是一个奇数,结果得证。

现在假设对于某个(未知的)偶指数 a,有 $\beta=\alpha^a$。那么或者

$$\beta^{(p+1)/4}\equiv\alpha^{a/2}(\bmod\ p)$$

或者

$$-\beta^{(p+1)/4}\equiv\alpha^{a/2}(\bmod\ p)$$

如果知道 $L_2(\beta)$,我们就可以确定这两种可能性中哪种是正确的。这是因为:

$$L_2(\beta)=L_1(\alpha^{a/2})$$

这个事实由我们的算法 6.6 所揭示。

算法 6.6　L_2OracleDiscreteLogarithm(p,α,β)

external L_1, OracleL_2

$x\leftarrow L_1(\beta)$

$\beta\leftarrow\beta/\alpha^{x_0}\bmod p$

$i\leftarrow 1$

while $\beta\neq 1$

do $\begin{cases} x_i \leftarrow \text{OracleL}_2(\beta) \\ \gamma \leftarrow \beta^{(p+1)/4} \bmod p \\ \textbf{if } L_1(\gamma) = x_i \\ \quad \textbf{then } \beta \leftarrow \gamma \\ \quad \textbf{else } \beta \leftarrow p - \gamma \\ \beta \leftarrow \beta / \alpha^{x_i} \bmod p \\ i \leftarrow i + 1 \end{cases}$

return$(x_{i-1}, x_{i-2}, \ldots, x_0)$

在算法 6.6 的最后,这些 x_i 构成了 $\log_\alpha \beta$ 的二进制表示的各个比特;即

$$\log_\alpha \beta = \sum_{i \geqslant 0} x_i 2^i$$

我们用一个小例子说明这个算法。

例 6.10 假设 $p = 19, \alpha = 2$ 且 $\beta = 6$。因为例子较小,我们对所有的 $\gamma \in \mathbb{Z}_{19}^*$,列出 $L_1(\gamma)$ 和 $L_2(\gamma)$的值(一般地,L_1 可以由 Euler 判别法有效的计算出来,L_2 用假设算法 ORACLEL$_2$ 计算)。这些值由表 6.2 给出。算法 6.6 的过程如图 6.2 所示。

表 6.2 对于 $p = 19, \alpha = 2$, L_1 和 L_2 的值

γ	$L_1(\gamma)$	$L_2(\gamma)$	γ	$L_1(\gamma)$	$L_2(\gamma)$	γ	$L_1(\gamma)$	$L_2(\gamma)$
1	0	0	7	0	1	13	1	0
2	1	0	8	1	1	14	1	1
3	1	0	9	0	0	15	1	1
4	0	1	10	1	0	16	0	0
5	0	0	11	0	0	17	0	1
6	0	1	12	1	1	18	1	0

$x_0 \leftarrow 0, \beta \leftarrow 6, i \leftarrow 1$

$x_1 \leftarrow L_2(6) = 1, \gamma \leftarrow 5, L_1(5) = 0 \neq x_1, \beta \leftarrow 14, \beta \leftarrow 7, i \leftarrow 2$

$x_2 \leftarrow L_2(7) = 1, \gamma \leftarrow 11, L_1(11) = 0 \neq x_2, \beta \leftarrow 8, \beta \leftarrow 4, i \leftarrow 3$

$x_3 \leftarrow L_2(4) = 1, \gamma \leftarrow 17, L_1(17) = 0 \neq x_3, \beta \leftarrow 2, \beta \leftarrow 1, i \leftarrow 4$

return$(1,1,1,0)$

图 6.2 利用对 L_2 的预言器计算$\mathbb{Z}_{19}^*$中的 $\log_2 6$

结果是 $\log_2 6 = 1110_2 = 14$,这很容易验证。

利用数学归纳法可以给出算法的一个正式的证明。记

$$x = \log_\alpha \beta = \sum_{i \geqslant 0} x_i 2^i$$

对于 $i \geqslant 0$,定义

$$Y_i = \left\lfloor \frac{x}{2^{i+1}} \right\rfloor$$

再者,定义 β_0 是到 **while** 循环开始时 β 的值; 对 $i \geqslant 1$,定义 β_i 是 **while** 循环第 i 次运行结束时 β 的值。利用归纳法可以证明,对 $i \geqslant 0$,

$$\beta_i \equiv \alpha^{2Y_i} \pmod{p}$$

可以看出

$$2Y_i = Y_{i-1} - x_i$$

说明

$$x_{i+1} = L_2(\beta_i)$$

$i \geqslant 0$。因为

$$x_0 = L_1(\beta)$$

所以算法是正确的。具体细节留给读者。

6.7.2　ElGamal 体制的语义安全性

我们首先来看,正如密码体制 6.1 所述,ElGamal 密码体制不是语义安全的。我们回顾一下这类体制,$\alpha \in \mathbb{Z}_p^*$ 是一个本原元,$\beta = \alpha^a \bmod p$,a 是私钥。给定一个明文 x,随机选取数 k,计算 $e_K(x,k) = (y_1, y_2)$。这里 $y_1 = \alpha^k \bmod p$ 且 $y_2 = x\beta^k \bmod p$。

我们利用这个事实: 用 Euler 判别法容易判定 $\mathbb{Z}_p$ 的元素是否是模 p 的二次剩余。由 6.7.1 小节我们知道,β 是一个模 p 的二次剩余当且仅当 a 是偶数。类似地,y_1 是一个模 p 的二次剩余当且仅当 k 是偶数。我们可以确定 a 和 k 的奇偶性,因而可以计算 ak 的奇偶性。所以,我们可以确定是否 $\beta^k (= \alpha^{ak})$是一个二次剩余。

现在,假设要区分 x_1 的加密与 x_2 的加密,这里 x_1 是二次剩余,x_2 不是模 p 二次剩余。确定 y_2 的二次剩余性是很简单的事情,我们已经讨论了确定 β^k 的二次剩余性的判定方法。那么,(y_1, y_2)是 x_1 的加密当且仅当 β^k 和 y_2 二者同为二次剩余或者同为非二次剩余。

如果 β 是一个二次剩余,每个明文 x 也要求均为二次剩余,上述攻击就失效了。事实上,如果 $p = 2q + 1$,q 是素数,可以证明,限制 β,y_1 和 x 为二次剩余等价于在模 p 的二次剩余子群中实现 ElGamal 密码体制($\mathbb{Z}_p^*$ 的这个子群是一个 q 阶循环子群)。如果$\mathbb{Z}_p^*$ 中离散对数问题是难解的,这个版本的 ElGamal 密码体制被猜想是语义安全的。

6.7.3　Diffie-Hellman 问题

现在我们介绍所谓 Diffie-Hellman 问题的两个变形,一个计算形式和一个判定形式。之所

以称为 Diffie-Hellman 问题,是因为这两个问题源于与 Diffie-Hellman 密钥协定协议的联系。本小节我们讨论这些问题与 ElGamal 型密码体制的安全性之间的有趣联系。

这两个问题描述如下。

问题 6.3 计算式 Diffie-Hellman

条件: 一个乘法群$(G,\cdot)$,一个 n 阶元素$\alpha\in G$,两个元素 $\beta,\gamma\in\langle\alpha\rangle$。

问题: 找出 $\delta\in\langle\alpha\rangle$,满足 $\log_\alpha\delta\equiv\log_\alpha\beta\times\log_\alpha\gamma(\mathrm{mod}\ n)$。(等价于,给定 α^b 和α^c,找出 α^{bc}。)

问题 6.4 判定式 Diffie-Hellman

条件: 一个乘法群$(G,\cdot)$,一个 n 阶元素$\alpha\in G$,三个元素 $\beta,\gamma,\delta\in\langle\alpha\rangle$。

问题: 是否有 $\log_\alpha\delta\equiv\log_\alpha\beta\times\log_\alpha\gamma(\mathrm{mod}\ n)$?(等价于,给定 α^b,α^c 和α^b,判定是否 $d\equiv bc\ (\mathrm{mod}\ n)$。)

容易看出,存在图灵归约:

$$\text{判定式 Diffie-Hellman}\propto_T\text{计算式 Diffie-Hellman}$$

且

$$\text{计算式 Diffie-Hellman}\propto_T\text{离散对数}$$

第一个归约证明如下:给定 $\alpha,\beta,\gamma,\delta$。利用一个算法来解计算式 Diffie-Hellman ,找到值 δ'满足:

$$\log_\alpha\delta'\equiv\log_\alpha\beta\times\log_\alpha\gamma(\mathrm{mod}\ p)$$

然后,检查是否有 $\delta=\delta'$

第二个归约也非常简单。给定 α,β,γ,利用一个算法来解离散对数,找到 $b=\log_\alpha\beta$ 和$c=\log_\alpha\gamma$。然后计算 $d\equiv bc\ \mathrm{mod}\ n$ 和$\delta=\alpha^d$。

这些归约说明,假设判定式 Diffie-Hellman 难解与假设计算式 Diffie-Hellman 难解至少有着相同的强度。而后者与离散对数是难解的假设至少有着相同的强度。

不难证明,ElGamal 密码体制的语义安全性等价于判定式 Diffie-Hellman 的难解性;ElGamal 解密(在不知道私钥时)等价于解计算式 Diffie-Hellman。因此,要证明 ElGamal 密码体制的安全性所必须的假设(潜在的)强于仅假设离散对数是难解的。的确,我们已经证明,$\mathbb{Z}_p^*$ 中的 ElGamal 密码体制不是语义安全的,而对适当选取的素数 p,离散对数问题被猜想是在$\mathbb{Z}_p^*$ 中难以处理的。这暗示着三个问题的安全性可能不等价。

这里,我们给出一个证明:任何解计算式 Diffie-Hellman 的算法,都可以用于解密 ElGamal 密文,反之亦然。假设 OracleCDH 是计算式 Diffie-Hellman 的一个算法,设(y_1,y_2)是 ElGamal 密码体制的密文,具有公钥 α 和β。计算

$$\delta = \text{OracalCDH}(\alpha, \beta, y_1)$$

然后定义：

$$x = y_2 \delta^{-1}$$

容易看出，x 是密文(y_1, y_2)的解密。

反过来，假设 OracleElGamalDecrypt 是解密 ElGamal 密文的一个算法。设 α, β, γ 如 计算式 Diffie-Hellman 中的假设给定。定义 α 和 β 是 ElGamal 密码体制的公钥。那么，定义 $y_1 = \gamma$，令 $y_2 \in \langle \alpha \rangle$为随机取定。计算

$$x = \text{OracleElGamalDecrypt}(\alpha, \beta, (y_1, y_2))$$

这是密文(y_1, y_2)的解密。最后，计算

$$\delta = y_2 x^{-1}$$

δ 是计算式 Diffie-Hellman 给定条件的解。

6.8　注释与参考文献

ElGamal 密码体制在[70]中提出。Pohlig-Hellman 算法发表于[168]。关于一般的离散对数问题，我们推荐 Odlyzko 的近期的综述性文章[159]。

Pollard ρ 算法首次发表于[170]。Brent[38]描述了一个更有效的方法检测循环(也就是碰撞)，这也可以用于对应的分解算法中。算法中使用的"随机行走"有许多定义方式，可参见 Teske [208]。

离散对数问题通用算法的下界由 Nechaev [152] 和 Shoup [192] 独立证明出来。我们的讨论基于 Chteauneuf、Ling 和 Stingson [46]。

关于有限域的主要参考书是 Lidl 和 Niederreiter [134]。Mcelice [14] 是这方面一本好的初等教科书。

关于$\mathbb{F}_{2^n}$上离散对数问题的很有参考价值的文章是 Gorden 和 McCurley [98]。一些较新的结论可以在 Thomé 的文章 [209] 中找到。Lenstra 和 Verheul 在[133]中预言了未来几年的离散对数问题的难解性，以及基于离散对数密码体制的密钥长度的不同选取。

在公钥密码体制中使用椭圆曲线的思想来自于 Koblitz [119] 和 Miller [148]。Koblitz, Menezes 和 Vanstone [122] 是关于这个主题的综述性文章。两个最近的有关椭圆曲线编码的论著是 Blake, Seroussi 和 Smart [26] 以及 Enge [71]。对于椭圆曲线的更基本的处理参见 Silverman 和 Tate [194]。

最近的完整讨论椭圆曲线快速运算的文章是 Solinas [201]。离散对数从椭圆曲线到有限域的 Menezes-Okamoto-Vanstone 归约在[143]中给出(也参见[142])。对于"迹一"曲线的攻击可参见 Smart、Satoh、Araki 和 Semaev 的相关著作，例如 Smart [198]。

我们给出的关于离散对数第 i 位问题的资料基于 Peralta [163]。讨论相关问题的其他文章有 Hastad、Schtift 和 Shamir [101] 以及 Long 和 Wigderson [135]。

Boneh [30] 是关于判定式 Diffie-Hellman 问题的有趣的综述性文章。Maurer 和 Wolf [40]

是6.7节所考虑的更进一步主题。

练习

6.1 实现计算$\mathbb{Z}_p^*$上的离散对数的Shanks算法,p是素数,α是一个本原元。用你的程序计算$\mathbb{Z}_{24\,691}^*$中的$\log_{106}12\,375$和$\mathbb{Z}_{458\,009}^*$中的$\log_6 248\,388$。

6.2 改进Shanks算法,使其可以事先指定β对基α在G中的对数位于区间$[s,t]$中,s,t是满足$0 \leqslant s < t < n$的整数,n是α的阶数。证明改进的算法的正确性,并且其计算复杂度是$O(\sqrt{t-s})$。

6.3 整数$p = 458\,009$是素数,$\alpha = 2$在$\mathbb{Z}_p^*$中的阶数是57 251。利用Pollard ρ算法计算$\beta = 56\,851$对于基α在$\mathbb{Z}_p^*$上的离散对数。取初始值$x_0 = 1$,如例6.3中同样定义划分(s_1, s_2, s_3)。找出满足$x_i = x_{2i}$的最小的整数i,计算要求的离散对数.

6.4 假设p是一个奇素数,k是一个正整数。$\mathbb{Z}_{p^k}^*$是一个$p^{k-1}(p-1)$阶乘法群,并且已知是循环群。它的任何生成元称为模p^k的本原元素。

(a) 假设α是一个模p的本原元。证明α和$\alpha + p$至少有一个是模p^2的本原元。

(b) 说明如何有效地验证3是一个模29和模29^2的本原根。注意,可以证明,如果是模p和模p^2的一个本原根,那么对于所有正整数k,它一定是模p^k的本原根(这个事实不必证明)。因此说明对于所有k,3是模29^k的本原元。

(c) 找出模29的本原根,并且不是模29^2的本原根α。

(d) 利用Pohlig-Hellman算法计算3344对于基3在乘法群$\mathbb{Z}_{24\,389}^*$中的离散对数。

6.5 实现计算$\mathbb{Z}_p$上的离散对数的Pohlig-Hellman算法,p是素数,α是一个本原元。用你的程序计算$\mathbb{Z}_{28\,703}$中的$\log_5 8563$和$\mathbb{Z}_{31\,153}$中的$\log_{10} 12\,611$。

6.6 令$p = 277$。元素$\alpha = 2$是$\mathbb{Z}_p^*$中的本原元。

(a) 计算$\alpha^{32}, \alpha^{40}, \alpha^{59}$和$\alpha^{156}$。在因子基$\{2,3,5,7,11\}$上分解它们。

(b) 利用$\log 2 = 1$,从上述分解结果计算$\log 3, \log 5, \log 7$和$\log 11$(这些都是$\mathbb{Z}_p^*$上对于基α的离散对数)。

(c) 假如我们要计算$\log 173$。用"随机数"2^{177}乘以173并在因子基上分解其结果,然后利用前面计算的因子基中数的对数,计算$\log 173$。

6.7 设$n = pq$是一个RSA模(即,p和q是不同的奇素数),$\alpha \in \mathbb{Z}_n^*$。对于正整数$m$以及$\alpha \in \mathbb{Z}_m^*$,定义$\mathrm{ord}_m(\alpha)$是$\alpha$在$\mathbb{Z}_m^*$中的阶数。

(a) 证明:

$$\mathrm{ord}_n(\alpha) = \mathrm{lcm}(\mathrm{ord}_p(\alpha), \mathrm{ord}_q(\alpha))$$

(b) 假设$\gcd(p-1, q-1) = d$。证明存在元素$\alpha \in \mathbb{Z}_n^*$满足:

$$\mathrm{ord}_n(\alpha) = \frac{\phi(n)}{d}$$

(c) 假设$\gcd(p-1, q-1) = 2$。并且我们有一个预言器来解子群$\langle\alpha\rangle$上的离散对数问题,这里$\alpha \in \mathbb{Z}_n^*$的阶数为$\phi(n)/2$。即,给定任何$\beta \in \langle\alpha\rangle$,预言器将计算离散

对数 $a = \log_\alpha \beta$，其中 $0 \leqslant a \leqslant \phi(n)/2 - 1$（但值 $\phi(n)/2$ 是保密的）。假设我们计算值 $\beta = \alpha^n \bmod n$，然后用预言器计算 $a = \log_\alpha \beta$。假设 $p > 3$ 和 $q > 3$，证明 $n - a = \phi(n)$。

(d) 由(c)给定离散对数 $a = \log_\alpha \beta$，说明如何容易地分解 n。

6.8　这个问题中，我们考虑 $(\mathbb{Z}_{19}, +)$ 上离散对数问题的通用算法。

(a) 假设集合 C 定义如下：

$$C = \{(1 - i^2 \bmod 19, i \bmod 19) : i = 0,1,2,4,7,12\}$$

计算 Good(C)。

(b) 给定 C 中有序对，假设群预言器的输出如下：

$$(0,1) \mapsto 10111$$
$$(1,0) \mapsto 01100$$
$$(16,2) \mapsto 00110$$
$$(4,4) \mapsto 01010$$
$$(9,7) \mapsto 00100$$
$$(9,12) \mapsto 11001$$

其中群元素编码为二进制 5 元组。对于“a”的值能做出什么说明性的结论呢？

6.9　解密表 6.3 中的 ElGamal 密文。体制中的参数是 $p = 31\,847$，$\alpha = 5$，$a = 7899$ 和 $\beta = 18\,074$。$\mathbb{Z}_n$ 的每个元素像练习 5.12 中一样代表三个字母。

表 6.3　ElGamal 密文

(3781,14409)	(31552,3930)	(27214,15442)	(5809,30274)
(5400,31486)	(19936,721)	(27765,29284)	(29820,7710)
(31590,26470)	(3781,14409)	(15898,30844)	(19048,12914)
(16160,3129)	(301,17252)	(24689,7776)	(28856,15720)
(30555,24611)	(20501,2922)	(13659,5015)	(5740,31233)
(1616,14170)	(4294,2307)	(2320,29174)	(3036,20132)
(14130,22010)	(25910,19663)	(19557,10145)	(18899,27609)
(26004,25056)	(5400,31486)	(9526,3019)	(12962,15189)
(29538,5408)	(3149,7400)	(9396,3058)	(27149,20535)
(1777,8737)	(26117,14251)	(7129,18195)	(25302,10248)
(23258,3468)	(26052,20545)	(21958,5713)	(346,31194)
(8836,25898)	(8794,17358)	(1777,8737)	(25038,12483)
(10422,5552)	(1777,8737)	(3780,16360)	(11685,133)
(25115,10840)	(14130,22010)	(16081,16414)	(28580,20845)
(23418,22058)	(24139,9580)	(173,17075)	(2016,18131)
(19886,22344)	(21600,25505)	(27119,19921)	(23312,16906)
(21563,7891)	(28250,21321)	(28327,19237)	(15313,28649)
(24271,8480)	(26592,25457)	(9660,7939)	(10267,20623)

(30499,14423)	(5839,24179)	(12846,6598)	(9284,27858)
(24875,17641)	(1777,8737)	(18825,19671)	(31306,11929)
(3576,4630)	(26664,27572)	(27011,29164)	(22763,8992)
(3149,7400)	(8951,29435)	(2059,3977)	(16258,30341)
(21541, 19004)	(5865,29526)	(10536,6941)	(1777,8737)
(17561,11884)	(2209,6107)	(10422,5552)	(19371,21005)
(26521,5803)	(14884,14280)	(4328,8635)	(28250,21321)
(28327,19237)	(15313,28649)		

明文选自 Michael Ondaatjes 所著的《英国病人(The English Patient)》。

6.10 确定下列哪些多项式在$\mathbb{Z}_2[x]$上是不可约的:x^5+x^4+1,x^5+x^3+1,$x^5+x^4+x^2+1$。

6.11 域$\mathbb{F}_{2^5}$可以由$\mathbb{Z}_2[x]/(x^5+x^2+1)$构造得到。在域中进行下面的计算:

(a) 计算$(x^4+x^2)\times(x^3+x+1)$。

(b) 利用扩展的欧几里德算法计算$(x^3+x^2)^{-1}$。

(c) 利用平方-乘算法计算x^{25}。

6.12 我们给出一个在$\mathbb{F}_{3^3}$中实现 ElGamal 密码体制的例子。多项式x^3+2x^2+1是$\mathbb{Z}_3[x]$上的不可约多项式,所以$\mathbb{Z}_3[x]/(x^3+2x^2+1)$是域$\mathbb{F}_{3^3}$。我们将 26 个字母与域的元素关联起来,这样就可以方便地加密普通文献。使用多项式的字典顺序建立这个对应。具体对应如下:

A $\leftrightarrow 1$	B $\leftrightarrow 2$	C $\leftrightarrow x$
D $\leftrightarrow x+1$	E $\leftrightarrow x+2$	F $\leftrightarrow 2x$
G $\leftrightarrow 2x+1$	H $\leftrightarrow 2x+2$	I $\leftrightarrow x^2$
J $\leftrightarrow x^2+1$	K $\leftrightarrow x^2+2$	L $\leftrightarrow x^2+x$
M $\leftrightarrow x^2+x+1$	N $\leftrightarrow x^2+x+2$	O $\leftrightarrow x^2+2x$
P $\leftrightarrow x^2+2x+1$	Q $\leftrightarrow x^2+2x+2$	R $\leftrightarrow 2x^2$
S $\leftrightarrow 2x^2+1$	T $\leftrightarrow 2x^2+2$	U $\leftrightarrow 2x^2+x$
V $\leftrightarrow 2x^2+x+1$	W $\leftrightarrow 2x^2+x+2$	X $\leftrightarrow 2x^2+2x$
Y $\leftrightarrow 2x^2+2x+1$	Z $\leftrightarrow 2x^2+2x+2$	

假设 Bob 在 ElGamal 密码体制中使用$\alpha=x$,$a=11$;则$\beta=x+2$。说明 Bob 如何解密下列密文串:

(K,H) (P,X) (N,K) (H,R) (T,F) (V,Y) (E,H) (F,A) (T,W) (J,D) (U,J)

6.13 设 E 是定义在$\mathbb{Z}_{71}$上的椭圆曲线$y^2=x^3+x+28$。

(a) 确定 E 上的点的个数。

(b) 证明 E 不是循环群。

(c) E 中元素的最高阶数是多少?找出一个具有这个阶数的元素。

6.14 假设$p>3$是一个奇素数,且$a,b\in\mathbb{Z}_p$。进一步假设方程$x^3+ax+b\equiv 0(\bmod p)$在$\mathbb{Z}_p$中具有三个不同的根。证明定义的椭圆曲线群$(E,+)$不是循环群。

提示：证明阶为 2 的元素生成$(E,+)$的一个同构于$\mathbb{Z}\times\mathbb{Z}_2$的子群。

6.15　考虑椭圆曲线 $E: y^2 \equiv x^3 + ax + b \pmod p$，其中 $4a^3 + 27b^2 \not\equiv 0 \pmod p$，$p > 3$ 是一个素数。

(a) 很显然点 $P = (y_1, y_2) \in E$ 具有阶数 3，当且仅当 $2P = -P$。利用这个事实，证明，如果 $P = (y_1, y_2) \in E$ 具有阶数 3，则有：

$$3x_1^4 + 6ax_1^2 + 12x_1 b - a^2 \equiv 0 \pmod p \tag{6.7}$$

(b) 从式(6.7)推出结论：E 上至多有 8 个阶数为 3 的点。

(c) 利用式(6.7)，确定椭圆曲线 $y^2 \equiv x^3 + 34x \pmod{73}$上所有阶数为 3 的点。

6.16　令 E 是一个定义在$\mathbb{Z}_p$ 上的椭圆曲线，$p > 3$ 是一个奇素数。设 $\#E$ 是素数，$P \in E$，且 $P \neq \mathcal{O}$。

(a) 证明离散对数 $\log_P(-P) = \#E - 1$。

(b) 如何利用 $\#E$ 上的 Hasse 界以及 Shanks 算法的改进，在 $O(p^{1/4})$时间内计算 $\#E$。给出算法的一个伪码描述。

6.17　椭圆曲线 $E: y^2 = x^3 + 2x + 7$ 定义在$\mathbb{Z}_{31}$上。可以证明 $\#E = 39$，$P = (2,9)$是 E 中阶数为 39 的点。简化的 ECIES 定义在 E 上，以$\mathbb{Z}_{31}^*$为其明文空间。假如私钥是 $m = 8$。

(a) 计算 $Q = mP$。

(b) 解密下述密文串：

$$((18,1),21),((3,1),18),((17,0),19),((28,0),8)$$

(c) 假设每个明文代表一个字母，将明文转换为英语单词(这里使用对应：A↔1，…，Z↔26，因为 0 不允许在(明文)有序对中出现)。

6.18　(a) 确定整数 87 的 NAF 表示。

(b) 点 $P = (2,6)$是定义在$\mathbb{Z}_{127}$上的椭圆曲线 $y^2 = x^3 + x + 26$ 上的一个点，利用 87 的 NAF 表示，应用算法 6.5 计算 $87P$。给出每次运行所得到的部分结果。

6.19　令$\mathcal{L}_i$ 代表 NAF 表示中恰有 i 个系数，并且首系数是 1 的所有正整数的集合。记 $k_i = |\mathcal{L}_i|$。

(a) 通过对$\mathcal{L}_i$ 进行适当的分解，证明 k_i 满足下述递推关系：

$$k_1 = 1$$

$$k_2 = 1$$

$$k_{i+1} = 2(k_1 + k_2 + \cdots + k_{i-1}) + 1(\text{对于 } i \geqslant 2)$$

(b) 导出 k_i 的一个二阶递归关系，算出递归关系的一个显式解。

6.20　假定对于 $\beta = 25,219$ 和 841，$L_2(\beta) = 1$，对于 $\beta = 163,532,625$ 和 656，$L_2(\beta) = 0$，利用算法 6.6 计算$\mathbb{Z}_{1103}$中的 $\log_5 896$。

6.21　本问题中，假设 $p \equiv 5 \pmod 8$是素数，a 是模 p 的一个二次剩余。

(a) 证明 $a^{(p-1)/4} \equiv \pm 1 \pmod p$。

(b) 如果 $a^{(p-1)/4} \equiv 1 \pmod p$，证明 $a^{(p+3)/8} \bmod p$ 是 a 模 p 的一个平方根。

(c) 如果 $a^{(p-1)/4} \equiv -1 \pmod p$，证明 $2^{-1}(4a)^{(p+3)/8} \bmod p$ 是 a 模 p 的一个平方根。

提示：利用事实：$p \equiv 5 \pmod 8$是素数时，$\left(\frac{2}{p}\right) = -1$。

(d) 给定本原元 $\alpha \in \mathbb{Z}_p^*$，对任意的 $\beta \in \mathbb{Z}_p^*$，证明 $L_2(\beta)$可以有效地计算。

提示：利用事实：模 p 的平方根是可以计算的，以及当 $p \equiv 5 \pmod 8$是素数时，对于所有的 $\beta \in \mathbb{Z}_p^*$，有 $L_1(\beta) = L_1(p-\beta)$。

6.22 ElGamal 密码体制可以在有限乘法群$(G, \cdot)$的任何子群$\langle \alpha \rangle$中实现，方法如下：设 $\beta \in \langle \alpha \rangle$，定义$(\alpha, \beta)$是公钥。明文空间是$\mathcal{P} = \langle \alpha \rangle$，加密运算为 $e_K(x) = (y_1, y_2) = (\alpha^k, x \cdot \beta^k)$，其中 k 是随机数。

我们要证明区分两个明文的 ElGamal 加密可以图灵归约到判定式 Diffie-Hellman，反之亦然。

(a) 设 OracleDDH 是一个解$(G, \cdot)$中判定式 Diffie-Hellman 的预言器。证明 OracleDDH可以被用做区分两个给定明文 x_1 和 x_2 的 ElGamal 加密的算法的一个子程序(即，给定 $x_1, x_2 \in \mathcal{P}$，一个密文(y_1, y_2)，它是某个 $x_i (i \in \{1,2\})$的加密，区分算法能够确定到底 $i = 1$ 还是 $i = 2$)。

(b) 对于任意的如上所述的于$(G, \cdot)$中实现的 ElGamal 密码体制，假设 OracleDistinguash是能够区分ElGamal密码体制中任意给定的明文 x_1 和 x_2 加密的预言器。再假设OracleDistinguash可以确定(y_1, y_2)是否是 x_1 和 x_2 的一个有效的加密。证明OracleDistinguash 可以用做解$(G, \cdot)$中判定式 Diffie-Hellman 算法的一个子程序。

第7章　签名方案

7.1　引言

这一章，我们将研究签名方案，签名方案又称为数字签名。一个附加在文件上的传统手写签名用来确定需要对该文件负责的某个人。日常生活中经常需要使用签名，例如写信、从银行取钱以及签署合同，等等。

签名方案是一种给以电子形式存储的消息签名的方法。正因为如此，签名之后的消息能通过计算机网络传输。这一章中，我们将研究几种签名方案，但是我们先来讨论一下传统签名与数字签名的一些基本差异。

首先是签署文件的问题。在传统签名模式中，手写签名是所签署文件的物理部分。然而，数字签名没有物理地附加在所签文件上，因此签名算法必须以某种形式将签名“绑”到所签文件上。

第二个问题是签名的验证。一个传统的签名通过比较其他已认证的签名来验证当前签名的真伪。例如，当某人在使用信用卡购物时签名，售货员需要对销售单上的签名与信用卡背面的签名比较以便验证该人的签名。当然，这不是很安全的方法，因为要伪造一个人的签名还是相对容易的。数字签名却能通过一个公开的验证算法对它进行确认。这样，“任何人”都能验证一个数字签名。安全数字签名方案的使用能阻止伪造签名的可能性。

手写签名与数字签名的另一个基本差异是数字签名文件的“拷贝”与原签名文件相同，而仿造的手写签名文件能与原来的签名文件区分开来。这一特征意味着我们必须采取措施防止一个数字签名消息被重复使用。例如，如果 Alice 使用数字签名签署一则消息来授权 Bob 从她的银行帐户上取 100 美元(即支票)，她仅仅让 Bob 取一次。因此，签名文件本身应该包含诸如日期在内的信息以防止该签名被重复使用。

一个数字签名方案包括两个部分：签名算法和验证算法。Alice 能够使用一个(私有的)签名算法 sig 来为消息 x 签名，签名结果 sig (x)随后能使用一个公开的验证算法 ver 得到验证。给定数据对(x, y)，验证算法根据签名是否有效而返回该签名为“真”或“假”的答案。

下面对签名方案做一个正式的定义。

定义 7.1　一个签名方案是一个满足下列条件的五元组$(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$：

1. $\mathcal{P}$是由所有可能的消息组成的一个有限集合。
2. $\mathcal{A}$是由所有可能的签名组成的一个有限集合。
3. $\mathcal{K}$为密钥空间，它是由所有可能的密钥组成的一个有限集合。

4. 对每一个 $K\in\mathcal{K}$,有一个签名算法 $\text{sig}_K\in\mathcal{S}$和一个相应的验证算法 $\text{ver}_K\in\mathcal{V}$。对每一个消息 $x\in\mathcal{P}$和每一个签名 $y\in\mathcal{A}$,每一个 $\text{sig}_K:\mathcal{P}\rightarrow\mathcal{A}$和 $\text{ver}_K:\mathcal{P}\times\mathcal{A}\rightarrow\{\text{true},\text{false}\}$都是满足下列条件的函数:

$$\text{ver}(x,y)=\begin{cases}\text{true} & y=\text{sig}(x)\\ \text{false} & y\neq\text{sig}(x)\end{cases}$$

由 $x\in\mathcal{P}$和 $y\in\mathcal{A}$组成的数据对(x,y)称为签名消息。

对每一个 $K\in\mathcal{K}$,sig_K 和 ver_K 应该是多项式时间函数。ver_K 是公开的函数,而 sig_K 是保密的。给定一个消息 x,除了 Alice 之外,任何人去计算使 $\text{ver}(x,y)=\text{true}$ 的数字签名 y 应该计算上不可行(注意,对给定的 x,可能存在不止一个 y,这要看函数 ver_K 是如何定义的)。如果 Oscar 能够计算出使得 $\text{ver}(x,y)=\text{true}$ 的数据对(x,y),而 x 没有事先被 Alice 签名,则签名 y 称为伪造签名。非正式地,一个伪造的签名是由 Alice 之外的其他人产生的一个有效数字签名。

作为签名方案的第一个例子,我们将 RSA 密码体制用于数字签名中。此时,我们将它称为 RSA 签名方案,见密码体制 7.1。

密码体制 7.1　RSA 签名方案

设 $n=pq$,其中 p 和 q 是素数。设$\mathcal{P}=\mathcal{A}=\mathbb{Z}_n$,并定义

$$\mathcal{K}=\{(n,p,q,a,b):n=pq,p\text{ 和 }q\text{ 为素数},ab\equiv1(\text{mod }\phi(n))\}$$

值 n 和 b 为公钥,值 p,q 和 a 为私钥。对 $K=(n,p,q,a,b)$,定义:

$$\text{sig}_K(x)=x^a \bmod n$$

以及

$$\text{ver}_K(x,y)=\text{true}\Leftrightarrow x\equiv y^b(\text{mod }n)$$

其中 $x,y\in\mathcal{Z}_n$。

因此,Alice 使用 RSA 解密规则 d_K 为消息 x 签名。因为 $d_K=\text{sig}_K$ 是保密的,所以 Alice 是能够产生这一签名的惟一的人。验证算法使用 RSA 加密规则 e_K。任何人都能验证签名,因为 e_k 是公开的。

注意,通过选择任意的 y 和计算 $x=e_K(y)$,任何人都能伪造 Alice 的 RSA 签名,因为 $y=\text{sig}_K(x)$是关于 x 的一个有效签名(然而,首先选择 x,然后计算相应的签名 y 似乎不是一种显而易见的方法;如果能这样做,那么 RSA 密码体制将是不安全的)。阻止这种攻击的一种方法是让消息具备足够的冗余,使得使用这种方法获得的伪造签名对应一个有“意义”的消息 x 的概率非常小。另外,Hash 函数与数字签名结合使用能阻止这种伪造(密码 Hash 函数已在第 4

章做了讨论)。我们将在下一节对这种方法做进一步的讨论。

最后,让我们简要地看看签名与公钥加密是如何结合的。假定 Alice 希望发送一个签名的加密消息给 Bob。给定明文 x,Alice 将计算她的签名 $y = \text{sig}_{\text{Alice}}(x)$,然后使用 Bob 的公开加密函数 e_{Bob}加密 x 和 y,获得 $z = e_{\text{Bob}}(x, y)$。密文 z 将被传送至 Bob。当 Bob 接收到 z 后,他首先使用解密函数 d_{Bob}获得(x, y),然后使用 Alice 的公开验证算法来验证 $\text{ver}_{\text{Alice}}(x, y) = \text{true}$。

如果 Alice 首先加密,然后对加密结果签名会怎样呢?那么她将计算 $z = e_{\text{Bob}}(x)$和 $y = \text{sig}_{\text{Alice}}(z)$。Alice 将把$(z, y)$发送给 Bob,Bob 解密 z,获得 x,然后用 $\text{ver}_{\text{Alice}}$来验证对 x 的签名 y。这种方法一个潜在的问题是,如果 Oscar 获得了(x, y),他能够用他自己的签名 $y' = \text{sig}_{\text{Oscar}}(z)$来替换 Alice 的签名(注意,Oscar 即使在不知道明文的情况下也能对密文 $z = e_{\text{Bob}}(x)$签名)。然后,如果 Oscar 将(z, y')发送给 Bob,Bob 将用 $\text{ver}_{\text{Oscar}}$来验证 Oscar 的签名,Bob 可能由此推断明文 x 来自 Oscar。因为这种潜在的危险,大多数人建议在加密之前签名。

本章余下部分是这样组织的。7.2 节介绍安全签名方案的概念,以及 Hash 函数怎样与签名方案进行结合。7.3 节介绍 ElGamal 签名方案并讨论其安全性。7.4 节介绍从 ElGamal 签名方案派生出的三个重要的签名方案,Schnorr 签名方案、数字签名算法和椭圆曲线数字签名算法。可证明的安全签名方案在 7.5 节中讨论。最后,在 7.6 节和 7.7 节中,我们考虑具有其他属性的签名方案。

7.2 签名方案的安全需求

这一节,我们对什么是安全的签名方案加以讨论。如同密码体制一样,我们需要确定一个攻击模型、攻击者的目的以及方案所提供的安全类型。

从 1.2 节可知,攻击模型定义了攻击者可获得的信息。对签名方案来讲,经常考虑下面的攻击模型:

惟密钥攻击(key-only attack)

Oscar 拥有 Alice 的公钥,即验证函数 ver_K。

已知消息攻击(known message attack)

Oscar 拥有一系列以前由 Alice 签名的消息,例如$(x_1, y_1), (x_2, y_2), \cdots$,其中 x_i 是消息而 y_i 是 Alice 对这些消息,的签名(因此,$y_i = \text{sig}_K(x_i), i = 1, 2, \cdots$)。

选择消息攻击(chosen message attack)

Oscar 请求 Alice 对一个消息列表签名。因此他选择消息 $x_1, x_2, \cdots$,并且 Alice 提供对这些消息的签名,它们分别是 $y_i = \text{sig}_K(x_i), i = 1, 2, \cdots$。

我们考虑攻击者可能有以下几种目的:

完全破译(total break)

攻击者能够确定 Alice 的私钥,即签名函数 sig_K。因此他能对任何消息产生有效的签名。

选择性伪造(selective forgery)

攻击者能以某一不可忽略的概率对另外某个人选择的消息产生一个有效的签名。换句话说,如果给攻击者一个消息 x,那么他能(以某种概率)决定签名 y,使得 $\mathrm{ver}_K(x,y)=\mathrm{true}$。该消息不应该是以前 Alice 曾经签名的消息。

存在性伪造(existential forgery)

攻击者至少能够为一则消息产生一个有效的签名。换句话说,攻击者能创建一个数对 (x,y),其中 x 是消息而 $\mathrm{ver}_K(x,y)=\mathrm{true}$。该消息 x 不应该是以前 Alice 曾经签名的消息。

一个签名方案不可能是无条件安全的,因为对一个给定的消息,Oscar 使用验证算法 ver_K 可以尝试所有的 $y\in\mathcal{A}$,直到他发现一个有效的签名。因此,给定足够的时间,Oscar 总能对任何消息伪造 Alice 的签名。这样,如同密码体制一样,我们的目的是找到计算上或可证明安全的签名方案。

注意,上面的定义与我们在 4.4 节中考虑的关于 MAC 的攻击有一些相似。在 MAC 中,不存在像公钥这样的东西,因此谈到惟密钥攻击也就没有意义(当然,一个 MAC 没有可以分开的签名和验证函数)。4.4 节中的攻击是使用选择消息攻击的存在性伪造。

我们用一些基于 RSA 签名方案的例子来解释上面所描述的概念。在 7.1 节中我们看到,通过选择一个签名 y 并计算 x 使得 $\mathrm{ver}_K(x,y)=\mathrm{true}$,Oscar 能生成一个有效的签名。这就是使用惟密钥攻击的存在性伪造。

对 RSA 签名方案的另一种攻击是基于它的乘法特性。假设 $y_1=\mathrm{sig}_K(x_1)$ 和 $y_2=\mathrm{sig}_K(x_2)$ 是任意两个由 Alice 以前签名的消息,那么

$$\mathrm{ver}_K(x_1x_2 \bmod n, y_1y_2 \bmod n)=\mathrm{true}$$

因此 Oscar 能对消息 $x_1x_2 \bmod n$ 产生有效的签名 $y_1y_2 \bmod n$。这是使用已知消息攻击进行存在性伪造的一个例子。

还有另外一种变化。假定 Oscar 要对消息 x 伪造一个签名,其中 x 可能被另外的人选择。对 Oscar 来说很容易找到 $x_1,x_2\in\mathbb{Z}_n$ 使 $x\equiv x_1x_2 \bmod n$。现在假设他请求 Alice 为 x_1 和 x_2 签名,签名结果分别是 y_1 和 y_2。然后,像前面的攻击一样,$y_1y_2 \bmod n$ 是消息 $x=x_1x_2 \bmod n$ 的签名。这是一种利用选择消息攻击的选择性伪造。

7.2.1 签名和 Hash 函数

签名方案几乎总是和一种非常快的公开密码 Hash 函数结合使用。为 Hash 函数 $h:\{0,1\}^*\rightarrow\mathcal{Z}$输入一任意长度的消息,它将返回一特定长度的消息摘要(160 比特是一种流行的选择)。产生的消息摘要用签名方案$(\mathcal{P},\mathcal{A},\mathcal{K},\mathcal{S},\mathcal{V})$签名,其中$\mathcal{Z}\subseteq\mathcal{P}$。Hash 函数和签名方案的使用如图 7.1 所示。

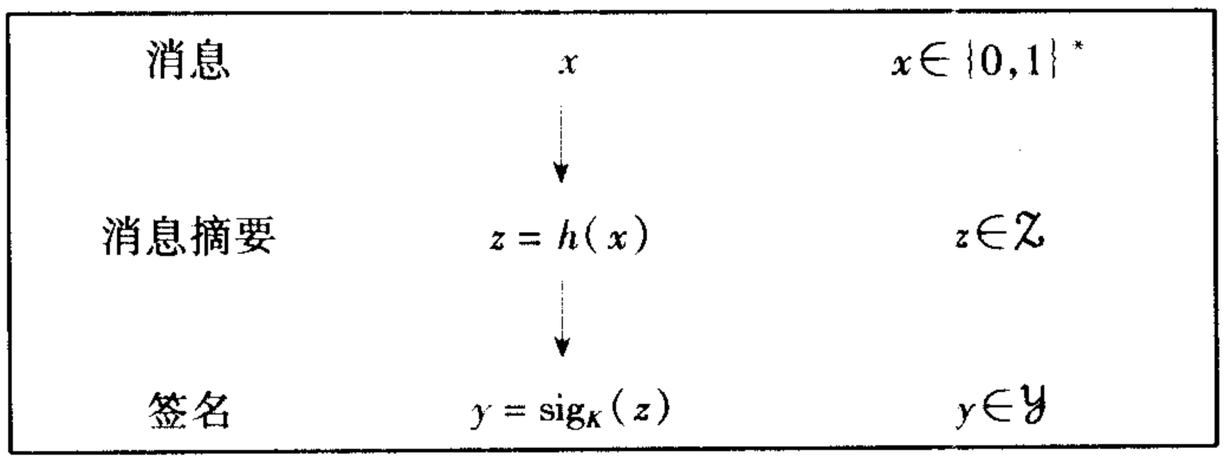

图 7.1 签名一个消息摘要

假设 Alice 要对消息 x 签名，这是一个任意长度的比特串。她首先生成消息摘要 $z = h(x)$，然后计算 z 的签名，即 $y = \mathrm{sig}_K(z)$。然后她将有序对 (x, y) 在信道上传输。验证者首先通过公开 Hash 函数 h 重构消息摘要 $z = h(x)$，然后检查 $\mathrm{ver}_K(z, y) = \mathrm{true}$。

必须认识到 Hash 函数 h 的使用并不削弱签名方案的安全性，因为签名的是消息摘要而非消息本身。有必要使 h 满足一定的属性以便阻止各种各样的攻击。Hash 函数需要具备的属性已经在 4.2 节中做过讨论。

Oscar 最显然的攻击类型是从一个有效的签名消息 (x, y) 开始，其中 $y = \mathrm{sig}_K(h(x))$。（有序对 (x, y) 可以是由 Alice 签名的任何信息。）然后他计算 $z = h(x)$ 并企图找到 $x' \neq x$ 使得 $h(x') = h(x)$。如果 Oscar 能做到这一点，(x', y) 将成为一个有效的签名消息；因此 y 是消息的伪造签名。这是一种使用已知消息攻击的存在性伪造。为了阻止这种攻击，要求 h 是二次原像稳固的（second preimage resistant）。

另外一种攻击如下：Oscar 首先找到两条消息 $x' \neq x$ 使得 $h(x') = h(x)$，然后他将消息发送给 Alice，并让 Alice 对消息摘要 $h(x)$ 签名以获得 y。那么 (x', y) 是有效的签名消息，而 y 是消息 x' 的伪造签名。这是一种利用选择消息攻击的存在性伪造；如果 h 是抗碰撞的，这种攻击就可以避免。

下面介绍第三种攻击。对一个随机的消息摘要 z 伪造签名对某些签名方案来讲是可能的（我们已经在 RSA 签名方案中看到了这种情况）。那就是，我们假定签名方案（没有 Hash 函数）受到了使用惟密钥攻击的存在性伪造。现在，假定 Oscar 要计算某个消息摘要的签名，然后他找到一个消息 x 使得 $z = h(x)$。如果他能做到这一点，那么 (x, y) 是有效的签名消息，而 y 是 x 的伪造签名。这种伪造是该签名方案受到惟密钥攻击的存在性伪造。为了阻止这种攻击，要求 Hash 函数 h 是原像稳固 Hash 函数。

7.3 ElGamal 签名方案

这一节，我们介绍一篇于 1985 年发表的论文中提出的 ElGamal 签名方案，该方案的变型已被美国国家标准技术研究所采纳为数字签名算法（DSA）。DSA 同时吸收了被称为 Schnorr 签名方案的一些思想。与 RSA 密码体制既可以用于公钥又可以用于签名方案不一样，这些方案都是为签名而专门设计的。

ElGamal 签名方案是非确定性的（ElGamal 公钥密码体制也是非确定性的）。这意味着对任何给定的消息有许多有效的签名，并且验证算法能够将它们中的任何一个作为可信的签名而接受。ElGamal 签名方案参见密码体制 7.2。

密码体制 7.2 ElGamal 签名方案

设 p 是一个使得在 $\mathbb{Z}_p$ 上的离散对数问题是难处理的素数,设 $\alpha \in \mathbb{Z}_p^*$ 是一个本原元。设 $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$,定义:

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

值 p, α, β 是公钥,a 是私钥。

对 $K = (p, \alpha, a, \beta)$和一个秘密的随机数 $k \in \mathbb{Z}_{p-1}^*$,定义 $\text{sig}_K(x, k) = (\gamma, \delta)$,其中 $\gamma = \alpha^k \bmod p$, $\delta = (x - a\gamma)k^{-1} \bmod (p-1)$。对 $x, \gamma \in \mathbb{Z}_p^*$ 和 $\delta \in \mathbb{Z}_{p-1}$,定义:

$$\text{ver}_K(x, (\gamma, \delta)) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta = \alpha^x \pmod{p}$$

如果签名被正确地构造出来,那么验证将会成功,因为:

$$\beta_\gamma \gamma_\delta \equiv \alpha^{a\gamma} \alpha^{k\delta} \pmod{p} \equiv \alpha^x \pmod{p}$$

这是基于

$$a\gamma + k\delta \equiv x \pmod{p-1}$$

实际上,从验证公式出发导出签名公式可能更清楚。假设我们从下式开始:

$$\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p} \tag{7.1}$$

然后在 7.1 式中用 $\gamma \equiv \alpha^k \pmod{p}$和 $\beta \equiv \alpha^a \pmod{p}$进行代替,而保留式中的指数 γ,我们得到:

$$\alpha^x \equiv \alpha^{a\gamma + k\delta} \pmod{p}$$

现在,α 是模 p 的本原元,因此,当且仅当指数是一个模 $p-1$ 的等式,即 $x \equiv a\gamma + k\delta \pmod{p-1}$时,上式成立。给定 x, a, γ 和 k,利用这个等式能够求解 δ,得出密码体制 7.2 中签名函数的公式。

Alice 使用私钥 a 和秘密随机数 k(k 用于一则消息 x)来计算签名。仅仅利用公开的信息就能验证该签名。

让我们用一个小例子来解释该算法。

例 7.1 假定选取 $p = 467$, $\alpha = 2$, $a = 127$,那么

$$\begin{aligned} \beta &= \alpha^a \pmod{p} \\ &= 2^{127} \bmod 467 \\ &= 132 \end{aligned}$$

若 Alice 要对消息 $x = 100$ 签名,她取 $k = 213$(注意,$\gcd(213, 466) = 1$ 且 $213^{-1} \bmod 466 = 431$)。那么

$$\gamma = 2^{213} \bmod 467 = 29$$

并且

$$\delta = (100 - 127 \times 29)431 \bmod 466 = 51$$

任何人都可通过计算

$$132^{29}29^{51} \equiv 189 (\bmod 467)$$

和

$$2^{100} \equiv 189 (\bmod 467)$$

来验证这个签名。因此,该签名是有效的。

7.3.1 ElGamal 签名方案的安全性

我们来看看 ElGamal 签名方案的安全性。假定 Oscar 在不知道 a 的情况下想对消息伪造签名。如果他选择一个值 γ,然后试图找出相应的 δ,那么他必须计算离散对数 $\log_\gamma \alpha^x \beta^{-\gamma}$。另一方面,如果他首先选择 δ,然后试图找到 γ,那么他必须"求解"等式:

$$\beta^\gamma \gamma^\delta \equiv \alpha^x (\bmod\ p)$$

以便获得这个"未知"的 γ,这是一个还没有已知可行的办法来求解的方程。然而,它与像离散对数问题这样已研究得比较透彻的问题似乎没有关系。也许仍然存在某种方法可同时计算 γ 和 δ,使得 (γ, δ) 是一个签名。但是,没有人发现解这个问题的方法,也没有人能够证明不能解这个问题。

如果 Oscar 先选择 γ 和 δ,然后去解 x,那么他又一次面临着求解离散对数问题的一个实例,也就是计算 $\log_\alpha \beta^\gamma \gamma^\delta$。因此,使用这种方法 Oscar 不能对给定的消息 x 签名。

然而,通过同时选择 γ,δ 和 x,存在一种方法使 Oscar 能对任意的消息签名。因此,在惟密钥攻击的情况下进行存在性伪造还是可能的(假定没有使用 Hash 函数)。我们现在对此做具体描述。

设 i 和 j 是满足 $0 \leqslant i \leqslant p-2, 0 \leqslant j \leqslant p-2$ 的整数,假设 γ 的表达式为 $\gamma = \alpha^i \beta^j \bmod p$。那么验证条件是:

$$\alpha^x \equiv \beta^\gamma (\alpha^i \beta^j)^\delta (\bmod\ p)$$

这等价于:

$$\alpha^{x - i\delta} \equiv \beta^{\gamma + j\delta} (\bmod\ p)$$

上式在

$$x - i\delta \equiv 0 (\bmod\ p-1)$$

且

$$\gamma + j\delta \equiv 0 (\bmod\ p-1)$$

时成立。

给定 i 和 j,在 $\gcd(j, p-1)=1$ 的条件下,很容易能够利用这两个模 $p-1$ 的等式求出 δ 和 x。我们得到如下的等式:

$$\gamma = \alpha^i \beta^j \bmod p$$
$$\delta = -\gamma j^{-1} \bmod (p-1)$$
$$x = \gamma i j^{-1} \bmod (p-1)$$

显然,按照这种方法构造出来的(γ, δ)是消息 x 的有效签名。下面我们用一个例子加以解释。

例 7.2 在前面的例子中,设 $p=467, \alpha=2, \beta=132$。假定 Oscar 选择 $i=99, j=179$;则 $j^{-1} \bmod (p-1)=151$,他能求出:

$$\gamma = 2^{99} 132^{179} \bmod 467 = 117$$
$$\delta = -117 \times 151 \bmod 466 = 41$$
$$x = 99 \times 41 \bmod 466 = 331$$

那么(117,41)是消息 331 的有效签名,这可从下面的式子得到验证:

$$132^{117} 117^{41} \equiv 303 (\bmod 467)$$
$$2^{331} \equiv 303 (\bmod 467)$$

下面介绍第二种伪造类型,采取这种伪造时,Oscar 从 Alice 已签名的消息开始。它属于已知消息攻击的存在性伪造。假定(γ, δ)是消息 x 的有效签名,那么 Oscar 可利用它给其他的消息签名。设 h, i 和 j 是整数,$0 \leqslant h, i, j \leqslant p-2$,且 $\gcd(h\gamma - j\delta, p-1)=1$。计算

$$\lambda = \gamma^h \alpha^i \beta^j \bmod p$$
$$\mu = \delta \lambda (h\gamma - j\delta)^{-1} \bmod (p-1)$$
$$x' = \lambda (hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1)$$

那么,验证条件

$$\beta^{\lambda} \lambda^{\mu} \equiv \alpha^{x'} \bmod p$$

显然成立。因此,(λ, μ)是 x'的有效签名。

这两种方法都是存在性伪造,但它们似乎还不能被修改成选择性伪造。因此,在使用如 7.2.1 小节中所描述的安全 Hash 函数的情况下,这两种方法似乎对 ElGamal 签名方案的安全性不构成威胁。

在 ElGamal 签名方案使用不仔细的情况下(它们是协议失败进一步的例子,其中一些在第 5 章中讨论过),我们也能提出攻击它的一些方法。首先,在计算签名时所使用的随机值不能泄露。如果被泄露出去,则计算

$$a = (x - k\delta)\gamma^{-1} \bmod (p-1)$$

将是很容易的事情。一旦 a 被泄露,那么系统就完全被破坏了,Oscar 能随意地伪造签名了。

系统的另外一个误用是对两个不同的消息签名时使用相同的 k 值。这将使 Oscar 计算 a 变得容易,因而攻破系统。具体做法如下:设(γ,δ_1)是消息 x_1 的签名,(γ,δ_2)是消息 x_2 的签名,那么我们有:

$$\beta^{\gamma}\gamma^{\delta_1}\equiv\alpha^{x_1}(\bmod\ p)$$

$$\beta^{\gamma}\gamma^{\delta_2}\equiv\alpha^{x_2}(\bmod\ p)$$

因此,

$$\alpha^{x_1-x_2}\equiv\gamma^{\delta_1-\delta_2}(\bmod\ p)$$

令 $\gamma=\alpha^k$,对未知的 k 我们获得如下的等式:

$$\alpha^{x_1-x_2}\equiv\alpha^{k(\delta_1-\delta_2)}(\bmod\ p)$$

该方程等价于:

$$x_1-x_2\equiv k(\delta_1-\delta_2)(\bmod\ p-1)$$

设 $d=\gcd(\delta_1-\delta_2,p-1)$。因为 $d|(p-1)$和 $d|(\delta_1-\delta_2)$,所以有 $d|(x_1-x_2)|$。定义

$$x'=\frac{(x_1-x_2)}{d}$$

$$\delta'=\frac{(\delta_1-\delta_2)}{d}$$

$$p'=\frac{p-1}{d}$$

那么等式变为:

$$x'\equiv k\,\delta'(\bmod\ p')$$

因为 $\gcd(\delta',p')=1$,我们可以计算出:

$$\varepsilon=(\delta')^{-1}\bmod\ p'$$

那么由模 p'决定的 k 值为 $k=x'\varepsilon\bmod\ p'$,这就产生了 d 个候选的 k 值:

$$k=x'\varepsilon+ip'\bmod(p-1)$$

其中 $0\leqslant i\leqslant d-1$。对于 d 个候选的 k 值,可通过等式

$$\gamma\equiv\alpha^k(\bmod\ p)$$

检测出其中惟一正确的那一个。

7.4　ElGamal 签名方案的变型

在许多情况下,一则消息可能仅仅被加密或解密一次,因此,使用当时被认为是安全的密

码体制来加密消息是能够满足需求的。另一方面,一则被签名的消息可能是合同或遗嘱这样的起法律效应的文档,它有可能在消息被签名多年之后仍需验证。因此,相对于密码体制而言,在签名方案的安全方面采取更多的措施是很重要的。因为 ElGamal 签名方案不比离散对数问题更安全,这就要求使用一个大的模 p。为了满足当前的安全性需求,大多数人认为 p 的长度至少需要 1024 比特,在可预见的将来甚至更大(这在 6.6 节中已经提及)。

一个 1024 比特的模导致 ElGamal 签名有 2048 比特。对其中许多都包括智能卡的使用的潜在应用而言,需要的是短的签名。在 1989 年,Schnorr 提出了一种可看做是 ElGamal 签名方案的变型的一种签名方案,其签名的长度被大大地缩短了。数字签名算法(DSA)是 ElGamal 签名方案的另一种变型,它吸收了 Schnorr 签名方案的一些思想。DSA 于 1994 年 5 月 19 日发表在 Federal Register 上(但它在 1991 年 8 月就被提出来了)。我们将在下面的小节中描述 Schnorr 签名方案、DSA 以及应用于椭圆曲线的 DSA 的变型(称为 ECDSA)。

7.4.1 Schnorr 签名方案

设 p 和 q 是满足 $p-1\equiv 0(\bmod\ q)$ 的两个素数,一般取 $p\approx 2^{1024}$,$q\approx 2^{160}$。Schnorr 签名方案对 ElGamal 签名方案以独特的方式进行了修改,使得长度为 $\log_2 q$ 比特的消息摘要有长度为 $2\log_2 q$ 比特的签名,但是计算是在 $\mathbb{Z}_p$ 上进行的。它是通过工作在 $\mathbb{Z}_p^*$ 的 q 元子群来实现的。该方案所认为的安全性是基于这样的思想:在特定的 $\mathbb{Z}_p^*$ 子群上求解离散对数是安全的(离散对数问题的这种情况在 6.6 节中已经讨论过)。

我们取 α 是模 p 同余 1 的 q 次根。(这样的 α 容易构造:设 α_0 是 $\mathbb{Z}_p$ 上的本原元,定义 $\alpha=\alpha_0^{(p-1)/q}\bmod p$)。Schnorr 签名方案中密钥的其他方面与 ElGamal 签名方案中的类似。然而,Schnorr 签名方案将 Hash 函数直接集成到了签名算法当中(与 7.2.1 小节中讨论的先 Hash 后签名的方法不同)。我们假定 $h:\{0,1\}^*\rightarrow\mathbb{Z}_q$ 是安全的 Hash 函数。下面是 Schnorr 签名方案的完整描述。

密码体制 7.3 Schnorr 签名方案

设 p 是使得 $\mathbb{Z}_p^*$ 上离散对数问题难处理的一个素数,q 是能被 $p-1$ 整除的素数。设 $\alpha\in\mathbb{Z}_p^*$ 是模 p 同余 1 的 q 次根,$\mathcal{P}=\{0,1\}^*$,$\mathcal{A}=\mathbb{Z}_q\times\mathbb{Z}_q$,并定义

$$\mathcal{K}=\{(p,q,\alpha,a,\beta):\beta\equiv\alpha^a(\bmod\ p)\}$$

其中 $1\leqslant a\leqslant q-1$,值 p,q,α 和 β 是公钥,a 为私钥。最后,设 $h:\{0,1\}^*\rightarrow\mathbb{Z}_q$ 是安全的 Hash 函数。

对于 $K=(p,q,\alpha,a,\beta)$ 和一个秘密的随机数 k,$1\leqslant k\leqslant q-1$,定义

$$\mathrm{sig}_K(x,k)=(\gamma,\delta)$$

其中 $\gamma=h(x\parallel\alpha^k)$ 且 $\delta=k+a\gamma\bmod q$。

对于 $x\in\{0,1\}^*$ 和 $\gamma,\delta\in\mathbb{Z}_q$,验证是通过下面的计算完成的:

$$\mathrm{ver}_K(x,(\gamma,\delta))=\mathrm{true}\Leftrightarrow h(x\parallel\alpha^{\delta}\beta^{-\gamma})=\gamma$$

容易检验 $\alpha^{\delta}\beta^{-\gamma}\equiv\alpha^{k}(\mathrm{mod}\ p)$，因此也就验证了 Schnorr 签名。下面用一个小例子加以说明。

例 7.3　假设我们取 $q=101$，$p=78q+1=7879$。3 是$\mathbb{Z}_{7879}^{*}$中的一个本原元，因此我们取

$$\alpha=3^{78}\bmod 7879=170$$

α 是模 p 同余 1 的 q 次根。假设 $a=75$，那么，

$$\beta=\alpha^{a}\bmod 7879=4567$$

现在，假定 Alice 要对消息 x 签名，她选择随机值 $k=50$，并首先计算

$$\alpha^{k}\bmod p=170^{50}\bmod 7879=2518$$

下一步计算 $h(x\parallel 2518)$，其中 h 是给定的 Hash 函数，2518 以二进制位串的形式表示。为了解释起见设 $h(x\parallel 2518)=96$，那么 δ 的计算结果为：

$$\delta=50+75\times 96\bmod 101=79$$

因此签名为(96,79)。

通过计算

$$170^{79}\,4567^{-96}\bmod 7879=2518$$

并检查 $h(x\parallel 2518)=96$，该签名即可得到验证。

7.4.2　数字签名算法(DSA)

我们将描述 DSA 签名规范中对 ElGamal 签名方案验证函数所做的修改。如 Schnorr 签名方案一样，DSA 使用了$\mathbb{Z}_p^{*}$ 的一个 q 元子群。在 DSA 中，要求 q 是 160 比特的素数，p 是长为 L 比特的素数，其中 $L\equiv 0(\mathrm{mod}\ 64)$且 $512\leqslant L\leqslant 1024$。DSA 中的密钥与 Schnorr 签名方案中的密钥具有相同的形式。DSA 同时还规定了在消息被签名之前，要用 SHA-1 算法将消息压缩。结果是 160 比特的消息摘要有 320 比特的签名，并且计算是在$\mathbb{Z}_p$ 和$\mathbb{Z}_q$ 上进行的。

在 ElGamal 签名方案中，假设我们在 δ 的定义中将“-”改变成“+”，即

$$\delta=(x+a\gamma)k^{-1}\mathrm{mod}(p-1)$$

容易看出验证条件变成了如下的形式：

$$\alpha^{x}\beta^{\gamma}\equiv\gamma^{\delta}(\mathrm{mod}\ p)\tag{7.2}$$

现在 α 的阶数为 q，β 和 γ 是 α 的幂次方，因此它们的阶数也为 q。这意味者(7.2)式中所有的指数模 q 减小而不影响等式的有效性。因为 DSA 中将被 160 比特的消息摘要所替代，我们假定 $x\in\mathbb{Z}_q$。进一步，为了使 $\delta\in\mathbb{Z}_q$，我们对 δ 的定义改变如下：

$$\delta=(x+a\gamma)k^{-1}\mathrm{mod}\ q$$

仍然考虑 $\gamma \equiv \alpha^k \bmod p$,若我们临时定义

$$\gamma' = \gamma \bmod q \equiv (\alpha^k \bmod p) \bmod q$$

既然,

$$\delta = (x + a\gamma')k^{-1} \bmod q$$

因此 δ 不变。我们将验证公式表示为:

$$\alpha^x \beta^{\gamma'} \equiv \gamma^{\delta} (\bmod p) \tag{7.3}$$

注意,对等式中的其余的 γ 不能用 γ' 来代替。

现在我们继续重写(7.3),将两边同时增大 δ^{-1} 次方并 mod q(这里要求 $\delta \neq 0$)。我们得到下式:

$$\alpha^{x\delta^{-1}} \beta^{\gamma'\delta^{-1}} (\bmod p) \equiv \gamma \tag{7.4}$$

现在我们对(7.4)两边同时模 q,得到下式:

$$(\alpha^{x\delta^{-1}} \beta^{\gamma'\delta^{-1}} \bmod p) \bmod q \equiv \gamma' \tag{7.5}$$

DSA 的完整描述见密码体制 7.4,这里我们将 γ' 用 γ 命名,并用 SHA-1(x)来替换 x。

密码体制 7.4 DSA

设 p 是长 L 比特的素数,在 $\mathbb{Z}_p$ 上其离散对数问题是难处理的,其中 $L = 0 (\bmod 64)$ 且 $512 \leqslant L \leqslant 1024$,$q$ 是能被 $p-1$ 整除的 160 比特的素数。设 $\alpha \in \mathbb{Z}_p^*$ 是 1 模 p 的 q 次根。设 $\mathcal{P} = \{0,1\}^*$,$\mathcal{A} = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$,并定义

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a (\bmod p)\}$$

其中 $1 \leqslant a \leqslant q-1$,值 p, q, α 和 β 是公钥,a 为私钥。对于 $K = (p, q, \alpha, a, \beta)$和一个秘密的随机数 k,$1 \leqslant k \leqslant q-1$,定义 $\mathrm{sig}_K(x, k) = (\gamma, \delta)$,其中

$$\gamma = (\alpha^k \bmod p) \bmod q$$
$$\delta = (\text{SHA-1}(x) + a\gamma)k^{-1} \bmod q$$

(如果 $\gamma = 0$ 或 $\delta = 0$,应该为 k 另选一个随机数)。

对于 $x \in \{0,1\}^*$ 和 $\gamma, \delta \in \mathbb{Z}_q^*$,验证是通过下面的计算完成的:

$$e_1 = \text{SHA-1}(x)\delta^{-1} \bmod q$$
$$e_2 = \gamma\delta^{-1} \bmod q$$
$$\mathrm{ver}_K(x, (\gamma, \delta)) = \text{true} \Leftrightarrow (\alpha^{e_1} \beta^{e_2} \bmod p) \bmod q = \gamma$$

2001 年 10 月,NIST 建议 p 选为 1024 位的素数(即 L 的惟一允许值为 1024)。这“既不是标准,也不是指南”,但确实表示了对离散对数问题安全性的一些担心。

注意,如果 Alice 使用 DSA 签名算法 $\delta \equiv 0 (\mathrm{mod}\ p)$计算出,她应该放弃该 δ,选择一个新的随机数 k 来构造新的δ。我们应该指出实际上不可能出现这种问题:$\delta \equiv 0 (\mathrm{mod}\ p)$的概率大约是 2^{-160},不管出于什么样的目的这种情况几乎不会发生。

下面用一个例子(其中 p 和 q 比 DSA 所要求得要小得多)来加以说明。

例 7.4 假设 p, q, α, a, β 和 k 的取值和例 7.3 相同,假设 Alice 要对消息摘要 SHA-1(x) = 22 签名。然后她计算

$$k^{-1} \bmod 101 = 50^{-1} \bmod 101 = 99$$
$$\gamma = (170^{50} \bmod 7879) \bmod 101$$
$$= 2518 \bmod 101$$
$$= 94$$

且

$$\delta = (22 + 75 \times 94) 99 \bmod 101$$
$$= 97$$

对消息摘要 22 的签名(94,97)通过下面的计算加以验证:

$$\delta^{-1} = 97^{-1} \bmod 101 = 25$$
$$e_1 = 22 \times 25 \bmod 101 = 45$$
$$e_2 = 94 \times 25 \bmod 101 = 27$$
$$(170^{45} 4567^{27} \bmod 7879) \bmod 101 = 2518 \bmod 101 = 94$$

DSA 在 1991 年被提出来时,就受到了一些批评。一种抱怨是 NIST 没有公开数字签名方案的选择过程。这个标准是由美国国家安全局(NSA)制订的,没有美国工业部门的参加。不管所选择的签名方案具有多少优点,许多人还是抱怨这种"关门"的做法。

对 DSA 提出的技术上的批评,最严重的要属最初模 p 的大小固定在 512 比特。许多人建议模数的大小应该是不固定的,需要时可使用更大的模。作为答复,NIST 对算法做了修改,允许使用不同大小的模。

7.4.3 椭圆曲线 DSA

在 2000 年,椭圆曲线数字签名算法(ECDSA)作为 FIPS 186-2 得到了批准。我们在这一小节介绍 ECDSA。

密码体制 7.5 椭圆曲线数字签名算法

设 p 是一个素数或 2 的幂次方,E 是定义在$\mathbb{F}_p$ 上的椭圆曲线。设 A 是E 上阶数为q 的一个点,使得在$\langle A \rangle$上的离散对数问题是难处理的。设$\mathcal{P} = \{0,1\}^*$,$\mathcal{A} = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$,定义

$$\mathcal{K} = \{(p, q, E, A, m, B) : B = mA\}$$

其中 $1 \leqslant m \leqslant q-1$,值 p, q, E 和 A 是公钥,m 为私钥。

对于 $K=(p, q, E, A, m, B)$和一个秘密的随机数 $k, 1 \leqslant k \leqslant q-1$,定义

$$\mathrm{sig}_K(x, k)=(r, s)$$

其中

$$\begin{aligned} kA &= (u, v) \\ r &= u \bmod q \\ s &= k^{-1}(\text{SHA-1}(x) + mr) \bmod q \end{aligned}$$

(如果 $r=0$ 或 $s=0$,应该为 k 另选一个随机数)。

对于 $x \in \{0,1\}^*$ 和 $r, s \in \mathbb{Z}_q^*$,验证是通过下面的计算完成的:

$$\begin{aligned} w &= s^{-1} \bmod q \\ i &= w\ \text{SHA-1}(x) \bmod q \\ j &= wr \pmod q \\ (u, v) &= iA + jB \\ \mathrm{ver}_K(x, (r, s)) &= \text{true} \Leftrightarrow u \bmod q = r \end{aligned}$$

我们通过一个小例子来说明 ECDSA 中的计算。

例 7.5 我们的例子是基于 6.5.2 小节中相同的椭圆曲线,即定义在 $\mathbb{Z}_{11}$ 上的 $y^2 = x^3 + x + 6$。签名方案的参数是 $p=11, q=13, A=(2,7), m=7$ 以及 $B=(7,2)$。

假设我们有消息 x,使得 SHA-1$(x)=4$,并且 Alice 要使用随机数 $k=3$ 为 x 签名。她将计算:

$$\begin{aligned} (u, v) &= 3(2,7) = (8,3) \\ r &= u \bmod 13 = 8 \\ s &= 3^{-1}(4 + 7 \times 8) \bmod 13 = 7 \end{aligned}$$

因此(8,7)是对 x 的签名。

Bob 执行下面的计算来验证签名:

$$\begin{aligned} w &= 7^{-1} \bmod 13 = 2 \\ i &= 2 \times 4 \bmod 13 = 8 \\ j &= 2 \times 8 \bmod 13 = 3 \\ (u, v) &= 8A + 3B = (8,3) \\ u &\bmod 13 = 8 = r \end{aligned}$$

因此,签名得到了验证。

7.5 可证明的安全签名方案

我们在这一节中提出了一些可证明的安全签名方案的例子。首先,我们描述基于任意单向函数 f(即原像稳固)的一次签名方案的构造。这个方案在 f 是一个双射函数的条件下可证明对惟密钥攻击是安全的。全域 Hash 是第二种可证明的安全签名方案的构造方法。这种签名方案如果是从陷门单向置换构造出来的,则它在随机预言模型中是可证明安全的。

7.5.1 一次签名

这一小节中,我们描述一个从单向函数构造一个可证明安全的一次签名方案的概念性的简单方法(如果一个签名方案仅给一则消息签名时是安全的,则该签名方案是一次签名方案。当然,该签名可进行任意次的验证)。该签名方案,又称为 Lamport 签名方案,由密码体制 7.6 给出。

密码体制 7.6 Lamport 签名方案

设 k 是一个正整数且$\mathcal{P}=\{0,1\}^k$。假定 $f: Y \rightarrow Z$ 是一个单向函数,并且$\mathcal{A}=Y^k$。设随机选择的 $y_{i,j}\in Y, 1\leqslant i\leqslant k, j=0,1$。设 $z_{i,j}=f(y_{i,j}), 1\leqslant i\leqslant k, j=0,1$。密钥 K 由 $2k$ 个 y 和 $2k$ 个 z 构成。y 是私钥而 z 是公钥。

对于 $K=(y_{i,j}, z_{i,j}: 1\leqslant i\leqslant k, j=0,1)$,定义

$$\mathrm{sig}_K(x_1,\cdots x_k)=(y_{1,x_1},\cdots,y_{k,x_k})$$

关于消息$(x_1,\cdots,x_k)$的签名$(a_1,\cdots,a_k)$验证如下:

$$\mathrm{ver}_K((x_1,\cdots,x_k),(a_1,\cdots,a_k))=\mathrm{true}\Leftrightarrow f(a_i)=z_{i,x_i}$$

其中 $1\leqslant i\leqslant k$。

非正式地讲,这就是该体制的工作流程。要签名的消息是一个二进制 k 元组。消息的每比特都得单独签名。如果消息的第 i 比特等于j(其中 $j\in\{0,1\}$),则签名的第 i 个元素是 $y_{i,j}$,它是公钥值 $z_{i,j}$的原像。验证就是检查签名的每一个元素对应于消息第 i 比特的公钥元素 $z_{i,j}$ 的原像。这可用公开的函数 f 来完成。

考虑使用指数函数 $f(x)=\alpha^x \bmod p$ 这一可能的方案(其中 α 是模 p 的本原元),我们来说明一下该签名模式。这里,$f: \{0,\cdots,p-2\}\rightarrow\mathbb{Z}_q^*$。我们用一个有趣的例子来说明该方案的计算过程。

例 7.6 7879 是一个素数并且 3 是$\mathbb{Z}_{7879}^*$ 的一个本原元,令

$$f(x) = 3^x \bmod 7879$$

假定 $k = 3$, Alice 选取 6 个秘密的随机数:

$$y_{1,0} = 5831$$
$$y_{1,1} = 735$$
$$y_{2,0} = 803$$
$$y_{2,1} = 2467$$
$$y_{3,0} = 4285$$
$$y_{3,1} = 6449$$

然后 Alice 计算在函数 f 下 6 个 y 值的像:

$$z_{1,0} = 2009$$
$$z_{1,1} = 3810$$
$$z_{2,0} = 4672$$
$$z_{2,1} = 4721$$
$$z_{3,0} = 268$$
$$z_{3,1} = 5731$$

这些 z 是公开的。现在,假设 Alice 要对消息

$$x = (1,1,0)$$

签名,签名结果是:

$$(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285)$$

为了验证该签名,只需计算:

$$3^{735} \bmod 7879 = 3810$$
$$3^{2467} \bmod 7879 = 4721$$
$$3^{4285} \bmod 7879 = 268$$

因此,这个签名得到了验证。

Oscar 不能伪造签名,因为他不能求单向函数 f 的逆以获得密钥 y。然而,该签名方案只能安全地给一则消息签名。在已知两则不同消息的签名的情况下,构造出与这两则消息不同的消息签名对 Oscar 来讲是很容易的事情(除非前两则消息仅有一比特不同)。

例如,假定使用相同的密钥对消息(0,1,1)和(1,0,1)签名,消息(0,1,1)的签名为($y_{1,0}$, $y_{2,1}$, $y_{3,1}$),消息(1,0,1)的签名为($y_{1,1}$, $y_{2,0}$, $y_{3,1}$)。已知这两个签名,Oscar 能对消息(1,1,1)和(0,0,1)构造一个签名(它们分别是($y_{1,1}$, $y_{2,1}$, $y_{3,1}$)及($y_{1,0}$, $y_{2,0}$, $y_{3,1}$))。

如果我们假定 $f: Y \rightarrow Z$ 是一个单向双射函数,且公钥包括 $2k$ 个属于 Z 的不同的元素,Lamport 签名方案的安全性是可证明的。我们考虑惟密钥攻击,这种情况下攻击者只知道公钥。我们假设攻击者能实现存在性伪造,换句话说,给定公钥,攻击者输出一个消息 x 和一个

有效的签名 y(我们假定 f, Y, Z 和 k 是固定的)。

以 Lamport-Forge 算法来建立攻击模型。为了简单起见,我们假设 Lamport-Forge 是确定性的:给定任何特定的公钥,它总能输出同样的伪造签名。我们先描述 Lamport-Preimage 算法,对于随机选择的元素 $z \in Z$,该算法找出关于函数 f 的原像。这个算法是将 Lamport-Forge 算法当成预言的简化算法。这种简化与函数 f 的单向性矛盾。因此,如果我们相信 f 是单向的,则我们得出唯密钥的存在性伪造是计算上不可行的结论。Lamport-Preimage 如算法 7.1 所示。

算法 7.1　Lamport-Preimage(z)

external f, Lamport-Forge

comment:我们假定 $f: Y \rightarrow Z$ 是对射函数

选择随机值 $i_0 \in \{1, \cdots, k\}$ 和随机值 $j_0 \in \{0,1\}$

构造随机公钥 $\mathcal{Z} = (z_{i,j}: 1 \leqslant i \leqslant k, j = 0,1)$,使得 $z_{i_0,j_0} = z$

$((x_1, \cdots x_k), (a_1, \cdots, a_k)) \leftarrow$Lamport-Forge($\mathcal{Z}$)

if $x_{i_0} = j_0$

　then return (a_{i_0})

　else return (fail)

让我们考虑算法 7.1 的平均成功概率,这个平均概率是在所有 $z \in Z$ 上计算出来的。我们假定 Lamport-Forge 总能成功地找到一个伪造的签名。如果在伪造签名中 $x_{i_0} = j_0$,则等式

$$f(a_{i_0}) = z_{i_0, x_{i_0}} = z_{i_0, j_0} = z$$

成立,我们就找到了所需的 $f^{-1}(z)$。由前面可知,每一个 x_i 的值为 0 或 1,我们将证明,对算法的所有可能的运行结果做平均,$x_{i_0} = j_0$ 的概率是 1/2。因此,算法 7.1 平均成功的概率等于 1/2。在下面的定理中我们对此给出证明。

定理 7.1　假设 $f: Y \rightarrow Z$ 是一个单向双射函数,并且存在一个确定性算法 Lamport-Forge,对于在 Z 上的包含 $2k$ 个不同元素的公钥 $\mathcal{Z}$,它能使用惟密钥攻击为 Lamport 签名方案构建一个存在性伪造签名。那么存在一个算法 Lamport-Preimage,使得至少以 1/2 的概率找到一个随机元素 $z \in Z$ 的原像。

证明　设 $\mathcal{S}$ 表示所有可能的公钥集合,并且对于任意的 $z \in Z$,$\mathcal{S}_z$ 表示包含 z 的所有可能的公钥集合。令 $s = |\mathcal{S}|$,并且对所有的 $z \in Z$,令 $s_z = |\mathcal{S}_z|$。设 $\mathcal{T}_z$ 包括所有的 $\mathcal{Z} \in \mathcal{S}_z$ 上的公钥,它们使得当 $\mathcal{Z}$ 是由 Lamport-Preimage(z)选择的公钥时,Lamport-Preimage(z)将成功,令 $t_z = |\mathcal{T}_z|$。

我们将利用两个从基本计数技术得出的等式。第一个等式是:

$$\sum_{z \in Z} t_z = ks \tag{7.6}$$

这个等式的意义如下:存在 s 个可能的公钥,对于每一个公钥$\mathcal{Z}\in\mathcal{S}$,Lamport-Forge 能在 Z 上找到 k 个元素的逆。另一方面,对所有可能的公钥,由 Lamport-Forge 计算出的逆的总数为 $\sum t_z$。

其次,对任意的 $z\in Z$,下面的等式成立:

$$2ks = s_z \mid Z \mid \tag{7.7}$$

这个等式不难证明。s 个可能的公钥中的任意一个都包含 Z 上的 $2k$ 个元素。然而,显然每一个元素 $z\in Z$ 在公钥中出现的数目相同,因此 s_z 是(与 z 无关的)一个常量且 $2ks = s_z \mid Z \mid$。

设 p_z 表示 Lamport-Preimage(z)成功的概率,很明显 $p_z = t_z / s_z$。我们以下面的公式计算 p_z 的平均值 $\overline{p}$:

$$\begin{aligned}
\overline{p} &= \frac{1}{\mid Z \mid}\sum_{z\in Z} p_z \\
&= \frac{1}{\mid Z \mid}\sum_{z\in Z} \frac{t_z}{s_z} \\
&= \frac{1}{s_z \mid Z \mid}\sum_{z\in Z} t_z \\
&= \frac{1}{2ks}\sum_{z\in Z} t_z \qquad \text{由公式(7.6)} \\
&= \frac{ks}{2ks} \qquad \text{由公式(7.7)} \\
&= \frac{1}{2}
\end{aligned}$$

Lamport 签名方案非常优美,但它并不实用。一个问题是它产生签名的长度。例如,如果我们使用模指数函数构造 f,像例 7.6 那样,那么一个安全的签名要求 p 至少有 1024 比特长。这意味着消息的每一比特使用 1024 比特签名。因此,签名的长度是消息的 1024 倍长! 使用安全性基于椭圆曲线离散对数问题的不可行的单向双射函数可能效率更高,但是这种方法仍然不太实用。

7.5.2 全域 Hash

在 5.9.2 小节中,我们介绍了如何(在随机预言模型下)从陷门单向置换来构造可证明安全的公钥密码体制。这些体制的实现实际上基于 RSA 密码体制,这些体制将用像 SHA-1 这样的 Hash 函数替代随机预言。在本小节中,我们利用陷门单向置换来构造在随机预言模型下的安全签名方案。我们称这种签名方案为全域 Hash。全域 Hash 签名方案的名字来自该方案要求随机预言的值域与陷门单向置换的定义域相同。全域 Hash 签名方案见密码体制 7.7。

密码体制 7.7 全域 Hash

设 k 是一个正整数,$\mathcal{F}$是对所有 $f\in\mathcal{F}$使 $f:\{0,1\}^k\rightarrow\{0,1\}^k$ 的陷门单向置换族,并设 $G:\{0,1\}^*\rightarrow\{0,1\}^k$ 为一"随机"函数。设$\mathcal{P}=\{0,1\}^*$且$\mathcal{A}=\{0,1\}^k$。定义:

$$\mathcal{K} = \{(f, f^{-1}, G) : f \in \mathcal{F}\}$$

给定密钥 $K = (f, f^{-1}, G)$，f^{-1} 是私钥而 (f, G) 是公钥。
对于 $K = (f, f^{-1}, G)$ 和 $x \in \{0,1\}^*$，定义：

$$\text{sig}_K(x) = f^{-1}(G(x))$$

关于消息 x 的签名 $y = (y_1, \cdots, y_k) \in \{0,1\}^k$ 验证如下：

$$\text{ver}_K(x, y) = \text{true} \Leftrightarrow f(y) = G(x)$$

全域 Hash 使用常见的先 Hash 后签名的方法。$G(x)$ 是通过随机预言 G 产生的消息摘要。f^{-1} 用来对消息摘要签名，而 f 用来验证签名。

让我们简要地考虑一个基于 RSA 的全域 Hash 签名方案的实现。函数 f^{-1} 是 RSA 签名（即解密）函数，而 f 是 RSA 验证（即加密）函数。为安全起见，我们取 $k = 1024$。现在假设随机预言 G 被 Hash 函数 SHA-1 所替代。SHA-1 构造一个 160 比特的消息摘要，因此 Hash 函数的值域，也就是 $\{0,1\}^{160}$，是 $\{0,1\}^k \to \{0,1\}^{1024}$ 的一个非常小的子集。实际上，在应用 f^{-1} 以前，有必要规定一些填充方式以便将 160 比特的消息扩展成 1024 位，它一般是通过一个固定的填充方式完成的。

现在我们继续讨论方案的安全性证明，这里假设 $\mathcal{F}$ 是陷门单向置换族，而 G 是一个“全域”随机预言（注意，当像 SHA-1 这样的 Hash 函数替换了随机预言时，我们将提出的安全性证明不适用）。可以证明全域 Hash 对于使用选择消息攻击的存在性伪造是安全的，但是我们只证明对于惟密钥攻击的存在性伪造是安全的这一更容易的结论。

同前面一样，安全性证明采用简化形式。假设有一个攻击者（即一个随机算法，我们将它表示为 PDH-Forge）在给定公钥并能获取随机预言的情况下能（以某种特定的概率）伪造一个签名（对于值 $G(x)$，它能查询随机预言，但是没有特定的算法来评估函数 G）。PDH-Forge 允许做一定次数的预言查询，例如 q_h。最终，它以某种概率输出一个有效的伪造签名，用 ε 表示。

我们构造一个 FDH-Invert 算法，该算法试图求出随机选择的元素 $z_0 \in \{0,1\}^k$ 的逆，也就是给定 $z_0 \in \{0,1\}^k$，我们希望 PDH-Invert$(z_0) = f^{-1}(z_0)$。PDH-Invert 如算法 7.2 所示。

算法 7.2　PDH-Invert(z_0, q_h)

```
external f
procedure SimG(x)
if j > q_h
   then return("failure")
   else if j = j_0
   then z ← z_0
   else 随机选择 z ∈ {0,1}^k
   j ← j + 1
```

return (z)

main

随机选择 $j_0 \in \{1, \cdots, q_h\}$

$j \leftarrow 1$

此处插入 FDH-Forge(f)代码

if FDH-Forge$(f) = (x, y)$

then { **if** $f(y) = z_0$ **then return** (y) **else return**("failure")

算法 7.2 相对简单。它主要运行 PDH-Forge。PDH-Forge 执行的 Hash 查询由函数 SimG 实施,这是随机预言的一种模拟。我们已经假设 PDH-Forge 将执行 q_h 次 Hash 查询,分别为 x_1, $\cdots, x_{q_h}$。为了简单起见,我们假设 x_i 互不相同(如果不是这样,我们就需要确保只要 $x_i = x_j$,那么 SimG(x_i) = SimG(x_j)。这一点不难做到,它只需要做一些记录,像算法 5.14 一样)。我们随机选择第 j_0 个查询,并定义 SimG$(x_{j_0}) = z_0$(z_0 是一个我们试图求逆的值)。对所有其他的查询,选一个随机数作为 SimG(x_j)的值。因为 z_0 也是一个随机数,容易看出 SimG 与一个真正的随机预言是不可区分的。由此得出结论,FDH-Forge 以概率 ϵ 输出一则消息和一个有效的伪造签名,表示为(x, y)。然后我们检查是否 $f(y) = z_0$,如果是,则 $y = f^{-1}(z_0)$,我们就成功地求出了 z_0 的逆。

我们的主要任务是分析 FDH-Invert 成功的概率,它是成功率为 ϵ 的 FDH-Forge 算法的一个函数。我们假定 $\epsilon > 2^{-k}$,因为随机选择的 y 将以 2^{-k}的概率成为消息 x 的有效签名。我们仅仅对那些拥有比随机猜测更高概率的攻击者感兴趣。像前面一样,我们对 FDH-Forge 所做的 Hash 查询表示为 $x_1 \cdots, x_{q_h}$,其中 x_j 是第 j 个 Hash 查询,$1 \leqslant j \leqslant q_h$。

我们从成功概率为 ϵ 这一条件开始,不论是否 $x \in x_1, \cdots, x_{q_h}$:

$$\begin{aligned} \epsilon = & \mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x \in \{x_1, \cdots, x_{q_h}\})] \\ & + \mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x \notin \{x_1, \cdots, x_{q_h}\})] \end{aligned} \tag{7.8}$$

不难看出

$$\mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x \notin \{x_1, \cdots, x_{q_h}\})] = 2^{-k}$$

这是因为不确定值 SimG(x)在$\{0,1\}^k$ 上等可能地取值,因此 SimG$(x) = f(y)$的概率为 2^{-k}(这是因为我们使用了 Hash 函数是全域 Hash 的假定)。代入(7.8)式,得到下面的等式:

$$\mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x \in \{x_1, \cdots, x_{q_h}\})] \geqslant \epsilon - 2^{-k} \tag{7.9}$$

现在我们转向 FDH-Invert 成功的概率。下面的不等式是显然的:

$$\mathbf{Pr}[\text{FDH-Invert succeeds}] \geq \mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x = x_{j_0})] \tag{7.10}$$

最后我们看到：

$$\begin{aligned} &\mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x = x_{j_0})] \\ &= \frac{1}{q_h}\mathbf{Pr}[\text{FDH-Forge succeeds} \wedge (x \in \{x_1, \cdots, x_{q_h}\})] \end{aligned} \tag{7.11}$$

注意，等式(7.11)为真，因为给定 $x \in \{x_1, \cdots, x_{q_h}\}$，$x = x_{j_0}$ 的可能性为 $1/q_h$。现在，如果我们合并(7.9)、(7.10)和 (7.11)，那么就得到下面的界：

$$\mathbf{Pr}[\text{FDH-Invert succeeds}] \geq \frac{\varepsilon - 2^{-k}}{q_h} \tag{7.12}$$

因此，我们获得了关于 FDH-Invert 成功的一个具体的下界。我们已经证明了下面的结论。

定理 7.2 假设存在一个算法 FDH-Forge，使用惟密钥攻击，它将以 $\varepsilon > 2^{-k}$ 的概率对全域 Hash 输出一个存在性伪造签名，那么，存在一个算法 FDH-Invert，它将以不小于 $\frac{\varepsilon - 2^{-k}}{q_h}$ 的概率找到 $z_0 \in \{0,1\}^k$ 的随机元素的逆。

可以看出，得出的求逆算法的有效性依赖于使用尽可能少的 Hash 查询找到伪造签名的 FDH-Forge 的能力。

7.6 不可否认的签名

不可否认的签名是由 Chaum 和 van Antwerpen 在 1989 年提出来的。不可否认的签名有一些新颖的特征，其中最主要的一点是没有签名者 Alice 的合作，签名就不能得到验证。这就防止了由她签署的电子文档在没有经过她的同意而被复制和分发的可能性。验证签名将通过口令-响应(Challenge-and-response)协议来实现。

如果验证一个签名需要 Alice 的合作，要阻止她否认一个早些时候的签名该怎么办呢？Alice 可能声称一个有效的签名是伪造的，并且要么拒绝验证它，要么实现一个协议以便该签名不能得到验证。为了阻止这种情况发生，一个不可否认的签名与一个否认协议结合，通过这种方式 Alice 能证明一个签名是一个伪造签名。因此，Alice 能“在法庭”证明一个伪造的签名在事实上是伪造的(如果她拒绝执行否认协议，则认为这件事本身就是该签名是事实上的真正签名的证据)。

因此，一个不可否认的签名由三个部分组成：一个签名算法、一个验证协议和一个否认协议。首先我们介绍 Chaum-van Antwerpen 签名方案的签名算法和验证协议，参见密码体制 7.8。

密码体制 7.8 Chaum-van Antwerpen 签名方案

设 $p = 2q + 1$ 是一个使得 q 是素数并且在 $\mathbb{Z}_p$ 上的离散对数问题是难处理的素数。设

$\alpha \in \mathbb{Z}_p^*$ 是一个阶为 q 的元素。设 $1 \leqslant a \leqslant q-1$,令 $\beta = \alpha^a \pmod p$。设 G 表示 $\mathbb{Z}_p^*$ 上阶为 q 的乘法子群(G 由模 p 的二次剩余构成)。设 $\mathcal{P} = \mathcal{A} = G$,定义

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$$

值 p,α 和 β 是公钥,a 为私钥。

对于 $K = (p, \alpha, a, \beta)$ 和 $x \in G$,定义

$$y = \mathrm{sig}_K(x) = \alpha^a \bmod p$$

对 $x, y \in G$,通过执行下面的协议来验证签名:

1. Bob 随机选择 $e_1, e_2 \in \mathbb{Z}_q$。

2. Bob 计算 $c = y^{e_1} \beta^{e_2} \bmod p$ 并将它发送给 Alice。

3. Alice 计算 $d = c^{a^{-1} \bmod q} \bmod p$ 并将它发送给 Bob。

4. 当且仅当 $d \equiv x^{e_1} \alpha^{e_2} \pmod p$ 时,Bob 将 y 作为合法的签名接收。

我们来解释一下不可否认签名方案中 p 和 q 的作用。Chaum-van Antwerpen 签名方案定义在 $\mathbb{Z}_p$ 上,然而,我们需要能在 $\mathbb{Z}_q^*$ 的素数阶乘法子群 G 上进行计算。我们尤其需要能计算模 $|G|$ 的逆,这就是 $|G|$ 应该是素数的原因。取素数 $p = 2q + 1$(其中 q 是素数),是为了使计算变得方便。这样,子群 G 就会尽可能得大,这正是所需要的,因为消息和签名都是 G 中的元素。

我们首先证明 Bob 将接收一个有效的签名。在下面的计算中,所有的指数将省掉模 p。首先,我们看到

$$d \equiv c^{a^{-1}} \equiv (\bmod\ p) \equiv y^{e_1 a^{-1}} \beta^{e_2 a^{-1}} (\bmod\ p)$$

因为

$$\beta \equiv \alpha^a (\bmod\ p)$$

我们有

$$\beta^{a^{-1}} \equiv \alpha (\bmod\ p)$$

类似地,由

$$y = x^a (\bmod\ p)$$

可推出

$$y^{a^{-1}} = x (\bmod\ p)$$

因此有:

$$d \equiv x^{e_1} \alpha^{e_2} (\bmod\ p)$$

下面给出一个小例子加以说明。

例 7.7　假设 $P=467$，因为 2 是一个本原元，$2^2=4$ 是 G 的一个生成元，G 由模 467 的二次剩余组成。因此我们取 $\alpha=4$，假设 $a=101$，那么 $\beta=\alpha^a \bmod 467=449$。Alice 将用 $y=119^{101} \bmod 467=129$ 对消息 $x=119$ 签名。

现在，假设 Bob 想验证签名 y。假定他选择随机值 $e_1=38$，$e_2=297$，他计算出 $c=13$，那么 Alice 将以 $d=9$ 作为响应。Bob 通过验证

$$119^{38}4^{397} \equiv 9 (\bmod\ 467)$$

来检查响应。因此，Bob 将它作为合法签名接收。

下面我们来证明 Alice 只能用很小的概率来愚弄 Bob 接收一个伪造的签名。这个结果不依赖于任何计算假设，即安全性是无条件的。

定理 7.3　如果 $y \not\equiv x^a (\bmod\ p)$，那么 Bob 将以 $1/q$ 的概率把 y 当作 x 的一个合法签名接收。

证明　首先，我们观察到每一个可能的口令 c 恰好对应于 q 个有序对 (e_1, e_2)（这是因为 y 和 β 都是 G 中阶为 q 的素数乘法子群的元素）。现在，当 Alice 接收到 c 时，她无法知道 Bob 是用 q 个可能的有序对 (e_1, e_2) 中的哪一个来构造 c。我们断言，如果 $y \not\equiv x^a (\bmod\ p)$，那么 Alice 做出的任何 $d \in G$ 的响应与 q 个可能的有序对 (e_1, e_2) 中的一个一致。

因为 α 是 G 的生成元，G 中的任何元素都可以写成 α 的幂次方，其中指数由模 q 惟一确定。设 $c=\alpha^i$，$d=\alpha^j$，$x=\alpha^k$ 以及 $y=\alpha^l$，其中 $i, j, k, l \in \mathbb{Z}_q$，且所有的计算都是模 p 操作。考虑下面的两个等式：

$$c \equiv y^{e_1} \beta^{e_2} (\bmod\ p)$$
$$d \equiv x^{e_1} \alpha^{e_2} (\bmod\ p)$$

这个方程组等价于下面的方程组：

$$i \equiv le_1 + ae_2 (\bmod\ q)$$
$$j \equiv ke_1 + e_2 (\bmod\ q)$$

现在，我们假设 $y \not\equiv x^a (\bmod\ p)$，由此可以推出：

$$l \not\equiv ak (\bmod\ q)$$

因此，这个模 q 的方程组的系数矩阵具有非 0 的判别式，因此具有惟一解。也就是说，每一个 $d \in G$ 恰好是 q 个可能的有序对 (e_1, e_2) 中的一个正确响应。因此，Alice 给 Bob 的响应 d 能通过验证的概率恰好是 $1/q$，由此定理得以证明。

我们现在转向否认协议。该协议由两轮验证协议组成，如算法 7.3 所示。

算法 7.3　否认

1. Bob 随机选择 $e_1, e_2 \in \mathbb{Z}_q$。

2. Bob 计算 $c = y^{e_1}\beta^{e_2} \bmod p$ 并将它发送给 Alice。

3. Alice 计算 $d = c^{a^{-1} \bmod q} \bmod p$ 并将它发送给 Bob。

4. Bob 验证 $d \not\equiv x^{e_1}\alpha^{e_2} \pmod p$。

5. Bob 随机选择 $f_1, f_2 \in \mathbb{Z}_q$。

6. Bob 计算 $C = y^{f_1}\beta^{f_2} \bmod p$ 并将它发送给 Alice。

7. Alice 计算 $D = C^{a^{-1} \bmod p} \bmod p$ 并将它发送给 Bob。

8. Bob 验证 $x^{f_1}\alpha^{f_2} \pmod p$。

9. 当且仅当 $(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod p$，Bob 推断签名 y 是伪造的。

步骤 1 ~ 步骤 4 以及步骤 5 ~ 步骤 8 包括两轮不成功的验证协议，步骤 9 是一个“一致性检测”，它能使 Bob 确定 Alice 是否按协议的规定形成她的响应。

下面的例子解释了否认协议。

例 7.8 同前面的例子一样，设 $p = 467, \alpha = 4, a = 101$ 且 $\beta = 449$。假设消息 $x = 286$ 的(伪造)签名为 $y = 83$，并且 Alice 要使 Bob 相信这个签名是无效的。

假设 Bob 从选择两个随机值 $e_1 = 45$ 和 $e_2 = 237$ 开始。他计算出 $c = 305$ 并且 Alice 的响应是 $d = 109$。然后 Bob 计算：

$$286^{45}4^{237} \bmod 467 = 149$$

因为 $109 \neq 149$，Bob 将继续执行协议的步骤 5。

现在假设 Bob 随机选择 $f_1 = 125, f_2 = 9$。Bob 计算出 $C = 270$ 而 Alice 的响应是 $D = 68$。Bob 计算：

$$286^{125}4^{9} \bmod 467 = 25$$

因为 $25 \neq 68$，Bob 继续执行到协议的步骤 9 来完成一致性验证。这个验证是成功的，因为

$$(109 \times 4^{-237})^{125} \equiv 188 \pmod{467}$$

并且

$$(68 \times 4^{-9})^{45} \equiv 188 \pmod{467}$$

因此 Bob 被证实该签名是无效的。

到现在为止，我们还需要证明两点：

1. Alice 能使 Bob 相信一个无效的签名是一个伪造签名。
2. Alice 能使 Bob 以一个很小的概率相信一个有效的签名是伪造的。

定理 7.4 如果 $y \not\equiv x^a \pmod p$，并且 Alice 和 Bob 都遵循否认协议，那么：

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod p$$

证明 因为

$$d \equiv c^{a^{-1}} \pmod{p}$$
$$c \equiv \gamma^{e_1} \beta^{e_2} \pmod{p}$$
$$\beta \equiv \alpha^{a} \pmod{p}$$

所以我们有：

$$\begin{aligned}(d\alpha^{-e_2})^{f_1} &\equiv ((\gamma^{e_1}\beta^{e_2})^{a^{-1}}\alpha^{-e_2})^{f_1} \pmod{p}\\ &\equiv \gamma^{e_1 a^{-1} f_1}\beta^{e_2 a^{-1} f_1}\alpha^{-e_2 f_1} \pmod{p}\\ &\equiv \gamma^{e_1 a^{-1} f_1}\alpha^{e_2 f_1}\alpha^{-e_2 f_1} \pmod{p}\\ &\equiv \gamma^{e_1 a^{-1} f_1} \pmod{p}\end{aligned}$$

同样地，因为 $D \equiv C^{a^{-1}} \pmod{p}$，$C \equiv \gamma^{f_1}\beta^{f_2} \pmod{p}$以及 $\beta \equiv \alpha^{a} \pmod{p}$，有

$$(D\alpha^{-f_2})^{e_1} \equiv \gamma^{e_1 a^{-1} f_1} \pmod{p}$$

因此否认协议中步骤9的一致性检查是成功的。

现在来看一下 Alice 企图否认一个有效签名的可能性。在这种情况下，我们并不认为 Alice 会遵守协议，即 Alice 可能不会遵照协议的规定来构造 d 和 D。因此，在下面的定理中，我们仅仅假定 Alice 能够产生满足算法 7.3 的步骤 4、8 和 9 中条件的 d 和 D。

定理 7.5 假设 $\gamma \equiv x^{a} \pmod{p}$，并且 Bob 遵循否认协议。如果 $d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$并且 $D \not\equiv x^{f_1}\alpha^{f_2} \pmod{p}$，那么$(d\alpha^{-e_2})^{f_1} \not\equiv (D\alpha^{-f_2})^{e_1} \pmod{p}$的概率是 $1 - 1/q$。

证明 假设下面的等式成立：

$$\gamma \equiv x^{a} \pmod{p}$$
$$d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$$
$$D \not\equiv x^{f_1}\alpha^{f_2} \pmod{p}$$
$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod{p}$$

我们推导出它们是矛盾的。

一致性检验(步骤 9)可以重写为以下的形式：$D \equiv d_0^{f_1}\alpha^{f_2} \pmod{p}$，其中 $d_0 = d^{1/e_1}\alpha^{-e_2/e_1} \bmod p$ 是一个仅仅依赖协议中步骤 1 ~ 步骤 4 的值。

应用定理 7.3，我们得出，对于 d_0 来讲，γ 是一个有效签名的概率为 $1 - 1/q$。但是我们假定 γ 是 x 的有效签名，即 $x^{a} \equiv d_0^{a} \pmod{p}$的概率很高，也就是 $x = d_0$。然而 $d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$，意味着 $x \not\equiv d^{1/e_1}\alpha^{-e_2/e_1} \pmod{p}$。因为 $d_0 = d^{1/e_1}\alpha^{-e_2/e_1} \bmod p$，我们可以得出 $x \neq d_0$ 的结论，与 $x = d_0$ 互相矛盾。因此，Alice 以这种方式愚弄 Bob 的概率为 $1/q$。

7.7 fail-stop 签名

fail-stop(失败即停)签名方案在防止一个能伪造签名的很强大的攻击者方面提供了增强的安全性。在 Oscar 对一则消息能伪造 Alice 的签名事件中,Alice 将随后能(以高概率)证明 Oscar 的签名是伪造的。

这一节,我们介绍由 van Heyst 和 Pedersen 在 1992 年提出的 fail-stop 数字签名方案。像 Lamport 签名方案一样,它属于一次签名方案。方案包括签名算法、验证算法和一个"伪造证明"算法。van Heyst 和 Pedersen 签名方案的签名算法和验证算法的描述如密码体制 7.9 所示。

密码体制 7.9 van Heyst 和 Pedersen 签名方案

设 $p = 2q + 1$ 是一个使得 q 是素数并且在$\mathbb{Z}_p$ 上的离散对数问题是难处理的素数。设 $\alpha \in \mathbb{Z}_p^*$ 是一个阶为q 的元素。设 $1 \leqslant a_0 \leqslant q - 1$,令 $\beta = \alpha^{a_0} \bmod p$。值 p, q, α, β 和 a_0 是由一个可信中心选择。p, q, α 和 β 是公开的并认为是固定不变的。值 a_0 对包括 Alice 在内的任何人都是保密的。

设$\mathcal{P} = \mathbb{Z}_q$,且$\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$。密钥具有

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$

的形式,其中 $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$,

$$\gamma_1 = \alpha^{a_1} \beta^{a_2} \bmod p$$
$$\gamma_2 = \alpha^{b_1} \beta^{b_2} \bmod p$$

(γ_1, γ_2)是公钥,而(a_1, a_2, b_1, b_2)是私钥。

对于 $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$和 $x \in \mathbb{Z}_q$,定义:

$$\mathrm{sig}_K(x) = (y_1, y_2)$$

其中,

$$y_1 = a_1 + xb_1 \bmod q$$
$$y_2 = a_2 + xb_2 \bmod q$$

对于 $y = (y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$,我们有:

$$\mathrm{ver}_K(x, y) = \text{true} \Leftrightarrow \gamma_1 \gamma_2^x \equiv \alpha^{y_1} \beta^{y_2} \pmod p$$

可直接看出,由 Alice 产生的签名满足验证条件,因此我们讨论 fail-stop 数字签名的安全性以及看它是如何工作的。我们首先建立起关于密钥的一些重要事实。我们从定义开始,如果 $\gamma_1 = \gamma'_2$ 且 $\gamma_2 = \gamma'_2$,则我们称两个密钥$(\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$和$(\gamma'_1, \gamma'_2, a'_1, a'_2, b'_1, b'_2)$是

等价的,容易看出在任何一个等价类里恰好有 q^2 个密钥。

我们建立几个引理。

引理 7.6　假定 K 和 K' 是等价的密钥,并且 $\mathrm{ver}_K(x,y)=\mathrm{true}$,那么 $\mathrm{ver}_{K'}(x,y)=\mathrm{true}$。

证明　设 $K=(\gamma_1,\gamma_2,a_1,a_2,b_1,b_2)$, $K'=(\gamma_1,\gamma_2,a'_1,a'_2,b'_1,b'_2)$,其中

$$\gamma_1=\alpha^{a_1}\beta^{a_2}\bmod p=\alpha^{a'_1}\beta^{a'_2}\bmod p$$

并且

$$\gamma_2=\alpha^{b_2}\beta^{b_2}\bmod p=\alpha^{b'_1}\beta^{b'_2}\bmod p$$

假设 x 使用 K 签名,产生的签名为 $y=(y_1,y_2)$,其中

$$y_1=a_1+xb_1\bmod q$$
$$y_2=a_2+xb_2\bmod q$$

现在我们计算 $\mathrm{ver}_{K'}(x,y)$:

$$\begin{aligned}\alpha^{y_1}\beta^{y_2}&\equiv\alpha^{a'_1+xb'_1}\beta^{a'_2+xb'_2}(\bmod p)\\&\equiv\alpha^{a'_1}\beta^{a'_2}(\alpha^{b'_1}\beta^{b'_2})^x(\bmod p)\\&\equiv\gamma_1\gamma_2{}^x(\bmod p)\end{aligned}$$

因此,y 也可以由 K' 验证。

引理 7.7　假设 K 是一个密钥并且 $y=\mathrm{sig}_K(x)$,那么恰好存在 q 个与 K 等价的密钥 K' 使得 $y=\mathrm{sig}_{K'}(x)$。

证明　假定(γ_1,γ_2)是公钥。我们要决定使得下面的等式成立的四元组(a_1,a_2,b_1,b_2):

$$\gamma_1\equiv\alpha^{a_1}\beta^{a_2}(\bmod p)$$
$$\gamma_2\equiv\alpha^{b_1}\beta^{b_2}(\bmod p)$$
$$y_1\equiv a_1+xb_1(\bmod q)$$
$$y_2\equiv a_2+xb_2(\bmod q)$$

因为 α 产生 G,存在惟一的指数 $c_1,c_2,a_0\in\mathbb{Z}_q$ 使得

$$\gamma\equiv\alpha^{c_1}(\bmod p)$$
$$\gamma_2\equiv\alpha^{c_2}(\bmod p)$$
$$\beta\equiv\alpha^{a_0}(\bmod p)$$

因此,下面的等式成立是充分必要的:

$$c_1\equiv a_1+a_0a_2(\bmod q)$$
$$c_2\equiv b_1+a_0b_2(\bmod q)$$
$$y_1\equiv a_1+xb_1(\bmod q)$$

$$y_2 \equiv a_2 + xb_2 (\mathrm{mod}\ q)$$

上面的方程组在$\mathbb{Z}_p$上能写成如下的矩阵形式:

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix}$$

该方程的系数矩阵的秩为3:很清楚,因为矩阵的第1、2和4行在$\mathbb{Z}_p$上线形无关,故它的秩至少为3。因为:

$$r_1 + xr_2 - r_3 - a_0 r_4 = (0,0,0,0)$$

其中r_i表示矩阵的第i行,因此秩最多为3。

这个方程组至少有一个解,由密钥K给出。因为系数矩阵的秩为3,所以该方程的解空间的维数是$4-3=1$,因此恰好有q个密钥。引理得到证明。

类似地,可证明下面的结论,我们省去证明过程。

引理7.8 假设K是一个密钥,$y = \mathrm{sig}_K(x)$,并且$\mathrm{ver}_K(x', y') = \mathrm{true}$,其中$x \neq x'$,那么至多有一个与$K$等价的密钥$K'$使得$y = \mathrm{sig}_{K'}(x)$且$y' = \mathrm{sig}_{K'}(x')$。

让我们解释一下前面两个引理在签名方案的安全方面表示的含义。假定y是x的有效签名,存在q个可能的密钥使得y是x的签名。但是对于任何消息$x \neq x'$,这q个不同的密钥将对x'产生q个不同的签名。因此,下面的定理成立。

定理7.9 假定$\mathrm{sig}_K(x) = y$,$x \neq x'$,Oscar能计算出$\mathrm{sig}_K(x')$的概率为$1/q$。

注意,这个定理不依赖于Oscar的计算能力:可以获得提出的安全性,因为Oscar不能区分q个可能的密钥中的哪一个被Alice使用。因此这种安全性是无条件的。

我们现在继续来看fail-stop签名的概念。到现在为止我们所说的是:给定消息x的签名,Oscar不能计算Alice对不同的消息x'的签名y'。仍然可以想像Oscar能计算出一个可以验证的伪造签名$y'' \neq \mathrm{sig}_K(x')$。然而,如果给Alice一个有效的伪造签名,那么她能以概率$1-1/q$来给出该伪造签名的证明。伪造证明的值是$a_0 = \log_\alpha \beta$,它只有可信中心知道。

我们假设Alice拥有(x', y'')使得$\mathrm{ver}_K(x', y'') = \mathrm{true}$且$y'' \neq \mathrm{sig}_K(x')$。也就是

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y''_1} \beta^{y''_2} (\mathrm{mod}\ p)$$

其中$y'' = (y''_1, y''_2)$。现在,对于消息x',Alice能计算出她自己的签名(y'_1, y'_2),并且

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y'_1} \beta^{y'_2} (\mathrm{mod}\ p)$$

因此,

$$\alpha^{\gamma''_1}\beta^{\gamma''_2} \equiv \alpha^{\gamma'_1}\beta^{\gamma'_2} (\text{mod } p)$$

令 $\beta = \alpha^{a_0} \bmod p$,我们有:

$$\alpha^{\gamma''_1 + a_0\gamma''_2} \equiv \alpha^{\gamma'_1 + a_0\gamma'_2} (\text{mod } p)$$

或者

$$\gamma''_1 + a_0\gamma''_2 \equiv \gamma'_1 + a_0\gamma'_2 (\text{mod } q)$$

由此可得:

$$\gamma''_1 - \gamma'_1 \equiv a_0(\gamma'_2 - \gamma''_2)(\text{mod } q)$$

因为 γ' 是伪造的,$\gamma'_2 \not\equiv \gamma''_2 (\text{mod } q)$。因此$(\gamma'_2 - \gamma''_2)^{-1} \bmod q$ 存在且

$$a_0 = \log_\alpha \beta = (\gamma''_1 - \gamma'_1)(\gamma'_2 - \gamma''_2)^{-1} (\text{mod } q)$$

当然,要接收这一伪造证明,我们假定 Alice 自己不能计算离散对数问题 $\log_\alpha \beta$。这是一种计算假设。

最后,我们指出该签名方案是一个一次签名方案,因为 Alice 在两个消息都用 K 签名的情况下,该密钥 K 就很容易被计算出来。

本节末我们用一个例子说明 Alice 如何给出一个伪造证明。

例 7.9 假设 $p = 3467 = 2 \times 1733 + 1$,元素 $\alpha = 4$ 在$\mathbb{Z}^*_{3467}$上其阶数为 1733。假设 $a_0 = 1567$,因此,

$$\beta = 4^{1567} \bmod 3467 = 514$$

(Alice 知道 α 和 β 的值,但不知道 a_0。)假定 Alice 使用 $a_0 = 888$, $a_2 = 1024$, $b_1 = 786$ 和 $b_2 = 999$生成她的密钥,则

$$\gamma_1 = 4^{888}514^{1024} \bmod 3467 = 3405$$

且

$$\gamma_2 = 4^{786}514^{999} \bmod 3467 = 2281$$

现在,假定给 Alice 一个关于消息 3383 的一个伪造签名(822,55)。这是一个有效的签名,因为满足验证条件:

$$3405 \times 2281^{3383} \equiv 2282(\text{mod } 3467)$$

且

$$4^{822}514^{55} \equiv 2282(\text{mod } 3467)$$

另一方面,这不是 Alice 可以构造的签名。Alice 能计算的她自己的签名是

$$(888 + 3383 \times 786 \bmod 1733, 1024 + 3383 \times 999 \bmod 1733) = (1504, 1291)$$

然后,她继续计算秘密离散对数:

$$a_0 = (882 - 1504)(1291 - 55)^{-1} \bmod 1733 = 1567$$

这就是伪造的证明。

7.8 注释与参考之献

关于签名方案的一个好的综述,我们推荐 Mitchell、Piper 和 Wild 的论文[149]。这篇论文也包含了在 7.3 节中提出的伪造 ElGamal 签名的两种方法。Pedersen 的论文[162]是最近发表的一篇更值得一读的综述文章。

ElGamal 签名方案由 ElGamal 在文献[70]中提出来的,而 Schnorr 签名方案由 Schnorr 在文献[187]中提出。另一个在本书中没有讨论的签名方案是 Fiat-Shamir 签名方案[86]。

数字签名算法(DSA)由 NIST 在 1991 年 8 月首次发表,在 1994 年 12 月作为 FIPS 186 被采纳[79]。在 1992 年 7 月出版的杂志《Communications of the ACM》有关于 DSA 签名算法冗长的讨论以及围绕它的一些争论。NIST 对其中一些问题的回应参见 [200]。FIPS 186-2[80]是 DSA 标准的修改版,它现在包括 RSA 签名算法和椭圆曲线数字签名算法。ECDSA 的一个完整描述在 Johnson、Menezes 和 Vanstone 的论文[107]中可以找到。

Lamport 签名方案在 Diffie 和 Hellman 于 1976 年发表的论文[67]中做了介绍。Lamport 以及 Bos 和 Chaum 分别对此在提高效率方面做的修改参见[34]。从任意的单向函数构造数字签名的更一般的处理由 Bleichenbacher 和 Maurer 给出[28]。

全域 Hash 归功于 Bellare 和 Rogaway[14,16]。论文[16]也包括了一个效率更高的称为概率签名方案(PSS)的全域 Hash 的变体。可证明安全的 ElGamal 类型的签名方案也被研究过,可以参考 Pointcheval 和 Stern 的论文[169]。

7.6 节提出的不可否认的签名方案归功于 Chaum 和 van Anwerpen[50]。7.7 节给出的 fail-stop 签名方案归功于 van Heyst 和 Pedersen[210],关于该主题的进一步内容参见 Pfitzmann [165]。

本章中的一些练习指出,如果"k"值被重复使用或以一种可预测的方式产生,ElGamal 类型的签名方案存在一些安全问题。目前有人在这方面开展了一些工作,例如,Bellare、Goldwasser 和 Micciancio[17],以及 Nguyen 和 Shparlinski[154]。

练习

7.1 假设 Alice 使用 ElGamal 签名方案 $p = 31\,847, \alpha = 5$ 以及 $\beta = 25\,703$。给定消息 $x = 8990$ 的签名(23 972,31 396)以及 $x = 31\,415$ 的签名(23 972,20 481),计算 k 和 a 的值(无需求解离散对数问题的实例)。

7.2 假设我们实现了 $p = 31\,847, \alpha = 5$ 以及 $\beta = 26\,379$ 的 ElGamal 签名方案。编制完成下面任务的计算机程序:

(a)验证对消息 $x = 20\,543$ 的签名(20 679,11 082)。

(b)通过求解离散对数问题的实例决定私钥 a。

(c)在无需求解离散对数问题的实例情况下,确定对消息 x 签名时使用的随机值 k。

7.3 假设 Alice 正在使用 ElGamal 签名方案。为了在产生对消息 x 签名时使用的随机值 k 时节省时间,她选择了一个初始的随机值 k_0,并在签署第 i 则消息时取 $k_i = k_0 + 2i \bmod p$(因此对所有的 $i \geqslant 1$ 有 $k_i = k_{i-1} + 2 \bmod p$)。

(a)假设 Bob 观测到两则连续的签名消息$(x_i, \text{sig}(x_i))$和$(x_{i+1}, \text{sig}(x_{i+1}))$。描述 Bob 在已知该信息且无需求解离散对数问题的实例的情况下,如何容易地计算 Alice 的密钥 a。(注意,为了使攻击成功,不必知道 i 的值。)

(b)假设该方案的参数是 $p = 28\,703$, $\alpha = 5$ 和 $\beta = 11\,339$,并且 Bob 观测到的信息为:

$$x_i = 12\,000 \qquad \text{sig}(x_i, k_i) = (26\,530, 19\,862)$$
$$x_{i+1} = 24\,567 \qquad \text{sig}(x_{i+1}, k_{i+1}) = (3081, 7604)$$

使用(a)中描述的攻击方法找到密钥 a。

7.4 (a)证明 7.3 节中介绍的 ElGamal 签名方案的第二种伪造方法,同时生成一个满足验证条件的签名。

(b)假设 Alice 在实现例 7.1 的 ElGamal 签名方案:$p = 467$, $\alpha = 2$ 且 $\beta = 132$。假设 Alice 已经用(29,51)对消息 $x = 200$ 签名。计算使用 $h = 102$, $i = 45$ 和 $j = 293$ 的 Oscar 伪造的签名。检验计算出的签名满足验证条件。

7.5 (a)在 ElGamal 签名方案或 DSA 中不允许 $\delta = 0$。证明如果对消息签名时 $\delta = 0$,那么攻击者很容易计算出密钥 a。

(b) 在 DSA 中的签名不允许 $\gamma = 0$。证明如果已知一个签名使用的是 $\gamma = 0$,那么"签名"所使用的 k 值就能确定。给定 k 值,证明对任何所期望的消息可伪造一个(在 $\gamma = 0$ 时)签名(即可实现选择性伪造)。

(c)评估 ECDSA 中允许 $r = 0$ 或 $s = 0$ 的签名的后果。

7.6 这里是 ElGamal 签名方案的一种变型。密钥用同前面相似的方法构造:Alice 选择 $\alpha \in \mathbb{Z}_p^*$ 是一个本原元,$0 \leqslant a \leqslant p-2$,其中 $\gcd(a, p-1)$且 $\beta = \alpha^a \bmod p$。密钥 $K = (\alpha, a, \beta)$,其中 α, β 值是公钥,a 是私钥。设 $x \in \mathbb{Z}_p$ 是一则要签名的消息,Alice 计算签名 $\text{sig}(x) = (\gamma, \delta)$,其中

$$\gamma = \alpha^k \bmod p$$

且

$$\delta = (x - k\gamma)a^{-1} \bmod (p-1)$$

与原来的 ElGamal 签名方案惟一的差别是计算 δ。回答有关该方案的下列问题。

(a)描述关于消息 x 的签名(γ, δ)是如何使用 Alice 的公钥进行验证的。

(b)描述修改后的方案的计算优点。

(c)简要比较原来的与修改后的方案的安全性。

7.7 假设 Alice 使用 $q = 101$, $p = 7879$, $\alpha = 170$, $\alpha = 75$ 和 $\beta = 4567$ 的 DSA 签名算法,如例 7.4 一样。给出使用随机值 $k = 49$ 的对消息 SHA-1$(x) = 52$ 的签名,并说明该签名

是如何被验证的。

7.8 我们已经说明在 ElGamal 签名方案中使用相同的 k 给两则消息签名时该方案是如何被攻破的(即攻击者在不用求解离散对数问题的实例的情况下决定私钥)。说明在 Schnorr 签名方案、DSA 签名方案和 ECDSA 签名方案中类似的攻击是如何实现的。

7.9 假设 $x_0 \in \{0,1\}^*$ 是一个使 SHA-1$(x_0) = 00\cdots0$ 的一个比特串,因此,当我们使用 DSA 或 ECDSA 时,就有 SHA-1$(x_0) \equiv 0 \pmod q$。

(a)说明如何对消息 x_0 伪造一个 DSA 签名。

提示:设 $\delta = \gamma$,其中 γ 是适当选择的。

(b)说明如何对消息 x_0 伪造一个 ECDSA 签名。

7.10 (a)我们对 DSA 描述一个潜在的攻击。假设给定 x,令 $z = (\text{SHA-1}(x))^{-1} \bmod q$ 且 $\epsilon = \beta^z \bmod p$。现在假设能找到 $\gamma, \lambda \in \mathbb{Z}_q^*$ 使得

$$((\alpha\epsilon^{\gamma})^{\lambda^{-1} \bmod q}) \bmod p \bmod q = \gamma$$

定义 $\delta = \lambda \text{SHA-1}(x) \bmod q$。证明$(\gamma, \delta)$是 x 的一个有效签名。

(b)描述对 ECDSA 的类似的攻击。

7.11 在验证使用 ElGamal 签名方案(或它的变型)构造的签名时,需要计算形如 $\alpha^c\beta^d$ 的值。如果 c 和 d 是一个随机 l 比特指数,那么直接使用平方-乘算法来计算每一个 α^c 和 β^d 平均需要 $l/2$ 次乘法和 l 次平方。本练习的目的是说明乘积 $\alpha^c\beta^d$ 怎样更高效地计算出来。

(a)假设 c 和 d 以二进制形式表示,如算法 5.5 一样。假设乘积 $\alpha\beta$ 已事先计算出来。描述对算法 5.5 的修改,使得算法的每一次迭代至多只进行一次乘法。

(b)假设 $c = 26$ 和 $d = 17$,说明如何用你的算法计算 $\alpha^c\beta^d$,即你的算法每次迭代之后指数 i 和 j 的值是什么(其中 $z = \alpha^i\beta^j$)。

(c)如果 c 和 d 是随机选择的 l 比特整数,解释为什么计算 $\alpha^c\beta^d$ 算法平均需要 l 次平方和 $3l/4$ 次乘法。

(d)假设平方操作和乘法操作花的时间大致相同,与原来的平方-乘算法分别计算 α^c 与 β^d 相比,估计该方法平均增加的计算速度。

7.12 证明在 ECDSA 中一个正确构造的签名将满足验证条件。

7.13 设 E 表示椭圆曲线 $y^2 \equiv x^3 + x + 26 \bmod 127$,可以看出 $\#E = 131$ 是一个素数。因此,E 中任何非单位元素是$(E, +)$的生成元。假设 ECDSA 在 E 上实现,$A = (2, 6)$,$m = 54$。

(a)计算公钥 $B = mA$。

(b)当 $k = 75$ 时,计算在 SHA-1$(x) = 10$ 的情况下关于 x 的签名。

(c)说明用于验证(b)构造出来的签名的计算过程。

7.14 在 Lamport 签名方案中,假设有两个 k 元组 x 和 x' 被 Alice 使用相同的密钥签名。设 l 表示 x 和 x' 不同的坐标数,即

$$l = |\{i : x_i \neq x'_i\}|$$

证明 Oscar 能对 $2^l - 2$ 个新的消息签名。

7.15　假设 Alice 像例 7.7 那样正在使用 Chaum-van Antwerpen 签名方案，即 $p = 467$，$\alpha = 4$，$a = 101$ 且 $\beta = 449$。假设 Alice 得到一个关于消息 $x = 157$ 的签名 $y = 25$，她想证明该签名是伪造的。假设在否认协议中 Bob 的随机数 $e_1 = 46$，$e_2 = 123$，$f_1 = 198$ 以及 $f_2 = 11$。计算 Bob 的挑战 c 和 C，以及 Alice 的响应 d 和 D，并证明 Bob 的一致性检查将成功。

7.16　证明 Pedersen-van Heyst 签名方案的每一个等价类包含 q^2 个密钥。

7.17　假设 Alice 使用 Pedersen-van Heyst 签名方案，$p = 3467$，$\alpha = 4$，$a_0 = 1567$ 以及 $\beta = 514$（当然，Alice 不知道 a_0 的值）。

(a)使用 $a_0 = 1567$ 的事实，决定所有可能的密钥

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$

使得 $\text{sig}_K(42) = (1118, 1449)$。

(b)假设 $\text{sig}_K(42) = (1118, 1449)$，$\text{sig}_K(969) = (899, 471)$。在不使用 $a_0 = 1567$ 这一事实的情况下，决定 K 的值(它表示该方案是一次签名)。

7.18　假设 Alice 正在使用 Pedersen-van Heyst 签名方案，$p = 5087$，$\alpha = 25$，$\beta = 1866$。假设密钥为

$$K = (5065, 5076, 144, 874, 1873, 2345)$$

现在，假设 Alice 关于消息 4785 的签名(2219,458)是伪造的。

(a)证明该伪造签名满足验证条件，因此是一个有效签名。

(b)给定这个伪造签名的情况下，说明 Alice 怎样计算伪造的证据 a_0。

参 考 文 献

[1] 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** (1988), 194–209.

[2] R. ANDERSON (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 809, Springer-Verlag, 1994.

[3] H. ANTON. *Elementary Linear Algebra, Eighth Edition*. John Wiley and Sons, 2000.

[4] E. BACH AND J. SHALLIT. *Algorithmic Number Theory, Volume 1: Efficient Algorithms*. The MIT Press, 1996.

[5] F. L. BAUER. *Decrypted Secrets, Methods and Maxims of Cryptology, Second Edition*. Springer-Verlag, 2000.

[6] P. BEAUCHEMIN AND G. BRASSARD. A generalization of Hellman's extension to Shannon's approach to cryptography. *Journal of Cryptology*, **1** (1988), 129–131.

[7] P. BEAUCHEMIN, G. BRASSARD, C. CRÉPEAU, C. GOUTIER AND C. POMERANCE. The generation of random numbers that are probably prime. *Journal of Cryptology*, **1** (1988), 53–64.

[8] H. BEKER AND F. PIPER. *Cipher Systems, The Protection of Communications*. John Wiley and Sons, 1982.

[9] M. BELLARE. Practice-oriented provable-security. In *Lectures on Data Security*, pages 1–15. Springer-Verlag, 1999.

[10] M. BELLARE (ED.) *Advances in Cryptology – CRYPTO 2000 Proceedings. Lecture Notes in Computer Science*, vol. 1880, Springer-Verlag, 2000.

[11] M. BELLARE, R. CANETTI AND H. KRAWCZYK. Keying hash functions for message authentication. *Lecture Notes in Computer Science*, **1109** (1996), 1–15. (Advances in Cryptology – CRYPTO '96.)

[12] M. BELLARE, R. GUERIN AND P. ROGAWAY. XOR MACs: new methods for message authentication using finite pseudorandom functions. *Lecture Notes in Computer Science*, **963** (1995), 15–28. (Advances in Cryptology – CRYPTO '95.)

[13] M. BELLARE, J. KILIAN AND P. ROGAWAY. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, **61** (2000), 362–399.

[14] M. BELLARE AND P. ROGAWAY. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[15] M. BELLARE AND P. ROGAWAY. Optimal asymmetric encryption. *Lecture Notes in Computer Science*, **950** (1995), 92–111. (Advances in Cryptology – EUROCRYPT '94.)

[16] M. BELLARE AND P. ROGAWAY. The exact security of digital signatures: how to sign with RSA and Rabin. *Lecture Notes in Computer Science*, **1070**

(1996), 399–416. (Advances in Cryptology – EUROCRYPT '96.)

[17] M. BELLARE, S. GOLDWASSER AND D. MICCIANCIO. "Pseudo-random" number generation within cryptographic algorithms: the DSS case. *Lecture Notes in Computer Science*, **1294** (1997), 277–292. (Advances in Cryptology – CRYPTO '97.)

[18] T. BETH (ED.) *Cryptography Proceedings, 1982. Lecture Notes in Computer Science*, vol. 149, Springer-Verlag, 1983.

[19] T. BETH, N. COT AND I. INGEMARSSON (EDS.) *Advances in Cryptology: Proceedings of EUROCRYPT '84. Lecture Notes in Computer Science*, vol. 209, Springer-Verlag, 1985.

[20] A. BEUTELSPACHER. *Cryptology*. Mathematical Association of America, 1994.

[21] E. BIHAM (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 1267, Springer-Verlag, 1997. (FSE '97.)

[22] E. BIHAM AND A. SHAMIR. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, **4** (1991), 3–72.

[23] E. BIHAM AND A. SHAMIR. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[24] E. BIHAM AND A. SHAMIR. Differential cryptanalysis of the full 16-round DES. *Lecture Notes in Computer Science*, **740** (1993), 494–502. (Advances in Cryptology – CRYPTO '92.)

[25] J. BLACK, S. HALEVI, H. KRAWCZYK, T. KROVETZ AND P. ROGAWAY. UMAC: fast message authentication via optimized universal hash functions. *Lecture Notes in Computer Science*, **1666** (1999), 234–251. (Advances in Cryptology – CRYPTO '99.)

[26] I. BLAKE, G. SEROUSSI AND N. SMART. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[27] G. R. BLAKLEY AND D. CHAUM (EDS.) *Advances in Cryptology: Proceedings of CRYPTO '84. Lecture Notes in Computer Science*, vol. 196, Springer-Verlag, 1985.

[28] D. BLEICHENBACHER AND U. M. MAURER. Directed acyclic graphs, one-way functions and digital signatures. *Lecture Notes in Computer Science*, **839** (1994), 75–82. (Advances in Cryptology – CRYPTO '94.)

[29] M. BLUM AND S. GOLDWASSER. An efficient probabilistic public-key cryptosystem that hides all partial information. *Lecture Notes in Computer Science*, **196** (1985), 289–302. (Advances in Cryptology – CRYPTO '84.)

[30] D. BONEH. The decision Diffie-Hellman problem. *Lecture Notes in Computer Science*, **1423** (1998), 48–63. (Proceedings of the Third Algorithmic Number Theory Symposium.)

[31] D. BONEH. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, **46** (1999), 203–213.

[32] D. BONEH. Simplified OAEP for the RSA and Rabin functions. *Lecture Notes in Computer Science*, **2139** (2001), 275–291. (Advances in Cryptology – CRYPTO 2001.)

[33] D. BONEH AND G. DURFEE. Cryptanalysis of RSA with private key d

less than $N^{0.292}$. *IEEE Transactions on Information Theory*, **46** (2000), 1339–1349.

[34] J. N. E. BOS AND D. CHAUM. Provably unforgeable signatures. *Lecture Notes in Computer Science*, **740** (1993), 1–14. (Advances in Cryptology – CRYPTO '92.)

[35] C. BOYD, (ED.) *Advances in Cryptology – ASIACRYPT 2001 Proceedings. Lecture Notes in Computer Science*, vol. 2248, Springer-Verlag, 2001.

[36] G. BRASSARD (ED.) *Advances in Cryptology – CRYPTO '89 Proceedings. Lecture Notes in Computer Science*, vol. 435, Springer-Verlag, 1990.

[37] G. BRASSARD AND P. BRATLEY. *Fundamentals of Algorithmics*. Prentice Hall, 1995.

[38] R. P. BRENT. An improved Monte Carlo factorization method. *BIT*, **20** (1980), 176–184.

[39] D. M. BRESSOUD AND S. WAGON. *A Course in Computational Number Theory*. Springer-Verlag, 2000.

[40] E. F. BRICKELL (ED.) *Advances in Cryptology – CRYPTO '92 Proceedings. Lecture Notes in Computer Science*, vol. 740, Springer-Verlag, 1993.

[41] J. A. BUCHMANN. *Introduction to Cryptography*. Springer-Verlag, 2001.

[42] J. L. CARTER AND M. N. WEGMAN. Universal classes of hash functions. *Journal of Computer and System Sciences*, **18** (1979), 143–154.

[43] S. CAVALLAR, B. DODSON, A. LENSTRA, W. LIOEN, P. MONTGOMERY, B. MURPHY, H. TE RIELE, K. AARDAL, J. GILCHRIST, G. GUILLERM, P. LEYLAND, J. MARCHAND, F. MORAIN, A. MUFFETT, C. PUTNAM, C. PUTNAM AND P. ZIMMERMANN. Factorization of a 512-bit RSA modulus. *Lecture Notes in Computer Science*, **1807** (2000), 1–18 (Advances in Cryptology – EUROCRYPT 2000.)

[44] F. CHABAUD AND A. JOUX. Differential collisions in SHA-0. *Lecture Notes in Computer Science*, **1462** (1998), 56–71. (Advances in Cryptology – CRYPTO '98.)

[45] F. CHABAUD AND S. VAUDENAY. Links between differential and linear cryptanalysis. *Lecture Notes in Computer Science*, **950** (1995), 356–365. (Advances in Cryptology – EUROCRYPT '94.)

[46] M. CHATEAUNEUF, A. C. H. LING AND D. R. STINSON. Slope packings and coverings, and generic algorithms for the discrete logarithm problem. Preprint.

[47] D. CHAUM (ED.) *Advances in Cryptology: Proceedings of CRYPTO '83*. Plenum Press, 1984.

[48] D. CHAUM AND W. L. PRICE (EDS.) *Advances in Cryptology – EUROCRYPT '87 Proceedings. Lecture Notes in Computer Science*, vol. 304, Springer-Verlag, 1988.

[49] D. CHAUM, R. L. RIVEST AND A. T. SHERMAN (EDS.) *Advances in Cryptology: Proceedings of CRYPTO '82*. Plenum Press, 1983.

[50] D. CHAUM AND H. VAN ANTWERPEN. Undeniable signatures. *Lecture Notes in Computer Science*, **435** (1990), 212–216. (Advances in Cryptology – CRYPTO '89.)

[51] D. COPPERSMITH. The data encryption standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, **38** (1994), 243–250.

[52] D. COPPERSMITH (ED.) *Advances in Cryptology – CRYPTO '95 Proceedings. Lecture Notes in Computer Science*, vol. 963, Springer-Verlag, 1995.

[53] J. DAEMEN, L. KNUDSEN AND V. RIJMEN. The block cipher Square. *Lecture Notes in Computer Science*, **1267** (1997), 149–165. (Fast Software Encryption '97.)

[54] J. DAEMEN AND V. RIJMEN. The block cipher Rijndael. *Lecture Notes in Computer Science*, **1820** (2000), 288-296. (Smart Card Research and Applications.)

[55] J. DAEMEN AND V. RIJMEN. *The design of Rijndael. AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[56] I. B. DAMGÅRD. A design principle for hash functions. *Lecture Notes in Computer Science*, **435** (1990), 416–427. (Advances in Cryptology – CRYPTO '89.)

[57] I. B. DAMGÅRD (ED.) *Advances in Cryptology – EUROCRYPT '90 Proceedings. Lecture Notes in Computer Science*, vol. 473, Springer-Verlag, 1991.

[58] I. DAMGÅRD, P. LANDROCK AND C. POMERANCE. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, **61** (1993), 177–194.

[59] D. W. DAVIES (ED.) *Advances in Cryptology – EUROCRYPT '91 Proceedings. Lecture Notes in Computer Science*, vol. 547, Springer-Verlag, 1991.

[60] J. M. DELAURENTIS. A further weakness in the common modulus protocol for the RSA cryptosystem. *Cryptologia*, **8** (1984), 253–259.

[61] B. DEN BOER AND A. BOSSELAERS. An attack on the last two rounds of MD4. *Lecture Notes in Computer Science*, **576** (1994), 194–203. (Advances in Cryptology – CRYPTO '92.)

[62] B. DEN BOER AND A. BOSSALAERS. Collisions for the compression function of MD5. *Lecture Notes in Computer Science*, **765** (1992), 293–304. (Advances in Cryptology – EUROCRYPT '94.)

[63] D. E. R. DENNING. *Cryptography and Data Security*. Addison-Wesley, 1982.

[64] A. DE SANTIS (ED.) *Advances in Cryptology – EUROCRYPT '94 Proceedings. Lecture Notes in Computer Science*, vol. 950, Springer-Verlag, 1995.

[65] Y. G. DESMEDT (ED.) *Advances in Cryptology – CRYPTO '94 Proceedings. Lecture Notes in Computer Science*, vol. 839, Springer-Verlag, 1994.

[66] W. DIFFIE. The first ten years of public-key cryptography. In *Contemporary Cryptology, The Science of Information Integrity*, pages 135–175. IEEE Press, 1992.

[67] W. DIFFIE AND M. E. HELLMAN. Multiuser cryptographic techniques. *Federal Information Processing Standard Conference Proceedings*, **45**

(1976), 109–112.

[68] W. DIFFIE AND M. E. HELLMAN. New directions in cryptography. *IEEE Transactions on Information Theory*, **22** (1976), 644–654.

[69] H. DOBBERTIN. Cryptanalysis of MD4. *Journal of Cryptology*, **11** (1998), 253–271.

[70] T. ELGAMAL. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, **31** (1985), 469–472.

[71] A. ENGE. *Elliptic Curves and their Applications to Cryptography: an Introduction*. Kluwer Academic Publishers, 1999.

[72] *Data Encryption Standard (DES)*. Federal Information Processing Standard Publication 46, 1977.

[73] *DES Modes of Operation*. Federal Information Processing Standard Publication 81, 1980.

[74] *Guidelines for Implementing and Using the NBS Data Encryption Standard*. Federal Information Processing Standard Publication 74, 1981.

[75] *Computer Data Authentication*. Federal Information Processing Standard Publication 113, 1985.

[76] *Secure Hash Standard*. Federal Information Processing Standard Publication 180, 1993.

[77] *Secure Hash Standard*. Federal Information Processing Standard Publication 180-1, 1995.

[78] *Secure Hash Standard*. Federal Information Processing Standard Publication 180-2 (Draft), 2001.

[79] *Digital Signature Standard*. Federal Information Processing Standard Publication 186, 1994.

[80] *Digital Signature Standard*. Federal Information Processing Standard Publication 186-2, 2000.

[81] *Advanced Encryption Standard*. Federal Information Processing Standard Publication 197, 2001.

[82] J. FEIGENBAUM (ED.) *Advances in Cryptology – CRYPTO '91 Proceedings. Lecture Notes in Computer Science*, vol. 576, Springer-Verlag, 1992.

[83] H. FEISTEL. Cryptography and computer privacy. *Scientific American*, **228**(5) (1973), 15–23.

[84] N. FERGUSON, J. KELSEY, S. LUCKS, B. SCHNEIER, M. STAY, D. WAGNER AND D. WHITING. Improved cryptanalysis of Rijndael. *Lecture Notes in Computer Science*, **1978** (2001), 1213–230. (Fast Software Encryption 2000.)

[85] N. FERGUSON, R. SCHROEPPEL AND D. WHITING. A simple algebraic representation of Rijndael. *Lecture Notes in Computer Science*, **2259** (2001), 103–111. (Selected Areas in Cryptography 2001.)

[86] A. FIAT AND A. SHAMIR. How to prove yourself: practical solutions to identification and signature problems. *Lecture Notes in Computer Science*, **263** (1987), 186–194. (Advances in Cryptology – CRYPTO '86.)

[87] E. FUJISAKI, T. OKAMOTO, D. POINTCHEVAL AND J. STERN. RSA-OAEP is secure under the RSA assumption. *Lecture Notes in Computer Science*, **2139** (2001), 260–274. (Advances in Cryptology – CRYPTO 2001.)

[88] W. FUMY (ED.) *Advances in Cryptology – EUROCRYPT '97 Proceedings. Lecture Notes in Computer Science*, vol. 1233, Springer-Verlag, 1997.

[89] P. GARRETT. *Making, Breaking Codes: An Introduction to Cryptology.* Prentice-Hall, 2001.

[90] J. VON ZUR GATHEN AND J. GERHARD. *Modern Computer Algebra.* Cambridge University Press, 1999.

[91] J. K. GIBSON. Discrete logarithm hash function that is collision free and one way. *IEE Proceedings-E*, **138** (1991), 407–410.

[92] E. N. GILBERT, F. J. MACWILLIAMS AND N. J. A. SLOANE. Codes which detect deception. *Bell Systems Technical Journal*, **53** (1974), 405–424.

[93] C. M. GOLDIE AND R. G. E. PINCH. *Communication Theory.* Cambridge University Press, 1991.

[94] S. GOLDWASSER (ED.) *Advances in Cryptology -- CRYPTO '88 Proceedings. Lecture Notes in Computer Science*, vol. 403, Springer-Verlag, 1990.

[95] S. GOLDWASSER AND S. MICALI. Probabilistic encryption. *Journal of Computer and Systems Science*, **28** (1984), 270–299.

[96] S. GOLDWASSER, S. MICALI AND P. TONG. Why and how to establish a common code on a public network. In *23rd Annual Symposium on the Foundations of Computer Science*, pages 134–144. IEEE Press, 1982.

[97] D. GOLLMANN (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 1039, Springer-Verlag, 1996.

[98] D. M. GORDON AND K. S. MCCURLEY. Massively parallel computation of discrete logarithms. *Lecture Notes in Computer Science*, **740** (1993), 312–323. (Advances in Cryptology – CRYPTO '92.)

[99] L. C. GUILLOU AND J.-J. QUISQUATER (EDS.) *Advances in Cryptology – EUROCRYPT '95 Proceedings. Lecture Notes in Computer Science*, vol. 921, Springer-Verlag, 1995.

[100] C. G. GUNTHER (ED.) *Advances in Cryptology – EUROCRYPT '88 Proceedings. Lecture Notes in Computer Science*, vol. 330, Springer-Verlag, 1988.

[101] J. HÅSTAD, A. W. SCHRIFT AND A. SHAMIR. The discrete logarithm modulo a composite hides $O(n)$ bits. *Journal of Computer and Systems Science*, **47** (1993), 376–404.

[102] M. E. HELLMAN. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, **26** (1980), 401–406.

[103] T. HELLESETH (ED.) *Advances in Cryptology – EUROCRYPT '93 Proceedings. Lecture Notes in Computer Science*, vol. 765, Springer-Verlag, 1994.

[104] H. M. HEYS. *A Tutorial on Linear and Differential Cryptanalysis.* Technical report CORR 2001-17, Dept. of Combinatorics and Optimization,

University of Waterloo, Waterloo, Canada, 2001.

[105] H. M. HEYS AND S. E. TAVARES. Substitution-permutation networks resistant to differential and linear cryptanalysis. *Journal of Cryptology*, **9** (1996), 1–19.

[106] H. IMAI, R. L. RIVEST AND T. MATSUMOTO (EDS.) *Advances in Cryptology – ASIACRYPT '91 Proceedings. Lecture Notes in Computer Science*, vol. 739, Springer-Verlag, 1993.

[107] D. JOHNSON, A. MENEZES AND S. VANSTONE. The elliptic curve digital signature algorithm (ECDSA). *International Journal on Information Security*, **1** (2001), 36–63.

[108] P. JUNOD. On the complexity of Matsui's attack. *Lecture Notes in Computer Science*, **2259** (2001), 199–211. (Selected Areas in Cryptography 2001.)

[109] D. KAHN. *The Codebreakers.* Scribner, 1996.

[110] B. KALISKI, JR. (ED.) *Advances in Cryptology – CRYPTO '97 Proceedings. Lecture Notes in Computer Science*, vol. 1294, Springer-Verlag, 1997.

[111] C. KAUFMAN, R. PERLMAN AND M. SPECINER. *Network Security. Private Communication in a Public World.* Prentice Hall, 1995.

[112] L. KELIHER, H. MEIJER AND S. TAVARES. New method for upper bounding the maximum average linear hull probability for SPNs. *Lecture Notes in Computer Science*, **2045** (2001), 420–436. (Advances in Cryptology – EUROCRYPT 2001.)

[113] L. KELIHER, H. MEIJER AND S. TAVARES. Improving the upper bound on the maximum average linear hull probability for Rijndael. *Lecture Notes in Computer Science*, **2259** (2001), 112–128. (Selected Areas in Cryptography 2001.)

[114] J. KILIAN (ED.) *Advances in Cryptology – CRYPTO 2001 Proceedings. Lecture Notes in Computer Science*, vol. 2139, Springer-Verlag, 2001.

[115] K. KIM AND T. MATSUMOTO (EDS.) *Advances in Cryptology – ASIACRYPT '96 Proceedings. Lecture Notes in Computer Science*, vol. 1163, Springer-Verlag, 1996.

[116] R. KIPPENHAHN. *Code Breaking, A History and Exploration.* Overlook Press, 1999.

[117] L. R. KNUDSEN. Contemporary block ciphers. *Lecture Notes in Computer Science*, **1561** (1999), 105–126. (Lectures on Data Security.)

[118] L. R. KNUDSEN (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 1636, Springer-Verlag, 1999. (FSE '99.)

[119] N. KOBLITZ. Elliptic curve cryptosystems. *Mathematics of Computation*, **48** (1987), 203–209.

[120] N. KOBLITZ. *A Course in Number Theory and Cryptography, Second Edition.* Springer-Verlag, 1994.

[121] N. KOBLITZ (ED.) *Advances in Cryptology – CRYPTO '96 Proceedings. Lecture Notes in Computer Science*, vol. 1109, Springer-Verlag, 1996.

[122] N. KOBLITZ, A. MENEZES AND S. VANSTONE. The state of elliptic curve cryptography. *Designs. Codes and Cryptography*, **19** (2000), 173–193.

[123] A. G. KONHEIM. *Cryptography, A Primer.* John Wiley and Sons, 1981.

[124] H. KRAWCZYK, (ED.) *Advances in Cryptology – CRYPTO '98 Proceedings. Lecture Notes in Computer Science*, vol. 1462, Springer-Verlag, 1998.

[125] K. KUROSAWA, T. ITO AND M. TAKEUCHI. Public key cryptosystem using a reciprocal number with the same intractability as factoring a large number. *Cryptologia*, **12** (1988), 225–233.

[126] X. LAI, J. L. MASSEY AND S. MURPHY. Markov ciphers and differential cryptanalysis. *Lecture Notes in Computer Science*, **547** (1992), 17–38. (Advances in Cryptology – EUROCRYPT '91.)

[127] K. Y. LAM, E. OKAMOTO AND C. XING (EDS.) *Advances in Cryptology – ASIACRYPT '99 Proceedings. Lecture Notes in Computer Science*, vol. 1716, Springer-Verlag, 1999.

[128] S. LANDAU. Standing the test of time: the data encryption standard. *Notices of the AMS*, **47** (2000), 341–349.

[129] S. LANDAU. Communications security for the twenty-first century: the advanced encryption standard. *Notices of the AMS*, **47** (2000), 450–459.

[130] A. K. LENSTRA. Integer factoring. *Designs, Codes and Cryptography*, **19** (2000), 101–128.

[131] A. K. LENSTRA AND H. W. LENSTRA, JR. (EDS.) *The Development of the Number Field Sieve. Lecture Notes in Mathematics*, vol. 1554. Springer-Verlag, 1993.

[132] A. K. LENSTRA AND H. W. LENSTRA, JR. Algorithms in number theory. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 673–715. Elsevier Science Publishers, 1990.

[133] A. K. LENSTRA AND E. R. VERHEUL. Selecting cryptographic key sizes. *Journal of Cryptology*, **14** (2001), 255–293.

[134] R. LIDL AND H. NIEDERREITER. *Finite Fields, Second Edition.* Cambridge University Press, 1997.

[135] D. L. LONG AND A. WIGDERSON. The discrete log hides $O(\log n)$ bits. *SIAM Jounal on Computing*, **17** (1988), 363–372.

[136] J. C. A. VAN DER LUBBE. *Basic Methods of Cryptography.* Cambridge, 1998.

[137] M. MATSUI. Linear cryptanalysis method for DES cipher. *Lecture Notes in Computer Science*, **765** (1994), 386–397. (Advances in Cryptology – EUROCRYPT '93.)

[138] M. MATSUI. The first experimental cryptanalysis of the data encryption standard. *Lecture Notes in Computer Science*, **839** (1994), 1–11. (Advances in Cryptology – CRYPTO '94.)

[139] U. MAURER (ED.) *Advances in Cryptology – EUROCRYPT '96 Proceedings. Lecture Notes in Computer Science*, vol. 1070, Springer-Verlag, 1996.

[140] U. MAURER AND S. WOLF. The Diffie-Hellman protocol. *Designs, Codes and Cryptography*, **19** (2000), 147–171.

[141] R. MCELIECE. *Finite Fields for Computer Scientists and Engineers.* Kluwer Academic Publishers, 1987.

[142] A. J. MENEZES. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[143] A. J. MENEZES, T. OKAMOTO AND S. A. VANSTONE. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, **39** (1993), 1639–1646.

[144] A. J. MENEZES AND S. A. VANSTONE (EDS.) *Advances in Cryptology – CRYPTO '90 Proceedings. Lecture Notes in Computer Science*, vol. 537, Springer-Verlag, 1991.

[145] A. J. MENEZES, P. C. VAN OORSCHOT AND S. A. VANSTONE. *Handbook of Applied Cryptography.* CRC Press, 1996.

[146] R. C. MERKLE. One way hash functions and DES. *Lecture Notes in Computer Science*, **435** (1990), 428–446. (Advances in Cryptology – CRYPTO '89.)

[147] G. L. MILLER. Riemann's hypothesis and tests for primality. *Journal of Computer and Systems Science*, **13** (1976), 300–317.

[148] V. MILLER. Uses of elliptic curves in cryptography. *Lecture Notes in Computer Science*, **218** (1986), 417–426. (Advances in Cryptology – CRYPTO '85.)

[149] C. J. MITCHELL, F. PIPER AND P. WILD. Digital signatures. In *Contemporary Cryptology, The Science of Information Integrity*, pages 325–378. IEEE Press, 1992.

[150] R. A. MOLLIN. *An Introduction to Cryptography.* Chapman & Hall/CRC, 2001.

[151] J. H. MOORE. Protocol failures in cryptosystems. In *Contemporary Cryptology, The Science of Information Integrity*, pages 541–558. IEEE Press, 1992.

[152] V. I. NECHAEV. On the complexity of a deterministic algorithm for a discrete logaritum. *Math. Zametki*, **55** (1994), 91–101.

[153] J. NECHVATAL, E. BARKER, L. BASSHAM, W. BURR, M. DWORKIN, J. FOTI AND E. ROBACK. *Report on the Development of the Advanced Encryption Standard (AES).* October 2, 2000. Available from `http://csrc.nist.gov/encryption/aes/`

[154] P. Q. NGUYEN AND I. E. SHPARLINSKI. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, to appear.

[155] K. NYBERG. Differentially uniform mappings for cryptography. *Lecture Notes in Computer Science*, **765** (1994), 55–64. (Advances in Cryptology – EUROCRYPT '93.)

[156] K. NYBERG. Linear approximation of block ciphers. *Lecture Notes in Computer Science*, **950** (1995), 439–444. (Advances in Cryptology – EUROCRYPT '94.)

[157] K. NYBERG (ED.) *Advances in Cryptology – EUROCRYPT '98 Proceedings. Lecture Notes in Computer Science*, vol. 1403, Springer-Verlag, 1998.

[158] A. M. ODLYZKO (ED.) *Advances in Cryptology – CRYPTO '86 Proceedings. Lecture Notes in Computer Science*, vol. 263, Springer-Verlag, 1987.

[159] A. M. ODLYZKO. Discrete logarithms: the past and the future. *Designs, Codes, and Cryptography*, **19** (2000), 129–145.

[160] K. OHTA AND D. PEI (EDS.) *Advances in Cryptology – ASIACRYPT '98 Proceedings. Lecture Notes in Computer Science*, vol. 1514, Springer-Verlag, 1998.

[161] T. OKAMOTO (ED.) *Advances in Cryptology – ASIACRYPT 2000 Proceedings. Lecture Notes in Computer Science*, vol. 1976, Springer-Verlag, 2000.

[162] T. P. PEDERSEN. Signing contracts and paying electronically. *Lecture Notes in Computer Science*, **1561** (1999), 134–157. (Lectures on Data Security.)

[163] R. PERALTA. Simultaneous security of bits in the discrete log. *Lecture Notes in Computer Science*, **219** (1986), 62–72. (Advances in Cryptology – EUROCRYPT '85.)

[164] B. PFITZMANN (ED.) *Advances in Cryptology – EUROCRYPT 2001 Proceedings. Lecture Notes in Computer Science*, vol. 2045, Springer-Verlag, 2001.

[165] B. PFITZMANN. *Digital Signature Schemes – General Framework and Fail-Stop Signatures. Lecture Notes in Computer Science*, vol. 1100, Springer-Verlag, 1996.

[166] F. PICHLER (ED.) *Advances in Cryptology – EUROCRYPT '85 Proceedings. Lecture Notes in Computer Science*, vol. 219, Springer-Verlag, 1986.

[167] J. PIEPRYZK AND R. SAFAVI-NAINI (EDS.) *Advances in Cryptology – ASIACRYPT '94 Proceedings. Lecture Notes in Computer Science*, vol. 917, Springer-Verlag, 1995.

[168] S. C. POHLIG AND M. E. HELLMAN. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, **24** (1978), 106–110.

[169] D. POINTCHEVAL AND J. STERN. Security arguments for signature schemes and blind signatures. *Journal of Cryptology*, **13** (2000), 361–396.

[170] J. M. POLLARD. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, **32** (1978), 918–924.

[171] C. POMERANCE (ED.) *Advances in Cryptology – CRYPTO '87 Proceedings. Lecture Notes in Computer Science*, vol. 293, Springer-Verlag, 1988.

[172] B. PRENEEL, (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 1008, Springer-Verlag, 1995.

[173] B. PRENEEL. The state of cryptographic hash functions. *Lecture Notes in Computer Science*, **1561** (1999), 158–182. (Lectures on Data Security.)

[174] B. PRENEEL (ED.) *Advances in Cryptology – EUROCRYPT 2000 Proceedings. Lecture Notes in Computer Science*, vol. 1807, Springer-Verlag, 2000.

[175] B. PRENEEL AND P. C. VAN OORSCHOT. On the security of iterated mes-

sage authentication codes. *IEEE Transactions on Information Theory*, **45** (1999), 188–199.

[176] J.-J. QUISQUATER AND J. VANDEWALLE (EDS.) *Advances in Cryptology – EUROCRYPT '89 Proceedings. Lecture Notes in Computer Science*, vol. 434, Springer-Verlag, 1990.

[177] M. O. RABIN. Digitized signatures and public-key functions as intractable as factorization. *MIT Laboratory for Computer Science Technical Report*, LCS/TR-212, 1979.

[178] M. O. RABIN. Probabilistic algorithms for testing primality. *Journal of Number Theory*, **12** (1980), 128–138.

[179] R. L. RIVEST. The MD4 message digest algorithm. *Lecture Notes in Computer Science*, **537** (1991), 303–311. (Advances in Cryptology – CRYPTO '90.)

[180] R. L. RIVEST. The MD5 message digest algorithm. Internet Network Working Group RFC 1321, April 1992.

[181] R. L. RIVEST, A. SHAMIR, AND L. ADLEMAN. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21** (1978), 120–126.

[182] K. H. ROSEN. *Elementary Number Theory and its Applications, Fourth Edition*. Addison-Wesley, 1999.

[183] R. A. RUEPPEL (ED.) *Advances in Cryptology – EUROCRYPT '92 Proceedings. Lecture Notes in Computer Science*, vol. 658, Springer-Verlag, 1993.

[184] A. SALOMAA. *Public-Key Cryptography*. Springer-Verlag, 1990.

[185] B. SCHNEIER. *Applied Cryptography, Protocols, Algorithms and Source Code in C, Second Edition*. John Wiley and Sons, 1995.

[186] B. SCHNEIER (ED.) *Fast Software Encryption. Lecture Notes in Computer Science*, vol. 1978, Springer-Verlag, 2001. (FSE 2000.)

[187] C. P. SCHNORR. Efficient signature generation by smart cards. *Journal of Cryptology*, **4** (1991), 161–174.

[188] J. SEBERRY AND J. PIEPRZYK (EDS.) *Advances in Cryptology – AUSCRYPT '90 Proceedings. Lecture Notes in Computer Science*, vol. 453, Springer-Verlag, 1990.

[189] J. SEBERRY AND Y. ZHENG (EDS.) *Advances in Cryptology – AUSCRYPT '92 Proceedings. Lecture Notes in Computer Science*, vol. 718, Springer-Verlag, 1993.

[190] C. E. SHANNON. A mathematical theory of communication. *Bell Systems Technical Journal*, **27** (1948), 379–423, 623–656.

[191] C. E. SHANNON. Communication theory of secrecy systems. *Bell Systems Technical Journal*, **28** (1949), 656–715.

[192] V. SHOUP. Lower bounds for discrete logarithms and related problems. *Lecture Notes in Computer Science*, **1233** (1997), 256–266. (Advances in Cryptology – EUROCRYPT '97.)

[193] V. SHOUP. OAEP reconsidered. *Lecture Notes in Computer Science*, **2139**

(2001), 239–259. (Advances in Cryptology – CRYPTO 2001.)

[194] J. H. SILVERMAN AND J. TATE. *Rational Points on Elliptic Curves*. Springer-Verlag, 1992.

[195] G. J. SIMMONS. A survey of information authentication. In *Contemporary Cryptology, The Science of Information Integrity*, pages 379–419. IEEE Press, 1992.

[196] G. J. SIMMONS (ED.) *Contemporary Cryptology, The Science of Information Integrity*. IEEE Press, 1992.

[197] S. SINGH. *The Code Book*. Doubleday, 1999.

[198] N. P. SMART. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, **12** (1999), 193–196.

[199] M. E. SMID AND D. K. BRANSTAD. The data encryption standard: past and future. In *Contemporary Cryptology, The Science of Information Integrity*, pages 43–64. IEEE Press, 1992.

[200] M. E. SMID AND D. K. BRANSTAD. Response to comments on the NIST proposed digital signature standard. *Lecture Notes in Computer Science*, **740** (1993), 76–88. (Advances in Cryptology – CRYPTO '92.)

[201] J. SOLINAS. Efficient arithmetic on Koblitz curves. *Designs, Codes and Cryptography*, **19** (2000), 195–249.

[202] R. SOLOVAY AND V. STRASSEN. A fast Monte Carlo test for primality. *SIAM Journal on Computing*, **6** (1977), 84–85.