,.

◎密码系统的安全性不在于算法的保密，而在于当对手获知了算法和密文后分析出密钥或明文的难度。

◎Vernam 密码的缺点（P45）

◎雪崩效应（P54）

在密码学中，雪崩效应是指加密算法的一个理想的属性，通常分组密码和加密散列函数。

雪崩效应是显而易见的，如果当一个输入稍微改变（例如，变换一个位），则输出显著变化（例如，一半的输出位变换）

在质量块密码的情况下，这样一个在关键或明文密文应该引起剧烈变化小的变化；

◎混淆：使得密钥和密文之间的关系尽可能的复杂;

扩散：明文统计中的冗余度在密文统计中消散;（P48）

◎分组密码与流密码的不同之处在于输出的每一位数字不是只与相应时刻输入的明文数字有关，而是与一组长为 m 的明文数字

在分组密码中，大小为 m（m> 1）的一组明文符号被一起加密，产生一组相同大小的密文。 即使密钥由多个值组成，单个密钥也用于加密整个块。

在流密码中，明文数字一次一个地被加密，并且连续数字的变换在加密期间变化。

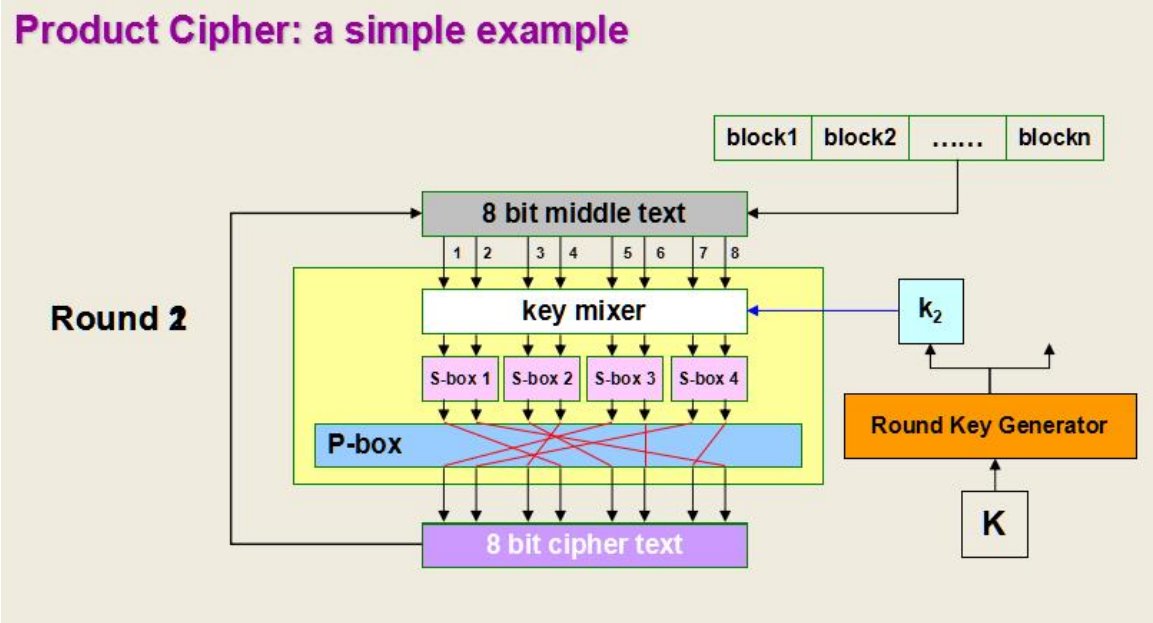
◎无条件安全、计算安全（P23~P24）

可证明安全：破译密码的难度与数学上某个困难问题的难度相同。

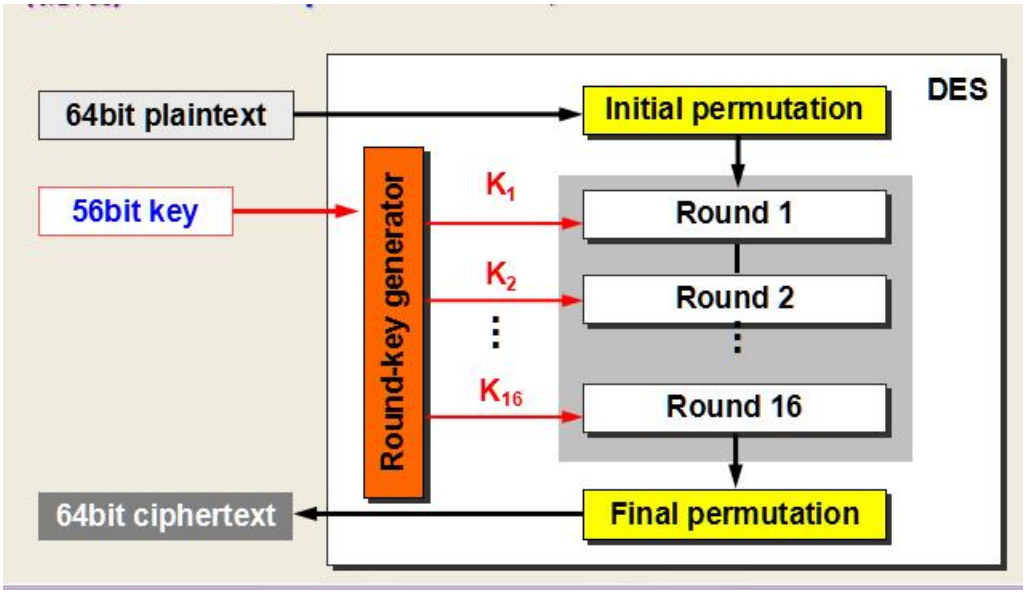
实际安全：包括可证明安全和计算安全

◎P-BOX（P46）

◎乘积密码：



◎DES 加密：Initial：初始置换



2.1 DES Encryption: Pseudocode (伪代码)

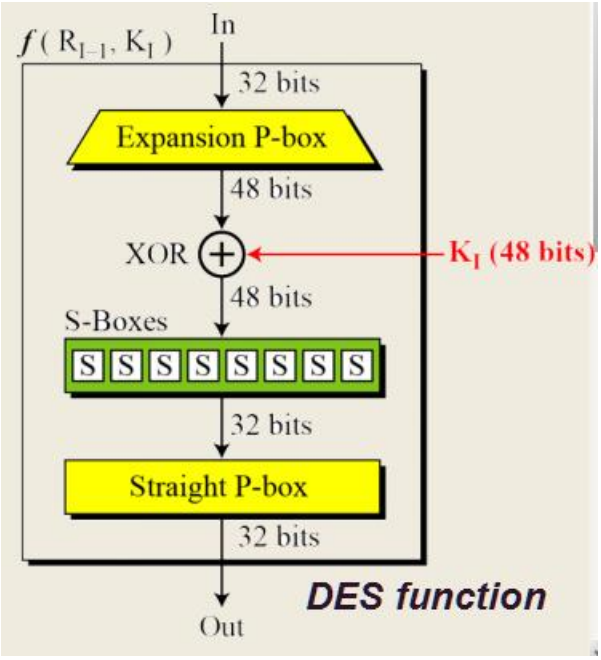
```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}
```

2.1 DES Encryption: Pseudocode for a Round

```
mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{
    copy (32, rightBlock, T1)
    function (T1, RoundKey, T2)
    exclusiveOr (32, leftBlock, T2, T3)
    copy (32, T3, rightBlock)
}
swapper (leftBlock[32], rigthBlock[32])
{
    copy (32, leftBlock, T)
    copy (32, rightBlock, leftBlock)
    copy (32, T, rightBlock)
}
```

2.1 DES Encryption: Pseudocode for DES Function

```
function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstituteTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}
```



在**扩展置换**之后，DES 对扩展的右部分和轮回密钥使用 XOR 运算。 注意，扩展的右部分和轮密钥都是 48 位长度。 还要注意，该轮密钥仅在该轮操作中使用。

S-box 做真正的混合 （混淆）。**DES 使用 8 个 S-boxes**，每个都有 6 位输入和 4 位输出。

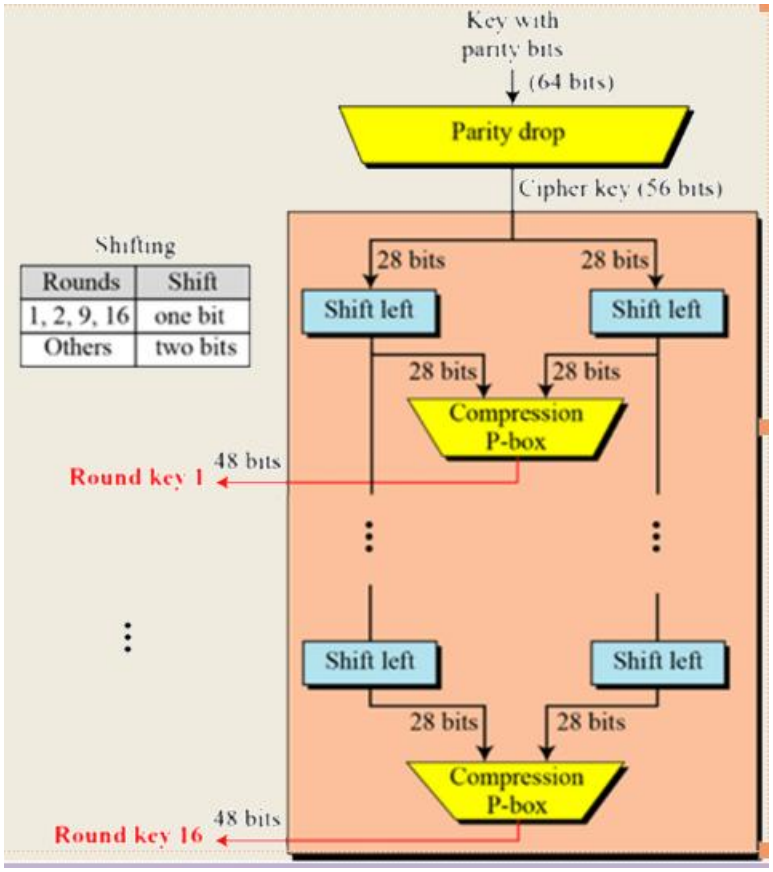
S-box

```
substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row ← 2 × inBlock[i × 6 + 1] + inBlock [i × 6 + 6]
        col ← 8 × inBlock[i × 6 + 2] + 4 × inBlock[i × 6 + 3] +
              2 × inBlock[i × 6 + 4] + inBlock[i × 6 + 5]

        value = SubstitutionTables [i][row][col]

        outBlock[[i × 4 + 1] ← value / 8;      value ← value mod 8
        outBlock[[i × 4 + 2] ← value / 4;      value ← value mod 4
        outBlock[[i × 4 + 3] ← value / 2;      value ← value mod 2
        outBlock[[i × 4 + 4] ← value
    }
}
```

◎64 位（包括 8 位奇偶校验位）DES 密钥->每轮轮密钥的转换过程：（64 位->56 位过程，是舍弃了 8 位奇偶校验位）



◎AES 的评价标准：

NIST 于 1997 年发出了 AES 的请求，NIST 定义的选择 AES 的标准分为三个方面：

安全、成本、实施。

1998 年 6 月接受了 15 名候选人

其中 5 个于 1999 年 8 月被列入名单：

1、MARS（IBM）:扩展 Feistel 密码，128 位分组，128-1248 位密钥。复杂、快速。安全系数高。

2、RC6（USA）：128 位分组，128-256 位密钥。非常简单，非常快。安全系数低。

3、Rijndael（Belgium）：128 位分组，128 - 256 位密钥。干净、快。安全系数良好。

,.

4、Serpent（Euro）：慢、干净。安全系数非常高。

5、Twofish（USA）：复杂、非常快。安全系数高

其中的一个（Rijndael）在 2000 年 10 月被选为 AES，2001 年 11 月作为 FIPS PUB 197 标准发布

◎AES（P99~P　）

基本描述：

分组大小：128 位

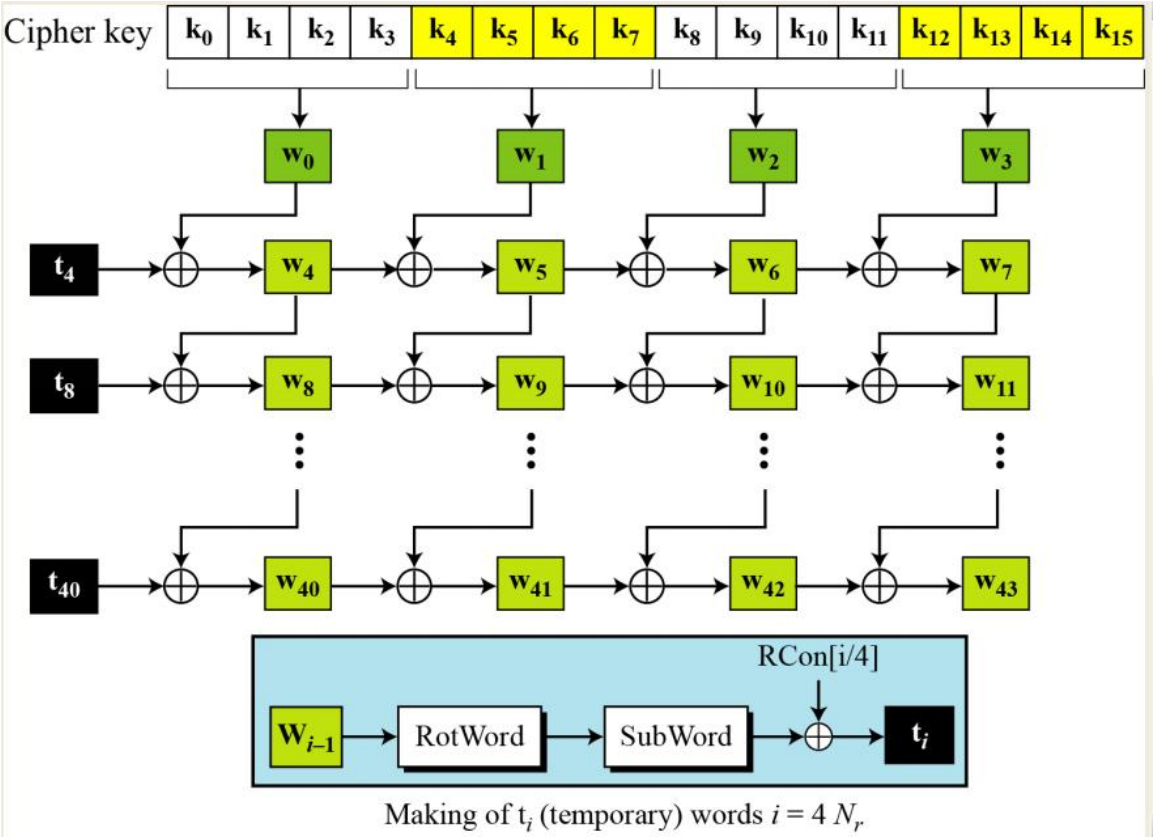
密钥大小：128 / 192 / 256 位（16 / 24 / 32 字节）

轮数：10 / 12 / 14

轮密钥大小：128 位

不是 Feistel 密码

◎AES 密钥拓展：



◎AES 可以在软件，硬件和固件中实现。实现可以使用表查找过程或使用一个定义良好的代数结构的例程。在 AES 中使用的算法很简单，可以使用便宜的处理器和最小量的存储器很轻松的实现。

◎AES 的安全系数：

AES 是在 DES 之后设计的。大多数对 DES 的已知攻击已经在 AES 上测试过。

暴力攻击：由于更大尺寸的密钥，AES 绝对比 DES 更安全。

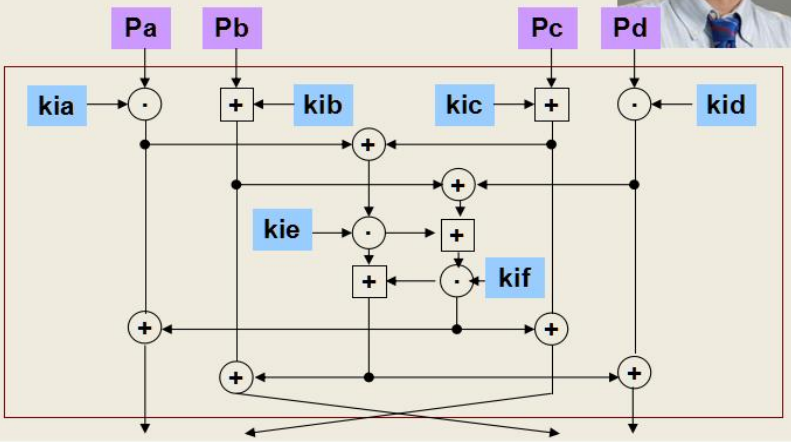
统计攻击：许多测试未能对密文进行统计分析。

差分和线性攻击：目前还没有对 AES 的差分和线性攻击。

◎更先进的对称加密算法

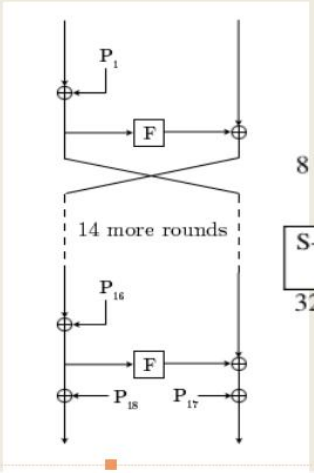
1、国际数据加密算法：

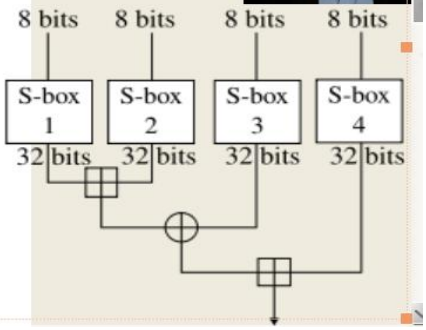
- 1.1 IDEA: International Data Encryption Algorithm,
  - 1992, Lai Xuejia (来学嘉)
  - 64位分组,
  - 128位密钥
  - 8 轮



2、

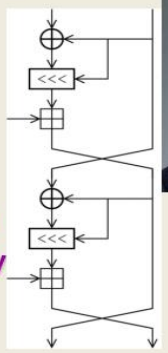
- 1.2 Blowfish
  - Bruce Schneier, 1993
  - Feistel 密码结构
  - 64位分组
  - 32–448 位密钥, 16 轮
  - Successor: Twofish （继承人）
- Key Generator （密钥生成器）
  - complex（复杂）





3、

- 1.3 RC5
  - 可变分组大小 (32, 64 or 128 bits),
  - 可变密钥大小 (0 to 2040 bits) and
  - 可变轮数 (0 to 255).
  - The original suggested choice of parameters (参数) were a block size of 64 bits, a 128-bit key and 12 rounds.
  - Successor: RC6



◎先进的分组密码的特点：

Variability（可变性）：密钥长度、分组大小、轮数、S 盒、轮函数

复杂的循环密钥生成

密钥有更多角色去执行：S 盒、循环移位

◎分组密码操作模式：

对称密钥加密可以使用现代分组密码来完成。

已经设计了操作模式以使用 DES 或 AES 对任何大小的文本进行加密。

在实质（实质上）中，操作模式是用于增强密码算法或使算法适应具体应用的算法的技术，诸如将分组密码应用于数据块组成的序列或者数据流。

1、电子本模式（ECB）：最简单的模式

每个明文块使用相同的密钥独立编码（独立地）

对 ECB 的总结：

相同的明文块，如果它在消息中出现多次，总是产生相同的密文。

无法阻止长消息的修改攻击

无错误传播：传输中的单个位错误可能在相应块中的多个位中产生错误。 但是，错误对其他块

没有任何影响。（为什么？因为 ECB 一次处理一个明文分块，每个明文块使用相同的密钥独立编码）

用法：ECB 模式是传输少量数据（例如加密/解密密钥或密码）的理想选择。

2、密文分组链接模式：

发送一个冗长的消息，我们会用所谓的密文分组链接模式（CBC）

CBC 解密：可以证明加密和解密是互逆的：

Pi = DK(Ci)⊕Ci-1 = DK(EK(Pi⊕Ci-1))⊕Ci-1 = Pi⊕Ci-1⊕Ci-1

IV：初始向量：为了产生第一块密文，初始化向量 IV 与第一个明文块进行异或。

IV 必须为发送者和接收者所共享。

典型应用：通用块定向传输、消息认证

3、密文反馈模式（CFB）

密码反馈模式：错误传播

消息认证

在可靠的通道

典型应用:通用面向流的传输

4、输出反馈模式（OFB）

无错误传播。

OFB 作为流密码，可以在不可靠信道中使用

5、计数器模式（CTR）

在计数器（CTR）模式下，没有反馈。 密钥流中的伪随机使用计数器来实现。

,.

◎**流密码（P162）**：

·概述

逐个处理消息（作为流），产生伪随机密钥流，与明文逐位（XOR）

流密钥的随机性完全破坏了消息中的统计的性质

$C_i = P_i \ \text{XOR} \ \text{StreamKey}i$

但不能重用流密钥，否则可以恢复明文

·属性

一些需要考虑的方面是：

长时间没有重复（加密序列周期长）

统计学上随机

取决于足够长的密钥

复杂性

正确设计，可以像具有相同大小密钥的块密码一样安全。但通常更简单，更快

◎**RC4**： 概述

RC4 是面向字节的流密码，其中明文的字节与密钥的字节一起 XOR 以产生密文的字节。

可变的密钥大小，面向字节操作的流密码

广泛应用 (web SSL/TLS，无线 WEP)

（关键的形式随机排列的所有 8 位值。使用 scramble(搅乱) 输入信息置换一次处理一个字节）

RC4 于 1987 年提出，和 DES 算法一样，是一种对称加密算法，也就是说使用的密钥为单钥（或称为私钥）。但不同于 DES 的是，RC4 不是对明文进行分组处理，而是字节流的方式依次加密明文中的每一个字节，解密的时候也是依次对密文中的每一个字节进行解密。

RC4 算法的特点是算法简单，运行速度快，而且**密钥长度是可变的**，可变范围为 1-256 字节(8-2048 比特)，在如今技术支持的前提下，当密钥长度为 128 比特时，用暴力法搜索密钥已经不太可行，所以可以预见 RC4 的密钥范围任然可以在今后相当长的时间里抵御暴力搜索密钥的攻击。实际上，如今也没有找到对于 128bit 密钥长度的 RC4 加密算法的有效攻击方法。

在介绍 RC4 算法原理之前，先看看算法中的几个关键变量：

1、密钥流：RC4 算法的关键是根据明文和密钥生成相应的密钥流，**密钥流的长度和明文的长度是对应的**，也就是说明文的长度是 500 字节，那么密钥流也是 500 字节。当然，加密生成的密文也是 500 字节，因为密文第 i 字节=明文第 i 字节⊕密钥流第 i 字节；

2、状态向量 **S**：**长度为 256**，S[0],S[1].....S[255]。每个单元都是一个字节（0~255），算法运行的任何时候，S 都包括 0-255 的 **8 比特数**的排列组合，只不过值的位置发生了变换；

3、临时向量 **T**：**长度也为 256**，每个单元也是一个字节。如果密钥的长度是 256 字节，就直接把密钥的值赋给 T，否则，轮转地将密钥的每个字节赋给 T；

4、密钥 **K**：**长度为 1-256 字节**，注意密钥的长度 keylen 与明文长度、密钥流的长度没有必然关系，通常密钥的长度趣味 16 字节（128 比特）。

一个例子：

- **RC4: simple case– RC4 3bits**
  - RC4 3-bit is a variation of the RC4 algorithm.
  - It has a variable length key from 3 bits to 24 bits
  - The 24 bit state vector S[ ] has elements S[0], S[1], … S[7]

- **eg:**
  - Plaintext: (7 2 5 1 5)oct {111 010 101 001 101}bin
  - Key: (7 4 3 2)oct {111 100 011 010} bin (keylen = 4*3bit=12)

- For initialization, use the following algorithm to initialize the state vector S and the vector T.

```
/* Initialization */
for i = 0 to 7 do
   S[i] = i;
   T[i] = K[ i mod keylen];
** NOTICE: this algorithm is the same as the RC4 except
the loop is from 0 to 7 but not 0 to 255)
```

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]
0	1	2	3	4	5	6	7
T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]
7	4	3	2	7	4	3	2

use the following algorithm to produce the initial permutation:

```
/*Initial Permutation of S */
j = 0;
for i = 0 to 7 do
 j = (j + S[i] + T[i]) mod 8;
 Swap(S[i], S[j]);
```

i	j	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]
		0	1	2	3	4	5	6	7
		T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]
		7	4	3	2	7	4	3	2
0	0+0+7	7	1	2	3	4	5	6	0
1	7+1+4	7	4	2	3	1	5	6	0
2	4+2+3	7	2	4	3	1	5	6	0
3	1+3+2	7	2	4	6	1	5	3	0
○○○									
7	3+5+2	7	2	5	1	3	0	6	4

产生一个 k，向量 S 交换一次，直到产生和明文一样多的 k：

- Next, since the plaintext contains 5 groups of 3 bits. 5 different values of k must be generated.
- /* Stream Generation */
- i, j = 0;
- while (true)
- i = (i + 1) mod 8;
- j = (j + S[i]) mod 8;
- Swap (S[i], S[j]);
- t = (S[i] + S[j]) mod 8;
- k = S[t];

i	j	S[0]	S[1]	S[2]	S[3]	S[4]	S[5]	S[6]	S[7]	t	k
		7	2	5	1	3	0	6	4	--	--
1	0+2	7	5	2	1	3	0	6	4	7	4
2	2+2	7	5	3	1	2	0	6	4	5	0
3	4+1	7	5	3	0	2	1	6	4	1	5
4	5+2	7	5	3	0	4	1	6	2	6	6
5	7+1	1	5	3	0	4	7	6	2	0	1

- **RC4 3bits: Encryption**
  - 产生的k的5个值是 (4 0 5 6 1). The RC4 3bit version generates random 3 bit patterns so these numbers represent 3 bit numbers. For example,
  - 0 = 000 2 = 010 4 = 100 6 = 110
  - 1 = 001 3 = 011 5 = 101 7 = 111
  - So $\{4\ 0\ 5\ 6\ 1\}_{oct} = \{100\ 000\ 101\ 110\ 001\}_{bin}$
- This can be XOR with the plaintext to produce the ciphertext.
  - Plaintext: (7 2 5 1 5) ={111 010 101 001 101}bin
  - ⊕ Key: (4 0 5 6 1) ={100 000 101 110 001}bin
  - =Cipher: (3 2 0 7 4) ={011 010 000 111 100}bin

◎密钥分发的历史：

在第二次世界大战期间，德国高级指挥官不得不向所有的谜运营商分发每月的日记账。

此外，U 型船，倾向于长时间离开基地，不得不以某种方式获得正常供应的钥匙。

美国政府密钥由 COMSEC 管理和分发。

,.

在 20 世纪 70 年代，COMSEC 每天负责运输吨钥匙。

当运载 COMSEC 材料的船舶进入船坞时，密码管理机构将在船上行进，收集堆叠的卡片，纸带，软盘，然后将其运送到预定的收件人。

◎

会话密钥使用由 KDC 和最终用户共享的主密钥以加密形式传输。

对于每个最终用户，存在与 KDC 共享的唯一主密钥，并且主密钥必须以某种方式递送。

◎Diffie-Hellman

Diffie 出生于 1944 年，像孩子一样被数学迷住，1965 年毕业于麻省理工学院，数学专业

1991 年之前，是北方电信网络安全系统研究经理

Diffie 对密钥分发问题特别感兴趣，到 20 世纪 70 年代初，他已经成熟为少数真正独立的安全专家之一，而不是受雇于政府或任何大公司。。

表演讲。他谈到了用于攻击关键分发问题的各种策略（策略），但他的观众没有积极的反应。

当完成他的谈话，迪菲被告知，有人最近访问实验室，并做了一个讲座，解决密钥分配的问题。

演讲者的名字是斯坦福大学教授马丁·赫尔曼（Martin Hellman）

Hellman 出生于 1945 年在犹太附近（犹太人）

他的同事批评他，他很疯狂地与美国国家安全局和他们的数十亿美元的预算竞争

当迪菲打电话给他时，赫尔曼从来没有听说过这个陌生人，但勉强地同意在下午晚些时候约会半小时。迪菲是在午夜左右离开赫尔曼的房子

Diffie 然后入学作为 Hellman 的研究生，因为 Hellman 没有大量的资金雇用他作为研究员。

◎加/解密不等于上/解锁，因为加/解密有顺序区分，必须是先进后出的。Fx(Fy(M)) ≠ Fy(Fx(M))

,.

◎Diffie-Hellman 密钥体制缺点：

1、比如 Alice 想要发送一封电子邮件给 Bob，但 Bob 可能睡着了。如果 Alice 想要加密她的消息，那么她需要与 Bob 在线同意密钥，或者当 Bob 唤醒时，她必须等待 12 个小时。

2、不能抵御中间人攻击（所以需要引入认证）

以上我们讨论的全是对称加密的密钥分配。

◎Diffie 已经提出了一种新型的密码，它包含了一个所谓的非对称密钥。加密和解密的密钥不同。

爱丽丝可以创建自己的一对密钥：加密密钥、解密密钥

Alice 保持解密密钥的秘密，所以它通常被称为 Alice 的私钥。她发布加密密钥，使每个人都可以访问它，这通常被称为爱丽丝的公钥。

这个系统的巨大优势是，没有浪费时间，鲍勃不得不等待从爱丽丝获得信息，然后才能加密和发送消息给她。

Alice 不必将公共加密密钥安全地传输到 Bob：完全相反，她现在可以尽可能广泛地公开她的公开（加密）密钥，以便整个世界可以使用它来发送她的加密消息。

同时，即使整个世界都知道爱丽丝的公共密钥，他们也不能使用爱丽丝的公共密钥解密消息。只有 Alice 可以使用她自己的私钥解密它们。

Diffie 实际上没有这个公钥/私钥方案的具体示例。

然而，他的想法是革命性的。

Hellman 和 Diffie 在 1975 年至 1976 年发表了他们的研究。他们（和许多其他密码学家）继续他们的研究，试图找到一个特殊的单向函数，使非对称密码成为现实。然而，他们没有发现。

A Revolutionary Breakthrough(突破性进展)

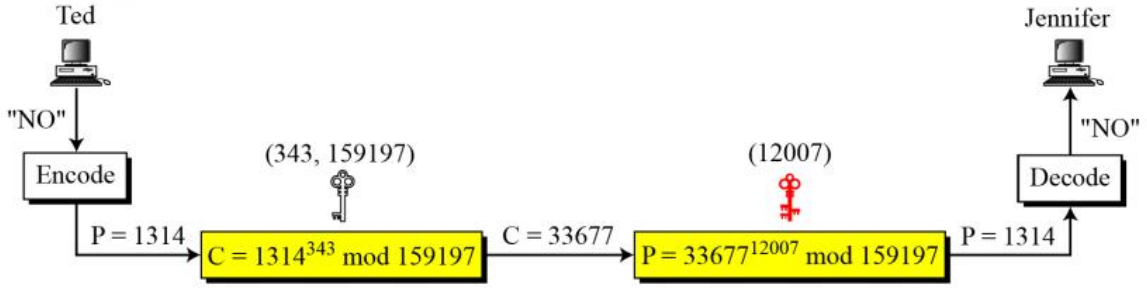
- 源于Diffie和Hellman的思想，先要找到一个符合这种性质的单向函数：$f_x f_y(M) = f_y f_x(M)$
- 可利用下面的单向函数：
  - $f = M^x \bmod n$;
- 设加密密钥为e，解密密钥为d （e≠d)
  - 若：$C = M^e \bmod n$ (加密)
  - 则：$M = C^d \bmod n$ (解密)
    $= (M^e \bmod n)^d \bmod n$
    $= M^{ed} \bmod n$
  - i.e., e和d如何能满足$M = M^{ed} \bmod n$?
  - a.k.a，如何找出符合上述条件的e, d和n ?

Principles of RSA

对于明文**M**和密文**C**，假设我们可以找到整数**e**，**d**和**n**，使得：

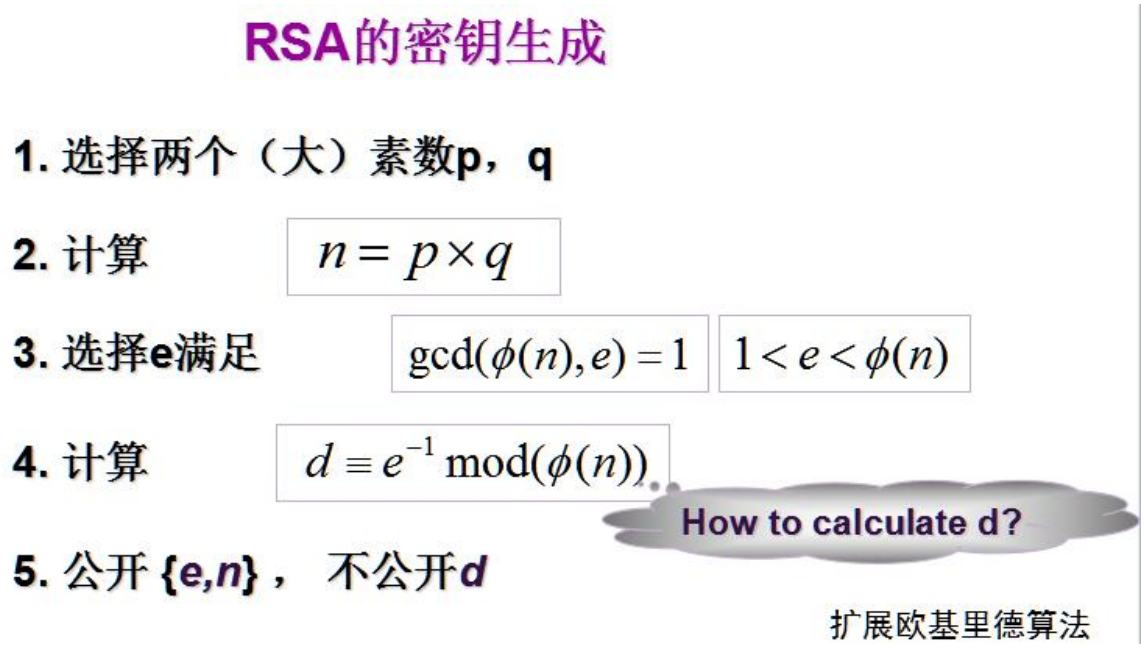
$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$



,.

◎（因为不知道生成 n 的 p、q，所以基于大数的因式分解很难，所以攻击者很难算出 d）



◎1994 年，Peter Shor 表示量子计算机可以考虑多项式时间（多项式时间），打破 RSA。

如果 n 是 300 位或更短，则可以在个人计算机上在几个小时内计算

截至 2008 年，通用因式分解算法所考虑的最大（已知）数字为 663 位长（参见 RSA-200），使用了最先进的分布式实现。

下一个记录可能是 768 位模数

,.

◎

4 SECURITY OF RSA

THE LAST BUT NOT LEAST

- 秘密研制于上个世纪60年代初，用于控制战略导弹的发射
- 70年代被麻省理工学院等高校的教授独立发明
- 建立在数论，特别是很难对大素数之积进行因式分解的基础之上
- 一旦量子计算机投入实用，将给目前的公共加密技术带来极大威胁

◎从最早的开始到现代，实际上所有的密码系统都是基于替代和排列的基本工具。公钥算法基于数学函数，而不是基于替换和置换。

公钥密码可以用于：加密/解密、电子签名、密钥交换/管理。

应该提到一些关于 PKC（公钥密码体制）的常见**误解**：

PKC 比对称密钥加密更安全（×）

PKC 是一种通用技术，所以传统密码已过时（×）

当使用 PKC 时，密钥分发是没有意义的

◎MAC 和加密之间的区别：

MAC 功能与加密类似：发送方和接收方共享公共密钥

不同：MAC 功能不必是可逆的（可逆）；MAC 功能不提供数字签名机制

,.

◎使用哈希函数的原因：（P241~P242）

◎哈希函数的要求

1.H（x）可以应用于任何大小的数据 x 的块

2.H（x）产生固定长度的输出

3.对于任何给定的 x，H（x）是相对容易计算的，使得硬件和软件实现都可行

4.单向性质：对于任何给定代码 h，计算上不可能找到 x 使得 H（x）= h。

5.弱碰撞阻力：对于任何给定块 x，计算上不可能找到

6.抗冲突性：计算上不可能找到任何对（x，y），使得 H（x）= H（y）

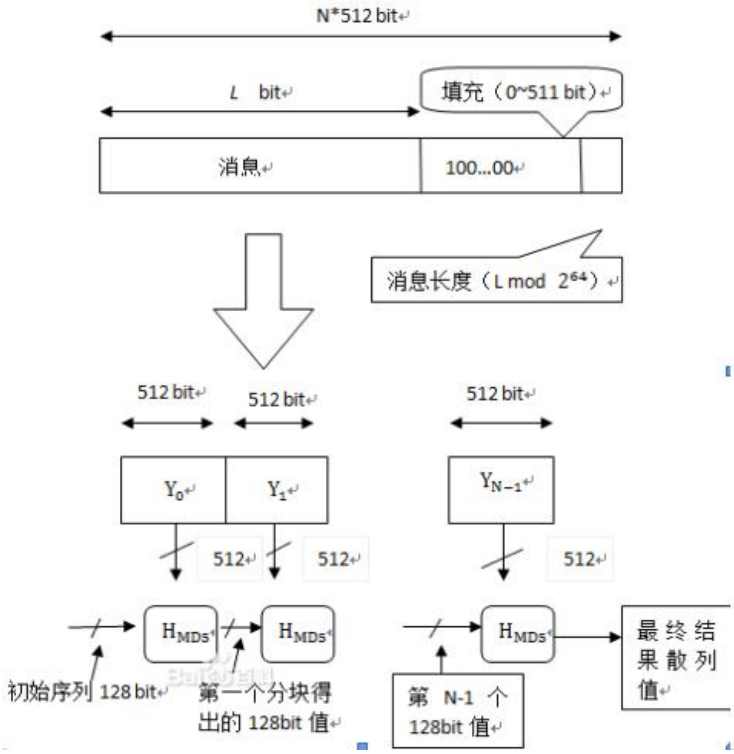
◎哈希算法：（P248~P249）

涉及重复使用压缩函数 f。

需要两个输入：来自上一步的 n 位结果（链接变量）和 b 位分组。最后产生 n 位输出

◎MD5：（P251）

它采用任意长度的消息作为输入（但阻塞它们各 512 位），并产生 128 位消息摘要作为输出



◎

Process of MD5

MD5的过程包括5个阶段：

Phase 1.附加填充位:

- **消息被填充以使其长度以位为单位**

$$length \equiv 448 \bmod 512$$

Phase 2. 追加长度

- **将原始消息的比特长度（填充之前）的64比特表示附加到阶段1的结果。**

第一, 二阶段结束后 **前两个阶段的结果产生为512位的整数倍的消息**

Phase 3. 初始化MD缓冲区

- 使用128位缓冲区来保持
  - ❖ MD5的中间结果
  - ❖ MD5的最终结果

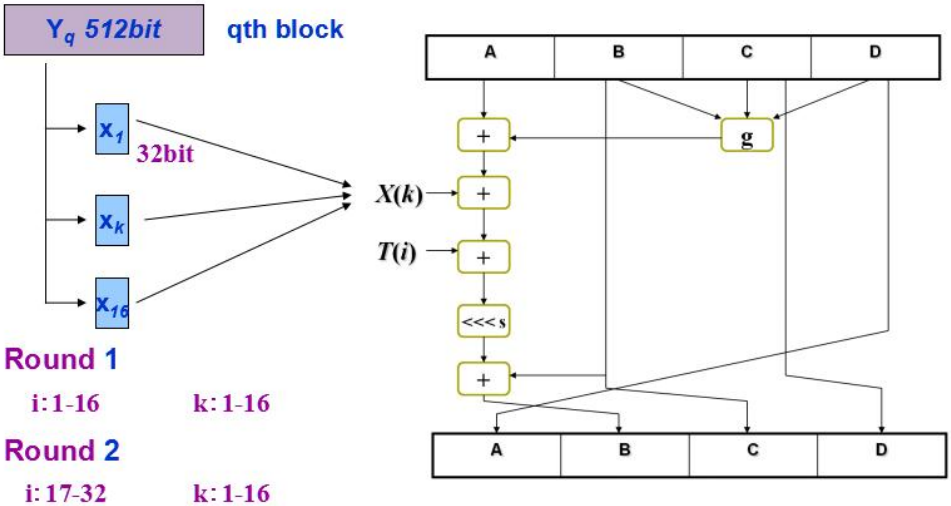
MD缓冲区: 缓冲区表示为四个32位寄存器（A，B，C，D），它采用初始值：

A	67452301
B	EFCDAB89
C	98BADCFE
D	10325476

Phase 4. 以512位块为单位的过程消息：：

- 使用复合压缩函数来处理512块
- processed in 4 rounds with different function

Round	Primitive Function	Definition
1	F(b, c, d)	$(b \wedge c) \vee (b \wedge d)$
2	G(b, c, d)	$(b \wedge d) \vee (c \wedge d)$
3	H(b, c, d)	$b \oplus c \oplus d$
4	I(b, c, d)	$c \oplus (b \vee d)$



Each round (I, H, G, and F) consists of sequence of 16 steps

Each step is of the form

基本算式

$$b + \left(\left(a + g(b,c,d) + X(k) + T(i) \right) <<< s \right)$$ where

- g(b, c, d) is one of I, H, F, G

$$X(k) = M\left(q \times 16 + k\right)$$

is the *k*th 32-bit word in the *q*th 512-bit block of the message

$$T(i) = 2^{32} \times abs\left(\sin\left(i\right)\right)$$

- *s* takes different values in the different rounds

For example: 7, 12, 17, 22

Phase 5. output 128bit digest

SUMMARY OF MD5

$$CV_0 = IV$$

$$CV_{q+1} = SUM_{32}\left(CV_q, RF_I\left[Y_q, RF_H\left[Y_q, RF_G\left[Y_q, RF_F\left[Y_q, CV_q\right]\right]\right]\right]\right)$$

$$0 \le q \le L$$

$$MD = CV_L$$

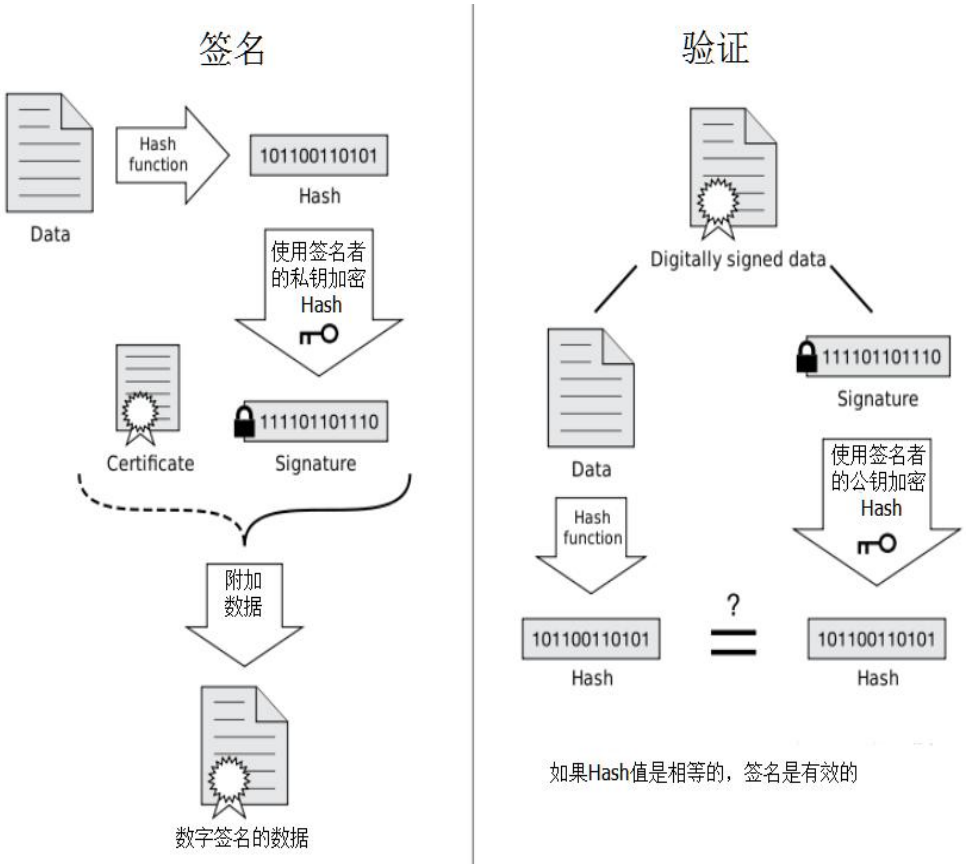
,.

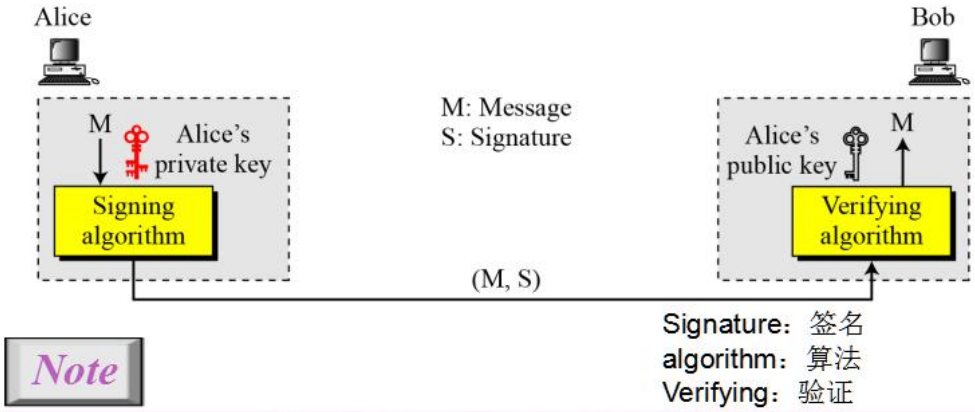
◎数字签名或数字签名方案是用于展示数字消息或文档的真实性（真实性）的数学方案。

有效的数字签名给予接收者相信该消息是由已知发送者创建的并且在传输中没有被改变的理由。

数字签名通常用于软件分发，金融交易，以及在其他需要检测修改的情况下。

◎



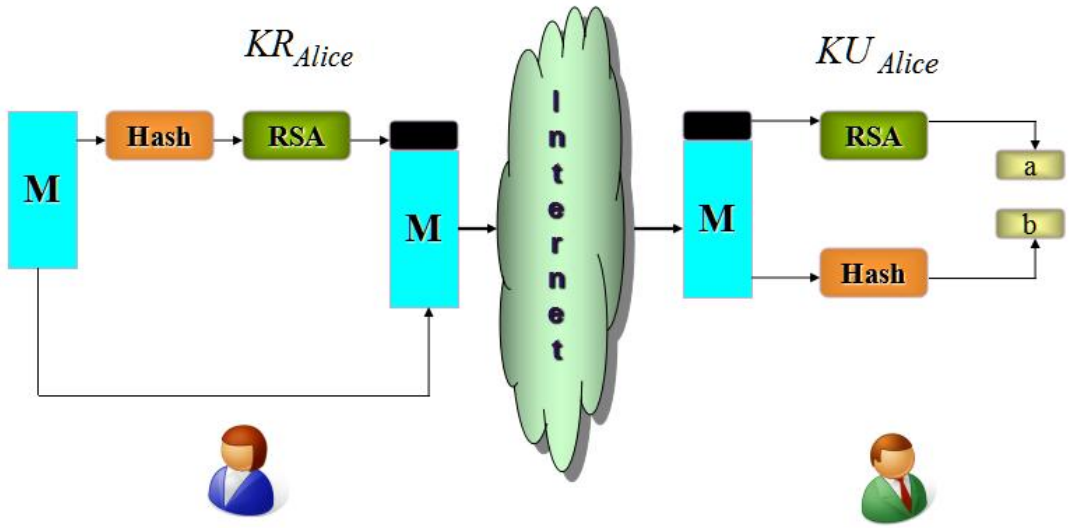


Note

数字签名需要公钥系统。
签名者用她的私钥签名;
验证者用签名者的公钥验证。

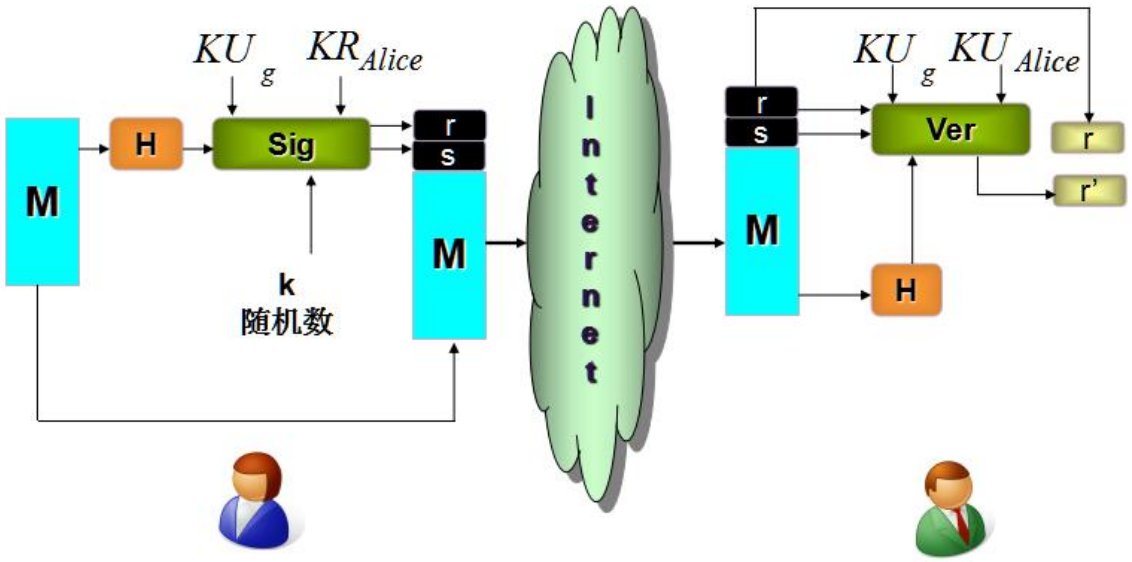
◎数字签名提供安全服务：消息认证；消息完整性；不可否认性；保密性；

- RSA数字签名方案



,.

- DSS: Digital Signature Standards（数字签名标准）



◎

Kerberos：概述

Kerberos（发音为/kɛərbərəs/）是计算机网络认证协议，其允许通过非安全网络通信的节点以安全的方式相互证明其身份。

它也是由 MIT 发布的一套实现此协议的免费软件

MIT 开发了 Kerberos 以保护雅典娜项目提供的网络服务

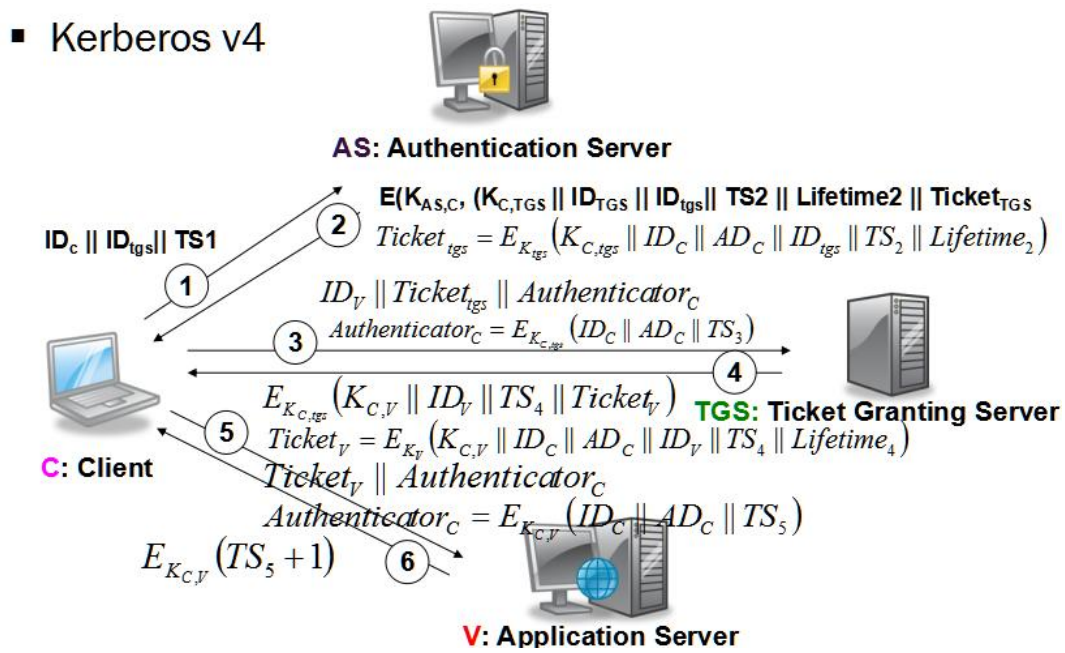
存在协议的几个版本；版本 1-3 仅在 MIT 内部发生。

版本 4 是在 20 世纪 80 年代末出版的

版本 5 在 1993 年作为 RFC 1510 出现

◎

,.



一个数字签名需要一个公钥系统。

她的签名私钥；验证者验证的签名者的公钥。

Kerberos 服务器；许多客户；多个应用服务器

这个 Kerberos 环境要求

Kerberos 服务器必须有用户的 ID 和用户密码

Kerberos 服务器必须共享一个密钥，每个服务器。

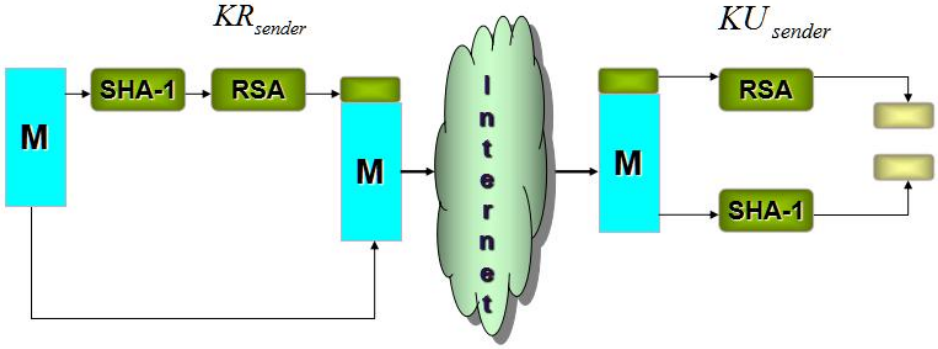
这样的环境被称为一个域

◎**X.509** 是 ITU-T（国际电信联盟）为一个公共密钥基础设施（PKI）标准（公钥基础设施）为单一登录（SSO）和特权（权益）管理基础设施（PMI）。

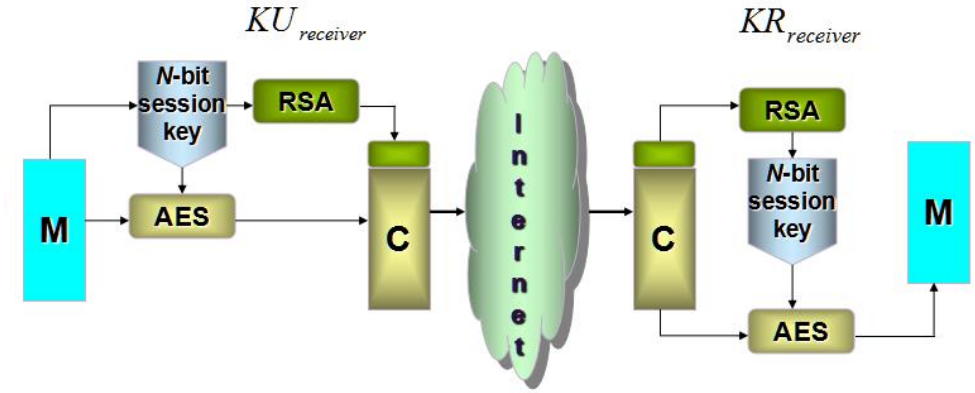
X.509 证书的规定，在其他的事情中，公钥证书的标准格式（证书）、证书撤销列表（撤回），属性证书和证书路径验证（确认）算法。

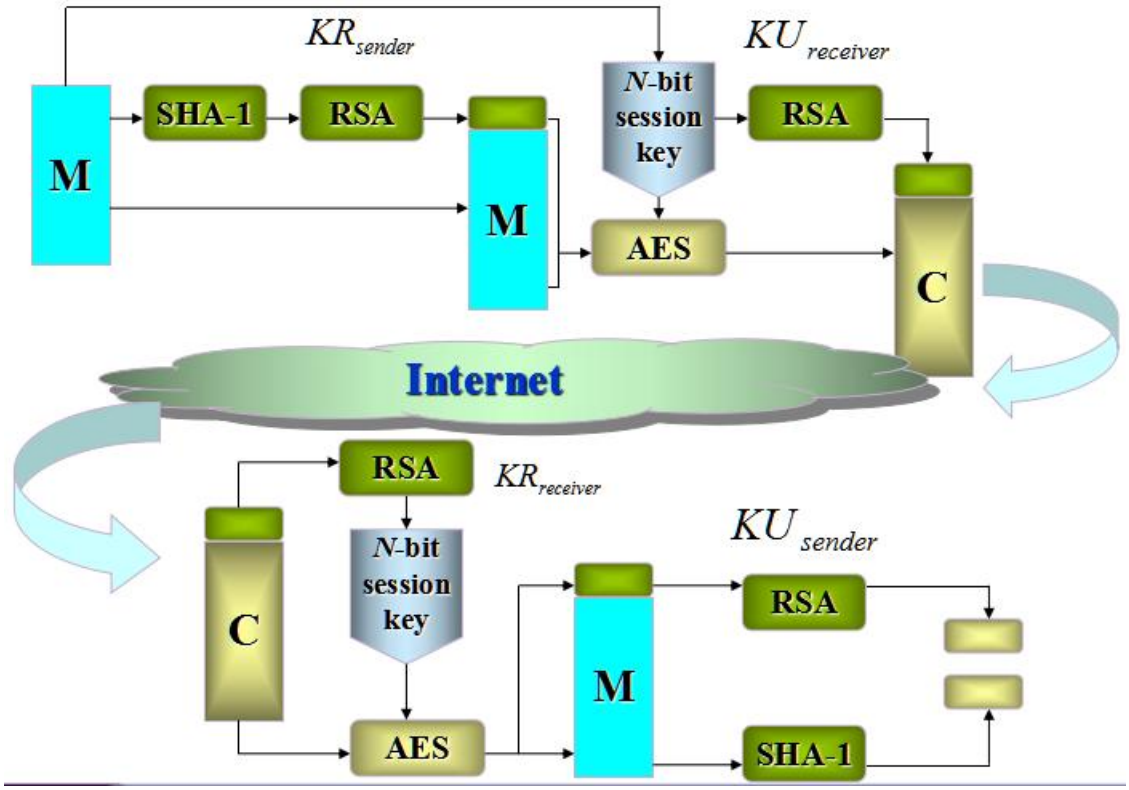
◎**PGP** 是一个提供加密隐私和身份验证的计算机程序。通常用于对电子邮件进行签名，加密和解密，以增加电子邮件通信的安全性。它是由 Philip Zimmermann 在 1991 年创建的。PGP 和其他类似产品遵循 OpenPGP 标准（RFC 4880）用于加密和解密数据。

- 使用PGP进行身份验证



- 使用PGP的保密性


$KU_{receiver}$
$KR_{receiver}$
N-bit session key
RSA
M
AES
C
Internet
C
RSA
N-bit session key
AES
M


KR_{sender}
$KU_{receiver}$
M
SHA-1
RSA
M
N-bit session key
RSA
AES
C
Internet
RSA
$KR_{receiver}$
C
N-bit session key
AES
KU_{sender}
M
RSA
SHA-1