第 3 章 消息（报文）认证

第 3 章 消息认证

- 3.1 概述
- 3.2 报文摘要MD
- 3.3 消息认证码MAC
- 3.4 数字签名DS

3.1 消息认证

- 密码技术：不仅能提供机密性服务，且能够保护数据的完整性和为数据来源的合法性提供认证服务
- 当消息、文件等是真实的而且来自合法来源，则其为可信的。
- 消息认证是一种允许通信者验证消息是否可信的措施。包括：
  - 验证消息内容是否被篡改
  - 验证消息来源是否可信、真实(且不可否认)
  - 验证消息的时效性
  - 验证消息流的相对顺序

消息认证

- 基本思路
  - 利用常规加密技术进行消息认证
  - 不依赖加密，消息内容不加密，生成认证标签附着在消息上

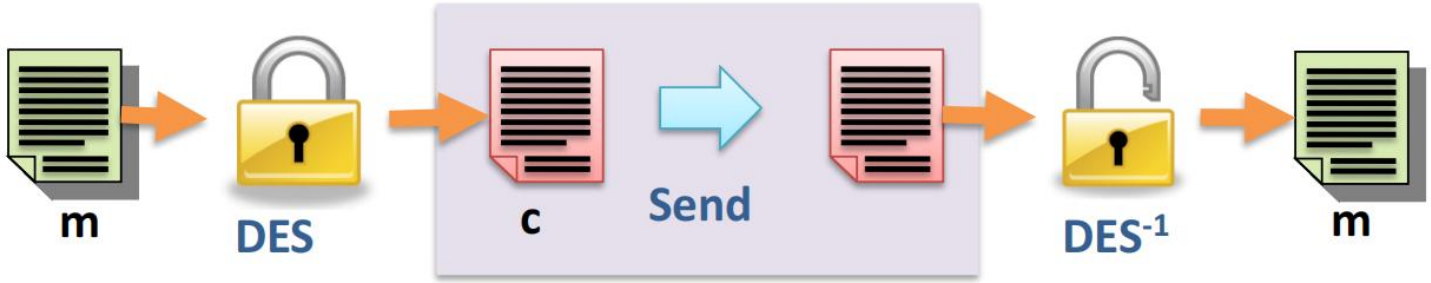
消息认证--基于网络传输技术

- 可靠传输协议：如TCP、HDLC等
  - 保证消息的顺序：消息序列号
  - 保证消息内容未被篡改：检错码
  - 保证消息的时效性：　时间戳
  - 保证消息来源真实性：源地址

上述内容都可以被伪造和修改

消息认证--基于常规加密技术

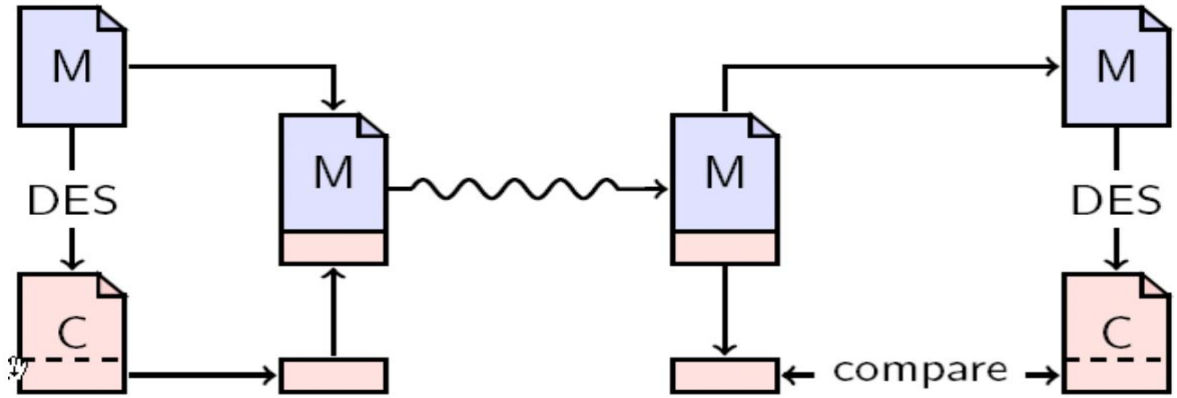
- 利用对称加密技术进行消息完整性保护



  - 收发双方共享密钥K
  - 保证消息的顺序和内容未被篡改：加密 检错码+序列号
  - 保证消息的时效性： 加密 时间戳
  - 保证消息来源真实性：加密 源地址
  - 保证消息的不可否认性：不能满足

消息认证--基于常规加密技术

- 有些应用场景不方便使用加密技术
  - 广播消息
  - 系统处理能力和计算能力有限
  - 计算机软件的完整性保护
- 举例：ECB-DES



能达到消息完整性保护的目标吗？

消息认证--基于常规加密技术

- 不希望使用加密技术进行认证的理由
  - 加密软件速度慢
  - 加密硬件成本不容忽视
  - 加密硬件的优化通常是针对大数据块的。对于小数据块，很多时间开销在初始化/调用上
  - 加密算法受专利保护，增加成本。
  - ……

消息认证

- 非加密的消息认证技术
  - 报文摘要MD=H(M)
  - 消息认证码MAC
  - 数字签名DS
- Hash(哈希)函数：也称散列、杂凑函数
  - 把任意长度的输入通过散列算法变换成固定长度的输出
  - 是一种压缩映射，任意长度的消息压缩到某一固定长度的消息摘要的函数

单向散列函数的两个特点

- (1) 散列函数的输入长度可以很长，但其输出长度则是固定的，并且较短。散列函数的输出叫做散列值，或更简单些，称为**散列。**

- (2) 不同的散列值肯定对应于不同的输入，但不同的输入却可能得出相同的散列值。这就是说，散列函数的输入和输出并非一一对应的，而是**多对一**的。

简单散列函数

- 最简单方案：纵向冗余校验
- 算法的基本操作过程
  - 输入：n比特块的序列
  - 操作
    - 每次处理一个输入数据块，得到n比特的散列值
    - 将各数据块的散列值一次异或操作，得到最终的散列值

$$C_i=b_{i1}\oplus b_{i2}\oplus \ldots \oplus b_{im}$$

  - 输出：n比特的散列值

能达到消息完整性保护的目标吗？
你有方法进行攻击吗？

简单散列函数

- 安全性：原始消息不加密，攻击者很容易生成新的具有相同散列码的消息

  将M'||N作为消息发出，其中N=C'⊕ C

  H(M)=C, H(M')=C', H(N)=N

- 改进：将原始消息与散列码一起加密使用
- 举例：美国NIST提出的方案
  - 散列码=64bit消息块异或
  - CBC方式加密消息与散列码M||C
- 安全性：密文分组置换位置散列码不变

3.2 报文摘要MD

- 报文摘要也称为消息指纹
- 在密码学中使用的散列函数称为**密码散列函数**(cryptographic hash function)是一种相对简单的对报文进行鉴别的方法。

密码散列函数

- 为满足消息认证中的应用，散列函数H须具有下列性质：
  - 1：输入长度可变：H可适用于任意长度的数据块
  - 2：输出长度固定：H能生成固定长度的输出
  - 3：效率：对任意给定的M，计算H(M)相对容易，用软/硬件均可实现
  - 4：抗原像攻击(单向性)：对于任意给定值h，找到满足H(M)=h的M在计算上不可行。
  - 5：抗第二原像攻击(抗弱碰撞性)：对于任意给定的数据块M，找到满足H(M')=H(M)的M'≠M在计算上不可行。
  - 6：抗碰撞攻击(抗强碰撞性)：找到满足H(M')=H(M)的任意一对(M',M)在计算上不可行。满足这一特性的散列函数称为，也称为。
  - 7: 伪随机性：H的输出满足伪随机性测试标准

密码散列函数

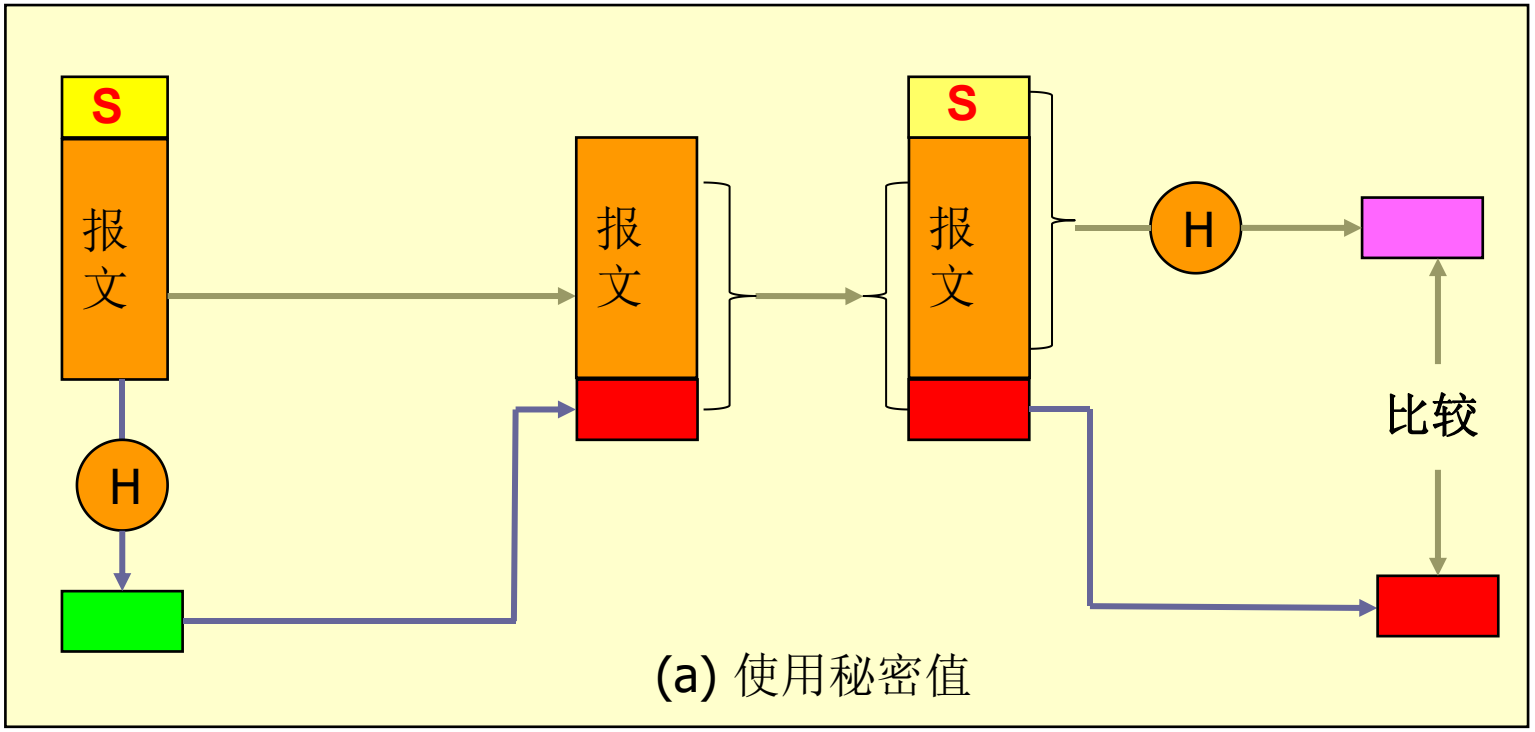
- 性质4：单向性（抗原像攻击）。
  - 给定M，很容易计算h h=H(M) ，其中，h称为M的像(image)，M是H(M)的原像(Preimage)
  - 给定h, 根据H(M)=h计算M很难

  - 应用场景

密码散列函数

- 性质5：抗第二原像攻击。
  - 给定M和h $h=H(M)$
  - 找到另一个输入M’，满足H(M’)=H(M)，计算上很难
  - 思考：应用场景？
- 性质6：抗碰撞
  - 对于给定的H函数
  - 找到任意两个输入M和N,满足H(N)=H(M)，计算上很难
  - 思考：应用场景？

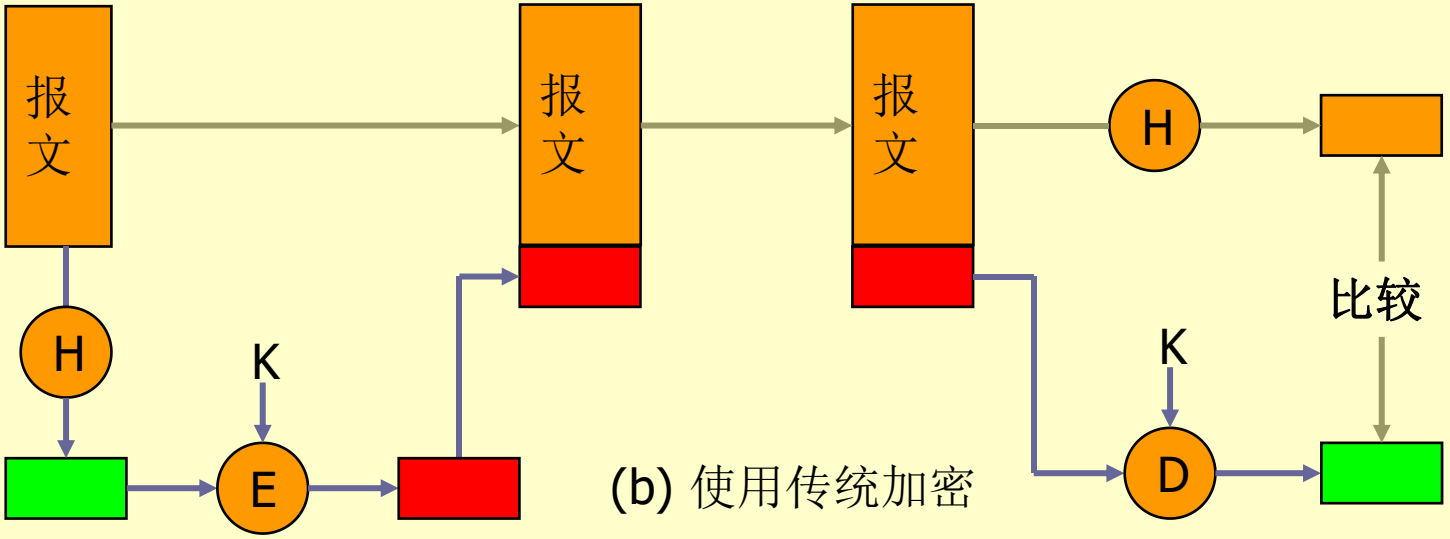
应用Hash函数进行报文鉴别



(a) 使用秘密值

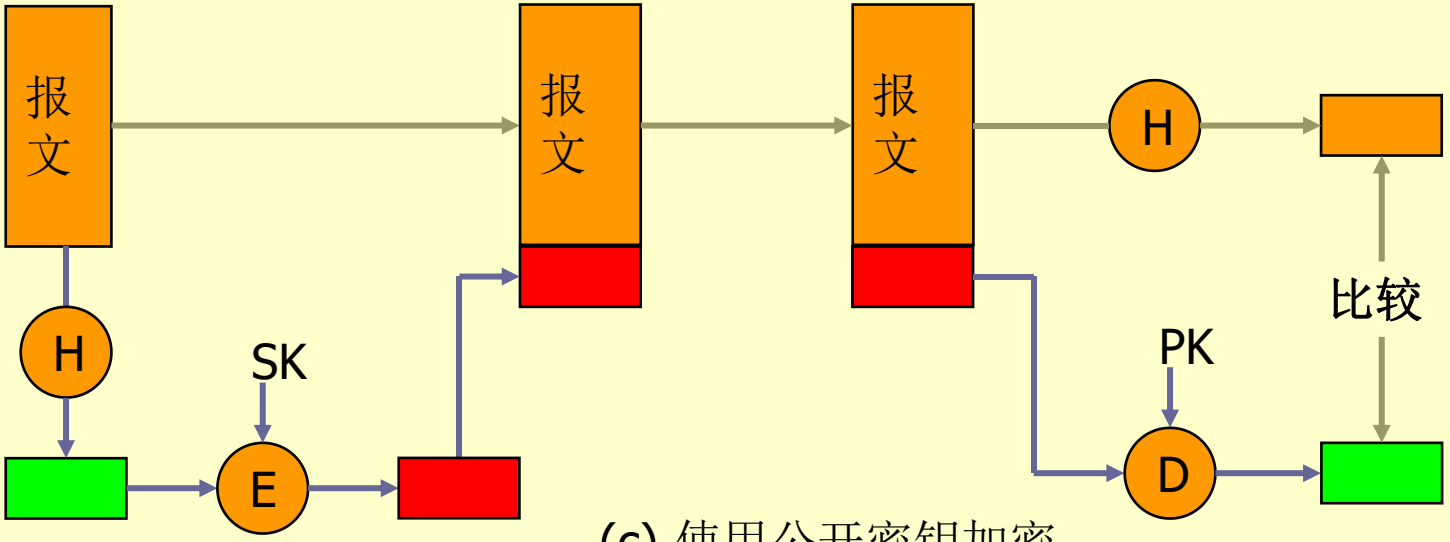
H(M||S) || M

使用Hash函数的报文鉴别



(b) 使用传统加密

$E_K(H(M)) \| M$



(c) 使用公开密钥加密

$E_{SK}(H(M)) \| M$

常用的Hash函数

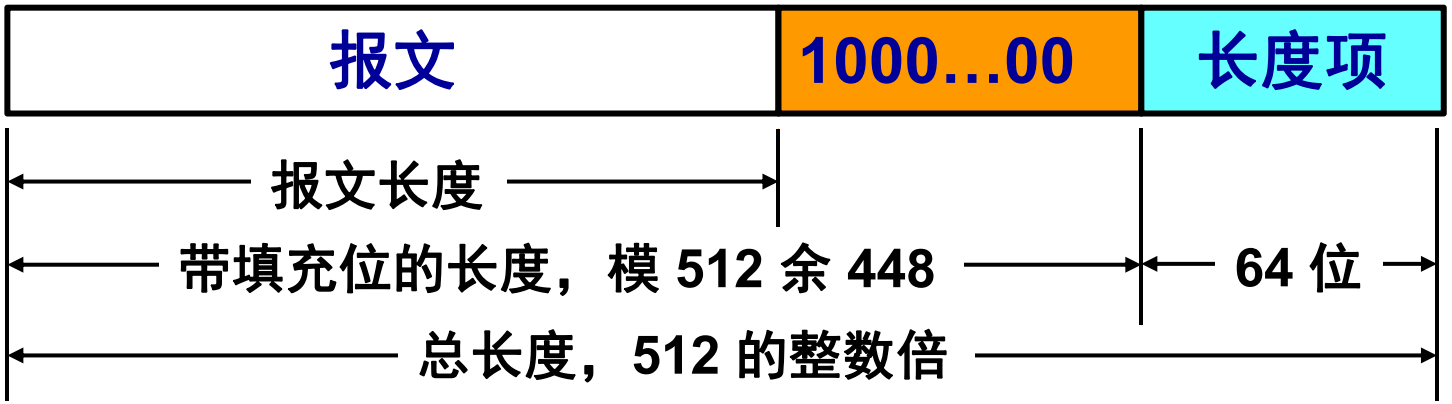
- 基于分组密码算法的Hash函数
- 系列Hash函数，MD2,MD4和MD5(均为128bit的输出)
- 美国政府的安全Hash标准（SHA-1）,产生160bit输出，与DSA算法匹配使用

MD5 算法

- MD5是报文摘要 MD (Message Digest) 的第5个版本。报文摘要算法MD5公布于RFC 1321 (1991年)，并获得了非常广泛的应用。
- MD5 的设计者 Rivest曾提出一个猜想，即根据给定的 MD5 报文摘要代码，要找出一个与原来报文有相同报文摘要的另一报文，其难度在计算上几乎是不可能的。
- 基本思想：
  - 用足够复杂的方法将报文的数据位充分“弄乱”，报文摘要代码中的每一位都与原来报文中的每一位有关。

MD5 算法

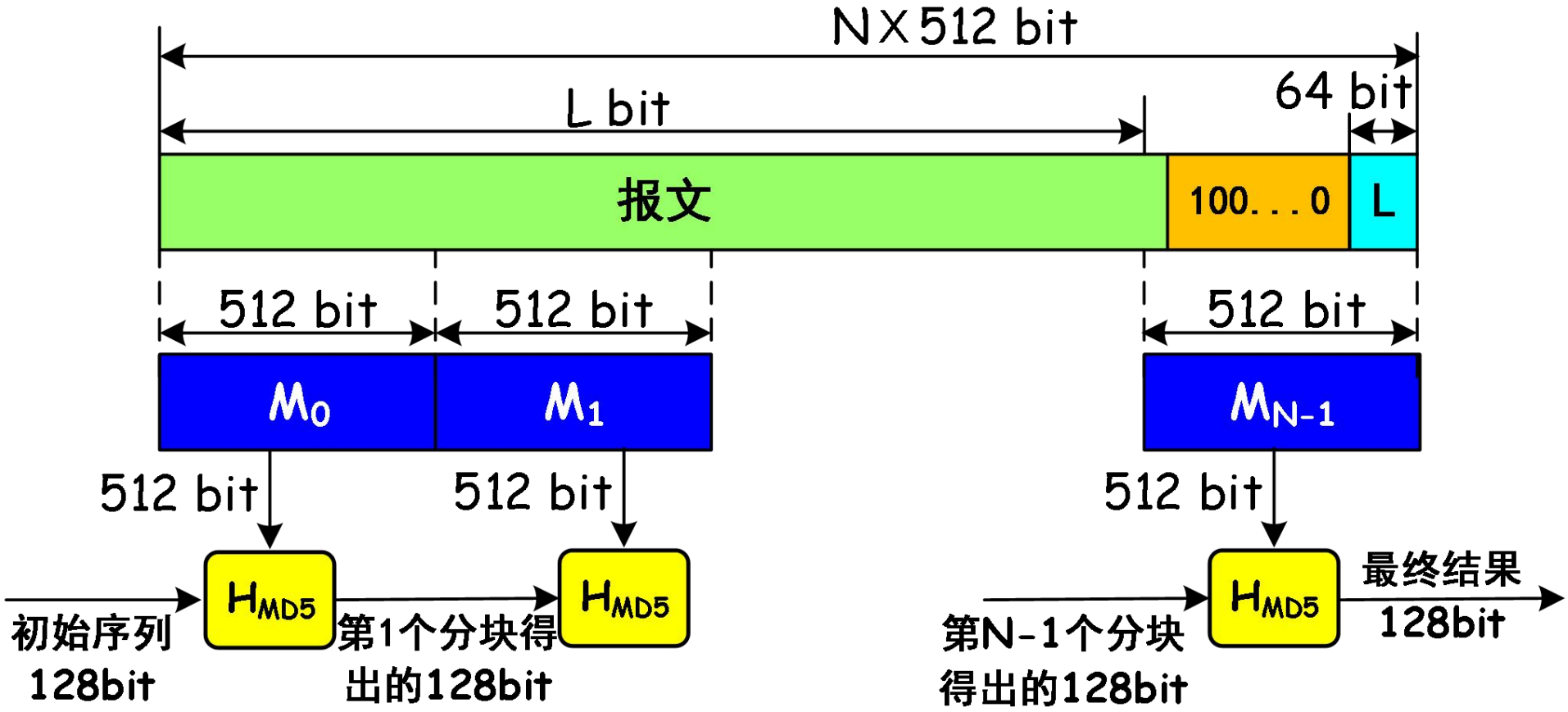
- 计算步骤：
  - 1，附加：把任意长的报文按模 2^{64} 计算其余数（64位），追加在报文的后面（长度项）。
  - 2，填充：在报文和长度项之间填充 1~512 位，使得填充后的总长度是 512 的整数倍。填充的首位是1，后面都是 0。



MD5 算法

- 计算步骤（续）：
  - 3，分组：把追加和填充后的报文分割为一个个 512 位的数据块，每个 512 位的报文数据再分成 4 个 128 位的数据块
  - 4，计算：将 4 个 128 位的数据块依次送到不同的散列函数进行4轮计算。每一轮又都按 32 位的小数据块进行复杂的运算。一直到最后计算出 MD5 报文摘要代码（128位）。

MD5算法


N X 512 bit
L bit
64 bit
报文
100. . . 0
L
512 bit
512 bit
512 bit
M0
M1
MN-1
512 bit
512 bit
512 bit
HMD5
HMD5
HMD5
初始序列
128bit
第1个分块得
出的128bit
第N-1个分块
得出的128bit
最终结果
128bit

安全散列算法（SHA）

- 安全散列算法 SHA (Secure Hash Algorithm) 是由美国标准与技术协会 NIST 提出一个散列算法系列。
- 其他每种被广泛使用的散列函数都已经被证实存在着密码分析学中的缺陷，SHA或许是最后仅存的标准散列函数
- SHA 比 MD5 更安全，但计算起来却比 MD5 要慢些。
- 已制定 SHA-1（2010年后不再推荐使用）、SHA-2等版本。

安全散列算法（SHA-1）

- 基本思想：
  - 要求输入码长小于 2^{64} 位，输出码长为 160 位。
  - 将明文分成若干 512 位的定长块，每一块与当前的报文摘要值结合，产生报文摘要的下一个中间结果，直到处理完毕
  - 共扫描 5 遍，效率略低于 MD5，抗穷举性更高。

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
消息摘要大小	160	224	256	384	512
消息大小	$<2^{64}$	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
块大小	512	512	512	1024	1024
字大小	32	32	32	64	64
步骤数	80	64	64	80	80

SHA-1

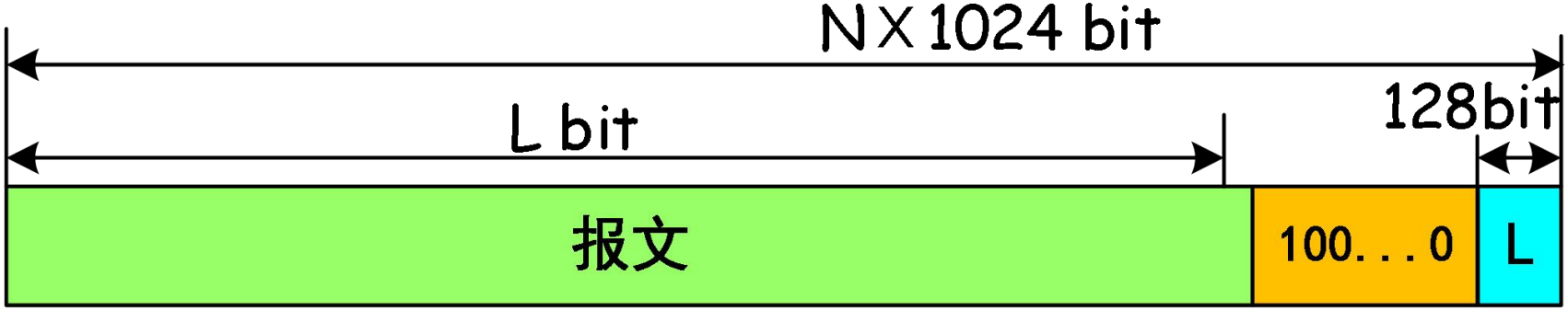
- 基本处理过程
  - 填充
  - 追加长度
  - 初始化散列缓冲区(5个32bit的word寄存器)
  - 处理512bit的数据块
  - 输出：160bit

SHA-512

- 基本处理过程
  - 填充
  - 追加长度
  - 初始化散列缓冲区(8个64bit的word寄存器)
  - 处理1024bit的数据块
  - 输出：512bit

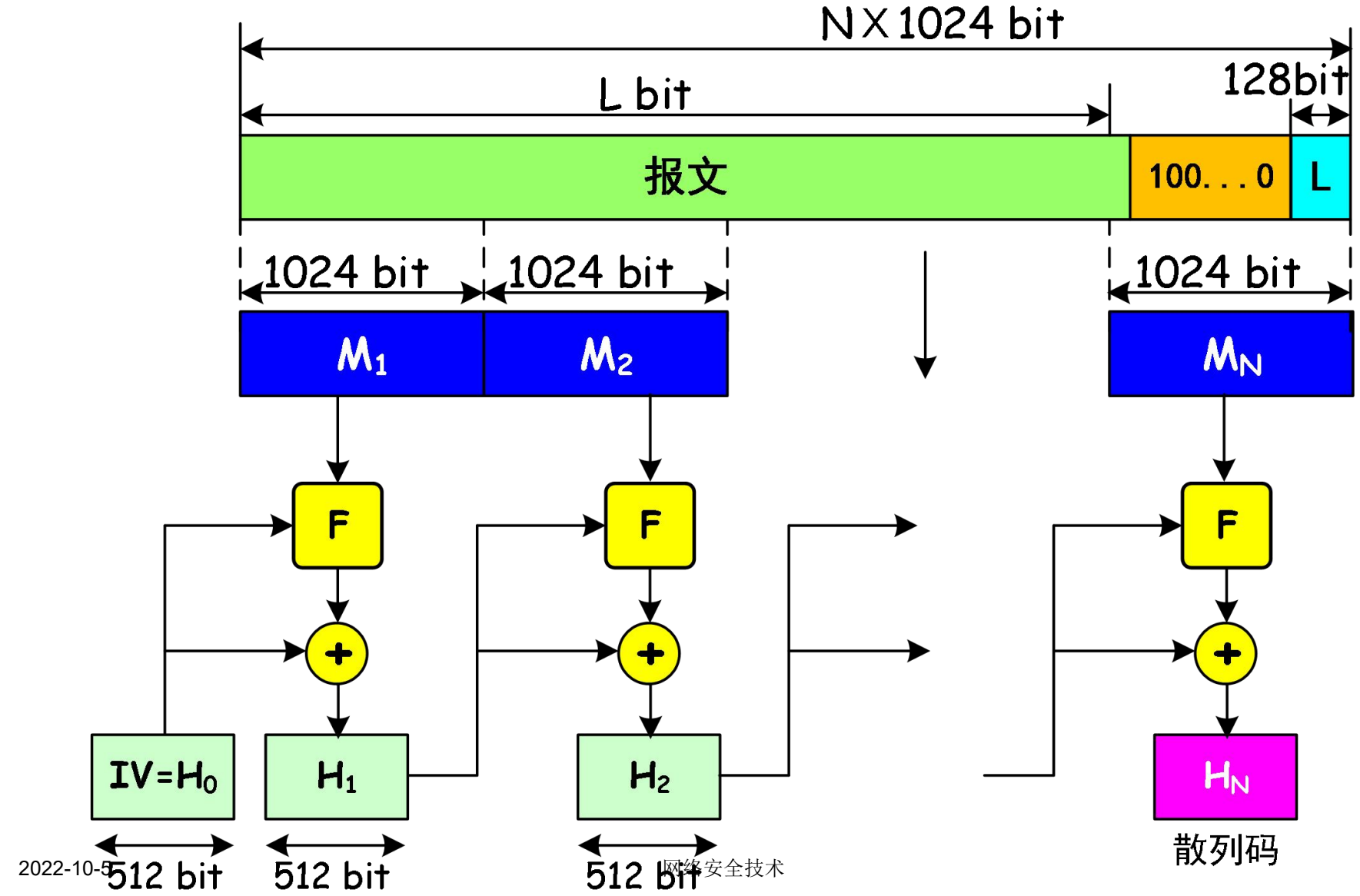
SHA-512

- 计算步骤
  - 1. 填充：填充消息使其长度模1024同余896。说明：即使消息已经是期望的长度，也需要添加填充，因此，填充比特的长度范围是1~1024.
  - 2. 追加长度：将128比特的长度数据块追加在消息上。



  - 3. 初始化散列缓冲区：用512比特的缓冲区保存散列函数中间和最终结果。缓冲区是8个64比特的寄存器(a,b,c,d,e,f,g,h)。

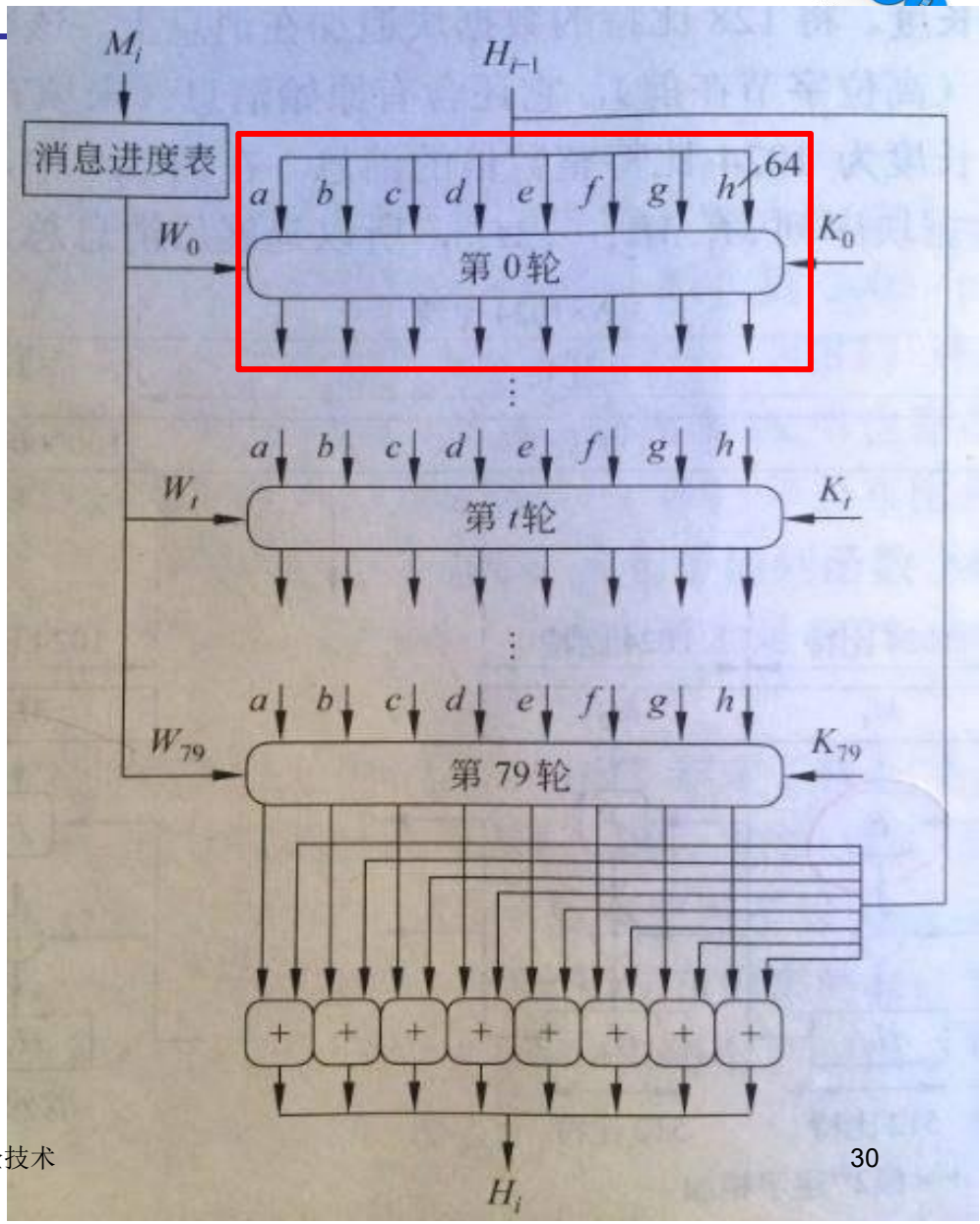
SHA-512


N X 1024 bit
L bit
128bit
报文
100. . . 0
L
1024 bit
1024 bit
1024 bit
M1
M2
MN
F
F
F
+
+
+
IV=H0
H1
H2
HN
512 bit
512 bit
512 bit
散列码

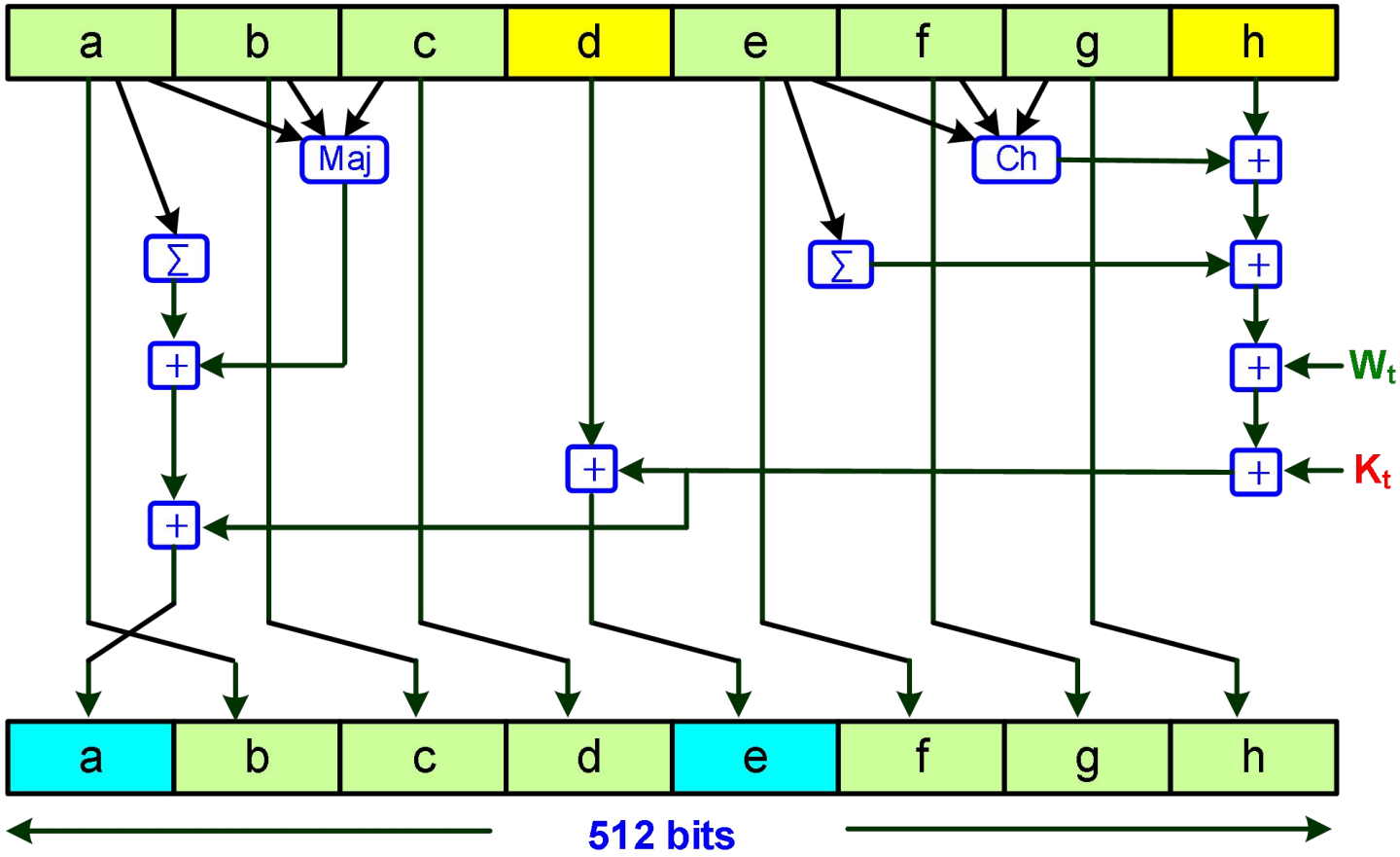
SHA-512

- **4. 处理1024比特的数据块消息：算法的核心是80轮迭代构成的F函数，F函数的复杂迭代产生很好的混淆效果，Ki为轮常数**

- **5. 输出：所有N个1024比特的数据块处理完毕后输出的便是512比特的消息摘要。**

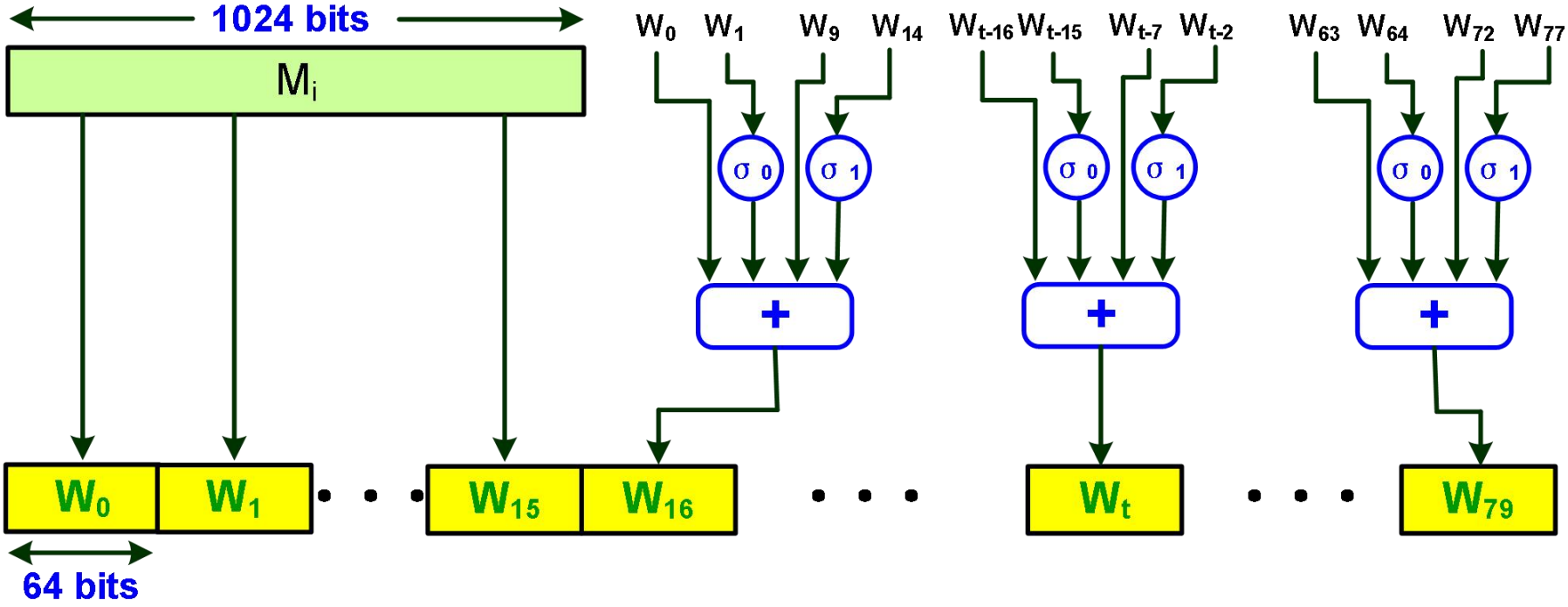


SHA-512



Maj(a,b,c)=(a^b) XOR (a^c) XOR (b^c) Ch(e,f,g)=(e^f) XOR (¬e^g)

SHA-512---字Wt的产生



$$W_t=\sigma_1(W_{t-2})+W_{t-7}+\sigma 0(W_{t-15})+W_{t-16}$$

$$\sigma_1(x)=ROTR^1(x)+ROTR^8(x)+SHR^7(x)$$

$$\sigma_0(x)=ROTR^{19}(x)+ROTR^{61}(x)+SHR^6(x)$$

+为模2^{64}加

$ROTR^n(x)$为对64位的变量x循环右移n位

$SHR^n(x)$为对64位变量向右移n位，左边填充0

SHA-512----轮常数Ki

表 11.4 SHA-512 轮常数

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

散列函数的安全性

- 攻击安全散列函数的方法
  - 密码分析法：利用算法在逻辑上的缺陷
  - 暴力攻击法
- 散列函数抵抗暴力攻击的强度完全依赖于算法生成的散列码长度。攻击一个长度为n的散列码所需要付出的代价如下：

抗原像	2^{n-1}
抗第二原像	2^{n-1}
抗碰撞	$2^{n/2}$

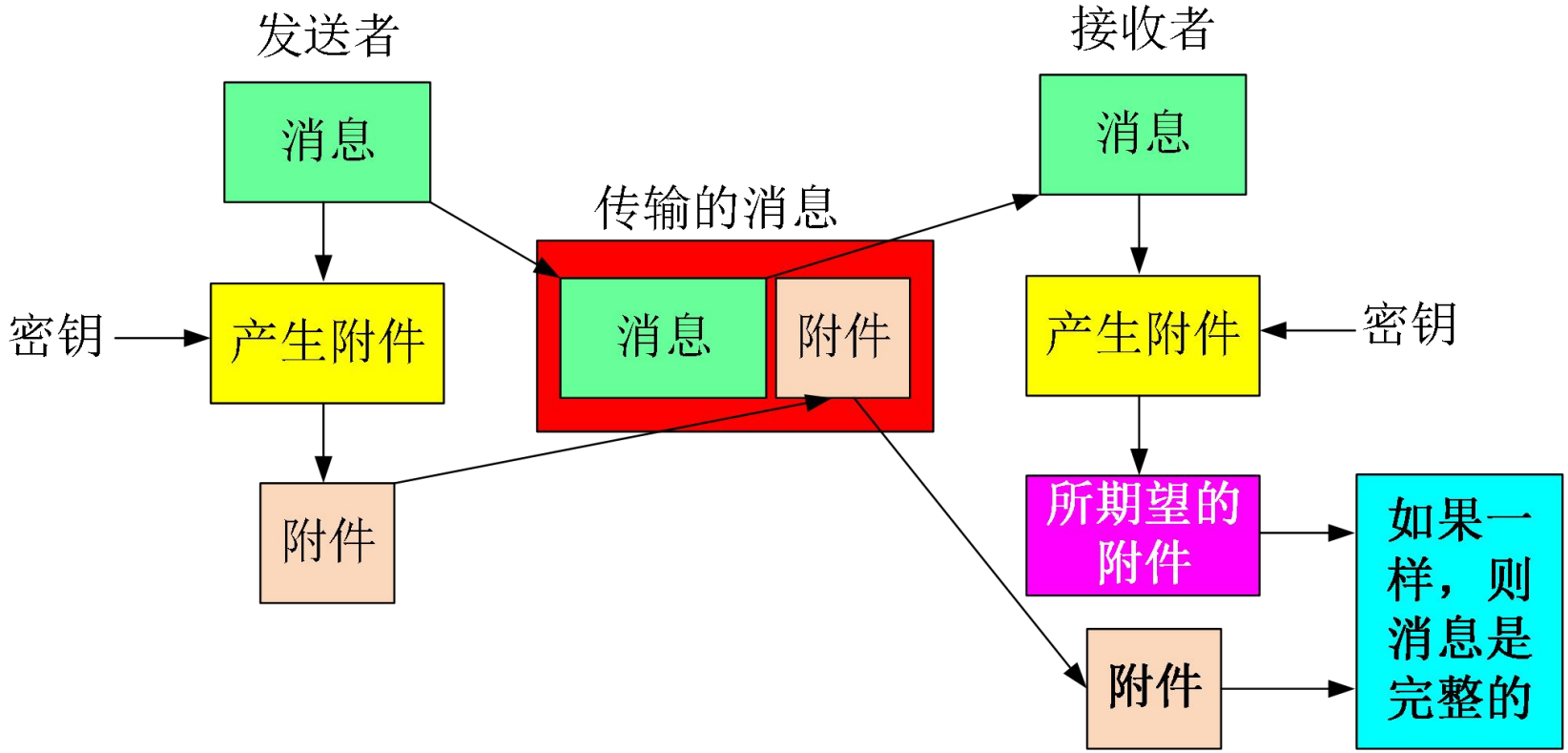
3.3 消息认证码MAC

- 报文摘要可以用于检查报文的完整性（即是否被篡改），但是不能用来实现对报文来源的确认。
- 消息认证码MAC (Message Authentication Code)是一种认证技术，它利用消息和双方共享的密钥通过认证函数来生成一个固定长度的短数据块，并将该数据块附加在消息后。MAC不仅可以对消息进行完整性保护，还可以对消息来源进行认证。

MAC安全功能

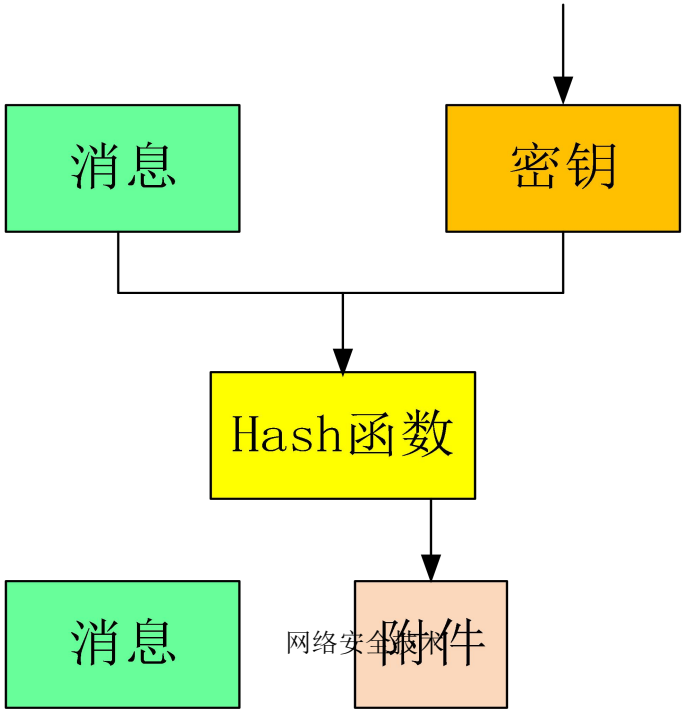
- **完整性保护**：接收方可以相信消息未被修改。虽然攻击者能够改变消息，但不能伪造对应的MAC值，因为攻击者不知道只有发送方和接受方知道的密钥。
- **源认证**：接受方可以确定消息来自真正的发送法，因为除了发送方和接受方外没有第三方知道密钥，因此第三方不可能产生正确的消息和MAC值。

MAC认证封装机制

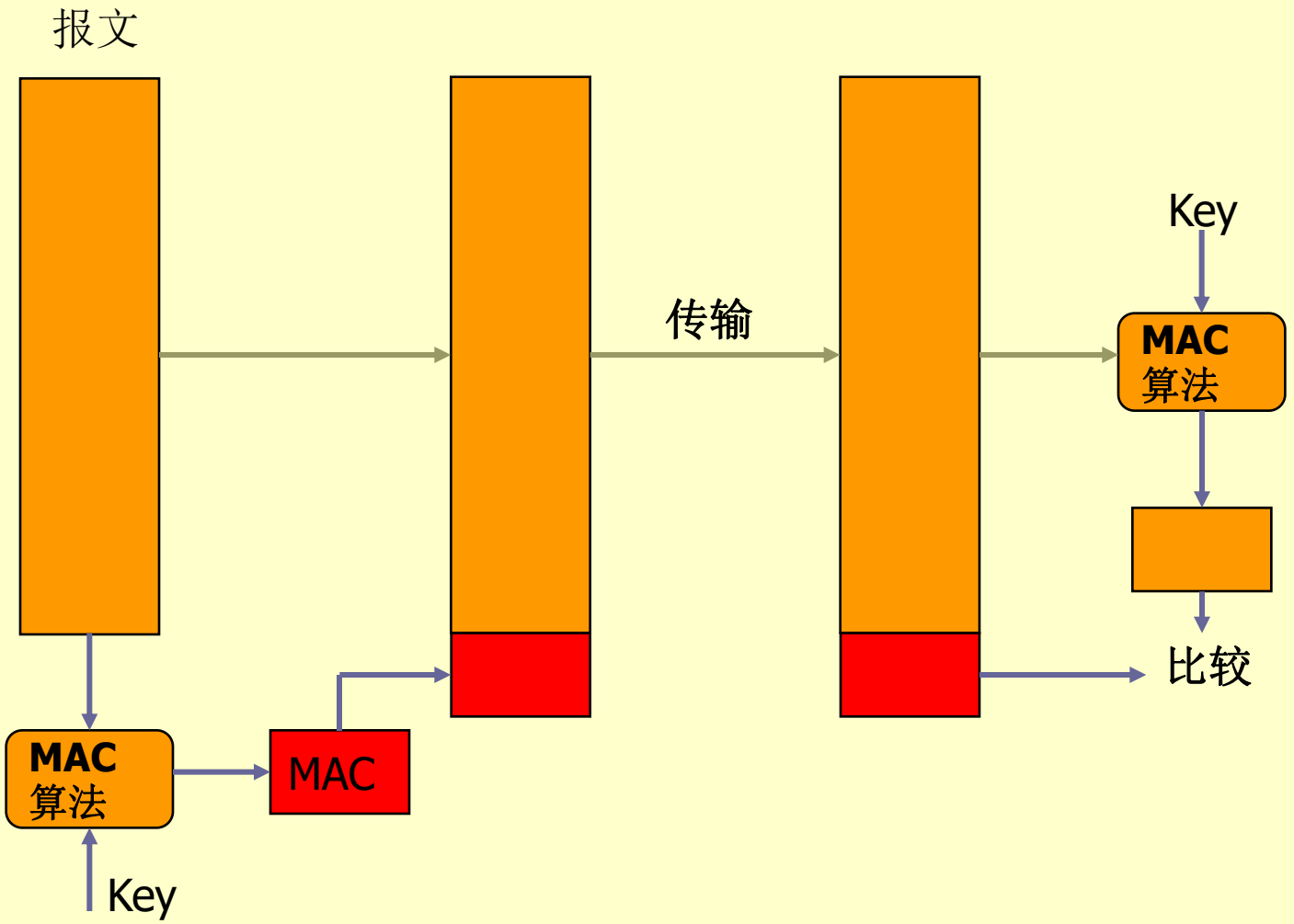

发送者
消息
密钥
产生附件件
附件
传输的消息
消息
附件
接收者
消息
产生附件件
密钥
所期望的附件
附件
如果一样，则消息是完整的

消息认证码(MAC)的产生

- 银行工业标准MAC的产生：**基于对称密码体制**(如DES),输入密钥，使用**CBC加密模式**产生附件
- 使用Hash函数产生: Hash函数是将任意长度的输入串变化成固定长度的输出串的一种函数



利用报文鉴别码MAC进行报文鉴别



HMAC

- HMAC是一个从密码Hash衍生出来的消息认证码，即在现有的哈希算法中加入了一个密钥。
- 原因
  - 比传统加密算法执行得更快
  - 有许多共享的密码Hash函数代码库
  - 多数国家没有出口限制

HMAC

- 设计目标（RFC2104）--非常重要
  - 不必修改而直接使用现有的Hash函数
  - 如果找到或需要更快/更安全的Hash函数，应能很容易地替代原来嵌入的Hash函数
  - 应保持Hash函数的原有性能，不能过分降低其性能
  - 对密钥的使用和处理应简单
  - 如果已知嵌入的Hash函数的强度，则完全可以知道认证机制抗密码分析的强度

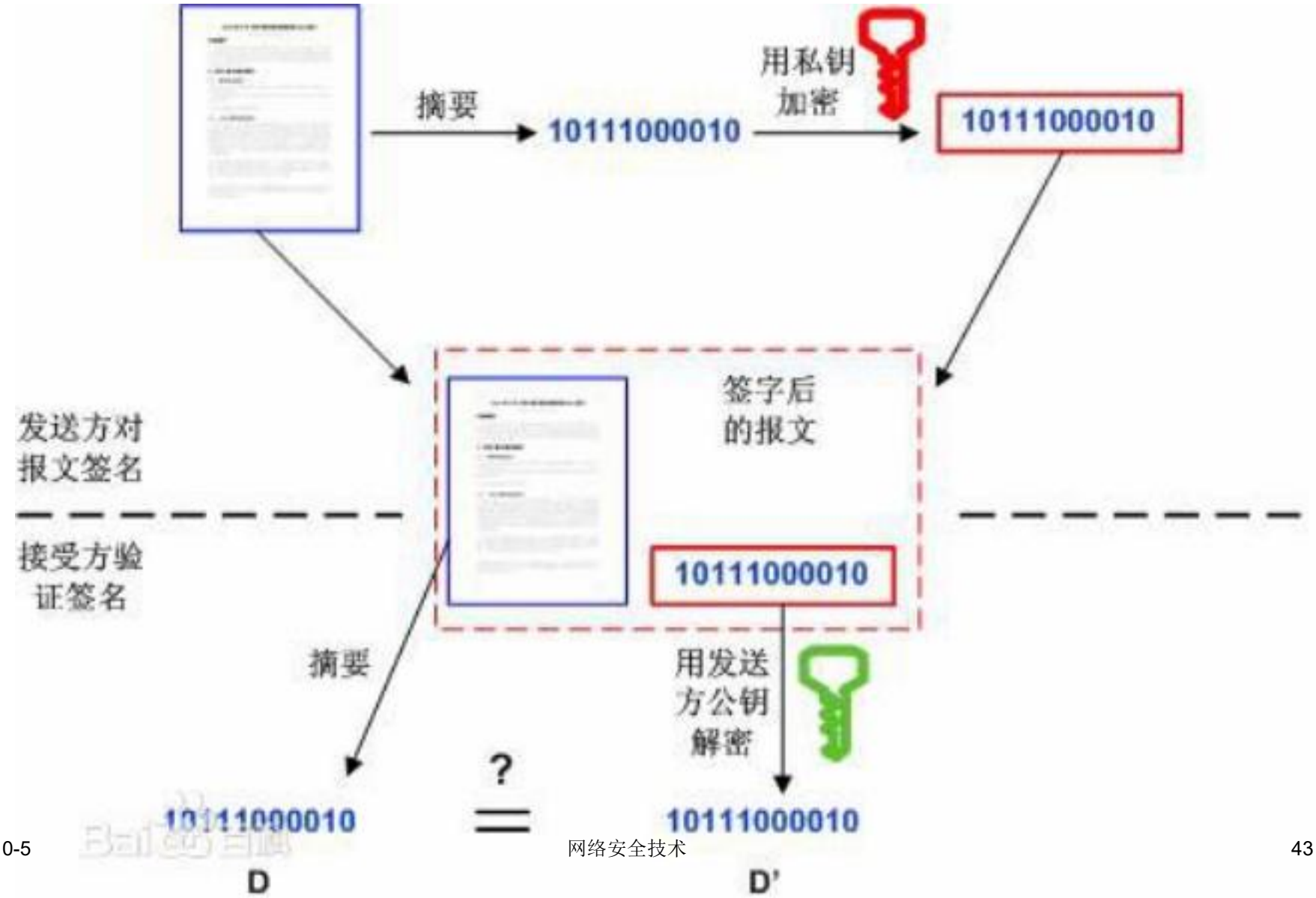
将Hash函数视为“黑盒”

- HMAC算法（具体参见教材图12.5 HMAC结构）

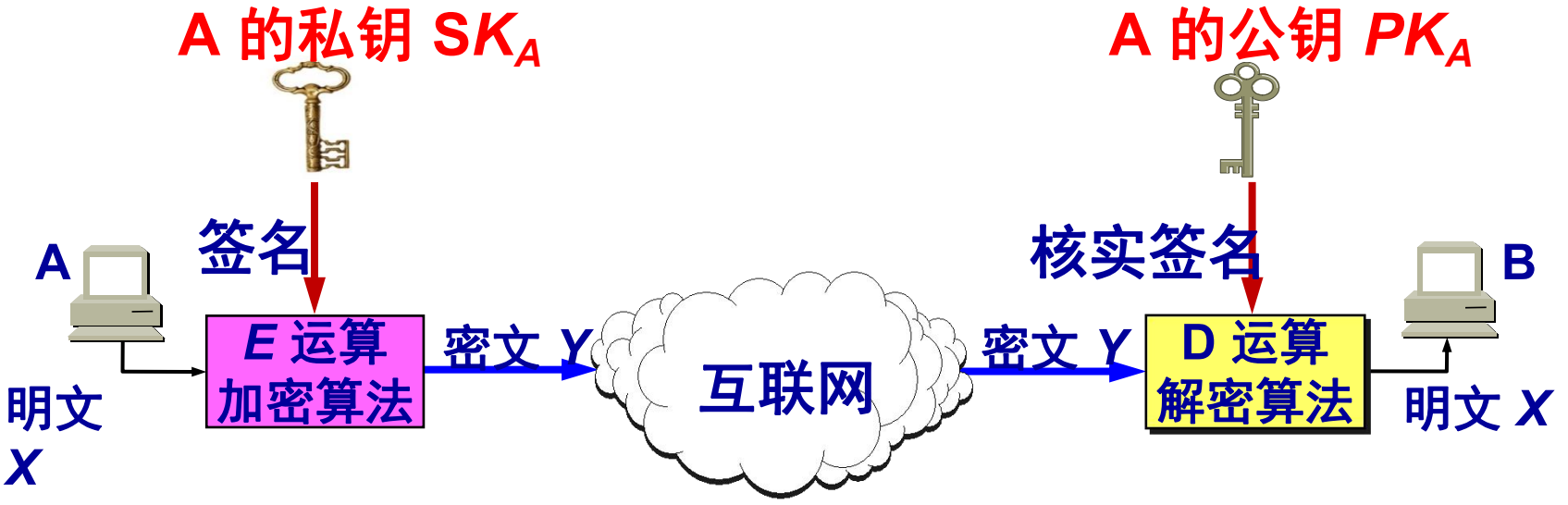
3.4 数字签名技术

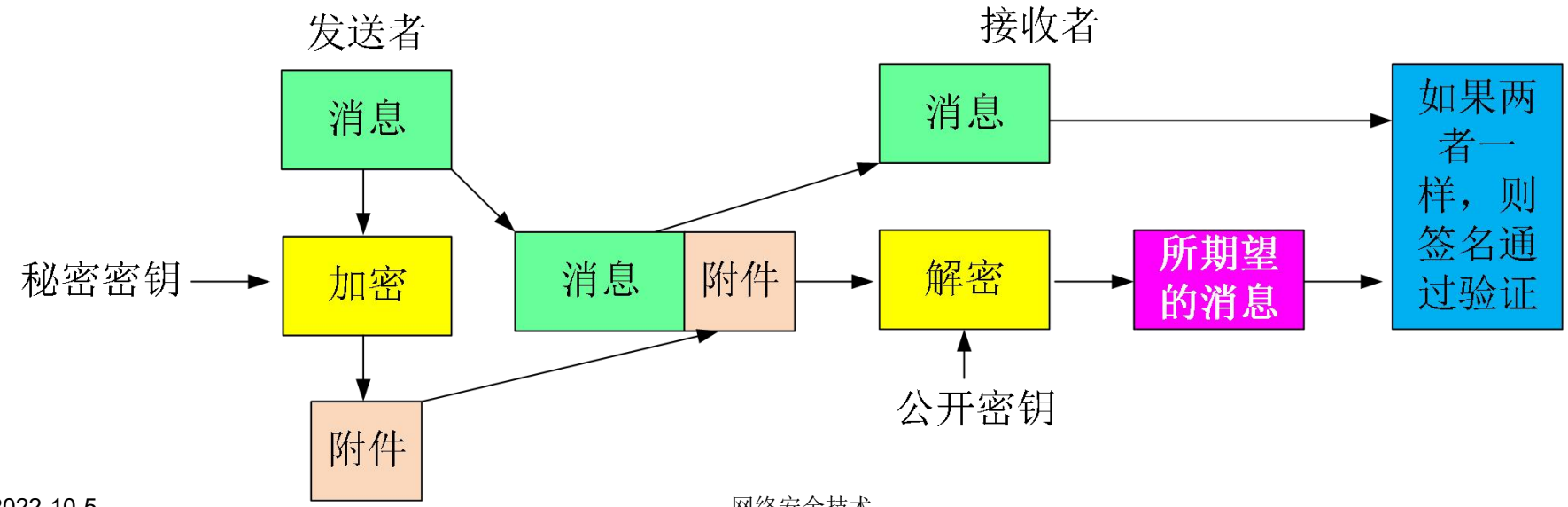
- 数字签名是另一种提供报文完整性和源鉴别的技术
- 数字签名必须保证以下三点：
  - (1) 报文源鉴别——接收者能够核实发送者对报文的签名（证明来源）；
  - (2) 报文的完整性——接收者不能伪造对报文的签名（防伪造）；
  - (3) 不可否认——发送者事后不能抵赖对报文的签名（防否认）。
- 现在已有多种实现各种数字签名的方法。但采用公钥算法更容易实现。

数字签名工作原理


摘要
10111000010
用私钥
加密
10111000010
签字后
的报文
发送方对
报文签名
接受方验
证签名
10111000010
摘要
用发送
方公钥
解密
?
10111000010
10111000010
D
D'

基于公钥的数字签名的实现



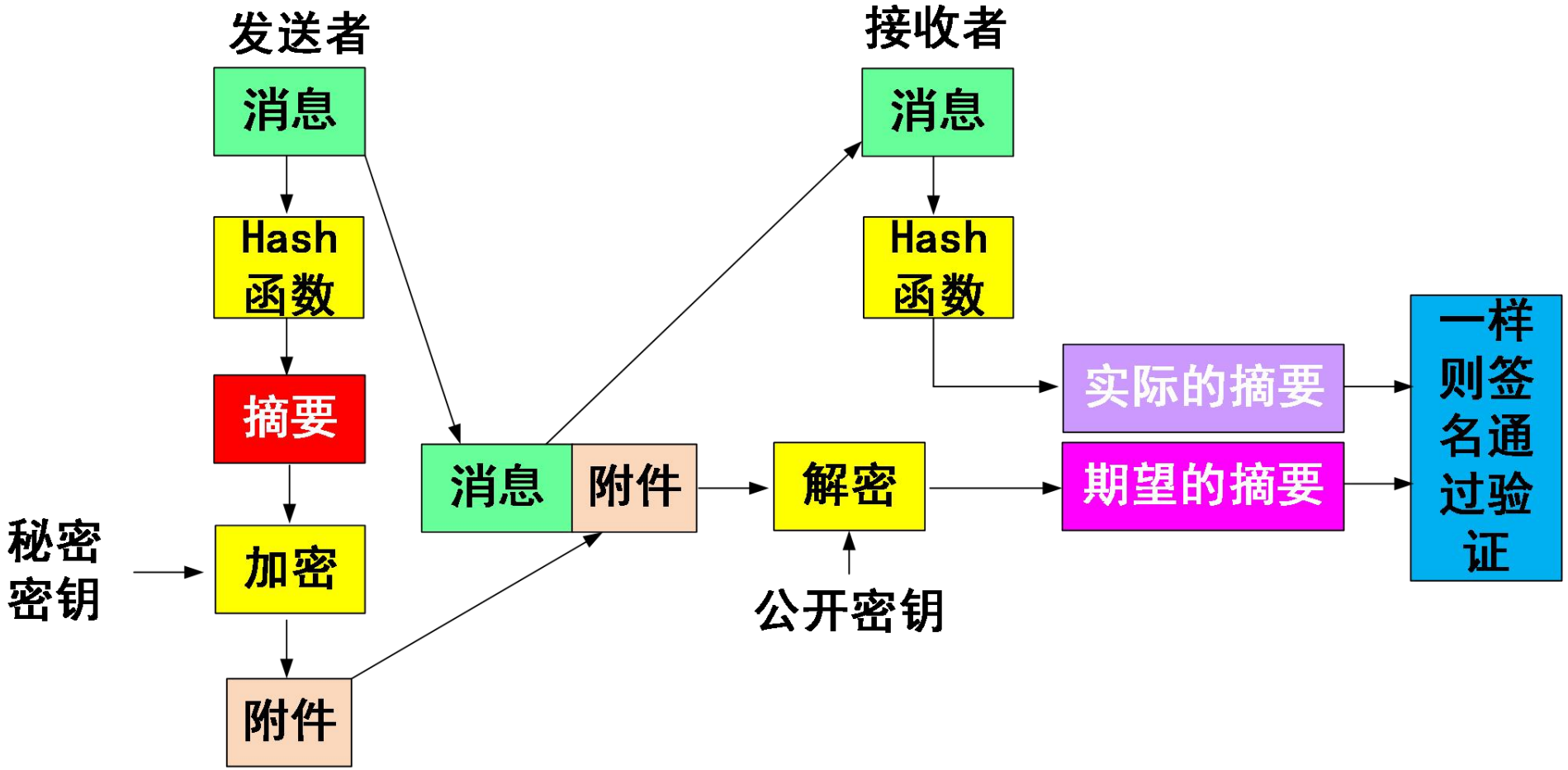


基本的数字签名方案

基于公钥的数字签名的实现

- 因为除 A 外没有别人能具有 A 的私钥，所以除 A 外没有别人能产生这个密文。因此 B 相信报文 X 是 A 签名发送的。
- 若 A 要抵赖曾发送报文给 B，B 可将明文和对应的密文出示给第三者。第三者很容易用 A 的公钥去证实 A 确实发送 X 给 B。
- 反之，若 B 将 X 伪造成 X’，则 B 不能在第三者前出示对应的密文。这样就证明了 B 伪造了报文。

使用Hash函数的数字签名方案



使用Hash函数的数字签名方案

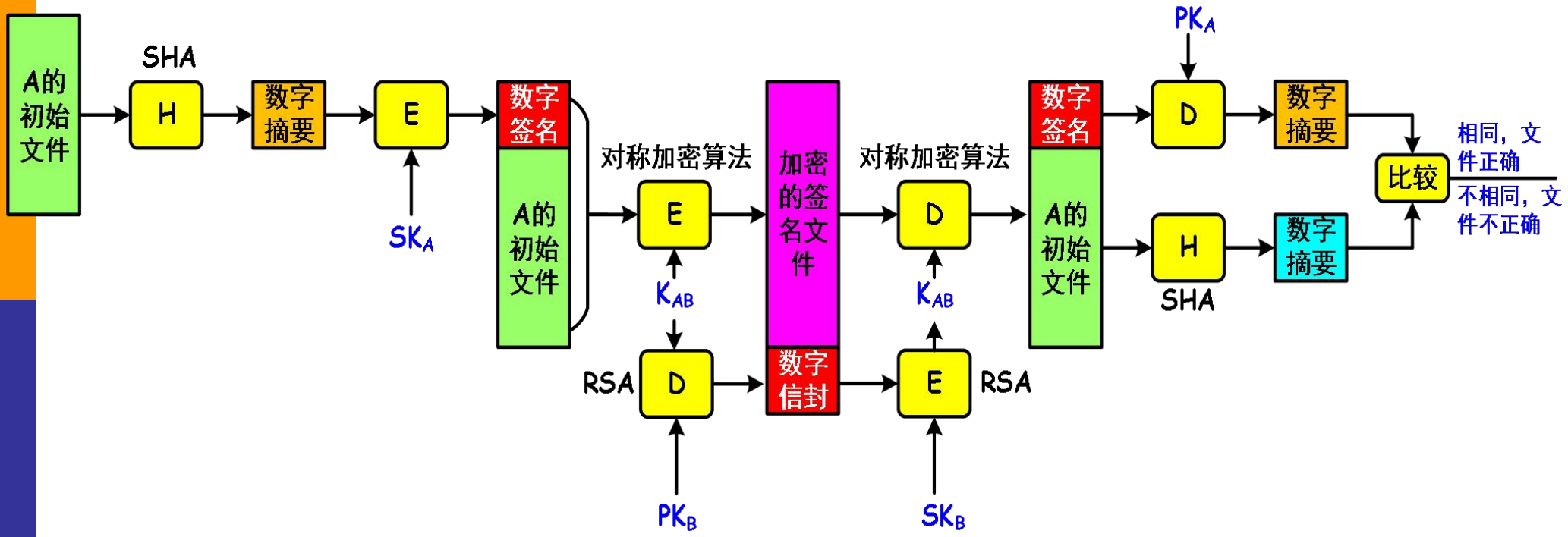
- 发送方
  - 使用Hash函数对要签名的消息进行压缩，生成摘要
  - 对压缩后的消息（即摘要）进行签名，获得附件
  - 将消息+附件一起发送给接受者
- 接收方
  - 接收者根据收到的消息重新计算摘要（收）
  - 解密附件获得的摘要（发）
  - 如果两者一样，则数字签名是合法的

美国数字签名算法（DSA）

- DSA（Digital Signature Algorithm）
  - 公开密钥技术，仅作为数字签名算法
- 数学基础
  - 基于ElGamal，安全性依赖计算离散对数的困难性
- DSA与RSA的区别
  - DSA仅支持数字签名，不支持加解密
  - RSA既支持数字签名，也支持加解密。

美国数字签名标准（DSS）

- 1991年8月，DSA由美国NIST标准化作为DSS(Digital Signature Standard)



小结

- 消息认证算法只需要单向计算，不需要算法可逆，而可逆对解密是必须的，因此跟加密算法相比更不易被破解。
- 消息认证与消息整体加密相比具有的优势
  - 软件加密运算速度慢
  - 硬件加密成本高，且仅对大批量数据加密有优势
  - 算法可能受专利保护

版权声明

本PPT部分图片直接使用如下材料中的图片：

[LoQM] Lecture of Queen Mary University of London, EBU7140 Security and Authentication, week2, Oct 2020