



中山大學
SUN YAT-SEN UNIVERSITY

Module I. Fundamentals of Information Security

Chapter 2

Cryptographic Techniques

Information Security: Theory & Applications

School of Data & Computer Science, Sun Yat-sen University

Outline

- **2.1 Cryptology Introduction**
 - Introduction
 - History
 - Concepts & Items
- **2.2 Symmetric Key Cryptographic Algorithms**
 - Introduction
 - Types & Modes
 - Data Encryption Standard (DES)
 - Advanced Encryption Standard (AES)



Outline

- 2.3 Mathematical Foundations of Public-Key Cryptography
 - Prime factorizations of integers
 - The *Euclidean* Algorithm
 - *Bézout's* Theorem
 - Linear Congruence
 - The Extended *Euclidean* Algorithm
 - The Chinese Remainder Theorem
 - *Euler's* φ function
 - *Euler's* Theorem
 - *Fermat's* Little Theorem

Outline

- 2.4 Asymmetric Key Cryptographic Algorithms
 - Introduction
 - The RSA Algorithm
 - Digital Signatures
- **2.5 MAC and Hashing Algorithms**
 - Message Authentication Code
 - Hash Function
 - Message-Digest Algorithm
- 2.6 Typical Applications
 - MD5 and Passwords
 - AES and WiFi Protected Access
 - RSA and e-Business

2.5 MAC and Hashing Algorithms

2.5.1 Introduction

- 消息认证的必要性
 - 网络通信针对消息内容的攻击方法
 - ✧ 伪造消息
 - ✧ 篡改消息内容
 - ✧ 改变消息顺序
 - ✧ 消息重放或者延迟
 - 消息认证
 - ✧ 如果接收方计算的认证信息与收到的匹配，则
 - 接收者可以确信消息未被改变
 - 接收者可以确信消息来自所声称的发送者
 - 如果消息中含有序列号，则可以保证正确的消息顺序

2.5 MAC and Hashing Algorithms

2.5.1 Introduction

- 消息认证的三种方式
 - Message Encryption
 - ✧ 消息加密方法。对整个消息进行加密，以密文作为消息的认证标识。可以采用现有的密码体制实现。
 - Message Authentication Code
 - ✧ 消息认证码。使用用一个公开函数，加上一个密钥，为消息产生一个固定长度的小数据块 (即为消息认证码 MAC，或称密码校验和 Cryptographic Checksum) 作为消息的认证标识，并附加到消息中一起传输。
 - Hash Function
 - ✧ 哈希方法使用一个公开函数，将任意长度的消息映射到一个固定长度的散列值，作为消息认证标识，并附加到消息中一起传输。

2.5 MAC and Hashing Algorithms

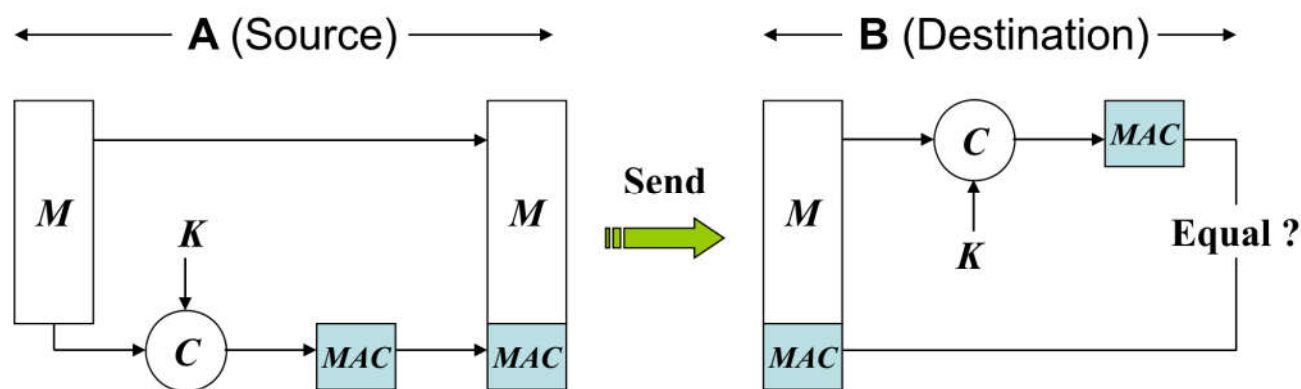
2.5.2 MAC

- 概述
 - 基本结构
 - ✧ MAC 方法使用一个双方共享的秘密密钥，为目标消息生成一个固定长度的小数据块，并加入到消息中。该数据块称为消息认证码 MAC，或密码校验和 (Cryptographic Checksum)。
 - ✧ MAC 方法需要对消息数据的全体进行加密运算，形成速度慢。
 - ✧ MAC 函数类似于加密函数，但不需要可逆性，因此受到攻击的弱点在数学上比加密算法要少。

2.5 MAC and Hashing Algorithms

2.5.2 MAC

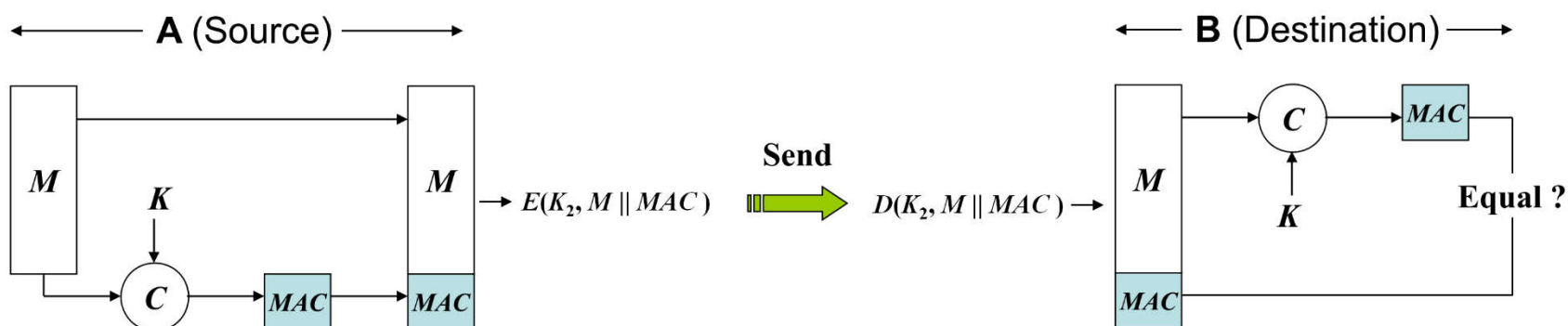
- 概述
 - MAC 方法只用于消息认证
 - ✧ 用户 A 和用户 B 共享密钥 K 。C 为 MAC 函数。对于消息 M ， $MAC = C_K(M)$ 。
 - ✧ 信道传输 $M \parallel MAC$ ，模型只提供消息认证，没有提供机密性。



2.5 MAC and Hashing Algorithms

2.5.2 MAC

- 概述
 - MAC 方法用于与明文相关的消息认证和保密性
 - ✧ 用户 A 和用户 B 共享密钥 K 和 K_2 。 E 和 D 为加解密函数。
 - ✧ 模型提供了与明文相关的消息认证。
 - $MAC = C_K(M)$ ，信道传输 $E(K_2, M \parallel MAC)$ 。
 - 接收端先解密再认证。



2.5 MAC and Hashing Algorithms

2.5.2 MAC

- 概述
 - MAC 方法用于与密文相关的消息认证和保密性
 - ✧ 用户 A 和用户 B 共享密钥 K 和 K_2 。 E 和 D 为加解密函数。
 - ✧ 与密文相关的消息认证：
 - $M' = E(K_2, M \parallel MAC)$
 - $MAC = C_K(M')$ ，信道传输 $M' \parallel MAC$ 。
 - 接收端先认证再解密。



2.5 MAC and Hashing Algorithms

2.5.2 MAC

- 概述
 - MAC 不等于数字签名
 - ✧ 通讯双方共享同一个私有密钥
 - MAC 结构的重要性
 - ✧ 密钥足够长+加密算法足够好 \neq 安全。
 - 攻击案例
 - ✧ $M = (X_1, X_2, \dots, X_t)$
 - ✧ 对 M 产生校验和 $\Delta M = X_1 \oplus X_2 \oplus \dots \oplus X_t$
 - ✧ $MAC = E_K(\Delta M)$
 - ✧ 攻击者选择 $M' = (Y_1, Y_2, \dots, Y_{t-1}, Y_t)$, 使得 Y_t 满足
$$Y_t = Y_1 \oplus Y_2 \oplus \dots \oplus Y_{t-1} \oplus \Delta M$$
 - ✧ 于是 $\Delta M' = \Delta M \Rightarrow E_K(\Delta M) = E_K(\Delta M') \Rightarrow C_K(M') = C_K(M)$
 - ✧ 尽管攻击者不知道 K , 仍然可以伪造消息 M' 。

2.5 MAC and Hashing Algorithms

2.5.2 MAC

- MAC 算法

- 条件

- ✧ 攻击者知道 MAC 函数但不知道密钥 K 。

- 要求

- ✧ 已知消息 M 和 $C_K(M)$ ，要想构造 M' 使得 $C_K(M)=C_K(M')$ 在计算上不可行 (计算上无碰撞)。

- ✧ $C_K(M)$ 均匀分布：随机选择 M 和 M'

$$\Pr[C_K(M) = C_K(M')] = 2^{-|MAC|}$$

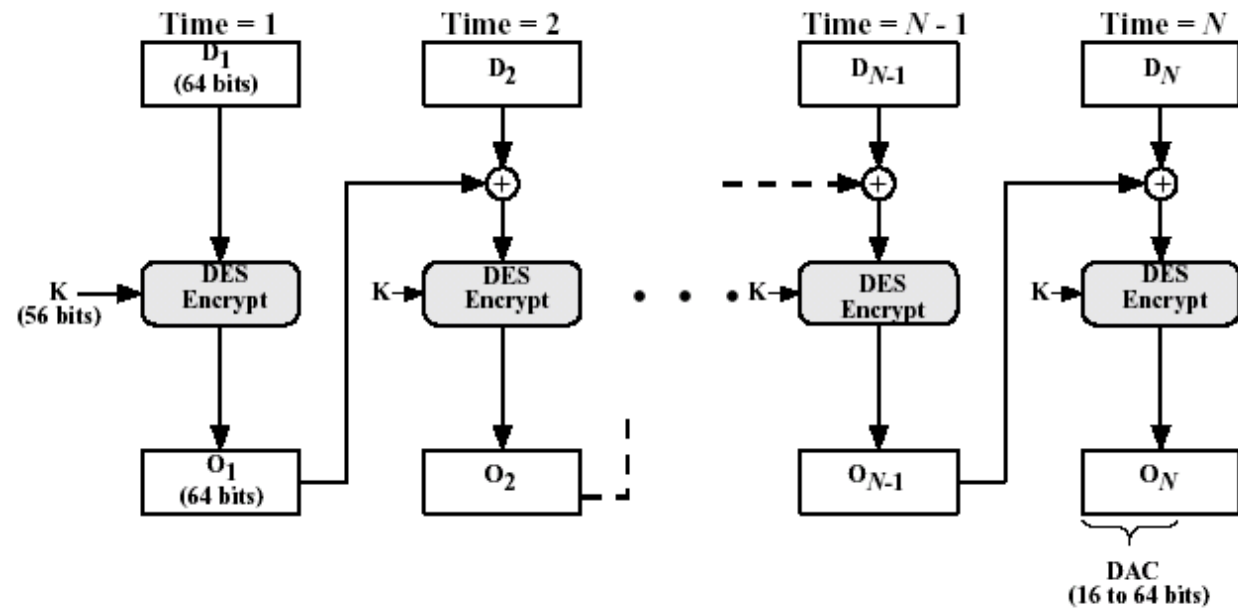
- ✧ 如果 f 是 M 的一个变换 (例如对某些位取反)，那么

$$\Pr[C_K(M) = C_K(f(M))] = 2^{-|MAC|}$$

2.5 MAC and Hashing Algorithms

2.5.2 MAC

- MAC 算法
 - ANSI x9.17 标准 (FIPS PUB 113)
 - ✧ CBC 模式结构, 初始向量为0
 - ✧ 基于 DES 加密算法, 也适用于其他加密算法



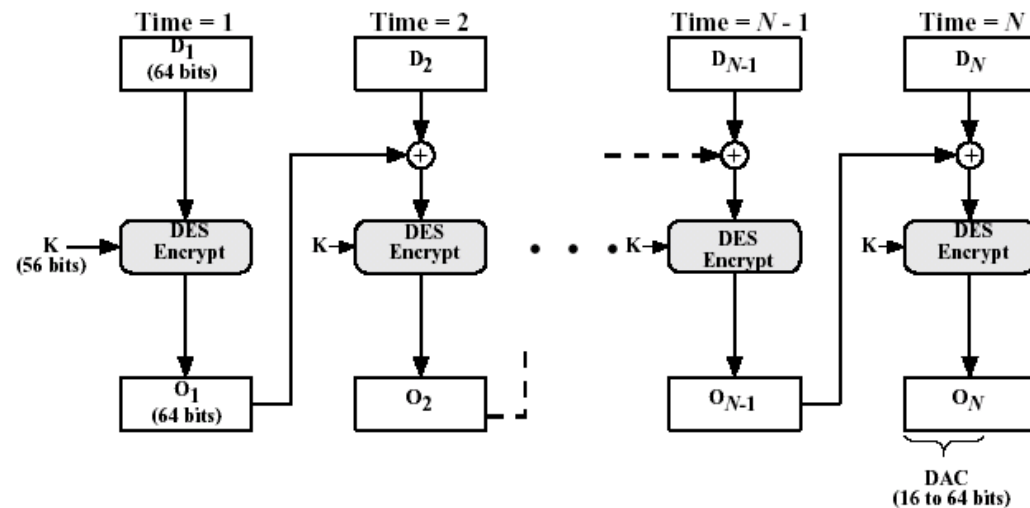
2.5 MAC and Hashing Algorithms

2.5.2 MAC

- MAC 算法
 - ANSI x9.17 标准 (FIPS PUB 113)

✧ 算法:

- $M = (X_1, X_2, \dots, X_t)$
- $\Delta M_1 = E_K(X_1)$
- $\Delta M_{j+1} = E_K(X_{j+1} \oplus \Delta M_j), 1 \leq j < t$
- $MAC = \Delta M_t$



2.5 MAC and Hashing Algorithms

2.5.3 Hash

- 概述
 - 散列 (Hashing) 通过某种特定的函数 (Hash 函数) 将要检索的项与用来检索的索引 (Hash Value 散列值) 关联起来，生成一种便于搜索的数据结构 (Hash List 散列表)。
 - 在安全数据库设计中，散列算法也被用来加密保存在数据库中的密码字符串。数据库中保管的是密码的散列值或密码指纹，而非密码本身，密码验证通过密码指纹验证实现。由于散列算法所计算出来的散列值具有不可逆 (无法逆向倒推原数值) 的性质，在数据库受到攻击的情况下，仍然能够有效地保护密码。

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- 概述
 - 散列方法应用在信息安全领域，将源数据经过散列算法计算出数据指纹 (Data Fingerprint)，用于识别经过传播途径得到的数据是否有误 (通信误码或被篡改)，以保证数据的来源真实性。

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- Hash 函数
 - Hash function : $h = H(M)$
 - ✧ M : 变长消息, $H(M)$: 定长的散列值
 - ✧ 散列函数 H 是一个公开的函数, 将任意长度的消息 M 变换为固定长度的散列码 h 。如 MD5 输出128位, SHA-1输出160位。
 - ✧ 散列函数是一种算法, 算法的输出内容称为散列码或者消息摘要。消息摘要与原始消息唯一对应, 不同的原始消息必须对应于不同的消息摘要, 因此散列函数能用来检测消息的完整性, 验证消息从建立开始到收到为止没有被改变或破坏。
 - ✧ 散列函数又称为: 哈希 (Hash) 函数、数字指纹 (Digital Fingerprint)、压缩 (Compression) 函数、数据认证码 (Data Authentication Code) 等。

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- Hash 函数
 - 对 Hash 函数 H 的要求
 - ✧ $H(x)$ can be applied to any size data
 - ✧ $H(x)$ produces a fixed-length output.
 - ✧ $H(x)$ is relatively easy to compute for any given x
 - ✧ One-way property
 - computationally infeasible to find x such that $H(x) = h$
 - ✧ Weak collision resistance
 - computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$
 - ✧ Strong collision resistance
 - computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- Hash 函数
 - 对 Hash 函数 H 的要求
 - ✧ H 能用于任意大小的分组
 - ✧ H 能产生定长的输出
 - ✧ 对任何给定的 x , $H(x)$ 要相对易于计算, 使得硬件和软件实现成为实际可能
 - ✧ 对任何给定的码 h , 寻找 x 使得 $H(x) = h$ 在计算上是不可行的, 即单向性 (one-way)
 - ✧ 对任意给定的分组 x , 寻找不等于 x 的 y , 使得 $H(x) = H(y)$ 在计算上是不可行的, 即弱抗冲突性 (Weak Collision-free)
 - ✧ 找到任何满足 $H(x) = H(y)$ 的一对数 (x, y) , 在计算上是不可行的, 即强抗冲突性 (Strong Collision-free)

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- 对散列方法的攻击

- 生日攻击理论

- ✧ 若 $k \geq 1.18 \times 2^{m/2} \approx 2^{m/2}$ ，则 k 个在 $[1, 2^m]$ 的随机数中有两个数相等的概率不低于0.5；若 $k \geq 0.83 \times n^{1/2}$ ，两个在 $[1, n]$ 的 k 个随机数集合有交集的概率不小于0.5。

- ✧ 因此，当 Hash 算法选用 N 位的 Hash 值时，两组消息（选择 $k=N/2$ ）中有一对消息产生相同 Hash 值的概率超过0.5。

- MD5 算法输出摘要为128位，要找出至少2个消息的 Hash 值相同的概率大于0.5，则利用生日攻击需要进行的计算量 k 或时间复杂度为 2^{64} 。类似地，SHA-1 算法输出摘要为160位，攻击需要进行的计算量为 2^{80} 。

- 对策

- ✧ 采用足够长的 Hash 值：64 -> 128 -> 160 -> 256

2.5 MAC and Hashing Algorithms

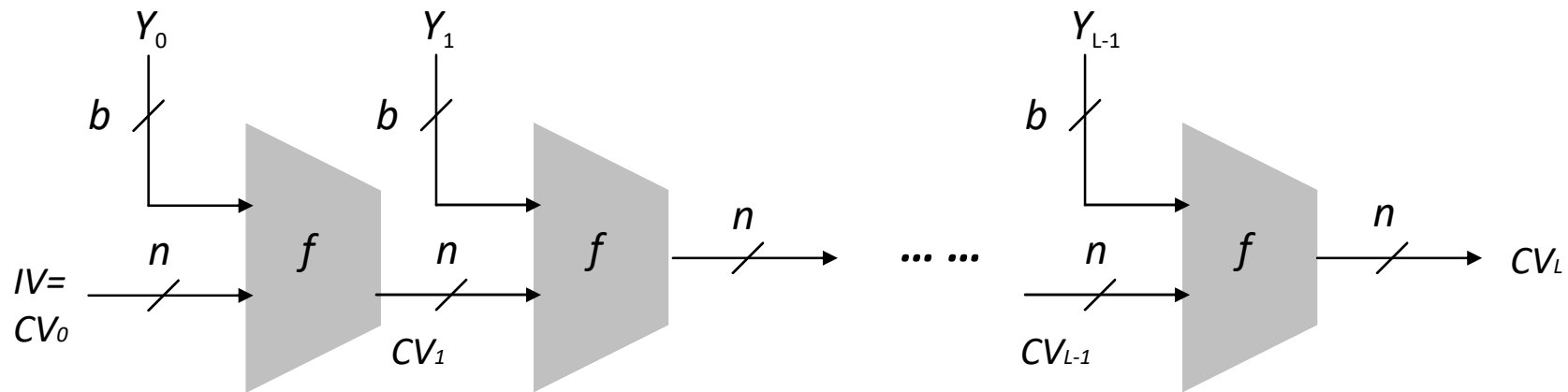
2.5.3 Hash

- Hash 函数通用模型
 - 由 *Ralph Merkle* 于1989年提出，得到广泛采用。
 - ✧ *R. Merkle* co-invented the *Merkle-Hellman* knapsack cryptosystem, invented cryptographic hashing, and invented *Merkle* trees.
 - 模型结构
 - ✧ 把原始消息 M 分成一些固定长度的块 Y_i
 - ✧ 最后一块 padding 并使其包含消息 M 的长度
 - ✧ 设定初始值 CV_0
 - ✧ 采用压缩函数 f , $CV_i = f(CV_{i-1}, Y_{i-1})$
 - ✧ 最后一个 CV_i 为 hash 值

2.5 MAC and Hashing Algorithms

2.5.3 Hash

- Hash 函数通用模型



- ✧ IV - initial value 初始值
- ✧ CV - chaining value 链接值
- ✧ Y_i - i th input block (第 i 个输入数据块)
- ✧ f - compression algorithm (压缩算法)
- ✧ n - length of hash code (散列码的长度, 128/160/256/512)
- ✧ b - length of input block (输入块的长度, 512/1024-SHA512)

2.5 MAC and Hashing Algorithms

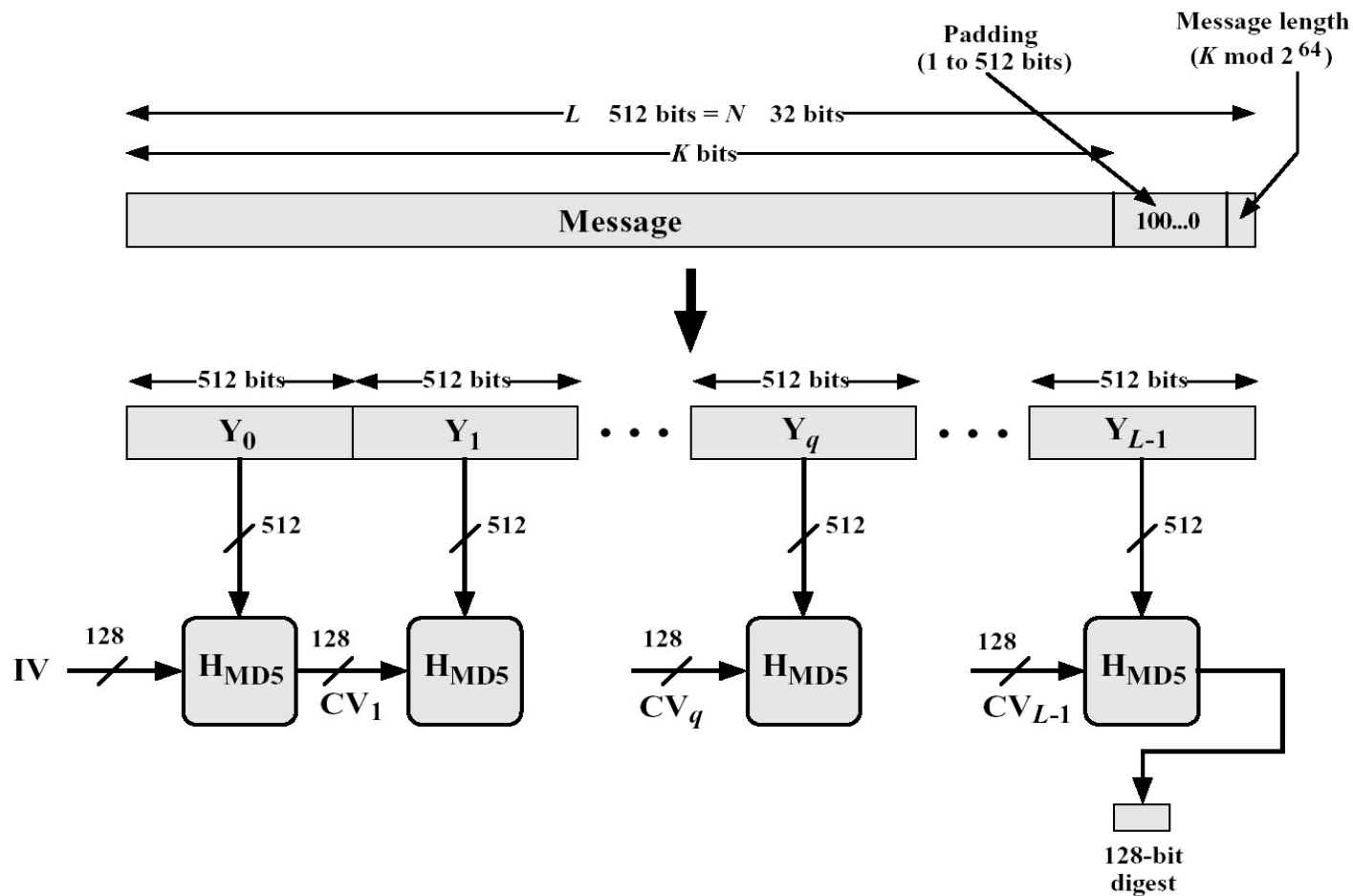
2.5.4 Message-Digest Algorithm (MD)

- 概述
 - MD5 即 Message-Digest Algorithm 5 (信息-摘要算法 5)
 - ✧ MD4 (1990)、MD5(1992, RFC 1321) 作者 *Ron Rivest*, 是广泛使用的散列算法, 经常用于确保信息传输的完整性和一致性。
 - ✧ MD5 使用 little-endian, 输入任意不定长度信息, 以512位长进行分组, 生成四个32位数据, 最后联合起来输出固定128位长的信息摘要。
 - ✧ MD5 算法的基本过程为: 求余、取余、调整长度、与链接变量进行循环运算、得出结果。
 - ✧ MD5 不是足够安全的
 - *Hans Dobbertin* 在1996年找到了两个不同的 512-bit 块, 它们在 MD5 计算下产生相同的 hash 值。
 - 至今还没有真正找到两个不同的消息, 它们的 MD5 的 hash 值相等。

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 的基本流程



2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑
 - 填充 padding
 - ✧ 在原始消息数据尾部填充标识 $100\dots0$ ，填充后的消息位数 $L \equiv 448 \pmod{512}$ 。至少要填充1个位，所以标识长度 $1\sim 512$ 位。
 - ✧ 再向上述填充好的消息尾部附加原始消息长度值的低64位，最后得到一个长度 L 是512位整数倍的消息。
 - 分块
 - ✧ 把填充后的消息结果分割为 L 个512位的分组： Y_0, Y_1, \dots, Y_{L-1} 。
 - ✧ 结果也表示成 N 个32位长的字 M_0, M_1, \dots, M_{N-1} ， $N = L \times 16$ 。
 - 初始化
 - ✧ 初始化一个128位的 MD 缓冲区，记为 CV_q ，也表示为4个32位寄存器 (A, B, C, D)； $CV_0 = IV$ 。迭代在 MD 缓冲区进行，最后一步的128位输出即为算法结果。

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑

- 初始化

- ✧ 寄存器 (A, B, C, D) 置16进制初值作为初始向量 IV，并采用小端存储 (little-endian) 的存储结构：

- A = 0x67452301

- B = 0xEFCDAB89

- C = 0x98BADCFE

- D = 0x10325476

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

- Little-Endian 将低位字节排放在内存的低地址端，高位字节排放在内存的高地址端。相反 Big-Endian 将高位字节排放在内存的低地址端，低位字节排放在内存的高地址端。存储结构与 CPU 体系结构有关。PowerPC 系列采用 big endian 方式存储数据，而 Intel x86 系列则采用 little endian 方式存储。

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑

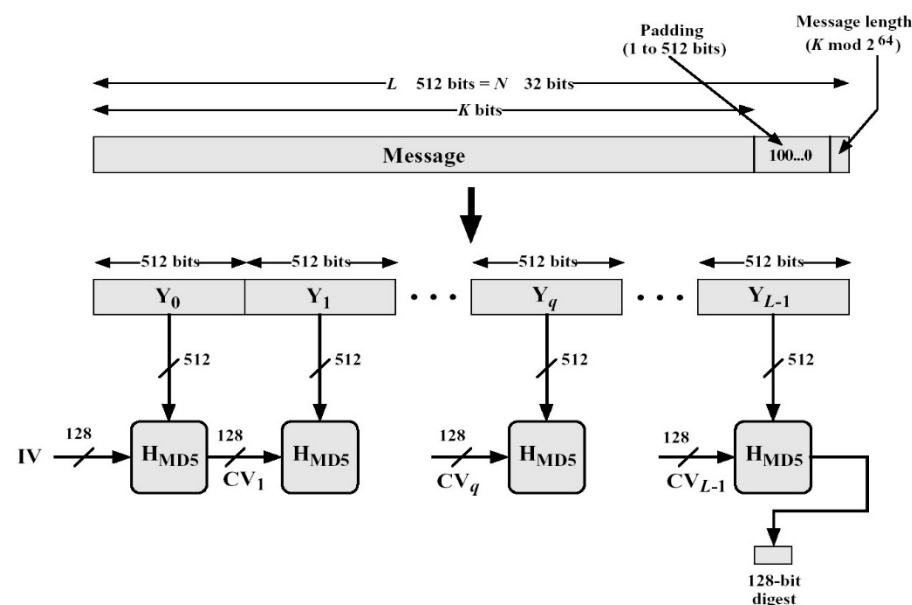
- 总控流程

- ✧ 以512位消息分组为单位，每一分组 Y_q ($q = 0, 1, \dots, L-1$) 经过4个循环的压缩算法，表示为：

$$CV_0 = IV$$

$$CV_i = H_{MD5}(CV_{i-1}, Y_i)$$

- ✧ 输出结果：MD = CV_L .



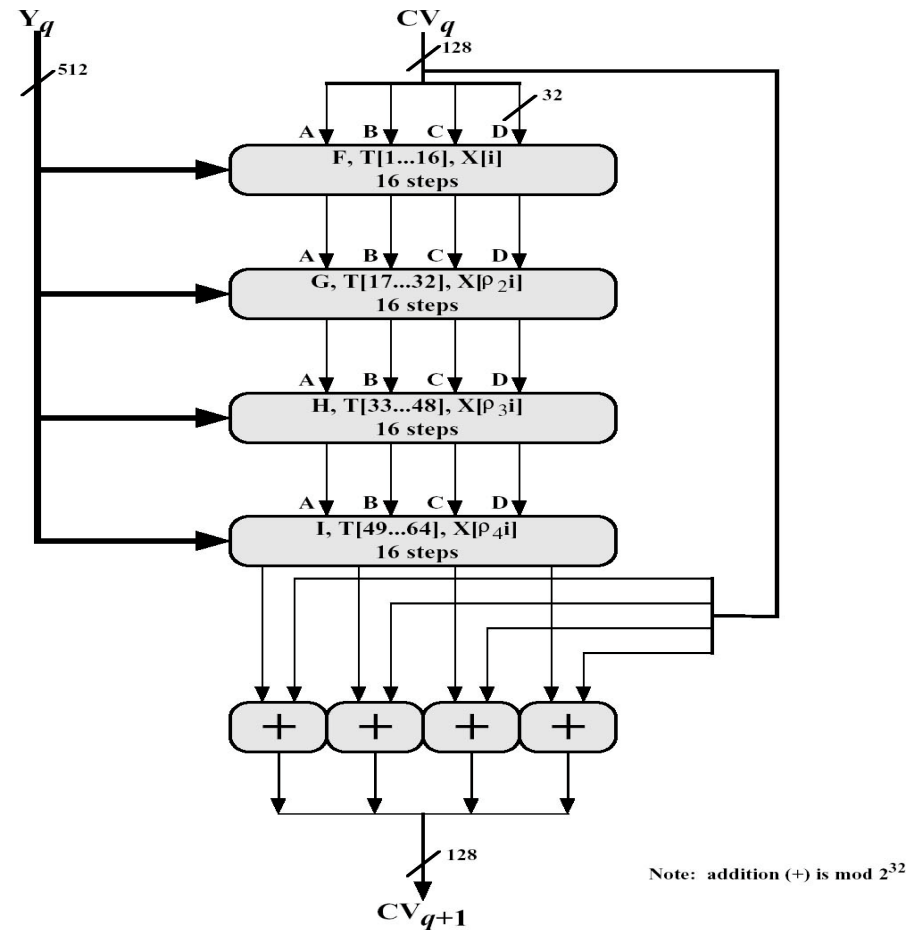
2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑

- MD5 压缩函数 H_{MD5}

- ✧ H_{MD5} 从 CV 输入128位，从消息分组输入512位，完成4轮循环后，输出128位，用于下一轮输入的 CV 值。
- ✧ 每轮循环分别固定不同的生成函数 F, G, H, I，结合指定的 T 表元素 T[] 和消息分组的不同部分 X[] 做 16 次运算，生成下一轮循环的输入。



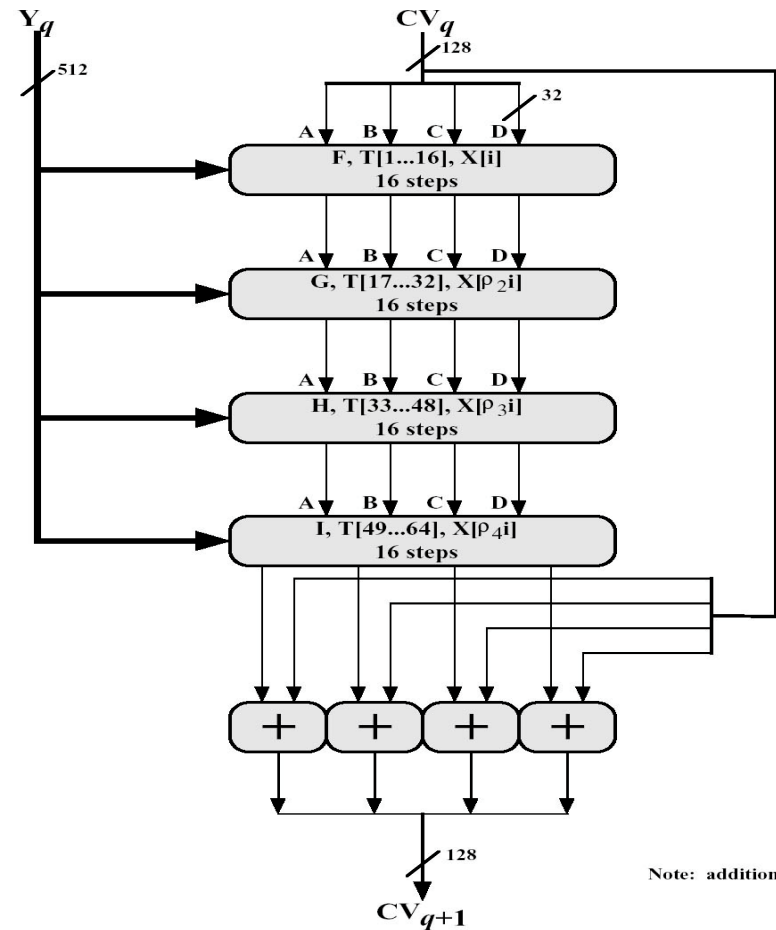
MD5 第 q 分组的4轮循环逻辑 (压缩函数)

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑
 - MD5 压缩函数 H_{MD5}
 - 4轮循环中使用的生成函数 (轮函数) g 的定义

轮次	Function g	$g(b, c, d)$
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	$I(b, c, d)$	$c \oplus (b \vee d)$



MD5 第 q 分组的4轮循环逻辑 (压缩函数)

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑

- MD5 压缩函数 H_{MD5}

- ✧ 每轮循环中的一步运算逻辑

$$a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i]) \lll s)$$

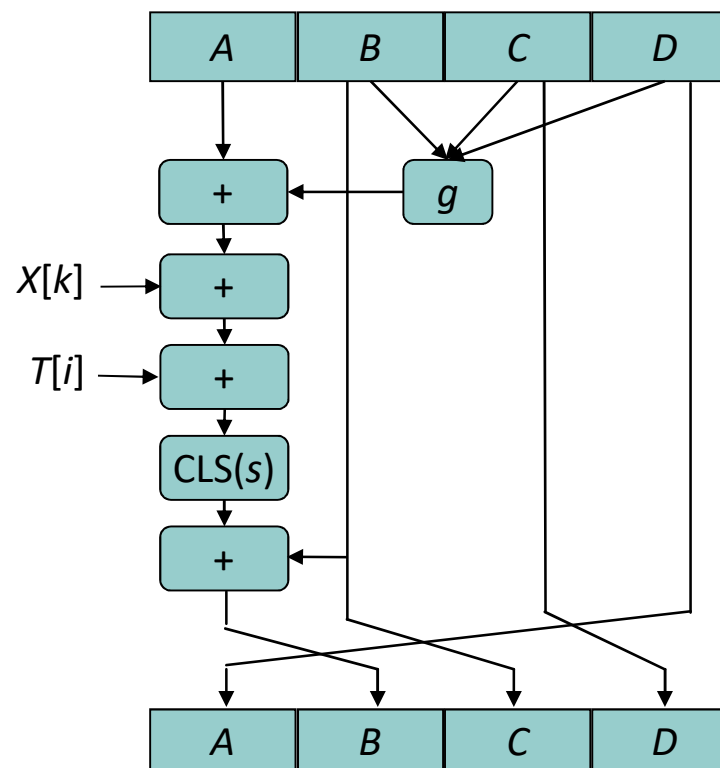
- ✧ 说明:

- a, b, c, d : MD 缓冲区 (A, B, C, D) 的当前值。
 - g : 轮函数 (F, G, H, I) 中的一个。
 - $\lll s$: 将32位输入循环左移 (CLS) s 位。
 - $X[k]$: 当前处理消息分组的第 k 个32位字, 即 $M_{q \times 16 + k}$ 。
 - $T[i]$: T 表的第 i 个元素, 32位字。
 - $+$: 模 2^{32} 加法。

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑
 - MD5 压缩函数 H_{MD5}
 - ✧ 每轮循环中的一步运算逻辑



MD5 每轮迭代中的一步运算逻辑

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑

- MD5 压缩函数 H_{MD5}

- ✧ 各轮迭代中 $X[k]$ 之间的关系:

- 第1轮迭代:

- $$X[j], j = 1..16.$$

- 第2轮迭代:

- $$X[\rho_2(j)], \rho_2(j) = (1 + 5j) \bmod 16, j = 1..16.$$

- 第3轮迭代:

- $$X[\rho_3(j)], \rho_3(j) = (5 + 3j) \bmod 16, j = 1..16.$$

- 第4轮迭代:

- $$X[\rho_4(j)], \rho_4(j) = 7j \bmod 16, j = 1..16.$$

- ✧ T 表的生成

- $T[i] = \text{int}(2^{32} \times |\sin(i)|)$

- int 取整函数, \sin 正弦函数, 以 i 作为弧度输入。

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- MD5 算法逻辑
 - MD5 压缩函数 H_{MD5}
 - ✧ 每次循环使用相同的迭代逻辑和 4×16 次运算的预设参数表。
 - ✧ Ref. to : RFC 1321

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- 几种散列函数的比较

	MD5	SHA-1	RIPEMD-160
摘要长度	128	160	160
分组长度	512	512	512
步数	64	80	160
最大消息长度	无限	$2^{64}-1$	$2^{64}-1$
基本逻辑函数	4	4	5
加法常数	64	4	9
Endianness	Little-endian	Big-endian	Little-endian
性能	32.4Mbps	14.1Mbps	13.6Mbps

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- SHA 的产生
 - 1992 NIST 制定 SHA (128位)
 - 1993 SHA 成为标准
 - 1994 修改产生 SHA-1 (160位)
 - 1995 SHA-1 成为新标准
 - ✧ SHA-1 要求输入消息长度 $< 2^{64}$
 - ✧ SHA-1 摘要长度为160位



2.5 MAC and Hashing Algorithms

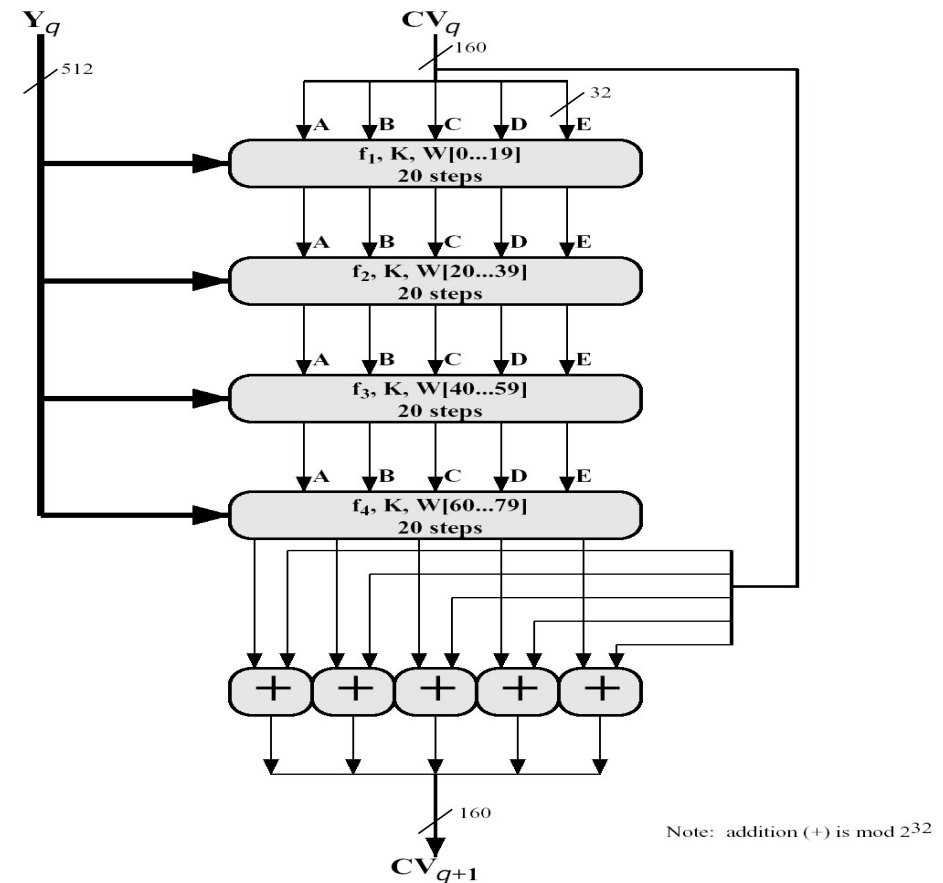
2.5.4 Message-Digest Algorithm (MD)

- SHA-1 的算法结构
 - SHA-1 的算法结构与 MD5 类似
 - ✧ 第一步: padding
 - 与 MD5 相同, 补齐到 512 的倍数
 - ✧ 第二步
 - 分块长度512位
 - ✧ 第三步
 - 初始化 MD buffer 为160位常量 (5个32位长度的字)
 - 进入循环, 160位输入+512位输入 \Rightarrow 160位输出
 - ✧ 第四步
 - 最后的输出为 SHA-1 的结果

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- SHA-1 的算法结构
 - 循环迭代

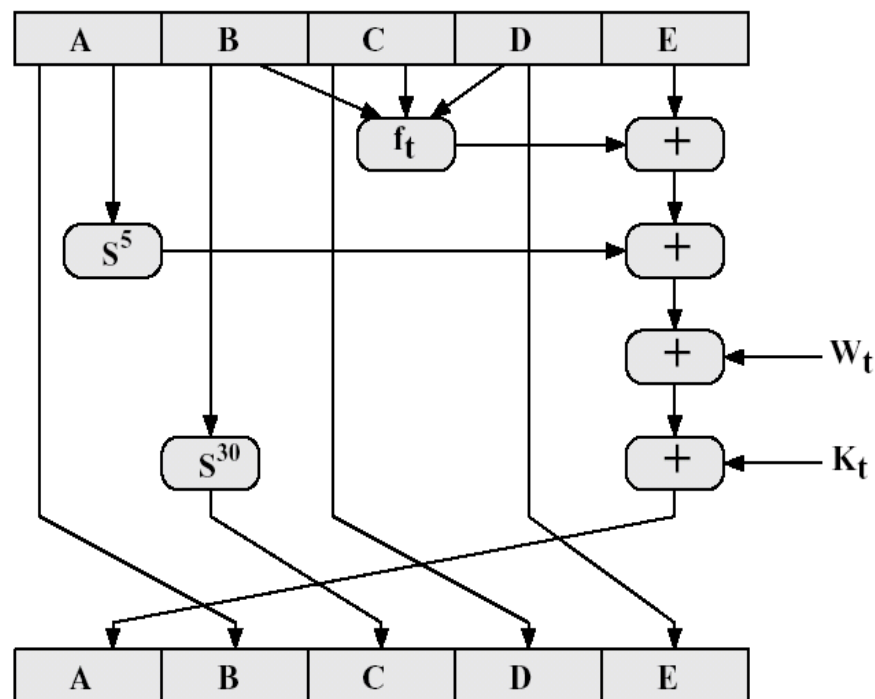


SHA-1 第 q 个分组的4轮循环逻辑 (压缩函数)

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- SHA-1 的算法结构
 - 循环迭代
 - ✧ W_t : 从消息块导出
 - ✧ K_t : 常量



SHA-1 每轮循环中的一步运算逻辑

2.5 MAC and Hashing Algorithms

2.5.4 Message-Digest Algorithm (MD)

- SHA-1 的一些讨论
 - 使用 big-endian
 - 抵抗生日攻击：160位 hash 值
 - 没有发现两个不同的 512-bit 块,它们在 SHA-1 计算下产生相同的 hash 值
 - 速度慢于 MD5
 - 安全性优于 MD5



Outline

- 2.1 Cryptology Introduction
- 2.2 Symmetric Key Cryptographic Algorithms
- 2.3 Mathematical Foundations of Public-Key Cryptography
- 2.4 Asymmetric Key Cryptographic Algorithms
- 2.5 MAC and Hashing Algorithms
- **2.6 Typical Applications**
 - MD5 and Passwords
 - AES and Wi-Fi Protected Access
 - RSA and e-Business

2.6 Typical Applications

2.6 Typical Applications

- Discussing
 - MD5 and Passwords
 - AES and Wi-Fi Protected Access
 - RSA and e-Business



References

1. William Stallings, Cryptography and Network Security: Principles and Practice (5th Edition), Prentice Hall 2010
2. [http://en.wikipedia.org/wiki/Birthday attack](http://en.wikipedia.org/wiki/Birthday_attack)
[Cryptography](http://en.wikipedia.org/wiki/Cryptography)
[Data Encryption Standard](http://en.wikipedia.org/wiki/Data_Encryption_Standard)
[Diffie–Hellman key exchange](http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange)
[Digital signature](http://en.wikipedia.org/wiki/Digital_signature)
[Hash function](http://en.wikipedia.org/wiki/Hash_function)
[Information_security](http://en.wikipedia.org/wiki/Information_security)
[MD5](http://en.wikipedia.org/wiki/MD5)
[Public-key cryptography](http://en.wikipedia.org/wiki/Public-key_cryptography)
[RSA](http://en.wikipedia.org/wiki/RSA)

End of Chapter 2

