第 3 章　认证与数字签名

认证或鉴别（authentication）是信息安全领域的一项重要技术，主要用于证实身份合法有效或者信息属性名副其实。身份认证是最常见的一种认证技术，可以确保只有合法用户才能进入系统。其次是消息认证，在网络通信过程中，黑客常常伪造身份发送信息，也可能对网络上传输的信息内容进行修改，或者将在网络中截获的信息重新发送，因此要验证信息的发送者是合法的，即信源的认证和识别；验证消息的完整性，即验证信息在传输和存储过程中是否被篡改；验证消息的顺序，即验证是否插入了新的消息、是否被重新排序、是否延时重放等。最后，在通信过程中，还需要解决通信双方互不信任的问题，如发送方否认消息是自己发送的，或接收方伪造一条消息谎称是发送方发送的，这就需要用到数字签名技术。本章主要介绍认证技术的基本原理，包括：消息认证、数字签名和身份认证。

3.1　散列函数

散列函数是现代密码学的重要组成部分，不仅用于认证，还与口令安全存储、恶意代码检测、正版软件检测、数字签名相关。因此，本节首先介绍散列函数。

3.1.1　散列函数的要求

散列函数（Hash function），也称为“哈希函数”或“杂凑函数”，在应用于长度任意的数据块时将产生固定长度的输出。散列函数可以表示为：

$$h = H(M)$$

其中，H 代表散列函数，M 代表任意长度的数据（可以是文件、通信消息或其他数据块），h 为散列函数的结果，称为“散列值”或“散列码”。当 M 为通信消息时，通常将 h 称为“报文摘要（message digest）”或“消息摘要”。对于特定的一种散列函数，散列值的长度是固定的。对 M 的任意修改都将使 M 的散列值出现变化，通过检查散列值即可判定 M 的完整性。因此散列值可以作为文件、消息或其他数据块的具有标识性的“指纹”（通常称为“数字指纹”）。

采用散列函数来保证文件（或消息）的完整性，首先需要通过散列函数获得文件（或消息）的散列值，并将该值妥善保存。在需要对文件（或消息）进行检查的时候，重新计算文件（或消息）的散列值，如果发现计算得到的散列值与保存的结果不同，则可以推断文件（或消息）被修改过。

安全领域使用的散列函数通常需要满足一些特性[1]，如表 3-1 所示。

表 3-1　散列函数 H 的安全性需求

安全性需求	说明
（1）输入长度可变	H 可应用于任意长度的数据块
（2）输出长度固定	H 产生定长的输出
（3）效率	对于任意给定的 x，计算 $H(x)$比较容易，并且可以用软件或硬件实现
（4）抗原像攻击（单向性）	对任意给定的散列值 h，找到满足 $H(x) = h$ 的 x 在计算上是不可行的
（5）抗第二原像攻击（抗弱碰撞性）	对任意给定的数据块 x，找到满足 $y \neq x$ 且 $H(x) = H(y)$的 y 在计算上是不可行的
（6）抗碰撞攻击（抗强碰撞性）	找到任意满足 $H(y) = H(x)$的偶对（x, y）在计算上是不可行的
（7）伪随机性	H 的输出满足伪随机性要求

具有单向性的散列函数可以应用于用户口令（或密码）存储。在信息系统中如果口令以明文形式存储，则存在很大的安全风险。一旦攻击者进入系统，或者系统由恶意的管理员管理，用户口令很容易泄露。而采用散列值来存储口令，散列函数的单向性可以确保即使散列值被攻击者获取，攻击者也无法简单地通过散列值推断出用户口令。同时，这种方法也不会影响对用户进行身份认证，用户在登录时输入的口令通过散列函数计算，如果所得的散列值与系统存储的相应账号的散列值相同，则允许用户进入系统。

抗弱碰撞性对于保证消息的完整性非常重要。举例来看，用户发送消息 M，为了确保消息的完整性，将消息 M 的散列值的加密结果一同发送。在此过程中，之所以对散列值进行加密，是因为散列函数是公开的，如果不加密，攻击者可能修改消息 M 并同时产生修改后消息的散列值，接收方将难以察觉异常。如果散列函数不满足抗弱碰撞性，则攻击者可以找到一个不同于 M 的消息 M'，M' 的散列值与 M 的散列值相同，攻击者如果用消息 M' 替换 M，消息的接收方将无法发现消息已经遭到了篡改。

一个函数如果是抗强碰撞的，那么也同样是抗弱碰撞的，但反之则不一定成立。一个函数可以是抗强碰撞的，但不一定是抗原像攻击的，反之亦然。一个函数可以是抗弱碰撞的，但不一定是抗原像攻击的，反之亦然。

如果一个散列函数满足安全性需求（1）至（5），则称该函数为弱散列函数，如果还满足第（6）条需求，则该散列函数为强散列函数。第（6）条需求可以防止像生日攻击[2]这种类型的复杂攻击，生日攻击把n比特的散列函数的强度从 2^n降低到 $2^{n/2}$。一个 40 比特长的散列码是很不安全的，因为仅仅用 2^{20}（大约一百万）次随机Hash就可至少以 1/2 的概率找到一个碰撞。为了抵抗生日攻击，建议散列码的长度至少应为 128 比特，此时生日攻击需要约 2^{64} 次Hash计算。通常将安全的Hash标准的散列码长度选为≥160 比特。

各种数据完整性应用中的散列函数安全性需求如表 3-2 所示。

1　数据结构中的“散列表（Hash Table）”使用的函数也称为“散列函数”，但这种散列函数与安全领域的散列函数的差别较大，产生的碰撞较多，通常不满足表 3-1 中所列的大部分安全性需求。为示区别，有些文献将安全领域的散列函数称为“安全的散列函数”或“密码学散列函数”。

2　这种攻击源于所谓的生日问题。在一个教室中最少应该有多少学生才使得至少有两个学生的生日在同一天（不考虑年份，只考虑出生月、日）的概率不小于 1/2？我们假定：不计闰年，一年 365 天，那么 n 个人中任何两人生日都不相同的概率为：1 – [(365/365) * (364/365) * (363/365) * ..... * [(365-n+1)/365]。当 n = 23 时，概率为 50.7%；当 n = 36 时，概率高达 83.2%，当 n 达到 60 左右时，概率几乎达到 100%。因此，这个问题的答案为 23。当然，这并不意味着当一个班级有 23 个人时，有人和你同一天生日的可能性会达到 50%。具体到某一个人时，其生日已固定，其余 22 人与他的生日相同的概率为：1 - (364/364)22 ≈ 6%。

表 3-2　各种数据完整性应用中的散列函数安全性需求

	抗原像攻击	抗弱碰撞攻击	抗强碰撞攻击
Hash + 数字签名	是	是	是*
恶意代码检测		是	
Hash + 对称加密			
单向口令文件	是		
消息认证码 MAC	是	是	是*

表注：标*处要求攻击者能够实现选择消息攻击。

除了前面提到的口令安全存储，散列函数还可用于多种安全场景。在恶意代码检测中，常常利用散列函数计算已知恶意代码（源代码或可执行代码）的散列值，并保存在恶意代码特征库中。当需要检测截获的一段代码是否是恶意代码时，首先计算这段代码的散列值，并将其与特征库的散列值进行比较，如果有相等的，则可快速判断出该代码是一个已知的恶意代码。如果不是进行散列值比较，而是进行整段代码比较，则面对海量恶意代码时，效率非常低下。

同样的道理，散列函数亦可用于原版软件检测。当用户下载一个软件后，需要知道这个软件是被修改过的（很多攻击者经常将正版软件重打包，将恶意代码插入其中），还是没改过的原版软件，此时他只需计算该软件的散列值，并将其与软件厂商公布的原版软件的散列值进行比较。如果相同，则可放心使用；否则，该软件被修改过，需要进一步检测。

散列函数在数字签名和消息认证中的应用将在本章的后续章节讨论。MD、SHA 是目前比较流行的散列函数，下面对它们进行简要介绍。

3.1.2　MD 算法

报文摘要（Message Digest, MD）算法是由 Rivest 从 20 世纪 80 年代末所开发的系列散列算法的总称，历称 MD2、MD3、MD4 和最新的 MD5。

1991 年 Den Boer 和 Bosselaers 发表文章指出 MD4 算法的第 1 步和第 3 步存在可被攻击的漏洞，将导致对不同的内容进行散列计算却可能得到相同的散列值。针对这一情况，Rivest 于 1991 年对 MD4 进行了改进，推出新的版本 MD5 (RFC 1321)。

与 MD4 相比，MD5 进行了下列改进：

（1）加入了第 4 轮。

（2）每一步都有唯一的加法常数。

（3）第 2 轮中的 G 函数从$((X\wedge Y)\vee(X\wedge Z)\vee(Y\wedge Z))$变为$((X\wedge Z)\vee(Y\wedge\sim Z))$，以减小其对称性。

（4）每一步都加入了前一步的结果，以加快“雪崩效应”。

（5）改变了第 2 轮和第 3 轮中访问输入子分组的顺序，减小了形式的相似程度。

（6）近似优化了每轮的循环左移位移量，以期加快“雪崩效应”，各轮循环左移都不同。

MD5 的输入为 512 位分组，输出是 4 个 32 位字的级联（128 位散列值）。具体过程如下：

消息首先被拆成若干个 512 位的分组，其中最后 512 位分组是“消息尾+填充字节(100…0)+64 位消息长度”，以确保对于不同长度的消息，该分组不相同。而 4 个 32 位寄存

器字初始化为 A = 0x01234567，B = 0x89abcdef，C = 0xfedcba98，D = 0x76543210，它们将始终参与运算并形成最终的散列结果。

接着各个 512 位消息分组以 16 个 32 位字的形式进入算法的主循环，512 位消息分组的个数决定了循环的次数。主循环有 4 轮，每轮分别用到的非线性函数如下：

$F(X, Y, Z) = (X \wedge Y) \vee (\sim X \wedge Z)$

$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \sim Z)$

$H(X, Y, Z) = X \oplus Y \oplus Z$

$I(X, Y, Z) = X \oplus (Y \vee \sim Z)$

这 4 轮变换是对进入主循环的 512 位消息分组的 16 个 32 位字分别进行如下操作：将 A、B、C、D 的副本 a、b、c、d 中的 3 个经 F、G、H、I 运算后的结果与第 4 个相加，再加上 32 位字和一个 32 位字的加法常数，并将所得之值循环左移若干位，最后将所得结果加上 a、b、c、d 之一，并回送至 A、B、C、D，由此完成一次循环。

所用的加法常数由表 T 来定义，$T[i]$是 i 的正弦绝对值之 4 294 967 296 次方的整数部分（其中 i 为 1…64），这样做是为了通过正弦函数和幂函数来进一步消除变换中的线性特征。

当所有 512 位分组都运算完毕，$ABCD$ 的级联将被输出为 MD5 散列的结果。下面是一些 MD5 散列结果的例子：

MD5 ("") = d41d8cd98f00b204e9800998ecf8427e

MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661

MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72

MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0

MD5 ("12345678901234567890123456789012345678901234567890123456789012345678
901234567890") = 57edf4a22be3c955ac49da2e2107b67a

尽管 MD5 比 MD4 要复杂，导致其计算速度较 MD4 要慢一些，但更安全，在抗分析和抗差分方面表现更好。有关 MD5 算法的详细描述可参见 RFC 1321。

MD5 在推出后很长一段时间内，人们认为它是安全的。但在 2004 年国际密码学会议(Crypto'2004)上，来自中国山东大学的王小云教授做了破译 MD5、HAVAL-128、MD4 和 RIPEMD 算法的报告，提出了密码哈希函数的碰撞攻击理论，即模差分比特分析法；2005 年王小云教授等人又提出了 SHA-1 的破解方法。王教授的相关研究成果提高了破解包括 MD5、SHA-1 在内的 5 个国际通用的哈希函数算法的概率，给出了系列消息认证码 MD5-MAC 等的子密钥恢复攻击和 HMAC-MD5 的区分攻击方法。自此，这 5 个散列函数被认为不再安全。尽管如此，在一些安全性要求不是特别高的场合下，MD5 仍然不失为一种可用的散列函数算法。

3.1.3　SHA 算法

SHA（Secure Hash Algorithm）算法是使用最广泛的 Hash 函数，由美国国家标准与技术

研究院（NIST）和美国国家安全局（NSA）设计，包括 5 个算法，分别是 SHA-1、SHA-224、SHA-256、SHA-384 和 SHA-512，后四个算法有时并称为 SHA-2。SHA-1 在许多安全协议中被广为使用，如 TLS、SSL、PGP、SSH、S/MIME 和 IPsec 等。

SHA 算法建立在 MD4 算法之上，其基本框架与 MD4 类似。SHA-1 算法产生 160 bit 的散列值，因此它有 5 个参与运算的 32 位寄存器字，消息分组和填充方式与 MD5 相同，主循环也同样是 4 轮，但每轮进行 20 次操作，非线性运算、移位和加法运算也与 MD5 类似，但非线性函数、加法常数和循环左移操作的设计有一些区别。SHA-2 与 SHA-1 类似，都使用了同样的迭代结构和同样的模算法运算与二元逻辑操作。

不同版本 SHA 算法参数如表 3-3 所示。

表 3-3　SHA 参数比较

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
消息摘要长度	160	224	256	384	512
消息长度	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
分组长度	512	512	512	1 024	1 024
字长度	32	32	32	64	64
步骤数	80	64	64	80	80

下面以 SHA-512 为例，简要介绍算法的操作过程。

算法的输入是最大长度小于 2^{128} 位的消息，并被分成 1 024 位的分组为单位进行处理，输出是 512 位的消息摘要。算法的主要步骤如下：

（1）附加填充位。填充消息使其长度模 1 024 与 896 同余，即长度 ≡ 896 (mod 1024)。即使消息已经满足上述长度要求，仍然需要进行填充，因此填充位数在 1~1 024 之间，填充由一个 1 和后续的 0 组成。

（2）附加长度。在消息后附加一个 128 位的块，将其作为 128 位的无符号整数（最高有效字节在前），它包含填充前消息的长度。

前两步的结果是产生了一个长度为 1 024 整数倍的消息。经过扩展后的消息为一串长度为 1 024 位的消息分组 $M_1, M_2, \ldots, M_N$，总长度为 $N \times 1024$ 位。

（3）初始化 Hash 缓冲区。Hash 函数的中间结果和最终结果保存于 512 位的缓冲区中，缓冲区用 8 个 64 位的寄存器（a, b, c, d, e, f, g, h）表示，并将这些寄存器初始化为下列 64 位的整数（十六进制值）：

a = 6A09E667F3BCC908　　b = BB67AE8584CAA73B　　c = 3C6EF372FE94F82B
d = A54FF53A5F1D36F1　　e = 510E527FADE682D1　　f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B　　h = 5BE0CD19137E2179

这些值以高位在前格式存储，其获取方式如下：前 8 个素数取平方根，取小数部分的前 64 位。

（4）以 1 024 位的分组（128 个字节）为单位处理消息。算法的核心是具有 80 轮运算的模块。每一轮都把 512 位缓冲区的值 *abcdefgh* 作为输入，并更新缓冲区的值。每一轮，如第

j 轮，使用一个 64 位的值 W_j，该值由当前被处理的 1024 位消息分组 M_i 导出，导出消息即是消息扩展算法。每一轮还将使用附加的常数 K_j，其中 $0 \leqslant j \leqslant 79$，用来使每轮的运算不同。$K_j$ 的获取方法如下：对前 80 个素数开平方根，取小数部分的前 64 位。这些常数提供了 64 位随机串集合，可以初步消除输入数据里的统计规律。第 80 轮的输出和第 1 轮的输入 H_{i-1} 相加产生 H_i。缓冲区中的 8 个字和 H_{i-1} 中对应的字分别进行模 2^{64} 的加法运算。SHA-512 轮函数比较复杂，读者可参考文献[16]的 11.5.1 节。

（5）输出。所有 N 个 1 024 位分组都处理完以后，从第 N 阶段输出的是 512 位的消息摘要。

SHA-512 算法具有如下特性：散列码的每一位都是全部输入位的函数。基本函数 F 多次复杂重复运算使得结果充分混淆，从而使得随机选择两个消息，甚至这两个消息有相似特征，都不太可能产生相同的散列码。正常情况下，要找到两个具有相同摘要的消息的复杂度是 2^{256} 次操作，而给定摘要寻找消息的复杂度是 2^{512} 次操作。

SHA-1 的安全性如今被密码学家严重质疑。2005 年 2 月，王小云等人发表了对 SHA-1 的攻击，只需少于 2^{69} 的计算复杂度，就能找到一组碰撞，而此前的利用生日攻击法找到碰撞需要 2^{80} 的计算复杂度。此外，王小云还展示了对 58 次加密循环 SHA-1 的破密，在 2^{33} 个单位操作内就找到一组碰撞。2019 年 10 月，密码学家盖坦·勒伦（Gaëtan Leurent）和托马·佩林（Thomas Peyrin）宣布已经对 SHA-1 成功计算出第一个选择前缀冲突，并选择安全电子邮件 PGP/GnuPG 的信任网络（参见第 10 章）来演示 SHA-1 的前缀冲突攻击。目前为止尚未出现对 SHA-2 有效的攻击，它的算法跟 SHA-1 基本上相似，因此人们开始发展其他替代的散列算法。NIST 在 2007 年公开征集新一代 NIST 的 Hash 函数标准，称为 SHA-3，并于 2012 年 10 月公布了设计算法的优胜者，未来将逐渐取代 SHA-2。SHA-3 的设计者使用一种称为海绵结构的迭代结构方案，详细情况读者可参考文献[16]的 11.6 节。

SM3 是我国政府采用的一种密码散列函数标准，由国家密码管理局于 2010 年 12 月 17 日发布，相关标准为“GM/T 0004-2012 《SM3 密码杂凑算法》”，其安全性及效率与 SHA-256 相当。

3.2　消息认证

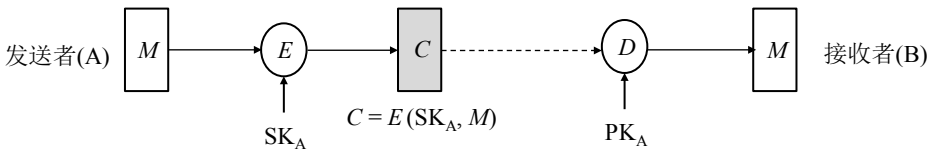
网络通信面临着诸多安全问题，例如，消息可能是由攻击者伪造身份发送的，消息在网络传输过程中可能遭受篡改，消息也可能是过时消息的重放或排序被打乱后的消息。消息认证，也称为“报文认证”“消息鉴别”，指的是通信双方对各自接收的消息进行验证，确定消息的一些属性是否真实的过程，经常被验证的消息属性包括发送方的身份、接收方的身份、内容的完整性以及消息的顺序等。以下介绍消息属性的主流认证方法。

3.2.1　报文源的认证

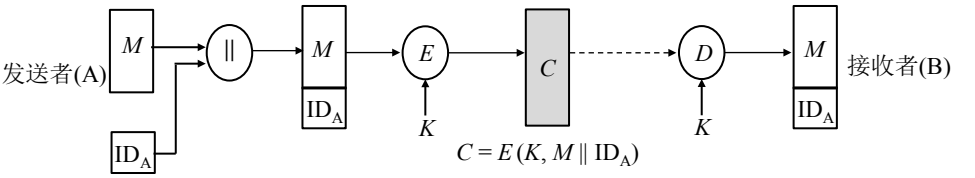
报文源的认证是指认证发送方的身份，确定消息是否由所声称的发送方发送而来。

首先，可以采用公开密钥密码系统来实现对报文源的认证，如图 3-1 (a)所示，发送方 A 用其私钥 SK_A 对消息加密，接收方用发送方的公钥 PK_A 进行解密。如果解密成功，则说明消

息是 A 发送的，因为只有 A 拥有 PK_A 对应的私钥 SK_A。这一过程也称为数字签名（将在 3.3 节介绍）。



（a）用公开密钥密码实现报文源认证



（b）用对称密钥密码实现报文源认证

图 3-1　报文源认证

上述方法中，如何判断解密成功是一个问题，这也是所有数据解密工作中必须面对的问题。如果传输的是可读的文本信息，那么接收方可以根据恢复出的文本信息是否有意义来判断解密是否成功。但是对于二进制数据，由于没有直观的可读内容，接收方很难判断解密是否成功。如果二进制数据是一种有结构的数据（比如，协议报文的字段信息，每个字段的特征信息等，或话音数据、视频数据），则可根据结构特征或是否听得懂（语音）或看得懂（视频）来判断消息的正确性。如果是纯随机的无结构二进制数据，则无法判断解密是否成功。我们将在本章的 3.2.3 节介绍这个问题的解决方案。

报文源认证也能够采用对称密钥密码系统实现。举例来看，A 是消息的发送方，B 是消息的接收方，甲乙双方共享密钥 K。在发送消息 M 之前，A 用密钥 K 通过对称密钥加密算法 E 加密消息 M，得到密文 $C = E(K, M)$，之后将密文 C 发送给 B。B 在接收消息以后，利用密钥 K 通过相应的对称密钥解密算法 D 解密，$M = D(K, C)$，恢复消息 M。在加解密的过程中，密钥 K 由 A、B 双方共享，对其他人保密。现在的问题是：B 作为接收方，如果接收到能够利用密钥 K 解密的密文，是否可以确定密文一定由 A 产生的？如果答案是肯定的，那么 B 可以采用这种方法确定消息的发送方。

对于上面提出的问题，答案是 B 方无法判定 A 是消息的发送方。举例来看，B 用密钥 K 加密消息发送给 A，消息被攻击者截获。由于消息经过了加密，攻击者无法获取消息内容，也不能对消息进行有效篡改。但是攻击者可以将消息保存，并在随后的时间发送给 B。在这种情况下，B 接收到的密文可以通过密钥 K 恢复，但并不是由 A 所产生，而是 B 自己在之前的时间发送的。

可以对前面提到的采用对称密钥密码系统加密的方法进行简单扩展达到报文源认证的目的。一种最简单的扩展方法是通过标识号标识用户身份，在发送消息时增加发送方的标识号。例如，以标识号 ID_A 作为 A 的身份标识。如图 3-1(b)所示，A 在发送消息时，将标识号 ID_A 与消息 M 拼接，利用密钥 K 加密后发送给 B，即发送的加密消息为 $E(K, M \| ID_A)$。B 在

接收密文后使用密钥 K 解密，依据消息中的标识号 ID_A 确定发送方的身份。

上述两种方法均不能应对重放攻击。如果不考虑信息重放攻击，用公开密钥密码系统实现的报文源认证可以保证信息的不可否认性，因为包括 B 在内的其他人不知道 A 的私钥，伪造不出报文 $E(SK_A, M)$。但是，用对称密钥密码系统实现的报文源认证则不能保证信息的不可否认性，因为通信双方都拥有共同的密钥 K，接收方 B 完全可以自己伪造一个报文 $E(K, M \| ID_A)$，声称是 A 发送的，而事实上 A 并没有发送这个报文。

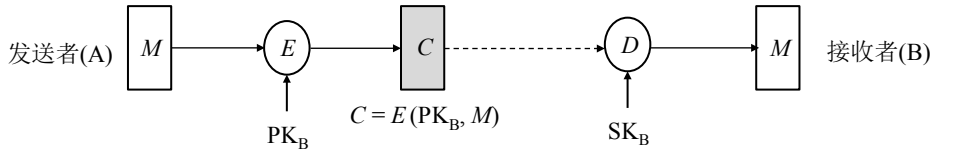
由于采用加密方式对传输的报文进行了加密，因此，上述两种方法在实现报文源认证的同时，还可保证报文传输的机密性。

3.2.2　报文宿的认证

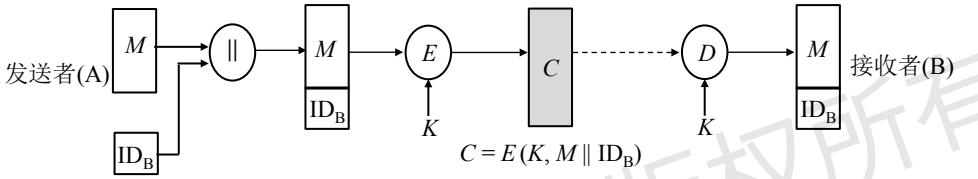
报文宿的认证是对报文接收方的身份进行认证，具体来看，它指的是消息的接收方在接收到报文以后判断报文是否是发送给自己的。

报文宿的认证在很多场合具有重要意义。例如，在战斗时我军的指挥员将一则要求撤退的命令发送给 A，并且为了确保接收方能够认证自己的身份，指挥员对命令进行了数字签名。攻击者截获该命令后，将该命令以指挥员的名义发送给 B。由于命令有指挥员的数字签名，B 通过数字签名确定命令由指挥员产生，同时，B 认为命令是由指挥员发送给自己的。在这种情况下，B 错误地执行了撤退命令，这个行动错误可能给我军带来很大损失。

报文宿的认证可以采用多种方法。公开密钥密码系统也可用于进行报文宿的认证，如图 3-2(a)所示。认证的核心思想是发送方 A 在发送消息 M 时，使用接收方 B 的公钥 PK_B 加密消息，密文 $C = E(PK_B, M)$。只有接收方使用自己的私钥 SK_B 才能解密消息，$D(SK_B, E(PK_B, M)) = M$，其他人由于没有相应密钥将无法解密。



（a）用公开密钥密码实现报文宿认证



（b）用对称密钥密码实现报文源认证

图 3-2　报文宿认证

同样，也可以用对称密钥密码系统为基础，采用与报文源认证相类似的方法认证报文宿，如图 3-2(b)所示。具体来看，通信双方共享一个密钥 K，发送方在发送消息时，在消息中增加能够标识接收方身份的标识号 ID_B，并将标识号与消息内容一起加密成 $E(K, M \| ID_B)$发送。接收方在接收到消息以后通过解密提取标识号，判断自己是否是报文的接收方。

3.2.3 报文内容的认证

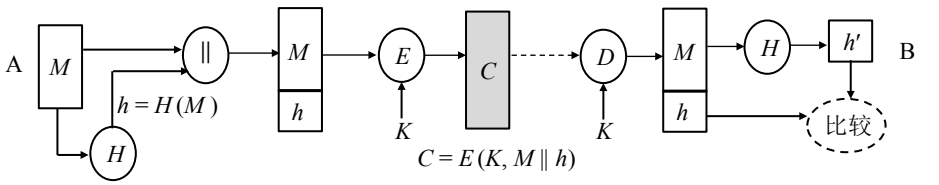
报文内容的认证是指接收方在接收到报文以后对报文进行检查，确保自己接收的报文与发送方发送的报文相同，即报文在传输过程中的完整性没有受到破坏，也称为“完整性检测”。

1. 加密的方法

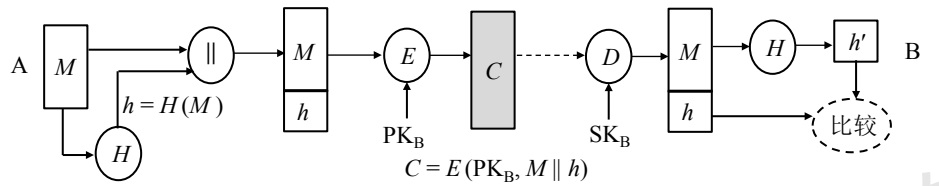
加密是确保报文内容完整性的有效方法。以 A 表示消息的发送方，B 表示消息的接收方，通信双方如果采用对称密钥密码系统，A 在发送消息前使用密钥加密消息，B 接收到消息后，使用相同的密钥解密消息。在此过程中，由于密钥为 A、B 共享，其他人没有掌握密钥也就无法对消息内容进行修改了。

这种方法必须解决一个问题，如何判断解密出来的消息是发送方发送的原始消息，而没有被修改过，也就是前面提到的如何判断解密是否成功的问题。

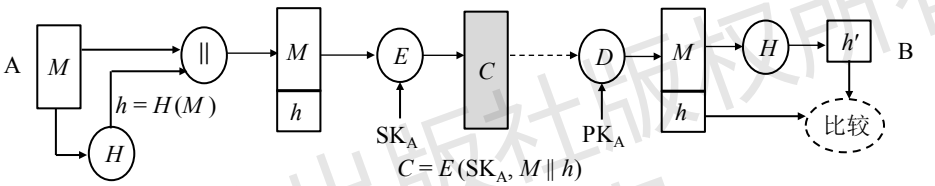
针对这一问题，可以利用前面介绍的哈希函数验证消息的完整性。如图 3-3(a)所示，以 M 表示消息，通过哈希函数 H 产生消息的哈希值 $h = H(M)$，利用 A、B 双方共享的密钥 K，A 使用加密函数 E，将消息与哈希值拼接后加密的密文 $E(K, M \| h)$发送给 B。B 通过解密提取消息 M，利用哈希函数获得消息的哈希值 h'，如果该值与 B 发送的哈希值 h 相同，则可以推断消息没有被篡改。这种方式可以保障消息传输的机密性和完整性。如上一小节所解释的那样，这种方案同样不能保证不可否认性。



（a）用对称密钥密码实现报文内容认证：机密性、完整性



（b）用公开密钥密码实现报文内容认证：机密性、完整性



（c）用公开密钥密码实现报文内容认证：不可否认性、完整性

图 3-3　用加密的方法实现报文内容认证

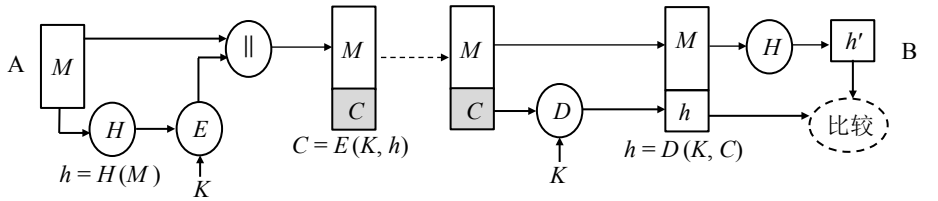
哈希函数属于典型的消息校验手段。除了使用哈希函数，还可以利用其他能够产生校验

信息的函数，比如通信中经常用来检测消息是在传输过程中发生了误码的循环冗余校验（Cyclic Redundancy Check, CRC）函数。这些函数的使用方法与哈希函数类似，发送方和接收方独立使用函数进行计算，并比较计算结果，根据结果是否相同判断消息的完整性。与哈希函数相比，校验和函数检测消息是否被修改的能力要弱一些。此外，为了防范攻击者同时修改消息以及消息的校验信息，发送方通常会加密消息的校验信息再进行传输。

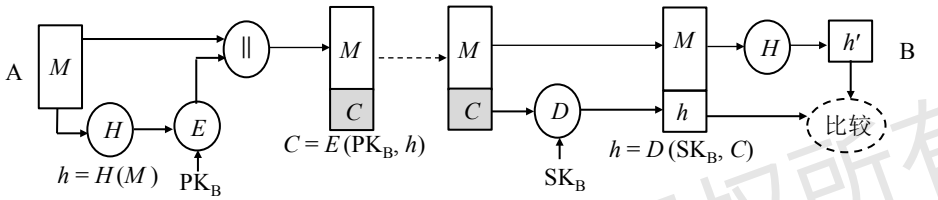
通信双方采用公开密钥密码系统认证报文内容的方法与采用对称密钥密码系统类似。消息发送方（A）利用接收方（B）的公钥 PK_B 加密消息以及消息的校验信息，接收方（B）通过自己的私钥 SK_B 解密后进行认证，计算消息的哈希值 h'，并与消息后面的哈希值 h 进行比较。如果相同，则说明消息没有被篡改，如图 3-3(b)所示。同样，这种方式可以保证消息传输的机密性和完整性。

消息发送方（A）还可以利用自己的私钥 SK_A 加密消息以及消息的校验信息，接收方（B）通过发送方的公钥 PK_A 解密后进行认证，计算消息的哈希值 h'，并与消息后面的哈希值 h 进行比较。如果相同，则说明消息没有被篡改，如图 3-3(c)所示。这种方式下，可以保证消息传输的不可否认性和完整性，但不能保证消息的机密性，因为发送方的公钥 PK_A 是公开的，任何了解其公钥的人均可解密消息。这种方式实现了对消息的数字签名和完整性保护。

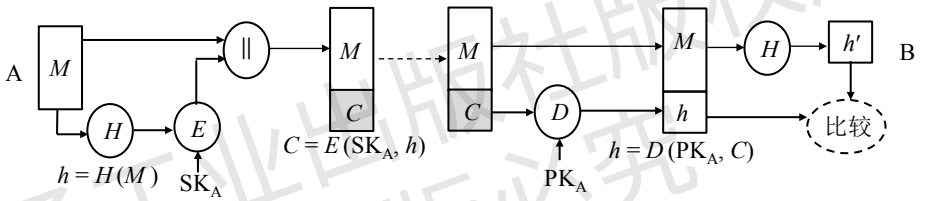
上述三种方式除了实现消息的完整性保护，还实现了其他安全功能，如不可否认性、机密性。如果只需要实现完整性保护，则发送方只需加密哈希值（h）即可，并将加密后的哈希值（C）拼接在明文 M 之后一起发送给接收方，接收方解密 C，计算消息 M 的散列值，并将它们进行比较就可验证消息的完整性。如图 3-4(a)和图 3-4(c)所示，攻击者由于缺少能够对



（a）用对称密钥密码实现报文内容认证：完整性



（b）用公开密钥密码实现报文内容认证：完整性？



（c）用公开密钥密码实现报文内容认证：完整性

图 3-4　用加密散列值的方法实现报文内容认证

完整性保护信息 C 进行修改的密钥，所以难以破坏消息的完整性。但在图 3-4(b)中，由于接收者的公钥 PK_B 是公开的且消息 M 没有加密，因此攻击者完全可以在截获消息后，对消息进行修改，重新计算修改后的消息的散列值，并用 PK_B 加密后发送给接收者。接收者是无法发现消息被修改过的，因此完整性无法保证。还有一个问题，图 3-4(c)是否可以如图 3-3(c)一样，可以实现不可否认性呢？答案是肯定的，因为除了 A，其他人因为不知道 SK_A，所以无法伪造出被加密保护的消息 M 的散列值，这也是下面将要介绍的最常见的不保证机密性的数字签名方法。

与图 3-3 所示的方法相比，图 3-4 的方法虽然不能保证机密性，但由于只对散列值加密，速度要快多了。

2. 消息认证码的方法

消息认证码（Message Authentication Code，MAC），也称为密码校验和（Cryptographic Checksum），是一种不依赖于加密技术实现报文内容完整性验证的方法。所有需要进行完整性验证的消息，将会以消息本身为基础，相应生成称为“消息认证码”的固定大小的数据块。这种消息认证码是对消息完整性进行验证的关键。

采用消息认证码的方法，通信双方需要共享一个密钥，以 K 来表示。同时，需要用于生成消息认证码的函数 F。对于消息 M，其消息认证码 MAC_M 与 M 和 K 相关，可以表示为：

$$MAC_M = F(M, K)$$

发送方将消息认证码 MAC_M 与消息 M 一起发给接收方。图 3-5 是利用消息认证码进行认证的过程。消息 M 在网络上传输，由于可能被篡改，因此在图中以 M'表示接收方接收到的消息。

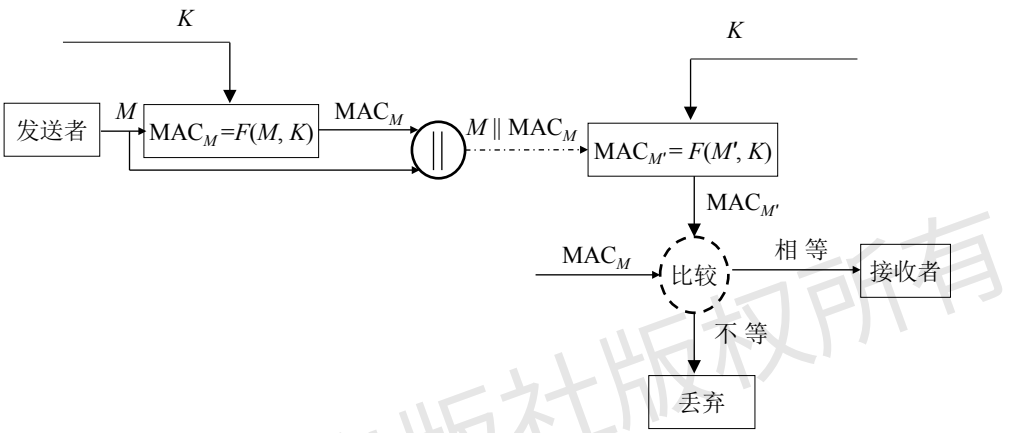


图 3-5　利用消息认证码进行认证

接收方在接收到消息 M'以后，利用与发送方共享的密钥 K，为消息生成消息认证码 $MAC_{M'} = F(M', K)$。如果 $MAC_{M'} = MAC_M$，则可以推断收到的消息 M'与发送方所发出的消息 M 相同。其原因在于 MAC_M 的计算涉及到收发双方共享的密钥 K，其他人不掌握该密钥，无

法对 MAC_M 进行有效篡改。如果消息 M 在传输过程中遭到篡改，那么接收方结合 M'和 K 计算得到的消息认证码 $MAC_{M'}$将不同于发送方发送来的 MAC_M。

有很多方法可以生成 MAC，NIST 标准 FIPS PUB 113 推荐使用 DES。用 DES 生成密文，将密文的最后若干个比特用作 MAC。典型的有 16 或 32 比特的 MAC。

如果函数 F 采用前面介绍的散列函数，如 MD5 或 SHA-1，则将这种消息认证码称为“散列消息认证码（Hashed Message Authentication Codes, HMAC）”。散列消息认证码是将消息 M 和密钥 K 拼接后作为散列函数 H 的输入进行散列运算后所产生的结果。

$$\mathrm{HMAC}_M = H(M \| K)$$

因为秘密值本身并不会发送，所以攻击者不可能修改所截获的消息。只要不泄露秘密值 K，攻击者就不可能伪造消息。

RFC 2104 中定义的 HMAC 算法，如下所述：

$$\mathrm{HMAC}_k(M) = H[(k^+ \oplus \mathrm{opad}) \| H(k^+ \oplus \mathrm{ipad}) \| M]$$

其中，

H：散列函数（MD5 或 SHA-1）。

M：HMAC 的消息输入。

k^+：密钥，长度不规则时左侧用 0 补齐，使其长度等于散列码块长（b，对 MD5 或 SHA-1 而言，其块长等于 512 比特）。

ipad：比特串 00110110 重复 b/8 次。

opad：比特串 01011100 重复 b/8 次。

简单来说，上述 HMAC 的主要过程如下：

（1）在 k 的左端追加 0 构成 b 比特的字符串 k^+（如 k 的长度为 160 比特，b = 512，k 将被追加 44 个 0 字节 0x00）。

（2）ipad 与 k^+进行按比特异或生成 b 比特的分组 S_i。

（3）将 M 追加在 S_i 上。

（4）将 H 应用步骤（3）所产生的数据流。

（5）opad 与 k^+进行异或生成 b 比特的分组 S_o。

（6）将步骤（4）产生的散列结果追加在 S_o 上。

（7）将 H 应用步骤（6）产生的数据流，输出结果。

上述 HMAC 方法被广泛应用于 IPsec、SSL/TLS、SET 等安全协议中，我们将在后续章节进行介绍。

消息认证码除了能够认证报文内容的完整性，也能够用于报文源认证和报文顺序认证。消息认证码的生成涉及了一个保密的密钥，只要在发送消息时增加发送方的标识号，将发送方的身份标识号 ID 与消息 M 拼接，并在生成消息认证码时包含此信息：$\mathrm{MAC}_M = F(M \| \mathrm{ID}, K)$，则消息认证码也能够准确判断发送方的身份。如果在生成消息认证码时包含能够表示报文先后顺序的信息（如时间戳、序列号等），而攻击者不能正确地修改认证码中所包含的顺序信息，则接收者就可以确认消息的正确序列，以避免重放攻击。

消息认证码方法与加密方法之间存在一些明显区别。首先，虽然两者都要求通信双方共

享密钥，但是消息认证码使用的密钥与密码系统中使用的密钥不同，其使用并不是为了控制加解密过程，而是利用这个共享密钥使得他人难以有效篡改消息认证码。其次，用于产生消息认证码的函数不需要具有可逆性，即生成消息认证码以后，不要求通过消息认证码恢复原始消息。而加密过程要求具有可逆性，对于加密得到的密文必须可以通过解密恢复成明文。因此，消息认证码的函数在设计上相对简单。

在图 3-5 中，消息认证码的使用并不能保证消息的保密性。如果攻击者在网络上监听，则能够获取传输的消息内容。如果要求确保消息的保密性，还需要对消息进行加密。

3.2.4 报文顺序的认证

报文的发送顺序在通信中具有重要的安全意义。重放攻击就是攻击者利用通信双方没有有效认证消息的发送顺序，将截获的消息在以后的时间重新发送，扰乱正常通信的一种攻击方式。举例来看，在战斗时我军的指挥员发送消息给甲，要求甲执行进攻任务。该消息被攻击者截获并保存，几天以后攻击者将消息再次发送给甲，甲如果没有对消息的发送顺序进行认证，则可能执行错误的命令。

对报文顺序进行认证，最重要的一点就是在通信中增加标识报文顺序的信息。报文顺序信息可以多种形式出现。

首先的一种形式是序列号。发送方为每条消息增加序列号，接收方根据序列号判断消息的发送顺序，并依据序列号及时发现消息的重复和乱序。重放攻击这种利用过时消息进行攻击的方法将无法成功。采用序列号的形式对报文顺序进行认证，很重要的一个条件是通信双方能准确记录通信的序列号信息，以确保通信按序进行。

第二种报文顺序信息的表现形式是时间戳。在发送的消息中增加时间戳，接收方接收消息时，可以根据时间戳判断消息是否按序到达。时间戳这种形式要求通信双方保持时间同步，以便能够以时间信息为依据判断消息的顺序。

采用挑战及响应（challenge/response）的认证方式可以根据通信需要进行认证，防范消息乱序导致的问题。采用这种方式，通信双方通常需要共享一个密钥，并在认证的过程中使用对称密钥密码算法进行加密。实施认证的一方产生不可预测的随机信息，这种随机信息被称为挑战（challenge）。随机信息发送到被认证方，被认证方在随机信息的基础上进行计算，并将计算结果作为响应返回。根据响应是否正确，实施认证的一方可以判断通信源的身份并防止重放攻击。

举例来看，如果甲向乙发送消息，其通信顺序如下：

（1）甲向乙发送消息，请求乙对自己进行认证。

（2）乙在接收到消息以后，随机产生一个字符串、整数或者其他类型的数据返回给甲，以 t 表示该随机信息。

（3）甲利用与乙共享的密钥 k 使用加密算法 E 加密 t，得到 $t_1 = E(k, t)$，并将 t_1 发送给乙。

（4）乙在收到 t_1 后，利用密钥 k 通过解密算法 D 解密 t_1，得到 $t_2 = D(k, t_1)$，如果 $t_2 = t$，则可以判断消息的发送者是甲。

以上过程可以看作甲向乙发送消息之前进行的握手。在此基础上，甲可以 t_1 为基础产生标识顺序信息的参数并加入到消息中，开始消息的发送。挑战及响应的认证方式由于必须先

完成认证再进行通信。因此，攻击者很难实施重放攻击。

比较三种顺序认证方式，采用序列号的方法需要维护具有全局性的序列号，每一方都要记住其他各方与其通信时的最后一个序列号，管理起来比较复杂。采用时间戳的方法要求保证通信主机间的时间同步。挑战及响应的认证方式在每次通信前握手，通过握手使双方确定通信状态，不必从始至终维护序列号、时间之类的状态信息，相对而言灵活性较强。

总体来看，报文顺序认证最重要的前提是确保攻击者无法修改消息的顺序信息，即顺序信息的完整性不受破坏。在实际应用中，常常将报文顺序的认证和报文内容的认证结合实现。最典型的一种方法是将顺序标识附加在消息上，同时利用报文内容的认证方法确保消息整体的完整性。

在上述顺序认证方案中，都有一个随时间变化的参数，一般将这个参数称为“现时（Nonce）”。Nonce 是 Number used once 或 Number once 的缩写，是一个随时间而改变的参数，在密码学中通常是一个只被使用一次的任意或非重复的随机数值。在计算机系统中，大部分程序设计语言中产生的随机数是由可确定的函数（比如线性同余），通过一个种子（如时钟）来产生随机数的。这意味着：如果知道了种子，或者已经产生的随机数，都可能获得接下来随机数序列的信息（可预测性）。因此，用这种方法产生的随机数通常称为“伪随机数（pseudorandom number）”。真随机数一般是用某些随机物理过程来产生的，如放射性衰变、电子设备的热噪音、宇宙射线的触发时间等。也可以通过软硬结合的方法来实现，通过不断收集非确定性的设备事件，如机器运行环境中产生的硬件噪音（如键盘敲击速度、鼠标移动速度、电磁波、磁盘写入速度等）作为种子来产生随机数。IETF 在 RFC 4086 中列出了一些可能的随机源，如声音/图像输入、磁盘驱动时产生的随机波动等，用于计算机来产生真随机数。在实际应用中，只有安全要求非常高的应用场景中，才使用真随机数发生器。

Nonce 在很多加密方法的初始向量和加密散列函数中发挥着重要作用（后续章节将会介绍），在各类认证协议中被用来确保认证信息不被重复使用以对抗重放攻击。此外，Nonce 也用于流密码加密过程中，如果需要使用相同的密钥加密一个以上的消息，就需要 Nonce 来确保不同的消息与该密钥加密的密钥流不同。

3.3　数字签名

前面介绍的报文认证可以保护信息交换双方不受第三方的攻击，但是它不能处理通信双方自己产生的攻击，例如：假定小张使用如图 3-4(a)所示的方法给小王发送一条消息，考虑下面的两种情形：

（1）小王可以伪造一条消息并称该消息来自小张。小王只需产生一条消息，用小张和小王共享的密钥对消息的散列码加密并附在消息的后面即可。

（2）小张可以否认曾发送过某条消息。因为小王可以用他们间共享的密钥伪造消息，所以无法证明小张确实发送过该消息。

上面的两种情况都是法律上比较关注的，特别是在涉及金融交易时。在收发双方不能完全相互信任的情况下，就需要除报文认证之外的其他方法来解决这些问题。

在以书面文件为基础的传统事务处理过程中，常常以手写签名的形式验证用户身份，手

写签名具有法律意义。以数据文件为基础的现代事务处理过程需要采用电子形式的签名认证用户身份，这种签名方式被称为数字签名（digital signature），也被称为电子签名。我国在 2005 年 4 月 1 日开始施行《电子签名法》，该法律确立了数字签名在我国的法律效力。

3.3.1 数字签名的基本概念

数字签名以密码技术为基础，其安全性取决于签名所使用的密码系统的安全程度。目前，数字签名主要通过公开密钥密码系统实现，此类数字签名可以看作公开密钥密码系统加密过程的颠倒使用。签名者使用自己的私钥处理文件，完成签名，其他用户用签名者的相应公钥验证文件，确定签名的真伪。签名过程利用了公开密钥密码系统的特性，只有签名者本人知道自己的私钥，他人无法通过签名者的公钥破解签名者的私钥，因此可以以私钥标识签名者的身份。信息如果是通过某个用户的私钥处理而得到的，则信息必定是由该用户所处理的，并愿意承担相应的责任。

完善的数字签名体制需要满足以下条件：

（1）签名不能伪造。签名是签名者对文件内容认同的证明，其他人无法对签名进行伪造。

（2）签名不可抵赖。这是对签名者的约束，签名者对文件施行签名以后，不能否认自己的签名行为。

（3）文件在签名后不可改变。在签名者对文件签名以后其他人不能再修改文件内容。

（4）签名不可重复使用。可以采用增加时间标记或者序号标记的方法，防止签名被攻击者重复使用。

（5）签名容易验证。对于签名的文件，一旦发生签名真伪性方面的纠纷，任何第三方都可以准确、有效地进行仲裁。

数字签名体制包括两方面的处理，一方面是签名者施加签名，另一方面是用户验证签名。假设签名者 A 的私钥为 SK_A。以 SIG 表示施加签名的算法，以 m 表示被签名的数据，以 s 表示产生的签名信息。A 使用自己的私钥 SK_A 对数据签名，签名过程可以描述为 $SIG(SK_A, m) = s$。

验证签名的算法以 VER 表示，用以鉴别特定的签名 s 是否的确由声称的签名者 A 产生。验证需要使用 A 的公钥 PK_A，验证过程可以描述为 $VER(PK_A, s)$。在验证过程中，如果可以通过 PK_A 从签名信息 s 中恢复被签名的数据 m，或者其他能够标识 m 的信息，则可推断数据 m 源于用户 A。

对传统的书面文件进行手写签名，签名后的文件包括两部分信息，首先是文件的主体内容，其次是签名者的手写签名。由于纸张的特性，一旦出现对纸张上信息内容的涂改、拼凑，其他人很容易发现异常。因此，攻击者难以在他人完成手写签名后，修改文件或者将手写签名与其他文件拼接。数字签名同样需要保证数据文件的完整性，防止攻击者篡改签名数据。

利用公开密钥密码系统进行数字签名，由于私钥为签名者所有，因此其他人难以伪造，签名者也无法否认自己的签名。同时，由于公钥完全公开，任何人都能够验证签名者的签名。为了防止在数字签名后，被签名的数据遭到修改，需要为签名算法增加一项限制条件：如果 $m_1 \neq m_2$，要求 $SIG(SK_A, m_1) \neq SIG(SK_A, m_2)$。

该限制条件可以理解为只要数据内容不同，签名信息就应当不同，从而避免攻击者采用

张冠李戴的手法，将签名者对某段数据的签名移植到其他数据上。此限制条件的满足要求签名算法产生的签名与被签名的数据长度相同，而在实际应用中为了节省存储开销都会希望签名能够短一些。因此，数字签名通常采用相对而言较为宽松的一个限制条件：即使存在 $m_1 \neq m_2$，$\mathrm{SIG}(\mathrm{SK_A}, m_1) = \mathrm{SIG}(\mathrm{SK_A}, m_2)$，但对于给定的数据 m_1 和签名 $\mathrm{SIG}(\mathrm{SK_A}, m_1)$，攻击者无法通过计算找到相应的 m_2。通过这样的限制条件为被签名数据的完整性提供保证。

在介绍公开密钥密码系统时已经提到，以 PK 表示公钥，SK 表示对应的私钥，E 表示加密算法，D 表示解密算法，m 表示任意的明文，如果 $D(\mathrm{PK}, E(\mathrm{SK}, m)) = m$，则该公开密钥密码系统能够用于认证数据发送方的身份，即可以用于数字签名。1994 年，美国国家标准与技术学会基于 ElGamal 公开密钥密码系统制定了数字签名标准 DSS（Digital Signature Standard）。2000 年，RSA 公开密钥密码系统被扩充到 DSS 中实现数字签名。此外，椭圆曲线密码等很多公开密钥密码系统都可以用于实现数字签名。下面首先介绍基于 RSA 的数字签名方法。

3.3.2　利用 RSA 密码系统实现数字签名

首先，验证 RSA 公开密钥密码系统是否能够应用于数字签名，即条件 $E(\mathrm{PK}, D(\mathrm{SK}, m)) = m$ 是否满足。RSA 公钥系统中用户的公钥 PK 以$\{e, n\}$表示，与之对应的私钥 SK 以$\{d\}$表示，采用 m 表示明文，c 为相应的密文，RSA 系统的加密过程可以表示为：

$$c = E(\mathrm{PK}, m) = m^e \bmod n$$

相应的解密过程表示为：

$$m = D(\mathrm{SK}, c) = c^d \bmod n$$

作为一种密码系统，RSA 首先必然满足 $D(\mathrm{SK}, E(\mathrm{PK}, m)) = m$，即明文通过公钥加密后，可以由对应的私钥解密。要判断条件 $E(\mathrm{PK}, D(\mathrm{SK}, m)) = m$ 是否满足，可以逐步分析：

$$E(\mathrm{PK}, D(\mathrm{SK}, m)) = (m^d)^e \bmod n = (m^e)^d \bmod n = D(\mathrm{SK}, \mathrm{E}(\mathrm{PK}, m)) = m$$

由上式可知 $E(\mathrm{PK}, D(\mathrm{SK}, m)) = m$，即 RSA 公开密钥密码系统符合数字签名的条件，能够应用于数字签名。

基于 RSA 算法进行数字签名时，如果需要签名的消息是 m，签名者的私钥 SK 为$\{d\}$，则签名者施加签名的过程可以描述为 $s = D(\mathrm{SK}, m) = m^d \bmod n$，其中 s 为签名者对 m 的签名。

与签名过程相对应，验证签名需要用到签名者的公钥 PK，以$\{e, n\}$表示签名者的公钥，在获得签名信息 s 后，验证签名的过程可以描述为：

$$m' = E(\mathrm{PK}, s) = s^{\mathrm{e}} \bmod n$$

如果 $m' = m$，则可以判断签名 s 的签名者身份真实。

举一个简单的实例来看，用户张三的公钥为{79, 3337}，对应的私钥为{1019}。张三想把消息 72 发送给其他用户，在发送前他用自己的私钥对消息进行签名：$s = 72^{1019} \bmod 3337 = 356$。其他用户在接收到消息和签名以后，可以通过张三的公钥对签名进行验证从而确定消息是否

的确由张三发出：$m' = 356^{79} \bmod 3337 = 72$，计算所得到的结果与接收到的消息 72 一致，可以确定消息是由张三发出的，而且消息在传输过程中没有被修改过。

下面对使用 RSA 进行数字签名与使用 RSA 进行数据加密进行比较。首先，两者的目的截然不同。加密的目的主要是确保消息的保密性，防范未授权用户获取消息内容。数字签名的目的是对消息发送方的身份进行认证，发送方在对消息进行数字签名后将无法否认发送的消息，此外，数字签名还能为消息的完整性提供有力保证。其次，两者在密钥的使用上存在明显区别。如果需要在网络上传送加密的消息，发送方使用接收方的公钥加密消息，通过网络传送密文，接收方使用自己的私钥进行解密操作，将密文恢复为明文。在数字签名时，发送方使用自己的私钥对消息进行签名，接收方以及所有其他用户可以通过发送方的公钥对签名进行验证，确定发送者的身份。

数字签名无法保证消息的保密性，因为公钥完全公开，所有用户都可以通过签名者的公钥恢复签名的信息。如果不仅要求认证消息的发送方，而且需要确保消息的保密性，可以对数字签名进行加密。发送方首先使用自己的私钥对消息进行签名，再使用接收方的公钥对数字签名进行加密，将加密结果通过网络发送。接收方在接收后，先用自己的私钥进行解密，恢复数字签名，而后利用发送方的公钥验证签名。采用这种发送方先签名后加密、接收方先解密后验证的方法，可以认证发送方的身份并同时确保消息的保密性。

3.3.3 利用 ElGamal 密码系统实现数字签名

第 2 章介绍了 ElGamal 密码系统在加密方面的应用，下面介绍如何用 ElGamal 实现数字签名。

选择一个素数 q，获取素数 q 的一个原根 α，将 q 和 α 公开。生成一个随机数 x 作为其秘密的解密密钥，使得 $1 \leqslant x \leqslant q-1$，计算 $y = \alpha^x \bmod q$，则公钥为 (y, α, q)，私钥是 x。

1. 生成签名

设用户 A 要对明文消息 m 进行签名，$1 \leqslant m \leqslant q-1$，签名过程如下：

（1）用户 A 随机选择一个随机数 k，$1 \leqslant k \leqslant q-1$。

（2）计算 $r = \alpha^k \bmod q$。

（3）计算 $s = (m - x \times r)k^{-1} \bmod (q\text{-}1)$。

（4）以(r, s)作为 m 的签名，并将（m, r, s）发给用户 B。

2. 验证签名

用户 B 收到（m, r, s）后的验证过程如下：

如果$(\alpha^m \bmod q) = (y^r \times r^s) \bmod q$ 成立，则验证通过。

为了安全，随机数 x 必须是一次性的。由于取(r, s)作为 m 的签名，所以 ElGamal 数字签名的数据长度是明文的两倍。另外，ElGamal 数字签名需要使用随机数 k，这就要求在实际应用中要有高质量的随机数生成器。

3.3.4　利用椭圆曲线密码实现数字签名

利用第 2 章介绍的椭圆曲线密码可以方便地实现 3.3.3 节的 ElGamal 数字签名，下面给出具体的签名过程。

在 SEC1 椭圆曲线密码标准（草案）中规定，一个椭圆曲线密码由一个六元组来定义：

$$T = <p, a, b, G, n, h>$$

其中，p 为大于 3 的素数，它确定了有限域 GF (p)；元素 $a, b \in$GF(p)，它确定了椭圆曲线；G 为循环子群 E_1 的生成元，n 为素数且为生成元 G 的阶，G 和 n 确定了循环子群 E_1，h 表示椭圆曲线群的阶与 n 的商。

1. 生成签名

（1）选择一个随机数 k，$k \in \{1, 2, \ldots, n - 1\}$。

（2）计算点 $R(x_R, y_R) = kG$，记 $r = x_R$。

（3）利用私钥 d 计算 $s = (m - d \times r)\, k^{-1} \bmod n$。

（4）以(r, s)作为 m 的签名，并将（m, r, s）发给用户 B。

2. 验证签名

用户 B 收到（m, r, s）后的验证过程如下：

（1）计算 $s^{-1} \bmod n$。

（2）利用公开的加密密钥 Q 计算 $U(x_U, y_U) = s^{-1}(mG - rQ)$。

（3）如果 $x_U = r$, 则<r, s>是用户 A 对 m 的签名。

3.3.5　散列函数在数字签名中的作用

前面介绍的数字签名方案针对完整的消息实施签名，这种方案存在两方面的缺陷。首先，如果消息内容长，则相应的数字签名也将很长。而实际应用中除了需要保留原始的消息内容，还要保存消息的数字签名以备随时验证，这将导致高昂的存储开销。其次，公开密钥密码系统存在处理速度慢、计算开销高昂的缺点，对长消息进行数字签名以及进行签名验证都需要付出高昂的计算代价，同时耗时冗长，实际应用难以接受。

考虑到以上因素，数字签名通常不是针对完整的消息进行的，而是针对能够标识消息的特征信息施行的。被签名的特征信息需要满足一些条件。首先，特征信息不能太长，否则签名的计算开销仍然较高。其次，当信息内容发生改变时，特征信息必须同时改变，为签名数据的完整性验证提供依据。

在实际的数字签名方案中，数字签名往往针对消息的散列值进行。发送方在发送消息前，首先计算消息的散列值。而后，发送方使用自己的私钥对消息的散列值进行数字签名。消息和数字签名一起被发送到接收方。在接收到消息以后，接收方一方面计算接收到的消息的散列值。另一方面，利用发送方的公钥对发送来的数字签名进行验证。如果计算得到的散列值与从数字签名中提取的散列值相同，那么可以判断消息由所声称的发送者发出，同时消息在传输的过程中没有被篡改。采用这种数字签名方案，数字签名和签名验证都是针对消息的散

列值进行，这样计算开销和存储开销都降低了。

例如，Internet 保密电子邮件系统 PGP（参见第 10 章）首先利用 MD5 散列函数对邮件正文进行散列处理，然后使用 RSA 对邮件正文的散列值进行数字签名，以确保真实性。

3.4 身份认证

对信息系统进行安全防护，常常需要正确识别与检查用户的身份，即身份认证 [1]。身份认证这种认证形式可以将非授权用户屏蔽在系统之外，它是信息系统的第一道安全防线，其防护意义主要体现在两方面。首先，防止攻击者轻易进入系统，在系统中收集信息或者进行各类攻击尝试。其次，有利于确保系统的可用性不受破坏。信息系统的资源都是有限的，非授权用户进入系统将消耗系统资源，如果系统资源被耗尽，正常的系统用户将无法获得服务。

身份认证的本质是由被认证方提供标识自己身份的信息，信息系统对所提供的信息进行验证从而判断被认证方是否是其声称的用户。具体来看，身份认证涉及到识别和验证两方面的内容。所谓识别，指的是系统需要确定被认证方是谁，即被认证方对应于系统中的哪个用户。为了达到此目的，系统必须能够有效区分各个用户。一般而言，被用于识别用户身份的参数在系统中是唯一的，不同用户使用相同的识别参数将使得系统无法区分。最典型的识别参数是用户名，像电子邮件系统、BBS 系统这类常见的网络应用系统都是以用户名标识用户身份的。而网上银行、即时通信软件系统常常以数字组成的账号、身份证号、手机号码作为用户身份的标识。验证则是在被认证方提供自己的身份识别参数以后，系统进行判断，确定被认证方是否对应于所声称的用户，防止身份假冒。验证过程一般需要用户输入验证参数，同身份标识一起由系统进行检验。

身份认证可以基于以下四种与用户有关的内容之一或它们的组合实现：

（1）所知。个人所知道的或所掌握的知识或信息，如密码、口令。

（2）所有。个人所具有的东西，如身份证、护照、信用卡、智能门卡等。

（3）所在。个人所用计算机的 IP 地址、办公室地址等。

（4）用户特征。主要是个人生物特征，如指纹、笔迹、声纹、手形、脸形、视网膜、虹膜、DNA，还有个人的一些行为特征，如走路姿态、击键动作、笔迹等。

目前，身份认证技术主要包括口令认证、信物认证、地址认证、用户特征认证和密码学认证。

1. 口令认证

口令（或密码）认证是最典型的基于用户所知的验证方式。系统为每一个合法用户建立用户名和口令的对应关系，当用户登录系统或者执行需要认证身份的操作时，系统提示用户输入用户名和口令，并对用户输入的信息与系统中存储的信息进行比较，以判断用户是否是其所声称的用户。

1 与身份认证中的“认证”密切相关的一个概念是“授权”。授权是指确定了身份（即经过了“认证”）的用户能够以何种方式访问何种资源的过程，是实现访问控制的一种重要实现方式。注意这两个概念的英文的差别，认证是 authentication，授权是 authorization。

口令认证简单、易于实施，应用非常广泛。但其存在很多缺点，以普通的网络应用系统为例，用户通过客户端向服务器发送用户名、口令信息进行身份认证，在客户端、通信链路以及服务器三处都有口令泄露的可能。具体来看，用户在使用客户端主机时，口令输入过程可能被其他人偷窥。此外，如果用户使用的客户端感染了盗号木马，木马可能采取键盘记录、屏幕截取等方式获取用户输入的账号、口令（密码）。在通信链路上，如果口令以明文传输，黑客采用网络监听工具就能对通信内容进行监视，可以轻易获取传输的用户名和口令信息。此外，在服务器端，如果服务器存在漏洞，黑客获取权限后，可以盗取存储口令信息的文件进行破解，获得用户口令。假若以上三方面的防护都很完善，但用户一旦使用的是比较简单的口令，则黑客可以很容易地猜解出来。

2. 信物认证

信物认证是典型的基于用户所有的验证方式，它通常采用特定的信物标识用户身份，所使用的信物通常是磁卡或者各种类型的智能存储卡。拥有信物的人被认定为信物对应的用户。信物认证方式一般需要专门的硬件设备对信物进行识别和判断，其优点是不需要用户输入信息，用户使用方便。这种认证方式的难点是必须保证信物的物理安全，防止遗失被盗等情况。如果信物落入其他人手中，其他人可以以信物所有人的身份通过验证进入系统。

3. 地址认证

地址认证是基于用户所在地址的一种认证方式。以 IP 地址为基础进行认证是使用最多的一种地址认证方式。系统根据访问者的源地址判断是否允许其访问或者完成其他操作。例如，在 Linux 环境下，可以在配置文件.rhost 中添加主机所信任主机的 IP 地址，然后通过这些 IP 地址访问主机就可以直接进入系统。此外，互联网上很多下载站点限定只有指定 IP 地址范围的主机允许下载资源，如一些大学网站的教学资源只允许本校 IP 地址范围的主机访问。这种基于用户所在的认证方式，其优点是对用户透明，用户使用授权的地址访问系统，可以直接获得相应权限。缺点是 IP 地址的伪造非常容易，攻击者可以采用这种方式轻易进入系统。

4. 用户特征认证

用户特征认证主要利用个人的生物特征和行为特征进行身份认证。指纹认证、人脸识别、虹膜扫描、语音识别都是较为常见的基于用户特征的认证方式。以使用广泛的指纹认证为例，每个人的指纹各不相同，采用这种验证方式的信息系统，必须首先收集用户的指纹信息并存储于专门的指纹库中。用户登录时，通过指纹扫描设备输入自己的指纹，系统将用户提供的指纹与指纹库中的指纹进行匹配，如果匹配成功，则允许用户以相应身份登录，否则用户的访问将被拒绝。

用户行为特征如果具有很强的区分度也可以被用于验证。例如，每个人的手写笔迹都不相同，手写签名在日常生活中被广泛用于标识用户身份。如果为信息系统增加专用的手写识别设备，也可以利用手写签名验证用户身份。每个用户键盘输入的速度、击键力量等均存在差异，可以利用击键模式作为用户认证的手段。

用户特征认证方式很难保证百分之百可靠，通常存在两种威胁。首先，系统可能由于特征判断的不准确，将非授权用户判定为正常用户接纳到系统中。其次，可能由于用户特征发

生变化，如手指受伤导致指纹发生变化，天太冷导致击键比平常慢等，或者由于判定条件的不同，如采用人脸识别的验证方法，光线的不同、角度的差异以及表情的变化都有可能使系统将授权用户判定为非法用户。

5. 密码学认证

密码学认证主要利用基于密码技术的用户认证协议进行用户身份的认证。协议规定了通信双方为了进行身份认证甚至建立会话密钥所需要进行交换的消息格式或次序。这些协议需要能够抵抗口令猜测、地址假冒、中间人攻击、重放攻击等。

常用的密码学认证协议有一次性口令认证、基于共享密钥的认证、基于公钥证书的认证、零知识证明和标识认证等。基于公钥证书的认证引入可信第三方认证机构（CA），以解决公钥认证时公钥的可靠获取问题，我们将在第 4 章介绍这一技术。零知识证明和标识认证读者可参考文献[21]。本节主要介绍其中的一次性口令认证协议 S/KEY、基于共享密钥的认证技术。

3.4.1 一次性口令认证

很多网络应用使用口令进行用户认证，这种认证方式最大问题是如果口令设置很简单，或者不经常更改的话，甚至以明文形式在网络上传输，则很容易被人破解或截获下来进行重放攻击。因此，网络环境下的口令认证方式不能使用长期不变的静态口令，而应该在每次登录过程中加入不确定因素，使得每次登录的口令均不同，这就是一次性口令（One-Time Password, OTP）。

一次性口令一般使用双运算因子来实现：固定因子，即用户的口令或口令散列值；动态因子，每次不一样的因子，如时间，把流逝的时间作为变动因子，用户密码产生器和认证服务器产生的密码在时间上必须同步；事件序列，把变动的数字序列作为密码产生器的一个运算因子，加上用户的口令或口令散列值一起产生动态密码；挑战/应答，由认证服务器产生的随机数字序列（challenge）作为变动因子，不会重复，也不需要同步。

比较著名的一次性口令认证系统是 1991 年贝尔通信研究中心研制的挑战/应答式（challenge/response）动态密码身份认证系统 S/KEY，其一次性口令生成原理如图 3-6 所示。

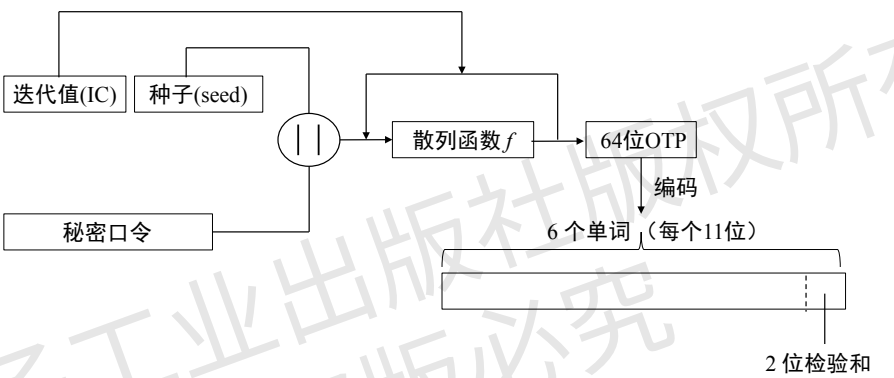


图 3-6 S/KEY 一次性口令生成原理

在 S/KEY 中，服务器产生挑战（challenge）信息。挑战信息由迭代值（Iteration Count, IC）和种子（seed）组成。迭代值，指定散列计算的迭代次数，为 1~100 之间的数，每执行一次挑战/响应过程，IC 减 1（当 IC 为 1 时，则必须重新进行初始化）。种子由两个字母和 5 个数字组成。例如，挑战信息“05 xa13783”表示迭代值为 05，种子为“xa13783”。客户端收到挑战后，要将秘密口令与种子“xa13783”拼接后，做 5 次散列运算。

在 S/KEY 中支持三种散列函数，即 MD4、MD5 和 SHA。OTP 服务器将散列函数的固定输出折叠成 64 位（OTP 的长度）。64 位 OTP 可以被转换为一个由 6 个英文单词组成的短语，每个单词由 1~4 个字母组成，被编码成 11 位，6 个单词共 66 位，其中最后 2 位（11*6－64 = 2）用于存储校验和。校验和的计算方法是：OTP 的 64 位被分解成许多位对，将这些位对进行求和，和的最低 2 位即为校验和。所有的 OTP 产生器必须计算出校验和，所有 OTP 服务器也必须能将校验和作为 OTP 的一部分进行校验。

在初始化阶段，认证服务器选取一个口令 pwd（由种子和用户的秘密口令拼接而成）和一个数 n（也就是 IC），以及一个散列函数 f，计算 $y = f^n(\text{pwd})$，并把 y（即用户的首个 OTP）和 n 的值存储在服务器上。初始登录时，服务器收到客户端的连接请求后，将 seed 和(n-1)作为挑战信息发送给客户端。客户端收到挑战信息后，计算 $y' = f^{(n-1)}(\text{pwd})$，并将 y'作为响应发送给服务器。服务器收到后，计算 $z = f^n(y')$。如果 z 等于服务器上保存的 y，则验证成功，然后将 y 的值替代成 y'，将 n 减 1。下次登录时，客户端计算 $y'' = f^{(n-1-1)}(\text{pwd})$，以此类推，直到 n 等于 1。当 n 等于 1 时，客户和服务器必须重新进行初始化。

下面我们来分析一下 S/KEY 的安全性。

在 S/KEY 中，用户的秘密口令没有在网络上传输，传输的只是一次性口令，并且一次性口令即使在传输过程中被窃取，也不能再次使用；客户端和服务器存储的是用户秘密口令的散列值，即使客户端和服务器被攻陷导致口令散列值被窃取，也需破解口令散列才能获得明文口令。因此，该方案有比较好的安全性。同时，该方案实现简单，成本不高，用户使用方便。由于使用散列函数计算一次性口令，因此，S/KEY 的安全性与散列函数的安全性密切相关。如前所述，S/KEY 使用的散列函数 MD4、MD5 和 SHA-1 都已不再安全。

S/KEY 也存在一些不足，主要包括：

（1）用户登录一定次数后，客户和服务器必须重新初始化口令序列。

（2）为了防止重放攻击，系统认证服务器具有唯一性，不适合分布式认证。

（3）S/KEY 是单向认证（即服务器对客户端进行认证），不能保证认证服务器的真实性。

（4）S/KEY 使用的种子和迭代值采用明文传输，攻击者可以利用小数攻击来获取一系列口令冒充合法用户。攻击的基本原理是：① 当用户向服务器请求认证时，攻击者截取服务器传给用户的种子和迭代值，并将迭代值改为较小的值；② 然后，假冒服务器，将得到的种子和较小的迭代值发送给用户；③ 用户利用种子和迭代值计算一次性口令，发送给服务器；④ 攻击者再次截取用户传过来的一次性口令，并利用已知的散列函数依次计算较大迭代值的一次性口令，就可获得该用户后续的一系列口令，从而在一段时间内冒充合法用户而不被发现。

在 S/KEY 中，用户口令散列在网络中传输，增加了被攻击和破解的风险。为了解决这一问题，研究人员提出了改进的 S/KEY 协议。主要思想是：使用用户的口令散列对挑战进行散列计算，并将计算结果发送给服务器。服务器收到后，同样使用服务器保存的用户口令散列

对挑战进行散列计算，并与客户端发来的应答进行比较，如果相同则认证通过，否则拒绝。认证通过后，服务器生成一个随机数作为后续会话的对称加密密钥，并用用户的口令散列加密后发送给客户端。后续的通信就可用该密钥加密，保障会话的机密性。方案中，一次性口令散列不会在网络上传输，降低了被截获的风险。Windows 2000 及其之后版本中的 NTLM 认证所实现的挑战/响应机制就使用了这个改进的 S/KEY 协议。

3.4.2 基于共享密钥的认证

基于共享密钥的认证的基本要求是示证者和验证者共享密钥（通常是对称密码体制下的对称密钥）。对于只有少量用户的系统，每个用户预先分配的密钥的数量不多，共享还比较容易实现。但是，如果系统规模较大，通常需要一个可信第三方作为在线密钥分配器。国际标准化组织（ISO）和国际电子协议（IEC）分别定义了几个不需要可信第三方的认证方案，读者可参考文献[21]。本节主要介绍两个基于可信第三方的共享密钥认证方案。

1. Needham–Schroeder 双向鉴别协议

下面我们来介绍著名的 Needham-Schroeder 双向鉴别协议（简称 N-S 协议），它综合利用了前面介绍的报文源认证、报文宿认证、报文内容认证、报文顺序认证技术，实现双向身份认证及密钥分配功能，后来很多鉴别协议（如 Kerberos）都是基于 N-S 协议的。

N-S 协议假定系统中有一个通信双方都信任的密钥分配中心（Key Distribution Center, KDC），负责双方通信会话密钥 K_s 的产生和分发。为了分配会话密钥，还必须有用于保护会话密钥的由通信双方和 KDC 共享的主密钥：K_a 和 K_b。主密钥通过带外方法分发，由于只用于保护会话密钥的分发，使用次数少，暴露机会少，因此只需定期更换。会话密钥 K_s 由 KDC 产生（每次不同），用主密钥保护分发，用于保护消息本身的传输，加密报文的数量多，但只使用有限时间，下次会话需重新申请。

N-S 协议过程如图 3-7 所示。

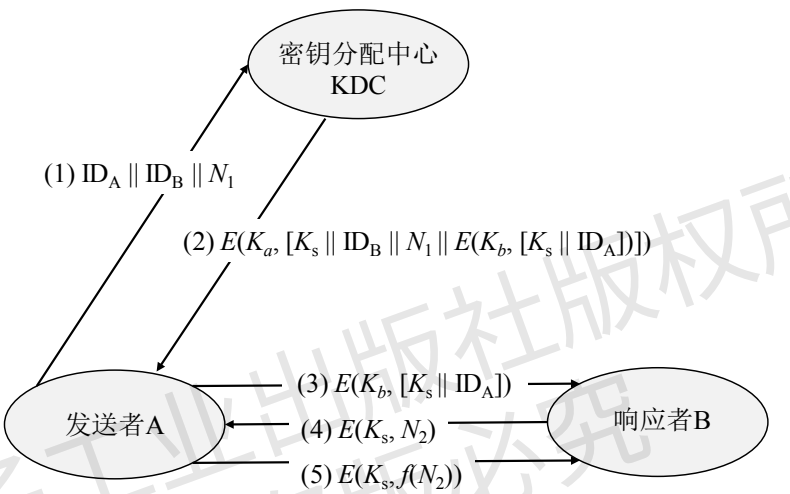


图 3-7 N-S 协议过程

具体步骤如下：

（1）A 向密钥分配中心申请与 B 通信的会话密钥 K_s，请求中带上自己的身份标识符（ID_A）和 B 的身份标志符（ID_B）以及一个现时值（N_1）。

（2）KDC 产生会话密钥 K_s，并用 A 的主密钥 K_a 对会话密钥分配消息（包括会话密钥 K_s、B 的 ID 号 ID_B、现时值 N_1、用 B 的主密钥 K_b 加密地发送给 B 的会话密钥信息）进行加密后发送给 A。由于消息是用 K_a 加密的，因此只有 A 能成功解密消息，并且 A 知道该消息是由 KDC 发来的。A 用 K_a 解密消息后，获得会话密钥 K_s，并通过 N_1 来判断响应是不是重放的。

（3）A 从会话密钥分配消息中取出 KDC 给其通信对象 B 的会话密钥分配信息 $E(K_b, [K_s \| ID_A])$。由于消息用 K_b 加密了，所以可以防止窃听。B 收到后，用自己的主密钥 K_b 解密消息，得到会话密钥密钥 K_s。由于消息只能被 B 解密，因此 A 可以证实对方是 B。解密后报文中的 ID_A 使得 B 证实该会话密钥用于与 A 的通信。

执行完上述三个步骤后，A 和 B 已得到了由 KDC 分配的一次性会话密钥，可用于后续的保密通信。但为了应对可能的重放攻击，还需执行以下两个步骤。

（4）B 产生现时值 N_2，并用得到的会话密钥 K_s 加密，将密文发送 A。用户 A 收到后用 K_s 解密，得到现时值 N_2。此步骤说明 B 已获得 K_s。

（5）A 用转换函数 f 对 N_2 进行处理后，用 K_s 对 $f(N_2)$加密，发送给 B。B 收到后，解密消息，并还原出 N_2。此步骤使 B 确信 A 也知道 K_s，现时 $f(N_2)$使 B 确信这是一条新的消息，而不是重放的。函数 f 通常是散列函数。

增加步骤（4）和（5）主要目的是防止攻击者截获步骤（3）中的报文并直接重放。尽管如此，上述 N-S 协议仍然有可能受到重放攻击。假设攻击者 X 得到了之前的会话密钥，虽然这一假设要比攻击者简单地观察和记录步骤（3）更难发生，但可能性是存在的，除非 B 无限期保存了所有之前和A会话时使用过的会话密钥，否则B就不确定下述过程是重放攻击：如果 X 能够截获步骤（4）的握手消息，他就能伪造步骤（5）中 A 的回复并将其发送给 B，而 B 却认为该消息来自于 A 且用已认证的会话密钥加密。为了解决这一问题，Denning 提出了改进措施，在步骤（2）和步骤（3）中添加时间戳来防止这种攻击，但这种方案需要实现网络中所有节点的时钟是同步的。详细情况读者可参考文献[16]的 15.2.1 节。

2. Kerberos 认证协议

Kerberos[1]认证协议以N-S协议为基础，通过可信第三方进行客户和服务器间的相互认证，交换会话密钥，以建立客户和服务器间的信任和安全传输信道。Kerberos由美国麻省理工学院（MIT）首先提出并实现，是该校雅典娜计划的一部分。Kerberos共有五个主要版本，其中第 1 版到第 3 版主要由该校内部使用，第 4 版在MIT校外得到了广泛应用，后来发展到了第 5 版（V5）。第 5 版由John Kohl和Clifford Neuman设计，于 1993 年作为IETF标准草案颁布（RFC 1510），后经多次修订（RFC 1964，RFC 4120，RFC 4121，RFC 4757），最新文档是 2012 年 7 月颁布的RFC 6649。Windows从Windows 2000 就开始支持基于Kerberos的认证协议。

Kerberos 协议包含很多子协议，提供认证功能的主要有三个：

1　Kerberos 的名字源于古希腊神话故事。Greek Kerberos 是希腊神话故事中一种长有 3 个头的狗，负责守卫地狱之门。Kerberos 协议意指提供“认证（Authentication）”、“授权（Authorization）”、“审计（Audit）”等 3 种功能的 Kerberos 是由 3 个部分（客户、服务器、KDC，相当于 3 个“头”）组成的网络之门的守卫者。

（1）认证服务交换（Authentication Service Exchange）协议。在客户（Client）和认证服务器（Authentication Server, AS）之间进行，客户向认证服务器发出认证请求。

（2）票证授予服务交换（Ticket Granting Service Exchange）协议。在完成 AS 交换后，在客户和票证授予服务器（Ticket Granting Server, TGS）之间进行，客户获得访问应用服务器的票证或许可证（Ticket Granting Ticket, TGT）。

（3）客户-服务器认证交换（Client/Server Exchange）协议。完成 TGS 交换后，客户使用获得的票证和应用服务器（Service Server, SS）或服务器（Server, S）进行交互。

在 Kerberos 中，密钥分配中心（KDC）由认证服务器（AS）和票证授予服务器（TGS）组成。将 KDC 分为两个子服务器的主要目的是为了方便实现用户的单点登录（Single Sign On, SSO）。单点登录主要发生在一个 Kerberos 域的用户访问其他 Kerberos 域的应用服务器的情况下，跨域之间的访问需要不同域的 TGS 预先建立信任。用户通过单个 AS 完成登录后，就能够获得多个 TGS 提供的服务，进而获得多个应用服务器提供的服务。

Kerberos 协议中相互认证或请求服务的实体被称为委托人（principal）。委托人是一个具有唯一标识的实体，可以是一台计算机或一项服务，通过使用 KDC 颁发的票证来进行通信。委托人可以分为两类：用户和服务，分别具有不同种类的标识符。用户通过如“user@REALM”格式的用户主体名称（User Principal Name, UPN）来标识。一般来说，名称中的域名用大写，例如用户“bob”在“bhusa.com”域中应该表示为 bob@BHUSA.COM。服务主体名称（Service Principal Name, SPN）是用于域中的服务和计算机账户。SPN 的格式形如“serviceclass/host_port/serviceName”。例如，主机“dc1.bhusa.com”上 LDAP 服务的 SPN 可能类似于“ldap/dc1.bhusa.com”，“ldap/dc1”和“ldap/dc1.bhusa.com/bhusa.com”。一个服务可能注册为多个 SPN。通常是通过 DNS 来规范化主机名称的。这就解释了 DNS 为什么是微软 Kerberos 环境中的一个必要组件。查询服务的“规范化”名称，然后生成请求服务的 SPN。

下面我们介绍 Kerberos V5 的基本内容（如无特别说明，下文出现的 Kerberos 指的是 Kerberos V5），有关 Kerberos V4 的详细内容读者可参考文献[16]。

Kerberos 的一般认证过程如表 3-4 所示。

表 3-4　Kerberos 协议过程

步骤	通信双方	通信报文	报文内容
1	Client → AS	AS-REQ	Options ‖ ID_C ‖ RE_C ‖ ID_{TGS} ‖ Times ‖ $Nonce_1$
2	AS → Client	TGT	RE_C ‖ ID_C ‖ $T_{C,TGS}$ ‖ $E(K_C, K_{C,TGS}$ ‖ Times ‖ $Nonce_1$ ‖ RE_{TGS} ‖ $ID_{TGS})$ $T_{C,TGS} = E(K_{TGS}$, Flags ‖ $K_{C,TGS}$ ‖ RE_C ‖ ID_C ‖ AD_C ‖ Times)
3	Client → TGS	TGS-REQ	Options ‖ ID_S ‖ Times ‖ $Nonce_2$ ‖ $T_{C,TGS}$ ‖ A_C $T_{C,TGS} = E(K_{TGS}$, Flags ‖ $K_{C,TGS}$ ‖ RE_C ‖ ID_C ‖ AD_C ‖ Times) $A_C = E(K_{C,TGS}, ID_C \| RE_C \| TS_1)$
4	TGS → Client	TKT	RE_C ‖ ID_C ‖ $T_{C,S}$ ‖ $E(K_{C,TGS}, K_{C,S}$ ‖ Times ‖ $Nonce_2$ ‖ RE_S ‖ $ID_S)$ $T_{C,S} = E(K_S$, Flags ‖ $K_{C,S}$ ‖ RE_C ‖ ID_C ‖ AD_C ‖ Times)

续表

步骤	通信双方	通信报文	报文内容
5	Client → S	AP-REQ	Options ‖ $T_{C,S}$ ‖ A_C $T_{C,S} = E(K_S, \text{Flags} \| K_{C,S} \| RE_C \| ID_C \| AD_C \| \text{Times})$ $A_C = E(K_{C,S}, ID_C \| RE_C \| TS_2 \| \text{Subkey} \| \text{seq\#})$
6	S → Client	AP-REP	$E(K_{C,S}, TS_2 \| \text{Subkey} \| \text{seq\#})$

表 3-4 中的“报文内容”所使用的符号说明如下。

- ID_C：用户主体标志；
- ID_S：应用服务器主体标志；
- ID_{TGS}：票证授予服务器（TGS）标志；
- RE_X：标志用户 X 所属的域（Realm / Domain）；
- ‖：报文拼接符；
- AD_X：用户 X 的网址；
- TS：时间戳（timestamp）；
- K_X：X 的秘密密钥；
- $K_{C,S}$：C 与 S 的会话密钥；
- $K_{C,TGS}$：C 与 TGS 交互的会话密钥，由 AS 创建；
- $E(K_X, \text{info})$：用密钥 K_X 对 info 加密；
- $T_{C,TGS}$：C 与 TGS 交互的许可证，用 K_{TGS} 加密；
- $T_{C,S}$：C 使用应用服务 S 的许可证，用 K_S 加密；
- A_C：客户端生成的认证码（Authenticator）；
- Times：用于客户请求在许可证中设置时间，包括：from（请求许可证的起始时间），till（请求许可证的过期时间），rtime（请求 till 更新时间）；
- Nonce：现时值，确保消息是新的，而不是攻击者重放的；
- Options：用于请求在返回的许可证中设置指定的标志位（Flags，如表 3-5 所示，详细解释读者可参考文献[16]）；
- Subkey：子密钥，客户选择一个子密钥来保护与某特定应用服务间的会话，如果此域被忽略，则客户使用 $K_{C,S}$ 作为会话密钥；
- seq#：可选域，用于说明在此次会话中服务器向客户发送消息的序列号，以防止重放攻击。

表 3-5　Kerberos V5 中的标志域（Flags）

标　志	含　义
INITIAL	按照 AS 协议发布的服务授权票证而不是基于票证授权票证发布的
PRE-AUTHENT	在初始认证中，客户在授予票证前即被 KDC 认证
HW-AUTHENT	初始认证协议要求使用带上客户端独占的硬件资源
RENEWABLE	告知 TGS 此票证可用于获得最近超时票证的新票证

续表

标　志	含　义
MAY-POSTDATE	告知 TGS 事后通知的票证可能是基于票证授权票证的
POSTDATED	表示该票证是事后通知的，终端服务器可以检查 authtime 域，查看认证发生的时间
INVALID	不合法的票证在使用前必须通过 KDC 使之合法化
PROXIABLE	告知 TGS 根据当前票证可以发放给不同网络地址新的服务授权票证
PROXY	表示该票证是一个代理
FORWARDABLE	告知 TGS 根据此票证授权票证可以发放给不同网络地址新的票证授权票证
FORWARDED	表示该票证或是经过转发的票证或是基于转发的票证授权票证认证后发放的票证

如表 3-4 所示，Kerberos 域内认证过程如下：

1. 用户（Client）向 AS 发起请求（AS-REQ），申请访问许可证（TGT）。TGT 与应用服务器（Server）无关，即用一个 TGT 可以申请多个 Server 的 Ticket。发送的请求中包含 Client 自己的标志（ID_C）、TGS 标志（ID_{TGS}）、起止时间（Times），用于客户请求在许可证中设置起止时间和现时（$Nonce_1$）。与 Kerberos V4 不同的是，V5 中将 V4 中的生命期（Lifetime，8 位长，每单位表示 5 分钟，因此最长生命期为 $2^8 \times 5 = 1280$ 分钟）修改为精确的起止时间，允许许可证拥有任意长的生命期，同时使用 $Nonce_1$ 来防止重放。此外，V5 在请求中增加了 Options 字段。

2. AS 收到 Client 发来的请求（AS-REQ）后，验证 ID_C。验证通过后根据用户标识 ID_C 在密钥数据库中检索得到该用户的基于用户口令的密钥 K_C，选择一个随机加密密钥 $K_{C,\ TGS}$，然后给 Client 发送响应（TGT）。响应中主要包括两部分：用客户密钥 K_C 加密的 $K_{C,\ TGS}$、AS-REQ 请求中设定的时间、现时和 TGS 标志，以及用 TGS 密钥 K_{TGS} 加密的客户和 TGS 交互的许可证 $T_{C.\ TGS}$，里面包含了 $K_{C,\ TGS}$、标志（Flags）、主体信息（RE_C, ID_C, AD_C）、起止时间（Times）等。AD_C 为客户的地址信息，V4 只使用 IP 地址，不支持其他地址类型，如 ISO 网络地址，V5 中用类型和长度标记网络地址，允许使用任何类型的网络地址。

3. Client 收到 AS 发回的响应（TGT）后，首先用自己的密钥 K_C（一般使用用户口令进行保护）解密得到 $K_{C,\ TGS}$，并保存加密的访问许可证 $T_{C,\ TGS}$。然后，Client 向 TGS 发送一个请求（TGS-REQ），其中包括一个加密的 $T_{C,\ TGS}$ 和新生成的认证码 A_C。A_C 包含一个时间戳（TS_1）和客户信息（RE_C, ID_C），并用会话密钥 $K_{C,\ TGS}$ 加密。

4. TGS 收到 Client 发来的请求（TGS-REQ）后，TGS 用自己的密钥 K_{TGS} 解密得到许可证 $T_{C,\ TGS}$，并用得到的会话密钥 $K_{C,\ TGS}$ 解密 A_C。通过比较它们之中的时间戳的有效性确定用户的请求是否合法。确认用户合法后，TGS 生成用户要访问的某个应用服务器 S 的许可证 $T_{C,\ S}$，包括标志 Flags、会话密钥 $K_{C.\ S}$、客户信息（RE_C, ID_C）、$T_{C,\ S}$ 的有效生存期，并且使用服务器 S 的密钥 K_S 加密。然后，将 $T_{C,\ S}$ 和用 $K_{C,\ TGS}$ 加密的 $K_{C.\ S}$、生存期、现时、应用服务器的标志等信息一起发给用户 C（响应 TKT）。

5. Client 收到 TGS 发来的响应（TKT）后，用 $K_{C,\ TGS}$ 解密得到会话密钥 $K_{C,\ S}$。当用户请求应用服务时（AP-REQ），提交该服务器的许可证 $T_{C,\ S}$ 以及认证码 A_C（包括时间戳、用户信息、子密钥等，并用会话密钥 $K_{C,\ S}$ 加密）。V5 对 V4 的客户与应用服务器间的认证交换进行

了改进，增加了两个新的域：子密钥（Subkey）和序列号（seq#）。在 V5 中，客户和服务器可以协商得到子会话密钥，使得每个会话密钥仅被使用一次。降低因为多次使用同一个许可证访问特定服务情况时，被攻击者获得以前的会话消息的可能性。

6. 应用服务器（Server）收到 $T_{C,S}$ 和 A_C 后，先用自己的密钥 K_S 解密 $T_{C,S}$，得到会话密钥 $K_{C,S}$，通过它解密 A_C。通过比较时间戳的有效性，确认信息是否被修改。如果信息合法即认证成功。至此，用户和应用服务器之间就完成了会话密钥 $K_{C,S}$ 的交换。在 V4 中，如果用户要求与应用服务器相互认证，则应用服务器将自己刚得到的时间戳递增，加密后发送给用户（AP-REP）。在 V5 中，由于攻击者不可能在不知道密钥的情况下，创建消息 AP-REP，因此不需要对时间戳进行上述处理。在许可证的有效期内，用户可随时用自己持有的许可证 $T_{C,S}$ 或使用协商的子密钥申请应用服务，并可用会话密钥或子密钥加密它们之间的通信。

上述过程中，完成了第 2 步，客户得到了自己与 TGS 通信的会话密钥 $K_{C,TGS}$ 和一个被 TGS 的密钥加密的数据包（包含 $K_{C,TGS}$ 和关于自己的一些确认信息）。接下来，客户必须提供更多的身份证明信息（称为 Authenticator，主要包括一些 Client 身份标识、地址等信息和当前时间戳）给 TGS 以证明自己的身份（第 3 步）。TGS 验证通过后，才将客户和应用服务器通信用的会话密钥（$K_{C,S}$）和许可证（$T_{C,S}$）发送给客户（第 4 步）。完成了上述 4 步，客户和应用服务器就可以直接进行双向认证了（第 5、6 步）。

在 Kerberos V4 中，使用 DES 作为加密算法，V5 对此进行了改进：用加密类型标记密文，这样就可以使用多种加密技术。用加密类型和长度标记的密钥允许同一个密钥在不同算法中使用，并允许在给定算法中具有不同的表现形式。

Kerberos 协议使用时间戳来防止重放，而基于时间戳的认证机制只有在 Client 和 Server 端的时间保持同步的情况才有意义。所以，使用 Kerberos 协议认证需保证域内所有节点的时间同步。在实际应用中，一般通过访问同一个时间服务器来获得当前时间的方式来实现时间的同步，如使用网络时间同步协议（NTP）实现时钟同步。

此外，KDC 并没有将客户和服务器间的会话密钥 $K_{C,S}$ 发送给服务器，而只是发送给了客户，然后由客户发送给服务器。这样做的目的主要基于两点考虑：

（1）由于一个 Server 会面对许多不同的 Client，而每个 Client 都有一个不同的会话密钥。那么 Server 就会为所有的 Client 维护这样一个会话密钥的列表，这样做对于 Server 来说是比较麻烦而低效的。

（2）由于网络传输的不确定性，可能出现这样一种情况：Client 很快获得了会话密钥，并用这个会话密钥加密认证请求发送到 Server，但是用于 Server 的会话密钥还没有收到，并且很有可能承载这个会话密钥的报文永远也到不了 Server 端。这样的话，Client 将永远得不到 Server 的认证。

上面介绍的是在一个域（Realm）内的认证，跨域认证过程如图 3-8 所示。每个互操作域的 Kerberos 服务器（TGS）应共享一个对称密钥，双方的 Kerberos 服务器应相应注册。这种模式要求每一个域的 Kerberos 服务器必须相互信任其他域的 Kerberos 服务器对其域内用户的认证。如果有 N 个域，则需要 $N \times (N - 1)/2$ 次安全密钥交换，才可以实现一个域与其他域的交互，可伸缩性不好。

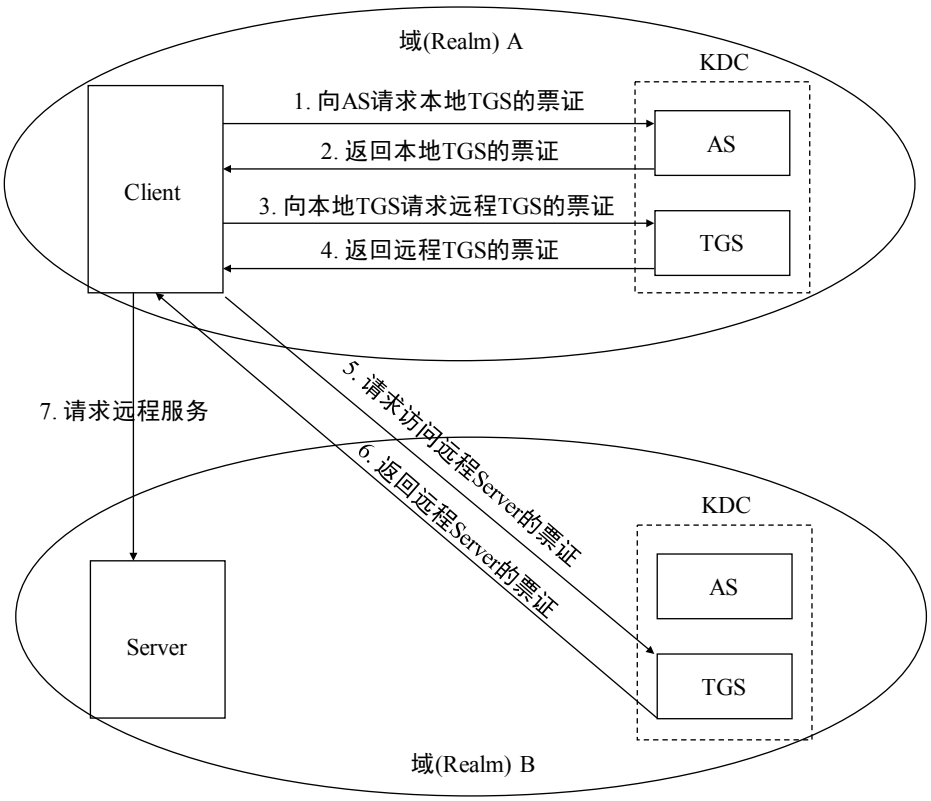


图 3-8　Kerberos 跨域认证

Kerberos 认证协议具有以下优点：

（1）一旦用户获得过访问某个服务器的许可证，只要在许可证的有效期内，该服务器就可根据这个许可证对用户进行认证，而无须 KDC 的再次参与。

（2）实现了客户和服务器间的双向认证。

（3）支持跨域认证。

（4）互操作性好。Kerberos 现在已经成为计算机安全领域一个被广泛接受的标准，所以使用 Kerberos 可以轻松实现不同平台之间的互操作。

Kerberos 也存在一些安全问题，主要包括：

（1）Kerberos 认证中使用的时间戳机制依赖于域内时钟同步，如果时间不同步则存在较大的安全风险。例如，如果主机时间不同步，原来的许可证可能会被替换；如果应用服务器的时间提前或落后于客户和 AS 的时间，则应用服务器可能会把有效的许可证看成是一个重放攻击而拒绝它。实践中，使用的时钟同步协议（NTP）也存在不少安全隐患。

（2）Kerberos 无法应付口令猜测攻击。AS 在传输用户与 TGS 间的会话密钥时是以用户密钥加密的，而用户密钥是由用户口令生成的，因此可能会受到口令猜测攻击。在 Kerberos V5 中，新增了一种称为“预认证机制”使得口令攻击更困难，但无法杜绝。

（3）用户必须保证他们的私钥的安全。如果一个入侵者通过某种方法窃取了用户的私钥，他就能冒充用户的身份。

（4）Kerberos 中 AS 和 TGS 采用集中式管理，容易形成瓶颈，系统的性能和安全性也

过分依赖于这两个服务的性能和安全。

此外，Kerberos 采用对称密码体制，因此很难实现不可否认性。Kerberos 使用的散列函数 MD4、MD5、SHA-1 也已不再安全。

3.4.3　可扩展认证协议 EAP

生活中，很多终端接入互联网是通过拨号网络（如 ADSL）、无线网络（如 Wi-Fi）、局域网（如单位内网）接入的。大多数情况下，只有注册用户使用的终端才能接入。接入过程主要包括两个步骤：认证使用终端的用户是否是注册用户和建立注册用户使用的终端与互联网中资源之间的传输路径，由接入控制设备完成。其中，第一步就是互联网接入控制过程中的身份认证。

在互联网接入控制中，早期拨号接入中常用的认证协议有口令认证协议（Password Authentication Protocol, PAP）、挑战握手认证协议（Challenge Handshake Authentication Protocol, CHAP）（参考文献[18]），无线接入网络中常用的认证协议有 WEP、WPA/WPA2（将在第 5 章介绍）等。PAP 协议和 CHAP 协议均是基于 PPP（Point to Point Protocol）协议来实现终端和接入控制设备之间认证报文的传输。随着接入方式的增多，认证协议仅以 PPP 作为认证协议的承载协议已不再可行，需要定义新的认证协议的传输协议来适配接入网络的数据链路层协议。在这样的背景下，IETF 在 RFC 3748 中定义了可扩展认证协议（Extensible Authentication Protocol, EAP）。

EAP 是一种普遍使用的支持多种认证方法的认证框架协议，在客户和认证服务器之间为认证信息交换提供了一种通用的传输服务，可以封装多种认证方法的协议报文。在数据链路建立阶段不需要选定某一种特定的认证机制，只需说明要使用 EAP 认证即可，而把具体认证过程推迟到后面一个独立的认证阶段。在该阶段进行认证方式的协商和具体认证过程，并根据认证成功与失败来决定是否允许终端接入网络。

EAP 协议层次结构如图 3-9 所示。

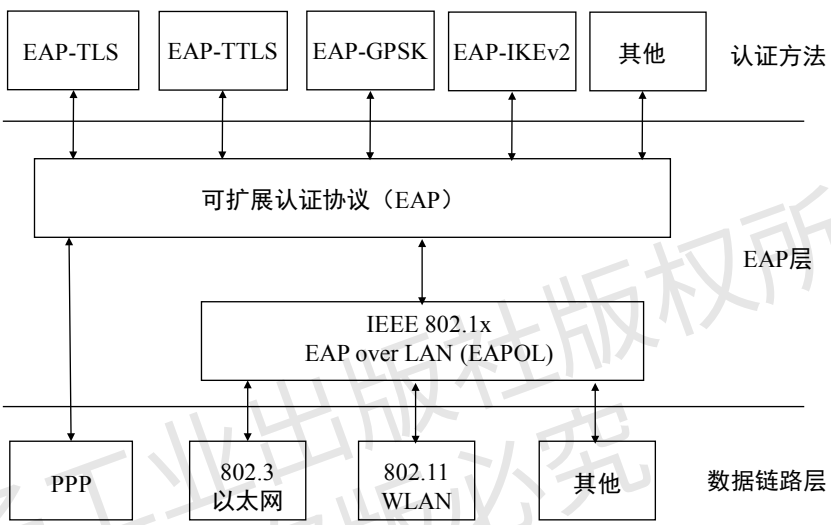


图 3-9　EAP 协议层次结构图

许多认证方法都可以在 EAP 协议上工作，如图 3-9 所示的 EAP-TLS 协议（RFC 5216）、EAP-TTLS 协议（RFC 5281）、EAP-GPSK（RFC 5433）、EAP-IKEv2（RFC 5106）。EAP-TLS（EAP Transport Layer Security）使用 TLS（Transport Layer Security）协议（将在第 7 章介绍）的握手协议进行认证，客户和服务器使用其数字证书进行双方认证。无线接入网（Wi-Fi）支持 EAP-TLS 认证，具体认证过程参见第 5.1.3 节。EAP-TTLS（EAP Tunneled TLS）和 EAP-TLS 基本相同，不同之处在于服务器首先要通过证书向客户证明自己，它使用密钥建立一个安全连接（“隧道”）用来对客户进行认证。此外，EAP-TTLS 中服务器也允许使用 PAP 和 CHAP 进行认证。EAP-GPSK（EAP Generalized Pre-Shared Key）指定了一个适用于交互认证的基于预共享密钥的 EAP 认证方法。EAP-IKEv2（EAP Internet Key Exchange version 2）在 IKEv2 协议（将在第 6 章介绍）的基础上进行交互认证。

无论使用何种认证方法，认证信息和认证协议报文都由 EAP 协议进行传输。基于 EAP 的认证系统中，一般包括以下组件：

（1）EAP 请求者（EAP peer）：请求访问网络的客户终端。

（2）EAP 认证者（EAP Authenticator）：一个接入点（Access Point, AP）或网络访问服务器（Network Access Server, NAS）。它们要求客户先进行认证后才能接入网络。

（3）认证服务器（Authenticator Server）：利用 EAP 认证方法和 EAP 客户进行交互，验证客户提供的信息，授权其访问网络。典型情况下，授权服务器是一个提供远程认证拨入用户服务（Remote Authentication Dial In User Service, RADIUS）的服务器。

认证服务器作为后台服务器可以为许多 EAP 认证者提供认证客户的服务，而后由 EAP 认证者进一步决定是否给客户提供接入服务，这就是 EAP 转移模式。有些情况下，认证者可能会集成了认证服务器的功能，此时就无须独立的认证服务器了。

认证前，客户通常使用一个底层协议（如 PPP 协议或 IEEE 802.1x）与认证者建立连接。EAP 客户和服务器之间交互的报文由以下几个域（Field）组成：

（1）类型（Code），1 字节：指定 EAP 报文的类型，共 4 种：Request、Response、Success、Failure，对应的类型码分别为：1、2、3、4。

（2）标志符（Identifier），1 字节：标识某一次请求/响应过程，用于匹配某一次认证过程中的 Request 和 Response 消息。

（3）长度（Length），2 字节：EAP 报文的长度，包含了 Code、Identifier、Length 和 Data 域。

（4）数据（Data）：具体与认证有关的信息，通常包括 Type 域（1 字节）和 Type-Data 域，其中 Type 域指定认证数据的类型（即认证机制或认证方法），如 1 表示身份（由于每个用户可能采用不同的认证机制，因此在开始认证过程前，需确定用户身份，然后才开始具体的认证过程），4 表示 CHAP 认证，5 表示 OTP 认证，13 表示 TLS 认证等。Success 和 Failure 类报文没有 Data 域。

在建立了客户和 EAP 认证者之间的底层通信连接后，就可以开始认证。EAP 转换模式下的报文交互过程如图 3-10 所示。

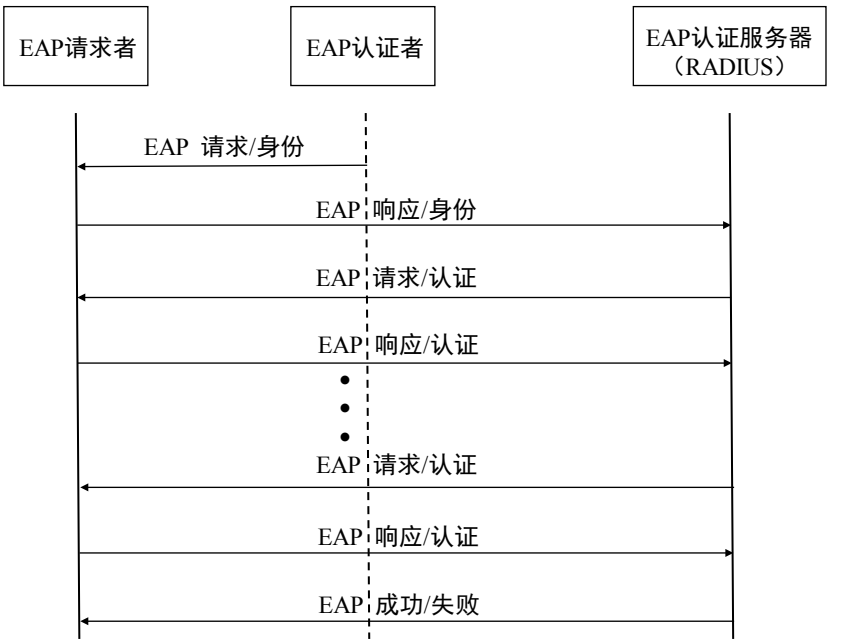


图 3-10　EAP 转换模式下的报文交互过程

首先，认证者向客户（EAP 请求者）发送一个对于其身份的请求（Data 域的 Type 域为 1），客户会返回一个带有其身份信息的响应，通过认证者转发给认证服务器。在收到身份响应后，认证服务器会根据用户的身份选择一种 EAP 认证方法，并发送第一个 EAP 请求报文，报文 Data 域的 Type 子域与认证方法有关。如果客户支持这种认证方法，就会响应一个该方法所对应的响应；否则，客户发送一个 NAK，服务器要么选择另一种 EAP 认证方法，要么发送 Failure 报文结束认证。客户与服务器间交换的请求和响应次数取决于所使用的认证方法。交换过程中还包括密钥信息等认证相关信息的交换。

两种情况下认证过程结束：一是认证者认为该客户不能通过认证，给其发认证失败报文（Failure），二是认证者认为其认证成功，给其发认证成功报文（Success）。

有关不同认证方法下的 EAP 报文交换过程的详细情况读者可参考文献[18]，本书的第 5 章、第 6 章、第 7 章也将有所介绍。

3.5　习题

一、单项选择题

1. 散列函数具有抗弱碰撞性是指（　　）。

A. 对于任意给定的 x，计算 $H(x)$比较容易

B. 对任意给定的散列值 h，找到满足 $H(x)=h$ 的 x 在计算上是不可行的

C. 对任意给定的数据块 x，找到满足 $y \neq x$ 且 $H(x)=H(y)$的 y 在计算上是不可行的

D. 找到任意满足 $H(y)=H(x)$的偶对（x, y）在计算上是不可行的

2. 散列函数具有抗强碰撞性是指（　　）。

A. 对于任意给定的 x, 计算 $H(x)$比较容易

B. 对任意给定的散列值 h，找到满足 $H(x)=h$ 的 x 在计算上是不可行的

C. 对任意给定的数据块 x，找到满足 $y \neq x$ 且 $H(x)=H(y)$的 y 在计算上是不可行的

D. 找到任意满足 $H(y)=H(x)$的偶对（x, y）在计算上是不可行的

3. 散列函数具有单向性是指（　　）。

A. 对于任意给定的 x, 计算 $H(x)$比较容易

B. 对任意给定的散列值 h，找到满足 $H(x)=h$ 的 x 在计算上是不可行的

C. 对任意给定的数据块 x，找到满足 $y \neq x$ 且 $H(x)=H(y)$的 y 在计算上是不可行的

D. 找到任意满足 $H(y)=H(x)$的偶对（x, y）在计算上是不可行的

4. 在通信过程中，如果仅采用数字签名，不能解决（　　）。

A. 数据的完整性　　B. 数据的抗抵赖性

C. 数据的防篡改　　D. 数据的保密性

5. 数字签名通常要先使用单向哈希函数进行处理的原因是(　　)。

A. 多一道加密工序使密文更难破译

B. 提高密文的计算速度

C. 缩小签名消息的长度，加快数字签名和验证签名的运算速度

D. 保证密文能正确还原成明文

6. A 向 B 发送消息 M，A 利用加密技术（E 为加密函数，D 为解密函数，K 为公共密钥，SK_A 为 A 的秘密密钥，SK_B 为 B 的秘密密钥）、散列函数（H）同时实现报文的认证、数字签名和保密性的方法是（　　）。

A. $E(K, M \| H(M \| S))$　　B. $M \| D(SK_A, H(M))$

C. $E(K, M \| D(SK_A, H(M)))$　　D. $E(K, M \| D(SK_B, H(M)))$

7. 采用公开密钥算法实现数字签名时，下面的描述（　　）是正确的。

A. 发送方用其私钥签名，接收方用发送方的公钥核实签名

B. 发送方用其公钥签名，接收方用发送方的私钥核实签名

C. 发送方用接收方的私钥签名，接收方用其公钥核实签名

D. 发送方用接收方的公钥签名，接收方用其私钥核实签名

8. A 向 B 发送消息 M，采用对称密钥密码（E 为加密算法，D 为解密算法，共享密钥为 K）进行报文内容认证时，下列说法正确的是（　　）。

A. 只需使用 $E(K, M)$ 就可实现报文内容的认证

B. 在报文 M 后附加 M 的 CRC 检验码后，使用 $E(K, M \| \text{CRC})$ 可实现报文内容的认证

C. 在报文 M 后附加 M 的 CRC 检验码后，使用 $(M \| \text{CRC})$ 可实现报文内容的认证

D. 使用对称密钥密码无法实现报文内容的认证

9. 要抵御重放攻击，可以采用（　　）。

A. 报文源认证　　B. 报文内容认证

C. 报文顺序认证　　D. 报文宿认证

10. 要抵御攻击者的假冒攻击，需要采用（　　）。
    A. 报文源认证
    B. 报文内容认证
    C. 报文顺序认证
    D. 必须同时采用报文源认证、报文内容认证和报文顺序认证
11. 下列属性中，（　　）不是 Hash 函数具有的特性。
    A. 单向性　　B. 可逆性
    C. 抗弱碰撞性　　D. 抗强碰撞性
12. 现代密码学中很多应用包含散列运算，下列应用中不包含散列运算的是（　　）。
    A. 消息加密
    B. 消息完整性保护
    C. 消息认证码
    D. 数字签名
13. MD5 算法以（　　）位分组来处理输入消息。
    A. 64　　B. 128　　C. 256　　D. 512
14. 签名者无法知道所签消息的具体内容，即使后来签名者见到这个签名时，也不能确定当时签名的行为，这种签名称为（　　）。
    A. 代理签名　　B. 群签名
    C. 多重签名　　D. 盲签名
15. 用户 A 利用公开密钥密码向用户 B 发送消息 M（假定 M 是无结构的随机二进制字符串），加密函数为 E，解密函数为 D，散列函数为 H，A 的公钥为 PU_a，私钥为 PR_a，B 的公钥为 PU_b，私钥为 PR_b，提供机密性、不可否认性、完整性保护的最佳方案是（　　）。
    A. $E(PU_b, M \| E(PR_a, H(M)))$　　B. $E(PU_a, M \| E(PRb, H(M)))$
    C. $E(PU_b, M \| H(M))$　　D. $E(PU_b, E(PR_a, M))$
16. 用户 A 利用公开密钥密码向用户 B 发送消息 M（假定 M 是可读的一句中文字符串），加密函数为 E，解密函数为 D，散列函数为 H，A 的公钥为 PU_a，私钥为 PR_a，B 的公钥为 PU_b，私钥为 PR_b，提供机密性、不可否认性、完整性保护的最佳方案是（　　）。
    A. $E(PU_b, M \| E(PR_a, H(M)))$　　B. $E(PU_a, M \| E(PR_b, H(M)))$
    C. $E(PU_b, M \| H(M))$　　D. $E(PU_b, E(PR_a, M))$
17. 用户 A 利用对称密钥密码向用户 B 发送消息 M（假定 M 是可读的一句中文字符串），加密函数为 E，解密函数为 D，散列函数为 H，对称密钥为 K，提供机密性、完整性保护的最佳方案是（　　）。
    A. $E(K, M \| H(M))$　　B. $E(K, M)$
    C. $M \| E(K, H(M))$　　D. $E(K, M) \| H(M)$
18. 用户 A 利用对称密钥密码向用户 B 发送消息 M（假定 M 是无结构的随机二进制字符串），加密函数为 E，解密函数为 D，散列函数为 H，对称密钥为 K，提供机密性、完整性保护的最佳方案是（　　）。
    A. $E(K, M \| H(M))$　　B. $E(K, M)$

C. $M \| E(K, H(M))$　　D. $E(K, M) \| H(M)$

19. 在 Needham-Schroeder 双向鉴别协议中，第（5）步 A 向 B 发送 $E(K_s, f(N_2))$而不是 $E(K_s, N_2)$的原因是(　　)。
A. 防重放攻击　　B. 没什么实际意义
C. 提高速度　　D. 防止攻击者窃听到 N_2

20. 若小张给小李发送一封邮件，并想让小李确信邮件是由小张发出的，则小张应该选用（　　）对邮件内容加密。
A. 小李的公钥　　B. 小李的私钥
C. 小张的公钥　　D. 小张的私钥

二、多项选择题

1. 数字签名可以用于解决通信双方发生（　　）时引发的争端。
A. 发送方不承认自己发送过某一报文
B. 接收方自己伪造一份报文，并声称它来自发送方
C. 网络上的某个用户冒充另一个用户发送报文
D. 接收方对收到的带有签名的信息进行篡改

2. 下列散列算法中，被认为不再安全的有（　　）。
A. MD5　　B. SHA-1　　C. SHA-2　　D. SM3

3. 下列机制可做现时（Nonce）的有（　　）。
A. 真随机数　　B. 时间戳　　C. 序号　　D. 伪随机数

4. 散列码经常被应用于（　　）等网络安全应用中。
A. 数字证书　　B. 正版软件检测　　C. 用户口令保护　　D. 恶意代码检测

5. 用户 A 利用公开密钥密码向用户 B 发送消息 M(假定 M 是可读的一句中文字符串)，加密函数为 E，解密函数为 D，散列函数为 H，A 的公钥为 PU_a，私钥为 PR_a，B 的公钥为 PU_b，私钥为 PR_b，可实现机密性、不可否认性、完整性保护的方案有(　　)。
A. $E(PU_b, M \| E(PR_a, H(M)))$　　B. $E(PU_a, M \| E(PR_b, H(M)))$
C. $E(PU_b, M \| H(M))$　　D. $E(PU_b, E(PR_a, M))$

三、简答题

1. 设 $H(m)$是一个抗碰撞的 Hash 函数，将任意长消息映射为定长的 n 位 Hash 值。对于所有的消息 $x, x', x \neq x'$, 都有 $H(x) \neq H(x')$。上述结论是否正确？说明原因。
2. 完善的数字签名体制需要满足哪些条件？
3. 数字签名的工作原理是什么？请简要分析。
4. 采用消息认证码技术认证报文内容与采用加密技术认证报文内容两种方法之间存在哪些区别？
5. 简要说明在报文中加入序号的作用。
6. 解释现时（Nonce）的含义及其作用。
7. 编程实现 RSA 数字签名方案。
8. 在基于 ElGamal 的数字签名方法中，为什么 k 必须是一次性的？

9. 简要分析消息认证码和散列函数的区别。
10. 简要分析 MD5 和 SHA-1 间的差异（建议从输入、输出、轮数、强度和速度等几个方面比较）。
11. 什么是消息重放？有哪些方法可以抵御消息的重放攻击，各有什么优缺点？
12. 对于 RSA 数字签名体制，假设 $p = 839$，$q = 983$，$n = p \times q = 824737$。已知私钥 $d = 132111$，计算公钥 e 和对消息 $m = 23547$ 的签名。
13. 假设在 ElGamal 数字签名体制中，$q = 31847$，$\alpha = 5$，公钥 $y = 25703$。已知(23972, 31396)是对消息 $M = 8990$ 的签名，(23972, 20481)是对消息 $M = 31415$ 的签名，求随机数 k 和私钥 x。
14. 对于 RSA 数字签名体制，假设模 $n = 824737$，公钥 $e = 26959$。
    （1）已知消息 m 的签名是 $s = 8798$，求消息 m。
    （2）数据对$(m, s) = (167058, 366314)$是有效的(消息, 签名)对吗？
    （3）已知两个有效的消息签名对$(m, s) = (629489, 445587)$与$(m', s') = (203821, 229149)$，求 $m \times m'$的签名。
15. 在 ElGamal 数字签名体制中，假设 $q = 19$，$\alpha = 13$。
    （1）如果签名者 A 选取的私钥为 $x = 10$，试计算公钥 y。
    （2）设签名者 A 要对消息 $m = 15$ 进行签名，且选取随机数 $k = 11$，计算机签名 s，并验证该数字签名的有效性。
16. 假定用户 A 和用户 B 之间的通信内容为无格式的二进制码消息，A 和 B 为了防止他人监听，拟采用对称加密算法对通信内容 M 进行加密，接收方解密消息得到的明文记为 M'。
    （1）接收方能够确信 $M' = M$ 吗？如果不能，请说明原因。
    （2）如果通信内容为可读的中文字符串或带格式化头部的明文，重新回答问题（1），并加以详细阐释。
17. 设 H 是一个安全的哈希函数，Alice 将消息和其哈希值 $M||H(M)$一并发送，以检测消息是否在传输过程中被篡改，问：这样做可否达到 Alice 的安全目标？为什么？
18. 从攻击者的角度来详细说明 N-S 协议中，为什么必须增加第（4）和第（5）步？
19. 在 Kerberos 协议的第（6）步中，为什么服务器（Server）不直接把通过会话进行加密的 A_C 原样发送给客户（Client），而是把 Timestamp 提取出来递增后加密发送给 Client？
20. Kerberos 协议是在 N-S 协议的基础上发展而来的双向认证协议，分析 Kerberos 协议对 N-S 协议做了哪些改进？这些改进是如何提高认证协议的安全性的？
21. Kerberos 协议报文中为什么要使用 Timestamp？
22. Kerberos 协议中，KDC 为什么不直接将加密的会话密钥分别发送给 Client 和 Server，而是只发送给 Client？
23. Kerberos 协议中，AS 并没有真正去认证这个发送请求的 Client 是否真的就是他所声称的那个人，就把会话密钥发送给他，这样做会不会有什么问题？请给出理由。

24. 在 Kerberos 协议的第（5）步中，为什么要发送两份（A_C 和 $T_{C,S}$）关于 Client 的信息给 Server？
25. 在 Kerberos 协议中，Client 是如何判断自己在访问真正的 Server？
26. Kerveros 协议 V5 对 V4 的请求生存期做了哪些改进？为什么？
27. 有人认为 EAP 协议主要功能是提供一种具体的认证方法。简要评述这种观点是否正确。
28. 分析口令认证、信物认证、地址认证、用户特征认证和密码学认证的优缺点。
29. 简述 S/KEY 协议的小数攻击原理。
30. 在集中式对称密钥分配协议 Needham—Schroeder 中，什么是通信方 A、B 的主密钥？这些主密钥和会话密钥 K_s 各有什么用途？主密钥如何进行安全分配？会话密钥如何进行安全分配？
31. 散列值和 HMAC 值从理论上都可以实现数据的完整性保护，HMAC 值和散列值在实现上有什么不同？HMAC 值除了能够保证数据的完整性，还能实现什么安全需求？在实际应用中，使用上述两种方法的哪一种能更好地实现数据的完整性？
32. S/KEY 是一种一次性口令技术，回答以下问题：
    （1）它是基于时间同步或事件同步的认证技术吗？那它是哪种认证技术？
    （2）它能实现双向鉴别还是单向鉴别？是哪方对哪方的鉴别？

3.6 实验

3.6.1 使用 GPG4win 进行数字签名

1. 实验目的

通过实验，让学生掌握使用 RSA 算法实施数字签名的过程，加深对数字签名原理的理解。

2. 实验内容与要求

（1）在 Windows 环境下安装 GPG4win，保持默认设置即可。

（2）打开 Kleopatra，生成一对 RSA 公钥和私钥（点击 File→New Key Pair）。密钥对生成好之后，有 3 个选项，1 是备份自己的密钥，2 是通过 Email 把密钥发送给自己的联系人，3 是把自己的公钥上传到目录服务器，方便别人查询下载。

（3）生成密钥对列表会显示在软件界面中，可以点击 Sign/Encrypt 对文件进行签名或加密。在弹出的对话框中，可以选择一个文件进行签名或加密，例如 test.txt，可以事先编辑一下文本。注意，签名要用自己的私钥进行加密，加密则使用对方的公钥。签名完成后，生成带签名的文件。

（4）使用签名人的公钥验证签名（点击 Decrypt/Verify 进行签名验证）。

（5）扩展实验内容：除了签名，还对文件进行加密，查看加密内容后，并进行解密。

（6）将相关输入和结果截图写入实验报告。

3. 实验环境

（1）平台：Windows 7 以上。
（2）签名文件可由教师提供，也可由学生自己创建（包含学生的姓名和学号等信息）。
（3）GPG4win 软件下载地址 http://www.gpg4win.org/，或使用教师提供的安装软件。

3.6.2　OpenSSL 软件的安装与使用

1. 实验目的

通过实验，让学生了解 OpenSSL 软件功能，掌握用 OpenSSL 产生密钥、生成散列值、加解密、数字签名等方法，加深对消息认证、数字签名原理的理解。

2. 实验内容与要求

（1）安装 OpenSSL。
（2）使用 OpenSSL 产生密钥。
（3）利用 OpenSSL 提供的命令对指定文件分别生成散列值、加解密、数字签名。
（4）利用 OpenSSL 提供的命令分别进行完整性检验、解密、验证签名。

3. 实验环境

（1）实验室环境：计算机一台，操作系统为 Linux。
（2）签名文件可由教师指定（一般包含学生的姓名和学号等信息）。
（3）OpenSSL 软件下载地址 https://www.openssl.org/source/，或使用教师提供的安装软件。