

高·等·院·校·信·息·安·全·专·业·系·列·教·材

中国计算机学会教育专业委员会与清华大学出版社联合组织编写



名誉主编: 何德全 编委会主任: 肖国镇

Modern Cryptography

现代密码学

杨波 编著

<http://www.tup.tsinghua.edu.cn>



清华大学出版社

内容简介

本书旨在介绍现代密码学的基本原理及方法。全书共分8章。第1章引言，介绍现代密码学的基本概念，其余各章分别介绍流密码、分组密码、公钥密码、密钥分配与密钥管理、消息认证和鉴别算法、数字签名和密钥协议、网络加密与认证。

本书内容翔实，概念表述严谨，语言精练，例题丰富，切合教学之用。

本书可作为高等院校信息安全、计算机、通信工程、密码学及相关专业大学本科生和研究生的教材，也可作为通信工程师和计算机网络安全师的参考读物。

作者简介

傅涛，博士，西安电子科技大学教授，博士生导师，主持多项国家自然科学基金项目、863计划项目、国家密码发展基金项目 and 国防科技重点实验室基金科研项目，已发表论文和专著出版著作多部。

高等院校信息安全专业系列教材 近期书目

信息与系统安全概论

Introduction to Information and Systems Security

信息安全数学基础

网络安全

现代密码学

PKI原理与技术

入侵检测技术

安全扫描技术

安全扫描技术习题与实验指导

网络攻击与防御技术

网络攻击与防御技术习题与实验指导

操作系统安全

安全协议导论

信息论与编码

计算机病毒及其防治技术

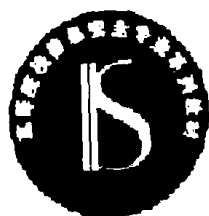
ISBN 7-302-06814-3



9 787302 068143 >

定价：24.00元

TN918.1
6



高·等·院·校·信·息·安·全·专·业·系·列·教·材

Modern Cryptography

现代密码学

杨波 编著

北方工业大学图书馆



00540916

清华大学出版社

北京

内 容 简 介

本书旨在介绍现代密码学的基本原理及方法。全书共分8章,第1章介绍现代密码学的基本概念,其余各章分别介绍流密码、分组密码、公钥密码、密钥分配与密钥管理、消息认证和杂凑算法、数字签字和密码协议、网络加密与认证。

本书内容翔实,概念表述严谨,语言精练,例题丰富,切合教学之用。

本书可作为高等院校信息安全、计算机、通信工程、密码学及相关专业大学本科生和研究生的教材,也可作为通信工程师和计算机网络工程师的参考读物。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

现代密码学/杨波编著. —北京:清华大学出版社,2003

(高等院校信息安全专业系列教材)

ISBN 7-302-06814-3

I. 现… II. 杨… 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(2003)第 050134 号

出 版 者: 清华大学出版社

<http://www.tup.tsinghua.edu.cn>

社 总 机: 010-62770175

地 址: 北京清华大学学研大厦

邮 编: 100084

客户服务: 010-62776969

责任编辑: 张 民

封面设计: 孟繁聪

版式设计: 刘伟森

印 刷 者: 北京四季青印刷厂

发 行 者: 新华书店总店北京发行所

开 本: 185×230 印 张: 15 字 数: 295 千字

版 次: 2003 年 8 月第 1 版 2003 年 8 月第 1 次印刷

书 号: ISBN 7-302-06814-3/TP·5065

印 数: 1~5000

定 价: 24.00 元

高等院校信息安全专业系列教材

编审委员会

名誉主编：何德全(中国工程院院士)

主 任：肖国镇

委 员：(按姓氏笔画为序)

方滨兴	冯登国	刘建亚	何大可	张玉清
杨 波	吴 刚	李建华	张焕国	陈克非
官 力	洪佩琳	胡振辽	胡铭曾	胡道元
侯整风	卿斯汉	钱德沛	曹珍富	谢冬青
焦金生	廖明宏	裴昌幸		

策划编辑：张 民

本书责任编委：肖国镇

序

在社会信息化的进程中,信息已成为社会发展的重要资源,信息安全也成为 21 世纪国际竞争的重要战场。为了保护国家的政治利益和经济利益,各国政府都非常重视信息和网络安全,信息安全已成为一个世纪性、全球性的研究课题。

我国的信息安全事业正在蓬勃发展,国家领导高度重视,各部门通力合作、统筹规划,大大加快了我国信息安全产业发展的步伐。随着信息安全产业的快速发展,社会对信息安全人才的需求在不断增加,在高等教育领域大力推进信息安全的专业化教育,将是国家在信息安全领域掌握自主权、占领先机的重要举措。

目前,许多大学和科研院所已创办了信息安全专业或是开设了相关课程。很高兴中国计算机学会教育专业委员会和清华大学出版社在近期联合组织了一系列信息安全专业的研讨活动。他们以严谨负责的态度,认真组织全国各高校和科研院所的专家、学者,共同研讨信息安全专业的教育方法和课程体系,并在进行大量前瞻性研究工作的基础上,启动了“高等院校信息安全专业系列教材”的编写工作。这套教材将是我国信息安全专业的第一套完整、权威的教材,相信可以对全国的高等院校信息安全专业的建设起到很好的促进作用。

希望中国计算机学会教育专业委员会和清华大学出版社能够将这个研究课题一直做下去,也希望这套教材能够取得成功并不断完善,以促进各高等院校培养出更多、更好的信息安全专门人才,为我国的信息安全事业做出更大的贡献。

何德全

中国工程院院士
高等院校信息安全专业系列教材编审委员会名誉主编
2003 年 7 月于北京

出版说明

21 世纪是信息时代,信息已成为社会发展的重要战略资源,社会的信息化已成为当今世界发展的潮流和核心,而信息安全在信息社会中将扮演极为重要的角色,它直接关系到国家安全、企业经营和人们的日常生活。随着信息安全产业的快速发展,国家对信息安全人才的需求量不断增加,但目前信息安全人才极度匮乏,远远不能满足金融、商业、公安、军事和政府等部门的需求。要解决供需矛盾,必须加快信息安全人才的培养,以满足社会的需求。为此,教育部继 2001 年批准在武汉大学开设信息安全本科专业之后,又批准了多所高等院校设立信息安全本科专业,而且许多高校和科研院所已设立了信息安全方向的具有硕士和博士学位授予权的学科点。

信息安全是计算机、通信工程、数学等领域的交叉学科,对于这一新兴学科的培养模式和课程设置,各高校普遍缺乏经验,因此中国计算机学会教育专业委员会和清华大学出版社联合主办了“信息安全专业教育教学研讨会”等一系列研讨活动,并成立了“高等院校信息安全专业系列教材”编审委员会,由我国信息安全领域著名专家何德全院士担任名誉主编,著名学者肖国镇教授担任编委会主任,共同指导“高等院校信息安全专业系列教材”的编写工作。编委会本着研究先行的指导原则,认真研讨国内外高等院校信息安全专业的教学体系和课程设置,进行了大量前瞻性的研究工作,而且这种研究工作将随着我国信息安全专业的发展不断深入。经过编委会全体委员及相关专家的推荐和审定,确定了编写教材的作者,这些作者绝大多数都是既在本专业领域有深厚的学术造诣,又在教学第一线有丰富的教学经验的学者、专家。

本系列教材是我国第一套专门针对信息安全专业的教材,其特点是:

- ① 体系完整,结构合理,内容先进。
- ② 适应面广,能够满足信息安全、计算机、通信工程等相关专业对信息安全领域课程的教材要求。
- ③ 立体配套,除主教材外,还配有多媒体电子教案、习题与实验指导等。
- ④ 版本更新及时,紧跟科学技术的新发展。

为了保证出版质量,我们坚持宁缺毋滥的原则,成熟一本,出版一本,并保持不断更新,力求将我国信息安全领域教育、科研的最新成果和成熟经验反映到教材中来。在全力做好本版教材,满足学生用书的基础上,还经由专家的推荐和审定,遴选了一批国外信息安全领域优秀的教材加入到本系列教材中,以进一步满足大家对外版书的需求。热切期望广大教师和科研工作者加入我们的队伍,同时也欢迎广大读者提出宝贵意见,以便我们对本系列教材的组织、编写与出版工作不断改进,为我国信息安全专业的教材建设与人才培养做出更大的贡献。

我们的 E-mail 地址是: zhangm@tup.tsinghua.edu.cn; 联系人: 张民。

中国计算机学会教育专业委员会

清华大学出版社

2003 年 7 月

前言

信息在社会中的地位和作用越来越重要,已成为社会发展的重要战略资源,信息技术改变着人们的生活和工作方式,信息产业已成为新的经济增长点,社会的信息化已成为当今世界发展的潮流和核心。与此同时信息的安全问题也已成为世人关注的社会问题。人们对信息安全的认识随着网络的发展经历了一个由简单到复杂的过程。

20 世纪 70 年代,主机时代的信息安全是面向单机的,由于早期的用户主要是军方,信息安全的主要内容是信息的保密性。

20 世纪 80 年代,微机和局域网的兴起带来了信息在微机间的传输和用户间的共享问题,丰富了信息安全的内涵,使人们认识到数据完整性、可用性的重要性。安全服务、安全机制等基本框架,成为信息安全的重要内容。

20 世纪 90 年代,因特网爆炸性的发展把人类带进了一个新的生存空间。因特网具有高度分布、边界模糊、层次欠清、动态演化,而用户又在其中扮演主角的特点,如何处理好这一复杂而又巨大的系统的安全,成为信息安全的主要问题。由于因特网的全球性、开放性、无缝连通性、共享性和动态性发展,使得任何人都可以自由地接入,其中有善者,也有恶者。恶者会采用各种攻击手段进行破坏活动。信息安全面临的攻击可能会来自独立的犯罪者、有组织的犯罪集团和国家情报机构。

信息安全可分为系统安全(包括操作系统的安全、数据库系统的安全等)、数据安全(包括数据的安全存储、安全传输)和内容安全(包括病毒的防护、不良内容的过滤等)3 个层次,是一个综合、交叉的学科领域,要利用数学、电子、信息、通信、计算机等诸多学科的长期知识积累和最新发展成果。信息安全研究的内容很多,它涉及安全体系结构、安全协议、密码理论、信息分析、安全监控、应急处理等,其中密码技术是保障数据安全的关键技术。

密码技术中的加密方法包括单钥密码体制(又称为对称密码体制)和公钥密码体制,而单钥密码体制又包括流密码和分组密码。本书在第 1 章介绍了现代密码学的基本概念后,第 2、3、4 章分别介绍流密码、分组密码、公钥密

码。不管哪种密码体制都需要用到密钥,因此密钥分配与密钥管理也是密码技术的重要内容,这部分内容在第5章介绍。信息的安全性除要考虑保密性外,还需考虑信息的真实性、完整性、顺序性、时间性和不可否认性,本书以两章的篇幅(第6章消息认证和杂凑算法、第7章数字签字和认证协议)介绍这部分内容。最后一章(第8章网络加密与认证)介绍了加密技术和认证技术在网络中的具体应用。

本书的特点一是内容新颖、深入、全面,涵盖了现代密码学的最新成果;二是在内容的取舍、安排等方面充分体现了教材的特点,便于教师在教学过程中实施。

本书在编写过程中参考了国内外的有关著作和文献,特别是 Stallings、王育民、卢开澄、朱文余等教授的著作。

最后要特别感谢西安电子科技大学通信工程学院的肖国镇教授,作为本书的责任编委,肖教授认真审阅了全书并提出了许多宝贵的指导意见。清华大学出版社也为本书的出版做了大量的工作,作者在此对他们表示衷心的感谢。

作 者

2003年7月

目 录

第 1 章 引言	1
1.1 信息安全面临的威胁	1
1.1.1 安全威胁	1
1.1.2 入侵者和病毒	3
1.1.3 安全业务	3
1.2 信息安全的模型	4
1.3 密码学基本概念	6
1.3.1 保密通信系统	6
1.3.2 密码体制分类	8
1.3.3 密码攻击概述	8
第 2 章 流密码	10
2.1 流密码的基本概念	10
2.1.1 同步流密码	10
2.1.2 有限状态自动机	11
2.1.3 密钥流产生器	13
2.2 线性反馈移位寄存器	14
2.3 线性移位寄存器的一元多项式表示	16
2.4 m 序列的伪随机性	19
2.5 m 序列密码的破译	22
2.6 非线性序列	25
2.6.1 Geffe 序列生成器	25
2.6.2 J-K 触发器	26
2.6.3 Pless 生成器	27
2.6.4 钟控序列生成器	27

习题	29
第 3 章 分组密码体制	31
3.1 分组密码概述	31
3.1.1 代换	32
3.1.2 扩散和混淆	34
3.1.3 Feistel 密码结构	35
3.2 数据加密标准	38
3.2.1 DES 描述	39
3.2.2 二重 DES	44
3.2.3 两个密钥的三重 DES	46
3.2.4 三个密钥的三重 DES	46
3.3 差分密码分析与线性密码分析	47
3.3.1 差分密码分析	47
3.3.2 线性密码分析	48
3.4 分组密码的运行模式	49
3.4.1 电码本(ECB)模式	49
3.4.2 密码分组链接(CBC)模式	50
3.4.3 密码反馈(CFB)模式	52
3.4.4 输出反馈(OFB)模式	53
3.5 IDEA	54
3.5.1 设计原理	54
3.5.2 加密过程	56
3.6 AES 算法——Rijndael	60
3.6.1 Rijndael 的数学基础和设计思想	61
3.6.2 算法说明	64
习题	73
第 4 章 公钥密码	75
4.1 数论简介	75
4.1.1 素数和互素数	75
4.1.2 模运算	76
4.1.3 费尔玛定理和欧拉定理	78

4.1.4	素性检验	80
4.1.5	欧几里得算法	81
4.1.6	中国剩余定理	83
4.1.7	离散对数	85
4.1.8	平方剩余	87
4.2	公钥密码体制的基本概念	91
4.2.1	公钥密码体制的原理	92
4.2.2	公钥密码算法应满足的要求	94
4.2.3	对公钥密码体制的攻击	95
4.3	RSA 算法	95
4.3.1	算法描述	95
4.3.2	RSA 算法中的计算问题	96
4.3.3	RSA 的安全性	98
4.3.4	对 RSA 的攻击	100
4.4	背包密码体制	100
4.5	Rabin 密码体制	103
4.6	椭圆曲线密码体制	105
4.6.1	椭圆曲线	105
4.6.2	有限域上的椭圆曲线	106
4.6.3	椭圆曲线上的密码	108
	习题	110
第 5 章	密钥分配与密钥管理	112
5.1	单钥加密体制的密钥分配	112
5.1.1	密钥分配的基本方法	112
5.1.2	一个实例	113
5.1.3	密钥的分层控制	114
5.1.4	会话密钥的有效期	114
5.1.5	无中心的密钥控制	114
5.1.6	密钥的控制使用	115
5.2	公钥加密体制的密钥管理	117
5.2.1	公钥的分配	117
5.2.2	用公钥加密分配单钥密码体制的密钥	120

5.2.3	Diffie-Hellman 密钥交换	122
5.3	密钥托管	122
5.3.1	美国托管加密标准简介	123
5.3.2	密钥托管密码体制的组成成分	127
5.4	随机数的产生	128
5.4.1	随机数的使用	128
5.4.2	随机数源	129
5.4.3	伪随机数产生器	130
5.4.4	基于密码算法的随机数产生器	131
5.4.5	BBS 产生器	133
5.5	秘密分割	134
5.5.1	秘密分割门限方案	134
5.5.2	Shamir 门限方案	135
5.5.3	Asmuth-Bloom 门限方案	137
	习题	138
第 6 章	消息认证和杂凑算法	139
6.1	消息认证码	139
6.1.1	消息认证码的定义及使用方式	139
6.1.2	产生 MAC 的函数应满足的要求	140
6.1.3	数据认证算法	142
6.2	杂凑函数	143
6.2.1	杂凑函数的定义及使用方式	143
6.2.2	杂凑函数应满足的条件	145
6.2.3	生日攻击	145
6.2.4	迭代型杂凑函数的一般结构	147
6.3	MD5 杂凑算法	148
6.3.1	算法描述	148
6.3.2	MD5 的压缩函数	151
6.3.3	MD5 的安全性	153
6.4	安全杂凑算法	154
6.4.1	算法描述	154
6.4.2	SHA 的压缩函数	156

6.4.3	SHA 与 MD5 的比较	157
6.5	HMAC 算法	158
6.5.1	HMAC 的设计目标	158
6.5.2	算法描述	159
6.5.3	HMAC 的安全性	161
	习题	161

第 7 章 数字签字和密码协议

7.1	数字签字的基本概念	163
7.1.1	数字签字应满足的要求	163
7.1.2	数字签字的产生方式	164
7.1.3	数字签字的执行方式	166
7.2	数字签字标准	168
7.2.1	DSS 的基本方式	168
7.2.2	数字签字算法 DSA	169
7.3	其他签字方案	170
7.3.1	基于离散对数问题的数字签字体制	170
7.3.2	基于大数分解问题的数字签字体制	174
7.4	认证协议	176
7.4.1	相互认证	176
7.4.2	单向认证	180
7.5	身份证明技术	182
7.5.1	交互证明系统	182
7.5.2	简化的 Fiat-Shamir 身份识别方案	182
7.5.3	零知识证明	184
7.5.4	Fiat-Shamir 身份识别方案	185
7.6	其他密码协议	186
7.6.1	智力扑克	186
7.6.2	掷硬币协议	187
7.6.3	不经意传输	189
	习题	192

第 8 章 网络加密与认证	193
8.1 网络通信加密	193
8.1.1 开放系统互连和 TCP/IP 分层模型	193
8.1.2 网络加密方式	195
8.2 Kerberos 认证系统	198
8.2.1 Kerberos V4	198
8.2.2 Kerberos 区域与多区域的 Kerberos	201
8.3 X. 509 认证业务	203
8.3.1 证书	203
8.3.2 认证过程	206
8.4 PGP	207
8.4.1 运行方式	208
8.4.2 密钥和密钥环	212
8.4.3 公钥管理	217
参考文献	221

第 1 章

引 言

1.1 信息安全面临的威胁

1.1.1 安全威胁

信息在社会中的地位 and 作用越来越重要,已成为社会发展的重要战略资源,信息技术改变着人们的生活和工作方式,信息产业已成为新的经济增长点,社会的信息化已成为当今世界发展的潮流和核心。与此同时信息的安全问题也已成为世人关注的社会问题。人们对信息安全的认识随着网络的发展经历了一个由简单到复杂的过程。

20 世纪 70 年代,主机时代的信息安全是面向单机的,由于早期的用户主要是军方,信息安全的主要内容是信息的保密性。

20 世纪 80 年代,微机和局域网的兴起带来了信息在微机间的传输和用户间的共享问题,丰富了信息安全的内涵,使人们认识到数据完整性、可用性的重要性。安全服务、安全机制等基本框架,成为信息安全的重要内容。

20 世纪 90 年代,因特网爆炸性的发展把人类带进了一个新的生存空间。因特网具有高度分布、边界模糊、层次欠清、动态演化,而用户又在其中扮演主角的特点,如何处理好这一复杂而又巨大的系统的安全,成为信息安全的主要问题。由于因特网的全球性、开放性、无缝连通性、共享性、动态性发展,使得任何人都可以自由地接人,其中有善者,也有恶者。恶者会采用各种攻击手段进行破坏活动。信息安全面临的攻击可能会来自独立的犯罪者、有组织的犯罪集团和国家情报机构。对信息的攻击具有以下新特点:无边界性、突发性、蔓延性和隐蔽性。因此要了解信息安全,首先应该知道信息安全面临哪些威胁。

信息安全所面临的威胁来自很多方面,并且随着时间的变化而变化。这些威胁可以宏观地分为人为威胁和自然威胁。

自然威胁可能来自于各种自然灾害、恶劣的场地环境、电磁辐射和电磁干扰、网络设

备自然老化等。这些事件有时会直接威胁信息的安全,影响信息的存储媒质。

本节主要讨论人为威胁,也就是对信息的人为攻击。这些攻击手段都是通过寻找系统的弱点,以便达到破坏、欺骗、窃取数据等目的,造成经济上和政治上不可估量的损失。人为攻击可分为被动攻击和主动攻击,如图 1 1 所示。

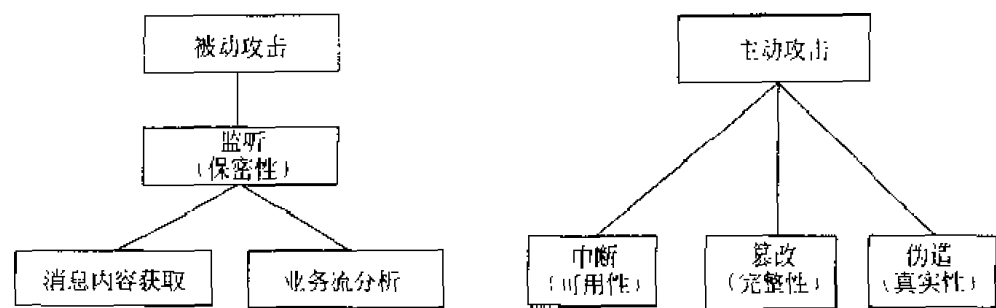


图 1-1 攻击类型分类

1. 被动攻击

被动攻击即窃听,是对系统的保密性进行攻击,如搭线窃听、对文件或程序的非法复制等,以获取他人的信息。被动攻击又分为两类:一类是获取消息的内容,很容易理解;另一类是进行业务流分析,假如通过某种手段,比如加密,使得敌手无法从截获的消息得到消息的真实内容,然而敌手却有可能获得消息的格式、确定通信双方的位置和身份以及通信的次数和消息的长度,这些信息对通信双方来说可能是敏感的,例如公司间的合作关系可能是保密的、电子函件用户可能不想让他人知道自己正在和谁通信、电子现金的支付者可能不想让别人知道自己正在消费、Web 浏览器用户也可能不愿意让别人知道自己正在浏览哪一站点。

被动攻击因不对消息做任何修改,因而是难以检测的,所以抗击这种攻击的重点在于预防而非检测。

2. 主动攻击

主动攻击包括对数据流的篡改或产生某些假的数据流。主动攻击又可分为以下 3 类:

- ① 中断 是对系统的可用性进行攻击。如破坏计算机硬件、网络或文件管理系统。
- ② 篡改 是对系统的完整性进行攻击。如修改数据文件中的数据、替换某一程序使其执行不同的功能、修改网络中传送的消息内容等。
- ③ 伪造 是对系统的真实性进行攻击。如在网络中插入伪造的消息或在文件中插入伪造的记录。

绝对防止主动攻击是十分困难的,因为需要随时随地对通信设备和通信线路进行物理保护,因此抗击主动攻击的主要途径是检测,以及对此攻击造成的破坏进行恢复。

1.1.2 入侵者和病毒

信息安全的人为威胁主要来自用户(恶意的或无恶意的)和恶意软件的非法侵入。入侵信息系统的用户也称为黑客,黑客可能是某个无恶意的人,其目的仅仅是破译和进入一个计算机系统;或者是某个心怀不满的雇员,其目的是对计算机系统实施破坏;也可能是一个犯罪分子,其目的是非法窃取系统资源(如窃取信用卡号或非法资金传送),对数据进行未授权的修改或破坏计算机系统。

恶意软件指病毒、蠕虫等恶意程序,可分为两类,如图 1-2 所示,一类需要主程序,另一类不需要。前者是某个程序中的一段,不能独立于实际的应用程序或系统程序;后者是能被操作系统调度和运行的独立程序。

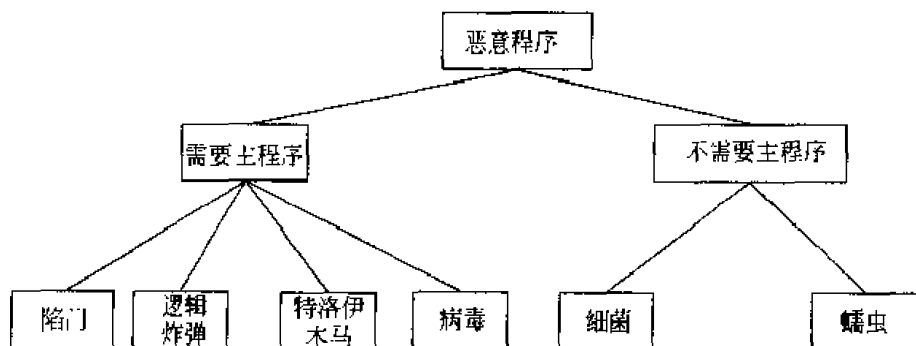


图 1-2 恶意程序分类

对恶意软件也可根据其能否自我复制来进行分类。不能自我复制的一般是程序段,这种程序段在主程序被调用执行时就可激活。能够自我复制的或者是程序段(病毒)或者是独立的程序(蠕虫、细菌等),当这种程序段或独立的程序被执行时,可能复制一个或多个自己的副本,以后这些副本可在这一系统或其他系统中被激活。以上仅是大致分类,因为逻辑炸弹或特洛伊木马可能是病毒或蠕虫的一部分。

1.1.3 安全业务

安全业务指安全防护措施,有以下 5 种。

1. 保密业务

保护数据以防被动攻击。保护方式可根据保护范围的大小分为若干级,其中最高级保护可在一定时间范围内保护两个用户之间传输的所有数据,低级保护包括对单个消息

的保护或对一个消息中某个特定域的保护。保密业务还包括对业务流实施的保密,防止敌手进行业务流分析以获得通信的信源、信宿、次数、消息长度和其他信息。

2. 认证业务

用于保证通信的真实性。在单向通信的情况下,认证业务的功能是使接收者相信消息确实是由它自己所声称的那个信源发出的。在双向通信的情况下,例如计算机终端和主机的连接,在连接开始时,认证服务则使通信双方都相信对方是真实的(即的确是它所声称的实体);其次,认证业务还保证通信双方的通信连接不能被第三方介入,以假冒其中的一方而进行非授权的传输或接收。

3. 完整性业务

和保密业务一样,完整性业务也能应用于消息流、单个消息或一个消息的某一选定域。用于消息流的完整性业务目的在于保证所接收的消息未经复制、插入、篡改、重排或重放,即保证接收的消息和所发出的消息完全一样;这种服务还能对已毁坏的数据进行恢复,所以这种业务主要是针对对消息流的篡改和业务拒绝的。应用于单个消息或一个消息某一选定域的完整性业务仅用来防止对消息的篡改。

4. 不可否认业务

用于防止通信双方中的某一方对所传输消息的否认,因此,一个消息发出后,接收者能够证明这一消息的确是由通信的另一方发出的。类似地,当一个消息被接收后,发出者能够证明这一消息的确已被通信的另一方接收了。

5. 访问控制

访问控制的目标是防止对网络资源的非授权访问,控制的实现方式是认证,即检查欲访问某一资源的用户是否具有访问权。

2 信息安全的模型

信息安全的基本模型可以用图 1-3 来表示。

通信双方欲传递某个消息,需通过以下方式建立一个逻辑上的信息通道:首先在网络中定义从发送方到接收方的一个路由,然后在该路由上共同执行通信协议。

如果需要保护所传信息以防敌手对其保密性、认证性等构成的威胁,则需要考虑通信的安全性。安全传输技术有以下两个基本成分:

- ① 消息的安全传输,包括对消息的加密和认证。加密的目的是将消息搞乱以使敌

1.2 信息安全的模型

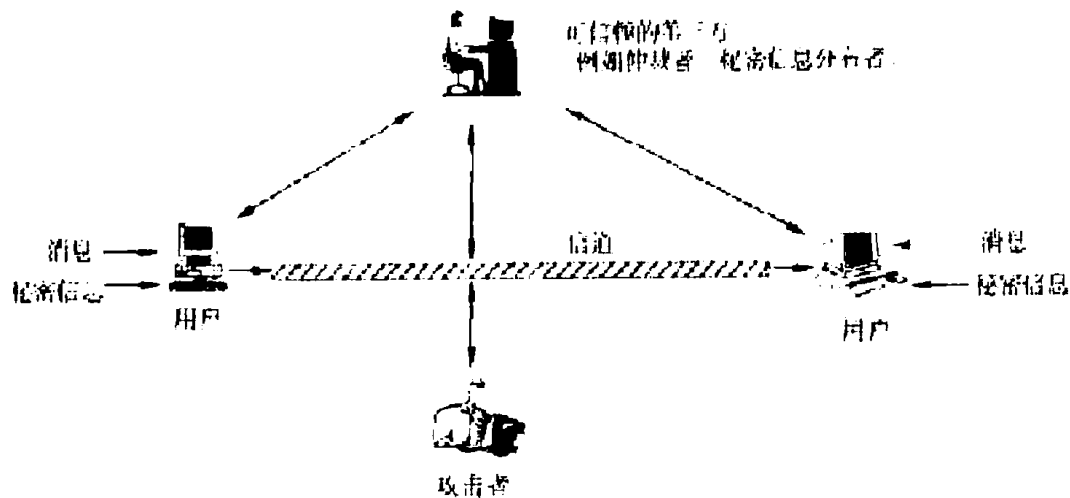


图 1-3 信息安全的基本模型

手无法读懂,认证的目的是检查发送者的身份。

② 通信双方共享的某些秘密信息,如加密密钥。

为获得消息的安全传输,可能还需要一个可信的第三方,其作用可能是负责向通信双方发布秘密信息或者在通信双方有争议时进行仲裁。

安全的网络通信必须考虑以下 4 个方面:

① 加密算法。

② 用于加密算法的秘密信息。

③ 秘密信息的分布和共享。

④ 使用加密算法和秘密信息以获得安全服务所需的协议。

以上考虑的是信息安全的一般模型,然而还有一些情况。图 1 4 表示保护信息系统以防未经授权访问的一个模型。

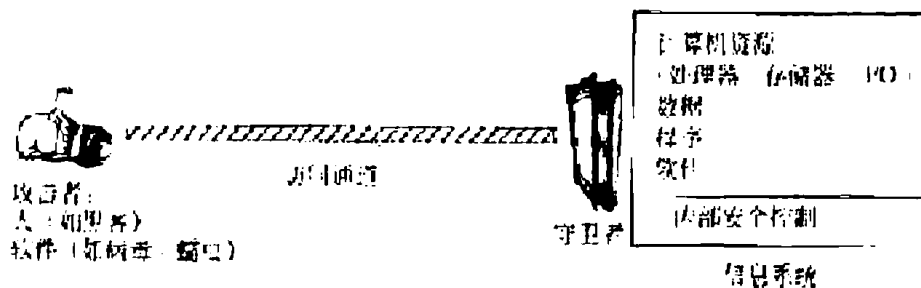


图 1 4 信息系统的保护模型

对付未经授权访问的安全机制可分为两道防线:第一道称为守卫者,它包括基于通行

字的登录程序和屏蔽逻辑程序,分别用于拒绝非授权用户的访问、检测和拒绝病毒;第二道防线由一些内部控制部件构成,用于管理系统内部的各项操作和分析所存有的信息,以检查是否有未授权的入侵者。

上面介绍了信息安全面临的威胁以及信息安全的一般模型。信息安全可分为系统安全(包括操作系统安全、数据库系统安全等)、数据安全(包括数据的安全存储、安全传输)和内容安全(包括病毒的防护、不良内容的过滤等)3个层次,是一个综合、交叉的学科领域,要利用数学、电子、信息、通信、计算机等诸多学科的长期知识积累和最新发展成果。信息安全研究的内容很多,它涉及安全体系结构、安全协议、密码理论、信息分析、安全监控、应急处理等,其中密码技术是保障数据安全的关键技术。

1.3 密码学基本概念

1.3.1 保密通信系统

通信双方采用保密通信系统可以隐蔽和保护需要发送的消息,使未授权者不能提取信息。发送方将要发送的消息称为明文,明文被变换成看似无意义的随机消息,称为密文,这种变换过程称为加密;其逆过程,即由密文恢复出原明文的过程称为解密。对明文进行加密操作的人员称为加密员或密码员。密码员对明文进行加密时所采用的一组规则称为加密算法。传送消息的预定对象称为接收者,接收者对密文进行解密时所采用的一组规则称为解密算法。加密和解密算法的操作通常都是在—组密钥控制下进行的,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,或实质上等同,即从一个易于得出另一个,称其为单钥或对称密码体制。若加密密钥和解密密钥不相同,从一个难于推出另一个,则称为双钥或非对称密码体制。密钥是密码体制安全保密的关键,它的产生和管理是密码学中的重要研究课题。

在信息传输和处理系统中,除了预定的接收者外,还有非授权者,他们通过各种办法(如搭线窃听、电磁窃听、声音窃听等)来窃取机密信息,称其为截收者。截收者虽然不知道系统所用的密钥,但通过分析可能从截获的密文推断出原来的明文或密钥,这一过程称为密码分析,从事这一工作的人称为密码分析员,研究如何从密文推演出明文、密钥或解密算法的学问称为密码分析学。对一个保密通信系统采取截获密文进行分析的这类攻击称为被动攻击。现代信息系统还可能遭受的另一类攻击是主动攻击,非法入侵者、攻击者或黑客主动向系统窜扰,采用删除、增添、重放、伪造等窜改手段向系统注入假消息,达到利己害人的目的。这是现代信息系统中更为棘手的问题。

保密通信系统可用图 1-5 表示,它由以下几部分组成:明文消息空间 M ,密文消息空间 C ,密钥空间 K_1 和 K_2 ,在单钥体制下 $K_1 = K_2 = K$,此时密钥 K 需经安全的密钥信道由发送方传给接收方;加密变换 $E_{k_1}: M \rightarrow C$,其中 $k_1 \in K_1$,由加密器完成;解密变换 $D_{k_2}: C \rightarrow M$,其中 $k_2 \in K_2$,由解密器实现。称总体 $(M, C, K_1, K_2, E_{K_1}, D_{K_2})$ 为保密通信系统。对于给定明文消息 $m \in M$,密钥 $k_1 \in K_1$,加密变换将明文 m 变换为密文 c ,即

$$c = f(m, k_1) = E_{k_1}(m) \quad m \in M, k_1 \in K_1$$

接收方利用通过安全信道送来的密钥 k ($k \in K$,单钥体制下)或用本地密钥发生器产生的解密密钥 k_2 ($k_2 \in K_2$,双钥体制下)控制解密操作 D ,对收到的密文进行变换得到恢复的明文消息,即:

$$m = D_{k_2}(c) \quad m \in M, k_2 \in K_2$$

而密码分析者,则用其选定的变换函数 h ,对截获的密文 c 进行变换,得到的明文是明文空间中的某个元素,即

$$m' = h(c)$$

一般 $m' \neq m$ 。如果 $m' = m$,则分析成功。

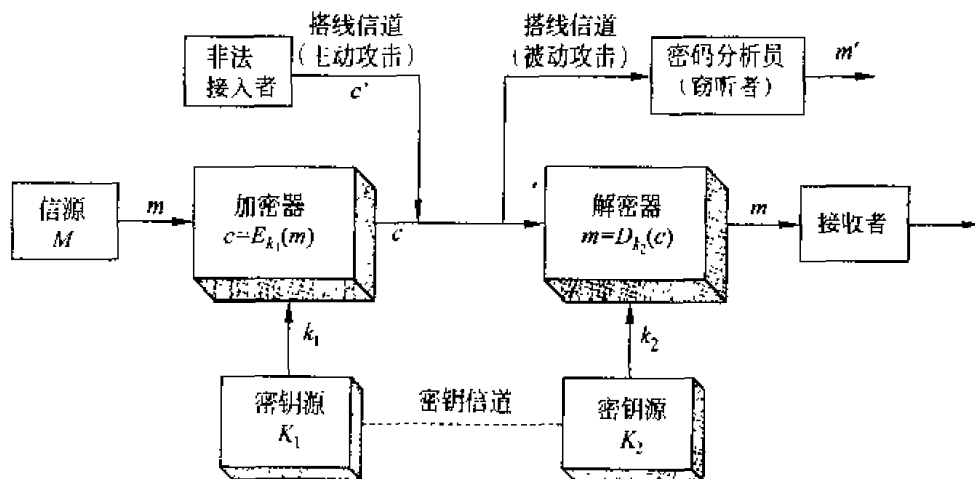


图 1-5 保密通信系统模型

为了保护信息的保密性,抗击密码分析,保密系统应当满足下述要求:

① 系统即使达不到理论上是不可破的,即 $p_r\{m' = m\} = 0$,也应当为实际上不可破的。就是说,从截获的密文或某些已知的明文密文对,要决定密钥或任意明文在计算上是不可行的。

② 系统的保密性不依赖于对加密体制或算法的保密,而依赖于密钥。这是著名的 Kerckhoff 原则。

③ 加密和解密算法适用于所有密钥空间中的元素。

④ 系统便于实现和使用。

1.3.2 密码体制分类

密码体制从原理上可分为两大类,即单钥体制和双钥体制。

单钥体制的加密密钥和解密密钥相同。采用单钥体制的系统的保密性主要取决于密钥的保密性,与算法的保密性无关,即由密文和加解密算法不可能得到明文。换句话说,算法无需保密,需保密的仅是密钥。根据单钥密码体制的这种特性,单钥加解密算法可通过低费用的芯片来实现。密钥可由发送方产生然后再经一个安全可靠的途径(如信使递送)送至接收方,或由第三方产生后安全可靠地分配给通信双方。如何产生满足保密要求的密钥以及如何将密钥安全可靠地分配给通信双方是这类体制设计和实现的主要课题。密钥产生、分配、存储、销毁等问题,统称为密钥管理。这是影响系统安全的关键因素,即使密码算法再好,若密钥管理问题处理不好,就很难保证系统的安全保密。

单钥体制对明文消息的加密有两种方式:一是明文消息按字符(如二元数字)逐位地加密,称之为流密码;另一种是将明文消息分组(含有多个字符),逐组地进行加密,称之为分组密码。单钥体制不仅可用于数据加密,也可用于消息的认证。

双钥体制是由 Diffie 和 Hellman 于 1976 年首先引入的。采用双钥体制的每个用户都有一对选定的密钥:一个是可以公开的,可以像电话号码一样进行注册公布;另一个则是秘密的。因此双钥体制又称为公钥体制。

双钥密码体制的主要特点是将加密和解密能力分开,因而可以实现多个用户加密的消息只能由一个用户解读,或由一个用户加密的消息而使多个用户可以解读。前者可用于公共网络中实现保密通信,而后者可用于实现对用户的认证。详细介绍在第 3 章。

1.3.3 密码攻击概述

表 1-1 是攻击者对密码系统的 4 种攻击类型,类型的划分由攻击者可获取的信息量决定。其中,最困难的攻击类型是惟密文攻击,这种攻击的手段一般是穷搜索法,即对截获的密文依次用所有可能的密钥试译,直到得到有意义的明文。只要有足够多的计算时间和存储容量,原则上穷搜索法总是可以成功的。但实际中,任何一种能保障安全要求的实用密码都会设计得使这一方法在实际上是不可行的。敌手因此还需对密文进行统计测试分析,为此需要知道被加密的明文的类型,比如英文文本、法文文本、MD-DOS 执行文件、Java 源列表等。

表 1-1 对密码系统的攻击类型

攻击类型	攻击者掌握的内容
惟密文攻击	<ul style="list-style-type: none"> • 加密算法 • 截获的部分密文
已知明文攻击	<ul style="list-style-type: none"> • 加密算法 • 截获的部分密文 • 一个或多个明文密文对
选择明文攻击	<ul style="list-style-type: none"> • 加密算法 • 截获的部分密文 • 自己选择的明文消息及由密钥产生的相应密文
选择密文攻击	<ul style="list-style-type: none"> • 加密算法 • 截获的部分密文 • 自己选择的密文消息及相应的被解密的明文

惟密文攻击时,敌手知道的信息量最少,因此最易抵抗。然而,很多情况下,敌手可能有更多的信息,也许能截获一个或多个明文及其对应的密文,也许知道消息中将出现的某种明文格式。例如 ps 格式文件开始位置的格式总是相同的,电子资金传送消息总有一个标准的报头或标题。这时的攻击称为已知明文攻击,敌手也许能够从已知的明文被转换成密文的方式得到密钥。

与已知明文攻击密切相关的一种攻击法称为可能字攻击。例如对一篇散文加密,敌手可能对消息含义知之甚少。然而,如果对非常特别的信息加密,敌手也许能知道消息中的某一部分。例如,发送一个加密的账目文件,敌手可能知道某些关键字在文件报头的位置。又如,一个公司开发的程序的源代码中,可能在某个标准位置上有该公司的版权声明。

如果攻击者能在加密系统中插入自己选择的明文消息,则通过该明文消息对应的密文,有可能确定出密钥的结构,这种攻击称为选择明文攻击。

选择密文攻击是指攻击者利用解密算法,对自己所选的密文解密出相应的明文。

还有两个概念值得注意。第一,一个加密算法是无条件安全的,如果算法产生的密文不能给出惟一决定相应明文的足够信息。此时无论敌手截获多少密文、花费多少时间,都不能解密密文。第二,Shannon 指出,仅当密钥至少和明文一样长时,才能达到无条件安全。也就是说除了一次一密方案外,再无其他加密方案是无条件安全的。因此,加密算法只要满足以下两条准则之一就行:

① 破译密文的代价超过被加密信息的价值。

② 破译密文所花的时间超过信息的有用期。

满足以上两个准则的加密算法称为计算上安全的。

第 2 章

流 密 码

2.1 流密码的基本概念

流密码的基本思想是利用密钥 k 产生一个密钥流 $z = z_0 z_1 \cdots$, 并使用如下规则对明文串 $x = x_0 x_1 x_2 \cdots$ 加密: $y = y_0 y_1 y_2 \cdots = E_{z_0}(x_0) E_{z_1}(x_1) E_{z_2}(x_2) \cdots$ 。密钥流由密钥流发生器 f 产生: $z_i = f(k, \sigma_i)$, 这里 σ_i 是加密器中的记忆元件(存储器)在时刻 i 的状态, f 是由密钥 k 和 σ_i 产生的函数。

分组密码与流密码的区别就在于有无记忆性(如图 2-1)。流密码的滚动密钥 $z_0 = f(k, \sigma_0)$ 由函数 f 、密钥 k 和指定的初态 σ_0 完全确定。此后, 由于输入加密器的明文可能影响加密器中内部记忆元件的存储状态, 因而 $\sigma_i (i > 0)$ 可能依赖于 $k, \sigma_0, x_0, x_1, \cdots, x_{i-1}$ 等参数。

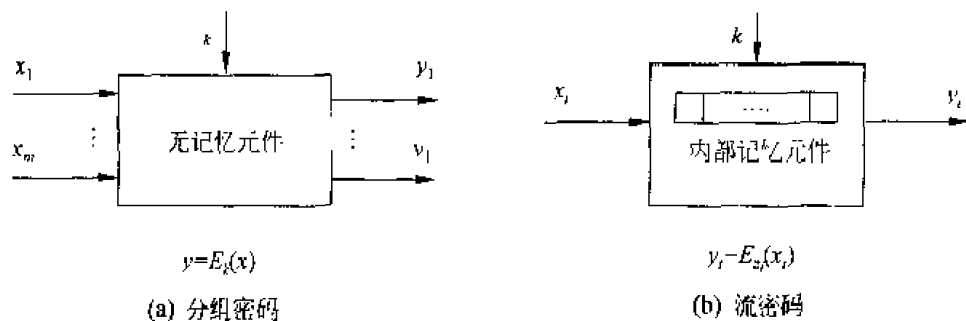


图 2-1 分组密码和流密码的比较

2.1.1 同步流密码

根据加密器中记忆元件的存储状态 σ_i 是否依赖于输入的明文字符, 流密码可进一步分成同步和自同步两种。 σ_i 独立于明文字符的叫做同步流密码, 否则叫做自同步流密码。由于自同步流密码的密钥流的产生与明文有关, 因而较难从理论上进行分析。目前大多数研究成果都是关于同步流密码的。在同步流密码中, 由于 $z_i = f(k, \sigma_i)$ 与明文字符无

关,因而此时密文字符 $y_i = E_{z_i}(x_i)$ 也不依赖于此前的明文字符。因此,可将同步流密码的加密器分成密钥流产生器和加密变换器两个部分。如果与上述加密变换对应的解密变换为 $x_i = D_{z_i}(y_i)$,则可给出同步流密码体制的模型如图 2-2 所示。

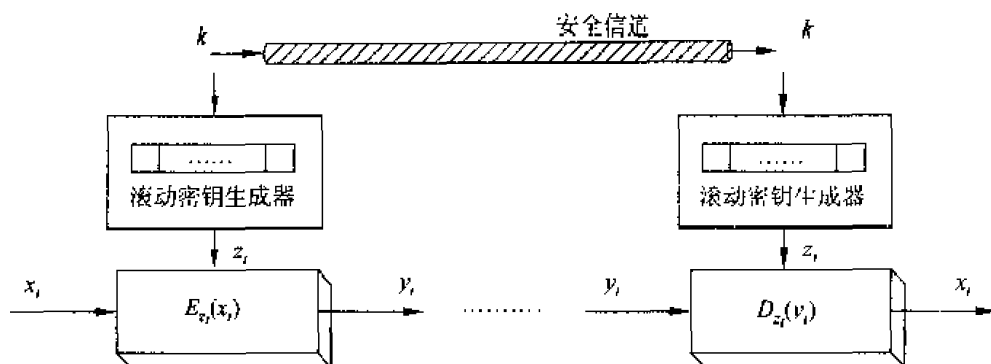


图 2-2 同步流密码体制模型

同步流密码的加密变换 E_{z_i} 可以有多种选择,只要保证变换是可逆的即可。实际使用的数字保密通信系统一般都是二元系统,因而在有限域 $GF(2)$ 上讨论的二元加法流密码(如图 2-3)是目前最为常用的流密码体制,其加密变换可表示为 $y_i = z_i \oplus x_i$ 。

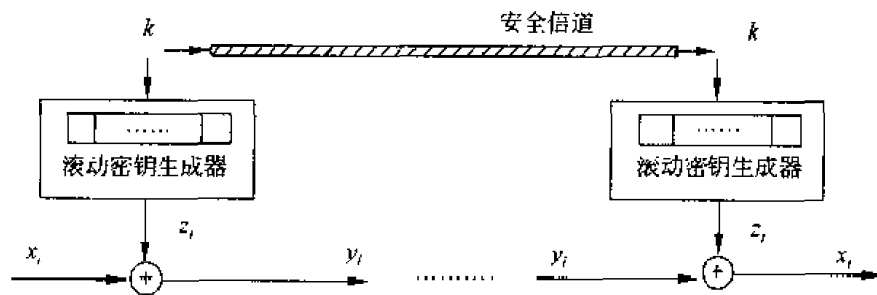


图 2-3 加法流密码体制模型

一次一密密码是加法流密码的原型。事实上,如果 $z_i = k_i$ (即密钥用作滚动密钥流),则加法流密码就退化成一次一密密码。实际工作中,密码设计者的最大愿望是设计出一个滚动密钥生成器,使得密钥 k 经其扩展成的密钥流序列 z 具有如下性质:极大的周期、良好的统计特性、抗线性分析、抗统计分析。

2.1.2 有限状态自动机

有限状态自动机是具有离散输入和输出(输入集和输出集均有限)的一种数学模型,由以下 3 部分组成:

① 有限状态集 $S = \{s_i | i = 1, 2, \dots, l\}$ 。

② 有限输入字符集 $A_1 = \{A_j^{(1)} | j = 1, 2, \dots, m\}$ 和有限输出字符集 $A_2 = \{A_k^{(2)} | k = 1, 2, \dots, n\}$ 。

③ 转移函数

$$A_k^{(2)} = f_1(s_i, A_j^{(1)}), \quad s_h = f_2(s_i, A_j^{(1)})$$

即在状态为 s_i 输入为 $A_j^{(1)}$ 时, 输出为 $A_k^{(2)}$, 而状态转移为 s_h 。

【例 2-1】 $S = \{s_1, s_2, s_3\}$, $A_1 = \{A_1^{(1)}, A_2^{(1)}, A_3^{(1)}\}$, $A_2 = \{A_1^{(2)}, A_2^{(2)}, A_3^{(2)}\}$, 转移函数由表 2-1 给出。

表 2-1 转移函数 f_1 和 f_2

f_1	$A_1^{(1)}$	$A_2^{(1)}$	$A_3^{(1)}$
s_1	$A_1^{(2)}$	$A_3^{(2)}$	$A_2^{(2)}$
s_2	$A_3^{(2)}$	$A_1^{(2)}$	$A_3^{(2)}$
s_3	$A_3^{(2)}$	$A_2^{(2)}$	$A_1^{(2)}$
f_2	$A_1^{(1)}$	$A_2^{(1)}$	$A_3^{(1)}$
s_1	s_2	s_1	s_3
s_2	s_1	s_2	s_1
s_3	s_1	s_3	s_2

有限状态自动机可用有向图表示, 称为转移图。转移图的顶点对应于自动机的状态, 若状态 s_i 在输入 $A_j^{(1)}$ 时转为状态 s_j , 且输出一字符 $A_k^{(2)}$, 则在转移图中, 从状态 s_i 到状态 s_j 有一条标有 $(A_j^{(1)}, A_k^{(2)})$ 的弧线, 见图 2-4。

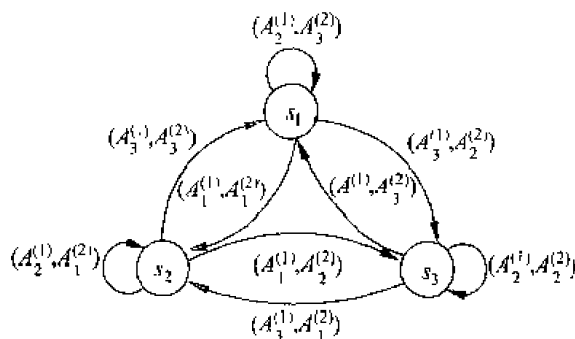


图 2-4 有限状态自动机的转移图

例 2-1 中, 若输入序列为 $A_1^{(1)} A_2^{(1)} A_1^{(1)} A_3^{(1)} A_3^{(1)} A_1^{(1)}$, 初始状态为 s_1 , 则得到状态序列

$$s_1 s_2 s_3 s_2 s_1 s_2$$

输出字符序列

$$A_1^{(2)} A_1^{(2)} A_2^{(2)} A_1^{(2)} A_3^{(2)} A_1^{(2)}$$

2.1.3 密钥流产生器

同步流密码的关键是密钥流产生器。一般可将其看成一个参数为 k 的有限状态自动机, 由一个输出符号集 Z 、一个状态集 Σ 、两个函数 φ 和 ψ 以及一个初始状态 σ_0 组成(如图 2-5)。状态转移函数 $\varphi: \sigma_i \rightarrow \sigma_{i+1}$, 将当前状态 σ_i 变为一个新状态 σ_{i+1} , 输出函数 $\psi: \sigma_i \rightarrow z_i$, 当前状态 σ_i 变为输出符号集中的一个元素 z_i 。这种密钥流生成器设计的关键在于找出适当的状态转移函数 φ 和输出函数 ψ , 使得输出序列 z 满足密钥流序列 z 应满足的几个条件, 并且要求在设备上节省的和容易实现的。为了实现这一目标, 必须采用非线性函数。

由于具有非线性的 φ 的有限状态自动机理论很不完善, 相应的密钥流产生器的分析工作受到极大的限制。相反地, 当采用线性的 φ 和非线性的 ψ 时, 将能够进行深入的分析并可以得到好的生成器。为方便讨论, 可将这类生成器分成驱动部分和非线性组合部分(如图 2-6)。驱动部分控制生成器的状态转移, 并为非线性组合部分提供统计性能好的序列; 而非线性组合部分要利用这些序列组合出满足要求的密钥流序列。

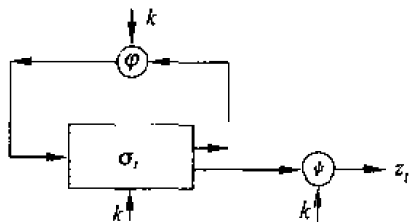


图 2-5 作为有限状态自动机的密钥流生成器

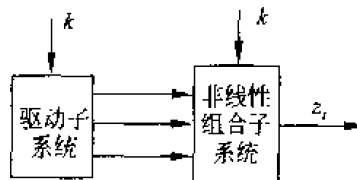


图 2-6 密钥流生成器的分解

目前最为流行和实用的密钥流产生器如图 2-7 所示, 其驱动部分是一个或多个线性反馈移位寄存器。

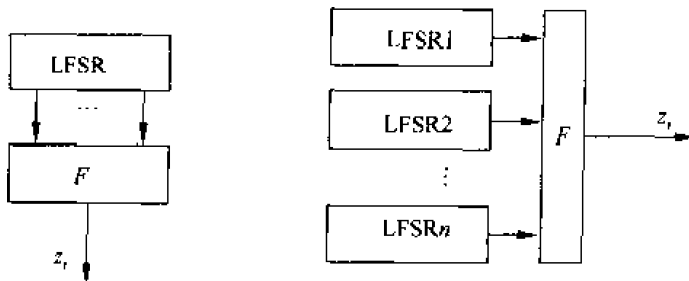


图 2-7 常见的两种密钥流产生器

2.2 线性反馈移位寄存器

移位寄存器是流密码产生密钥流的一个主要组成部分。 $GF(2)$ 上一个 n 级反馈移位寄存器由 n 个二元存储器与一个反馈函数 $f(a_1, a_2, \dots, a_n)$ 组成,如图 2-8 所示。

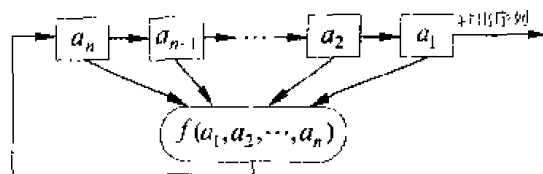


图 2-8 $GF(2)$ 上的 n 级反馈移位寄存器

每一存储器称为移位寄存器的一级,在任一时刻,这些级的内容构成该反馈移位寄存器的状态,每一状态对应于 $GF(2)$ 上的一个 n 维向量,共有 2^n 种可能的状态。每一时刻的状态可用 n 长序列

$$a_1, a_2, \dots, a_n$$

或 n 维向量

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

表示,其中 a_i 是第 i 级存储器的内容。初始状态由用户确定,当第 i 个移位时钟脉冲到来时,每一级存储器 a_i 都将其内容向下一级 a_{i-1} 传递,并根据寄存器此时的状态 a_1, a_2, \dots, a_n 计算 $f(a_1, a_2, \dots, a_n)$,作为下一时刻的 a_n 。反馈函数 $f(a_1, a_2, \dots, a_n)$ 是 n 元布尔函数,即 n 个变元 a_1, a_2, \dots, a_n 可以独立地取 0 和 1 这两个可能的值,函数中的运算有逻辑与、逻辑或、逻辑补等运算,最后的函数值也为 0 或 1。

【例 2-2】 图 2-9 是一个 3 级反馈移位寄存器,其初始状态为 $(a_1, a_2, a_3) = (1, 0, 1)$,输出可由表 2-2 求出。

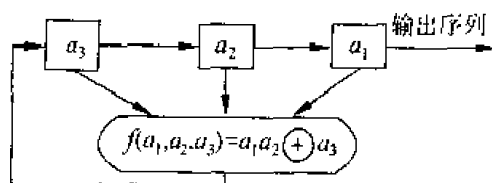


图 2-9 一个 3 级反馈移位寄存器

表 2-2 一个 3 级反馈移位寄存器的状态和输出

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
\vdots	\vdots	\vdots	\vdots

即输出序列为 101110111011..., 周期为 4。

如果移位寄存器的反馈函数 $f(a_1, a_2, \dots, a_n)$ 是 a_1, a_2, \dots, a_n 的线性函数, 则称之为线性反馈移位寄存器 LFSR (linear feedback shift register)。此时 f 可写为

$$f(a_1, a_2, \dots, a_n) = c_n a_1 \oplus c_{n-1} a_2 \oplus \dots \oplus c_1 a_n$$

其中常数 $c_i = 0$ 或 1, \oplus 是模 2 加法。 $c_i = 0$ 或 1 可用开关的断开和闭合来实现, 如图 2-10 所示。

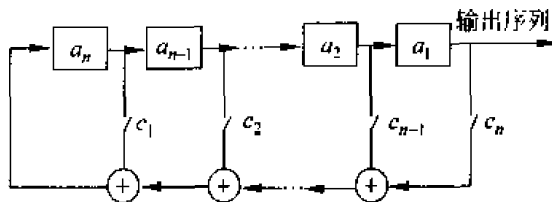


图 2-10 $GF(2)$ 上的 n 级线性反馈移位寄存器

输出序列 $\{a_i\}$ 满足

$$a_{n+t} = c_n a_t \oplus c_{n-1} a_{t+1} \oplus \dots \oplus c_1 a_{n+t-1}$$

其中 t 为非负正整数。

线性反馈移位寄存器因其实现简单、速度快、有较为成熟的理论等优点而成为构造密钥流生成器的最重要的部件之一。

【例 2-3】 图 2-11 是一个 5 级线性反馈移位寄存器, 其初始状态为 $(a_1, a_2, a_3, a_4, a_5) = (1, 0, 0, 1, 1)$, 可求出输出序列为

1001101001000010101110110001111100110...

周期为 31。

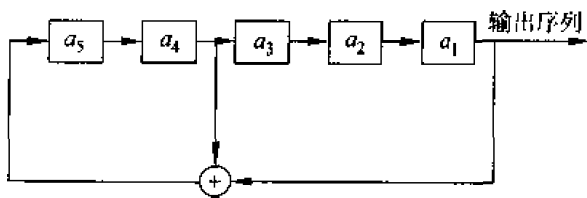


图 2-11 一个 5 级线性反馈移位寄存器

在线性反馈移位寄存器中总是假定 c_1, c_2, \dots, c_n 中至少有一个不为 0, 否则 $f(a_1, a_2, \dots, a_n) = 0$, 这样的话, 在 n 个脉冲后状态必然是 $00\dots 0$, 且这个状态必将一直持续下去。若只有一个系数不为 0, 设仅有 c_j 不为 0, 实际上是一种延迟装置。一般对于 n 级线性反馈移位寄存器, 总是假定 $c_n = 1$ 。

线性反馈移位寄存器输出序列的性质完全由其反馈函数决定。 n 级线性反馈移位寄

寄存器最多有 2^n 个不同的状态。若其初始状态为 0, 则其状态恒为 0。若其初始状态非 0, 则其后继状态不会为 0。因此 n 级线性反馈移位寄存器的状态周期小于等于 $2^n - 1$ 。其输出序列的周期与状态周期相等, 也小于等于 $2^n - 1$ 。只要选择合适的反馈函数便可使序列的周期达到最大值 $2^n - 1$, 周期达到最大值的序列称为 m 序列。

2.3 线性移位寄存器的一元多项式表示

设 n 级线性移位寄存器的输出序列 $\{a_i\}$ 满足递推关系

$$a_{n+k} = c_1 a_{n+k-1} \oplus c_2 a_{n+k-2} \oplus \cdots \oplus c_n a_k \quad (*)$$

对任何 $k \geq 1$ 成立。这种递推关系可用一个一元高次多项式

$$p(x) = 1 + c_1 x + \cdots + c_{n-1} x^{n-1} + c_n x^n$$

表示, 称这个多项式为 LFSR 的特征多项式或特征多项式。

设 n 级线性移位寄存器对应于递推关系 $(*)$, 由于 $a_i \in GF(2)$ ($i=1, 2, \dots, n$), 所以共有 2^n 组初始状态, 即有 2^n 个递推序列, 其中非恒零的有 $2^n - 1$ 个, 记 $2^n - 1$ 个非零序列的全体为 $G(p(x))$ 。

定义 2-1 给定序列 $\{a_i\}$, 幂级数

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

称为该序列的生成函数。

定理 2-1 设 $p(x) = 1 + c_1 x + \cdots + c_{n-1} x^{n-1} + c_n x^n$ 是 $GF(2)$ 上的多项式, $G(p(x))$ 中任一序列 $\{a_i\}$ 的生成函数 $A(x)$ 满足:

$$A(x) = \frac{\phi(x)}{p(x)}$$

其中

$$\phi(x) = \sum_{i=1}^n \left(c_{n-i} x^n + \sum_{j=1}^i a_j x^{j-1} \right)$$

证明: 在等式

$$\begin{aligned} a_{n+1} &= c_1 a_n \oplus c_2 a_{n-1} \oplus \cdots \oplus c_n a_1 \\ a_{n+2} &= c_1 a_{n+1} \oplus c_2 a_n \oplus \cdots \oplus c_n a_2 \\ &\vdots \end{aligned}$$

两边分别乘以 x^n, x^{n+1}, \dots , 再求和, 可得

$$A(x) = (a_1 + a_2 x + \cdots + a_n x^{n-1})$$

$$= c_1 x [A(x) - (a_1 + a_2 x + \cdots + a_{n-1} x^{n-2})] \\ + c_2 x^2 [A(x) - (a_1 + a_2 x + \cdots + a_{n-2} x^{n-3})] + \cdots + c_n x^n A(x)$$

移项整理得

$$(1 + c_1 x + \cdots + c_{n-1} x^{n-1} + c_n x^n) A(x) \\ = (a_1 + a_2 x + \cdots + a_n x^{n-1}) + c_1 x (a_1 + a_2 x + \cdots + a_{n-1} x^{n-2}) \\ + c_2 x^2 (a_1 + a_2 x + \cdots + a_{n-2} x^{n-3}) + \cdots + c_{n-1} x^{n-1} a_1$$

即

$$p(x)A(x) = \sum_{i=1}^n (c_{n-i+1} x^{n-i} \sum_{j=1}^i a_j x^{j-1}) = \phi(x) \quad (\text{证毕})$$

注意在 $GF(2)$ 上有 $a+a=0$ 。

定理 2-2 $p(x) \mid q(x)$ 的充要条件是 $G(p(x)) \subset G(q(x))$ 。

证明：若 $p(x) \mid q(x)$ ，可设 $q(x) = p(x)r(x)$ ，因此

$$A(x) = \frac{\phi(x)}{p(x)} = \frac{\phi(x)r(x)}{p(x)r(x)} = \frac{\phi(x)r(x)}{q(x)}$$

所以若 $\{a_i\} \in G(p(x))$ ，则 $\{a_i\} \in G(q(x))$ ，即 $G(p(x)) \subset G(q(x))$ 。

反之，若 $G(p(x)) \subset G(q(x))$ ，则对于多项式 $\phi(x)$ ，存在序列 $\{a_i\} \in G(p(x))$ 以 $A(x) = \frac{\phi(x)}{p(x)}$ 为生成函数。特别地，对于多项式 $\phi(x) = 1$ ，存在序列 $\{a_i\} \in G(p(x))$ 以 $\frac{1}{p(x)}$ 为生成函数。由于 $G(p(x)) \subset G(q(x))$ ，序列 $\{a_i\} \in G(q(x))$ ，所以存在函数 $r(x)$ ，

使得 $\{a_i\}$ 的生成函数也等于 $\frac{r(x)}{q(x)}$ ，从而 $\frac{1}{p(x)} = \frac{r(x)}{q(x)}$ ，即 $q(x) = p(x)r(x)$ ，所以 $p(x) \mid q(x)$ 。

(证毕)

上述定理说明可用 n 级 LFSR 产生的序列，也可用级数更多的 LFSR 来产生。

定义 2-2 设 $p(x)$ 是 $GF(2)$ 上的多项式，使 $p(x) \mid (x^p - 1)$ 的最小 p 称为 $p(x)$ 的周期或阶。

定理 2-3 若序列 $\{a_i\}$ 的特征多项式 $p(x)$ 定义在 $GF(2)$ 上， p 是 $p(x)$ 的周期，则 $\{a_i\}$ 的周期 $r \mid p$ 。

证明：由 $p(x)$ 周期的定义得 $p(x) \mid (x^p - 1)$ ，因此存在 $q(x)$ ，使得 $x^p - 1 = p(x)q(x)$ ，又由 $p(x)A(x) = \phi(x)$ 可得 $p(x)q(x)A(x) = \phi(x)q(x)$ ，所以 $(x^p - 1)A(x) = \phi(x)q(x)$ 。由于 $q(x)$ 的次数为 $p-n$ ， $\phi(x)$ 的次数不超过 $n-1$ ，所以 $(x^p - 1)A(x)$ 的次数不超过 $(p-n) + (n-1) = p-1$ 。这就证明了对于任意正整数 i 都有 $a_{i-p} = a_i$ 。

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 $r \mid p$ 。(证毕)

n 级 LFSR 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 LFSR 遍历 $2^n - 1$ 个非零状态,这时序列的周期达到最大 $2^n - 1$,这种序列就是 m 序列。显然对于特征多项式一样,而仅初始条件不同的两个输出序列,一个记为 $\{a_i^{(1)}\}$,另一个记为 $\{a_i^{(2)}\}$,其中一个必是另一个的移位,即存在一个常数 k ,使得

$$a_i^{(1)} = a_{i+k}^{(2)}, \quad i = 1, 2, \dots$$

下面讨论特征多项式满足什么条件时, LFSR 的输出序列为 m 序列。

定理 2-4 设 $p(x)$ 是 n 次不可约多项式,周期为 m , 序列 $\{a_i\} \in G(p(x))$, 则 $\{a_i\}$ 的周期为 m 。

证明: 设 $\{a_i\}$ 的周期为 r , 由定理 2-3 有 $r|m$, 所以 $r \leq m$ 。

设 $A(x)$ 为 $\{a_i\}$ 的生成函数, $A(x) = \frac{\phi(x)}{p(x)}$, 即 $p(x)A(x) = \phi(x) \neq 0$, $\phi(x)$ 的次数不超过 $n-1$ 。而

$$\begin{aligned} A(x) &= \sum_{i=1}^{\infty} a_i x^{i-1} = a_1 + a_2 x + \dots + a_r x^{r-1} + x^r (a_1 + a_2 x + \dots + a_r x^{r-1}) \\ &\quad + (x^r)^2 (a_1 + a_2 x + \dots + a_r x^{r-1}) + \dots \\ &= \frac{a_1 + a_2 x + \dots + a_r x^{r-1}}{1 - x^r} \\ &= \frac{a_1 + a_2 x + \dots + a_r x^{r-1}}{x^r - 1} \end{aligned}$$

于是 $A(x) = \frac{a_1 + a_2 x + \dots + a_r x^{r-1}}{x^r - 1} = \frac{\phi(x)}{p(x)}$, 即

$$p(x)(a_1 + a_2 x + \dots + a_r x^{r-1}) = \phi(x)(x^r - 1)$$

因 $p(x)$ 是不可约的, 所以 $\gcd(p(x), \phi(x)) = 1$, $p(x) | (x^r - 1)$, 因此 $m \leq r$ 。

综上 $r = m$ 。

(证毕)

定理 2-5 n 级 LFSR 产生的序列有最大周期 $2^n - 1$ 的必要条件是其特征多项式为不可约的。

证明: 设 n 级 LFSR 产生的序列周期达到最大 $2^n - 1$, 除 0 序列外, 每一序列的周期由特征多项式惟一决定, 而与初始状态无关。设特征多项式为 $p(x)$, 若 $p(x)$ 可约, 可设为 $p(x) = g(x)h(x)$, 其中 $g(x)$ 不可约, 且次数 $k < n$ 。由于 $G(g(x)) \subset G(p(x))$, 而 $G(g(x))$ 中序列的周期一方面不超过 $2^k - 1$, 另一方面又等于 $2^n - 1$, 这是矛盾的, 所以 $p(x)$ 不可约。

(证毕)

该定理的逆不成立, 即 LFSR 的特征多项式为不可约多项式时, 其输出序列不一定是 m 序列。

【例 2-4】 $f(x) = x^4 + x^3 + x^2 + x + 1$ 为 $GF(2)$ 上的不可约多项式, 这可由 $x, x+1, x^2 + x + 1$ 都不能整除 $f(x)$ 得到。以 $f(x)$ 为特征多项式的 LFSR 的输出序列可由

$$a_k = a_{k-1} \oplus a_{k-2} \oplus a_{k-3} \oplus a_{k-4} \quad (k \geq 4)$$

和给定的初始状态求出, 设初始状态为 0001, 则输出序列为 000110001100011..., 周期为 5, 不是 m 序列。

定义 2-3 若 n 次不可约多项式 $p(x)$ 的阶为 $2^n - 1$, 则称 $p(x)$ 是 n 次本原多项式。

定理 2-6 设 $\{a_i\} \in G(p(x))$, $\{a_i\}$ 为 m 序列的充要条件是 $p(x)$ 为本原多项式。

证明: 若 $p(x)$ 是本原多项式, 则其阶为 $2^n - 1$, 由定理 2-4 得 $\{a_i\}$ 的周期等于 $2^n - 1$, 即 $\{a_i\}$ 为 m 序列。

反之, 若 $\{a_i\}$ 为 m 序列, 即其周期等于 $2^n - 1$, 由定理 2-5 知 $p(x)$ 是不可约的。由定理 2-3 知 $\{a_i\}$ 的周期 $2^n - 1$ 整除 $p(x)$ 的阶, 而 $p(x)$ 的阶不超过 $2^n - 1$, 所以 $p(x)$ 的阶为 $2^n - 1$, 即 $p(x)$ 是本原多项式。 (证毕)

$\{a_i\}$ 为 m 序列的关键在于 $p(x)$ 为本原多项式, n 次本原多项式的个数为

$$\frac{\phi(2^n - 1)}{n}$$

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。所以对于任意的 n 级 LFSR, 至少存在一种连接方式使其输出序列为 m 序列。

【例 2-5】 设 $p(x) = x^4 + x + 1$, 由于 $p(x) \mid (x^{15} - 1)$, 但不存在小于 15 的常数 l , 使得 $p(x) \mid (x^l - 1)$, 所以 $p(x)$ 的阶为 15。 $p(x)$ 的不可约性可由 $x, x+1, x^2+x+1$ 都不能整除 $p(x)$ 得到, 所以 $p(x)$ 是本原多项式。

若 LFSR 以 $p(x)$ 为特征多项式, 则输出序列的递推关系为

$$a_k = a_{k-1} \oplus a_{k-4} \quad (k \geq 4)$$

若初始状态为 1001, 则输出为

$$100100011110101100100011110101\cdots$$

周期为 $2^4 - 1 = 15$, 即输出序列为 m 序列。

2.4 m 序列的伪随机性

流密码的安全性取决于密钥流的安全性, 要求密钥流序列有好的随机性, 以使密码分析者对它无法预测。也就是说, 即使截获其中一段, 也无法推测后面是什么。如果密钥流是周期的, 要完全做到随机性是困难的。严格地说, 这样的序列不可能做到随机, 只能要求截获比周期短的一段密钥流时不会泄露更多信息, 这样的序列称为伪随机序列。

为讨论 m 序列的随机性, 先要讨论随机序列的一般特性。

设 $\{a_i\} = (a_1 a_2 a_3 \cdots)$ 为 0、1 序列, 例如 00110111, 其前两个数字是 00, 称为 0 的 2 游

程;接着是11,是1的2游程;再下来是0的1游程和1的3游程。

定义 2-4 $GF(2)$ 上周期为 T 的序列 $\{a_i\}$ 的自相关函数定义为

$$R(\tau) = \frac{1}{T} \sum_{i=1}^T (-1)^{a_i} (-1)^{a_{i+\tau}}, \quad 0 \leq \tau \leq T-1$$

定义中的和式表示序列 $\{a_i\}$ 与 $\{a_{i+\tau}\}$ (序列 $\{a_i\}$ 向后平移 τ 位得到) 在一个周期内对应位相同的位数与对应位不同的位数之差。当 $\tau=0$ 时, $R(\tau)=1$; 当 $\tau \neq 0$ 时, 称 $R(\tau)$ 为异相自相关函数。

Golomb 对伪随机周期序列提出了应满足的如下 3 个随机性公设:

① 在序列的一个周期内, 0 与 1 的个数相差至多为 1。

② 在序列的一个周期内, 长为 i 的游程占游程总数的 $\frac{1}{2^i}$ ($i=1, 2, \dots$), 且在等长的游程中 0 的游程个数和 1 的游程个数相等。

③ 异相自相关函数是一个常数。

公设①说明 $\{a_i\}$ 中 0 与 1 出现的概率基本上相同, ②说明 0 与 1 在序列中每一位置上出现的概率相同; ③意味着通过对序列与其平移后的序列做比较, 不能给出其他任何信息。

从密码系统的角度看, 一个伪随机序列还应满足下面的条件:

① $\{a_i\}$ 的周期相当大。

② $\{a_i\}$ 的确定在计算上是容易的。

③ 由密文及相应的明文的部分信息, 不能确定整个 $\{a_i\}$ 。

下一定理说明, m 序列满足 Golomb 的 3 个随机性公设。

定理 2-7 $GF(2)$ 上的 n 长 m 序列 $\{a_i\}$ 具有如下性质:

① 在一个周期内, 0、1 出现的次数分别为 $2^{n-1}-1$ 和 2^{n-1} 。

② 在一个周期内, 总游程数为 2^{n-1} ; 对 $1 \leq i \leq n-2$, 长为 i 的游程有 2^{n-1-i} 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

$$R(\tau) = \begin{cases} 1, & \tau = 0 \\ -\frac{1}{2^n-1}, & 0 < \tau \leq 2^n-2 \end{cases}$$

证明: ① 在 n 长 m 序列的一个周期内, 除了全 0 状态外, 每个 n 长状态 (共有 2^n-1 个) 都恰好出现一次, 这些状态中有 2^{n-1} 个在 a_1 位是 1, 其余 $2^n-1-2^{n-1}=2^{n-1}-1$ 个状态在 a_1 位是 0。

② 对 $n=1, 2$, 易证结论成立。

对 $n > 2$, 当 $1 \leq i \leq n-2$ 时, n 长 m 序列的一个周期内, 长为 i 的 0 游程数目等于序列中如下形式的状态数目: $\underbrace{100\cdots 01}_{i \uparrow 0} * \cdots *$, 其中 $n-i-2$ 个 $*$ 可任取 0 或 1。这种状态共有 2^{n-i-2} 个。同理可得长为 i 的 1 游程数目也等于 2^{n-i-2} , 所以长为 i 的游程总数为 2^{n-i-1} 。

由于寄存器中不会出现全 0 状态, 所以不会出现 0 的 n 游程, 但必有一个 1 的 n 游程, 而且 1 的游程不会更大, 因为若出现 1 的 $n+1$ 游程, 就必然有两个相邻的全 1 状态, 但这是不可能的。这就证明了 1 的 n 游程必然出现在如下的串中:

$$\underbrace{011\cdots 10}_{n \uparrow 1}$$

当这 $n+2$ 位通过移位寄存器时, 便依次产生以下状态:

$$\underbrace{011\cdots 1}_{n-1 \uparrow 1} \quad \underbrace{11\cdots 1}_{n \uparrow 1} \quad \underbrace{11\cdots 10}_{n-1 \uparrow 1}$$

由于 $\underbrace{011\cdots 1}_{n-1 \uparrow 1}, \underbrace{11\cdots 10}_{n-1 \uparrow 1}$ 这两个状态只能各出现一次, 所以不会有 1 的 $n-1$ 游程。

0 的 $n-1$ 游程有一个:

$$\underbrace{100\cdots 01}_{n-1 \uparrow 0}$$

它产生 $\underbrace{100\cdots 0}_{n-1 \uparrow 0}$ 和 $\underbrace{00\cdots 01}_{n-1 \uparrow 0}$ 两个状态。

于是在一个周期内, 总游程数为

$$1 + 1 + \sum_{i=1}^{n-2} 2^{n-i-1} = 2^{n-1}$$

③ $\{a_i\}$ 是周期为 2^n-1 的 m 序列, 对于任一正整数 $\tau (0 < \tau < 2^n-1)$, $\{a_i\} + \{a_{i+\tau}\}$ 在一个周期内为 0 的位的数目正好是序列 $\{a_i\}$ 和 $\{a_{i+\tau}\}$ 对应位相同的位的数目。

设序列 $\{a_i\}$ 满足递推关系:

$$a_{h+n} = c_1 a_{h+n-1} \oplus c_2 a_{h+n-2} \oplus \cdots \oplus c_n a_h$$

故

$$a_{h+n+\tau} = c_1 a_{h+n+\tau-1} \oplus c_2 a_{h+n+\tau-2} \oplus \cdots \oplus c_n a_{h+\tau}$$

$$a_{h+n} \oplus a_{h+n+\tau} = c_1 (a_{h+n-1} \oplus a_{h+n+\tau-1}) \oplus c_2 (a_{h+n-2} \oplus a_{h+n+\tau-2}) \oplus \cdots \oplus c_n (a_h \oplus a_{h+\tau})$$

令 $b_i = a_i \oplus a_{i+\tau}$, 由递推序列 $\{a_i\}$ 可推得递推序列 $\{b_i\}$, $\{b_i\}$ 满足

$$b_{h+n} = c_1 b_{h+n-1} \oplus c_2 b_{h+n-2} \oplus \cdots \oplus c_n b_h$$

$\{b_i\}$ 也是 m 序列。为了计算 $R(\tau)$, 只要用 $\{b_i\}$ 在一个周期中 0 的个数减去 1 的个数, 再除以 2^n-1 , 即

$$R(\tau) = \frac{2^{n-1} - 1 - 2^{n-1}}{2^n - 1} = -\frac{1}{2^n - 1} \quad (\text{证毕})$$

2.5 m 序列密码的破译

上面说过,有限域 $GF(2)$ 上的二元加法流密码(如图 2-3)是目前最为常用的流密码体制,设滚动密钥生成器是线性反馈移位寄存器,产生的密钥是 m 序列。又设 S_h 和 S_{h+1} 是序列中两个连续的 n 长向量,其中

$$S_h = \begin{bmatrix} a_h \\ a_{h+1} \\ \vdots \\ a_{h+n-1} \end{bmatrix}, \quad S_{h+1} = \begin{bmatrix} a_{h+1} \\ a_{h+2} \\ \vdots \\ a_{h+n} \end{bmatrix}$$

设序列 $\{a_i\}$ 满足线性递推关系:

$$a_{h+n} = c_1 a_{h+n-1} \oplus c_2 a_{h+n-2} \oplus \cdots \oplus c_n a_h$$

可表示为

$$\begin{bmatrix} a_{h+1} \\ a_{h+2} \\ \vdots \\ a_{h+n} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ c_n & c_{n-1} & c_{n-2} & \cdots & c_1 \end{bmatrix} \begin{bmatrix} a_h \\ a_{h+1} \\ \vdots \\ a_{h+n-1} \end{bmatrix}$$

或 $S_{h+1} = M \cdot S_h$, 其中

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ c_n & c_{n-1} & c_{n-2} & \cdots & c_1 \end{bmatrix}$$

又设敌手知道一段长为 $2n$ 的明密文对,即已知

$$x = x_1 x_2 \cdots x_{2n}, y = y_1 y_2 \cdots y_{2n}$$

于是可求出一段长为 $2n$ 的密钥序列

$$z = z_1 z_2 \cdots z_{2n}$$

其中 $z_i = x_i \oplus y_i = x_i \oplus (x_i \oplus z_i)$ 。由此可推出线性反馈移位寄存器连续的 $n+1$ 个状态:

$$S_1 = (z_1 z_2 \cdots z_n) \xrightarrow{\text{记为}} (a_1 a_2 \cdots a_n)$$

$$S_2 = (z_2 z_3 \cdots z_{n+1}) \xrightarrow{\text{记为}} (a_2 a_3 \cdots a_{n+1})$$

...

$$S_{n+1} = (z_{n+1} z_{n+2} \cdots z_{2n}) \stackrel{\text{记为}}{=} (a_{n+1} a_{n+2} \cdots a_{2n})$$

做矩阵

$$X = (S_1 S_2 \cdots S_n)$$

而

$$\begin{aligned} (a_{n+1} a_{n+2} \cdots a_{2n}) &= (c_n c_{n-1} \cdots c_1) \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ a_2 & a_3 & \cdots & a_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n+1} & \cdots & a_{2n-1} \end{pmatrix} \\ &= (c_n c_{n-1} \cdots c_1) X \end{aligned}$$

若 X 可逆, 则

$$(c_n c_{n-1} \cdots c_1) = (a_{n+1} a_{n+2} \cdots a_{2n}) X^{-1}$$

下面证明 X 的确是可逆的。因为 X 是由 S_1, S_2, \cdots, S_n 作为列向量, 要证 X 可逆, 只要证明这 n 个向量线性无关。

由序列递推关系:

$$a_{h+n} = c_1 a_{h+n-1} \oplus c_2 a_{h+n-2} \oplus \cdots \oplus c_n a_h$$

可推出向量的递推关系:

$$S_{h+n} = c_1 S_{h+n-1} \oplus c_2 S_{h+n-2} \oplus \cdots \oplus c_n S_h = \sum_{i=1}^n c_i S_{h+n-i} \pmod{2}$$

设 $m(m \leq n+1)$ 是使 S_1, S_2, \cdots, S_m 线性相关的最小整数, 即存在不全为 0 的系数 l_1, l_2, \cdots, l_m , 其中不妨设 $l_1 = 1$, 使得

$$S_m + l_2 S_{m-1} + l_3 S_{m-2} + \cdots + l_m S_1 = 0$$

即

$$S_m = l_m S_1 + l_{m-1} S_2 + \cdots + l_2 S_{m-1} = \sum_{j=1}^{m-1} l_{j+1} S_{m-j}$$

对于任一整数 i 有

$$\begin{aligned} S_{m+i} &= M^i S_m = M^i (l_m S_1 + l_{m-1} S_2 + \cdots + l_2 S_{m-1}) \\ &= l_m M^i S_1 + l_{m-1} M^i S_2 + \cdots + l_2 M^i S_{m-1} \\ &= l_m S_{i+1} + l_{m-1} S_{i+2} + \cdots + l_2 S_{m+i-1} \end{aligned}$$

由此又推出密钥流的递推关系:

$$a_{m+i} = l_2 a_{m+i-1} \oplus l_3 a_{m+i-2} \oplus \cdots \oplus l_m a_{i+1}$$

即密钥流的级数小于 m 。若 $m \leq n$, 则得出密钥流的级数小于 n , 矛盾。所以 $m = n+1$, 从而矩阵 X 必是可逆的。

【例 2-6】 设敌手得到密文串 101101011110010 和相应的明文串 01100111111001, 因此可计算出相应的密钥流为 110100100001011。进一步假定敌手还知道密钥流是使用 5 级线性反馈移位寄存器产生的, 那么敌手可分别用密文串中的前 10 个比特和明文串中的前 10 个比特建立如下方程

$$(a_6 a_7 a_8 a_9 a_{10}) = (c_5 c_4 c_3 c_2 c_1) \begin{pmatrix} a_1 a_2 a_3 a_4 a_5 \\ a_2 a_3 a_4 a_5 a_6 \\ a_3 a_4 a_5 a_6 a_7 \\ a_4 a_5 a_6 a_7 a_8 \\ a_5 a_6 a_7 a_8 a_9 \end{pmatrix}$$

即

$$(0 \ 1 \ 0 \ 0 \ 0) = (c_5 c_4 c_3 c_2 c_1) \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

而

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

从而得到

$$(c_5 c_4 c_3 c_2 c_1) = (01000) \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

所以

$$(c_5 c_4 c_3 c_2 c_1) = (10010)$$

密钥流的递推关系为

$$a_{i+5} = c_5 a_i \oplus c_2 a_{i+3} = a_i \oplus a_{i+3}$$

2.6

非线性序列

在 2.1.3 节已介绍密钥流生成器可分解为驱动子系统和非线性组合子系统,如图 2-6 所示。驱动子系统常用一个或多个线性反馈移位寄存器来实现,非线性组合子系统用非线性组合函数 F 来实现,如图 2-7 所示。本节介绍第 2 部分非线性组合子系统。

为了使密钥流生成器输出的二元序列尽可能复杂,应保证其周期尽可能大、线性复杂度和不可预测性尽可能高,因此常使用多个 LFSR 来构造二元序列,称每个 LFSR 的输出序列为驱动序列,显然密钥流生成器输出序列的周期不大于各驱动序列周期的乘积,因此,提高输出序列的线性复杂度应从极大化其周期开始。

二元序列的线性复杂度指生成该序列的最短 LFSR 的级数,最短 LFSR 的特征多项式称为二元序列的极小特征多项式。

下面介绍 4 种由多个 LFSR 驱动的非线性序列生成器。

2.6.1 Geffe 序列生成器

Geffe 序列生成器由 3 个 LFSR 组成,其中 LFSR2 作为控制生成器使用,如图 2-12 所示。

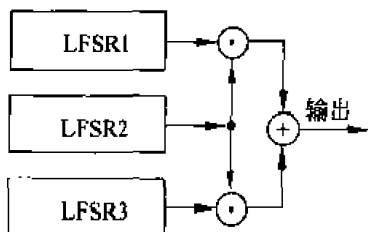


图 2-12 Geffe 序列生成器

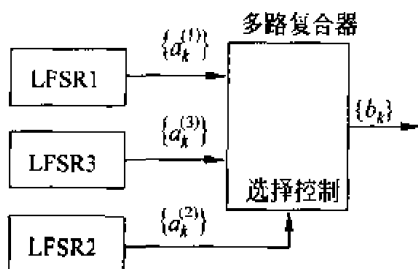


图 2-13 多路复合器表示的 Geffe 序列生成器

当 LFSR2 输出 1 时, LFSR2 与 LFSR1 相连接; 当 LFSR2 输出 0 时, LFSR2 与 LFSR3 相连接。若设 LFSR i 的输出序列为 $\{a_k^{(i)}\}$ ($i=1, 2, 3$), 则输出序列 $\{b_k\}$ 可以表示为

$$b_k = a_k^{(1)} a_k^{(2)} + a_k^{(3)} \overline{a_k^{(2)}} = a_k^{(1)} a_k^{(2)} + a_k^{(3)} a_k^{(2)} + a_k^{(3)}$$

Geffe 序列生成器也可以表示为图 2-13 的形式, 其中 LFSR1 和 LFSR3 作为多路复合器的输入, LFSR2 控制多路复合器的输出。设 LFSR i 的特征多项式分别为 n_i 次本原多项式, 且 n_i 两两互素, 则 Geffe 序列的周期为

$$\prod_{i=1}^s (2^{n_i} - 1)$$

线性复杂度为

$$(n_1 + n_s)n_2 + n_3$$

Geffe 序列的周期实现了极大化,且 0 与 1 之间的分布大体上是平衡的。

2.6.2 J-K 触发器

J-K 触发器如图 2-14 所示,它的两个输入端分别用 J 和 K 表示,其输出 c_k 不仅依赖于输入,还依赖于前一个输出位 c_{k-1} ,即

$$c_k = (\overline{x_1 + x_2})c_{k-1} + x_1$$

其中 x_1 和 x_2 分别是 J 和 K 端的输入。由此可得 J-K 触发器的真值表,如表 2-3。

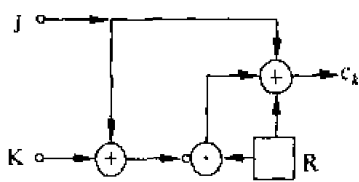


图 2-14 J-K 触发器

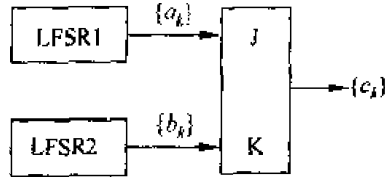


图 2-15 利用 J-K 触发器的非线性序列生成器

表 2-3 J-K 触发器的真值表

J	K	c_k
0	0	c_{k-1}
0	1	0
1	0	1
1	1	$\overline{c_{k-1}}$

利用 J-K 触发器的非线性序列生成器见图 2-15。

在图 2-15 中,令驱动序列 $\{a_k\}$ 和 $\{b_k\}$ 分别为 m 级和 n 级 m 序列,则有

$$c_k = (\overline{a_k + b_k})c_{k-1} + a_k = (a_k + b_k + 1)c_{k-1} + a_k$$

如果令 $c_{-1} = 0$,则输出序列的最初 3 项为

$$c_0 = a_0$$

$$c_1 = (a_1 + b_1 + 1)a_0 + a_1$$

$$c_2 = (a_2 + b_2 + 1)((a_1 + b_1 + 1)a_0 + a_1) + a_2$$

当 m 与 n 互素且 $a_0 + b_0 = 1$ 时,序列 $\{c_k\}$ 的周期为 $(2^m - 1)(2^n - 1)$ 。

【例 2-7】 令 $m=2, n=3$,两个驱动 m 序列分别为

$$\{a_k\} = 0, 1, 1, \dots$$

和

$$\{b_k\} = 1, 0, 0, 1, 0, 1, 1, \dots$$

于是, 输出序列 $\{c_k\}$ 是 $0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, \dots$, 其周期为 $(2^2 - 1)(2^3 - 1) = 21$ 。

由表达式 $c_k = (a_k + b_k + 1)c_{k-1} + a_k$ 可得

$$c_k = \begin{cases} a_k, & c_{k-1} = 0 \\ \overline{b_k}, & c_{k-1} = 1 \end{cases}$$

因此, 如果知道 $\{c_k\}$ 中相邻位的值 c_{k-1} 和 c_k , 就可以推断出 a_k 和 b_k 中的一个。而一旦知道足够多的这类信息, 就可通过密码分析的方法得到序列 $\{a_k\}$ 和 $\{b_k\}$ 。为了克服上述缺点, Pless 提出了由多个 J-K 触发器序列驱动的多路复合序列方案, 称为 Pless 生成器。

2.6.3 Pless 生成器

Pless 生成器由 8 个 LFSR、4 个 J-K 触发器和 1 个循环计数器构成, 由循环计数器进行选通控制, 如图 2-16 所示。假定在时刻 t 输出第 $t \pmod{4}$ 个单元, 则输出序列为

$$a_0 b_1 c_2 d_3 a_4 b_5 d_6$$

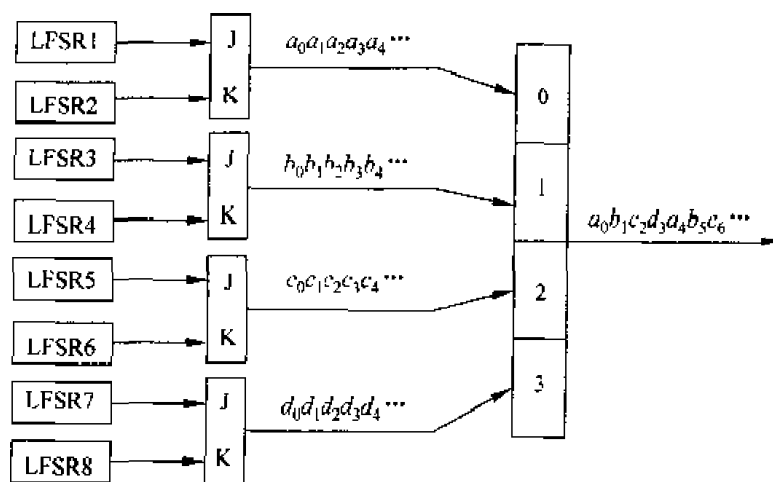


图 2-16 Pless 生成器

2.6.4 钟控序列生成器

钟控序列最基本的模型是用一个 LFSR 控制另外一个 LFSR 的移位时钟脉冲, 如图 2-17 所示。

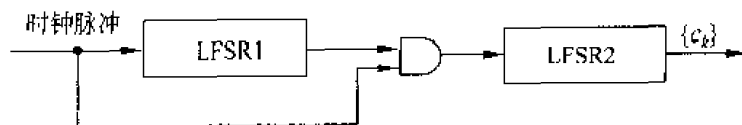


图 2-17 最简单的钟控序列生成器

假设 LFSR1 和 LFSR2 分别输出序列 $\{a_k\}$ 和 $\{b_k\}$ ，其周期分别为 p_1 和 p_2 。当 LFSR1 输出 1 时，移位时钟脉冲通过与门使 LFSR2 进行一次移位，从而生成下一位。当 LFSR1 输出 0 时，移位时钟脉冲无法通过与门影响 LFSR2。因此 LFSR2 重复输出前一位。假设钟控序列 $\{c_k\}$ 的周期为 p ，可得如下关系：

$$p = \frac{p_1 p_2}{\gcd(w_1, p_2)}$$

其中 $w_1 = \sum_{i=0}^{p_1-1} a_i$ 。

又设 $\{a_k\}$ 和 $\{b_k\}$ 的极小特征多项式分别为 $GF(2)$ 上的 m 和 n 次本原多项式 $f_1(x)$ 和 $f_2(x)$ ，且 $m|n$ 。因此， $p_1 = 2^m - 1$ ， $p_2 = 2^n - 1$ 。又知 $w_1 = 2^{m-1}$ ，因此 $\gcd(w_1, p_2) = 1$ ，所以 $p = p_1 p_2 = (2^m - 1)(2^n - 1)$ 。

此外，也可推导出 $\{c_k\}$ 的线性复杂度为 $n(2^m - 1)$ ，极小特征多项式为 $f_2(x^{2^m - 1})$ 。

【例 2-8】 设 LFSR1 为 3 级 m 序列生成器，其特征多项式为 $f_1(x) = 1 + x + x^3$ 。设初态为 $a_0 = a_1 = a_2 = 1$ ，于是输出序列为 $\{a_k\} = 1, 1, 1, 0, 1, 0, 0, \dots$ 。

又设 LFSR2 为 3 级 m 序列生成器，且记其状态向量为 σ_k ，则在图 2-17 的构造下 σ_k 的变化情况如下：

σ_0	σ_1	σ_2	σ_3	σ_3	σ_4	σ_4	σ_4
	σ_5	σ_6	σ_6	σ_0	σ_1	σ_1	σ_1
	σ_2	σ_3	σ_4	σ_4	σ_5	σ_5	σ_5
	σ_6	σ_0	σ_1	σ_1	σ_2	σ_2	σ_2
	σ_0	σ_1	σ_2	σ_2	σ_3	σ_3	σ_3
	σ_4	σ_5	σ_6	σ_6	σ_0	σ_0	\dots

$\{c_k\}$ 的周期为 $(2^3 - 1)^2 = 49$ ，在它的一个周期内，每个 σ_k 恰好出现 7 次。

设 $f_2(x) = 1 + x^2 + x^3$ 为 LFSR2 的特征多项式，且初态为 $b_0 = b_1 = b_2 = 1$ ，则 $\{b_k\} = 1, 1, 1, 0, 0, 1, 0, \dots$ 。

由 σ_k 的变化情况得

$$\begin{aligned} \{c_k\} = & 1, 1, 1, 0, 0, 0, 0, 0, \\ & 1, 0, 1, 1, 1, 1, 1, \end{aligned}$$

1,0,0,0,1,1,1,
 0,1,1,1,1,1,1,
 0,0,1,1,0,0,0,
 1,1,1,1,0,0,0,
 0,1,0,0,1,1,...

$\{c_k\}$ 的极小特征多项式为 $1+x^{14}x^{21}$, 其线性复杂度为 $3 \cdot (2^3 - 1) = 21$, 图 2-18 是其线性等价生成器。

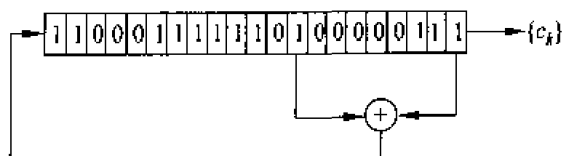


图 2-18 一个钟控序列的线性等价生成器

实际应用中, 可以用上述最基本的钟控序列生成器构造复杂的模型, 具体构造方式可参阅有关文献。

设计一个性能良好的序列密码是一项十分困难的任务。最基本的设计原则是“密钥流生成器的不可预测性”, 它可分解为下述基本原则:

- ① 长周期。
- ② 高线性复杂度。
- ③ 统计性能良好。
- ④ 足够的“混乱”。
- ⑤ 足够的“扩散”。
- ⑥ 抵抗不同形式的攻击。

习 题

1. 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, 设其初始状态为 $(a_1, a_2, a_3) = (1, 0, 1)$, 求各线性反馈函数的输出序列及周期。

2. 设 n 级线性反馈移位寄存器的特征多项式为 $p(x)$, 初始状态为 $(a_1, a_2, \dots, a_{n-1}, a_n) = (00 \dots 01)$, 证明输出序列的周期等于 $p(x)$ 的阶。

3. 设 $n=4$, $f(a_1, a_2, a_3, a_4) = a_1 \oplus a_4 \oplus 1 \oplus a_2 a_3$, 初始状态为 $(a_1, a_2, a_3, a_4) = (1, 1, 0, 1)$, 求此非线性反馈移位寄存器的输出序列及周期。

4. 已知流密码的密文串 1010110110 和相应的明文串 0100010001, 而且还已知密钥

流是使用 3 级线性反馈移位寄存器产生的,试破译该密码系统。

5. 设 J-K 触发器中 $\{a_k\}$ 和 $\{b_k\}$ 分别为 3 级和 4 级 m 序列,且

$$\{a_k\} = 11101001110100\cdots$$

$$\{b_k\} = 001011011011000001011011011000\cdots$$

求输出序列 $\{c_k\}$ 及周期。

6. 设基本钟控序列产生器中 $\{a_k\}$ 和 $\{b_k\}$ 分别为 2 级和 3 级 m 序列,且

$$\{a_k\} = 101101\cdots$$

$$\{b_k\} = 10011011001101\cdots$$

求输出序列 $\{c_k\}$ 及周期。

第3章

分组密码体制

3.1 分组密码概述

在许多密码系统中,单钥分组密码是系统安全的一个重要组成部分,用分组密码易于构造伪随机数生成器、流密码、消息认证码(MAC)和杂凑函数等,还可进而成为消息认证技术、数据完整性机制、实体认证协议以及单钥数字签身体制的核心组成部分。实际应用中对于分组密码可能提出多方面的要求,除了安全性外,还有运行速度、存储量(程序的长度、数据分组长度、高速缓存大小)、实现平台(硬件、软件、芯片)、运行模式等限制条件。这些都需要与安全性要求之间进行适当的折中选择。

分组密码是将明文消息编码表示后的数字序列 $x_0, x_1, \dots, x_{t-1}, \dots$ 划分成长为 n 的组 $x = (x_0, x_1, \dots, x_{n-1})$, 各组(长为 n 的矢量)分别在密钥 $k = (k_0, k_1, \dots, k_{t-1})$ 控制下变换成等长的输出数字序列 $y = (y_0, y_1, \dots, y_{m-1})$ (长为 m 的矢量), 其加密函数 $E: V_n \times K \rightarrow V_m$, V_n 和 V_m 分别是 n 维和 m 维矢量空间, K 为密钥空间, 如图 3-1 所示。它与流密码不同之处在于输出的每一位数字不是只与相应时刻输入的明文数字有关, 而是与一组长为 n 的明文数字有关。在相同密钥下, 分组密码对长为 n 的输入明文组所实施的变换是等同的, 所以只需研究对任一组明文数字的变换规则。这种密码实质上是字长为 n 的数字序列的代换密码。

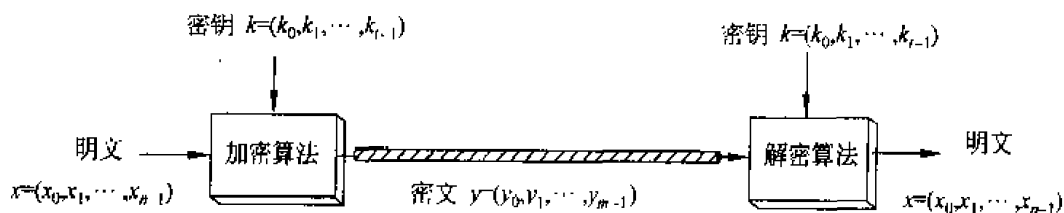


图 3-1 分组密码框图

通常取 $m=n$ 。若 $m>n$, 则为有数据扩展的分组密码; 若 $m<n$, 则为有数据压缩的分

组密码。在二元情况下, x 和 y 均为二元数字序列, 它们的每个分量 $x_i, y_i \in GF(2)$ 。本节将主要讨论二元情况。设计的算法应满足下述要求:

① 分组长度 n 要足够大, 使分组代换字母表中的元素个数 2^n 足够大, 防止明文穷举攻击法奏效。DES、IDEA、FEAL 和 LOKI 等分组密码都采用 $n=64$, 在生日攻击下用 2^{32} 组密文成功概率为 $1/2$, 同时要求 $2^{32} \times 64b = 2^{15}$ MB 存贮, 故采用穷举攻击是不现实的。

② 密钥量要足够大(即置换子集中的元素足够多), 尽可能消除弱密钥并使所有密钥同等地好, 以防止密钥穷举攻击奏效。但密钥又不能过长, 以便于密钥的管理。DES 采用 56 比特密钥, 看来太短了, IDEA 采用 128 比特密钥, 据估计, 在今后 30~40 年内采用 80 比特密钥是足够安全的。

③ 由密钥确定置换的算法要足够复杂, 充分实现明文与密钥的扩散和混淆, 没有简单的关系可循, 能抗击各种已知的攻击, 如差分攻击和线性攻击; 还要有高的非线性阶数, 实现复杂的密码变换; 使他人破译时除了用穷举法外, 无其他捷径可循。

④ 加密和解密运算简单, 易于软件和硬件高速实现。如将分组 n 划分为子段, 每段长为 8、16 或者 32。在以软件实现时, 应选用简单的运算, 使作用于子段上的密码运算易于以标准处理器的基本运算, 如加、乘、移位等实现, 避免用以软件难于实现的逐比特置换。为了便于硬件实现, 加密和解密过程之间的差别应仅在于由秘密密钥所生成的密钥表不同而已。这样, 加密和解密就可用同一器件实现。设计的算法采用规则的模块结构, 如多轮迭代等, 以便于软件和 VLSI 快速实现。此外, 差错传播和数据扩展要尽可能地小。

⑤ 数据扩展。一般无数据扩展, 在采用同态置换和随机化加密技术时可引入数据扩展。

⑥ 差错传播尽可能地小。

要实现上述几点要求并不容易。首先, 要在理论上研究有效而可靠的设计方法, 而后进行严格的安全性检验, 并且要易于实现。

下面介绍设计分组密码时的一些常用方法。

3.1.1 代换

如果明文和密文的分组长都为 n 比特, 则明文的每一个分组都有 2^n 个可能的取值。为使加密运算可逆(使解密运算可行), 明文的每一个分组都应产生惟一的一个密文分组, 这样的变换是可逆的, 称明文分组到密文分组的可逆变换为代换。不同可逆变换的个数有 $2^n!$ 个。

图 3-2 表示 $n=4$ 的代换密码的一般结构, 4 比特输入产生 16 个可能输入状态中的一个, 由代换结构将这一状态映射为 16 个可能输出状态中的一个, 每一输出状态由 4 个密文比特表示。加密映射和解密映射可由代换表来定义, 如表 3-1 所示。这种定义法是

分组密码最常用的形式,能用于定义明文和密文之间的任何可逆映射。

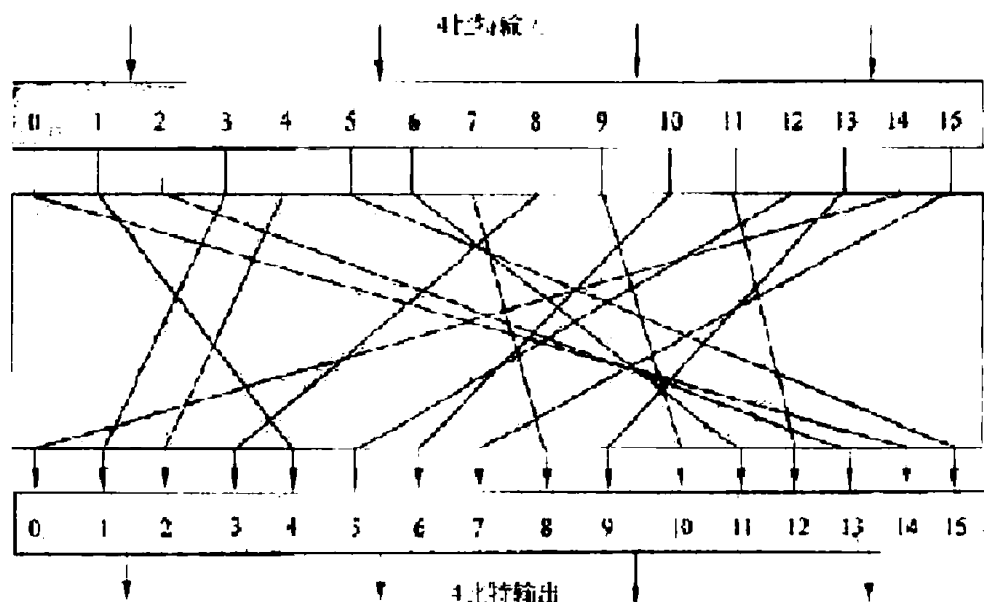


图 3-2 代换结构

但这种代换结构在实用中还有一些问题需考虑。如果分组长度太小,如 $n=4$,系统则等价于古典的代换密码,容易通过对明文的统计分析而被攻破。这个弱点不是代换结构固有的,只是因为分组长度太小。如果分组长度 n 足够大,而且从明文到密文可有任意可逆的代换,那么明文的统计特性将被隐藏而使以上的攻击不能奏效。

表 3-1 图 3-2 对应的代换表

明文	密文	明文	密文	密文	明文	密文	明文
0000	1110	1000	0011	0000	1110	1000	0111
0001	0100	1001	1010	0001	0011	1001	1101
0010	1101	1010	0110	0010	0100	1010	1001
0011	0001	1011	1100	0011	1000	1011	0110
0100	0010	1100	0101	0100	0001	1100	1011
0101	1111	1101	1001	0101	1100	1101	0010
0110	1011	1110	0000	0110	1010	1110	0000
0111	1000	1111	0111	0111	1111	1111	0101

然而,从实现的角度来看,分组长度很大的可逆代换结构是不实际的。仍以表 3-1 为例,该表定义了 $n=4$ 时从明文到密文的一个可逆映射,其中第 2 列是每个明文分组对应

的密文分组的值,可用来定义这个可逆映射。因此从本质上来说,第2列是从所有可能映射中决定某一特定映射的密钥。这个例子中,密钥需要64比特。一般地,对 n 比特的代换结构,密钥的大小是 $n \times 2^n$ 比特。如对64比特的分组,密钥大小应是 $64 \times 2^{64} = 2^{71} \approx 10^{21}$ 比特,因此难以处理。实际中常将 n 分成较小的段,例如可选 $n = r \cdot n_0$,其中 r 和 n_0 都是正整数,将设计 n 个变量的代换变为设计 r 个较小的子代换,而每个子代换只有 n_0 个输入变量。一般 n_0 都不太大,称每个子代换为代换盒,简称为S盒。例如DES中将输入为48比特、输出为32比特的代换用8个S盒来实现,每个S盒的输入端数仅为6比特,输出端数仅为4比特。

3.1.2 扩散和混淆

扩散和混淆是由Shannon提出的设计密码系统的两个基本方法,目的是抗击敌手对密码系统的统计分析。如果敌手知道明文的某些统计特性,如消息中不同字母出现的频率、可能出现的特定单词或短语,而且这些统计特性以某种方式在密文中反映出来,那么敌手就有可能得出加密密钥或其一部分,或者得出包含加密密钥的一个可能的密钥集合。在Shannon称之为理想密码的密码系统中,密文的所有统计特性都与所使用的密钥独立。图3-2讨论的代换密码就是这样的一个密码系统,然而它是不实用的。

所谓扩散,就是将明文的统计特性散布到密文中去,实现方式是使得明文的每一位影响密文中多位的值,等价于说密文中每一位均受明文中多位影响。例如对英文消息 $M = m_1 m_2 m_3 \dots$ 的加密操作

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

其中 $ord(m_i)$ 是求字母 m_i 对应的序号, $chr(i)$ 是求序号 i 对应的字母。这时明文的统计特性将被散布到密文中,因而每一字母在密文中出现的频率比在明文出现的频率更接近于相等,双字母及多字母出现的频率也更接近于相等。在二元分组密码中,可对数据重复执行某个置换,再对这一置换作用于一函数,可获得扩散。

分组密码在将明文分组依靠密钥变换到密文分组时,扩散的目的是使明文和密文之间的统计关系变得尽可能复杂,以使敌手无法得到密钥;混淆是使密文和密钥之间的统计关系变得尽可能复杂,以使敌手无法得到密钥。因此即使敌手能得到密文的一些统计关系,由于密钥和密文之间的统计关系复杂化,敌手也无法得到密钥。使用复杂的代换算法可以得到预期的混淆效果,而简单的线性代换函数得到的混淆效果则不够理想。

扩散和混淆成功地实现了分组密码的本质属性,因而成为设计现代分组密码的基础。

3.1.3 Feistel 密码结构

很多分组密码的结构从本质上说都是基于一个称为 Feistel 网络的结构。Feistel 提出利用乘积密码可获得简单的代换密码,乘积密码指顺序地执行两个或多个基本密码系统,使得最后结果的密码强度高于每个基本密码系统产生的结果,Feistel 还提出了实现代换和置换的方法。其思想实际上是 Shannon 提出的利用乘积密码实现混淆和扩散思想的具体应用。

1. Feistel 加密结构

图 3-3 是 Feistel 网络示意图,加密算法的输入是分组长为 $2w$ 的明文和一个密钥 K 。将每组明文分成左右两半 L_0 和 R_0 ,在进行完 n 轮迭代后,左右两半再合并到一起以产生密文分组。其第 i 轮迭代的输入为前一轮输出的函数:

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

其中 K_i 是第 i 轮用的子密钥,由加密密钥 K 得到。一般地,各轮子密钥彼此不同而且与 K 也不同。

Feistel 网络中每轮结构都相同,每轮中右半数据被作用于轮函数 F 后,再与左半数据进行异或运算,这一过程就是上面介绍的代换。每轮的轮函数的结构都相同,但以不同的子密钥 K_i 作为参数。代换过程完成后,再交换左、右两半数据,这一过程称为置换。这种结构是 Shannon 提出的代换—置换网络(substitution-permutation network, SPN)的特有形式。

Feistel 网络的实现与以下参数和特性有关:

① 分组大小 分组越大则安全性越高,但加密速度就越慢。分组密码设计中最为普遍使用的分组大小是 64 比特。

② 密钥大小 密钥越长则安全性越高,但加密速度就越慢。现在普遍认为 64 比特或更短的密钥长度是不安全的,通常使用 128 比特的密钥长度。

③ 轮数 单轮结构远不足以保证安全性,但多轮结构可提供足够的安全性。典型地,轮数取为 16。

④ 子密钥产生算法 该算法的复杂性越大,则密码分析的困难性就越大。

⑤ 轮函数 轮函数的复杂性越大,密码分析的困难性也越大。

在设计 Feistel 网络时,还有以下两个方面需要考虑:

① 快速的软件实现 在很多情况中,算法是被镶嵌在应用程序中,因而无法用硬件

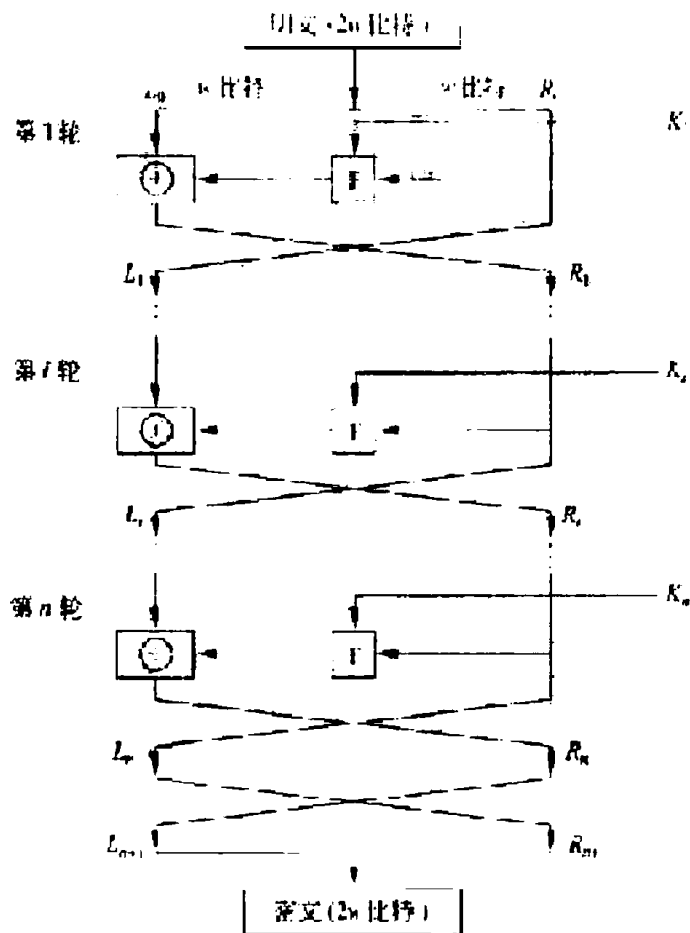


图 3-3 Feistel 网络示意图

实现。此时算法的执行速度是考虑的关键。

② 算法容易分析 如果算法能被无疑义地解释清楚,就可容易地分析算法抵抗攻击的能力,有助于设计高强度的算法。

2. Feistel 解密结构

Feistel 解密过程本质上和加密过程是一样的,算法使用密文作为输入,但使用子密钥 K_i 的次序与加密过程相反,即第 1 轮使用 K_n ,第 2 轮使用 K_{n-1}, \dots ,最后…轮使用 K_1 。这一特性保证了解密和加密可采用同一算法。

图 3-4 的左边表示 16 轮 Feistel 网络的加密过程,右边表示解密过程,加密过程由上而下,解密过程由下而上。为清楚起见,加密算法每轮的左右两半用 LE_i 和 RE_i 表示,解密算法每轮的左右两半用 LD_i 和 RD_i 表示。图中右边标了解密过程中每一轮的中间值与左边加密过程中中间值的对应关系,即加密过程第 i 轮的输出是 $LE_i \parallel RE_i$ (\parallel 表示链

接),解密过程第 $16-i$ 轮相应的输入是 $RD_i \parallel LD_i$ 。

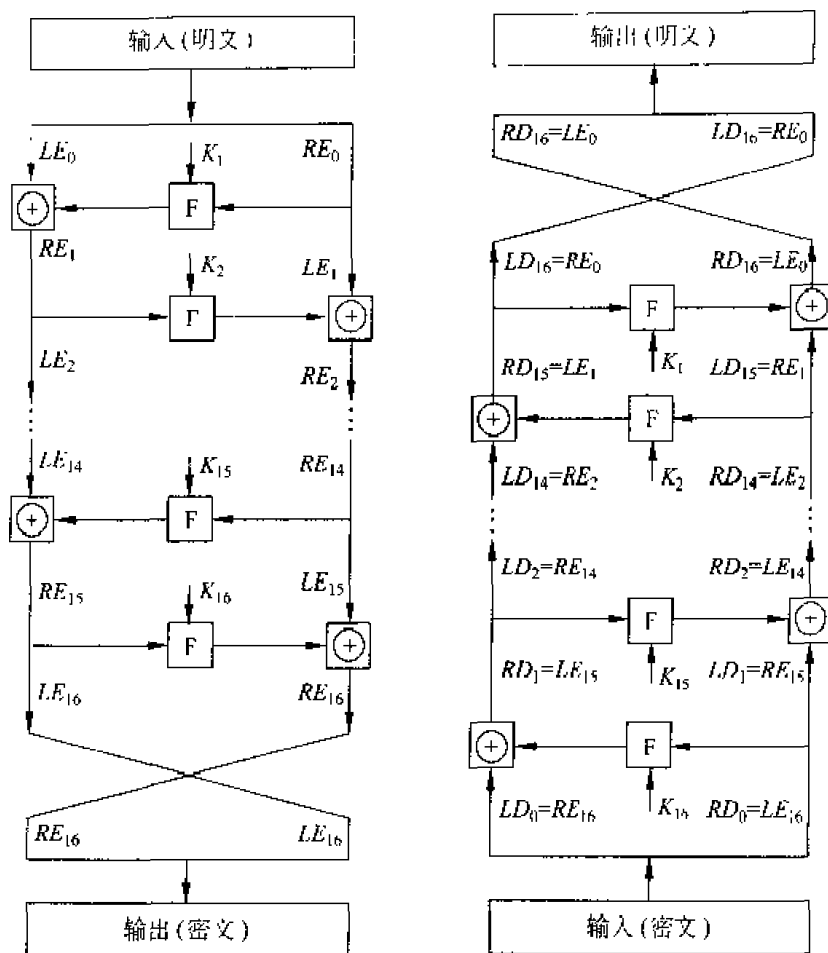


图 3-4 Feistel 加解密过程

加密过程的最后一轮执行完后,两半输出再经交换,因此密文是 $RE_{16} \parallel LE_{16}$ 。解密过程取以上密文作为同一算法的输入,即第 1 轮输入是 $RE_{16} \parallel LE_{16}$,等于加密过程第 16 轮两半输出交换后的结果。下面证明解密过程第 1 轮的输出等于加密过程第 16 轮输入左右两半的交换值。

在加密过程中:

$$\begin{aligned} LE_{16} &= RE_{15} \\ RE_{16} &= LE_{15} \oplus F(RE_{15}, K_{16}) \end{aligned}$$

在解密过程中:

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

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

所以解密过程第 1 轮的输出为 $LE_{15} \parallel RE_{15}$, 等于加密过程第 16 轮输入左右两半交换后的结果。容易证明这种对应关系在 16 轮中每轮都成立。一般地, 加密过程的第 i 轮有

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

因此

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

以上两式描述了加密过程中第 i 轮的输入与第 i 轮输出的函数关系, 由此关系可得图 3-4 右边显示的 LD_i 和 RD_i 的取值关系。

最后可以看到, 解密过程第 16 轮的输出是 $RE_0 \parallel LE_0$, 左右两半再经一次交换后即得最初的明文。

3.2 数据加密标准

数据加密标准(data encryption standard, DES)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, 它的分组长度为 64 比特, 密钥长度为 56 比特, 它是由美国 IBM 公司研制的, 是早期的称作 Lucifer 密码的一种发展和修改。DES 在 1975 年 3 月 17 日首次被公布在联邦记录中, 经过大量的公开讨论后, DES 于 1977 年 1 月 15 日被正式批准并作为美国联邦信息处理标准, 即 FIPS-46, 同年 7 月 15 日开始生效。规定每隔 5 年由美国国家保密局(national security agency, NSA)作出评估, 并重新批准它是否继续作为联邦加密标准。最近的一次评估是在 1994 年 1 月, 美国已决定 1998 年 12 月以后将不再使用 DES。1997 年 DESCHALL 小组经过近 4 个月的努力, 通过 Internet 搜索了 3×10^{16} 个密钥, 找出了 DES 的密钥, 恢复出了明文。1998 年 5 月美国 EFF(electronics frontier foundation)宣布, 他们以一台价值 20 万美元的计算机改装成的专用解密机, 用 56 小时破译了 56 比特密钥的 DES。美国国家标准和技术协会已征集并进行了几轮评估、筛选, 产生了称之为 AES(advanced encryption standard)的新加密标准。尽管如此, DES 对于推动密码理论的发展和应用毕竟起了重大作用, 对于掌握分组密码的基本理论、设计思想和实际应用仍然有着重要的参考价值, 下面首先来描述这一算法。

3.2.1 DES 描述

图 3-5 是 DES 加密算法的框图,其中明文分组长为 64 比特,密钥长为 56 比特。图的左边是明文的处理过程,有 3 个阶段,首先是一个初始置换 IP,用于重排明文分组的 64 比特数据。然后是具有相同功能的 16 轮变换,每轮中都有置换和代换运算,第 16 轮变换的输出分为左右两半,并被交换次序。最后再经过一个逆初始置换 IP^{-1} (为 IP 的逆)从而产生 64 比特的密文。除初始置换和逆初始置换外,DES 的结构和图 3-3 所示的 Feistel 密码结构完全相同。

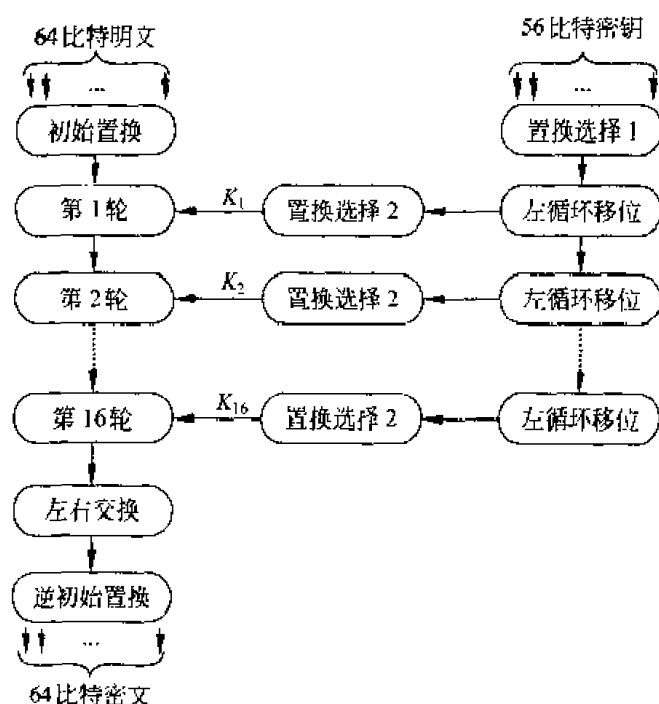


图 3-5 DES 加密算法框图

图 3-5 的右边是使用 56 比特密钥的方法。密钥首先通过一个置换函数,然后,对加密过程的每一轮,通过一个左循环移位和一个置换产生一个子密钥。其中每轮的置换都相同,但由于密钥被重复迭代,所以产生的每轮子密钥不相同。

1. 初始置换

DES 的置换表见表 3-2。

表 3-2(a)和表 3-2(b)分别给出了初始置换和逆初始置换的定义,为了显示这两个置换的确是彼此互逆的,考虑下面 64 比特的输入 M :

表 3-2 DES 的置换表

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

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

(c) 选择扩展运算 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

(d) 置换运算 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

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

其中 M_i 是二元数字。由表 3-2(a) 得 $X=IP(M)$ 为:

M_{58} M_{60} M_{42} M_{34} M_{26} M_{18} M_{10} M_2
 M_{60} M_{52} M_{44} M_{36} M_{28} M_{20} M_{12} M_4
 M_{62} M_{54} M_{46} M_{38} M_{30} M_{22} M_{14} M_6
 M_{64} M_{56} M_{48} M_{40} M_{32} M_{24} M_{16} M_8
 M_{57} M_{49} M_{41} M_{33} M_{25} M_{17} M_9 M_1

$$\begin{array}{cccccccc} M_{59} & M_{51} & M_{43} & M_{35} & M_{27} & M_{19} & M_{11} & M_3 \\ M_{61} & M_{53} & M_{45} & M_{37} & M_{29} & M_{21} & M_{13} & M_5 \\ M_{63} & M_{55} & M_{47} & M_{39} & M_{31} & M_{23} & M_{15} & M_7 \end{array}$$

如果再取逆初始置换 $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, 可以看出, M 各位的初始顺序将被恢复。

2. 轮结构

图 3-6 是 DES 加密算法的轮结构。

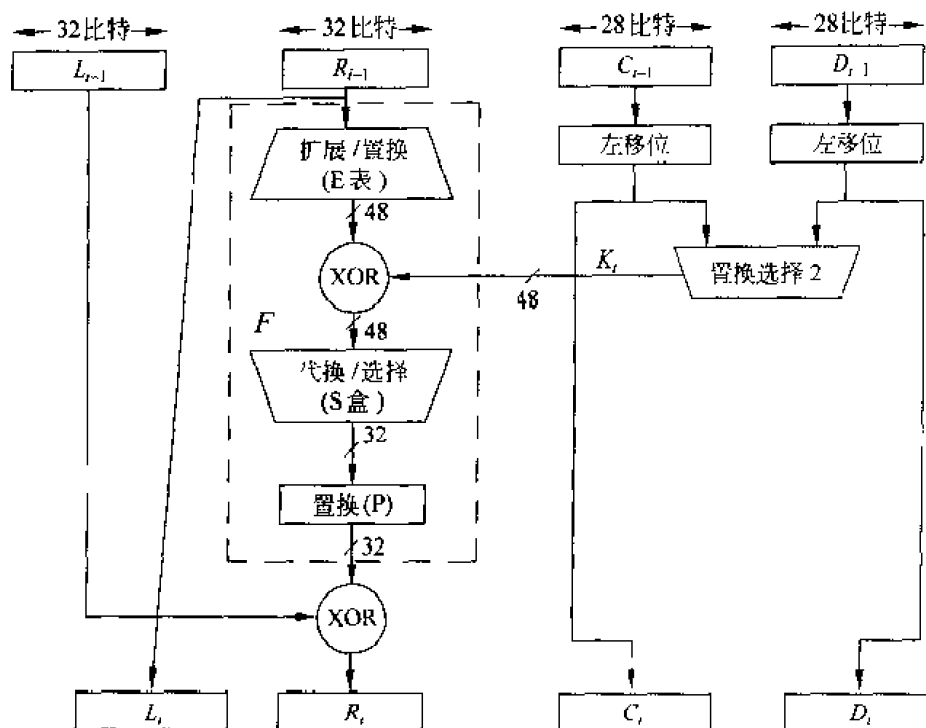


图 3-6 DES 加密算法的轮结构

首先看图左半部分。将 64 比特的轮输入分成各为 32 比特的左、右两半, 分别记为 L 和 R 。和 Feistel 网络一样, 每轮变换可由以下公式表示:

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

其中轮密钥 K_i 为 48 比特, 函数 $F(R, K)$ 的计算过程如图 3-7 所示。轮输入的右半部分 R 为 32 比特, R 首先被扩展成 48 比特, 扩展过程由表 3-2(c) 定义, 其中将 R 的 16 个比特各重复一次。扩展后的 48 比特再与子密钥 K_i 异或, 然后再通过一个 S 盒, 产生 32 比特的输出。该输出再经过一个由表 3-2(d) 定义的置换, 产生的结果即为函数 $F(R, K)$ 的

输出。

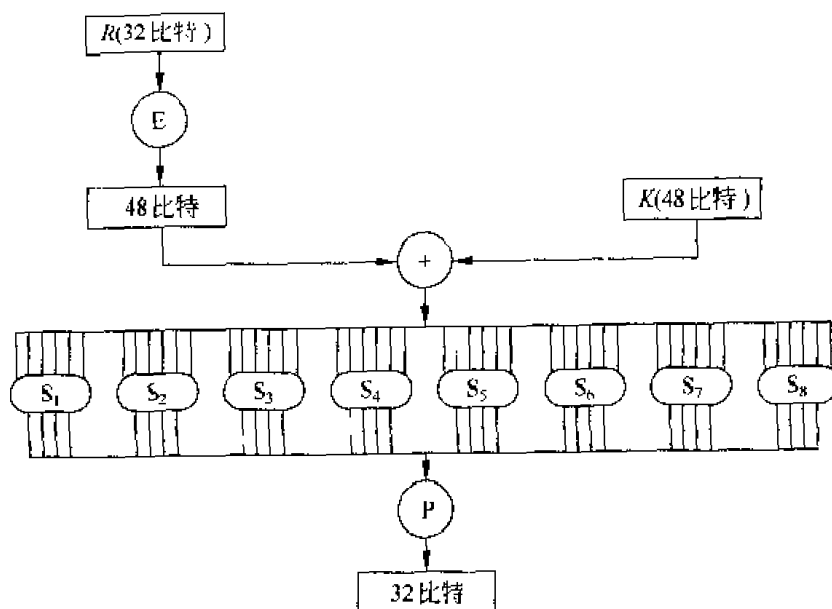


图 3-7 函数 $F(R, K)$ 的计算过程

F 中的代换由 8 个 S 盒组成, 每个 S 盒的输入长为 6 比特、输出长为 4 比特, 其变换关系由表 3-3 定义, 每个 S 盒给出了 4 个代换(由一个表的 4 行给出)。

表 3-3 DES 的 S 盒定义

列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
行	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

续表

列 行		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	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
S_2	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	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
S_3	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	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
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	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
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	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

对每个盒 S_i , 其 6 比特输入中, 第 1 个和第 6 个比特形成一个 2 位二进制数, 用来选择 S_i 的 4 个代换中的一个。6 比特输入中, 中间 4 位用来选择列。行和列选定后, 得到其交叉位置的十进制数, 将这个数表示为 4 位二进制数即得这一 S 盒的输出。例如, S_1 的输入为 011001, 行选为 01 (即第 1 行), 列选为 1100 (即第 12 列), 行列交叉位置的数为 9, 其 4 位二进制表示为 1001, 所以 S_1 的输出为 1001。

S 盒的每一行定义了一个可逆代换, 图 3-2 (在 3.1.1 节) 表示 S_1 第 0 行所定义的代换。

3. 密钥的产生

再看图 3-5 和图 3-6, 输入算法的 56 比特密钥首先经过一个置换运算, 该置换由表 3-4(a) 给出, 然后将置换后的 56 比特分为各为 28 比特的左、右两半, 分别记为 C_0 和 D_0 。

在第 i 轮分别对 C_{i-1} 和 D_{i-1} 进行左循环移位, 所移位数由表 3-4(c) 给出。移位后的结果作为求下一轮子密钥的输入, 同时也作为置换选择 2 的输入。通过置换选择 2 产生的 48 比特的 K_i , 即为本轮的子密钥, 作为函数 $F(R_{i-1}, K_i)$ 的输入。其中置换选择 2 由表 3-4(b) 定义。

表 3-4 DES 密钥编排中使用的表

(a) 置换选择 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

(b) 置换选择 2

PC-2

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

(c) 左循环移位位数

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

4. 解密

和 Feistel 密码一样, DES 的解密和加密使用同一算法, 但子密钥使用的顺序相反。

3.2.2 二重 DES

为了提高 DES 的安全性, 并利用实现 DES 的现有软硬件, 可将 DES 算法在多密钥下多重使用。

二重 DES 是多重使用 DES 时最简单的形式, 如图 3-8 所示。其中明文为 P , 两个加密密钥为 K_1 和 K_2 , 密文为:

$$C = E_{K_2}[E_{K_1}[P]]$$

解密时, 以相反顺序使用两个密钥:

$$P = D_{K_1}[D_{K_2}[C]]$$

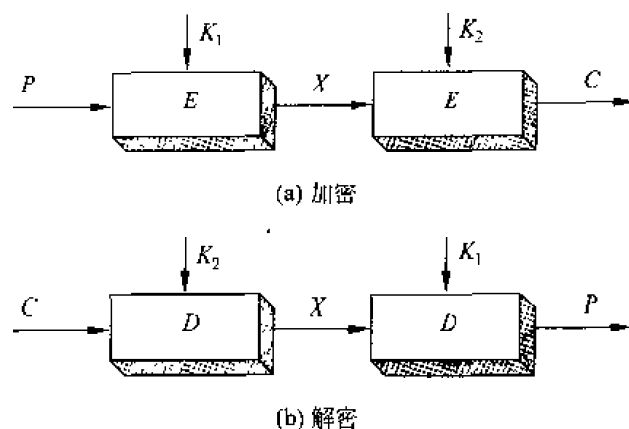


图 3-8 二重 DES

因此,二重 DES 所用密钥长度为 112 比特,强度极大地增加。然而,假设对任意两个密钥 K_1 和 K_2 ,能够找出另一密钥 K_3 ,使得

$$E_{K_2}[E_{K_1}[P]] = E_{K_3}[P]$$

那么,二重 DES 以及多重 DES 都没有意义,因为它们与 56 比特密钥的单重 DES 等价,好在这种假设对 DES 并不成立。将 DES 加密过程 64 比特分组到 64 比特分组的映射看作一个置换,如果考虑 2^{64} 个所有可能的输入分组,则密钥给定后,DES 的加密将把每个输入分组映射到一个惟一的输出分组。否则,如果有两个输入分组被映射到同一分组,则解密过程就无法实施。对 2^{64} 个输入分组,总映射个数为 $(2^{64})! > (10^{10^{20}})$ 。

另一方面,对每个不同的密钥,DES 都定义了一个映射,总映射数为 $2^{56} < 10^{17}$ 。

因此,可假定用两个不同的密钥两次使用 DES,可得一个新映射,而且这一新映射不出现在单重 DES 定义的映射中。这一假定已于 1992 年被证明。所以使用二重 DES 产生的映射不会等价于单重 DES 加密。但对二重 DES 有以下一种称为中途相遇攻击的攻击方案,这种攻击不依赖于 DES 的任何特性,因而可用于攻击任何分组密码。其基本思想如下:

如果有

$$C = E_{K_2}[E_{K_1}[P]]$$

那么(见图 3-8)

$$X = E_{K_1}[P] = D_{K_2}[C]$$

如果已知一个明文密文对 (P, C) ,攻击的实施可如下进行:首先,用 2^{56} 个所有可能的 K_1 对 P 加密,将加密结果存入一表并对表按 X 排序,然后用 2^{56} 个所有可能的 K_2 对 C 解密,在上述表中查找与 C 解密结果相匹配的项,如果找到,则记下相应的 K_1 和 K_2 。最后

再用一新的明文密文对 (P', C') 检验上面找到的 K_1 和 K_2 , 用 K_1 和 K_2 对 P' 两次加密, 若结果等于 C' , 就可确定 K_1 和 K_2 是所要找的密钥。

对已知的明文 P , 二重 DES 能产生 2^{64} 个可能的密文, 而可能的密钥个数为 2^{112} , 所以平均来说, 对一个已知的明文, 有 $2^{112}/2^{64} = 2^{48}$ 个密钥可产生已知的密文。而再经过另外一对明文密文的检验, 误报率将下降到 $2^{48-61} = 2^{-16}$ 。所以在实施中途相遇攻击时, 如果已知两个明文密文对, 则找到正确密钥的概率为 $1 - 2^{-16}$ 。

3.2.3 两个密钥的三重 DES

抵抗中途相遇攻击的一种方法是使用 3 个不同的密钥做 3 次加密, 从而可使已知明文攻击的代价增加到 2^{112} 。然而, 这样又会使密钥长度增加到 $56 \times 3 = 168$ 比特, 因而过于笨重。一种实用的方法是仅使用两个密钥做 3 次加密, 实现方式为加密-解密-加密, 简记为 EDE (encrypt-decrypt-encrypt), 见图 3-9, 即:

$$C = E_{K_1}[D_{K_2}[E_{K_1}[P]]]$$

第 2 步解密的目的仅在于使得用户可对二重 DES 加密的数据解密。

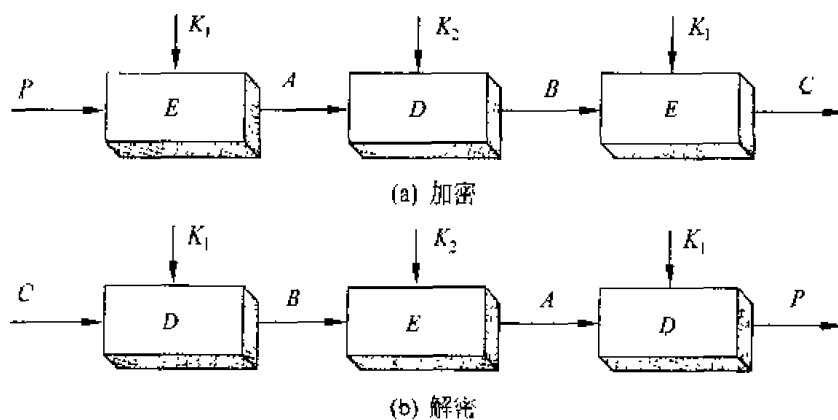


图 3-9 两个密钥的三重 DES

此方案已在密钥管理标准 ANSI X.917 和 ISO 8732 中被采用。

3.2.4 三个密钥的三重 DES

三个密钥的三重 DES 密钥长度为 168 比特, 加密方式为

$$C = E_{K_3}[D_{K_2}[E_{K_1}[P]]]$$

令 $K_3 = K_2$ 或 $K_1 = K_2$, 则变为一重 DES。

三个密钥的三重 DES 已在因特网的许多应用(如 PGP 和 S/MIME)中被采用。

3.3 差分密码分析与线性密码分析

3.3.1 差分密码分析

差分密码分析是迄今已知的攻击迭代密码最有效的方法之一,其基本思想是:通过分析明文对的差值对密文对的差值的影响来恢复某些密钥比特。

对分组长度为 n 的 r 轮迭代密码,两个 n 比特串 Y_i 和 Y_i^* 的差分定义为

$$\Delta Y_i = Y_i \otimes (Y_i^*)^{-1}$$

其中 \otimes 表示 n 比特串集上的一个特定群运算, $(Y_i^*)^{-1}$ 表示 Y_i^* 在此群中的逆元。

由加密对可得差分序列

$$\Delta Y_0, \Delta Y_1, \dots, \Delta Y_r$$

其中 Y_0 和 Y_0^* 是明文对, Y_i 和 Y_i^* ($1 \leq i \leq r$) 是第 i 轮的输出,它们同时也是第 $i+1$ 轮的输入。第 i 轮的子密钥记为 K_i , F 是轮函数,且 $Y_i = F(Y_{i-1}, K_i)$ 。

定义 3-1 r -轮特征(r -round characteristic) Ω 是一个差分序列:

$$\alpha_0, \alpha_1, \dots, \alpha_r$$

其中 α_0 是明文对 Y_0 和 Y_0^* 的差分, α_i ($1 \leq i \leq r$) 是第 i 轮输出 Y_i 和 Y_i^* 的差分。 r -轮特征 $\Omega = \alpha_0, \alpha_1, \dots, \alpha_r$ 的概率是指在明文 Y_0 和子密钥 K_1, \dots, K_r 独立、均匀随机时,明文对 Y_0 和 Y_0^* 的差分为 α_0 的条件下,第 i ($1 \leq i \leq r$) 轮输出 Y_i 和 Y_i^* 的差分为 α_i 的概率。

定义 3-2 在 r -轮特征 $\Omega = \alpha_0, \alpha_1, \dots, \alpha_r$ 中,定义

$$p_i^{\alpha} = P(\Delta F(Y) = \alpha_i \mid \Delta Y = \alpha_{i-1})$$

即 p_i^{α} 表示在输入差分为 α_{i-1} 的条件下,轮函数 F 的输出差分为 α_i 的概率。

r -轮特征 $\Omega = \alpha_0, \alpha_1, \dots, \alpha_r$ 的概率近似看作 $\prod_{i=1}^r p_i^{\alpha}$ 。

对 r -轮迭代密码的差分密码分析过程可综述为如下的步骤:

① 找出一个 $(r-1)$ -轮特征 $\Omega(r-1) = \alpha_0, \alpha_1, \dots, \alpha_{r-1}$,使得它的概率达到最大或几乎最大。

② 均匀随机地选择明文 Y_0 并计算 Y_0^* ,使得 Y_0 和 Y_0^* 的差分为 α_0 ,找出 Y_0 和 Y_0^* 在实际密钥加密下所得的密文 Y_r 和 Y_r^* 。若最后一轮的子密钥 K_r (或 K_r 的部分比特)有 2^m 个可能值 K_r^j ($1 \leq j \leq 2^m$),设置相应的 2^m 个计数器 Λ_j ($1 \leq j \leq 2^m$);用每个 K_r^j 解密密文 Y_r 和 Y_r^* ,得到 Y_{r-1} 和 Y_{r-1}^* ,如果 Y_{r-1} 和 Y_{r-1}^* 的差分是 α_{r-1} ,则给相应的计数器 Λ_j 加 1。

③ 重复步骤②,直到一个或几个计数器的值明显高于其他计数器的值,输出它们所对应的子密钥(或部分比特)。

一种攻击的复杂度可以分为两部分：数据复杂度和处理复杂度。数据复杂度是实施该攻击所需输入的数据量；而处理复杂度是处理这些数据所需的计算量。这两部分的主要部分通常被用来刻画该攻击的复杂度。

差分密码分析的数据复杂度是成对加密所需的选择明文对 (Y_0, Y_0^*) 个数的两倍。差分密码分析的处理复杂度是从 $(\Delta Y_{r-1}, Y_r, Y_r^*)$ 找出子密钥 K_r (或 K_r 的部分比特) 的计算量, 它实际上与 r 无关, 而且由于轮函数是弱的, 所以此计算量在大多数情况下相对较小。因此, 差分密码分析的复杂度取决于它的数据复杂度。

3.3.2 线性密码分析

线性密码分析是对迭代密码的一种已知明文攻击, 它利用的是密码算法中的“不平衡(有效)的线性逼近”。

设明文分组长度和密文分组长度都为 n 比特, 密钥分组长度为 m 比特。记明文分组为 $P[1], P[2], \dots, P[n]$, 密文分组为 $C[1], C[2], \dots, C[n]$, 密钥分组为 $K[1], K[2], \dots, K[m]$ 。定义 $A[i, j, \dots, k] = A[i] \oplus A[j] \oplus \dots \oplus A[k]$ 。

线性密码分析的目标就是找出以下形式的有效线性方程:

$$P[i_1, i_2, \dots, i_a] \oplus C[j_1, j_2, \dots, j_b] = K[k_1, k_2, \dots, k_c]$$

其中 $1 \leq a \leq n, 1 \leq b \leq n, 1 \leq c \leq m$ 。

如果方程成立的概率 $p \neq \frac{1}{2}$, 则称该方程是有效的线性逼近。如果 $\left| p - \frac{1}{2} \right|$ 是最大的, 则称该方程是最有效的线性逼近。

设 N 表示明文数, T 是使方程左边为 0 的明文数。如果 $T > N/2$, 则令

$$K[k_1, k_2, \dots, k_c] = \begin{cases} 0, & p > \frac{1}{2} \\ 1, & p < \frac{1}{2} \end{cases}$$

如果 $T < N/2$, 则令

$$K[k_1, k_2, \dots, k_c] = \begin{cases} 0, & p < \frac{1}{2} \\ 1, & p > \frac{1}{2} \end{cases}$$

从而可得关于密钥比特的一个线性方程。对不同的明文密文对重复以上过程, 可得关于密钥的一组线性方程, 从而确定出密钥比特。

研究表明, 当 $\left| p - \frac{1}{2} \right|$ 充分小时, 攻击成功的概率是

$$\frac{1}{\sqrt{2\pi}} \int_{-2\sqrt{N} \left| p - \frac{1}{2} \right|}^{\infty} e^{-\frac{x^2}{2}} dx$$

这一概率只依赖于 $\sqrt{N} \left| p - \frac{1}{2} \right|$, 并随着 N 或 $\left| p - \frac{1}{2} \right|$ 的增加而增加。

如何对差分密码分析和线性密码分析进行改进,降低它们的复杂度仍是现在理论研究的热点,目前已推出了很多改进方法,例如,高阶差分密码分析、截段差分密码分析(truncated differential cryptanalysis)、不可能差分密码分析、多重线性密码分析、非线性密码分析、划分密码分析和差分-线性密码分析,再如针对密钥编排算法的相关密钥攻击、基于 Lagrange 插值公式的插值攻击及基于密码器件的能量分析(power analysis),另外还有错误攻击、时间攻击、Square 攻击和 Davies 攻击等。

3.4 分组密码的运行模式

分组密码在加密时,明文分组的长度是固定的,而实际应用中待加密消息的数据量是不定的,数据格式可能是多种多样的。为了能在各种应用场合使用 DES,美国在 FIPS PUS 74 和 81 中定义了 DES 的 4 种运行模式,如表 3-5 所示。这些模式也可用于其他分组密码,下面以 DES 为例来介绍这 4 种模式。

表 3-5 DES 的运行模式

模 式	描 述	用 途
电码本(ECB)模式	每个明文组独立地以同一密钥加密	传送短数据(如一个加密密钥)
密码分组链接(CBC)模式	加密算法的输入是当前明文组与前一密文组的异或	传送数据分组;认证
密码反馈(CFB)模式	每次只处理输入的 r 比特,将上一次的密文用作加密算法的输入以产生伪随机输出,该输出再与当前明文异或以产生当前密文	传送数据流;认证
输出反馈(OFB)模式	与 CFB 类似,不同之处是本次加密算法的输入为前一次加密算法的输出	有扰信道上(如卫星通信)传送数据流

3.4.1 电码本(ECB)模式

ECB(electronic codebook)模式是最简单的运行模式,它一次对一个 64 比特长的明文分组加密,而且每次的加密密钥都相同,如图 3-10 所示。当密钥取定时,对明文的每一个分组,都有一个惟一的密文与之对应。因此形象地说,可以认为有一个非常大的电码本,对任意一个可能的明文分组,电码本中都有一项对应于它的密文。

如果消息长于 64 比特,则将其分为长为 64 比特的分组,最后一个分组如果不够 64 比特,则需要填充。解密过程也是一次对一个分组解密,而且每次解密都使用同一密钥。图 3-10 中,明文是由分组长为 64 比特的分组序列 P_1, P_2, \dots, P_N 构成,相应的密文分组序列是 C_1, C_2, \dots, C_N 。

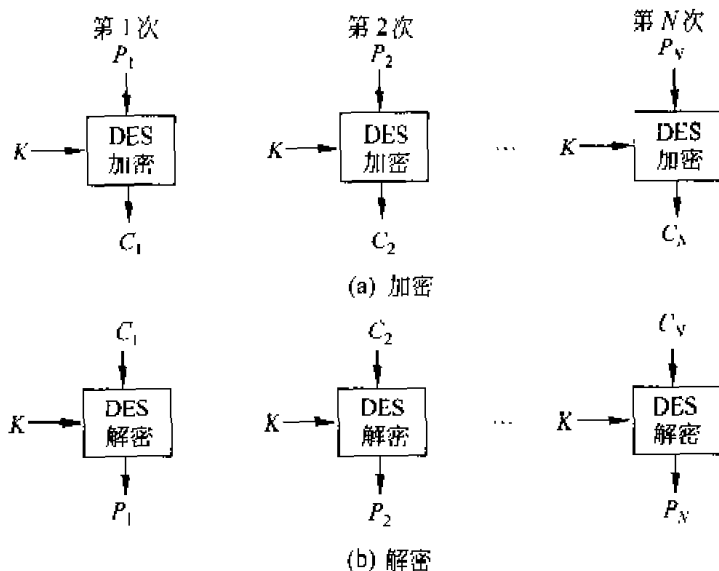


图 3-10 ECB 模式示意图

ECB 在用于短数据(如加密密钥)时非常理想,因此如果需要安全地传递 DES 密钥,ECB 是最合适的模式。

ECB 的最大特性是同一明文分组在消息中重复出现的话,产生的密文分组也相同。

ECB 用于长消息时可能不够安全,如果消息有固定结构,密码分析者有可能找出这种关系。例如,如果已知消息总是以某个预定义字段开始,那么分析者就可能得到很多明文-密文对。如果消息有重复的元素而重复的周期是 64 的倍数,那么密码分析者就能够识别这些元素。以上这些特性都有助于密码分析者,有可能为其提供对分组的代换或重排的机会。

3.4.2 密码分组链接(CBC)模式

为了解决 ECB 的安全缺陷,可以让重复的明文分组产生不同的密文分组,CBC (cipher block chaining)模式就可满足这一要求。

图 3-11 是 CBC 模式示意图,它一次对一个明文分组加密,每次加密使用同一密钥,加密算法的输入是当前明文分组和前一次密文分组的异或,因此加密算法的输入不会显示出与这次的明文分组之间的固定关系,所以重复的明文分组不会在密文中暴露出这种重复关系。

解密时,每一个密文分组被解密后,再与前一个密文分组异或,即

$$\begin{aligned} D_K[C_n] \oplus C_{n-1} &= D_K[E_K[C_{n-1} \oplus P_n]] \oplus C_{n-1} \\ &= C_{n-1} \oplus P_n \oplus C_{n-1} = P_n \quad (\text{设 } C_n = E_K[C_{n-1} \oplus P_n]) \end{aligned}$$

因而产生出明文分组。

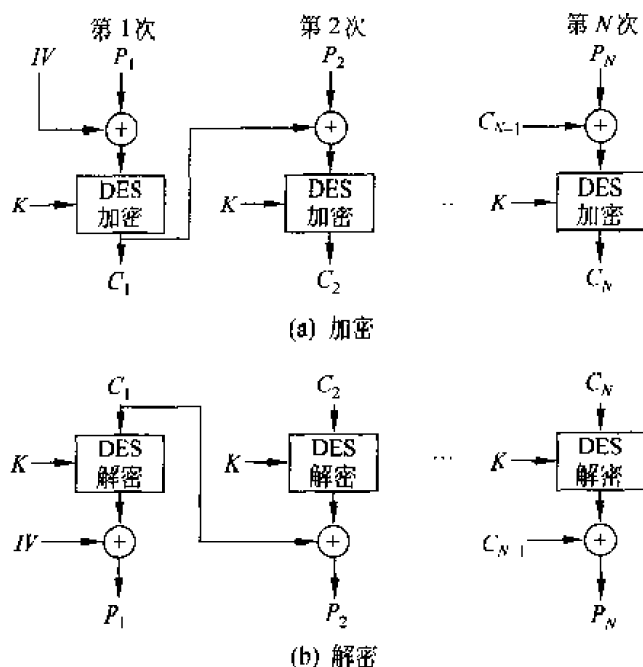


图 3-11 CBC 模式示意图

在产生第 1 个密文分组时,需要有一个初始向量 IV 与第 1 个明文分组异或。解密时, IV 和解密算法对第 1 个密文分组的输出进行异或以恢复第 1 个明文分组。

IV 对于收发双方都应是已知的,为使安全性最高, IV 应像密钥一样被保护,可使用 ECB 加密模式来发送 IV 。保护 IV 的原因如下:如果敌手能欺骗接收方使用不同的 IV 值,敌手就能够在明文的第 1 个分组中插入自己选择的比特值,这是因为:

$$C_1 = E_K[IV \oplus P_1]$$

$$P_1 = IV \oplus D_K[C_1]$$

用 $X(i)$ 表示 64 比特分组 X 的第 i 个比特,那么 $P_1(i) = IV(i) \oplus D_K[C_1](i)$,由异或的性质得

$$P_1(i)' = IV(i)' \oplus D_K[C_1](i)$$

其中撇号表示比特补。上式意味着如果敌手篡改 IV 中的某些比特,则接收方收到的 P_1 中相应的比特也发生了变化。

由于 CBC 模式的链接机制,CBC 模式对加密长于 64 比特的消息非常合适。

CBC 模式除能够获得保密性外,还能用于认证。

3.4.3 密码反馈(CFB)模式

如上所述,DES 是分组长为 64 比特的分组密码,但利用 CFB(cipher feedback)模式或 OFB 模式可将 DES 转换为流密码。流密码不需要对消息填充,而且运行是实时的。因此如果传送字母流,可使用流密码对每个字母直接加密并传送。

流密码具有密文和明文一样长这一性质,因此,如果需要发送的每个字符长为 8 比特,就应使用 8 比特密钥来加密每个字符。如果密钥长超过 8 比特,则造成浪费。

图 3 12 是 CFB 模式示意图,设传送的每个单元(如一个字符)是 j 比特长,通常取 $j=8$,与 CBC 模式一样,明文单元被链接在一起,使得密文是前面所有明文的函数。

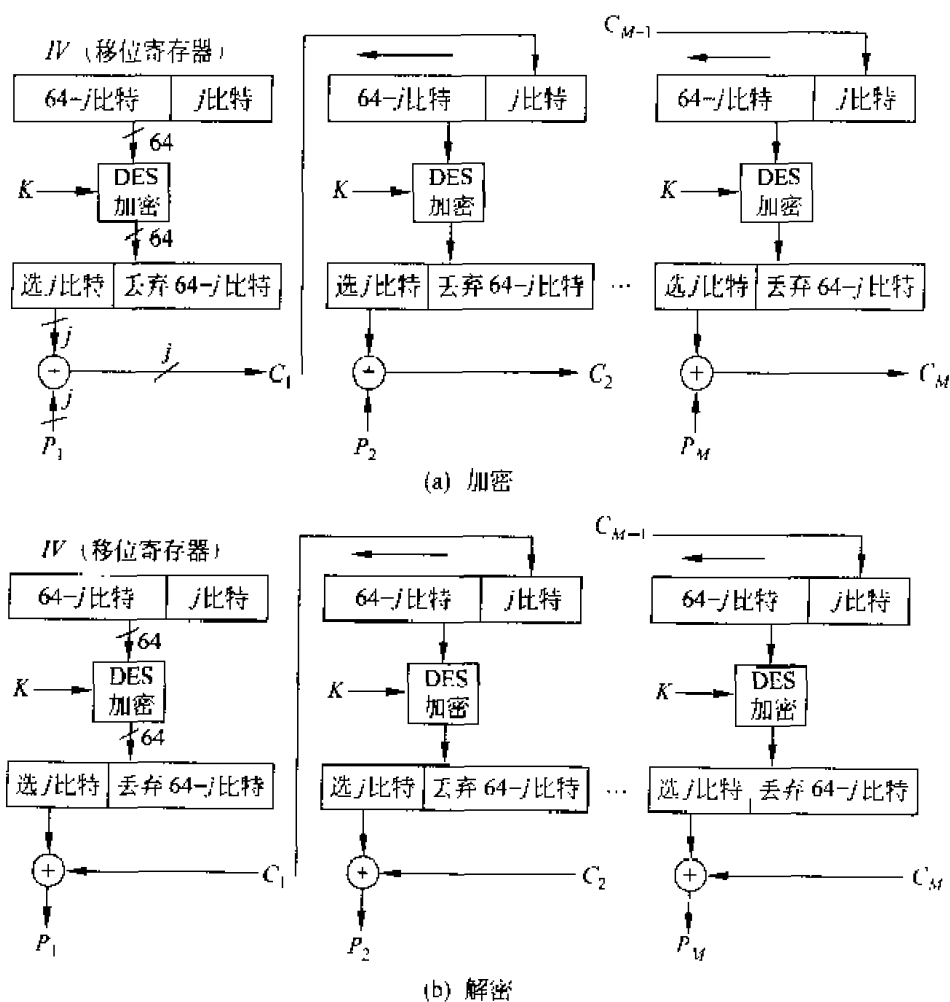


图 3-12 CFB 模式示意图

加密时,加密算法的输入是 64 比特移位寄存器,其初值为某个初始向量 IV 。加密算

法输出的最左(最高有效位) j 比特与明文的第一个单元 P_1 进行异或,产生出密文的第 1 个单元 C_1 ,并传送该单元。然后将移位寄存器的内容左移 j 位并将 C_1 送入移位寄存器最右边(最低有效位) j 位。这一过程继续到明文的所有单元都被加密为止。

解密时,将收到的密文单元与加密函数的输出进行异或。注意这时仍然使用加密算法而不是解密算法,原因如下:

设 $S_j(X)$ 是 X 的 j 个最高有效位,那么 $C_1 = P_1 \oplus S_j(E(IV))$,因此

$$P_1 = C_1 \oplus S_j(E(IV))$$

可证明以后各步也有类似的这种关系。

CFB 模式除能获得保密性外,还能用于认证。

3.4.4 输出反馈(OFB)模式

OFB(output feedback)模式的结构类似于 CFB,见图 3-13。不同之处如下: OFB 模式是

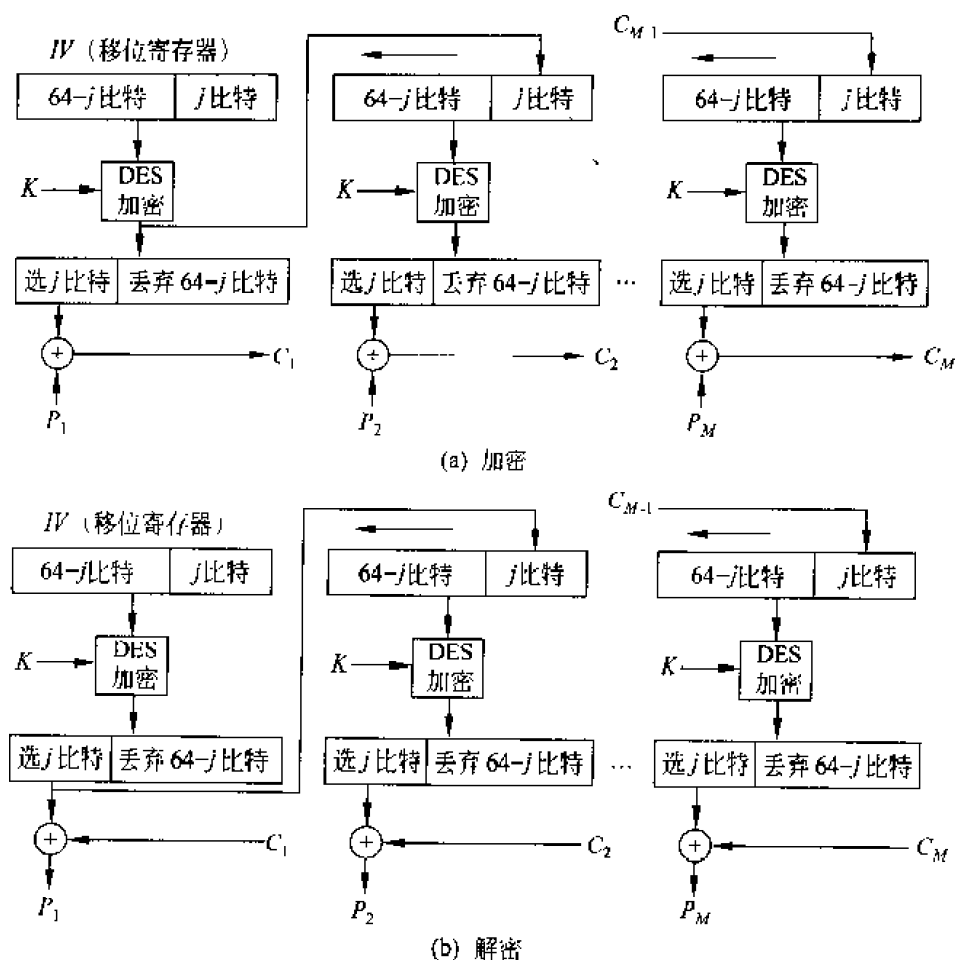


图 3-13 OFB 模式示意图

将加密算法的输出反馈到移位寄存器,而CFB模式中是将密文单元反馈到移位寄存器。

OFB模式的优点是传输过程中的比特错误不会被传播。例如 C_i 中出现1比特错误,在解密结果中只有 P_i 受到影响,以后各明文单元则不受影响。而在CFB中, C_i 也作为移位寄存器的输入,因此它的1比特错误会影响解密结果中各明文单元的值。

OFB的缺点是它比CFB模式更易受到对消息流的篡改攻击,比如在密文中取1比特的补,那么在恢复的明文中相应位置的比特也为原比特的补。因此使得敌手有可能通过对消息校验部分的篡改和对数据部分的篡改,而以纠错码不能检测的方式篡改密文。

3.5 IDEA

瑞士联邦技术学院来学嘉(X. J. Lai)和J. L. Massey提出的第1版IDEA(international data encryption algorithm,国际数据加密算法)于1990年公布,当时称为PES(proposed encryption standard,建议加密标准)。1991年,在Biham和Shamir提出差分密码分析之后,设计者推出了改进算法IPES,即改进型建议加密标准。1992年,设计者又将IPES改名为IDEA。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。

3.5.1 设计原理

算法中明文和密文分组长度都是64比特,密钥长128比特。其设计原理可从强度和实现两方面考虑。

1. 密码强度

算法的强度主要是通过有效的混淆和扩散特性而得以保证。

混淆是通过使用以下3种运算而获得,3种运算都有两个16比特的输入和一个16比特的输出:

① 逐比特异或,表示为 \oplus 。

② 模 2^{16} (即65536)整数加法,表示为 \boxplus ,其输入和输出作为16位无符号整数处理。

③ 模 $2^{16}+1$ (即65537)整数乘法,表示为 \odot ,其输入、输出中除16位全为0作为 2^{16} 处理外,其余都作为16位无符号整数处理。例如

$$0000000000000000 \odot 1000000000000000 = 1000000000000001$$

这是因为 $2^{16} \times 2^{15} \bmod (2^{16}+1) = 2^{15}+1$ 。

表3-6给出了操作数为2比特长时3种运算的运算表。在以下意义下,3种运算是

不兼容的:

① 3 种运算中任意两种都不满足分配律,例如

$$a \boxplus (b \odot c) \neq (a \boxplus b) \odot (a \boxplus c)$$

② 3 种运算中任意两种都不满足结合律,例如

$$a \boxplus (b \oplus c) \neq (a \boxplus b) \oplus c$$

3 种运算结合起来使用可对算法的输入提供复杂的变换,从而使得对 IDEA 的密码分析比对仅使用异或运算的 DES 更为困难。

表 3-6 IDEA 中的 3 种运算(操作数为 2 比特长)

X		Y		$X \boxplus Y$		$X \odot Y$		$X \ominus Y$	
0	00	0	00	0	00	1	01	0	00
0	00	1	01	1	01	0	00	1	01
0	00	2	10	2	10	3	11	2	10
0	00	3	11	3	11	2	10	3	11
1	01	0	00	1	01	0	00	1	01
1	01	1	01	2	10	1	01	0	00
1	01	2	10	3	11	2	10	3	11
1	01	3	11	0	00	3	11	2	10
2	10	0	00	2	10	3	11	2	10
2	10	1	01	3	11	2	10	3	11
2	10	2	10	0	00	0	00	0	00
2	10	3	11	1	01	1	01	1	01
3	11	0	00	3	11	2	10	3	11
3	11	1	01	0	00	3	11	2	10
3	11	2	10	1	01	1	01	1	01
3	11	3	11	2	10	0	00	0	00

算法中扩散是由称为乘加(multiplication/addition, MA)结构(见图 3-14)的基本单元实现的。该结构的输入是两个 16 比特的子段和两个 16 比特的子密钥,输出也为两个 16 比特的子段。这一结构在算法中重复使用了 8 次,获得了非常有效的扩散效果。

2. 实现

IDEA 可方便地通过软件和硬件实现。

① 软件 软件实现采用 16 比特子段处理,可通过使用容易编程的加法、移位等运算实现算法的 3 种运算。

② 硬件 由于加、解密相似,差别仅为使用密钥的方式,因此可用同一器件实现。再者,算法中规则的模块结构,可方

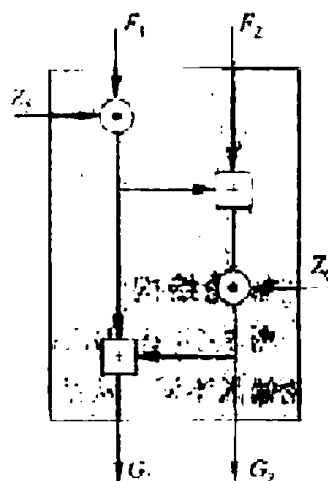


图 3-14 MA 结构

便 VLSI 的实现。

3.5.2 加密过程

加密过程(如图 3-15 所示)由连续的 8 轮迭代和一个输出变换组成,算法将 64 比特的明文分组分成 4 个 16 比特的子段,每轮迭代以 4 个 16 比特的子段作为输入,输出也为 4 个 16 比特的子段。最后的输出变换也产生 4 个 16 比特的子段,链接起来后形成 64 比特的密文分组。每轮迭代还需使用 6 个 16 比特的子密钥,最后的输出变换需使用 4 个 16 比特的子密钥,所以子密钥总数为 52。图 3-15 的右半部分表示由初始的 128 比特密钥产生 52 个子密钥的子密钥产生器。

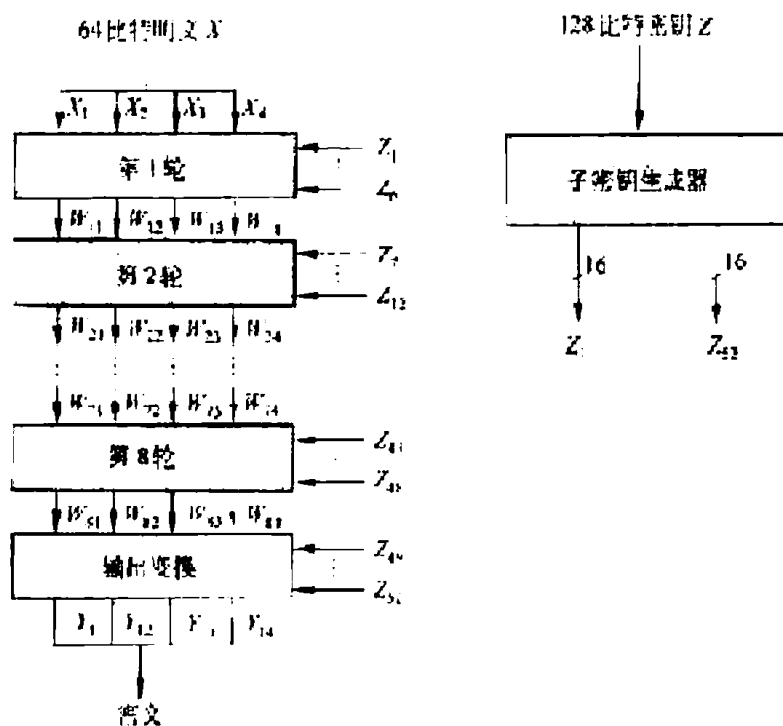


图 3-15 IDEA 的加密框图

1. 轮结构

图 3-16 是 IDEA 第 1 轮的结构示意图,以后各轮也都是这种结构,但所用的子密钥和轮输入不同。从结构图可见,IDEA 不是传统的 Feistel 密码结构。每轮开始时有一个变换,该变换的输入是 4 个子段和 4 个子密钥,变换中的运算是两个乘法和两个加法,输出的 4 个子段经过异或运算形成了两个 16 比特的子段作为 MA 结构的输入。MA 结构也有两个输入的子密钥,输出是两个 16 比特的子段。

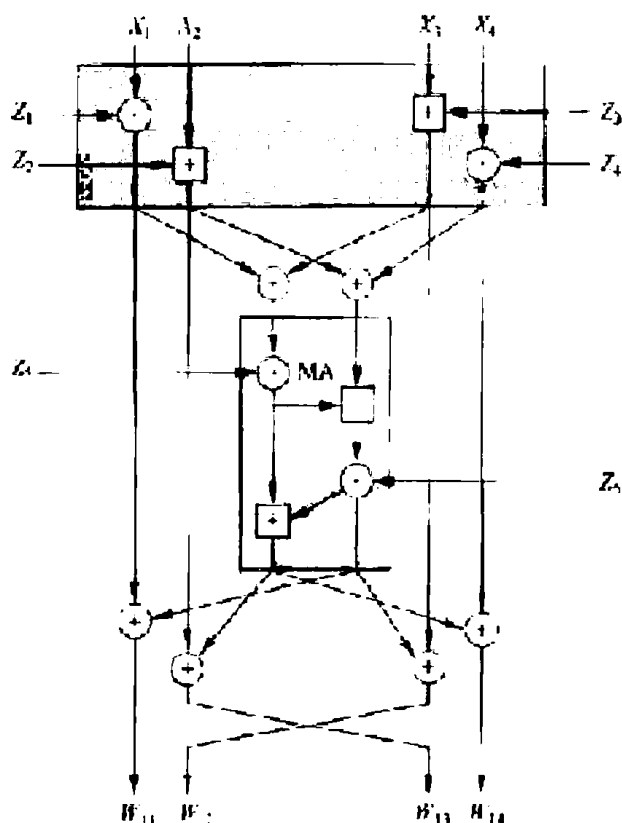


图 3-16 IDEA 第 1 轮的轮结构

最后,变换的 4 个输出子段和 MA 结构的两个输出子段经过异或运算产生这一轮的 4 个输出子段。注意,由 X_2 产生的输出子段和由 X_4 产生的输出子段交换位置后形成 W_{12} 和 W_{14} , 目的在于进一步增加混淆效果,使得算法更易抵抗差分密码分析。

算法的第 9 步是一个输出变换,如图 3-17 所示。它的结构和每一轮开始的变换结构一样,不同之处在于输出变换的第 2 个和第 3 个输入首先交换了位置,目的在于撤销第 8 轮输出中两个子段的交换。还需注意,第 9 步仅需 4 个子密钥,而前面 8 轮中每轮需要 6 个子密钥。

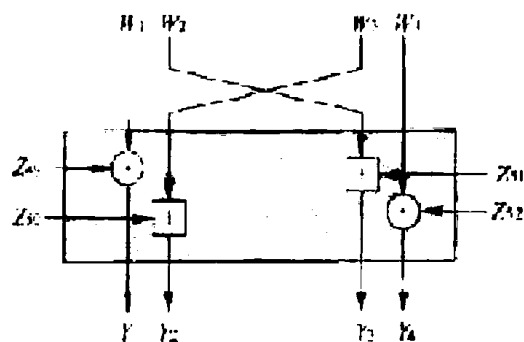


图 3-17 IDEA 的输出变换

2. 子密钥的产生

加密过程中 52 个 16 比特的子密钥是由 128 比特的加密密钥按如下方式产生的: 前 8 个子密钥 Z_1, Z_2, \dots, Z_8 直接从加密密钥中取,即 Z_1 取前 16 比特(最高有效位), Z_2 取下部的 16 比特,依次类推。然后加密密钥循环左移 25 位,再取下面 8 个子密钥 $Z_9, Z_{10}, \dots, Z_{15}$,取法与 Z_1, Z_2, \dots, Z_8 的取法相同。这

一过程重复下去,直到52子密钥都被产生为止。

3. 解密过程

解密过程和加密过程基本相同,但子密钥的选取不同。解密子密钥 U_1, U_2, \dots, U_{52} 是由加密子密钥按如下方式得到(将加密过程最后一步的输出变换当作第9轮):

① 第 $i(i=1, \dots, 9)$ 轮解密的前4个子密钥由加密过程第 $(10-i)$ 轮的前4个子密钥得出: 其中第1和第4个解密子密钥取为相应的第1和第4个加密子密钥的模 $2^{16}+1$ 乘法逆元,第2和第3个子密钥的取法为: 当轮数 $i=2, \dots, 8$ 时,取为相应的第3个和第2个加密子密钥的模 2^{16} 加法逆元。 $i=1$ 和 9 时,取为相应的第2个和第3个加密子密钥的模 2^{16} 加法逆元。

② 第 $i(i=1, \dots, 8)$ 轮解密的后两个子密钥等于加密过程第 $(9-i)$ 轮的后两个子密钥。

表 3-7 是对以上关系的总结。其中 Z_j 的模 $2^{16}+1$ 乘法逆元为 Z_j^{-1} , 满足:

$$Z_j \odot Z_j^{-1} = 1 \bmod (2^{16} + 1)$$

表 3-7 IDEA 加密、解密子密钥表

轮数	加密子密钥	解密子密钥	
1	$Z_1, Z_2, Z_3, Z_4, Z_5, Z_6$	$U_1, U_2, U_3, U_4, U_5, U_6$	$Z_{49}^{-1}, -Z_{50}, -Z_{51}, Z_{52}^{-1}, Z_{47}, Z_{48}$
2	$Z_7, Z_8, Z_9, Z_{10}, Z_{11}, Z_{12}$	$U_7, U_8, U_9, U_{10}, U_{11}, U_{12}$	$Z_{15}^{-1}, -Z_{45}, -Z_{41}, Z_{46}^{-1}, Z_{41}, Z_{42}$
3	$Z_{13}, Z_{14}, Z_{15}, Z_{16}, Z_{17}, Z_{18}$	$U_{13}, U_{14}, U_{15}, U_{16}, U_{17}, U_{18}$	$Z_{37}^{-1}, -Z_{43}, -Z_{18}, Z_{40}^{-1}, Z_{15}, Z_{36}$
4	$Z_{19}, Z_{20}, Z_{21}, Z_{22}, Z_{23}, Z_{24}$	$U_{19}, U_{20}, U_{21}, U_{22}, U_{23}, U_{24}$	$Z_{31}^{-1}, -Z_{45}, -Z_{32}, Z_{34}^{-1}, Z_{29}, Z_{30}$
5	$Z_{25}, Z_{26}, Z_{27}, Z_{28}, Z_{29}, Z_{30}$	$U_{25}, U_{26}, U_{27}, U_{28}, U_{29}, U_{30}$	$Z_{25}^{-1}, -Z_{27}, -Z_{25}, Z_{28}^{-1}, Z_{23}, Z_{24}$
6	$Z_{31}, Z_{32}, Z_{33}, Z_{34}, Z_{35}, Z_{36}$	$U_{31}, U_{32}, U_{33}, U_{34}, U_{35}, U_{36}$	$Z_{19}^{-1}, -Z_{21}, -Z_{21}, Z_{22}^{-1}, Z_{17}, Z_{18}$
7	$Z_{37}, Z_{38}, Z_{39}, Z_{40}, Z_{41}, Z_{42}$	$U_{37}, U_{38}, U_{39}, U_{40}, U_{41}, U_{42}$	$Z_{13}^{-1}, -Z_{15}, -Z_{14}, Z_{16}^{-1}, Z_{11}, Z_{12}$
8	$Z_{43}, Z_{44}, Z_{45}, Z_{46}, Z_{47}, Z_{48}$	$U_{43}, U_{44}, U_{45}, U_{46}, U_{47}, U_{48}$	$Z_7^{-1}, -Z_9, -Z_8, Z_{10}^{-1}, Z_5, Z_6$
输出变换	$Z_{49}, Z_{50}, Z_{51}, Z_{52}$	$U_{49}, U_{50}, U_{51}, U_{52}$	$Z_1^{-1}, -Z_2, -Z_1, Z_4^{-1}$

因 $2^{16}+1$ 是一素数,所以每一个不大于 2^{16} 的非 0 整数都有一个惟一的模 $2^{16}+1$ 乘法逆元。 Z_j 的模 2^{16} 加法逆元为 $-Z_j$, 满足:

$$-Z_j \boxplus Z_j = 0 \bmod (2^{16})$$

下面验证解密过程的确可以得到正确的结果。图 3-18 中左边为加密过程,由上至下,右边为解密过程,由下至上。将每一轮进一步分为两步,第1步是变换,其余部分作为第2步,称为子加密。

现在从下往上考虑。对加密过程的最后一个输出变换,以下关系成立:

$$\begin{aligned} Y_1 &= W_{81} \odot Z_{49} & Y_2 &= W_{83} \boxplus Z_{50} \\ Y_3 &= W_{82} \boxplus Z_{51} & Y_4 &= W_{84} \odot Z_{52} \end{aligned}$$

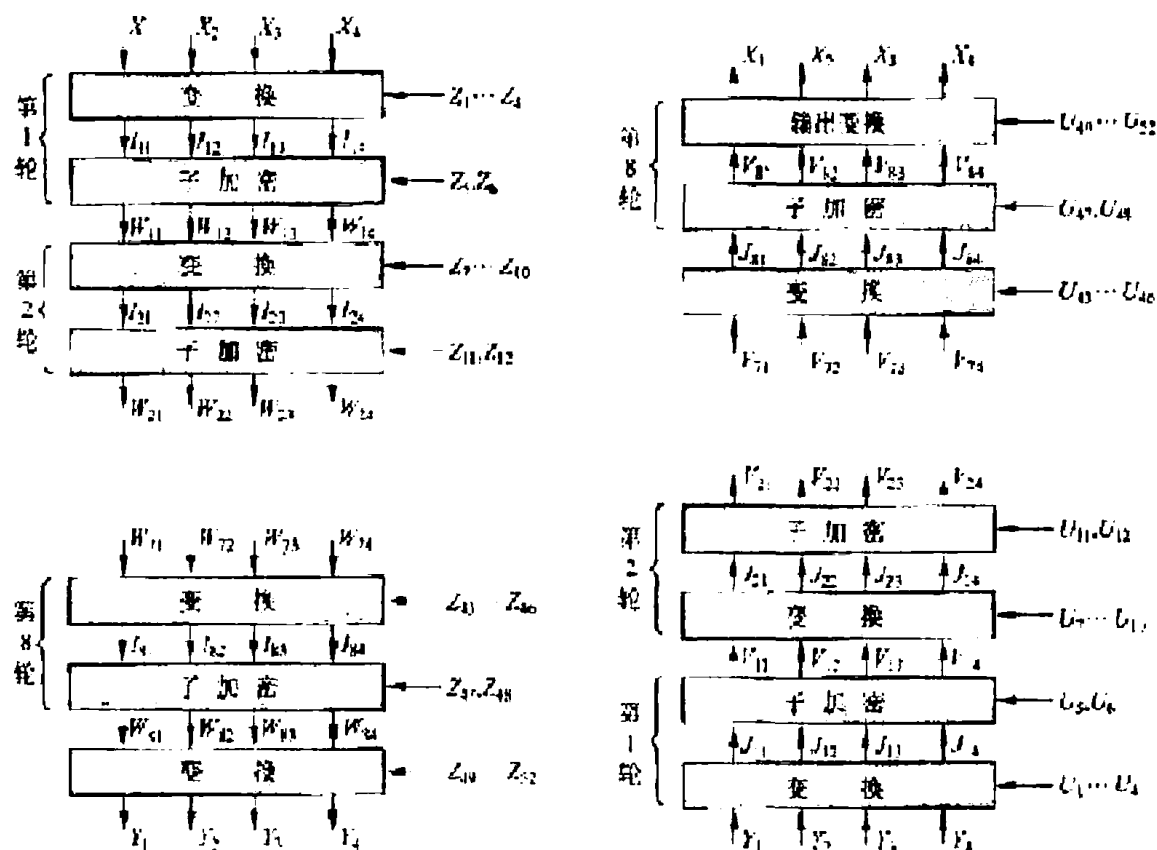


图 3-18 IDEA 加密和解密框图

解密过程中第 1 轮的第 1 步产生以下关系：

$$\begin{aligned} J_{11} &= Y_1 \odot U_1 & J_{12} &= Y_2 \oplus U_2 \\ J_{13} &= Y_3 \oplus U_3 & J_{14} &= Y_4 \odot U_4 \end{aligned}$$

将解密子密钥由加密子密钥表达并将 Y_1, Y_2, Y_3, Y_4 代入以下关系，有

$$\begin{aligned} J_{11} &= Y_1 \odot Z_{49} = W_{81} \odot Z_{49} \odot Z_{49}^{-1} = W_{81} \\ J_{12} &= Y_2 \oplus Z_{50} = W_{83} \oplus Z_{50} \oplus Z_{50} = W_{83} \\ J_{13} &= Y_3 \oplus Z_{51} = W_{82} \oplus Z_{51} \oplus Z_{51} = W_{82} \\ J_{14} &= Y_4 \odot Z_{52} = W_{84} \odot Z_{52} \odot Z_{52}^{-1} = W_{84} \end{aligned}$$

可见解密过程第 1 轮第 1 步的输出等于加密过程最后一步输入中第 2 个子段和第 3 个子段交换后的值。从图 3-16, 可得以下关系：

$$\begin{aligned} W_{81} &= I_{81} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \\ W_{82} &= I_{83} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \\ W_{83} &= I_{82} \oplus MA_L(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \end{aligned}$$

$$W_{84} = I_{84} \oplus MA_L(I_{81} \oplus I_{83}, I_{82} \oplus I_{84})$$

其中 $MA_R(X, Y)$ 是 MA 结构输入为 X 和 Y 时的右边输出, $MA_L(X, Y)$ 是左边输出。则

$$\begin{aligned} V_{11} &= J_{11} \oplus MA_R(J_{11} \oplus J_{13}, J_{12} \oplus J_{14}) \\ &= W_{81} \oplus MA_R(W_{81} \oplus W_{82}, W_{83} \oplus W_{84}) \\ &= I_{81} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus \\ &\quad MA_R[I_{81} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus I_{83}, I_{82} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84})], \\ &\quad I_{82} \oplus MA_L(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus I_{84} \oplus MA_L(I_{81} \oplus I_{83}, I_{82} \oplus I_{84})] \\ &= I_{81} \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) \oplus MA_R(I_{81} \oplus I_{83}, I_{82} \oplus I_{84}) = I_{81} \end{aligned}$$

类似地, 可有

$$V_{12} = I_{83} \quad V_{13} = I_{82} \quad V_{14} = I_{84}$$

所以解密过程第 1 轮第 2 步的输出等于加密过程倒数第 2 步输入中第 2 个子段和第 3 个子段交换后的值。

同理可证图 3-18 中每步都有上述类似关系, 这种关系一直到

$$V_{81} = I_{11} \quad V_{82} = I_{14} \quad V_{83} = I_{12} \quad V_{84} = I_{13}$$

即除第 2 个子段和第 3 个子段交换位置外, 解密过程的输出变换与加密过程第 1 轮第 1 步的变换完全相同。

所以最后可得知, 整个解密过程的输出等于整个加密过程的输入。

3.6 AES 算法——Rijndael

1997 年 4 月 15 日, 美国 ANSI 发起征集 AES(advanced encryption standard)的活动, 并为此成立了 AES 工作小组。此次活动的目的是确定一个非保密的、可以公开技术细节的、全球免费使用的分组密码算法, 以作为新的数据加密标准。1997 年 9 月 12 日, 美国联邦登记处公布了正式征集 AES 候选算法的通告。对 AES 的基本要求是: 比三重 DES 快、至少与三重 DES 一样安全、数据分组长度为 128 比特、密钥长度为 128/192/256 比特。1998 年 8 月 12 日, 在首届 AES 候选会议(first AES candidate conference)上公布了 AES 的 15 个候选算法, 任由全世界各机构和个人攻击和评论, 这 15 个候选算法是 CAST256、CRYPTON、E2、DEAL、FROG、SAFER +、RC6、MAGENTA、LOKI97、SERPENT、MARS、Rijndael、DFC、Twofish、HPC。1999 年 3 月, 在第 2 届 AES 候选会议(second AES candidate conference)上经过对全球各密码机构和个人对候选算法分析结果的讨论, 从 15 个候选算法中选出了 5 个。这 5 个是 RC6、Rijndael、SERPENT、Twofish 和 MARS。2000 年 4 月 13 日至 14 日, 召开了第 3 届 AES 候选会议(third AES

candidate conference), 继续对最后 5 个候选算法进行讨论。2000 年 10 月 2 日, NIST 宣布 Rijndael 作为新的 AES。至此, 经过 3 年多的讨论, Rijndael 终于脱颖而出。

Rijndael 由比利时的 Joan Daemen 和 Vincent Rijmen 设计, 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。

3.6.1 Rijndael 的数学基础和设计思想

1. 有限域 $GF(2^8)$

有限域中的元素可以用多种不同的方式表示。对于任意素数的方幂, 都有惟一的一个有限域, 因此 $GF(2^8)$ 的所有表示是同构的, 但不同的表示方法会影响到 $GF(2^8)$ 上运算的复杂度, 本算法采用传统的多项式表示法。

将 $b_7b_6b_5b_4b_3b_2b_1b_0$ 构成的字节 b 看成系数在 $\{0, 1\}$ 中的多项式

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

例如: 十六进制数 '57' 对应的二进制为 01010111, 看成一个字节, 对应的多项式为 $x^3 + x^4 + x^5 + x^6 + 1$ 。

在多项式表示中, $GF(2^8)$ 上两个元素的和仍然是一个次数不超过 7 的多项式, 其系数等于两个元素对应系数的模 2 加(比特异或)。

例如: '57' + '83' = 'D4', 用多项式表示为

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \pmod{m(x)}$$

用二进制表示为

$$01010111 + 10000011 = 11010100$$

由于每个元素的加法逆元等于自己, 所以减法和加法相同。

要计算 $GF(2^8)$ 上的乘法, 必须先确定一个 $GF(2)$ 上的 8 次不可约多项式; $GF(2^8)$ 上两个元素的乘积就是这两个多项式的模乘(以这个 8 次不可约多项式为模)。在 Rijndael 密码中, 这个 8 次不可约多项式确定为

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

它的十六进制表示为 '11B'。

例如, '57' · '83' = 'C1' 可表示为以下的多项式乘法:

$$(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) = x^7 + x^6 + 1 \pmod{m(x)}$$

乘法运算虽然不是标准的按字节的运算, 但也是比较简单的计算部件。

以上定义的乘法满足交换律, 且有单位元 '01'。另外, 对任何次数小于 8 的多项式 $b(x)$, 可用推广的欧几里得算法得

$$b(x)a(x) + m(x)c(x) = 1$$

即 $a(x) \cdot b(x) = 1 \bmod m(x)$, 因此 $a(x)$ 是 $b(x)$ 的乘法逆元。再者, 乘法还满足分配律:

$$a(x) \cdot (b(x) + c(x)) = a(x) \cdot b(x) + a(x) \cdot c(x)$$

所以, 256 个字节值构成的集合, 在以上定义的加法和乘法运算下, 有有限域 $GF(2^8)$ 的结构。

$GF(2^8)$ 上还定义了一个运算, 称之为 x 乘法, 其定义为

$$x \cdot b(x) = b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x \pmod{m(x)}$$

如果 $b_7 = 0$, 求模结果不变, 否则为乘积结果减去 $m(x)$, 即求乘积结果与 $m(x)$ 的异或。由此得出 x (十六进制数 '02') 乘 $b(x)$ 可以先对 $b(x)$ 在字节内左移一位 (最后一位补 0), 若 $b_7 = 1$, 则再与 '1B' (其二进制为 00011011) 做逐比特异或来实现, 该运算记为 $b = \text{xtime}(a)$ 。在专用芯片中, xtime 只需 4 个异或。 x 的幂乘运算可以重复应用 xtime 来实现。而任意常数乘法可以通过对中间结果相加实现。

例如, '57' · '13' 可按如下方式实现:

$$\begin{aligned} '57' \cdot '02' &= \text{xtime}(57) = 'AE'; \\ '57' \cdot '04' &= \text{xtime}(AE) = '47'; \\ '57' \cdot '08' &= \text{xtime}(47) = '8E'; \\ '57' \cdot '10' &= \text{xtime}(8E) = '07'; \\ '57' \cdot '13' &= '57' \cdot ('01' \oplus '02' \oplus '10') \\ &= '57' \oplus 'AE' \oplus '07' = 'FE'。 \end{aligned}$$

2. 系数在 $GF(2^8)$ 上的多项式

4 个字节构成的向量可以表示为系数在 $GF(2^8)$ 上的次数小于 4 的多项式。多项式的加法就是对应系数相加; 换句话说, 多项式的加法就是 4 字节向量的逐比特异或。

规定多项式的乘法运算必须要取模 $M(x) = x^4 + 1$, 这样使得次数小于 4 的多项式的乘积仍然是一个次数小于 4 的多项式, 将多项式的模乘运算记为 \otimes , 设 $a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$, $b(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0$, $c(x) = a(x) \otimes b(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0$ 。由于 $x^j \bmod (x^4 + 1) = x^{j \bmod 4}$, 所以

$$\begin{aligned} c_0 &= a_0 b_0 \oplus a_3 b_1 \oplus a_2 b_2 \oplus a_1 b_3; \\ c_1 &= a_1 b_0 \oplus a_0 b_1 \oplus a_3 b_2 \oplus a_2 b_3; \\ c_2 &= a_2 b_0 \oplus a_1 b_1 \oplus a_0 b_2 \oplus a_3 b_3; \\ c_3 &= a_3 b_0 \oplus a_2 b_1 \oplus a_1 b_2 \oplus a_0 b_3。 \end{aligned}$$

可将上述计算表示为

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

注意到 $M(x)$ 不是 $GF(2^8)$ 上的不可约多项式 (甚至也不是 $GF(2)$ 上的不可约多项式), 因此非 0 多项式的这种乘法不是群运算。不过在 Rijndael 密码中, 对多项式 $b(x)$, 这种乘法运算只限于乘一个固定的有逆元的多项式 $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ 。

定理 3-1 系数在 $GF(2^8)$ 上的多项式 $a_3x^3 + a_2x^2 + a_1x + a_0$ 是模 $x^4 + 1$ 可逆的, 当且仅当矩阵

$$\begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix}$$

在 $GF(2^8)$ 上可逆。

证明: $a_3x^3 + a_2x^2 + a_1x + a_0$ 是模 $x^4 + 1$ 可逆的, 当且仅当存在多项式 $h_3x^3 + h_2x^2 + h_1x + h_0$ 使得

$$(a_3x^3 + a_2x^2 + a_1x + a_0)(h_3x^3 + h_2x^2 + h_1x + h_0) = 1 \pmod{x^4 + 1}$$

因此有

$$(a_3x^3 + a_2x^2 + a_1x + a_0)(h_2x^3 + h_1x^2 + h_0x + h_3) = x \pmod{x^4 + 1}$$

$$(a_3x^3 + a_2x^2 + a_1x + a_0)(h_1x^3 + h_0x^2 + h_3x + h_2) = x^2 \pmod{x^4 + 1}$$

$$(a_3x^3 + a_2x^2 + a_1x + a_0)(h_0x^3 + h_3x^2 + h_2x + h_1) = x^3 \pmod{x^4 + 1};$$

将以上关系写成矩阵形式即得

$$\begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} h_0 & h_3 & h_2 & h_1 \\ h_1 & h_0 & h_3 & h_2 \\ h_2 & h_1 & h_0 & h_3 \\ h_3 & h_2 & h_1 & h_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{证毕})$$

$c(x) = x \otimes b(x)$ 定义为 x 与 $b(x)$ 的模 $x^4 + 1$ 乘法, 即 $c(x) = x \otimes b(x) = b_2x^4 + b_1x^2 + b_0x + b_3$ 。其矩阵表示中, 除 $a_1 = '01'$ 外, 其他所有 $a_i = '00'$, 即

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

因此, x (或 x 的幂) 模乘多项式相当于对字节构成的向量进行字节循环移位。

3. 设计思想

Rijndael 密码的设计力求满足以下 3 条标准:

- ① 抵抗所有已知的攻击。
- ② 在多个平台上速度快, 编码紧凑。
- ③ 设计简单。

当前的大多数分组密码, 其轮函数是 Feistel 结构, 即将中间状态的部分比特不加改变地简单放置到其他位置。Rijndael 没有这种结构, 其轮函数是由 3 个不同的可逆均匀变换组成的, 称它们为 3 个“层”。所谓“均匀变换”是指状态的每个比特都是用类似的方法进行处理的。不同层的特定选择大部分是建立在“宽轨迹策略”的应用基础上的。简单地说, “宽轨迹策略”就是提供抗线性密码分析和差分密码分析能力的一种设计。为实现宽轨迹策略, 轮函数 3 个层中的每一层都有它自己的功能:

- 线性混合层 确保多轮之上的高度扩散;
- 非线性层 将具有最优的“最坏情况非线性特性”的 S 盒并行使用;
- 密钥加层 单轮子密钥简单地异或到中间状态上、实现一次性掩盖。

在第一轮之前, 用了一个初始密钥加层, 其目的是在不知道密钥的情况下, 对最后一个密钥加层以后的任一层 (或者是当进行已知明文攻击时, 对第一个密钥加层以前的任一层) 可简单地剥去, 因此初始密钥加层对密码的安全性无任何意义。许多密码的设计中都在轮变换之前和之后用了密钥加层, 如 IDEA、SAFER 和 Blowfish。

为了使加密算法和解密算法在结构上更加接近, 最后一轮的线性混合层与前面各轮的线性混合层不同, 这类似于 DES 的最后一轮不做左右交换。可以证明这种设计不以任何方式提高或降低该密码的安全性。

3.6.2 算法说明

Rijndael 是一个迭代型分组密码, 其分组长度和密钥长度都可变, 各自可以独立地指定为 128 比特、192 比特、256 比特。

1. 状态、种子密钥和轮数

类似于明文分组和密文分组, 算法的中间结果也须分组, 称算法中间结果的分组为状态, 所有的操作都在状态上进行。状态可以用以字节为元素的矩阵阵列表示, 该阵列有 4 行, 列数记为 N_b , N_b 等于分组长度除以 32。

种子密钥类似地用一个以字节为元素的矩阵阵列表示, 该阵列有 4 行, 列数记为 N_k , N_k 等于分组长度除以 32。表 3-8 是 $N_b=6$ 的状态和 $N_k=4$ 的种子密钥的矩阵阵列表示。

表 3-8 $N_b=6$ 的状态和 $N_k=4$ 的种子密钥

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

有时可将这些分组当作一维数组,其每一元素是上述阵列表示中的 4 字节元素构成的列向量,数组长度可为 4、6、8,数组元素下标的范围分别是 $0\sim 3$ 、 $0\sim 5$ 和 $0\sim 7$ 。4 字节元素构成的列向量有时也称为字。

算法的输入和输出被看成是由 8 比特字节构成的一维数组,其元素下标的范围是 $0\sim (4N_b-1)$,因此输入和输出以字节为单位的分组长度分别是 16、24 和 32,其元素下标的范围分别是 $0\sim 15$ 、 $0\sim 23$ 和 $0\sim 31$ 。输入的种子密钥也看成是由 8 比特字节构成的一维数组,其元素下标的范围是 $0\sim (4N_k-1)$,因此种子密钥以字节为单位的分组长度也分别是 16、24 和 32,其元素下标的范围分别是 $0\sim 15$ 、 $0\sim 23$ 和 $0\sim 31$ 。

算法的输入(包括最初的明文输入和中间过程的轮输入)以字节为单位按 $a_{0,0} a_{1,0} a_{2,0} a_{3,0} a_{0,1} a_{1,1} a_{2,1} a_{3,1} \cdots$ 的顺序放置到状态阵列中。同理,种子密钥以字节为单位按 $k_{0,0} k_{1,0} k_{2,0} k_{3,0} k_{0,1} k_{1,1} k_{2,1} k_{3,1} \cdots$ 的顺序放置到种子密钥阵列中。而输出(包括中间过程的轮输出和最后的密文输出)也是以字节为单位按相同的顺序从状态阵列中取出。若输入(或输出)分组中第 n 个元素对应于状态阵列的第 (i, j) 位置上的元素,则 n 和 (i, j) 有以下关系:

$$i = n \bmod 4; \quad j = \lfloor n/4 \rfloor; \quad n = i + 4j$$

迭代的轮数记为 N_r , N_r 与 N_b 和 N_k 有关,表 3-9 给出了 N_r 与 N_b 和 N_k 的关系。

表 3-9 迭代轮数 N_r 为 N_b 和 N_k 的函数

N_r	$N_b=4$	$N_b=6$	$N_b=8$
$N_k=4$	10	12	14
$N_k=6$	12	12	14
$N_k=8$	14	14	14

2. 轮函数

Rijndael 的轮函数由 4 个不同的计算部件组成,分别是:字节代换(ByteSub)、行移位(ShiftRow)、列混合(MixColumn)、密钥加(AddRoundKey)。

(1) 字节代换(ByteSub)

字节代换是非线性变换,独立地对状态的每个字节进行。代换表(即 S-盒)是可逆

的,由以下两个变换的合成得到:

- ① 首先,将字节看作 $GF(2^8)$ 上的元素,映射到自己的乘法逆元,‘00’映射到自己。
- ② 其次,对字节做如下的 $(GF(2))$ 上的,可逆的)仿射变换:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

上述 S-盒对状态的所有字节所做的变换记为

ByteSub (State)

图 3-19 是字节代换示意图。

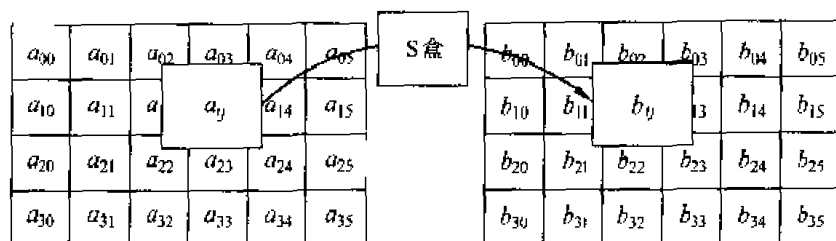


图 3-19 字节代换示意图

ByteSub 的逆变换由代换表的逆表做字节代换,可通过如下两步实现: 首先进行仿射变换的逆变换,再求每一字节在 $GF(2^8)$ 上逆元。

(2) 行移位(ShiftRow)

行移位是将状态阵列的各行进行循环移位,不同状态行的位移量不同。第 0 行不移动,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。位移量 C_1 、 C_2 、 C_3 的取值与 N_s 有关,由表 3-10 给出。

表 3-10 对应于不同分组长度的位移量

N_s	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

按指定的位移量对状态的行进行的行移位运算记为

ShiftRow (State)

图 3-20 是行移位示意图。

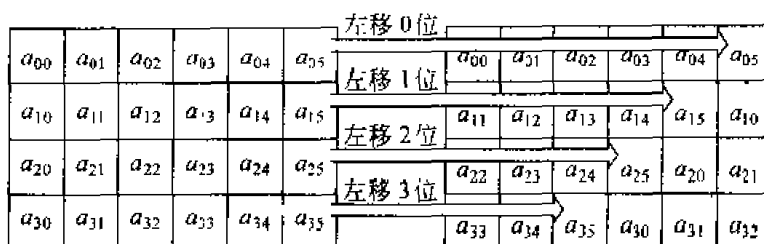


图 3-20 行移位示意图

ShiftRow 的逆变换是对状态阵列的后 3 列分别以位移量 $N_b - C_1$ 、 $N_b - C_2$ 、 $N_b - C_3$ 进行循环移位,使得第 i 行第 j 列的字节移位到 $(j + N_b - C_i) \bmod N_b$ 。

(3) 列混合(MixColumn)

在列混合变换中,将状态阵列的每个列视为 $GF(2^8)$ 上的多项式,再与一个固定的多项式 $c(x)$ 进行模 $x^4 + 1$ 乘法。当然要求 $c(x)$ 是模 $x^4 + 1$ 可逆的多项式,否则列混合变换就是不可逆的,因而会使不同的输入分组对应的输出分组可能相同。Rijndael 的设计者给出的 $c(x)$ 为(系数用十六进制数表示):

$$c(x) = '03'x^3 + '01'x^2 + '01'x + '02'$$

$c(x)$ 是与 $x^4 + 1$ 互素的,因此是模 $x^4 + 1$ 可逆的。列混合运算也可写为矩阵乘法。设 $b(x) = c(x) \otimes a(x)$, 则

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

这个运算需要做 $GF(2^8)$ 上的乘法,但由于所乘的因子是 3 个固定的元素 02、03、01,所以这些乘法运算仍然是比较简单的。

对状态 State 的所有列所做的列混合运算记为

MixColumn(State)

图 3-21 是列混合运算示意图。

列混合运算的逆运算是类似的,即每列都用一个特定的多项式 $d(x)$ 相乘。 $d(x)$ 满足

$$('03'x^3 + '01'x^2 + '01'x + '02') \otimes d(x) = '01'$$

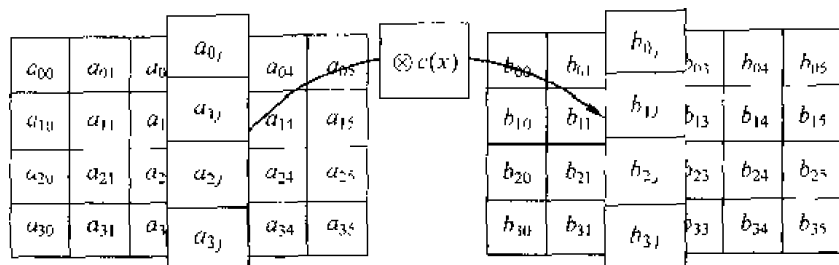


图 3-21 列混合运算示意图

由此可得

$$d(x) = '0B'x^3 + '0D'x^2 + '09'x + '0E'$$

(4) 密钥加 (AddRoundKey)

密钥加是将轮密钥简单地与状态进行逐比特异或。轮密钥由种子密钥通过密钥编排算法得到, 轮密钥长度等于分组长度 N_b 。

状态 State 与轮密钥 RoundKey 的密钥加运算表示为

$$\text{AddRoundKey}(\text{State}, \text{RoundKey})$$

图 3-22 是密钥加运算示意图。

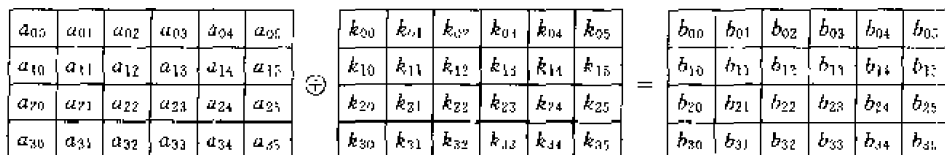


图 3-22 密钥加运算示意图

密钥加运算的逆运算是其自身。

综上所述, 组成 Rijndael 轮函数的计算部件简捷快速, 功能互补。轮函数的伪 C 代码如下:

```
Round (State, RoundKey)
{
    ByteSub (State);
    ShiftRow (State);
    MixColumn (State);
    AddRoundKey (State, RoundKey)
}
```

结尾轮的轮函数与前面各轮不同, 将 MixColumn 这一步去掉, 其伪 C 代码如下:


```

FinalRound (State, RoundKey)
{
    ByteSub (State);
    ShiftRow (State);
    AddRoundKey (State, RoundKey)
}

```

在以上伪 C 代码记法中, State、RoundKey 可用指针类型, 函数 Round、FinalRound、ByteSub、ShiftRow、MixColumn、AddRoundKey 都在指针 State、RoundKey 所指向的阵列上进行运算。

3. 密钥编排

密钥编排指从种子密钥得到轮密钥的过程, 它由密钥扩展和轮密钥选取两部分组成。其基本原则如下:

- 轮密钥的比特数等于分组长度乘以轮数加 1;
- 种子密钥被扩展成为扩展密钥;
- 轮密钥从扩展密钥中取, 其中第 1 轮轮密钥取扩展密钥的前 N_b 个字, 第 2 轮轮密钥取接下来的 N_b 个字, 如此下去。

(1) 密钥扩展

扩展密钥是以 4 字节字为元素的一维阵列, 表示为 $W[N_b * (N_r + 1)]$, 其中前 N_k 个字取为种子密钥, 以后每个字按递归方式定义。扩展算法根据 $N_k \leq 6$ 和 $N_k > 6$ 有所不同。

① 当 $N_k \leq 6$ 时, 扩展算法如下

```

KeyExpansion (byte Key[4 * Nk], W[Nb * (Nr + 1)])
{
    for (i = 0; i < Nk; i++)
        W[i] = (Key[4 * i], Key[4 * i + 1], Key[4 * i + 2], Key[4 * i + 3]);
    for (i = Nk; i < Nb * (Nr + 1); i++)
    {
        temp = W[i - 1];
        if (i % Nb == 0)
            temp = SubByte (RotByte (temp)) * Rcon[i / Nb];
        W[i] = W[i - Nb] * temp;
    }
}

```

其中 $\text{Key}[4 * N_k]$ 为种子密钥, 看作以字节为元素的一维阵列。函数 $\text{SubByte}()$ 返回 4 字节字, 其中每一个字节都是用 Rijndael 的 S 盒作用到输入字对应的字节得到。函数 $\text{RotByte}()$ 也返回 4 字节字, 该字由输入的字循环移位得到, 即当输入字为 (a, b, c, d) 时, 输出字为 (b, c, d, a) 。

可以看出, 扩展密钥的前 N_k 个字即为种子密钥, 之后的每个字 $W[i]$ 等于前一个字 $W[i-1]$ 与 N_k 个位置之前的字 $W[i-N_k]$ 的异或; 不过当 i/N_k 为整数时, 须先将前一个字 $W[i-1]$ 经过以下一系列的变换:

1 字节的循环移位 RotByte → 用 S 盒进行变换 SubByte → 异或轮常数 $\text{Rcon}[i/N_k]$ 。

② 当 $N_k > 6$ 时, 扩展算法如下:

```
KeyExpansion (byte Key[4 * Nk], W[Nk * (Nr + 1)])
{
    for (i=0; i < Nk; i++)
        W[i] = (Key[4 * i], Key[4 * i + 1], Key[4 * i + 2], Key[4 * i + 3]);
    for (i = Nk; i < Nk * (Nr + 1); i++)
    {
        temp = W[i - 1];
        if (i % Nk == 0)
            temp = SubByte (RotByte (temp)) ^ Rcon[i / Nk];
        else if (i % Nk == 4)
            temp = SubByte (temp);
        W[i] = W[i - Nk] ^ temp;
    }
}
```

$N_k > 6$ 与 $N_k \leq 6$ 的密钥扩展算法的区别在于: 当 i 为 N_k 的 4 的倍数时, 须先将前一个字 $W[i-1]$ 经过 SubByte 变换。

以上两个算法中, $\text{Rcon}[i/N_k]$ 为轮常数, 其值与 N_k 无关, 定义为 (字节用十六进制表示, 同时理解为 $GF(2^8)$ 上的元素):

$$\text{Rcon}[i] = (\text{RC}[i], '00', '00', '00')$$

其中 $\text{RC}[i]$ 是 $GF(2^8)$ 中值为 x^{i-1} 的元素, 因此

$$\text{RC}[1] = 1 \text{ (即 '01')}$$

$$\text{RC}[i] = x \text{ (即 '02')} \cdot \text{RC}[i-1] = x^{i-1}$$

(2) 轮密钥选取

轮密钥 i (即第 i 个轮密钥) 由轮密钥缓冲字 $W[N_k * i]$ 到 $W[N_k * (i+1)]$ 给出, 如

图 3-23 所示。



图 3-23 $N_b=6$ 且 $N_r=4$ 时的密钥扩展与轮密钥选取

4. 加密算法

加密算法为顺序完成以下操作：初始的密钥加； (N_r-1) 轮迭代；一个结尾轮。即

```
Rijndael (State, CipherKey)
{
    KeyExpansion (CipherKey, ExpandedKey);
    AddRoundKey (State, ExpandedKey);
    for ( $i=1$ ;  $i < N_r$ ;  $i++$ ) Round (State, ExpandedKey +  $N_b * i$ );
    FinalRound (State, ExpandedKey +  $N_b * N_r$ )
}
```

其中 CipherKey 是种子密钥, ExpandedKey 是扩展密钥。密钥扩展可以事先进行(预计算),且 Rijndael 密码的加密算法可以用这一扩展密钥来描述,即

```
Rijndael (State, ExpandedKey)
{
    AddRoundKey (State, ExpandedKey);
    for ( $i=1$ ;  $i < N_r$ ;  $i++$ ) Round (State, ExpandedKey +  $N_b * i$ );
    FinalRound (State, ExpandedKey +  $N_b * N_r$ )
}
```

5. 加解密的相近程度及解密算法

首先给出几个引理。

引理 3-1 设字节代换(ByteSub)、行移位(ShiftRow)的逆变换分别为 InvByteSub、InvShiftRow。则组合部件“ByteSub \rightarrow ShiftRow”的逆变换为“InvByteSub \rightarrow InvShiftRow”。

证明：组合部件“ByteSub \rightarrow ShiftRow”的逆变换原本为“InvShiftRow \rightarrow InvByteSub”。由于字节代换(ByteSub)是对每个字节进行相同的变换,故“InvShiftRow”与“InvByteSub”两个计算部件可以交换顺序。(证毕)

引理 3-2 设列混合(MixColumn)的逆变换为 InvMixColumn。则列混合部件与密钥加部件(AddRoundKey)的组合部件“MixColumn \rightarrow AddRoundKey(\cdot , Key)”的逆变换为“InvMixColumn \rightarrow AddRoundKey(\cdot , InvKey)”。其中密钥 InvKey 与 Key 的关系为: InvKey=InvMixColumn(Key)。

证明: 组合部件“MixColumn \rightarrow AddRoundKey(\cdot , Key)”的逆变换原本为“AddRoundKey(\cdot , Key) \rightarrow InvMixColumn”, 由于列混合(MixColumn)实际上是线性空间 $GF(2^8)^4$ 上的可逆线性变换, 因此

$$\begin{aligned} & \text{“AddRoundKey}(\cdot, \text{Key}) \rightarrow \text{InvMixColumn} \text{”} \\ &= \text{“InvMixColumn} \rightarrow \text{AddRoundKey}(\cdot, \text{InvMixColumn}(\text{Key})) \text{”} \quad (\text{证毕}) \end{aligned}$$

引理 3-3 将某一轮的后两个计算部件和下--轮的前两个计算部件组成组合部件, 该组合部件的程序为

```
MixColumn (State);
AddRoundKey (State, Key(i));
ByteSub (State);
ShiftRow (State)
```

则该组合部件的逆变换程序为

```
InvByteSub (State);
InvShiftRow (State);
InvMixColumn (State);
AddRoundKey (State, InvMixColumn (Key(i)))
```

证明: 这是引理 3-1 和引理 3-2 的直接推论。

注意到在引理 3-3 所描述的逆变换中, 第 2 步到第 4 步在形状上很像加密算法的轮函数, 这将是解密算法的轮函数。注意到结尾轮只有 3 个计算部件, 因此可以得到如下定理。

定理 3-2 Rijndael 密码的解密算法为顺序完成以下操作: 初始的密钥加; $(N, -1)$ 轮迭代; 一个结尾轮。其中解密算法的轮函数为

```
InvRound (State, RoundKey)
{
    InvByteSub (State);
    InvShiftRow (State);
    InvMixColumn (State);
    AddRoundKey (State, RoundKey)
}
```

解密算法的结尾轮为

```

InvFinalRound (State, RoundKey)
{
    InvByteSub (State);
    InvShiftRow (State);
    AddRoundKey (State, RoundKey)
}

```

设加密算法的初始密钥加、第 1 轮、第 2 轮、…、第 N_r 轮的子密钥依次为

$$k(0), k(1), k(2), \dots, k(N_r - 1), k(N_r)$$

则解密算法的初始密钥加、第 1 轮、第 2 轮、…、第 N_r 轮的子密钥依次为

$$k(N_r), \text{InvMixColumn}(k(N_r - 1)), \text{InvMixColumn}(k(N_r - 2)), \dots, \text{InvMixColumn}(k(1)), k(0).$$

证明：这是上述 3 个引理的直接推论。

综上所述, Rijndael 密码的解密算法与加密算法的计算网络相同, 只是将各计算部件换为对应的逆部件。

习 题

1. (1) 设 M' 是 M 的逐比特取补, 证明在 DES 中, 如果对明文分组和加密密钥都逐比特取补, 那么得到的密文也是原密文的逐比特取补, 即

$$\text{如果 } Y = \text{DES}_K(X)$$

$$\text{那么 } Y' = \text{DES}_{K'}(X')$$

提示：对任意两个长度相等的比特串 A 和 B , 证明 $(A \oplus B)' = A' \oplus B$ 。

(2) 对 DES 进行穷搜索攻击时, 需要在由 2^{56} 个密钥构成的密钥空间进行。能否根据(1)的结论减小进行穷搜索攻击时所用的密钥空间。

2. 证明 DES 的解密变换是加密变换的逆。

3. 在 DES 的 ECB 模式中, 如果在密文分组中有一个错误, 解密后仅相应的明文分组受到影响。然而在 CBC 模式中, 将有错误传播。例如在图 3-11 中 C_1 中的一个错误明显地将影响 P_1 和 P_2 的结果。

(1) P_2 后的分组是否受到影响?

(2) 设加密前的明文分组 P_1 中有 1 比特的错误, 问这一错误将在多少个密文分组中传播? 对接收者产生什么影响?

4. 在 8 比特 CFB 模式中, 如果在密文字符中出现 1 比特的错误, 问该错误能传播

多远?

5. 在实现 IDEA 时,最困难的部分是模 $2^{16}+1$ 乘法运算。以下关系给出了实现模乘法的一种有效方法,其中 a 和 b 是两个 n 比特的非 0 整数:

$$ab \bmod (2^n + 1) = \begin{cases} (ab \bmod 2^n) - (ab \operatorname{div} 2^n), & (ab \bmod 2^n) \geq (ab \operatorname{div} 2^n) \\ (ab \bmod 2^n) - (ab \operatorname{div} 2^n) + 2^n - 1, & (ab \bmod 2^n) \leq (ab \operatorname{div} 2^n) \end{cases}$$

注意: $(ab \bmod 2^n)$ 相应于 ab 的 n 个最低有效位, $(ab \operatorname{div} 2^n)$ 是 ab 右移 n 位。

(1) 证明存在惟一的非负整数 q 和 r 使得 $ab = q(2^n + 1) + r$ 。

(2) 求 q 和 r 的上下界。

(3) 证明 $q + r < 2^{n+1}$ 。

(4) 求 $(ab \operatorname{div} 2^n)$ 关于 q 的表达式。

(5) 求 $(ab \bmod 2^n)$ 关于 q 和 r 的表达式。

(6) 用(4)和(5)的结果求 r 的表达式,说明 r 的含义。

6. (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,为什么将模数取为 2^{16} 而不是 $2^{16}+1$?

第4章

公钥密码

数论是密码学特别是公钥密码学的基本工具,本章首先介绍密码学中常用的一些数论知识,然后介绍公钥密码体制的基本概念和几种重要算法。

4.1 数论简介

4.1.1 素数和互素数

1. 因子

设 $a, b (b \neq 0)$ 是两个整数, 如果存在另一整数 m , 使得 $a = mb$, 则称 b 整除 a , 记为 $b|a$, 且称 b 是 a 的因子。

整数具有以下性质:

① $a|1$, 那么 $a = \pm 1$ 。

② $a|b$ 且 $b|a$, 则 $a = \pm b$ 。

③ 对任一 $b (b \neq 0), b|0$ 。

④ $b|g, b|h$, 则对任意整数 m, n 有 $b|(mg + nh)$ 。

这里只给出④的证明, 其他3个性质的证明都很简单。

性质④的证明: 由 $b|g, b|h$ 知, 存在整数 g_1, h_1 , 使得

$$g = bg_1, \quad h = bh_1$$

所以 $mg + nh = mbg_1 + nbh_1 = b(mg_1 + nh_1)$, 因此 $b|(mg + nh)$ 。

2. 素数

称整数 $p (p > 1)$ 是素数, 如果 p 的因子只有 $\pm 1, \pm p$ 。

任一整数 $a (a > 1)$ 都能惟一地分解为以下形式:

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

其中 $p_1 > p_2 > \cdots > p_t$ 是素数, $a_i > 0 (i = 1, \cdots, t)$ 。例如

$$91 = 7 \times 13, \quad 11011 = 7 \times 11^2 \times 13$$

这一性质称为整数分解的惟一性,也可如下陈述:

设 P 是所有素数集合,则任意整数 a ($a > 1$) 都能惟一地写成以下形式:

$$a = \prod_{p \in P} p^{a_p}$$

其中 $a_p \geq 0$,等号右边的乘积项取所有的素数,然而大多指数项 a_p 为 0。相应地,任一正整数也可由非 0 指数列表表示。例如:11011 可表示为 $\{a_7=1, a_{11}=2, a_{13}=1\}$ 。

两数相乘等价于对应的指数相加,即由 $k=mn$ 可得:对每一素数 $p, k_p = m_p + n_p$ 。而由 $a|b$ 可得:对每一素数 $p, a_p \leq b_p$ 。这是因为 p^k 只能被 p^j ($j \leq k$) 整除。

3. 互素数

称 c 是两个整数 a, b 的最大公因子,如果

① c 是 a 的因子也是 b 的因子,即 c 是 a, b 的公因子。

② a 和 b 的任一公因子,也是 c 的因子。

表示为 $c = \gcd(a, b)$ 。

由于要求最大公因子为正,所以 $\gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$ 。一般 $\gcd(a, b) = \gcd(|a|, |b|)$ 。由任一非 0 整数能整除 0,可得 $\gcd(a, 0) = |a|$ 。如果将 a, b 都表示为素数的乘积,则 $\gcd(a, b)$ 极易确定。

例如:

$$300 = 2^2 \times 3^1 \times 5^2$$

$$18 = 2^1 \times 3^2$$

$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

一般由 $c = \gcd(a, b)$ 可得:对每一素数 $p, c_p = \min(a_p, b_p)$ 。

由于确定大数的素因子不很容易,所以这种方法不能直接用于求两个大数的最大公因子,如何求两个大数的最大公因子在下面介绍。

如果 $\gcd(a, b) = 1$,则称 a 和 b 互素。

4.1.2 模运算

设 n 是一正整数, a 是整数,如果用 n 除 a ,得商为 q ,余数为 r ,则

$$a = qn + r, \quad 0 \leq r < n, \quad q = \lfloor a/n \rfloor$$

其中 $\lfloor x \rfloor$ 为小于或等于 x 的最大整数。

用 $a \bmod n$ 表示余数 r ,则 $a = \lfloor a/n \rfloor n + a \bmod n$ 。

如果 $(a \bmod n) = (b \bmod n)$,则称两整数 a 和 b 模 n 同余,记为 $a \equiv b \pmod{n}$ 。称与 a 模 n 同余的数的全体为 a 的同余类,记为 $[a]$,称 a 为这个同余类的表示元素。

注意: 如果 $a \equiv 0 \pmod{n}$, 则 $n \mid a$ 。

同余有以下性质:

- ① 若 $n \mid (a-b)$, 则 $a \equiv b \pmod{n}$ 。
- ② $(a \pmod{n}) \equiv (b \pmod{n})$, 则 $a \equiv b \pmod{n}$ 。
- ③ $a \equiv b \pmod{n}$, 则 $b \equiv a \pmod{n}$ 。
- ④ $a \equiv b \pmod{n}, b \equiv c \pmod{n}$, 则 $a \equiv c \pmod{n}$ 。

从以上性质易知, 同余类中的每一元素都可作为这个同余类的表示元素。

求余数运算(简称求余运算) $a \pmod{n}$ 将整数 a 映射到集合 $\{0, 1, \dots, n-1\}$, 称求余运算在这个集合上的算术运算为模运算, 模运算有以下性质:

- ① $[(a \pmod{n}) + (b \pmod{n})] \pmod{n} = (a+b) \pmod{n}$ 。
- ② $[(a \pmod{n}) - (b \pmod{n})] \pmod{n} = (a-b) \pmod{n}$ 。
- ③ $[(a \pmod{n}) \times (b \pmod{n})] \pmod{n} = (a \times b) \pmod{n}$ 。

性质①的证明: 设 $(a \pmod{n}) = r_a, (b \pmod{n}) = r_b$, 则存在整数 j, k 使得 $a = jn + r_a, b = kn + r_b$ 。

因此

$$\begin{aligned} (a+b) \pmod{n} &= [(j+k)n + r_a + r_b] \pmod{n} = (r_a + r_b) \pmod{n} \\ &= [(a \pmod{n}) + (b \pmod{n})] \pmod{n} \end{aligned} \quad (\text{证毕})$$

性质②、③的证明类似。

【例 4-1】 设 $Z_8 = \{0, 1, \dots, 7\}$, 考虑 Z_8 上的模加法和模乘法, 结果如表 4-1 所示。

表 4-1 模 8 运算

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

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

从加法结果可见, 对每一 x , 都有一 y , 使得 $x+y \equiv 0 \pmod{8}$ 。如对 2, 有 6, 使得 $2+6 \equiv 0 \pmod{8}$, 称 y 为 x 的负数, 也称为加法逆元。

对 x , 若有 y , 使得 $x \times y \equiv 1 \pmod{8}$, 如 $3 \times 3 \equiv 1 \pmod{8}$, 则称 y 为 x 的倒数, 也称为乘法逆元。本例可见并非每一 x 都有乘法逆元。

一般地,定义 Z_n 为小于 n 的所有非负整数集合,即

$$Z_n = \{0, 1, \dots, n-1\},$$

称 Z_n 为模 n 的同余类集合。其上的模运算有以下性质:

① 交换律 $(w+x) \bmod n = (x+w) \bmod n$

$$(w \times x) \bmod n = (x \times w) \bmod n$$

② 结合律 $[(w+x)+y] \bmod n = [w+(x+y)] \bmod n$

$$[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$$

③ 分配律 $[w \times (x+y)] \bmod n = [w \times x + w \times y] \bmod n$

④ 单位元 $(0+w) \bmod n = w \bmod n$

$$(1 \times w) \bmod n = w \bmod n$$

⑤ 加法逆元对 $w \in Z_n$, 存在 $z \in Z_n$, 使得 $w+z \equiv 0 \bmod n$, 记 $z = -w$ 。

此外还有以下性质:

如果 $(a+b) \equiv (a+c) \bmod n$, 则 $b \equiv c \bmod n$, 称为加法的可约律。

该性质可由 $(a+b) \equiv (a+c) \bmod n$ 的两边同加上 a 的加法逆元得到。

然而类似性质对乘法却不一定成立。例如 $6 \times 3 \equiv 6 \times 7 \equiv 2 \bmod 8$, 但 $3 \not\equiv 7 \bmod 8$ 。原因是 6 乘以 0 到 7 得到的 8 个数仅为 Z_8 的一部分, 见例 4-1。即如果将对 Z_8 作 6 的乘法 $6 \times Z_8$ (即用 6 乘 Z_8 中每一数) 看作 Z_8 到 Z_8 的映射的话, Z_8 中至少有两个数映射到同一数, 因此该映射为多到一的, 所以对 6 来说, 没有惟一的乘法逆元。但对 5 来说, $5 \times 5 \equiv 1 \bmod 8$, 因此 5 有乘法逆元 5。仔细观察可见, 与 8 互素的几个数 1, 3, 5, 7 都有乘法逆元。

这一结论可推广到任一 Z_n 。

定理 4-1 设 $a \in Z_n$, $\gcd(a, n) = 1$, 则 a 在 Z_n 中有乘法逆元。

证明: 首先证明 a 与 Z_n 中任意两个不相同的数 b, c (不妨设 $c < b$) 相乘, 其结果必然不同。否则设 $a \times b \equiv a \times c \bmod n$, 则存在两个整数 k_1, k_2 , 使得 $ab = k_1n + r, ac = k_2n + r$, 可得 $a(b-c) = (k_1-k_2)n$, 所以 a 是 $(k_1-k_2)n$ 的一个因子。又由 $\gcd(a, n) = 1$, 得 a 是 k_1-k_2 的一个因子, 设 $k_1-k_2 = k_3a$, 所以 $a(b-c) = k_3an$, 即 $b-c = k_3n$, 与 $0 < c < b < n$ 矛盾。所以 $|a \times Z_n| = |Z_n|$, 又知 $a \times Z_n \subseteq Z_n$, 所以 $a \times Z_n = Z_n$ 。因此对 $1 \in Z_n$, 存在 $x \in Z_n$, 使得 $a \times x \equiv 1 \bmod n$, 即 x 是 a 的乘法逆元。记为 $x = a^{-1}$ 。(证毕)

设 p 为一素数, 则 Z_p 中每一非 0 元素都与 p 互素, 因此有乘法逆元。类似于加法可约律, 可有以下乘法可约律:

如果 $(a \times b) \equiv (a \times c) \bmod n$ 且 a 有乘法逆元, 那么对 $(a \times b) \equiv (a \times c) \bmod n$ 两边同乘以 a^{-1} , 即得 $b \equiv c \bmod n$

4.1.3 费尔玛定理和欧拉定理

费尔玛 (Fermat) 定理和欧拉 (Euler) 定理在公钥密码体制中起着重要作用。

1. 费尔玛定理

定理 4-2(Fermat) 若 p 是素数, a 是正整数且 $\gcd(a, p)=1$, 则 $a^{p-1} \equiv 1 \pmod{p}$.

证明: 由 4.1.2 的讨论知当 $\gcd(a, p)=1$ 时, $a \times Z_p = Z_p$, 其中 $a \times Z_p$ 表示 a 与 Z_p 中每一元素做模 p 乘法。又知 $a \times 0 \equiv 0 \pmod{p}$, 所以 $a \times Z_p - \{0\} = Z_p - \{0\}$, $a \times (Z_p - \{0\}) = Z_p - \{0\}$ 。即

$$\{a \bmod p, 2a \bmod p, \dots, (p-1)a \bmod p\} = \{1, 2, \dots, p-1\}$$

所以

$$a \times 2a \times \dots \times (p-1)a \equiv [(a \bmod p) \times (2a \bmod p) \times \dots \times ((p-1)a \bmod p)] \bmod p \equiv (p-1)! \bmod p$$

另一方面

$$a \times 2a \times \dots \times (p-1)a = (p-1)! a^{p-1}$$

因此

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

由于 $(p-1)!$ 与 p 互素, 因此 $(p-1)!$ 有乘法逆元, 由乘法可约律得 $a^{p-1} \equiv 1 \pmod{p}$ 。

(证毕)

Fermat 定理也可写成如下形式: 设 p 是素数, a 是任一正整数, 则 $a^p \equiv a \pmod{p}$ 。

2. 欧拉函数

设 n 是一正整数, 小于 n 且与 n 互素的正整数的个数称为 n 的欧拉函数, 记为 $\varphi(n)$ 。

例如: $\varphi(6)=2$, $\varphi(7)=6$, $\varphi(8)=4$ 。

若 n 是素数, 则显然有 $\varphi(n)=n-1$ 。

定理 4-3 若 n 是两个素数 p 和 q 的乘积, 则 $\varphi(n) = \varphi(p) \times \varphi(q) = (p-1) \times (q-1)$ 。

证明: 考虑 $Z_n = \{0, 1, \dots, pq-1\}$, 其中不与 n 互素的数有 3 类, $A = \{p, 2p, \dots, (q-1)p\}$, $B = \{q, 2q, \dots, (p-1)q\}$, $C = \{0\}$, 且 $A \cap B = \emptyset$, 否则 $ip = jq$, 其中 $1 \leq i \leq q-1, 1 \leq j \leq p-1$, 则 p 是 jq 的因子, 因此是 j 的因子, 设 $j = kp, k \geq 1$ 。则 $ip = kpq, i = kq$, 与 $1 \leq i \leq q-1$ 矛盾。所以

$$\begin{aligned} \varphi(n) &= |Z_n| - [|A| + |B| + |C|] = pq - [(q-1) + (p-1) + 1] \\ &= (p-1) \times (q-1) = \varphi(p) \times \varphi(q) \end{aligned} \quad (\text{证毕})$$

例如: 由 $21=3 \times 7$, 得 $\varphi(21) = \varphi(3) \times \varphi(7) = 2 \times 6 = 12$ 。

3. 欧拉定理

定理 4-4(Euler) 若 a 和 n 互素, 则 $a^{\varphi(n)} \equiv 1 \pmod{n}$ 。

证明: 设 $R = \{x_1, x_2, \dots, x_{\varphi(n)}\}$ 是由小于 n 且与 n 互素的全体数构成的集合, $a \times R =$

$\{ax_1 \bmod n, ax_2 \bmod n, \dots, ax_{\varphi(n)} \bmod n\}$, 对 $a \times R$ 中任一元素 $ax_i \bmod n$, 因 a 与 n 互素, x_i 与 n 互素, 所以 ax_i 与 n 互素, 且 $ax_i \bmod n < n$, 因此 $ax_i \bmod n \in R$, 所以 $a \times R \subseteq R$ 。

又因 $a \times R$ 中任意两个元素都不相同, 否则 $ax_i \bmod n = ax_j \bmod n$, 由 a 与 n 互素知 a 在 $\bmod n$ 下有乘法逆元, 得 $x_i = x_j$ 。所以 $|a \times R| = |R|$, 得 $a \times R = R$, 所以

$\prod_{i=1}^{\varphi(n)} (ax_i \bmod n) = \prod_{i=1}^{\varphi(n)} x_i, \prod_{i=1}^{\varphi(n)} ax_i \equiv \prod_{i=1}^{\varphi(n)} x_i \pmod{n}, a^{\varphi(n)} \cdot \prod_{i=1}^{\varphi(n)} x_i \equiv \prod_{i=1}^{\varphi(n)} x_i \pmod{n}$, 由每一 x_i 与 n 互素, 知 $\prod_{i=1}^{\varphi(n)} x_i$ 与 n 互素, $\prod_{i=1}^{\varphi(n)} x_i$ 在 $\bmod n$ 下有乘法逆元。所以 $a^{\varphi(n)} \equiv 1 \pmod{n}$ 。

(证毕)

4.1.4 素性检验

素性检验是指对给定的数检验其是否为素数。对于大数的素性检验来说没有简单直接的方法, 本节介绍一个概率检验法, 为此需要以下引理。

引理 如果 p 为大于 2 的素数, 则方程 $x^2 \equiv 1 \pmod{p}$ 的解只有 $x \equiv 1$ 和 $x \equiv -1$ 。

证明: 由 $x^2 \equiv 1 \pmod{p}$, 有 $x^2 - 1 \equiv 0 \pmod{p}, (x+1)(x-1) \equiv 0 \pmod{p}$, 因此 $p|(x+1)$ 或 $p|(x-1)$ 或 $p|(x+1)$ 且 $p|(x-1)$ 。

若 $p|(x+1)$ 且 $p|(x-1)$, 则存在两个整数 k 和 j , 使得 $x+1=kp, x-1=jp$, 两式相减得 $2=(k-j)p$, 为不可能结果。所以有 $p|(x+1)$ 或 $p|(x-1)$ 。

设 $p|(x+1)$, 则 $x+1=kp$, 因此 $x \equiv -1 \pmod{p}$ 。

类似地可得 $x \equiv 1 \pmod{p}$ 。

(证毕)

引理的逆否命题为: 如果方程 $x^2 \equiv 1 \pmod{p}$ 有一解 $x_0 \notin \{-1, 1\}$, 那么 p 不为素数。

例如: 考虑方程 $x^2 \equiv 1 \pmod{8}$ 由 4.1.2 节 \mathbb{Z}_8 上模乘法的结果得

$$1^2 \equiv 1 \pmod{8}, 3^2 \equiv 1 \pmod{8}, 5^2 \equiv 1 \pmod{8}, 7^2 \equiv 1 \pmod{8}$$

又 $5 \equiv -3 \pmod{8}, 7 \equiv -1 \pmod{8}$, 所以方程的解为 $1, -1, 3, -3$, 可见 8 不是素数。

下面介绍 Miller-Rabin 的素性概率检测法。首先将 $n-1$ 表示为二进制形式 $b_k b_{k-1} \dots b_0$, 并给 d 赋初值 1, 则算法 Witness(a, n) 的核心部分如下:

```
for i=k downto 0 do
{
  x ← d;
  d ← (d × d) mod n;
  if d=1 and (x ≠ 1) and (x ≠ n-1) then return False;
  if b_i = 1 then d ← (d × a) mod n
}
```

if $d \neq 1$ then return *False*;
return *True*.

此算法有两个输入参数, n 是待检验的数, a 是小于 n 的整数。如果算法的返回值为 *False*, 则 n 肯定不是素数, 如果返回值为 *True*, 则 n 有可能是素数。

for 循环结束后, 有 $d \equiv a^{n-1} \pmod n$, 由 Fermat 定理知, 若 n 为素数, 则 d 为 1。因此若 $d \neq 1$, 则 n 不为素数, 所以返回 *False*。

因为 $n-1 \equiv -1 \pmod n$, 所以 $(x \neq 1) \text{ and } (x \neq n-1)$, 指 $x^2 \equiv 1 \pmod n$ 有不在 $\{-1, 1\}$ 中的根, 因此 n 不为素数, 返回 *False*。

该算法有以下性质: 对 s 个不同的 a , 重复调用这一算法, 只要有一次算法返回为 *False*, 就可肯定 n 不是素数。如果算法每次返回都为 *True*, 则 n 是素数的概率至少为 $1-2^{-s}$, 因此对于足够大的 s , 就可以非常肯定地相信 n 为素数。

4.1.5 欧几里得算法

欧几里得(Euclid)算法是数论中的一个基本技术, 是求两个正整数的最大公因子的简化过程。而推广的 Euclid 算法不仅可求两个正整数的最大公因子, 而且当两个正整数互素时, 还可求出其中一个数关于另一个数的乘法逆元。

1. 求最大公因子

Euclid 算法是基于下面一个基本结论:

对任意非负整数 a 和正整数 b , 有 $\gcd(a, b) = \gcd(b, a \bmod b)$ 。

证明: b 是正整数, 因此可将 a 表示为 $a = kb + r \equiv r \pmod b$, $a \bmod b = r$, 其中 k 为一整数, 所以 $a \bmod b = a - kb$ 。

设 d 是 a, b 的公因子, 即 $d|a, d|b$, 所以 $d|kb$ 。由 $d|a$ 和 $d|kb$ 得 $d|(a \bmod b)$, 因此 d 是 b 和 $a \bmod b$ 的公因子。

所以得出 a, b 的公因子集合与 $b, a \bmod b$ 的公因子集合相等, 两个集合的最大值也相等。 (证毕)

例如: $\gcd(55, 22) = \gcd(22, 55 \bmod 22) = \gcd(22, 11) = \gcd(11, 0) = 11$ 。

在求两个数的最大公因子时, 可重复使用以上结论。

例如: $\gcd(18, 12) = \gcd(12, 6) = \gcd(6, 0) = 6$,

$\gcd(11, 10) = \gcd(10, 1) = \gcd(1, 0) = 1$ 。

Euclid 算法就是用这种方法, 因 $\gcd(a, b) = \gcd(|a|, |b|)$, 因此可假定算法的输入是两个正整数, 设为 d, f , 并设 $f > d$ 。

Euclid(f, d)

1. $X \leftarrow f; Y \leftarrow d;$
2. if $Y=0$ then return $X=\gcd(f, d);$
3. $R=X \bmod Y;$
4. $X=Y;$
5. $Y=R;$
6. goto 2.

【例 4-2】 求 $\gcd(1970, 1066)$ 。

$$\begin{array}{ll}
 1970=1 \times 1066+904, & \gcd(1066, 904) \\
 1066=1 \times 904+162, & \gcd(904, 162) \\
 904=5 \times 162+94, & \gcd(162, 94) \\
 162=1 \times 94+68, & \gcd(94, 68) \\
 94=1 \times 68+26, & \gcd(68, 26) \\
 68=2 \times 26+16, & \gcd(26, 16) \\
 26=1 \times 16+10, & \gcd(16, 10) \\
 16=1 \times 10+6, & \gcd(10, 6) \\
 10=1 \times 6+4, & \gcd(6, 4) \\
 6=1 \times 4+2, & \gcd(4, 2) \\
 4=2 \times 2+0, & \gcd(2, 0)
 \end{array}$$

因此 $\gcd(1970, 1066)=2$ 。

2. 求乘法逆元

如果 $\gcd(a, b)=1$, 则 b 在 $\bmod a$ 下有乘法逆元(不妨设 $b < a$), 即存在一 x ($x < a$), 使得 $bx \equiv 1 \bmod a$ 。推广的 Euclid 算法先求出 $\gcd(a, b)$, 当 $\gcd(a, b)=1$ 时, 则返回 b 的逆元。

Extended Euclid(f, d) (设 $f > d$)

1. $(X_1, X_2, X_3) \leftarrow (1, 0, f); (Y_1, Y_2, Y_3) \leftarrow (0, 1, d);$
2. if $Y_3=0$ then return $X_3=\gcd(f, d)$; no inverse;
3. if $Y_3=1$ then return $Y_3=\gcd(f, d); Y_2=d^{-1} \bmod f;$
4. $Q = \left\lfloor \frac{X_3}{Y_3} \right\rfloor;$
5. $(T_1, T_2, T_3) \leftarrow (X_1 - QY_1, X_2 - QY_2, X_3 - QY_3);$
6. $(X_1, X_2, X_3) \leftarrow (Y_1, Y_2, Y_3);$
7. $(Y_1, Y_2, Y_3) \leftarrow (T_1, T_2, T_3);$

8. goto 2。

算法中的变量有以下关系：

$$fT_1 + dT_2 = T_3; fX_1 + dX_2 = X_3; fY_1 + dY_2 = Y_3$$

在算法 EUCLID(f, d)中, X 等于前一轮循环中的 Y , Y 等于前一轮循环中的 $X \bmod Y$ 。而在算法 Extended Euclid(f, d)中, X_3 等于前一轮循环中的 Y_3 , Y_3 等于前一轮循环中的 $X_3 - QY_3$, 由于 Q 是 Y_3 除 X_3 的商, 因此 Y_3 是前一轮循环中的 Y_3 除 X_3 的余数, 即 $X_3 \bmod Y_3$, 可见 Extended Euclid(f, d)中的 X_3, Y_3 与 Euclid(f, d)中的 X, Y 作用相同, 因此可正确产生 $\gcd(f, d)$ 。

如果 $\gcd(f, d)=1$, 则在最后一轮循环中 $Y_3=0, X_3=1$, 因此在新一轮循环中 $Y_3=1$ 。由 $Y_3=1$ 可得: $fY_1 + dY_2 = Y_3, fY_1 + dY_2 = 1, dY_2 = 1 + (-Y_1) \times f, dY_2 \equiv 1 \pmod f$, 所以 $Y_2 \equiv d^{-1} \pmod f$ 。

【例 4-3】求 $\gcd(1769, 550)$ 。

算法的运行结果及各变量的变化情况如表 4-2 所示:

表 4-2 求 $\gcd(1769, 550)$ 时推广 Euclid 算法的运行结果

循环次数	Q	X_1	X_2	X_3	Y_1	Y_2	Y_3
初值	—	1	0	1769	0	1	550
1	3	0	1	550	1	-3	119
2	4	1	-3	119	-4	13	74
3	1	-4	13	74	5	-16	45
4	1	5	-16	45	-9	29	29
5	1	-9	29	29	14	-45	16
6	1	14	-45	16	-23	74	13
7	1	-23	74	13	37	-119	3
8	4	37	-119	3	-171	350	1

所以 $\gcd(1769, 550)=1, 550^{-1} \bmod 1769=350$ 。

4.1.6 中国剩余定理

中国剩余定理是数论中最有用的一个工具, 定理说如果已知某个数关于一些两两互素的数的同余类集, 就可重构这个数。

例如: Z_{10} 中每个数都可从这个数关于 2 和 5 (10 的两个互素的因子) 的同余类重构。比如已知 x 关于 2 和 5 的同余类分别是 $[0]$ 和 $[3]$, 即 $x \bmod 2 \equiv 0, x \bmod 5 \equiv 3$ 。可知是偶数且被 5 除后余数是 3, 所以可得 8 是满足这一关系的惟一的 x 。

定理 4-5 (中国剩余定理) 设 m_1, m_2, \dots, m_k 是两两互素的正整数, $M = \prod_{i=1}^k m_i$, 则一

次同余方程组

$$\begin{cases} a_1 \pmod{m_1} \equiv x \\ a_2 \pmod{m_2} \equiv x \\ \dots \\ a_k \pmod{m_k} \equiv x \end{cases}$$

对模 M 有惟一解:

$$x \equiv \left(\frac{M}{m_1} e_1 a_1 + \frac{M}{m_2} e_2 a_2 + \dots + \frac{M}{m_k} e_k a_k \right) \pmod{M}$$

其中 e_i 满足 $\frac{M}{m_i} e_i \equiv 1 \pmod{m_i} (i=1, 2, \dots, k)$ 。

证明: 设 $M_i = M/m_i = \prod_{\substack{j=1 \\ j \neq i}}^k m_j, i=1, 2, \dots, k$, 由 M_i 的定义得 M_i 与 m_i 是互素的, 可

知 M_i 在模 m_i 下有惟一的乘法逆元, 即满足 $\frac{M}{m_i} e_i \equiv 1 \pmod{m_i}$ 的 e_i 是惟一的。

下面证明对 $\forall i \in \{1, 2, \dots, k\}$, 上述 x 满足 $a_i \pmod{m_i} \equiv x$ 。注意到当 $j \neq i$ 时, $m_i | M_j$, 即 $M_j \equiv 0 \pmod{m_i}$ 。所以

$$\begin{aligned} & (M_j \times e_j \pmod{m_j}) \pmod{m_i} \\ & \equiv ((M_j \pmod{m_i}) \times ((e_j \pmod{m_j}) \pmod{m_i})) \pmod{m_i} \\ & \equiv 0 \end{aligned}$$

而 $(M_i \times (e_i \pmod{m_i})) \pmod{m_i} \equiv (M_i \times e_i) \pmod{m_i} \equiv 1$

所以 $x \pmod{m_i} \equiv a_i$, 即 $a_i \pmod{m_i} \equiv x$

下面证明方程组的解是惟一的。设 x' 是方程组的另一解, 即

$$x' \equiv a_i \pmod{m_i} (i=1, 2, \dots, k)$$

由 $x \equiv a_i \pmod{m_i}$ 得 $x' - x \equiv 0 \pmod{m_i}$, 即 $m_i | (x' - x)$ 。再根据 m_i 两两互素, 有 $M | (x' - x)$, 即 $x' - x \equiv 0 \pmod{M}$, 所以 $x' \pmod{M} = x \pmod{M}$ 。 (证毕)

中国剩余定理提供了一个非常有用的特性, 即在模 M 下可将非常大的数 x 由一组小数 (a_1, a_2, \dots, a_k) 表达。

【例 4-4】 由以下方程组求 x 。

$$\begin{cases} x \equiv 1 \pmod{2} \\ x \equiv 2 \pmod{3} \\ x \equiv 3 \pmod{5} \\ x \equiv 5 \pmod{7} \end{cases}$$

解: $M = 2 \cdot 3 \cdot 5 \cdot 7 = 210$, $M_1 = 105$, $M_2 = 70$, $M_3 = 42$, $M_4 = 30$, 易求 $e_1 \equiv M_1^{-1} \bmod 2 \equiv 1$, $e_2 \equiv M_2^{-1} \bmod 3 \equiv 1$, $e_3 \equiv M_3^{-1} \bmod 5 \equiv 3$, $e_4 \equiv M_4^{-1} \bmod 7 \equiv 4$, 所以 $x \bmod 210 \equiv (105 \times 1 \times 1 + 70 \times 1 \times 2 + 42 \times 3 \times 3 + 30 \times 4 \times 5) \bmod 210 \equiv 173$, 或写成 $x \equiv 173 \bmod 210$ 。

【例 4-5】 将 $973 \bmod 1813$ 由模数分别为 37 和 49 的两个数表示。

解: 取 $x = 973$, $M = 1813$, $m_1 = 37$, $m_2 = 49$ 。

由 $a_1 \equiv 973 \bmod m_1 \equiv 11$, $a_2 \equiv 973 \bmod m_2 \equiv 42$ 得 x 在模 37 和模 49 下的表达为 $(11, 42)$ 。

4.1.7 离散对数

1. 求模下的整数幂

Euler 定理指出如果 $\gcd(a, n) = 1$, 则 $a^{\varphi(n)} \equiv 1 \bmod n$ 。现在考虑如下的一般形式:

$$a^n \equiv 1 \bmod n$$

如果 a 与 n 互素, 则至少有一整数 m (比如 $m = \varphi(n)$) 满足这一方程。称满足方程的最小正整数 m 为模 n 下 a 的阶。

例如: $a = 7, n = 19$, 则易求出 $7^1 \equiv 7 \bmod 19$, $7^2 \equiv 11 \bmod 19$, $7^3 \equiv 1 \bmod 19$, 即 7 在模 19 下的阶为 3。

由于 $7^{3+i} = 7^3 \cdot 7^i \equiv 7^i \bmod 19$, 所以 $7^4 \equiv 7 \bmod 19$, $7^5 \equiv 7^2 \bmod 19$, ..., 即从 $7^4 \bmod 19$ 开始所求的幂出现循环, 循环周期为 3, 即循环周期等于元素的阶。

定理 4-6 设 a 的阶为 m , 则 $a^k \equiv 1 \bmod n$ 的充要条件是 k 为 m 的倍数。

证明: 设存在整数 q , 使得 $k = qm$, 则 $a^k \equiv (a^m)^q \equiv 1 \bmod n$ 。

反之, 假定 $a^k \equiv 1 \bmod n$, 令 $k = qm + r$, 其中 $0 < r \leq m-1$, 那么

$$a^k \equiv (a^m)^q a^r \equiv a^r \equiv 1 \pmod{n}$$

与 m 是阶矛盾。

(证毕)

推论: a 的阶 m 整除 $\varphi(n)$ 。

如果 a 的阶 m 等于 $\varphi(n)$, 则称 a 为 n 的本原根。如果 a 是 n 的本原根, 则

$$a, a^2, \dots, a^{\varphi(n)}$$

在 $\bmod n$ 下互不相同且都与 n 互素。

特别地, 如果 a 是素数 p 的本原根, 则

$$a, a^2, \dots, a^{p-1}$$

在 $\bmod p$ 下都不相同。

例如: $n = 9$, 则 $\varphi(n) = 6$, 考虑 2 在 $\bmod 9$ 下的幂 $2^1 \bmod 9 \equiv 2$, $2^2 \bmod 9 \equiv 4$, $2^3 \bmod 9 \equiv 8$, $2^4 \bmod 9 \equiv 7$, $2^5 \bmod 9 \equiv 5$, $2^6 \bmod 9 \equiv 1$ 。即 2 的阶为 $\varphi(9)$, 所以 2 为 9 的本

原根。

例如： $n=19, a=3$ 在 mod 19 下的幂分别为

$$3, 9, 8, 5, 15, 7, 2, 6, 18, 16, 10, 11, 14, 4, 12, 17, 13, 1.$$

即 3 的阶为 $18=\varphi(19)$ ，所以 3 为 19 的本原根。

本原根不惟一。可验证除 3 外，19 的本原根还有 2, 10, 13, 14, 15。

注意并非所有的整数都有本原根，只有以下形式的整数才有本原根：

$$2, 4, p^a, 2p^a$$

其中 p 为奇素数。

2. 指标

首先回忆一下一般对数的概念，指数函数 $y=a^x (a>0, a\neq 1)$ 的逆函数称为以 a 为底 x 的对数，记为 $y=\log_a x$ 。对数函数有以下性质：

$$\log_a 1 = 0, \log_a a = 1, \log_a xy = \log_a x + \log_a y, \log_a x^y = y \log_a x$$

在模运算中也有类似的函数。设 p 是一素数， a 是 p 的本原根，则 a, a^2, \dots, a^{p-1} 产生出 1 到 $p-1$ 之间的所有值，且每一值只出现一次。因此对任意 $b \in \{1, \dots, p-1\}$ ，都存在惟一的 $i (1 \leq i \leq p-1)$ ，使得 $b \equiv a^i \pmod p$ 。称 i 为模 p 下以 a 为底 b 的指标，记为 $i = \text{ind}_{a,p}(b)$ 。指标有以下性质：

$$\textcircled{1} \text{ ind}_{a,p}(1) = 0.$$

$$\textcircled{2} \text{ ind}_{a,p}(a) = 1.$$

分别由以下关系得出： $a^0 \pmod p = 1 \pmod p = 1, a^1 \pmod p = a$ 。

以上假定模数 p 是素数，对于非素数也有类似的结论。

【例 4-6】 设 $p=9$ ，则 $\varphi(p)=6, a=2$ 是 p 的一个本原根， a 的不同的幂为(模 9 下)

$$2^0 \equiv 1, 2^1 \equiv 2, 2^2 \equiv 4, 2^3 \equiv 8, 2^4 \equiv 7, 2^5 \equiv 5, 2^6 \equiv 1$$

由此可得 a 的指数表如表 4-3(a)所示。

表 4-3 指数和指标举例

(a) 模 9 下 2 的指数表

指标	0	1	2	3	4	5
指数	1	2	4	8	7	5

(b) 与 9 互素的数的指标

数	1	2	4	5	7	8
指标	0	1	2	5	4	3

重新排列表 4-3(a)，可求每一与 9 互素的数的指标如表 4-3(b)所示。

在讨论指标的另两个性质时，需要利用如下结论：

若 $a^z \equiv a^q \pmod p$ ，其中 a 和 p 互素，则有 $z \equiv q \pmod{\varphi(p)}$ 。

证明: 因 a 和 p 互素, 所以 a 在模 p 下存在逆元 a^{-1} , 在 $a^x \equiv a^q \pmod{p}$ 两边同乘以 $(a^{-1})^q$, 得 $a^{x-q} \equiv 1 \pmod{p}$. 由 Euler 定理 $a^{\varphi(p)} \equiv 1 \pmod{p}$ 知存在一整数 k , 使得 $x-q \equiv k\varphi(p)$, 所以 $x \equiv q \pmod{\varphi(p)}$.

由上述结论可得指标的以下两个性质:

$$\textcircled{3} \text{ } ind_{a,p}(xy) = [ind_{a,p}(x) + ind_{a,p}(y)] \pmod{\varphi(p)}.$$

$$\textcircled{4} \text{ } ind_{a,p}(y^r) = [r \times ind_{a,p}(y)] \pmod{\varphi(p)}.$$

性质③的证明: 设 $x \equiv a^{ind_{a,p}(x)} \pmod{p}$, $y \equiv a^{ind_{a,p}(y)} \pmod{p}$, $xy \equiv a^{ind_{a,p}(xy)} \pmod{p}$,

由模运算的性质得:

$$a^{ind_{a,p}(xy)} \pmod{p} = (a^{ind_{a,p}(x)} \pmod{p})(a^{ind_{a,p}(y)} \pmod{p}) = (a^{ind_{a,p}(x) + ind_{a,p}(y)}) \pmod{p}$$

所以 $ind_{a,p}(xy) = [ind_{a,p}(x) + ind_{a,p}(y)] \pmod{\varphi(p)}$ (证毕)

性质④是性质③的推广。

从指标的以上性质可见, 指标与对数的概念极为相似。

3. 离散对数

设 p 是素数, a 是 p 的本原根, 即 a^1, a^2, \dots, a^{p-1} 在模 p 下产生 1 到 $p-1$ 的所有值, 所以对 $\forall b \in \{1, \dots, p-1\}$, 有惟一的 $i \in \{1, \dots, p-1\}$ 使得 $b \equiv a^i \pmod{p}$. 称 i 为模 p 下以 a 为底 b 的离散对数, 记为 $i \equiv \log_a b \pmod{p}$.

当 a, p, i 已知时, 用 4.3.2 节将介绍的快速指数算法可比较容易地求出 b , 但如果已知 a, b 和 p , 求 i 则非常困难。目前已知的最快的求离散对数算法其时间复杂度为:

$$O\left(\exp\left((\ln p)^{\frac{1}{2}} \ln(\ln p)\right)^{\frac{2}{3}}\right)$$

所以当 p 很大时, 该算法也是不可行的。

4.1.8 平方剩余

设 p 是一素数, a 小于 p , 称 a 是 p 的平方剩余, 如果方程

$$x^2 \equiv a \pmod{p}$$

有解。否则称为非平方剩余。

例如: $x^2 \equiv 1 \pmod{7}$ 有解 $x=1, x=6$;

$x^2 \equiv 2 \pmod{7}$ 有解 $x=3, x=4$;

$x^2 \equiv 3 \pmod{7}$ 无解;

$x^2 \equiv 4 \pmod{7}$ 有解 $x=2, x=5$;

$x^2 \equiv 5 \pmod{7}$ 无解;

$x^2 \equiv 6 \pmod{7}$ 无解。

可见共有 3 个数(1, 2, 4)是模 7 的平方剩余, 且每个平方剩余都有两个平方根(即例

中的 x 。

容易证明,模 p 的平方剩余的个数为 $(p-1)/2$,且与模 p 的非平方剩余的个数相等。如果 a 是模 p 的一个平方剩余,那么 a 恰有两个平方根,一个在 0 到 $(p-1)/2$ 之间,另一个在 $(p-1)/2$ 到 $(p-1)$ 之间,且这两个平方根中的一个也是模 p 的平方剩余。

定义 4-1 设 p 是素数, a 是一整数,符号 $\left(\frac{a}{p}\right)$ 的定义如下:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{如果 } a \text{ 被 } p \text{ 整除} \\ 1 & \text{如果 } a \text{ 是模 } p \text{ 的平方剩余} \\ -1 & \text{如果 } a \text{ 是模 } p \text{ 的非平方剩余} \end{cases}$$

称符号 $\left(\frac{a}{p}\right)$ 为 Legendre 符号。

$$\text{例如: } \left(\frac{1}{7}\right) = \left(\frac{2}{7}\right) = \left(\frac{4}{7}\right) = 1,$$

$$\left(\frac{3}{7}\right) = \left(\frac{5}{7}\right) = \left(\frac{6}{7}\right) = -1.$$

计算 $\left(\frac{a}{p}\right)$ 有一个简单公式: $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}$ 。

例如: $p=23, a=5, a^{(p-1)/2} \pmod{p} \equiv 5^{11} \pmod{p} = -1$, 所以 5 不是模 23 的平方剩余。

Legendre 符号有以下性质:

定理 4-7 设 p 是奇素数, a 和 b 都不能被 p 除尽, 则

$$\textcircled{1} \text{ 若 } a \equiv b \pmod{p}, \text{ 则 } \left(\frac{a}{p}\right) = \left(\frac{b}{p}\right).$$

$$\textcircled{2} \left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \left(\frac{b}{p}\right).$$

$$\textcircled{3} \left(\frac{a^2}{p}\right) = 1.$$

$$\textcircled{4} \left(\frac{a+p}{p}\right) = \left(\frac{a}{p}\right).$$

证明从略。

以下定义的 Jacobi 符号是 Legendre 符号的推广。

定义 4-2 设 n 是正整数, 且 $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$, 定义 Jacobi 符号为

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{a_1} \left(\frac{a}{p_2}\right)^{a_2} \cdots \left(\frac{a}{p_k}\right)^{a_k}$$

其中右端的符号是 Legendre 符号。

当 n 为素数时, Jacobi 符号就是 Legendre 符号。

Jacobi 符号有以下性质:

定理 4-8 设 n 是正合数, a 和 b 是与 n 互素的整数, 则

$$\textcircled{1} \text{ 若 } a \equiv b \pmod{n}, \text{ 则 } \left(\frac{a}{n}\right) = \left(\frac{b}{n}\right),$$

$$\textcircled{2} \left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \left(\frac{b}{n}\right),$$

$$\textcircled{3} \left(\frac{ab^2}{n}\right) = \left(\frac{a}{n}\right),$$

$$\textcircled{4} \left(\frac{a+n}{n}\right) = \left(\frac{a}{n}\right).$$

对一些特殊的 a , Jacobi 符号可如下计算:

$$\left(\frac{1}{n}\right) = 1, \left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}, \left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$$

定理 4-9 (Jacobi 符号的互反律) 设 m, n 均为大于 2 的奇数, 则

$$\left(\frac{m}{n}\right) = (-1)^{(m-1)(n-1)/4} \left(\frac{n}{m}\right)$$

若 $m \equiv n \equiv 3 \pmod{4}$, 则 $\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right)$; 否则 $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)$.

以上性质表明: 为了计算 Jacobi 符号 (包括 Legendre 符号作为它的特殊情形), 并不要求素因子分解式. 例如 105 虽然不是素数, 在计算 Legendre 符号 $\left(\frac{105}{317}\right)$ 时, 可以先把它看作 Jacobi 符号来计算, 由上述两个定理得

$$\left(\frac{105}{317}\right) = \left(\frac{317}{105}\right) = \left(\frac{2}{105}\right) = 1$$

一般在计算 $\left(\frac{m}{n}\right)$ 时, 如果有必要, 可用 $m \pmod{n}$ 代替 m , 而互反律用以减小 $\left(\frac{m}{n}\right)$ 的分母.

可见, 引入 Jacobi 符号对计算 Legendre 符号是十分方便的, 但应强调指出 Jacobi 符号和 Legendre 符号的本质差别是: Jacobi 符号 $\left(\frac{a}{n}\right)$ 不表示方程 $x^2 \equiv a \pmod{n}$ 是否有解. 比如 $n = p_1 p_2$, a 关于 p_1 和 p_2 都不是平方剩余, 即 $x^2 \equiv a \pmod{p_1}$ 和 $x^2 \equiv a \pmod{p_2}$ 都无解, 由中国剩余定理知 $x^2 \equiv a \pmod{n}$ 也无解. 但是, 由于 $\left(\frac{a}{p_1}\right) = \left(\frac{a}{p_2}\right) = -1$, 所以 $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \left(\frac{a}{p_2}\right) = 1$. 即 $x^2 \equiv a \pmod{n}$ 虽无解, 但 Jacobi 符号 $\left(\frac{a}{n}\right)$ 却为 1.

【例 4-7】考虑方程 $x^2 \equiv 2 \pmod{3599}$, 由于 $3599 = 59 \times 61$, 所以方程等价于方程组

$$\begin{cases} x^2 \equiv 2 \pmod{59} \\ x^2 \equiv 2 \pmod{61} \end{cases}$$

由于 $\left(\frac{2}{59}\right) = -1$, 所以方程组无解, 但 Jacobi 符号 $\left(\frac{2}{3599}\right) = (-1)^{(3599^2-1)/8} = 1$ 。

下面考虑公钥密码体制中一个非常重要的问题。

设 n 是两个大素数 p 和 q 的乘积。由上述结论, 1 到 $p-1$ 之间有一半数是模 p 的平方剩余, 另一半是模 p 的非平方剩余, 对 q 也有类似结论。另一方面, a 是模 n 的平方剩余, 当且仅当 a 既是模 p 的平方剩余也是模 q 的平方剩余。所以对满足

$$0 < a < n, \quad \gcd(a, n) = 1$$

的 a , 有一半满足 $\left(\frac{a}{n}\right) = 1$, 另一半满足 $\left(\frac{a}{n}\right) = -1$ 。而在满足 $\left(\frac{a}{n}\right) = 1$ 的 a 中, 有一半满足 $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = 1$, 这些 a 就是模 n 的平方剩余; 另一半满足 $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$, 这些 a 是模 n 的非平方剩余。

设 a 是模 n 的平方剩余, 即存在 x 使得 $x^2 \equiv a \pmod{n}$ 成立, 因 a 既是模 p 的平方剩余, 又是模 q 的平方剩余, 所以存在 y, z , 使得 $(\pm y)^2 \equiv a \pmod{p}, (\pm z)^2 \equiv a \pmod{q}$, 因此

$$x \equiv \pm y \pmod{p}, \quad x \equiv \pm z \pmod{q}$$

由中国剩余定理可求得 $a \pmod{n}$ 的 4 个平方根, 记为 $\pm u \pmod{n}$ 和 $\pm w \pmod{n}$, 且 $u \not\equiv \pm w \pmod{n}$ 。

以上结果表明, 已知 n 的分解 $n = pq$, 且 a 是模 n 的平方剩余, 就可求得 $a \pmod{n}$ 的 4 个平方根。

下面考虑相反的问题, 即已知 $a \pmod{n}$ 的两个不同的平方根 ($u \pmod{n}$ 和 $w \pmod{n}$, 且 $u \not\equiv \pm w \pmod{n}$), 就可分解 n 。事实上由 $u^2 \equiv w^2 \pmod{n}$ 得 $(u+w)(u-w) \equiv 0 \pmod{n}$, 但 n 不能整除 $u+w$ 也不能整除 $u-w$, 所以必有

$$p \mid (u+w), \quad q \mid (u-w)$$

或

$$p \mid (u-w), \quad q \mid (u+w)$$

所以

$$\gcd(n, u+w) = p, \quad \gcd(n, u-w) = q$$

或

$$\gcd(n, u-w) = p, \quad \gcd(n, u+w) = q$$

因此得到了 n 的分解式。

将以上讨论总结为:

定理 4-10 求解方程 $x^2 \equiv a \pmod n$ 与分解 n 是等价的。

第 2 个重要结论是: 当 $p \equiv q \equiv 3 \pmod 4$ 时, $a \pmod n$ 的两个不同的平方根 u 和 w 的 Jacobi 符号有如下关系:

$$\left(\frac{u}{n}\right) = -\left(\frac{w}{n}\right)$$

证明: 由以上讨论知, u, w 满足

$$p \mid (u+w), \quad q \mid (u-w)$$

或

$$p \mid (u-w), \quad q \mid (u+w)$$

即 $u \equiv -w \pmod p, u \equiv w \pmod q$ 或 $u \equiv w \pmod p, u \equiv -w \pmod q$ 。

若为第一种情况,

$$\left(\frac{u}{n}\right) = \left(\frac{u}{p}\right)\left(\frac{u}{q}\right) = \left(\frac{-w}{p}\right)\left(\frac{w}{q}\right) = \left(\frac{-1}{p}\right)\left(\frac{w}{p}\right)\left(\frac{w}{q}\right) = -\left(\frac{w}{p}\right)\left(\frac{w}{q}\right) = -\left(\frac{w}{n}\right)。$$

(证毕)

同理可证第二种情况。

4.2 公钥密码体制的基本概念

在公钥密码体制以前的整个密码学史中, 所有的密码算法, 包括原始手工计算的、由机械设备实现的以及由计算机实现的, 都是基于代换和置换这两个基本工具。而公钥密码体制则为密码学的发展提供了新的理论和技术基础, 一方面公钥密码算法的基本工具不再是代换和置换, 而是数学函数; 另一方面公钥密码算法是以非对称的形式使用两个密钥, 两个密钥的使用对保密性、密钥分配、认证等都有着深刻的意义。可以说公钥密码体制的出现在密码学史上是迄今为止最大的而且是惟一真正的革命。

公钥密码体制的概念是在解决单钥密码体制中最难解决的两个问题时提出的, 这两个问题是密钥分配和数字签字。

单钥密码体制在进行密钥分配时(见第 5 章), 要求通信双方或者已经有一个共享的密钥, 或者可借助于一个密钥分配中心。对第 1 个要求, 常常可用人工方式传送双方最初共享的密钥, 这种方法成本很高, 而且还完全依赖于信使的可靠性。第 2 个要求则完全依赖于密钥分配中心的可靠性。

第 2 个问题数字签字考虑的是如何为数字化的消息或文件提供一种类似于为书面文

件手书签字的方法。

1976年 W. Diffie 和 M. Hellman 对解决上述两个问题有了突破,从而提出了公钥密码体制。

4.2.1 公钥密码体制的原理

公钥密码算法的最大特点是采用两个相关密钥将加密和解密能力分开,其中一个密钥是公开的,称为公开密钥,简称公开钥,用于加密;另一个密钥是为用户专用,因而是保密的,称为秘密密钥,简称秘密钥,用于解密。因此公钥密码体制也称为双钥密码体制。算法有以下重要特性:已知密码算法和加密密钥,求解密密钥在计算上是不可行的。

图 4-1 是公钥体制加密的框图,加密过程有以下几步:

① 要求接收消息的端系统,产生一对用来加密和解密的密钥,如图中的接收者 B,产生一对密钥 PK_B, SK_B ,其中 PK_B 是公开钥, SK_B 是秘密钥。

② 端系统 B 将加密密钥(如图中的 PK_B)予以公开。另一密钥则被保密(图中的 SK_B)。

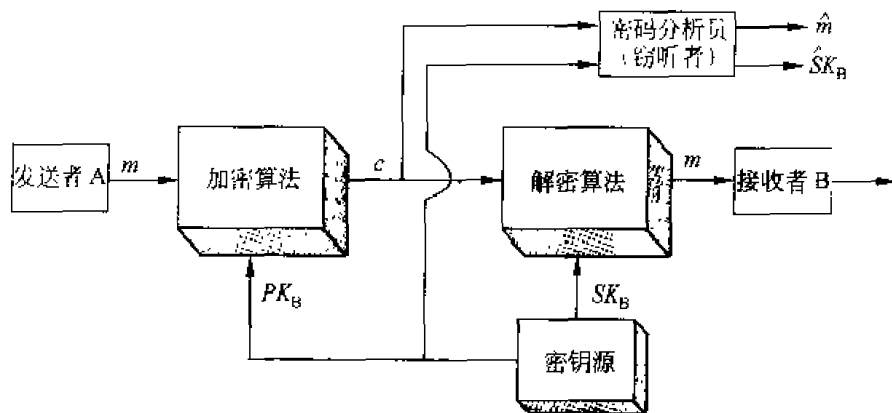


图 4-1 公钥体制加密的框图

③ A 要想向 B 发送消息 m , 则使用 B 的公开钥加密 m , 表示为 $c = E_{PK_B}[m]$, 其中 c 是密文, E 是加密算法。

④ B 收到密文 c 后, 用自己的秘密钥 SK_B 解密, 表示为 $m = D_{SK_B}[c]$, 其中 D 是解密算法。

因为只有 B 知道 SK_B , 所以其他人都无法对 c 解密。

公钥加密算法不仅能用于加、解密, 还能用于对发方 A 发送的消息 m 提供认证, 如图 4-2 所示。用户 A 用自己的秘密钥 SK_A 对 m 加密, 表示为

$$c = E_{SK_A}[m]$$

将 c 发往 B。B 用 A 的公开钥 PK_A 对 c 解密, 表示为

$$m = D_{PK_A}[c]$$

因为从 m 得到 c 是经过 A 的秘密钥 SK_A 加密, 只有 A 才能做到。因此 c 可当做 A 对 m 的数字签字。另一方面, 任何人只要得不到 A 的秘密钥 SK_A 就不能篡改 m , 所以以上过程获得了对消息来源和消息完整性的认证。

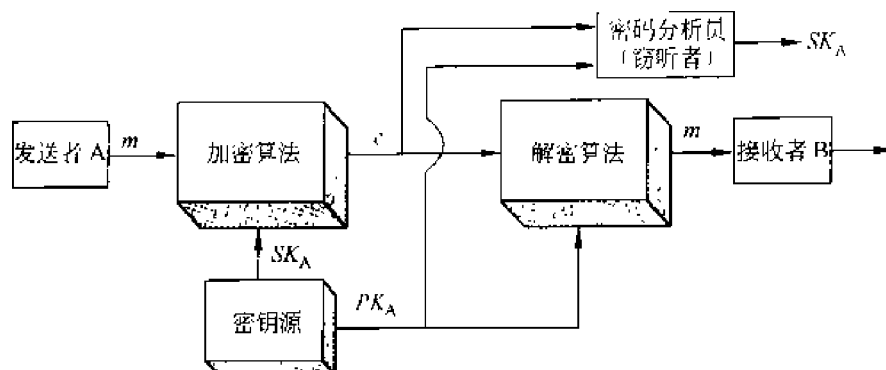


图 4-2 公钥密码体制认证框图

在实际应用中, 特别是用户数目很多时, 以上认证方法需要很大的存储空间, 因为每个文件都必须以明文形式存储以方便实际使用, 同时还必须存储每个文件被加密后的密文形式即数字签字, 以便在有争议时用来认证文件的来源和内容。改进的方法是减小文件的数字签字的大小, 即先将文件经过一个函数压缩成长度较小的比特串, 得到的比特串称为认证符。认证符具有这样一个性质: 如果保持认证符的值不变而修改文件这在计算上是不可行的。用发送者的秘密钥对认证符加密, 加密后的结果为原文件的数字签字。这一内容在第 7 章中会详细介绍。

以上认证过程中, 由于消息是由用户自己的秘密钥加密的, 所以消息不能被他人篡改, 但却能被他人窃听。这是因为任何人都能用用户的公开钥对消息解密。为了同时提供认证功能和保密性, 可使用双重加、解密。如图 4-3 所示。

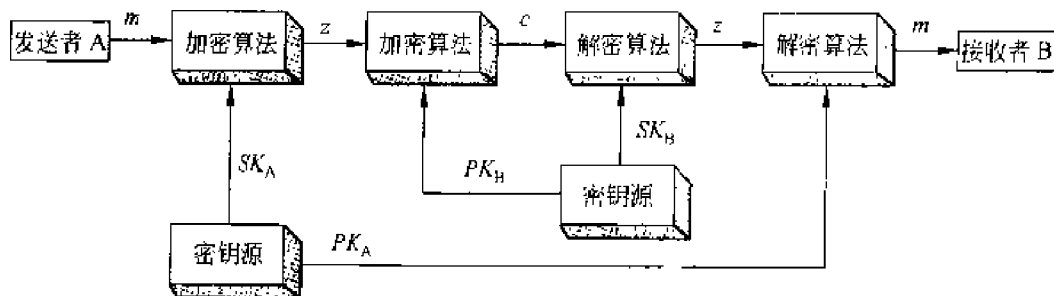


图 4-3 公钥密码体制的认证、保密框图

发方首先用自己的秘密钥 SK_A 对消息 m 加密, 用于提供数字签字。再用收方的公

开钥 PK_B 第2次加密,表示为

$$c = E_{PK_B}[E_{SK_A}[m]]$$

解密过程为

$$m = D_{PK_A}[D_{SK_B}[c]]$$

即收方先用自己的秘密钥,再用发方的公开钥对收到的密文两次解密。

4.2.2 公钥密码算法应满足的要求

公钥密码算法应满足以下要求:

- ① 接收方 B 产生密钥对(公开钥 PK_B 和秘密钥 SK_B)在计算上是容易的。
- ② 发方 A 用收方的公开钥对消息 m 加密以产生密文 c ,即

$$c = E_{PK_B}[m]$$

在计算上是容易的。

- ③ 收方 B 用自己的秘密钥对 c 解密,即

$$m = D_{SK_B}[c]$$

在计算上是容易的。

- ④ 敌手由 B 的公开钥 PK_B 求秘密钥 SK_B 在计算上是不可行的。
- ⑤ 敌手由密文 c 和 B 的公开钥 PK_B 恢复明文 m 在计算上是不可行的。
- ⑥ 加、解密次序可换,即

$$E_{PK_B}[D_{SK_B}(m)] = D_{SK_B}[E_{PK_B}(m)]$$

其中最后一条虽然非常有用,但不是对所有的算法都作要求。

以上要求的本质之处在于要求一个陷门单向函数。单向函数是两个集合 X, Y 之间的一个映射,使得 Y 中每一元素 y 都有惟一的一个原像 $x \in X$,且由 x 易于计算它的像 y ,由 y 计算它的原像 x 是不可行的。这里所说的易于计算是指函数值能在其输入长度的多项式时间内求出,即如果输入长 n 比特,则求函数值的计算时间是 n^a 的某个倍数,其中 a 是一固定的常数。这时称求函数值的算法属于多项式类 P ,否则就是不可行的。例如,函数的输入是 n 比特,如果求函数值所用的时间是 2^n 的某个倍数,则认为求函数值是不可行的。注意这里的易于计算和不可行两个概念与计算复杂性理论中复杂度的概念极为相似,然而又存在着本质的区别。在复杂性理论中,算法的复杂度是以算法在最坏情况或平均情况时的复杂度来度量的。而在此所说的两个概念是指算法在几乎所有情况下的情形。称一个函数是陷门单向函数,是指该函数是易于计算的,但求它的逆是不可行的,除非再已知某些附加信息。当附加信息给定后,求逆可在多项式时间完成。总结为:陷门单向函数是一族可逆函数 f_k ,满足

- ① $Y = f_k(X)$ 易于计算(当 k 和 X 已知时)。

② $X = f_k^{-1}(Y)$ 易于计算(当 k 和 Y 已知时)。

③ $X = f_k^{-1}(Y)$ 计算上是不可行的(当 Y 已知但 k 未知时)。

因此,研究公钥密码算法就是要找出合适的陷门单向函数。

4.2.3 对公钥密码体制的攻击

和单钥密码体制一样,如果密钥太短,公钥密码体制也易受到穷搜索攻击。因此密钥必须足够长才能抗击穷搜索攻击。然而又由于公钥密码体制所使用的可逆函数的计算复杂性与密钥长度常常不是呈线性关系,而是增大得更快。所以密钥长度太大又会使得加解密运算太慢而不实用。因此公钥密码体制目前主要用于密钥管理和数字签字。

对公钥密码算法的第 2 种攻击法是寻找从公开钥计算秘密钥的方法。目前为止,对常用公钥算法还都未能证明这种攻击是不可行的。

还有一种仅适用于对公钥密码算法的攻击法,称为可能字攻击。例如对 56 比特的 DES 密钥用公钥密码算法加密后发送,敌手用算法的公开钥对所有可能的密钥加密后与截获的密文相比较。如果一样,则相应的明文即 DES 密钥就被找出。因此不管公钥算法的密钥多长,这种攻击的本质是对 56 比特 DES 密钥的穷搜索攻击。抵抗方法是在欲发送的明文消息后添加一些随机比特。

4.3 RSA 算法

RSA 算法是 1978 年由 R. Rivest, A. Shamir 和 L. Adleman 提出的一种用数论构造的、也是迄今为止理论上最为成熟完善的公钥密码体制,该体制已得到广泛的应用。

4.3.1 算法描述

1. 密钥的产生

- ① 选两个保密的大素数 p 和 q 。
- ② 计算 $n = p \times q$, $\varphi(n) = (p-1)(q-1)$, 其中 $\varphi(n)$ 是 n 的欧拉函数值。
- ③ 选一整数 e , 满足 $1 < e < \varphi(n)$, 且 $\gcd(\varphi(n), e) = 1$ 。
- ④ 计算 d , 满足 $d \cdot e \equiv 1 \pmod{\varphi(n)}$, 即 d 是 e 在模 $\varphi(n)$ 下的乘法逆元, 因 e 与 $\varphi(n)$ 互素, 由模运算可知, 它的乘法逆元一定存在。
- ⑤ 以 $\{e, n\}$ 为公开钥, $\{d, n\}$ 为秘密钥。

2. 加密

加密时首先将明文比特串分组, 使得每个分组对应的十进制数小于 n , 即分组长度小

于 $\log_2 n$ 。然后对每个明文分组 m , 作加密运算:

$$c \equiv m^e \pmod{n}$$

3. 解密

对密文分组的解密运算为:

$$m \equiv c^d \pmod{n}$$

下面证明 RSA 算法中解密过程的正确性。

证明: 由加密过程知 $c \equiv m^e \pmod{n}$, 所以

$$c^d \pmod{n} \equiv m^{ed} \pmod{n} \equiv m^{1 \pmod{\varphi(n)}} \pmod{n} \equiv m^{k\varphi(n)+1} \pmod{n}$$

下面分两种情况:

① m 与 n 互素, 则由 Euler 定理得

$$m^{\varphi(n)} \equiv 1 \pmod{n}, \quad m^{k\varphi(n)} \equiv 1 \pmod{n}, \quad m^{k\varphi(n)+1} \equiv m \pmod{n}$$

即 $c^d \pmod{n} \equiv m$ 。

② $\gcd(m, n) \neq 1$, 先看 $\gcd(m, n) = 1$ 的含义, 由于 $n = pq$, 所以 $\gcd(m, n) = 1$ 意味着 m 不是 p 的倍数也不是 q 的倍数。因此 $\gcd(m, n) \neq 1$ 意味着 m 是 p 的倍数或 q 的倍数, 不妨设 $m = cp$, 其中 c 为一正整数。此时必有 $\gcd(m, q) = 1$, 否则 m 也是 q 的倍数, 从而 pq 的倍数, 与 $m < n = pq$ 矛盾。

由 $\gcd(m, q) = 1$ 及 Euler 定理得 $m^{\varphi(q)} \equiv 1 \pmod{q}$, 所以

$$m^{k\varphi(q)} \equiv 1 \pmod{q}, \quad [m^{k\varphi(q)}]^{\varphi(p)} \equiv 1 \pmod{q}, \quad m^{k\varphi(n)} \equiv 1 \pmod{q}$$

因此存在一整数 r , 使得 $m^{k\varphi(n)} = 1 + rq$, 两边同乘以 $m = cp$ 得

$$m^{k\varphi(n)+1} = m + rc pq = m + rcn$$

即 $m^{k\varphi(n)+1} \equiv m \pmod{n}$, 所以 $c^d \pmod{n} \equiv m$ 。 (证毕)

【例 4-8】 选 $p=7, q=17$ 。求 $n=p \times q=119, \varphi(n)=(p-1)(q-1)=96$ 。取 $e=5$, 满足 $1 < e < \varphi(n)$, 且 $\gcd(\varphi(n), e) = 1$ 。确定满足 $d \cdot e \equiv 1 \pmod{96}$ 且小于 96 的 d , 因为 $77 \times 5 = 385 = 4 \times 96 + 1$, 所以 d 为 77, 因此公开钥为 $\{5, 119\}$, 秘密钥为 $\{77, 119\}$ 。设明文 $m=19$, 则由加密过程得密文为

$$c \equiv 19^5 \pmod{119} \equiv 2476099 \pmod{119} \equiv 66$$

解密为

$$66^{77} \pmod{119} \equiv 19$$

4.3.2 RSA 算法中的计算问题

1. RSA 的加密与解密过程

RSA 的加密、解密过程都为求一个整数的整数次幂, 再取模。如果按其含义直接计算, 则中间结果非常大, 有可能超出计算机所允许的整数取值范围。如上例中解密运算

$66^{77} \bmod 119$, 先求 66^{77} 再取模, 则中间结果就已远远超出了计算机允许的整数取值范围。而用模运算的性质:

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

就可减小中间结果。

再者, 考虑如何提高加、解密运算中指数运算的有效性。例如求 x^{16} , 直接计算的话需做 15 次乘法。然而如果重复对每个部分结果做平方运算即求 x, x^2, x^4, x^8, x^{16} 则只需 4 次乘法。

求 a^m 可如下进行, 其中 a, m 是正整数:

将 m 表示为二进制形式 $b_k b_{k-1} \cdots b_0$, 即

$$m = b_k 2^k + b_{k-1} 2^{k-1} + \cdots + b_1 2 + b_0$$

因此

$$a^m = (((\cdots ((a^{b_k})^2 a^{b_{k-1}})^2 a^{b_{k-2}})^2 \cdots a^{b_1})^2 a^{b_0})$$

例如: $19 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$, 所以

$$a^{19} = (((((a^1)^2 a^0)^2 a^1)^2 a^1)^2 a^1)$$

从而可得以下快速指数算法:

```

c=0; d=1;
for i=k downto 0 do {
    c=2×c;
    d=(d×d) mod n;
    if bi=1 then {
        c=c+1;
        d=(d×a) mod n
    }
}
return d.
    
```

其中 d 是中间结果, d 的终值即为所求结果。 c 在这里的作用是表示指数的部分结果, 其终值即为指数 m , c 对计算结果无任何贡献, 算法中完全可将之去掉。

【例 4-9】 求 $7^{560} \bmod 561$ 。

将 560 表示为 1000110000, 算法的中间结果如表 4-4 所示:

表 4-4 快速指数算法示例

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	0	1	2	4	8	17	35	70	140	280
d	1	7	49	157	526	160	241	298	166	67

所以 $7^{560} \bmod 561 = 1$ 。

2. RSA 密钥的产生

产生密钥时,需要考虑两个大素数 p, q 的选取,以及 e 的选取和 d 的计算。

因为 $n=pq$ 在体制中是公开的,因此为了防止敌手通过穷搜索发现 p, q , 这两个素数应是在一个足够大的整数集中选取的大数。如果选取 p 和 q 为 10^{100} 左右的大素数,那么 n 的阶为 10^{200} , 每个明文分组可以含有 664 位 ($10^{200} \approx 2^{664}$), 即 83 个 8 比特字节, 这比 DES 的数据分组(8 个 8 比特字节)大得多, 这时就能看出 RSA 算法的优越性了。因此如何有效地寻找大素数是第一个需要解决的问题。

寻找大素数时一般是先随机选取一个大的奇数(例如用伪随机数产生器), 然后用素性检验算法检验这一奇数是否为素数, 如果不是则选取另一大奇数, 重复这一过程, 直到找到素数为止。素性检验算法通常都是概率性的, 但如果算法被多次重复执行, 每次执行时输入不同的参数, 算法的检验结果都认为被检验的数是素数, 那么就可以比较有把握地认为被检验的数是素数。例如 4.1.4 节中介绍的 Miller-Rabin 算法。

可见寻找大素数是一个比较繁琐的工作。然而在 RSA 体制中, 只有在产生新密钥时才需执行这一工作。

p 和 q 决定出后, 下一个需要解决的问题是如何选取满足 $1 < e < \varphi(n)$ 和 $\gcd(\varphi(n), e) = 1$ 的 e , 并计算满足 $d \cdot e \equiv 1 \bmod \varphi(n)$ 的 d 。这一问题可由推广的 Euclid 算法完成。

4.3.3 RSA 的安全性

RSA 的安全性是基于分解大整数的困难性假定, 之所以为假定是因为至今还未能证明分解大整数就是 NP 问题, 也许有尚未发现的多项式时间分解算法。如果 RSA 的模数 n 被成功地分解为 $p \times q$, 则立即获得 $\varphi(n) = (p-1)(q-1)$, 从而能够确定 e 模 $\varphi(n)$ 的乘法逆元 d , 即 $d \equiv e^{-1} \bmod \varphi(n)$, 因此攻击成功。

随着人类计算能力的不断提高, 原来被认为是不可能分解的大数已被成功分解。例如 RSA-129 (即 n 为 129 位十进制数, 大约 428 个比特) 已在网络上通过分布式计算历时 8 个月于 1994 年 4 月被成功分解, RSA-130 已于 1996 年 4 月被成功分解。

对于大整数的威胁除了人类的计算能力外, 还来自分解算法的进一步改进。分解算法过去都采用二次筛法, 如对 RSA-129 的分解。而对 RSA-130 的分解则采用了一个新算法, 称为推广的数域筛法, 该算法在分解 RSA-130 时所做的计算仅比分解 RSA-129 多 10%。将来也可能还有更好的分解算法, 因此在使用 RSA 算法时对其密钥的选取要特别注意其大小。估计在未来一段比较长的时期, 密钥长度介于 1024 比特至 2048 比特之间的 RSA 是安全的。

是否有不通过分解大整数的其他攻击途径? 下面证明由 n 直接确定 $\varphi(n)$ 等价于对 n 的分解。

设 $n = p \times q$ 中, $p > q$, 由 $\varphi(n) = (p-1)(q-1)$, 则有

$$p + q = n - \varphi(n) + 1$$

以及

$$p - q = \sqrt{(p+q)^2 - 4n}$$

由此可见, 由 p, q 确定 $\varphi(n)$ 和由 $\varphi(n)$ 确定 p, q 是等价的。

为保证算法的安全性, 还对 p 和 q 提出以下要求:

(1) $|p - q|$ 要大

由 $\frac{(p+q)^2}{4} - n = \frac{(p+q)^2}{4} - pq = \frac{(p-q)^2}{4}$, 如果 $|p - q|$ 小, 则 $\frac{(p-q)^2}{4}$ 也小, 因此

$\frac{(p+q)^2}{4}$ 稍大于 n , $\frac{p+q}{2}$ 稍大于 \sqrt{n} 。可得 n 的如下分解步骤:

① 顺序检查大于 \sqrt{n} 的每一整数 x , 直到找到一个 x 使得 $x^2 - n$ 是某一整数 (记为 y) 的平方。

② 由 $x^2 - n = y^2$, 得 $n = (x+y)(x-y)$ 。

(2) $p-1$ 和 $q-1$ 都应有大素因子

这是因为 RSA 算法存在着可能的重复加密攻击法。设攻击者截获密文 c , 可如下进行重复加密:

$$c^e \equiv (m^e)^e \equiv m^{e^2} \pmod{n}$$

$$c^{e^2} \equiv (m^e)^{e^2} \equiv m^{e^3} \pmod{n}$$

...

$$c^{e^{t-1}} \equiv (m^e)^{e^{t-1}} \equiv m^{e^t} \pmod{n}$$

$$c^{e^t} \equiv (m^e)^{e^t} \equiv m^{e^{t+1}} \pmod{n}$$

若 $m^{e^{t+1}} \equiv c \pmod{n}$, 即 $(m^e)^{e^t} \equiv c \pmod{n}$, 则有 $m^{e^t} \equiv m \pmod{n}$, 即 $c^{e^{t-1}} \equiv m \pmod{n}$, 所以在上述重复加密的倒数第 2 步就已恢复出明文 m , 这种攻击法只有在 t 较小时才是可行的。为抵抗这种攻击, p, q 的选取应保证使 t 很大。

设 m 在模 n 下阶为 k , 由 $m^e \equiv m \pmod{n}$ 得 $m^{e-1} \equiv 1 \pmod{n}$, 所以 $k | (e-1)$, 即 $e \equiv 1 \pmod{k}$, t 取为满足上式的最小值 (为 e 在模 k 下的阶)。又当 e 与 k 互素时 $t | \varphi(k)$ 。为使 t 大, k 就应大且 $\varphi(k)$ 应有大的素因子。又由 $k | \varphi(n)$, 所以为使 k 大, $p-1$ 和 $q-1$ 都应有大的素因子。

此外, 研究结果表明, 如果 $e < n$ 且 $d < n^{1/4}$, 则 d 能被容易地确定。

4.3.4 对 RSA 的攻击

RSA 存在以下两种攻击,并不是因为算法本身存在缺陷,而是由于参数选择不当造成的。

1. 共模攻击

在实现 RSA 时,为方便起见,可能给每一用户相同的模数 n ,虽然加解密密钥不同,然而这样做是不行的。

设两个用户的公开钥分别为 e_1 和 e_2 ,且 e_1 和 e_2 互素(一般情况都成立),明文消息是 m ,密文分别是

$$c_1 \equiv m^{e_1} \pmod{n}$$

$$c_2 \equiv m^{e_2} \pmod{n}$$

敌手截获 c_1 和 c_2 后,可如下恢复 m 。用推广的 Euclid 算法求出满足

$$re_1 + se_2 = 1$$

的两个整数 r 和 s ,其中一个为负,设为 r 。再次用推广的 Euclid 算法求出 c_1^{-1} ,由此得 $(c_1^{-1})^r c_2^s \equiv m \pmod{n}$ 。

2. 低指数攻击

假定将 RSA 算法同时用于多个用户(为讨论方便,以下假定 3 个),然而每个用户的加密指数(即公开钥)都很小。设 3 个用户的模数分别为 $n_i (i=1, 2, 3)$,当 $i \neq j$ 时, $\gcd(n_i, n_j) = 1$,否则通过 $\gcd(n_i, n_j)$ 有可能得出 n_i 和 n_j 的分解。设明文消息是 m ,密文分别是

$$c_1 \equiv m^3 \pmod{n_1}$$

$$c_2 \equiv m^3 \pmod{n_2}$$

$$c_3 \equiv m^3 \pmod{n_3}$$

由中国剩余定理可求出 $m^3 \pmod{n_1 n_2 n_3}$ 。由于 $m^3 < n_1 n_2 n_3$,可直接由 m^3 开立方根得到 m 。

4.4 背包密码体制

设 $A = (a_1, a_2, \dots, a_n)$ 是由 n 个不同的正整数构成的 n 元组, s 是另一已知的正整数。背包问题就是从 A 中求出所有的 a_i ,使其和等于 s 。其中 A 称为背包向量, s 是背包的容积。

例如, $A = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)$, $s = 3231$ 。由于

$$3231 = 129 + 473 + 903 + 561 + 1165$$

所以从 A 中找出的满足要求的数有 129、473、903、561、1165。

原则上讲,通过检查 A 的所有子集,总可找出问题的解(如果有解的话)。本例 A 的子集共有 $2^{10} = 1024$ 个(包括空集)。然而如果 A 中元素个数 n 很大,子集个数 2^n 将非常大。如 A 中有 300 个元素, A 的子集有 2^{300} 。寻找满足要求的 A 的子集没有比穷搜索更好的算法,因此背包问题是 NPC 问题。

由背包问题构造公钥密码体制同样是要构造一个单向函数 f , 将 $x (1 \leq x \leq 2^n - 1)$ 写成长为 n 的二元表示 $0 \cdots 001, 0 \cdots 010, 0 \cdots 011, \dots, 1 \cdots 111$, $f(x)$ 定义为 A 中所有 a_i 的和, 其中 x 的二元表示的第 i 位为 1, 即

$$f(1) = f(0 \cdots 001) = a_n$$

$$f(2) = f(0 \cdots 010) = a_{n-1}$$

$$f(3) = f(0 \cdots 011) = a_{n-1} + a_n$$

$$\vdots$$

$$f(2^n - 1) = f(1 \cdots 111) = a_1 + a_2 + \cdots + a_n$$

使用向量乘,有 $f(x) = A \cdot B_x$, 其中 B_x 是将 x 的二元表示写成的列向量。

上例中 $f(364) = f(0101101100) = 129 + 473 + 903 + 561 + 1165 = 3231$, 类似地可求出:

$$f(609) = 2942, f(686) = 3584, f(32) = 903, f(46) = 3326,$$

$$f(128) = 215, f(261) = 2817, f(44) = 2629, f(648) = 819.$$

由 f 的定义可见,已知 x 很容易求 $f(x)$, 但已知 $f(x)$ 求 x 就是要解背包问题。当然在实际应用中, n 不能太小,比如说,至少为 200。

用 f 对明文消息 m 加密时,首先将 m 写成二元表示,再将其分成长为 n 的分组(最后一个分组不够长的话,可在后面填充一些 0),然后求每一分组的函数值,以函数值作为密文分组。

例如,明文消息是英文文本,则可将每个字母用其在字母表中的序号表示,再将该序号转换为二进制形式(5 位即可),如表 4-5 所示,其中符号 ' ' 表示空格。

背包向量仍取上例中的 A , 设待加密的明文是 SAUNA AND HEALTH。因为 A 长为 10, 所以应将明文分成长为 10 比特(即两个明文字母)的分组

SA, UN, A ' ', AN, D ' ', HE, AL, TH

相应的二元序列为

1001100001, 1010101110, 0000100000, 0000101110,

0010000000, 0100000101, 0000101100, 1010001000。

表 4-5 英文字母表及字母的二进制表示

字母	序号	二进制	字母	序号	二进制	字母	序号	二进制
,	0	00000	I	9	01001	R	18	10010
A	1	00001	J	10	01010	S	19	10011
B	2	00010	K	11	01011	T	20	10100
C	3	00011	L	12	01100	U	21	10101
D	4	00100	M	13	01101	V	22	10110
E	5	00101	N	14	01110	W	23	10111
F	6	00110	O	15	01111	X	24	11000
G	7	00111	P	16	10000	Y	25	11001
H	8	01000	Q	17	10001	Z	26	11010

分别对以上二元序列作用于函数 f , 得密文为

(2942, 3584, 903, 3326, 215, 2817, 2629, 819)。

为使接收方能够解密, 就需找出单向函数 $f(x)$ 的陷门。为此需引入一种特殊类型的背包向量。

定义背包向量 $A=(a_1, a_2, \dots, a_n)$ 称为超递增的, 如果

$$a_j > \sum_{i=1}^{j-1} a_i \quad j=2, \dots, n$$

超递增背包向量对应的背包问题很容易通过以下算法求解。

已知 s 为背包容积, 对 A 从右向左检查每一元素, 以确定是否在解中。若 $s \geq a_n$, 则 a_n 在解中, 令 $x_n=1$; 若 $s < a_n$, 则 a_n 不在解中, 令 $x_n=0$ 。下面令

$$s = \begin{cases} s, & s < a_n \\ s - a_n, & s \geq a_n \end{cases}$$

对 a_{n-1} 重复上述过程, 一直下去, 直到检查出 a_1 是否在解中。检查结束后得 $x=(x_1 x_2 \dots x_n)$, $B_i=(x_1 x_2 \dots x_n)^T$ 。

然而, 敌手如果也知道超递增背包向量, 同样也很容易解密。为此可用模乘对 A 进行伪装, 模乘的模数 k 和乘数 t 皆取为常量, 满足 $k > \sum_{i=1}^n a_i$, $\gcd(t, k)=1$, 即 t 在模 k 下有乘法逆元。设

$$b_i \equiv t \cdot a_i \pmod{k}, \quad i=1, 2, \dots, n$$

得一新的背包向量 $B=(b_1, b_2, \dots, b_n)$, 记为 $B \equiv t \cdot A \pmod{k}$, 用户以 B 作为自己的公开钥。

【例 4-10】 $A=(1, 3, 5, 11, 21, 44, 87, 175, 349, 701)$ 是一超递增背包向量, 取

$k=1590, t=43, \gcd(43, 1590)=1$, 得 $B=(43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)$ 。

在得到 B 后, 对明文分组 $x=(x_1 x_2 \cdots x_n)$ 的加密运算为

$$c = f(x) = B \cdot B_x \equiv t \cdot A \cdot B_x \pmod{k}$$

对单向函数 $f(x)$, t, t^{-1} 和 k 可作为其秘密的陷门信息, 即解密密钥。解密时首先由 $s \equiv t^{-1} c \pmod{k}$, 求出 s 作为超递增背包向量 A 的容积, 再解背包问题即得 $x=(x_1 x_2 \cdots x_n)$ 。这

是因为 $t^{-1} c \pmod{k} \equiv t^{-1} t A B_x \pmod{k} \equiv A B_x \pmod{k}$, 而由 $k > \sum_{i=1}^n a_i$, 知 $A B_x < k$, 所以 $t^{-1} c \pmod{k} = A B_x$, 是惟一的。

【例 4-11】 接例 4-10, $A=(1, 3, 5, 11, 21, 44, 87, 175, 349, 701)$ 是一超递增背包向量, $k=1590, t=43$, 得 $t^{-1} \equiv 37 \pmod{1590}$, 设用户收到的密文是 $(2942, 3584, 903, 3326, 215, 2817, 2629, 819)$, 由

$$\begin{aligned} 37 \times 2942 &\equiv 734 \pmod{1590}, & 37 \times 3584 &\equiv 638 \pmod{1590}, & 37 \times 903 &\equiv 21 \pmod{1590}, \\ 37 \times 3326 &\equiv 632 \pmod{1590}, & 37 \times 215 &\equiv 5 \pmod{1590}, & 37 \times 2817 &\equiv 879 \pmod{1590}, \\ 37 \times 2629 &\equiv 283 \pmod{1590}, & 37 \times 819 &\equiv 93 \pmod{1590}, \end{aligned}$$

得 $(734, 638, 21, 632, 5, 879, 283, 93)$ 。取 $s=734$, 由 $734 > 701$, 得 $x_{10}=1$; 令 $s=734-701=33$, 由 $33 < 349$, 得 $x_9=0$; 重复该过程得第一个明文分组是 1001100001, 它对应的英文文本是 SA; 类似地得其他明文分组, 解密结果为 SAUNA AND HEALTH。

背包密码体制是 Diffie 和 Hellman 1976 年提出公钥密码体制的设想后的第一个公钥密码体制, 由 Merkle 和 Hellman 1978 年提出。然而又过了两年该体制即被破译, 破译的基本思想是不必找出正确的模数 k 和乘数 t (即陷门信息), 只须找出任意模数 k' 和乘数 t' , 使得用 k' 和 t' 去乘公开的背包向量 B 时, 能够产生超递增的背包向量即可。

4.5 Rabin 密码体制

对 RSA 密码体制, n 被分解成功, 该体制便被破译, 即破译 RSA 的难度不超过大整数的分解。但还不能证明破译 RSA 和分解大整数是等价的, 虽然这一结论已得到普遍共识。Rabin 密码体制已被证明对该体制的破译与分解大整数一样困难。

Rabin 密码体制是对 RSA 的一种修正, 它有以下两个特点:

- 它不是以一一对应的单向陷门函数为基础, 对同一密文, 可能有两个以上对应的明文;
- 破译该体制等价于对大整数的分解。

RSA 中选取的公开钥 e 满足 $1 < e < \varphi(n)$, 且 $\gcd(e, \varphi(n)) = 1$ 。Rabin 密码体制则取 $e = 2$ 。

1. 密钥的产生

随机选择两个大素数 p, q , 满足 $p \equiv q \equiv 3 \pmod{4}$, 即这两个素数形式为 $4k+3$; 计算 $n = p \times q$ 。以 n 作为公开钥, p, q 作为秘密钥。

2. 加密

$$c \equiv m^2 \pmod{n}$$

其中 m 是明文分组, c 是对应的密文分组。

3. 解密

解密就是求 c 模 n 的平方根, 即解 $x^2 \equiv c \pmod{n}$, 由中国剩余定理知解该方程等价于解方程组

$$\begin{cases} x^2 \equiv c \pmod{p} \\ x^2 \equiv c \pmod{q} \end{cases}$$

由于 $p \equiv q \equiv 3 \pmod{4}$, 下面将看到, 方程组的解可容易地求出, 其中每个方程都有两个解, 即

$$\begin{aligned} x &\equiv m \pmod{p}, & x &\equiv -m \pmod{p} \\ x &\equiv m \pmod{q}, & x &\equiv -m \pmod{q} \end{aligned}$$

经过组合可得 4 个同余方程组

$$\begin{aligned} &\begin{cases} x \equiv m \pmod{p} \\ x \equiv m \pmod{q} \end{cases} & \begin{cases} x \equiv m \pmod{p} \\ x \equiv -m \pmod{q} \end{cases} \\ &\begin{cases} x \equiv -m \pmod{p} \\ x \equiv m \pmod{q} \end{cases} & \begin{cases} x \equiv -m \pmod{p} \\ x \equiv -m \pmod{q} \end{cases} \end{aligned}$$

由中国剩余定理可解出每一方程组的解, 共有 4 个, 即每一密文对应的明文不惟一。为了有效地确定明文, 可在 m 中加入某些信息, 如发送者的身份号、接收者的身份号、日期、时间等。

下面证明, 当 $p \equiv q \equiv 3 \pmod{4}$, 两个方程

$$x^2 \equiv c \pmod{p}, \quad x^2 \equiv c \pmod{q}$$

的平方根都可容易地求出。

由 $p \equiv 3 \pmod{4}$ 得, $p+1=4k$, 即 $\frac{1}{4}(p+1)$ 是一整数。因 c 是模 p 的平方剩余, 故

$$\left(\frac{c}{p}\right) \equiv c^{(p-1)/2} \equiv 1 \pmod{p}$$

$$\left(c^{\frac{p+1}{4}}\right)^2 \equiv c^{\frac{p+1}{2}} \equiv c^{(p-1)/2} \cdot c \equiv c \pmod{p}$$

所以 $c^{\frac{p+1}{4}}$ 和 $p - c^{\frac{p+1}{4}}$ 是方程 $x^2 \equiv c \pmod{p}$ 的两个根。同理 $c^{\frac{q+1}{4}}$ 和 $q - c^{\frac{q+1}{4}}$ 是方程 $x^2 \equiv c \pmod{q}$ 的两个根。

由 4.1.8 节知, 求解方程 $x^2 \equiv a \pmod{n}$ 与分解 n 是等价的, 所以破译 Rabin 密码体制的困难程度等价于大整数 n 的分解。

4.6 椭圆曲线密码体制

4.3 节已经说过, 为保证 RSA 算法的安全性, 它的密钥长度需一再增大, 使得它的运算负担越来越大。相比之下, 椭圆曲线密码体制 ECC(elliptic curve cryptography) 可用短得多的密钥获得同样的安全性, 因此具有广泛的应用前景。ECC 已被 IEEE 公钥密码标准 P1363 采用。

4.6.1 椭圆曲线

椭圆曲线并非椭圆, 之所以称为椭圆曲线是因为它的曲线方程与计算椭圆周长的方程类似。一般来讲, 椭圆曲线的曲线方程是以下形式的三次方程:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \quad (4-1)$$

其中 a, b, c, d, e 是满足某些简单条件的实数。定义中包括一个称为无穷点的元素, 记为 O 。图 4-4 是椭圆曲线的两个例子。

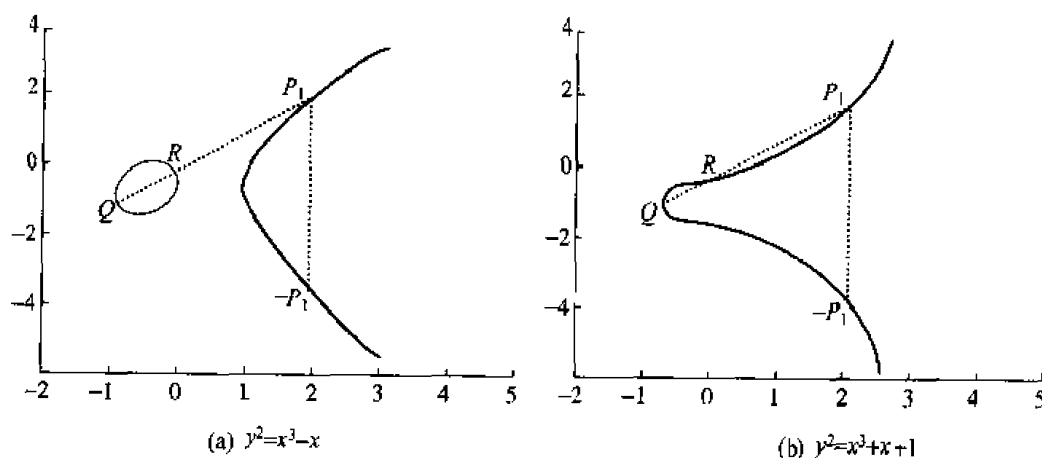


图 4-4 椭圆曲线的两个例子

从图可见,椭圆曲线关于 x 轴对称。

椭圆曲线上的加法运算定义如下:如果其上的 3 个点位于同一直线上,那么它们的和为 O 。进一步可如下定义椭圆曲线上的加法律(加法法则):

① O 为加法单位元,即对椭圆曲线上任一点 P ,有 $P+O=P$ 。

② 设 $P_1=(x,y)$ 是椭圆曲线上的一点(如图 4-4),它的加法逆元定义为 $P_2=-P_1=(x,-y)$ 。

这是因为 P_1, P_2 的连线延长到无穷远时,得到椭圆曲线上的另一点 O ,即椭圆曲线上的 3 点 P_1, P_2, O 共线,所以 $P_1+P_2+O=O, P_1+P_2=O$,即 $P_2=-P_1$ 。

由 $O+O=O$,还可得 $O=-O$ 。

③ 设 Q 和 R 是椭圆曲线上 x 坐标不同的两点, $Q+R$ 的定义如下:画一条通过 Q, R 的直线与椭圆曲线交于 P_1 (这一交点是惟一的,除非所做的直线是 Q 点或 R 点的切线,此时分别取 $P_1=Q$ 和 $P_1=R$)。由 $Q+R+P_1=O$ 得 $Q+R=-P_1$ 。

④ 点 Q 的倍数定义如下:在 Q 点做椭圆曲线的一条切线,设切线与椭圆曲线交于点 S ,定义 $2Q=Q+Q=-S$ 。类似地可定义 $3Q=Q+Q+Q, \dots$, 等。

以上定义的加法具有加法运算的一般性质,如交换律、结合律等。

4.6.2 有限域上的椭圆曲线

密码中普遍采用的是有限域上的椭圆曲线,有限域上的椭圆曲线是指曲线方程定义式(4-1)中,所有系数都是某一有限域 $GF(p)$ 中的元素(其中 p 为一大素数)。其中最为常用的是由方程

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (a, b \in GF(p), 4a^3 + 27b^2 \pmod{p} \neq 0) \quad (4-2)$$

定义的曲线。

【例 4-12】 $p=23, a=b=1, 4a^3+27b^2 \pmod{23} \equiv 8 \neq 0$, 方程(4-2)为 $y^2 \equiv x^3 + x + 1$, 其图形是连续曲线,由图 4-4(b)所示。然而我们感兴趣的是曲线在第一象限中的整数点。设 $E_p(a,b)$ 表示方程(4-2)所定义的椭圆曲线上的点集 $\{(x,y) | 0 \leq x < p, 0 \leq y < p, \text{且 } x, y \text{ 均为整数}\}$ 并上无穷远点 O 。本例中 $E_{23}(1,1)$ 由表 4-6 给出,表中未给出 O 。

表 4-6 椭圆曲线上的点集 $E_{23}(1,1)$

(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)

一般来说, $E_p(a,b)$ 由以下方式产生:

① 对每一 $x(0 \leq x < p \text{ 且 } x \text{ 为整数})$, 计算 $x^3 + ax + b \pmod{p}$ 。

② 决定①中求得的值在模 p 下是否有平方根, 如果没有, 则曲线上没有与这一 x 相对应的点; 如果有, 则求出两个平方根 ($y=0$ 时只有一个平方根)。

$E_p(a, b)$ 上的加法定义如下:

设 $P, Q \in E_p(a, b)$, 则

① $P + O = P$ 。

② 如果 $P = (x, y)$, 那么 $(x, y) + (x, -y) = O$, 即 $(x, -y)$ 是 P 的加法逆元, 表示为 $-P$ 。

由 $E_p(a, b)$ 的产生方式知, $-P$ 也是 $E_p(a, b)$ 中的点, 如上例, $P = (13, 7) \in E_{23}(1, 1)$, $-P = (13, -7)$, 而 $-7 \bmod 23 \equiv 16$, 所以 $-P = (13, 16)$, 也在 $E_{23}(1, 1)$ 中。

③ 设 $P = (x_1, y_1), Q = (x_2, y_2), P \neq -Q$, 则 $P + Q = (x_3, y_3)$ 由以下规则确定:

$$\begin{aligned} x_3 &\equiv \lambda^2 - x_1 - x_2 \pmod{p} \\ y_3 &\equiv \lambda(x_1 - x_3) - y_1 \pmod{p} \end{aligned}$$

其中

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & P = Q \end{cases}$$

【例 4-13】仍以 $E_{23}(1, 1)$ 为例, 设 $P = (3, 10), Q = (9, 7)$, 则

$$\lambda = \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} \equiv 11 \pmod{23}$$

$$x_3 = 11^2 - 3 - 9 = 109 \equiv 17 \pmod{23}$$

$$y_3 = 11(3 - 17) - 10 = -164 \equiv 20 \pmod{23}$$

所以 $P + Q = (17, 20)$, 仍为 $E_{23}(1, 1)$ 中的点。

若求 $2P$ 则

$$\lambda = \frac{3 \cdot 3^2 + 1}{2 \times 10} = \frac{5}{20} = \frac{1}{4} \equiv 6 \pmod{23}$$

$$x_3 = 6^2 - 3 - 3 = 30 \equiv 7 \pmod{23}$$

$$y_3 = 6(3 - 7) - 10 = -34 \equiv 12 \pmod{23}$$

所以 $2P = (7, 12)$ 。

倍点运算仍定义为重复加法, 如 $4P = P + P + P + P$ 。

从例 4-13 看出, 加法运算在 $E_{23}(1, 1)$ 中是封闭的, 且能验证还满足交换律。对一般的 $E_p(a, b)$, 可证其上的加法运算是封闭的、满足交换律, 同样还能证明其上的加法逆元运算也是封闭的, 所以 $E_p(a, b)$ 是一个 Abel 群。

4.6.3 椭圆曲线上的密码

为使用椭圆曲线构造密码体制,需要找出椭圆曲线上的数学困难问题。

在椭圆曲线构成的 Abel 群 $E_p(a, b)$ 上考虑方程 $Q = kP$, 其中 $P, Q \in E_p(a, b), k < p$, 则由 k 和 P 易求 Q , 但由 P, Q 求 k 则是困难的, 这就是椭圆曲线上的离散对数问题, 可应用于公钥密码体制。Diffie-Hellman 密钥交换和 ElGamal 密码体制是基于有限域上离散对数问题的公钥体制, 下面考虑如何用椭圆曲线来实现这两种密码体制。

1. Diffie-Hellman 密钥交换

首先取一素数 $p \approx 2^{160}$ 和两个参数 a, b , 则得方程(4-2)表达的椭圆曲线及其上面的点构成的 Abel 群 $E_p(a, b)$ 。第 2 步, 取 $E_p(a, b)$ 的一个生成元 $G(x_1, y_1)$, 要求 G 的阶是一个非常大的素数, G 的阶是满足 $nG = O$ 的最小正整数 n 。 $E_p(a, b)$ 和 G 作为公开参数。

两用户 A 和 B 之间的密钥交换如下进行:

① A 选一小于 n 的整数 n_A , 作为秘密钥, 并由 $P_A = n_A G$ 产生 $E_p(a, b)$ 上的一点作为公开钥。

② B 类似地选取自己的秘密钥 n_B 和公开钥 P_B 。

③ A、B 分别由 $K = n_A P_B$ 和 $K = n_B P_A$ 产生出双方共享的秘密钥。

这是因为 $K = n_A P_B = n_A (n_B G) = n_B (n_A G) = n_B P_A$ 。

攻击者若想获取 K , 则必须由 P_A 和 G 求出 n_A , 或由 P_B 和 G 求出 n_B , 即需要求椭圆曲线上的离散对数, 因此是不可行的。

【例 4-14】 $p=211, E_p(0, -4)$, 即椭圆曲线为 $y^2 \equiv x^3 - 4$ 。 $G=(2, 2)$ 是 $E_{211}(0, -4)$ 的阶为 241 的一个生成元, 即 $241G = O$ 。 A 的秘密钥取为 $n_A = 121$, 公开钥为 $P_A = 121(2, 2) = (115, 48)$ 。 B 的秘密钥取为 $n_B = 203$, 公开钥为 $P_B = 203(2, 2) = (130, 203)$ 。 由此得到的共享密钥为 $121(130, 203) = 203(115, 48) = (161, 169)$, 即共享密钥是一对数。 如果将这一密钥用作单钥加密的会话密钥, 则可简单地取其中的一个, 如取 x 坐标, 或取 x 坐标的某一简单函数。

2. ElGamal 密码体制

(1) ElGamal 密码体制的原理

密钥产生过程: 首先选择一素数 p 以及两个小于 p 的随机数 g 和 x , 计算 $y \equiv g^x \bmod p$ 。 以 (y, g, p) 作为公开密钥, x 作为秘密密钥。

加密过程: 设欲加密明文消息 M , 随机选一与 $p-1$ 互素的整数 k , 计算 $C_1 \equiv g^k \bmod p, C_2 \equiv y^k M \bmod p$, 密文为 $C = (C_1, C_2)$ 。

解密过程: $M = \frac{C_2}{C_1} \bmod p$

这是因为 $\frac{C_2}{C_1} \bmod p = \frac{y^k M}{g^{kx}} \bmod p = \frac{y^k M}{y^k} \bmod p = M \bmod p$

(2) 利用椭圆曲线实现 ElGamal 密码体制

首先选取一条椭圆曲线, 并得 $E_p(a, b)$, 将明文消息 m 通过编码嵌入到曲线上得点 P_m , 再对点 P_m 做加密变换。这里不对具体的编码方法做进一步介绍, 读者可参考有关文献。

取 $E_p(a, b)$ 的一个生成元 G , $E_p(a, b)$ 和 G 作为公开参数。

用户 A 选 n_A 作为秘密钥, 并以 $P_A = n_A G$ 作为公开钥。任一用户 B 若想向 A 发送消息 P_m , 可选取一随机正整数 k , 产生以下点对作为密文:

$$C_m = \{kG, P_m + kP_A\}$$

A 解密时, 以密文点对中的第二个点减去用自己的秘密钥与第一个点的倍乘, 即

$$P_m + kP_A - n_A kG = P_m + k(n_A G) - n_A kG = P_m$$

攻击者若想由 C_m 得到 P_m , 就必须知道 k 。而要得到 k , 只有通过椭圆曲线上的两个已知点 G 和 kG , 这意味着必须求椭圆曲线上的离散对数, 因此不可行。

【例 4-15】 取 $p=751, E_p(-1, 188)$, 即椭圆曲线为 $y^2 \equiv x^3 - x + 188, E_p(-1, 188)$ 的一个生成元是 $G=(0, 376)$, A 的公开钥为 $P_A=(201, 5)$ 。假定 B 已将欲发往 A 的消息嵌入到椭圆曲线上的点 $P_m=(562, 201)$, B 选取随机数 $k=386$, 由 $kG=386(0, 376)=(676, 558)$, $P_m + kP_A=(562, 201) + 386(201, 5)=(385, 328)$, 得密文为 $\{(676, 558), (385, 328)\}$ 。

3. 椭圆曲线密码体制的优点

与基于有限域上离散对数问题的公钥体制(如 Diffie-Hellman 密钥交换和 ElGamal 密码体制)相比, 椭圆曲线密码体制有如下优点。

(1) 安全性高

攻击有限域上的离散对数问题可以用指数积分法, 其运算复杂度为 $O(\exp(\sqrt[3]{(\log p)(\log \log p)^2}))$, 其中 p 是模数(为素数)。而它对椭圆曲线上的离散对数问题并不有效。目前攻击椭圆曲线上的离散对数问题的方法只有适合攻击任何循环群上离散对数问题的大步小步法, 其运算复杂度为 $O(\exp(\log \sqrt{p_{\max}}))$, 其中 p_{\max} 是椭圆曲线所形成的 Abel 群的阶的最大素因子。因此, 椭圆曲线密码体制比基于有限域上的离散对数问题的公钥体制更安全。

(2) 密钥量小

由攻击两者的算法复杂度可知,在实现相同的安全性能条件下,椭圆曲线密码体制所需的密钥量远比基于有限域上的离散对数问题的公钥体制的密钥量小。

(3) 灵活性好

有限域 $GF(q)$ 一定的情况下,其上的循环群(即 $GF(q) - \{0\}$)就定了。而 $GF(q)$ 上的椭圆曲线可以通过改变曲线参数,得到不同的曲线,形成不同的循环群。因此,椭圆曲线具有丰富的群结构和多选择性。

正是由于椭圆曲线具有丰富的群结构和多选择性,并可在保持和 RSA/DSA 体制同样安全性能的前提下大大缩短密钥长度(目前 160 比特足以保证安全性),因而在密码领域有着广阔的应用前景。表 4-7 给出了椭圆曲线密码体制和 RSA/DSA 体制在保持同等安全的条件下各自所需的密钥的长度。

表 4-7 ECC 和 RSA/DSA 在保持同等安全的条件下所需的密钥长度(单位为比特)

RSA/DSA	512	768	1024	2048	21000
ECC	106	132	160	211	600

习 题

1. 证明以下关系:

(1) $(a \bmod n) = (b \bmod n)$, 则 $a \equiv b \pmod n$;

(2) $a \equiv b \pmod n$, 则 $b \equiv a \pmod n$;

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

2. 证明以下关系:

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

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

3. 用 Fermat 定理求 $3^{201} \bmod 11$ 。

4. 用推广的 Euclid 算法求 $67 \bmod 119$ 的逆元。

5. 求 $\gcd(4655, 12075)$ 。

6. 求解下列同余方程组:

$$\begin{cases} x \equiv 2 \pmod 3 \\ x \equiv 1 \pmod 5 \\ x \equiv 1 \pmod 7 \end{cases}$$

7. 计算下列 Legendre 符号:

(1) $\left(\frac{2}{59}\right)$; (2) $\left(\frac{6}{53}\right)$; (3) $\left(\frac{65}{107}\right)$.

8. 求 25 的所有本原根。

9. 设通信双方使用 RSA 加密体制,接收方的公开钥是 $(e, n) = (5, 35)$,接收到的密文是 $C=10$,求明文 M 。

10. 在 ElGamal 加密体制中,设素数 $p=71$,本原根 $g=7$,

(1) 如果接收方 B 的公开钥是 $y_B=3$,发送方 A 选择的随机整数 $k=2$,求明文 $M=30$ 所对应的密文。

(2) 如果 A 选择另一个随机整数 k ,使得明文 $M=30$ 加密后的密文是 $C=(59, C_2)$,求 C_2 。

11. 设背包密码系统的超递增序列为 $(3, 4, 9, 17, 35)$,乘数 $t=19$,模数 $k=73$,试对 good night 加密。

12. 设背包密码系统的超递增序列为 $(3, 4, 8, 17, 33)$,乘数 $t=17$,模数 $k=67$,试对密文 25、2、72、92 解密。

13. 在 Rabin 密码体制中设 $p=53, q=59$

(1) 确定 1 在模 n 下的 4 个平方根。

(2) 求明文消息 2347 所对应的密文。

(3) 对上述密文,确定可能的 4 个明文。

14. 椭圆曲线 $E_{11}(1, 6)$ 表示 $y^2 \equiv x^3 + x + 6 \pmod{11}$,求其上的所有点。

15. 已知点 $G=(2, 7)$ 在椭圆曲线 $E_{11}(1, 6)$ 上,求 $2G$ 和 $3G$ 。

16. 利用椭圆曲线实现 ElGamal 密码体制,设椭圆曲线是 $E_{11}(1, 6)$,生成元 $G=(2, 7)$,接收方 A 的秘密钥 $n_A=7$ 。

(1) 求 A 的公开钥 P_A 。

(2) 发送方 B 欲发送消息 $P_m=(10, 9)$,选择随机数 $k=3$,求密文 C_m 。

(3) 显示接收方 A 从密文 C_m 恢复消息 P_m 的过程。

第 5 章

密钥分配与密钥管理

5.1 单钥加密体制的密钥分配

5.1.1 密钥分配的基本方法

两个用户(主机、进程、应用程序)在用单钥密码体制进行保密通信时,首先必须有一个共享的秘密密钥,而且为防止攻击者得到密钥,还必须时常更新密钥。因此,密码系统的强度也依赖于密钥分配技术。两个用户 A 和 B 获得共享密钥的方法有以下几种:

- ① 密钥由 A 选取并通过物理手段发送给 B。
- ② 密钥由第三方选取并通过物理手段发送给 A 和 B。
- ③ 如果 A、B 事先已有一密钥,则其中一方选取新密钥后,用已有的密钥加密新密钥并发送给另一方。
- ④ 如果 A 和 B 与第三方 C 分别有一保密信道,则 C 为 A、B 选取密钥后,分别在两个保密信道上发送给 A、B。

前两种方法称为人工发送。在通信网中,若只有个别用户想进行保密通信,密钥的人工发送还是可行的。然而如果所有用户都要求支持加密服务,则任意一对希望通信的用户都必须有一共享密钥。如果有 n 个用户,则密钥数目为 $n(n-1)/2$ 。因此当 n 很大时,密钥分配的代价非常大,密钥的人工发送是不可行的。

对于第 3 种方法,攻击者一旦获得一个密钥就可获取以后所有的密钥;而且用这种方法对所有用户分配初始密钥时,代价仍然很大。

第 4 种方法比较常用,其中的第三方通常是一个负责为用户分配密钥的密钥分配中心。这时每一用户必须和密钥分配中心有一个共享密钥,称为主密钥。通过主密钥分配给一对用户的密钥称为会话密钥,用于这一对用户之间的保密通信。通信完成后,会话密钥即被销毁。如上所述,如果用户数为 n ,则会话密钥数为 $n(n-1)/2$ 。但主密钥数却只需 n 个,所以主密钥可通过物理手段发送。

5.1.2 一个实例

图 5-1 是密钥分配的一个实例。假定两个用户 A、B 分别与密钥分配中心 KDC (key distribution center) 有一个共享的主密钥 K_A 和 K_B , A 希望与 B 建立一个共享的一次性会话密钥, 可通过以下几步来完成:

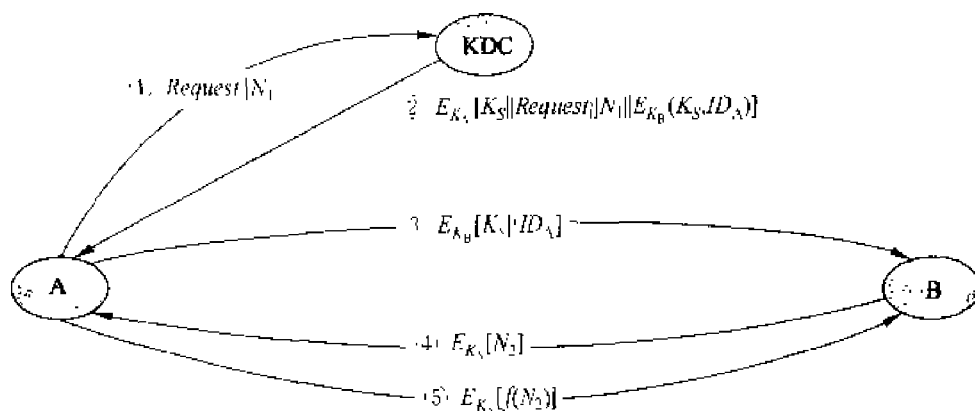


图 5-1 密钥分配实例

① A 向 KDC 发出会话密钥请求。表示请求的消息由两个数据项组成, 第 1 项是 A 和 B 的身份, 第 2 项是这次业务的惟一识别符 N_1 , 称 N_1 为一次性随机数, 可以是时戳、计数器或随机数。每次请求所用的 N_1 都应不同, 且为防止假冒, 应使敌手对 N_1 难以猜测。因此用随机数作为这个识别符最为合适。

② KDC 为 A 的请求发出应答。应答是由 K_A 加密的消息, 因此只有 A 才能成功地对这一消息解密, 并且 A 可相信这一消息的确是由 KDC 发出的。消息中包括 A 希望得到的两项内容:

- 一次性会话密钥 K_S ;
- A 在①中发出的请求, 包括一次性随机数 N_1 , 目的是使 A 将收到的应答与发出的请求相比较, 看是否匹配。

因此 A 能验证自己发出的请求在被 KDC 收到之前, 是否被他人篡改。而且 A 还能根据一次性随机数相信自己收到的应答不是重放的过去的应答。

此外, 消息中还有 B 希望得到的两项内容:

- 一次性会话密钥 K_S ;
- A 的身份(例如 A 的网络地址) ID_A 。

这两项由 K_B 加密, 将由 A 转发给 B, 以建立 A、B 之间的连接并用于向 B 证明 A 的身份。

③ A 存储会话密钥,并向 B 转发 $E_{K_B}[K_S \parallel ID_A]$ 。因为转发的是由 K_B 加密后的密文,所以转发过程不会被窃听。B 收到后,可得会话密钥 K_S ,并从 ID_A 可知另一方是 A,而且还从 E_{K_B} 知道 K_S 的确来自 KDC。

这一步完成后,会话密钥就安全地分配给了 A、B。然而还能继续以下两步工作:

④ B 用会话密钥 K_S 加密另一个一次性随机数 N_2 ,并将加密结果发送给 A。

⑤ A 以 $f(N_2)$ 作为对 B 的应答,其中 f 是对 N_2 进行某种变换(例如加 1)的函数,并将应答用会话密钥加密后发送给 B。

这两步可使 B 相信第③步收到的消息不是一个重放。

注意:第③步就已完成密钥分配,第④、⑤两步结合第③步执行的是认证功能。

5.1.3 密钥的分层控制

网络中如果用户数目非常多而且分布的地域非常广,一个 KDC 就无法承担为用户分配密钥的重任。问题的解决方法是使用多个 KDC 的分层结构。例如,在每个小范围(如一个 LAN 或一个建筑物)内,都建立一个本地 KDC。同一范围的用户在进行保密通信时,由本地 KDC 为他们分配密钥。如果两个不同范围的用户想获得共享密钥,则可通过各自的本地 KDC,而两个本地 KDC 的沟通又需经过一个全局 KDC。这样就建立了两层 KDC。类似地,根据网络中用户的数目及分布的地域,可建立 3 层或多层 KDC。

分层结构可减少主密钥的分布,因为大多数主密钥是在本地 KDC 和本地用户之间共享。再者,分层结构还可将虚假 KDC 的危害限制到一个局部区域。

5.1.4 会话密钥的有效期

会话密钥更换得越频繁,系统的安全性就越高。因为敌手即使获得一个会话密钥,也只能获得很少的密文。但另一方面,会话密钥更换得太频繁,又将延迟用户之间的交换,同时还造成网络负担。所以在决定会话密钥的有效期时,应权衡矛盾的两个方面。

对面向连接的协议,在连接未建立前或断开时,会话密钥的有效期可以很长。而每次建立连接时,都应使用新的会话密钥。如果逻辑连接的时间很长,则应定期更换会话密钥。

无连接协议(如面向业务的协议),无法明确地决定更换密钥的频率。为安全起见,用户每进行一次交换,都用新的会话密钥。然而这又失去了无连接协议主要的优势,即对每个业务都有最少的费用和最短的延迟。比较好的方案是在某一固定周期内或对一定数目的业务使用同一会话密钥。

5.1.5 无中心的密钥控制

用密钥分配中心为用户分配密钥时,要求所有用户都信任 KDC,同时还要求对 KDC

加以保护。如果密钥的分配是无中心的,则不必有以上两个要求。然而如果每个用户都能和自己想与之建立联系的另一用户安全地通信,则对有 n 个用户的网络来说,主密钥应多达 $n(n-1)/2$ 个。当 n 很大时,这种方案无实用价值,但在整个网络的局部范围却非常有用。

无中心的密钥分配时,两个用户 A 和 B 建立会话密钥需经过以下 3 步,见图 5-2:

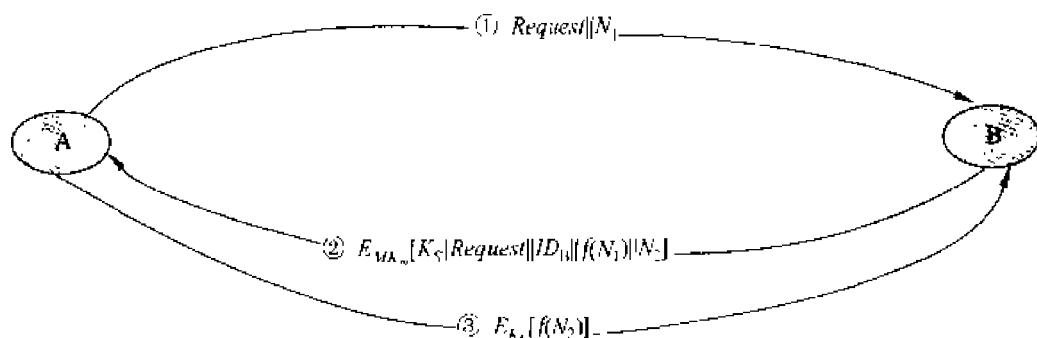


图 5-2 无中心的密钥分配

① A 向 B 发出建立会话密钥的请求和一个一次性随机数 N_1 。

② B 用与 A 共享的主密钥 MK_m 对应答的消息加密,并发送给 A。应答的消息中有 B 选取的会话密钥、B 的身份、 $f(N_1)$ 和另一个一次性随机数 N_2 。

③ A 使用新建立的会话密钥 K_s 对 $f(N_2)$ 加密后返回给 B。

5.1.6 密钥的控制使用

密钥可根据其不同用途分为会话密钥和主密钥两种类型,会话密钥又称为数据加密密钥,主密钥又称为密钥加密密钥。由于密钥的用途不同,因此对密钥的使用方式也希望加以某种控制。

如果主密钥泄露了,则相应的会话密钥也将泄露,因此主密钥的安全性应高于会话密钥的安全性。一般在密钥分配中心以及终端系统中主密钥都是物理上安全的,如果把主密钥当作会话密钥注入加密设备,那么其安全性则降低。

单钥体制中的密钥控制技术有以下两种。

(1) 密钥标签

用于 DES 的密钥控制,将 DES 的 64 比特密钥中的 8 个校验位作为控制使用这一密钥的标签。标签中各比特的含义为:

- 一个比特表示这个密钥是会话密钥还是主密钥;
- 一个比特表示这个密钥是否能用于加密;
- 一个比特表示这个密钥是否能用于解密;

- 其他比特无特定含义,留待以后使用。

由于标签是在密钥之中,在分配密钥时,标签与密钥一起被加密,因此可对标签起到保护作用。本方案的缺点:第一,标签的长度被限制为 8 比特,限制了它的灵活性和功能;第二,由于标签是以密文形式传送,只有解密后才能使用,因而限制了对密钥使用的控制方式。

(2) 控制矢量

这一方案比上一方案灵活。方案中对每一会话密钥都指定了一个相应的控制矢量,控制矢量分为若干字段,分别用于说明在不同情况下密钥是被允许使用还是不被允许使用,且控制矢量的长度可变。控制矢量是在 KDC 产生密钥时加在密钥之中的,过程由图 5-3(a)所示。首先由一杂凑函数将控制矢量压缩到与加密密钥等长,然后与主密钥异或后作为加密会话密钥的密钥,即

$$H = h(CV)$$

$$K_m = K_m \oplus H$$

$$K_{out} = E_{K_m \oplus H}[K_s]$$

其中 CV 是控制矢量, h 是杂凑函数, K_m 是主密钥, K_s 是会话密钥。会话密钥的恢复过程如图 5-3(b)所示,表示为:

$$K_s = D_{K_m \oplus H}[E_{K_m \oplus H}[K_s]]$$

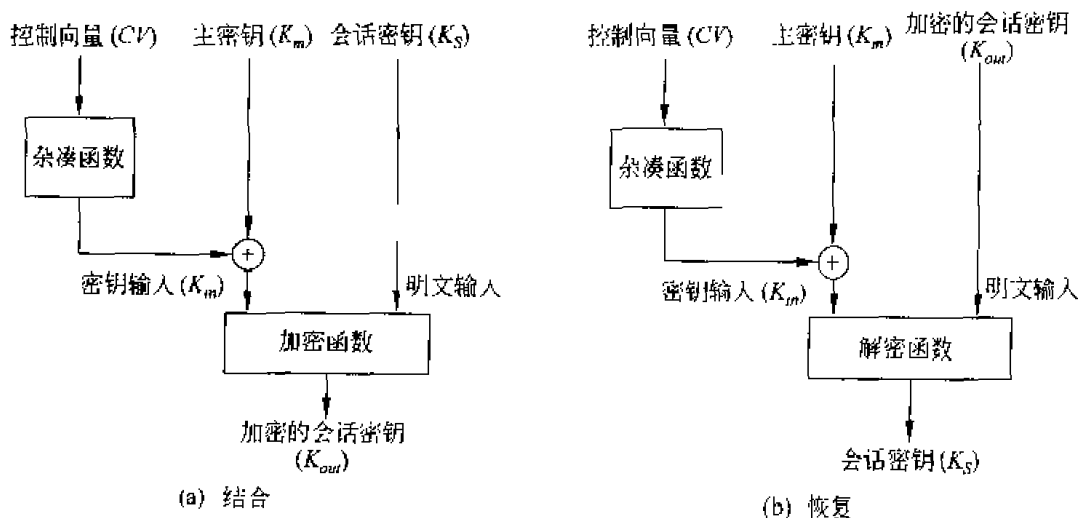


图 5-3 控制矢量的使用方式

KDC 在向用户发送会话密钥时,同时以明文形式发送控制矢量。用户只有使用与 KDC 共享的主密钥以及 KDC 发送来的控制矢量才能恢复会话密钥,因此还必须保留会话密钥和它的控制矢量之间的对应关系。

与使用 8 比特的密钥标签相比,使用控制矢量有两个优点:第一,控制矢量的长度没有限制,因此可对密钥的使用施加任意复杂的控制;第二,控制矢量始终是以明文形式存在,因此可在任一阶段对密钥的使用施加控制。

5.2 公钥加密体制的密钥管理

前一节介绍了单钥密码体制中的密钥分配问题,而公钥加密的一个主要用途是分配单钥密码体制使用的密钥。本节介绍两方面内容:一是公钥密码体制所用的公开密钥的分配,二是如何用公钥体制来分配单钥密码体制所需的密钥。

5.2.1 公钥的分配

本小节讲述公钥的分配方法。

1. 公开发布

公开发布指用户将自己的公钥发给每一其他用户,或向某一团体广播。例如 PGP (pretty good privacy) 中采用了 RSA 算法,它的很多用户都是将自己的公钥附加到消息上,然后发送到公开(公共)区域,如因特网邮件列表。

这种方法虽然简单,但有一个非常大的缺点,即任何人都可伪造这种公开发布。如果某个用户假装是用户 A 并以 A 的名义向另一用户发送或广播自己的公开钥,则在 A 发现假冒者以前,这一假冒者可解读所有意欲发向 A 的加密消息,而且假冒者还能用伪造的密钥获得认证。

2. 公用目录表

公用目录表指一个公用的公钥动态目录表,公用目录表的建立、维护以及公钥的分布由某个可信的实体或组织承担,称这个实体或组织为公用目录的管理员。与第 1 种分配方法相比,这种方法的安全性更高。该方案有以下一些组成部分:

① 管理员为每个用户都在目录表中建立一个目录,目录中有两个数据项:一是用户名,二是用户的公开钥。

② 每一用户都亲自或以某种安全的认证通信在管理者那里为自己的公开钥注册。

③ 用户如果由于自己的公开钥用过的次数太多或由于与公开钥相关的秘密钥已被泄露,则可随时用新密钥替换现有的密钥。

④ 管理员定期公布或定期更新目录表。例如,像电话号码本一样公布目录表或在发行量很大的报纸上公布目录表的更新。

⑤ 用户可通过电子手段访问目录表,这时从管理员到用户必须有安全的认证通信。

本方案的安全性虽然高于公开发布的安全性,但仍易受攻击。如果敌手成功地获取管理员的秘密钥,就可伪造一个公钥目录表,以后既可假冒任一用户又能监听发往任一用户的信息。而且公用目录表还易受到敌手的窜扰。

3. 公钥管理机构

如果在公钥目录表中对公钥的分配施加更严密的控制,安全性将会更强。与公用目录表类似,这里假定有一个公钥管理机构来为各用户建立、维护动态的公钥目录,但同时系统提出以下要求,即:每个用户都可靠地知道管理机构的公开钥,而只有管理机构自己知道相应的秘密钥。公开钥的分配步骤如下,如图 5-4:

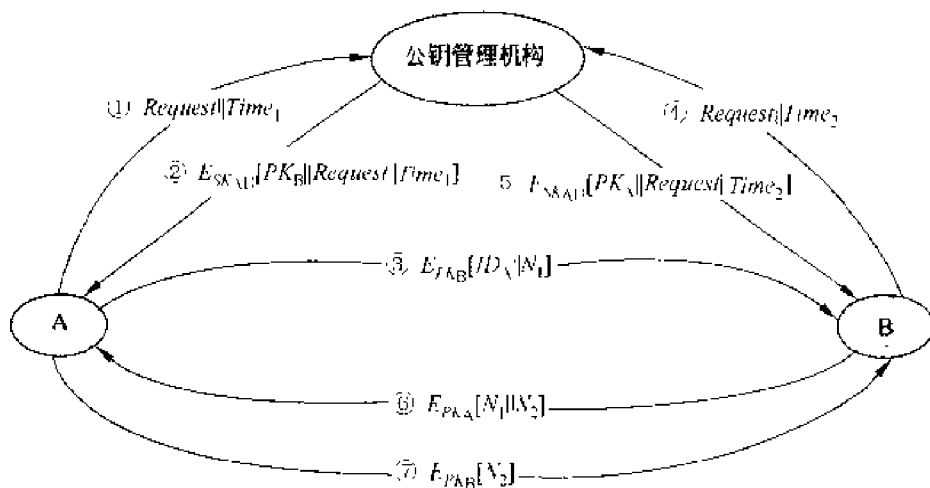


图 5-4 公钥管理机构分配公钥

① 用户 A 向公钥管理机构发送一个带时戳的消息,消息中有获取用户 B 的当前公钥的请求。

② 管理机构对 A 的请求作出应答,应答由一个消息表示,该消息由管理机构用自己的秘密钥 SK_{PM} 加密,因此 A 能用管理机构的公开钥解密,并使 A 相信这个消息的确是来源于管理机构。

应答的消息中有以下几项:

- B 的公钥 PK_B ,A 可用之对将发往 B 的消息加密;
- A 的请求,用于 A 验证收到的应答的确是对相应请求的应答,且还能验证自己最初发出的请求在被管理机构收到以前是否被篡改;
- 最初的时戳,以便 A 相信管理机构发来的消息不是一个旧消息,因此消息中的公

开钥的确是 B 当前的公钥。

③ A 用 B 的公开钥对一个消息加密后发往 B, 这个消息有两个数据项: 一是 A 的身份 ID_A , 二是一个一次性随机数 N_1 , 用于惟一地标识这次业务。

④ B 以相同方式从管理机构获取 A 的公开钥(与步骤①、②类似)。这时, A 和 B 都已安全地得到了对方的公钥, 所以可进行保密通信。然而, 他们也许还希望有以下两步, 以认证对方。

⑤ B 用 PK_A 对一个消息加密后发往 A, 该消息的数据项有 A 的一次性随机数 N_1 和 B 产生的一个一次性随机数 N_2 。因为只有 B 能解密③的消息, 所以 A 收到的消息中的 N_1 可使其相信通信的另一方的确是 B。

⑥ A 用 B 的公开钥对 N_2 加密后返回给 B, 可使 B 相信通信的另一方的确是 A。

以上过程共发送了 7 个消息, 其中前 4 个消息用于获取对方的公开钥。用户得到对方的公开钥后保存起来可供以后使用, 这样就不必再发送前 4 个消息了, 然而还必须定期地通过密钥管理中心获取通信对方的公开钥, 以免对方的公开钥更新后无法保证当前的通信。

4. 公钥证书

上述公钥管理机构分配公开钥时也有缺点, 由于每一用户要想和他人联系都需求助于管理机构, 所以管理机构有可能成为系统的瓶颈, 而且由管理机构维护的公钥目录表也易被敌手窜扰。

分配公钥的另一方法是公钥证书, 用户通过公钥证书来互相交换自己的公钥而无须与公钥管理机构联系。公钥证书由证书管理机构 CA (certificate authority) 为用户建立, 其中的数据项有与该用户的秘密钥相匹配的公开钥及用户的身份和时戳等, 所有的数据项经 CA 用自己的秘密钥签字后就形成证书, 即证书的形式为 $C_A = E_{SK_{CA}}[T, ID_A, PK_A]$, 其中 ID_A 是用户 A 的身份, PK_A 是 A 的公钥, T 是当前时戳, SK_{CA} 是 CA 的秘密钥, C_A 即是为用户 A 产生的证书。产生过程如图 5-5 所示。用户可将自己的公开钥通过公钥证书发给另一用户, 接收方可用 CA 的公钥 PK_{CA} 对证书加以验证, 即

$$D_{PK_{CA}}[C_A] = D_{PK_{CA}}[E_{SK_{CA}}[T, ID_A, PK_A]] = (T, ID_A, PK_A)$$

因为只有用 CA 的公钥才能解读证书, 接收方从而验证了证书的确是由 CA 发放的, 且也获得了发送方的身份 ID_A 和公开钥 PK_A 。时戳 T 为接收方保证了收到的证书的新鲜性, 用以防止发送方或敌方重放一旧证书。因此时戳可被当作截止日期, 证书如果过旧, 则被吊销。

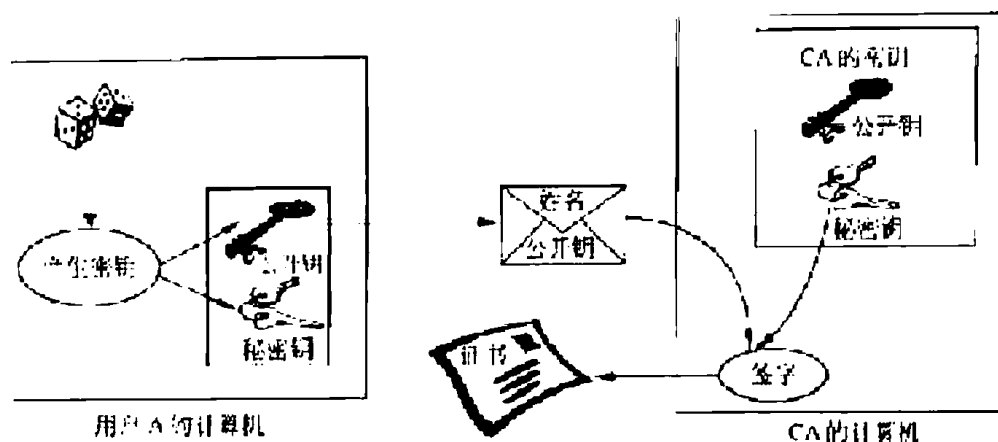


图 5-5 证书的产生过程

5.2.2 用公钥加密分配单钥密码体制的密钥

公开钥分配完成后,用户就可用公钥加密体制进行保密通信。然而由于公钥加密的速度过慢,以此进行保密通信不太合适,但用于分配单钥密码体制的密钥却非常合适。

1. 简单分配

图 5-6 表示简单使用公钥加密算法建立会话密钥的过程,如果 A 希望与 B 通信,可通过以下几步建立会话密钥:

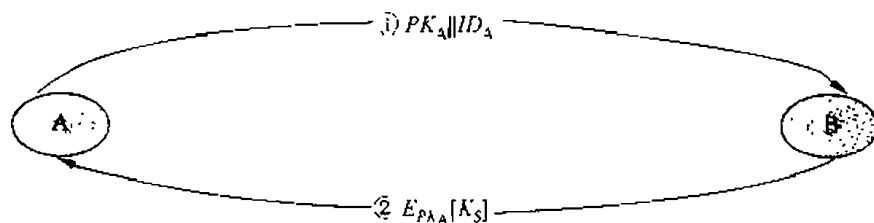


图 5-6 简单使用公钥加密算法建立会话密钥

① A 产生自己的一对密钥 $\{PK_A, SK_A\}$, 并向 B 发送 $PK_A || ID_A$, 其中 ID_A 表示 A 的身份。

② B 产生会话密钥 K_S , 并用 A 的公开钥 PK_A 对 K_S 加密后发往 A。

③ A 由 $D_{SK_A}[E_{PK_A}[K_S]]$ 恢复会话密钥。因为只有 A 能解读 K_S , 所以仅 A、B 知道这一共享密钥。

④ A 销毁 $\{PK_A, SK_A\}$, B 销毁 PK_A 。

A、B 现在可以用单钥加密算法以 K_S 作为会话密钥进行保密通信, 通信完成后, 又都

将 K_S 销毁。这种分配法尽管简单,但却由于 A、B 双方在通信前和完成通信后,都未存储密钥,因此,密钥泄露的危险性为最小,且可防止双方的通信被敌手监听。

这一协议易受到主动攻击,如果敌手 E 已接入 A、B 双方的通信信道,就可通过以下不被察觉的方式截获双方的通信:

① 与上面的步骤①相同。

② E 截获 A 的发送后,建立自己的一对密钥 $\{PK_E, SK_E\}$,并将 $PK_E \parallel ID_A$ 发送给 B。

③ B 产生会话密钥 K_S 后,将 $E_{PK_E}[K_S]$ 发送出去。

④ E 截获 B 发送的消息后,由 $D_{SK_E}[E_{PK_E}[K_S]]$ 解读 K_S 。

⑤ E 再将 $E_{PK_A}[K_S]$ 发往 A。

现在 A 和 B 知道 K_S ,但并未意识到 K_S 已被 E 截获。A、B 在用 K_S 通信时,E 就可以实施监听。

2. 具有保密性和认证性的密钥分配

图 5-7 所示的密钥分配过程具有保密性和认证性,因此既可防止被动攻击,又可防止主动攻击。

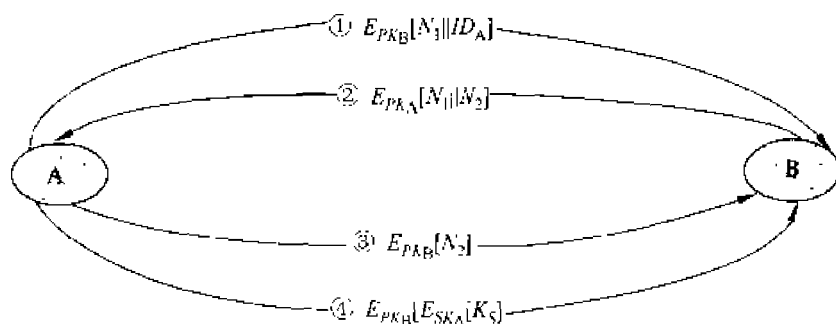


图 5-7 具有保密性和认证性的密钥分配

假定 A、B 双方已完成公钥交换,可按以下步骤建立共享会话密钥:

① A 用 B 的公开钥加密 A 的身份 ID_A 和一个一次性随机数 N_1 后发往 B,其中 N_1 用于唯一地标识这一业务。

② B 用 A 的公开钥 PK_A 加密 A 的一次性随机数 N_1 和 B 新产生的一次性随机数 N_2 后发往 A。因为只有 B 能解读①中的加密,所以 B 发来的消息中 N_1 的存在可使 A 相信对方的确是 B。

③ A 用 B 的公钥 PK_B 对 N_2 加密后返回给 B,以使 B 相信对方的确是 A。

④ A 选一会话密钥 K_S ,然后将 $M = E_{PK_B}[E_{SK_A}[K_S]]$ 发给 B,其中用 B 的公开钥加密

是为保证只有 B 能解读加密结果,用 A 的秘密钥加密是保证该加密结果只有 A 能发送。

⑤ B 以 $D_{PK_A}[D_{SK_B}[M]]$ 恢复会话密钥。

5.2.3 Diffie-Hellman 密钥交换

Diffie-Hellman 密钥交换是 W. Diffie 和 M. Hellman 于 1976 年提出的第一个公钥密码算法,已在很多商业产品中得以应用。算法的惟一目的是使得两个用户能够安全地交换密钥,得到一个共享的会话密钥,算法本身不能用于加、解密。

算法的安全性基于求离散对数的困难性。

图 5-8 表示 Diffie-Hellman 密钥交换过程,其中 p 是大素数, a 是 p 的本原根, p 和 a 作为公开的全程元素。用户 A 选择一保密的随机整数 X_A ,并将 $Y_A = a^{X_A} \bmod p$ 发送给用户 B。类似地,用户 B 选择一保密的随机整数 X_B ,并将 $Y_B = a^{X_B} \bmod p$ 发送给用户 A。然后 A 和 B 分别由 $K = Y_B^{X_A} \bmod p$ 和 $K = Y_A^{X_B} \bmod p$ 计算出的就是共享密钥,这是因为

$$\begin{aligned} Y_B^{X_A} \bmod p &= (a^{X_B} \bmod p)^{X_A} \bmod p = (a^{X_B})^{X_A} \bmod p = a^{X_B X_A} \bmod p \\ &= (a^{X_A})^{X_B} \bmod p = (a^{X_A} \bmod p)^{X_B} \bmod p = Y_A^{X_B} \bmod p \end{aligned}$$

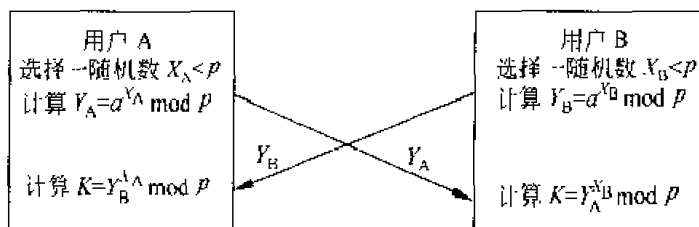


图 5-8 Diffie-Hellman 密钥交换

因 X_A, X_B 是保密的,敌手只能得到 p, a, Y_A, Y_B ,要想得到 K ,则必须得到 X_A, X_B 中的一个,这意味着需要求离散对数。因此敌手求 K 是不可行的。

例如: $p = 97, a = 5, A$ 和 B 分别秘密选 $X_A = 36, X_B = 58$,并分别计算 $Y_A = 5^{36} \bmod 97 = 50, Y_B = 5^{58} \bmod 97 = 44$ 。在交换 Y_A, Y_B 后,分别计算

$$K = Y_B^{X_A} \bmod 97 = 44^{36} \bmod 97 = 75, K = Y_A^{X_B} \bmod 97 = 50^{58} \bmod 97 = 75$$

5.3 密钥托管

密钥托管也称为托管加密,其目的是保证对个人没有绝对的隐私和绝对不可跟踪的匿名性,即在强加密中结合对突发事件的解密能力。其实现手段是把已加密的数据和数据恢复密钥联系起来,数据恢复密钥不必是直接解密的密钥,但由它可得解密密钥。数据

恢复密钥由所信任的委托人持有,委托人可以是政府机构、法院或有契约的私人组织。一个密钥可能是在数个这样的委托人中分拆。调查机构或情报机构通过适当的程序,如获得法院证书,从委托人处获得数据恢复密钥。

密钥托管加密技术提供了一个备用的解密途径,政府机构在需要时,可通过密钥托管技术解密用户的信息,而用户的密钥若丢失或损坏,也可通过密钥托管技术恢复自己的密钥。所以这个备用的手段不仅对政府有用,而且对用户自己也有用。

5.3.1 美国托管加密标准简介

1993年4月,美国政府为了满足其电信安全、公众安全和国家安全,提出了托管加密标准 EES(escrowed encryption standard),该标准所使用的托管加密技术不仅提供了强加密功能,同时也为政府机构提供了实施法律授权下的监听功能。这一技术是通过一个防窜扰的芯片(称为 Clipper 芯片)来实现的。它有两个特性:

① 一个加密算法——Skipjack 算法,该算法是由 NSA 设计的,用于加(解)密用户间通信的消息。该算法已于 1998 年 3 月公布。

② 为法律实施提供“后门”的部分——法律实施存取域 LEAF(law enforcement access field)。通过这个域,法律实施部门可在法律授权下,实现对用户通信的解密。

1. Skipjack 算法

Skipjack 算法是一个单钥分组加密算法,密钥长 80 比特,输入和输出的分组长均为 64 比特。可使用 4 种工作模式:电码本模式,密码分组链接模式,64 比特输出反馈模式,1、8、16、32 或 64 比特密码反馈模式。

算法的内部细节在向公众公开以前,政府邀请了一些局外人士对算法作出评价,并公布了评价结果。评价结果认为算法的强度高于 DES,并且未发现陷门。Skipjack 的密钥长是 80 比特,比 DES 的密钥长 24 比特,因此通过穷搜索的蛮力攻击比 DES 多 2^{24} 倍的搜索。所以若假定处理能力的费用每 18 个月减少一半,那么破译它所需的代价要 $1.5 \times 24 = 36$ 年才能减少到今天破译 DES 的代价。

2. 托管加密芯片

Skipjack 算法以及在法律授权下对加密结果的存取是通过防窜扰的托管加密芯片来实现的。芯片装有以下部分:

- Skipjack 算法;
- 80 比特的族密钥 KF (family key),同一批芯片的族密钥都相同;
- 芯片单元识别符 UID (unique identifier);
- 80 比特的芯片单元密钥 KU (unique key),它是两个 80 比特的芯片单元密钥分量

(KU_1, KU_2) 的异或;

- 控制软件。

这些部分被固化在芯片上。编程过程是在由两个托管机构的代表监控下的安全工厂中进行的,一段时间一批。编程过程如图 5-9 所示。

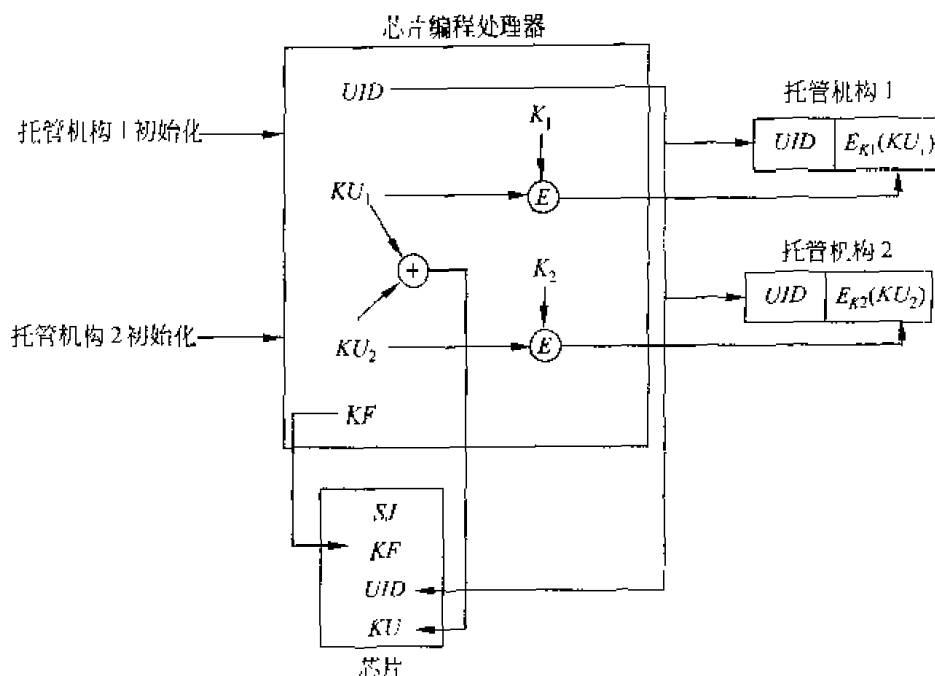


图 5-9 托管加密芯片的编程过程

首先,托管机构的代表通过向编程设备输入两个参数(随机数)对芯片编程处理器初始化。芯片编程处理器对每个芯片,分别计算以上两个初始参数和 UID 的函数,作为单元密钥的两个分量 KU_1 和 KU_2 。求 $KU_1 \text{ XOR } KU_2$,作为芯片单元密钥 KU 。 UID 和 KU 放在芯片中。然后,用分配给托管机构 1 的密钥 K_1 加密 KU_1 得 $E_{K_1}(KU_1)$ 。类似地,用分配给托管机构 2 的加密密钥 K_2 加密 KU_2 得 $E_{K_2}(KU_2)$ 。 $(UID, E_{K_1}(KU_1))$ 和 $(UID, E_{K_2}(KU_2))$ 分别给托管机构 1 和托管机构 2,并以托管形式保存。以加密方式保存单元密钥分量是为了防止密钥分量被窃或泄露。

编程过程结束后,编程处理器被清除,以使芯片的单元密钥不能被他人获得或被他人计算,只能从两个托管机构获得加了密的单元密钥分量,并且使用特定的政府解密设备来解密。

3. 用托管加密芯片加密

通信双方为了使用 Skipjack 算法加密他们的通信,都必须有一个装有托管加密芯片的安全的防窜扰设备,该设备负责实现建立安全信道所需的协议,包括协商或分布用于加

密通信的 80 比特秘密会话密钥 KS 。例如,会话密钥可使用 Diffie-Hellman 密钥协商协议,该协议执行过程中,两个设备仅交换公共值即可获得公共的秘密会话密钥。

80 比特的会话密钥 KS 建立后,被传送给加密芯片,用于与初始化向量 IV (由芯片产生)一起产生 LEAF。控制软件使用芯片单元密钥 KU 加密 KS ,然后将加密后的结果和芯片识别符 UID 、认证符 A 链接,再使用公共的族密钥 KF 加密以上链接的结果而产生 LEAF。其过程如图 5-10 所示。

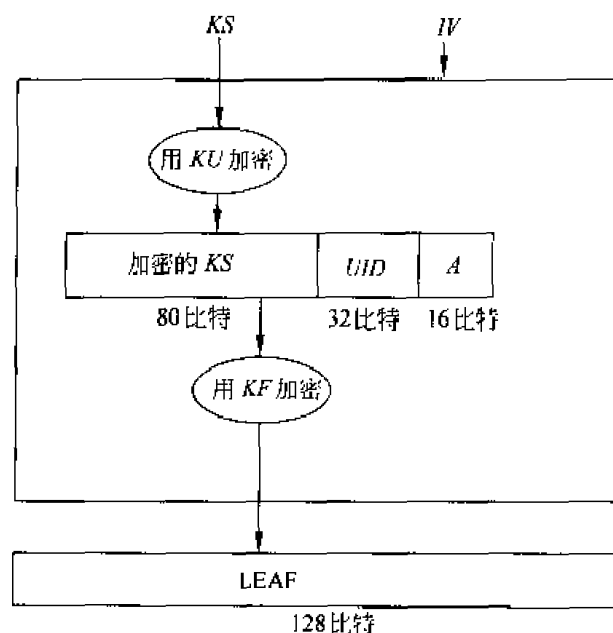


图 5-10 LEAF 产生过程示意图

最后将 IV 和 LEAF 传递给接收芯片,用于建立同步。同步建立后,会话密钥就可用于通信双方的加解密。对语音通信,消息串(语音)首先应被数字化。图 5-11 显示的是在发送者的安全设备和接收者的安全设备之间传送 LEAF 以及用会话密钥 KS 加密明文消息 hello 的过程。图中未显示初始向量。

在双向通信(如电话)中,通信每一方的安全设备都需传送一个 IV 和由其设备芯片计算出的 LEAF。然后,两个设备使用同一会话密钥 KS 来加密传送给通信对方的消息,并解密由对方传回的消息。

4. 法律实施存取

政府机构在进行犯罪调查时,为了监听被调查者的通信,首先必须取得法院的许可证书,并将许可证书出示给通信服务的提供者(电信部门),并从电信部门租用线路用来截取被监听者的通信。如果被监听者的通信是经过加密的,则被截获的通信首先通过一个政

府控制的解密设备,如图 5-11 所示,其中 D 表示解密。解密设备可识别由托管芯片加密的通信,取出 LEAF 和 IV,并使用族密钥 KF 解密 LEAF 以取出芯片识别符 UID 和加密的会话密钥 $E_{KU}(KS)$ 。

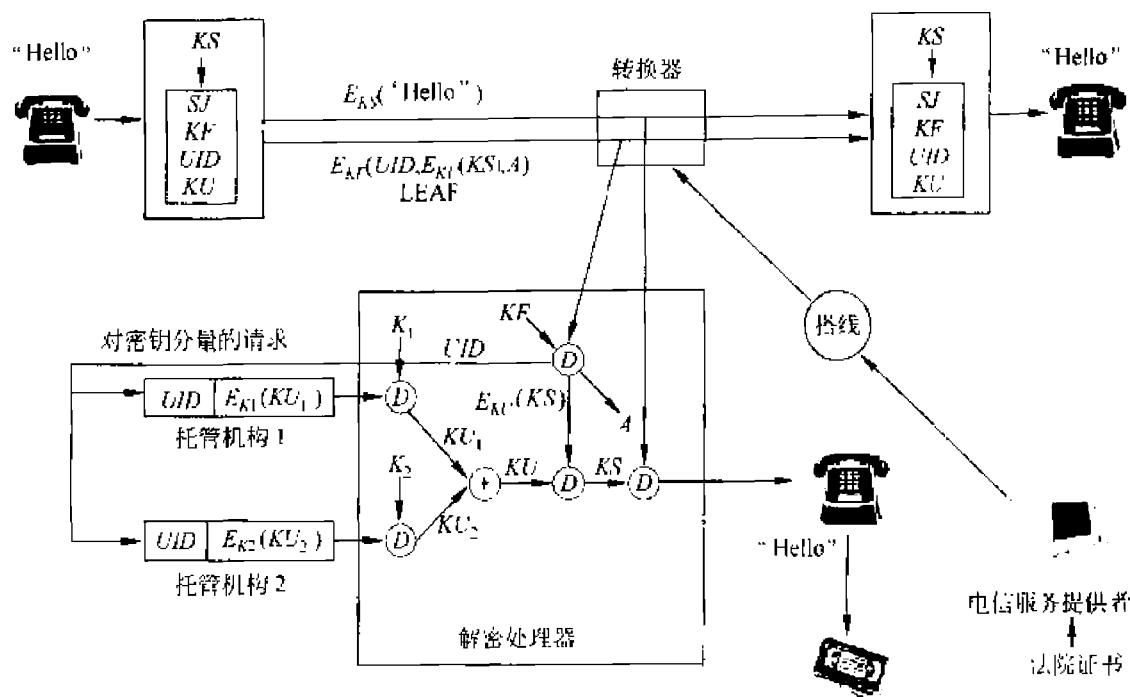


图 5-11 对加密通信的法律实施存取过程

政府机构将芯片识别符 UID 、法院许可监听的许可证书、解密设备的顺序号以及政府机构对该芯片的单元密钥分量的要求一起给托管机构。托管机构在收到并验证政府机构传送的内容后,将被加密的单元密钥分量 $E_{K1}(KU_1)$ 和 $E_{K2}(KU_2)$ 传送给政府机构的解密设备,解密设备分别使用加密密钥 K_1 和 K_2 解密 $E_{K1}(KU_1)$ 和 $E_{K2}(KU_2)$ 以得到 KU_1 、 KU_2 ,求它们的异或 $KU_1 \text{ XOR } KU_2$,即为单元密钥 KU 。由单元密钥 KU 解密 $E_{KU}(KS)$,得被调查者的会话密钥 KS 。最后解密设备使用 KS 解密被调查者的通信。为了实现解密,解密设备在初始化阶段,应安装族密钥 KF 和密钥加密密钥 K_1 、 K_2 。

托管机构在传送加密的密钥分量时,也传送监听的截止时间。因此解密设备的设计应使得它到截止时间后,可自动销毁芯片单元密钥及用于得到单元密钥的所有信息。同时,因为每一次新的会话用一新的会话密钥加密,所以解密设备在监听的截止时间之前,在截获调查者新的会话时,可不经托管机构而直接从 LEAF 中提取并解密会话密钥。因此,除在得到密钥时可有一个时间延迟外,对被截获通信的解密也可在监听的有效期内有一个时间延迟。这种时间延迟对有些案情极为重要,如监听进行绑架的犯罪分子或监

听有计划的恐怖活动。

因为被调查的通信双方使用相同的会话密钥,所以解密设备不需要对通信双方都取出 LEAF 及芯片单元密钥,解密设备只需取出被调查者一方的 LEAF 及芯片单元密钥。如果某人想监听他人的通信,他必须首先能够截获他人的通信,然后必须有一个解密设备和两个经过加密的芯片单元密钥分量。因为制造解密设备必须知道保密算法、族密钥 KF 和密钥加密密钥 K_1 、 K_2 ,任何未经授权的人,都不可能私自制造出解密设备,因此无法获得对他人的监听。

5.3.2 密钥托管密码体制的组成成分

EFS 提出以后,密钥托管密码体制受到了普遍关注,已提出了各种类型的密钥托管密码体制,包括软件实现的、硬件实现的、有多个委托人的、防用户欺诈的、防委托人欺诈的等。密钥托管密码体制从逻辑上可分为 3 个主要部分:用户安全成分 USC(user security component)、密钥托管成分 KEC(key escrow component)和数据恢复成分 DRC(data recovery component)。三者的关系如图 5-12 所示,USC 用密钥 KS 加密明文数据,并且在传送密文时,一起传送一个数据恢复域 DRF(data recovery field)。DRC 使用包含在 DRF 中的信息及由 KEC 提供的信息恢复明文。

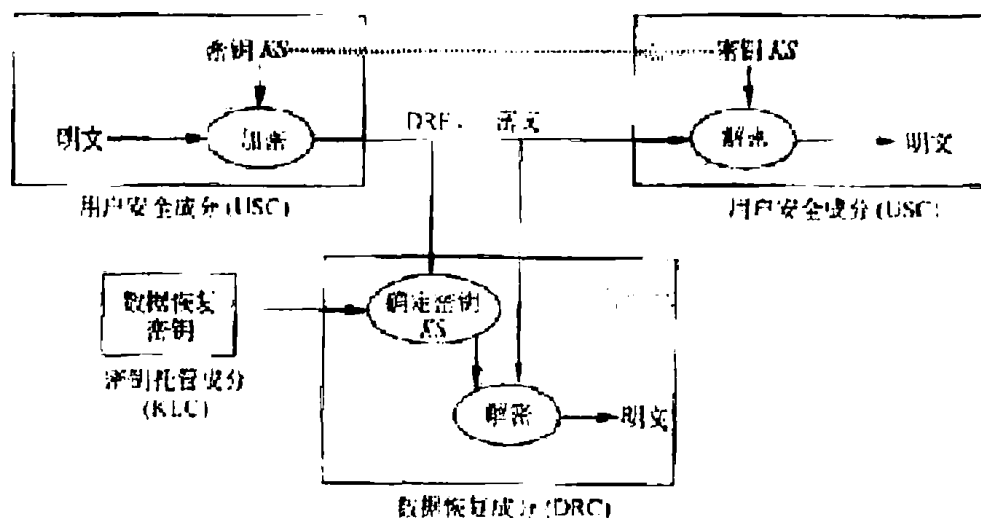


图 5-12 密钥托管密码体制的组成成分

用户安全成分 USC 是提供数据加解密能力以及支持密钥托管功能的硬件设备或软件程序。USC 可用于通信和数据存储的密钥托管,通信情况包括电话通信、电子邮件及其他一些类型的通信,由法律实施部门在获得法院对通信的监听许可后执行对突发事件的解密。数据的存储包括简单的数据文件和一般的存储内容,突发解密由数据的所有者

在密钥丢失或损坏时进行,或者由法律实施部门在获得法院许可证书后对计算机文件进行。USC 使用的加密算法可以是保密的、专用的,也可以是公钥算法。

密钥托管成分 KEC 用于存储所有的数据恢复密钥,通过向 DRC 提供所需的数据和服务以支持 DRC。KEC 可以作为密钥管理系统的一部分,密钥管理系统可以是单一的密钥管理系统(如密钥分配中心),也可以是公钥基础设施。如果是公钥基础设施,托管代理机构可作为公钥证书机构。托管代理机构也称为可信赖的第三方,负责操作 KEC,可能需要在密钥托管中心注册。密钥托管中心的作用是协调托管代理机构的操作或担当 USC 或 DRC 的联系点。

数据恢复成分 DRC 是由 KEC 提供的用于通过密文及 DRF 中的信息获得明文的算法、协议和仪器。它仅在执行指定的已授权的恢复数据时使用。要想恢复数据,DRC 必须获得数据加密密钥,而要获得数据加密密钥则必须使用与收发双方或其中一方相联系的数据恢复密钥。如果只能得到发送方托管机构所持有的密钥,DRC 还必须获得向某一特定用户传送消息的每一方的被托管数据,此时可能无法执行实时解密,尤其是在各方位于不同的国家并使用不同的托管代理机构时。如果 DRC 只能得到收方托管机构所持有的密钥,则对从某一特定用户发出的所有消息也可能无法实时解密。如果能够使用托管代理机构所持有的密钥恢复数据,那么 DRC 一旦获得某一特定 USC 所使用的密钥,就可对这一 USC 发出的消息或发往这一 USC 的消息实时解密。对两方同时通信(如电话通信)的情况,如果会话双方使用相同的数据加密密钥,系统就可实时地恢复加密数据。

5.4 随机数的产生

随机数在密码学中起着重要的作用。这一节首先介绍随机数在密码学中的作用,然后介绍产生随机数的一些方法。

5.4.1 随机数的使用

很多密码算法都需使用随机数,例如:

- 相互认证,如在图 5-1、图 5-2、图 5-4 和图 5-7 所示的密钥分配中,都使用了一次性随机数来防止重放攻击。
- 会话密钥的产生,用随机数作为会话密钥。
- 公钥密码算法中密钥的产生,用随机数作为公钥密码算法中的密钥,如图 5-5;或以随机数来产生公钥密码算法中的密钥,如图 5-8。

在随机数的上述应用中,都要求随机数序列满足随机性和不可预测性。

1. 随机性

以下两个准则常用来保障数列的随机性:

- ① 均匀分布 数列中每个数出现的频率应相等或近似相等。
- ② 独立性 数列中任意一数都不能由其他数推出。

数列是否满足均匀分布可通过检测得出,而是否满足独立性则无法检测。然而有很多检测方法能证明数列不满足独立性,因此通常检测数列是否满足独立性的方法是在对数列进行了足够多次检测后都不能证明不满足独立性,就可比较有把握地相信该数列满足独立性。

在设计密码算法时,经常使用似乎是随机的数列,称为伪随机数列,例如 RSA 算法中素数的产生。一般来说,决定一个大数 N 是否为素数是很困难的。最原始的方法是用小于 \sqrt{N} 的每个数去除 N ,如果 N 很大,比如 10^{150} ,这一方法则超出了人类的分析能力和计算能力。很多有效的算法是通过使用随机选择的整数序列作为相对简单计算的输入,可检测一个数的素性。如果随机选择的序列足够长(当然,远小于 $\sqrt{10^{150}}$),就可比较肯定地得出这个数的素性。这种方法称为随机化,在设计密码算法时经常使用。

2. 不可预测性

在诸如相互认证和会话密钥的产生等应用中,不仅要求数列具有随机性而且要求对数列中以后的数是不可预测的。对于真随机数列来说,数列中每个数都独立于其他数,因此是不可预测的。对于伪随机数来说,就需要特别注意防止敌手从数列前边的数预测出后边的数。

5.4.2 随机数源

真随机数很难获得,物理噪声产生器,如离子辐射脉冲检测器、气体放电管、漏电容等都可作为随机数源,但在网络安全系统中很少采用,一方面是因为数的随机性和精度不够,另一方面这些设备又很难连接到网络系统中。

一种方法是将高质量的随机数作为随机数库编辑成书,供用户使用。然而与网络安全对随机数巨大的需求相比,这种方式提供的随机数数目非常有限。再者,虽然这时的随机数的确可被证明具有随机性,但由于敌手也能得到这个随机数源,面难以保证随机数的不可预测性。

因此网络安全中所需的随机数都借助于安全的密码算法来产生。但由于算法是确定性的,因此产生的数列不是随机的。然而如果算法设计得好,产生的数列就能通过各种随机性检验,这种数就是伪随机数。

5.4.3 伪随机数产生器

最为广泛使用的伪随机数产生器是线性同余算法。算法有 4 个参数:模数 $m(m > 0)$, 乘数 $a(0 \leq a < m)$, 增量 $c(0 \leq c < m)$, 初值即种子 $X_0(0 \leq X_0 < m)$; 由以下迭代公式得到随机数数列 $\{X_n\}$:

$$X_{n+1} = (aX_n + c) \bmod m$$

如果 m, a, c, X_0 都为整数, 则产生的随机数数列 $\{X_n\}$ 也都是整数, 且 $0 \leq X_n < m$ 。

a, c 和 m 的取值是产生高质量随机数的关键。例如取 $a=c=1$, 则结果数列中每一个数都是前一个数增 1, 结果显然不能令人满意。如果 $a=7, c=0, m=32, X_0=1$ 则产生的数列为 $\{7, 17, 23, 1, 7, \dots\}$, 在 32 个可能值中只有 4 个出现, 数列的周期为 4, 因此结果仍不能令人满意。如果取 $a=7$, 其他值不变, 则产生的数列为 $\{1, 5, 25, 29, 17, 21, 9, 13, 1, \dots\}$, 周期增加到 8。

为使随机数数列的周期尽可能大, m 应尽可能大。普遍原则是选 m 接近等于计算机能表示的最大整数, 如接近或等于 2^{31} 。

评价线性同余算法的性能有以下 3 个标准:

- ① 迭代函数应是整周期的, 即数列中的数在重复之前应产生出 0 到 m 之间的所有数。
- ② 产生的数列看上去应是随机的。因为数列是确定性产生的, 因此不可能是随机的, 但可用各种统计检测来评价数列具有多少随机性。
- ③ 迭代函数能有效地利用 32 位运算实现。

通过精心选取 a, c 和 m , 可使以上 3 个标准得以满足。对第 3 条来说, 为了方便 32 位运算的实现, m 可取为 $2^{31} - 1$ 。对第 1 条来说, 如果 m 为素数 ($2^{31} - 1$ 即为素数) 且 $c=0$, 则当 a 是 m 的一个本原根, 即满足

$$\begin{cases} a^n \bmod m \neq 1 & n = 1, 2, \dots, m-2 \\ a^{m-1} \bmod m = 1 \end{cases}$$

时, 产生的数列是整周期的。例如, $a=7^5=16807$ 即为 $m=2^{31}-1$ 的一个本原根, 由此得到的随机数产生器 $X_{n+1} = (aX_n) \bmod (2^{31}-1)$ 已被广泛应用, 而且与其他产生器相比, 经历过更多的检验, 这种产生器常用于统计和模拟工作。

线性同余算法的强度在于如果将乘数和模数选择得好, 则产生的数列和从 $1, 2, \dots, m-1$ 中随机选取的数列是不可区分的。但是除了初值 X_0 的选取具有随机性外, 算法本身并不具有随机性, 因为 X_0 选定后, 以后的数就被确定性地产生了。这个性质可用于对该算法的密码分析, 如果敌手知道正在使用线性同余算法并知道算法的参数, 则一旦获得数列中的一个数, 就可得到以后的所有数。甚至如果敌手只知道正在使用线性同余算法

以及产生的数列中极少一部分,就足以确定出算法的参数。假定敌手能确定 X_0 、 X_1 、 X_2 、 X_3 ,就可通过以下方程组

$$X_1 = (aX_0 + c) \bmod m$$

$$X_2 = (aX_1 + c) \bmod m$$

$$X_3 = (aX_2 + c) \bmod m$$

解出 a 、 c 和 m 。

改进的方法是利用系统时钟修改随机数数列。一种方法是每当产生 N 个数后,就利用当前的时钟值模 m 后作为新种子。另一种方法是直接将当前的时钟值加到每个随机数上(模 m 加)。

下面考虑随机数产生器 $X_{n+1} = (aX_n) \bmod m$ 的实现。

算法实现时,需解决的主要问题是溢出,为此将迭代关系写为 $X = f(X) = (aX) \bmod m$,对算法做如下修改:

设 $m = aq + r$, 其中 $q = \lfloor m/a \rfloor$, $r = m \bmod a$ 。

$$f(X) = (aX) \bmod m = aX - m \cdot \lfloor aX/m \rfloor \Leftrightarrow f(X) = p_1(X) + m \cdot p_2(X)$$

其中 $p_1(X) = a \cdot (X \bmod q) - r \cdot \lfloor X/q \rfloor$, 满足 $|p_1(X)| \leq m-1$,

$$p_2(X) = \lfloor X/q \rfloor - \lfloor aX/m \rfloor, \text{ 满足 } p_2(X) = 0 \text{ 或 } 1,$$

所以

$$f(x) = \begin{cases} p_1(X) & \text{如果 } p_2(X) = 0 (\text{此时 } 1 \leq p_1(X) \leq m-1) \\ p_1(X) + m & \text{如果 } p_2(X) = 1 (\text{此时 } -(m-1) \leq p_1(X) \leq -1) \end{cases}$$

算法进一步修改如下:

设 $p_1(X) = a \cdot (X \bmod q)$, $p_2(X) = r \cdot \lfloor X/q \rfloor$, 那么

$$f(x) = \begin{cases} p_1(X) - p_2(X) & \text{如果 } p_1(X) > p_2(X) \\ p_1(X) + (m - p_2(X)) & \text{否则如果 } p_1(x) \leq p_2(x) \end{cases}$$

5.4.4 基于密码算法的随机数产生器

为了产生密码中可用的随机数,可使用加密算法。本小节介绍 3 个具有代表性的例子。

1. 循环加密

图 5-13 是通过循环加密由主密钥产生会话密钥的示意图,其中周期为 N 的计数器用来为加密算法产生输入。例如要想产生 56 比特的 DES 密钥,可使用周期为 2^{36} 的计数

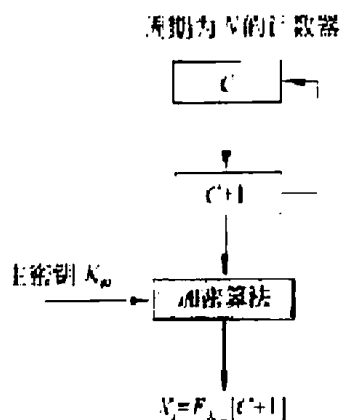


图 5-13 循环加密产生伪随机数

器,每产生一个密钥后,计数器加 1。因此本方案产生的伪随机数以整周期循环,输出数列 X_0, X_1, \dots, X_{N-1} 中的每个值都是由计数器中的不同值得到,因此 $X_0 \neq X_1 \neq \dots \neq X_{N-1}$ 。又因为主密钥是受到保护的,所以知道前面的密钥值想得到后面的密钥在计算上是不可行的。

为进一步增加算法的强度,可用整周期的伪随机数产生器代替计数器作为方案中加密算法的输入。

2. DES 的输出反馈(OFB)模式

DES 的 OFB 模式(如图 3-13 所示)能用来产生密钥并能用于流加密。加密算法的每一步输出都为 64 比特,其中最左边的 j 个比特被反馈回加密算法。因此加密算法的一个个 64 比特输出就构成了一个具有很好统计特性的伪随机数序列。同样,如此产生的会话密钥可通过对主密钥的保护而得以保护。

3. ANSI X9.17 的伪随机数产生器

ANSI X9.17 的伪随机数产生器是密码强度最高的伪随机数产生器之一,已在包括 PGP 等许多应用过程中被采纳。

图 5-14 是 ANSI X9.17 伪随机数产生器的框图,产生器有 3 个组成部分:

① 输入 输入为两个 64 比特的伪随机数,其中 DT_i 表示当前的日期和时间,每产生一个数 R_i 后, DT_i 都更新一次; V_i 是产生第 i 个随机数时的种子,其初值可任意设定,以后每次自动更新。

② 密钥 产生器用了 3 次三重 DES 加密,3 次加密使用相同的两个 56 比特的密钥 K_1 和 K_2 ,这两个密钥必须保密且不能用作他用。

③ 输出 输出为一个 64 比特的伪随机数 R_i 和一个 64 比特的新种子 V_{i+1} ,其中,

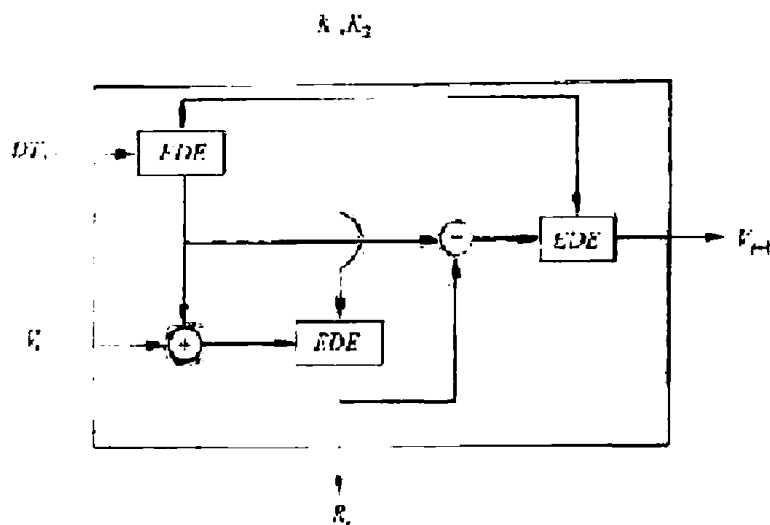


图 5-14 ANSI X9.17 伪随机数产生器

$$R_i = EDE_{K_1, K_2}[V_i \oplus EDE_{K_1, K_2}[DT,]]$$

$$V_{i+1} = EDE_{K_1, K_2}[R_i \oplus EDE_{K_1, K_2}[DT,]]$$

EDE 表示两个密钥的三重 DES。

本方案具有非常高的密码强度,这是因为采用了 112 比特长的密钥和 9 个 DES 加密,同时还由于算法由两个伪随机数输入驱动,一个是当前的日期和时间,另一个是算法上次产生的新种子。而且即使某次产生的随机数 R_i 泄露了,但由于 R_i 又经一次 EDE 加密才产生新种子 V_{i+1} ,所以别人即使得到 R_i 也得不到 V_{i+1} ,从而得不到新随机数 R_{i+1} 。

5.4.5 BBS 产生器

BBS(blum-blum-shub)产生器是已经过证明的密码强度最强的伪随机数产生器,它的整个过程如下:

首先,选择两个大素数 p, q , 满足 $p \equiv q \equiv 3 \pmod{4}$, 令 $n = p \times q$ 。再选一随机数 s , 使得 s 与 n 互素。然后按以下算法产生比特序列 $\{B_i\}$:

$$X_0 = s^2 \bmod n$$

for $i = 1$ to ∞ do {

$$X_i = X_{i-1}^2 \bmod n;$$

$$B_i = X_i \bmod 2 \}$$

即在每次循环中取 X_i 的最低有效位。

例如: $n = 192649 = 383 \times 503$, 种子 $s = 101355$, 结果由表 5-1 给出。

表 5-1 BBS 产生器的一个例子

i	X_i	B_i	i	X_i	B_i
0	20749		11	137922	0
1	143135	1	12	123175	1
2	177671	1	13	8630	0
3	97048	0	14	114386	0
4	89992	0	15	14863	1
5	174051	1	16	133015	1
6	80649	1	17	106065	1
7	45663	1	18	45870	0
8	69442	0	19	137171	1
9	186894	0	20	48060	0
10	177046	0			

BBS 的安全性基于大整数分解的困难性,它是密码上安全的伪随机比特产生器。如果伪随机比特产生器能通过下一比特检验,则称之为密码上安全的伪随机比特产生器,具体定义为:以伪随机比特产生器的输出序列的前 k 个比特作为输入,如果不存在多项式时间算法,能以大于 $1/2$ 的概率预测第 $k+1$ 个比特。换句话说,已知一个序列的前 k 个比特,不存在实际可行的算法能以大于 $1/2$ 的概率预测下一比特是 0 还是 1。

5.5 秘密分割

5.5.1 秘密分割门限方案

在导弹控制发射、重要场所通行检验等情况下,通常必须由两人或多人同时参与才能生效,这时都需要将秘密分给多人掌管,并且必须有一定人数的掌管秘密的人同时到场才能恢复这一秘密。

由此,引入门限方案(threshold schemes)的一般概念。

定义 5-1 设秘密 s 被分成 n 个部分信息,每一部分信息称为一个子密钥或影子,由一个参与者持有,使得:

- ① 由 k 个或多于 k 个参与者所持有的部分信息可重构 s 。
- ② 由少于 k 个参与者所持有的部分信息则无法重构 s 。

则称这种方案为 (k, n) -秘密分割门限方案, k 称为方案的门限值。

如果一个参与者或一组未经授权的参与者在猜测秘密 s 时,并不比局外人猜秘密时有优势,即③由少于 k 个参与者所持有的部分信息得不到秘密 s 的任何信息。

则称这个方案是完善的,即 (k, n) -秘密分割门限方案是完善的。

下面介绍最具代表性的两个秘密分割门限方案。

5.5.2 Shamir 门限方案

Shamir 门限方案是基于多项式的 Lagrange 插值公式的。

设 $\{(x_1, y_1), \dots, (x_k, y_k)\}$ 是平面上 k 个点构成的点集,其中 $x_i (i=1, \dots, k)$ 均不相同,那么在平面上存在一个惟一的 $k-1$ 次多项式 $f(x)$ 通过这 k 个点。若把密钥 s 取作 $f(0)$, n 个子密钥取作 $f(x_i) (i=1, 2, \dots, n)$, 那么利用其中的任意 k 个子密钥可重构 $f(x)$, 从而可得密钥 s 。

这种门限方案也可按如下更一般的方式来构造。设 $GF(q)$ 是一有限域,其中 q 是一大素数,满足 $q \geq n+1$, 秘密 s 是在 $GF(q) \setminus \{0\}$ 上均匀选取的一个随机数,表示为 $s \in {}_R GF(q) \setminus \{0\}$ 。 $k-1$ 个系数 a_1, a_2, \dots, a_{k-1} 的选取也满足 $a_i \in {}_R GF(q) \setminus \{0\} (i=1, 2, \dots, k-1)$ 。在 $GF(q)$ 上构造一个 $k-1$ 次多项式 $f(x) = a_0 + a_1 x + \dots + a_{k-1} x^{k-1}$ 。

n 个参与者记为 P_1, P_2, \dots, P_n , P_i 分配到的子密钥为 $f(i)$ 。如果任意 k 个参与者 $P_{i_1}, \dots, P_{i_k} (1 \leq i_1 < i_2 < \dots < i_k \leq n)$ 要想得到秘密 s , 可使用 $\{(i_l, f(i_l)) | l=1, \dots, k\}$ 构造如下的线性方程组:

$$\begin{cases} a_0 + a_1(i_1) + \dots + a_{k-1}(i_1)^{k-1} = f(i_1) \\ a_0 + a_1(i_2) + \dots + a_{k-1}(i_2)^{k-1} = f(i_2) \\ \dots\dots\dots \\ a_0 + a_1(i_k) + \dots + a_{k-1}(i_k)^{k-1} = f(i_k) \end{cases} \quad (5-1)$$

因为 $i_l (1 \leq l \leq k)$ 均不相同,所以可由 Lagrange 插值公式构造如下的多项式:

$$f(x) = \sum_{j=1}^k f(i_j) \prod_{\substack{l=1 \\ l \neq j}}^k \frac{(x - i_l)}{(i_j - i_l)} \pmod{q}$$

从而可得秘密 $s = f(0)$ 。

然而参与者仅需知道 $f(x)$ 的常数项 $f(0)$ 而无需知道整个多项式 $f(x)$, 所以仅需以下表达式就可求出 s :

$$s = (-1)^{k-1} \sum_{j=1}^k f(i_j) \prod_{\substack{l=1 \\ l \neq j}}^k \frac{i_l}{(i_j - i_l)} \pmod{q}$$

如果 $k-1$ 个参与者想获得秘密 s , 他们可构造出由 $k-1$ 个方程构成的线性方程组, 其中有 k 个未知量。对 $GF(q)$ 中的任一值 s_0 , 可设 $f(0) = s_0$, 这样可得第 k 个方程。

并由 Lagrange 插值公式得出 $f(x)$ 。因此对每一 $s_0 \in GF(q)$ 都有一个惟一的多项式满足式(5-1), 所以已知 $k-1$ 个子密钥得不到关于秘密 s 的任何信息, 因此这个方案是完善的。

【例 5-1】 设 $k=3, n=5, q=19, s=11$, 随机选取 $a_1=2, a_2=7$, 得多项式为

$$f(x) = (7x^2 + 2x + 11) \bmod 19$$

分别计算

$$f(1) = (7 + 2 + 11) \bmod 19 = 20 \bmod 19 = 1$$

$$f(2) = (28 + 4 + 11) \bmod 19 = 43 \bmod 19 = 5$$

$$f(3) = (63 + 6 + 11) \bmod 19 = 80 \bmod 19 = 4$$

$$f(4) = (112 + 8 + 11) \bmod 19 = 131 \bmod 19 = 17$$

$$f(5) = (175 + 10 + 11) \bmod 19 = 196 \bmod 19 = 6$$

得 5 个子密钥。

如果知道其中的 3 个子密钥 $f(2)=5, f(3)=4, f(5)=6$, 就可按以下方式重构 $f(x)$:

$$\begin{aligned} 5 \frac{(x-3)(x-5)}{(2-3)(2-5)} &= 5 \frac{(x-3)(x-5)}{(-1)(-3)} = 5 \frac{(x-3)(x-5)}{3} \\ &= 5 \cdot (3^{-1} \bmod 19) \cdot (x-3)(x-5) \\ &= 5 \cdot 13 \cdot (x-3)(x-5) = 65(x-3)(x-5) \\ 4 \frac{(x-2)(x-5)}{(3-2)(3-5)} &= 4 \frac{(x-2)(x-5)}{(1)(-2)} \\ &= 4 \frac{(x-2)(x-5)}{-2} = 4 \cdot ((-2)^{-1} \bmod 19) \cdot (x-2)(x-5) \\ &= 4 \cdot 9 \cdot (x-2)(x-5) = 36(x-2)(x-5) \\ 6 \frac{(x-2)(x-3)}{(5-2)(5-3)} &= 6 \frac{(x-2)(x-3)}{(3)(2)} = 6 \frac{(x-2)(x-3)}{6} \\ &= 6 \cdot (6^{-1} \bmod 19) \cdot (x-2)(x-3) \\ &= 6 \cdot 16 \cdot (x-2)(x-3) = 96(x-2)(x-3) \end{aligned}$$

所以

$$\begin{aligned} f(x) &= [65(x-3)(x-5) + 36(x-2)(x-5) + 96(x-2)(x-3)] \bmod 19 \\ &= [8(x-3)(x-5) + 17(x-2)(x-5) + (x-2)(x-3)] \bmod 19 \\ &= (26x^2 - 188x + 296) \bmod 19 \\ &= 7x^2 + 2x + 11 \end{aligned}$$

从而得秘密为 $s=11$ 。

5.5.3 Asmuth-Bloom 门限方案

首先选取一大素数 q , 正整数 s , 以及 n 个严格递增的数 m_1, m_2, \dots, m_n , 满足

$$\textcircled{1} \quad q > s.$$

$$\textcircled{2} \quad (m_i, m_j) = 1 (\forall i, j, i \neq j).$$

$$\textcircled{3} \quad (q, m_i) = 1 (\forall i).$$

$$\textcircled{4} \quad N = \prod_{i=1}^k m_i > q \prod_{i=1}^{k-1} m_{n-i+1}.$$

以 s 作为秘密数据, 条件 $\textcircled{1}$ 指出, 秘密数据小于 q , 条件 $\textcircled{4}$ 指出, N/q 大于任取的 $k-1$ 个不同的 m_i 之积。进而, 随机选取 A , 满足 $0 \leq A \leq [N/q]-1$, 并公布 q 和 A 。

求 $y = s + Aq$ (由上知 $y < N$), 和 $y_i \equiv y \pmod{m_i} (i=1, \dots, n)$, (m_i, y_i) 即为一个子密钥, 集合 $\{(m_i, y_i)\}_{i=1}^k$ 即构成了一个 (k, n) 门限方案。

这是因为, 当 k 个参与者 (记为 i_1, i_2, \dots, i_k) 提供出自己的子密钥时, 由 $\{(m_{i_j}, y_{i_j})\}_{j=1}^k$ 建立方程组

$$\begin{cases} y_{i_1} \pmod{m_{i_1}} = y \\ y_{i_2} \pmod{m_{i_2}} = y \\ \dots \\ y_{i_k} \pmod{m_{i_k}} = y \end{cases}$$

根据中国剩余定理可求得

$$y \equiv y' \pmod{N'}$$

$$\text{式中 } N' = \prod_{j=1}^k m_{i_j} \geq N.$$

因为 $y < N \leq N'$, 所以 $y = y'$ 是惟一的, 再由 $y - Aq$ 即得秘密数据 s 。

另一方面, 若仅有 $k-1$ 个参与者提供出自己的子密钥 (m_i, y_i) , 则只能求得 $y'' \equiv y \pmod{N''}$, 式中 $N'' = \prod_{j=1}^{k-1} m_{i_j}$,

由条件 $\textcircled{4}$ 得 $N'' < N/q$, 即 $N/N'' > q$ 。

由于 $N/N'' > q$, $(N'', q) = 1$, 故集合 $\{[x] \mid [x] = \{x; x \pmod{N''} = y''\}\}$ 将覆盖所有的模 q 同余类。或者更简单地说, 至少以下整数 $y'' + aN'' (0 \leq a < q)$ 都是 y 的可能取值。因此无法确定 y 。

【例 5-2】 设 $k=2, n=3, q=7, s=4, m_1=9, m_2=11, m_3=13$, 则

$$N = m_1 m_2 = 99 > 91 = 7 \cdot 13 = q \cdot m_3$$

在 $\left[0, \left\lceil \frac{99}{7} \right\rceil - 1\right] = [0, 13]$ 之间随机地取 $A = 10$, 求 $y = s + Aq = 4 + 10 \times 7 = 74$, $y_1 = y \bmod m_1 = 74 \bmod 9 = 2$, $y_2 = y \bmod m_2 = 74 \bmod 11 = 8$, $y_3 = y \bmod m_3 = 74 \bmod 13 = 9$ 。

$\{(9, 2), (11, 8), (13, 9)\}$ 即构成 $(2, 3)$ 门限方案。

若知道 $\{(9, 2), (11, 8)\}$, 可建立方程组

$$\begin{cases} 2 \bmod 9 = y \\ 8 \bmod 11 = y \end{cases}$$

解之得 $y = (11 \times 5 \times 2 - 9 \times 5 \times 8) \bmod 99 = 74$, 所以 $s = y - Aq = 74 - 10 \times 7 = 4$ 。

习 题

1. 在公钥体制中, 每一用户 U 都有自己的公开钥 PK_U 和秘密钥 SK_U 。如果任意两个用户 A, B 按以下方式通信, A 发给 B 消息 $(E_{PK_B}(m), A)$, B 收到后, 自动向 A 返回消息 $(E_{PK_A}(m), B)$, 以使 A 知道 B 确实收到报文 m ,

(1) 问用户 C 怎样通过攻击手段获取报文 m ?

(2) 若通信格式变为:

A 发给 B 消息: $E_{PK_B}(E_{SK_A}(m), m, A)$

B 向 A 返回消息 $E_{PK_A}(E_{SK_B}(m), m, B)$

这时的安全性如何? 分析 A, B 这时如何相互认证并传递消息 m 。

2. Diffie-Hellman 密钥交换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。详细分析攻击者如何实施攻击。

3. 在 Diffie-Hellman 密钥交换过程中, 设大素数 $p = 11$, $a = 2$ 是 p 的本原根,

(1) 用户 A 的公开钥 $Y_A = 9$, 求其秘密钥 X_A 。

(2) 设用户 B 的公开钥 $Y_B = 3$, 求 A 和 B 的共享密钥 K 。

4. 线性同余算法 $X_{n+1} = (aX_n) \bmod 2^4$, 问:

(1) 该算法产生的数列的最大周期是多少?

(2) a 的值是多少?

(3) 对种子有何限制?

5. 在 Shamir 秘密分割门限方案中, 设 $k = 3, n = 5, q = 17$, 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秘密数据 s 。

6. 在 Asmuth-Bloom 秘密分割门限方案中, 设 $k = 2, n = 3, q = 5, m_1 = 7, m_2 = 9, m_3 = 11$, 3 个子密钥分别是 6, 3, 4, 求秘密数据 s 。

第 6 章

消息认证和杂凑算法

第 1 章曾介绍过信息安全所面临的基本攻击类型,包括被动攻击(获取消息的内容、业务流分析)和主动攻击(假冒、重放、消息的篡改、业务拒绝)。抗击被动攻击的方法是前面已介绍过的加密,本章介绍的消息认证则是用来抗击主动攻击的。消息认证是一个过程,用以验证接收消息的真实性(的确是由它所声称的实体发来的)和完整性(未被篡改、插入、删除),同时还用于验证消息的顺序性和时间性(未重排、重放、延迟)。除此之外,在考虑信息安全时还需考虑业务的不可否认性,即防止通信双方中的某一方对所传输消息的否认。实现消息的不可否认性可通过数字签字,数字签字也是一种认证技术,它也可用于抗击主动攻击。

消息认证机制和数字签字机制都需有产生认证符的基本功能,这一基本功能又作为认证协议的一个组成部分。认证符是用于认证消息的数值,它的产生方法又分为消息认证码 MAC(message authentication code)和杂凑函数(hash function)两大类,下面分别介绍。

6.1 消息认证码

6.1.1 消息认证码的定义及使用方式

消息认证码是指消息被一密钥控制的公开函数作用后产生的、用作认证符的、固定长度的数值,也称为密码校验和。此时需要通信双方 A 和 B 共享一密钥 K。设 A 欲发送给 B 的消息是 M, A 首先计算 $MAC = C_K(M)$, 其中 $C_K(\cdot)$ 是密钥控制的公开函数, 然后向 B 发送 $M \parallel MAC$, B 收到后做与 A 相同的计算, 求得一新 MAC, 并与收到的 MAC 做比较, 如图 6-1(a) 所示。

如果仅收发双方知道 K, 且 B 计算得到的 MAC 与接收到的 MAC 一致, 则这一系统就实现了以下功能:

① 接收方相信发送方发来的消息未被篡改, 这是因为攻击者不知道密钥, 所以不能够在篡改消息后相应地篡改 MAC, 而如果仅篡改消息, 则接收方计算的新 MAC 将与收

到的 MAC 不同。

② 接收方相信发送方不是冒充的,这是因为除收发双方外再无其他人知道密钥,因此其他人不可能对自己发送的消息计算出正确的 MAC。

MAC 函数与加密算法类似,不同之处为 MAC 函数不必是可逆的,因此与加密算法相比更不易被攻破。

上述过程中,由于消息本身在发送过程中是明文形式,所以这一过程只提供认证性而未提供保密性。为提供保密性可在 MAC 函数以后(如图 6-1(b))或以前(如图 6-1(c))进行一次加密,而且加密密钥也需被收发双方共享。在图 6-1(b)中, M 与 MAC 链接后再被整体加密,在图 6-1(c)中, M 先被加密再与 MAC 链接后发送。通常希望直接对明文进行认证,因此图 6-1(b)所示的使用方式更为常用。

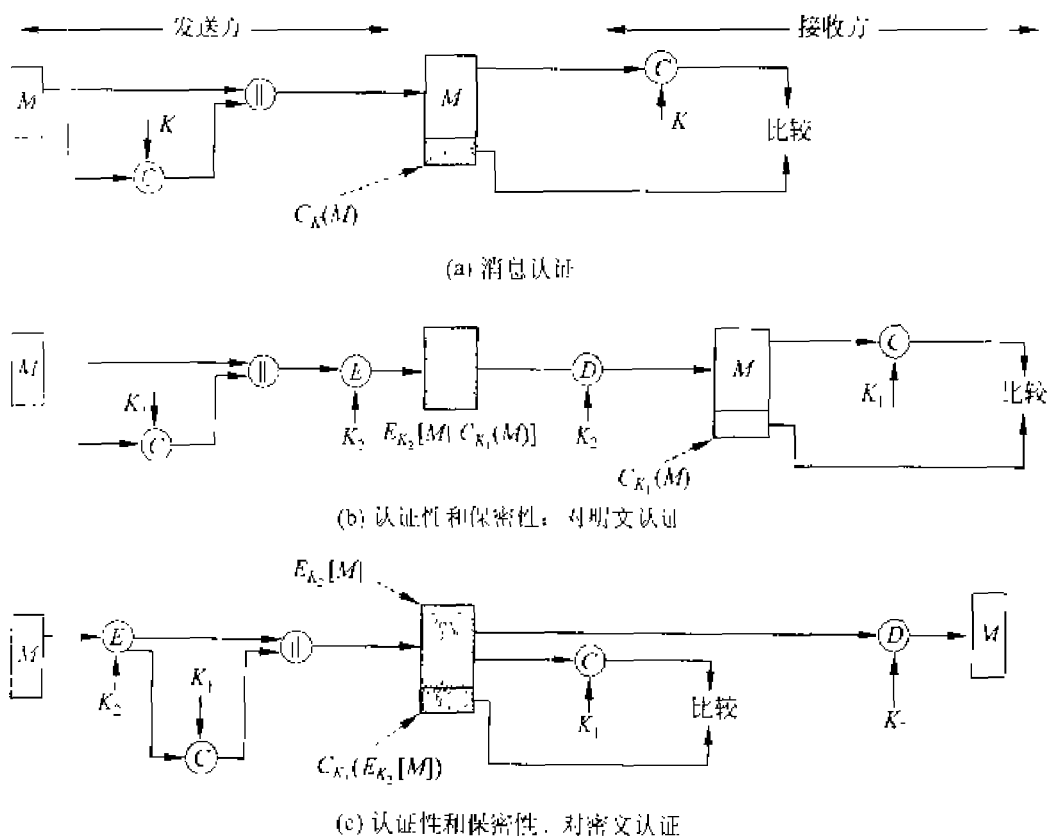


图 6-1 MAC 的基本使用方式

6.1.2 产生 MAC 的函数应满足的要求

使用加密算法(单钥算法或公钥算法)加密消息时,其安全性一般取决于密钥的长度。如果加密算法没有弱点,则敌手只能使用穷搜索攻击以测试所有可能的密钥。如果密钥

长为 k 比特,则穷搜索攻击平均将进行 2^{k-1} 个测试。特别地,对惟密文攻击来说,敌手如果知道密文 C ,则将对所有可能的密钥值 K_i 执行解密运算 $P_i = D_{K_i}(C)$,直到得到有意义的明文。

对 MAC 来说,由于产生 MAC 的函数一般都为多到一映射,如果产生 n 比特的 MAC,则函数的取值范围即为 2^n 个可能的 MAC,函数输入的可能的消息个数 $N \gg 2^n$,而且如果函数所用的密钥为 k 比特,则可能的密钥个数为 2^k 。如果系统不考虑保密性,即敌手能获取明文消息和相应的 MAC,那么在这种情况下要考虑敌手使用穷搜索攻击来获取产生 MAC 的函数所使用的密钥。假定 $k > n$,且敌手已得到 M_1 和 MAC_1 ,其中 $MAC_1 = C_{K_1}(M_1)$,敌手对所有可能的密钥值 K_i 求 $MAC_i = C_{K_i}(M_1)$,直到找到某个 K_i 使得 $MAC_i = MAC_1$ 。由于不同的密钥个数为 2^k ,因此将产生 2^k 个 MAC,但其中仅有 2^n 个不同,由于 $2^k > 2^n$,所以有很多密钥(平均有 $2^k/2^n = 2^{k-n}$ 个)都可产生出正确的 MAC_1 ,而敌手无法知道进行通信的两个用户用的是哪一个密钥,还必须按以下方式重复上述攻击:

第 1 轮 已知 M_1 、 MAC_1 ,其中 $MAC_1 = C_K(M_1)$ 。对所有 2^k 个可能的密钥计算 $MAC_i = C_{K_i}(M_1)$,得 2^{k-n} 个可能的密钥。

第 2 轮 已知 M_2 、 MAC_2 ,其中 $MAC_2 = C_K(M_2)$ 。对上一轮得到的 2^{k-n} 个可能的密钥计算 $MAC_i = C_{K_i}(M_2)$,得 $2^{k-2 \times n}$ 个可能的密钥。

如此下去,如果 $k = an$,则上述攻击方式平均需要 a 轮。例如,密钥长为 80 比特,MAC 长为 32 比特,则第 1 轮将产生大约 2^{48} 个可能密钥,第 2 轮将产生 2^{16} 个可能的密钥,第 3 轮即可找出正确的密钥。

如果密钥长度小于 MAC 的长度,则第 1 轮就有可能找出正确的密钥,也有可能找出多个可能的密钥,如果是后者,则仍需执行第 2 轮搜索。

所以对消息认证码的穷搜索攻击比对使用相同长度密钥的加密算法的穷搜索攻击的代价还要大。然而有些攻击法却不需要寻找产生 MAC 所使用的密钥。

例如,设 $M = (X_1 \parallel X_2 \parallel \cdots \parallel X_m)$ 是由 64 比特的分组 $X_i (i = 1, \cdots, m)$ 链接得到的,其消息认证码由以下方式得到:

$$\Delta(M) = X_1 \oplus X_2 \oplus \cdots \oplus X_m$$

$$C_K(M) = E_K[\Delta(M)]$$

其中 \oplus 表示异或运算,加密算法是电码本模式的 DES。因此,密钥长为 56 比特,MAC 长为 64 比特,如果敌手得到 $M \parallel C_K(M)$,那么敌手使用穷搜索攻击寻找 K 将需做 2^{56} 次加密。然而敌手还可用以下方式攻击系统:将 X_1 到 X_{m-1} 分别用自己选取的 Y_1 到 Y_{m-1} 替换,求出 $Y_m = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{m-1} \oplus \Delta(M)$,并用 Y_m 替换 X_m 。因此敌手可成功伪造一新消息 $M' = Y_1 \oplus \cdots \oplus Y_m$,且 M' 的 MAC 与原消息 M 的 MAC 相同。

考虑到 MAC 所存在的以上攻击类型,可知它应满足以下要求,其中假定敌手知道函数 C ,但不知道密钥 K :

① 如果敌手得到 M 和 $C_K(M)$,则构造一满足 $C_K(M')=C_K(M)$ 的新消息 M' 在计算上是不可行的。

② $C_K(M)$ 在以下意义下是均匀分布的:随机选取两个消息 $M, M', P_r[C_K(M)=C_K(M')]=2^{-n}$, 其中 n 为 MAC 的长。

③ 若 M' 是 M 的某个变换,即 $M'=f(M)$,例如 f 为插入一个或多个比特,那么 $P_r[C_K(M)=C_K(M')]=2^{-n}$ 。

第 1 个要求是针对上例中的攻击类型的,此要求是说敌手不需要找出密钥 K 而伪造一个与截获的 MAC 相匹配的新消息在计算上是不可行的。第 2 个要求是说敌手如果截获一个 MAC,则伪造一个相匹配的消息的概率为最小。最后一个要求是说函数 C 不应在消息的某个部分或某些比特弱于其他部分或其他比特,否则敌手获得 M 和 MAC 后就有可能修改 M 中弱的部分,从而伪造出一个与原 MAC 相匹配的新消息。

6.1.3 数据认证算法

数据认证算法是最为广泛使用的消息认证码中的一个,已作为 FIPS Publication (FIPS PUB 113) 并被 ANSI 作为 X9.17 标准。

算法基于 CBC 模式的 DES 算法,其初始向量取为零向量。需被认证的数据(消息、记录、文件或程序)被分为 64 比特长的分组 D_1, D_2, \dots, D_N , 其中最后一个分组不够 64 比特的话,可在其右边填充一些 0,然后按以下过程计算数据认证码(见图 6-2):

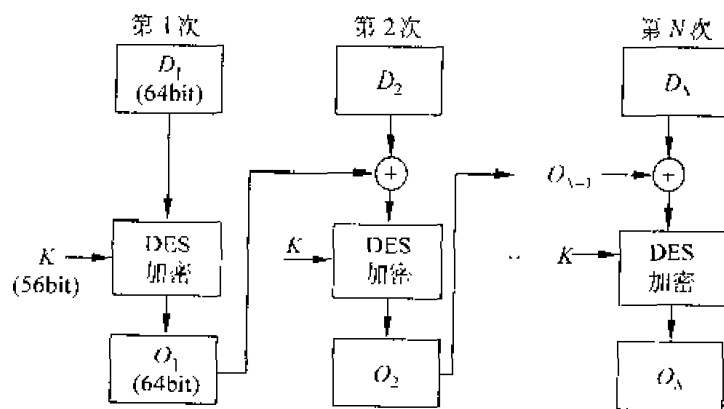


图 6-2 数据认证算法

$$O_1 = E_K(D_1)$$

$$O_2 = E_K(D_2 \oplus O_1)$$

$$\begin{aligned}
 O_3 &= E_K(D_2 \oplus O_2) \\
 &\vdots \\
 O_N &= E_K(D_N \oplus O_{N-1})
 \end{aligned}$$

其中 E 为 DES 加密算法, K 为密钥。

数据认证码或者取为 O_N 或者取为 O_N 的最左 M 个比特, 其中 $16 \leq M \leq 64$ 。

6.2 杂凑函数

6.2.1 杂凑函数的定义及使用方式

杂凑函数 H 是一公开函数, 用于将任意长的消息 M 映射为较短的、固定长度的一个值 $H(M)$, 作为认证符, 称函数值 $H(M)$ 为杂凑值、杂凑码或消息摘要。杂凑码是消息中所有比特的函数, 因此提供了一种错误检测能力, 即改变消息中任何一个比特或几个比特都会使杂凑码发生改变。

图 6-3 表示杂凑函数用来提供消息认证的基本使用方式, 共有以下 6 种:

① 消息与杂凑码链接后用单钥加密算法加密。由于所用密钥仅为收发双方 A、B 共享, 因此可保证消息的确来自 A 并且未被篡改。同时还由于消息和杂凑码都被加密, 这种方式还提供了保密性, 见图 6-3(a)。

② 用单钥加密算法仅对杂凑码加密。这种方式用于不要求保密性的情况下, 可减少处理负担。注意这种方式和图 6-1(a) 的 MAC 结果完全一样, 即将 $E_K[H(M)]$ 看作一个函数, 函数的输入为消息 M 和密钥 K , 输出为固定长度, 见图 6-3(b)。

③ 用公钥加密算法和发送方的秘密钥仅加密杂凑码。和②一样, 这种方式提供认证性, 又由于只有发送方能产生加密的杂凑码, 因此这种方式还对发送方发送的消息提供了数字签字, 事实上这种方式就是数字签字, 见图 6-3(c)。

④ 消息的杂凑值用公钥加密算法和发送方的秘密钥加密后与消息链接, 再对链接后的结果用单钥加密算法加密, 这种方式提供了保密性和数字签字, 见图 6-3(d)。

⑤ 使用这种方式时要求通信双方共享一个秘密值 S , A 计算消息 M 和秘密值 S 链接在一起的杂凑值, 并将此杂凑值附加到 M 后发往 B。因 B 也有 S , 所以可重新计算杂凑值以对消息进行认证。由于秘密值 S 本身未被发送, 敌手无法对截获的消息加以篡改, 也无法产生假消息。这种方式仅提供认证, 见图 6-3(e)。

⑥ 这种方式是在⑤中消息与杂凑值链接以后再增加单钥加密运算, 从而又可提供保密性, 见图 6-3(f)。

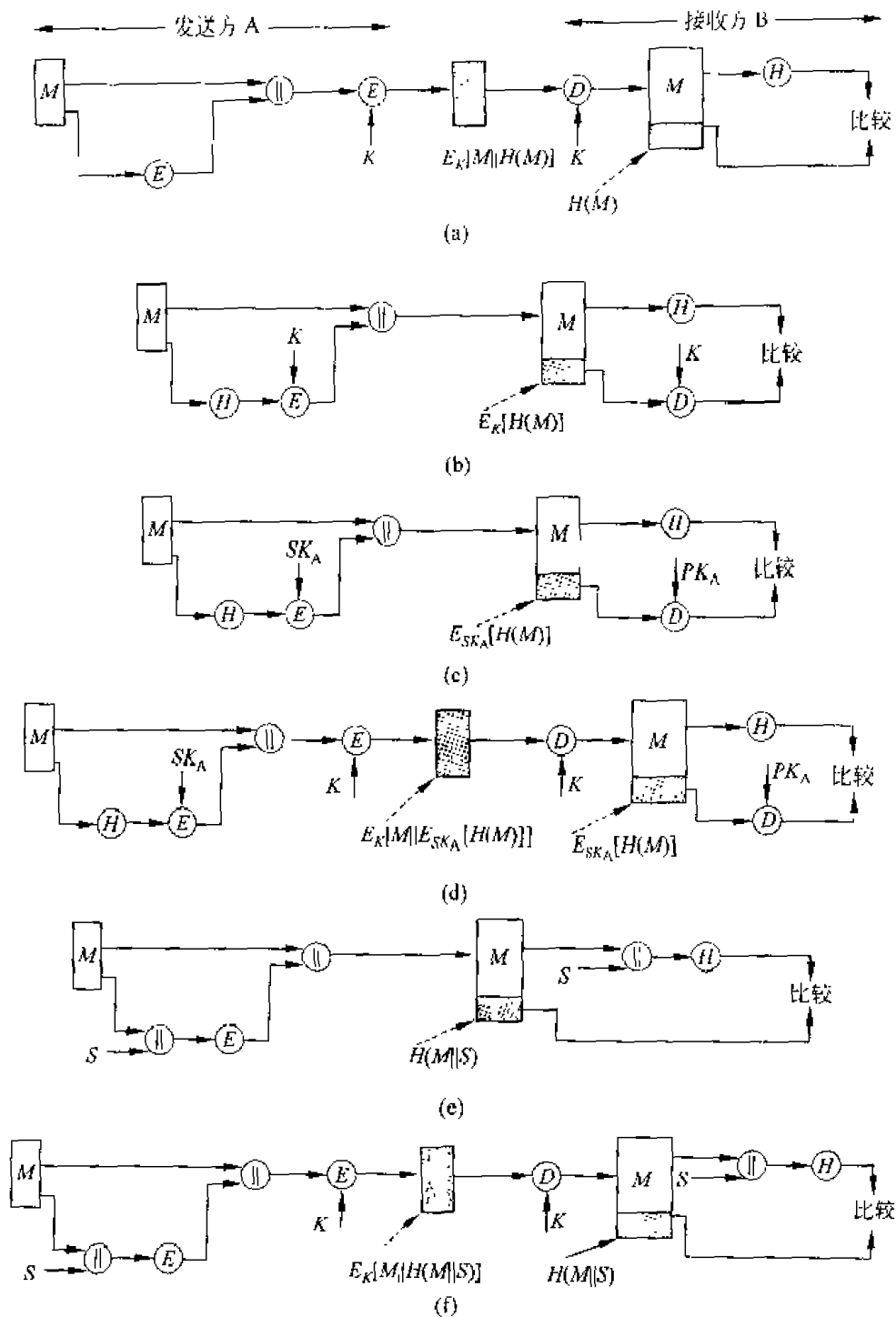


图 6-3 杂凑函数的基本使用方式

由于加密运算的速度较慢,代价较高,而且很多加密算法还受到专利保护,因此在不要求保密性的情况下,方式②和③将比其他方式更具优势。

6.2.2 杂凑函数应满足的条件

杂凑函数的目的是为需认证的数据产生一个“指纹”。为了能够实现对数据的认证,杂凑函数应满足以下条件:

① 函数的输入可以是任意长。

② 函数的输出是固定长。

③ 已知 x , 求 $H(x)$ 较为容易, 可用硬件或软件实现。

④ 已知 h , 求使得 $H(x)=h$ 的 x 在计算上是不可行的, 这一性质称为函数的单向性, 称 $H(x)$ 为单向杂凑函数。

⑤ 已知 x , 找出 $y(y \neq x)$ 使得 $H(y)=H(x)$ 在计算上是不可行的。

如果单向杂凑函数满足这一性质, 则称其为弱单向杂凑函数。

⑥ 找出任意两个不同的输入 x, y , 使得 $H(y)=H(x)$ 在计算上是不可行的。

如果单向杂凑函数满足这一性质, 则称其为强单向杂凑函数。

第⑤和第⑥个条件给出了杂凑函数无碰撞性的概念, 如果杂凑函数对不同的输入可产生相同的输出, 则称该函数具有碰撞性。

以上 6 个条件中, 前 3 个是杂凑函数能用于消息认证的基本要求。第 4 个条件(即单向性)则对使用秘密值的认证技术(见图 6-3(e))极为重要。假如杂凑函数不具有单向性, 则攻击者截获 M 和 $C=H(S \parallel M)$ 后, 求 C 的逆 $S \parallel M$, 就可求出秘密值 S 。第 5 个条件使得敌手无法在已知某个消息时, 找到与该消息具有相同杂凑值的另一消息。这一性质用于杂凑值被加密情况时(见图 6-3(b)和图 6-3(c))防止敌手的伪造, 由于在这种情况下, 敌手可读取传送的明文消息 M , 因此能产生该消息的杂凑值 $H(M)$ 。但由于敌手不知道用于加密杂凑值的密钥, 他就不可能既伪造一个消息 M , 又伪造这个消息的杂凑值加密后的密文 $E_k[H(M)]$ 。然而, 如果第 5 个条件不成立, 敌手在截获明文消息及其加密的杂凑值后, 就可按以下方式伪造消息: 首先求出截获的消息的杂凑值, 然后产生一个具有相同杂凑值的伪造消息, 最后再将伪造的消息和截获的加密的杂凑值发往通信的接收方。第 6 个条件用于抵抗生日攻击。

6.2.3 生日攻击

1. 相关问题

已知一杂凑函数 H 有 n 个可能的输出, $H(x)$ 是一个特定的输出, 如果对 H 随机取 k 个输入, 则至少有一个输入 y 使得 $H(y)=H(x)$ 的概率为 0.5 时, k 有多大?

以后为叙述方便,称对杂凑函数 H 寻找上述 y 的攻击为第 1 类生日攻击。

因为 H 有 n 个可能的输出,所以输入 y 产生的输出 $H(y)$ 等于特定输出 $H(x)$ 的概率是 $1/n$,反过来说 $H(y) \neq H(x)$ 的概率是 $1-1/n$ 。 y 取 k 个随机值而函数的 k 个输出中没有一个是等于 $H(x)$,其概率等于每个输出都不等于 $H(x)$ 的概率之积,为 $[1-1/n]^k$,所以 y 取 k 个随机值得到函数的 k 个输出中至少有一个等于 $H(x)$ 的概率为 $1-[1-1/n]^k$ 。

由 $(1+x)^k \approx 1+kx$,其中 $|x| \ll 1$,可得

$$1-[1-1/n]^k \approx 1-[1-k/n] = k/n$$

若使上述概率等于 0.5,则 $k=n/2$ 。特别地,如果 H 的输出为 m 比特长,即可能的输出个数 $n=2^m$,则 $k=2^{m-1}$ 。

2. 生日悖论

生日悖论是考虑这样一个问题:在 k 个人中至少有两个人的生日相同的概率大于 0.5 时, k 至少多大?

为了回答这一问题,首先定义下述概率:设有 k 个整数项,每一项都在 1 到 n 之间等可能地取值,则 k 个整数项中至少有两个取值相同的概率为 $P(n, k)$ 。因而生日悖论就是求使得 $P(365, k) \geq 0.5$ 的最小 k ,为此首先考虑 k 个数据项中任意两个取值都不同的概率,记为 $Q(365, k)$ 。如果 $k > 365$,则不可能使得任意两个数据都不相同,因此假定 $k \leq 365$ 。 k 个数据项中任意两个都不相同的所有取值方式数为

$$365 \times 364 \times \cdots \times (365 - k + 1) = \frac{365!}{(365 - k)!}$$

即第 1 个数据项可从 365 个中任取一个,第 2 个数据项可在剩余的 364 个中任取一个,依次类推,最后一个数据项可从 $365 - k + 1$ 个值中任取一个。如果去掉任意两个都不相同这一限制条件,可得 k 个数据项中所有取值方式数为 365^k 。所以可得

$$Q(365, k) = \frac{365!}{(365 - k)! 365^k}$$

$$P(365, k) = 1 - Q(365, k) = 1 - \frac{365!}{(365 - k)! 365^k}$$

当 $k=23$ 时, $P(365, 23)=0.5073$,即上述问题只需 23 人,人数如此之少。若 k 取 100,则 $P(365, 100)=0.9999997$,即获得如此大的概率。之所以称这一问题是悖论是因为当人数 k 给定时,得到的至少有两个人的生日相同的概率比想象的要大得多。这是因为在 k 个人中考虑的是任意两个人的生日是否相同,在 23 个人中可能的情况数为 $C_{23}^2=253$ 。

将生日悖论推广为下述问题:已知一个在 1 到 n 之间均匀分布的整数型随机变量,若该变量的 k 个取值中至少有两个取值相同的概率大于 0.5,则 k 至少多大?

与上类似, $P(n, k) = 1 - \frac{n!}{(n-k)! n^k}$, 令 $P(n, k) > 0.5$, 可得 $k = 1.18\sqrt{n} \approx \sqrt{n}$ 。

若取 $n=365$, 则 $k=1.18 \sqrt{365}=22.54$ 。

3. 生日攻击

生日攻击是基于下述结论: 设杂凑函数 H 有 2^m 个可能的输出 (即输出长 m 比特), 如果 H 的 k 个随机输入中至少有两个产生相同输出的概率大于 0.5, 则 $k \approx \sqrt{2^m} = 2^{m/2}$ 。

称寻找函数 H 的具有相同输出的两个任意输入的攻击方式为第 II 类生日攻击。

第Ⅱ类生日攻击可按以下方式进行:

① 设用户将用图 6-3(c)所示的方式发送消息,即 A 用自己的秘密钥对消息的杂凑值加密,加密结果作为对消息的签字,连同明文消息一起发往接收者。

② 敌手对 A 发送的消息 M 产生出 $2^{m/2}$ 个变形的消息, 每个变形的消息本质上的含义与原消息相同, 同时敌手还准备一个假冒的消息 M' , 并对假冒的消息产生出 $2^{m/2}$ 个变形的消息。

③ 敌手在产生的两个消息集合中,找出杂凑值相同的一对消息 \tilde{M}, \tilde{M} ,由上述讨论可知敌手成功的概率大于 0.5。如果不成功,则重新产生一个假冒的消息,并产生 $2^{n/2}$ 个变形,直到找到杂凑值相同的一对消息为止。

④ 敌手将 \tilde{M} 提交给 A 请求签字, 由于 \tilde{M} 与 \tilde{M} 的杂凑值相同, 所以可将 A 对 \tilde{M} 的签字当作对 \tilde{M} 的签字, 将此签字连同 \tilde{M} 一起发给意欲的接收者。

上述攻击中如果杂凑值的长为 64 比特, 则敌手攻击成功所需的时间复杂度为 $O(2^{12})$ 。

将一个消息变形为具有相同含义的另一消息的方法有很多,例如对文件,敌手可在文件的单词之间插入很多“space-space-backspace”字符对,然后将其中的某些字符对替换为“space-backspace-space”就得到一个变形的消息。

6.2.4 迭代型杂凑函数的一般结构

目前使用的大多数杂凑函数如 MD5、SHA, 其结构都是迭代型的, 如图 6-4 所示。其中函数的输入 M 被分为 L 个分组 Y_0, Y_1, \dots, Y_{L-1} , 每一个分组的长度为 b 比特, 最后一个分组的长度不够的话, 需对其做填充。最后一个分组中还包括整个函数输入的长度值, 这样一来, 将使得敌手的攻击更为困难, 即敌手若想成功地产生假冒的消息, 就必须保证假冒消息的杂凑值与原消息的杂凑值相同, 而且假冒消息的长度也要与原消息的长度相等。

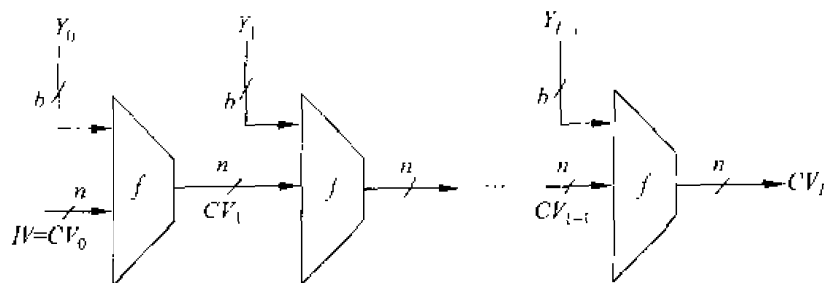


图 6-4 迭代型杂凑函数的一般结构

算法中重复使用一压缩函数 f (注意, 有些书将杂凑函数也称为压缩函数, 在此用压缩函数表示杂凑函数中的一个特定部分), f 的输入有两项, 一项是上一轮 (第 $i-1$ 轮) 输出的 n 比特值 CV_{i-1} , 称为链接变量, 另一项是算法在本轮 (第 i 轮) 的 b 比特输入分组 Y_i , f 的输出为 n 比特值 CV_i , CV_i 又作为下一轮的输入。算法开始时还需对链接变量指定一个初值 IV , 最后一轮输出的链接变量 CV_L 即为最终产生的杂凑值。通常有 $b > n$, 因此称函数 f 为压缩函数。算法可表达如下:

$CV_0 = IV = n$ 比特长的初值;

$CV_i = f(CV_{i-1}, Y_{i-1}), 1 \leq i \leq L$;

$H(M) = CV_L$

算法的核心技术是设计无碰撞的压缩函数 f , 而敌手对算法的攻击重点是 f 的内部结构, 由于 f 和分组密码一样是由若干轮处理过程组成, 所以对 f 的攻击需通过对各轮之间的位模式的分析来进行, 分析过程常常需要先找出 f 的碰撞。由于 f 是压缩函数, 其碰撞是不可避免的, 因此在设计 f 时就应保证找出其碰撞在计算上是不可行的。

下面介绍几个重要的迭代型杂凑函数。

6.3 MD5 杂凑算法

MD4 是 MD5 杂凑算法的前身, 由 Ron Rivest 于 1990 年 10 月作为 RFC 提出, 1992 年 4 月公布的 MD4 的改进 (RFC 1320, 1321) 称为 MD5。

6.3.1 算法描述

MD5 算法采用图 6-4 描述的迭代型杂凑函数的一般结构, 算法的框图如图 6-5 所示。算法的输入为任意长的消息 (图中为 K 比特), 分为 512 比特长的分组, 输出为 128 比特的消息摘要。

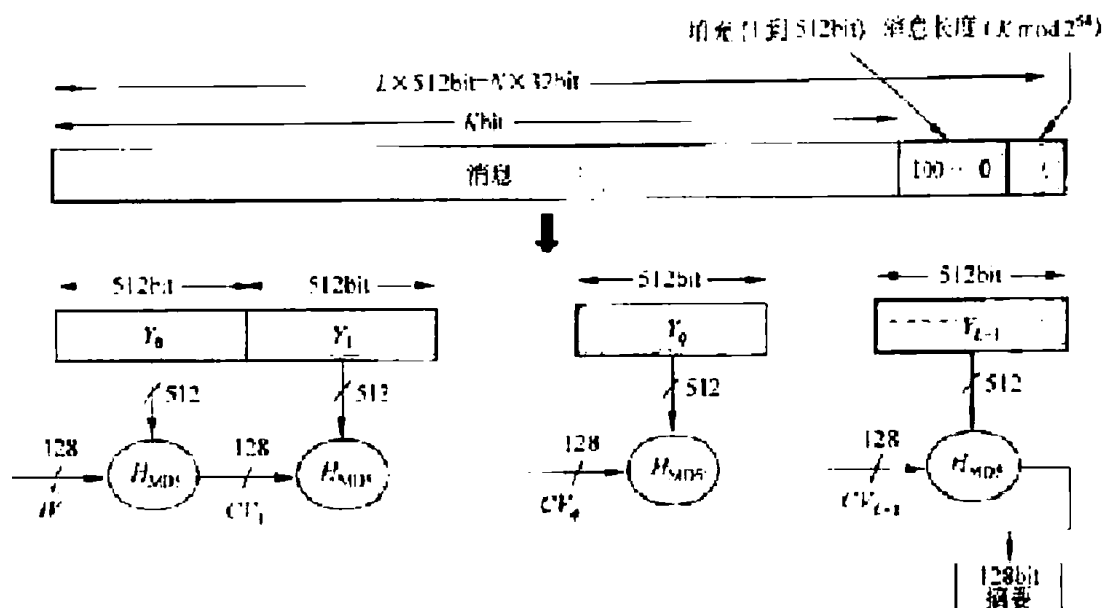


图 6-5 MD5 的算法框图

处理过程有以下几步：

① 对消息填充 对消息填充,使得其比特长 $\text{在模 } 512 \text{ 下为 } 448$,即填充后消息的长度为 512 的某一倍数减 64,留出的 64 比特备第 2 步使用。步骤①是必需的,即使消息长度已满足要求,仍需填充。例如,消息长为 448 比特,则需填充 512 比特,使其长度变为 960,因此填充的比特数大于等于 1 而小于等于 512。

填充方式是固定的,即第 1 位为 1,其后各位皆为 0。

② 附加消息的长度 用步骤①留出的 64 比特以 little-endian 方式来表示消息被填充前的长度。如果消息长度大于 2^{64} ,则以 2^{64} 为模数取模。

little-endian 方式是指按数据的最低有效字节(byte)(或最低有效位)优先的顺序存储数据,即将最低有效字节(或最低有效位)存于低地址字节(或位)。相反的存储方式称为 big endian 方式。

前两步执行完后,消息的长度为 512 的倍数(设为 L 倍),则可将消息表示为分组长为 512 的一系列分组 Y_0, Y_1, \dots, Y_{L-1} ,而每一分组又可表示为 16 个 32 比特长的字,这样消息中的总字数为 $N=L \times 16$,因此消息又可按字表示为 $M[0, \dots, N-1]$ 。

③ 对 MD 缓冲区初始化 算法使用 128 比特长的缓冲区以存储中间结果和最终杂凑值,缓冲区可表示为 4 个 32 比特长的寄存器(A, B, C, D),每个寄存器都以 little-endian 方式存储数据,其初值取为(以存储方式) $A=01234567, B=89ABCDEF, C=FEDCBA98, D=76543210$,实际上为 67452301, EFCDA89, 98BADCFE, 10325476。

④ 以分组为单位对消息进行处理 每一分组 Y_q ($q=0, \dots, L-1$) 都经一压缩函数 H_{MD5} 处理。 H_{MD5} 是算法的核心, 其中又有 4 轮处理过程, 如图 6-6 所示。

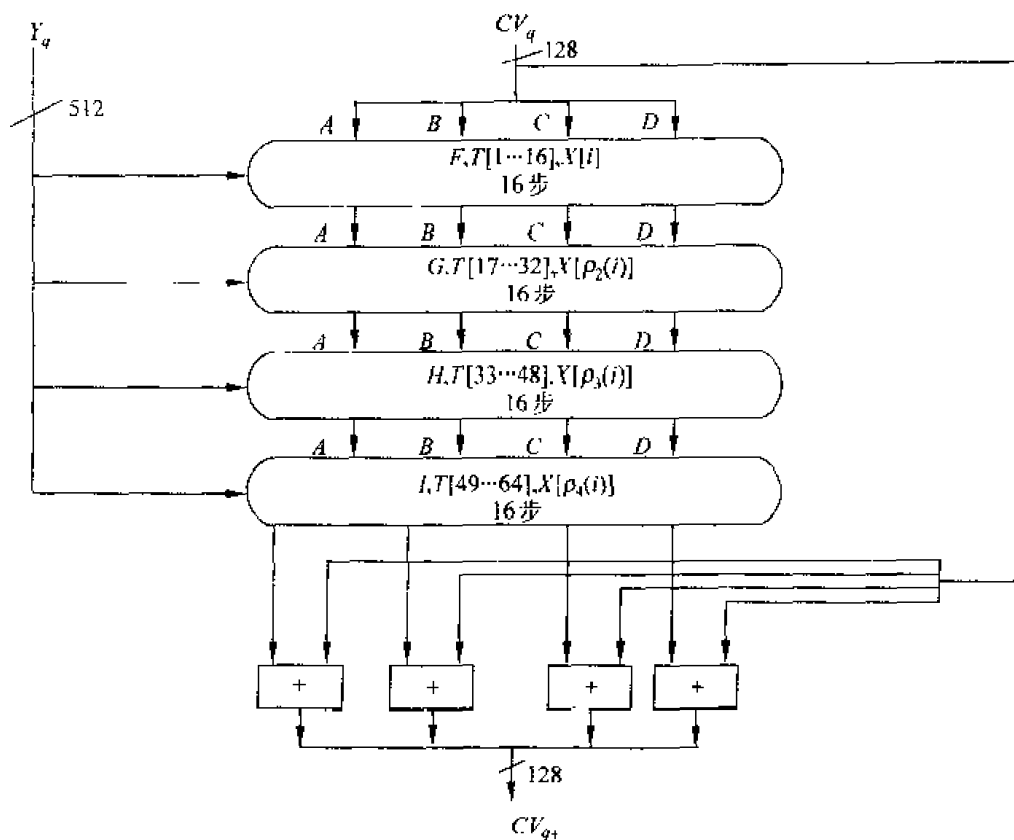


图 6-6 MD5 的分组处理框图

H_{MD5} 的 4 轮处理过程结构一样, 但所用的逻辑函数不同, 分别表示为 F 、 G 、 H 、 I 。每轮的输入为当前处理的消息分组 Y_q 和缓冲区的当前值 A 、 B 、 C 、 D , 输出仍放在缓冲区中以产生新的 A 、 B 、 C 、 D 。每轮处理过程还需加上常数表 T 中四分之一元素, 分别为 $T[1 \dots 16]$, $T[17 \dots 32]$, $T[33 \dots 48]$, $T[49 \dots 64]$ 。表 T 有 64 个元素, 见表 6-1, 第 i 个元素 $T[i]$ 为 $2^{32} \times \text{abs}(\sin(i))$ 的整数部分, 其中 \sin 为正弦函数, i 以弧度为单位。由于 $\text{abs}(\sin(i))$ 大于 0 小于 1, 所以 $T[i]$ 可由 32 比特的字表示。第 4 轮的输出再与第 1 轮的输入 CV_q 相加, 相加时将 CV_q 看作 4 个 32 比特的字, 每个字与第 4 轮输出的对应的字按模 2^{32} 相加, 相加的结果即为压缩函数 H_{MD5} 的输出。

⑤ 输出 消息的 L 个分组都被处理完后, 最后一个 H_{MD5} 的输出即为产生的消息摘要。

步骤③到步骤⑤的处理过程可总结如下:

$$CV_0 = IV;$$

$$CV_{q-1} = CV_q + RF_I[Y_q, RF_H[Y_q, RF_G[Y_q, RF_F[Y_q, CV_q]]]]$$

$$MD = CV_L$$

其中 IV 是步骤③所取的缓冲区 $ABCD$ 的初值, Y_q 是消息的第 q 个 512 比特长的分组, L 是消息经过步骤①和步骤②处理后的分组数, CV_q 为处理消息的第 q 个分组时输入的链接变量(即前一个压缩函数的输出), RF_i 为使用基本逻辑函数 x 的轮函数, $+$ 为对应字的模 2^{32} 加法, MD 为最终的杂凑值。

表 6-1 常数表 T

$T[1]=D76AA478$	$T[17]=F61E2562$	$T[33]=FFFA3942$	$T[49]=F4292244$
$T[2]=E8C7B756$	$T[18]=C040B340$	$T[34]=8771F681$	$T[50]=432AFF97$
$T[3]=242070DB$	$T[19]=265E5A51$	$T[35]=699D6122$	$T[51]=AB9423A7$
$T[4]=C1BDCFEF$	$T[20]=E9B6C7AA$	$T[36]=FDE5380C$	$T[52]=FC93A039$
$T[5]=F57C0FAF$	$T[21]=D62F105D$	$T[37]=A4BEEA44$	$T[53]=655B59C3$
$T[6]=4787C62A$	$T[22]=02441453$	$T[38]=4BDECFA9$	$T[54]=8F0CCC92$
$T[7]=A8304613$	$T[23]=D8A1E681$	$T[39]=F6BR4R60$	$T[55]=FFEFF47D$
$T[8]=FD469501$	$T[24]=E7D3FBC8$	$T[40]=BEBFBC70$	$T[56]=85845DD1$
$T[9]=698098D8$	$T[25]=21E1CDE6$	$T[41]=289B7EC6$	$T[57]=6FA87E4F$
$T[10]=8B44F7AF$	$T[26]=C33707D6$	$T[42]=EAA127FA$	$T[58]=FE2CE6E0$
$T[11]=FFFF5BB1$	$T[27]=F4D50D87$	$T[43]=D4EF3085$	$T[59]=A3014314$
$T[12]=895CD7BE$	$T[28]=455A14ED$	$T[44]=04881D05$	$T[60]=4E0811A1$
$T[13]=6B901122$	$T[29]=A9E3E905$	$T[45]=D9D4D039$	$T[61]=F7537E82$
$T[14]=FD987193$	$T[30]=FCEFA3F8$	$T[46]=E6DB99E5$	$T[62]=BD3AF235$
$T[15]=A679438E$	$T[31]=676F02D9$	$T[47]=1FA27CF8$	$T[63]=2AD7D2BB$
$T[16]=49B40821$	$T[32]=8D2A4C8A$	$T[48]=C4AC5665$	$T[64]=EB86D391$

6.3.2 MD5 的压缩函数

压缩函数 H_{MD5} 中有 4 轮处理过程, 每轮又对缓冲区 $ABCD$ 进行 16 步迭代运算, 每一步的运算形式为(见图 6-7)

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

其中 a, b, c, d 为缓冲区的 4 个字, 运算完成后再右循环一个字, 即得这一步迭代的输出。 g 是基本逻辑函数 F, G, H, I 之一。 CLS_s 是左循环移 s 位, s 的取值由表 6-2 给出。

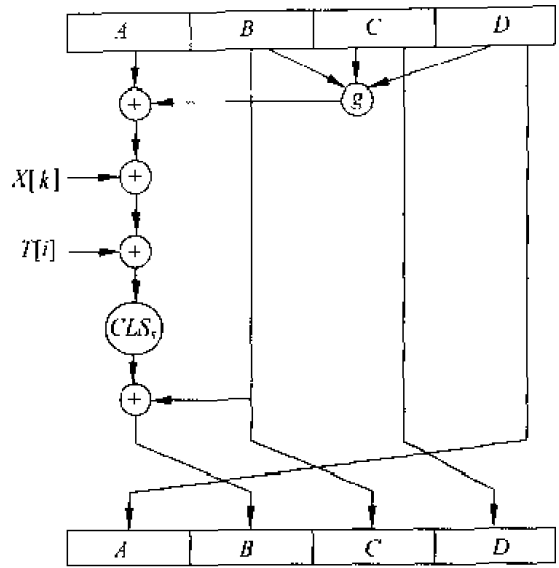


图 6-7 压缩函数中的一步迭代示意图

表 6-2 压缩函数每步左循环移位的位数

步数 轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	7	12	17	22	7	12	17	22	7	12	17	22	7	12	17	22
2	5	9	14	20	5	9	14	20	5	9	14	20	5	9	14	20
3	4	11	16	23	4	11	16	23	4	11	16	23	4	11	16	23
4	6	10	15	21	6	10	15	21	6	10	15	21	6	10	15	21

$T[i]$ 为表 T 中的第 i 个字, $|$ 为模 2^{32} 加法。 $X[k]=M[q \times 16+k]$, 即消息第 q 个分组中的第 k 个字($k=1, \cdots, 16$)。4 轮处理过程中, 每轮以不同的次序使用 16 个字, 其中在第 1 轮以字的初始次序使用。第 2 轮到第 4 轮分别对字的次序 i 做置换后得到一个新次序, 然后以新次序使用 16 个字。3 个置换分别为

$$\begin{aligned} \rho_2(i) &= (1 + 5i) \bmod 16 \\ \rho_3(i) &= (5 + 3i) \bmod 16 \\ \rho_4(i) &= 7i \bmod 16 \end{aligned}$$

4 轮处理过程分别使用不同的基本逻辑函数 F, G, H, I , 每个逻辑函数的输入为 3 个 32 比特的字, 输出是一个 32 比特的字, 其中的运算为逐比特的逻辑运算, 即输出的第 n 个比特是 3 个输入的第 n 个比特的函数, 函数的定义由表 6-3 给出, 其中 $\wedge, \vee, -, \oplus$ 分别是逻辑与、逻辑或、逻辑非和异或运算, 表 6-4 是四个函数的真值表。

表 6-3 基本逻辑函数的定义

轮数	基本逻辑函数	$g(b, c, d)$
1	$F(b, c, d)$	$(b \wedge c) \vee (\bar{b} \wedge d)$
2	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \bar{d})$
3	$H(b, c, d)$	$b \oplus c \oplus d$
4	$I(b, c, d)$	$c \oplus (b \vee \bar{d})$

表 6-4 基本逻辑函数的真值表

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

6.3.3 MD5 的安全性

MD5 有这样—个性质,即杂凑码中的每一个比特是所有输入比特的函数,因此获得了很好的混淆效果,从而使得不可能随机选择两个具有相同杂凑值的消息。Rivest 猜想作为 128 比特长的杂凑值来说,MD5 的强度达到了最大,比如说找出具有相同杂凑值的两个消息需执行 $O(2^{64})$ 次运算,而寻找具有给定杂凑值的一个消息需要执行 $O(2^{128})$ 次运算。

目前对 MD5 的攻击已取得以下结果:

① 对单轮 MD5 使用差分密码分析,可在合理的时间内找出具有相同杂凑值的两个消息。但这种攻击还未能成功地推广到 4 轮 MD5。

② 可找出一个消息分组和两个相关的链接变量(即缓冲区变量 ABCD),使得算法产生出相同的输出。目前这种攻击还未能成功地推广到整个算法。

③ 对单个 512 比特长的消息分组已成功地找出了碰撞,即可找出另一个消息分组,

使得算法对两个消息分组的 128 比特长的输出相同。目前这种攻击还未成功推广到在有初值 IV 时对整个消息运行该算法。

因此从密码分析的角度来看,MD5 被认为是易受攻击的。而从穷搜索的角度来看,第 II 类生日攻击需进行 $O(2^{64})$ 次运算,因此认为 MD5 易受第 II 类生日攻击的威胁。所以必须寻找新的杂凑算法,以使其产生的杂凑值更长,且抵抗已知密码分析攻击的能力更强。下面要介绍的 SHA 即为这样的一个算法。

6.4 安全杂凑算法

安全杂凑算法(secure hash algorithm, SHA)由美国 NIST 设计,于 1993 年作为联邦信息处理标准(FIPS PUB 180)公布。SHA 是基于 MD4 的算法,其结构与 MD4 非常类似。

6.4.1 算法描述

算法的输入为小于 2^{64} 比特长的任意消息,分为 512 比特长的分组,输出为 160 比特长的消息摘要。算法的框图与图 6-5 一样,但杂凑值的长度和链接变量的长度为 160 比特。

算法的处理过程有以下几步:

① 对消息填充 与 MD5 的步骤①完全相同。

② 附加消息的长度 与 MD5 的步骤②类似,不同之处在于以 big-endian 方式表示填充前消息的长度。即步骤①留出的 64 比特当作 64 比特长的无符号整数。

③ 对 MD 缓冲区初始化 算法使用 160 比特长的缓冲区存储中间结果和最终杂凑值,缓冲区可表示为 5 个 32 比特长的寄存器(A, B, C, D, E),每个寄存器都以 big-endian 方式存储数据,其初始值分别为 $A = 67452301, B = \text{EFCDA389}, C = 98\text{BADCFB}, D = 10325476, E = \text{C3D2E1F0}$ 。

④ 以分组为单位对消息进行处理 每一分组 Y_i 都经一压缩函数处理,压缩函数由 4 轮处理过程(如图 6-8 所示)构成,每一轮又由 20 步迭代组成。4 轮处理过程结构一样,但所用的基本逻辑函数不同,分别表示为 f_1, f_2, f_3, f_4 。每轮的输入为当前处理的消息分组 Y_i 和缓冲区的当前值 A, B, C, D, E ,输出仍放在缓冲区以替代 A, B, C, D, E 的旧值,每轮处理过程还需加上一个加法常量 K_t ,其中 $0 \leq t \leq 79$ 表示迭代的步数。80 个常量中实际上只有 4 个不同取值,如表 6-5 所示,其中 $\lfloor x \rfloor$ 为 x 的整数部分。

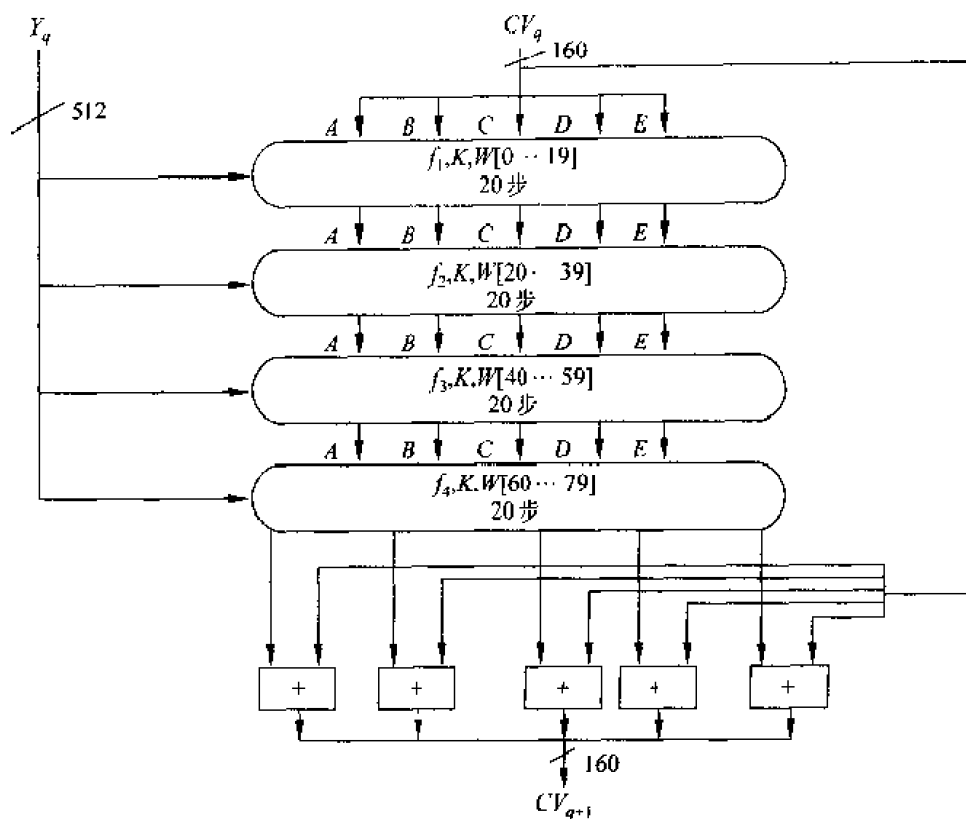


图 6-8 SHA 的分组处理框图

表 6-5 SHA 的加法常量

迭代步数 t	常量 K_t (十六进制)	K_t (十进制)
$0 \leq t \leq 19$	5A827999	$\lfloor 2^{30} \times \sqrt{2} \rfloor$
$20 \leq t \leq 39$	6ED9EBA1	$\lfloor 2^{30} \times \sqrt{3} \rfloor$
$40 \leq t \leq 59$	8F1BBCDC	$\lfloor 2^{30} \times \sqrt{5} \rfloor$
$60 \leq t \leq 79$	CA62C1D6	$\lfloor 2^{30} \times \sqrt{10} \rfloor$

第 4 轮的输出(即第 80 步迭代的输出)再与第 1 轮的输入 CV_q 相加,以产生 CV_{q+1} ,其中加法是缓冲区 5 个字中的每一个字与 CV_q 中相应的字模 2^{32} 相加。

⑤ 输出 消息的 L 个分组都被处理完后,最后一个分组的输出即为 160 比特的消息摘要。

步骤③到步骤⑤的处理过程可总结如下:

$$CV_0 = IV;$$

$$CV_{q+1} = \text{SUM}_{32}(CV_q, ABCDE_q);$$

$$MD = (V_L$$

其中 IV 是步骤③定义的缓冲区 $ABCDE$ 的初值, $ABCDE_q$ 是第 q 个消息分组经最后一轮处理过程处理后的输出, L 是消息(包括填充位和长度字段)的分组数, SUM_{12} 是对应字的模 2^{32} 加法, MD 为最终的摘要值。

6.4.2 SHA 的压缩函数

如上所述, SHA 的压缩函数由 4 轮处理过程组成, 每轮处理过程又由对缓冲区 $ABCDE$ 的 20 步迭代运算组成, 每一步迭代运算的形式为(见图 6-9)

$$A, B, C, D, E \leftarrow (E + f_t(B, C, D) + CLS_s(A) + W_t + K_t), A, CLS_{30}(B), C, D$$

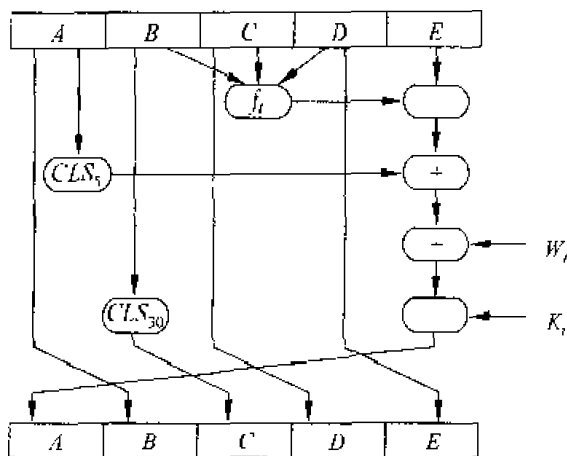


图 6-9 SHA 的压缩函数中一步迭代示意图

其中 A, B, C, D, E 为缓冲区的 5 个字, t 是迭代的步数 ($0 \leq t \leq 79$), $f_t(B, C, D)$ 是第 t 步迭代使用的基本逻辑函数, CLS_s 为左循环移 s 位, W_t 是由当前 512 比特长的分组导出的一个 32 比特长的字(导出方式见下面), K_t 是加法常量, $+$ 是模 2^{32} 加法。

基本逻辑函数的输入为 3 个 32 比特的字, 输出是一个 32 比特的字, 其中的运算为逐比特逻辑运算, 即输出的第 n 个比特是 3 个输入的相应比特的函数。函数的定义如表 6-6。表中 $\wedge, \vee, -, \oplus$ 分别是与、或、非、异或 4 个逻辑运算, 函数的真值表如表 6-7 所示。

表 6-6 SHA 中基本逻辑函数的定义

迭代的步数	函 数 名	定 义
$0 \leq t \leq 19$	$f_1 = f_t(B, C, D)$	$(B \wedge C) \vee (\bar{B} \wedge D)$
$20 \leq t \leq 39$	$f_2 = f_t(B, C, D)$	$B \oplus C \oplus D$
$40 \leq t \leq 59$	$f_3 = f_t(B, C, D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \leq t \leq 79$	$f_4 = f_t(B, C, D)$	$B \oplus C \oplus D$

表 6-7 SHA 的基本逻辑函数的真值表

B	C	D	f_1	f_2	f_3	f_4	B	C	D	f_1	f_2	f_3	f_4
0	0	0	0	0	0	0	1	0	0	0	1	0	1
0	0	1	1	1	0	1	1	0	1	0	0	1	0
0	1	0	0	1	0	1	1	1	0	1	0	1	0
0	1	1	1	0	1	0	1	1	1	1	1	1	1

下面说明如何由当前的输入分组(512 比特长)导出 W_i (32 比特长)。前 16 个值(即 W_0, W_1, \dots, W_{15})直接取为输入分组的 16 个相应的字,其余值(即 $W_{16}, W_{17}, \dots, W_{79}$)取为

$$W_i = CLS_1(W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3})$$

见图 6-10。与 MD5 比较,MD5 直接用一个消息分组的 16 个字作为每步迭代的输入,而 SHA 则将输入分组的 16 个字扩展成 80 个字以供压缩函数使用,从而使得寻找具有相同压缩值的不同的消息分组更为困难。

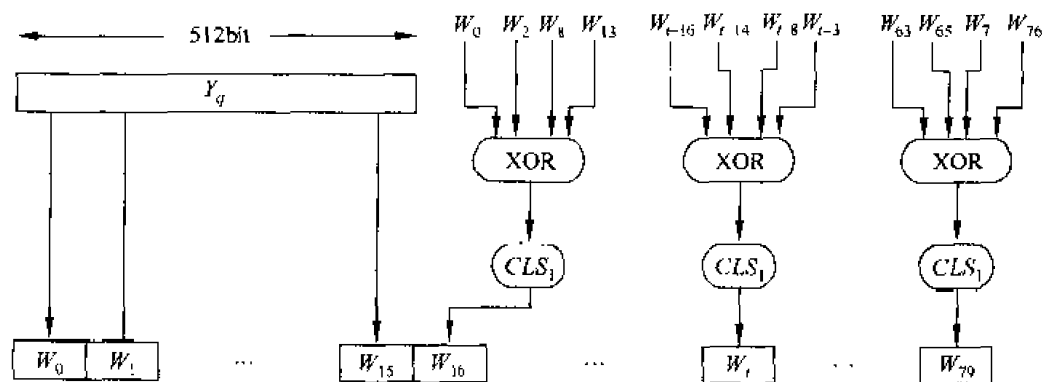


图 6-10 SHA 分组处理所需的 80 个字的产生过程

6.4.3 SHA 与 MD5 的比较

由于 SHA 与 MD5 都是由 MD4 演化而来,所以两个算法极为相似。

1. 抗穷搜索攻击的强度

由于 SHA 和 MD5 的消息摘要长度分别为 160 和 128,所以用穷搜索攻击寻找具有给定消息摘要的消息分别需做 $O(2^{160})$ 和 $O(2^{128})$ 次运算,而用穷搜索攻击找出具有相同消息摘要的两个不同消息分别需做 $O(2^{80})$ 和 $O(2^{64})$ 次运算。因此 SHA 抗击穷搜索攻击的强度高于 MD5 抗击穷搜索攻击的强度。

2. 抗击密码分析攻击的强度

由于 SHA 的设计准则未被公开,所以它抗击密码分析攻击的强度较难判断,似乎高

于 MD5 的强度。

3. 速度

由于两个算法的主要运算都是模 2^{32} 加法,因此都易于在 32 位结构上实现。但比较起来,SHA 的迭代步数(80 步)多于 MD5 的迭代步数(64 步),所用的缓冲区(160 比特)大于 MD5 使用的缓冲区(128 比特),因此在相同硬件上实现时,SHA 的速度要比 MD5 的速度慢。

4. 简洁与紧致性

两个算法描述起来都较为简单,实现起来也较为简单,都不需要大的程序和代换表。

5. 数据的存储方式

MD5 使用 little-endian 方式,SHA 使用 big-endian 方式。两种方式相比看不出哪个更具优势,之所以使用两种不同的存储方式是因为设计者最初实现各自的算法时,使用的机器的存储方式不同。

6.5 HMAC 算法

6.1.3 一节中曾介绍过一个 MAC 的例子——数据认证算法,该算法反映了传统上构造 MAC 最为普遍使用的方法,即基于分组密码的构造方法。但近年来研究构造 MAC 的兴趣已转移到基于密码杂凑函数的构造方法,这是因为:

- 密码杂凑函数(如 MD5、SHA)的软件实现快于分组密码(如 DES)的软件实现;
- 密码杂凑函数的库代码来源广泛;
- 密码杂凑函数没有出口限制,而分组密码即使用于 MAC 也有出口限制。

杂凑函数并不是为用于 MAC 而设计的,由于杂凑函数不使用密钥,因此不能直接用于 MAC。目前已提出了很多将杂凑函数用于构造 MAC 的方法,其中 HMAC 就是其中之一,已作为 RFC2104 被公布,并在 IPsec 和其他网络协议(如 SSL)中得以应用。

6.5.1 HMAC 的设计目标

RFC2104 列举了 HMAC 的以下设计目标:

- ① 可不经修改而使用现有的杂凑函数,特别是那些易于软件实现的、源代码可方便获取且免费使用的杂凑函数。
- ② 其中镶嵌的杂凑函数可易于替换为更快或更安全的杂凑函数。
- ③ 保持镶嵌的杂凑函数的最初性能,不因用于 HMAC 而使其性能降低。

④ 以简单方式使用和处理密钥。

⑤ 在对镶嵌的杂凑函数合理假设的基础上,易于分析 HMAC 用于认证时的密码强度。

其中前两个目标是 HMAC 被公众普遍接受的主要原因,这两个目标是将杂凑函数当作一个黑盒使用,这种方式有两个优点:第一,杂凑函数的实现可作为实现 HMAC 的一个模块,这样一来,HMAC 代码中很大一块就可事先准备好,无需修改就可使用;第二,如果 HMAC 要求使用更快或更安全的杂凑函数,则只需用新模块代替旧模块,例如用实现 SHA 的模块代替 MD5 的模块。

最后一条设计目标则是 HMAC 优于其他基于杂凑函数的 MAC 的一个主要方面,HMAC 在其镶嵌的杂凑函数具有合理密码强度的假设下,可证明是安全的,这一问题将在 6.5.3 节 HMAC 的安全性中介绍。

6.5.2 算法描述

图 6-11 是 HMAC 算法的运行框图,其中 H 为嵌入的杂凑函数(如 MD5、SHA), M 为 HMAC 的输入消息(包括杂凑函数所要求的填充位), $Y_i (0 \leq i \leq L-1)$ 是 M 的第 i 个分组, L 是 M 的分组数, b 是一个分组中的比特数, n 为由嵌入的杂凑函数所产生的杂凑值的长度, K 为密钥,如果密钥长度大于 b ,则将密钥输入到杂凑函数中产生一个 n 比特长的密钥, K^+ 是左边经填充 0 后的 K , K^+ 的长度为 b 比特, $ipad$ 为 $b/8$ 个 00110110, $opad$ 为 $b/8$ 个 01011010。

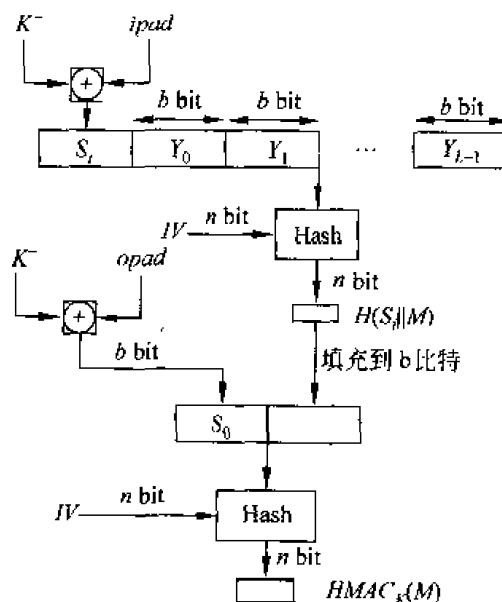


图 6-11 HMAC 的算法框图

算法的输出可如下表示:

$$HMAC_k = H[(K^+ \oplus opad) \parallel H[(K^- \oplus ipad) \parallel M]]$$

算法的运行过程可描述如下:

- ① K 的左边填充 0 以产生一个 b 比特长的 K^+ (例如 K 的长为 160 比特, $b=512$, 则需填充 44 个零字节 0x00)。
- ② K^- 与 $ipad$ 逐比特异或以产生 b 比特的分组 S_i 。
- ③ 将 M 链接到 S_i 后。
- ④ 将 H 作用于步骤③产生的数据流。
- ⑤ K^+ 与 $opad$ 逐比特异或, 以产生 b 比特长的分组 S_0 。
- ⑥ 将步骤④得到的杂凑值链接在 S_0 后。
- ⑦ 将 H 作用于步骤⑥产生的数据流并输出最终结果。

注意, K^+ 与 $ipad$ 逐比特异或以及 K^+ 与 $opad$ 逐比特异或的结果是将 K 中的一半比特取反, 但两次取反的比特的位置不同。而 S_i 和 S_0 通过杂凑函数中压缩函数的处理, 则相当于以伪随机方式从 K 产生两个密钥。

在实现 HMAC 时, 可预先求出下面两个量 (见图 6-12, 虚线以左为预计算):

$$f(IV, (K^+ \oplus ipad))$$

$$f(IV, (K^+ \oplus opad))$$

其中 $f(cv, block)$ 是杂凑函数中的压缩函数, 其输入是 n 比特的链接变量和 b 比特的分组, 输出是 n 比特的链接变量。这两个量的预先计算只在每次更改密钥时才需进行。事

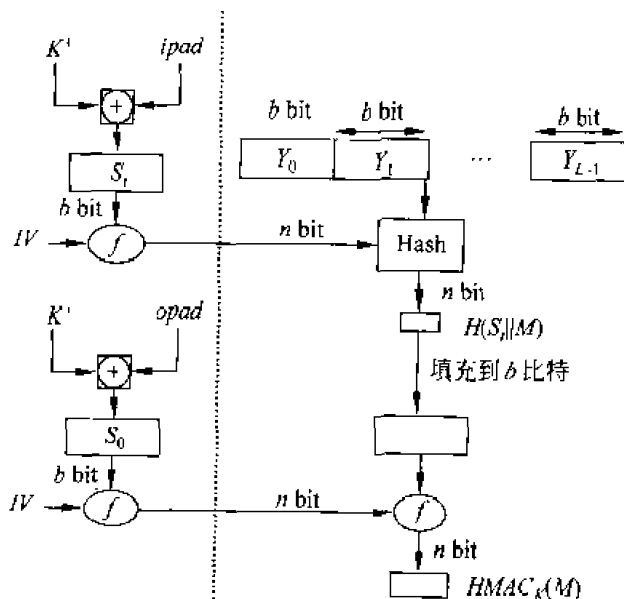


图 6-12 HMAC 的有效实现

实上这两个预先计算的量用于作为杂凑函数的初值 IV 。

6.5.3 HMAC 的安全性

基于密码杂凑函数构造的 MAC 的安全性取决于镶嵌的杂凑函数的安全性,而 HMAC 最吸引人的地方是它的设计者已经证明了算法的强度和嵌入的杂凑函数的强度之间的确切关系,证明了对 HMAC 的攻击等价于对内嵌杂凑函数的下述两种攻击之一:

① 攻击者能够计算压缩函数的一个输出,即使 IV 是随机的和秘密的。

② 攻击者能够找出杂凑函数的碰撞,即使 IV 是随机的和秘密的。

在第一种攻击中,可将压缩函数视为与杂凑函数等价,而杂凑函数的 n 比特长 IV 可视为 HMAC 的密钥。对这一杂凑函数的攻击可通过对密钥的穷搜索来进行,也可通过第 II 类生日攻击来实施,通过对密钥的穷搜索攻击的复杂度为 $O(2^n)$,通过第 II 类生日攻击又可归结为上述第二种攻击。

第二种攻击指攻击者寻找具有相同杂凑值的两个消息,因此就是第 II 类生日攻击。对杂凑值长度为 n 的杂凑函数来说,攻击的复杂度为 $O(2^{n/2})$ 。因此第二种攻击对 MD5 的攻击复杂度为 $O(2^{64})$,就现在的技术来说,这种攻击是可行的。但这是否意味着 MD5 不适合用于 HMAC? 回答是否定的,原因如下:攻击者在攻击 MD5 时,可选择任何消息集合后脱线寻找碰撞。由于攻击者知道杂凑算法和默认的 IV ,因此能为自己产生的每个消息求出杂凑值。然而,在攻击 HMAC 时,由于攻击者不知道密钥 K ,从而不能脱线产生消息和认证码对。所以攻击者必须得到 HMAC 在同一密钥下产生的一系列消息,并对得到的消息序列进行攻击。对长 128 比特的杂凑值来说,需要得到用同一密钥产生的 2^{64} 个分组 (2^{73} 比特)。在 1Gbit/s 的链路上,需 250000 年,因此 MD5 完全适合于 HMAC,而且就速度而言,MD5 要快于 SHA 作为内嵌杂凑函数的 HMAC。

习 题

1. 6.1.3 节的数据认证算法是由 CBC 模式的 DES 定义的,其中初始向量取为 0,试说明使用 CFB 模式也可获得相同结果。

2. 有很多杂凑函数是由 CBC 模式的分组加密技术构造的,其中的密钥取为消息分组。例如将消息 M 成分组 $M_1, M_2, \dots, M_N, H_0 = \text{初值}$,迭代关系为 $H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1} (i=1, 2, \dots, N)$,杂凑值取为 H_N ,其中 E 是分组加密算法。

(1) 设 E 为 DES,第 3 章习题 1 已证明如果对明文分组和加密密钥都逐比特取补,那么得到的密文也是原密文的逐比特取补,即如果 $Y = \text{DES}_K(X)$,那么 $Y' = \text{DES}_{K'}(X')$ 。利用这一结论证明在上述杂凑函数中可对消息进行修改但却保持杂凑值不变。

(2) 若迭代关系改为 $H_i = E_{H_{i-1}}(M_i) \oplus M_i$, 证明仍可对其进行上述攻击。

3. 考虑用公钥加密算法构造杂凑函数, 设算法是 RSA, 将消息分组后用公开钥加密第一个分组, 加密结果与第二个分组异或后, 再对其加密, 一直进行下去。设一消息被分成两个分组 B_1 和 B_2 , 其杂凑值为 $H(B_1, B_2) = \text{RSA}(\text{RSA}(B_1) \oplus B_2)$ 。证明对任一分组 C_1 可选 C_2 , 使得 $H(C_1, C_2) = H(B_1, B_2)$ 。证明用这种攻击法, 可攻击上述用公钥加密算法构造的杂凑函数。

4. 在图 6-10 中, 假定有 80 个 32 比特长的字用于存储每一 W_i , 因此在处理消息分组前, 可预先计算出这 80 个值。为节省存储空间, 考虑用 16 个字的循环移位寄存器, 其初值存储前 16 个值 (即 W_0, W_1, \dots, W_{15}), 设计一个算法计算以后的每一 W_i 。

5. 对 SHA, 计算 $W_{16}, W_{17}, W_{18}, W_{19}$ 。

6. 设 $a_1 a_2 a_3 a_4$ 是 32 比特长的字中的 4 个字节, 每一 a_i 可看作由二进制表示的 0 到 255 之间的整数, 在 big-endian 结构中, 该字表示整数 $a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$, 在 little-endian 结构中, 该字表示整数 $a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$ 。

(1) MD5 使用 little endian 结构, 因消息的摘要值不应依赖于算法所用的结构, 因此在 MD5 中为了对以 big-endian 结构存储的两个字 $X = x_1 x_2 x_3 x_4$ 和 $Y = y_1 y_2 y_3 y_4$ 进行模 2 加法运算, 必须对这两个字进行调整, 试说明如何调整?

(2) SHA 使用 big-endian 结构, 试说明如何对以 little-endian 结构存储的两个字 X 和 Y 进行模 2 加法运算。

第 7 章

数字签字和密码协议

数字签字由公钥密码发展而来,它在网络安全,包括身份认证、数据完整性、不可否认性以及匿名性等方面有着重要应用。本章首先介绍数字签字的基本概念和一些常用的数字签字算法,然后介绍身份认证协议、身份证明技术以及其他一些常用的密码协议。

7.1 数字签字的基本概念

7.1.1 数字签字应满足的要求

上一章介绍的消息认证其作用是保护通信双方以防第三方的攻击,然而却不能保护通信双方中的一方防止另一方的欺骗或伪造。通信双方之间也可能有多种形式的欺骗,例如通信双方 A 和 B(设 A 为发方,B 为收方)使用图 6-1 所示的消息认证码的基本方式通信,则可能发生以下欺骗:

① B 伪造一个消息并使用与 A 共享的密钥产生该消息的认证码,然后声称该消息来自于 A。

② 由于 B 有可能伪造 A 发来的消息,所以 A 就可以对自己发过的消息予以否认。

这两种欺骗在实际的网络安全应用中都有可能发生,例如在电子资金传输中,收方增加收到的资金数,并声称这一数目来自发方。又如用户通过电子邮件向其证券经纪人发送对某笔业务的指令,以后这笔业务赔钱了,用户就可否认曾发送过相应的指令。

因此在收发双方未建立起完全的信任关系且存在利害冲突的情况下,单纯的消息认证就显得不够。数字签字技术则可有效解决这一问题。类似于手书签字,数字签字应具有以下性质:

① 能够验证签字产生者的身份,以及产生签字的日期和时间。

② 能用于证实被签消息的内容。

③ 数字签字可由第三方验证,从而能够解决通信双方的争议。

由此可见,数字签字具有认证功能。为实现上述 3 条性质,数字签字应满足以下

要求:

- ① 签字的产生必须使用发方独有的一些信息以防伪造和否认。
- ② 签字的产生应较为容易。
- ③ 签字的识别和验证应较为容易。
- ④ 对已知的数字签字构造一新的消息或对已知的消息构造一假冒的数字签字在计算上都是不可行的。

7.1.2 数字签字的产生方式

数字签字的产生可用加密算法或特定的签字算法。

1. 由加密算法产生数字签字

利用加密算法产生数字签字是指将消息或消息的摘要加密后的密文作为对该消息的数字签字,其用法又根据是单钥加密还是公钥加密而有所不同。

(1) 单钥加密

如图 7-1(a)所示,发送方 A 根据单钥加密算法以与接收方 B 共享的密钥 K 对消息 M 加密后的密文作为对 M 的数字签字发往 B。该系统能向 B 保证所收到的消息的确来自 A,因为只有 A 知道密钥 K 。再者 B 恢复出 M 后,可相信 M 未被篡改,因为敌手不知道 K 就不知如何通过修改密文而修改明文。具体来说,就是 B 执行解密运算 $Y=D_K(X)$,如果 X 是合法消息 M 加密后的密文,则 B 得到的 Y 就是明文消息 M ,否则 Y 将是无意义的比特序列。

(2) 公钥加密

如图 7-1(b)所示,发送方 A 使用自己的秘密钥 SK_A 对消息 M 加密后的密文作为对 M 的数字签字,B 使用 A 的公开钥 PK_A 对消息解密,由于只有 A 才拥有加密密钥 SK_A ,因此可使 B 相信自己收到的消息的确来自 A。然而由于任何人都可使用 A 的公开钥解密密文,所以这种方案不提供保密性。为提供保密性,A 可用 B 的公开钥再一次加密,如图 7-1(c)所示。

下面以 RSA 签字体制为例说明数字签字的产生过程。

① 体制参数。

选两个保密的大素数 p 和 q ,计算 $n=p \times q$, $\varphi(n)=(p-1)(q-1)$;选一整数 e ,满足 $1 < e < \varphi(n)$,且 $\gcd(\varphi(n), e) = 1$;计算 d ,满足 $d \cdot e \equiv 1 \pmod{\varphi(n)}$;以 $\{e, n\}$ 为公开钥, $\{d, n\}$ 为秘密钥。

② 签字过程。

设消息为 M ,对其签字为

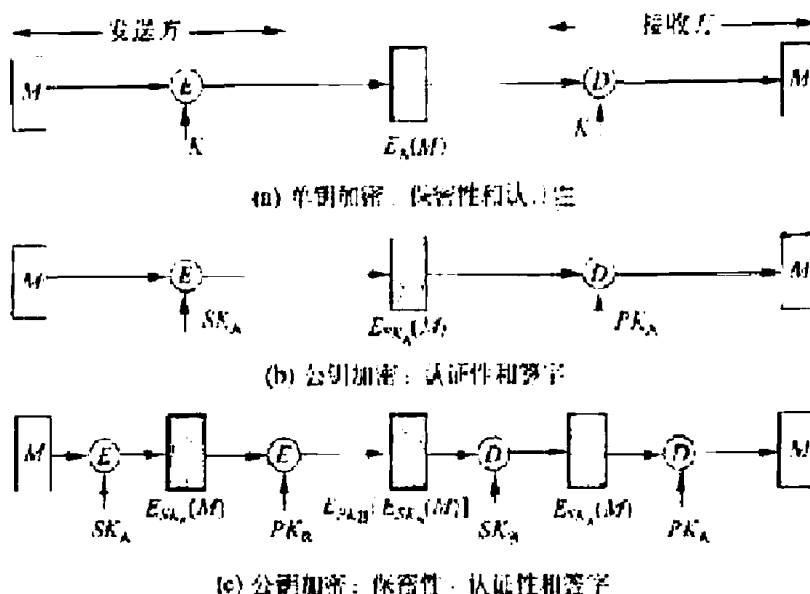


图 7-1 消息加密产生数字签字的基本方式

$$S \equiv M^d \bmod n$$

③ 验证过程。

接收方在收到消息 M 和签字 S 后, 验证 $M \equiv S^e \bmod n$ 是否成立, 若成立, 则发送方的签字有效。

实际应用时, 数字签字是对消息摘要加密产生, 而不是直接对消息加密产生, 如图 6-3(a)~图 6-3(d) 所示。

由加密算法产生数字签字又分为外部保密方式和内部保密方式, 外部保密方式是指数字签字是直接对需要签字的消息生成而不是对已加密的消息生成, 否则称为内部保密方式。外部保密方式便于解决争议, 因为第 3 方在处理争议时, 需得到明文消息及其签字。但如果采用内部保密方式, 第 3 方必须得到消息的解密密钥后才能得到明文消息。如果采用外部保密方式, 接收方就可将明文消息及其数字签字存储下来以备以后万一出现争议时使用。

2. 由签字算法产生数字签字

签字算法的输入是明文消息 M 和密钥 x , 输出是对 M 的数字签字, 表示为 $S = \text{Sig}_x(M)$ 。相应于签字算法, 有一验证算法, 表示为 $\text{Ver}_x(S, M)$, 其取值为

$$\text{Ver}_x(S, M) = \begin{cases} \text{True} & S = \text{Sig}_x(M) \\ \text{False} & S \neq \text{Sig}_x(M) \end{cases}$$

算法的安全性在于从 M 和 S 难以推出密钥 x 或伪造一个消息 M' 使 M' 和 S 可被验

证为真。

7.1.3 数字签字的执行方式

数字签字的执行方式有两类：直接方式和具有仲裁的方式。

1. 直接方式

直接方式是指数字签字的执行过程只有通信双方参与,并假定双方有共享的秘密钥或接收一方知道发方的公开钥。

直接方式的数字签字有一公共弱点,即方案的有效性取决于发方秘密钥的安全性。如果发方想对已发出的消息予以否认,就可声称自己的秘密钥已丢失或被窃,因此自己的签字是他人伪造的。可采取某些行政手段,虽然不能完全避免但可在某种程度上减弱这种威胁。例如,要求每一被签字的消息都包含有一个时戳(日期和时间)并要求密钥丢失后立即向管理机构报告。这种方式的数字签字还存在发方的秘密钥真的被偷的危险,例如敌手在时刻 T 偷得发方的秘密钥,然后可伪造一消息,用偷得的秘密钥为其签字并加上 T 以前的时刻作为时戳。

2. 具有仲裁方式的数字签字

上述直接方式的数字签字所具有的缺陷都可通过使用仲裁者得以解决。和直接方式的数字签字一样,具有仲裁方式的数字签字也有很多实现方案,这些方案都按以下方式运行:发方 X 对发往收方 Y 的消息签字后,将消息及其签字先发给仲裁者 A , A 对消息及其签字验证完后,再连同一个表示已通过验证的指令一起发往收方 Y 。此时由于 A 的存在, X 无法对自己发出的消息予以否认。在这种方式中,仲裁者起着重要的作用并应取得所有用户的信任。

以下是具有仲裁方式数字签字的几个实例,其中 X 表示发方, Y 表示收方, A 是仲裁者, M 是消息, $X \rightarrow Y: M$ 表示 X 给 Y 发送一消息 M 。

【例 7-1】 签字过程如下:

① $X \rightarrow A: M \parallel E_{K_{XA}}[ID_X \parallel H(M)]$ 。

② $A \rightarrow Y: E_{K_{AY}}[ID_X \parallel M \parallel E_{K_{XA}}[ID_X \parallel H(M)] \parallel T]$ 。

其中 E 是单钥加密算法, K_{XA} 和 K_{AY} 分别是 X 与 A 共享的密钥和 A 与 Y 共享的密钥, $H(M)$ 是 M 的杂凑值, T 是时戳, ID_X 是 X 的身份。

在①中, X 以 $E_{K_{XA}}[ID_X \parallel H(M)]$ 作为自己对 M 的签字,将 M 及签字发往 A 。在②中 A 将从 X 收到的内容和 ID_X 、 T 一起加密后发往 Y ,其中的 T 用于向 Y 表示所发的消息不是旧消息的重放。 Y 对收到的内容解密后,将解密结果存储起来以备出现争议时使用。

如果出现争议, Y 可声称自己收到的 M 的确来自 X ,并将

$$E_{K_{AY}}[ID_X \parallel M \parallel E_{K_{XA}}[ID_X \parallel H(M)]]$$

发给 A, 由 A 仲裁, A 由 K_{AY} 解密后, 再用 K_{XA} 对 $E_{K_{XA}}[ID_X \parallel H(M)]$ 解密, 并对 $H(M)$ 加以验证, 从而验证了 X 的签字。

以上过程中, 由于 Y 不知 K_{XA} , 因此不能直接检查 X 的签字, 但 Y 认为消息来自于 A 因而是可信的。所以在整个过程中, A 必须取得 X 和 Y 的高度信任:

- X 相信 A 不会泄露 K_{XA} , 并且不会伪造 X 的签字;
- Y 相信 A 只有在对 $E_{K_{AY}}[ID_X \parallel M \parallel E_{K_{XA}}[ID_X \parallel H(M)]] \parallel T$ 中的杂凑值及 X 的签字验证无误后才将之发给 Y;
- X, Y 都相信 A 可公正地解决争议。

如果 A 已取得各方的信任, 则 X 就能相信没有人能伪造自己的签字, Y 就可相信 X 不能对自己的签字予以否认。

本例中消息 M 是以明文形式发送的, 因此未提供保密性, 下面两个例子可提供保密性。

【例 7-2】 签字过程如下:

$$\textcircled{1} X \rightarrow A: ID_X \parallel E_{K_{XY}}[M] \parallel E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}[M])].$$

$$\textcircled{2} A \rightarrow Y: E_{K_{AY}}[ID_X \parallel E_{K_{XY}}[M] \parallel E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}[M])]] \parallel T].$$

其中 K_{XY} 是 X, Y 共享的密钥, 其他符号与例 7-1 相同。X 以 $E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}[M])]$ 作为对 M 的签字, 与由 K_{XY} 加密的消息 M 一起发给 A。A 对 $E_{K_{XA}}[ID_X \parallel H(E_{K_{XY}}[M])]$ 解密后通过验证杂凑值以验证 X 的签字, 但始终未能读取明文 M。A 验证完 X 的签字后, 对 X 发来的消息加一时戳, 再用 K_{AY} 加密后发往 Y。解决争议的方法与例 7-1 一样。

本例虽然提供了保密性, 但还存在与上例相同的一个问题, 即仲裁者可和发方共谋以否认发方曾发过的消息, 也可和收方共谋以伪造发方的签字。这一问题可通过下例所示的采用公钥加密技术的方法得以解决。

【例 7-3】 签字过程如下:

$$\textcircled{1} X \rightarrow A: ID_X \parallel E_{SK_X}[ID_X \parallel E_{PK_Y}[E_{SK_X}[M]]].$$

$$\textcircled{2} A \rightarrow Y: E_{SK_A}[ID_X \parallel E_{PK_Y}[E_{SK_X}[M]]] \parallel T].$$

其中 SK_A 和 SK_X 分别是 A 和 X 的秘密钥, PK_Y 是 Y 的公开钥, 其他符号与前两例相同。第①步中, X 用自己的秘密钥 SK_X 和 Y 的公开钥 PK_Y 对消息加密后作为对 M 的签字, 以这种方式使得任何第 3 方(包括 A)都不能得到 M 的明文消息。A 收到 X 发来的内容后, 用 X 的公开钥可对 $E_{SK_X}[ID_X \parallel E_{PK_Y}[E_{SK_X}[M]]]$ 解密, 并将解密得到的 ID_X 与收到的 ID_X 加以比较, 从而可确信这一消息是来自于 X 的(因只有 X 有 SK_X)。第②步, A 将 X 的身份 ID_X 和 X 对 M 的签字加上一时戳后, 再用自己的秘密钥加密发往 Y。

与前两种方案相比,第3种方案有很多优点。首先,在协议执行以前,各方都不必有共享的信息,从而可防止共谋。第二,只要仲裁者的秘密钥不被泄露,任何人包括发方就不能发送重放的消息。最后,对任何第三方(包括A)来说,X发往Y的消息都是保密的。

7.2 数字签字标准

数字签字标准 DSS(digital signature standard)是由美国 NIST 公布的联邦信息处理标准 FIPS PUB 186,其中采用了上一章介绍的 SHA 和一种新的签字技术,称为 DSA(digital signature algorithm)。DSS 最初于 1991 年公布,在考虑了公众对其安全性的反馈意见后,于 1993 年公布了其修改版。

7.2.1 DSS 的基本方式

首先将 DSS 与 RSA 的签字方式做一比较。RSA 算法既能用于加密和签字,又能用于密钥交换。与此不同,DSS 使用的算法只能提供数字签字功能。图 7-2 用于比较 RSA 签字和 DSS 签字的不同方式。

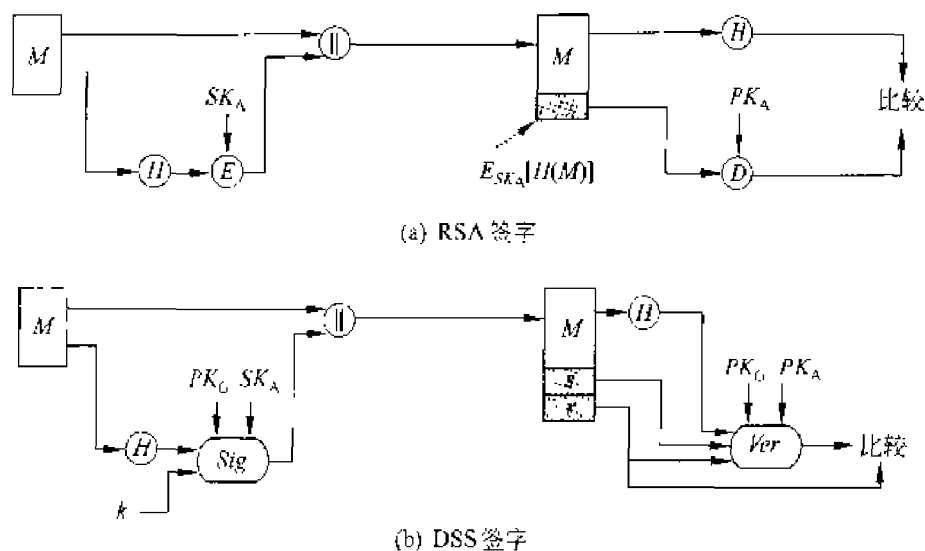


图 7-2 RSA 签字与 DSS 签字的不同方式

采用 RSA 签字时,将消息输入到一个杂凑函数以产生一个固定长度的安全杂凑值,再用发方的秘密钥加密杂凑值就形成了对消息的签字。消息及其签字被一起发给接收方,接收方得到消息后再产生出消息的杂凑值,且使用发送方的公开钥对收到的签字解密。这样

接收方就得到了两个杂凑值,如果两个杂凑值是一样的,则认为收到的签字是有效的。

DSS 签字也利用一杂凑函数产生消息的一个杂凑值,杂凑值连同一随机数 k 一起作为签字函数的输入,签字函数还需使用发送方的秘密钥 SK_A 和供所有用户使用的一族参数,称这一族参数为全局公开钥 PK_G 。签字函数的两个输出 s 和 r 就构成了消息的签字 (s, r) 。接收方收到消息后再产生出消息的杂凑值,将杂凑值与收到的签字一起输入验证函数,验证函数还需输入全局公开钥 PK_G 和发送方的公开钥 PK_A 。验证函数的输出如果与收到的签字成分 r 相等,则验证了签字是有效的。

7.2.2 数字签字算法 DSA

DSA 是在 ElGamal 和 Schnorr 两个签字方案(见下一节)的基础上设计的,其安全性基于求离散对数的困难性。

算法描述如下:

(1) 全局公开钥

p : 满足 $2^{L-1} < p < 2^L$ 的大素数,其中 $512 \leq L \leq 1024$ 且 L 是 64 的倍数。

q : $p-1$ 的素因子,满足 $2^{159} < q < 2^{160}$,即 q 长为 160 比特。

g : $g \equiv h^{(p-1)/q} \pmod{p}$,其中 h 是满足 $1 < h < p-1$ 且使得 $h^{(p-1)/q} \pmod{p} > 1$ 的任一整数。

(2) 用户秘密钥 x

x 是满足 $0 < x < q$ 的随机数或伪随机数。

(3) 用户的公开钥 y

$$y \equiv g^x \pmod{p}.$$

(4) 用户为待签消息选取的秘密数 k

k 是满足 $0 < k < q$ 的随机数或伪随机数。

(5) 签字过程

用户对消息 M 的签字为 (r, s) ,

其中 $r \equiv (g^k \pmod{p}) \pmod{q}$, $s \equiv [k^{-1}(H(M) + xr)] \pmod{q}$, $H(M)$ 是由 SHA 求出的杂凑值。

(6) 验证过程

设接收方收到的消息为 M' , 签字为 (r', s') 。计算

$$w \equiv (s')^{-1} \pmod{q}, \quad u_1 \equiv [H(M')w] \pmod{q}$$

$$u_2 \equiv r'w \pmod{q}, \quad v \equiv [(g^{u_1} y^{u_2}) \pmod{p}] \pmod{q}$$

检查 $v \stackrel{?}{=} r'$, 若相等, 则认为签字有效。这是因为若 $(M', r', s') = (M, r, s)$, 则

$$\begin{aligned}
 v &\equiv [(g^{H(M)w} g^{xr}) \bmod p] \bmod q \equiv [g^{(H(M)+xr)s^{-1}} \bmod p] \bmod q \\
 &\equiv (g^k \bmod p) \bmod q \equiv r
 \end{aligned}$$

算法的框图如图 7-3 所示,其中的 4 个函数分别为

$$s \equiv f_1[H(M), k, x, r, q] \equiv [k^{-1}(H(M) + xr)] \bmod q$$

$$r = f_2(k, p, q, g) \equiv (g^k \bmod p) \bmod q$$

$$w = f_3(s', q) \equiv (s')^{-1} \bmod q$$

$$v = f_4(y, q, g, H(M'), w, r') \equiv [(g^{H(M')w} \bmod q) y^{r'w} \bmod q] \bmod p] \bmod q$$

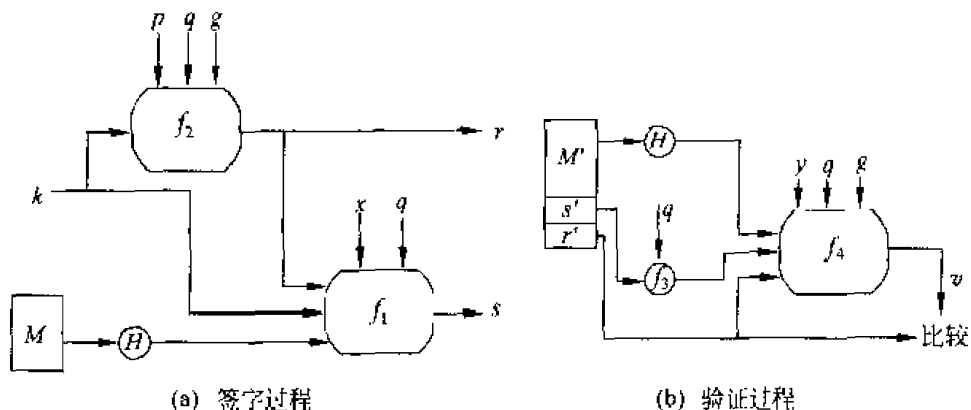


图 7-3 DSA 的框图

由于离散对数的困难性,敌手从 r 恢复 k 或从 s 恢复 x 都是不可行的。

还有一个问题值得注意,即签字产生过程中的运算主要是求 r 的模指数运算 $r = (g^k \bmod p) \bmod q$,而这一运算与待签的消息无关,因此能被预先计算。事实上,用户可以预先计算出很多 r 和 k^{-1} 以备以后的签字使用,从而可大大加快产生签字的速度。

7.3 其他签字方案

7.3.1 基于离散对数问题的数字签身体制

基于离散对数问题的数字签身体制是数字签身体制中最为常用的一类,其中包括 ElGamal 签身体制、DSA 签身体制、Okamoto 签身体制等。

1. 离散对数签身体制

ElGamal、DSA、Okamoto 等签身体制都可归结为离散对数签身体制的特例。

(1) 体制参数

 p : 大素数; q : $p-1$ 或 $p-1$ 的大素因子; g : $g \in {}_R Z_p^*$, 且 $g^q \equiv 1 \pmod{p}$, 其中 $g \in {}_R Z_p^*$ 表示 g 是从 Z_p^* 中随机选取的, 其中 $Z_p^* = Z_p - \{0\}$; x : 用户 A 的秘密钥, $1 < x < q$; y : 用户 A 的公开钥, $y \equiv g^x \pmod{p}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:① 计算 m 的杂凑值 $H(m)$ 。② 选择随机数 k : $1 < k < q$, 计算 $r \equiv g^k \pmod{p}$ 。③ 从签字方程 $ak \equiv b + cx \pmod{q}$ 中解出 s 。方程的系数 a, b, c 有许多种不同的选择方法, 表 7-1 给出了这些可能选择中的一小部分, 以 (r, s) 作为产生的数字签字。表 7-1 参数 a, b, c 可能的置换取值表

$\pm r'$	$\pm s$	$H(m)$
$\pm r' H(m)$	$\pm s$	1
$\pm r' H(m)$	$\pm H(m)s$	1
$\pm H(m)r'$	$\pm r's$	1
$\pm H(m)s$	$\pm r's$	1

(3) 签字的验证过程

接收方在收到消息 m 和签字 (r, s) 后, 可以按照以下验证方程检验:

$$Ver(y, (r, s), m) = True \Leftrightarrow r^s \equiv g^b y^c \pmod{p}$$

2. ElGamal 签身体制

(1) 体制参数

 p : 大素数; g : Z_p^* 的一个生成元; x : 用户 A 的秘密钥, $x \in {}_R Z_p^*$; y : 用户 A 的公开钥, $y \equiv g^x \pmod{p}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:① 计算 m 的杂凑值 $H(m)$ 。② 选择随机数 k : $k \in Z_p^*$, 计算 $r \equiv g^k \pmod{p}$ 。

③ 计算 $s \equiv (H(m) - xr)k^{-1} \pmod{p-1}$ 。

以 (r, s) 作为产生的数字签字。

(3) 签字验证过程

接收方在收到消息 m 和数字签字 (r, s) 后, 先计算 $H(m)$, 并按下式验证:

$$\text{Ver}(y, (r, s), H(m)) = \text{True} \Leftrightarrow y^r r^s \equiv g^{H(m)} \pmod{p}$$

正确性可由下式证明:

$$y^r r^s \equiv g^{rx} g^{ks} \equiv g^{rx + H(m) - rx} \equiv g^{H(m)} \pmod{p}。$$

3. Schnorr 签身体制

(1) 体制参数

p : 大素数, $p \geq 2^{512}$;

q : 大素数, $q | (p-1)$, $q \geq 2^{160}$;

g : $g \in {}_R Z_p^*$, 且 $g^q \equiv 1 \pmod{p}$;

x : 用户 A 的秘密钥, $1 < x < q$;

y : 用户 A 的公开钥, $y \equiv g^x \pmod{p}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:

① 选择随机数 k : $1 < k < q$, 计算 $r \equiv g^k \pmod{p}$ 。

② 计算 $e = H(r, m)$ 。

③ 计算 $s \equiv xe + k \pmod{q}$ 。

以 (e, s) 作为产生的数字签字。

(3) 签字验证过程

接收方在收到消息 m 和数字签字 (e, s) 后, 先计算 $r' \equiv g^e y^{-s} \pmod{p}$, 然后计算 $H(r', m)$, 并按下式验证

$$\text{Ver}(y, (e, s), m) = \text{True} \Leftrightarrow H(r', m) = e$$

其正确性可由下式证明:

$$r' = g^e y^{-s} \equiv g^{xe + k - xe} \equiv g^k \equiv r \pmod{p}。$$

4. Neberg-Rueppel 签身体制

该体制是一个消息恢复式签身体制, 即验证人可从签字中恢复出原始消息, 因此签字人不需要将被签消息发送给验证人。

(1) 体制参数

p : 大素数;

q : 大素数, $q | (p-1)$;

$g: g \in {}_R Z_p^*$, 且 $g^q \equiv 1 \pmod{p}$;

x : 用户 A 的秘密钥, $x \in {}_R Z_p^*$;

y : 用户 A 的公开钥, $y \equiv g^x \pmod{p}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:

① 计算出 $\tilde{m} = R(m)$, 其中 R 是一个单一映射, 并且容易求逆, 称为冗余函数。

② 选择一个随机数 k ($0 < k < q$), 计算 $r \equiv g^{-k} \pmod{p}$ 。

③ 计算 $e \equiv \tilde{m}r \pmod{p}$ 。

④ 计算 $s \equiv xe + k \pmod{q}$ 。

以 (e, s) 作为对 m 的数字签字。

(3) 签字的验证过程

接收方收到数字签字 (e, s) 后, 通过以下步骤来验证签字的有效性:

① 验证是否 $0 < e < p$ 。

② 验证是否 $0 \leq s < q$ 。

③ 计算 $v \equiv g^s y^{-e} \pmod{p}$ 。

④ 计算 $m' \equiv ve \pmod{p}$ 。

⑤ 验证是否 $m' \in \mathcal{R}(m)$, 其中 $\mathcal{R}(m)$ 表示 R 的值域。

⑥ 恢复出 $m = R^{-1}(m')$ 。

这个签身体制的正确性可以由以下等式证明:

$$m' = ve \pmod{p} \equiv g^s y^{-e} e \pmod{p} \equiv g^{sr+ke} e \pmod{p} \equiv g^k e \pmod{p} = \tilde{m}$$

5. Okamoto 签身体制

(1) 体制参数

p : 大素数, 且 $p \geq 2^{512}$;

q : 大素数, $q \mid (p-1)$, 且 $q \geq 2^{146}$;

g_1, g_2 : 两个与 q 同长的随机数;

x_1, x_2 : 用户 A 的秘密钥, 两个小于 q 的随机数;

y : 用户 A 的公开钥, $y \equiv g_1^{-x_1} g_2^{-x_2} \pmod{p}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:

① 选择两个小于 q 的随机数 $k_1, k_2 \in {}_R Z_q^*$ 。

② 计算杂凑值: $e \equiv H(g_1^{k_1} g_2^{k_2} \pmod{p}, m)$ 。

③ 计算: $s_1 \equiv (k_1 + ex_1) \pmod{q}$ 。

④ 计算: $s_2 \equiv (k_2 + ex_2) \pmod{q}$ 。

以 (e, s_1, s_2) 作为对 m 的数字签字。

(3) 签字的验证过程

接收方在收到消息 m 和数字签字 (e, s_1, s_2) 后, 通过以下步骤来验证签字的有效性:

① 计算 $v \equiv g_1^{s_1} g_2^{s_2} y^e \pmod{p}$ 。

② 计算 $e' = H(v, m)$ 。

③ 验证: $Ver(y, (e, s_1, s_2), m) = True \Leftrightarrow e' = e$ 。

其正确性可通过下式证明:

$$v \equiv g_1^{s_1} g_2^{s_2} y^e \pmod{p} = g_1^{k_1 + ex_1} g_2^{k_2 + ex_2} g_1^{x_1 e} g_2^{-x_2 e} \pmod{p} \equiv g_1^{k_1} g_2^{k_2} \pmod{p}。$$

7.3.2 基于大数分解问题的数字签字体制

设 n 是一个大合数, 找出 n 的所有素因子是一个困难问题, 称之为大数分解问题。下面介绍的两个数字签字体制都基于这个问题的困难性。

1. Fiat-Shamir 签字体制

(1) 体制参数

n : $n = pq$, 其中 p 和 q 是两个保密的大素数;

k : 固定的正整数;

y_1, y_2, \dots, y_k : 用户 A 的公开钥, 对任何 $i (1 \leq i \leq k)$, y_i 都是模 n 的平方剩余;

x_1, x_2, \dots, x_k : 用户 A 的秘密钥, 对任何 $i (1 \leq i \leq k)$, $x_i \equiv \sqrt{y_i}^{-1} \pmod{n}$ 。

(2) 签字的产生过程

对于待签字的消息 m , A 执行以下步骤:

① 随机选取一个正整数 t 。

② 随机选取 t 个介于 1 和 n 之间的数 r_1, r_2, \dots, r_t , 并对任何 $j (1 \leq j \leq t)$, 计算 $R_j \equiv r_j^2 \pmod{n}$ 。

③ 计算杂凑值 $H(m, R_1, R_2, \dots, R_t)$, 并依次取出 $H(m, R_1, R_2, \dots, R_t)$ 的前 kt 个比特值 $b_{11}, \dots, b_{1t}, b_{21}, \dots, b_{2t}, \dots, b_{k1}, \dots, b_{kt}$ 。

④ 对任何 $j (1 \leq j \leq t)$, 计算 $s_j \equiv r_j \prod_{i=1}^k x_i^{b_{ij}} \pmod{n}$ 。

以 $((b_{11}, \dots, b_{1t}, b_{21}, \dots, b_{2t}, \dots, b_{k1}, \dots, b_{kt}), (s_1, \dots, s_t))$ 作为对 m 的数字签字。

(3) 签字的验证过程

收方在收到消息 m 和签字 $((b_{11}, \dots, b_{1t}, b_{21}, \dots, b_{2t}, \dots, b_{k1}, \dots, b_{kt}), (s_1, \dots, s_t))$ 后, 用

以下步骤来验证:

- ① 对任何 $j(1 \leq j \leq t)$, 计算 $R'_j \equiv s_j^2 \cdot \prod_{i=1}^t y_i^{b_{ij}} \pmod{n}$ 。
- ② 计算 $H(m, R'_1, R'_2, \dots, R'_t)$ 。
- ③ 验证 $b_{11}, \dots, b_{1t}, b_{21}, \dots, b_{2t}, \dots, b_{k1}, \dots, b_{kt}$ 是否依次是 $H(m, R'_1, R'_2, \dots, R'_t)$ 的前 kt 个比特。如果是, 则以上数字签字是有效的。

正确性可以由以下算式证明:

$$\begin{aligned} R'_j &\equiv s_j^2 \cdot \prod_{i=1}^k y_i^{b_{ij}} \pmod{n} \equiv \left(r_j \cdot \prod_{i=1}^k x_i^{b_{ij}} \right)^2 \cdot \prod_{i=1}^k y_i^{b_{ij}} \\ &\equiv r_j^2 \cdot \prod_{i=1}^k (x_i^2 y_i)^{b_{ij}} \equiv r_j^2 \equiv R \pmod{n}。 \end{aligned}$$

2. Guillou-Quisquater 签字体制

(1) 体制参数

n : $n = pq$, p 和 q 是两个保密的大素数;

v : $\gcd(v, (p-1)(q-1)) = 1$;

x : 用户 A 的秘密钥, $x \in {}_R Z_n^*$;

y : 用户 A 的公开钥, $y \in Z_n^*$, 且 $x^v y \equiv 1 \pmod{n}$ 。

(2) 签字的产生过程

对于待签消息 m , A 进行以下步骤:

- ① 随机选择一个数 $k \in Z_n^*$, 计算 $T \equiv k^v \pmod{n}$ 。
- ② 计算杂凑值: $e \equiv H(m, T)$, 且使 $1 \leq e < v$; 否则, 返回步骤①。
- ③ 计算 $s \equiv kx^e \pmod{n}$ 。

以 (e, s) 作为对 m 的签字。

(3) 签字的验证过程

接收方在收到消息 m 和数字签字 (e, s) 后, 用以下步骤来验证:

- ① 计算出 $T' \equiv s^v y^e \pmod{n}$ 。
- ② 计算出 $e' \equiv H(m, T')$ 。
- ③ 验证: $\text{Ver}(y, (e, s), m) = \text{True} \Leftrightarrow e' = e$ 。

正确性可由以下算式证明:

$$\begin{aligned} T' &\equiv s^v y^e \pmod{n} \equiv (kx^e)^v y^e \pmod{n} \equiv k^v (x^v y)^e \pmod{n} \\ &\equiv k^v \pmod{n} = T。 \end{aligned}$$

7.4 认证协议

6.1 节介绍过消息认证的基本概念,事实上安全可靠的通信除需进行消息的认证外,还需建立一些规范的协议对数据来源的可靠性、通信实体的真实性加以认证,以防止欺骗、伪装等攻击。本节以网络通信的一个基本问题的解决引出认证协议的基本意义,这一基本问题陈述如下: A 和 B 是网络的两个用户,他们想通过网络先建立安全的共享密钥再进行保密通信。那么 A(B)如何确信自己正在和 B(A)通信而不是和 C 通信呢? 这种通信方式为双向通信,因此,此时的认证称为相互认证。类似地,对于单向通信来说,认证称为单向认证。

7.4.1 相互认证

A、B 两个用户在建立共享密钥时需要考虑的核心问题是保密性和实时性。为了防止会话密钥的伪造或泄露,会话密钥在通信双方之间交换时应为密文形式,所以通信双方事先就应有密钥或公开钥。第 2 个问题实时性则对防止消息的重放攻击极为重要,实现实时性的一种方法是对交换的每一条消息都加上一个序列号,一个新消息仅当它有正确的序列号时才被接收。但这种方法的困难性是要求每个用户分别记录与其他每一用户交换的消息的序列号,从而增加了用户的负担,所以序列号方法一般不用于认证和密钥交换。保证消息的实时性常用以下两种方法:

- 时戳 如果 A 收到的消息包括一时戳,且在 A 看来这一时戳充分接近自己的当前时刻, A 才认为收到的消息是新的并接受之。这种方案要求所有各方的时钟是同步的。

- 询问-应答 用户 A 向 B 发出一个一次性随机数作为询问,如果收到 B 发来的消息(应答)也包含一正确的一次性随机数, A 就认为 B 发来的消息是新的并接受之。

其中时戳法不能用于面向连接的应用过程,这是由于时戳法在实现时固有的困难性。首先是需要不同的处理器时钟之间保持同步,那么所用的协议必须是容错的以处理网络错误,并且是安全的以对付恶意攻击。第二,如果协议中任一方的时钟出现错误而暂时地失去了同步,则将使敌手攻击成功的可能性增加。最后还由于网络本身存在着延迟,因此不能期望协议的各方能保持精确的同步。所以任何基于时戳的处理过程、协议等都必须允许同步有一个误差范围。考虑到网络本身的延迟,误差范围应足够大;考虑到可能存在的攻击,误差范围又应足够小。

而询问-应答方式则不适合于无连接的应用过程,这是因为在无连接传输以前需经

询问-应答这一额外的握手过程,这与无连接应用过程的本质特性不符。对无连接的应用程序来说,利用某种安全的时间服务器保持各方时钟同步是防止重放攻击最好的方法。

通信双方建立共享密钥时可采用单钥加密体制和公钥加密体制。

1. 单钥加密体制

正如 5.1 节所介绍的,采用单钥加密体制为通信双方建立共享的密钥时,需要有一个可信的密钥分配中心 KDC,网络中每一用户都与 KDC 有一共享的密钥,称为主密钥。KDC 为通信双方建立一个短期内使用的密钥,称为会话密钥,并用主密钥加密会话密钥后分配给两个用户。这种分配密钥的方式在实际应用中较为普遍采用,如下一章介绍的 Kerberos 系统采用的就是这种方式。

(1) Needham-Schroeder 协议

5.1 节中图 5-1 所示的采用 KDC 的密钥分配过程,可用以下协议(称为 Needham-Schroeder 协议)来描述:

- ① $A \rightarrow KDC: ID_A \parallel ID_B \parallel N_1$
- ② $KDC \rightarrow A: E_{K_A}[K_s \parallel ID_B \parallel N_1 \parallel E_{K_B}[K_s \parallel ID_A]]$
- ③ $A \rightarrow B: E_{K_B}[K_s \parallel ID_A]$
- ④ $B \rightarrow A: E_{K_s}[N_2]$
- ⑤ $A \rightarrow B: E_{K_s}[f(N_2)]$

式中 K_A 、 K_B 分别是 A、B 与 KDC 共享的主密钥。协议的目的是由 KDC 为 A、B 安全地分配会话密钥 K_s ,A 在第②步安全地获得了 K_s ,而第③步的消息仅能被 B 解读,因此 B 在第③步安全地获得了 K_s ,第④步中 B 向 A 示意自己已掌握 K_s , N_2 用于向 A 询问自己在第③步收到的 K_s 是否为一新会话密钥,第⑤步 A 对 B 的询问作出应答,一方面表示自己已掌握 K_s ,另一方面由 $f(N_2)$ 回答了 K_s 的新鲜性。可见第④、⑤两步用于防止一种类型的重放攻击,比如敌手在前一次执行协议时截获第③步的消息,然后在这次执行协议时重放,如果双方没有第④、⑤两步的握手过程的话,B 就无法检查出自己得到的 K_s 是重放的旧密钥。

然而以上协议却易遭受另一种重放攻击,假定敌手能获取旧会话密钥,则冒充 A 向 B 重放第③步的消息后,就可欺骗 B 使用旧会话密钥。敌手进一步截获第④步 B 发出的询问后,可假冒 A 作出第⑤步的应答。进而,敌手就可冒充 A 使用经认证过的会话密钥向 B 发送假消息。

(2) Needham-Schroeder 协议的改进方案之一

为克服以上弱点,可在第②步和第③步加上一时戳,改进后的协议如下:

- ① $A \rightarrow KDC: ID_A \parallel ID_B$

② $KDC \rightarrow A: E_{K_A}[K_S \parallel ID_B \parallel T \parallel E_{K_B}[K_S \parallel ID_A \parallel T]]$

③ $A \rightarrow B: E_{K_B}[K_S \parallel ID_A \parallel T]$

④ $B \rightarrow A: E_{K_S}[N_1]$

⑤ $A \rightarrow B: E_{K_S}[f(N_1)]$

其中 T 是时戳,用以向 A、B 双方保证 K_S 的新鲜性。A 和 B 可通过下式检查 T 的实时性:

$$|\text{Clock} - T| < \Delta t_1 + \Delta t_2$$

其中 Clock 为用户(A 或 B)本地的时钟, Δt_1 是用户本地时钟和 KDC 时钟误差的估计值, Δt_2 是网络的延迟时间。

以上协议中由于 T 是经主密钥加密的,所以敌手即使知道旧会话密钥,并在协议的去执行期间截获第③步的结果,也无法成功地重放给 B,因 B 对收到的消息可通过时戳检查其是否为新的。

以上改进还存在以下问题:方案主要依赖网络中各方时钟的同步,这种同步可能会由于系统故障或计时误差而被破坏。如果发送方的时钟超前于接收方的时钟,敌手就可截获发送方发出的消息,等待消息中时戳接近于接收方的时钟时,再重发这个消息。这种攻击称为等待重放攻击。

抗击等待重放攻击的一种方法是要求网络中各方以 KDC 的时钟为基准定期检查并调整自己的时钟,另一种方法是使用一次性随机数的握手协议,因为接收方向发送方发出询问的随机数是他人无法事先预测的,所以敌手即使实施等待重放攻击,也可被下面的握手协议检查出来。

(3) Needham-Schroeder 协议的改进方案之二

下面的协议可解决 Needham-Schroeder 协议以及改进方案一可能遭受的攻击:

① $A \rightarrow B: ID_A \parallel N_A$

② $B \rightarrow KDC: ID_B \parallel N_B \parallel E_{K_B}[ID_A \parallel N_A \parallel T_B]$

③ $KDC \rightarrow A: E_{K_A}[ID_B \parallel N_A \parallel K_S \parallel T_B] \parallel E_{K_B}[ID_A \parallel K_S \parallel T_B] \parallel N_B$

④ $A \rightarrow B: E_{K_B}[ID_A \parallel K_S \parallel T_B] \parallel E_{K_A}[N_B]$

协议的具体含义如下:

① A 将新产生的一次性随机数 N_A 与自己的身份 ID_A 一起以明文形式发往 B, N_A 以后将与会话密钥 K_S 一起以加密形式返回给 A,以保证 A 收到的会话密钥的新鲜性。

② B 向 KDC 发出与 A 建立会话密钥的请求,表示请求的消息包括 B 的身份、一次性随机数 N_B 以及由 B 与 KDC 共享的主密钥加密的数据项。其中 N_B 以后将与会话密钥一起以加密形式返回给 B 以向 B 保证会话密钥的新鲜性,请求中由主密钥加密的数据项用于指示 KDC 向 A 发出一个证书,其中的数据项有证书接收者 A 的身份、B 建议的证书

截止时间 T_B 、B 从 A 收到的一次性随机数。

③ KDC 将 B 产生的 N_B 连同由 KDC 与 B 共享的密钥 K_B 加密的 $ID_A \parallel K_S \parallel T_B$ 一起发给 A, 其中 K_S 是 KDC 分配的会话密钥, $E_{K_B}[ID_A \parallel K_S \parallel T_B]$ 由 A 当作票据用于以后的认证。KDC 向 A 发出的消息还包括由 KDC 与 A 共享的主密钥加密的 $ID_B \parallel N_A \parallel K_S \parallel T_B$, A 用这一消息可验证 B 已收到第①步发出的消息(通过 ID_B), A 还能验证这一步收到的消息是新的(通过 N_A), 这一消息中还包括 KDC 分配的会话密钥 K_S 以及会话密钥的截止时间 T_B 。

④ A 将票据 $E_{K_B}[ID_A \parallel K_S \parallel T_B]$ 连同由会话密钥加密的一次性随机数 N_B 发往 B, B 由票据得到会话密钥 K_S , 并由 K_S 得 N_B 。 N_B 由会话密钥加密的目的是 B 认证了自己收到的消息不是一个重放, 而的确是来自于 A。

以上协议为 A、B 双方建立共享的会话密钥提供了一个安全有效的手段。再者, 如果 A 保留由协议得到的票据, 就可在有效时间范围内不再求助于认证服务器而由以下方式实现双方的新认证:

① $A \rightarrow B: E_{K_B}[ID_A \parallel K_S \parallel T_B], N'_A$

② $B \rightarrow A: N'_B, E_{K_S}[N'_A]$

③ $A \rightarrow B: E_{K_S}[N'_B]$

B 在第①步收到票据后, 可通过 T_B 检验票据是否过时, 而新产生的一次性随机数 N'_A 、 N'_B 则向双方保证了没有重放攻击。

以上协议中时间期限 T_B 是 B 根据自己的时钟定的, 因此不要求各方之间的同步。

2. 公钥加密体制

第 5 章曾介绍过使用公钥加密体制分配会话密钥的方法, 下面的协议也用于这个目的。

① $A \rightarrow AS: ID_A \parallel ID_B$

② $AS \rightarrow A: E_{SK_{AS}}[ID_A \parallel PK_A \parallel T] \parallel E_{SK_{AS}}[ID_B \parallel PK_B \parallel T]$

③ $A \rightarrow B: E_{SK_{AS}}[ID_A \parallel PK_A \parallel T] \parallel E_{SK_{AS}}[ID_B \parallel PK_B \parallel T] \parallel E_{PK_B}[E_{SK_A}[K_S \parallel T]]$

其中 SK_{AS} 、 SK_A 分别是 AS 和 A 的秘密钥, PK_A 、 PK_B 分别是 A 和 B 的公开钥, E 为公钥加密算法, AS 是认证服务器(authentication server)。第①步, A 将自己的身份及欲通信的对方的身份发送给 AS。第②步, AS 发给 A 的两个链接的数据项都是由自己的秘密钥加密(即由 AS 签字), 分别作为发放给通信双方的公钥证书。第③步, A 选取会话密钥并经自己的秘密钥和 B 的公开钥加密后连同两个公钥证书一起发往 B。因会话密钥是由 A 选取, 并以密文形式发送给 B, 因此包括 AS 在内的任何第 3 者都无法得到会话密钥。时戳 T 用以防止重放攻击, 所以需要各方的时钟是同步的。

下一协议使用一次性随机数,因此不需要时钟的同步:

- ① $A \rightarrow KDC: ID_A \parallel ID_B$
- ② $KDC \rightarrow A: E_{SK_{AU}}[ID_B \parallel PK_B]$
- ③ $A \rightarrow B: E_{PK_B}[N_A \parallel ID_A]$
- ④ $B \rightarrow KDC: ID_B \parallel ID_A \parallel E_{PK_{AU}}[N_A]$
- ⑤ $KDC \rightarrow B: E_{SK_{AU}}[ID_A \parallel PK_A] \parallel E_{PK_B}[E_{SK_{AU}}[N_A \parallel K_S \parallel ID_B]]$
- ⑥ $B \rightarrow A: E_{PK_A}[E_{SK_{AU}}[N_A \parallel K_S \parallel ID_B] \parallel N_B]$
- ⑦ $A \rightarrow B: E_{K_S}[N_B]$

其中 SK_{AU} 和 PK_{AU} 分别是 KDC 的秘密钥和公开钥。第①步, A 通知 KDC 他想和 B 建立安全连接。第②步, KDC 将 B 的公钥证书发给 A, 公钥证书包括经 KDC 签字的 B 的身份和公钥。第③步, A 告诉 B 想与他通信, 并将自己选择的一次性随机数 N_A 发给 B。第④步, B 向 KDC 发出得到 A 的公钥证书和会话密钥的请求, 请求中由 KDC 的公开钥加密的 N_A 用于让 KDC 将建立的会话密钥与 N_A 联系起来, 以保证会话密钥的新鲜性。第⑤步, KDC 向 B 发出 A 的公钥证书以及由自己的秘密钥和 B 的公开钥加密的三元组 $\{N_A, K_S, ID_B\}$ 。三元组由 KDC 的秘密钥加密可使 B 验证三元组的确是由 KDC 发来的, 由 B 的公开钥加密是防止他人得到三元组后假冒 B 建立与 A 的连接。第⑥步, B 新产生一个一次性随机数 N_B , 连同上一步收到的由 KDC 的秘密钥加密的三元组一起经 A 的公开钥加密后发往 A。第⑦步, A 取出会话密钥, 再由会话密钥加密 N_B 后发往 B, 以使 B 知道 A 已掌握会话密钥。

以上协议可进一步做如下改进: 在第⑤、⑥两步出现 N_A 的地方加上 ID_A , 以说明 N_A 的确是由 A 产生的而不是其他人产生的, 这时 $\{ID_A, N_A\}$ 就可惟一地识别 A 发出的连接请求。

7.4.2 单向认证

电子邮件等网络应用有一个最大的优点就是不要求收发双方同时在线, 发送方将邮件发往接收方的信箱, 邮件在信箱中存着, 直到接收方阅读时才打开。邮件消息的报头必须是明文形式以使 SMTP (simple mail transfer protocol-简单邮件传输协议) 或 X.400 等存储-转发协议能够处理。然而通常都不希望邮件处理协议要求邮件的消息本身是明文形式, 否则就要求用户对邮件处理机制的信任。所以用户在进行保密通信时, 需对邮件消息进行加密以使包括邮件处理系统在内的任何第 3 者都不能读取邮件的内容。再者邮件接收者还希望对邮件的来源即发方的身份进行认证, 以防他人的假冒。与双向认证一样, 在此仍分为单钥加密和公钥加密两种情况来考虑。

1. 单钥加密

对诸如电子邮件等单向通信来说,图 5-2 所示的无中心的密钥分配情况不适用。因为该方案要求发送方给接收方发送一请求,并等到接收方发回一个包含会话密钥的应答后,才向接收方发送消息,所以本方案与接收方和发送方不必同时在线的要求不符。在图 5-1 所示的情况中去掉第④步和第⑤步就可满足单向通信的两个要求。协议如下:

- ① $A \rightarrow KDC: ID_A \parallel ID_B \parallel N_1$
- ② $KDC \rightarrow A: E_{K_A}[K_s \parallel ID_B \parallel N_1 \parallel E_{K_B}[K_s \parallel ID_A]]$
- ③ $A \rightarrow B: E_{K_B}[K_s \parallel ID_A] \parallel E_{K_s}[M]$

本协议不要求 B 同时在线,但保证了只有 B 能解读消息,同时还提供了对消息的发送方 A 的认证。然而本协议不能防止重放攻击,为此需在消息中加上时戳,但由于电子邮件处理中的延迟,时戳的作用极为有限。

2. 公钥加密

公钥加密算法可对发送的消息提供保密性、认证性或既提供保密性又提供认证性,为此要求发送方知道接收方的公开钥(保密性),或要求接收方知道发送方的公开钥(认证性),或要求每一方都知道另一方的公开钥。

如果主要关心保密性,则可使用以下方式:

$$A \rightarrow B: E_{PK_B}[K_s] \parallel E_{K_s}[M]$$

其中 A 用 B 的公开钥加密一次性会话密钥,用一次性会话密钥加密消息。只有 B 能够使用相应的秘密钥得到一次性会话密钥,再用一次性会话密钥得到消息。这种方案比简单地用 B 的公开钥加密整个消息要有效得多。

如果主要关心认证性,则可使用以下方式:

$$A \rightarrow B: M \parallel E_{SK_A}[H(M)]$$

这种方式可实现对 A 的认证,但不提供对 M 的保密性。如果既要提供保密性又要提供认证性,可使用以下方式:

$$A \rightarrow B: E_{PK_B}[M \parallel E_{SK_A}[H(M)]]$$

后两种情况要求 B 知道 A 的公开钥并确信公开钥的真实性。为此 A 还需同时向 B 发送自己的公钥证书,表示为

$$A \rightarrow B: M \parallel E_{SK_A}[H(M)] \parallel E_{SK_{AS}}[T \parallel ID_A \parallel PK_A]$$

或

$$A \rightarrow B: E_{PK_B}[M \parallel E_{SK_A}[H(M)] \parallel E_{SK_{AS}}[T \parallel ID_A \parallel PK_A]]$$

其中 $E_{SK_{AS}}[T \parallel ID_A \parallel PK_A]$ 是认证服务器 AS 为 A 签署的公钥证书。

7.5 身份证明技术

在很多情况下,用户都需证明自己的身份,如登录计算机系统、存取电子银行中的账户数据库、从自动出纳机 ATM(automatic teller machine)取款等。传统的方法是使用通行字或个人身份识别号 PIN(personal identification number)来证明自己的身份,这些方法的缺点是检验用户通行字或 PIN 的人或系统可使用用户的通行字或 PIN 冒充用户。

本节介绍的身份的零知识证明技术,可使用户在不泄露自己的通行字或 PIN 的情况下向他人证实自己的身份。

7.5.1 交互证明系统

交互证明系统由两方参与,分别称为证明者(prover,简记为 P)和验证者(verifier,简记为 V),其中 P 知道某一秘密(如公钥密码体制的秘密钥或一平方剩余 x 的平方根),P 希望使 V 相信自己的确掌握这一秘密。交互证明由若干轮组成,在每一轮,P 和 V 可能需根据从对方收到的消息和自己计算的某个结果向对方发送消息。比较典型的方式是在每轮 V 都向 P 发出一询问,P 向 V 做出一应答。所有轮执行完后,V 根据 P 是否在每一轮对自己发出的询问都能正确应答,以决定是否接受 P 的证明。

交互证明和数学证明的区别是:数学证明的证明者可自己独立地完成证明,而交互证明是由 P 产生证明、V 验证证明的有效性来实现,因此双方之间通过某种信道的通信是必需的。

交互证明系统须满足以下要求:

- ① 完备性 如果 P 知道某一秘密,V 将接受 P 的证明。
- ② 正确性 如果 P 能以一定的概率使 V 相信 P 的证明,则 P 知道相应的秘密。

7.5.2 简化的 Fiat-Shamir 身份识别方案

1. 协议及原理

设 $n=pq$,其中 p 和 q 是两个不同的大素数, x 是模 n 的平方剩余, y 是 x 的平方根。又设 n 和 x 是公开的,而 p, q 和 y 是保密的。证明者 P 以 y 作为自己的秘密。4.1.8 节已证明,求解方程 $x^2 \equiv a \pmod{n}$ 与分解 n 是等价的。因此他人不知 n 的两个素因子 p, q 而计算 y 是困难的。P 和验证者 V 通过交互证明协议,P 向 V 证明自己掌握秘密 y ,从而证明了自己的身份。

协议如下:

- ① P 随机选 $r(0 < r < n)$, 计算 $a \equiv r^2 \pmod n$, 将 a 发送给 V。
- ② V 随机选 $e \in \{0, 1\}$, 将 e 发送给 P。
- ③ P 计算 $b \equiv ry^e \pmod n$, 即 $e=0$ 时, $b=r$; $e=1$ 时, $b=ry \pmod n$ 。将 b 发送给 V。
- ④ 若 $b^2 \equiv ax^e \pmod n$, V 接受 P 的证明。

在协议的前 3 步, P 和 V 之间共交换了 3 个消息, 这 3 个消息的作用分别是: 第 1 个消息是 P 用来声称自己知道 a 的平方根; 第 2 个消息 e 是 V 的询问, 如果 $e=0$, P 必须展示 a 的平方根, 即 r , 如果 $e=1$, P 必须展示被加密的秘密, 即 $ry \pmod n$; 第 3 个消息 b 是 P 对 V 询问的应答。

2. 协议的完备性、正确性和安全性

(1) 完备性

如果 P 和 V 遵守协议, 且 P 知道 y , 则应答 $b \equiv ry^e \pmod n$ 应是模 n 下 ax^e 的平方根, 在协议的第④步 V 接受 P 的证明, 所以协议是完备的。

(2) 正确性

假冒的证明者 E 可按以下方式以 $\frac{1}{2}$ 的概率骗得 V 接受自己的证明:

- ① E 随机选 $r(0 < r < n)$ 和 $\tilde{e} \in \{0, 1\}$, 计算 $a \equiv r^2 x^{-\tilde{e}} \pmod n$, 将 a 发送给 V。
- ② V 随机选 $e \in \{0, 1\}$, 将 e 发送给 E。
- ③ E 将 r 发送给 V。

根据协议的第④步, V 的验证方程是 $r^2 \equiv ax^e \pmod n \equiv r^2 x^{-\tilde{e}} x^e \pmod n$, 当 $\tilde{e}=e$ 时, 验证方程成立, V 接受 E 的证明, 即 E 欺骗成功。因 $\tilde{e}=e$ 的概率是 $\frac{1}{2}$, 所以 E 欺骗成功的概率是 $\frac{1}{2}$ 。另一方面, $\frac{1}{2}$ 是 E 能成功欺骗的最好概率, 否则假设 E 以大于 $\frac{1}{2}$ 的概率使 V 相信自己的证明, 那么 E 知道一个 a , 对这个 a 他可正确地应答 V 的两个询问 $e=0$ 和 $e=1$, 意味着 E 能计算 $b_1^2 \equiv a \pmod n$ 和 $b_2^2 \equiv ax \pmod n$, 即 $\frac{b_2^2}{b_1^2} \equiv x \pmod n$, 因此 E 由 $\frac{b_2}{b_1} \pmod n$ 即可求得 x 的平方根 y , 矛盾。

(3) 安全性

协议的安全性可分别从证明者 P 和验证者 V 的角度来考虑。根据上面的讨论, 假冒的证明者 E 欺骗 V 成功的概率是 $\frac{1}{2}$, 对 V 来说, 这个概率太大了。为减小这个概率, 可将协议重复执行多次, 设执行 t 次, 则欺骗者欺骗成功的概率将减小到 2^{-t} 。

下面考虑 P 的安全性。因为 V 的询问是在很小的集合 $\{0, 1\}$ 中选取的, V 没有机会产生其他信息, 而 P 发送给 V 的信息仅为 P 知道 x 的平方根这一事实, 因此 V 无法得知

x 的平方根。

7.5.3 零知识证明

零知识证明起源于最小泄露证明。在交互证明系统中,设 P 知道某一秘密,并向 V 证明自己掌握这一秘密,但又不向 V 泄露这一秘密,这就是最小泄露证明。进一步,如果 V 除了知道 P 能证明某一事实外,不能得到其他任何信息,则称 P 实现了零知识证明,相应的协议称为零知识证明协议。

【例 7-4】 图 7-4 表示一个简单的迷宫,C 与 D 之间有一道门,需要知道秘密口令才能将其打开。 P 向 V 证明自己能打开这道门,但又不愿向 V 泄露秘密口令。可采用如下协议:

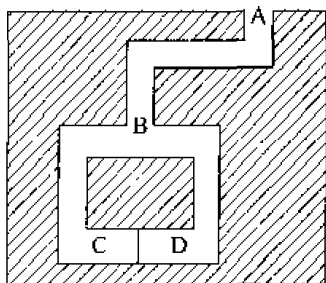


图 7-4 零知识证明协议示例

- ① V 在协议开始时停留在位置 A 。
- ② P 一直走到迷宫深处,随机选择位置 C 或位置 D 。
- ③ P 消失后, V 走到位置 B ,然后命令 P 从某个出口返回位置 B 。
- ④ P 服从 V 的命令,必要时利用秘密口令打开 C 与 D 之间的门。
- ⑤ P 和 V 重复以上过程 n 次。

协议中,如果 P 不知道秘密口令,就只能从来路返回 B ,而不能走另外一条路。此外, P 每次猜对 V 要求走哪一条路的概率是 $\frac{1}{2}$,因此每一轮中 P 能够欺骗 V 的概率是 $\frac{1}{2}$ 。

假定 n 取 16,则执行 16 轮后, P 成功欺骗 V 的概率是 $\frac{1}{2^{16}} = \frac{1}{65536}$ 。于是,如果 P 16 次都能按 V 的要求返回, V 即能证明 P 确实知道秘密口令。还可以看出, V 无法从上述证明过程中获取丝毫关于 P 的秘密口令的信息,所以这是一个零知识证明协议。

【例 7-5】 哈密尔顿(Hamilton)回路

图中的回路是指始点和终点相重合的路径,若回路通过图的每个顶点一次且仅一次,则称为哈密尔顿回路,构造图的哈密尔顿回路是 NPC 问题。现在假定 P 知道图 G 的哈密尔顿回路,并希望向 V 证明这一事实,可采用如下协议:

- ① P 随机地构造一个与图 G 同构的图 \tilde{G} , 并将 \tilde{G} 交给 V。
- ② V 随机地要求 P 做下述两件工作之一:
 - 证明图 G 和图 \tilde{G} 同构;
 - 指出图 \tilde{G} 的一条哈密尔顿回路。
- ③ P 根据要求做下述两件工作之一:
 - 证明图 G 和图 \tilde{G} 同构, 但不指出图 G 或图 \tilde{G} 的哈密尔顿回路;
 - 指出图 \tilde{G} 的哈密尔顿回路, 但不证明图 G 和图 \tilde{G} 同构。
- ④ P 和 V 重复以上过程 n 次。

协议执行完后, V 无法获得任何信息使自己可以构造图 G 的哈密尔顿回路, 因此该协议是零知识证明协议。事实上, 如果 P 向 V 证明图 G 和图 \tilde{G} 同构, 这个结论对 V 并没有意义, 因为构造图 \tilde{G} 的哈密尔顿回路和构造图 G 的哈密尔顿回路同样困难。如果 P 向 V 指出图 \tilde{G} 的一条哈密尔顿回路, 这一事实也无法向 V 提供任何帮助, 因为求两个图之间的同构并不比求一个图的哈密尔顿回路容易。在协议的每一轮中, P 都随机地构造一个与图 G 同构的新图, 因此不论协议执行多少轮, V 都得不到任何有关构造图 G 的哈密尔顿回路的信息。

注: 两个图 G_1 和 G_2 是同构的是指从 G_1 的顶点集到 G_2 的顶点集之间存在一个一一映射 π , 当且仅当若 x, y 是 G_1 上的相邻点, $\pi(x)$ 和 $\pi(y)$ 是 G_2 上的相邻点。

7.5.4 Fiat-Shamir 身份识别方案

1. 协议及原理

在简化的 Fiat-Shamir 身份识别方案中, 验证者 V 接受假冒的证明者证明的概率是 $\frac{1}{2}$, 为减小这个概率, 将证明者的秘密改为由随机选择的 t 个平方根构成的一个向量 $y = (y_1, y_2, \dots, y_t)$, 模数 n 和向量 $x = (y_1^2, y_2^2, \dots, y_t^2)$ 是公开的, 其中 n 仍是两个不相同的大素数的乘积。

协议如下:

- ① P 随机选 $r (0 < r < n)$, 计算 $a \equiv r^2 \pmod{n}$, 将 a 发送给 V。
- ② V 随机选 $e = (e_1, e_2, \dots, e_t)$, 其中 $e_i \in \{0, 1\} (i = 1, 2, \dots, t)$, 将 e 发送给 P。
- ③ P 计算 $b \equiv r \prod_{i=1}^t y_i^{e_i} \pmod{n}$, 将 b 发送给 V。
- ④ 若 $b^2 \not\equiv a \prod_{i=1}^t x_i^{e_i} \pmod{n}$, V 拒绝 P 的证明, 协议停止。
- ⑤ P 和 V 重复以上过程 k 次。

2. 协议的完备性、正确性和安全性

(1) 完备性

若 P 和 V 遵守协议, 则 V 接受 P 的证明。

(2) 正确性

如果假冒者 E 欺骗 V 成功的概率大于 2^{-kt} , 意味着 E 知道一个向量 $A = (a^1, a^2, \dots, a^k)$, 其中 a^j 是第 j 次执行协议时产生的, 对这个 A , E 能正确地回答 V 的两个不同的询问 $E = (e^1, e^2, \dots, e^k)$ 、 $F = (f^1, f^2, \dots, f^k)$ (每一元素是一向量), $E \neq F$ 。由 $E \neq F$ 可设 $e^j \neq f^j$, e^j 和 f^j 是第 j 次执行协议时 V 的两个不同的询问 (为向量), 简记为 $e = e^j$ 和 $f = f^j$, 这一轮对应的 a^j 简记为 a 。所以 E 能计算两个不同的值 $b_1^2 \equiv a \prod_{i=1}^t x_i^{e_i} \pmod{n}$, $b_2^2 \equiv a \prod_{i=1}^t x_i^{f_i} \pmod{n}$, 即 $\frac{b_2^2}{b_1^2} \equiv \prod_{i=1}^t x_i^{f_i - e_i} \pmod{n}$, 所以 E 可由 $\frac{b_2}{b_1} \pmod{n}$ 求得 $x \equiv \prod_{i=1}^t x_i^{f_i - e_i} \pmod{n}$ 的平方根, 矛盾。

(3) 安全性

Fiat-Shamir 身份识别方案是对简化的 Fiat-Shamir 身份识别方案的推广, 首先将 V 的询问由一个比特推广到由 t 个比特构成的向量, 再者基本协议被执行 k 次。假冒的证明者只有能正确猜测 V 的每次询问, 才可使 V 相信自己的证明, 成功的概率是 2^{-kt} 。

7.6 其他密码协议

7.6.1 智力扑克

假设两个人 A 和 B 通过计算机网络进行智力扑克比赛, 比赛中不用第三方做裁判。发牌者可由任一方担任, 发牌过程应满足以下要求:

- ① 任一副牌 (即发给参赛人员手中的牌) 是等可能的。
- ② 发给 A、B 手中的牌没有重复的。
- ③ 每人都知道自己手中的牌, 但却不知对方手中的牌。
- ④ 比赛结束后, 每一方都能发现对方的欺骗行为 (如果有的话)。

为满足这些要求, A、B 之间必须以加密形式交换一些信息。在下面的协议中, 加密体制可以是单钥密码也可以是公钥密码, 设 E_A 和 E_B 、 D_A 和 D_B 分别表示 A 和 B 的加密变换和解密变换, 在比赛结束之前, 这些变换都是保密的, 比赛结束后予以公布用以证明比赛的公正性。要求加密变换满足交换律, 即对任意消息 M 有:

$$E_A(E_B(M)) = E_B(E_A(M))$$

比赛开始前 A、B 协商好用以表示 52 张牌的消息 w_1, w_2, \dots, w_{52} 。协议中设 A 为发牌人, 并设给每人发 5 张牌。协议如下:

① B 先洗牌, 然后用 E_B 对 52 个消息分别加密, 将加密结果 $E_B(w_i)$ 发送给 A。

② A 从收到的 52 个加密的消息中随机选 5 个 $E_B(w_i)$, 并发送给 B, B 用自己的解密变换 D_B 对这 5 个值解密, 解密后的值作为发给自己的一副牌。因为 B 的加密变换 E_B 和解密变换 D_B 都是保密的, 所以 A 无法知道 B 手中的牌。

③ A 另选 5 个 $E_B(w_i)$, 用 E_A 加密后发送给 B。

④ B 用 D_B 对收到的值解密后再发送给 A, A 用 D_A 对收到的值解密后作为发给自己的一副牌, 这是因为 B 发送给 A 的值是

$$D_B(E_A(E_B(w_i))) = D_B(E_B(E_A(w_i))) = E_A(w_i)$$

其中用到加密变换的交换律。

下面考虑该协议是否满足发牌过程的 4 个要求。

对第②个要求, B 可在协议的第③步检查 A 发来的 5 个值是否和第②步发来的 5 个值有重复。为满足第 4 个要求, 可在比赛结束后公开所有的加密变换和解密变换, 双方都可检查对方的牌看是否有欺诈。对第①个和第③个要求来说, 关键在于加密变换 E_B 的强度, 由 $E_B(w_i)$ 可能得不出 w_i , 但有可能得出 w_i 的部分信息。例如, w_i 是一个比特串, 则有可能从 $E_B(w_i)$ 得出 w_i 的最后一个比特, 因此 A 可将 52 个值 $E_B(w_1), E_B(w_2), \dots, E_B(w_{52})$ 分成两个子集, A 在发牌时可将发给 B 的牌集中在某一子集中, 因此使得第①个和第③个要求无法满足。

7.6.2 掷硬币协议

在某些密码协议中要求通信双方在无第三方协助的情况下, 产生一个随机序列, 因为 A、B 之间可能存在不信任关系, 因此随机序列不能由一方产生再通过电话或网络告诉另一方。这一问题可通过掷硬币协议来实现, 掷硬币协议有多种实现方式, 下面介绍其中的 3 种。

1. 采用平方根掷硬币

协议如下:

① A 选择两个大素数 p, q , 将乘积 $n = pq$ 发送给 B。

② B 在 1 和 $n/2$ 之间, 随机选择一个整数 u , 计算 $z \equiv u^2 \pmod{n}$, 并将 z 发送给 A。

③ A 计算模 n 下 z 的 4 个平方根 $\pm x$ 和 $\pm y$ (因 A 知道 n 的分解, 所以可做到), 设 x' 是 $x \pmod{n}$ 和 $-x \pmod{n}$ 中较小者, y' 是 $y \pmod{n}$ 和 $-y \pmod{n}$ 中较小者, 则由于

$1 < u < \frac{n}{2}$, 所以 u 为 x' 和 y' 之一。

④ A 猜测 $u = x'$ 或 $u = y'$, 或者 A 找出最小的 i 使得 x' 的第 i 个比特与 y' 的第 i 个比特不同, A 猜测 u 的第 i 个比特是 0 还是 1。A 将猜测发送给 B。

⑤ B 告诉 A 猜测正确或不正确, 并将 u 的值发送给 A。

⑥ A 公开 n 的因子。

因 u 是 B 随机选取的, A 不知道 u , 所以要猜测 u 只能是计算模 n 下 z 的 4 个平方根, 猜中的概率是 $\frac{1}{2}$ 。再考虑 B 如何能欺骗 A, 如果 B 在 A 猜测完后能够改变 u 的值, 则

A 的猜测必不正确, A 可通过 $z \equiv u^2 \pmod{n}$ 检查出 B 是否改变了 u 的值, 所以 B 要想改变 u 的值就只能在 x' 和 y' 之间进行。而 B 若掌握 x' 和 y' , 就可通过 $\gcd\{x' - y', n\}$ 或 $\gcd\{x' + y', n\}$ 求出 p 和 q , 说明 B 的欺骗与分解 n 是等价的。

【例 7-6】 本例是采用平方根掷硬币的一个具体实现过程:

① A 取 $p=3, q=7$, 将 $n=21$ 发送给 B。

② B 在 1 和 $\frac{21}{2}$ 之间, 随机选择一个整数 $u=2$, 计算 $z \equiv 2^2 \pmod{21} \equiv 4$ 并将 $z=4$ 发送给 A。

③ A 计算模 21 下 $z=4$ 的 4 个平方根 $x=2, -x=19, y=5, -y=16$, 取 $x'=2, y'=5$ 。

④ A 猜测 $u=5$ 并将猜测发送给 B。

⑤ B 告诉 A 猜测不正确, 并将 $u=2$ 发送给 A, A 检验 $u=2$ 在 1 和 $\frac{21}{2}$ 之间且满足 $4 \equiv 2^2 \pmod{21}$, A 知道自己输了。

⑥ A 公开 $n=21$ 的因子 $p=3, q=7$, B 检验 $n=pq$, 知道自己赢了。

2. 利用单向函数掷硬币

设 A、B 都知道某一单向函数 $f(x)$, 但都不知道该函数的逆函数, 协议如下:

① B 选择一个随机数 x , 求 $y=f(x)$ 并发送给 A。

② A 对 x 的奇偶性进行猜测, 并将结果告诉 B。

③ B 告诉 A 猜测正确或不正确, 并将 x 发送给 A。

由于 A 不知道 $f(x)$ 的逆函数, 因此无法通过 B 发过来的 y 得出 x , 即只能猜测 x 的奇偶性。而 B 若在 A 做出猜测以后改变 x , A 可通过 $y \stackrel{?}{=} f(x)$ 检查出来。

3. 利用二次剩余掷硬币

设 n 是两个大素数 p, q 的乘积, 即 $n=pq$ 。整数 a 满足 $0 < a < n$ 和 $\gcd(a, n)=1$, 则

有一半的 a , 其 Jacobi 符号 $\left(\frac{a}{n}\right) = -1$, 而在满足 $\left(\frac{a}{n}\right) = 1$ 的所有 a 中, 只有一半是模 n 的平方剩余, 而判断 a 是否为模 n 的平方剩余与分解 n 是等价的。

协议如下:

- ① B 选择 p, q , 计算 $n = pq$; 再选取满足 $\left(\frac{a}{n}\right) = 1$ 的随机数 a , 将 n 和 a 发送给 A。
- ② A 猜测 a 是模 n 的平方剩余或非平方剩余, 并将结果告诉 B。
- ③ B 告诉 A 猜测正确或不正确, 并将 p, q 发送给 A。
- ④ A 检查 p, q 都是素数且 $n = pq$ 。

显然, A 猜中的概率是 $\frac{1}{2}$ 。协议执行完后, A 根据 p, q 可求出 $a \bmod n$ 的 4 个平方根 (如果 a 是模 n 的平方剩余), 以检查 B 是否在 A 猜测完后将结果做了修改。

7.6.3 不经意传输

设 A 有一个秘密, 想以 $\frac{1}{2}$ 的概率传递给 B, 即 B 有 50% 的机会收到这个秘密, 另外 50% 的机会什么也没有收到, 协议执行完后, B 知道自己是否收到了这个秘密, 但 A 却不知 B 是否收到了这个秘密。这种协议就称为不经意传输协议。

例如 A 是机密的出售者, A 列举了很多问题, 意欲出售各个问题的答案, B 想买其中一个问题的答案, 但又不想让 A 知道自己买的是哪个问题的答案。

1. 基于大数分解问题的不经意传输协议

设 A 想通过不经意传输协议传递给 B 的秘密是整数 n (为两个大素数之积) 的因数分解。这个问题具有普遍意义, 因为任何秘密都可通过 RSA 加密, 得到 n 的因数分解就可得到这个秘密。

协议基于如下事实: 已知某数在模 n 下两个不同的平方根, 就可分解 n 。

协议如下:

- ① B 随机选一数 x , 将 $x^2 \bmod n$ 发送给 A。
- ② A (掌握 $n = pq$ 的分解) 计算 $x^2 \bmod n$ 的 4 个平方根 $\pm x$ 和 $\pm y$, 并将其中之一发送给 B。由于 A 只知道 $x^2 \bmod n$, 并不知道 4 个平方根中哪一个是 B 选的 x 。
- ③ B 检查第②步收到的数是否与 $\pm x$ 在模 n 下同余, 如果是, 则 B 没有得到任何新信息; 否则 B 就掌握了 $x^2 \bmod n$ 的两个不同的平方根, 从而能够分解 n 。而 A 却不知究竟是哪种情况。

显然, B 得到 n 的分解的概率是 $\frac{1}{2}$ 。

2. 基于离散对数问题的不经意传输协议

下一个不经意传输协议是非交互的,其中B不向A发送任何消息。设系统中所有用户都知道一个大素数 p 、 $GF(p)-\{0\}$ 的生成元 g 和另一大素数 c ,但无人知道 c 的离散对数。假定计算离散对数是不可行的,因此从 $g^x \bmod p$ 和 $g^y \bmod p$ 无法计算 $g^{xy} \bmod p$ 。协议中所有运算都在 $GF(p)$ 中进行。

B按如下方式产生公开的加密密钥和秘密的解密密钥:随机选取一个比特 i 和一个数 $x(0 \leq x \leq p-2)$,计算 $y_i = g^x$, $y_{1-i} = c(g^x)^{-1}$,以 (y_0, y_1) 作为公开的加密密钥,以 (i, x) 作为秘密解密密钥。由于B不知道 c 的离散对数,所以他知道 y_0 或 y_1 的离散对数,而A无法知道 y_0 和 y_1 中哪个离散对数是B已知的。A可通过方程 $y_0 y_1 = c$ 来检查B的公开加密密钥是否正确。

协议中设A的两个秘密 s_0 和 s_1 是二进制数, \oplus 是异或运算,若进行异或运算的两个数不等长,可在较短数前面补0。

协议如下:

① A在0到 $p-2$ 之间随机取两个整数 k_0, k_1 ,对 $j=0, 1$ 计算 $c_j = g^{k_j}$, $d_j = y_j^{k_j}$, $m_j = s_j \oplus d_j$,将 c_0, c_1, m_0, m_1 发送给B。

② B用自己的秘密解密密钥计算 $c_i^x = g^{x k_i} = y_i^{k_i} = d_i$, $s_i = m_i \oplus d_i$ 。由于B不知道 y_{1-i} 的离散对数,所以无法得到 d_{1-i} 和 s_{1-i} 。

3. “多传一”的不经意传输协议

设A有多个秘密,想将其中一个传递给B,使得只有B知道A传递的是哪个秘密。设A的秘密是 s_1, s_2, \dots, s_k ,每一秘密是一比特序列。协议如下:

① A告诉B一个单向函数 f ,但对 f^{-1} 保密。

② 设B想得到秘密 s_i ,他在 f 的定义域内随机选取 k 个值 x_1, x_2, \dots, x_k ,将 k 元组 (y_1, y_2, \dots, y_k) 发送给A,其中

$$y_j = \begin{cases} x_j & j \neq i \\ f(x_j) & j = i \end{cases}$$

③ A计算 $z_j = f^{-1}(y_j)$ ($j=1, 2, \dots, k$),并将 $z_j \oplus s_j$ ($j=1, 2, \dots, k$)发送给B。

④ 由于 $z_i = f^{-1}(y_i) = f^{-1}(f(x_i)) = x_i$,所以B知道 z_i ,因此可从 $z_i \oplus s_i$ 获得 s_i 。

由于B没有 z_j ($j \neq i$)的信息,因此无法得到 s_j ($j \neq i$),而A不知 k 元组 (y_1, y_2, \dots, y_k) 中哪个是 $f(x_i)$,因此无法确定B得到的是哪个秘密。

然而如果B不遵守协议,他用 f 对多个 x_j 求得 $f(x_j)$,就可获得多个秘密。B的欺骗可分为被动欺骗和主动欺骗。被动欺骗是指B遵守协议,但却意欲获得比诚实用户更多的信息;主动欺骗是指B根本就不遵守协议。显然这种“多传一”协议中若存在主动欺

骗,协议的安全性就很差,因此总假定这种“多传一”协议中所有用户都遵守协议,协议的安全性主要考虑防止被动欺骗。

4. 基于大数分解问题的“多传一”不经意传输协议

设 A 有多个秘密,并对自己的每个秘密都使用一个不同的 RSA 体制加密,A 要想向 B 传递其中的一个秘密,就可告诉 B 加密该密钥的 RSA 体制的模数。协议如下:

① A 构造 k 个 RSA 加密体制,使得在每个体制中的两个素数 p_j 和 q_j 满足 $p_j \equiv q_j \equiv 3 \pmod{4}$ (因此可保证同一数 a 在模 $n_j = p_j q_j$ 下的两个平方根有相反的 Jacobi 符号),将加密密钥 (e_j, n_j) 及加密后的秘密 $s_j^{e_j} \pmod{n_j}$ 发送给 B,其中 $j=1,2,\dots,k$ 。

② B 选 k 个数 x_1, x_2, \dots, x_k , 分别计算 Jacobi 符号 $\left(\frac{x_i}{n_j}\right)$ 和 $x_i^2 \pmod{n_j}$ ($j=1,2,\dots,k$)。

B 如果想获得秘密 s_i , 则将 $x_i^2 \pmod{n_i}$ 和 $-\left(\frac{x_i}{n_i}\right)$ 发送给 A, 而对所有 $j \neq i$, 将 $x_j^2 \pmod{n_j}$ 和 $\left(\frac{x_j}{n_j}\right)$ 发送给 A。

③ 对每一 j , A 计算 $x_j^2 \pmod{n_j}$ 的平方根和平方根的 Jacobi 符号, 比较每一平方根的 Jacobi 符号是否与第②步收到的 Jacobi 符号相同, 将 Jacobi 符号相同的那一平方根发送给 B。

④ B 现在获得 $x_i^2 \pmod{n_i}$ 的两个不同的平方根, 因此能够分解 n_i , 求出解密密钥 d_i , 进一步求出 s_i ; 而对 $j \neq i$, B 在第③步收到的平方根是自己已知的, 因此无法求出 n_j 和 s_j 。

因为 A 不知道 B 选择的是哪个 i , 因此不知道 B 获得的是哪个秘密。协议中仍假定 A、B 都遵守协议, 否则 B 在第②步进行主动欺骗的话, A 仍无法识别。

【例 7-7】 在上述协议中, 设 A 用于加密某个秘密 s 的 RSA 体制的模数 $n=2773=47 \times 59$, 满足 $47 \equiv 59 \equiv 3 \pmod{4}$ 。B 在第②步选择的相应 $x=2001$, 计算 $x^2 \pmod{n}=2001^2 \pmod{2773}=2562$ 及

$$\begin{aligned} \left(\frac{2001}{2773}\right) &= \left(\frac{2773}{2001}\right) = \left(\frac{772}{2001}\right) = \left(\frac{193}{2001}\right) = \left(\frac{2001}{193}\right) = \left(\frac{71}{193}\right) \\ &= \left(\frac{193}{71}\right) = \left(\frac{51}{71}\right) = -\left(\frac{71}{51}\right) = -\left(\frac{20}{51}\right) = -\left(\frac{5}{51}\right) \\ &= -\left(\frac{51}{5}\right) = \left(\frac{1}{5}\right) = -1 \end{aligned}$$

如果 B 想获得 s , 则将 $(2562, 1)$ 发送给 A。

第③步, A 如下计算 $2562 \pmod{2773}$ 的平方根:

求 $2562 \pmod{47}=24$, $2562 \pmod{59}=25$, 求出 24 在 $\pmod{47}$ 时的平方根为 ± 27 , 25 在 $\pmod{59}$ 时的平方根为 ± 5 (求模下的平方根存在有多项式时间的算法, 可参考相关文献),

用推广的 Euclid 算法求出 $59^{-1} \bmod 47 \equiv 4, 47^{-1} \bmod 59 \equiv 54$, 由中国剩余定理求出平方根为 $\pm 27 \cdot 59 \cdot 4 \pm 5 \cdot 47 \cdot 54$, 即 349, 772, 2001, 2424。

因 $\left(\frac{349}{2773}\right) = \left(\frac{2424}{2773}\right) = 1, \left(\frac{772}{2773}\right) = \left(\frac{2001}{2773}\right) = -1$, A 将 349 或 2424 发送给 B。

第④步, B 由 $\gcd\{2773, 349 + 2001\} = 47$ 或 $\gcd\{2773, 2424 - 2001\} = 47$ 得 n 的一个因子, 从而得到 n 的分解式 47×59 。若 B 不想获得 s , 则将 $(2562, -1)$ 发送给 A, A 将 772 或 2001 发送给 B, 因 $772 \equiv 2001 \bmod 2773$, 所以 B 未收到任何新信息。

习 题

1. 在 DSS 数字签名标准中, 取 $p = 83 = 2 \times 41 + 1, q = 41, h = 2$, 于是 $g \equiv 2^2 \equiv 4 \bmod 83$, 若取 $x = 57$, 则 $y \equiv g^x \equiv 4^{57} \equiv 77 \bmod 83$ 。在对消息 $M = 56$ 签名时, 选择 $k = 23$, 计算签名并进行验证。

2. 在 DSA 签字算法中, 参数 k 泄露会产生什么后果?

3. 假设你知道一个背包问题的解, 试设计一个协议, 以零知识证明方式证明你的确知道问题的解。

4. 在 7.6.3 节, 基于大数分解问题的“多传一”不经意传输协议中为什么要求 $p_i \equiv q_i \equiv 3 \bmod 4$? 设 $n_i = 55 = 5 \times 11$, B 选择 $x_i = 2$, 且想获得 A 的秘密 s_i , 分析 B 是否能成功获得。

第 8 章

网络加密与认证

8.1 网络通信加密

8.1.1 开放系统互连和 TCP/IP 分层模型

1. 开放系统互连参考模型

开放系统互连 OSI(open systems interconnection)参考模型描述信息如何从一台计算机的应用层软件通过网络媒体传输到另一台计算机的应用层软件,它是由 7 层协议组成的概念模型,每一层都说明了特定的网络功能。

OSI 参考模型把网络中计算机之间的信息传递分成 7 个较小的易于管理的层,它的 7 层协议中的每一层协议分别执行一个(或一组)任务,各层间相互独立,互不影响。7 层由低至高分别为物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。如图 8-1 所示,其中左边数字表示层次,右边表示可将 7 层继续分为高层和低层两类,其中高层论述的是应用问题,通常用软件实现。最高层(应用层)最接近用户,用户和应用层通过通信应用软件相互作用。在参考模型中,上层意指某一层之上的任何层。

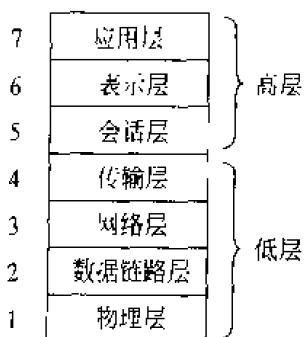


图 8-1 OSI 参考模型的层次划分

低层负责处理数据传输问题,物理层和数据链路层由硬件和软件共同实现,而其他层

通常只是用软件来实现。最底层(物理层)最接近物理网络介质(如网络电缆),其职责是将信息放置到介质上。下面给出各层的具体含义。

物理层:物理层定义了用于执行、维护、终止物理链路所需要的电子、机械、过程及功能的规则。

数据链路层:数据链路层通过物理网络链路提供可靠的数据传输。不同的数据链路层定义了不同的网络和协议特性,其中包括物理编址、网络拓扑结构、错误校验、帧序列以及流控。

网络层:用于提供路由选择及其相关的功能,网络层为高层协议提供面向连接的服务和无连接服务。网络层协议一般都是路由选择协议,但其他类型的协议也可在网络层上实现。

传输层:用于实现向高层可靠地传输数据的服务。传输层的功能一般包括流量控制、多路传输、虚电路管理及差错校验和恢复。

会话层:用于建立、管理和终止表示层与实体之间的通信会话,通信会话包括发生在不同网络设备的应用层之间的服务请求和服务应答,这些请求和应答通过会话层的协议实现。

表示层:提供多种用于应用层数据的编码和转化功能,以确保从一个系统应用层发送的信息可以被另一系统的应用层识别。

应用层:是最接近终端用户的 OSI 层,这就意味着 OSI 应用层与用户之间是通过软件直接相互作用的。应用层的功能一般包括标识通信伙伴、定义资源的可用性和同步通信。

OSI 模型系统间的通信方式如下:信息从一个计算机系统的应用层软件传输到另一个计算机系统的应用层软件,必须经过 OSI 参考模型的每一层。例如,系统 A 的应用层软件要将信息传送到系统 B 的应用层软件,那么系统 A 的应用程序先把该信息传送到 A 的应用层(第 7 层),然后应用层又把信息传送到表示层(第 6 层),表示层再把信息传送到会话层(第 5 层),依次下去,直到信息传送到物理层(第 1 层)。在物理层,信息被放置到物理网络介质上,并通过介质发送到系统 B。系统 B 的物理层从物理介质上获取信息,然后把信息从物理层传送到数据链路层(第 2 层),数据链路层再把信息传送到网络层(第 3 层),依次上去,直到信息传送到系统 B 的应用层(第 7 层)。最后 B 的应用层再把信息传送到接收应用程序中,这样便完成了整个通信过程。

2. TCP/IP 分层模型

TCP/IP 是因特网(Internet)的基本协议,它是“传输控制协议 TCP(transmission control protocol)和互联网协议 IP(Internet protocol)”的简称。事实上,TCP/IP 是个协

议系统,是由一系列支持网络通信的协议组成的集合。本节仅介绍 TCP/IP 的分层模型,对具体的协议不做介绍。

TCP/IP 可以采用与 OSI 结构相同的分层方法来建立模型,其模型分为 4 层,分别称为应用层、传输层、IP 层和接口层。

- ① 应用层:这一层将 OSI 高层(应用层、表示层和会话层)的功能合并为一层。
- ② 传输层:在功能上,这一层等价于 OSI 的传输层。
- ③ IP 层:在功能上,这一层等价于 OSI 的网络层。
- ④ 接口层:在功能上,这一层等价于 OSI 的数据链路层和物理层。

其中在传输层上的协议有两个:传输控制协议 TCP 和用户数据报协议 UDP(user datagram protocol)。TCP 协议是一个面向连接的传输协议,是为在无连接的网络业务上运行面向连接的业务而设计的;UDP 协议是一个无连接传输协议,它与 OSI 的无连接传输协议相对应。

8.1.2 网络加密方式

1. 基本方式

为了将数据在网络中传送,需要在数据前面加上它的目的地址,称加在数据前面的目的地址为报头,用户数据加上报头称为数据报。加强网络通信安全性的最有效且最常用的方法是加密,网络加密的基本方式有两种:链路加密和端端加密。

链路加密是指每个易受攻击的链路两端都使用加密设备进行加密,因此整个通信链路上传输都是安全的。缺点是数据报每进入一个分组交换机后都需要一次解密,原因是交换机必须读取数据报报头以便为数据报选择路由。因此在交换机中数据报易受到攻击。

链路加密时,每一链路两端的一对结点都应共享一个密钥,不同结点对共享不同的密钥。因此需提供很多密钥,每个密钥仅分配给一对结点。

端端加密是指仅在一对用户的通信线路两端(即源结点和终端结点)进行加密,因此数据是以加密的形式通过网络由源结点传送到目标结点,目标结点用与源结点共享的密钥对数据解密。所以端端加密可防止对网络上链路和交换机的攻击。

端端加密还能提供一定程度的认证,因为源结点和终端结点共享同一密钥,所以终端结点相信自己收到的数据报的确是由源结点发来的。链路加密方式不具备这种认证功能。

端端加密也有自己的缺点,由于只有目标结点能对加密结果解密,所以如果对整个数据报加密,则分组交换结点收到加密结果后无法读取报头,从而无法为该数据报选择路由。所以主机只能对数据报中的用户数据部分加密而报头则以明文形式传送,这样虽然用户数据部分是安全的,然而却容易受业务流量分析的攻击。

为提高安全性,可将两种加密方式结合起来使用,如图 8-2 所示。其中主机用端端加密密钥加密数据报中用户的数据部分,然后用链路加密密钥对整个数据报再加密一次。当被加密的数据报在网络中传送时,每一交换机都使用链路加密密钥解密数据报以读取报头,然后再用下一链路的链路加密密钥加密整个数据报并发往下一交换机。所以当两种加密方式结合起来使用时,除了在每个交换机内部数据报报头是明文形式外,其他整个过程数据报都是密文形式。

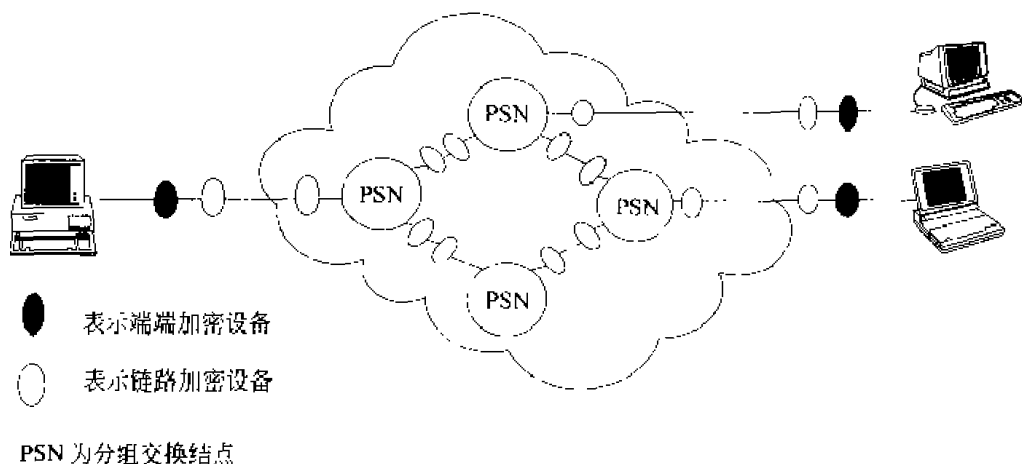


图 8-2 分组交换网中的加密

2. 端端加密的逻辑位置

端端加密的逻辑位置是指将加密功能放在 OSI 参考模型的哪一层,可有几种选择,其中最低层的加密可在网络层进行,这时,被保护的实体数目与网络中终端数目一样,任意两个终端如果共享同一密钥,就可进行保密通信。一终端系统若想和另一终端系统进行保密通信,则两个端系统用户的所有处理程序和应用程序都将使用同一加密方案和同一密钥。

在端端协议中利用加密功能,可为通信业务提供端端的安全性。然而这种方案不能为穿过互联网的通信业务(如电子邮件、电子数据交换 EDI、文件传输)提供这种端端的安全性。图 8-3 表示用电子邮件网关沟通两个互联网,其中一个是使用 OSI 结构,另一个是使用 TCP/IP 结构。这时在两个互联网之间的应用层以下不存在端端协议,从一个端系统发出的传输和连接到邮件网关后即终止,邮件网关再建立一个新的传输并连接到另一端系统。即使邮件网关连接的两个互联网使用同一结构,传输过程也是如此。因此,对诸如电子邮件这种具有存储转发功能的应用,只有在应用层才有端端加密功能。

应用层加密的缺点是需考虑的实体数目将显著地增加,比如网络中有数百个主机,则需考虑的实体(用户和进程)可能有数千个,不同的一对实体需产生一个不同的密钥,因此,需要产生和分布更多的密钥。改进的方法是在分层结构上,越往上层则加密的内容越

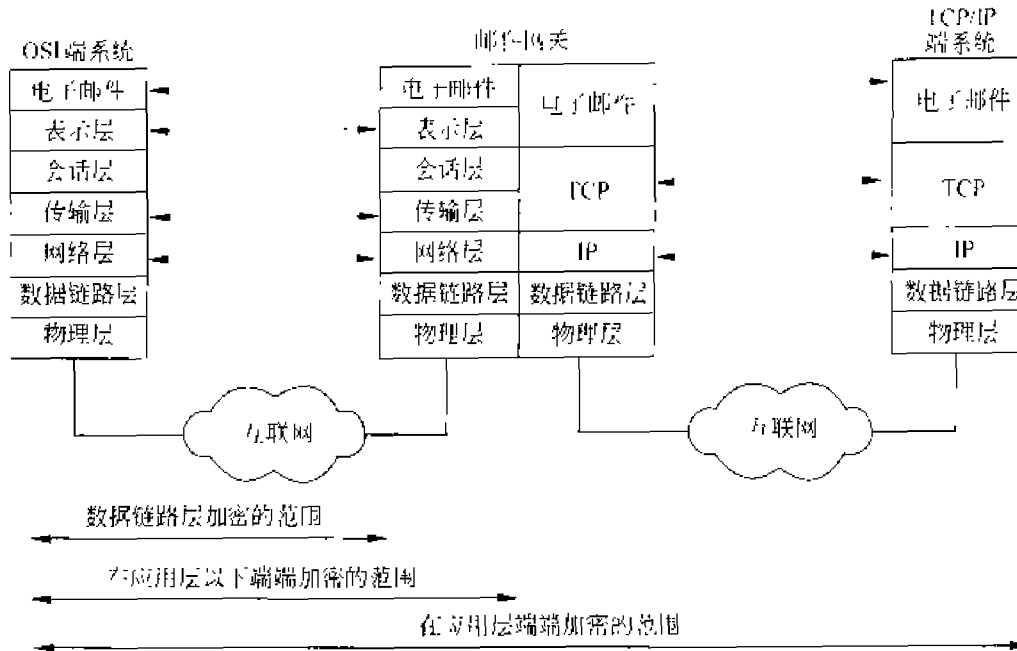
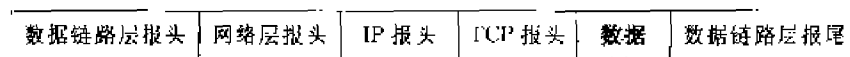
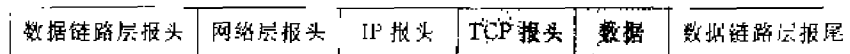


图 8-3 存储转发通信的加密范围

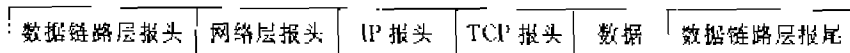
少。图 8-4 以 TCP/IP 结构为例说明这种改进方法,其中应用层网关指在应用层上操作的存储转发设备,阴影部分表示加密。图 8-4(a)表示在应用层加密,这时仅对 TCP 数据



(a) 应用层加密(链路上和路由器、网关中)

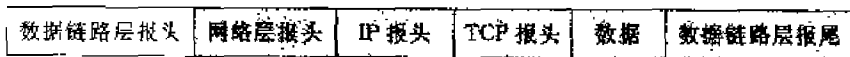


链路上和路由器中

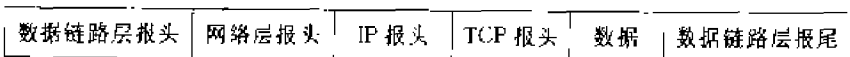


网关中

(b) TCP 层加密



链路上



路由器和网关中

(c) 数据链路层加密

图 8-4 不同层次的加密方案

段中的用户数据部分加密,而 TCP 报头、IP 报头、网络层报头、数据链路层报头以及数据链路层报尾则是明文形式。图 8-4(b)表示在 TCP 层加密,这时在链路上和路由器中,用户数据和 TCP 报头被加密,而 IP 报头则是明文形式,这是因为路由器需要为 IP 数据报选择从源结点到目标结点的路由。然而如果数据报通过网关,则终止 TCP 连接,并为下一跳建立一个新的传输连接,这时 IP 也将网关当作目标结点。因此,在网关,数据单元又被解密。如果下一跳又连接到 TCP/IP 网络上,用户数据和 TCP 报头在传输以前又将被加密。图 8-4(c)表示在数据链路层加密,在每个链路上除了数据链路层报头外,所有数据单元都被加密,但在路由器和网关之中所有数据单元都是明文形式。

8.2 Kerberos 认证系统

Kerberos 是 MIT 作为 Athena 计划的一部分开发的认证服务系统,Kerberos 系统建立了一个中心认证服务器用以向用户和服务器提供相互认证。目前该系统已有 5 个版本,其中 V1 到 V3 是内部开发版,V4 是 1988 年开发的,现已得到广泛应用,而 V5 则进一步对 V4 中的某些安全缺陷做了改进,已于 1994 年作为 Internet 标准(草稿)公布(RFC 1510)。

系统的目的是解决以下问题:在开放的分布式环境中,用户希望访问网络中的服务器,而服务器则要求能够认证用户的访问请求,并仅允许那些通过认证的用户访问服务器,以防未授权用户得到服务和数据。下面以 Kerberos V4 为例介绍该系统,系统使用的协议是基于上一章介绍的 Needham-Schroeder 认证协议。

8.2.1 Kerberos V4

如果网络环境未加任何保护手段,则任一用户都可获取任一服务器(V)提供的服务。这时明显的安全威胁是假冒,即敌手可假装是一客户以获取访问服务器的特权。为防止这种假冒,服务器应能够确定要求服务的客户的身份,但在开放环境中则给服务器增加了过重的负担。为此引入一个称为认证服务器 AS(authentication server)的第三方来承担对用户的认证,AS 知道每个用户的口令,并将口令存在一个中心数据库。用户如果想访问某一服务器,得首先向 AS 发出请求(其中包括用户的口令),AS 将收到的用户口令和中心数据库存储的口令相比较以验证用户的身份。如果验证通过,AS 则向用户发放一允许用户得到服务器服务的票据,用户则根据这一票据去获取服务器 V 的服务。

如果用户需多次访问同一服务器或不同服务器,则为了避免每次都重复以上获取票据的过程,再引入另一新服务器称为票据许可服务器 TGS(ticket-granting server)。TGS

向已经过 AS 认证的客户发放用于获取服务器 V 的服务的票据。为此用户应改为首先向 AS 获取访问 TGS 的票据 $Ticket_{tg}$ (称为票据许可票据) 存起来以后可反复使用。用户每次欲获得服务器 V 的服务时, 将 $Ticket_{tg}$ 出示给 TGS, TGS 再向用户发放获得服务器 V 服务的许可票据 $Ticket_v$ (称为服务许可票据)。

现在还有两个问题需加以解决, 一是票据许可票据 $Ticket_{tg}$ 的有效期限。如果有效期过短, 用户就需频繁地向 AS 输入自己的口令。如果过长, 则遭受敌手攻击的可能性就会增大。敌手通过对网络监听以获得用户的 $Ticket_{tg}$, 然后冒充合法用户向 TGS 申请获取服务器的服务。类似地, 敌手也可截获服务许可票据 $Ticket_v$ 。所以除了需对两个票据都加上合理的时间限制外, 还需保证客户持有的票据的确是发放给他的真实的票据。第 2 个需要解决的问题是服务器也应该向用户证明自己, 否则敌手可通过破坏网络结构而使用户发往服务器的消息到达另一假冒的服务器, 假冒的服务器获取用户的信息后再拒绝对用户提供服务。

以上是 Kerberos V4 过程的简要描述, 详细过程分为以下 3 个阶段, 共 6 步(见图 8-5)。

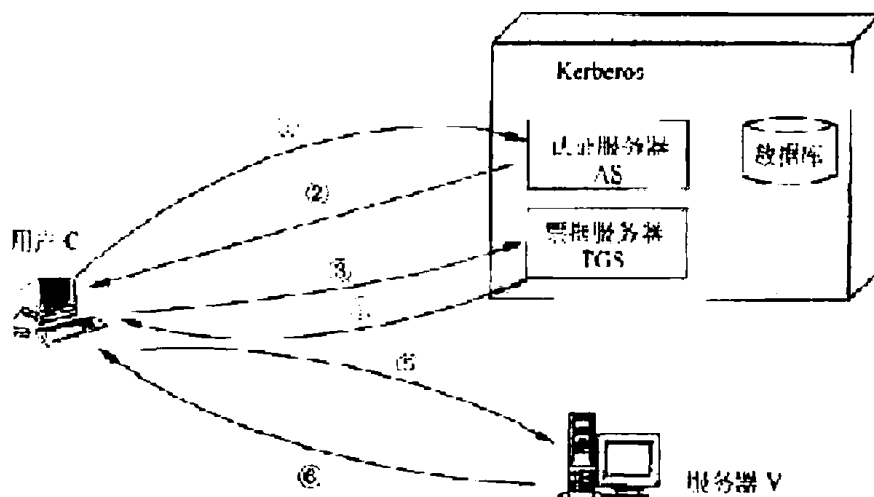


图 8-5 Kerberos 认证框图

首先来认识一下认证系统中的符号。

C: 客户机; AS: 认证服务器; V: 服务器; ID_c : 客户机用户的身份; TGS: 票据许可服务器; ID_v : 服务器 V 的身份; ID_{gs} : TGS 的身份; AD_c : C 的网络地址; P_c : C 上用户的口令; TS_i : 第 i 个时戳; $lifetime_i$: 第 i 个有效期限; K_c : 由用户口令导出的用户和 AS 的共享密钥; $K_{c,gs}$: C 与 TGS 的共享密钥; K_v : TGS 与 V 的共享密钥; K_{gs} : AS 与 TGS 的共享密钥; $K_{c,v}$: C 与 V 的共享密钥。

协议如下:

第 I 阶段(认证服务交换)用户从 AS 获取票据许可票据:

① $C \rightarrow AS: ID_C \parallel ID_{TGS} \parallel TS_1$ 。

② $AS \rightarrow C: E_{K_C}[K_{C,TGS} \parallel ID_{TGS} \parallel TS_2 \parallel lifetime_2 \parallel Ticket_{TGS}]$ 。

其中

$$Ticket_{TGS} = E_{K_{TGS}}[K_{C,TGS} \parallel ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_2 \parallel lifetime_2]$$

第 II 阶段(票据许可服务交换)用户从 TGS 获取服务许可票据:

③ $C \rightarrow TGS: ID_V \parallel Ticket_{TGS} \parallel Authenticator_C$ 。

④ $TGS \rightarrow C: E_{K_{C,TGS}}[K_{C,V} \parallel ID_V \parallel TS_4 \parallel Ticket_V]$ 。

其中

$$Ticket_{TGS} = E_{K_{TGS}}[K_{C,TGS} \parallel ID_C \parallel AD_C \parallel ID_{TGS} \parallel TS_2 \parallel lifetime_2]$$

$$Ticket_V = E_{K_V}[K_{C,V} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel lifetime_4]$$

$$Authenticator_C = E_{K_{C,TGS}}[ID_C \parallel AD_C \parallel TS_3]$$

第 III 阶段(客户机与服务服务器间的认证交换)用户从服务器获取服务:

⑤ $C \rightarrow V: Ticket_V \parallel Authenticator_V$ 。

⑥ $V \rightarrow C: E_{K_{C,V}}[TS_5 + 1]$ 。

其中

$$Ticket_V = E_{K_V}[K_{C,V} \parallel ID_C \parallel AD_C \parallel ID_V \parallel TS_4 \parallel lifetime_4]$$

$$Authenticator_C = E_{K_{C,V}}[ID_C \parallel AD_C \parallel TS_5]$$

具体解释如下:

① 客户向 AS 发出访问 TGS 的请求,请求中的时戳用以向 AS 表示这一请求是新的。

② AS 向 C 发出应答,应答由从用户的口令导出的密钥 K_C 加密,使得只有 C 能解读。应答的内容包括 C 与 TGS 会话所使用的密钥 $K_{C,TGS}$ 、用以向 C 表示 TGS 身份的 ID_{TGS} 、时戳 TS_2 、AS 向 C 发放的票据许可票据 $Ticket_{TGS}$ 以及这一票据的截止期限 $lifetime_2$ 。

③ C 向 TGS 发出一个由请求提供服务的服务器的身份、第②步获得的票据以及一个认证符构成的消息。其中认证符中包括 C 上用户的身份、C 的地址及一个时戳。认证符与票据不同,票据可重复使用且有效期较长,而认证符只能使用一次且有效期很短。TGS 用与 AS 共享的密钥 K_{TGS} 解密票据后知道 C 已从 AS 处得到与自己会话的会话密钥 $K_{C,TGS}$,票据 $Ticket_{TGS}$ 在这里的含义事实上是“使用 $K_{C,TGS}$ 的人就是 C”。TGS 也使用 $K_{C,TGS}$ 解读认证符,并将认证符中的数据与票据中的数据加以比较,从而可相信票据的发送者的确是票据的实际持有者,这时认证符的含义实际上是“在时间 TS_3 ,C 使用 $K_{C,TGS}$ ”。注意,

这时的票据不能证明任何人的身份,只是用来安全地分配密钥,而认证符则是用来证明客户的身份。因为认证符仅能被使用一次且其有效期限很短,所以可防止敌手对票据和认证符的盗取使用。

④ TGS 向 C 应答的消息由 TGS 和 C 共享的会话密钥加密后发往 C,应答消息中的内容有 C 和 V 共享的会话密钥 $K_{c,v}$ 、V 的身份 ID_v 、服务许可票据 *Ticket_v* 及票据的时戳,而票据中也包括应答消息中的上述数据项。

⑤ C 向服务器 V 发出服务许可票据 *Ticket_v* 和认证符 *Authenticator_c*。服务器解密票据后得到会话密钥 $K_{c,v}$,并由 $K_{c,v}$ 解密认证符,以验证 C 的身份。

⑥ 服务器 V 向 C 证明自己的身份。V 对从认证符得到的时戳加 1,再由与 C 共享的密钥加密后发给 C,C 解密后对增加的时戳加以验证,从而相信增加的时戳的确是 V。

整个过程结束以后,客户和服务器 V 之间就建立起了共享的会话密钥,以后可用来加密通信或者交换新的会话密钥。

8.2.2 Kerberos 区域与多区域的 Kerberos

Kerberos 的一个完整服务范围由一个 Kerberos 服务器、多个客户机和多个服务器构成,并且满足以下两个要求:

① Kerberos 服务器必须在它的数据库中存有所有用户的 ID 和口令的杂凑值,所有用户都已向 Kerberos 服务器注册。

② Kerberos 服务器必须与每一服务器有共享的密钥,所有服务器都已向 Kerberos 服务器注册。

满足以上两个要求的 Kerberos 的一个完整服务范围称为 Kerberos 的一个区域。网络中隶属于不同行政机构的客户和服务器则构成不同的区域,一个区域的用户如果希望得到另一个区域中的服务器的服务,则还需满足以下第③个要求。

③ 每个区域的 Kerberos 服务器必须和其他区域的服务器有共享的密钥,且两个区域的 Kerberos 服务器已彼此注册。

多区域的 Kerberos 服务还要求在两个区域间,第 1 个区域的 Kerberos 服务器信任第 2 区域的 Kerberos 服务器对本区域中用户的认证,而且第 2 区域的服务器也应信任第 1 区域的 Kerberos 服务器。

图 8-6 是两个区域的 Kerberos 服务示意图,其中区域 A 中的用户希望得到区域 B 中服务器的服务。为此,用户通过自己的客户机首先向本区域的 TGS 申请一个访问远程 TGS(即区域 B 中的 TGS)的票据许可票据,然后用这个票据许可票据向远程 TGS 申请获得服务器服务的服务许可票据。具体描述如下:

① 客户向本地 AS 申请访问本区域 TGS 的票据:

$C \rightarrow AS: ID_C \parallel ID_{AS} \parallel TS_1$ 。

② AS 向客户发放访问本区域 TGS 的票据:

$AS \rightarrow C: E_{K_c} [K_{c, tgs} \parallel ID_{tgs} \parallel TS_2 \parallel lifetime_2 \parallel Ticket_{tgs}]$ 。

③ 客户向本地 TGS 申请访问远程 TGS 的票据许可票据:

$C \rightarrow TGS: ID_{tgsrem} \parallel Ticket_{tgs} \parallel Authenticator_c$ 。

④ TGS 向客户发放访问远程 TGS 的票据许可票据:

$TGS \rightarrow C: E_{K_{c, tgs}} [K_{c, tgsrem} \parallel ID_{tgsrem} \parallel TS_4 \parallel Ticket_{tgsrem}]$ 。

⑤ 客户向远程 TGS 申请获得服务器服务的服务许可票据:

$C \rightarrow TGS_{rem}: ID_{vrem} \parallel Ticket_{tgsrem} \parallel Authenticator_c$ 。

⑥ 远程 TGS 向客户发放服务许可票据:

$TGS \rightarrow C: E_{K_{c, tgsrem}} [K_{c, vrem} \parallel ID_{vrem} \parallel TS_6 \parallel Ticket_{vrem}]$ 。

⑦ 客户申请远程服务器的服务:

$C \rightarrow V_{rem}: Ticket_{vrem} \parallel Authenticator_c$ 。

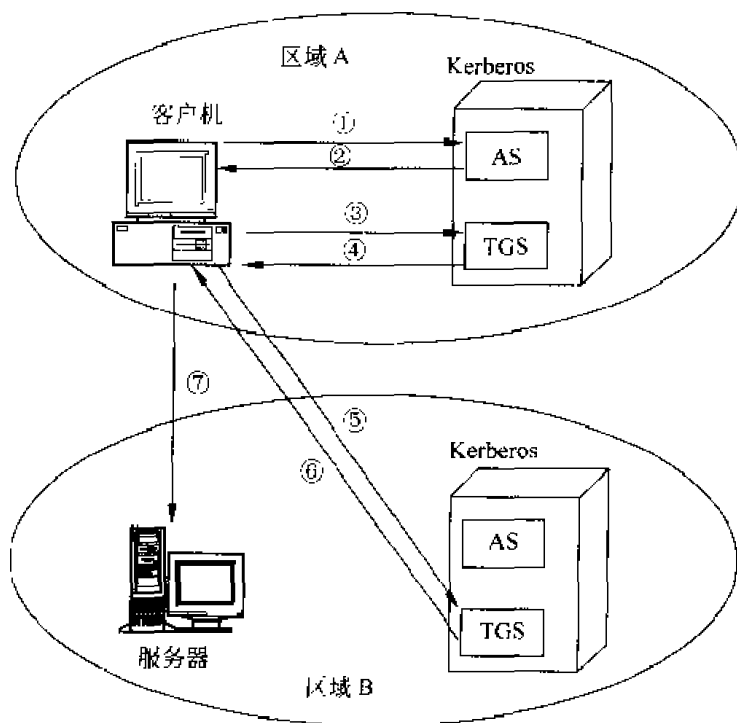


图 8-6 两个区域的 Kerberos 服务

对有很多个区域的情况来说,以上方案的扩充性不好,因为如果有 N 个区域,则必须有 $N(N-2)/2$ 次密钥交换才可使每个 Kerberos 区域和其他所有的 Kerberos 区域能够互操作,当 N 很大时,方案变得不现实。

8.3 X.509 认证业务

X.509 作为定义目录业务的 X.500 系列的一个组成部分,是由 ITU-T 建议的,这里所说的目录实际上是维护用户信息数据库的服务器或分布式服务器集合,用户信息包括用户名到网络地址的映射和用户的其他属性。X.509 定义了 X.500 目录向用户提供认证业务的一个框架,目录的作用是存放用户的公钥证书。X.509 还定义了基于公钥证书的认证协议。由于 X.509 中定义的证书结构和认证协议已被广泛应用于 S/MIME、IPSec、SSL/TLS 以及 SET 等诸多应用过程,因此 X.509 已成为一个重要的标准。

X.509 的基础是公钥密码体制和数字签字,但其中未特别指明使用哪种密码体制(建议使用 RSA),也未特别指明数字签字中使用哪种杂凑函数。

8.3.1 证书

1. 证书的格式

用户的公钥证书是 X.509 的核心问题,证书由某个可信的证书发放机构 CA 建立,并由 CA 或用户自己将其放入目录中,以供其他用户方便地访问。目录服务器本身并不负责为用户建立公钥证书,其作用仅仅是为用户访问公钥证书提供方便。

X.509 中公钥证书的一般格式如图 8-7(a)所示,证书中的数据域有:

① 版本号 默认值为第 1 版。如果证书中需有发放者惟一识别符或主体惟一识别符,则版本号一定是 2,如果有一个或多个扩充项,则版本号为 3。

② 顺序号 为一整数,由同一 CA 发放的每一证书的顺序号是惟一的。

③ 签字算法识别符 签署证书所用的算法及相应的参数。

④ 发放者名称 指建立和签署证书的 CA 名称。

⑤ 有效期 包括证书有效期的起始时间和终止时间两个数据项。

⑥ 主体名称 指证书所属用户的名称,即这一证书用来证明持有秘密钥用户的相应公开钥。

⑦ 主体的公开钥信息 包括主体的公开钥、使用这一公开钥的算法的标识符及相应的参数。

⑧ 发放者惟一识别符 这一数据项是可选用的,当发放者(CA)的名称被重新用于其他实体时,则用这一识别符来惟一标识发放者。

⑨ 主体惟一识别符 这一数据项也是可选用的,当主体的名称被重新用于其他实体时,则用这一识别符来惟一地标识主体。

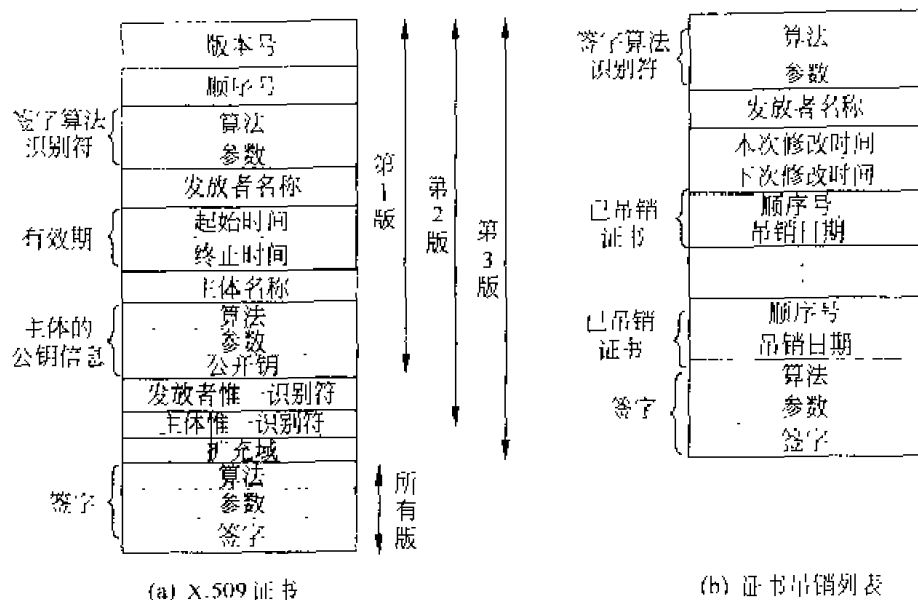


图 8-7 X.509 的证书格式和证书吊销列表

⑩ 扩充域 其中包括一个或多个扩充的数据项,仅在第3版中使用。

⑪ 签字 CA 用自己的秘密钥对上述域的杂凑值签字的结果,此外,这个域还包括签字算法标识符。

X.509 中使用以下表示法来定义证书:

$$CA\langle\langle A \rangle\rangle = CA\{V, SN, AI, CA, T_A, A, \Lambda_P\}$$

其中 $Y\langle\langle X \rangle\rangle$ 表示证书发放机构 Y 向用户 X 发放的证书, $Y\{I\}$ 表示 I 链接上 Y 对 I 的杂凑值的签字。

2. 证书的获取

CA 为用户产生的证书应有以下特性:

- ① 其他任一用户只要得到 CA 的公开钥,就能由此得到 CA 为该用户签署的公开钥。
- ② 除 CA 以外,任何其他人都不能以不被察觉的方式修改证书的内容。

因为证书是不可伪造的,因此放在目录后无需对目录施加特别的保护措施。

如果所有用户都由同一 CA 为自己签署证书,则这一 CA 就必须取得所有用户的信任。用户证书除了能放在目录中以供他人访问外,还可以由用户直接发给其他用户。用户 B 得到用户 A 的证书后,可相信用 A 的公开钥加密的消息不会被他人获悉,还相信用 A 的秘密钥签署的消息是不可伪造的。

如果用户数量极多,则仅一个 CA 负责为用户签署证书就有点不现实,因为每一用户都必须以绝对安全(指完整性和真实性)的方式得到 CA 的公开钥,以验证 CA 签署的证

书。因此在用户数目极多的情况下,应有多个 CA,每一 CA 仅为一部分用户签署证书。

设用户 A 已从证书发放机构 X_1 处获取了公钥证书,用户 B 已从 X_2 处获取了证书。如果 A 不知 X_2 的公开钥,他虽然能读取 B 的证书,但却无法验证 X_2 的签字,因此 B 的证书对 A 来说是没有用处的。然而,如果两个 CA X_1 和 X_2 彼此间已经安全地交换了公开钥,则 A 可通过以下过程获取 B 的公开钥:

① A 从目录中获取由 X_1 签署的 X_2 的证书,因 A 知道 X_1 的公开钥,所以能验证 X_2 的证书,并从中得到 X_2 的公开钥。

② A 再从目录中获取由 X_2 签署的 B 的证书,并由 X_2 的公开钥对此加以验证,然后从中得到 B 的公开钥。

以上过程中, A 是通过一个证书链来获取 B 的公开钥,证书链可表示为

$$X_1 \langle X_2 \rangle X_2 \langle B \rangle$$

类似地, B 能通过相反的证书链获取 A 的公开钥,表示为 $X_2 \langle X_1 \rangle X_1 \langle A \rangle$ 。

以上证书链中有两个证书, N 个证书的证书链可表示为

$$X_1 \langle X_2 \rangle X_2 \langle X_3 \rangle \cdots X_N \langle B \rangle$$

此时任意两个相邻的 CA X_i 和 X_{i+1} 已彼此间为对方建立了证书,对每一 CA 来说,由其他 CA 为这一 CA 建立的所有证书都应存放于目录中,并使用户知道所有证书相互之间的连接关系,从而可获取另一用户的公钥证书。X.509 建议将所有 CA 以层次结构组织起来。

图 8-8 是 X.509 的 CA 层次结构的一个例子,其中的内部结点表示 CA,叶结点表示用户。用户 A 可从目录中得到相应的证书以建立到 B 的以下证书链:

$$X \langle W \rangle W \langle V \rangle V \langle Y \rangle Y \langle Z \rangle Z \langle B \rangle$$

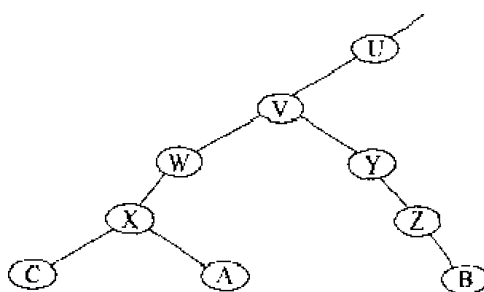


图 8-8 X.509 的层次结构

并通过该证书链获取 B 的公开钥。

类似地, B 可建立以下证书链以获取 A 的公开钥:

$$Z \langle Y \rangle Y \langle V \rangle V \langle W \rangle W \langle X \rangle X \langle A \rangle$$

3. 证书的吊销

每一证书都有一有效期,然而有些证书还未到截止日期就会被发放该证书的 CA 吊销,这是由于用户的秘密钥有可能已被泄露,或者该用户不再由该 CA 来认证,或者 CA 为该用户签署证书的秘密钥有可能已泄露。为此每一 CA 还必须维护一个证书吊销列表 CRL(certification revocation list),见图 8-7(b),其中存放所有未到期而被提前吊销的证书,包括该 CA 发放给用户和发放给其他 CA 的证书。CRL 还必须经该 CA 签字,然后存放于目录以供他人查询。

CRL 中的数据域包括发放者 CA 的名称、建立 CRL 的日期、计划公布下一 CRL 的日期,以及每一被吊销的证书数据域,而被吊销的证书数据域包括该证书的顺序号和被吊销的日期。因为对一个 CA 来说,他发放的每一证书的顺序号是惟一的,所以可用顺序号来识别每一证书。

每一用户收到他人消息中的证书时,都必须通过目录检查这一证书是否已被吊销。为避免搜索目录引起的延迟以及由此而增加的费用,用户自己也可维护一个有效证书和被吊销证书的局部缓存区。

8.3.2 认证过程

X. 509 有 3 种认证过程以适应不同的应用环境。3 种认证过程都使用公钥签字技术,并假定通信双方都可从目录服务器获取对方的公钥证书,或对方最初发来的消息中包括公钥证书,即假定通信双方都知道对方的公钥。3 种认证过程如图 8-9 所示。

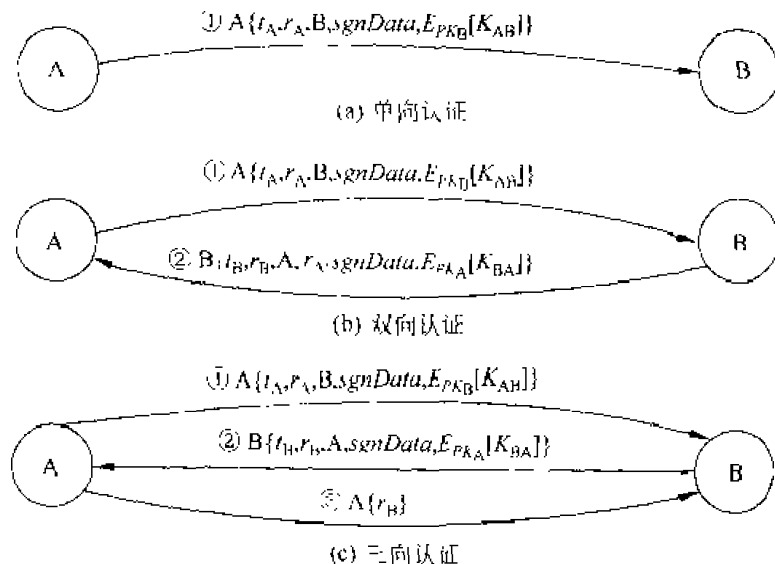


图 8-9 X. 509 的认证过程

1. 单向认证

单向认证指用户 A 将消息发往 B,以向 B 证明: A 的身份、消息是由 A 产生的;消息的意欲接收者是 B;消息的完整性和新鲜性。

为实现单向认证, A 发往 B 的消息应是由 A 的秘密钥签署的若干数据项组成。数据项中应至少包括时戳 t_A 、一次性随机数 r_A 、B 的身份,其中时戳又有消息的产生时间(可选项)和截止时间,以处理消息传送过程中可能出现的延迟,一次性随机数用以防止重放攻击。 r_A 在该消息未到截止时间以前应是这一消息惟一所有的,因此 B 可在这一消息的截止时间以前,一直存有 r_A ,以拒绝具有相同 r_A 的其他消息。

如果仅单纯为了认证,则 A 发往 B 的上述消息就可作为 A 提交给 B 的凭证。如果不单纯为了认证,则 A 用自己的公开钥签署的数据项还可包括其他信息 $sgnData$,将这个信息也包括在 A 签署的数据项中可保证该信息的真实性和完整性。数据项中还可包括由 B 的公开钥 PK_B 加密的双方意欲建立的会话密钥 K_{AB} 。

2. 双向认证

双向认证是在上述单向认证的基础上, B 再向 A 作出应答,以证明: B 的身份、应答消息是由 B 产生的;应答的意欲接收者是 A;应答消息是完整的和新鲜的。

应答消息中包括由 A 发来的 一次性随机数 r_A (以使应答消息有效)、由 B 产生的时戳 t_B 和 一次性随机数 r_B 。与单向认证类似,应答消息中也可包括其他附加信息和由 A 的公开钥加密的会话密钥。

3. 三向认证

在上述双向认证完成后, A 再对从 B 发来的 一次性随机数签字后发往 B,即构成第三向认证。三向认证的目的是双方将收到的对方发来的一次性随机数又都返回给对方,因此双方不需检查时戳只需检查对方的一次性随机数即可检查出是否有重放攻击。在通信双方无法建立时钟同步时,就需使用这种方法。

8.4 PGP

PGP(pretty good privacy)是目前使用最为普遍的一种电子邮件系统,该系统能为电子邮件和文件存储应用过程提供认证业务和保密业务。

本节分别介绍 PGP 的运行方式、密钥的产生和存储以及公钥的管理。

8.4.1 运行方式

PGP 有 5 种业务：认证性、保密性、压缩、电子邮件的兼容性、分段。表 8-1 是这 5 种业务的总结。其中 CAST-128 是一种分组密码，算法具有传统 Feistel 网络结构，采用 16 轮迭代，明文分组长度为 64 比特，密钥长以 8 比特为增量，从 40 比特到 128 比特可变。

表 8-1 PGP 的业务

功 能	所用算法	描 述
数字签字	DSS SHA 或 RSA SHA	发送方使用 SHA 产生消息摘要，再用自己的秘密钥按 DSS 算法或 RSA 算法对消息摘要签字。
消息加密	CAST 或 IDEA 或三个密钥的三重 DES/ElGamal 或 RSA	消息由用户产生的一次性会话密钥按 CAST-128 或 IDEA 或三重 DES 加密，会话密钥用接收方的公开钥按 ElGamal 或 RSA 加密。
压缩	ZIP	消息经 ZIP 算法压缩后存储或传送。
电子邮件的兼容性	基数 64 变换	使用基数 64 变换将加密的消息转换为 ASCII 字符串，以提供电子邮件应用系统的透明性。
分段		对消息进行分段和重组以适应 PGP 对消息最大长度的限制。

图 8-10 是 PGP 的认证业务和保密业务示意图，其中 K_s 为分组加密算法所用的会话密钥， EC 和 DC 分别为分组加密算法和解密算法， EP 和 DP 分别为公钥加密算法和解密算法， SK_A 和 PK_A 分别为发送方的秘密钥和公开钥， SK_B 和 PK_B 分别为接收方的秘密钥和公开钥， H 表示杂凑函数， \parallel 表示链接， Z 为 ZIP 压缩算法， $R64$ 表示基 64 变换。

1. 认证业务

图 8-10(a)表示 PGP 中通过数字签字提供认证的过程，分为 5 步：

- ① 发送方产生消息 M 。
- ② 用 SHA 产生 160 比特长的消息摘要 $H(M)$ 。
- ③ 发送方用自己的秘密钥 SK_A 按 RSA 算法对 $H(M)$ 加密，并将加密结果 $EP_{SK_A}[H(M)]$ 与 M 链接后发送。
- ④ 接收方用发送方的公开钥对 $EP_{PK_A}[H(M)]$ 解密得 $H(M)$ 。
- ⑤ 接收方对收到的 M 计算消息摘要，并与④中的 $H(M)$ 比较。如果一致，则认为 M 是真实的。

过程中结合使用了 SHA 和 RSA 算法，类似地也可结合使用 DSS 算法和 SHA 算法。

以上过程将消息的签字与消息链接后一起发送或存储。在有些情况下，需要将消息的签字与消息分开发送或存储。例如将可执行程序的签字分开存储，以后可用来检查程

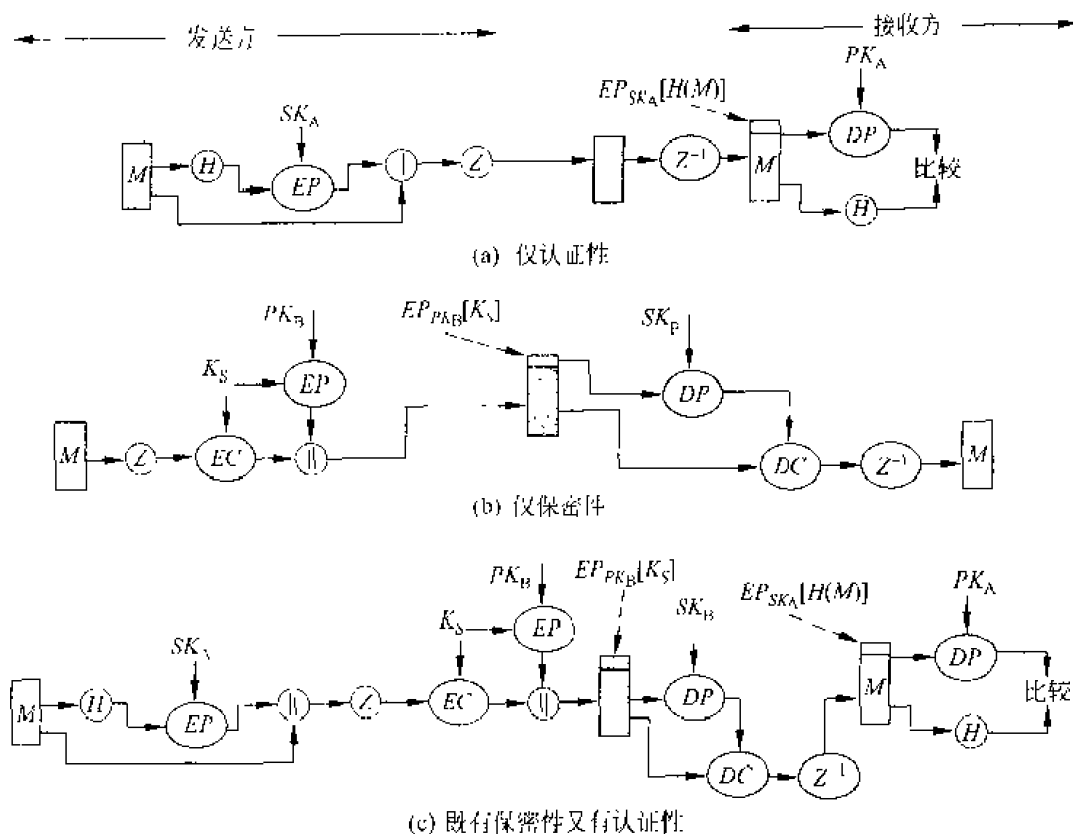


图 8-10 PGP 的认证业务和保密业务

序是否有病毒感染。再如多人签署同一文件(如法律合同),每人的签字都应与被签文件分开存放。否则第一个人签完字后将消息与签字链接在一起,第 2 个人签字时既要签消息,又要签第 1 个人的签字,因此就形成了签字的嵌套。

2. 保密业务

PGP 的另一业务是为传输或存储的文件提供加密的保密性业务。加密算法用 CAST-128,也可用 IDEA 或三重 DES,运行模式为 64 比特 CFB 模式。加密算法的密钥为一次性的,即每加密一消息时都需产生一新的密钥,称为一次性会话密钥,且新密钥也需用接收方的公开钥加密后与消息一起发往接收方,整个过程如下(见图 8-10(b)):

- ① 发送方产生消息 M 及一次性会话密钥 K_S 。
- ② 用密钥 K_S 按 CAST-128(或 IDEA 或 3DES)加密 M 。
- ③ 用接收方的公开钥 PK_B 按 RSA 算法加密一次性会话密钥 K_S ,将②、③中的两个加密结果链接起来发往接收方。
- ④ 接收方用自己的秘密钥按 RSA 算法恢复一次性会话密钥。

⑤ 接收方用一次性会话密钥恢复发送方发来的消息。

PGP 为加密一次性会话密钥还提供了 ElGamal 算法以供选用。

以上方案有以下几个优点：首先，由于分组加密速度远快于公钥加密速度，因此使用分组加密算法加密消息，使用公钥加密算法加密一次性会话密钥可比单纯使用公钥算法大大地减少加密时间；第二，因为会话密钥是一次性的，因此没有必要使用会话密钥的交换协议，同时，由于电子邮件的存储转发特性，也无法使用握手交换协议，本方案使用公钥加密算法来传送一次性会话密钥，保证了仅接收方能得到；第三，一次性会话密钥的使用进一步加强了本来就很强的分组加密算法，因此只要公钥加密算法是安全的，整个方案就是安全的，PGP 可允许用户选择的密钥长度范围为 768 比特到 3072 比特，而若使用 DSS，则其密钥限制为 1024 比特。

3. 保密性与认证性

如果对同一消息同时提供保密性与认证性，可使用图 8-10(c)的方式。发送方首先用自己的秘密钥对消息签字，将明文消息和签字链接在一起，再使用一次性会话密钥按 CAST-128(或 IDEA 或 3DES)对其加密，同时用 ElGamal 算法对会话密钥加密，最后将两个加密结果一同发往接收方。这一过程中，先对消息签字再对签字加密。这一顺序优于先加密、再对加密结果签字。这是因为将签字与明文消息在一起存储比与密文消息在一起存储会带来很多方便，同时也给第三方对签字的验证带来方便。

4. 压缩

图 8-10 中 Z 表示 ZIP 压缩算法，Z⁻¹表示解压算法。压缩的目的是为邮件的传输或文件的存储节省空间。压缩运算的位置是在签字以后、加密以前，压缩前产生签字的原因有二：

① 对不压缩的消息签字，可便于以后对签字的验证。如果对压缩后的消息签字，则为了以后对签字的验证，还需存储压缩后的消息或在验证签字时对消息重做压缩。

② 即使用户愿意对压缩后的消息签字且愿意验证时对原消息重做压缩，实现起来也极为困难，这是因为 ZIP 压缩算法是不确定性的，该算法在不同的实现中会由于在运行速度和压缩率之间产生不同的折中，因而产生出不同的压缩结果(虽然解压结果相同)。

对消息压缩后再进行加密可加强其安全性，这是因为消息压缩后比压缩前的冗余度要小，因此会使得密码分析更为困难。

5. 电子邮件的兼容性

PGP 在如图 8-10 所示的 3 种业务中，传输的消息都有被加密的部分(也许是所有部分)，这些部分构成了任意 8 比特位组串。然而许多电子邮件系统只允许使用 ASCII 文

本串,为此 PGP 提供了将 8 比特位串转换为可打印的 ASCII 字符的服务。转换方法是使用基数 64 变换,将每 3 个 8 比特位组的二元数据映射为 4 个 ASCII 字符。基数 64 变换可将被变换的消息扩展 33%,然而由于扩展是对会话密钥和消息的签字部分进行,而这一部分又比较紧凑的,所以对明文消息的压缩足以弥补基数 64 变换所引起的扩展。有实例显示,ZIP 的平均压缩率大约为 2.0,因此如果不考虑相对小的签字和密钥部分,对长度为 X 的文件来说,压缩和扩展的总体效果为 $1.33 \times 0.5 \times X = 0.665 \times X$,即总体上有 1/3 的压缩率。

PGP 变换具有“盲目性”,即不管输入变换的消息内容是否是 ASCII 文件,都将变换为基数 64 格式。因此在图 8-10 所示的仅提供认证的服务中,对消息及其签字进行基数 64 变换,变换后的结果对不经意的观察者来说是不可读的,从而可提供一定程度的保密性。作为一种配置选择,PGP 可以只将消息的签字部分转换为基数 64 格式,从而使得接收方不使用 PGP 就可阅读消息,但对签字的验证仍然需要使用 PGP。

图 8-11 分别是发送方和接收方对消息的处理过程框图,发送方首先对消息的杂凑值签字(如果需要的话),然后明文消息及其签字(如果有的话)再经压缩函数压缩。如果要求保密性,则用一次性会话密钥按分组加密算法加密压缩结果,同时用公钥加密算法加密一次性会话密钥。将两个加密结果链接在一起后,再经基数 64 变换为基数 64 格式。

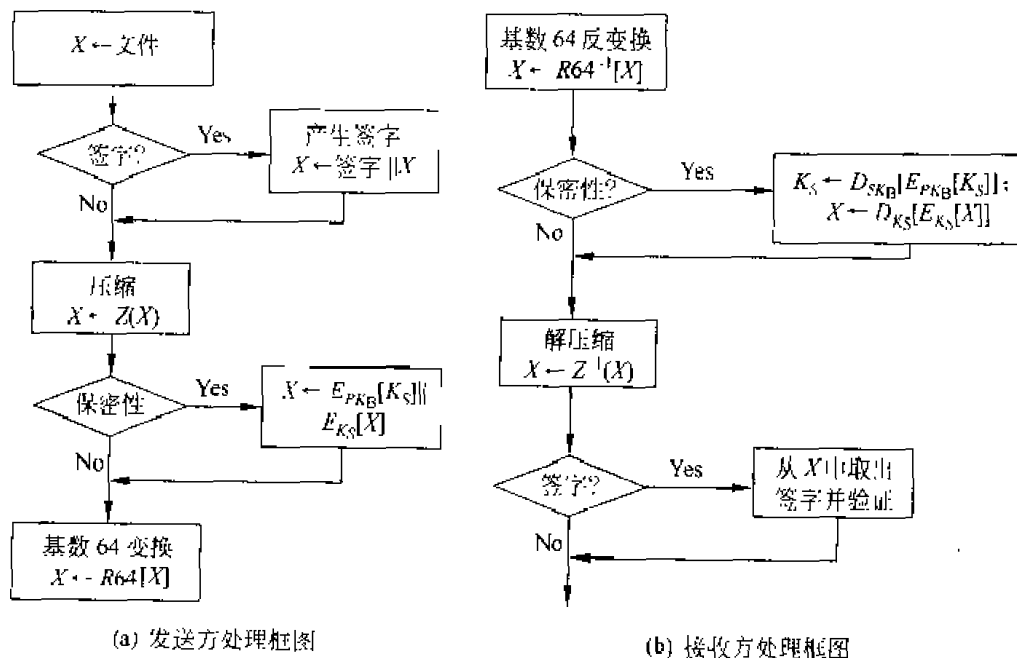


图 8-11 PGP 的消息处理框图

接收方首先将接收到的结果由基数 64 格式转换为二元数字串。第 2 步,如果消息是

密文,则恢复一次性会话密钥,由一次性会话密钥恢复加密的消息,并对之解压缩。如果消息还经过签字,则从上一步恢复的消息中取出消息的杂凑值,并与自己计算的消息的杂凑值进行比较。

6. 分段与重组

电子邮件通常都对最大可用的消息长度有所限制,如果消息长度大于最大可用长度,则将消息分为若干子段并分别发送。分段是在图 8-11(a)基数 64 变换以后进行,因此会话密钥报头和签字报头仅在第一子段的开头处出现一次。接收方在图 8-11(b)的处理过程以前,首先去掉第一子段开头处的报头,再将各子段拼装在一起。

8.4.2 密钥和密钥环

PGP 所用的密钥有 4 类:分组加密算法所用的一次性会话密钥和基于通行字短语的密钥,公钥加密算法所用的公开钥和秘密钥。为此 PGP 必须满足以下 3 个要求:

- 能够产生不可猜测的会话密钥;
- 用户可有多个公开钥-秘密钥对,这是因为用户可能希望随时更换自己的密钥对,另一方面用户可能希望在同一时间和多个通信方同时通信时分别使用不同的密钥对,或者用户可能希望通过限制一个密钥加密的内容的数量来增加安全性,所以用户和他的密钥对不是一一对应的关系,必须采取某一方式对密钥加以识别;
- PGP 的每一用户都必须对存储自己密钥对的文件加以维护,同时还需对存储所有通信对方公钥的文件加以维护。

1. 会话密钥的产生

会话密钥的使用是一次性的,其中 CAST-128 和 IDEA 所用会话密钥长为 128 比特,3DES 所用的会话密钥长为 168 比特。下面以 CAST-128 为例介绍其密钥的生成。

产生 CAST-128 密钥的随机数产生器仍由 CAST-128 加密算法构成(构成方式略),其输入为一个 128 比特的密钥和两个 64 比特的明文,采用 CFB 模式,对两个明文分组加密,再将得到的两个 64 比特密文分组链接在一起即形成所要产生的 128 比特密钥。其中两个 64 比特(即 128 比特)的明文分组由用户随机地键盘输入而得,将输入的一个字符表示成 8 比特的数值,共随机输入 12 个字符,得 96 比特长的数值,剩下 32 比特则用来表示键盘输入所用的时间。随机数产生器输入的 128 比特长的密钥则取为它上一次输出的 128 比特长的会话密钥。

2. 密钥识别符

如前所述,PGP 在对消息加密的同时,还需用接收方的公开钥对一次性会话密钥加密。从而使得只有接收方能恢复会话密钥,进而恢复加密的消息。如果接收方只有一个

密钥对(即公开钥-秘密钥对),就可直接恢复会话密钥。然而接收方通常都有多个密钥对,他怎么知道会话密钥是用他的哪个公开钥加密的?一种解决办法是发送方将所用的接收方的公开钥一起发给接收方,这种方法对空间的浪费太多,RSA 的公开钥其长度可达数百位十进制数。另一种办法是对每一用户的每一公开钥都指定一个惟一的识别符,称为密钥 ID ,因此发送方用接收方的哪个公开钥就将这个公开钥的 ID 发给接收方。但使用这种方法必须考虑密钥 ID 的存储和管理,且收发双方都必须能够从密钥 ID 得到对应的公开钥,从而增加了不必要的负担。PGP 采用的方法是用公开钥中 64 个最低有效位表示该密钥的 ID ,即公开钥 PK_A 的 ID 是 $PK_A \bmod 2^{64}$ 。由于 64 位已足够长,因而不同密钥的 ID 相重的概率非常小。

PGP 在数字签字时 also 需对密钥加上识别符,这是因为发送方签字时可能有很多秘密钥可供使用,接收方必须知道使用发送方的哪一个公开钥来验证数字签字。PGP 用签字中的 64 比特来表示相应公开钥的 ID 。

3. PGP 的消息格式

图 8-12 表示 PGP 中发送方 A 发往接收方 B 的消息格式,其中 E_{PK_B} 表示用接收方 B 的公开钥加密, E_{SK_A} 表示用发送方 A 的秘密钥加密(即 A 的签字), E_{K_S} 表示用一次性会话密钥 K_S 的加密,ZIP 是压缩算法,R64 是基数 64 变换。

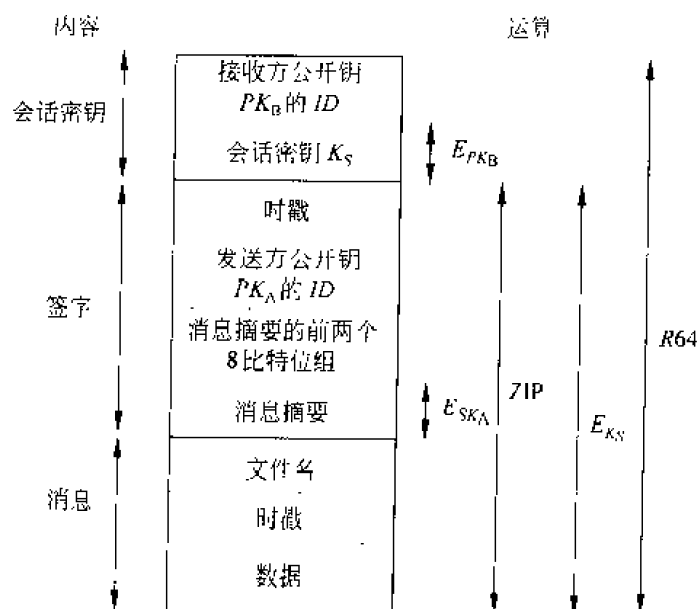


图 8-12 PGP 的消息格式

PGP 的消息由 3 部分组成:消息、消息的签字(可选)、会话密钥(可选)。

消息部分包括被存储或被发送的实际数据、文件名以及时戳,时戳用来表示产生消息的时间。

签字部分包括以下成分:

① 时戳 产生签字的时间。

② 消息摘要 消息摘要是由 SHA 对签字的时戳链接上消息本身后求得的 160 比特输出,再由发送方用秘密钥签字。求消息摘要时以签字时戳作为输入的一部分,其目的是防止重放攻击,而不以消息的文件名和产生消息的时戳作为输入的一部分,其目的是使得对无报头域的实际数据计算的签字与作为前缀而附加在消息前的签字完全一样。

③ 消息摘要的前两个 8 比特位组 接收方用于与解密消息摘要后得到的前两个 8 比特位组进行比较,以确定自己在验证发送方的数字签字时是否正确地使用了发送方的公开钥。消息摘要的前两个 8 比特位组,也可用作消息的 16 比特帧校验序列。

④ 发送方公开钥的 ID 用于标识解密消息摘要(即验证签字)的公开钥,相应地也标识了签字的秘密钥。

消息部分和签字部分经 ZIP 算法压缩后再用会话密钥加密。

会话密钥部分包括会话密钥和接收方公开钥标识符,标识符用以识别发送方加密会话密钥时用接收方的哪个公开钥。

发送消息前,对整个消息作基数 64 变换。

4. 密钥环

为了有效存储、组织密钥,同时也为了便于用户的使用,PGP 为每个结点(即用户)都提供了两个表型数据结构,一个用于存储用户自己的密钥对(即公开钥/秘密钥),另一个用于存储该用户所知道的其他各用户的公开钥。这两个数据结构分别称为秘密钥环和公钥环,如表 8-2 所示,其中带 * 的字段可作为标识字段。

表 8-2 密钥环

(a) 秘密钥环

时 戳	密 钥 ID*	公 开 钥	被加密的秘密钥	用 户 ID*
⋮	⋮	⋮	⋮	⋮
T_i	$PK_i \bmod 2^{64}$	PK_i	$E_{H(P_i)}[SK_i]$	用户 i
⋮	⋮	⋮	⋮	⋮

(b) 公钥环

时戳	密钥 ID*	公开钥	拥有者可信字段	用户 ID*	密钥合法性字段	签字	签字可信字段
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	$PK_i \bmod 2^{64}$	PK_i	trust_flag _i	用户 i	trust_flag _i	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

在秘密钥环中,每行表示该用户的一个密钥对,其数据项有:产生密钥对的时戳、密钥 ID、公开钥、被加密的秘密钥、用户 ID。其中密钥 ID 和用户 ID 可作为该行的标识符。用户 ID 可用用户的邮件地址,用户也可以为一个密钥对用多个不同的用户 ID,还可以在不同的密钥对中使用相同的用户 ID。

秘密钥环由用户自己存储,仅供用户自己使用,而且为使秘密钥尽可能地安全,秘密钥是通过 CAST 128(或 IDEA 或 3DES)加密后以密文形式存储,加密过程为:用户首先选择一个通行字短语作为 SHA 的输入,产生出一个 160 比特的杂凑值后销毁通行字短语,再用杂凑值的 128 比特作为密钥按 CAST-128 对秘密钥加密,加密完成后再销毁杂凑值。以后若要取出秘密钥,必须重新输入通行字短语,PGP 产生出通行字的杂凑值,并以此杂凑值为秘密钥按 CAST-128 解密被加密的秘密钥。

从加密秘密钥的过程可见,秘密钥的安全性取决于所用通行字的安全性,所以用户使用的通行字应是易于记住的但又是不易被他人猜出的。

公钥环中每行存储的是该用户所知道的其他用户的公开钥,其数据项包括:时戳(表示这一行产生的时间)、密钥 ID(指这一行的公开钥)、公开钥、用户 ID(指该公开钥的属主),其中密钥 ID 和用户 ID 可作为该行的标识符,还有其他几个数据项以后再介绍。

下面介绍消息传输和接收时,密钥环是如何使用的。为简单起见,下面过程中省略了压缩过程和基数 64 变换过程。

假定消息既要被签字,也要被加密,则发送方 A 需执行以下过程(见图 8-13,其中 RNG 是随机数产生器,其他符号和图 8-10 相同):

(1) 签署消息

① PGP 使用 A 的用户 ID 作为索引(即关键字)从 A 的秘密钥环中取出 A 的秘密钥。如果用户 ID 为缺省值,则从秘密钥环中取出第一个秘密钥。

② PGP 提示用户输入通行字短语用于恢复被加密的秘密钥。

③ 由 A 的秘密钥产生消息的签字。

(2) 加密消息

① PGP 产生一会话密钥,并由会话密钥对消息及签字加密。

② PGP 使用接收方 B 的用户 ID 作为关键字,从公钥环中取出 B 的公开钥。

③ PGP 用 B 的公开钥加密会话密钥以形成发送消息中的会话密钥部分。

接收方 B 执行以下过程(见图 8-14):

(1) 解密消息

① PGP 从接收到的消息中会话密钥部分取出接收方 B 的密钥 ID,并以此作为关键字从 B 的秘密钥环中取出相应的被加密的秘密钥。

② PGP 提示 B 输入通行字短语以恢复秘密钥。

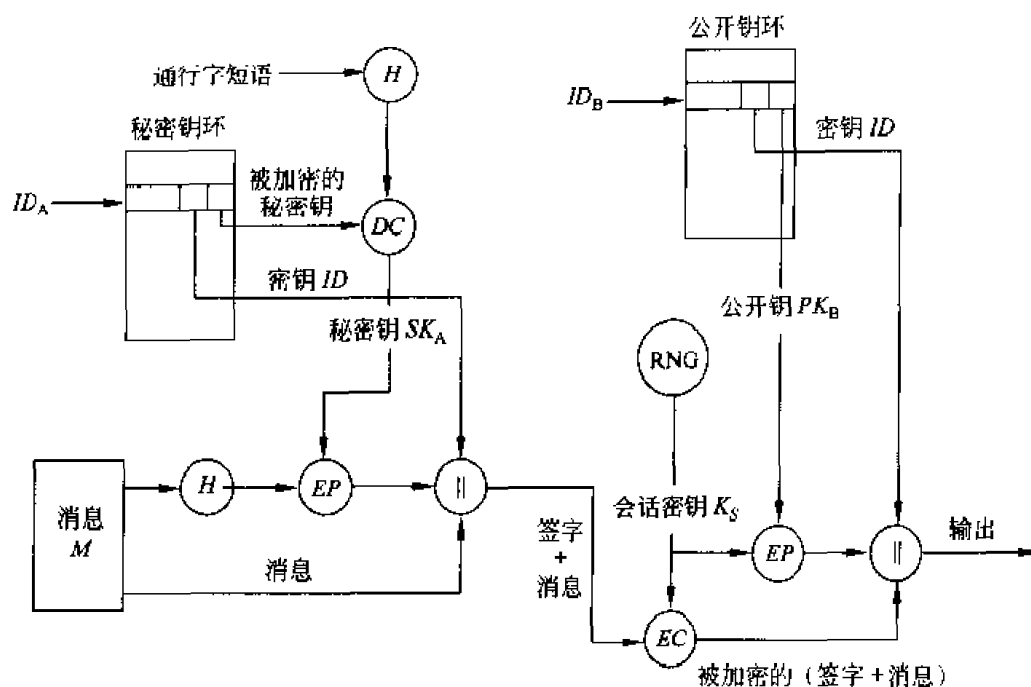


图 8-13 PGP 的消息产生过程

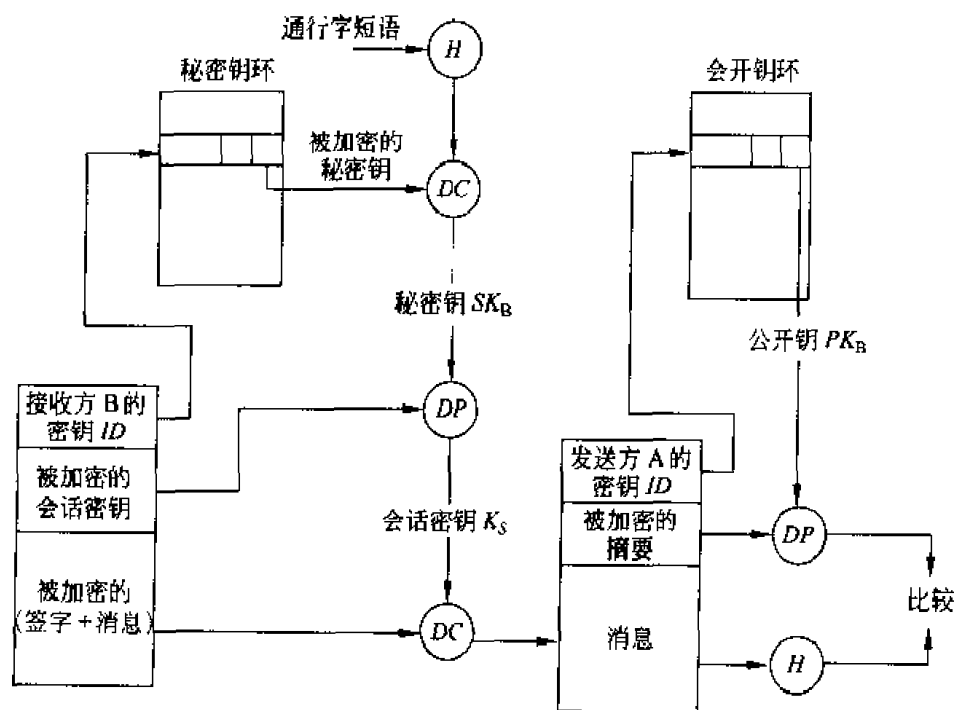


图 8-14 PGP 的消息接收过程

③ PGP 用秘密钥恢复出会话密钥,并进而解密消息。

(2) 认证消息

① PGP 从收到的消息中签字部分取出发送方 A 的密钥 ID,并以此作为关键字从发送方的公钥环中取出发送方的公开钥。

② PGP 用发送方的公开钥恢复消息摘要。

③ 对收到的消息重新求消息摘要,并与恢复出的消息摘要进行比较。

8.4.3 公钥管理

如何保护公钥不被他人窜扰是公钥密码体制中最为困难的问题,也是公钥密码体制的一个薄弱环节,很多软件复杂性都是由于解决这一问题而产生的。

PGP 由于可用于各种环境,所以未建立严格的公钥管理方案,而是提供了一种解决公钥管理问题的结构,其中有好几种建议的方案可供选用。

1. 公钥管理方法

用户 A 为了使用 PGP 和其他用户通信,必须建立一个公钥环,用以存放其他用户的公开钥。如果 A 的公钥环中有一公开钥表明是属于 B 的,但实际上是属于 C 的,比如说 A 是通过 B 发布公开钥的公告牌系统 BBS(bulletin board system)获取 B 的公开钥,但在 A 获取之前,C 就已将 B 的公开钥给替换了,这就可能有以下两种危险:第一,C 可以假冒 B 对伪造的消息签字,再发给 A,A 收到 C 发来的消息却以为是 B 发来的;第二,A 发给 B 的任何加密的消息都被 C 解读。

所以必须采取措施以减小用户公钥环中包含虚假公钥的危险。用户 A 可采取以下措施以获取用户 B 的可靠公开钥:

① 物理手段 B 可将自己的公开钥存在一软盘中,亲手交给 A。这个方法最为安全,但受实际应用的限制。

② 通过电话验证 如果 A 能在电话中识别 B,就可要求 B 以基数 64 的格式在电话中口述自己的公开钥。更实际的方法是 B 通过电子邮件向 A 发送自己的公开钥,A 通过 PGP 产生公开钥的 160 比特的摘要,并以十六进制格式显示,称其为公开钥的指纹。然后 A 要求 B 在电话中口述公开钥的指纹,如果两个指纹一致,B 的公开钥就得以验证。

③ 通过 A、B 都信赖的第三方,即介绍人 D D 首先建立一个证书,其中包括 B 的公开钥、公开钥建立的时间、公开钥的有效期,然后用 SHA 求出证书的数字摘要,并用自己的秘密钥为数字摘要签字,再将签字链接在证书后由 B 或 D 发给 A,或在 BBS 中发布。因为其他人不知道 D 的秘密钥,所以即使能够伪造 B 的公开钥也无法伪造 D 的签字。

④ 通过可信的证书发放机构 方法同③。

后两种措施中, A 为获取 B 的公开钥, 必须首先获取可信赖的第三方或可信赖的证书发放机构的公开钥, 并相信该公开钥的有效性。所以最终还取决于 A 对第三方或证书发放机构的信任程度。

2. PGP 中的信任关系

PGP 中虽然未对建立证书发放机构或建立可信赖机构作任何说明, 但却提供了一种方便的方法来使用 PGP 中的信任关系, 建立用户对公开钥的信任程度。

PGP 建立信任关系的基本方法是用户在建立公钥环时, 以一个公钥证书作为公钥环中的一行。其中有 3 个数据项用来表示对该公钥证书的信任程度(见表 8-2)。

① 密钥合法性字段(key legitimacy field) 表示 PGP 以多大程度信任这一公开钥是用户的有效公开钥, 该字段是由 PGP 根据这一证书的签字可信字段计算的。

② 签字可信字段(signature trust field) 拥有该公开钥的用户可收集 0 个或多个为该公钥证书的签字, 而每一签字后面有--签字可信字段, 用来表示该用户对签字者的信任程度。

③ 拥有者可信字段(owner trust field) 用于表示用这一公开钥签署其他公钥证书的可信程度, 这一字段由用户自己指定。

以上 3 个字段的取值是用来表示信任程度的标志, 标志的具体含义在此不再赘述。

假定用户 A 已有一个公钥环, PGP 对公钥环的信任处理过程如下:

① 当 A 在公钥环中插入一新的公开钥时, PGP 必须为拥有者可信字段指定一个标志。如果 A 插入的新公开钥是自己的, 则也将被插入到 A 的秘密钥环, PGP 自动地指定标志为 ultimate trust (绝对可信)。否则 PGP 要求 A 为该字段指定一个标志, A 指定的标志可以是 unknown、untrusted、marginally trusted、completely trusted 中的一个, 这 4 个标志的含义分别为不相识(与该公开钥的拥有者)、不信任、勉强信任、完全信任。

② 当新公开钥插入公钥环时, 新公钥可能已有一个或多个签字, 以后可能还会为这一公钥搜集多个签字。当为该公钥插入一新签字时, PGP 将在公钥环中查看签字产生者是否在已知的公钥拥有者中。如果在, 则将拥有者可信字段中的标志赋值给签字可信字段; 如果不在, 则为签字可信字段赋值 unknown user。

③ 密钥合法性字段的取值是由它的各签字可信字段的取值计算的。如果签字可信字段中至少有一个标志为 ultimate, 则将密钥合法性字段的标志取为 complete。否则将其赋值为各签字可信字段的加权和, 其中签字可信字段标志值为 always trusted 的权取为 $1/X$, 标志值为 usually trusted 的权取为 $1/Y$, X 、 Y 分别是标志为 always trusted 和 usually trusted 的签字的个数。

由于在公钥环中既可插入新的公开钥, 又可插入新的签字, 因此为了保持公钥环中的

一致性,PGP 都将定期地对其从上到下进行处理。对每个拥有者可信字段,PGP 将找出该拥有者的所有签字,并将签字可信字段的值修改为拥有者可信字段中的值。

图 8-15 是一个信任关系与密钥合法性之间关系的示例。图中圆结点表示密钥,圆结点旁边的字母表示密钥的拥有者。图的结构反映了标记为 You 的用户的公钥环。最顶层的结点是用户 You 自己的公开钥,它自然是合法的,且拥有者可信字段的标志为 ultimate,其他结点的拥有者可信字段的标志由用户 You 指定,如果未指定则设置为 undefined,例如用户 D、E、F、L 的密钥(图中黑色结点表示)的拥有者可信字段指定为 always trusts,意指 You 信任这 4 个密钥签署其他密钥,而用户 A、B 的密钥(图中浅黑色结点表示)的拥有者可信字段指定为 partially trusts,意即 You 部分信任这两个密钥签署其他密钥。

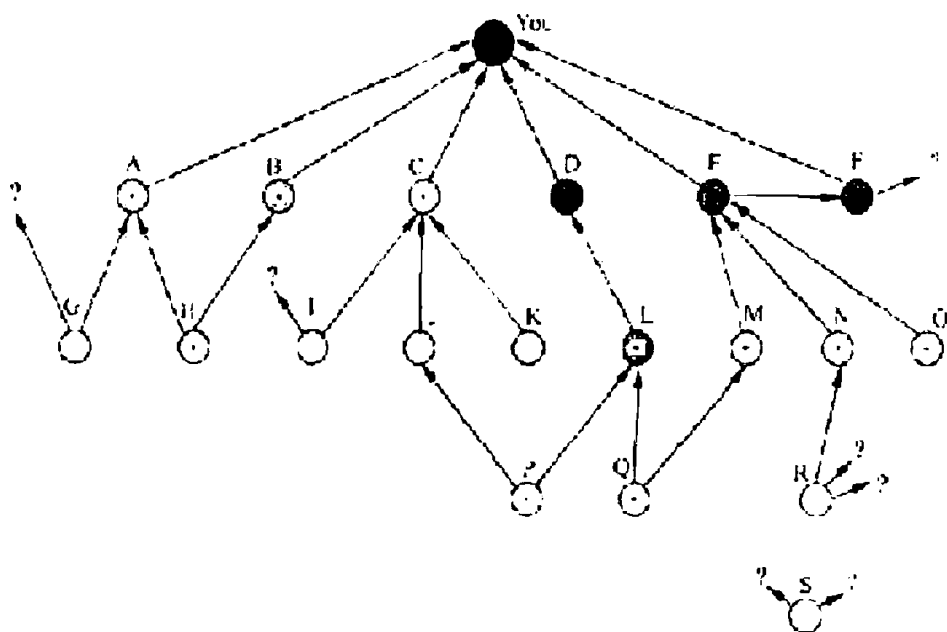


图 8-15 PGP 信任关系示例

图的树结构表示哪个密钥已经被其他用户所签,例如 G 的密钥被用户 A 签署,则用一个由 G 指向 A 的箭头表示。如果一个密钥被一个其密钥不在公钥环中的用户签署,如 R,则用一个由 R 指向一问号的箭头表示,问号表示签字者对用户 You 来说是未知的。

该图可说明以下几个问题:

① 所有被 You 信任的用户(包括完全信任和部分信任)的公开钥都被 You 签署,结点 L 除外。这种签字并不总是有必要的,但实际中大多数用户都愿意为他们信任的用户的公开钥签字,例如 E 的公开钥已被 F 签署,但 You 仍直接为 E 的公开钥签字。

② 两个被部分信任的签字可用于证明其他密钥,例如 A 和 B 被 You 部分信任,则

PGP 认为 A 和 B 同时签署的 H 的公开钥是合法的。

③ 由一个完全可信的或两个部分可信的签字者签署的公开钥被认为是合法的,但这一公开钥的拥有者可能不被信任签署其他公开钥。例如,由于 E 是 You 完全可信的,You 认为 E 为 N 签署的公开钥是合法的,但 N 却不被信任签署其他公开钥。所以 R 的公开钥虽然被 N 签字,但 PGP 却不认为 R 的公开钥是合法的。这种情况具有实际意义,例如,如果想向某一用户发送一保密消息,则不必在各方面都信任这一用户,只要确信这一用户的公开钥是正确的就行了。

④ 图中 S 是一个孤立结点,具有两个未知的签字者。这种公开钥可能是 You 从公钥服务器中得到的。PGP 不能由于这个公钥服务器的信誉好就简单地认为这个公开钥是合法的。

最后需指出的是同一公开钥可能有多个用户 ID,这是因为用户可能改过自己的名(即 ID)或者在多个 ID 名下申请了同一公开钥的签字,例如同一用户可能以自己的多个邮件地址作为自己的 ID 名。所以可将具有多用户 ID 的同一公开钥以树的结构组织起来,其中树根为这个公开钥,根的每一儿子是公开钥在一个 ID 下所获得的签字。对这个公开钥签署其他公开钥的信任程度取决于这一公开钥在不同 ID 下所获得的各个签字。

3. 公钥的吊销

用户如果怀疑与公开钥相应的秘密钥已被泄露,或者仅为避免使用同一密钥对的时间过长,就可吊销自己当前正使用的公开钥。这里所说的泄露指敌手从用户的秘密钥环和通行字短语恢复了用户经加密的秘密钥。吊销公开钥可通过发放一个经自己签字的公开钥吊销证书,证书的格式与前述公钥证书的格式一样,但多了一个标识符,用来表示该证书的目的在于吊销这一公开钥。注意用户签署公钥吊销证书的秘密钥是与被吊销的公开钥相对应的。用户还需尽可能快、尽可能广泛地散发自己的公开钥吊销证书,以使自己的各通信对方都尽可能快地更新自己的公钥环。

敌手如果得到用户的秘密钥也可以发放这种公钥吊销证书,从而使敌手自己和其他用户都不能继续使用这一公开钥。敌手以这种方式使用用户的秘密钥显然要比以其他恶意方式使用用户的秘密钥所带来的危险要小得多。

参考文献

1. 杨波. 网络安全理论与应用. 北京: 电子工业出版社, 2002
2. William Stallings. Cryptography and Network Security: Principles and Practice, 2nd ed. Prentice Hall, 1998
3. 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
4. 卢开澄. 计算机密码学——计算机网络中的数据保密与安全. 第2版. 北京: 清华大学出版社, 1998
5. 朱文余, 孙琦. 计算机密码应用基础. 北京: 科学出版社, 2000
6. 卿斯汉. 密码学与计算机网络安全. 北京: 清华大学出版社, 广西科学技术出版社, 2001
7. 胡予濮, 张玉清, 肖国镇. 对称密码学. 北京: 机械工业出版社, 2002
8. Arto Salomaa. Public-Key Cryptography, 2nd ed. Springer-Verlag, 1996
9. Hans Delfs, Helmut Knehl. Introduction to Cryptography. Springer-Verlag, 2002
10. Joan Danmen, Vincent Rijmen. AES Proposal: Rijndael. AES algorithm submission, September 3, 1999, AES home page: <http://www.nist.gov/aes>