网络空间信息安全

第5章 数字签名与认证技术

本章主要内容

- 5.1 数字签名
- 5.2 安全散列函数
- 5.3 认证技术

5.1 数字签名

- 5.1.1 数字签名概念
- 5.1.2 数字签名的实现过程
- 5.1.3 ElGamal数字签名算法
- 5.1.4 Schnorr数字签名算法
- 5.1.5 数字签名标准DSS

5.1.1 数字签名概念

- 数字签名是网络中进行安全交易的基础，数字签名不仅可以保证信息的完整性和信息源的可靠性，而且可以防止通信双方的欺骗和抵赖行为。虽然报文认证能够保证通信双方免受任何第三方的攻击，然而却不能保护通信双方中的一方防止另一方的欺骗和伪造。

5.1.1 数字签名概念

- 例如，当用户A和用户B进行通信时，若未使用数字签名，则用户A可以随意地伪造报文，并声称该报文是来之用户B的；同时用户B也可以否认曾经真正发送给用户A的报文。因此，在收发双方未建立起完全信任关系时，单纯的报文认证就显得不够充分，因而需要数字签名技术。

5.1.1 数字签名概念

- 数字签名应具有以下性质：
- (1) 必须能够验证签名生成者的身份以及生成签名的时间；
- (2) 能够用于证实被签名消息的内容；
- (3) 数字签名必须能被第三方验证，从而解决通信双方的争议。

5.1.1 数字签名概念

- 数字签名应满足以下安全要求：
- (1)签名的产生必须使用对方发送来说是唯一的信息，以防止伪造和抵赖；
- (2) 签名的产生必须相对简单；
- (3) 数字签名的识别和验证必须相对简单；
- (4) 对已有的数字签名伪造一个新的报文或对已知的报文伪造一个虚假的数字签名，在计算上是不可行的。

5.1.1 数字签名概念

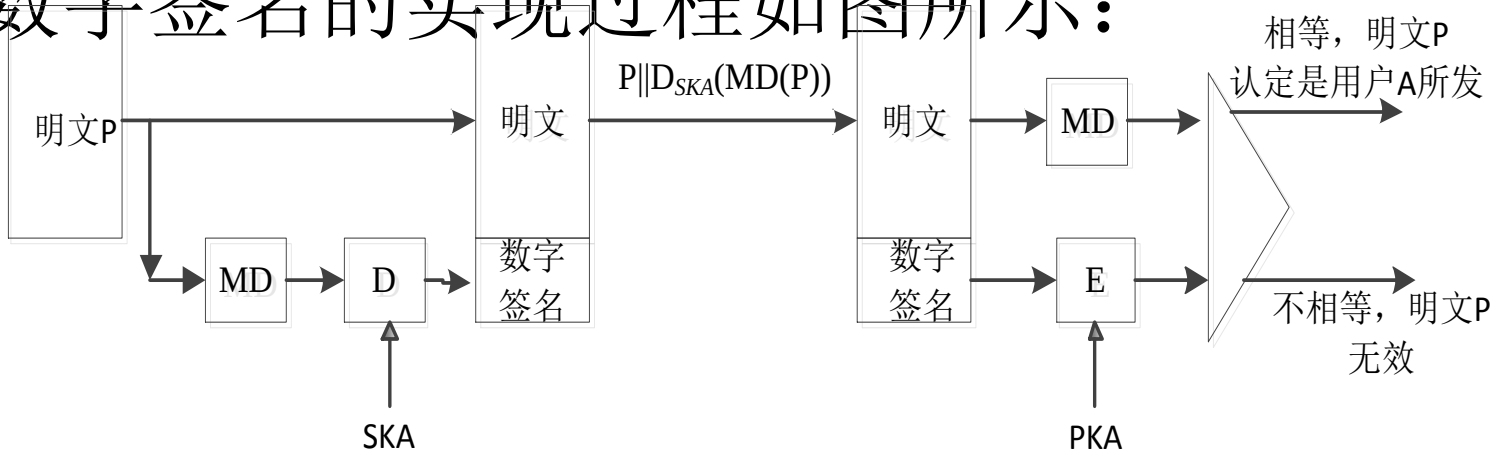
- 数字签名有直接数字签名和需仲裁的数字签名两种使用方式。
- 直接数字签名方式是在数字签名的使用过程中只有通信双方参与，并假定通信双方有共享的秘密密钥或接收端知道发送端的公钥。在直接的数字签名中，数字签名可以通过使用发送端的私钥对整个报文进行加密形成，或者通过使用发送端的私钥对报文的散列值进行加密来形成。

5.1.1 数字签名概念

- 然而，所有的直接数字签名方案都具有共同的弱点，即方案的有效性依赖发送端私钥的安全性。例如，发送端在用私钥对报文签名后，想否认发送过该报文，此时发送端可以声称该私钥丢失，签名被伪造。另一种可能的情况是，私钥确实在时刻T在X处被盗，攻击者可以发送带有X的签名报文并附加小于等于T的时间戳。
- 为了解决直接数字签名存在的问题，广泛采用的方法是使用数字证书的证书权威机构CA等可信的第三方参与，即使用需仲裁的数字签名方案。

5.1.2 数字签名的实现过程

- 公开密钥算法中公钥和私钥一一对应。私钥具有私密性，只有用户本身知道。如果公钥和用户之间的绑定关系能够被权威机构证明，就具有不可否认性。
- 数字签名的实现过程如图所示：



5.1.2 数字签名的实现过程

- 用户A用私钥SKA对明文P经过报文摘要算法后得到的摘要MD(P)进行解密运算，产生数字签名(DSKA(MD(P))),将明文P和数字签名一同发送给用户B。用户B认定明文P是用户A发送的前提是：用与用户A绑定的公钥PKA对数字签名进行加密运算后得到的结果和对明文P进行报文摘要运算后得到的结果相同，即EPKA(数字签名)=MD(P)。

5.1.2 数字签名的实现过程

- DSKA(MD(P))能够作为发送端用户A对报文P的数字签名依据如下：
- (1) 私钥SKA只有用户A知道，因此，只有用户A才能实现DSKA(MD(P))运算过程，保证了数字签名的唯一性。
- (2) 根据报文摘要算法的特性，即从计算可行性上讲，其他用户无法生成某个报文P′,P≠P′，但MD(P)=MD(P′)，因此，MD(P)只能是针对报文P的报文摘要算法的计算结果，保证了数字签名和报文P之间的关联性。

5.1.2 数字签名的实现过程

- (3) 数字签名能够被核实。公钥PKA和私钥SKA一一对应，如果公钥PKA和用户A之间的绑定关系得到权威机构证明，那么一旦证明用公钥PKA对数字签名进行加密运算后还原的结果(EPKA(数字签名))等于报文P的报文摘要(MD(P))，就可以证明数字签名是DSKA(MD(P))。
- 用公开密钥算法实现数字签名的前提是由权威机构出具证明用户和公钥之间绑定关系的证书，只有公钥和用户之间的绑定关系得到有公信力的权威机构的证实，才能核定该用户的数字签名。

5.1.3 ElGamal数字签名算法

- ELGamal密码体制机制能够使用用户的公钥进行加密，使用私钥进行解密，从而提供机密性。ELG挨骂了数字签名算法则是使用私钥进行加密，使用公钥进行解密。

5.1.3 ElGamal数字签名算法

- ELGamal数字签名算法描述如下：
- 假设用户A与用户B通信，A与B共享大素数q，α是q的一个原根，其中gcd(q,α)=1,α<q。
- (1) 用户A：
- ① 随机生成整数X_A，$1<X_A<q-1$;
- ② 计算$Y_A=\alpha^{X_A} \bmod q$;
- ③ A的公钥为$\{q,\alpha,Y_A\}$，私钥为X_A。

5.1.3 ElGamal数字签名算法

- ⑵ 为了对报文M进行签名，用户A首先计算散列值h=H(M)，其中h是满足0≤h≤q-1的整数。然后用户A生成数字签名：
- ① 秘密地随机选择一个整数K，其中1≤K≤q-1，且gcd(K,q-1)=1;
- ② 计算$S1=\alpha^K \bmod q$;
- ③ 计算K模(q-1)的逆$K^{-1} \bmod (q-1)$;
- ④ 计算$S2=K^{-1}(h-X_A S_1) \bmod (q-1)$;
- ⑤ 生成数字签名(S1,S2)对。

5.1.3 ElGamal数字签名算法

- (3) 用户B验证数字签名：
- ① 计算$V1=\alpha^{h} \bmod q$；
- ② 计算$V2=(Y_A)^{S1}(S_1)^{S2} \bmod q$;
- ③ 若V1=V2，则签名合法。

5.1.3 ElGamal数字签名算法

- 下面是ELGamal数字签名算法正确性的证明。
- 证明：假设V1=V2，则有
- $\alpha^h \bmod q=(Y_A)^{S1}(S_1)^{S2} \bmod q$
- $\alpha^h \bmod q=\alpha^{XA\ S1}\alpha^{KS2} \bmod q$
- $\alpha^{h-XA\ S1} \bmod q=\alpha^{KS2} \bmod q$
- 由于去是素数，α是去的原根，则有
- ⑴ 对于任意整数m，$\alpha^m \equiv 1 \bmod q$当且仅当$m \equiv 0 \bmod (q-1)$；
- ⑵ 对于任意整数i，j，$\alpha^i \equiv \alpha^j \bmod q$当且仅当$i \equiv j \bmod (q-1)$。
- 因此，根据原根的性质可以得到
- $h-X_AS_1 \equiv KS_2 \bmod (q-1)$
- $h-X_AS_1 \equiv KK^{-1}(h-X_AS_1) \bmod (q-1)$
- 等式成立，证毕。

5.1.4 Schnorr数字签名算法

- Schnorr数字签名算法的目标是将生成签名所需的报文计算量最小化，其生成签名的主要工作不依赖于报文，而是可以在空闲时执行，与报文相关的部分需要进行2nbit长度的整数与nbit长度的整数相乘。与ELGamal数字签名算法相同，Schnorr数字签名算法的安全性同样依赖于计算有限域上离散对数的难度。

5.1.4 Schnorr数字签名算法

- Schnorr数字签名算法描述如下：
- (1) 选择素数p和q，使得q是p-1的素因子，一般地，取$p \approx 2^{1024}$和$q \approx 2^{160}$，即p为1024bit整数，而q是160bit整数；
- (2) 选择整数α，使得$\alpha^q = 1 \bmod p$；
- (3) 全局公钥参数为{α，p，q}，用户组内的所有用户均可使用该参数；
- (4) 随机选择整数s作为作为私钥，0<s<q;
- (5) 计算$v=\alpha^{-s} \bmod p$，作为公钥；
- (6) 生成公钥/私钥对(v,s)。

5.1.4 Schnorr数字签名算法

- ⑺ 用户生成签名：
- ① 随机选择整数r，0<r<q，并计算x=α^rmodp；（预处理过程，与待签名的报文无关）
- ② 将报文M与x连接在一起计算散列值h=H(M||x);
- ③ 计算y=(r+sh)modq；
- ④ 生成签名对(h,y)。
- ⑻ 其他用户验证签名(h,y)：

5.1.4 Schnorr数字签名算法

- ① 计算$x'=\alpha^{y}v^{h}$modp；
- ② 验证h=H(M||x')是否成立。
- 对于该验证过程，有
- $x' \equiv \alpha^{y}v^{h} \equiv \alpha^{y}\alpha^{-\mathrm{sh}} \equiv \alpha^{\mathrm{y}-\mathrm{sh}}\equiv\alpha^{r}\equiv$xmodp
- 于是，H(M||x')= H(M||x)成立。

5.1.5 数字签名标准DSS

- 在现实社会中，特别是在IT行业界，标准化可以降低成本，还具有兼容性等优点。密码与信息安全技术大量使用于网络通信中，标准化必然是其中一项重要工作，数字签名的标准制定就是其必备部分之一。影响较大的制定信息安全相关标准的组织有：ISO和国际电子技术委员会（IEC），美国国家标准协会（ANSI），美国国家标准与技术委员会（NIST）制定的美国联邦信息处理标准（FIPS）系列，Internet研究和发展共同体制定的标准，IEEE微处理器标准委员会制定的标准，RSA公司制定的PKCS系列标准等等。

5.1.5 数字签名标准DSS

- 20世纪1994年12月，由美国国家标准与技术委员会（NIST）正式发布了数字签名标准DSS(Digital Signature Standard)即联邦信息处理标准 FIPS PUB 186。它是在EIGamal和Schnorr数字签名体制的基础上设计的，其安全性基于有限域上离散对数问题求解的困难性。DSS最早发表于1991年8月，该标准中提出了数字签名算法DSA和安全散列算法SHA，它使用公开密钥，为接收者提供数据完整性和数据发送者身份的验证，也可由第三方用来验证签名和所签数据的完整性。

5.1.5 数字签名标准DSS

- 人们对DSS提出了很多意见，主要包括：
- (1) DSA不能用于加密和密钥分配；
- (2) DSA是由美国国家安全局NSA研制的，因为有人对NSA不信任，怀疑其中可能存在陷门，特别是NIST一开始声称DSA是他们自己设计的，后来表示得到了NSA的帮助，最后承认该算法的确是由NSA设计的；DSA算法未经过公开选择阶段，未公开足够长的时间以便人们分析其完全强度和弱点；

5.1.5 数字签名标准DSS

- (3) DSA与RSA在签名时的速度相同，但验证签名时的速度DSA要慢10到40倍；
- (4) 密钥长度只有512位，由于DSA的安全性取决于计算离散对数的难度，因此有很多密码学家对此表示担心。NIST于1994年5月19日正式颁布了该标准，并将密钥长度的规定改在512位至1024位之间可变。

5.1.5 数字签名标准DSS

- 数字签名标准DSS主体部分条目
- 下面介绍的内容是以2000年修订标准为基础：
- (1) 首先对数字签名标准DSS作了简单的介绍和说明。
- (2) 规定使用数字签名算法（DSA），消息散列值使用SHA-1。
- (3) 描述数字签名算法（DSA）使用的参数。
- (4) 数字签名算法（DSA）的产生。
- (5) 数字签名算法（DSA）的验证。
- (6) RSA数字签名算法。
- (7) 椭圆曲线数字签名算法（ECDSA）的介绍。

5.1.5 数字签名标准DSS

- DSA数字签名算法
  - 在数字签名算法DSA中，有3个参数（全局公开密钥分量）对于一组用户是公开的和公用的：素数p，其中$2^{L-1}<p<2^{L}$，$512\leq L\leq 1024$，且L是64的倍数；素数q，其中q是（p-1）的因数，$2^{159}<q<2^{160}$；常数$g=h^{(p-1)/q} \bmod p$，其中整数h满足条件$1<h<(p-1)$且使得$g>1$。每个用户的私有密钥x是随机或伪随机整数，且$0<x<q$以及公开密钥$y=g^{x} \bmod p$。
  - 现在利用上述5个参数以及安全散列算法SHA，可以实现数字签名。

5.1.5 数字签名标准DSS

- 假设发送方A对消息M签名：
- 第一步，发送方A产生一个随机整数k，其中0＜k＜q；
- 第二步，利用k和SHA散列算法计算$r=(g^k \bmod p) \bmod q$和$s=(k^{-1}(H(m)+x^r)) \bmod q$，然后发送方A将r和s作为自己对信息M的签名，把它们发送给接收方B；
- 第三步，接收方B收到发送方A的消息M1和签名(r_1,s_1)后，计算$w=(s_1)^{-1} \bmod q$，$u_1=[H(M_1)\ w] \bmod q$，$u_2=r_1\ w \bmod q$，$v=[(g^{u1}y^{u2}) \bmod p] \bmod q$；
- 第四步，如果 $v=r_1$，则接收方B认为发送方A对消息M的签名有效。

5.1.5 数字签名标准DSS

- 由于DSA中的素数p和q是公用的，它们必须公开，因此人们关心它们的产生方法。国际密码学家 Lenstra和Haber指出：如果使用某些特定的素数，那么可以很容易地伪造签名。于是美国国家标准与技术委员会在DSS中特别推荐了一种公开的产生素数的方法。该方法中的变量S称为“种子”，C称为“计数”，N称为“偏差”，同时将安全散列算法SHA用于素数的产生中，以防止有人在背后做手脚。

5.1.5 数字签名标准DSS

- 下面来看一个用DSS数字签名的例子。
- 设q=101、p=78*101+1=7879，3为F_{7879}的一个本原元，所以能取$\alpha=3^{78}(\text{mod } 7879)=170$为模$p$的$q$的单位根。假设α=75，那么$\beta=\alpha^{\alpha}$（mod 7879）=4567。现在，假设Bob要签名一个消息为x=1234，而且已经选择拉随即值k=50，可算出k^{-1}（mod 101）=99，则计算签名如下：
- $\gamma=\alpha^{k}(\text{mod } p)(\text{mod } q)=170^{50}(\text{mod } 7879)(\text{mod } 101)=2518(\text{mod } 101)=94$
- $\delta=(x+\alpha\gamma)\ k^{-1}(\text{mod } q)=(1234+75*94)*99(\text{mod } 101)=97$

5.1.5 数字签名标准DSS

- 所以签名为（1234，94，97）。签名的验证过程为：
- $\gamma^{-1}=97^{-1}$ （mod 101）=2
- $e_1=x\delta^{-1}(\text{mod } q)=1234*25(\text{mod } 101)=45$
- $e_2=\gamma\delta^{-1}(\text{mod } p)=94*25(\text{mod } 101)=27$
- $(\alpha^{e1}\beta^{e2}(\text{mod } p))(\text{mod } q)=(170^{45}*4567^{27}(\text{mod } 7879))(\text{mod } 101)$ $=2518(\text{mod } 101)=94$
- 因此该签名是有效的。

5.2 安全散列函数

- 5.2.1 安全散列函数的应用
- 5.2.2 散列函数的安全性要求
- 5.2.3 MD5报文摘要算法
- 5.2.4 SHA-1安全散列算法

5.2.1 安全散列函数的应用

- 散列函数又称为Hash函数或杂凑函数，散列函数H以变长的报文M作为输入，产生一个定长的散列码H(M)作为输出。在安全应用中使用的散列函数称为密码学散列函数或安全散列函数。
- 安全散列函数可以实现报文认证和数字签名，因而被广泛应用于不同的安全应用和网络协议。

5.2.1 安全散列函数的应用

- 1 报文认证
- 报文认证是用来验证消息完整性的安全服务或安全机制，报文认证确保收到的报文来自可信的源点且在传输过程中未被篡改。当散列函数应用于报文认证时，散列值H(M)又称为报文摘要MD(Message Digest)。

5.2.1 安全散列函数的应用

- 散列函数能够通过不同的方式提供报文认证功能。 散列函数用于报文认证的方法可以分为以下几种：
- (1) 使用对称加密方法对附加散列值的报文进行加密
- 假定源端A向目的端B发送报文EK[M||H(M)]，由于仅有A和B共享密钥K，所以，可以确定该报文必定来自A且未被篡改，其中的散列值提供了实现认证所需要的结构。
- 另外，由于对报文和散列值整体进行加密，因此也提供了机密性。

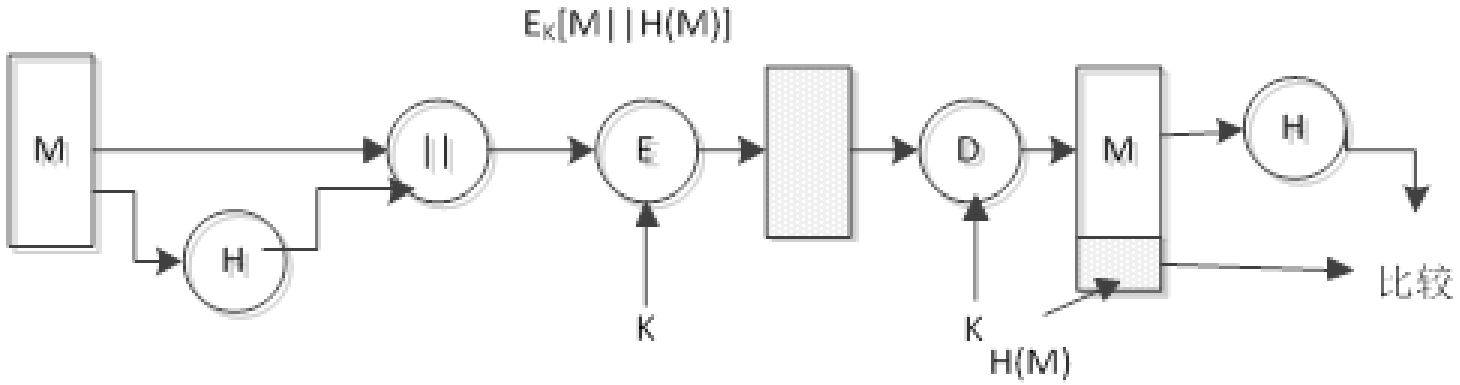
5.2.1 安全散列函数的应用

- (2) 使用对称加密方法仅对散列值进行加密
- 源端A向目的端B发送报文M||Ek[H(M)]，由于只对散列值进行加密，因而无法提供机密性，仅提供认证功能，但减少了加解密操作的开销。
- (3) 使用散列值、公共秘密值的明文方案
- 该方案使用了公共的秘密值S，假定通通信双方共享该秘密值S。源端A对报文M和公共秘密值S的连接计算散列值H(M||S)，并将得到的散列值附加在报文M之后得到M||H(M||S)，并向目的端B发送该报文。由于目的端B知道该秘密值S，因而能够重新计算该散列值H并进行验证。另外，因为秘密值本身并不被发送，所以攻击者无法更改中途截获的报文，也就无法产生假报文。

5.2.1 安全散列函数的应用

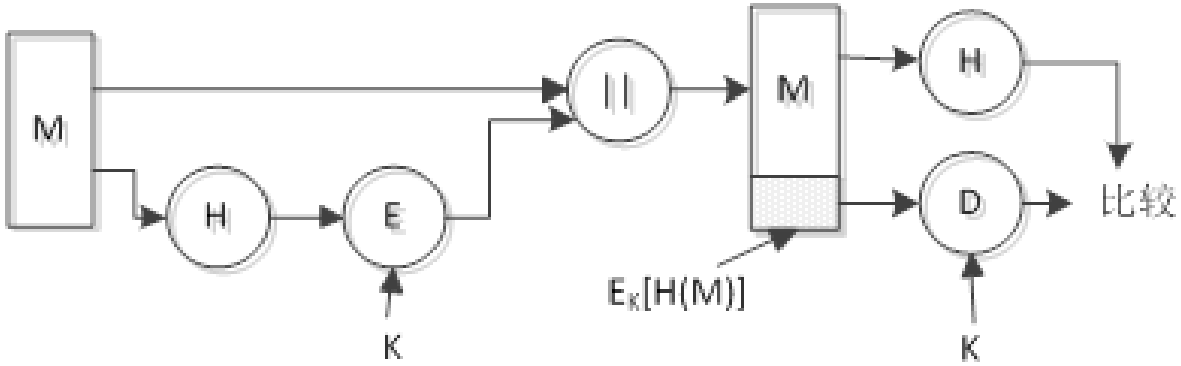
- 该方案并未对明文M进行加密，因而无法提供机密性，仅提供报文认证功能。
- (4) 使用散列值、公共秘密值的密文方案
- 该方案与方案(C)相似，但对明文M和散列值整体进行加密，因而可以同时提供机密性和报文认证功能。

5.2.1 安全散列函数的应用



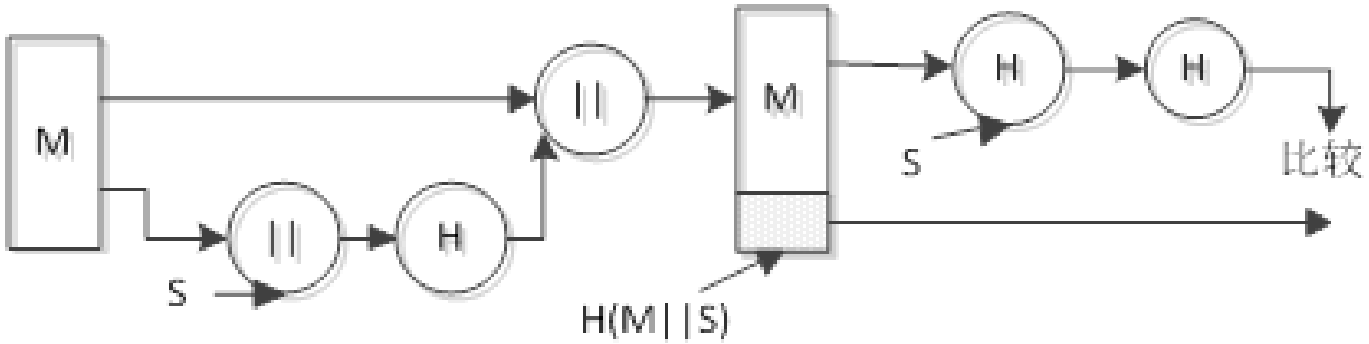
(a) 使用常规加密方法对附加散列值的报文进行加密

5.2.1 安全散列函数的应用



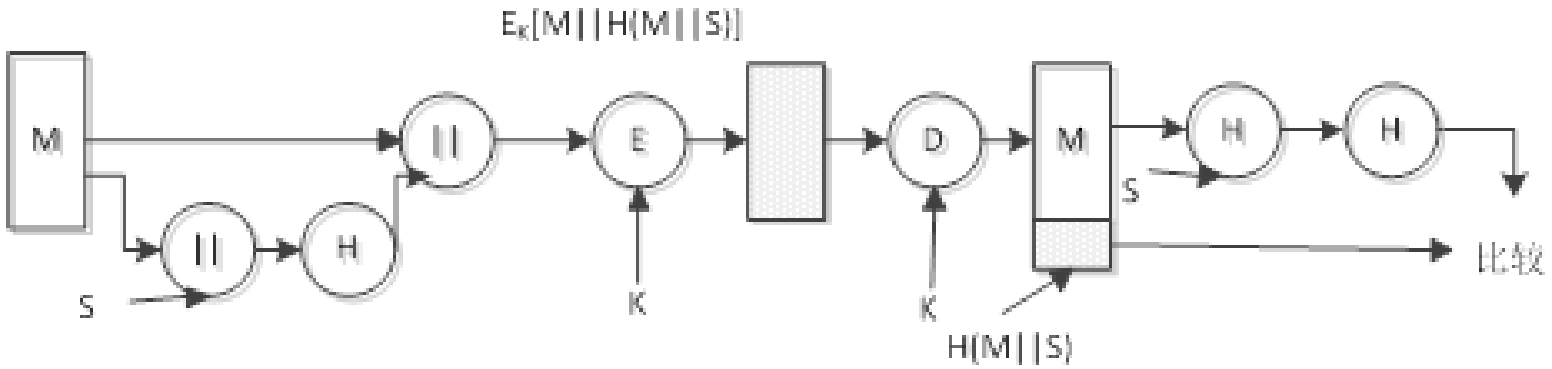
(b) 使用常规加密方法仅对散列值进行加密

5.2.1 安全散列函数的应用



(c) 使用散列值、公共秘密值的明文方案

5.2.1 安全散列函数的应用



(d) 使用散列值、公共秘密值的密文方案

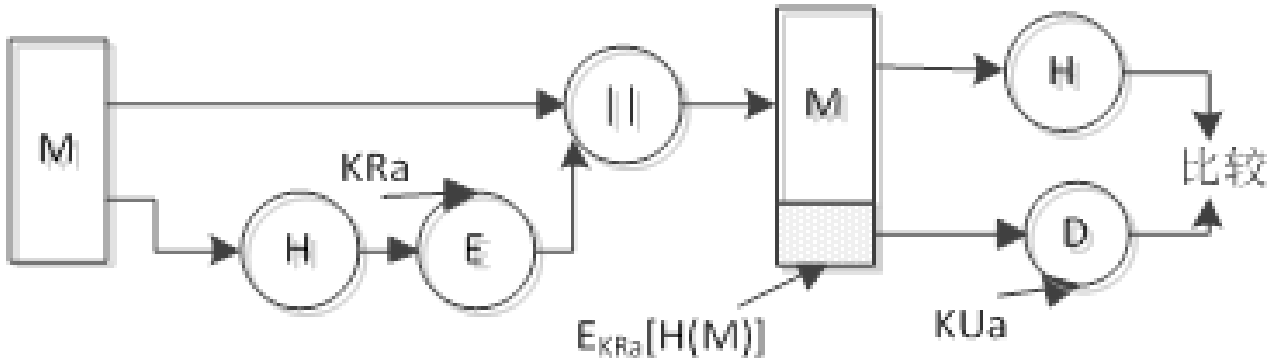
5.2.1 安全散列函数的应用

- 2 数字签名
- 散列函数的另外一个重要应用是数字签名。数字签名是一种防止源点或终点抵赖的技术，在进行数字签名过程中使用用户的私钥对报文的散列值进行加密，其他任何知道该用户公钥的用户均能通过数字签名来验证报文的完整性。因此，攻击者若想篡改报文，则需要知道用户的私钥。与报文认证相比，数字签名的应用更为广泛。

5.2.2 散列函数的安全性要求

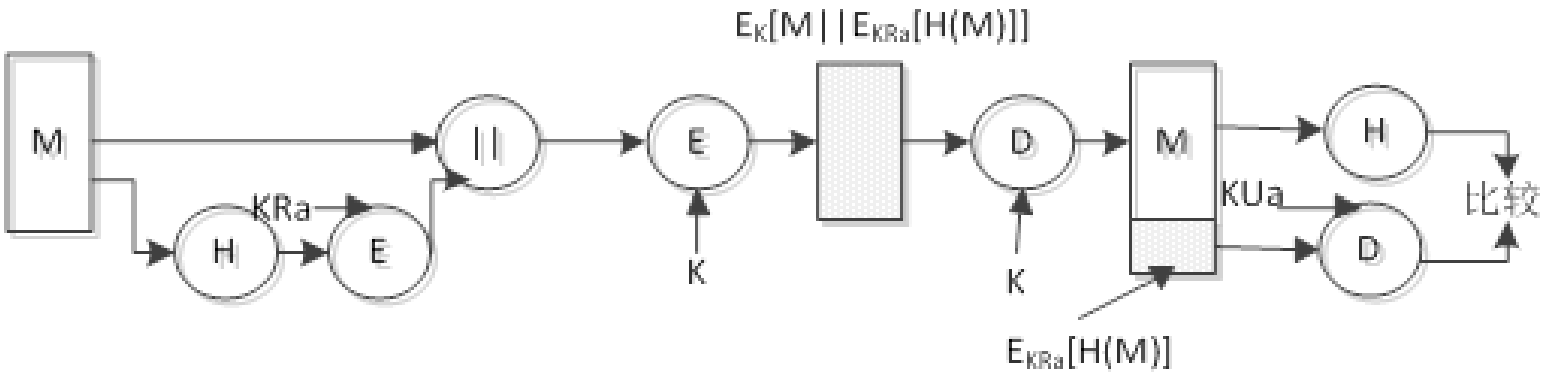
- 散列函数用于数字签名的方式可以分为以下两种。
- (a) 使用公钥加密及源端私钥仅对散列值进行加密
- 源端A首先使用自己的私钥对散列值进行加密，然后与报文M进行连接形成M||EKRa[H(M)]发送给目的端B。该方案可提供认证功能，但不能提供机密性。
- 另外，由于仅有源端A能生成M||EKRa[H(M)]，因而提供了数字签名。
- (b) 使用对称加密对报文和已使用公钥加密的散列值进行加密。
- 该方案在方案(a)的基础上，采用对称加密对报文M和以私钥加密的散列值EKRa[H(M)]再次进行加密，形成EK[M||EKRa[H(M)]]，从而在提供数字签名的同时，也提供了机密性。
- 方案(b)是较为常用的散列函数使用方法。

5.2.1 安全散列函数的应用



(a) 使用公钥加密及源端私钥仅对散列值进行加密

5.2.2 散列函数的安全性要求



(b) 使用对称加密对报文和已使用公钥加密的散列值进行加密

5.2.2 散列函数的安全性要求

- 使用散列函数的目的是为文件、报文或其他分组数据产生“指纹”。因此可以说，散列函数的首要目标是保证数据的完整性，报文M的任何微小的改变均会导致散列函数值H(M)的变化。

5.2.2 散列函数的安全性要求

- 1 散列函数的性质
  - 若期望在安全应用中使用散列函数，则散列函数M必须具有如下性质：
    - (1) H能用于任意大小的数据分组；
    - (2) H产生定长的输出；
    - (3) 对任意给定的x，H(x)要相对易于计算，使得硬件和软件实现成为可能；
    - (4) H应具有单向性(one way)，或称为抗原像攻击性(preimage resistant)，即对任意给定的散列值h，寻找x使得H(x)=h在计算上是不可行的；
    - (5) H应具有抗弱碰撞性(weak collision resistant)，或称为抗第二原像攻击性，即对任意给定的分组x，寻找y≠x，使得H(y)=H(x)在计算上是不可行的；
- (6) H应具有抗强碰撞性(strong collision resistant)，即寻找任意的(x,y)对，使得H(x)=H(y)在计算上是不可行的。

5.2.2 散列函数的安全性要求

- 前3个性质是散列函数应用于报文认证和数据签名的实际应用需求，而后3个性质则是散列函数的安全性需求。

5.2.2 散列函数的安全性要求

- 2 针对具有抗弱碰撞性散列函数的攻击
- 此类攻击问题可以描述为：散列函数H有n个可能的散列值，给定x和散列值H(x)，则在散列函数H随机生成的K个散列值中，至少存在一个y使得H(y)=H(x)的概率为0.5时，k的取值是多少？

5.2.2 散列函数的安全性要求

- 首先，对于特定的y值，使得H(x)=H(y)的概率为1/n，相应地，H(x)≠H(y)的概率为1-$\frac{1}{n}$。那么，若y取k个随机值，使得函数的k个散列值H(y)没有任何一个等于H(x)的概率则为$\left(1-\frac{1}{n}\right)^k$，因而k个散列值中至少有一个等于H(x)的概率为1-$\left(1-\frac{1}{n}\right)^k$。

5.2.2 散列函数的安全性要求

- 根据二项式定理
  - $(1-\mathrm{a})^k=1-\mathrm{ka}+\frac{k(\mathrm{k}-1)}{2!}a^2-\frac{k(k-1)(k-2)}{3!}a^3+\cdots$
- 对于非常小的a，$(1-\mathrm{a})^k$近似等于(1-ka)，因此
  - $1-\left(1-\frac{1}{n}\right)^k\approx 1-\left(1-\frac{k}{n}\right)=\frac{k}{n}$
- 因此，在散列函数H随机生成的k个散列值中，至少存在一个y使得H(y)=H(x)的概率约为k/n。若使概率为0.5，则k=n/2。

5.2.2 散列函数的安全性要求

- 特别地，如果散列函数H生成的散列值长度为m bit，可能存在2^m个散列值，则k个值$H(y_1)$，$H(y_2)$，…，$H(y_k)$中至少有一个等于H(x)的概率为1/2的k值为2^{m-1}。

5.2.2 散列函数的安全性要求

- 3 生日攻击
- 生日攻击是建立在生日悖论基础上的，具有抗强碰撞性的散列函数对“生日攻击”具有抵抗能力。
- 所谓生日悖论是指k个人中至少有两个人生日相同的概率大于0.5的最小k值是多少。

5.2.2 散列函数的安全性要求

- 首先可以定义：
  - P(n，k)=P_r[k个元素中至少有一个元素重复出现，其中每个元素出现的概率均为1/n]，因此，需要找到使得P(365，k)≥0.5的最小值。
  - 令Q(365，k)表示没有重复的概率，设k个元素均不重复共有N种取值方法，则
    - $N=365\times 364\times\cdots\times(365-k+1)=\frac{365!}{(365-k)!}$
  - 那么，如果允许重复，则共有365^k种取值方法，所以不重复的概率为
    - $Q(365,k)=\frac{365!/(365-k)!}{365^k}=\frac{365!}{(365-k)!365^k}$
  - 则
    - $P(365，k)=1-Q(365，k)=1-\frac{365!}{(365-k)!365^k}$

5.2.2 散列函数的安全性要求

- 事实上，因为P(365,23)=0.5037，当人数为23时，两个人生日相同的概率既可以超过50%，可见，生日相同的概率是较大的。
- 由生日悖论可以推广为一般情形下的元素重复，即，随机变量X服从1到n之间的均匀分布X~U(1，n)，则取k个这样的随机变量，使得至少有一个重复的概率P(n，k)是多少。

5.2.2 散列函数的安全性要求

- 由生日悖论可知
  - $P(n, k) = 1 - \frac{n!}{(n-k)!n^k}$
- 则
- $P(n, k) = 1 - \frac{n \times (n-1) \times \cdots \times (n-k+1)}{n^k} = 1 - \left[\left(1 - \frac{1}{n}\right)\left(1 -\right.\right.$

5.2.2 散列函数的安全性要求

- 计算k值，使得P(n，k)>1/2，则
  - $1-e^{-\frac{k(k-1)}{2n}}=\frac{1}{2}$
- $\mathrm{In}2=\frac{k(k-1)}{2n}$
- 对于较大的k值，可以使用k^2代替k(k-1)，有
  - $\frac{k^2}{2n}\approx \mathrm{In}2$
  - $k\approx\sqrt{2n\mathrm{In}2}=1.18\sqrt{n}\approx\sqrt{n}$
- 即，在散列函数H随机生成的k个散列值中，至少有一个重复的概率P(n，k)>1/2时，k的取值约为√n。

5.2.3 MD5报文摘要算法

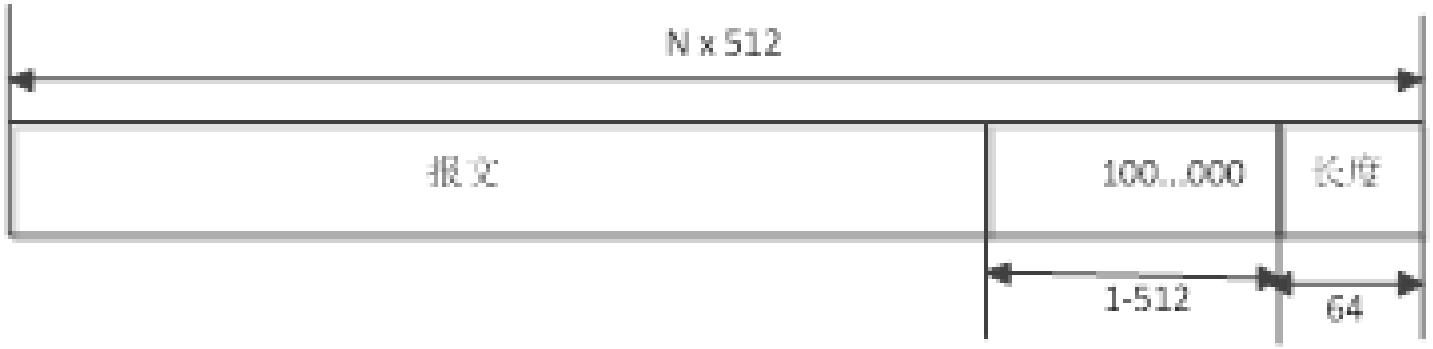
- MD5(Message Digest,Version5)报文摘要算法是经过MD2、MD3、和MD4发展而来的，它将任意长度的报文转变为128位的报文摘要，即假定P为任意长度的报文，h=MD5(P)，则h的长度为128位，其本质是将大容量信息在数字签名前被“压缩”成一种保密的格式。

5.2.3 MD5报文摘要算法

- 1 添加填充位
  - 假定报文的长度为X，首先要添加首位1、其余位为0的填充位，填充位的长度Y由下式确定。
    - (X+Y)mod512=448
  - 由于填充位是不可缺少的，因此，填充位的长度Y在1～512之间。填充位后面是64位的报文长度，报文长度X是任意的，当报文长度无法用64位二进制数表示时，取报文长度的最低64位。

5.2.3 MD5报文摘要算法

- 添加填充位和报文长度后的数据序列如图所示，它的长度是512位的整数倍。



5.2.3 MD5报文摘要算法

- 2 分组操作
  - 分组操作过程如图5.5所示。MD5将添加填充位和报文长度后的数据序列分割成长度为512位的数据段，每一组数据段单独进行报文摘要运算，报文摘要运算的输入是512位的数据段和前一段数据段进行报文摘要运算后的128位结果，第一段数据段进行报文摘要运算时，需要输入第一段数据段和初始向量IV，初始向量IV和中间结果分别为4个32位的字，分别称为A、B、C和D。初始向量的4个字的初始值如下：
    - A=67452301H
    - B=EFCDAB89H
    - C=98BADCFEH
- D=10325476H

5.2.3 MD5报文摘要算法

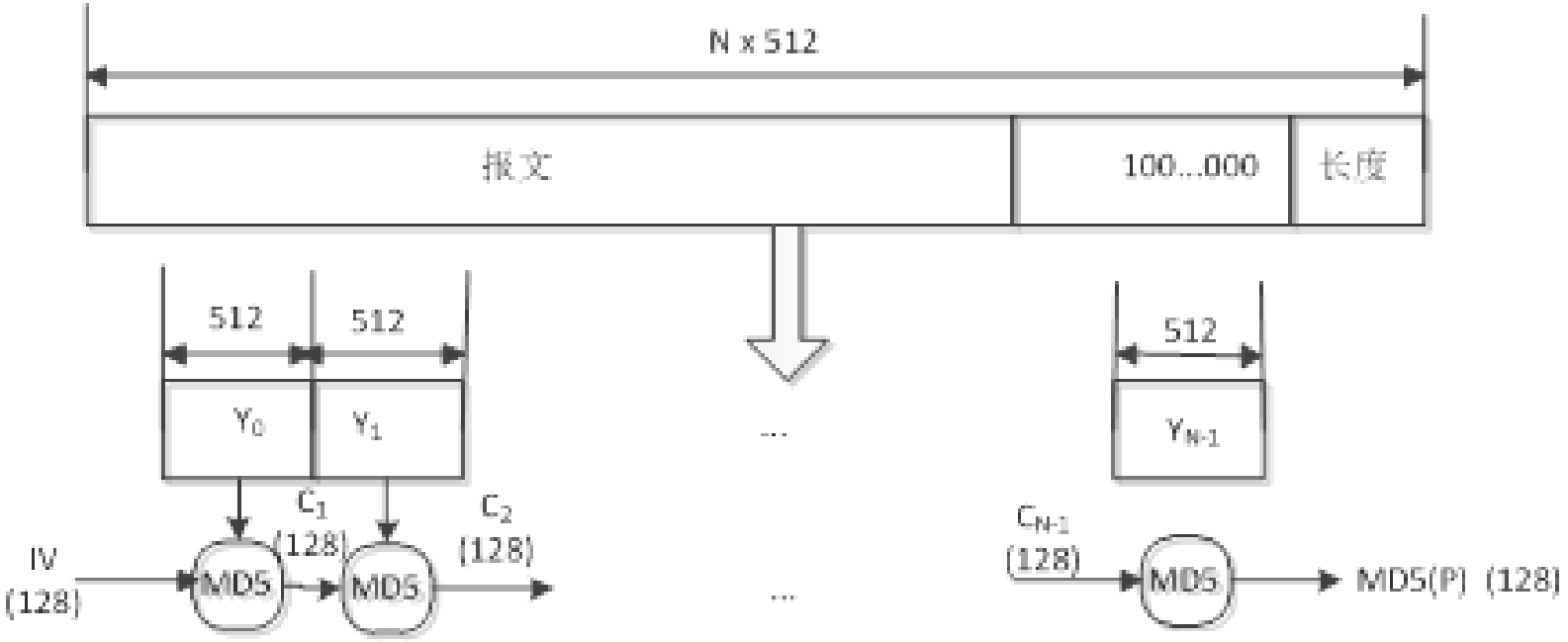


图5.5 分组操作过程

5.2.3 MD5报文摘要算法

- 3 MD5运算过程
- MD5运算过程(如图5.6所示)包含4级运算，每一级运算过程的输入是512位的数据段和上一级的运算的结果，输出是4个32位的字。第1级运算过程输入的4个32位的字是对前一段数据段进行MD5运算得到的结果或是初始向量IV。512位数据段被分成16个32位的字，分别是M[k]，0≤≤15。同时MD5也产生64个32位的常数，分别是T[i]，1≤i≤64。每一级运算过程进行16次迭代运算，每一次迭代运算都有构成数据段的其中一个字和其中一个常数参加，构成数据段的16字参加每一级的16次迭代运算，但参加每一级16次迭代运算的常数是不同的，参加第i级16次迭代运算的常数是T[j] ((i-1)x16+1≤j≤ix16)。

5.2.3 MD5报文摘要算法

- 每一级16次迭代运算的公式如下：
  - FF(a,b,c,d,M[k],s,i)； a=b+((a+F(b,c,d)+M[k]+T[i])≪ s)
  - GG(a,b,c,d,M[k],s,i)；
    a=b+((a+G(b,c,d)+M[k]+T[i])≪ s)
  - HH(a,b,c,d,M[k],s,i)；
    a=b+((a+H(b,c,d)+M[k]+T[i])≪ s)
  - II(a,b,c,d,M[k],s,i)； a=b+((a+I(b,c,d)+M[k]+T[i])≪ s)

5.2.3 MD5报文摘要算法

- 4级迭代运算公司中对应的函数如下：
- F(X,Y,Z)=X · Y　OR/X · Z (/X 表 示 对 X 非 操 作 ， · 表示与操作，OR表示或操作)
- G(X,Y,Z)=X·Z ORY ·/Z
- H(X,Y,Z)=X⊕Y⊕Z(⊕表示异或操作)
- I(X,Y,Z)=Y⊕(XOR/Z)

- 这4个函数的输入是3个32位的字，输出是1个32位的字。

5.2.3 MD5报文摘要算法

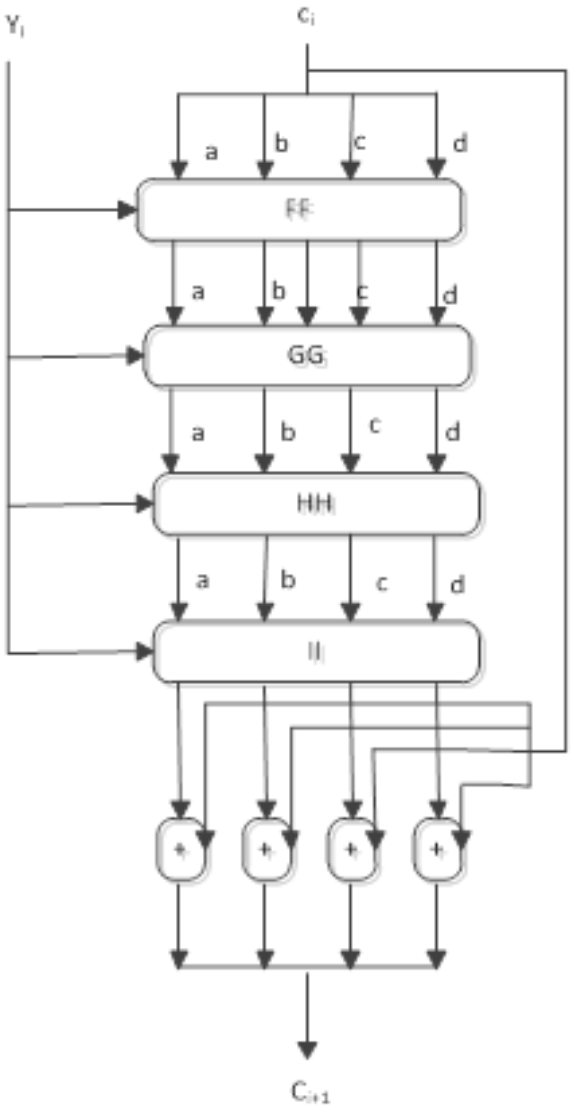


图5.6 MD5运算过程

5.2.3 MD5报文摘要算法

- 最后一级输出的4个32位字和作为这次MD5运算输入的前一段数据段的MD5运算结果逐字相加，产生这一段数据段的MD5运算结果。最后一段数据段的MD5结果作为报文的报文摘要。

5.2.3 MD5报文摘要算法

第一级运算过程的16次迭代运算			
FF(a,b,c,d,M[0],7,1)	FF(d,a,b,c,M[1],12,2)	FF(c,d,a,b,M[2],17,3)	FF(b,c,d,a,M[3],22,4)
FF(a,b,c,d,M[4]7,5)	FF(d,a,b,c,M[5],12,6)	FF(c,d,a,b,M[6],17,7)	FF(b,c,d,a,M[7],22,8)
FF(a,b,c,d,M[8],7,9)	FF(d,a,b,c,M[9],12,10)	FF(c,d,a,b,M[10],17,11)	FF(b,c,d,a,M[11],22,12)
FF(a,b,c,d,M[12],713)	FF(d,a,b,c,M[13],12,14)	FF(c,d,a,b,M[14],17,15)	FF(b,c,d,a,M[15],22,16)
第二级运算过程的16次迭代运算			
GG(a,b,c,d,M[1],5,17)	GG(d,a,b,c,M[6],9,18)	GG(c,d,a,b,M[11],14,19)	GG(b,c,d,a,M[0],20,20)
GG(a,b,c,d,M[5],5,21)	GG(d,a,b,c,M[10],9,22)	GG(c,d,a,b,M[15],14,23)	GG(b,c,d,a,M[4],20,24)
GG(a,b,c,d,M[9],5,25)	GG(d,a,b,c,M[14],9,26)	GG(c,d,a,b,M[3],14,27)	GG(b,c,d,a,M[8],20,28)
GG(a,b,c,d,M[13],5,29)	GG(d,a,b,c,M[2],9,30)	GG(c,d,a,b,M[7],14,31)	GG(b,c,d,a,M[12],20,32)
第三级运算过程的16次迭代运算			
HH(a,b,c,d,M[5],4,33)	HH(d,a,b,c,M[8],11,34)	HH(c,d,a,b,M[11],16,35)	HH(b,c,d,a,M[14],23,36)
HH(a,b,c,d,M[1],4,37)	HH(d,a,b,c,M[4],11,38)	HH(c,d,a,b,M[7],16,39)	HH(b,c,d,a,M[10],23,40)
HH(a,b,c,d,M[13],4,41)	HH(d,a,b,c,M[0],11,42)	HH(c,d,a,b,M[3],16,43)	HH(b,c,d,a,M[6],23,44)
HH(a,b,c,d,M[9],4,45)	HH(d,a,b,c,M[12],11,46)	HH(c,d,a,b,M[15],16,47)	HH(b,c,d,a,M[2],23,48)
第四级运算过程的16次迭代运算			
II(a,b,c,d,M[0],6,49)	II(d,a,b,c,M[7],10,50)	II(c,d,a,b,M[14],15,51)	II(b,c,d,a,M[5],21,52)
II(a,b,c,d,M[12],6,53)	II(d,a,b,c,M[3],10,54)	II(c,d,a,b,M[10],15,55)	II(b,c,d,a,M[1],21,56)
II(a,b,c,d,M[8],6,57)	II(d,a,b,c,M[15],10,58)	II(c,d,a,b,M[6],15,59)	II(b,c,d,a,M[13],21,60)
II(a,b,c,d,M[4],6,61)	II(d,a,b,c,M[11],10,62)	II(c,d,a,b,M[2],15,63)	II(b,c,d,a,M[9],21,64)

表5.1 MD5运算过程

5.2.4 SHA-1安全散列算法

- 安全散列算法第1版(Secure Hash Algorithm 1，SHA)和MD5非常相似，主要有两点不同。
- (1) 初始向量IV和每一段数据段经过SHA-1运算后的结果为5个32位的字，即160位，而不是128位。这样，对于任何报文X，找出另一个报文Y，X≠Y，但MD(X)=MD(Y)的可能性更低。SHA-1初始向量的前4个字的值和MD5相同，第5个字的值为：
- E=C3D2E1F0H

5.2.4 SHA-1安全散列算法

- (2) 每一级的运算过程不同，SHA-1将16个32位字数据段M[k](0 ≤ k ≤ 15)扩展为80个32位的字W[i](0≤i≤79)。每一级运算使用的函数如下：
- $F_1(X,Y,Z)=X\cdot Y \text{ OR } \overline{X}\cdot Z$
- $F_2(X,Y,Z)=X \oplus Y \oplus Z$
- $F_3(X,Y,Z)=X\cdot Y \text{ OR } X\cdot Z \text{ OR } Y\cdot Z$
- $F_4(X,Y,Z)= X \oplus Y \oplus Z$

5.2.4 SHA-1安全散列算法

- 完成每一级运算过程需要20次迭代运算，第i级运算进行的20次迭代运算如下：
- FOR j=(i-1)× 20 to $(i-1)\times 20+19$
- {
- TEMP=$S^5(A)+F_i(B,C,D)+E+W[j]+K_i$;
- E=D; D=C; C=S^{30}(B); B=A; A=TEMP;
- }
- S^5(A)表示对字A循环左移5位。

5.2.4 SHA-1安全散列算法

- 每一级运算时使用的常数K_i如下：
- K_1=5A827999H
- K_2=6ED9EBA1H
- K3=8F1BBCDCH
- K4=CA62C1D6H

5.3 认证技术

- 5.3.1 用户认证原理
- 5.3.2 信息认证技术
- 5.3.3 PKI技术
- 5.3.4 基于PKI的RBAC模型与实现

5.3.1 用户认证原理

- 当人们在住宿、求职、登机、银行存款等时，通常要出示自己的身份证（如果出国，则是护照）来证明自己的身份。但是，如果警察要求你出示身份证以证明你的身份，按照规定，警察必须首先出示自己的证件来证明自身的身份。前者是一方向另一方证明身份，而后者则是对等双方相互证明自己的身份。

5.3.1 用户认证原理

- 如果A要登录某服务器（ATM，计算机或其他类型的终端），服务器怎么知道登录的人就是A而不是其他的人呢？传统的方法（现在大多数情况下仍然是这样）是用口令来解决这个问题。A必须输入他的用户名（或ID）和口令，服务器将它们与保存在主机数据库中的口令表进行匹配，如果匹配成功，表明登录的人就是A。这里存在着一个明显的安全隐患，如果口令表被偷窥（黑客或者系统管理员都有可能），就会产生严重的后果。

5.3.1 用户认证原理

- 使用单向函数可以解决这个问题，假定服务器保存每个用户口令的单向函数值，则认证协议如下：
- ①A将自己的用户名和口令传送给服务器；
- ②服务器计算出口令的单向函数值；
- ③服务器将用户名和单向函数值与口令表中的值进行匹配。

5.3.1 用户认证原理

- 这个协议从理论上讲是可行的，可惜它很脆弱，很难经受字典式攻击，主要原因是口令一般较短，只有8个字节。对该协议最简单的改进，就是将口令与一个称作salt的随机字符串连接在一起，再用单向函数对其进行运算，而服务器保存用户名、salt值和对应的单向函数值。例如，大多数UNIX系统使用12位的salt。

5.3.1 用户认证原理

- 不幸的是，Daniel Klein开发了一个猜测口令的程序，在大约一个星期里，经常能破译出一个给定系统中的40%的口令。而David Fedmerier和Philip Karn编辑了一份包含732000个口令的口令表，表中的口令与4096个可能的salt值中的每一个值都有关联。估计利用这张表可以破译出一个给定系统口令表中的30%的口令。这表明，增加salt的位数不能解决所有的问题，它可以防止对口令表实施字典式攻击，但不能防止对单个口令的攻击。

5.3.1 用户认证原理

- 有一种称为S/KEY的一次使用口令的方法，它基于单向函数的安全性。用户A预先产生一串口令：A首先输入随机数R，服务器使用单向函数f和R，计算出N (例如N=100)个口令$P_k(1 \le k \le N)$，其中$P_0=R$，$P_k=f(P_{k-1})$，服务器将P_{N+1}和A的用户名一起保存在口令表中，并将P_k依次打印出来交由A保存。A在第一次登录时使用P_N作为自己的口令，服务器则根据用户名和$f(P_N)$在口令表中寻找匹配，如果匹配成功，那么证明A的身份是真的，同时服务器用P_N代替口令表中的P_{N+1}，用户A则将P_N从自己的口令串中划掉。

5.3.1 用户认证原理

- A在每一次登录时总是使用下标最大的那个口令，并在使用后把它从自己的口令串中划掉；服务器在每一次认证时总是计算$f(P_k)$，再在口令表中寻找匹配，并用当前的口令P_k替换掉先前的P_k+1。当A的口令串用完时，A必须请求服务器为自己再产生一串口令。该协议的优点是，服务器中的口令表以及通信线路中传输的口令对于攻击者来说毫无意义。其缺点是异地使用起来不方便。

5.3.1 用户认证原理

• 另一种方法是使用非对称密钥系统，登录系统时，按如下协议进行认证：

- ①服务器发送一个随机数R给A；
- ②A用自己的私钥对R加密，得到R_1并将它和用户名一起回传给服务器；
- ③服务器根据用户名在数据库中找到A的公钥，用它解密R_1，得到R_2；
- ④如果R_2=R，则A登录成功。
- 该协议可以有效地对付窃听或偷窥，但A所使用的终端必须具有计算的能力。

5.3.2 信息认证技术

- 信息认证技术是网络信息安全技术的一个重要方面，它用于保证通信双方的不可抵赖性和信息的完整性。在Internet深入发展和普遍应用的时代，信息认证显得十分重要。
- 例如，在网络银行、电子商务等应用中，对于所发生的业务或交易，我们可能并不需要保密交易的具体内容，但是交易双方应当能够确认是对方发送（接收）了这些信息，同时接收方还能确认接收的信息是完整的，即在通信过程中没有被修改或替换。
- 再如，在电子政务应用中，通过网络发送（传输）信息（数字文件），此时接收方主要关心的是信息真实性和信息来源的可靠性。

5.3.2 信息认证技术

- 1 基于私钥密码体制的信息认证
- 假设通信双方为A和B。A、B共享的密钥为K_{AB}，M为A发送给B的信息。为防止信息M在传输信道被窃听，A将M加密后再传送，如图5.7所示。

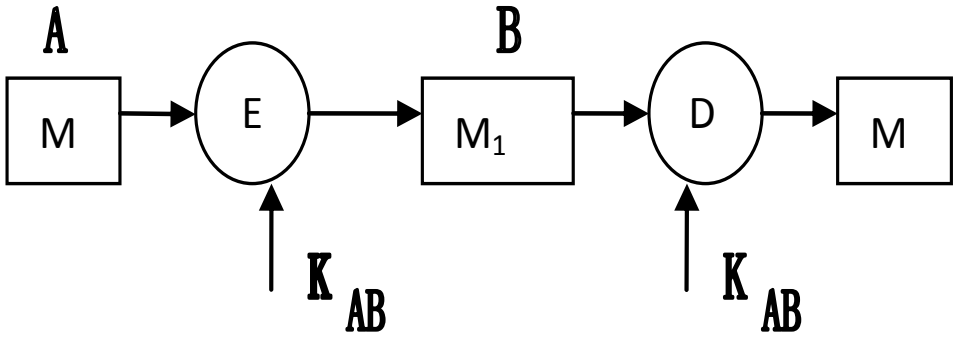


图5.7基于私钥的基本信息认证机制

5.3.2 信息认证技术

- 由于K_{AB}为用户A和B的共享密钥，所以用户B可以确定信息M是由用户A所发出的。因此，这种认证方法可以对信息来源进行认证，而且它在认证的同时对信息M也进行了加密。这种方法的缺点是不能提供信息完整性的鉴别。

5.3.2 信息认证技术

- 通过引入单向hash函数，可以解决信息完整性的签别检测问题，如图5.8所示。

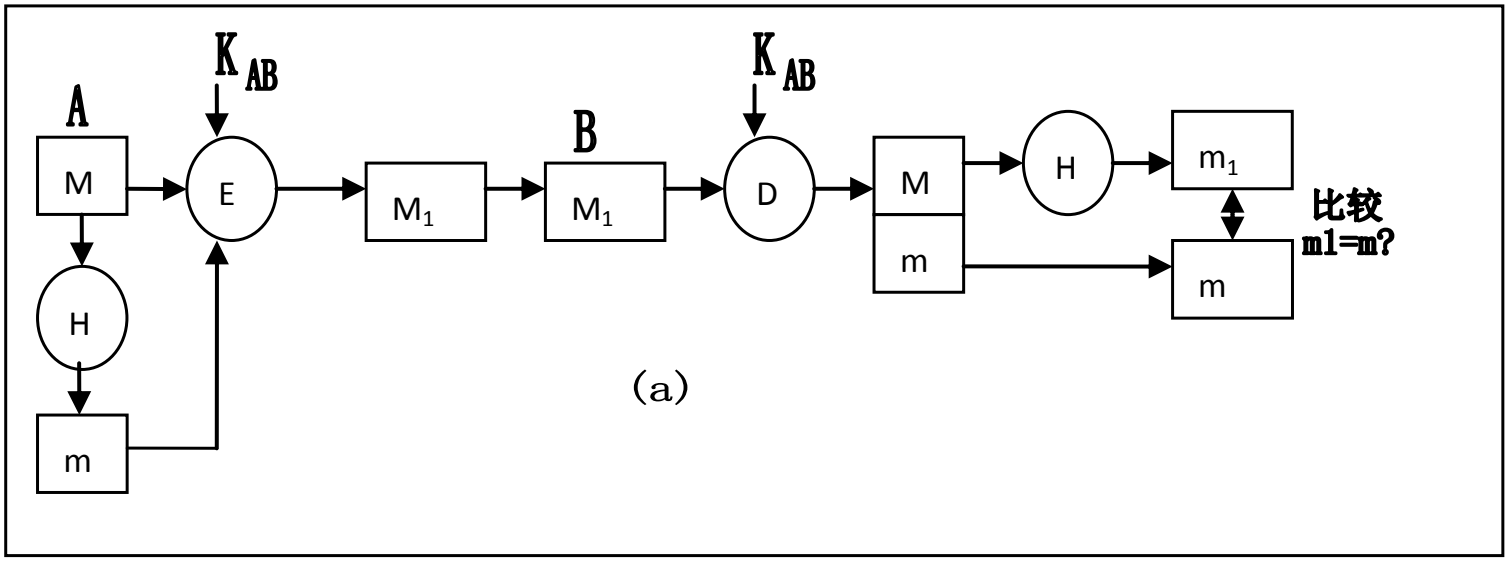


图5.8(a) 基于私钥的信息认证机制

5.3.2 信息认证技术

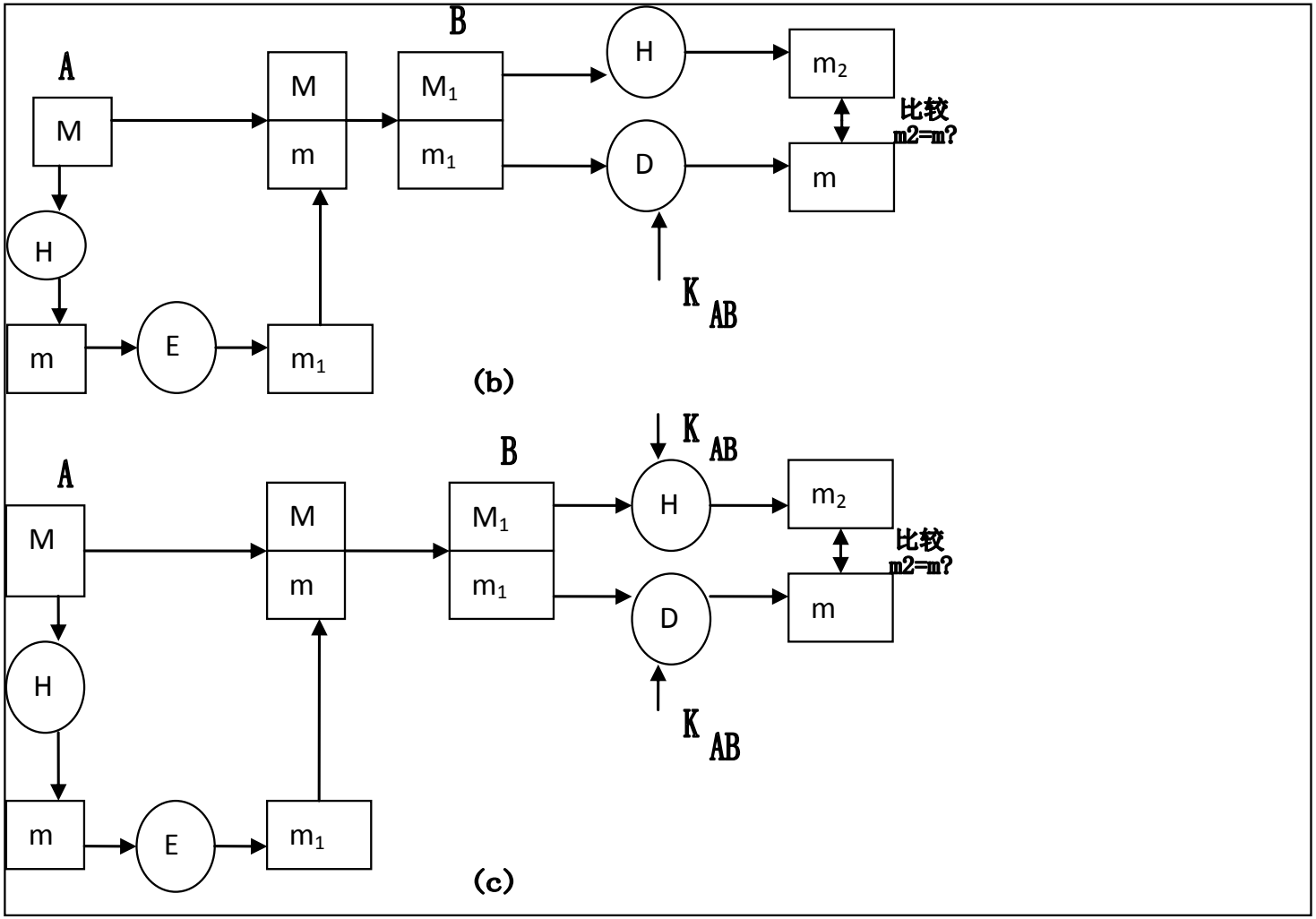


图5.8(b)(c) 基于私钥的信息认证机制

5.3.2 信息认证技术

- 在图5.8（a）的信息认证机制中，用户A首先对信息M求hash值H(M)，然后将M H(M)加密后传送给用户B。用户B通过解密并验证附于信息M之后的hash值是否正确。图5.8 (b)的信息认证机制和5.8 (a)的信息认证机制唯一不同的地方是对hash值加密。
- 而在图5.8 (c)的信息认证机制中，则使用一种带密钥的hash函数H（H可取为ANSI X9.9标准中规定的DES CBC模式，K_{AB}为DES的加密密钥，DES全称为Data Encryption Standard，即数据加密标准，是一种使用密钥加密的块算法），函数H以M和K_{AB}为参数。图5.8 (c)的信息认证机制与图5.8 (b)的信息认证机制主要区别在于产生信息认证码（MAC）的方式不同。图5.8给出的三种信息认证方案均实现信息来源和完整性的认证。

5.3.2 信息认证技术

- 基于私钥的信息认证机制的优点是速度较快，缺点是通信双方A和B需要事先约定共享密钥K_{AB}，而且如果用户A需要与其他n个用户进行秘密通信的话，那么用户A需要事先与这些用户约定和妥善保存n-1个共享密钥，这本身就存在安全问题。

5.3.2 信息认证技术

- 2 基于公钥体制的信息认证
- 基于公钥体制的信息认证技术主要利用数字签名和hash函数来实现。假设用户A对信息M的hash值H(M)的签名为SigSA(d_A,（H（m））），其中d_A为用户A的私钥。用户A将M SigSA（d_A,H（m））发送给用户B，用户B通过A的公钥来确认信息是否由A所发出，并且通过计算hash值来对信息M进行完整性鉴别。如果传输的信息需要保密，那么用户A和B可以通过密钥分配中心（KDC）获得一个共享密钥K_{AB}，A将信息签名和加密后再传送给B，如图5.9所示。

5.3.2 信息认证技术

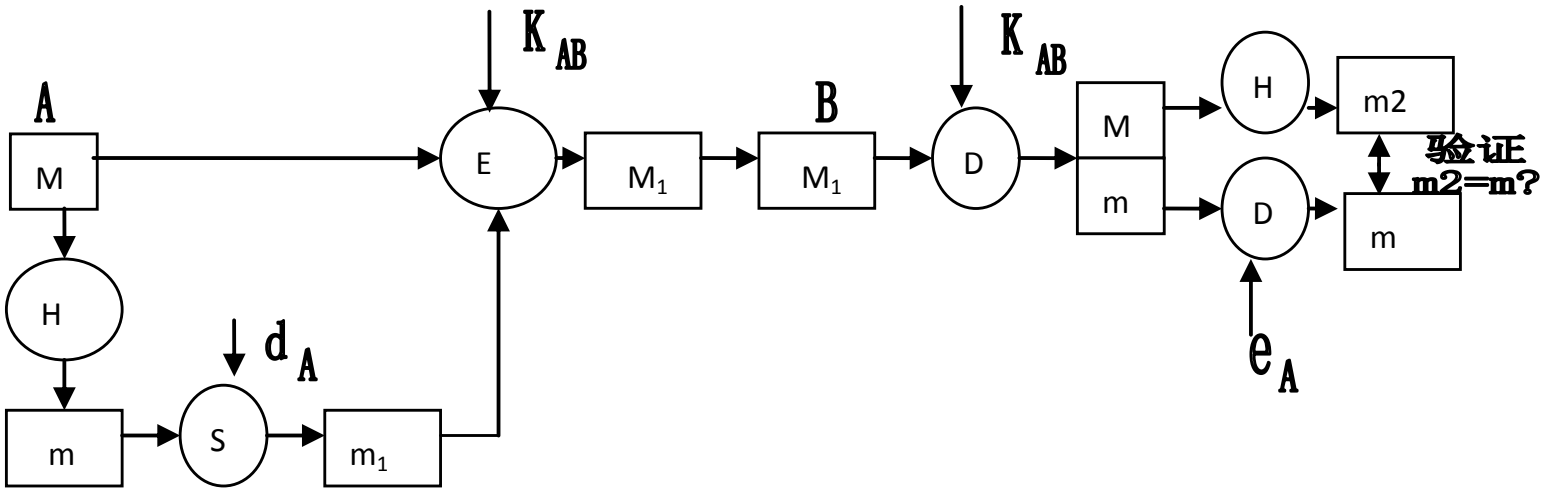


图5.9基于公钥的信息认证机制

从图5.9可知，因为只有用户A和B拥有共享密钥K_{AB}，所以B能够确信信息来源的可靠性和完整性。

5.3.3 PKI技术

- 网络中的信息一方面要允许值得信赖的用户访问，另一方面要防止非法用户以及黑客的破坏和入侵。因此，认证与授权一直是现代网络信息系统中需要解决的很重要的问题。在过去的几年中，公钥基础设施（PKI：Public Key Infrastructure）已经成为网络信息系统中不可或缺的认证系统。PKI通过公钥加密手段，以公钥证书为核心，提供了在线身份认证的有效手段。

5.3.3 PKI技术

- 在20世纪1976年Diffie和Hellman提出公钥密码的思想：试图寻找一种密码系统，使得在公钥公开的情况下仍然无法推算出私钥，这样公钥便可以安全发布出去。为此公钥证书将用户的身份与其公钥绑定在一起，作为用户在网络中的身份证明。证书权威中心CA负责公钥证书的签发、撤销与维护。PKI实现了信息交换的保密性、抗抵赖性、完整性和有效性等安全准则，总之，PKI在网络应用中解决了“他是谁”的问题。

5.3.3 PKI技术

- 1 PKI 概述
- 公开密钥体制早期主要是针对开放式大型因特网的应用环境而设计的，在这种网络环境中需要有一个协调的公开密钥管理机制，来保证公开密钥的可用性和可靠性。
- 公开密钥的管理一般是基于证书机构（CA，Certificate Authority）,即建立一个通信双方都信任的第三方机构来管理通信双方的公开密钥。在现代网络环境中，这种基于证书机构可能许多个，这些证书机构可以存在信任关系，用户可通过一个签名链去设法验证任一个证书机构颁发的证书。

5.3.3 PKI技术

- PKI的核心是对信任关系的管理。第三方信任与直接信任是所有网络信息安全实现的基础。所谓第三方信任，是指两个人或实体之间可以通过第三方间接地达到彼此信任。
- 举例：如果两个陌生人都与同一个第三方彼此信任并且第三方也担保他们的可信度时，这两个陌生人就可以做到彼此信任。当在很多人或实体中建立第三方信任时，就需要有一个权威机构来确保信任度。

5.3.3 PKI技术

- 数字证书又称电子证书，起着标志证书持有者身份的作用。证书持有者可以通过它进行安全的通信、事务处理以及贸易等活动。目前有关部门可以提供如下类型的数字证书：
- 一是用来进行各种网上事务处理活动后事务型数字证书。二是用来进行网上安全电子交易的交易型数字证书。另外，为了使广大的网络用户以及电子商务爱好者熟悉数字证书的使用，还提供免费型数字证书。

5.3.3 PKI技术

- PKI通过使用公开密匙技术和数字证书来保证网络系统信息安全并负责验证数字证书持有者身份。举例：如某企业可以建立PKI体系来控制对其计算机网络的访问。PKI让个人或企业用户安全地从事其商业行为。
- 企业员工以及网络用户可以在互联网上安全地发送电子邮件而不必当心其发送的信息被非法的第三方（竞争对手等）截获。当然，企业也可建立其内部网络站点，只对其信任的客户发送信息。

5.3.3 PKI技术

- 2 PKI 的目的和特点
- 当前人们认可PKI是国际上较为成熟的解决开放式互联网络信息安全需求的一套体系，而且还在发展之中。PKI体系支持：
- （1）身份认证；
- （2）信息传输、存储的完整性；
- （3）信息传输、存储的机密性；
- （4）操作的不可否认性；

5.3.3 PKI技术

- “PKI基础设施”的目的就是：只要遵循必要的原则，对于不同的实体就可以方便地使用基础设施提供的服务。PKI公开密匙基础设施就是为整个组织提供安全的基本框架，可以被组织中任何需要安全的应用和对象使用。
- PKI公开密匙基础设施具有同样的特性:
- （1）具有易用的、众所周知的界面；
- （2）基础设施提供的服务是可预知的且一致有效的；
- （3）应用设施无需了解基础设施是如何提供服务的。

5.3.3 PKI技术

- 3 PKI 的功能
- 一个完整的PKI系统应具备以下主要功能：根据X.509标准发放证书，证书与CA产生密匙对，密钥备份及恢复，证书，密钥对的自动更换，加密密钥及签名密钥的分隔，管理密钥和证书，支持对数字签名的不可抵赖性，密钥历史的管理，为用户提供PKI服务，如网络用户安全登陆、增加和删除用户、恢复密钥、检验证书等。

5.3.3 PKI技术

- 4 PKI 的基本组成
- 一个典型的PKI系统由五个基本的部分组成：证书申请者、注册机构、认证中心、证书库和证书信任方。
- 其中，认证中心、注册机构和证书库三部分是PKI的核心，证书申请者和证书信任方则是利用PKI进行网上交易的参与者。PKI在实际应用上是一套软件、硬件系统和安全策略的集合，它提供了一整套安全机制，使网络用户在不知道对方身份或者分布得很广的情况下，以证书为基础，通过一系列的信任关系进行网络通信和电子商务交易。

5.3.4 基于PKI的RBAC模型与实现

- 1基于角色访问控制模型
  - 一种基于角色访问控制（RBAC）核心思想是权限并不直接分配给用户，而是分配给角色，角色是分配权限的一种间接手段 。在RBAC模式下，首先定义出若干个角色，这些角色代表着企业中不同的职务，例如经理、秘书、职员等等，角色与一定的权限相对应。然后将角色分配给用户，用户可以属于一个或者多个角色，这样用户就继承了该角色所具有的权限。

5.3.4 基于PKI的RBAC模型与实现

- 将PKI引入企业网络，可以解决企业网络的认证问题。在PKI模式下，用户申请证书后，只需记住一个密钥（保护私钥的密钥）就可以访问系统中的不同资源，真正实现了“一次认证全局通用”的访问模式。
- 实现过程分为三个阶段：身份认证阶段、访问控制阶段和服务提供阶段，具体过程如下：

5.3.4 基于PKI的RBAC模型与实现

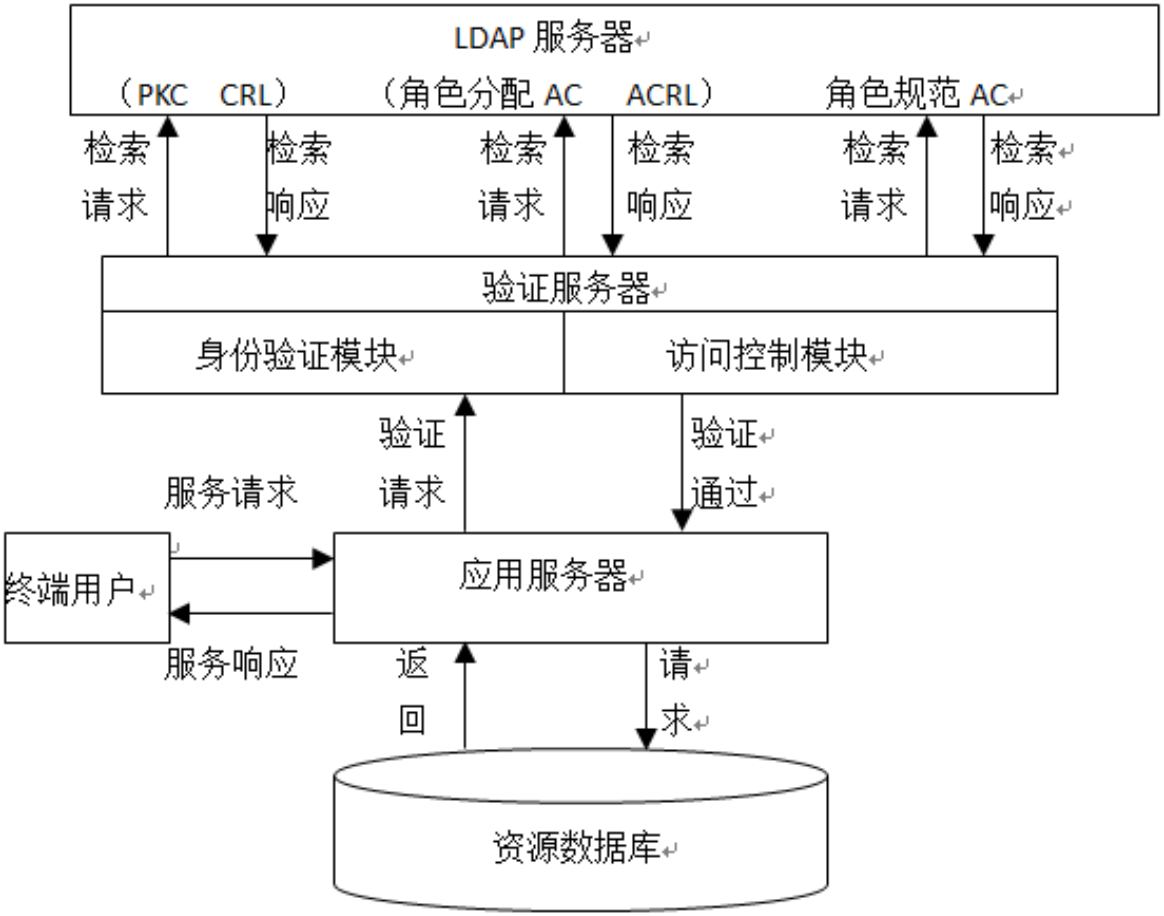


图 5.10 RBAC 访问控制模型

5.3.4 基于PKI的RBAC模型与实现

- 第一阶段：身份认证阶段。
- 1）用户将服务请求与其公钥证书PKC签名后提交给应用服务器，应用服务器将它们以验证请求的方式转交给验证服务器；
- 2）验证服务器收到上述信息后，身份认证模块通过PKC库验证用户公钥证书的真实性（库中是否存在该证书），再通过CRL库验证证书的有效性（该证书是否被吊销），然后再利用用户公钥证书中的公钥对用户数字签名进行验证；
- 3）以上三项验证如果有一项未通过，则向应用服务器发送验证通不过的拒绝服务消息，否则身份认证模块通过用户的身份认证。

5.3.4 基于PKI的RBAC模型与实现

- 第二阶段：访问控制阶段
- 身份认证通过后，系统进入访问控制阶段，访问控制阶段由验证服务器的访问控制模块来完成，其过程如下：
- 1）访问控制模块根据终端用户公钥证书PKC中的证书惟一标识（即证书ID），从LDAP目录服务器中检索该用户的角色指派属性证书。
- 2）访问控制模块对该角色指派AC进行真实性和有效性的验证，具体包括：证书是否在有效期内，是否被AA签名（以防篡改证书内容），是否在属性证书吊销列表中，如果上述内容有一项未通过，那么就向应用服务器返回验证通不过的响应信息，否则从角色指派AC中获取角色属性值。

5.3.4 基于PKI的RBAC模型与实现

- 3）访问控制模块利用该角色属性值从LDAP目录服务器中检索角色规范AC库，获取与该角色对应的角色规范证书，然后对该证书进行相关验证，验证通过后，从该证书中获取权限集合，并检查权限集中每个权限是否在授权验证列表（ACL）中。
- 4）访问控制模块将用户请求与权限集合相比较，判定该用户请求是否在权限允许的范围之内，不在权限范围之内，就向应用服务器返回拒绝服务的响应信息，否则通过应用服务器响应用户请求。

5.3.4 基于PKI的RBAC模型与实现

- 第三阶段：服务提供阶段。
- 该阶段的工作主要由应用服务器和资源数据库来完成，应用服务器在收到验证服务器的通知后，根据用户访问请求，对资源数据库进行处理，并将处理结果集返回给该终端用户。

5.3.4 基于PKI的RBAC模型与实现

- 在企业网络信息系统中引入PKI机制，以PKI作为身份认证，并采用基于角色的访问控制策略，实现安全访问控制，具有灵活、方便等等特点。企业中人员的调动和其他变动，以及人员职务的升迁，职权的变动非常容易在本方案中得到体现，只需要在身份证书和属性证书上作少许改动就可以了，所以适应性很强，也非常符合一般企业的管理模式和管理习惯。

思考题

- 5.1数字签名应该具有哪些性质？
- 5.2试问数字签名的目的是什么？
- 5.3试问直接数字签名方案存在哪些共同的弱点？
- 5.4试问现在数字签名有哪些相关标准？
- 5.5 什么是安全散列函数？安全散列函数有哪些应用？
- 5.6 散列函数有哪些安全性要求？
- 5.7 什么是MD5报文摘要算法？
- 5.8 使用非对成秘钥系统登录系统时，其协议怎样进行认证？
- 5.9 比较基于私钥和公钥的信息认证机制的优缺点？
- 5.10 PKI的组成分为哪几大部分？它们的主要功能各是什么？

- 谢谢！