

密码学基本概念

两种加密形式

- 传统加密：又称为**对称加密**、单钥加密，安全性在于保持算法本身的保密性
- 现代加密：又称为**非对称加密**、公钥加密，把算法和密钥分开，密码算法公开，密钥保密，安全性在于保持密钥的保密性

基本概念

- 明文/密文
- 加密/解密
- 密码算法/密码：用来加密和解密的数学函数
- 密钥：密码算法中的一个变量， $D_{Kd}(E_{Ke}(m)) = m$

密码学的基本模型

- 密码编码学
- 密码分析学
- 密码编码学和密码分析学统称为密码学

密码编码学

- 研究各种加密方案的学科称为密码编码学
- 密码编码学系统有三个独立的特征：
 1. 转换明文为密文的运算类型，基于两种原理：
 - **置换**：将明文中的元素重新排列
 - **代换**：将明文中每个元素映射成为另外一个元素
 - 原则是不允许丢失信息，即所有的运算都是可逆的
 2. 所用的密钥数：
 - **对称密码**：发送方和接收方使用相同的密钥
 - **非对称密码**：发送方和接收方使用不同的密钥
 3. 处理明文的方法：
 - **分组密码/块密码**：每次处理一个输入分组，相应地输出一个输出分组
 - **流密码/序列密码**：连续地处理输入元素，每次输出一个元素
- 无条件安全和计算安全：
 - 无条件安全：无论有多少可以使用密文，都不足以唯一地确定由该体制产生密文所对应的明文，则加密机制是无条件安全的
 - 计算安全：满足以下条件之一：
 - 破译密码的代价大于加密数据本身的价值
 - 破译密码的时间超过了密文信息的生命期

古典密码

代换技术

- 代换技术是将明文字符替换成其它字母、数字或者符号的方法
- **Caesar密码**: $c = (m + 3) \text{ Mod } 26$, 单表代换密码, 只有25个密钥 (1~25)
- **秘钥词密码**: 设一个秘钥词放在前面, 其余字母按顺序排列, 单表代换密码
- **Playfair密码**: 多表代换密码, 将明文中的双字母作为一个单元并将其转换为密文的双字母音节, 即将单字母映射关系变为了一个字母对到另一个字母对的映射关系

- 构造密钥词的方法是:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- 加密规则是 (一次加密两个字母) :

- 如果该字母对是两个相同的字母, 添加一个填充字
 - 例如x, 先把balloon变成ba lx lo on; 4个字母对
- 落在矩阵同一行的明文字母对中的字母, 由其右边的字母代换。
 - 例如: ar被代换为RM
- 落在矩阵同一列的明文字母对中字母, 由其下面的字母来代换。
 - 例如: mu被代换成CM
- 其它的明文字母对中字母按照如下方法代换:
它所在的行是该字母所在的行, 列是另外一个字母所在的列
 - 例如: hs被代换为BP

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- **Hill密码**: 多表代换密码, 将m个连续的明文字符代换成m个密文字符, 如下图:

用矩阵表示 $C=KP \bmod 26$

- C是密文, K是 3×3 的加密密钥矩阵, P是明文

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} \bmod 26$$

- n维矩阵能够隐藏 $(n-1)$ 大小的字母对的频率特征

- **Vigenere密码**: 是使用一系列Caesar密码组成密码字母表的加密算法, 属于多表代换密码

- **Vernam密码**: 选择一个与明文毫无统计关系并且和明文一样长的密钥, 其运算是基于二进制数据而非字母的

- $c_i = p_i \oplus k_i$
- 一次一密 (One-time Pad; OTP), 基于Vernam密码的改进方案, 使用与消息一样长且无重复的随机密钥来加密消息, 无条件安全

置换技术

- 置换密码是通过置换形成新的排列
- 单步置换还是容易被识破, 一般采用多步置换密码就安全多了
- 转轮机采用多层加密原理

破译举例

- 穷举法
- 频率分析法: 单码/双码/三码统计特征

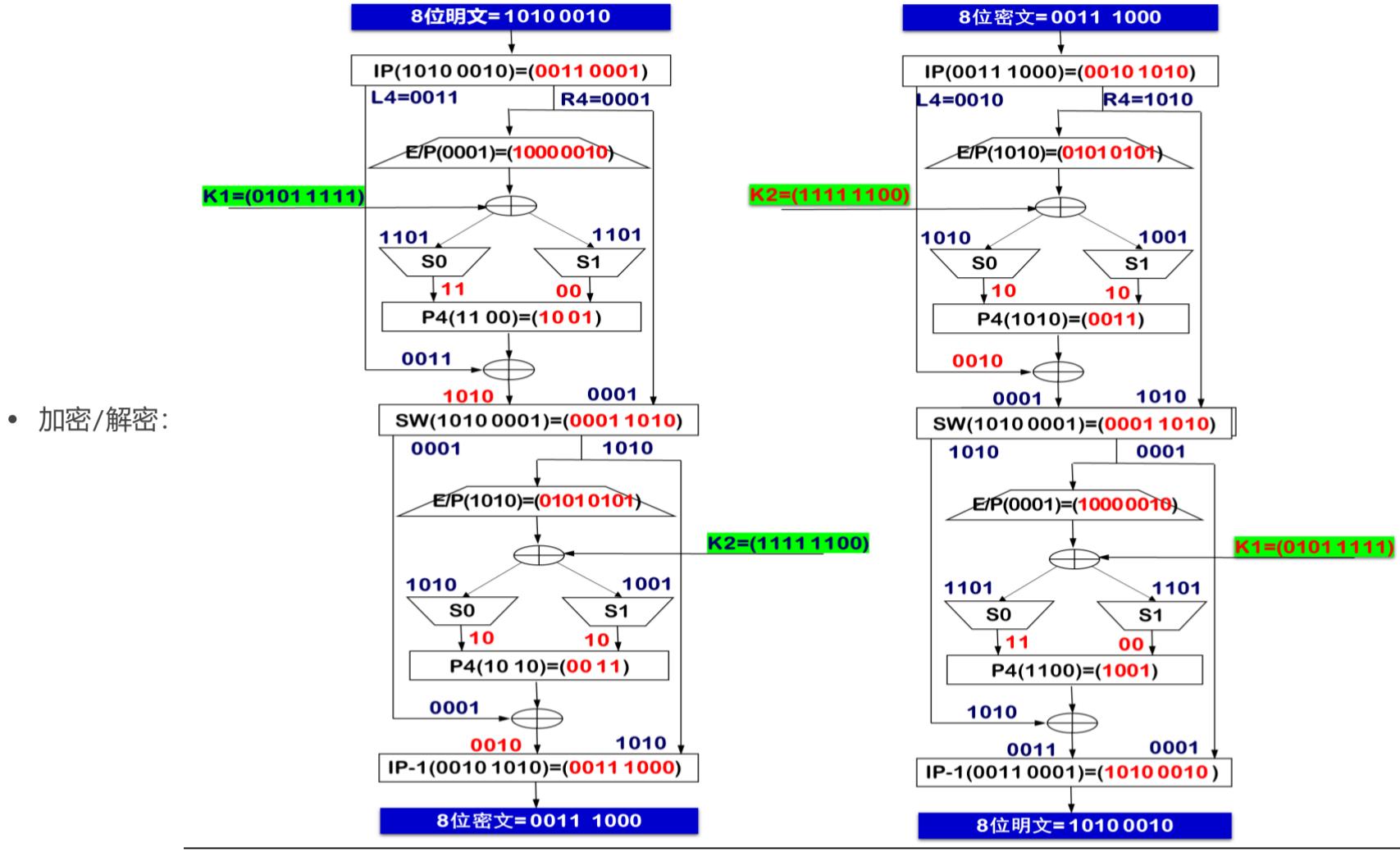
对称密码算法

简介

- 加密和解密使用相同的密钥: $K_E = K_D$
- 密钥必须使用秘密的信道分配

S-DES

- 简化DES(S-DES)是为教学使用的一个加密算法，与DES有着相似的性质和结构，但是参数要小很多，便于理解
- 加密/解密算法的输入为一个8位明文/密文组和一个10位密钥，输出为8位密文组/明文组



- 加密/解密：

- 加密流程和解密流程几乎相同，不同之处在于密钥 K_1, K_2 输入的位置

- 加密过程的函数：

- IP：初始置换函数， $IP(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8) = (n_2, n_6, n_3, n_1, n_4, n_8, n_5, n_7)$
- E/P：扩展位宽的置换函数， $E/P(n_1, n_2, n_3, n_4) = (n_4, n_1, n_2, n_3, n_2, n_3, n_4, n_1)$
- S盒：

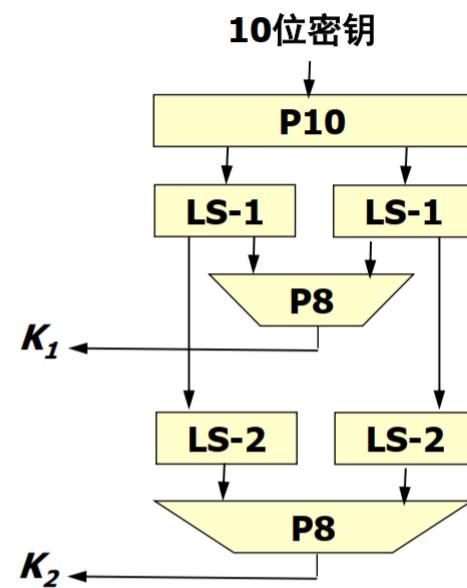
- 第1、4位作为二进制数决定S盒的行
- 第2、3位作为二进制数决定S盒的列
- 输出即是二进制的2位输出

$$S_0 : \begin{array}{c} \begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 & 1 & 0 & 3 & 2 \\ 1 & 3 & 2 & 1 & 0 \\ 2 & 0 & 2 & 1 & 3 \\ 3 & 3 & 1 & 3 & 2 \end{matrix} \end{matrix} \\ \begin{matrix} & 0 & 1 & 2 & 3 \\ \begin{matrix} 0 & 0 & 1 & 2 & 3 \\ 1 & 2 & 0 & 1 & 3 \\ 2 & 3 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 3 \end{matrix} \end{matrix} \end{array}$$

- P_4 ：置换函数， $P_4(n_1, n_2, n_3, n_4) = (n_2, n_4, n_3, n_1)$
- SW：交换函数，SW将输入的左4位和右4位交换
- IP^{-1} ：末尾置换函数， $IP^{-1}(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8) = (n_4, n_1, n_3, n_5, n_7, n_2, n_8, n_6)$

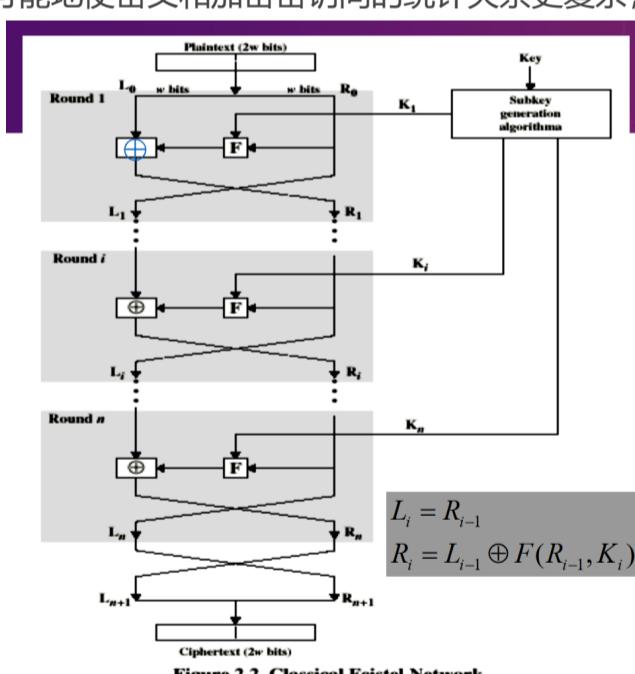
- 子密钥产生过程中的函数：

- S-DES依赖于收发双方共享的10位密钥，它产生的两个8位子密钥分别用在加密和解密的不同阶段。
- 10位密钥即($k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}$)。
- 置换P10定义为
 $P10(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)$
- LS-1表示前5位和后5位分别循环左移1位。
- LS-2表示前5位和后5位分别循环左移2位。
- 置换P8定义为
 $P8(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_6, k_3, k_7, k_4, k_8, k_5, k_{10}, k_9)$
- 由此可以计算得到子密钥K1和K2。



Feistel密码结构

- 现在使用的对称分组密码算法都基于Feistel分组密码结构的
- Feistel建议使用乘积密码的概念来逼近简单代换密码
- 乘积密码是指依次使用两个或以上的基本密码
- Feistel 建议交替使用代换和置换
- 混淆和扩散：
 - 扩散**是指使明文的统计特征消散在密文中，让每个明文数字尽可能地影响多个密文数字
 - 混淆**是尽可能地使密文和加密密钥间的统计关系更复杂，以挫败推导出密钥的企图
-



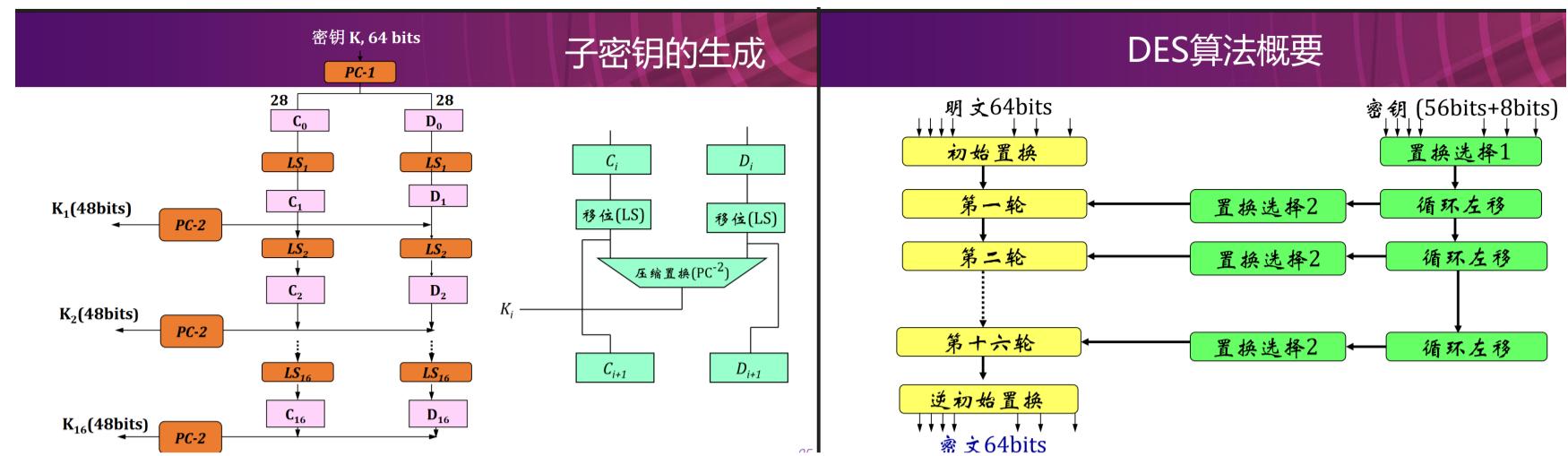
Feistel密码结构

- 输入是长度为 $2w$ 位的明文组和密钥K
- 明文组被分成两个部分：L0和R0；这两半数据经过n轮迭代后组合成密文组
- 第*i*轮迭代的输入来自于上轮迭代的输出
- 子密钥Ki是由整个密钥K推导出来的
 - Ki不同于K，也互不相同

- Feistel密码每轮迭代都有相同的结构

DES

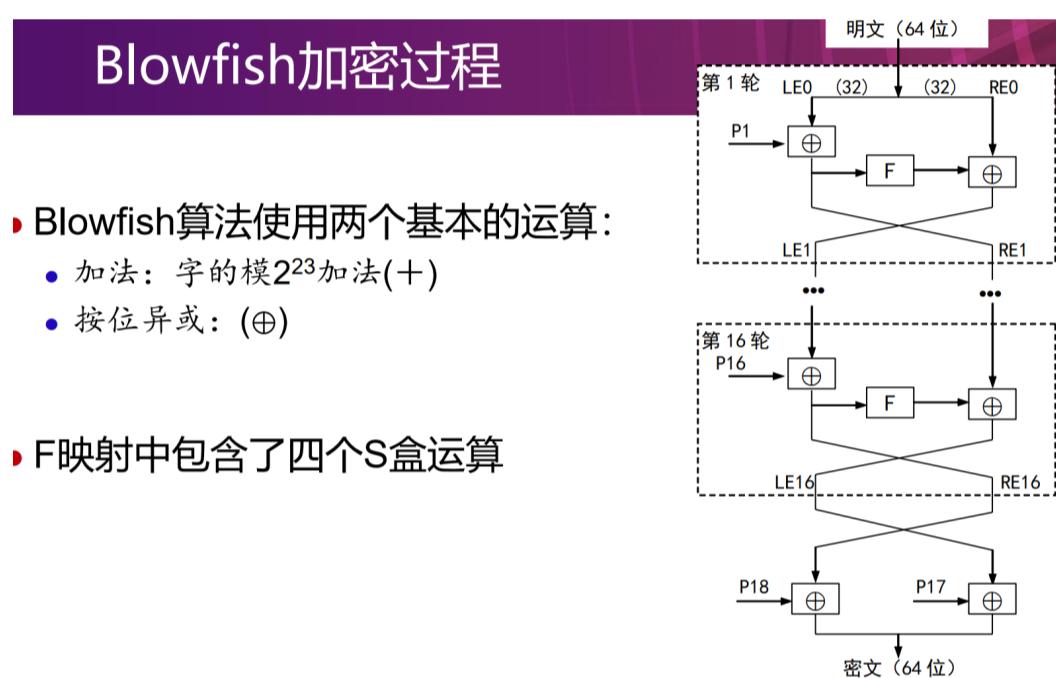
- DES采用分组加密，是一种对称密钥算法
- 分组长度是64 bits
- 除了初始置换和末尾置换，DES的结构与Feistel密码结构完全相同



- E 盒扩展置换：将 R_i 从 32 位扩展到 48 位，输入的一位影响下一步的两个替换，使得输出对输入的依赖性传播得更快，密文的每一位都依赖于明文的每一位
- S 盒代换选择：将 48 比特压缩成 32 比特

最常用的对称密码

- 3DES：三重DES加密，密钥长度为112比特。
- Blowfish：分组长度是64位，密钥长度可以在32 ~ 448位之间变化



- Blowfish 算法使用两个基本的运算：
 - 加法：字的模 2^{23} 加法 (+)
 - 按位异或：(\oplus)
- 加密过程：
- F 映射中包含了四个S盒运算

- Blowfish 算法的子密钥和 S 盒都是用 Blowfish 算法本身生成的，这使得数据完全不可以辨认，对它的密钥分析也就异常困难
- Blowfish 算法每轮运算都是对数据的左右两个部分同时执行运算，与古典 Feistel 结构不同；这使得密码的强度又增强了

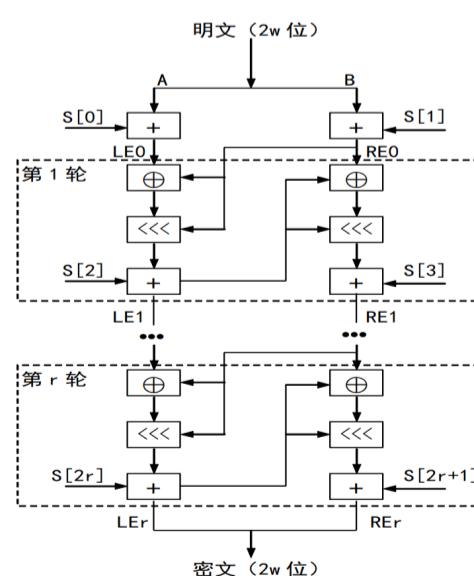
- RC5：
- RC5 的分组长度可以位 32、64 或者 128 位，密钥长度则可取 0~2040 位
- RC5 是一个由三个参数确定的加密算法族：

- w：分组长度
- r：迭代次数
- b：密钥 K 的 8 位字节数

o

RC5 使用了下面三种运算：

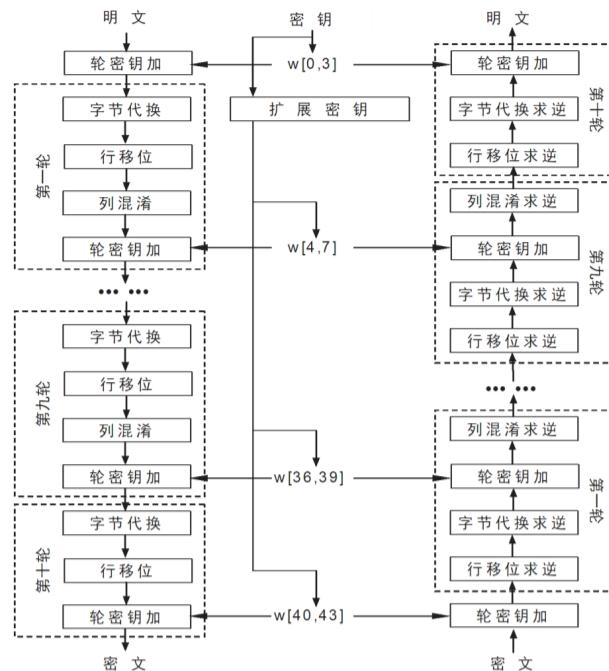
- 加法：
字的模 2^w 加法 (+)
- 按位异或：(\oplus)
- 循环左移：
字 \times 循环左移 (<<<)



- AES：AES 的分组长度 128 位，密钥长度 128 位
 - AES 不是 Feistel 结构，AES 的每一轮都使用代换和置换并行的处理整个数据分组

- AES的每一轮迭代都包括四个阶段：
 - 字节代换：用一个S盒完成分组中的字节代换
 - 行移位：一个简单的置换
 - 列混淆：算术代换
 - 轮密钥加：利用当前分组和扩展密钥的一部分进行按位异或

◦



- 密钥被扩展成44个32位字所构成的数组
- 仅仅在轮密钥加阶段使用密钥；轮密钥加实际上是一种Vernam密码

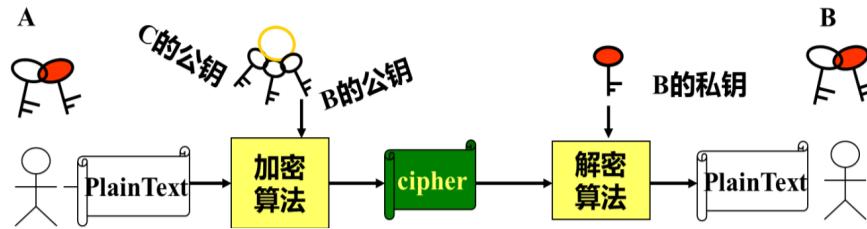
非对称密码算法

公钥密码原理

- 对称密码的缺陷：
 - 密钥必须经过安全的信道分配
 - 无法用于数字签名
 - 密钥管理复杂
- 公钥密码是基于数学函数而不是代换和置换
- 公钥密码的组成部分：
 - 明文：可读的信息，做为加密算法的输入
 - 加密算法：对明文进行的各种变换
 - 公钥/私钥：一个用于加密，一个用于解密。加密算法执行的变换依赖于公钥和私钥
 - 密文：加密算法的输出，不可读信息
 - 解密算法：根据密文和相应的密钥，产生出明文
 - 会话密钥K_s
 - 用户A的公钥 K_{Ua}
 - 用户A的私钥 K_{Ra}
- 符号说明：
 - E_{KUa}[P]：用A的公钥对明文P加密
 - E_{KRa}[P]：用A的私钥对明文P加密
 - D_{KUa}[C]：用A的公钥对密文C解密
 - D_{KRa}[C]：用A的私钥对密文C解密
- 公钥密码体制：每个用户产生一对密钥：用于加密和解密。其中一个密钥存于公开的寄存器或者文件中，即公钥；另外一个密钥是私有的，称为私钥
 - 公钥公开，用于加密和验证签名
 - 私钥保密，用作解密和签名
 - 利用这种方法，通信各方皆可以访问公钥，而私钥是个通信方在本地产生的，所以不必进行分配
 - 只要系统控制了私钥，那么它的通信是安全的
 - 任何时刻，系统都可以改变自己的私钥，而公布相应的公钥代替原来的公钥

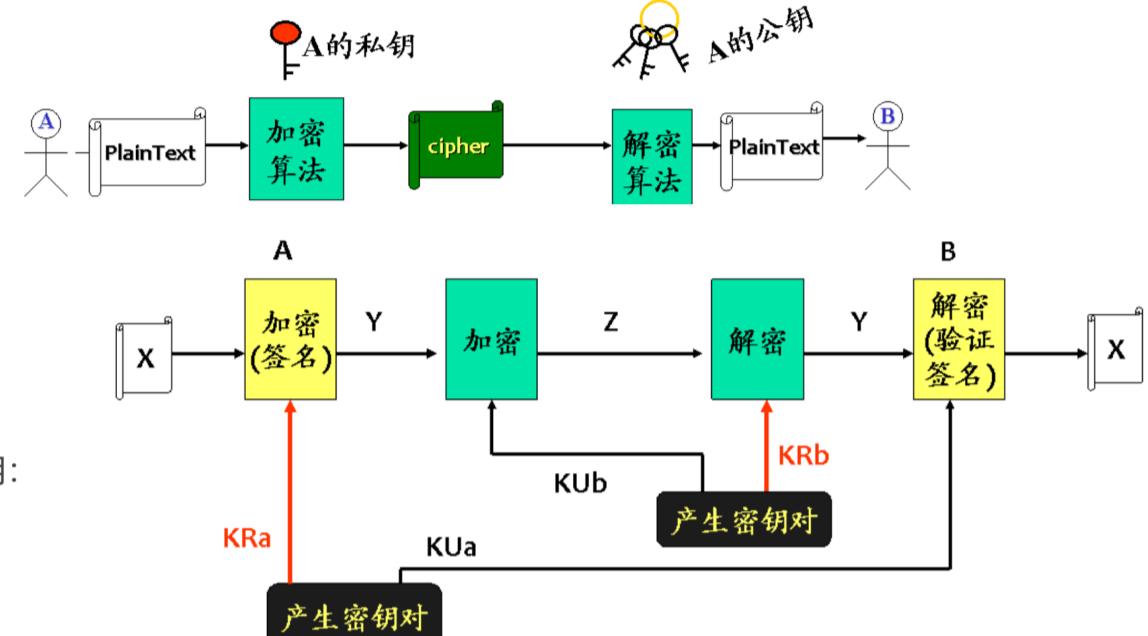
- A向B发送消息，用B的公钥加密
- B收到密文后，用自己的私钥解密
- 任何人向B发信息都使用同一个密钥(B的公钥)加密。没有人可以得到B的私钥，只有B可以解密

◦ 加密原理示例图：



- A向B发送消息，用A的私钥加密(签名)
- B收到密文后，用A的公钥解密(验证)

◦ 签名原理示例图：



◦ 数字签名和加密同时使用：

$$Y = E_{KRa}(X), \quad Z = E_{KUb}[Y] = E_{KUb}[E_{KRa}(X)] \\ Y = D_{KRb}(Z), \quad X = D_{KUb}[Y] = D_{KUb}[D_{KRa}(Z)]$$

• 公钥密码的数学原理：陷门单向函数

- 单向函数是求逆困难的函数；单向陷门函数，是在不知陷门信息下求逆困难的函数，当知道陷门信息后，求逆是易于实现的
- 单向陷门函数 $f(x)$ ，必须满足以下三个条件：
 - 给定 x ，计算 $y=f(x)$ 是容易的
 - 给定 y ，计算 x 使 $y=f(x)$ 是困难的
 - 存在 δ ，已知 δ 时对给定的任何 y ，若相应的 x 存在，则计算 x 使 $y=f(x)$ 是容易的
- 仅满足前两条的称为单向函数，第三条为陷门性， δ 称为陷门信息

• 公钥密码系统的应用：

公钥密码系统有三种用途：

- 加密/解密
- 数字签名：如果电子文件都需要签名，如何能够确保数字签名是出自某个特定人，而且通信双方无异议
 - 发送方用自己的**私钥**签署报文，接收方用对方的公钥验证对方的签名
- 密钥交换：双方协商会话密钥，用于对称密钥数据加密

公钥密码算法	加密/解密	数字签名	密钥交换
RSA	Y	Y	Y
Diffie-Hellman	N	N	Y
DSA	N	Y	N

- 在实际应用中，公钥密码目前仅局限于密钥管理和数字签名

◦

对称密码 vs 非对称密码

- 非对称的公钥密码学，使用两个独立的密钥，这对于密钥分配、数字签名、认证技术等有深远影响；但存在一些误解
- 误解一：从密码分析角度看，公钥密码比传统密码更安全
 - 任何加密方法的安全性依赖于密钥长度和破译密文所需要的计算量
 - 从抗击密码分析的角度看，不能简单地说传统密码和公钥密码哪个更安全
- 误解二：公钥密码是一种通用方法，传统密码已经过时
 - 由于公钥密码需要大量计算，仅限于密钥管理和签名这类应用中，所以基本不太可能取代传统密码
- 误解三：传统密码中与密钥分配中心的握手是一件异常麻烦的事情，而公钥密码实现密钥分配则是非常简单的
 - 使用公钥密码也需要某种形式的协议，通常也包括一个中心代理，所包含的处理过程即不比传统密码简单，也不比其更有效

- 数论基础：
 - 欧拉函数： $\phi(n)$ ：n是正整数， $\phi(n)$ 是比n小且与n互素的正整数个数，如果p是素数，则 $\phi(p)=(p-1)$
 - 欧拉定理：若整数m和n互素，则 $m^{\phi(n)} \equiv 1 \pmod{n}$ → $M^{k\phi(n)+1} \equiv m \pmod{n}$

RSA算法

- RSA体制是一种分组密码，其明文和密文都是0~n-1之间的整数，通常n的大小为1024位二进制数或者309位十进制数

- 密钥产生
 - 取两个大素数 p, q, 保密；
 - 计算 $n=pq$, 公开 n;
 - 计算欧拉函数 $\phi(n) = (p-1)(q-1)$;
 - 任意取一个与 $\phi(n)$ 互素的小整数 e，
即 $\gcd(e, \phi(n))=1$; $1 < e < \phi(n)$
- 密钥的产生：
 - 寻找 d, $d < \phi(n)$, 使得
 $de \equiv 1 \pmod{\phi(n)}$, 即 $de = k\phi(n) + 1$
 - 公开 (e, n)
 - 将 d 保密，丢弃 p, q。
- 公开密钥: $KU=\{e, n\}$
- 秘密密钥: $KR=\{d, n\}$

① 设: $p=7, q=17$
② 则: $n=119$
③ $\Phi(n)=6 \times 16=96$
④ 选择 $e=5$
⑤ $5d=k \times 96+1$
⑥ 令 $k=4$, 得到
 $d=77$
⑦ 故知道:
⑧ $KU = \{5, 119\}$
⑨ $KR = \{77, 119\}$

- 利用：公钥 $KU=\{e, n\}$ 和私钥 $KR=\{d, n\}$
- 加密过程
 - 把待加密的内容分成k比特的分组， $k \leq \log_2 n$ ，并写成数字，设为M，则： $C = M^e \pmod{n}$
- 加密/解密过程：
 - 例如： $C=M^5 \pmod{119}$
 - 解密过程
 - $M = C^d \pmod{n}$
 - 例如： $M=C^{77} \pmod{119}$

- RSA算法举例1:

- 用RSA算法对下面数据实现加密和解密。
 - $p=5$; $q=11$; $e=3$; $M=9$

RSA算法举例1

- 密钥产生
 - 取两个大素数 p, q , 保密;
 - 计算 $n=pq$, 公开 n ;
 - 计算欧拉函数
 $\phi(n) = (p-1)(q-1)$;
 - 任意取一个与 $\phi(n)$ 互素的小整数 e ,
即 $\gcd(e, \phi(n)) = 1$; $1 < e < \phi(n)$
 - 寻找 d , $d < \phi(n)$, 使得
 $de \equiv 1 \pmod{\phi(n)}$,
即 $de = k\phi(n) + 1$
 - 公开 (e, n)
 - 将 d 保密, 丢弃 p, q 。
- 公开密钥: $KU = \{e, n\}$
- 秘密密钥: $KR = \{d, n\}$

• 设: $p=5, q=11$
 • 则: $n=5 \times 11=55$
 $\Phi(n)=(5-1)(11-1)=4 \times 10=40$

• 因为 $e=3$
 根据 $de = k\phi(n) + 1$
 故得 $3d=k \times 40 + 1$
 • 令 $k=2$, 得到 $d=27$
 • 故知道:
 $KU = \{3, 55\}$
 $KR = \{27, 55\}$

• 加密过程
 $C=M^e \pmod{n}$
 $=9^3 \pmod{55}=729 \pmod{55}=14$
 • 解密过程
 $M=C^d \pmod{n}$
 $=14^{27} \pmod{55}=9$

- RSA算法举例2:

- 假设Alice需要将明文“key”通过RSA加密后传递给Bob。

● 第一步：设计公钥和密钥

- 令 $p=3, q=11$, 得出 $n=p \times q=3 \times 11=33$
- $\Phi(n)=(p-1)(q-1)=2 \times 10=20$
- 取 $e=3$, 则 $e \times d \equiv 1 \pmod{\Phi(n)}$, 即 $3 \times d \equiv 1 \pmod{20}$; 取 $d=7$
- 从而可以设计出一对公私密钥,
 - 加密密钥（公钥）为: $KU = (e, n) = (3, 33)$
 - 解密密钥（私钥）为: $KR = (d, n) = (7, 33)$

● 第二步：明文信息数字化

- 假定明文英文字母编码表为按字母顺序排列数值
- 则得到分组后的key的明文信息为: 11, 05, 25。

字母	A	B	C	D	E	F	G	H	I	J	K	L	M
字母	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
码值	01	02	03	04	05	06	07	08	09	10	11	12	13
字母													
码值	14	15	16	17	18	19	20	21	22	23	24	25	26

● 第三步：明文加密

- Alice用加密密钥(3,33)将数字化明文分组信息加密成密文
- 由 $C \equiv M^e \pmod{n}$ 得到
相应的密文信息为:
11, 26, 16

$$C_1 \equiv (M_1)^e \pmod{n} = 11^3 \pmod{33} = 11$$

$$C_2 \equiv (M_2)^e \pmod{n} = 26^3 \pmod{33} = 26$$

$$C_3 \equiv (M_3)^e \pmod{n} = 16^3 \pmod{33} = 16$$

● 第四步：密文解密

- Bob收到密文后, 若将其解密, 只需要计算 $m_i = C_i^d \pmod{n}$
 - $M_1 = 11^7 \pmod{33} = 11$; $M_2 = 26^7 \pmod{33} = 05$; $M_3 = 16^7 \pmod{33} = 25$
 - 得到明文信息为: 11, 05, 25; 根据编码表将其转换为“key”

- RSA算法的安全性: 对RSA算法的攻击方法: 蛮力攻击、数学攻击、计时攻击

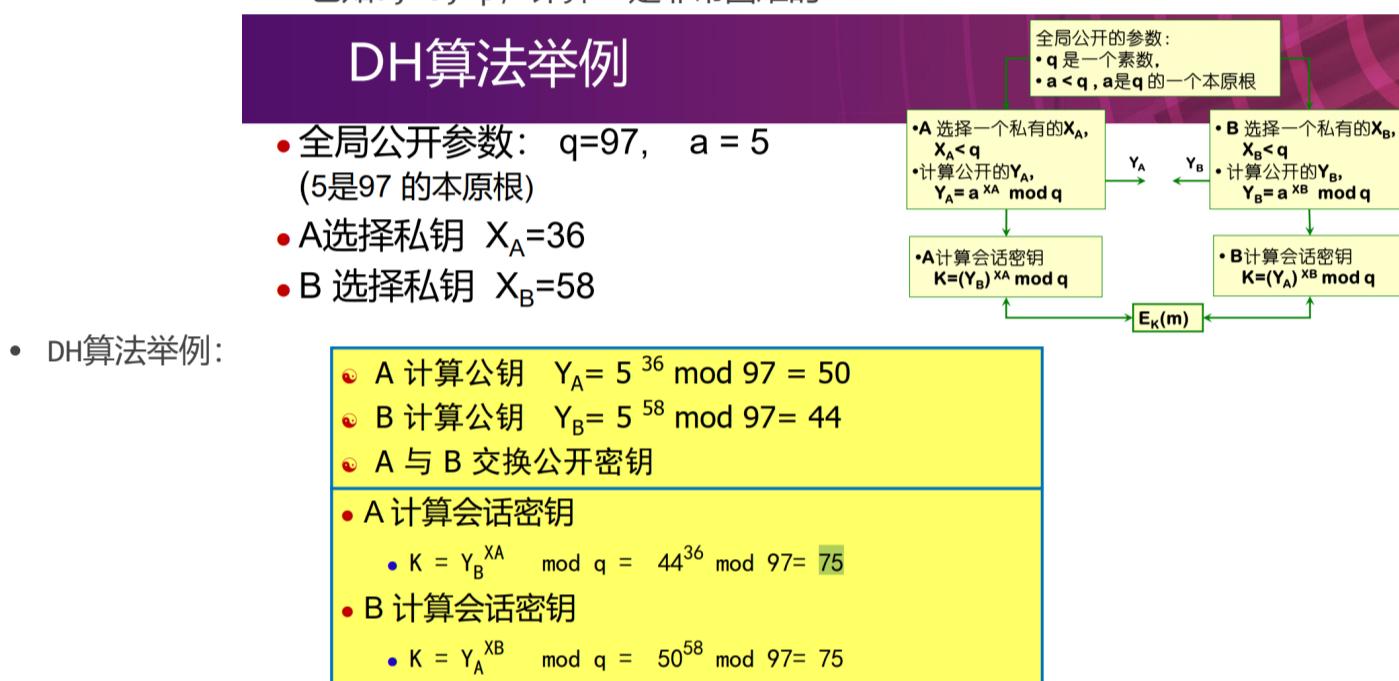
- 蛮力攻击: 对所有密钥都进行尝试
- 数学攻击: 实质是两个素数乘积(n)的因子分解
- 计时攻击: 攻击者可以通过记录计算机解密消息所用的时间来确定私钥

- RSA算法的性能:

- 软件实现比DES慢100倍
- 硬件实现比DES慢1000倍

DH密钥交换算法

- Diffie-Hellman密钥交换算法的目的是使两个用户能够安全地交换密钥，该算法本身也只局限于进行密钥交换
- 数学背景：
 - 本原根： a 是素数 p 的一个本原根，如果 $a \bmod p$, $a^2 \bmod p$, ..., $a^{p-1} \bmod p$ 是1到 $p-1$ 的排列，即各不相同，是整数1到 $p-1$ 的一个置换
 - 对于整数 b ($b < p$) 和素数 p 的一个本原根 a ,可以找到一个唯一的指数 i , 使得: $b \equiv a^i \bmod p$, 其中 $0 \leq i \leq (p-1)$
 - i 称为 b 的 以 a 为底 模 p 的离散对数或指数 ，记为 $\text{ind}_{a,p}(b)$
 - $\text{ind}_{a,p}(1) = 0$
 - $\text{ind}_{a,p}(a) = 1$
 - 对于 $b = a^x \bmod p$
 - 已知 a , x , p , 计算 b 是容易的
 - 已知 a , b , p , 计算 x 是非常困难的



其他公钥密码算法

- DSA: 数字签名算法, 算法的安全性是基于计算离散对数的难度
- 椭圆曲线密码系统
- RSA是事实上的标准

密钥的分配

- 存在两种加密方法：
 - 链路加密
 - 端到端加密
- 密钥分配方法就是将密钥发放给希望交换数据的双方而不让别人知道的方法

传统的对称密码分配

- 方法一：密钥由A选择，并亲自交给B
- 方法二：第三方选择密钥后亲自交给A和B
- 方法三：如果A和B以前或者最近使用过某个密钥，其中一方可用它加密一个新密钥后再发送给另外一方
- 方法四： A和B与第三方C均有秘密渠道，则C可以将一密钥分别秘密发送给A和B
- 方法一和二需要人工传送密钥，适用于链路加密；对于端到端加密，则使用密钥分配中心
- 密钥分配中心KDC模式：

- 假定每个用户与密钥分配中心KDC共享唯一的一个主密钥：
 - A有一个除了自己只有KDC知道的主密钥Ka
 - B有一个除了自己只有KDC知道的主密钥Kb
- • A和B获得一次性会话密钥Ks的工作过程
 - STEP1: A向KDC请求一个会话密钥以保护与B的逻辑连接；消息中有A和B的标识以及一个临时交互号N1
 - STEP 2: KDC用Ka加密的消息做出响应；此时，A可知：
 - 一次性会话密钥Ks，用于会话
 - 原始请求消息，含N1，使A可以做出反映
 - 此外，还含有两项给B的内容，用Kb加密了
 - 一次性会话密钥Ks 和 A的标识符IDA
 - STEP 3: A存下会话密钥Ks备用，将KDC消息中的后两项内容发给B，B也知道会话密钥了
- • 这样会话密钥就安全地发给了A和B

- 在网络规模很大的时候，密钥的分配功能不限定在单个的KDC上面，而是使用层次式的KDC
- 层次式KDC密钥分配使得主密钥分配的代价变小了。而且，如果一个本地KDC出错，或者被攻击了，破坏只是集中在一个区域中，不会影响全局

公钥的分配

- 公钥密码可用于下面两个方面：
 - 公钥的分配
 - 公钥密码用于传统密码体制的密钥分配
- 常用的公钥分配方法有四种：
 - 公开发布：缺点：任何人都可以伪造这种公钥的公开发布
 - 公开可访问目录：维护一个动态可访问的公钥目录可以获得更大程度的安全性；某个可信的实体或组织负责这个公开目录的维护和分配
 - 公钥授权：缺点在于公钥管理员就会成为系统的瓶颈

Kohnfelder最早提出不通过管理员，而用整数来交换密钥；此方法与公钥授权的安全性相同

 - 证书包含者公钥和其它一些信息，由证书管理员产生，并发给拥有相应私钥的通讯方
 - 通信一方通过传递证书将密钥信息传递给另外一方，其它通信各方可以验证该证书确实是证书管理员发出的
 - 公钥证书：

这种方法需要满足下面的要求：

 - 任何通信方可以读取证书并确定证书拥有者的姓名和公钥
 - 任何通信方可以验证该证书出自证书管理员，而不是伪造的
 - 只有证书管理员才可以产生并更新证书
 - 任何通信方可以验证证书的当前性

利用公钥分配传统密码的密钥

例如：A要和B通信，则执行：

- 简单的密钥分配：
 - A产生公/私钥对{KUa, KRa}，并将含有KUa和A表示的消息发给B。
 - B产生秘密钥Ks，用A的公钥对Ks加密后传给A。
 - A计算DKRa[EKUa[Ks]]得到秘密钥Ks。
 - 这样A和B就可以利用Ks和对称密码进行安全通信。

不过，这种协议容易受到主动攻击

- 具有保密性和真实性的密钥分配：

- | | |
|--|--|
| <ul style="list-style-type: none"> ◦ 下面这种方法即可以抗击主动攻击也可以抗击被动攻击 <ul style="list-style-type: none"> • 假设A和B已经交换了公钥 • A用B的公钥对含有A标识和临时交互号N1的消息加密，并发给B • B发送一条用A的公钥加密的消息，包括A的N1和B的临时交互号N2 <ul style="list-style-type: none"> • 因为只有B能够解密A发来的消息，所有本条消息中的N1可以使A确信其通信伙伴是B • A用B的公钥对N2加密，并返回B，这样可使B确信其通信伙伴是A | <ul style="list-style-type: none"> ◦ A选择密钥Ks，并将M=EKUb[EKRb[Ks]]发送给B ◦ B计算DKUa[DKRb[M]]得到密钥Ks ◦ A和B可以利用对称密码进行安全通信了 |
| <ul style="list-style-type: none"> ◦ 利用公钥密码进行密钥分配，也需要密钥分配中心KDC，KDC与每个用户共享一个主密钥，通过该主密钥加密实现会话密钥的分配 | |

Lettuce4 认证技术

消息认证

消息认证基本概念

- 泄密：将消息透露给没有合法密钥的接收方
- 传输分析
- 伪装
 - 内容修改：对消息内容的修改，包括插入、删除、转换和修改。
 - 顺序修改：对通信双方消息顺序的修改，包括插入/删除/重排。
 - 计时修改：对消息的延时和重放。
 - 发送方否认：发送方否认发送过某条消息。
 - 接收方否认：接收方否认接收过某条消息。
- 网络环境中的攻击：
 - 对付泄密、传输分析属于消息保密性范畴。
 - 对付伪装、内容修改、顺序修改、计时修改属于消息认证范畴。
 - 对付发送方否认可以使用数字签名。
 - 对付接收方否认需使用数字签名和为抗击此类攻击而设计的协议。
- **消息认证就是验证所收到的消息确实是来自真正的发送方且未被修改的消息**
- 任何消息认证都能在功能上看做两层：
 - 下面一层中有某种产生认证符的函数，认证符是一个用来认证消息的值
 - 上面一层将该函数作为原语，使得接收方可以验证消息的真实性

认证函数

- 认证函数：产生认证符的函数
- 可以分为三类：
 - 消息加密：将整个消息的密文作为认证符
 - 消息认证码MAC： MAC是消息和密钥的公开函数，它产生定长的值，该值作为认证符
 - Hash函数：它是将任意长的消息映射为定长的hash值的公开函数，以该hash值作为认证符

认证函数1：消息加密

- 对称加密：
 - 发送方A用A和B共享的密钥K，对发送到接收方B的消息M进行加密，如果没有其他人知道该密钥，就可以提供保密性
 - B可以在接收到消息后，用密钥K解密，确认该消息是由A发出的
 - 加密前对每个消息附加一个错误检测码，也称之为帧校验序列FCS或者校验和
 - A准备发送明文消息M，那么A将M作为函数F的输入，产生FCS，将FCS附加在M后并对M和FCS一起加密
- 对称加密的例子：
 - 在接收端，B解密其收到的信息，并将其看作是消息和附加FCS；B用相同的函数F重新计算FCS，若所得FCS和收到FCS相等，B认为消息是真实的
 - FCS和加密函数的顺序是很重要的

- 对称加密时需要先构造FCS，再加密，顺序不能变化
- 对称加密的协议：TCP协议，可以对除了IP报头之外的所有数据进行加密
- 公钥加密：
 - 公钥加密只提供保密性，不能提供认证
 - 如果既要提供保密性，又要提供认证，发送方A可以先用其私钥加密(数字签名)，然后用B的公钥加密
 - 这种方法的缺点是执行了四次附加的公钥算法运算
- **对称加密**
 - 提供保密性
 - 只有A和B共享密钥K
 - 提供认证
 - 只能发自A
 - 传输中未被改变
 - 需要某种数据形式/冗余
 - 不能提供数字签名
 - 接收方可以伪造信息
 - 发送方可以否认信息
- **公钥加密**
 - 提供保密性
 - 只有B拥有用于解密的密钥K_B，但是任何人都可用K_{Ub}对消息加密并假称是A
 - 提供认证和签名
 - 只有A拥有用于加密的密钥K_A
 - 传输中未被改变
 - 需要某种数据组织形式/冗余
 - 任何一方可用K_{Ua}来验证签名
 - 提供保密性、认证+签名
 - 提供保密性（因为K_{Ub}）
 - 提供认证和签名（因为K_A）

认证函数2：消息认证码MAC

- 消息认证码：
 - 消息认证码MAC也是一种认证技术，它利用密码生成一个固定长度的短数据块，并将该数据块附加在消息之后
 - MAC是消息和密钥的函数： $MAC = C_k(M)$
 - MAC函数与加密类似，区别就是MAC算法不要求可逆性，而加密算法必须可逆
 - A和B共享密码K
 - A向B发送消息，A计算MAC，将其附加在消息后面，一起发给接收方
- MAC认证的过程：
 - 接收方对收到的消息用相同的密钥K进行计算得到MAC，并与收到的MAC进行对比
 - 因收发双方共享密钥，MAC不能提供数字签名
 - 将同一消息广播给很多接收者的情况
 - 一个接收者负责验证真实性，告知其它人
 - 在信息交换中，接收者希望可以随机地对消息进行认证
 - 对明文形式的计算机程序进行认证很有意义的服务
- 使用MAC的情况：
 - 运行一个计算机程序，不需要每次都进行加密解密；只在需要保护程序完整性的时候才检验消息认证码MAC
 - 一些应用只关心消息的认证，而不关心消息的保密性
 - 例如：网络管理协议SNMPv3
 - 将认证和保密性分开，可以使层次结构更灵活
 - 例如：可以在应用层提供认证，在传输层提供保密性

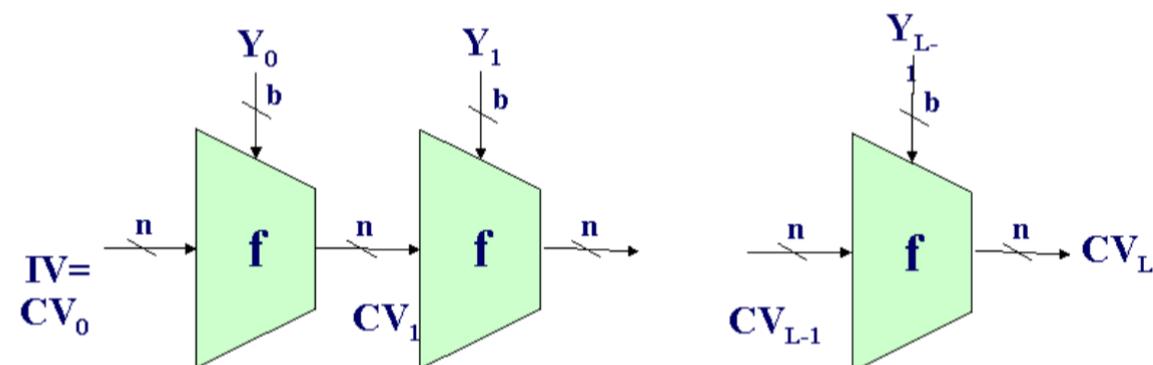
认证函数3：Hash函数

- 单向的Hash函数是消息认证码的一种变形
- hash函数的输入是长度可变的消息M，输出是长度固定的hash码
- hash函数不使用密钥，仅是消息M的函数，因此也称为消息摘要(Message Digest， MD)
- Hash码是所有消息位的函数，它具有错误检测能力
- 将hash码用于消息认证的多种形式：
 1. 用对称密码对消息以及hash码进行加密：提供了**认证和保密性**

2. 用对称密码仅对hash加密：仅提供**认证**（相当于MAC），对于不要求保密性的应用，会减少处理代价
 3. 用公钥密码和发送方的私钥对hash码加密：提供了**认证**和**数字签名**（数字签名有发送方的私钥所保证）
 4. 先用发送方的私钥对hash码进行加密，再用对称密码中的密钥对消息和上述加密结果进行加密：提供了**认证**、**数字签名**和**保密性**
 5. 假定通信双方共享公共的秘密值S，A将M和S联接后再计算hash值，并将其附加在M后面，由于B也知道S，B可以计算hash值，并验证其正确性：提供了**认证**
 6. 在假设（5）的基础上对整个消息和hash码加密：提供了**认证**和**保密性**
- 对hash函数的要求：
 - 可以操作任意大小的报文m
 - 生成的h长度固定
 - 单向性
 - 抗弱碰撞性
 - 抗强碰撞性

安全hash算法的一般结构

- Hash函数将输入消息分为L个固定长度的分组，每个分组长度为b位，最后一个分组不足b位时，需要填充成b位
- 输入中包含长度，增加了攻击的难度
- **Hash函数公式**
 - $CV_0 = IV = \text{初始 } n \text{ 位值}$
 - $CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \leq i \leq L$
 - $H(M) = CV_L$ (其中输入M由 Y_0, Y_1, \dots, Y_{L-1} 组成)



**IV=初始值； CV=连接变量； Y_i =第*i*个输入分组； f =压缩函数
L=输入分组数； n=hash码的长度； b=输入分组的长度**

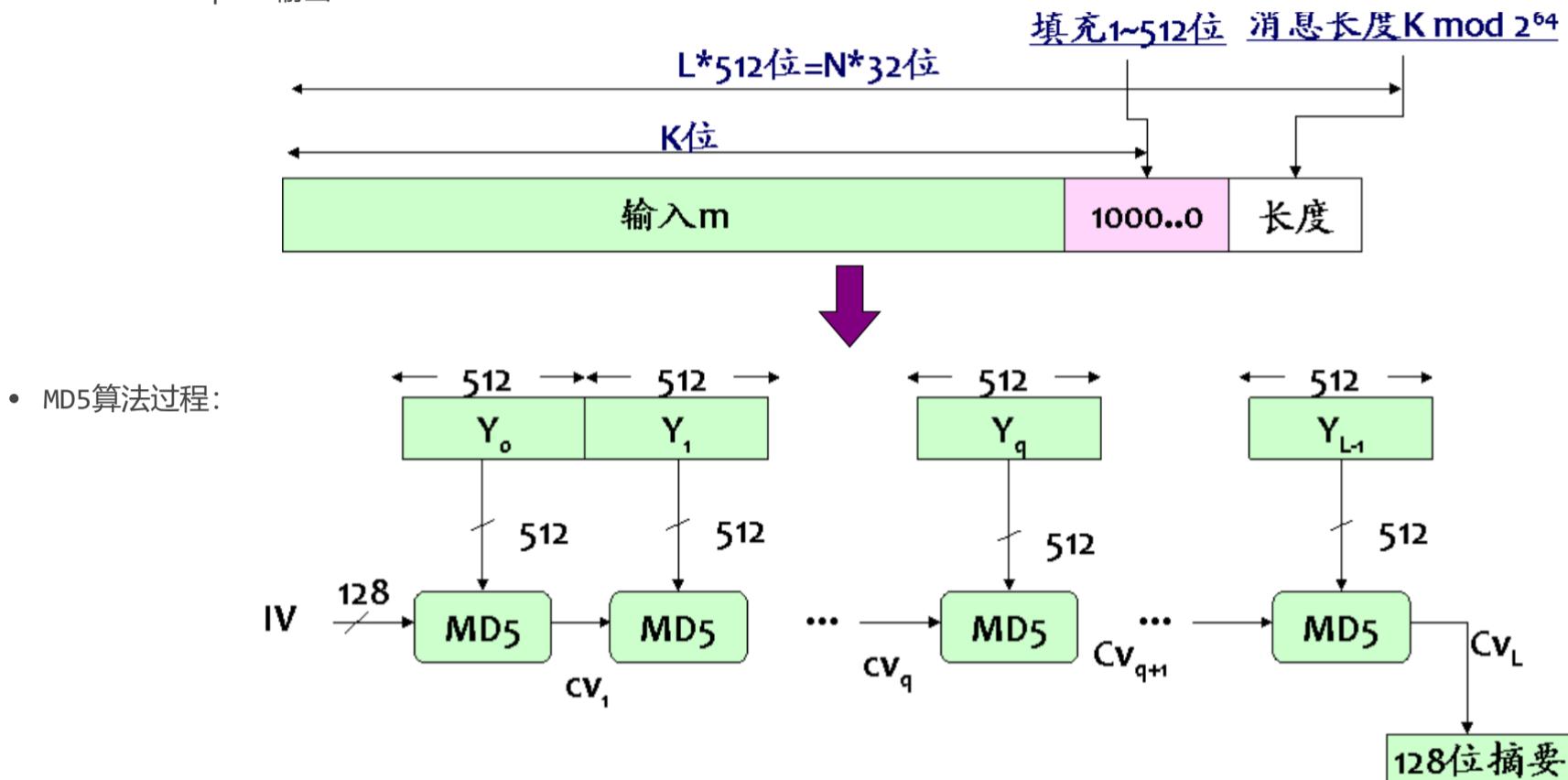
- Hash函数中重复使用了压缩函数f：
 - 它的输入是前一步中得出的n位结果（即连接变量）和一个b位分组，输出位一个n位分组
 - 连接变量的初始值在算法开始的时候指定，其终值即为hash值。通常 $b > n$ ，因此称为压缩函数
- 设计安全hash函数可以归纳为设计具有抗碰撞能力的压缩函数问题，并且该压缩函数的输入是定长的

主要的hash算法概述

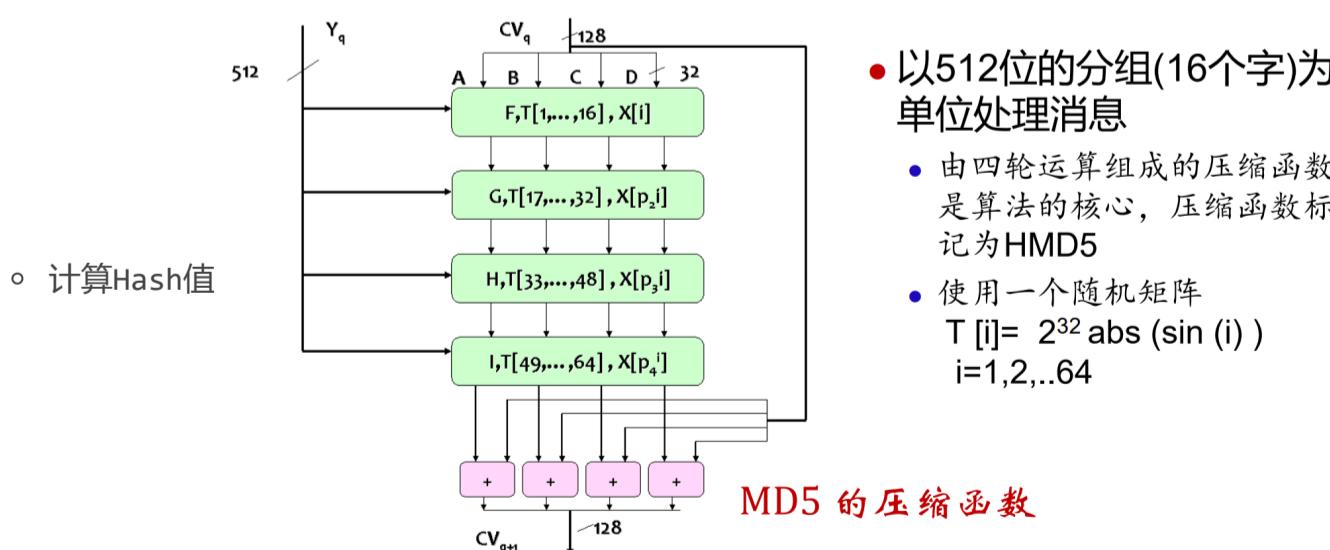
- MD族：
 - MD = Message Digest (消息摘要)
 - MD2、MD4和MD5都产生一个128位的消息摘要
- SHA族：
 - SHA = Secure Hash Algorithm
 - SHA-0：正式地称作SHA系列，发行后不久即被指出存在弱点
 - SHA-1：1994年发布的，与MD4和MD5散列算法非常相似，被认为是MD4和MD5的后继者。
 - SHA-2：实际上分为SHA-224、SHA-256、SHA-384和SHA-512算法。
 - SHA-3：方案正在征集中。
- HAVAL：加密哈希算法
- Gost：是一套俄国标准
- RIPEMD-128/160/320：RIPEMD由欧洲财团开发和设计的MD算法
- 以上算法中，仅有SHA-2和RIPEMD-160未被攻破

Hash算法：MD5

- MD5的输入是任意长度的消息，对输入按照512位的分组为单位进行处理，算法的输出是128位的消息摘要
 - 使用小端结构（低端结构）
- MD5算法步骤：
 - Step1：增加填充位
 - Step2：填充长度
 - Step3：初始化MD缓存
 - Step4：以512位的分组处理信息
 - Step5：输出

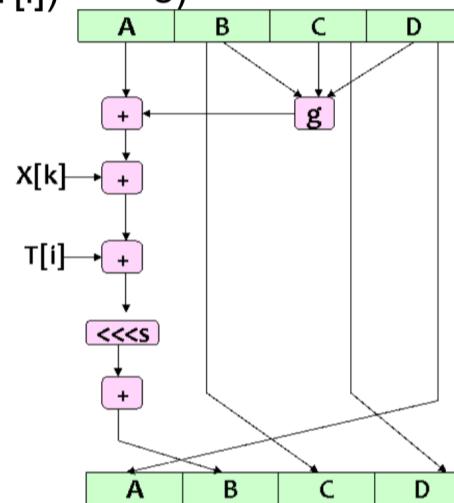


- MD5算法过程：
 - Step1：增加填充位
 - 填充bit：第1位为1，其后所有位都为0
 - 填充bit后，使得消息长度与 $448 \ (512-64) \bmod 512$ 同余
 - Step2：填充长度
 - 用64位表示填充前的报文长度，附加填充比特的后面，如果长度大于 2^{64} ，则对 2^{64} 取模
 - 以低端结构方式表示被填充前的消息长度
 - 完成填充后，消息的总长度是512的整数倍
 - Step3：初始化MD缓存
 - Hash函数的中间结果和最终结果都保存于128位的缓冲区中，缓冲区用4个32位的寄存器(A, B, C, D)表示
 - A, B, C, D寄存器以低端格式存储，初始值为 $0 \sim F \ F \sim 0$
 - Step4：



- 压缩函数由四轮运算构成，四轮运算结构相同，但各轮使用不同的基本逻辑函数，分别称之为F/G/H/I
 - 每轮的输入是当前要处理的512位的分组(Y_q)和128位缓冲区ABCD的内容
 - MD5的压缩函数
 - 表T有64个元素，每轮使用表T[1...64]中的16个元素，并更新缓冲区
 - 表T是通过正弦函数构造的，T的第i个元素记为T[i]
 - T的每个元素都可以用32位表示，堆积化的32位输入数据，消除了输入数据的规律性
 - $T[i] = 2^{32} \text{abs}(\sin(i)) \quad i=1,2,\dots,64$
 - 第四轮的输出与第一轮CV_q的输入相加得到CV_{q+1}。
 - 加法是指缓冲区中的4个字与CV_q中对应的4个字分别模 2^{32} 相加。
 - MD5中每轮对缓冲区ABCD进行16步迭代，每步迭代为：

$$b \leftarrow a + ((a + g(b,c,d) + X[k] + T[i]) \ll s)$$
 - a,b,c,d：缓冲区的四个字，它按照一定的次序随迭代步变化
 - g：基本逻辑函数F/G/H/I之一
 - <<< s：32位的变量循环左移s位
 - X[k] = M[q*16+k]：消息第q个512位分组的第k个32位字
 - T[i]：矩阵T中的第i个32位字
 - +：模 2^{32} 加法
 - MD5 每轮处理512位分组的过程
 - <<< s：32位的变量循环左移s位
- 续上图
 - b $\leftarrow a + ((a + g(b,c,d) + X[k] + T[i]) \ll s)$
 - X[k]：消息第q个512位分组的第k个32位字
 - T[i]：矩阵T中的第i个32位字



- Step5：输出
 - 所有的L个512位的分组处理完之后，第L个分组的输出即是128位的消息摘要
- MD5算法中，hash函数的每一位都是输入的每一位的函数；基本逻辑函数的复杂迭代使得输出对输入的依赖性非常小

Hash算法：各类算法比较

MD5/SHA-1/RIPEMD-160比较			
	MD5	SHA-1	RIPEMD-160
摘要长度	128 bits	160 bits	160 bits
基本处理单元	512 bits	512 bits	512 bits
步数	64(4 轮，每轮 16 步)	80(4 轮，每轮 20 步)	160(5 轮，每轮 16 步)
最大消息长度	∞	$2^{64}-1$ 位	$2^{64}-1$ 位
基本逻辑函数	4	4	5
使用的加法常量	64	4	9
低端位/高端位结构	低位在前	高位在前	低位在前

● 抗穷举攻击的能力

- 三个算法都不易受到弱碰撞性的攻击
- MD5由于消息摘要短，易于收到强碰撞性的攻击
- SHA-1和RIPEMD-160在将来，对强碰撞性的攻击还是安全的

● 抗密码分析的能力

- 对MD5的密码分析取得了很大进展
- 对SHA-1的密码分析比MD5要困难
- 对RIPEMD-160的密码分析比SHA-1又要困难

● 速度

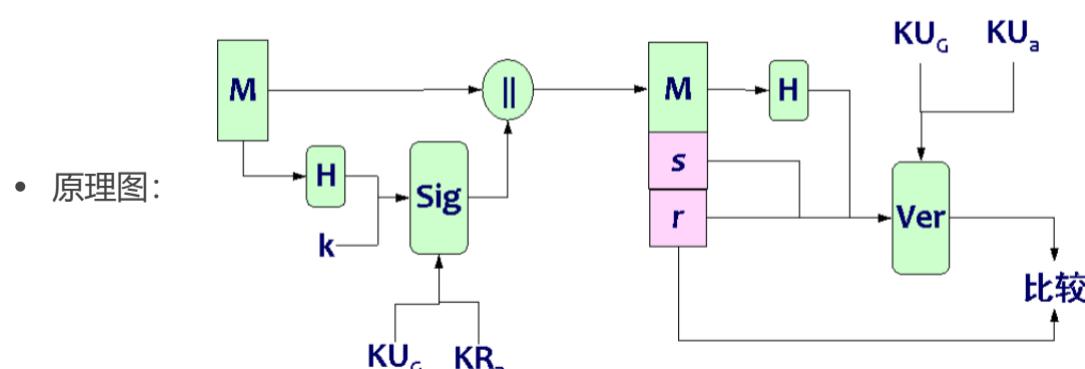
- 三个算法都基于模 2^{32} 加法和简单的位逻辑运算，所以它们在32位机上执行速度较快
- 由于MD5相对简单，迭代次数少，所以运行速度最快

● 低端位/高端位结构

- MD5和RIPEMD-160采用低位在前结构，SHA-1采用高位在前结构

数字签名算法DSS

- 数字签名：消息认证可以保证通信双方不受第三方的攻击，但是它不能处理通信双方自身发生的攻击
- 数字签名的特征：
 - 它必须能够验证签名者、签名日期和时间
 - 它必须能认证被签的消息内容
 - 签名应能由第三方仲裁，以解决争执
- 数字签名可以分成两类：直接数字签名和仲裁数字签名
 - 直接数字签名只涉及通信双方：
 - 假定接收方已知发送方的公钥，则发送方可以通过用自己的私钥对整个消息或者消息的hash码加密来产生数字签名
 - 用接收方的公钥(公钥密码)和共享的密钥(对称密码)对整个消息和签名进行加密，则可以获得保密性
 - 直接数字签名的弱点在于方法的有效性依赖于发送方私钥安全性
 - 仲裁数字签名：
 - 从发送方X到接收方Y的每条已签名的消息都先发给仲裁者A， A对消息及其签名进行检查以验证消息源及其内容，然后给消息加上日期，并发给Y，同时指明该消息已通过仲裁者的检验
 - 通过A的加入解决了直接数字签名的问题
- DSS使用SHA-1算法给出的数字签名方法叫做数字签名算法DSA
- DSS只提供数字签名功能，不能用于加密或者密钥分配



- 用hash函数产生的hash码和随机数k，以及发送方的私钥 KR_a 和 KU_G 作为数字签名函数sig的输入
- 签名函数sig的输出分为s和r两部分，将m, s, r, 拼接为新的消息
- 接收方对接收到的消息产生hash码，此hash码和签名以其作为验证函数Ver的输入，验证函数依赖于全局公钥 KU_G 和 KU_a ，若验证函数Ver的输出等于签名中的r，则签名有效
-

- 全局公钥 (p, q, g)
 - p : 为 L 位长的素数。其中， L 为 512~1024 之间且是 64 倍数的数。
 - q : 是 160 位长的素数，且为 $p-1$ 的因子。
 - g : $g = h^{(p-1)/q} \bmod p$
其中， h 是满足 $1 < h < p-1$ 且 $h^{(p-1)/q} \bmod p$ 大于 1 的整数。

- 用户私钥 x : x 为在 $0 < x < q-1$ 内的随机数

- 用户公钥 y : $y = g^x \bmod p$

- 用户每个消息用的秘密随机数 k , $0 < k < q$

参数 p 、 q 、 g 是公开的； x 为私钥， y 为公钥；
 对于每一次签名都应该产生一次 k ； x 和 k 用于数字签名，必须保密；

签名过程与验证过程

签名过程

用户随机选取 k , 计算：

- $r = (g^k \bmod p) \bmod q$
- $s = [k^{-1}(H(M) + xr)] \bmod q$

(r, s) 即为消息 M 的数字签名

验证过程

接收者收到 M, r, s 后，首先验证 $0 < r < q$, $0 < s < q$, 如通过则计算：

- $w = (s)^{-1} \bmod q$
- $u_1 = [M^w] \bmod q$
- $u_2 = [r^w] \bmod q$
- $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$

如果 $v=r$, 则确认签名正确

电子身份认证

- 身份认证是指在主客体交互行为过程中确认行为参与（主体或客体，通常称为用户）身份的解决方法
- 电子身份认证是在计算机及网络中确认操作者身份的解决方法

网站身份认证技术

- 最简单的网站用户认证：HTTP的Basic认证
- HTTP的Basic认证：
 - 用户身份凭证：账号+静态口令 (U, P)
 - 由于HTTP协议是面向一次连接的无状态网络协议，因此需要在每次发出HTTP请求时，把用户身份凭证的明文发送到服务器端，服务器与存储在服务器端的用户凭证进行比较
 - 发送时，对账号口令进行Base64编码，服务器端接收后进行Base64解码
- HTTP的Basic认证：改进1
 - 解决账号口令明文传输可能被监听盗取的问题
 - 使用加密技术
 - 使用消息认证技术
 - 解决账号口令长期本地保存以及服务器端每次都进行账号口令验证的问题
 - 使用表单验证+session的机制
- HTTP的Basic认证：改进2
 - 将口令 P 作为密钥 E_k , 每次发送 $U || E_k[U]$
- HTTP的Basic认证：改进3
 - 使用MAC认证
 - 采用
 - 挑战/响应(Challenge/Response)机制，需要进行两次HTTP请求：
 - 第一次请求：服务器向客户端返回一个随机生成的挑战码 M
 - 第二次请求：服务器向客户端返回一个挑战码 M ，服务器端进行验证

基于表单的身份认证

- Web的Session机制:
 - 一个Session包括特定的客户端、特定的服务器端和特定的操作时间段
- Session的工作原理:
 - 当某个Session首次启用时，服务器会产生一个唯一的标识符发送到客户端
 - 唯一标识符通常是一串随机字符串
 - 在客户端浏览器上，唯一标识符通常用Cookie技术存储
 - 在Session存活期间，客户端每次向服务器发送的HTTP请求都会包含上述唯一标识符，使得服务器能够把前后多次请求关联起来
 - 当Session结束时，服务器端和客户端都应当销毁上述唯一标识符
- 基于表单的Web身份认证过程通常包括三个步骤:
 1. 客户端向服务器发送请求，服务器返回包含表单的页面
 2. 用户按要求填写表单的内容完成后，客户端把表单内容发送到服务器；服务器获取表单中的内容后，进行验证，验证通过则启动Session并返回给客户端（这里账号和口令通过明文传输）
 3. 客户端后续的请求包含Session的唯一标识符，服务端验证唯一标识符的合法性
- 缺点是安全性低，账号和口令明文在网络上传输
- 基于表单的身份认证：改进1
 - 引入Challenge/Response机制，避免口令明文通过网路传输，并且能避免重放攻击：
 - 第一步，服务器返回表单页面时，包含一个Challenge（通常为随机字符串+时间戳，记为M）
 - 第二步，用户完成表单填写提交请求之前，以口令为加密密钥（k=口令）计算Ek[M||U]，并将账号U和Ek[M||U]发送到服务器，服务器收到后，用存储的口令k'计算Ek'[M||U]，与收到的Ek[M||U]比较，完成用户的身份验证
- 基于表单的身份认证：改进2
 - 使用传输层SSL协议传输HTTP请求
- 网站身份增强认证的其它方式:
 - 手机短信口令
 - 动态口令
 - USB KEY
 - 数字证书

Lettuce5 WLAN安全

无线网络概述

- 无线电通信技术是在没有物理连接的情况下多个设备之间能够互相通信的技术

无线网络	网络类型	应用范围/典型技术	传输距离
	无线个人域网WPAN (Wireless Personal Area Network)	点对点短距离链接，个人办公家庭环境 IEEE 802.15x/Bluetooth/ZigBee等	10m左右
	无线局域网WLAN (Wireless Local Area Network)	点对多点无线连接，支持AP间切换 IEEE 802.11 (a.b.g.n) 等	100m
	无线城域网WMAN (Wireless Metropolitan Area Network)	点对多点无线连接，支持基站间漫游与切换 IEEE 802.16等	1-50KM
	无线广域网WWAN (Wireless Wide Area Network)	全球通讯，通信卫星 IEEE 802.20等	1-15KM

- WLAN的安全威胁:

- 无线窃听
- 假冒攻击
- 信息篡改
- 重放、重路由、错误路由、删除消息
- 网络泛洪

无线局域网加密认证技术

- 无线局域网加密认证技术
 - 无加密认证
 - WEP
 - WPA
 - 无加密认证:
 - 无线接入点AP：网络的中心节点
 - STA站点：连接到无线网络中的终端
 - SSID：便于用户识别，为每个AP配置的标志名，通常情况下，AP会向外广播SSID，可以通过disable Broadcast提高安全性
 - 有线等效保密协议WEP：
 - 有线等效保密协议WEP (Wired Equivalent Privacy) 目的是为无线局域网提供与有线网络相同级别的安全保护，协议标准为IEEE 802.11b
 - 使用对称加密算法
-
- WEP的安全措施**

 - 认证:
 - 开放系统认证
 - 默认认证方式
 - 对请求认证的任何人提供明文认证
 - 共享密钥认证

• 保密性: RC4流密码加密，密钥长度40bit/104bit

• 完整性: 循环冗余校验CRC32

• 密钥管理:
 - 各个设备与接入点共享一组默认密钥，可能被泄露
 - 每个设备与其他设备建立密钥对关系，密钥人工分发困难

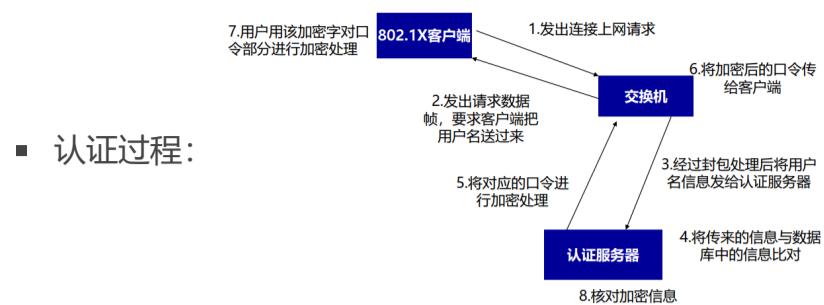
WEP 加密过程

 - 计算校验和
 - 对明文P进行完整性CRC-32计算，得到校验和ICV
 - 串接明文P和校验和ICV，用于下一步加密过程的输入
 - 加密
 - 将24位初始化向量IV和40位共享密钥K0连接得到64位的数据，输入到虚拟随机数产生器中，产生一次性密钥KE
 - 将KE和上一步的计算结果进行按位异或运算，得到密文Y
 - 传输
 - 将24位初始化向量IV和密文Y串联起来，在无线链路上传输

WEP的解密过程

 - 计算一次性密钥KE
 - 将收到的信息中初始化向量IV和共享密钥K0串接，输入到虚拟随机数产生器中，产生一次性密钥KE
 - 解密
 - 将收到信息中的密文P和一次性密钥KE进行按位异或运算，得到明文P和ICV串接信息
 - 认证
 - 将计算得到的明文P进行完整性CRC-32计算，得到校验和ICV'
 - 比较ICV和ICV'，相等接收，不相等丢弃
- ## WEP的安全性分析
- 同一个SSID中，所有STA和AP共享同一个密钥，容易泄露
 - 生成RC4密码流的初始化向量IV明文发送
 - 24 bits的初始化向量IV过短，容易重复
 - 完整性校验算法CRC-32是非加密的线性运算，主要用于检测消息中的随机错误，不是安全的杂凑函数，无法实现消息认证
 - RC4是一个序列密码加密算法，发送者用一个密钥序列和明文异或产生密文，接收者用相同的密钥序列与密文异或恢复出明文，容易被破译
 - WEP协议中不含序列号，无法确定帧顺序，无法提供抵抗重放攻击
 - WPA1：因为WEP的严重安全漏洞，2002年Wi-Fi联盟制定了WPA1，这是一个过度性的中间标准，其核心就是IEEE 802.1x和TKIP
 - 临时密钥完整性协议TKIP：包在已有的WEP密码外围的一层“外壳”
 - TKIP使用WEP同样的加密引擎和RC4算法，但是TKIP中密钥使用的密钥长度是128位
 - 动态变化每个数据包所使用的密钥
 - 利用TKIP传送的每一个数据包都具有独有的48位序列号
 - IEEE 802.1x：
 - IEEE 802.1x是针对以太网而提出的，基于端口的网络访问控制利用物理层特性对连接到无线端口的设备进行身份认证

- IEEE 802.1x基于客户/服务器模式，可以在无线终端与AP建立连接之前，对用户身份的合法性进行认证



■ 认证过程：

• WEP-WPA1-WPA2比较：

- WPA使公共场所和学术环境安全地部署无线网络成为可能
 - WEP使用一个静态的密钥来加密所有的通信，这意味着为了更新密钥，技术人员必须亲自访问每台机器，而这在学术环境和公共场所是不可能的
 - WPA采用有效的密钥分发机制，不断转换密钥

加密技术	全称	加密算法	协议
WEP	Wired Equivalent Privacy (有线对等保密)	RC4	IEEE 802.11b
WPA	Wi-Fi Protected Access (WiFi安全存取)	RC4, 使用TKIP	IEEE 802.11i draft 3
WPA2	Wi-Fi Protected Access 2 (WiFi安全存取 第二版)	支持AES, 使用CCMP 需要新硬件支持	IEEE 802.11i

Lettuce 6 CIA & VPN

安全目标

• 安全目标、服务、机制的关系：



安全目标：CIA

- Confidentiality: 保密性、机密性
- Integrity: 完整性
- Availability: 可用性

OSI 安全框架

OSI安全框架主要关注安全服务、安全机制和安全攻击

- 安全服务：一种由系统提供的对系统自愿进行特殊的处理或通信服务，安全服务通过安全机制来实现安全策略
- 安全机制：用来保护系统免受监听、阻止安全攻击及恢复系统的机制
- 安全攻击：主动攻击、被动攻击

安全服务

- X.800提供了以下安全服务：
 - Authentication: 认证服务
 - Confidentiality: 保密服务
 - Integrity: 数据完整性保护
 - Access Control: 访问控制服务
 - Non-repudiation: 抗抵赖服务
 - Availability: 可用性服务

Authentication 认证服务

- 认证服务Authentication与保证通信的真实性有关
- 在单向通信时：认证服务向接受方保证发送方的真实性
- 在双方通信时：

- 在连接的初始化阶段，认证服务保证双方的真实性
- 认证服务还需要保证该连接不受第三方非法干扰
- 两个特殊的认证服务：
 - 对等实体认证：
 - 参与通信的实体的身份是真实的
 - 一个实体不能试图进行伪装或者对以前连接进行非授权的重放
 - 面向连接的应用
 - 数据源认证：
 - 对数据的来源提供确认，但是对数据的复制和修改不提供保护
 - 保证接收到的信息的确来自它所宣称的来源
 - 面向无连接的应用

Confidentiality 保密服务

- 保密服务Confidentiality是防止传输数据遭到被动攻击
- 分为连接保密服务和无连接保密服务
- 保密力度：流(stream)、消息(message)、选择字段(field)
- 保密服务防止攻击者观察到消息的源、目的、频率、长度或者其它流量特征

Integrity & Access Control

- 数据完整性服务Integrity：可对消息流、单条消息或消息的选定部分进行保护
- 访问控制服务Access Control：是指限制实体的访问权限，通常是经过认证的合法的实体才可以访问

Non-repudiation & Availability

- 抗抵赖服务Non-repudiation：防止发送方或者接收方否认传输或者接收过某条消息
- 可用性服务Availability：根据系统的性能说明，能够按照授权的系统实体的要求存取或使用系统或系统资源的性质

安全机制

分类

安全机制分为两类：

- 普通安全机制：不属于任何协议层或者安全服务的安全机制
- 特定安全机制：在特定的协议层实现的安全机制

普通的安全机制

- 可信功能(trusted functions)
 - 根据某些标准被认为是正确的
- 安全标签(security Labels)
 - 资源的标志，指明该资源的安全属性
- 事件检测 (Event Detection)
 - 检测与安全相关的事件
- 审计跟踪 (security audit Trail)
 - 收集用于安全审计的数据，对系统记录和行为的独立回顾和检查
- 安全恢复 (security recovery)
 - 处理来自安全机制的请求，如事件处理、管理功能和采取恢复行为

特定的安全机制

● 八种特定的安全机制包括

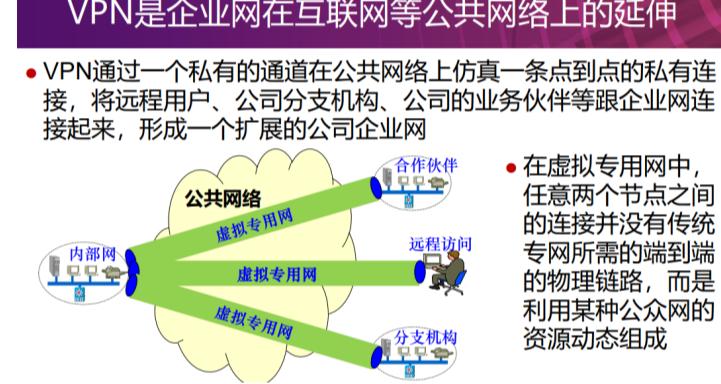
- ① 加密机制
- ② 数字签名机制
- ③ 访问控制机制
- ④ 数据完整性机制
- ⑤ 认证机制
- ⑥ 业务流填充机制
- ⑦ 路由控制机制
- ⑧ 公证机制

安全性攻击

- 主动攻击：试图改变系统资源或者影响系统运行
- 被动攻击：试图了解或者利用系统的信息但不影响系统资源
 - 窃听
 - 流量分析

虚拟专用网VPN

- 需求背景：TCP/IP协议本身存在许多固有的安全缺陷，架设物理专用网难度大
- VPN的定义：Virtual Private Network，虚拟专用网
-



- VPN提供的安全功能
 - 数据机密性保护
 - 数据完整性保护
 - 数据源身份验证
 - 重放攻击保护
- VPN的解决方案
 - 基于数据链路层的VPN解决方案:L2TP/PPTP/L2F
 - 基于网络层的VPN解决方案:IPsec/IKE
 - 基于传输层的VPN解决方案:SSL
- 基于数据链路层的VPN解决方案：
 - 由于数据链路层的VPN技术在认证、数据完整性以及密钥管理等方面的应用不足，现在已经很少应用
- 基于传输层的VPN解决方案:
 - 基于传输层SSL协议的VPN解决方案，零客户端是其最大优势
 - 在实际应用中，SSL VPN和IPsec VPN两种方案往往结合实行
 - 像视频会议这样的非B/S (Browser-Server) 结构的业务是无法通过SSL VPN建立和开展的

Lettuce 7 IPsec & IKE

■ 网络层安全协议：IPsec

IP级安全性

- IPsec保证了IP级的安全性，包括：
 - 认证
 - 保密
 - 密钥管理
- 现有IP协议的安全特性：
 - 无连接，不保证顺序到达；存在着重复包、丢失包；设备简单、无状态
 - 所提供的安全服务：无认证、完整性、保密性，基于IP地址的访问控制，不完备
- IPsec的原理在于可以在IP层加密和/或认证所有流量
- IPsec可以在主机、防火墙或路由器上实施

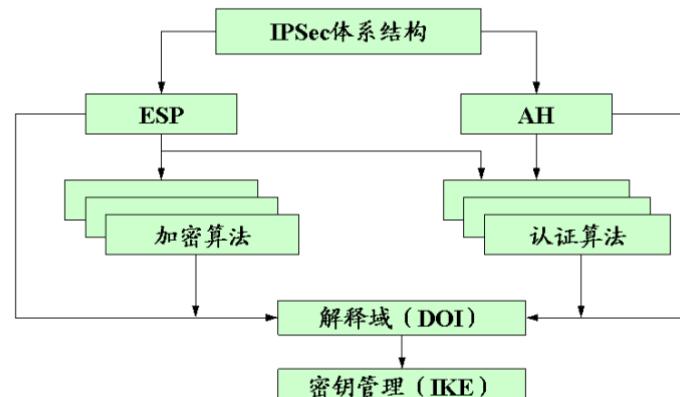
IPsec：文档

- IPsec文档：
 - 重要的有1998年发布的RFC 2401/2402/2406/2408

◦ 整个IPsec文档被分为7个部分：

- 体系结构
- 认证头AH
- 封装安全载荷ESP
- 加密算法
- 认证算法
- 密钥管理
- 解释域

◦



IPsec提供的服务

- 它们提供的安全服务包括：

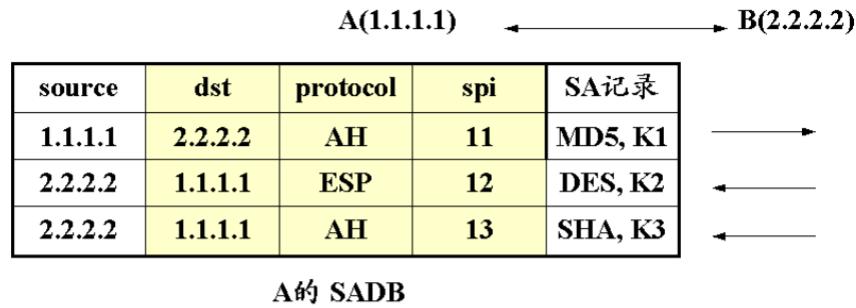
- 访问控制
- 无连接完整性
- 数据源发认证
- 拒绝重放数据包
- 保密性（加密）
- 有限的信息流保密性

	AH	ESP(只加密)	ESP(加密并认证)
访问控制	✓	✓	✓
无连接的完整性	✓		✓
数据源发认证	✓		✓
检测重放攻击	✓	✓	✓
机密性		✓	✓
有限的通信流保密		✓	✓

安全关联

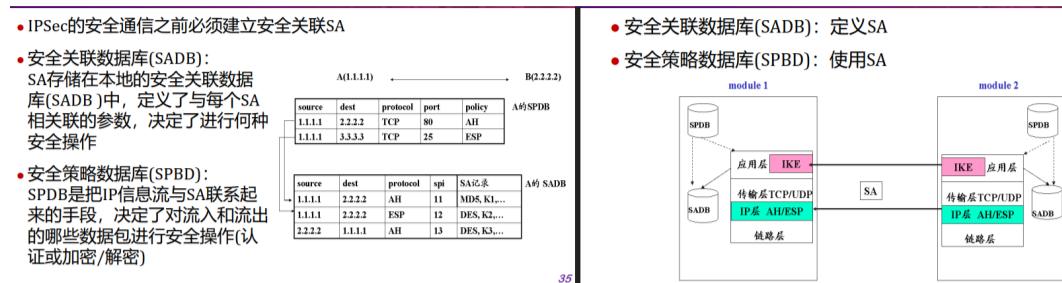
- IP认证和保密机制中的一个核心概念是安全关联SA
- 安全关联SA是IPsec通信双方之间对某些要素的一种协商，一组安全信息参数集合，包括：协议、操作模式、密码算法、认证算法、密钥、密钥生存期等
- 关联是发送方和接收方之间的单向关系，该关联为双方的通信提供了安全服务
 - 如果需要双方安全交换，则建立两个安全关联
 - 安全服务可由AH或者ESP提供，但不能两者都提供
- 一个安全关联SA由3个参数唯一确定：

- 安全参数索引SPI：一个和SA相关的位串，仅在本地有意义，由AH和ESP携带，使得接收方能够选择合适的SA处理接收包
- IP目的地址IPDA：只允许使用单一地址
- 安全协议标识：标识该关联是一个AH安全关联或ESP安全关联
- 安全关联数据库SADB：包含了 (SPI, IPDA, AH/ESP) 三元组索引
 - 在任何IPSec实现中，都有一个安全关联数据库SADB，它定义了与每个SA相关联的参数



- SA的参数：
 - 序列号计数器：一个32位整数，刚开始通常为0，用于生成AH或ESP头中的序列号域；每次用SA保护一个包时增1；在溢出之后，SA会重新进行协商
 - 序列号溢出标志：表明序列号是否溢出，序列号计数器的溢出时，该值为1时，产生审查事件并阻止该SA继续下发数据包
 - 反重放窗口：决定输入AH或ESP报文是否是replay的32位计数器
 - AH信息组：(AH必须实现) 认证算法，密钥，密钥生存期和AH的相关参数
 - ESP信息组：(ESP必须实现) 加密和认证算法，密钥，初始值，密钥生存期和ESP的相关参数
 - SA的生存期：(可选) 一个时间间隔或字节计数 生存期结束时，SA必须终止，或用一个新的SA替换
 - IPSec协议模式：隧道模式或者传输模式
 - Path MTU：任何遵从的最大传送单位路径和迟滞变量
- SADB的工作过程：
 - 在IP数据包中，安全管理SA由IPv4或IPv6报头中的目的地址唯一标识，SPI被封装在AH或者ESP扩展头中
 - 任何IPSec实现中都必须实现安全关联数据库SADB；对于收到的数据包，解析出三元组【SPI、目的地址、AH/ESP】，并据此查找SADB
 - 如果查找到一个匹配的SA条目，则将该SA的参数与AH或ESP头中相关域进行比较：参数相一致，就处理该数据包；参数不一致，就丢弃该数据包
 - 如果没有查找到匹配的SA条目：如果数据包是输入包，丢弃；如数据包是输出包，将创建一个新的SA，并将其存入SADB中
- SA选择子：每个SPDB入口由一个IP集合和上层协议定义，称为选择子(SA selector)
 - IP流量与特定SA相关是通过安全策略数据库SPDB定义的
 - SPDB至少应该包括定义IP流量子集的入口、指向该流量SA的指针
 - IPSec对需要IPSec保护的流量和不需要IPSec保护的流量进行了大粒度区分
 - SA选择子用于过滤输出流量，并将它们映射到某个特定的SA
 - 每个IP包的输出过程如下：
 - 在SPDB中比较相应域的值，寻找匹配的入口，可能是零或多个
 - 如果存在SA，则选定SA和其关联的SPI执行所需的IPSec处理 (AH或ESP)
 - 目的IP地址、源IP地址
 - 用户标识
 - 数据敏感性级别
 - 传输层协议
- SPDB入口由下列选择子确定：
 - IPSec协议
 - 源端口和目的端口
 - IPv6报类
 - IPv6报流标签
 - IPv6报服务类型
- 安全策略数据库SPDB：
 - 在IPSec中，安全策略通过SPDB来定义、标识、管理和维护

- 对于数据包的操作策略包括：
 - Discard**: 不让这个包进入或外发
 - bypass IPsec**: 不对进入或外发的数据包进行安全服务
 - apply IPsec**: 对外发的数据包提供安全服务，同时认为接收的数据包已经进行过安全服务
- SADB与SPDB：

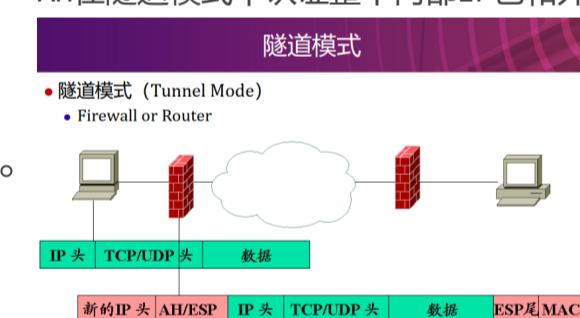


传输模式/隧道模式

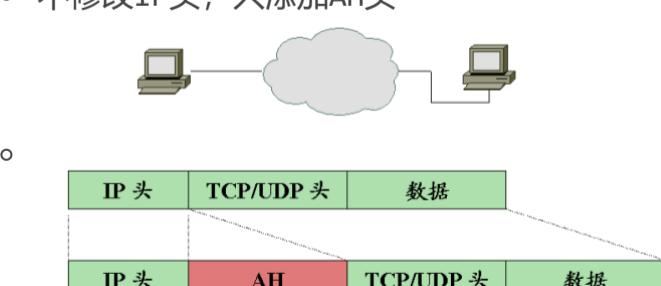
- AH和ESP均支持两种模式：传输模式和隧道模式
- 传输模式：主要为上层协议提供保护，同时增加了IP包载荷的保护
 - 典型的传输模式用于两台主机之间进行的端到端通信
 - 传输模式的ESP加密和认证（可选）IP载荷，不包括报头
 - 传输模式的AH认证IP载荷和报头的选中部分



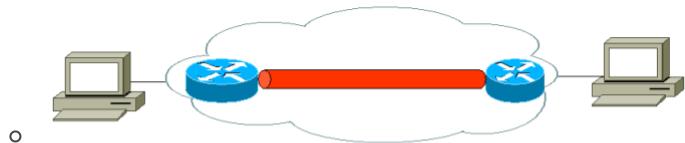
- 隧道模式：对整个IP包提供保护
 - 当IP包加上AH/ESP域后，整个数据包和安全域被当作一个新的IP载荷，并拥有一个新的外部IP报头
 - 新的IP数据包利用隧道在网络中传输，途中的路由器不能检查内部IP报头
 - ESP在隧道模式中加密和认证（可选）整个内部IP包，包括内部IP报头
 - AH在隧道模式中认证整个内部IP包和外部IP报头的选中部分



- 在传输模式和隧道模式中使用AH
 - 途径1：直接在服务器和客户工作站之间提供传输模式的认证
 - 途径2：在服务器不支持认证的情况下，远程工作站向防火墙证明自己身份，以便访问整个内部网络的时候，使用隧道模式SA
- 在传输模式中使用AH：
 - 通常以端到端方式实现（在主机上实现）
 - 不修改IP头，只添加AH头



- 在隧道模式中使用AH：
 - 通常在防火墙或路由器上实现
 - 把整个IP包作为数据，增加一个新的IP头、AH头



IP 头 TCP/UDP 头 数据

新的IP头 AH IP 头 TCP/UDP 头 数据

- 在传输模式和隧道模式中使用ESP：

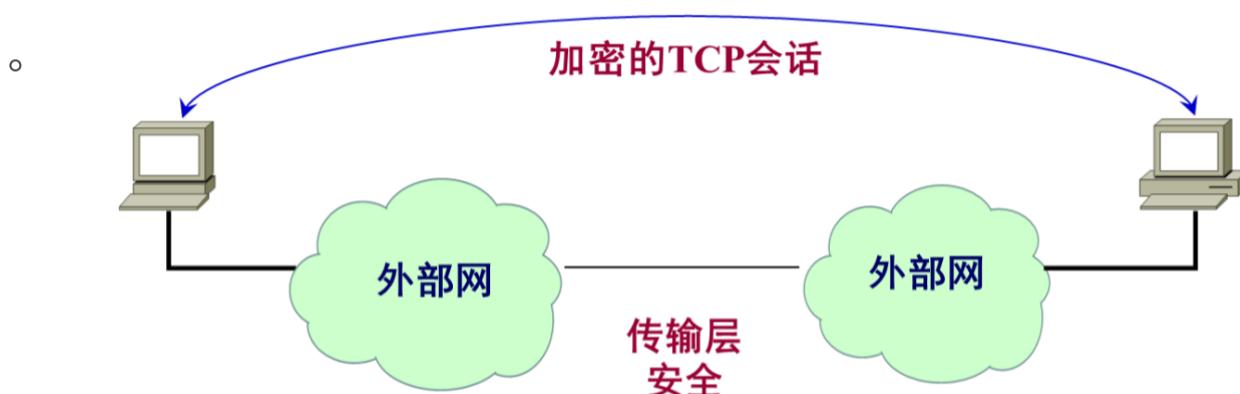
- 途径1：加密和认证（可选）直接由两个主机提供，采用传输模式
- 途径2：隧道模式的ESP用于加密整个IP包，可以创建VPN

- 在传输模式中使用ESP：

- 在源端，包括ESP尾和整个传输层分段的数据块被加密，块中的明文被密文替代，形成要传输的IP包；如果选择了认证，则加上认证
- 将包送往目的地：中间路由器需要检查和处理IP报头以及任何附加的IP扩展头，但不需要检查密文
- 目的节点对IP报头和任何附加的IP扩展报头进行处理后，利用ESP报头中的SPI解密包的剩余部分，恢复传输层分段数据

IP 头 TCP/UDP 头 数据

IP 头 ESP 头 TCP/UDP 头 数据 ESP 尾 MAC

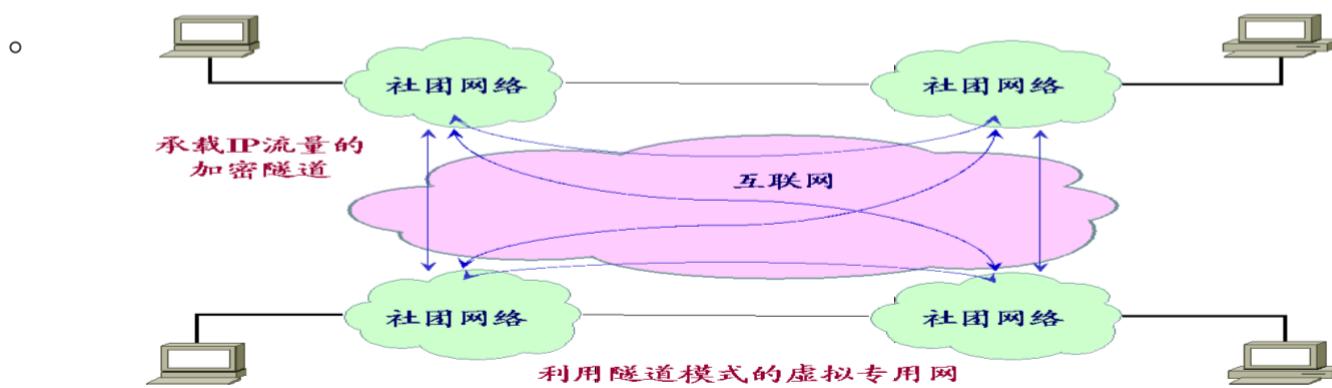


- 在隧道模式中使用ESP：

- 隧道模式的ESP用于加密整个IP包，可用于建设虚拟专用网
- 将ESP头作为包的前缀，并在包后附加ESP尾，然后对其进行加密

IP 头 TCP/UDP 头 数据

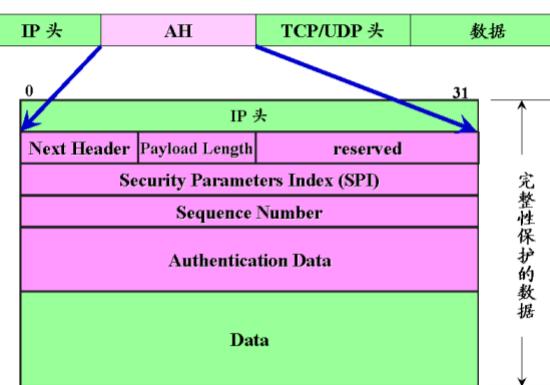
新的IP头 ESP 头 IP 头 TCP/UDP 头 数据 ESP 尾 MAC



IPsec：认证头AH

- 认证头AH支持数据完整性和IP包的认证
- 数据完整性确保在包的传输过程中内容不可更改
- 认证基于消息认证码MAC，双方必须共享一个公钥
- 认证确保末端系统或者网络设备对用户或者应用程序进行认证，并提供相应的流量过滤功能，同时还能防止地址欺诈攻击和重放攻击
-

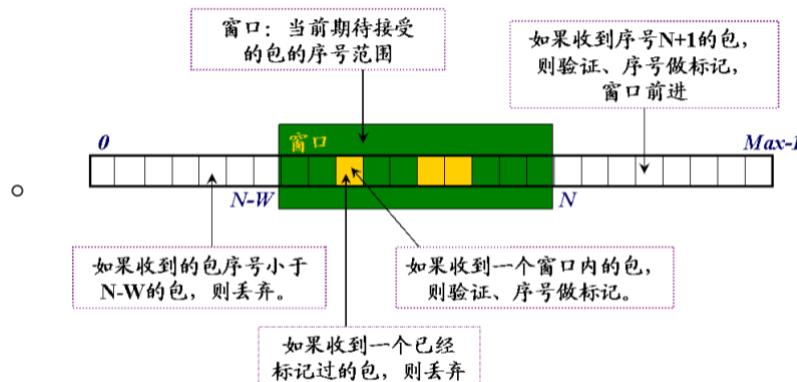
认证头的组成



- Next header(邻接头)
 - 8 bits, 标识数据载荷中的封装方式或协议
 - 例如: TCP/UDP/ICMP、AH、ESP
 - Payload length(有效载荷长度)
 - 8 bits, 以32位字为单位的认证数据字段长度
 - Reserved(保留字)
 - 16 bits, 保留以供将来使用
 - 发送时必需设置为全零
 - 这个值包含在认证数据计算中, 否则被接收方忽略
 - Security Parameters Index (安全参数索引)
 - SPI为数据识别安全联合的32位伪随机值, SPI=0被保留, 表明“没有安全关联存在”
 - Sequence Number(序列号)
 - 32bits, 单调递增计数器, 用于防范重放类型攻击
 - Authentication Data(认证数据AD)
 - 变长域, 必须是32 bits的整数倍
 - 包含完整性校验值ICV或者包的MAC

• 反重放攻击:

- 重放攻击是指攻击者在得到一个经过认证的包后, 又将其传送到目的站点的行为
- 重复接收经过认证的IP包可能会以某种方式中断服务或者产生不可预料的后果, 序列号域就可以防止重放攻击
- 当建立了一个新的SA时, 发送方将序列号初值设为0, 每次在SA上发送一个包, 则计数器加1并将值放入序列号域, 则使用的第一值就是1
- 发送方不允许循环计数, 否则, 同一个序列号就可以产生多个合法的包; 如果该序列号到达 $2^{32}-1$, 则SA必须中止, 用新的密钥协商声称新的SA
- 由于IP协议是无连接的, IPsec认证文档中声明, 接收方应该实现一个大小为w的窗口 (w 的默认值为64), 如下图:

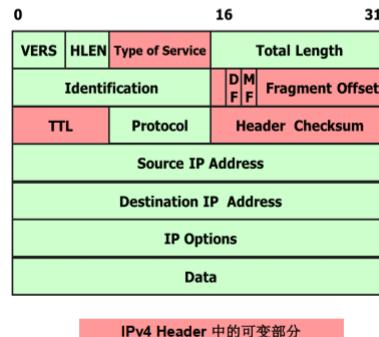


• 完整性校验值:

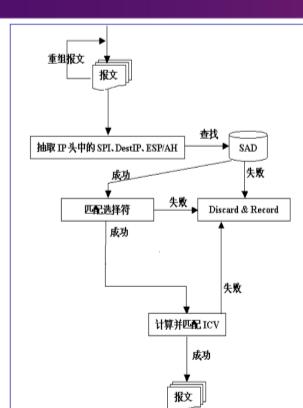
- 认证数据AD域包含完整性校验值ICV, ICV是一种报文认证编码MAC或者MAC算法生成的截断代码

在以下字段计算MAC值:

- IP报头: 传输过程中不变的部分和AH SA终点可以预测的部分参与计算MAC
 - 对于可变部分和不可预测的部分全部置0, 便于在源端和目的端计算
- AH报头中不计算认证数据域; 认证数据域被置成0, 便于在源端和目的端计算
- 整个上层协议数据, 如TCP/UDP数据, 假设在传输过程中不变

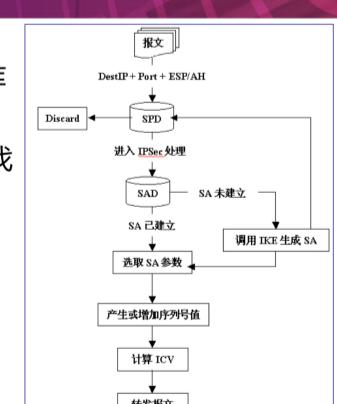


• AH对接收数据包 (Inbound)的处理



- 从端口收到输入的数据包, 解析出其SA三元组, 查找SADB
 - 如查找到一个匹配的SA条目, 将该SA参数与数据包相关域参数进行比较: 参数一致, 处理该数据包; 参数不一致, 丢弃该数据包
 - 如果没有找到匹配的SA条目, 丢弃该数据包
- 检查序列号(可选, 针对重放攻击)
 - 使用滑动窗口来检查序列号重放
- 计算数据包的ICV, 将其和数据包中的值进行比较:
 - 相等, 恢复数据包, 转IP协议栈进行路由
 - 不相等, 丢弃该数据包并审计事件

• AH对输出数据包(Outbound)的处理过程



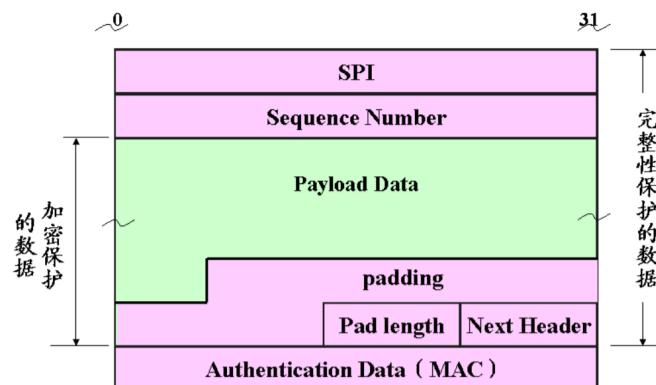
- 从IP协议栈中收到需要转发的数据包, 使用相应的选择子查找安全策略数据库SPDB, 获取对数据包的安全策略
- 如果确定对数据包实施IPsec处理, 查找安全关联数据库SADB
 - 如SA已经建立, 选取参数, 计算ICV, 转发报文
 - 如SA未建立, 调用IKE协商新的SA; 在选取参数, 计算ICV, 转发报文

IPsec:封装安全载荷ESP

- 封装安全载荷ESP提供保密性服务包括报文内容保密和流量限制保密, ESP还可以提供和AH同样的认证服务
- ESP的格式:

● Security Parameters Index (安全参数索引)

- SPI为数据报识别安全联合的32位伪随机值，
- SPI=0被保留，表明“没有安全关联存在”



● Sequence Number (序列号)

- 32bits，单调递增的计数器，用于防范重放类型的攻击

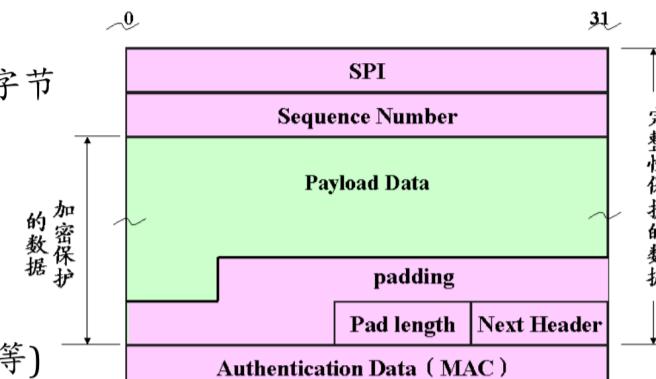
● Padding(填充域)

- 可变长域，范围在0~255字节

● Pad Length(填充域长度)

● Next header(邻接头)

- 8 bits，标识载荷中的封装方式或协议 (TCP/UDP/ICMP/AH/ESP等)



● Authentication Data (认证数据AD)

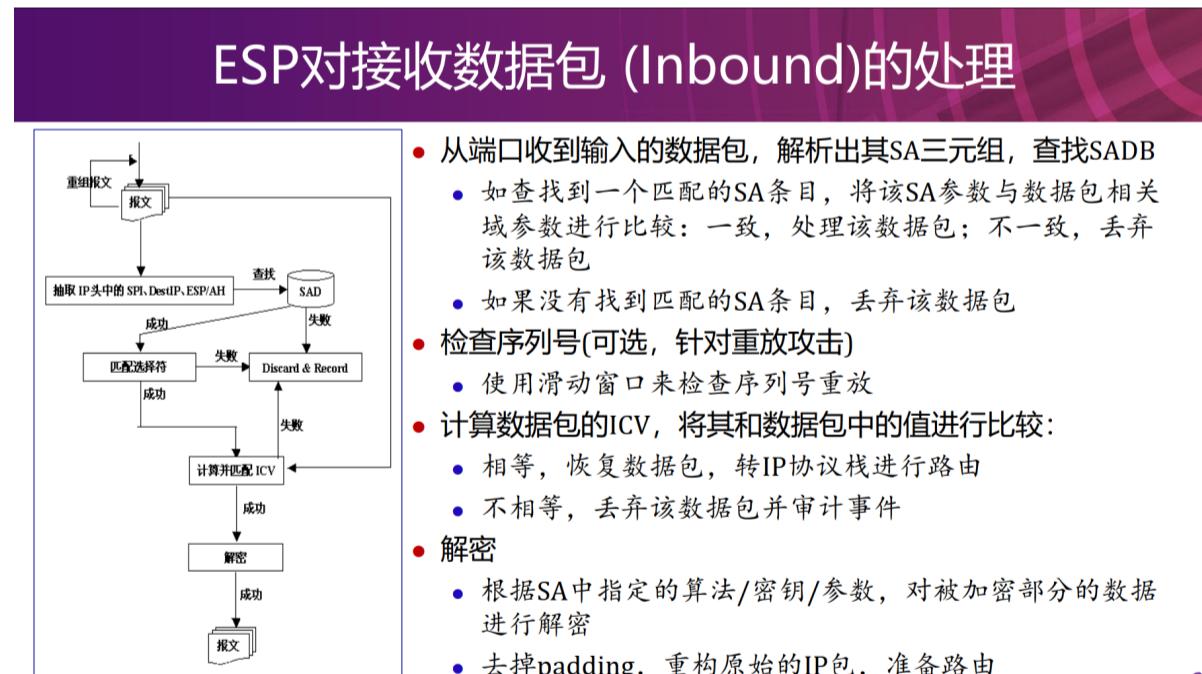
- 变长域，必须是32 bits的整数倍
- 包含完整性校验值ICV或者包的MAC

● ESP的加密算法、认证算法和填充域

- 载荷数据、填充数据、填充长度和邻接头域都在ESP中被加密，可用加密算法有3DES、RC5、IDEA、CAST、Blowfish
- 与AH相同，ESP使用默认截断至96位的MAC
- 填充域的功能如下：

- 如果加密算法需要明文是某个字节的倍数，则填充域可以用于扩展明文长度
- 填充域用来保证ESP格式需要填充长度和邻接头域为右对齐的32位字
- 增加额外的填充域可以隐藏载荷的实际长度，并提供部分流量保护

● ESP对接收数据包的处理：

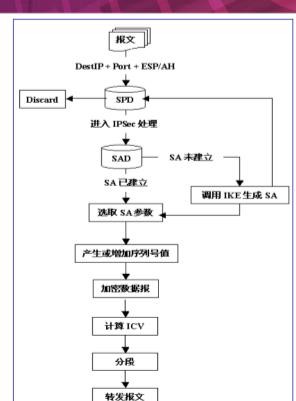


- 从端口收到输入的数据包，解析出其SA三元组，查找SADB
 - 如查找到一个匹配的SA条目，将该SA参数与数据包相关域参数进行比较：一致，处理该数据包；不一致，丢弃该数据包
 - 如果没有找到匹配的SA条目，丢弃该数据包
- 检查序列号(可选，针对重放攻击)
 - 使用滑动窗口来检查序列号重放
- 计算数据包的ICV，将其和数据包中的值进行比较：
 - 相等，恢复数据包，转IP协议栈进行路由
 - 不相等，丢弃该数据包并审计事件
- 解密
 - 根据SA中指定的算法/密钥/参数，对被加密部分的数据进行解密
 - 去掉padding，重构原始的IP包，准备路由

64

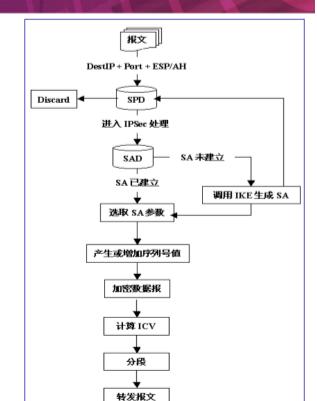
● ESP对输出数据包(Outbound)的处理过程[1]

- 从IP协议栈中收到需要转发的数据包，使用相应的选择子查找安全策略数据库SPDB，获取对数据包的安全策略
- 如果确定对数据包实施IPsec处理，查找安全关联数据库SADB
 - 如果SA已经建立，选取参数，计算ICV，转发报文
 - 如果SA未建立，调用IKE协商新的SA；在选取参数，计算ICV，转发报文
- 产生序列号：防止重放攻击



● ESP对输出数据包(Outbound)的处理过程[2]

- 加密
 - 封装必要的数据，放到payload data域中
 - 不同的模式，封装数据的范围不同
 - 增加必要的padding数据
 - 加密操作
- 计算ICV
 - 针对加密后的数据计算ICV
- 分片
 - 根据最大传输单元MTU，将数据包分成适当大小的包发往目的节点



IPsec：安全关联（SA）组合

- 单个SA可以实现AH或者ESP，但是不能全都实现
- 安全关联组合（安全关联束）是指提供特定的IPSec服务集所需的一个SA序列，SA可通过两种方式组合成束：
 - 传输邻接：在不使用隧道的情况下，对一个IP包使用多个安全协议；组合AH和ESP的方法仅允许一级组合
 - 隧道迭代：指通过IP隧道应用多层安全协议；由于每个隧道可以在路径上的不同IPSec节点起始和结束，因此该方法允许多层嵌套
- AH和ESP典型组合

Transport	Tunnel

1. [IP1][AH][upper]	4. [IP2][AH][IP1][upper]
◦ 2. [IP1][ESP][upper]	5. [IP2][ESP][IP1][upper]
3. [IP1][AH][ESP][upper]	

• 这里upper指上层协议数据
• IP1指原来的IP头
• IP2指封装之后的IP头

IKE：为IPsec管理密钥

IKE简介