Computer Networks

Chapter 8
Network Security

主要内容

8.1 网络安全概述

8.2 *密码学*

8.3 *对称密钥算法*

8.4 *公开密钥算法*

8.5 *数字签名*

8.6 公钥的管理

8.7 *通信安全*

8.8 计算机网络病毒

8.1 网络安全概述（1）

- 网络上敏感信息日益增多
  - 军事机密
  - 政府机要
  - 商业机密
  - 个人电子帐户、电子购物、电子纳税
  - 个人隐私
  - …

8.1 网络安全概述（2）

- 引发安全问题的人群和目标

攻击者	目标
学生	喜欢窥探别人的电子邮件
破坏者（craker）	考验某人的安全系统，偷取数据
销售代表	声称可以代表整个欧洲，而不仅是安道尔
商人	想找到竞争对手的市场计划策略
离职雇员	被解雇之后实施报复
会计	挪用公司钱款
股票经纪人	否认自己曾用电子邮件对顾客作过承诺
骗子	头取信用卡号码再转卖
间谍	了解敌方的军事或者工业机密
恐怖分子	偷取细菌战机密

8.1 网络安全概述（3）

- 网络安全问题
  - 保密—又称为机密，任务是确保信息不会被未授权的用户访问
  - 认证—是指当你在展示敏感信息或者进入商务交易前你必须要确定在跟谁通话
  - 不可否认—牵涉到签名，指你发出信息之后从技术上说不能够否认这条信息不是你发出的
  - 完整性控制—确保信息的传输途中未遭到任何改变

8.1 网络安全概述（4）

- 安全涉及协议栈的每一层
  - 物理层将传输线封装在内含高压气体的密封管线中可以对付搭线窃听
  - 数据链路层 采用点到点链路加密（link encryption），
    - 缺点一是易受来自路由器内部的攻击
    - 缺点二是必须对所有的应用实施，缺乏灵活性
  - 网络层采用防火墙进行分组过滤
    - IPSec
  - 传输层端到端的加密
  - 应用层解决用户认证和不可否认性这样的问题

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.2 密码学

8.2.1 密码学简介

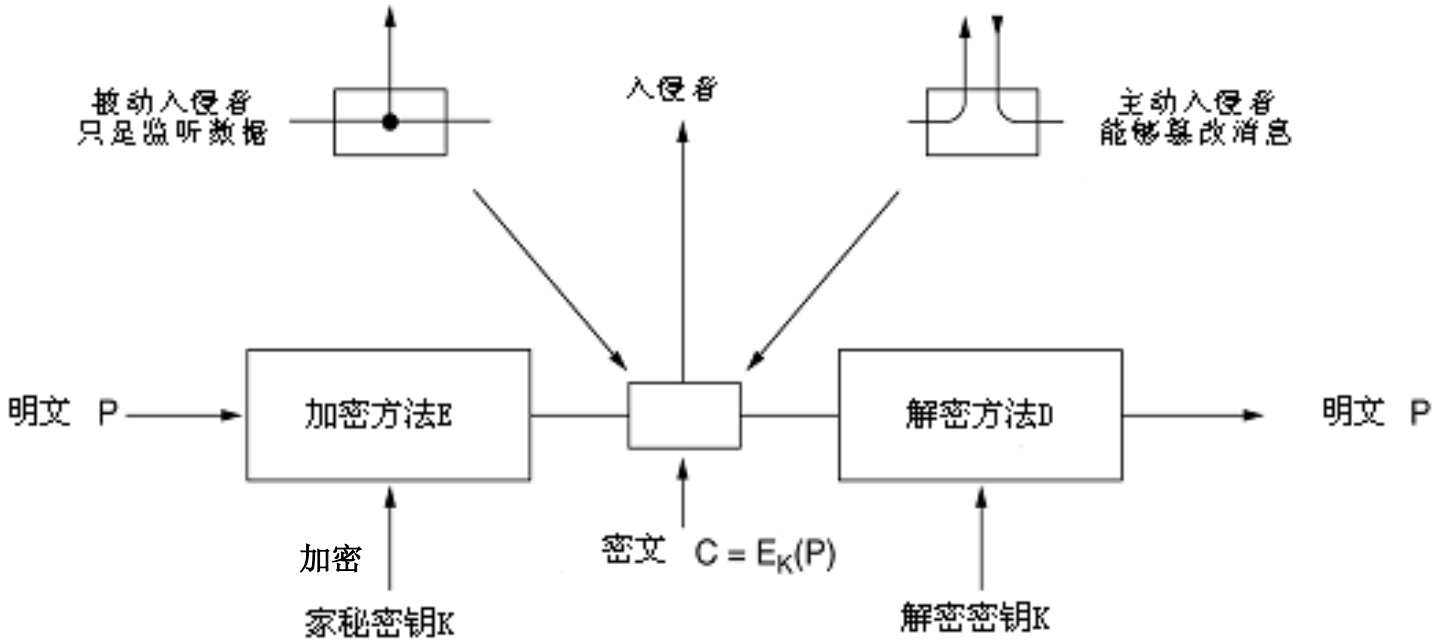
8.2.2 置换密码

8.2.3 转置密码

8.2.4 一次一密

8.2.5 两条密码学原则

8.2.1 密码学简介（1）



- 待加密的消息为明文（plaintext），它经过一个以密钥（key）为参数的函数变换，这个过程称为加密，输出的结果称为密文（ciphertext）
- 设计加密和解密方法的艺术称为密码编码学，破解密码的艺术称为密码分析学，两者统称为密码学

8.2.1 密码学简介（2）

- 符号标记
  - 明文P；密钥K；密文C
- 加密过程
  - C = Dk (P)
- 解密过程
  - P = Ek (C)
- Kerckhoff原则：“让密码分析者知道加解密算法，把所有的秘密信息全放在密钥中”
- 密钥长度越长，破解难度越大！

8.2.1 密码学简介（3）

- 密码分析方法
  - 只有密文攻击
    - 密码分析者只得到了一定量的密文，但是没有对应的明文
  - 已知明文攻击
    - 密码分析者有一些相匹配的密文和明文
  - 选择明文攻击
    - 密码分析者能够加密某些他自己选择的明文
- 密码必须能抵抗上述三种攻击！

8.2.2 置换密码（1）

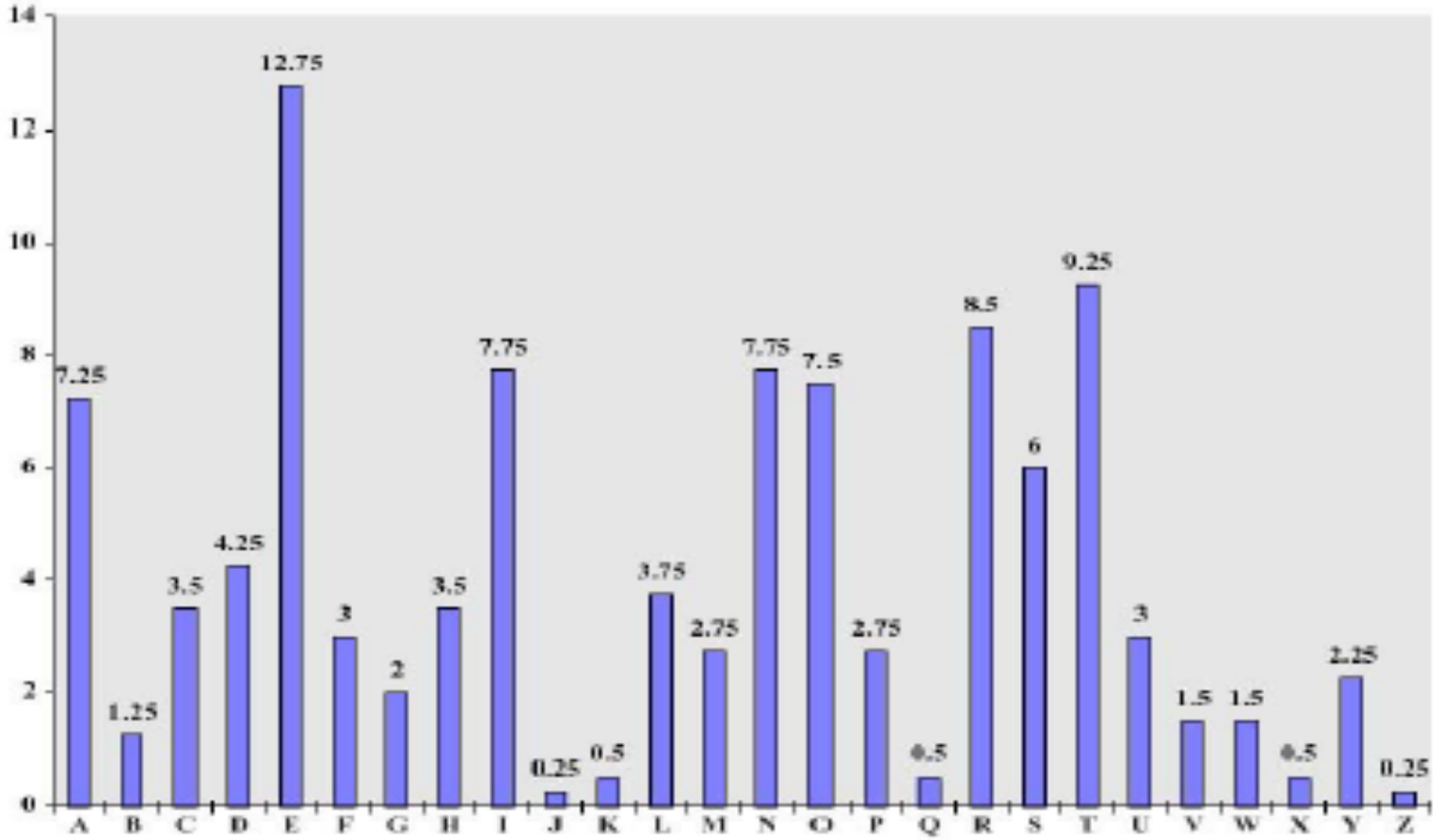
- 置换密码（substitution cipher）
  - 每个字母或者每一组字母被另一个字母或者另一组字母取代
  - 不改变明文字母的顺序，但是把字母伪装起来
- 凯撒密码（Caesar cipher）
  - 每一个字母被位于其后的第三个字母代替
    - 一个例子：attack -> dwwdfn
  - 一个通用方案是允许字母表被移动k个字母而不是3个，此时k相当于字母表循环移动算法的密钥
  - 优点：简单；缺点：极其容易破译

8.2.2 置换密码（2）

- 单字母表置换
  - 让明文中的每一个字符都映射到其他一个字符上
  - 例如abcdefghijklmnopqrstuvwxyz
    QWERTYUIOPASDFGHJKLZXCVBNM
    则attack -> QZZQEA
  - 密钥共有26！≈ 4×10^26种可能，穷举攻击计算上不可行

8.2.2 置换密码（3）

- 利用自然语言的统计特性很容易破解密码。例如英文中e，t，o，a，n，i是最常出现的字母



8.2.3 转置密码（1）

- 转置密码（transposition cipher）
  - 重新对字母进行排序，但是不伪装明文
  - 下图给出了一种常见的转置密码：柱形转置
    - MEGABUCK是密钥，作用是对矩阵列进行编号，例中第一列为A代表的那一列，以此类推
    - 明文按水平方向的行依次填充矩阵
    - 密文按列读出，列次序如密钥所规定

M	E	G	A	B	U	C	K
7	4	5	1	2	8	3	6
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Plaintext

pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo

Ciphertext

AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

8.2.3 转置密码（2）

- 转置密码破译方法
  - 观察英文常用字母出现的频率是否与明文的常规频率吻合，是的话则可以确定是转置密码
  - 根据特定的环境信息猜测一个可能的单词或者短语，观察单词中两字母组合的规律来确定列数
  - 当列数较小的时候，共有K(K-1)种可能的列对，对每一个列对进行检查看它的两字母组合的频率是否与英文两字母组合频率相匹配，依此逐步确认列的顺序。

8.2.4 一次一密（1）

- 一次一密（one-time pad）
  - 选择一个随机的位串作为密钥，将明文转变成一个位串，然后逐位计算两个串的异或值作为密文
  - 信息论表明，在没有密钥的情况下密文无法破解
  - 缺点
    - 密钥量大，无法记忆，必须携带书面副本
    - 对于丢失字符和插入字符敏感，发送方和接收方必须保持同步

8.2.4 一次一密（2）

- 一次一密示例
  - 明文：I love you.
  - 密钥是Pad1，若选择Pad2用来解密，得到明文结果Elvis lives，似是而非

Message 1:	1001001	0100000	1101100	1101111	1110110	1100101	0100000	1111001	1101111	1110101	0101110
Pad 1:	1010010	1001011	1110010	1010101	1010010	1100011	0001011	0101010	1010111	1100110	0101011
Ciphertext:	0011011	1101011	0011110	0111010	0100100	0000110	0101011	1010011	0111000	0010011	0000101
Pad 2:	1011110	0000111	1101000	1010011	1010111	0100110	1000111	0111010	1001110	1110110	1110110
Plaintext 2:	1000101	1101100	1110110	1101001	1110011	0100000	1101100	1101001	1110110	1100101	1110011

8.2.4 一次一密（3）

- 量子密码学
  - BB84协议

8.2.5 两条密码学原则（1）

- 原则1：被加密的消息必须包含一定的冗余度
  - 解密一条消息之后，接收者必须能够通过简单的检查手段或者计算来判断消息是否有效
  - 冗余信息对理解消息是不必要的，但是能够阻止主动入侵者发送垃圾信息，并欺骗接收者解密垃圾信息并在所谓的“明文”上执行相应的处理
  - 缺憾是冗余信息使被动入侵者也更加容易破解系统

8.2.5 两条密码学原则（2）

- 原则2：需要采取某种方法来对抗重放攻击
  - 确保接收到的消息能被验证是新鲜的，是最近被发出来的，而不是重放的
  - 一种措施是在每条消息中包括一个时间戳，以便表明消息的有效时间段
  - 其他措施后面继续讨论

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.3 对称密码算法

8.3.1 概述

8.3.2 数据加密标准DES

8.3.3 高级加密标准AES

8.3.4 密码算法的使用模式

8.3.5 其他密码算法

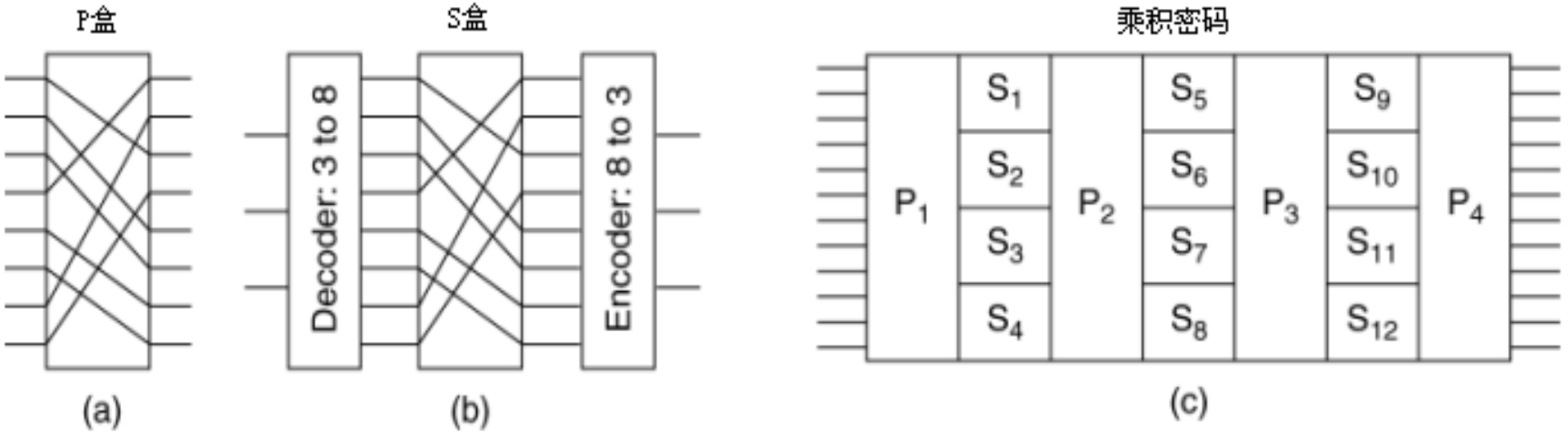
8.3.6 密码分析

8.3.1 概述（1）

- 现代密码学仍然使用置换和转置的思想，但是加密算法尽可能地错综复杂
- 对称密钥算法（symmetric-key algorism）
  - 使用同样的密钥完成加密和解密操作
- 块密码（block cipher）
  - 接受一个n位的明文块作为输入，利用密钥变换成n位的密文块

8.3.1 概述（2）

- 乘积密码（product cipher）
  - P盒完成转置，S盒完成置换
  - 一系列S盒和P盒叠加起来构成乘积密码，现代加密算法大多是乘积密码

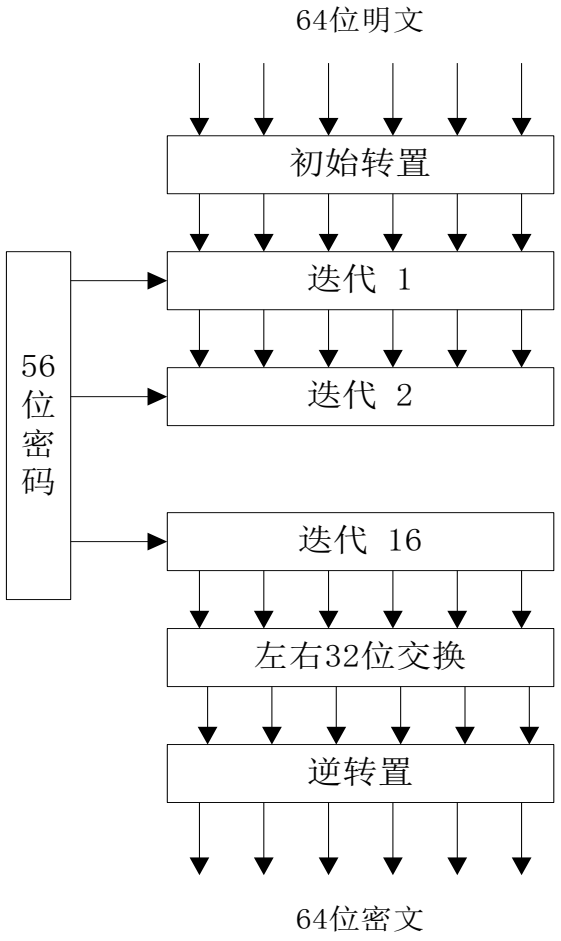


8.3.2 数据加密标准DES（1）

- 1977年，美国政府采纳了IBM开发的一个乘积密码作为无密级信息的官方加密标准，这个算法称为DES（Data Encryption Standard，数据加密标准）
- 关于DES的争议不断。美国国家安全局NSA使IBM把密钥从128位降低到56位，并且把DES的设计过程保密起来，人们怀疑DES算法隐藏了后门，NSA很容易破解。
- 最初的DES已经不适用，一个改进形式仍然非常有用

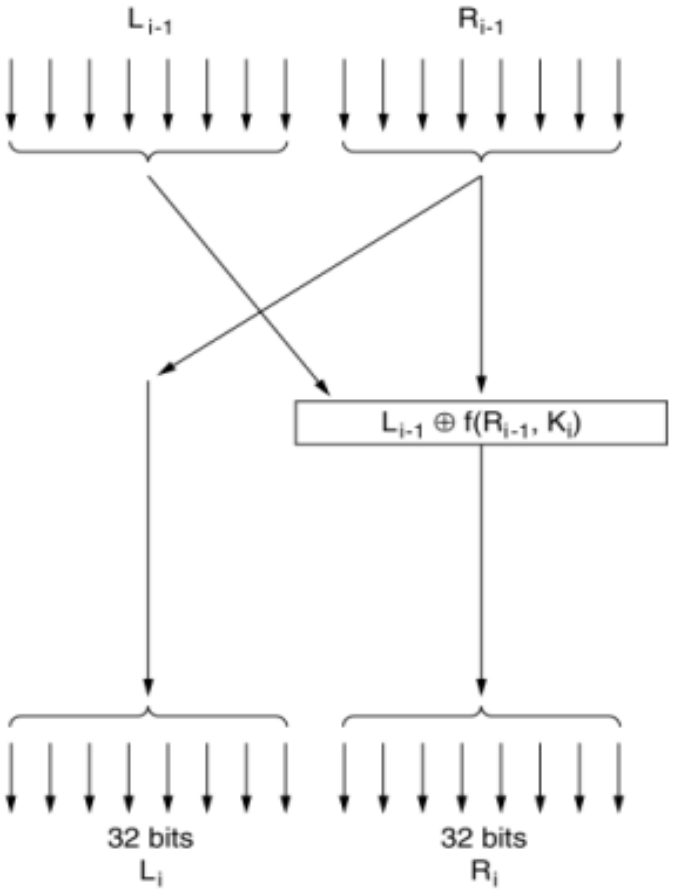
8.3.2 数据加密标准DES（2）

- DES工作流程
  - 明文按64位数据块的单元被加密，生成64位密文；密钥56位
  - 共19个步骤。第一步是一个与密钥无关的转置操作，直接作用在明文上；最后一步是这次转置的逆操作；倒数第二步是交换左32位和右32位；其他16步在功能上完全相同，但是使用原始密钥的不同函数作为参数
  - 用同样的密钥可以完成解密，但是解密步骤与加密步骤相反



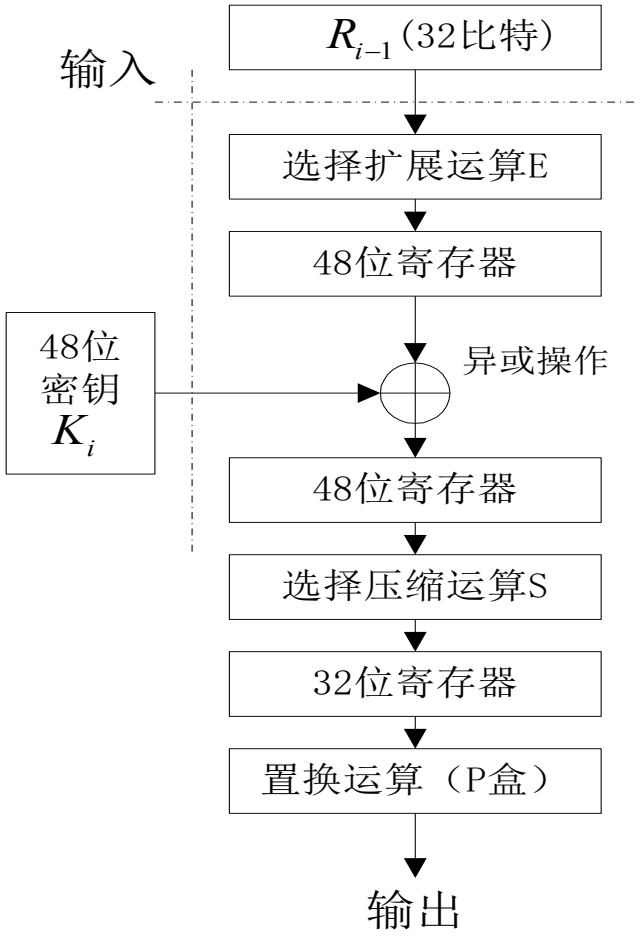
8.3.2 数据加密标准DES（3）

- 一次迭代过程（一轮）
  - 接受两个32位输入(L_{i-1}, R_{i-1})，产生两个32位输出(L_i, R_i)
  - 左边的输出L_i 等于右边的输入R_{i-1}
  - 右边的输出是左边输入L_{i-1}，与一个函数值$f(R_{i-1}, K_i)$逐位异或的结果，该函数的输入参数是右边的输入R_{i-1}和本轮的密钥K_i



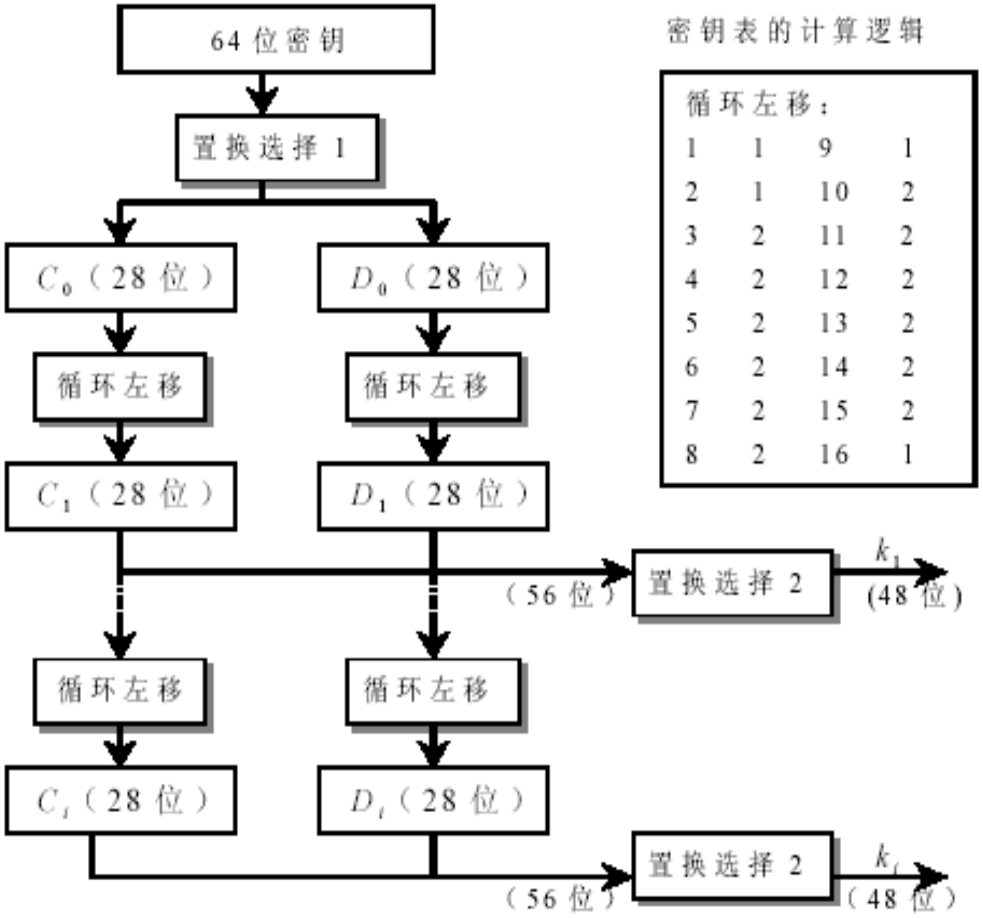
8.3.2 数据加密标准DES（4）

- F函数
  - 第一步：根据一个固定的转置和复制规则，将32位的R_{i-1}扩展成一个48位的数字E
  - 第二步：E和K_i被异或在一起
  - 第三步：异或结果被分成8个6位组，每个6位组被输入到不同的S盒中。每个S盒将6位输入映射到一个4位输出
  - 第四步：8个4位通过一个P盒，得到32位输出

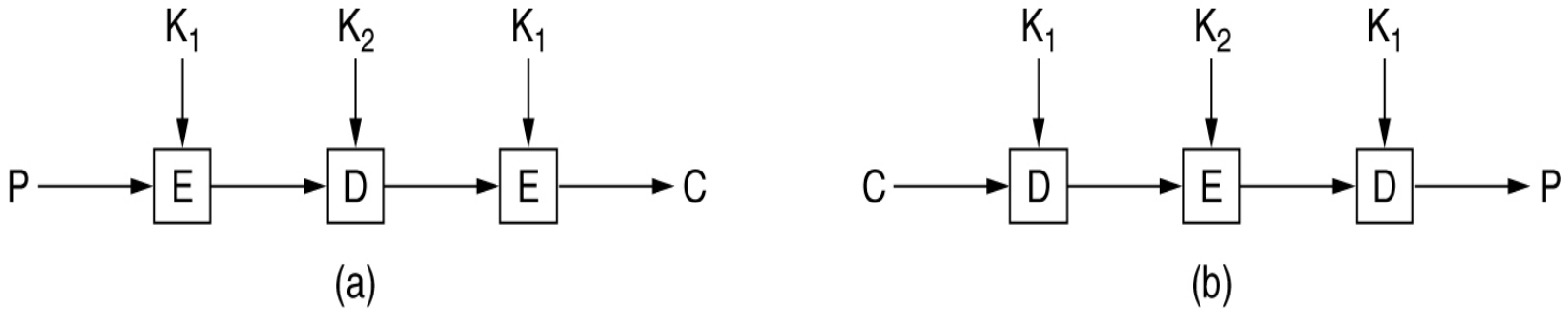


8.3.2 数据加密标准DES（5）

- 轮密钥的产生
  - 首先从64位密钥K中提取出56位，并执行转置操作
  - 每一次迭代之前，密钥被分成两个28位单元，每个单元循环左移位，左移位数取决于当前的迭代号
  - 移位之后再执行另外一个转置操作，即得到K_i



8.3.2 数据加密标准DES（6）



- 三重DES
  - 1979年初，IBM意识到DES的密钥长度太短，与是设计了三重DES来增加密钥长度
  - 共三步：第一步按照常规的方式用密钥K1执行DES加密；在第二步中，DES以解密方式运行，并使用K2作为密钥；第三步再次用K1执行DES加密
  - EDE（加密-解密-加密）模式是为了与现有的DES系统保持兼容性，并且112位密钥已经足够长了

8.3.3 高级加密标准AES（1）

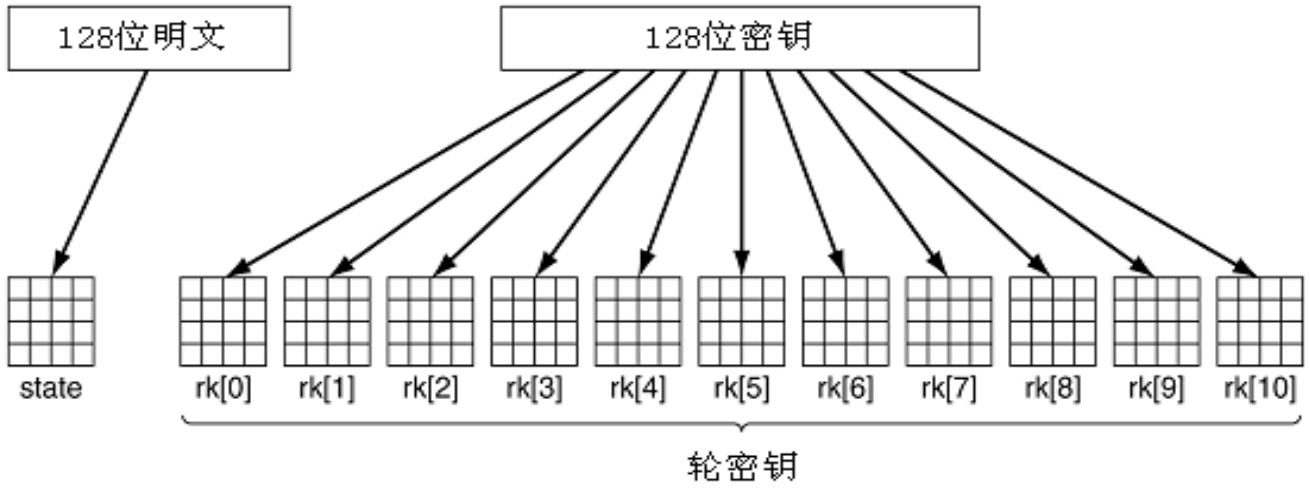
- 1997年4月15日，（美国）国家标准技术研究所（ NIST）发起征集高级加密标准（ Advanced Encryption Standard）AES的活动，活动目的是确定一个非保密的、可以公开技术细节的、全球免费使用的分组密码算法，作为新的数据加密标准。
- 对AES的基本要求是：比三重DES快、至少与三重DES一样安全、数据分组长度为128比特、密钥长度为128/192/256比特
- 1998年8月12日，在首届AES会议上指定了15个候选算法；2000年10月2日，NIST宣布了获胜者—Rijndael算法，2001年11月出版了最终标准FIPS PUB197

8.3.3 高级加密标准AES（2）

- **Rijndael算法**
  - 在数学上，Rijndael算法是以迦罗瓦域理论为基础的，具有一定的可以证明的安全性
  - 该算法采用的是代替/置换网络。每一轮由三层组成：第一层是线性混合层，确保多轮之上的高度扩散；第二层是非线性层，由16个S-盒并置而成，起到混淆的作用；第三层是密钥加层，子密钥简单的异或到中间状态上
  - 该算法明文和密文128bit,密钥可为128bit, 192bit, 256bit。迭代轮数也是可变的,分别为10轮,12轮,14轮
  - 所有的操作都以整个字节为单位，可以用软件和硬件高效率实现

8.3.3 高级加密标准AES（3）

- 算法结构



- State数组被初始化为明文数据，rk[0]-rk[10]是根据密钥经过复杂操作产生的11个与state同样长的数组，rk[1]-rk[10]分别是10轮的轮密钥

8.3.3 高级加密标准AES（4）

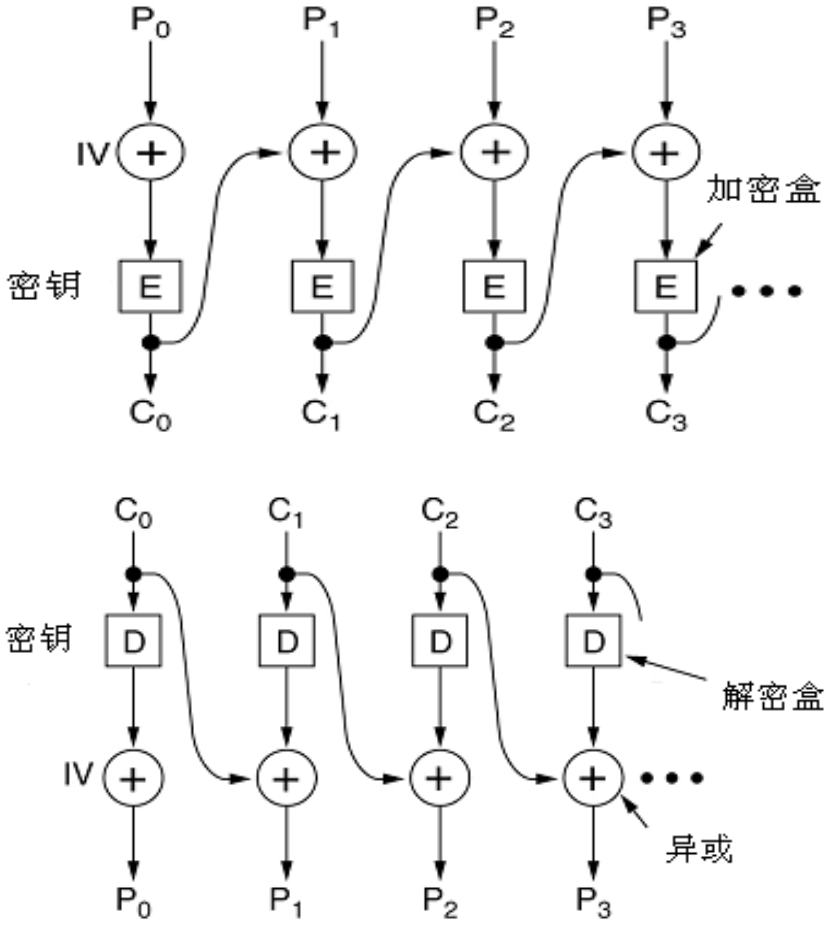
- 算法流程
  - 将明文复制到state数组中，前四个字节复制到第0列，接着四个字节复制到第1列，以此类推
  - state数组与rk[0]逐字节异或，结果保存在state中
  - 进入主循环，一共有10轮迭代，每次迭代如下：
    - 第1步：在state上执行逐字节替换，替换过程利用一个S盒进行，是直接的单字目表置换
    - 第2步：state数组每一行循环左移，第 j 行左移 j 个字节
    - 第3步：通过矩阵乘法混合state的每一列，新列是旧列与一个常量矩阵的乘积
    - 第4步：将本轮密钥rk[i]与state异或，结果保存在state中
  - 最后state数组即为密文

8.3.4 密码算法的使用模式（1）

- 电子代码簿模式（ECB）
  - 例如使用DES来加密一长段明文，将它分割成连续的8字节的数据块，然后用同样的密钥逐个加密这些数据块，这种加密方式称为ECB模式
  - 由于各个数据块是独立加密的，因此可以把两块密文互换位置，仍能够正确地解密出明文，只不过明文中相应两个数据块被颠倒了位置。如果这个数据块恰好是两个人不同的奖金数目，则事情就弄得很糟

8.3.4 密码算法的使用模式（2）

- 密码块链接模式（CBC）
  - 每个明文块在加密之前，先与上一个密文块执行异或操作
  - 第一个明文块与一个随机选择的IV（Initialization vector，初始向量）执行异或，该IV和密文一起传输
  - 解密同样使用异或操作将过程翻转
  - 特点是明文根据它出现的位置不同而生成不同的密文

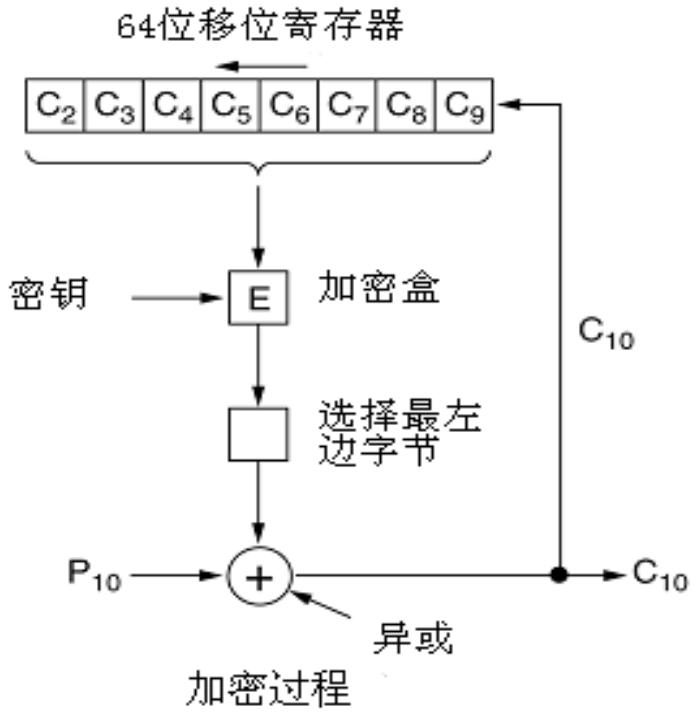


8.3.4 密码算法的使用模式（3）

- 密码反馈模式
  - CBC模式只能当整个64位数据块到来之后才能够加密，对于逐个字节加密的应用，可以采用密码反馈模式（cipher feedback mode）
  - 下面以DES算法为例说明该模式的使用

8.3.4 密码算法的使用模式（4）

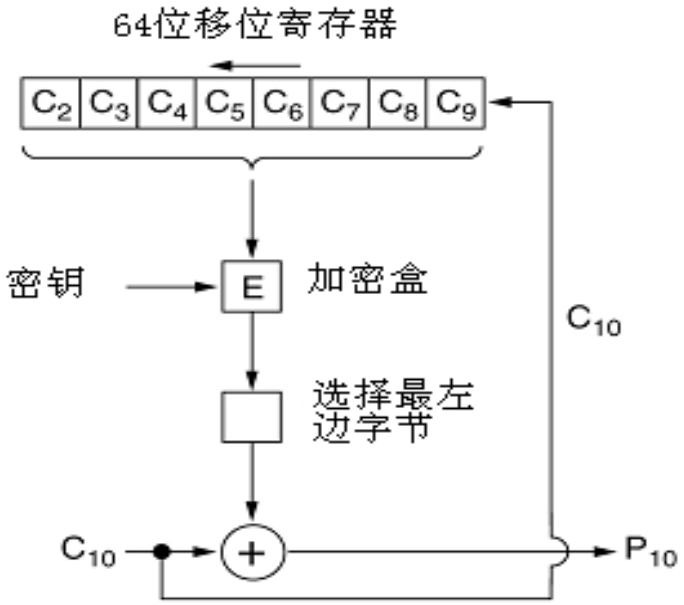
- 加密过程
  - 图中加密机处于“0-9字节已经被加密发送出去”之后的状态
  - DES算法作用在64位移位寄存器上并产生一个64位密文，密文最左边的字节被提取出来与明文字节10异或，异或值被传送到输出线路上
  - 异位寄存器左移8位，C2移出，C10移入最右边。本字节加密过程结束
  - 明文加密结果依赖于明文之前的历史；需要初始向量来启动加密过程



加密过程

8.3.4 密码算法的使用模式（5）

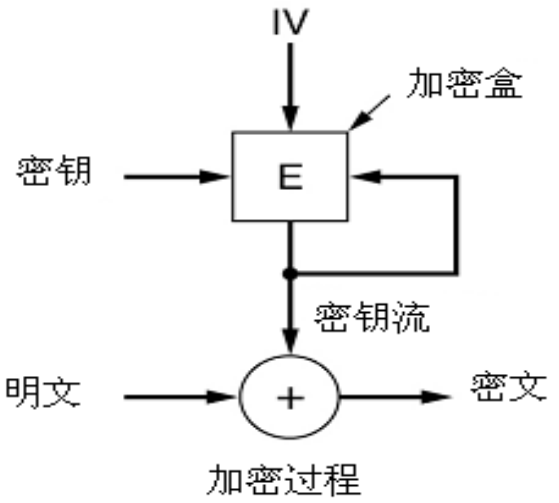
- 解密过程
  - 与加密过程类似，需要注意移位寄存器上仍然要使用DES加密！
  - 密码反馈模式的一个问题是，即使密文在传输过程中只有一位发生翻转，解密时该字节处于异位寄存器中时所波及的8个字节都被破坏。但是这种影响是局部的



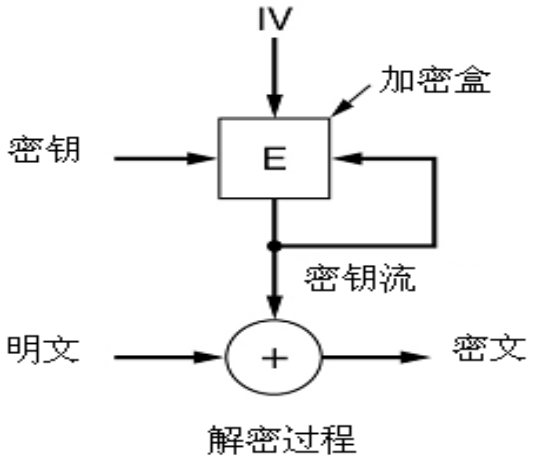
解密过程

8.3.4 密码算法的使用模式（6）

- 流密码模式
  - 用一个密钥加密一个初始向量得到一个输出块，用同一个密钥加密这个输出块得到第二个输出块，以此方式得到输出块的序列称为密钥流
  - 把密钥流和明文逐位异或得到密文输出
  - 初始向量和密文一起传输
  - 解密过程类似



加密过程



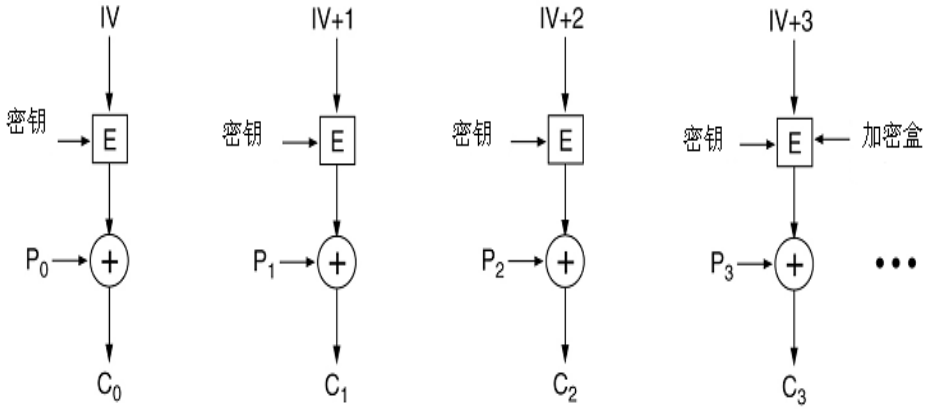
解密过程

8.3.4 密码算法的使用模式（7）

- 流密码模式的优缺点
  - 可以将密钥流提前计算出来以提高速度
  - 密文中的一位传输错误仅仅导致解密明文时一位发生错误，对传输错误不敏感
  - 不要两次使用同样的（密钥，IV）对，这样会得到同样的密钥流，使密文容易受到密钥流重用攻击，拿同一密钥流加密的两个密文异或即可消去密钥的影响

8.3.4 密码算法的使用模式（8）

- 计数器模式
  - 除了电子代码簿模式以外的其它模式都不能随机访问密文，必须解密前面的之后才能解密后面的内容，计数器模式能够解决这个问题
  - 初始向量再加上一个常数被加密，结果得到的密文再与明文异或输出
  - 同样存在密钥流重用攻击问题



8.3.5 其他密码算法

- 一些常见的对称密钥算法

密码算法	作者	密钥长度	说明
Blowfish	Bruce Schneier	1-448	又老又慢
DES	IBM	56	对于现在使用，太弱了
IDEA	Massey和Xuejia	128	好，但是属于专利算法
RC4	Ronald Rivest	1-2048	小心：有一些弱密钥
RC5	Ronald Rivest	128-256	好，但是属于专利算法
Rijndael	Daemen和Rijmen	128-256	最佳的选择
Serpent	Anderson，Biham，Knudsen	128-256	很强
三重DES	IBM	168	第二最佳选择
Twofish	Bruce Schneier	128-256	很强，被广泛使用

8.3.6 密码分析

- 差分密码分析
  - 原理：首先用一对仅在少量的位上有差异的明文块来加密，然后仔细观察在加密过程中每一次迭代时所发生的情况
- 线性密码分析
  - 将明文和密文中特定的位异或在一起，然后检查各种模式的结果。当重复这样的过程时，应该一半为0，一半为1。但是通常密码算法会在某一个方向上引入偏差，不管偏差多小，都可以用来降低工作量
- 电子功率分析
  - 计算机用3v表示“1”，0v表示“0”，因此处理“1”要比“0”耗费更多的电能。检测电能的变化可以破译逐位加密的算法
- 时间分析
  - 密码算法中充满对位测试的if、else语句，它们的执行时间不同。减慢时钟并观察每一步的时间，可以推测出轮密钥

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.4 公开密钥算法（1）

- 在对称密钥系统中，分发密钥往往是最薄弱的环节
- 1976年，斯坦福大学的Diffie和Hellman提出了一种全新的密钥系统
  - 加密密钥和解密密钥不同，从加密密钥不能轻易地推出解密密钥
  - 加密算法E和解密算法D必须满足三个要求：
    - D(E(P)) = P
    - 从E推出D极其困难
    - 用选择明文攻击不可能破解E
  - 具备了上述条件之后，加密密钥和加密算法都可以被公开
- 该系统命名为公开密钥密码系统（public-key cryptography）

8.4 公开密钥算法（2）

- 若干术语
  - 公开密钥密码系统要求每个用户拥有两个密钥，一个是公开密钥（公钥），另一个是私有密钥（私钥）
  - 其他人给某个用户发送消息的时候，用该用户的公钥加密
  - 用户解密收到的消息的时候，使用私钥解密

8.4 公开密钥算法（3）

- RSA
  - RSA算法由MIT的Rivest、Shamir、Adleman发现
  - 预计算
    - 选择两个大的素数p和q（通常1024位）
    - 计算n = p * q和 z = (p - 1)*(q - 1)
    - 选择一个与z互素的数，称为d
    - 找到e，使其满足 e * d = 1 mod z
  - 明文分块
    - 将明文化分成k位的块，k是满足2^k < n的最大整数

8.4 公开密钥算法（4）

- 加密
  - 加密消息P，只需要计算C = P^e (mod n)即可
- 解密
  - 解密密文C，只需要计算P = C^d (mod n)即可
- 公钥对是（e，n），私钥对是（d，n）

- RSA算法的安全性建立在大数分解的难度基础上，如果能够分解大数n，则算法自动破解。但是数学家探索了大数分解300年也进展不大
- 算法的缺点是速度太慢，一般用来分发密钥，而不直接用来加密数据

RSA例子

明文 (P)			密文 (C)		解密之后	
符号	数值	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	符号
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
Z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E

发送方的计算　　接收方的计算

- 取p = 3, q = 11
- 计算n = 33　z = 20
- 取d = 7,计算出e = 3
- C = P^3 (mod 33) ; P = C^7 (mod 33)

8.4 公开密钥算法（4）

- 其它公开密钥算法
  - 背包算法（Merkle and Hellman, 1978）
    - 算法思想是，某人拥有大量的物品，每个物品的重量各不相同。为了编码一个消息，这个人秘密选择一组物品并将物品放在一个背包中。背包中物品的总重量公开，所有可能的物品也被公开列出，但是背包中物品的细则是保密的。“根据给定的总重量找出可能的物品明细列表”被认为是一个计算上不可行的问题
  - El Gamal算法
    - 建立在计算离散对数的困难度基础之上
  - 椭圆曲线算法

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.5 数字签名

8.5.1 概述

8.5.2 对称密钥签名

8.5.3 公开密钥数字签名

8.5.4 消息摘要

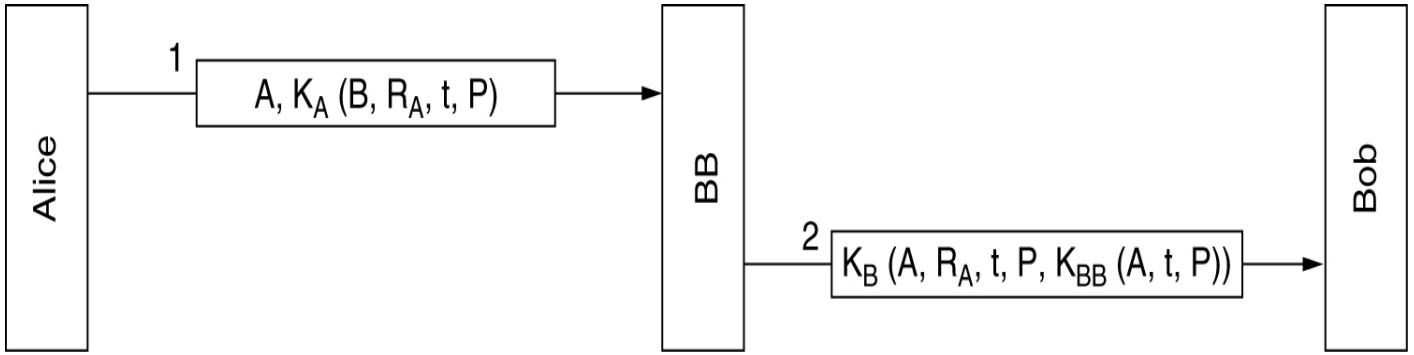
8.5.5 生日攻击

8.5.1 概述

- 现实生活中，法律的、金融的文档的真实性往往依据经授权的手写签名来决定，怎样在电子世界中实现签名？
- 数字签名消息的三个要求
  - 接收方可以验证发送方所宣称的身份
  - 发送方以后不能否认该消息的内容
  - 接收方不可能自己编造这样的消息

8.5.2 对称密钥签名（1）

- 设立一个人人都信任的中心机构，简称BB(Big Brother)。BB知道每个用户的秘密密钥K_A
- 当Alice要给Bob发送一条签名的明文消息P时，她生成$K_A(B, R_A, t, P)$，其中B是Bob的标识，R_A是Alice选择的一个随机数，t是一个时间戳(可用来证明该消息是最新的)，$K_A(B, R_A, t, P)$表示她用密钥加密之后的消息。然后按图中所示方式发送出去
- BB看到该消息来自Alice，于是用她的秘密密钥解密消息并按照图中所示给Bob发送一条消息。消息中包括了Alice的原始消息明文P和一条经过签名的消息$K_{BB}(A,t,P)$

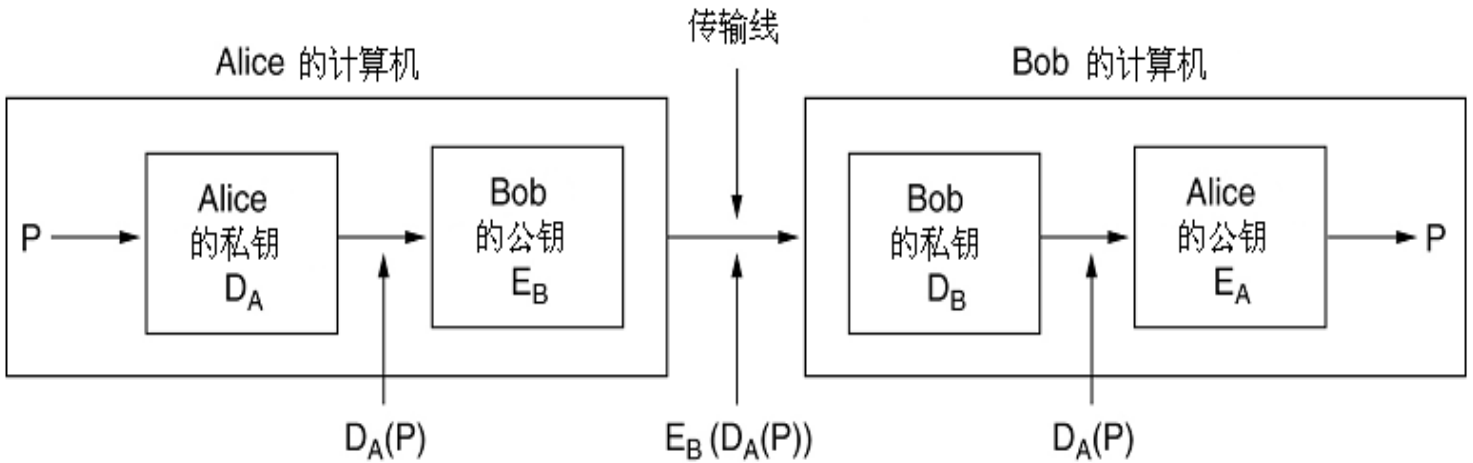


8.5.2 对称密钥签名（2）

- 如果Alice否认自己曾经发送过消息P怎么办
  - Bob首先指出，当BB看到来自Alice的消息时，除非消息是用K_A加密的，否则BB不会接受
  - Bob出示证据A:$K_{BB}(A,t,P)$，这是一条BB签名的消息
  - BB（大家都信任BB）解密证据A，表明Bob说的是真话，于是Alice否认失败
- 一个潜在的问题是Trudy可能重放其中一条消息，所以每一条消息中都包括时间戳，若新收到的消息时间戳与过去某一段时间内一条消息的时间戳相同，则丢弃新消息

8.5.3 公开密钥数字签名（1）

- 对称密钥签名系统存在的结构性问题是每个人都必须信任一个中心机构，但是现实中很难协调出一个各方都信任的权威机构
- 某些公开密钥算法具有D(E(P)) = P和E(D(P)) = P的双重属性，这为端到端的数字签名提供了可能



8.5.3 公开密钥数字签名（2）

- 如果Alice否认曾经发送过消息P，Bob只需要在法庭上出示P和$D_A(P)$，法官很容易验证P的确是Alice发出
- 如果$D_A(P)$被Alice很好地保存，则方案是可行的，如果Alice声称她的私钥丢失，则签名就失去了效力；另一个问题是Alice不能改变她的私钥，虽然她有权这么做
- 公开密钥数字签名算法的事实工业标准是RSA算法
- 1991年，NIST建议使用El Gamal算法的一个变种作为新的数字签名标准DSS（Digital Signature Standard），然而招致各种非议

8.5.4 消息摘要（1）

- 前面两种数字签名方案对整个消息加密，实际上将认证和保密两种不同的功能耦合在一起。但有些情况下，认证是必要的而保密不一定需要，采用消息摘要能够满足这一需求
- 消息摘要是一个单向散列函数，它接受一个任意长度的明文输入，计算出一个固定长度的位串
  - 给定P，很容易计算MD(P)
  - 给出MD(P)，要想找到P在实践中不可行
  - 在给定P的情况下，不能找到满足MD(P^) = MD(P)的P^
  - 在输入明文中只要有一位变化，也导致完全不同的输出
- 散列结果一般至少128位或者更长

8.5.4 消息摘要（2）

- 消息摘要既可以用在公钥密码系统中，又可以用在对称密钥密码系统中
- 下图是公钥系统应用的例子
  - Alice首先计算明文的摘要，然后对摘要进行公钥签名，将签名之后的摘要和明文一起发送给Bob

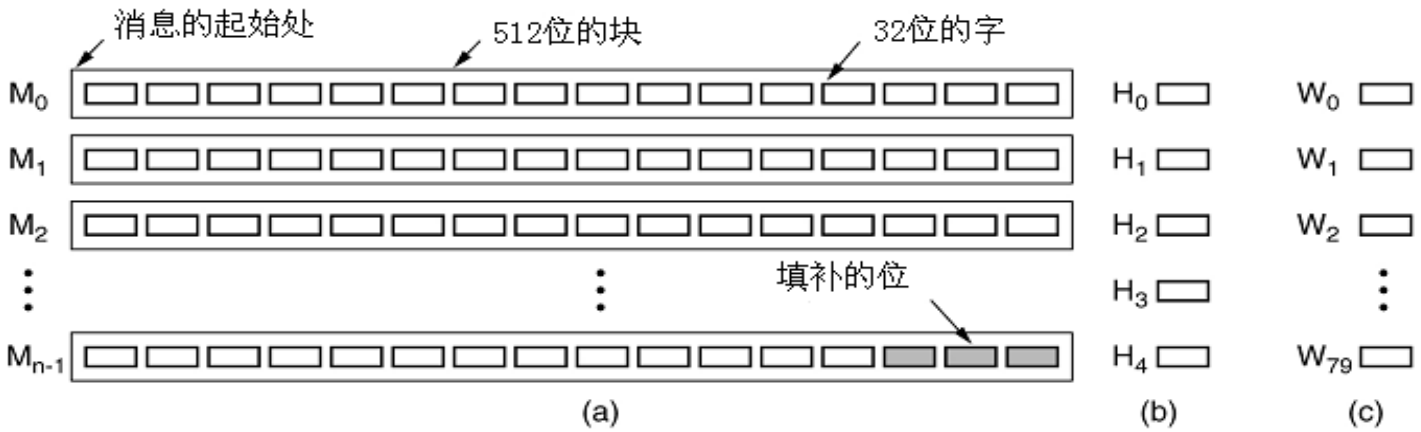
8.5.4 消息摘要（3）

- MD5
  - MD5算法是Ronald Rivest设计的一系列消息摘要算法中的第5个
  - 预处理过程
    - 将原始的明文消息填补到448位，然后追加一个64位整数来表示消息的原始长度，因而整个输入长度是512的倍数
    - 将一个128位的缓冲区初始化为一个固定的值
  - 计算过程
    - 每一轮取出一个512字节的输入块，将它与128位的缓冲区彻底地混合起来，这里使用了一个正弦函数表格
    - 对每一个输入块执行4轮，直至所有的输入块都执行完毕，最后，128位缓冲区中的内容即是消息摘要

8.5.4 消息摘要（4）

- SHA-1
  - 由NSA开发，并被标准化在FIPS180-1中
  - 输入块长度512位，消息摘要长度160位
  - 预计算
    - 在消息的末尾填补一个1位，然后跟上一定数量的0位，填补之后的长度保证是512的倍数
    - 构造一个表示填补之前消息长度的64位整数，将它与消息的低64位或操作

8.5.4 消息摘要（5）



(a)一个消息被填补至512的倍数；(b)输出的变量；(c)字数组

- 计算过程
  - 计算过程中维持了5个32位变量，从H_0到H_4，最后散列结果保存在这5个变量中
  - 按照顺序逐个处理数据块M_0至M_{n-1}。对当前的块，首先将16复制到一个80字的辅助数组的前面部分，然后利用下面公式填充其他64个字：

$$W_i = S^1(W_{i-3} XOR W_{i-8} XOR W_{i-14} XOR W_{i-16})$$

8.5.4 消息摘要（5）

$$W_i = S^1(W_{i-3} XOR W_{i-8} XOR W_{i-14} XOR W_{i-16}) \qquad (16 \le i \le 79)$$

- 利用H0-H4分别初始化5个临时变量A-E，

$$for(i = 0; i < 80; i++)\{$$

$$temp = S^5(A) + f_i(B,C,D) + E + W_i + K_i;$$

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

混合函数定义为：

$$f_i(B,C,D) = (B \,\&\, C) \,||\, (\overline{B} \,\&\, D) \qquad (0 \le i \le 19)$$

$$f_i(B,C,D) = (B \oplus C) \oplus D \qquad (20 \le i \le 39)$$

$$f_i(B,C,D) = (B \,\&\, C) \,||\, (B \,\&\, D) \,||\, (C \,\&\, D) \qquad (40 \le i \le 59)$$

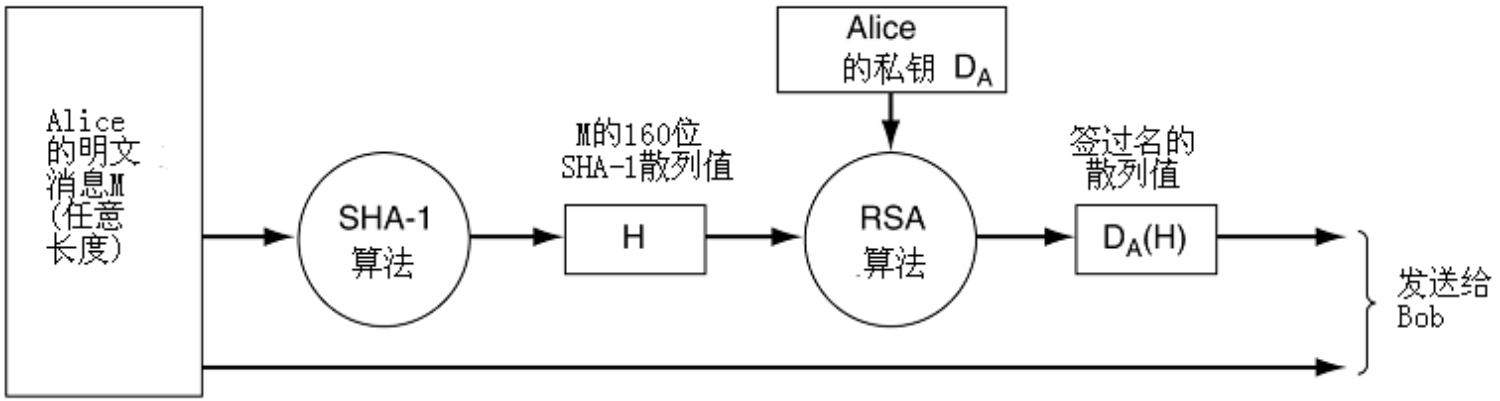
$$f_i(B,C,D) = (B \oplus C) \oplus D \qquad (60 \le i \le 79)$$

80次循环结束之后，A-E分别被加到H0-H4上

- 第一个512位块处理结束之后，下一块开始，W数组按照新的块初始化，H数组保持不变。所有块处理完毕之后，H数组的5个32位字级联起来就是160位散列值

RSA和SHA-1签名的例子

- Alice将明文输入SHA-1得到散列值，用RSA算法对散列值进行签名，最后把明文和签名的三列值发送出去
- Bob收到消息之后，自己计算明文的三列值，然后用Alice的公钥解密签名的三列值，得到原始的三列值。如果两者一致的话，则认为消息是有效的
- 保护完整性，而不需要保密。计算开销小



8.5.5 生日攻击

- Yuval（1979）在论文“How to Swindle Rabin”中发表了生日攻击方法，用2^(m/2)量级的操作次数就可以攻破一个m位的消息摘要，而不是表面看起来需要2^m量级的操作才能破解
- 一个班级中有多少个学生时，才使得有两个学生具有相同生日的概率超过1/2?
  - 大多数学生认为需要至少100，但是根据概率论只需要23人。23个学生的情况下，可以组成22*23/2 = 253对不同的组合，每一对组合有1/365的命中概率，所以总概率超过1/2！

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.6 公钥的管理

8.6.1 概述

8.6.2 证书

8.6.3 X.509

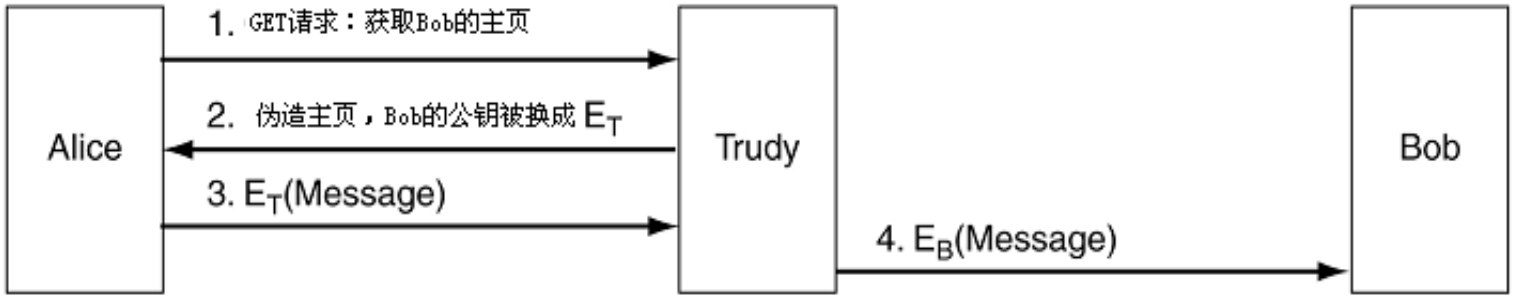
8.6.4 公开密钥基础设施

8.6.1 概述（1）

- 公开密钥密码系统使得人们有可能在不共享公共密钥的情况下安全地进行通信，但是存在一个问题：首先必须获得对方的公钥
- 一个很直观的方法是将公钥放在自己的Web站点上，是否可行？中间人攻击！

8.6.1 概述（2）

- Alice想要在Bob的主页上获得Bob的公钥
  - Alice在浏览器中输入URL ，她的浏览器通过DNS找到Bob的主页地址，并向该地址发送GET请求
  - 该请求被Trudy截获，回送Alice一个伪造的主页，将Bob的公钥替换成了Trudy的公钥E_T
  - 当Alice用E_T加密消息发送给Trudy时，Trudy就可以解密阅读消息内容



Trudy破坏公钥加密机制的一种方法

8.6.2 证书（1）

- 为了解决公钥分发的问题，产生了证书权威机构CA（Certification Authority），负责证明一个证书属于某个人、公司或组织
- 证书的基本任务是把一个公钥与安全个体的名字绑定在一起，证书包含CA对证书的消息摘要的签名

I hereby certify that the public key 19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A belongs to Robert John Smith 12345 University Avenue Berkeley, CA 94702 Birthday: July 4, 1958 Email: bob@superdupernet.com
SHA-1 hash of the above certificate signed with the CA's private key

一个可能的证书和签过名的散列值

8.6.2 证书（2）

- 引入证书之后的中间人攻击
  - Trudy截获Alice的请求之后，她把自己的证书放在伪造的主页上，Alice收到主页之后发现证书不是Bob的，攻击失败
  - Trudy临时修改Bob证书中的公钥，那么当Alice收到主页之后，她计算发现用CA的公钥解密之后的三列值与自己计算出来的Bob证书的三列值不同，Alice知道发生了攻击，攻击失败
- 只要Trudy拿不到CA的私钥，中间人攻击就无法实施

8.6.3 X.509

- X.509是ITU设计的专门针对证书格式的标准，其核心是一种描述证书的格式

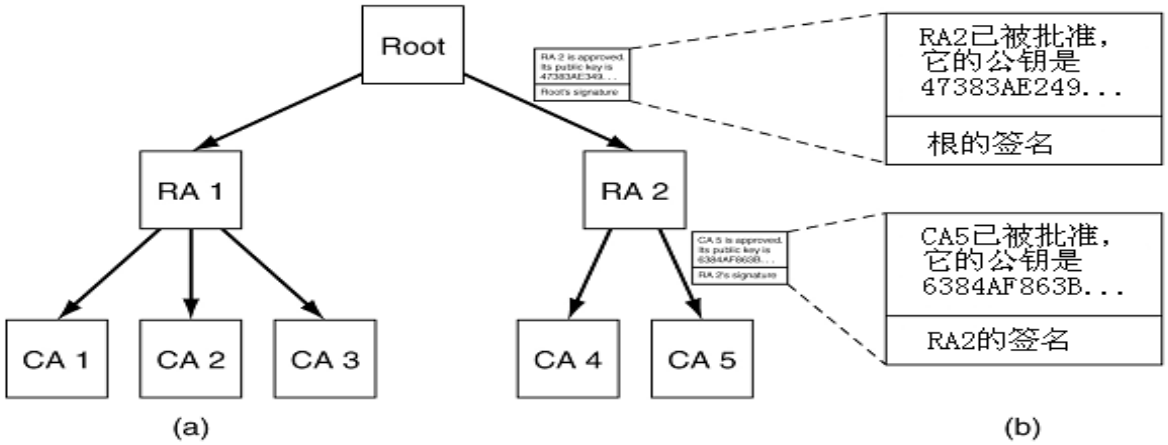
英文域名	中文	含义
Version	版本	X.509的哪个版本
Serial number	序列号	序列号加上CA的名字可以唯一标识当前证书
Signature algorism	签名算法	用于为证书签名的算法
Issuer	颁发者	CA的X.500名字
Validity period	有效期	有效期的起止时间
Subject name	主体名	该证书所证明的密钥持有者实体
Public key	公钥	主体的公钥，以及使用该公钥算法的ID
Issuer ID	颁发者标识符	一个可选的ID，唯一标识了证书的颁发者
Subject ID	主体标识符	一个可选的ID，唯一标识了证书的主体
Extensions	扩展域	目前已经定义了许多扩展域
Signature	签名	证书的签名（用CA的私钥做的签名）

8.6.4 公开密钥基础设施（1）

- 问题由来
  - 由一个CA来颁发全世界的证书是不切实际的，它将不堪重负而且成为单一失败点
  - 一个解决方案是采用多个由同一组织运行的CA，它们用同一个私钥来签名证书
    - 密钥泄漏问题！
    - 如何选择这样一个世界公认的组织？
  - 公开密钥基础设施PKI(Public Key Infrastructure)。PKI有多个部件，包括用户、CA、证书和目录

8.6.4 公开密钥基础设施（2）

- CA层次
  - 最顶级的CA，即层次的根，它的责任是证明第二级的CA（称为RA，区域权威机构）
  - RA的责任是证明下一级CA的真实性，这些下级CA真正为组织和个人颁发证书
  - 根CA授权一个新的RA时，它生成一个X.509证书，声明它批准了这个RA并在证书中包含该RA的公钥。根CA用私钥签名之后，将证书颁发给该RA



(a)一个层次状的PKI (b)证书链

8.6.4 公开密钥基础设施（3）

- PKI工作方式
  - 假设Alice需要与Bob通信，于是她搜寻到一个包含Bob的公钥的证书，该证书已经由CA5签过名
  - Alice不相信CA5，于是要求CA5证实自己的合法性，CA5用从RA2得到的证书回答Alice
  - Alice不相信RA2，于是要求RA2证实自己的合法性，RA2用从根CA得到的证书回答Alice
  - PKI假设每个人都知道根CA的公钥，Alice无需再怀疑，于是Bob的身份被证实

8.6.4 公开密钥基础设施（4）

- 一种节约时间的方法是Bob收集CA5和RA2的证书，然后将自己的证书和它们两个的证书一起交给Alice。Alice就可以根据自己知道的根CA的公钥，直接验证RA2和CA5的合法性。这种由底下回溯到树根的证书链有时称为信任链，或者证书路径
- 一般有多个根，每个根有自己的RA和CA。现代的浏览器在安装的时候预装了100多个根的公钥，这些根被称为信任锚（trust anchor）。用户可以检查信任锚并删除不信任的那些

8.6.4 公开密钥基础设施（5）

- 目录
  - 对于PKI，一个问题是在哪里存放证书以及证书链
    - 用户自己保存；安全但是不方便
    - 用DNS做成证书目录，返回IP地址的时候顺便返回证书链
    - 采用专门的目录服务器来管理X.509证书，允许用户利用X.500名字的属性提交查询任务。LDAP是一个候选目录标准

8.6.4 公开密钥基础设施（6）

- 证书的撤销
  - 某些情况下证书需要被撤销，比如用户私钥泄漏，甚至是CA私钥泄漏
  - 一种方法是让CA定期地发布一个CRL(证书撤销列表)，列出了所有已被撤消的证书的序列号。由于证书本身带着有效日期，所以CRL只需要包含需要撤销而且尚未过期的证书序列号
    - 用户在使用证书之前必须知道CA的CRL，以确定该证书是否被撤销。然而，即使证书不在CRL中，也有可能在CA发布CRL之后刚刚被撤销，因此真正的保证是每次使用一个证书都要查询CA看它是否已经被撤销，这带来很大的麻烦

8.6.4 公开密钥基础设施（7）

- 另外，一个撤消的证书也可能被恢复为有效的证书。由于必须处理证书的撤销和恢复，证书的最佳特性之一，即用户使用证书的时候不必与CA联系，就不复存在了
- CRL保存在哪里？一种方案是CA定期发布CRL，然后各个目录对CRL进行处理，它们只需要移除掉被撤消的证书即可
- 如果证书的有效期很长，那么CRL也会很长。一个办法是每过一个较长的时间才发布一个全列表，但是期间频繁地发行CRL更新消息，这样可以减少分发CRL需要的带宽

主要内容

8.1 网络安全概述
8.2 密码学
8.3 对称密钥算法
8.4 公开密钥算法
8.5 数字签名
8.6 公钥的管理
8.7 通信安全
8.8 认证协议
8.9 电子邮件安全
8.10 Web安全
8.11 社会问题

8.7 通信安全

- 8.7.1 IPSec
- 8.7.2 防火墙
- 8.7.3 虚拟私有网络
- 8.7.4 无线网络安全

8.7.1 IPSec（1）

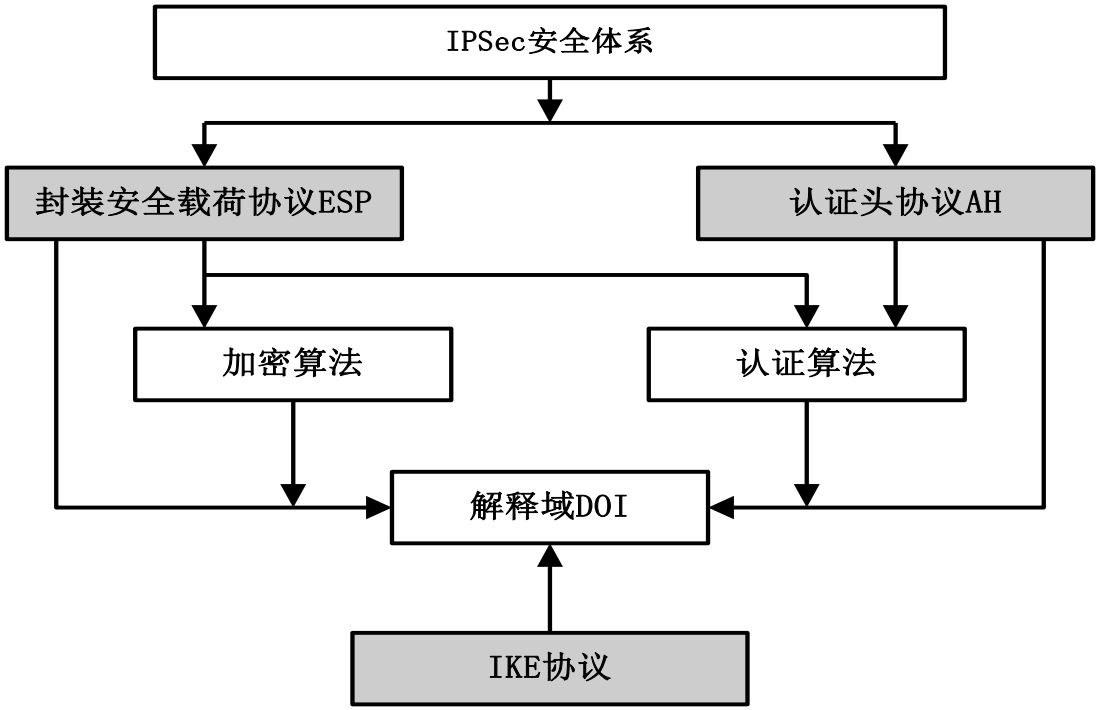
- 关于在Internet什么位置加入安全性爆发的争论
  - 一方认为，为了达到真正的安全性加密和完整性检查必须是端到端的，即放在应用层之上，这种做法的麻烦是需要改变所有的应用系统。从这个角度看，也可以放在传输层之上，既可以做到端到端加密又不需要改变应用系统
  - 另一方认为用户并不理解安全性，他们无法正确地使用安全性，而且没有人愿意修改已有的程序，所以应该把安全性加入到网络层
  - 争论结果是IETF制定了一个网络层的安全标准IPSec(IP security，IP安全)

8.7.1 IPSec（2）

- IPSec是一个多服务、多算法和多粒度的框架
  - 多服务：保密性、数据完整性、重放攻击保护
  - 多算法：与具体算法无关，支持多个算法
  - 多粒度：既能保护单个TCP连接，又能保护一对主机或者一对安全路由器之间的所有流量
- IPSec是面向连接的，一个连接称为一个SA(security association，安全关联)，是IPSec的基础和最重要的概念。SA是两个端点之间的单工连接，它定义了两端之间通信的安全措施，如是否采用加密和认证

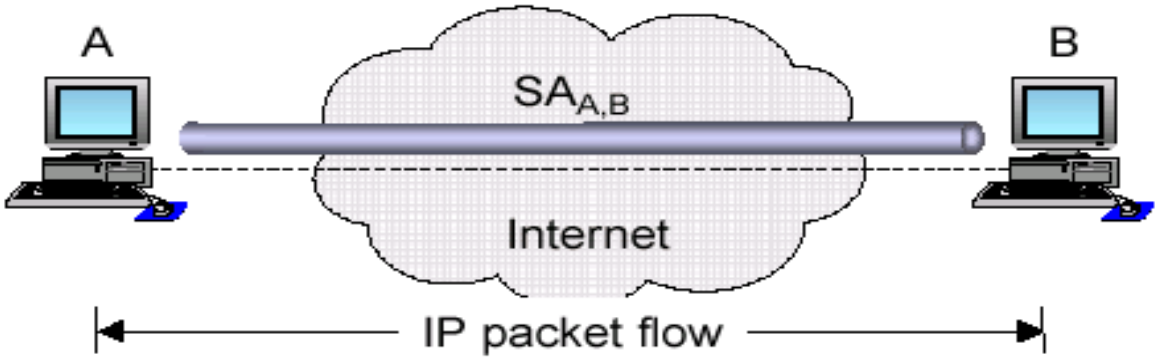
8.7.1 IPSec（3）

- IPSec框架
  - IPSec主要包括两部分，第一部分是两个新的头，AH(认证头)和ESP(安全封装净荷)；第二部分是ISAKMP(Internet安全关联和密钥管理协议)以及IKE(Internet密钥交换协议)，解决的是密钥管理和交换的问题



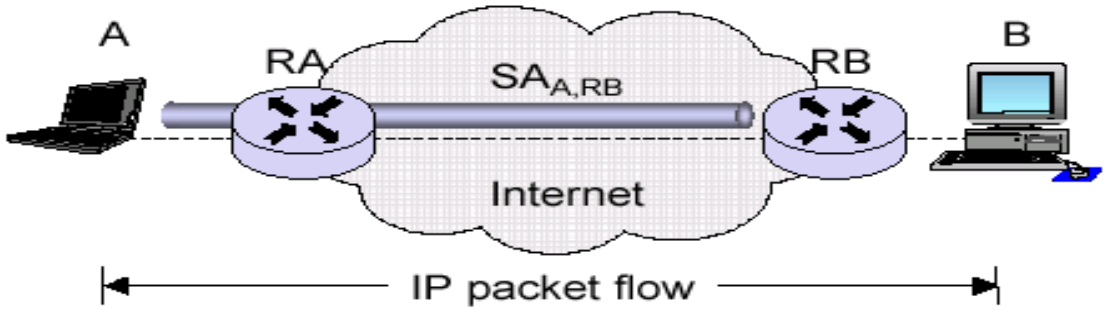
8.7.1 IPSec（4）

- IPSec传输模式
  - 用于两个主机之间的通信，两个通信端点同时都是加密端点
  - IPSec头插入到IP头和上层协议之间，只保护IP包中的上层协议部分



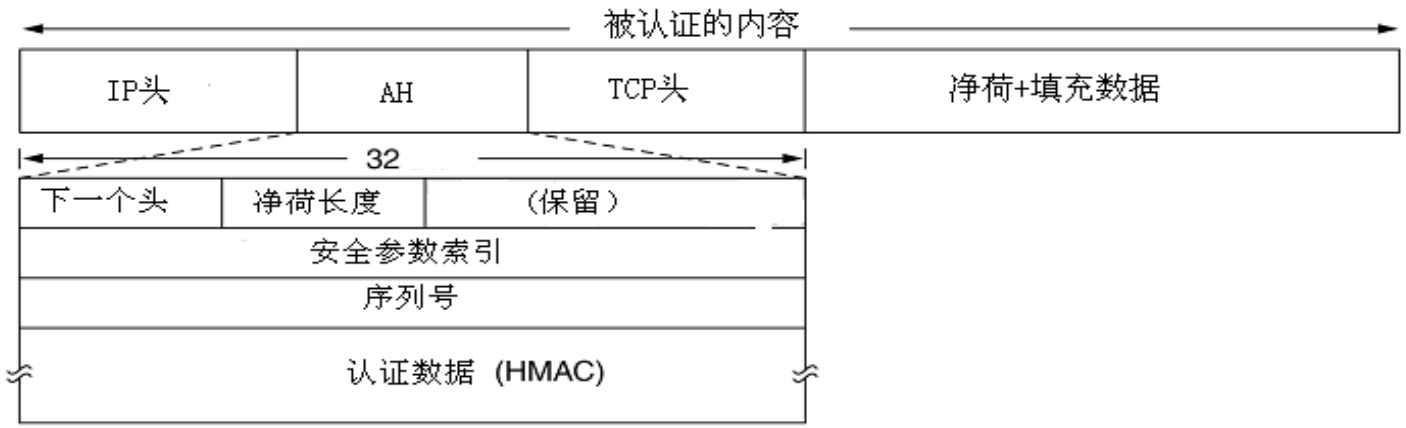
8.7.1 IPSec (5)

- IPSec隧道模式
  - 当一个加密端点是安全网关时，就必须使用隧道模式
  - 对整个IP包进行封装和保护。其中：外部IP头是IPSec处理以后添加上去的新IP头，其目的地址指向IPSec隧道终点的安全网关，源地址是本安全网关。在这种保护模式中，通信终点是由受保护的内部IP头指定的地址，而通信数据受保护的终点是由外部IP头指定的地址。



8.7.1 IPSec（6）

- 认证头AH
  - 提供完整性检查和防重放攻击
  - 在IPv4中，AH被放在IP头(包括所有可能选项)和TCP头之间；在IPv6中，AH只是一个扩展头



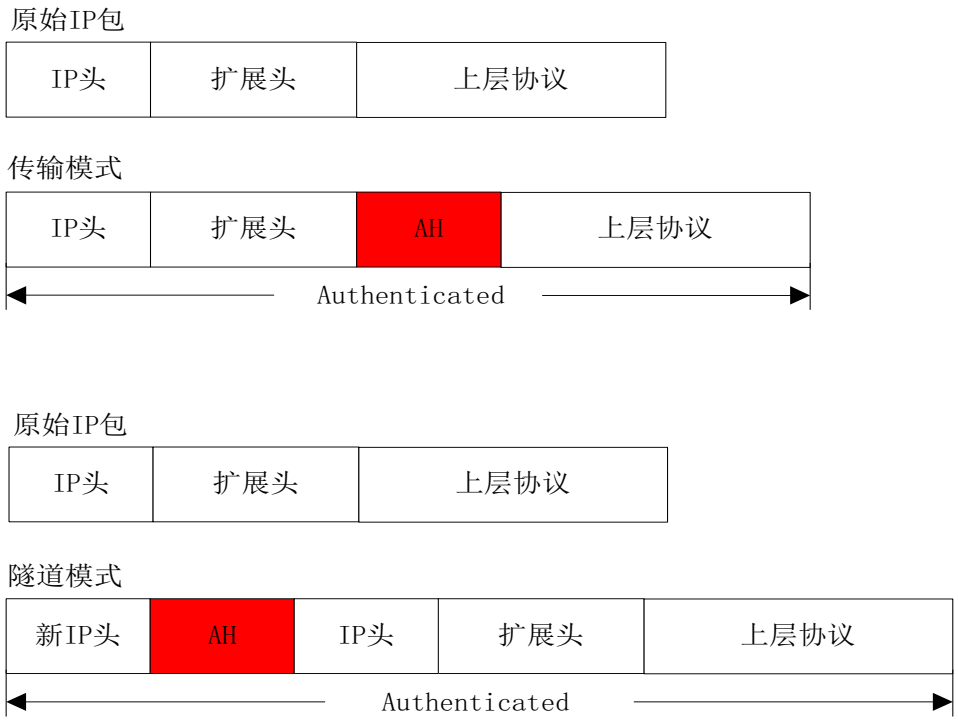
IPv4传输模式中的IPSec认证头

8.7.1 IPSec（7）

- AH各字段的解释
  - 下一个头：原始分组中的protocol域已经被替换成51代表后面紧跟一个AH头，AH中的下一个头域的值是传输层协议号，一般是6，代表TCP
  - 安全参数索引：是连接标识符，用以指定接收方数据库中一条特定的记录，该记录包括了这个连接上所使用的共享密钥以及加密算法等其他参数
  - 序列号：用来对一个SA上发送的所有分组进行编号，每个分组（包括重传分组）有唯一的序列号，检测重放攻击。若2^32个序列号全部用完，则SA必须重新协商，而不是回绕
  - 认证数据：可变长度，包含了净荷数据的数字签名。签名算法由SA建立时确定。一个简单方法是在分组和共享密钥串接起来的位串上计算散列值，称为HMAC(Hashed Message Authentication Code，散列消息验证码)

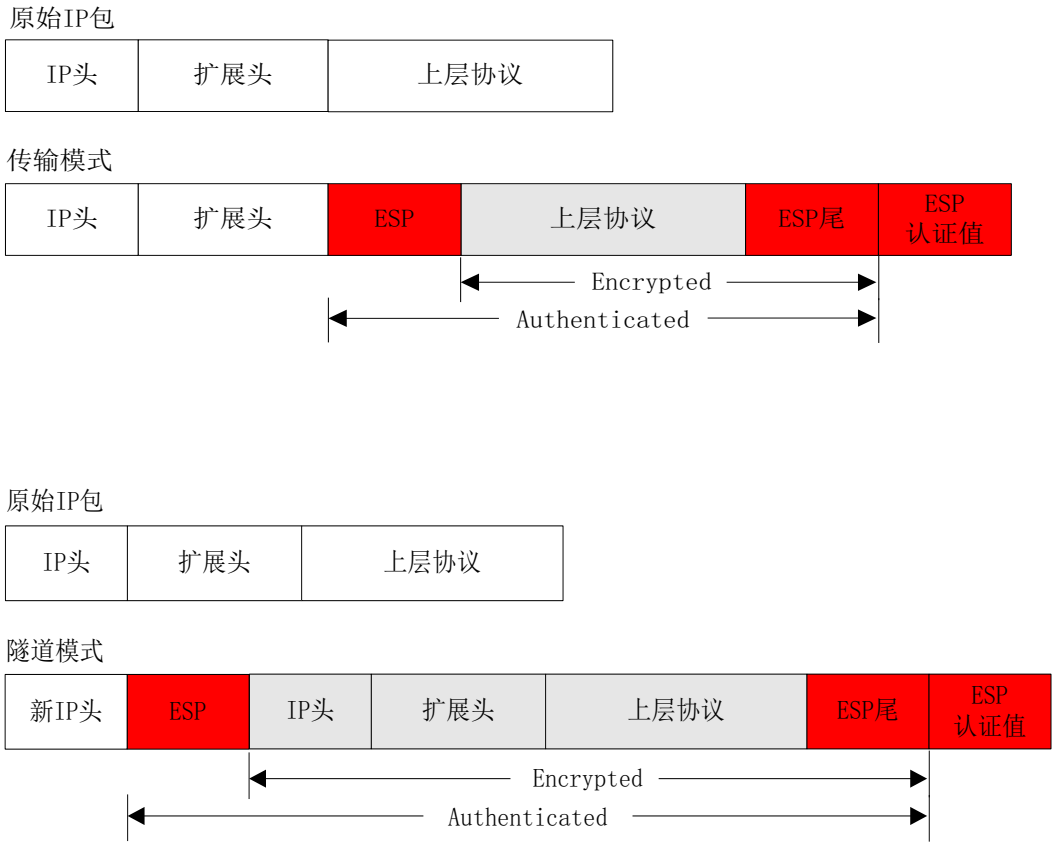
8.7.1 IPSec（8）

- AH的两种模式
- AH的完整性检查覆盖了IP头部的一些不随路由器变化的字段，可以对IP头部做完整性控制
- AH不支持数据加密，因此应用范围有限



8.7.1 IPSec（9）

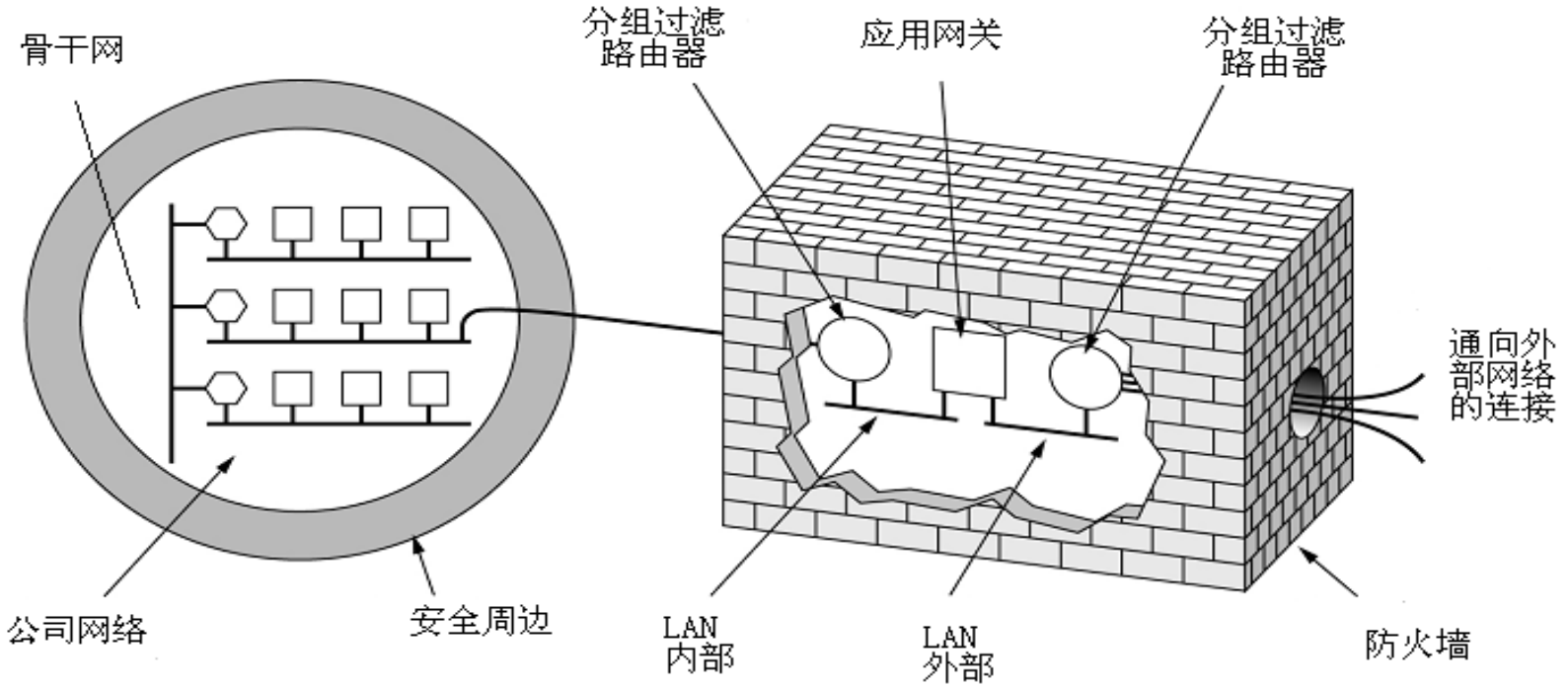
- 安全封装净荷
  - ESP头由安全参数索引和序列号组成
  - ESP支持对IP头部之后的部分进行完整性检查，HMAC放在消息尾部，以便于一边发送消息一边计算HMAC
  - ESP同时支持加密和完整性控制，很有可能完全取代AH



8.7.2 防火墙（1）

- 大多数公司有大量的保密信息放在LAN上，这些信息暴露给竞争对手将会带来可怕的后果
- 病毒、蠕虫和其他恶意代码泄漏进内部LAN之后也会对内部数据造成破坏
- 需要一些机制来管理和控制哪些信息能够出去，哪些信息能够进来——防火墙

8.7.2 防火墙（2）



- 典型的防火墙包括两个分组过滤路由器和一个应用网关。两个分组过滤路由器工作在网络层，分别负责监控流入流量和流出流量；应用层网关工作在应用层，负责监控具体的应用协议，比如电子邮件

8.7.2 防火墙（3）

- 分组过滤
  - 一般通过ACL（Access Control List，访问控制列表）进行
  - ACL的每一个表项称为一条过滤规则，一般包括（源IP地址，目的IP地址，源端口，目的端口，上层协议）五元组以及一个处理动作。
  - 对于到达的报文，首先检查该五元组，若有匹配表项，则按该表项的动作处理，若没有匹配项，则按照默认的处理规则处理
  - 过滤规则目前是由网络管理员手工添加的

8.7.2 防火墙（4）

- 应用网关
  - 是建立在应用层上的协议过滤机制，针对特别的网络应用服务协议的数据进行过滤
  - 针对每个不同的应用使用不同的处理方法
  - 优点是易于记录并控制所有的进出通信，并对Internet的访问做到内容级的过滤，控制灵活而全面，安全性高。具有日志、统计、审计和用户认证功能
  - 缺点是需要针对每一种应用编写不同的代码，维护比较困难，速度比较慢

8.7.2 防火墙（5）

- 防火墙是不是万能的？
  - 利用IP地址欺骗技术可以绕过防火墙
  - 防火墙允许穿过它建立连接，外部攻击者可以向内部服务器发送建立TCP连接请求，服务器以为是正常请求于是为该请求建立一个状态并分配一定的资源。如果攻击者同时发送大量的连接请求，就会耗尽服务器的资源而无法继续提供服务，致使合法用户也不能享用服务，这种攻击称为拒绝服务攻击（Denial of Service，DoS）。更糟糕的是攻击者可以利用系统漏洞控制成千上万台合法主机来进行DoS攻击，这种攻击危害力大而且难以预防，称为分布式拒绝服务攻击（DDoS）
- 防火墙不是万能的！

网络病毒与防治

- 导致计算机病毒产生的社会渊源
- 计算机病毒概述
- 计算机病毒的特征
- 计算机病毒传染方式
- 计算机病毒的分类
- 国内外计算机病毒的立法

导致计算机病毒产生的社会渊源

- 计算机病毒是计算机技术和以计算机为核心的社会信息化进程发展到一定阶段的必然产物，是计算机犯罪的一种新的衍化形式
- 产生计算机病毒的原因，大致可以分为以下几种：
  - 计算机爱好者出于好奇或兴趣，也有的是为了满足自己的表现欲或制作恶作剧
  - 产生于个别人的报复、破坏心理
  - 来源于游戏软件
  - 来源于软件加密
  - 用于研究或实验而设计的“有用”的程序
  - 出于政治、经济和军事等特殊目的，一些组织或个人也会编写一些程序用于进攻对方的计算机

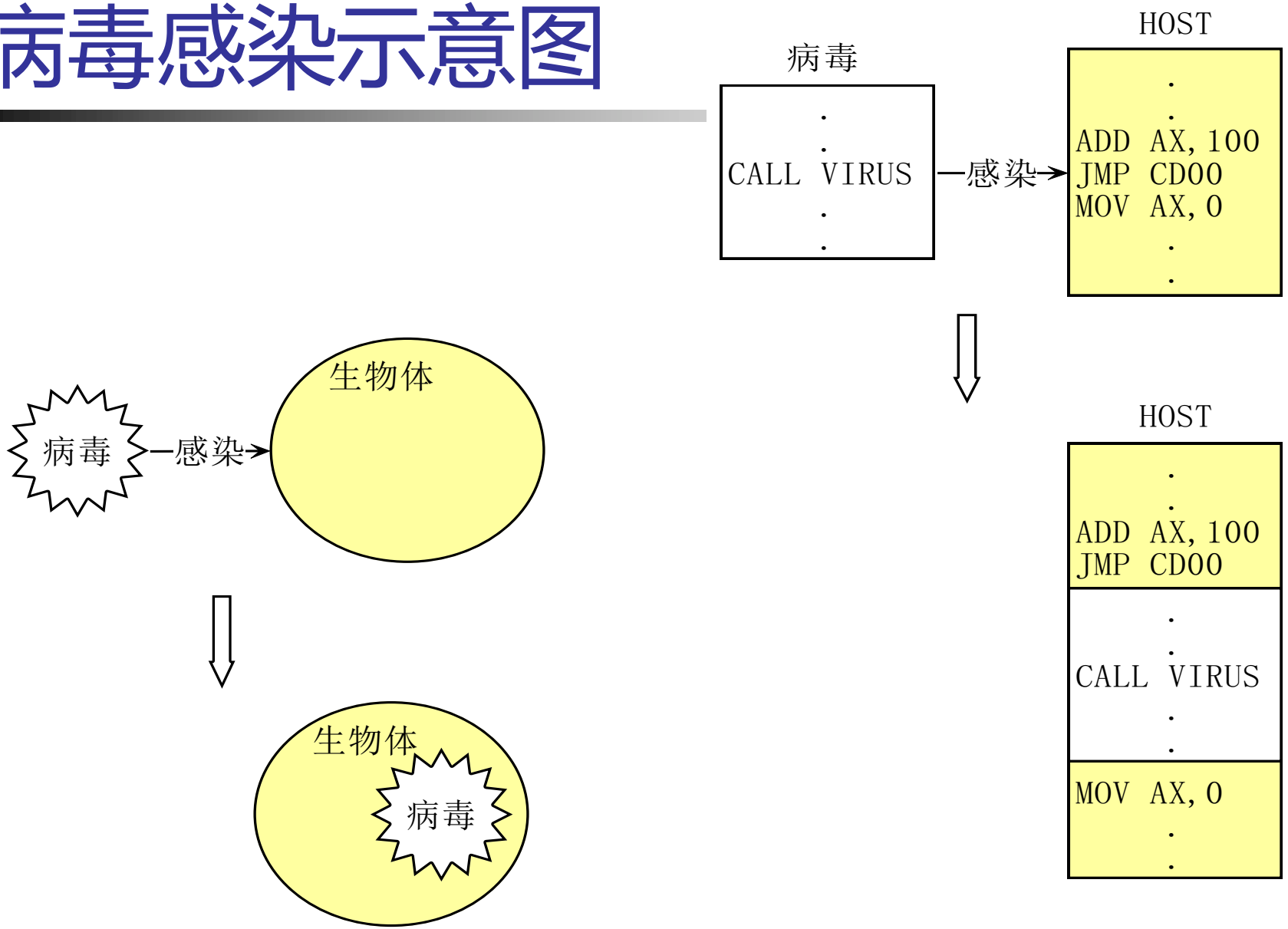
狭义定义

- 1994年2月18日，我国正式颁布实施了《中华人民共和国计算机信息系统安全保护条例》，在《条例》第二十八条中明确指出：“计算机病毒是指编制或者在计算机程序中插入的破坏计算机功能或者破坏数据，影响计算机使用并且能够自我复制的一组计算机指令或者程序代码”
  - 这一定义，具有一定的法律性和权威性

广义定义

- 计算机病毒(Computer Virus)，是一种具有自我复制能力的计算机程序，它不仅能破坏计算机系统，而且还能传播、感染到其他的系统，能影响计算机软件、硬件的正常运行，破坏数据的正确与完整。
  - 类似于生物病毒，它能把自身附着在各种类型的文件上或寄生在存储媒介中，能对计算机系统和网络进行各种破坏；
  - 有独特的复制能力和传染性，能够自我复制——主动传染，另一方面，当文件被复制或在网络中从一个用户传送到另一个用户时——被动传染，它们就随同文件一起蔓延开来

病毒感染示意图



生物病毒感染与寄生

计算机病毒感染与寄生

计算机病毒的特征

根据对计算机病毒的产生、传染和破坏行为的分析，病毒有以下几个主要特点。

1. 传染性(基本特征)
2. 夺取系统控制权
3. 隐蔽性
4. 破坏性
5. 潜伏性
6. 可触发性
7. 不可预见性

计算机病毒传染方式

- 计算机不运行时不存在对磁盘的读写或数据共享；没有磁盘的读写，病毒不能传入磁性存储介质，而只能驻留于内存之中或驻留于存储设备之中。计算机运行就含有很频繁的磁盘读写工作，就容易得到病毒传染的先决条件。
- 有了条件病毒是如何传染的呢？

（1）驻入内存：病毒要达到传染的目的必须驻留于内存之中，所以病毒传染的第一步是驻留内存；

（2）寻找传染机会：病毒驻入内存之后，首先寻找可进行攻击的对象，并判定这一对象是否可被传染（有些病毒的传染是无条件的）；

（3）进行传染：当病毒寻找到传染的对象并判定可进行传染之后，通过INT 13H这一磁盘中断服务程序达到传染磁盘的目的，并将其写入磁盘系统；

- 一般病毒将在内存中申请一空间，以便常驻内存或为常驻内存程序（TSR），病毒为使自己不被发现，此时一般不覆盖其它数据的，通常病毒程序写给于内存高端，其传染方式如上病毒存储方式。

计算机病毒的分类

各种不同种类的病毒有着各自不同的特征，它们有的以感染文件为主、有的以感染系统引导区为主、大多数病毒只是开个小小的玩笑、但少数病毒则危害极大（如臭名昭著CIH病毒），这就要求我们采用适当的方法对病毒进行分类，以进一步满足日常操作的需要。

分类（一）

- 按破坏性分类
  - 良性病毒：是指那些只是为了表现自己而并不破坏系统数据，只占用系统CPU资源或干扰系统工作的一类计算机病毒。
  - 恶性病毒：是指病毒制造者在主观上故意要对被感染的计算机实施破坏，这 类病毒一旦发作就破坏系统的数据、删除文件、加密磁盘或格式化操作系统盘，使系统处于瘫痪状态。
- 按传染方式分类
  - 系统引导型：感染引导区，系统引导时病毒装入内存，同时获得对系统的控制权，对外传播病毒，并且在一定条件下发作，实施破坏。
  - 文件型病毒：一般只传染磁盘上的可执行文件(COM，EXE)。将自身包围在系统可执行文件的周围、对原文件不作修改，运行可执行文件时，病毒程序首先被执行，进入到系统中获得对系统的控制权。
  - 混合型病毒：兼有以上两种病毒的特点，既染引导区又染文件，因此扩大了这种病毒的传染途径。

分类（二）

- 按连接方式分类
  - 源码型病毒：源码病毒较为少见，亦难以编写。因为它要攻击高级语言编写的源程序，在源程序编译之前插入其中，并随源程序一起编译、连接成可执行文件。此时刚刚生成的可执行文件便已经带毒了。在源被编译之前，插入到源程序中，经编译之后，成为合法程序的一部分。
  - 入侵型病毒：将自身入侵到现有程序之中，使其变成合法程序的一部分。
  - 操作系统病毒：可用其自身部分加入或替代操作系统的部分功能。因其直接感染操作系统，这类病毒的危害性也较大。
  - 外壳病毒：将自身附在正常程序的开头或结尾，相当于给正常程序加了个外壳。大部份的文件型病毒都属于这一类。

分类（三）

- 按广义病毒概念分类
  - 蠕虫(worm)：监测IP地址，网络传播
  - 逻辑炸弹(logic bomb)：条件触发，定时器
  - 特洛伊木马(Trojan Horse)： 隐含在合法程序上的一段非法程 序。
  - 陷门：在某个系统或者某个文件中设置机关，使得当提供特定的输入数据时，允许违反安全策略。

病毒、蠕虫和木马的区别

1．病毒的特点

计算机病毒是编写的一段程序，它可以在未经用户许可，甚至在用户不知情的情况下改变计算机的运行方式。病毒必须满足两个条件：

- 自行执行：它通常将自己的代码置于另一个程序的执行路径中。
- 自我复制：病毒代码的明确目的是自我复制。如：可以用受毒感染的文件副本替换其他可执行文件。

与蠕虫相比，病毒可破坏计算机硬件、软件和数据。

蠕虫的特点

蠕虫属于计算机病毒的子类，所以也称为“蠕虫病毒”。通常，蠕虫的传播无需人为干预，并可通过网络进行自我复制，在复制过程中可能有改动。与病毒相比，蠕虫可消耗内存或网络带宽，并导致计算机停止响应。

与病毒类似，蠕虫也在计算机与计算机之间自我复制，但蠕虫可自动完成复制过程，因为它接管了计算机中传输文件或信息的功能。

蠕虫与病毒之间的区别及联系

	病　毒	蠕　虫
存在形式	寄生	独立个体
复制机制	插入到宿主程序(文件)中	自身的拷贝
传染机制	宿主程序运行	系统存在漏洞(Vulnerability)
搜索机制(传染目标)	主要是针对本地文件	主要针对网络上的其它计算机
触发传染	计算机使用者	程序自身
影响重点	文件系统	网络性能、系统性能
计算机使用者角色	病毒传播中的关键环节	无关
防治措施	从宿主程序中摘除	为系统打补丁(Patch)

特洛伊木马

- 特洛伊木马(Trojan Horse)，简称木马，是一种恶意程序，是一种基于远程控制的黑客工具，一旦侵入用户的计算机，就悄悄地在宿主计算机上运行，在用户毫无察觉的情况下，让攻击者获得远程访问和控制系统的权限，进而在用户的计算机中修改文件、修改注册表、控制鼠标、监视/控制键盘，或窃取用户信息
- 古希腊特洛伊之战中利用木马攻陷特洛伊城；现代网络攻击者利用木马，采用伪装、欺骗(哄骗，Spoofing)等手段进入被攻击的计算机系统中，窃取信息，实施远程监控

木马的特点

木马(trojan)与病毒的重大区别是木马并不像病毒那样复制自身。

- 木马的特点：
  - 不感染其他的文件
  - 不破坏计算机系统
  - 不进行自我复制
- 木马的主要作用：作为远程控制、窃取密码的工具，它是一个具有内外连接功能的后门程序。
- 木马的两种主要传播途径：电子邮件和文件下载。

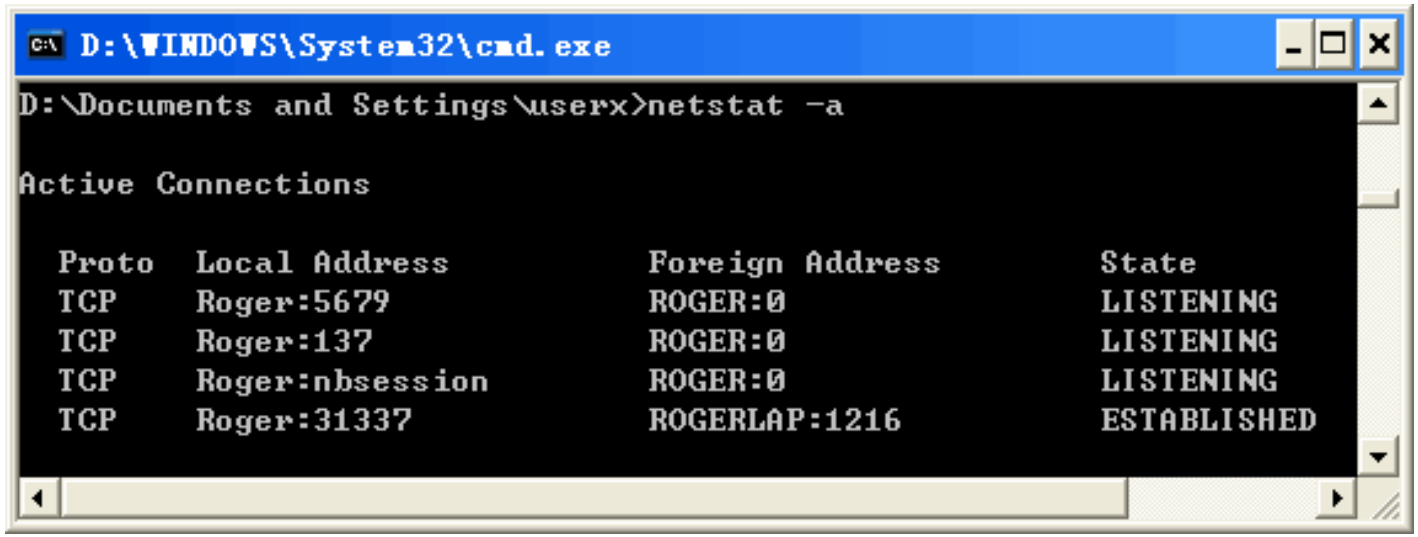
木马的组成

- 一般的木马程序包括客户端和服务器端两个程序，其中：
  - 客户端：用于攻击者远程控制植入木马的计算机（即服务器端）
  - 服务器端：是植入木马程序的远程计算机。
- 当木马程序或带有木马的其他程序执行后，木马首先会在系统中潜伏下来，并修改系统的配置参数，每次启动系统时都能够实现木马程序的自动加载。

检测和清除木马的方法

- 检测和清除木马的一般流程:

- 特洛伊木马入侵的一个明显证据是受害计算机上意外地打开了某个端口



用Netstat检测木马

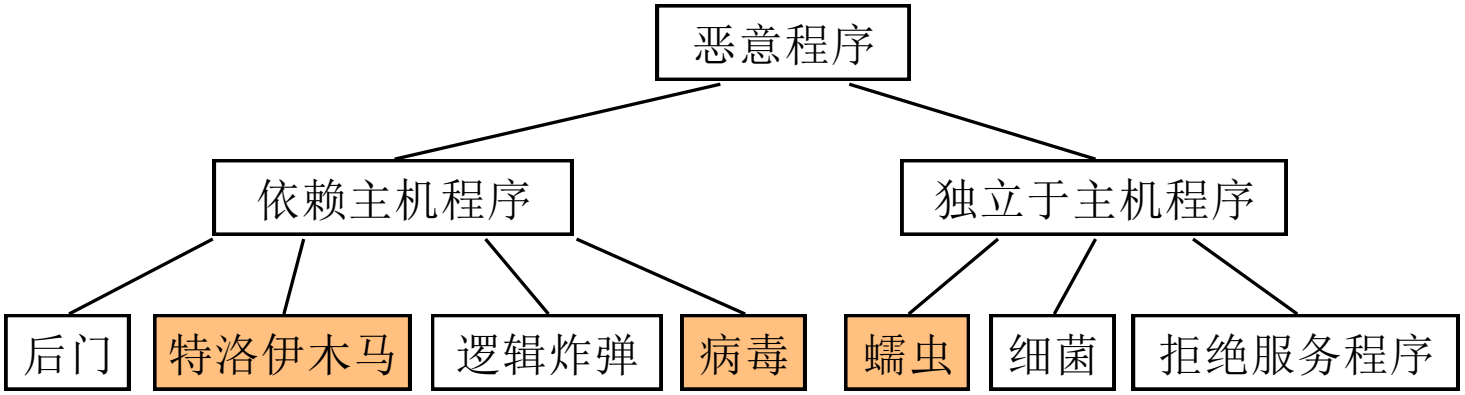
检测和清除木马的方法

进程标识符PID与映射名称

- Windows XP的Netstat工具提供了一个新的-o选项，能够显示出正在使用端口的程序或服务的进程标识符(PID)。有了PID，用任务管理器就可以方便地根据PID找到对应的程序，以便终止之

恶意程序

- 未经授权便干扰或破坏计算系统/网络的程序或代码称之为恶意程序/恶意代码
- 恶意程序大致可以分为两类：
  - 依赖于主机程序的恶意程序
    - 不能独立于应用程序或系统程序，即存在宿主
  - 独立于主机程序的恶意程序
    - 能在操作系统上运行的、独立的程序



本章小结

- 数据加密模型
- 对称密钥密码体制和公钥密码体制
- 鉴别与数字签名
- 通信安全机制
- 计算机网络病毒

Thanks !