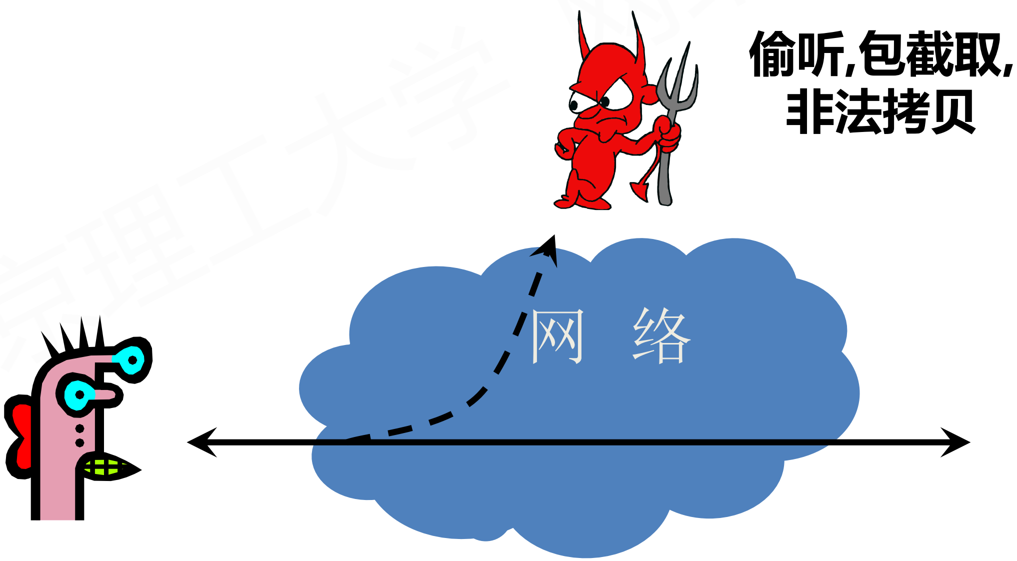
网络信息安全特点

- **网络安全的特点**

机密性、完整性、可用性、可控性、不可否认性

– **机密性：杜绝有用信息泄露给非授权个人或实体；**

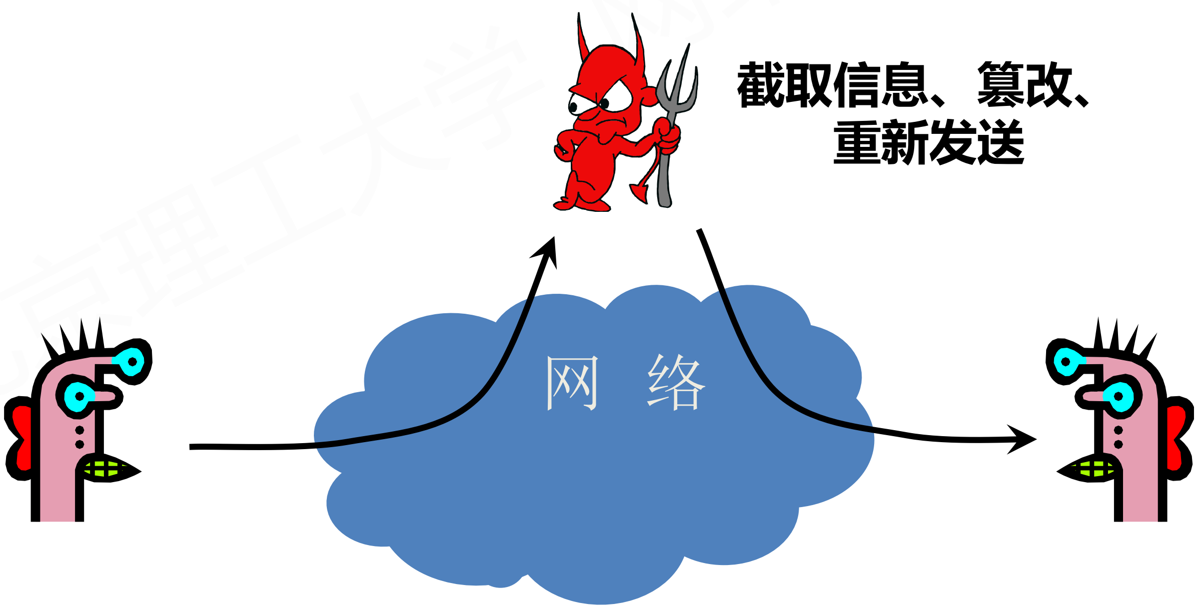


网络信息安全特点

- **网络安全的特点**

  机密性、完整性、可用性、可控性、不可否认性

  – **完整性：信息保持非修改、非破坏和非丢失；**



网络信息安全特点

- **网络安全的特点**

机密性、完整性、可用性、可控性、不可否认性

- **可用性：可被授权实体正常访问；**

网络信息安全特点

- **网络安全的特点**

  机密性、完整性、可用性、可控性、不可否认性

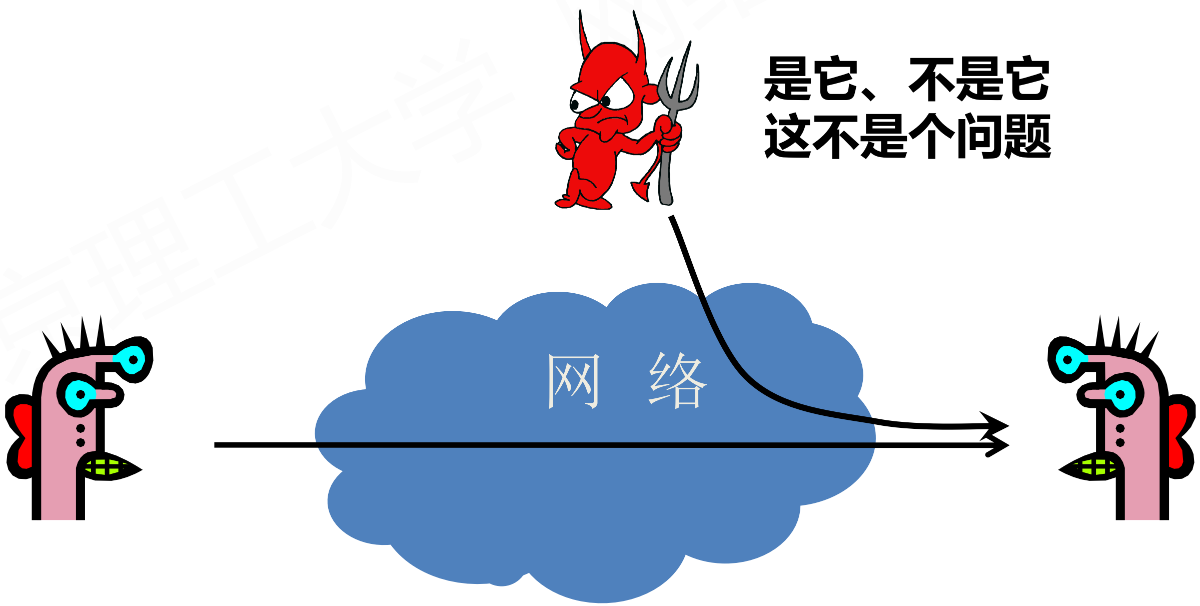
  – **可控性：对网络中信息传播及内容能实现有效控制；**

网络信息安全特点

- **网络安全的特点**

机密性、完整性、可用性、可控性、不可否认性

– **不可否认性：所有参与者不可否认或者抵赖本人的真实身份。**



是它、不是它
这不是个问题

密码算法的分类

- 古典密码算法和现代密码算法
  - 根据算法和密钥是否分开来区分
- 对称密钥密码和非对称密钥密码
  - 根据加密和解密是否使用相同的密钥来区分
- 分组密码和序列密码
  - 根据每次操作的数据单元是否分块来区分
- 双向加密和单向加密
  - 根据明文加密后是否需要还原来区分

非对称密钥密码算法

- **对称密钥密码系统的缺点**
  - 密钥分发需要经过安全通道
  - 无法用于鉴别身份(数字签名)
  - 密钥管理复杂，O(n^2)
- **非对称密钥密码系统**
  - 1976年由W. Diffie和M. Hellman提出
  - 受到了Ralph Merkle工作的启发



非对称密钥密码算法

- **非对称密钥密码系统**
  - **两个密钥，一公一私**
  - **对密钥分配、数字签名、认证等具有深远影响**
  - **相关算法真正基于数学理论，而不是代替和换位**
  - **密码学史上一次真正意义的革命**

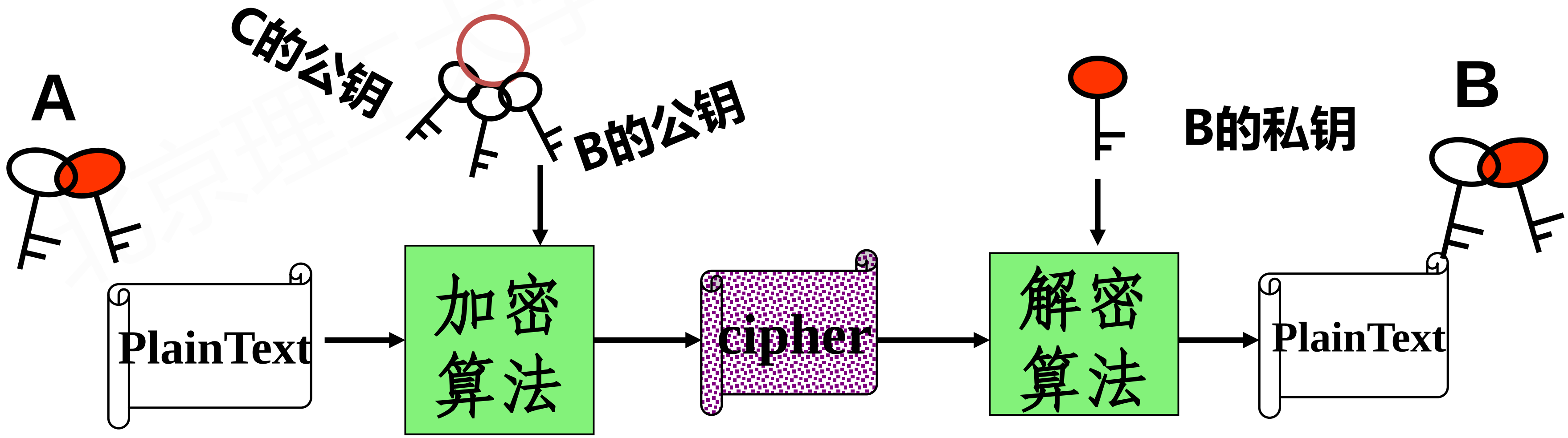
非对称密钥密码算法

- **常用的非对称密钥密码算法**
  - **Diffie-Hellman密钥交换协议**
  - **ElGamal**
  - **RSA**
  - **ECC，椭圆曲线**
  - **Cramer-Shoup算法**

非对称密钥密码算法

信息接收

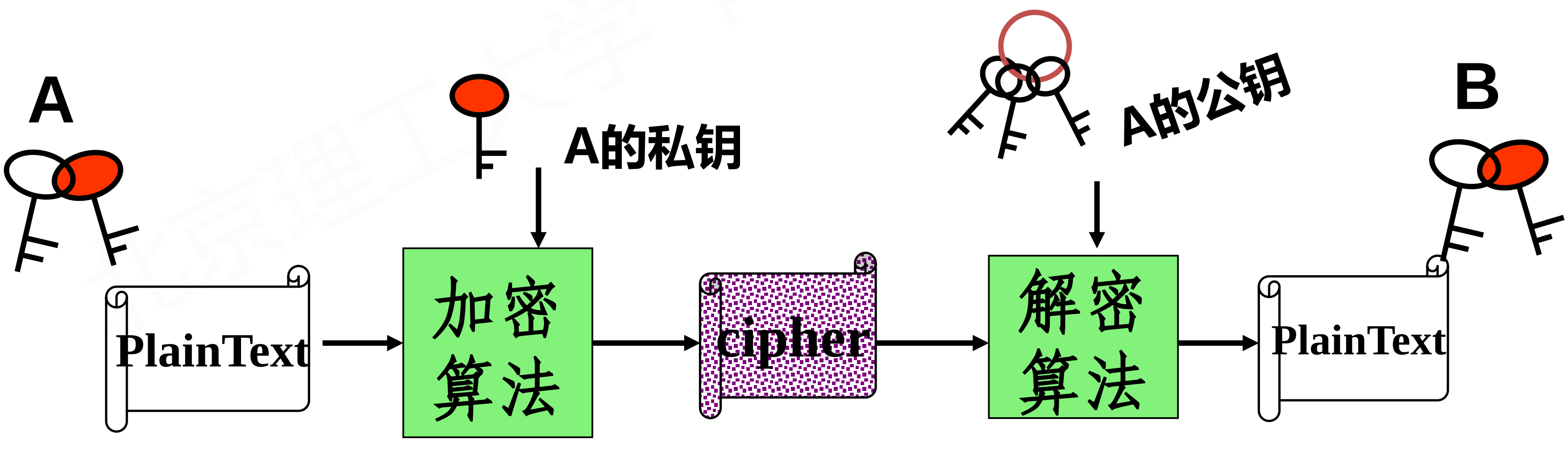
- **加密原理**
  - 每个通信实体有一对密钥，公钥公开，用于加密和验证签名，私钥保密，用作解密和签名
  - A向B发送消息，用B公钥加密；B收到后，用其私钥解密



非对称密钥密码算法

来源证明

- **签名原理**
  - A向B发送消息，用A的私钥加密——签名过程
  - B收到密文后，用A的公钥解密——验证签名过程

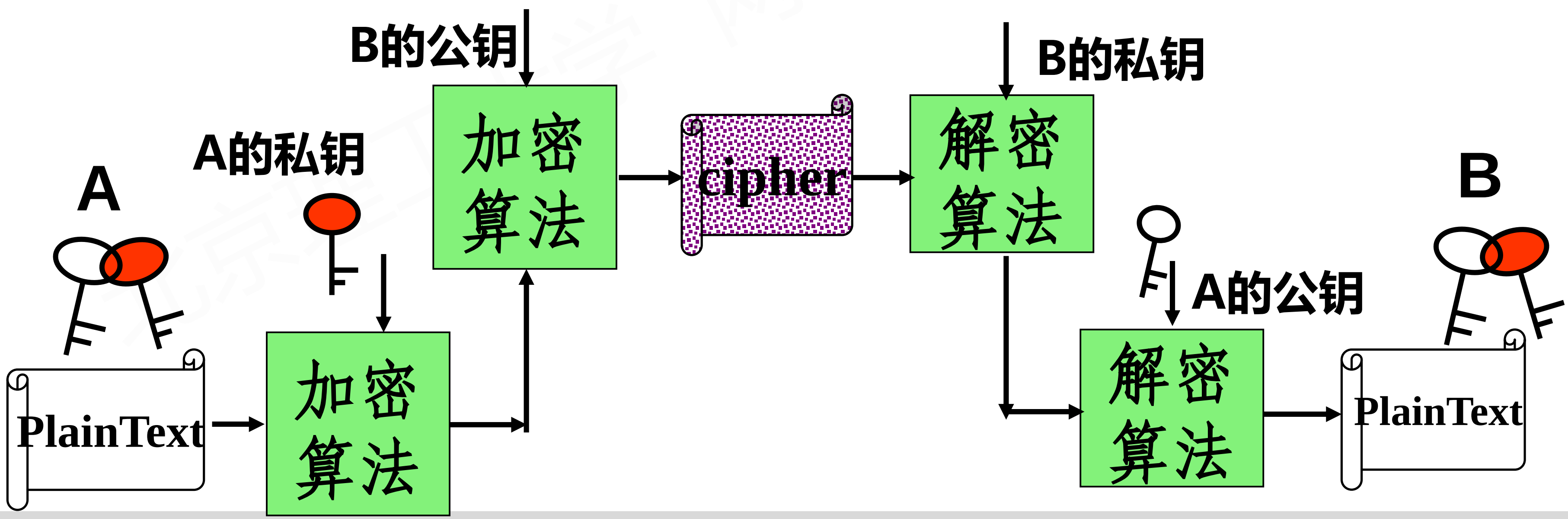


非对称密钥密码算法

- **签名和加密同时使用**

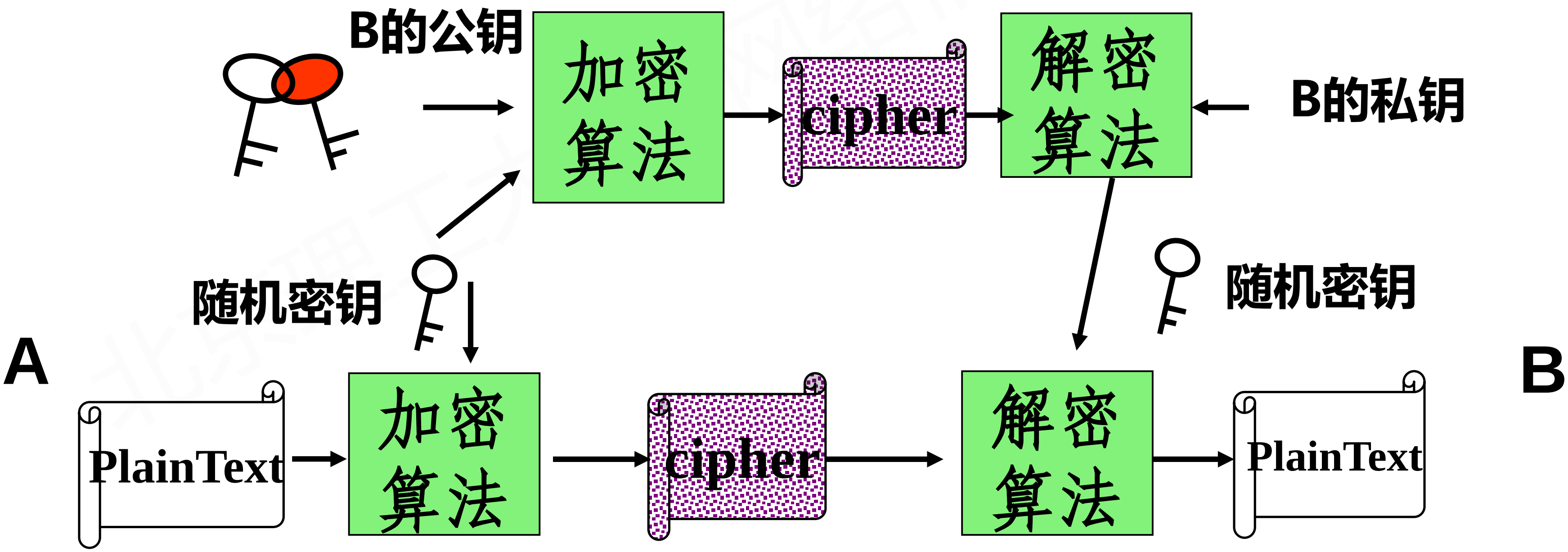
可证明来源的信息接收

  - A向B发送消息，用A的私钥加密——签名过程
  - B收到密文后，用A的公钥解密——验证签名过程



非对称密钥密码算法

- **与对称密钥算法同时使用**　**高速数据加密传输**
  - 对称密钥算法加密消息，非对称密钥算法加密密钥



非对称密钥密码算法

- **算法要求**
  - 参与双方A和B都包含一对密钥，且密钥产生容易（k_a，k_a^{-1}）和（k_b，k_b^{-1}），A向B发送消息
  - 已知k_b，A的加密操作是容易的，$C = E_{kb}(P)$
  - 已知k_b^{-1}，B对密文解密操作是容易的，$P=D_{kb}^{-1}(C)$
  - 已知k_b，求k_b^{-1}在计算上不可行（由公钥无法计算私钥）
  - 仅知k_b和C，恢复P在计算上不可行（仅知公钥，无法破解）

非对称密钥密码算法

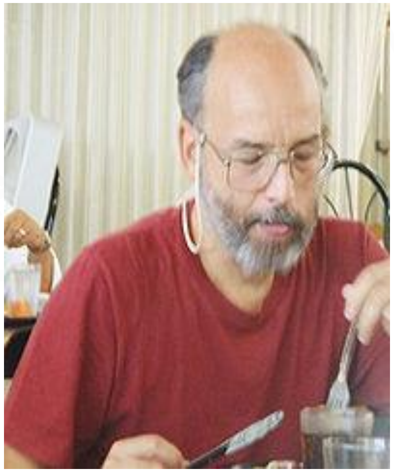
- **非对称密钥密码算法的理解**
  - **公开密钥算法和对称密钥算法那种更安全?**

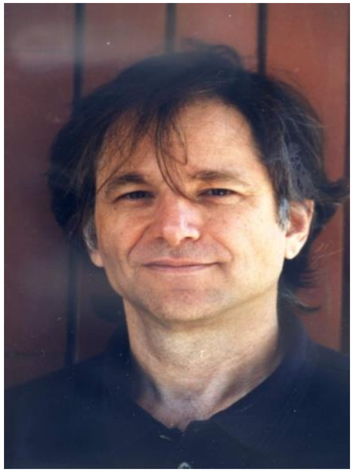
    任何一种算法的安全性都依赖于密钥的长度和破解的工作量，从密码分析角度来说，两种方法具有同样优势
  - **对称密钥算法过时了吗?**

    公开密钥算法相对很慢，适合用在数字签名和密钥管理等片段加密的情况，对称密钥仍然是主流方法。

RSA算法

- **RSA算法介绍**
  - **Ron Rivest， Adi Shamir， Leonard Adleman**





  - **2002年，图灵奖获得者**

RSA算法

- **RSA算法介绍**
  - RSA安全性基于分解极大数的困难性（两素数积的因式分解）
  - 至今，只有短的RSA密钥才可以被强力破解
  - 至今，世界上还没有任何可靠的攻击RSA算法的方式
  - 只要密钥足够长，RSA加密的信息在计算上不能被破解
  - RSA算法于1977年以论文形式发表，1983年在美国申请了专利， U.S. Patent 4,405,829，但已经过时，其他国家没有被授予专利

RSA算法

- **数论基础**
  - **模运算的特点：可交换、可结合、可分配**

$$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$

$$(a-b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$$

$$(a\times b) \bmod n = ((a \bmod n)\times (b \bmod n)) \bmod n$$

$$(a \times (b+c)) \bmod n = (a\times b) \bmod n + (a\times c) \bmod n$$

RSA算法

- **数论基础**
  - **幂的模运算**

$$m^2 \bmod n = (m \times m) \bmod n = (m \bmod n)^2 \bmod n$$

$$m^4 \bmod n = (m^2 \bmod n)^2 \bmod n$$

$$m^8 \bmod n = ((m^2 \bmod n)^2 \bmod n)^2 \bmod n$$

$$m^{25} \bmod n = (m \times m^8 \times m^{16}) \bmod n$$

RSA算法

- **欧拉函数** $\phi(n)$
  - 正整数n，欧拉函数是小于或等于n的正整数中与n互质的数的数目

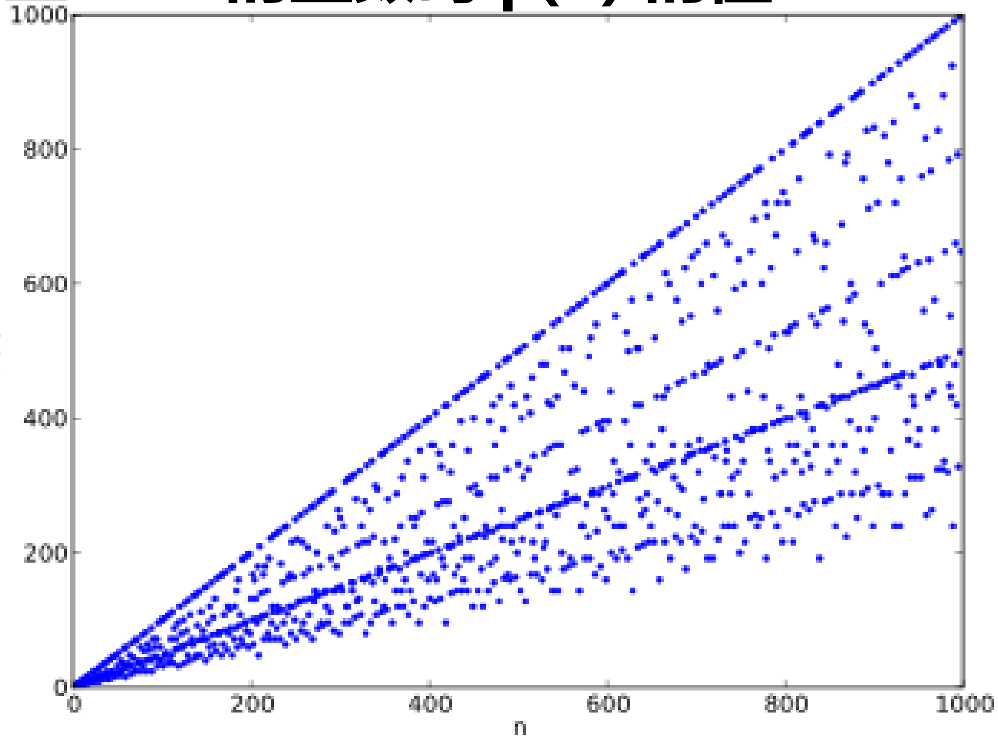
    $\phi(3)=|\{1, 2\}| =2$

    $\phi(7)=|\{1, 2, 3, 4, 5, 6\}| =6$

    $\phi(10)=|\{1, 3, 7, 9\}| =4$
  - 如果p是素数，则φ(p)=(p-1)，比如φ(5)、φ(11)
  - 如果p,q 是素数，则$\phi(pq)=\phi(p)\phi(q) = (p-1)(q-1)$

RSA算法

- **欧拉函数 *φ(n)***
  - 当n为1至1000的整数时φ(n) 的值



RSA算法

- **欧拉定理（费马-欧拉定理）**
  - 若整数*m*和*n*互素，则 $m^{\phi(n)} \equiv 1 \mod n$
  - 欧拉定理用来简化幂的模运算
  - 例如，计算7^{222}的个位数，即 $7^{222} \mod 10$

    因为7和10是互质，而且φ(10) = 4，

    根据欧拉定理知， $7^4 \equiv 1 \pmod{10}$

    所以：$7^{222} \equiv 7^{4\times55+2} \equiv (7^4)^{55} \times 7^2 \equiv 1^{55} \times 7^2 \equiv 49$

    $\equiv 9 \pmod{10}$

RSA算法

- **算法组成**
  - **密钥生成算法（产生公钥和私钥）**
  - **加密算法**
  - **解密算法**

RSA算法

- **密钥生成算法** RSA的安全性基于分解极大数的困难性
  - 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$\phi(N)$互素的整数e，即 $1<e<\phi(N)$

    e作为公钥公开
  - 寻找d，使得 $de \equiv 1 \bmod \phi(N)$， d作为私钥保密

$p=7, q=17$

$N=119$

$\Phi(N)=96$

公钥$e=5$

$5d \equiv 1 \bmod 96$

得到$d=77$

RSA算法

- **加密算法（加密过程）**
  - **密钥对　({e, N}，{d, N})**　　({5,119}, {77,119})
  - **把待加密内容分成k比特分组， $k \le \log_2 N$，写成数字M，**

    则：$C = M^e \bmod N$　　$C=M^5 \bmod 119$
- **解密算法（解密过程）**

    $M = C^d \bmod N$　　$M=C^{77} \bmod 119$
  - **可以证明，解密是正确的**

RSA算法

- **加密数字例子**
  - 明文M=19， $19^5 \equiv 66 \bmod 119$ ， 密文C= 66
  - 解密过程： $66^{77} \bmod 119 = ?$
- **加密字符串例子**
  - 每次读取字符串中多个字节，变成M进行加解密
  - RSA　　focusRSA
  - 5CB6CD6BC　　9F47D51C325D67B5CB6CD6BC
  - RSA在实现上需要结合填充方法使用

RSA算法

- **填充**
  - 在消息中填充随机信息，使得密文和明文的对应关系存在随机性
  - ANSI X.923 （零+个数）

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 04 |
```

  - ISO 10126 （随机数+个数）

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 81 A6 23 04 |
```

  - RSA实现中采用PKCS#1填充方法（个数）

```
... | DD DD DD DD DD DD DD DD | DD DD DD DD 04 04 04 04 |
```

RSA算法的安全性

- **攻击方法**
  - **蛮力攻击：尝试所有密钥**
  - **密码分析：等效于对两个素数乘积N的因子分解（求p和q）**
- **大数的因子分解是数论中的一个经典难题**

十进制数字位数	近似比特数	得到的数据	MIPS年
100	332	1991	7
110	365	1992	75
120	398	1993	830
129	428	1994	5000
130	431	1996	500
193	640	2005	2.2GHz-CPU 运算30年

RSA算法的安全性

- **大数的因子分解是数论中的一个经典难题**
  - **一般使用1024位密钥，证书认证机构采用2048位**

RSA-640

16347336458092538484431338838650908598417836700330
92312181110852389333100104508151212118167511579

and

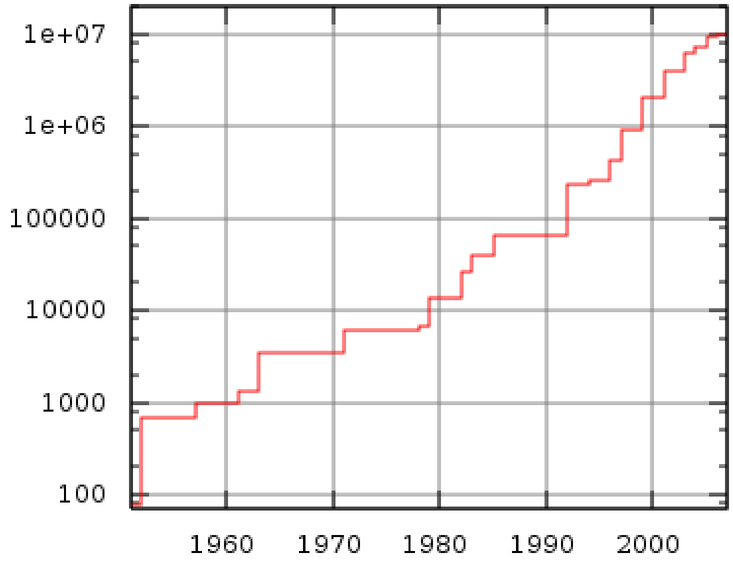
19008712816648221131268515739354139754718967899685
15493666638539088027103802104498957191261465571

RSA算法的安全性

- **已知最大素数**
  - **欧几里得证明存在无限多个素数**
  - **已知最大素数于2008年8月23日发现（梅森素数）UCLA**

$$2^{43,112,609} - 1$$

  - **十进制12,978,189位**
  - **悬赏第一个1亿位和10亿位素数**



RSA算法

- **RSA的算法性能**
  - 软件实现比DES软件慢100倍
  - 硬件实现比DES硬件慢1000倍

	512位	768位	1024位
加密	0.03	0.05	0.08
解密	0.16	0.48	0.93
签名	0.16	0.52	0.97
验证	0.02	0.07	0.08

椭圆曲线密码系统

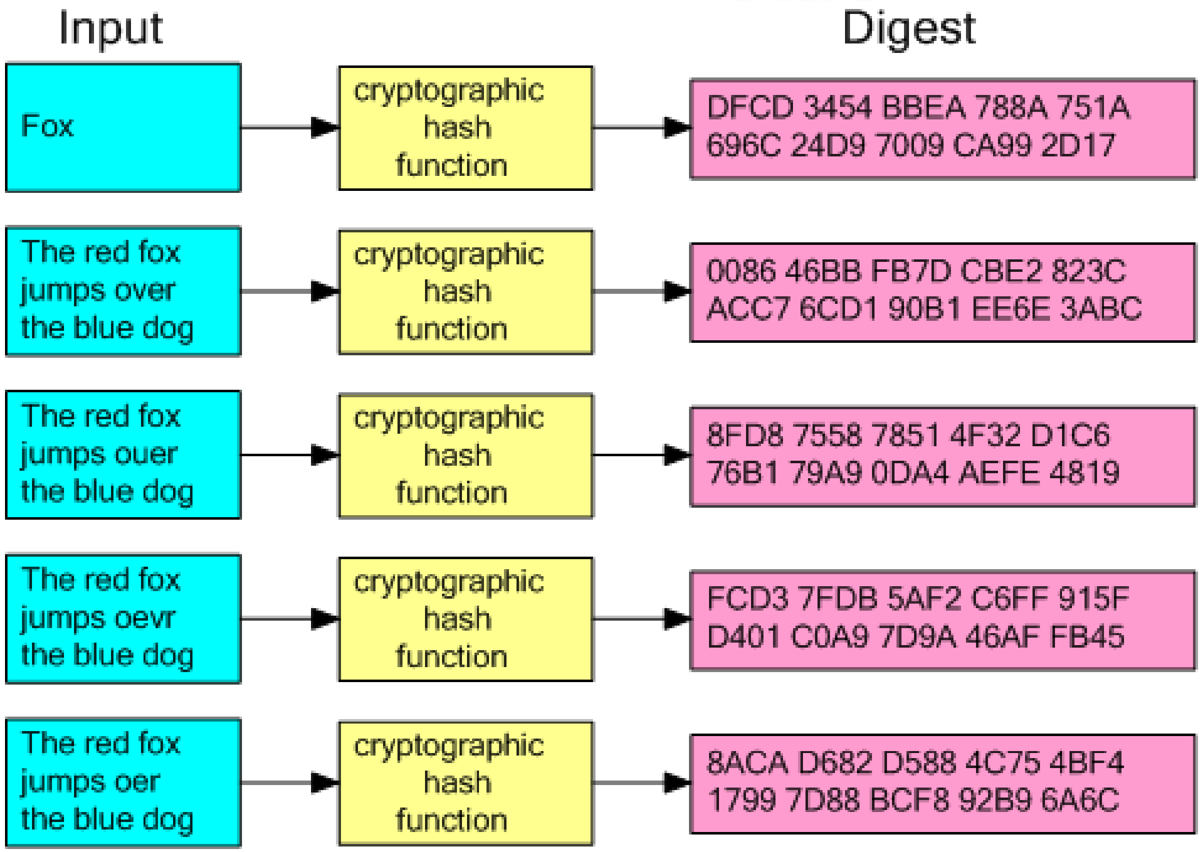
- **算法概况**
  - **Elliptic Curve Cryptography，缩写为ECC**
  - **理论基础是解决椭圆曲线离散对数的困难性**
  - **主要优势是使用更短的密钥达到与RSA相同的安全性**
  - **160位密钥可达到RSA1024位密钥的安全性，甚至更高**

本节大纲

- **非对称密钥密码算法**
- **单向散列函数**

单向散列函数

- **Hash：哈希函数、散列函数**
  – $h = H(m)$

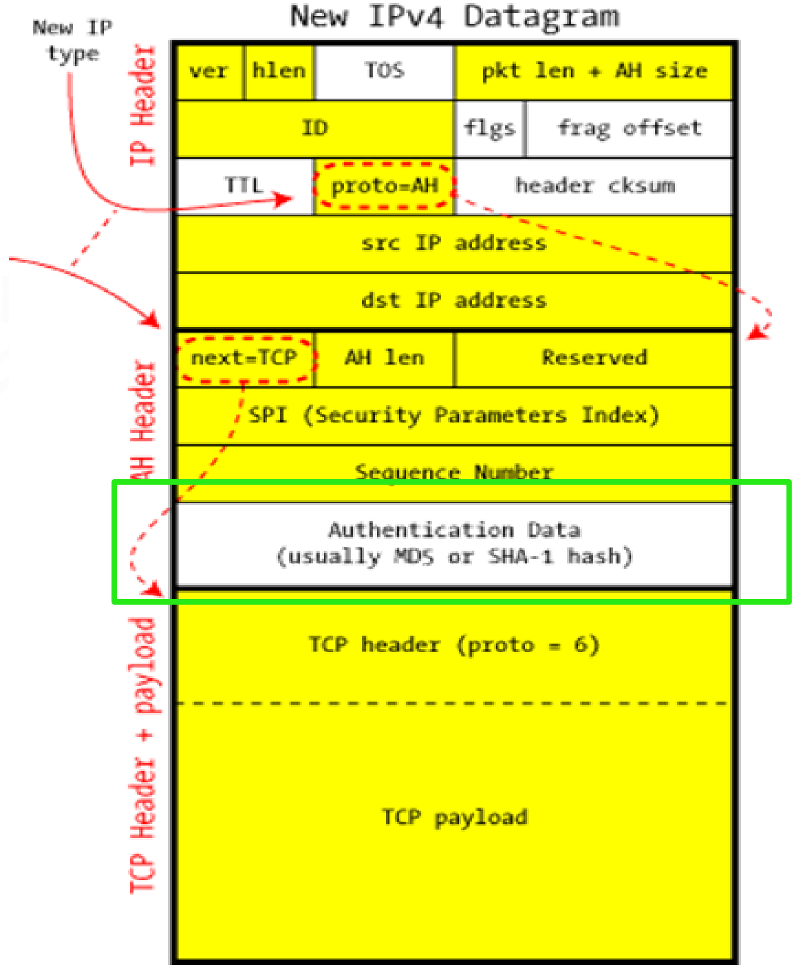


单向散列函数

- **H的特点和要求**
  - 可以操作任意长度内容*m*
  - 给定*m*，计算*h*是容易的
  - 给定任意*m*，产生的h的长度固定
  - 给定*h*，寻找*m*′，使得$h = H(m')$是困难的
  - 寻找任何(m,m')，$m \neq m'$，使得$H(m) = H(m')$计算上不可行

单向散列函数

- **常用的单向散列函数**
  - *MD2、MD4、MD5*
  - *SHA-0、SHA-1*
  - *SHA-256/224*
  - *SHA-512/384*
  - 例如：IPSec安全协议

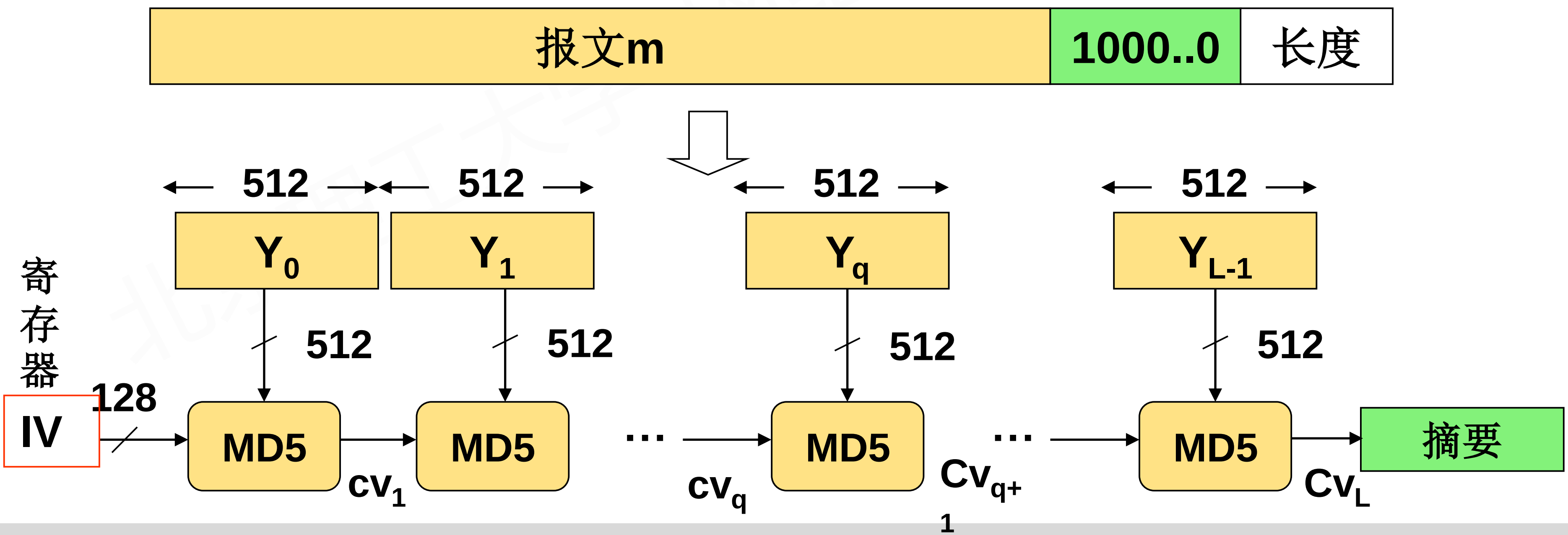


MD5算法

- **算法历史**
  - MD5是1992年，由Ronald Rivest设计
  - MD, MD2, MD3, MD4, MD5, MD6
  - MD：Message Digest，信息摘要算法
  - 对任意输入均产生128bit的输出
  - 基于32位的简单操作，易于软件实现
  - 简单而紧凑，没有复杂的程序和大数据结构
  - 详细：RFC 1321

MD5算法

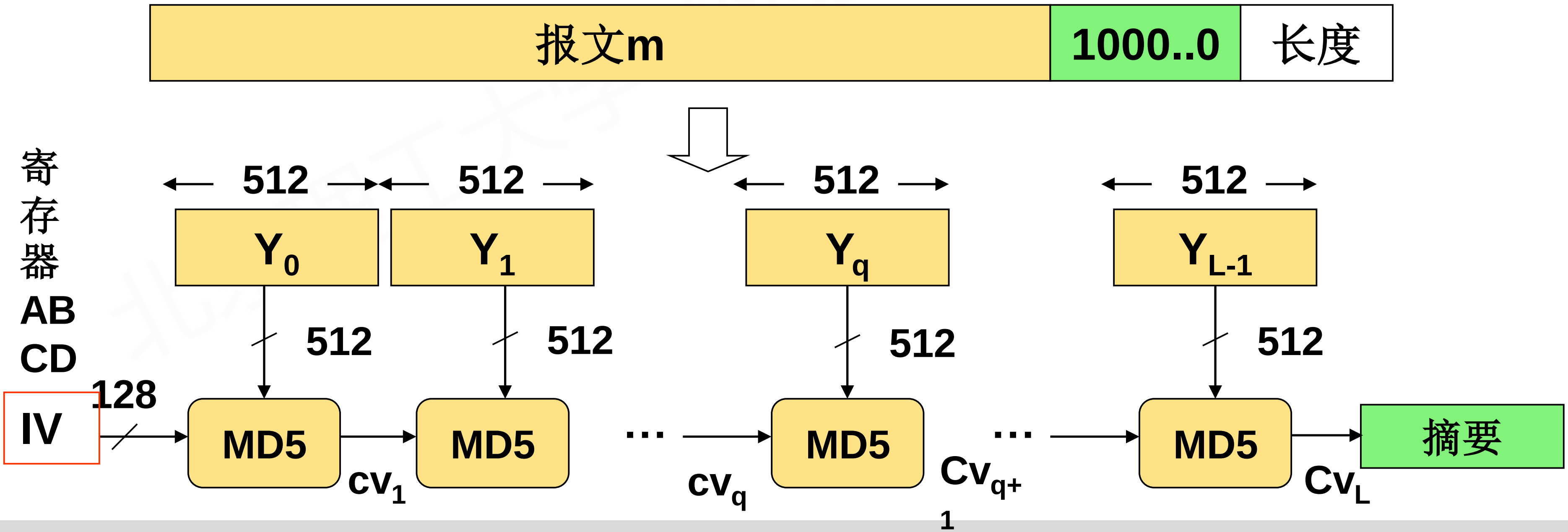
- **算法步骤**
  - Step1：填充，使报文的长度为512减64位
  - Step2：将填充前的长度写入最后64位，总长为512整数倍



MD5算法

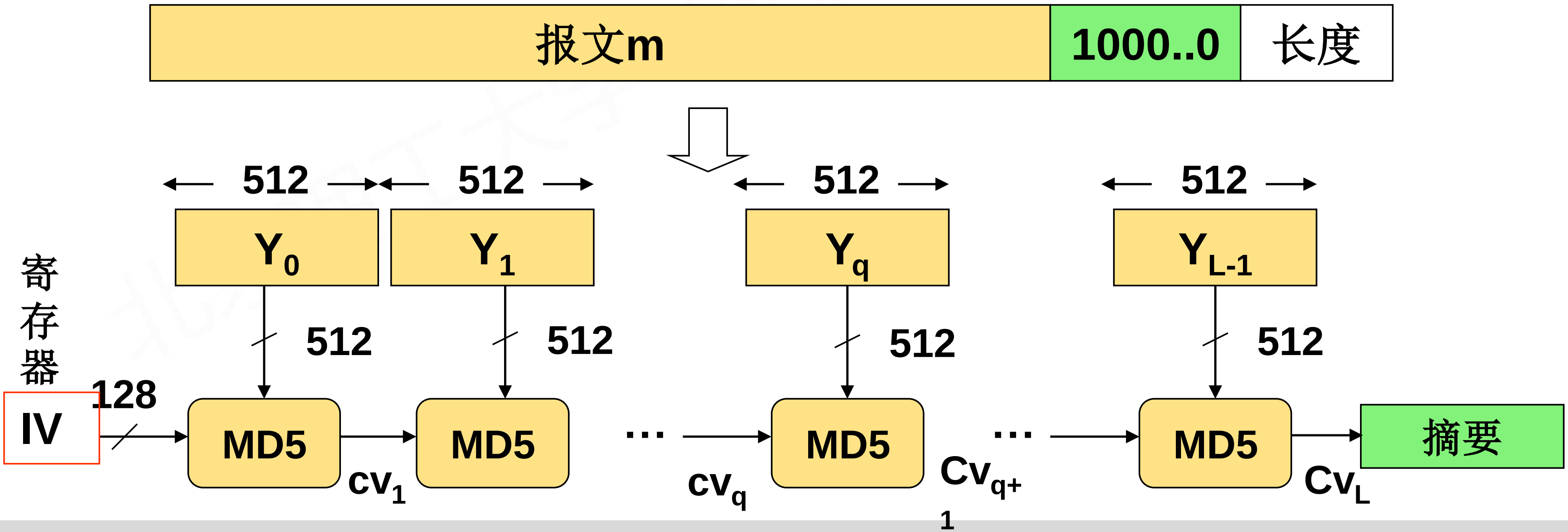
- **算法步骤**
  - **Step3：初始化4个32位寄存器**

A= 01 23 45 67；B= 89 AB CD EF；C=FE DC BA 98；D=76 54 32 10

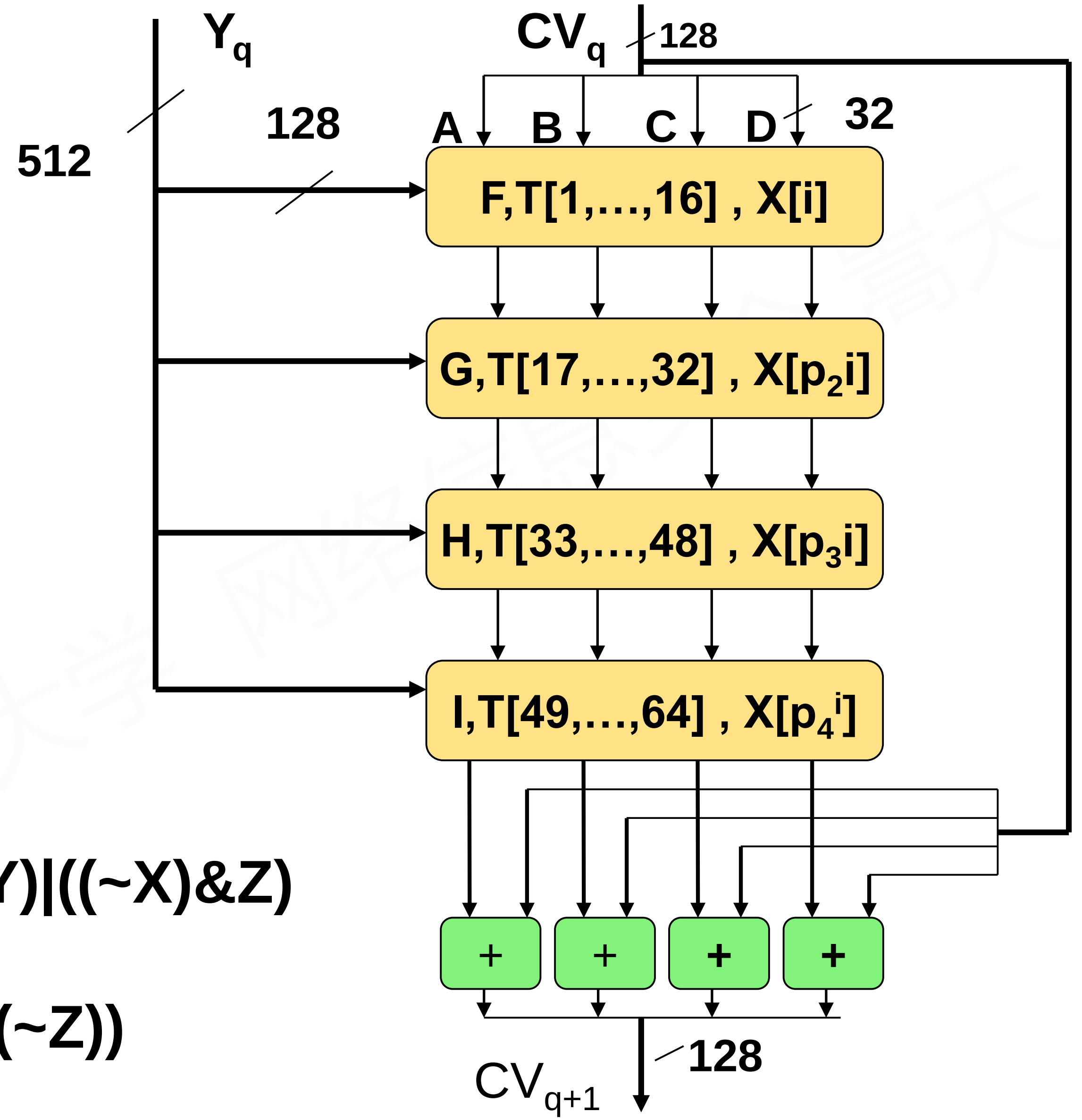


MD5算法

- **算法步骤**
  - Step4：处理每个报文分组，核心是4轮循环的压缩函数



- **算法步骤**
  - Step4

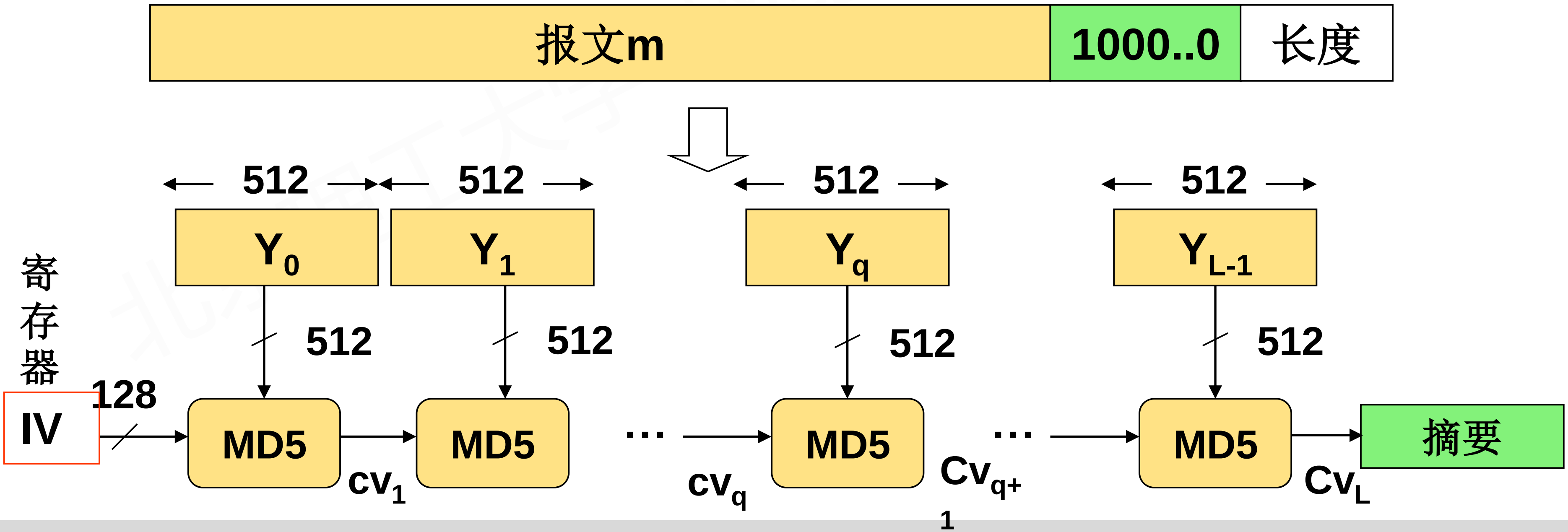


F(X,Y,Z) =(X&Y)|((~X)&Z)

I(X,Y,Z)=Y^(X|(~Z))

MD5算法

- **算法步骤**
  - Step5：所有L 个 512 bit 的分组处理完之后，第L阶段的输出便是128bit 的报文摘要



SHA算法

- **算法介绍**
  - SHA：Security Hash Algorithm，是美国政府密码标准
  - SHA-0和SHA-1算法产生160位的摘要信息（散列值）
  - 单次输入信息最长 $(2^{64} – 1)$ 位
  - 基于MD4设计，和MD5方法十分类似

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)
SHA-0		160	160	512	$2^{64} - 1$
SHA-1		160	160	512	$2^{64} - 1$
SHA-2	*SHA-256/224*	256/224	256	512	$2^{64} - 1$
	SHA-512/384	512/384	512	1024	$2^{128} - 1$

SHA算法

- **算法的安全性**
  - SHA-1在安全协议中广为使用：TLS/SSL、PGP、SSH和IPSec
  - SHA-1被视为MD5的后继者
  - 2005年2月，王小云团队对SHA-0进行破解，在2^{39}的计算复杂度内可以找到碰撞（每秒10万次运算，63天）
  - 2005年8月，王小云和姚期智夫妇对SHA-1进行破解，在2^{63}的计算复杂度内可以找到碰撞
  - SHA算法已经不安全了，但尚无完善替换算法（SHA-2/SHA-3）