第五讲：认证与杂凑函数

一、认证与认证系统

二、消息完整性

三、杂凑函数

四、杂凑函数的设计理论

五、杂凑函数算法示例

六、消息认证码

前 言

保密的目的是防止对手破译系统中的机密信息；认证(Authentication)则是防止主动攻击，如伪装、窜扰等的重要技术，包括对消息的内容、顺序、和时间的窜改以及重发等。

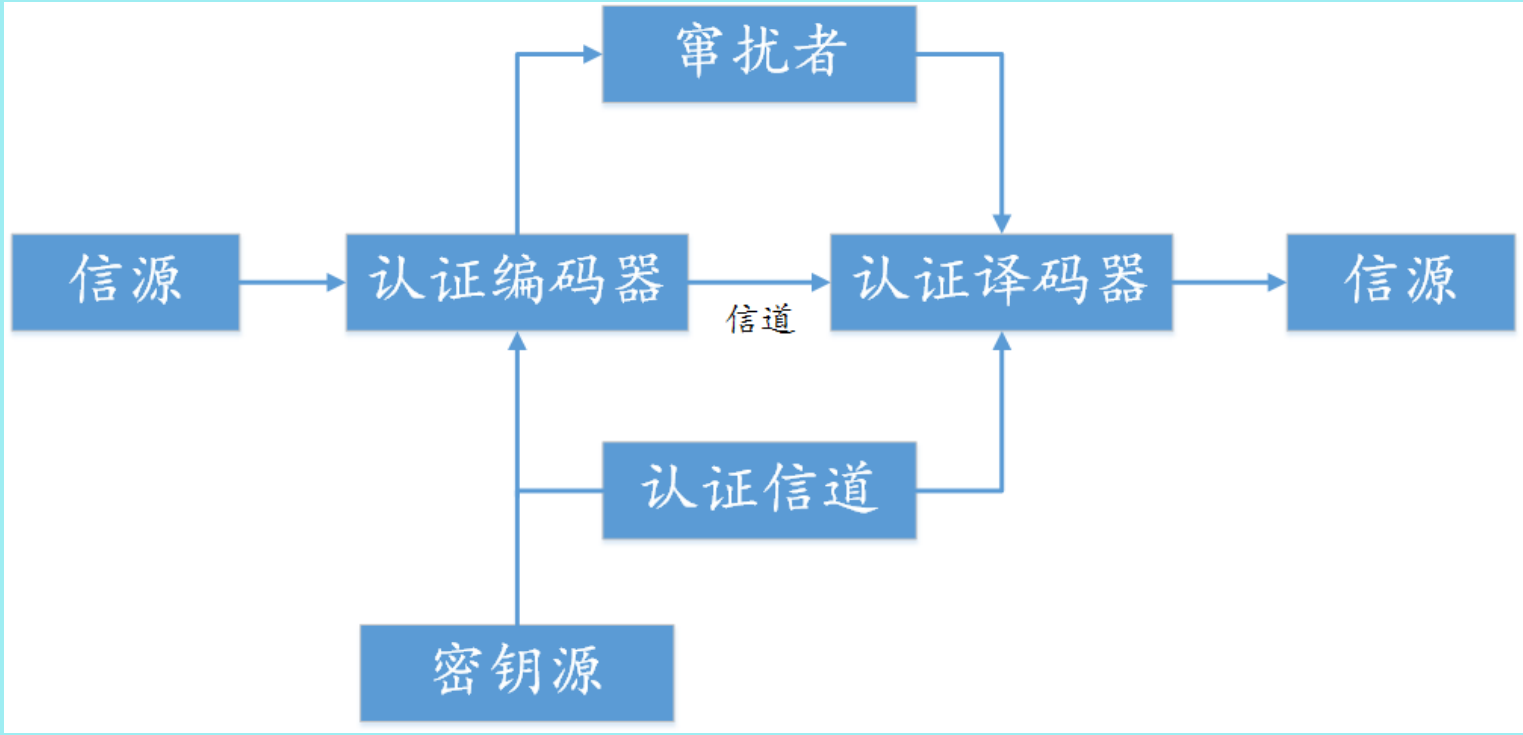
认证类别：

- **实体认证** (entity Authentication)：验证信息的发送者是真的、而不是冒充的，包括信源、信宿等的认证和识别；
- **数据源认证** (Data origin Authentication)：也称为消息认证(message authentication)，验证数据在传送或存储过程中未被窜改、重放或延迟等。

一、认证与认证系统

类似于保密系统的信息理论，可将信息论用于研究认证系统的理论安全性和实际安全性问题，指出认证系统的性能极限以及设计认证码必须遵循的原则。保密和认证同是信息系统安全的两个重要方面，但它们是两个不同属性的问题。认证不能自动地提供保密性，而保密也不能自然地提供认证功能。在这个系统中的发送者通过一个公开信道将消息送给接收者，接收者不仅想收到消息本身，而且还要验证消息是否来自合法的发送者及消息是否经过窜改。系统中的密码分析者不仅要截收和分析信道中传送的密报，而且可伪造密文送给接收者进行欺诈，称其为系统的**窜扰者**(Tamper) 更加贴切。实际认证系统可能还要防止收、发之间的相互欺诈。

认证与认证系统



认证系统模型

认证与认证系统

类似于保密系统的安全性，认证系统的安全性也划分为两大类，即**理论安全性**和**实际安全性**。

理论安全性又称作**无条件安全性**，就是我们上面所讨论的。它与窜扰者的计算能力或时间无关，也就是说窜扰者破译体制所做的任何努力都不会优于随机试凑方式。

实际安全性是根据破译认证体制所需的计算量来评价其安全性的。如果破译一个系统在原理上是可能的，但以所有已知的算法和现有的计算工具不可能完成所要求的计算量，就称其为**计算上安全**的。如果能够证明破译某体制的困难性等价于解决某个数学难题，就称其为**可证明安全的**，如RSA体制。这两种安全性虽都是从计算量来考虑，但不尽相同，计算安全要算出或估计出破译它的计算量下限，而可证明安全则要从理论上证明破译它的计算量不低于解已知难题的计算量。

二、消息完整性

- 早期的教科书认为数据源认证和消息完整性验证无本质的区别。
- 两者之间的区别：
  - 数据源认证 必须涉及通信，而消息完整性则不一定，它有可能是存储的数据。
  - 数据源认证 涉及到消息源的身份，而消息完整性则不一，它也提供无源认证。
  - 数据源认证 涉及到消息的更新，而消息完整性则不一定，一个陈旧的消息也有完善的完整性。

消息完整性

- **传送错误：**由于通信中或处理数据过程中的故障而引起消息中的错误。
- **主动攻击：**敌手有意的篡改、伪造、重放消息。
- **解决方法：**错误检测码技术和消息完整性技术。

这两种技术本质上一样的。消息的传送者通过给消息编码注入一定的冗余度来产生“校验值”，并将消息和该校验值一起发送；接受者通过该校验值来验证消息的正确性。

消息完整性

编码方式和方法不同：

- 错误检测码中，接收者能从他所接收到的可能有错的码中推断出真正最可能传递的消息。
- 在数据完整性保护中，所附加的检验值正态的分布于整个它所在的消息空间中，使得攻击者伪造一个正确的检测值的概率是可忽略的。

消息完整性

数据完整性保护： k_e是编码密钥，k_v是对应的验证密钥。

- **窜改检测码生成：**

$$MDC < f(k_e, Data)$$

- **验证算法：**

$$Ver_a = \begin{cases} 真, 当\ MDC = f(k_e, Data) \\ 假, 当\ MDC \neq f(k_e, Data) \end{cases}$$

- **方法：** 对称技术和非对称技术

三、杂凑函数

杂凑 (Hash) 函数： 压缩 (Compression) 函数、紧缩 (Contraction) 函数、数据认证码 (Data Authentication Code) 、消息摘要 (Message Digest)、数字指纹、数据完整性校验 (Data Integity Check)、密码检验和 (Cryptographic Check Sum) 、消息认证码MAC (Message Authentication Code) 、窜改检测码MDC (Manipulation Detection Code)等。

将任意长数字串M 映射成一个较短的定长输出数字串H 的函数，以h 表示，$h(M)$易于计算，称$H = h(M)$为M 的杂凑值，也称杂凑码、杂凑结果等或简称杂凑。

特点： H 打上了输入数字串的烙印，为数字指纹 (Digital Finger Print)。h 是多对一映射，因此我们不能从H 求出原来的M，但可以验证任一给定序列M'是否与M 有相同的杂凑值。

杂凑函数

分类：

- **密码杂凑函数**，有密钥控制，以 $h(k,M)$表示。其杂凑值不仅与输入有关，而且与密钥有关，只有持此密钥的人才能计算出相应的杂凑值，因而**具有身份验证功能**，如消息认证码 MAC[ANSI X 9.9]。杂凑值也称作**认证符**(Authenticator)或**认证码**。它要满足各种安全性要求，密码杂凑函数在现在密码学中有重要作用。
- **一般杂凑函数**，无密钥控制。无密钥控制的单向杂凑函数，其杂凑值只是输入字串的函数，任何人都可以计算，因而**不具有身份认证功能**，只用于检测接收数据的完整性，如窜改检测码MDC，用于非密码计算机应用中。

应用：在密码学和数据安全技术中，它是实现有效、安全可靠数字签字和认证的重要工具，是安全认证协议中的重要模块。

1. 单向杂凑函数

密码学中所用的杂凑函数必须满足安全性的要求，要能防伪造，抗击各种类型的攻击，如**生日攻击**、**中途相遇攻击**等等。为此，必须深入研究杂凑函数的性质，从中找出能满足密码学需要的杂凑函数。首先我们引入一些基本概念。

1. 单向杂凑函数

单向函数不仅在构造双钥密码体制中有重要意义，而且也是杂凑函数理论中的一个核心概念。

定义1 若杂凑函数h为单向函数，则称其为**单向杂凑函数**。

显然，对一个单向杂凑函数h，由M 计算$H = h(M)$是容易的，但要产生一个M'使$h(M')$等于给定的杂凑值H是件难事。

单向杂凑函数

定义2 弱单向杂凑函数：若单向杂凑函数h，对任意给定M 的杂凑值$H = h(M)$ 下，找一M'使 $h(M') = H$ 在计算上不可行。

定义3 强单向杂凑函数：对单向杂凑函数h，若要找任意一对输入M_1和 M_2， $M_1 \neq M_2$， 使$h(M_1) = h(M_2)$ 在计算上不可行。

无碰撞(Collision-free)性：弱单向杂凑，是在给定M 下，考察与特定M的无碰撞性；而强单向杂凑函数是考察输入集中任意两个元素的无碰撞性。显然，对于给定的输入数字串的集合，后一种碰撞要容易实现。因为从**生日悖论**知，在N 个元素的集中，给定M 找与M 相匹配的M' 的概率要比从N 中任取 一对元素M，M'相匹配的概率小得多。

2. 杂凑函数的安全性

2. 杂凑函数的安全性：杂凑函数的安全性取决于其抗击各种攻击的能力，对手的目标是找到两个不同消息映射为同一杂凑值。一般假定对手知道杂凑算法，采用选择明文攻击法。

对杂凑函数的基本攻击方法：

- **穷举攻击法：**给定$h = h(H_0, M)$，其中H_0 为初值，攻击者在所有可能的M 中寻求有利于攻击者的M'，使$h(H_0, M') = h(H_0, M)$，由于限定了目标$h(H_0, M)$ 来寻找$h(H_0, M')$，这种攻击法称为**目标攻击**。若对算法的初值H_0 不限定，使其$h(H_0', M)$ 等于$h(H_0, M')$，则称这种攻击法为**自由起始目标攻击**。

杂凑函数的安全性

- **生日攻击：**这种攻击法不涉及杂凑算法的结构，可用于攻击任何杂凑算法。强杂凑函数正是基于生日悖论一类的攻击法定义的。穷举和生日攻击都属选择明文攻击。生日攻击给定初值H_0，寻找$M' \neq M$，使$h(H_0, M') = h(H_0, M)$，也可对初始值H_0不加限制，即寻找H_0'，M'使$h(H_0', M') = h(H_0, M)$。

生日悖论：

1. 在一个房间里，至少需要多少人，才可以找到一个与 Alice 的生日为同一天的概率大于1/2?
2. 在一个房间里，至少需要多少人，才可以找到两个人的生日为同一天的概率大于1/2？

$Answer$: $P = 1 - \left(\frac{364}{365}\right)^{t-1}$ $t = 183$

$P = 1 - \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\ldots\left(1 - \frac{t-1}{365}\right)$ $t = 23$

杂凑函数的安全性

例 令h是一个杂凑值为 80 bit 的单向杂凑函数。给定消息 M 和$H(M)$下，假定2^{80} 个杂凑值等概，则对 M 有$\mathrm{P_r}\{h(M)\} = 2^{-80}$。今进行 k 次试验，找到一个 M' 能使 $h(M') = h(M)$ 的概率为$k2^{-80}$。因此，当进行2^{80}次试验时，找到满足要求的 M' 的概率近于1。

若在不限定杂凑值的情况下，在消息集中找出一对消息 M 和 M'使$h(M) = h(M')$，根据生日悖论所需的试验次数(p75)，至少为$1.177\times 2^{40} < 2\times 10^{12}$。利用大型计算机，至多用几天时间就能实现。

可见，杂凑值仅为 80bit 的杂凑函数不是强杂凑函数。因此，能抗击生日攻击的杂凑函数值至少为 128 bit。

- **中途相遇攻击法** 这是一种选择明文/密文的攻击。用于迭代和级连分组密码体制，其概率计算同生日攻击。由于杂凑算法也可用迭代和级连结构，因而设计算法时必须能抗击这类攻击。

3. 杂凑函数的应用

- 在数字签名方案中，hash 函数被用作产生消息摘要。签名实际上是对摘要的签名。
- 为公钥加密方案提供可证明的安全性。提供密文的正确性验证机制，也就是提供了数据的完整性证明，但是不保证消息源的身份证明，如RSA-OAEP 加密方案。
- 在密钥协商方案中或零知识证明协议中，提供实用的伪随机数。
- 随机预言：(random oracle) 结合了确定性、有效性、正态分布性三个性质。在真实世界中其实不存在的一种非常理想的函数。与 hash 函数的区别：hash 函数的输出只是计算上不可区分的正态分布，实际上不可能是真正的正态分布。

四、单向迭代杂凑函数的设计理论

安全的单向迭代函数是构造安全消息杂凑值的核心和基础。有了好的单向迭代函数，就可以利用6-3节中给出的迭代方式形成杂凑值了。为了抗击生日和中途相遇攻击，杂凑值至少应为128 bit。从实际应用出发，要求杂凑算法快速、易于实现。

满足安全性要求，从理论上看最重要的有两点：一是函数的**单向性**，二是函数映射的**随机性**。

- **单向性。**单向性保证了求逆的困难性，这不仅对密码体制有决定意义，而且对杂凑函数设计也是一个基本条件。不具有单向性的函数是不能作认证用的。

一个函数的单向性，如果求逆的困难性超过$O(2^{64})$，则可认为此函数为**计算复杂度意义下的单向函数。**

单向迭代杂凑函数的设计理论

杂凑函数如果生日攻击下的难度为$O(2^{64})$，则可认为是**无碰撞的**。函数的单向性和无碰撞性的关系如何？Rompel [1990] 证明了一个重要结果，即**无碰撞函数存在的充要条件是单向函数的存在**。但这是否就有函数的单向性⇔函数的碰撞性，还不清楚。

- **伪随机性。**

迭代函数是一种映射，如果能通过某种可判定的检验逻辑验证，则称其为**伪随机映射**。已经证明若要迭代函数能抗击生日攻击，则它必须是伪随机映射函数。

如果伪随机性迭代函数的逆也伪随机的，就称其为**超伪随机函数**。

单向迭代杂凑函数的设计理论

已经证明要迭代函数能抗击中途相遇攻击，它必须是超伪随机的。这个结果告诉我们超伪随机性函数作为迭代函数才能抗击中途相遇攻击。已经证明了一些特定结构的迭代函数的伪随机性和超伪随机性。从而可用来构造安全迭代函数。

- **抗差分攻击能力。**可以用差分分析来攻击迭代函数。因此，要求在构造杂凑算法的迭代函数时，必须使其能经受此类攻击。
- **非线性性。**类似于分组密码，还有许多性质影响安全性。如雪崩特性、高阶相关免疫性等等。

单向迭代杂凑函数的设计理论

对杂凑函数来说。主要要求是能够提供认证和抗击生日及中途相遇碰撞能力。因此，函数的单向性和伪随机及超伪随机性是最根本的要求。如何构造具有伪随机性和超伪随机性的函数，它们的结构特点等是杂凑函数理论的核心。

虽然单向杂凑函数和分组密码设计上有很多相似的要求，但一个好的单向杂凑函数未必就是一个安全的分组加密算法，反之亦然，因为两者要求不一样。例如，线性攻击对单向杂凑函数没有太大意义，有些单向杂凑函数如 SHA 可有线性特性，但并不影响其安全性，但用于消息加密则不能保证安全性。如前所述，对杂凑迭代函数而言，能抵抗生日和中途相遇碰撞攻击是本质的要求，但一个安全的分组密码算法未必能承受这类攻击。

五、安全杂凑算法示例

- http://pajhome.org.uk/
- http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html
- **MD5**：Ron Rivest 于1990 年提出MD-4 杂凑算法, 特别适于软、硬件快速实现。输入消息可任意长，压缩后输出为128bits。MD-5是MD-4 的改进形式。
- **SHA**：**安全杂凑算法**是美国NIST 和NSA 设计的一种标准算法SHA (Secure Hash Algorithm)，用于数字签字标准算法DSS (Digital Signature Standard)，亦可用于其它需要用Hash 算法的情况。SHA具有较高的安全性。

MD-5杂凑算法

MD-5的安全性。

求具有相同Hash 值的两个消息在计算上是不可行的。MD-5 的输出为128-bit，若采用纯强力攻击寻找一个消息具有给定Hash 值的计算困难性为2^{128}，用每秒可试验1000 000 000 个消息的计算机需时1.07×10^{22}年。若采用生日攻击法，寻找有相同Hash 值的两个消息需要试验2^{64}个消息，用每秒可试验1000 000 000 个消息的计算机需时585年。差分攻击对MD-5 的安全性不构成威胁。

对单轮MD-5 已有攻击结果。与Snefru 比较，两者均为32 bits 字运算。Snefru 采用S-BOX，XOR函数，MD5用mod 2^{32}加。

MD-5杂凑算法

MD-5的实现

速度：用32 bits 软件易于高速实现。

简洁与紧致性：描述简单，短程序可实现，易于对其安全性进行评估。

采用了little-endian结构。Intel 80xxx 将字的最低位byte 存于低地址byte 位 (little endian) 而SUN Sparcstation 将最高位byte 存于低地址byte 位 (big endian) 在处理32 bit 字时，两者位置相反。Rivest 选择little endian 来表述消息 (32 bit)，这样做处理速度要快些。

MD-5杂凑算法

MD4 与MD5 算法差别：MD-5较MD-4复杂，且较慢，但安全性较高。

- MD-4三轮，每轮16步；MD-5，四轮，每轮16步。
- MD-4第一轮中不用加常数，第二轮每一步用同一个加常数，另外一个加常数用于第三轮中每一步。MD-5在64步中都用不同的加常数M[*i*]。
- MD-5采用4个基本逻辑函数，每轮一个；MD-4采用3个基本逻辑函数，每轮一个。
- MD-5每一步都与前一步的结果相加，可加快雪崩，MD-4这类最后相加运算。

SHA—安全杂凑算法

输入消息长度小于2^{64} bit，输出压缩值为160 bit，而后送给DSA (Digital Signature Algorithm) 计算此消息的签字。这种对消息Hash 值的签字要比对消息直接进行签字的效率更高。计算接收消息的Hash值，并与收到的Hash值的签字证实值 (即解密后恢复的Hash值) 相比较进行验证。伪造一个消息，其Hash 值与给定的Hash 值相同在计算上是不可行的，找两个不同消息具有同一Hash 值在计算上也是不可行的。消息的任何改变将以高概率得到不同的Hash 值，从而使验证签字失败。SHA 的基本框架与MD4 类似。

SHA的安全性

SHA的安全性

SHA很类似于MD-4，主要改变是增加了扩展变换，将前一轮的输出加到下一轮，以加速雪崩效应。SHA 与重新设计的MD-5 的差别较大。R.L.Rivest 公开了MD-5 的设计决策，但SHA 的设计者则不愿公开。

SHA的安全性

SHA与MD-4和MD-5的比较

	MD4	SHA	MD5
Hash值	128 bit	160 bit	128 bit
分组处理长	512 bit	512 bit	512 bit
基本字长	32 bit	32 bit	32 bit
步数	48（3×16）	80（4×20）	64 (4×16)
消息长	$\leq 2^{64}$ bit	$\leq 2^{64}$ bit	不限
基本逻辑函数	3	3（第2，4轮相同）	4
常数个数	3	4	64
速度	—	约为MD4的3/4	约为MD4的1/7

其它的杂凑算法

还有一些有名的杂凑算法

- **SNEFRU算法。** R. Merkle 设计的一种适于32-bits 处理器实现的单向杂凑算法。它将任意长消息压缩成128 bit 或256 bit。
- **RIPE-MD算法。**它是欧共体RIPE 计划下开发的杂凑算法，为MD4的变型。杂凑值为128bit，修正了MD4 的旋转和消息的次序。此外，所用常数也不同于MD-4，且采用并行；每组之后，两种情况的输出都加于链接变量上。REPE-MD 的改进型为REPEMD-160。
- **HAVAL算法。**它是一种变长杂凑函数，由Y. Zheng 等提出，是MD5的一种修正形式。以8 个32bit 链接变量，两倍于MD5 的杂凑值，轮数亦可变 (3～5轮，每轮16步)。杂凑值可取n =128, 160, 192, 224, 256 bit。

其它的杂凑算法

- **RIPE-MAC**。由B. Preneel 提出并被RIPE 采用，它基于ISO 9797，以DES 作为基本迭代函数。有两种型号：RIPE-MAC1和RIPE-MAC3。前者用一个DES，后者用三个DES。先将消息填充为64 bit 的倍数，而后划分成64 bit 的组，最后在密钥的控制下对消息进行杂凑。
- **其它**。利用模n运算构造的杂凑算法，如MASH-1 (Modular Arithmetic Secure Hash, Algorithm- l) 已作为ISO/IEC标准 (草案)。Coppersmith 对MASH-1 进行了分析。MAA算法也采用模n 整数运算[并已成为ISO 银行标准[ISO 8731-2]。

MAC (消息认证码)

- 基于密钥控制的hash 函数：$MAC = h(k||M)$
- 基于分组加密算法 (CBC模式)

$$C_i = E_k(m_i \oplus C_{i-1}), i = 1,2, \ldots, l$$

$$C_0 = IV$$

$$MAC = (IV, C_l)$$

thank you