普通高等教育“十五”国家级规划教材

现代密码学基础

章照止　主编

北京邮电大学出版社
·北京·

信息安全专业系列教材

编　委　会

总　　序

办好信息安全本科专业的第一要素是拥有高质量的教材。由于各方面的原因,我国开办信息安全本科专业的历史很短,刚刚起步,但是,当前以各种形式开办信息安全本科专业的高等院校却非常多,学生总数也相当可观,而且其中大部分学生已经学完基础课程,即将进入专业课的学习阶段。

与信息安全本科专业招生的火爆场面形成鲜明对比的是,到目前为止,我国还没有一套自己的信息安全本科专业系列教材。为了保证信息安全本科专业学生的培养质量,2001 年,北京市教委以“精品教材立项”的形式委托我们北京邮电大学信息安全中心负责编写《现代密码学基础》、《信息安全概论》、《网络安全》、《信息隐藏与数字水印》、《入侵检测》、《计算机病毒原理及防治》等 6 本教材,随后,教育部又将此套系列教材列入了“普通高等教育‘十五’国家级教材规划”。由此可见,此套教材的编写确实受到了各级教育主管部门的高度重视。

北京邮电大学信息安全中心是一专门从事信息安全的教学、科研和成果转化的重点实验室。该实验室已经培养出了我国第一位密码学博士,而且在“信息安全”和“密码学”两个专业领域内健全了博士后、博士、硕士和本科的培养教育体系,已经培养出了数以百计的信息安全研究生。

在接受了北京市教委和教育部的编写信息安全本科系列教材的任务之后,我们立即组织了最强的师资队伍投入到教材的编写工作之中。经过两年多的不懈努力,数易其稿,反复研讨,按照教育目标和大学生基本素质培养的要求,本着推进理工融合及学科交叉的思想,经过优化课程体系和精选课程内容,我们终于完成了信息安全本科专业系列教材的第一批教材(共 6 本)。现在我们正在着手规划信息安全本科专业的第二批教材,它们的暂定名分别是《安全操作系统》、《安全数据库》、《安全访问控制》、《安全检测与监控》、《数字证书与管理》、《安全备份与灾难恢复》、《安全隔离技术》、《安全服务技术》、《安

全系统工程》、《安全规范与标准》等。我们诚意邀请国内所有高等院校的权威安全专家加入第二批教材的编写工作(有意者请与我们直接联系。地址:100876,北京邮电大学信息安全中心126信箱)。我们希望这套信息安全本科专业系列教材最终完成之后能够基本满足国内各类高校信息安全本科专业的普遍需求。

虽然我们的目标是编写一套适合信息安全专业本科生使用的精品教材,但是,由于水平有限,时间仓促,且信息安全本科专业刚刚开始,我们还没有足够的实践机会,不足之处和错误在所难免,恳请读者和同行专家多提意见,以便我们再版时充分修改,不断完善。

衷心感谢北京邮电大学胡正名教授对本套教材的大力支持,感谢北京邮电大学信息安全中心二百余位成员的支持与配合。本套教材也是国家自然科学基金项目(90204017, 60372094, 60373059)和国家“973”项目(G1999035804)资助的成果,在此一并表示感谢。

杨义先 教授、博士生导师、全国政协委员
2004年1月于北京邮电大学信息安全中心

内 容 简 介

本书全面深入地介绍了现代密码学的基础理论。全书共分 15 章和 1 个附录。内容包括密码学研究的基本问题、古典密码学、密码学的信息论基础和计算复杂性理论基础、单向函数和伪随机序列生成器的严格理论、序列密码、分组密码和公钥密码、数字签名、杂凑函数、身份识别、认证码、密钥管理和零知识证明,附录的内容包括本书用到的代数学和初等数论方面的基础知识,每章还包括注记和习题。本书注意了严格理论和直观描述的配合,在介绍经典密码体制的同时,注意从中总结出一般的原则和方法及基本工具,并注重介绍一些新的密码体制。

本书是为信息安全专业编写的专业基础课教材,适用于高等院校信息安全本科专业的学生以及计算机应用、信息工程、应用数学等相关本科专业的学生,同时也可供从事信息安全工作的科技人员以及相关专业的研究生参考。

图书在版编目(CIP)数据

现代密码学基础/ 章照止主编 .—北京:北京邮电大学出版社,2004
ISBN 7-5635-0651-9

Ⅰ .现 ... Ⅱ .章 ... Ⅲ .密码—理论—高等学校—教材 Ⅳ .TN918 .1

中国版本图书馆 CIP 数据核字(2004)第 014350 号

书　　名:现代密码学基础
主　　编:章照止
责任编辑:王琴秋
出版发行:北京邮电大学出版社
社　　址:北京市海淀区西土城路 10 号(邮编:100876)
电话传真:010-62282185(发行部) 010-62283578(FAX)
电子信箱: publish@bupt .edu .cn
经　　销:各地新华书店
印　　刷:北京源海印刷厂印刷
开　　本:787 mm×1 092 mm 1/ 16
印　　张:18 .5
字　　数:401 千字
印　　数:1—5 000 册
版　　次:2004 年 4 月第 1 版 2004 年 4 月第 1 次印刷

ISBN 7-5635-0651-9/ TP·83　　定　价:29 .00 元

前　　言

本书是为信息安全专业编写的专业基础课教材,其选材及内容的组织安排是在参考了国内外已出版的若干同类教材的基础上,根据现代密码学的特点以及对信息安全专业学生的培养目标确定的。编入本教材的内容都是相对成熟的、公认的理论与方法。与国内已出版的这类教材相比,本教材还具有以下一些特点:

(1) 由于密码学是在密码设计者和密码分析者之间的不断斗争中发展起来的。密码分析方法的奏效将促使密码设计方法的改进,这又迫使密码分析者研究新的分析方法。因此设计一个密码体制不仅要考虑已知的分析方法,而且要考虑密码体制设计出来后可能出现的新的分析方法。故密码体制的安全性必须建立在严格的理论基础上,单凭直观和显然是不行的。为了培养学生掌握严格的信息安全理论基础,本书对现代密码学的基础——计算复杂性理论——作了比较深入的介绍,并介绍了现代密码学的源头概念——单向函数——的严格定义。本书还介绍了基于计算复杂性理论的伪随机序列生成器的严格理论,应用这种生成器输出的伪随机序列作密钥所设计的密码系统,其安全性在理论上是可证明的。关于密码学协议的主要工具——Hash 函数和零知识证明系统——的介绍,本书采用严格和直观相配合的方法,使学生更易理解和掌握。

(2) 为了培养学生的灵活思维方法,并具有一定的研究、开发和创新能力。本书的某些较难的章节可以作为选学内容,某些定理被略去的证明可以让学生自己补上。本书在各章的注记中还提供了补充阅读的文献,可以作为学生的自学内容,以扩大知识面。本书每章还提供了一些习题,其中有一些较容易的旨在巩固学习内容或锻炼计算能力的题,也有一些较难的旨在培养研究和创新能力的题。

(3) 为了适应通信技术和计算技术迅速发展对密码学提出的新的要求,本书在介绍经典的密码体制时,注意了从中总结出一般的原则和方法以及基本工具,并注重介绍一些新提出的密码体制,如 AES。

量子密码是一个新方向,但这方面的技术还处于实验室阶段,要达到应用还有较长时间,因此本书不介绍量子密码学的内容。本书参考文献[33]6.3节对量子密码学作了简短介绍,想了解此方面内容的读者可参看该文献及其所引文献。

密码学要用到的数学知识很多,特别是概率论、代数学和数论方面的基础知识。学习本书要求学生具有一定的概率论基础知识。本书附录简要地介绍了书中用到的有关代数学和初等数论方面的基础知识,其他数学知识(如纠错码、素性检验、椭圆曲线、图论等)只能在有关章节中作简短的介绍和说明。学习一些这方面的数学知识对学习本书是有帮助的;但是,没有学过这些数学知识也不会使本书的学习受到影响。

本书共分15章和一个附录。第1～4章介绍密码学的基础知识,主要包括密码学研究的基本问题、古典密码学、密码学的信息论基础和计算复杂性理论基础;第5,6,11和15章介绍设计密码体制和安全协议的主要工具,包括单向函数和伪随机序列生成器、杂凑(Hash)函数和零知识证明;第7～9章介绍各类密码体制及其攻击(分析)方法,包括序列(流)密码、分组密码和公钥密码;第10,12和13章分别介绍了数字签名、身份识别、认证码;第14章介绍了密钥管理;附录中介绍了代数和数论的若干基础知识。

目前,密码学教材和文献中所用的术语和符号还很不统一,本书各章之间所引用的术语和符号也没有完全统一。为了避免混淆,必要时通过在括号中对术语加注英语符号加以说明。此外,本书未列出所有参考文献,只在书的最后列出了本书各章所用的主要参考文献。

本书的编写工作由杨义先和温巧燕负责组织,由温巧燕草拟了本书的编写原则和章节目录。章照止编写了书稿的第1,4,5,6,11,13和15章,周世祥编写了书稿的第2,3,12和14章以及附录,并对温巧燕准备的书稿第7,8两章的素材进行了整理和补充,邓玉峰根据徐国爱的意见编写了书稿的第9,10两章,章照止对全书作了一些必要的修改。此外,周世祥和邓玉峰还在本书的电子版及书稿打印方面做了许多工作;张龙在校对书稿等方面做了许多工作;张吉力帮助作者回答了书稿中的一个疑问。作者对他们表示衷心的感谢。

作　者

2004年1月

目　　录

第 7 章　序列密码

第 8 章　分组密码

第 10 章　数字签名

第 11 章　杂凑(Hash)函数

第 12 章　身份识别方案

第 1 章　引　　论

1.1　密码学研究的基本问题

在历史上(1975 年以前),为了通过公开的通信媒体(如电话)进行秘密通信,密码学研究的问题仅限于设计和分析密码体制(通称密码系统)。但从 1975 年以来,构造不能伪造的数字签名问题和设计抗欺骗协议(fault-tolerant protocol)问题也被包括在现代密码学的研究范围内。总之,设计和分析任何安全协议(为抗拒参与者各方内部或外部可能有不诚实者为了达到种种目的而进行的恶意破坏)的问题都被认为是现代密码学研究的领域。因此,如何度量或评价一个密码系统(包括协议)的安全性则成为密码学研究的一个重要问题。现有两种定义“安全性”的方法:一种是基于信息论的方法(经典方法);另一种是基于计算复杂性理论的方法(现代方法)。为了构造安全的密码系统,有几个基本工具是非常有用的,它们是单向函数、伪随机序列生成器、零知识证明和杂凑函数(Hash function)。对于上述问题和涉及的有关概念,在给出它们的正式定义之前,先给出一个简短的基于直观的说明,以使读者对密码学要研究的问题有一个直观的了解。

1.1.1　密码体制

通过不安全的通信媒介进行安全通信问题是密码学研究的最基本的问题。它的背景是两个参与者通过一个信道(如电话)进行通信,可能被第三者(称为搭线者)搭线窃听。通信双方希望相互交换信息而让搭线者对信息内容尽可能无知。不严格地说,一个密码体制(有时也称加密方案)是一个使通信双方能进行秘密通信的协议。一个典型的加密方案包括两个算法:一个称为加密算法,由发方用来发送消息;另一个称为解密算法,由收方用来接收消息。因此,为了发送一个消息,发方首先用加密算法处理消息,发送处理结果,称为密文。收到密文后,收方用解密算法将密文恢复为原始消息,称为明文。为使这一方案能提供秘密通信,通信双方(至少收方)必须知道某些搭线者不知道的东西,否则搭线者也能像收方一样地恢复消息。这个外加知识的形式,可以是解密算法本身或它的某些参数和(或)辅助输入,称这个外加知识为解密密钥。不失一般性,可设搭线者知道加密算法

和解密算法,解密算法需要两个输入,即密文和解密密钥。要注意的是,存在一个搭线者不知道的解密密钥只是提供秘密通信的一个必要条件。

如何估价一个密码体制的安全性是件要仔细推敲的事。前面已提到现有两种定义“安全性”的方法。基于信息论的定义是用密文中是否蕴含明文的信息作为标准。不严格地说,若密文中不含明文的任何信息,则认为该密码体制是安全的,否则就认为是不安全的。已经证明,达到这样高等级(完善)的安全性,仅当所用密钥的长度不短于加密的发送消息的总长度才有可能。这种安全性称为无条件安全性,即无论搭线者有多少计算资源,他也不能从截取到的密文恢复出明文。但这种密码体制的应用受到严重的限制,特别是需要发送大量秘密信息的情形。

基于计算复杂性理论的安全性定义则不考虑密文中是否蕴含明文的信息,而是考虑这些信息是否能有效地被提取出来。换句话说,把搭线者提取明文信息的可能性改为搭线者提取明文信息的可行性,这种安全性称为有条件安全性,即搭线者在一定的计算资源条件下,不能从密文恢复出明文。已经证明,为达到基于计算复杂性理论定义的安全性,所用密钥长度可以比加密的发送消息的总长度短得多。例如,可用伪随机序列生成器将短的随机密钥扩展为较长的伪随机密钥,用它构造的密码体制具有与用长度相当的随机密钥构造的密码体制同样的安全性。

更有意思的是,用基于计算复杂性理论的方法可以引入一些新的概念和源头,这些概念和源头在信息论方法下是不可能存在的,典型的例子是公钥密码体制。前面集中讨论了解密算法和解密密钥。可以证明,加密算法除了输入发送消息外,还必须有一个辅助输入(依赖于解密密钥),这个辅助输入称为加密密钥。在经典的密码体制中,特别是在1980年以前所用的所有密码体制中,加密密钥与解密密钥是相同的,称这类密码体制为私钥(对称)密码体制。在这类密码体制中,搭线者必须不知道加密密钥,从而产生密钥分发问题,即如何使通信双方得到同样的加解密密钥。传统的方法是用比公开信道代价高得多的安全信道交换密钥。在公钥密码体制中,加密密钥可以公开让大家知道而并不损害密码体制的安全性。当然,加密密钥与解密密钥是不同的,且要求从加密密钥计算解密密钥是不可行的,这就自然地解决了密钥分发问题。

密码分析是研究密码体制的破译问题,即试图在不知道密钥的情况下,从截取到的密文恢复出明文消息或密钥,这正好是搭线者想做的事情。同时,密码体制的设计者和用户也应关心密码分析问题,因为对一个具体的密码体制的分析结果是评价这一体制安全性的一种检验。根据密码分析者可能取得的分析资料的不同,密码分析(或称攻击)可分为下列四类:

(1) 唯密文分析(攻击),密码分析者取得一个或多个用同一密钥加密的密文;

(2) 已知明文分析(攻击),除要破译的密文外,密码分析者还取得一些用同一密钥加密的明密文对;

(3) 选择明文分析(攻击),密码分析者可取得他所选择的任何明文所对应的密文(当

然,不包括他要恢复的明文),这些明密文对和要破译的密文是用同一密钥加密的;

(4) 选择密文分析(攻击),密码分析者可取得他所选择的任何密文所对应的明文(要破译的密文除外),这些密文和明文和要破译的密文是用同一解密密钥解密的,它主要应用于公钥密码体制。

1.1.2　单向函数与伪随机序列生成器

单向函数是基于计算复杂性理论的方法引入的一个最基本的新概念。不严格地说,一个单向函数是一个函数 $y = f(x)$,由 x 计算函数值 y 是容易的,但由 y 计算函数的逆 $x = f^{-1}(y)$ 是困难的(在某种平均意义下)。"容易"和"困难"的确切含意由计算复杂性理论定义。单向函数是现代密码学的一个基本工具,大部分安全的密码系统(包括协议)的构造依赖于"单向函数存在"这一假设,但单向函数的存在性至今没有证明。虽然如此,密码学界还是普遍相信单向函数是存在的,而且还给出了一些经过分析、检验的被认为是单向函数的例子。

伪随机序列生成器也是密码学的一个基本工具,在构造密码系统中起着重要作用,特别是它可构造简单的私钥密码体制(流密码)。在实际应用中,伪随机序列生成器虽然不是计算复杂性理论方法引入的新概念,在密码学文献及更广的概率计算文献中却早有研究和应用,但很少给出确切的定义,这对密码学应用是很不安全的。应用计算复杂性理论方法可给出伪随机序列生成器的一个确切定义。直观地说,一个伪随机序列生成器是一个确定算法,它把短的随机比特(种子)扩展为长得多的貌似随机的比特序列。换句话说,伪随机序列生成器的输出虽然不是真正的随机序列,但在计算资源一定的条件下,要判别这个输出与等长的真随机序列的不同是不可行的。可以证明,伪随机性与计算困难性有密切联系,因为可用单向函数来构造伪随机序列生成器。事实上,可证伪随机序列生成器的存在性和单向函数的存在性是等价的。

1.1.3　数字签名与杂凑(Hash)函数

数字签名是一个在全球计算机网发展以前不存在的概念。它是在计算机通信网上从事商贸和有关事务的环境下提出和需要研究的问题。某些参与者需要在他们发送的电子文件上签名以示承担责任。当然,不能伪造的签名在几个世纪以前就有讨论,但讨论的只是手写签名而不是数字签名,而且看不出这种讨论与密码学有什么关系。

加密和签名数字化了,并引入了安全性的计算复杂性理论方法,这使它们之间可能建立起一定的关系。不严格地说,对一个不能伪造的数字签名方案有下列要求:

(1) 每个用户能有效(容易)地在他选择的文件上产生他自己的签名;

(2) 每个用户能有效(容易)地验证一给定的数字串是否是另一特定用户在某特定文件上的签名;

(3) 没人能在其他用户没签名的文件上有效(容易)地产生他们的签名。

事实上,上述要求也给出了不能伪造的手写签名的基本要求,即个人的签名能力,一个公认的验证过程和相信伪造签名要通过这一验证是不可行(困难)的。对过去已签的手写签名,很难说出它们在多大程度上达到这些基本要求,但对于现代密码学中构造的数字签名,可以严格证明它们是达到上述三个要求的。

杂凑函数也是密码学的一个基本工具,在数字签名、检验信息的完整性(未被篡改)等有关协议中有重要应用。不严格地说,杂凑函数是一个将不等长消息压缩为固定长度消息的确定性算法 h,它具有如下性质:

(1) 任给消息 x,计算 $h(x)$是容易的;

(2) 要找两个不同的消息 x_1, x_2使得 $h(x_1) = h(x_2)$是计算上不可行(困难)的。

可以证明,杂凑函数的存在性和单向函数的存在性是等价的。

1.1.4 消息认证和身份识别

消息认证问题的背景与消息加密方案的背景很相似,通信双方也在一个不安全信道上传送消息,如互联网(Internet),但现在的第三者不仅可能截取消息进行分析,而且可能伪造或篡改发送的消息,称为入侵者。通信双方希望交换消息而拒绝接受入侵者伪造或篡改的消息。不严格地说,一个不保密的认证方案是一个使通信双方能进行公开通信而不受入侵者欺骗的协议。一个典型的认证方案包括一个算法,称为认证算法,它需要两个输入,即消息和认证密钥,用它来产生消息的认证标签。为了发送一个消息,发方先将消息和认证密钥输入认证算法,其计算结果称为消息的认证标签,再将消息和认证标签一起发送;收方收到消息和认证标签后,先用同样的认证密钥和收到的消息输入认证算法,检验其计算结果是否与收到的认证标签相同。若它们相同,则认为消息未经篡改,接受该消息;反之,若它们不相同,则认为消息被入侵者篡改,拒绝接受该消息。不失一般性,可设入侵者知道认证算法,但不知道认证密钥,因此认证方案与私钥密码体制一样有密钥分发问题。对一个安全的认证方案有下列要求:

(1) 通信双方能有效(容易)地产生任意消息的认证标签;

(2) 通信双方能有效(容易)地验证一给定的数字串是否是一给定消息的认证标签;

(3) 没有入侵者能有效(容易)地产生通信双方没发送消息的消息认证标签。

和密码体制一样,认证方案的安全性也有两种定义,即基于信息论方法的定义(无条件安全性)和基于计算复杂性理论的定义(有条件安全性)。

还要指出,消息认证与数字签名有某种相似性,所不同的只是消息认证不要求通信双方以外的任何人能有效地验证认证标签的真实性,但数字签名却要求签名者以外的其他人都能有效地验证签名的真实性。因此,数字签名方案也是消息认证问题的解,但消息认证方案不必构成数字签名方案。

另一个与数字签名和消息认证有某种相似性的问题是身份识别。身份识别也是在计算机通信和各种电子服务系统广泛应用的环境下提出和需要研究的问题。用户方需要向

提供服务方证明自己的身份而使提供服务方(可能是计算机)能验证是真正用户还是假冒该用户的入侵者。当然,某些简单的身份识别方法(如通行字)早有应用,但这些方法很不安全,不严格地说,一个安全的身份识别方案有下列要求:

(1) 每个用户能有效地提供自己的身份证明信息;

(2) 每个提供服务者能有效地验证提供身份证明信息者的身份;

(3) 没人(包括服务提供者)能有效地提供其他人的身份证明信息。

容易看出,身份识别与数字签名有某种相似性。事实上,在理论上有一种标准的方法可将每一个身份识别方案转化为一个数字签名方案。

1.1.5　抗欺骗协议和零知识证明

从前面的讨论可以看出,在某种意义上,数字签名、消息认证和身份识别等问题可以看作是保密通信问题的推广。提出这些问题的共同背景是在系统的参与者各方的内部或外部可能存在不诚实者,他们为了各自的目的用非法(欺骗)的行为有意进行破坏。研究这些问题的共同目标是要限制这些不诚实者的非法(欺骗)行为能达到目的。具体地说,就是:

(1) 保密通信或消息认证问题中,通信双方属于系统的参与者内部,搭线者或入侵者为系统外部的不诚实者,使用加密方案是为了限制搭线者提取明文信息,使用消息认证方案是为了限制入侵者用假消息欺骗接收者;

(2) 在数字签名问题中,所有用户都认为是属于系统内部的参与者,其中可能有不诚实者,使用数字签名方案是为了限制任何不诚实用户伪造其他用户的签名,将自己的文件强加于其他用户;

(3) 身份识别问题中,所有用户和提供服务者都属于系统内部的参与者,其中可能有不诚实者,使用身份识别协议是为了限制不诚实用户冒充其他用户的身份。

因此,数字签名引入密码学拓广了密码学的研究范围。在这种拓广的意义下,密码学关心任何希望限制不诚实者达到目的的问题。抗欺骗协议(fault-tolerant protocol)就是对这类问题的一般处理。抗欺骗协议又称密码学协议,简称安全协议。下面对两个典型的安全协议作以下一些说明:

(1) 同时交换秘密问题,签订合同是一特例。其背景是甲乙双方各掌握一个秘密,如合同上自己的数字签名。他们希望共同执行一个协议。若双方都按协议的规定执行,则当协议结束时双方都得到对方的秘密,且在任何情况下(即使一方欺骗另一方)甲方得到乙方秘密当且仅当乙方得到甲方的秘密。完善的同时交换秘密仅当有可信的第三方参加才能达到。事实上,只要双方都将自己的秘密通过安全信道传给可信第三方,当可信者收到他们的秘密后,再将甲方的秘密传给乙方,乙方的秘密传给甲方。这样解决有两个问题:一是在任何情况下(即使甲乙双方都是诚实的)总要有系统外部的可信者参加;二是在某些应用中根本不存在这样的可信者。安全协议就是更现实的解决方案。

(2) 函数的多方计算问题。这一问题的背景是由多个局部输入计算一个函数,每个输入由不同的用户掌握。投票问题是其特例,即计算一个多变量的布尔函数 $f(x_1, x_2, \cdots, x_n)$。每个变量 x_i 是一个局部输入代表用户 i 投的票,$x_i = 0$ 表示赞成,$x_i = 1$ 表示反对。函数值 $f = 0$ 表示赞成票占多数,$f = 1$ 表示反对票占多数。不严格地说,对一个安全的函数多方计算协议有下列要求:

Ⅰ.秘密性。任何用户不能得到超过函数值所能导出的关于其他用户输入的信息。

Ⅱ.鲁棒性。任何用户只能通过选择自己的输入来影响函数值。

有时要求这两个条件是对小的用户团体而不是对单个用户成立。

显然,若存在一个可信中心,则上述问题很容易解决。只要所有用户将自己的输入通过安全信道传给可信中心,可信中心再用所有收到的局部输入计算出函数值传给所有用户,然后抹去所有中间结果(包括所有收到的输入),但存在这样的可信中心是不现实的。安全协议就是更现实的解决方案。

零知识证明是构造安全协议的一个主要工具。不严格地说,零知识证明除了给出结论的正确性外,不给出任何其他信息,它提供一个工具使各方能放心地执行所给协议。今举一例来说明零知识证明的作用:甲方收到乙方发给他的一个加密消息,他要将消息的最低位发给丙方,显然,若甲方仅发消息的最低位给丙,则丙无法得知甲是否欺骗他,甲为了证明他没欺骗丙,可将整个加密消息和解密密钥也发给丙,但这样做大大超出了丙所要求的信息。一个好得多的方案是甲发给丙一位信息和一个断言该位信息是消息的最低位的零知识证明。关于零知识证明的存在条件将在以后讨论。

以上是密码学中的基本问题和基本概念的直观描述,也是本书将要正式定义和详细讨论的问题和概念。当然,从这些基本问题还可以发展出许多其他问题,这里无法一一提到。

1.2 密码学的广泛应用

前面已经提到,在公钥密码出现以前(1978 年以前)所用的密码体制都是私钥密码体制。由于应用这种密码代价昂贵,因此密码学主要应用于军事、政府和外交等机要通信的保密和识破别国的密码。当时密码学几乎是国家安全机构独占的领域,对于企业和个人,既无密码学的专业知识又无足够的财力来保护自己的秘密。1976 年,Diffie 和 Hellman 发表了论文《密码学的新方向》,提出了公钥密码体制的新思想,证明在发方与收方之间不需要传送密钥的保密通信是可能的。这一发明引发了科技界对研究密码学的极大兴趣。大量的密码学论文开始公开发表,改变了过去只是少数人关起门来研究密码学的状况。为了适应计算机通信和电子商务迅速发展的需要,密码学的研究领域逐渐从消息加密扩大到数字签名、消息认证、身份识别、抗欺骗协议等新课题。同时,密码学的应用也不再局限于军事、政治和外交,而扩大到商务、金融和社会各个领域,特别是全球范围的互联网

(Internet)的出现和发展,为人们提供了快速、高效和廉价的通信,大量敏感信息(如考试成绩、个人简历、体检结果、实验数据等)常常要通过互联网进行交换。现代电子商务也是以互联网为基础。由于互联网的开放性,任何人都可以自由地接入互联网,使得有些不诚实者就有可能采用各种非法手段进行破坏。因此人们十分关心在网络上交换信息的安全性。普遍认为密码学方法是解决信息安全保护的一个最有效的方法。事实上,现在网络上应用的保护信息安全的技术(如数据加密技术、数字签名技术、消息认证与身份识别技术、防火墙技术以及反病毒技术等)都是以密码学为基础的。电子商务中应用各种支付系统(如智能卡)也是基于密码学来设计的,可以说密码学是信息安全技术的基础。由此可见,现代密码学的应用非常广泛。现在任何求业、单位和个人都可以应用密码学来保护自己的信息安全。由于在当今高度信息化的社会里,信息安全关系到国家全局和长远利益以及安全,各国政府都十分重视密码学的研究和应用。美国国家标准局(NBS)首先制定并于 1977 年向世界公布了美国数据加密标准(DES)。1994 年又公布了美国国家标准技术研究所(NIST)提出的一个数字签名标准。1997 年,美国国家标准技术研究所又在全世界公开征集高级加密标准(AES)的活动,通过公布 15 个候选加密方案进行公开的评论和专家讨论,最后从中选出了一个方案作为 AES。AES 将成为新的美国数据加密标准和一个供全球免费使用的数据加密标准。最近,欧洲委员会的信息社会技术(IST)规划中出资 33 亿欧元支持一项新欧洲数字签名、信息完整性和加密方案(NESSIE)工程,目标是推出一套密码标准,包括分组密码、流密码、杂凑函数、消息认证码、数字签名和公钥密码等。所有这些密码标准的研究和公布对密码学的研究和广泛应用起了积极推动作用,以后将对这些密码标准作详细介绍。

1.3　本书选材的组织与安排

从前两节的说明中可以看出,密码学这门科学具有不同于一般科学的特点,这些特点是:

(1) 密码学是在密码设计者和密码分析者之间不断的斗争中发展起来的。密码分析方法的奏效将促使密码设计方法的改进;这又将迫使密码分析者研究出新的分析方法。因此设计一个密码体制是件不容易的工作,不仅要考虑已知的分析方法,而且要考虑密码体制设计出来后可能出现的新的分析方法。因此密码体制的安全性要建立在严格的理论基础上,密码学概念要有严格的定义,结论要有严格的证明,只凭直观和显然是很危险的。

(2) 密码学的发展与通信技术和计算技术的发展密切相关。因为密码体制的安全性基于破译者的计算能力,计算能力提高了,原来不能破译的密码体制变得可能破译了。如较简单的古典密码体制在没有计算机的时代曾被应用,但有了快速的计算机以后,这些密码体制就不再适用了。又如,DES 经过 20 多年的破译研究和计算技术的发展,现在要由

更强的密码体制——AES——来代替。再如,量子计算机和量子通信的研究引起了量子密码学的研究,由于这方面的技术还处于实验室阶段,要到达应用还有较长的时间。因此本书不包括量子密码学的内容。但重点要放在介绍新的和有应用价值的密码体制,更重要的是,要从已有的密码系统的设计中总结出一般的原则和方法以及基本工具,这样才能适应通信技术和计算技术迅速发展对密码学提出的新的要求。

(3) 密码学是密码体制和安全协议(包括它们的分析和破译方法)的总体,不同的密码体制的设计所用的数学知识可能属于不同的数学分支,有的还涉及到高深的数学理论,如椭圆曲线、整数分解等,因此密码学要用到的数学知识很多,除了密码学的信息论基础和计算复杂性理论基础作专章介绍外,其他数学知识只能在有关章节和附录中作简短介绍。关于密码体制的分类,按照传统分为私钥密码体制和公钥密码体制,按加密方式的不同又分为序列密码和分组密码。

根据密码学的上述特点,本书的选材、组织与安排如下:全书共分 15 章和一个附录。第 1～4 章介绍密码学的基本知识和理论基础,包括密码学研究的基本问题、古典密码体制、密码学的信息论基础和密码学的计算复杂性理论基础;第 5,6,11,15 章介绍密码学的基本工具,包括单向函数、伪随机序列生成器、杂凑函数和零知识证明;第 7～10,12～14 章主要介绍有代表性的和有应用价值的密码体制和安全协议,包括序列密码、分组密码、公钥密码、数字签名、身份识别、认证码、密钥管理。附录为数学基础。

注 记

本章的目的是为学习现代密码学基础做准备,主要介绍了密码学研究的基本问题、密码学的广泛应用和密码学的特点。主要参考了本书参考文献[1]和[26],在 Shannon 的密码学信息理论(见文献[2])建立以前的密码学发展历史可参阅文献[5]和[33]的 1.1 节。近年来国内外出版了很多本密码学教材,如文献[1],[6],[7],[8],[9],[26],[33],[34]等。本书的选材与组织安排是在参考上述教材的基础上,考虑到密码学的特点以及对信息安全专业学生的要求确定的。

习 题 一

1 密码学研究的主要问题是什么?

2 密码系统安全性的定义有几种?它们的含义是什么?

3 密码分析可分为哪几类,它们的含义是什么?

4 单向函数的直观含义是什么,它为什么重要?

5 就你所知,密码学有哪些应用?

第 2 章　古典密码学

古典密码学是现代密码学的渊源,这些密码大多比较简单,用手工或机械操作即可实现加解密。1917 年《Scientific American》上声称 Vigenère 密码是不可破译的,今天演示这个结论的不真实,只是密码课上的一个练习。然而,研究这些密码的原理,对于理解、构造和分析现代密码都是十分有益的。

2.1　古典密码体制

2.1.1　定义和分类

密码学最初是为保密的目的而设置的。通信双方(文献中常提到作为 Alice 和 Bob)通过一个不安全的信道通信,若 Oscar 是一个窃听者,要求其不能理解所截取的消息。例如,这个信道可以是电话线或计算机网。Alice 想送给 Bob 的消息称作明文(Plaintext),可能是英语单词、数据或其他符号——它的结构完全是任意的。Alice 用预先指定的密钥(Key)加密(Encryption)明文,得到相应的密文(Ciphertext),并通过信道发送给 Bob。Oscar 通过搭线窃听到密文,却无法确定明文是什么。但接收者 Bob,因知道解密密钥,可以解密密文并重构明文。发送者加密消息时所采用的一组规则称作加密算法(Encryption Algorithm)。接收者对密文解密时所采用的一组规则称为解密算法(Decryption Algorithm)。加密和解密算法的操作通常是在一组密钥的控制下进行的,分别称为加密密钥和解密密钥。传统密码体制所用的加密密钥和解密密钥相同,或本质上等同,即从一个易于得到另一个,所以也称其为单钥或对称密码体制 (One-key or Symmetric Cryptosystem)。所有的古典密码体制都是单钥密码体制。这里值得一提的是,密码学中术语“系统”或“体制”(System)、“方案”(Scheme)和“算法”(Algorithm)本质上是一回事。

可用下列数学定义更正式地描述这个概念:

定义 2.1　一个密码系统(Cryptosystem)是一个五元组(P, C, K, E, D),满足条件:

(1) P 是可能明文的有限集(明文空间);

(2) C 是可能密文的有限集(密文空间);

(3) K 是一切可能密钥构成的有限集(密钥空间);

(4) 任意 $k \in K$,有一个加密算法 $e_k \in E$ 和相应的解密算法 $d_k \in D$,使得 $e_k: P \to C$ 和 $d_k: C \to P$ 分别为加密、解密函数,满足 $d_k(e_k(x)) = x$,这里,$x \in P$。

其中,特性(4)是关键,说明了如果一个明文 x 是用 e_k 加密的,且得到相应的密文 y,随后只要用 d_k 解密,就可获得起初的明文 x,也就是说,e_k 和 d_k 的作用相互抵消。显然,每个加密函数 e_k 一定是个双射(一对一)函数,否则在一个模棱两可的情况下,解密无法进行。例如,$y = e_k(x_1) = e_k(x_2)$,$x_1 \neq x_2$,则 Bob 没有办法知道 y 应当解密成 x_1 还是 x_2。

明文常常是指编码后的消息,前面提到现实世界中的消息可以是任何形式,如文本、声音、视频等,为处理需要,人们用编码和解码的知识翻译任何信息,最常见的是翻译成数字序列。这一步不带任何隐藏信息的目的。如在一些古典密码体制中,26 个英文字母常被抽象为 0～25 间的整数。

注意,如果明文空间与密文空间相同,每个加密函数是一个置换(Permutation),即是说,加密函数的作用仅仅是重排这个集合。

Alice 和 Bob 将采用下列方案具体化一个密码系统:

首先,他们选择一个随机密钥 $k \in K$。为了不被 Oscar 知道,他们可以在一起做这一步,或者如果他们在不同的地方,这时只能通过另一个安全的信道协商密钥。在以后某个时间,假定 Alice 要通过一个不安全的信道传递消息给 Bob。可以把这个消息表示为一个字符串 $x = x_1 x_2 \cdots x_n$,对某个 $n \geq 1$,其中每个明文符号 $x_i \in P$,$1 \leq i \leq n$。每个 x_i 在预先确定的密钥 k 的作用下,用加密函数 e_k 加密。因此 Alice 计算 $y_i = e_k(x_i)$,$1 \leq i \leq n$ 并传送所得的密文串 $y = y_1 y_2 \cdots y_n$。当 Bob 收到密文 y 后,他用解密函数 d_k 解密,获得起初的明文串 x。

为便于理解,下面用图 2 .1 示意。

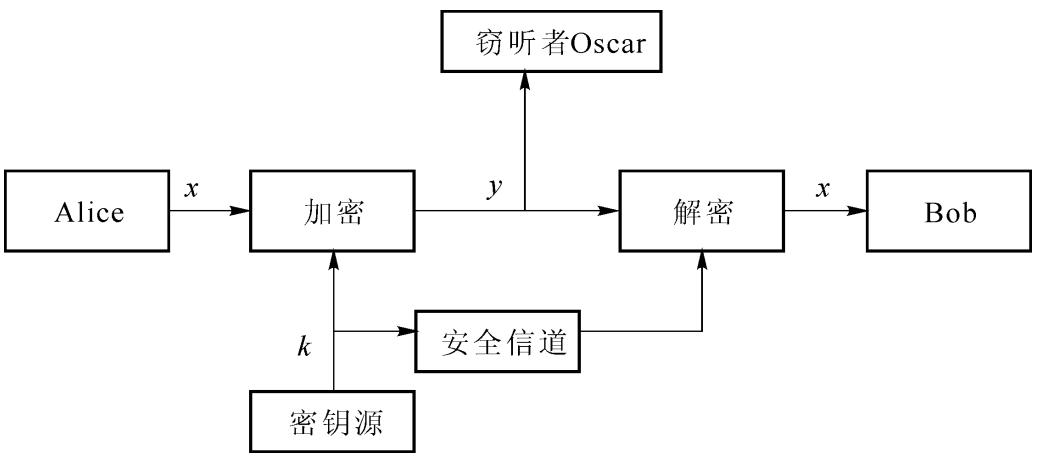


图 2 .1　Alice 与 Bob 的通信过程

如果一个密码系统是实用的,它将满足某种特性。下面正式地列举出这些特性中的两个:

(1) 每个加密函数 e_k 和每个解密函数 d_k 应当能有效地被计算。

(2) 即使看到密文串 y,窃听者 Oscar 确定所用的密钥 k 或明文串 x 是不可行的。

第一个特性是说合法用户应当很“容易”地使用系统,这里,“容易”的含义可参考第 4 章复杂性理论(假定用户有合理的计算力)。

第二个特性以非常含糊的方式定义了“安全”的想法。已知密文串 y 的情况下试图计算密钥 k 的过程称为密码分析(Cryptanalysis)。注意,如果 Oscar 能决定 k,则他能和 Bob 一样用 d_k 解密 y。因此,决定 k 至少应当和决定密文串 x 一样困难。这里的不可行可用“计算上不可行”来替代。假定入侵者也是有一定的计算力的。

非常古老的分类是代换(Substitution)系统和置换(Permutation)系统,亦称为代换密码和置换密码。代换密码又分为单字母代换(Monogram Substitution)密码和多字母代换(Polygram Substitution)密码;单字母代换密码又分为单表代换(Monoalphabetic Substitution)密码和多表代换(Polyalphabetic Substitution)密码。分类的原则不是根据密码系统质量(好或坏),而是根据他们设计的内部特性。另一个自然的分类来自形式语言理论,分系统为无上下文的(Context-free)和上下文敏感的(Context-sensitive),在前者中单个字母被加密,后者中一组字母被加密,这碰巧对应于单字母和多字母代换。

2.1.2　代换密码(Substitution Cipher)

令 Γ 表示明文字母表,内有 q 个“字母”或“字符”。例如可以是普通的英文字母 $A\sim Z$,也可以是数字、空格、标点符号或任意可以表示明文消息的符号。如前所述,可以将 Γ 抽象地表示为一个整数集 $Z_q=\{0,1,\cdots,q-1\}$。在加密时通常将明文消息划分成长为 L 的消息单元,称为明文组,以 m 表示,如 $m=(m_0,m_1,\cdots,m_{L-1})$,$m_l\in Zq, 0\leq l\leq L-1$。$m$ 也称作 L-报文,它可以被视为定义在 Z_q^L 上的随机变量

$$Z_q^L=Z_q\times Z_q\times\cdots\times Z_q(L\text{ 个})=\{m=(m_0,m_1,\cdots,m_{L-1})\mid m_l\in Z_q,0\leq l\leq L-1\}。$$

$L=1$ 为单字母报(1-gram),$L=2$ 为双字母报(digrams),$L=3$ 为三字母报(trigrams)。这时明文空间 $P=Z_q^L$。

令 Γ' 表示 q' 个“字母”或“字符”的密文字母表,抽象地可用整数集 $Z_{q'}=\{0,1,\cdots,q'-1\}$。密文单元或组为 $c=(c_0,c_1,\cdots,c_{L'-1})(L'\text{个})$,$c_{l'}\in Z_{q'}, 0\leq l'\leq L'-1$。$c$ 是定义在 $Z_q^{L'}$ 上的随机变量。密文空间 $C=Z_q^{L'}$。

一般地,明文和密文由同一字母表构成,即 $\Gamma=\Gamma'$。

代换密码可以看作是从 Z_q^L 到 $Z_q^{L'}$ 的映射。$L=1$ 时,称为单字母代换,也称为流密码(Stream Cipher)。$L>1$ 时,称为多字母代换,亦称分组密码(Block Cipher)。

一般地,选择相同明文和密文字母表。此时,若 $L=L'$,则代换映射是一一映射,密码无数据扩展。若 $L<L'$,则有数据扩展,可将加密函数设计成一对多的映射,即明文组可以找到多于一个密文组来代换,这称为多名(或同音)代换密码(Homophonic Substitution Cipher)。若 $L>L'$,则明文数据被压缩,此时代换映射不可能构成可逆映射,从而密文有时也就无法

完全恢复出原明文消息,因此保密通信中必须要求 $L \leq L'$。但 $L > L'$的映射可以用在认证系统中。

在 $\Gamma = \Gamma'$, $q = q'$, $L = 1$ 时,若对所有明文字母都用一种固定的代换进行加密,则称这种密码为单表代换。若用一个以上的代换表进行加密,称作多表代换。这是古典密码中的两种重要体制,曾被广泛地使用过。

1.单表代换密码

单表代换密码是对明文的所有字母都用一个固定的明文字母表到密文字母表的映射,即 $f: Z_q \rightarrow Z_q$。令明文 $m = m_0\ m_1 \cdots$,则相应的密文为 $c = e_k(m) = c_0\ c_1 \cdots = f(m_0)\ f(m_1) \cdots$。下面分别介绍几类简单的单表代换密码。

(1) 移位密码(Shift Cipher)

用图 2.2 表示移位密码。因为英文字符有 26 个字母,可以建立英文字母和模 26 的剩余之间的对应关系(见表 2.1)。对于英文文本,则明文、密文空间都可定义为 Z_{26}。当然很容易推广到 n 个字母的情况。容易看出,移位满足密码系统的定义,即 $d_k(e_k(x)) = x$,对每个 $x \in Z_{26}$。

设$P=C=K=Z_{26}$,对$0 \leqslant k \leqslant 25$,定义

$e_k(x)=x+k \mod 26$ 且

$d_k(y)=y-k \mod 26 \quad (x,y \in Z_{26})$

图 2.2 移位密码

表 2.1 英文字母和模 26 的剩余之间的对应关系

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13
O	P	Q	R	S	T	U	V	W	X	Y	Z		
14	15	16	17	18	19	20	21	22	23	24	25		

如果明文字母和密文字母被数字化,且各自表示为 x, y,则对每个明文 $x \in Z_{26}$,被加密为 $y = x + k \mod 26$。mod 26 意味着等式左右两边仅仅相差一个 26 的倍数。

例 2.1 凯撒(Caesar)密码是 $k = 3$ 的情况,即通过简单的向右移动源字母表 3 个字母,则形成如下代换字母表(表 2.2):

表 2 .2　代换字母表

Γ:	a	b	c	d	e	f	g	h	i	j	k	l	m
Γ′:	D	E	F	G	H	I	J	K	L	M	N	O	P

n	o	p	q	r	s	t	u	v	w	x	y	z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

若明文为: please　　confirm　　receipt

则密文为: SOHDVH　FRQILUP　UHFHLSW

我们观察到移位密码是极不安全的(mod 26),**因为它可被穷举密钥搜索所分析**:**仅有** 26 **个可能的密钥**,**尝试每一个可能的解密规则** d_k,**直到一个有意义的明文串被获得。平均地说**,**一个明文在尝试** 26/ 2 = 13 **解密规则后将显现出来。**

可以设想:**如果密文字母表是用随机的次序放置**,**而不是简单地相应于源字母表的偏移**,**密钥量将大幅度增加。这就是下面要介绍的替换密码。**

(2) **替换密码**

另一个众所周知的密码系统是替换密码。这个密码系统已被用了好几百年。定义见图 2 3。

设$P=C=Z_{26}$，密钥空间K由所有可能的26个符号0，1，…，25的置换组成。对每一个置换 $\pi \in K$定义

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

则

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

其中 π^{-1}是 π 的逆置换。

图 2 3　**替换密码**

置换 π**表示为**:

$$\pi=\begin{matrix}0 & 1 & 2 & \cdots & 23 & 24 & 25\\ 0' & 1' & 2' & \cdots & 23' & 24' & 25'\end{matrix}。$$

替换密码的密钥是由 26 **个字母的置换组成。这些置换的数目是** 26 !,**超过** 4 .0×10^{26},**一个非常大的数。这样**,**即使对现代计算机来说**,**穷举密钥搜索也是不可行的。然而**,**下面会看到**,**替换密码容易被其他的分析方法所破译。**

显然,**替换密码的密钥**(26 **个元素的随机置换**)**太复杂而不容易记忆**,**因此实际中密钥句子常被使用。密钥句子中的字母被依次填入密文字母表**(**重复的字母只用一次**),**未用的字母按自然顺序排列。**

例 2 2　密钥句子为:the message was transmitted an hour ago

源字母表为: a b c d e f g h i j k l m n o p q r s t u v w x y z

代换字母表为: THEMSAGWRNIDOUBCFJKLPQVXYZ

明文: please confirm receipt

密文: CDSTKS EBUARJO JSESRCL

(3) 仿射密码

Caesar 密码可能的密钥数太小,而且,从安全的角度看,另一个很大的不利是:在代换后的字母系统中,字母的次序仍未改变,仅起始位置发生改变。下面介绍的仿射系统(见图 2.4)就不存在这个弱点。

在仿射密码中,我们用形如

$$e_k(x) = ax + b \mod 26 \quad a, b \in Z_{26}$$

的加密函数,这些函数被称为仿射函数,所以命名为仿射密码(注意,当 $a = 1$ 时,为移位密码)。

设$P=C=Z_{26}$，且 $K=\{(a,b)\in Z_{26}\times Z_{26}：\gcd(a,26)=1\}$，

对$k=(a,b)\in K$，定义

$e_k(x)=ax+b \mod 26$ 且$d_k(y)=a^{-1}(y-b)\mod 26$

$(x,y\in Z_{26})$

图 2.4 仿射密码

为了解密是可能的,必须要求仿射函数是双射。换句话说,对任何 $y\in Z_{26}$,我们要使得同余方程 $ax + b\equiv y \pmod{26}$ 有惟一的解。数论知识告诉我们,当且仅当 $\gcd(a, 26) = 1$(gcd 函数表示两个数的最大公因子)时,上述同余方程对每个 y 有惟一的解。

因为满足 $a\in Z_{26}$, $\gcd(a,26) = 1$ 的 a 只有 12 种候选,对参数 b 没有要求,所以仿射密码有 $12\times 26 = 312$ 种可能的密钥。

例 2.3 假定 $k = (7,3)$, $7^{-1} \bmod 26 = 15$,加密函数为 $e_k(x) = 7x + 3$,则相应的解密函数为 $d_k(y) = 15(y - 3) = 15y - 19$,其中所有的运算都是在 Z_{26} 中。容易验证,

$$d_k(e_k(x)) = d_k(7x + 3) = 15(7x + 3) - 19 = x + 45 - 19 = x。$$

加密明文 hot。首先转化这三个字母分别为数字 7, 14 和 19。然后加密

$$\begin{array}{ccccccc} 7 & & 3 & & 0 & & A \\ 7 & 14 & + & 3 & = & 23 & = & X \pmod{26}。 \\ 19 & & 3 & & 6 & & G \end{array}$$

密文串为 AGX。留下解密作为练习。

至此,可得出结论:通常,上述所介绍的代换方式(单表代换)不是非常抗密码攻击的,因为语言的特征仍能从密文中提取出来。可以通过运用不止一个代换表来进行代换,从而掩盖了密文的一些统计特征。与单表代换相对应,称其为多表代换密码。

2.多表代换密码

多表代换密码是以一系列(两个以上)代换表依次对明文消息的字母进行代换的加密方法。令明文字母表为 Z_q, $f=(f_1, f_2, \cdots)$为代换序列,明文字母序列 $x=x_1\,x_2\cdots$,则相应的密文字母序列为 $c=e_k(x)=f(x)=f_1(x_1)\,f_2(x_2)\cdots$。若 f 是非周期的无限序列,则相应的密码称为非周期多表代换密码。这类密码,对每个明文字母都采用不同的代换表(或密钥)进行加密,称作一次一密密码(One-time pad cipher),这是一种理论上惟一不可破的密码(见第 3 章)。这种密码完全可以隐蔽明文的特点,但由于需要的密钥量和明文消息长度相同而难于广泛使用。为了减少密钥量,在实际应用中多采用周期多表代换密码,即代换表个数有限,重复地使用。

有名的多表代换密码有 Vigenère、Beaufort、Running-Key、Vernam 和转轮机(Rotor Machine)等密码。下面介绍 Vigenère 密码(见图 2.5)。

设m是某固定的正整数，定义$P=C=K=(Z_{26})^m$，对一个密钥$k=(k_1,k_2,\cdots,k_m)$，定义

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

且

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

所有的运算都在$\mathbf{Z}_{26}$中。

图 2.5　Vigenère 密码

Vigenère 密码是由法国密码学家 Blaise de Vigenère 于 1858 年提出的,它是一种以移位代换(当然也可以用一般的字母代换表)为基础的周期代换密码。

称 $k=(k_1,\cdots,k_m)$为长为 m 的密钥字(Key Word)。密钥量为 26^m,所以对一个相当小的值 m,穷举密钥法也需要很长的时间。若 $m=5$,则密钥空间大小超过 1.1×10^7,手工搜索也不容易。当明文串的长度大于 m 时,可将明文串按 m 一组分段,然后再逐段使用密钥字 k。

在 Vigenère 密码中,一个字母可被映射到 m 个可能的字母之一(假定密钥字包含 m 个不同的字符),所以分析起来比单表代换更困难。

例 2.4　设 $m=6$,且密钥字是 CIPHER,这相应于密钥 $k=(2,8,15,7,4,17)$。假定明文串是

this cryptosystem is not secure 。

首先将明文串转化为数字串,按 6 个一组分段,然后模 26“加”上密钥字得:

19	7	8	18	2	17
2	8	15	7	4	17
21	15	23	25	6	8

24	15	19	14	18	24
2	8	15	7	4	17
0	23	8	21	22	15

18	19	4	12	8	18
2	8	15	7	4	17
21	1	19	19	12	9

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

20	17	4
2	8	15
22	25	19

相应的密文串将是:

VPXZGIAXIVWPUBTTMJPWIZITWZT

解密过程与加密过程类似,不同的只是进行模 26 减,而不是模 26 加。

3 .多字母代换密码

在这部分,我们介绍一种多字母系统,即 Hill 密码(见图 2 .6)。这个密码是 1929 年由 S .Hill 提出的。

设m是某个固定的正整数，$P=C=(Z_{26})^m$，又设$\boldsymbol{K}=\{Z_{26}$上的$m\times m$可逆阵$\}$；对任意$\boldsymbol{k}\in\boldsymbol{K}$，定义$e_k(x)=x\boldsymbol{k}$,则$d_k(y)=y\boldsymbol{k}^{-1}$。

其中，所有的运算都是在Z_{26}中进行。

图 2 .6　Hill 密码

多字母代换密码的特点是每次对 $L>1$ 个字母进行代换,这样做的优点是容易将字母的自然频度隐蔽或均匀化而有利于抗统计分析。

可以看出,当 $m=1$ 时,系统退化为单字母仿射代换密码,可见 Hill 密码是仿射密码体制的推广。

例如,如果 $m=2$,可以将明文写为 $x=(x_1, x_2)$,密文写为 $y=(y_1, y_2)$。这里,y_1,

y_2 都将是 x_1, x_2 的线性组合。若取 $y_1 = 11x_1 + 3x_2$, $y_2 = 8x_1 + 7x_2$,简记为 $y = x\boldsymbol{k}$,其中 $\boldsymbol{k} = \begin{matrix} 11 & 8 \\ 3 & 7 \end{matrix}$ 为密钥。

熟悉线性代数的读者将意识到可用矩阵 $\boldsymbol{k}^{-1}$ 来解密,此时的解密公式为 $x = y\boldsymbol{k}^{-1}$。可以验证

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

当然,要记住的是,运算是在 Z_{26} 中进行的。

除了 m 取很小的值($m = 2, 3$)时,计算 $\boldsymbol{k}^{-1}$ 没有有效的方法,所以大大限制了它的广泛应用,但对密码学的早期研究很有推动作用。

例 2.5　假定密钥是 $\begin{matrix} 11 & 8 \\ 3 & 7 \end{matrix}$,则 $\boldsymbol{k}^{-1} = \begin{matrix} 7 & 18 \\ 23 & 11 \end{matrix}$。现在加密明文 july,分为两个明文组(9,20)(相应于 ju)和(11,24)(相应于 ly)。计算如下:

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

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

因此,july 的加密是 DELW。

解密过程留作读者练习。

2.1.3　置换密码(Permutation Cipher)

置换密码(见图 2.7)的想法是保持明文字符未改变,但通过重排而更改它们的位置,所以有时也称为换位密码(Transposition Cipher)。

> 设m是某固定的整数，定义$P=C=(Z_{26})^m$，且K由所有$\{1, 2\cdots, m\}$的置换组成。对一个密钥π(即一个置换)，定义
>
> $$e_\pi(x_1, x_2, \cdots, x_m) = (x_{\pi(1)}, x_{\pi(2)}, \cdots, x_{\pi(m)}),$$
>
> 且
>
> $$d_\pi(y_1, y_2, \cdots, y_m) = (y_{\pi^{-1}(1)}, y_{\pi^{-1}(2)}, \cdots, y_{\pi^{-1}(m)}),$$
>
> 其中，π^{-1}是π的逆置换。

图 2.7　置换密码

例 2.6　假定 $m = 6$,密钥是以下置换 π:

$$\begin{matrix} \underline{1} & \underline{2} & \underline{3} & \underline{4} & \underline{5} & \underline{6} \\ 3 & 5 & 1 & 6 & 4 & 2 \end{matrix},$$

则逆置换 π^{-1} 为:

1	2	3	4	5	6
3	6	1	5	2	4

现在,假定给出明文

shesellsseashellsbytheseashore

首先把明文分为 6 个字母一组:

shesel lsseas hellsb ythese ashore

每 6 个字母按 π 重排,得密文:

EESLSHSALSESLSHBLEHSYEETHRAEOS

用 π^{-1} 类似地解密。

事实上,置换密码是 Hill 密码的特例。对于一个给定集 $1,2,\cdots,m$ 的置换 π,可以定义相应的 $m\times m$ 置换阵 $\boldsymbol{k}_\pi=(k_{i,j})$,依据公式,

$$k_{i,j}=\begin{cases}1 & \text{如果 } j=\pi(i)\\ 0 & \text{否则}\end{cases}。$$

(置换阵是每一行和每一列刚好有一个元素“1”,其余元素都为“0”的矩阵)

对于上述置换 π,相关的置换阵为 $\boldsymbol{k}_\pi=\begin{matrix}0&0&1&0&0&0\\0&0&0&0&0&1\\0&0&0&1&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&0&0&1\end{matrix}$,

对于置换 π^{-1},相关的置换阵为 $\boldsymbol{k}_{\pi^{-1}}=\begin{matrix}1&0&0&0&0&0\\0&0&0&0&1&0\\0&1&0&0&0&0\\0&0&0&1&0&0\end{matrix}$。

可以验证 $\boldsymbol{k}_\pi\boldsymbol{k}_{\pi^{-1}}=\boldsymbol{I}$,说明 $\boldsymbol{k}_{\pi^{-1}}=\boldsymbol{k}_\pi^{-1}$。

因此,用矩阵 $\boldsymbol{k}_\pi$ 的 Hill 密码加密等价于用置换 π 的置换密码加密;此时 Hill 密码解密等价于用逆置换 π^{-1} 的置换密码解密。

2.2 古典密码体制分析

密码史表明,密码分析者的成就似乎比密码设计者的成就更令人惊叹!许多开始时

被设计者吹为‘百年或千年难破’的密码,没过多久就被密码分析者巧妙地攻破了。在第二次世界大战中,美军破译了日本的紫密,使得日本在中途岛战役中大败。一些专家估计,同盟军在密码破译上的成功至少使第二次世界大战缩短了 8 年。

在这部分,我们讨论一些密码分析技巧。一般的假定是:攻击方知道所用的密码系统。这个假设被称为 Kerckhoff 假设。当然,如果攻击方不知道所用的密码体制,这将使得任务更加艰巨:分析者不得不尝试新的密码系统,但这时程序的复杂性基本上与限定在一个具体密码系统上相同。所以我们不想把系统的安全性基于对手不知道所用的系统。因此我们的目标是设计一个在 Kerckhoff 假设下达到安全的系统。

简单的单表代换密码(如移位密码)极易破译。仅统计标出最高频度字母再与明文字母表字母对应决定出移位量,就差不多得到正确解了。一般的仿射密码要复杂些,但多考虑几个密文字母统计表与明文字母统计表的匹配关系也不难解出。另外单表代换密码(如移位密码)也很容易用穷举密钥搜索来破译。可见,一个密码系统安全的一个必要条件是密钥空间必须足够大,使得穷举密钥搜索破译是不可行的,但这不是一个密码系统安全的充分条件。

多表代换密码的破译要比单表代换密码的破译难得多,因为在单表代换下,字母的频度、重复字母模式、字母结合方式等统计特性除了字母名称改变外,都未发生变化,依靠这些不变的统计特征就能破译单表代换,而在多表代换下,原来明文中的这些特性通过多个表的平均作用而被隐藏了起来。已有的事实表明,用唯密文攻击法分析单表和多表代换密码是可行的,但用唯密文攻击法分析多字母代换密码(如 Hill 密码)是比较困难的。分析多字母代换多用已知明文攻击法。

人类语言是高度冗余的,许多分析技巧用到了英语语言统计特性。通过对大量的小说、杂志、新闻报纸等汇编统计,人们已经获得 26 个字母的概率分布(见表 2 .3)。

表 2 .3　26 个英文字母的概率分布

字母	概率	字母	概率	字母	概率	字母	概率
A	0 .082	H	0 .061	O	0 .075	V	0 .010
B	0 .015	I	0 .070	P	0 .019	W	0 .023
C	0 .028	J	0 .002	Q	0 .001	X	0 .001
D	0 .043	K	0 .008	R	0 .060	Y	0 .020
E	0 .127	L	0 .040	S	0 .063	Z	0 .001
F	0 .022	M	0 .024	T	0 .091		
G	0 .020	N	0 .067	U	0 .028		

基于以上概率,可以把 26 个字母分为以下五组:

(1) E,有概率大约 0 .12;

(2) T, A, O, I, N, S, H, R,每个有概率在 0 .06～0 .09 间;

(3) D, L,每个有概率大约 0 .04;

(4) C, U, M, W, F, G, Y, P, B,每个有概率在 0 .015～0 .023 之间;

(5) V,K,J,X,Q,Z,每个概率少于 0 .01。

应该强调的是,这些表并不包含结论性的信息。字母的分布大大依赖于明文文本的类型:诗歌、标语、科技等等,所以有些出入也是正常的。

一般说,字母 E 总是最高频的字母,T 是排在第二位置,A 或 O 排在第三位置,E,T,A,O,N,I,S,R,H 比任何其他字母有高得多的频率,约占英文文本的 70%。

当考虑位置特征时,字母 A,I,H 不常作为单词的结尾,而 E,N,R 出现在起始位置比终了位置更少,T,O,S 的出现在前后基本相等。当然,分组的划分破坏了一些位置特征。

2.2.1 单表代换密码分析

下面是一个简单的例子,看看我们如何用语言统计特征来分析密码。

例 2.7 假设从仿射密码获得的密文为:

FMXVEDKAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKAPRKDLYEVLRHHRH

仅有 57 个密文字母,但这对仿射密码是足够的。最高频的密文字母是:R(8 次),D(7 次),E,H,K(各 5 次),F,S,V(各 4 次)。开始,可以假定 R 是 e 的加密,且 D 是 t 的加密,因为 e 和 t 分别是两个最常见的字母。数值化后,有 $e_k(4)=17$,且 $e_k(19)=3$。回忆加密函数 $e_k(x)=ax+b$,可得到两个含两个未知量的线性方程组:

$$4a+b=17$$

$$19a+b=3$$

这个系统有惟一的解 $a=6$, $b=19$(在 Z_{26} 上)。但这是一个非法的密钥,因为 $\gcd(a,26)=2>1$,所以上面的假设有误。

我们下一个猜想可能是 R 是 e 的加密,E 是 t 的加密,得 $a=13$,又是不可能的。继续假定 R 是 e 的加密,且 K 是 t 的加密。于是产生了 $a=3$, $b=5$,这至少是一个合法的密钥。剩下的事是计算相应于 $k=(3,5)$的解密函数,然后解密密文看是否得到了有意义的英文串。容易证明这是一个有效的密钥。

最后的密文是:

algorithms are quite general definitions of arithmetic processes

2.2.2 多表代换密码分析

这部分介绍分析 Vigenère 密码的一些方法。第一步是决定密钥字的长度,用 m 表示。有两种方法可供考虑,一个是所谓的 Kasiski 测试法,另一个是重合指数法(Coincidence Index)。

Kasiski 测试法是由普鲁士军官 Kasiski 于 1863 年提出的一种重码分析法。这种方法的基本原理是:若用给定的 m 个密钥表周期地对明文字母加密,则当明文中有两个相同字母组在明文序列中间隔的字母数为 m 的倍数时,这两个明文字母组对应的密文字母组必相同。但反过来,若密文中出现两个相同的字母组,它们所对应的明文字母组未必相

同,但相同的可能性很大。如果将密文中相同的字母组找出来,并对其相同字母数综合研究,找出它们的相同字母数的最大公因子,就有可能提取出有关密钥字的长度 m 的信息。

1 . Kasiski 测试

相同的字母序列可能出现在密文的不同地方,这些重复模式是有趣的,因为它们提供文本里周期的信息。考虑下列简单的例子:

明文: REQUESTS ADDITIONAL TEST

密钥: TELEXTEL EXTELEXTEL EXTE

密钥: CAVKTBLT EUQWSWJGEA LTBL

明文包含字母序列 EST 两次,而这两次又碰巧被同样的密钥部分加密,因而对应的密文都是 TBL。出现这种情况反映了下列事实:序列 EST 位于密钥长度(或周期)的倍数处。清楚地,相同字母序列的距离可告诉我们关于加密文本的周期的一些信息。

例如,一个给定密文包含下列重复的序列,且有距离:

字母序列	距　离
PQA	$150 = 2 \times 5^2 \times 3$
RET	$42 = 7 \times 2 \times 3$
FRT	$10 = 2 \times 5$
ROPY	$81 = 3^4$
DER	$57 = 19 \times 3$
RUN	$117 = 13 \times 3^2$

因为 3 是出现最频繁的因子,所以密文的周期最有可能是 3。

2 . 重合指数法

如果考虑一个来自 26 个字母表的完全随机文本,则每个字母有同样的概率发生,等于 1/ 26。假定另有第二个随机文本,把它放在第一个下面,然后计算有多大的机会找到上下两个字母相等。因为每个字母都是一个随机字符,找到两个都是 a 的概率是 $\left(\frac{1}{26}\right)^2$。显然,对于其他字母而言,这个概率是不变的,所以找到两个同样字母的总的概率是:

$$26\left(\frac{1}{26}\right)^2 = \frac{1}{26} = 0.0385。$$

然而,对英语文本,与随机文本不同,发现字母发生的概率是不同的。由表 2 .3 知,字母 A,B,C,…,Z 出现的期望概率分别为 $p_0, p_1, \cdots, p_{25}$,此时找到两个等同字母发生的概率为: $\sum_{i=0}^{25} p_i^2 = 0.065$。这个值比随机文本的情况大得多。我们把它称为重合指数。

定义 2.2 设一门语言由 n 个字母构成，每个字母 i 发生的概率为 p_i，$1 \leq i \leq n$，则重合指数是指其中两个随机元素相同的概率，记为 $CI = \sum_{i=1}^{n} p_i^2$。

这样，对一个完全随机的文本 $CI = 0.0385$，与一个有意义的英语文本 $CI = 0.065$，差异是比较明显的。

实际分析中，重合指数的计算的价值体现在几个方面。在单表代换的情况下，明文的字母被其他字母所代替，但不影响文本的统计参数，即加密后密文的重合指数仍不变，CI(明文) = CI(密文)，用这个信息可以判断文本是用单表还是用多表代换加密的。如果密文的重合指数证明较低，一个多表代换可能被应用，因为一个多表代换倾向于隐藏文本的统计参数，重合指数的值将更接近于随机文本。

另外，重合指数的估算能提供对两个不同密文的洞察力，比如接收到两段文本 C_1，C_2，如果它们是用同样方式加密的，则 $CI(C_1) \approx CI(C_2)$。

因为在实际中密文有有限长度，从密文中计算的重合指数值总是不同于理论值，所以通常用的是 CI 的估计值 CI'，合适的 $CI' = \sum_{i=1}^{n} x_i(x_i - 1)/L(L - 1)$，其中 L 代表密文长，x_i 是密文符号 i 发生的数目。可以证明，CI' 是 CI 的无偏估计值。

3. Chi 测试

Chi 测试提供一个比较两个频率分布的直接方式。计算下面的和：

$$\chi = \sum_{i=1}^{n} p_i q_i ,$$

其中，p_i 表示符号 i 在第一个分布中发生的概率，q_i 表示符号 i 在第二个分布中发生的概率。

当两个频率分布类似时，χ 的值相对要高。假定收到两个密文 C_1，C_2，它们都是 Caesar 代换的结果。设第一个代换表是通过源字母表移动 k_1 个字母得到，第二个代换表是源字母表移动 k_2 个字母得到。如果 $k_1 = k_2$，说明 C_1，C_2 是由同样 Caesar 代换加密的，这时 χ 值将大些，因为 C_1 的统计特性与 C_2 类似。反之，$k_1 \neq k_2$，χ 值将要小些。

χ 除了用来决定是否采用同样或不同的代换外，也能用来简化多表代换为单表代换。

例如，一个密钥字为 RADIO，用 Vigenère 加密的明密文如下：

明文：EXECUTE THESE COMMANDS

密钥：RADIORA DIORA DIORADIO

密文：VXHKIKE WPSJE FWADAQLG

为了还原密文到明文，我们用下面的矩阵来表示(列数相当于密钥字的长度)：

R	A	D	I	O
V	X	H	K	I

K　E　W　P　S
J　E　F　W　A
D　A　Q　L　G

可以看出,矩阵的第一列 R 下的密文字母通过“减”R 解密,第二列 A 下的密文字母通过“减”A 解密,等等。第一列和第二列的密文是两个不同的移位密码加 R 的结果。

考虑密钥字中每个字母和第一个字母 R 的相对距离如下:

R	A	D	I	O
0	9	12	17	23

现在,我们做这样的处理:把第二列所有字母提前 9 个位置,第三列所有字母提前 12 个位置,其他类似,可获得下面的文本块:

V　O　V　T　L
K　V　K　Y　V
J　V　T　F　D
D　R　E　U　J

这样,用密钥字 RADIO 加密的密文,就转化为只用 R 加密的密文,即把基于多表代换密码的解密问题转化为基于单表代换密码的解密问题。

当密钥字字母的相对距离分析者知道时,攻击才有效。尽管分析者可能没有这个信息,然而,Chi 测试提供发现这个距离的迹象。在上例中,密文列被移位使得它们都用同一个代换密码解密。如果两列是用同样单表代换加密的,则两列的 χ 值将等同。现在分析者剩下的事就是重复移动某一列字母,直到这一列和第一列的最大值出现,然后就可用和第一列同样的方式解密。

例 2.8　在相距很短的时间间隔内我们收到了两段密文:

密文 1:

k o o m m a c o m o q e g l x x m q c c k u e y f c u r
y l y l i g z s x c z v b c k m y o p n p o g d g i a z
t x d d i a k n v o m x h i e m r d e z v x b m z r n l
z a y q i q x g k k k p n e v h o v v b k k t c s s e p
k g d h x y v j m r d k b c j u e f m a k n t d r x b i
e m r d p r r j b x f q n e m x d r l b c j h p z t v v
i x y e t n i i a w d r g n o m r z r r e i k i o x r u
s x c r e t v

密文 2:

z a o z y g y u k n d w p i o u o r i y r h h b z x r c

e a y v x u v r x k c m a x s t x s e p b r x c s 1 r u
k v b x t g z u g g d w h x m x c s x b i k t n s l r j
z h b x m s p u n g z r g k u d x n a u f c m r z x j r
y w y m i

因为这两段密文相隔时间很短,很有可能是用同样方式加密的。这个猜想可以通过计算两个文本的 *CI* '值来证实:

$$CI'(C_1) = 0.0421 \qquad CI'(C_2) = 0.0445$$

这两个值近似相等,所以可放心地假定两段文本的确是用同样方式加密的。

另外,因为 *CI* '的值处在随机文本和有意义的英语文本的 *CI* '值之间,因此可猜想是用多表代换加密的。为了确证之,可以采用 Kasiski 测试。首先要找到重复的字母序列及它们的距离(中间结果略):分解这些距离值为素因子的乘积,我们看到数字 7 出现最频繁,周期可能为 7。现在把密文写成 7 列的矩阵形式(见表 2.4)。

表 2.4　密文的矩阵表示

k	o	o	m	m	a	c		k	g	d	h	x	y	v		h	h	b	z	x	r	c
o	m	o	q	e	g	l		j	m	r	d	k	b	c		e	a	y	v	x	u	v
x	x	m	q	c	c	k		j	u	e	f	m	a	k		r	x	k	c	m	a	x
u	e	y	f	c	u	r		n	t	d	r	x	b	i		s	t	x	s	e	p	b
y	l	y	l	i	g	z		e	m	r	d	p	r	r		r	x	c	s	l	r	u
s	x	c	z	v	b	c		j	b	x	f	q	n	e		k	v	b	x	t	g	z
k	m	y	o	p	n	p		m	x	d	r	l	b	c		u	g	g	d	w	h	x
o	g	d	g	i	a	z		j	h	p	z	t	v	v		m	x	c	s	x	b	i
t	x	d	d	i	a	k		i	x	y	e	t	n	i		k	t	n	s	l	r	j
n	v	o	m	x	h	i		i	a	w	d	r	g	n		z	h	b	x	m	s	p
e	m	r	d	e	z	v		o	m	r	z	r	r	e		u	n	g	z	r	g	k
x	b	m	z	r	n	l		i	k	i	o	x	r	u		u	d	x	n	a	u	f
z	a	y	q	i	q	x		s	x	c	r	e	t	v		c	m	r	z	x	j	r
g	k	k	k	p	n	e		z	a	o	z	y	g	y		y	w	y	m	i		
v	h	o	v	v	b	k		u	k	n	d	w	p	i								
k	t	c	s	s	e	p		o	u	o	r	i	y	r								

然后计算每列的重合指数,有:

$CI'(\text{列}\ 1) = 0.0522$　$CI'(\text{列}\ 2) = 0.0801$　$CI'(\text{列}\ 3) = 0.0734$
$CI'(\text{列}\ 4) = 0.0744$　$CI'(\text{列}\ 5) = 0.0705$　$CI'(\text{列}\ 6) = 0.0717$
$CI'(\text{列}\ 7) = 0.0606$

但观察每一列的重合指数,似乎每一列都是用一个单表代换密码加密的。现在我们试图转化多表代换的密文为某个单个的单表代换加密的密文。首先,我们重复移动各列字母,移动的距离分别为 1～25,并分别计算相对于第一列的 χ 值,把最大值用下划线标示出来。

列 1 和 2: 0.0388 0.0487 0.0317 0.0326 0.0274 0.0340 0.0421 0.0402
0.0321 0.0350 0.0425 0.0411 <u>0.0662</u> 0.0350 0.0317 0.0359
0.0491 0.0331 0.0236 0.0378 0.0345 0.0288 0.0567 0.0525
0.0302

列 1 和 3: 0.0378 0.0274 0.0331 0.0331 0.0250 0.0581 0.0491 0.0458
0.0284 0.0383 0.0529 0.0491 0.0307 0.0250 0.0312 0.0444
0.0392 0.0359 0.0392 0.0354 0.0421 <u>0.0657</u> 0.0416 0.0269
0.0232

列 1 和 4: 0.0317 0.0369 0.0364 0.0312 0.0454 0.0383 <u>0.0558</u> 0.0302
0.0388 0.0345 0.0520 0.0250 0.0359 0.0336 0.0477 0.0260
0.0435 0.0520 0.0406 0.0369 0.0468 0.0468 0.0326 0.0307
0.0326

列 1 和 5: 0.0430 <u>0.0586</u> 0.0279 0.0274 0.0331 0.0473 0.0298 0.0359
0.0336 0.0354 0.0302 0.0491 0.0548 0.0265 0.0359 0.0406
0.0506 0.0312 0.0345 0.0336 0.0440 0.0354 0.0506 0.0411
0.0321

列 1 和 6: 0.0382 0.0271 0.0502 0.0449 0.0295 0.0319 0.0440 0.0522
0.0372 0.0372 0.0343 0.0396 0.0391 0.0391 0.0280 0.0324
0.0454 0.0430 <u>0.0614</u> 0.0362 0.0324 0.0343 0.0498 0.0338
0.0275

列 1 和 7: 0.0285 0.0444 0.0362 0.0382 0.0357 0.0353 0.0343 0.0415
0.0377 0.0483 0.0333 0.0396 0.0425 0.0300 <u>0.0565</u> 0.0348
0.0329 0.0348 0.0454 0.0304 0.0377 0.0324 0.0449 0.0295
0.0444

基于以上结果推知,各列相对于第一列的距离分别是:

列 2:13　列 3:22　列 4:7　列 5:2　列 6:19　列 7:15

这些值用来转化密文 C_1, C_2 到下列文本:

k b k t o t r o z k x g z a x k i x e
v z u r u m e n g y y u s k z o s k y
g x u r k z u v r g e o t z n k t o t
k z k k t z n i k t z a x e z n k g s
k x o i g t g a z n u x k j m g x g r
r g t v u k c x u z k g y z u x e k t
z o z r k j z n k m u r j h a m o t z
n g z y z u x e z n k r k g j o t m s
g t m k z y n u r j u l g v o k i k u
l v g x i n s k t z c o z n g t k t i
x e v z k j s k y y g m k z n k g a z
n u x j k y i x o h k y k r g h u x g
z k r e n u c z n k r k g j o t m s g
t z g i q r k y z n k j k i x e v z o
u t c k y a m m k y z z u x k g j z n
k y z u x e o l e u a c g t z z u q t
u c n u c z n g z c g y j u t k

现在就可以用一单表代换密码的解密方法解密这段文本,解密的任务留给读者。

2.2.3 对 Hill 密码的已知明文分析

Hill 密码在唯密文攻击下是很难破的,但很容易被已知明文攻击所攻破。首先假定我们确定了 m 的值,且得到至少 m 对不同的 m 元组:

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

已知 $y_j = e_k(x_j)$。$1 \leq j \leq m$ 如果定义两个 $m \times m$ 矩阵 $\boldsymbol{X} = (x_{ij})$, $\boldsymbol{Y} = (y_{ij})$,则有矩阵方程 $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{k}$, $\boldsymbol{k}$ 是未知密钥。假定 $\boldsymbol{X}$ 是可逆的,则可计算 $\boldsymbol{k} = \boldsymbol{X}^{-1}\boldsymbol{Y}$,因此可攻破系统(如果 $\boldsymbol{X}$ 不可逆,尝试其他 m 个明密文对)。

可以看一个简单的例子。

例 2.9 明文 friday 是用 Hill 密码加密的, $m = 2$,得到密文 POCFKU,则有

$$e_k(5,17) = (15,16) \qquad e_k(8,3) = (2,5), \qquad e_k(0,24) = (10,20)。$$

从最初两个明文对,我们得到矩阵方程

$$\begin{matrix} 15 & 16 \\ 2 & 5 \end{matrix} = \begin{matrix} 5 & 17 \\ 8 & 3 \end{matrix} \boldsymbol{k},$$

容易计算 $\begin{matrix} 5 & 17 \\ 8 & 3 \end{matrix}^{-1} = \begin{matrix} 9 & 1 \\ 2 & 15 \end{matrix}$,所以

$$\boldsymbol{k}=\begin{matrix}9 & 1\\ 2 & 15\end{matrix}=\begin{matrix}15 & 16\\ 2 & 5\end{matrix}=\begin{matrix}7 & 19\\ 8 & 3\end{matrix}。$$

然后可用第三个明密文对 $\boldsymbol{k}$ 进行验证。

注 记

本章简要介绍了几种有代表性的古典密码体制及对这些体制的一些破译方法,用来说明设计和分析密码的基本方法。虽然这些密码大都比较简单而且容易破译,但研究这些密码的设计原理和分析方法对于理解、设计和分析现代密码是十分有益的。事实上,代换和置换方法正是增强密码安全性的两个基本手段,在进一步学习了分组密码之后,读者一定会有所体会。

鉴于 Kasiski 测试法、重合指数法和 Chi 测试法的具体操作一般教科书叙述较少,所以在介绍 Vigenère 密码分析方法时对此作了详细的叙述。

尽管古典密码有几千年的历史,但由于发展缓慢且局限于军事和外交领域,遗留下来的专题文献并不多见。本章主要参考国内外有关的密码学专著或教材,如文献[6],[15],[26],[46]。

习 题 二

1. 已知仿射加密变换为 $c=5m+7(\mathrm{mod}\ 26)$,试对明文 help me 加密。

2. 已知仿射加密变换为 $c=11m+2(\mathrm{mod}\ 26)$,试对密文 VMWZ 解密。

3. 已知下列密文是通过单表代替密码的结果,试求其明文:

YIF QFMZRW QFYV ECFMD ZPCVMRZW NMD ZVEJB TXCDD UMJN DIFEFMDZ CD MQ ZKCEYFCJMYR NCW JCSZR EXCHZ UNMXZ NZ UCDRJ XYYSMRJ M EYIFZW DYVZ VYFZ UMRZ CRW NZ DZJJXZW GCHS MR NMD HNCMF QCHZ JMXJZW IE JYUCFWD JNZ DIR。

4. 设已知 Vigenère 密码的密钥为 matrix,试对明文 beijing university of posts and telecommunications 加密。

5. 已知下列密文是通过 Vigenère 密码加密得来的,试求其明文:

OOBQBPQAIUNEUSRTEKASRUMNARRMNRROPIODEEADERUNRQLJUGCZC CUNRTEUARJPTMPAWUTNDOBGCCEMSOHKARCMNBYUATMMDERDUQFWMD TFKILROPGPQARUNDXUCZCCGPMZTFQPMXIAUEQAFEAVCDNKQNREYCEIRT AQZETQRFMDYOHPANGOLCD。

6. 假设 Hill 密码加密使用密钥 $\boldsymbol{K}=\begin{matrix}4 & 9\\ 3 & 7\end{matrix}$, 试对明文 best 加密。

7. 假设 Hill 密码加密使用密钥 $\boldsymbol{K}=\begin{matrix}4 & 9\\ 3 & 7\end{matrix}$, 试对密文 UMFL 解密。

8. 假定我们被告知明文 conversation 产生密文 HIARRTNUYTUS, 所用的密码体制为 Hill 密码(但 m 不限定), 确定加密矩阵。

9. 在“一次一密”制中, 假定下列比特串是随机选择的密钥:

$$k=(1100101000110011110001010111000101111111010101001),$$

并且, 假定下列比特串是由 k 加密得到:

$$c=(1011100101111010111100010100000111110000010101001 0),$$

试找出明文串 m(设 c 是由 m 与 k 逐比特模 2 相加得到的)。

第 3 章　密码学的信息论基础

Shannon 于 1948 年确立了现代信息论[3]。他在 1949 年发表了《保密系统的通信理论》一文[2],用信息论的观点对信息保密问题作了全面的阐述。他以概率统计的观点对消息源、密钥源、接收和截获的消息进行数学描述和分析,用不确定性和惟一解距离度量了密码体制的安全性,阐明了密码系统、完善保密性、纯密码、理论安全性和实际安全性等重要概念,从而大大深化了人们对于保密学的理解。这使信息论成为研究密码学和密码分析学的一个重要理论基础,宣告了科学的密码学信息理论时代的到来。

3.1　保密系统的数学模型

首先,我们简单地比较一下通信系统和保密系统(如图 3.1 和 3.2 所示)的一般模型。

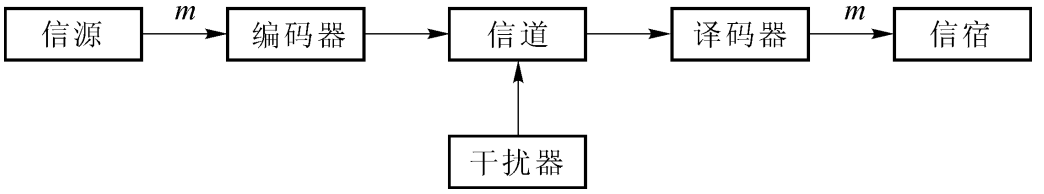


图 3.1　通信系统模型

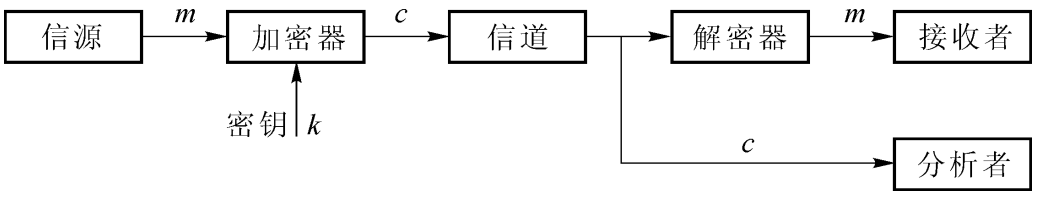


图 3.2　保密系统模型

可以看出,这两个系统最关键的不同在于:通信系统信道上传的是明文,而保密系统信道上则要求是密文。这也是由两者的侧重点不同所决定的:通信系统设计的目的是在信道有干扰的情况下,使接收的信息无错或差错尽可能地小;而保密系统设计的目的是使窃听者即使在完全准确地截取了密文的情况下也无法恢复出原始消息。或者,简单地说,通信系统的目的是去噪,保密系统的目的是加噪。所不同的是,这种噪声不是信道中自然

产生的,而是发送者有意加的,目的是使窃听者不能从密文 c 恢复出原来的消息。

一个更准确的 Shannon 提出的保密系统模型如图 3 3 所示。

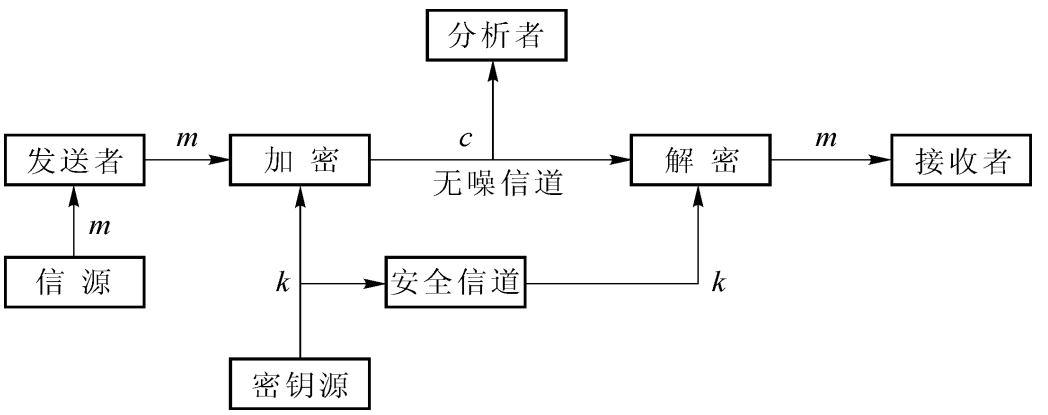


图 3 3　Shannon 的保密系统模型

与通信系统类似,也可用数学语言描述保密系统。

信源是产生消息的源,在离散情况下可以产生字母或符号。可以用简单的概率空间来描述离散无记忆信源。设信源字母表为 $M = \{a_i, i = 0,1,2,\cdots, N-1\}$,字母 a_i 出现的概率为 $p(a_i) \geq 0$,且 $\sum_{i=0}^{N-1} p(a_i) = 1$。信源产生的任一长为 L 个符号的消息序列为

$$m = (m_1, m_2, \cdots, m_L),\ m_l \in M, 1 \leq l \leq L$$

(可看做一个 L 维的随机向量)。

若我们研究的是所有长为 L 的信源输出,则称

$$P = M^L = \{m = (m_1, m_2, \cdots, m_L) \mid m_l \in M, 1 \leq l \leq L\}$$

为消息空间或明文空间(即样本空间),它含有 N^L 个元素。为简单起见,一般取信源为离散无记忆的,此时明文空间的概率分布为 $p_P(m) = p(m_1, m_2, \cdots, m_L) = \prod_{l=1}^{L} p(m_l)$。

在第 2 章我们已看到了如何利用明文的统计特征来破译密码。

类似地,设密钥源字母表为 $B = \{b_t \mid t = 0,1,2,\cdots, s-1\}$,字母 b_t 出现的概率为 $p(b_t) \geq 0$,且 $\sum_{t=0}^{s-1} p(b_t) = 1$,我们在设计中一般尽量使密钥源为无记忆均匀分布源,所以各密钥符号为独立等概。把长为 r 的密钥序列 $k = (k_1, k_2, \cdots, k_r),\ k_i \in B$ 的全体称为密钥空间 K。

密文自然也可看做统计的量。由于三者的制约关系:在密钥的控制下明文被变换为密文 $c = (c_1, c_2, \cdots, c_{L'}) = e_k(m_1, m_2, \cdots, m_L)$,称 c 的全体为密文空间,以 $C = Y^{L'}$ 表示,其中 Y 表示密文字母表,L' 表示密文长度。通常,密文字母表与明文字母表相同,因而一般有 $L = L'$,且对每个 $k \in K$, e_k 为一一映射。密文 c 通过无噪信道传送给接收者,而密钥 k 是通过安全信道传送给接收者的,因此,接收者可用同样的密钥 k 和 e_k 的逆映射 d_k 将密文 c 解密为 $m = d_k(c)$,从而得到发送者所发送的明文消息;而分析者虽然可从信道中截得密文 c,但他不知道加密所用的密钥 k,故他要从密文 c 恢复出明文 m 或分析出加

密所用的密钥 k 是较困难的。

密文的统计特征由明文和密钥的统计特征决定。假定明文 $m \in P$ 发生的概率为 $p_P(m)$,密钥 k 被选择的概率为 $p_K(k)$,因为密钥是在发送者发送消息之前选定的,因而可假定消息空间和密钥空间是独立的。对一个密钥 $k \in K$,令 $C_k = \{e_k(m) \mid m \in P\}$ 表示所有可能密文的集合,对每一个 $c \in C$,我们有密文的概率分布为

$$p_C(c) = \sum_{k \mid c \in C_k} p_K(k)\, p_P(d_k(c))。\tag{3.1}$$

同时也可观察到,对每一个 $c \in C$ 和 $m \in P$,可计算条件概率 $p_C(c \mid m) = \sum_{k \mid m = d_k(c)} p_K(k)$。

现在计算条件概率 $p_P(m \mid c)$(即在给定密文 c 发生的条件下,某个明文 m 发生的概率),由 Bayes 公式可得

$$p_P(m \mid c) = \frac{p_P(m) \sum_{k \mid m = d_k(c)} p_K(k)}{\sum_{k \mid c \in C_k} p_K(k)\, p_P(d_k(c))}。\tag{3.2}$$

注意,任何知道明文空间和密钥空间的概率分布的人,都可计算密文空间的概率分布以及明文空间关于密文空间的概率分布。可见,密文空间的统计特性由明文空间和密钥空间的统计特性完全决定。

3.2　信息量和熵

熵(Entropy)是来自信息论的概念,由 Shannon 于 1948 年引进。熵可被视作信息或不确定性的一个数学度量,是定义在一个概率分布上的函数。

定义 3.1　设随机变量 $X = \{x_i \mid i = 1, 2, \cdots, n\}$,$x_i$ 出现的概率为 $p(x_i) \geq 0$,且 $\sum_{i=1}^{n} p(x_i) = 1$,则 X 的不确定性或熵定义为

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)。\tag{3.3}$$

熵的表达式实际为相应概率分布负对数的数学期望。因此 $H(X)$ 反映了集 X 中事件出现的平均不确定性,或为确定集 X 中出现一个事件平均所需的信息量(观测之前),或集 X 中每出现一个事件平均给出的信息量(观测之后)。如果从编码的角度来考虑,熵也可以理解成用最优的二进制编码形式表示 X 所需的比特数。我们规定 $0 \cdot \log_2 0 = 0$。采用以 2 为底的对数时,相应的信息单位称作比特(bit)。

注意,如果集 X 为均匀分布时,即 $p_i = 1/n, 1 \leq i \leq n$,则 $H(X) = \log_2 n$,且容易看

到，$H(X) \geq 0$，当 X 为确定性的事件时，即 X 概率分布为 $p(X=a)=1$，则 $H(X)=0$。

例 3.1 设有一个密码系统明文空间 $P=\{a,b\}$ 的概率分布为

$$p_P(a)=1/4, \quad p_P(b)=3/4,$$

密钥空间 $K=\{k_1,k_2,k_3\}$ 的概率分布为

$$p_K(k_1)=1/2, \quad p_K(k_2)=p_K(k_3)=1/4,$$

密文空间 $C=\{1,2,3,4\}$，且假定加密函数为

$$e_{k_1}(a)=1,\ e_{k_1}(b)=2;\ e_{k_2}(a)=2,\ e_{k_2}(b)=3;\ e_{k_3}(a)=3,\ e_{k_3}(b)=4。$$

可用下面的加密矩阵表示：

	a	b
k_1	1	2
k_2	2	3
k_3	3	4

则按公式(3.1)和(3.2)很容易计算出密文空间的概率分布及密文关于明文的条件分布：

(1) 密文空间的概率分布表如下：

1	2	3	4
1/8	7/16	1/4	3/16

(2) 明文关于密文的条件分布 $p(m|c)$ 表如下：

c \ m	a	b
1	1	0
2	1/7	6/7
3	1/4	3/4
4	0	1

明文空间的熵为：

$$H(P)=-\frac{1}{4}\log_2\frac{1}{4}-\frac{3}{4}\log_2\frac{3}{4}=2-\frac{3}{4}(\log_2 3)\approx 0.81。$$

类似地可计算 $H(K)=1.5$，且 $H(C)\approx 1.85$。

当要考虑到随机变量之间的关系时，我们自然地想到联合熵和条件熵。

定义 3.2 设 $X=\{x_i \mid i=1,2,\cdots,n\}$，$x_i$ 出现的概率为 $p(x_i)\geq 0$，且

$\sum_{i=1}^{n} p(x_i) = 1$。$Y = \{y_j \mid j = 1,2,\cdots,m\}$，$y_j$ 出现的概率为 $p(y_j) \geq 0$，且 $\sum_{j=1}^{m} p(y_j) = 1$，则联合事件集 $XY = \{x_i y_j \mid i = 1,2,\cdots,n; j = 1,2,\cdots,m\}$，令 $x_i y_j$ 的概率为 $p(x_i y_j) \geq 0$，此时 $\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i y_j) = 1$。集 X 和 Y 的联合熵定义为

$$H(XY) = H(X, Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i y_j)\log_2 p(x_i y_j),$$

集 X 相对于事件 $y_j \in Y$ 的条件熵定义为

$$H(X \mid y_j) = -\sum_{i=1}^{n} p(x_i \mid y_j)\log_2 p(x_i \mid y_j),$$

集 X 相对于集 Y 的条件熵定义为

$$H(X \mid Y) = \sum_{j=1}^{m} p(y_j) H(X \mid y_j) = -\sum_{j=1}^{m}\sum_{i=1}^{n} p(y_j)\, p(x_i \mid y_j)\log_2 p(x_i \mid y_j)。$$

条件熵 $H(X \mid Y)$ 理解为观察到集 Y 后集 X 还保留的不确定性，或 X 未被 Y 所泄露的信息量的平均值。

若将 X 视为一个系统的输入空间，Y 视为系统的输出空间。在通信中，通常将条件熵 $H(X \mid Y)$ 称作含糊度(Equivocation)，将条件熵 $H(Y \mid X)$ 称为散布度(Divergence)，X 和 Y 之间的平均互信息 $I(X, Y) = H(X) - H(X \mid Y)$ 表示 X 熵减少量。

下面将给出一些关于熵的基本特性。首先，我们介绍一个重要的不等式——Jensen 不等式。

引理 3.1(Jensen 不等式)　假定 f 是区间 I 上的一个连续的严格凸函数，$\sum_{i=1}^{n} a_i = 1, a_i > 0, 1 \leq i \leq n$，那么

$$\sum_{i=1}^{n} a_i f(x_i) \leq f\left(\sum_{i=1}^{n} a_i x_i\right), x_i \in I, 1 \leq i \leq n,$$

当且仅当 $x_1 = x_2 = \cdots = x_n$ 时等号成立。

由高等数学知识易知，函数 $f(x) = \log_2 x$ 在区间 $I = (0, \infty)$ 上是一个连续的严格凸函数。

定理 3.1　假定 X 是有概率分布 $p_1, p_2, \cdots, p_n$ 的随机变量，其中 $p_i > 0, 1 \leq i \leq n$，则 $H(X) \leq \log_2 n$，当 $p_i = \frac{1}{n}, 1 \leq i \leq n$(即样本点是等概率分布)时取等号，即均匀分布下集 X 的不确定性最大。

证明　由 Jensen 不等式得

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i} \leq \log_2 \sum_{i=1}^{n} p_i \times \frac{1}{p_i} = \log_2 n。$$

定理 3.2　$H(X, Y) \leq H(X) + H(Y)$，当且仅当 X 和 Y 独立时等号成立。

证明 假定 X 的取值为 $x_i, 1 \le i \le m$, Y 的取值为 $y_j, 1 \le j \le n$, 记 $p(x_i) = p_i$, $1 \le i \le m$, 且 $p(y_j) = q_j, 1 \le j \le n$, $p(x_i y_j) = r_{ij}, 1 \le i \le m, 1 \le j \le n$。

$$p_i = \sum_{j=1}^{n} r_{ij} \quad 1 \le i \le m, \qquad q_j = \sum_{i=1}^{m} r_{ij} \quad 1 \le j \le n。$$

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

另一方面,
$$H(X, Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} \log_2 r_{ij},$$

$$H(X, Y) - H(X) - H(Y) = \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} \log_2 \frac{p_i q_j}{r_{ij}} \le \log_2 \sum_{i=1}^{m}\sum_{j=1}^{n} p_i q_j = \log_2 1 = 0$$

下面的两个结论是直接的,留下证明作为练习。

定理 3.3 $H(X, Y) = H(Y) + H(X|Y) = H(X) + H(Y|X)$。

推论 3.1 $H(X|Y) \le H(X)$, 当且仅当 X 和 Y 独立时等号成立。

有关各类熵之间的关系可以用如下的维拉图清晰表示(如图 3.4 所示)。

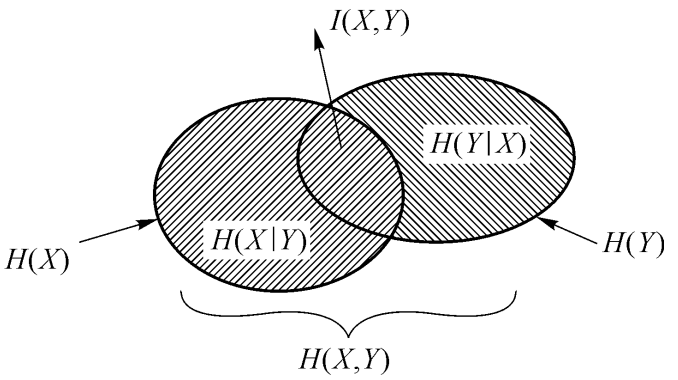


图 3.4 各类熵之间的关系

3.3 完善保密性

在这部分,我们运用熵的结果到一个密码系统,得出密码系统各部分的熵之间的一些基本关系。对于一个给定的密码系统,设明文熵为 $H(P)$, 密文熵为 $H(C)$, 密钥熵为 $H(K)$, 已知密文条件下明文含糊度为 $H(P|C)$, 条件熵 $H(K|C)$ 称为已知密文条件下密钥含糊度,它意味着密钥未被密文泄露了多少信息。

定理 3.4 设 (P, C, K, e, d) 是一个密码系统,则 $H(K|C) = H(K) + H(P) - H(C)$。

证明 首先,观察到 $H(K, P, C) = H(C|K, P) + H(K, P) = H(P|K, C) + H(K, C)$。由于密钥和明文惟一地确定密文。密钥和密文惟一确定明文,密钥和明文统计独立,所以 $H(C|K, P) = H(P|K, C) = 0$, $H(K, P) = H(K) + H(P)$。从而 $H(K) + H(P) =$

$H(K, C)$,又 $H(K, C) = H(C) + H(K|C)$,故 $H(K|C) = H(K) + H(P) - H(C)$。

明文密文之间的平均互信息为 $I(P, C) = H(P) - H(P|C)$。它反映了密文给出的关于明文的信息,所以实现一个密码系统的目标将是最小化 $I(P, C)$。如果密文不提供任何关于明文的信息(或敌手通过观察密文不能获得任何关于明文的信息),则 $H(P|C) = H(P)$,或者 $I(P, C) = 0$。这正是下面将要定义的完善保密系统。

定义 3.3　一个保密系统(P, C, K, e, d)称为是完善的或无条件的保密系统,如果 $H(P|C) = H(P)$或 $I(P, C) = 0$。

定理 3.5　$I(P, C) \geq H(P) - H(K)$。

证明　因为 $H(P|C, K) = 0$, $H(P|C) \leq H(P|C) + H(K|P, C) = H(P, K|C) = H(K|C) + H(P|C, K) = H(K|C) \leq H(K)$,

这样, $H(P) - H(P|C) \geq H(P) - H(K)$,即

$$I(P, C) \geq H(P) - H(K)。$$

由定理 3.5 知,完善保密系统存在的必要条件是 $0 = I(P, C) \geq H(P) - H(K)$,即 $H(P) \leq H(K)$。换句话说,系统密钥量的对数(假定密钥空间为均匀分布)必须不小于明文集的熵。

一个众所周知的无条件安全的实现是一次一密制,1926 年由 G.S.Vernam 引进。我们基于 26 个英文字母来描述这个系统。设消息 $M = (m_1, m_2, \cdots, m_L)$由 L 个字母组成。为了加密 M,首先随机产生一个 L 字母序列(来自同样字母表)。每个字母有 $1/26$ 概率被挑选,并且每次选择都是独立的。这个随机产生的序列 $k = (k_1, k_2, \cdots, k_L)$被用来作为密钥加密 M。密文 $C = (c_1, c_2, \cdots, c_L)$通过下面的计算得到:

$$c_i = m_i + k_i (\mathrm{mod}\ 26) \qquad i = 1, \cdots, L,$$

其中,每个字母按其在字母表中的位置分别用 0～25 的数字表示。

因为有 26^L 等可能的密钥,所以 $H(K) = L\log_2 26$。因为 k 有独立选择的字母组成,对每 m_i 和 c_i 有惟一 k_i 存在,则对每个 M 和 C 组合,

$$p(C|M) = \frac{1}{26^L}。$$

有 26^L 个等可能的密文,即 $p(C) = 1/26^L$。这样, M 和 C 是独立的,且 $I(M, C) = 0$,显然,一次一密制提供了极好的安全性。

无条件安全的密码系统安全性依赖于每个密钥仅仅用在一次加密中,在每个消息被传送之前,一个新的密钥必须被产生。另外,每个消息必须与同样长度的密钥相伴,这是极其不利的,因为密钥应当在消息之前被安全传送。这些都给密钥管理带来了严重的问题。再加上一次一密系统对已知明文攻击非常脆弱。因此无条件安全的保密系统是很不实用的,也具有很大的局限性,但在军事和外交上很早就使用了这种体制。

密码学的历史发展一直在试图设计一个用一个密钥就可以加密一个相当长的明文串(即一个密钥可用来加密许多消息)的密码系统,且仍能保持至少计算上安全。这样的系

统可参见第 8 章介绍的 DES 体制。

3.4 理论安全性和实际安全性

在前面的章节中,我们没有考虑被分析者所截取的密文长度与系统安全性的关系。

直觉告诉我们,分析者成功地解密一段文本的机会随文本长度的增加而增加。表示一个长为 S 的密文序列为 $C^S = (C_1, \cdots, C_S)$,密钥含糊度 $H(K|C^S) = H(K|C_1, \cdots, C_S)$ 将随 S 的增加而减少,即所用密钥的不确定性将随着 S 的增加而减少(如图 3.5 所示)。在图 3.5 中,我们可见密钥含糊度甚至可以达到零,此时意味着密钥可由密文恢复。

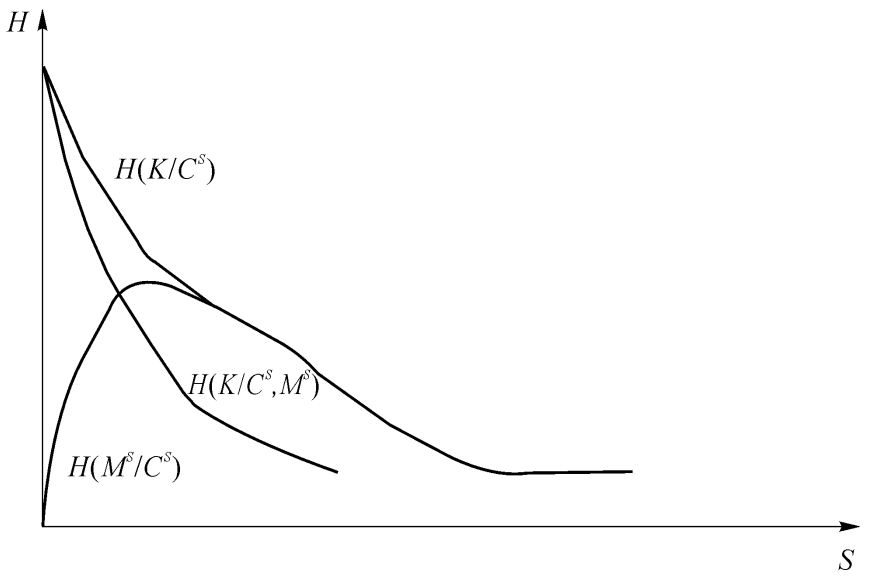


图 3.5 密钥、消息和密钥显现含糊度作为 S 的函数

类似地可作出消息含糊度曲线 $H(M^S|C^S)$。尽管有一个例外:当 S 很小时,可能的消息数目也少,因此消息含糊度仍然很低。只要 S 增加,可能的消息数目就会戏剧性地增加,相应地,消息含糊度也将增加。一旦 S 充分大,密文将包含足够的信息,最可能的消息数目将不再增加。不久以后就会达到这一点,消息含糊度曲线与密钥含糊度曲线重合,因为此时,密文包含足够的信息来恢复密钥,并且该密钥也可以从发现的明文中以同样的不确定性恢复得到。图 3.5 对此做了演示。

最后,图 3.5 也显示了密钥显现的含糊度(key appearance equivocation) $H(K|C^S, M^S)$。显然,密钥显现的含糊度比密钥含糊度更快地减少到零,因为在这种情况下,我们假定分析者既拥有明文又拥有密文,因此将会更快地找到密钥。密钥显现含糊度是用来测量密码抗击已知明文攻击的能力,而密钥含糊度和消息含糊度是测量密钥和明文抗击唯密文攻击的能力。

从上面的分析可知,收到的密文越长,分析者找到密钥或明文的可能性越大。为了确信分析者可找到惟一的密钥,我们想知道理论上 S 至少将要多长。

在第 2 章中我们看到,分析密码时,常常利用自然语言的统计特征,从信息论的角度看,实际上利用了所谓语言的冗余度或称多余度(redundancy)。

定义 3.4　假如 L 是一种自然语言,语言 L 的熵定义为

$$H_L = \lim_{n\to\infty} \frac{H(A^n)}{n},$$

语言 L 的多余度定义为

$$R_L = 1 - \frac{H_L}{\log_2 \varepsilon}。$$

其中 A 表示语言 L 的字母集,ε 表示 A 中字母的个数,A^n 表示所有明文 n—字母报构成的集。

H_L 表示了语言 L 的每个字母的平均信息比特数,R_L 度量了“多余字母”的比例或理解为某种语言中平均每个字母的多余度。据估计,英语语言 $1.0 \leq H_L \leq 1.5$。若用 1.25 作为 H_L 的估计,得到大约 0.75 的冗余。这意味着英语语言是 75% 冗余(这并不是说对任意的英文文本从每四个中移去三个后仍能解读它,所意味的是可以找到一个对 n—字母报的 Huffman 编码,可将文本压缩到其原文长度的 1/4)。

密钥含糊度的最大值是 $H(K)$,当对每个密文而言,每个密钥的可能性是相等的,这个值就会出现。在这种情况下,关于所用密钥的最大不确定性达到。

定理 3.6　密钥含糊度有下列下界

$$H(K|C^S) \geq H(K) - SR_L \log_2 \varepsilon,$$

其中,S 表示接收到的密文序列长度,R_L 表示明文语言的冗余度,ε 表示明文和密文空间中符号或字母的数目。

证明　由于明密文一对一的关系,下列表达式总是成立的:

$$H(K, C^S) = H(K, M^S)。$$

设密钥独立于明文,我们可推导出密钥含糊度

$$H(K|C^S) = H(K, C^S) - H(C^S) = H(K, M^S) - H(C^S) = H(K) + H(M^S) - H(C^S)。$$

假定明文空间和密文空间的不同的符号数目均为 ε,则长为 S 密文的可能的总数目为 ε^S。由定理 3.1,$H(C^S) \leq S\log_2 \varepsilon$,$H(K|C^S) \geq H(K) + H(M^S) - S\log_2 \varepsilon$。

因 $R_L = 1 - \frac{H_L}{\log_2 \varepsilon}$,$H_L = \lim_{n\to\infty} \frac{H(A^n)}{n}$,当 S 充分大时,$\frac{H(M^S)}{S} \approx H_L = -R_L \log_2 \varepsilon + \log_2 \varepsilon$,则有 $SR_L \log_2 \varepsilon = S\log_2 \varepsilon - H(M^S)$为明文的多余度(它是度量一个实际产生的信息源和均匀分布的信息源之间差别大小)。

因此,$H(K|C^S) \geq H(K) - SR_L \log_2 \varepsilon$。

我们可以这样来解释定理 3.6:随着明文多余度的增加,平均密钥含糊度将减少,因此所用密钥的不确定性将减少。或者换句话说,冗余使得找到密钥更容易。相反,减少冗余的方法可以增加密码系统的安全性。

定理 3 .6 的另一个解释是:只要 $H(K) > SR_L \log_2 \varepsilon$,密钥含糊度就不会为零,密钥也就不能惟一地恢复。

定理 3 .7 阐明:对 S 来讲,在什么情况下密钥含糊度将可能变为零。

定理 3 .7 当明文由一个离散独立信源产生时,如果

$$S \geq \frac{H(K)}{\log_2 \varepsilon - H(M)}$$ (其中,ε 是字母表的大小),

密钥的含糊度能变为零。

证明 一个独立信源意味,$H(M^S) = S \cdot H(M)$。明文总的信息量等于文本中每个符号信息量的 S 倍。再联合定理 3 .6 得

$$H(K \mid C^S) \geq H(K) + S \quad H(M) - \log_2 \varepsilon ,$$

这样,当 $S \geq H(K) / \quad \log_2 \varepsilon - H(M)$ 时,平均密钥含糊度可变为零。因此当信源的平均信息量较少时,恢复密钥只需一个较小数目的密文量。实际上,为了避免这种情况,可通过增加明文消息的信息量,比如采用信源编码。临界值 $S = H(K) / \quad \log_2 \varepsilon - H(M)$ 称为惟一解距离(Unicity Distance),简记为 UD。这是理论上找到密钥所需的最小密文长度。

例如,考虑一个通过替换密码加密的英文文本,假定英文语言的信息量 $H(M)$ 是 2 bit。因为有 26 个英文字母,$\log_2 \varepsilon - H(M) = \log_2 26 - 2 = 4 .7 - 2 = 2 .7$ bit。一个替换密码有 26 ! 个可能的密钥。(假定每个密钥的发生是等可能的)每个密钥的信息量等于 $H(K) = \log 26 ! = 88 .38$ bit。

求出惟一解距离为:$\mathrm{UD} = H(K) / \quad \log_2 \varepsilon - H(M) = 88 .38 / 2 .7 \approx 32$。

因此,平均需要 32 个字母的密文来找到正确的密钥。

在 Caesar 代换的情况下,$H(K)$ 将减少到 $H(K) = \log_2 26 = 4 .7$ bit。现在惟一解距离非常小:

$$\mathrm{UD} = \frac{4 .7}{2 .7} \approx 2。$$

最后,我们声明:这仅仅是理论上的可能性,即当截获的密文长度大于惟一解距离时,原则上就可破译。但有两点需要指出:第一,这一可能是假定分析者能利用明文语言的全部统计知识的条件下得到的,实际上由于自然语言的复杂性,没有任何一种分析方法能够做到这一点,所以,一般破译所需的密文量都远大于理论值;第二,这里没有涉及为了得到惟一解所需要作出的努力,或需完成多少计算量。从实际破译来看,有时虽然截获的密文长度远大于惟一解距离,但由于所需的工作量太大而难以实现破译。没能将计算量考虑进去,是以这种信息理论方法研究密码学的一个重要缺陷。Shannon 在 1949 年的论文中已意识到了这点,并提出了实际保密性概念来弥补它。

理论保密性是假定密码分析者有无限的时间、设备和资金的条件下,研究唯密文攻击时密码系统的安全性。一个密码系统,如果对手有无限的资源可利用,而在截获任意多密

报下仍不能被破译,则它在理论上是保密的,比如一次一密体制。另一方面,一个在理论上安全的系统在实际上也可能是很脆弱的。因为“理论上不可破”这句话忽略了许多很重要的因素。例如,一次一密体制中假定密钥的传送不经过密码系统本身,而且要求的密钥量至少和明文一样多。但要将大量密钥送给收端有很多实际困难,致使密钥管理系统很脆弱而易受攻击。因此,实际密码系统不能单纯地追求理论保密性。

实际密码分析者所具有的资金、设备和时间总是有限的。在这种条件下来研究密码体制的安全性,就是研究系统的实际保密性。一个密码系统的破译所需要的努力,如果超过了对手的能力,则该系统为实际上安全的。

具体地说,实际安全性又称为计算上的安全性,这个方法关心的是破译一个具体的密码系统所需的计算量。我们可以这样设想:如果用一个最好的算法破译该系统需要 N 次运算,而 N 是某个确定的、很大的数,则称该密码系统是计算上安全的。问题是还没有一个已知的算法在这个定义下是可证明安全的。在实际中,人们说一个密码系统是“计算上安全的”,意指利用已有的最好的方法破译该系统所需要的努力超过了敌手的破译能力(诸如时间、空间和资金等资源)或破译该系统的难度等价于解数学上的某个已知难题(这跟计算复杂性有关,参见第 4 章)。当然,这只是提供了系统是计算上安全的一些证据,并没有真正证明系统是计算上安全的。

如何估计一个系统的实际保密性?最主要的需考虑两个因素:一是密码分析者的计算能力;二是他所采用的破译算法的有效性。

随着技术发展提供更大更快的计算机,这点变得更加重要。不考虑密钥空间的大小,尝试一个大数目的密钥变得越来越有吸引力。

在估计系统的保密性时,首先要估计破译它所需的基本运算次数和存储量,看是否能满足我们的安全需要。

破译算法的有效性是十分重要的,密码分析者总是在搜寻新的方法来减少破译所需的运算量。例如,假定每微秒可以试验一个单表代换的密钥,则要穷尽所有 26! 个单表代换的密钥需要 1.28×10^{13} 年。但实际分析者绝不会采用此等笨拙的方法。如第 2 章所述,采用一些统计分析法,即使对相当短的密文,也可破译。

密码设计者的任务是尽力设计一个理想的或完善的保密系统,即使做不到这点,也要保证使分析者必须付出相当的代价(时间、费用),甚至在收到密文量超过惟一解距离时,也能满足实际保密的要求。

最后,我们简要介绍一下仙农(Shannon)所提到的关于设计密码的一些基本观点。

首先介绍 Shannon 另一个创新的想法:通过合并简单密码系统而形成它们的“积”。这个想法在设计分组密码时特别重要的。设有两个子密码系统 T_1 和 T_2,则先用 T_1 对明文加密,然后再用 T_2 对所得结果进行加密。其中 T_1 的“密文空间”需作 T_2 的“明文空间”。乘积密码可表示为 $T = T_2 T_1$。类似地,可定义有限个子密码系统的乘积。特殊

情况是一个密码系统自身乘积，T^n 表示密码系统 T 的 n 次迭代，如果是幂等的，即 $T^2 = T$，这时迭代并不能提高系统的安全性。如果是非幂等的，则迭代可潜在地提高密码系统的安全性，这个想法被用在分组密码 DES 算法里，DES 有 16 轮迭代过程。幸运的是有一个简单的构造非幂等密码系统常用的技巧：代换密码和置换密码的乘积。

其次，我们介绍 Shannon 的挫败统计分析的观点。用统计分析可以破译多种密码。为了挫败统计分析，Shannon 建议采用两种方法，一种是在加密之前将语言的一些多余度除去。诸如，在计算机系统中，在加密之前，采用 Huffman 编码除去多余度来压缩一个文件。另一种是采用所谓的“扩散(Diffusion)”和“混淆(Confusion)”这两种加密技术扩散或混淆多余度。所谓扩散，就是将每一位明文数字的影响尽可能迅速地散布到较多个输出的密文数字中，以便隐藏明文数字的统计特性。所谓混淆，就是将密文和明文的统计特性之间的关系复杂化。

注 记

Shannon 在 1948 年提出的信息论对后来发展出的编码理论及密码学产生了重大的影响。Shannon 可以利用信息论作为工具，从理论上定量分析密码系统的安全性[2]。

本章主要介绍了 Shannon 保密系统的模型、完善保密性的条件、惟一解距离、理论安全性和实际安全性等概念和观点，这些观点至今对密码学的研究仍具有重要的指导意义。文献[51]用惟一解距离分析了古典密码的破译问题，限于篇幅我们没有叙述。

关于信息论方面的知识，本章只做了简单介绍，感兴趣的读者可参阅本书参考文献[47]～[50]。

习 题 三

1．设密文空间共含有 5 个信息 $m_i = (1 \leq i \leq 5)$，并且

$p(m_1) = p(m_2) = 1/4$，$p(m_3) = 1/8$，$p(m_4) = p(m_5) = 3/16$，求 $H(M)$。

2．考虑一个密码体制 $M = \{a, b, c\}$，$K = \{k_1, k_2, k_3\}$ 和 $C = \{1,2,3,4\}$。假设加密矩阵为

	a	b	c
k_1	2	3	4
k_2	3	4	1
k_3	1	2	3

已知密钥概率分布为：$p(k_3)=1/2$，$p(k_1)=p(k_2)=1/4$，且明文概率分布为 $p(a)=1/3$，$p(b)=8/15$，$p(c)=2/15$，计算 $H(M)$，$H(K)$，$H(C)$，$H(M|C)$，$H(K|C)$。

3．证明公式 $H(X,Y)=H(Y)+H(X|Y)=H(X)+H(Y|X)$。

4．证明公式 $H(X,Y,Z)=H(X,Y)+H(Z|X,Y)$。

第 4 章　密码学的计算复杂性理论基础

从引论的直观说明中可知,计算复杂性理论是密码系统安全性定义(现代方法)的理论基础,也是安全的现代密码系统构造方法的理论依据。它给出求解一个问题的计算是“容易”还是“困难”的确切定义。由此对问题和算法的复杂性加以分类。由于这一理论对现代密码学的发展起着重要作用,所以本章将专门介绍问题与算法的复杂性理论。主要介绍密码学中要用到的基本概念和结论,想知道更多内容的读者可参看本书参考文献[1]和[31]。

4.1　问题与算法的复杂性

4.1.1　问题与语言

问题是指有待回答的一般性陈述或提问,通常包括若干未定参数或自由变量。一个问题的描述由两部分组成:(1) 对其所有参数的一般性描述;(2) 说明其答案或解应具有的特性。一个问题的例子是指给定了一组具体参数值后所对应的问题。

例 4.1　整数的因子分解问题。

任给一个整数 $n \neq 0$(参数),求一个整数 q,满足 $1 < q < n$ 且 q 整除 n。固定一个 n 就给出因子分解问题的一个例子。

例 4.2　背包问题。

有一个背包,其容积为 s,有 n 种东西,它们的体积分别为 $a_1, a_2, \cdots, a_n$,要从中挑出若干东西正好把背包装满。其中设 s 和 $a_1, a_2, \cdots, a_n$ 都是正整数。这一问题的数学描述即为:任给一 n-数组$(a_1, a_2, \cdots, a_n)$,$a_i \in \mathbf{Z}^+$(正整数集)和一正整数 s(参数)。求一个 n-数组$(x_1, x_2, \cdots, x_n)$,$x_i \in \{0,1\}$使 $\sum_{i=1}^{n} a_i x_i = s$。固定一组 s 和 $a_1, a_2, \cdots, a_n$,就给出背包问题的一个例子。

在千变万化的实际问题中,有一类问题起着特殊的作用,这类问题的答案只有“是”或“否”两种可能,称之为判定问题。实际应用中的绝大多数问题都可直接或间接地转化为

判定问题。如例 4.1 和例 4.2 可转述为:

例 4.1′　任给一个整数 $n\neq 0$,是否有一个整数 q 满足 $1<q<n$ 且 q 整除 n。

例 4.2′　任给一 n-数组$(a_1, a_2, \cdots, a_n)$, $a_i\in \mathbf{Z}^+$,和一正整数 s,是否存在一个 n-数组$(x_1, x_2, \cdots, x_n)$, $x_i\in\{0,1\}$,使 $\sum_{i=1}^{n} a_i x_i = s$。

一个判定问题 D 可由它的所有例子构成的集 I_D 和 I_D 中那些答案为"是"的例子构成的集 I_D^+ 来表示,记作 $D=(I_D, I_D^+)$。显然, I_D 中那些答案为"否"的例子构成的集 $I_D^- = I_D - I_D^+$。在含意清楚时我们省去 I_D, I_D^+, I_D^- 中的下标 D,分别记作 I, I^+, I^-,通常 D 的一个例子可用一组参数值 θ 表示,因此 I, I^+, I^- 都是数集或数组集。下面我们要将判定问题统一转化为语言成员的识别问题。先引进语言的概念。

考虑一个有限集 B 称为字符集,它的元素称为字符,字符的有限序列称为字。一个字 $x=b_1 b_2\cdots b_p$ 中的字符数 p 称为字 x 的长,记作$|x|$。若干个字可以通过连接运算构成一个新字。如字 $x=b_1 b_2\cdots b_p$ 和字 $x'=b_1' b_2'\cdots b_q'$ 可连接为 $xx'=b_1 b_2\cdots b_p b_1' b_2'\cdots b_q'$,其长$|xx'|=p+q$。约定一个空字 λ 为长为 0 的字,且 $\lambda x = x\lambda = x$ 对任一字 x 成立。记 B^* 为 B 中字符的所有有限长字(包括空字)所构成的集。

定义 4.1　B^* 的任一子集 L 称为一个 B-语言(或简称语言)。语言 L 中的字称为语言 L 的成员。

定义 4.2　设一个语言 $L\subseteq B^*$ 已给定。语言 L 成员的识别问题可描述为:任给 $x\in B^*$ (参数),问 x 是否是语言 L 的成员(是否 $x\in L$)?

判定问题(以下简称问题)到语言成员识别问题的转化是通过例子的合理编码来实现的。

定义 4.3　设 $D=(I, I^+)$ 为一个问题, B 为一个字符集。从 I 到 B^* 中的一个映射 c,满足条件 $c(I^+)\cap c(I^-)=\varnothing$(空集),称为问题 D 的一个 B-编码。若 c 为 D 的一个编码,集 $L(D, c)=\{c(\theta); \theta\in I^+\}=c(I^+)$称为 D 的一个 c-语言。

由定义 4.2 和定义 4.3 立即可得。

引理 4.1　若 c 为 D 的一个编码,则求解问题 D 和求解语言 $L(D, c)$的成员识别问题是等价的,即问题 D 的任一例子 $\theta\in I$,其答案与语言 $L(D, c)$的成员识别问题的例子 $c(\theta)$的答案是相同的。

不严格地说,一个合理编码还应满足下列两个基本要求:

(1) 编码是容易实现的;

(2) 求解问题 D 的任一例子 θ 的计算复杂性(通常用计算时间来表示)与 $c(\theta)$的长$|c(\theta)|$有某种正比关系。

合理编码的要求(2)的意义在于它将研究一般问题的计算复杂性分类化为研究语言成员识别问题的计算复杂性分类,这就便于用统一的计算模型来定义。

关于合理编码的严格定义和一般构造方法已超出了本书的介绍范围,这里仅通过例

子对一类实际问题的合理编码给出一种直观的构造方法。这类问题的特性是:其例子的计算复杂性与有限个正整数的大小成某种正比关系,密码学中遇到的问题多半属于这一类。

例 4.1″ 在例 4.1′的因子分解问题中, $D=(I, I^+)$,其中 $I=\mathbf{Z}$(整数集) - $\{0\}$, I^+ 为所有合数构成的集, $I^-=I-I^+$ 为所有正负素数构成的集。在一般情况下, D 的例子 $\theta=\pm n$的计算复杂性依赖于正整数 n 的大小。将 n 表示为二进制数,即

$$n = b_1 b_2 \cdots b_k = \sum_{i=1}^{k} b_i 2^{k-i},\ k = \lfloor \log_2 n \rfloor + 1。 \qquad (4.1)$$

其中 $\lfloor \alpha \rfloor$ 表示不大于 α 的最大整数,例如 $7=111=4+2+1$, $8=1000=8+0+0+0$, $13=1101=8+4+0+1$。再用一位二进制数 b_0 表示例子 θ 的答案,即 $b_0=1$,若 $\theta\in I^+$, $b_0=0$,若 $\theta\in I^-$,构造问题 D 的一个编码 c 为 $c(\theta)=c(\pm n)=b_0 b_1 \cdots b_k$,其中 $k=\lfloor \log_2 n \rfloor + 1$, c 满足合理编码的两个要求。

例 4.2″ 在例 4.2′的背包问题中, $D=(I, I^+)$,其中

$$I = \{(a_1, a_2, \cdots, a_n, s) \mid a_i\in\mathbf{Z}^+, i=1,\cdots,n, s\in\mathbf{Z}^+\},$$

$$I^+ = \{(a_1, a_2, \cdots, a_n, s) \mid (a_1, a_2, \cdots, a_n, s)\in I,$$

$$s=\sum_{i=1}^{k} x_i a_i,\ (x_1, x_2, \cdots, x_n)\in\{0,1\}^n - \{(0,0,\cdots,0)\}\},$$

$I^-=I-I^+$。在一般情况下, D 的例子 $\theta=(a_1, a_2, \cdots, a_n, s)$的计算复杂性依赖于 $a_1, a_2, \cdots, a_n, s$ 的大小,将它们表示为二进数, $a_i=b_{i1}b_{i2}\cdots b_{ik_i}$, $k_i=\lfloor \log_2 a_i \rfloor + 1$, $i=1,2,\cdots,n$, $s=s_1 s_2 \cdots s_k$, $k=\lfloor \log_2 s \rfloor + 1$,仍用 b_0 表示例子 θ 的答案。构造 D 的$\{0,1\}$ - 编码 c 为 $c(\theta)=c(a_1, \cdots, a_n, s)=b_0 b_{11}\cdots b_{1k_1} b_{21}\cdots b_{2k_2}\cdots b_{n1}\cdots b_{nk_n} s_1 \cdots s_k$, c 满足合理编码的要求。

以后将会看到,对于一个问题 D 的两个不同的合理编码 c 和 c',若它们多项式相关,即存在两个多项式 $p(n)$和 $p'(n)$使对任一 $\theta\in I$ 有 $|c(\theta)|\le p(|c'(\theta)|)$和 $|c'(\theta)|\le p'(|c(\theta)|)$成立,则语言 $L(D, c)$和 $L(D, c')$的成员识别问题有同样的计算复杂性分类。由于合理编码的要求(2)就蕴含了要求任意两个合理编码是多项式相关的,因此不失一般性,今后我们总可假定对每个问题 D 已固定了一个字符集为$\{0,1\}$的合理编码而不管它是哪一个。

4.1.2 算法与图灵机

算法是指可用来求解某一问题的、一步一步地执行的计算过程。通常可理解为在通用计算机上求解某个问题所编写的计算程序。称一个算法可解某个问题是指这个算法可应用于这个问题的任何例子并求得其解答。若至少存在一个算法可解这个问题,就称这个问题是可解的(可判定的,可计算的),否则就称这个问题是不可解的(不可判定的,不可计算的)。

一个算法的计算复杂性或有效性可以用执行该算法所需的运行时间和内存空间来度量,但在实际应用中,特别是在密码学应用中,人们更关心的是在产生最终答案前算法所花费的时间。因此,我们今后只研究算法的时间复杂性,但一个算法的运行时间随着选用的计算语言、用这一语言编写程序的方法以及计算所用的计算机等因素的改变会有很大的差别。为了克服这个问题带来的困难,在计算复杂性理论研究中,采用统一的计算模型。由于这一计算模型是英国数学家图灵(A .Turing)在 1936 年首先提出的,因此后人称它为图灵机。图灵机模型虽然简单,但它的计算能力相当于通用计算机。一般可用通用计算机求解的问题也可用图灵机求解。因此,图灵机至今仍被广泛应用于计算复杂性的理论研究中。下面先给出确定性单带图灵机的简单描述,然后再给出它的正式数学定义。

图灵机是计算机的一个理想化的数学模型,它由一条无限长的磁带和一个读写头组成,如图 4 .1 所示。

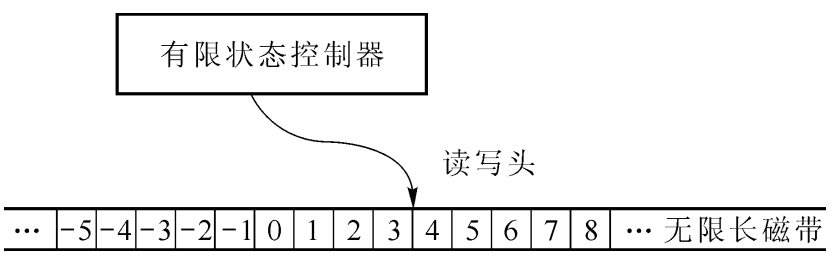


图 4 .1　确定性单带图灵机示意图

磁带被划分为一系列小方格,每个小方格可写上一个二进数(0 或 1)或空。读写头可沿磁带左右移动扫描磁带上每一小方格中的内容。在每一离散时刻读写头处于一个具体的内部状态(所有可能的状态数是有限的),且包含若干指令,规定由现在所处状态 s 和所读字符 b 转移到状态 s' 和写字符 b' 代替 b(允许 $b' = b$),且按指令规定读写头向左或向右移动一格或保持原位不动。正式数学定义如下。

定义 4 .4　一个确定性单带图灵机由下列集和函数构成:

1 . (1) 带中所用字符集 B,通常可设 $B = \{0, 1, \lambda\}$,其中 λ 表示空。

(2) 读写头所处的可能状态集 S,其中包含一个初始状态 s_0 和若干个停机状态 T　S。

(3) 读写头所处状态的转移函数 σ,它是读写头现在所处状态 s 和所读字符 b 的函数,表示为 $\sigma: S \times B \to S$。

(4) 读写头动作的指令函数 μ,它也是读写头现在所处状态 s 和所读字符 b 的函数,表示为 $\mu: S \times B \to B \cup \{l, r\}$,其中 $l \neq r$ 且都不属于 B。若 $\mu(s, b) = b' \in B$,则读写头写字符 b' 代替 b,且保持原位不动。若 $\mu(s, b) = l$(或 r),则原字符 b 保持不变,读写头向左(或向右)移动一个小方格。

2 . 磁带上的每个小方格用一个整数坐标 i 表示(见图 4 .1)。小方格 i 中的字符记作 $t(i)$,磁带表示为函数 $t: \mathbf{Z}$(整数集)$\to B$。

3.图灵机在某一时刻的形是指一个三元组(s, t, i),它们分别表示该时刻读写头所处状态,磁带和读写头所扫描的小方格坐标,$t(i)$为读写头在该时刻所读字符。

4.一个图灵机的计算程序(算法)是一个形的有限或无限序列$(s_0, t_0, i_0), (s_1, t_1, i_1), (s_2, t_2, i_2), \cdots$,其中$(s_0, t_0, i_0)$为图灵机在初始时刻的形,即$s_0$为初始状态,$t_0$为初始磁带,它由输入数据(字)$x \in B^*$给出,通常存放在$1 - |x|$小方格中,其他小方格中为空字符$\lambda$,通常$i_0 = 1$。图灵机在$k$时刻的形$(s_k, t_k, i_k)$,$k = 1, 2, \cdots$由下面的递推式给出。

$$s_k = \sigma(s_{k-1}, t_{k-1}(i_{k-1})) \tag{4.2}$$

若$\mu(s_{k-1}, t_{k-1}(i_{k-1})) = b \in B$,则$i_k = i_{k-1}$

$$t_k(i) = \begin{matrix} b & i = i_{k-1} \\ t_{k-1}(i) & i \neq i_{k-1} \end{matrix} \tag{4.3}$$

若$\mu(s_{k-1}, t_{k-1}(i_{k-1})) = l$,则$t_k = t_{k-1}$, $i_k = i_{k-1} - 1$

若$\mu(s_{k-1}, t_{k-1}(i_{k-1})) = r$,则$t_k = t_{k-1}$, $i_k = i_{k-1} + 1$

若存在形(s_k, t_k, i_k)使$s_k \in T$,则计算在时刻$K = \min(k; s_k \in T)$终止,同时停机,称(t_k, i_k)或$t_k(i_k)$为计算的输出结果,K称为图灵机(算法)的运行(计算)时间;否则计算将不终止,不停机,直到无限。

图灵机的定义虽然有点长,但它很重要,有了这个理论的算法模型,我们就可来定义算法的计算复杂性了。以后我们把一个图灵机看作一个算法,不再说明。根据4.1.1节的讨论,在研究算法的计算复杂性时,只要考虑语言的成员识别问题即可。由于一个语言的成员识别问题的例子集$I = \{0,1\}^*$,一个算法的计算复杂性可定义为输入图灵机的数据$x \in \{0,1\}^*$的长$|x| = n$的函数。由于同样长的例子的计算时间也可能不同,我们考虑最坏情况,即用其中计算时间最长的例子来定义其计算复杂性。正式定义如下。

定义4.5 称一个图灵机M可解一个语言L的成员识别问题,若对任一输入数据$x \in \{0,1\}^*$,M在有限时刻$K(x)$停机,且M的输出$t_{K(x)}(i_{K(x)}) = 1$,则$x \in L$;否则$t_{K(x)}(i_{K(x)}) = 0$。图灵机M的计算复杂性定义为

$$f_M(n) = \max_{x;\,|x|=n} [K(x)]。 \tag{4.4}$$

由于算法的计算复杂性是正整数的函数,因此,要比较两个算法的计算复杂性主要是比较当n充分大时它们随n增大而增大的量级。为此,我们要引进计算复杂性理论中常用的符号O和Θ。

定义4.6 设$f(n)$和$g(n)$为两个正整数函数,若存在正整数n_0和常数c使当$n \geq n_0$时有$f(n) \leq cg(n)$,则记作$f(n) = O(g(n))$;若$f(n) = O(g(n))$,$g(n) = O(f(n))$,则记作$f(n) = \Theta(g(n))$。

由定义4.6可见,$f(n) = O(g(n))$的含意是:当n充分大时,$f(n)$增长的量级小于等于$g(n)$增长的量级。$f(n) = \Theta(g(n))$的含意是:当n充分大时,$f(n)$和$g(n)$增长

的量级相同。例如，$f(n)=O(n^3)$意味着当 n 充分大时，近似地有 $f(2n)/f(n)\leq(2n)^3/n^3=8$。$f(n)=\Theta(g(n))$意味着当 n 充分大时近似地有，$f(2n)/f(n)=8$。

不难证明 O 和 Θ 有下列性质：

(1) $f(n)=O(g(n)),g(n)=O(h(n))\quad f(n)=O(h(n))$，
$f(n)=\Theta(g(n)),g(n)=\Theta(h(n))\quad f(n)=\Theta(h(n))$。

(2) $f(n)=n^d+c_1n^{d-1}+\cdots+c_{d-1}n+c_d$ 为一 d 次多项式　$f(n)=\Theta(n^d)$　$f(n)=O(n^{d'})$，$d'\geq d$。

(3) 对任一正多项式 $p(n)$及任一大于 1 的整数 m 有 $p(n)=O(m^n)$。

习题：证明上述三个性质。

下面列出若干表示各种增长量级的函数，按从小到大的次序排列：

1（常数），$\log n\rfloor$（对数函数），n(线性函数)，n^2（二次函数），n^3（三次函数），$n^d(1\leq d<\infty)$(多项式函数)，$2^{\log n\rfloor}$（亚指数函数），2^n，10^n(指数函数)。

应用符号 O 和 Θ 可以比较算法的有效性和对算法的计算复杂性进行分类。

定义 4.7　设 $f_M(n)$和 $f_{M'}(n)$为图灵机 M 和 M'的计算复杂性，若 $f_M(n)=O(f_{M'}(n))$，则称算法 M'不比算法 M 有效；若 $f_M(n)=\Theta(f_{M'}(n))$，则称算法 M 和 M'是等效的；若存在正整数 d，$f_M(n)=O(n^d)$，则称 M 为多项式时间算法，按密码学中的传统观念，认为多项式时间算法为有效算法；若 $f_M(n)=\Theta 2^{\log n}$，则称 M 为亚指数时间算法；若 $f_M(n)=\Theta(2^n)$或 $\Theta(10^n)$，则称 M 为指数时间算法。亚指数和指数时间算法也被称为超多项式时间算法，被认为不是有效算法。

要说明一点，从理论上讲，不是任何两个算法都能比较的，但在实际中，可解一个实际问题的不同算法一般都是可以比较的。

4.2　问题的计算复杂性分类

4.2.1　P、NP、NP 完全类问题

一个问题的计算复杂性是由可解这个问题的算法的计算复杂性所决定的。由于可解一个问题的算法可能有多个，它们的计算复杂性也各不相同，故在理论上定义一个问题的计算复杂性为可解该问题的最有效算法的计算复杂性。由于要证明一个算法是解某一问题的最有效算法是很困难的，因此在实际应用中，只能把解该问题的已知最有效算法的计算复杂性粗分为三类，即 P 类(确定性多项式时间可解类)、NP 类(不确定性多项式时间可解类)和 NP 完全类(不确定性多项式时间可解完全类)。P 类问题称为易解问题，NP 类问题称为难解问题，NP 完全类问题称为困难问题。密码学中主要关心后两类问题，因为这两类问题的求解在计算上是不可行的。由于一般问题可化为语言的成员识别问题，

我们只对语言的成员识别问题给出分类定义。

定义 4.8 一个语言 L 的成员识别问题属于 P 类,若存在一个可解该问题的图灵机 M 和一个正多项式 $p(n)$,使 M 的计算复杂性 $f_M(n)=O(p(n))$,所有 P 类问题构成的集记作 P。

为了定义 NP 类问题,传统的方法是先要定义非确定性图灵机,一个非确定性图灵机与确定性图灵机的主要区别就是在前者的计算程序中,下一时刻的形(s,t,i)并不是由现在时刻的形惟一确定的,而是有多种选择;而在后者的计算程序中,下一时刻的形是由现在时刻的形惟一确定的。为了简单起见,我们直接给出下面的等价定义。

定义 4.9 一个语言 L 的成员识别问题属于 NP 类,若存在一个$\{0,1\}^*\times\{0,1\}^*$的子集 $R_L=\{(x,y)\}$(称为一个布尔关系)及一个正多项式 $p(n)$满足下列两个条件:

(1) R_L 的成员识别问题属于 P 类;

(2) $x\in L$ 当且仅当存在一个 y,其长$|y|\le p(|x|)$,且$(x,y)\in R_L$。这样的 y 称为是 $x\in L$ 的证据。所有 NP 类问题构成的集记作 NP。

若 L 的成员识别问题属于 P 类,则它也属于 NP 类,因为 $R_L=\{(x,1);\ x\in L\}$就满足定义 4.9 中的条件,故 P NP。虽然大家相信 $P\neq NP$,但证明这个结论至今还是计算机科学中一个著名的公开问题。NP 类问题的难解性体现在要试探所有可能与 x 有关系的 y,检验(x,y)是否是 R_L 的成员。

定义 4.10 称一个图灵机 M 可计算一个函数 $f(x):\{0,1\}^*\to\{0,1\}^*$,若对任一输入数据 $x\in\{0,1\}^*$,M 在有限时刻 $K(x)$停机,且 M 的输出磁带 $t_{K(x)}$ 上的二进数序列(不包含空 λ) $M(x)=f(x)$。若 M 是多项式时间算法,则称 $f(x)$是多项式时间可计算的。

定义 4.11 一个语言 L 称为可多项式时间化为另一语言 L',若存在一个多项式时间可计算函数 $f(x)$,使 $x\in L$ 当且仅当 $f(x)\in L'$,这时也称语言 L 的成员识别问题可多项式时间化为语言 L'的成员识别问题。

定义 4.12 一个语言 L 的成员识别问题属于 NP 完全(NPC)类,若它属于 NP 类,且每个 NP 类语言成员识别问题都可多项式时间化为语言 L 的成员识别问题。所有 NP 完全类问题构成的集记作 NPC。

NP 完全类问题所以称为困难问题是因为若 P≠NP 果真成立,则至少有一个语言 L 的成员识别问题属于 NP 而不属于 P,故任何可解 L 的成员识别问题的算法都是超多项式时间算法。由定义 4.12,L 的成员识别问题可多项式时间化为任一 NP 完全类问题,因此所有 NP 完全类问题都有超多项式时间计算复杂性,否则将与 L 的成员识别问题不属于 P 矛盾。一般说来,若语言 L 可多项式时间化为语言 L',则认为 L 的成员识别问题不比 L'的成员识别问题难。故 NP 完全类问题被认为是最难的问题。

例 4.3 可满足性问题。

可满足性问题是数理逻辑中的一个重要问题。考虑一个有限或可数集 $\chi=\{x_1,\cdots,$

$x_n, \cdots\}$,它的元素 x_i 称为变量,$\overline{x_i}$称为 x_i 的共轭变量,y 取值为 x_i 或$\overline{x_i}$称为一个字,若 $y_1, y_2, \cdots, y_k$ 为不同的字,则集$\{y_1, y_2, \cdots, y_k\}$称为一个 k 次子句,记作 $C = y_1 \vee y_2 \vee \cdots \vee y_k$。若 $C_1, C_2, \cdots, C_m$ 为 m 个子句,则集$\{C_1, C_2, \cdots, C_m\}$称为一个公式,记作 $A = C_1 \wedge C_2 \wedge \cdots \wedge C_m$。一个二值函数 $f(x)$: $x \to \{0,1\}$称为一个真(假)值函数,$f(x_i) = 1$(真),$f(x_i) = 0$(假)。函数 f 可按如下规则延拓到字、子句和公式,令 $f(\overline{x_i}) = 1 - f(x_i)$,$f(y_1 \vee y_2 \vee \cdots y_k) = \max\limits_{1 \le i \le k} f(y_i)$,$f(C_1 \wedge C_2 \wedge \cdots \wedge C_m) = \min\limits_{1 \le j \le m} f(C_j)$。有时直接写成 $x_i = 0$ 或 1,代替 $f(x_i) = 0$ 或 1,这时$\vee$,$\wedge$,$^-$分别表示逻辑运算或、与、非。公式 A 称为可满足当且仅当存在一个真值函数 f 使 $f(A) = 1$。可满足性问题可描述如下:

任给一个公式 A(参数),是否存在一个真值函数 f,使 $f(A) = 1$。

Cook (1971)证明可满足性问题是 NP 完全类问题。自此以后人们已证明了数千个 NP 完全类问题。有了一批已知的 NP 完全类问题,再要证明一个 NP 类问题是一个 NP 完全类问题,就只要证明某个已知的 NP 完全类问题可多项式时间化为它即可,这是因为若 $f(x)$和 $g(y)$是两个多项式时间可计算函数,则它们的复合函数 $h(x) = g(f(x))$也是多项式可计算函数。

4.2.2　概率算法与 BPP 类问题

概率算法或称随机算法在密码学中起着重要作用,发送者可用它对消息概率加密,搭线者也可用它来作密码分析,因此有必要对它作些介绍。直观地说,概率算法就是在执行计算的过程中允许用随机数。例如随机扔一枚硬币,若结果出正面,则表示 0,出反面则表示 1,0 和 1 出现的概率都是 1/ 2。算法的每一步计算都有两个不同的选择。按随机数出现的结果是 0 还是 1 来选择第一种计算还是第二种计算。在通用计算机中,一般都有产生随机数的子程序,只要调用即可。在计算复杂性理论中,则引入概率图灵机作为概率算法的统一计算模型。

一个概率图灵机或称随机图灵机比一个确定性图灵机多一个随机数生成器,称为内部扔硬币,各次扔硬币的结果是相互统计独立的随机变量(随机数)。概率图灵机的读写头的状态转移函数 σ和动作指令函数 μ都有两个不同的可能取值,取哪个值依赖于一个二进数 $a = 0$ 或 1,它们分别表示为

$$\sigma: S \times B \times \{0,1\} \to S,$$

$$\mu: S \times B \times \{0,1\} \to B \cup \{l, r\}。$$

一个概率图灵机的计算程序(概率算法)也是一个形的有限或无限序列(s_0, t_0, i_0),(s_1, t_1, i_1),(s_2, t_2, i_2),$\cdots$,其中(s_0, t_0, i_0)是概率图灵机在初始时刻的形,与确定性的图灵机一样,它也是由初始状态和输入数据给定的,概率图灵机在 k 时刻的形(s_k, t_k, i_k),$k = 1,2,\cdots$是由定义 4.4 中的递推式(4.2)和(4.3)将其中的 $\sigma(s_{k-1}, t_{k-1}(i_{k-1}))$和 $\mu(s_{k-1}, t_{k-1}(i_{k-1}))$分别换成 $\sigma(s_{k-1}, t_{k-1}(i_{k-1}), a_{k-1})$和 $\mu(s_{k-1}, t_{k-1}(i_{k-1}), a_{k-1})$给

出,其中 a_{k-1}为 $k-1$时刻随机扔硬币的结果。由假定 $a_0, a_1, \cdots, a_{k-1}$是一个相互统计独立在集$\{0,1\}$上等概分布的随机变量序列,因此其结果 r 可能是$\{0,1\}^k$ 中的任一二进数序列,$\{0,1\}^k$ 中的每个二进数序列出现的概率都是 $1/2^k$。由形的递推式可见,概率图灵机的形序列也是一个随机变量序列,每个扔硬币结果 $r\in\{0,1\}^k$ 对应一个确定的形序列$(s_0, t_0, i_0)_r, (s_1, t_1, i_1)_r, \cdots, (s_k, t_k, i_k)_r$,它的出现概率也是 $1/2^k$。于是对每个确定的形序列$(s_k, t_k, i_k)_r$, $k=0,1\cdots$,可以如定义 4.4 一样地定义停机时间 K_r 和输出结果(t_{K_r}, i_{K_r})或 $t_{K_r}(i_{K_r})$,为了表示停机时间 K_r 依赖图灵机的输入数据 x,以下将它记为 $K_r(x)$,并将输出结果 $t_{K_r(x)}(i_{K_r(x)})$记为 $b_r(x)$。由以上描述可知,若一个概率图灵机的输入数据为 $x\in\{0,1\}^*$,则其运行(计算)时间 $K(x)$和输出 $b(x)$都是随机变量,它们分别为 $K_r(x)$和 $b_r(x)$的概率为 $1/2^{K_r(x)}$,记作

$$\Pr\{K(x)=K_r(x)\}=1/2^{K_r(x)}, \Pr\{b(x)=b_r(x)\}=1/2^{K_r(x)}。$$

定义 4.13 一个概率算法(图灵机)称为多项式时间概率算法。若存在一个多项式 $p(n)$,对任一 $x\in\{0,1\}^*$,有 $\Pr\{K(x)\leq p(|x|)\}=1$。换句话说,对所有扔硬币结果 r(可设$|r|\leq p(|x|)$)都有 $K_r(x)\leq p(|x|)$。

今后我们只考虑多项式时间概率算法。顺便提醒读者注意,概率图灵机与不确定性图灵机有本质区别,不确定性图灵机是不现实的假想模型,目的是为了搜索 x 为语言成员的证据 y(参看定义 4.9),而概率图灵机与确定性图灵机一样,完全是现实的。

定义 4.14 称一个多项式时间概率算法 M 可解一个语言 L 的成员识别问题,若对任一输入数据 $x\in\{0,1\}^*$,有

(1) 若 $x\in L$,则 $\Pr\{b(x)=1\}\geq 2/3$, (4.5)

(2) 若 $x\notin L$,则 $\Pr\{b(x)=0\}\geq 2/3$, (4.6)

称一个语言 L 的成员识别问题属于 BPP 类,若存在一个可解该问题的多项式时间概率算法。所有 BPP 类问题构成的集记作 BPP。

可以证明,在定义 4.14 的(4.5)式和(4.6)式中,常数 $2/3$ 可换成 $1-2^{-p(|x|)}$,并不影响 BPP 类问题的定义,其中 $p(n)$为任一正多项式,换句话说,任何 BPP 类语言成员识别问题,可用多项式时间概率算法求解,其错误概率是可以忽略的,称一个函数 $f(n)$是可以忽略的,若对任一多项式 $p(n)$,存在 n_0 使当 $n\geq n_0$ 时,有 $f(n)<1/p(n)$。

由于多项式时间算法可以看作多项式时间概率算法的特殊情形。故 P $\subseteq$ BPP,但 P$\neq$ BPP 是否成立也还未得到证明。在现代密码学中,由于安全性考虑,有些作者认为多项式时间概率算法也是有效算法,基于这一观点所设计的密码系统将有更强的抗攻击能力。

在结束本章之际,还要作一点说明。本章引进语言成员识别问题和图灵机,目的是让读者更好地利用计算复杂性理论的研究成果去设计安全的密码系统。但这一理论不够直观,故我们并不要求读者自己去构造图灵机求解实际问题。在应用中,读者仍可用通常的算法求解实际问题。通过估计执行算法中需要的整数的基本运算(加、乘、比较等)次数,

再将它们都化为比特(一位二进数)运算次数,来估计算法的计算复杂性(参看本书参考文献[13],[33])。

注 记

计算复杂性理论是现代密码学的理论基础，每本现代密码学教材中，或多或少都要介绍一点计算复杂性理论。关于算法的时间复杂性有两种定义方法：一是用一个图灵机表示一个算法，该算法的时间复杂性定义为该图灵机运行的步数，如文献[1],[26],[14]等;二是定义一个算法的时间复杂性为该算法的比特运算次数,如本书参考文献[13],[33]等。本章使用前者。本章主要参考了本书参考文献[1]和[26]。关于判定问题到语言成员识别问题的合理编码参考了文献[12],关于估计算法的比特运算次数的方法可参看文献[13]的第一章第一节和文献[33]的1.4节,关于NP完全问题,可参看[26]第3章所引的文献[3]。关于图灵机的定义可参看文献[31]的第X章第一节。

习 题 四

1.举出一个不同于例4.1和4.2的问题,并将它转述为判定问题 D,再将 D 用它的例子表示为 $D=(I_D, I_D^+)$。

2.证明定义4.6定义的符号O和Θ有那里所述的性质(1),(2),(3)。

3.设 $f(n)=O(\log_2 n!)$, $g(n)=O(n^2)$, $h(n)=O(2^{\log_2 n!})$,求出下列函数的增长量级,并给出证明:

(a) $f(n)+g(n)$　(b) $f(n)g(n)$　(c) $f(n)h(n)$　(d) $f(g(n))$,设 $f(n)$ 为增函数。

4.设 $f(n)=\Theta(\log_2 n!)$, $g(n)=\Theta(p(n))$其中 $p(n)$为一 d 次正多项式, $h(n)=\Theta(2^n)$,求出下列函数的增长量级,并给出证明:

(a) $h(n)/g(n)$　(b) $f(n)h(n)$　(c) $g(n)/f(n)$　(d) $g(n)^{f(n)}$

5.证明在定义4.14中条件(1)(4.5式)和条件(2)(4.6式)中的常数2/3可以换成$1/2+1/p(|x|)$,其中 $p(n)$为任一正多项式,并不影响BPP类问题的定义。

提示　给了一个多项式时间概率算法 M 满足条件(4.5)和(4.6),构造一个新的多项式时间概率算法 M 如下, M 输入为 x,用 M 和同样的输入 x 进行 $O(p(|x|)^2)$次统计独立的计算。用多数准则决定 M 的输出,并对统计独立的 $O(p(|x|)^2)$个输出应用下面的Chebyshev不等式。

Chebyshev不等式:设 $x_1, x_2, \cdots, x_n$ 为 n 个统计独立同分布的随机变量, x_i 的相同

均值和方差分别记作 μ 和 σ,则对每个 ξ>0,有

$$\Pr\left|\frac{1}{n}\sum_{i=1}^{n}x_i-\mu\right|\geq\xi\leq\sigma^2/\xi^2 n$$

6.证明在定义 4.14 中条件(1)(4.5 式)和条件(2)(4.6 式)中的常数 2/3 可以换成 $1-2^{-p(|x|)}$ 并不影响 BPP 类问题的定义,其中 $p(n)$ 为任一正多项式。

提示　类似于习题 5,但要用一个更强的不等式,称为 Chernoff 界。

Chernoff 界:设 $p\leq 1/2$ 和 $x_1, x_2, \cdots, x_n$ 为 n 个统计独立同分布的在 $\{0,1\}$ 中取值的随机变量,且 $\Pr\{x_i=1\}=p(1\leq i\leq n)$,则对一切 ξ,$0<\xi\leq p(1-p)$,有

$$\Pr\left|\frac{1}{n}\sum_{i=1}^{n}x_i-\mu\right|>\xi<2e^{\frac{\xi^2 n}{2p(1-p)}}$$

7.(1) 将正整数 160 和 199 表示为二进数;

(2) 求 160 + 199 的比特(一位二进数)运算次数;

(3) 求 160×199 的比特运算次数;

(4) 计算二进数 10011001 被 1011 除,并求出其比特运算次数。

8.用 O(f(n))(f(n)为 n 的简单函数)估计下列计算的比特运算次数。

(1) 5^n　　(2) n^n　　(3) $n!$

第 5 章　单向函数

在引论中已经提到,单向函数是计算复杂性理论方法引入的新概念,是现代密码学的一个基本工具。单向函数可用于构造伪随机序列生成器(密钥流生成器),单向陷门函数可用来构造公钥密码体制,单向杂凑(Hash)函数可用于数字签名和消息完整性(防改动)检测等。作为这些应用的基础,本章介绍单向函数的严格定义和一般理论。

5.1　一般单向函数

5.1.1　单向函数的定义

考虑函数 $f:\{0,1\}^* \to \{0,1\}^*$。为了避免不必要的麻烦,下面主要考虑正则单向函数。

定义 5.1　函数 $f:\{0,1\}^* \to \{0,1\}^*$ 称为正则的,若存在正多项式 $p(n)$ 使对任一 $x \in \{0,1\}^*$,有 $p(|f(x)|) \geq |x|$。正则函数的含意是 $f(x)$ 的长不比 x 的长缩短太多。正则函数的一个重要的特殊情形是保长函数,即对任一 $x \in \{0,1\}^*$,有 $|f(x)| = |x|$。

以下若不特别说明,总设函数 $f(x)$ 为正则函数。

定义 5.2　一个函数 $f:\{0,1\}^* \to \{0,1\}^*$ 称为强单向函数,若下列两个条件成立:

(1) 计算 $f(x)$ 是容易的,即 $f(x)$ 是多项式时间可计算的(参看定义 4.10);

(2) 计算 $f(x)$ 的逆 $f^{-1}(f(x))$ 是困难的,即对每一多项式时间概率算法(参看定义 4.13) M',每一正多项式 $p(n)$ 和一切充分大的 $n(n \geq n_0)$ 有

$$\Pr[M'(f(U_n)) \in f^{-1}(f(U_n))] < 1/p(n)。 \tag{5.1}$$

其中 U_n 表示一个在 $\{0,1\}^n$ 上均匀(等概)分布的随机变量,假定它与算法 M' 的内部扔硬币结果是统计独立的,(5.1)式中的概率是对 U_n 的所有可能值和扔硬币的所有可能结果而取的,$f^{-1}(f(x))$ 表示 $f(x)$ 的所有原像构成的集,当 $f(x)$ 是 1-1 映射,则 x 是惟一的原像,在一般情况下可能还有其他原像。

在定义 5.2 中,计算 $f(x)$ 的逆是困难的解释为用任一多项式时间概率算法来计算 $f(x)$ 的逆,若算法的输入 $y = f(x)$ 由 $\{0,1\}^n$ 中随机抽取 x 给出,其成功概率(作为

$|x|$的函数)是可忽略的,这就蕴含计算 $f(x)$的逆这一问题不属于 BPP 类(参看定义 4 .14)。若 NP 不包含在 BPP 中,则单向函数求逆至少是一个 NP 类问题;另一方面,条件(5 .1)式中,成功概率可以是一个小的正数,因此可能有少量 x 给出的 $f(x)$可用多项式时间概率算法求逆。为了更好地理解对成功概率所加的条件(5 .1)式,下面考虑两个最简单的算法。

(1) 随机猜测算法 M_1。无论输入哪个 $y=f(x)$, $x\in\{0,1\}^n$, M_1 总是输出 n 次扔硬币结果 r,作为对 x 的猜测。将 M_1 代入(5 .1)式。因 U_n 与 r 统计独立,故得

$$\Pr\ M_1(f(U_n))\in f^{-1}(f(U_n)) = \sum_x 2^{-n}\sum_r 2^{-n}\delta(r,x)\geq 2^{-n}。 \qquad (5.2)$$

其中

$$\delta(r,x)=\begin{cases}1 & 若\ r\in f^{-1}(f(x))\\ 0 & 其他\ r\end{cases}。 \qquad (5.3)$$

当 $f(x)$为 1-1 映射时,(5 .2)式中最后的不等式变为等式。这是因为 $f^{-1}(f(x))$中仅有 x 一个原像。由(5 .2)式和(5 .3)式可见:①若 $f(x)$为任一函数,则 M_1 的成功概率是一个正数(大小可从 2^{-n}到 1),因此不能要求任一多项式时间概率算法都不可计算函数 $f(x)$的逆;②若 $f(x)$为任一 1-1 映射,则 M_1 的成功概率是可忽略的,当然,这并不表示 $f(x)$为单向函数,因为 M_1 只是一个最简单的算法;③若 $f(x)$为一单向函数(不要求为 1-1 映射),则由于 M_1 为多项式时间概率算法,故(5 .1)式要求 M_1 的成功概率是可忽略的。

(2) 固定输出算法 M_2。无论输入哪个 $y=f(x)$, $x\in\{0,1\}^n$, M_2 总是输出一个固定的 $x'\in\{0,1\}^n$,将 M_2 代入(5 .1)式得

$$\begin{aligned}&\Pr\ M_2(f(U_n))\in f^{-1}(f(U_n)) = \Pr\ x'\in f^{-1}(f(U_n))\\ &= \sum_x 2^{-n}\delta(x',x) = |f^{-1}(f(x'))|2^{-n}\geq 2^{-n}。\end{aligned} \qquad (5.4)$$

其中 $\delta(x',x)$由(5 .3)式给出,(5 .4)式中第二个等式是由于 $x'\in f^{-1}(f(x))$ $x\in f^{-1}(f(x'))$, $|f^{-1}(f(x'))|$表示集 $f^{-1}(f(x'))$中的元素个数。当 x'是 $f(x')$的惟一原像时,(5 .4)式中最后一个不等式变为等式。由(5 .4)式可见, M_2 的成功概率与 M_1 有类似的结论。读者可以自己讨论。

在某种意义上,定义 5 .2 相当于说,对于单向函数,用最有效的多项式时间概率算法求逆,其成功概率也不会比用上述最简单算法求逆有实质性的改进。

定义 5 3 一个函数称为弱单向函数,若下列两个条件成立:

(1) 计算 $f(x)$是容易的,它与定义 5 .2 的(1)相同;

(2) 计算 $f(x)$的逆 $f^{-1}(f(x))$是稍难的,即存在一个正多项式 $p(n)$,使对每一多项式时间概率算法 M'和一切充分大的 $n(n\geq n_0)$有

$$\Pr\ M'(f(U_n))\notin f^{-1}(f(U_n)) > 1/p(n)。 \qquad (5.5)$$

在定义 5 .3 中,计算 $f(x)$的逆是稍难的解释为用任一多项式时间概率算法来计算 $f(x)$的逆,若算法的输入 $y = f(x)$由 $\{0,1\}^n$ 中随机抽取 x 给出,其失败概率(作为 $|x|$的函数)是显著的。一个函数 $g(n)$称为是显著的,若存在一个正多项式 $p(n)$,使当 $n \geq n_0$ 时,有 $g(n) > 1/p(n)$。从表面上看,弱单向函数的条件(5 .5)式要比强单向函数的条件(5 .1)式弱得多,从(5 .5)式看不出弱单向函数的求逆问题不属于 BPP 类,但从理论上可以证明,有一个标准的方法可将一个弱单向函数变换为一个强单向函数。因此,在某种意义上这两个概念是等价的。

显然,定义 5 .2 和定义 5 .3 可以推广到定义在 $\{0,1\}^*$ 的一个无穷子集上的单向函数。如定义在 $\bigcup_{n=1}^{\infty}\{0,1\}^{l(n)}$ 上的单向函数,其中 $l(n)$ 为一正多项式。

5 1 2　候选单向函数

在引论中已提到,前面定义的单向函数,它的存在性在理论上还没有给出证明。但有一些函数,经过人们长期的研究,至今未找到多项式时间求逆算法(包括概率算法),这些函数被称为候选单向函数。下面给出几个例子。

例 5 1　整数因子分解(参看例 4 .1)

$$f_{\text{mult}}(x,y) = x \cdot y, |x| = |y|。 \tag{5.6}$$

其中 $x, y, x \cdot y$ 分别表示整数 x, y 和它们乘积的二进数表示(参看(4 .1)式)。

显然, $f_{\text{mult}}(x,y)$是多项式时间可计算的。设 x, y 为两个随机选取的 n 比特长(n 位二进数)素数,目前已知的计算 $f_{\text{mult}}^{-1}(x,y)$的最快因子分解算法是超多项式时间算法。由素数的密度定理知,在小于 N 的正整数中,至少有 $N/\log_2 N$ 个素数,由此可得 f_{mult}至少是弱单向函数。

例 5 2　背包函数(参看例 4 .2)

背包函数也称子集和函数,定义为

$$f_{\text{ssum}}(a_1,\cdots,a_n,x_1,\cdots,x_n) = \sum_{i=1}^{n} x_i a_i = s。 \tag{5.7}$$

其中 $a_1,\cdots,a_n,s$ 表示正整数的二进数表示,它们满足条件 $|a_1| = \cdots = |a_n| = n;\ x_i \in \{0,1\},\ i = 1,\cdots,n$。

显然, $f_{\text{ssum}}(a_1,\cdots,a_n,x_1,\cdots,x_n)$是多项式时间可计算的。背包问题是一个 NP 完全类问题这个事实还不能证明 f_{ssum}是单向函数。这是因为解背包问题的算法的计算复杂性是由相同长的例子中最坏例子的计算时间定义的,而背包函数的求逆算法的成功概率是按均匀分布在相同长的例子中随机选取来定义的。另一方面,背包问题的许多特例(如某些特殊结构和低密度的例子)是易解的这一事实,也不能说明背包函数不是单向函数,这一点在定义 5 .2 后面的说明中也已提到。 f_{ssum}被认为是候选单向函数是基于至今尚未找到可解随机选取的高密度例子(即近似满足 $|a_1| = \cdots = |a_n| = n$ 的例子)的多项

式时间算法这一事实。

例 5 3　线性编码函数

记 F_2 为二元域，可描述为 $F_2=\{0,1\}$，其中定义了整数的模 2 加和乘。F_2^n 为 F_2 上的 n 维线性向量空间。F_2^n 的一个 k 维线性子空间 C 称为一个码率为 k/n 的线性码。$c=(c_1,\cdots,c_n)\in C$ 称为码字。F_2 上的一个 $k\times n$ 矩阵（k 行 n 列）矩阵 $\boldsymbol{G}$ 称为线性码 C 的生成阵，若 $C=\{m\boldsymbol{G};\ m\in F_2^k\}$（其中的运算为 F_2 中运算），即 k 长消息 $m=(m_1,\cdots,m_k)$ 可通过 $\boldsymbol{G}$ 编码为 C 中的 n 长码字（$k<n$）。若码字 $c=m\boldsymbol{G}$ 通过有噪音信道传送，收到向量为 $r=c+e$，其中 $e=(e_1,\cdots,e_n)\in F_2^n$ 称为错误，由 r 计算 m 称为译码。若 C 的最小距离为 $d=\min(w(c);\ c\neq 0,\ c\in C)$，其中 $w(c)$ 表示码字 c 中 1 的个数，则称 C 为 (n,k,d) 码。由编码理论知，一个 (n,k,d) 线性码可纠正所有 $w(e)\leq(d-1)/2$ 的错误 e，而且若存在 $\varepsilon>0$，使 $\frac{k}{n}<1-H_2\left(\frac{(1+\varepsilon)d}{n}\right)$，则几乎所有的 F_2 上 $k\times n$ 矩阵 $\boldsymbol{G}$ 都生成一个 (n,k,d) 码，其中

$$H_2(p)=\begin{cases}-p\log_2 p-(1-p)\log_2(1-p) & \text{若 } p<1/2\\ 1 & \text{其他}\end{cases}。$$

现在来定义线性编码函数。设常数 $k,\varepsilon,\delta>0$ 满足 $k<1-H_2((1+\varepsilon)\delta)$，则

$$f_{\text{code}}(\boldsymbol{G},m,e)=(\boldsymbol{G},m\boldsymbol{G}+e)=(\boldsymbol{G},r)\tag{5.8}$$

其中，$\boldsymbol{G}$ 为 F_2 上 $k\lfloor n\rfloor\times n$ 矩阵，m 为 $k\lfloor n\rfloor$ 长消息，e 为一个 $w(e)\leq\delta n/2$ 的错误。

显然，$f_{\text{code}}(\boldsymbol{G},m,e)$ 是多项式时间可计算的，但寻找随机线性码的多项式时间译码算法是编码理论中的一个著名公开问题，因此，说 f_{code} 为一候选单向函数是有理由的。

5.2　单向函数族

5.1 节定义的一般单向函数，其定义域是 $\{0,1\}^*$ 或它的一个无穷子集。这一形式描述简单，适用于作理论分析。为了描述可应用于密码系统的候选单向函数，我们要考虑单向函数族的形式，即把单向函数看作是由无限多个定义域为有限集的单向函数构成的一个无限族。

5.2.1　单向函数族的定义

定义 5.4　一个函数族是由 $\{0,1\}^*$ 的一个无穷子集 I（称为指标集）和每个指标 $i\in I$ 对应一个函数 $f_i(x):D_i\to\{0,1\}^*$ 构成，其中 D_i 为 $\{0,1\}^*$ 的一个有穷子集。

定义 5.5　一个函数族 $\{f_i:D_i\to\{0,1\}^*;\ i\in I\}$ 称为强单向函数族。若存在三个多项式时间概率算法 A,D,F 满足下列两个条件：

(1)　i 和 x 的抽取和 $f_i(x)$ 的计算是容易的，即若算法 A 的输入为 1^n（n 个 1 的比特

串),则其输出为 $I\cap\{0,1\}^n$ 上的一个随机变量 I_n,若算法 D 的输入为 $i\in I\cap\{0,1\}^n$,则其输出为 D_i 上的一个随机变量 X_n,若算法 F 的输入为 $i\in I$ 和 $x\in D_i$,则其输出为 $f_i(x)$。

(2) 计算 $f_i(x)$的逆 $f_i^{-1}(f_i(x))$是困难的,即对每一多项式时间概率算法 M',每一正多项式 $p(n)$和一切充分大的 n 有

$$\Pr\ M'(I_n, f_{I_n}(X_n))\in f_{I_n}^{-1}(X_n))\ <1/\ p(n)。\qquad (5.9)$$

其中 I_n 和 X_n 由条件(1)给出。

弱单向函数族的定义是类似的(参照定义 5.3)。

在定义 5.5 中没有要求随机变量 I_n 在 $I\cap\{0,1\}^n$ 上均匀分布,甚至不排除 $I_n=i$(集中于一个指标)。同样也不要求随机变量 X_n 在 D_i 上均匀分布,但条件(5.9)式蕴含 X_n 不集中在 D_i 的一个少于 $p'(n)$个 x 的子集上,其中 $p'(n)$为一正多项式。可是在公钥密码系统的应用中,通常要求 I_n 在 $I\cap\{0,1\}^n$ 上均匀分布, X_n 在 D_i 上均匀分布。换句话说,要求算法 A 和 D 是相应集上的随机(均匀)抽样算法,另外,在定义 5.5 的条件(1)中,因 D 是多项式时间概率算法,由定义 4.13 知,存在一个正多项式 $p(n)$使 $D[i]$(表示 D 的输入为 i)的运行时间 $K_r(i)\le p(|i|)$,其中 r 为 $D[i]$内部扔硬币结果(可设 $|r|\le p(|i|)$)。故 $D[i]$的输出 $X_{|i|}$ 的长 $|X_{|i|}|\le p(|i|)$。这就要求 f_i 的定义域 D_i $\subseteq\bigcup_{m\le p(|i|)}\{0,1\}^m$。因此不失一般性可设 D_i $\{0,1\}^{p'(|i|)}$, $p'(n)$为一正多项式。从算法 F 的定义可见,可设 F 为确定性算法。一个单向函数族也可记作(A, D, F)。

定理 5.1　任一单向函数 $f:\{0,1\}^*\to\{0,1\}^*$ 可表示为一个单向函数族,反之,任一单向函数族$\{f: D_i\to\{0,1\}^*;\ i\in I\}$也可表示为一单向函数 $f: E\to\{0,1\}^*$,其中 E 为 $\{0,1\}^*$ 的一个无穷子集。

证明　(1)若 $f:\{0,1\}^*\to\{0,1\}^*$ 为一单向函数,令指标集 $I=\{1^n; n\ge1\}$, f_{1^n}的定义域 $D_{1^n}=\{0,1\}^n$, $f_{1^n}(x)=f(x)$, $x\in D_{1^n}$。由定义 5.2 容易验证所定义的函数族 $\{f_i: D_i\to\{0,1\}^*;\ i\in I\}$为一单向函数族。

(2) 若$\{f_i: D_i\to\{0,1\}^*;\ i\in I\}$为一单向函数族,令 $E=\{y=(i, x);\ i\in I,\ x\in D_i\}$,显然 E 为$\{0,1\}^*$ 的一个无穷子集。定义 $f(y)=(i, f_i(x))$,若 $y=(i, x)\in E$,由定义 5.5 容易验证所定义的 $f: E\to\{0,1\}^*$ 为一单向函数。

5.2.2　候选单向函数族

本节要用到若干数论知识,不熟悉的读者可参看附录。本节中的正整数都看作它们的二进数表示,不再另加说明。

例 5.4　RSA 函数族。

RSA 函数族的指标集 $I_{RSA}=\{i=(N, e);\ N=PQ,\ P, Q$ 为素数,$2^{n-1}\le P<Q<2^n$, e 与 $(P-1)(Q-1)$ 互素$\}$。 $f_{(N,e)}$ 的定义域 $D_{(N,e)}=\{1,\cdots, N\}$, $f_{(N,e)}(x)=x^e\ (\mathrm{mod}\ N)$,

$x \in D_{(N,e)}$，$f_{(N,e)}(x)$的定义域也可限制在 $D_{(N,e)}$的子集 $D'_{(N,e)} = \{x;\ x$ 与 N 互素$\}$。这样，函数 $f_{(N,e)}(x)$就成为 $D'_{(N,e)}$上的一个置换。已经证明存在多项式时间算法(A, D, F)满足定义 5.5 中的条件(1)，即有多项式时间随机算法 A，输入 1^n，A 在 $[2^{n-1}, 2^n]$ 中近似随机(均匀)选取两个不同的素数 P，Q 和在与 $(P-1)(Q-1)$ 互素的正整数中近似随机(均匀)选取一个 e，输出(N, e)，其中 $N = PQ$。这一算法的具体实现详见本书参考文献[1]中的附录 1。有多项式时间概率算法 D，输入(N, e)，D 在 $D_{(N,e)}$(或 $D'_{(N,e)}$)中近似随机(均匀)选取一个 x 作为其输出；有多项式时间确定性算法 F，输入(N, e)和 x，F 输出函数值 $f_{(N,e)}(x)$。算法 D 和 F 的存在性和实现方法是显然的。其中 A 和 D 的随机抽样的近似性是由于随机算法采用内部扔硬币来实现。

RSA 函数族被认为是候选单向函数族是由于 $f_{(N,e)}(x)$求逆的困难性。经过长期研究，目前最好的求逆 $f_{(N,e)}(x)$的算法都要直接或间接因子分解 N。例 5.1 中已经说过，这是个困难问题。

例 5.5　Rabin 函数族。

Rabin 函数族与 RSA 函数族类似，其指标集 $I_{\text{Rabin}} = \{N;\ N = PQ,\ P, Q$ 为素数，$2^{n-1} \leq P < Q < 2^n\}$。$f_N$ 的定义域 $D_N = \{1, \cdots, N\}$或其子集 $D'_N = \{x;\ x$ 与 N 互素$\}$。$f_N(x) = x^2 \pmod N$，与 $f_{(N,e)}$不同，f_N 不是 D'_N 上的置换而是 4 对 1 映射。

可以证明，f_N 的求逆问题可通过多项式时间概率算法与 N 的因子分解问题相互转化，因此，Rabin 函数族被认为是候选单向函数。

例 5.6　Rabin-Blum 函数族。

Rabin-Blum函数族的指标集 $I_{\text{RB}} = \{N;\ N \in I_{\text{Rabin}},\ P \equiv Q \equiv 3 \pmod 4\}$。$f_N(x) = x^2 \pmod N$即为 Rabin 函数，不过，$f_N(x)$的定义域限制在 D'_N 的子集 $D''_N = \{x;\ x \in D'_N$，且 x 为模 N 的二次剩余$\}$。x 称为一个模 N 的二次剩余，若存在正整数 r 使 $x \equiv r^2 \pmod N$。由于可证明 D''_N 中的每个 x 只有惟一的平方根，也属于 D''_N，故 $f_N(x)$为 D''_N 上的一个置换。与 Rabin 函数族类似，Rabin-Blum 函数族也为一候选单向函数族。

例 5.7　离散对数函数族。

离散对数函数族是基于计算数论中的另一个困难问题，即在有限域(特别是元素个数为素数的有限域)上求离散对数的难解性。其指标集 $I_{\text{DLP}} = \{i = (p, g);\ p$ 为素数，$2^{n-1} \leq p < 2^n$，g 为 p 元有限域的本原元(阶为 $p-1)\}$。函数 $f_{(p,g)}$的定义域 $D_{(p,g)} = \{1, \cdots, p-1\}$。$f_{(p,g)}(x) = g^x \pmod p$。函数 $f_{(p,g)}(x)$是 $D_{(p,g)}$ 上的一个置换。$f_{(p,g)}(x)$的求逆问题，即在 p 元有限域上求离散对数问题。由于这一问题的困难性，故离散对数函数族也被认为是候选单向函数族。

5.3　单向函数族的其他性质

5.3.1　单向陷门置换族

要将单向函数族应用于公钥密码体制，它还需要有一个外加的性质。不严格地说，这个外加的性质就是，对每个 $i\in I$，存在一个 $\tau(i)\in\{0,1\}^*$，称为陷门，若已知 $\tau(i)$，则计算 $f_i(x)$ 的逆 x 是容易的。由 i 计算 $\tau(i)$ 是困难的，但有有效算法成对地选取 $(i,\tau(i))$。下面给出正式定义。

定义 5.6　设 $\{f_i: D_i\to D_i; i\in I\}$ 为一单向置换族。由定义 5.5 及其后的说明知，存在多项式时间概率算法 A_1 和 D 及多项式时间确定性算法 F 满足定义 5.5 中的条件(1)和(2)(参看定义 5.5)。单向置换族 $\{f_i: D_i\to D_i; i\in I\}$ 称为单向陷门置换族，若存在一个多项式时间概率算法 $A: 1^*\to\{0,1\}^*\times\{0,1\}^*$ 和一个多项式时间确定性算法 F^{-1} 满足如下条件，其中 $1^*=\{1^n; n=1,2,\cdots\}$。(3) $A(1^n)=(A_1(1^n))$，$\tau(A_1(1^n))=(I_n,\tau(I_n))$，其中 $A(1^n)$ 和 $A_1(1^n)$ 分别表示算法 A 和 A_1 输入 1^n 时的输出，I_n 为 $I\cap\{0,1\}^n$ 上的随机变量，且对每对 $(i,\tau(i))$，$i\in I$ 和每个 $x\in D_i$ 有 $F^{-1}(\tau(i), f_i(x))=x$，即陷门 $\tau(i)$ 作为求逆算法 F^{-1} 的辅助输入。

例 5.4′　考虑例 5.4 中的 RSA 单向置换族 $\{f_{(N,e)}: D'_{(N,e)}\to D'_{(N,e)}; (N,e)=i\in I_{RSA}\}$。可以证明，$\tau(i)=(N,d)$，其中 d 满足 $ed\equiv 1(\mathrm{mod}(P-1)(Q-1))$ 为其陷门(d 的存在性由 e 与 $(P-1)(Q-1)$ 互素推出)，$F^{-1}((N,d),y)=y^d(\mathrm{mod}\ N)$，$y\in D'_{(N,e)}$ 为 $y=f_{(N,e)}(x)$ 的求逆算法。因此对 RSA 置换族存在多项式时间概率算法 A 和多项式时间确定性算法 F^{-1} 满足定义 5.6 中的条件(3)。因此它是一个候选单向陷门置换族。

例 5.6′　考虑例 5.6 中的 Rabin-Blum 单向置换族 $\{f_N: D''_N\to D''_N; N\in I_{RB}\}$。

可以证明，$\tau(N)=(P,Q)$(N 的因子分解)为其陷门，$F^{-1}((P,Q),y)=c_P y^{(P+1)/4}+c_Q y^{(Q+1)/4}(\mathrm{mod}\ N)$，$y\in D''_N$ 为 $y=f_N(x)$ 的求逆算法，其中常数 c_P 和 c_Q 按中国剩余定理计算。因此对 Rabin-Blum 置换族存在多项式时间概率算法 A 和多项式时间确定性算法 F^{-1} 满足定义 5.6 中的条件(3)，因此它是一个候选单向陷门置换族。

5.3.2　单向无爪函数族

两个函数族 $\{f_i^0: D_i^0\to\{0,1\}^*, i\in I\}$ 和 $\{f_i^1: D_i^1\to\{0,1\}^*, i\in I\}$ 称为一个成对函数族，若 f_i^0 和 f_i^1 有相同的值域，即对任一 $i\in I$ 有 $\{y; y=f_i^0(x), x\in D_i^0\}=\{y; y=f_i^1(x), x\in D_i^1\}$。记函数 f_i^0 和 f_i^1 的公共值域为 R_i。一个成对函数族记作 $\{(f_i^0, f_i^1); f_i^\sigma: D_i^\sigma\to R_i, \sigma\in\{0,1\}, i\in I\}$。一个成对函数族 $\{(f_i^0, f_i^1); f_i^\sigma: D_i^\sigma\to R_i, \sigma\in\{0,1\}, i\in I\}$ 称为一个成对

单向函数族，若函数族$\{f_i^\sigma: D_i^\sigma \to R_i, i \in I\}$，$\sigma = 0,1$都是单向函数族。

定义 5.7 设$\{(f_i^0, f_i^1); f_i^\sigma: D_i^\sigma \to R_i, \sigma \in \{0,1\}, i \in I\}$为一成对单向函数族。由定义 5.5 及其后的说明知，对$\sigma = 0,1$都存在多项式时间概率算法A, D^σ及多项式时间确定性算法F^σ满足定义 5.5 中的条件(1)和(2)。一个成对单向函数族称为单向无爪(claw-free)函数族，若存在多项式时间算法(A, D, F)满足下列条件：

(1) $A(1^n)$为$I \cap \{0,1\}^n$上的随机变量I_n，$D(\sigma, i) = D^\sigma(i)$为$D_i^\sigma$上的随机变量$(i \in I, \sigma \in \{0,1\})$，$F(\sigma, i, x) = F^\sigma(i, x) = f_i^\sigma(x)$，$x \in D_i^\sigma$，$i \in I, \sigma \in \{0,1\}$；

(2) 对每个$i \in I$，随机变量$f_i^0(D(0, i))$和$f_i^1(D(1, i))$有相同的分布；

(3) (x_0, x_1)满足$f_i^0(x_0) = f_i^1(x_1)$称为对于指标i的一个爪(claw)，记C_i为对于指标i的一切爪构成的集，要求计算对于指标的爪是困难的，即对每一多项式时间概率算法M'，每一正多项式$p(n)$和一切充分大的n有

$$\Pr\{M'(I_n) \in C_{I_n}\} < 1/p(n)。 \tag{5.10}$$

在定义 5.7 中，条件(1)就是把两个单向函数族(A, D^0, F^0)和(A, D^1, F^1)结合成一个成对单向函数族(A, D, F)。条件(2)和(3)表示矛盾的两个方面：一方面，对于指标i的爪确实存在；另一方面，要计算出它们是计算上不可行的。

单向无爪函数族在密码中的应用是构造无碰撞杂凑函数(见第 11 章)。

例 5.7′ 考虑例 5.7 中的离散对数函数族，指标集$I'_{DLP} = \{i = (p, g, z); p$为素数，$2^{n-1} \leq p < 2^n$，$g$为$p$元有限域的本原元，$z \in \{1, \cdots, p-1\}\}$，$f^\sigma_{(p,g,z)}(x) = z^\sigma \cdot g^x \pmod p$，$x \in D^\sigma_{(p,g,z)} = \{1, \cdots, p-1\}$，$\sigma = 0,1$为两个离散对数置换族，它们有同样的定义域和值域，故它们构成一个成对单向置换族(A, D, F)。若要求算法D满足$D(0, (p, g, z))$和$D(1, (p, g, z))$都在$\{1, \cdots, p-1\}$上均匀分布，则定义 5.7 中的条件(2)满足。由于求离散对数问题是困难的，故定义 5.7 中的条件(3)也满足。因为若计算出一个对于(p, g, z)的爪(x_0, x_1)，由定义有$g^{x_0} \equiv z \cdot g^{x_1} \pmod p$，即有$g^{(x_0 - x_1)} \equiv z \pmod p$，这就等价于计算出了$z \pmod p$的以$g$为底的离散对数。因此$\{(f^0_{(p,g,z)}, f^1_{(p,g,z)}); (p, g, z) \in I'_{DLP}\}$为一单向无爪函数族。

5.4 单向函数的硬核

不严格地说，一个函数$f: \{0,1\}^* \to \{0,1\}^*$是单向的，若给了函数值$y = f(x)$，要计算它的一个原像$f^{-1}(y)$是困难的，这并不意味着要由$y$计算原像$f^{-1}(y)$的某些部分信息也是困难的。具体说，要恢复原像的一半比特也许是容易的。例如考虑单向函数$g(x, r) = (f(x), r)$，其中$f(x)$为单向函数，且$|r| = |x|$。单向函数不必隐藏原像的部分信息这个性质限制了它们直接应用于密码系统。幸好在单向函数存在的假定下，可

构造出单向函数,使它可隐藏原像的指定的部分信息,这个部分信息(由原像计算它是容易的)就被称为单向函数(求逆困难)的硬核(hard-core)。

5.4.1　单向函数的硬核谓词

定义 5.8　一个布尔函数 $b:\{0,1\}^* \to \{0,1\}$称为一个谓词(predicate)。一个多项式时间可计算谓词 b 称为一个函数 $f:\{0,1\}^* \to \{0,1\}^*$ 的硬核谓词(hard-core predicate),若对每一多项式时间概率算法 M',每个正多项式 $p(n)$和一切充分大的 n 有

$$\Pr\{M'(f(U_n)) = b(U_n)\} < \frac{1}{2} + \frac{1}{p(n)}。 \tag{5.11}$$

显然,对任意 b 和 f 总存在多项式时间概率算法 M'使(5.11)式中的成功概率至少为 1/2(用随机猜测算法,即无论算法输入什么,总是输出一次扔硬币结果作为 b 的猜测)。另一方面,若 b 是任意函数(如常数函数) f 的硬核谓词,则有 $|\Pr\{b(U_n)=0\} - \Pr\{b(U_n)=1\}|$为一可忽略函数。由于 b 本身是多项式时间可计算的,因此,若 $b(x)$和 $f(x)$使(5.11)式满足,则仅有两种情况:一是 $f(x)$不是 1-1 的(信息损失),例如 $b(\alpha\alpha)=\sigma$,是 $f(\alpha\alpha)=0\alpha$ 的硬核谓词,其中 $\sigma\in\{0,1\}$, $\alpha\in\{0,1\}^*$;另一情况是 $f(x)$是单向函数。事实上,可以证明,若 $b(x)$是 $f(x)$的硬核谓词,且 $f(x)$是 1-1 的,则 $f(x)$一定是单向函数。我们感兴趣的是第二种情况。

定义 5.9　一个多项式时间可计算布尔函数族$\{b_i: D_i \to \{0,1\};\ i\in I\}$称为函数族$\{f_i: D_i \to \{0,1\}^*;\ i\in I\}$的硬核谓词族,若对每一多项式时间概率算法 M',每一正多项式 $p(n)$和一切充分大的 n 有

$$\Pr\{M'(I_n, f_{I_n}(X_n)) = b_{I_n}(X_n)\} < \frac{1}{2} + \frac{1}{p(n)}, \tag{5.12}$$

其中 $I_n = A(1^n)$, $X_n = D(I_n)$, A, D 为函数族的随机抽样算法(参看定义 5.5)。

例 5.4″　考虑例 5.4 中的 RSA 单向置换族$\{f_{(N,e)}: D'_{(N,e)} \to D'_{(N,e)};\ (N, e) = i\in I_{\mathrm{RSA}}\}$。可以证明, $b_{(N,e)}(x) = x \pmod 2$(二进数 x 的最低位), $x\in D'_{(N,e)}$为其硬核谓词族。

例 5.6″　考虑例 5.6 中的 Rabin-Blum 单向置换族$\{f_{(N)}: D''_{(N)} \to D''_{(N)};\ N = i\in I_{\mathrm{RB}}\}$。可以证明, $b_N(x) = x \pmod 2$, $x\in D''_N$ 为其硬核谓词族。

例 5.7″　考虑例 5.7 中的离散对数单向置换族$\{f_{(p,g)}: D_{(p,g)} \to D_{(p,g)};\ (p, g) = i\in I_{\mathrm{DLP}}\}$。可以证明,

$$b_{(p,g)}(x) = \begin{cases} 1 & x > p/2 \\ 0 & x < p/2 \end{cases} \qquad x \in D_{(p,g)},$$

为其硬核谓词族。

对于一般单向函数有如下结果。

定理 5.2　设 $f(x)$为任一强单向函数。构造强单向函数 g 为 $g(x, r) = (f(x), r)$,

其中$|r|=|x|$。定义 $b(x,r)=\sum_j x_j r_j \pmod 2$,则谓词 b 为单向函数 g 的硬核谓词。

证明见本书参考文献[1]。

对于单向函数族有类似的定理。

5.4.2 单向函数的硬核函数

由定理 5.2 可见,每个单向函数可改变为另一单向函数,使它们有一个硬核谓词,换句话说,可以确定一位从函数值难于计算的原像信息。下面我们要将这一结果推广到多位信息的情形。我们采用一种稍微不同的方法,即要求该多位信息的真实值与随机值是计算难于区分的。不严格地说,一个多项式时间可计算函数 h 称为一个函数 f 的硬核函数,若任何有效(多项式时间)算法都不能区分$(f(x),h(x))$和$(f(x),r)$,其中 r 为长$|h(x)|$的随机比特串。关于计算不可区分的概念将在第 6 章中正式给出。

定义 5.10 设 $h:\{0,1\}^* \to \{0,1\}^*$ 是一个多项式时间可计算函数,满足 $h(x)=h(y)$,对一切$|y|=|x|$,记 $l(n)=|h(1^n)|$ $(l(n)\le n)$。h 称为 f 的硬核函数,若对每一多项式时间概率算法 M',每一正多项式 $p(n)$和一切充分大的 n 有

$$|\Pr\{M'(f(U_n),h(U_n))=1\}-\Pr\{M'(f(U_n),R_{l(n)})=1\}|<1/p(n), \tag{5.13}$$

其中,U_n 为在$\{0,1\}^n$ 上均匀分布的随机变量,$R_{l(n)}$为与 U_n 相互统计独立的在$\{0,1\}^{l(n)}$上均匀分布的随机变量。

可以证明,若在定义 5.10 中函数 h 为一谓词 b,即 $l(n)\equiv 1$,则定义 5.10 与硬核谓词的定义 5.8 等价。

关于函数族 $f_i:D_i\to\{0,1\}^*;\ i\in I$ 的硬核函数族 $h_i:D_i\to\{0,1\}^*;\ i\in I$ 的定义与定义 5.10 是类似的,读者不妨自己给出。

例 5.4″(续) 可以证明,$h_{(N,e)}(x)=x \pmod{\log_2 N\rfloor}$(二进数 x 的最低 $\log_2 N\rfloor$ 位),$x\in D_{(N,e)}$为 RSA 单向置换族的硬核函数族。

例 5.6″(续) 可以证明,$h_N(x)=x \pmod{\log_2 N\rfloor}$,$x\in D_N''$ 为 Rabin-Blum 单向置换族的硬核函数族。

例 5.7″(续) 可以证明,$h_{(p,g)}(x)=j$,若$(j-1)\left(\frac{p}{\log_2 p\rfloor}\right)<x<j\frac{p}{\log_2 p\rfloor}$ $(1\le j\le \log_2 p\rfloor)$,$x\in D_{(p,g)}$为离散对数单向置换族的硬核函数族。

对于一般单向函数,定理 5.2 有如下推广。

定理 5.3 设 $f(x)$为任一强单向函数,构造强单向函数 g_1 为 $g_1(x,r)=(f(x),r)$,其中,$|r|=|x|+l(|x|)-1$,$l(n)=\log_2 n\rfloor$。定义 $h(x,r)=b_1(x,r)b_2(x,r)\cdots b_{l(n)}(x,r)$,其中 $b_k(x,r)=\sum_j x_j r_{j+k-1}$ $(1\le k\le l(n))$,则函数 h 为单向函数 g_1 的硬核函数。

对于单向函数族也有类似的定理。

注 记

鉴于单向函数在现代密码学中的重要性，本书用一章专门介绍单向函数的理论(包括单向函数的严格定义及其性质)以及密码学中最常用的若干候选单向函数族。本章主要参考了文献[1]。限于篇幅，许多定理的证明被略去，有兴趣的读者可参看文献[1]。在过去出版的大多数密码学教材中，只对单向函数作了简单的直观的描述，如文献[26]，本书对单向函数作了较详细的介绍是为了使学生更多地了解现代密码学的基础，从而能更好地应用它。

习 题 五

1．证明若单向函数存在，则 P≠NP。

提示　对任一多项式时间可计算函数 $f:\{0,1\}^* \to \{0,1\}^*$，定义一个语言 $L_f \in \mathrm{NP}$(即 L_f 的成员识别问题∈NP)使得：若 $L_f \in \mathrm{P}$，则存在一个多项式时间算法计算 $f(x)$的逆($f^{-1}(x)$是多项式时间可计算的)。

2．设 $f:\{0,1\}^* \to \{0,1\}^*$ 为一单向函数，且对某一正整数函数 $l(n)$满足下列条件：

(1) $|f(x)| = l(|x|)$，对每个 x 成立；

(2) $l(n)$是 1-1 函数；

(3) $l(n) \geq n$，$n \geq 1$。

证明给了 $f(x)$，可在关于 $|x|$ 的多项式时间产生 $1^{|x|}$($|x|$个 1)。

提示　上述条件保证 $|x| \leq |f(x)|$ 且 $|x|$ 由 $|f(x)| = |f(1^{|x|})|$ 惟一确定。

3．设 f 为一强单向函数，证明对每个多项式时间概率图灵机 M 和每个正多项式 $p(n)$，密度函数 $|d_{M,p} \cap \{0,1\}^n| / 2^n$ 是可忽略的，即对每个正多项式 $q(n)$ 及一切充分大的 n 有 $|d_{M,p} \cap \{0,1\}^n| / 2^n < 1/q(n)$。

其中，定义 $d_{M,p}$ 为 $d_{M,p} = \{x; \Pr(M(f(x)) \in f^{-1}f(x))) \geq 1/p(|x|)\}$。

4．定义函数 $f_{\mathrm{add}}:\{0,1\}^* \to \{0,1\}^*$ 如下：

$$f_{\mathrm{add}}(xy) = x + y,\ |x| = |y|,$$

其中，x，y 为正整数的二进制表示，xy 为数串 x 和 y 的连接。证明 f_{add} 不是单向函数，也不是弱单向函数。

5．设 $f:\{0,1\}^* \to \{0,1\}^*$ 为一保长单向函数，证明 $g(x) = f(x)\ \ x$ 也是单向函数。

6．设 $f:\{0,1\}^* \to \{0,1\}^*$ 为一单向函数，则对每个正多项式 $p(n)$ 及一切充分大的 n 有

$$|\{ f(x);\ x \in \{0,1\}^n \}| > p(n)。$$

提示　用反证法。

7．详细验证定理 5.1 的证明，即验证定理 5.1 的(2)所给出的单向函数满足定义 5.2 的条件。

8．证明 Rabin 函数族 f_N 为单向函数族的充要条件是正整数 $N = PQ$（P，Q 为素数，$2^{n-1} \leq P < Q < 2^n$）的分解问题是困难的（参看例 5.5）。

9．设 $f:\{0,1\}^* \to \{0,1\}^*$ 为一 1-1 保长函数，$b:\{0,1\}^* \to \{0,1\}$ 为 f 的硬核谓词，证明若 f 是多项式时间可计算的，则 f 为一单向函数。

提示　用硬核谓词的求逆算法。

10．设 $f(x)$ 和 $g(x)$ 都是单向函数，问 $f(x)+g(x)$，$f(x)g(x)$，$f(g(x))$ 是否都是单向函数，指出哪个是，哪个不是，并给出直观说明。

第 6 章　伪随机序列生成器

在第 3 章已经证明,要构造在理论上完善保密的密码体制,所需要的随机密钥数不能少于所传送的明文消息数(如一次一密体制),但要产生很长的随机比特序列是要付出很高代价的(如在随机算法中的内部扔硬币),因此这种密码体制是不实用的。在实用的密码体制中,要求所用的随机密钥序列比所传送的明文消息序列短得多。要设计这样的密码体制,特别是简单的流密码体制,就要应用伪随机序列生成器了。不严格地说,一个伪随机序列生成器是一个确定性算法,它把短的随机比特序列(种子)延伸为长得多的貌似随机的比特序列。由此可见,伪随机序列生成器是密码学的一个基本工具,在构造实用的密码系统中起着重要的作用。伪随机序列生成器的研究和应用不只限于密码学,在概率计算(Monte Carlo)和通信等领域也有广泛的研究和应用,但很少给出确切的定义,一般都用直观的方法来描述,即要求伪随机序列生成器的输出序列满足随机序列所具有的某些统计性质或其输出序列能通过某种统计检验。用这种方法设计的伪随机序列生成器应用于密码学是有点冒险的,因为密码学应用面对的是敌方的密码分析者。设计者很难预料所遇到的密码分析者将采取怎样的破译或攻击策略,对付不同的策略要求伪随机序列满足不同的统计性质或通过不同的统计检验,假定设计者知道敌方所用的具体攻击策略是不合理的。

因此本章主要介绍用计算复杂性理论方法建立的伪随机序列生成器理论。这一方法的优点是可以给出伪随机序列生成器的确切定义。其主要思想是伪随机序列生成器的输出序列虽然不是真正的随机序列,但在计算资源受到一定限制的条件下,要区别这个输出序列与等长的真随机序列的不同是困难(不可行)的。用这一方法设计的伪随机序列生成器应用于密码学,即将其输出序列用作随机密钥,就能对付计算资源受到一定限制的敌方密码分析者可能采取的任何策略,故性能是稳健的。

关于经典的基于移位寄存器的伪随机序列生成器将在序列密码一章中介绍。

6.1　计算不可区分性

从前面的讨论可见,序列值的随机变量(即随机序列)族的计算不可区分性是本章的

基本概念。首先我们要将函数族的定义(参看定义 5 .4)推广到概率分布族的情形。

定义 6 .1 一个概率分布族是由$\{0,1\}^*$的一个无穷子集 I(称为指标集)和每个指标 $i\in I$ 对应一个概率分布 $p_i(x)$∶$D_i\rightarrow[0,1]$, $p_i(x)\geq 0$, $\sum_{x\in D_i} p_i(x)=1$ 构成,其中 D_i 为$\{0,1\}^*$的一个有穷子集。

随机变量族的定义与概率分布族的定义完全类似。只要将一个概率分布 $p_i(x)$对应一个在 D_i 上取值的随机变量 X_i,其分布为 $p_i(x)$,即 $\Pr\{X_i=x\}=p_i(x)$。因此,一个概率分布族对应一个随机变量族,反之亦然,故它们是等价的。下面我们用随机变量族代替概率分布族,叙述将更为方便。

定义 6 .2 两个随机变量族$\{X_i \mid x_i\in D_i, i\in I\}$和$\{Y_i \mid y_i\in E_i, i\in I\}$称为多项式时间不可区分,若对每个多项式时间概率算法 M',每个正多项式 $p(n)$和一切充分大的 n 有

$$\left|\Pr\{M'(X_i,i)=1\}-\Pr\{M'(Y_i,i)=1\}\right|<1/p(|i|)。\qquad (6.1)$$

(6 .1)式中的概率是对随机变量 X_i(或 Y_i)的一切可能取值和概率算法 M'的内部扔硬币的一切可能结果而取的。

一个重要的特殊情形是:指标集 $I=\{i=1^n; n=1,2,\cdots\}$,这时定义 6 .2 中的随机变量族$\{X_i; i\in I\}$和$\{Y_i; i\in I\}$就化为随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$。以下我们考虑随机变量序列的情形,由于定义不够直观,从它看不出两个多项式时间不可区分的随机变量序列在统计性质上有什么相似之处,因此我们首先对条件(6 .1)式在序列情形作些分析。为了方便起见,设 $D_{1^n}=E_{1^n}=\{0,1\}^n$,由(6 .1)式可见,对$\{0,1\}^*$上的每个多项式时间可计算的布尔函数 f∶$\{0,1\}^*\rightarrow\{0,1\}$(即 M'为计算 f 的算法)有

$$\left|\Pr\{f(X_n)=1\}-\Pr\{f(Y_n)=1\}\right|<1/p(n),\qquad (6.2)$$

对每个正多项式 $p(n)$和一切充分大的 n 成立。这就是说,(6 .2)式中的每个多项式时间可计算布尔函数 f,给出 X_n 和 Y_n 的一个统计性质相似性的要求。例如定义 $f(x)=1$ 当且仅当 x 中 0 的个数多于 1 的个数。显然 $f(x)$是多项式可计算的,代入(6 .2)式得

$$\left|\Pr\{w(X_n)<n/2\}-\Pr\{w(Y_n)<n/2\}\right|<1/p(n),$$

对每个正多项式 $p(n)$和一切充分大的 n 成立。其中 $w(x)$表示 x 中 1 的个数(x 的汉明重量)。

定义 6 .3 两个随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$的变差距离定义为

$$\Delta(n)=\frac{1}{2}\sum_{x\in\{0,1\}^n}\left|\Pr\{X_n=x\}-\Pr\{Y_n=x\}\right|, n\geq 1。\qquad (6.3)$$

若 $\Delta(n)$是可忽略的,即对每个正多项式 $p(n)$和一切充分大的 n 有 $\Delta n<1/p(n)$,则称$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是统计接近的。显然变差距离的定义 6 .3 可推广到 $D_{1^n}=E_{1^n}$的一般情形。

可以证明,若两个随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是统计接近的,则它们也是

多项式不可区分的,反之则不然,可以构造随机变量序列$\{X_n; n\geq 1\}$,它与均匀分布随机变量序列$\{U_n; n\geq 1\}$不是统计接近的,但它们是多项式时间不可区分的,因此,多项式时间不可区分对两个随机变量序列的统计性质所加的要求没有统计接近的要求高。

定义 6.4　称一个随机变量序列$\{X_n; n\geq 1\}$是伪随机的,若它与$\{0,1\}^{l(n)}$上的均匀分布随机变量序列$\{U_{l(n)}; n\geq 1\}$是多项式时间不可区分的,其中设 X_n 取值于集$\{0,1\}^{l(n)}$。

6.2　伪随机序列生成器的定义和性质

定义 6.5　一个伪随机序列生成器是一个确定性多项式时间算法,G 满足下列两个条件:

(1) 延伸性,存在一个正整数函数 $l(n) > n(n = 1,2,\cdots)$,使得对一切 $x\in\{0,1\}^*$ 有 $|G(x)| = l(|x|)$;

(2) 伪随机性,随机变量序列$\{G(U_n); n\geq 1\}$是伪随机的,即它与均匀分布随机变量序列$\{U_{l(n)}; n\geq 1\}$是多项式时间不可区分的。

生成器 G 的输入 x 称为它的种子,要求 G 将长 n 比特的种子延伸为 $l(n)$长比特的序列,且该序列与 $l(n)$长的随机比特序列是多项式时间不可区分的。$l(n)$称为 G 的延伸因子。

下面讨论伪随机序列生成器的性质。

(1) 统计性质

由定义 6.5 的(2)可知,G 的输出序列是伪随机序列。由(6.1)式后面的讨论知,它的统计性质与等长的真随机序列有许多相似之处。

例如,当 n 充分大时,$\Pr\left[w(G(U_n)) < \frac{l(n)}{2}\right]$ 近似等于 1/2。但伪随机序列与等长的真随机序列不是统计接近的,严格地说有下面的定理。

定理 6.1　若 G 是一个伪随机序列生成器,即满足定义 6.5 中的条件,则随机变量序列$\{G(U_n); n\geq 1\}$与$\{U_{l(n)}; n\geq 1\}$不是统计接近的。

证明　按推广的定义 6.3,它们的变差距离为

$$\Delta(l(n)) = \frac{1}{2}\sum_{x\in\{0,1\}^{l(n)}} \left|\Pr\{G(U_n) = x\} - \Pr\{U_{l(n)} = x\}\right|。$$

由于 G 是确定性算法,它将$\{0,1\}^n$ 中的每个比特串延伸为$\{0,1\}^{l(n)}$中的比特串。不妨假定得到$\{0,1\}^{l(n)}$中的 2^n 个不同的比特串。当 x 为这些比特串之一时,$\Pr\{G(U_n) = x\} = 2^{-n}$,而当 x 为其他比特串时 $\Pr\{G(U_n) = x\} = 0$。另一方面,$U_{l(n)}$是$\{0,1\}^{l(n)}$上的均匀分布随机变量,故对每个 $x\in\{0,1\}^{l(n)}$有 $\Pr\{U_{l(n)} = x\} = 2^{-l(n)}$,于是有

$$\Delta(l(n)) = \frac{1}{2}\sum_{x,\Pr\{G(U_n) = x\} > 0} \left|\Pr\{G(U_n) = x\} - \Pr\{U_{l(n)} = x\}\right|$$

$$+\frac{1}{2}\sum_{x,\Pr\{G(U_n)=x\}=0}\left|\Pr\{G(U_n)=x\}-\Pr\{U_{l(n)}=x\}\right|$$

$$=\frac{1}{2}2^{n}(2^{-n}-2^{-l(n)})+\frac{1}{2}(2^{l(n)}-2^{n})2^{-l(n)}$$

$$=\frac{1}{2}(1-2^{n-l(n)})+\frac{1}{2}(1-2^{n-l(n)})\geq(1-2^{n-(n+1)})=\frac{1}{2}。$$

上式中的不等式是用了 $l(n)\geq n+1$。因 $\Delta(l(n))$不是可忽略的,故$\{G(U_n);n\geq1\}$与$\{U_{l(n)};n\geq1\}$不是统计接近的。

(2) 多项式延伸性

若给了一个延伸因子为 $l_1(n)=n+1$ 的伪随机序列生成器 G_1。我们可用下面的方法构造一个延伸因子为 $l(n)=p(n)$的伪随机序列生成器 G,其中 $p(n)>n$ 为任一多项式。

构造方法 6.1 设 G_1 为一确定性多项式时间算法,它将每个 n 长比特串延伸为一个 $n+1$ 长比特串,$p(n)>n$ 为任一多项式。我们用 G_1 作 $p(n)$次迭代,即置 $s_0=s$,$|s|=n$ 为 G_1 的初始输入,计算 $G_1(s_{i-1})=x_is_i$,$i=1,2,\cdots,p(n)$,其中 $x_i\in\{0,1\}$,$|s_i|=|s_{i-1}|=n$,即 x_is_i 为输入 s_{i-1} 时 G_1 输出的长 $n+1$ 比特串。定义算法 G 为 $G(s)=x=x_1x_2\cdots x_{p(n)}$,它将一个 n 长比特串 s 延伸为一个 $p(n)$长比特串 x。由于 G_1 是确定性多项式时间算法,故 G 也是确定性的多项式时间算法。

定理 6.2 若 G_1,$p(n)>n$ 和 G 如上述构造方法所给,且 G_1 为伪随机序列生成器,则 G 也是伪随机序列生成器。

由于定理证明过长,这里略去,读者可参看本书参考文献[1]。

(3) 不可预测性

伪随机序列是多项式时间不可预测的。不严格地讲,就是用多项式时间算法从伪随机序列的前 k 个比特计算(预测)它的第 $k+1$ 比特,其正确概率至多为 1/2 加一个可忽略的量。直观上这个性质是显然的,因真随机序列是不可预测的。若伪随机序列是多项式时间可预测,则它就与真随机序列是多项式时间可区分,与伪随机序列的定义矛盾。理论上有下面的结论。

定义 6.6 随机变量序列$\{X_n;n\geq1\}$称为多项式时间不可预测的,若对每个多项式时间概率算法 M',每个正多项式 $p(n)$和一切充分大的 n 有

$$\Pr\{M'(1^{|X_n|},X_n)=\mathrm{next}_{M'}(X_n)\}<\frac{1}{2}+\frac{1}{p(n)},\qquad(6.4)$$

其中 $\mathrm{next}_{M'}(x)$表示 x 的第 $k+1$ 比特,若 M'输入$(1^{|x|},x)$后只读了 x 的前 k 个比特($k<|x|$),否则(即 M'读了全部 x),$\mathrm{next}_{M'}(x)$表示一个均匀分布的随机比特。

定理 6.3 一个随机变量序列$\{X_n;n\geq1\}$是伪随机的(参看定义 6.4)当且仅当它是多项式时间不可预测的。

伪随机序列的多项式时间不可预测性是它可应用于密码系统的关键性质。

(4) 单向函数性

可以证明,伪随机序列生成器的存在性与单向函数的存在性是等价的。换句话说,给了一个伪随机序列生成器,就可用它构造一个单向函数;反过来,给了一个单向函数,也可用它构造一个伪随机序列生成器。事实上,有下面的简单定理。

定理 6.4　设 G 为一延伸因子 $l(n)=2n$ 的伪随机序列生成器,若对每对 $x, y\in\{0,1\}^*$ 满足 $|x|=|y|$,定义函数 $f(x,y)=G(x)$,则 f 为一强单向函数。

在密码应用中更关心由单向函数构造伪随机序列生成器。这将在 6.3 节中讨论。

6.3　伪随机序列生成器的构造

6.3.1　用一般单向置换构造伪随机序列生成器

一个单向置换是一个 1-1 保长单向函数。由定理 6.2 及其前所述的构造方法知,为了构造一个延伸因子为任意多项式 $p(n)>n$ 的伪随机序列生成器,只需构造一个延伸因子为 $n+1$ 的伪随机序列生成器即可。

定理 6.5　设 $f:\{0,1\}^*\to\{0,1\}^*$ 为一 1-1 保长强单向函数,$b:\{0,1\}^*\to\{0,1\}$ 为函数 f 的硬核谓词(多项式时间可计算)。定义 $G(x)=f(x)\,b(x)$($f(x)$和 $b(x)$的连接),则 G 为一伪随机序列生成器。

证明　因 $f(x)$和 $b(x)$都是多项式时间可计算,故 $G(x)$也是多项式时间可计算。显然 $|G(x)|=|x|+1$,由定理 6.3 可知,为证 G 是伪随机序列生成器,只需证随机变量序列 $\{G(U_n)=f(U_n)\,b(U_n);\ n\geq 1\}$ 是多项式时间不可预测的,因 f 是 1-1 保长函数,故 $f(U_n)$是 $\{0,1\}^n$ 上的均匀分布随机变量,因此,从 $f(U_n)$的前 $k(k<n)$比特预测其 $k+1$比特的成功概率不超过 $1/2$。又因 b 是 f 的硬核谓词,由定义 5.8 知,由 $G(U_n)$的前 n 比特多项式时间预测其 $n+1$ 比特(即由 $f(U_n)$多项式时间预测 $b(U_n)$),其成功概率小于 $1/2$ 加一个可忽略的量,证得随机变量序列 $\{G(U_n);\ n\geq 1\}$ 是多项式时间不可预测的,定理证毕。

下一节将给出对密码应用更重要的用单向置换族构造伪随机序列生成器的方法。

6.3.2　用单向置换族构造伪随机序列生成器

设 A, D, F 是定义 5.5 中给出的定义一个单向置换族 $\{f_i: D_i\to D_i;\ i\in I\}$ 的三个多项式时间概率算法,且对每个 $i\in I$, $D(i)$为在 D_i 上均匀分布的随机变量。不妨设存在一个多项式 $q(n)$,使得算法 A 和 D 的内部扔硬币次数不超过 $q(n)$(对 n 长输入)。记 $A(1^n, r)\in\{0,1\}^n$ 为算法 A 输入为 1^n,内部扔硬币结果为 r 时的输出,类似地,$D(i, s)$为算法 D,输入为 i,内部扔硬币结果为 s 时的输出。不妨设 $r, s\in\{0,1\}^{q(n)}$,再记 $\{b_i:$

$D_i \to \{0,1\};\ i \in I\}$为单向置换族$\{f_i : D_i \to D_i;\ i \in I\}$的硬核谓词族(参看定义 5.9)。下面利用定理 6.2 前所述构造方法的思想及定理 6.5 给出一个用单向置换族构造伪随机序列生成器 G 的方法。

构造方法 6.2

(1) $r, s \in \{0,1\}^{q(n)}$作为种子输入到算法 G, G 用 r 及算法 A 确定指标 $i = A(1^n, r)$;

(2) G 用 s 及 i 由算法 D 确定初始输入 $s_0 = D(i, s)$(在 D_i 上均匀分布);

(3) 任给一多项式 $p(n) > 2q(n)$, G 用初始输入 s_0 及 i 由算法 F 计算 $s_j = F(i, s_{j-1}) = f_i(s_{j-1})$, $j = 1, 2, \cdots, p(n)$;

(4) G 计算 $x_j = b_i(s_{j-1})$, $j = 1, 2, \cdots, p(n)$;

(5) G 的输出为 $x = G(r, s) = x_1 x_2 \cdots x_{p(n)}$。

定理 6.6 设多项式时间概率算法 A, D, F 定义一单向置换族$\{f_i : D_i \to D_i;\ i \in I\}$, $\{b_i : D_i \to \{0,1\};\ i \in I\}$为该单向置换族的硬核谓词族, $q(n)$及 $p(n) > 2q(n)$, G 如构造方法 6.2 中所给。再设对每个 $i \in I$, $D(i)$为在 D_i 上均匀分布的随机变量,则 G 为一伪随机序列生成器,它将 $2q(n)$长的种子(r, s)延伸为 $p(n)$长的伪随机序列。

证明 由于构造方法 6.2 中的每一步计算都是确定性的和多项式时间的,故 G 为确定性多项式时间算法, G 的延伸性由假定 $p(n) > 2q(n)$给出, G 的伪随机性可由类似于定理 6.5 和定理 6.2 的证明方法证明。这里不作详述。

例 6.1 用例5.4中的 RSA 单向置换族$\{f_{(N,e)} : D'_{(N,e)} \to D'_{(N,e)};\ (N, e) = i \in I_{RSA}\}$及例 5.4′中的硬核谓词族$\{b_{(N,e)} : D'_{(N,e)} \to \{0,1\};\ (N, e) = i \in I_{RSA}\}$构造伪随机序列生成器 G_{RSA},可以完全按照构造方法 6.2 进行。G_{RSA}的输入(种子)是在随机选取素数对(P, Q)和与$(P-1)(Q-1)$互素的正整数 e时内部扔硬币的结果 r 以及随机选取 $D_{(N,e)}$中的 s_0 时内部扔硬币的结果 s, G_{RSA}的输出序列为 $x = x_1 x_2 \cdots x_j \cdots$,其中 $x_j = s_{j-1} \pmod 2$, $s_j = s_{j-1}^e \pmod N$ $(N = PQ)$, $j = 1, 2, \cdots$。

例 6.2 用例5.6中的 RB 单向置换族$\{f_N : D''_N; \to D''_N;\ N \in I_{RB}\}$及例 5.6′中的硬核谓词族$\{b_N : D''_N \to \{0,1\};\ N \in I_{RB}\}$构造伪随机序列生成器 G_{RB},构造方法与 G_{RSA}类似。G_{RB}的输入(种子)为在随机选取满足 $P \equiv Q \equiv 3 \pmod 4$的素数对$(P, Q)$时内部扔硬币的结果 r 以及随机选取 D''_N 中的 s_0 时内部扔硬币的结果 s。G_{RB}的输出序列为 $x = x_1 x_2 \cdots x_j \cdots$,其中 $x_j = s_{j-1} \pmod 2$, $s_j = s_{j-1}^2 \pmod N$ $(N = PQ)$, $j = 1, 2, \cdots$。

例 6.3 用例5.7中的离散对数单向置换族$\{f_{(p,g)} : D_{(p,g)} \to D_{(p,g)};\ (p, g) = i \in I_{DLP}\}$及例 5.7′中的硬核谓词族$\{b_{(p,g)} : D_{(p,g)} \to \{0,1\};\ (p, g) = i \in I_{DLP}\}$构造伪随机序列生成器 G_{DLP}, G_{DLP}的输入为在随机选取素数 p 及 p 元有限域的本原元 g 时内部扔硬币的结果 r 以及随机选取 $D_{(p,g)}$中的 s_0 时内部扔硬币的结果 s, G_{DLP}的输出序列为 $x = x_1 x_2 \cdots x_j \cdots$,其中 $x_j = 1$,若 $s_{j-1} > p/2$; $x_j = 0$,若 $s_{j-1} < p/2$; $s_j = g^{s_{j-1}} \pmod p$, $j = 1, 2, \cdots$。

6.4　用伪随机序列生成器构造伪随机函数

为了定义伪随机函数,首先要把前面在序列集中取值的随机变量序列推广到函数集中取值的随机变量(随机函数)组成的序列的情形。为了简单起见,我们只考虑一切保长函数组成的集 $\Theta_n=\{f:\{0,1\}^n\to\{0,1\}^n, n\geq 1\}$,其中有 2^{n2^n} 个不同的 n - 变量函数。

定义 6.7　一个随机函数序列 $\{F_n; n\geq 1\}$ 是一个在函数集 Θ_n 中取值的随机变量序列,其概率分布为 $\Pr\{F_n=f\}=p_n(f)$, $f\in\Theta_n$,即 $p_n(f)\geq 0$, $\sum_{f\in\Theta_n} p_n(f)=1$, $n=1,2,\cdots$,特别地,一个随机函数序列称为真随机函数序列,若其概率分布为 Θ_n 上的均匀分布,即对每个 $f\in\Theta_n$ 有 $p_n(f)=1/2^{n2^n}$,记真随机函数序列为 $\{H_n: n\geq 1\}$。

不严格地说,伪随机函数虽然不是真随机函数,但没有有效(多项式时间)算法能区分它与真随机函数的区别,即使算法在计算过程中,可根据需要来选择自变量并得到对应的函数值,其含意是算法(计算程序)在某些步(时刻)可向所区分的函数提问并获得回答(后面的提问可依赖于前面的回答),仍不能区分这些回答是来自伪随机函数还是真随机函数。

在计算复杂性理论中,引入神图灵机(Oracle Machine)的概念来描述上述意义下的算法,神图灵机是一种设想的计算模型,它是图灵机的推广,下面给出定义。

定义 6.8　一个确定性神图灵机是一个确定性图灵机(见定义 4.4)附加一条磁带(称为神带)和两个特殊状态 $s_{inv}, s_{ora}\in S$, s_{inv} 称为求神状态, s_{ora} 称为神现状态。当一个神图灵机 M 输入 x,存取函数 $f:\{0,1\}^*\to\{0,1\}^*$ 时,其计算也是一个形的有限或无限序列, $(s_0,t_0,i_0),(s_1,t_1,i_1),\cdots,(s_j,t_j,i_j),\cdots$,其关系由读写头所处状态的转移函数和读写头动作的指令函数确定,若 $s_j\neq s_{inv}$,则第 $j+1$ 个形如定义 4.4 所给,若第 j 个形中的状态 $s_j=s_{inv}$,且神带中的内容为 $q\in\{0,1\}^*$,则第 $j+1$ 个形中的状态 $s_{j+1}=s_{ora}$,且神带中的内容为 $f(q)$, q 称为 M 的提问, $f(q)$ 称为神的回答。神图灵机 M 的输出,记作 $M^f(x)$,以及运行(计算)时间如定义 4.4 所定义。

类似地可定义神概率图灵机(参看 4.2.2 节)。

定义 6.9　一个随机函数序列 $\{F_n; n\geq 1\}$ 称为伪随机函数序列。若对每个多项式时间神概率图灵机 M,每个正多项式 $p(n)$ 和一切充分大的 n 有

$$|\Pr\{M^{F_n}(1^n)=1\}-\Pr\{M^{H_n}(1^n)=1\}|<1/p(n), \tag{6.5}$$

其中假定输入为 1^n 时 M 的提问 q 都满足 $|q|=n$, $\{H_n; n\geq 1\}$ 表示真随机函数序列。

下面介绍一种用伪随机列数序列生成器构造一个伪随机函数序列的简单方法。

构造方法 6.3　设 G 为一个确定性算法,它将 n 长比特串 s 延伸为 $2n$ 长比特串 $G(s)$,定义函数 $G_0(s)$ 为 $G(s)$ 的前 n 个比特, $G_1(s)$ 为 $G(s)$ 的后 n 个比特(即 $G(s)=$

$G_0(s)\ G_1(s))$对每个 $s\in\{0,1\}^n$ 和每个 $x=(x_1, x_2, \cdots, x_n)\in\{0,1\}^n$，定义函数

$$f_s(x) = G_{x_n}(\cdots(G_{x_2}(G_{x_1}(s))\cdots)。 \tag{6.6}$$

定义随机函数 $F_n = f_{U_n}(x)$，其中 U_n 为$\{0,1\}^n$ 上的均匀分布随机变量(即 s 在$\{0,1\}^n$ 中按均匀分布随机抽取)，$\{F_n; n\geq 1\}$即为所构造的随机函数序列。

定理 6.7 设 G 和$\{F_n; n\geq 1\}$如构造方法 6.3 中所给，若 G 为一伪随机序列生成器，则$\{F_n; n\geq 1\}$为一伪随机函数序列。

定理证明较长，这里略去，读者可参看本书参考文献[1]。

这里要强调一点，用构造方法 6.3 构造的伪随机函数序列$\{F_n; n\geq 1\}$是可用有效多项式时间算法实现的，可在密码中应用。由于 F_n 的值仅在函数集$\{f_s; s\in\{0,1\}^n\}$中随机抽取，故可用一个多项式时间概率算法来选定 f_s(选定 s)，由于伪随机序列生成器 G 是多项式时间可计算的，故可用一个多项式时间算法输入 s 和 x，按公式(6.6)计算 $f_s(x)$。

伪随机函数在密码学中的应用是明显的，它与一个私钥分组密码体制很相似，即将 s 看作密钥，x 看作明文，$f_s(x)$看作密文，问题是 $f_s(x)$可能不是 1-1 的(一个密文对应多个明文)，且是否存在有效算法计算 $f_s(x)$的逆。6.5 节要讨论和解决这个问题，在这之前先举一例说明所构造的随机函数序列的应用。

例 6.4 某协会的会员共享一项秘密，故希望它的会员之间能互相识别，以免非会员混入窃密，其解决办法是让它的会员共享一个秘密函数 $f_s(x)$，$s\in\{0,1\}^n$，$f_s(x)$按伪随机函数序列$\{F_n; n\geq 1\}$的分布选取。任何非会员是不知道的。当会员甲想识别乙是否是会员，只要随机选一个 x，要求乙给出函数值 $f_s(x)$，若乙给出的函数值与甲计算的 $f_s(x)$相同，则认为乙是会员，否则认为乙不是会员。由于伪随机函数序列$\{F_n; n\geq 1\}$和真随机函数序列$\{H_n; n\geq 1\}$是计算不可区分的，因此这一方法是安全的。这是因为若乙不是会员，则乙不知道 $f_s(x)$，他只能抽取一个函数 f 来自真随机函数序列$\{H_n; n\geq 1\}$，给出函数值 $f(x)$，故乙被误认为是会员$(f(x) = f_s(x))$的概率是可以忽略的。

6.5 伪随机置换的构造

在 5.2.2 节中讨论候选单向函数族的例子时，已经遇到过置换的概念。事实上，置换就是一个定义域和值域相同的 1-1 函数。在密码体制中，由于要求明文与密文是一一对应的，故通常用置换作加密编码，本节介绍一个用伪随机函数序列构造伪随机置换序列的方法，它可应用于构造分组密码。

记$\prod_n = \{\pi: \{0,1\}^n \to \{0,1\}^n\}$，$n\geq 1$ 为$\{0,1\}^n$ 上的一切置换组成的集，其中有 $2^n!$ 个不同的置换。

定义 6.10 一个随机置换序列$\{P_n; n\geq 1\}$是一个在置换集$\prod_n$ 中取值的随机变量序

列,其概率分布为 $\Pr\{P_n=\pi\}=p_n(\pi)$, $\pi\in\prod_n$,即 $p_n(\pi)\geq 0$, $\sum_{\pi\in\prod_n} p_n(\pi)=1$, $n=1,2,\cdots$,特别地,一个随机置换序列称为真随机置换序列,若其概率分布为 $\prod_n$ 上的均匀分布,即对每个 $\pi\in\prod_n$ 有 $p_n(\pi)=1/2^n!$,记真随机置换序列为 $\{K_n; n\geq 1\}$。

伪随机置换序列的定义与定义 6.9 类似。

定义 6.11　一个随机置换序列 $\{P_n; n\geq 1\}$ 称为伪随机置换序列,若对每个多项式时间神概率图灵机 M,每个正多项式 $p(n)$ 和一切充分大的 n 有

$$\left|\Pr\{M^{P_n}(1^n)=1\}-\Pr\{M^{K_n}(1^n)=1\}\right|<1/p(n), \tag{6.7}$$

其中假定输入为 1^n 时 M 的提问 q 都是 n 长的,$\{K_n; n\geq 1\}$ 表示真随机置换序列。

构造方法 6.4　设 $f:\{0,1\}^n\to\{0,1\}^n$ 为 Θ_n 中任一函数,对每一点对 $(x,y)\in\{0,1\}^n\times\{0,1\}^n$,定义

$$\mathrm{DES}_f(x,y)=(y,x+f(y)), \tag{6.8}$$

式中"+"表示 F_2^n 中的向量加,同样地,对 t 个函数 $f_1,\cdots,f_t\in\Theta_n$,定义

$$\mathrm{DES}_{f_t,\cdots,f_1}(x,y)=\mathrm{DES}_{f_t,\cdots,f_2}(x,y)\mathrm{DES}_{f_1}(x,y))。 \tag{6.9}$$

设 F_n 为在 Θ_n 上取值的随机变量,$F_n^{(1)}$, $F_n^{(2)}$, $\cdots$, $F_n^{(t)}$ 为 t 个相互独立与 F_n 有相同分布的随机变量。设 $t(n)$ 为一正整数函数,定义随机函数序列 $\{\mathrm{DES}_{F_n}^{t(n)}; n\geq 1\}$,其中

$$\{\mathrm{DES}_{F_n}^{t(n)}=\mathrm{DES}_{F_n^{(t)},\cdots,F_n^{(1)}}(x,y)\}。 \tag{6.10}$$

定理6.8　设 F_n, $t(n)$, $\mathrm{DES}_{F_n}^{t(n)}$ 如构造方法 6.4 中所给。若随机函数序列 $\{F_n; n\geq 1\}$ 是多项式时间可实现(计算)的,且 $t(n)$ 是多项式时间可计算的,则随机函数序列 $\{\mathrm{DES}_{F_n}^{t(n)}; n\geq 1\}$ 也是多项式时间可实现的,且为多项式时间可逆的随机置换序列,更进一步,若 $\{F_n; n\geq 1\}$ 是一个伪随机函数序列,则 $\{\mathrm{DES}_{F_n}^{3}; n\geq 1\}$ 为一伪随机置换序列。

证明　容易直接证明 $\mathrm{DES}_f(x,y)$ 是 $\{0,1\}^{2n}$ 上的一个置换(1-1 函数)。作为习题读者可以自己证明。由于每个 $\mathrm{DES}_{F_n^{(j)}}$, $j=1,2,\cdots,t(n)$ 都是多项式时间可实现的,故由定义((6.9)式和(6.10)式)可见,$\mathrm{DES}_{F_n}^{t(n)}$ 是多项式时间可实现的。定义函数 $0(y)=0^n$ (n 个 0),对每个 $y\in\{0,1\}^n$,则有 $\mathrm{DES}_0(x,y)=(y,x)$。由此可得

$$\mathrm{DES}_{0,f,0}(\mathrm{DES}_f(x,y))=\mathrm{DES}_{0,f,0}(y,x+f(y))$$

$$=\mathrm{DES}_{0,f}(x+f(y),y))=\mathrm{DES}_0(y,x+f(y)+f(y)),$$

$$\mathrm{DES}_0(y,x)=(x,y)。$$

故 $\mathrm{DES}_f(x,y)$ 是多项式时间可逆的。因此 $\{\mathrm{DES}_{F_n}^{t(n)}; n\geq 1\}$ 为多项式时间可逆的随机置换序列。

这里略去定理 6.8 中第二部分(即 $\{\mathrm{DES}_{F_n}^{3}; n\geq 1\}$ 为伪随机置换序列)的证明。读者可参看本书参考文献[1]。

最后我们指出,绝大多数分组密码体制都是一个伪随机置换(有若干较简单的伪随机

置换复合而成),因此构造和分析具有好的密码学性质的多项式时间可实现的伪随机置换序列是密码学研究的一个重要问题。

注 记

本章介绍了用计算复杂性理论方法建立的伪随机序列生成器理论，这一理论在现代密码学中的重要性是显而易见的。本章主要参考了本书参考文献[1]，被省去的证明和更多的介绍可在文献[1]中找到。关于经典的基于移位寄存器的伪随机序列生成器理论可参看第7章，关于其他一些具体的伪随机序列生成器可参看本书参考文献[17]中J.C.Lagarias的论文《Pseudorandom number generators in cryptography and number theory》。关于本章介绍的伪随机序列生成器理论在现代密码学中的应用可参看本书参考文献[16]。

习 题 六

1.设在$\{0,1\}^n$中取值的随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是多项式时间不可区分的,证明对任一多项式时间概率算法A,随机变量序列$\{A(X_n); n\geq 1\}$和$\{A(Y_n); n\geq 1\}$也是多项式时间不可区分的。

2.设在$\{0,1\}^n$中取值的随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是统计接近的,$f:\{0,1\}^* \to \{0,1\}^*$为一函数,证明随机变量序列$\{f(X_n); n\geq 1\}$和$\{f(Y_n); n\geq 1\}$也是统计接近的。

3.若在$\{0,1\}^n$中取值的随机变量序列$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是统计接近的,则它们也是多项式时间不可区分的。

提示　证明$\{X_n; n\geq 1\}$和$\{Y_n; n\geq 1\}$是统计接近的充要条件是对任一集$S_n \subset \{0,1\}^n$有$\Delta(S_n) = |\Pr\{X_n \in S_n\} - \Pr\{Y_n \in S_n\}|$是$n$的可忽略函数。

4.设G是一个伪随机序列生成器,$h:\{0,1\}^n \to \{0,1\}^n$是一个多项式时间可计算置换$(n\geq 1)$,证明$G'(s) = G(h(s))$和$G''(s) = G'(h(s))$都是伪随机序列生成器。

5.阅读本书参考文献[1]中的定理3.3.3的证明后,自己证明定理6.2。

6.证明若一个随机变量序列$\{X_n; n\geq 1\}$是伪随机的(定义6.4),则它也是多项式时间不可预测的。

7.证明定理6.4。

8.计算例6.1中给出的伪随机序列生成器G_{RSA}的长100的输出序列x,设$N = PQ = 97\times 167$，$e = 797$，$s_0 = 75$。

第 7 章　序列密码

密码按加密形式分为序列密码和分组密码,序列密码是密码体制中一种重要的体制。序列密码又称流密码,它是将明文消息字符串逐位地加密成密文字符,以二元加法流密码为例:

设

$$m_0,\ m_1,\cdots,\ m_k,\cdots \quad \text{是明文字符;}$$

$$z_0,\ z_1,\cdots,\ z_k,\cdots \quad \text{是密钥流;}$$

则

$$c_k = m_k \quad z_k \quad \text{是加密变换;}$$

$$c_0,\ c_1,\cdots,\ c_k,\cdots \quad \text{是密文字符序列。}$$

容易看出,当密钥流 $z_0,\ z_1,\cdots,\ z_k,\cdots$ 是离散无记忆二元均匀分布的信源产生的,即完全随机时,则该体制就是“一次一密”体制,Shannon 曾证明它是不可破译的。“一次一密”密码在理论上不可破译的事实使人们感觉到,如果能以某种方式仿效“一次一密”密码,则将可以得到保密性很高的密码。

在通常的序列密码中,加、解密用的密钥序列是由密钥流生成器用确定性算法产生的,因而密钥流序列是伪随机序列。序列密码体制的安全强度取决于密钥流(或滚动密钥),因而什么样的伪随机序列是安全可靠的密钥流序列,以及如何实现这种序列就成了序列密码中研究的一个关键问题。

到了 20 世纪 50 年代,数字电子技术的发展,使得密钥流可以方便地利用以移位寄存器为基础的电路来产生,这促使线性和非线性移位寄存器理论迅速发展,加上有效的数学工具,如代数和谱分析理论的引入,使得流密码理论迅速走向成熟的阶段。同时由于它的加、解密容易实现,实时性好,以及没有或只有有限的错误传播,因此流密码在实际应用中,特别是在专用和机密机构中仍保持着优势。鉴于布尔函数是流密码实现的重要工具,本章首先简单介绍布尔函数的基本概念。

7.1　布尔函数

作为表示逻辑运算的函数,布尔函数是研究数字逻辑电路的重要数学工具,也是研究

以此为基础的一切科学技术的重要工具,从而也是研究密码学和密码技术的重要工具。无论是在流密码中还是分组密码中,无论是在私钥密码中还是公钥密码中,布尔函数都有重要的应用。尤其在流密码中,所使用的主要数学工具之一就是布尔函数。本节主要介绍布尔函数的表示和其密码学性质。

7.1.1 布尔函数的表示

布尔函数有多种不同的表示。

1.真值表

布尔函数由于其定义域和值域都是有限集,自然可以用列表法表示。

例如,表 7.1 定义一个二元布尔函数 $f(x)=f(x_1,x_2)$。

表 7.1 布尔函数的例子

x_1	x_2	$f(x)$
0	0	0
0	1	1
1	0	1
1	1	0

习惯上,总是按二进制表示之值递增顺序由上而下排列真值表。在此约定下,我们把右列函数值构成的矢量称为 $f(x)$的函数值向量。该向量的汉明重量称为 $f(x)$的重量,记为 $w(f)$或 w_f。若 $w(f)=2^{n-1}$,则称 $f(x)$是平衡函数。

列表法表示的实函数不一定有解析表达式。有趣的是,任何布尔函数都有解析表达式。

2.小项表示

对于 $x_i, c_i\in GF(2)$,规定 $x_i^1=x_i$, $x_i^0=\overline{x}_i$,于是

$$x_i^{c_i}=\begin{cases}1,\text{当 } x_i=c_i\\0,\text{当 } x_i\neq c_i\end{cases}。$$

设 $c=(c_i,\cdots,c_n)$, $x=(x_i,\cdots,x_n)$,则具有下述“正交性”:

$$x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}=\begin{cases}1,\text{当}(x_1,\cdots,x_n)=(c_1,\cdots,c_n)\\0,\text{当}(x_1,\cdots,x_n)\neq(c_1,\cdots,c_n)\end{cases}。$$

为了方便,今后亦记

$$x_1^{c_1}x_2^{c_2}\cdots x_n^{c_n}=x^c,$$

于是 $f(x)=\sum_{c=0}^{2^n-1}f(c)x^c$ 称为 $f(x)$的小项表示,其中的求和符号是指在 GF(2)上,每一个被加项 $f(c)x^c$ 称为一个小项。小项表示实际上是布尔代数表达方式,即逻辑表达方式。

此种表示法常用于布尔函数的设计实现。

例如,表 7 .1 所示的布尔函数的小项表示为

$$f(x) = 0 \cdot x_1^0 x_2^0 + 1 \cdot x_1^0 x_2^1 + 1 \cdot x_1^1 x_2^0 + 0 \cdot x_1^1 x_2^1$$
$$= x_1^0 x_2^1 + x_1^1 x_2^0 。$$

3 . 多项式表示

对于前面举例的小项表达式中,可将 $x_1^0 = x_1 + 1$, $x_2^0 = x_2 + 1$ 代入并化简得

$$f(x) = (x_1 + 1) x_2 + x_1 (x_2 + 1) = x_1 + x_2 。$$

这是 $f(x)$的多项式表达式。显然,任意 n 元函数 $f(x)$都可由小项表达式化为多项式表达式:

$$f(x) = a_0 + a_1 x_1 + \cdots + a_n x_n + a_{12} x_1 x_2 + \cdots$$
$$+ a_{n-1\,n} x_{n-1} x_n + \cdots + a_{12\cdots n} x_1 x_2 \cdots x_n ,$$

称上式为 $f(x)$的代数标准型。

一个乘积项 $x_{i_1} x_{i_2} \cdots x_{i_r}$ 的次数定义为 r,非零常数项的次数定义为 0,0 多项式的次数定义为 $-\infty$。布尔函数 $f(x)$的次数定义为 $f(x)$的代数标准型中具有非零系数的乘积项中的最大次数,记为 $\deg f$,当 $\deg f = 1$ 时,称 $f(x)$为线性布尔函数;当 $\deg f \geq 2$ 时,称 $f(x)$为非线性布尔函数。在电路设计中,实现一个线性函数的电路只需使用模 2 加法器,这是最简单的;而非线性函数还需要乘法器,相对复杂些。

4 . Walsh 谱表示

定义 7 .1　设 $x = (x_1, \cdots, x_n)$, $w = (w_1, \cdots, w_n) \in \mathrm{GF}(2)^n$, x 与 w 的点积定义为 $x \cdot w = x_1 w_1 + \cdots + x_n w_n \in \mathrm{GF}(2)$, n 元布尔函数 $f(x)$的 Walsh 变换定义为 $S_f(w) = 2^{-n} \sum_{x=0}^{2^n-1} (-1)^{w \cdot x} f(x)$,其逆变换为 $f(x) = \sum_{w=0}^{2^n-1} S_f(w) (-1)^{w \cdot x}$。 $S_f(w)\ (w \in \mathrm{GF}(2)^n)$称为 $f(x)$的第一种谱或 Walsh 谱。

定义 7 .2　定义 $S_{(f)}(w) = 2^{-n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{w \cdot x}$为 $f(x)$的第二种谱或循环谱。

定理 7 .1　$S_{(f)}(w)$与 $S_f(w)$关系如下:

$$S_{(f)}(w) = \begin{cases} -2 S_f(w), & w \neq 0 \\ 1 - 2 S_f(w), & w = 0 \end{cases} 。$$

证明略。

Walsh 变换是可逆的,所以布尔函数的 Walsh 谱和循环谱都是惟一的。

若将 $f(x)$的 Walsh 谱按字典序从小到大排列,便得一实向量:

$$(S_f(0), \cdots, S_f(2^n - 1))。$$

于是,可将布尔函数的某些问题转化为实向量来研究。

从 $S_{(f)}(w)$的定义出发,容易推导出如下结果:

定理 7.2 设 $x=(x_1,\cdots,x_n)$, $w=(w_1,\cdots,w_n)\in GF(2)^n$, $f(x)$是 n 元布尔函数, 则

$$P(f(x)=w\cdot x)=\frac{1+S_{(f)}(w)}{2},$$

$$P(f(x)\neq w\cdot x)=\frac{1-S_{(f)}(w)}{2},$$

这里 $P(\cdot)$表示概率。

此定理说明布尔函数与线性函数的符合率(即相等的概率)可用其循环谱刻画。

7.1.2 布尔函数的非线性

代数次数大于 1 的函数是非线性函数, 但同是非线性函数, 其在实用上差异很大。如 $f_1(x)=x_1+x_2+x_3x_4$ 和 $f_2(x)=x_1x_2+x_3x_4$ 都是 2 次非线性函数, 但 $f_1(x)$是部分线性的, 而 $f_2(x)$是非退化的, $f_1(x)$比 $f_2(x)$更接近于线性函数, 或者说更容易用线性函数来逼近。于是人们引入“非线性度”的概念来刻画一个函数的非线性程度。

定义 7.3 设 $f(x)$是一个 n 元布尔函数, 记 $L_n[x]$为所有 n 元线性函数(包括仿射函数)之集。$f(x)$的非线性度定义为

$$\min_{l\in L_n[x]} d(f,l)=\min_{l\in L_n[x]} w(f+l),$$

记为 N_f, 即 $f(x)$的非线性度为其与所有线性函数之最短距离, 于是线性函数的非线性度为 0。称 $\max_{l\in L_n[x]} d(f,l)$为 $f(x)$的线性度, 记为 C_f, 即线性度是 $f(x)$与所有线性函数的最大距离。

显然, 对任意 n 元布尔函数 $f(x)$, 有 $N_f+C_f=2^n$。

定义 7.4 若 $l(x)$使得 $d(f,l)=N_f$, 则称 $l(x)$为 $f(x)$的最佳线性逼近。

显然, $l(x)$为 $f(x)$的最佳线性逼近, 意味着 $f(x)$与 $l(x)$符合率最高。

定理 7.3 任意 n 元布尔函数 $f(x)$的非线性度满足 $N_f\leq 2^{n-1}-2^{\frac{n}{2}-1}$, 使等式成立(即非线性度最高)的函数定义为 Bent 函数。

证明略。

Bent 函数是非线性度最大的函数, 还有几类函数非线性度虽未达到最大, 但实用上却更有价值。

定义 7.5 若对任意 $c=(c_1,\cdots,c_n)\in GF(2)^n$, $w(c)=1$, 有 $w(f(x)+f(x+c))=2^{n-1}$, 即 $f(x)+f(x+c)$是平衡函数, 则称 $f(x)$满足严格雪崩准则。若将 $f(x)$的任意 k 个分量固定为常数, 得到的 $n-k$ 元函数均满足严格雪崩准则, 则称 $f(x)$满足 $k(0\leq k\leq n-2)$阶雪崩准则。严格雪崩准则记为 SAC, k 阶雪崩准则记为 SAC(k)。满足严格雪崩准则的函数称为 SAC 函数。

定义 7.6 设 $\alpha\in GF(2)^n$, $\alpha\neq 0$, 若 $f(x)+f(x+\alpha)$是平衡函数, 即 $w(f(x)+f(x$

$+\alpha)) = 2^{n-1}$,则称 $f(x)$关于 α满足扩散准则。若对任意满足 $1 \leq w(\alpha) \leq k$ 的 α, $f(x)$关于 α满足扩散准则,则称 $f(x)$满足 k 次扩散准则。

7.1.3　布尔函数的相关免疫性

相关免疫函数是为防止攻击者对密码系统进行相关攻击而提出的。首先由 Siegenthaler 给出定义。

定义 7.7　设 $z = f(x_1, \cdots, x_n)$是 n 个彼此独立,均匀分布的二元随机变量的布尔函数,称 $f(x)$是 m 阶相关免疫的,当且仅当 z 与 $x_1, \cdots, x_n$ 中的任意 m 个随机变量 $x_{i_1}, \cdots, x_{i_m}$统计独立,或者,当且仅当互信息 $I(z; x_{i_1}, \cdots, x_{i_m}) = 0$,对任一组 $x_{i_1}, \cdots, x_{i_m}$, $1 \leq i_1 < \cdots < i_m \leq n$ 成立。

当 $m = 1$ 时,称 $f(x)$是 1 阶相关免疫函数,或一般地称为相关免疫函数;当 $m \geq 2$ 时,亦称 $f(x)$为高阶相关免疫函数。

一个函数 $f(x)$是相关免疫的,也说 $f(x)$具有相关免疫性,或说 $f(x)$满足相关免疫准则。

相关免疫函数的概念一经提出,便倍受重视。人们围绕着这一概念讨论了一系列问题,比如相关免疫函数的特征刻划、存在的充要条件、构造方法、计数问题、密码学价值等等,限于篇幅,在此不作介绍,若想了解详细情况请参看本书参考文献[21]。

7.1.4　布尔函数不同性质之间的关系

前面我们介绍了布尔函数的一些基本性质,这些性质大都是在不同的应用背景下提出来的。一种性质表示了函数在某一应用中的性能,其量化便是这种性能的衡量指标,如非线性是密码系统中为抵抗线性攻击而提出的性能,非线性度则是衡量其非线性性能强弱的指标。若从这个意义上讲,非线性度越高越好,但非线性度达到最高的函数,其他性能将变弱。如当非线性度达到最高时,将失去相关免疫性。因此,研究不同性质之间的关系,特别是不同性能指标之间的数量关系是布尔函数研究中的一个重要课题。

7.1.5　多输出布尔函数

对于 $GF(2)^n$ 到 $GF(2)^m$ 的映射,我们也可用布尔函数来描述。设 F 是 $GF(2)^n$ 到 $GF(2)^m$ 的映射,即

$$F: (x_1, \cdots, x_n) \to (y_1, \cdots, y_m), (x_1, \cdots, x_n) \in GF(2)^n, (y_1, \cdots, y_m) \in GF(2)^m。$$

每个 y_i 都是 $x = (x_1, \cdots, x_n)$的函数,记 $y_i = f_i(x) = f_i(x_1, \cdots, x_n)$,于是上述映射便可用 m 个布尔函数 $f_1, \cdots, f_m$ 来表示,记为 $F_n^m(x) = (f_1(x), \cdots, f_m(x))$,称 $F_n^m(x)$为多输出布尔函数。

与单输出函数不同,多输出函数描述的是 n 个输入、m 个输出的机器(也称变换盒),

如 DES 中的 S 盒就是 6 个输入、4 个输出的多输出函数。变换盒是分组密码体制的核心部分,因此为了提高系统的安全性,所使用的多输出函数必须满足一定的准则,从而研究多输出函数及其性质便成为密码学,特别是分组密码设计与分析中一个重要的研究内容。类似于单输出布尔函数,多输出布尔函数也有相应的性质。本小节简要介绍多输出函数的若干重要性质。

定义 7.8 设 $F(x)=(f_1(x),\cdots,f_m(x))$ 是 $GF(2)^n$ 到 $GF(2)^m$ 的多输出布尔函数,令

$$D(F)=\min\{\deg(\beta\cdot F)\mid \beta\neq 0,\beta\in GF(2)^m\}$$

$$=\min\{\deg\sum_{i=1}^{m}b_if_i(x)\mid (b_1,\cdots,b_m)=\beta\neq 0,(b_1,\cdots,b_m)\in GF(2)^m\},$$

则称 $D(F)$ 为 $F(x)$ 的代数次数。这里 $f_i(x)(1\leq i\leq m)$ 是 n 元布尔函数,$\deg(\cdot)$ 表示布尔函数的代数次数。当 $D(F)=k$ 时,称 $F(x)$ 为 k 次函数。

应当指出,从各分量函数的所有非零线性组合角度来定义多输出函数的代数次数比通常仅从各分量函数来定义代数次数的方法更具有一般性和合理性。

定义 7.9 设 $F(x)=(f_1(x),\cdots,f_m(x))$,$m\leq n$,若对任意 $\beta\in GF(2)^m$,集合

$$\{\alpha\mid \alpha\in GF(2)^n,F(\alpha)=\beta\}$$

中恰好有 2^{n-m} 个元素,则称 $F(x)$ 是正交的。

定义 7.10 设 $F(x)=(f_1(x),\cdots,f_m(x))$,称

$$\max_{\alpha\in GF(2)^n,\alpha\neq 0}\max_{\beta\in GF(2)^m}|\{x\mid F(x)+F(x+\alpha)=\beta\}|$$

为 $F(x)$ 的差分均匀性,记为 $\delta(F)$ 或 δ_F,称 $F(x)$ 为差分 δ_F 均匀的。

显然,$\delta_F\geq 2^{n-m}(m\leq n)$。当 $\delta_F=2^{n-m}$ 时,对任意 α,$F(x)+F(x+\alpha)$ 是正交的,此类函数是分组密码设计中非常重要的函数。

定义 7.11 设 $F(x)=(f_1(x),\cdots,f_m(x))$,如果对任意 $\alpha\in GF(2)^n$,$w(\alpha)=1$ 时,$f_i(x+\alpha)+f_i(x)(1\leq i\leq m)$ 是平衡的,则称 $F(x)$ 满足严格雪崩准则(SAC)。称 $F(x)$ 是 SAC 函数。

雪崩准则被称为传播准则,历史上,是 Feistel 最先提出“雪崩”定义且认识到它在分组密码设计中的作用。

定义 7.12 设 $F(x)=(f_1(x),\cdots,f_m(x))$,$m\leq n$,若 $\delta_F=2^{n-m}$,则称 $F(x)$ 是完全非线性函数。

定义 7.13 设 $F(x)=(f_1(x),\cdots,f_m(x))$,$m\leq n$,称

$$N_F=\min_{0\neq u\in GF(2)^m,l(x)\in L_n[x]}d(u\cdot F(x),l(x))$$

为 $F(x)$ 的非线性度。这里 $L_n[x]$ 表示全体线性函数(包括仿射函数)之集。

定理 7.4 设 $F(x)=(f_1(x),\cdots,f_m(x))$,$m\leq n$,则 $N_F\leq 2^{n-1}-2^{\frac{n}{2}-1}$。

证明略。

定理 7 5　$F(x)$是完全非线性函数，当且仅当 $N_F = 2^{n-1} - 2^{\frac{n}{2}-1}$。

证明略。

研究表明，S-盒的布尔函数具有高非线性度对“混乱”有积极的影响，具有好的雪崩特性对“扩散”有积极影响。它们有助于增强分组密码抗击各种攻击，包括差分和线性攻击的力量。

7 2　序列密码的原理

一个保密的通信系统的模型如图 7 .1 所示。

信源可以是报文、语言、图像、数据等，一般都是经编码器转化为 0，1 序列，加密是针对 0，1 序列进行。因此，可将信源看成是 0，1 序列，而将图 7 .1 简化为图 7 .2。

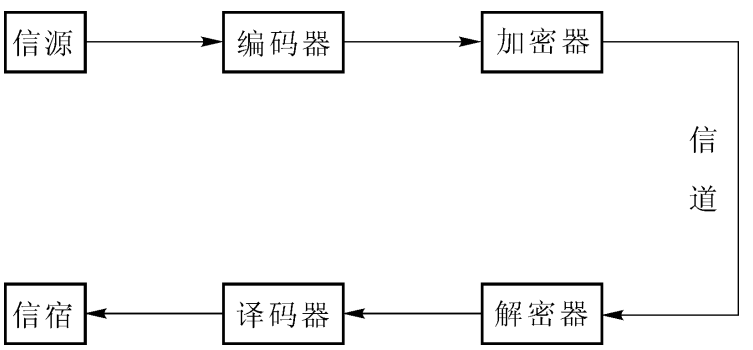


图 7 .1　保密通信系统模型

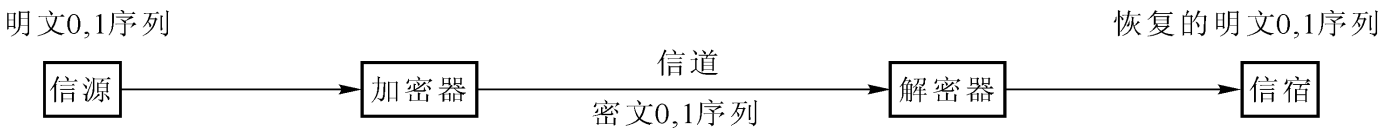


图 7 .2　简化的保密通信系统模型

序列密码将明文消息序列 $m = m_1, m_2, \cdots$，用密钥流序列 $k = k_1, k_2, \cdots$，逐位加密，得密文序列 $c = c_1, c_2, \cdots$，其中加密变换为 E_k：

$$c_i = E_k(m_i),$$

记为 $c = E_k(m)$，其解密变换为 D_k：

$$m_i = D_k(c_i),$$

记为 $m = D_k(c)$。

在序列密码中，加密变换常采用二元加法运算，即

$$c_i = m_i \quad k_i, \qquad m_i = c_i \quad k_i。$$

图 7 .3 是一个二元加法流密码系统的模型。其中 k_I 为密钥序列生成器的初始密钥或称种子密钥。为了密钥管理的方便，k_I 一般较短，它的作用是控制密钥序列生成器生

成长的密钥流序列 $k = k_1, k_2, \cdots$。

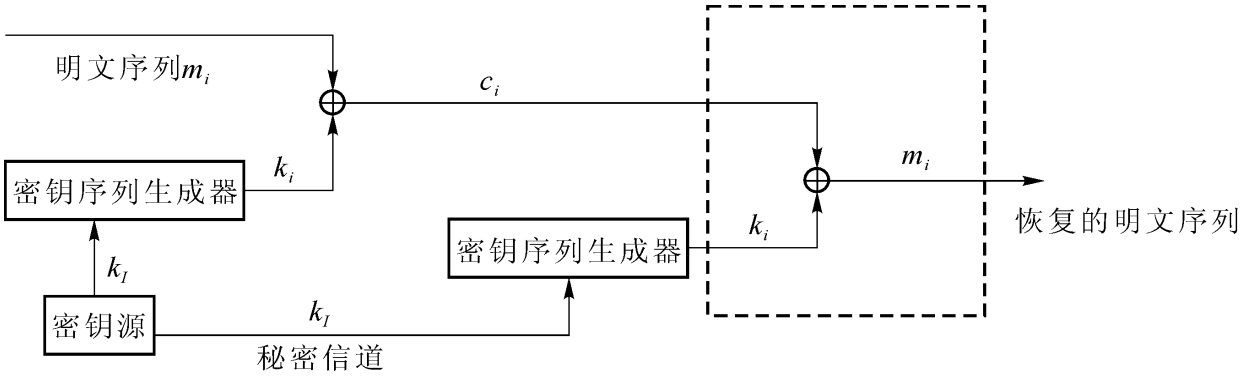


图 7.3　二元加法流密码系统模型

恢复明文的关键是知道 k_i。如果非法接收者知道了 k_i，当然也就知道了 m_i，因此密码系统的安全性取决于密钥流的性能。当密钥流序列是完全随机序列时，该系统便被称为完善保密系统，即不可破的。然而，在通常的序列密码中，加、解密用的密钥序列是伪随机序列。

7.3　序列的伪随机性

定义 7.14　序列$\{a_i\}$称为周期序列，若存在正整数 T，使得

$$a_{i+T} = a_i, \qquad i = 0, 1, 2, \cdots \tag{7.1}$$

满足(7.1)式的最小正整数 T 称为序列$\{a_i\}$的周期。若存在 n_0，使得 $a_{n_0}, a_{n_0+1}, \cdots$ 是周期序列，则称$\{a_i\}$是终归周期的。

称两个周期序列为不同的，若其中一个不能由另一个经适当移位而得到。

定义 7.15　序列$\{a_i\}$的一个周期中，若

$$a_{t-1} \neq a_t = a_{t+1} = \cdots = a_{t+l-1} \neq a_{t+l},$$

则称$(a_t, a_{t+1}, \cdots, a_{t+l-1})$为序列的一个长为 l 的游程。

定义 7.16　GF(2)上周期为 T 的序列$\{a_i\}$的自相关函数定义为

$$R_a(\tau) = \frac{1}{T}\sum_{k=0}^{T-1}(-1)^{a_k}(-1)^{a_{k+\tau}}, 0 \leq \tau \leq T-1。$$

周期序列的自相关函数表示序列$\{a_i\}$与$\{a_{i+\tau}\}$在一个周期内对应位相同的位数与对应位相异的位数之差的一个参数，因而它是序列随机性的一个指标。对独立均匀分布的二元随机变量序列$\{X_i\}$，它的自相关函数 $R_x(\tau)$的期望值为

$$E[R_x(\tau)] = \begin{cases} 0, \text{当 } \tau \neq 0 \text{ 时} \\ 1, \text{当 } \tau = 0 \text{ 时} \end{cases}。$$

为了度量周期序列的随机性，Golomb 对序列的随机性提出下述三条假设，即 Golomb

随机性假设:

(1) 在序列的一个周期内,0 与 1 的个数相差至多为 1;

(2) 在序列的一个周期圈内,长为 1 的游程数占总游程数的 1/ 2,长为 2 的游程数占总游程数的 1/ 2^2,…,长为 i 的游程数占总游程数的 1/ 2^i,…,且在等长的游程中,0,1 游程各占一半;

(3) 自相关函数为二值。

满足上述三个条件的序列称为伪随机序列。条件(1)说明 0-1 序列$\{a_i\}$中 0 与 1 出现的概率"基本"相同;条件(2)说明在已知位置 n 前若干位置上的值的条件下 0 与 1 在第 n 位置上出现的概率是相同的;条件(3)表明若将$\{a_i\}$与$\{a_{i+\tau}\}$比较,无法得到关于$\{a_i\}$的实质性信息(例如周期)。伪随机序列的应用很广,如通信中同步序列、码分多址(CDMA)等。以后我们会看到,满足上述三条性质的周期序列并不能满足我们对密钥流序列在安全性方面的要求。一个二元随机序列 $a_0, a_1, a_2, \cdots$可视为一个二元对称信源(BSS)的输出(也可用掷硬币方式产生),它的随机性包含着当前输出位 a_n 对以前输出段 $a_0, a_1, a_2, \cdots, a_{n-1}$的完全独立性,即 $H(a_n \mid a_0, a_1, \cdots, a_{n-1}) = H(a_n)$,对一切 $n \geq 1$。故在已知 $a_0, a_1, \cdots, a_{n-1}$的条件下,$a_n$ 是不可预测的。然而,要移植关于 BSS 的不可预测性这一概念并不是一件容易的事情,因为 BSS 提供的无限长随机序列与我们用作密钥流的周期序列之间存在着本质的差别。一种度量有限长或周期序列的随机性的方法是由 Lempel 和 Ziv 建议的所谓"线性复杂度"方法,规定用产生该序列的最短线性反馈移位寄存器(LFSR)的长度来度量,这种方法实际上衡量了序列的线性不可预测性。

7.4　序列密码对密钥流的要求

没有绝对不可破的密码,比如穷搜索总能破译,只是破译所需时间是否超过信息的有效期以及破译所需代价是否值得。因此密码能否被破译只是相对的,而不是绝对的。所以一般说某个密码系统是安全的,实际上都只是相对意义上的安全,称之为计算安全性。一个破译密码的算法,若计算量大于或等于穷搜索,则不会被视为一个破译方法。若一个加密算法没有比穷搜索更好的破译方法,则被认为是不可破的。一个密钥流序列若是完全随机的,则没有比穷搜索更好的方法破译它,这就是前边将其称为完善保密系统的理由。关于密码系统安全性的详细叙述参见第 3 章。

实际使用的密钥流序列(以下简称密钥)都是按一定算法生成的,因而不可能是完全随机的,所以也就不可能是完善保密系统。为了尽可能提高系统的安全强度,就必须要求所产生的密钥流序列尽可能具有随机序列的某些特征。一般地,序列密码中对密钥流有如下要求:

(1) 极大的周期。因为随机序列是非周期的,而按任何算法产生的序列都是周期的

或终归周期的,因此,应要求其有尽可能大的周期。现代密码机的数据率高达每秒 10^8 比特,如果 10 年内不使用周期重复的$\{k_i\}$,则要求$\{k_i\}$的周期不少于 3×10^{16}或 2^{55}。

(2) 良好的统计特性,即满足或部分满足 Golomb 的三个随机性假设。

(3) 不能用级数较小的(可实现长度)线性移位寄存器近似代替,即有很高的线性复杂度。

(4) 用统计方法由密钥序列$\{k_i\}$提取密钥生成器结构或密钥源的足够信息在计算上是不可能的。

这些要求对于保证序列密码的安全性是必需的。因为按任何确定性算法产生的序列都是周期或终归周期的。在高速率的现代通信中,若密钥周期 P 很短,则从下面两组密文 $m_1+k_1,\cdots,m_L+k_L$ 和 $m_{P+1}+k_{P+1},\cdots m_{P+L}+k_{P+L}$相加得 $m_1+m_{P+1},\cdots,m_L+m_{P+L}$以及语音冗余度就可获得一些关于明文的信息,因而周期长是必要的;良好的随机统计特性是为了更好地掩盖明文。高线性复杂度防止从部分密钥序列通过线性关系简单推出整个密钥序列。若知道一些明密文对$(m_i,c_i),(m_{i+1},c_{i+1}),\cdots$,便可简单地确定部分密钥序列 $k_i,k_{i+1},\cdots$,因此,安全的密码系统应能抵抗从部分密钥 $k_i,k_{i+1},\cdots$确定整个密钥序列的攻击。

以上要求对保证系统安全性是必要的,但不是充分的。随着对安全问题研究的深入,某种新的攻击方法的出现以及设计密钥流生成器的方法不同,还会为确保系统安全性提出一些更强的要求。

7.5 密钥流生成器

第 7.4 节讨论了序列密码对密钥流的安全性要求。一般地,安全性要求越高,设计越复杂。因此,密钥流生成器设计中,在考虑安全性要求的前提下还应考虑以下两个因素:

(1) 密钥 K 易于分配、保管,更换简单;

(2) 易于实现,快速。

为了满足这些要求,目前密钥流生成器大都是基于移位寄存器的。因为移位寄存器结构简单,易于实现且运行速度快。这种基于移位寄存器的密钥流序列称为移位寄存器序列。

通常采用的方法是:由线性移位寄存器(LFSR)和一个非线性组合函数,即布尔函数组合,构成一个密钥流生成器,图 7.4 所示即为密钥流生成器。其中(a)由一个线性移位寄存器和一个滤波器构成,(b)由多个线性移位寄存器和一个组合器构成。通常将这类生成器分解成两部分,其中线性移位寄存器部分称为驱动部分,另一部分称为非线性组合部分。其工作原理是将驱动部分,即线性移位寄存器在 j 时刻的状态变量 x 作为一组值输入非线性组合部分的 f,将 $f(x)$作为当前时刻的 k_j。驱动部分负责提供非线性组合部分

使用的周期大、统计性能好的序列,而非线性组合部分以各时刻移位寄存器的状态组合出密钥序列 j 时刻的值 k_j,驱动部分负责状态转移。

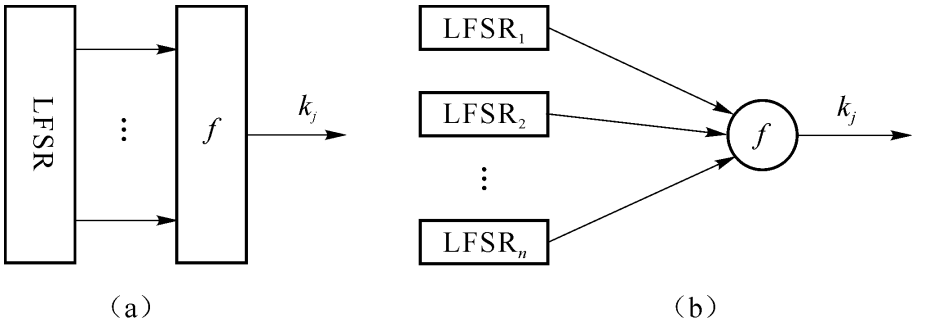


图 7 .4　密钥流生成器

由于线性移位寄存器的理论已经成熟,特别是 m 序列具有良好的伪随机性,所以驱动部分的设计相对容易,于是非线性组合部分的设计便成为密钥流生成器的重点与难点。而在二元情况下,非线性组合部分就可用一个布尔函数表示。于是密钥流生成器的研究归结为布尔函数的研究。

7 .6　线性移位寄存器

移位寄存器是序列密码中密钥流生成器的一个重要组成部分。一个 GF(2)上的反馈移位寄存器可用图 7 .5 表示。

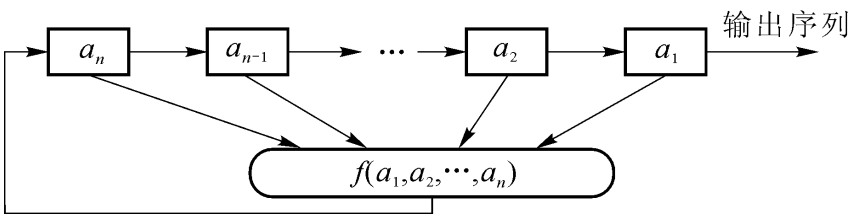


图 7 .5　反馈移位寄存器

图中标有 a_1 , a_2 ,⋯, a_{n-1} , a_n 的小方框表示二值(0 ,1)存储单元,可以是一个双稳触发器,信号流从左向右。这 n 个二值存储单元称为该反馈移位寄存器的级。在任一时刻,这些级的内容构成该反馈移位寄存器的状态。这个反馈移位寄存器的状态对应于一个 GF(2)上的 n 维向量,共有 2^n 种可能的状态。每个时刻的状态可用 n 长序列

$$a_1 , a_2 , \cdots , a_n$$

或 n 维向量

$$(a_1 , a_2 , \cdots , a_n)$$

表示,其中 a_i 为当前时刻第 i 级存储器中的内容。

在主时钟确定的周期区间上,每一级存储器 a_i 都将其内容向下一级 a_{i-1} 传递,并根

据寄存器当时的状态计算 $f(a_1, a_2, \cdots, a_n)$作为 a_n 下一时间周期的内容,称函数 $f(a_1, a_2, \cdots, a_n)$为反馈函数,其中反馈函数 $f(a_1, a_2, \cdots, a_n)$为 n 元布尔函数。在时钟脉冲时,如果反馈移位寄存器的状态为

$$S_i = (a_i, \cdots, a_{i+n-1}),$$

则

$$a_{i+n} = f(a_i, a_{i+1}, \cdots, a_{i+n-1})。 \tag{7.2}$$

这个 a_{i+n}又是移位寄存器的输入。在 a_{i+n}的驱动下,移位寄存器的各个数据向前推移一位,使状态变为

$$S_{i+1} = (a_{i+1}, \cdots, a_{i+n}),$$

同时,整个移位寄存器的输出为 a_i。由此我们得到一系列数据

$$a_1, a_2, \cdots, a_n, \cdots, \tag{7.3}$$

它们满足关系式(7.2),我们称无穷序列(7.3)为一个反馈移位寄存器序列。

例 7.1 有一个三级移位寄存器如图7.6所示。

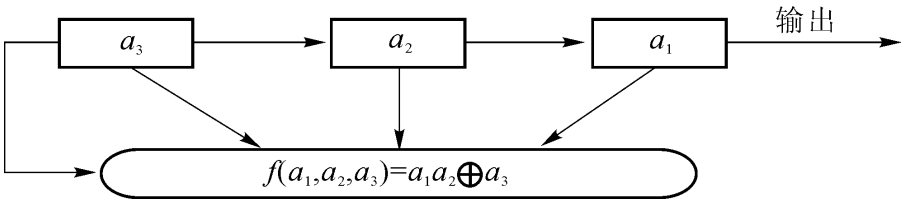


图 7.6 三级移位寄存器

其中初态为 $S_1 = (a_1, a_2, a_3) = (1,0,1)$,则

状态$(a_3\ a_2\ a_1)$			输 出
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	1
1	0	1	1
1	1	0	0
…	…	…	…

所以此移位寄存器输出序列为 1 0 1 1 1 0 1 1 1 0 1 1 …,它是周期为 4 的序列。

若移位寄存器的反馈函数 $f(a_1, a_2, \cdots, a_n)$是 $a_1, a_2, \cdots, a_n$ 的线性函数,则称为线性移位寄存器(LFSR),否则称为非线性移位寄存器。

设 $f(a_1, a_2, \cdots, a_n)$为线性函数,则 f 可写成

$$f(a_1, a_2, \cdots, a_n) = c_n a_1 \quad c_{n-1} a_2 \quad \cdots \quad c_1 a_n \tag{7.4}$$

其中 $c_i = 0$ 或 1, $c_1, c_2, \cdots, c_n$ 为反馈系数,假定其中至少有一非零,一般总假定 $c_n = 1$。

对于二进制作用下，c_1，c_2，…，c_n 的作用就相当于一个开关，用断开和闭合来表示 0 和 1，这样的线性函数共 2^n 个。

线性移位寄存器如图 7.7 所示。

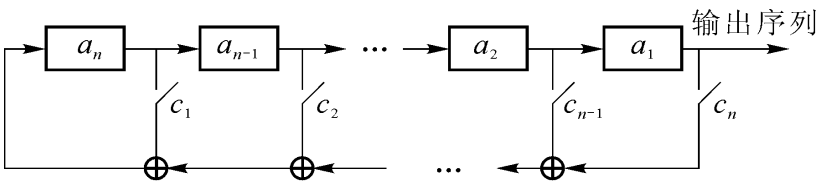


图 7.7　线性移位寄存器

输出序列$\{a_i\}$满足

$$a_{n+i} = c_n a_i \quad c_{n-1} a_{i+1} \quad \cdots \quad c_1 a_{n+i-1}, \tag{7.5}$$

其中 i 为正整数。

例 7.2　有一个四级线性移位寄存器如图7.8所示。其中初态为$(a_1, a_2, a_3, a_4) = (1000)$，则输出序列为

1 0 0 0 1 0 0 1 1 0 1 0 1 1 1 1 …

周期为 $2^4 - 1 = 15$。

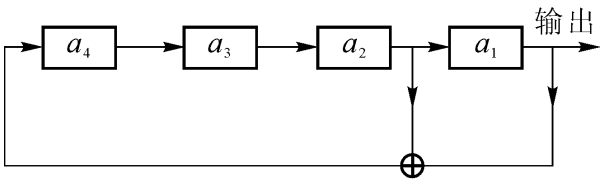


图 7.8　四级线性移位寄存器

由于在 n 级的条件下，最多给出 2^n 个状态；而在线性运算下，全“0”状态不会转入其他状态；所以，线性移位寄存器序列的最长周期为 $2^n - 1$。

定义 7.17　当 n 级线性移位寄存器产生的序列$\{a_i\}$的周期为 $T = 2^n - 1$ 时，则称$\{a_i\}$为 n 级 m 序列。

定理 7.6　n 级 m 序列$\{a_i\}$具有如下性质：

(1) 在一个周期内，0，1 出现次数分别为 $2^{n-1} - 1$ 和 2^{n-1} 次；

(2) 在一个周期圈内，总游程数为 2^{n-1}，对 $1 \leq i \leq n-2$，长为 i 的游程有 2^{n-i-1} 个，且 0，1 游程各半，长为 $n-1$ 的 0 游程一个，长为 n 的 1 游程一个；

(3) $\{a_i\}$的自相关函数为二值：

$$R_a(\tau) = \begin{cases} 1, & \text{当 } \tau = 0 \\ -\dfrac{1}{T}, & \text{当 } 0 < \tau \leq T-1 \end{cases}。$$

证明略(参见本书参考文献[52])。

可见 m 序列具有很好的随机特性，正好满足 Golomb 随机性的三个假设。如何选择合适的反馈函数使得序列的周期达到最大，即为 m 序列，这是我们急切想知道的问题。

根据密码学的需要，对于线性移位寄存器主要考虑下面两个问题：

(1) 如何利用级数尽可能小的线性移位寄存器产生周期长、统计性能好的序列；

(2) 已知一个序列$\{a_i\}$，如何构造一个尽可能短的线性移位寄存器来产生它。

然而,我们首先需要一些相关的代数学知识。

因为 n 级线性移位寄存器的输出序列$\{a_i\}$满足递推关系 $a_{n+i} = c_n a_i \quad c_{n-1} a_{i+1} \cdots \quad c_1 a_{n+i-1}$ 对任何 $i \geq 1$ 成立。这种递推关系可用一个一元高次多项式 $p(x) = \sum_{i=0}^{n} c_i x^i$ 表示,其中 $c_0 = 1$,称为该线性移位寄存器的特征多项式,当 $c_n \neq 0$ 时,则线性移位寄存器是非奇异的,有时也称非奇异的线性移位寄存器为非退化的。

当 $a_i \in GF(2)$, $i = 1, \cdots, n$ 时,上述 n 级线性移位寄存器共有 2^n 组初始状态,即有 2^n 个递推序列,其中非恒为零的序列有 $2^n - 1$ 个。令这 $2^n - 1$ 个非零序列的全体为 $\Omega(p(x))$,显然,$\Omega(p(x))$加上恒为 0 的序列构成一 n 维线性空间。

对于由上述线性移位寄存器输出的任一序列$\{a_i\}$,可以用母函数来表达:

$$A(x) = \sum_{i=1}^{\infty} a_i x^{i-1}。$$

定理 7.7 设线性移位寄存器的特征多项式为 $p(x) = \sum_{i=0}^{n} c_i x^i$, 递推序列 $\{a_i\} \in \Omega(p(x))$。令 $A(x) = \sum_{i=1}^{\infty} a_i x^{i-1}$,则

$$A(x) = \frac{\phi(x)}{p(x)}, \text{其中 } \phi(x) = \sum_{i=1}^{n} c_{n-i} x^{n-i} \sum_{j=1}^{i} a_j x^{j-1}。$$

证明略(参见本书参考文献[58])。

定理 7.7 说明 n 级线性移位寄存器的特征多项式与其母函数之间的关系,注意到 $\phi(x)$的次数低于 n,最多为 $n-1$ 次。

定义 7.18 设 $p(x)$为 GF(2)上的多项式,使得 $p(x) \mid x^p - 1$的最小 p 称为 $p(x)$的周期或 $p(x)$的阶。

定理 7.8 若 $p(x)$为 GF(2)上的 n 次多项式,且 $p(x)$是序列$\{a_i\}$的特征多项式, p 为 $p(x)$的阶,则$\{a_i\}$的周期 $r \mid p$。

证明 因为 $p(x)$的阶为 p,所以 $p(x) \mid x^p - 1$。故有 $q(x)$,使得 $p(x)q(x) = x^p - 1$。又因为 $p(x)A(x) = \phi(x)$,所以 $p(x)q(x)A(x) = \phi(x)q(x)$,即$(x^p - 1)A(x) = \phi(x)q(x)$。

$q(x)$的方次为 $p - n$,$\phi(x)$的方次不超过 $n-1$,故$(x^p - 1)A(x)$方次不超过 $p-1$。这就证明了 $a_{i+p} = a_i$,对任意的正整数 i 都成立。设 $p = kr + t, 0 \leq t < r$,则

$$a_{i+p} = a_{i+kr+t} = a_{i+t} = a_i,$$

所以 $t = 0$,即 $r \mid p$。

可见, n 级线性移位寄存器的输出序列的周期 r 不依赖于初始条件,而依赖于特征多项式 $p(x)$。我们感兴趣的是线性移位寄存器遍历 $2^n - 1$ 个非零状态,这时序列的周期达到最大$(2^n - 1)$,即 m 序列。

假定序列$\{a_i\}$的周期为 r。显然,特征多项式为 $1+x^r$ 的 r 级线性移位寄存器(纯轮换移位寄存器)可以输出$\{a_i\}$,特征多项式为 $1+x^{2r}$, $1+x^{3r}$,…的线性移位寄存器亦然。可见,能产生某一序列的特征多项式构成一个非空集合 J,我们把其中次数最低的多项式称为该周期序列的极小多项式,记为 $m(x)$,可以证明 J 中元素都是由 $m(x)$倍式构成。显然,用 $m(x)$作为特征多项式的线性移位寄存器产生序列的方案是最优的。

定理 7.9　设 $m(x)$是产生线性移位寄存器序列$\{a_i\}$的极小多项式,则$\{a_i\}$的周期等于多项式 $m(x)$的阶。

证明　设 $m(x)$的阶为 p,序列$\{a_i\}$的周期为 r,则 $m(x) \mid x^p-1$,说明 x^p-1 也是序列$\{a_i\}$的特征多项式,而 x^p-1 的阶为 p,因而由定理 7.8 知 $r \mid p$。

另一方面,序列$\{a_i\}$显然可由 r 级纯轮换线性移位寄存器产生, x^r-1 为其特征多项式,故 $m(x) \mid (1-x^r)$,因此有 $p \mid r$。

综上所述, $r=p$。

定理 7.10　n 级线性移位寄存器产生的状态序列有最大周期 2^n-1 的必要条件是其特征多项式 $p(x)$是不可约的。

证明　设特征多项式为 $p(x)$,若 $p(x)$可约,令 $p(x)=g(x)h(x)$,不妨设 $g(x)$不可约,且 $g(x)$的次数为 $k<n$,显然,$\Omega(g(x)) \subset \Omega(p(x))$,而 $\Omega(g(x))$中序列的周期一方面不超过 2^k-1,另一方面又等于 2^n-1,因此推出 $n \leq k$,这与前面的假设相矛盾。故 $p(x)$是不可约的。

以下例子说明定理 7.10 的逆命题不一定成立,这就是线性移位寄存器的特征多项式为不可约多项式,但其输出序列不一定是 m 序列。

例 7.3　设 $f(x)=x^4+x^3+x^2+x+1$ 是 GF(2)上的不可约多项式,但它的输出序列 000110001100011…周期为 5,不是 m 序列。

解　$f(x)$的不可约性由多项式 x, $x+1$, x^2+x+1 不能整除 $f(x)$而得。对任何 $k \geq 5$,输出序列只需用 $a_k=a_{k-1}\ a_{k-2}\ a_{k-3}\ a_{k-4}$进行检验即可。

定义 7.19　$p(x)$为 n 次不可约多项式,若 $p(x)$阶为 2^n-1,称 $p(x)$为 n 次本原多项式。

定理 7.11　$\{a_i\}$为 n 级 m 序列的充要条件是其特征多项式 $p(x)$为 n 次本原多项式。

证明略(参见本书参考文献[52])。

定理 7.11 表明$\{a_i\}$为 n 级 m 序列的关键在于 $p(x)$为 n 次本原多项式, n 次本原多项式的个数为 $\lambda(n)=\dfrac{\phi(2^n-1)}{n}$,其中 ϕ 为欧拉函数。已经证明,对于任意的正整数 n,至少存在一个 n 次本原多项式,且 n 次本原多项式有表可查。这表明,对于任意的 n 级线性移位寄存器,至少有一种连接方式使其输出序列为 m 序列。

例 7.4　设, $p(x)=x^4+x+1$, $p(x)$为 4 次本原多项式,以其为特征多项式的线性

移位寄存器输出序列为 1001000111101011001000111101 01…,它是周期为 $2^4 - 1 = 15$ 的 m 序列。

解　由 $p(x) \mid x^{15} - 1$,但不存在 $l < 15$,使得 $p(x) \mid x^l - 1$,所以 $p(x)$ 的阶为 15。

$p(x)$ 的不可约性由 x, $x+1$, $x^2 + x + 1$ 多项式不能整除 $p(x)$ 即得,于是 $p(x)$ 为本原多项式。

$p(x)$ 为特征多项式的输出序列满足递推关系。对任何 $k \geq 5$,用 $a_k = a_{k-1}$　a_{k-4} 进行检验即可。

下面讨论第二个问题,即已知一个序列 $\{a_i\}$,如何构造一个尽可能短的线性移位寄存器来产生它。

对于一个长为 N 的序列 $a = a_0, a_1, \cdots, a_{N-1}$,显然它可被一个 N 级的线性移位寄存器产生,只要取 $f(x) = 1 + x^N$ 即可。但通常 a 还可能由更短的线性移位寄存器产生。

定义 7.20　二元序列 $a = a_0, a_1, \cdots, a_{N-1}$ 的线性复杂度 $C(a)$ 定义为产生该序列的级数最少的线性移位寄存器的级数。对全零序列 a,约定 $C(a) = 0$。

定理 7.12　设 $a = \{a_i\}$ 是二元周期序列,且序列 $\{a_i\}$ 的线性复杂度 $C(a) = L \geq 1$,则只要知道 $\{a_i\}$ 中任意相继的 $2L$ 位就可确定整个序列 $\{a_i\}$ 及产生 $\{a_i\}$ 的极小多项式。

证明略(参见本书参考文献[52])。

由定理 7.12 可见,线性复杂度是序列线性预测性的一个指标,作为密钥序列,其线性复杂度很小是不安全的。对于 n 级 m 序列,它的线性复杂度为 n,尽管它具有周期 $2^n - 1$ 及很好的随机统计特性,它还是不能直接用作密钥序列。

一些线性序列的线性复杂度可以通过分析给定。对于一些较复杂的、特别是非线性序列的线性复杂度可用 Berlekamp-Massey 算法(即 B-M 算法)求解(参见本书参考文献[52])。

虽然 n 级线性移位寄存器产生的 m 序列具有良好的伪随机特性,然而直接用其构造密钥流序列是极不安全的,因为利用 $2n$ 个输出位可以找到它的起始状态及特征多项式。例如,特征多项式 $p(x) = \sum_{i=0}^{n} c_i x^i = x^3 + x + 1$,初始状态为(1,0,1)的线性移位寄存器,所产生的序列为(1,0,1,0,0,1)。图 7.3 中设明文 m 为(0,1,1,0,1,0),则密文 c 为(1,1,0,0,1,1)。破译者计算 m　c 得到密钥序列为 $k = (k_1, k_2, \cdots, k_6) = (1,0,1,0,0,1)$,则从线性移位寄存器的结构很容易得到下列矩阵方程式

$$\begin{matrix} k_1 & k_2 & k_3 & c_3 & & k_4 \\ k_2 & k_3 & k_4 & c_2 & = & k_5 \\ k_3 & k_4 & k_5 & c_1 & & k_6 \end{matrix},$$

所以

$$\begin{matrix}1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0\end{matrix}\begin{matrix}c_3 \\ c_2 \\ c_1\end{matrix}=\begin{matrix}0 \\ 0 \\ 1\end{matrix}。$$

可以得到 $c_3=1, c_2=0, c_1=1$,所以特征多项式 $p(x)=x^3+x+1$。

7.7　非线性序列

线性移位寄存器序列密码在已知明文攻击下是可破译的这一事实促使人们向非线性领域探索。目前研究得比较充分的方法有非线性移位寄存器序列,对线性移位寄存器序列进行非线性组合等。

7.7.1　非线性移位寄存器序列

根据图 7.5 可知,令反馈函数 $f(a_1, a_2, \cdots, a_n)$ 为非线性函数,便构成非线性移位寄存器,其输出序列为非线性序列。输出序列的周期最大可达 2^n,并称周期达到最大值的非线性移位寄存器序列为 M 序列。M 序列具有下面定理所述的随机统计特性:

定理 7.13　在 n 级 M 序列的一个周期内,0 与 1 的个数各为 2^{n-1};在 M 序列的一个周期圈中,总游程为 2^{n-1},对 $1\leq i\leq n-2$,长为 i 的游程数为 2^{n-1-i},其中 0,1 游程各半,长为 $n-1$ 的游程不存在,长为 n 的 0 游程和 1 游程各 1 个。

证明　在 M 序列的状态构成的一个周期内,GF(2)上的每个 n 长状态恰好出现一次,而由状态圈中各状态的第 1 分量构成的序列就是对应的 M 序列的一个周期,故其中 0,1 各为 2^{n-1} 个。关于游程特性,对 $n=1,2$ 易证结论成立。对 $n>2$,当 $1\leq i\leq n-2$ 时,n 级 M 序列的一个周期圈中,长为 i 的 0 游程数目等于序列中如下形式的状态数目:$1\underbrace{00\cdots0}_{i\text{个}0}1*\cdots*$,其中 $n-i-2$ 个 * 号位置可任取 0 和 1 值。因而这种状态数目为 2^{n-i-2}。

对长为 n 的 0 游程,其形式为 $1\underbrace{0\cdots0}_{n\text{个}0}1$,由序列周期为 2^n 知 $f(0,0,\cdots,0)=1$,故长为 n 的 0 游程一个;若有 $n-1$ 长的 0 游程,其形式为 $\underbrace{10\cdots01}_{n-1\text{个}0}$,而 $\underbrace{10\cdots00}_{n-1\text{个}0}$ 已出现在 n 长的 0 游程中,故周期圈中它不可能再出现,因而不存在 $n-1$ 长 0 游程。1 游程数目可类似求得。

定理 7.14　GF(2)上 n 级 M 序列的数目为 $2^{2^{n-1}-n}$。

证明略。

由上面性质可见,M 序列具有很好的随机统计特性,又有大量的不同序列可供选用,因而它在序列密码中一直是人们研究的主要内容之一。

n 级移位寄存器共有 2^{2^n} 种不同的反馈函数,而线性反馈函数只有 2^n 种,其余均为非线性的。可见,非线性反馈函数的数量是巨大的。但是值得注意的是,并非这些非线性反馈函数都能产生良好的密钥序列,其中 M 序列是比较好的一种。

一般的非线性反馈移位寄存器研究尚处于艰难的研究之中,所以目前人们研究更多的还是在线性移位寄存器基础上的非线性化问题。

7.7.2 非线性前馈序列

第 7.5 节已提到,线性移位寄存器序列虽然不能直接作为密钥流使用,但可作为驱动源以其输出推动一个非线性组合函数所决定的电路来产生非线性序列。实际上,这就是所谓的非线性前馈序列生成器。线性移位寄存器用来保证密钥流的周期长度,非线性组合函数用来保证密钥流的各种密码性能,以抗击各种可能的攻击。许多专用流密码算法都是用这种方法构成,其中不少是安全的。

图 7.4(a)表示的生成器是由 E.J.Groth 于 1971 年提出的,为了叙述方便用图 7.9 表示。

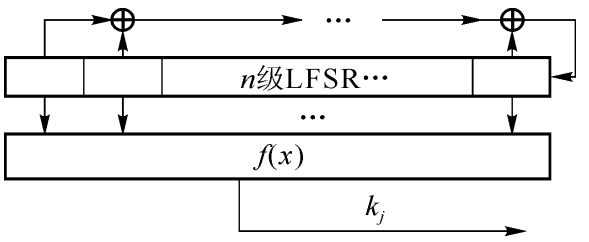


图 7.9 非线性前馈序列生成器

图中 $f(x)$ 是一个 n 元布尔函数,对于 LFSR 的状态变量,由非线性函数滤波后得到输出序列 $\{k_j\}$,称这种生成器为前馈网络,称 $\{k_j\}$ 为前馈序列,于是布尔函数 $f(x)$ 在这里也被称为前馈函数。用 $\delta_j=(s_j, s_{j+1}, \cdots, s_{j+n-1})$ 表示 n 级 LFSR 在时刻 j 的移存器状态,用 δ_0 表示初态。显然,前馈序列 $\{k_j\}$ 的周期不会超过 δ_j 可能达到的最大周期 2^n-1,所以总是选取 n 级 m 序列生成器作为驱动器 LFSR。假定 LFSR 是某个 n 级 m 序列生成器,$\delta_0 \neq 0$,$f(0)=0$,则任意给定前馈序列的前 2^n-1 位 $k_j(j=0,1,2,\cdots,2^n-2)$ 时,$f(x)$ 便惟一确定了。因为这时 $\delta_j(j=0,1,\cdots,2^n-2)$ 取遍 $GF(2)^n$ 中非零向量。$f(0)=0$,$f(\delta_j)=k_j(j=1,2,\cdots,2^n-2)$ 确定了 $f(x)$ 的真值表。这一事实可叙述如下:

引理 7.1 在图 7.9 中,n 级 LFSR 为 n 级 m 序列生成器时,对任一组不全为 0 的 $k_j(j=0,1,2,\cdots,2^n-2)$,存在惟一的前馈函数 $f(x)$,使前馈序列是周期序列

$$k = k_0\, k_1 \cdots k_{2^n-1}\, k_0\, k_1 \cdots,$$

这里 $f(0)=0$。

引理 7.1 表明线性复杂度为 n 的 m 序列经过适当的前馈函数滤波,可以得到一个复杂度接近 2^n-1 的前馈序列,相对于驱动序列,其复杂度指数地增长。

一般地,对于一个给定的如图 7.9 的前馈网络,其前馈序列 $\{k_j\}$ 的线性复杂度有如下结论:

定理 7.15 设 LFSR 为 n 级 m 序列生成器,前馈函数 $f(x)=f(x_0, x_1, \cdots, x_{n-1})$,

若 $f(x)$的次数为 k,则$\{k_j\}$的线性复杂度 $C(\{k_j\})$满足

$$C(\{k_j\}) \leq \sum_{j=1}^{k} C_n^j。$$

证明略。

可见,前馈序列的线性复杂度和前馈函数的次数密切相关。另外前馈序列的统计特性与 $f(x)$密切相关,如增加前馈函数的项数可改善前馈序列的统计特性。

根据以上讨论,前馈序列生成器中,布尔函数的特性决定着前馈序列的性能,因此,布尔函数是前馈密钥流设计的一个关键。

7.7.3　非线性组合序列

前面讨论的前馈序列是由一个线性移位寄存器驱动的非线性前馈序列生成器所产生的序列。这类序列的周期只能是 $2^n - 1$ 的因子。为了提高序列的线性复杂度和随机性,一种自然的方法就是在驱动部分用多个 LFSR 进行组合,这就是本节要讨论的由多个线性移位寄存器驱动的非线性组合序列(生成器),如图 7.10 所示。

$\mathrm{LFSR}_i(i=1,2,\cdots,n)$为 n 个级数分别为 $r_1, r_2, \cdots, r_n$ 的线性移位寄存器,相应的序列分别为 $a_i = \{a_{i_j}\}(i=1,2,\cdots,n)$, $f(x) = f(x_1,\cdots,x_n)$是 n 元布尔函数。令 $k_j = f(a_{1_j},\cdots,a_{n_j})$,则序列 $k=\{k_j\}$是由图 7.10 所示生成器产生的序列。

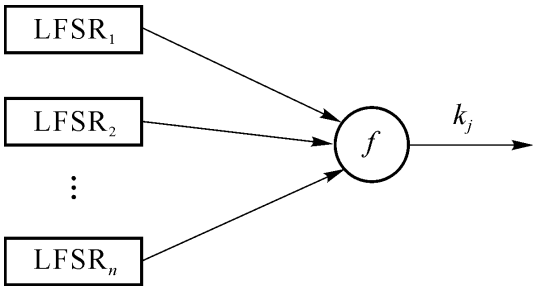


图 7.10　非线性组合序列生成器

称 $f(x)$为非线性组合函数,$\{k_j\}$为非线性组合序列。关于非线性组合序列有如下结论:

定理 7.16　设 $a_{ij}, \{k_j\}, f(x), r_i, (i=1,2,\cdots,n)$,如前所述,若 r_i 两两互素,$f(x)$与各变元均有关,则$\{k_j\}$的周期为

$$\prod_{i=1}^{n} (2^{ri} - 1),$$

线性复杂度为

$$C(\{k_j\}) = f(r_1,\cdots,r_n),$$

其中 $f(r_1,\cdots,r_n)$中按实数域上运算。

证明略。

可见,采用非线性组合函数,对多个 m 序列进行组合,可极大提高序列的周期和线性复杂度。

除以上两类序列之外,还有多路复合序列和钟控序列,这类序列也可归结为非线性组合序列,可看做非线性组合序列的特殊形式。因此,不再专门叙述,有兴趣的读者可参见本书参考文献[58]。

7.8 序列密码分析

前面已叙述过,流密码体制是以其密钥流生成器的不同区分的。如非线性前馈流密码、非线性组合流密码、钟控流密码等就是与相应的密钥流生成器对应的。我们仍假定所有加密变换都是采用二元加法,即模二相加。这种密码体制即所谓二元加法流密码。下面以二元加法非线性组合流密码为例讨论下述两种分析方法。

7.8.1 二元加法非线性组合流密码的相关攻击

一个二元加法非线性组合流密码如图 7.11 所示。如前所述,这类流密码的安全性关键在于密钥流生成器,即非线性组合器部分。它由 n 个不同的线性移位寄存器 LFSR_i 和一个非线性组合函数 $f(x)$构成。在二进制情况下, $f(x)$就是一个布尔函数。这种密钥流生成器不但结构简单,易于工程实现,更重要的是密钥流 z_k 具有较好的伪随机性,因而是一类比较理想的密钥流生成器,但与许多其他类型的密钥流生成器一样,这种生成器也有其自身的弱点。Siegenthaler 于 1985 年给出了一种称为“分别征服”的相关攻击方法(即 DC 攻击)。下面将对这一攻击方法做具体描述。

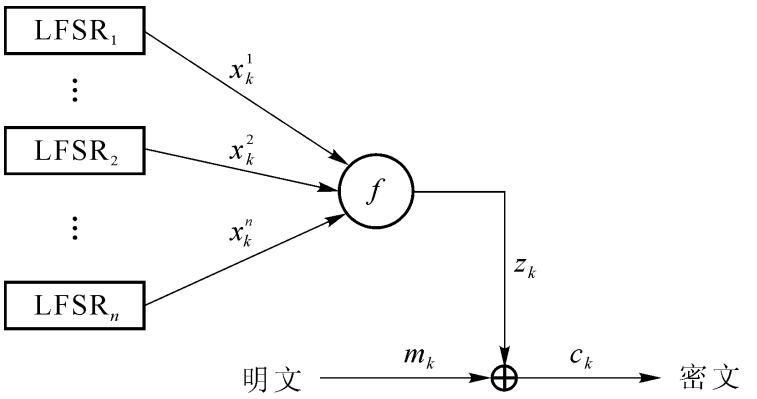


图 7.11 二元加法非线性组合流密码

现在讨论图 7.11 的由 n 个 LFSR_i 驱动的非线性组合序列产生器的破译问题。如果某个驱动序列 x_k^i 与输出序列 z_k 的信号符合率等于 $\frac{1}{2}+\varepsilon$, $\varepsilon>0$,就称$\{x_k^i\}$与$\{z_k\}$是统计相关的,这是驱动序列$\{x_k^i\}$的信息在输出序列$\{z_k\}$中的一种泄漏(或一种熵漏)。利用这种统计相关性实施的攻击称为相关攻击。更一般地,如果 n 个驱动序列被分成若干小组,某小组中序列的线性和序列与输出序列$\{z_k\}$的信号符合率大于 1/2,相关攻击的方案仍然有效。对密钥流生成器的这种基于统计分析的相关攻击方法,首先由 Blaser 和 Heinzmann 于 1978 年提出并进行了研究。Siegenthaler 发展了这种方法,给出了 DC 攻击——“分别征服”——的相关攻击方法。

DC 攻击的基本思想是:根据非线性组合器的输出序列$\{z_k\}$与组合函数 $f(x)$的每个输入序列$\{x_k^i\}$之间的相关性,用统计方法恢复出 LFSR_i 的初态和反馈函数。用这种方法可大大降低寻找密钥所需的试验次数,在图 7.11 中假设每个 LFSR_i 都产生最大周期序列,并设 LFSR_i 的级数为 r_i, r_i 次本原多项式的个数为 R_i,则对密码分析者来说,第 i 个 LFSR的未知参数有 $R_i(2^{r_i}-1)$个,因此,该系统的密钥量为 $K=\prod_{i=1}^{n} R_i(2^{r_i}-1)$。如果使用穷搜索密钥攻击方法,那么最坏的情况下 K 个密钥都需试一次,若 r_i 足够大,所需计算量是无法实现的。

DC 攻击采用的是对 n 个 LFSR 各个击破的方式。因此,最坏的情况下,试验次数大约可减至

$$\sum_{i=1}^{n} R_i 2^{r_i}。$$

这种攻击是利用某些输入 x_k^i 与 z_k 之间的相关性,逐步确定每个 LFSR_i 子密钥的。因此,需先根据最大长度序列的统计特性,建立统计模型。设 $f(x)$的输入 $x_k^1, x_k^2, \cdots, x_k^n$ 是独立的,服从均匀分布的随机变量,即 $p(x_k^i=0)=p(x_k^i=1)$, $p(\cdot)$表示概率, $f(x)$生成的 z_k 亦是服从均匀分布的随机变量, $p(z_k=0)=p(z_k=1)$。设

$$p(z_k = x_k^i) = q_i, \qquad (7.6)$$

假定明文是二元无记忆信源,满足

$$p(m_k = 0) = p_0。 \qquad (7.7)$$

DC 攻击是惟密文攻击,注意到密文 c_k 与 z_k 和 m_k 有关,而 z_k 又与 x_k^i 有关,因而 c_k 也间接地与 x_k^i 有关,为了分析 N 个比特密文 $c_1 c_2 \cdots c_N$ 中含有 LFSR_i 的信息量,先计算 $c_1 c_2 \cdots c_N$ 与 $x_1^i x_2^i \cdots x_N^i$ 之间的相关度

$$\alpha = \sum_{k=1}^{N} (1 - 2(c_k + x_k^i))/N$$

$$= 1 - 2\sum_{k=1}^{N} (c_k + x_k^i)/N \qquad (i = 0,1,\cdots,n)。 \qquad (7.8)$$

这里 x_k^0 是一个与 $x_k^i(i=1,2,\cdots,n)$统计独立的服从同一分布(均匀分布)的随机变量。

根据中心极限定理,当 N 很大时,α服从正态分布。同时,容易计算 c_k 与 x_k^i 之间的符合率 p_e。

$$\begin{aligned} p_e &= p(c_k = x_k^i) = p(z_k + m^k = x_k^i) \\ &= p(z_k = x_k^i)\,p(m_k = 0) + p(z_k \neq x_k^i)\,p(m_k = 1)。 \qquad (7.9) \\ &= 1 - (p_0 + q_i) + 2p_0 q_i。 \end{aligned}$$

从(7.9)式可见, p_e 越大,说明 c_k 与 x_k^i 之间的符合率越大。从而密文序列段含有

$LFSR_i$ 子密钥的信息量就越大。极端情形下，p_e 接近于 1，这时显然密文序列段与对应的 $LFSR_i$ 输出段近似地相同，从而该系统是极不安全的。从 p_e 和 α 的定义和表达式易见，密文序列段 $c_1 \cdots c_N$ 中所含 $LFSR_i$ 子密钥的信息量大小由明文特性 p_0、z_k 和 x_k^i 的符合率 q_i 及密文长度 N 所决定。

进一步通过相关度 α 的概率分布、假设检验、错误决策概率的分析，可给出攻击每个 $LFSR_i$ 子密钥所需密文数字个数 N_1 的上界：

$$N_1 < \frac{(1/2)\ \ln(R_i 2^{r_i-1}) + r_0\ \ p_e(1-p_e)}{p_e - \frac{1}{2}}^2, \qquad (7.10)$$

式中，r_0 是常数，选取长度不小于 N_1 的密文序列就能成功地对图 7.11 所示的模型进行攻击。

以上是对 DC 攻击思想及方法的一个简要描述，详细讨论见本书参考文献[58]。从上面的讨论知，DC 攻击是建立在组合函数与其变元的相关性及信源的 0，1 不平衡基础上的。这可以从(7.10)式中 N_1 的上界表达式中清楚地看出，该式分子基本上随 R_i 和 r_i 线性增加，所以增加 N_1 的有效方法是使 p_e 尽可能地接近 1/2，而 $p_e = 1 - (p_0 + q_i) + 2p_0 q_i$。所以，使 p_e 接近 1/2 有两种途径，其一是对明文进行适当编码，使 p_0 尽可能接近 1/2，这是信源编码问题，本书不作讨论。其二是设计好的组合函数，使得 $q_i = 1/2$(或充分接近 1/2)。这时 N_1 趋于无穷大，从而 DC 攻击无法实现。这就是 Siegenthaler 提出相关免疫函数的动机。相关免疫函数，即 $q_i = p(f(x) = x_i) = \frac{1}{2}$，对任意 i 都成立的函数。

7.8.2 二元加法非线性组合流密码的线性逼近攻击

最佳线性逼近方法在逻辑电路的简化中得到了广泛的应用。通过使用最佳线性逼近方法，电路实现的复杂度大大降低。本节简要介绍这种方法的思想，仍以图 7.11 为模型，并沿用有关记号。

这种方法假定分析者已知如下参数：

(1) 非线性组合函数 $f(x)$；

(2) $LFSR_i$ 的级数 $r_i (i = 1, \cdots, n)$；

(3) 明文编码及语言统计特性；

(4) m 比特密钥流 $z_1\ z_2 \cdots z_m\ (m > 2(r_1 + \cdots + r_n))$。

攻击的目的不是恢复原密钥流生成器，而是利用已知信息构造一个新生成器，级数不超过 $\sum r_i$，以此代替原密钥流生成器，从而达到对密文的近似解密。

由于新构造的生成器是一个级数不超过 $\sum r_i$ 的线性移位寄存器，其复杂度不超过 $\sum r_i$，远远小于原生成器的复杂度 $f(r_1, \cdots, r_n)$。所以该方法的本质是用低复杂度序列

去逼近高复杂度序列。

定义 7.21　若线性函数 $w\cdot x+w_0$ 使得

$$p(f(x)=w\cdot x+w_0)\qquad (w\in \mathrm{GF}(2)^n,\ w_0\in \mathrm{GF}(2))$$

取最大值，则称 $w\cdot x+w_0$ 是 $f(x)$ 的最佳线性逼近。根据 7.1 节定理 7.2

$$p(f(x)=w\cdot x)=\frac{1+s_{(f)}(w)}{2} \tag{7.11}$$

记 $a=\max\limits_{w}\ |s_{(f)}(w)|\ |\ w\in \mathrm{GF}(2)^n$，则(7.11)式的最大值为 $\frac{1+a}{2}$，于是可得

定理 7.22　设 $f(x)$ 是 n 元布尔函数，w 满足 $|s_{(f)}(w)|=a$，则

当 $s_{(f)}(w)>0$ 时，$w\cdot x$ 是 $f(x)$ 的最佳线性逼近；

当 $s_{(f)}(w)<0$ 时，$w\cdot x+1$ 是 $f(x)$ 的最佳线性逼近。

$f(x)$ 与其最佳线性逼近的符合率为 $\frac{1+a}{2}$。

根据定理 7.22 知，若 $f(x)$ 在 w 点谱的绝对值最大，则 $w\cdot x+w_0$ 为 $f(x)$ 的最佳线性逼近，即它与 $f(x)$ 的符合率最高(为 $\frac{1+a}{2}$)，用这个与 $f(x)$ 符合率最高的线性函数(记为 $l(x)$)来代替 $f(x)$，产生一个序列 $z'=\{z_j'\}$，$z_j'=l(x_j^1,\cdots,x_j^n)$。根据定理 7.16，以 $f(x)$ 为组合函数的序列 $z=\{z_k\}$ 的复杂度为 $f(r)$，以 $l(x)$ 为组合函数的序列 $z'=\{z_j'\}$ 的复杂度为 $l(r)\le\sum\limits_{i=1}^{n}r_i$，这里，$r=(r_1,\cdots,r_n)$，由于 z_j 和 z_j' 的符合率为 $f(x)$ 与 $l(x)$ 的符合率：$\frac{1+a}{2}$，于是 $\{z_j\}$ 和 $\{z_j'\}$ 的符合率有如下关系：$\{z_j\}$ 的任一 N 长截段和 $\{z_j'\}$ 的相应截段的符合率为 $(\frac{1+a}{2})^N$，即当 $f(x)$ 和 $l(x)$ 的符合率很高时，$\{z_j\}$ 和 $\{z_j'\}$ 的符合率也很高。

由于 $\{z_j'\}$ 的线性复杂度为 $l(r)\le\sum\limits_{i=1}^{n}r_i=r$，故只要知道其一个 $2r$ 长截段，便可用 B-M 算法确定出产生它的 LFSR，r 是各 r_i 的线性和，从而可用硬件实现。

下面的问题是如何确定 $\{z_j'\}$ 的一个 $2r$ 截段。

分两种情况：

(1) 据假设已知 $\{z_j\}$ 的一个 m 长截段，$m>2r$，所以共有 $m-2r+1$ 个 $2r$ 截段，$z_iz_{i+1}\cdots z_{i+2r-1}$，$1\le i\le m-2r+1$。$\{z_j'\}$ 的相应截段为 $z_i'\cdots z_{i+2r-1}'$。如果 $\frac{1+a}{2}$ 很高，比如说 $\frac{1+a}{2}>\frac{2r-1}{2r}$，则上面两组 $m-2r+1$ 个截段中，可期望有一对是相同的(因 $(\frac{2r-1}{2r})^{2r}\approx\frac{1}{e}>\frac{1}{3}$，所以每 3 个截段至少有一个是相同的)。当然，这时我们并不知道 $\{z_j'\}$，所以并不知道哪一对截段是相同的，故对所有 $m-2r+1$ 个截段都求出相应的

LFSR,到底哪一个是产生$\{z_j'\}$的 LFSR,留到后面讨论。

(2) 当$\frac{1+a}{2}$很小时,若 m 不足够大,上述方法失效,这时可用$\{z_j'\}$的一个 $2r$ 截段对其所有可能位置改变 1 个比特,2 个比特,……,直到 t 个比特,$t \geq 1-(\frac{1+a}{2})^{2r} 2r$,在得到的序列中必有一个是$\{z_j'\}$的相应截段,因此只要对上述改变比特后的截段求出相应的 LFSR,这时和情形(1)一样。我们仍然得到了一系列 LFSR,其中有一个是真正产生$\{z_j'\}$的 LFSR。

综上所述,无论哪种情形,我们都得到了多个 LFSR。为确定哪一个是真正产生$\{z_j'\}$的 LFSR,还需进一步检验。可用语言多余度或密码系统本身的参数。总之,无论哪种情形,都可确定产生$\{z_j'\}$的 LFSR。

以上是对线性逼近攻击方法的简单描述。可以看出,这种方法的成功与否取决于$\frac{1+a}{2}$(从而取决于 a)的大小和密钥比特 m 的大小,但关键还是 a 的大小。a 越大,实施攻击付出的计算代价越小。下面以一个攻击实例结束本节的讨论。

假定在图 7.11 中 $n=5$, $r_1=3$, $r_2=4$, $r_3=5$, $r_4=6$, $r_5=7$,非线性组合函数 $f(x)$的真值表为(0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0),再已知如下 51 个比特密钥流序列段:

$z_1 \cdots z_{51}$ = 1 0 0 1 1 0 0 0 0 1 1 0 1 0 0 0 10 0 0 0 1 0 0 0 10 0 1 1 0 1 0 1 0 1 0 1 1 01 1 0 0 1 0 1 1 0 0 1。

如果每个 LFSR 均使用本原多项式,密钥流序列的线性复杂度为 6677, $f(x)$的最大谱值 $a=15/16$,最大谱值点为(0 1 0 1 0),最佳线性逼近函数为 x_2+x_4,这样,$r=4+6=10$,按前述方法对第 10 个截段后的每个 $2r$ 段都有$(g(x), l)$为$1+x^2+x^4+x^5+x^6+x^7+x^{10}$,10),因而基本上可认为这个 LFSR 即为产生$\{z_j'\}$的 LFSR。可进一步检验判断其正确性。从而就构造出了级数为 10 的 LFSR,使它与原密钥流生成器的符合率为 31/32。

从这个例子可以看出,一个线性复杂度高达 6677 的密钥流序列可用复杂度为 10 的序列去逼近,其符合率为 31/32。这是由其最大谱值 $a=15/16$ 决定的,即当 a 值较大(接近 1)时,一个复杂度很高的序列可以以很高的概率$(\frac{1+a}{2})$,用一个复杂度很低的序列逼近。因此 a 值是衡量密钥流序列安全性的一个重要指标。

注 记

流密码理论相对而言比较成熟,主要原因之一是数学这一工具可用于流密码的研究。有限域和移位寄存器序列是流密码的基础,关于移位寄存器,可参阅本书参考文献[52]～[55]。关于有限域,可参看本书参考文献[52],[56]。目前有几本流密码专著,感兴趣的读者可参阅本书参考文献[57],[58]。当然,一些密码学著作也用专门的章节介绍了流密码。

本书参考文献[26]把构造流密码的方法归纳为四种:信息论方法、系统论方法、复杂度方法、随机化方法,并给出了大量的密钥流生成器,有兴趣的读者可自己阅读。

对布尔函数感兴趣的读者请参看本书参考文献[21]。

关于有限域与环上的序列设计,感兴趣的读者可参阅本书参考文献[24]。

习 题 七

1.设 4 级线性移位寄存器的反馈函数为

$f(a_1, a_2, \cdots, a_4) = c_4 a_1 \quad c_3 a_2 \quad \cdots \quad c_1 a_4$,其中 $c_1 = c_4 = 1$,且 $c_2 = c_3 = 0$,初始状态 (1000),写出该移位寄存器的输出。

2.如果序列$\{a_i\}$的周期为 r,且对任意非负整数 i 有 $a_{i+q} = a_i$,证明 $r \mid q$。

3.设 3 级线性反馈移位寄存器在 $c_3 = 1$ 时可有 4 种线性反馈函数,设初态为 $(a_3\ a_2\ a_1) = (101)$,求 4 种线性反馈函数的输出序列及周期。

4.假设破译者得到密文串 1010110110 和相应的明文串 0100010001。假定攻击者也知道密钥流是使用 3 级线性移位寄存器产生的,试破译该密码系统。

5.设 5 级线性移位寄存器中,反馈系数为$(c_0, c_1, c_2, c_3, c_4) = (1,0,0,1,1)$,输入状态$(11101)$。计算该寄存器的输出序列,并验证该序列是 m 序列。

6.设 $n = 4$, $f(a_1, a_2, a_3, a_4) = a_1 \quad a_4 \quad 1 \quad a_2 a_3$ 初态为$(a_1\ a_2\ a_3\ a_4) = (1011)$,试求此非线性移位寄存器的输出序列及周期。

第 8 章　分组密码

分组密码是现代密码学的重要分支之一,其主要任务是提供数据保密性。在信息化网络时代,越来越多的敏感或机密信息需要通过网络传输、存储和处理,因而,保密是人们的一个迫切需要。

所谓分组密码,通俗地说就是数据在密钥的作用下,一组一组、等长地被处理,且通常情况是明、密文等长。这样做的好处是处理速度快,节约了存储,避免了浪费带宽。

分组密码也是许多密码组件的基础。比如很容易转化为流密码、Hash 函数。

分组密码的另一特点是容易标准化,分组密码由于其固有的特点(高强度、高速率、便于软硬件实现)而成为标准化进程的首选体制。DES 就是首先成为数据加密标准的分组密码典型代表。作为数据加密标准,DES 算法完全公开,任何个人和团体都可以使用,其信息的安全性取决于各自密钥的安全性,这正是现代分组密码的特征。

DES(Data Encryption Standard)是曾被广泛使用的分组密码,遍及世界的政府、银行和标准化组织把 DES 作为安全和认证通信的基础。DES 算法的公开是密码学史上里程碑式的事件,开创了密码学民间应用之先河,大大推进了现代密码学的进展。随着计算技术的进步,DES 的 56 比特的密钥长已不适应现在的商业应用。

1997 年 4 月,NIST(美国国家标准技术研究所)发起了征集 AES(Advanced Encryption Standard)的活动,许多优秀的算法被提交,进一步刺激了分组密码设计理论和实践的进展。

8.1　分组密码概述

有两种重要类型的密码系统,单钥(私钥)和双钥(公钥)密码系统。在单钥密码系统中,明文的加密和密文的解密是用同样的密钥。直到 1976 年 Diffie、Hellman 引入公钥(双钥)密码学之前,所有的密码都是单钥系统,因此单钥系统也称为传统密码系统。传统密码系统广泛地用在今天的世界上,有两种单钥密码体制:流密码和分组密码。分组密码又分为三类:代替密码(Substitution)、移位密码(Transposition)和乘积密码。随着计算技术的发展,早期的代替和移位密码已无安全可言。一个增加密码强度的显然的方法是合

并代替和移位密码,这样的密码称为乘积密码。如果密文是由明文运用轮函数多次而得,这样的乘积密码又称为迭代分组密码。DES 和今天的大多数分组密码都是迭代分组密码。

分组密码就是将明文消息序列

$$m_1, m_2, \cdots, m_k, \cdots$$

分成等长的消息组$(m_1, m_2, \cdots, m_n), (m_{n+1}, m_{n+2}, \cdots, m_{2n}), \cdots$

在密钥控制下,按固定的算法 E_k 一组一组进行加密。加密后输出等长密文组

$$(y_1, \cdots, y_m), (y_{m+1}, \cdots, y_{2m}), \cdots$$

可用图 8.1 所示的框图来表示。

一般地,可以对分组密码作如下定义。

定义 8.1　一个分组密码是一种映射:

$$F_2^n \times F_2^t \rightarrow F_2^m$$

记为 $E(X, K)$或 $E_K(X)$, $X \in F_2^n$, $K \in F_2^t$, F_2^n 称为明文空间, F_2^m 称为密文空间, F_2^t 为密钥空间。n 为明文分组长度,当 $n > m$ 时,称为有数据压缩的分组密码;当 $n < m$ 时,称为有数据扩展的分组密码,当 $n = m$ 且为一一映射时, $E_k(x)$就是 $GF(2)^n$ 到 $GF(2)^n$ 的置换。通常的最常见的情况是 $n = m$,以后在不作特别说明时,都是指这种情况。

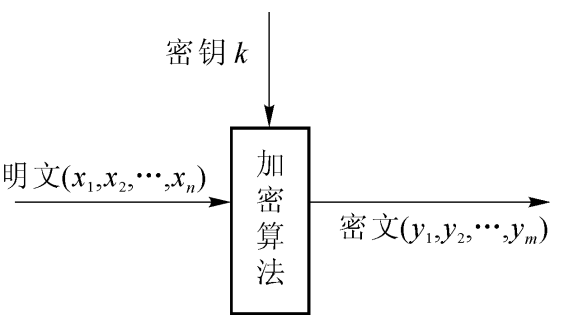


图 8.1　加密一组明文的过程

一个分组长为 n 比特、密钥长为 t 比特的分组密码,数学上可以看作是在 2^t 个密钥控制下的 $GF(2)^n \rightarrow GF(2)^n$ 的置换。而从 $GF(2)^n \rightarrow GF(2)^n$ 的置换有 $2^n!$ 个不同的方式。故一个极好的 n 比特的分组密码可以接受的密钥长度可达 $\log_2(2^n)!$ 比特,例如,$n = 8$时,可有近 1683 比特密钥;对 $n = 64$,近似 10^{21} 比特密钥。当然,我们不需要如此长的密钥,实用的密钥长一般为 128、196、256 比特。因而用来加密的置换只是全体置换所构成集合的一个子集。设计分组密码的问题,关键在于找到一种算法,能在密钥的控制下,从一个足够大且"好"的置换子集中,简单而迅速地选出一个置换。

8.2　分组密码的设计原则

针对安全性的一般设计原则:影响安全性的因素很多,诸如分组长度 n 和密钥长度 t 等,但有关实用密码的两个一般设计原则是 Shannon 提出的混乱原则和扩散原则。

混乱是指人们所设计的密码应使得密钥和明文以及密文之间的依赖关系相当复杂,以至于这种依赖性对密码分析者来说是无法利用的。

扩散是指人们所设计的密码应使得密钥的每一位数字影响密文的许多位数字,以防止对密钥进行逐段破译,而且明文的每一位数字也应影响密文的许多位数字,以便隐蔽明文数字统计特性。

针对实现的设计原则:分组密码可以用软件和硬件来实现。硬件实现的优点是可获得高速率,而软件实现的优点是灵活性强、代价低。基于软件和硬件的不同性质,分组密码的设计原则可根据预定的实现方法来考虑。

软件实现的设计原则:使用子块和简单的运算。密码运算在子块上进行,要求子块的长度能自然地适应软件编程,比如 8、16、32 比特等。在软件实现中,按比特置换是难于实现的,因此我们应尽量避免使用它。子块上所进行的一些密码运算应该是一些易于软件实现的运算,最好是用一些标准处理器所具有的一些基本指令,比如加法、乘法和移位等。

硬件实现的设计原则:加密和解密可用同样的器件来实现。尽量使用规则结构,因为密码应有一个标准的组件结构以便其能适应于用超大规模集成电路实现。

前面提到的乘积密码是实现 Shannon 提出的混乱原则和扩散原则的一种有效的方法,DES 就是一种这样的乘积密码。

当然,以上的原则是非常概括的,离我们构造安全的分组密码还差得远。下面一些原则也是我们常常需要考虑的。

简单性原则:包括规范的简单性和分析的简单性。规范的简单性,如果它仅采用了有限个运算,并且这些运算本身很容易解释,简单规范的明显优点是便于正确实现;另一个优点是人们在研究密码时,似乎对具有简单规范的密码算法更感兴趣。分析的简单性,好处是便于阐述和理解密码算法以何种方式来抗击已知类型的密码分析,这样,在设计阶段就开始考虑抵抗已知攻击,从而在设计之初就提供了一定程度的密码可信度。规范的简单性并非意味着分析的简单性,提出一个描述简单而已知攻击手段又难以分析的密码算法是相对容易的。Rijndael 算法作为 AES 的中标者,其中原因之一是分析起来简单。

必要条件:分析水平是不断进步的,所以设计者不仅要熟悉现存的各种攻击方法,而且要预想到一些未知的攻击。显然,设计一个安全的分组密码起码的要求是:它必须能抗击所有已知的攻击,特别是差分攻击和线性攻击。

可扩展性:AES 的许多候选算法要求能提供 128、192、256 比特的可变分组或密钥长,这样才能灵活适应多级安全需要,这一点在设计上也应考虑到。

8.3 分组密码的结构

如 8.2 节所讨论的,一个分组密码既要难于分析——“复杂”,又要易于实现——“简单”。迭代密码就是为了克服这一对矛盾而产生的一种分组密码,其加密变换(置换)一般采取如下结构:由一个简单的函数 F(易于实现)迭代若干次而形成,如图 8.2 所示。

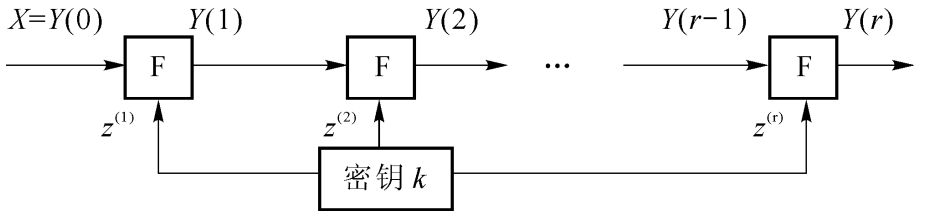


图 8.2　以轮函数 F 构造的迭代密码

其中 $Y(i-1)$是第 i 轮置换的输入，$Y(i)$是第 i 轮的输出，$z^{(i)}$是第 i 轮的子密钥，k 是种子密钥。每次迭代称为一轮，每轮的输出是输入和该轮子密钥的函数，每轮子密钥由 k(称为种子密钥)导出，这种密码就是迭代密码，如 DES 就是 16-轮迭代密码。函数 F 称为圈函数或轮函数，一个适当选择的圈函数通过多次迭代可实现必要的混淆和扩散。

如果把一个 $GF(2)^n$ 到 $GF(2)^m$ 的变换看做一个网络的话，常用的圈函数 F 都是基于代换——置换的网络，即以多次变换的乘积构成，置换的变换可提供扩散作用，而代换的变换可提供混淆作用，其中代换网络是精心设计且起关键作用的，人们常称其为黑盒子。为了增强安全性，n 一般都比较大，在代换的实现中，其难度将随 n 指数增长，而难于处理，不易实现，因此，实际中常将 n 划分成一些较短的段，如将 n 分成长为 n_0 的 r 个段。将设计 n 长变换的“黑盒子”简化为设计 r 个较小的子代换网络，大大降低了实现的难度。这些称为子代换盒，简称 S-盒，如 DES 中有 8 个 S-盒。S-盒的设计是分组密码设计的核心，其遵循的准则是保证整个密码系统安全性的关键所在。以上描述了在一个分组密码设计中为了实现既“复杂”(为了安全)，又“简单”(为了实现方便)而采取的典型结构形式。DES 和 AES 体制是这种结构的典型代表。

进一步，分组密码又采用两种类型的总体结构：Feistel 网络与 SP 网络，它们的主要区别在于：SP 结构每轮改变整个数据分组，而 Feistel 密码每轮只改变输入分组的一半。AES 和 DES 分别是这两种结构的代表。Feistel 网络(又称 Feistel 结构)可把任何轮函数转化为一个置换，它是由 Horst Feistel 在设计 Lucifer 分组密码时发明的，并因 DES 的使用而流行，“加解密相似”是 Feistel 型密码的实现优点。SP 网络(又称 SP 结构)是 Feistel 网络的一种推广，其结构清晰，S 一般称为混淆层，主要起混淆作用，P 一般称为扩散层，主要起扩散作用。SP 网络与 Feistel 网络相比，可以得到更快速的扩散，不过，SP 网络的加解密通常不相似。有关这两种结构的特点，读者在了解了 DES 和 AES 算法之后再细细体会。

8.4　分组密码的安全性

安全性仍是分组密码最重要的设计准则，它要求即使攻击者知道分组密码的内部结构，仍不能破译该密码，这也意味着，不存在针对该密码的某种攻击方法，其工作量小于穷

举密钥搜索。对于一个 r 轮的分组密码,如果存在一个对 $r-k$ 轮简化版本的密码分析攻击,那么称该密码具有 k 轮绝对安全冗余。密码分析技术的发展,使得对于具有更多轮的分组密码的破译成为可能。因此,安全冗余也意味着即使对现有的密码分析手段进行改进,该密码仍然能够抗击此类攻击。

8.4.1 安全需求

分组密码提供一种把 N 比特长的明文消息转换为 N 比特长的密文消息的方法,其中,加密操作是由长为 k 比特的秘密密钥串决定的,密钥常常是随机选择的;解密是加密的逆操作,用同样的密钥把 N 比特的密文还原成明文。在考虑安全性时,我们还是沿用 Kerchhoff 假设:除了密钥之外,攻击者知道所有有关加解密的详细过程。所以,直观的要求是:如果不知道密钥的知识,攻击者从密文恢复出明文是实际不可能的。

对一个固定的 k 比特密钥,如果一个分组长为 N 比特的分组密码只用来加密 $\left\lfloor \frac{k}{N} \right\rfloor$ 个明文,则对每个明文比特来说,每个密钥比特只使用一次,这种情况即为“一次一密”制,明文和密文是统计独立的,分组密码是绝对安全的。然而,在大多数的情况下,一次一密是不实际的,因此,用同样固定的 k 比特的密钥,加密 T 个明文,其中 $T\leqslant \left\lfloor \frac{k}{N} \right\rfloor$,最大化 T 后,仍能达到一个可接受的安全性是现代分组密码追求的目标。

既然是一个分组长为 N,密钥长为 k 的分组密码,那么可以看作对每一个可能的密钥定义了一个 2^N 个元素上的置换。一个简单的带密钥的随机查表(实现一个 N 比特的置换),不用附加任何算法,将实现一个非常强的 N 比特分组密码;不幸的是,这样一个实现将花费太多的存储,意味着一个巨大的实现复杂性,不能用作任何实际的用途。当今的几乎所有分组密码都用更小的查表(代替盒或 S 盒)合并其他变换(线性变换)模仿这样一个大的随机查表,这种做法实际上是安全性和可接受的复杂性的一个折中。

8.4.2 安全模型

对于一个使用的分组密码,几乎很难对它的安全性给出一个精确的测量。然而,还是有许多可以接受的安全模式。

1.无条件安全性

Shannon 假定,攻击者具有无限的计算资源。在这种模式下,只有当密钥大小与明文大小一样(即一次一密制)时,安全的加密才存在。每个固定的密钥只能用来加密 $\left\lfloor \frac{k}{N} \right\rfloor$ 个明文消息,所以无条件安全性对实用的分组密码来说不是一个有用的模式。

2.多项式安全性

与无条件安全性形成鲜明对照,当代密码学假定攻击者的计算资源是受限的。具体地讲,假定攻击者有一个运行多项式时间的概率算法,根据攻击密码系统的可行性来考虑

密码系统的安全性。这个模式起源于复杂性理论,基于攻击者仅有多项式的计算资源的假定。该模式常常考虑最坏的情况,渐近地分析确定一个密码的多项式攻击是否存在,然而,即使这种攻击存在,仍不能保证这种攻击是实际可操作的。

3.“可证明”安全性

一般这意味着两件事情:第一,如果能证实破译一个分组密码与解决某个众所周知的困难问题(如离散对数或大数分解问题)一样困难,则可认为该密码是安全的。当然,“困难问题”常常不一定是困难的,因为这些困难问题是否属于 P 或 NP 在计算机科学中仍然是一个未解决的问题。事实上,这种可证明安全性需要证实 $P \neq NP$ 和单向函数的存在性。第二,一个分组密码可被证明抗击某种已知的攻击。典型的例子是某分组密码可证实抗击差分和线性分析。然而,应该强调这并不意味着密码可以抗击所有的攻击,所以我们把‘可证明’加上引号,以免被误解。

4.实际的安全性

在这种模式下,如果一个最好的攻击方法需要的计算资源超出安全边界太多,则可认为该密码是实际安全的。对不同的已知攻击,我们可以分别测试其所需的时间和空间资源,然后估计其强度。在对具体的密码分析时,这种模式提供更多的回答,显然,应该注意到它不能回答抗未知攻击的能力。

5.历史的安全性

根据密码吸引的注意力的多少来估计密码安全水平是相当有用的。例如密码 A 和密码 B 可能都被认为设计很好,但密码 A 有十年的历史,曾被详细审查过且没有发现严重的漏洞,因此安全性上 A 密码更具优势。然而,也应该注意到,一个密码的强度并不总是依据其存在的时间来测量。

8.4.3　分组密码作为一个伪随机置换

考察一个分组密码作为一个置换的集合是自然的事情。为定义分组密码安全性,Luby 和 Rackoff 提出的基于伪随机函数发生器作为轮函数的三轮 Feistel 密码是伪随机置换的证据,因此是可证明抗选择明文攻击,四轮 Feistel 密码是抗自适应选择明、密文攻击。如果可证明一个分组密码渐近等价于伪随机函数或伪随机置换的集合,则可认为该密码是安全的。换句话说,我们无法区分输出密文比特和某个随机输出。理论上可以说:依赖于可证明安全的伪随机函数发生器,若不知密钥,在多项式时间内无法与真正的随机置换相区分。有关可证明安全的伪随机置换的构造请参看第 6.5 节。

伪随机意味着在多项式时间内,加密询问使得没有攻击者可以区分分组密码和一真实的随机置换,相应于选择明文攻击。

超伪随机意味着在多项式时间内,加、解密询问使得没有攻击者可以区分分组密码和一真实的随机置换,相应于选择明、密文攻击。

由超伪随机性可推出伪随机性,如 3-轮 DES 是一个伪随机置换,4-轮 DES 是一个超

伪随机置换。

分组密码也可以与单向函数联系起来。给定一个 N 比特明文消息 m,一个 K 比特的密钥 k,加密函数 E,产生一个 N 比特的密文输出消息 c,其中 $c = E(k, m)$,如果固定 m_0,可定义 $f(k) = E(k, m_0)$。Luby 和 Rackoff 证实,如果 E 是伪随机置换的集合,则 f 近似于一个单向函数。然而单向函数概念比安全的分组密码要弱,例如,泄漏一个单向函数输入的一半,其仍然是一个单向函数。为了构造安全的分组密码,还需要引进一个函数“硬核比特”的想法(参见第 5.4 节)。

8.4.4 攻击的分类

如前所述,大多数分析方法是在实际安全模式下进行的。我们常常根据所需的时间、存储、数据复杂性来测量一个攻击成功的水平。对一个 N 比特的分组密码,下列复杂度应被考虑:

(1) 数据复杂度:实施一个攻击所需输入的数据量,用长为 N 的分组作单位;

(2) 处理复杂度:执行一个攻击所花的时间,时间单位可取攻击者不得不亲自做的加密次数;

(3) 存储复杂度:需要记忆的字,单位可取长为 N 的分组。

按照拇指规则,攻击复杂度取三者之中最大的。一般说,这个规则还有一些例外,而且还要考虑相关攻击者的环境。例如,有些预处理尽管所花的时间很多,但攻击者可以负担;有时候加密在专用硬件上很快地完成,这时处理复杂度可以不考虑。一般地,目前存储的需要对大多数攻击者来说都不太现实,所以认为存储比时间更昂贵。

具体地讲,分组密码的攻击方法目前有穷举密钥搜索法、差分分析、截断差分分析、不可能差分分析、高阶差分分析、线性分析、差分线性分析,Boomerang 攻击、相关密钥攻击、插值攻击、非双射攻击、Slide 攻击、χ^2 攻击。

下面介绍穷举密钥搜索攻击。

该方法适用于所有的分组密码,攻击者只需几个已知的明-密文对。攻击者尝试所有的密钥,检查是否能把给定的明文加密成给定的密文,试验成功即可得到密钥。

差分和线性分析将在第 8.6 节详细叙述。读者若对其他分析方法感兴趣,请参看本书参考文献[23]。

8.5 典型的分组密码算法——DES

DES 是由 IBM 公司在 1971 年设计出的一个加密算法。DES 在 1977 年经过美国国家标准局(NBS)采用为联邦标准(FIPS PUB 46)之后,已成为金融界及其他各种行业最广泛应用的对称密钥密码系统。DES 是分组密码的典型代表,也是第一个被公布出来的

标准算法。1977 年,美国正式公布美国数据加密标准——DES,并广泛用于商用数据加密,算法完全公开,这在密码学史上是一个创举。计算机通信网络发展对信息安全保密的需求日益增长,大量敏感和机密信息、数据的传输和存储都要求有密码保护,为了实现同一水平的安全性和兼容性,而提出了数据加密标准化,DES 就是这种需求的产物。

二十多年来,尽管计算机硬件及破解密码技术的发展日新月异,若撇开 DES 的密钥太短,易于被使用穷举密钥搜寻法找到密钥的攻击法不谈,目前所知攻击法,如差分攻击法或是线性攻击法,对于 DES 的安全性也仅仅做到了“质疑”的地步,并未从根本上破解 DES。换言之,若是能用类似 Triple - DES 或是 DESX 的方式加长 DES 密钥长度,仍不失为一个安全的密码系统。

DES 仍是迄今为止世界上最为广泛使用和流行的一种分组密码算法。美国政府已经征集评估和判定出了新的数据加密标准 AES 以取代 DES,但 DES 对现代分组密码理论的发展和应用起了奠基性作用,它的基本理论和设计思想仍有重要参考价值。下面简要描述 DES 算法。

8.5.1　算法描述

DES 是对二元数字分组加密的分组密码算法,分组长度为 64 比特。每 64 位明文加密成 64 位密文,没有数据压缩和扩展,密钥长度为 56 比特,若输入 64 比特,则第 8、16、24、32、40、48、56、64 为奇偶检验位,所以,实际密钥只有 56 位。DES 算法完全公开,其保密性完全依赖密钥。

图 8.3 是 DES 全部 16 轮(Round)的加/解密结构图,其最上方的 64 比特输入分组数据,可能是明文,也可能是密文,视使用者要做加密或解密而定。而加密与解密的不同处,仅在于最右边的 16 个子密钥的使用顺序不同,加密的子密钥顺序为 $K_1, K_2, \cdots, K_{16}$,而解密的子密钥顺序正好相反,为 $K_{16}, K_{15}, \cdots, K_1$。

DES 算法首先对输入的 64 位明文 X 进行一次初始置换 IP(见图 8.4),以打乱原来的次序。对置换后的数据 X_0 分成左右两半,左边记为 L_0,右边记为 R_0,对 R_0 施行在子密钥控制下的变换 f,其结果记为 $f(R_0, K_1)$,得到的 32 比特输出再与 L_0 做逐位异或(XOR)运算,其结果成为下一轮的 R_1,R_0 则成为下一轮的 L_1。对 L_1,R_1 施行和 L_0,R_0 同样的过程得 L_2,R_2,如此循环 16 次,最后得 L_{16},R_{16}。再对 64 位数字 R_{16},L_{16} 施行初始置换的逆置换 IP^{-1}(见图 8.4),即得密文 Y。运算过程可用公式(式 8.1)简洁地表示如下:

$$\begin{aligned} R_i &= L_{i-1} \quad f(R_{i-1}, K_i), \\ L_i &= R_{i-1}, \qquad i = 1,2,\cdots,16。 \end{aligned} \tag{8.1}$$

读者注意,在 16 次加密后并未交换 L_{16},R_{16},而直接将 R_{16},L_{16} 作为 IP^{-1} 的输入,这样做使得 DES 的解密和加密完全相同,在以上过程中只需输入密文并反序输入子密钥,最后获得的就是相应的明文。

以上是对 DES 加解密过程的描述。我们把从 $L_{i-1}R_{i-1}$，到 L_iR_i 的变换过程称为一轮加密，所以 DES 要经过 16 轮加密，或称为 16 轮迭代，每一轮施行的变换完全相同，只是每轮输入数据不同。

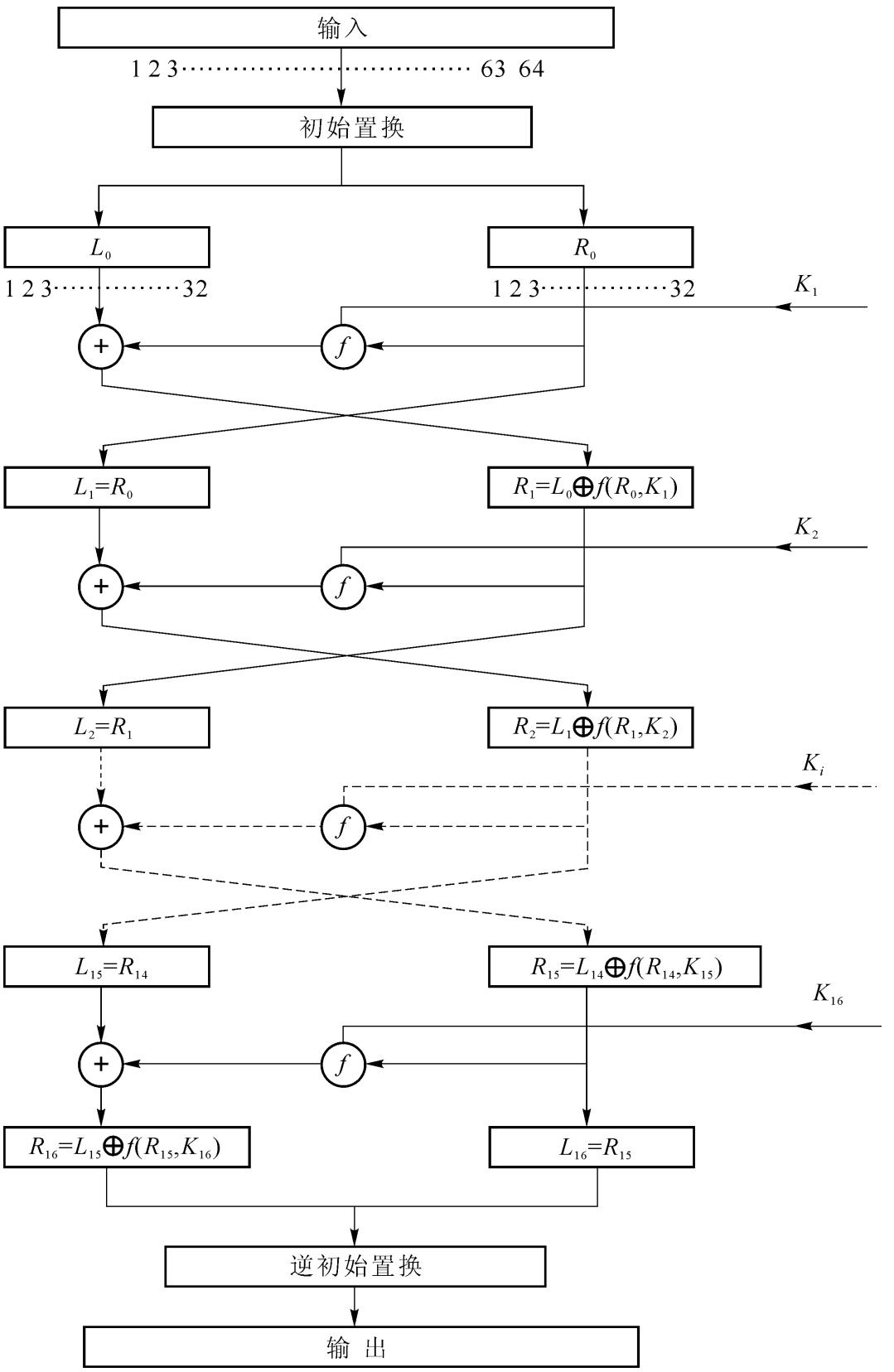


图 8.3　DES 加/解密流程

初始置换 IP 及其逆置换 IP^{-1} 并没有密码学意义，因为 X 与 $IP(X)$（或 Y 与 $IP^{-1}(Y)$）的一一对应关系是已知的，如 X 的第 58 比特是 $IP(X)$ 的第 1 比特，X 的第 50 比特是

IP(*X*)的第 2 比特等等。它们的作用在于打乱原来输入 *X* 的 ASCII 码字划分的关系,并将原来明文的第 x_8, x_{16}, …, x_{64}位(校验位)变成 IP 的输出的一个字节。

IP

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

IP^{-1}

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

图 8 .4　初始置换 IP 及逆初始置换 IP^{-1}

f 函数是整个 DES 加密法中最重要的部分,而其中的重点又在 S-盒(Substitution Boxes)上。*f* 函数可记作 *f*(*A*, *J*),其中 *A* 为 32 位输入, *J* 为 48 位输入,在第 *i* 轮 $A = R_{i-1}$, $J = K_i$, K_i 为由初始密钥(亦称种子密钥)导出的第 *i* 轮子密钥, *f*(*A*, *J*)输出为 32 比特。

f(*A*, *J*)的计算过程如下:

将 *A* 经过一个选择扩展运算 *E*(见图 8 .5)变为 48 位,记为 *E*(*A*)。计算 *E*(*A*)　*J*

E

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

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

图 8 .5　扩展运算 *E* 与置换 *P*

= *B*,对 *B* 施行代换 S,此代换由 8 个代换盒组成,就是前面说过的 S-盒。每个 S-盒有 6 个输入,4 个输出,将 *B* 依次分为 8 组,每组 6 位,记 $B = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8$,其中 B_j 作为第 *j* 个 S-盒 S_j 的输入, S_j 的输出为 C_j, $C = C_1\ C_2\ C_3\ C_4\ C_5\ C_6\ C_7\ C_8$ 就是代换 S 的输出,所以代换 S 是一个 48 位输入,32 位输出的选择压缩运算,将结果 *C* 再施行一个置换 *P*(见图 8 .5),即得 *f*(*A*, *J*)。其中在第 *i* 轮为 $f(R_{i-1}, K_i)$。 *f*(*A*, *J*)可用图 8 .6 表示。

其中,扩展运算 *E* 与置换 *P* 主要作用是增加算法的扩散效果,具体运算如图 8 .6 所示。

S-盒是 DES 算法中惟一的非线性部件,当然也就是整个算法的安全性所在。它的设

计原则与过程一直因为种种不为人知的因素所限,而未被公布出来。有些人甚至还大胆猜测,是否设计者故意在 S-盒的设计上留下了一些陷门(Trapdoor),以便他们能轻易地破解出别人的密文,当然以上的臆测是否属实,迄今仍无法得知,不过有一点可以确定,那就是 S-盒的设计的确相当神秘。

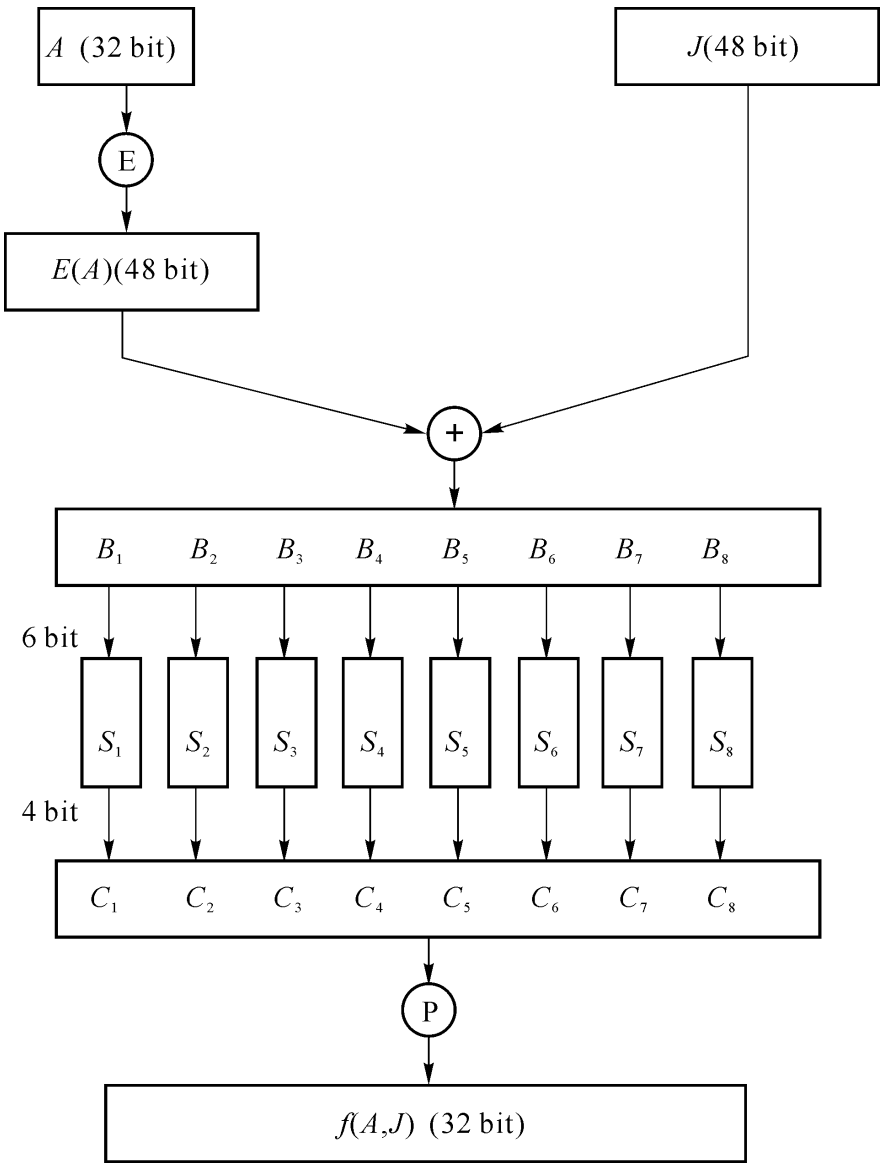


图 8 .6　f 函数运算框图

每个 S-盒是有 6 个输入、4 个输出的变换,其变换规则为:取{0,1,…,15}上的 4 个置换,即它的 4 个排列排成 4 行,得一 4×16 矩阵。若给定该 S-盒的输入 $b_0\ b_1\ b_2\ b_3\ b_4\ b_5$,其输出对应该矩阵第 L 行 n 列所对应的数的二进制表示。这里 L 的二进制表示为 $b_0\ b_5$,n 的二进制表示为 $b_1\ b_2\ b_3\ b_4$,这样,每个 S-盒可用一个 4×16 矩阵或数表来表示。8 个 S-盒的表示可用表 8 .1 给出。

表 8 .1　S-盒 (S-Boxes)

行																
列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	

续表

行																
列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	1	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

密钥方案的计算:子密钥产生过程(见图 8 .7)中的输入,为使用者所持有的 64 比特初始密钥。在加密或解密时,使用者先将初始密钥输入至子密钥产生流程中即可。首先

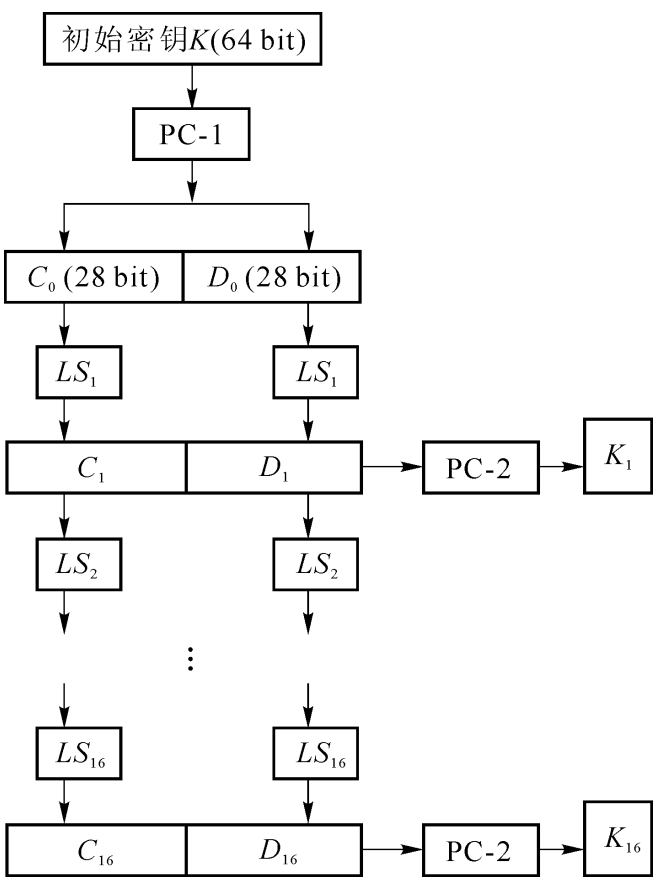


图 8 .7　子密钥的产生过程

经过密钥置换 PC-1(见表 8 .2),将初始密钥的 8 个奇偶校验位剔除掉,而留下真正的 56 比特初始密钥。接着兵分两路为两个 28 比特的分组 C_0 及 D_0,再分别经过一个循环左移函数 LS_1,得到 C_1 与 D_1,连成 56 比特数据,再依密钥置换 PC-2(见表 8 .3)做重排动作,

便可输出子密钥 K_1,而 K_2 至 K_{16}的产生方法,依此类推。其中需要注意的是:置换 PC-1 的输入为 64 比特,输出为 56 比特;而密钥置换 PC-2 的输入和输出分别为 56 和 48 比特。

表 8 2　密钥置换 PC-1

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

表 8 3　密钥置换 PC-2

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

对每个 i,$1 \leq i \leq 16$,计算 $C_i = LS_i(C_{i-1})$, $D_i = LS_i(D_{i-1})$, K_i = PC-2$(C_i D_i)$,其中 LS_i表示一个或两个位置的左循环移位,当 $i = 1,2,9,16$ 时,移一个位置,当 $i = 3,4,5,6,7,8,10,11,12,13,14,15$ 时,移两个位置。

8 5 2　DES 的设计思想和特点

DES 综合应用了置换、代替、移位多种密码技术,是一种乘积密码。在算法结构上采用迭代结构,从而使结构紧凑,条理清楚,而且算法为对合运算,便于实现。DES 使用了初始置换 IP 和逆初始置换 IP^{-1} 各一次,置换 P 16 次,安排使用这三个置换的目的是把数据彻底打乱重排,它们在密码意义上作用不大,因为它们与密钥无关,置换关系固定,一旦公开后便无多大密码意义了。扩展运算 E 一方面把数据打乱重排,另一方面把 32 位输入扩展为 48 位。算法中除了 S-盒是非线性变换外,其余变换均为线性变换,所以保密性的关键是选择 S-盒。

S-盒是经过精心设计和严格挑选的,美国国家安全局(NSA)曾经确认过下列 3 条“设计准则”:

(1) 对任意一个 S-盒而言,没有任何线性方程式等价于此 S-盒的输出/输入关系。即是说,S-盒是非线性函数:

(2) 改变 S-盒的任何一位的输入,则至少有两个以上的输出位会因此而有所改变。换句话说,任一输入位可以影响的输出位愈多愈好。

(3) 当固定某一个位的输入时,我们希望 S-盒的 4 个输出位之间,其“0”和“1”个数之差愈小愈好。

这个非线性变换的本质是数据压缩,它把 6 位输入压缩为 4 位输出,并且若 S-盒函数的输入中任意改变数位,其输出至少变化 2 位。因为算法中使用了 16 次迭代,从而使得即使是改变明文或密钥中的 1 位,密文都会发生约 32 位的变化,大大提高了保密性。DES 的子密钥产生与使用上也很有特色,它确保了原密钥中各位的使用次数基本上相

等。试验表明,56 位密钥的每位的使用次数在 12～15 次之间,这也使保密性得到进一步提高。

DES 在总的方面是极其成功的,但同时也不可避免地存在着一些弱点和不足:

(1) 存在一些弱密钥和半弱密钥

在 16 次加密迭代中分别使用不同的子密钥是确保 DES 强度的一种重要措施,但由于子密钥产生过程的设计不当,实际上却存在着一些密钥,由它们产生的 16 个子密钥不是互不相同,而是有相重合的,称这些密钥为弱密钥或半弱密钥。称使 16 个子密钥全相同的密钥为弱密钥,称使 16 个子密钥中有部分相同的密钥为半弱密钥。

弱密钥的使用会降低 DES 的安全性。若 k 为弱密钥,则下列关系式成立:

$$E_k(E_k(m)) = m \text{ 及 } D_k(D_k(m)) = m。$$

但由于弱密钥和半弱密钥的数量与密钥的总数相比仍是微不足道的,所以这并不构成对 DES 的太大威胁,只要在实际应用中避免使用这些密钥即可。

(2) 存在互补对称性

在 DES 的明文 m,密文 c 与密钥 k 之间存在着互补的特性。此互补性,简单地说,可以用下列两个式子表示:若 $E_k(m) = c$,则 $E_{\bar{k}}(\overline{m}) = \overline{c}$。

这个关系是说,如果以密钥 k 对明文 m 加密,得到密文 c,则相对地,以密钥 $\overline{k}$ 对明文 $\overline{m}$ 加密,亦可得到 $\overline{c}$。其中 $\overline{X}$ 表示 X 逐位取补。

这个性质使得非法者有机可乘,假设破译者 A 要破解使用者 B 的密钥 k,而且 A 又拥有 B 使用密钥 k 对明文 m 及 $\overline{m}$ 加密的密文 $E_k(m)$及 $E_k(\overline{m})$,则 A 可利用 DES 的互补性来找出密钥 k;这比穷举密钥搜索法少花了一半的时间(时间复杂度为 2^{55})。

尽管如此,在实际上却不太可行。因为两个明文互为补码的概率相当小,所以破译者获得 $E_k(m)$及 $E_k(\overline{m})$也相当困难,因此,这种互补性不能算是 DES 的漏洞。

8.5.3 DES 的工作模式(对其他分组密码也适用)

实际应用中,DES 是根据其加密算法所定义的明文分组的大小(64 比特),将数据分割成若干 64 比特的加密区块,再以加密区块为单位,分别进行加密处理。如果最后剩下不足一个区块的大小,我们称之为短块,关于短块的处理方法一般有填充法、序列密码加密法、密文挪用技术[18]。根据数据加密时每个加密区块间的关联方式来区分,可以分为 4 种加密模式,包括 ECB(Electronic Code Book)、CBC(Cipher Block Chaining)、CFB(Cipher Feedback)及 OFB(Output Feedback)。

1.电码本模式(ECB)

ECB 模式是分组密码的基本工作模式,图 8.8 为 ECB 加密模式示意图。

在 ECB 模式下,每一个加密区块依次独立加密,产生独立的密文区块,每一加密区块的加密结果均不受其他区块的影响,使用此种方式下,可以利用并行处理来加速加解密运算,且在网络传输时任一区块有任何错误发生,也不会影响到其他区块传输的结果,这是

该模式的优点。

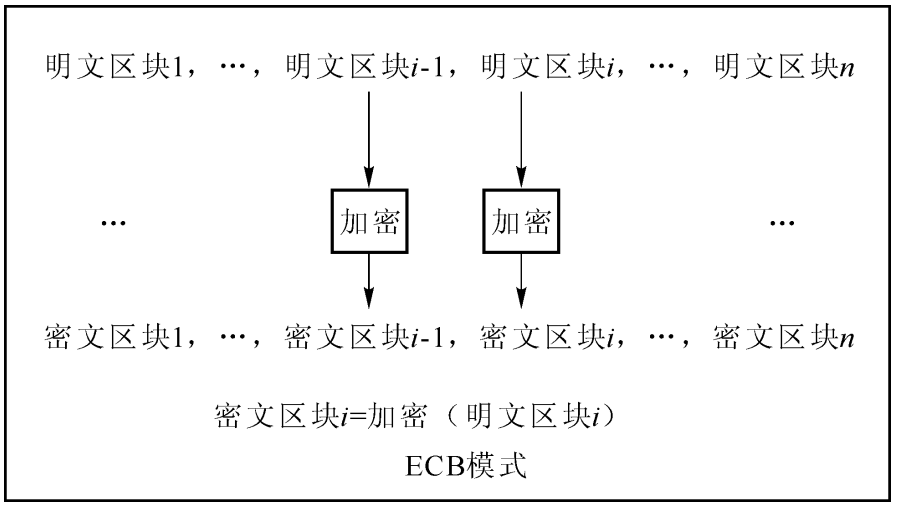


图 8 .8　ECB 加密模式示意图

ECB 模式的缺点是容易暴露明文的数据模式。在计算机系统中,许多数据都具有固有的模式,这主要是由数据结构和数据冗余引起的,如果不采取措施,对于在要加密的文件中出现多次的明文,此部分明文若恰好是加密区块的大小,可能会产生相同的密文,且密文内容若遭剪贴、替换,也不易被发现。

2 .密码分组链接模式(CBC)

图 8 .9 为 CBC 加密模式示意图。

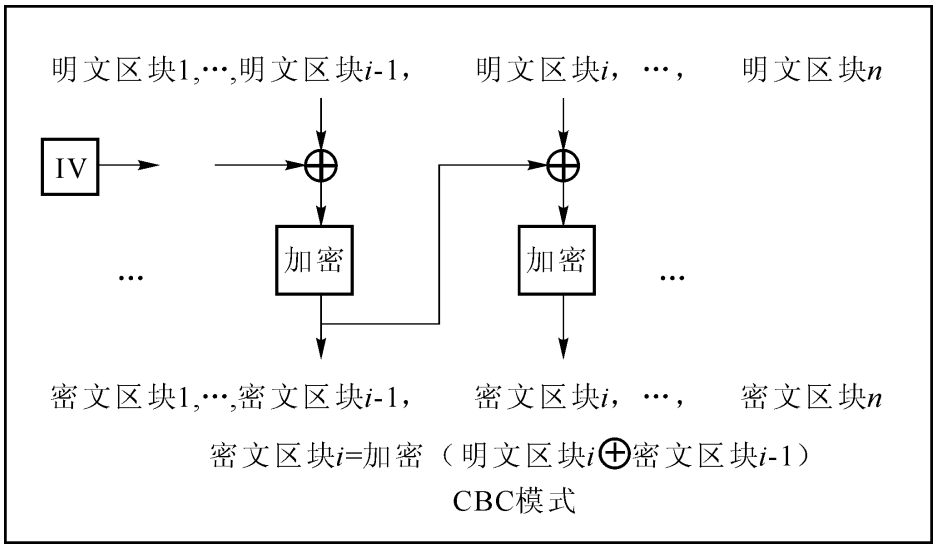


图 8 .9　CBC 加密模式示意图

第一个加密区块先与初始向量(IV: Initialization Vector)做异或(XOR)运算,再进行加密。其他每个加密区块加密之前,必须与前一个加密区块的密文作一次异或运算,再进行加密。每一个区块的加密结果均会受到前面所有区块内容的影响,所以即使在明文中出现多次相同的明文,也会产生不同的密文。

再者,密文内容若遭剪贴、替换,或在网络传输过程中发生错误,则其后续的密文将被

破坏,无法顺利解密还原,这是这一模式的优点,也是缺点。

其次,必须选择一个初始向量,用以加密第一个区块,且在加密作业时无法利用并行处理来加速加密运算,但其解密运算,因做异或的加密区块结果已存在,仍可以利用并行处理来加速。

3 .密文反馈方式(CFB)

图 8 .10 为 CFB 加密模式示意图。

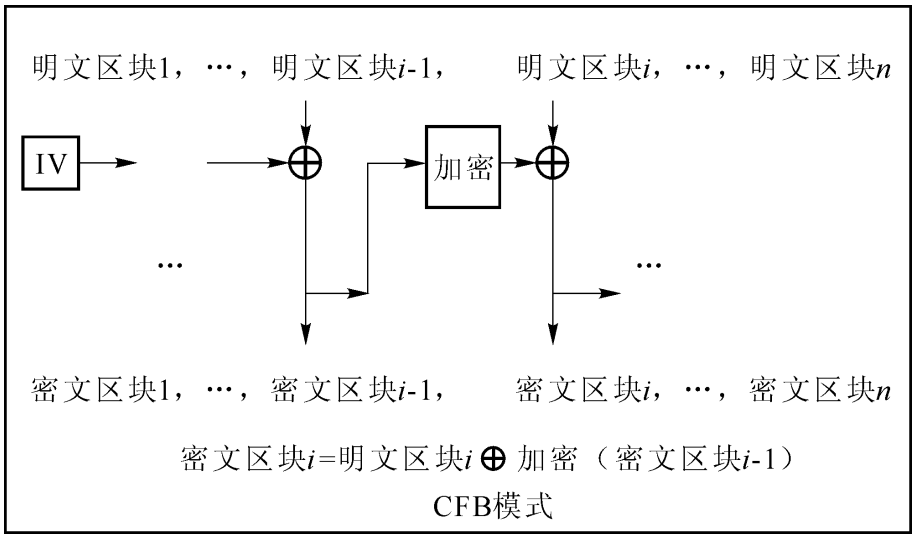


图 8 .10　CFB 加密模式示意图

可以将区块加密算法当作流密码加密器(Stream Cipher)使用,流密码加密器可以按照实际上的需要,每次加密区块大小可以自订(如每次 8 个位),每一个区块的明文与前一个区块加密后的密文做异或后成为密文。因此,每一个区块的加密结果也受之前所有区块内容的影响,也会使得在明文中出现多次相同的明文均产生不相同的密文。在此模式下,与 CBC 模式一样,为了加密第一个区块,必须选择一个初始向量,且此初始向量必须惟一,每次加密时必须不一样,也难以利用并行处理来加快加密作业。

4 .输出反馈模式(OFB)

图 8 .11 为 OFB 加密模式示意图。

输出反馈模式与 CFB 大致相同,都是每一个区块的明文与之前区块加密后的结果作异或后产生密文,不同的是之前区块加密后的结果为独立产生,每一个区块的加密结果不受之前所有密文区块的内容的影响,如果有区块在传输过程中遗失或发生错误,将不至于无法完全解密,在此模式下,为了加密第一个区块,必须设置一个初始向量,否则难以利用并行处理来加快加密作业。

容易看出,这 4 种操作模式有不同的优点和缺点。在 ECB 和 OFB 模式中改变一个明文块将引起相应的密文块的改变,而其他密文块不变,有些情况下这可能是一个好的特性。例如,OFB 模式通常用来加密卫星传输。

另一方面,在 CBC 和 CFB 模式中改变一个明文块,那么相应的密文块及其后的所有密文块将会改变,这个特性意味着 CBC 和 CFB 模式适用于鉴别的目的。更明确地说,这

些模式能用来产生消息鉴别码(MAC)(参见第 11 章),MAC 附在明文块序列的后面,用来保护消息的完整性。

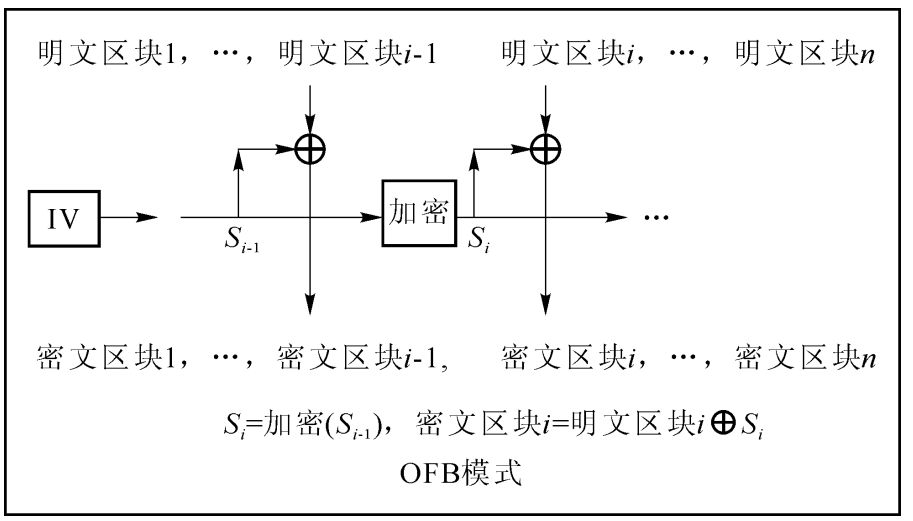


图 8 .11　OFB 加密模式示意图

8 5 4　DES 的实现

DES 正式颁布后,世界各国的许多公司都推出了自己实现 DES 的软硬件产品。虽然 DES 的描述是相当长的,但它能以硬件或软件方式非常有效地实现。需完成的算术运算仍为比特串的异或。扩展函数 E,S-盒,置换 IP 和 P 以及 16 个子密钥的计算都能在一个固定时间内通过查表(以软件)或电路中的硬件布线来完成。

现在的硬件实现能达到非常快的加密速度。数字设备公司在 Crypto '92 上宣布他们已经制造了带有 50 个晶体管的芯片,时钟速率 250 MHz 时,加密速度达 1 Gbit/ s,这个芯片的价格大约是 300 美元。到 1991 年,共有 45 个 DES 的硬件和微程序实现,得到了美国国家标准局的认可。

8 5 5　DES 的安全性

20 多年来,尽管计算机硬件及破解密码技术的发展日新月异,若撇开 DES 的密钥太短,易于被使用穷尽密钥搜索法找到密钥的攻击法不谈,目前所知攻击法,如差分攻击法或线性攻击法,对于 DES 的安全性也仅仅做到了“质疑”的地步,并未从根本破解 DES。换言之,若是能用类似 Triple - DES 或是 DESX 的方式加长 DES 密钥长度,仍不失为一个安全的密码系统。

由于目前尚不存在一个评价密码系统的统一标准和严格的理论,人们只能从一个密码系统抵抗现有的密码分析手段的能力来评价它的好坏。自 1975 年来,许多机构、公司和学者(包括美国国家安全局 NSA、美国国家标准与技术研究所 NIST、IBM 公司、Bell 实验室和一大批著名的密码学家),对 DES 进行了大量的研究与分析。

对 DES 的批评主要集中在以下几点:

(1) DES 的密钥长度(56 位)可能太小;

(2) DES 的迭代次数可能太少;

(3) S-盒中可能有不安全因素;

(4) DES 的一些关键部分不应当保密。

比较一致的看法是 DES 的密钥太短,密钥量仅为 2^{56}(约为 10^{17})个,不能抵抗穷尽密钥搜索攻击(所谓穷尽密钥搜索攻击是指攻击者在得到一组明文 - 密文对条件下,可对明文用不同的密钥加密,直到得到的密文与已知的明文 - 密文对中的相符,就可确定所用的密钥,也许有不只一个这样的密钥),事实证明的确如此。1997 年 1 月 28 日,美国的 RSA 数据安全公司在 RSA 安全年会上公布了一项“秘密密钥挑战”竞赛,分别悬赏 1000 美金、5000 美金和 1 万美金用于攻破不同密钥长度的 RC5,同时还悬赏 1 万美金破译密钥长度为 56 比特的 DES。RSA 发起这场挑战赛是为了调查 Internet 上分布式计算的能力,并测试不同密钥长度的 RC5 和密钥长度为 56 比特的 DES 的相对强度。美国克罗拉多州的程序员 Verser 从 1997 年 3 月 13 日起,用了 96 天的时间,在 Internet 上数万名志愿者的协同工作下,于 6 月 17 日成功地找到了 DES 的密钥,获得了 RSA 公司颁发的 1 万美金的奖励。这一事件表明依靠 Internet 的分布式计算能力,用穷尽密钥搜索攻击方法破译已成为可能。1998 年 7 月,电子边境基金会(EFF)使用一台 25 万美金的电脑在 56 小时内破解了 56 比特的 DES。1999 年 1 月 RSA 数据安全会议期间,电子边境基金会用 22 小时 15 分钟就宣告完成 RSA 公司发起的 DES 的第三次挑战。

最有意义的分析技巧就是差分分析(在密码学中,“分析”和“攻击”这两个术语的含义相同,以后不加区别)。差分分析(differential cryptanalysis)由 Biham 和 Shamir 于 1991 年提出的选择明文攻击,可以攻击很多分组密码(包括 DES)。差分分析涉及带有某种特性的密文对和明文对比较,其中分析者寻找明文有某种差分的密文对。这些差分中的一些有较高的重现概率,差分分析用这些特征来计算可能密钥的概率,最后定位最可能的密钥。据说,这种攻击很大程度上依赖于 S-盒的结构,然而,DES 的 S-盒被优化可以抗击差分分析。尽管差分攻击比 DES 公布更迟,IBM 公司 Don Coppersmith 在一份内部报告中说:“IBM 设计小组早在 1974 年已经知道差分分析,所以设计 S-盒和换位变换时避开了它,这就是 DES 能够抵抗差分分析方法的原因。我们不希望外界掌握这一强有力的密码分析方法,因此这些年来我们一直保持沉默。”

轮数对差分分析有一个较大的影响。如果 DES 仅使用 8 轮,则在个人计算机上只需几分钟就可破译密码。在完全的 16 轮,差分分析仅比穷尽密钥搜索稍微有效。然而,如果增加到 17 或 18 轮,则差分分析攻击和穷尽密钥搜索攻击花费同样的时间。如果 DES 被增加到 19 轮,则穷尽密钥搜索攻击比差分分析更容易。这样,尽管差分分析是理论可破的,但因为须花费大量的时间和数据支持,所以并不实用,然而,差分分析攻击显示的是,对任何少于 16 轮的 DES,在已知明文攻击下比穷尽密钥搜索更有效。1993 年,Matsui 介绍了线性攻击(Linear Cryptanalysis),是一种已知明文攻击,用线性近似来描述分组

密码的行为。线性分析证明是比差分分析更有效。事实上,Matsui 在试验性条件下能恢复一个 DES 密钥,线性分析能用 2^{21} 个已知明文破 8 轮 DES,2^{43} 已知明文破译 16 轮 DES。有关差分分析和线性分析的原理和步骤见第 8.6 节。

如前所述,DES 已经达到它的信任终点。1997 年 4 月 15 日,美国国家标准技术研究所(NIST)发起征集 AES(Advanced Encryption Standards)算法的活动,目的是确定一个非保密的、公开披露的、全球免费使用的分组密码算法,用于保护 21 世纪政府的敏感信息,并希望能够成为秘密和公开部门的数据加密标准。

8.6　典型的分组密码的分析方法

分析的历史至少和密码学的历史一样长,1917 年"Scientific American"声称 Vigenère 密码是不可破的。今天演示这个结论的不真实,只是密码学上的一个练习。

密码分析与密码设计是相互对立、相互依存的。伴随着任何一种密码的设计,分析者便会千方百计地从该密码中寻找"漏洞"和缺陷进行攻击。对现代分组密码更是如此,如 DES 自从诞生以来,对它的分析工作一刻也没有停止过。归纳起来,对分组密码的分析方法主要有如下几种类型:

(1) 穷尽密钥搜索(强力攻击);
(2) 线性分析方法;
(3) 差分分析方法;
(4) 相关密钥密码分析;
(5) 中间相遇攻击。

尽管对分组密码的攻击方法没有像序列密码那样成熟和完善,比如,对 DES 的攻击至今没有完全破译的报道,仅限于 3 轮、6 轮,亦或有 8 轮,完全破译 16 轮似乎是不可能的。然而,这些分析方法的出现,毕竟对分组密码的安全性构成了威胁,也对分组密码的设计提出了更高的要求。差分和线性分析是两个被人们所熟知的分析分组密码的方法,任何一个新的密码的提出首先得经过他们的检验 。我们以 DES 为背景,介绍两种典型分析方法:差分分析和线性分析。

8.6.1　差分分析法

今天最为人所熟知的分析传统密码系统的方法是差分分析,1991 年由 Eli Biham 和 Adi Shamir 公布。差分分析的方法被证明是非常有效的,许多曾被认为强的密码,但都可破,差分分析应用到 DES, DDES, Lucifer, FEAL, LOKI, PES, Khafre, SAFER, RC5, IDEA。差分分析引起好几个密码的修改和重新设计。差分分析是普遍的,因为它可用来攻击任何由迭代一个固定的轮函数的结构的密码。

差分分析方法是一种选择明文攻击,其基本思想是通过分析特定明文差分对相应密文差分影响来获得可能性最大的密钥。

尽管差分分析方法对破译 16-轮的 DES 还不能提供一种实用的方法,但用它破译轮数较低的 DES 是很成功的,例如,8-轮 DES 在个人计算机上几分钟就可以破译。下面首先介绍差分分析的理论依据,然后给出对 3-轮 DES 进行差分攻击的一个简单的分析实例。

1.差分分析的理论依据

因为 DES 中的初始置换 IP 及其逆置换 IP^{-1} 是公开的,所以为了方便起见,可以忽略掉初始置换 IP 及其逆置换 IP^{-1},这并不影响分析。人们一般说的 DES 是 16-轮 DES,实际上,它可以扩展为任意轮 DES。这里只考虑 n-轮 DES, $n \leq 16$。在 n-轮 DES 中,将 $L_0 R_0$ 视作明文, $L_n R_n$ 是密文(注意,这里也没有交换 L_n 和 R_n 的位置)。差分分析的基本观点是比较两个明文的异或与相应的两个密文的异或。一般地,将考虑两个具有确定的异或值 $L_0' R_0' = L_0 R_0 \quad L_0^* R_0^*$ 的明文 $L_0 R_0$ 和 $L_0^* R_0^*$。

定义 8 2 设 S_j 是一个给定的 S-盒$(1 \leq j \leq 8)$, (B_j, B_j^*) 是一对长度为 6 比特的串。称 S_j 的输入异或是 $B_j \quad B_j^*$, S_j 的输出异或是 $S_j(B_j) \quad S_j(B_j^*)$。

对任何 $B_j' \in Z_2^6$,记 $\Delta(B_j') = \{(B_j, B_j^*) \mid B_j \quad B_j^* = B_j'\}$。易知, $|\Delta(B_j')| = 2^6 = 64$, 且 $\Delta(B_j') = \{(B_j, B_j \quad B_j') \mid B_j \in Z_2^6\}$。对 $\Delta(B_j')$ 中的每一对,都能计算出 S_j 的一个输出异或,共可计算出 64 个输出异或,它们分布在 $2^4 = 16$ 个可能的值上。将这些分布列成表,其分布的不均匀性将是差分攻击的基础。

例 8 .1 设第一个 S-盒 S_1 的输入异或为 110100,那么 $\Delta(110100) = \{(000000, 110100), (000001, 110101), \cdots, (1111111, 001011)\}$。现在对集合 $\Delta(110100)$ 中的每一个有序对,计算 S_1 的输出异或。例如, $S_1(000000) = E_{16} = 1110$, $S_1(110100) = 9_{16} = 1001$, 所以有序对(000000, 110100)的输出异或为 0111。对 $\Delta(110100)$ 中的每一对,都做这样的处理后,可获得下列的输出异或分布:

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12

1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

在例 8 .1 中,16 个可能的输出异或中实际上只有 8 个出现。一般地,如果固定一个 S-盒 S_j 和一个输入异或 B_j',那么平均来讲,所有可能的输出异或中实际上出现大约 75%～80%。

对 $1 \leq j \leq 8$, 长度为 6 比特的串 B_j' 和长度为 4 比特的串 C_j', 定义

$$IN_j(B_j', C_j') = \{B_j \in Z_2^6 \mid S_j(B_j) \quad S_j(B_j \quad B_j') = C_j'\},$$

$$N_j(B_j'C_j') = |IN_j(B_j', C_j')|,$$

$N_j(B_j', C_j')$ 表示对 S-盒 S_j 具有输入异或为 B_j' 输出异或为 C_j' 的对的数量。易知, $IN_j(B_j', C_j')$ 可分成 $N_j(B_j', C_j')/2$ 对, 使得每一对的异或为 B_j'。

在例 8.1 中的分布表由值 $N_1(110100, C_1')$, $C_1' \in Z_2^4$ 构成。集合 $IN_1(110100, C_1')$ 中的元素如表 8.4 所列。

表 8.4　具有输入异或 110100 的所有可能输入

输出异或	可能的输入
0000	
0001	000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011
0010	000100, 000101, 001110, 010001, 010010, 010100, 011010, 011011, 100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010
0011	000001, 000010, 010101, 100001, 110101, 110110
0100	010011, 100111
0101	
0110	
0111	000000, 001000, 001101, 010111, 011000, 011101, 100011, 101001, 101100, 110100, 111001, 111100
1000	001001, 001100, 011001, 101101, 111000, 111101
1001	
1010	
1011	
1100	
1101	000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010
1110	
1111	000111, 001010, 001011, 110011, 111110, 111111

对 8 个 S-盒中的每一个, 都有 64 个可能的输入异或, 所以共需计算 512 个分布。这些通过计算机很容易算出。

在第 i 轮, S-盒的输入可写作 $B = E \quad J$, 其中 $E = E(R_{i-1})$ 是 R_{i-1} 的扩展, $J = K_i$ 由第 i 轮的密钥比特构成。此时, 输入异或(对所有 8 个 S-盒)可通过下式计算:

$$B \quad B^* = (E \quad J) \quad (E^* \quad J) = E \quad E^*。$$

显然,输入异或不依赖于密钥比特 J。然而输出异或必定依赖于这些密钥比特。

将 B, E, J 均写成长为 6 的比特串的级联:

$$B = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8,$$

$$E = E_1\ E_2\ E_3\ E_4\ E_5\ E_6\ E_7\ E_8,$$

$$J = J_1\ J_2\ J_3\ J_4\ J_5\ J_6\ J_7\ J_8。$$

将 B^*, E^* 也写作类似形式。此时,假定对某一个 j, $1\leq j\leq 8$, 我们知道 E_j 和 E_j^* 的值以及 S_j 的输出异或的值 $C_j' = S_j(B_j)\ \ S_j(B_j^*)$, 则必然有 $E_j\ \ J_j \in IN_j(E_j', C_j')$, 其中 $E_j' = E_j\ \ E_j^*$。

设 E_j 和 E_j^* 是两个长度为 6 比特的串, C_j'是一个长度为 4 比特的串,定义

$$\text{test}_j(E_j, E_j^*, C_j') = \{B_j\ \ E_j \mid B_j \in IN_j(E_j', C_j')\}。$$

这里 $E_j' = E_j\ \ E_j^*$, $\text{test}_j(E_j, E_j^*, C_j')$ 也就是 E_j 和集合 $IN_j(E_j', C_j')$ 中的每一个元素取异或所得的异或值构成的集合。

综上所述,可以得出如下定理。

定理 8.1 设 E_j 和 E_j^* 是 S-盒 S_j 的两输入, S_j 的输出异或是 C_j', 记 $E_j' = E_j\ \ E_j^*$, 则密钥比特 J_j 出现在集合 $\text{test}_j(E_j, E_j^*, C_j')$ 之中,即 $J_j \in \text{test}_j(E_j, E_j^*, C_j')$。

在集合 $\text{test}_j(E_j, E_j^*, C_j')$ 中恰有 $N_j(E_j', C_j')$ 个长度为 6 比特的串, J_j 的正确值必定是这些可能值中的一个。

例 8.2 设 $E_1 = 000001$, $E_1^* = 110101$, $C_1' = 1101$。因为 $N_1(110100, 1101) = 8$, 所以在集合 $\text{test}_1(000001, 110101, 1101)$ 中恰有 8 个比特串。从表 8.4 可知,

$$IN_1(110100, 1101) = \{000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010\}$$

因此,

$$\text{test}_1(000001, 110101, 1101) = \{000111, 010001, 010111, 011101, 100011, 100101, 101001, 110011\}$$

如果有第二个这样的三重组 E_1, E_1^*, C_1', 那么我们能获得包含密钥比特 J_1 的第二个集合 test_1, 则 J_1 必定是在这两个集合的交集之中。如果有一些这样的三重组,那么就能很快地确定密钥比特 J_1。一个直接的方法是:建立一个有 64 个计数器的计数矩阵,用来记录密钥比特 J_1 的 64 种可能的取值情况。每计算一个 test_1, 如果某一 6 比特长的串在 test_1 之中,那么该 6 比特的串对应的计数器增加 1, 否则不增加。给定 t 个三重组 (E_j, E_j^*, C_j'), 希望在计数矩阵中找到一个惟一的计数器,该计数器的计数值为 t, 则这个计数器对应的 6 比特长的串即为密钥比特 J_1。

2.差分分析的应用实例

现在应用上节的观点来攻击 3-轮 DES,作为差分分析的一个应用实例。

设 $L_0\ R_0$ 和 $L_0^*\ R_0^*$ 是两对明文,对应的密文分别为 $L_3\ R_3$ 和 $L_3^*\ R_3^*$。可将 R_3 表示为

$$R_3 = L_2\ \ f(R_2, K_3) = R_1\ \ f(R_2, K_3) = L_0\ \ f(R_0, K_1)\ \ f(R_2, K_3)。$$

同样地,

$$R_3^* = L_3^* \quad f(R_0^*, K_1) \quad f(R_2^*, K_3)。$$

因此,

$$R_3' = R_3 \quad R_3^* = L_0' \quad f(R_0, K_1) \quad f(R_0^*, K_1) \quad f(R_2, K_3) \quad f(R_2^*, K_3),$$

其中 $L_0' = L_0 \quad L_0^*$。

现在,假定选择明文使得 $R_0 = R_0^*$,即 $R_0' = R_0 \quad R_0^* = 00\cdots0$(因为是选择明文攻击,所以这种假定是合理的),则

$$R_3' = L_0' \quad f(R_2, K_3) \quad f(R_2^*, K_3),$$

因为 L_0, L_0^*, R_3, R_3^* 为已知,所以 R_3' 和 L_0' 可计算出。这样,$f(R_2, K_3) \quad f(R_2^*, K_3)$ 可由下式算出:

$$f(R_2, K_3) \quad f(R_2^*, K_3) = R_3' \quad L_0'。$$

又 $f(R_2, K_3) = P(C)$, $f(R_2^*, K_3) = P(C^*)$, C, C^* 分别表示 8 个 S-盒的两个输出,所以 $P(C) \quad P(C^*) = R_3' \quad L_0'$。而 P 是固定的、公开的、线性的,故 $C \quad C^* = P^{-1}(R_3' \quad L_0')$,这正是 3-轮 DES 的 8 个 S-盒的输出异或。

另外,由于 $R_2 = L_3$ 和 $R_2^* = L_3^*$ 也是知道的(因为它们是密文的一部分),所以可使用公开知道的扩展函数 E 计算 $E = E(L_3)$和 $E^* = E(L_3^*)$。

对 3-轮 DES 的第 3 轮,已经知道 E, E^* 和 C',现在的问题是构造 test_j, $1 \leq j \leq 8$, $J_i \in \text{test}_j$。其构造如下:

输入: $L_0 \; R_0$, $L_0^* \; R_0^*$, $L_3 \; R_3$ 和 $L_3^* \; R_3^*$,其中, $R_0 = R_0^*$。

(1) 计算 $C' = P^{-1}(R_3' \quad L_0')$

(2) 计算 $E = E(L_3)$和 $E^* = E(L_3^*)$

(3) 对 $j = 1, 2, \cdots, 8$ 计算 $\text{test}_j(E_j, E_j^*, C_j')$。

通过建立 8 个具有 64 个计数器的计数矩阵,最终只能确定 K_3 中的 $6 \times 8 = 48$ 比特密钥,而其余的 $56 - 48 = 8$ 比特可通过搜索 $2^8 = 256$ 种可能的情况来确定。

下面用一个实例来图示 3-轮 DES 的攻击过程。

例 8.3　假定有下列的三对明文和密文,这里明文具有确定的异或,并且使用同一个密钥加密。为简单起见,使用十六进制表示。

先从第一对计算第 3 轮 S-盒的输入,它们分别是

$$E = E(L_3) = 000000000111111000001110100000000110100000001100$$

$$E^* = E(L_3^*) = 101111100000010101011000000010101000000010100 10$$

S-盒的输出异或是:

明文	密文
748502CD38451097 3874756438451097	03C70306D8A09F10 78560A0960E6D4CB
486911026ACDFF31 375BD31F6ACDFF31	45FA285BE5ADC730 134F7915AC253457
357418DA013FEC86 12549847013FEC86	D8A31B2F28BBC5CF 0F317AC2B23CB944

$C' = C \quad C^* = P^{-1}(R_3' \quad L_0') =$ 10010110010111010101101101100111

从第二对计算第 3 轮 S-盒的输入,它们分别是

$E =$ 10100000101111111111010000010101000000101111011 0

$E^* =$ 10001010011010100101111010111111001010001010101 0

S-盒的输出异或是

$C' =$ 10011100100111000001111101010110

从第三对计算第 3 轮 S-盒的输入,它们分别是

$E =$ 11101111000101010000011010001111011010010101111

$E^* =$ 00000101111010011010001010111111010101100000010 0

S-盒的输出异或是

$C' =$ 11010101011101011101101100101011

现在,建立 8 个具有 64 个计数器的计数矩阵。将这三对中的每一对都进行计数。

下面具体说明第一对关于 J_1 的计数矩阵的计数过程。在第 1 对中,我们有

$E_1' =$ 101111, $C_1' =$ 1001, $IN_1(101111,1001) = \{000000,000111,101000,101111\}$,

因为 $E_1 =$ 000000,所以我们有

$J_1 \in \text{test}_1(000000,101111,1001) = \{000000,000111,101000,101111\}$。

因此,我们在 J_1 的计数矩阵中的位置 0,7,40 和 47 处增加 1。

这里是将一个长度为 6 的比特串视作一个 0～63 之间的整数的二元表示,用 64 个值对应位置 0,1,2,…,63。最终的计数矩阵如下:

J_1															
1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

J_2															
0	0	0	1	0	3	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	2	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0	1	0	2	0	0	0

J_3															
0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	1
0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

J_4															
3	1	0	0	0	0	0	0	0	0	2	2	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1
1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	2	1

J_5															
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	1	0	0	0	0	2	0

J_6															
1	0	0	1	1	0	0	3	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0

J_7															
0	0	2	1	0	1	0	3	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	2	0	0	0	2	0	0	0	0	1	2	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1

J_8															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1
0	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0

在 8 个计数矩阵的每一个中,都有惟一的一个计数器具有值 3。这些计数器的位置确定 $J_1, J_2, \cdots, J_8$ 中的密钥比特。这些位置分别是 47,5,19,0,24,7,7,49。将这些整数转化为二进制数,我们可获得 $J_1, J_2, \cdots, J_8$:

$$J_1 = 101111; \quad J_2 = 000101; \quad J_3 = 010011; \quad J_4 = 000000;$$

$$J_5 = 011000; \quad J_6 = 000111; \quad J_7 = 000111; \quad J_8 = 110001。$$

现在可通过查找第三轮的密钥方案构造出密钥的 48 比特。密钥 K 具有下列形式:

0 0 0 1 1 0 1　0 1 1 0 0 0 1　0 1 ?0 1 ?0　1 ?0 0 1 0 0

0 1 0 1 0 0 1　0 0 0 0 ? ?0　1 1 1 ?1 1 ?　?1 0 0 0 1 1

这里已经略去了校验比特,"?"表示一个未知的密钥比特。完全的密钥是(用十六进制表示,并包括校验比特)1A624C89520DEC46。

8.6.2 线性密码分析

线性分析是一种已知明文攻击,最早由 Matsui 在 1993 年提出。该攻击方法利用了明文、密文和密钥的若干位之间的线性关系。在对 DES 的攻击下,这种线性关系可以通过组合各轮的线性关系而得到(假定各轮子密钥相互独立)。

此时,攻击者的目的是希望找到一个等式

$$(P \cdot \alpha) \oplus (C \cdot \beta) = (K \cdot r), \tag{8.2}$$

使其成功的概率 $p \neq \frac{1}{2}$,且 $\left| p - \frac{1}{2} \right|$ 最大。这时称(8.2)式为有效的线性表达式。此处 P 是明文,C 是密文,K 是密钥,α, β, r 是和 P, C, K 位数相同的 GF(2)上的常向量。若 α, β, r 的非零位分别为 $i_1, \cdots, i_a; j_1, j_2, \cdots, j_b; k_1, \cdots, k_c$,则,(8.2)式可变成

$$P_{[i_1, \cdots, i_a]} \oplus C_{[j_1, \cdots, j_b]} = K_{[k_1, \cdots, k_c]}。 \tag{8.3}$$

$P_{[i_1, \cdots, i_a]}$ 表示 P 的 $i_1, \cdots, i_a$ 位,$C_{[j_1, \cdots, j_b]}$ 和 $K_{[k_1, \cdots, k_c]}$ 意义相同。

若能找到(8.3)式,在已知 N 个明密文对时,线性攻击可实施如下:

(1) 对所有明文 P 和密文 C,用 T 表示(8.3)式左边为 0 的个数。

(2) 若 $T > \frac{N}{2}$,那么当 $p > \frac{1}{2}$ 时,猜测 $K_{[k_1, \cdots, k_c]} = 0$;当 $p < \frac{1}{2}$ 时,猜测 $K_{[k_1, \cdots, k_c]} = 1$;否则,当 $p > \frac{1}{2}$ 时,猜测 $K_{[k_1, \cdots, k_c]} = 1$;当 $p < \frac{1}{2}$,猜测 $K_{[k_1, \cdots, k_c]} = 0$。

当 $\left| p - \frac{1}{2} \right|$ 充分小时,成功的概率为

$$\frac{1}{2\pi} \int_{-2N\left|p-\frac{1}{2}\right|}^{\infty} e^{-x^2/2} dx。$$

显然,成功的概率依赖于 $N\left|p-\frac{1}{2}\right|$,当 N 或 $\left|p-\frac{1}{2}\right|$ 增大时,成功的概率增大。N 固定时,成功的概率依赖于 $\left|p-\frac{1}{2}\right|$。把 $\left|p-\frac{1}{2}\right|$ 达到最大时的表达式(8 .3)称为最佳逼近式,相应的概率 p 称为最佳概率。

下面以 DES 为例,说明线性分析方法。

众所周知,DES 的核心是 S-盒。对于一个给定的 S-盒, $S_i(1 \leq i \leq 8)$, $1 \leq \alpha \leq 63$ 和 $1 \leq \beta \leq 15$,定义

$$NS_i(\alpha, \beta) = \left| x \middle| 0 \leq x \leq 63, \sum_{s=0}^{5} x_{[s]} \cdot \alpha_{[s]} = \sum_{t=0}^{3} S_i(x)[t] \cdot \beta_{[t]} \right|。$$

这里 $x_{[s]}$ 表示 x 的二进制表示的第 S 个比特, $S_i(x)[t]$ 表示 $S_i(x)$ 的二进制表示的第 t 个比特, $\sum$ 表示逐比特异或和, 表示逐比特与运算, NS_i 度量了 S-盒 S_i 的非线性程度。对线性逼近式

$$\sum_{S=0}^{5} (x_{[s]} \cdot \alpha_{[s]}) = \sum_{t=0}^{3} (S_i(x)[t] \cdot \beta_{[t]}) \qquad (8.4)$$

而言, $p = \frac{NS_i(\alpha, \beta)}{64}$。当 $NS_i(\alpha, \beta) \neq 32$ 时,式(8 .4)就是一个有效的线性表达式,这时,也称 S_i 的输入和输出比特是相关的。例如, $NS_5(16, 15) = 12$,这表明 S_5 的第 4 个输入比特和所有输出比特的异或值符合的概率为 12/ 64 = 0 .19。因此,通过考虑 f 函数中的 E 扩展和 P 置换,可以推出对一个固定的密钥 K 和一个随机给定的中间输入 X,下列等式成立的概率为 0 .19:

$$X_{[15]} \bar{\imath} \quad f(X, K)[7, 18, 24, 29] = K_{[22]}。 \qquad (8.5)$$

表 8 5 描述了 S-盒 S_5 的分布表的一部分,这里垂直轴和水平轴分别表示 α 和 β,每个元素表示 $NS_5(\alpha, \beta) - 32$,通过计算所有的表,可以看出等式(8 .4)是所有 S-盒中最有效的线性逼近(也就是 $\left| NS_i(\alpha, \beta) - 32 \right|$ 是最大的),因此,等式(8 .5)是 f - 函数的最佳逼近。

表 8 5　S_5 的分布表(部分)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	4	- 2	2	- 2	2	- 4	0	4	0	2	- 2	2	- 2	0	- 4
3	0	- 2	6	- 2	- 2	4	- 4	0	0	- 2	6	- 2	- 2	4	- 4
4	2	- 2	0	0	2	- 2	0	0	2	2	4	- 4	- 2	- 2	0
5	2	2	- 4	0	10	- 6	- 4	0	2	- 10	0	4	- 2	2	4
6	- 2	- 4	- 6	- 2	- 4	2	0	0	- 2	0	- 2	- 6	- 8	2	0
7	2	0	2	- 2	8	6	0	- 4	6	0	- 6	- 2	0	- 6	- 4
8	0	2	6	0	0	- 2	- 6	- 2	2	4	- 12	2	6	- 4	4
9	- 4	6	- 2	0	- 4	- 6	- 6	6	- 2	0	- 4	2	- 6	- 8	- 4
10	4	0	0	- 2	- 6	2	2	2	2	- 2	2	4	- 4	- 4	0
11	4	4	4	6	2	- 2	- 2	- 2	- 2	- 2	2	0	- 8	- 4	0
12	2	0	- 2	0	2	4	10	- 2	4	- 2	- 8	- 2	4	- 6	- 4
13	6	0	2	0	- 2	4	- 10	- 2	0	- 2	4	- 2	8	- 6	0
14	- 2	- 2	0	- 2	4	0	2	- 2	0	4	2	- 4	6	- 2	- 4
15	- 2	- 2	8	6	4	0	2	2	4	8	- 2	8	- 6	2	0
16	2	- 2	0	0	- 2	- 6	- 8	0	- 2	- 2	- 4	0	2	10	- 20
17	2	- 2	0	4	2	- 2	- 4	4	2	2	0	- 8	- 6	2	4
18	- 2	0	- 2	2	- 4	- 2	- 8	4	6	4	6	- 2	4	- 6	0
19	- 6	0	2	- 2	4	2	0	4	- 6	4	2	- 6	4	- 2	0
20	4	- 4	0	0	0	0	0	- 4	- 4	4	4	0	4	- 4	0
21	4	0	- 4	- 4	4	- 8	- 8	0	0	- 4	4	8	4	0	4
22	0	6	6	2	- 2	4	0	4	0	6	2	2	2	0	0
23	4	- 6	- 2	6	- 2	- 4	4	4	- 4	- 6	2	- 2	2	0	4
24	6	0	2	4	- 10	- 4	2	2	0	- 2	0	2	4	- 2	- 4
25	2	4	- 6	0	- 2	4	- 2	6	8	6	4	10	0	2	- 4
26	2	2	- 8	- 2	4	0	2	- 2	0	4	2	0	- 2	- 2	0
27	2	6	- 4	- 6	0	0	2	6	8	0	- 2	4	- 6	- 2	0
28	0	- 2	2	4	0	- 6	2	- 2	6	- 4	0	2	- 2	0	0
29	4	- 2	6	- 8	0	- 2	2	10	- 2	- 8	- 8	2	2	0	4
30	- 4	- 8	0	- 2	- 2	- 2	2	- 2	2	- 2	6	4	4	4	0
31	- 4	8	- 8	2	- 6	- 6	- 2	- 2	2	- 2	- 2	- 8	0	0	- 4
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

易知，$NS_i(\alpha,\beta)$是偶数。另外，通过计算可知，如果 $\alpha=1,32$ 或 33，那么对所有的 S_i 和 β，都有 $NS_i(\alpha,\beta)=32$。

现在，将 f 函数的线性逼近扩展到整个 DES 算法上。先以 3-轮 DES 为例，将等式(8.5)应用于第一轮，可得到下列等式，该等式成立的概率为 12/64：

$$X_2[7,18,24,29]\oplus P_H[7,18,24,29]\oplus P_L[15]=K_1[22]。\qquad(8.6)$$

同样地，将等式(8.5)应用于最后一轮，即第三轮，可得下列等式，该等式成立的概率为 12/64：

$$X_2[7,18,24,29]\oplus C_H[7,18,24,29]\oplus C_L[15]=K_3[22]。\qquad(8.7)$$

将上述两式异或，可获得 3-轮 DES 的线性逼近表达式：

$$P_H[7,18,24,29]\oplus C_H[7,18,24,29]\oplus P_L[15]\oplus C_L[15]=K_1[22]\oplus K_3[22]。\qquad(8.8)$$

对随机给定的明文 P 和相应的密文 C，等式(8.8)成立的概率为 $(12/64)^2+(1-12/64)^2=0.70$。因为等式(8.5)是 f 函数的最佳线性逼近，所以等式(8.8)是 3-轮 DES 的最佳逼近表达式。现在可用前述方法求解等式(8.8)来获得 $K_1[22]\oplus K_3[22]$。

通过对一系列明密文对的分析，可获得关于 $K_1[22]$，$K_3[22]$ 的一系列方程，从而可确定出 $K_1[22]$ 和 $K_3[22]$。

对 5-轮 DES，类似于前面的讨论，首先使用 $NS_1(27,4)=22$ 获得一个 f 函数的线性逼近表达式，然后将这一表达式应用于 5-轮 DES 的第一轮和最后一轮可得如下线性等式：

$$X[27,28,30,31]\oplus f(X,K)[15]=K[42,43,45,46]。\qquad(8.9)$$

再将等式(8.5)应用于 5-轮 DES 的第二轮和第四轮，并结合等式(8.9)可得 5-轮 DES 的一个线性逼近表达式：

$$P_H[15]\oplus P_L[7,18,24,27,28,29,30,31]\oplus C_H[15]\oplus C_L[7,18,24,27,28,29,30,31]=K_1[42,43,45,46]\oplus K_2[22]\oplus K_4[22]\oplus K_5[42,43,45,46]。\qquad(8.10)$$

等式(8.10)成立的概率可计算为

$$1/2+2^3(-10/64)^2(-20/64)^2=0.519。$$

8.7　美国高级数据加密标准——AES

鉴于破解密码技术的快速演进，虽然到目前为止并无一致命的攻击法可以直接破解 DES 算法(目前的破解方法大多针对 DES 密钥长度太短来破解)，但是这些发展已直接影响了 DES 密码系统的安全性，所以美国政府在 1998 年对世界公开征求下一代的密码算法，用以取代 DES 算法。

1997 年 4 月 15 日美国国家标准技术研究所(NIST)发起征集 AES(Advanced Encryption Standards)算法的活动,并专门成立了 AES 工作组。目的是确定一个非保密的、公开披露的、全球免费使用的分组密码算法,用于保护下一世纪政府的敏感信息,也希望能够成为秘密和公开部门的数据加密标准(DES)。1997 年 9 月 12 日在联邦登记处公布了征集 AES 候选算法的通告。AES 的基本要求是比三重 DES 快而且至少和三重 DES 一样安全,分组长度为 128 比特,密钥长度为 128/ 192/ 256 比特。1998 年 8 月 20 日,NIST 召开了第一次 AES 候选会议,并公布了满足候选要求的 15 个 AES 算法。1999 年 3 月 22 日召开了第二次 AES 候选会议,公布了第一阶段的分析与测试结果。1999 年 8 月,NIST 从这 15 个算法中选出了 5 个 AES 候选算法,它们是 MARS, RC6, Rijndael, Serpent, Twofish, NIST 声称最终将从这 5 个候选算法中遴选出一个或多个算法作为 AES。2000 年 10 月 2 日,NIST 正式宣布 Rijndael 将被不加修改地作为 AES。NIST 发表了一篇长达 116 页的报告,总结了所有的论文并阐明了选择的理由。在该报告的结论部分,NIST 使用下面这段话说明选择的理由:

无论使用反馈模式还是无反馈模式,Rijndael 在广泛的计算环境中的硬件和软件实现性能都表现出始终如一的优秀。它的密钥建立时间极短,且灵敏性良好。Rijndael 极低的内存需求使它非常适合于在存储器受限的环境中使用,并且表现出了极好的性能。Rijndael 的运算易于抵抗强力和时间选择攻击。此外,无需显著地降低 Rijndael 的性能就可以提供对抗这些攻击的防护。

最后,Rijndael 的内部循环结构将会从指令级并行中获得潜在的益处。

8.7.1 AES 的评估准则

在 AES 候选算法的征集公告中,NIST 指定了用来比较候选算法的评估准则。这些准则是按照公众意见而开发的。

评估准则主要分为 3 个方面:(1)安全性;(2)代价;(3)算法和实现特性。

安全性是评估中的最重要的因素,包括下述要点:算法抗密码分析的强度,稳定的数学基础,算法输出的随机性,与其他候选算法比较的相对安全性。

代价是评估的第二个重要因素,主要包括许可要求,在各种平台上的计算效率(速度)和内存空间的需求。由于最终的 AES 算法在免费的基础上能够在世界范围内使用,因此,在选择过程中必须考虑知识产权要求和潜在的矛盾。同时也必须考虑算法在各种平台上的速度。在第一阶段评估期间,关注的焦点主要是 128 比特密钥时的速度。此外,内存空间需求和候选算法软件实现的限制也是重要的考虑因素。

算法和实现特性主要包括灵活性、硬件和软件适应性、算法的简单性等。算法的灵活性应包括下述要点:(1)处理的密钥和分组长度必须要超出最小的支持范围;(2)在许多不同类型的环境中能够安全和有效地实现;(3)可以作为序列密码、杂凑算法实现,并且可提供附加的密码服务。算法必须能够用软件和硬件两种方法实现,并且有利于有效的固件

实现。算法设计相对简单也是一个评估因素。

在介绍算法之前,还有一点要说明,Rijndael 和 AES 还是有些区别的。Rijndael 和 AES 之间的惟一差别在于各自所支持的分组长度和密钥长度的范围不同。Rijndael 是具有可变分组长度和可变密钥长度的分组密码。其分组长度和密钥长度均可独立地设定为 32 比特的任意倍数,最小值为 128 比特,最大值为 256 比特。AES 将分组长度固定为 128 比特,而且仅支持 128、192 或 256 比特的密钥长度。在 AES 的选择过程中,Rijndael 所具有的额外的分组长度和密钥长度没有评估。因此,这一点在当前的 FIPS 标准中也未被采用。

8.7.2　高级加密标准算法 AES——Rijndael

Rijndael 算法是比利时的 Joan Daemaen 和 Vincent Rijmen 提交的一个候选算法。该算法的原型是 Square 算法,它的设计策略是宽轨迹策略(wide trail strategy)。宽轨迹策略是针对差分分析和线性分析提出的,它的最大优点是可以给出算法的最佳差分特征的概率及最佳线性逼近的偏差的界,由此,可以分析算法抵抗差分密码分析及线性密码分析的能力。

Rijndael 采用的是代替/置换网络,即 SP 结构。每一轮由三层组成:线性混合层确保多轮之上的高度扩散;非线性层由非线性 S-盒构成,起到混淆的作用;密钥加层的子密钥简单的异或到中间状态上。S-盒选取的是有限域 $GF(2^8)$ 中的乘法逆运算,它的差分均匀性和线性偏差都达到了最佳。本节将首先介绍一些数学概念,并给出有关术语的解释。这些内容在阐述 Rijndael 的细节、研究其实现方法和对设计选择进行讨论时均要用到。

1.数学准备

Rijndael 中的操作,一部分是以字节(有限域 $GF(2^8)$ 的元素)定义,另一部分是以 4 字节的字定义的。这里的有限域 $GF(2^8)$ 是由不可约多项式 $m(x) = x^8 + x^4 + x^3 + x + 1$ (或记作十六进制 11B')定义的,其中的元素可以表示 GF(2)上的多项式形式,也可以表示成字节的形式,也可以表示成十六进制的形式。例如,一个由 01010111 组成的字节可表示成多项式 $x^6 + x^4 + x^2 + x + 1$,也可以是十六进制 57'。在实际的操作过程中,到底采用哪种表示方法,要看具体的情况。这里所选的 $m(x)$ 是所有次数为 8 的不可约多项式列表中的第一个。

如:'57'+'83'='D4',即是简单的二进制位异或结果,或者也可用多项式概念:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2。$$

再如:'57'*'83'='C1'。

由 $(x^6 + x^4 + x^2 + x + 1) * (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$,

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod m(x) = x^7 + x^6 + 1$。如果 $a(x) * b(x) \bmod m(x) = 1$,则称 $b(x)$ 为 $a(x)$ 的逆元。

在 Rijndae 算法中还要用到有限环 $GF(2^8)[x]/(x^4 + 1)$ 中的运算。该环中的加法定

义为简单的比特位异或,乘法运算相对复杂。假定有两个系数为 GF(2^8)上的多项式:

$$a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0 \text{ 和 } b(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0,$$

则显然 $c(x) = a(x)\,b(x)$定义如下:

$$c(x) = c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0,$$

其中,

$c_0 = a_0 * b_0$,

$c_1 = a_1 * b_0 \oplus a_0 * b_1$,

$c_2 = a_2 * b_0 \oplus a_1 * b_1 \oplus a_0 * b_2$,

$c_3 = a_3 * b_0 \oplus a_2 * b_1 \oplus a_1 * b_2 \oplus a_0 * b_3$,

$c_4 = a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3$,

$c_5 = a_3 * b_2 \oplus a_2 * b_3$,

$c_6 = a_3 * b_3$,

因为 $x^i \bmod x^4 + 1 = x^{i \bmod 4}$,所以该环中 $d(x) = a(x) \otimes b(x)$可计算为:

$d(x) = d_3 x^3 + d_2 x^2 + d_1 x + d_0$,其中

$$d_3 = a_3 * b_0 \oplus a_2 * b_1 \oplus a_1 * b_2 \oplus a_0 * b_3,$$

$$d_2 = a_2 * b_0 \oplus a_1 * b_1 \oplus a_0 * b_2 \oplus a_3 * b_3,$$

$$d_1 = a_1 * b_0 \oplus a_0 * b_1 \oplus a_3 * b_2 \oplus a_2 * b_3,$$

$$d_0 = a_0 * b_0 \oplus a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3,$$

简记为

$$\begin{matrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{matrix} = \begin{matrix} a_0 & a_3 & a_2 & a_1 & b_0 \\ a_1 & a_0 & a_3 & a_2 & b_1 \\ a_2 & a_1 & a_0 & a_3 & b_2 \\ a_3 & a_2 & a_1 & a_0 & b_3 \end{matrix} \circ$$

注:之所以选取 $x^4 + 1$,是为了算法的对称性,表述简单,且运算线性。由于 $x^4 + 1$ 不是 GF(2^8)上的不可约多项式,,因此多项式 $a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$, $a_i \in$ GF(2^8) 不一定有可逆元素,但是如果 $\gcd(a(x), x^4 + 1) = 1$,则 $a(x)$在 GF$(2^8)[x]/(x^4 + 1)$中有可逆元。

2.算法描述

Rijndael 算法是一个数据块长度和密钥长度都可变的迭代分组加密算法,数据块长和密钥长可分别为 128,192,256 位。在加密之前,对数据块做预处理。首先,把数据块写成字的形式,每个字包含 4 个字节,每个字节包含 8 比特信息。其次,把字记为列的形式。这样数据块就可以记为以下的形式:

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	…
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	…
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	…
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	…

其中，每列表示一个字 珮 = $(a_{0,j}, a_{1,j}, a_{2,j}, a_{3,j})$，每个 $a_{i,j}$ 表示一个 8 比特的字节，即 珮 $\in GF(2^8)[x]/(x^4+1)$，$a_{i,j} \in GF(2^8)$。

我们用 Nb 表示一个数据块中字的个数，那么 Nb = 4，6 或 8。

类似地，用 Nk 表示密钥中字的个数，那么 Nk = 4，6 或 8。例如，Nk = 6 的密钥可以记为以下的形式：

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

算法轮数 Nr 由 Nb 和 Nk 共同决定，具体值见表 8.6。

表 8.6　Nb 和 Nk 决定 Nr 的取值

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

在加密和解密过程中，数据都是以这种字或字节形式表示的。

Rijndael 算法的加密过程可由图 8.12 所示的流程图表示。

字节代换(ByteSub)是作用在字节上的一种非线性字节变换，这个变换(或称 S-box)是可逆的，它定义为：

其中 $a_{i,j}^{-1}$ 是 $a_{i,j}$ 在 $GF(2^8)$ 中的乘法逆。记

ByteSub(珮) = (ByteSub($a_{0,j}$), ByteSub($a_{1,j}$), ByteSub($a_{2,j}$), ByteSub($a_{3,j}$))。

这种利用有限域上的逆映射来构造 S-盒的好处是:表述简单,使人相信没有陷门,最重要的是其具有良好的抗差分和线性分析的能力。附加的仿射变换,目的是用来复杂化 S-盒的代数表达,以防止代数插值攻击。当然具体实现时,S-盒也可用查表法来实现。

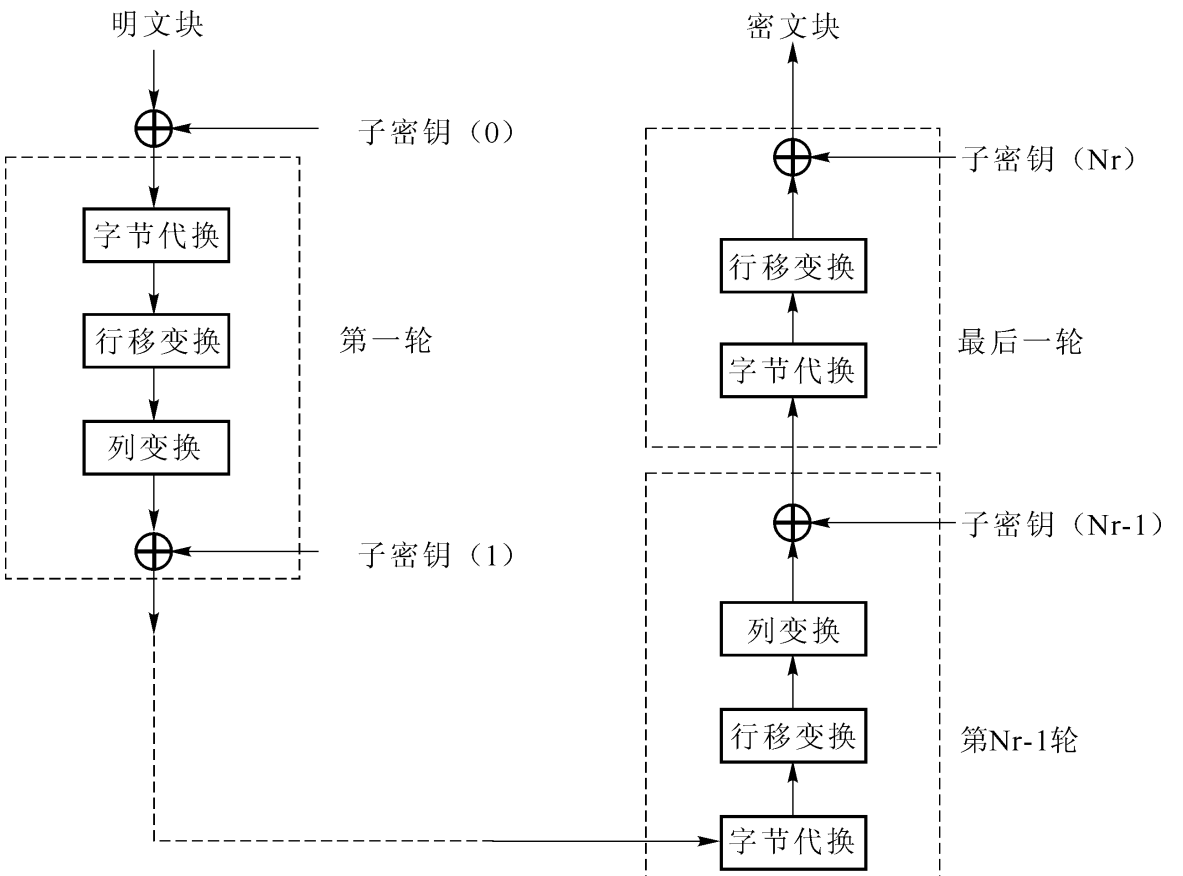


图 8.12　Rijndael 加密流程

$$\mathrm{ByteSub}(a_{i,j}) = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{matrix} a_{i,j}^{-1} + \begin{matrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{matrix},$$

行移变换(ShiftRow):在此变换的作用下,数据块的第 0 行保持不变,第 1 行循环左移 C_1 个字节,第 2 行循环左移 C_2 个字节,第 3 行循环左移 C_3 个字节,其中移位值 C_1,C_2 和 C_3 与加密块长 Nb 有关,具体列在表 8.7 中。

表 8.7　不同块长的移位值

Nb	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

列混合变换(MixColumn):MixColumn(珚)=珚　珮,这里的珚看成环 GF(2^8)[x]/(x^4+1)中的元素,珮=($'03'$,$'01'$,$'01'$,$'02'$)=$'03'x^3+'01'x^2+'01'x+'02'$,乘法是在环 GF($2^8$)[$x$]/($x^4+1$)中进行的。注意,因为珮与 x^4+1 互素,所以珮有可逆元珚=($'0B'$,$'0D'$,$'09'$,$'0E'$)=$'0B'x^3+'0D'x^2+'09'x+'0E'$。例如,如果珚=$x^3+1$,则珚　珮=$5x^3+4x^2+2x+3$。

行移变换和列混合变换相当于 SP 结构密码中的 P 层或称线性层,起着扩散作用。这里的常量之所以选珮=$'03'x^3+'01'x^2+'01'x+'02'$是为了运算简单,且最大化线性层的扩散力量。

子密钥的生成:加密和解密过程分别需要 Nr+1 个子密钥。子密钥的生成包括主密钥 珚$_0$珚$_1$…珚$_{Nk-1}$的扩展和子密钥的选取两个步骤,其中根据 Nk≤6 和 Nk>6 两种不同的情况,采取不同的主密钥扩展方式。

主密钥的扩展:

(1) 对于 Nk≤6

当 $i=0,1,\cdots,Nk-1$ 时,定义 珢$_i$=珚$_i$。当 Nk≤i≤Nb(Nr+1)-1 时,若 $i$modNk≠0,定义 珢$_i$=珢$_{i-Nk}$ī 珢$_{i-1}$;若 $i$modNk=0,令 RC[i]=x^{i-1}∈GF(2^8),$R\overline{con}[i]$=(RC[i],$'00'$,$'00'$,$'00'$)∈GF(2^8)[x]/(x^4+1)。

定义 珢$_i$=珢$_{i-Nk}$ī ByteSub(Rotate(珚$_{i-1}$))ī R$\overline{con}$[i/Nk]

其中 Rotate(a,b,c,d)是左移一个字节,即 Rotate(a,b,c,d)=(b,c,d,a)。

(2) 对于 $Nk>6$

当 $i=0,1,\cdots,Nk-1$ 时,定义 珢$_i$=珚$_i$。当 Nk≤i≤Nb($Nr+1$)-1 时,若 $i$modNk≠0 且 $i$modNk≠4,定义 珢$_i$=珢$_{i-Nk}$ī 珢$_{i-1}$;若 $i$modNk=0,令 RC[i]=x^{i-1}∈GF(2^8),R$\overline{con}$[i]=(RC[i],$'00'$,$'00'$,$'00'$)∈GF(2^8)[x]/(x^4+1),定义 珢$_i$=珢$_{i-Nk}$ī ByteSub(Rotate(珚$_{i-1}$))ī R$\overline{con}$[i/Nk];若 $i$modNk=4,定义 珢$_i$=珢$_{i-Nk}$ī ByteSub(珚$_{i-1}$)。这样,我们就得到了 Nb(Nr+1)个字 珢$_j$。第 i 个子密钥就是 珢$_{Nb\times i}$珢$_{Nb\times i+1}$…珢$_{Nb(i+1)-1}$。

Rijndael 解密算法的结构与 Rijndael 加密算法的结构相同,其中的变换为加密算法变换的逆变换,且使用了一个稍有改变的密钥编制。行移变换的逆是状态的后三行分别移动 Nb-C_1,Nb-C_2,Nb-C_3 个字节,这样,在 i 行 j 处的字节移到(j+Nb-C_i)modNb

处。字节代换的逆是 Rijndael 的 *S*-盒的逆作用到状态的每个字节,这可由如下得到:先进行仿射的逆变换,然后把字节的值用它的乘法逆代替。列混合变换的逆类似于列混合变换,状态的每一列都乘以一个固定的多项式 $d(x)$, $d(x) = '0B'x^3 + '0D'x^2 + '09'x + '0E'$。

我们可能已经注意到,在 Rijndael 算法中,仅限于使用诸如异或和 GF(2^8)中的常量乘法等简单运算。*S*-盒利用了 GF(2^8)中的乘法逆和仿射变换。Rijndael 排除了大量的简单有效的运算,如模 2^n 下的加、减、乘法以及数据相依移位等,这些运算往往是许多分组密码所广泛使用的基本模块,并且它们似乎也具有了良好的非线性和/或扩散性质。如果想知道 Rijndael 密码不使用这些运算的理由,可以参考本书参考文献[69],[70]。

有关算法的更详细描述请参看本书参考文献[70]或 AES 主页 www.nist.gov/aes/。

3. Rijndael 算法的安全性

Rijndael 加解密算法中,每轮常数的不同消除了密钥的对称性,密钥扩展的非线性消除了相同密钥的可能性;加解密使用不同的变换,消除了在 DES 里出现的弱密钥和半弱密钥存在的可能性;总之,在 Rijndael 的加解密算法中,对密钥的选择没有任何限制。

经过验证,Rijndael 加解密算法能有效地抵抗目前已知的攻击方法的攻击。如部分差分攻击、相关密钥攻击、插值攻击(interpolation attack)等。对于 Rijndael,最有效的攻击还是穷尽密钥搜索攻击。

依靠有限域/有限环的有关性质给加密解密提供了良好的理论基础,使算法设计者可以既高强度地隐藏信息,又同时保证了算法可逆,又因为 Rijndael 算法在一些关键常数(例如,在 $m(x)$)的选择上非常巧妙,使得该算法可以在整数指令和逻辑指令的支持下高速完成加解密,在专用的硬件上,速率可高于 1 Gbit/s,从而得到了良好的效率。除了加解密功能外,Rijndael 算法还可以实现如 MAC、Hash、同步流密码、生成随机数、自身同步流密码等功能;Rijndael 算法可有效地应用于奔腾机、智能卡、ATM、HDTV、B-ISDN、声音、卫星通信等各个方面。

8.8 欧洲 21 世纪数据加密标准

NESSIE(New European Schemes for Signature, Integrity, and Encryption)是欧洲的信息社会技术(IST)委员会计划出资 33 亿欧元支持的一项工程——数字签名、完整性和加密新欧洲方案。

项目的伙伴包括:

鲁汶天主教大学(比利时);

巴黎高等师范学校(法国);

皇家霍洛威学院,伦敦大学(英国);

西门子股份有限公司(德国);
以色列理工学院(以色列);
卑尔根大学(挪威)。

NESSIE 是一个三年的项目,起始于 2000 年 1 月 1 日。该项目的主要目标就是通过公开征集和进行公开的、透明的测试评价提出一套强的密码标准。这些标准包括分组密码、流密码、杂凑函数、消息认证码(MAC)、数字签名和公钥加密。此外,该项目将发展一种密码体制评估方法(包括安全性和性能评价) 和支持评估的一个软件工具箱。该项目的另一个目标就是广泛传播这些工程成果。当然不言而喻,还有一个目标就是保持欧洲工业界在密码学研究领域的领先地位。

为了便于公开的评价处理,设置了 4 个 NESSIE 工作间。第一工作间(2000 年 11 月 13～14 日)致力于提交组件的介绍。第二工作间(2001 年 9 月 12～13 日)致力于有关组件的早期结果。第三工作间(2002 年 11 月 6～7 日)致力于新结果及安全汇报和执行汇报 1 .0 版本的讨论。第四工作间(2003 年 2 月 26 日)总结了进入第二轮候选算法的安全汇报和执行汇报并发布最终结果。

NESSIE 项目分为两个阶段。在安全评估的第一阶段, NESSIE 合作者对所有提交作初步分析,NESSIE 也接受来自外部的评论。第一阶段的任务是对所有的提交作初步评估,包括安全评估和性能评估,据此决定哪些候选将在第二阶段被考虑。在第二阶段,余下的提交受到更详细的审查,公告 D21 是第二阶段执行评价的最终文件,它提供一个对所有提交执行效率的详细回顾,并提供对候选提交改进了的性能估计。

执行评价主要考虑密码组件在各种平台上(包括从低端智能卡到高端处理器)的运行效率,所采用的方法是在共同的测试环境下测试相关算法的运行快慢,并与一些现存的标准作比较(如 3DES,AES)。公告 D20 总结了第二阶段安全评估的结论,是一个最终的安全汇报,汇总了所有提交的安全评估。密码学的实践告诉我们:要证明一个密码算法是安全的一般是不可能。人们只能尽量保证一个密码算法在目前的已有的分析水平上是充分安全的,所以算法一般都规定安全期限。在评估算法的安全性方面,除了 AES 外,几乎没有别的经验可利用,因而较易受到评估者主观因素的限制。为避免主观因素,NESSIE 作了许多较有开创性的工作,NESSIE 制定了一系列的安全评估的方法和准则。

安全评估与性能汇报一起决定哪些候选将有资格成为 NESSIE 标准。

与美国相比,尽管欧洲在密码算法标准化方面起步较晚,但是却发挥了其后发优势,NESSIE 项目是一个庞大的工程,覆盖面相当广,加密、签名、认证等方面都有所考虑。

NESSIE 评选是一个公开的、透明的处理过程。欧洲在密码研制、使用方面的管制比较宽松,但并没有因此而影响其信息安全的保障。

NESSIE 的成就不仅在于推出自己的一套强的算法标准,更重要的是发展了一套密码体制的评估方法,大大地推动了密码学的实际应用。NESSIE 项目的经验和教训可为我国民用密码的研制和管理提供有益的借鉴。

8.8.1 NESSIE 建议

1.64 比特分组密码

中标的算法是 MISTY1。

NESSIE 项目没有发现对 MISTY1 的攻击。MISTY1 类似于分组密码 KASUMI,众多对 KASUMI 的分析适用于 MISTY1。早在 KASUMI 成为一个 3GPP 标准之前,KASUMI 就被详细审查过。

第二阶段对其他 64 比特分组密码的研究评价:

(1) 没有发现对 Khazad 的攻击,在 NESSIE 项目看来,它是一个值得进一步研究的有趣算法。然而,Khazad 结构的对称性令人担心。

(2) NESSIE 项目没有发现对 IDEA 的攻击,然而,不选择它是因为知识产权问题和担心其密钥方案。

(3) 没有发现对 SAFER + + 64 的攻击。然而,在 NESSIE 安全评估汇报中谈到,它的某种结构问题令人担心,同时在智能卡中比其他密码更慢。

(4) 3DES 是安全的但却是一个慢的分组密码。

2.128 比特分组密码

NESSIE 推荐 128 比特的分组密码是 AES 和 Camellia。

AES 已被美国国家标准和技术研究所‘ National Institute of standards and Technology’评估为一个安全的分组密码并采纳为美国联邦信息处理标准。Camellia 有许多类似于 AES 的特性,许多 AES 的分析可应用于 Camellia。NESSIE 项目没有发现对 AES 和 Camellia 的攻击。不过,NESSIE 的合作伙伴及密码界,通过对 AES 和 Camellia 广泛的观察,担心 AES 简单的代数结构可能导致未来新的攻击,由于设计 Camellia 时意识到这个问题,在某种程度上复杂化了算法的代数表述。

对第二阶段其他 128 比特分组密码评价: 由于最近严重的知识产权问题,NESSIE 未考虑 RC6;没有发现对 SAFER + + 128 的攻击,然而问题是其某种结构的特性和较低的安全冗余。

3.256 比特分组密码

NESSIE 推荐的是 SHACAL-2。

对其他分组长大于 128 比特密码评价:由于知识产权问题,不选择 RC6; 不选择 SHACAL-1,因为它的弱密钥方案。

8.8.2 Camellia 算法简介

Camellia 密码是由日本电报电话公司和日本三菱电子公司联合设计的,支持 128 比特的分组大小,128,192 和 256 比特密钥长度,和高级加密标准(AES)有同样的安全限定。

1 . Camellia 构造

(1) 加密过程

图 8 .13 显示了 128 比特的密钥加密过程,数据随机化部分是一个 18 轮的 Feistel 结构,另外在第 6 轮和第 12 轮之后插入两个 $FL/\ FL^{-1}$ 层,首轮之前和末轮之后增加一个白化层。期间所使用的子密钥 $kw_{t(64)}(t=1,2,3,4)$, $k_{u(64)}(u=1,2,\cdots,18)$, $kl_{v(64)}(v=1,2,3,4)$由种子密钥 K 生成(见密钥方案)。

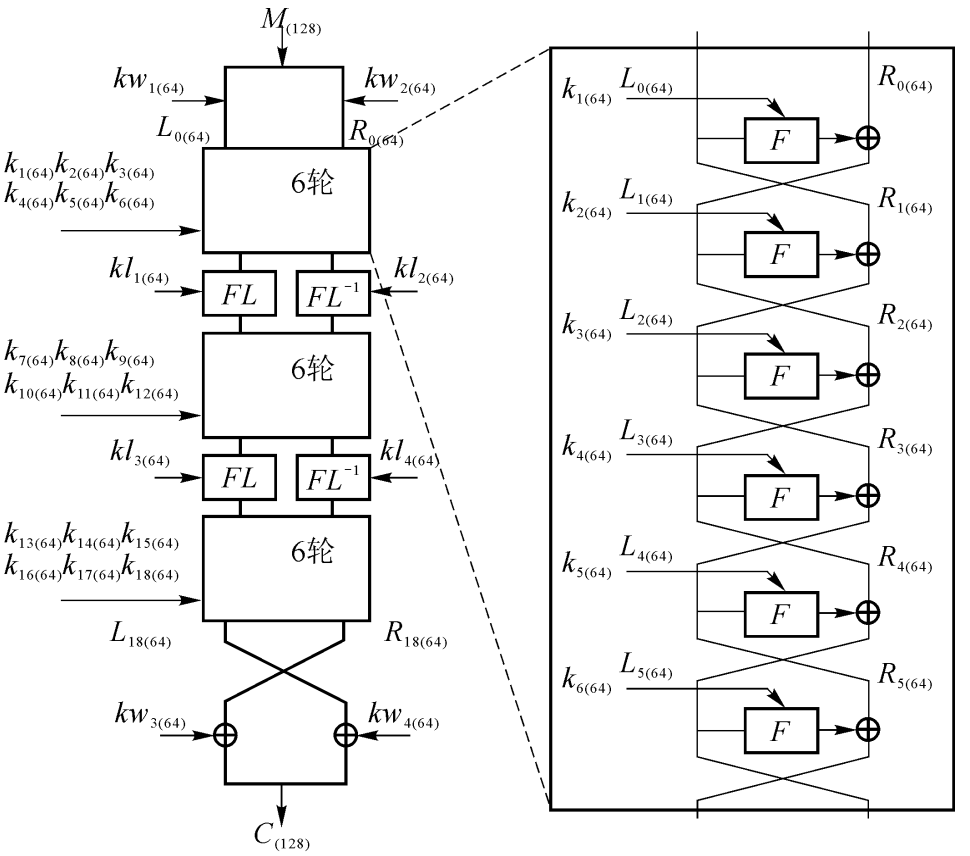


图 8 .13　128 比特密钥加密过程

192 和 256 比特密钥的加密规定为 24 轮,所以只需在 128 比特密钥加密流图的第 18 轮之后再增加一个 $FL/\ FL^{-1}$ 函数层和一个 6 轮的迭代模块。

(2) 解密过程

除了颠倒子密钥的次序,解密过程与加密过程是类似的,这是由 Camellia 密码的 Feistel 结构所决定的。

(3) 密钥方案

设 Camellia 的 128 比特、192 比特、256 比特的密钥为 $K_{(128)}$, $K_{(192)}$, $K_{(256)}$。假设对于 128 比特的密钥 $K_{(128)}=K_{L(128)}$, $K_{R(128)}=0$;对于 192 比特的密钥 $K_{(192)}=K_{L(128)}\parallel K_{RL(64)}$, $K_{RR(64)}=\overline{K_{RL(64)}}$;对于 256 比特的密钥,定义 $K_{(256)}=K_{L(128)}\parallel K_{R(128)}$。对于任意长度的密钥,定义 $K_{L(128)}=K_{LL(64)}\parallel K_{LR(64)}$, $K_{R(128)}=K_{RL(64)}\parallel K_{RR(64)}$。

用上述变量,可以产生两个 128 比特的中间变量 $K_{A(128)}$, $K_{B(128)}$ (见图 8 .14)。其中所用的常量 $\sum_i$ 定义为第 i 个素数的平方根十六进制表达从第二到第十七个位连续值

(见表 8 .8)。

表 8 .8　密钥常量表

$\sum_{1(64)}$ = A09E667F3BCC908B	$\sum_{4(64)}$ = 54FF53A5F1D36F1C
$\sum_{2(64)}$ = B67AE8584CAA73B2	$\sum_{5(64)}$ = 10E527FADE682D1D
$\sum_{3(64)}$ = C6EF372FE94F82BE	$\sum_{6(64)}$ = B05688C2B3E6C1FD

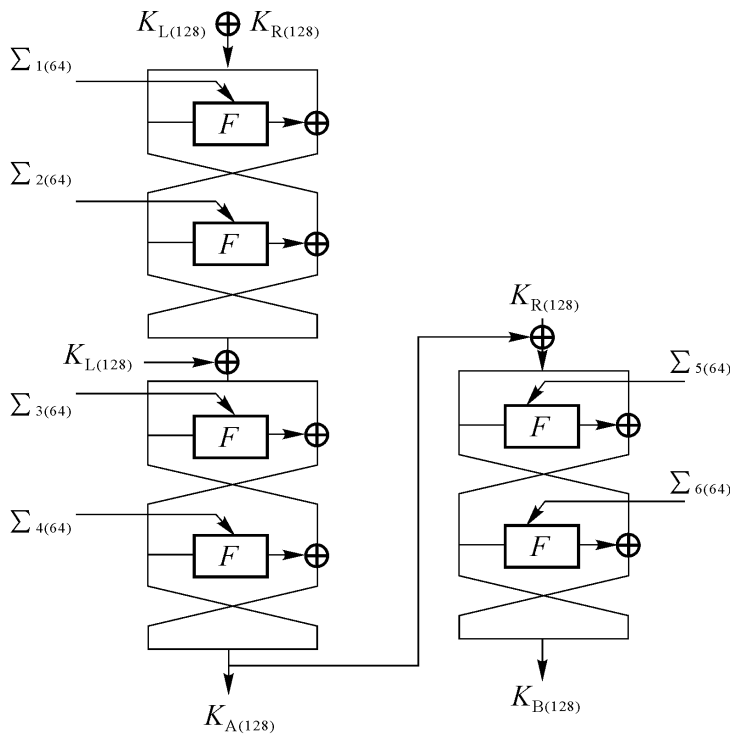


图 8 .14　密钥方案

子密钥, $kw_{t(64)}$, $k_{u(64)}$, $kl_{v(64)}$ 由 $K_{L(128)}$, $K_{R(128)}$, $K_{A(128)}$, $K_{B(128)}$ 循环移位而成(见表 8 .9 或 8 .10)。

密钥方案非常简单且共享了加密程序的一部分,支持子密钥的在线产生。另外,子密钥的存储需求很小。

表 8.9　128 比特密钥生成的子密钥

$kw_{1(64)}$	$(K_L \lll 0)\ L_{(64)}$
$kw_{2(64)}$	$(K_L \lll 0)\ R_{(64)}$
$k_{1(64)}$	$(K_A \lll 0)\ L_{(64)}$
$k_{2(64)}$	$(K_A \lll 0)\ R_{(64)}$
$k_{3(64)}$	$(K_L \lll 15)\ L_{(64)}$
$k_{4(64)}$	$(K_L \lll 15)\ R_{(64)}$
$k_{5(64)}$	$(K_A \lll 15)\ L_{(64)}$
$k_{6(64)}$	$(K_A \lll 15)\ R_{(64)}$
$kl_{1(64)}$	$(K_A \lll 30)\ L_{(64)}$
$kl_{2(64)}$	$(K_A \lll 30)\ R_{(64)}$
$k_{7(64)}$	$(K_L \lll 45)\ L_{(64)}$
$k_{8(64)}$	$(K_L \lll 45)\ R_{(64)}$
$k_{9(64)}$	$(K_A \lll 45)\ L_{(64)}$
$k_{10(64)}$	$(K_L \lll 60)\ R_{(64)}$
$k_{11(64)}$	$(K_A \lll 60)\ L_{(64)}$
$k_{12(64)}$	$(K_A \lll 60)\ R_{(64)}$
$kl_{3(64)}$	$(K_L \lll 77)\ L_{(64)}$
$kl_{4(64)}$	$(K_L \lll 77)\ R_{(64)}$
$k_{13(64)}$	$(K_L \lll 94)\ L_{(64)}$
$k_{14(64)}$	$(K_L \lll 94)\ R_{(64)}$
$k_{15(64)}$	$(K_A \lll 94)\ L_{(64)}$
$k_{16(64)}$	$(K_A \lll 94)\ R_{(64)}$
$k_{17(64)}$	$(K_L \lll 111)\ L_{(64)}$
$k_{18(64)}$	$(K_L \lll 111)\ R_{(64)}$
$kw_{3(64)}$	$(K_A \lll 111)\ L_{(64)}$
$kw_{4(64)}$	$(K_A \lll 111)\ R_{(64)}$

表 8.10　192 和 256 比特密钥生成的子密钥

$kw_{1(64)}$	$(K_L \lll 0)\ L_{(64)}$
$kw_{2(64)}$	$(K_L \lll 0)\ R_{(64)}$
$k_{1(64)}$	$(K_B \lll 0)\ L_{(64)}$
$k_{2(64)}$	$(K_B \lll 0)\ R_{(64)}$
$k_{3(64)}$	$(K_R \lll 15)\ L_{(64)}$
$k_{4(64)}$	$(K_R \lll 15)\ R_{(64)}$
$k_{5(64)}$	$(K_A \lll 15)\ L_{(64)}$
$k_{6(64)}$	$(K_A \lll 15)\ R_{(64)}$
$kl_{1(64)}$	$(K_R \lll 30)\ L_{(64)}$
$kl_{2(64)}$	$(K_R \lll 30)\ R_{(64)}$
$k_{7(64)}$	$(K_B \lll 30)\ L_{(64)}$
$k_{8(64)}$	$(K_B \lll 30)\ R_{(64)}$
$k_{9(64)}$	$(K_L \lll 45)\ L_{(64)}$
$k_{10(64)}$	$(K_L \lll 45)\ R_{(64)}$
$k_{11(64)}$	$(K_A \lll 45)\ L_{(64)}$
$k_{12(64)}$	$(K_A \lll 45)\ R_{(64)}$
$kl_{3(64)}$	$(K_L \lll 60)\ L_{(64)}$
$kl_{4(64)}$	$(K_L \lll 60)\ R_{(64)}$
$k_{13(64)}$	$(K_R \lll 60)\ L_{(64)}$
$k_{14(64)}$	$(K_R \lll 60)\ R_{(64)}$
$k_{15(64)}$	$(K_B \lll 60)\ L_{(64)}$
$k_{16(64)}$	$(K_B \lll 60)\ R_{(64)}$
$k_{17(64)}$	$(K_L \lll 77)\ L_{(64)}$
$k_{18(64)}$	$(K_L \lll 77)\ R_{(64)}$
$k_{19(64)}$	$(K_R \lll 94)\ L_{(64)}$
$k_{20(64)}$	$(K_R \lll 94)\ L_{(64)}$
$k_{21(64)}$	$(K_A \lll 94)\ R_{(64)}$
$k_{22(64)}$	$(K_A \lll 94)\ R_{(64)}$
$k_{23(64)}$	$(K_L \lll 111)\ L_{(64)}$
$k_{24(64)}$	$(K_L \lll 111)\ R_{(64)}$
$kw_{3(64)}$	$(K_B \lll 111)\ L_{(64)}$
$kw_{4(64)}$	$(K_B \lll 111)\ R_{(64)}$

2 . Camellia 组件

*(*1*)* F 函数

Camellia 的轮函数 *F* 设计沿用了 *E*2*(*AES 提交*)*的 *F* 函数, 主要区别在于, Camellia 采用1 -轮SPN 结构*(*见图 8 .15*)*。 *F* 函数用一个 64 比特的子密钥变换 64 比特的明文为 64 比特的密文, 可以简单表示为

$$Y_{(64)} = F(X_{(64)}, k_{i(64)}) = P(S(X_{(64)} \oplus k_{i(64)}),$$

P 是一个基于字节的线性变换, *S* 是 8 个并行的 8×8 *S*-盒代换运算。

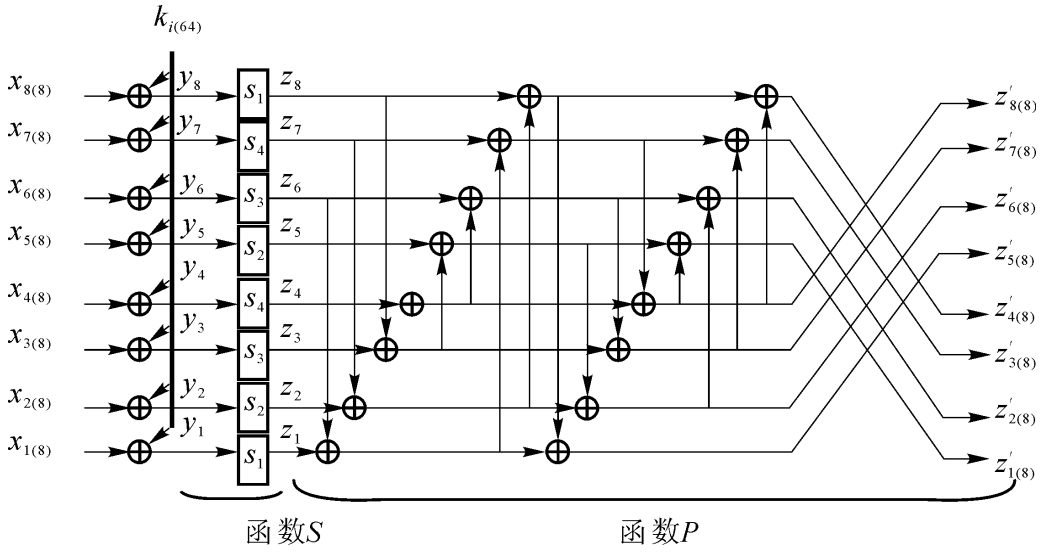


图 8 .15　函数 *F*

*(*2*)* *FL* 和 FL^{-1}函数

每 6 轮插入 *FL*/ FL^{-1}函数层, 提供轮间的不规则性, 作用是击退未知的攻击。同时 *FL*/ FL^{-1}函数层的插入并未改变 Feistel 密码的加解密相似的特点。

FL/ FL^{-1}函数仅由逻辑运算, 如与、或、异或和轮转组成*(*见图 8 .16*)*, 所以软、硬件实现很快。

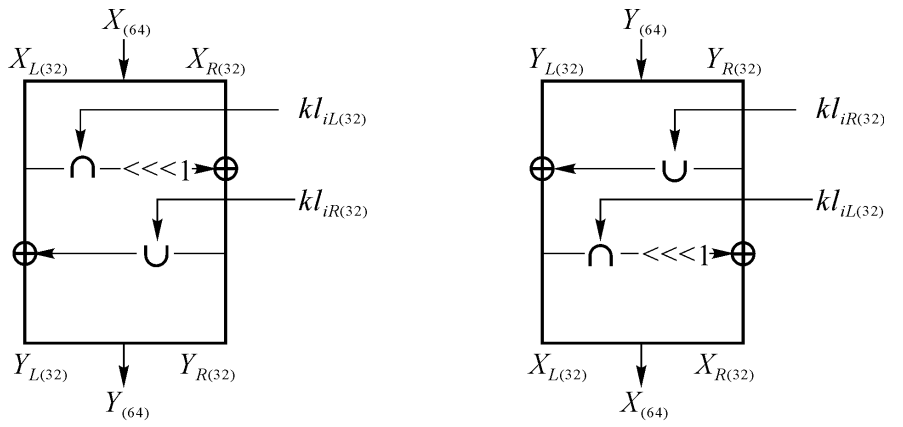


图 8 .16　函数 *FL* 和 FL^{-1}

*(*3*)* *P* 函数

P 函数是 *F* 函数的一部分,是一个基于字节的线性变换。为计算的有效性,仅用字节异或;为抗击差分和线性分析,它的分支数必须是最优的,在所有满足条件的线性变换中,设计者选择了一个在 32 位处理器和低端智能卡上能高效实现的线性变换,可表示为

$$\begin{matrix} z_8 \\ z_7 \\ \cdots \\ z_1 \end{matrix} \rightarrow \begin{matrix} z_8' \\ z_7' \\ \cdots \\ z_1' \end{matrix} = P \begin{matrix} z_8 \\ z_7 \\ \cdots \\ z_1 \end{matrix}, P = \begin{matrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$

(4) S-盒

Camellia 总共用了 4 个不同的 8×8 双射 S-盒。类似于 Rijndael 算法(AES 标准),它们都是 GF (2^8)上 x^{-1}的仿射变换。4 个 S-盒稍微不同,目的是改进密码的抗截断差分分析的能力。4 个 S-盒可这样表示:

s_1 : $x_{(8)} \rightarrow h(g(f(C5 \bar{1} x_{(8)}))) \bar{1} 6E$,　s_2 : $x_{(8)} \rightarrow s_1(x_{(8)}) \lll_1$,

s_3 : $x_{(8)} \rightarrow s_1(x_{(8)}) \ggg_1$,　s_4 : $x_{(8)} \rightarrow s_1(x_{(8)} \lll_1)$。

f 函数是一个字节变换,若记 $f: a_1\ a_2 \cdots a_8 \rightarrow b_1\ b_2 \cdots b_8$,则 $b_1 = a_6 \bar{1} a_2$, $b_2 = a_7 \bar{1} a_1$, $b_3 = a_8 \bar{1} a_5 \bar{1} a_3$, $b_4 = a_8 \bar{1} a_3$, $b_5 = a_7 \bar{1} a_4$, $b_6 = a_5 \bar{1} a_2$, $b_7 = a_8 \bar{1} a_1$, $b_8 = a_6 \bar{1} a_4$。

g 函数也是字节变换,若记 $g: a_1\ a_2 \cdots a_8 \rightarrow b_1\ b_2 \cdots b_8$,则有

$$(b_8 + b_7\alpha + b_6\alpha^2 + b_5\alpha^3) + (b_4 + b_3\alpha + b_2\alpha^2 + b_1\alpha^3)\beta$$

$$= 1/((a_8 + a_7\alpha + a_6\alpha^2 + a_5\alpha^3) + (a_4 + a_3\alpha + a_2\alpha^2 + a_1\alpha^3)\beta),$$

其中,β是 GF(2^8)上满足 $\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0$ 的元素。$\alpha = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$ 是 GF(2^8)中满足 $\alpha^4 + \alpha + 1 = 0$ 的元素。

h 函数也是一个字节变换,若记 $h: a_1\ a_2 \cdots a_8 \rightarrow b_1\ b_2 \cdots b_8$,则 $b_1 = a_5 \bar{1} a_6 \bar{1} a_2$, $b_2 = a_6 \bar{1} a_2$, $b_3 = a_7 \bar{1} a_4$, $b_4 = a_8 \bar{1} a_2$, $b_5 = a_7 \bar{1} a_3$, $b_6 = a_8 \bar{1} a_1$, $b_7 = a_5 \bar{1} a_1$, $b_8 = a_6 \bar{1} a_3$。

为最小化硬件设计,GF(2^8)中的元素可表示为子域 GF(2^4)上的多项式。换句话说,能用子域 GF(2^4)上的几个运算实现 S-盒,使得硬件实现 S-盒只需几个逻辑门。

附加的两个仿射变换 f,h 的作用是复杂化 S-盒在 GF(2^4)中的表达。

当然,S-盒也可转换为查表运算,限于篇幅,未在此列出。

3 .安全性

Camellia 的设计目标有两个:高水平的安全性和多平台的有效性。

Camellia 没有发现安全缺陷。设计者声称,12 轮以上不存在线性偏差和差分特征概率大于 2^{-128}的差分/ 线性特征。15 轮以上不存在概率大于 2^{-135}的差分和线性特征。这

是因为 4 个基于域 GF(2^8)上 x^{-1} 仿射变换的 *S*-盒应用,这个特殊的数学函数确保 *S*-盒具有最优的差分和线性特征 2^{-6}。使得密码具有很强的抗差分和线性攻击能力的另外一个原因是扩散层线性变换的使用,该线性变换具有最优的分支数 5。仿射变换作用在于使得 *S*-盒的代数表达复杂化,进而避免了插值攻击。鉴于密码由同样轮迭代而成,设计者在每 6 轮插入 *FL* 和 FL^{-1} 函数,引进某种不规则性,以便抗击 Slide 攻击,同时也使得差分攻击更为困难,因为这两个依赖于密钥函数的应用改变了差分路径。对 6 轮版本,Square 攻击需要 2^{112} 加密和 13×2^8 个明文,设计者声称 10 轮版本就足够安全。

Camellia 被仔细地设计,以便能击退所有已知的密码分析,设计者设想 10 到 20 年安全期限。该密码非常适合软件、硬件实现,应用范围包括从低耗智能卡到高速网络系统。用汇编语言的优化实现在 Pentium Ⅲ 800 MHz 上,加密速度超过 276 Mbit/ s,比优化的 DES 实现快多了。另外,一个显著的特征是它的小的硬件设计,硬件设计包括密钥方案,加密和解密,只占用大约 1 .1 万个门,这在现存的 128 比特分组密码中是最小的,极好地迎合了当前无线卡市场需求。

8 .9 其他分组密码算法综述

除了前面介绍的分组密码外,还有其他一些分组密码,比如 DES 变形(包括 NewDES、多重 DES、白化了的 DES、S-盒可变的 DES、使用独立子密钥的 DES、广义 DES 等),RC 系列分组密码(包括 RC2、RC5、RC6 等), Lucifer, Madryga, Feal-N, LOKI 系列分组密码(包括 LOKI89、LOKI91、LOKI97 等), Khufa, Safer 系列, CAST, 3-WAY, TEA, SHARK, BEAR, LION, Blowfish, GOST, AQUARE, MISTY, 15 个 AES 候选算法等。国际上目前公开的分组密码不下 100 种,这么多的算法不可能逐一介绍,本节仅介绍 IDEA 算法和 RC6 算法。对其他算法感兴趣的读者可参阅相关文献。

8 .9 .1 IDEA 算法

X J .Lai 和 J .L .Massey 提出的第一版 IDEA(国际数据加密算法)于 1990 年公布,当时称之为 PES(建议加密标准)。1991 年,在 Biham Shamir 对其采用了差分分析之后,设计者为抗此种攻击,增加了他们的密码算法的强度。它们把新算法称为 IPES,即改进型建议加密标准。1992 年进行了改进,强化了抗差分分析的能力。这是近年来提出的各种分组密码中一个很成功的方案,已在 PGP 中采用。IDEA 的明文和密文分组都是 64 比特,密钥的长度是 128 比特,同一算法既可用于加密又可用于解密,该算法所依据的设计思想是“混合来自不同代数群中的运算”。该算法需要的“混乱”可通过连续使用三个“不相容”的群运算于两个 16 比特子块来获得,并且该算法所选择使用的密码结构可提供必要的“扩散”。该算法的密码结构的选择也考虑了该密码算法硬件和软件实现功能。

IDEA 的描述:IDEA 是由 8 轮和随后的一个输出变换组成,图 8 .18 的计算框图刻画了该密码算法的整个第一轮和输出变换。

在 3 个不同的群运算中,要特别注意模 $2^{16}+1$ 整数乘法运算⊙,这里除了将 16 比特的全零子块处理为 2^{16} 外,其余 16 比特的子块均按通常处理成一个整数的二进制表示对待,例如,$(0,0,\cdots,0)\odot(1,0,\cdots,0)=(1,0,\cdots,0,1)$,这是因为 $2^{16}2^{15}\bmod(2^{16}+1)=2^{15}+1$。

对任何加密算法的设计,混淆(Confusion)及扩散(Diffusion)是两个最重要的安全特性。混淆是说加密算法的密文与明文及密钥关系十分复杂,无法从数学上描述,或从统计上去分析。扩散是说明文中的任一位以及密钥中的任一位,对全体密文位有影响,经由此种扩散作用,可以隐藏许多明文在统计上的特性,增加密码的安全。

IDEA 的混淆特性是经由混合下述三种函数而达成:

(1) 比特为单位的异或运算,用 ⊕ 表示。

(2) 定义在模 2^{16} (mod 65536)的模加法运算,其操作数都是 16 位二进数表示的整数,用⊞表示这个运算。

(3) 定义在模 $2^{16}+1$(mod 65537)的模乘法运算。因为 65537 是一素数,所以对任何数(除 0 以外)的乘法逆元是存在的。值得一提的是,为了保证即使当“0”出现在 16 位的操作数时也有乘法逆元存在,“0”被定义为 2^{16},用⊙表示这个运算。

以上 3 个函数,由于基于以下的“非兼容”(Incompatible)性,当应用在 IDEA 时,可以充分发挥出混淆的特性。

(1) 3 个函数中的任意两个函数,都无法满足“分配律”,例如对运算⊙及⊞,存在 $a,b,c\in GF(2)^{16}$,而且 $a\boxplus(b\odot c)\neq(a\boxplus b)\odot(a\boxplus c)$。

(2) 3 个函数中的任意 2 个函数,都无法满足“结合律”。例如对运算⊞及⊕,存在 $a,b,c\in GF(2)^{16}$,而且 $a\boxplus(b\oplus c)\neq(a\boxplus b)\oplus c$。

因此在 IDEA 的设计中,使用了这三种函数的混合组合来打乱数据。攻击者无法用化简的方式来分析密文与明文及密钥之间的关系。

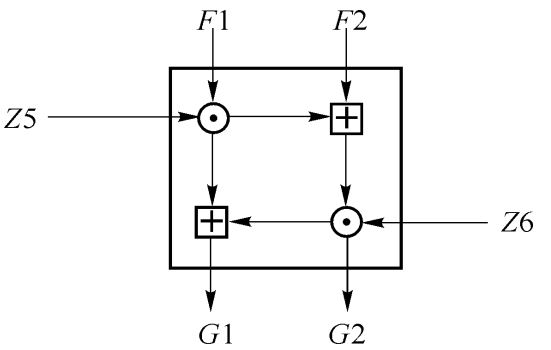


图 8 .17　MA 运算器结构

IDEA 的扩散特性是建立在乘法/ 加法(MA)的基本结构上。图 8 .17 表示 MA 的基本结构。该结构一共有 4 个 16 位的输入,两个 16 位的输出。其中的两个输入来源于明文,另两个输入是子密钥,源于 128 位加密密钥。Lai 经过分析验证,数据经过 8 轮的 MA 处理,可以得到完整的扩散特性。

IDEA 的加密流程如图 8 .18 所示。

图 8 .18 显示了 IDEA 加密方法的流程。IDEA 方法包括 8 轮的重复运算,加上最后的输出变换。64 位的明文分组在每一轮中都被分成 4 份,每份 16 位为一单元来处理。

每一轮中有 6 个不同的子密钥来参与作用。最后的输出变换运算用到了另外的 4 个子密钥。所以在 IDEA 加密过程中共享到了 52 个子密钥。这些子密钥都是由一个 128 位的加密密钥产生的。

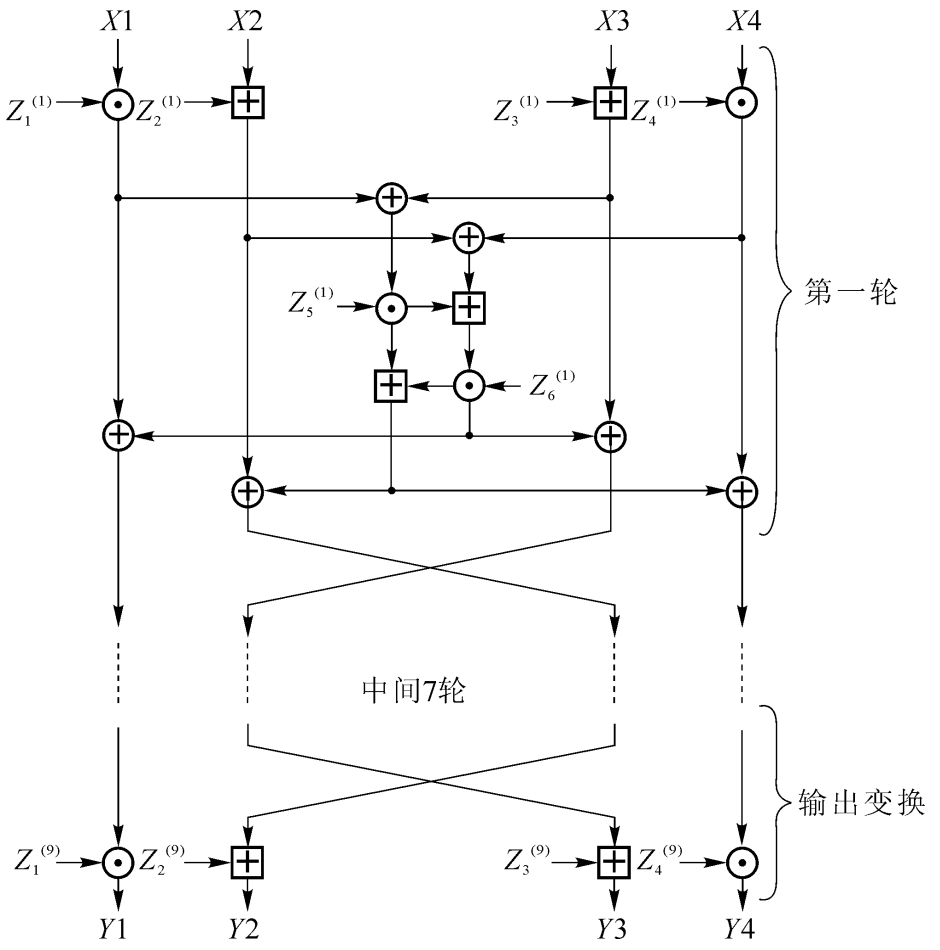


图 8 .18　IDEA 的加密流程

每一轮的运算又分成两部分。第一部分即是前面所说的变换运算,利用加法及乘法运算将 4 份 16 位的子明文分组与 4 个子密钥混合,产生 4 份 16 位的输出,这 4 份输出又两两配对,以逻辑异或将数据混合,产生两份 16 位的输出。这两份输出,连同另外的两个子密钥成为第二部分的输入。第二部分即是前面提到用以产生扩散特性的 MA 运算,MA 运算生成两份 16 位输出,MA 的输出再与变换运算的输出以异或作用生成 4 份 16 位的最后结果,这 4 份结果即成为下一轮运算的原始输入。值得一提的是,这 4 份最后结果中的第二、三份输出,是经过位置互换而得到,此举的目的在于对抗差分分析法。

明文分组在经过 8 轮加密后,仍需经过最后的输出变换运算才能形成正式的密文。最后的输出变换运算与每一轮的变换运算大致相同。惟一不同之处在于,第二、三份输出不需经过位置互换。这个特殊安排的目的在于使我们可以使用与加密算法相同结构的解密算法解密,简化了设计及使用上的复杂性。

IDEA 一次完整的加密运算需要 52 个子密钥。这 52 个 16 位的子密钥都是由用户选择的 128 位加密密钥产生的。生成过程如下:

将 128 位分成 8 份,直接用来作为最前面要用的 8 个密钥子块,这 52 个密钥子块的

顺序为: $Z_1^{(1)}, Z_2^{(1)}, \cdots, Z_6^{(1)}; Z_1^{(2)}, Z_2^{(2)}, \cdots, Z_6^{(2)}; \cdots; Z_1^{(8)}, Z_2^{(8)}, \cdots, Z_6^{(8)}; Z_1^{(9)}, Z_2^{(9)}, Z_3^{(9)}, Z_4^{(9)}$,将 128 位的用户选择密钥循环左移 25 位,由此产生的 128 比特密钥分成 8 个子块,它们被选作随后的 8 个密钥子块;将上述第二次所产生的 128 位再循环左移 25 比特,并将所得的 128 位密钥分成 8 个子块,这个过程可重复到 52 个 16 位密钥子块完全产生为止。

IDEA 的解密过程本质上与加密过程相同,惟一不同的是解密密钥子块 $K_i^{(r)}$ 是从加密密钥子块 $Z_i^{(r)}$ 按下列方式计算出来的:

$$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = ((Z_1^{(10-r)})^{-1}, -Z_3^{(10-r)}, -Z_2^{(10-r)}, (Z_4^{(10-r)})^{-1})$$

$$r = 2,3,\cdots,8$$

$$(K_1^{(r)}, K_2^{(r)}, K_3^{(r)}, K_4^{(r)}) = ((Z_1^{(10-r)})^{-1}, -Z_2^{(10-r)}, -Z_3^{(10-r)}, (Z_4^{(10-r)})^{-1})$$

$$r = 1,9$$

$$(K_5^{(r)}, K_6^{(r)}) = (Z_5^{(9-r)}, Z_6^{(9-r)}) \quad r = 1,2,\cdots 8$$

其中 Z^{-1} 表示模$(2^{16}+1)$的乘法逆,亦即 $Z \odot Z^{-1} = 1$, $-Z$ 表示 Z 的模 2^{16} 的加法逆,亦即 $-Z \boxplus Z = 0$。

IDEA 的设计理念可归纳如下:

(1) IDEA 的设计主要考虑针对 16 位为单位的处理器。因此无论明文、密钥都是分成 16 位为一个单元进行处理。

(2) IDEA 使用了 3 种简单的函数运算,因此在执行时可以达到非常快的操作。在 33 MHz386 机器上运行,加密速度可以达到 880 kbit/ s。经过特殊设计的 VLSI 芯片,更可以达到 55 Mbit/ s 的速度。

(3) IDEA 采用 3 种非常简单的函数,混合操作,以达到混淆目的。而相对地,DES 采用经过特殊设计的 S-盒,而对这些 S-盒的分析又不对外公开,相比之下,IDEA 的安全性评估较易被大家接受。

(4) IDEA 的整体设计非常规律。MA 运算器及变换运算器重复使用在系统上。因此非常适合 VLSI 实现。

由于 IDEA 的密钥长度是 DES 的 2 倍,所以在强力攻击 Brute - force Attack 下,其安全性要比 DES 强。

8.9.2　RC6 算法

RC6 是 RSA 公司提交给 NIST 的一个候选算法,它是在 RC5 的基础上设计的。RC6 继承了 RC5 的优点,使用 4 个寄存器,并加进 32 比特的整数乘法,用于加强扩散特性。RC6 更精确的表示是 RC6 - $w/r/b$,其中字长为 w 比特,r 为加密轮数,b 为加密密钥用字节表示的长度。这里规定: $w = 32$, $r = 20$, $b = 16(24,32)$。RC6 用到了下列几种基本运算: (1)整数模 2^w 加和减,分别表示为“ + ”和“ - ”。(2)比特字的逐位模 2 加,表示为

“ī ”。(3)整数模 2^w 乘,表示为“ × ”。(4)循环左移 ROL,右移 ROR,分别表示为《《,》》。

加密过程(见图 8 .19):把 128 比特明文放入 4 个 32 比特的寄存器 A,B,C,D 之中。

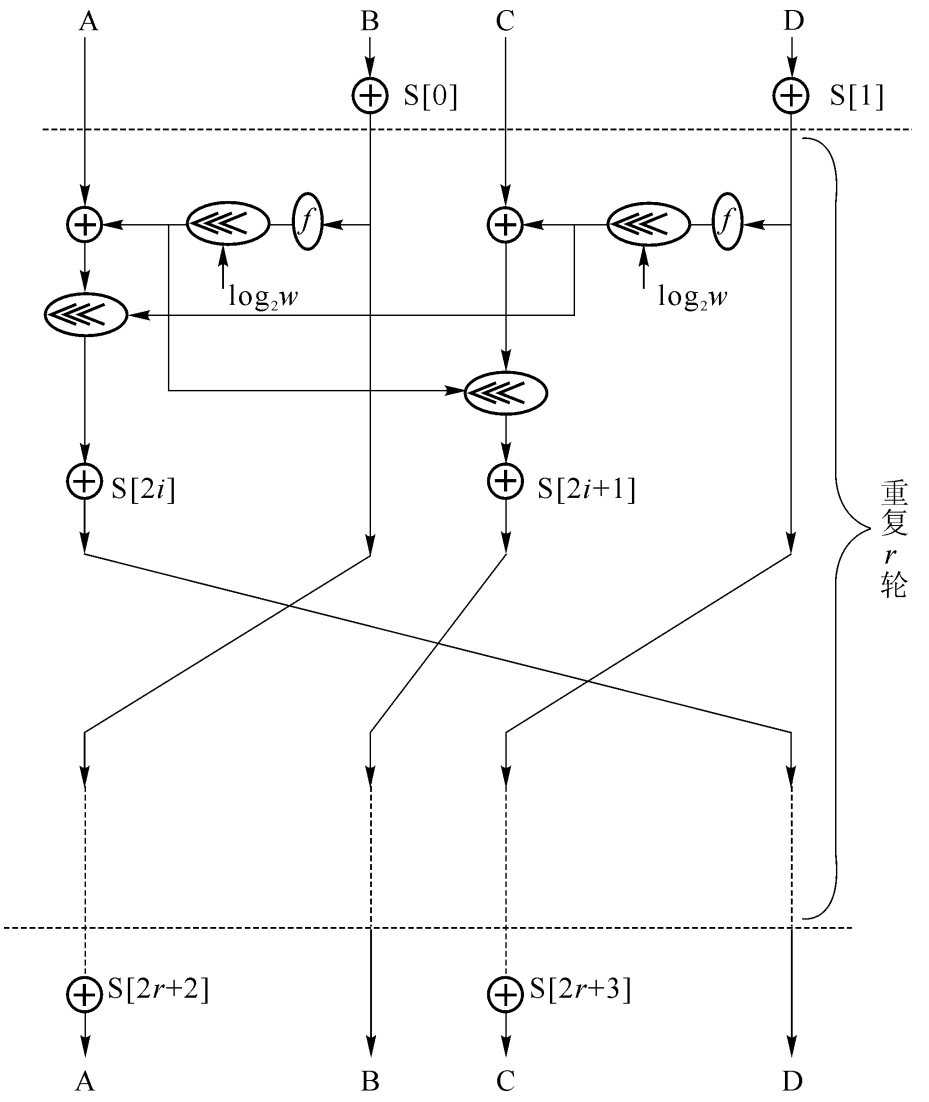


图 8 .19 RC6 的加密过程

B = B + S[0], D = D + S[1];
For i = 1 to r do
t = ROL(B × (2B + 1), $\log_2$ w)
u = ROL(D × (2D + 1), $\log_2$ w)
A = ROL(Aī t, u) + S[2i]
C = ROL(Cī u, t) + S[2i + 1]
(A, B, C, D) = (B, C, D, A)
A = A + S[2r + 2], C = C + S[2r + 3]

(A,B,C,D)即为密文。

解密过程:把 128 比特明文放入 4 个 32 比特的寄存器 A,B,C,D 之中。

A = A - S[2r + 2], C = C - S[2r + 3]

For　i = r　to　1　do

(A, B, C, D) = (D, A, B, C)

u = ROL(D × (2D + 1), $\log_2 w$)

t = ROL(B × (2B + 1), $\log_2 w$)

C = ROR(C - S[2i + 1], t) ⊕ u

A = ROR(A - S[2i], u) ⊕ t

B = B - S[0], D = D - S[1]

(A, B, C, D)即为明文。

密钥扩展方案:在密钥扩展中用到了两个常数 P_{32} 和 Q_{32}, P_{32} = B7E15163(十六进制), Q_{32} = 9E3779B9(十六进制)。首先,将种子密钥 K 输入 c 个 w 比特字的阵列, $L[0]$ … $L[c-1]$,若不够,用 0 字节填充,其中 c 为 $8b/w$ 的整数部分。

S[0] = P_w

For　i = 1　to　2r + 3　do

S[i] = S[i - 1] + Q_w

A = B = i = j = 0

v = 3 × max{c, 2r + 4}

For　s = 1　to　v　do

A = S[i] = ROL(A + B + S[i], 3)

B = L[j] = ROL(A + B + L[j], A + B)

i = (i + 1)　mod(2r + 4)

j = (j + 1)　mod c

输出即 S[0], S[1], …, S[2r + 3]即为子密钥。

值得一提的是,与大多数加密算法不同,RC6 算法在加密过程中不需要查找表,加之算法中的乘法运算也可以用平方代替,所以该算法对内存的要求很低,这使得 RC6 特别适合在单片机上实现。比如 IC 卡,它内部集成的高速缓存,代价高昂,算法所需的内存少,就能大大降低制作成本。

RC6 的简单性是非常吸引人的,尤其是便于在有限的时间范围内进行安全性分析。就目前的分析,对 RC6 算法最有效的攻击是强力攻击。当然,由于分组长度和密钥长度都至少是 128 比特,穷举法并不可行。对 20 轮的 RC6,用线性分析法至少需要 2^{155} 个明文,用差分分析法至少需要 2^{238} 个明文。

注 记

分组密码是现代密码学的重要分支之一,也是许多其他密码组件的基础。分组密码可以转化为流密码,例如 OFB 模式的分组密码是实现自同步流密码的一个流行的方法。分组密码也可被用来构造其他组件,例如 Hash 函数和消息认证码(相关章节有所阐述)。

分组密码的流行是和 DES 紧密联系的,DES 是分组密码的典型代表,也是第一个被公布出来的标准算法,有一段时间,DES 的存在使得接受其他算法很困难,可见 DES 在现代分组密码理论和应用中的地位和作用,所以我们不惜用大量的篇幅来介绍。有关 DES 的研究成果很多,本书参考文献[59]对此作了比较好的总结。

鉴于破解密码技术的快速演进,虽然到目前为止并无一致命的攻击法可以直接破解 DES 算法,但是这些发展已直接影响了 DES 密码系统的安全性,为此,美国和欧洲先后启动 AES 和 NESSIE 项目,有兴趣的读者可以前往 NIST 的网站 http://csrc.nist.gov/encryption/aes 和 NESSIE 的网站 http://www.cryptonessie.org,这些网站上有对算法及其相关数据的详细说明。实践证明,这些项目的公开执行又大大刺激了分组密码设计和分析的进步。基于新算法分析文献的缺乏和更新太快,我们只对 Rijndael 和 Camellia 算法的设计作介绍。当然,读者若对分析感兴趣,一些最新的密码学方面杂志和会议论文集,如 Journal of Cryptology 杂志,Crypt,Asiacrypt,Eurocrypt,Auscrypt,Fast Software Encryption 中皆有论述。

分组密码在过去二十多年取得了巨大的成就,比如,如何认识 Feistel 网络和 SP 网络两种结构的优缺点,如何设计效率高的线性变换层,如何证明密码抗差分和线性分析的能力,这些较新的主题,本章中皆有所论述。

大多数快的密码算法仍然以“尝试和纠错”方式提出,带来的问题是评估阶段比设计阶段更费力,NESSIE 的成就在于发展了一套密码体制的评估方法。每一种新的密码算法在被认定可信之前,必须被深入分析好几年,因此,我们常被提醒不要太快采用一个新的密码算法。

习 题 八

1.证明可以通过颠倒密钥方案用 DES 加密算法加密密文实现 DES 解密。

2.设 $\mathrm{DES}(x, K)$ 表示 DES 密码系统中用密钥 K 加密明文 x。假定 $y = \mathrm{DES}(x, K)$, $y' = \mathrm{DES}(c(x), c(K))$,其中 $c(\cdot)$ 表示按比特位取补。证明 $y' = c(y)$(即明文和密钥取补,则密文也是取补)。

3.在 DES 数据加密标准中:

明文 m = 0011　1000　1101　0101　1011　1000　0100　0010　1101　0101　0011　1001　1001　0101　1110　0111,

密钥 K = 1010　1011　0011　0100　1000　0110　1001　0100　1101　1001　0111　0011　1010　0010　1101　0011,

试求 L_1 与 R_1。

4. 已知 IDEA 密码算法中:

明文 m = 01011100　10001101　10101001　11011110　10101101　00110101　00010011　10010011;

密钥 K = 00101001　10100011　11011000　11100111　10100101　01010011　10100010　01011001　00100100　01011001　11001010　11100111　10100010　00101010　11010101　00110101,

求第一轮的输出与第二轮的输入。

5. 增强 DES 的一种方法是进行两次加密:给定两个密钥 K_1, K_2,记 $C = e_{K_2}(e_{K_1}(m))$。如果加密函数 e_{K_2} 与解密函数 d_{K_1} 是相同的,那么 K_1 和 K_2 称为对偶密钥(对两次加密来说,非常不希望出现这种情况,因为所得的密文与明文相同)。如果一个密钥是它自己的对偶密钥,这个密钥是自对偶的。

(1) 证明如果 C_0 是全 0 或全 1,且 D_0 也是全 0 或全 1,那么 K 是自对偶的。

(2) 证明下列密钥是自对偶的(十六进制表示):

0101010101010101

FEFEFEFEFEFEFEFE

(3) 证明下列密钥对是对偶的(以十六进制表示):

E001E001F101F101　01E001E001F101F1

FE1FFE1FFE0EFE0E　1FFE1FFE0EFE0EFE

E01FF01FF10EF10E　1FE01FE00EF10EF1

6 编程实现 AES(Rijndael)算法。

7.(1) 在 IDEA 的模乘运算中,为什么将模数取为 $2^{16}+1$ 而不是 2^{16} ?

(2) 在 IDEA 的模加运算中,为什么模数取为 2^{16} 而不是 $2^{16}+1$?

8. 对于分组长 128 比特,密钥长 128 比特的 Rijndael 算法,若已知

明文:3243f6a8885a308d313198a2e0370734;

密钥:2b7e151628aed2a6abf7158809cf4f3c,

试验证第一轮的各步输出:

字节代换:d42711aee0bf98f1b8b45de51e415230

行移变换:d4bf5d30e0b452aeb84111f11e2798e5

列变换:046681e5e0cb199a48f8d37a2806264c。

第 9 章　公钥密码学

9.1　公钥密码学思想

公钥密码体制的概念是由 Diffie 和 Hellman 于 1976 年提出的,也被称作非对称密码体制。在当时所有的经典的密码系统都是对称的密码体制,也就是通信双方共享一个秘密密钥,此密钥既能用于加密也能解密,这就导致了一些密钥分配的问题。例如:对于一个密码系统,我们必须通过安全信道将秘密密钥分配给通信用户,如果有 N 个通信者的话,则有 C_N^2 个秘密密钥必须交换。如果有一个秘密密钥泄漏了,则攻击者能够用此秘密密钥解密所有用此秘密密钥加密的消息。

公钥密码体制通过将密钥分成两部分而解决了上述的密钥分配的问题,即分成公开密钥和私有密钥。公开密钥被记录在一个公共的数据库;私有密钥被用户秘密地保存。这样,公开密钥能被用于加密信息,而在解密的过程中用户必须知道私有密钥。

一个公钥密码体制具体的描述如下:

定义 9.1　一个公钥密码体制是这样的一个5元组$\{M, C, K, E, D\}$,且满足如下的条件:

(1) M 是可能消息的集合;

(2) C 是可能密文的集合;

(3) 密钥空间 K 是一个可能密钥的有限集;

(4) 对每一个 $k=(k_1, k_2)\in K$,都对应一个加密算法 $E_{k_1}\in E$, $E_{k_1}: M\to C$ 和解密算法 $D_{k_2}\in D$, $D_{k_2}: C\to M$,满足对于任意的 $m\in M$,都有 $c=E_{k_1}(m)$, $m=D_{k_2}(c)=D_{k_2}(E_{k_1}(m))=m$;

(5) 对于所有的 $k\in K$,在已知 E_{k_1} 的情况下推出 D_{k_2} 是计算上不可能的。

对每一个 $k\in K$,函数 E_{k_1} 和 D_{k_2} 都是多项式时间可计算的函数。E_{k_1} 是一个公开函数, k_1 称作公钥;而 D_{k_2} 是一个秘密函数, k_2 称作私钥,由用户秘密地保存。

公钥密码体制的核心问题是 E_{k_1}, D_{k_2} 的设计,它必须满足条件(4)和(5),而条件(5)正是从计算复杂性理论的角度去考虑的。如果我们回顾一下单向函数的定义,易知条件

(4)和(5)正是单向函数的不严格定义。

一个公钥密码体制能够利用单向陷门函数 f 按如下的方式构造:用函数 f 去加密,而解密者只需用函数 f 的陷门信息即可求出 f 的逆 f^{-1},而攻击者所面对的问题就是直接计算 f^{-1}。我们可以从一些 NPC 问题(如背包问题)或者困难的数学理论方面的问题(尽管我们不能确切地知道它们的复杂性,但是研究者研究了这么多年仍未找出有效的解法)去构造单向函数。具体的关于单向函数的候选形式可参考单向函数一章。

这样看来,公钥密码体制的安全性完全建立在计算复杂性理论的基础上,而这个基础是否能够保证密码系统的安全呢?

在安全的加密算法中,合法用户应该能应用他们私人信息很容易地从密文恢复明文,而攻击者(无私人信息)却不能有效地对密文解密(在多项式时间内);而另一方面,一个非确定性图灵机却能很快地将密文解密(通过猜测私人信息)。因此,安全加密算法的存在,就意味着有这样一种工作(如"破译"加密算法),这种工作只能由非确定性多项式时间图灵机,而不能由确定性多项式时间图灵机(即使是随机的)来完成。换句话说,安全加密算法存在的一个必要条件是 $P \neq NP$,而 $P \neq NP$ 是否成立仍然是计算理论界的一个悬而未解的难题。

尽管 $P \neq NP$ 是现代密码编码学的一个必要条件,却不是一个充分条件。假设破译一个加密算法是 NPC 的,那么 $P \neq NP$ 意味着这种加密算法在最坏的情况下是难攻破的,但它仍然不能排除一个加密算法在多数情况下很易被攻破的可能性。实际上,可以构造一个破译问题为 NPC 的,但同时存在一个能以 99% 概率成功的破译算法,因此最坏情况下难破译不是安全性的一个好的评估。安全,要求在绝大多数情况下难破译,或至少"在通常情况下是难破译的"。

只考虑通常情况下难计算的 NP 问题的存在性也没有取得满意成绩。为了能应用在通常情况下难计算的问题,必须有能很快解决这些难题的辅助信息(陷门);否则,他们对合法用户也是难处理的,也就是建立在单向函数的基础上,但单向函数的存在性至今没有证明。虽然如此,密码学界还是普遍相信单向函数是存在的,而且还给出了一些经过分析、检验的、被认为是单向函数的例子。

公钥密码体制不仅解决了密钥分配的问题,我们将在后面的章节里看到它还为签名和认证提供了手段,具体的可参看后面的章节。这里举一个简单的例子说明它是怎样提供认证的,假设用户 A 给用户 B 发了一个消息 ,B 不能证明该消息是由 A 发出的。如果公钥密码体制满足附加的特性:对于所有的 $m \in M$, $D_{k_2}(E_{k_1}(m)) = E_{k_1}(D_{k_2}(m))$(也就是加解密是可交换的),此时认证的功能可被包含在加密的过程中:A 先用他的私钥 $D_{k_{2A}}$ 然后用接受者 B 的公钥 $E_{k_{1B}}$ 加密消息 m 产生密文 $c = E_{k_{1B}}(D_{k_{2A}}(m))$;而 B 先用他的私钥 $D_{k_{2B}}$ 然后利用 A 的公钥 $E_{k_{1A}}$ 解密。由于仅有 A 知道他的私钥,因此 B 就能确定消息确实是由 A 发出的。

9.2 RSA 公钥密码体制

9.2.1 RSA 体制

RSA 体制的具体描述见图 9.1。

> **系统参数**：取两个大素数 p 和 q，$n = pq$，$\phi(n) = (p-1)(q-1)$，随机选择整数 d，满足 $\gcd(d, \phi(n)) = 1$，$ed \equiv 1 \pmod{\phi(n)}$。
>
> 公开密钥：$k_1 = (n, e)$
>
> 私有密钥：$k_2 = (p, q, d)$
>
> **加密算法**：对于待加密消息 m，其对应的密文为 $c = E(m) \equiv m^e \pmod{n}$
>
> **解密算法**：$D(c) \equiv c^d \pmod{n}$

图 9.1 RSA 体制

下面证明该体制的正确性。

引理 9.1(欧拉定理)：若整数 a 和 m 互素，则

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

其中 $\phi(m)$ 是比 m 小但与 m 互素的正整数个数。

具体证明过程可参看附录。

证明 当 $(m, n) = 1$ 时，则由引理 9.1 可知 $m^{\phi(n)} \equiv 1 \pmod{n}$。

当 $(m, n) > 1$ 时，由于 $n = pq$，故 (m, n) 必含 p，q 之一。不妨设 $(m, n) = p$，则 $m = cp$，$(1 \leq c < q)$，由引理 9.1 知 $m^{\phi(q)} \equiv 1 \pmod{q}$。

因此，对于任何 k，总有 $m^{k(q-1)} \equiv 1 \pmod{q}$，$m^{k(p-1)(q-1)} \equiv (1)^{k(p-1)} \equiv 1 \pmod{q}$，即 $m^{k\phi(n)} \equiv 1 \pmod{q}$。

于是存在 h（h 是某个整数）满足 $m^{k\phi(n)} + hq = 1$。

由假定 $m = cp$，故 $m = m^{k\phi(n)+1} + hcpq = m^{k\phi(n)+1} + hcn$。

这就证明了 $m = m^{k\phi(n)+1} \pmod{n}$。

因此对于 n 及任何 m（$m < n$），恒有 $m^{k\phi(n)+1} = m \pmod{n}$。

所以，$D(c) \equiv c^d = m^{ed} = m^{k\phi(n)+1} = m \pmod{n}$。

9.2.2 RSA 的参数选择

1．p 和 q 应为强素数

定义 9.2 任一素数 p，若满足下列条件，称为强素数或一级素数。

(1) 存在两个大素数 p_1 及 p_2,使得 $p_1 \mid p-1$, $p_2 \mid p+1$。

(2) 存在 4 个大素数 r_1, r_2, s_1 和 s_2,使得 $r_1 \mid p_1-1$, $s_1 \mid p_1+1$, $r_2 \mid p_2-1$, $s_2 \mid p_2+1$。

我们称 r_1, r_2, s_1 和 s_2 为 3 级素数,称 p_1 及 p_2 为 2 级素数,称 p 为 1 级素数。

RSA 的安全性基于因子分解,故 n 的素因子 p 和 q 必须选择恰当,以确保因子分解在计算上(有效时间内)不可能实现。若 p 和 q 不是强素数,可通过下面的方法求得它们。

若 $p-1$ 有 k 个素因子,且可写成 $p-1=\prod_{i=1}^{k} p_i^{a_i}$,其中 a_i 为非负整数, p_i 为素数, $i=1,2,\cdots,k$。因为 $p-1$ 的素因子 $p_1, p_2, \cdots, p_k$ 均很小,不妨设 $p_i < B$(B 为已知小整数)。令正整数 a, R 满足: $a \geq a_i$, $R=\prod_{i=1}^{k} p_i^{a}$, $p-1 \mid R$。因为 p 为素数,任取小于 p 的正整数 t,不妨设 $t=2$,由费尔马定理(见附录)知, $2^R \equiv 1 \pmod{p}$。计算出 2^R 在模数 n 中的约化数 X($X \equiv 2^R \bmod n$),若 $X=1$,则令 $t=3$,计算 X,直到 $X \neq 1$,则 $\gcd(X-1, n)=p$,即分解 n 成功,对于 q 同理。因此,若 p 和 q 不为强素数,则可在有限时间内分解。

2．*p* 和 *q* 的位数差的问题

若 p 和 q 的位数相差不大,可通过下面的方法分解 n。设 $t=\frac{p+q}{2}$, $h=\frac{p-q}{2}$,由于 $(p-q)^2=(p+q)^2-4pq$,则 $t^2-h^2=n$。又由于 h 比较小,故可从大于 n的整数依次尝试 t,并通过 $t=\frac{p+q}{2}$, $h=\quad t^2-n$计算 p 和 q。

但 p 和 q 的位数又不能相差很大,若很大, 可通过尝试法,从小的素数用依次实验的办法分解 n。因此 p 和 q 的位数相差不能大也不能小,一般是几比特。

3．解密密钥 *d* 的选择

为了提高解密效率,应尽可能设想选择小的 d,但可通过尝试法验证这种追求片面的利益是有安全隐患的。比如已知明文 m,加密得密文 $c=m^e \bmod n$,可通过穷举 d 依次检验 $c^d=m \bmod n$ 是否成立。因此 d 不应选得太小,一般为 $d \geq n^{\frac{1}{4}}$。

4．加密密钥 *e* 的选择

由刚才的迭代法知, e 应满足它的阶尽可能的大,并且本身不能太小。对于明文 m,密文 $c=m^e \bmod n$,若 e 选得较小时,且 $m^e < n$,可直接将 c 开 e 次方得到明文 m。

5．模数 *n* 的使用限制

对于给定的模数 n,满足 $e_i d_i \equiv 1 \pmod{n}$的加/ 解密密钥对(e_i, d_i)很多,因此有人建议在通信中用同一个参数 n 以节约存储空间,但可证明这对于系统来说是有安全隐患的。

假设密钥对(e_i, d_i), (e_j, d_j)是同一参数 n 的两个不同参数对,当(e_i, e_j) = 1 时,对于

同一明文 m 分别加密得密文 $c_i = m^{e_i} \bmod n$，$c_j = m^{e_j} \bmod n$。由欧几里得算法知，必然存在整数 t 和 s，满足 $te_i + se_j = 1$，因此有 $c_i^t c_j^s = m^{te_i} m^{se_j} = m^{te_i + se_j} \equiv m \bmod n$。

9.2.3 概率素性检测

在建立 RSA 密码体制时，产生大的随机素数是必要的，在讨论素数的生成理论之前，我们先解决如下的一些问题：

(1) 如果每个人都需要一个不同的素数，那么素数是否足够用了？在数论中有一个著名的素数定理表明，对于正整数 N，不超过 N 的素数数目大约为 $N/\ln N$。也就是任选一个整数，它小于 N 且是素数的概率约为 $1/\ln N$。事实上，在长度为 512 位或略短一些的数中，有超过 10^{151} 个素数。宇宙中仅有 10^{77} 个原子，如果宇宙中的每一个原子从宇宙诞生到现在为止，每一秒都需要 10 亿个新的素数，那么总共需要 10^{109} 个素数，现在仍将剩下接近 10^{151} 个 512 位的素数。

(2) 是否会有两个人偶然地选择了同样的素数的情况？这种情况是不会发生的。从超过 10^{151} 个素数中选择相同的素数，发生这种情况的可能性比你获得抽奖时你的计算机恰好自然烧毁的可能性还要小。

(3) 如果有人建立了所有素数的数据库，难道他不能用这个数据库来破译公开密钥算法？是的，他可以，但他不会这样做。如果你能将 10 亿字节的信息存储在 1 克重的设备上，那么所有 512 位的素数的重量将超过 Chandrasekhar 限，导致系统崩溃，进入黑洞……那样的话，你将无论如何也不能重新找回你的数据。

如果因子的分解很困难，那么怎样才能使素数的生成容易些呢？这是一个判定 YES/NO 的问题，问题“n 是素数吗？”比复杂点的问题“n 的因子是什么？”更容易回答。因此如果你产生随机候选数 n，然后试着分解它们，从而找出素数是错误的方法，正确的方法是对产生的随机数检测是否是素数，可用可信度来测试一个数是否是素数。这里将利用描述的多项式时间 Monte Carlo 算法，用 Solovay - Strassen 算法进行素数测试。

定义 9.3 一个 YES - biased(偏向于“是”的) Monte Carlo 算法对一个判定问题来说是一个概率算法，其中“YES”回答是正确的，而“NO”回答则有可能不正确。一个 NO - biased(偏向于“不”的) Monte Carlo 算法有一个错误概率 ε，如果在一个实例中应回答“YES”，算法给出一个“NO”(不正确)的概率至多为 ε(当算法以给定的输入运行时，可用关于该算法所作的所有可能的随机选择来计算此概率)。

一个合数的判定问题可描述如下：

条件：一个正整数 $n \geq 2$。

问题：n 是合数吗？

注意，一个判定问题的算法仅需回答“YES”或“NO”，因此，对于合数问题，在 n 是合数的情况下不需找到它的分解。

下面描述 Solovay-Strassen 算法，它是一个错误概率为 1/2 的合数的 YES-biased

Monte Carlo 算法。因此如果算法回答“YES”,那么 n 是合数;反之,如果 n 是合数,那么回答“YES”的概率至少为 1/2。在进一步论述算法之前,需要阐述一些数论背景(可参考后面的附录数论一节)。

定义 9.4　如果 p 是素数,且 a 小于 p,若至少存在一个 $x\in[1, p-1]$ 满足 $x^2\equiv a(\mathrm{mod}\ p)$,则称 a 是模 p 的二次剩余(quadratic residue)。

定义 9.5　设 p 是一奇素数,对任何 $a\geq 0$,定义勒让德符号(Legendre symbol) $L(a, p)$ 为

$$L(a, p)=\begin{cases}0 & \text{如果 } a\equiv 0(\mathrm{mod}\ p)\\ 1 & \text{如果 } a \text{ 是模 } p \text{ 的二次剩余}\\ -1 & \text{如果 } a \text{ 是模 } p \text{ 的非二次剩余}\end{cases}$$

为了得到一种计算勒让德符号的有效算法,下面将介绍欧拉准则。

定理 9.1(欧拉准则)　设 p 是素数,那么 x 是模 p 的二次剩余当且仅当

$$x^{(p-1)/2}\equiv 1(\mathrm{mod}\ p)$$

证明　如果 x 是模 p 的二次剩余,则存在 $y\in[1, p-1]$ 满足 $y^2\equiv x(\mathrm{mod}\ p)$,又由于对素数 p 和任何 $x\neq 0(\mathrm{mod}\ p)$, $x^{p-1}\equiv 1(\mathrm{mod}\ p)$ 成立,得 $x^{(p-1)/2}\equiv (y^2)^{(p-1)/2}\equiv y^{p-1}\equiv 1(\mathrm{mod}\ p)$。

反之,假设 $x^{(p-1)/2}\equiv 1(\mathrm{mod}\ p)$,设 b 是一个模 p 的本原元,那么存在某个 i 使 $x\equiv b^i(\mathrm{mod}\ p)$,那么有 $x^{(p-1)/2}\equiv (b^i)^{(p-1)/2}\equiv b^{i(p-1)/2}\equiv 1(\mathrm{mod}\ p)$。因为 b 的阶为 $p-1$,故必有 $p-1\mid i(p-1)/2$,因此, i 为偶数,那么 x 的模 p 的平方根为 $\pm b^{i/2}$,即 x 是模 p 的二次剩余。

我们已经看到, $a^{(p-1)/2}\equiv 1(\mathrm{mod}\ p)$ 当且仅当 a 是模 p 的二次剩余。如果 a 是 p 的倍数,那么显然有 $a^{(p-1)/2}\equiv 0(\mathrm{mod}\ p)$,最后如果 a 是 p 的非二次剩余,因为 $a^{p-1}\equiv 1(\mathrm{mod}\ p)$,所以 $a^{(p-1)/2}\equiv -1(\mathrm{mod}\ p)$。因此提供了一种计算勒让德符号的有效算法。

推论 9.1　假设 p 是素数,那么 $L(a, p)\equiv a^{(p-1)/2}(\mathrm{mod}\ p)$

下面进一步地定义勒让德符号的一般形式。

定义 9.6　雅可比符号(Jacobi symbol),记作 $J(a, n)$,是勒让德符号的一般化表示,它定义在任意正整数 a 和奇整数 n 上。设 n 的素数因子分解式为 $p_1^{e_1}\cdots p_k^{e_k}$,则

$$J(a, n)=L(a, p_1)^{e_1}\times\cdots\times L(a, p_k)^{e_k}。$$

由定义可得雅可比符号的计算中的几种特殊情况:

(1) 雅可比符号 $J(0, n)=0$;

(2) 如果 n 是素数,且 n 能整除以 a,那么雅可比符号 $J(a, n)=0$;

(3) 如果 n 是素数,且 a 是模 n 的一个二次剩余,那么雅可比符号 $J(a, n)=1$;

(4) 如果 n 是素数,且 a 是模 n 的一个非二次剩余,那么雅可比符号 $J(a, n)=-1$。

幸运的是,可利用数论中的一些结果计算雅可比符号,避免了分解 n 后再计算的方法。计算雅可比符号的递归算法如下:

规则 1: $J(1, n) = 1$。

规则 2: $J(a \times b, n) = J(a, n) \times J(b, n)$, 特别是, 当 $m = 2^k t$ 时, 这里的 t 是奇数, 则

$$J(m, n) = J(2, n)^k J(t, n)。$$

规则 3: 如果 $(n^2 - 1)/8$ 是偶数, $J(2, n) = 1$, 否则为 -1。

规则 4: $J(a, n) = J((a \bmod n), n)$。

规则 5: $J(a, b1 \times b2) = J(a, b1) \times J(a, b2)$。

规则 6: 如果 a 和 b 都是奇数, 且它们的最大公因子是 1, 那么:

规则 6a: $J(a, b) = J(b, a)$, 如果 $(a-1)(b-1)/4$ 是偶数;

规则 6b: $J(a, b) = -J(b, a)$, 如果 $(a-1)(b-1)/4$ 是奇数。

Solovay-Strassen 素性测试算法如图 9.2 所示。

1 选择一随机整数 a, 满足 $a \in [1, n-1]$

2 如果 $J(a, n) = a^{(n-1)/2} \bmod n$ 则

回答"n 是素数"

否则 回答"n 是合数"

图 9.2 对一个奇整数 n 的 Solovay-Strassen 素性测试

下面提出一个问题: 如果运行此算法 m 次仍判定数 n 是素数, 那么相信 n 真是素数的概率究竟有多大呢?

下面将用概率论中的贝叶氏定理来说明这个问题。

首先定义下面的随机变量: a 表示事件"一个规定尺寸的随机奇数 n 是合数", b 表示事件"算法连续 m 次后仍回答 n 是素数", 则要求的问题是估算出 $\mathrm{Prob}(a \mid b)$ 的大小。

设 $n \in [N, 2N]$, 根据素数定理可得, 介于 N 与 $2N$ 之间的素数的数目大约为

$$\frac{2N}{\ln 2N} - \frac{N}{\ln N} \approx \frac{N}{\ln N} \approx \frac{n}{\ln n}。$$

又因为 $[N, 2N]$ 内的奇整数的数目为 $N/2 \approx n/2$, 所以可估计事件 a 发生的概率为

$$\mathrm{Prob}(a) \approx 1 - \frac{n}{\ln n} \Big/ \frac{n}{2} = 1 - \frac{2}{\ln n},$$

由贝叶氏定理得

$$\mathrm{Prob}(a \mid b) = \frac{\mathrm{Prob}(a)\mathrm{Prob}(b \mid a)}{\mathrm{Prob}(b)} = \frac{\mathrm{Prob}(a)\mathrm{Prob}(b \mid a)}{\mathrm{Prob}(a)\mathrm{Prob}(b \mid a) + \mathrm{Prob}(a')\mathrm{Prob}(b \mid a')}$$

$$\approx \frac{\mathrm{Prob}(b \mid a)\left(1 - \frac{2}{\ln n}\right)}{\mathrm{Prob}(b \mid a)\left(1 - \frac{2}{\ln n}\right) + \frac{2}{\ln n}} = \frac{\mathrm{Prob}(b \mid a)(\ln n - 2)}{\mathrm{Prob}(b \mid a)(\ln n - 2) + 2}$$

$$\leq \frac{2^{-m}(\ln n - 2)}{2^{-m}(\ln n - 2) + 2} = \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}},$$

其中 a' 代表 a 的补事件。

9.2.4　RSA 的攻击

RSA 的安全性完全依赖于大数分解问题。从技术上来说它是这样的，假设 RSA 的安全性依赖于大数分解问题，而从未从数学上证明需要分解 n 才能从 c 和 e 中计算出 m，可能会发现一种完全不同的方法来对 RSA 进行密码分析。因而，如果这种新方法能让密码分析者推算出 d，它也可作为分解大数的一种新方法。

也可猜测 $(p-1)(q-1)$ 的值来攻击 RSA。这种攻击没有分解 n 容易。

对那些持极端怀疑态度的人来说，有一些 RSA 的变型已被证明和大数分解同样困难，显示出从 RSA 加密的密文中恢复某一比特与恢复出整个文本同样困难。

有些攻击是针对 RSA 的实现。它们不是攻击基本的算法，而是攻击协议，仅会使用 RSA 而不重视它的实现是不够的，实现细节也很重要。

情况 1　攻击者在发方 A 的通信过程中进行窃听，设法成功选取了一个用他的公开密钥加密的密文 c，攻击者想揭示出明文。从数学上讲，他想得到 m，这里：

$$m = c^d。$$

为了恢复出 m，他首先选取一个随机数 r，满足 r 小于 n。他得到 A 的公钥 e，然后计算：

$$x = r^e \bmod n,$$

$$y = xc \bmod n,$$

$$t = r^{-1} \bmod n。$$

如果 $x = r^e \bmod n$，那么 $r = x^d \bmod n$。

现在，攻击者让 A 用他的私钥对 y 签名（关于签名的概念在第 10 章里介绍），以便解密 y。（A 必须对消息，而非消息的 Hash 值签名）记住，A 以前从未见过 y。A 发送给攻击者：

$$u = y^d \bmod n。$$

现在攻击者计算：

$$tu = r^{-1} y^d = r^{-1} x^d c^d = c^d = m \bmod n。$$

因此攻击者获得了明文 m。

情况 2　T 是一个公开的计算机公证人。如果 A 打算让一份文件被公证，他将它发送给 T，T 将它用 RSA 进行数字签名，然后发送回来（这里没有使用单向 Hash 函数，T 用他的私钥加密整个消息）。

M 想让 T 对一个 T 本来不愿签名的消息签名，或许他有一个假的时间标记，或是另外的人所为。不管是何种理由，如果允许 T 选择的话，他是绝不会对它签名的。让我们将这个消息称作 m'。

首先，M 选取任意一个值 x，计算 $y = x^e \bmod n$。他能很容易地获得 e，这是 T 的公开

密钥,必须公开以便用来验证他的签名。然后,他计算 $m = ym' \bmod n$,并将 m 发送给 T 并让 T 对它签名。T 回送 $m^d \bmod n$,现在 M 计算$(m^d \bmod n)\, x^{-1} \bmod n$,它等于$(m')^d \bmod n$,是 m 的签名。

实际上,M 有几种方法可用来完成相同的事,他们利用的缺陷都是指数运算保持了输入的乘积结构,即是:

$$(xm)^d = x^d m^d \bmod n。$$

情况 3 E 想让 A 对 m_3 签名。他产生两份消息 m_1, m_2 满足

$$m_3 \equiv m_1\, m_2 \pmod n。$$

如果他能让 A 对 m_1 和 m_2 签名,他能计算 m_3:

$$m_3^{\,d} = (m_1^{\,d} \bmod n)(m_2^{\,d} \bmod n) \pmod n。$$

因此绝对不要对一个陌生人提交给你的随机消息进行签名,如果要签名,应总是首先利用一个单向 Hash 函数对消息进行 Hash 变换。

9.3 ELGamal 公钥密码体制和离散对数问题

已经在单向函数那章介绍了离散对数问题是单向函数的候选形式,下面要介绍的 ELGamal 公钥密码体制就是基于离散对数问题的。

ELGamal 公钥密码体制的描述如图 9.3 所示。

系统参数:设 p 是一素数,满足 Z_p 中离散对数问题是难解的,g 是 Z_p^* 中的本原元,选取 $a \in [0, p-1]$,计算 $\beta = g^a \bmod p$,则

私有密钥:$k_2 = a$

公开密钥:$k_1 = (g, \beta, p)$

加密算法:对于待加密消息 m,随机选取数 $k \in [0, p-1]$,则密文为 $e_{k_1}(m, k) = (y_1, y_2)$,

其中,$y_1 = g^k \bmod p$, $y_2 = m\beta^k \bmod p$。

解密算法:消息接收者收到密文(y_1, y_2)后,解密得明文为 $d_{k_2}(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p$

图 9.3 ELGamal 公钥密码体制

ELGamal 公钥密码体制的正确性由下面的式子可看出:

$$y_2 (y_1^{\,a})^{-1} = m\beta^k g^{-ka} = mg^{ka} g^{-ka} = m \bmod p。$$

这里是基于乘法群 Z_p^* 建立的 ELGamal 公钥密码体制,实际上可基于任何离散对数问题难处理的群实现 ELGamal 公钥密码体制。下面将描述一般有限群 G 的离散对数问

题,然后介绍推广的 ELGamal 公钥密码体制。

设有限群 G 中的运算符为 * ,则其上的离散对数问题的描述可见图 9 .4。

条件:设群 G 是运算符为 * 的有限群, $\alpha \in G$ 和 $\beta \in H$, 其中 $H = \{\alpha^i \mid i \geq 0\}$, 即 H 是由 α 生成的子群。

问题:能否找到惟一的整数 $a \in [0, |H| - 1]$, 满足 $\alpha^a = \beta$, α^a 是 a 个 α 的 * 运算。

我们记 $a = \log_\alpha \beta$。

图 9 .4　在群 $G(*)$ 中的离散对数问题

设群 G 是一运算符为 * 的有限群,子群 H 是由 α 生成的,且满足其上的离散对数问题是难处理的。具体的推广的 ELGamal 公钥密码体制的描述见图 9 .5。

系统参数:设群 G 是运算符为 * 的有限群, $\alpha \in G$, $H = \{\alpha^i \mid i \geq 0\}$, (即 H 是由 α 生成的子群), H 满足其上的离散对数问题是难处理的。选取 $a \in H$,计算 $\beta = \alpha^a$,则

私有密钥: a

公开密钥: α, β

加密算法:对于待加密消息 m,随机选取数 $k \in [0, |H| - 1]$,则密文为 $e_{k_1}(m, k) = (y_1, y_2)$,其中, $y_1 = \alpha^k$, $y_2 = m * \beta^k$。

解密算法:消息接收者收到密文 (y_1, y_2) 后,解密得明文为

$$d_{k_2}(y_1, y_2) = y_2 * (y_1^a)^{-1}$$

图 9 .5　推广的 ELGamal 公钥密码体制

我们可仿照 ELGamal 公钥密码体制的正确性的证明方法证明它的正确性。

下面我们假设 p 是素数, α 是 Z_p 中的本原元, p 和 α 是固定的,则离散对数问题可表示成:对于任意的数 $\beta \in [1, p - 1]$ 是否能找到某个数 $a \in [0, p - 2]$ 满足 $\alpha^a = \beta \bmod p$。

易知离散对数问题能在时间 $O(p)$ 和空间 $O(1)$ 内通过穷举搜索找到解(忽略离散对数因子)。可通过计算所有可能值 α^a,然后按第二坐标排成有序对 $(a, \alpha^a \bmod p)$,我们能在 $O(1)$ 时间内采用 $O(p)$ 个计算和 $O(p)$ 个存储来解离散对数问题(再次忽略离散对数因子),下面描述的两个求模 p 的离散对数的算法,第一个是 Shanks 提出的时间 - 存储算法(见图 9 .6),第二个是 Pohlig - Hellman 算法(见图 9 .7)。

1 . Shanks 算法

1 .计算 $\alpha^{mi} \bmod p$, $i \in [0, m-1]$, 按第二个坐标排序 m 个有序对$(i, \alpha^{mi} \bmod p)$, 得到表 T_1。

2 .计算 $\beta\alpha^{-j} \bmod p$, $j \in [0, m-1]$, 按第二个坐标排序 m 个有序对$(j, \beta\alpha^{-j} \bmod p)$, 得到表 T_2。

3 .从 T_1、T_2 中分别各自寻找一对$(i, y) \in T_1$ 和$(j, y) \in T_2$, 定义 $\log_\alpha \beta = mi + j \bmod (p-1)$。

图 9 .6　离散对数问题的 Shanks 算法

Shanks 算法的正确性可这样证明:当$(i, y) \in T_1$ 和$(j, y) \in T_2$ 时, 有

$$\alpha^{mi} = y = \beta\alpha^{-j} \bmod p \quad \alpha^{mi+j} = \beta \bmod p,$$

即 $\log_\alpha \beta = mi + j \bmod (p-1)$。

该算法在 $O(m)$ 个存储空间运行, 且在 $O(m)$ 时间内是不难实现的(忽略离散对数因子)。

2 . Pohlig - Hellman 算法

一个 Pohlig - Hellman 算法的描述见图 9 .7, 在这个算法中, α 是 Z_p^* 的本原元, q 是素数, 且有

$$p - 1 = 0 \bmod q^c,$$

$$p - 1 \neq 0 \bmod q^{c+1},$$

则算法用来计算满足 $\log_\alpha \beta = \sum_{i=0}^{c-1} a_i q^c \bmod q^c$ 的$(a_0, a_1, a_2, \cdots, a_{c-1})$。

1 .计算 $\gamma_j = \alpha^{(p-1)j_q - 1} \bmod p$, $j \in [0, q-1]$。

2 .i, β_i 的初始值为 $i \leftarrow 0$ 和 $\beta_i \leftarrow \beta$

While　$i \leq c-1$　do

计算 $\delta = \beta_i^{(p-1)q^{-(j+1)}} \bmod p$,

找一个 j 满足 $\delta = \gamma_j$,

$a_i \leftarrow j$, $\beta_{i+1} \leftarrow \beta_i \alpha^{-a_i q^i}$, $i \leftarrow i+1$。

图 9 .7　计算模 q^c 的离散对数问题的 Pohlig - Hellman 算法

关于这个算法的正确性的证明留作作业请大家自己练习。

9 .4　基于纠错码的公钥密码体制

纠错码和密码学是两门不同的学科, 在 20 世纪 70 年代以前, 它们几乎互不相关, 各

自独立地向前发展。随着 1976 年 W .Diffie 和 M .E .Hellman 发表了在密码学领域具有里程碑的文章——《密码学新方向》,提出了新的密码思想,使得密码体制的安全性建立在某个难解的数学问题之上,即 NPC 问题之上。1978 年 Berlekamp 等人证明了纠错码中的一些译码问题是 NPC 问题,并提出了一个著名的猜想:对于一般的线性码 C,求它的最小 Hamming 重量是一个 NPC 问题(此猜想于 1997 年被证明)。这两项成果建立起纠错码和密码学相结合的理论基础。1978 年,McEliece 设计出第一个基于纠错码的公钥密码体制,被称作 McEliece 公钥密码体制,他利用了一般线性分组码的译码问题是 NPC 问题,且 Goppa 码具有快速译码算法的特点。此后关于纠错码中的 NPC 问题的研究和利用纠错码构造各种密码体制和签名认证方案得到了迅速发展。

下面介绍一些纠错码理论中的 NPC 问题。

问题 1　陪集重量问题

实例　GF(2)上的 $m\times n$ 阶矩阵 $\boldsymbol{H}$,一个向量 $\boldsymbol{s}\in F_2^m$ 和一个正整数 $w>0$。

问题　是否存在一个向量 $\boldsymbol{x}\in F_2^n$,满足 $w(x)\leq w$ 且 $\boldsymbol{H}x^t=\boldsymbol{s}$?(这里的 $w(x)$为 x 的 Hamming 重量,以下均同)。

问题 2　重量分布问题

实例　GF(2)上的 $m\times n$ 阶矩阵 $\boldsymbol{H}$ 和一个正整数 $w>0$。

问题　是否存在一个向量 $\boldsymbol{x}\in F_2^n$,满足重量为 w 且 $\boldsymbol{H}x^t=0$?

问题 3　最小距离问题

实例　GF(2)上的 $m\times n$ 阶矩阵 $\boldsymbol{H}$ 和一个正整数 $w>0$。

问题　是否存在一个非零向量 $\boldsymbol{x}\in F_2^n$,满足重量$\leq w$ 且 $\boldsymbol{H}x^t=0$?

下面介绍 McEliece 密码体制,McEliece 密码体制的具体描述见图 9 .8。

系统参数:设 $\boldsymbol{G}$ 是 GF(2)上的[n, k, d] Goppa 码的生成矩阵,其中 $n=2^m$, $d=2t+1$, $k=n-mt$。

设明文集合为 $GF(2)^k$,密文集合为 $GF(2)^n$

随机选取 GF(2)上的 $k\times k$ 阶可逆矩阵 $\boldsymbol{S}$ 和 $n\times n$ 阶置换矩阵 $\boldsymbol{P}$,令 $\boldsymbol{G}'=\boldsymbol{SGP}$ 则

私有密钥:$\boldsymbol{S}$, $\boldsymbol{G}$, $\boldsymbol{P}$

公开密钥:$\boldsymbol{G}'$

加密算法:对于待加密的明文 $m\in GF(2)^k$,其对应的密文为 $c=m\boldsymbol{G}'+z$,这里的 z 是 $GF(2)^n$ 上重量为 t 的随机向量。

解密算法:接收者收到密文 c 后,计算 $cP^{-1}=m\boldsymbol{SGPP}^{-1}+z\boldsymbol{P}^{-1}=m\boldsymbol{SG}+z'$,由于 $\boldsymbol{P}$ 是置换阵,故 $w(z')=w(z)=t$,利用 Goppa 的快速译码算法将其译成 $m'=m\boldsymbol{S}$,则密文 c 对应的明文为 $m=m'\boldsymbol{S}^{-1}$。

图 9 .8　McEliece 体制

由上面可知该体制的一个明文对应可能的 C_n^t 个密文中的某一个,故为概率性加密

体制。

下面通过一个简单的例子说明 McEliece 体制。

例 9.1 对于 Hamming 码$[7,4,3]$,其生成矩阵为

$$\boldsymbol{G}=\begin{pmatrix}1&0&0&0&1&1&0\\0&1&0&0&1&0&1\\0&0&1&0&0&1&1\\0&0&0&1&1&1&1\end{pmatrix}。$$

假设用户 A 选择

$$\boldsymbol{S}=\begin{pmatrix}1&1&1&0\\1&0&1&0\\1&1&0&1\\0&1&1&0\end{pmatrix},$$

$$\boldsymbol{P}=\begin{pmatrix}0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\1&0&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&1&0\end{pmatrix},$$

和

则公开密钥为 $\boldsymbol{G}'=\boldsymbol{SGP}=\begin{pmatrix}1&0&0&1&1&0&0\\1&0&1&1&0&1&0\\0&0&1&1&1&0&1\\1&1&1&0&1&0&0\end{pmatrix}$。

设用户 A 随机选取一重量为 1 的错误向量 $\boldsymbol{z}=(1\ 0\ 0\ 0\ 0\ 0\ 0)$来加密。设待加密的明文为 $m=(0\ 1\ 0\ 1)$,则密文为

$$c=m\boldsymbol{G}'+z=(0\ 1\ 0\ 1)\begin{pmatrix}1&0&0&1&1&0&0\\1&0&1&1&0&1&0\\0&0&1&1&1&0&1\\1&1&1&0&1&0&0\end{pmatrix}+(1\ 0\ 0\ 0\ 0\ 0\ 0)$$

$$=(1\ 1\ 0\ 1\ 1\ 1\ 0)。$$

当接受者 B 收到密文 c 后,计算

$$c' = c\boldsymbol{p}^{-1} = (1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 0)\begin{pmatrix}0&0&1&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&1\\0&0&0&1&0&0&0\end{pmatrix}$$

$$= (1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1)。$$

通过快速译码算法得到 $m' = (1\ \ 1\ \ 0\ \ 0)$。

故 $m = m'\boldsymbol{S}^{-1} = (1\ \ 1\ \ 0\ \ 0)\begin{pmatrix}1&0&0&1\\1&1&0&0\\1&1&0&1\\0&1&1&1\end{pmatrix} = (0\ \ 1\ \ 0\ \ 1)$。

与用户 A 发送的明文相符。

在 McEliece 最初的论文中，他建议 $n = 1024$，$t = 50$，且 $k = 524$，这是要达到安全要求的最小值。

尽管该算法是最早的公开密钥算法之一，并且对它没有成功的密码分析结果，但是它从未获得密码学界的广泛接受。该方案比 RSA 快两到三个数量级，但亦存在着若干问题。公开密钥太庞大：为 2^{19} 比特长。数据扩展太大，密文是明文的两倍长。

随着 McEliece 体制的提出，提出了一种新的方法，即建立基于纠错码理论之上的密码体制，使密码学能充分利用编码方面的丰富的成果。1986 年，Niederreiter 提出了另一个基于纠错码理论的公钥密码体制，被称作 N 公钥密码体制。1994 年，王新梅等人证明了 McEliece 体制与 N 公钥密码体制的等价性。人们针对 McEliece 体制的缺陷纷纷提出了一些它的变体，具体地可参见本书的参考文献[39]。

值得一提的是，1991 年，两位俄罗斯的密码学家声称采用一些参数破解了 McEliece 系统。这篇论文没有证据证实他们的宣称，大多数密码学家对此结果持怀疑态度。

当前关于建立基于纠错码的密码体制很多是基于传统的 Hamming 距离的。1985 年俄罗斯学者 Gabidulin 提出了秩距离(rank distance)的概念，实践证明构造基于秩距离的密码体制和认证系统的安全性比基于 Hamming 距离的更高。因此怎样构造更多的基于秩距离的密码体制成了一个急待研究的问题。关于秩距离的概念和性质可参考本书参考文献[39]。

9.5 椭圆曲线公钥体制

9.5.1 椭圆曲线

椭圆曲线理论是代数几何、数论等多个数学分支的一个交叉点，一直被认为是纯理论学科。近年来，由于公钥密码学的产生与发展，该学科也找到了它的应用领域。在RSA密码体制的基础性问题——大整数分解和素性检测——的研究方面，椭圆曲线是一个强有力的工具。特别地，以椭圆曲线上的(有理)点构成的Abel群为背景结构，实现各种密码体制已是公钥密码学领域的一个重要课题。由于椭圆曲线密码体制本身的优点，自20世纪80年代中期被引入以来，椭圆曲线密码体制逐步成为一个十分令人感兴趣的密码学分支，1997年以来形成了一个研究热点，特别是移动通信安全方面的应用更是加快了这一趋势。

定义9.7 设 p 是一个大于3的素数，在 Z_p 上的椭圆曲线 $y^2 = x^3 + ax + b$ 由一个基于同余式 $y^2 = x^3 + ax + b \bmod p$ 的解集 $(x, y) \in Z_p \times Z_p$ 和一个称为无穷远点的特定点 O 组成，这里的 $a, b \in Z_p$ 是二个满足 $4a^3 + 27b^2 \neq 0 \bmod p$ 的常数。

我们通过定义椭圆曲线 E 上的点的适当运算，从而使它们构成阿贝尔群，这个运算记为+，且运算都在 Z_p 中完成。具体定义如下：设 $P = (x_1, y_1) \in E$，$Q = (x_2, y_2) \in E$，若 $x_1 = x_2$ 且 $y_1 = -y_2$，那么 $P + Q = O$；否则 $P + Q = (x_3, y_3)$，这里的 $x_3 = \lambda^2 - x_1 - x_2$，$y_3 = \lambda(x_1 - x_3) - y_1$，$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{如果 } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{如果 } P = Q \end{cases}$。

最后对于所有的 $P \in E$，定义 $P + O = O + P = P$。

可证明 E 中的点在这种定义的运算下构成了阿贝尔群，其单位元为 O。对于 E 中元素 (x, y)，其逆元为 $-(x, y) = (x, -y)$。

例9.2 Z_{23} 上的一个椭圆曲线。

令 $y^2 = x^3 + x + 1$ 是 Z_{23} 上的一个方程 $(a = b = 1)$。我们可以对每个 $x \in Z_{23}$，计算 $x^3 + x + 1 \bmod 23$，试着解方程 $y^2 = x^3 + x + 1 \bmod 23$。对于给定的 x，可用Euler判别法来测试 $z = x^3 + x + 1 \bmod 23$ 是否是一个二次剩余。同时，对于素数 $p = 3 \bmod 4$，有一个计算模 p 的剩余公式，利用这个公式，二次剩余 z 的平方根是：$\pm z^{(11+1)/4} \bmod 23 = \pm z^3 \bmod 23$。

通过计算得该椭圆曲方程在 Z_{23} 上的解(即椭圆曲线上的点)：

(0,1) (0,22) (1,7) (1,16) (3,10) (3,13) (4,0) (5,4) (5,19) (6,4) (6,19) (7,

11)　(7,12)　(9,7)　(9,16)　(11,3)　(11,20)　(12,4)　(12,19)　(13,7)　(13,16)　(17,3)　(17,20)　(18,3)　(18,20)　(19,5)　(19,18), O(无穷远点)

举一个例子说明椭圆曲线上的点的运算。例如计算点(3,10) + (6,4):

$$\lambda = \frac{(y_2 - y_1)}{x_2 - x_1} = 21,$$

$$x_3 = \lambda^2 - x_1 - x_2 = 4 - 3 - 6 = 18,$$

$$y_3 = \lambda(x_1 - x_3) - y_1 = 21(3 - 18) - 10 = 20,$$

因此, (3,10) + (6,4) = (18,20)。

9 5 2　椭圆曲线密码体制

在介绍椭圆曲线密码体制之前, 先叙述一个定理。

定理 9 2　(Hasse 定理)　如果 E 是定义在域 GF(p)上的椭圆曲线, N 是 E 上的点 $(x, y) \in$ GF(p)的数目, 则

$$| N - (p + 1) | \leq 2 \quad p。$$

该定理是由 Artin 提出的猜想, 后来 Hasse 证明了。具体的证明请参考有关文献。

已在 9 3 节里定义了一般的离散对数问题, 下面定义椭圆曲线上的离散对数问题。

定义 9 8　椭圆曲线离散对数问题: 给定群中的点 P 与 Q, 求数 k, 使得 $kP = Q$。

目前解决该问题的最快算法比解决标准的离散对数问题的最快算法要慢得多。

图 9 9 介绍了 ELGamal 密码体制的椭圆曲线形式。

> 系统参数: 设 E 是一个定义在 Z_p($p > 3$ 的素数)上的椭圆曲线, 令 $\alpha \in E$, 则由 α 生成的子群 H 满足其上的离散对数问题是难处理的, 选取 a, 计算 $\beta = a\alpha$, 则
>
> 私有密钥: $k_2 = a$
>
> 公开密钥: $k_1 = (\alpha, \beta, p)$
>
> 加密算法: 对于明文 x, 随机选取正整数 $k \in Z_{p-1}$,
>
> $e_{k_1}(x, k) = (y_1, y_2)$, 其中 $y_1 = k\alpha$, $y_2 = x + k\beta$。
>
> 解密算法:
>
> $d_{k_2}(y_1, y_2) = y_2 - ay_1$。

图 9 9　椭圆曲线密码体制

就以例 9 2 的椭圆曲线来说明椭圆曲线密码体制的工作细节。

设 $\alpha = (6,4)$, 取私钥 $a = 3$, 有 $\beta = 3\alpha = (7,12)$。

若 A 想加密明文 $x = (5,4)$, 如果随机选取 $k = 2$, 那么

$$y_1 = k\alpha = (13,7),$$

$$y_2 = x + k\beta = (5,4) + 2(7,12) = (5,4) + (17,3) = (5,19),$$

所以密文为 $y=(y_1,y_2)=((13,7),(5,19))$。

B 收到密文 y 后,解密如下:

$$x=y_2-ay_1=(5,19)-3(13,7)=(5,4)。$$

椭圆曲线密码体制有如下的一些特点:

(1) 在安全性相当的前提下,可使用较短的密钥。一般认为,p 元域上的椭圆曲线密码体制,当 p 的长度为 160 bit 时,其安全性相当于 RSA 使用 1024 bit 模数。这适合于那些存储空间及处理能力有限的设备,例如特别适合于智能卡。

(2) 椭圆曲线密码体制是建立在一个不同于大整数分解及素域乘法群离散对数问题的数学难题之上。自公钥密码产生以来,人们基于各种数学难题提出了大量的密码方案,但能够经受住时间考验而广泛为人们所接受的只有基于大整数分解及离散对数问题的方案,且不说这两种问题受到亚指数算法的严重威胁,就如此狭窄的数学背景来说,也不能不引起人们的担忧,寻找新的数学难题作为密码资源早就是人们努力的一个方向。

(3) 椭圆曲线资源丰富,同一个有限域上存在着大量不同的椭圆曲线,这为安全性增加了额外的保证。

(4) 在执行速度方面,由于椭圆曲线上的一次群运算最终化为域上的不超过 15 次乘法运算,因而便于实现。目前难以对椭圆曲线密码体制与现存密码体制,比如 RSA、DSA 等作出准确的定量比较,粗略地说,椭圆曲线密码体制较对应的离散对数体制要快,且在签名和解密方面较 RSA 快,但在签名验证和加密方面较 RSA 慢。

(5) 安全性显然是任何密码体制的必备条件,椭圆曲线密码体制的安全性分析因而也引起了各国密码学家及有关部门的关注与重视,但成果却并不丰硕。也许这可视为椭圆曲线密码体制具有高强度的一种证据,因此,大多数密码学家对这种密码体制的前景持乐观态度。

9.6 其他公开密钥密码体制

9.6.1 Goldwasser - Micali 概率公开密钥密码系统

1984 年,Goldwasser 和 Micali 提出了概率公开密钥密码系统(Probabilistic Encryption)的概念。在此类系统中,对于固定的明文 m 和加密密钥 k_1,消息发送者在加密时需引入随机数 r_i,使得密文为 $C_i=e_{k_1}(m,r_i)$,当引入不同的 r_i 时,明文 m 所对应的密文也就不同。这反映了该体制的一个很重要的特点是相同的明文 m 可对应多个不同的密文 C_i。而在解密时,这些不同的密文 C_i(相同明文对应的)在相同的解密密钥 k_2 的作用下得到同一个明文 m,即对所有的 C_i,有 $d_{k_2}(C_i)=m$。

下面介绍的是 Goldwasser-Micali 概率公开密钥密码系统,具体描述如图 9.10 所示。

系统参数: A 随机选取两个大的强素数 p, q, 计算 $n = pq$, 随机选取数 y 且满足 $J(y, n) = 1$, $J(y, p) = J(y, q) = -1$, 则

公开密钥: y, n

私有密钥: p, q

加密算法: 若 B 想给 A 传送明文 m, m 表示成二进制形式, 即 $m = (m_1, m_2, \cdots, m_k) \in F_2^k$,

对于 $m_i (0 \leq i \leq k)$, 任意选取 $x_i \in [1, n-1]$,

当 $m_i = 1$ 时, B 计算 $C_i = yx_i^2 \bmod n$; 当 $m_i = 0$ 时, B 计算 $C_i = x_i^2 \bmod n$, 则密文为 $E(m) = (C_1, C_2, \cdots, C_k)$。

解密算法: A 收到 B 传来的密文 $(C_1, C_2, \cdots, C_k)$ 后, 对每一个 $C_i (0 \leq i \leq k)$, 计算 $J(C_i, p)$ 和 $J(C_i, q)$。

当 $J(C_i, p) = J(C_i, q) = -1$, 则 $m_i = 1$; 当 $J(C_i, p) = J(C_i, q) = 1$, 则 $m_i = 0$。

$(m_1, m_2, \cdots, m_k)$ 即为明文 m。

图 9.10　Goldwasser-Micali 概率公开密钥密码系统

该体制的正确性可由下面的式子证明:

当 $m_i = 0$ 时, $J(C_i, p) = J(x_i^2, p) = 1$, $J(C_i, q) = J(x_i^2, q) = 1$。

当 $m_i = 1$ 时, $J(C_i, p) = J(yx_i^2, p) = J(y, p)\,J(x_i^2, p) = -1$, $J(C_i, q) = J(yx_i^2, q) = J(y, q)\,J(x_i^2, q) = -1$。

Goldwasser-Micali 概率公开密钥密码系统的安全性分析与讨论

(1) 在 Goldwasser-Micali 密码系统中, 对于明文 m, 随着随机数 x_i 的选择的不同, 而可对应不同的密文, 故其为概率密码系统。对于攻击者来说, 当他截获到密文 $(C_1, C_2, \cdots, C_k)$ 时, 他能求出 $J(C_i, n)$, 但当 $m_i = 0$, $J(C_i, n) = J(x_i^2, n) = 1$, 当 $m_i = 1$, $J(yx_i^2, n) = J(y, n)\,J(x_i^2, n) = 1$, 攻击者无法获得其他的任何信息, 而对 A 来说, 因为他拥有私有密钥 p 和 q, 可求出 $J(C_i, p)$, $J(C_i, q)$, 从而得到明文。

(2) 从传输效率来看, 由于明文对应至 $[1, n-1]$ 之间, 故其效率为 $\frac{1}{|n|}$, $|n|$ 为 n 的长度。由于此系统的安全性是基于因子分解的, 故 $|n|$ 必须大于等于 512 位。因此本系统的传输效率 $\leq 1/512$, 效率非常差。Blum 和 Goldwasser 接着提出了一效率较佳的概率公开密钥密码系统, 他们的做法是这样的: 信息发送者首先利用公开密钥密码系统, 将一随机数产生器的初始状态 S 送给接收者。接着将明文与随机数产生器的输出异或得密文发送给接收者。由此看出, 该方法除了初始状态 S 以公开密钥密码系统的方式分配外, 其余与序列加密系统均相同。He 和 Lu 提出一种递归式的概率密码系统, 其传输效率为 $\frac{t}{t+k-1}$, 这里的 t 为明文长度, k 为安全参数。后来 Harn 和 Kiesler 提出一种类似于 He 和 Lu 的系统, 但其加解密速度较快, 且容许一次递归时做多位加解密(He 和 Lu 的系统, 一次递归时只有一位的加解密)。

9.6.2 Merkle-Hellman 背包公钥密码体制

Merkle-Hellman 背包公钥密码体制是基于子集和问题是 NPC 问题而设计的，已在有关单向函数的章节里介绍了子集和问题。下面将介绍一种特殊的序列——超递增序列，背包公钥密码体制就是基于它的一个特殊的性质设计的。

定义 9.9　超递增序列

一个序列 $(s_1, s_2, \cdots, s_n)$ $(s_i \in N)$ 被称作超递增序列当且仅当对任意的 $i \in [2, n]$，有

$$s_i > \sum_{j=1}^{i-1} s_j 。$$

当一个序列表是超递增时，那它上的子集和问题能在搜索时间为 $O(n)$ 内求解，并且解是惟一的。图 9.11 描述了具体的算法，其中 $T = \sum_{i=1}^{n} x_i s_i$。

1. for i = n down to 1 do
   if $T \geq s_i$ then
   $T \leftarrow T - s_i, x_i \leftarrow 1$
   else $x_i \leftarrow 0$
2. if $\sum_{i=1}^{n} x_i s_i = T$ then $x = (x_1, x_2, \cdots, x_n)$ 是解。

图 9.11　解超递增序列中的子集和问题算法

有了上面的准备工作，便可描述 Merkle-Hellman 背包公钥密码体制了。具体的描述如图 9.12 所示。

系统参数：A 任选一个超递增序列 $T = (t_1, t_2, \cdots, t_n)$，整数 W, N 满足 $N > \sum_{i=1}^{n} t_i$ 且 $\gcd(W, N) = 1$。

将递增序列 T 转化成为伪随机序列 $G = (g_1, g_2, \cdots, g_n)$，这里的 $g_i (i = 1, 2, \cdots, n)$ 满足

$$g_i = t_i W \bmod N。$$

公开密钥：$k_1 = G$，

私有密钥：$k_2 = (T, N, W)$。

加密算法：对于待加密消息 $m = (m_1, m_2, \cdots m_n) \in F_2^n$，B 想将之传给 A，B 找出 A 的公开密钥 G，则密文为 $V = \sum_{i=1}^{n} m_i g_i$。

解密算法：收到密文 V 后，计算

$$V' = VW^{-1} = \sum_{i=1}^{n} m_i t_i W W^{-1} = \sum_{i=1}^{n} m_i t_i \bmod N。$$

由于 T 是超递增序列，利用超递增序列算法就可算出明文 m。

图 9.12　Merkle-Hellman 背包公钥密码体制

9.6.3　有限自动机公开密钥密码体制

有限自动机公开密钥密码体制是由我国学者陶仁骥发明的,它的思想与 RSA 体制类似。RSA 是基于分解两个强素数的乘积是困难的而设计的;而此类体制是基于分解两个有限自动机的合成也是困难的而构造的,尤其是当其中的一个或两个为非线性时,难度就会更大。

陶仁骥所利用的结论主要是:当且仅当某些非线性自动机(准自动机)有某种梯形矩阵结构时,它们具有弱的可逆性,如果是其他矩阵结构(甚至是线性结构),就不具有该特性。在公开密钥密码体制中,私有密钥就是一些可逆的准自动机,也是线性的自动机;公开密钥就由私有自动机逐项相乘获得的自动机。待传送的信息经过公开的自动机后加密,解密即是让密文经过上述自动状态机的各组件的反向路径来得到。

此系统的性能有点像 McEliece 体制,它们运行都比 RSA 快,且都需要更长的密钥。要获得 512 比特的 RSA 相似的安全性,密钥长应考虑为 2 792 比特;而要获得 1024 比特的 RSA 相似的安全性,密钥长就应考虑为 4 152 比特。在一台 33 MHz 的 80486 电脑上运行前一种情况,系统加密数据速率为 20 869 bit/ s,解密数据速率为 17 117 bit/ s。

陶仁骥已公开了 3 个算法。第一个为 FAPKC0。这是一个弱的系统,它使用了线性组件,且主要是用来解释用的。两个重要的系统为 FAPKC1 和 FAPKC2,每一个都是采用线性和非线性的组件,后者更复杂,研究出来用以支持基于身份鉴别的操作。

注 记

公钥密码学是现代密码学的很重要的组成部分,本章主要介绍了公钥密码的思想和几种基于计算复杂性的公钥密码体制。本章主要参考了本书参考文献[15],[39]和[30]。有关公钥密码的思想可参看本书参考文献[4],有关 RSA 体制 20 年来研究成果的最好总结可参看本书参考文献[66],有关更多的基于纠错码的公钥密码体制可参看本书参考文献[39],文献[67]、[68]很好地介绍了有关基于椭圆曲线的公钥密码体制。

习 题 九

1.一个公钥密码体制的一般定义是怎样的?

2.回顾一下以下概念的定义及性质:强素数、超递增序列、概率公开密钥密码系统。

3.证明 9.3 节中的 Pohlig-Hellman 算法的正确性。

4.在使用 RSA 公钥系统中,如果截取了发送给其他用户的密文 $C = 10$,若此用户的

公钥为 $e=5$，$n=35$，请问明文的内容是什么？

5.对于椭圆曲线：$y^2=x^3+17$，已知其上的点 $P_1=(-2,3)$，$P_2=(2,5)$，求：

① P_1+P_2；

② $2P_1$；

③ $-P_1$。

6.Diffie-Hellman 密钥交换算法如下：Alice 和 Bob 协商好一个大素数 p 和 Z_p^* 上的本原元 $g,1<g<p$，p 和 g 无须保密可为网络上的所有用户共享。当 Alice 和 Bob 要进行保密通信(交换密钥)时，他们可以按如下步骤来做：

(1) Alice 选取大的随机数 x(作为其私钥)，并计算 $X=g^x \pmod p$(作为其公钥)；

(2) Bob 选取大的随机数 x'(作为其私钥)，并计算 $X'=g^{x'} \pmod p$(作为其公钥)；

(3) Alice 将 X 传送给 Bob，Bob 将 X' 传送给 Alice；

(4) Alice 计算 $K=(X')^x \pmod p$；Bob 计算 $K'=(X)^{x'} \pmod p$。

易见，$K=K'=g^{xx'} \pmod p$，则 Alice 和 Bob 的密钥交换完成，K(或 K')是他们的共享密钥。试问当 $p=11$ 且 $g=2$ 时：

(1) 如果用户 A 的公钥为 $Y_A=9$，则 A 的私钥 X_A 是多少？

(2) 如果用户 B 的公钥为 $Y_B=3$，则共享的密钥 K 是多少？

7.利用求离散对数问题的 Shanks 算法求 GF(23)域上的 $\log_5 4$。

8.研究题：试着找一个基于某个候选单向函数的公钥密码体制。

第 10 章　数字签名

随着信息技术的不断发展,电子商务也日趋走进人们的生活,它对传统的商务活动带来了巨大的冲击,突出的标志就是增加了贸易机会,降低了贸易成本,简化了贸易流程,提高了贸易效率。电子商务极大地改变了商务模式,带动了经济结构的变革,同时,电子商务也是一个充满挑战的领域,这种挑战在很大程度上来源于对可使用的安全技术的信赖。数字签名作为一种特殊的安全机制成为电子商务安全性的重要组成部分。

数字签名(Digital Signature)是一种以电子形式给一个消息签名的方法,是只有信息发送方才能够进行的签名,是任何其他人都无法伪造的一段数字串,这段特殊的数字串同时也是对签名真实性的一种证明。在电子信息的传输过程中,通过数字签名来达到与传统手写签名相同的效果。数字签名有如下一些特点:

(1) 签名是不可伪造的;

(2) 签名是可靠的;

(3) 签名是不可重用的;

(4) 签名是不可改变的;

(5) 签名是不可抵赖的。

下面定义数字签名的一般化模型。

定义 10.1　一个签名方案是一个5元组(M, A, K, S, V),满足如下的条件:

(1) M 是一个可能消息的有限集;

(2) A 是一个可能签名的有限集;

(3) 密钥空间 K 是一个可能密钥的有限集;

(4) 对每一个 $k=(k_1, k_2)\in K$,都对应一个签名算法 $\mathrm{Sig}_{k_2}\in S$ 和验证算法 $\mathrm{Ver}_{k_1}\in V$。

每一个 $\mathrm{Sig}_{k_2}: M\to A$ 和 $\mathrm{Ver}_{k_1}: M\times A\to\{\mathrm{TRUE}, \mathrm{FALSE}\}$是一个对每一个消息 $x\in M$和每一个签名 $y\in A$ 满足下列方程的函数:

$$\mathrm{Ver}(x, y)=\begin{cases}\mathrm{TRUE} & \text{当 } y=\mathrm{Sig}_{k_2}(x)\\ \mathrm{FALSE} & \text{当 } y\neq\mathrm{Sig}_{k_2}(x)\end{cases}。$$

对每一个 $k\in K$,函数 Sig_{k_2}和 Ver_{k_1}都是多项式时间可计算的函数。Ver_{k_1}是一个公

开函数，k_1 称作公钥；而 Sig_{k_2} 是一个秘密函数，k_2 称作私钥，由用户秘密地保存。

10.1 基于 RSA 和离散对数的签名体制

10.1.1 RSA 签名方案

RSA 签名方案的算法如图 10.1 所示。

系统参数：设 $n=pq$，且 p 和 q 是两个大素数，则 $M=A=Z_n$，定义 $K=\{(n,e,p,q,e)\}$。

这里 e 和 d 满足 $ed\equiv 1(\bmod \phi(n))$（$\phi(\cdot)$为欧拉函数）。

公开密钥：n，e

私有密钥：p，q，d

签名算法：$\mathrm{Sig}_{k_2}(x)=x^d \bmod n$

验证算法：$\mathrm{Ver}(x,y)=\mathrm{TRUE}$　$y^e=x(\bmod\ n)$，$(x,y)\in Z_n\times Z_n$。

图 10.1　RSA 签名方案

对于消息对 (x,y)，若 y 是 x 有效签名，则 $y^e=x^{ed}=x^{t\phi(n)+1}=x^{t\phi(n)}x=x \bmod n$，即验证算法成立。

这样，用户 A 若想用 RSA 签名方案对消息 x 签名，它只需公开他的公钥 n 和 e，由于签名算法 Sig_{k_2} 是保密的，因此 A 是惟一能产生签名的人，任何要验证用户 A 签名的用户只需查到 A 的公钥即可验证签名。

下面提一个有意思的话题：怎样才能实现签名和公钥加密的组合？我们有两种常用的方法。假定通信双方为 A 和 B，方法 1：对于明文 x，A 计算他的签名 $y=\mathrm{Sig}_A(x)$，然后利用 B 的公开加密函数 E_B，加密得 $Z=E_B(x,y)$，将密文 Z 传给 B，当 B 收到密文 Z 后，他首先用他的解密函数 D_B 来解密得到 $(x,y)=D_B(Z)=D_B(E_B(x,y))$，然后利用 A 的验证算法来检查 $\mathrm{Ver}_A(x,y)=\mathrm{TRUE}$ 是否成立。方法 2：对于明文 x，A 先利用用户 B 的公开加密函数 E_B，加密得 $z=E_B(x)$，然后再用他的签名算法签名得 $y=\mathrm{Sig}_A(E_B(x))$，然后将消息对 (z,y) 传送给用户 B，当 B 收到消息对后，解密 z 得到 x，然后利用验证算法来检查 $\mathrm{Ver}_A(z,y)=\mathrm{TRUE}$ 是否成立。对于方法 2，一个很重要的潜在问题是：如果攻击者 C 截获到消息对 (z,y)，它能通过自己的签名来代替 A 的签名。$y=\mathrm{Sig}_C(E_B(x))$（C 完全可以在不知道明文 x 的前提下对密文 z 签名）。然后将 (z,y) 传给 B，B 通过利用 C 的验证算法 Ver_C 来验证，且 B 可能推断明文来自于 C，因为这种潜在的缺点，许多人推荐用先签名后加密的方法。

10.1.2　ELGamal 签名方案及其一般化的模型

1985 年,ELGamal 提出了一种基于离散对数问题的数字签名方案,被称作 ELGamal 签名方案。它是专为签名而设计的,不同于既用作加密又用作签名体制的 RSA。这个方案的改进已被美国国家标准和技术研究所(NIST)采纳为数字签名标准。

ELGamal 签名方案像 ELGamal 体制一样,是非确定型的,这意味着对于任意给定的消息,可产生多个有效的签名,验证算法必须能接受合法的有效签名中的任何一个,ELGamal 签名方案的描述如图 10.2 所示。

系统参数:设 p 是一大素数,g 是 Z_p^* 的一个生成元,定义 $K=\{(p,g,y,x): y=g^x \bmod p\}$ 其中 $x\in Z_p^*$。

公开密钥:y, p, g

私有密钥:x

签名算法:对于 $(k_1,k_2)=(p,g,y,x)$,随机数 $k\in Z_p^*$ 和待签消息 m,定义 $\mathrm{Sig}_{k_2}(x,k)=(r,s)$。这里的 $r=g^k \bmod p$; $s=(m-xr)k^{-1} \bmod (p-1)$。

(r,s) 即为生成的签名。

验证算法:$\mathrm{Ver}(m,r,s)=\mathrm{TRUE}$　$y^r r^s = g^m \bmod p$。

图 10.2　ELGamal 签名方案

这个签名体制的正确性可由以下等式证明:$y^r r^s = g^{rx} g^{ks} = g^{rx+m-rx} = g^m \bmod p$。

ELGamal 签名方案的安全性分析及讨论:

(1) 本方案是基于离散对数问题的,若离散对数问题被解决了,则由 y 和 g 可得私有密钥 x,本系统完全被破译。

(2) 对于随机数 k,应注意两方面的情况:首先,k 不能泄露,因为如果知道 k 的话,那么计算 $x=(m-sk)r^{-1} \bmod (p-1)$ 是容易的;其次,随机数不能重复使用,假设 k 用来对两个不同的消息签名,(r,s_1) 是 m_1 的签名,(r,s_2) 是 m_2 的签名,有

$$m_1 = xr + ks_1 \bmod (p-1),$$

$$m_2 = xr + ks_2 \bmod (p-1),$$

两式相减得 $m_1-m_2=k(s_1-s_2) \bmod (p-1)$。设 $d=\gcd(s_1-s_2, p-1)$,因为 $d \mid p-1$ 且 $d \mid s_1-s_2$,可证明 $d \mid m_1-m_2$,定义 $m'=\dfrac{m_1-m_2}{d}$, $s'=\dfrac{s_1-s_2}{d}$, $p'=\dfrac{p-1}{d}$,则有 $m'=ks' \bmod p'$ 且 $\gcd(s',p')=1$,那么 $k=m'(s')^{-1} \bmod p'$, $k=tp'+m'(s')^{-1}$ $(0\le t\le d-1)$,通过测试 $r=g^k \bmod p$ 来确定哪一个是 k。

(3) 由于验证算法只是核实等式 $y^r r^s = g^m \bmod p$ 是否成立,故可考虑通过伪造使该等式成立的签名 (r,s) 来攻击此方案。下面提出一种攻击方法,至于更一般的方法将在后面讨论。攻击者选择两随机数 u 和 w,满足 $0\le u\le p-2$, $0\le w\le p-2$ 且

gcd $(w, p-1)=1$。

计算　　$r=g^u y^{-w} \bmod p$, $s=rw^{-1} \bmod (p-1)$, $m=us$。

由以上三式得　$y^r r^s = y^r (g^u y^{-w})^s = y^r g^{us} y^{-ws} = y^r g^{us} y^{-r} = g^{us} = g^m \bmod p$。

因此(r, s)为 m 的合法签名。这种方法虽说能产生有效的伪造签名,但是在没有解决离散对数问题时,攻击者是不能对任意消息进行成功的伪造签名的,因此这不意味着ELGamal 签名方案受到很大的威胁。为了抵抗这种所有签名体制都存在的类似的攻击方法,一般采用单向 Hash 函数(在 Hash 函数一章里介绍)与签名方案相结合的方法。

下面介绍一般的 ELGamal 型签名方案(这里按照系统初始化、签名算法和验证算法的过程介绍):

(1) 系统初始化

设 p 是一大素数, g 是 Z_p^* 的一个生成元, $h()$是一单向 Hash 函数,用户选取任意的 $x \in Z_p^*$ 作为秘密密钥,计算 $y=g^x \bmod p$ 作为公开密钥。系统中的用户可以获得公开密钥 p, g, y。

(2) 签名算法

用户随机选取数 $k \in Z_p^*$,对于消息 m 一般的 ELGamal 型签名方程为:

$$Ax = Bk + C \bmod (p-1),$$

其中(A, B, C)为参数$(h(m), r, s)$的排列。这里的 $r=g^k \bmod p$,由签名方程可解出 s, (r, s)即为用户对消息 m 的签名。

(3) 验证算法

当接受者收到消息组(m, r, s)后,由(m, r, s)和签名方程算出(A, B, C),验证等式 $y^A = r^B g^C \bmod p$ 是否成立。若成立,则为有效签名;反之,则为无效签名。

由此可知,一般的 ELGamal 型签名方案可通过$(h(m), r, s)$的数学组合得到,例如$(h(m), r, s)$的单一参数重排,$(h(m), r, s)$的任两者组合的重新排列,参数 A, B, C 的符号的改变,但不是所有的组合都是安全的数字签名方案,Harn 和 Xu 给出了安全的数字签名方案的设计规则,并列出了所有符合这种设计规则的 ELGamal 型签名方案,如表 10.1所示。

表 10.1　一般 ELGamal 型签名方案

	签名方程	验证方程
(1)	$mx = rk + s \bmod (p-1)$	$y^m = r^r g^s \bmod p$
(2)	$mx = sk + r \bmod (p-1)$	$y^m = r^s g^r \bmod p$
(3)	$rx = mk + s \bmod (p-1)$	$y^r = r^m g^s \bmod p$
(4)	$rx = sk + m \bmod (p-1)$	$y^r = r^s g^m \bmod p$
(5)	$sx = rk + m \bmod (p-1)$	$y^s = r^r g^m \bmod p$
(6)	$sx = mk + r \bmod (p-1)$	$y^s = r^m g^r \bmod p$

续 表

	签名方程	验证方程
(7)	$mrx = k + s \bmod (p - 1)$	$y^{mr} = rg^s \bmod p$
(8)	$x = mrk + s \bmod (p - 1)$	$y = r^{mr}g^s \bmod p$
(9)	$sx = k + mr \bmod (p - 1)$	$y^s = rg^{mr} \bmod p$
(10)	$x = sk + rm \bmod (p - 1)$	$y = r^s g^{mr} \bmod p$
(11)	$rmx = sk + 1 \bmod (p - 1)$	$y^{mr} = r^s g \bmod p$
(12)	$sx = mrk + 1 \bmod (p - 1)$	$y^s = r^{mr} g \bmod p$
(13)	$(r + m)x = k + s \bmod (p - 1)$	$y^{(m+r)} = rg^s \bmod p$
(14)	$x = (m + r)k + s \bmod (p - 1)$	$y = r^{(m+r)} g^s \bmod p$
(15)	$sx = k + (m + r) \bmod (p - 1)$	$y^s = rg^{(m+r)} \bmod p$
(16)	$x = sk + (r + m) \bmod (p - 1)$	$y = r^s g^{(m+r)} \bmod p$
(17)	$(r + m)x = sk + 1 \bmod (p - 1)$	$y^{(m+r)} = r^s g \bmod p$
(18)	$sx = (r + m)k + 1 \bmod (p - 1)$	$y^s = r^{(m+r)} g \bmod p$

10.1.3　DSS

数字签名标准(即 DSS)是由美国国家标准和技术研究所(NIST: National Institute of Standard and Technology)于 1991 年 8 月公布的,该标准是为了使接收者能验证数据的完整性和数据发送者的身份而制定的,此标准采用的算法被称作 DSA 算法,它是 ELGamal 签名体制的变形。详细的描述如图 10.3 所示。

系统参数:设 p 是一 512 位到 1024 位的大素数,它满足 Z_p 中的离散对数问题是难解决的,q 是 160 位长的素数,且 $q \mid p - 1$, $g \in Z_p^*$ 是 Z_p 域中的 q 次单位根。

定义 $K = \{(p, q, g, y, x) : y = g^x \bmod p\}$, $h(\cdot)$为一个公开的 Hash 函数。

公开密钥: p, q, g, y。

私有密钥: x。

签名算法:对于随机数 $k \in Z_p^*$ 和待签消息 $m \in Z_p^*$,计算 $r = (g^k \bmod p) \bmod q$。

$s = (h(m) + xr)k^{-1} \bmod q$,消息对$(r, s)$即为生成的签名。

验证算法: $\mathrm{Ver}(m, r, s) = \mathrm{TRUE} \quad (y^{er}g^{er} \bmod p) \bmod q = r$,

其中 $e_1 = h(m)s^{-1} \bmod q$, $e_2 = rs^{-1} \bmod q$。

图 10.3　DSA

DSA 公布之后,引起了学术界与商业界的热烈反应,赞成和反对的都有。这里引用

本书参考文献[30]的描述来说明专家学者对 DSA 的看法。

1.专家学者对 DSA 的正面评价

(1) 标准的公布已显示美国政府终于确认公开密钥密码系统的使用;

(2) 此标准(DSA)所产生的签名长度较短(仅有 320 位);

(3) 在签名体制中,对于 r 的计算可预先做好,可减少产生签名的时间;

(4) DSS 标准在金融服务业特别有用;

(5) 假若 DSS 被实现的话,只需最小的成本;

(6) 此系统的密钥产生相当快速;

(7) 这是世界上惟一由政府所公布的签名算法;

(8) 这个算法,应该被采纳为联邦信息处理标准(FIPS)。

2.专家学者对 DSA 的负面评价

(1) DSA 可能侵犯了两个现存的美国专利(Schnorr 签名方法及 Public Key Partness,其中 PKP 为 Diffie 及 Hellman 的公开密钥分配方法)。这是 NIST 急需解决的首要问题。

(2) DSS 并不和现存的国际标准兼容。国际标准组织,诸如 ISO、CCITT 及 SWIFT 已接受 RSA 为标准。但 DSS 并不和它兼容,这个问题将导致美国工业界必须同时采用两种不同标准的困扰。

(3) 根据 Rivest 教授(RSA 发明人之一)的说法,DSS 的验证程序中有点小毛病,就是当 s 刚好为 0 时,会产生 1 除以 0 的情况。但 NIST 似乎并没有注意到这件事。这一问题的解决方法其实只要在签名程序时规定只要 $s=0$ 出现的时候,就再另选一个 k 值,产生另一组 s 不为 0 的签名文就可以了。如果忽略掉这一问题的话,将会危及安全性,因为若 $s=0$,就很容易可以算出使用者个人的秘密密钥为 $x=\frac{-h(m)}{r} \bmod q$。

(4) DSS 的验证程序和 RSA 比起来大约慢了 100 倍。

(5) DSS 的安全性所依靠的数学难题和长久被研究的求离散对数问题并不完全一样,其安全性还需要一段时间来做彻底的审验才能够确定。

(6) DSS 并未很明确地指明 k 值的选取,其实只要一个 k 值泄露就足以危害到系统的安全。所以 k 值的产生方式对系统有致命的影响。

(7) 在 DSS 的最初版本中,p 的长度被指定为‘固定’的 512 位,这是很多专家矛头所指向的地方。很多人建议最好的方式是采取一个固定的上限及下限。在这个范围中,因不同的应用场合,采取一个合理最佳的值,这一建议已经被 NIST 所采纳。

(8) NIST 在选择某种系统作为国家标准的程序上,有瑕疵落人口实;而且 NIST 公布 DSS 之后,只保留 3 个月时间来接受各种意见。在异议、攻击不断出现之后,才又延长为 6 个月。DSS 出自美国国家安全局(NSA)之手,由 NIST 采用并公布,工业界没有任何插手的余地。

(9) 有人怀疑美国政府公布此一标准的动机并不单纯,其背后的重要目的可能是为

了美国政府某些部门的执法方便,而订出这种专家眼中问题丛生的标准。

(10) DSS 并未明确指出数字签名系统所需的“Hash 函数”及“密钥交换”的做法准则(但最近美国政府已公布了一套 Hash 函数的标准,而且 NIST 说明 DSS 并不是要用来做密钥交换的功用,会另有更符合联邦密码准则的密钥交换技术的出现)。

10.1.4　Lamport 签名方案

Lamport 签名方案是一种利用单向 Hash 函数构造的一次签名方案,这里的“一次”是指仅有一个消息能被签名,当然,该签名能被验证任意多次。具体的 Lamport 签名方案的描述如图 10.4 所示。

系统参数:设 k 是一个正整数, $P=\{0,1\}^k$,假设 $f: Y\rightarrow Z$ 是一单向 Hash 函数, $A=Y^k$,随机选择 $y_{ij}\in Y$,这里 $1\leq i\leq k, j=0,1$ 且 $z_{ij}=f(y_{ij}), 1\leq i\leq k, j=0,1$。

私有密钥: $y_{ij}, 1\leq i\leq k, j=0,1$

公开密钥: $z_{ij}, 1\leq i\leq k, j=0,1$

签名算法: $\mathrm{Sig}_{k_2}(x_1,\cdots,x_k)=(y_{1x_1},\cdots,y_{kx_k})$

验证算法: $\mathrm{Ver}_{k_1}(x_1,\cdots,x_k,a_1,\cdots,a_k)=\mathrm{TRUE}\quad f(a_i)=z_{ix_i}, 1\leq i\leq k$

图 10.4　Lamport 签名方案

10.1.5　不可否认签名方案

不可否认签名方案是 Chaum 和 Van Antwerpen 于 1989 年提出的,它有几种新的特点,其中一个很根本的特点是:如果没有签名者的合作,一个签名就得不到验证。因此,当签名被验证后,签名者是不可抵赖和否认他的签名的,故被称作不可否认签名方案。这样就防止了未经签名者的同意就随意复制他的签名文件并进行电子分发的可能性,验证将通过询问和应答协议来完成。

一个不可否认签名方案由三个部分组成:签名算法、验证协议和否认协议。

图 10.5 描述了 Chaum - van Antwerpen 不可否认签名方案的一个签名算法和验证协议,在图的下面描述了该方案的否认协议。这里假定通信双方是 A 和 B, B 是签名者, A 是验证者。

否认协议如下:

(1) A 随机选取 $e_1, e_2\in Z_p^*$。

(2) A 计算 $c=y^{e_1}\beta^{e_2}\bmod p$ 且把它传给 B。

(3) B 计算 $d=c^{a^{-1}\bmod q}\bmod p$,并将其传给 A。

(4) A 验证 $d \neq x^{e_1}\alpha^{e_2} \bmod p$。

(5) A 随机选取 f_1, $f_2 \in Z_p^*$。

(6) A 计算 $c' = y^{f_1}\beta^{f_2} \bmod p$ 且把它传给 B。

(7) B 计算 $d' = c^{a^{-1} \bmod q} \bmod p$, 并将其传给 A。

(8) A 验证 $d' \neq x^{f_1}\alpha^{f_2} \bmod p$。

(9) A 推出 y 是伪造的当且仅当

$$(d\alpha^{-e_2})^{f_1} = (d'\alpha^{-f_2})^{e_1} \bmod p。$$

系统参数: 设 $p = 2q + 1$ 是一个素数, 这里的 q 是素数且 Z_p 中的离散对数问题是难解决的, $\alpha \in Z_p^*$ 是 Z_p 域中的 q 次单位根, $1 \leq a \leq q - 1$, 设 G 表示阶为 q 的 Z_p^* 的乘法子群, $M = A = G$, 且定义

$$K = \{(p, \alpha, \beta, a): \beta = \alpha^a \bmod p\}$$

私有密钥: a

公开密钥: p, α, β

签名算法: 设待签消息为 $x \in G$, $y = \mathrm{Sig}_{k_2}(x) = x^a \bmod p$, 这里, $y \in G$。

验证协议: 1. A 随机选取 e_1, $e_2 \in Z_p^*$。

2. A 计算 $c = y^{e_1}\beta^{e_2} \bmod p$ 且把它传给 B。

3. B 计算 $d = c^{a^{-1} \bmod q} \bmod p$, 并将其传给 A。

4. A 接受 y, 并将它作为一有效签名当且仅当

$d = x^{e_1}\alpha^{e_2} \bmod p$。

图 10.5 Chaum - van Anterpen 不可否认签名方案

不可否认签名方案的安全性分析:

定理 10.1 当 $y \neq x^a \bmod p$ 时, 则 A 接受 y 作为 x 的真正签名的概率为 $1/q$。

这说明 B 除了一个非常小的概率之外, 不能使 A 接受一个错误的签名, 这个结果不依赖于任何计算上的假设, 它是无条件安全的。

下面证明以下的事实:

(1) B 能使 A 相信一个无效的签名是伪造的;

(2) 除了一个非常小的概率外, B 不能使 A 相信一个有效的签名是伪造的。

定理 10.2 若 $y \neq x^a \bmod p$, 且 A 和 B 都遵守否认协议, 则 $(d\alpha^{-e_2})^{f_1} = (d'\alpha^{-f_2})^{e_1} \bmod p$。

下面的定理给出了 B 可能否认一个有效签名的概率。

定理 10.3 若 $y = x^a \bmod p$ 且 A 遵守否认协议, 又 $d \neq x^{e_1}\alpha^{e_2} \bmod p$, $d' \neq x^{f_1}\alpha^{f_2} \bmod p$, 则 $(d\alpha^{-e_2})^{f_1} \neq (d'\alpha^{-f_2})^{e_1} \bmod p$ 成立的概率为 $1 - 1/q$。

证明 首先假定下面的等式成立:

$y = x^a \bmod p$, $d \neq x^{e_1}\alpha^{e_2} \bmod p$, $d' \neq x^{f_1}\alpha^{f_2} \bmod p$, $(d\alpha^{-e_2})^{f_1} = (d'\alpha^{-f_2})^{e_1} \bmod p$。

我们将推导出一个矛盾。

由最后一式得　$d' = d_0^{f_1}\alpha^{f_2} \bmod p$, 其中 $d_0 = d^{\frac{1}{e_1}}\alpha^{-\frac{e_2}{e_1}} \bmod p$。

这是一个仅依赖于否认协议(1)～(4)的值。

应用定理 10.1, 得到 y 以概率 $1 - 1/q$ 是 d_0 的一个有效签名, 但假设 y 是 x 的有效签名, 即以高概率成立 $x^a = d_0^a \bmod p$, 即 $x = d_0$。

然而 $d \neq x^{e_1}\alpha^{e_2} \bmod p$, 意味着 $x \neq d^{\frac{1}{e_1}}\alpha^{-\frac{e_2}{e_1}} \bmod p$。

又 $d_0 = d^{\frac{1}{e_1}}\alpha^{-\frac{e_2}{e_1}} \bmod p$, 所以 $x \neq d_0$, 与 $x = d_0$ 产生矛盾。因此, B 以此愚弄 A 的概率为 $1/q$。

10.1.6　故障停止式签名方案

故障停止式签名方案是针对一个非常强的对手能伪造一个签名的可能性而设计的, 它提供了一个增强的安全性, 即使对手能够伪造关于某一消息的签名, 随后也能以很高的概率证明对手的签名是伪造的。

我们将描述由 van Heyst 和 Pedersen 于 1992 年构造的故障停止式签名方案, 它是一个一次方案, 也就是给定一个密钥, 仅有一个消息能签名, 这个系统由签名算法和验证算法以及一个伪造的证明算法组成。它的具体描述如下:

系统参数　设 $p = 2q + 1$ 是一个素数, 这里的 q 是素数且 Z_p 中的离散对数问题是难解决的, $\alpha \in Z_p^*$ 满足 $\alpha^q = 1 \bmod p$ 且 $\alpha \neq 1$, $1 \leq a_0 \leq q - 1$, $\beta = \alpha^{a_0} \bmod p$, p, q, α, β 和 a_0 是通过一个可信的中心机构选定, p, q, α 和 β 对外公开且被视为固定不变的, a_0 由权威机构保密, 参与通信的用户是不知道的。

设 $M = Z_q$ 和 $A = Z_q \times Z_q$, 定义 $\gamma_1 = \alpha^{a_1}\beta^{a_2} \bmod p$, $\gamma_2 = \alpha^{b_1}\beta^{b_2} \bmod p$。

密钥具有如下形式: $k = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$, 这里的 γ_1, γ_2 是公钥, $a_1, a_2, b_1, b_2 \in Z_q$ 是私钥。

签名算法　对于 $k = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ 和待签消息 $x \in Z_q$, 定义 $\mathrm{Sig}_{k_2}(x) = (y_1, y_2)$, $y_1 = a_1 + xb_1 \bmod q$, $y_2 = a_2 + xb_2 \bmod q$, 消息对 (y_1, y_2) 即为生成的签名。

验证算法　对 $y = (y_1, y_2) \in Z_q \times Z_q$, 有

$$\mathrm{Ver}_{k_1}(x, y) = \mathrm{TRUE} \quad \gamma_1\gamma_2^x = \alpha^{y_1}\beta^{y_2} \bmod p。$$

故障停止式签名方案的安全性分析:

为了分析的方便, 在这里引入一个概念: 两个密钥被认为是等价的, 如果两个密钥 $(\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ 和 $(\gamma_1', \gamma_2', a_1', a_2', b_1', b_2')$ 满足条件 $\gamma_1 = \gamma_1'$, $\gamma_2 = \gamma_2'$。

引理 10.1　如果 k 和 k' 是等价的密钥, 且 $\mathrm{Ver}_{k_1}(x, y) = \mathrm{TRUE}$, 则 $\mathrm{Ver}_{k_1'}(x, y) =$

TRUE。

证明 设 $k=(\gamma_1,\gamma_2,a_1,a_2,b_1,b_2)$和 $k'=(\gamma_1',\gamma_2',a_1',a_2',b_1',b_2')$且 $\gamma_1=\gamma_1'$, $\gamma_2=\gamma_2'$。

假设消息 x 是用密钥 k_2 来签名的,产生的签名为 $y=(y_1,y_2)$。

满足 $y_1=a_1+xb_1 \bmod q$, $y_2=a_2+xb_2 \bmod q$。

下面证明如果用密钥 k_1'来验证的话,它也是有效的签名。

$$\alpha^{y_1}\beta^{y_2}=\alpha^{a_1'+xb_1'}\beta^{a_2'+xb_2'}=\alpha^{a_1'}\beta^{a_2'}(\alpha^{b_1'}\beta^{b_2'})^x=\gamma_1(\gamma_2)^x=\gamma_1'(\gamma_2')^x \bmod p$$

后面等价的原因是 $\gamma_1=\alpha^{a_1}\beta^{a_2} \bmod p$, $\gamma_2=\alpha^{b_1}\beta^{b_2} \bmod p$ 和 $\gamma_1'=\alpha^{a_1'}\beta^{a_2'}$, $\gamma_2'=\alpha^{b_1'}\beta^{b_2'}$且 $\gamma_1=\gamma_1'$, $\gamma_2=\gamma_2'$。

故 y 也可由 k_1'来验证。

引理 10.2 如果 $k=(k_1,k_2)$是一个密钥且 $y=\mathrm{Sig}_{k_2}(x)$,则刚好有 q 个密钥 $k'=(k_1',k_2')$与 k 等价,且满足 $y=\mathrm{Sig}_{k_2'}(x)$。

证明 假设 γ_1 和 γ_2 是 k 的公开密钥,我们想确定满足下列同余式的 4 元组 (a_1,a_2,b_1,b_2)的数目:

$$\gamma_1=\alpha^{a_1}\beta^{a_2} \bmod p,\quad \gamma_2=\alpha^{b_1}\beta^{b_2} \bmod p,$$

$$y_1=a_1+xb_1 \bmod q,\ y_2=a_2+xb_2 \bmod q。$$

因为 α生成了群 $G=Z_q$,这里存在惟一指数 $c_1,c_2,a_0\in Z_q$ 满足

$\gamma_1=\alpha^{c_1} \bmod p$ $\gamma_2=\alpha^{c_2} \bmod p$ 和 $\beta=\alpha^{a_0} \bmod p$。

因此,满足下列同余方程组是充分且必要的:

$$c_1=a_1+a_0a_2 \bmod q,\quad c_2=b_1+a_0b_2 \bmod q,$$

$$y_1=a_1+xb_1 \bmod q,\quad y_2=a_2+xb_2 \bmod q。$$

结果这个方程组能写成如下的 Z_q 中的矩阵方程:

$$\begin{matrix}1&a_0&0&0\\0&0&1&a_0\\1&0&x&0\\0&1&0&x\end{matrix}\begin{matrix}a_1\\a_2\\b_1\\b_2\end{matrix}=\begin{matrix}c_1\\c_2\\y_1\\y_2\end{matrix}。$$

现在能看到这个方程组的系数矩阵的秩为“3”。显然,秩至少为 3,因为 1,2 和 4 行在 Z_q 上是线性无关的,且秩至多为 3,因为 $r_1+xr_2-r_3-a_0r_4=(0,0,0,0)$。

这里的 r_i 表示矩阵的第 i 行。

现在这个方程组通过使用密钥 k,至少可得到一个解,因为系数矩阵的秩为 3,它就证明了解空间的维数为 1,且刚好有 q 个解,结论成立。

引理 10.3 如果 $k=(k_1,k_2)$是一个密钥,且 $y=\mathrm{Sig}_{k_2}(x)$ 和 $\mathrm{Ver}_{k_1}(x',y')=$ TRUE,这里的 $x\neq x'$, $y'=\mathrm{Sig}_{k_2}(x')$,则至多存在一个与 k 等价的密钥 k'满足

$y = \mathrm{Sig}_{k'_2}(x)$，且 $y' = \mathrm{Sig}_{k'_2}(x')$。

可仿照引理 10.2 的证明去证，为节约篇幅，这里省去证明。

定理 10.4　给定 $y = \mathrm{Sig}_{k_2}(x)$，且 $x \neq x'$，敌手只能以 $1/q$ 的概率计算出 $\mathrm{Sig}_{k_2}(x')$。

现在来看一下故障停止式概念的内涵：如果给了发方一个有效的伪造签名，则发方能以 $1 - 1/q$ 的概率产生一个“伪造证明”。该伪造证明是一个值 $a_0 = \log_\alpha \beta$，仅有中心机构才知道该值。

假设发方具有一个满足 $\mathrm{Ver}(x', y'') = \mathrm{TRUE}$，$y'' \neq \mathrm{Sig}(x')$的消息对$(x', y'')$，即

$$\gamma_1 \gamma_2^{x'} = \alpha^{y''_1} \beta^{y''_2} \bmod p。$$

这里 $y'' = (y_1'', y_2'')$，发方能计算他自己关于 x'的签名，即 $y' = (y_1', y_2')$ 且满足 $\gamma_1 \gamma_2^{x'} = \alpha^{y'_1} \beta^{y'_2} \bmod p$，得 $\alpha^{y''_1} \beta^{y''_2} = \alpha^{y'_1} \beta^{y'_2} \bmod p$

$$y_1'' + a_0 y_2'' = y_1' + a_0 y_2' \bmod q$$

$$y_1'' - y_1' = a_0 (y_1' - y_2'') \bmod q。$$

因为 $y_2' \neq y_2''$，所以$(y_2' - y_2'')^{-1} \bmod q$ 存在，且

$$a_0 = \log_\alpha \beta = (y_1'' - y_1')(y_2' - y_2'')^{-1} \bmod q。$$

整个体系都是建立在发方计算离散对数 $\log_\alpha \beta$ 是困难的基础之上的。

10.1.7　Schnorr 数字签名方案

Schnorr 数字签名方案的描述如下：

系统参数

p 是一大素数，$p \geq 2^{512}$ 且它满足 Z_p 中的离散对数问题是难解决的。

q 是大素数，$q \geq 2^{160}$ 且 $q \mid p - 1$。

$g \in Z_p^*$，满足 $g^q = 1 \bmod p$ 且 $g \neq 1$。

定义　$K = \{(p, q, g, y, x) : y = g^x \bmod p\}$，其中 $x \in Z_p^*$，

公开密钥：p, q, g, y，　　私有密钥：x。

签名算法

对于待签消息 $m \in Z_p^*$，选择随机数 $k(1 < k < q)$，计算 $r = g^k \bmod p$，$e = h(r, m)$，$h(\cdot)$为一个 Hash 函数，从签名方程 $s = k - xe \bmod q$ 中解出 s，消息对(e, s)即为生成的签名。

验证算法

收方在收到消息 m 和数字签名(e, s)后 计算 $r' = g^s y^e \bmod p$，则

$$\mathrm{Ver}(m, (e, s)) = \mathrm{TRUE} \quad h(r', m) = e。$$

这个签名方案的正确性证明如下：$r' = g^s y^e = g^s g^{ex} = g^{(s+ex)} = g^k = r \bmod p$，故

$$h(r', m) = h(r, m) = e。$$

Schnorr 数字签名方案的安全性分析：

(1) ELGamal 系统中，g 为 Z_p 中的本原元，而于 Schnorr 系统中则不是。从安全的角

度来看,ELGamal 安全性较高,这是因为本原元的阶为 $p-1$,而 Schnorr 系统中,g 的阶为 $q(>2^{160})$,在此阶下,基于离散对数问题的体制是否安全有待进一步研究。

(2) Schnorr 系统的签名文较短,它的 e 的长度由 Hash 函数 h 决定。s 的长度小于 $|q|$。若 h 的输出长度为 128 位,$|q|$ 为 160 位,则其签名长度为 288 位,远比 ELGamal 系统的 1024 位小。

(3) Schnorr 首先提出 $r=g^k \bmod p$ 可以事先计算,由于 k 是与 m 无关的随机数,故 Schnorr 系统在签名中只需一次乘法及减法(模运算),比 ELGamal 系统快很多。因此,Schnorr 数字签名方案特别适合于智能卡的应用。

10.2 群签名

1991 年,Chaum 和 Van Heyst 首次提出了一种新的签名方案,叫做群签名方案(Group Signature)。他们是基于下面的一个问题提出的:

一个公司有几台计算机,每台都连在局域网上。公司的每个部门有其自己的打印机(也连在局域网上),并且只有本部门的人员才能被允许使用其部门的打印机,因此,打印前,必须是打印机确认用户在哪个部门工作,同时,公司想保密,不可以暴露用户的姓名。然而,如果有人在当天结束时发现打印机用得太频繁,主管者必须能找出谁滥用了那台打印机,并给他一个账单。

群签名方案允许组中的合法用户以用户组的名义签名,同时在出现争论的情况时,权威或组的所有成员联合起来可辨认并揭示出签名者。一般说来,群签名方案由组、组成员(签名者)、签名接受者(签名验证者)和权威(Authority)或 GC(Group Center)组成,具有如下特点:

(1) 只有组中的合法用户才能对消息签名,并产生群签名;

(2) 签名的接收者能验证群签名的有效性;

(3) 签名的接收者不能辨认是谁的签名;

(4) 一旦发生争论,群签名的权威或组中所有成员的联合可以辨别出签名者。

在 Chaum 和 Van Heyst 的开创性论文里提出了 4 种签名体制,但它们都具有如下的缺点:(1)当群体改变时,每个成员需要重新分配密钥;(2)权威不能辨别出签名者。为了解决这个问题,本文介绍一种称作 K-P-W 可变群签名方案(Convertible Group Signature)。

K-P-W 可变群签名方案如图 10.6 所示。

系统参数：选择 $n = pq = (2fp' + 1)(2fq' + 1)$，这里的 p，q，f，p'和 q'为相异的大素数，g 的阶为 f，γ 和 d 为整数，且 $\gamma d = 1 \bmod \phi(n)$，$\gcd(\gamma, \phi(n)) = 1$，$h$ 为安全的 Hash 函数，ID_G 为 GC 的身份消息。

签名组的公钥：$(n, \gamma, g, f, h, ID_G)$

签名组的私钥：(d, p', q')。

设 ID_A 为组成员 A 的身份消息，A 随机选取 $s_A \in (0, f)$，并将消息 $(ID_A, g^{s_A} \bmod n)$ 发送给 GC。GC 计算 $x_A = (ID_{G'})^{-d} \bmod n$，并将 x_A 秘密地传送给成员 A，则 A 的私有密钥：(x_A, s_A)。

签名算法：对于待签消息 m：组中成员 A 随机选择整数 $(r_1, r_2) \in [0, f)$，计算

$$V = g^{r_1} r_2^{\gamma} \bmod n,\ e = h(V, m),$$

则群签名为 (e, z_1, z_2)。

其中 $z_1 = r_1 + s_A e \pmod f$，$z_2 = r_2 x_A^e \pmod n$

签名验证算法：$e = h(\overline{V}, m)$，这里的 $\overline{V} = (ID_G)^e g_{z_1} z_2^{\gamma} \pmod n$。

身份验证算法：$g^{z_1} = (V r_2^{-\gamma})(g^{s_A})^e \pmod n$，其中 $r_2 = z_2 x_A^{-e} \pmod n$。

图 10.6　K - P - W 可变群签名方案

K - P - W 可变群签名方案的安全性分析：

(1) 当 p'和 q'具有相同的比特位时，攻击者可以采用对参数 n 进行因子分解的方法。分解 $n = pq = (2fp' + 1)(2fq' + 1)$ 只需要 $2^{(|p| - |f|)^2}$ 次整数乘法，这里的 $|x|$ 为 x 的比特位数。

(2) 在组中成员诚实的情况下，虽说权威能辨别出签名者，可是当组中的成员伪造或共谋伪造时，仍能生成有效的签名。设组中成员 A 随机选取整数 a 和 b，计算 $s_A = ab \bmod f$ 和 $s_A' = s_A + b \bmod f$，将 s_A，s_A'分别作为 A 的两个私钥，公钥为 $y_A = g^{s_A} \bmod n$，$y_A' = g^{s_A'} \bmod n$，从 GC 处秘密地收到相应的另外两个私钥 x_A 和 x_A'，则有 $g^{bd} = x_A (x_A')^{-1} \bmod n$，$ID_G^{-d} = x_A (g^{bd})^a \bmod n$，于是可以得到 $g^d = (g^{bd})^{b^{-1}} \bmod n$。对于任意的 s_A，由于 A 知道 ID_G^{-d}和 g^d，故可算出私钥 (x_A, s_A)。然后利用 K - P - W 签名方案，对任意的消息，都可以产生有效的群签名，而权威无法辨认签名用户。如果组中两成员共谋，利用上述方法同样可产生有效的群签名，而权威无法辨认签名用户。

10.3　多重数字签名方案

多重数字签名方案(Digital Multisignature)是一种能够实现多个用户对同一消息签名的数字签名方案。根据签名过程的不同，多重数字签名方案可分成广播多重数字签名方

案(Broadcasting Multisignature)和有序多重数字签名方案(Sequential Multisignature),这两类方案都包含消息的发送者(Issuer)、消息的签名者(Signers)和签名的验证者(Verifier)。只是在广播多重数字签名方案中还包含签名的收集者(Collector)。

对于广播多重数字签名方案,首先由消息的发送者将待签名的消息同时发给每一个签名者进行签名,然后签名者将签名的消息发给签名的收集者,收集者将签名的消息整理后发给签名的验证者,最后由签名的验证者验证多重签名的有效性。完整的广播多重数字签名方案过程的描述如图 10 .7 所示。

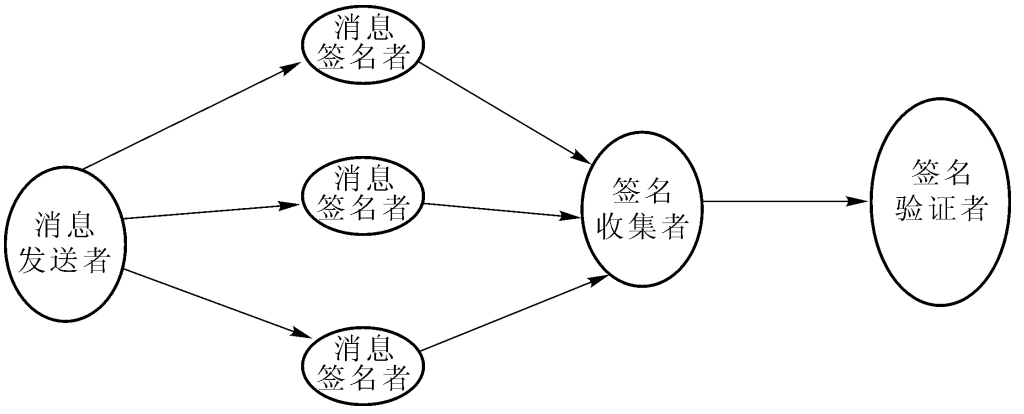


图 10 .7　广播多重数字签名方案

而在有序多重数字签名方案中,消息的发送者规定了消息的签名顺序,并按此顺序将待签消息依次发给每一个签名者,除了第一个签名者外,任意一个签名者在签名之前都要验证上一签名者的签名的有效性。如果签名无效,则拒绝对消息签名,终止整个签名;若签名有效,则继续签名,并将签名消息发给下一个签名者。最后,当签名的验证者收到签名消息后,验证多重签名的有效性。完整的有序多重数字签名方案过程的描述如图 10 .8 所示。

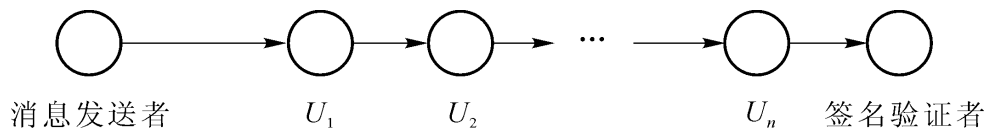


图 10 .8　有序多重数字签名方案

下面分别介绍这两种类型的一种代表性签名方案。

1 . ELGamal 有序多重数字签名方案

(1) 系统初始化

选取大素数 p,且它满足 Z_p 中的离散对数问题是难解决的,g 是 GF(p)的本原元,函数 h:GF$(p)\rightarrow$GF(p)是一单向 Hash 函数。每一位签名者 $U_i(i=1,2,\cdots,n)$的私钥 x_i 是从$[1, p-1]$中任意选取的,公钥为 $y_i = g^{x_i} \bmod p$,系统参数 p, g, $y_i(i=1,2,\cdots,n)$和 h 是公开的,参数 x_i 是由相应的签名者秘密保管的。

(2) 签名算法

签名者 $U_i(i=2,\cdots,n)$收到上一签名者发送的待签消息$(m,(s_{i-1},r_{i-1}))$后$(s_0=0)$，首先对签名消息进行验证，然后做如下的工作：首先随机选取 $k_i\in[1,p-1]$，计算 $r_i=g^{k_i} \bmod p$，$s_i=s_{i-1}+m'x_i-r_ik_i \bmod \phi(p)$，这里的 $m'=h(m)$，最后将签名消息$(m,(s_i,r_i))$发给下一个签名者 U_{i+1}，并将 r_i 发给 U_i 以后的签名者和签名验证者。

(3) 验证算法

验证包括两方面的要求：签名者 $U_i(i=2,\cdots,n)$要对 $U_1,U_2,\cdots,U_{i-1}$的签名的有效性进行验证；验证者 U_V 要对所有签名者 $U_1,U_2,\cdots,U_n$ 的签名进行验证。

对于 $U_i(i=2,\cdots,n)$，通过验证等式 $g^{s_{i-1}}\prod_{j=1}^{i-1} r_j^{r_j}=\prod_{j=1}^{i-1} y_j^{m'_j} \bmod p$ 是否成立，来判断 $U_1,U_2,\cdots,U_{i-1}$的签名是否有效。若等式成立则有效，反之无效。

对于验证者 U_V，通过验证等式 $g^{s_n}\prod_{j=1}^{n} r_j^{r_j}=\prod_{j=1}^{n} y_j^{m'_j} \bmod p$ 是否成立来判断 $U_1,U_2,\cdots,U_n$的签名是否有效。若等式成立则有效，反之无效。

下面证明该方案的正确性。由递推关系式 $s_i=s_{i-1}+m'x_i-r_ik_i \bmod \phi(p)$得 $s_i=s_0+\sum_{j=1}^{i}(m'x_j-r_jk_j) \bmod \phi(p)$，变形后得

$$s_0+\sum_{j=1}^{i} m'x_j=\sum_{j=1}^{i} m'x_j=\sum_{j=1}^{i} r_jk_j+s_i \bmod \phi(p)。$$

因此对任意的 i，下面的等式恒成立：$g^{\sum_{j=1}^{i} r_jk_j+s_i \bmod \phi(p)}=g^{\sum_{j=1}^{i} m'x_j \bmod \phi(p)} \bmod p$，即

$$g^{s_{i-1}}\prod_{j=1}^{i-1} r_j^{r_j}=\prod_{j=1}^{i-1} y_j^{m'_j} \bmod p。$$

故 ELGamal 有序多重数字签名方案的正确性得证。

2．Harn 广播多重数字签名方案

1994 年，Harn 首次设计了一种广播多重数字签名方案，我们称之为 Harn 广播多重数字签名方案。它包括如下几个过程：系统初始化、单用户签名的产生、单用户签名的验证、多重签名的产生、多重签名的验证。该方案的参与者包括消息发送者 U_I，签名者 $U_i(i=1,2,\cdots,n)$，签名收集者 U_C 和签名验证者 U_V。签名方案的具体过程如下：

(1) 系统初始化

选取大素数 p，且它满足 Z_p 中的离散对数问题是难解决的，g 是 GF(p)的本原元，函数 h：GF$(p)\rightarrow$GF(p)是一单向 Hash 函数。每一位签名者 $U_i(i=1,2,\cdots,n)$的私钥 x_i 是从$[1,p-1]$中任意选取的，公钥为 $y_i=g^{x_i} \bmod p$，系统参数 p，g，$y_i(i=1,2,\cdots,n)$和 h 是公开的，参数 x_i 是由相应的签名者秘密保管的。

(2) 单用户签名的产生

消息发送者 U_I 将待签消息 m 发送给每一签名者 $U_i(i=1,2,\cdots,n)$。当签名者 $U_i(i$

$=1,2,\cdots,n)$收到 U_I 发送的待签消息 m 后,做如下的工作:首先随机选取 $k_i\in[1,p-1]$,计算 $r_i=g^{k_i}\bmod p$,并将 r_i 发给其他的签名者 $U_j(j\neq i)$;然后计算 $R=\prod_{i=1}^{n}r_i^{r_i}\bmod p$, $s_i=(R+m')x_i-r_ik_i\bmod\phi(p)$,这里的 $m'=h(m)$;最后将签名消息$(m,(s_i,r_i))$发给签名的收集者 U_C。

(3) 单用户签名的验证

U_C 收到签名消息$(m,(s_i,r_i))$后,做如下的验证工作:计算 $R=\prod_{i=1}^{n}r_i^{r_i}\bmod p$,通过验证方程 $r_i^{r_i}g^{s_i}=y_i^{m'+R}\bmod p$ 是否成立来判断签名者 $U_i(i=1,2,\cdots,n)$的签名的有效性,若方程不成立,则签名无效,终止签名;若方程成立,则签名有效, U_C 继续产生多重签名。

(4) 多重签名的产生

U_C 计算 $S=s_1+s_2+\cdots+s_n\bmod(p-1)$, (R,S)即为消息 m 的多重签名, U_C 将 (R,S)发送给签名的验证者 U_V。

(5) 多重签名的验证

当 U_V 收到消息$(m,(R,S))$后 $\mathrm{Ver}(m,R,S)=\mathrm{TRUE}$ $Rg^S=\prod_{i=1}^{n}y_i^{m'+R}$ 该方案的正确性留作作业请同学们自己练习。

10.4 代理数字签名体制

定义 10.2 设 A,B 是某个数字签名体制(M,A,K,S,V)的两个用户,他们的私钥和公钥对分别是$(x_A,y_A),(x_B,y_B)$。如果满足下列条件:

(1) A 利用它的秘密密钥 x_A 计算出一个数 σ,并将 σ 秘密交给 B;

(2) 任何人(包括 B)在试图求出 x_A 时,σ 不会对他有任何帮助;

(3) B 可以用 σ 和 x_B 生成一新的签名密钥 $\sigma_{A\to B}$;

(4) 存在一个公开的验证算法 $\mathrm{Ver}_{A\to B}$,使得对任何 $s\in S$ 和 $m\in M$ 都有

$$\mathrm{Ver}_{A\to B}(y_A,s,m)=\mathrm{TRUE}\quad s=\mathrm{Sig}(\sigma_{A\to B},m);$$

(5) 任何人在试图求出 x_A, x_B, σ 或 $\sigma_{A\to B}$时,任何数字签名 $\mathrm{Sig}(\sigma_{A\to B},m)$都不会对他产生帮助。

那么就称用户 A 将他的数字签名权力委托给用户 B,并且称 A 为 B 的原始签名人,B 为 A 的代理签名人,称 σ 为委托密钥,称 $\sigma_{A\to B}$为代理签名密钥,称以 $\sigma_{A\to B}$作为签名密钥对消息 $m\in M$ 生成的数字签名 $\mathrm{Sig}(\sigma_{A\to B},m)$为 A 的代理签名(Proxy Signature)。

能够生成代理签名的数字签名体制被称为代理数字签名体制 。

代理签名体制具有下面的特定的安全特性:

(1) 可区别性。任何人都可区别代理签名和正常的签名。

(2) 代理签名的不可伪造性。只有原始签名者和指定的代理签名者能够产生有效的代理签名。

(3) 密钥的依赖性。代理签名密钥依赖于原始签名人的秘密密钥。

(4) 可验证性。从代理签名中,验证者能够相信原始的签名者认同了这份签名消息。

(5) 可识别性。原始签名者能够从代理签名中识别代理签名者的身份。

(6) 不可抵赖性。代理签名者不能否认他创立的且被认可的代理签名。

(7) 可注销性。如果原始签名者希望代理签名人只能在一定时间内拥有生成代理签名的能力,那么必须能让代理签名人的代理签名密钥在指定时间内失去作用。

在某些情况中,需要更强的可识别性形式,即任何人都能从代理签名中确定代理签名者的身份。

下面介绍一种基于离散对数问题的代理签名体制。

设签名体制(M, A, K, S, V)是一基于离散对数问题的签名体制。它的用户的密钥对(x, y),满足 $y = g^x \bmod p$,其中用户 A 和 B 的私钥和公钥对分别是(x_A, y_A),(x_B, y_B),这里的 $y_A = g^{x_A} \bmod p$, $y_B = g^{x_B} \bmod p$。我们欲生成 A→B 的代理签名,具体过程如图 10.9 所示。

系统参数:(M, A, K, S, V)是一基于离散对数问题的签名体制,其参数 p, q, g 满足 p 是一大素数,它满足 Z_p 中的离散对数问题是难解决的。

q 是大素数,且 $q \mid p-1$, $g \in Z_p^*$ 是 Z_p 域中的 q 次单位根。

用户 A 和 B 的私钥和公钥对分别是(x_A, y_A),(x_B, y_B),这里的 $y_A = g^{x_A} \bmod p$, $y_B = g^{x_B} \bmod p$。

委托过程:A 随机选择一整数 $k \in Z_p^*$,计算 $r = g^k \bmod p$, $\sigma = x_A + kr \bmod q$,将$(\sigma, r)$秘密传给 B,B 收到消息后验证等式 $g^\sigma = y_A r^r \bmod p$ 是否成立。

代理签名的生成算法:对于待签消息 m, B 生成普通的数字签名 $s = \mathrm{Sig}(\sigma, m)$,将$(s, r)$作为他代表 A 对消息 m 的数字签名,即代理签名。

代理签名的验证算法:当代理签名的接收者收到了消息 m 和代理签名(s, r)后,计算 $v = y_A r^r \bmod p$;验证

$\mathrm{Ver}(y_A, (s, r), m) = \mathrm{TRUE}$　$\mathrm{Ver}(v, s, m) = \mathrm{TRUE}$。

图 10.9　代理签名体制

该签名体制的正确性由下面的算式证明:$v = y_A r^r = g^{x_A} g^{kr} = g^{x_A + kr} = g^\sigma \bmod p$,即 v 可以作为 σ 的验证公钥。

该代理签名体制的安全性分析:

(1) 基本的不可伪造性。在这个代理签名体制中,B 难以根据所得到的(σ, r)计算出x_A,从而不能伪造 A 的普通数字签名。由此说明,任何其他的攻击者都难以伪造 A 的普通数字签名。

(2) 代理签名的不可伪造性。由于 A 和 B 都知道(σ, r),所以,A 和 B 都能生成代理签名。但是除了 A 和 B 以外,其他任何人都难以伪造一个有效的代理签名。

(3) 代理签名的可区分性。代理签名(s, r)由两部分组成,一部分是普通数字签名s,另一部分是某个数 r。由于代理签名比普通签名多出一部分,所以,容易将代理签名与普通的数字签名区分开。假如除了 B 以外,还有另外一个代理签名人 C,A 在委托过程中发送给 C 的消息是(σ_1, r_1)。这里的 $r_1 = g^{k_1} \bmod p$, $k_1 \neq k$,于是 C 生成的代理签名体制的形式一定为(s_1, r_1),可以看出,由于 $k_1 \neq k$,所以 $r_1 \neq r$,从而将 B 和 C 生成的代理签名区分开。

(4) 不可抵赖性。由于任何人都不能伪造 A 的普通数字签名,所以 A 不能否认他的一个有效的普通签名。由于除了 A 和 B 外,任何人都不能伪造 B 的代理签名,所以 A 和 B 都不能否认一个有效的代理签名(s, r)是由他们中的某个人生成的。但是,A 和 B 可以互相对有效的代理签名进行抵赖,声称它是由对方而不是由自己生成的。

(5) 身份证实性。在这个代理签名体制中,如果 A 向 B 发送了(σ, r)的时候,将 r 和 B 的身份保存在一起,那么当 A 看到一个有效的代理签名(s, r)的时候,就可以通过 r 确认 B 的身份。

(6) 密钥依赖性。在这个代理签名体制中,代理签名密钥 σ 是 A 的秘密密钥 x_A 的函数值,即它依赖于 x_A。

(7) 可注销性。A 如果想注销 B 拥有的代理签名密钥 σ,那么就可以向大家“广播”一条消息(这个消息应该由 A 签名),宣布 r 不再有效,从而 B 生成的所有代理签名随之失效。

10.5 基于纠错码的数字签名体制

若仔细观察前面几节所讲到的签名体制的实现方法,不难发现它们都是基于 RSA 和离散对数问题的,是否有基于其他数学问题的签名体制了?答案是肯定的。自从 1978 年 McEliece 首次构造出了一个基于纠错码的公钥密码体制,实现了基于纠错码的加密方案以来,人们一直在思考是否能够运用纠错码实现别的安全要求,比如签名与认证。终于于 1990 年我国的著名学者王新梅教授提出了一类基于纠错码的数字签名方案——Xinmei 方案,并在此后提出了一系列的兼有加密和签名功能的方案。下面让我们享受这种新型的签名方案带给我们的快乐吧!

Xinmei 数字签名方案如图 10.10 所示。

系统参数：用户选择一个可纠 t 个错误的 GF(2)上的 $[n, k, d]$ 既约 Goppa 码 C，这里的 $d \geq 2t+1$，或者有大码类的其他码。码 C 的生成矩阵和校验矩阵分别为 $k \times n$ 阶的 $\boldsymbol{G}$ 和 $(n-k) \times n$ 阶的 $\boldsymbol{H}$。

用户的公开密钥：

$$J = \boldsymbol{P}^{-1}\boldsymbol{G}^{*}\boldsymbol{S}^{-1},\ W = \boldsymbol{G}^{*}\boldsymbol{S}^{-1},\ \boldsymbol{T} = \boldsymbol{P}^{-1}\boldsymbol{H}^{\mathrm{T}},\ \boldsymbol{H},\ t。$$

用户的私有密钥：$\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P}$。

$\boldsymbol{G}$ 和 $\boldsymbol{G}^{*}$ 满足关系式 $\boldsymbol{G}\boldsymbol{G}^{*} = \boldsymbol{I}_k$，这里的 $\boldsymbol{I}_k$ 是 $k \times k$ 阶单位矩阵，$\boldsymbol{P}$ 和 $\boldsymbol{S}$ 分别是 GF(2)上的 $n \times n$ 阶和 $k \times k$ 阶满秩的随机矩阵，$\boldsymbol{P}^{-1}$ 和 $\boldsymbol{S}^{-1}$ 分别是它们的右逆阵。

签名算法：若待签的消息 $M \in F_2^k$，用户 A 随机选取 $E \in F_2^n$ 且 $w(E) \leq t$，则签名如下：

$$\mathrm{Sig}(M) = (E + M\boldsymbol{S}\boldsymbol{G})\boldsymbol{P} = C'。$$

验证算法：由于信道中可能存在干扰，因此用户 B 收到的消息为 $C'' = C' + E' = (E + M\boldsymbol{S}\boldsymbol{G})P + E'$，$(E' \in F_2^n)$，计算

$$C''\boldsymbol{T} = [(E + M\boldsymbol{S}\boldsymbol{G})\boldsymbol{P} + E']\boldsymbol{P}^{-1}\boldsymbol{H}^{\mathrm{T}} = E''\boldsymbol{H}^{\mathrm{T}}。$$

这里的 $E'' = E + E'\boldsymbol{P}^{-1}$，运用已知的译码算法，如 Berlekamp-Massy 算法译码。当译码器发现 $w(E'') > t$，则要求用户重发签名，当 $E' = 0$ 时，正确译码得到 C' 和 E。

$$\mathrm{Ver}(M, C'') = \mathrm{TRUE} \quad M = C'J + EW。$$

图 10.10　Xinmei 数字签名方案

由签名算法中的 E 的随机性知对于同一消息 M 我们可产生不同的签名，因此这是一类概率性的签名算法。该签名体制的正确性由下面的证明可知：

$$C'J + EW = [(E + M\boldsymbol{S}\boldsymbol{G})P]P^{-1}\boldsymbol{G}^{*}\boldsymbol{S}^{-1} + EW = E\boldsymbol{G}^{*}\boldsymbol{S}^{-1} + M + E\boldsymbol{G}^{*}\boldsymbol{S}^{-1} = M。$$

Xinmei 数字签名方案的安全性分析与改进：

自从 Xinmei 方案被提出以来，国内外不少学者对它进行了多方面的安全性分析，提出了一些它的不足及改进的方法，下面将介绍一些典型的关于这方面的结果。

(1) Harn 攻击

1992 年 Harn 等首先对 Xinmei 方案提出一种攻击方法，并给出了一种修正方案。他们是利用了 Xinmei 方案固有的线性性，他们的做法如下：

假设攻击者知道用户对消息 M_1 和 M_2 的签名：

$$C_1 = (E_1 + M_1\boldsymbol{S}\boldsymbol{G})\boldsymbol{P},$$

$$C_2 = (E_2 + M_2\boldsymbol{S}\boldsymbol{G})\boldsymbol{P}。$$

若 $w(E_1 + E_2) \leq t$，则攻击者可通过组合签名 $C_1 + C_2$ 来冒充这一用户对消息 $M_1 + M_2$ 的签名，这是因为

$$\begin{aligned} C_1 + C_2 &= (E_1 + M\boldsymbol{S}\boldsymbol{G})\boldsymbol{P} + (E_2 + M\boldsymbol{S}\boldsymbol{G})\boldsymbol{P} \\ &= [(E_1 + E_2) + (M_1 + M_2)\boldsymbol{S}\boldsymbol{G}]\boldsymbol{P}。 \end{aligned}$$

显然当 $w(E_1 + E_2) \leq t$ 时，$C_1 + C_2$ 就是用户对消息 $M_1 + M_2$ 的签名。

为了防止这种攻击方法，Harn 等建议运用非线性的 Hash 函数 $h(x)$ 先对消息 M 进行处理，然后再进行签名运算：

$$\mathrm{Sig}(M) = [E + h(M)\boldsymbol{SG}]\boldsymbol{P}。$$

由于 Hash 函数的非线性，故能防止这种线性攻击。

(2) Alabhadi - Wieker 攻击

Alabhadi 和 Wieker 于 1992 年提出了两种 Xinmei 方案攻击的方法，并提出了一种改进方案，下面介绍他们的具体做法：

假设攻击者能设法使被攻击的用户对同一消息 m 进行 $n+1$ 次签名：$C_1, C_2, \cdots, C_n, C_{n+1}$ 且满足其中的签名 $C_1, C_2, \cdots, C_n$ 是线性无关的，则 $C_1 + C_{n+1}, C_2 + C_{n+1}, \cdots, C_n + C_{n+1}$ 也是线性无关的，由

$$C_1 + C_{n+1} = (E_1 + M\boldsymbol{SG})\boldsymbol{P} + (E_{n+1} + M\boldsymbol{SG})\boldsymbol{P} = (E_1 + E_{n+1})\boldsymbol{P};$$

$$C_2 + C_{n+1} = (E_2 + M\boldsymbol{SG})\boldsymbol{P} + (E_{n+1} + M\boldsymbol{SG})\boldsymbol{P} = (E_2 + E_{n+1})\boldsymbol{P};$$

$$\cdots$$

$$C_n + C_{n+1} = (E_n + M\boldsymbol{SG})\boldsymbol{P} + (E_{n+1} + M\boldsymbol{SG})\boldsymbol{P} = (E_n + E_{n+1})\boldsymbol{P}。$$

我们可构造出如下形式的矩阵 $\boldsymbol{X} = \boldsymbol{YP}$，其中

$$\boldsymbol{X} = \begin{matrix} C_1 + C_{n+1} \\ C_2 + C_{n+1} \\ \cdots \\ C_n + C_{n+1} \end{matrix} \quad \text{和} \quad \boldsymbol{Y} = \begin{matrix} E_1 + E_{n+1} \\ E_2 + E_{n+1} \\ \cdots \\ E_n + E_{n+1} \end{matrix}$$

都是 $n \times n$ 阶矩阵。因为是 $\boldsymbol{X}$ 满秩矩阵，所以 $\boldsymbol{Y}$ 也是满秩矩阵，故我们可通过 $O(n^3)$ 次的计算得到 $\boldsymbol{Y}^{-1}$，也即得到了 $\boldsymbol{P} = \boldsymbol{Y}^{-1}\boldsymbol{X}$。

得到 $\boldsymbol{P}$ 后，用下面的方法就可得到 $\boldsymbol{SG}$。

若假设 k 个不同并且线性无关的消息 $M_1, M_2, \cdots, M_k$ 的签名分别是 $C_1, C_2, \cdots, C_k$。由于

$$C_i = (E_i + M_i\boldsymbol{SG})\boldsymbol{P},\ i = 1, 2, \cdots, k,$$

故 $C_i\boldsymbol{P}^{-1} = (E_i + M_i\boldsymbol{SG})$。

通过译码算法得到 E_i，从而可得 k 个等式：

$$E_i + C_i\boldsymbol{P}^{-1} = M_i\boldsymbol{SG},\ i = 1, 2, \cdots, k。$$

用同样的构造方法，我们构造出矩阵方程：$\boldsymbol{D} = \boldsymbol{MSG}$。其中

$$\boldsymbol{D} = \begin{matrix} C_1\boldsymbol{P}^{-1} + E_1 \\ C_2\boldsymbol{P}^{-1} + E_2 \\ \cdots \\ C_k\boldsymbol{P}^{-1} + E_k \end{matrix}, \quad \boldsymbol{M} = \begin{matrix} M_1 \\ M_2 \\ \cdots \\ M_k \end{matrix}$$

由于 $M_1, M_2, \cdots, M_k$ 的线性无关性,则 $\boldsymbol{M}$ 有逆矩阵 $\boldsymbol{M}^{-1}$,从而 $\boldsymbol{M}^{-1}\boldsymbol{D} = \boldsymbol{SG}$。求 $\boldsymbol{M}^{-1}$,其工作量仅为 $O(n^3)$。

由上面的叙述可知利用这种选择性明文攻击,仅需 $O(n^3)$ 就能破译 Xinmei 方案。

10.6　批验证协议

在数字签名方案中,有时为了验证一个消息的有效性需耗费大量的时间。为了解决这类问题,有人提出了批验证协议 (Batch Verification Protocol)的概念。对于多组消息的签名,批验证协议可以提高签名和验证的效率。每一个批验证协议都包含签名收集协议和批验证原则两部分。在签名产生过程中根据签名收集协议一次产生多个消息的签名,在签名验证过程中,签名验证者根据批验证协议一次验证多个签名的有效性。

安全的批验证协议需满足如下的条件:

(1) 签名者在签名过程中一次产生多个消息的签名;

(2) 多个签名的有效性由签名验证者一次验证完成;

(3) 多个有效的签名一定满足批验证协议中的批验证原则;

(4) 在批验证协议中有效的多个签名一定是有效的签名。

根据签名产生过程的不同,批验证协议可分为交互式和非交互式两类。在交互式批验证协议中,签名者需同签名验证者交互地传递信息;而在非交互式批验证协议中,签名者不需同签名验证者交互地传递信息。

下面分别介绍一种非交互式批验证协议和一种交互式批验证协议。

1. Harn 非交互式批验证协议

系统参数:设 p 是一 512 位到 1024 位的大素数,它满足 Z_p 中的离散对数问题是难解决的,q 是 160 位长的素数,且 $q \mid p-1$, $g \in Z_p^*$ 是 Z_p 域中的 q 次单位根。h 为一个 Hash 函数,定义

$$\mathbf{K} = \{(p, q, g, y, x): y = g^x \bmod p\},$$

公开密钥: p, q, g, y,私有密钥: x。

签名算法:对于待签的消息 m_i, $(i = 1, 2, \cdots, n)$,签名者 i 随机选取 $k_i \in Z_q$,计算 $r_i = g^{k_i} \bmod p$, $s_i = r_i k_i - h(m_i)x \bmod q$, (r_i, s_i), $(i = 1, 2, \cdots, n)$即为生成的签名。

批验证算法:验证者收到消息的签名 (r_1, s_1), (r_2, s_2), $\cdots$, (r_n, s_n)后,验证

$$\mathrm{Ver}(m, s, r) = \mathrm{TRUE} \quad \prod_{i=1}^{n} r_i = g^{\sum_{i=1}^{n} s_i r_i^{-1}} y^{\sum_{i=1}^{n} h(m_i) r_i^{-1}} \bmod p \bmod q$$

2. N-M-R-V 交互式批验证协议

系统参数:设 p 是一 512 位到 1024 位的大素数,它满足 Z_p 中的离散对数问题是难解决的,q 是 160 位长的素数,且 $q \mid p-1$, $g \in Z_p^*$ 是 Z_p 域中的 q 次单位根。h 为一个

Hash 函数,定义

$$K = \{(p, q, g, y, x): y = g^x \bmod p\}$$

公开密钥: p, q, g, y,私有密钥: x。

签名算法: 对于消息 $m_i\ (i = 1, 2, \cdots, n)$,签名者 i 随机选取 $k_i \in Z_q$,计算 $r_i = g^{k_i} \bmod p$,并将 r_i 发送到验证者;验证者随机选取一个 e 比特的随机数 b_i,并将 b_i 发送到签名者 i;签名者计算 $s_i = k_i^{-1}(h(m_i, b_i) + r_i x) \bmod q$,$(s_i, r_i, b_i)$即为 i 对消息 m_i 的签名。

批验证算法: 验证者收到签名(s_i, r_i, b_i),$(i = 1, 2, \cdots, n)$后,验证

$$\mathrm{Ver}(m, s, r, b) = \mathrm{TRUE} \quad \prod_{i=1}^{n} r_i = g^{\sum_{i=1}^{n} s_i^{-1} h(m_i, b_i)} y^{\sum_{i=1}^{n} s_i^{-1} r_i} \bmod p。$$

注 记

数字签名是保证数据完整性、不可抵赖性和身份识别的一个很好的密码手段。本章主要介绍了数字签名的一般模型和多种签名体制,这些签名体制很多都是由于实际需求而产生的,并且现代的签名体制主要是基于公钥的思想。本章的主要参考文献是本书参考文献[24],[15],[39]。欲了解更多签名体制的读者可参看本书参考文献[24]。

习 题 十

1.一个数字签名体制的一般形式是怎样的?

2.回顾一下群签名、代理签名、批验证协议、多重数字签名的概念及各自所满足的安全特性。

3.证明 Harn 广播多重数字签名方案的正确性。

4.证明定理 10.4。

5.盲签名是这样一种签名体制: 签名者看不见他签名的对象。我们介绍一种基于 RSA 体制的盲签名。设 d 为签名者的私钥,e 和 n 是签名者的公钥。附加的参数 k,被称为盲因子,由消息的提供者选定:

提供者隐藏信息 M: $M' = Mk^e \bmod n$;

签名者计算盲签名: $S' = (M')^d \bmod n = kM^d \bmod n$;

提供者除去盲因子: $S = S'k^{-1} = M^d \bmod n$。

试问: 我们能否基于第 9 章介绍的一些公钥体制,如离散对数的、纠错码的和背包密码体制来设计盲签名?

6.研究题: 我们能否想出一种针对 Xinmei 方案的 Alabhadi-Wieker 攻击方法的改进方法。

7.研究题: 对 Harn 非交互式批验证协议进行安全分析,并试着提出攻击方法。

第 11 章　杂凑(Hash)函数

在引论中已经提到杂凑(Hash)函数也是密码学的一个基本工具,在密码学中有许多应用,特别是在数字签名和消息的完整性检测方面有重要的应用。

从 ELGamal 数字签名方案及数字签名标准可见,一个消息的数字签名是很长的,而且消息越长,其数字签名也越长(为消息长的 2 倍)。为了克服这一缺点,通常在对消息签名之前,先用一个杂凑函数将消息压缩为一个短得多的摘要,再对消息摘要签名。

为了保证一个文件的完整性(不被非法改动),文件的所有者通常先用一个杂凑函数计算该文件的杂凑值,并将其秘密保存,然后再将文件存放在可公开的地方(如计算机里)。每当他要使用该文件时,先计算所存放文件的杂凑值,并与自己秘密保存的杂凑值比较。若二者相等,则文件是完整的(没被改动),否则,说明文件已被改动过。

从上述应用的有效性和安全性,对杂凑函数 h 提出下列几点直观要求:

(1) h 能应用于任意长的消息或文件 x;

(2) h 的值(输出的杂凑值)是固定长的,但要足够长;

(3) 计算 h 的值 $h(x)$是容易的;

(4) 给定算法 h,要找两个不同的消息 $x_1 \neq x_2$,使其杂凑值 $h(x_1) = h(x_2)$是困难的(计算上不可行的)。

本章将介绍杂凑函数的一般理论和构造方法。

11.1　杂凑函数的定义

给定函数族 $h_n: \{0,1\}^n \to \{0,1\}^m; n > m$,记集 $C_n = \{(x, y); x \neq y, x, y \in h_n^{-1}(h_n(x))\}$,$(x, y) \in C_n$ 称为碰撞消息。记 $C_n(x) = \{y; (x, y) \in C_n\}$。首先给出用来构造杂凑函数的压缩函数的定义。

定义 11.1　一个函数族 $h_n: \{0,1\}^n \to \{0,1\}^m; n > m$ 称为强无碰撞压缩函数族,若下面两个条件成立:

(1) 计算 $h_n(x)$是容易的,即存在一个多项式时间算法 F,若 F 的输入为 1^n 和 $x \in \{0,1\}^n$,则其输出为 $h_n(x)$。

(2) 给定算法 F 要找两个不同的消息 $x_1 \neq x_2$ $|x_1| = |x_2|$，使得 $h_{|x_1|}(x_1) = h_{|x_2|}(x_2)$ 是困难的，即对每一个多项式时间概率算法 M'，每一正多项式 $p(n)$ 和一切充分大的 n 有

$$\Pr[M'(h_n(U_n), 1^n) \in C_n(U_n)] < 1/p(n), \tag{11.1}$$

其中 U_n 表示 $\{0,1\}^n$ 上的均匀分布随机变量。

定理 11.1 若一个函数族 $h_n: \{0,1\}^n \to \{0,1\}^m; n > m$ 为一强单向函数族，则它也是一个强无碰撞压缩函数族。反之，若函数族 $h_n; n > m$ 为一强无碰撞压缩函数族，且对(充分大的)每个 n 及 $x \in \{0,1\}^n$，$h_n(x)$ 的原象集 $h_n^{-1}(h_n(x))$ 中至少包含两个原象，则它也是一个强单向函数族。

证明 (1) 若函数族(序列) $h_n; n > m$ 为一强单向函数族，则由定义 5.5 的特殊情形(序列情形)有，对每一多项式时间概率算法 M'，每一正多项式 $p(n)$ 及一切充分大的 n，

$$\Pr[M'(h_n(U_n), 1^n) \in h_n^{-1}(h_n(U_n))] < 1/p(n), \tag{11.2}$$

成立，即在(5.9)式中设指标集 I_n 为 1^n，随机变量 X_n 为均匀分布随机变量 U_n。由于 $C_n(x) \supseteq h_n^{-1}(h_n(x))$ 对一切 $x \in \{0,1\}^n$ 成立，故由(11.2)成立推出(11.1)成立。证得函数族 $h_n; n > m$ 为一强无碰撞压缩函数族。

(2) 用反证法，若函数族 $h_n; n > m$ 不是强单向函数族，则由定义 5.5 的特殊情形(序列情形)，存在一个多项式时间概率算法 M 及一个正多项式 $q(n)$，使

$$\Pr[M(h_n(U_n), 1^n) \in h_n^{-1}(h_n(U_n))] \geq 1/q(n) \tag{11.3}$$

对 n 的一个无穷子序列成立。由定理中的条件对子序列中的每个 n 及 $x \in \{0,1\}^n$，$h_n(x)$ 的原象集 $h_n^{-1}(h_n(x))$ 至少包含二个原象。今考虑使 $M(h_n(x), 1^n) \in h_n^{-1}(h_n(x))$ 成立的 x，设 $h_n^{-1}(h_n(x))$ 中有 k 个原象，记作 $y_1, \cdots, y_k, (x = y_1)$。在 $U_n \in h_n^{-1}(h_n(x))$ 的条件下，$U_n = y_j$ 的条件概率为

$$\Pr[U_n = y_j \mid U_n \in h_n^{-1}(h_n(x))] = 1/k, j = 1, \cdots, k。$$

记 $M(h_n(x), 1^n) = y_j$ 的概率为 p_j，$j = 1, \cdots, k$，$\sum_{j=1}^{k} p_j = 1$。由于概率算法 M 的内部扔硬币结果与随机变量 U_n 是统计独立的，故有

$$\begin{aligned}
&\sum_{j=1}^{k} \Pr[M(h_n(x), 1^n) \in C_n(y_j) \mid U_n \in h_n^{-1}(h_n(x))] \\
&= \sum_{j=1}^{k} \Pr[M(h_n(x), 1^n) \neq y_j, U_n = y_j \mid U_n \in h_n^{-1}(h_n(x))] \\
&= \sum_{j=1}^{k} \frac{1 - p_j}{k} = \sum_{j=1}^{k} \frac{1}{k} - \frac{p_j}{k} = 1 - \frac{1}{k} \geq \frac{1}{2}。
\end{aligned} \tag{11.4}$$

最后一个不等式用了 $k \geq 2$ 的假定。联合(11.4)式和(11.3)式得

$$\Pr\left[M(h_n(U_n),1^n)\in C_n(U_n)\right]\geq 1/2q(n) \tag{11.5}$$

对 n 的一个无穷子序列成立。不满足定义 11.1 的条件(2),故函数族 $h_n; n>m$ 不是强无碰撞压缩函数族。

强无碰撞压缩函数族中的函数也称无碰撞压缩函数或单向压缩函数。

类似于单向函数族的定义,强无碰撞压缩函数族的定义也可推广到定义在正整数集 $n; n>m$ 的一个无穷子集上的强无碰撞压缩函数族。如

$$h_{l(n)}: \{0,1\}^{l(n)}\rightarrow\{0,1\}^{m}; l(n)>m,$$

其中 $l(n)$为 n 的一个多项式。

现在给出杂凑函数的定义。

定义 11.2　一个强无碰撞杂凑函数是一个满足下列条件的函数 h:

(1) h 可应用于任意长的消息或文件;

(2) h 的值(杂凑值)是固定长的,但要足够长才能抵抗(后面要介绍的)生日攻击;

(3) 计算 h 的值 $h(x)$是容易的,即 $h(x)$是多项式时间可计算的;

(4) 给定算法 h,要找两个不同的消息 $x_1\neq x_2$,使其杂凑值 $h(x_1)=h(x_2)$是困难的(计算不可行的),即 h 是由强无碰撞压缩函数族中的压缩函数所构造的(构造方法见后)。

在大多数密码学的有关杂凑函数的书和文献中还给出了弱无碰撞杂凑函数的定义,那些定义是直观的。为了方便读者阅读有关文献,这里也给出弱无碰撞杂凑函数的直观定义。

定义 11.3　一个弱无碰撞杂凑函数是一个满足下列条件的函数 h:

条件(1),(2),(3)与定义 11.2 中的条件(1),(2),(3)相同。

条件(4):给定算法 h 及一个随机选出的消息 x_1,要找另一个消息 $x_2\neq x_1$,使得它们的杂凑值 $h(x_1)=h(x_2)$是困难的(计算不可行的)。

考察定义 11.2 和定义 11.3 的条件(4)可见,强无碰撞杂凑函数比弱无碰撞杂凑函数更安全。因为弱无碰撞杂凑函数并不要求任意找出两个不同的消息 $x_1\neq x_2$ 使 $h(x_1)=h(x_2)$是困难的。因此,可能有一些这样的碰撞消息对是容易找到的。在随机选出消息时,不能故意选这些消息,但容易找到的碰撞消息对不可太多,否则条件(4)将不满足。

若要将弱无碰撞杂凑函数用于给定(非随机)的消息,则需要在杂凑函数中引入随机性。随机化杂凑函数的方法主要有三种:(1)将消息先用好的分组密码和随机密钥加密为密文,且将随机密钥加在密文前;(2)在消息前加上一个随机选择的前缀,然后再用杂凑算法;(3)从许多杂凑函数中随机选出一个杂凑函数来使用。

应当注意的是:弱无碰撞杂凑函数随着重复使用次数的增加而安全性逐渐降低。这是由于用同一个弱无碰撞杂凑函数的消息越多,找到不同消息发生碰撞的机会就越大,从而系统的总体安全性降低。对于强无碰撞杂凑函数,则不会因其重复使用而降低安全性。

强无碰撞杂凑函数有时也称为强单向杂凑函数,也可简称为无碰撞杂凑函数或单向

杂凑函数。

11.2 无碰撞杂凑函数的构造方法

定理 11.1 已证明,一个强单向函数族 $h_n; n > m$ 也是一个强无碰撞压缩函数族,因此绝大多数无碰撞杂凑函数都是用单向压缩函数构造的。但单向函数的存在性在理论上尚未证明,因此,我们只能用候选单向函数来构造无碰撞杂凑函数。本节首先介绍一个用单向压缩函数构造无碰撞杂凑函数的一般方法,然后介绍若干类典型的无碰撞杂凑函数的构造方法。对杂凑函数的可能的攻击将在 11 .3 节中讨论。

11.2.1 用单向压缩函数构造无碰撞杂凑函数的一般方法

设 h_l: $0,1^{l} \to 0,1^{m}$ 为一单向压缩函数,其中 $l \geq m+2$ 为一选定的正整数。我们要构造一个满足定义 11.2 的杂凑函数 h: $\bigcup_{n \geq l} 0,1^{n} \to 0,1^{m}$。方法如下:首先将输入 h 的消息 $x \in 0,1^{*}$ 分为长 $l-m-1$ 的组,记作 $x = x_1 x_2 \cdots x_r$。若 x 的长 $|x|$ 不能被 $l-m-1$ 整除,则在 x_r 后面添加 d 个 0,使 $n = |x| + d$ 被 $l-m-1$ 整除。不妨将 $x_r 0^d$ 仍记作 x_r,使 $|x_r| = l-m-1$,,再附加一个分组 x_{r+1},它由 d 的二进数表示在其前面添加若干个 0 构成,使 $|x_{r+1}| = l-m-1$。杂凑函数 $h(x)$ 的值由下面的迭代算法定义:

$$h_1 = h_l(0^{m+1} x_1),$$
$$h_{i+1} = h_l(h_i 1 x_{i+1}) \quad i = 1,2\cdots, r,$$
$$h(x) = h_{r+1}。$$

定理 11.2 杂凑函数 h 和用来构造它的单向压缩函数 h_l 有几乎同样的安全性。更确切地说,若能找到 h 的一对碰撞消息 $x \neq x'$,使 $h(x) = h(x')$,则可以利用这对消息 x, x' 在多项式时间内找到 h_l 的一对碰撞消息 $y \neq y'$,使 $h_l(y) = h_l(y')$。

证 按照杂凑函数 h 的构造方法,记 $x = x_1 x_2 \cdots x_r$, $x' = x_1' x_2' \cdots x_r'$; $x_r = x_r 0^d$, $x_r' = x_r' 0^{d'}$; x, x' 的附加分组分别为 x_{r+1} 和 $x_{r'+1}$。输入 x, x' 的迭代算法计算结果分别记作 $h_1, h_2, \cdots, h_{r+1}$ 和 $h_1', h_2', \cdots, h_{r'+1}'$。

(1) 若 $|x| \not\equiv |x'| \bmod (l-m-1)$,则 $d \neq d'$,从而 $x_{r+1} \neq x_{r'+1}'$。而 $h_l(h_r 1 x_{r+1}) = h_{r+1} = h(x) = h(x') = h_{r'+1}' = h_l(h_{r'}' 1 x_{r'+1}')$,因此 $h_r 1 x_{r+1}$ 和 $h_r' 1 x_{r'+1}'$ 是 h_l 的一对碰撞消息。

(2) 若 $|x| \equiv |x'| \bmod (l-m-1)$,则可分二种情况讨论:

a .若 $|x| = |x'|$,则 $d = d'$, $r = r'$, $x_{r+1} = x_{r+1}'$。由 h 的定义有 $h_l(h_r 1 x_{r+1}) = h_{r+1} = h(x) = h(x') = h_{r+1}' = h_l(h_r' 1 x_{r+1}')$。若 $h_r \neq h_r'$,则 $h_r 1 x_{r+1}$ 和 $h_r' 1 x_{r+1}'$ 是 h_l 的一

对碰撞消息。否则 $h'_r = h_r$,由迭代算法有

$$h_l(h_{r-1}1x_r) = h_r = h_r' = h_l(h'_{r-1}1x_r')。$$

若 $h_{r-1} \neq h'_{r-1}$或 $x_r \neq x_r'$,则 $h_{r-1}1x_r$ 和 $h'_{r-1}1x_r'$是 h_l 的一对碰撞消息;否则 $h_{r-1} = h'_{r-1}$, $x_r = x_r'$。于是由迭代算法可继续向前面找,直到最后得

$$h_l\ 0^{m+1}x_1 = h_1 = h_1' = h_l\ 0^{m+1}x_1'。$$

若 $x_1 \neq x_1'$,则 $0^{m+1}x_1$ 和 $0^{m+1}x_1'$即为 h_1 的一对碰撞消息;否则由 $x_i = x_i' i = 1, 2, \cdots, r+1$ 推出 $x = x'$,与假设矛盾。因此我们总能找到 h_l 的一对碰撞消息。

b.若 $|x| \neq |x'|$,则 $d = d'$, $x_{r+1} \neq x'_{r'+1}$。但 $r \neq r'$,不妨设 $r < r'$。首先我们可按 a.中的方法一步一步地找 h_l 的碰撞消息。若一直找不到,则最后可得

$$h_l\ 0^{m+1}x_1 = h_l = h_{r'-r+1}' = h_l\ h'_{r'-r}1x'_{r'-r+1}。$$

因为 $0^{m+1}x_1$ 的第 $m+1$ 位是 0,而 $h'_{r'-r}1x'_{r'-r+1}$的第 $m+1$ 位是 1,故它们就是 h_l 的一对碰撞消息。

上述构造杂凑函数的方法称为 Damgard 的杂凑函数设计原理或称 Merkle 的 meta 方法,因为这一方法是他们二人独立发明的。由定理 11.2 可见,用这种设计原理构造的杂凑函数与所用的压缩函数几乎有相同的安全性。

11.2.2　用分组加密函数构造杂凑函数

对于一个安全的私钥分组密码,给了密文和相应的明文,要找出所用的密钥是困难的。因此,安全的私钥分组加密函数可以认为是候选单向函数。

一个一般的私钥分组加密函数可表示为

$$y = E(x, k),$$

其中 x 为输入的明文消息,设其长为 m, k 为所用密钥,设其长为 l, y 为输出的密文,其长也是 m。故函数 E 将 $0,1^{m+l}$中的序列压缩为 $0,1^{m}$ 中的序列,也是一个单向压缩函数,基于分组加密函数的杂凑算法都是迭代算法,可将任意 n 长的消息压缩为 m 长的摘要。一般方法是先将 n 长消息或文件分组,分组长等于加密函数 E 的输入长 m 或密钥长 l(依赖于不同的杂凑算法)。若要计算杂凑值的消息 x 的长不是分组长的整数倍,则将消息编码,如添加若干校验位,使码长为分组长的整数倍,再附加一个包含所加校验位个数 d 的二进数表示的分组。为了引入随机化,通常用一个初值分组,用 IV 表示,它的值可以是公开的,或随密钥改变,或作为消息的前缀,迭代算法与 11.2.1 节中的迭代算法类似。下面介绍几个基本的杂凑算法,可用于任选的加密函数。为了表述方便,假定消息长 n 为分组长的整数倍。关于这些算法和更多算法的详细介绍,读者可参看本书参考文献[32]。

记消息 x 的分组为　$x = x_1 x_2 \cdots x_t$。

(1) Rabin 方案(分组长 $|x_i| = l$)

$$h_0 = IV$$

$$h_i = E(h_{i-1}, x_i),\ i = 1, 2, \cdots, t$$

$$h(x) = h_t$$

(2) 密码分组链接方案(分组长为 $|x_i| = m$)

$$h_0 = IV$$

$$h_i = E(x_i + h_{i-1}, k),\ i = 1, 2, \cdots, t$$

$$h(x) = h_t$$

其中 + 表示 F_2^m 中的向量加法。在算法的每次迭代中也可用不同的密钥 k_i。这时，$y = h_1\ h_2 \cdots h_t$ 可作为明文 x 的对应密文。

(3) 密钥链接方案(分组长 $|x_i| = m = l$)

$$h_0 = IV$$

$$h_i = E(h_{i-1}, x_i + h_{i-1}),\ i = 1, 2, \cdots, t$$

$$h(x) = h_t$$

在应用这一类方案时，要求分组加密函数 E 具有如下性质：给了密文和所用密钥，要找出相应的明文是困难的，但通常的分组加密函数不满足这一要求。Winternitz 建议将加密函数改为

$$h_i = E(x_i, h_{i-1}) + h_{i-1}$$

然后再应用这类方案。

11.2.3 用候选单向函数构造杂凑函数

1. 单向函数输出值链接方案

设 f 为一候选单向函数族中选出的函数，如 5.2.2 节中的 RSA 函数 $f_{(N,e)}$(例 5.4)，Rabin 函数 f_N(例 5.5)。离散对数函数 $f_{(p,g)}$(例 5.7)等。与 11.2.2 节中的密码分组链接方案类似，先将要杂凑的消息 $x \in D^t$ 分组，其中 D 为单向函数的定义域(与值域相同)。

迭代算法如下(记 $x = x_1\ x_2 \cdots x_t,\ x_i \in D$)：

$$h_0 = IV$$

$$h_i = f(h_{i-1} + x_i),\ i = 1, 2, \cdots, t$$

$$h(x) = h_t$$

例如，若 $f = f_{(N,e)}$ 为 RSA 函数，则

$D = D_{(N,e)} = \{1, 2, \cdots, N\}$，$h_i = (x_i + h_{i-1})^e \pmod N$，$i = 1, 2, \cdots, t$(0 与 N 等同)。

若 $f = f_{(p,g)}$ 为离散对数函数，则

$D = D_{(p,g)} = \{1, 2, \cdots, p-1\}$，$h_i = g^{(h_{i-1} + x_i)} \pmod p$，$i = 1, 2, \cdots, t$。

2. 单向无爪函数输出值链接方案

设 $f^0_{(p,g,z)}$，$f^1_{(p,g,z)}$ 为单向无爪函数族中选出的函数(见 5.3.2 节中的例 5.7′)，与

1. 中的方法类似,先将要杂凑的消息 $x \in D^t$ 分组,其中 $D = \{1,2,\cdots,p-1\}$。迭代算法如下(记 $x = x_1 x_2 \cdots x_t$, $x_i \in D$):

$h_0 = IV$,

$h_i = z^{\sigma(x_i)} g^{(h_{i-1} + x_i)} \pmod p$, $i = 1,2,\cdots,t$,

$h(x) = h_t$,

其中 $\sigma(x_i) = x_i \pmod 2$。

11.2.4　软件杂凑算法 MD4 和 MD5

前两节介绍的杂凑算法是基于私钥分组加密函数和为公钥密码体制提供的候选单向函数,由于设计分组密码和设计杂凑函数的目的不完全相同,前二节介绍的杂凑算法速度比较慢。本节将介绍专为杂凑算法设计的软件 MD4 和 MD5,它们的速度要快得多。

MD4 可以将任意长的消息 $x \in \{0,1\}^*$ 杂凑为 128 比特长的消息摘要。其方法如下:

首先由 x 构造一个数组序列

$$M = M[0]\, M[1] \cdots M[N-1],$$

其中每个 $M[i]$ 为 32 比特长数组, $N \equiv 0 \pmod{16}$。

由 x 构造 M 的算法如下:

(1) $d = 447 - |x| \bmod 512$,(当 $d < 0$ 时,按 512 处理);

(2) 置 l 为 $|x| \pmod{2^{64}}$的二进数表示, $|l| = 64$;

(3) $M = x 10^d l$。

在 M 的构造中,先在 x 后面加一个 1,再加 0^d,使得 $x10^d \pmod{512} = 448$,故所得 M 的长为 512 的倍数,因此可将 M 分成 32 比特长数组 $M[0]\, M[1] \cdots M[N-1]$,其个数 N 为 16 的倍数。这里构造 M 的方法与 11.2.1 节中消息 x 的分组方法有点类似。

现在再给出由 M 构造杂凑函数 $h(x)$的杂凑算法:

(1) 给 4 个寄存器 A, B, C, D 赋初值

A = 67452301

B = *efcdab*89

C = 98*badcfe*

D = 10325476

其中 0123456789*abcdef* 中的每个数字或字母都表示 4 位二进数。故一个寄存器可存放或表示一个 32 比特数组。

(2) for i = 0 to (N/16) - 1 do

(3) for j = 0 to 15 do

$$X[j] = M[16i + j]$$

(4) 将 4 个寄存器 A, B, C, D 中的数组存放到另外 4 个寄存器 AA, BB, CC, DD

中:

$$AA = A, BB = B, CC = C, DD = D。$$

(5) 执行第一轮。

(6) 执行第二轮。

(7) 执行第三轮。

(8) $A = A + AA$, $B = B + BB$, $C = C + CC$, $D = D + DD$。

算法的输出 $h(x) = A, B, C, D$,即为 4 个寄存器 A, B, C, D 中的 4 个 32 比特数组的连接,得到 128 比特的消息摘要 $h(x)$。

算法从(2)开始每个循环处理 M 中的 16 个 32 比特数组,从(3)到(8)为一个循环,直到最后一个循环(第 N 16 个循环)结束后输出 $h(x)$。

下面再对算法中涉及的运算及三个轮函数作进一步说明(设 X 和 Y 表示输入的数组):

(1) $X \wedge Y$ 表示 X 和 Y 按位逻辑"与"(and)。

(2) $X \vee Y$ 表示 X 和 Y 按位逻辑"或"(or)。

(3) $X \quad Y$ 表示 X 和 Y 按位逻辑"异或"(xor)。

(4) $\overline{X}$ 表示 X 的按位逻辑"补"(complement)。

(5) $X + Y$ 表示整数模 2^{32} 加法。

(6) $X \lll s$ 表示将 X 循环左移 s 位($0 \leq s \leq 31$)。

以上运算都是很快的。只有模 2^{32} 加法运算,在实现时要考虑所用计算机的结构。MD4 的设计是以 Little-endian 结构为基础的。在一个 Little-endian 中(如 Intel 80 XXX line), a_1, a_2, a_3, a_4 为一个数组的 4 个子组,每个子组 8 比特,把它看作 0 到 255 之间的一个整数的二进数表示,则这个数组 a_1, a_2, a_3, a_4 表示整数 $a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$;若用其他结构的计算机,则需要对数组的表示作相应的修改,算法中的三个轮函数分别使用了函数 f, g 和 h,它们分别定义为:

$$f(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z),$$

$$g(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z),$$

$$h(X, Y, Z) = X \quad Y \quad Z。$$

下面给出三个轮函数的计算程序:

第一轮

(1) for k = 0 to 3 do

(2) $A = A + f(B, C, D) + X[4k] \lll 3$

(3) $D = D + f(A, B, C) + X[4k+1] \lll 7$

(4) $C = C + f(D, A, B) + X[4k+2] \lll 11$

(5) $B = B + f(C, D, A) + X[4k+3] \lll 19$

第二轮

(1) for k = 0 to 3 do

(2) A = A + g B, C, D + X[k] + 5 a827999 ≪3

(3) D = D + g A, B, C + X[4 + k] + 5 a827999 ≪5

(4) C = C + g D, A, B + X[8 + k] + 5 a827999 ≪9

(5) B = B + g C, D, A + X[12 + k] + 5 a827999 ≪13

第三轮

(1) A = A + h B, C, D + X[0] + 6 ed9 eba1 ≪3

(2) D = D + h A, B, C + X[8] + 6 ed9 eba1 ≪9

(3) C = C + h D, A, B + X[4] + 6 ed9 eba1 ≪11

(4) B = B + h C, D, A + X[12] + 6 ed9 eba1 ≪15

(5) A = A + h B, C, D + X[2] + 6 ed9 eba1 ≪3

(6) D = D + h A, B, C + X[10] + 6 ed9 eba1 ≪9

(7) C = C + h D, A, B + X[6] + 6 ed9 eba1 ≪11

(8) B = B + h C, D, A + X[14] + 6 ed9 eba1 ≪15

(9) A = A + h B, C, D + X[1] + 6 ed9 eba1 ≪3

(10) D = D + h A, B, C + X[9] + 6 ed9 eba1 ≪9

(11) C = C + h D, A, B + X[5] + 6 ed9 eba1 ≪11

(12) B = B + h C, D, A + X[13] + 6 ed9 eba1 ≪15

(13) A = A + h B, C, D + X[3] + 6 ed9 eba1 ≪3

(14) D = D + h A, B, C + X[11] + 6 ed9 eba1 ≪9

(15) C = C + h D, A, B + X[7] + 6 ed9 eba1 ≪11

(16) B = B + h C, D, A + X[15] + 6 ed9 eba1 ≪15

MD4 的速度很快,在 Sun SPARC stations 上软件实现的速度达到 11 2 Mbit/ s。由于它的安全性不基于任何密码体制和已知单向函数,故它的安全性只能像其他候选单向函数一样由时间来证实。目前,MD4 杂凑算法还未找到有效的攻击方法能攻破它。

MD5 是在 MD4 算法的基础上作了一些改进,使它更为安全。主要修改有以下 6 点:

(1) 增加了第四轮,第四轮所使用的函数为

$$i(X, Y, Z) = (X \vee \overline{Z}) \quad Y。$$

(2) 第二轮所使用的函数改为

$$g(X, Y, Z) = (X \wedge Z) \vee (Y \wedge 瑃)。$$

(3) 在进入第二轮和第三轮,输入数组的次序被改变。

(4) 每一轮中的移位数作了改变,且各轮中的移位数都不相同。

(5) 每一步有一个惟一的加法常数。

(6) 每一步加上了前一步的结果。

MD4 是 Rivest 于 1990 年设计的,他于 1991 年为了加强其安全性,对 MD4 作了上述

改进,称修改后的杂凑算法为 MD5。

11.2.5 安全 Hash 标准(SHS)

安全 Hash 标准(SHS)是 1992 年 1 月 31 日在美国联邦记录中发表的,在 1993 年 5 月 11 日被采用为标准。1994 年 7 月 11 日修正了 SHS 中一个技术缺陷。1995 年 4 月 17 日公布了修改了的版本。SHS 比 MD4 更复杂,但与 MD4 基于同一设计方法,它是 MD4 的变形。这里我们不打算详细介绍 SHS,只是指出 SHS 中使用的几点改变:

(1) SHS 的设计是以 big-endian 结构为基础的。在一个 big-endian 结构中(如 Sun SPARC station),若 a_1, a_2, a_3, a_4 为一个数组中的 4 个子组,则这个数组表示整数 $a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$(参看 11.2.4 节中 MD4 所用的 little-endian)。

(2) SHS 可将任意长的消息 x 杂凑到 160 比特长的消息摘要 $h(x)$。因此算法中要用 5 个寄存器 A, B, C, D, E。

(3) SHS 与 MD4 一样每次处理 16 个 32 比特数组,可是与 MD4 不同,SHS 首先将输入的 16 个数组扩展为 80 个数组,然后再对这 80 个数组执行四轮计算。设输入的 16 个数组为 $X[0], X[1], \cdots, X[15]$,则其他 64 个数组由下面的递推公式计算:

$$X[j] = X[j-3] \quad X[j-8] \quad X[j-14] \quad X[j-16], 16 \leq j \leq 79。$$

后来将这个公式修改为

$$X[j] = X[j-3] \quad X[j-8] \quad X[j-14] \quad X[j-16] \lll 1, 16 \leq j \leq 79。$$

更详细的介绍可参看本书参考文献[26]。

11.3 杂凑函数的攻击方法与安全性

评价一个杂凑函数的安全性,最好的方法是看是否有有效的攻击方法来破译它,换句话说,攻击者是否能在可实现的条件下,找出该杂凑函数的一对或多对碰撞消息。一般假定攻击者知道杂凑算法,他可以实行选择消息(明文)攻击,即他可以随意地选择若干消息,算出它们的杂凑值,再试图计算出与它们碰撞的消息。目前已经提出了一些攻击杂凑函数和计算碰撞消息的方法。其中,有的是一般方法,可用来攻击任何类型的杂凑函数,如生日攻击;有的是特殊方法,只能用来攻击某些特殊类型的杂凑函数,如中间相遇攻击,可以用来攻击具有分组链接结构的杂凑函数,如 11.2 节中介绍的密码分组链接方案、密钥链接方案和单向函数输出值链接方案等等;又如修正分组攻击,主要用于攻击基于模算术的杂凑函数。这一节将简要介绍这些攻击方法及保证杂凑函数安全的必要条件。

11.3.1 生日攻击

生日攻击的思想来源于概率论中一个著名的问题,称为生日问题。该问题是问在一

个教室里至少要有多少个学生才能使得有两个学生的生日相同的概率不小于 1/2,其答案是 23。这个回答似乎不很直观,但可以给出证明。

首先考虑一个初等概率论问题。设有 J 个球,将它们逐个随机地扔进 N 个匣子里,问存在一个匣子中至少有两个球的概率有多大。容易求出没有一个匣子中至少有两个球的概率为

$$1-\frac{1}{N}\quad 1-\frac{2}{N}\quad \cdots\quad 1-\frac{J-1}{N}\quad =\prod_{j=1}^{J-1}\quad 1-\frac{j}{N}\quad ,$$

故存在一个匣子中至少有两个球的概率为

$$p=1-\prod_{j=1}^{J-1}\quad 1-\frac{j}{N}\quad 。$$

由于 $e^{-x}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\cdots$,故当 x 较小时有 $1-x\approx e^{-x}$。因此有

$$p\approx 1-\prod_{j=1}^{J-1}e^{-\frac{j}{N}}=1-e^{-\frac{J(J-1)}{2N}}。\qquad (11.6)$$

或写成

$$J^2-J\approx -2N\ln(1-p)。\qquad (11.7)$$

当 N 比 J 大得多时成立。若忽略上式中的一次项 J,得

$$J\approx\quad -2\ln\quad 1-p\quad N。\qquad (11.8)$$

现在我们将所得结果应用到生日问题中去。J 个球看作 J 个学生,他们逐个进入教室,每个学生的生日为 365 天中的某一天是随机的,故可以看作 J 个球随机地扔进 365 个匣子里。用 $p=0.5$, $N=365$ 代入(11.8)式得 $J\approx 22.49$。这就证明了生日问题的答案是 23。

现在再来考虑如何将这个方法应用于攻击杂凑函数 $h(x)$。设 $0,1^n$ 为消息集,$0,1^m$ 为杂凑值集($n>m$)。不妨设对每一个 $x\in 0,1^n$,原象集的子集 $h^{-1}(h(x))\cap 0,1^n$ 至少包含两个原象,这个条件是容易满足的(取 n 适当大)。从 $0,1^n$ 中随机选出 J 个消息 $x_1, x_2, \cdots, x_J$,每个消息的杂凑值 $h(x_j)(1\leq j\leq J)$ 可以取 $0,1^m$ 中的某个值也是随机的,因此我们可用 $N=2^m$ 代入公式(11.8)来计算为了得到碰撞消息的概率为 p 所需要选出的消息数 J。当 p 给定时,如 $p=0.5$,则 J 的大小只依赖于杂凑值(消息摘要)的长 m。由于 J 的大小表示攻击者为了找到一对碰撞消息所需进行的计算,它与 $N=2^{\frac{m}{2}}$ 成正比,由此给出保证杂凑函数安全的一个必要条件,按目前的计算技术条件而言,一个杂凑函数的输出值 $h(x)$ 的长 m 应不小于 128 比特才是安全的,这时攻击者要找到一对碰撞消息,至少要计算 2^{64} 个消息的杂凑值。因此用 DES 构造的杂凑函数是不够安全的,安全 Hash 标准的输出长度选为 160 比特也是为了抵抗生日攻击。

11.3.2 特殊攻击

1. 中间相遇攻击

中间相遇攻击的思想与生日攻击的思想基本相同,但它只适用于攻击具有分组链接结构的杂凑函数。攻击者可随意选出若干消息分组,将它们分为两部分,第一部分消息分组从初值开始进行迭代,到中间某一步为止,得到 J_1 个输出;第二部分消息分组从杂凑值 $h(x)$(可随意选定)开始用逆函数反向迭代,到中间某一步为止,得到 J_2 个输出,然后比较这两部分输出,若能找到一对输出值相等,则可得到一对碰撞消息,如以 11.2.2 节中的密码分组链接方案为例。第一部分消息按

$$h_0 = IV,\ h_i = E\ x_i' + h_{i-1},\ k\ ,\ i = 1, \cdots, t/2$$

迭代(不妨设 t 为偶数)。第二部分消息按

$$h_t' = h(x),\ h_{i-1}' = D(h_i', k) + x_i',\ i = t, \cdots, t/2 + 1$$

迭代,其中 D 为与加密函数 E 对应的解密函数,即 $h_{i-1}' = D\ h_i', k\ + x_i'$ 是 $h_i = E\ x_i' + h_{i-1}, k$ 的逆函数。因此两部分输出值中若有一对相等,即有 $h_{\frac{t}{2}} = h_{\frac{t}{2}}'$,则可将对应的消息链接成一个消息 $x' = x_1'x_2' \cdots x_t'$ 使 $h(x') = h(x)$。由于消息 x 和 x' 是攻击者随意选的,故可选 $x \neq x'$,这就得到一对碰撞消息。类似于生日攻击,可以计算为了得到碰撞消息的概率为 p 所需要选出的消息数 $J = J_1 + J_2$。这时,J 不仅依赖于分组加密函数的分组长(消息摘要长) m,而且还与所用分组加密函数的性质有关。这是因为中间相遇攻击可以看作是对分组密码的选择明文/密文攻击,攻击者不但可以随意选择明文消息(第一部分消息)计算出相应的密文,而且可以随意选择密文(第二部分消息及杂凑值 $h(x)$)计算出相应的明文。为了抵抗选择明文/密文攻击,要求分组加密函数的分组长至少为 128 比特,且有伪随机置换性质(参看第 6.4 节和 6.5 节),这是具有分组链接结构的杂凑函数保证安全的一个必要条件。有关这方面的更多研究和讨论,读者可参看本书参考文献[32]及其所引文献。

防止中间相遇攻击的另一个方法是避免用可逆的分组加密函数。

2. 修正分组攻击

这种攻击方法是用伪造消息与一个分组连接,目的是要修正杂凑值,使它等于所期望的值。修正分组攻击通常用于最后一个分组,故也称修正最后分组攻击。基于模算术的杂凑函数对修正最后分组攻击特别敏感,如基于 RSA 函数和离散对数的单向函数输出值链接方案就属于这一类杂凑函数。

3. 差分攻击

差分分析是攻击分组密码的一种方法,在第 8.6.1 节中已作了详细介绍。它可以用来攻击基于分组加密函数构造的杂凑函数,也可以用于攻击另外一些特殊的杂凑函数,如 MD5、Snefru 等软件杂凑函数。

11.4 时　戳

时戳是密码学中用来证明某一事件发生时间的一种方法。例如在使用第 10 章中介绍数字签名方法时,如果不用时戳使得消息被签名的时间可以得到证明,那就会出现一些问题。如一个签名者的签名算法可能被一个不诚实者偷走,则签名者在此之前所作的签名的真伪性就无法辨别;也有可能签名者签了名以后又想否认,故意将签名算法泄露出去,然后声称他对某个消息的签名是一个伪造签名。另一个使用时戳的例子是在密钥分配方案中为的是要规定密钥使用的有效期(见第 14.2 节)。下面将介绍一个简单的时戳方法。

签名者 B 自己可产生一个可信的时戳。B 可首先收集若干目前公开可获得消息,这些消息在它们发生之前是不可预测的。例如,这些消息可由北京前一天所有主要股票的交易额构成,将这些消息记作 pub。

现在若 B 想时戳他对消息 x 的签名。设 h 为一公开的杂凑函数,B 可采用下面的方法:

(1) B 计算 $z = h(x)$;

(2) B 计算 $z' = h(z \| \text{pub})$, $z \| \text{pub}$ 表示 z 和 pub 的连接;

(3) B 计算 $y = \text{Sig}_{k_2}(z')$(z'的签名);

(4) B 在第二天的报纸上公布(z, pub, y)。

这样做的目的是使人们相信(证明)B 的签名时间只能在 pub 发生到报纸公布(z, pub, y)这段时间内,而且 B 没有泄漏消息 x 只是公布了 z,必要时 B 能证明 x 是他签的消息。

如果存在一个可信的时戳服务中心 TSS,那么上述方法中的(4)可以改为:B 将(z, pub, y)发送给 TSS,TSS 加上日期 D 并对(z, pub, y, D)签名。

注 记

本章主要介绍了杂凑(Hash)函数的定义、构造方法和攻击方法,主要参考了本书参考文献[15],[26]和[32]。为了和第 5 章中定义的单向函数族相联系,本章引进无碰撞压缩函数族定义,并证明了在很弱的条件下,单向函数族与无碰撞压缩函数族等价。在有些文献中把压缩函数也称为 Hash 函数,这是因为用压缩函数可以构造安全性相同的 Hash 函数(本书参考文献[26]中称为 Hash 函数的延拓准则),想知道更多 Hash 函数知识的读者可参看本书参考文献[32]和[26]第 8 章所引的文献[42]。

习题十一

1. 用离散对数函数族(见例 5.7)构造无碰撞压缩函数族,构造方法如下:

设 p, $q(p=2q+1)$为两个大素数, e, g 为 Z_p 的两个本原元,设计算离散对数 $\log_e g$ 是困难的,定义压缩函数 $h(x_1, x_2) = e^{x_1} g^{x_2} (\bmod p)$, $x_1, x_2 \in 0, \cdots, q-1$ 。

证明:若给了 h 的一对碰撞消息(x_1, x_2)和(x_3, x_4),即 $h(x_1, x_2) = h(x_3, x_4)$,或 $e^{x_1} g^{x_2} \equiv e^{x_3} g^{x_4} (\bmod p)$,则有有效算法计算离散对数 $\log_e g$。

2. 在习题 1 中设 $p = 15083$, $q = 7541$, $e = 154$, $g = 2307$。若给了压缩函数 $h(x_1, x_2)$的一对碰撞消息$(7431, 5564)$和$(1459, 954)$,即有 $154^{7431} \cdot 2307^{5564} = 154^{1459} \cdot 2307^{954} \bmod 15083$,计算离散对数 $\log_{154} 2307$。

3. 设 p, $p_1(p=2p_1+1)$, q, $q_1(q=2q_1+1)$都是大素数,设 $n = pq$,且分解 n 是困难的,设 $\alpha \in Z_n^*$, α 的阶为 $2p_1q_1$(Z_n^* 中阶最大的元素),定义压缩函数 $h(x) = \alpha^x \bmod n$, $x \in 1, \cdots, n^2$,今设 $n = 603241$, $\alpha = 11$,若给了 $h(x)$的 3 个碰撞消息 $h(1294755) = h(80115359) = h(52738737)$,试利用这个消息分解 603241。

4. 设 $h_1: 0,1^{2m} \to 0,1^{m}$ 为一压缩函数,定义压缩函数 $h_2: 0,1^{4m} \to 0,1^{m}$ 如下:记 $x = x_1 x_2$ 为 x_1, x_2 的连接,其中 $x_1, x_2 \in 0,1^{2m}$,定义 $h_2(x) = h_1(h_1(x_1) h_1(x_2))$,其中 $h_1(x_1) h_1(x_2)$为 $h_1(x_1)$, $h_1(x_2)$的连接,证明:若找到一对 h_2 的碰撞消息 $h_2(x) = h_2(x')$,则利用这个消息容易找到 h_1 的一对碰撞消息 $h_1(y) = h_1(y')$。

5. 计算用 11.2.3 节 1. 中给出的迭代公式由 RSA 函数 $f_{(N,e)}$所构造的 Hash 函数 $h(x)$,设 $N = PQ = 1943(P=29, Q=67)$, $e = 701$, $h_0 = 1024$,设 $x = x_1 x_2 x_3 x_4 x_5$, $x_i \in D' = y; y$ 与 1943 互素 , $h_i = (x_i + h_{i-1})^{701} \bmod 1943, 1 \le i \le 5$, $h(x) = h_5$,每个同学独立地随机选取 10 个消息 x,计算出 $h(x)$。看班上是否找到 h 的一对碰撞消息?

6. 利用解背包(子集和)问题的困难性构造一个 Hash 函数。

7. 任选一 9.4 节中介绍的纠错码理论中的 NPC 问题构造一个 Hash 函数。

8. 用 SHS 的数组扩展公式(修改前)将每个数组 $X[16], \cdots, X[79]$写成 $X[0], \cdots, X[15]$的表达式。记 $\lambda_{ij} = | k; X[i]$和 $X[j]$同时出现在 $X[k]$的表达式中$, 16 \le k \le 79 |, 1 \le i < j \le 15$。用一个计算程序计算所有的 $\lambda_{ij}, 1 \le i < j \le 15$,并给出 λ_{ij}取值的分布(取每个值的 λ_{ij}个数)。

9. 若用生日攻击来攻击第 5 题中计算的 Hash 函数 $h(x)$,则大约需要计算多少个消息 x 的杂凑值才能使成功(找到一对碰撞消息)的概率达到 0.5。

第 12 章　身份识别方案

加密方案和单向函数被广泛地应用于密码保护方案和访问控制列表中,访问控制列表用于访问的控制,它能根据人们的身份识别来允许或拒绝他们访问特定的资源。自行识别是一种重要的安全服务,它在许多应用中都需要使用,访问自动出纳机、登录计算机、识别入网的蜂窝电话用户等等都要用到识别方案。注意识别(identification)和身份验证(authentication)是不同的。当说到身份验证时,通常存在一些承载信息的消息在通信双方之间交换,其通信的一方或双方需要被验证,识别(有时称为实体验证)是对一个用户身份的实时验证,它不需要交换承载信息的消息。

弱识别是基于非时变的口令或密码。A 先输入他的秘密口令,然后计算机确认他的正确性。A 和计算机都知道这个秘密口令,A 每次登录时,计算机都要求 A 输入秘密口令。人们注意到计算机无需知道秘密口令,它有能力区别有效口令和无效口令,这种办法是通过用单向函数(参见第 5 章)来实现的。计算机存储口令的单向函数值而不是存储秘密口令。其认证过程是:

(1) A 将他的秘密口令传送给计算机;

(2) 计算机完成口令的单向函数值计算;

(3) 计算机把单向函数值和机器存储的值进行比较。

由于计算机不再存储每个人的有效秘密口令表,某些人入侵计算机也无法从秘密口令的单向函数值表中获得秘密口令,但这种保护也是脆弱的,它抵抗不住字典式攻击。所谓字典式攻击是指用单向函数对常用的口令运算,将运算结果存储起来,并与计算机中存储的单向函数表对照,找出匹配的口令。为了抵抗字典式攻击,人们建议采用 salt 方法,即将一个随机串 salt 与秘密口令连接在一起,再用单向函数对其进行运算,然后将 salt 值和单向函数值存入数据库中,但这样做仍有严重的安全问题。首先,当 A 将他的口令输入系统时,能够访问他的数据库通道的任何人都可以读取他的口令。其次在系统对秘密口令加密前,能访问计算机系统的存储器的任何人都可以看到秘密口令。

目前在实际中所使用的一些别的技术,诸如 ID 卡(身份证)、信用卡、个人识别号(PIN)等都存在一些安全问题。这并不是说我们不能解决识别技术的安全问题,利用密码技术我们就可以设计出安全的识别协议(方案)。

一个安全的识别协议至少应该满足以下两个条件:

(1) 证明者 A 能向验证者 B 证明他的确是 A;

(2) 在证明者 A 向验证者 B 证明他的身份后,验证者 B 没有获得任何有用的信息,B 不能模仿 A 向第三方证明他是 A。

这两个条件是说证明者 A 能向验证者 B 电子地证明他的身份,而又没有向 B 泄露他的识别信息。目前已设计出了许多满足这两个条件的识别协议。

从实用角度讲,我们最关心的是设计简单的而且能在一个 Smart 卡上实现的安全识别方案。Smart 卡本质上是一个装有一个能完成算术运算的芯片的信用卡(Credit Card),因此,需求的计算量和存储量都应该保持尽可能的小。用这样的一个卡代替目前使用的 ATM 卡将是更安全的。但是,因为只用卡证明他的身份,我们对一个丢失的卡没有额外的保护,所以我们必须用额外的安全性来监控卡。为了确保只有卡的真正拥有者才能触发识别协议,拥有一个个人识别号(PIN)仍然是必要的。

本章将介绍一些比较流行的、有代表性的识别协议。这里先介绍一个在无线网络中通常使用的询问—响应识别(challenge-response identification)或强识别方案。

这里,A 通过表明他知道秘密进而向 B 证明他的身份,而不是通过提供该秘密。为了实现这个目的,使用了一个称为‘暂时号’的量。一个暂时号在同一个目的中只使用一次,它可以消除重发攻击。在实际中,随机号码、时戳、序列号等都可用作暂时号。

一个询问—响应协议的例子列举如下:

(1) A 通过密码和用户名向 B 登记;

(2) B 发给 A 一个随机号码(询问);

(3) A 用一个随机号码的加密值答复,其中使用了 A 的密码作为密钥完成加密(响应);

(4) B 证明 A 确实拥有密钥(密码)。

窃听者 Oscar 不能重发这个响应,因为当他试图联系 B 时,询问已经不相同了。Oscar 也无法确定密码,因为加密方法相当强健,密码决不会泄漏。

上述协议的安全性是建立在 A 和 B 相互信任的基础之上。然而,在大量的应用场合往往他们不一定相互信任,甚至可能是敌对的,所以更有用的方案应该是不需要共享秘密,这一观点将贯彻到本章所介绍的其余方案中。但有一点是共同的,识别方案本质上都是询问—响应协议。

12.1 Schnorr 身份识别方案

Schnorr 在 1991 年提出了一种计算量、通信量均少,且特别适用于智能卡(Smart Card)上的用户识别方案,Schnorr 识别协议融合了 ELGamal 协议、Fiat-Shamir 协议和 Chaum - Evertse - Van de Graff 交互式协议等协议的思想,其安全性建立在计算离散对数

问题的困难性之上。该协议是最有吸引力的实用识别协议之一,在许多国家都申请了专利。

Schnorr 协议需要一个可信中心,记为 TA。TA 将为协议选择下列一些参数:

(1) p 及 q 是两个大素数,且 $q \mid (p-1)$。q 至少 140 位,而 p 至少 512 位;

(2) $\alpha \in Z_p^*$ 为 q 阶元(诸如可取 $\alpha = g^{(p-1)/q}$, g 为 Z_p 的本原元);

(3) h 是一输出为 t 位的单向函数, t 为一个安全参数;

(4) 公开密钥 v 和秘密密钥 s,用作签名。

p, q, h 及其公开密钥都公布。每位用户自己选定个人秘密密钥 $s \in [1, q-1]$,且计算公开密钥 $v = \alpha^{-s} \bmod p$。

每位用户都必需到 TA 注册其公开密钥。TA 验明用户身份后,对每位用户指定一识别名 I。I 中包括用户的姓名、性别、生日、职业、电话号码、指纹信息、DNA 码等识别信息。TA 再对 $h(I, v)$ 予以签名,以便将来验证之用。

证明者 A 向验证者 B 证明他身份的协议(Schnorr 识别协议)可描述为:

(1) 用户 A 将其身份名 I 及公开密钥送交验证者 B。验证者根据 TA 的数字签名来验证用户 A 的公开密钥;

(2) 用户 A 任选一整数 r, $1 \leq r \leq q-1$,计算 $X = \alpha^r \bmod p$,并将 X 送给验证者 B;

(3) 验证者 B 任选一整数 $e \in [1, 2^t]$,送给用户 A;

(4) 用户 A 送给验证者 B: $y = r + se \bmod q$;

(5) 验证者 B 验证: $X = \alpha^y \times v^e \bmod p$。

现在,我们对 Schnorr 协议作一些解释。第(2)步可进行预处理,即在 B 出现以前完成。t 是安全参数,目的是阻止冒充者 Oscar 伪装成 A 猜测 B 的口令 e。因为如果 Oscar 能事先猜测到 e 的正确值,那么他能选择任何 y,并计算 $X = \alpha^y \times v^e \bmod p$, Oscar 将在第(2)步把 X 发送给 B,当他收到 B 发送来的口令 e 时,他将他已选好的值 y 提供给 B,那么在第(5)步中 B 能验证 X。如果 e 由 B 随机选择,那么 Oscar 将能正确地猜测 e 的值的概率是 2^{-t}。这样对大部分应用来说, $t = 40$ 将是合理的,为了更高的安全性, Schorr 建议使用 $t = 72$。

细心的同学会注意到,对于用户 A 的私钥和公钥对 (s, v), s 是仅 A 自己知道的秘密, s 功能上类似于 PIN,它使 B 相信完成识别协议的人的确是 A。但它与 PIN 有着本质的区别:在识别协议中, s 的值一直没有泄露,而是 A(更精确地说是 A 的智能卡)向 B 证明他知道 s 的值。这一证明过程在接下来的识别方案的第(4)步完成,它通过 A 计算值 y 回答 B 选取的口令 e 来完成。因为 s 的值没有被泄露过,所以这种技术称作知识的证明(Proof of Knowledge)。

安全性分析:

若双方都是诚实者,则因为 $v = \alpha^{-s} \bmod p$,及 $y = r + se \bmod q$,所以在第(5)步中满足 $\alpha^y \times v^e = \alpha^{r+se} \times \alpha^{-se} \bmod p = \alpha^r \bmod p = X$。

从 v 求 s 涉及到求离散对数问题,我们已假定在 Z_p 上计算离散对数是不可行的。鉴于 r 和 s 都是用户的秘密,所以假冒者 Oscar 无法从 y 中推出用户的秘密。

我们以后会看到 Schnorr 这类协议本质上是一种不需要双方事先分享秘密的交互式用户身份证明方法。不过必须存在一所谓的密钥认证中心 TA(Trusted Authority)负责管理一切用户,每位用户都必须先到 TA 注册,唯有注册完成的合法用户,才能使用这些方法进行认证,在认证过程中,TA 是不需介入的。

针对一般交互式用户身份证明协议,都必须满足以下 3 种性质。

(1) 完全性(Completeness):若用户与验证者双方都是诚实地执行协议,则有非常大的概率(趋近于 1),验证者将接受用户的身份。

(2) 健全性或合理性(Soundness):若用户根本不知道与用户名字相关的密钥,且验证者是诚实的,则有非常大的概率,验证者将拒绝接受用户的身份。

(3) 隐藏性(Witness hiding):若用户是诚实的,则不论协议进行了多少次,以及不论任何人(包括验证者)都无法从协议中推出用户的密钥,并且无法冒充用户的身份。

易知,Schorr 识别协议满足完全性。假如 Oscar 不知道 A 的秘密指数 s,那么他成功地执行识别协议的概率可忽略,这说明 Schnorr 识别协议满足合理性。

值得提醒的是,一个满足完全性和健全性的协议并不能保证协议是安全的。例如,A 可以通过简单地泄露他的指数 s 的值来向 Oscar 证明他的身份,该协议仍然是完全的和健全的,然而,这个协议是完全不安全的,因为 Oscar 能模仿 A。在密码学中,我们希望一个协议能够在 A 向 Oscar 证明他的身份的时候没有泄露任何信息,这就是第 15 章要介绍的零知识思想。到目前为止,仍然没有证明 Schnorr 协议是安全的。不过,它的一个修改,即 12.2 节将要介绍的 Okamoto 协议可证明是安全的。

实现方面

Schnorr 协议从计算量和需要交换的信息量两方面来看都是很快的和有效的。它也极小化了由 A 所完成的计算量,这是因为在许多实际应用中,A 的计算将由一个低计算能力的 Smart 卡来完成,而 B 的计算将由一个具有较强计算能力的计算机来完成。

对 A 而言,最耗时的计算是发生在第(2)步(需要完成一个模指数运算),但可由用户预先离线计算,因此这个协议特别适用于对智能卡的持有人做身份证明。

12.2 Okamoto 身份识别方案

Okamoto 识别协议是 Schnorr 协议的一种改进,这种改进使得在假定 Z_p 上计算一个特定的离散对数是难解的情况下,可证明其安全性。但仅从速度和有效性来讲,Schnorr 协议比 Okamoto 协议更实用。

为了建立方案,TA 像在 Schnorr 方案中一样,选择两个大素数 p 和 q,TA 也在 Z_p 中

选择两个阶为 q 的元素 α_1, α_2, 对系统的所有参加者包括 A, TA 保密 $c = \log_{\alpha_1} \alpha_2$。我们假定任何人(即使 A 与 Oscar 合伙)计算值 c 是不可行的。TA 选择一个签名方案和一个 Hash 函数。

TA 向 A 颁布一个证书的协议为:

(1) TA 建立 A 的身份并颁布一个识别串 ID(A);

(2) A 秘密地选择两个随机指数 m_1, m_2, $0 \leq m_1, m_2 \leq q-1$, 并计算 $v = \alpha_1^{-m_1} \alpha_2^{-m_2} \bmod p$, 将 v 发送给 TA;

(3) TA 对(ID, v)签名, $s = \mathrm{Sig}_{TA}(\mathrm{ID}, v)$。TA 将证书 C(A) = (ID(A), v, s)发送给 A。

Okamoto 识别协议为:

(1) A 随机选择两个数 r_1, r_2, $0 \leq r_1, r_2 \leq q-1$, 并计算 $X = \alpha_1^{r_1} \alpha_2^{r_2} \bmod p$;

(2) A 将他的证书 C(A) = (ID(A), v, s)和 X 发送给 B;

(3) B 通过检测 $\mathrm{Ver}_{TA}(\mathrm{ID}, v, s) = \mathrm{TURE}$ 来验证 TA 的签名;

(4) B 随机选择一个数 e, $1 \leq e \leq 2^t$, t 为安全参数并将 e 发送给 A;

(5) A 计算 $y_1 = (r_1 + m_1 e) \bmod q$, $y_2 = (r_2 + m_2 e) \bmod q$, 并将 y_1, y_2 发给 B;

(6) B 验证 $X = \alpha_1^{y_1} \alpha_2^{y_2} v^e \bmod p$。

易知, 如果 A 遵循协议, 那么 B 将接收 A 的身份证明。这表明 Okamoto 协议满足完全性。类似于 Schnorr 方案可以判断 Okamoto 方案是健全的(证明过程略)。

Okamoto 协议和 Schnorr 协议的主要差别在于: 在假定计算离散对数 $c = \log_{\alpha_1} \alpha_2$ 是不可行的情况下, 能证明 Okamoto 方案是安全的。其安全性证明的基本思路是: A 通过执行协议多项式次向 Oscar 识别自己。假定 Oscar 能获得 A 的秘密指数 α_1, α_2 的某些信息, 我们将证明 A 和 Oscar 一起能以很高的概率在多项式时间内计算离散对数 c, 这和我们的假设矛盾。这样我们就证明了 Oscar 通过参加协议一定不能获得关于 A 的指数的任何信息。

12.3　Guillou-Quisquater 身份识别方案

Guillou-Quisquater 识别协议的安全性是基于 RSA 体制的安全性。可证明识别协议满足完全性和健全性, 但即使在假定 RSA 体制安全的情况下, 也没有被证明该协议是安全的。

协议的建立过程如下:

TA 选择两个大素数 p 和 q, 形成 $n = pq$。保密 p 和 q, 公开 n。TA 选择一个大素数 b(用作一个安全参数)和一个公开的 RSA 加密指数。一般假定 b 是一个 40 比特长的素

数。TA 选择一个签名方案和 Hash 函数。

TA 向证明者 A 颁发一个证书的协议如下:

(1) TA 建立 A 的身份并颁布一个识别串 ID(A);

(2) A 秘密地选择一个整数 $m, 0 \leq m \leq n-1$。A 计算 $v = (m^{-1})^b \bmod n$, 并将 v 发送给 TA;

(3) TA 对(ID, v)签名, $s = \mathrm{Sig}_{TA}(ID, v)$。TA 将证书 C(A) = (ID(A), v, s)发送给 A。

证明者 A 向验证者 B 证明他的身份的协议,即 Guillou-Quisquater 识别协议为:

(1) A 随机选择一个整数 $r, 0 \leq r \leq n-1$ 并计算 $X = r^b \bmod n$;

(2) A 将他的证书 C(A) = (ID(A), v, s)和 X 发送给 B;

(3) B 通过检测 $\mathrm{Ver}_{TA}(ID, v, s)$ = TURE 来验证 TA 的签名;

(4) B 随机选择一个数 $e, 0 \leq e \leq b-1$, 并将 e 发送给 A;

(5) A 计算 $y = rm^e \bmod n$ 并将 y 发送给 B;

(6) B 验证 $X = v^e y^b \bmod n$。

容易看出, Guillou-Quisquater 识别协议满足完全性和健全性。

12.4 基于身份的身份识别方案

12.4.1 Shamir 的基于身份的密码方案的基本思想

Shamir 在 1984 年提出了一类新型的密码方案,这类方案能使网上的任何一对用户无需交换秘密密钥或公开密钥、无需保存密钥簿、无需使用第三方服务可进行安全的通信和相互验证签名。在这类方案中,假定了存在一个可信的密钥产生中心,该中心的主要作用是给每一个第一次入网的用户颁发一个个人化的 Smart 卡。嵌入在这个卡中的信息能使用户签名和加密他所发送的消息,并且能使用户用一个完全独立的方式解密和验证他所收到的消息。

Shamir 的基于身份的密码方案仍然是一类公钥密码方案,但它不是去直接生成一对随机的公开密钥和秘密密钥,而是由用户选择他的名字和网络地址来作为公开密钥。当用户入网时,密钥产生中心首先对用户进行识别。如果接纳用户,密钥产生中心就为该用户以 Smart 卡的形式颁布一个秘密密钥。该卡包含一个微处理器、一个 I/O 端口、一个 RAM、一个带秘密密钥的 ROM 以及加密/解密消息和产生/验证签名的程序。于是,当一个网上用户想与另一个用户通信时,他只需知道对方的姓名和地址就行了。这个方案与我们目前使用的邮政系统十分类似。

当用户 A 想给用户 B 发送一条消息 m 时,他就用自己的 Smart 卡中的秘密密钥对

消息 m 签名,用 B 的名字和网络地址来加密签名和消息 m,然后将密文连同 A 的名字和地址发送给 B。当 B 收到消息时,他使用他的 Smart 卡中的秘密密钥对消息进行解密,最后用发送者的名字和网络地址作为验证密钥验证签名。

基于身份的密码方案的安全性主要依赖于以下几个方面:

(1) 所使用的密码变换(诸如加密变换、签名变换等)的安全性;

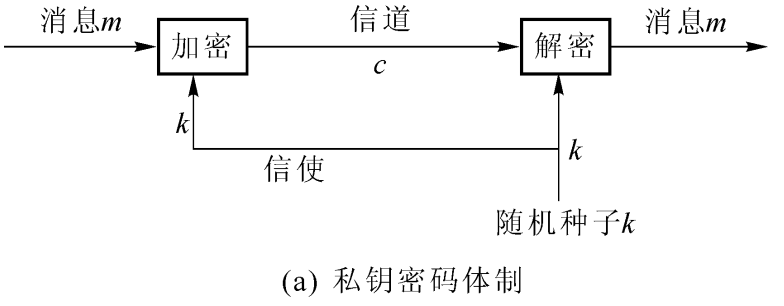
(2) 存储在密钥产生中心的特权信息的保密性;

(3) 在密钥产生中心给用户颁布 Smart 卡之前所完成的识别检测的严格性(要求用户提供的身份确实能惟一确定用户,而且用户不能否认);

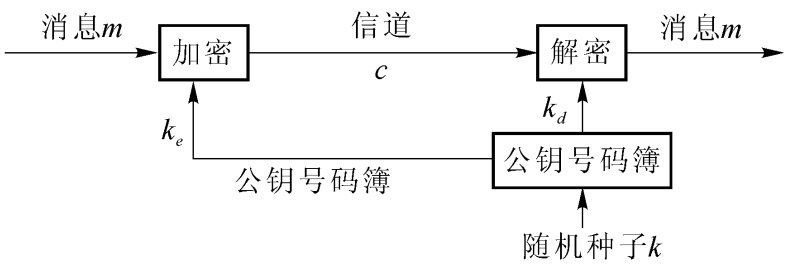
(4) 为了阻止用户的 Smart 卡的丢失、复制或未授权的使用,用户所采取的措施。

在基于身份的密码方案中,值得注意的是用户的秘密密钥必须由密钥产生中心来生成,不能由用户自己来生成,否则这个方案将是不安全的。

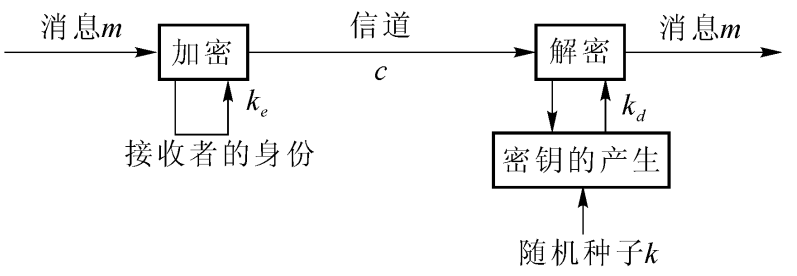
图 12.1 中给出了私钥密码体制、公钥密码体制和基于身份的密码体制三者之间的差别。



(a) 私钥密码体制



(b) 公钥密码体制



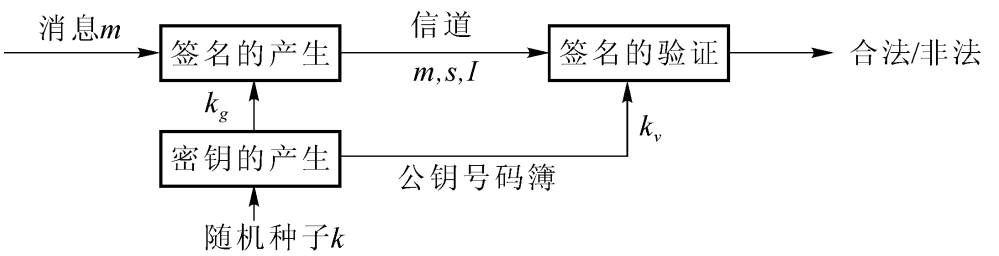
(c) 基于身份的密码体制

图 12.1　三种密码体制之间的差别

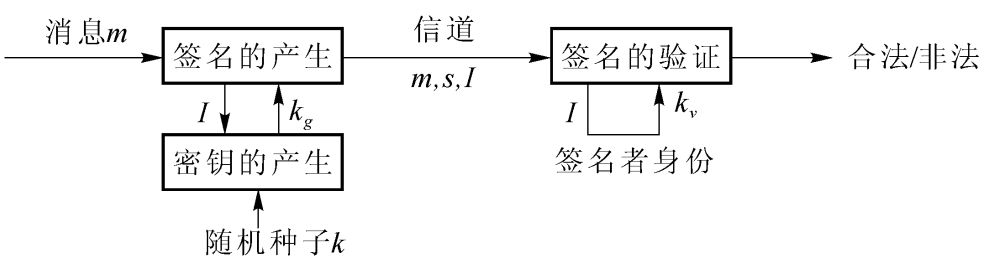
在这三种密码体制中,消息 m 用密钥 k_e 加密,密文 c 通过信道传送,并用密钥 k_d 解密,密钥的选择都是基于真正的随机种子 k。在私钥密码体制中,$k_e = k_d = k$ 密钥的保密和认证都需要使用单独的密钥信道(通常是一个信使(Courier));在公开密码体制中,加密

密钥和解密密钥使用两个不同的函数来生成,即 $k_e = f_e(k)$, $k_d = f_d(k)$。在进行密码认证时,需要有一个单独的信道(通常是一个簿子(Directory));在基于身份的密码体制中,加密密钥就是用户的身份,$k_e = I$,解密密钥由 I 和 k 导出,$k_d = f(I, k)$,这样,两个用户之间完全不需要独立的密钥信道。

图 12.2 给出了公钥签名方案和基于身份的签名方案之间的差别。用签名密钥 k_g 来对消息 m 签名,并将 m 及其签名 s 和签名者的身份 I 一起发送给验证者,验证者用验证密钥 k_v 来验证签名。



(a) 公钥签名方案



(b) 基于身份的签名方案

图 12.2 两种签名方案之间的差别

要实现 shamir 的基于身份的密码方案的思想,我们需要一个具有下列两个附加条件的公钥密码体制:

(1) 当知道种子 k 时,秘密密钥能从公钥中很容易地求出;

(2) 从一对特定的公开密钥和秘密密钥求出它们的种子 k 是困难的。

自从 Shamir 的基于身份的密码方案这一思想提出以来,人们围绕该话题进行了大量的研究,基于这一思想设计出的密码方案有很多。本小节将介绍 Guillou-Quisquater 的基于身份的识别协议。

12.4.2 Guillou-Quisquater 的基于身份的识别协议

Guillou-Quisquater 的基于身份的识别协议是由 Guillou-Quisquater 协议转化而来的。在这个方案中,TA 为用户 A 颁布一个数 u,该数是 A 的身份 ID 串的一个函数,而无需为用户 A 颁布一个证书。参数的选择与 Guillou-Quisquater 协议中一样,这里 $ab \equiv 1 \pmod{\phi(n)}$。

TA 为 A 颁布一个值 u 的过程如下:

(1) TA 建立 A 的身份并颁布一个识别串 ID(A);

(2) TA 计算 $u = (H(\mathrm{ID}(A))^{-1})^{a} \bmod n$, 并将 u 发给 A。

其中 $H(\quad)$是一个公开的 Hash 函数。

Guillou-Quisquater 的基于身份的识别协议如下:

(1) A 随机选择一个整数 $r, 0 \leq r \leq n-1$, 并计算 $X = r^{b} \bmod n$;

(2) A 把 ID(A)和 X 发送给 B;

(3) B 计算 $v = H(\mathrm{ID}(\mathrm{A}))$;

(4) B 随机选择一个数 $e, 0 \leq e \leq b-1$, 并将 e 发送给 A;

(5) A 计算 $y = ru^{e} \bmod n$, 并将 y 发送给 B;

(6) B 验证 $X = v^{e} y^{b} \bmod n$。

12.5　转换身份识别为签名方案

有一个标准的方法可将一个识别协议转化为一个签名方案。基本的观点是用一个公开的 Hash 函数来代替验证者 B。可见, 每一个识别协议都可派生出一个数字签名方案。下面以前面的身份识别协议为例说明如何将一个识别协议转化为一个签名方案。

1. Schnorr 签名方案(参看 10.1.7 节)

设 p 及 q 是两个大素数, 且 $q \mid (p-1)$, 在 Z_p 上离散对数问题是难处理的。设 $\alpha \in Z_p^*$ 是一个阶为 q 的元素。H 是一个 Hash 函数。

$$P = Z_p^*,\ A = Z_p^* \times Z_q,\ K = \{(p, q, \alpha, s, v) \mid v \equiv \alpha^{-s} \bmod p\}。$$

值 p, q, α, v 是公开的, s 是保密的。对 $k = (p, q, \alpha, s, v)$ 和一个(秘密的)随机数 $r \in Z_q^*$, 定义对消息 m 的签名

$$\mathrm{Sig}_{k_2}(m, r) = (X, y), \text{其中 } X = \alpha^{r} \bmod p,\ y = (r + sH(X, m)) \bmod q。$$

对 $m, X \in Z_p^*$ 和 $y \in Z_q$, 定义 $\mathrm{Ver}(m, X, y) = \mathrm{TRUE} \quad X = \alpha^{y} \times v^{H(X, m)} \bmod p$。

2. Okamoto 签名方案

系统的参数为 p, q, α_1, α_2 和 $t, q \mid (p-1)$, p 及 q 是两个大素数, p 至少为 512 比特长, q 至少为 140 比特长, t 是一个安全参数, t 至少为 20, α_1, α_2 都是 Z_p 中阶为 q 的元素。签名者 A 的公开密钥为 $v = \alpha_1^{-m_1} \alpha_2^{-m_2} \bmod p$, 秘密密钥是一对参数 $m_1, m_2, 0 \leq m_1, m_2 \leq q-1$。$H$ 是一个 Hash 函数。

当 A 对消息 m 签名时, 首先随机选择两个数 $r_1, r_2, 0 \leq r_1, r_2 \leq q-1$, 然后计算:

$$X = \alpha_1^{r_1} \alpha_2^{r_1} \bmod p,\ e = H(\mathrm{X}, \mathrm{m}),$$

$$y_1 = (r_1 + m_1 e) \bmod q,\ y_2 = (r_2 + m_2 e) \bmod q,$$

A 对消息 m 的签名是三元组(e, y_1, y_2)。

当接收者 B 收到签名 (e, y_1, y_2) 时，B 计算 $X \equiv \alpha_1^{y_1} \alpha_2^{y_2} v^e \bmod p$，并验证 $e = H(X, m)$。

3. Guillou-Quisquater 签名方案

签名者 A 选择两个大素数 p 和 q，形成 $n = pq$。A 选择一个大素数 b 和一个 Hash 函数 H。A 秘密地选择一个整数 u，$\gcd(u, n) = 1$，将 u 作为他的秘密密钥。A 的公开密钥为 $v = (u^{-1})^b \bmod n$。公开 n, b, v, H，保密 p, q, u。

A 对消息 m 的签名过程为：

(1) 随机选择一个整数 r，$0 \leq r \leq n - 1$ 并计算 $X = r^b \bmod n$；

(2) 计算 $e = H(X, m)$；

(3) 计算 $y = ru^e \bmod n$，A 对消息 m 的签名是 (e, y)。

接收者 B 验证签名的过程为：

(1) 获得 A 的公钥 n, b, v；

(2) 计算 $X' = v^e y^b \bmod n$ 和 $e' = H(X', m)$；

(3) 验证是否有 $e = e'$，如果 $e = e'$，则 B 接受 A 的签名；否则，拒绝。

注 记

本章主要介绍了三类身份识别方案：Schnorr 身份识别方案、Okamoto 身份识别方案、Guillou-Quisquater 身份识别方案。这些方案都是较实用的识别方案。

在识别协议中，有两类协议是特别诱人的，一类是零知识识别协议，另一类是基于身份的识别协议。有关零知识识别协议的典型代表是 Feige-Fiat-Shamir 识别协议，参见文献[61]；有关基于身份的识别协议是 Shamir 首次提出的一种观点，参见文献[60]。

Brickell 及 McCurley 将 Schnorr 协议中的参数略做修改，发展出一种美国 Sandia 国家实验室所用的协议，该协议数学理论上较 Schnorr 协议更完整[62]。

习题十二

1. 识别和身份验证有什么区别。

2. 一个安全的识别协议应满足什么条件？

3. 假设在 Guillou-Quisquater 身份识别方案中，设 $n = 199543$，$b = 523$，$v = 146152$，试验证 $v^{456}101360^b \equiv v^{257}36056^b \pmod n$，并由此求解 m。

4. 一般是通过什么原理将一个识别协议转化为一个签名方案的？

5. 对于一般交互式用户身份证明协议它应满足什么性质？

6. 阅读有关文献，举一个实际电子商务中的身份识别的方案。

第 13 章　认证码

13.1　认证理论与认证码

在引论中已对消息认证方案作了直观描述,那里还指出消息认证方案的安全性与消息加密方案的安全性一样,有基于信息论(无条件)的和基于计算复杂性理论(有条件)的两种标准。按照 Stinson[15],从计算安全性观点研究的消息认证方案称为消息认证码(MAC),从信息论安全性观点研究的消息认证方案称为认证码。本章只介绍认证码。认证码(系统)的信息理论是 G.J.Simmons 在 1984 年首先系统发展的,他研究了入侵者欺骗成功概率的下界及设计达到或接近这一下界的认证码,这一理论是认证码的理论基础。认证码的功用是要保证接收者收到的消息是由法定的发送者所发送的原始消息,即使在信道中存在能伪造消息和窜改消息的入侵者的情况下,这一功能仍应保持,简单地说,就是要保证发送消息的完整性。认证系统分为有仲裁的认证系统和没有仲裁的认证系统两种模型。本章只考虑没有仲裁的认证系统,一个没有仲裁的认证系统模型如图 13.1 所示。认证码还分保密的和不保密的两种,不保密的认证码也称笛卡尔(Cartesian)码。本章只介绍不保密的认证码。

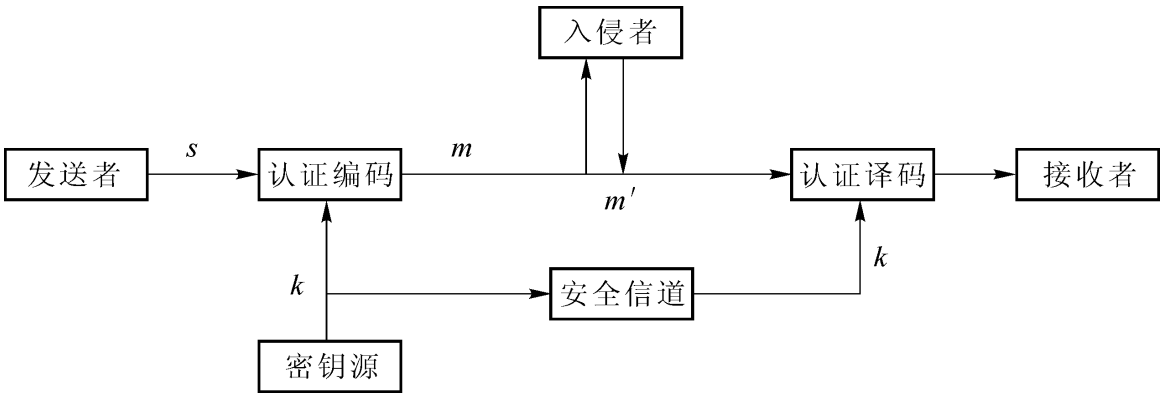


图 13.1　没有仲裁的认证系统模型

没有仲裁的不保密的认证系统的工作原理简述如下:发送者要发送信源状态 $s\in S$ (信源状态的有限集)给接收者,他首先将 s 输入认证编码器,认证编码由密钥 $k\in K$ (密

钥有限集)控制,认证编码器输出 $e_k(s)=a\in A$(认证标签有限集)。发送者将信源状态 s 加上认证标签 a 构成已签消息 $m=(s,a)\in S\times A=M$(已签消息有限集),然后将 m 在信道上发送到接收端,同时将密钥 k 通过安全信道发送到接收端。在公开(不安全)信道上,中间可能存在入侵者,他知道认证系统的全部知识,他惟一不知道的就是认证所用的密钥 k。入侵者可作两种攻击:(1)模仿攻击。入侵者在截得发送者发送的 $m=(s,a)$ 之前伪造消息 $m'=(s',a')$ 发送给接收者,希望接收者把 m' 当作发送者所发送的已签消息而加以接受。(2)代换攻击。入侵者在截得发送者发送的 $m=(s,a)$ 后,将 m 代换为 $m'=(s',a')$,其中 $s'\neq s$,将 m' 发送给接收者,同样希望接收者把 m' 当作发送者所发送的已签消息而加以接受。接收者收到 m 或 m' 后,先进行认证译码,即计算 $e_k(s)$ 或 $e_k(s')$,看它是否等于 a 或 a',若相等,则接受,否则就拒绝(不接受)。没有仲裁的认证系统模型与 Shannon 的保密系统模型很相似,只是把被动攻击的分析者换为主动攻击的入侵者。

下面给出认证码的正式定义。以下认证码总是指笛卡儿码。

定义 13.1 一个认证码是一个满足下列条件的四元组$(S,A,K,)$:

(1) S 是一个可能信源状态的有限集;

(2) A 是一个可能认证标签的有限集;

(3) K 是一个可能密钥的有限集,称为密钥空间;

(4) 对每个 $k\in K$,有一个认证编码规则 $e_k\in$,其中 $e_k: S\rightarrow A$ 为一映射。

定义 $M=S\times A$ 为已签消息集。

说明几点:一个信源状态类似于一个明文;一个明文附加一个认证标签为一个已签消息,简称消息;一个认证编码规则不必是 1-1 映射。

一个认证码$(S,A,K,)$可以用一个$|K|$行$|S|$列矩阵表示,称为认证矩阵。

定义 13.2 设 $(S,A,K,)$ 为一认证码,记 $S=s_1,s_2,\cdots,s_{|S|}$,$K=k_1,k_2,\cdots,k_{|K|}$。定义与认证码$(S,A,K,)$对应的认证矩阵为(a_{ij}),其中,$a_{ij}=e_{k_i}(s_j)$,$1\leq i\leq|K|$,$1\leq j\leq|S|$。

13.2 计算欺骗概率

定义 13.3 对入侵者所作的模仿攻击和代换攻击,定义相应的欺骗概率为入侵者采用最优策略的情况下欺骗成功的概率,分别记作 Pd_0 和 Pd_1。

为了计算 Pd_0 和 Pd_1,需要给定发送信源状态的概率分布和使用密钥的概率分布,分别记作 $p_S=(p_S(s);s\in S)$ 和 $p_K=(p_K(k);k\in K)$。下面先举一个例子。

例 13.1 设 $S=A=Z_3$(剩余类环),$K=Z_3\times Z_3$,对每一个$(i,j)\in K$ 和 $s\in S$,定义认证编码规则为 $e_{ij}(s)=is+j(\mathrm{mod}3)$,与例中的认证码$(S,A,K,)$对应的认证矩阵列表如下:

k \ s	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

设使用密钥的分布为等概分布，即 $p_K(k) = 1/9$, $k \in K$，在这个例子中，发送信源状态的分布是无关紧要的。

先计算 Pd_0。由认证矩阵表可见，对每一消息 $(s', a') \in M$，恰有 3 个密钥 $(i, j) \in K$，使 $e_{ij}(s') = a'$，因此 $Pd_0 = 3/9 = 1/3$。

再计算 Pd_1。由认证矩阵表可见，对每一消息 $(s, a) \in M$，恰有 3 个密钥 $(i, j) \in K$，使 $e_{ij}(s) = a$，故入侵者截得消息 (s, a) 后，知道所使用的密钥在这三个密钥之中，若入侵者将它代换为 (s', a')，$s' \neq s$，则 3 个密钥中仅有一个密钥 (i, j) 能满足 $e_{ij}(s') = a'$，因此入侵者欺骗成功的概率 $Pd_1 = 1/3$。

现在来推导计算 Pd_0 和 Pd_1 的一般公式。先考虑 Pd_0。设 k_0 为发送者和接收者使用的密钥，对 $s' \in S$, $a' \in A$，定义支付函数 payoff(s', a') 为接收者把 (s', a') 当作发送者发送的消息加以接受的概率，容易看出

$$\text{payoff}(s', a') = \Pr\left[e_{k_0}(s') = a'\right] = \sum_{k \in K;\, e_k(s') = a'} p_K(k) \tag{13.1}$$

由于入侵者要采用最优策略，即要选择支付函数最大的 (s', a')，因此

$$Pd_0 = \max\{\text{payoff}(s', a') : s' \in S, a' \in A\} \tag{13.2}$$

Pd_0 与发送信源状态的概率分布无关。

再考虑 Pd_1。Pd_1 较难计算，它可能与发送信源状态的概率分布有关。设入侵者在信道中截得消息 (s, a)，他要将 (s, a) 代换为 (s', a')，$s' \neq s$。现在定义支付函数 payoff$(s', a'; s, a)$ 为接受者把入侵者代换的消息 (s', a') 当作发送者发送的消息加以接受的概率。我们有下面的计算：

$$\text{payoff}(s', a'; s, a) = \Pr\left[e_{k_0}(s') = a' \,\middle|\, e_{k_0}(s) = a\right]$$

$$= \Pr\left[e_{k_0}(s') = a', e_{k_0}(s) = a\right] \Big/ \Pr\left[e_{k_0}(s) = a\right]$$

$$= \sum_{k\in K;\, e_k(s)=a,\, e_k(s')=a'} p_K(k) \Big/ \sum_{k\in K;\, e_k(s)=a} p_K(k)$$

$$= \sum_{k\in K;\, e_k(s)=a,\, e_k(s')=a'} p_K(k) \Big/ \text{payoff}(s, a)。 \tag{13.3}$$

由于入侵者要采用最优策略,他要计算

$$p_{s,a} = \max \text{ payoff } (s', a'; s, a) ; s' \in S, s' \neq s, a' \in A 。 \tag{13.4}$$

由于入侵者在信道中截得消息(s, a)等于发送者发送消息(s, a)的概率 $p_M(s, a)$,故

$$\text{Pd}_1 = \sum_{(s,a)\in M} p_M(s, a)\, p_{s,a} \tag{13.5}$$

其中

$$p_M(s, a) = p_S(s) \cdot \sum_{k\in K;\, e_k(s)=a} p_K(k)$$

$$= p_S(s) \cdot \text{payoff}(s, a)。 \tag{13.6}$$

13.3 组合界

从 13.2 节可见,认证码的安全性由欺骗概率 $P\text{d}_0$ 和 $P\text{d}_1$ 度量,因此我们要构造使 $P\text{d}_0$ 和 $P\text{d}_1$ 尽可能小的认证码。同时下面两个条件也很重要:(1) 信源状态数要足够多,使发送者可发送他所需要发送的所有信息。(2)密钥空间应尽可能小,因为密钥要通过昂贵的安全信道传送,但每发送一次消息,必须更换密钥以保证安全。

本节主要研究 Pd_0 和 Pd_1 可能达到的下界以及达到这些下界所要满足的条件。第 13.4 节再考虑认证码的构造问题。

定理 13.1 设$(S, A, K,)$为一认证码,则

$$P\text{d}_0 \geq 1/|A| \tag{13.7}$$

等号成立,当且仅当

$$\sum_{k\in K;\, e_k(s)=a} p_K(k) = 1/|A| \tag{13.8}$$

对一切 $s \in S, a \in A$ 成立。

证 对固定的 s',将(13.1) 式对一切 $a' \in A$ 相加得

$$\sum_{a\in A} \text{payoff}(s', a') = \sum_{k\in K} p_K(k) = 1。$$

因此,对每一 $s' \in S$,存在一个认证标签 $a'(s')$ 使

$$\text{payoff}(s', a'(s')) \geq 1/|A|,$$

于是由 Pd_0 的计算公式(13.2)证得(13.7)。

等号成立的充要条件是显然的。

定理 13.2　设$(S, A, K,)$为一认证码,则

$$\mathrm{Pd}_1 \geq 1/|A| \tag{13.9}$$

等号成立,当且仅当

$$\mathrm{payoff}(s', a'; s, a) = 1/|A| \tag{13.10}$$

对一切 $s, s' \in S, s' \neq s, a, a' \in A$ 成立。

证　对固定的 s, a 和 s',其中 $s' \neq s$,将(13.3)式对一切 $a' \in A$ 相加得

$$\sum_{a' \in A} \mathrm{payoff}(s', a'; s, a)$$

$$= \sum_{a' \in A,\, k \in K;\, e_k(s) = a,\, e_k(s') = a'} p_K(k) \Big/ \mathrm{payoff}(s, a) = 1。$$

因此,存在一个认证标签 $a'(s', s, a)$ 使

$$\mathrm{payoff}(s', a'(s', s, a); s, a) \geq 1/|A|。$$

于是由 Pd_1 的计算公式(13.4)和(13.5)证得(13.9)。

由(13.5)式可见,$\mathrm{Pd}_1 = 1/|A|$的充要条件是 $p_{s,a} = 1/|A|$对一切(s, a)成立,但这等价于条件(13.10)。

定理 13.3　设$(S, A, K,)$为一认证码,则,$\mathrm{Pd}_0 = \mathrm{Pd}_1 = 1/|A|$,当且仅当

$$\sum_{k \in K;\, e_k(s) = a,\, e_k(s') = a'} p_K(k) = 1/|A|^2 \tag{13.11}$$

对一切 $s, s' \in S, s' \neq s, a, a' \in A$ 成立。

证　显然,条件(13.11)成立的充要条件是条件(13.8)和条件(13.10)同时成立。

若使用密钥的概率分布为 K 上的等概分布,则可得到如下的系:

定理 13.4　设$(S, A, K,)$为一认证码,使用密钥的概率分布为 K 上的等概分布,则 $\mathrm{Pd}_0 = \mathrm{Pd}_1 = 1/|A|$,当且仅当

$$|\ k \in K;\, e_k(s) = a,\, e_k(s') = a'\ | = |K|/|A|^2$$

对一切 $s, s' \in S, s' \neq s, a, a' \in A$ 成立。

13.4　用正交矩阵构造认证码

一个正交阵列 **OA**(n, l, λ)是一个有 n 个不同符号的 $\lambda n^2 \times l$ 矩阵,且具有如下性质:在矩阵的任意两列中,每对不同符号恰好在矩阵的 λ 行中相遇,故也称正交矩阵。

正交阵列在组合设计中已有很好的研究,它可用来构造使 $\mathrm{Pd}_0 = \mathrm{Pd}_1 = 1/n$ 的认证码。

定理 13.5　若存在一个正交阵列 **OA**(n, l, λ),则可构造一个认证码$(S, A, K,)$,其中$|S| = l, |A| = n, |K| = \lambda n^2$,使 $Pd_0 = Pd_1 = 1/n$。

证　将正交阵列 **OA**(n, l, λ)定义的 $\lambda n^2 \times l$ 矩阵看作一个认证矩阵,构造这一认证

矩阵相对应的认证码$(S, A, K,)$,故有$|S| = l, |A| = n, |K| = \lambda n^2$。设使用密钥的概率分布为 K 上的等概分布。于是这一认证码满足系(13.4)的所有条件。因此有 $Pd_0 = Pd_1 = 1/n$ 达到组合界。

例 13.1 的表中列出的认证矩阵就是一个正交阵列 **OA**(3,3,1)。

由定理 13.5 可见,正交阵列 **OA**(n, l, λ)的参数与它所构造的认证码$(S, A, K,)$的安全性水平、信源状态数、认证标签数和密钥数有关。n 决定安全性水平 Pd_0 和 Pd_1,l 决定信源状态数,λ 则只与密钥数有关,因此,为了构造所需要的认证码,就要构造相应的 **OA**(n, l, λ)。按照第 13.3 节中提到的构造认证码的条件,若需要构造一个安全性水平为 α 的认证码(即 $Pd_0 \leq \alpha, Pd_1 \leq \alpha$),则相应的 **OA**$(n, l, \lambda)$应满足如下条件。

(1) $n \geq 1/\alpha$;

(2) $l \geq |S|$(由于从任一正交阵列中删去若干列所得的阵列仍是正交阵列,故不要求 $l = |S|$);

(3) 在满足上述两个条件的前提下,要使 λ 尽可能小。

为了读者选择适当的认证码的方便,下面不加证明地陈述若干关于构造正交阵列及其界的已知结果,它们的证明可参看本书参考文献[15]。

先考虑 $\lambda = 1$。对给定的 n,要构造 l 最大的正交阵列,下面是一个限制 l 的条件。

定理 13.6 设 **OA**$(n, l, 1)$存在,则 $l \leq n + 1$;另一方面,可以构造出无限多个正交阵列达到定理 13.6 的界。

定理 13.7 若 p 为素数,则正交阵列 **OA**$(p, p + 1, 1)$存在。

定理 13.6 告诉我们,若 $l > n + 1$,则 $\lambda > 1$。对给定 n 和 l 的正交阵列,下面给出一个限制 λ 的条件。

定理 13.8 设 **OA**(n, l, λ)存在,则

$$\lambda \geq l(n - 1) + 1 / n^2。$$

另一方面,可以构造出无限多个正交阵列达到定理 13.8 的界。

定理 13.9 设 p 为素数和 $d \geq 2$ 为整数,则正交阵列 **OA**$(p, (p^d - 1)/(p - 1), p^{d-2})$存在。

例 13.2 设 $p = 2, d = 3$,则可构造正交阵列 **OA**$(2, (2^3 - 1)/(2 - 1), 2^{3-2})$ = **OA**(2, 7, 2)如下:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}。$$

最后指出,本节介绍了用正交阵列方法构造安全的认证码。读者可能会问:是否有更好的方法呢?研究表明,若要构造欺骗概率尽可能小的认证码,则没有比正交阵列方法更好的方法了。

注 记

本章简短介绍了认证理论与认证码的最简单模型。本章主要参考了文献[15]。有关认证系统与认证码的一般理论可参看本书参考文献[26]第 2 章所引文献[9]～[36],若干具体的消息认证码(MAC)可在本书参考文献[28]中找到。

习题十三

1. 设认证码$(S, A, K,)$表示为下面的认证矩阵:

K \ S	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

设发送信源状态的概率分布为 $p_S(1) = p_S(4) = 1/6$, $p_S(2) = p_S(3) = 1/3$;使用密钥的概率分布为 $p_K(1) = p_K(6) = 1/4$, $p_K(2) = p_K(3) = p_K(4) = p_K(5) = 1/8$。

(1) 计算该认证码的欺骗概率 Pd_0 和 Pd_1。

(2) 给出与 Pd_0 和 Pd_1 相应的最优策略。

2. 证明定理 13.7,并构造 OA(5,6,1)。

提示　先证明 OA$(p, p, 1)$存在(p 为素数),然后证每一 OA$(p, p, 1)$可加上一列构成 OA$(p, p+1, 1)$。

3. 证明由符号集 $A = \{1, 2, \cdots, n_1\}$上的正交阵列 OA$(n_1, l, \lambda_1)$和符号集 $B = \{1, 2, \cdots, n_2\}$上的正交阵列 OA$(n_2, l, \lambda_2)$可构造一个在符号集 $C = A \times B$ 上的正交阵列 OA$(n_1 n_2, l, \lambda_1 \lambda_2)$。

提示　构造 $A \times B$ 上的 $n_1 n_2 \times l$ 矩阵,其行$(z_1, z_2, \cdots, z_l) = ((x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l))$,其中$(x_1, x_2, \cdots, x_l)$和$(y_1, y_2, \cdots, y_1)$分别为 OA$(n_1, l, \lambda_1)$和 OA$(n_2, l,$

λ_2)的行,再证明所构造的矩阵为 **OA**(n_1 n_2, l, λ_1 λ_2)。

4. 构造一个正交阵列 **OA**(3,13,3)。

5. 设(S, A, K,)为一认证码,其中$|A| = n$,且 $Pd_0 = Pd_1 = 1/n$,则$|K| \geq n^2$,等号成立的充要条件是存在一个正交矩阵 **OA**(n, l,1)使$|S| = l$,且使用密钥的概率分布 $p_K(k) = 1/n^2$, $k \in K$。

6. 设(S, A, K,)为一认证码,给定发送信源状态的概率分布 p_S 和使用密钥的概率分布 p_K 后,S 和 K 可看作概率空间或随机变量,其分布分别为 $p_S(s)$, $s \in S$ 和 $p_K(k)$, $k \in K$。设随机变量 S 和 K 是统计独立的,由于认证码可表示为认证矩阵,故 A 也可看作一个随机变量,其分布由 $p_S(s)$和 $p_K(k)$完全确定,因此已签消息 $M = (S, A)$也是一个随机变量,要证

$$\log Pd_0 \geq H(K|M) - H(K) = -I(K, M),$$

其中 $H(K)$和 $H(K|M)$为熵和条件熵(参看 3.2 节)。

提示　应用 Jensen 不等式和熵的性质。

第 14 章　密钥管理

14.1　密钥管理概述

在先前的章节,我们所关注的是采用何种密码算法来实现数据的机密性、完整性、认证性。然而,在一个安全系统中(比如电子商务系统),总体安全性依赖于许多不同的因素,例如算法的强度、密钥的大小、口令的选择、协议的完全性等,其中对密钥或口令的保护是尤其重要的。因为按 Kerckhoff 假设(见第 2 章),一个密码系统的安全性完全取决于对密钥的保密而与算法无关,即算法可以公开,而密钥则必须是保密的。简单地说,不管算法是多么强有力,一旦密钥丢失或出错,不但合法用户不能提取信息,而且可能会使非法用户窃取信息。可见,密钥的保密和安全管理在数据系统安全中是尤为重要的。

一个安全系统不仅要阻止入侵者获得密钥,还要避免对密钥的未授权的使用、有预谋的修改和其他形式的操作等。另外,人们还希望即使这些不利的情况发生了,系统应当能够察觉。密钥管理是处理密钥自产生到最终销毁的整个过程中的有关问题,大体上讲,密钥管理包括密钥的产生、装入、存储、备份、分配、更新、吊销和销毁等内容,其中分配和存储可能是最棘手的问题。密钥管理不仅影响系统的安全性,而且涉及到系统的可靠性、有效性和经济性,当然,密钥管理过程中也不可能避免物理上、人事上、规程上等一些问题,但我们主要从理论上讨论密钥管理的有关问题。

在分布式系统中,人们已经设计了用于自动密钥分配业务的几个方案。其中某些方案已被成功使用,如 Kerberos 和 ANSI X.9.17 方案采用了 DES,而 ISO-CCITT X.509 目录认证方案主要依赖于公钥密码。

本章主要讨论密钥的分配,密钥共享和密钥托管问题。在详细介绍它们之前,我们还是先简短地叙述上面提到的密钥管理涉及的方面。

1. 密钥的种类

密钥的种类繁多,但主要有如下几种:

(1) 基本密钥(Base Key),或称初始密钥(Primary Key)

此类密钥以 k_p 表示,是由用户选或由系统分配给用户的可在较长时间(相对于会话

密钥)内由一对用户所专用的秘密密钥,故又称用户密钥(User Key)。要求它既安全又易于更换,和会话密钥一起去启动和控制某种算法所构造的密钥产生器,来产生用于加密数据的密钥流,参见图 14.1。

(2) 会话密钥(Session Key)

两个通信终端用户在一次通话交换数据时所采用的密钥,以 k_s 表示。当用其对传输的数据进行保护时称其为数据加密密钥(Date Encrypting Key),当用其保护文件时称为文件密钥(File Key)。会话密钥的作用是使我们可以不必太频繁地更换基本密钥,有利于密钥的安全和方便管理。这类密钥可由用户双方预先约定,也可由系统动态地产生并赋予通信双方,它为通信双方专用,故又称为专用密钥(Private Key)。

(3) 密钥加密密钥(Key Encrypting Key)

这是对传送的会话或文件密钥进行加密时采用的密钥,也称次主密钥(Submaster Key)或辅助(二级)密钥(Secondary Key),以 k_e 表示。通信网中每个节点都分配有一个这类密钥。为了安全,各节点的密钥加密密钥应互不相同,在主机和主机之间以及主机和各终端之间传送会话密钥时都需要有相应的密钥加密密钥。每台主机都须存储有关到其他各主机和本主机范围内各终端所用的密钥加密密钥,而各终端只需要一个与其主机交换会话密钥时所需的密钥加密密钥,称之为终端主密钥(Terminal Master Key)。在主机和一些密码设备中,存储各种密钥的装置应有断电保护和防窜扰、防欺诈等控制功能。

(4) 主机主密钥(Host Master Key)

它是对密钥加密密钥进行加密的密钥,存于主机处理器中,以 k_m 表示。

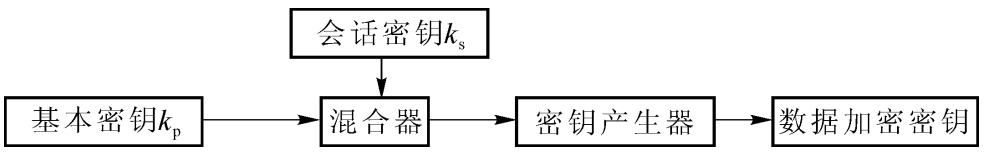


图 14.1　几种密钥之间的关系

除了上述几种密钥之外,还有用户选择密钥(Custom Option Key)(用来保证同一类密码机的不同用户可使用的不同的密钥)、族密钥(Family Key)及算法更换密钥(Algorithm Changing Key)等。这些密钥的主要作用是在不增大更换密钥的工作量的条件下,扩大可使用的密钥量。基本密钥一般通过面板开关或键盘选定,而用户选择密钥常要通过更改设备内部连线、改变插接组件或开关位置实现,这种改变常起到更改密钥产生算法的作用。例如,在非线性移存器密钥流产生器中,基本密钥和会话密钥用于确定寄存器的初态,而用户选择密钥可决定寄存器反馈线抽头的连接。

2. 密钥的生成

现代通信网需要产生大量的密钥分配给各主机、节点和用户。过去靠人工方法产生和管理密钥的方式已被时代所淘汰。密钥产生和管理的自动化,不仅可以减轻人们的工作负担,而且可以消除人为差错引起的泄密。目前已有各种各样的密钥产生器可为大型

系统提供所需的各类密钥。当然,生成密钥的算法要满足一定的要求才能启用。

主机主密钥通常要用诸如掷硬币、骰子,从随机数表中选数等随机方式产生,以保证密钥的随机性,避免可预测性。而任何机器和算法所产生的密钥都有被预测的危险。主机主密钥是控制产生其他加密密钥的密钥,而且长时间保持不变,因此它的安全性是至关重要的。

加密密钥可以由机器自动产生,也可以由密钥操作员选定。加密密钥构成的密钥表存储在主机中的辅助存储器中,只有密钥产生器才能对此表进行增加、修改、删除和更换密钥,其副本则以秘密方式送给相应的终端或主机。一个有 N 个终端用户的通信网,若要求任一对用户之间彼此能进行保密通信,则需要 C_N^2 个密钥加密密钥。当 N 较大时,难免有一个或数个被敌手掌握。因此密钥产生算法应当能够保证其他用户的密钥加密密钥仍有足够的安全性。可用随机比特产生器(如噪声二极管振荡器等)或伪随机数产生器生成这类密钥,也可用主密钥控制下的某种算法来产生。

会话密钥可在密钥加密密钥作用下通过某种加密算法动态地产生,如用初始密钥控制一非线性移存器或用密钥加密密钥控制 DES 算法产生。

初始密钥可用产生密钥加密密钥或主机主密钥的方法生成。

对许多密码体制,我们分析了成功攻击所花的代价,然而那是理论的情况,非常不严格,未考虑到协议、口令等相关的问题。在我们的经验中,尽管密钥的大小常常得到最大的重视,例如 AES 算法采用了比 DES 更长的密钥,但几乎所有的失败不是因为不合适的密钥大小,而是由于协议或口令的失败。如果用一个弱的密钥产生方法,那么整个系统将是弱的。因为能破译你的密钥产生算法,所以攻击者就不需要去破译你的加密算法了。最常见的攻击方法是字典攻击(Dictionary Attack)。所谓字典攻击,就是攻击者使用一本公用密钥字典搜索,字典中记录的是人们习惯用的英文名、简写字母、专有名词等。有人证实,25%以上的口令可由此而破。当字典攻击被用作破译密钥文件而不是单个密钥时,就显得更加有力。单个用户可以机灵地选择到好密钥,如果一千个人各自选择自己的密钥作为计算机系统的口令,那么至少有一个人将选择到攻击者字典中的词作为密钥。

关于好的密钥我们有一些建议:

(1) 真正随机、等概,如掷硬币、执骰子等;

(2) 避免使用特定算法的弱密钥;

(3) 采用密钥揉搓或杂凑技术,将易记的长句子(通行短语),经单向杂凑函数变换成伪随机数串。

3. 密钥的更换

密钥的使用时间越长,它泄露的机会就越大;而且一旦泄露,如果密钥使用越久,损失就越大。所以,密钥,特别是会话密钥的定期更换对系统的安全性非常必要。通过不断地更换密钥,可以挫败攻击者对密钥搜索,特别是穷举密钥搜索,因为等到敌手成功地搜索出密钥时,系统已经换用了另一个不同的密钥。

显然,密钥被更换的频率依赖于:

(1) 被保护的数据的敏感性:安全性要求越高,密钥更换得越频繁。

(2) 被保护数据的有效期:例如,密钥加密密钥就无需频繁更换,因为它们只是偶尔地用作密钥交换。

(3) 密码算法的力量:用 DES 算法的系统与用 AES 算法的系统相比,密钥应该更换得更频繁。

4. 密钥的存储

密钥存储时必须保证密钥的机密性、认证性和完整性,防止泄露和篡改。比较好的解决方案是:将密钥储存在磁条卡中,使用嵌入 ROM 芯片的塑料密钥或智能卡,通过计算机终端上特殊的读入装置把密钥输入到系统中。当用户使用这个密钥时,他并不知道它,也不能泄露它,他只能用这种方法使用它。若将密钥平分成两部分,一半存入终端,一半存入 ROM 钥卡上,即使丢掉了 ROM 钥卡也不致泄露密钥。将密钥做成物理形式会使存储和保护它更加直观。对于难以记忆的密钥可用加密形式存储,利用密钥加密密钥来加密。例如,RSA 的秘密钥可用 DES 加密后存入硬盘,用户须有 DES 密钥,运行解密程序才能将其恢复。理想的情况是密钥永远也不会以未加密的形式暴露在加密设备以外。

5. 密钥的销毁

不用的旧密钥必须销毁,否则可能造成危害,别人可用它来读原来曾用它加密的文件,且旧密钥有利于分析密码体制。要安全地销毁密钥,如采用高质量碎纸机处理记录密钥的纸张,使攻击者不可能通过收集旧纸片来寻求有关秘密信息。对于硬盘、EEPROM 中的存数,要进行多次冲写。

6. 密钥的吊销

如果密钥丢失或因其他原因在密钥未过期之前,需要将它从正常运行使用的集合中除去,称为密钥吊销。采用证书的公钥可通过吊销公钥证书实现对公钥的吊销。

14.2 密钥分配协议

密钥分配(Key Distribution)是这样的一种机制:系统中的一个成员先选择一个秘密密钥,然后将它传送给另一个成员。在下面的一些方案中,有一个可信方(Trusted Authority),简记为 TA,负责验证用户身份、选择和传送密钥给用户等事情。

先看看一个最简单的密钥预分配方案:设想一个有 n 个用户的不安全网络,当用户 U,V 想进行保密通信时,先向 TA 提出申请,TA 为他们随机选择一个会话密钥 K,并通过一个安全信道(密钥的传送可以不在网络上进行,因为网络是不安全的)传送给他们。这个方法是无条件安全的,但需要在 TA 和网络上的每个用户之间共享一个安全信道。

而且每个用户必须存储 $n-1$ 个密钥，TA 需要安全地传送 $n(n-1)$ 个密钥（有时称为 n^2 问题）。甚至对一个相当小的网络，也可能变得相当昂贵，所以不是一个实际的解决。显然，试图减少安全信道的数目和每个用户需要存储的密钥量是我们所关心的问题。

当考虑密钥分配所使用的密码技术时，通常区分非对称与对称算法。

1. 对称系统的密钥分配

对称算法的密钥分配起初总是涉及物理上分发一个密钥集。例如，网络安装时将需要一个安全官员装载初始密钥到所谓的防窜扰模型里，这种模型对储存在其中的密钥提供物理保护。初始分发之后，实际的密钥分配，即会话密钥的分配，可通过网络进行。密钥分配中心可完成两个功能。首先，例如在安装期间，密钥分配中心仅仅供给那些可以相互通信的双方密钥。这个阶段常常称为离线的密钥分配系统。

在一个在线的密钥分配系统中，每个用户都有一个惟一的、秘密的和密钥分配中心通信的密钥，当用户向中心请求会话密钥时，该密钥起作用。在这种情况下，每个用户只需储存非常少的密钥，而离线密钥分配系统，每个用户需要储存的密钥数目与用户的总数成正比。

如果使用在线密钥分配系统，那么每个用户和可信中心共享一个密钥而无需再存储别的密钥。会话密钥将通过请求可信中心来传送。确保密钥新鲜(Key Freshness)是可信中心的职责。因此，人们更喜欢使用在线方法分配密钥，也就是一对用户想通信时每次产生一个新的会话密钥。

下面介绍 Kerboros 协议。

Kerboros 协议是一个基于私钥密码体制的流行的密钥服务系统。在这个系统中，每个用户 U 和可信中心（对称密钥分发中心(KDC)）共享一个秘密的 DES 密钥。在协议(版本 v5)的最新版本中，传送的所有消息都通过 CBC 模式进行加密。

用 ID(U)表示用户 U 的某些识别信息。使用 Kerboros 协议传输一个会话密钥的过程可描述如下：

(1) 用户 U 为了和用户 V 通信，他向可信中心要一个会话密钥；

(2) 可信中心随机选择一个会话密钥 K，一个时戳 T 和一个生存期 L；

(3) 可信中心计算 $m_1 = E_{K_U}(K, \mathrm{ID}(V), T, L)$ 和 $m_2 = E_{K_V}(K, \mathrm{ID}(U), T, L)$，并将 m_1 和 m_2 发给 U；

(4) 用户 U 首先解密 m_1 获得 K, ID(V), T 和 L。然后计算 $m_3 = E_K(\mathrm{ID}(U), T)$ 并将 m_3 和可信中心发送来的 m_2 一起发给 V；

(5) 用户 V 首先解密 m_2 获得 K, ID(U), T 和 L 然后使用 K 解密 m_3 获得 T 和 ID(U)并检测 T 的两个值和 ID(U)的两个值是否一样。如果是一样的，那么 V 计算 $m_4 = E_K(T+1)$，并将 m_4 发送给 U；

(6) U 使用 K 解密 m_4 获得 $T+1$ 并验证解密结果是 $T+1$。

传输在协议中的信息如图 14.2 所示。

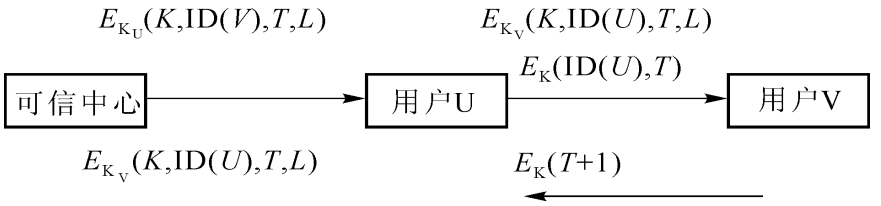


图 14.2　协议流程图

现在我们对协议中的各步骤作一些解释。在第(2)步,可信中心产生 T, K, L。在第(3)步,可信中心使用它与用户 U 共享的密钥 K_U 加密 T, K, L 和 ID(V)形成 m_1。使用它与用户 V 共享的密钥 K_V 加密 T, K, L 和 ID(U)形成 m_2。将这些加密消息发送给用户 U。U 能使用他的密钥解密 m_1,获得 T, K, L 和 ID(V)。他将验证目前的时间是在区间 T, $T+L$ 内。他通过验证解密获得的 ID(V)检测可信中心颁布给他的密钥 K 是他和用户 V 的会话密钥。用户 U 将 m_2 转发给 V,并且 U 将使用新的会话密钥 K 加密 T 和 ID(U),将加密结果 m_3 发送给 V。当用户 V 从 U 那里收到 m_2 和 m_3 时,他解密 m_2 获得 T, K, L 和 ID(U),然后使用 K 解密 m_3 获得 T 和 ID(U),并验证两个 T 值和两个 ID(U)值一样。这可使 V 相信加密在 m_2 中的会话密钥和用于加密 T,ID(U)的密钥是一样的。V 使用 K 加密 $T+1$ 并把所得的结果 m_4 送回 U。当 U 收到 m_4 时,他使用 K 解密并验证解密结果是 $T+1$。这可使 U 相信会话密钥 K 已经被成功地传输给了 V,因为产生消息 m_4 时需要 K。消息 m_1 和 m_2 用来提供会话密钥 K 在传输过程中的秘密性。m_3 和 m_4 用来提供密钥确证性(Key Confirmation),也就是能使 U 和 V 相互相信他们拥有同样的会话密钥 K。在大多数密钥分配方案中,都可实现密钥确证性这一特性。一般地,可类似于 Kerboros 协议中的做法来实现这一特性,即通过使用新的会话密钥 K 加密已知的量。在 Kerboros 协议中,U 使用 K 加密 T 和 ID(U)。V 使用 K 加密 $T+1$。

时戳 T 和生存期 L 的目的是阻止一个主动的敌手存储旧消息并在后来重发,这种攻击称为重放攻击(Replay Attack)。这种方法之所以奏效,是因为一旦密钥过期,它不能再被作为合法的密钥接收。

Kerboros 协议的缺点之一是网络中的所有用户都得同步时钟,因为目前的时间被用来确定是否一个给定的会话密钥 K 是合法的。在实际中,提供完全的同步是很困难的,所以允许有一定量的时差。

Kerberos 的口令没有进行额外的特殊处理,使用穷举攻击法的时间复杂性仅和口令长度成比例。如果增加密钥长度换取更高的安全,又会造成用户的使用困难和增加系统加/ 解密开销。

Kerberos 认证中心要求保存大量的共享私钥,无论是管理还是更新都有很大的困难,需要特别细致的安全保护措施,将付出极大的系统代价。无疑,这是基于对称密码体制的安全认证体系固有的局限性,如果采用基于公钥体制的安全认证方式,就能彻底解决密钥管理的问题。

2. 非对称系统的密钥分配

这部分先讨论一个有趣的无条件安全的密钥预分配协议,即 Blom 密钥分配方案,它大大减少了用户需要存储的秘密消息总量。然后介绍基于离散对数问题的计算上安全的 Diffie 和 Hellman 密钥预分配方案。

(1) Blom 密钥分配方案

假定我们有一个 n 个用户的网络。为方便起见,假定密钥选自有限域 Z_p, $p \geq n$,是一个素数。设 k 是一个整数,$1 \leq k \leq n-2$, k 是使得网络中的任何 k 个用户合伙方案仍是安全的最大值。在 Blom 方案中(也称规模为 k 的 Blom 方案),可信中心通过一个安全信道给每个用户传送 $k+1$ 个 Z_p 中元素。每一对用户 U 和 V 将能计算一个密钥 $K_{U,V} = K_{V,U}$。安全性条件为:至多 k 个不同的用户的任何用户集必须不能确定关于 $K_{U,V}$ 的任何信息(这里所说安全性是指无条件安全性)。我们首先提出 Blom 方案的一个特例,即 $k=1$(规模为 1 的 Blom 方案)。此时,可信中心给每个用户分发 Z_p 中的两个元素。任何单个用户 $W(W \neq U, V)$ 不能确定关于 $K_{U,V}$ 的任何信息。下面描述 Blom 密钥分配方案($k=1$)。

① 公开一个素数 p,每个用户 U 公开一个元素 $r_U \in Z_p$,这些元素 r_U 必须互不相同。

② 可信中心选择 3 个随机元素 $a, b, c \in Z_p$(未必不同),并且形成多项式

$$f(x,y) = (a + b(x+y) + cxy) \bmod p。$$

③ 对每一个用户 U,可信中心计算多项式:$g_U(x) = f(x, r_U) \bmod p$,并将 $g_U(x)$ 在一个安全信道上传送给 U。注意,$g_U(x)$ 是 x 的一个线性函数,所以它可以写为:$g_U(x) = a_U + b_U x$,这里,$a_U = (a + br_U) \bmod p$, $b_U = (b + cr_U) \bmod p$。

④ 如果 U 和 V 想通信,那么他们使用共同密钥 $K_{U,V} = K_{V,U} = f(r_U, r_V)$。

这里,U 计算 $K_{U,V} = f(r_U, r_V) = g_U(r_V)$, V 计算 $K_{V,U} = f(r_V, r_U) = g_V(r_U) = K_{U,V}$。

下面证明没有别的用户可以确定某两个用户的密钥的任何信息。

定理 14.1　$k=1$ 时的 Blom 方案对任何单个用户是无条件安全的。

证明　假定用户 W 试图计算密钥

$$K_{U,V} = (a + b(r_U + r_V) + cr_U r_V) \bmod p,$$

其中,值 r_U 和 r_V 是公开的,但值 a, b, c 是未知的。因为可信中心将 $g_W(x)$ 通过一个安全信道发送给了 W,所以,W 知道值 $a_W = (a + br_W) \bmod p$, $b_W = (b + cr_W) \bmod p$。

下面说明 W 知道的信息和密钥 $K_{U,V}$ 的任何的取值 $l \in Z_p$ 一样。因此,W 不能排除 $K_{U,V}$ 的任何值。考虑下列矩阵方程:

$$\begin{matrix} 1 & r_U + r_V & r_U r_V \\ 1 & r_W & 0 \\ 0 & 1 & r_W \end{matrix} \begin{matrix} a \\ b \\ c \end{matrix} = \begin{matrix} l \\ a_W \\ b_W \end{matrix}$$

第一个方程来自假定 $K_{U,V}=l$,第二个和第三个方程来自 W 从 $g_W(x)$知道的关于 a,b,c 的信息。系数矩阵的行列式为

$$r_W^2+r_Ur_V-(r_U+r_V)r_W=(r_W-r_U)(r_W-r_V)。$$

这里所有的运算都是 Z_p 上的运算。因为 $r_W\neq r_U,r_V$,所以,上述矩阵方程关于 a,b,c 有惟一的一个解。换句话说,$K_{U,V}$的任何可能的值 l 和 W 知道的信息一样。

两个用户(W 和 X)合伙便能确定任何密钥 $K_{U,V}$,这里 $W,X\cap U,V=\varnothing$。W 和 X 知道

$$a_W=a+br_W$$

$$b_W=b+cr_W$$

$$a_X=a+br_X$$

$$b_X=a+cr_X$$

这样,有 4 个方程式 3 个未知量,用户就可容易地计算出 a,b,c 的值。一旦用户知道 a,b,c 的值,他们就能形成多项式 $f(x,y)$,并能计算他们所希望的任何密钥。

规模为 1 的 Blom 方案可直接推广为规模为 $k(1\leq k\leq n-2)$的 Blom 方案。惟一需要改变的就是第②步。可信中心将使用一个如下形式的多项式:

$$\sum_{i=0}^{k}\sum_{j=0}^{k}a_{ij}x^iy^j \bmod p。$$

这里,$a_{ij}\in Z_p(0\leq i\leq k,0\leq j\leq k)$,并且对所有的 i,j,$a_{ij}=a_{ji}$。协议的其余步骤不需要改变。

(2) Diffie-Hellman 密钥预分配方案

如果离散对数问题是难解的,则这个方案是计算上安全的。

我们仅描述在 Z_p 上的一个方案,p 是一个素数。实际上该方案可在计算离散对数问题是难处理的任何有限群上实现。假定 α 是 Z_p 的一个本原元,网络中的任何用户都知道 p 和 α 的值。

在这个方案中,我们用 ID(U)表示网络中用户 U 的某些识别信息,诸如姓名、E-mail 地址、电话号码或别的有关信息。每个用户 U 有一个秘密指数 $a_U(0\leq a_U\leq p-2)$和一个相应的公钥 $b_U=\alpha^{a_U}\bmod p$。

可信中心有一个签名方案,该签名方案的公开验证算法记为 Ver_{TA},秘密签名算法记为 Sig_{TA}。我们暗含着在签名消息之前,先将消息用一个公开的 Hash 函数杂凑。但为了叙述简单起见,这里略去这一步。

当一个用户 U 入网时,可信中心需给他颁发一个证书(Certificate)。用户 U 的证书为:$C(U)=(ID(U),b_U,\mathrm{Sig}_{TA}(ID(U),b_U))$,可信中心无需知道 a_U 的值。证书可存储在一个公开的数据库中,也可由用户自己存储。可信中心对证书的签名允许网络中的任何人能验证它所包含的信息。U 和 V 计算共同的密钥 $k_{U,V}=\alpha^{a_Ua_V}\bmod p$ 的协议,即

Diffie-Hellman 密钥预分配协议如下:

① 公开一个素数 p 和一个本原元 $\alpha \in Z_p^*$。

② V 使用他自己的秘密值 α_V 及从 U 的证书中获得的公开值 b_U, 计算

$$k_{U,V} = \alpha^{a_U a_V} \bmod p = b_U^{a_V} \bmod p。$$

③ U 使用他自己的秘密值 a_U 及从 V 的证书中获得的公开值 b_V, 计算

$$k_{U,V} = \alpha^{a_U a_V} \bmod p = b_V^{a_U} \bmod p。$$

现在我们来考虑一下 Diffie-Hellman 密钥预分配方案的安全性。可信中心对用户的证书的签名有效地阻止了敌手 W 改变别人的证书的任何信息, 从而阻止了 W 的主动攻击。因此, 我们关心的是 W 的被动攻击, 这样, 与此相关问题就是一个用户 $W(\neq U, V)$能否计算 $k_{U,V}$? 换句话说, 给定 $\alpha^{a_U} \bmod p$ 和 $\alpha^{a_V} \bmod p$, 但不知道 a_U 和 a_V, 计算 $\alpha^{a_U a_V} \bmod p$ 是否可行 ? 这个问题称为 Diffie-Hellman 问题。问题可等价地陈述为: 给定 p, α, β 和 γ, p 为一个素数, $\alpha \in Z_p^*$ 是一个本原元, $\beta, \gamma \in Z_p^*$, 计算 $\beta^{\log_\alpha \gamma} \bmod p = \gamma^{\log_\alpha \beta} \bmod p$。显然, Diffie-Hellman 密钥预分配方案对一个被动的敌手是安全的, 当且仅当 Diffie-Hellman 问题是难处理的。

如果 W 能从 b_U 确定 a_U, 或从 b_V 确定 a_V, 那么他一定能像 U 或 V 一样计算出 $k_{U,V}$。但我们假定 Z_p 上的离散对数问题是难处理的, 所以对 Diffie-Hellman 密钥预分配方案的这种类型的攻击是计算上不可行的。人们猜测解 Diffie-Hellman 问题的任何算法也可用来解离散对数问题。但至今这个猜测还未得到证明。正像讨论 RSA 体制的安全性一样, 人们猜测但未被证明, 破译 RSA 体制多项式等价于因子分解。

虽然我们不能精确地说 Diffie-Hellman 问题有多难, 但我们可以将这个问题和我们已讨论过的 ELGamal 体制的安全性联系起来。

定理 14.2　破译 ELGamal 体制等价于解 Diffie-Hellman 问题。

证明　ElGamal 体制的秘密密钥为 a, 公钥 $\beta = \alpha^a \bmod p$, p 和 α 是公开知道的, p 是一个素数, α 是 Z_p 的一个本原元。

对消息 $x \in Z_p$ 的加密过程为: 随机选择一个数 $k \in Z_{p-1}$, $E_k(x, k) = (y_1, y_2)$, 这里, $y_1 = \alpha^k \bmod p$, $y_2 = x\beta^k \bmod p$。

解密过程为:

$$D_k(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p, \ y_1, y_2 \in Z_p^*。$$

假定我们有一个算法 A, A 能解 Diffie-Hellman 问题, 并给定一个用 ELGamal 体制加密的密文(y_1, y_2)。我们将 p, α, y_1 和 β 作为算法 A 的输入, 则我们获得 $A(p, \alpha, y_1, \beta) = A(p, \alpha, \alpha^k, \alpha^a) = \alpha^{ka} \bmod p = \beta^k \bmod p$。此时密文$(y_1, y_2)$对应的明文 x, 可由下式容易求出: $x = y_2 (\beta^k)^{-1} \bmod p$。

反之, 假定我们有一个算法 B, B 能完成 ELGamal 体制的解密过程。也就是说, 如果

将 p, α, β, y_1 和 y_2 作为 B 的输入，那么我们能求得 $B(p, \alpha, \beta, y_1, y_2) = x = y_2 y_1^{-\log_\alpha \beta} \bmod p$。现在给定 p, α, β 和 γ, p 为一个素数，$\alpha \in Z_p^*$ 是一个本原元，$\beta, \gamma \in Z_p^*$，则可利用下列办法计算出

$$\gamma^{\log_\alpha \beta} : B(p, \alpha, \beta, \gamma, 1)^{-1} = 1 \quad \gamma^{-\log_\alpha \beta \cdot (-1)} \bmod p = \gamma^{\log_\alpha \beta} \bmod p。$$

14.3 密钥共享

存储在系统中的所有密钥的安全性(从而整个系统的安全性)可能最终取决于一个主密钥。这样做有两个缺陷：一是若主密钥偶然地或蓄意地被暴露，整个系统就易受攻击；二是若主密钥丢失或毁坏，系统中所有信息就用不成了。后一个问题可通过将密钥的副本发给信得过的用户来解决，但这样做时，系统对背叛行为又无法对付。解决这两个问题的一个办法是使用密钥共享方案(Secret Key Share Scheme)。密钥共享方案的基本观点是：将主密钥 k 按下列方式分成 n 个共享(Share) $k_1, k_2, \cdots, k_n$：

(1) 已知任意 t 个 k_i 值易于算出 k；

(2) 已知任意 $t-1$ 个或更少个 k_i，则由于信息短缺而不能确定出 k。这种方法也称为 (t, n)门限(Threshold)法。

将 n 个共享 $k_1, k_2, \cdots, k_n$ 分给 n 个用户。由于要重构密钥要求至少有 t 个共享，故暴露 $s(s \leq t-1)$个共享不会危及密钥，且少于 t 个用户的组不可能共谋得到密钥；同时，若一个共享被丢失或毁坏，仍可恢复密钥(只要至少有 t 个有效的共享)，这种方法为将密钥分给多人掌管提供了可能。下面介绍两个密钥共享方案。

1. Shamir 门限方案

Shamir 于 1979 年基于拉格朗日内插多项式提出了一个门限方案。本小节来介绍这一方案。

设 GF(q)是一有限域，共享的密钥 $k \in K = \mathrm{GF}(q)$。可信中心给 $n(n < q)$个共享者 $P_i(1 \leq i \leq n)$分配共享的过程如下：

(1) 可信中心随机选择一个 $t-1$ 次多项式 $h(x) = a_{t-1} x^{t-1} + \cdots + a_1 x + a_0 \in \mathrm{GF}(q)[x]$，常数 $a_0 = k$ 为主密钥；

(2) 可信中心在 GF(q)中选择 n 个非零的互不相同元素 $x_1, x_2 \cdots, x_n$，计算 $y_i = h(x_i), 1 \leq i \leq n$；

(3) 将子密钥$(x_i, y_i)(1 \leq i \leq n)$分配给共享者 $P_i(1 \leq i \leq n)$，值 $x_i(1 \leq i \leq n)$是公开知道的，$y_i(1 \leq i \leq n)$作为 $P_i(1 \leq i \leq n)$的密钥共享。

每对(x_i, y_i)就是“曲线”$h(x)$上的一个点。因为 t 个点惟一地确定 $t-1$ 次多项式 $h(x)$，所以 k 可以从 t 个共享中重构出。但是从 $t_1(t_1 < t)$个共享无法确定 $h(x)$或 k。

给定 t 个共享 $y_{i_s}(1\leq s\leq t)$,从拉格朗日多项式重构的 $h(x)$为

$$h(x)=\sum_{s=1}^{t} y_{i_s}\prod_{\substack{j=1\\ j\neq s}}^{t}\frac{x-x_{i_j}}{x_{i_s}-x_{i_j}},$$

运算都是 GF(q) 上的运算。

一旦知道 $h(x)$,通过 $k=h(0)$ 易于计算出密钥 k。因为

$$k=h(0)=\sum_{s=1}^{t} y_{i_s}\prod_{\substack{j=1\\ j\neq s}}^{t}\frac{-x_{i_j}}{x_{i_s}-x_{i_j}}。$$

若令

$$b_s=\prod_{\substack{j=1\\ j\neq s}}^{t}\frac{-x_{i_j}}{x_{i_s}-x_{i_j}},$$

则

$$k=h(0)=\sum_{s=1}^{t} b_s y_{i_s}。$$

因为 $x_i(1\leq i\leq n)$的值是公开知道的,所以可预计算 $b_s(1\leq s\leq n)$以加快重构时的运算速度。

例 14.1　设 $t=3, n=5, q=19, k=11$,随机选取 $a_1=2, a_2=7$,得多项式为

$$h(x)=(7x^2+2x+11)\bmod 19,$$

选择 $x_i=i$,则由 $y_i=h(x_i), 1\leq i\leq 5$,很容易得 5 个子密钥

$$h(1)=1,\ h(2)=5,\ h(3)=4,\ h(4)=17,\ h(5)=6。$$

如果知道其中的 3 个子密钥 $h(2)=5,\ h(3)=4,\ h(5)=6$,就可按以下方式重构 $h(x)$:

$$5\frac{(x-3)(x-5)}{(2-3)(2-5)}=5\frac{(x-3)(x-5)}{(-1)(-3)}=5\cdot(3^{-1}\bmod 19)\cdot(x-3)(x-5)$$
$$=5\cdot 13(x-3)(x-5)=65(x-3)(x-5),$$

$$4\frac{(x-2)(x-5)}{(3-2)(3-5)}=4\frac{(x-2)(x-5)}{(1)(-2)}=4\cdot((-2)^{-1}\bmod 19)\cdot(x-2)(x-5)$$
$$=4\cdot 9(x-2)(x-5)=36(x-2)(x-5),$$

$$6\frac{(x-2)(x-3)}{(5-2)(5-3)}=6\frac{(x-2)(x-3)}{(3)(2)}=6\cdot(6^{-1}\bmod 19)\cdot(x-2)(x-3)$$
$$=6.16(x-2)(x-3)=96(x-2)(x-3),$$

所以

$$h(x)=[65(x-3)(x-5)+36(x-2)(x-5)+96(x-2)(x-3)]\bmod 19$$
$$=(26x^2-188x+296)\bmod 19$$
$$=7x^2+2x+11。$$

从而得密钥 $k=11$。

2．Asmuth-Bloom 门限方案

Asmuth 和 Bloom 于 1980 年基于中国剩余定理提出了一个门限方案。在他们的方案中，共享是和密钥 k 相联系的一个数的同余类。令 $p, d_1, d_2, \cdots, d_n$ 是满足下列条件的一组正整数：

(1) $p > k$;

(2) $d_1 < d_2 < \cdots < d_n$;

(3) 对所有的 i; $\gcd(p, d_i) = 1$; 对 $i \neq j$, $\gcd(d_i, d_j) = 1$;

(4) $d_1 d_2 \cdots d_t > p d_{n-t+2} d_{n-t+3} \cdots d_n$。

令 $N = d_1 d_2 \cdots d_t$ 是 t 个最小整数之积，则 N/p 大于任意 $t-1$ 个 d_i 之积。令 r 是区间 $0, \lfloor N/p \rfloor - 1$ 中的一个随机整数，并公布 p, r。为了将 k 划分为 n 个共享，计算 $k' = k + rp$，则 $k' \in [0, N-1]$。n 个共享为 $k_i = k' \bmod d_i$, $i = 1, 2, \cdots, n$，将子密钥 (d_i, k_i) 分配给 p_i，d_i 是公开知道的，k_i 为 p_i 的密钥共享。

为了恢复 k，找到 k' 就足够了。若给定 t 个共享 $k_{i_1}, \cdots, k_{i_t}$，则由中国剩余定理可知，同余方程组

$$x' \equiv k_{i_1} (\bmod d_{i_1})$$
$$x' \equiv k_{i_2} (\bmod d_{i_2})$$
$$\cdots$$
$$x' \equiv k_{i_t} (\bmod d_{i_t})$$

关于模 $N_1 = d_{i_1} d_{i_2} \cdots d_{i_t}$ 在 $[0, N_1 - 1]$ 内有惟一解 x，因为 $N_1 \geq N \geq k'$，推出 $k' = x$。最后，从 k', r 和 p 计算 k: $k = k' - rp$，即 $k = k' \bmod p$。

若仅知道 $t-1$ 个共享 $k_{i_1}, \cdots, k_{i_{t-1}}$，可能就只知道 k' 关于模 $N_2 = d_{i_1} d_{i_2} \cdots d_{i_{t-1}}$ 在 $[0, N_2 - 1]$ 内有惟一解 x。因为 $N/N_2 > p$, $\gcd(p, N_2) = 1$，所以使 $x \leq N$ 且 $x \equiv k' \bmod p$ 的数 x 在模 p 的所有同余类上均匀地分布，因此，没有足够的信息去确定 k'。

例 14.2 设 $t=2$, $n=3$, $p=7$, $k=4$, $d_1=9$, $d_2=11$, $d_3=13$，则

$$N = d_1 d_2 = 99 > 91 = 7 \cdot 13 = p \cdot d_3。$$

在 $[0, \lfloor 99/7 \rfloor - 1] = [0, 13]$ 之间随机地取 $r=10$，求 $k' = k + rp = 4 + 10 \times 7 = 74$，

$k_1 \equiv k' \bmod d_1 = 74 \bmod 9 \equiv 2$, $k_2 \equiv k' \bmod d_2 = 74 \bmod 11 \equiv 8$,

$k_3 \equiv k' \bmod d_3 = 74 \bmod 13 \equiv 9$。

$\{(9,2), (11,8), (13,9)\}$ 即构成 $(2,3)$ 门限方案。

若知道 $\{(9,2), (11,8)\}$，可建立方程组

$$2 \bmod 9 \equiv k'$$
$$8 \bmod 11 \equiv k'。$$

解之得 $k' \equiv (11\times5\times2+9\times5\times8) \bmod 99 \equiv 74$，所以 $k = k' - rp = 74 - 10\times7 = 4$。

在上述两个密钥共享的门限方案中，n 个共享者(也称受托人)的权限是同样的。但在实际应用中，由于受托人的地位和职能的不同，其权限通常是不同的，因此，需要考虑更一般的密钥共享体制。先引进一般存取结构的概念。

设 $S = \{P_1, P_2, \cdots, P_n\}$ 为受托人集。记 2^S 为 S 的所有子集(包括空集)组成的集族。一个 S 的子集族 $\Gamma \quad 2^S$ 称为 S 上的一个存取结构(Access Structure)，若对受托人集 A 满足如下性质：$A \in \Gamma$ 当且仅当 A 可以恢复主密钥 k，子集 $A \in \Gamma$ 称为授权集。同时，把 $\Omega = 2^S - \Gamma$ 称为 S 上的非存取结构，子集 $B \in \Omega$ 称为非授权集。这样，(t, n) 门限方案的存取结构就可表示为 $\Gamma = \quad A; A \quad S, |A| \geq t$，是上述一般存取结构的特殊情形，也称门限存取结构。

容易看出，存取结构 Γ 满足单调性，即若 $A \in \Gamma$ 且 $A \quad A' \quad S$，则 $A' \in \Gamma$。任给一存取结构 Γ，记

$$\Gamma_m = \{A; A \in \Gamma, \text{且对任一 } A' \quad A, A' \neq A \text{ 有 } A' \notin \Gamma\},$$

称 Γ_m 为 S 上的极小存取结构，$A \in \Gamma_m$ 称为极小授权集。由 Γ_m 的定义可见，Γ_m 由 Γ 惟一确定；反之，由存取结构 Γ 的单调性，Γ 也由 Γ_m 惟一确定。因此，在描述存取结构 Γ 时，通常只需给出 Γ_m 即可。

在实际应用中，存取结构是根据需要预先给定的。Ito 等人证明：对于集族 $\Gamma \quad 2^S$，存在实现存取结构 Γ 的密钥共享方案的充要条件是 Γ 满足单调性。因此，可把含于 2^S 中的单调的子集族称作存取结构。Ito 等人利用 Shamir 门限方案的思想，给出了实现一般存取结构的密钥共享方案。下面介绍他们的方法。

设 S 为受托人集，Γ 为任给的 S 上的存取结构，$\Omega = 2^S - \Gamma$ 为与 Γ 对应的非存取结构。构造

$$\Omega_M = \quad B; B \in \Omega, \text{且对任一 } B' \quad B, B' \neq B \text{ 有 } B' \notin \Omega,$$

并算出其中集的数目 $|\Omega_M| = t$。不妨将 Ω_M 记作 $\Omega_M = \{B_1, B_2, \cdots, B_t\}$。设 $GF(q)$ 为有限域且满足 $q > t$。可信中心给 n 个受托人 $P_i (i = 1, \cdots, n)$ 分配共享的算法如下：

(1) 可信中心随机选取一个 $t-1$ 次多项式 $h(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + a_0 \in GF(q)[x]$，其中 $a_0 = k$ 为主密钥；

(2) 可信中心在 $GF(q)$ 中选择 t 个非零的互不相同元素 $x_1, x_2, \cdots, x_t$，计算 $y_i = h(x_i), (i = 1, \cdots, t)$；

(3) 可信中心将子密钥集 $K_i = \quad (x_j, y_j); 1 \leq j \leq t, P_i \notin B_j \in \Omega_M$ 分配给 $P_i (1 \leq i \leq n)$ 作为 $P_i (1 \leq i \leq n)$ 的密钥共享。

现在证明上述算法是存取结构 Γ 的一个密钥共享方案。先证 $A \in \Gamma$ 可恢复主密钥 k。事实上，由于 $\Omega_M \quad \Omega = 2^S - \Gamma$，故对每个 $B_j \in \Omega_M$，存在 $i \in \{1, \cdots, n\}$ 使 $P_i \in A$ 且 $P_i \notin B_j$，否则有 $A \quad B_j$，与 Ω 的定义矛盾。因此，A 拥有的子密钥集满足 $\left|\bigcup_{P_i \in A} K_i\right| = t$，按照

Shamir 门限方案中的方法 A 可恢复主密钥 k。再证 $B\in\Omega$ 无法得到主密钥 k。事实上，由 Ω_M 的定义，存在 $B_j\in\Omega_M$ 使 $B \quad B_j$，再由 K_i 的定义得$(x_j, y_j)\notin\bigcup_{P_i\in B}K_i$，即 B 拥有的子密钥数 $t_1\leq t-1$，故无法得到主密钥 k。

上述算法说明，对任何事先给定的存取结构，都存在一个实现这个存取结构的密钥共享方案，因而一般存取结构概念是有实际意义的。当然 Ito 等人给出的实现一般存取结构的密钥共享方案不是很理想的。不严格地说，一个理想的密钥共享方案应具有如下两个性质：

(1) 完全性，即任何授权集 $A\in\Gamma$ 可恢复主密钥 k，而任何非授权集 $B\in\Omega$ 不能得到主密钥 k 的任何信息；

(2) 无数据扩散性，即表示每个共享 $k_i(1\leq i\leq n)$的比特数与表示主密钥 k 的比特数在平均的意义下都是相等的。

从上述算法中的(3)可见，Ito 等人给出的实现一般存取结构的密钥共享方案是有数据扩散的，甚至可能扩散的很多。目前已用各种数学方法研究出许多实现一般存取结构的更理想的密钥共享方案。有关这方面的理论以及各种类型的密钥(秘密)共享方案，有兴趣的读者可参看本书参考文献[26]第 10 章所引的有关文献，这里就不多介绍了。

14.4 密钥托管

随着保密通信在现代社会中的作用日益加强，保密通信设备已广泛地用于政府部门、企业内部、银行系统等，对这些部门的信息保密起了很好的作用。但这些部门内部人员以及一些犯罪分子也可利用该通信设备合法传送他们的犯罪信息，进行犯罪活动，如恐怖威胁、有组织贩毒、泄露企业内部秘密、危害国家及企业的利益，为了监视和防止这种犯罪活动，人们提出了密钥托管概念。

密钥托管系统是具有备份能力的加密系统，允许授权者，包括用户、民间组织、政府机构在特定条件下，借助于一个以上持有专用数据恢复密钥的、可信赖的委托方所提供的信息来解密密文。“数据恢复密钥”不同于通常用于数据加解密的密钥，它只提供一种确定数据加解密密钥的方法。所谓密钥托管是指存储这些数据恢复密钥的方案。

密钥托管的前提是：用户不应具有在他们中间分配其他秘密密钥的能力，否则用户用这些密钥替代密钥托管机构能控制的密钥，从而绕过密钥托管，使密钥托管失效。密钥托管可以简单地理解为把通信双方每次使用的会话密钥交给合法的第三方，以便让合法的第三方利用得到的会话密钥解密双方通信的内容，从而来监视双方通信。一般而言，合法的第三方为政府部门和法律执行部门等。

密钥托管的一个最简单方法就是由一个(或多个)可信的政府代理机构为个人保管秘密密钥。这样经过适当的授权，政府就能检索存储的秘密密钥，进而对窃听到的通信进行

脱密,然而简单地存储几个秘密密钥可能很便宜,但这样一种解决办法能够被讹误。为了防止讹误,人们总结和提出了各种密钥托管方法,它包括使用私钥密码体制、公钥密码体制、私钥密码与公钥密码体制结合、密钥共享;它们不仅可完成密钥托管,而且能完成数字签名、鉴别等其他一些公钥密码和私钥密码所具有的性质。

通过对众多托管方案的总结,人们形成了密钥托管体制的一般模式。

14.4.1　密钥托管体制的基本组成

逻辑上,一个密钥托管加密体制可分为用户安全模块、密钥托管模块和数据恢复模块三个主要部分。它们之间密切相关,相互影响。图 14.3 表明这几个分量之间的相互关系。

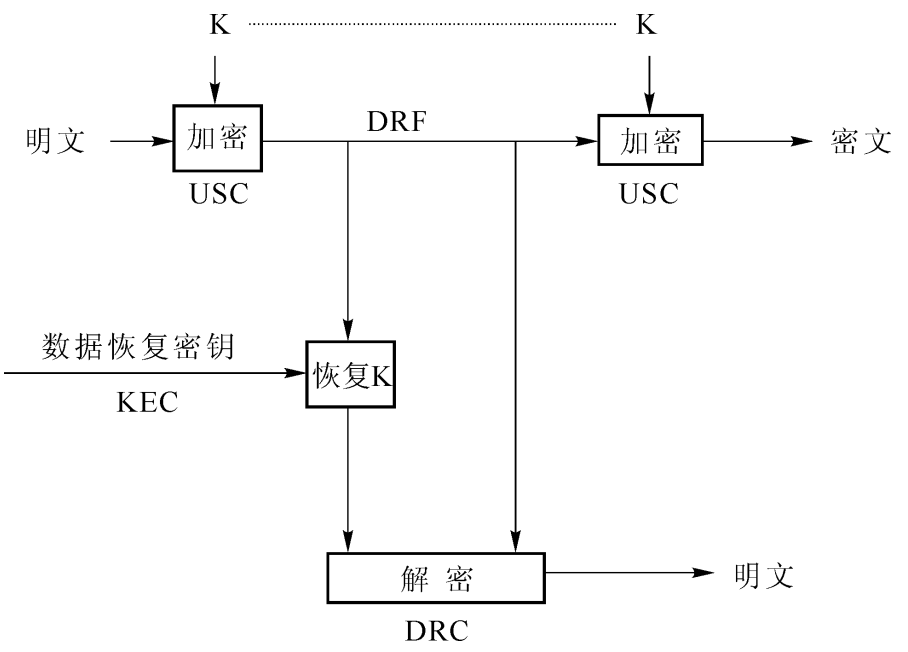


图 14.3　密钥托管加密体制

用户安全模块(USC: User Security Component)是硬件设备或软件程序,提供数据加密、解密能力,同时也支持密钥托管。这种支持体现在将数据恢复字段(Data Recovery Field-DRF)附加到加密数据上,DRF 可作为通用密钥分配体制的组成部分。

密钥托管模块(KEC: Key Escrow Component)是由密钥托管代理操作的,管理着数据恢复密钥的存储、传送或使用,它可以作为公钥管理系统的组成部分,也可以作为通用密钥管理的基础部分。

数据恢复模块(DRC: Data Recovery Component)由算法、协议和设备组成。仪器设备可以从 KEC 所提供的和 DRF 中包含的信息中恢复出数据加密密钥,从而解密密文。只有在执行规定的方法数据恢复时才能使用 DRC。

USC 使用密钥 K 加密明文数据,并把 DRF 附加于密文上;DRC 则从 KEC 提供的和 DRF 中包含的信息恢复出密钥 K、解密密文。

有关各个模块的功能、任务和安全要求的详细描述请参考本书参考文献[25]。

14.4.2　密钥托管体制实例

1993 年 4 月 16 日,美国政府宣布了一项新的建议,该建议倡导联邦政府和工业界使

用新的具有密钥托管功能的联邦加密标准。该建议称为托管加密标准(EES: Escrowed Encryption Standard),又称 Clipper 建议。其目的是为用户提供更好的安全通信方式,同时允许政府为了国家安全监听某些通信。这个建议的核心是一个新的称之为 Clipper 的防窜扰芯片,它是由 NSA 主持开发的硬件实现的密码设备,采用了称之为 Skipjack 的私钥密码算法,芯片的单元密钥(UK)由两个称之为 Escrow 的机构联合提供。1994 年 2 月,美国政府公布采用 EES(参见本书参考文献[27])。

EES 采用单钥分组密码 Skipjack,字长 64 比特,密钥 80 比特,32 轮置乱。加密速度为 12Mbit/ s,较 DES 安全得多。

在使用私钥密码体制进行密钥托管时,必须要求把私钥密码体制放在一个具有防拆性功能的芯片中,所以 EES 的安全性集中于防拆芯片硬件。

但是,EES 也存在明显的问题:它需要广泛的政府托管密钥数据库的支持;而且,如果某硬件芯片的密钥泄密了,芯片就永远失效了。政府颁布了一系列规则来维护双方利益并防止滥用,包括建立两个密钥托管部门,使得任何一方都不能掌握全部密钥,也不能控制由谁接触托管密钥。实际上,EES 的最大问题在于它没有带给用户任何利益,政府总是能解密用户的通信内容,而用户丢失了密钥和密钥损坏时,EES 却无能为力。

EES 推广最终遭遇失败,1995 年 7 月,美国政府宣布放弃用 EES 来加密数据,只将它用于语音通信。尽管如此,EES 的方法对随后的密钥托管体制的研究还是起了很大的推动作用。有关 EES 的详细描述请读者自己参看本书参考文献[27]。

注 记

密钥管理是一个重要而又棘手的问题,因此,必须给予高度重视和保护。本章介绍了密钥管理所涉及的方面(密钥的种类、密钥的生成、密钥的更换、密钥的存储、密钥的销毁和密钥的吊销等),两类密钥分配协议(基于对称的和基于非对称的密码体制),密钥共享的方案以及密钥托管体制。目前有关密钥管理的讨论已有不少,本章主要参考了本书参考文献[63]和[6]。

Diffie 与 Hellman 的公开密钥分配系统只考虑到两个人的通信情况。随着多媒体的发展和网络的普及,电子会议的应用将成为一个崭新而重要的课题。如何让一群使用者在网络上分享一把共同的加解密密钥,而使他们在网络上能安全地交谈,将是一个重要的研究课题。Chen 与 Hwang 提出了具有使用者辨认能力的会议密钥分配系统[64]。

对密钥共享感兴趣的读者可以参阅本书参考文献[26]第 10 章所引的有关文献。

最后需要提醒注意的是,每个具体系统的密钥管理必须与具体的使用环境和保密要求相结合,万能的密钥管理体制是不存在的。

习题十四

1．Diffie-Hellman 密钥预分配协议易受中间人攻击，即攻击者截获通信双方通信的内容后可分别冒充通信双方，以获得通信双方协商的密钥。请详细分析攻击者如何实施攻击。

2．在 Diffie-Hellman 密钥交换过程中，设大素数 $p = 11$，$\alpha = 2$ 是 Z_p 的本原元，

(1) 用户 U 的公开钥 $b_U = 9$，求其秘密钥 a_U。

(2) 设用户 V 的公开钥 $b_V = 3$，求 U 和 V 的共享密钥 K。

3．在 Shamir 门限方案中，设 $t = 3$，$n = 5$，$p = 17$，5 个子密钥分别是 8、7、10、0、11，从中任选 3 个，构造插值多项式并求共享密钥 k。

4．在 Asmuth-Bloom 密钥分享方案中，设 $t = 2$，$n = 3$，$p = 5$，$k = 4$，$d_1 = 7$，$d_2 = 9$，$d_3 = 11$，3个子密钥分别是 6、3、4，求共享的密钥。

第 15 章　零知识证明

在引论中已提到,零知识证明是构造安全(密码学)协议的主要工具。不严格地说,零知识证明除了证明结论的正确性外不泄漏任何其他信息(知识),因此,它可以用作一个密码学协议的一部分使各方能放心地执行该协议。零知识证明可分为交互式的和非交互式的两个类型。本章将介绍交互式零知识证明系统与非交互式零知识证明系统的定义及其构造方法。

15.1　交互零知识证明系统的定义

在给出正式定义之前,我们先对交互式零知识证明系统作一些非正式的描述。一个交互式证明系统有两个参与者甲和乙,甲称为证明者,乙称为验证者。甲知道某一事实,如 $x \in I_D^+$ 是一个判定问题 D 的答案为"是"的例子,甲希望给乙证明他知道这一事实。我们必须描述甲和乙允许进行怎样的计算,还要描述他们之间如何交互。较易研究的系统是甲和乙都可用概率算法,他们可进行各自的计算且都拥有一个随机比特产生器。他们可通过一个信道相互通信,开始时甲和乙有共同的输入 x,交互证明的目的是甲要使乙相信 $x \in I_D^+$。交互证明是一个一问一答的协议,它由若干轮构成,在每一轮中,甲和乙轮流作如下工作:(1) 接收从对方发送来的消息;(2) 进行自己的计算;(3) 发送消息给对方。

典型的一轮是先由乙提问,然后甲回答。在协议的最后,乙根据甲是否成功地回答了他所提出的所有问题,接受或拒绝甲证明的事实。一个判定问题 D 的交互证明系统还要满足下面的两个条件:

(1) 完全性:若 x 为判定问题 D 的答案为"是"的例子,则乙接受甲的证明的概率很大;

(2) 合理性:若 x 为判定问题 D 的答案为"否"的例子,则乙接受甲的证明的概率很小。

对交互式证明系统有了一些初步了解后,再来描述交互式零知识证明系统。回忆本章开头的不严格说法,零知识证明就是在乙与甲执行交互证明协议结束后,乙除了知道

$x\in I_D^+$外,没有得到任何其他知识。换句话说,尽管乙通过与甲执行了交互证明协议,相信 $x\in I_D^+$ 这一事实,但他仍没有任何思想如何自己来证明这一事实。要说明的是,这里所说的"得到知识"与(Shannon)信息论中所说的"得到信息"是两个不同的概念:(1)知识与计算复杂性有关而信息则无关。举个例子,若甲和乙的共同输入 x 是一个大正整数,乙问甲 x 是否被 3 整除,甲计算后将结果回答乙。乙从甲的回答中没有得到任何知识,因为乙可以自己计算出这个结果。若乙问甲 x 是否为素数,甲计算后知道 x 是素数,将这结果回答乙。由于乙目前所知的最快素性检验算法也是超多项式时间的,则乙自己不可能计算出这一结果,因此乙从甲的回答中得到了知识。但是从信息论的观点,两个问题的答案都是由问题决定的,故乙从甲的回答中都没有得到信息。假定甲在计算乙的后一问题时,用了一个附加提示(乙不知道这个提示)。甲在回答乙的问题时将这个提示告诉了乙,使乙可在多项式时间内计算出 x 是素数,我们说乙从甲的回答中得到了超过"x 是素数"这一结果的知识。(2)知识是确定性的事物而信息是有某种不确定性的事物。在上面的例子中,若甲用扔硬币所得结果回答乙,那么从信息论的观点,乙从甲的回答中得到了信息,但乙并未得到任何知识,因为乙可以自己扔硬币得到结果。

从以上讨论可见,在描述交互式零知识证明系统时,规定证明者和验证者具有的计算能力是十分重要的。通常,我们限制验证者所进行的计算都是多项式时间的,而对证明者所进行的计算不作任何限制。

现在来给出上面所描述的各个概念的正式定义。首先要定义证明者和验证者所用的计算模型和他们一问一答的交互过程。在第 4.1.2 节中我们定义了确定性单带图灵机表示算法,作为统一的计算模型,在第 4.2.2 节中将确定性单带图灵机推广到概率图灵机表示概率算法。本节要将概率图灵机进一步推广到交互图灵机作为证明者和验证者各自的计算模型,将他们各自的交互图灵机连接起来联合计算就构成一问一答的交互协议。

定义 15.1　一个交互图灵机是一个(确定性)多带图灵机。它具有下列带:(1)一条只读输入带;(2)一条只读随机带;(3)一条读写工作带;(4)一条只写输出带;(5)一条只读通信带和一条只写通信带;(6)一条仅有一个小方格的读写开关带。每个交互图灵机赋予一个二进制数 $\sigma\in\{0,1\}$作为它的识别标记。当它的开关带的内容与它的识别标记 σ 相等时,交互图灵机处于工作周期;否则它处于休息周期。在此期间,交互图灵机的状态、各带读写头的位置以及可写带的内容均不改变。输入带的内容称为输入,随机带的内容称为随机输入,输出带在计算终止时的内容称为输出。在一个工作周期间只写通信带上所写的内容称为在该工作周期发送的消息,同样,在一个工作周期间从只读通信带读出的内容称为在该周期接收的消息。

定义 15.2　二个交互图灵机称为连接在一起,若一个图灵机的识别标记为 1,而另一个图灵机的识别标记为 0,它们的输入带合一,开关带合一,一个图灵机的只读通信带与另一图灵机的只写通信带合一,反之前者的只写通信带与后者的只读通信带合一,但两个图灵机的随机带、工作带和输出带是分开的。一对连接起来的交互图灵机在初始时刻有

共同的输入 x,并置开关带的内容为 0。它们的联合计算程序是一对形的有限或无限序列(参看定义 4.4),其中一个形序列代表一个图灵机的计算程序。序列中的每一对形总是有一个图灵机(不必是同一个)工作而另一个图灵机休息。第一对形对应于图灵机的初始状态,共同输入 x 和开关带的内容 0。若一个图灵机停机了但开关带的内容仍与它的识别标记相等,称这时两个图灵机都已停机了,即计算在这一时刻终止。两个图灵机的输出由这一时刻输出带的内容决定。

还有一个约定,两个图灵机的随机带的内容都是无限长的随机比特序列,即都是相互统计独立等概分布的二值(0 和 1)随机变量序列;且两个序列也是统计独立的。从一个图灵机的随机带读出一个随机数可以看作该图灵机的一个内部扔硬币结果,就像通常的概率图灵机一样(参看 4.2.2 节)。因此,交互图灵机实际上是概率图灵机的推广,这个约定称为随机数约定。

为了定义交互证明系统还要引入下面的两个定义。

定义 15.3 设 A, B 为连接起来的一对交互图灵机,设对每一共同输入,它们的联合计算在有限步终止。记 $(A, B)(x)$ 为共同输入 x 联合计算终止时 B 的输出。它是在 $\{0,1\}$ 中取值的随机变量,即在联合计算终止时刻图灵机 B 的只写头在输出带所写的二进数。由于 A, B 的随机输入满足随机数约定,故 $(A, B)(x)$ 为随机变量。

定义 15.4 一个交互图灵机 A 称为有时间复杂性 $t: \boldsymbol{Z}^+ \to \boldsymbol{Z}^+$ (正整数集),若它与每个交互图灵机 B 连接起来和对每个共同输入 x,它总是在 $t(|x|)$ 步内停机(与 A 和 B 的随机输入无关,只要满足随机数约定即可)。特别若存在一个正多项式 $p(n)$,使图灵机 A 有时间复杂性 $p(|x|)$,则称图灵机 A 是多项式时间的。

现在给出语言 L 成员识别问题的交互证明系统的定义,其中 P 表示证明者所用的图灵机,其识别标记为 0, V 表示验证者所用的图灵机,其识别标记为 1。

定义 15.5 一对连接起来的交互图灵机 (P, V) 称为语言 L 成员识别问题的交互(式)证明系统,若 V 是多项式时间的,且满足下列两个条件。

(1) 完全性:对每个 $x \in L$,

$$\Pr[(P, V)(x) = 1] \geq 2/3 \tag{15.1}$$

(2) 合理性:对每个 $x \notin L$ 和每个交互图灵机 B, B 与 V 连接起来,

$$\Pr[(B, V)(x) = 0] \geq 2/3 \text{ 或 } \Pr[(B, V)(x) = 1] \leq 1/3 \tag{15.2}$$

其中 V 的输出 $(P, V)(x)$ 和 $(B, V)(x)$ 表示验证者是否接受 $x \in L$,输出 1 表示接受 $x \in L$,输出 0 表示拒绝 $x \in L$。

对于定义 15.5,要注意下列几点:

(1) 满足合理性条件是指对所有可能的证明者 B(15.2)式成立;而满足完全性条件是指对规定的一个证明者 P(15.1)式成立。

(2) 要求验证者所用的图灵机 V 是多项式时间的,而对证明者的计算能力和计算资源不作任何限制(包括完全性和合理性条件中的证明者)。

(3) 与 BPP 类问题的定义 4.14 一样,在定义 15.5 的(15.1)式和(15.2)式中的常数 2/3 可换成 $1-2^{-p(|x|)}$,并不影响交互证明系统的定义,其中 $p(n)$为任一固定的正多项式,也可将 2/3 和 1/3 换成两个有明显差别的常数。

每一 NP 类语言 L(的成员识别问题)都有一个简单的交互证明系统。具体地讲,设 $L\in$NP, R_L 为(NP 类问题的定义 4.9 中)与 L 相应的布尔关系(即 $R_L=\{(x,y)\}$满足定义 4.9 中的两个条件)。于是可构造一个交互证明系统如下:当证明者得到一个共同输入 $x\in L$ 后,通过计算找到 $x\in L$ 的一个证据 y,将 y 发送给验证者。验证者收到 y 后,通过计算验证 y 是否满足 $|y|=p(|x|)$(p 为一多项式)以及 $(x,y)\in R_L$。若上述条件满足就输出 1,否则就输出 0。显然 NP 类语言 L 的上述交互证明系统满足定义 15.5 的所有条件,且在(15.1)和(15.2)式中的常数 2/3 都是 1。这是由于证明者和验证者都只用了确定性算法,即没有用他们的随机带。反之容易证明,仅当 $L\in$NP 时才有证明者和验证者都只用确定性算法的交互证明系统。

综上所述,我们得到下面的定理:

定理 15.1　语言 L 的成员识别问题属于 NP,当且仅当它有一个交互证明系统具有下列两个性质:

(1) 交互是单向的(从证明者到验证者);

(2) 证明者和验证者都只用确定性算法(不出错)。

此外,BPP 类语言 L 有一个退化的交互证明系统,即证明者不需要做任何事,验证者自己可通过多项式时间随机算法的输出来决定接受或拒绝 $x\in L$。在一般的交互证明系统中,双向交互和用随机算法看来是交互证明系统的本质所在。

现在我们再来定义交互零知识证明系统。定义零知识的主要思想是验证者与证明者交互(联合)所能计算的随机变量,验证者单独(不与证明者交互)也能计算本质上相同的随机变量。根据随机变量本质相同的不同含意,零知识可分为完全零知识,计算零知识和统计(几乎完全)零知识三种。再根据验证者是否诚实,每种零知识又可分为诚实验证者的零知识和不诚实验证者的零知识。对于诚实的验证者 V,证明者 P 只要与 V 交互且不泄漏知识即可。而对于不诚实的验证者,他为了从证明者 P 得到知识,在与 P 交互的过程中,可能对 P 进行各种欺骗,因此 P 必须与所有可能的验证者(交互图灵机) V^* 交互都不泄漏知识才行。

定义 15.6　设(P,V)为语言 L 的交互证明系统(参看定义 15.5),称(P,V)为语言 L 的一个关于不诚实验证者的交互完全零知识证明系统,若对每个多项式时间的交互图灵机 V^*,存在一个多项式时间的概率图灵机 M^*,使对每个 $x\in L$,下面两个条件成立:

(1)当 M^* 的输入为 x 时,它以 $2^{-p(|x|)}$ 的概率输出一个特殊符号,记作⊥,即 $\Pr[M^*(x)=\perp]\leq 2^{-p(|x|)}$,其中 $p(n)$为任一固定的正多项式。

(2)记 $m^*(x)$为一随机变量,其分布为 $M^*(x)\neq\perp$的条件下 $M^*(x)$的条件分布,即

$$\Pr\left[m^*(x)=\alpha\right]=\Pr\left[M^*(x)=\alpha \mid M^*(x)\neq\perp\right],\ \alpha\in\{0,1\}^*,$$

再记 $\mathrm{View}_{P,V^*}(x)$为共同输入 x 时在 P 与 V^* 交互(联合计算)过程中从 V^* 的随机带读出的随机数以及 V^* 从 P 收到的消息,称为 V^* 的观察。

于是有 $\mathrm{View}_{P,V^*}(x)$与 $m^*(x)$是相同分布的随机变量。

M^* 称为 P 与 V^* 交互的完善模拟器。

前面已提到,在 BPP 类语言 L 的交互证明系统中,证明者不发送任何消息给验证者。因此它是一个退化的关于不诚实验证者的交互完全零知识证明系统。另一方面,在 NP 类语言 L 的交互证明系统中,由于证明者发送给验证者的 $x\in L$ 的证据 y,验证者用多项式时间概率算法不一定能搜索到,因此它不一定是一个交互完全零知识证明系统。

定义 15.7 设(P, V)为语言 L 的交互证明系统,称(P, V)为语言 L 的一个关于不诚实验证者的交互计算零知识证明系统,若对每个多项式时间的交互图灵机 V^*,存在一个多项式时间的概率图灵机 M^*,使随机变量族 $\{\mathrm{View}_{P,V^*}(x)\}_{x\in L}$与 $\{\mathrm{M}^*(x)\}_{x\in L}$是计算不可区分的(参看定义 6.2)。

M^* 称为 P 与 V^* 交互的模拟器。

定义 15.8 设(P, V)为语言 L 的交互证明系统,称(P, V)为语言 L 的一个关于不诚实验证者的交互统计(几乎完全)零知识证明系统,若对每个多项式时间的交互图灵机 V^*,存在一个多项式时间的概率图灵机 M^*,使随机变量族 $\{\mathrm{View}_{P,V^*}(x)\}_{x\in L}$与 $\{\mathrm{M}^*(x)\}_{x\in L}$是统计接近的,即它们的变差距离

$$\Delta(x)=\frac{1}{2}\sum_{\alpha\in\{0,1\}^*}\left|\Pr\left[\mathrm{View}_{P,V^*}(x)=\alpha\right]-\Pr\left[\mathrm{M}^*(x)=\alpha\right]\right|,\ x\in L \quad (15.3)$$

是$|x|$的一个可忽略函数,即对每个正多项式 $p(n)$及一切充分大的$|x|$有 $\Delta(x)<1/p(|x|)$。

从定义 15.6(完全零知识)、15.8(统计零知识)和 15.7(计算零知识)可见,三种定义表示了两个随机变量族 $\{\mathrm{View}_{P,V^*}(x)\}_{x\in L}$(由验证者与证明者交互(联合)计算)和 $\{M^*(x)\}_{x\in L}$(由验证者单独计算)接近程度的三个层次,即完全接近、统计接近和计算接近,也可称为完全不可区分,统计不可区分和计算不可区分。显然,完全零知识蕴含统计零知识,统计零知识蕴含计算零知识。

定义 15.9 设(P, V)为语言 L 的交互证明系统,称(P, V)为语言 L 的一个关于诚实验证者的交互计算零知识证明系统,若存在一个多项式时间概率图灵机 M,使随机变量族 $\{\mathrm{View}_{P,V}(x)\}_{x\in L}$与 $\{M(x)\}_{x\in L}$是计算不可区分的(参看定义 6.2)。

定义 15.9 是定义 12.7 的特殊情形。类似地,可定义语言 L 的关于诚实验证者的交互完全零知识证明系统和交互统计(几乎完全)零知识证明系统,它们分别是定义 15.6 和定义 15.8 的特殊情形。

15.2　交互零知识证明系统的构造

先给出几个特殊问题的交互零知识证明系统的构造。

1. 无向图的同构问题

一个无向图(以下简称图) G 由一个顶点集 V 和一个边集 E 构成,记作 $G=(V,E)$。V 中的顶点记作 $u_1,u_2,\cdots,u_{|V|}$,即 $V=\{u_1,u_2,\cdots,u_{|V|}\}$。$E$ 中的边可由一对顶点 (u,v)表示,即 $E=\{(u,v)\}$,$u,v\in V$。两个图 $G_1=(V_1,E_1)$和 $G_2=(V_2,E_2)$称为同构,若存在一个从 V_1 到 V_2 的1-1映射 π,使$(u,v)\in E_1$,当且仅当$(\pi(u),\pi(v))\in E_2$,π称为 G_1 到 G_2 的同构。任给两个图 G_1 和 G_2,问它们是否同构是一个判定问题,它所对应的语言,记作 GI,是所有同构的图对(G_1,G_2)的集,现在还不知语言 GI 是否属于 BPP,也不知它是否属于 NPC。下面来构造语言 GI 的一个非退化的交互完全零知识证明系统,它的构造思想可用一个故事来描述。

P(证明者)声称在她的迷宫的北门与南门之间有一条步行小道(穿过迷宫),V(验证者)不信,P 愿意给 V 证明她的断言,但不想给他提供任何附加知识(特别不想帮他找到一条从北门穿过迷宫到南门的小道)。P 的证明方法是与 V 一起重复进行如下的过程很多次,足以使 V 相信她的断言。

首先 P 神奇地将 V 置于她迷宫的一个随机地点,然后 V 要求 P 给他指示去北门或南门的路。V 的选择是随机的,但是他可以欺骗。于是 P 选择一个充分长的随机步行路,并引导 V 沿这条路从现在地点走到 V 要求去的门。

显然,P 的迷宫若没有一条如 P 声称的步行小道,那么 P 不可能每次都引导 V 走到他所要求去的门,至少有半数的次会走错,这就使 V 信服 P 的断言是正确的。但 V 从 P 引导的步行没有得到知识,理由是 V 自己也能模拟一个 P 引导的步行,首先他选择南门或北门如 P 引导步行前他所选的相同,然后从迷宫外走到那个门,再从所选的门进入迷宫作一次随机步行,同时从一个线轴上放线到地上,最后他沿着地上的线迹回到所选的门。与 P 引导的步行同样长的随机步行保证 V 到达迷宫的一个随机地点,回来的路就好像是从随机地点到所选门的随机步行路。

下面是语言 GI 的交互零知识证明系统的程序。

程序 GI:

1. 共同输入为两个同构的图 $G_1=(V_1,E_1)$和 $G_2=(V_2,E_2)$。设 ϕ 为 G_1 到 G_2 的同构,即 ϕ 为从 V_1 到 V_2 的1-1映射,使$(u,v)\in E_1$,当且仅当$(\phi(u),\phi(v))\in E_2$。

2. 重复执行下列3～6步 n 次。

3. P 按等概分布随机选择 V_2 的一个置换 π 并构造图 $G'=(V',E')$,其中

$$V'=\{\pi(u_1),\pi(u_2),\cdots,\pi(u_{|V_2|})\},\quad =V_2=\{u_1,u_2,\cdots,u_{|V_2|}\};$$

$E'=\{\pi(u),\pi(v)\mid(u,v)\in E_2\}$。$P$ 将 $G'=(V',E')$ 发给 V。

4. V 收到 P 发送的图 $G'=(V',E')$ 后，按等概分布随机选择一个 $\sigma\in\{1,2\}$，V 将 σ 发送给 P，要求 P 给出 G_σ 到 G 的同构。

5. 若 P 收到 V 发送的 $\sigma=2$，则 P 将 π 发送给 V；否则，即 $\sigma\neq 2$，则 P 将 $\rho=\pi\cdot\phi(\rho(u)=\pi(\phi(u)),u\in V_1)$ 发送给 V。

6. 若 V 收到 P 发送的消息，记作 Ψ，是 G_σ 到 G 的同构，则 V 输出 1(接受)；否则 V 输出 0(拒绝)。

定理 15.2 上面给出的程序 GI 是语言 GI 的一个关于不诚实验证者的交互完全零知识证明系统。更确切地说，它满足下列性质：

(1) 若 $x=(G_1,G_2)\in$GI(G_1 与 G_2 同构)，则

$$\Pr\{(P,V)(x)=1\}=1,$$

即验证者总是接受 $x\in$GI。

(2) 若 $x=(G_1,G_2)\notin$GI(G_1 与 G_2 不同构)，则对每个交互图灵机 B

$$\Pr[(B,V)(x)=0]\geq 1/2,$$

即对每一个可能的证明者 B 与 V 交互，验证者仍用 V 的程序 4 和 6，则验证者拒绝 $x\in$GI 的概率至少是 1/2。

(3) 对每个多项式时间的交互图灵机 V^*，存在一个多项式时间的概率图灵机 M^*，当输入 $x=(G_1,G_2)\in$GI 时，$\mathrm{View}_{P,V^*}(x)$ 与 $m^*(x)$ 是相同分布的随机变量(参看定义 15.6)，其中，证明者仍用 P 的程序 3 和 5。

证　性质(1)和(2)说明程序 GI 是语言 GI 的交互证明系统。证明很容易。事实上，若 $x=(G_1,G_2)\in$GI，则 π 是 G_2 到 G 的同构，ρ 是 G_1 到 G 的同构。因此按程序的 5 和 6，验证者总是接受 $x\in$GI。证得(1)。另一方面，若 $x=(G_1,G_2)\notin$GI，则没有一个图能与 G_1 和 G_2 都同构。因此无论证明者 B 发送什么图 G 给 V，G_1 和 G_2 中至少有一个图与 G 不同构，故按 V 的程序 4 和 6，验证者拒绝的概率至少是 1/2。证得(2)。

性质(3)说明程序 GI 是不诚实验证者完全零知识的。证明较困难且较长，不宜在这里给出，读者可参看本书参考文献[1]。另一方面，证明程序 GI 是诚实验证者完全零知识的并不难。这是由于在程序 GI 中，除了计算 G_1 到 G_2 的同构 ϕ 以外，其他计算都是多项式时间的，读者可作为习题自己试着证明。

2. 二次剩余问题

模 N 的二次剩余($N=PQ$，P，Q 为素数)在介绍 Rabin-Blum 函数族(见例 5.6)时遇到过，y(与 N 互素)称为一个模 N 的二次剩余，若存在正整数 x，使 $y=x^2\pmod N$。任给 N，x($N=PQ$，P，Q 为素数，x 与 N 互素)，问 x 是否为模 N 的二次剩余是一个判定问题，它所对应的语言，记作 QR，是所有的 N 与模 N 的二次剩余(N,x)的集，记 QR(N) 为所有模 N 的二次剩余的集。

下面是语言 QR 的交互零知识证明系统的程序。

程序 QR:

1. 共同输入为 N, x,其中 N 为未知因子分解的 $N=PQ$, P, Q 为素数, x 与 N 互素, $x\in \mathrm{QR}(N)$。

2. 重复执行下列 3～6 步 $\lceil \log N \rceil$ 次(N 看作二进数表示)。

3. P 按等概分布从 Z_N^* 中随机选出一个 v,计算 $y=v^2 \pmod N$, P 将 y 发送给 V。

4. V 收到 P 发送的 y 后,按等概分布随机选择一个 $\sigma\in\{0,1\}$, V 将 σ 发送给 P。

5. P 收到 V 发送的 σ 后,计算 $z=u^{\sigma}v \pmod N$,其中 u 为 x 的一个模 N 的平方根, P 将 z 发送给 V。

6. 若 V 收到 P 发送的 z 满足 $z^2\equiv x^{\sigma}y \pmod N$,则 V 输出 1(接受);否则 V 输出 0(拒绝)。

定理 15.3　上面给出的程序 QR 是语言 QR 的一个关于不诚实验证者的交互完全零知识证明系统。更确切地说,它满足下列性质:

(1) 若 $x\in \mathrm{QR}(N)$,则

$$\Pr\{(P,V)(x)=1\}=1。$$

(2) 若 $x\notin \mathrm{QR}(N)$,则对每个交互图灵机 B

$$\Pr[(B,V)(x)=0]\geq 1/2 \text{ 或 } \Pr[(P,V)(x)=1]\leq 1/2。$$

(3) 对每个多项式时间的交互图灵机 V^*,存在一个多项式时间的概率图灵机 M^*,当输入 $x\in \mathrm{QR}(N)$时, $\mathrm{View}_{P,V^*}(x)$ 与 $m^*(x)$ 是相同分布的随机变量,其中,证明者仍用 P 的程序 3 和 5。

证　容易看出,程序 QR 与程序 GI 非常相似,故定理 15.3 与定理 15.2 的证明也很相似。

由二次剩余的性质(见数学基础),若 $x\in \mathrm{QR}(N)$,按程序 5 和 6, $z^2\equiv x^{\sigma}y \pmod N$ 一定满足,故验证者总是接受 $x\in \mathrm{QR}(N)$,证得(1);若 $x\notin \mathrm{QR}(N)$,则无论证明者 B 发送什么 z 给 V, $z^2\equiv x^{\sigma}y \pmod N$ 总不可能对 $\sigma=0$ 和 1 都满足,故按 V 的程序 4 和 6,验证者拒绝 $x\in \mathrm{QR}(N)$ 的概率至少为 $1/2$。证得(2)。

性质(3)的证明不宜在此给出,它与定理 15.2 性质(3)的证明类似。

3. 子群成员问题

任给两个正整数 N, l 和两个不同元素 $\alpha,\beta\in Z_N^*$,其中, α 的阶为 l($\alpha^l=1$)。于是由 α 生成 Z_N^* 的一个子群,记作 G_α,即 $G_\alpha=\langle\alpha\rangle=\{\alpha,\alpha^2,\cdots,\alpha^l\}$。问 β 是否为 G_α 的成员是一个判定问题。这个问题与离散对数问题有关,因为问 β 是否为 G_α 的成员等价于问是否存在 $m(1\leq m\leq l)$,使 $\beta=\alpha^m$。若 m 存在, m 就是 β 的以 α 为底的离散对数。子群成员问题对应的语言,记作 SM,是所有 (N,l,α,β) 的集,其中 $\alpha,\beta\in Z_N^*$, $\alpha\neq\beta$, α 的阶为 l。

下面是语言 SM 的交互零知识证明系统的程序。

程序 SM:

1. 共同输入为 (N,l,α,β),其中 $\alpha,\beta\in Z_N^*$, $\alpha\neq\beta$, α 的阶为 l。

2．重复执行下列 3～6 步 $\lceil \log N$ 次(N 看作二进数表示)。

3．P 按等概分布从 $0,1,\cdots,l-1$ 中随机选出一个 j，计算 $r=\alpha^{j} \pmod N$，P 将 r 发送给 V。

4．V 收到 P 发送的 r 后，按等概分布随机选择一个 $\sigma\in\{0,1\}$，V 将 σ 发送给 P。

5．P 收到 V 发送的 σ 后，计算 $h=j+\sigma m \pmod N$，其中 $m=\log_{\alpha}\beta$（β 的以 α 为底的离散对数），P 将 h 发送给 V。

6．若 V 收到 P 发送的 h 满足 $\alpha^{h}\equiv\beta^{\sigma}r \pmod N$，则 V 输出 1（接受）；否则 V 输出 0（拒绝）。

定理 15.4 上面给出的程序 SM 是语言 SM 的一个关于不诚实验证者的交互完全零知识证明系统。

由于定理 15.4 的确切陈述和证明与定理 15.2 和定理 15.3 都很相似，这里不再详述，读者可自己给出。

上面三个问题的交互零知识证明系统虽有启发性，但这几个问题还不（未证明）是 NP 完全类问题。更有用的是构造 NP 完全类问题的交互零知识证明系统。只要构造了一个 NP 完全类问题的交互零知识证明系统，就可以构造任何 NP 类问题的交互零知识证明系统。这一构造方法的重要性在于它的一般性，几乎所有在密码学应用中人们希望证明的陈述都可以化为证明 NP 类语言成员的断言。这里我们只介绍图的 3 - 着色问题的交互零知识证明系统的构造方法的主要思想，图的 3 - 着色问题是一个 NP 完全类问题。由于这一构造方法基于一个工具，称为比特提交，故先介绍这一概念的直观思想。

直观地说，一个比特提交（bit commitment）方案是一个类似于保险箱的工具。P（证明者）在一纸条上写了一个二进数 $b\in\{0,1\}$（比特），将它放在保险箱里，并将保险箱交给 V（验证者）保管，即 P 已将一比特消息提交给 V 了，因为她不能再改变它。但 V 没有保险箱的密码，他不能打开它，故 V 没得到 b 的任何知识，即 P 要对她提交的比特保密。到第二阶段（一段时间后），P 将保险箱的密码告诉 V，V 打开保险箱，要求 V 从里面看到的比特只能是 P 在提交时放入的比特 b。由此可见，一个比特提交方案要满足下列两个性质：

（1）保密性（或隐藏性）。直到第一阶段结束时，V 没有得到 P 提交比特的任何知识，即使 V 进行欺骗这一性质仍然要满足。

（2）不含糊性（或连结性）。到第二阶段，P 打开她提交的比特给 V 看，V 看到的比特只能是 P 提交的比特 b，即使 P 进行欺骗这一性质仍然要满足。

现在再回到图的 3 - 着色问题。一个图 $G=(V,E)$（定义见本节 1）称为可 3 - 着色，若存在一个从 V 到 $\{1,2,3\}$ 的映射 ϕ，使 $\phi(u)\neq\phi(v)$ 对每条边 $(u,v)\in E$ 成立。ϕ 称为图 G 的 3 - 着色。任给一个图 $G=(V,E)$，问图 G 是否可 3 - 着色是一个判定问题，它所对应的语言，记作 G3C，是由所有可 3 - 着色的图 $G=(V,E)$ 构成的集。已经证明语言 G3C 是属于 NPC。构造语言 G3C 的交互零知识证明系统的主要思想概述如下：

(1) 共同输入为一可 3 - 着色的图 $G=(V,E)$。

(2) 重复执行下列 3～6 步 $|E|^2$ 次。

(3) 设 Ψ 为图 G 的一个 3 - 着色。P 按等概分布随机选择{1,2,3}的一个置换 π,并构造 $\phi(u)=\pi(\Psi(u))$, $u\in V$,即 φ 为图 G 的一个随机 3 - 着色(颜色的标记 1,2,3 是随机的)。P 用 $|V|$ 个保险箱,每个保险箱里放一个 $\phi(u)$, $u\in V$,将所有 $|V|$ 个保险箱都发送给 V(验证者)。

(4) V(验证者)按等概分布从 E 中随机选出一条边(u,v),将它发送给 P,要求检查 u 和 v 的着色。

(5) P 收到 V 发送的(u,v)后,将放有 $\phi(u)$和 $\phi(v)$的两个保险箱的密码发送给 V。

(6) V 打开这两个保险箱,若他看到的是{1,2,3}中两个不同的数,则 V 输出 1(接受);否则 V 输出 0(拒绝)。

显然,若输入的图 G 是可 3 - 着色的,则按上述协议 V 总是接受 $G\in$G3C。若输入的图 G 不是可 3 - 着色的,则无论证明者 B 用什么着色 Ψ,按 V 的程序 4 和 6,至少有一条边$(u,v)\in E$ 不满足要求,因此 V 拒绝 $G\notin$G3C 的概率至少为 $1/|E|$。二者接受的概率有明显的差别,故上述程序是语言 G3C 的一个交互证明系统。它的零知识性质也是很显然的,因为 V(验证者)自己可模拟(P,V)的交互,即从{1,2,3}中随机选出两个不同的数放在 V 指示的两个保险箱里。

由于比特提交方案是一个数字的随机加密方案,毕竟不像保险箱那样简单。关于比特提交方案以及 NP 类问题的交互零知识证明系统的一般理论,读者可参看本书参考文献[1]。

15.3　非交互零知识证明系统理论

交互零知识证明系统是 Goldwasser 等人在上个世纪 80 年代初提出的,但它对某些情况不适用,例如证明者是流动的,且他不愿将他的最新通信地址告诉验证者。为了适用这种情况,Blum 等人在上个世纪 80 年代末通过用一个共同的随机比特序列,称为参考序列,代替交互,提出了非交互零知识证明系统。这种证明系统是单向的,只需证明者发送消息给验证者。它的模型在某种意义上可看作 NP 类语言 L 的简单交互证明系统推广到允许证明者和验证者引用一个共同的随机比特序列和他们各自可进行内部扔硬币(参看定理 15.1)。不过,如前所述,NP 类语言 L 的简单交互证明系统不是零知识的。关于非交互零知识证明系统也有与交互零知识证明系统平行的理论,这里我们只对非交互零知识证明系统的定义及其构造方法的思想作一简单介绍,读者想知道详细论述可参看本书参考文献[1]。

先给出非交互证明系统的定义。

定义 15.10 一对概率图灵机(P, V)称为语言 L 的非交互证明系统，若 V 是多项式时间的，且满足下列两个条件：

(1) 完全性：对每个 $x\in L$，

$$\Pr\{V(x, R, P(x, R))=1\}\geq 2/3, \tag{15.4}$$

其中，x 为(P, V)的共同输入；R 为(P, V)的共同参考序列，它是在$\{0,1\}^{p(|x|)}$ 中等概分布的随机序列，$p(n)$为任一固定的正多项式；$P(x, R)$为 P 发送给 V 的消息(P 的输出和 V 的输入)。

(2) 合理性：对每个 $x\notin L$ 和每个交互图灵机 B，

$$\Pr\{V(x, R, B(x, R))=0\}\geq 2/3 \text{ 或 } \Pr\{V(x, R, B(x, R))=1\}\leq 1/3, \tag{15.5}$$

其中 $V(x, R, B(x, R))=1$，表示验证者接受 $x\in L$，$V(x, R, B(x, R))=0$ 表示验证者拒绝 $x\in L$。

与定义 15.5 一样，在定义 15.10 的(15.4)和(15.5)式中的常数 2/3 可换成 $1-2^{-p(|x|)}$，并不影响非交互证明系统的定义，其中 $p(n)$为任一固定的正多项式，也可将 2/3 和 1/3 换成两个有明显差别的常数。

对于零知识的定义，非交互证明系统比交互证明系统更简单，因为验证者不能影响证明者的程序，因此只需考虑指定验证者 V 的观察的可模拟性。

定义 15.11 一个语言 L 的非交互证明系统(P, V)称为是计算零知识的，若存在一正多项式 $p(n)$和一个多项式时间概率图灵机 M，使随机变量族 $\{M(x, U_{p(|x|)}, P(x, U_{p(|x|)}))\}_{x\in L}$ 与 $\{M(x)\}_{x\in L}$ 是计算不可区分的。

非交互零知识证明系统的构造方法是先引入一个过渡性的证明系统，称为隐比特证明系统。由隐比特证明系统变换为非交互证明系统是容易的。隐比特证明系统和非交互证明系统的主要区别是只有证明者才能看到共同参考序列的全部。证明者发送给验证者的消息(证明)包括两部分：第一部分称为证明书，第二部分是共同参考序列中某些指定的比特位置。验证者只能检查证明者指定的位置上的比特，当然验证者还可检查共同输入和证明书。

定义 15.12 一对概率图灵机(P, V)称为语言 L 的隐比特证明系统，若 V 是多项式时间的，且满足下列两个条件：

(1) 完全性：对每个 $x\in L$，

$$\Pr\{V(x, R_I, I, \mathrm{Cer})=1\}\geq 2/3$$

其中，$(I, \mathrm{Cer})=P(x, R)$为 P 发送给 V 的消息(P 的输出和 V 的输入)，Cer 称为证明书，$I=\{i_1, \cdots, i_\tau\}\subseteq\{1, 2, \cdots, p(|x|)\}$为 R 的指定位置集，称为泄漏的比特位置集，x 为(P, V)的共同输入，$R=r_1\cdots r_t$ 是在$\{0,1\}^{p(|x|)}$ 中等概分布的随机序列，$R_I=r_{i_1}\cdots r_{i_\tau}$为 R 在指定位置集 I 上的子序列，称为泄漏的比特序列。

(2) 合理性：对每个 $x\notin L$ 和每个概率图灵机 B，

$$\Pr\{V(x, R_I, \mathrm{I}, \mathrm{Cer}) = 0\} \geq 2/3 \text{ 或 } \Pr\{V(x, R_I, I, \mathrm{Cer}) = 1\} \leq 1/3,$$

其中，$(I, \mathrm{Cer}) = \mathrm{B}(x, R)$是 B 发送给 V 的消息(B 的输出和 V 的输入)。

定义 15.13　一个语言 L 的隐比特证明系统(P, V)称为是计算零知识的，若存在一个多项式时间概率图灵机 M，使随机变量族$\{(x, R_I, P(x, R))\}_{x\in L} = \{(x, R_I, I, \mathrm{Cer})\}_{x\in L}$与$\{\mathrm{M}(x)\}_{x\in L}$是计算不可区分的。

利用一个单向置换 $f:\{0,1\}^* \to \{0,1\}^*$ (f 是保长的和 1-1 映射的)以及 f 的一个硬核谓词 $b:\{0,1\}^* \to \{0,1\}$(可参看定义 5.8 和定义 5.9)，这里 b 可看作单向置换族$\{f_n:\{0,1\}^n \to \{0,1\}^n; n\geq 1\}$的硬核谓词族$\{b_n:\{0,1\}^n \to \{0,1\}; n\geq 1\}$，可将一个隐比特零知识证明系统变换为一个非交互零知识证明系统，其变换方法如下：

设(P, V)为一个语言 L 的隐比特证明系统，$f:\{0,1\}^* \to \{0,1\}^*$ 和 $b:\{0,1\}^* \to \{0,1\}$都是多项式时间可计算的，其中 f 是保长的和 1-1 的，记 $m = p(n)$为(P, V)的共同输入 x 的长$|x| = n$所对应的参考序列 R 的长$(p(n)$为任一固定的正多项式$)$。今构造一个语言 L 的非交互证明系统(P', V')如下：

1．共同输入 $x\in\{0,1\}^n$。

2．共同参考序列为 $s = \{s_1, \cdots, s_m\}$，其中每个 $s_i \in \{0,1\}^n$。

3．证明者(记作 P')的程序。

(1) 对每个 $i\in\{1, \cdots, m\}$，计算 $r_i = b(f^{-1}(s_i))$。

(2) 向 P 要 $\mathrm{P}(x, r_1, \cdots, r_m) = (I, \mathrm{Cer})$。

(3) 输出(I, Cer, p_I)(P'发送给 V 的消息)，其中，$I = \{i_1, \cdots, i_\tau\}$，$p_1 = (f^{-1}(s_{i_1}) \cdots f^{-1}(s_{i_\tau})) = (p_1 \cdots p_\tau)$。

4．验证者(记作 V')的程序

(1) 输入 P 输出的(I, Cer, P_I)后，对每个 $i_j \in I$，检验 $s_{i_j} = f(p_j)$是否成立。若发现有一个不成立，则 V 停止和拒绝。

(2) 计算 $r_i = b(p_i)$，$i = 1, \cdots, \tau$，记 $r = (r_1, \cdots, r_\tau)$。

(3) 向 V 要输出 $V(x, r, I, \mathrm{Cer})$作为 V 的输出，即 V 接受当且仅当 V 接受。

容易证明，上面构造的(P', V')是一个语言 L 的非交互计算零知识证明系统，若(P, V)为一个语言 L 的隐比特计算零知识证明系统以及 b 为单向置换 f 的硬核谓词。

下一步的问题是要构造一个 NP 完全问题的隐比特零知识证明系统。由于每个 NP 类问题都可多项式时间化为 NP 完全类问题(参看定义 4.12)，故只要构造了一个 NP 完全类问题的隐比特零知识证明系统，就可构造任何 NP 类问题的隐比特零知识证明系统。再用前面介绍的变换方法就可构造任何 NP 类问题的非交互零知识证明系统。这与交互零知识证明系统的理论是完全平行的。

下面介绍另一个 NP 完全类问题，有向图的哈密尔顿(Hamilton)问题的隐比特零知识证明系统的构造。一个有向图 G 也是由一个顶点集 V(不妨设 $V = \{1, 2, \cdots, n\}$)和一

个边集 E 构成,记作 $G=(V,E)$。不过 E 中的边要由一对有序顶点(u,v)表示,u 称为边的起点,v 称为边的终点。若干条边 $e_1=(u_1,v_1)$,$e_2=(u_2,v_2)$,…,$e_m=(u_m,v_m)$,若 $v_i=u_{i+1}$,$i=1,2,\cdots,m-1$,则它们可连接起来,称为路,记作$(e_1,e_2,\cdots,e_m)$;若更有 $v_m=u_1$,则称$(e_1,e_2,\cdots,e_m)$为一回路,或称为圈。经过 V 中每个顶点但仅经过一次的圈$(e_1,e_2,\cdots,e_n)$称为有向图 $G=(V,E)$的一个哈密尔顿圈。任给一个有向图 $G=(V,E)$,问 G 是否包含一个哈密尔顿圈是一个判定问题(哈密尔顿问题),它所对应的语言,记作 HC,是所有包含哈密尔顿圈的有向图 G 构成的集。

在构造证明系统之前先引进几个要用到的关于矩阵的概念:(1)一个置换 $\pi:\{1,2,\cdots,n\}\to\{1,2,\cdots,n\}$可以表示为一个 $n\times n$ 矩阵,它的 i 行 $\pi(i)$列的元素为 $1(1\leq i\leq n)$,其他元素均为 0,称为置换矩阵。(2)一个哈密尔顿圈$(e_1,e_2,\cdots,e_n)$,其中 $e_i=(u_i,v_i)(1\leq i\leq n)$,可由一个置换矩阵表示,它所对应的置换 π 定义为 $v_i=\pi(u_i)$,$i=1,2,\cdots,n$,称为哈密尔顿矩阵。(3)一个 $n^3\times n^3$ 矩阵 $\boldsymbol{M}$ 称为有用,若 $\boldsymbol{M}$ 中包含一个 $n\times n$ 子矩阵 $\boldsymbol{H}$ 是哈密尔顿矩阵,而其他 n^6-n^2 个元素都是 0。

语言 HC 的隐比特零知识证明系统。

1. 共同输入 $x=G=(V,E)\in$HC,其中$|V|=n$。

2. 共同参考序列看作一个 $n^3\times n^3$ 矩阵 $\boldsymbol{M}$,其元素为 1 的概率为 n^{-5},元素为 0 的概率为 $1-n^{-5}$。

3. 证明者(记作 P)的程序。设 C 为 $\boldsymbol{G}$ 中的一个哈密尔顿圈。

(1) 证明者考察矩阵 $\boldsymbol{M}$,分如下两种情况:

情况一:$\boldsymbol{M}$ 为有用。记 $\boldsymbol{H}$ 为 $\boldsymbol{M}$ 中的哈密尔顿 $n\times n$ 子矩阵,C_H 为 $\boldsymbol{H}$ 对应的哈密尔顿圈。

(2) 证明者泄漏 $\boldsymbol{M}$ 中除 $\boldsymbol{H}$ 以外的所有 n^6-n^2 个元素。

(3) 证明者计算出(ϕ_1,ϕ_2),其中 ϕ_1 为 V 到 $\boldsymbol{H}$ 的行的 1-1 映射,ϕ_2 为 V 到 $\boldsymbol{H}$ 的列的 1-1 映射,使 G 中的哈密尔顿圈 C 上的边映射到 $\boldsymbol{H}$ 中的元素 1,即将任一有序顶点对 (u,v),$u,v\in V$,映射到 $\boldsymbol{H}$ 的$\phi_1(u)$行 $\phi_2(v)$列的元素。若$(u,v)\in C$,则 $\boldsymbol{H}$ 的$\phi_1(u)$行 $\phi_2(v)$列的元素为 1,即映射(ϕ_1,ϕ_2)是 C 到 C_H 的一个同构。

(4) 证明者泄漏所有$(u,v)\notin E$ 对应的 $\boldsymbol{H}$ 的 $\phi_1(u)$行 $\phi_2(v)$列的 $n^2-|E|$个元素。

(5) 证明者输出(I,Cer),M_I(P 发送给 V(验证者)的消息),其中,$\mathrm{Cer}=(\phi_1,\phi_2)$,$I$ 为 P 泄漏的 $\boldsymbol{M}$ 中所有 $n^6-|E|$个元素的位置,M_I 为 P 泄漏的 $\boldsymbol{M}$ 中所有 $n^6-|E|$个元素,都是 0。

情况二:$\boldsymbol{M}$ 不为有用。

(6) 证明者只输出(I,M_I)(没有 Cer),其中 I 为 $\boldsymbol{M}$ 的所有 n^6 个元素的位置,$\mathrm{M}_I=\boldsymbol{M}$。

4. 验证者(记作 $\boldsymbol{V}$)的程序。

(1) 验证者输入证明者发送的$(I, \mathrm{Cer}, \mathrm{M}_I)$,检查 M 中除了$\phi_1(u)$行 $\phi_2(v)$的元素,$(u, v) \in E$ 外,其他元素是否都包含在 M_I 中,且都为 0,若是,则接受,否则拒绝。

(2) 验证者输入 M,检查 M 是否为有用,若不为有用则接受,否则拒绝。

可以证明上面给出的(P, V)的程序是一个语言 HC 的隐比特零知识证明系统。

综上所述我们得到下面的定理:

定理 15.5　若单向置换存在,则每个 NP 类语言存在一个非交互零知识证明系统。

注 记

零知识证明是构造安全(密码学)协议的重要工具。本章主要介绍了交互零知识证明系统和非交互零知识证明系统的一些基本知识。本章主要参考了本书参考文献[1],[15]和[26]。关于更一般和更严格的零知识证明理论可参看本书参考文献[1],限于篇幅,本章没有给出比特提交的正式定义,有些定理的较难证明部分的证明也没有给出,读者可参看本书参考文献[1]。比特提交的非正式描述及例子可参看文献[15]的 13.3 节,关于零知识证明理论的综述性文章可参看本书参考文献[26]第 3 章所引的文献[34],[37],[38]。

习题十五

1. 证明在定义 15.5 的(15.1)式和(15.2)式中的常数 2/3 可换成 $1 - 2^{-|x|^p}$ 并不影响交互证明系统的定义,其中 $p(n)$为任一固定的正多项式。

2. 证明若语言 L 有一个交互证明系统满足定理 15.1 的两个条件,则 $L \in \mathrm{NP}$。

3. 证明一个语言 L 的交互完全零知识证明系统一定是语言 L 的一个交互统计零知识证明系统。

4. 设 $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$为两个无向图,其中 $V_1 = V_2 = \{1,2,3,4\}$, $E_1 = \{(1,2),(1,3),(1,4),(3,4)\}$, $E_2 = \{(1,2),(1,3),(2,3),(2,4)\}$, $\Phi: (1,2,3,4) \to (2,4,3,1)$为 G_1 到 G_2 的一个同构,故 $x = (G_1, G_2) \in \mathrm{GI}$。

按照语言 GI 的交互零知识证明系统的程序,给出验证者 V 的观察 $\mathrm{View}_{P,V}(x)$(用随机变量表示),再给出一个通常的概率算法(用 +, -, ×, ÷, =, >, <)模拟随机变量 $\mathrm{View}_{P,V}(x)$。

5. 对二次剩余问题做与第 4 题相同的工作,即按照语言 QR 的交互零知识证明系统的程序,给出验证者的观察 $\mathrm{View}_{P,V}(x)$和模拟 $\mathrm{View}_{P,V}(x)$的通常的概率算法,其中设 $x \in \mathrm{QR}$。

6. 概述构造语言 HC(有向图的哈密尔顿圈问题)的交互零知识证明系统的主要思

想,其中用保险箱代替比特提交方案。

7. 给出语言 G3C(无向图的 3-着色问题)的隐比特零知识证明系统。

8. 阅读练习:在读懂文献[1]中比特提交方案的正式定义的基础上,给出语言 G3C 的正式的交互零知识证明系统(用数字的比特提交方案代替保险箱)。

附录　数学基础

在现代密码学中,需要使用到许多数学理论,例如数论、信息论、复杂度理论、组合论、概率及线性代数等等数学理论,均为设计密码系统及协议不可或缺的工具。在第 3 章和第 4 章中,分别介绍过密码学中涉及到的信息理论和复杂度理论的基础知识。本附录将对现代密码学中必要的数学基础做一重点整理,以便读者能很快地了解现代密码学中大部分系统的工作原理及如何分析和证明其安全性。一些特殊,但在密码学中也非常重要的数学原理,将在相关章节中再作介绍。

1. 代数基础

这里简要介绍 3 种最基本的代数结构:群、环、域。

1.1　群

定义 1.1　设 G 是非空集合,并在 G 内定义了一种代数运算,若满足下述公理:

(1) 有封闭性,对任意 $a, b \in G$,恒有 $a \quad b \in G$;

(2) 结合律成立,对任意 $a, b \in G$,有 $(a \quad b) \quad c = a \quad (b \quad c)$;

(3) G 中有一恒等元(单位元) e 存在,对任意 $a \in G$,有 $a \quad e = e \quad a = a$;

(4) 对任意 $a \in G$,存在有 a 的逆元 $a^{-1} \in G$,使 $a \quad a^{-1} = a^{-1} \quad a = e$,则称 G 构成一个群。

上述定义中,G 的运算‘　’可以是通常的乘法或加法。对乘法群而言,恒等元常记作 1;对加法群而言,则恒等元常记为 0; a 的逆元记为 $-a$。群中元素的个数,称为群的阶。若群中元素个数有限,称为有限群;否则,称为无限群。

若群 G 中,对任何 $a, b \in G$,有 $a \quad b = b \quad a$,则称 G 为交换群或加法群或 Abel 群。

例 1.1　整数集 **Z** 关于普通的加法形成一个 Abel 群,恒等元是 0, a 的逆元是 $-a$。

例 1.2　剩余类集 Z_n 关于模 n 加法运算形成一个阶为 n 的 Abel 群, Z_n 关于模 n 的

乘法运算不是一个群,因为不是所有的元素都有乘法逆。然而,集 $Z_n^* = \{a \in N: \gcd(a, n) = 1, 1 \leq a < n\}$ 关于模 n 的乘法运算形成一个阶为 $\phi(n)$($\phi()$ 为欧拉函数,见定义 2.4)的 Abel 群,恒等元为 1。

定义 1.2 设 H 是 G 的一个非空子集,如果 H 本身在群 G 的运算 之下构成一个群,我们说 H 是 G 的一个子群。如果 H 是 G 的一个子群且 $H \neq G$,则称 H 是 G 的一个真子群。

定义 1.3 设 G 是一个群,如果 G 中有一个元素 α 使得对每一个 $b \in G$,都存在一个整数 i,使得 $b = \alpha^i$,则称 G 是一个循环群,α 称为 G 的一个生成元。

定义 1.4 设 G 是一个群,$a \in G$,a 的阶定义为使得 $a^t = 1$ 的最小正整数 t(假定这样的正整数存在的话)。如果这样的正整数 t 不存在,那么 a 的阶定义为∞。

易知,如果 G 是一个群,$a \in G$,则$\{a^k \mid k \in \mathbf{Z}\}$形成 G 的一个循环子群,称作由 a 生成的子群,记为$\langle a \rangle$。如果 a 的阶为 t,则$|\langle a \rangle| = t$。

关于有限群有下列重要的基本定理:

定理 1.1(Lagrange 定理) 设 G 是一个有限群,H 是 G 的一个子群,则$|H| \mid |G|$,因此,如果 $a \in G$,则 a 的阶整除$|G|$。

关于循环群有以下基本性质:

(1) 循环群的子群也是循环群。如果 G 是一个阶为 n 的循环群,那么对 n 的每一个正因子 d,G 恰好包含一个阶为 d 的子群。

(2) 设 G 是一个群,如果 $a \in G$ 的阶为 t,则 a^k 的阶是 $t/\gcd(t, k)$。

(3) 设 G 是一个阶为 n 的循环群,$d \mid n$,则 G 恰好有 $\phi(d)$ 个阶为 d 的元素。特别地,G 有 $\phi(n)$ 个生成元。

1.2 环

定义 1.5 一个环$(R, +, \times)$是由两个满足下列条件的 R 上的二元运算“ + ”(称为加法)和“ × ”(称为乘法)构成:

(1) $(R, +)$是一个 Abel 群,恒等元用 0 表示;

(2) 运算“ × ”是可结合的,即是说,对所有的 $a, b, c \in R$,有 $a \times (b \times c) = (a \times b) \times c$;

(3) 有一个乘法恒等元 1,$1 \neq 0$,使得对所有的 $a \in R$,有 $a \times 1 = 1 \times a = a$;

(4) 乘法对加法的分配律,即对所有的 $a, b, c \in R$,有 $a \times (b + c) = (a \times b) + (a \times c)$和$(b + c) \times a = (b \times a) + (c \times a)$。

如果一个环$(R, +, \times)$还满足条件:对所有的 $a, b \in R$,有 $a \times b = b \times a$,则称环$(R, +, \times)$为交换环。

例 1.3　整数集 **Z** 关于通常的加法和乘法运算形成一个交换环。

例 1.4　剩余类环 Z_n 关于模 n 加法和乘法形成一个交换环。

定义 1.6　设 R 是一个环，$a, b \in R$，且 $a \neq 0$，$b \neq 0$，但 $a \times b = 0$，则称 a, b 为零因子，称有零因子的环为有零因子环。

有零因子环中消去律不一定成立，如$(Z_6, +, \times)$中运算 $2 \times 1 = 2 \times 4$，但 $1 \neq 4$。

1.3　域

定义 1.7　满足下列条件的一个交换环称为一个域：所有的非零元素都有乘法逆。

可见域是一个非常完备的代数系统，其中的元素要求满足较多的性质或约束。

定义 1.8　设 F 是一个域，如果对任何 $m \geq 1$，，$1 + 1 + \cdots + 1 \neq 0$（$m$ 个 1 相加），则称 F 的特征为 0；否则，F 的特征是使得 $\sum_{i=1}^{m} 1 = 0$ 的最小正整数 m。

例 1.5　整数环 Z 不是一个域，因为只有 1 和 - 1 有乘法逆。然而，有理数集 **Q**、实数集 **R** 和复数集 **C** 在通常的加法和乘法运算下形成特征为 0 的域。

定理 1.2　Z_n 是一个域，当且仅当 n 是素数。如果 n 是素数，则 Z_n 的特征是 n。

关于域的特征有以下重要事实：如果一个域的特征不是 0，则它的特征必是素数。

定义 1.9　设 E 是一个域，F 是 E 的一个子集，如果 F 关于 E 的运算本身形成一个域，则称 F 为 E 的子域，也称 E 为 F 的扩域。

根据域所包含的元素是否有限，将域分为无限域和有限域。包含有限个元素的域称为有限域；否则称为无限域。域 F 中的元素个数也称为有限域 F 的阶。

在大多数的计算机工程领域中（包括密码学），人们感兴趣的是有限域。有限域常以数学家 Galois 的名字命名，称作 Galois 域，并以 GF(q)表示，其中 q 表示有限域的阶。

关于有限域，有以下的重要结果：

(1)（有限域的存在性和惟一性）如果 F 是一个有限域，那么 F 包含素数幂个元素，即存在素数 p 和整数 $m \geq 1$，使得 $|F| = p^m$；反之，对每一个素数幂 p^m，（在同构的意义下）存在惟一的一个阶为 p^m 的有限域。将这个域记为 F_{p^m} 或 GF(p^m)。

(2)（有限域的子域）设 F_q 是一个阶为 $q = p^m$ 的有限域，p 是素数，则 F_q 的每个子域有阶 p^n，且 $n \mid m$。反之，如果 $n \mid m$，则 F_q 恰有一个阶为 p^n 的子域，而且 $a \in F_q$ 是 F_{p^n} 的一个元素，当且仅当 $a^{p^n} = a$。

(3) $F_q^* = F_q \setminus \{0\}$ 关于乘法形成一个阶为 $q - 1$ 的循环群。因此，对所有的 $a \in F_q$，有 $a^q = a$。这个群称为 F_q 的乘法群，乘法群 F_q^* 的生成元称为 F_q 的本原元，共有 $\varphi(q - 1)$ 个本原元。

(4) 设 F_q(其中 $q=p^m$)是一个有限域，p 是一个素数，$m\geq 1$，则 F_q 的特征为 p，而且对所有的 $a, b\in F_q$ 和 $t\geq 0$，有 $(a+b)^{p^t}=a^{p^t}+b^{p^t}$。

1.4 多项式环

系数在整数环 **Z**、有理数域 **Q**、实数域 **R** 或复数域 **C** 上的一元多项式的全体形成一个环。类似地可以定义系数属于一般域 F 上的多项式。

定义 1.10 多项式

$$f(x)=a_0+a_1x+\cdots+a_nx^n,\ a_i\in F,\ i=0,1,2,\cdots,n,$$

如果 $a_n\neq 0$，就说 $f(x)$是域 F 上的 n 次多项式，记作 $\deg f=n$，并称 a_n 为 $f(x)$的首项系数。当 $f(x)$的所有系数都是 0 时，就说 $f(x)$是零多项式，仍用 0 表示之，约定 $\deg(0)=-\infty$。

记 F 上全体多项式所成之集为 $F[x]$。现在定义 $F[x]$中的加法和乘法运算。设

$$f(x)=\sum_{i=0}^{n}a_ix^i,\ g(x)=\sum_{i=0}^{m}b_ix^i\in F[x],$$

令 $M=\max(n,m)$，当 $n<m$ 时，约定 $a_{n+1}=a_{n+2}=\cdots=a_m=0$。当 $n>m$ 时约定 $b_{m+1}=b_{m+2}=\cdots=b_n=0$。此时 $f(x)$ 和 $g(x)$ 可分别写成

$$f(x)=\sum_{i=0}^{M}a_ix^i,\ g(x)=\sum_{i=0}^{M}b_ix^i。$$

定义加法运算：$f(x)+g(x)=\sum_{i=0}^{M}(a_i+b_i)x^i$，显然，$f(x)+g(x)\in F(x)$。再令 $a_{n+1}=a_{n+2}=\cdots=a_{n+m}=0$，$b_{m+1}=b_{m+2}=\cdots=b_{n+m}=0$，此时，$f(x)$ 和 $g(x)$ 可分别写成

$$f(x)=\sum_{i=0}^{m+n}a_ix^i,\ g(x)=\sum_{i=0}^{m+n}b_ix^i。$$

定义乘法运算：$f(x)\cdot g(x)=\sum_{i=0}^{m+n}\sum_{j=0}^{i}a_jb_{i-j}x^i$，显然，$f(x)\cdot g(x)\in F(x)$。

容易验证，$F[x],+,\cdot$ 是一个交换环，通常称之为多项式环。

例 1.6 设二元域上多项式为 $f(x)=x^3+x+1$，$g(x)=x^2+x\in Z_2[x]$，则

$$f(x)+g(x)=x^3+x^2+1,\ f(x)\cdot g(x)=x^5+x^4+x^3+x。$$

多项式环 $F[x]$与整数环 **Z** 非常类似，有许多共同的特性。

对应于整数环 **Z** 中的素数的概念，多项式环 $F[x]$中引入了不可约多项式的概念，它的作用类似于素数的作用。

定义 1.11 一个多项式 $p(x)\in F[x]$称为 F 上的不可约多项式，是指 $\deg p\geq 1$，而

$p(x)$在 F 上仅有平凡分解,即如果有一个分解 $p(x)=f(x)g(x)$,其中 $f(x),g(x)\in F[x]$,则

$$\deg f=0 \text{ 或 } \deg g=0。$$

因此,除了一个常数倍数外,$p(x)$的因式只有自身和 1。

定理 1.3　(带余除法)设 $f(x),g(x)\in F[x],g(x)\neq 0$,则 $F[x]$中存在惟一的一对多项式 $q(x)$和 $r(x)$使得 $f(x)=q(x)g(x)+r(x),\deg(r(x))<\deg(g(x))$。多项式 $q(x)$称作商式,$r(x)$称作余式,通常记 $r(x)=f(x)\bmod g(x)$。当 $r(x)=0$ 时,我们说 $g(x)$整除 $f(x)$,并称 $g(x)$是 $f(x)$的因式,或 $f(x)$是 $g(x)$的倍式,记为 $g(x)\mid f(x)$。当 $r(x)\neq 0$ 时,我们就说 $g(x)$不整除 $f(x)$,记为 $g(x)\nmid f(x)$。

例 1.7　设 $f(x)=x^6+x^5+x^3+x^2+x+1,g(x)=x^4+x^3+1\in Z_2[x]$,则

$$f(x)=x^2g(x)+x^3+x+1,\text{因此},f(x)\bmod g(x)=x^3+x+1。$$

类似于整数环的情形,可在 $F[x]$中定义最大公因式的概念。如果 $h(x)\in F[x]$是 $f(x)$的因式,又是 $g(x)$的因式,我们说是 $f(x)$和 $g(x)$的公因式。我们把 $f(x)$和 $g(x)$的公因式中次数最大的而且首项系数为 1 的公因式 $d(x)$叫做 $f(x)$和 $g(x)$的最大公因式,记为 $\gcd(f(x),g(x))$。约定 $\gcd(0,0)=0$。

当 $\gcd(f(x),g(x))=1$ 时,我们就说 $f(x)$和 $g(x)$互素。像整数环中一样,求两个不全为零的多项式的最大公因式也有相应的 Euclidean 算法和扩展的 Euclidean 算法,这里不再赘述。平行于算术基本定理,在 $F[x]$中也有相应的定理,称为惟一因式分解定理。

定理 1.4　$F[x]$中任一正次数的多项式可分解为 $F[x]$中有限个首项系数为 1 的不可约多项式与 F 中常数之积;并且这些不可约多项式是惟一确定的(如不计它们在乘积中的次序)。

最后我们回顾多项式剩余类或同余类环。

定义 1.12　设 $f(x),g(x),m(x)\in F[x],m(x)\neq 0$,如果 $m(x)\mid(f(x)-g(x))$,则称 $f(x)$和 $g(x)$模 $m(x)$同余,记作 $f(x)\equiv g(x)\pmod{m(x)}$。

多项式的同余式和整数中的同余式有着类似的性质。“模 $m(x)$同余”是 $F[x]$中的一个等价关系。按此关系将 $F[x]$分类,$f(x)$的同余类是 $F[x]$中所有与 $f(x)$模 $m(x)$同余的多项式所构成的集合。与整数中情形不同的是,模 $m(x)$同余类的类数不必是有限的。

用 $F[x]/(m(x))$表示 $F[x]$中次数小于 $n=\deg m(x)$的全体多项式之集,也就是模 $m(x)$的等价类之集。$F[x]/(m(x))$在模 $m(x)$的加法和乘法下形成一个交换环,称为多项式剩余类环。进一步,如果 $m(x)$是 F 上不可约多项式,则模 $m(x)$的同余类中还可以作除法,从而形成一个域。这是代数学中构作域的基本手法。

定理 1.5　如果 $m(x)\in F[x]$在 F 上不可约,则 $F[x]/(m(x))$是一个域。

由定理 1.5 可知,如果取 $F=Z_p$(p 为素数),$m(x)$是一个 n 次不可约多项式,则

$F[x]/(m(x))$是一个阶为 p^n 的有限域。这表明,可用不可约多项式来构造密码学中常用的有限域。那么不可约多项式是否存在呢?人们已经证明,对任意的有限域 F 和任意的正整数 n, $F[x]$中一定存在 n 次不可约多项式;不仅如此,人们还给出了不可约多项式的精确计数公式。

2 数论基础

2.1 素数与互素数

我们知道整数集合中除了加法和乘法之外还可以作减法运算,但是一般不能作除法,由此引出初等数论中第一个基本概念:数的整除性。

定义 2.1 设 a 和 b 是整数, $b\neq 0$,如果存在整数 c 使得 $a=bc$,则称 b 整除 a,表示成 $b\mid a$,并称 b 是 a 的因子,而 a 为 b 的倍数。如果不存在上述的整数 c,则称 b 不整除 a,表示成 $b\nmid a$。

由整除的定义,立即导出整除的如下的基本性质:

(1) $b\mid b$;

(2) 如果 $b\mid a$, $a\mid c$,则 $b\mid c$;

(3) 如果 $b\mid a$, $b\mid c$,则对任意整数 x, y,有 $b\mid(ax+cy)$;

(4) 如果 $b\mid a$, $a\mid b$,则 $b=\pm a$。

定理 2.1 (带余除法)设 a 和 b 是整数, $b\neq 0$,则存在整数 q, r 使得

$$a=bq+r,\text{其中 } 0\leq r<|b|;$$

并且整数 q, r 由上述条件惟一决定。式中,整数 q 称为 a 被 b 除的商,数 r 叫做 a 被 b 除得的余数。

定义 2.2 设 $a, b, \cdots, c$ 是有限个不全为零的整数,满足下面两个条件(惟一)的整数 d 称为它们的最大公约数,记作$(a, b, \cdots, c)$或 $\gcd(a, b, \cdots, c)$:

(1) d 是 $a, b, \cdots, c$ 公共约数,即 $d\mid a$, $d\mid b, \cdots, d\mid c$;

(2) d 是 $a, b, \cdots, c$ 的所有公约数中最大的,即如果整数 d_1 也是 $a, b, \cdots, c$ 的公约数,则 $d_1\leq d$。

任意整数 $a, b, \cdots, c$ 必然有公约数(例如± 1)。如果它们不全为零,则易知它们的公约数只有有限多个,所以它们的最大公约数必然存在并且是惟一的。此外,最大公约数一定是正整数。

如果$(a, b, \cdots, c)=1$,则称 $a, b, \cdots, c$ 是互素的。如果 $a, b, \cdots, c$ 中任意两个是互素的,则称两两互素。

最大公约数有下面性质:

(1) 设 $a, b, \cdots, c$ 是不全为零的整数，则存在整数 $x, y, \cdots, z$，使得 $ax + by + \cdots + cz = (a, b, \cdots, c)$。特别地，如果 $a, b, \cdots, c$ 互素，则存在整数 $x, y, \cdots, z$，使得 $ax + by + \cdots + cz = 1$。

(2) 设 $(a, b, \cdots, c) = d$，则 $\frac{a}{d}, \frac{b}{d}, \cdots, \frac{c}{d} = 1$;

(3) 设 $(a, m) = (b, m) = 1$，则 $(ab, m) = 1$;

(4) 如果 $c \mid ab$，且 $(c, b) = 1$，则 $c \mid a$。

关于求两个整数最大公约数的问题，有欧几里德(Euclidean)算法解决(参见附录 2.3 节)。

定义 2.3　设 p 为大于 1 的整数，如果 p 没有真因子，即 p 的正约数只有 1 和 p 自身，则称 p 为素数，否则称为合数。

关于素数，有以下一些事实:

(1) 素数有无穷多个。

(2) 设 p 是素数，$a, b, \cdots, c$ 是整数。如果 p 整除乘积 $ab\cdots c$，则 $a, b, \cdots, c$ 中至少有一个被 p 整除。

(3) (素数定理)设 $\pi(x)$ 表示不大于 x 的素数的数目，则 $\lim_{x\to\infty} (\pi(x)\ln x)/x = 1$。素数定理表明，对充分大的 x，$\pi(x)$ 可用 $x/\ln x$ 来近似表示。

(4) (算术基本定理)每个整数 $n \geq 2$，均可分解成素数幂之积

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}。$$

若不计因子的顺序，这个分解式是惟一的。其中 $p_i (1 \leq i \leq k)$ 是不同的素数，$e_i (1 \leq i \leq k)$ 是正整数。

2.2　同余及模运算

令三整数 a, b 及 n，我们称 a 在模 n 时与 b 同余，当且仅当 a 与 b 的差为 n 的整数倍，即 $a - b = kn$，其中 k 为任一整数。若 a 与 b 在模 n 中同余，我们可写成 $a \equiv b \bmod n$。

由上述定义可知:若 a 与 b 在模 n 中同余，则 n 必整除 a 与 b 的差，即 n 整除 $(a - b)$，在符号上我们可写成 $n \mid (a - b)$。

很明显地，利用同余概念，所有整数在模 n 中被分成 n 个不同的剩余类;为 n 所整除的数为一剩余类;以 n 除余数为 1 的数为一剩余类，余 2 的数为一剩余类，依此类推。若将每一剩余类中取一数为代表，形成一集合，则此集合称为模 n 的完全剩余系，以 Z_n 表示，显然 Z_n 对于加法形成一个加法群。很明显地，集合 $\{0, 1, 2, \cdots, n - 1\}$ 为模 n 的一完全剩余系。

在同余的基本运算中，存在以下的基本定理。

定理 2.2 $a = a \bmod n$(自反性)。

定理 2.3 若 $a = b \bmod n$,则 $b = a \bmod n$(对称性)。

定理 2.4 若 $a = b \bmod n$ 且 $b = c \bmod n$,则 $a = c \bmod n$(传递性)。

定理 2.5 若 $a = b \bmod n$ 且 $c = d \bmod n$,则 $a + c = b + d \bmod n$,

$a - c = b - d \bmod n$, $ac = bd \bmod n$。

定理 2.6
$$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$$
$$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$$
$$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$$

定理 2.7 若 $ac = bd \bmod n$ 且 $c = d \bmod n$ 及 $(c, n) = 1$,则 $a = b \bmod n$((c, n)表示 c 和 n 的最大公因子,$(c, n) = 1$ 表示 c 与 n 互素)。

证 由 $(a - b)c + b(c - d) = ac - bd = 0 \bmod n$,可得 $n \mid (a - b)c$。

因为 $(c, n) = 1$,故得 $n \mid (a - b)$。因此 $a = b \bmod n$。

(注意:定理 2.7 中,若 c 与 n 不互素,则此定理不成立。)

例如,$3 \times 2 = 1 \times 2 \bmod 4$ 且 $2 = 2 \bmod 4$,但 $3 \neq 1 \pmod 4$。

在模 n 的完全剩余系中,若将所有与 n 互素的剩余类形成一集合,则此集合称为模 n 的既约剩余系,以 Z_n^* 表示。例如 $n = 10$ 时,$\{0,1,2,3,4,5,6,7,8,9\}$为模 10 的完全剩余系;而$\{1,3,7,9\}$为模 10 的既约剩余系。在模 n 中取既约剩余系的原因,为在模 n 的既约剩余系中取一整数 a,则必存在另一整数 b(属于此既约剩余系)使得 $ab = 1 \bmod n$ 且此解惟一。例如,

a	0	1	2	3	4	5	6	7	8	9
b	×	1	×	7	×	×	×	3	×	9

其中“×”表示“无意义”。

若 $ab = 1 \bmod n$,则称 b 为 a 在模 n 的乘法逆元,b 可表示为 a^{-1}。定理 2.7 说明上述的叙述为正确。

定理 2.8 若 $(a, n) = 1$,则存在惟一整数 b, $0 < b < n$,且 $(b, n) = 1$,使得 $ab = 1 \bmod n$。

证 由定理 2.7 知,若 $(a, n) = 1$,且 $i \neq j \bmod n$,则 $ai \neq aj \bmod n$。因此,集合$\{ai \bmod n\}_{i = 1, \cdots, n-1}$为集合$\{1, 2, \cdots, n - 1\}$的一排列(Permutation)。因此必存在 $1 \leq b \leq n - 1$,使得 $ab = 1 \bmod n$,即 b 为 $ab = 1 \bmod n$ 的惟一解。此外,因 $ab - 1 = kn$, k 为整数,若 $(b, n) = g$,则 $g \mid (ab - 1)$。因为 $g \mid ab$,所以 $g \mid 1$。因此 $g = 1$。故 b 也与 n 互素。

定义 2.4 令 $\phi(n)$为小于 n,且与 n 互素的所有正整数的个数,即 $\phi(n)$为模 n 既约剩余系中所有元素的个数,此 $\phi(n)$称为欧拉函数(Euler Totient Function)。

定理 2.9 令 $r_1, r_2, \cdots, r_{\phi(n)}$ 为模 n 的一既约剩余系,且 $(a, n) = 1$,则 $ar_1, ar_2, \cdots, ar_{\phi(n)}$ 也为模 n 的一既约剩余系。

证　设$(ar_j, n) = g > 1$,则 $g \mid a$ 或 $g \mid r_j$。因此我们得以下两种情况:

(1) $g \mid a$ 且 $g \mid n$,或(2) $g \mid r_j$ 且 $g \mid n$。

(1) 不可能,因为$(a, n) = 1$; (2)也不可能,因为 r_j 为模 n 既约剩余系的一元素。因此$(ar_j, n) = 1$。此外,若 $r_i \neq r_j$,则 $ar_i \neq ar_j$。因此 $ar_1, ar_2, \cdots, ar_{\Phi(n)}$ 为模 n 的一既约剩余系。

定理 2.10　欧拉定理(Euler's Theorem)

若$(a, n) = 1$,则 $a^{\Phi(n)} = 1 \bmod n$。

证　令$\{r_1, r_2, \cdots, r_{\Phi(n)}\}$为模 n的既约剩余系,由定理2.9知若$(a, n) = 1$,则$\{ar_1, ar_2, \cdots, ar_{\Phi(n)}\}$ 也为一既约剩余系。因此, $\prod_{i=1}^{\Phi(n)} (ar_i) \bmod n = \prod_{i=1}^{\Phi(n)} r_i \bmod n$。故得

$$a^{\Phi(n)} \bmod n \prod_{i=1}^{\Phi(n)} r_i \bmod n = \prod_{i=1}^{\Phi(n)} r_i \bmod n。$$

由消去法可得 $a^{\Phi(n)} = 1 \bmod n$。

例 2.1　$\{1,3,5,7\}$为模 8 的一既约剩余系,3 与 8 互素,因此由定理 2.10, $3^4 = 3^{\Phi(8)} = 1 \bmod 8$。

定理 2.11　费尔马定理(Fermat's Theorem)

令 p 为素数,且$(a, p) = 1$,则 $a^{p-1} = 1 \bmod p$。

证　若 p 为素数,则 $\Phi(p) = p - 1$,由欧拉定理可得证。

例 2.2　设 p 是素数,则$(p - 1)! = -1 \bmod p$。

证　$p = 2$ 时结论显然成立。设 $p \geq 3$,由于对模 p 的既约剩余系中任一元素 a,存在惟一逆元 a^{-1},满足 $a \cdot a^{-1} = 1$;而 $a = a^{-1}$等价于 $a^2 = 1$,即 $a = 1$ 或 $a = p - 1$,于是剩下的 $p - 3$ 个同余类 $2, 3, \cdots, p - 2$ 可按互逆配成$(p - 3)/2$ 个对。因此

$$(p - 1)! = 1 \cdot 2 \cdot \cdots \cdot (p - 2)(p - 1) = p - 1 = -1 \bmod p。$$

2.3　乘法逆元的求法

已给 a 及 n 且$(a, n) = 1$,如何求 $aa^{-1} = 1 \bmod n$?

方法一　若 $\Phi(n)$已知,则由欧拉定理可知 $aa^{\Phi(n)-1} = 1 \bmod n$,因此, $a^{\Phi(n)-1} = a^{-1} \bmod n$。

显然,若 n 为素数,则 $\Phi(n) = n - 1$;若 n 为合数,则 $\Phi(n)$不一定容易计算。

方法二　利用欧几里得算法。

我们首先回顾欧几里得算法如何求两整数 a, n 的最大公因子 $\gcd(a, n)$。

令 $r_0 = a, r_1 = n, n \leq a$,利用辗转相除法可得:

用 r_1 除 r_0: $r_0 = r_1 q_1 + r_2 \quad 0 \leq r_2 < r_1$,

用 r_2 除 r_1：$r_1 = r_2 q_2 + r_3 \quad 0 \leq r_3 < r_2$，

…

用 r_{m-1} 除 r_{m-2}：$r_{m-2} = r_{m-1} q_{m-1} + r_m \quad 0 \leq r_m < r_{m-1}$，

用 r_m 除 r_{m-1}：$r_{m-1} = r_m q_m$。

事实上，由于诸余数 $r_0, r_1, \cdots$ 为整数，且满足

$$r_0 > r_1 > \cdots > r_{m-1} > \cdots \geq 0,$$

从而上述的带余除法有限步后余数必为零。另一方面，

$$(a, n) = (r_0, r_1) = (r_1, r_2) = \cdots = (r_{m-1}, r_m) = (r_m, 0) = r_m。$$

欧几里得算法不仅可以求出 (a, n)，还可以求出方程 $sa + tn = (a, n)$ 的一组整数解（注意 s, t 并非惟一）。具体做法是将欧氏算法倒推回去：

由算法的倒数第二行，得到 $(a, n) = r_m = r_{m-2} - r_{m-1} q_{m-1}$，这就将 (a, n) 表示成 r_{m-2}, r_{m-1} 的整系数线性组合。再用算法中其前面的一行 $r_{m-1} = r_{m-3} - r_{m-2} q_{m-2}$ 代入上式，消去 r_{m-1}，得出 $(a, n) = (1 + q_{m-1} q_{m-2}) r_{m-2} - q_{m-1} r_{m-3}$，即 (a, n) 为 r_{m-2}, r_{m-3} 的线性组合，如此进行，最终可得 $(a, n) = sa + tn$。若 $(a, n) = 1$，则 $1 = sa + tn$。所以 $sa = 1 \bmod n$，因此 $s = a^{-1} \bmod n$。

通常方法二效率较高。

模 n 的完全剩余系 Z_n 在加法中为一交换群，且单位元素为 0。模 n 的既约剩余系 Z_n^* 在乘法中为一交换群。当 n 为素数时，Z_n^* 为 Z_n 中所有非零元素集合，因此，若 n 为素数时，Z_n 为一有 n 个元素的有限域。

2.4 线性同余

已给整数 a, b 及 $n > 0$，下式称为单变量线性同余式

$$ax = b \bmod n。$$

其中 x 为变量。下面的定理告诉我们上式是否有解，及若有解时解的个数。

定理 2.12 令 a, b 及 n 为整数，且 $n > 0$ 及 $(a, n) = d$。

(1) 若 $d \nmid b$，则 $ax = b \bmod n$ 无解。

(2) 若 $d \mid b$，则 $ax = b \bmod n$ 恰好有 d 个模 n 不同余的解。

证 由定义知，线性同余式等价于求两变量 x 及 y 满足 $ax - ny = b$。整数 x 为 $ax = b \bmod n$ 的一个解，当且仅当存在整数 y，使得 $ax - ny = b$。当 $d \nmid b$ 时，因 $d \mid ax$ 及 $d \mid yn$，使得 $d \mid (ax - yn) = b$，矛盾，故当 $d \nmid b$ 时，$ax - ny = b$ 无解。当 $d \mid b$ 时，$ax = b \bmod n$ 有无限多个解。因为若 x_0 及 y_0 为解时，所有 $x = x_0 + \frac{n}{d}t$，$y = y_0 +$

$\frac{a}{d}^{t}$,均为其解,其中 t 为任意整数。但上述解中只有 d 个模 n 的不同余类,因为 $\frac{n}{d}$ t mod n 中只有 d 个不同的同余类,即 t = 0,1,2,…, d - 1。

由本定理知,若$(a, n) = 1$,则 $ax = b \bmod n$ 有惟一解。

定理 2.12 只告诉我们线性同余式 $ax = b \bmod n$ 是否有解,及若有解时,有多少个解。以下我们介绍当有解时,如何求出其解:

(1) 利用欧几里德算法求出$(a, n) = d$,若 $d \nmid b$,则上式无解;

(2) 若 $d \mid b$,则令 $a' = \frac{a}{d}$, $b' = \frac{b}{d}$, $n' = \frac{n}{d}$,则 $a'x' = b' \bmod n'$有惟一解,因为$(a', n') = 1$。

此解可以由欧几里德算法求出。例如,先求 a'为模 n'的乘法逆元 a'^{-1},令 $x' = a'^{-1} b' \bmod n'$即为其解。接着令 $x_0 = x' \bmod n$,则 x_0 即为 $ax = b \bmod n$ 的一个解。令 $x = x_0 + \frac{n}{d} t \bmod n$, $t = 0, 1, 2, \cdots, d - 1$,则所有 d 个解均可求出。

例 2.3　求解 $24x = 7 \bmod 59$。

解　由于$(24, 59) = 1$,从而方程有惟一的解

$$x = \frac{7}{24} = \frac{7 + 59}{24} = \frac{11}{4} = \frac{-48}{4} = -12 \bmod 59。$$

例 2.4　求解 $9x = 12 \bmod 15$。

解　(1) $(9, 15) = 3$ 且$3 \mid 12$,故有 3 个解。

(2) 求解 $3x' = 4 \bmod 5$,由于 $3 \cdot 2 = 1 \bmod 5$,故 $3^{-1} = 2 \bmod 5$。所以 $x' = 2 \cdot 4 = 3 \bmod 5$。令 $x = x_0 = 3 \bmod 15$ 且 $x = x_0 + 5 = 8 \bmod 15$, $x = x_0 + 5 \cdot 2 = 13 \bmod 15$,此为所有 3 个解。

2.5　中国剩余定理(Chinese Remainder Theorem)

定理 2.13　令 $n_1, n_2, \cdots, n_t$ 为两两互素的正整数,令 $N = n_1 n_2 \cdots n_t$,则以下同余系统中, $x = a_1 \bmod n_1$, $x = a_2 \bmod n_2$, …, $x = a_t \bmod n_t$ 会在$[0, N - 1]$中有惟一解。

证　由于 $n_1, n_2, \cdots, n_t$ 两两互素,故对所有 $i = 1, 2, \cdots, t$, $n_i, \frac{N}{n_i} = 1$。因此,存在 y_i,使得 $\frac{N}{n_i} y_i = 1 \bmod n_i$。此外, $\frac{N}{n_i} y_i = 0 \bmod n_j$,当 $j \neq i$,这是由于$\frac{N}{n_i}$为 n_j 的整数倍。若我们令

$$x = \frac{N}{n_1} y_1 a_1 + \frac{N}{n_2} y_2 a_2 + \cdots + \frac{N}{n_t} y_t a_t \bmod N = \sum_{i=1}^{t} \frac{N}{n_i} y_i a_i \bmod N,$$

则 x 为上述同余系统的解,因为对于所有

$$i, 1 \leq i \leq t, \ x \bmod n_i = \frac{N}{n_i} y_i a_i \bmod n_i = a_i。$$

若上述同余系统有两个解为 x 及 z,则对所有 i,$1 \leq i \leq t$,满足 $x = z = a_i \bmod n_i$,故 $n_i \mid (x - z)$。因此 $N \mid (x - z)$,即 $x = z \bmod N$。因此,此系统有在 $[0, N - 1]$ 中有惟一解。

注　中国剩余定理最早记载于第一世纪的孙子算经中,为线性同余式的起源,故名之为中国剩余定理。其原问题为:今有物不知其数,三三数之剩二,五五数之剩三,七七数之剩二,问物几何?

即求正整数 x 满足 $x = 2 \bmod 3$, $x = 3 \bmod 5$, $x = 2 \bmod 7$。

例 2.5　求 x 满足上式。

解　$N = 3 \cdot 5 \cdot 7 = 105$, $\frac{N}{n_1} = \frac{105}{3} = 35$, $\frac{N}{n_2} = 21$, $\frac{N}{n_3} = 15$,

所以由 $35 y_1 = 1 \bmod 3$,得 $y_1 = 2$;由 $21 y_2 = 1 \bmod 5$,得 $y_2 = 1$;由 $15 y_3 = 1 \bmod 7$,得 $y_3 = 1$。故 $x = 35 \cdot 2 \cdot 2 + 21 \cdot 1 \cdot 3 + 15 \cdot 1 \cdot 2 = 23 \bmod 105$。

中国剩余定理在密码学上有非常重要的应用。例如在 RSA 解密时,利用中国剩余定理,可以使解密速度加快 4 倍。下面介绍在求解 $t = 2$ 时较快速的方法,此方法较定理 2.14 的标准解法,可节约约一半的时间。

例 2.6　已给 a_1, a_2, n_1, n_2 且 $n_1 < n_2$,及 $(n_1, n_2) = 1$,求 x 使得 $0 \leq x \leq n_1 n_2$,满足 $x = a_1 \bmod n_1$, $x = a_2 \bmod n_2$。

解　首先求出 u 满足 $u \cdot n_2 = 1 \bmod n_1$,接着

(1) 若 $a_1 \geq (a_2 \bmod n_1)$,则

$$x = (((a_1 - (a_2 \bmod n_1)) \times u) \bmod n_1) \times n_2 + a_2。$$

(2) 若 $a_1 < (a_2 \bmod n_1)$,则

$$x = (((a_1 + n_1 - (a_2 \bmod n_1)) \times u) \bmod n_1) \times n_2 + a_2。$$

2.6　二次剩余(Quadratic Residue)

二次剩余在概率密码系统、素数测试、因子分解等,均占有很重要的地位。在此,我们介绍其数学基础,其应用将于各章节中讨论。

定义 2.5　令 n 为正整数,若整数 a,$(a, n) = 1$,满足 $x^2 = a \bmod n$ 有解,则称 a 为模 n 的二次剩余。否则称 a 为模 n 的二次非剩余。

我们通常以符号 QR_n 表示所有模 n 的二次剩余的集合;以 QNR_n 表示所有模 n 的二次非剩余的集合。

例 2.7　若 $n = 7$,模 n 的二次剩余有 $\{1^2, 2^2, 3^2, 4^2, 5^2, 6^2\} \bmod 7 = \{1, 2, 4\}$,其余 $\{3, 5, 6\}$ 为二次非剩余,因为无法找到整数解满足 $x^2 = a \bmod 7$, $a \in \{3, 5, 6\}$。

定理 2.14　令 p 为奇数,且 $0 < a < p$,则

(1) 若 $a \in \mathrm{QR}_p$,则 $x^2 = a \bmod p$ 有两个解。若 $a \in \mathrm{QNR}_p$,则无解。

(2) 令 p 为奇素数,则 $|\mathrm{QR}_p| = \frac{p-1}{2}$,且 $|\mathrm{QNR}_p| = \frac{p-1}{2}$。其中 $|\mathrm{QR}_p|$ 表示集合 QR_p 的元素个数。

证　(1)若 $a \in \mathrm{QR}_p$,则 $x^2 = a \bmod p$ 至少有一解 x_1。但 $p - x_1$ 也为其解,因为 $(p - x_1)^2 = (p^2 - 2px_1 + x_1^2) \bmod p = x_1^2 = a \bmod p$,且对于 $p > 2$, $p - x_1 \neq x_1$,易见同余方程也只有这两个解。

(2) 明显地,$1^2, 2^2, \cdots, \left(\frac{p-1}{2}\right)^2 \bmod p$ 均为二次剩余。此外,并无其他二次剩余。因为,若 $a \in \mathrm{QR}_p$,则至少其平方根 x_1,或 $p - x_1$,必落在 $1, \frac{p-1}{2}$ 范围中。

定理 2.15　若 p 为奇素数,且 $0 < a < p$,则

$$a^{(p-1)/2} \bmod p = \begin{cases} 1 & \text{若 } a \in \mathrm{QR}_p \\ -1 & \text{若 } a \in \mathrm{QNR}_p \end{cases}$$

证　由费尔马定理知 $a^{p-1} - 1 = 0 \bmod p$。若 p 为奇数,则上式可分解为 $(a^{(p-1)/2} + 1)(a^{(p-1)/2} - 1) = 0 \bmod p$,这意味着素数 p 可整除 $a^{(p-1)/2} + 1$ 或 $a^{(p-1)/2} - 1$,即 $a^{(p-1)/2} \bmod p$ 或 $a^{(p-1)/2} = -1 \bmod p$。若 $a \in \mathrm{QR}_p$,则存在 x,使得 $x^2 = a \bmod p$,因此, $a^{(p-1)/2} = 1 \bmod p = x^{2(p-1)/2} = x^{p-1} \bmod p = 1 \bmod p$。故 $a^{(p-1)/2} = 1 \bmod p$ 有 $\frac{p-1}{2}$ 个解。但因其次数为 $\frac{p-1}{2}$,故最多仅有 $\frac{p-1}{2}$ 个解。其余的 $\frac{p-1}{2}$ 个二次非剩余,必为 $a^{(p-1)/2} = -1 \bmod p$ 的解。

定义 2.6　勒让德符号(Legendre Symbol)

若 p 为奇素数,且 a 为正整数,勒让德符号 $\frac{a}{p}$ 定义为

$$\frac{a}{p} = \begin{cases} 1 & \text{if } a \in \mathrm{QR}_p \\ -1 & \text{if } a \in \mathrm{QNR}_p \\ 0 & \text{if } p \mid a \end{cases}$$

注　由定理 2.15,可求出 $\frac{a}{p}$;若 p 不是素数,则称雅可比符号。

定义 2.7　雅可比符号(Jacobi Symbol)

令 n 为奇正整数,且 $n = p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m}$。其中 p_i 为素数, $t_i > 0$。 a 为正整数,则雅可比符号定义为 $J\left(\frac{a}{n}\right) = J\left(\frac{a}{p_1^{t_1} p_2^{t_2} \cdots p_m^{t_m}}\right) = \left(\frac{a}{p_1}\right)^{t_1} \left(\frac{a}{p_2}\right)^{t_2} \cdots \left(\frac{a}{p_m}\right)^{t_m}$。

以下是雅可比符号的性质。这里设 n 为奇正整数,且 a 与 b 为正整数。

性质 1　当 n 为奇素数时, $J\left(\frac{a}{n}\right) = \left(\frac{a}{n}\right)$。

性质 2 $J\left(\frac{1}{n}\right)=1, J\left(\frac{-1}{n}\right)=(-1)^{\frac{n-1}{2}}$。

性质 3 $J\left(\frac{ab}{n}\right)=J\left(\frac{a}{n}\right)J\left(\frac{b}{n}\right)$。

性质 4 $J\left(\frac{2}{n}\right)=(-1)^{\frac{n^2-1}{8}}$。**即若** $n=1 \bmod 8$ **或** $n=7 \bmod 8$, **则** $J\left(\frac{2}{n}\right)=1$; **若** $n=3 \bmod 8$ **或** $n=5 \bmod 8$ **则**, $J\left(\frac{2}{n}\right)=-1$。

性质 5 $J\left(\frac{b}{a}\right)=J\left(\frac{b \bmod a}{a}\right)$。

性质 6 a, b **为正奇数**, **若**$(a, b)=1$, **则** $J\left(\frac{a}{b}\right)J\left(\frac{b}{a}\right)=(-1)^{\frac{(a-1)(b-1)}{4}}$。

由于勒让德符号为雅可比符号的特殊情况(n **为奇素数**), **故上述性质也存在于勒让德符号中。由勒让德符号的定义及定理** 2.15 **可知, 求勒让德符号只需一次指数模乘法。由雅可比符号定义知, 若我们能分解因子** $n=p_1^{t_1}p_2^{t_2}\cdots p_m^{t_m}$, **使** n **成为所有素数的乘积, 则求雅可比符号只需最多** m **次指数模乘法。事实上, 利用性质** 3 **及性质** 6, **我们可以更快速地求得雅可比符号** $J\left(\frac{a}{n}\right)$。

问题 1 **若我们无法分解因子** n, **是否我们仍能快速地求出** $J\left(\frac{a}{n}\right)$? **定理** 2.17 **告诉我们其答案是肯定的。**

定理 2.16 **令** a, n **为正整数, 且**$(a, n)=1$, **则** $J\left(\frac{a}{n}\right)$ **可以在** $O((\log_2 n)^3)$**位运算中**(**相当于指数运算**)**求出。有关符号** O **的定义请参阅复杂性理论一章。**

性质 3 **提供了一个很重要的事实: 设** p **为奇素数, 令** $c=ab \pmod p$, **则若**

(1) $a \in \mathrm{QR}_p$ **且** $b \in \mathrm{QR}_p$, **则** $c \in \mathrm{QR}_p$;

(2) $a \in \mathrm{QR}_p$ **且** $b \in \mathrm{QNR}_p$, **则** $c \in \mathrm{QNR}_p$;

(3) $a \in \mathrm{QNR}_p$ **且** $b \in \mathrm{QR}_p$, **则** $c \in \mathrm{QNR}_p$;

(4) $a \in \mathrm{QNR}_p$ **且** $b \in \mathrm{QNR}_p$, **则** $c \in \mathrm{QR}_p$。

问题 2 **若** $J\left(\frac{a}{n}\right)=1$, **是否表示** $x^2=a \bmod n$ **均有解?**

此问题的回答为否定的。(**注意, 若** $\frac{a}{n}=1$, **则答案是肯定的**)。**其原因为, 若** $J\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{t_1}\left(\frac{a}{p_2}\right)^{t_2}\cdots\left(\frac{a}{p_m}\right)^{t_m}=1$, **并不表示所有** $\left(\frac{a}{p_i}\right)^{t_i}=1$, $i=1,2,\cdots,m$。**例如** $J\left(\frac{2}{15}\right)=\frac{2}{3}\ \frac{2}{5}=(-1)(-1)=1$。**但因** $x^2=2 \bmod 3$, **及** $x^2=2 \bmod 5$ **均无解, 故** $x^2=2 \bmod 15$ **也无解。例如** $J\left(\frac{1}{15}\right)=1$, **但因** $x^2=1 \bmod 3$, **及** $x^2=1 \bmod 5$ **均有两个解。故** $x^2=1 \bmod 15$ **有** 4 **个解:** $x=1$, $x=4$, $x=11$, $x=14$。

当 $n = pq$, p 及 q 均为大素数时,为计算机密码学中最重要的应用。例如 RSA 系统即为此种情况。当 $n = pq$ 时的二次剩余,有以下重要定理。

定理 2.17　设 p 及 q 均为奇素数且 $n = pq$,令 $a \in Z_n^*$,则

(1) a 为模 n 的二次剩余,当且仅当 a 为模 p 的二次剩余,且 a 为模 q 的二次剩余。

(2) 若 a 为模 n 的二次剩余,则 $z^2 = a \bmod n$ 在 Z_n^* 中有 4 个解,表示为$\{x, n - x, y, n - y\}$。我们通常将$\{x, n - x\}$记为一组解,$\{y, n - y\}$记为另一组解。

(3) 若 a 为模 n 的二次剩余,且 p 及 q 均为已知,则 $z^2 = a \bmod n$ 的 4 个解均可在多项式时间内求出。

(4) 若 a 为模 n 的二次剩余,则若我们已知 $z^2 = a \bmod n$ 两组解中的任一组解,则 n 可在多项式时间内分解 p 及 q。

(证明略)

定理 2.18　已给 $n = pq$, p 及 q 为两大素数且为未知,若有一算法 AL,可在任给模 n 的二次剩余 a 时,求出其任一平方根满足 $x^2 = a \bmod n$,则 AL 可在多项式时间内分解因子 n。

(证明略)。

设 $n = pq$,为两奇素数的乘积,若 $x \in Z_n^*$,由于雅可比符号

$$J\left(\frac{x}{n}\right) = J\left(\frac{x}{p}\right) J\left(\frac{x}{q}\right) = \left(\frac{x}{p}\right)\left(\frac{x}{q}\right),$$

因此,当 $J\left(\frac{x}{n}\right) = 1$,则 $\left(\frac{x}{p}\right) = 1$,且 $\left(\frac{x}{q}\right) = 1$,或 $\left(\frac{x}{p}\right) = -1$,且 $\left(\frac{x}{q}\right) = -1$。

当 $J\left(\frac{x}{n}\right) = -1$,则 $\left(\frac{x}{p}\right) = 1$,且 $\left(\frac{x}{q}\right) = -1$,或 $\left(\frac{x}{p}\right) = -1$,且 $\left(\frac{x}{q}\right) = 1$,

所以,依雅可比符号可以被划分为下列 4 类,如下表所示:

$J\left(\frac{x}{p}\right)$	$J\left(\frac{x}{q}\right)$	$Q_{ij}(n)$	$J\left(\frac{x}{n}\right)$
1	1	$Q_{00}(n)$	1
-1	-1	$Q_{11}(n)$	1
1	-1	$Q_{01}(n)$	-1
-1	1	$Q_{10}(n)$	-1

在上述分类中,只有 $x \in Q_{00}(n)$ 时,x 才是模 n 的二次剩余。若 x 属于其他三类时,则 x 为二次非剩余。由于 $J\left(\frac{x}{n}\right) = -1$ 可在不需知 p 及 q 情况下于多项式时间内求解。因此,当 $x \in Q_{01}(n)$ 或 $Q_{10}(n)$ 时我们可在多项式时间内判断其为二次非剩余。若 $x \in Q_{00}(n)$ 或 $Q_{11}(n)$,则我们无法在多项式时间内判断 x 是否为二次剩余,除非我们能求出 p 及 q。

由雅可比符号的性质3,若 $n = pq$ 且 $p, q > 2$,则有以下结果:

(1) 对于 $x, y \in Z_n^*$, $xy \bmod n$ 为模 n 的二次剩余,当且仅当 $x, y \in Z_n^*$ 属于相同的 $Q_{ij}(n)$;

(2) 两个二次剩余的乘积,仍为二次剩余;

若 $n = pq$,且 $p = 3 \bmod 4$ 及 $q = 3 \bmod 4$,则 n 称为 Blum 整数。当 n 为 Blum 整数时,其二次剩余具有以下有趣的性质(详细证明请参见本书参考文献[30]):

(1) $J\left(\frac{-x}{n}\right) = J\left(\frac{x}{n}\right)$。

(2) 若 $x, y \in Z_n^*$,满足 $x^2 = y^2 \bmod n$,且 $x \neq y$ 或 $-y, \bmod n$ 则 $J\left(\frac{x}{n}\right) = -J\left(\frac{y}{n}\right)$。

(3) 若 a 为模 n 的二次剩余,则 $x^2 = a \bmod n$ 的4个解恰好在 $Q_{00}(n)$, $Q_{11}(n)$, $Q_{10}(n)$, $Q_{01}(n)$中各有一解。

2.7 有限域 GF p^m 的运算

当 p 为素数时,任何整数 $a \in Z_p^*$,必存在一乘法逆元 $a^{-1} \in Z_p^*$,使得 $aa^{-1} = 1 \bmod p$。故在模 p 的同余运算中,Z_p 为一有限域,通常以 GF(p)表示。数学家已证明,一有限域,GF(q),其 q 必为 p 或 p^m(p 为素数), $m > 1$。在此,我们介绍 GF(p^m)的运算方式。

令 $a \in$GF(p^m),则 a 可表示成下列次数为 $m-1$ 或更小的多项式:

$$a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1 x + a_0。$$

其中系数 a_i 为模 p 的整数。每一元素 a 均为模 $f(x)$的余数, $f(x)$为一次数为 m 系数为模 p 的整数的不可约多项式。

1. GF(p^m)的加法

令 $a, b \in$GF(p^m),且 $a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1 x + a_0$, $b = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1 x + b_0$,则 $c = a + b = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_1 x + c_0$,其中 $c_i = a_i + b_i \bmod p, 0 \leq i \leq m-1$。

例 2.8 在 $GF(3^3)$中,若 $a = 2x^2 + x + 2$, $b = 2x^2 + 2x + 2$,则 $c = x^2 + 1$。

2. GF(p^m)中乘法

若 $a, b \in$GF(p^m), a 与 b 的乘积与一般多项式的乘积相同。其差别为,若其积的阶数等于或比 m 大时,必须以不可约多项式 $f(x)$除之。故若 $d = ab$,则 d 可表示为

$$d = \sum_{i=0}^{m-1} (a_i b) x^i \bmod f(x)。$$

例 2.9 如上例中设 $f(x) = x^3 + 2x + 1$, $d = ab = (2x^2 + x + 2)(2x^2 + 2x + 2)$。

(1) 先求: $ab = x^4 + x^2 + 1$。

(2) 再除以 $f(x) = x^3 + 2x + 1$,得

$$d = 2x^2 + 2x + 1。$$

由于 GF(p^m)为一有限域,因此若 $a \in GF(p^m)$,且不为零元素,则 a 的乘法逆元存在,即存在 $a^{-1} \in GF(p^m)$,使得 $aa^{-1} = 1 \bmod f(x)$。如同 GF(p)中的运算,在 GF(p^m)中也有两种求乘法逆元的方法。

3. GF(p^m)的乘法逆元

我们以 $a = 2x^2 + x + 2$ 和 $f(x) = x^3 + 2x + 1$ 为例子说明这两种方法。

方法一　欧几里德算法

$x^3 + 2x + 1 = (2x + 2)(2x^2 + x + 2) + 2x;$

$2x^2 + x + 2 = (x + 2)(2x) + 2;$

$2x = x(2),$

故 $2 = (2x^2 + x + 2) - (x + 2)(2x) =$

$(2x^2 + x + 2) - (x + 2)[(x^3 + 2x + 1) - (2x + 2)(2x^2 + x + 2)] =$

$(2x^2 + 2)(2x^2 + x + 2) - (x + 2)(x^3 + 2x + 1)$。

两边乘以 2 得

$4 = (x^2 + 1)(2x^2 + x + 2) - (2x + 1)(x^3 + 2x + 1)$,即

$$(x^2 + 1)(2x^2 + x + 2) = 1 \bmod (x^3 + 2x + 1)。$$

所以 $a = 2x^2 + x + 2$ 的乘法逆元 a^{-1} 为 $x^2 + 1$。

方法二　本方法是由定理 2.19,因 $a^{3^3 - 1} \bmod f(x) = 1$,故

$$a^{-1} = a^{3^3 - 2} = a^{25} \bmod f(x) = (x^2 + 1)。$$

定理 2.19　若 $a \in GF(p^m)$,且 $a^t \bmod f(x) = 1$,则 $t \mid p^m - 1$。

证明略。

2.8　在 GF(p)中求解平方根的方法

以前提到,若 $n = pq$ 为两大素数的乘积,且 a 为模 n 的二次剩余,欲求 x 满足 $x^2 = a \bmod n$,其困难度等于分解因子。但我们欲求 x 满足 $x^2 = a \bmod p$, p 为素数,则 x 可在多项式时间内求出。若 $p = 3 \bmod 4$,则下述的确定式方法一可求出 x:

方法一　输入: 素数 $p(= 3 \bmod 4)$及模 p 的二次剩余 a。

输出: x 满足 $x^2 = a \bmod p$。

步骤(1): 求 $x = a^{\frac{p+1}{4}} \bmod p$。

步骤(2): 输出 x。

验证 x 的正确性:

$$x^2 = a^{\frac{p+1}{4}\cdot 2} a^{\frac{p+1}{2}} = a^{\frac{p-1}{2}} \cdot a \bmod p,$$

由于 $a \in QR_p$,推出 $a^{\frac{p-1}{2}} = 1 \bmod p$,故 $x^2 = a \bmod p$。

方法二　输入:素数 p 及模 p 的二次剩余 a。

输出: x 满足 $x^2 = a \bmod p$。

步骤(1):任选一整数 r,若 $r^2 = a \bmod p$,输出 $x = r$。

步骤(2):计算 $(r + y)^{\frac{p-1}{2}} = u + vy \bmod (f(y) = y^2 - a)$。

步骤(3):若 $u = 0$,则输出 $x = v^{-1} \bmod p$,否则重复步骤(1)。

上述步骤(2)中 y 为一符号,相当于在复数中的 i, $(b + cy)(d + ey) = (bd + cey^2) + (be + cd)y = (bd + cea) + (be + cd)y \bmod y^2 - a$。

由于 a 为模 p 的二次剩余,因此 $f(y) = y^2 - a = (y + x)(y - x)$。故 $f(y)$并不是不可约多项式,因此,其运算虽与 $GF(p^2)$类似,但在 $f(y)$中并不是有限域。有人证明,每一次任选 r,本算法可求出 x 的概率,大于或等于 $1/2$。

例 2.10　求 x 使其满足 $x^2 = 12 \bmod 13$。

解　(1) 任选 $r = 3, 3^2 = 9 \neq 12$,故进行步骤(2)求 $(3 + y)^{\frac{13-1}{2}} = (3 + y)^6 = 12 + 0y \bmod (y^2 - 12)$。因为 $12 \neq 0$,故重复步骤(1)。

(2) 任选 $r = 7, 7^2 = 10 \neq 12$,故进行步骤(2)求 $(7 + y)^6 = 0 + 8y \bmod (y^2 - 12)$,故 $x = 8^{-1} = 5 \bmod 13$。

参 考 文 献

[1] Goldreich O .Foundations of Cryptography: Basic tools .Cambridge: Cambridge University Press, 2001

[2] Shannon C E .Communication Theory of Secrecy System .Bell Syst Tech J .1949, 28: 656～715 .

[3] Shannon C E .A Mathematical Theory of Communication .Bell Syst Tech J .1948, 27: 423, 623～656 .

[4] Diffie W: Hellman M .New Directions in Cryptography .IEEE Trans on Info Theory . 1976, IT-22(6): 644～654

[5] Kahn D .The Codebreakers: The Story of Secret Writing . New York: Macmillan Publishing Co, 1967

[6] Steve Gee .Basic Methods of Cryptography . Cambridge: Cambridge University Press, 1998

[7] 王育民,何大可 .保密学——基础与应用 .西安:西安电子科技大学出版社, 1990

[8] 卢开澄 .计算机密码学——计算机网络中的数据保密与安全 .北京:清华大学出版社, 1998

[9] 王育民,刘建伟 .通信网的安全——理论与技术 . 西安:西安电子科技大学出版社, 1999

[10] 章照止,杨义先,马晓敏 .信息理论密码学的新进展及研究问题 .电子学报 .1998, 26 (7): 9～18

[11] 余成波 .信息理论与编码 .重庆:重庆大学出版社, 2002

[12] 陈志平,徐宗布 .计算机数学——计算复杂性理论与 NPC、NP 难问题的求解 .北京: 科学出版社, 2001

[13] Neal Koblitz . A Course in Number Theory and Cryptography .Beijing: World Publishing Corp, 1994

[14] Arto Salomaa .公钥密码学 .丁存生,单炜娟译 .北京:国防工业出版社, 1998

[15] Stinson D R .Cryptography: Theory and Practice .Boca Raton: CRC Press, 1995

[16] Michael Luby .Pseudorandomness and Cryptographic Applications .Princeton: Princeton University Press, 1996

[17] Carl Pomerance (Editor) .Cryptology and Computational Number Theory .Rhode Islamd: American Mathematical Society, 1990

[18] 朱文余,孙琦 .计算机密码应用基础 .北京:科学出版社, 2000

[19] 陈彦学 .信息安全理论与实务 .北京:中国铁道出版社,2001
[20] 黄元飞,陈麟 唐三平 .信息安全与加密解密核心技术 .上海:浦东电子出版社,2001
[21] 温巧燕,钮心忻,杨义先 .现代密码学中的布尔函数 .北京:科学出版社,2000
[22] 冯登国,密码分析学 . 北京:清华大学出版社,2000
[23] 冯登国,吴文玲 .分组密码的设计与分析 . 北京:清华大学出版社,2000
[24] 杨义先,孙伟,钮心忻 .现代密码新理论 .北京:科学出版社,2002
[25] 孙晓蓉,王育民 .密钥托管加密系统的分类 .通信保密 .1997,32(3):32～36
[26] 冯登国,裴定一 .密码学导引 .北京:科学出版社,1999
[27] NIST .Escrowed Encryption Standard .Federal Information Processing Standards Publication 185 .U .S Dept of Commerce,1994
[28] Schneier B .应用密码学——协议、算法与 C 源程序 .吴世忠,等译 .北京:机械工业出版社,2000
[29] Kaveh Pahlavan .无线网络通信原理与应用 .刘剑,安晓波,李春生,等译 .北京:清华大学出版社,2002
[30] 赖溪松,韩亮,张真诚,张玉清 .计算机密码学及其应用 .肖国镇改编 .北京:国防工业出版社,2001
[31] Barthelemy J P,Cohen G,Lobstein A .(Translated by Catherine Fritsch-Mignotte and Maurice Mignotte) .Algorithmic Complexity and Communication Problems .London: University College London(UCL)Press,1996
[32] Pieprzyk J,Sadeghiyan B .Design of Hashing Algorithms . Berlin:Springer-Verlag,1993
[33] Mollin R A .An Introduction to Cryptography .Boca Raton: Chapman&Hall/ CRC Press, 2001
[34] Yashchenko V V .Cryptography: An Introduction .Translated to English from Russian and Published by AMS, 2002
[35] Chaum D ., Van Heyst E .Group Signature . Advances in Cryptology-Eurocrypt'91, 1991,LNCS 547:257～265
[36] Kim S J,Park S J,Won D H .Convertible Group Signature .Advances in Cryptology-Asiacrypt'96,1996,LNCS 1163:311～321
[37] Lim C H,Lee P J .Remark on Convertible Group Signatures of Asiacrypt'96 .Electronics Letter .1997,33(5):383～384
[38] Wang C H,Hwang T,Lee N Y .Comments on Two Group Signatures .Information Processing Letter .1999,69:95～97
[39] 王新梅,马文平,武传坤 .纠错密码理论 .北京:人民邮电出版社,2001
[40] Wang X M .Digitial Signature Scheme Based on Error-correcting Codes .IEE Electronics Letters .1990,26(13):898～899

[41] 王新梅 .纠错码数字签名、加密纠错公钥体制 .电子学报 .1992,19(5):48～54
[42] 王新梅 .纠错码数字签名公钥体制 .通信学报 .1998,14(1):34～39
[43] Harn L, Wang D C .Cryptanalysis and Modification of Digital Signature Scheme Based on Error-correcting codes . IEE Electronics Letters .1992,28(2):157～159 .
[44] Alabbadi M, Wicker S B .Security of Xinmei digital signature scheme .IEE Electronics Letters 1992,28(9):890～891
[45] Alabbadi M, Wicker S B .Cryptanalysis of the Harn and Wang Modification of the Xinmei Digital Signature Scheme .IEE Electronics Letters .1992,28(18):1756～1757
[46] Johannes A B .Introduction to Cryptography . New York:Springer-Verlag,2001
[47] Gallager R G .Information Theory and Reliable Communication .New York: John . & Sons, Inc 1968
[48] 王育民,梁传甲 .信息与编码理论 .西安:西北电讯工程学院出版社,1986
[49] 常迥 .信息理论基础 .北京:清华大学出版社,1993
[50] 周炯磐 .信息理论基础 .北京:人民邮电出版社,1983
[51] Deavours C A Kahn D, Kruh L, Mellen G, Winkel Cryptology Yesterday, Today, and Tomorrow .Boston:Artech House,1987
[52] 万哲先 .代数和编码 .北京:科学出版社,1985
[53] 肖国镇,梁传甲,王育民 .伪随机序列及其应用 .北京:国防工业出版社,1985
[54] Golomb S W .Shift Register Sequences .San Francisco: Holden-Day,1967
[55] 万哲先,戴宗铎,刘木兰,冯绪宁 .非线性移位寄存器 .北京:科学出版社,1978
[56] Lidl R, Niederreiter .Introduction to Finite Fields and Their Applications .Cambridge: Cambridge University Press,1986
[57] Rueppel R .A Analysis and Design of Stream Ciphers .New York:Springer-Verlag,1986
[58] 丁存生,肖国镇 .流密码学及其应用 .北京:国防工业出版社,1994
[59] Simmons G J (Editor) . Contemporary Cryptology, The Science of Information Integrity .New York:IEEE Press,1992
[60] Shamir A .Identity-Based Cryptosystems and Signature Schemes . Advances in Cryptology-Crypto '84 .New York:Springer-Verlag,1985
[61] Feige U, Fiat A, Shamir A . Zero Knowledge Proofs of Identity .Proceedings of STOC, 1987
[62] Brickell E F, Mccurley K S . An Interactive Identification Scheme Based on Discrete Logarithms and Factoring .Advances in Cryptology-Eurocrypt 90,1991
[63] Menezes A J, Van Oorschot P C, Vanstone S A .Handbook of Applied Cryptography . Boca Raton:CRC Press,1996
[64] Hwang T Chen J L . Identity-Based Conference Key Broadcast System . IEE proc-Com-

put Digit Tech .1994, 141(1):57～60

[65] Stinson D R .An Explication of Secret Sharing Schemes .Designs, Codes and Cryptography .1992(2):357～390

[66] Stefan Katzenbeisser .Recent Advances in RSA Cryptography .Boston: Kluwer Academic Publishers, 2001

[67] Menezes .A J . Elliptic Curve Public Key Cryptosystems .Boston: Kluwer Academic Publishers, 1993

[68] Araki Miura, S .Overview of Elliptic Curve Cryptography .Lecture Notes in Computer Science .1998, 1431:29～49

[69] Nechvatal J, Barker E, Bassham L, Burr W, Dworkin M, Foti J, Roback E .Report on the Development of the Advanced Encryption Standard(AES) .available from NIST's AES homepage .URL: http:// www .nist .gov/ aes

[70] John Daemen, Vincent Rijmen .高级加密标准(AES)算法——Rijndael 的设计 .谷大武, 徐胜波译 .北京:清华大学出版社, 2003