



# 密码学基础

## 理论理解与实践 (Lecture Notes)

作者：李晓峰 (cy\_lxf@163.com)

组织：北京联合大学智慧城市学院

时间：December 14, 2021

版本：3.0

课程主页：<http://uisu.gitee.io/lxf/>



*Victory won't come to us unless we go to it. — M. Moore*

## 前　　言

2018 年学院成立了信息安全专业，同年开始第一次招生，2020 年春季学期开始上“现代密码学”一课，在备课中形成此讲义。内容是在参考已有教材的基础上，按照自己对课程理解组织而成，希望符合本科生的特点。

本科密码学重点应该使学生了解各类密码算法和协议设计的基本思想，能够看懂设计好的密码算法，并能够编程进行实现，同时了解常用的几种密码分析方法。

在编写此讲义的过程中，基本思路，或者预期达到的目的是：

1. 学生能够通过查阅相关基础材料，能够看懂一个密码算法、协议。
2. 学生能够编程实现一个密码算法、协议。算法的实现应该是基本能力要求，所以在授课中特别注意了实验课程的安排，本课程实验让学生通过码云（gitee）提交，主要是想同学们能够熟悉一些常用的工程工具，另外要求基本算法使用 GNU MP 库来实现。
3. 学生了解常用的密码分析方法，并能够编程实现常见的分析算法。
4. 在讲义中有时会直接拷贝原始文献的内容，主要目的是想通过原始文献的内容，让学生更多了解、体会解决问题的思路和过程，而不是只了解结论。

在整理本书的过程中，我将觉得帮助大的资料直接放到本书里，并且给出了链接，这些资料在学习相关概念时帮助很大，有兴趣的同学可从这个点上扩展开来。

还有些资料没有放在正文里面，放在附录，这些资料的目的是希望能够扩展一下阅读，扩展启发一下思路，特别是对英文的一些术语有所了解，因为毕竟很多原始文献是来自于英文，在学习中积累一些专业词汇，有利于大家后面开展深入研究。

由于水平有限，在整理此讲义过程中，难免会出现错误，请大家不吝指出。

为了方便大家在讲课、学习中使用本讲义，本讲义的 tex 文件上传到码云上，希望对大家有所帮助。

关于课程的其他内容编排，可以看我的课程网页<http://uisu.gitee.io/lxf/>。

# 目 录

---

<b>1 引言</b>	<b>2</b>
1.1 信息与数据 (information and data) . . . . .	2
1.1.1 Information vs. Message . . . . .	3
1.1.2 数据 (data) . . . . .	5
1.2 编码 (coding) . . . . .	5
1.3 计算机字符显示分析示例 . . . . .	6
1.3.1 信息的表达 . . . . .	6
1.3.2 字母和数字的计算机内编码 . . . . .	6
1.3.2.1 ASCII 编码表还编码了什么字符 . . . . .	6
1.3.2.2 汉字编码 . . . . .	7
1.3.2.3 如何知道字符编码 . . . . .	7
1.3.3 字符在计算机屏幕上如何显示 . . . . .	7
1.3.3.1 如何解释同一个字符有不同字体（宋体、黑体） . . . . .	7
1.3.4 字体的显示方式如何编码 . . . . .	8
1.3.4.1 直接描述点阵 . . . . .	8
1.3.4.2 描述轮廓 . . . . .	8
1.3.4.3 两种方式的区别 . . . . .	8
1.3.4.4 如何编辑字体 . . . . .	8
1.3.4.5 对于新造的字如何输入 . . . . .	8
1.3.5 字符的输入编码 . . . . .	9
1.3.5.1 汉字输入 . . . . .	9
1.3.6 为什么搞这么多编码 . . . . .	9
1.4 信息安全模型 (Information Security Model) . . . . .	9
1.4.1 保密通信模型 (Secret Communication Model) . . . . .	9
1.5 何为安全 . . . . .	9
1.5.1 安全威胁 (threats) . . . . .	12
1.6 柯克霍夫斯原则 . . . . .	12
1.7 完全保密 (Perfect secrecy) or 香农安全 (Shannon Security) . . . . .	12
1.7.1 Perfect secrecy . . . . .	12
1.7.2 一次一密 (one-time pad) . . . . .	14
1.8 语义安全 (Semantic Security) . . . . .	14
1.8.1 攻击游戏 (attack game) . . . . .	15
1.8.2 语义安全定义 . . . . .	15

1.9 密码中的常识 . . . . .	15
<b>2 密码学研究内容</b>	<b>16</b>
2.1 密码设计 . . . . .	16
2.1.1 密码体制 . . . . .	16
2.1.2 基于计算困难问题的密码体制设计 . . . . .	17
2.1.3 常见的密码困难假设 (Common cryptographic hardness assumptions) . . . . .	19
2.2 密码分析 . . . . .	20
2.3 密码测评 . . . . .	20
<b>3 古典密码 (Classical Encryption)</b>	<b>22</b>
3.1 凯撒密码 Caesar cipher . . . . .	22
3.2 移位变换 (shift transformation)/加法密码 . . . . .	22
3.3 乘法密码 . . . . .	22
3.4 仿射变换 (affine transformation) . . . . .	23
3.5 多表代换密码 . . . . .	23
<b>4 流密码 (Stream Ciphers)</b>	<b>24</b>
4.1 基本概念 . . . . .	25
4.2 密钥的产生 . . . . .	26
4.2.1 线性同余发生器 LCG(linear congruential generator) . . . . .	26
4.2.2 线性反馈移位寄存器 LSFR(Linear Feedback shift register) . . . . .	29
4.2.2.1 有限域上的多项式 . . . . .	30
4.2.2.2 LFSR 的多项式表示 . . . . .	35
4.3 序列的伪随机性 . . . . .	36
4.4 伪随机序列名称的由来 [1] . . . . .	38
4.5 m 序列的破解 . . . . .	38
4.6 非线性序列 . . . . .	43
4.6.1 Geff 发生器 . . . . .	43
4.7 流密码的应用 . . . . .	43
<b>5 分组密码 (Block Ciphers)</b>	<b>45</b>
5.1 Feistel 结构 . . . . .	45
5.2 DES . . . . .	45
5.2.1 初始置换 IP . . . . .	45
5.2.2 轮结构 f 函数 . . . . .	47
5.2.3 密钥生成 . . . . .	49
5.2.3.1 置换选择 1(PC-1) . . . . .	49
5.2.3.2 置换选择 2(PC-2) . . . . .	51
5.2.3.3 密钥左循环移位 . . . . .	51

5.2.4	解密 . . . . .	51
5.3	2DES . . . . .	51
5.4	填充 . . . . .	52
5.4.1	NoPadding . . . . .	52
5.4.2	PKCS5Padding . . . . .	52
5.4.3	ISO 10126 padding . . . . .	52
5.4.4	Zero padding . . . . .	52
5.5	分组密码的运行模式 . . . . .	53
5.5.1	电码本模式 (Electronic Code Book)/ECB 模式 . . . . .	53
5.5.2	密码分组链接模式 (Cipher Block Chain)/CBC 模式 . . . . .	53
5.5.3	密码反馈模式 (Cipher Feedback)/CFB 模式 . . . . .	53
5.5.4	输出反馈模式 (Output Feedback)/OFB 模式 . . . . .	55
5.6	ZUC . . . . .	55
5.7	SM4 . . . . .	56
5.7.1	轮函数 F . . . . .	56
5.7.2	密钥扩展 . . . . .	56
5.7.3	加密过程 . . . . .	57
5.7.4	解密过程 . . . . .	57
6	公钥密码体制 (Public Key Cryptography) . . . . .	58
6.1	陷门单向函数 (trap-door one-way function) . . . . .	58
6.2	MH 方法 [2] . . . . .	59
6.2.1	0-1 背包问题 (0-1 Knapsack Problem) . . . . .	59
6.2.2	MH 方法 . . . . .	61
6.2.2.1	陷门单向函数构造 . . . . .	61
6.2.2.2	加密方法 . . . . .	61
6.2.2.3	签名方法 . . . . .	62
6.2.3	背包问题求解的发展 . . . . .	63
6.3	RSA . . . . .	64
6.3.1	RSA 依赖的困难问题 . . . . .	64
6.3.2	RSA 加密方法 . . . . .	65
6.3.2.1	RSA 解密过程证明 . . . . .	65
6.3.3	RSA 简单示例 . . . . .	66
6.3.3.1	生成公私钥对 . . . . .	66
6.3.3.2	用公钥加密字符串 . . . . .	66
6.3.3.3	用私钥解密密文 . . . . .	67
6.3.3.4	用私钥简单签名字符串 . . . . .	67
6.3.3.5	用公钥验证字符串 . . . . .	67

6.3.4	公私钥对的生成 . . . . .	67
6.4	ECC(Elliptic Curve Cryptography) . . . . .	68
6.4.1	基本概念 . . . . .	68
6.4.2	有限域上的椭圆曲线 . . . . .	70
6.4.3	构造密码算法 . . . . .	73
6.4.3.1	消息到椭圆曲线上的映射 . . . . .	73
6.4.3.2	椭圆曲线上的计算困难问题 . . . . .	73
6.4.3.3	椭圆曲线上的密码算法 . . . . .	73
7	<b>哈希函数 (Hash Function)</b> . . . . .	75
7.1	哈希函数特点 . . . . .	75
7.2	对哈希函数的攻击 . . . . .	75
7.2.1	穷举攻击 . . . . .	75
7.2.2	生日攻击 (birthday attack) . . . . .	75
7.3	MD5(Message-Digest algorithm 5) . . . . .	76
7.3.1	MD5 算法 . . . . .	76
7.3.1.1	消息填充 . . . . .	76
7.3.1.2	附加消息长度 . . . . .	76
7.3.1.3	缓冲区初始化 . . . . .	76
7.3.1.4	分组流水处理 . . . . .	76
7.3.1.5	输出 . . . . .	76
7.3.1.6	MD5 算法表示 . . . . .	76
7.3.2	核心算法 $H_{MD5}$ . . . . .	77
7.3.2.1	$H_{MD5}$ 算法 . . . . .	77
7.3.2.2	四轮运算 R . . . . .	77
7.3.2.3	R 中的 $M_i^n, (n = 1, 2, 3, 4)$ . . . . .	78
7.3.2.4	R 中的 $s_i^n, (n = 1, 2, 3, 4)$ . . . . .	78
7.3.2.5	R 中的 $t_i$ . . . . .	78
7.3.2.6	核心运算 . . . . .	79
7.3.2.7	非线性函数 . . . . .	79
7.4	MD5 安全性 . . . . .	79
7.4.1	History and cryptanalysis . . . . .	80
7.4.2	security . . . . .	81
7.5	SHA . . . . .	82
7.5.1	SHA-1 . . . . .	82
7.5.1.1	Introduction . . . . .	82
7.5.1.2	BIT strings and integers . . . . .	82
7.5.1.3	Operations on Words . . . . .	83

7.5.1.4	Message Padding . . . . .	83
7.5.1.5	Functions Used . . . . .	84
7.5.1.6	Constants Used . . . . .	85
7.5.1.7	Computing the Message digest . . . . .	85
7.5.1.8	Alternate method of computation . . . . .	85
7.5.2	SHA-1 的破解 . . . . .	86
7.5.2.1	王小云院士的十年一剑 . . . . .	86
7.5.2.2	进一步崩盘 . . . . .	88
<b>8</b>	<b>消息认证码 (Message Authentication Code)</b>	<b>89</b>
8.1	MAC 定义 . . . . .	89
8.2	MAC 函数要求 . . . . .	89
8.3	基于 DES 的 MAC . . . . .	89
8.4	基于哈希函数的 MAC(HMAC) . . . . .	90
8.4.1	HMAC 的安全目标 . . . . .	90
8.4.2	HMAC 方法 . . . . .	91
8.4.2.1	认证密钥 . . . . .	91
8.4.2.2	HMAC 计算过程 . . . . .	91
<b>9</b>	<b>数字签名 (Digital Signatures)</b>	<b>92</b>
9.1	基于对称密钥算法的数字签名 . . . . .	92
9.2	数字签名的基本概念示例 . . . . .	93
9.3	证书实例 . . . . .	96
9.3.1	PEM, DER, CRT, and CER: X.509 Encodings and Conversions . . . . .	97
9.3.1.1	What is OpenSSL? . . . . .	97
9.3.1.2	PEM . . . . .	97
9.3.1.3	DER . . . . .	99
9.3.1.4	Convert PEM certificate with chain of trust to PKCS#7 . . . . .	102
9.3.1.5	Convert PEM certificate with chain of trust and private key to PKCS#12 . . . . .	102
9.3.1.6	Convert DER-encoded certificate with chain of trust and private key to PKCS#12 . . . . .	103
9.3.2	证书服务公司 . . . . .	103
9.4	Https 实例 . . . . .	104
9.5	Https 标准 . . . . .	106
9.6	签名方案与标准 . . . . .	107
9.6.1	DSS . . . . .	107
9.6.1.1	DSA . . . . .	108
9.6.1.2	RSA ds algorithm . . . . .	110

<b>10 密钥管理 (Key Management)</b>	<b>111</b>
10.1 密钥分发 (key distribution) 的基本方法 . . . . .	115
10.1.1 有中心的密钥分发 . . . . .	115
10.1.2 无中心的密钥分发 . . . . .	115
10.2 Diffie-Hellman 密钥交换 . . . . .	117
10.2.1 DH 中间人攻击 . . . . .	117
10.3 密钥分割 . . . . .	118
10.3.1 Shamir 门限方案 . . . . .	118
10.3.1.1 拉格朗日插值法 (Lagrange Interpolation Polynomial ) . . . . .	118
10.3.1.2 门限方案 . . . . .	119
10.3.2 CRT 门限方案 . . . . .	119
10.3.2.1 中国剩余定理 [3] . . . . .	119
10.3.2.2 门限方案 . . . . .	119
<b>11 安全服务和安全机制 (Security Services &amp; Mechanisms)</b>	<b>121</b>
11.1 密码学支撑的安全服务 (Security Services) . . . . .	121
11.2 密码算法 (Cryptographic Algorithm) . . . . .	122
11.3 OSI 7 层模型与安全服务与机制 . . . . .	122
11.3.1 安全服务 (Security Services) . . . . .	123
11.3.2 特定的安全机制 (Specific security mechanisms) . . . . .	123
11.3.3 普遍安全机制 (Pervasive security mechanisms) . . . . .	124
11.3.4 安全服务与安全机制间关系的实例 (Illustration of relationship of security services and mechanisms) . . . . .	124
11.3.5 安全服务与层的关系的实例 (Illustration of the relationship of security services and layers) . . . . .	124
<b>12 协议 (Protocols)</b>	<b>126</b>
12.1 零知识协议 (Zero-knowledge Protocol) . . . . .	126
12.1.1 零知识协议的形象例子 [2] . . . . .	127
12.2 认证协议 (Authentication protocol) . . . . .	127
12.2.1 Needham-Schroeder . . . . .	127
12.2.1.1 基于对称密钥体制的 Needham-Schroeder(Needham-Schroeder Symmetric Key Protocol) . . . . .	128
12.2.1.2 基于公钥密码体制的 Needham-Schroeder(Needham-Schroeder Public-Key Protocol) . . . . .	129
12.2.1.3 安全性分析 . . . . .	129
12.3 智力扑克 . . . . .	130
12.3.1 SRA 方案 [2] . . . . .	131
12.4 kerberos 协议 & 系统 . . . . .	132



12.5 Windows Logon . . . . .	133
12.6 TLS/SSL . . . . .	134
12.7 IPv6 . . . . .	134
12.8 5G AKA . . . . .	134
<b>13 协议的安全分析</b>	<b>136</b>
13.1 BAN 逻辑 [2] . . . . .	136
13.1.1 基本概念 . . . . .	136
13.1.2 逻辑公设 . . . . .	136
13.1.3 推理步骤 . . . . .	137
13.1.4 分析 Needham-Schroeder 协议 . . . . .	138
13.1.4.1 初始假设集合 . . . . .	138
13.1.4.2 理想化协议模型 . . . . .	138
13.1.4.3 建立协议预期目标 . . . . .	139
13.1.4.4 利用初设和逻辑公设推理 . . . . .	139
13.1.4.5 推导出协议最终目标集合 . . . . .	141
13.1.4.6 讨论 . . . . .	141
13.2 逻辑编程和协议分析工具 . . . . .	141
<b>14 安全多方计算 (Secure Multi-Party Computation)</b>	<b>143</b>
14.1 姚的百万富翁方案 . . . . .	145
14.2 同态加密 . . . . .	146
14.3 开源项目 . . . . .	147
<b>15 比特币 (Bitcoin) &amp; 区块链 (Block Chains)</b>	<b>148</b>
15.1 比特币起源 . . . . .	148
15.2 技术发展脉络 . . . . .	148
15.2.1 账本 . . . . .	149
15.2.2 交易 (Transaction) . . . . .	149
15.2.3 拜占庭协议 . . . . .	150
15.2.3.1 拜占庭将军问题 (The Byzantine Generals Problem) . . . . .	150
15.2.3.2 拜占庭共识算法之 PBFT . . . . .	155
15.2.4 挖矿原理 . . . . .	155
15.2.5 钱包 . . . . .	157
15.3 比特币 . . . . .	157
15.4 其他数字货币或虚拟货币 . . . . .	158
15.5 区块链 (block chains) . . . . .	158
15.5.1 区块链的几种类型 . . . . .	160
15.6 开源系统 . . . . .	161

<b>16 可信计算 (Trusted Computing)</b>	<b>163</b>
16.1 可信平台模块 (TPM:Trusted Platform Module) . . . . .	164
16.2 相关资源 . . . . .	168
<b>17 扩展阅读</b>	<b>171</b>
17.1 姚期智——图灵奖的介绍 . . . . .	171
17.2 2021 年理论计算科学和离散数学领域学者获 Abel 奖 . . . . .	175
17.3 ZUC . . . . .	176
17.3.1 密钥装载 . . . . .	177
17.3.2 初始化阶段 . . . . .	177
17.3.3 工作阶段 . . . . .	177
17.3.4 F 函数 . . . . .	178
17.3.5 ZUC 设计理念及准则 . . . . .	178
17.3.5.1 设计理念 . . . . .	178
17.3.5.2 设计准则 . . . . .	179
17.4 finite field . . . . .	179
17.5 One way function . . . . .	180
17.6 enigma . . . . .	181
17.7 Dan Boneh,Cryptography I . . . . .	181
17.8 各种安全模型 . . . . .	181
17.8.1 Bell-LaPadula model . . . . .	182
17.8.2 ISO 安全模型 . . . . .	184
17.8.3 IATF 安全模型 (ISSE) . . . . .	187
17.8.4 BS7799 PDCA 模型 . . . . .	187
17.9 课程设计基本概念 . . . . .	187
17.9.1 课程设计的目的 . . . . .	187
17.9.2 课程设计的类型 . . . . .	187
17.9.2.1 学科为中心的课程设计 (subject-centered curriculum design) .	187
17.9.2.2 学习者为中心的课程设计 (learner-centered curriculum de-sign) .	187
17.9.2.3 问题为中心的课程设计 (problem-centered curriculum design) .	188
17.9.2.4 课程设计时的几点注意事项 . . . . .	188
17.10 The Vernam Cipher . . . . .	188
17.11 工具 RSA-Tool 2 中的公私钥生成方法 . . . . .	188
<b>A 练习</b>	<b>194</b>
<b>B 课程参考分值分配</b>	<b>195</b>



<b>C 相关机构</b>	<b>196</b>
C.1 国家密码管理局 . . . . .	196
C.2 密码行业标准化技术委员会 . . . . .	196
C.3 商用密码检测中心 . . . . .	197
C.4 中国密码学会 . . . . .	198
<b>D 相关法律、条例、标准</b>	<b>200</b>
D.1 基本概念 . . . . .	200
D.2 中华人民共和国网络安全法 . . . . .	200
D.3 中华人民共和国密码法 . . . . .	201
D.4 中华人民共和国反恐怖主义法 . . . . .	202
D.5 中华人民共和国电子签名法 . . . . .	202
D.6 国家商用密码管理条例 . . . . .	203
D.7 《信息安全技术信息系统密码应用基本要求》国家标准 . . . . .	204
<b>E 代数基础 (Algrbra)</b>	<b>205</b>
E.1 群、环、域 . . . . .	205
E.2 有限域的结构 . . . . .	205
E.2.1 有素数元素的有限域 . . . . .	205
E.2.2 有限域模不可约多项式 . . . . .	206
E.2.2.1 一个代数机构的多项式 . . . . .	206
E.2.2.2 用不可约多项式构造域 . . . . .	206
E.2.3 用多项式基构造有限域 . . . . .	206





# 第1章 引言

## 1.1 信息与数据 (information and data)

“信息”一词在英文、法文、德文、西班牙文中均是“information”，日文中为“情报”，我国台湾称之为“资讯”，我国古代用的是“消息”。作为科学术语最早出现在哈特莱（R.V.Hartley）于1928年撰写的《信息传输》一文中。20世纪40年代，信息的奠基人香农（C.E.Shannon）<sup>1</sup>给出了信息的明确定义，此后许多研究者从各自的研究领域出发，给出了不同的定义。具有代表意义的表述如下<sup>2</sup>：

- 信息奠基人香农（Shannon）认为“信息是用来消除随机不确定性的东西”，这一定义被人们看作是经典性定义并加以引用。
- 控制论创始人维纳（Norbert Wiener）认为“信息是人们在适应外部世界，并使这种适应反作用于外部世界的过程中，同外部世界进行互相交换的内容和名称”，它也被作为经典性定义加以引用。
- 经济管理学家认为“信息是提供决策的有效数据”。
- 美国著名物理化学家吉布斯（Josiah Willard Gibbs）创立了向量分析并将其引入数学物理中，使事件的不确定性和偶然性研究找到了一个全新的角度，从而使人类在科学把握信息的意义上迈出了第一步。他认为“熵”是一个关于物理系统信息不足的量度。
- 电子学家、计算机科学家认为“信息是电子线路中传输的信号”。
- 我国著名的信学专家钟义信教授认为“信息是事物存在方式或运动状态，以这种方式或状态直接或间接的表述”。
- 美国信息管理专家霍顿（F.W.Horton）给信息下的定义是：“信息是为了满足用户决策的需要而经过加工处理的数据。”简单地说，信息是经过加工的数据，或者说，信息是数据处理的结果。

根据对信息的研究成果。科学的信息概念可以概括为，信息是对客观世界中各种事物的运动状态和变化的反映，是客观事物之间相互联系和相互作用的表征，表现的是客观事物运动状态和变化的实质内容。

<sup>1</sup>香农被称为信息论的创始人，1938年香农在MIT获得电气工程硕士学位，硕士论文题目是《A Symbolic Analysis of Relay and Switching Circuits》(继电器与开关电路的符号分析)。当时他已经注意到电话交换电路与布尔代数之间的类似性，即把布尔代数的“真”与“假”和电路系统的“开”与“关”对应起来，并用1和0表示。于是他用布尔代数分析并优化开关电路，这就奠定了数字电路的理论基础。在1948年6月和10月在《贝尔系统技术杂志》(Bell System Technical Journal)上连载发表了具有深远影响的论文《A Mathematical Theory of Communication》(《通讯的数学原理》)。1949年，香农又在该杂志上发表了另一著名论文《Communication in the Presence of Noise》(《噪声下的通信》)。在这两篇论文中，香农阐明了通信的基本问题，给出了通信系统的模型，提出了信息量的数学表达式，并解决了信道容量、信源统计特性、信源编码、信道编码等一系列基本技术问题。两篇论文成为了信息论的奠基性著作。1949年香农发表了另外一篇重要论文《Communication Theory of Secrecy Systems》(保密系统的通信理论)，它的意义是使保密通信由艺术变成科学。

<sup>2</sup>本部分信息来自于<https://baike.baidu.com/item/>信息

现在让我们思考一个问题，假设你看到了以下景观 (如图1.1)。



图 1.1: 看到的景观

你看到了什么？

你想到了什么？

你想告诉大家什么？(你想发送的信息)

你怎么告诉大家？(编码和传输)

别人收到了什么？(接受编码)

别人得到了什么？(解码)

别人想到了什么？

你面前的事物在你感受的同时，就已经片面化了，这种感受进入你的意识里，就已经被你意识编码了，再把这种意识转化为想告诉大家的事，成为想法，然后告诉大家想法用语言进行编码，把语言通过语音传给对方，比如用普通话，对方接收、识别，然后进行反向操作，一直恢复到“想法”层（知道了对方的这个想法），到这一层，大多数时候没有问题，再往上恢复其实就很难了。

### 1.1.1 Information vs. Message

在 Oxford 网络词典中，information 的解释为：

information:

1. facts provided or learned about something or someone.
2. what is conveyed or represented by a particular arrangement or sequence of things.

在 Merriam-Webster 网络词典中，information 解释为：

information:

1. the communication or reception of knowledge or intelligence
- 2.
- a
  - (1) : knowledge obtained from investigation, study, or instruction
  - (2) : intelligence, news
  - (3) : facts, data
- b: the attribute inherent in and communicated by one of two or more alternative sequences or arrangements of something (such as nucleotides in DNA or binary digits in a computer program) that produce specific effects
- c
  - (1) : a signal or character (as in a communication system or computer) representing data
  - (2) : something (such as a message, experimental data, or a picture) which justifies change in a construct (such as a plan or theory) that represents physical or mental experience or another construct
- d: a quantitative measure of the content of information  
specifically : a numerical quantity that measures the uncertainty in the outcome of an experiment to be performed
- 3: the act of informing against a person
- 4: a formal accusation of a crime made by a prosecuting officer as distinguished from an indictment presented by a grand jury

在 Oxford 网络词典中，message 的解释为：

message:

1. a verbal, written, or recorded communication sent to or left for a recipient who cannot be contacted directly.
2. a significant point or central theme, especially one that has political, social, or moral importance.

在 Merriam-Webster 网络词典中，message 解释为：

message:

1: a communication in writing, in speech, or by signals

Please take this message for me to my friend.

2: a messenger's mission

the girl will go on a message to the shop

— Cahir Healy

3: an underlying theme or idea

the message is that it is time to change

— The Economist

我们可以从字典里的解释可以看出, information 侧重于内在, 侧重于实质 “facts, knowledge or intelligence”, message 侧重于特定 (in writing, in speech, or by signals) 的通信 (communication)。

### 1.1.2 数据 (data)

”数据 (data) 是事实或观察的结果, 是对客观事物的逻辑归纳, 是用于表示客观事物的未经加工的原始素材。数据可以是连续的值, 比如声音、图像, 称为模拟数据。也可以是离散的, 如符号、文字, 称为数字数据。在计算机系统中, 数据以二进制信息单元 0,1 的形式表示。数据是信息的表现形式和载体, 而信息是数据的内涵, 信息是加载于数据之上, 对数据作具有含义的解释。”

数据可以承载信息, 但不是信息的唯一载体。

## 1.2 编码 (coding)

“编码理论是数学和计算机科学的一个分支, 处理在噪声信道传送资料时的错误倾向。按照编码理论, 资料传送时会采用更好的方法以修正传送途中所产生的大量错误。编码共分两类: 信源编码 (数据压缩)、信道编码 (前向纠错) <sup>3</sup>。” <sup>4</sup>

<sup>5</sup>Coding theory is the study of the properties of codes and their respective fitness for specific applications. Codes are used for data compression, cryptography, error detection and correction, data transmission and data storage. Codes are studied by various scientific disciplines—such as information theory, electrical engineering, mathematics, linguistics, and computer science—for the purpose of designing efficient and reliable data transmission methods. This typically involves the removal of redundancy and the correction or detection of errors in the transmitted data.

There are four types of coding:

- Data compression (or, source coding)
- Error correction (or channel coding)
- Cryptographic coding
- Line coding

Data compression attempts to compress the data from a source in order to transmit it more efficiently. For example, Zip data compression makes data files smaller to reduce Internet traffic. Data compression and error correction may be studied in combination.

Error correction adds extra data bits to make the transmission of data more robust to disturbances present on the transmission channel. The ordinary user may not be aware of many applications using error correction. A typical music CD uses the Reed-Solomon code to correct

<sup>2</sup><https://baike.baidu.com/item/> 数据

<sup>3</sup>目前还有一种分法, 加了一种保密编码, 为了得到安全特性而采用的编码方法。

<sup>4</sup><https://baike.baidu.com/item/> 编码理论

<sup>5</sup><https://encyclopedia.thefreedictionary.com/coding+theory>

for scratches and dust. In this application the transmission channel is the CD itself. Cell phones also use coding techniques to correct for the fading and noise of high frequency radio transmission. Data modems, telephone transmissions, and NASA all employ channel coding techniques to get the bits through, for example the turbo code and LDPC codes.

Cryptography or cryptographic coding is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). More generally, it is about constructing and analyzing protocols that block adversaries; various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, and electrical engineering.

A line code (also called digital baseband modulation or digital baseband transmission method) is a code chosen for use within a communications system for baseband transmission purposes. Line coding is often used for digital data transport.

Line coding consists of representing the digital signal to be transported by an amplitude-and time-discrete signal that is optimally tuned for the specific properties of the physical channel (and of the receiving equipment). The waveform pattern of voltage or current used to represent the 1s and 0s of a digital data on a transmission link is called line encoding. The common types of line encoding are unipolar, polar, bipolar, and Manchester encoding.

## 1.3 计算机字符显示分析示例

### 1.3.1 信息的表达

语言是人们进行信息表达的一种主要方式，或者说语言是人类对于信息的一种编码方式，比如英文，汉语。除了语言，人类还有其他信息编码方式，比如绘画和音乐也应该是一种信息的编码方式，但是这样的编码方式更多或者说更加适合感觉、情感的编码。

### 1.3.2 字母和数字的计算机内编码

你所知道的 26 个英文字母和 0 9 的数字在计算机中的的编码方式是什么？很多人都会想到 ASCII 码。

ASCI 码的定义如图1.2。

#### 1.3.2.1 ASCII 编码表还编码了什么字符

ASCII 码除了对字母和数字进行了编码外，还对 SPACE、ESC、Enter 等等控制字符和部分格式字符进行的编码。

ASCII 第一次以规范标准的型态发表是在 1967 年，最后一次更新则是在 1986 年，至今为止共定义了 128 个字符，其中 33 个字符无法显示（这里的无法显示，是指在现代操作系统 Windows 下无法显示，但在 DOS 模式下可显示出一些诸如笑脸、扑克牌花式等

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;		96	60	140	&#96;	`
1	1 001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2 002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3 003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4 004	ETB	(end of transmission)	36	24	044	&#36;	\$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5 005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6 006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7 007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8 010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9 011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A 012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B 013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C 014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D 015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E 016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F 017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10 020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11 021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12 022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13 023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14 024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15 025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16 026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17 027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18 030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19 031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A 032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B 033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	{	123	7B	173	&#123;	{
28	1C 034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	\
29	1D 035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E 036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F 037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

图 1.2: ASCII 码定义表<sup>6</sup>

8-bit 符号), 且这 33 个字符多数都已是陈废的控制字符, 控制字符的用途主要是用来操控已经处理过的文字, 在 33 个字符之外的是 95 个可显示的字符, 包含用键盘敲下空白键所产生的空白字符也算 1 个可显示字符 (显示为空白)。

### 1.3.2.2 汉字编码

当计算机来到中国后, 我们要想形成本土应用, 首先一个问题就是解决汉字在计算机中的编码问题, 现在常用的编码模式为 Unicode。

### 1.3.2.3 如何知道字符编码

查看字符 Unicode 编码, 有很多第三方工具, 微软也有工具, 如 C:\Windows\System32\charmap.exe 用这个工具查两种字体“等线”和“仿宋”中的“尺”, 可以看到其 Unicode 编码显示都是: U+5C3A (0xB3DF)

### 1.3.3 字符在计算机屏幕上如何显示

计算机屏幕是由一个个点组成的, 那么最直观的想法是, 每个字符显示时候都对应一个点阵, 提前定义好每个字符的点阵, 就定义了字符在屏幕上的显示方式。

#### 1.3.3.1 如何解释同一个字符有不同字体 (宋体、黑体)

同一个字符有不同分格的书写方式, 我们只要按照不同的书写风格定义出不同的点阵就可以实现, 我一类书写风格的字符我们称为字体, 我们把所有字符定义的某种显示方式统称为字库。

### 1.3.4 字体的显示方式如何编码

#### 1.3.4.1 直接描述点阵

在单片机控制的液晶显示面板上，通常使用这种编码方式，比如 8\*8 的点阵就是 64 个点。对于小的液晶面板通常会有一个显示控制板，这个控制板有个重要内容就是有字库。

**1.3.4.1.1 点阵的数据如何存储** 大家想想，对于点阵来说，点亮的点比较而言是稀疏的，如何用最小的空间存储。

#### 1.3.4.2 描述轮廓

王选先生当时在计算机发展初期，其做的一项重要内容就是汉字的轮廓描述。

我们常见的 True type font 就是一直轮廓描述字体。

#### 1.3.4.3 两种方式的区别

想想字体放大缩小这个问题，我们就可以清楚看到直接点阵编码和轮廓编码的区别，轮廓编码原则上可以无限放大，而不影响显示效果，而直接点阵编码，直接放大会出现马赛克，要想获得放大后好的显示效果，需要进行显示前预处理。

#### 1.3.4.4 如何编辑字体

你可以通过字体编码工具，定义一个新的字符（用未占有的编码号），或者定义一个已有字符的新的显示方式。

**1.3.4.4.1 微软自带的工具** 微软 Windows 操作系统中有自带的字体编辑工具，C:\Windows\System32\edcedit.exe，可以在命令上中执行此命令，打开了解此工具。

**1.3.4.4.2 第三方工具** Fontforge 原来是 sourceforge 开源网站上的项目，目前有自己的主页<https://fontforge.org/en-US/>，源代码托管在 github 上，链接为<https://github.com/fontforge>。这个开源项目一直处于活跃状态，功能对标商业软件，目前被广泛使用。

#### 1.3.4.5 对于新造的字如何输入

这就是下面要讲的一个问题，输入编码问题。



### 1.3.5 字符的输入编码

#### 1.3.5.1 汉字输入

汉字输入，我么常用的有拼音、五笔等输入法，如何“简单”（这里“简单”的含义是降低学习成本）、快速、准确的输入是输入编码需要解决的问题。

### 1.3.6 为什么搞这么多编码

在解决问题时，往往问题是发生在一个受限环境中（constrained），问题的解决办法也受这些限制条件的制约，我们就是要在这样的受限条件下寻找一个合适的解法。

## 1.4 信息安全模型 (Information Security Model)

### 1.4.1 保密通信模型 (Secret Communication Model)

香农在其 1949 年公开发表的一篇经典论文“Communication theory of secrecy system”系统的论述了对加密系统的理解，图 1.3 是一个系统框图，后来做为保密通信的一般模型被大家广泛引用。

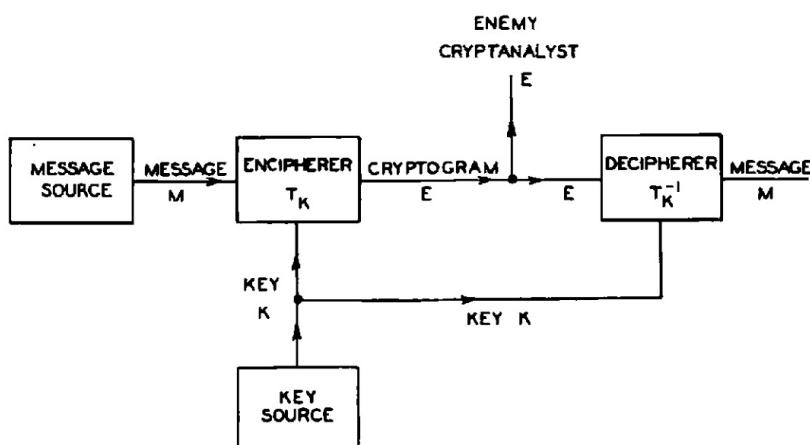


Fig. 1—Schematic of a general secrecy system.

图 1.3: Shannon 在 Communication theory of secrecy system

## 1.5 何为安全

我们首先看看在 Merriam-Webster 网络词典中，security 和 secure 的解释：

#### Definition of security

- 1 : the quality or state of being secure: such as
- a : freedom from danger : SAFETY
- b : freedom from fear or anxiety
- c : freedom from the prospect of being laid off.  
job security

Definition of secure (Entry 1 of 2)

a : free from danger  
b : affording safety // a secure hideaway  
c : TRUSTWORTHY, DEPENDABLE // a secure foundation  
d : free from risk of loss

可以看出安全是一种状态，是一种“感觉”，这种对“安全的感觉”在不同时期有变化，在不同情境下也有所不同。

在[4]一书中将安全的这种感觉，我们也称为安全的属性，概括为：机密性、完整性、非否认、可认证、访问控制。

在 WIKI 的 information security 词条解释中<sup>7</sup>，有关于安全属性理解的发展描述：

\*\*\*\*\*

The CIA triad of confidentiality, integrity, and availability is at the heart of information security.[14] (The members of the classic InfoSec triad—confidentiality, integrity and availability—are interchangeably referred to in the literature as security attributes, properties, security goals, fundamental aspects, information criteria, critical information characteristics and basic building blocks.) There is continuous debate about extending this classic trio.[5] Other principles such as Accountability[15] have sometimes been proposed for addition—it has been pointed out that issues such as non-repudiation do not fit well within the three core concepts.

In 1992 and revised in 2002, the OECD's Guidelines for the Security of Information Systems and Networks[16] proposed the nine generally accepted principles: awareness, responsibility, response, ethics, democracy, risk assessment, security design and implementation, security management, and reassessment. Building upon those, in 2004 the NIST's Engineering Principles for Information Technology Security[17] proposed 33 principles. From each of these derived guidelines and practices.

In 2002, Donn Parker proposed an alternative model for the classic CIA triad that he called the six atomic elements of information. The elements are confidentiality, possession, integrity, authenticity, availability, and utility. The merits of the Parkerian Hexad are a subject of debate amongst security professionals.

In 2011, The Open Group published the information security management standard O-ISM3. This standard proposed an operational definition of the key concepts of security, with elements called "security objectives", related to access control (9), availability (3), data quality (1), compliance and technical (4). This model is not currently widely adopted.

In 2013, based on a thorough analysis of Information Assurance and Security (IAS) literature, the IAS-octave was proposed as an extension of the CIA-triad.[18] The IAS-octave includes Confidentiality, Integrity, Availability, Accountability, Auditability, Authenticity/Trustworthiness, Non-repudiation and Privacy. The completeness and accuracy of the IAS-octave was

<sup>7</sup><https://encyclopedia.thefreedictionary.com/Information+security>

evaluated via a series of interviews with IAS academics and experts. The IAS-octave is one of the dimensions of a Reference Model of Information Assurance and Security (RMIAS), which summarizes the IAS knowledge in one all-encompassing model.

The CIA triad is infrequently criticised, but there are some authors who have objections

### **Confidentiality**

In information security, confidentiality "is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes" (Excerpt ISO27000).

### **Integrity**

In information security, data integrity means maintaining and assuring the accuracy and completeness of data over its entire life-cycle.[19] This means that data cannot be modified in an unauthorized or undetected manner. This is not the same thing as referential integrity in databases, although it can be viewed as a special case of consistency as understood in the classic ACID model of transaction processing. Information security systems typically provide message integrity in addition to data confidentiality.

### **Availability**

For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. High availability systems aim to remain available at all times, preventing service disruptions due to power outages, hardware failures, and system upgrades. Ensuring availability also involves preventing denial-of-service attacks, such as a flood of incoming messages to the target system essentially forcing it to shut down.[20]

### **Non-repudiation**

In law, non-repudiation implies one's intention to fulfill their obligations to a contract. It also implies that one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction. Note: This is also regarded as part of Integrity.

It is important to note that while technology such as cryptographic systems can assist in non-repudiation efforts, the concept is at its core a legal concept transcending the realm of technology. It is not, for instance, sufficient to show that the message matches a digital signature signed with the sender's private key, and thus only the sender could have sent the message and nobody else could have altered it in transit. The alleged sender could in return demonstrate that the digital signature algorithm is vulnerable or flawed, or allege or prove that his signing key has been compromised. The fault for these violations may or may not lie with the sender himself, and such assertions may or may not relieve the sender of liability, but the assertion would invalidate the claim that the signature necessarily proves authenticity and integrity and thus prevents repudiation.

\*\*\*\*\*



### 1.5.1 安全威胁 (threats)

安全威胁是对安全目标的破坏，大类来说有：

- 信息泄露
- 完整性破坏
- 拒绝服务
- 非法使用

具体来说有 [4]：

- 假冒
- 旁路控制
- 授权侵犯（内部威胁）
- 特洛伊木马
- 陷门
- 窃听
- 业务流分析
- 人员疏忽
- 媒体清理（从废弃的媒体获得信息，比如废硬盘，光盘，打印纸等）

## 1.6 柯克霍夫斯原则

现代密码系统的基本设计原则是一个德语教授在其《军事密码学》<sup>8</sup>一书中提出，他指出“密码系统的安全性应该仅仅取决于所使用的密钥的机密性，而不是对该方案本身的保密”[5]，这也就是现在被大家广泛接受的“柯克霍夫斯原则”。

 **注意** 柯克霍夫斯是巴黎大学的德语教授，出生后起名为“Jean-Guillaume-Huber-Victor-Francois-Alexandre-Auguste kerckhoffs von Nieuwenhof”，后将其名字缩短为“Auguste kerckhoffs”，他的职业生涯大部分在教英语和德语，偶尔也教意大利语、拉丁语、希腊语、历史学和数学。他出版的书包括语法学、德国喜剧起源、艺术与宗教的关系，当然也包括密码学。[5]

柯克霍夫斯原则的确立也是有个过程。20世纪70年代，DES的公开，实践证明是安全的。80年代EES，采用内部设计，只提供芯片，不公开算法，实践证明是不安全的。90年代，AES世界范围公开征集、评价，最后证明确定的方案是安全的。

## 1.7 完全保密 (Perfect secrecy) or 香农安全 (Shannon Security)

### 1.7.1 Perfect secrecy

完全保密系统，通俗来讲，就是当我们获得密文后，对于我们了解明文，没有任何帮助。换句话说，消息  $M$  的概率为  $P(M)$ ，在知道密文  $E$  的情况下，消息  $M$  的概率为

<sup>8</sup>在 [2] 一书中，提到此原则提出是 1883 年，而在 [5] 一书中，提到此原则提出是 1881 年

$P_E(M)$ , 一个完全保密系统就是说对于所有可能的 E 和 M, 有  $P(M) = P_E(M)$ , 这个公式的含义也可以说成, 消息 M 的概率与在知道密文 E 后消息 M 的概率一样。

我们看看香农在其著名的文章 [6] 中给出的数学定义。

Let us suppose the possible messages are finite in number  $M_1, \dots, M_n$  and have a **priori** probabilities  $P(M_1), \dots, P(M_n)$ , and that these are enciphered into the possible cryptograms  $E_1, \dots, E_m$  by

$$E = T_i M$$

The cryptanalyst intercepts a particular E and can then calculate, in principle at least, the **a posteriori** probabilities for the various messages,  $P_E(M)$ . It is natural to define perfect secrecy by the condition that, for all E the **a posteriori** probabilities are equal to the **a priori** probabilities independently of the values of these. In this case, intercepting the message has given the cryptanalyst no information. Any action of his which depends on the information contained in the cryptogram cannot be altered, for all of his probabilities as to what the cryptogram contains remain unchanged. On the other hand, if the condition is not satisfied there will exist situations in which the enemy has certain **priori** probabilities, and certain key and message choices may occur for which the enemy's probabilities do change. This in turn may affect his actions and thus perfect secrecy has not been obtained. Hence the definition given is necessarily required by our intuitive ideas of what perfect secrecy should mean.

A necessary and sufficient condition for perfect secrecy can be found as follows: We have by Bayes' theorem <sup>9</sup>

$$P_E(M) = \frac{P(M)P_M(E)}{P(E)}$$

in which:

- $P(M)$ :a priori probability of message M.
- $P_M(E)$ :conditional probability of cryptogram E if message M is chosen, i.e. the sum of the probabilities of all keys which produce cryptogram E from message M.
- $P(E)$ :probability of obtaining cryptogram E from any cause.
- $P_E(M)$ :a posteriori probability of message M if cryptogram E is intercepted.

For perfect secrecy  $P_E(M)$  must equal  $P(M)$  for all E and all M. Hence either  $P(M) = 0$ , a solution that must be excluded since we demand the equality independent of the values of  $P(M)$ , or

$$P_M(E) = P(E)$$

for every M and E. Conversely if  $P_M(E) = P(E)$  then

$$P_E(M) = P(M)$$

---

<sup>9</sup>Baye's 定理在很多课本里的的一般式为  $P(B_i|A) = \frac{P(B_i)P(A|B_i)}{\sum_{j=1}^n P(B_j)P(A|B_j)}$ , 在香农的论文中, 其条件概率的写法与我们目前的课本略有不同

and we have perfect secrecy. Thus we have the result:

Theorem6: A necessary and sufficient condition for perfect secrecy is that

$$P_M(E) = P(E)$$

for all M and E. That is,  $P_M(E)$  must be independent of M.

Stated another way, the total probability of all keys that transform  $M_i$  into a given cryptogram E is equal to that of all keys transforming  $M_j$  into the same E, for all  $M_i, M_j$  and E.

Now there must be as many E's as there are M's since, for a fixed i,  $T_i$  gives a one-to-one correspondence between all the M's and some of the E's. For perfect secrecy  $P_M(E) = P(E) \neq 0$  for any of these E's and any M. Hence there is at least one key transforming any M into any of these E's. But all the keys from a fixed M to different E's must be different, and therefore *the number of different keys is at least as great as the number of M's*.

### 1.7.2 一次一密 (one-time pad)

一次一密的重要特征是其密钥是一个随机序列，密钥只使用一次，且密钥的长度等于明文序列的长度。一次一密理论上不可攻破的密码系统。也就是说一次一密系统是一种完全保密系统，或者说是完全保密系统的一个一类实现。在历史上 AT&T 曾经实现过一个一次一密的加密机 Vernam，在本资料的扩展阅读部分有此加密机的信息线索，感兴趣的同學可以阅读。

## 1.8 语义安全 (Semantic Security)

设  $\Pi = (E, D)$  是定义在  $(K, M, C)$  上的香农密码，随机变量  $\kappa$  在  $K$  上均匀分布， $\phi$  是 C 上的所有谓词<sup>10</sup>，为了使得概率表达更明显，我们使用  $\Pr[]$  表示概率。

对于任意两个明文  $m_0, m_1$ ，如果  $\Pr[\phi(E(\kappa, m_0))] = \Pr[\phi(E(\kappa, m_1))]$ ，我们称  $\Pi$  是完全安全的密码系统 (或者称为香农安全的密码系统)，也就是说消息  $m_0, m_1$  加密后，概率意义上不可区分。

如果 E 是一个概率算法，用  $C \xleftarrow{R} E(\kappa, m)$  表示执行  $E(\kappa, m)$  输出一个随机变量 c. 由于 E 是加密算法，所以虽然其是一个概率算法，但其还是应该满足加密算法的基本要求，那就是可逆，也就是当 E、D 是概率算法时，其应该满足的要求是：

$$c \xleftarrow{R} E(\kappa, m), m' \xleftarrow{R} D(\kappa, c), m = m' \text{ 的概率为 } 1.$$

目前我们把加解密函数进行了扩展，所以密码体系就分成了两种，如果加解密函数是确定型函数 (deterministic function)，我们称为确定型密码 (deterministic cipher)，如果加解密函数是概率算法，我们称为计算型密码 (computational cipher)。

---

<sup>10</sup>这是逻辑中的谓词逻辑概念，在大学课程中应该是离散数学中学过，比如一元谓词 S 表示“是一个学生”，那么“小明是一个学生”就记为 S(" 小明")

### 1.8.1 攻击游戏 (attack game)

我们看一个思想实验，我们称为攻击游戏 (attack game)，在此实验中有一个挑战者 (chanllenger) 和一个敌手 (adversary)。

实验 0(Experiment 0, 简写为 Exp0):

1. 敌手计算  $m_0, m_1 \in M, m_0, m_1$  的长度相同，发给挑战者。  
 $\stackrel{R}{K}, \stackrel{R}{c} E(\kappa, m_0)$
2. 挑战者计算  $\kappa K, c E(\kappa, m_0)$ , 将  $c$  发给敌手。
3. 敌手输出  $b$ ,  $b$  为 0 或为 1.

实验 1(Experiment 1, 简写为 Exp1):

1. 敌手计算  $m_0, m_1 \in M, m_0, m_1$  的长度相同，发给挑战者。  
 $\stackrel{R}{K}, \stackrel{R}{c} E(\kappa, m_1)$
2. 挑战者计算  $\kappa K, c E(\kappa, m_1)$ , 将  $c$  发给敌手。
3. 敌手输出  $b$ ,  $b$  为 0 或为 1.

### 1.8.2 语义安全定义

我们定义两个事件  $W_0, W_1$ ,  $W_0$  表示敌手在 Exp0 实验中返回 1,  $W_1$  表示敌手在 Exp1 实验中最终返回 1. 我们定义敌手的优势 (advantage) 为:

$$Adv = Pr(W_0) - Pr(W_1)$$

如果敌手的优势  $Adv$  可以忽略，我们认为这个加密系统是语义安全<sup>11</sup>。

语义安全的定义要比香农安全的定义弱，但是更加接近实际情况。在语义安全中加解密算法可以是确定型，也可以是概率性。我们这里给出的语义安全的定义依然不是一个严格的形式化定义，比如“什么是可忽略”就没有严格定义出来，但是从这个半形式化定义上，我们依然可以看到语义安全的基本思想。

## 1.9 密码中的常识

下面是参考文献 [7] 给出的对于密码应该有的一些常识。

1. 不要使用保密的算法。
2. 使用低强度密码比不进行任何加密更危险，这是因为给使用者错误的安全感。16 世纪苏格兰女王玛丽对密码盲信，将刺杀伊丽莎白女王计划写入密信，被送上了断头台就是一个典型的事例。
3. 任何密码总有一天都会被破解。
4. 密码只是安全的一部分。例如社会工程 (social engineering) 的例子，攻击者通过内线电话，打给单位职工，冒充网管，说要进行系统测试，让您修改为指定的密码。

---

<sup>11</sup>我们会看到不同的教材对于语义安全的形式化定义会略有不同，但其本质上一样

## 第 2 章 密码学研究内容

---

### 2.1 密码设计

就是研究如何设计一种满足“安全目标的”密码系统，包括密码设计的方法学和具体的密码设计。按照不同的分类方法，可以分为公钥密码体制 (public key cryptosystem) 和对称密码体制 (secret key cryptosystem / symmetric cryptosystem)。也可以分为流密码（也称序列密码，stream ciphers）和分组密码（block ciphers）等。

但是通常在实际应用时，基本的密码算法会在系统的不同实现层次进行应用，并且会用来实现不同的安全目标，而这些也应该是密码应用的研究范畴。

在 [5] 一书中，其根据密码的应用将密码学分为：

- 含糊的安全目标
- 形式化的安全目标
- 密码协议
- 密码学构建模块
- 密码学构建模块的实现

密码算法设计中的通用要求：

- 可逆性 (reversible): 这是基本要求，其实就是要解密算法的存在。
- 对合性 (involution): 这是要求加解密算法中的基础计算部分是可以重用的，这样可以使算法实现的工作量减半，这是从实现角度方面考虑的结果。

#### 2.1.1 密码体制

一个密码体制是指这样一种数学映射：

1. 明文消息空间  $M$ , 某个字母表上的串集,  $M$  表示 message。
2. 密文消息空间  $C$ , 可能的密文消息集,  $C$  表示 cipher。
3. 加密密钥空间  $K_e$ , 可能的加密密钥集,  $K$  表示 key,  $e$  表示 encrypt。
4. 解密密钥空间  $K_d$ , 可能的解密密钥集。 $d$  表示 decrypt。
5. 加密算法  $E : M \times K_e \rightarrow C$ , 也可写为  $c = E_{k_e}(m)$ .
6. 解密算法  $D : C \times K_d \rightarrow M$ , 也可写为  $m = D_{k_d}(c)$ .
7. 密码体制满足  $D_{k_d}(E_{k_e}(m)) = m$ .

一个消息的加密密钥和其对应的解密密钥相同时，我们称这种加密方法为对称密码体制，反之称为非对称密码体制。

## 2.1.2 基于计算困难问题的密码体制设计

我们从香农的讨论中知道，我们可以构造一个完全保密 (perfect secrecy) 密码算法，但是这类密码算法的一个要求就是密钥长度和明文一样长，显然这不具有实操性。那么我们是否能把“数学上的安全”降低为“计算安全”，也就是说，我们并不保证“完全安全”，我们只要保证破解是个需要消耗大量资源的事情即可，比如需要投入 100 万台计算机，运行 100 年。也就是说，如果破译者在选择最有效的算法前提以下，破解一个密码体制依然需要很大的资源，而这个资源是破译者无法承受的，那么我们称为这个密码体制是计算安全的。

下面是 standford 大学的 Dan Boneh 和 Victor Shoup 在 “A Graduate Course in Applied Cryptography” [8]<sup>1</sup> 书中的描述：

### 2.2 Computational ciphers and semantic security

As we have seen in Shannon's theorem(Theorem 2.5), the only way to achieve perfect security is to have keys that are as long as messages. However, this is quite impractical: we would like to be able to encrypt a long message(say, a document of several megabytes) using a short key(say, a few hundred bits). The only way around Shannon's theorem is to relax our security requirements. The way we shall do this is to consider not all possible adversaries, but only computationally feasible adversaries, that is, "real world" adversaries that must perform their calculations on real computers using a reasonable amount of time and memory. This will lead to a weaker definition of security called semantic security. Furthermore, our definition of security will be flexible enough to allow ciphers with variable length message spaces to be considered secure so long as they do not leak any useful information about an encrypted message to an adversary other than the length of message. Also, since our focus is now on the "practical, instead of the "mathematically possible," we shall also insist that the encryption and decryption functions are themselves efficient algorithms, and not just arbitrary functions.

计算安全比较通俗的解释是“如果我们使用最好的算法来破译一个密码体制至少需要  $n$  次操作，而  $n$  是一个非常大的数，则我们称这个密码体制是计算上安全的。”<sup>2</sup>。<sup>3</sup>

下面文字来自 Denning 书 “Cryptography and Data Security” [9] 的 1.5.3。

### 1.5.3. Ciphers Based on Computationally Hard Problems

In their 1976 paper, Diffie and Hellman [10] suggested applying computational complexity to the design of encryption algorithms. They noted that NP-complete problems might make excellent candidates for ciphers because they cannot be solved in polynomial time by any known techniques. Problems that are computationally more difficult than the problems in NP are not suitable for encryption because the enciphering and deciphering transformations must be fast (i.e. computable in polynomial time). But this means the cryptanalyst could guess a key and check

<sup>1</sup>此书的电子版下载地址<https://toc.cryptobook.us/>

<sup>2</sup><http://dict.youdao.com/w/eng/>

<sup>3</sup><https://wenku.baidu.com/view/25ee2f937dd184254b35eefdc8d376eeafaa17c7.html>

the solution in polynomial time<sup>4</sup> (e. g, by enciphering known plaintext). Thus, the cryptanalytic effort to break any polynomial-time encryption algorithm must be in NP.

Diffie and Hellman speculated that cryptography could draw from the theory of NP complexity by examining ways in which NP-complete problems could be adapted to cryptographic use. Information could be enciphered by encoding it in an NP-complete problem in such a way that breaking the cipher would require solving the problem in the usual way. With the deciphering key, however, a shortcut solution would be possible.

To construct such a cipher, secret "trapdoor" information is inserted into a computationally hard problem that involves inverting a one-way function. A function is a one-way function if it is easy to compute  $f(x)$  for any  $x$  in the domain of  $f$ , while, for almost all  $y$  in the range of  $f$ , it is computationally infeasible to compute  $f^{-1}(y)$  even if  $y$  is known. It is a trapdoor one-way function if it is easy to compute  $f^{-1}(y)$  given certain additional information. This additional information is the secret deciphering key.

Public-key systems are based on this principle. The trapdoor knapsack schemes described in Section 2.8 are based on the knapsack problem. The RSA scheme described in Section 2.7 is based on factoring composite numbers.

The strength of such a cipher depends on the computational complexity of the problem on which it is based. A computationally difficult problem does not necessarily imply a strong cryptosystem, however. Shamir gives three reasons[11]:

1. Complexity theory usually deals with single isolated instances of a problem. A cryptanalyst often has a large collection of statistically related problems to solve(e.g, several ciphertexts generated by the same key).
2. The computational complexity of a problem is typically measured by its worst-case or average-case behavior. To be useful as a cipher, the problem must be hard to solve in almost all cases.
3. An arbitrarily difficult problem cannot necessarily be transformed into a cryptosystem, and it must be possible to insert trapdoor information into the problem in such a way that a shortcut solution is possible with this information and only with this information.

Lempel [Lemp79] illustrates the first deficiency with a block cipher for which the problem of finding an  $n$ -bit key is NP-complete when the plaintext corresponding to one block of ciphertext is known. But given enough known plaintext, the problem reduces to solving  $n$  linear equations in  $n$  unknowns. The cipher is described in Section 2.8.4.

Shamir [Sham79] proposes a new complexity measure to deal with the second difficulty, Given a fraction  $r$  such that  $0 < r \leq 1$ , the percentile complexity  $T(n, r)$  of a problem measures the time to solve the easiest proportion  $r$  of the problem instances of size  $n$ . For example,  $T(n, 0.5)$  gives the median complexity; that is, at least half of the instances of size  $n$  can be solved

---

<sup>4</sup>多项式时间是指  $O(n^k)$ , 其中  $k$  为常数,  $n$  为运算的规模, 如果对于一个问题能找到一个多项式算法, 我们称其为 P 类问题。

within time  $T(n, 0.5)$ . The problem of deciding whether a given integer is prime has median complexity  $O(1)$  because half of the numbers have 2 as a factor, and this can be tested in constant time.

With respect to the third difficulty, Brassard [Bras79b] shows it may not be possible to prove that the cryptanalytic effort to invert a trapdoor one-way function is NP-complete. If the function satisfies a few restrictions, then a proof of NP-completeness would imply **NP=CoNP**.



现代密码理论中的计算安全是基于计算复杂理论 (computational complexity) 进行讨论的，关于这部分较详细的理论可以参考 Dan Boneh[8] 和 Mao Wenbo[12] 的书。计算复杂理论一直是理论计算重要研究内容，并且成为数学研究的一个重要分支，2021 年 Abel 讲就授予了理论计算和离散数学的研究者，相关报道内容见扩展阅读<sup>5</sup>。

### 2.1.3 常见的密码困难假设 (Common cryptographic hardness assumptions)

This is a list of some of the most common cryptographic hardness assumptions, and some cryptographic protocols that use them<sup>6</sup>.

- Integer factorization (整数分解)
  - Rabin cryptosystem
  - Blum Blum Shub generator
  - Okamoto-Uchiyama cryptosystem
  - Hofheinz-Kiltz-Shoup cryptosystem
- RSA problem (weaker than factorization)
  - RSA cryptosystem
- Quadratic residuosity problem (stronger than factorization) (二次剩余问题)
  - Goldwasser-Micali cryptosystem
- Decisional composite residuosity assumption (stronger than factorization)<sup>7</sup>
  - Paillier cryptosystem
- Higher residuosity problem (stronger than factorization)
  - Benaloh cryptosystem
  - Naccache-Stern cryptosystem
- Phi-hiding assumption (stronger than factorization)
  - Cachin-Micali-Stadler PIR

<sup>5</sup>2021 年 3 月 17 日晚，被誉为数学界“诺贝尔奖”的阿贝尔奖揭晓。挪威科学和文学学院决定将 2021 年阿贝尔奖授予匈牙利厄特沃什·罗兰大学教授拉兹洛·洛瓦兹 (László Lovász) 和美国普林斯顿高等研究院教授艾维·维格森 (Avi Wigderson)，以“表彰他们在理论计算机科学 (theoretical computer science) 和离散数学 (discrete mathematics) 方面作出的杰出贡献，以及使其在现代数学中心领域中发挥主导作用。”

<sup>6</sup><https://encyclopedia.thefreedictionary.com/Computational+hardness+assumption>

<sup>7</sup>DCRA，是指给定一个合数  $n$  和一个整数  $z$ ，判断  $z$  是否是模  $n^2$  的一个  $n$  剩余或是否存在  $y$ ，满足  $z \equiv y^n \pmod{n^2}$

- Discrete log problem (DLP)(离散对数问题)
- Computational Diffie–Hellman assumption (CDH; stronger than DLP)
  - Diffie–Hellman key exchange
- Decisional Diffie–Hellman assumption (DDH; stronger than CDH)
  - ElGamal encryption
- Shortest Vector Problem (最短向量问题)
  - NTRUEncrypt
  - NTRUSign

## 2.2 密码分析

密码分析（英语：cryptanalysis，来源于希腊语 *kryptós*，即“隐藏”，以及 *analýein*，即“解开”），是一门研究在不知道通常解密所需要的秘密信息的情况下对信息进行解密的学问。通常，这需要寻找一个秘钥。也就是通常我们说的破解密码。

根据密码分析员的能力，可以将密码分析员对密码系统的攻击分为以下几种 [2]：

1. 密码分析员掌握除了密钥外，密码系统的加密和解密算法.
2. 仅知密文攻击 (ciphertext-only attack)，密码分析员能够获得密文.
3. 已知明文攻击 (known-plaintext attack)，密码分析员能够获得某些明文和这些明文对应的密文.
4. 选择明文攻击 (chosen-plaintext attack)，密码分析员能够有选择地获得明文和这些明文对应的密文.
5. 选择密文攻击 (chosen-ciphertext attack)，密码分析员可以像合法用户那样发送加密的信息.
6. 密码分析员可以改变、截取或重新发送信息。

按照分析的方法，可以分为：

- 穷举法，字典攻击。
- 数学攻击：统计攻击、差分攻击、线性攻击、代数攻击、相关攻击。
- 物理攻击：侧信道分析 (side channel attack) 和与硬件相关的能量分析、时间分析、声音分析、电磁辐射。通常理论上安全的算法，但由于物理实现上的不足，而使得安全性出现问题，这就对硬件设计、密码芯片设计提出更高要求。

## 2.3 密码测评

如何评价密码算法的安全性，如何评价好坏，有多好，多坏，以及如何评价一个加密算法在产品实现中的安全性，在系统的实现中都是很重要的。

密码测评 (cipher evaluation) 是与密码分析有着很多相同点，但是又不同的概念，特别是随着密码的产业化应用，这种不同越来越重要，并且相互不能等同。

以下是来自论文 [13] 中对于密码测评和分析概念不同的描述。

*Evaluation* is a process intended for highlight some unconformities or deficiencies of a cryptosystem which can be used by a cracker.

The evaluation of a cryptographic module can be done using NIST FIPS 140-2 standard (structured on fourth levels) and the evaluation of a product can be done using Common Criteria (ISO 15408)<sup>8</sup>, methodology adopted by USA, Canada and EU (structured on seventh levels).

*Cracking* represents an operation helping to design a technique, method or algorithm that permit the recovery of the system key or of the plain text having a reduced complexity than brute force attack method:

- the evaluator wants to find the minimum quantity of output information that help him to determine, using some strong mathematical tools, a series of information about the cipher algorithm, used key and/or plain text;
- the cracker wants to find the maximum quantity of information that help him to deduce the plain text.

---

<sup>8</sup>此标准在我国对应的是国家标准 GB/T18336

# 第3章 古典密码 (Classical Encryption)

这部分我们介绍一下古典密码。

注意 问：什么是古典

答：很早以前的

问：多早以前的

答：What? Can u speak English? <sup>1</sup>

## 3.1 凯撒密码 Caesar cipher

$$\begin{cases} c = m + 3 \pmod{26} & 0 \leq m \leq 25 \\ m = c - 3 \pmod{26} & 0 \leq c \leq 25 \end{cases} \quad (3.1)$$

例 3.1 利用凯撒密码对"LI XIAO FENG SHI LAO SHI" 进行加密。

解 首先需加密的都是大写，我们使用 Ascii 码对其编码，并且去掉空格，变为 "LIXIAOFENGSHILAOSHI"，A~Z 的 Ascii 码为 65~90，所以计算时编码要减去 65。

## 3.2 移位变换 (shift transformation)/加法密码

$$\begin{cases} c = m + k \pmod{26}, m \geq 0, k \leq 25 \\ m = c - k \pmod{26}, c \geq 0, k \leq 25 \end{cases} \quad (3.2)$$

## 3.3 乘法密码

$$\begin{cases} c = am \pmod{26} \\ m = bc \pmod{26}, b = a^{-1} \pmod{26} \end{cases} \quad (3.3)$$

<sup>1</sup> “古典”本身就是一个模糊的定义，通常我们认为香农经典论文发表之前的为“古典”，或者利用香农的基本思想设计的密码系统将不是“古典密码”。

### 3.4 仿射变换 (affine transformation)

$$\begin{cases} c = am + b \pmod{26} \\ m = a^{-1}(c - b) \pmod{26}, \quad a \geq 0, b \leq 25, \gcd(a, 26) = 1, a^{-1}a = 1 \pmod{26} \end{cases} \quad (3.4)$$

仿射变换中  $a, b$  是密钥。

### 3.5 多表代换密码

多表代换密码首先将明文  $M$  进行分组，每组长度为  $n$  个字母<sup>2</sup>，分组后的明文序列表示为  $M_1, M_2, \dots, M_f, M_i (i = 1, 2, \dots, f)$  表示分组消息，加密为：

$$C_i = AM_i + B \pmod{N}, i = 1, 2, \dots, f$$

其中  $A$  是  $n \times n$  可逆矩阵，满足  $\gcd(|A|, N) = 1$ ,  $|A|$  表示矩阵  $A$  的行列式， $B$  为  $n \times 1$  矩阵， $M_i$  为一分组的  $n \times 1$  矩阵表示， $C_i$  是加密后所得密文分组的  $n \times 1$  矩阵表示。对密文分组的解密为：

$$M_i = A^{-1}(C_i - B) \pmod{N}$$

通常对字母进行加密时， $N = 26$ 。

---

<sup>2</sup>注意分组密码时的填充问题，也就是说当最后一组不够  $n$  的长度时，需要补齐，补齐时需要考虑，解密方在解密后，如何知道是补齐的信息，还是原信息。



## 第4章 流密码 (Stream Ciphers)

---

利用密钥产生一个密钥流，利用密钥流对数据流进行加密。密钥流和数据流，这个流是什么意思？流（stream）从直观上来讲，应该是一种一个接着一个的运动方式。一个接着一个，这个“个”是什么？应该是一个最小单位，比如“人流”，这个“个”就是人，“水流”“泥石流”这个单位就不太好确定，与研究者的研究粒度和方法有关。

我们看看下面的一段文字 (65 个字符，包含标点符号):

这是一门网络空间安全学科的基础课，是信息安全专业的一门专业基础课，  
学生通过学习这门课要理解加解密的基本原理，掌握基本的加解密方法。

这些文字存成纯文本模式（想想这个含义是什么？），共有占 195 个字节（我的计算机是 MacBook，我查看存储的 txt 文件，文件占用了 4k 的磁盘空间，想想为什么？），简单计算可知一个字符占 3 个字节，用十六进制显示为：

```
E8BF99 E698AF E4B880 E997A8 E7BD91 E7BB9C E7A9BA E997B4 E5AE89 E585A8  
E5ADA6 E7A791 E79A84 E59FBA E7A180 E8AFBE EFBC8C E698AF E4BFA1 E681AF  
E5AE89 E585A8 E4B893 E4B89A E79A84 E4B880 E997A8 E4B893 E4B89A E59FBA  
E7A180 E8AFBE EFBC8C E5ADA6 E7949F E9809A E8BF87 E5ADA6 E4B9AO E8BF99  
E997A8 E8AFBE E8A681 E79086 E8A7A3 E58AA0 E8A7A3 E5AF86 E79A84 E59FBA  
E69CAC E58E9F E79086 EFBC8C E68E8C E68FA1 E59FBA E69CAC E79A84 E58AA0  
E8A7A3 E5AF86 E696B9 E6B395 E38082
```

同样把这段信息以二进制编码的方式显示：

```
00000000: 11101000 10111111 10011001 11100110 10011000 10101111 .....  
00000006: 11100100 10111000 10000000 11101001 10010111 10101000 .....  
0000000c: 11100111 10111101 10010001 11100111 10111011 10011100 .....  
00000012: 11100111 10101001 10111010 11101001 10010111 10110100 .....  
00000018: 11100101 10101110 10001001 11100101 10000101 10101000 .....  
0000001e: 11100101 10101101 10100110 11100111 10100111 10010001 .....  
00000024: 11100111 10011010 10000100 11100101 10011111 10111010 .....  
0000002a: 11100111 10100001 10000000 11101000 10101111 10111110 .....  
00000030: 11101111 10111100 10001100 11100110 10011000 10101111 .....  
00000036: 11100100 10111111 10100001 11100110 10000001 10101111 .....  
0000003c: 11100101 10101110 10001001 11100101 10000101 10101000 .....  
00000042: 11100100 10111000 10010011 11100100 10111000 10011010 .....  
00000048: 11100111 10011010 10000100 11100100 10111000 10000000 .....  
0000004e: 11101001 10010111 10101000 11100100 10111000 10010011 .....  
00000054: 11100100 10111000 10011010 11100101 10011111 10111010 .....  
0000005a: 11100111 10100001 10000000 11101000 10101111 10111110 .....  
00000060: 11101111 10111100 10001100 11100101 10101101 10100110 .....  
00000066: 11100111 10010100 10011111 11101001 10000000 10011010 .....
```

```

0000006c: 11101000 10111111 10000111 11100101 10101101 10100110 .....
00000072: 11100100 10111001 10100000 11101000 10111111 10011001 .....
00000078: 11101001 10010111 10101000 11101000 10101111 10111110 .....
0000007e: 11101000 10100110 10000001 11100111 10010000 10000110 .....
00000084: 11101000 10100111 10100011 11100101 10001010 10100000 .....
0000008a: 11101000 10100111 10100011 11100101 10101111 10000110 .....
00000090: 11100111 10011010 10000100 11100101 10011111 10111010 .....
00000096: 11100110 10011100 10101100 11100101 10001110 10011111 .....
0000009c: 11100111 10010000 10000110 11101111 10111100 10001100 .....
000000a2: 11100110 10001110 10001100 11100110 10001111 10100001 .....
000000a8: 11100101 10011111 10111010 11100110 10011100 10101100 .....
000000ae: 11100111 10011010 10000100 11100101 10001010 10100000 .....
000000b4: 11101000 10100111 10100011 11100101 10101111 10000110 .....
000000ba: 11100110 10010110 10111001 11100110 10110011 10010101 .....
000000c0: 11100011 10000000 10000010 00001010 .. .

```

对于怎么看上面的这一个数据流，其实与我们的处理方法，或者加密方法的是有关的。通常我们将“流”看为一个二进制数据流。

## 4.1 基本概念

我们用小写  $m, c, k \dots$  表示二进制的位,  $m$  表示原始数据,  $c$  表示加密后数据,  $k$  表示密钥, 大写  $E$  表示二进制的运算 (也可称为函数、算子), 那么对于一个流加密我们可以表示为  $c_i = E(m_i, k_i)$ , 其中  $i = 0, 1, 2, \dots$ ,  $i$  表示是第几位。我们可以看到流密码的核心就是设计  $E$  和密钥流  $k_0, k_1, k_2, \dots$ 。但是由于是位运算,  $E$  也没有什么好设计的 (为什么?), 通常会用异或运算, 异或运算好处是加解密都可以用这个运算。

表 4.1: "异或" 运算/函数定义 ("异或" 运算真值表)

x	y	$z = x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

那么流密码的设计将归结为密钥的设计, 就是设计如何产生一个密钥流  $k_i = G(s) i = 0, 1, 2, \dots$ ,  $s$  是密钥产生的种子 (seeds), 也就是产生密钥流的一个初始值,  $G$ (generator) 是密钥生成算法, 可见  $s$  是整个算法的秘密。流密码的加解密可以写为:

$$G(s) \oplus m_i = c_i \quad (4.1)$$

$$G(s) \oplus c_i = m_i \quad (4.2)$$

## 4.2 密钥的产生

根据香浓的完全安全系统的定义, 如果密钥流是一个随机数, 那么就可以构成一个完全安全加密系统, 但是无法达到这个要求, 所以通常我们会用伪随机数发生器 PRG(pseudo-random generator) 来产生密钥流。

### 4.2.1 线性同余发生器 LCG(linear congruential generator)

可以利用  $x_n = (ax_{n-1} + b) \bmod m$  产生一个周期不超过  $m$  的伪随机数序列,  $x_0$  为种子 (seed), 也就是密钥。

LCG 不用在密码学中, 因其在 1977 年被 J.A. Reeds 破译, 也就是找到预测的方法, 但是其在一些需要产生随机数的场合下依然有应用, 比如在一些测试中。

下面我么实际给出一个例子, 计算一下, 使得大家有个直观感觉。

**例 4.1** 序列的同余递推式:

$$x_n = 2x_{n-1} \pmod{5}$$

初始值  $x_0 = 3$ , 产生的序列依次为:  $x_1 = 1, x_2 = 2, x_3 = 4, x_4 = 3, x_5 = 1, x_6 = 2, x_7 = 4$ , 可见其产生的循环序列  $(3, 1, 2, 4)^*$ .

$$x_n = 2x_{n-1} + 3 \pmod{307} \quad (4.3)$$

我们在 SageMath 内编写一小段程序 (见表4.2), 看看其所产生的序列。

**表 4.2:** 同余方程  $x_n = 2x_{n-1} + 3 \pmod{307}$  输出序列的 SageMath 程序

```
sage: b=67
.....: for i in range(0,307):
.....:     a=b
.....:     b=mod(2*a+3,307)
.....:     print i,":",b
.....:
```

start	6 : 54	13 : 232	20 : 298
0 : 137	7 : 111	14 : 160	21 : 292
1 : 277	8 : 225	15 : 16	22 : 280
2 : 250	9 : 146	16 : 35	23 : 256
3 : 196	10 : 295	17 : 73	24 : 208
4 : 88	11 : 286	18 : 149	25 : 112
5 : 179	12 : 268	19 : 301	26 : 227

27 : 150	65 : 141	102 : 137	140 : 201
28 : 303	66 : 285	103 : 277	141 : 98
29 : 302	67 : 266	104 : 250	142 : 199
30 : 300	68 : 228	105 : 196	143 : 94
31 : 296	69 : 152	106 : 88	144 : 191
32 : 288	70 : 0	107 : 179	145 : 78
33 : 272	71 : 3	108 : 54	146 : 159
34 : 240	72 : 9	109 : 111	147 : 14
35 : 176	73 : 21	110 : 225	148 : 31
36 : 48	74 : 45	111 : 146	149 : 65
37 : 99	75 : 93	112 : 295	150 : 133
38 : 201	76 : 189	113 : 286	151 : 269
39 : 98	77 : 74	114 : 268	152 : 234
40 : 199	78 : 151	115 : 232	153 : 164
41 : 94	79 : 305	116 : 160	154 : 24
42 : 191	80 : 306	117 : 16	155 : 51
43 : 78	81 : 1	118 : 35	156 : 105
44 : 159	82 : 5	119 : 73	157 : 213
45 : 14	83 : 13	120 : 149	158 : 122
46 : 31	84 : 29	121 : 301	159 : 247
47 : 65	85 : 61	122 : 298	160 : 190
48 : 133	86 : 125	123 : 292	161 : 76
49 : 269	87 : 253	124 : 280	162 : 155
50 : 234	88 : 202	125 : 256	163 : 6
51 : 164	89 : 100	126 : 208	164 : 15
52 : 24	90 : 203	127 : 112	165 : 33
53 : 51	91 : 102	128 : 227	166 : 69
54 : 105	92 : 207	129 : 150	167 : 141
55 : 213	93 : 110	130 : 303	168 : 285
56 : 122	94 : 223	131 : 302	169 : 266
57 : 247	95 : 142	132 : 300	170 : 228
58 : 190	96 : 287	133 : 296	171 : 152
59 : 76	97 : 270	134 : 288	172 : 0
60 : 155	98 : 236	135 : 272	173 : 3
61 : 6	99 : 168	136 : 240	174 : 9
62 : 15	100 : 32	137 : 176	175 : 21
63 : 33	101 : 67	138 : 48	176 : 45
64 : 69	*****	139 : 99	177 : 93

178 : 189	211 : 111	244 : 199	277 : 21
179 : 74	212 : 225	245 : 94	278 : 45
180 : 151	213 : 146	246 : 191	279 : 93
181 : 305	214 : 295	247 : 78	280 : 189
182 : 306	215 : 286	248 : 159	281 : 74
183 : 1	216 : 268	249 : 14	282 : 151
184 : 5	217 : 232	250 : 31	283 : 305
185 : 13	218 : 160	251 : 65	284 : 306
186 : 29	219 : 16	252 : 133	285 : 1
187 : 61	220 : 35	253 : 269	286 : 5
188 : 125	221 : 73	254 : 234	287 : 13
189 : 253	222 : 149	255 : 164	288 : 29
190 : 202	223 : 301	256 : 24	289 : 61
191 : 100	224 : 298	257 : 51	290 : 125
192 : 203	225 : 292	258 : 105	291 : 253
193 : 102	226 : 280	259 : 213	292 : 202
194 : 207	227 : 256	260 : 122	293 : 100
195 : 110	228 : 208	261 : 247	294 : 203
196 : 223	229 : 112	262 : 190	295 : 102
197 : 142	230 : 227	263 : 76	296 : 207
198 : 287	231 : 150	264 : 155	297 : 110
199 : 270	232 : 303	265 : 6	298 : 223
200 : 236	233 : 302	266 : 15	299 : 142
201 : 168	234 : 300	267 : 33	300 : 287
202 : 32	235 : 296	268 : 69	301 : 270
203 : 67	236 : 288	269 : 141	302 : 236
204 : 137	237 : 272	270 : 285	303 : 168
205 : 277	238 : 240	271 : 266	304 : 32
206 : 250	239 : 176	272 : 228	305 : 67
207 : 196	240 : 48	273 : 152	306 : 137
208 : 88	241 : 99	274 : 0	
209 : 179	242 : 201	275 : 3	
210 : 54	243 : 98	276 : 9	



**注意** [14] 线性同余发生器的优点是：速度快，每位只需很少的运算。

然而，线性同余发生器不能用在密码学中，因为它们是可预测的。线性同余发生器首先被 Jim Reeds 破译（文章“Cracking Random Number Generator”），然后被 Joan Boyar（见文章“Inferring a sequence generated by a linear congruence”）破译，Boyar 还破译了二次同余

发生器：

$$X_n = (aX_{n-1}^2 + bX_{n-1} + c) \pmod{m}$$

和三次同余发生器：

$$X_n = (aX_{n-1}^3 + bX_{n-1}^2 + cX_{n-1} + d) \pmod{m}$$

另一些研究人员将 Boyar 的成果扩展到了任意多项式同余发生器 3N 假短线性同余发生器和未知参数的截短线性同余发生器也被破译。上述证据表明：同余发生器在密码学中并不适用。

然而线性同余发生器在非密码学应用中得到了使用，比如仿真，根据最合理的经验测试，它们是有效的，并具有很好的统计性能。关于线性同余发生器实现方面的重要信息可在 P L'Ecuyer 的文章"Random Numbers for Simulation" 中找到。

许多人考察了组合线性同余发生器，结果是它并没有增加安全性，但是它有更长的周期并在某些随机性测试方面具有更好的性能。

### 4.2.2 线性反馈移位寄存器 LFSR(Linear Feedback shift register)

线性反馈移位寄存器可以有图形化表示方法，如4.1：

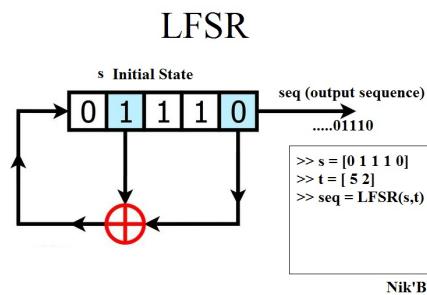


图 4.1: LFSR 的图形化表示

也有用逻辑电路图来表示，如4.2：

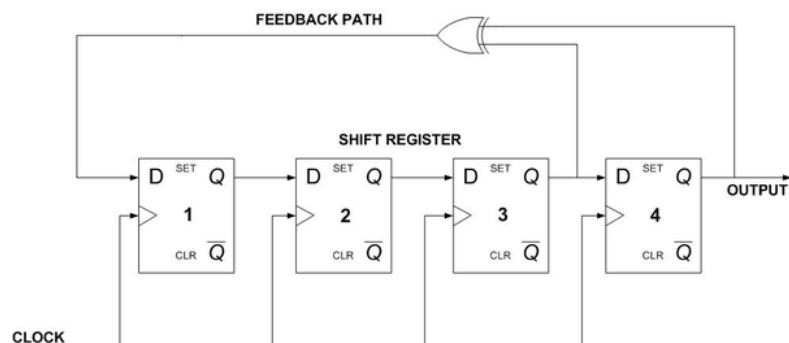


图 4.2: LFSR 的逻辑电路图表示

下面我们看一个有三个寄存器的例子。

我们首先设计一个移位寄存器，反馈直接由最低位（或者说输出）到最高位。如图4.3所

示，寄存器初始值为 101，输出序列变化过程如表4.3所示，其中输出比特用波浪线标识出来。我们可以看出输入序列为：101101101…，输出序列的周期为 3.

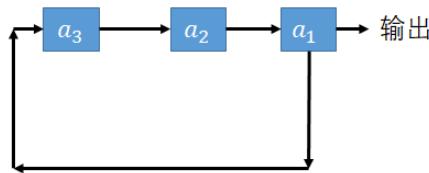


图 4.3: 3 个寄存器直接由输出反馈最高位

表 4.3: 初始值为 101 的直接反馈移位寄存器输出序列

1	0	1									
1	1	0	1								
0	1	1	0	1							
1	0	1	1	0	1						
1	1	0	1	1	0	1					
0	1	1	0	1	1	0	1				
1	0	1	1	0	1	1	0	1			
1	1	0	1	1	0	1	1	0	1		
0	1	1	0	1	1	0	1	1	0	1	

下面我们同样是 3 个寄存器，但是采用不同的反馈方式，如图4.4所示，寄存器初始值依然为：101，输出序列如4.4所示，其中输出比特用波浪线标识出来，我们可以看到输出序列为：10100111010011101001110，周期为 7。

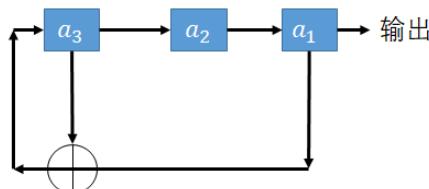


图 4.4: 3 个寄存器最大循环周期反馈方式

通过上面的例子，我们可以看出同等数量的寄存器，不同的反馈方法，输出序列的周期不同，那么这就引出一个问题，如何能输出最长周期？最长周期是多少？

#### 4.2.2.1 有限域上的多项式

有限域 (finite field, 称为伽罗瓦域, Galois field) GF(2)，集合为 0,1，集合上的两个运算为异或 Xor(对应通常的域定义中的加, 记为 +) 和与 and(对应通常域定义中的乘, 记为 ·)，我们看看 GF(2) 的多项式 (polynomial)，这个多项式的系数 (coefficients) 来自于 GF(2)。

**例 4.2** 多项式的完整表达。

$$x^7 + x^6 + 1 = 1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$$

**例 4.3** 加/减 (addition/subtraction) 运算示例。

表 4.4: 初始值为 101 的反馈移位寄存器输出 M 序列

$$\begin{aligned}
 & (x^4 + x^3 + x + 1) + (x^4 + x^2 + x) \\
 = & (1 \cdot x^4 + 1 \cdot x^4) + (1 \cdot x^3 + 0 \cdot x^3) + (0 \cdot x^2 + 1 \cdot x^2) + (1 \cdot x + 1 \cdot x) + (1 \cdot x^0 + 0 \cdot x^0) \\
 = & 0 \cdot x^4 + 1 \cdot x^2 + 1
 \end{aligned}$$

**例 4.4** 乘 (multiplication) 运算示例。

$$\begin{aligned} & (x^2 + x + 1)(x + 1) \\ &= x^3 + x^2 + x + x^2 + x + 1 \\ &= x^3 + 1 \end{aligned}$$

**例 4.5** 除 (division) 运算示例。

$$\begin{aligned}
 & \text{我们先根据除的基本定义来计算: } (x^4 + x^3 + x + 1) / (x^2 + 1) \Rightarrow \\
 & (x^4 + x^3 + x + 1) \text{ xor } [(x^2 + 1) \cdot \underline{x^2}] = (x^4 + x^3 + x + 1) \text{ xor } (x^4 + x^2) = x^3 + x^2 + x + 1 \Rightarrow \\
 & (x^3 + x^2 + x + 1) \text{ xor } [(x^2 + 1) \cdot \underline{x}] = (x^3 + x^2 + x + 1) \text{ xor } (x^3 + x) = x^2 + 1 \Rightarrow \\
 & (x^2 + 1) \text{ xor } [(x^2 + 1) \cdot \underline{1}] = 0
 \end{aligned}$$

商是  $x^2 + x + 1$ , 余是 0.

我们把上面的计算过程可以写成长除式：

$$\begin{array}{r}
 \begin{array}{cccc} x^2 & +x & +1 \\ \hline x^4 & +x^3 & +x & +1 \end{array} \\
 x^2 + 1 \sqrt{\quad} \begin{array}{c} x^4 \\ \hline x^3 & +x^2 & +x \\ \hline x^3 & +x & \\ \hline x^2 & & +1 \\ \hline x^2 & & +1 \\ \hline 0 \end{array}
 \end{array}$$

**定义 4.1 (多项式的度)** 一个多项式的最高次数，我们称为此多项式的度 (degree)。

**例 4.6** 看看下面的多项式度是多少。

$$1. x^4 + x^3 + x + 1$$

$$2. x^2 + x + 1$$

解答：度分别为 4 和 2.

**定义 4.2 (多项式的周期或阶)** 设  $p(x)$  是 GF(2) 上的多项式，且  $p(0) \neq 0$ ，多项式  $p(x)$  的阶 (order) 是一个最小整数  $e$ ,  $e$  满足  $p(x)$  能整除  $x^e + 1$ . <sup>1</sup>

一个素多项式 (prime polynomial) 是指不能表示为两个多项式乘积的多项式. 对于任  
何度  $n$ , 最少存在一个素多项式  $p(x)$ , 并且使用这个素多项式, 可以构成一个有限域, 我们记为  $GF(2^n)$ , 有些教材为了明确是多项式域, 有时候也记为  $GF(2^n)[x]$ , 运算为模  $p(x)$ .

**例 4.7** 示例  $GF(2^4)$ .

$p(x) = x^4 + x + 1$  是一个素多项式，我们用这个素多项式构造  $GF(2^4)$ .

$x^0 \pmod{p(x)} = 1 \pmod{p(x)}$ , 注意此处 **1** 的含义是指 **1 · anyone = anyone**, 是单位元

$x^1 \pmod{p(x)} = x \pmod{p(x)}$

$x^2 \pmod{p(x)} = x^2 \pmod{p(x)}$

$x^3 \pmod{p(x)} = x^3 \pmod{p(x)}$

<sup>1</sup>The highest order power in a univariate polynomial is known as its order (or, more properly, its polynomial degree). For example, the polynomial  $P(x) = a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$  is of order  $n$ , denoted  $\deg P(x)=n$ . It is preferable to use the word "degree" for the highest exponent in a polynomial, since a completely different meaning is given to the word "order" in polynomials taken modulo some integer (where this meaning is the one used in the multiplicative order of a modulus). In particular, the order of a polynomial  $P(x)$  with  $P(0) \neq 0$  is the smallest integer  $e$  for which  $P(x)$  divides  $x^e + 1$  (Lidl and Niederreiter 1994). For example, in the finite field  $GF(2)$ , the order of  $x^5 + x^2 + 1$  is 31, since

$$\frac{x^{31} + 1}{x^5 + x^2 + 1} = 1 + x^2 + x^4 + x^5 + x^6 + x^8 + x^9 + x^{13} + x^{14} + x^{15} + x^{16} + x^{17} + x^{20} + x^{21} + x^{23} + x^{26} \pmod{2}$$
. From <https://mathworld.wolfram.com/PolynomialOrder.html>

$$\begin{aligned}
 x^4(\text{mod } p(x)) &= x^4 \text{ xor } p(x) = x^4 \text{ xor } x^4 + x + 1 = x + 1^2 \\
 x^5(\text{mod } p(x)) &= x \cdot x^4 = x(x+1) = x^2 + x(\text{mod } p(x)) \\
 x^6(\text{mod } p(x)) &= x \cdot x^5 = x(x^2+x) = x^3 + x^2(\text{mod } p(x)) \\
 x^7(\text{mod } p(x)) &= x \cdot x^6 = x(x^3+x^2) = x^4 + x^3(\text{mod } p(x)) = (x^4+x^3)\text{ xor }(x^4+x+1) = x^3 + x + 1 \\
 x^8(\text{mod } p(x)) &= x \cdot x^7 = x(x^3+x+1) = x^4 + x^2 + x(\text{mod } p(x)) = (x^4+x^2+x)\text{ xor }(x^4+x+1) = x^2 + 1 \\
 x^9(\text{mod } p(x)) &= x \cdot x^8 = x(x^2+1) = x^3 + x(\text{mod } p(x)) \\
 x^{10}(\text{mod } p(x)) &= x \cdot x^9 = x(x^3+x) = x^4 + x^2(\text{mod } p(x)) = (x^4+x^2)\text{ xor }(x^4+x+1) = x^2 + x + 1 \\
 x^{11}(\text{mod } p(x)) &= x \cdot x^{10} = x(x^2+x+1) = x^3 + x^2 + x(\text{mod } p(x)) \\
 x^{12}(\text{mod } p(x)) &= x \cdot x^{11} = x(x^3+x^2+x)(\text{mod } p(x)) = (x^4+x^3+x^2)\text{ xor }(x^4+x+1) = x^3 + x^2 + x + 1 \\
 x^{13}(\text{mod } p(x)) &= x \cdot x^{12} = x(x^3+x^2+x+1)(\text{mod } p(x)) = (x^4+x^3+x^2+x)\text{ xor }(x^4+x+1) = x^3 + x^2 + 1 \\
 x^{14}(\text{mod } p(x)) &= x \cdot x^{13} = x(x^3+x^2+1)(\text{mod } p(x)) = (x^4+x^3+x)\text{ xor }(x^4+x+1) = x^3 + 1 \\
 x^{15}(\text{mod } p(x)) &= x \cdot x^{14} = x(x^3+1)(\text{mod } p(x)) = (x^4+x)\text{ xor }(x^4+x+1) = 1 \\
 x^{16}(\text{mod } p(x)) &= x \cdot x^{15} = x(\text{mod } p(x))
 \end{aligned}$$

通常找一个素多项式是一个困难问题，通常都会查表。

我们看看 Partow 给出的一个本原多项式列表（这个表共给出  $n$  为 32 的，但是整个表太大，所以此处只拷贝了  $n=2\sim 8$  的）<sup>3</sup>：

```

x^2 + x^1 + 1
x^3 + x^1 + 1
x^3 + x^2 + 1
x^4 + x^1 + 1
x^4 + x^3 + 1
x^4 + x^3 + x^2 + x^1 + 1
x^5 + x^2 + 1
x^5 + x^3 + 1
x^5 + x^3 + x^2 + x^1 + 1
x^5 + x^4 + x^2 + x^1 + 1
x^5 + x^4 + x^3 + x^1 + 1
x^5 + x^4 + x^3 + x^2 + 1
x^6 + x^1 + 1
x^6 + x^3 + 1
x^6 + x^4 + x^2 + x^1 + 1
x^6 + x^4 + x^3 + x^1 + 1
x^6 + x^5 + 1
x^6 + x^5 + x^2 + x^1 + 1
x^6 + x^5 + x^3 + x^2 + 1

```

<sup>2</sup>注意此处模的基本概念，模是余的概念，多少基本单元剩下的。

<sup>3</sup>来自于 Partow 的网站 <https://www.partow.net/programming/polynomials/index.html>

```
x^6 + x^5 + x^4 + x^1 + 1
x^6 + x^5 + x^4 + x^2 + 1
x^7 + x^1 + 1
x^7 + x^3 + 1
x^7 + x^3 + x^2 + x^1 + 1
x^7 + x^4 + 1
x^7 + x^4 + x^3 + x^2 + 1
x^7 + x^5 + x^2 + x^1 + 1
x^7 + x^5 + x^3 + x^1 + 1
x^7 + x^5 + x^4 + x^3 + 1
x^7 + x^5 + x^4 + x^3 + x^2 + x^1 + 1
x^7 + x^6 + 1
x^7 + x^6 + x^3 + x^1 + 1
x^7 + x^6 + x^4 + x^1 + 1
x^7 + x^6 + x^4 + x^2 + 1
x^7 + x^6 + x^5 + x^2 + 1
x^7 + x^6 + x^5 + x^3 + x^2 + x^1 + 1
x^8 + x^4 + x^3 + x^1 + 1
x^8 + x^4 + x^3 + x^2 + 1
x^8 + x^5 + x^3 + x^1 + 1
x^8 + x^5 + x^3 + x^2 + 1
x^8 + x^5 + x^4 + x^3 + 1
x^8 + x^5 + x^4 + x^3 + x^2 + x^1 + 1
x^8 + x^6 + x^3 + x^2 + 1
x^8 + x^6 + x^4 + x^3 + x^2 + x^1 + 1
x^8 + x^6 + x^5 + x^1 + 1
x^8 + x^6 + x^5 + x^2 + 1
x^8 + x^6 + x^5 + x^3 + 1
x^8 + x^6 + x^5 + x^4 + 1
x^8 + x^6 + x^5 + x^4 + x^2 + x^1 + 1
x^8 + x^6 + x^5 + x^4 + x^3 + x^1 + 1
x^8 + x^7 + x^2 + x^1 + 1
x^8 + x^7 + x^3 + x^1 + 1
x^8 + x^7 + x^3 + x^2 + 1
x^8 + x^7 + x^4 + x^3 + x^2 + x^1 + 1
x^8 + x^7 + x^5 + x^1 + 1
x^8 + x^7 + x^5 + x^3 + 1
x^8 + x^7 + x^5 + x^4 + 1
x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1
x^8 + x^7 + x^6 + x^1 + 1
```

```

x^8 + x^7 + x^6 + x^3 + x^2 + x^1 + 1
x^8 + x^7 + x^6 + x^4 + x^2 + x^1 + 1
x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + 1
x^8 + x^7 + x^6 + x^5 + x^2 + x^1 + 1
x^8 + x^7 + x^6 + x^5 + x^4 + x^1 + 1
x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1
x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1

```

### 4.2.2.2 LFSR 的多项式表示

一个 LFSR 可以用一元多项式来表示，移位寄存器与多项式的对应一般对应关系如图4.5所示，对应的多项式为  $1 + c_1x + c_{n-1}x^{n-1} + c_nx^n$ ，其中  $c_i \in 0, 1$ 。

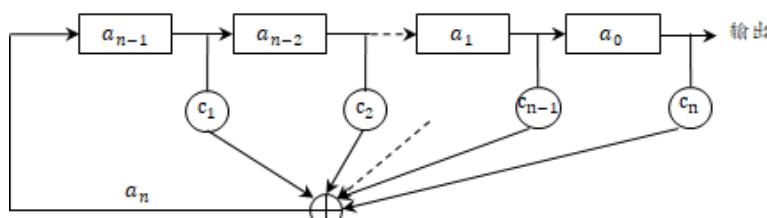


图 4.5: n 位移位寄存器对应一元多项式

如图4.3的三位循环移位寄存器，其对应的多项式可表示为:  $x^3 + 1$ ，如图4.4的三位移位寄存器，其对应的多项式可以表示为:  $x^3 + x + 1$ 。

**例 4.8** 我们给定一个二进制序列 1001011，其多项式表示为:

$$1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x$$

我们选择反馈函数为一个 7 阶本原多项式  $x^7 + x + 1$ ，对上面的多项式乘 x，变为:

$$1 \cdot x^8 + 1 \cdot x^7 + 0 \cdot x^6 + 1 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2$$

乘 x 相等于右移，右移后 x 的一次项系数由反馈函数决定。

对于一个由 n 个寄存器组成的 LFSR，其最多可能的状态是  $2^n$  个，但是在全零状态，其后续状态不会变化，所以我们除去这个状态，那也就是说 n 个寄存器组成的 LFSR 最多可能有  $2^n - 1$  个状态。

**我们希望在一定的资源配置下，能够得到周期最长的伪随机序列。**

在二十世纪六十年代左右，人们对 LFSR 进行了系统的研究，国外对其系统研究的学者 Golomb 出版了一本书 Shift Register Sequences[15] 对移位寄存器的研究成果有系统的阐述，我国中科院数学研究所的戚征也在 1972 在“数学的实践与认识”期刊上发表了“伪随机序列”，肖国镇在 1985 年也出版“伪随机序列及其应用” [16] 一书，书中也有系统的阐述。下面我们只看几个重要的结论，关于移位寄存器详细讨论可以参考相关文献。

**定理 4.1 (LFSR 最大周期)**  $n$  级 LFSR 产生的序列最大周期为  $2^n - 1$ . 我们将最大周期序列，称为  $m$  序列。

**定理 4.2 (LFSR 最大周期必要条件)**  $n$  级 LFSR 产生的序列最大周期  $2^n - 1$  的必要条件是其特征多项式是不可约的。

**定义 4.3 (本原多项式)** 若  $n$  次不可约多项式  $p(x)$  的阶为  $2^n - 1$ , 则称  $p(x)$  为  $n$  次本原多项式。

**定理 4.3 ( $m$  序列充要条件)**  $\{a_i\} \in G(p(x))$ ,  $\{a_i\}$  为  $m$  序列的充要条件是  $p(x)$  是本原多项式。

当我们从移位寄存器的比特输出上考虑问题时，我们通常会采用生成方程 (generating function) 的研究方法，但是如果考虑整个移位寄存器状态时，我们会选用矩阵的方法 (Matrix method)。

## 4.3 序列的伪随机性

我们知道流密码的安全性取决于密钥流的“安全性”，密钥流的安全性就是指其不易被破解，也就是是有良好的随机性，那么什么是“好的随机性”？

我们先看几个基本概念，然后看看 Golomb 定义随机的。

我们先看看游程的概念，对于一个 0、1 序列 0100110001110101111(长度 19)，从左到右，依次有：

0 的 1 游程
1 的 1 游程
0 的 2 游程
1 的 2 游程
0 的 3 游程
1 的 3 游程
0 的 1 游程
1 的 1 游程
0 的 1 游程
1 的 4 游程

总结可知 0 的 1 游程 3 个，1 的 1 游程 2 个，0 的 2 游程 1 个，1 的 2 游程 1 个，0 的 3 游程 1 个，1 的 3 游程 1 个，1 的 4 游程 1 个，可以根据游程计算序列长度：

$$1 \times 3 + 1 \times 2 + 2 \times 1 + 2 \times 1 + 3 \times 1 + 3 \times 1 + 4 \times 1 = 19$$

序列中 0 的个数：

$$1 \times 3 + 2 \times 1 + 3 \times 1 = 8$$

序列中 1 的个数:

$$1 \times 2 + 2 \times 1 + 3 \times 1 + 4 \times 1 = 11$$

**定义 4.4 (序列的自相关函数 (Autocorrelation Function))** GF(2) 上的周期为 T 的序列  $a_i$  的自相关函数为

$$R(\tau) = \frac{1}{T} \sum_{k=1}^T (-1)^{a_k} (-1)^{a_{k+\tau}}, 0 \leq \tau \leq T - 1 \quad (4.4)$$

我们知道  $(-1)^1 = -1, (-1)^0 = 1, (-1)^1(-1)^1 = 1, (-1)^0(-1)^0 = 1, (-1)^1(-1)^0 = -1$ , 所以有  $a_k = a_{k+\tau}$  时,  $(-1)^{a_k}(-1)^{a_{k+\tau}} = 1, a_k \neq a_{k+\tau}$  时,  $(-1)^{a_k}(-1)^{a_{k+\tau}} = -1$ , 也就是说将序列  $a_i$  向后移动  $\tau$  形成一个新序列, 然后这两个序列按上面的公式计算, 数字越大, 表示不同的比特越多, 越小, 表示相同的比特越多。

Golomb 提出了伪随机周期序列应该满足 3 个随机性公设 [17]:

1. 在序列的一个周期内, 0 与 1 的个数相差最多为 1.
2. 在序列的一个周期内, 长为 1 的游程占游程总数的  $\frac{1}{2}$ , 长为 2 的游程占游程总数的  $\frac{1}{2^2}$ , ..., 长为 i 的游程占游程总数的  $\frac{1}{2^i}$ , ..., 且在等长的游程中 0 的游程个数与 1 的游程个数相等。
3. 自相关函数是 0 和一个常数。

第一个公设说明序列  $a_i$  中 0 和 1 出现的概率基本上相同, 第二个公设说明 0 与 1 在序列中每一个位置上出现的概率相同, 公设三表示, 通过对序列与其平移后的序列进行比较, 不能给出其他任何信息。

Golomb 在 "Shift register sequences" 一书的 Chaper III 4.RANDOMNESS PROPERTIES OF SHIFT REGISTER SEQUENCES 中对移位寄存器序列满足这三个性质进行了证明。总结书中结论, 其英文表述如图4.6所示。<sup>4</sup>

Definition (Golomb's randomness postulates)

- ▶ **Balance property.** In every period, the number of zeros is nearly equal to the number of ones (the disparity does not exceed 1, or  $|\sum_{i=0}^{N-1} (-1)^{a_i}| \leq 1$ ).
- ▶ **The run property.** In every period, half of the run have length 1, one fourth have length 2, one eighth have length 3, and so on. For each of these lengths there are the same number of runs of 0's and runs of 1's.
- ▶ **Two level autocorrelation.** The autocorrelation function  $c(\tau)$  is two-valued given by

$$c(\tau) = \begin{cases} N & \text{if } \tau = 0 \pmod{N} \\ k & \text{if } \tau \neq 0 \pmod{N}, \end{cases}$$

where  $k$  is a constant. If  $k = -1$  for  $N$  odd, or  $k = 0$  for  $N$  even, we say that the sequence has the *ideal two level autocorrelation function*.

图 4.6: Golomb's Randomness postulates<sup>5</sup>

从密码系统的角度看, 伪随机序列还应满足以下条件 [17]:

<sup>4</sup>Golomb's Randomness postulates 很多中文书翻译为 Golomb 随机性公设, 在 Bing 的电子词典<https://cn.bing.com/dict> 中 postulate 解释为:

1. <formal>an idea that is an important part of a theory, argument, or explanation
2. an unfounded or disputable unproved assumption
3. a proposition or assumption taken to be self-evident or obvious
4. a demand or request

公设的表述不是很贴切, 因为通常公设是无需去证明的。

1.  $a_i$  的周期相当大。
2.  $a_i$  的确定在计算上时容易的。
3. 由密文和相应的明文的部分信息，不能确定整个密钥序列  $a_i$ 。

**定理 4.4 (m 序列随机性定理)** 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  $2^{n-i-1}$  个, 且 0、1 游程各半。长为 n-1 的 0 游程一个, 长为 n 的 1 游程一个.
- (3)  $a_i$  的自相关函数为:

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

## 4.4 伪随机序列名称的由来 [1]

在抛掷一枚均匀的硬币时, 若出现正面记为 1, 出现背面记为 -1, 我们就得到一个随机的二元序列; 当抛掷的次数足够多时, 所得序列具有下列三条随机特性:

- 1) 序列中 1 的个数与 -1 的个数接近相等。
- 2) 把连在一起的 1(或 -1) 称为“游程”, 其中 1(或 -1) 的个数称为此游程的长度. 列中长度为 1 的游程约占游程总数的  $1/2$ , 长度为 2 的约占  $1/4$ , 长度为 3 的约占  $1/8$ , ..., 在同长度的所有游程中, 1 的游程与 -1 的游程各占半数光景.
- 3) 序列的 x 自相关函数的均值(期望值)在原点为最高, 在离开原点时迅速下降。

以上是真正的二元随机序列的特性, 通常所谓的“伪随机序列”是指具有此三条特性的一部或全部的二元序列。其所以加上个“伪”字, 是因为这些序列尽管表面上满足随机特性, 但实际上他们都是按一定规律形成的“确定性”序列。序列是否随机取决于其产生过程, 仅由其最终形式是无法判定的。在实际应用中, 自相关函数的绝对值在原点之值应远远大于在其他点之值, 才便于在相关检测时把它们区别开来, 所以, 自相关性 3) 在以下特别受到注意。

## 4.5 m 序列的破解

我们看对于 m 序列的流加密, 已知明文攻击的讨论。

假设我们已知密钥是 n 级 LFSR 生成的 m 序列, 我们获得连续的  $n+1$  个长度的信息和对应的明文, 我们可以根据密文  $c_i$  和明文  $m_i$  对计算密钥, 我们知道  $c_i = m_i \oplus k_i$ , 所以有  $c_i \oplus m_i = (m_i \oplus k_i) \oplus m_i = k_i$ , 所以我们可以计算出  $n+1$  长的密钥序列, 我们记为:

$$a_h, a_{h+1}, a_{h+2}, \dots, a_{h+n}$$

我们设密钥序列的特征方程系数分别为  $c_n, c_{n-1}, \dots, c_1$ , 我们有递推式:

$$a_{h+n} = c_n a_{h+n-1} + c_{n-1} a_{h+n-2} + \dots + c_1 a_h$$

用向量表示：

$$a_{h+n} = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix} \begin{bmatrix} a_h \\ a_{h+1} \\ \dots \\ a_{h+n-1} \end{bmatrix}$$

假设我们还可以获取后面的  $n-1$  个密文和对应的明文信息，同理，我们可以计算其对应的密钥序列  $a_{h+n+1}, a_{h+n+2}, \dots, a_{h+2n-1}$ ，我们可以列出下式：

$$\begin{bmatrix} a_{h+n} \\ a_{h+n+1} \\ \dots \\ a_{h+2n-1} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix} \begin{bmatrix} a_h & a_{h+1} & \dots & a_{h+n} \\ a_{h+1} & a_{h+2} & \dots & a_{h+n+1} \\ \dots \\ a_{h+n-1} & a_{h+n} & \dots & a_{h+2n-1} \end{bmatrix}$$

我们记：

$$D = \begin{bmatrix} a_h & a_{h+1} & \dots & a_{h+n} \\ a_{h+1} & a_{h+2} & \dots & a_{h+n+1} \\ \dots \\ a_{h+n-1} & a_{h+n} & \dots & a_{h+2n-1} \end{bmatrix}$$

如果  $D$  存在逆，我们就可以根据我们获得的  $2n$  个数据求出特征方程系数，也就是破解此密码系统。我们可以证明  $D$  的存在 [17]。关于有限域 (Finite Field 或 Galois Field) 上的布尔矩阵 (BOOLEAN MATRICES) 的逆的讨论<sup>6</sup>，可以参考 D. E. Rutherford 在 1962 年的文章 “INVERSES OF BOOLEAN MATRICES”。

求得特征方程后，我们就可以根据计算出来的密钥流得到后续的密钥序列，并且可以根据后续获取的密文序列计算出来明文。

**例 4.9** 假设对手获得流加密系统密文串“101101011110010”和对应的明文串“01100111111001”，并且知道密钥流使用的是 5 级线性反馈寄存器产生的，破解此加密系统。

**解** (1) 利用已知明文和对应密文进行异或，获取密钥流：

cypher	1	0	1	1	0	1	0	1	1	1	1	0	0	1	0
plain	0	1	1	0	0	1	1	1	1	1	1	0	0	0	1
key	1	1	0	1	0	0	1	0	0	0	0	1	0	1	1
note	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$	$a_8$	$a_9$	$a_{10}$	$a_{11}$	$a_{12}$	$a_{13}$	$a_{14}$	$a_{15}$

(2) 因为是 5 级 LFSR，所以我们只用连续取密钥流中 10 个 bit 就可以计算特征方程

<sup>6</sup>注意同一个东西，在发展过程中称谓的演变，这对于去查阅资料，尤其时原始资料时非常有用，比如这里的布尔矩阵，有时人们也会称为有限域的二值矩阵、Galois filed binary matrices、inver of matix over GF(2) 等。

系数，罗列方程如下：

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

即：

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

那么：

$$\begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}^{-1}$$

(3) 下面我们求逆矩阵，这里面需要注意的是加和乘运算是定义在 GF(2) 上的，这里我们用 Gauss-Jordan Elimination 方法（通常翻译为“高斯-约旦消元法”，也有翻译为“高斯-若尔当消元法”、“高斯-约当消元法”<sup>7</sup>），计算过程如下：

首先进行扩展：

$$\left[ \begin{array}{ccccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

因为 GF(2) 上元素只有 0 或 1，所以我们只要选定对角线上元素不为零的，固定这个元素，只需进行行加（在此代数结构中就是异或）运算，就可以将固定下来的这个对角线元素所在列的其他元素全变为零，或者进行行对调，在后面的运算过程中，用  $R_2 \leftrightarrow R_4$  表示第二行和第四行互换，如果固定元素是 (1,1)，那么我们用  $R_1 + R_3$  表示将第一行和第

<sup>7</sup>刚开始看到这个翻译，就有个疑问，Jordan 为什么不翻译为“乔丹”，篮球明星不就是这么翻译的吗？而且通常名字翻译是音译，所以翻译为“乔丹”每毛病。后来查了以下，才知道，数学家 Jordan，全名 Jordan, Marie Ennemond Camille (1838~1922)，是个法国人，一生都在法国，而 Jordan 的法语发音是“若尔当”，原来如此。顺便说一下问什么会在很多地方使用因为，这个用意是便于大家查阅，一是避免翻译上的不同造成的困扰，二是方便大家查阅原始文献。

三行相加，替换第三行，下面我们先进行行对调，调整以下矩阵：

$$\left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \xrightarrow{R_2 \leftrightarrow R_3} \left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

$\xrightarrow{R_2+R_5 \rightarrow R_5}$

$$\left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right]$$

下面我们先固定  $(1,1)$ , 把同列变换为 0:

$$\left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right] \xrightarrow{R_1+R_3, R_1+R_4} \left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right]$$

固定(2,2), 把同列变换为0:

$$\left[ \begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \xrightarrow{R_2 + R_1, R_2 + R_3, R_2 + R_4, R_2 + R_5} \left[ \begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right]$$

固定(3,3), 把同列变换为0:

$$\left[ \begin{array}{cc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{array} \middle| \begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right] \xrightarrow{R_3+R_5} \left[ \begin{array}{cc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array} \middle| \begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{array} \right]$$

$\xrightarrow{R_4 \leftrightarrow R_5}$

$$\left[ \begin{array}{cc|cc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{array} \middle| \begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{array} \right]$$

固定 (4,4), 把同列变换为 0:

$$\left[ \begin{array}{cc|cc|cc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \right] \xrightarrow{R_4+R_1, R_4+R_3} \left[ \begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \right]$$

固定 (5,5), 把同列变换为 0:

$$\left[ \begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \right] \xrightarrow{R_5+R_2, R_5+R_4} \left[ \begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array} \right]$$

现在我们知道:

$$\left[ \begin{array}{ccccc} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{array} \right]^{-1} = \left[ \begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{array} \right]$$

(4) 计算特征方程系数:

$$\begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

通常在解决实际问题时, 比如在 CTF 竞赛时, 我们可以使用 Sagemath 求解 GF(2) 域上矩阵的逆, 上面例题中 Sagemath<sup>8</sup>求解代码如下:

```
a = matrix(GF(2),5,5)
a[0]=[1,1,0,1,0]
a[1]=[1,0,1,0,0]
```

<sup>8</sup>SageMath is a free open-source mathematics software system licensed under the GPL. It builds on top of many existing open-source packages: NumPy, SciPy, matplotlib, Sympy, Maxima, GAP, FLINT, R and many more. Access their combined power through a common, Python-based language or directly via interfaces or wrappers. Mission: Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab. 官网地址<https://www.sagemath.org/index.html>.

Sagemath 创建人是 William Stein(个人网页[https://wstein.org/idx\\_personal.html](https://wstein.org/idx_personal.html), 他是华盛顿大学的一个数学副教授, 其为了教学方便开始集成 Python 的数学包, 形成 Sage)

```

a[2]=[0,1,0,0,1]
a[3]=[1,0,0,1,0]
a[4]=[0,0,1,0,0]
a.is_invertible()
b = matrix(GF(2),5,5)
b=a.inverse()
b
c= matrix(GF(2),1,5)
c[0]=[0,1,0,0,0]
c*b

```

输出结果为：

```

True
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

[1 0 0 1 0]

```

## 4.6 非线性序列

为了使得密钥流更加随机(这样的说法其实是不严谨的，什么是更加随机？序列符合随机性定义？前面讨论过 LFSR 已经满足了，那么更加随机是什么？<sup>9</sup>)，可使用多个 LFSR 来构造序列生成器，这多个 LFSR 的输出可以作为一个非线性组合函数的输入，新的序列生成器产生的序列周期应该尽可能大，但是从理论上可以证明新序列的周期不会大于各 LFSR 周期的乘积。

### 4.6.1 Geff 发生器

三个 LFSR 产生的输出分别为  $a^1, a^2, a^3$ ，在 t 时刻的输出记为  $a_t^1, a_t^2, a_t^3$ ，Geff 的 t 时刻输出为  $b_t = (a_t^1 \wedge a_t^2) \oplus (\neg a_t^1 \wedge a_t^3)$ .

## 4.7 流密码的应用

流密码有很多实际应用，比如用于 DVD(Digital Videodisk) 的加密 CSS(content scramble system)，有广泛用于汽车的 PKE(Passive Keyless entry)，大家注意，并不是所有 PKE 使用的都是流加密，PKE 有些方案使用的分组加密。

---

<sup>9</sup>此处更加随机应该是指周期更长



RC4(Rivest Cipher 4 的缩写)也是一个流密码，但这个流的最小单位(基本单位)是字节，RC4 算法采用  $mod\ 256$  运算产生一个伪随机字节流，然后与明文异或。"RC4 已经成为一些常用的协议和标准的一部分，如 1997 年的 WEP 和 2003/2004 年无线卡的 WPA, 1995 年的 SSL，以及后来 1999 年的 TLS。让它如此广泛分布和使用的主要因素是它不可思议的简单和速度，不管是软件还是硬件，实现起来都十分容易。但是研究者发现存在部分弱密钥，使得子密钥序列在不到 100 万字节内就发生了完全的重复，如果是部分重复，则可能在不到 10 万字节内就能发生重复，因此，推荐在使用 RC4 算法时，必须对加密密钥进行测试，判断其是否为弱密钥。目前发现的主要问题是在无线网络中 IV（初始化向量）不变性漏洞。<sup>10</sup>"

---

<sup>10</sup><https://baike.baidu.com/item/RC4>

# 第 5 章 分组密码 (Block Ciphers)

---

分组密码就是将数据序列划分为一定长度的组，然后在密钥的控制下，将数据分组变换为等长数据序列或者组。

1949 年香农 (C.D. Shannon) 在其论文《保密系统的通信理论》中提到密码设计的基本方法<sup>1</sup>，扩散 (Diffusion)、混淆 (Confusion) 和乘积迭代<sup>2</sup>。

“1948 年,C.E.Shannon 发表了关于通信的数学理论的重要文章，标志着信息论的诞生.1949 年，他在另一篇著名文章“秘密系统的通信理论”中，指出了通信和信息加密的一般特征，以及将信息论用于密码学的基本方法. Shannon 建议交替使用代替和换位两种方法，即他称之为混乱 (confusion) 和扩散 (diffusion) 的过程，破坏对密码系统所进行的各种统计分析，从 1968 年到 1975 年，在 Shannon 这种思想的影响下，许多公司积极着手研制保密性好的乘积密码，其中比较成功的是以 Horst Feistel 为首的 IBM 公司设计小组的产品 LUCIFER 分组乘积密码 (block product cipher)，不久，在 LUCIFER 设计的基础上，以 W. L. Tuchman 为首的 IBM 小组又在探索建立一种保密性更好的乘积密码的途径，他们研究的成果就是著名的 DES。” [2]

## 5.1 Feistel 结构

## 5.2 DES

DES 是 Data Encryption Standard 的所需，中文翻译为“数据加密标准”，其明文分组长度为 64 比特，密钥长为 56 比特，密文长度 64 比特。

DES 的加密算法框图如5.1所示，DES 首先将输入的 64 比特明文进行初始置换，然后将 64 比特分为左 32 比特  $L_i$  和右 32 比特  $R_i$ ，然后进行一轮处理，处理完后，刚开始分出来的右边 32 比特做为下一轮处理的左 32 比特，也就是  $L_{i+1} = R_i$ ，下一轮的右 32 比特为  $R_{i+1} = L_i \oplus f(R_i, K_{i+1})$ ，依次进行 12 轮这样的变换，然后进行逆初始置换，形成最终的密文。

### 5.2.1 初始置换 IP

初始置换输入是 64bit，输出也是 64bit，初始置换表 (如表5.1) 中元素表示 64bit 中的第几个，原始顺序是 1, 2, 3, …, 63, 64，置换表就是调换后的顺序，其顺序是由左至右，由上到下。逆初始置换表 (如表5.2) 也是同样的含义。

<sup>1</sup>香农在其文章引入扩散和混淆只要是为了阻止密码分析中的统计分析方法，可以看此文章中的第 23 部分 “23. STATISTICAL METHODS”

<sup>2</sup>在实际应用中，为了加强密码系统的保密性，常常采用多个密码复合的方法，这就是所谓的乘积密码 (product cipher)，乘积密码是 m 个函数的复合，其中每个函数是一个替换或换位函数。[2]

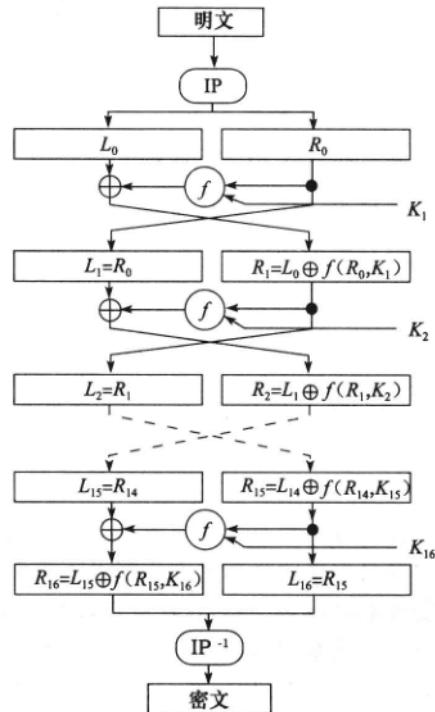


图 5.1: DES 加密过程 [2]

表 5.1: 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

表 5.2:  $IP^{-1}$ 

40	8	48	16	56	24	64	32
39	4	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

对于 64 比特长的数据，其中某个比特经过 IP 变换，再经过  $IP^{-1}$  变换，其位置应该不发生变化，我们取一个位置验证一下。

我们取第 10 个比特，查 IP 表，在第二行第二列，值为 52，也就是第 10 个比特，变换后的位置为第 52 个比特，我们再进行逆变换。查  $IP^{-1}$  表，第 52 比特，第 7 行第 4 列，值为 10，表示此比特变换到第 10 个比特，又回到原来位置。验证了其为逆变换。

初始置换的 C 语言实现参考代码如下：

```
// initial permutation (IP)
const static char IP_Table[64] = {
    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
};

void DES_InitialPermuteData(char* src, char* dst){
    //IP
    int i=0;
    for(i=0;i<64;i++)
    {
        dst[i] = src[IP_Table[i]-1];
    }
}
```

### 5.2.2 轮结构 f 函数

轮结构 (f 函数) 处理过程如图5.2所示。

轮结构中有两个置换 E 和 P，E 置换输入 32bit，输出 48bit，是将其中一些位重复，并进行重排，利用扩展表形式表示 (如表5.3)。P 变换输入 32bit，输出 32bit，只是将比特位进行了重排，如表5.4所示。

表 5.3：位选择表 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

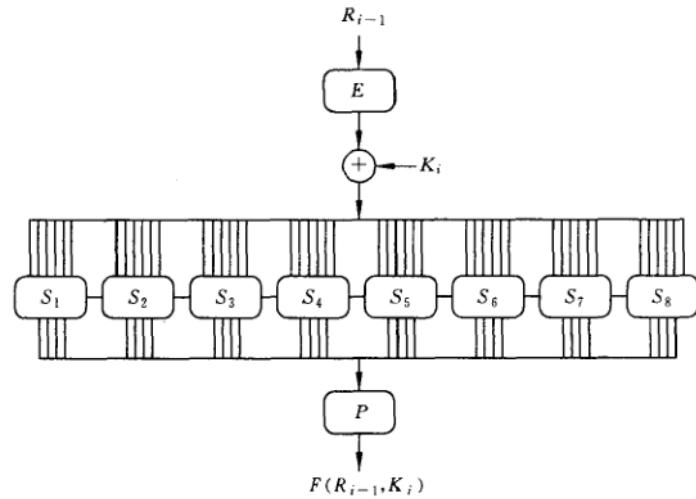


图 5.2:  $f(R_{i-1}, K_i)$  的计算 或 轮结构处理过程 [2]

表 5.4: 换位表 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

轮结构中的 S 盒是 DES 中重要的设计部分，DES 中共有 8 个 S 盒(如图5.3)，每个 S 盒输入 6bit，输出 4bit。对于一个 S 盒的 6bit 输入，第一位和第六位，用来选择 S 盒中的一种代换方法，所以一个 S 盒共定义了四种代换方法，输入的另外 4bit，用来确定其输出数据，我们从 S 定义表中的输出数据区可以看到，输出的最大数据为 15，最小为 0，这与输出是个 4bit 数据相符。

		列															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1$	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
$S_2$	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
$S_3$	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
$S_4$	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
$S_5$	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
$S_6$	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

图 5.3: S 盒 [2]

### 5.2.3 密钥生成

DES 初始密钥 K 是一个 64 位的二进制块，其中有 8 位奇偶校验位，对 DES 的 64 比特初始密钥进行置换选择 1(PC-1)有两个作用：

一是将分别位于 8、16、24、32、40、48、56、64 的校验位去掉。

二是对剩余 56 比特进行换位。

$C_0$  和  $D_0$  是将换位后的 56 比特密钥分为两半，各为 28 比特。 $LS_1, LS_2, LS_9, LS_{16}$  是循环左移 1 位变换， $LS_3, LS_4, LS_5, LS_6, LS_7, LS_8, LS_{10}, LS_{11}, LS_{12}, LS_{13}, LS_{14}, LS_{15}$  是循环左移 2 位变换，通常用一个表来表示。

#### 5.2.3.1 置换选择 1(PC-1)

此置换输入 56bit，输出 56bit，置换表如5.5所示，表的含义同初始置换表。

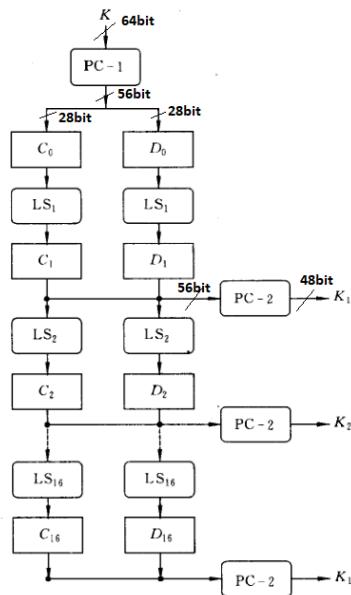


图 5.4: DES 加密中密钥处理过程 [2]

表 5.5: 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

### 5.2.3.2 置换选择 2(PC-2)

此置换输入 56bit，输出 48bit，置换表如5.6所示，表的含义同初始置换表，只是舍掉了 8 位，舍掉的为 54, 43, 38, 35, 25, 22, 18, 9。

表 5.6: 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

### 5.2.3.3 密钥左循环移位

在每一轮的加密中，会将 56bit 密钥分为左右两半，分别对左右两半进行左循环移位，移动几位和第几轮加密有关，其对应关系如表5.7。

表 5.7: 密钥处理过程中左移位数表

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

### 5.2.4 解密

" 在经过所有的代替、置换、异或和循环移动之后，你或许认为解密算法与加密算法完全不同，并且也像加密算法一样有很强的混乱效果。恰恰相反，经过精心选择各种运算，获得了这样一个非常有用的性质：加密和解密可使用相同的算法。

DES 使得用相同的函数来加密或解密每个分组成为可能。两者的唯一不同是密钥的次序相反。这就是说，如果各轮的加密密钥分别是 K1,K2,K3,⋯,K16，那么解密密钥就是 K16,K15,K14,⋯,K1。为各轮产生密钥的算法也是循环的。密钥向右移动，每次移动的个数为 0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1。[14]"

## 5.3 2DES

我们将密钥为 k 的 DES 加密用  $c = E_k(m)$  来表示，那么二重 DES 加密可写为  $c = E_{k_2}(E_{k_1}(m))$ ,  $k_1 \neq k_2$ ，所以 2DES 的密钥长度为  $56 \times 2 = 112$ bit.

5.4 填充

### 5.4.1 NoPadding

这种方法，API 或算法本身不对数据进行处理，加密数据由加密双方约定填补算法。例如若对字符串数据进行加解密，可以补充 0 或者空格，然后 trim。

### 5.4.2 PKCS5Padding

加密时，先把数据字节长度对 8 取余，余数为 m，若  $m > 0$ ，则补足  $8-m$  个字节，字节数值为  $8-m$ ，即差几个字节就补几个字节，字节数值即为补充的字节数，若为 0 则补充 8 个字节的 8。

解密后时，取最后一个字节，值为  $m$ ，则从数据尾部删除  $m$  个字节，剩余数据即为加密前的原文。

### 5.4.3 ISO 10126 padding

ISO 10126 specifies that the padding should be done at the end of that last block with random bytes, and the padding boundary should be specified by the last byte.

Example: In the following example the block size is 8 bytes and padding is required for 4 bytes

#### 5.4.4 Zero padding

All the bytes that are required to be padded are padded with zero. The zero padding scheme has not been standardized for encryption, although it is specified for hashes and MACs as Padding Method 1 in ISO/IEC 10118-1 and ISO/IEC 9797-1.

Example: In the following example the block size is 8 bytes and padding is required for 4 bytes

... | PP PP PP PP PP PP PP PP | PP PP PP PP 00 00 00 00 |

Zero padding may not be reversible if the original file ends with one or more zero bytes, making it impossible to distinguish between plaintext data bytes and padding bytes. It may be used when the length of the message can be derived out-of-band. It is often applied to binary encoded strings as the null character can usually be stripped off as whitespace.

Zero padding is sometimes also referred to as "null padding" or "zero byte padding". Some implementations may add an additional block of zero bytes if the plaintext is already divisible by the block size.

## 5.5 分组密码的运行模式

### 5.5.1 电码本模式 (Electronic Code Book)/ECB 模式

首先对明文进行分组，最后一个分组不够 64bit 需要进行填充。每个明文组独立地用同一密钥加密。 $M = m_1, m_2, m_3, \dots, C = E_k(m_1), E_k(m_2), E_k(m_3), \dots$ ，如图5.5所示。

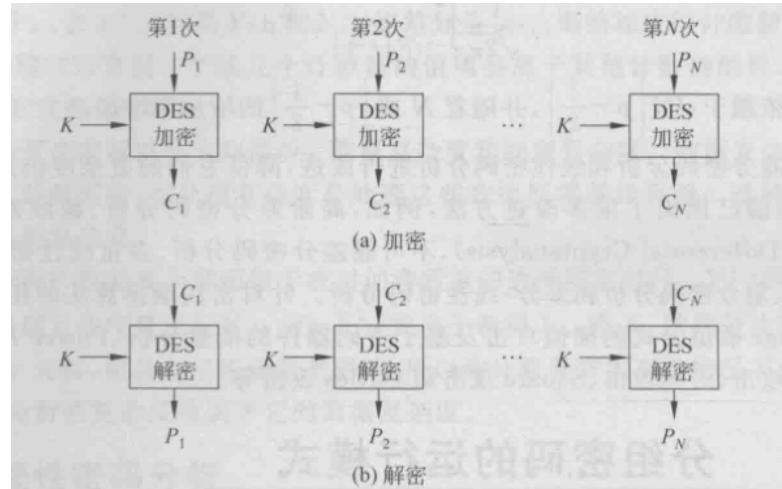


图 5.5: ECB 过程 [17]

### 5.5.2 密码分组链接模式 (Cipher Block Chain)/CBC 模式

首先对明文进行分组，最后一个分组不够 64bit 需要进行填充。加密算法输入时当前明文组与前一个密文组的异或。 $c_i = E_k(c_{i-1} \oplus m_i)$ ,  $c_0 = IV$ ,  $IV$  为一个初始值，如图5.6所示。

### 5.5.3 密码反馈模式 (Cipher Feedback)/CFB 模式

CFB 模式将 DES 转换为流密码进行加密，所以不要进行填充，而且可以实时运行。CFB 模式每次加密单元为  $j$  比特 ( $j \leq 64$ )。

CFB 模式可以表示为：

$$c_1 = p_1 \oplus S_j(E_k(SL_{64}(IV)))$$

$$c_i = p_i \oplus S_j(E_k(SL_{64}(IV \parallel c_1 \parallel \dots \parallel c_{i-1}))), i \geq 2$$

其中：

$S_j(X)$  表示取  $X$  的高  $j$  位

$A \parallel B$  表示将  $A$  和  $B$  按照高低位进行拼接，比如  $11110000 \parallel 10100101 = 1111000010100101 = (F0A5)_{16}$

$SL_j(X)$  表示取  $X$  的低  $j$  位

$E_k$  表示密钥为  $k$  的 DES 加密算法

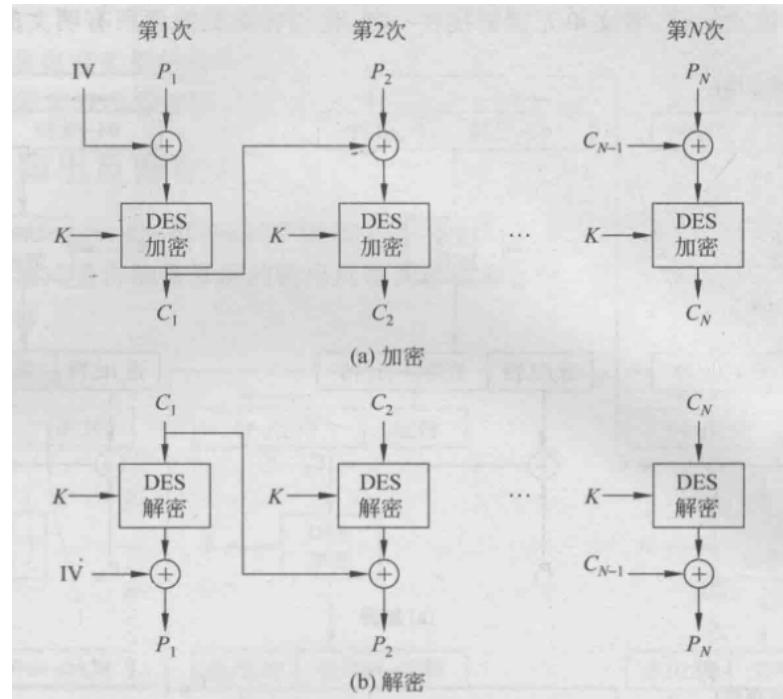
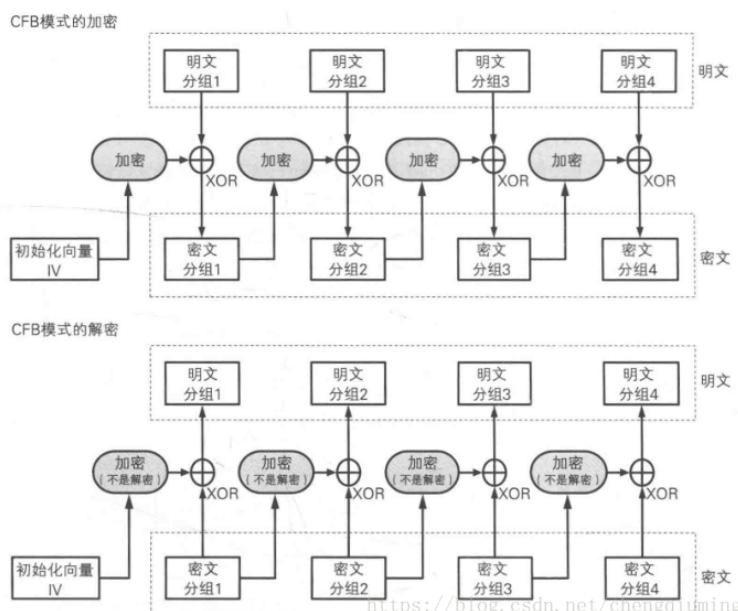


图 5.6: CBC 过程 [17]

$IV$  是一个 64bit 初始向量.

$p_i$  为明文，共有  $j$  比特.

CFB 的过程如图 5.7 所示。

图 5.7: CFB 过程<sup>3</sup>

### 5.5.4 输出反馈模式 (Output Feedback)/OFB 模式

OFB 模式也是转换为流模式处理，但是其流密钥生成独立于密文。 $r_i$  看成流加密的密钥， $j$  为明文  $p_i$  的比特数，OFB 加密过程可以写成：

$$c_i = p_i \oplus r_i, i = 1, 2, \dots$$

密钥流为：

$$\begin{cases} r_1 = S_j(E_k(IV)) \\ r_2 = S_j(E_k(SL_{64}(IV \parallel r_1))) \\ r_3 = S_j(E_k(SL_{64}(IV \parallel r_1 \parallel r_2))) \\ r_4 = S_j(E_k(SL_{64}(IV \parallel r_1 \parallel r_2 \parallel r_3))) \\ \dots \end{cases} \quad (5.1)$$

OFB 加解密框图如图5.8所示。

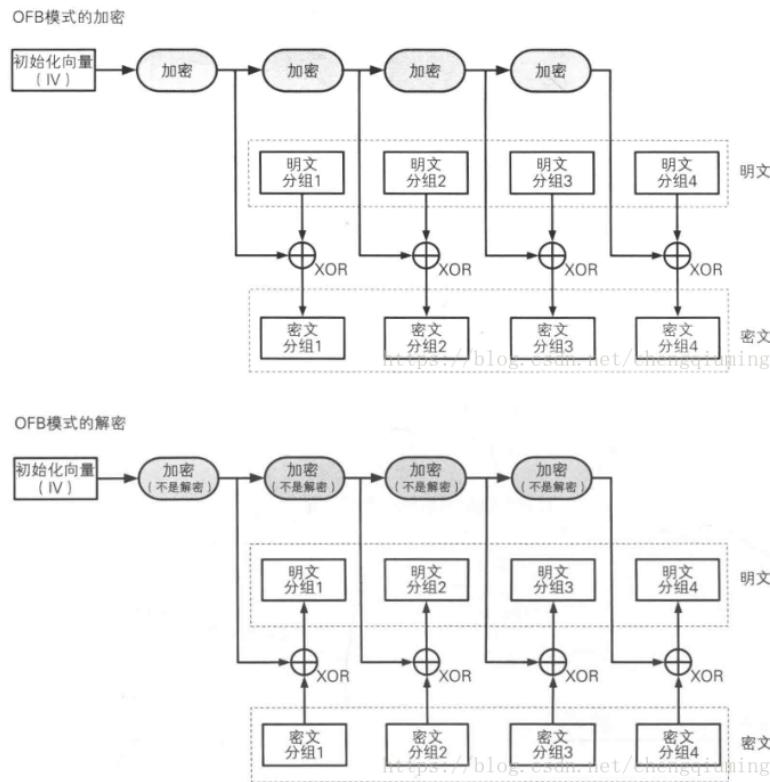


图 5.8: OFB 过程<sup>4</sup>

## 5.6 ZUC

可以参考国密标准 <祖冲之序列密码算法第 1 部分：算法描述 (GM/T 0001.1-2012)> 和 ETSI(European Telecommunications Standard Institute) 的标准 <Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification> 进行 ZUC 的学习。注意以下几点：

- 由大到小。
- 注意问题边界，一个问题，一个问题解决，不要在模块中来回看，可以先把别的模块的当成一个黑盒子。
- 注意原始文献的查看。
- 注意对文献的交叉查阅。

## 5.7 SM4

SM4 是我国无线局域网标准 WAPI 中所采用的分组密码标准，2012 年被我国商业密码标准采用，发表国密标准 <GM/T 0002 SM4 分组密码算法>,2016 年成为国家标准 <GBT32907-2016 信息安全技术-SM4 分组密码算法>。

SM4 的密钥长度和分组长度均为 128 比特, 加密算法与密钥扩展算法都采用 32 轮迭代结构, 以字节 (8 bit) 和字 (32 bit) 为单位进行数据处理。

### 5.7.1 轮函数 F

轮函数 F 的输入为 4 个 32bit 字  $X_0, X_1, X_2, X_3$ , 共 128bit:

$$F(X_0, X_1, X_2, X_3, rk) = X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk)$$

其中 T 为一个合成变换:

$$T(X) = L(\tau(X))$$

$\tau$  为一个 S 盒 ( $2^4 \times 2^4$ ) 非线性变换,  $\tau$  输入为 32 bit, 输出也为 32 bit, 则  $\tau$  变换可以表示为:

$$\tau(a) = S(a)$$

L 为一线性变换, 输入 32 bit a, 输出 32bit:

$$L(a) = a \oplus (a \ll 2) \oplus (a \ll 10) \oplus (a \ll 18) \oplus (a \ll 24)^5$$

### 5.7.2 密钥扩展

SM4 算法加密时输入 128 bit 的密钥, 采用 32 轮迭代结构, 每一轮产生一个 32 bit 的轮密钥。

输入密钥为  $MK = (MK_0, MK_1, MK_2, MK_3)$ , 其中  $MK_i (i = 0, 1, 2, 3)$  为 32 bit。产生的轮密钥 (round key) 记为  $rk_0, rk_1, \dots, rk_{31}$ , 轮密钥的生成算法为:

$$(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$$

$$rk_i = K_{i+4} = K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$$

其中变换  $T'$  与轮函数中的 T 基本相同, 只是将 T 中的 L 变换替换为  $L'$  变换:

$$L'(B) = B \oplus (B \ll 13) \oplus (B \ll 23)$$

---

<sup>5</sup> $\ll$  表示循环左移操作。

在密钥扩展中  $FK_i (i = 0, 1, 2, 3)$  是常数 (fixed key)<sup>6</sup>:

$$FK_0 = (A3B1BAC6)_{16}, FK_1 = (56AA3350)_{16}$$

$$FK_2 = (677D9197)_{16}, FK_3 = (B27022DC)_{16}$$

$CK_i (i = 0, 1, 2, \dots, 30, 31)$  也是一组固定常数, 这组数产生规则为:

$$CK_i = (ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3}), i = 0, 1, 2, \dots, 30, 31$$

$$ck_{i,j} = (4i + j) \times 7 \pmod{256}, j = 0, 1, 2, 3$$

通常在实际算法实现是  $CK_i$  事先计算好, 存在表中使用。

### 5.7.3 加密过程

加密算法采用 32 轮迭代, 每轮使用一个轮密钥, 设输入明文  $X = (X_0, X_1, X_2, X_3)$ , 输入密钥 MK, 产生轮密钥为  $rk_i (i = 0, 1, 2, \dots, 30, 31)$ , 输出密文为 Y, 加密算法伪代码为:  
for i from 0 to 31

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i)$$

$$Y = R(X_{32}, X_{33}, X_{34}, X_{35})$$

其中 R 为反序处理:

$$R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32})$$

### 5.7.4 解密过程

解密算法与加密算法相同, 只是轮密钥顺序想反, 解密伪代码为:

for i from 0 to 31

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_{31-i})$$

$$Y = R(X_{32}, X_{33}, X_{34}, X_{35})$$

此处 X 为密文, Y 为解密后的明文。

---

<sup>6</sup>在下面的表达式中,  $0_{16}$  表示这个数用 16 进制形式表示



# 第 6 章 公钥密码体制 (Public Key Cryptography)

1976 年 W. Diffie 和 N. E. Hellman 发表了著名的文章 “New Directions in Cryptography”，奠定了公钥密码的基础，与传统的密码系统不同的是不需要额外分发密钥的可信信道，加密密钥和解密密钥是本质上不同的，知道一个密钥不能有效地计算出来另外一个。图6.1是传统加密体系和公钥加密体系框图的对比。公钥密码体系中，信息传递者有

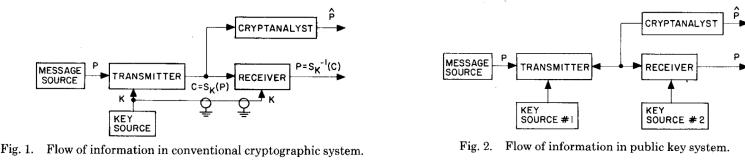


图 6.1：“New Directions in Cryptography”一文中传统加密和公钥加密框图比较

一个私钥 ( $SK$ ,secrete key) 和一个公钥 ( $PK$ ,public key)，公钥  $PK$  是公开的，现在有两个信息传递者 A 和 B(或者是 Alice:/) 和 Bob:/)，其公私钥分别为  $SK_a, PK_a, SK_b, PK_b$ ，这时 A 和 B 之间不需要进行密钥协商或者交换就可以进行保密通信，假如 A 要发信息给 B，A 只需要用 B 的公钥对信息进行加密  $c = E_{PK_b}(m)$ ，然后把加密后的信息  $c$  发给 B,B 收到后，用其私钥进行解密  $m = D_{SK_b}(c)$ ，恢复出  $m$ 。

## 6.1 陷门单向函数 (trap-door one-way function)

公钥密码体系的思想很好理解，但是要构造这样一个体系，并且能够抵抗住密码分析者的攻击却不是一件容易的事情，要求密码分析者从公开的各种信息中无法获得密钥信息和有效的解密办法，Diffie 和 Hellman 在提出公钥密码思想后也给出了如何构造这种密码体系的基本方向或者方法，那就是寻找一个陷门单向函数。

陷门单向函数是这样一类函数族  $y = f(x, k)$ ，其中  $k$  为参数，对于每个  $k$ ， $x$  和  $f(x, k)$  一一对应，给定  $x$  和  $k$ ， $f(x, k)$  很“容易”计算，但是若给定  $y$ (或者说  $f(x, k)$ ) 和  $k$ ，计算  $x$  是“困难”的，这就是说  $f(x, k)$  是个单向函数。

如果存在一个“陷门信息” $k' = d(k)$  及函数  $g(y, k')$ ，使得当  $y = f(x, k)$  时， $x = g(y, k')$ ，并且在给定  $y$  和  $k'$  时， $g(y, k')$  是“容易”计算的，也就是说陷门信息使得给定  $y$  和  $k'$  时，可以“容易”地计算  $x$ 。

仅仅具备单向性的函数可以用来存储口令文件(例如 Hash 函数)，而陷门单向函数则可以用来建立公钥密码系统。

## 6.2 MH 方法 [2]

1978 年，美国斯坦福大学的 R. C. Merkle 和 M. E. Hellman 在 "Hiding information and signatures in trapdoor knapsacks" 一文中，建立了一种基于陷门背包的公钥密码系统。

### 6.2.1 0-1 背包问题 (0-1 Knapsack Problem)

0-1 背包问题是指这样一个问题。有  $n$  件物品和容量为  $m$  的背包，知道每件物品的重量以及价值，每件物品只有一件供你选择，你可以选择装还是不装，找到一种装法，使背包里的物品重量不超过背包容量且价值最大。

我们看看 0-1 背包问题的一个变种，有  $n$  件物品和容量为  $m$  的背包，知道每件物品的重量以及价值分别为  $w_i, v_i$ ，我们先不考虑背包容量，我们只找到一种装法使得物品价值正好为  $S$ ，我们用数学语言描述，或者说构建一个数学模型，就是：

有一个正整数  $S$  和一个背包向量  $V = (v_1, v_2, \dots, v_n)$ ，找到一个二进制向量  $X = (x_1, x_2, \dots, x_n)$ ，使得  $S = \sum_{i=1}^n x_i v_i$ 。

“这个问题是一个著名的 NP(Non-polynomial) 问题，目前解一般背包问题最好的算法需要  $O(2^{n/2})$ ，存储量  $O(2^{n/4})$ 。但是背包问题的困难程度和背包向量的选择关系很大，如果  $V = (1, 2, 4, \dots, 2^{n-1})$ ，给定  $S$  求  $X$  就会很容易” [2]。

上面变种的 0-1 背包问题中还有一种特殊背包问题，我们称为简单背包 (simple knapsack)，简单背包问题中的背包向量  $V$  是一个超上升 (super increasing) 向量，即  $v_i > \sum_{j=1}^{i-1} v_j$ ，对于简单背包问题可以用线性时间解出，也就是说简单背包问题是个 P(Polynomial) 类问题。

下面我们分别举几个例子来看看。

**例 6.1**  $n = 5, S = 14, V = (1, 10, 5, 22, 3)$ ，我们可以检验一下， $X = (1, 1, 0, 0, 1)$  是这个背包问题的一个解，因为容易算得  $XV^T = 1 \cdot 1 + 10 \cdot 1 + 5 \cdot 0 + 22 \cdot 0 + 3 \cdot 1 = 1 + 10 + 3 = 14$ 。[\[9\]](#)

**例 6.2** 我们用穷尽法计算一个背包问题，并且以此方法，看看背包问题的时间复杂度。我

们有,  $n = 4, S = 14, V = (1, 10, 5, 3)$ , 我们列出所有的可能的解向量:

$$(0, 0, 0, 1) \xrightarrow{\text{背包重量}} 3, \quad (0, 0, 1, 0) \xrightarrow{\text{背包重量}} 5 \quad (6.1)$$

$$(0, 0, 1, 1) \xrightarrow{\text{背包重量}} 8, \quad (0, 1, 0, 0) \xrightarrow{\text{背包重量}} 10 \quad (6.2)$$

$$(0, 1, 0, 1) \xrightarrow{\text{背包重量}} 13, \quad (0, 1, 1, 0) \xrightarrow{\text{背包重量}} 15 \quad (6.3)$$

$$(0, 1, 1, 1) \xrightarrow{\text{背包重量}} 18, \quad (1, 0, 0, 0) \xrightarrow{\text{背包重量}} 1 \quad (6.4)$$

$$(1, 0, 0, 1) \xrightarrow{\text{背包重量}} 4, \quad (1, 0, 1, 0) \xrightarrow{\text{背包重量}} 6 \quad (6.5)$$

$$(1, 0, 1, 1) \xrightarrow{\text{背包重量}} 9, \quad (1, 1, 0, 0) \xrightarrow{\text{背包重量}} 11 \quad (6.6)$$

$$(1, 1, 0, 1) \xrightarrow{\text{背包重量}} 14, \quad (1, 1, 1, 0) \xrightarrow{\text{背包重量}} 16 \quad (6.7)$$

$$(1, 1, 1, 1) \xrightarrow{\text{背包重量}} 19, \quad (0, 0, 0, 0) \xrightarrow{\text{背包重量}} 0 \quad (6.8)$$

我们可以看出  $X = (1, 1, 0, 1)$  是这个问题的解.

我们假设背包向量长度为 50, 那么按上面的算法有  $2^{50}$  种可能向量, 如果每检验一个向量用时 0.1 微秒 ( $10^{-6}$  秒), 则最坏情况下需要用时:

$$(10^{-6} \cdot 2^{50}) / 60 / 60 / 24 / 365 \approx 35.7 \text{ 年}$$

**例 6.3** 我们看一个特殊背包向量不是困难问题的例子。我们有,  $n = 8, S = 20, V = (1, 2, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7)$ .

我们计算一下看看是否有解:

$$V = (1, 2, 4, 8, 16, 32, 64, 128) = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)$$

$$v_5, v_6, v_7, v_8 > S \rightarrow v_5 = 0, v_6 = 0, v_7 = 0, v_8 = 0$$

$$v_4 < S \rightarrow v_4 = 1, S = S - v_4 = 20 - 16 = 4$$

$$v_3 > S \rightarrow v_3 = 0$$

$$v_2 = S \rightarrow v_2 = 1, S = S - v_2 = 4 - 4 = 0$$

$$v_1 = 0$$

通过以上计算过程, 我们可以看出有解, 且解向量为  $X = (0, 1, 0, 1, 0, 0, 0, 0)$ , 我们看到这个背包问题的实质其实就是要一个数的二进制表示, 其当然是 P 类问题。

**例 6.4** 我们有,  $n = 5, S = 14, V = (1, 10, 5, 22, 3)$ , 我们对背包向量排序得  $V' = (1, 3, 5, 10, 22)$ , 我们可以检验得知这是一个简单背包向量, 他的解就很容易求得:

$$S < 22 \rightarrow m_5 = 0$$

$$S > 10 \rightarrow m_4 = 1, S = S - 10 = 4$$

$$S < 5 \rightarrow m_3 = 0$$

$$S > 3 \rightarrow m_2 = 1, S = S - 3 = 1$$

$$S = 1 \rightarrow m_1 = 1, S = S - 1 = 0$$



可知此问题有解，且解向量为  $X = (0, 1, 0, 1, 0)$ .

## 6.2.2 MH 方法

### 6.2.2.1 陷门单向函数构造

简单背包问题是一个 P 类问题，MH 方法的本质是将简单背包变成一个陷门背包，如果不知道陷门信息，就是一个难解问题，知道陷门，就是一个容易解的问题。

下面我来看看 MH 实现方法。

首先选择一个简单背包向量  $V = (v_1, v_2, \dots, v_n)$ ，给定  $S$ ，我们容易求得  $X = (x_1, x_2, \dots, x_n)$ ，使得  $S = \sum_{i=1}^n x_i v_i$ 。

然后我们选择一个整数  $t$ ， $t$  满足  $t > \sum_{i=1}^n v_i$ ，选择一个与  $t$  互素的整数  $p$ ，计算  $p^{-1}, pp^{-1} = 1 \pmod{t}$ ，我们计算一个新向量  $V' = pV \pmod{t}$ ，新向量中的元素伪随机分布，故单纯看  $S = XV'$  是个困难问题，但是如果我们知道  $p^{-1}, m$ ，我们再可以把  $S = XV'$  进行如下转换：

$$S' = p^{-1}S \pmod{t} = p^{-1}XV' \pmod{t} = p^{-1}XpV \pmod{t} = p^{-1}pXV \pmod{t} = XV \pmod{t} \quad (6.9)$$

因为  $t > \sum_{i=1}^n v_i$ ，所以有  $S' = XV$ .

我们知道简单背包问题是个容易计算的问题，把所以我们把求  $S = XV'$  这个一般背包问题，转化为简单背包问题  $S' = XV$ ，其中  $S' = p^{-1}S \pmod{t}, V = p^{-1}V'$ ，求得  $X$ 。

从上面的分析可以看到，把这个一般背包问题转化为简单背包问题的关键是知道  $t$  和  $p^{-1}$ ，所以  $(t, p^{-1})$  就是私钥。

### 6.2.2.2 加密方法

对于 MH 方法，用户公布困难背包向量  $V' = (v_1, v_2, \dots, v_n), V'$  是公钥， $(V, t, p^{-1})$  保密， $(t, p^{-1})$  为私钥，由于  $V$  可以根据私钥和公钥计算出来，也可以不做存储或者考虑。

我们假设 A 要向 B 发送一个保密信息，这句话意味着，A 发送的信息只想让 B 看到，A 获得 B 公开发布的公钥  $V'$ ，然后将信息预处理为长度为 n 的二进制块  $X = (x_1, x_2, \dots, x_n)$ <sup>1</sup>，加密过程为：

$$C = E_{V'}(X) = \sum_{i=1}^n (x_i v'_i)$$

A 将密文 C 发送给 B，B 收到 C 后的解密过程为，首先计算  $C' = p^{-1}C \pmod{t}$ ，然后求解简单背包问题  $C' = XV$ 。

**例 6.5** 我们看一个背包加密的简单的例子，我们选简单背包向量  $V = (1, 3, 5, 10)$ ，选  $t >$

<sup>1</sup> 其实就是加密值的二进制编码

$\sum_{i=1}^4 v_i = 19$ , 此处我们选  $t = 20$ , 选择一个与  $t$  互素的数  $p$ , 我们选  $p = 7$ , 那么  $p^{-1} \pmod{20} = 3$ , 我们生成一个“困难背包” $V' = pV \pmod{20} = (7, 1, 15, 10)$ , 发布公钥  $(V', t)$ , 保存好私钥  $(V, p^{-1}, t)$ <sup>2</sup>。我们给定原文为 13, 计算密文, 并写出接受放解密过程。

**解** (1) 计算密文: 13 的二进制向量为  $(1, 1, 0, 1)$ , 计算  $(1, 1, 0, 1)V'^T = 18$ , 密文为  $c = 18$ .  
(2) 解密:  $c' = p^{-1}c \pmod{20} = 3 \times 18 \pmod{20} = 14$ , 计算  $X$ ,  $c' = XV^T$ ,  $X = (1, 1, 0, 1)$ ,  $X$  看做一个二进制向量, 解密后的值为  $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 = 13$ .

### 6.2.2.3 签名方法

我们假设 A 要向 B 发送一个信息 X, B 收到这个信息后, 要有“足够的理由”相信, 信息就是 A 发送的。

我们用 MH 方法来构造一个 MH 签名算法。

A 要发送信息 M 给 B, A 用自己的私钥信息 V 对信息进行“加密”(其实这里叫“加密”已经不合适了, 通常我们会说 A 对信息进行“签名”):

$$C = E_V(X) = \sum_{i=1}^n (x_i v_i)$$

然后将 M 和 C(签名信息), 一起发送给 B, 记为:

$$A \rightarrow B : (M, C)$$

B 首先获得了 A 的公钥信息  $V'$ , 在收到这个信息后用, 可以验证以下等式是否成立:

$$M \stackrel{?}{=} E_{V'}(X)$$

成立表示 M 是 A 发送的, 并且没有经过第三方篡改, 不成立则证明这个信息不是 A 要发送的。从而形成一个电子签名。

**例 6.6** 我们依然以前一个例子为基础看看 MH 签名过程:

私钥:  $(V, p^{-1}, t)$ ,  $V = (1, 3, 5, 10)$ ,  $p^{-1} = 3$ ,  $t = 20$

公钥:  $(V', t)$ ,  $V' = (7, 1, 15, 10)$ ,  $t = 20$

要发送的消息  $m = 13$ (十进制), 写出签名过程。

**解** 我们首先用私钥计算签名值:

$$t_1 = p^{-1}m \pmod{t} = 3 \times 13 \pmod{20} = 19$$

$t_1 = XV^T \rightarrow 19 = X(1, 3, 5, 10)^T$ , 求解 X, 得  $X = (1, 1, 1, 1)$ , X 看做一个二进制向量, 可得签名值 Sig 为 15.

将消息 13 和签名值 15 一起发送给接收方, 接收方进行如下验证:

签名值 Sig 的二进制表示为  $X' = (1, 1, 1, 1)$ , 用公钥计算  $m' = (1, 1, 1, 1)V'^T = (1, 1, 1, 1)(7, 1, 15, 10)^T =$

<sup>2</sup>私钥也可以写成  $(p^{-1}, t)$ , 因为可以  $V = p^{-1}V'$

$33 \pmod{20} = 13$ , 因为  $m = m'$ , 所以我们可以判定此消息是知道私钥的人发送的, 并且消息没有被修改。

从上面的例子看, 好像 MH 方法可以构成公钥方式的数字签名(如同 RSA 一样), 其实不然, “陷门背包公开密钥密码系统和 RSA 不同, 不能实现数字签名, 这是因为: 它的加密变换不是整个信息空间上的映成函数 (onto function)<sup>3</sup>, 因此某些信息 (实际上大多数信息) 不能先解密后加密, A. Shamir 建立了一种不能用于加密的陷门背包数字签名系统<sup>4</sup>, 这种经典的方法现在虽然不再应用, 但其构思有许多方面值得我们借鉴。” [2]

### 6.2.3 背包问题求解的发展

1978 年, 斯坦福大学的 Merkle 和 Hellman 在“Hiding Information and Signatures in Trapdoor Knapsacks”文章中, 提出利用背包问题设计公钥密码系统的方法。

" 1979 年, R. Schroeppel 和 A. Shamir 发表了一种求解一般背包问题的算法<sup>5</sup>, 所需时间为  $O(2^{n/2})$ , 存储量为  $O(2^{n/4})$ , 当  $n = 100$  时,  $2^{50} \approx 10^{15}$ , 因此一个处理机大约要用 11574 天才能算出一个解, 但是当 1000 个处理机并行工作时, 大约 12 天就可以求出解(假定一个处理机每天能执行  $8.64 \times 10^{10}$  条指令), 所以, Merkle 和 Hellman 原先的建议不甚合适, 至少应当取  $n=200$ , 此时  $2^{100} \approx 10^{30}$ , 方可保证 MH 公开密码系统的安全性。 "[2].

背包问题一直被持续研究, 2010 年 Nick Howgrave-Graham 和 Antoine Joux 发表了一篇文章 <New generic algorithm for hard knapsacks>, 将密度为 1 背包问题的求解时间降为  $O(2^{0.3113n})$ (见图6.2), 从这篇文章的参考文献中, 我们也可以看到 Shamir 对这个问题的持续研究(见图6.3), 1979、1981、1983 本别有成果发表。

---

<sup>3</sup>Any function is said to be onto function if, in the function, every element of codomain has one or more relative elements in the domain. Onto function is also popularly known as a surjective function.

<sup>4</sup>卿斯汉老师的书里没有给出参考文章, 从 Denning 的文章 “Digital Signatures with RSA and Other Public Key Cryptosystems ” 中给出的参考文献, 可以推断, 这应该是个技术报告 “Shamir, A. A fast signature scheme. Tech. Rept. MIT/LCS/TM-107, MIT Lab. for Computer Science, Cambridge, Mass., July 1978.”

<sup>5</sup>文章名为: A  $T = O(2^{n/2})$ ,  $S = O(2^{n/4})$  algorithm for certain NP – complete problems.

## New generic algorithms for hard knapsacks

Nick Howgrave-Graham<sup>1</sup> and Antoine Joux<sup>2</sup>

<sup>1</sup> 35 Park St, Arlington, MA 02474

[nickhg@gmail.com](mailto:nickhg@gmail.com)

<sup>2</sup> DGA and Université de Versailles Saint-Quentin-en-Yvelines  
UVSQ PRISM, 45 avenue des États-Unis, F-78035, Versailles CEDEX, France  
[antoine.joux@m4x.org](mailto:antoine.joux@m4x.org)

**Abstract.** In this paper<sup>3</sup>, we study the complexity of solving hard knapsack problems, i.e., knapsacks with a density close to 1 where lattice-based low density attacks are not an option. For such knapsacks, the current state-of-the-art is a 31-year old algorithm by Schroeppel and Shamir which is based on birthday paradox techniques and yields a running time of  $\tilde{O}(2^{n/2})$  for knapsacks of  $n$  elements and uses  $\tilde{O}(2^{n/4})$  storage. We propose here two new algorithms which improve on this bound, finally lowering the running time down to  $\tilde{O}(2^{0.3113n})$  for almost all knapsacks of density 1. We also demonstrate the practicality of these algorithms with an implementation.

图 6.2: 2010 年公开发表的背包算法研究

21. Richard Schroeppel and Adi Shamir. A  $T = O(2^{n/2}), S = O(2^{n/4})$  algorithm for certain NP-complete problems. In *FOCS*, pages 328–336, 1979.
22. Richard Schroeppel and Adi Shamir. A  $T = O(2^{n/2}), S = O(2^{n/4})$  algorithm for certain NP-complete problems. *SIAM Journal on Computing*, 10(3):456–464, 1981.
23. Adi Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 279–288, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.

图 6.3: Shamir 对背包算法的持续研究

## 6.3 RSA

1978 年，美国的麻省理工学院的 R. L. Rivest, A. Shamir 和 M. Adleman 在其论文 "A method for obtaining digital signatures and public-key cryptosystems" 中提出了一种实现 Diffie —— Hellman 公钥思想的方法，简称为 RSA 方法。

1982 年，Rivest、Shamir 和 Adleman 三个创始人正式成立了 RSA Data Security 公司，1989 年刚刚发展起来的互联网采用了 RSA 加密软件，1994 年 RSA 源代码在互联网上被匿名公布，RSA 公司成立后不断发展，目前已是一家顶级的安全服务公司。<sup>6</sup>

### 6.3.1 RSA 依赖的困难问题

RSA 依赖的困难问题是：整数分解为素数是个困难问题，有时候我们也称“整数分解是个困难问题”、“素分解是个计算上困难的问题”。

但是如果有一组素数，计算出这组素数的合数，却是一个容易的问题。

需要注意的是，虽然 RSA 计算困难问题基于的“整数分解”，但是 RSA 破解并不等价于“整数分解问题”，我们从下面 RSA 方法的介绍中可以知道，我们只是说如果能够获得整数的分解，那么就可以破解 RSA，并没有证明破解 RSA 与“整数分解问题”计算复杂性等价。

<sup>6</sup>信息来源于<https://blog.csdn.net/wangjianno2/article/details/19262887>

### 6.3.2 RSA 加密方法

1. 用户选取两个不同的大素数  $p$  和  $q$ , 计算  $n = pq$ , 我们知道  $\phi(n) = (p-1)(q-1)$ .
2. 选一个正整数  $d$ , 满足  $\gcd(d, \phi(n)) = 1$ .
3. 计算  $d$  模  $\phi(n)$  的逆元  $e$ , 也就是说  $ed = 1 \pmod{\phi(n)}$ .
4.  $(e, n)$  做为公钥,  $(d, n)$  做为私钥.
5. 加密: RSA 是分组加密, 取一个明文分组  $M, 0 \leq M \leq n-1$ <sup>7</sup>,  $C = E_e(M) = M^e \pmod{n}$ .
6. 解密:  $M' = D_d(C) = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n} = M \pmod{n}$

已知公钥  $(e, n)$ , 如果能够计算出  $n$  的素分解, 也就是知道  $p$  和  $q$ , 那么就能很容易计算出  $d$ , 可见 RSA 算法依赖的困难问题就是分解因子问题。

通常在算法具体实现中, 我们会在产生公私钥后, 销毁  $p$  和  $q$ , 如果算法设计有漏洞, 泄露了  $p$  和  $q$ , 这是有可能造成“侧信道攻击”(Side Channel Attack, 简称 SCA)<sup>8</sup>。

#### 6.3.2.1 RSA 解密过程证明

下面我们证明 RSA 的正确性, 也就是解密过程可以获得明文  $M$ 。

1.  $M$  为明文,  $C$  为密文, 公钥  $(e, n)$ .
2. 加密:  $C = M^e \pmod{n}$ .
3. 解密:  $M' = C^d \pmod{n} = m^{ed} \pmod{n}$ .
4.  $ed = 1 \pmod{\phi(n)} \Rightarrow ed = k\phi(n) + 1$
5.  $M' = C^d \pmod{n} = M^{ed} \pmod{n} = M^{k\phi(n)+1} \pmod{n}$
6.  $M$  与  $n$  互素, 由 Euler 定理  $M^{\phi(n)} = 1 \pmod{n}$ , 可得  $M^{k\phi(n)+1} \pmod{n} = M^{k\phi(n)}M \pmod{n} = M \pmod{n}$ .
7.  $M$  与  $n$  不互素, 也就是说  $\gcd(M, n) \neq 1$ , 我们知道  $n$  的素分解为  $p, q$ , 那也就是意味着  $M$  是  $p$  或  $q$  的倍数, 不失一般性, 我们假设是  $p$  的倍数, 也就是说  $M = tp, t \in \mathbb{N}$ , 此时  $M$  一定不是  $q$  的倍数, 如果  $M$  是  $q$  的倍数, 那么  $M$  就是  $n$  的倍数, 我们知道模  $n$  其实是整个运算空间的边界, 也就是  $M < n$ , 显然如果  $M$  就是  $n$  的倍数, 与  $M < n$  矛盾, 所以如果,  $M$  是  $p$  的倍数, 那么一定有  $\gcd(M, q) = 1$ .
8.  $\gcd(M, q) = 1$ , 由 Euler 定理  $M^{\phi(q)} = 1 \pmod{q}$ , 可得,

$$\begin{aligned} & M^{k\phi(n)} \pmod{q} \\ &= M^{k\phi(p)(q)} \pmod{q} \\ &= [M^{k\phi(q)}]^{(\phi(p))} \pmod{q} \\ &= 1 \pmod{q} \end{aligned}$$

<sup>7</sup>对与实际加密的数据可以按照一定方法把数据变换为符合合格要求, 比如将其变为  $n$  进制编码

<sup>8</sup>侧信道攻击, 也有称为边信道攻击的, 但是我比较喜欢侧信道攻击, 因为, 好听, 通常来讲, 侧信道攻击就是利用加密软件或硬件运行时通过各种“途径”产生的各种泄漏信息进行攻击。有针对密码算法的计时攻击(Timing attacks)、能量攻击(power analysis attacks)、电磁分析(EM-attacks)攻击等, 也有一些比如针对键盘敲击内容的侧信道攻击, 比如分析声音、电磁攻击等等, 所以广义上的侧信道攻击, 脑洞大开, 这种攻击的主要根源在于我们通常在设计过程中, 基本上考虑在“正常信道”上对攻击的对抗, 而因为侧信道种类繁多, 很少有系统考虑, 或者及时考虑也会有遗漏

, 也就是说, 存在整数  $r$ ,  $M^{k\phi(n)} = 1 + rq$ , 两边同乘  $M = tp$ , 有,  $M^{k\phi(n)+1} = M + rqt = M + rtn$ , 从上式我们可得:  $M^{k\phi(n)+1} \pmod{n} = M \pmod{n}$ . [证毕]

### 6.3.3 RSA 简单示例

说一个 RSA 系统是多少位的, 通常是指模  $n$  转换为二进制后是多少位, 因为  $n$  的大小限定了运算空间的大小, 体现了破解的难度, 也可以说体现了密码系统的安全性。下面我们给出一个 12 位 RSA 的加密解密方法<sup>9</sup>。

#### 6.3.3.1 生成公私钥对

RSA 是非对称加密, 有公钥和私钥, 公钥公开给加密方加密, 私钥留给自己解密, 是不公开的。

1. 随机选两个素数, 用  $p$ 、 $q$  来代替 (素数的数值越大, 位数就越多, 可靠性就越高。假设我们取  $p = 47$ ,  $q = 59$ 。)
2. 计算这两个素数的乘积,  $n = p \times q = 47 \times 59 = 2773$ ,  $n$  的长度就是公钥长度。2773 写成二进制是 101011010101, 一共有 12 位, 所以这个密钥就是 12 位。实际应用中, RSA 密钥一般是 1024 位, 重要场合则为 2048 位。
3. 计算  $n$  的欧拉函数  $\phi(n)$ ,  $\phi(n) = (p-1)(q-1)$ ,  $\phi(2773) = (47-1) \times (59-1) = 46 \times 58 = 2668$ .
4. 随机选择一个整数  $e$ ,  $1 < e < \phi(n)$ , 且  $e$  与  $\phi(n)$  互素 (我们知道, 此时  $e^{\phi(n)} \equiv 1 \pmod{n}$ )。例如我们在 1 到 2668 之间, 随机选择了 17,  $e = 17$ 。
5. 计算  $e$  对于  $\phi(n)$  的模乘法逆元  $d$ , 当  $\gcd(e, \phi(n)) = 1$ ,  $ed \equiv 1 \pmod{\phi(n)}$   $\Rightarrow d = (1 + k\phi(n))/e$ ,  $k \in \mathbb{Z}$ , 代入各值,  $d = (1 + 2668)/17$ , 可以依次给  $k$  赋值, 取  $d$  为整数的序偶, 得到一系列  $(k, d)$ ,  $(1, 157)$ 、 $(18, 2825)$ 、 $(35, 5493)$  ..., 随机选一个序偶, 比如  $(1, 157)$ , 也就是  $d=157$ .
6. 将  $n$  和  $e$  封装成公钥,  $n$  和  $d$  封装成私钥, 即公钥为:  $n = 2773$ ,  $e = 17$ , 私钥为:  $n = 2773$ ,  $d = 157$ 。

#### 6.3.3.2 用公钥加密字符串

假设我们加密一个字符 "A", 首先字符要用数值表示 (这就是编码, 信源编码), 一般用 Unicode 或 ASCII 码表示, 此处我们用 ASCII 码表示, "A" 的 ASCII 码十进制为 65(十六进制 0x41), 我们用  $m$  来代替明文 (message),  $c$  来代替密文 (cipher),  $m = 65$ , RSA 加密公式:  $m^e \equiv c \pmod{n}$ , 代入各值  $65^{17} \pmod{2773} \equiv 6, 599, 743, 590, 836, 592, 050, 933, 837, 890, 625 \pmod{2773} \equiv 332 \pmod{2773}$ ,  $c = 332$

---

<sup>9</sup>此节采用的例子素材来源于 <https://www.jianshu.com/p/4e302869d057>

### 6.3.3.3 用私钥解密密文

RSA 解密公式:  $c^d \equiv m \pmod{n}$ , 代入各值,  $c^d \equiv 332^{157} \equiv 6.5868707484014117339891253968203e+395 \equiv 65 \pmod{2773}$ ,  $m = 65$ .

### 6.3.3.4 用私钥简单签名字串

RSA 签名消息  $m$ ,  $s = m^d \pmod{n} = 65^{157} \pmod{2773} = 126$ , 签名值  $S$  为“126”, 将  $m||s$  发送给接收方。

### 6.3.3.5 用公钥验证字符串

接收到  $m||s$  后, 计算  $m' = s^e \pmod{n} = 126^{17} \pmod{2773} = 65$ , 当  $m' = m$ , 我们推断出签名值是只有知道私钥的人计算出来的, 并且接收到的  $m$  没有被修改。

## 6.3.4 公私钥对的生成

RSA 的理论很容易看明白, 但是当您坐下来准备实现一个 RSA 算法时, 我这里说的实现, 不是利用 OpenSSL 等库来实现, 这些库基本的 RSA 加密过程函数已实现好了, 你只需要了解 RSA 的过程原理依次调用就可以, 我们这里的实现, 是指利用一些基础的数学函数库来实现 RSA 算法。

而在具体做这些底层实现时, 会有很多细节问题需要考虑, 首先确定好 RSA 系统的位数后, 大素数  $p$  和  $q$  如何产生? 大数如何计算?  $e$  如何产生? 等等, 整个过程是有很多细节需要考虑, 而且在密钥产生过程中, 如果考虑不够全面, 对攻击者来说会有“漏”可捡, 而这些要求, 你对一个编程者提出来, 显然是不公平的, 他很难达到这样的要求, 道理很简单, 他不是数学家, 不是密码学家, 那么怎么解决? 标准。这也就是说你可以看到大量的 NIST FIPS 和 SP 标准, 以及 RFC、ANSI 标准在详细描述实现的细节, 可以去看看文档"OpenSSL FIPS 140-2 Security Policy", 这里面提到 OpenSSL 在实现时参考的一些标准。实现细节大家可以去看这些标准, 以及其他资料。

下面我引用一个帖子里<sup>10</sup>的一段原文, 对于 RSA 密钥生成会有一个宏观的了解。

The algorithm for generating an RSA key pair boils down to finding a set of big, prime numbers, that fulfil some algebraical properties and that are of appropriate size. If you need a 2048 bit RSA key, you will typically look for 2 prime number, each having a rough length of 1024 bits.

The process of finding a prime number is trial-and-error: you randomly pick an integer of appropriate size, and test if it is prime. If it is not, you retry.

<sup>10</sup><https://stackoverflow.com/questions/18264314/generating-a-public-private-key-pair-using-an-initial-key>

In the real world, the random generator that drives the algorithm is a deterministic PRNG which is seeded with a secret of appropriate entropy (e.g. 128 bits of true randomness).

In your case, the PRNG seed can be derived from a user secret or even from another key (provided it is secret of course). Derivation should be performed with a salted KDF like HKDF, PBKDF2, etc.

You don't specify which crypto library you use: whatever it is, you must be clear on how it draw randomness and how to define the seed of the PRNG.

从上面的讨论中，我们也可以体会到随机数生成在加密体制中是非常重要的功能，他不仅仅在流加密中。

## 6.4 ECC(Elliptic Curve Cryptography)

### 6.4.1 基本概念

椭圆曲线 (elliptic curve) 是指由 Weierstrass 韦尔斯特拉 (中文翻译有韦尔斯特拉、魏尔斯特拉斯) 方程确定的平面，韦尔斯特拉方程为:  $E : y^2 + axy + by = x^3 + cx^2 + dx + e$ , E 是 Elliptic curve 的缩写，表示这个方程描述了一个椭圆曲线，其中 a, b, c, d 和 e 属于域 F, F 可以是有理数域、复数域、有限域，密码学中通常采用有限域。

下面我们用 Sagemath 绘制几个实数域上的椭圆曲线。

**例 6.7** 椭圆曲线:  $y^2 = x^3 - 2x$

#### sagemath: 椭圆曲线

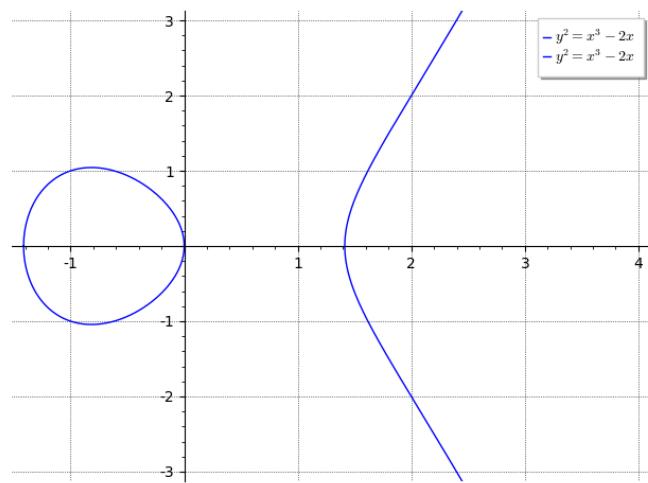
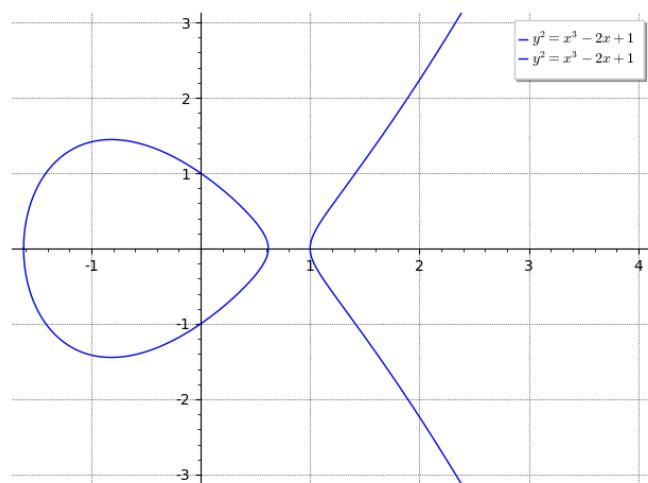
```
# y^2=x^3-2x
p= plot(EllipticCurve([0,0,0,-2,0]),gridlines='true', xmin=-4, xmax=4,
ymin=-3, ymax=3,legend_label='$y^2=x^3-2x$')
show(p)
```

**例 6.8** 椭圆曲线:  $y^2 = x^3 - 2x + 1$

#### sagemath: 椭圆曲线

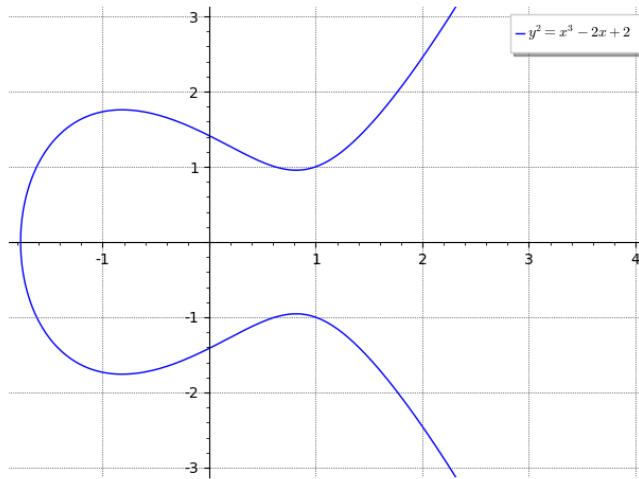
```
# y^2=x^3-2x
p= plot(EllipticCurve([0,0,0,-2,1]),gridlines='true', xmin=-4, xmax=4,
ymin=-3, ymax=3,legend_label='$y^2=x^3-2x+1$')
show(p)
```

**例 6.9** 椭圆曲线:  $y^2 = x^3 - 2x + 2$

图 6.4: 椭圆曲线  $y^2 = x^3 - 2x$ 图 6.5: 椭圆曲线  $y^2 = x^3 - 2x + 1$

## sagemath: 椭圆曲线

```
# y^2=x^3-2x+2
p= plot(EllipticCurve([0,0,0,-2,2]),gridlines='true', xmin=-4, xmax=4,
ymin=-3, ymax=3,legend_label='y^2=x^3-2x+2')
show(p)
```

图 6.6: 椭圆曲线  $y^2 = x^3 - 2x + 2$ 

通过定义恰当的“加法”运算，椭圆曲线上的点全体构成一个加法群，正因为椭圆曲线存在加法结构，所以它包含了很多重要的数论信息。下面我们看看椭圆曲线上的加法群的构建。

- 单位元： $O$  为单位元，椭圆曲线上的所有点  $P$  有  $P + O = P$ ， $O$  也是一个椭圆曲线上的一点，是一个无穷远的点。
- 逆元：对点  $P = (x, y)$ ，其加法逆元为  $(x, -y)$ ，记为  $-P$ ， $P + (-P) = O$ ，由此也可以定义减法  $P - P = O$
- 加法：对于两个不同且不互逆的点  $P, Q$ ，我们画一条通过  $P, Q$  的直线，与椭圆曲线交于一点，这个交点是唯一的（除非所做的直线是  $P$  或  $Q$  的切线），此交点的逆元为  $R$ ，定义  $P + Q = R$ 。加法定义如图6.7所示。
- 倍数：点  $P$  的倍数定义为，在  $P$  点做椭圆曲线的一条切线，设切线与椭圆曲线交于一点， $R$  为此交点的逆元，定义  $2P = P + P = R$ ，一般将  $\overbrace{Q + Q + \dots + Q}^{n \text{ 个 } Q}$  记为  $nQ$ 。可以证明以上定义的加法运算具有交换律和结合律等一般性质。

## 6.4.2 有限域上的椭圆曲线

为了简单起见，我们通常考虑在有限域  $GF(p)$  上的椭圆曲线， $p$  为大于 3 的素数，在有限域  $GF(p)$  上的曲线  $y^2 = x^3 + ax + b$  ( $a, b \in GF(p), 4a^3 + 27b^2 \neq 0$ ) 称为有限域上的椭圆曲线，通常记为  $E_p(a, b)$ 。

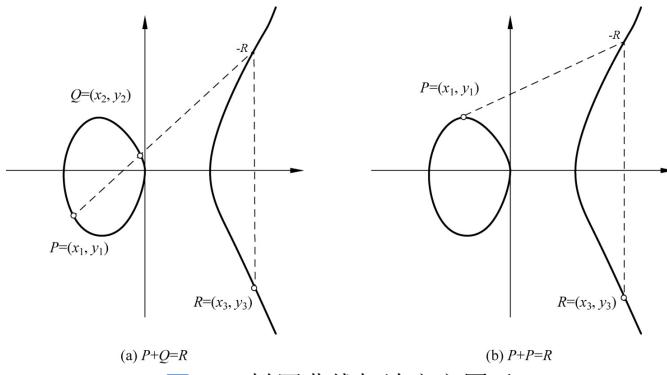


图 6.7: 椭圆曲线加法定义图示

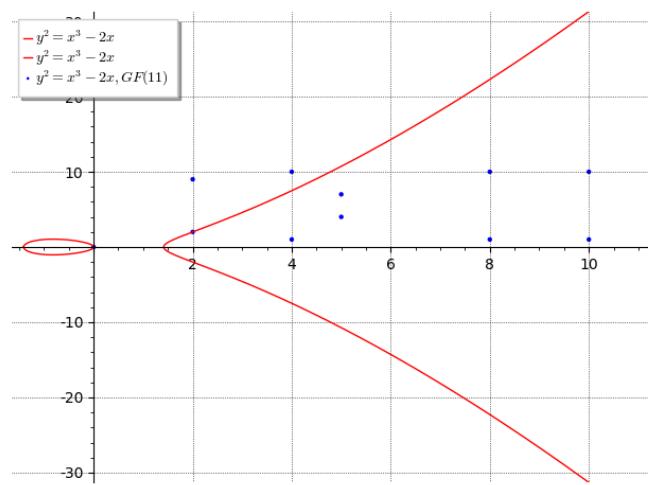
椭圆曲线的定义也要求曲线是非奇异的(即处处可导的)。几何上来说,这意味着图像里面没有尖点、自相交或孤立点。代数上来说,这成立当且仅当判别式 $\delta = 4a^3 + 27b^2$ 不等于0,这里主要是满足其可导性。

我们看看同一方程在不同域上的曲线。

**例 6.10** 椭圆曲线:  $y^2 = x^3 - 2x$  在有限域  $GF(11)$  和实数域上表示的曲线。

### sagemath: 椭圆曲线

```
# y^2=x^3-2x在有限域GF(11)
p=plot(EllipticCurve(GF(11),[0,0,0,-2,0]),gridlines='true',
xmin=-11, xmax=11, ymin=-30, ymax=30,
legend_label='$y^2=x^3-2x, GF(11)$')
# y^2=x^3-2x在实数域
p+=plot(EllipticCurve([0,0,0,-2,0]),gridlines='true', color=hue(1),
xmin=-11, xmax=11, ymin=-30, ymax=30, legend_label='$y^2=x^3-2x$')
show(p)
```

图 6.8: 椭圆曲线  $y^2 = x^3 - 2x$  在有限域  $GF(11)$  和实数域上的图

有限域椭圆曲线构成的群如下:

- 单位元:  $O$  为单位元, 椭圆曲线上的所有点  $P$  有  $P + O = P$ ,  $O$  也是一个椭圆曲线上的一点, 是一个无穷远的点。
- 逆元: 对点  $P = (x, y)$ , 其加法逆元为  $(x, -y)$ , 记为  $-P$ ,  $P + (-P) = O$ , 由此也可以定义减法  $P - P = O$
- 加法: 对于两个不同且不互逆的点  $P(x_1, y_1), Q(x_2, y_2), x_1 \neq x_2, P(x_1, y_1) + Q(x_2, y_2) = S(x_3, y_3)$ , 其中:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

- 倍数: 点  $P$  的倍数定义为,  $2P = P(x_1, y_1) + P(x_1, y_1) = S(x_3, y_3)$ , 其中:

$$x_3 = \lambda^2 - 2x_1 p \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

$$\lambda = \frac{3x_1^2 + a}{2y_1}, \text{ 其中 } a \text{ 是方程中的常数。}$$

假设  $E/F_p$  是一个椭圆曲线,  $e \geq 1$ , 如果  $(x_1, y_1), x_1, y_1 \in F_{p^e}$ , 满足曲线方程  $E$ , 我们说点  $(x_1, y_1)$  在椭圆曲线上。当  $e = 1$  时, 点  $(x_1, y_1)$  定义在基础域 (base field)  $F_p$  上, 当  $e > 1$  时, 点  $(x_1, y_1)$  定义在域  $F_p$  的扩展上。在域  $F_{p^e}$  上曲线  $E$  上的所有点, 包括无穷远点 (the point of infinity)  $O$ , 我们记为  $E(F_{p^e})$ , 用  $|E(F_{p^e})|$  表示椭圆曲线上点的个数。

根据哈赛 (Hase) 的研究结果, 我们有  $|E(F_{p^e})| = p^e + 1 - t$ , 其中  $|t| \leq 2\sqrt{p^e}$ , 这表明  $|E(F_{p^e})|$  非常接近  $p^e - 1$ 。对于  $E(F_p)$  来说, 这个值为  $p + 1$ .[?]

**例 6.11 [18]**  $GF(11)$  上的一个椭圆曲线  $E_{11}(1, 6) : y^2 = x^3 + x + 6 \pmod{11}$  构成的交换群。

**解** 1、计算椭圆曲线上所有的点

对于  $GF(11)$  上的每一个点  $x$  计算  $s = x^3 + x + 6 \pmod{11}$ , 然后求解  $y^2 = s \pmod{11}$ , 如果此方程有解 (即  $s$  为模 11 的平方剩余), 解为  $y$ , 则  $(x, \pm y)$  是  $E_{11}(1, 6)$  上的点, 按照这个方法, 我们可以获得  $E_{11}(1, 6)$  上的点, 共 12 个点:

$$(2, 4), (2, 7), (3, 5), (3, 6), (5, 2), (5, 9), (7, 2), (7, 9), (8, 3), (8, 8), (10, 2), (10, 9)$$

2、交换群

我们可以看到  $(2, 4)$  和  $(2, 7)$  互为逆元, 下面我们计算  $(2, 4) + (3, 5)$ :

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5 - 4}{3 - 2} = 1$$

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{11} = 1^2 - 2 - 3 \pmod{11} = 7$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{11} = (2 - 7) - 4 = 2$$

$(2, 4) + (3, 5) = (7, 2)$ , 可以看到  $(7, 2)$  仍然是椭圆曲线上的点, 可见加法运算在  $E_{11}(1, 6)$  上是封闭的。

**例 6.12 [17]** 在  $E_{23}(1, 1)$  上计算  $P + Q, P = (3, 10), Q = (9, 7)$ 。

$$\text{解 } \lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{7 - 10}{9 - 3} = \frac{-3}{6} = \frac{-1}{2} = 11 \pmod{23} \text{ <sup>11</sup>}$$

<sup>11</sup> 此处计算解释, 我们需要找到一个数  $a, 2a = 1 \pmod{23}$ , 可知  $a = 12, -12 \pmod{23} = 11$

$$\begin{aligned}x_3 &= \lambda^2 - x_1 - x_2 \pmod{23} = 11^2 - 3 - 9 \pmod{23} = 17 \\y_3 &= \lambda(x_1 - x_3) - y_1 \pmod{23} = 11(3 - 17) - 10 = -164 = 20 \\P + Q &= (17, 20), \text{ 结果仍然为 } E_{23}(1, 1) \text{ 中的点。}\end{aligned}$$

### 6.4.3 构造密码算法

要想利用椭圆曲线构造密码算法，从大的方面来说，要解决以下几个问题：

1. 将一个信息，也就是一个二进制串或者一个数，映射到椭圆曲线的一个点，同时也可将椭圆曲线上的一点映射到一个信息。
2. 然后在椭圆曲线上找到一个计算困难问题，但这个困难问题当有某个信息时要变得不困难。
3. 设计一个信息变换或计算过程。

下面我们依次看看以上这几个问题的考虑。

#### 6.4.3.1 消息到椭圆曲线上的映射

设消息是  $m, 0 \leq m \leq M, E : y^3 = x^2 + ax + b$ , 给定一个整数  $k$ , 实际中  $k$  在 30~50 之间取值，在这里我们取  $k=30$ , 对明文  $m$  计算一系列  $x$ ,  $x = \{mk + j, j = 0, 1, 2, \dots\} = \{30m + j, j = 0, 1, 2, \dots\}$ , 直到  $x^2 + ax + b \pmod{p}$  是平方根, 那么就将  $m$  映射到椭圆曲线上这一点  $(x, \sqrt{x^2 + ax + b})$ 。

反之，有一个椭圆曲线上的一点  $(x, y)$ , 我们将其还原为消息  $m$  的过程为  $m = \lfloor \frac{x}{30} \rfloor$ .

#### 6.4.3.2 椭圆曲线上的计算困难问题

在  $E_p(a, b)$  上考虑方程  $Q = kP, Q, P \in E_p(a, b), k < p$ , 我们知道了  $k, P$  很容易求解  $Q$ , 但是知道了  $P, Q$ , 求确  $k$  是一个困难问题, 这就是椭圆曲线上的离散对数问题。

#### 6.4.3.3 椭圆曲线上的密码算法

**6.4.3.3.1 ECC(Elliptic Curve Cryptography) 加密算法** 通常软件实现采用的域为  $GF(p)$  域, 在硬件实现的采用  $GF(2^m)$  域, 下面我们看一个经典的  $GF(p)$  域上的 ECC 算法。[?]

ECC 属于公钥密码体制, 下面假设用 Alice 给 Bob 准备发送一个秘密信息的实例来说明整个过程。

##### (1) Bob 生成公私钥对

- 选择椭圆曲线  $E : y^2 = x^3 + ax + b \pmod{p}$ , 构造群  $E_p(a, b)$ .
- 在  $E_p(a, b)$  上挑选生成元  $g = (x_0, y_0), g$  应使得满足  $ng = O$  的最小  $n$  是一个非常大的素数。
- 选择一个随机数  $\alpha, \alpha \in [1, n - 1]$ , 计算  $\beta = \alpha g$ .
- Bob 的公钥为  $(E_p(a, b), n, g\beta)$ , 私钥为  $\alpha$ 。

**(2)Alice 对消息  $\mathbf{m}$  加密**

- 选择一个随机数  $k, k \in [1, n - 1]$ .
- 计算点  $C_1 = (x_1, y_1) = kg$ .
- 随机选择一个点  $P_t = (x_t, y_t)$ , 计算  $C_2 = P_t + k\beta$ .
- 计算密文  $C_3 = mx_t + y_t$ .
- 将  $(C_1, C_2, C_3)$  做为密文发给 Bob。

**(3)Bob 对密文  $(C_1, C_2, C_3)$  解密**

- 使用私钥  $\alpha$  计算  $C_2 - \alpha C_1 = (x'_t, y'_t) = P'_t$ .
- 计算  $m = \frac{(C_3 - y'_t)}{x'_t}$ ,  $m$  即为解密后的明文。

我们简单看看以上过程的正确性:

$$\begin{aligned} \alpha C_1 &= \alpha kg = k\alpha g = k\beta \\ (x'_t, y'_t) &= C_2 - \alpha C_1 = P_t + k\beta - k\beta = P_t = (x_t, y_t) \\ \frac{C_3 - y'_t}{x'_t} &= \frac{mx_t + y_t - y'_t}{x'_t} = m \end{aligned}$$

#### 6.4.3.3.2 Diffie-Hellman 密钥交换

下面我们介绍一下如何用椭圆曲线进行 DH 密钥交换.

- 选一个素数  $p \approx 2^{180}$  和两个参数  $a, b$ , 构造  $E_p(a, b)$ .
- 取  $E_p(a, b)$  的一个生成元  $G_1(x_1, y_1)$ , 使 G 的阶  $n$  是一个非常大的素数。<sup>12</sup>
- $E_p(a, b), G, n$  公开。
- 用户 A 任选  $n_A, n_A \in [1, n - 1]$ ,  $n_A$  为 A 的私钥,  $P_A = n_A G$  是 A 的公钥。
- 用户 B 任选  $n_B, n_B \in [1, n - 1]$ ,  $n_B$  为 B 的私钥,  $P_B = n_B G$  是 B 的公钥。
- A、B 双方利用对方的公钥和自身的私钥, 即可产生双方共享的密钥  $K$ , A 计算  $K = n_A P_B$ , B 计算  $K = n_B P_A$

---

<sup>12</sup>G 的阶是满足  $nG = O$  的最小正整数 n。



# 第 7 章 哈希函数 (Hash Function)

---

## 7.1 哈希函数特点

哈希函数 (Hash function)H 也称为单向散列函数，其特点是：

1. 函数输入为任意长度的消息 M，输出为固定长度的消息 h， $h = H(M)$ .
2. 函数 H 是容易计算的，也就是说给定 M，很容易计算  $h = H(M)$ .
3. 给定 h，计算 M，这是个困难问题。
4. 给定 M，找到另外一个消息 M'，使得  $H(M) = H(M')$  是个困难问题。也就是哈希函数要能抗碰撞 (collision-resistance)。

## 7.2 对哈希函数的攻击

### 7.2.1 穷举攻击

已知一个哈希值 H(M)，攻击者依据一定的遍历规则创建消息，直到找到一个消息 M'，使得  $H(M') = H(M)$ 。

### 7.2.2 生日攻击 (birthday attack)

攻击者通过一定方法寻找两个随机消息 M' 和 M，使得  $H(M') = H(M)$ .

这种冲突通常比较容易实现，我们下面进行解释，这也是生日攻击名字的来源。

生日悖论是指在不少于 23 个人中至少有两人生日相同的概率大于 50%。例如在一个 30 人的小学班级中，存在两人生日相同的概率为 70%。对于 60 人的大班，这种概率要大于 99%。从引起逻辑矛盾的角度来说，生日悖论并不是一种“悖论”。但这个数学事实十分反直觉，故称之为一个悖论。生日悖论的数学理论被应用于设计密码学攻击方法——生日攻击。<sup>1</sup>

对于一个哈希函数，假定其输出为 m 位，利用穷举攻击，找到一个碰撞 (collision) 需要  $2^m$  次试探，而需要寻找两个随机消息，其发生碰撞，只需要  $2^{m/2}$  次试探，我们看看这两个区别。假如 m=64，计算一次哈希值  $1/10^6$  秒 (也就是说 1 秒计算 1 百万个哈希值)，那么用穷举法最多需要计算  $2^{64} = 18446744073709551616$  次，大约需要 58 万年，如果我们只是寻找两个一样的数，需要计算  $2^{32} = 4294967296$  次，大约需要 1 个多小时。

从上面的例子可以看到，如果你认为哈希值长度为 m 位是就够了，如果你需要考虑生日攻击，那么你选择的哈希函数生成的哈希值长度应该是 2m。

例如如果你想让攻击者破解哈希函数的可能性低于  $1/2^{80}$ ，那么你应该选择使用 160 位的哈希函数。

---

<sup>1</sup>此段话摘抄自<https://baike.baidu.com/item/>

## 7.3 MD5(Message-Digest algorithm 5)

MD5 是 Ron Rivest 在 1990 年 10 月作为 RFC 提出的 (RFC 1321)，其前身是 MD4(RFC 1320)。MD5 输入为任意长度的信息，输出 128bit 的摘要信息 (散列值)。

### 7.3.1 MD5 算法

#### 7.3.1.1 消息填充

填充后的消息比特长度  $L$  为模 512 下是 448，也就是说  $L \equiv 448 \pmod{512}$ ，或者说  $L = n \times 512 + 448$ ， $n$  为大于等于 1 的正整数。填充方式为第一位为 1，后面为 0.

#### 7.3.1.2 附加消息长度

前面填充后最后有 448bit， $448+64=512$ bit，留下的这 64bit 用于填写附加消息长度，用 little-endian 方式存储。那么可存储的最大附加消息长度为  $2^{64}$ ，如果消息长度大于此数，存储长度模  $2^{64}$  后的值。

附加消息长度后，进行后续运算的消息长度是 512 的倍数。

#### 7.3.1.3 缓冲区初始化

就是初始化 MD5 核心算法中的  $W_0, X_0, Y_0, Z_0, W_0 = 01\ 23\ 45\ 67; X_0 = 89\ AB\ CD\ EF; Y_0 = FE\ DC\ BA\ 98; Z_0 = 76\ 54\ 32\ 10$ 。

#### 7.3.1.4 分组流水处理

将消息分为  $n$  个 512bit 长度的分组  $m_1, m_2, \dots, m_n$ ，然后对  $n$  个分组依次执行：

1.  $(W_1, X_1, Y_1, Z_1) = H_{MD5}(W_0, X_0, Y_0, Z_0, m_1)$
2.  $(W_2, X_2, Y_2, Z_2) = H_{MD5}(W_1, X_1, Y_1, Z_1, m_2)$
3.  $(W_3, X_3, Y_3, Z_3) = H_{MD5}(W_2, X_2, Y_2, Z_2, m_3)$
- ...
- $n. (W_n, X_n, Y_n, Z_n) = H_{MD5}(W_{n-1}, X_{n-1}, Y_{n-1}, Z_{n-1}, m_n)$

#### 7.3.1.5 输出

MD5 的输出为  $W_n || X_n || Y_n || Z_n$ 。

#### 7.3.1.6 MD5 算法表示

整个 MD5 算法，输入为原始消息  $M_s$ ，初始量  $W_0, X_0, Y_0, Z_0$ ，输出为 128bit， $W || X || Y || Z$ ，我们记为  $(W, X, Y, Z)$  or  $W || X || Y || Z = MD5(W_0, X_0, Y_0, Z_0, M_s)$ .

### 7.3.2 核心算法 $H_{MD5}$

核心算法  $H_{MD5}$  是个分组算法，输入是 512 位的分组数据，输出是 128 位。

#### 7.3.2.1 $H_{MD5}$ 算法

$H_{MD5}$  算法的伪代码如下：

```

 $M, W_0, X_0, Y_0, Z_0$ 
 $(W_1, X_1, Y_1, Z_1) = R(F, G, H, I, W_0, X_0, Y_0, Z_0, M)$ 
 $W = W_0 + W_1 \pmod{2^{32}}$ 
 $X = X_0 + X_1 \pmod{2^{32}}$ 
 $Y = Y_0 + Y_1 \pmod{2^{32}}$ 
 $Z = Z_0 + Z_1 \pmod{2^{32}}$ 
输出 W,X,Y,X

```

上面伪代码表示的这个算法，其输入是  $M, W_0, X_0, Y_0, Z_0$ ，算法输出是  $W, X, Y, X$ ，用  $(W, X, Y, X) = H_{MD5}(W_0, X_0, Y_0, Z_0, M)$  符号化来表示。

#### 7.3.2.2 四轮运算 R

算法的输入是 512bit 的消息  $M$ ，我们再将其分为 16 组，每组 32bit， $M = M_0||M_1||M_2||\dots||M_{15}$ ，然后对其进行四轮核心运算。我们将整个四轮运算过程记为  $R(F, G, H, I, W, X, Y, Z, M)$ ：  
 $W \times X \times Y \times Z \times M \rightarrow W \times X \times Y \times Z$ ，也就是说这四轮运算后输出是新的  $W, X, Y, X$ ，下面我们看看四轮运算的具体过程

◆ 第一轮 C 为 F，所以核心运算是  $FF(W, X, Y, Z, M, s, t)$ ，在第一轮中依次对 16 个分组数据执行 FF 运算，也就是说第一轮执行 16 次 FF 运算，每一次的执行 FF 时， $W, X, Y, X$  是上一次运算后的  $W, X, Y, X$ ，在执行第一次 FF 时， $W, X, Y, X$  的初始值为， $W = 0x01234567, X = 0x89abcdef, Y = 0xfedcba98, Z = 0x76543210$ <sup>2</sup>。我们用伪代码描述此轮算法：

W,X,Y,Z 初始化

for i=0 to 15

$FF(W, X, Y, Z, M_i^1, s_i^1, t_{i+1})$

$WXYZ = WXYZ \gg 32$ ; 表示将四个值连接，然后右循环移位 32，也就是新 W 值为旧 Z，新 X 值为旧 W，新 Y 值为旧 X，新 Z 值为旧 Y

◆ 第二轮 C 为 G:

W,X,Y,Z 上一轮遗留值

for i=0 to 15

$GG(W, X, Y, Z, M_i^2, s_i^2, t_{i+17})$

<sup>2</sup>可以看出这 4 个 32bit 变量是 0 f 正序逆序排列，然后分割为四个变量所得

$WXYZ = WXYZ \gg 32$ ; 表示将四个值连接，然后右循环移位 32，也就是新 W 值为旧 Z，新 X 值为旧 W，新 Y 值为旧 X，新 Z 值为旧 Y

◆ 第三轮 C 为 H:

W,X,Y,Z 上一轮遗留值

for i=0 to 15

$$HH(W, X, Y, Z, M_i^3, s_i^3, t_{i+33})$$

$WXYZ = WXYZ \gg 32$ ; 表示将四个值连接，然后右循环移位 32，也就是新 W 值为旧 Z，新 X 值为旧 W，新 Y 值为旧 X，新 Z 值为旧 Y

◆ 第四轮 C 为 I:

W,X,Y,Z 上一轮遗留值

for i=0 to 15

$$II(W, X, Y, Z, M_i^4, s_i^4, t_{i+49})$$

$WXYZ = WXYZ \gg 32$ ; 表示将四个值连接，然后右循环移位 32，也就是新 W 值为旧 Z，新 X 值为旧 W，新 Y 值为旧 X，新 Z 值为旧 Y

### 7.3.2.3 R 中的 $M_i^n$ , ( $n = 1, 2, 3, 4$ )

四轮运算中，每轮消息顺序不同，第一轮为顺次，消息从  $M_0, M_2, \dots, M_{15}$ ，第二、三、四轮，对 M 的顺序进行了置换：

$$\text{第一轮 } M_i^1 = M_i, i = 0, 1, \dots, 15$$

$$\text{第二轮 } M_i^2 = M_{1+5i \bmod 16}, i = 0, 1, \dots, 15$$

$$\text{第三轮 } M_i^3 = M_{5+3i \bmod 16}, i = 0, 1, \dots, 15$$

$$\text{第四轮 } M_i^4 = M_{7i \bmod 16}, i = 0, 1, \dots, 15$$

按上面公式，可知，第二轮 M 的顺序是：

$$M_1, M_6, M_{11}, M_0, M_5, M_{10}, M_{15}, M_4, M_9, M_{14}, M_3, M_8, M_{13}, M_2, M_7, M_{12}$$

### 7.3.2.4 R 中的 $s_i^n$ , ( $n = 1, 2, 3, 4$ )

每轮每次运算移位数  $s_i^n$  的取值如表7.1，n 表示时第几轮，i 表示是本轮的第几次运算。

### 7.3.2.5 R 中的 $t_i$

每轮中的每次运算都有  $t_i$ ，他是个常数，在四轮运算中其是顺序变换，共有 64 个常数值  $t_1, t_2, \dots, t_{64}$ ，我们这样设定这个常数值,  $t_i = [2^{32} \times \text{abs}(\sin(i))]$ ，其中 i 的单位为弧

表 7.1: MD5 中循环移位操作参数选择

参数 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

度 (RAD), [] 表示取整运算, 就是取一个数的整数部分, abs 是绝对值, 其实我们可以事先做好这个常数表, 以备计算时使用, 下面我们计算  $t_1$ 。

$$t_1 = [2^{32} \times \text{abs}(\sin(1))] = [3614090360.3] = 3614090360 = [d76aa478]_{16}$$

### 7.3.2.6 核心运算

我们定义一个核心运算, 其中 C 为非线性函数 F,G,H,I 中的一个,  $d << s$  表示对数 d 循环左移 s 位, + 表示模  $2^{32}$  加法, 后面为了方便我们直接写 +, 核心运算 CC 我们定义为:

$$CC(W, X, Y, Z, M, s, t) : W = X + ((W + C(X, Y, Z) + M + t) << s) \quad (7.1)$$

如果 C 用 F 替换, 那么核心运算 CC 就是 FF; 如果 C 用 G 替换, 那么核心运算 CC 就是 GG; 如果 C 用 H 替换, 那么核心运算 CC 就是 HH; 如果 C 用 I 替换, 那么核心运算 CC 就是 II。

这里我们其实是定义了 R 运算中的函数 FF、GG、HH、II。

### 7.3.2.7 非线性函数

在核心运算定义中有四个非线性函数  $F(X, Y, Z), G(X, Y, Z), H(X, Y, Z), I(X, Y, Z)$ , 下面我们分别定义这四个非线性函数。

我们定义四个非线性函数, 其中 X, Y, Z 是 32bit 数,  $\wedge$  是与操作,  $\vee$  是或操作,  $\oplus$  是异或操作,  $\neg$  是非操作。

1.  $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
2.  $G(X, Y, Z) = (X \wedge Z) \vee (\neg Y \wedge \neg Z)$
3.  $H(X, Y, Z) = X \oplus Y \oplus Z$
4.  $I(X, Y, Z) = Y \oplus (X \wedge \neg Z)$

## 7.4 MD5 安全性

这节文字摘抄自网络上<sup>3</sup>。

<sup>3</sup><https://en.bitcoinwiki.org/wiki/MD5>

### 7.4.1 History and cryptanalysis

MD5 is one in a series of message digest algorithms designed by Professor Ronald Rivest of Massachusetts Institute of Technology|MIT (Rivest, 1992). When analytic work indicated that MD5's predecessor MD4 was likely to be insecure, Rivest designed MD5 in 1991 as a secure replacement. (Hans Dobbertin did indeed later find weaknesses in MD4.)

In 1993, Den Boer and Bosselaers gave an early, although limited, result of finding a "pseudo-collision" of the MD5 compression function; that is, two different initialization vectors that produce an identical digest.

In 1996, Dobbertin announced a collision of the compression function of MD5 (Dobbertin, 1996). While this was not an attack on the full MD5 hash function, it was close enough for cryptographers to recommend switching to a replacement, such as SHA-1 or RIPEMD-160.

The size of the hash value (128 bits) is small enough to contemplate a birthday attack. MD5CRK was a distributed project started in March 2004 with the aim of demonstrating that MD5 is practically insecure by finding a collision using a birthday attack.

MD5CRK ended shortly after 17 August 2004, when collisions for the full MD5 were announced by Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Their analytical attack was reported to take only one hour on an IBM p690 cluster.

On 1 March 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger demonstrated construction of two X.509 certificates with different public keys and the same MD5 hash value, a demonstrably practical collision. The construction included private keys for both public keys. A few days later, Vlastimil Klima described an improved algorithm, able to construct MD5 collisions in a few hours on a single notebook computer. On 18 March 2006, Klima published an algorithm that could find a collision within one minute on a single notebook computer, using a method he calls tunneling.

Various MD5-related RFC errata have been published. In 2009, the United States Cyber Command used an MD5 hash value of their mission statement as a part of their official emblem.

On 24 December 2010, Tao Xie and Dengguo Feng announced the first published single-block (512-bit) MD5 collision. (Previous collision discoveries had relied on multi-block attacks.) For "security reasons", Xie and Feng did not disclose the new attack method. They issued a challenge to the cryptographic community, offering a US\$10,000 reward to the first finder of a different 64-byte collision before 1 January 2013. Marc Stevens responded to the challenge and published colliding single-block messages as well as the construction algorithm and sources.

In 2011 an informational RFC 6151 was approved to update the security considerations in MD5 and HMAC-MD5.

### 7.4.2 security

The security of the MD5 hash function is severely compromised. A collision attack exists that can find collisions within seconds on a computer with a 2.6 GHz Pentium 4 processor . Further, there is also a chosen-prefix collision attack that can produce a collision for two inputs with specified prefixes within hours, using off-the-shelf computing hardware.

The ability to find collisions has been greatly aided by the use of off-the-shelf GPUs. On an NVIDIA GeForce 8400GS graphics processor, 16 – 18 million hashes per second can be computed. An NVIDIA GeForce 8800 Ultra can calculate more than 200 million hashes per second.

These hash and collision attacks have been demonstrated in the public in various situations, including colliding document files and digital certificates.



## 7.5 SHA

SHA 是 Secure Hash Algorithm 的缩写。1993 年发布 SHA，1995 年发布 SHA-1，2002 年发布 SHA-2，2008 年发布 SHA-3，2015 年发布 SHA-4。

### 7.5.1 SHA-1

SHA-1 输入为小于  $2^{64}$  比特长的任意消息，生成一个 160 比特 (20 字节，40 个 16 进制值) 的摘要信息。SHA-1 的介绍，我们直接节选引用 NIST 的标准<sup>4</sup>，也许看的更直观一些。

#### 7.5.1.1 Introduction

The Secure Hash Algorithm (SHA) is required for use with the Digital Signature Algorithm (DSA) as specified in the Digital Signature Standard (DSS) and whenever a secure hash algorithm is required for federal applications. For a message of length < 2<sup>64</sup> bits, the SHA produces a 160-bit condensed representation of the message called a message digest. The message digest is used during generation of a signature for the message. The SHA is also used to compute a message digest for the received version of the message during the process of verifying the signature. Any change to the message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

The SHA is designed to have the following properties: it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.

#### 7.5.1.2 BIT strings and integers

The following terminology related to bit strings and integers will be used:

a. A hex digit is an element of the set 0, 1, . . . , 9, A, . . . , F. A hex digit is the representation of a 4-bit string. Examples: 7 = 0111, A = 1010.

b. A word equals a 32-bit string which may be represented as a sequence of 8 hex digits.

To convert a word to 8 hex digits each 4-bit string is converted to its hex equivalent as described in (a) above. Example:

1010 0001 0000 0011 1111 1110 0010 0 01 1. =A103FE23

c. An integer between 0 and  $2^{32} - 1$  inclusive may be represented as a word. The least significant four bits of the integer are represented by the right-most hex digit of the word representation. Example: the integer  $291 = 28 + 25 + 21 + 2 = 256 + 32 + 2 + 1$  is represented by the hex word, 00000123.

---

<sup>4</sup>Federal Information Processing Standards Publication 180-1, 1995 April 17, Specifications for the SECURE HASH STANDARD

If  $z$  is an integer,  $0 \leq z < 2^{64}$ , then  $z = 2^{32}x + y$  where  $0 \leq x < 2^{32}$  and  $0 \leq y < 2^{32}$ . Since  $x$  and  $y$  can be represented as words  $X$  and  $Y$ , respectively,  $z$  can be represented as the pair of words  $(X, Y)$ .

- d. block= 512-bit string. A block (e.g.,  $B$ ) may be represented as a sequence of 16 words.

### 7.5.1.3 Operations on Words

The following logical operators will be applied to words:

- a. Bitwise logical word operations

$X \wedge Y$  = bitwise logical "and" of  $X$  and  $Y$ .

$X \vee Y$  = bitwise logical "inclusive-or" of  $X$  and  $Y$ .

$X \text{ XOR } Y$  = bitwise logical "exclusive-or" of  $X$  and  $Y$ .

$\sim X$  = bitwise logical "complement" of  $X$ .

- b. The operation  $X + Y$  is defined as follows: words  $X$  and  $Y$  represent integers  $x$  and  $y$ , where  $0 \leq x < 2^{32}$  and  $0 \leq y < 2^{32}$ . For positive integers  $n$  and  $m$ , let  $n \bmod m$  be the remainder upon dividing  $n$  by  $m$ . Compute

$$z = (x + y) \pmod{2^{32}}$$

Then  $0 \leq z < 2^{32}$  Convert  $z$  to a word,  $Z$ , and define  $Z = X + Y$ .

- c. The circular left shift operation  $S^n(X)$ , where  $X$  is a word and  $n$  is an integer with  $0 \leq n < 32$ , is defined by

$$S^n(X) = (X \ll n) \vee (X \gg 32 - n).$$

In the above,  $X \ll n$  is obtained as follows: discard the left-most  $n$  bits of  $X$  and then pad the result with  $n$  zeroes on the right (the result will still be 32 bits).  $X \gg n$  is obtained by discarding the right-most  $n$  bits of  $X$  and then padding the result with  $n$  zeroes on the left. Thus  $S^n(X)$  is equivalent to a circular shift of  $X$  by  $n$  positions to the left.

### 7.5.1.4 Message Padding

The SHA is used to compute a message digest for a message or data file that is provided as input. The message or data file should be considered to be a bit string. The length of the message is the number of bits in the message (the empty message has length 0). If the number of bits in a message is a multiple of 8, for compactness we can represent the message in hex. The purpose of message padding is to make the total length of a padded message a multiple of 512. The SHA sequentially processes blocks of 512 bits when computing the message digest. The following specifies how this padding shall be performed. As a summary, a "1" followed by  $m$  "0"s followed by a 64-bit integer are appended to the end of the message to produce a padded message of length  $512 \times n$ . The 64-bit integer is  $l$ , the length of the original message. The padded message is then processed by the SHA as  $n$  512-bit blocks.



Suppose a message has length  $l < 2^{64}$ . Before it is input to the SHA, the message is padded on the right as follows:

a. "1" is appended. Example: if the original message is "01010000", this is padded to "010100001".

b. "0"s are appended. The number of "0"s will depend on the original length of the message. The last 64 bits of the last 512-bit block are reserved for the length  $l$  of the original message.

Example: Suppose the original message is the bit string

01100001 01100010 01100011 01100100 01100101.

After step (a) this gives

01100001 01100010 01100011 01100100 01100101 1.

Since  $l = 40$ , the number of bits in the above is 41 and 407 "0"s are appended, making the total now 448. This gives (in hex)

61626364 65800000 00000000 00000000  
00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000000  
00000000 00000000.

c. Obtain the 2-word representation of  $l$ , the number of bits in the original message. If  $l < 2^{32}$  then the first word is all zeroes. Append these two words to the padded message.

Example: Suppose the original message is as in (b). Then  $l = 40$  (note that  $l$  is computed before any padding). The two-word representation of 40 is hex 00000000 00000028. Hence the final padded message is hex

61626364 65800000 00000000 00000000  
00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000028.

The padded message will contain  $16n$  words for some  $n > 0$ . The padded message is regarded as a sequence of  $n$  blocks  $M_1, M_2, \dots, M_n$ , where each  $M_i$  contains 16 words and  $M_1$  contains the first characters (or bits) of the message.

### 7.5.1.5 Functions Used

A sequence of logical functions  $f_0, f_1, \dots, f_{79}$  is used in the SHA. Each  $f_t, 0 \leq t \leq 79$ , operates on three 32-bit words and produces a 32-bit word as output.  $f_t$  is defined as follows: for words, B, C, D,

$$f_t(B, C, D) = (B \wedge C) \vee (\sim B \wedge D), 0 \leq t \leq 19$$

$$f_t(B, C, D) = B \text{ XOR } C \text{ XOR } D, 20 \leq t \leq 39$$

$$f_t(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D), 40 \leq t \leq 59$$



$$f_t(B, C, D) = B \text{ XOR } C \text{ XOR } D, 60 \leq t \leq 79$$

### 7.5.1.6 Constants Used

A sequence of constant words  $K_0, K_1, \dots, K_{79}$  is used in the SHA. In hex these are given by

$$K_t = 5A827999, 0 \leq t \leq 19$$

$$K_t = 6ED9EBA1, 20 \leq t \leq 39$$

$$K_t = 8F1BBCDC, 40 \leq t \leq 59$$

$$K_t = CA62C1D6, 60 \leq t \leq 79$$

### 7.5.1.7 Computing the Message digest

The message digest is computed using the final padded message. The computation uses two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labeled A,B,C,D,E. The words of the second 5-word buffer are labeled  $H_0, H_1, H_2, H_3, H_4$ . The words of the 80-word sequence are labeled  $W_0, W_1, \dots, W_{79}$ . A single word buffer TEMP is also employed.

To generate the message digest, the 16-word blocks  $M_1, M_2, \dots, M_n$  defined in Section 4 are processed in order. The processing of each involves 80 steps.

Before processing any blocks, the  $H_j$  are initialized as follows: in hex,

$$H_0 = 67452301$$

$$H_1 = EFCDAB89$$

$$H_2 = 98BADCFE$$

$$H_3 = 10325476$$

$$H_4 = C3D2E1F0$$

Now  $M_1, M_2, \dots, M_n$  are processed. To process  $M_i$ , we proceed as follows:

a. Divide  $M_i$  into 16 words  $W_0, W_1, \dots, W_{15}$ , where  $W_0$  is the left-most word.

b. For  $t= 16$  to  $79$  let  $W_t = W_{t-3} \text{ XOR } W_{t-8} \text{ XOR } W_{t-14} \text{ XOR } W_{t-16}$

c. Let  $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$ .

d. For  $t = 0$  to  $79$  do

$$\text{TEMP} = S^5(A) + f_t(B, C, D) + E + W_t + K_t;$$

$$E = D; D = C; C = S^{30}(B); B = A; A = \text{TEMP};$$

e. Let  $H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C, H_3 = H_3 + D, H_4 = H_4 + E$ .

After processing  $M_n$ , the message digest is the 160-bit string represented by the 5 words

$$H_0 \ H_1 \ H_2 \ H_3 \ H_4$$

### 7.5.1.8 Alternate method of computation

The above assumes that the sequence  $W_0, \dots, W_{79}$  is implemented as an array of eighty 32-bit words. This is efficient from the standpoint of minimization of execution time, since the addresses



of  $W_{t-3}, \dots, W_{t-16}$  in step (b) are easily computed. If space is at a premium<sup>5</sup>, an alternative is to regard  $W_t$  as a circular queue, which may be implemented using an array of sixteen 32-bit words  $W[O], \dots, W[15]$ . In this case, in hex let MASK = 000000F. Then processing of  $M_i$  is as follows:

- a. Divide  $M_i$  into 16 words  $W[O], \dots, W[15]$ , where  $W[O]$  is the left-most word.
- b. Let  $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$ .
- c. For  $t = 0$  to 79 do

$s = t \wedge MASK;$

if ( $t \geq 16$ )

$W[s] = W[(s + 13) \wedge MASK] XOR W[(s + 8) \wedge MASK] XOR W[(s + 2) \wedge MASK] XOR W[s];$

$TEMP = S^5(A) + f_t(B, C, D) + E + W[s] + K_t;$

$E = D; D = C; C = S^{30}(B); B = A; A = TEMP;$

## 7.5.2 SHA-1 的破解

### 7.5.2.1 王小云院士的十年一剑

以下来自 NIST 网址的一个信息<sup>6</sup>。

NIST Comments on Cryptanalytic Attacks on SHA-1

April 26, 2006

In 2005 Prof. Xiaoyun Wang announced a differential attack on the SHA-1 hash function; with her recent improvements, this attack is expected to find a hash collision (two messages with the same hash value) with an estimated work of 263 operations, rather than the ideal 280 operations that should be required for SHA-1 or any good 160-bit hash function. This is a very large computation, and to our knowledge nobody has yet verified Prof. Wong's method by finding a SHA-1 collision, but 263 operations is plainly within the realm of feasibility for a high resource attacker. NIST accepts that Prof. Wang has indeed found a practical collision attack on SHA-1.

NIST held a workshop to consider the status of hash functions on Oct. 31-Nov. 1, 2005 and has reviewed the implications of Prof. Wang's attack. The attack primarily affects some digital signature applications, including timestamping and certificate signing operations, where one party prepares a message for the generation of a digital signature by a second party, and third parties then verify the signature. There are many applications of hash functions, and many do not require strong collision resistance; for example, keyed hash applications, such as the Hash-based Message

<sup>5</sup>这是早期 Intel 的一种 CPU

<sup>6</sup>信息来自于 <https://csrc.nist.gov/News/2006/NIST-Comments-on-Cryptanalytic-Attacks-on-SHA-1>

Authentication Code (HMAC) or key derivation applications of hash functions do not seem to be affected.

Several steps are now prudent. The first of these is to transition rapidly to the stronger “SHA-2” family of hash functions (SHA-224, SHA-256, SHA-384 and SHA-512) for digital signature applications. The SHA-2 hash functions are in the same general family of hash functions as SHA-1. They could potentially be attacked with similar techniques, but they are much stronger than SHA-1. Practical SHA-2 attacks are unlikely in the next decade; and might never be found, except through decades of exponential growth of available computing power. The SHA-2 hash functions are well along in the commercial system deployment process and are available in many newer systems and applications, but are not yet available in the majority of deployed systems. The primary constraint on the current use of the SHA-2 hash functions for signatures is interoperability; many relying party systems do not yet implement them, and may not do so for several more years. NIST encourages a rapid adoption of the SHA-2 hash functions for digital signatures, and, in any event, Federal agencies must stop relying on digital signatures that are generated using SHA-1 by the end of 2010.

The second step is to encourage hash function research to better understand hash function design and attacks in preparation for selecting additional hash functions. The cryptographic community is in a period of rapid development in the theory of hash functions and their cryptanalysis. NIST plans to host additional hash function workshops; the next of these will be held on Aug. 24-25, 2006 in Santa Barbara, California to follow immediately after the Crypto 2006 Conference.

The third step will be a hash function competition, similar to the successful Advanced Encryption Standard (AES) development and selection process. The schedule for this competition has not yet been determined, but presupposes a sense of sufficient maturity and stability in hash function theory and technology, that the results will improve on or otherwise complement the SHA-2 hash functions, and will occur before the current hash functions are determined to be insecure. NIST does not have strong preconceptions about the number of new hash functions to be selected from this competition, since the very broad range of hash function applications may argue for two or more specialized hash functions.

Lily Chen

NIST

Manager, Cryptographic Technology Group

301-975-6974

### 7.5.2.2 进一步崩盘

其实从找到一个碰撞到能够把这个漏洞应用，这之间还是有一定距离的，但是如果找到了碰撞，就是证明其已经不安全，也许很早以前就有人已经开始利用他了，或者把他做为一种能力握到手里。

下面是 2017 年发的另外一个 SHA-1 不安全的信息<sup>7</sup>。

#### Research Results on SHA-1 Collisions

February 24, 2017

On Thursday, February 23rd, Google announced that a team of researchers from the CWI Institute in Amsterdam and Google have successfully demonstrated an attack on the SHA-1 hash algorithm by creating two files that hash to the same value.

Their results further emphasize the need to migrate to stronger hash algorithms for digital signatures and other applications that require collision resistance.

NIST deprecated the use of SHA-1 in 2011 and disallowed its use for digital signatures at the end of 2013, based on both the Wang, et. al, attack and the potential for brute-force attack. To ensure that practitioners have secure and efficient hash algorithms to provide long-term security, NIST organized an international competition to select a new hash algorithm standard, SHA-3, which is specified in FIPS 202.

Government and industry have made great strides to migrate from SHA-1 to the stronger hash algorithms in the SHA-2 and SHA-3 families. Those who have not done so yet should migrate as soon as possible.

The work by the CWI-Google team is the culmination of over a decade of research into the SHA-1 algorithm, beginning with the groundbreaking paper by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu in 2005 that described the first cryptanalytic techniques capable of finding collisions with much less work than brute force. Cryptographers around the world continued to improve upon these techniques. The techniques used by this attack were developed by Marc Stevens, one of the members of the joint CWI-Google team. While all of these researchers have made substantial contributions to the field of cryptography, today we recognize the work by these Google-CWI team members who made the challenging jump from theory to a practical demonstration of an attack:

Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), Yarik Markov (Google), Alex Petit Bianco (Google), Clement Baisse (Google)

The research team has posted additional information at [Shattered.io](http://Shattered.io).

<sup>7</sup>信息来自于网址<https://csrc.nist.gov/News/2017/Research-Results-on-SHA-1-Collisions>

# 第8章 消息认证码 (Message Authentication Code)

---

消息认证码是用于消息认证的，消息认证是一个过程，用来保证接受到消息的真实性(是宣称者发来的)和消息的完整性(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)。

## 8.1 MAC 定义

The message authentication code (MAC) is generated from an associated message as a method for assuring the integrity of the message and the authenticity of the source of the message. A secret key to the generation algorithm must be established between the originator of the message and its intended receiver(s). Currently, there are three (3) approved\* general purpose MAC algorithms: HMAC, KMAC and CMAC.<sup>1</sup>

$M$  是消息， $k$  是密钥， $C_k$  是加密函数， $t$  是 MAC(也有称为 tag 的)， $t = C_k(M)$ ，A 要发送  $M$  给 B，其将  $M||t$  发送给 B，B 收到  $M'||t'$  后，计算  $C_k(M')$ ，如果  $t' = C_k(M')$ ，表示消息是 A 发的(密钥只有他俩知道)，并且没有被修改，否则表示消息有问题<sup>2</sup>。

## 8.2 MAC 函数要求

假设敌手知道  $C$ ，但不知道密钥  $k$ ，那么整个 MAC 系统应该满足：

- 敌手知道  $M$  和  $t$ ，构造一个  $M'$ ，满足  $C_k(M') = t$ ，计算上不可行。
- 随机选取两个消息  $M$  和  $M'$ ，两个消息的 MAC 相等的概率为  $2^{-n}$ ，其中， $n$  为 MAC 的比特长度，用公式来表示为  $P[C_k(M) = C_k(M')] = 2^{-n}$ .
- 若  $M'$  是  $M$  的某个变换，也就是说  $M' = f(M)$ ，那么  $P[C_k(M) = C_k(M')] = 2^{-n}$ 。

## 8.3 基于 DES 的 MAC

可以用对称加密的算法设计 MAC 码，利用 DES 算法的 CBC 模式构造 MAC，已被美国的 FIPS(Federal Information Processing Standards) 采用，FIPS PUB 113，也被美国的 ANSI(American National Standards Institute ) 采用，也是 ANSI 的 X9.17 标准。通过 DES 的 CBC(Cipher Block Chaining) 工作模式对信息加密后，初始向量为 0 向量，MAC 取最后一次的加密输出，或者加密输出的最左  $M$  个比特 ( $64 \geq M \geq 16$ )。数据填充分组后为  $D_1, D_2, \dots, D_N$ ，认证码计算过程可以描述为：

$$O_1 = DES_K(D_1)$$

<sup>1</sup>Copy from <https://csrc.nist.gov/projects/message-authentication-codes>

<sup>2</sup>大家可以考虑，这里问题指的是什么？完整性和真实性。

$$\begin{aligned}
 O_2 &= DES_K(D_2 \oplus O_1) \\
 O_3 &= DES_K(D_3 \oplus O_2) \\
 &\dots\dots \\
 O_N &= DES_K(D_N \oplus O_{N-1}) \\
 DAC &= L_i(O_N)
 \end{aligned}$$

DAC 是 Data Authentication Code 的缩写,  $L_i(X)$  表示 X 的最左 i 个比特。

认证码的验证过程是, 当接收者收到消息  $\hat{M}$  和 DAC 后, 根据  $\hat{M}$  计算认证码  $\hat{DAC}$ , 如果  $DAC = \hat{DAC}$ , 认证通过。

## 8.4 基于哈希函数的 MAC(HMAC)

HMAC 是密钥相关的哈希运算消息认证码 (Hash-based Message Authentication Code) 的缩写, 就是利用哈希函数构造 MAC 的方法, HMAC 可以与任何哈希函数 (MD5、SHA 等) 捆绑使用, 是由 H.Krawczyk, M.Bellare, R.Canetti 于 1996 年提出的一种基于 Hash 函数和密钥进行消息认证的方法, 并于 1997 年作为 RFC 2104<sup>3</sup>被公布, 并在 IPSec 和其他网络协议 (如 SSL) 中得以广泛应用, 现在已经成为事实上的 Internet 安全标准。<sup>4</sup>

我们对照看看 NIST 在 HMAC 引言部分的文字。

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such integrity checks based on a secret key are usually called message authentication codes (MACs). Typically, message authentication codes are used between two parties that share a secret key in order to authenticate information transmitted between these parties. This Standard defines a MAC that uses a cryptographic hash function in conjunction with a secret key. This mechanism is called HMAC [HMAC]. HMAC shall use an Approved cryptographic hash function [FIPS 180-3]. HMAC uses the secret key for the calculation and verification of the MACs.<sup>5</sup>

### 8.4.1 HMAC 的安全目标

HMAC 的安全目标我们直接引用 RFC 2104 中的描述。

HMAC can be used in combination with any iterated cryptographic hash function. MD5 and SHA-1 are examples of such hash functions. HMAC also uses a secret key for calculation and verification of the message authentication values. The main goals behind this construction are

<sup>3</sup>文档 URL 地址为: <https://datatracker.ietf.org/doc/html/rfc2104>

<sup>4</sup>信息引用自<https://baike.baidu.com/item/hmac/7307543?fr=aladdin>

<sup>5</sup>引自 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION (FIPS PUB) 198-1 The Keyed-Hash Message Authentication Code (HMAC) ,July 2008 中的 1.Introduction

- \* To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available.
- \* To preserve the original performance of the hash function without incurring a significant degradation.
- \* To use and handle keys in a simple way.
- \* To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions on the underlying hash function.
- \* To allow for easy replaceability of the underlying hash function in case that faster or more secure hash functions are found or required.

## 8.4.2 HMAC 方法

$M$  是根据哈希函数  $H$  的要求填充后的消息，对此消息进行分组，每组  $b$  位，共分了  $L$  组， $m_1, m_2, \dots, m_L$ ， $n$  为此哈希函数输出长度。

### 8.4.2.1 认证密钥

选取一个认证密钥  $K$ ，认证密钥  $K$  可以为任意长度，如果  $K$  长度大于明文分组的长度  $b$ ，则将  $K$  进行哈希运算，产生长度为  $n$  的新密钥，如果  $K$  小于分组长度，可以进行填充，在 HMAC 中直接填充 0，填充到长度为  $b$ 。参与实际后续运算的认证密钥记为  $K^+$ 。

### 8.4.2.2 HMAC 计算过程

HMAC 的计算过程，我们用伪代码表示为：

```

ipad=00110110
opad=01011010
 $S_i = K^+ \oplus ipad$ ; 按位异或
 $NM1 = S_i || m_1 || m_2 || \dots || m_L$ 
 $v_1 = H(IV, NM1)$ 
 $NM2 = v_1$  填充到长度为  $b$ 
 $S_0 = K^+ \oplus opad$ ; 按位异或
 $NM3 = S_0 || NM2;$ 
 $v_2 = H(IV, NM3)$ 
算法输出  $v_2$ 

```

如果我们在 HMAC 中选用 MD5，即  $H$  为  $H_{MD5}$ ，那么  $b=512$  bit,  $n=128$  bit,  $IV=(W,X,Y,Z)$ 。

# 第 9 章 数字签名 (Digital Signatures)

---

数字签名是利用密码技术，达到在数字通信中与纸质签名类似的效果。

下面是 NIST 对于数字签名 (digital signature) 的概述<sup>1</sup>。

As an electronic analogue of a written signature, a digital signature provides assurance that:

- the claimed signatory signed the information, and
- the information was not modified after signature generation.

Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard (DSS), specifies three NIST-approved digital signature algorithms: DSA, RSA, and ECDSA. All three are used to generate and verify digital signatures, in conjunction with an approved hash function specified in FIPS 180-4, Secure Hash Standard or FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.

## 9.1 基于对称密钥算法的数字签名

首先看到很多帖子说“数字签名（又称公钥数字签名）是只有信息的发送者才能产生的别人无法伪造的一段数字串，这段数字串同时也是对信息的发送者发送信息真实性的一个有效证明。”<sup>2</sup> 这种说法有点偏颇，只是现在数字签名很多场合都在使用公钥密码算法来实现，我们这里加了这一小节，我们引用卿斯汉老师书 [2] 中一段话，概念性地给大家介绍一下对称密码算法的签名方案。

" R. C. Merkle 建议，如果有一个可以信赖的第三方 TTP(trusted third party)，用下面的方法可以用传统的密码系统实现数字签名。A 将自己的一对可逆的秘密变换  $E_A$  和  $D_A$  告诉 TTP，当 A 传送签名的信息 M 给 B 时，A 计算出  $C = D_A(M)$ ，然后将 C 发送给 B。为了验证 C 并得到 M，B 将 C 传送给 TTP。TTP 计算出  $E_A(C) = M$ ，然后通过 B 的秘密变换将 M 传送给 B。用传统的密码系统实现数字签名还有许多其他方法，这里就不再介绍了。"[2]

---

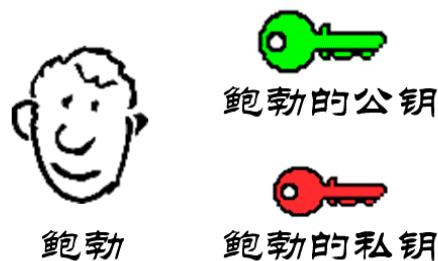
<sup>1</sup><https://csrc.nist.gov/Projects/Digital-Signatures>

<sup>2</sup><https://baike.baidu.com/item/>

## 9.2 数字签名的基本概念示例

此节内容来自阮一峰的网络日志<sup>3</sup>，其翻译了国外一个数字签名的介绍<sup>4</sup>，形象，故此直接引用。

1. 鲍勃有两把钥匙，一把是公钥，另一把是私钥。



2. 鲍勃把公钥送给他的朋友们——帕蒂、道格、苏珊——每人一把。



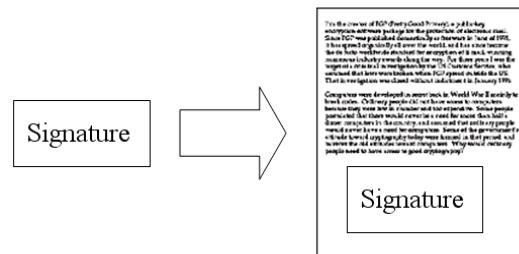
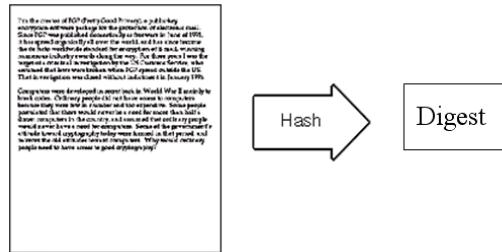
3. 苏珊要给鲍勃写一封保密的信。她写完后用鲍勃的公钥加密，就可以达到保密的效果。



4. 鲍勃收信后，用私钥解密，就看到了信件内容。这里要强调的是，只要鲍勃的私钥不泄露，这封信就是安全的，即使落在别人手里，也无法解密。
5. 鲍勃给苏珊回信，决定采用“数字签名”。他写完后先用 Hash 函数，生成信件的摘要（digest）。
6. 然后，鲍勃使用私钥，对这个摘要加密，生成“数字签名”（signature）。
7. 鲍勃将这个签名，附在信件下面，一起发给苏珊。

<sup>3</sup>阮一峰的网络日志为[http://www.ruanyifeng.com/blog/2011/08/what\\_is\\_a\\_digital\\_signature.html](http://www.ruanyifeng.com/blog/2011/08/what_is_a_digital_signature.html)

<sup>4</sup>What is a Digital Signature? An introduction to Digital Signatures, by David Youd, url is <http://www.youdzone.com/signature.html>

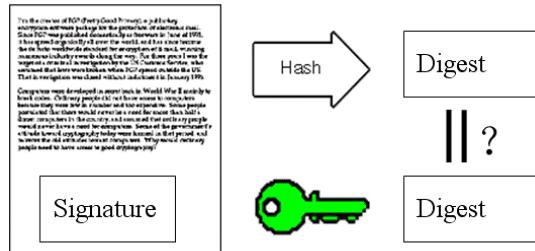


- 苏珊收信后，取下数字签名，用鲍勃的公钥解密，得到信件的摘要。由此证明，这封信确实是鲍勃发出的。



- 苏珊再对信件本身使用 Hash 函数，将得到的结果，与上一步得到的摘要进行对比。如果两者一致，就证明这封信未被修改过。

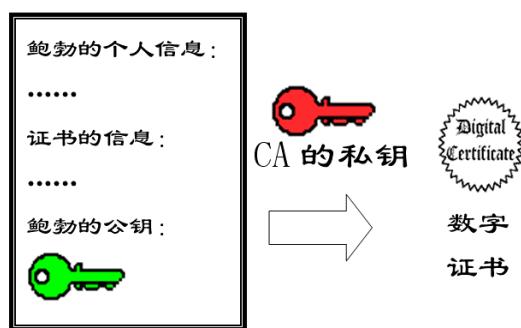




10. 复杂的情况出现了。道格想欺骗苏珊，他偷偷使用了苏珊的电脑，用自己的公钥换走了鲍勃的公钥。此时，苏珊实际拥有的是道格的公钥，但是还以为这是鲍勃的公钥。因此，道格就可以冒充鲍勃，用自己的私钥做成"数字签名"，写信给苏珊，让苏珊用假的鲍勃公钥进行解密。



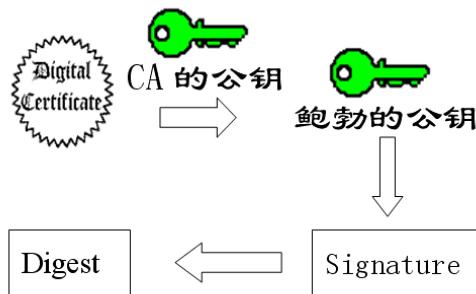
11. 后来，苏珊感觉不对劲，发现自己无法确定公钥是否真的属于鲍勃。她想到了一个办法，要求鲍勃去找"证书中心"（certificate authority，简称 CA），为公钥做认证。证书中心用自己的私钥，对鲍勃的公钥和一些相关信息一起加密，生成"数字证书"（Digital Certificate）。



12. 鲍勃拿到数字证书以后，就可以放心了。以后再给苏珊写信，只要在签名的同时，再附上数字证书就行了。



- 苏珊收信后，用 CA 的公钥解开数字证书，就可以拿到鲍勃真实的公钥了，然后就能证明“数字签名”是否真的是鲍勃签的。



## 9.3 证书实例

下图就是我们用 OpenSSL 生成了 BUU 的 CA 公私钥证书，然后再用 OpenSSL 生成个人的公私钥对，用 BUU CA 颁发的证书 (.cer) 文件中的内容，图9.1是用 Thunderbird 内的证书查看工具看到的内容。



目前常用的证书相关标准是 ITU-T 的 PKI(Public key Infrastructure) 系列标准 X.509，下面我给出一个英文帖子“PEM, DER, CRT, and CER: X.509 Encodings and Conversions”<sup>5</sup>作为大家的阅读资料，同时了解一下证书的格式。

### 9.3.1 PEM, DER, CRT, and CER: X.509 Encodings and Conversions

You may have seen digital certificate files with a variety of filename extensions, such as .crt, .cer, .pem, or .der. These extensions generally map to two major encoding schemes for X.509 certificates and keys: PEM (Base64 ASCII), and DER (binary). However, there is some overlap and other extensions are used, so you can't always tell what kind of file you are working with just from looking at the filename; you may need to open it in a text editor and take a look for yourself.

As you work with digital certificates, you may find yourself with the need to convert between PEM and DER files, view their contents as human-readable text, or combine them into common container formats like PKCS#12 or PKCS#7. This guide points out the major differences between PEM and DER files and common filename extensions associated with them. It also provides visual examples of each encoding, and illustrates some common file format conversions with OpenSSL.

#### 9.3.1.1 What is OpenSSL?

OpenSSL is a very useful open-source command-line toolkit for working with X.509 certificates, certificate signing requests (CSRs), and cryptographic keys. If you are using a UNIX variant like Linux or macOS, OpenSSL is probably already installed on your computer. If you would like to use OpenSSL on Windows, you can enable Windows 10's Linux subsystem or install Cygwin.

#### 9.3.1.2 PEM

PEM (originally “Privacy Enhanced Mail”) is the most common format for X.509 certificates, CSRs, and cryptographic keys. A PEM file is a text file containing one or more items in Base64 ASCII encoding, each with plain-text headers and footers (e.g. —BEGIN CERTIFICATE— and —END CERTIFICATE—). A single PEM file could contain an end-entity certificate, a private key, or multiple certificates forming a complete chain of trust. Most certificate files downloaded from SSL.com will be in PEM format.

##### PEM Filename Extensions

PEM files are usually seen with the extensions .crt, .pem, .cer, and .key (for private keys), but you may also see them with different extensions. For example, the SSL.com CA bundle file

<sup>5</sup>帖子的 URL 地址为<https://www.ssl.com/guide/pem-der-crt-and-cer-x-509-encodings-and-conversions/>

available from the download table in a certificate order has the extension .ca-bundle.

### What does a PEM certificate look like?

The SSL/TLS certificate for www.ssl.com is shown below in PEM format :

```
-----BEGIN CERTIFICATE-----
MIIE/TCCBeWgAwIBAgIQaBYE3/M08XHYCnNVmcFBcjANBhkqkG9w0BAQsFADBy
MQswCQYDVQQGEwJVUzEOMAwGA1UECAwFVGV4YXMxEDAOBgNVBAcMB0hvdXN0b24x
ETAPBgNVBAoMCFNTTCBDb3JwMS4wLAYDVQQDCVTU0wuY29tIEVWIFNTTCBJbnR1
cm1lZG1hdGUgQOEgU1NBIFIzMB4XDTIwMDQwMTAwNTgzM1oXDTIxMDcxNjAwNTgz
M1owgb0xCzAJBgNVBAYTA1VTMQ4wDAYDVQQIDAVUZXhhczEQMA4GA1UEBwwHSG91
c3RvbjERMA8GA1UECgwIU1NMIEvncnAxFjAUBgNVBAUTDU5WMjAwODE2MTQyNDMx
FDASBgNVBAMMC3d3dy5zc2wuY29tMR0wGwYDVQQPDBRQcm1lYXR1IE9yZ2FuaXph
dG1vbjEXMBUGCysGAQQBgjc8AgECDAZ0ZXZhZGEExEzARBgsrBgeEAYI3PAIBAxMC
VVMwggEiMA0GCSqGSIB3DQEBAQUAA4IBDwAwggEKAoIBAQDHheRkbb1FCc7xRKst
wKOJIGaKY8t7JbS2bQ2b6YIJDgnHuIYHqBrCUV79oelikkokRkFvcvpaKinFHDQH
UpWEI6RUERYmSCg308Wi42u0cV2B5ZabmXCKwdxY5Ec151BbM8UnGdoAGbdNmirm
SmTjcs+lhMxg4fFY61BpiEVFiGUjGRR+61R67Lz6U4KJeLnCcM07QwFYKBmpio8g
dygSvRdUw55Jopredj+VGtjUkB4hFT4GQX/ght69R1qz/+8u0dEQkhuUuucrqalm
SGy43HRwBfDKFwYeWm7CPMd5e/d0+t08t8PbjzVTTv5hQDCsEYIV2T7AFI9ScNxM
kh7/AgMBAAGjggNBMIIDPTAfBgNVHSMEGDAwBS/wVqH/yj6QT39t0/kHa+gYVgp
vTB/BggrBgEFBQcBAQRzMHEwTQYIKwYBBQUHMAKGQWh0dHA6Ly93d3cuc3NsLmNv
bS9yZXBvc210b3J5L1NTTGNvbS1TdWJDQS1FVi1TU0wtU1NBLTQwOTYtUjMuY3J0
MCAGCCsGAQUFBzABhhRodHRw0i8vb2NzcHMuc3NsLmNvbTAfBgNVHREEGDAwgg73
d3cuc3NsLmNvbYIHc3NsLmNvbTBfBgvNVHSAEWDBWMAcGBWeBDAEBMAOGCyqEaAGG
9ncCBQEBMDwGDCsGAQQBgqkwAQMBBDAsMCoGCCsGAQUFBwIBFh5odHRwczovL3d3
dy5zc2wuY29tL3J1cG9zaXRvcnkwhQYDVR01BBwFAYIKwYBBQUHAWIGCCsGAQUF
BwMBMEdGA1UdHwRBMD8wPaA7oDmGN2h0dHA6Ly9jcmxzLnNzbC5jb20vU1NMY29t
LVN1YkNBLUVWLNTTC1SU0EtNDA5Ni1SMy5jcmwwHQYDVR0OBBYEFADAFUIazw5r
ZIHapnRxIUnpw+GLMA4GA1UdDwEB/wQEAvIFoDCCAX0GCisGAQQB1nkCBAIEggFt
BIIBaQFnAHcA91yUL9F3MCUVBgiMJRWjuNNExkzv98MLyALzE7xZOMAAAFxM0ho
bwAABAMASDBGAiEA6xeliNR8Gk/63pYdnS/v0x/CjptEMEv89WWh1/urWIECIQDy
BreHU25DzwukQaRQjwW655ZLkqCnxbxQWRi0emj9JAB1AJQgvB601Y1siHMfgosi
LA3R2k1ebE+UPWhbTi9YTaLCAAAcTNiNwAAAQDAEYwRAIgGRE4wzabNRdD8kq/
vFP3tQe2hm0x5nXuloh4Ibw3lkCIFYb/31SDplS7AcR4r+XpWtEKSTFWJmNCRbc
XJur2RGBAHUA7sCV7o1yZA+S4805G8cSo21qCXtLahoU00ZHssvtxfkAAAFxM0ho
8wAABAMARjBEAiB6Ivb0Wss3R4ItVwjeb17D3yoFaXONDh2dWhhgwCxrHwIgCfq7
ocMC5t+1ji5M5xaLmPC4I+WX3I/ARkWSyi07IQcwDQYJKoZIhvcNAQELBQADggIB
ACeuur4QnujqmguSrHU3mhfc+cJodzTQNqo4tde+PD1/eFdYAEIu8xF+0At7xJiPY
i5RKwilyP56v+3iY2T91w7S8TJ041VLhaIKp14MzSUzRyeo0AsJ7QADMClHKUD1H
UU2pNuo88Y6igovT3bsnwJN1EQNqymSSYhktw0taduoqjqXn06gsVioWTDXysd5
qEx4t6sIgIcMm26YH1vJpCQEhKpc2y07gRkk1BZRtMjThv4cXyyMX7uTcdT7AJBP
ueifCoV25JxXuo8d5139gwP1BAe7IBVPx2u7KN/Uy0XdZmwMf/TmFGwDdCfsyHf/
ZsB2wLHozTYoAVmQ9FoU1JLgcVivqJ+vN1BhHXh1xMdN0j80R9Nz6EIglQjeK308
I/cFGm/B8+42h01CID9ZdtndJcRJVj10wD0qwevCafA9jJ1Hv/jsE+I9Uz6cpCyh
```

```
sw+lrFdxUgqU58axqeK89FR+No4q0II0+Ji1rJKr9nkSB0BqXozVnE1YB/KLvdIs
uYZJuqb2pKku+zzT6gUwHUTZvBiN0tXL4Nxwc/KT7Wz0Sd2wP10QI8DKg4vfiNDs
HWmB1c4Kji6g0gA5uSUzaGmq/v4VncK5Ur+n9LbfnfLc28J5ft/GotinMyDk3iar
F10Ylqc0meX1uFmKbdi/XorGlkCoMF3TDx8rmp9DBiB/
-----END CERTIFICATE-----
```

### 9.3.1.3 DER

DER (Distinguished Encoding Rules) is a binary encoding for X.509 certificates and private keys. Unlike PEM, DER-encoded files do not contain plain text statements such as —BEGIN CERTIFICATE—. DER files are most commonly seen in Java contexts.

#### DER Filename Extensions

DER-encoded files are usually found with the extensions .der and .cer.

#### What does a DER-encoded certificate look like?

The DER-encoded SSL/TLS certificate for www.ssl.com is shown below (click to view):

```
3082 07fd 3082 05e5 a003 0201 0202 1068
1604 dff3 34f1 71d8 0a73 5599 c141 7230
0d06 092a 8648 86f7 0d01 010b 0500 3072
310b 3009 0603 5504 0613 0255 5331 0e30
0c06 0355 0408 0c05 5465 7861 7331 1030
0e06 0355 0407 0c07 486f 7573 746f 6e31
1130 0f06 0355 040a 0c08 5353 4c20 436f
7270 312e 302c 0603 5504 030c 2553 534c
2e63 6f6d 2045 5620 5353 4c20 496e 7465
726d 6564 6961 7465 2043 4120 5253 4120
5233 301e 170d 3230 3034 3031 3030 3538
3333 5a17 0d32 3130 3731 3630 3035 3833
335a 3081 bd31 0b30 0906 0355 0406 1302
5553 310e 300c 0603 5504 080c 0554 6578
6173 3110 300e 0603 5504 070c 0748 6f75
7374 6f6e 3111 300f 0603 5504 0a0c 0853
534c 2043 6f72 7031 1630 1406 0355 0405
130d 4e56 3230 3038 3136 3134 3234 3331
1430 1206 0355 0403 0c0b 7777 772e 7373
6c2e 636f 6d31 1d30 1b06 0355 040f 0c14
5072 6976 6174 6520 4f72 6761 6e69 7a61
7469 6f6e 3117 3015 060b 2b06 0104 0182
373c 0201 020c 064e 6576 6164 6131 1330
1106 0b2b 0601 0401 8237 3c02 0103 1302
5553 3082 0122 300d 0609 2a86 4886 f70d
0101 0105 0003 8201 0f00 3082 010a 0282
0101 00c7 85e4 646d bd45 09ce f144 ab2d
```

```
c0ad 0920 668a 63cb 7b25 b4b6 6d0d 9be9  
8209 0e09 c7b8 8607 a81a c251 5efd a1e9  
6292 4a24 4641 6f72 fa5a 2a29 c51c 3407  
5295 8423 a454 1116 2648 2837 3bc5 a2e3  
6b8e 715d 81e5 969b 9970 a4c1 dc58 e447  
25e7 505b 33c5 2719 da00 19b7 4d9a 2466  
4a64 e372 cfa5 84cc 60e1 f158 ea50 6988  
4545 8865 2319 147e eb54 7aec bcfa 5382  
8978 b35c 0a6d 3b43 0158 2819 a98b 4f20  
7728 12bd 1754 c39e 49a2 9ade 763f 951a  
d8d4 901e 2115 3e06 417f e086 debd 465a  
b3ff ef2e d1d1 1092 1b94 bae7 2ba9 a966  
486c b8dc 7470 05f0 ca17 061e 58ce c23c  
c779 7bf7 4efa dd3c b7c3 db8f 3553 4efe  
6140 30ac 1182 15d9 3ec0 148f 5270 dc4c  
921e ff02 0301 0001 a382 0341 3082 033d  
301f 0603 551d 2304 1830 1680 14bf c15a  
87ff 28fa 413d fdb7 4fe4 1daf a061 5829  
bd30 7f06 082b 0601 0505 0701 0104 7330  
7130 4d06 082b 0601 0505 0730 0286 4168  
7474 703a 2f2f 7777 772e 7373 6c2e 636f  
6d2f 7265 706f 7369 746f 7279 2f53 534c  
636f 6d2d 5375 6243 412d 4556 2d53 534c  
2d52 5341 2d34 3039 362d 5233 2e63 7274  
3020 0608 2b06 0105 0507 3001 8614 6874  
7470 3a2f 2f6f 6373 7073 2e73 736c 2e63  
6f6d 301f 0603 551d 1104 1830 1682 0b77  
7777 2e73 736c 2e63 6f6d 8207 7373 6c2e  
636f 6d30 5f06 0355 1d20 0458 3056 3007  
0605 6781 0c01 0130 0d06 0b2a 8468 0186  
f677 0205 0101 303c 060c 2b06 0104 0182  
a930 0103 0104 302c 302a 0608 2b06 0105  
0507 0201 161e 6874 7470 733a 2f2f 7777  
772e 7373 6c2e 636f 6d2f 7265 706f 7369  
746f 7279 301d 0603 551d 2504 1630 1406  
082b 0601 0505 0703 0206 082b 0601 0505  
0703 0130 4806 0355 1d1f 0441 303f 303d  
a03b a039 8637 6874 7470 3a2f 2f63 726c  
732e 7373 6c2e 636f 6d2f 5353 4c63 6f6d  
2d53 7562 4341 2d45 562d 5353 4c2d 5253  
412d 3430 3936 2d52 332e 6372 6c30 1d06  
0355 1d0e 0416 0414 00c0 1542 1acf 0e6b  
6481 daa6 7471 2149 e9c3 e18b 300e 0603
```

```
551d 0f01 01ff 0404 0302 05a0 3082 017d
060a 2b06 0104 01d6 7902 0402 0482 016d
0482 0169 0167 0077 00f6 5c94 2fd1 7730
2214 5418 0830 9456 8ee3 4d13 1933 bfdf
0c2f 200b cc4e f164 e300 0001 7133 4868
6f00 0004 0300 4830 4602 2100 eb17 a588
d47c 1a4f fade 961d 9d2f ef3b 1fc2 8e9b
4430 4bfc f565 a1d7 fbab 5881 0221 00f2
06b7 8753 6e43 cf0b a441 a450 8f05 bae7
964b 92a0 a7c5 bc50 5918 8e7a 68fd 2400
7500 9420 bc1e 8ed5 8d6c 8873 1f82 8b22
2c0d d1da 4d5e 6c4f 943d 61db 4e2f 584d
a2c2 0000 0171 3348 68dc 0000 0403 0046
3044 0220 1911 38c3 369b 3517 43f2 4abf
bc53 f7b5 07b6 866d 31e6 75ee 968c 21e0
86f0 de59 0220 561b ff79 520e 9952 ec07
11e2 bf97 a56b 4429 24c5 5899 8d09 16dc
5c9b abd9 1181 0075 00ee c095 ee8d 7264
0f92 e3c3 b91b c712 a369 6a09 7b4b 6a1a
1438 e647 b2cb edc5 f900 0001 7133 4868
f300 0004 0300 4630 4402 207a 22f6 e85a
cb37 4782 2d57 08de 6e5e c3df 2a05 697d
0d0e 1d9d 5a18 60c0 2c6b 1f02 2009 fabb
a1c3 02e6 dfb5 8e2e 4ce7 168b 98f0 b823
e597 dc8f c046 4592 ca23 bb21 0730 0d06
092a 8648 86f7 0d01 010b 0500 0382 0201
0027 aeba be10 9ee8 ea9a 0b92 ac75 379a
17fe 709a 1dc0 340d aa8e 2d75 ef8f 0f5f
de15 d600 10bb bcc4 5fb4 02de f126 23d8
8b94 4ac2 2972 3f9e affb 7898 d93f 65c3
b4bc 4c9d 38d5 52e1 6882 a9d7 8333 494c
d1c9 ea0e 02c2 7b40 00cc 0a51 ca50 3947
514d a936 ea3c f18e a282 8bd3 ddbb 27c0
9362 1103 6aca 6492 6219 2dc3 4b5a 76ea
2a8e a5e7 d3a8 2c56 2a16 4d50 d7ca c779
a84c 78b7 ab08 8087 0c9b 6e98 1f5b c9a4
2404 84aa 5cdb 2d3b 8119 2494 1651 b4c8
d386 fe1c 5f2c 8c5f bb93 71d4 fb00 904f
b9e8 9f0a 8576 e49c 57ba 8f1d e75d fd83
03f5 0407 bb20 154f c76b bb28 dfd4 c8e5
dd66 6c0c 7ff4 e614 6c03 7427 ecc8 77ff
66c0 76c0 b1e8 cd36 2801 5990 f45a 14d4
92e0 7158 afa8 9faf 3650 611d 7865 c4c7
```

```
4dd2 3f34 47d3 73e8 4220 9508 de2b 73bc
23f7 051a 6fc1 f3ee 3684 e942 21df 5976
d9dd 25c4 4956 38b4 c03d 2ac1 ebc2 69f0
3d8c 9947 bff8 ec13 e23d 533e 9ca4 2ca1
b30f a5ac 5771 520a 94e7 c6b1 a9e2 bcf4
547e 368e 2ad0 820e f898 b5ac 92ab f679
1207 406a 5e8c d59c 4d58 07f2 8bbd d22c
b986 49ba a6f6 a4a9 2efb 3cd3 ea05 301d
44d9 bc18 8d3a d5cb e0dc 7073 f293 ed6c
ce49 ddb0 3f5d 1023 c0ca 838b df88 d0ec
1d69 81d5 ce0a 8e2e a03a 0039 b925 3368
69aa fefe 159d c2b9 52bf a7f4 b6df 9df2
dcdb c279 7edf c6a2 d8a7 3320 e4de 26ab
175d 1896 a70e 99e5 f5b8 598a 6dd8 bf5e
8ac6 9640 a830 5dd3 0f1f 2b9a 9f43 0620
7f
```

### 9.3.1.4 Convert PEM certificate with chain of trust to PKCS#7

PKCS#7 (also known as P7B) is a container format for digital certificates that is most often found in Windows and Java server contexts, and usually has the extension .p7b. PKCS#7 files are not used to store private keys. In the example below, -certfile MORE.pem represents a file with chained intermediate and root certificates (such as a .ca-bundle file downloaded from SSL.com).

*Example:* openssl crl2pkcs7 -nocrl -certfile CERTIFICATE.pem -certfile MORE.pem -out CERTIFICATE.p7b

### 9.3.1.5 Convert PEM certificate with chain of trust and private key to PKCS#12

PKCS#12 (also known as PKCS12 or PFX) is a common binary format for storing a certificate chain and private key in a single, encryptable file, and usually have the filename extensions .p12 or .pfx. In the example below, -certfile MORE.pem adds a file with chained intermediate and root certificates (such as a .ca-bundle file downloaded from SSL.com), and -inkey PRIVATEKEY.key adds the private key for CERTIFICATE.crt(the end-entity certificate). Please see this how-to for a more detailed explanation of the command shown.

*Example:* openssl pkcs12 -export -out CERTIFICATE.pfx -inkey PRIVATEKEY.key -in CERTIFICATE.crt -certfile MORE.crt

After executing the command above you will be prompted to create a password to protect the PKCS#12 file. Remember this password. You will need it to access any certificates and keys stored in the file.

### 9.3.1.6 Convert DER-encoded certificate with chain of trust and private key to PKCS#12

To convert a DER certificate to PKCS#12 it should first be converted to PEM, then combined with any additional certificates and/or private key as shown above. For a more detailed description of converting DER to PKCS#12, please see [this how-to](#)<sup>6</sup>.

### 9.3.2 证书服务公司

基于PKI体系，在我国目前有很多提供数字证书服务的公司，这些公司在支撑电子政务的同时，也向社会提供服务，比如北京数字认证股份有限公司（网址<https://www.bjca.cn/>）、南京数字认证有限公司（网址<http://www.njca.com.cn/>）、上海市数字证书认证中心有限公司（公司网址<https://www.sheca.com/>）等，还有提供证书服务的公司有阿里云、天威诚信、TurstAsia等。

---

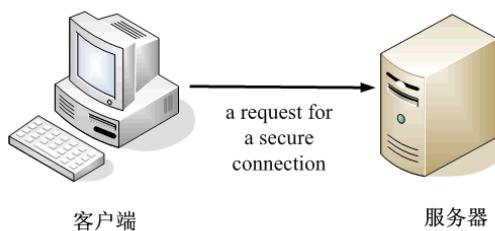
<sup>6</sup>Create a .pfx/.p12 Certificate File Using OpenSSL, 网络地址<https://www.ssl.com/how-to/create-a-pfx-p12-certificate-file-using-openssl/>

## 9.4 Https 实例

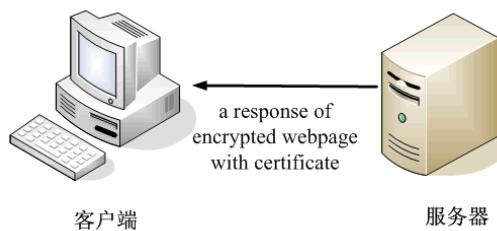
我们经常会用到 https 协议，这个协议主要用于网页加密，这就是一个应用" 数字证书" 的实例。以下这个例子来自于阮一峰的网络日志<sup>7</sup>。



1. 首先，客户端向服务器发出加密请求。

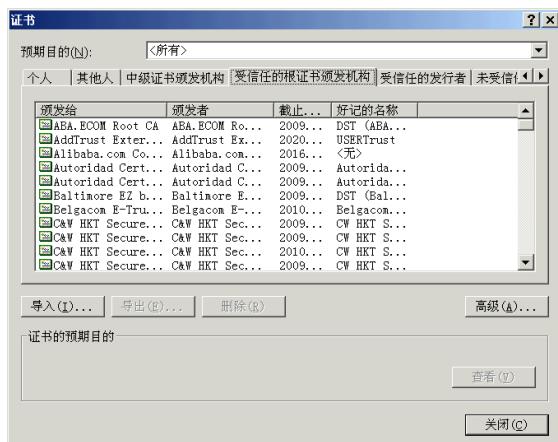


2. 服务器用自己的私钥加密网页以后，连同本身的数字证书，一起发送给客户端。



3. 客户端（浏览器）的"证书管理器"，有"受信任的根证书颁发机构"列表。客户端会根据这张列表，查看解开数字证书的公钥是否在列表之内。

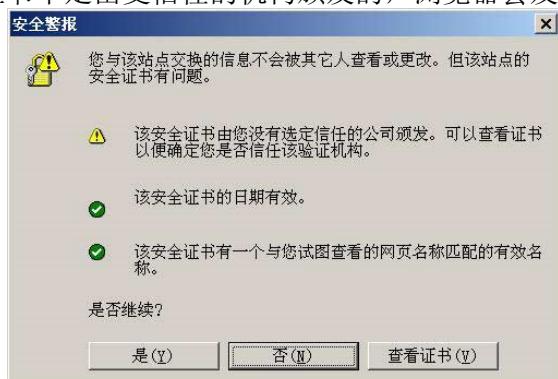
<sup>7</sup>阮一峰的网络日志为[http://www.ruanyifeng.com/blog/2011/08/what\\_is\\_a\\_digital\\_signature.html](http://www.ruanyifeng.com/blog/2011/08/what_is_a_digital_signature.html)



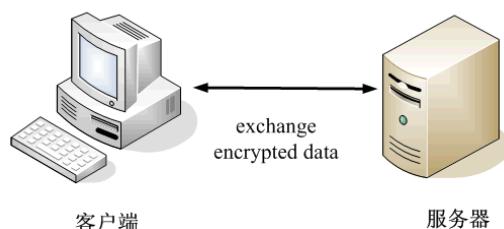
4. 如果数字证书记载的网址，与你正在浏览的网址不一致，就说明这张证书可能被冒用，浏览器会发出警告。



5. 如果这张数字证书不是由受信任的机构颁发的，浏览器会发出另一种警告。



6. 如果数字证书是可靠的，客户端就可以使用证书中的服务器公钥，对信息进行加密，然后与服务器交换加密信息。



## 9.5 Https 标准

Https 的意思是“HTTP Over TLS”，此协议是 IETF Request for Comments: 2818 HTTP Over TLS。TLS 是 SSL 协议的升级版，SSL/TLS 是介于 Https 和 TCP 之间的协议。Https 是 IETF 的 RFC 2818 “HTTP Over TLS” 标准，利用 TLS 层提供的服务进行安全通信，如果看 RFC2818 标准，你会看到描述很简单，详细的初始化、密钥交换协议都是在 TLS 层实现，TLS 也是 IETF 的标准，标准号为 RFC 8446，标准名为“The Transport Layer Security (TLS) Protocol Version 1.3”，我们可以看到在这个标准中，认证用的是公钥体制，加密用的对称加密体制，在协议中定义了 AES 加密选项，密钥交换使用的是 Diffie-Hellman 协议。

## 9.6 签名方案与标准

### 9.6.1 DSS

DSS(Digital Signature Standard) 是美国的国家标准，目前最新发布的是 2013 版<sup>8</sup>，其封皮如图9.2。在 DSS 2013 版中，对 DSS 概要描述见图9.3。而 DSS 的基本过程如图9.4。



图 9.2: 2013 年发布的 DSS

#### 1. Introduction

This Standard defines methods for digital signature generation that can be used for the protection of binary data (commonly called a message), and for the verification and validation of those digital signatures. Three techniques are approved.

- (1) The Digital Signature Algorithm (DSA) is specified in this Standard. The specification includes criteria for the generation of domain parameters, for the generation of public and private key pairs, and for the generation and verification of digital signatures.
- (2) The RSA digital signature algorithm is specified in American National Standard (ANS) X9.31 and Public Key Cryptography Standard (PKCS) #1. FIPS 186-4 approves the use of implementations of either or both of these standards and specifies additional requirements.
- (3) The Elliptic Curve Digital Signature Algorithm (ECDSA) is specified in ANS X9.62. FIPS 186-4 approves the use of ECDSA and specifies additional requirements. Recommended elliptic curves for Federal Government use are provided herein.

This Standard includes requirements for obtaining the assurances necessary for valid digital signatures. Methods for obtaining these assurances are provided in NIST Special Publication (SP) 800-89, *Recommendation for Obtaining Assurances for Digital Signature Applications*.

图 9.3: 2013 年发布的 DSS 中的 Introduction 部分

<sup>8</sup>最新版可以从美国 NIST 网站上下载，地址为 <https://www.nist.gov/publications/digital-signature-standard-dss-2>

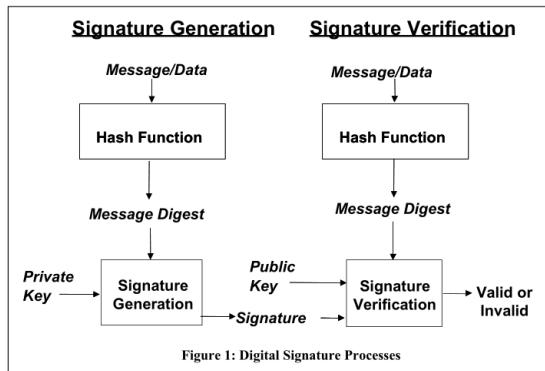


图 9.4: 2013 年发布的 DSS 中的 Introduction 部分

从文档描述中我们可看到, DSS 标准支持三种签名算法: DSA(Digital Signature Algorithm)、RSA Digital Signature Algorithm、Elliptic Curve Digital Signature Algorithm (ECDSA)。

FIPS 186-2 Specification for the DIGITAL SIGNATURE STANDARD(DSS) 对于数据签名算法(简写为 ds)的一般性描述如下:

A ds algorithm is used by a signatory to generate a digital signature on data and by a verifier to verify the authenticity of the signature. Each signatory has a public and private key. The private key is used in the signature generation process and the public key is used in the signature verification process. For both signature generation and verification, the data which is referred to as a message, M, is reduced by means of the Secure Hash Algorithm (SHA-1) specified in FIPS 180-1. An adversary, who does not know the private key of the signatory, cannot generate the correct signature of the signatory. In other words, signatures cannot be forged. However, by using the signatory's public key, anyone can verify a correctly signed message. A means of associating public and private key pairs to the corresponding users is required. That is, there must be a binding of a user's identity and the user's public key. This binding may be certified by a mutually trusted party. For example, a certifying authority could sign credentials containing a user's public key and identity to form a certificate. Systems for certifying credentials and distributing certificates are beyond the scope of this standard. NIST intends to publish separate document(s) on certifying credentials and distributing certificates.

### 9.6.1.1 DSA<sup>9</sup>

#### 1.DSA 参数 (DSA parameters)

The DSA makes use of the following parameters:

1.  $p =$  a prime modulus, where  $2^{L-1} < p < 2^L$  for  $512 \leq L \leq 1024$  and L a multiple of 64
2.  $q =$  a prime divisor of  $p - 1$ , where  $2^{159} < q < 2^{160}$
3.  $g = h^{(p-1)/q} \pmod{p}$ , where  $h$  is any integer with  $1 < h < p - 1$  such that  $h^{(p-1)/q} \pmod{p} > 1$  ( $g$  has order  $q$  mod  $p$ )
4.  $x =$  a randomly or pseudorandomly generated integer with  $0 < x < q$

<sup>9</sup>摘抄自 FIPS 186-2 Specifications for the DIGITAL SIGNATURE STANDARD (DSS)

5.  $y = g^x \pmod{p}$
6.  $k =$  a randomly or pseudorandomly generated integer with  $0 < k < q$

The integers p, q, and g can be public and can be common to a group of users. A user's private and public keys are x and y, respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature.

Parameters p and q shall be generated as specified in Appendix 2, or using other FIPS approved security methods. Parameters x and k shall be generated as specified in Appendix 3, or using other FIPS approved security methods.

## 2.DSA 签名生成 (DSA signature generation)

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$$\begin{aligned} r &= (g^k \pmod{p}) \pmod{q} \text{ and} \\ s &= (k^{-1}(SHA - 1(M) + xr)) \pmod{q}. \end{aligned}$$

In the above,  $k^{-1}$  is the multiplicative inverse of k, mod q; i.e.,  $(k^{-1}k) \bmod q = 1$  and  $0 < k^{-1} < q$ . The value of SHA-1(M) is a 160-bit string output by the Secure Hash Algorithm specified in FIPS 180-1. For use in computing s, this string must be converted to an integer. The conversion rule is given in Appendix 2.2.

As an option, one may wish to check if  $r = 0$  or  $s = 0$ . If either  $r = 0$  or  $s = 0$ , a new value of k should be generated and the signature should be recalculated (it is extremely unlikely that  $r = 0$  or  $s = 0$  if signatures are generated properly).

The signature is transmitted along with the message to the verifier.

## 3.DSA 签名验证 (DSA signature verification)

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner.

Let  $M'$ ,  $r'$ , and  $s'$  be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verify the signature, the verifier first checks to see that  $0 < r' < q$  and  $0 < s' < q$ ; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes:

$$\begin{aligned} w &= (s')^{-1} \pmod{q} \\ u1 &= ((SHA - 1(M'))w) \pmod{q} \\ u2 &= ((r')w) \pmod{q} \\ v &= (((g)^{u1}(y)^{u2}) \pmod{p}) \pmod{q} \end{aligned}$$

If  $v = r'$ , then the signature is verified and the verifier can have high confidence that the



received message was sent by the party holding the secret key  $x$  corresponding to  $y$ . For a proof that  $v = r'$  when  $M' = M$ ,  $r' = r$ , and  $s' = s$ , see Appendix 1<sup>10</sup>.

If  $v$  does not equal  $r'$ , then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid.

### 9.6.1.2 RSA ds algorithm

DSS 中并没有规范 RSA 签名算法，而是引用了另外两个标准，这两个标准定义了 RSA 签名算法（见图9.5）。

#### 5. The RSA Digital Signature Algorithm

The use of the RSA algorithm for digital signature generation and verification is specified in American National Standard (ANS) X9.31 and Public Key Cryptography Standard (PKCS) #1. While each of these standards uses the RSA algorithm, the format of the ANS X9.31 and PKCS #1 data on which the digital signature is generated differs in details that make the algorithms non-interchangeable.

图 9.5: DSS 中使用的 RSA 签名标准

下面我们简单说一下 RSA 的签名原理。

$pk_a, sk_a$  分别表示  $a$  的公钥和私钥， $H$  表示哈希函数， $E$  表示加密函数， $D$  表示解密函数， $C$  表示一个数的比较函数，相当为真，不等为假， $a \rightarrow b : M$  表示  $a$  把消息  $M$  发给  $b$ ， $a \leftarrow b : M$  表示  $b$  把消息  $M$  发到给  $a$ ， $b : H(M)$  表示在  $b$  端计算  $M$  的哈希值，那么 RSA 签名可以表示为：

1.  $a :: ds = E_{sk_a}(H(M))$
2.  $a \rightarrow b :: M||ds$
3.  $b :: C(D_{pk_a}(ds), H(M))$  为真则签名有效，否则签名无效。

在卿斯汉老师“密码学与计算机网络安全”[2]一书中，讨论了 A 发送给 B 消息时，需要签名并加密：

$$A :: C = E_{pk_B} (D_{sk_A} (M))$$

$$A \rightarrow B :: C$$

B 在收到 C 以后用私钥解密，用 A 的公钥解密，可以起到签名 + 加密，但是在这里其提到，当  $n_A > n_B$  时， $D_{sk_A}(M)$  二进制块可能不在  $[0, n_B - 1]$  的范围之内，将  $D_{sk_A}(M)$  对  $n_B$  取模也解决不了问题，因为取模后无法恢复 M 了。在卿老师的书中介绍了几种解决办法，比如重新分块、阈值 (threshold value) 方法和 Konfelder 方法。

---

<sup>10</sup>这里的 Appendix 1 是标准的 Appendix，不是本讲义的。

第 10 章 密钥管理 (Key Management)

密钥管理是一个很大的话题，其涉及到<sup>1</sup>:

1. 密钥生成
  2. 密钥分发
  3. 验证密钥
  4. 更新密钥
  5. 密钥存储
  6. 备份密钥
  7. 密钥有效期
  8. 销毁密钥

图10.1是来自于简书上一个帖子里的一张图，内容比上面多了一项“密钥使用”其他都一样，只不过叫法略有不同，大家参考对比来学习。

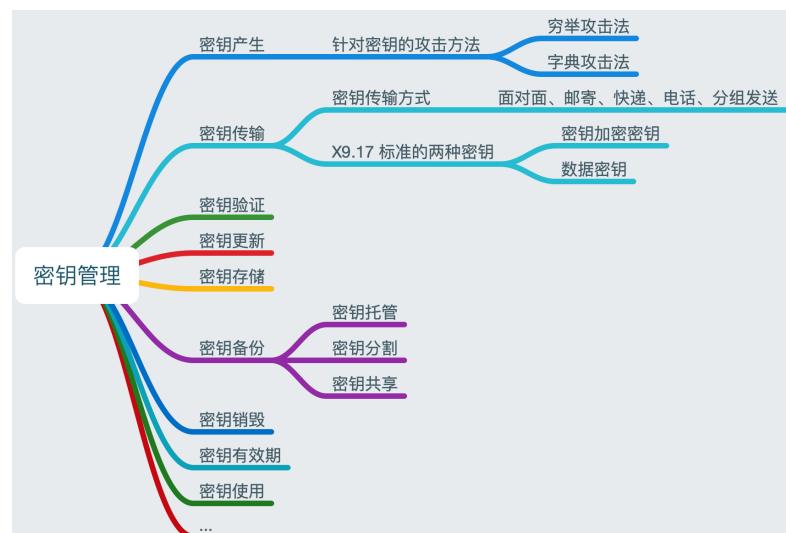


图 10.1: 密钥管理的主要内容<sup>2</sup>

因为密钥的安全在现在的密码体制安全中是决定性的，所以美国的 NIST 也就密钥管理，发布了一些标准，比如 SP 800-57 Recommendation for Key Management: Part 1 – General, Part 2 – Best Practices for Key Management Organizations, Part 3: Application-Specific Key Management Guidance 等，在 SP 800-57 Part1 中，其从几个侧面讨论了密钥管理，可以从 NIST SP 800-57 Part 1 的目录上看到，标准的第 5 部分介绍了一般密钥管理要求，如图10.2所示，第 6 部分介绍了密钥信息的保护需求，如图10.3所示，第 7 部分介绍了密钥生命周期的几个阶段，如图10.4所示，第 8 部分介绍了密钥管理的各个阶段，如图10.5所示，第 9 部分对 CKMS(Cryptographic Key Management System) 系统的安全要求进行了说

<sup>1</sup>对这些密钥管理内容的简单解释，可以参考<https://baike.baidu.com/item/>，百度的这个词条由“科普中国”科学百科词条编写与应用工作项目审核。

明, 如图10.6所示。<sup>3</sup>

<b>5 General Key Management Guidance .....</b>	<b>33</b>
5.1 Key Types and Other Information.....	33
5.1.1 Cryptographic Keys .....	33
5.1.2 Other Related Information .....	36
5.2 Key Usage .....	37
5.3 Cryptoperiods .....	38
5.3.1 Factors Affecting Cryptoperiods.....	38
5.3.2 Consequence Factors Affecting Cryptoperiods .....	39
5.3.3 Other Factors Affecting Cryptoperiods .....	40
5.3.3.1 Communications versus Storage .....	40
5.3.3.2 Cost of Key Revocation and Replacement.....	40
5.3.4 Asymmetric Key Usage Periods and Cryptoperiods .....	40
5.3.5 Symmetric Key Usage Periods and Cryptoperiods.....	41
5.3.6 Cryptoperiod Recommendations for Specific Key Types .....	43
5.3.7 Recommendations for Other Related Information.....	52
5.4 Assurances .....	52
5.4.1 Assurance of Integrity (Integrity Protection).....	53
5.4.2 Assurance of Domain Parameter Validity .....	53
5.4.3 Assurance of Public-Key Validity .....	53
5.4.4 Assurance of Private-Key Possession.....	54
5.4.5 Key Confirmation .....	54
5.5 Compromise of Keys and other Keying Material.....	54
5.5.1 Implications.....	54
5.5.2 Protective Measures.....	56
5.6 Guidance for Cryptographic Algorithm and Key-Size Selection.....	58
5.6.1 Comparable Algorithm Strengths .....	58
5.6.1.1 Security Strengths of Symmetric Block Cipher and Asymmetric-Key Algorithms.....	59
5.6.1.2 Security Strengths of Hash Functions and Hash-based Functions .....	61
5.6.2 Using Algorithm Suites and the Effective Security Strength .....	63
5.6.3 Projected Security Strength Time Frames and Current Approval Status .....	65
5.6.4 Transitioning to New Algorithms and Key Sizes .....	66
Figure 2: Algorithm originator-usage period example .....	67
5.6.5 Decrease of Security Strength Over Time .....	69

图 10.2: NIST SP 800-57 一般密钥管理要求

---

<sup>3</sup>Accountability, 常见的翻译有可检查性、可核查性、可问责、可追溯性

<b>6 Protection Requirements for Key Information.....</b>	<b>71</b>
6.1 Protection and Assurance Requirements .....	71
6.1.1 Summary of Protection and Assurance Requirements for Cryptographic Keys.	72
6.1.2 Summary of Protection Requirements for Other Related Information.....	77
6.2 Protection Mechanisms .....	79
6.2.1 Protection Mechanisms for Key Information in Transit.....	79
6.2.1.1 Availability.....	80
6.2.1.2 Integrity .....	80
6.2.1.3 Confidentiality.....	81
6.2.1.4 Association with Usage or Application.....	82
6.2.1.5 Association with Other Entities.....	82
6.2.1.6 Association with Other Related Information.....	82
6.2.2 Protection Mechanisms for Information in Storage.....	82
6.2.2.1 Availability.....	83
6.2.2.2 Integrity .....	83
6.2.2.3 Confidentiality.....	84
6.2.2.4 Association with Usage or Application.....	84
6.2.2.5 Association with the Other Entities.....	85
6.2.2.6 Association with Other Related Information.....	85
6.2.3 Metadata for Keys.....	85

图 10.3: NIST SP 800-57 密钥信息的保护需求

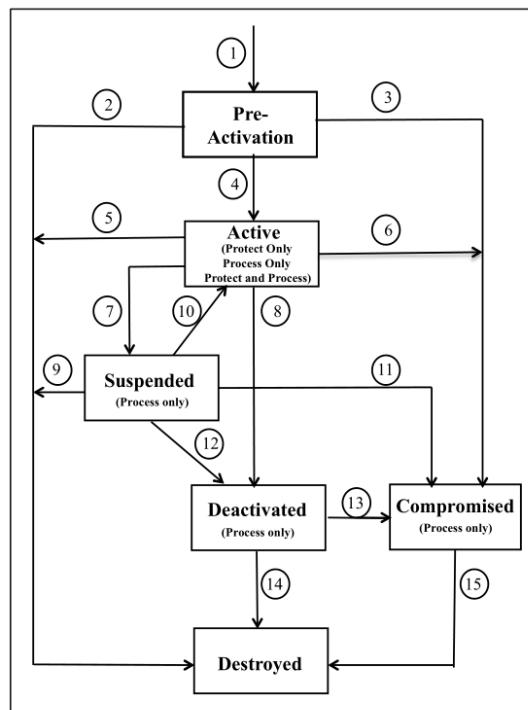
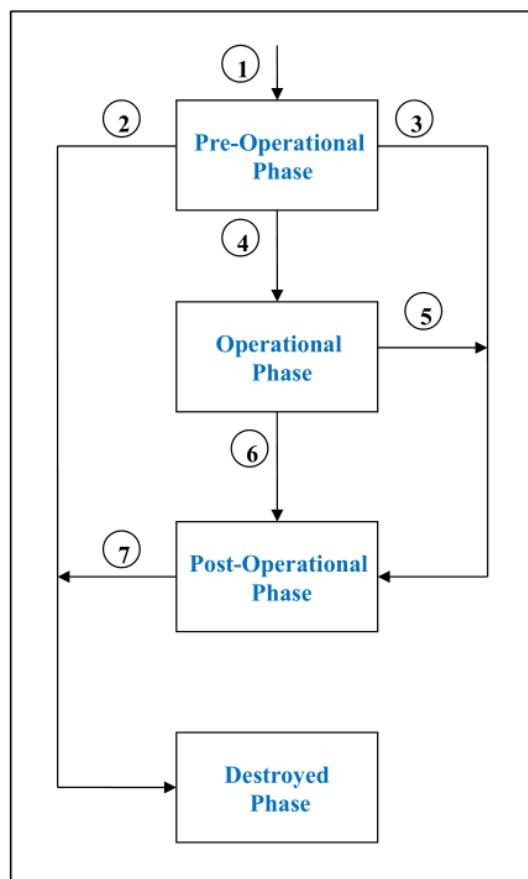


Figure 3: Key state and transition example

图 10.4: NIST SP 800-57 密钥生命周期



**Figure 4: Key-management phases**

图 10.5: NIST SP 800-57 密钥管理的各个阶段

## 9 Accountability, Audit, Survivability and Key-Inventory Management ..... 126

9.1	Access Control and Identity Authentication.....	126
9.2	Accountability .....	126
9.3	Audit .....	127
9.4	Key Management System Survivability .....	128
9.4.1	Backed Up and Archived Keys.....	128
9.4.2	Key Recovery.....	129
9.4.3	System Redundancy/Contingency Planning.....	131
9.4.3.1	General Principles .....	131
9.4.3.2	Cryptography and Key Management-Specific Recovery Issues .....	133
9.4.4	Compromise Recovery.....	133
9.5	Inventory Management.....	134
9.5.1	Key Inventories .....	135
9.5.2	Certificate Inventories.....	135

图 10.6: NIST SP 800-57CKMS 系统的安全要求

## 10.1 密钥分发 (key distribution) 的基本方法

加密算法中密钥非常关键，而密钥或者生成密钥的相关信息是需要在通信双方或多方面之间传递的，那么如何安全地对密钥或者密钥相关信息进行相互之间的传递就是一个重要的问题，这也是密钥分发研究的问题。

### 10.1.1 有中心的密码分发

我们假定有个一个密钥分配中心 KDC(key distribution center) 与所有的通信方之间都有一个主密钥(可以说建立了一个可信信道)，比如 A 和 KDC 之间的主密钥为  $K_A$ ，B 和 KDC 之间的主密钥为  $K_B$ ，C 和 KDC 之间的主密钥为  $K_C$  等等。A 如果想和 B 建立一个会话密钥(session key)  $K_S$  的过程为：

1. A 向 KDC 发送请求和 B 建立会话密钥的请求消息 R 和一个随机数  $N_1$ 。
2. KDC 将会话密钥  $K_S$ ，应答消息， $N_1$ ， $E_{K_B}(K_S, ID_A)$  一起用  $K_A$  加密，发送给 A，其中  $ID_A$  是 A 的身份信息。
3. A 用  $K_A$  解密后，比较  $N_1$  确定是本次请求，然后将  $E_{K_B}(K_S, ID_A)$  发送给 B。
4. B 收到后用  $K_B$  解密，然后生成一个新随机数  $N_2$ ，将  $E_{K_S}(N_2)$  发送给 A。
5. A 用  $K_S$  解密后，将  $E_{K_S}(f(N_2))$  发送给 B，A 和 B 的会话密钥 A 方已经确认。
6. B 用  $K_S$  解密后，验证  $f(N_2)$ ，正确，则 A 和 B 的会话密钥 B 方已经确认。

把以上的过程用一个符号体系来表述：

1.  $A \rightarrow KDC :: R, N_1$
2.  $KDC \rightarrow A :: E_{K_A}(K_S, N_1, E_{K_B}(K_S, ID_A))$
3.  $A :: D_{K_A}(E_{K_A}(K_S, N_1, E_{K_B}(K_S, ID_A))), J(N_1)$ <sup>4</sup>
- $A \rightarrow B :: E_{K_B}(K_S, ID_A)$
4.  $B \rightarrow A :: E_{K_S}(N_2)$
5.  $A \rightarrow B :: E_{K_S}(f(N_2))$
6.  $B :: J(f(N_2))$

### 10.1.2 无中心的密钥分发

无中心的密钥分发要求双方已经共享了一个主密钥 (main key) MK，再此基础上，A、B 双方协商一个新的通信密钥 (session key) SK，协商过程如下：

1. A 向 B 发送请求密钥消息 R 和一个随机数  $N_1$ 。
2. B 选取会话密钥 SK，将其和 B 的身份、 $f(N_1)$ 、新随机数  $N_2$  用  $MK$  加密后一起发给 A。
3. A 收到后，用  $MK$  解密，验证  $f(N_1)$ ，证明是此次的协商。A 将 SK 和  $f(N_2)$  用  $MK$  加密后发给 B。A 与 B 的新会话开始使用 SK。
4. B 收到后，用  $MK$  解密，验证  $f(N_2)$ ，B 与 A 的新会话开始使用 SK，

---

<sup>4</sup> $J(N_1)$  表示判断  $N_1$  是否与其生成的一致，防止重放攻击



把以上的过程用一个符号体系来表述：

1.  $A \rightarrow B :: R, N_1.$
2.  $B \rightarrow A :: E_{MK}(SK, ID_B, f(N_1), N_2)$
3.  $A :: V(f(N_1))$ <sup>5</sup>
- $A \rightarrow B :: E_{MK}(SK, f(N_2))$
4.  $B :: V(f(N_2))$

A、B 协商成功新的会话密钥 SK。

---

<sup>5</sup> $V(f(N_1))$  表示验证 (validate) $f(N_1)$

## 10.2 Diffie-Hellman 密钥交换

大素数  $p$  和  $p$  的原根  $a$ <sup>6</sup>公开，在这一基础上通信双方就可以协商会话密钥，过程如下：

1. A 选择一个保密随机数  $X_A$ , 将  $\alpha = a^{X_A} \text{ mod } p$  发给 B。
2. B 选择一个保密随机数  $X_B$ , 将  $\beta = a^{X_B} \text{ mod } p$  发给 A。
3. A,B 可分别计算密钥, A 的计算过程为,  $\beta^{X_A} = a^{X_B X_A} \text{ (mod } p)$ , B 的计算过程为,  $\alpha^{X_B} = a^{X_A X_B} \text{ (mod } p)$ 。

### 10.2.1 DH 中间人攻击

假设在 A 和 B 协商密钥的时候, 有个攻击者 Mallory(记为 M), 可以形成中间人攻击 (Man-in-the-middle attack, 缩写 MITM), 其过程如下：

1. A ::  $X_A$   
 $A \rightarrow M :: \alpha = a^{X_A} \text{ (mod } p)$
2. M ::  $X_{MA}, X_{MB}$   
 $M \rightarrow B :: \beta' = a^{X_{MB}} \text{ (mod } p)$   
 $M \rightarrow A :: \alpha' = a^{X_{MA}} \text{ (mod } p)$
3. B ::  $X_B$   
 $B \rightarrow M :: \beta = a^{X_B} \text{ (mod } p)$
4. A ::  $\alpha'^{X_A} = a^{X_A X_{MA}} \text{ (mod } p))$   
 $B :: \beta'^{X_B} = a^{X_B X_{MB}} \text{ (mod } p)$   
 $M :: \alpha'^{X_A} = a^{X_{MA} X_A} \text{ (mod } p), \beta'^{X_B} = a^{X_{MB} X_B} \text{ (mod } p)$

从以上密钥交换的过程可以看出, 最终 A 和 M 之间形成一个密钥, M 和 B 之间形成一个密钥, 而 A 以为和 B 在通信, B 以为和 A 在通信。

Diffie-Hellman 协议之所以不能抵抗中间人攻击, 是因为在通信过程中并没有对参与方进行认证, 可以在密钥交换过程中加入认证技术来抵抗中间人攻击。

---

<sup>6</sup> $a^{\phi p} \text{ (mod } p) = 1$ ,  $p$  为素数, 所以有  $a^{p-1} \text{ (mod } p) = 1$



## 10.3 密钥分割

在有些应用场合，为了保证密钥的安全，需要密钥（或者秘密）由多人持有，任何一个人持有的信息不能得到完整的密钥，只有全部或达到规定的人数时，将这些人持有的信息合并，才可以获得（或者计算）完整的密钥。

### 10.3.1 Shamir 门限方案

Shamir 门限方案是基于多项式的拉格朗日（Lagrange）插值公式的。

 **注意** Shamir 这个名字是不是很熟悉？他就是 RSA 中的 S, Shamir. 图灵奖获得者 Shamir, Shamir 门限方案也是其获得图灵奖时提到的主要贡献<sup>7</sup>。

#### 10.3.1.1 拉格朗日插值法 (Lagrange Interpolation Polynomial )

在数值分析中，拉格朗日插值法是以法国十八世纪数学家约瑟夫·拉格朗日命名的一种多项式插值方法。许多实际问题中都用函数来表示某种内在联系或规律，而不少函数都只能通过实验和观测来了解。如对实践中的某个物理量进行观测，在若干个不同的地方得到相应的观测值，拉格朗日插值法可以找到一个多项式，其恰好在各个观测的点取到观测到的值。这样的多项式称为拉格朗日（插值）多项式。数学上来说，拉格朗日插值法可以给出一个恰好穿过二维平面上若干个已知点的多项式函数。拉格朗日插值法最早被英国数学家爱德华·华林于 1779 年发现<sup>8</sup>，不久后（1783 年）由莱昂哈德·欧拉再次发现。1795 年，拉格朗日在其著作《师范学校数学基础教程》中发表了这个插值方法，从此他的名字就和这个方法联系在一起<sup>9</sup>。<sup>10</sup>

拉格朗日插值多项式<sup>11</sup>:

$$P(x) = \sum_{j=1}^n P_j(x)$$

此多项式代表的曲线通过  $n$  个点  $(x_1, y_1 = f(x_1)), (x_2, y_2 = f(x_2)), \dots, (x_n, y_n = f(x_n))$ ，此处：

$$P_j(x) = y_j \prod_{k=1; k \neq j}^n \frac{x - x_k}{x_j - x_k}$$

把拉格朗日插值多项式展开写：

$$P(x) = y_1 \frac{(x - x_2)(x - x_3) \dots (x - x_n)}{(x_1 - x_2)(x_1 - x_3) \dots (x_1 - x_n)} + y_2 \frac{(x - x_1)(x - x_3) \dots (x - x_n)}{(x_2 - x_1)(x_2 - x_3) \dots (x_2 - x_n)} +$$

<sup>7</sup>关于 Shamir 的简单介绍，可以看 ACM 关于图灵奖获得者的介绍页面 [https://amturing.acm.org/award\\_recipients/shamir\\_2327856.cfm](https://amturing.acm.org/award_recipients/shamir_2327856.cfm)

<sup>8</sup>E. Waring. Problems Concerning Interpolations. Philosophical Transactions of the Royal Society of London. 1779, 69: 59–67.

<sup>9</sup>E. Meijering. A chronology of interpolation: From ancient astronomy to modern signal and image processing,. Proceedings of the IEEE: 323.

<sup>10</sup>此段文字引自 <https://www.cnblogs.com/ECJTUACM-873284962/p/6833391.html>，引用这段文字的原因是，这段文字里面有参考文献的引用，本人没有去考证。

<sup>11</sup><https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html>

$$\dots + y_n \frac{(x - x_1)(x - x_2) \dots (x - x_{n-1})}{(x_n - x_2)(x_n - x_3) \dots (x_n - x_{n-1})}$$

利用  $f(x)$  的  $n$  个采样点构造拉格朗日多项式  $P(x)$ , 以此来拟合  $f(x)$ 。

### 10.3.1.2 门限方案

假设有  $n$  个参与者, 选择一个大素数  $q$ ,  $q \geq n+1$ , 在  $GF(q) \setminus 0$  上取随机数  $a_0$ , 在有限域  $GF(q)$  上构造一个多项式:

$$f(x) = a_0 + a_1 x^1 + a_2 x^2 + \dots + a_{k-1} x^{k-1} \pmod{q}$$

其中  $k-1$  个系数  $a_1, a_2, \dots, a_{k-1}$  也是在  $GF(q) \setminus 0$  上选取。

把  $a_0$  做为密钥  $S$ ,  $n$  个参与者分到的子密钥分别为  $f(1), f(2), \dots, f(n)$ , 任意  $k$  个参与者合作都可以获得密钥  $S$ , 方法是利用他们的  $k$  个子密钥  $(k_i, f(k_i)), i = 1, 2, \dots, k$ , 其中,  $k_i \in \{1, 2, \dots, n\}$ , 构造拉格朗日插值多项式:

$$P(x) = \sum_{j=1}^k f(k_j) \prod_{i=1; i \neq j}^k \frac{x - x_i}{x_j - x_i} \pmod{q}$$

通过多项式  $P(x)$  可以直接计算获得密钥  $S = P(0)$ . 而当想共同获得密钥的参与者少于  $k$  时, 其无法获得密钥。

### 10.3.2 CRT 门限方案

CRT(chinese remainder theorem) 门限方案, 顾名思义, 是利用中国剩余定理构造的门限方案。

#### 10.3.2.1 中国剩余定理 [3]

设  $m_1, m_2, \dots, m_k$  是  $k$  个两两互素的正整数, 若令  $m = m_1 m_2 \dots m_k$ ,  $M_i = m_1 m_2 \dots m_{i-1} m_{i+1} \dots m_k$ ,  $m = m_i M_i$ , 则对于任意的整数  $b_1, b_2, \dots, b_k$ , 同余方程组

$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_2 \pmod{m_2} \\ \dots \\ x \equiv b_k \pmod{m_k} \end{cases}$$

有唯一解  $x \equiv M'_1 M_1 b_1 + M'_2 M_2 b_2 + \dots + M'_k M_k b_k \pmod{m}$ , 其中  $M'_i$  为  $M_i$  的逆元,  $M'_i M_i \equiv 1 \pmod{m_i}, i = 1, 2, \dots, k$ 。

#### 10.3.2.2 门限方案

Asmuth 和 Bloom 在其 1980 发表的文章 "A Modular Approach to Key Safeguarding" 一文中提出了一个基于中国剩余定理的密码共享方案。[9]

假设共有  $n$  个参与者，取  $n$  个大于 1 的整数  $m_1, m_2, \dots, m_n$ ，满足  $m_1 \leq m_2 \leq \dots \leq m_n$ ,  $\gcd(m_i, m_j) = 1, \forall i, j, i \neq j$  (即两两互素),  $m_1 m_2 \dots m_k > p m_n m_{n-1} m_{n-k+2}$ , 其中  $p$  和  $m_i (i = 1, \dots, n)$  互素, 计算  $M = m_1 m_2 \dots m_n$ , 随机取一数  $S$  做为密钥,  $m_1 m_2 \dots m_k > S > m_n m_{n-1} m_{n-k+2}$ , 将  $(s_i, m_i, M) (i = 1, \dots, n)$  做为子密钥分别发给  $n$  个参与者, 其中  $s_i = S \pmod{m_i}$ , 至此我们构建了一个基于 CRT 的  $(k, n)$  门限方案。

每个参与者  $i$  都可以计算:

$$\begin{cases} M_i = \frac{M}{m_i} \pmod{m_1} \\ M'_i = M_i^{-1} \pmod{m_i} \end{cases}$$

现在有  $k$  个参与者想要恢复密钥  $S$ , 他们的私钥记为  $(s_{c_i}, m_{c_i}, M) (i = 1, \dots, k; c_i \in \{1, 2, \dots, n\})$ , 我们有:

$$\begin{cases} S \equiv s_{c_1} \pmod{m_{c_1}} \\ \dots \\ S \equiv s_{c_k} \pmod{m_{c_k}} \end{cases}$$

根据中国剩余定理我们可得:

$$S = \sum_{j=1}^k s_{c_j} M_{c_j} M'_{c_j} s_{c_j} \pmod{\prod_{i=1}^k m_{c_i}}$$

从而这  $k$  个参与这把子密钥信息联合可计算密钥  $S$ 。

而当少与  $k$  个人合作时, 比如  $k-1$  个人, 我么可以得到方程组:

$$\begin{cases} x \equiv s_{c_1} \pmod{m_{c_1}} \\ \dots \\ x \equiv s_{c_{k-1}} \pmod{m_{c_{k-1}}} \end{cases}$$

利用 CRT 定理, 我么可得方程解为:

$$S' = x = \sum_{j=1}^{k-1} s_{c_j} M_{c_j} M'_{c_j} s_{c_j} \pmod{\prod_{i=1}^{k-1} m_{c_i}}$$

我们有:

$$m_1 m_2 \dots m_k > S > p m_n m_{n-1} m_{n-k+2} > \prod_{i=1}^{k-1} m_{c_i} > S'$$

显然  $S \neq S'$ .

# 第 11 章 安全服务和安全机制 (Security Services & Mechanisms)

---

## 11.1 密码学支撑的安全服务 (Security Services)

在 NIST 标准 “SP 800-57 Part 1 Rev. 5 Recommendation for Key Management: Part 1 - General” 中对密码学支撑的安全服务进行了总结，密码技术可以支撑 7 种安全服务：

1. 机密性 Confidentiality: Confidentiality is the property whereby information is not disclosed to unauthorized parties.
2. 数据完整性 Data Integrity: Data integrity is a property whereby data has not been modified in an unauthorized manner since it was created, transmitted or stored. Modification includes the insertion, deletion and substitution of data.
3. 认证 Authentication: Three types of authentication services can be provided using cryptography: identity authentication, integrity authentication, and source authentication.
  - An identity authentication service is used to provide assurance of the identity of an entity interacting with a system.
  - An integrity authentication service is used to verify that data has not been modified (i.e., this service provides integrity protection).
  - A source authentication service is used to verify the identity of the user or system that created information (e.g., a transaction or message).
4. 授权 Authorization: Authorization is concerned with providing an official sanction or permission to perform a security function or activity (e.g., accessing a room).
5. 非否认性 Non-reputation: In key management, non-repudiation is a term associated with digital signature keys and digital certificates that bind the name of the certificate subject to a public key.
6. 支持服务 Support Services: The basic cryptographic security services discussed above often require other supporting services. For example, cryptographic services often require the use of key establishment and random number generation services. Key establishment is the process by which cryptographic keys are securely established among entities using manual transport methods (e.g., key loaders), automated methods (e.g., key-transport and/or key-agreement protocols), or a combination of automated and manual methods. Random numbers are needed during the generation of cryptographic keys, challenge values and nonces (see SP 800-175B).
7. 组合服务 Combining Services: In many applications, a combination of security services (e.g., confidentiality, integrity authentication, source authentication, and support for non-

repudiation) is desired. Designers of secure systems often begin by considering which security services are needed to protect the information contained within and processed by the system. After these services have been determined, the designer then considers what mechanisms will best provide these services. Not all mechanisms are cryptographic in nature. For example, physical security may be used to protect the confidentiality of certain types of data, and identification badges or biometric identification devices may be used for identity authentication. However, cryptographic mechanisms consisting of algorithms, keys, and other keying material often provide the most cost-effective means of protecting the security of information. This is particularly true in applications where the information would otherwise be exposed to unauthorized entities.

## 11.2 密码算法 (Cryptographic Algorithm)

NIST SP.800-57 V.5 将密码算法分为三类:

1. 密码哈希函数 (Cryptographic Hash Functions): Cryptographic hash functions do not require keys for their basic operation. A cryptographic hash function (also called a hash algorithm) is a cryptographic primitive that produces a condensed representation of its input (e.g., a message or other data). Common names for the output of a hash function include hash value, hash, message digest, and digital fingerprint. The maximum number of input and output bits is determined by the design of the hash function.
2. 对称密钥算法 (Symmetric-Key Algorithms): Symmetric-key algorithms (sometimes known as secret-key algorithms) transform data in a way that is fundamentally difficult to undo without knowledge of a secret key. The key is “symmetric” because the same key is used for a cryptographic operation and its inverse (e.g., for both encryption and decryption).
3. 非对称密钥算法 (Asymmetric-Key Algorithms): Asymmetric-key algorithms, commonly known as public-key algorithms, use two related keys (i.e., a key pair) to perform their functions: a public key and a private key. The public key may be known by anyone; the private key should be under the sole control of the entity that “owns” the key pair.<sup>1</sup> Even though the public and private keys of a key pair are related, knowledge of the public key cannot be used to determine the private key.

## 11.3 OSI 7 层模型与安全服务与机制

ISO(internet standard organization) 国际标准在 1989 发布了一个标准 “Information processing systems ——Open Systems Interconnection ——Basic Reference Model ——Part 2 : Security Architecture”，在此标准中对安全服务、安全机制和安全服务和机制在 OSI 7 层模型中的部署情况。此标准对应我们的国标 “GB/T 9387.2-1995 信息处理系统——开放

---

<sup>1</sup>Sometimes a key pair is generated by a party that is trusted by the key owner rather than by the key owner.

系统互连——基本参考模型——第 2 部分：安全体系结构”，下面的中文翻译来自 GB/T 9387.

### 11.3.1 安全服务 (Security Services)

9387.2 标准总结了五种安全服务。

1. 鉴别对等实体鉴别数据原发鉴别
2. 访问控制
3. 数据机密性
  - (a). 连接机密性
  - (b). 无连接机密性
  - (c). 选择字段机密性
  - (d). 通信业务流机密性
4. 数据完整性
  - (a). 带恢复的连接完整性
  - (b). 不带恢复的连接完整性
  - (c). 选择字段的连接完整性
  - (d). 无连接完整性
  - (e). 选择字段无连接完整性
5. 抗抵赖
  - (a). 有数据原发证明的抗抵赖
  - (b). 有交付证明的抗抵赖

### 11.3.2 特定的安全机制 (Specific security mechanisms)

这些安全机制可以设置在适当的系统层上，提供安全服务。

1. 加密
2. 数字签名机制
3. 访问控制机制
4. 数据完整性机制
5. 鉴别交换机制
6. 通信业务填充机制：通信业务填充机制能用来提供各种不同级别的保护，抵抗通信业务分析。这种机制只有在通信业务填充受到机密服务保护时才是有效的。
7. 路由选择控制机制
8. 公证机制 (Notarization)：有关在两个或多个实体之间通信的数据的性质，如它的完整性、原发、时间和目的地等能够借助公证机制而得到确保。这种保证是由第三方公证人提供的。公证人为通信实体所信任，并掌握必要信息以一种可证实方式提供所需的保证。每个通信事例可使用数字签名、加密和完整性机制以适应公证人提供的那种服务。当这种公证机制被用到时，数据便在参与通信的实体之间经由受保护的通信实例和公证方进行通信。

### 11.3.3 普遍安全机制 (Pervasive security mechanisms)

普遍性安全机制不是为任何特定的服务而设定的，普遍安全机制可以认为属于安全管理方面。

1. 可信功能度 (trusted functionality)
2. 安全标记
3. 事件检测
4. 安全审计跟踪
5. 安全恢复

### 11.3.4 安全服务与安全机制间关系的实例 (Illustration of relationship of security services and mechanisms)

安全服务和安全机制之间的对应关系如11.1所示。

表 11.1: 安全服务和安全机制的关系

机制 服务 Mechanism Services	加密 Encipherment	数字签名 Digital Signature	访问控制 Access Control	数据完整性 Data Integrity	鉴别交换 Authentication Exchange	通信业务填充 Traffic Padding	路由控制 Routing Control	公证 Notarization
对等实体鉴别	Y	Y	•	•	Y	•	•	•
数据原发鉴别	Y	Y	•	•	•	•	•	•
访问控制服务	•	•	Y	•	•	•	•	•
连接机密性	•	Y	Y	Y	Y	Y	•	Y
无连接机密性	•	Y	Y	Y	Y	Y	•	Y
选择字段机密性	•	Y	Y	Y	Y	Y	Y	Y
通信业务流机密性	•	Y	Y	Y	Y	•	•	Y
带恢复的连接完整性	•	Y	Y	•	Y	Y	Y	Y
不带恢复的连接完整性	•	Y	Y	•	Y	Y	Y	Y
选择字段连接完整性	•	Y	Y	•	Y	Y	Y	Y
无连接完整性	•	•	Y	•	Y	Y	Y	Y
选择字段无连接完整性	•	•	Y	•	Y	Y	Y	Y
抗抵赖, 带数据原发证据	•	Y	•	Y	•	•	•	Y
抗抵赖, 带交付证据	•	Y	•	Y	•	•	•	Y

a. Y 表示这种机制被认为是适宜的, 或单独使用, 或与别的机制联合使用。

b. • 表示这种机制被认为是不适宜的。

### 11.3.5 安全服务与层的关系的实例 (Illustration of the relationship of security services and layers)

下面是 OSI 参考模型的 7 层结构每层可提供的安全服务列表, 如图11.2所示。

表 11.2: 安全服务和层的关系

层 服务 \ 层	1 物理层	2 数据链路层	3 网络层	4 传输层	5 会话层	6 表示层	7 应用层 *
对等实体鉴别	•	•	Y	Y	•	•	Y
数据原发鉴别	•	•	Y	Y	•	•	Y
访问控制服务	•	•	Y	Y	•	•	Y
连接机密性	Y	Y	Y	Y	•	•	Y
无连接机密性	•	Y	Y	Y	•	•	Y
选择字段机密性	•	•	•	•	•	•	Y
通信业务流机密性	Y	•	Y	•	•	•	Y
带恢复的连接完整性	•	•	•	Y	•	•	Y
不带恢复的连接完整性	•	•	Y	Y	•	•	Y
选择字段连接完整性	•	•	•	Y	•	•	Y
无连接完整性	•	•	Y	Y	•	•	Y
选择字段无连接完整性	•	•	•	•	•	•	Y
抗抵赖, 带数据原发证据	•	•	•	•	•	•	Y
抗抵赖, 带交付证据	•	•	•	•	•	•	Y

a. Y 表示这种机制被认为是适宜的, 或单独使用, 或与别的机制联合使用。

b. • 表示这种机制被认为是不适宜的。

c.\* 表示, 就第 7 层而言, 应用层进程本身可以提供安全服务。

# 第 12 章 协议 (Protocols)

---

协，众之同和也，议，语也，后引申为，商议，讨论，协议就是通过商议达到一致。英文对应的单词为 Protocol。

在密钥学中我们可以看到很多类型的协议，这些协议都是要完成一定的安全目标，并且是有两个或以上不同的参与方，并且这些参与方相互交互，最终达成安全目标。

## 12.1 零知识协议 (Zero-knowledge Protocol)

零知识协议 (zero-knowledge protocol) 是基于这样一个思考，就是是否能够在不给通信方提供任何有用信息的前提下，使得通信方确信知道对方拥一个给定的信息。下面的英文信息是维基<sup>1</sup>上有关 zero-knowledge protocol 的解释。

In cryptography, a zero-knowledge proof or zero-knowledge protocol is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true.

If proving the statement requires knowledge of some secret information on the part of the prover, the definition implies that the verifier will not be able to prove the statement in turn to anyone else, since the verifier does not possess the secret information. Notice that the statement being proved must include the assertion that the prover has such knowledge (otherwise, the statement would not be proved in zero-knowledge, since at the end of the protocol the verifier would gain the additional information that the prover has knowledge of the required secret information). If the statement consists only of the fact that the prover possesses the secret information, it is a special case known as zero-knowledge proof of knowledge, and it nicely illustrates the essence of the notion of zero-knowledge proofs: proving that one has knowledge of certain information is trivial if one is allowed to simply reveal that information; the challenge is proving that one has such knowledge without revealing the secret information or anything else.

For zero-knowledge proofs of knowledge, the protocol must necessarily require interactive input from the verifier, usually in the form of a challenge or challenges such that the responses from the prover will convince the verifier if and only if the statement is true (i.e., if the prover does have the claimed knowledge). This is clearly the case, since otherwise the verifier could record the execution of the protocol and replay it to someone else: if this were accepted by the new party as proof that the replaying party knows the secret information, then the new party's acceptance is either justified –the replayer does know the secret information –which means that the protocol leaks knowledge and is not zero-knowledge, or it is spurious –i.e. leads to a party accepting someone's proof of knowledge who does not actually possess it.

---

<sup>1</sup>信息链接为<https://encyclopedia.thefreedictionary.com/zero-knowledge+proof>

Some forms of non-interactive zero-knowledge proofs of knowledge exist,[1][2] but the validity of the proof relies on computational assumptions (typically the assumptions of an ideal cryptographic hash function).

### 12.1.1 零知识协议的形象例子 [2]

1989 年 J. Quisquater 和 L. Guillou 在文章 “How to Explain Zero-Knowledge Protocols to Your Children” 中列举了一个形象的基本零知识协议的例子<sup>2</sup>，如图12.1 所示，此图刻画了一个简单的迷宫，只有知道秘密口令的人才能打卡 C 和 D 之间的门。现在，P 希望向 V 证明 P 能够打开这个门，但是不愿意向 V 泄露 P 掌握的秘密口令。为此，P 采用了所谓“分割与选择”(cut and choice) 技术实现一个零知识协议。采用这种技术的零知识协议如下：

1. V(Verifier) 在初始状态下停留在位置 A;
2. P(Prover) 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，并命令 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 和 D 之间的门;
5. P 和 V 重复以上 (1)-(4)n 次。

在上述协议中，如果 P 不知道秘密口令，他只能从来路返回，而不能走另外一条路。此外，P 每一次猜对 V 要求他走哪一条路的概率是 50%，因此，每一轮协议 P 能欺骗 V 的概率是 50%，假定 n=16，则执行 n 轮协议后，P 成功欺骗 V 的概率是  $\frac{1}{2^n} = \frac{1}{65536}$ ，于是，如果 P 能够 16 次按 V 的要求路线返回，V 即能证明 P 确实知道秘密口令。我们还看出，V 无法从上述证明中获取丝毫有关 P 的秘密口令的信息，所以，这是一个零知识协议。

## 12.2 认证协议 (Authentication protocol)

认证协议就是完成认证目标的安全协议，也就是 A 通过这个过程让对方 B "确认" 她是 A，或者，"相信" 她是 A。

### 12.2.1 Needham-Schroeder

Needham 和 Schroeder 在 1978 年的 Communications of the ACM 上发表了文章 "Using Encryption for Authentication in Large Networks of Computers"，提出了著名的 Needham-Schroeder 认证协议，在此论文中他们提出了两种协议，一个就是后来称为基于对称密钥体制的 Needham-Schroeder 协议，另一个称为基于公钥密码体制的 Needham-Schroeder 协议。<sup>3</sup>

<sup>2</sup>Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Thomas A. Berson. 1989. How to Explain Zero-Knowledge Protocols to Your Children. In Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '89). Springer-Verlag, Berlin, Heidelberg, 628–631.

<sup>3</sup>R..M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM, 21(12):993-999, December 1978. 13

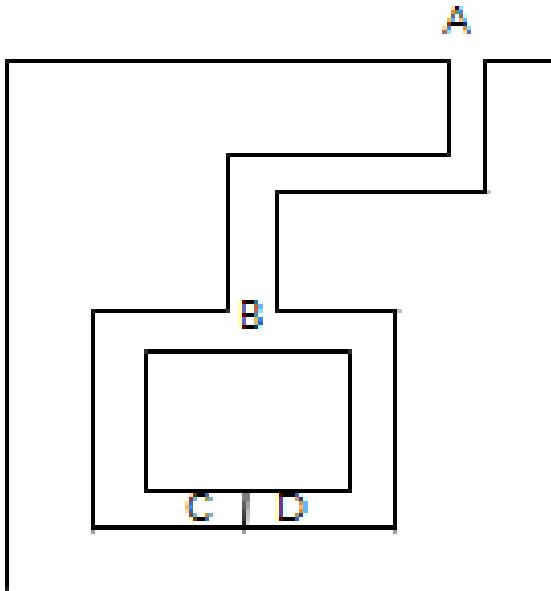


图 12.1: 一种零知识协议的形象说明

### 12.2.1.1 基于对称密钥体制的 Needham-Schroeder(Needham-Schroeder Symmetric Key Protocol)

大家在学习 Needham-Schroeder 协议时，要特别注意协议的这种描述方式，要能读懂，同时也会利用这种方式描述协议。

我们先对符号进行说明。Here, Alice (A) initiates the communication to Bob (B). S is a server trusted by both parties. In the communication:

- A and B are identities of Alice and Bob respectively
- $K_{AS}$  is a symmetric key known only to A and S
- $K_{BS}$  is a symmetric key known only to B and S
- $N_A$  and  $N_B$  are nonces generated by A and B respectively
- $K_{AB}$  is a symmetric, generated key, which will be the session key of the session between A and B

此处特别要理解 nonces 的含义，在 Needham-Schroeder 的原始论文中对 Nonce 解释是：Nonce means "used only once."

基于对称密码体制的 NH 协议描述如下：

- $A \rightarrow S :: A, B, N_A$      Alice sends a message to the server identifying herself and Bob, telling the server she wants to communicate with Bob.
- $S \rightarrow A :: \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$      The server generates and sends back to Alice a copy encrypted under for Alice to forward to Bob and also a copy for Alice. Since Alice may be requesting keys for several different people, the nonce assures Alice that the message is fresh and that the server is replying to that particular message and the inclusion of Bob's name tells Alice who she is to share this key with.
- $A \rightarrow B :: \{K_{AB}, A\}_{K_{BS}}$      Alice forwards the key to Bob who can decrypt it with the

key he shares with the server, thus authenticating the data.

- $B \rightarrow A :: \{N_B\}_{K_{AB}}$  Bob sends Alice a nonce encrypted under to show that he has the key.
- $A \rightarrow B :: \{N_B - 1\}_{K_{AB}}$  Alice performs a simple operation on the nonce, re-encrypts it and sends it back verifying that she is still alive and that she holds the key.

其中,  $\{M\}_K$  表示消息 M 用密钥 K 加密。我们在用这样的符号描述时通常不需要进行文字解释, 所以正式的描述应该是:

- $A \rightarrow S :: A, B, N_A$
- $S \rightarrow A :: \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
- $A \rightarrow B :: \{K_{AB}, A\}_{K_{BS}}$
- $B \rightarrow A :: \{N_B\}_{K_{AB}}$
- $A \rightarrow B :: \{N_B - 1\}_{K_{AB}}$

### 12.2.1.2 基于公钥密码体制的 Needham-Schroeder(Needham-Schroeder Public-Key Protocol)

Here, Alice (A) and Bob (B) use a trusted server (S) to distribute public keys on request. These keys are:

- $K_{PA}$  and  $K_{SA}$ , respectively public and private halves of an encryption key-pair belonging to A (S stands for "secret key" here)
- $K_{PB}$  and  $K_{SB}$ , similar belonging to B
- $K_{PS}$  and  $K_{SS}$ , similar belonging to S. (Note this has the property that  $K_{SS}$  is used to encrypt and  $K_{PS}$  to decrypt).

The protocol runs as follows:

- $A \rightarrow S :: A, B$
- $S \rightarrow A :: \{K_{PB}, B\}_{K_{SS}}$
- $A \rightarrow B :: \{N_A, A\}_{K_{PB}}$
- $B \rightarrow S :: B, A$
- $S \rightarrow B :: \{K_{PA}, A\}_{K_{SS}}$
- $B \rightarrow A :: \{N_A, N_B\}_{K_{PA}}$
- $A \rightarrow B :: \{N_B\}_{K_{PB}}$

At the end of the protocol, A and B know each other's identities, and know both NA and NB. These nonces are not known to eavesdroppers.

### 12.2.1.3 安全性分析

对于基于私钥密码体制的 Needham-Schroeder 其安全性描述为, “The protocol is vulnerable to a replay attack (as identified by Denning and Sacco<sup>4</sup>). If an attacker uses an older,

---

<sup>4</sup>Denning, Dorothy E.; Sacco, Giovanni Maria (1981). "Timestamps in key distributed protocols". Communication of the ACM. 24 (8): 533–535. doi:10.1145/358722.358740.

compromised value for  $K_{AB}$ , he can then replay the message to Bob, who will accept it, being unable to tell that the key is not fresh.<sup>5</sup>

下面我们看看 Denning 1981 年的文章中对针对 Needham-Schroeder 重放攻击的描述。

假设有个第三方 C 可以阻拦和记录 A、B 间所有的通信，同时 C 获得了一次通信密钥  $K_{AB}$ ,C 可以在以后的通信中欺骗 B 使用原来的密钥  $K_{AB}$ ，过程如下：

- $C \rightarrow B :: \{K_{AB}, A\}_{K_{BS}}$ , C 重放以前的消息
- $B \rightarrow A :: \{N_B\}_{K_{AB}}$  , C 截取此消息，解密此消息，伪造如下 A 的响应
- $C \rightarrow B :: \{N_B - 1\}_{K_{AB}}$

至此，C 截取和解密 B 的回应，发送伪造的消息 (bogus messages) 给 B，这些消息好像是从 A 发出，依次建立了 B 和 C 之间一条保密通信链路，但是 B 以为是和 A 建立的。这条伪造的“A-B”的保密通信链路，不会影响 A 继续和 B 建立保密通信链路，也不会影响 B 与其他人建立保密通信链路。

Denning 在其文章中给出了使用时间戳 (Timestamps) 的解决方案，Neuman 在其 1993 年发表的论“*A Note on the use of timestamps as nonces*”中给出了利用 Nonce 解决的方案。

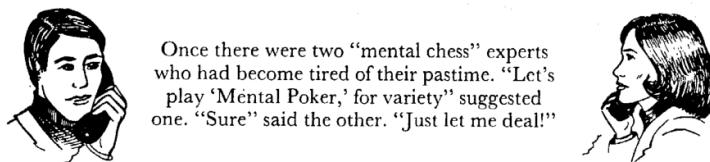
## 12.3 智力扑克

智力扑克是 RSA 的三个发明者 Shamir,Rivest,Adleman 在 1981 年麻省理工工作时，发表的一篇文章“*Mental Poker*<sup>6</sup>”提出的，下面我们看看文章中的原始表述 (12.2)。

### ABSTRACT

Can two potentially dishonest players play a fair game of poker without using any cards—for example, over the phone? This paper provides the following answers:

- 1 No. (Rigorous mathematical proof supplied.)
- 2 Yes. (Correct and complete protocol given.)



Once there were two “mental chess” experts who had become tired of their pastime. “Let’s play ‘Mental Poker,’ for variety” suggested one. “Sure” said the other. “Just let me deal!”

Our anecdote suggests the following question (proposed by Robert W. Floyd): “Is it possible to play a fair game of ‘Mental Poker?’” We will give a complete (but paradoxical) answer to this question. First we will prove that the problem is intrinsically insoluble, and then describe a fair method of playing “Mental Poker”.

图 12.2: mental poker paper -abstract

<sup>5</sup><https://encyclopedia.thefreedictionary.com/Needham-Schroeder+protocol>

<sup>6</sup><https://people.csail.mit.edu/rivest/ShamirRivestAdleman-MentalPoker.pdf>

### 12.3.1 SRA 方案 [2]

SRA 方案用交换秘密 (commutative cipher) 实现治理扑克的公正博弈。令  $E_A$  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，在比赛结束之后才公布这些变换，用以证明比赛的公正性。加密变换必须满足交换律 (这就是交换秘密名称的由来)，即对任何信息 M 下述公式

$$E_A(E_B(M)) = E_B(E_A(M))$$

均成立。52 张扑克牌可以用 52 条信息表示如下：

M1:	梅花2
M2:	梅花3
...	
M12:	梅花K
M13:	梅花A
...	
M51:	黑桃K
M52:	黑桃A

假设由 B 发牌，则“公正发牌协议”是：

(1)B 用自己的秘密加秘密变换  $E_B$  对 52 个信息进行加密，得到

$$E_B(M_i), i = 1, \dots, 52$$

然后，他洗这副加密的牌，即随机地改变  $E_B(M_i)$  的顺序，洗完牌后，B 将它们全部发送给 A。

(2)A 从中随机地选取 5 张牌，即 5 个信息，并把它们发送给 B，B 用自己的秘密的解密变换  $D_B$  将这 5 个信息解密，从而知道自己手中有哪 5 张牌。A 无法知道 B 手中的牌，因为 A 不知道 B 的秘密的加密和解密变换。

(3)A 再次随机地挑选 5 个加密后的信息  $C_1, \dots, C_5$ ，并且自己的秘密的加密变换  $E_A$  将它们加密，得到

$$C'_i = E_A(C_i), i = 1, \dots, 5$$

然后，A 将这 5 个双重加密的信息  $C'_1, \dots, C'_5$  发送给 B。(4)B 用自己的解密变换  $D_B$  对每个  $C'_i$  解密，得到

$$D_B(C'_i) = D_B(E_A(C_i)) = D_B(E_A(E_B(M_j))) = D_B(E_B(E_A(M_j))) = E_A(M_j)$$

其中  $1 \leq j \leq 52$ ，然后 B 将它们发回给 A，A 可以利用他自己的秘密的解密变换  $D_A$  算出各  $M_j$ ，从而知道自己手中有哪 5 张牌。但是 B 对 A 手中的牌毫无所知，因为  $D_A$  是 A 的秘密的解密变换。

如果在牌局进行中，A 或 B 要从剩下的牌中取牌，可以重复执行上述公正发牌协议，在牌局结束后，A 和 B 都公开他们的加密和解密变换，因而可以校验对方是否如他们声称的那样进行了公正的博弈。



交换密码是否存在呢？因为模数的指数运算满足交换律，所以可以如下构造 A、B 的加解密变换。

A、B 事先约定一个大模数 n 和与它相应的欧拉函数  $\phi(n)$ 。

A 选一对密钥  $(e_A, d_A)$ , 其中  $gcd(e_A, \phi(n)) = 1$ , 且  $e_A d_A = 1 \pmod{\phi(n)}$ , 加解密为:

$$E_A(M) = M^{e_A} \pmod{n}$$

$$D_A(M) = M^{d_A} \pmod{n}$$

类似 B 选一对密钥  $(e_B, d_B)$ , 其中  $gcd(e_B, \phi(n)) = 1$ , 且  $e_B d_B = 1 \pmod{\phi(n)}$ , 加解密为:

$$E_B(M) = M^{e_B} \pmod{n}$$

$$D_B(M) = M^{d_B} \pmod{n}$$

## 12.4 kerberos 协议 & 系统

Kerberos 是一种计算机网络授权协议，用来在非安全网络中，对个人通信以安全的手段进行身份认证。这个词又指麻省理工学院为这个协议开发的一套计算机软件。<sup>7</sup>

麻省理工研发了 Kerberos 协议来保护 Project Athena 提供的网络服务器。这个协议以希腊神话中的人物 Kerberos（或者 Cerberus）命名，他在希腊神话中是 Hades 的一条凶猛的三头保卫神犬。目前该协议存在一些版本，版本 1-3 都只有麻省理工内部发行。

Kerberos 版本 4 的主要设计者 Steve Miller 和 Clifford Neuman，在 1980 年末发布了这个版本。这个版本主要针对 Project Athena。版本 5 由 John Kohl 和 Clifford Neuman 设计，在 1993 年作为 RFC 1510 颁布（在 2005 年由 RFC 4120 取代），目的在于克服版本 4 的局限性和安全问题。

麻省理工在版权许可的情况下，制作了一个 Kerberos 的免费实现工具，这种情况类似于 BSD。在 2007 年，麻省理工组成了一个 Kerberos 协会，以此推动 Kerberos 的持续发展。

因为使用了 DES 加密算法（用 56 比特的密钥），美国出口管制当局把 Kerberos 归类为军需品，并禁止其出口。一个非美国设计的 Kerberos 版本 4 的实现工具 KTH-KRB 由瑞典皇家理工研制，它使得这套系统在美国更改密码出口管理条例（2000 年）前，在美国境外就可以使用。瑞典的实现工具基于一个叫做 eBones 的版本，而 eBones 基于麻省理工对外发行的基于 Kerberos 版本 4 的补丁 9 的 Bones（跳过了加密公式和对它们的函数调用）。这些在一定程度上决定了 Kerberos 为什么没有被叫做 eBones 版。Kerberos 版本 5 的实现工具，Heimdal，基本上也是由发布 KTH-KRB 的同一组人发布。

Windows2000 和后续的操作系统都默认 Kerberos 为其默认认证方法。RFC 3244 记录整理了微软的一些对 Kerberos 协议软件包的添加。RFC4757“微软 Windows2000Kerberos 修改密码并设定密码协议”记录整理了微软用 RC4 密码的使用。虽然微软使用了 Kerberos 协议，却并没有用麻省理工的软件。苹果的 Mac OS X 也使用了 Kerberos 的客户和服务

<sup>7</sup>除非特别说明，本节内容均摘录自 <https://baike.baidu.com/item/Kerberos>

器版本。Red Hat Enterprise Linux4 和后续的操作系统使用了 Kerberos 的客户和服务器版本。

在微软的网站上，有一篇介绍 Kerberos 的技术文档"Basic Concepts for the Kerberos Protocol"<sup>8</sup>，我们将其最后介绍的一个用例摘录下来，以使大家更好理解 Kerberos。

For example, consider a network with three domains, East, West, and CorpHQ. The KDC in East does not share an inter-domain key with the KDC in West, but both East and West do share inter-domain keys with CorpHQ. In this case, when a user with an account in East wants access to a server with an account in West, the referral path begins at the KDC for the user's account domain, East, passes through an intermediate domain, CorpHQ, and ends at the KDC for the server's account domain, West. The client must send its request for a session ticket three times, to three different KDCs.<sup>9</sup>

1. The client asks the KDC for East to give it a ticket to the server in West. The KDC for East sends the client a referral ticket to the KDC for CorpHQ. This ticket is encrypted in the interdomain key East shares with CorpHQ.
2. The client asks the KDC for CorpHQ to give it a ticket to the server in West. The KDC for CorpHQ sends the client a referral ticket to the KDC for West. This ticket is encrypted in the interdomain key CorpHQ shares with West.
3. The client asks the KDC for West to give it a ticket to the server in West. The KDC in West replies by sending a ticket for the server in West.

## 12.5 Windows Logon

Windows 操作系统是日常办公常用的操作系统之一，下面我么摘录关于 LSA 认证的介绍<sup>10</sup>，摘录的介绍不涉及一些细节，有兴趣的同学可以通过这个内容扩展了解。

LSA Authentication describes the parts of the Local Security Authority (LSA) that applications can use to authenticate and log users on to the local system. It also describes how to create and call authentication packages and security packages.

The LSA Authentication functions let you write an authentication package, a subauthentication package, or a combined security support provider/authentication package (SSP/AP).

The following topics provide more information about LSA Authentication:

1. LSA Authentication Model
2. LSA Logon Sessions
3. LSA User Logon Authentication

<sup>8</sup>[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc961976\(v=technet.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc961976(v=technet.10)?redirectedfrom=MSDN)

<sup>9</sup>通过这段英文来理解 Kerberos 基本过程，如果是课堂上课，可以与同学们一起读这段文字，来共同画一张图理解。

<sup>10</sup>摘录的内容链接为<https://docs.microsoft.com/zh-cn/windows/win32/secauthn/lsa-authentication>

#### 4. Authentication Packages

For information about the LSA Authentication functions, see Authentication Reference.

For information about related technologies, see LSA Policy and Winlogon and Credential Providers.

## 12.6 TLS/SSL

传输层安全性协议（英语：Transport Layer Security，缩写作 TLS），及其前身安全套接层（Secure Sockets Layer，缩写作 SSL）是一种安全协议，目的是为互联网通信提供安全及数据完整性保障。网景公司（Netscape）在 1994 年推出首版网页浏览器，网景导航者时，推出 HTTPS 协议，以 SSL 进行加密，这是 SSL 的起源。IETF 将 SSL 进行标准化，1999 年公布第一版 TLS 标准文件。随后又公布 RFC 5246（2008 年 8 月）与 RFC 6176（2011 年 3 月）。在浏览器、邮箱、即时通信、VoIP、网络传真等应用程序中，广泛支持这个协议。主要的网站，如 Google、Facebook 等也以这个协议来创建安全连线，发送数据。目前已成为互联网上保密通信的工业标准。SSL 包含记录层（Record Layer）和传输层，记录层协议确定传输层数据的封装格式。传输层安全协议使用 X.509 认证，之后利用非对称加密演算来对通信方做身份认证，之后交换对称密钥作为会谈密钥（Session key）。这个会谈密钥是用来将通信两方交换的数据做加密，保证两个应用间通信的保密性和可靠性，使客户与服务器应用之间的通信不被攻击者窃听。<sup>11</sup>

## 12.7 IPv6

IPv6，顾名思义，就是 IP 地址的第 6 版协议。我们现在用的是 IPv4，你的外网地址可能是这样一串数字：59.123.123.123。IPv4 的地址是 32 位，总数有 43 亿个左右，还要减去内网专用的 192、170 地址段，就更少了。而 IPv6 的地址是 128 位的，大概是 43 亿的 4 次方，地址极为丰富，几乎是取之不尽的，打个比方，地球上的每一粒沙子都能分配到自己的地址。<sup>12</sup>

IPv6 把 IPSec 做为必备协议，从而保证了网络层端到端通信的完整性和机密性。

## 12.8 5G AKA

5G AKA 是 5G Authentication and Key Agreement 的缩写<sup>13</sup>。随着通信网络技术的发展，第 5 代移动通信网络被提上日程。5G 通信网络的设计目标面向 3 大场景：增强型移动宽带（eMBB）、高可靠低时延（uRLLC）以及海量机器类通信（mMTC）。因此，5G

<sup>11</sup><https://baike.baidu.com/item/TLS>

<sup>12</sup><https://blog.csdn.net/u012997396/article/details/88707262>

<sup>13</sup>本节内容摘录自：

齐曼鹏, 彭晋.5G 网络的认证体系 [J]. 中兴通讯技术, 2019, 25(04):14-18.

通信不仅考虑人与人之间的通信，还将考虑人与物、物与物之间的通信，进入万物互联的状态。

这种情况下，5G 认证面临着新的安全需求。一方面，为了适应多种类型的通信终端，并使得它们能够接入通信网络，5G 系统将进一步地扩展非蜂窝技术的接入场景。例如，电表、水表、摄像头等物联网设备通常采用蓝牙、WLAN 等技术与网络相连，这类设备采用 5G 系统通信时仍倾向于使用原有的连接方式。这就导致传统面向蜂窝接入的认证机制需要进一步地向非蜂窝接入的方式扩展。另一方面，传统认证机制下，拜访地/归属地的两级移动网络架构下的认证机制要求归属网络无条件信任拜访网络的认证结果。但随着网络的发展，出现了越来越多的安全隐患，拜访网络和归属网络之间的信任程度在不断降低。例如，拜访地运营商可以声称为某运营商的用户提供了接入服务而实际未提供，导致计费纠纷。对于 5G 的通信来说，相比于人与人之间的语音通信和数据交互，万物互联下的移动通信将会承载更多的设备测控类信息，因此对接入安全的要求更高。例如，当 5G 系统被垂直行业用于传递远程操控的控制消息。这使得 5G 认证还需要加强归属网络对用户终端的认证能力，使其摆脱对拜访网络的依赖，实现用户在归属地和拜访地等不同地点间的认证机制统一。5G 还引入了对用户身份的进一步增强保护，使得用户的永久身份不在空中接口上进行传输，拜访网络还需要从归属网络获得用户的身份信息。这就导致拜访网络和用户终端之间无法直接对身份信息进行确认。因此，为了简化设计，5G 认证过程中对用户的认证信息也需要进行确认。

基于上述需求，5G 提供了 2 种认证框架，分别被称为 5G-AKA 和 EAP-AKA。

# 第 13 章 协议的安全分析

协议的分析方法有很多，针对不同的目标，有不同的形式化分析方法，我们这里主要介绍针对某个或某些安全目标的协议分析理论和方法。对于形式化分析方法，构造的形式化系统的描述能力和推理能力是很重要的两方面。

## 13.1 BAN 逻辑 [2]<sup>1</sup>

### 13.1.1 基本概念

BAN 逻辑是一种基于信念的模态逻辑，是 1989 年由 Michael Burrows, Martin Abadi 和 Roger Needham 提出的<sup>2</sup>。在 BAN 逻辑的推理过程中，参加协议的主体的信念随消息交换的发展，不断变化和发展。应用 BAN 逻辑时，首先需进行“理想化步骤”，将协议的消息转换为 BAN 逻辑中的公式，再根据具体情况进行合理的假设，由逻辑的推理规则根据理想化协议和假设进行推理，推断协议能否完成预期的目标，依照 BAN 逻辑的惯例，A,B,S 通常表示具体的主体； $K_{ab}, K_{as}, K_{bs}$  等表示具体的共享密钥； $K_a, K_b, K_c$  等表示具体的公开密钥； $K_a^{-1}, K_b^{-1}, K_c^1$  等表示相应的秘密密钥； $N_a, N_b, N_c$  等表示临时值；P,Q 和 R 表示任意主体；X 和 Y 表示任意语句；K 表示任意密钥。

BAN 逻辑的结构如下，其中 P,Q,R,X,Y,K 等的含义如上所述。

表 13.1: BAN 逻辑符号定义

语义	符号表示
P believes X	P 相信 X
P sees X	P 曾收到 X
P said X	P 曾发送 X
P controls X	P 对 X 有管辖权
(X,Y)	X 和 Y 链接
fresh(X)	X 是新的
$X_K$	用 K 加密 X 后的结果
$P \xleftrightarrow{\kappa} Q$	P 和 Q 可用共享密钥 $\kappa$ 通信
H(X)	X 是单向杂凑函数
$\xrightarrow{\kappa} P$	$\kappa$ 是 P 的公开密钥
$\langle X \rangle_Y$	$\langle X, Y \rangle \wedge Y$ 是某种秘密

### 13.1.2 逻辑公设

BAN 逻辑的主要有以下 5 条逻辑公设：

<sup>1</sup>本节内容摘录自卿斯汉老师“密码学与计算机网络安全”一书中的“11.1 认证协议和 BAN 逻辑”。

<sup>2</sup>论文公开发表于 1990 年，Michael Burrows, Martin Abadi, and Roger Needham. 1990. A logic of authentication. ACM Trans. Comput. Syst. 8, 1 (Feb. 1990), 18–36. DOI: <https://doi.org/10.1145/77648.77649>

1. 消息含义法则逻辑公设

$$\frac{P \text{ believes } Q \xleftrightarrow{K} P, P \text{ sees } X_K}{P \text{ believes } Q \text{ said } X}$$

这一逻辑公设说明：如果  $P$  相信  $K$  是  $P$  和  $Q$  之间的共享密钥，且  $P$  曾收到用  $K$  加密  $X$  后的结果，则  $P$  相信  $Q$  曾发送过  $X$ 。对于公钥系统，我们有类似的公设：

$$\frac{P \text{ believes } \xleftrightarrow{K} Q, P \text{ sees } X_{K^{-1}}}{P \text{ believes } Q \text{ said } X}$$

2. 临时值校验法则逻辑公设

$$\frac{P \text{ believes } \text{fresh}(X), P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}$$

这一逻辑公设说明：如果  $P$  相信消息  $X$  是新的，且  $P$  相信  $Q$  曾发送过  $X$ ，则  $P$  相信  $Q$  相信  $X$ 。

3. 管辖法则逻辑公设

$$\frac{P \text{ believes } Q \text{ controls } X, P \text{ believes } Q \text{ believes } X}{P \text{ believes } X}$$

这一逻辑公设说明：如果  $P$  相信  $Q$  对  $X$  具有管辖权，且  $P$  相信  $Q$  相信  $X$ ，则  $P$  相信  $X$ 。

4. 逻辑公设

$$\frac{P \text{ sees } (X, Y)}{P \text{ sees } X} \quad \frac{P \text{ sees } < X >_Y}{P \text{ sees } X}$$

$$\frac{P \text{ believes } Q \xleftrightarrow{K} P, P \text{ sees } X_K}{P \text{ sees } X}$$

$$\frac{P \text{ believes } \xleftrightarrow{K} P, P \text{ sees } X_K}{P \text{ sees } X}$$

$$\frac{P \text{ believes } \xleftrightarrow{K} Q, P \text{ sees } X_{K^{-1}}}{P \text{ sees } X}$$

这一逻辑公设说明：如果一个主体曾收到一个公式，且该主体知道相关的密钥，则该主体曾收到该公式的组成部分。

5. 逻辑公设

$$\frac{P \text{ believes } \text{fresh}(X)}{P \text{ believes } \text{fresh}(X, Y)}$$

这一逻辑公设说明：如果一个公式的一部分是新的，则该公式全部是新的。

### 13.1.3 推理步骤

BAN 逻辑的推理步骤是：

- (1) 建立初始假设集合  $\alpha$ ;
- (2) 建立理想化协议模型;
- (3) 建立协议预期目标集合  $\gamma$ ;
- (4) 利用初设和逻辑公设推理;
- (5) 推导出协议最终目标集合  $\Gamma$ ;

(6) 若  $\Gamma \supseteq \gamma$ , 则协议可行.

以上步骤可能会重复进行, 例如: 通过分析增加新的初设、改进理想化协议等. 通过 BAN 逻辑分析, 可以回答下述问题: 认证协议是否正确<sup>3</sup>, 认证协议的目标是否达到, 认证协议的初设是否合适, 认证协议是否冗余.

### 13.1.4 分析 Needham-Schroeder 协议

#### 13.1.4.1 初始假设集合

$$\begin{array}{ll}
 A \text{ believes } A \xleftrightarrow{K_{as}} S & B \text{ believes } B \xleftrightarrow{K_{bs}} S \\
 S \text{ believes } A \xleftrightarrow{K_{as}} S & S \text{ believes } B \xleftrightarrow{K_{bs}} S \\
 S \text{ believes } A \xleftrightarrow{K_{ab}} B & \\
 A \text{ believes } S \text{ controls } A \xleftrightarrow{K} B & B \text{ believes } S \text{ controls } A \xleftrightarrow{K} B \\
 A \text{ believes } S \text{ controls } \text{fresh}(A \xleftrightarrow{K} B) & \\
 A \text{ believes } \text{fresh}(N_a) & B \text{ believes } \text{fresh}(N_b) \\
 S \text{ believes } \text{fresh}(A \xleftrightarrow{K} B) & B \text{ believes } \text{fresh}(A \xleftrightarrow{K} B)
 \end{array}$$

以上大部分初始假设都是自然的, 第一组 5 个初始假设涉及主体拥有的初始密钥。下面 3 个初始假设说明客户相信认证服务器所具有的功能。如同上例, A 和 B 相信认证服务器 S 能为 A 和 B 生成新的会话密钥。但是, 在此处 A 还相信 S 所生成的会话密钥同时具有临时值的性质。以上初始假设是有道理的, 因为一个“好”的加密密钥通常具备临时值的特征。最后 4 个初始假设指出, 相关的临时值是新的。

下面要特别讨论的是最后一个初始假设

$$B \text{ believes } \text{fresh}(A \xleftrightarrow{K} B)$$

这个假设是不寻常的。在下面的讨论中, 我们将说明, 许多对此协议的批评均来自这个假设, 而协议的发明者并未认识到他们实际上应用了这一假设。

#### 13.1.4.2 理想化协议模型

去掉与协议分析无关部分, 理想化协议模型为:

1.  $S \rightarrow A : \left\{ N_a, (A \xleftrightarrow{K_{ab}} B), \text{fresh}(A \xleftrightarrow{K_{ab}} B), (A \xrightarrow{K})_{K_{bs}} \right\}_{K_{as}}$
2.  $A \rightarrow B : (A \xleftrightarrow{K_{ab}} B)_{K_{bs}}$
3.  $B \rightarrow A : \left\{ N_b, A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$  from B
4.  $A \rightarrow B : \left\{ N_b, A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$  from A

在 3, 4 条中包含了发送者的名称, 为的是区分两条消息, 避免混淆。在原协议中, 区分的方法是通过  $N_b$  和  $N_b - 1$  实现的, 事实上, 在原协议中  $N_b - 1$  可以用任何  $N_b$  的函数来取代。

---

<sup>3</sup>何为正确?

在理想化协议模型中，1, 3, 4 条消息中关于  $K_{ab}$  的语句说明，A 可以将会话密钥  $K_{ab}$  作为临时值应用，同时每个客户均认可对方相信该会话密钥  $K_{ab}$  是“好”密钥。

### 13.1.4.3 建立协议预期目标

协议的预期目标是：

$$A \text{ believes } A \xleftrightarrow{K_{ab}} B$$

$$B \text{ believes } A \xleftrightarrow{K_{ab}} B$$

### 13.1.4.4 利用初设和逻辑公设推理

• 消息  $S \rightarrow A : \left\{ N_a, (A \xleftrightarrow{K_{ab}} B), \text{fresh}(A \xleftrightarrow{K_{ab}} B), (A \xrightarrow{K_{ab}})_{K_{bs}} \right\}_{K_{as}}$  的推导链条

由消息  $S \rightarrow A : \left\{ N_a, (A \xleftrightarrow{K_{ab}} B), \text{fresh}(A \xleftrightarrow{K_{ab}} B), (A \xrightarrow{K_{ab}})_{K_{bs}} \right\}_{K_{as}}$ ，我们可知：

$$A \text{ sees } \left\{ N_a, (A \xleftrightarrow{K_{ab}} B), \text{fresh}(A \xleftrightarrow{K_{ab}} B), (A \xrightarrow{K_{ab}})_{K_{bs}} \right\}_{K_{as}}$$

$$\frac{A \text{ believes } A \xleftrightarrow{K_{as}} S; \frac{P \text{ believes } Q \xrightarrow{K} P, P \text{ sees } X}{P \text{ believes } Q \text{ said } X}}{\Rightarrow}$$

$$A \text{ believes } S \text{ said } (N_a, A \xleftrightarrow{K_{ab}} B, \text{fresh}(A \xleftrightarrow{K_{ab}} B))$$

$$\frac{A \text{ believes } \text{fresh}(N_a); \frac{P \text{ believes } \text{fresh}(X), P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}}{\Rightarrow}$$

$$\begin{cases} A \text{ believes } S \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \text{记为 Result1} \\ A \text{ believes } S \text{ believes } \text{fresh}(A \xleftrightarrow{K_{ab}} B) \dots \text{记为 Result2} \end{cases}$$

$$\frac{A \text{ believes } S \text{ controls } A \xleftrightarrow{K} B; A \text{ believes } S \text{ controls } \text{fresh}(A \xleftrightarrow{K})}{\frac{P \text{ believes } Q \text{ controls } X, P \text{ believes } Q \text{ believes } X}{P \text{ believes } X}} \Rightarrow$$

$$\begin{cases} A \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \text{记为 Result3} \\ A \text{ believes } \text{fresh}(A \xleftrightarrow{K_{ab}} B) \dots \text{记为 Result4} \end{cases}$$

另外：

$$A \text{ sees } \left\{ N_a, (A \xleftrightarrow{K_{ab}} B), \text{fresh}(A \xleftrightarrow{K_{ab}} B), (A \xrightarrow{K_{ab}})_{K_{bs}} \right\}_{K_{as}}$$

$$\frac{\text{消息中包含要推出内容}}{\Rightarrow}$$

$$A \text{ sees } \{A \xleftrightarrow{K_{ab}} B\} \dots \text{记为 Result5}$$



- 消息  $A \rightarrow B : (A \xleftrightarrow{K_{ab}} B)_{K_{bs}}$  的推导链条

由消息  $A \rightarrow B : (A \xleftrightarrow{K_{ab}} B)_{K_{bs}}$ , 我们可知:

$$B sees (A \xleftrightarrow{K_{ab}} B)_{K_{bs}}$$

$$\frac{B believes B \xleftrightarrow{K_{bs}} S; \frac{P believes Q \xleftrightarrow{K} P, P sees X_K}{P believes Q said X}}{\xrightarrow{\quad\quad\quad}}$$

$$B believes S said A \xleftrightarrow{K_{ab}} B$$

因为 B 只能假定来自认证服务器 S 的消息是新的, 而不能断定 A 发送的消息是新的还是重放的消息, 所以, 我们无法继续分析和证明下去, 除非加上那条可疑的初始假设:

$$B believes fresh (A \xleftrightarrow{K} B)$$

一旦利用这个初始假设, 我们可以继续推导。

$$B believes S said A \xleftrightarrow{K_{ab}} B$$

$$\frac{B believes fresh (A \xleftrightarrow{K} B); \frac{P believes fresh(X), P believes Q said X}{P believes Q believes X}}{\xrightarrow{\quad\quad\quad}}$$

$$B believes S believes A \xleftrightarrow{K} B \dots\dots \text{记为 Result6}$$

$$B believes S said A \xleftrightarrow{K_{ab}} B$$

$$\frac{B believes S controls A \xleftrightarrow{K} B; \frac{P believes Q controls X, P believes Q believes X}{P believes X}}{\xrightarrow{\quad\quad\quad}}$$

$$B believes A \xleftrightarrow{K_{ab}} B \dots\dots \text{记为 Result7}$$

我们根据 Result3: $A believes A \xleftrightarrow{K_{ab}} B$  和消息 “ $B \rightarrow A : \left\{ N_b, A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$  from B” 可以进行如下推理:

$$B \rightarrow A : \left\{ N_b, A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$$

$$\xrightarrow{\quad\quad\quad A \text{收到消息}}$$

$$A sees \left\{ A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$$

$$\frac{A believes A \xleftrightarrow{K_{ab}} B; \frac{P believes Q \xleftrightarrow{K} P, P sees X_K}{P believes Q said X}}{\xrightarrow{\quad\quad\quad}}$$



$A \text{ believes } B \text{ said } A \xleftrightarrow{K_{ab}} B$

Result4;  $\frac{P \text{ believes } \text{fresh}(X), P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}$

$A \text{ believes } B \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots$  记为 Result8

类似，我们根据消息：“ $A \rightarrow B : \left\{ N_b, A \xleftrightarrow{K_{ab}} B \right\}_{K_{ab}}$  from A” 可以推出：

$B \text{ believes } A \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots$  记为 Result9

### 13.1.4.5 推导出协议最终目标集合

至此为止，我们推导出的最终目标集合有 Result3、Result7、Result8、Result9，即：

$\left\{ \begin{array}{l} A \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots \text{记为 Result3} \\ B \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots \text{记为 Result7} \\ A \text{ believes } B \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots \text{记为 Result8} \\ B \text{ believes } A \text{ believes } A \xleftrightarrow{K_{ab}} B \dots \dots \text{记为 Result9} \end{array} \right.$

将最终目标集合和预期目标集合比较，可知预期目标集合是最终目标集合的子集，也就是说协议达到预期目的。

### 13.1.4.6 讨论

Needham-Schroeder 协议得出了较强的结论，其代价是：必须额外假定 B 获得的会话密钥时新的会话密钥。这一代价是巨大的，一次招致许多批评，也为 Needham-Schroeder 协议的进一步改进提供了依据。主要原因是：一旦会话密钥被破译会带来严重的后果。攻击者可以有无限的时间去寻找一个旧的会话密钥，然后重新使用它，好像这个密钥时新的密钥。很明显，问题出在 B 和认证服务器 S 没有打交道。这样，上述 BAN 逻辑的形式化分析过程就为 Needham-Schroeder 协议的改进提供了方向。例如，一种改进方法是：认证服务器 S 先发送消息给 B 而不是 A，这里就不赘述了。

另外，从 BAN 逻辑的形式化分析过程我们也看到一些 Needham-Schroeder 协议的冗余性。例如，在第 2 条消息中，认证证书  $\{K_{ab}, A\}_{K_b}$  通过 A 的密钥  $K_{as}$  加密是没有必要的。因为，这个认证证书随后立即由 A 发送给 B，没有再继续加密。从 BAN 逻辑形式化分析的整个过程来看，这种加密丝毫不影响分析和证明的过程，旁证了认证证书  $\{K_{ab}, A\}_{K_b}$  通过 A 的密钥  $K_{as}$  加密的冗余性。

这个例子证明了 BAN 逻辑在分析认证协议时的重要作用，它可以证明协议的正确性和安全性，发现协议中不易察觉的缺陷，指出进一步改进认证协议的方向。

## 13.2 逻辑编程和协议分析工具

除了使用一些针对安全协议分析设计的逻辑工具外，有一些通用的逻辑编程语言，如 Prolog、Datalog 可以用于协议分析，还有一些形式化语言，比如 Z 语言，也可以用于协

议安全分析。

# 第 14 章 安全多方计算 (Secure Multi-Party Computation)

---

多方计算问题是由姚期智在其 1982 年发表的文章“Protocols for secure computations”<sup>1</sup>中提出，在其文章的第一段对这个问题描述为：

Two millionaires wish to know who is richer; however, they do not want to find out inadvertently any additional information about each other's wealth. How can they carry out such a conversation?

这个问题也可以描述为：一组互不信任的参与方之间在保护隐私信息以及没有可信第三方的前提下协同计算问题。姚期智在其文章中，给出了“多方安全计算”(Secure Multi-Party Computation，简称 MPC) 的理论框架。

随着大数据、机器学习的应用的不断普及，隐私问题越来越被关注，多方计算的实践价值逐渐凸显，2019 年《人民日报》就隐私计算技术采访了姚期智先生<sup>2</sup>。采访中姚期智如此解释 MPC：“我们两个人中每个人有一个数据，想要两个人数据合起来，但不想把数据交给对方。我们希望使这个计算实现，但是完全不透露我们的数据是什么。我提出这个概念的时候，完全出于科学的好奇心。现在，这个方向成为密码安全领域的一个大方向。”，他认为，多方安全计算会在金融科技甚至人工智能、医药保护共享数据等方面发挥重要作用。值得一提的是，姚期智在当时的采访中表示，MPC 也将是中国贡献给世界的一个原创关键技术。提出 MPC 的 4 年后，姚期智于 1986 年提出了基于混淆电路的通用解决方案，进一步验证了多方安全计算的通用可行性，同时也奠定了现代计算机密码学的理论基础。此后，经 Oded Goldreich、Shafi Goldwasser 等密码学学者进一步的研究和创新，多方安全计算逐渐发展成为现代密码学的一个重要分支。

在此篇报道中也罗列了一些多方计算的应用场景，我们摘抄部分内容：

MPC 之所以近几年开始受到关注，一方面是因为产业互联网、AI 等关键领域的发展越来越离不开数据上云、离不开数据挖掘，数据隐私问题的解决迫在眉睫。

MPC 已经成长到了一定的阶段，其产业价值潜力开始彰显，特别是在涉及隐私敏感型输入数据（如客户行为信息、身份信息、金融信息、征信信息、医疗信息）的应用场景。

拥有隐私敏感型数据的金融、物流、供应链、物联网、汽车业，都会是 MPC 很有应用价值的地方。而且，在解决数据隐私问题的同时，数据孤岛的困境也能得到缓解，因为一部分数据孤岛现象存在正是基于数据隐私的考量。

<sup>1</sup>Yao A C . Protocols for secure computations[C] Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982.

<sup>2</sup>采访报道为“姚期智 40 年前提出，“百万富翁”设想走进现实：这项隐私计算技术，将是下一个产业热点？”，网址：[https://www.sohu.com/a/329506304\\_354973](https://www.sohu.com/a/329506304_354973)

尤其对于以海量数据作为训练根基、正在隐私保护合规中寻求落地的 AI 技术来说，这将是一个好消息。

以医疗场景中的基因数据为例，基因数据具有隐私性要求高、数据体量大的特点。此前就有业内人士表示，“生物信息是个人信息安全的最后一道防线”。目前，基因数据一般会保存在研究机构或者医疗公司的本地系统中，但这些“新石油”处于共享、流通的状态其实才更利于生物医疗技术的发展，例如基于基因数据挖掘研究某种疾病，开发出更有针对性的药物等等。例如，如果不同的机构能够部署 MPC 节点，那么，这些数据就可以通过 MPC 协议间接实现数据共享：基因数据仍保留在本地，但是不同的机构可以共同实现计算出需要的数据结果。类似的项目已经在国内出现。2018 年，民生健康（万向区块链和民生人寿保险有限公司共同成立）就和宁波保险行业协会合作，以健康险为业务场景，模拟联盟内保险公司之间的数据查询，证明了 MPC 在建立共享价值网络上是完全可行的。不过，需要指出，由于模拟的数据规模较小，那个项目并没有产生实际的商业价值。

NIST 网站对 SMPC 的定义和解释如下，供大家参考阅读<sup>3</sup>：

*Secure multi-party computation solves the following problem: several parties have private inputs; they wish to compute a function of these inputs without compromising the privacy of the inputs. An example of such a scenario is elections via secret ballot. Another is the so-called "dating problem", in which two people would like to know if they both like each other but each party is loath to say "I like you" only to be rejected by the other.*

*Another example, extracted from this author's academic experience, is the following. An 11-member Departmental Executive Committee is to vote on a contentious academic issue. The motion passes if 7 or more "yes" votes are received. For maintaining the secrecy of the vote it is critical that the actual number of votes is not revealed. That is, the Committee only wishes to know the result of the Boolean function  $F(\text{votes}) = \text{"at least seven yes votes"}$ .*

*Solutions to these protocol problems exist, but can be costly and even impractical, as they are typically based on challenge-response methods. The "authenticated broadcast" feature of the Beacon can be leveraged to construct practical solutions to many of these problems in the online environment.*

多方计算的产业化进程方面也有多方力量进入。

"作为 MPC 的提出者和重要奠基人，姚期智所在的清华交叉院与清华大学、清华五道口金融学院于 2018 年 6 月联合成立了华控清交信息科技（北京）有限公司（下称“华控清交”），华控清交专注于研究、开发和营运基于密码学的 MPC 技术、标准和基础设施。团队通过综合运用密码学混淆电路、不经意传输、秘密分享、同态加密、同态承诺、

---

<sup>3</sup>Beacon Project-Secure Multi-party Computation, <https://www.nist.gov/itl/csd/cryptographic-technology/beacon-project-secure-multi-party-computation>

零知识证明等多种理论和协议，结合计算机工程技术，研发出了一个软硬件结合的多方安全计算平台。据介绍，这个计算平台可以在多方输入且不暴露输入信息的情况下进行密文计算，最终得出与明文一致的密文计算结果，可支持涵盖 AI 算法训练在内的几乎全部计算类型和多种数据格式。目前，华控清交已经在金融行业多方联合风控、多方联合建模，能源行业风电效率优化、政府领域电子政务等场景有具体落地和试点项目。”

“早在 2012 年，蚂蚁金服就开始研究 MPC，2019 年 5 月，蚂蚁金服推出其基于 MPC 的安全计算平台“摩斯”，据称提供了一种全新的安全和保护隐私的数据合作方式，能够在本地数据不泄露、原始数据不出域的前提下，通过密码学算法，分布式执行既定逻辑的运算并获得预期结果，高效、安全地完成数据合作。目前，“摩斯”已广泛应用于联合金融风控、保险快速理赔、民生政务、多方联合营销、多方联合科研、跨境数据合作等多个领域。”

## 14.1 姚的百万富翁方案

姚期智在其 1982 年发表的文章 “Protocols for secure computations” 中首先给出了一个百万富翁的解决方案。

设 Alice 有  $i$  百万，Bob 有  $j$  百万，并且  $1 \leq i, j \leq 10$ ，我们需要一个协议来判断 “ $i$  是否小于  $j$ ”，并且最后我们只能得到这个信息(也就是说没有获得多余的有关  $i, j$  的信息)，设  $M$  是所有  $N$  比特非负整数集合， $Q_N$  是所有  $M$  到  $M$  的 1-1 满射函数 (onto function) 集合， $E_a$  是 Alice 的公钥，从  $Q_N$  中随机选取的。

姚方案如下：

1. Bob 取一个  $N$  比特随机数  $x$ ，计算  $k = E_a(x)$ 。
2. Bob 把  $k - j$  发给 A.<sup>4</sup>
3. Alice 计算  $y_u = D_a(k - j + u), u = 1, 2, \dots, 10$ .
4. Alice 随机选一个  $N/2$  比特随机素数  $p$ ，计算  $z_u = y_u \pmod{p}, u = 1, 2, \dots, 10$ ，如果所有  $z_u$  在模  $p$  下至少相差 2，则停止，否则产生新的  $p$ ，直至条件成立，也就是说直至  $|z_u - z_v| \geq 2, u, v \in \{1, 2, \dots, 10\}, u \neq v$ 。<sup>5</sup>
5. Alice 把  $p$  和是个数： $z_1, z_2, \dots, z_i, z_{i+1} + 1, z_{i+2} + 1, \dots, z_{10} + 1$  发送给 Bob。以上数都是在模  $p$  的运算。
6. Bob 取 Alice 发来的第  $j$  个数  $w$ ，如  $w = x \pmod{p}$ ，那么  $i \geq j$ ，否则  $i < j$ .
7. Bob 告诉 Alice 比较结果。

首先我们可以看到在 Alice 计算  $y_u$  时，当  $u = j$  时， $y_j = D_a(k) = x$ ，所以当 Bob 选择这个位置上的数时，如果是  $x$  表示  $i$  在  $j$  之后，因为没有改变，如果  $i$  在  $j$  之前，第  $j$  个数应该被修改。

在上面的讨论中其实我们有一个假设，就是 Alice 和 Bob 都是诚实的，或者说可信的，而在实际中，这是个很强的假设，我们把这个假设强度降低一些，如果我们认为计

<sup>4</sup>原文中是: $k - j + 1$

<sup>5</sup>此处要求相差为 2，是因为在下面的步骤中通过加一改变了原数值，而这种改变应该是能够与原值进行区分的。

算方存在获取其他计算方原始数据的需求，但仍按照计算协议执行，我们称其为“半诚实对手模型”，半诚实对手模型参与方之间有一定信任关系，如果将假设强度再降低，认为，参与方根本就不按照计算协议执行计算过程，能够随意中断协议的运行，破坏协议的正常执行过程，也能随意修改协议的中间结果或者与其他参与方相互勾结，我们称这种情况为“恶意对手模型”，在恶意对手模型中参与方可采用任何（恶意）方式与对方通信，且没有任何信任关系，结果可能是协议执行不成功，双方得不到任何数据；或者协议执行成功，双方仅知道计算结果。

姚在 1982 年的文章中提出了一个安全多方计算 (Secure Multi-party computation, 简称 MPC) 的理论框架，四年后，在 1986 年的文章 “How to generate and exchange secrets”<sup>6</sup> 中提出了基于混淆电路 (Garbled Circuits) 的通用解决方案，进一步验证了多方安全计算的通用可行性。

## 14.2 同态加密

1978 年 RL Rivest, LM Adleman, ML Dertouzos 发表了一篇文章 "On Data Banks and Privacy Homomorphisms"<sup>7</sup>，通常大家将这篇文章看做同态加密的开山之作，下面我们介绍一下这篇文章里提出的隐私同态方法的核心思想。

首先将计算场景抽象为两个代数系统一个是用户代数系统 U<sup>8</sup>，一个是计算机系统的代数系统 C：

$$U = \langle S; f_1, \dots, f_k; p_1, \dots, p_l; s_1, \dots, s_m \rangle$$

$$C = \langle S'; f'_1, \dots, f'_k; p'_1, \dots, p'_l; s'_1, \dots, s'_m \rangle$$

其中 S、S' 是集合， $f_i, f'_i (i = 1, \dots, k)$  是函数， $p_i, p'_i (i = 1, \dots, l)$  是谓词， $s_i, s'_i (i = 1, \dots, m)$  是常数。

C 从实现角度来看，就是系统中的一些子程序，用来计算的  $f'_i$  和  $p'_i$  的。

同时，我们有一个编码函数  $\phi : S' \rightarrow S$  和他的逆函数，解码函数  $\phi^{-1} : S \rightarrow S'$ ，用户要处理的数据序列为  $d_1, d_2, \dots, d_i \in S, i = 1, 2, \dots$ ，用户把这些数据在 C 系统中处理前，先对这些数据进行加密  $\phi^{-1}(d_1), \phi^{-1}(d_2), \dots$

为了使系统能够在加密的数据上进行处理，函数  $\phi$  必须是 C 到 U 的同态映射，也就是说满足一下三条：

1.  $f'_i(a, b, \dots) = c \Rightarrow f_i(\phi(a), \phi(b), \dots) = \phi(c)$
2.  $p'_i(a, b, \dots) \equiv p_i(\phi(a), \phi(b), \dots)$
3.  $\phi(s'_i) = s_i$

<sup>6</sup>A. C. Yao, "How to generate and exchange secrets," 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), 1986, pp. 162-167, doi: 10.1109/SFCS.1986.25.

<sup>7</sup>Rivest R L, Adleman L M, Dertouzos M L. On Data Banks and Privacy Homomorphisms[J]. Foundations of Secure Computation, 1978.

<sup>8</sup>比如一个实例是  $\langle Z; +, -, \times, \div; 0, 1 \rangle$ ，Z 为整数集。

下面我们看看数据操作过程,假定用户 U 想让 C 计算  $f_1(d_1, d_2)$ ,他把数据  $\phi^{-1}(d_1), \phi^{-1}(d_2)$  给 C, C 计算  $f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))$ , 对结果使用  $\phi$  运算:

$$\phi(f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))) = f_1(d_1, d_2)$$

从而实现在不透露  $d_1, d_2$  的前提下, 实现运算。

代数系统 C 和函数  $\phi, \phi^{-1}$  在选择时应该满足一下条件:

1.  $\phi, \phi^{-1}$  容易计算。
2. C 中的  $f'_i, p'_i$  可以有效计算。
3. 加密后的数据  $\phi^{-1}(d_i)$  和原数据  $d_i$  相比, 不应该占用更多的存储空间。
4. 获得多个  $\phi^{-1}(d_i)$  后, 不会暴露  $\phi$ 。(唯密文攻击)
5. 获得多个  $d_i$  和其对应的  $\phi^{-1}(d_i)$  后, 不会暴露  $\phi$ 。(选择明文攻击)
6. 根据 C 中的  $f_i, p_i$  不能获得  $\phi$ 。

## 14.3 开源项目

有很多开源项目可以作为我们做相关研究和系统实现的基础, 下面我们介绍几个。

SEAL, 这是 Microsoft 的同态加密库, 由微软的 Cryptography Research 小组开发, 原始仓库: <https://github.com/microsoft/SEAL>, 国内 Gitee 上有个镜像, 每日同步一次, 网址为[https://gitee.com/mirrors/SEAL?utm\\_source=alading&utm\\_campaign=repo](https://gitee.com/mirrors/SEAL?utm_source=alading&utm_campaign=repo)。

HElib 是一个开源的同态加密库, 仓库: <https://github.com/homenc/HElib>, 其项目 About 内容为 “HElib is an open-source software library that implements homomorphic encryption. It supports the BGV scheme with bootstrapping and the Approximate Number CKKS scheme. HElib also includes optimizations for efficient homomorphic evaluation, focusing on effective use of ciphertext packing techniques and on the Gentry-Halevi-Smart optimizations.” .

HEAAN 是一个开源的同态加密库, 仓库: <https://github.com/snucrypto/HEAAN>, 其项目 README 第一段的说明为 “HEAAN is software library that implements homomorphic encryption (HE) that supports fixed point arithmetics. This library supports approximate operations between rational numbers. The approximate error depends on some parameters and almost same with floating point operation errors. The scheme in this library is on the paper "Homomorphic Encryption for Arithmetic of Approximate Numbers" (<https://eprint.iacr.org/2016/421.pdf>).” .

FHE 是 Google 的一个开源全同态加密库, 仓库: <https://github.com/google/fu-fully-homomorphic-encryption>, Gitee 上有镜像[https://gitee.com/mirrors\\_google/fu-fully-homomorphic-encryption](https://gitee.com/mirrors_google/fu-fully-homomorphic-encryption), 项目的 README 第一段的说明为 “This repository contains open-source libraries and tools to perform fully homomorphic encryption (FHE) operations on an encrypted data set.”

# 第 15 章 比特币 (Bitcoin) & 区块链 (Block Chains)

---

## 15.1 比特币起源

<sup>1</sup> 2008 年 11 月 1 日 (也有称准确的时间是 2008 年 10 月 31 日), Satoshi Nakamoto(中文翻译为“中本聪”)在“metzdowd.com”网站的密码学邮件列表中发表了一篇论文, 题为《Bitcoin: A Peer-to-Peer Electronic Cash System》(中文翻译为《比特币: 一种点对点式的电子现金系统》)<sup>2</sup>。论文中详细描述了如何创建一套去中心化的电子交易体系, 且这种体系不需要创建在交易双方相互信任的基础之上。很快, 2009 年 1 月 3 日, 他开发出首个实现了比特币算法的客户端程序并进行了首次“采矿”(mining), 获得了第一批的 50 个比特币。这也标志着比特币金融体系的正式诞生。[19]

2010 年 12 月 5 日, 在维基解密泄露美国外交电报事件期间, 比特币社区呼吁维基解密接受比特币捐款以打破金融封锁。中本表示坚决反对, 认为比特币还在摇篮中, 经不起冲突和争议。七天后的 12 月 12 日, 他在比特币论坛中发表了最后一篇文章, 提及了最新版本软件中的一些小问题, 随后不再露面, 电子邮件通讯也逐渐终止。

从发表论文以来, 中本聪的真实身份长期不为外界所知, 维基解密创始人朱利安·阿桑奇 (Julian Assange) 宣称中本聪是一位密码朋克 (Cypherpunk)。另外, 有人称“中本聪是一名无政府主义者, 他的初衷并不希望数字加密货币被某国政府或中央银行控制, 而是希望其成为全球自由流动、不受政府监管和控制的货币。”

## 15.2 技术发展脉络

ACM Queue 在 2017 年发表了一篇文章 “Bitcoin’s Academic Pedigree”<sup>3</sup>, 此文章对比特币的技术发展脉络进行了梳理 (15.1)。并按一下组织方式进行了基本概念的说明:

1. ledger(账本)
  - (a). Linked timestamping(时间戳链)
  - (b). Merkle trees(默克尔树)
  - (c). Byzantine fault tolerance(拜占庭容错)
2. Proof of work(工作量证明)
  - (a). The origins(概念起源)
  - (b). hashcash(哈希现金)
  - (c). proof of work and digital cash:A catch-22(工作量证明和数字现金: 左右为难)

<sup>1</sup>此节文字来源于 <https://baike.baidu.com/item/>

<sup>2</sup>此文章的网页版<https://nakamotoinstitute.org/bitcoin/>, 此网页上也有此文章的中文链接。

<sup>3</sup>ACM Queue 文章不需付费, 此文章下载地址为<https://queue.acm.org/detail.cfm?id=3136559>, 在网上有这篇文章的中文译稿, 大家可以参考阅读, 地址为<https://blog.csdn.net/tangxiaoyin/article/details/80131400>

### 3. putting it all together(集成)

#### (a). public keys as identities(公钥作为身份)

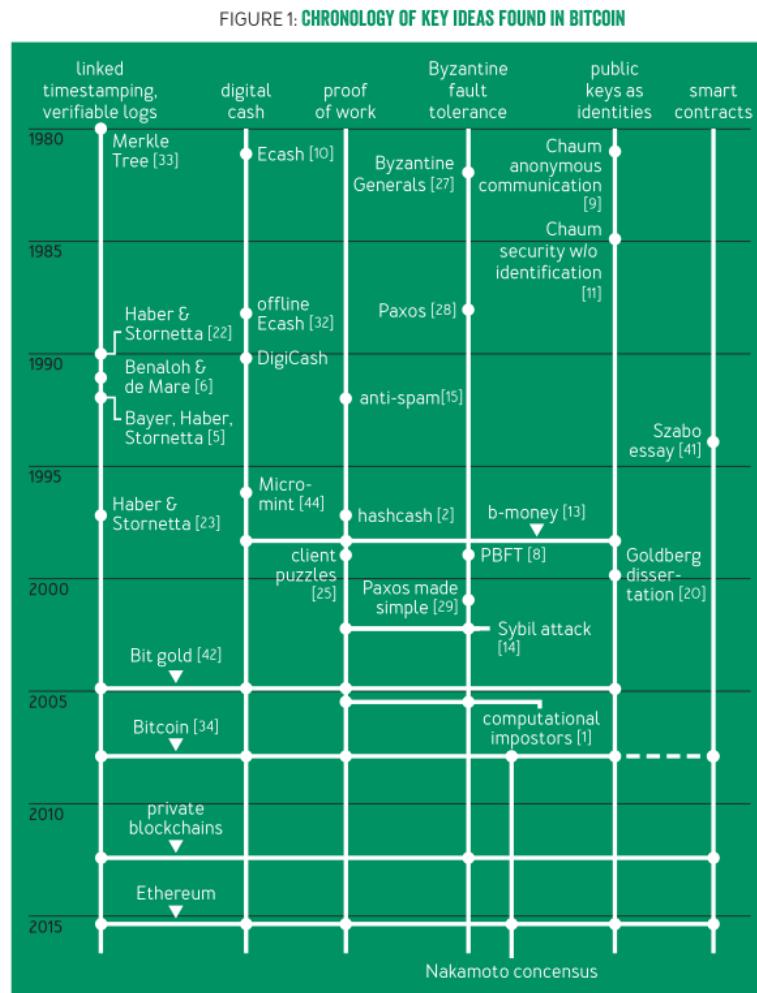


图 15.1: 比特币技术发展脉络

#### 15.2.1 账本

账本记录了比特币的交易，比特币系统是一个去中心化电子货币系统，所以每个人都拥有一个账本的拷贝，账本是一个区块链 (block chain)。

#### 15.2.2 交易 (Transaction)

在比特币系统中，电子货币 (electronic coin) 是一个签名链 (a chain of digital signatures)，每一位货币所有者，通过对“前一次交易和下一位拥有者公钥”哈希值进行数字签名，并且把这个签名附加在电子货币尾端，完成货币的转移。收款人通过验证签名就能够验证该链条的所有者。表示电子货币的签名链如图15.2所示。

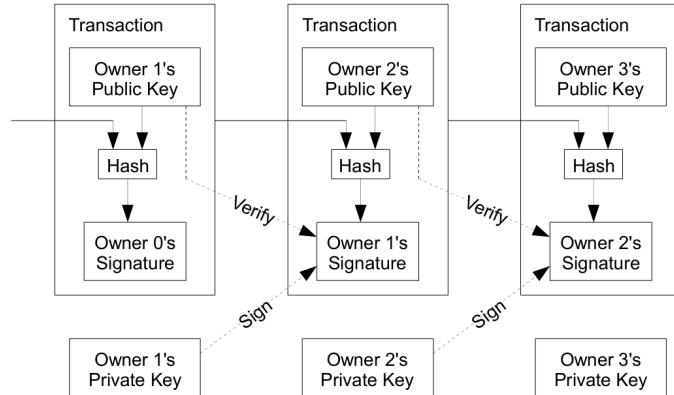


图 15.2: 电子币的签名链

### 15.2.3 拜占庭协议

由于比特币是一个去中心化的系统，但是账本信息又要保持一致，这其实是个不容易实现的问题，这是分布式系统里面的一个经典问题，同步问题或者共识问题。

这个问题通常用“拜占庭将军问题”来形象表述。

#### 15.2.3.1 拜占庭将军问题 (The Byzantine Generals Problem)

拜占庭将军问题 (The Byzantine Generals Problem)，是由 LESLIE LAMPORT, ROBERT SHOSTAK 和 MARSHALL PEASE 在 1982 年的 ACM Transactions on Programming Languages and Systems 上发表的文章 “The Byzantine Generals Problem” 中提出的，文章中称“可靠的计算机系统必须能够应对一个或多个故障组件，失败的组件可能会向不同的部分发送冲突信息，系统应对这个问题抽象地描述为拜占庭将军问题。”，当然这个故事是虚构出来的。

我们看看网络上一个帖子<sup>4</sup>基于原论文对此问题的描述。

拜占庭帝国 (*Byzantine Empire*) 军队的几个师驻扎在敌城外，每个师都由各自的将军指挥。将军们只能通过信使相互沟通。在观察敌情之后，他们必须制定一个共同的行动计划，如进攻 (*Attack*) 或者撤退 (*Retreat*)，且只有当半数以上的将军共同发起进攻时才能取得胜利。然而，其中一些将军可能是叛徒，试图阻止忠诚的将军达成一致的行动计划。更糟糕的是，负责消息传递的信使也可能是叛徒，他们可能篡改或伪造消息，也可能使得消息丢失。

为了更加深入的理解拜占庭将军问题，我们以三将军问题为例进行说明。

当三个将军都忠诚时，可以通过投票确定一致的行动方案，图 15.3 展示了此种场景，即 *General A, B* 通过观察敌军军情并结合自身情况判断可以发起攻击，而 *General C* 通过观察敌军军情并结合自身情况判断应当撤退。最终三个将军经过投票表决得到结果为进攻：撤退 = 2:1，所以将一同发起进攻取得胜利。对于三个将军，每个将军都能执行两

<sup>4</sup>一文读懂拜占庭将军问题，链接<https://www.cnblogs.com/aspirant/p/13321816.html>

种决策(进攻或撤退)的情况下,共存在6中不同的场景,图15.3展示的是其中一种,对于其他5中场景可简单地推得,通过投票三个将军都将达成一致的行动计划。

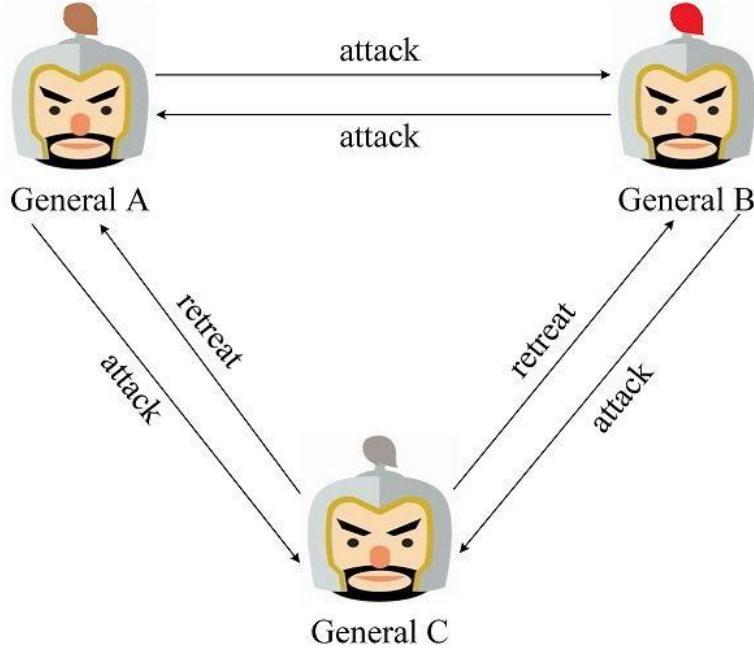


图 15.3: 三个将军都忠诚的情况

当三个将军中存在一个叛徒时,将可能扰乱正常的作战计划。图15.4展示了General C为叛徒的一种场景,他给General A和General B发送了不同的消息,在这种场景下General A通过投票最终得出撤退的结论;General B通过投票最终得出进攻的结论。General C是叛将,他不会根据投票结果行事,他会根据是否对己方有力,做出进攻或撤退的决断,比如C撤退,结果只有General B发起了进攻并战败。

事实上,对于三个将军中存在一个叛徒的场景,想要总能达到一致的行动方案是不可能的。详细的证明可参看Leslie Lamport的论文。此外,论文中给出了一个更加普适的结论:如果存在 $m$ 个叛将,那么至少需要 $3m+1$ 个将军,才能最终达到一致的行动方案。

Leslie Lamport在论文中给出了两种拜占庭将军问题的解决方案,即口信消息型解决方案(*A solution with oral message*)和签名消息型解决方案(*A solution with signed message*)。

### 1、口信消息型解决方案

首先,对于口信消息(*Oral message*)的定义如下:

- A1. 任何已经发送的消息都将被正确传达;
- A2. 消息的接收者知道是谁发送了消息;
- A3. 消息的缺席可以被检测。

基于口信消息的定义,我们可以知,口信消息不能被篡改但是可以被伪造。基于对图15.4场景的推导,我们知道存在一个叛将时,必须再增加3个忠将才能达到最终的行动一致。为加深理解,我们将利用3个忠将1个叛将的场景对口信消息型解决方案进行推导。在口信消息型解决方案中,首先发送消息的将军称为指挥官,其余将军称为副官。对于3忠1叛的场景需要进行两轮作战信息协商,如果没有收到作战信息那么默认撤退。

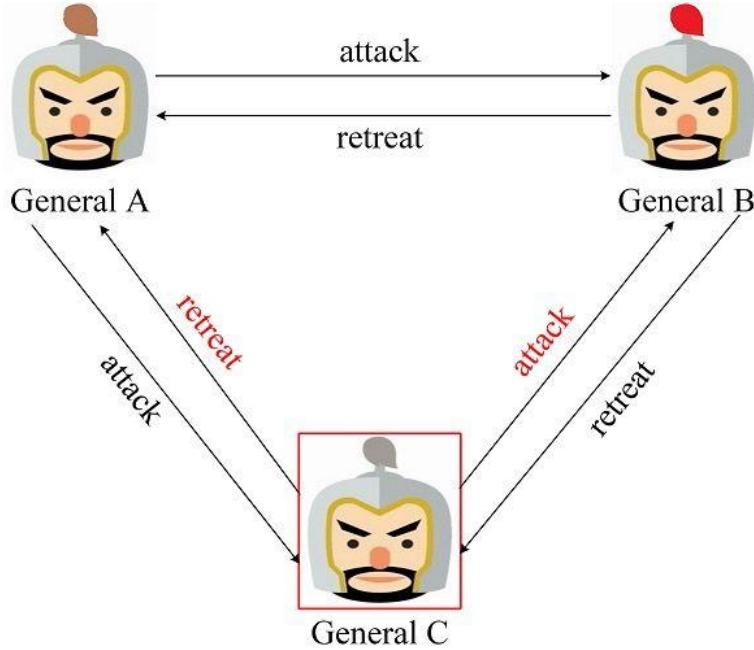


图 15.4: 三个将军中两个忠诚, 一个叛将的情况

图 15.5 是指挥官为忠将的场景, 在第一轮作战信息协商中, 指挥官向 3 位副官发送了进攻的消息; 在第二轮中, 三位副官再次进行作战信息协商, 由于 General A、B 为忠将, 因此他们根据指挥官的消息向另外两位副官发送了进攻的消息, 而 General C 为叛将, 为了扰乱作战计划, 他向另外两位副官发送了撤退的消息。最终 Commanding General, General A 和 B 达成了一致的进攻计划, 可以取得胜利。

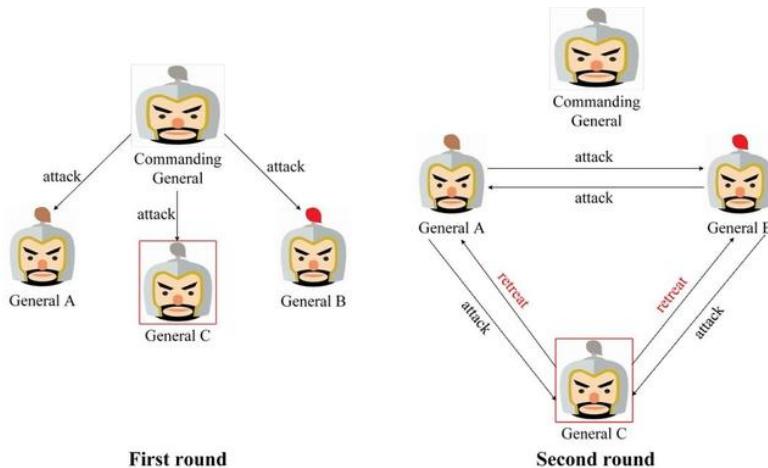


图 15.5: 指挥官为忠将的情况

图 15.6 是指挥官为叛将的场景, 在第一轮作战信息协商中, 指挥官向 General A、B 发送了撤退的消息, 但是为了扰乱 General C 的决定向其发送了进攻的消息。在第二轮中, 由于所有副官均为忠将, 因此都将来自指挥官的消息正确地发送给其余两位副官。最终所有忠将都能达成一致撤退的计划。

如上所述, 对于口信消息型拜占庭将军问题, 如果叛将人数为  $m$ , 将军人数不少于  $3m+1$ , 那么最终能达成一致的行动计划。值的注意的是, 在这个算法中, 叛将人数  $m$  是

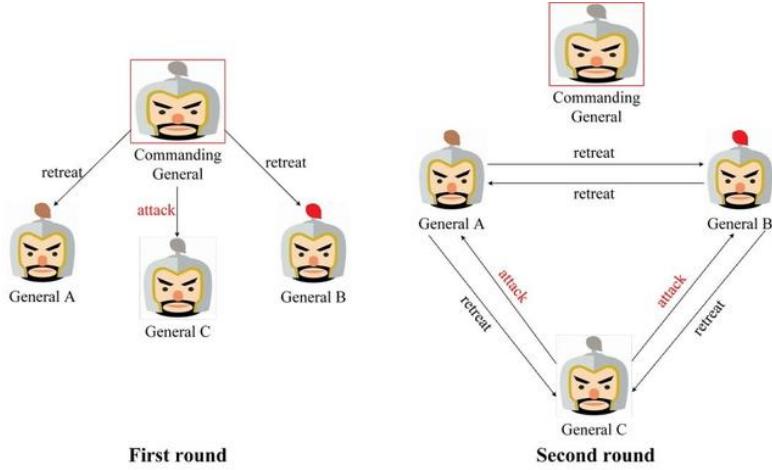


图 15.6: 指挥官为忠将的情况

已知的，且叛将人数  $m$  决定了递归的次数，即叛将数  $m$  决定了进行作战信息协商的轮数，如果存在  $m$  个叛将，则需要进行  $m+1$  轮作战信息协商。这也是上述存在 1 个叛将时需要进行两轮作战信息协商的原因。

## 2、签名消息型解决方案

同样，对签名消息的定义是在口信消息定义的基础上增加了如下两条：

- A4. 忠诚将军的签名无法伪造，而且对他签名消息的内容进行任何更改都会被发现；
- A5. 任何人都能验证将军签名的真伪。

基于签名消息的定义，我们可以知道，签名消息无法被伪造或者篡改。为了深入理解签名消息型解决方案，我们同样以 3 三将军问题为例进行推导。图 15.7 是忠将率先发起作战协商的场景，General A 率先向 General B、C 发送了进攻消息，一旦叛将 General C 篡改了来自 General A 的消息，那么 General B 将将发现作战信息被 General C 篡改，General B 将执行 General A 发送的消息。

图 15.8 是叛将率先发起作战协商的场景，叛将 General C 率先发送了误导的作战信息，那么 General A、B 将发现 General C 发送的作战信息不一致，因此判定其为叛将。可对其进行处理后再进行作战信息协商。签名消息型解决方案可以处理任何数量叛将的场景。

拜占庭将军问题是为了解释计算机分布式处理中的问题而构造出来的故事，但是其很形象的描述了这个问题。将军，对应计算机节点；忠诚的将军，对应运行良好的计算机节点；叛变的将军，被非法控制的计算机节点；信使被杀，通信故障使得消息丢失；信使被间谍替换，通信被攻击，攻击者篡改或伪造信息。如上文所述，拜占庭将军问题提供了对分布式共识问题的一种情景化描述，是分布式系统领域最复杂的模型。此外，它也为我们理解和分类现有的众多分布式一致性协议和算法提供了框架。现有的分布式一致性协议和算法主要可分为两类：

一类是故障容错算法 (*Crash Fault Tolerance, CFT*)，即非拜占庭容错算法，解决的是分布式系统中存在故障，但不存在恶意攻击的场景下的共识问题。也就是说，在该场景下可能存在消息丢失，消息重复，但不存在消息被篡改或伪造的场景。一般用于局域网

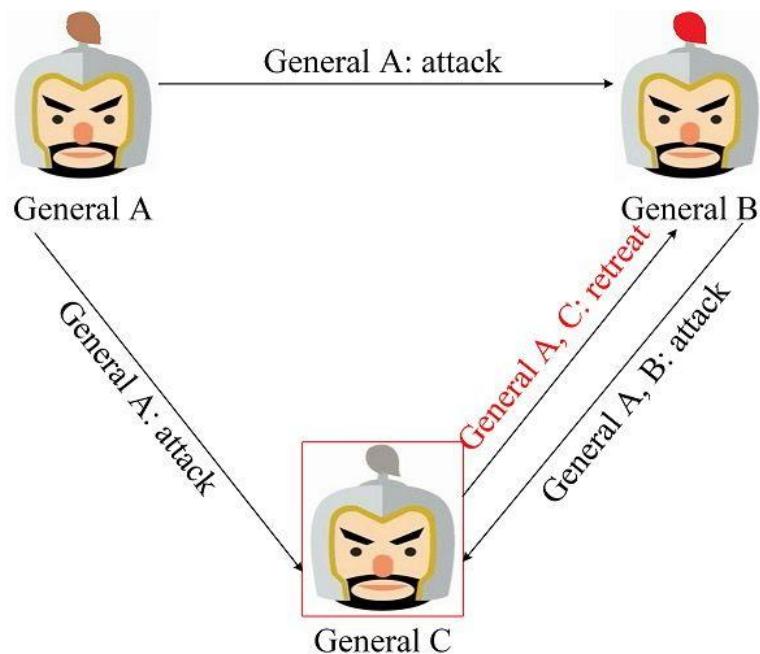


图 15.7: 指挥官为忠将的情况

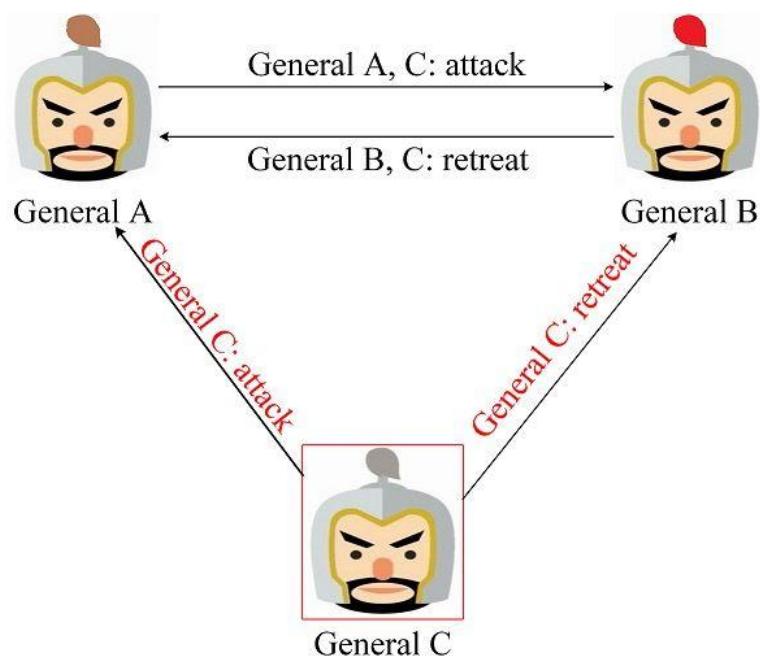


图 15.8: 指挥官为叛将的情况

场景下的分布式系统，如分布式数据库。属于此类的常见算法有 *Paxos* 算法<sup>5</sup>、*Raft* 算法<sup>67</sup>、*ZAB*(*Zookeeper Atomic Broadcast*) 协议<sup>89</sup>等。

一类是拜占庭容错(*Byzantine Fault Tolerance*)算法，可以解决分布式系统中既存在故障，又存在恶意攻击场景下的共识问题。一般用于互联网场景下的分布式系统，如在数字货币的区块链技术中。属于此类的常见算法有 *PBFT*(*Practical Byzantine Fault Tolerance*) 算法、*PoW*(*Proof-of-Work*) 算法<sup>10</sup>。

### 15.2.3.2 拜占庭共识算法之 PBFT

*PBFT* 是 *Practical Byzantine Fault Tolerance* 的缩写，意为实用拜占庭容错算法。该算法是 Miguel Castro 和 Barbara Liskov 在 1999 年的“操作系统设计与实现国际会议”(OSDI99) 上的文章“*Practical Byzantine Fault Tolerance*”中提出来的，解决了原始拜占庭容错算法效率不高的问题，将算法复杂度由指数级降低到多项式级，使得拜占庭容错算法在实际系统应用中变得可行。

### 15.2.4 挖矿原理<sup>11</sup>

比特币每个区块的数据结构由区块头和区块体两部分组成。区块体中包含了矿工搜集的若干交易信息，图15.9中假设有 8 个交易被收录在区块中，所有的交易生成一颗默克尔树，默克尔树是一种数据结构，它将叶子节点两两哈希，生成上一层节点，上层节点再哈希，生成上一层，直到最后生成一个树根，称之为默克尔树根，只有树根保留在区块头中，这样可以节省区块头的空间，也便于交易的验证。区块头中包含父区块的哈希，版本号，当前时间戳，难度值，随机数和上面提到的默克尔树根。

假设区块链已经链接到了某个块，有 ABCD 四个节点已经搜集了前十分钟内全网中的一些交易信息，他们选出其中约 4k 条交易，打包好，生成默克尔树根，将区块头中的信息，即父区块哈希 + 版本号 + 时间戳 + 难度值 + 随机数 + 默克尔树根组成一个字符串 str，通过两次哈希函数得出一个 256 的二进制数，即  $\text{SHA256}(\text{SHA256}(\text{str})) = 10010011 \dots \dots$  共 256 位，比特币要求，生成的结果，前 n 位必须是 0，n 就是难度值，如果现在生

<sup>5</sup>Leslie Lamport. 1998. The part-time parliament. ACM Trans. Comput. Syst. 16, 2 (May 1998), 133–169. DOI:<https://doi.org/10.1145/279227.279229>

<sup>6</sup>Heidi Howard,ARC: Analysis of Raft Consensus,Technical Report,University of Cambridge, Computer Laboratory,<https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-857.pdf>

<sup>7</sup><https://raft.github.io/>

<sup>8</sup>F. P. Junqueira, B. C. Reed and M. Serafini, "Zab: High-performance broadcast for primary-backup systems," 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), 2011, pp. 245-256, doi: 10.1109/DSN.2011.5958223.

<sup>9</sup>F. Junqueira, B. Reed and M. Serafini, "Dissecting Zab" in Tech. Rep. YL-2010-007 12, Sunnyvale, CA, USA:Yahoo!Research,2010.<https://cwiki.apache.org/confluence/download/attachments/24193444/yl-2010-007.pdf?version=1&modificationDate=1468443609000&api=v2>

<sup>10</sup>PoW 思想是 1993 年 Cynthia Dwork 和 Moni Naor 文章“Pricing via Processing or Combatting Junk Mail”中首先被提出，首次以 PoW 名字被提出，出现在文章 Jakobsson, M. and A. Juels. “Proofs of Work and Bread Pudding Protocols.” Communications and Multimedia Security (1999).

<sup>11</sup>比特币原理详解，链接地址[https://blog.csdn.net/zcg\\_741454897/article/details/102796022](https://blog.csdn.net/zcg_741454897/article/details/102796022)

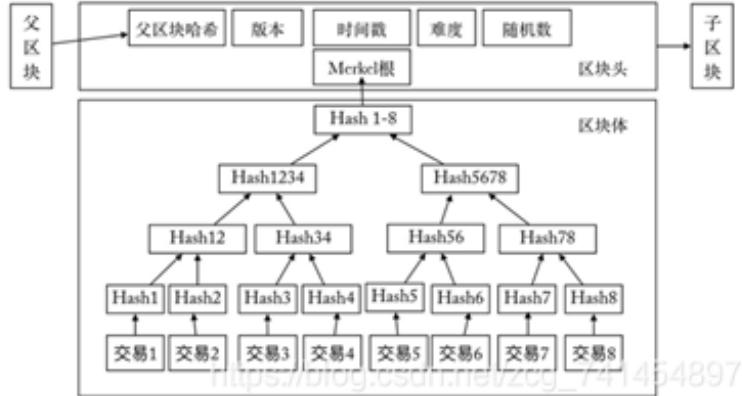


图 15.9: 比特币区块数据

成的二进制数不符合要求，就必须改变随机数的值，重新计算，只到算出满足条件的结果为止。假设现在  $n$  是 5，则生成的二进制数必须是 00000……(共 256 位)。一旦挖矿成功，矿工就可以广播这个消息到全网，其他的矿工就会基于该区块继续挖矿。下一个区块头中的父区块哈希值就是上一个区块生成的 00000……这个数。

解决这个数学难题要靠运气，理论上，运气最好的矿工可能 1 次哈希就能算出结果，运气差的可能永远都算不出来。但是总体来看，如果一个矿工的算力越大，单位时间内进行的哈希次数就越多，就越可能在短时间内挖矿成功。

那么  $n$  是如何确定的呢？比特币设计者希望，总体上平均每十分钟产生一个区块，总体上来看，挖矿成功的概率为  $\frac{1}{2^n}$ 。现假设世界上有 1W 台矿机，每台矿机的算力是  $14T\text{次}/s = 1.4 \times 10^{13}\text{次}/s^{12}$ ，单位次/s 称之为哈希率，10 分钟是 600s，所以 1W 台矿机 10 分钟可以做  $1.4 \times 10^{13} \times 6 \times 10^2 \times 10^4 = 8.4 \times 10^{19}$  次哈希运算，从概率角度看，想要挖矿成功需要做  $2^n$  次运算，可以列出等式  $2^n = 8.4 \times 10^{19}$ ，可以解出  $n$  约为 66，这个意思就是说，1W 台矿机，在 10 分钟内，可以挖矿成功的难度为 66。所以对于这种方法，我们没有办法使得自己的运气变的更好，只能提高自己的算力，尽快的算出结果。

"当一个矿工算出了合适的随机数，并且没有收到其他矿工算出来的通知时，他就认为自己首先找到了随机数（当然有时候可能是网络延迟他没有收到通知）。于是他要做的就是向全网广播，告诉大家他算出的随机数以及他这个包的数据。其他矿工收到这个消息后，如果这个时间段没有更早的消息，他们会验证这个随机数，以及这个包里的数据是否合法。验证通过的话，矿工们会接受这就是这个时间段的最终挖出区块，他们会把这个区块添加到自己的账本的末端，跟原来的账本 chain 在一起，并且停掉之前的计算随机数的进程，把自己那个包里包含的但未包含在新区块里的交易数据重新放到待打包交易池中，然后进行下一轮的挖矿。<sup>13</sup>"

<sup>12</sup>1 kH / s = 每秒 1,000 哈希

1 MH / s = 每秒 1,000,000 次哈希

1 GH / s = 每秒 1,000,000,000 次哈希

1 TH / s = 每秒 1,000,000,000,000 次哈希

1 PH / s = 每秒 1,000,000,000,000,000 次哈希

1 EH / s = 每秒 1,000,000,000,000,000,000 次哈希

<sup>13</sup>完整讲清比特币（三）：挖矿和记账，链接地址<https://zhuanlan.zhihu.com/p/34413557>

### 15.2.5 钱包

比特币的底层技术是公钥体系，比特币钱包就是包含私钥和地址，比特币的公钥体系用的是椭圆曲线密码 (ECC)，ECC 的公私钥生成可以查看相关文献，地址是这个钱包的唯一标识，我们简单介绍一下地址的生成<sup>14</sup>：

1. 计算公钥摘要。这里公钥既可以是完整版，也可以是压缩版，我们选择压缩版。有了公钥之后，对公钥进行两次哈希运算，第一次通过 SHA-256 算法得到运算结果后，对结果再进行一次 RIPEMD-160 运算，最终得到的结果就是所谓的加密版的公钥了，比如：

453233600a96384bb8d73d400984117ac84d7e8b。

2. 对加密版公钥添加网络标识字节。比特币一共有两个网络：主网和测试网。如果我们需要生成一个主网地址，就要在加密版公钥开头添加 0x00，比如：

00453233600a96384bb8d73d400984117ac84d7e8b。

3. 添加校验值。校验值是通过对第二步得到的结果运行两次 SHA-256 哈希运算，然后取最终哈希值的前四个字节得到的，例如：表示成十六进制就是 512f43c4。把这个校验值添加到第二步结果的末尾，得到的就是钱包地址了，比如：

00453233600a96384bb8d73d400984117ac84d7e8b512f43c4

有了校验值，钱包软件就很容易帮我们判定地址有没有填错或者损坏了。但是，很多时候我们看到的钱包地址不是用十六进制表示的，而是用 Base58<sup>15</sup> 格式，所以最终钱包地址看起来是这个样子：

17JsmEygbEUEpvt4PFtYaTeSqfb9ki1F1。

关于较详细的介绍，可以查看 Bitcoin WIKI (链接为：[https://en.bitcoin.it/wiki/Invoice\\_address](https://en.bitcoin.it/wiki/Invoice_address)).

## 15.3 比特币

从比特币的本质说起，比特币的本质其实就是一个复杂算法所生成的特解。特解是指方程组所能得到有限个解中的一组。而每一个特解都能解开方程并且是唯一的。以钞票来比喻的话，比特币就是钞票的冠字号码，你知道了某张钞票上的冠字号码，你就拥有了这张钞票。而挖矿的过程就是通过庞大的计算量不断的去寻求这个方程组的特解，这个方程组被设计成了只有 2100 万个特解，所以比特币的上限就是 2100 万个。<sup>[5]</sup> 要挖掘比特币可以下载专用的比特币运算工具，然后注册各种合作网站，把注册来的用户名和密码填入计算程序中，再点击运算就正式开始。完成 Bitcoin 客户端安装后，可以直接获得一个 Bitcoin 地址，当别人付钱的时候，只需要自己把地址贴给别人，就能通过同样的客户端进行付款。在安装好比特币客户端后，它将会分配一个私钥和一个公钥。需要备

<sup>14</sup>如何生成比特币地址和私钥？，链接地址<https://zhuanlan.zhihu.com/p/54296648>

<sup>15</sup>Base58 编码的作用是将非可视字符可视化，或者说 ASCII 化。与 Base64 不同的是 base58 编码去掉了几个看起来会产生歧义的字符，如 0(零), O(大写字母 O), I(大写的字母 i) and l(小写的字母 L)，和几个影响双击选择的字符，如 /, +。结果字符集正好 58 个字符(包括 9 个数字，24 个大写字母，25 个小写字母)。

份你包含私钥的钱包数据，才能保证财产不丢失。如果不完全格式化硬盘，个人的比特币将会完全丢失。<sup>16</sup>

可以说比特币的整个实现就是建立在已有的甚至存在多年的计算机科学领域里的技术或概念的整合，其中哈希算法在比特币中的应用几乎是方方面面，主要包括 SHA256 和 RIPEMD160，比特币将这两个哈希算法的应用组合成两个函数：hash256(d)=sha256(sha256(d)) 和 hash160(d)=ripemd160(sha256(d))，其中 d 为待哈希的字节数组，两者分别生成 256 位（32 字节）和 160 位（20 字节）的 16 进制数值。hash256 主要用于生成标志符，如区块 ID，交易 ID 等，而 hash160 主要用于生成比特币地址。

对于 hash160 比较认同的答案是 ripemd160 可以使得生成的地址更短，但是只做 ripemd160 一次哈希可能会存在安全漏洞所以同时使用 sha256 起到安全加固；至于 hash256 使用两次 sha256 哈希算法的原因来源于 sha1 算法，由于一次 sha1 哈希存在被生日攻击（birthday attack）的风险，所以当使用 sha1 运算时一种有效方式就是做两次 sha1 哈希，sha256 本身并不存在生日攻击漏洞，但是防御性的使用两次 sha256 哈希借鉴于 sha1。<sup>17</sup>

## 15.4 其他数字货币或虚拟货币

比特币是中本聪在 2009 年设计开发，以开源的方式发布，并在此软件的基础上构建 P2P 网络，形成的一种 P2P 形式的虚拟货币。

后来又有很多采用相同思想发布的虚拟货币，下面两张图（图15.10, 图15.11）是用搜索引擎获得两个页面，以此来说明现在已经出现了很多类似于比特币的虚拟货币，严格意义上讲，比特币应该是指中本聪发起的虚拟货币，但是现在很多场合比特币成为了虚拟货币或者数字货币的代名词，但这种替换应该以不引起歧义为前提。

2017年虚拟货币资料

货币	符号	发行时间	创始人	活跃	市值	比特币基础	算法
比特币	BTC	2009	中本聪	是	2000亿美元	是	SHA-256
以太币	ETH	2014	维塔利克·布特林	是	320亿美元	否	Ethash
瑞波币	XRP	2013	克里斯·拉森	是	170亿美元	是	SHA-256
柚子币	EOS	2017	丹尼尔·拉里默	是	55亿美元	否	DPOS
莱特币	LTC	2011	李启威	是	75亿美元	是	Scrypt
比特币现金	BCH	2017	吴忌寒	是	75亿美元	是	SHA-256

图 15.10：百度百科上的数字货币<sup>18</sup>

## 15.5 区块链 (block chains)

通常人们都认为区块链是和比特币的概念一起提出的，但是阅读中本聪的原始论文你会发现，论文中根本没有提及“区块链”这个概念，而是后来大家用来指代与比特币及其账本类似系统的一个宽泛术语。

<sup>16</sup><https://baike.baidu.com/item/>

<sup>17</sup><https://zhuanlan.zhihu.com/p/31961153>

The screenshot shows a table of cryptocurrency data from the BTC123 website. The columns include: #, 币种 (Coin), 总发行量 (Total Supply), 市值 (Market Cap), 价格 (Price), 24H成交额 (24H Trading Volume), 24H涨跌幅 (24H Price Change), and 交易所 (Exchange). The data is sorted by market cap. Some rows are partially cut off.

#	币种	总发行量	市值	价格	24H成交额	24H涨跌幅	交易所
1	BTC—Bitcoin (比特币)	21000000	¥11,989,8119亿	¥65,371,7429	¥287,3381亿	-1.22%	22
2	ETH—以太坊	104599798	¥1,805,3788亿	¥1,630,0388	¥99,2195亿	-2.78%	22
3	XRP—瑞波币	10000000000	¥668,6729亿	¥1,5227	¥13,9397亿	-0.40%	16
4	USDT—泰达币	200000000	¥637,7139亿	¥7,1286	¥2,8704亿	0.08%	8
5	BCH—Bitcoin Cash (比特币)	21000000	¥330,4632亿	¥1,800,7992	¥12,2327亿	-2.26%	11
6	LTC—Litecoin (莱特币)	84000000	¥231,3339亿	¥357,7494	¥12,2251亿	-2.02%	16
7	EOS—柚子	1006245120	¥202,0492亿	¥21,6379	¥17,4904亿	-1.34%	16
8	BNB—Binance Coin	190799015	¥181,4167亿	¥123,3247	¥4,9568亿	-3.92%	2
9	XTZ—Tezos	763306930	¥142,6881亿	¥20,0218	¥3,9224亿	-0.50%	7
10	ADA—艾达币	31112483745	¥134,1289亿	¥0,5171	¥22,2559亿	2.82%	7
11	OKB—OKB 全球通用积分	100000000	¥104,3665亿	¥36,9600	¥7,1540亿	-2.01%	2
12	LINK—ChainLink	100000000	¥103,3409亿	¥29,5170	¥5,2012亿	3.31%	7

图 15.11: BTC123 网站上列出的数字货币 (部分)<sup>19</sup>

我国于 2016 年 10 月 18 号，在北京成立“中国区块链技术和产业发展论坛”<sup>20</sup>，论坛由中国电子技术标准化研究院、蚂蚁金服、万向、微众银行、平安集团、乐视联服、万达网络、用友、三一集团、海航科技等国内从事区块链的重点企事业单位构成。该论坛一个主要工作是编写区块链的相关标准，其于 2017 年 5 月 16 日，发布首个区块链标准《区块链参考架构》，其对区块链的概念定义为：

1. 区块链 (blockchain): 一种在对等网络环境下，通过透明和可信规则，构建不可伪造、不可篡改和可追溯的块链式数据结构，实现和管理事务处理的模式。(注：事务处理包括但不限于可信数据的产生、存取和使用等。)
2. 对等网络 (peer-to-peer network): 一种仅包含对控制和操作能力等效的节点的计算机网络。[GB/T 5271.18-2008]
3. 块链式数据结构 (chained-block data structure): 一段时间内发生的事务处理以区块为单位进行存储，并以密码学算法将区块按时间顺序连接成链条的一种数据结构。

NIST 对 Blockchain 的定义或者说简要描述为<sup>21</sup>:

Blockchain represents a new paradigm for digital interactions and serves as the underlying technology for most cryptocurrencies.

A blockchain is a collaborative, tamper-resistant ledger that maintains transactional records. The transactional records (data) are grouped into blocks. A block is connected to the previous one by including a unique identifier that is based on the previous block's data. As a result, if the data is changed in one block, its unique identifier changes, which can be seen in every subsequent block (providing tamper evidence). This domino effect allows all users within the blockchain to know if a previous block's data has been tampered with. Since a blockchain network is difficult to alter or destroy, it provides a resilient method of collaborative record keeping.

NIST researchers have been investigating blockchain technologies at multiple levels: from use cases, applications and existing services, to protocols, security guarantees, and cryptographic

<sup>20</sup>论坛网址<http://www.cbdforum.cn/bcweb/>

<sup>21</sup>此信息来源于<https://www.nist.gov/topics/blockchain>

mechanisms. Research outcomes include scientific papers and the production of software for experimentation as well as providing direction for other NIST endeavors in this space. Blockchain has the potential to be implemented in many different systems, to include manufacturing supply chains, data registries, digital identification, and records management.

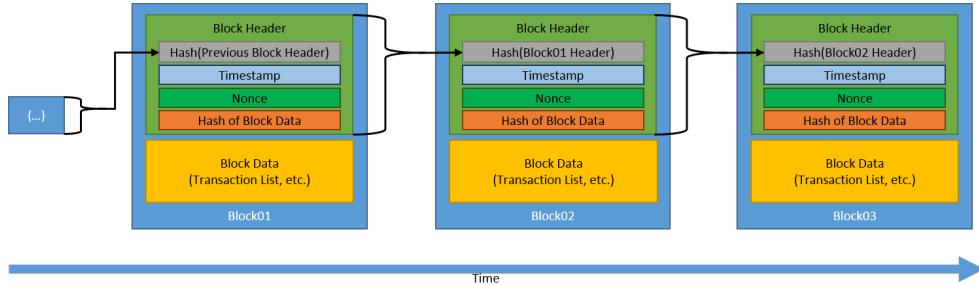


图 15.12: NIST 区块链示意图

虽然区块链的概念产生晚于比特币，但是由于其应用场景更加宽泛，所以比特币反而成了区块链的一种应用，或者是区块链技术在特定场景下的具体应用。

### 15.5.1 区块链的几种类型

区块链按照准入机制，可以分为公有链（Public Blockchain）、私有链（Private blockchain）和联盟链（Consortium Blockchain）三类。

*A public blockchain is permissionless. Anyone can join the network and read, write, or participate within the blockchain. A public blockchain is decentralized and does not have a single entity which controls the network. Data on a public blockchain are secure as it is not possible to modify or alter data once they have been validated on the blockchain. Bitcoin and Ethereum are well-known examples of a public blockchain.<sup>22</sup>*

*A private blockchain is a permissioned blockchain. Private blockchains work based on access controls which restrict the people who can participate in the network. There are one or more entities which control the network and this leads to reliance on third-parties to transact. In a private blockchain, only the entities participating in a transaction will have knowledge about it, whereas the others will not be able to access it. Hyperledger Fabric of Linux Foundation is a perfect example of a private blockchain.<sup>23</sup>*

*Consortium blockchains do not allow any person with an internet connection to participate*

<sup>22</sup>PUBLIC VS. PRIVATE BLOCKCHAIN : A COMPREHENSIVE COMPARISON,web link:<https://www.blockchain-council.org/blockchain/public-vs-private-blockchain-a-comprehensive-comparison/>

<sup>23</sup>PUBLIC VS. PRIVATE BLOCKCHAIN : A COMPREHENSIVE COMPARISON,web link:<https://www.blockchain-council.org/blockchain/public-vs-private-blockchain-a-comprehensive-comparison/>

*nor do they grant full control to a single entity but rather a group of approved individuals. They therefore provide the efficiency and security of public blockchains while still allowing for some degree of central control, monitoring and safeguarding. Consortium blockchains are most often the banking sector, for instance Quorum is an Ethereum-powered consortium blockchain created by JP Morgan to service the needs of financial industries and beyond. Another example, Hyperledger, open source collaborative effort that unites finance, banking, Internet of Things, supply chains, manufacturing and Technology on a consortium blockchain.*<sup>24</sup>

不同类型的区块链其节点之间的信任程度不同，所以其采用了不同的共识算法，公链中的节点可以很自由的加入或者退出，不需要严格的验证和审核，所以公链不仅需要考虑网络中存在故障节点，还需要考虑作恶节点，公链使用的共识算法有 *PoW*、*POS*、*DPOS*、*Ripple* 等。私链的适用环境一般是不考虑集群中存在作恶节点，只考虑因为系统或者网络原因导致的故障节点，私链的共识算法有 *Paxos*、*Raft*。联盟链中每个新加入的节点都是需要验证和审核的，联盟链的适用环境除了需要考虑集群中存在故障节点，还需要考虑集群中存在作恶节点，联盟链的共识算法有 *PBFT*、*DBFT*。<sup>25</sup>

有时我们还能看到一个概念，侧链（Side Chain），侧链不是一种区块链类型，而是一种协议“这个协议具体是：可以让比特币安全地从比特币主链转移到其他区块链，又可以从其他区块链安全地返回比特币主链的一种协议。侧链技术为什么会出现？简单来讲，在比特币、以太坊等公链上做创新或拓展是比较困难的。同时，公链每秒处理交易笔数有限，比如，以太坊 25tps，比特币 7tps，并且在交易用户过多时会发生拥堵，甚至瘫痪。这时，侧链技术应运而生。侧链就像是一条条通路，将不同的区块链互相连接在一起，以实现区块链的扩展。公链本身是一本分布式账本，侧链是独立于公链的另一本分布式账本。但是这两个账本之间能够“互相操作”，实现交互。”<sup>26</sup>

## 15.6 开源系统

Bitcoincore(<https://bitcoincore.org/>) 是一个开源比特币软件，网站简单介绍为“Bitcoin Core is an open source project which maintains and releases Bitcoin client software called “Bitcoin Core”. It is a direct descendant of the original Bitcoin software client released by Satoshi Nakamoto after he published the famous Bitcoin whitepaper. Bitcoin Core consists of both “full-node” software for fully validating the blockchain as well as a bitcoin wallet. The project also currently maintains related software such as the cryptography library libsecp256k1 and others located at GitHub. Anyone can contribute to Bitcoin

<sup>24</sup>What Are Consortium Blockchains?, web link: <https://blockchainlabs.asia/news/what-are-consortium-blockchains/>

<sup>25</sup>深入剖析区块链的共识算法 Raft & PBFT, 网络链接<https://www.cnblogs.com/davidwang456/articles/9001331.html>

<sup>26</sup>公有链、联盟链、私有链，一文让你读懂“眼花缭乱”的各种链, 文章链接[https://www.sohu.com/a/250564024\\_99949057](https://www.sohu.com/a/250564024_99949057)

Core.", Github 上的仓库地址为<https://github.com/bitcoin/bitcoin>, Gitee 上的镜像地址为<https://gitee.com/mirrors/bitcoin>。

还有一些其他的开源项目比如加密钱包 (例如 Tor-Crypto-Wallet)、交易平台 (比如 CCXT -CryptoCurrency eXchange Trading Library、Gryphon) 等。

RChain 是一个开源的区块链项目,Github 上仓库地址为<https://github.com/rchain/rchain>, RChain 项目的 README 中的简介为"The open-source RChain project is building a decentralized, economic, censorship-resistant, public compute infrastructure and blockchain. It will host and execute programs popularly referred to as “smart contracts”. It will be trustworthy, scalable, concurrent, with proof-of-stake consensus and content delivery."。

# 第 16 章 可信计算 (Trusted Computing)

---

冯登国老师在 2013 年的出版的专著“可信计算——理论与实践”[20] 中对可信计算进行了全面深入的介绍，想对可信计算进行全面了解的，可以阅读此书。下面我们引用其“绪论”中的一段话，来介绍可信计算的基本背景。

随着云计算、物联网和移动互联网等新型技术的快速发展，新型技术已经深刻影响到社会的管理方式和人们的生活方式，无处不在的信息已经成为国家、企业和个人的重要资产。随着病毒和恶意软件等的泛滥，黑客攻击技术和能力的增强，这些重要信息资产将暴露在越来越多的威胁中。毫无疑问，提供一个可信赖的计算环境，保障信息的机密性、完整性、真实性和可靠性，已经成为国家、企业和个人优先考虑的安全需求。传统的防火墙、入侵检测和病毒防御等网络安全防护手段都侧重于保护服务器的信息安全，而想对脆弱的终端就越来越成为信息系统安全的主要薄弱环节。针对这些系统安全需求和各类攻击手段，可信计算从计算机体系结构着手，从硬件安全出发建立一种信任传递体系以保证终端的可信，从源头上解决人与程序、人与机器以及人与人之间的信任问题。

可信计算就是在这种背景下应运而生的。对于“可信”这一概念目前有着众多不同的理解，为明确可信的含义，ISO/IEC、IEEE 和 TCG(*Trusted Computing Group*) 等组织都给出了可信的准确定义。TCG 组织在其可行理念下提出了通过嵌入在硬件平台上的可信平台模块 (*Trusted Platform Module*, TPM) 来提高计算机系统安全性的技术思路，得到了产业界的普遍认同。我们的思路与 TCG 类似，认为可信是指以安全芯片为基础建立可信的计算环境，确保系统实体按照预计的行为执行。

Stanford 在介绍 Trusted Computing 的网页<sup>1</sup>上有一句高度概括的话，对理解 TC 很有帮助：

*If you take one thing away from this slide, memorize this: Trusted Computing allows a piece of data to dictate what Operating System and Application must be used to open it.*

*In current situations, given a piece of data, you can choose what OS and what application you use to open it (though some do not make practical sense). Under Trusted Computing, the data dictates what Operating System and Application must be used to access it.*

在可信计算发展过程中，2003 微软将可信计算的总结为四个特征<sup>2</sup>：

<sup>1</sup>What is Trusted Computing, <https://cs.stanford.edu/people/eroberts/cs201/projects/trusted-computing/what.html>

<sup>2</sup>Trusted Computing: Promise and Risk, 文章链接<https://www.eff.org/wp/trusted-computing-promise-and-risk>

1. Memory curtaining(存储器屏蔽)
2. Secure input and output(安全输入和输出)
3. Sealed storage(密封存储)
4. Remote attestation(远程证明)

我们看看原文中的描述。

### **1. Memory curtaining**

*Memory curtaining refers to a strong, hardware-enforced memory isolation feature to prevent programs from being able to read or write one another's memory.*

### **2. Secure I/O**

*Secure input and output, or secure I/O, aims to address the threats posed by keyloggers and screen-grabbers, software used by snoops and intruders to spy on computer users' activities.*

### **3. Sealed storage**

*Sealed storage addresses a major PC security failing: the inability of a PC to securely store cryptographic keys.*

### **4. Remote attestation**

*Remote attestation is the most significant and the most revolutionary of the four major feature groups described by Microsoft. Broadly, it aims to allow "unauthorized" changes to software to be detected.*

## 16.1 可信平台模块 (TPM:Trusted Platform Module)

整个 TC 环境中，以可信平台模块 (Trusted Platform Module, 缩写为 TPM) 为核心，构建整个可信计算环境 (如图16.1所示)。

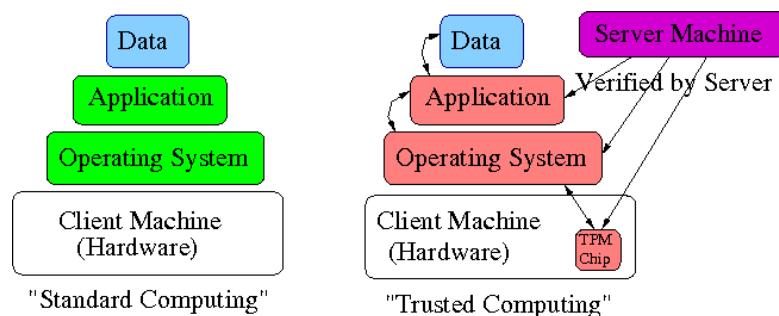


图 16.1: 以 TPM 为核心构建 TC

TPM 整体架构如图16.2所示。

国外生产的 TPM 安全芯片的主要厂家有：英飞凌、意法半导体 (ST)、Atmel、华邦电子（收购美国国家半导体公司）等。国内厂商这方面的研发起步较晚，有兆日公司的产品。

图16.3是 TPM 芯片，图16.4是 TPM 模组，根据不同的主板，应用场景有不同的模组。

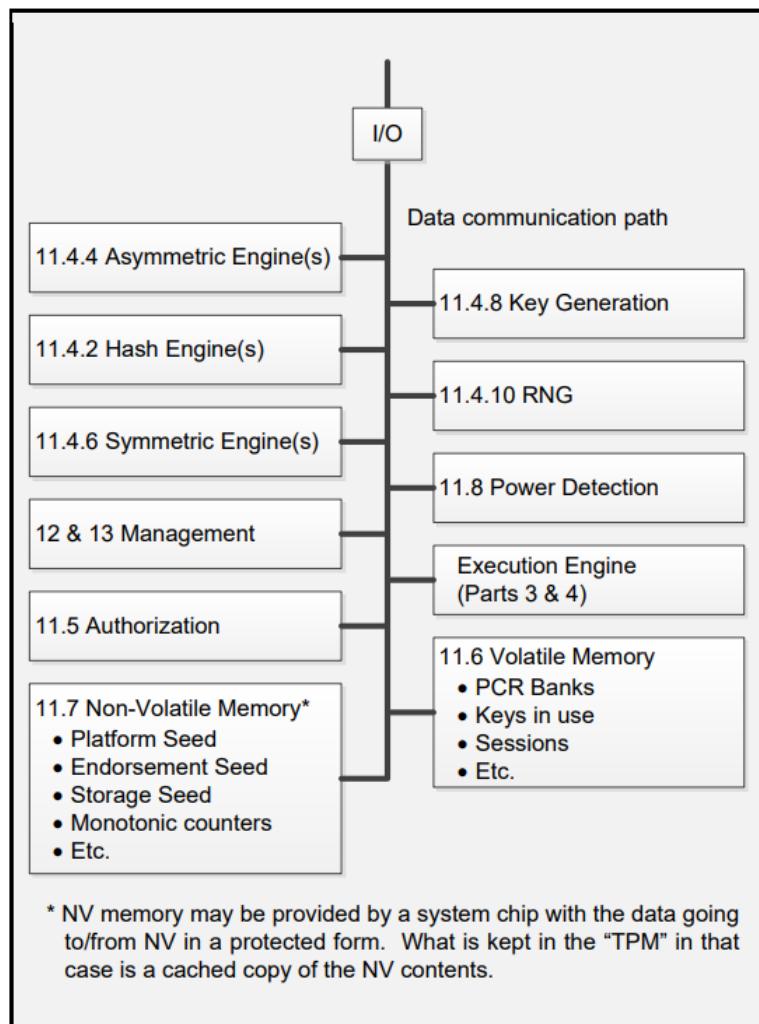


Figure 2 — Architectural Overview

图 16.2: TPM 架构<sup>3</sup>

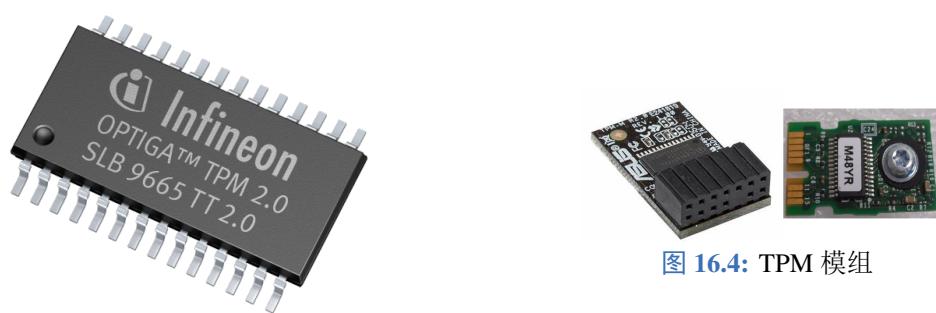


图 16.3: TPM 芯片

图 16.4: TPM 模组

TPM 芯片参数的示例见图16.5。

Parametrics	SLB 9665TT2.0
Ambient Temperature min max	-20.0 °C 85.0 °C
Applications	embedded security ; trusted computing ; PC and mobile computing with Intel x86, ARM platforms and others; embedded devices e.g. communication, gateways, printer, PoS systems, networking, ATMs
Asymmetric Cryptography	ECC ; ECC BN-256 ; ECC NIST P-256 ; ECC256 ; ECDH ; RSA1024 ; RSA2048
CPU	16-bit
Certifications	CC EAL4+ ; FIPS 140-2 level 2 (with FW update)
Delivery Forms	TSSOP-28
Interfaces	LPC
Package	TSSOP-28
Product Description	FW5.63 ; standardized security controller for computing platforms and embedded systems
Symmetric Cryptography	HMAC ; SHA-1 ; SHA-256
Use Cases	PC TPM

图 16.5: Infineon SLB 9665TT2.0 参数截图<sup>4</sup>

TPM 是基于公钥秘密体系的，所以其需要有个可信 CA，及其相应的证书，英飞凌 TPM 的 CA 证书如图16.6所示<sup>5</sup>。

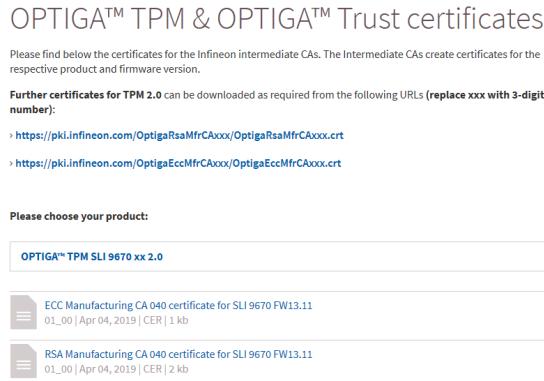


图 16.6: Infineon 网站上提供 CA 证书的网页截图

TPM 的功能组成如图16.7所示。TPM 功能组成介绍我们引用冯登国老师书中的描述 [20]：

1. 密码学系统：实现数据加密、数字签名、密码杂凑和随机数生成等各类密码算法的逻辑计算引擎，是一个不对外提供接口的内部功能模块，大部分 TPM 功能都以密码学系统为基础。
2. 平台数据保护功能：对外提供密钥管理和各类数据机密性、完整性保护功能，是直接体现密码学系统的功能类别。计算平台可依赖该功能构建安全的密钥管理和密码学计算器，这是 TPM 最基本的应用方式。
3. 身份标识功能：对外提供身份标识密钥的申请与管理功能，是远程证明（即对远程验证方报告本机完整性）的基础。
4. 完整性存储与报告功能：对外提供完整性值存储和签署（报告）功能，直接体现“可

<sup>5</sup>网页截图[https://www.infineon.com/cms/en/product/promopages/optiga TPM\\_certificates/](https://www.infineon.com/cms/en/product/promopages/optiga TPM_certificates/)

“可信性”，计算平台可依赖该功能构建平台内部的信任链，还可以在内部信任链的基础上向外部实体进行远程证明，这是 *TPM* 的主要应用方式。

5. 资源保护功能：保护 *TPM* 内部资源的各类访问控制机制。
6. 辅助功能：为 *TPM* 正常运转提供支持。计算平台可利用这些功能设定 *TPM* 的启动和工作方式，提高工作便捷性；还可以利用这些功能保护应用程序和 *TPM* 之间的通信信道，获取可信的时戳和计数器服务等。

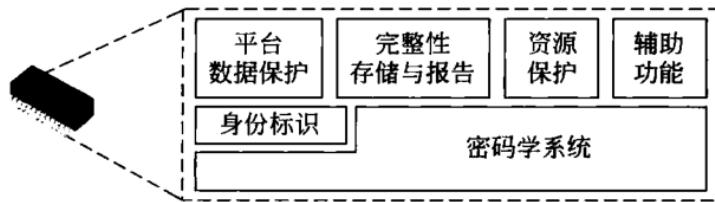


图 16.7: TPM 功能模块组成 [20]

*TPM* 芯片的组成结构如图 16.7 所示，各个组成部分介绍我们引用冯登国老师书中的描述 [20]：

1. 标志位管理器：存储与维护对于 *TPM* 正常与安全工作至关重要的内部标志位（包括使能标志位、激活标志位和属主标志位）的管理模块。
2. RSA 算法引擎：依据 PKCS#1 标准，提供 RSA 密码算法进行数据加解密以及数字签名、验证功能的运算模块，支持 2048b（推荐）、1024b 和 512b 三种安全级别。
3. 对称加密算法引擎：采用 Vernam 对称加密算法以及相关的 MGF1 密钥生成算法，可用于实现 AES 算法。
4. 随机数生成器：依据 IEEE P1363 规范，生成协议中的随机数以及对称加密算法使用的密钥的运算模块。
5. 密码杂凑引擎：依据 FIPS-180-1 标准，采用 SHA-1 算法计算密码杂凑值的运算模块。
6. 非易失性存储器：存储 *TPM* 的长期密钥（背书密钥和存储根密钥）、完整性信息、所有者授权信息及少量重要应用数据的存储模块。
7. 易失存储器：存储计算中产生的临时数据的存储模块。
8. 电源管理模块：负责常规的电源管理和物理现场信号的检测，后者对动态度量信任根等依赖物理信号的技术至关重要。
9. I/O 模块：负责 *TPM* 与外界之间以及 *TPM* 内部各物理模块之间的通信，具体包括信息的编解码、转发以及模块访问控制。

我们看一下 ST 公司一个 *TPM* 芯片的实际介绍资料，图 16.9 是 ST33TPMLPC 芯片的硬件框图，此芯片有两种封装 TSSOP28 和 VQFN32 如图 16.10 所示，芯片的主要指标描述

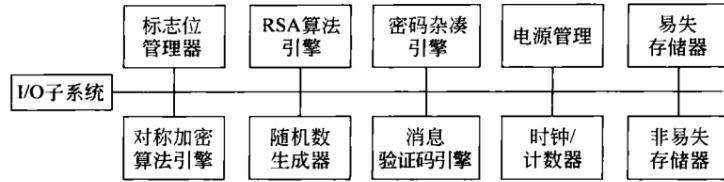
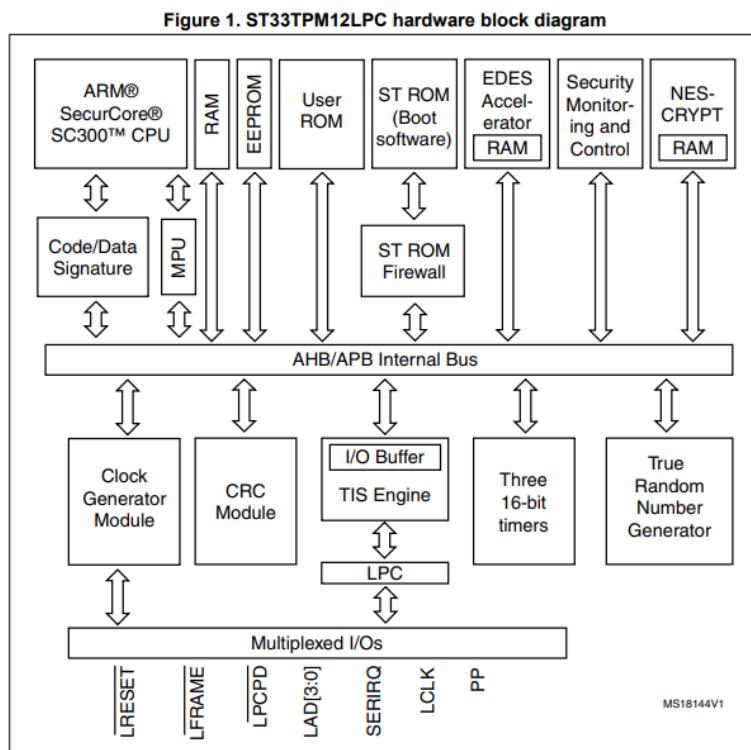


图 16.8: TPM 硬件模块组成 [20]

如16.11、16.12、16.13、16.14，ST33TPMLPC 外部接口采用的 LPC, ST 还有其他一些接口的 TPM 芯片，比如  $I^2C$ 、SPI 的，并且有针对新 TPM 2.0 规范的芯片，如 ST33GTPMI2C、ST33GTPMASPI 等。

图 16.9: TPM 硬件模块组成<sup>6</sup>

在 Windows 操作系统中可以在命令行执行:tpm.msc，来查看 TPM 管理界面。要想在操作系统中看到 TPM，在硬件配置上要使能 TPM，通常是在 BIOS 里设置，在 BIOS 里使能 TPM 的操作可参考 Intel 的技术文档“Installing and Configuring the TPM module”<sup>7</sup>。

## 16.2 相关资源

科学计算组 (Trusted Computing Group)，简称 TCG，是一个国际上的可信计算联盟，网址是<https://trustedcomputinggroup.org/>，网站上有其制定的标准，介绍资料，相关研讨活动以及 TPM 产品认证。

<sup>7</sup>Installing and Configuring the TPM module, 网址[https://www.intel.com/content/dam/support/us/en/documents/server-products/Configuring\\_the\\_TPM\\_2.0.pdf](https://www.intel.com/content/dam/support/us/en/documents/server-products/Configuring_the_TPM_2.0.pdf)



图 16.10: ST33TPM12LPC 提供的封装

**TPM features**

- Single-chip Trusted Platform Module (TPM)
- Compliant with Trusted Computing Group (TCG) Trusted Platform Module (TPM) Main specifications 1.2, Level 2, Revision 116
- Based on TCG PC Client Specific TPM Interface Specifications 1.21
- Common criteria (CC) certification based on the certified TPM Protection Profile (Revision 116) with Evaluation Assurance Level (EAL) 4+
- Up to 33-MHz Low Pin Count (LPC) interface V1.1
- Provisioned with Endorsement key and Endorsement Key certificate
- Support of clock suspension for power saving mode
- Support of Field Upgrade and Dictionary Attack protection
- Monotonic counter endurance guaranteed for 7 years
- Support of software and hardware physical presence

图 16.11: ST33TPM12LPC TPM 特点

**Security features**

- Active shield and environmental sensors
- Memory protection unit (MPU)
- Monitoring of environmental parameters (power and clock)
- Hardware and software protection against fault injection
- AIS-31 Class P2 compliant true random number generator (TRNG)
- Cryptographic algorithms:
  - RSA key generation from 512 to 2048 with a 2-byte step
  - RSA signature and encryption
  - SHA-1 and SHA-256
  - AES-128 in CTR mode

图 16.13: ST33TPM12LPC 安全特点

**Hardware features**

- ARM® SecurCore® SC300™ 32-bit RISC core
- Highly reliable CMOS EEPROM submicron technology
  - 30-year data retention at 25° C
  - 500,000 Erase/Write cycles endurance typical at 25° C
- Temperature range: 0°C to +70°C
- ESD protection up to 4 kV (HBM)
- 3.3 V supply voltage range
- 28-lead thin shrink small outline and 32-lead very thin fine pitch quad flat pack ECOPACK® packages

图 16.12: ST33TPM12LPC 硬件特点

**Performance and resource features**

- SHA1 computation for 64-byte block: 155 µs<sup>(a)</sup>
- Signature with a 2048-bit key: 150 ms<sup>(a)</sup>
- Signature with a 1024-bit key: 30 ms<sup>(a)</sup>
- NV storage allocated space: 4 Kbytes (1.2 Kbytes used by EK certificate)
- Supported 2048-bit key slots:
  - up to 10 key slots (without EK and SRK)
  - 1 key slot in volatile memory for high-frequency loading use case

图 16.14: ST33TPM12LPC 性能特点

在 TPM 网站上有一个短文“TMP 2.0 A Brief Introduction”，罗列了一本开放书籍“A Practical Guide to TPM 2.0 - Using the Trusted Platform Module in the New Age of Security”<sup>8</sup>，还有一些软件资源：

- <https://sourceforge.net/projects/ibmswtpm2/>
- [https://github.com/vbendebe/tpm2\\_server](https://github.com/vbendebe/tpm2_server)
- <http://research.microsoft.com/en-US/downloads/35116857-e544-4003-8e7b-584182dc6833/default.aspx>
- <https://github.com/PeterHuewe/linux-tpmdd/tree/tpm-emulator>
- <https://github.com/PeterHuewe/linux-tpmdd/commit/9329f13c403daf1f4bd1e715d2ba0866e089fb1d>
- <https://github.com/PeterHuewe/linux-tpmdd/commit/bbf2f7064c1452b47f11dfad340326b1205d863a>

---

<sup>8</sup><http://www.springer.com/us/book/9781430265832>

## 第 17 章 扩展阅读

---

### 17.1 姚期智——图灵奖的介绍

姚期智教授获得 2000 年图灵奖，在美国计算机协会 ACM 的图灵奖获得者介绍中，我们可以看到姚先生的主要贡献，下面摘录在下面。

In recognition of his fundamental contributions to the theory of computation, including the complexity-based theory of pseudorandom number generation, cryptography, and communication complexity.

Andrew Chi-Chih Yao was born in Shanghai, China, on December 24, 1946. After moving with his family to Hong Kong for two years he immigrated to Taiwan. In 1967 he received a B.S. in Physics from the National University of Taiwan. He then started graduate studies in Physics at Harvard University, where he received an A.M. in 1969 and a Ph.D. in 1972 under the supervision of Sheldon Glashow, winner of the 1979 Nobel Prize in Physics. He subsequently entered the Ph.D. program in Computer Science at the University of Illinois Urbana-Champaign, and received his degree just two years later, in 1975. Yao completed his dissertation, *A Study of Concrete Computational Complexity*, under the supervision of Chung Laung Liu.

After a year as an Assistant Professor in the Mathematics Department at MIT, Yao joined the Computer Science Department at Stanford University as an Assistant Professor in 1976. Over the next five years there he made a number of fundamental contributions to the theory of algorithms.

His 1977 paper “Probabilistic computations: toward a unified measure of complexity,” [1] introduced what is now known as Yao’s min-max principle, which uses von Neumann’s minimax theorem from game theory to relate average-case complexity for deterministic algorithms to worst-case complexity for randomized algorithms. Yao proved that the expected running time of any randomized algorithm on worst-case input is equal to the average-case running time of any deterministic algorithm for the worst-case distribution of inputs. Yao’s principle has become a fundamental technique for reasoning about randomized algorithms and complexity, and has also been applied in areas such as property testing and learning theory.

Around this time Yao also made fundamental contributions to the theory of data structures. His 1978 paper, “Should tables be sorted?” [2] introduced the cell-probe model, an abstract model of data structures where the cost of a computation is measured by the total number of memory accesses. This model has been widely used in creating lower bound proofs of algorithms.

Yao spent a year as a Professor in the Computer Science Division of the University of

California, Berkeley, and subsequently returned to Stanford as a full Professor in 1982. During the early 1980's, Yao produced a number of papers which had a lasting impact on the foundations of cryptography, computer security, computational complexity and randomized computation. This work was significant not only for results obtained, but also for the introduction of problems, models and techniques which are now considered fundamental in their respective areas.

His 1981 paper with Danny Dolev, "On the security of public-key protocols," [8] introduced a formal model for symbolic reasoning about security protocols. Since its introduction, this "Dolev-Yao model" has been the starting point for most work done on symbolic security, including recent work on the security of complexity-based cryptography. This continues to be an active area of research. Yao also made significant contributions to cryptography and complexity-based approaches to security. In 1982 he published "Theory and applications of trapdoor functions" [7] and "Protocols for secure computations" [5]. These works, which were introduced at the same conference, stand as seminal contributions in cryptography and secure computation.

The first of these papers addresses the then newly-emerging field of public-key cryptography from a theoretical perspective, lays the foundation for a theory of computational randomness, and initiates a study of its relationship to computational hardness. Yao provides a definition of pseudorandom number generator which is based on computational complexity, and proposes a definition of "perfect" —in current terminology, "pseudorandom" —probability distribution ensembles. (An ensemble is perfect if it cannot be distinguished from a truly random ensemble by any feasible distinguisher, where such distinguishers are formalized using the notion of a polynomial time randomized algorithm.) Yao relates his notion of pseudorandomness to the idea of a statistical test, a notion already used in the study of pseudorandom number generators, and shows that one particular test, known as the next-bit test, is adequate for characterizing pseudorandomness. Having defined perfect ensembles, Yao then defined a pseudorandom number generator as an efficient randomized algorithm which uses a limited number of truly random bits in order to output a sample from a perfect distribution whose size is polynomial in the number of random bits used.

The next fundamental contribution of the paper addresses the question of what computational assumptions are adequate for the existence of pseudorandom number generators. Advances in public-key cryptography indicated that secure encryption could be based on the assumed hardness of certain computational problems such as quadratic residuosity, or problems related to factoring integers. Yao asked whether one could make a general assumption about computational hardness which could be used to obtain pseudorandomness and hence, through standard techniques, cryptographic security. For this he formalized the notion of a one-way function that is easy to compute but hard to invert for a large fraction of inputs. He proved that one-way functions with certain properties may be used to construct pseudorandom number generators. This inspired a series of important results that refined Yao's work, and it continues to be an area of active research. The contributions of this paper to pseudorandomness form an essential component of modern

cryptography. In addition, the paper proposes a new field of computational information theory which refines Shannon's theory by taking computational resources into account. Yao gives a definition of computational entropy, and uses it to give a characterization of encryption security. This definition of entropy is now important in areas such as leakage-resilient cryptography.

The second paper introduces a new paradigm for secure function evaluation, and introduces the famous "Millionaires' Problem". Yao gives a protocol which allows two parties, each holding a number, to determine who has the larger number without revealing the actual values. The Millionaires' Problem is a two-party instance of a more general class of secure multiparty computation problems which are now essential to the study of secure cryptographic protocols. With the advent of wide-scale distributed computing and the ubiquity of cryptographic protocols, Yao's contributions in this area have had a significant impact on networked computing.

In the 1980's, Professor Yao introduced models and techniques whose ramifications are still being felt in research in complexity, computational randomness, cryptography, and security. Some of his most influential ideas were disseminated in lectures building on his published results. One example is the XOR-lemma, which uses computational hardness to produce pseudorandomness. Yao addressed whether the hardness of a problem may be amplified by combining multiple instances of the problem, in this case through the use of the bitwise exclusive-OR operation. While interesting in its own right, the XOR-lemma is an essential technique in the area of derandomization, which seeks generic methods for eliminating the need for randomness in the efficient solution of algorithmic problems. More generally, it helps determine whether certain classes of problems that are solved efficiently with randomization can instead be solved deterministically.

A second example is the garbled circuit technique, an important tool in secure multiparty computation which was used implicitly by Yao in his 1982 secure computation paper as well as in a 1986 paper "How to generate and exchange secrets" [6]. Recent advances in computing power have made the garbled circuit technique practical for large-scale computational problems, for example in privacy-preserving matching in DNA databases.

In 1986, Yao moved to Princeton University, where he became the William and Edna Macaleer Professor of Engineering and Applied Science. During this period he continued his work on the foundations of cryptography. He also built on previous work in areas such as decision tree and communication complexity. Yao made substantial contributions to the theory of lower bounds for algebraic decision trees, an area he established in a 1982 Journal of Algorithms paper [9] co-authored with J.M. Steele. This work exploited deep relationships between algebraic decision trees and mathematical results in algebraic geometry. He also investigated the use of randomization in decision trees. Professor Yao introduced the theory of communication complexity in a 1979 paper "Some complexity questions related to distributive computing" [3]. In his 1993 paper, "Quantum circuit complexity" [4] he extended communication complexity to quantum computing. Starting in the 1990's, Professor Yao began to work extensively on

quantum computing, communication and information theory. He continues to make significant contributions in these areas.

Andrew Yao became a Professor in the Center for Advanced Study and Director of the Institute for Theoretical Computer Science at Tsinghua University, Beijing, in 2004. Since 2005 he has also been Distinguished Professor-at-Large of the Chinese University of Hong Kong. His recent contributions include work in security protocols, universally composable secure computation, quantum computing, and the theory of algorithms.

Yao is active in graduate supervision, and has mentored over twenty Ph.D. students. He is married to Professor Frances Yao, a computer scientist and leading researcher in computational geometry, algorithms and cryptography.

Author: Bruce Kapron

### Short Annotated Bibliography

1.Yao, Andrew Chi-Chih, “Probabilistic Computations: Toward a Unified Measure of Complexity” (Extended Abstract), 18th Annual Symposium on Foundations of Computer Science (FOCS ’ 77), IEEE Computer Society, 1977, pp. 222-227.

This paper considers probability in computation from two perspectives: probability distributions on inputs, and the use randomization in algorithms. The two perspectives are unified with Yao’s min-max principle, which states that the worst-case expected running time of a randomized algorithm for a problem is equal to the average-case running time of any deterministic algorithm, for the worst-case distribution on problem inputs.

2.Yao, Andrew Chi-Chih, “Should Tables Be Sorted?” (Extended Abstract), 19th Annual Symposium on Foundations of Computer Science (FOCS ’ 78). IEEE Computer Society, 1978, pp. 22-27.

In this paper Yao introduces the cell-probe model for the analysis of data structures in which the cost of a computation is the number of accesses to a random access memory with cell size  $\log n$ .

3.Yao, Andrew Chi-Chih, “Some Complexity Questions Related to Distributive Computing” (Preliminary Report), Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC ’ 79), ACM Press, 1979, pp. 209-213, available [here](#).

4.Yao, Andrew Chi-Chih, “Quantum Circuit Complexity,” 34th Annual Symposium on Foundations of Computer Science (FOCS ’ 93), IEEE Computer Society, 1993, pp. 352-361.

The first of these two papers introduces communication complexity, a measure which has proven to be useful in obtaining complexity bounds in areas including parallel computation, circuits, and data structures. The second paper extends the classical model of the first paper to situations in which participants in a protocol may exchange quantum bits.

5.Yao, Andrew Chi-Chih, “Protocols for Secure Computations” (Extended Abstract), 23rd Annual Symposium on Foundations of Computer Science (FOCS ’ 82), IEEE Computer Society, 1982, pp. 160-164.

6.Yao, Andrew Chi-Chih, “How to Generate and Exchange Secrets” (Extended Abstract), 27th Annual Symposium on Foundations of Computer Science (FOCS ’ 87), IEEE Computer Society, 1986, pp. 162-167.

The first of these two papers introduces secure function evaluation, and gives a protocol for the famous Millionaires’ Problem. In talks related to these papers Yao introduced a technique for secure computation known as garbled circuits.

7.Yao, Andrew Chi-Chih, “Theory and Applications of Trapdoor Functions” (Extended Abstract), 23rd Annual Symposium on Foundations of Computer Science (FOCS ’ 82), IEEE Computer Society, 1982, pp. 80-91.

This influential paper introduces a number of notions and techniques that are essential to modern complexity-based cryptography, including computational indistinguishability as a basis for characterizing pseudorandom generators, and the use of one-way functions as a complexity-theoretic basis for obtaining cryptographic security via pseudorandomness. Yao also introduces ideas which have had a major impact in other areas such as computational information theory and derandomization.

8.Dolev, Danny and Andrew Chi-Chih Yao, “On the security of public key protocols,” IEEE Transactions on Information Theory, Vol. 29, Num. 2, 1983, pp. 198-207.

The authors introduce a model for the analysis of cryptography-based security protocols which has become known as the Dolev-Yao model, and is the basis for many current approaches to formal verification of such protocols.

9.Steele, J. Michael and Andrew Chi-Chih Yao, “Lower Bounds for Algebraic Decision Trees,” Journal of Algorithms, Vol. 3, Num. 1, 1982, pp. 1-8.

10.Yao, Andrew Chi-Chih, “Lower Bounds to Randomized Algorithms for Graph Properties,” Journal of Computing Systems Science, Vol. 42, Num. 3, 1991, pp. 267-287.

Professor Yao has made extensive contributions to structured models of computation, including decision trees. The first of these two papers proposes algebraic decision trees, a generalization of the linear decision trees of Dobkin and Lipton. The second paper considers randomization in the Boolean decision tree model for deciding graph properties.

## 17.2 2021 年理论计算科学和离散数学领域学者获 Abel 奖

3月17日，2021年阿贝尔奖揭晓。挪威科学和文学院决定将2021年阿贝尔奖授予来自匈牙利，布达佩斯罗兰大学的László Lovász 和来自美国，普林斯顿高等研究院的Avi Wigderson，以表彰两位科学家在理论计算机科学和离散数学方面做出的杰出贡献，以及在将之塑造为现代数学中心领域中发挥的主导作用。获奖者将分享750万挪威克朗的奖

金(约合 580 万人民币)。

理论计算机科学 (TCS, 全称 theory computer science) 是研究计算的能力和局限性的科学。其根源可追溯至 Kurt Gdel、Alonzo Church、Alan Turing 和 John von Neumann 所做的基础性研究, 这些研究推动了真正的物理计算机的发展。TCS 包含两个互补的分支学科, 即算法设计 (为大量计算问题开发有效方法) 和计算复杂性 (证明算法效率的固有限制)。

自然离散数学 (discrete mathematic) 和 TCS 一直是紧密联系的两个领域。虽然这两个领域都从更传统的数学领域中获益匪浅, 但其对传统数学领域的反向影响也越来越大。TCS 的应用、概念和技术带来了新的挑战, 开辟了新的研究方向, 解决了纯数学和应用数学中的重要开放性问题。

在过去几十年中, Lászlé Lovász 和 Avi Wigderson 一直是推动实现相关发展的主导力量。Lászlé Lovász 与 Arjen Lenstra 和 Hendrik Lenstra 一起开发出了 LLL 格基约减算法。给定一个高维整数格 (网格), 此算法可以为之找到一个不错的近乎正交基。除了因式分解有理多项式的算法等一些应用之外, LLL 算法也是一个受密码专家欢迎的工具, 并成功破解了所提出的几个加密系统。令人惊讶的是, LLL 算法的分析还用于设计和保证较新的格基加密系统的安全性, 这些系统甚至能够抵御量子计算机的攻击。

Avi Wigderson 对计算复杂性的各个方面, 特别是随机性在计算中的作用, 做出了广泛而深刻的贡献。随机算法是指通过抛硬币的方法, 以高概率计算正确解的算法。Wigderson 与 Impagliazzo 和 Valentine Kabanets 的后续研究进一步证明了即使是对已知的随机算法的具体问题, 有效的确定性算法也意味着必须存在这样一个难解的问题。

阿贝尔委员会主席 Hans Munthe-Kaas 表示, “在过去几十年中, Lovász 和 Wigderson 一直是推动实现相关发展的主导力量。他们的研究在很多方面是相互交错的, 并都对理解计算中的随机性和探索高效计算的边界做出了巨大贡献。”

他说: “正是由于这两位所做出的突破性贡献, 离散数学和相对”年轻的理论计算机科学领域现已牢固确立为现代数学的中心领域。

#### 关于阿贝尔奖 Abel prize

阿贝尔奖设立于 2002 年 1 月 1 日, 于 2003 年 6 月 3 日首次颁发。阿贝尔奖与菲尔兹奖、沃尔夫奖并称为国际最高数学“三大奖”。

(文章来源: <https://baijiahao.baidu.com/s?id=1694544933927331714&wfr=search&for=pc>)

## 17.3 ZUC

首先我们把 ZUC 看成一个加密盒子, 一个黑盒子, 看看 ZUC 是做什么的。输入一个 128 bits 的初始密钥, 一个 128 bits 的初始向量, 生成一个 32bit 的密钥流 (也就是生成一个密钥字流 (key words)), 如图17.1所示.



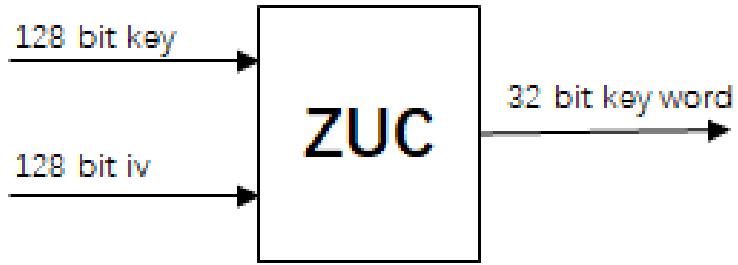


图 17.1: ZUC 算法

### 17.3.1 密钥装载

$k = k_0 \| k_1 \| \dots \| k_{15}$ ,  $k_i$  为 8 bits 长。

$iv = iv_0 \| iv_1 \| \dots \| iv_{15}$ ,  $iv_i$  为 8 bits 长。

$D = d_0 \| d_1 \| \dots \| d_{15}$ ,  $d_i$  为 15 bits 长,  $D$  为 240 bits 常量。

$s_i = k_i \| d_i \| iv_i$ ,  $0 \leq i \leq 15$ ,  $s_i$  为 31 bits 长, 用于初始化 LFSR。

### 17.3.2 初始化阶段

密钥装载完成  $s_0, \dots, s_{15}$  初始化, 并且  $R_1, R_2$  全为 0, 以下过程执行 32 次。

$$\left\{ \begin{array}{l} BitReconstruction(); \\ W = F(X_0, X_1, X_2); \\ LFSRWithInitialisationMode(u); \end{array} \right.$$

其中  $u = W >> 1$ , 也就是  $W$  去掉最低位得  $u$ 。

其中 LFSR 的初始化过程如下:

```

LFSRWithInitialisationMode(u)
{
(1)  $v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0 \pmod{2^{31} - 1}$ ;
(2)  $s_{16} = (v + u) \pmod{2^{31} - 1}$ ;
(3) if  $s_{16} = 0$ , then set  $s_{16} = 2^{31} - 1$ ;
(4)  $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ ;
}
  
```

### 17.3.3 工作阶段

执行一次:  $\left\{ \begin{array}{l} BitReconstruction(); \\ W = F(X_0, X_1, X_2); \\ LFSRWithInitialisationMode(u); \end{array} \right.$

循环执行:  $\left\{ \begin{array}{l} BitReconstruction(); \\ Z = F(X_0, X_1, X_2) \oplus X_3; \\ LFSRWithWorkMode(); \end{array} \right.$

其中 LFSR 在工作模式下的过程如下:

```
LFSRWithInitialisationMode(u)
{
(1)  $v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0 \pmod{2^{31}-1}$ ;
(2) if  $s_{16} = 0$ , then set  $s_{16} = 2^{31} - 1$ ;
(3)  $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ ;
}
```

### 17.3.4 F 函数

$L_1, L_2$  是两个线性变换,  $S = (S_0, S_1, S_2, S_3) = (S_0, S_1, S_0, S_1), S_i$  是个 8 入 8 出的盒子, 查表方法是高四比特为行号, 低 4 比特为列号。具体可以看英文标准给出的示例比较易懂。

### 17.3.5 ZUC 设计理念及准则

我就 ZUC 的设计理念去邮件询问了设计者之一冯秀涛博士, 本部分内容摘自其邮件中提供的 ppt(截图见17.2)。

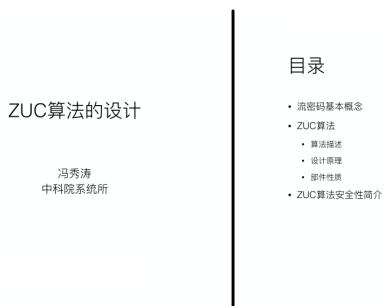


图 17.2: 冯秀涛博士 ppt

#### 17.3.5.1 设计理念

我们主要采用分解 + 融合的设计思想, 即将算法分解成若干部件, 每个部件偏重某些攻击方法:

- **LFSR:** 采用非 2 特征素域设计, 其在  $F_2$  上是非线性的, 可以抵抗当前许多针对二元域上的密码攻击方法, 譬如区分分析、相关分析、代数分析等。
- **比特重组:** 打破素域  $F_p$  上的代数结构, 使得针对  $F_p$  上的密码攻击方法变得十分困难, 譬如代数分析。
- **非线性函数 F:** 借鉴分组密码设计思想, 采用 S 盒和最佳扩展准则的线性变换, 进一步打破  $F_p$  上的代数结构, 同时增强抵抗传统基于  $F_2$  的密码攻击的能力, 譬如区分分析、相关分析等。

最后通过它们以及不同代数结构之间相互融合达到整体安全性的目的。

### 17.3.5.2 设计准则

LFSR 的设计遵循如下准则：

- 为了保证序列源具有大的周期和好的统计特性，其特征多项式  $f(x)$  须是  $F_p$  上的本原多项式；
- 为了便于快速软硬件实现， $f(x)$  的非零项系数的二进制汉明重量必须尽可能的低；
- 为了保证在模  $2^{31} - 1$  和模 2 之间低的比特符合率， $f(x)$  的所有非零系数的汉明总量之和必须为奇数；
- $f(x)$  的所有非零项系数中 1 的位置尽可能的两两不同；
- $f(x)$  没有明显的低权重低次数的倍式。

基于以上准则，我们选择了如下本原多项式：

$$f(x) = x^{16} - (2^{15}x^{15} + 2^{17}x^{13} + 2^{21}x^{10} + 2^{20}x^4 + (2^8 - 1))$$

比特重组的设计主要遵循以下准则：

- 便于软硬件实现；
- 4 个 32 比特输出必须拥有好的统计特性；
- 4 个 32 比特的输出在不同时刻之间重叠的比特数尽可能的低。

非线性函数 F 的设计准则如下：

- 带 64 比特记忆；
- 使用 S 盒提高非线性；
- 使用具有好的扩散特性的线性变换；
- 每个记忆单元的更新必须同时依赖 3 个以上独立寄存器单元的值；
- 非线性函数 F 的输出必须是平衡的且具有好的随机性；
- 非线性函数 F 应该便于软硬件实现，且具有低的硬件实现代价。

## 17.4 finite field

A field with a finite number of elements. First considered by E. Galois.

The number of elements of any finite field is a power  $p^n$  of a prime number  $p$ , which is the [[Characteristic of a field|characteristic]] of this field. For any prime number  $p$  and any natural number  $n$  there exists a (unique up to an isomorphism) field of  $p^n$  elements. It is denoted by  $\text{GF}(p^n)$  or by  $\mathbb{F}_{p^n}$ . The field  $\text{GF}(p^m)$  contains the field  $\text{GF}(p^n)$  as a subfield if and only if  $m$  is divisible by  $n$ . In particular, any field  $\text{GF}(p^n)$  contains the field  $\text{GF}(p)$ , which is called the [[prime field]] of characteristic  $p$ . The field  $\text{GF}(p)$  is isomorphic to the field  $\mathbb{Z}/p\mathbb{Z}$  of residue classes of the ring of integers modulo  $p$ . In any fixed [[Algebraic closure|algebraic closure]]  $\Omega$  of  $\text{GF}(p)$  there exists exactly one subfield  $\text{GF}(p^n)$  for each  $n$ . The correspondence  $n \leftrightarrow \text{GF}(p^n)$  is an isomorphism between the lattice of natural numbers with respect to division and the lattice of finite algebraic extensions (in  $\Omega$ ) of  $\text{GF}(p)$  with respect to inclusion. The lattice of finite algebraic extensions of any Galois field within its fixed algebraic closure is such a lattice.



The algebraic extension  $\text{GF}(p^n)/\text{GF}(p)$  is simple, i.e. there exists a primitive element  $\alpha \in \text{GF}(p^n)$  such that  $\text{GF}(p^n) = \text{GF}(p)(\alpha)$ . Such an  $\alpha$  will be any root of any irreducible polynomial of degree  $n$  from the ring  $\text{GF}(p)[X]$ . The number of primitive elements of the extension  $\text{GF}(p^n)/\text{GF}(p)$  equals

$$\sum_{d|n} \mu(d)p^{n/d}$$

where  $\mu$  is the [[Möbius function|Möbius function]]. The additive group of the field  $\text{GF}(p^n)$  is naturally endowed with the structure of an  $n$ -dimensional vector space over  $\text{GF}(p)$ . As a basis one may take  $1, \alpha, \dots, \alpha^{n-1}$ . The non-zero elements of  $\text{GF}(p^n)$  form a multiplicative group,  $\text{GF}(p^n)^*$ , of order  $p^n - 1$ , i.e. each element of  $\text{GF}(p^n)^*$  is a root of the polynomial  $X^{p^n-1} - 1$ . The group  $\text{GF}(p^n)^*$  is cyclic, and its generators are the primitive roots of unity of degree  $p^n - 1$ , the number of which is  $\phi(p^n - 1)$ , where  $\phi$  is the [[Euler function|Euler function]]. Each primitive root of unity of degree  $p^n - 1$  is a primitive element of the extension  $\text{GF}(p^n)/\text{GF}(p)$ , but the converse is not true. More exactly, out of the

$$\frac{1}{n} \sum_{d|n} \mu(d)p^{n/d}$$

irreducible unitary polynomials of degree  $n$  over  $\text{GF}(p)$  there are  $\phi(p^n - 1)/n$  polynomials of which the roots are generators of  $\text{GF}(p^n)$ .

The set of elements of  $\text{GF}(p^n)$  coincides with the set of roots of the polynomial  $X^{p^n} - X$  in  $\Omega$ , i.e.  $\text{GF}(p^n)$  is characterized as the subfield of elements from  $\Omega$  that are invariant with respect to the automorphism  $\tau : x \mapsto x^{p^n}$ , which is known as the Frobenius automorphism. If  $\text{GF}(p^m) \supset \text{GF}(p^n)$ , the extension  $\text{GF}(p^m)/\text{GF}(p^n)$  is normal (cf. [[Extension of a field|Extension of a field]]), and its [[Galois group|Galois group]]  $\text{Gal}(\text{GF}(p^m)/\text{GF}(p^n))$  is cyclic of order  $m/n$ . The automorphism  $\tau$  may be taken as the generator of  $\text{Gal}(\text{GF}(p^m)/\text{GF}(p^n))$ .

[[Category:Field theory and polynomials]]

## 17.5 One way function

Informally, a function  $f$  is a one-way function if

1. The description of  $f$  is publicly known and does not require any secret information for its operation.
2. Given  $x$ , it is easy to compute  $f(x)$ .
3. Given  $y$ , in the range of  $f$ , it is hard to find an  $x$  such that  $f(x)=y$ . More precisely, any efficient algorithm solving a P-problem succeeds in inverting  $f$  with negligible probability.

The existence of one-way functions is not proven. If true, it would imply  $P \neq NP$ . Therefore, it would answer the complexity theory NP-problem question of whether all apparently NP-problems are actually P-problems. Yet a number of conjectured one-way functions are routinely used in commerce and industry. For example, it is conjectured, but not proved, that the following are one-way functions:



1. Factoring problem:  $f(p,q)=pq$ , for randomly chosen primes p,q.
2. Discrete logarithm problem:

$$f(p, g, x) = (p, g, g^x \pmod{p})$$

for g a generator of  $Z_p^*$  for some prime p.

3. Discrete root extraction problem:  $f(p, q, e, y) = (p, q, e, y^e \pmod{pq})$ , for y in  $Z_{pq}^*$ , e in  $Z_{pq}$  and relatively prime to  $(p-1)(q-1)$ , and p,q primes. This is the function commonly known as RSA encryption.

4. Subset sum problem:  $f(a, b) = < \sum_{i=1}^n a_i b_i, b >$ , for  $a_i \in \{0, 1\}$ , and n-bit integers  $b_i$ .

5. Quadratic residue problem.

#### REFERENCES:

Luby, M. Pseudorandomness and Cryptographic Applications. Princeton, NJ: Princeton University Press, 1996.

Ziv, J. "In Search of a One-Way Function" 4.1 in Open Problems in Communication and Computation (Ed. T. M. Cover and B. Gopinath). New York: Springer-Verlag, pp. 104-105, 1987.

FROM: <https://mathworld.wolfram.com/One-WayFunction.html>

## 17.6 enigma

在密码学史中，恩尼格玛密码机（见图17.3）（德语：Enigma，又译哑谜机，或“谜”式密码机）是一种用于加密与解密文件的密码机。确切地说，恩尼格玛是对二战时期纳粹德国使用的一系列相似的转子机械加解密机器的统称，它包括了许多不同的型号，为密码学对称加密算法的流加密。

参考网站 <https://www.public-enigma.com/>，这个网站上有 enigma 密码机的模拟程序 Enigma Sim。

## 17.7 Dan Boneh,Cryptography I

Dan Boneh 是 Stanford 大学的教授，他共享了一个六周的密码学基础的课程：<https://www.coursera.org/learn/crypto/supplement/DGhhJ/lecture-slides-for-all-six-weeks>。

## 17.8 各种安全模型

安全模型有很多种，不同模型有不同的作用范围，不同的安全目标，不同的视角。





图 17.3: 德国海军使用的恩尼格码 (Enigma) 机器

### 17.8.1 Bell-LaPadula model

Bell-Lapadula 模型简称为 BLP 模型，是一种强制访问控制（MAC:Mandatory Access Control）模型。下面是 WIKI<sup>1</sup>关于此模型的介绍。

The Bell-LaPadula Model (BLP) is a state machine model used for enforcing access control in government and military applications.[1] It was developed by David Elliott Bell [2] and Leonard J. LaPadula, subsequent to strong guidance from Roger R. Schell to formalize the U.S. Department of Defense (DoD) multilevel security (MLS) policy.[3][4][5] The model is a formal state transition model of computer security policy that describes a set of access control rules which use security labels on objects and clearances for subjects. Security labels range from the most sensitive (e.g., "Top Secret"), down to the least sensitive (e.g., "Unclassified" or "Public").

The Bell-LaPadula model is an example of a model where there is no clear distinction of protection and security.[6]

#### Features

The Bell-LaPadula model focuses on data confidentiality and controlled access to classified information, in contrast to the Biba Integrity Model which describes rules for the protection of data integrity. In this formal model, the entities in an information system are divided into subjects and objects. The notion of a "secure state" is defined, and it is proven that each state transition preserves security by moving from secure state to secure state, thereby inductively proving that the system satisfies the security objectives of the model. The Bell-LaPadula model is built on

<sup>1</sup><https://encyclopedia.thefreedictionary.com/Bell-LaPadula+model> , 此网站是 wiki 的一个镜像。

the concept of a state machine with a set of allowable states in a computer system. The transition from one state to another state is defined by transition functions.

A system state is defined to be "secure" if the only permitted access modes of subjects to objects are in accordance with a security policy. To determine whether a specific access mode is allowed, the clearance of a subject is compared to the classification of the object (more precisely, to the combination of classification and set of compartments, making up the security level) to determine if the subject is authorized for the specific access mode. The clearance/classification scheme is expressed in terms of a lattice. The model defines two mandatory access control (MAC) rules and one discretionary access control (DAC) rule with three security properties:

The Simple Security Property - a subject at a given security level may not read an object at a higher security level.

The \*-property (read "star"-property) - a subject at a given security level must not write to any object at a lower security level, and not read any object at a higher security level.

The Discretionary Security Property - use of an access matrix to specify the discretionary access control.

The transfer of information from a high-sensitivity document to a lower-sensitivity document may happen in the Bell-LaPadula model via the concept of trusted subjects. Trusted Subjects are not restricted by the Star-property. Trusted Subjects must be shown to be trustworthy with regard to the security policy. This security model is directed toward access control and is characterized by the phrase: "no read up, no write down." Compare the Biba model, the Clark-Wilson model and the Chinese Wall model.

With Bell-LaPadula, users can create content only at or above their own security level (i.e. secret researchers can create secret or top-secret files but may not create public files; no write-down). Conversely, users can view content only at or below their own security level (i.e. secret researchers can view public or secret files, but may not view top-secret files; no read-up).

The Bell - LaPadula model explicitly defined its scope. It did not treat the following extensively:

Covert channels. Passing information via pre-arranged actions was described briefly. Networks of systems. Later modeling work did address this topic.

Policies outside multilevel security. Work in the early 1990s showed that MLS is one version of boolean policies, as are all other published policies.

**Strong Star Property** The Strong Star Property is an alternative to the \*-Property, in which subjects may write to objects with only a matching security level. Thus, the write-up operation permitted in the usual \*-Property is not present, only a write-to-same operation. The Strong Star Property is usually discussed in the context of multilevel database management systems and is motivated by integrity concerns.[7] This Strong Star Property was anticipated in the Biba model where it was shown that strong integrity in combination with the Bell-LaPadula model resulted in reading and writing at a single level.

**Tranquility principle** The tranquility principle of the Bell-LaPadula model states that the classification of a subject or object does not change while it is being referenced. There are two forms to the tranquility principle: the "principle of strong tranquility" states that security levels do not change during the normal operation of the system. The "principle of weak tranquility" states that security levels may never change in such a way as to violate a defined security policy. Weak tranquility is desirable as it allows systems to observe the principle of least privilege. That is, processes start with a low clearance level regardless of their owners clearance, and progressively accumulate higher clearance levels as actions require it.

### Limitations

Only addresses confidentiality, control of writing (one form of integrity), \*-property and discretionary access control Covert channels are mentioned but are not addressed comprehensively. The tranquility principle limits its applicability to systems where security levels do not change dynamically. It allows controlled copying from high to low via trusted subjects. The state-transition model does not contain any state invariants. The overall process may take more time.

## 17.8.2 ISO 安全模型

本节内容来源于开源电子书<sup>2</sup>, 本部分内容在中文参考书 [4] 也有详尽描述。

### ISO security architecture

The ISO OSI (open systems interconnect) seven-layer model of distributed systems is well known and is repeated in this figure 17.4.

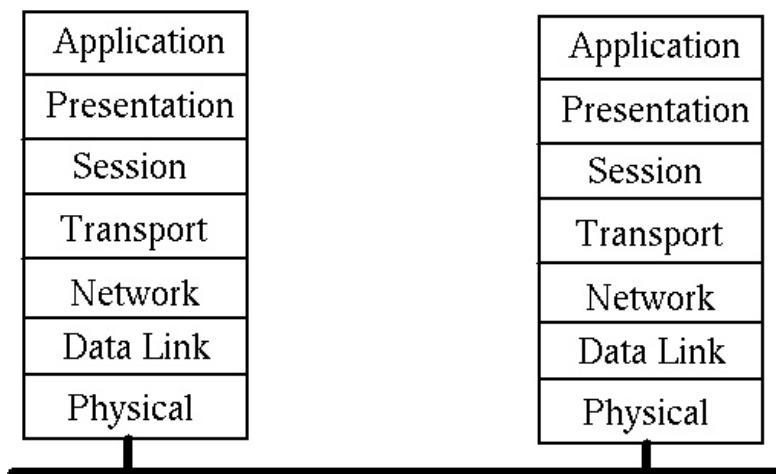


图 17.4: ISO 安全架构

What is less well known is that ISO built a whole series of documents upon this architecture. For our purposes here, the most important is the ISO Security Architecture model, ISO 7498-2.

### Functions and levels

<sup>2</sup> 内容来源于开源电子书, [https://tumregels.github.io/Network-Programming-with-Go/security/iso\\_security\\_architecture.html](https://tumregels.github.io/Network-Programming-with-Go/security/iso_security_architecture.html)

The principal functions required of a security system are:

- Authentication - proof of identity
- Data integrity - data is not tampered with
- Confidentiality - data is not exposed to others
- Notarization/signature
- Access control
- Assurance/availability

These are required at the following levels of the OSI stack:

- Peer entity authentication (3, 4, 7)
- Data origin authentication (3, 4, 7)
- Access control service (3, 4, 7)
- Connection confidentiality (1, 2, 3, 4, 6, 7)
- Connectionless confidentiality (1, 2, 3, 4, 6, 7)
- Selective field confidentiality (6, 7)
- Traffic flow confidentiality (1, 3, 7)
- Connection integrity with recovery (4, 7)
- Connection integrity without recovery (4, 7)
- Connection integrity selective field (7)
- Connectionless integrity selective field (7)
- Non-repudiation at origin (7)
- Non-repudiation of receipt (7)

### Mechanisms

- Peer entity authentication
  - encryption
  - digital signature
  - authentication exchange
- Data origin authentication
  - encryption
  - digital signature
- Access control service
  - access control lists
  - passwords
  - capabilities lists
  - labels
- Connection confidentiality
  - encryption
  - routing control
- Connectionless confidentiality



- encryption
- routing control
- Selective field confidentiality
  - encryption
- Traffic flow confidentiality
  - encryption
  - traffic padding
  - routing control
- Connection integrity with recovery
  - encryption
  - data integrity
- Connection integrity without recovery
  - encryption
  - data integrity
- Connection integrity selective field
  - encryption
  - data integrity
- Connectionless integrity
  - encryption
  - digital signature
  - data integrity
- Connectionless integrity selective field
  - encryption
  - digital signature
  - data integrity
- Non-repudiation at origin
  - digital signature
  - data integrity
  - notarization
- Non-repudiation of receipt
  - digital signature
  - data integrity
  - notarization

### 17.8.3 IATF 安全模型 (ISSE)

### 17.8.4 BS7799 PDCA 模型

## 17.9 课程设计基本概念

本部分内容主要参考自“Curriculum Design: Definition, Purpose and types”<sup>3</sup>。

课程设计 (curriculum design) 是一个用来描述在一节课或者一个课程中, 有目的、有意地和系统地组织 (purposeful, deliberate and systematic organization) 教学模块 (instructional blocks), 换句话来说, 是教师教学计划 (plan instruction) 的一种方式。教师在设计课程时必须明确“做什么、谁做、日程安排” (what will be done, who will do it, and what schedule to follow) .

### 17.9.1 课程设计的目的

课程设计的终极目标是改善学生的学习 (improve student learning)。教师在课程设计时, 脑中一定要有一个明确的教育目的 (specific educational purpose in mind)。在课程设计时也要特别注意学习目标在不同阶段的衔接和互补 (learning goals are aligned and complement each other from one stage to the next)。

### 17.9.2 课程设计的类型

#### 17.9.2.1 学科为中心的课程设计 (subject-centered curriculum design)

学科为中心的课程设计围绕一个特定的学科材料或者准则, 需要描述“需要学什么和怎么学”, 通常教师会给出一个需要交给学生的知识点表, 同时附上一个例子来说说明如何交给学生, 目前绝大多数课程设计都是采用这种方式。这种方式不是以学生为中心, 课程设计时没有考虑学生的学习方式, 学生的主动性和积极性会是一个问题。

#### 17.9.2.2 学习者为中心的课程设计 (learner-centered curriculum design)

学习者为中心的方法考虑到每个学习者的需要、兴趣和目标 (each individual's needs, interests, and goals), 这种方法承认学生的不一致性, 根据学生的需要来调整, 这意味着赋能学习者 (empower learners), 同时也允许学习者通过选择来重塑他们的教育 (shape their education through choices) .

以学习者为中心的教学方案 (instructional plans) 给学生选择作业、学习体验或者活动的机会, 这可以调动学生, 帮助他们在他们学习的内容上投入更多 (help them stay engaged in the material that they are learning).

这种方法的缺点是劳动密集型 (labor-intensive), 开发有区别的指导 (developing differentiated instruction) 对老师是个巨大压力, 需要老师开发出不同的指导, 针对每个学生的需求找到其所需的引导材料, 在这种情况下, 老师的时间、经验、技巧都要有足够的

<sup>3</sup><https://www.thoughtco.com/for-educators-4132509>

保障和支持，同时也要求老师在学生的需要、课程所需的输出之间寻找平衡，这也是一個巨大的挑战。

### 17.9.2.3 问题为中心的课程设计 (problem-centered curriculum design)

问题为中心的课程设计也是一种学生为中心的形式，问题为中心的课程设计聚焦在“学生如何开一个问题和如何找到解决问题的方案”，问题为中心的课程设计增加了课程之间的相关性，允许学生在学习时更具创造性和创新性，缺点是并不总是考虑到学习的方式 (does not always take learning styles into consideration).

### 17.9.2.4 课程设计时的几点注意事项

- 搞清学生需求, 可对学生的需求进行分析保留收集分析学习者的相关数据, 比如学习者在某些领域或者相关技能已经知道的、需要知道的, 也可以包括学习者的认知、优缺点 (perception, strengths and weaknesses).
- 建立一个清晰的学习目标 (learning goals) 和产出 (learning outcomes) 列表, 学习目标是老师想让学生在课程中达到的结果, 学习产出是学生在课程学习后应该达到的、可检测的知识、技能和态度 (measurable knowledge, skills, and attitudes that students should have achieved in the course)
- 搞清限制条件, 要明确影响课程设计的因素, 比如课程时间等。
- 考虑建立一个课程地图, 通常也称为课程矩阵。
- 明确教学方法, 教学方法应该考虑如何和学生的学生类型共同起作用, 如果教学方法无益于课程, 就需要进行更换。
- 建立不同的评估方法, 在过程结束或者过程中需要评估学习者、教师和课程, 评估帮助你确定课程设计是否起作用或者没有, 还有就是评估课程输出的达成度, 最有效的评估是持续的和累计的 (ongoing and summative)。
- 谨记课程设计不是一蹴而就, 需要持续改进。

## 17.10 The Vernam Cipher

Vernam 是 AT&T 公司设计的一次一密系统, 下面图17.5是<https://cryptomuseum.com/crypto/vernam.htm>关于 Vernam 密码机的资料。

## 17.11 工具 RSA-Tool 2 中的公私钥生成方法

RSA Tool 2 是互联网上一个广为流程的 RSA 工具, 界面如17.6。其在软件说明中, 对于密钥生成的描述如下:

### 2. Parameters

P = 1st large prime number

Q = 2nd large prime number (sizes of P and Q should not differ too much!)





[Twitter](#) [YouTube](#) [Search](#)

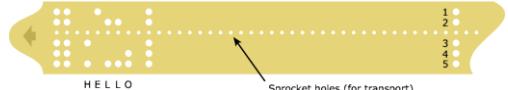
[Crypto](#) [Mixer](#) [OTT](#)

## The Vernam Cipher

Digital bit-wise XOR

The Vernam Cipher is based on the principle that each **plaintext** character from a message is 'mixed' with one character from a **key stream**. If a truly **random** key stream is used, the result will be a truly 'random' **ciphertext** which bears no relation to the original **plaintext**. In that case the cipher is similar to the unbreakable **One-Time Pad (OTP)**. As it was generally used with **teleprinters** and **5-level punched tape**, the system is also known as **One-Time Tape** or **OTT**.

If the resulting **ciphertext** in the OTT system described above is truly random, it can safely be sent over the air, without the risk of being deciphered by an eavesdropper. All the recipient has to do is mix the ciphertext with the same **OTT** to reveal the original **plaintext**. One only has to guarantee that the OTT is truly random, that there are only two copies of it, that both copies are destroyed immediately after use and that they are only used once. (More about security [below](#).)



The above became possible after the introduction of digital telegraphy, also known as **Teletype** <sup>1</sup> or **Telex**. With a teletypewriter, each character is substituted by a digital 5-bit code – represented by the 5 holes in a punched paper tape – commonly used with telex machines. This is commonly known as **ITA2** or the **Baudot–Murray code**. Digital codes can also be represented by a series of '1's and '0's, where 1 represents the presence of a hole and 0 represents the absence of a hole.

The **ciphertext** is generated by applying the logical **XOR** operation (exclusive-or) to the individual bits of **plaintext** and the **key stream**. The advantage of using the **XOR** operation for this, is that it can be reverted, simply by carrying out the same operation again. In other words:

$$\text{plaintext} + \text{key} = \text{ciphertext} \Rightarrow \text{ciphertext} + \text{key} = \text{plaintext}$$

In mathematics, the **XOR** operation is known as **modulo-2 addition**. In our case, the individual bits of the **plaintext** are **XOR-ed** with the individual bits of the **key**. The resulting bit will only be '1' if the two input bits are different. If they are equal (both 1 or both 0), the result will be '0'.

A 00011	B 11001	G 11010
B 11001	+ B 11001	+
G 11010		A 00011

Take the letter 'A', which is represented by 00011, and add it to the letter 'B', represented by 11001. A bit-wise XOR operation yields 11010 which, in the **ITA2 table**, is the letter 'G'. In fact, each bit from the key tells us whether or not the corresponding bit from the plaintext should be inverted. By inverting these key-bits again, as shown above, the original character is revealed.

1. Although 'Teletype' is actually a brand name of the Teletype Corporation, it has become a generic expression for **digital 5-bit telegraphy**. The system is also known as Teleprinter, Teletypewriter and Telex.

2. XOR = Exclusive OR.

### Example

The principle of the Vernam Cipher is perhaps easier understood by looking at a message stored on a punched paper tape. In the example below, we want to transmit the word **HELLO** which is stored on the plain text tape. We also have a pre-recorded **key tape**, with a series of random characters; in this case the sequence **A X H J B**. The contents of the **plaintext** tape are now **XOR-ed** with the contents of the **key tape**. The result (**K M I V E**) is shown here as the **ciphertext** tape:

图 17.5: Vernam 密码机介绍

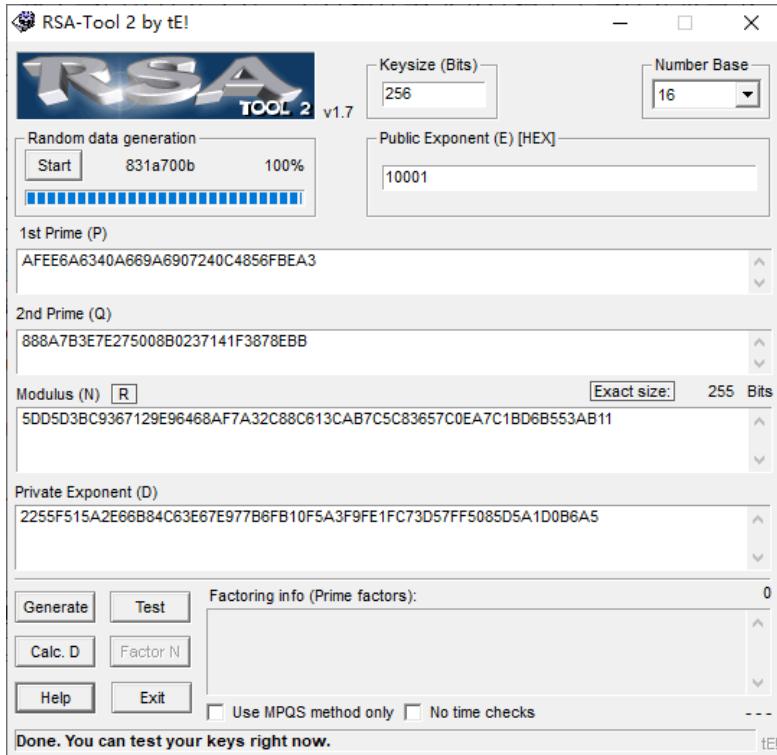


图 17.6: RSA Tool 2 生成公私钥的界面

E = Public Exponent (a random number which must fulfil:

$$\text{GCD}(E, (P-1)*(Q-1)) == 1$$

N = Public Modulus, the product of P and Q:  $N = P * Q$

D = Private Exponent:  $D = E^{( - 1)} \bmod ((P - 1) * (Q - 1))$

Parameters N and E are public whereas D is -private- and must NEVER be published! P and Q are not longer needed after keygeneration and should be destroyed.

To obtain D from the public key (N, E) one needs to try splitting N in its both prime factors P and Q. For a large Modulus N (512 bit and more) with carefully chosen primefactors P and Q this is a very difficult problem.

All the security of the RSA encryption scheme relies on that integer factorization problem (tough there's no mathematical proof for it). To find out more read here:

<http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa155.html>

### 3. Encryption

To encrypt a messageblock (M) (which must be < N), compute:

$$\text{Ciphertext} = C = M^E \bmod N.$$

Note: If the entire message (M) is > N it must be split into smaller blocks with size < N

### 4. Decryption

To decrypt a given Ciphertext (C) to retrieve the Plaintext (M) as result, compute:  $M = C^D \bmod N$ .

The '<sup>d</sup>' sign in the above equations means 'power of', not XOR !

Note that the RSA scheme does also work the other way round:

$C = M^D \text{mod} N$  and  $M = C^E \text{mod} N$ . It's on you how you implement it. Just ALWAYS make SURE that you -NEVER- publish the private exponent D, P and//or Q !

### 5. How to ...?

...Generate a RSA keypair ?

1) Press the 'Start' Button to collect some (pseudo) random data by moving around your mousepointer.

This must be done only one time, because the collected data will be saved in a file in your RSA-Tool folder.

2) Select the number base you want to use for your keys. Base 10, 16, 60 and 64 are available.

3) Select the size of of the key (=size of N) you want to create. Max. keysize is 4096 bit.

4) Choose your public exponent (E) and type it in the corresponding Editbox as DECIMAL number. Most common values (calculation-speed reasons) used for E are: 3, 17, 257 and 65537

(decimal) 5) Press the Generate button and wait until keygeneration has been finished. Note that generation of very large keys can take some time, depending on the power of your CPU.

Important: You can press Generate as often as you like. The internal random number generation system, which is used in a part of the key generation process, will be re-initialized during runtime.

This is done on purpose, as it makes it much harder to abuse this tool for certain things...Note that this also makes it almost \*impossible\* to create the same keypair twice or more.

It can happen that your Modulus will become e.g. 159 Bits only, even when you selected 160 Bits as keysize. The reason is the little size difference between P and Q. If you are not satisfied, simply press the Generate button again until the desired keylength meets your needs :-)

## 参考文献

---

- [1] 戚征. 伪随机序列 [J]. 数学的实践与认识, 1972(4): 29-48.
- [2] 卿斯汉. 密码学与计算机网络安全 [M]. 中国: 清华大学出版社, 广西科学技术出版社, 2002.
- [3] 李晓峰. 网络空间安全数学基础讲义 [M]. 中国: 开源书, [http://buuer\\_xxtxiaoeng.gitee.io/lxf/CSSMathFound.html](http://buuer_xxtxiaoeng.gitee.io/lxf/CSSMathFound.html), 2019.
- [4] 冯登国. 计算机通信网络安全 [M]. 中国: 清华大学出版社, 2004 年 7 月第 3 次印刷.
- [5] Klein, 徐秋亮、蒋瀚、王皓(译)Philip N. 密码学基础教程——秘密与承诺 (A Cryptography Primer——Secrets and Promises)[M]. 中国: 机械工业出版社, 2017 年 12 月第 2 次印刷.
- [6] Shannon C E. Communication theory of secrecy systems[J/OL]. The Bell System Technical Journal, 1949, 28 (4): 656-715. DOI: [10.1002/j.1538-7305.1949.tb00928.x](https://doi.org/10.1002/j.1538-7305.1949.tb00928.x).
- [7] 【日】结城浩. 图解密码技术 [M]. 中国: 中信出版社, 2017.
- [8] BONEH D, SHOUP V. A graduate course in applied cryptography[M]. 中国: Open Book, Shared in internet, 2020.
- [9] DENNING D E R. Cryptography and data security[M]. 中国: Addison-Wesley Publishing Company, 1982.
- [10] Diffie W, Hellman M. New directions in cryptography[J/OL]. IEEE Transactions on Information Theory, 1976, 22(6): 644-654. DOI: [10.1109/TIT.1976.1055638](https://doi.org/10.1109/TIT.1976.1055638).
- [11] SHAMIR A. On the cryptocomplexity of knapsack systems[J]. Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing, 1979, 29: 118-129.
- [12] MAO W. Modern cryptography: Theory and practice[M]. 中国: Prentice Hall Professional Technical Reference, 2003.
- [13] ANDRAIU M, SIMION E. Evaluation of cryptographic algorithms[J]. Romanian Economic Business Review, 2011, 5: 52-62.
- [14] Bruce Schneier(美) 吴世忠等译. 应用密码学: 协议、算法与 C 源程序 (原书第 2 版)[M]. 中国: 电子工业出版社, 2017.
- [15] GOLOMB S W. Shift register sequences: Secure and limited-access code generators, efficiency code generators, prescribed property generators, mathematical models[M]. 3rd ed. USA: World Scientific Publishing Co., Inc., 2017.
- [16] 肖国震. 伪随机序列及其应用 [M]. 中国: 国防工业出版社, 1985.
- [17] 杨波. 现代密码学 (第四版)[M]. 中国: 清华大学出版社, 2019 年 1 月第 5 次印刷.
- [18] 李浪, 邹祐, 郭迎. 密码工程学 [M]. 北京: 清华大学出版社, 2014.
- [19] 百度百科. 比特币 [EB/OL]. 2020. <https://baike.baidu.com/item/%E6%AF%94%E7%89%B9%E5%B8%81>.
- [20] 冯登国. 可信计算——理论与实践 [M]. 中国: 清华大学出版社, 2013 年 5 月.
- [21] 刘镇严波涛. 一种无可信第三方的智力扑克协议 [J]. 计算机应用, 2009, 29: 1836-1838.
- [22] 【英】斯蒂芬. 破译者 [M]. 中国: 商务印刷, 2017.
- [23] 张换国. 密码学引论 [M]. 中国: 武汉大学出版社, 2017.
- [24] 张文政. 密码学的基本理论与技术 [M]. 中国: 国防出版社, 2017.
- [25] 刘建伟. 网络安全——技术与实践 [M]. 中国: 清华大学出版社, 2017.
- [26] 贾春福. 信息安全数学基础 [M]. 中国: 机械工业出版社, 2017.
- [27] NARAYANAN A, CLARK J. Bitcoin's academic pedigree[J/OL]. Commun. ACM, 2017, 60(12): 36-45. <https://doi.org/10.1145/3132259>.
- [28] 聂旭云. 现代密码学 (中国 MOOC)[EB/OL]. 2020. <https://www.icourse163.org/course/UESTC-1003046001>.
- [29] BARAK B. An intensive introduction to cryptography[EB/OL]. 2020. <https://intensecrypto.org/public/index.html>.

- [30] SHANNON C E. A symbolic analysis of relay and switching circuits[M]. [S.l.: s.n.], 1938.
- [31] Shannon C E. A mathematical theory of communication[J/OL]. The Bell System Technical Journal, 1948, 27(3): 379-423. DOI: [10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x).

## 附录 A 练习

---

请看单独的练习文档 CryExercise.pdf。

## 附录 B 课程参考分值分配

---

教师对学生在此门课程上的期望，具体体现在教师对此课程上的评价设计上，落实在评分上。此课程评价指导思想是，在了解基本理论的基础上，能够应用，同时重点放在过程环节的评价，课程参考分值分配见表B.1。

表 B.1: 课程参考分值分配

考评项	所占总分数
考勤	10
作业 + 单元测验	30
编程练习	30
期末考试	30

## 附录 C 相关机构

### C.1 国家密码管理局

国家密码管理局的职责时组织贯彻落实党和国家关于密码工作的方针、政策，研究提出解决密码工作发展中重大问题的建议；拟订密码工作发展规划，起草密码工作法规并负责密码法规的解释，组织拟订密码相关标准；依法履行密码行政管理职能，管理密码科研、生产、装备（销售），测评认证及使用，查处密码失泄密事件和违法违规研制、使用密码行为，负责有关密码的涉外事宜；对密码工作部门实施业务领导；负责网络与信息系统中密码保障体系的规划和管理，规划、建设和管理国家密码基础设施；指导密码专业教育和密码学术交流，组织密码专业人才教育培训，对高等院校、科研机构、学术团体开展密码基础理论与应用技术研究、交流进行指导。



图 C.1：国家密码管理局官网主页 (2020 年 3 月 5 日)

### C.2 密码行业标准化技术委员会

为满足密码领域标准化发展需求，充分发挥密码科研、生产、使用、教学和监督检验等方面专家作用，更好地开展密码领域的标准化工作，2011年10月，经国家标准化管理委员会和国家密码管理局批准，成立“密码行业标准化技术委员会”（以下简称“密标委”），英文名称“Cryptography Standardization Technical Committee”（简称“CSTC”）。密标委是在密码领域内从事密码标准化工作的非法人技术组织，归口国家密码管理局领导和管理，主要从事密码技术、产品、系统和管理等方面的标准工作。密标委委员由政府、企业、科研院所、高等院校、检测机构和行业协会等有关方面的专家组成。密标委目前下设秘书处和总体、基础、应用、测评四个工作组。



图 C.2: 密码行业标准化技术委员会网站主页 (2020 年 3 月 5 日)

CSTC 主要任务就是汇集资源，制定密码相关标准，在其网站上可以找到此委员会制定的相关标准，可以免费查看。

行标号	标准名称	类别	状态	英文名称	牵头单位	合作单位	发布	实施	上升国际	推荐中
GM/T 0001-2012	祖冲之序列密码算法：第1部分：算法描述	现行	推荐性 GM/T	ZUC stream cipher algorithm-Part 1:Description of the algorithm	中国科学院软件研究所	中国科学院软件研究所、中国科学院数据与通信保护研究教育中心	20120321	20120321	已上升为国际	文件查看
GM/T 0001-2-2012	祖冲之序列密码算法：第2部分：基于祖冲之算法的机密性算法	现行	推荐性 GM/T	ZUC stream cipher algorithm-Part 2:The ZUC-based confidentiality algorithm	中国科学院软件研究所	中国科学院软件研究所、中国科学院数据与通信保护研究教育中心	20120321	20120321	已上升为国际	文件查看
GM/T 0001-3-2012	祖冲之序列密码算法：第3部分：基于祖冲之算法的完整性算法	现行	推荐性 GM/T	ZUC stream cipher algorithm-Part 3:The ZUC-based integrity algorithm	中国科学院数据与通信保护研究教育中心	中国科学院软件研究所、中国科学院数据与通信保护研究教育中心	20120321	20120321	--	文件查看
GM/T 0002-2012	SM4分组密码算法	现行	推荐性 GM/T	SM4 block cipher algorithm	中国科学院数据与通信保护研究教育中心	中国科学院数据与通信保护研究教育中心、国家密码管理局商用密码检测中心	20120321	20120321	已上升为国际	文件查看

图 C.3: 密码行业标准化技术委员会网站主页

## C.3 商用密码检测中心

商用密码检测中心（以下简称检测中心）是国内权威的商用密码检测机构，主要职责包括：商用密码产品密码检测、信息安全产品认证密码检测、含有密码技术的产品密码检测、信息安全等级保护商用密码测评、商用密码行政执法密码鉴定、国家电子认证根 CA 建设和运行维护、密码技术服务、商用密码检测标准规范制订等。检测中心经过多年的建设，配备了各类先进的专业检测设备和测试工具，先后获得中国合格评定国家认可委员会的实验室认可（证书编号：CNAS L2109）和中国国家认监委认可监督委员会的计量认证认定（证书编号：2013002730Z），并由中国国家认监委指定为第一批信息安全产品认证检测实验室。检测中心承担并完成了多项国家和部门重要科研课题，参与研制的数十项技术标准已作为国家标准或密码行业标准正式发布实施，

获得国家技术发明二等奖一项，国家科技进步二等奖两项，省部科技进步奖十余项。

网址为<http://www.scctc.org.cn/index.aspx>



图 C.4: 商用密码检测中心的 LOGO

在商用密码检测中心网址上有些检测标准和算法资源。

The screenshot shows the website's navigation bar with links for Home, About Us, Submission Process, Testing Standards, Testing Results Release, Download Center, Contact Us, and Job Recruitment. Below the navigation bar is a banner with the text "Commercial Cryptography Testing center" and "同心思远 守正敏行". The main content area is titled "Algorithm Implementation Resources" and includes sections for "Source Code Download" and "License Statement". It lists several algorithm implementations with download links: SM2 Algorithm Source Code (发布日期: 2017-05-05), SM3 Algorithm Source Code (发布日期: 2017-05-05), SM4 Algorithm Source Code (发布日期: 2017-05-05), SM9 Algorithm Source Code (发布日期: 2017-05-05), and ZUC Algorithm Source Code (发布日期: 2017-05-05). At the bottom, there are navigation buttons for page numbers.

图 C.5: 商用密码检测中心的参考算法实现资源

The screenshot shows three separate sections of the website's "Download Center" for "Information Security Products". Each section has tabs for "Testing Guidelines", "Testing Standard Overview", and "Other". The first section lists "Security System Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), and "Security Product Testing Guidelines" (发布日期: 2016-06-14), each with a "Download" button. The second section lists "Security System Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), and "Security Product Testing Guidelines" (发布日期: 2016-06-14), each with a "Download" button. The third section lists "Security Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), "Security Product Testing Guidelines" (发布日期: 2016-06-14), and "Security Product Testing Guidelines" (发布日期: 2016-06-14), each with a "Download" button.

图 C.6: 商用密码检测中心的相关文档资源

## C.4 中国密码学会

中国密码学会是由密码学及相关领域的科技工作者和单位自愿结成并依法登记的全国性、学术性、非营利性法人社会团体，是中国科协组成部分。经民政部批准于 2007 年 3 月成立，业务主管单位中国科协，挂靠单位国家密码管理局。设有 14 个分支机构：即组织、学术、教育与科普、青年、密码应用工作委员会和量子密码、密码数学理论、密码芯片、密码算法、电子认证、安全协议、混沌保密通信、密码测评专业委员会，商用密码应用安全性评估联委会。学会创办了《密码学报》(CN10—1195/TN)，中文，双月刊。



图 C.7: 中国密码学会官网主页 (2020 年 3 月 5 日)

## 附录 D 相关法律、条例、标准

---

### D.1 基本概念

法律是由国家制定或认可并以国家强制力保证实施的，反映由特定物质生活条件所决定的统治阶级意志的规范体系。法律是统治阶级意志的体现，是国家的统治工具。法律是由享有立法权的立法机关行使国家立法权，依照法定程序制定、修改并颁布，并由国家强制力保证实施的基本法律和普通法律总称。法律是法典和律法的统称，分别规定公民在社会生活中可进行的事务和不可进行的事务。法律可以划分为：（1）宪法；（2）法律；（3）行政法规；（4）地方性法规；（5）自治条例和单行条例。法律是从属于宪法的强制性规范，是宪法的具体化。宪法是国家法的基础与核心，法律则是国家法的重要组成部分。<sup>1</sup>

条例是国家权力机关或行政机关依照政策和法令而制定并发布的，针对政治、经济、文化等各个领域内的某些具体事项而作出的，比较全面系统、具有长期执行效力的法规性公文。条例是法的表现形式之一。一般只是对特定社会关系作出的规定。条例是由国家制定或批准的规定某些事项或某一机关组织、职权等规范性的法律文件，也是指团体制定的章程。它具有法的效力，是根据宪法和法律制定的，是从属于法律的规范性文件，人人必须遵守，违反它就要带来一定的法律后果。<sup>2</sup>

规章是各级领导机关及其职能部门、社会团体、企事业单位，为实施管理，规范工作、活动和有关人员行为，在其职权范围内制定并发布实施的、具有行政约束力和道德行为准则的规范性文书的总称。<sup>3</sup>

管理制度是组织、机构、单位管理的工具，对一定的管理机制、管理原则、管理方法以及管理机构设置的规范。它是实施一定的管理行为的依据，是社会再生产过程顺利进行的保证。合理的管理制度可以简化管理过程，提高管理效率。<sup>4</sup>

### D.2 中华人民共和国网络安全法

《中华人民共和国网络安全法》是为保障网络安全，维护网络空间主权和国家安全、社会公共利益，保护公民、法人和其他组织的合法权益，促进经济社会信息化健康发展而制定的法律。《中华人民共和国网络安全法》由中华人民共和国第十二届全国人民代表大会常务委员会第二十四次会议于 2016 年 11 月 7 日通过，自 2017 年 6 月 1 日起施行。

下面是安全法中摘录的内容。

---

<sup>1</sup> 来自 <https://baike.baidu.com/item//84813>

<sup>2</sup> 来自 <https://baike.baidu.com/item/>

<sup>3</sup> 来自 <https://baike.baidu.com/item//13021988>

<sup>4</sup> 来自 <https://baike.baidu.com/item/>

第十条 建设、运营网络或者通过网络提供服务，应当依照法律、行政法规的规定和国家标准的强制性要求，采取技术措施和其他必要措施，保障网络安全、稳定运行，有效应对网络安全事件，防范网络违法犯罪活动，维护网络数据的完整性、保密性和可用性。

第二十一条 国家实行网络安全等级保护制度。网络运营者应当按照网络安全等级保护制度的要求，履行下列安全保护义务，保障网络免受干扰、破坏或者未经授权的访问，防止网络数据泄露或者被窃取、篡改：

(四) 采取数据分类、重要数据备份和加密等措施；

第三十六条 关键信息基础设施的运营者采购网络产品和服务，应当按照规定与提供者签订安全保密协议，明确安全和保密义务与责任。

## D.3 中华人民共和国密码法

《中华人民共和国密码法》是为了规范密码应用和管理，促进密码事业发展，保障网络与信息安全，维护国家安全和社会公共利益，保护公民、法人和其他组织的合法权益，制定的法律。是中国密码领域的综合性、基础性法律。《中华人民共和国密码法》由中华人民共和国第十三届全国人民代表大会常务委员会第十四次会议于 2019 年 10 月 26 日通过，自 2020 年 1 月 1 日起施行。

下面是密码法中摘录的内容。

第六条 国家对密码实行分类管理。

密码分为核心密码、普通密码和商用密码。

第七条 核心密码、普通密码用于保护国家秘密信息，核心密码保护信息的最高密级为绝密级，普通密码保护信息的最高密级为机密级。

核心密码、普通密码属于国家秘密。密码管理部门依照本法和有关法律、行政法规、国家有关规定对核心密码、普通密码实行严格统一管理。

第八条 商用密码用于保护不属于国家秘密的信息。

公民、法人和其他组织可以依法使用商用密码保护网络与信息安全。

第九条 国家鼓励和支持密码科学技术研究和应用，依法保护密码领域的知识产权，促进密码科学技术进步和创新。

国家加强密码人才培养和队伍建设，对在密码工作中作出突出贡献的组织和个人，按照国家有关规定给予表彰和奖励。

第十条 国家采取多种形式加强密码安全教育，将密码安全教育纳入国民教育体系和公务员教育培训体系，增强公民、法人和其他组织的密码安全意识。

第十一条 县级以上人民政府应当将密码工作纳入本级国民经济和社会发展规划，所需经费列入本级财政预算。

第十二条 任何组织或者个人不得窃取他人加密保护的信息或者非法侵入他人的密码保障系统。

任何组织或者个人不得利用密码从事危害国家安全、社会公共利益、他人合法权益等违法犯罪活动。

## D.4 中华人民共和国反恐怖主义法

《中华人民共和国反恐怖主义法》为了防范和惩治恐怖活动，加强反恐怖主义工作，维护国家安全、公共安全和人民生命财产安全，根据宪法制定。由中华人民共和国主席于 2015 年 12 月 27 日发布，2016 年 1 月 1 日起施行。

下面是反恐怖法中摘录的内容。

第十八条电信业务经营者、互联网服务提供者应当为公安机关、国家安全机关依法进行防范、调查恐怖活动提供技术接口和解密等技术支持和协助。

第四十八条反恐怖主义工作领导机构、有关部门和单位、个人应当对履行反恐怖主义工作职责、义务过程中知悉的国家秘密、商业秘密和个人隐私予以保密。违反规定泄露国家秘密、商业秘密和个人隐私的，依法追究法律责任。

第八十四条电信业务经营者、互联网服务提供者有下列情形之一的，由主管部门处二十万元以上五十万元以下罚款，并对其直接负责的主管人员和其他直接责任人员处十万元以下罚款；情节严重的，处五十万元以上罚款，并对其直接负责的主管人员和其他直接责任人员，处十万元以上五十万元以下罚款，可以由公安机关对其直接负责的主管人员和其他直接责任人员，处五日以上十五日以下拘留：

(一) 未依照规定为公安机关、国家安全机关依法进行防范、调查恐怖活动提供技术接口和解密等技术支持和协助的；

## D.5 中华人民共和国电子签名法

《中华人民共和国电子签名法》是为了规范电子签名行为，确立电子签名的法律效力，维护有关各方的合法权益而制定的法律。《中华人民共和国电子签名法》由中华人民共和国第十届全国人民代表大会常务委员会第十一次会议于 2004 年 8 月 28 日通过，自 2005 年 4 月 1 日起施行。当前版本为 2019 年 4 月 23 日第十三届全国人民代表大会常务委员会第十次会议修正。

下面是电子签名法中摘录的内容。

第十三条电子签名同时符合下列条件的，视为可靠的电子签名：

- (一) 电子签名制作数据用于电子签名时，属于电子签名人专有；
- (二) 签署时电子签名制作数据仅由电子签名人控制；
- (三) 签署后对电子签名的任何改动能够被发现；
- (四) 签署后对数据电文内容和形式的任何改动能够被发现。

当事人也可以选择使用符合其约定的可靠条件的电子签名。

第十四条可靠的电子签名与手写签名或者盖章具有同等的法律效力。

第十五条电子签名人应当妥善保管电子签名制作数据。电子签名人知悉电子签名制作数据已经失密或者可能已经失密时，应当及时告知有关各方，并终止使用该电子签名制作数据。

第十六条电子签名需要第三方认证的，由依法设立的电子认证服务提供者提供认证服务。

第十七条提供电子认证服务，应当具备下列条件：

- (一) 取得企业法人资格；
- (二) 具有与提供电子认证服务相适应的专业技术人员和管理人员；
- (三) 具有与提供电子认证服务相适应的资金和经营场所；
- (四) 具有符合国家安全标准的技术和设备；
- (五) 具有国家密码管理机构同意使用密码的证明文件；
- (六) 法律、行政法规规定的其他条件。

## D.6 国家商用密码管理条例

《国家商用密码管理条例》是为了加强商用密码管理，保护信息安全，保护公民和组织的合法权益，维护国家的安全和利益，制定本条例。条例与 1999 年 10 月 7 日由国务院颁布同时生效。

以下是条例部分内容。

第七条商用密码产品由国家密码管理机构指定的单位生产。未经指定，任何单位或者个人不得生产商用密码产品。商用密码产品指定生产单位必须具有与生产商用密码产品相适应的技术力量以及确保商用密码产品质量的设备、生产工艺和质量保证体系。

第八条商用密码产品指定生产单位生产的商用密码产品的品种和型号，必须经国家密码管理机构批准，并不得超过批准范围生产商用密码产品。

第九条商用密码产品，必须经国家密码管理机构指定的产品质量检测机构检测合格。

第十条商用密码产品由国家密码管理机构许可的单位销售。未经许可，任何单位或者个人不得销售商用密码产品。

第十一条销售商用密码产品，应当向国家密码管理机构提出申请，并应当具备下列条件：

- (一) 有熟悉商用密码产品知识和承担售后服务的人员；
- (二) 有完善的销售服务和安全管理规章制度；
- (三) 有独立的法人资格。

经审查合格的单位，由国家密码管理机构发给《商用密码产品销售许可证》。

## D.7 《信息安全技术信息系统密码应用基本要求》国家标准

2021年3月9日，《信息安全技术信息系统密码应用基本要求》(GB/T 39786-2021)正式发布，并于2021年10月1日起实施。“基本要求”从行业标准上升为国家标准，是商用密码应用与安全性评估工作的重要里程碑，对促进我国密码事业发展，规范密码应用，具有重要意义。

密码技术作为网络安全的基础核心技术，是信息保护和网络信任体系建设的基础，是保障网络空间安全的关键技术。《信息安全技术信息系统密码应用基本要求》(GB/T 39786-2021)标准，适用于指导、规范信息系统密码应用的规划、建设、运行、测评。在“基本要求”基础上，各个领域与行业可以结合本领域行业的密码应用需求来指导、规范信息系统密码应用。

GB/T 39786-2021标准与等保测评相衔接，定级对应等保定级，体现了对信息安全以整体的思路全方位防护的基本理念。一下内容摘取字自标准。

本标准从信息系统的物理和环境安全、网络和通信安全、设备和计算安全、应用和数据安全四个层面提出密码应用技术要求，保障信息系统的实体身份真实性、重要数据的机密性和完整性、操作行为的不可否认性；并从信息系统的管理制度、人员管理、建设运行和应急处置四个方面提出密码应用管理要求，为信息系统提供管理方面的密码应用安全保障。

## 附录 E 代数基础 (Algrbra)

---

### E.1 群、环、域

**定义 E.1 (群 (group))**  $G$  是一个非空集合,  $*$  是定义在集合  $G$  上的一个二元运算,  $(G, *)$  被称为群, 如果  $(G, *)$  满足以下条件:

- (1) 单位元 (identity element):  $\exists e \in G, \forall a \in G \Rightarrow e * a = a$ , 此时我们称  $e$  为  $G$  的左幺元 (member of the upper-left)。
- (2) 逆元 (inverse element):  $\forall a \in G, \exists a' \in G, a' * a = e$ , 此时我们称元素  $a'$  为  $a$  的左逆元 (left inverse element)。
- (3) 封闭 (closure): 对于任意  $a, b \in G \Rightarrow a * b \in G$ .
- (4) 结合律 (associative): 对于任意  $a, b, c \in G \Rightarrow a * (b * c) = (a * b) * c$ .

**定义 E.2 (环 (ring) 和交换环 (communicative ring))** 设  $R$  是一个给定的集合, 在其上定义了两种二元运算  $+, \circ$ , 且满足以下条件:

- (1)  $(R, +)$  是一个交换群。
- (2)  $(R, \circ)$  是一个半群。
- (3)  $a, b, c \in R, a \bullet (b + c) = (a \bullet b) + (a \bullet c)$ 。

我们称  $(R, +, \circ)$  为环, 若  $(R, \circ)$  是一个交换群, 则称其为交换环。

**定义 E.3 (域 (field)[12])** 如果一个环中的非零元素与乘法运算结合, 形成一个群, 那么这个环叫做一个域。

### E.2 有限域的结构

#### E.2.1 有素数元素的有限域

**定义 E.4 (素域 (prime field)[12])** 无真子域 (no proper subfield) 的域称为素域。

**定义 E.5 (素数阶的有限域 (finite field of prime order)[12])** 设  $p$  是任一素数, 整数模  $p$  是一个  $p$  阶有限域, 记为  $\mathbb{Z}_p, \mathbb{Z}_p$  的非零元素组成一个乘法群, 记为  $\mathbb{Z}_p^*$ 。通常也用符号  $\mathbb{F}_p$  表示  $\mathbb{Z}_p$ 。

**定义 E.6 (代数结构的特征 (Characteristic of an Algebraic Struture)[12])** 一个代数结构  $A$  的特征是这样一个最小正整数  $n$ , 对于每一个  $a \in A$ , 都有  $na = \nu$ , 如果不存在这样一个正整数, 那么称此代数结构特征为 0, 特征记为  $char(A)$ 。

**定义 E.7 (有限域的特征 [12])** 每一个有限域都有一个素数特征。

## E.2.2 有限域模不可约多项式

### E.2.2.1 一个代数机构的多项式

**定义 E.8 (代数结构上的多项式 (Polynomials over an algebraic structure)[12])** 设  $\mathbb{A}$  是一个代数结构, 有  $+, \times$  两种运算, 形如以下的表达式, 称为在代数结构  $\mathbb{A}$  的多项式。

$$f(x) = \sum_{i=0}^n a_i x^i$$

$n$  是非负整数, 系数  $a_i$  是  $\mathbb{A}$  中元素。我们把  $\mathbb{A}$  上的所有多项式组成的集合记为  $\mathbb{A}[x]$ .

**定义 E.9 (不可约多项式 (irreducible polynomial)[12])**  $f \in \mathbb{A}[x]$ , 如何  $f$  可以表示为  $f = gh, g \in \mathbb{A}[x], h \in \mathbb{A}[x]$ , 则称  $f$  是在  $\mathbb{A}[x]$  上的可约多项式, 反之称  $f$  为  $\mathbb{A}[x]$  上的不可约多项式。

同一形式的多项式, 可能在另一个代数结构上可约, 在另外一个代数结构上不可约。

### E.2.2.2 用不可约多项式构造域

**定义 E.10 (多项式模 [12])** 代数结构  $\mathbb{A}, f, g, q, r \in \mathbb{A}[x], g \neq 0$ , 满足  $f = qg + r$ , 我们称  $r$  是  $f$  除以  $g$  的余数, 记为  $r = f \pmod{g}$ ,  $\mathbb{A}[x]$  中所有多项式模  $g$  的余数组成的集合记为  $\mathbb{A}[x]_g$ 。

**定理 E.1 (多项式域 [12])**  $F$  是一个域,  $f$  是  $F[x]$  上的一个非零多项式, 如果  $f$  是一个不可约多项式, 那么  $F[x]_f$  是一个环, 并且是一个域。

**定理 E.2 (多项式域的元素个数 [12])** 素数  $p$  形成域  $F$ ,  $f$  是  $F$  上的  $n$  阶不可约多项式, 那么域  $F[x]_f$  的元素个数为  $p^n$ 。

## E.2.3 用多项式基构造有限域

**定理 E.3 (线性独立 [12])**  $F$  是一个有限域,  $f(x) \in F[x]$ ,  $f(x)$  是一个  $n$  阶不可约多项式,  $\theta$  是  $f(x) = 0$  的一个根, 元素  $1, \theta, \theta^2, \dots, \theta^{n-1}$  称为在域  $F$  上线性独立, 这意味着, 只有  $r_0 = r_1 = \dots = r_{n-1} = 0$  时,  $r_0 + r_1\theta + r_2\theta^2 + \dots + r_{n-1}\theta^{n-1} = 0$  才能成立。

**定义 E.11 (多项式基 (polynomial basis)[12])**  $F$  是一个有限域,  $f(x) \in F[x]$ ,  $f(x)$  是一个  $n$  阶不可约多项式,  $\theta$  是  $f(x) = 0$  的任意一个根, 元素  $1, \theta, \theta^2, \dots, \theta^{n-1}$  称为域  $F$  的多项式基。