第 3 章 安全协议

3.1 初级安全协议

尽管我们已经在第 2 章给出了许多安全的密码算法。但是，在计算机系统中如何使用这些算法，使得计算系统安全可靠是本章需要解决的问题。而且，并不是使用了安全的密码算法就能保证计算机系统的安全，如果密码算法使用不当，会给计算系统构成重大的安全危险。

3.1.1 协议与密码协议

定义 3.1 所谓协议是指双方(或多方)通过执行一系列步骤来完成一项任务。这里的“双方(或多方)”是指协议至少包括两个人，一个人完成的事情不构成协议；“一系列步骤”是指协议是一个由始到终的序列，每一步必须执行，在前一步没有执行时，不能执行后一步；“完成一项任务”是指必须做一些事情，不能徒劳无功。

1. 协议必须满足以下特点

1)协议中的各方必须了解协议，明白自己在协议中所执行的步骤;

2)协议中的各方必须同意执行此协议;

3)协议中的每一步骤的定义必须清楚，不能引起误解;

4)协议必须完整，对每一步骤中的各种情况都要有具体的规定。

密码协议是指用于安全系统的协议，参与协议的各方可能是朋友或可信赖的人，还可能是敌人或完全互相不信任的人。密码协议通常是通过密码算法来完成某种秘密性，或相互确定身份，或共同签署一份文件等。通过密码协议，可以使得计算机网络系统中的互相不信任的人建立协议，完成某些任务。例如：网上银行，就可以满足储户的各种要求，节省开支。

在日常生活中，我们经常使用一些协议。例如：玩扑克、投票、电话购物等。所有这些协议人们都没有仔细考虑过它们的安全机制，执行这些协议大都基于人们面对面的真实性和完全性，当这种情形发生在计算机网络上时，许多简单的协议也会出现安全问题。

那些认为计算机网络系统是安全的人是太天真了！将一个计算机系统的安全建立在可信赖的系统管理员身上是不可取的。我们认为，对计算机系统的任何人员都不能轻信，包括计算机系统的设计者。当然，计算机系统中的绝大多数人是诚实的，但是要防止极少数人的破坏和捣乱。通过建立比较完备的协议，可以防止和发现不诚实的人，防止欺骗和抵赖等不法行为。

2. 下面介绍三种基本协议

(1)仲裁协议　协议假设：A(Alice)、B(Bob)为互不信任的协议双方，C 为中间人，C 充当仲裁者，C 在协议中不偏向 A、B 任何一方，A、B 都接受这样的事实，即 C 说的都是对的，做的都是正确的，涉及 C 的部分是完备的。仲裁人 C 可以帮助互不信任的双方完成协议。

例如：B 向 A 购买一个东西，B 怕钱给 A 后，A 不把东西给 B；A 怕将东西给了 B 后，B 不向 A 付款；这时，一个公证人 C 可以帮助完成 A、B 之间的协议。B 把钱交给 C；A 把东西也交给 C，C 将东西交给 B，C 将钱交给 A。C 不会将钱和东西全部占为己有，也不会偏袒一方，合谋欺骗另一方。

现实生活中的仲裁者如律师、公证员或法庭等，是一些有声望的人或机构，如果将这类协议移植到计算机网络中，会出现以下问题：

1)人们不会轻易相信网络上的不曾露面的第三方(仲裁者)；

2)网络必须负担仲裁者的费用；

3)仲裁者执行协议有延迟；

4)网络规模扩大时，仲裁者是网络效率发挥的瓶颈；

5)仲裁者成为最容易遭受安全攻击的目标。

尽管如此，有中心的仲裁者这类协议是目前计算机系统的主要协议之一，银行等金融部门仍在继续使用这种协议，这主要是依靠强大的中心计算机系统来完成的。

(2)裁决协议　为了减少付给仲裁者的费用，提高协议执行效率，可以将上面的协议拆成两个低级子协议：执行协议与判决协议。

执行协议是指协议双方按照自己的步骤，在正常状态下各自执行协议中的项目，一般不会出现问题，例如合同签订后，对合同的执行。但是，一旦协议执行中出现问题，必须有一个公正、可信的第三方来裁决，好比日常生活中的法官。

例如，合同签署协议：

1)执行协议：A、B 谈判合同条款；

A 签署这个合同；

B 签署这个合同。

2)判决协议：A、B 出现在法官面前；

A 出示她的证据；

B 出示他的证据；

法官根据证据裁决。

这类协议的特点是，只有发生争议，法官才被请出裁决；如果有人欺骗，中立的第三方可以通过搜集数据进行判断是否有人欺骗。目前，已经设计出了确定欺骗者身份的协议；所以，可以使得一旦有人欺骗，裁决者可以马上揭穿其骗局，使人们保持诚实。这一点在以后的协议中会体会到。

(3)自动执行协议　自动执行协议是最好的一种协议。协议只包含协议执行各方，不需要第三方来裁决或仲裁，协议本身具有公平性，也不会产生争端。如果协议中的一方试图欺骗，其他各方就会马上发现并终止协议；无论欺骗者采取什么方法，不会得到任何有价值的东西。例如后面的零知识证明协议。

3.1.2　攻击协议

任何协议都会面临各式各样的攻击和威胁，所以在计算机系统中，初始的协议设计必须考虑它们所要面临的威胁，以避免遭到攻击。攻击者会对计算机系统中的密码算法、密钥管理、协议等环节进行攻击，我们在此假设计算机系统中的密码技术是安全的，仅仅讨论对协议的攻击。

从攻击协议的方法和目的来看，可以把对协议的攻击分为两类：

(1)被动攻击　攻击者只能被动地窃听协议的部分或全部内容，然后将窃听到的内容进行密码分析，得到信息。攻击者不能影响协议，好比密码攻击中的唯密文攻击。

只要使得协议中的各个传输环节以密文方式发送，就可有力地挫败这种攻击。

(2)主动攻击　攻击者通过改变协议，删除、更改协议信息，破坏通信信道达到攻击目的；攻击者通过对协议的主动破坏和干扰，来达到其非法目的。

主动攻击产生的后果要比被动攻击严重得多。主动攻击可能造成信息泄漏、丢失、冒名、抵赖、系统效率降低、破坏原始信息等严重后果，并得到非法授权。

主动攻击除了攻击方法的多样性以外，攻击者还可能是第三方入侵者，也有可能是协议中的某一方或几个方面联合，通过合谋骗取合法用户的信息；他们在执行协议过程中可能撒谎，也可能根本不遵守协议来达到获取额外信息和破坏协议的目的。当然，如果系统中的多数人是主动攻击者，就很难保证系统的安全性。设计防止主动攻击的协议是很复杂的，使用能够判断有主动攻击者的协议对保护合法用户很重要，当前已经有了这类协议，可以有效挫败主动攻击。

3.1.3 保密通信与密钥交换协议

我们多次强调，在计算机系统中采用密码技术是很重要的；而在密码技术中，密钥的管理尤为重要。以对称密码系统为例，A 和 B 的通信协议为：

1)A、B 同时选用相同的密码系统；

2)A、B 选用相同的密钥；

3)A 用密钥加密信息，得到密文；

4)A 将密文发送给 B；

5)B 用密钥解密，得到明文。

1. 此通信协议的不安全隐患有两点

(1)第三方攻击者在密码算法是安全的前提下可能直接攻击 1)和 2)，获取密钥。所以，在对称密码系统中，密钥的传递和交换是最重要的。

(2)A 或 B 可能将密钥交给第三方，使得信息泄漏；而 A、B 之间无法向仲裁者确定是谁在捣鬼。

2. 传统密码系统存在的问题

(1)一旦密钥泄漏(偷窃、猜测、逼迫交出、行贿受贿等)，那么，攻击者就可解密所有密文，还可能假装两人中的其中一人，产生虚假信息愚弄另一人。经常改变密钥可以减轻危险。

(2)密钥必须秘密分配，必须派安全可靠的信使完成此重任，信使不能出卖密钥；敌方可能会对信使重点攻击，信使面临巨大的威胁和风险。

(3)在计算机网络系统中，随着用户数目的增加，密钥总数会急剧增加，这就增加了网络负担。适当控制系统用户规模是必要的。

使用公开密钥密码系统可以解决以上问题。

3. 使用公开密钥密码系统

在公开密钥密码系统中，A 发送信息给 B 的过程：

1)A、B 共同选择一公开密钥密码系统；

2)B 将其公开密钥发送给 A；

3)A 用 B 的公开密钥加密信息，得到密文，并将密文发送给 B；

4)B 用其秘密密钥解开 A 发送来的密文；

为了应用方便，通常将所有用户的公开密钥放在一个公开、可靠、不可篡改的数据库中，所有用户在通信时可以事先到数据库中获得对方的公开密钥。通常将存放公开密钥的地方称为密钥分配中心(KDC)，它负责用户公开密钥的真实性，往往需要中心的数字签名来保证其真实可靠性。

然而，到目前为止，所有安全的公开密钥密码系统都存在计算量大和运算速度慢的缺陷，无法实现保密通信的要求。通过设计专用的芯片，可以提高加密速度，但是又有成本太大的不足，所以通常使用混合密码协议。

4. 混合密码协议

通常来说，对称密码算法比非对称密码算法快 1 000 倍，实际通信系统使用公开密

钥体制交换会话密钥，使用会话密钥通过对称密码体制加密明文。

(1)A、B 使用相同公钥系统，获得同一密钥(使用密钥交换协议)。

(2)使用相同的对称密码算法和共同的密钥进行保密通信。

通过上面的保密通信协议可以看出，密钥交换是保密通信的前提和条件，也是保证通信安全的关键。采用对称密码体制和非对称密码体制都可进行密钥分配与交换。

5. 对称密码体制密钥分配协议

前提：KDC 掌管所有用户的密钥，每个用户拥有自己的密钥。使用对称密码算法，如 DES，IDEA 等。

(1)A 向 KDC 发出与 B 进行保密通信的请求。

(2)KDC 随机产生一会话密钥 k，用 A 的密钥加密 k，并将加密结果 c 发送给 A，KDC 将 k 及 A 的身份用 B 的密钥加密，也将加密结果 c' 发送给 A。

(3)A 解密 KDC 发送的密文 c。

(4)A 将 KDC 发送的密文 c' 及 A 的身份给 B。

(5)B 用自己的密钥解密，得会话密钥。

(6)A、B 拥有相同的会话密钥，用此密钥进行保密通信。

此协议的安全性依赖于 KDC 的绝对安全。由于 KDC 拥有所有用户的密钥，所以对 KDC 必须严格保护。

6. 公开密钥密码系统的密码交换协议

前提：A、B 使用共同的公开密码体制。

(1)B 把他的公开密钥给 A。

(2)A 随机产生会话密钥 k，用 B 的公开密钥加密，将密文给 B。

(3)B 用其秘密密钥解密，得到会话密钥 k。

(4)A、B 可以使用 k 进行保密通信。

使用离散对数型的单向函数同样可以达到密钥交换的目的。以有限域上的离散对数问题为例，协议如下：

前提：系统公开有限域 $GF(P)$，生成元 g。

(1)A 随机产生一数 x，计算 $g^x \bmod P$ 并将结果发送给 B。

(2)B 随机产生一数 y，计算 $g^y \bmod P$，并将结果发送给 A。

(3)A 利用其 x 和 $g^y \bmod P$，计算 $g^{xy} \bmod P$。

(4)B 利用其 y 和 $g^x \bmod P$，计算 $g^{xy} \bmod P$。

(5)A、B 拥有共同的会话密钥 $g^{xy} \bmod P$；。

同样使用离散对数难题，爱格玛尔(ElGamal)密码体制是另一种自带密钥交换功能的密码系统，协议如下：

前提：B 的公开密钥为有限域 $GF(P)$，生成元 g，$g^x \bmod P$，秘密密钥为 x，A 欲发送消息 m 给 B。

(1)A 随机选择工作密钥 k，计算并发送($g^k \bmod P$, $mg^{xk} \bmod P$)给 B。

(2)B 通过其秘密密钥 x 计算 $g^{xk} \bmod P$，求逆可恢复出明文 m。

这里的工作密钥 k 是通过 $g^{xk} \bmod P$ 来完成传递与商定的。

7. EKE 密钥交换协议

此协议使用公开密钥系统和秘密密钥系统，提供会话密钥和鉴别。A、B 共同使用秘密口令 p、秘密算法 E_1 和公开密钥算法 E_2。

(1)A 随机产生公开密钥 k'，发送：A，$E_1(k', p)$给 B。

(2)B 解开 k'，随机产生一会话密钥 k，用公开密钥 k' 和秘密口令 p 加密 k，给 A，$E_1(E_2(k, k'), p)$。

(3)A 解密 k，随机产生一随机数 R_A，用 k 加密发送给 B，$E_1(R_A, k)$。

(4)B 解密 R_A，随机产生 R_B，加密两串给 A，$E_1(R_A R_B, k)$。

(5)A 恢复 R_A，比较是否一致，若一致，用 k 加密 R_B，$E_1(R_B, k)$。

(6)B 解密 R_B，比较是否与以前的一致，如果一致，则 A、B 具有共同的密钥 k。

其中公开密钥算法 E_2 可以是 RSA 体制、爱格玛尔(ElGamal)算法及狄菲—赫尔曼(Diffie - Hellman)算法，秘密密钥算法可以使用 DES、IDEA 等安全分组算法。

可以将 EKE 密钥交换协议修改成加强型的协议，会话密钥 k 仅起密钥交换作用。这样，可以防止破译者利用旧的 k 所进行的攻击。修改如下：

将第(3)步中 A 产生的 R_A 增加到 R_A、S_A，发送 $E_1(R_A S_A k)$给 B；在第(4)步，B 同样产生 R_B、S_B，发送 $E_1(R_A, R_B, S_B, K)$给 A，这样 A 和 B 就会拥有共同密钥 $S_A \oplus S_B$。

这里的协议起了一种秘密放大的作用。使用简单的口令和公钥系统，可以使得会话密钥有足够的长度，保证系统的整体安全。

3.1.4 多级密钥与密钥分存协议

RSA 密码体制的一种推广形式是将加解密密钥拆成多个密钥，其中一部分人解密，另一部分人加密。

选择两大素数 p, q，$n = pq$，选择 t 个 k_i，使得 $k_1 k_2 \cdots k_t = 1 \bmod((p-1)(q-1))$，则有：对任意消息 m，有 $m^{k_1 k_2 \cdots k_t} \bmod n = m$。

例如：假设 $t = 5$，利用 k_1、k_3 加密，可利用 k_2、k_4、k_5 解密。

$$C = m^{k_1 k_3} \bmod n, \qquad m = C^{k_2 k_4 k_5} \bmod n,$$

另一种密钥分配目的是秘密广播。假设中心 A 想通过广播发送一个消息 m，使得听众中的一部分能够解密消息，而其他人什么也得不到。秘密广播的解密方案有三种。

1. 第一种解密方案

假设 A、B、C、D、E，5 个人，A 欲发消息 m 给 B、C，而 D、E 什么也得不到。

(1)A 与 B、C、D、E 共享一组密钥(秘密或公开)。

(2)A 产生一随机密钥 K，用 K 加密消息 m，再用 B、C 各自的密钥加密 K，将所有的结果广播出去。

(3)B 和 C 解密有意义的 K，用 K 可解密 m，而 D 和 E 不能。

由于 B 和 C 要解密所有的密文，才能获得有意义的明文 m，如果 A 不介意消息是发给谁的话，可以附加上密文的标志 B 或 C。B 和 C 就可避免多余的解密过程。

2. 第二种解决方案

A 和 B、C、D、E 共享各自的秘密密钥 K_B，K_C，K_D，K_E，用随机密钥 K 加密消息 m，A 选择计算 R，使得

$$R \bmod K_B = K$$
$$R \bmod K_C = K$$
$$R \bmod K_D = O$$
$$R \bmod K_E = O$$

A 发送 R 和用 K 加密的消息 m，只有 B 和 C 能够恢复 K，其他人不行。

3. 第三种解密方案

这种方法是利用在第 2 章讲的秘密分存方案。我们在第 2 章中介绍了沙米尔(Shamir)的多项式插值方案，保证了(K, n)分存的特点，使得任意 K 个人可以恢复秘密 S，而少于 K 个人得不到 S 的任何信息。将每个人的密钥称为“影子”，只有多于 K 个人(包括 K)的“影子”在一起，经过秘密计算可得到 S。

利用密钥分存方案(也称为门限方案)解决秘密广播的方法是：

A 设计一个门限方案，分给每个接收者一个秘密密钥，这些密钥全是恢复 K 的影子，A 自己保留一些，以增加系统的随机性，用 K 来加密消息 m，假设有 k 个人听广播，协议如下：

(1)A 选择随机数字，用于隐藏消息接收者的数目。

(2)A 产生一个$(k+j+1, 2k+j+1)$门限方案，可恢复的秘密为 K，允许接收者的秘密密钥作为影子，不允许的人的密钥不作为影子。

(3)A 广播 $k+j$ 个影子，这些影子不是允许接收者的影子。

(4)接收者将自己的秘密影子加到收到的影子中，通过秘密计算得到 K，可恢复 m。

就秘密分存而言，除了在第 2 章讲述的多项式插值法以外，还有其它几种方案，这些方案后来被证明是一种一般方案的特例。

勃勒克雷(Blakley)矢量方案：

假设消息是 m 维空间的一个点，每个影子是过此点的$(m-1)$维超平面，任意 m 个超平面在一起，便可恢复出此点；而少于 m 个人是不能得到这个点的。

阿斯木—布龙姆(Asmuth - Bloom)方案：明文 M

对(m, n)门限方案，选择比 M 大的素数 p，选择 n 个比 p 小的数 $d_1, d_2, \cdots, d_n$，满足：$d_i < d_i + 1$，且两两互素，并且

$$d_1 d_2 \cdots d_m > p\, d_{n-m+2} d_{n-m+3} \cdots d_n,$$

随机选择 r，计算 $M' = m + rp$，

影子 $k_i = M' \bmod d_i$，$i = 1, 2 \cdots n$，

利用中国剩余定理，任意 m 个影子可恢复出 M，而 $m-1$ 个却不能。

卡林—格林—赫尔曼(Karnin - Greene - Hellman)矩阵方案：

选择 $n+1$ 个 m 维矢量 v_0，v_1，…，v_n，使得它们任意可能的 $m \times m$ 阶子矩阵的秩为 m，矢量 u 是 m 维的行矢量，$M = u \cdot v_0$

影子是 $u \cdot v_i$，$i = 1, 2 \cdots n$，

任意 m 个影子都可用来解 $m \times m$ 的线性方程组，用以解出 u，进而求得 M；而任意 $m-1$ 个却不能。

以上几种方案都是一些最简单的门限方案，可以将它们修改成为更复杂的协议，以满足不同的要求。附加条件的共享秘密协议有以下几种：

(1)加权分存协议　一个系统中，其中某个人比其他人更重要，这时就可多分给这个人更多的影子，也可以给不同权重的人分配相应的影子数，以达到预期的分存效果。

另一种情形是，有两个敌对的代表团 A、B，A 中有 7 人，B 中有 10 人，只有 A 中的任意 2 人和 B 中的任意 3 人才能恢复秘密。这也是一种带有权重的分存要求，可以将一个三次多项式作为秘密，分给 A 一个线性方程，分给 B 一个二次方程，A 中每人分享这个线性方程上的一个点，B 中的人分享二次方程上的点，这样，A 中的任意 2 人便可恢复线性方程，B 中任意 3 人可恢复一个二次方程，将它们相乘，便可得到秘密的三次方程。

(2)不泄露各自秘密的共享协议　以上各种协议，要求在恢复秘密时，每个人必须提交自己的影子，这样就等于泄露了自己的秘密，许多应用场合是不适合的。例如：对一个文件进行群签名，使用秘密密钥，n 个人分享，只有每个人对文件进行签名后，文件才有效；而各自的签名是不能假冒和伪造的，更不能泄露，必须对各自的持有物——影子进行掩盖，以满足要求。这种情况就像密卡尼(Micali)提出的秘密计算一样。

(3)防止骗子的秘密共享　在大家提交自己的影子恢复秘密时，如果其中有人不诚实而使用假影子，便恢复不出真的秘密。有时需要将共享协议设计成检测和防止骗子的协议，使得骗子被很快检测出来。有两种情形：一是发现骗子协议，当有人欺骗时，协议能够告诉大家有人欺骗；更强的一种是，能够指明何人在欺骗。

当然，还有许多其它类型的带有附加条件的秘密分存方案，可以针对不同的条件，设计不同的协议。读者也可自行设计自己需要的协议，以满足自己的要求。

秘密分存的另一重要应用是 1994 年沙米尔(Shamir)等人提出的图视密码新概念。利用门限方案，可以将二值黑白图像进行任意分存，当 k 个以上图像叠加在一起的时候，便可恢复出秘密图像。

3.2 中级安全协议

3.2.1 数字签名与身份验证协议

在第 2 章中，我们讲述了几个数字签名与身份验证算法，有著名的 RSA 算法、DSS

标准等。这些签名认证协议在许多场合得到了应用。但是，并不能满足其它条件下的签名和认证要求。有以下几种情况需要认真对待。

1. 不可否认签名

传统的数字签名能够任意复制，交给任何人进行验证。例如：对一个声明的签名，通过广播形式，任何人都可验证。但是，如果是两人之间的签名，被其他人验证是很窘迫的情形。应该是 A 给 B 的签名，只有当 B 征得 A 同意后，才能将 A 的签名由第三方 C 验证，这样就可防止 B 无休止地复制 A 的签名。

为了满足以上要求，David—Chaum 设计了如下的签名协议：

设大素数 p 和本原元 g 是公开的，供签名者使用，A 的秘密密钥是 x，公开密钥是 g^x，A 为消息 m 签名。

协议：

(1)A 计算 $Z = m^x \bmod p$，发送 Z 给 B，

(2)B 随机选择 a，$b<P$，计算 $C = Z^a (g^x)^b \bmod P$ 发送 c 给 A。

(3)A 计算 $x^{-1} \bmod (p-1)$，$d = c^{x^{-1}} \bmod P$，发送 d 给 B。

(4)B 验证 $d = m^a g^b \bmod P$ 是否成立。

如果成立，B 接受 A 的签名。

现在 B 将 Z，d，c 交给第三方 C，C 能通过验证第(4)步相信 B，m 是经过 A 的签名吗?不能相信 B。因为 B 完全有可能伪造所有的信息 Z，d，c，B 可以这样做：

1)B 伪造消息 d，使得 $d = m^a g^b$，a，b 是 B 随机选的；

2)B 根据 A 送给他的 Z，计算 c。

所以，B 完全不需要第(3)步，就可造出 c，d，这样第三方 C 就不能相信 B。除非 C 参与了 B 的全过程，知道 B 没有违背协议。

为了挫败 B 违背协议，Chaum 设计了如下协议：

此协议是根据零知识证明方法设计的，关于零知识证明协议，可见 3.3 节。

协议条件同上。p，g 公开，A 的秘密密钥为 x，公开密钥为 g^x。

(1)A 计算 $Z = m^x$，发 m，Z 给 B。

(2)B 随机产生两个小于 p 的数 a，b，计算 $c = m^a g^b$，发送 c 给 A。

(3)A 随机选择 $q<p$，计算 $s_1 = cg^q$，$S_2 = (cg^q)^x$，发送 s_1，s_2 给 B。

(4)B 发 a，b 给 A，使 A 相信 B 在第(2)步中没有欺骗她。

(5)若 B 没骗 A，A 发 q 给 B。

(6)B 检查 $S_1 = c \cdot g^q$，$S_2 = (g^x)^{b+q} \cdot Za$ 是否成立，若成立，则签名有效。

A 也能拒绝对消息 m 的签名。Lein Harn 和 Shoubao Yang 还提出了一种面向组的不可否认签名协议。

2. 可转换的不可否认签名

协议要求：A 给 B 进行消息 m 的一个不可否认签名，当 A 决定其他人也能对签名验证时，A 通过公布另一数据，可以将签名转换成普通的数字签名。

协议准备：选择两个素数 p，q，$q|(p-1)$，$g<q$，选择 $Z \leqslant h \leqslant q-1$，计算

$g = h^{(p-1)/q} \bmod q$，如果 $g = 1$，随机重新选择 h，公开 p，q，g。

A 的秘密密钥为 A 随机选择的数 x，z，A 计算 $y = g^x \pmod p$，$u = g^z \pmod p$，A 的公开密钥是：p、q、g、y、u。

下面是 A 计算对消息 m 的不可否认签名：

(1)A 随机选择 t，$1<t<q-1$，计算

$$T = g^t \bmod p$$

$$m' = Ttzm \bmod q$$

(2)A 用标准的爱格玛尔(ElGamal)签名方案对 m'签名。A 选择随机数 R，$R<p-1$，$(R, p-1)=1$，计算 $r = g^R$，用欧几里得算法计算 S，使得

$$m' = rx + R \cdot S \pmod q$$

(3)A 的签名是(r, s)，T。

A 让 B 验证其签名的协议是：

(1)B 产生两个随机数 a，b，计算 $c = T^{Tma} g^b \bmod p$，发送 c 给 A。

(2)A 产生随机数 k,计算 $h_1 = cg^R \pmod p$和 $h_2 = h_1^z \pmod p$,发送 h_1,h_2 给 B。

(3)B 发送 a，b 给 A。

(4)A 验证 $C = T^{Tma} g^b \bmod p$ 是否成立;若成立，发送 k 给 B。

(5)B 验证 $h_1 = T^{Tma} g^{b+k} \pmod p$和

$$h_2 = y^{ta} r^{ta} u^{b+k} \pmod p$$

是否成立。若成立，A 的签名有效。

如果 A 想让其他人验证其签名，A 告诉其 z，可以像普通的 ElGamal 签名一样，由(r, s)，T验证其签名。

标准的 ElGamal 签名协议是：

A 的公开密钥 p，$y = g^x$，$g<p$ 为本原元，A 的秘密密钥为 x。对消息 m 进行签名。

协议：

(1)A 选择随机数 k,$(k, p-1)=1$,计算 $a = g^k \bmod p$,利用欧几里德算法计算 b

$$m = xa + kb \pmod{p-1}$$

A 对 m 的签名为(a, b)，k 保持秘密。

(2)B 验证 $y^a a^b \pmod p = g^m$ 是否成立。若成立，则签名有效;不成立，无效。

托本—皮德逊(Torben—Pederson)把数字签名概念和秘密分享组合在一起，可以产生分布式的不可否认的可转换数字签名协议。人们先签名消息，然后将验证签名的能力分散，允许几个人中的 k 个人参与协议，可以验证签名的有效性。

3. 盲签名

协议要求：协议使得签名者对文件签字，但让其不知道签字的内容。一般的签字是没有这种功能的，这种盲签名协议可以用在安全投票和选举上。下面是一个基于 RSA 体制的盲签名协议。

设 B 的公开密钥为 e，模数为 n，秘密密钥为 d，A 打算让 B 对消息 m 签名，又要将 m 掩盖起来。

(1)A 选择随机数 $k<n$，隐蔽 m，计算 $t = mk^e \bmod n$，将 t 给 B。

(2)B 对 t 签名，$S = t^d = (mk^e)^d \bmod n$，将 S 给 A。

(3)A 获得了 B 对 m 的签名

$$S' = S/k = m^d \bmod n。$$

(4)任何人都能验证 B 对 m 的签名 S'。因为只有 $S'^e = m \bmod n$，只有 B 有 e。

4. 同时签名。

A 与 B 同时签约一份合同，如果 A 与 B 是面对面的，这很容易；但如果 A 与 B 相距很远，就会产生许多问题。有仲裁者时会容易处理，而对无仲裁者时，虽然已有了一些解决方案，但仍有一些数学问题值得研究。

有一种签名要求是：A 有一个秘密密钥，用来对文件签名，而第三方 C 拥有巨大的计算能力，可以破译出 A 的密钥，这样 C 就可伪造 A 的签名。为了防止这种欺诈，一种故障停止式数字签名可以做到，如果 C 伪造签名，A 可以证明它们都是伪造的；如果 A 签名后抵赖自己的签名，法院可以证实签名不是伪造的。

其基本方法是：对一个公开密钥，有许多秘密密钥和它配对，这些秘密密钥可以产生许多不同的有效签名，A 只有一个私钥，而不知道其它的秘密密钥。C 为了破译 A 的密钥，通过大量计算，终于算出了一个有效密钥，可以做签名，但是他的密钥与 A 的相同的概率非常小，小到可以忽略不计。

法院在收到一份 A 的签名后，将 A 对同一电文的签名比较。如果产生两份不同的签字，可以证明第一份为伪造的；如果对同一份签名，A 无法签出不同的电文，则 A 对该电文负责。

另一种签名形式是组签名。一个团体称为一个组，有若干人组成，组签名的特点是：

(1)对一个电文，只有组内成员可以对它签名。

(2)签名的接收者可以证明是否是来自同一组的签名，但无法确定是组内哪个成员的签名。

(3)当发生争议时，签名能够被“打开”，能获得签名者的身份。

还有基于零知识证明理论的数字签名与身份验证方案，这些方案的设计可参见 3.3 节。

数字签名与认证协议可以根据不同的要求而设计不同的协议。论证这些协议的安全性有很强的理论性，读者可以自行设计。

3.2.2 阈下信道

阈下信道的概念是指 A 与 B 通过公开信道传送 A 已签名的信息，任何人可以对 A 的签名进行验证。但是，B 通过掌握的秘密密钥，恢复出 A 隐藏在签名中的秘密信息。

协议过程如下：

(1)A 随机产生一个无害信息，用于掩盖秘密信息。

(2)B 与 A 共享一个秘密密钥，A 对无害信息签名，并掩盖其秘密信息。

(3)A 将签名交给第三方 C，C 检验签名，有效，不能发现其中的秘密信息。

(4)C 将消息给 B，B 检查签名，确信来自于 A。

(5)B 用秘密密钥，取出秘密信息。

1. 以典型的爱格玛尔(ElGamal)签名为例

协议如下：密钥产生和基本的 ElGamal 签名方案相同。首先选择素数 p，生成元 g 和随机数 r，计算 $k = g^r \bmod p$，公开 k 作为 A 的公开身份，r 为 A 的密钥。B 除了知道 k，还知道 r；通过无害消息 M'来发送阈下消息 M。设 M，M'都小于 p，且$(M, P-1)=1$。

A 计算 $x = g^M \bmod p$，并解下列方程组

$$M' = rx + My，\text{解得 } y,$$

签名是(x, y)，第三者 C 可以通过公开密钥验证 A 签名的有效性，这和正常的ElGamal 签名验证一样，看 $k^x x^y \bmod p = g^{M'} \bmod p$ 是否成立，若成立，签名有效。

同样，B 也首先验证$(g^r)^x x^y = g^M \bmod p$ 是否成立，若成立，他可恢复阈下消息 M

$$M = y^{-1}(M' - rx) \bmod (p-1)。$$

2. ESIGN 签名协议及其阈下信道

该算法的安全性与 RSA 和 DSA 一样，但速度要快得多。

秘密密钥是一对大素数 p，q，公开密钥是 $n = p^2 q$，$H(x)$是消息列散函数(Hash 函数)，签名协议为：

(1)A 随机选择一个小于 pq 的随机数 x。

(2)A 计算：w 和 s

$$w \text{ 是大于}(H(m) - x^k \bmod n)/pq \text{ 的最小正整数},$$

$$s = x + ((w/k \cdot x^{k-1}) \bmod p) \bmod qp$$

A 发送 s 给 B。

(3)B 验证签名。B 计算 $s^k \bmod n$ 和 a。

这里的 a 是大于 n 的比特数除以 3 的数的最小正整数。

如果 $H(m) \leqslant s^k \bmod n$ 并且 $s^k \bmod n < H(m) + 2^a$，那么签名有效，其它情况无效。这里的参数 s 不能取作 2，3 等。因为这时的协议已被攻破，设计者建议 $k=8$，16，32，64，128，256，512，1024 等，p，q 的长度应在 192 比特以上，保证 n 至少 576 比特长。

协议允许 A 事先做预计算，以增加速度：

1)A 计算 $u = x^k \bmod n$，$v = 1/(kx^{k-1}) \bmod p$；

2)签名时，A 计算 w 和 s

3. ESIGN 协议中的阈下信道

在以上协议中，如果 A 以 pq 作为秘密密钥，可以加入阈下信道。公开密钥为 $n = p^2 qr$，A 和 B 共同掌握秘密密钥 r，协议如下：

(1)A 随机选择 x，$x<pqr$，计算 w，s

w 是大于 $(H(m)-x^k \bmod n)/pqr$ 的最小正整数，

$$s=x+((w/kx^{k-1}) \bmod p)pqr$$

其中：$H(m)$ 是消息 m 的列散值；

k 为安全参数。

(2)B 验证签名。B 计算 $s^k \bmod n$ 和 a。

其中：a 是大于 $|n|/3$ 的最小正整数；

$|n|$ 表示 n 的比特数。

如果 $H(m) \leqslant s^k \bmod n$，并且 $s^k \bmod n<H(m)+2^a$ 时，签名有效，否则无效。

当发送阈下信息时，设 M 为阈下信息，M' 为无害信息；A 在计算 s 时，用 M 代替 $H(m)$，这时要求 $M<r$

A 随机选择 u，计算 $x'=M+ur$，

A 用 x' 代替以上协议的 x，作签名 s，

作为任何第三方 C，他可以检查 s 的确是 M' 的有效签名。

B 同样可以验证 s 签名的有效性是 A，并且可以恢复 M

$$s \bmod r=x'+ypqr=M+ur+ypqr \bmod r=M \bmod r$$

ESIGN 阈下信息的隐藏要比前面的 ElGamal 协议的隐藏更加巧妙。在 ElGamal 协议中，A、B 共同拥有彼此的秘密密钥，B 可以假冒 A 签发任何消息；而在 ESIGN 协议中，由于 A 知道 p，q，r，而 B 仅仅知道 r，所以 B 是无论如何也冒充不了 A 的。

可以这样说，所有的数字签名协议都可以隐藏一个阈下信道。例如：DSA 等数字签名标准，这些协议为从事间谍活动的人提供了极好的保护，任何人揭不穿其中的秘密。

除了以上介绍的签名协议以外，在 3.3 节的几种零知识证明的认证签名协议中，同样可以隐藏阈下信道。

3.2.3 比特托管与遗忘传递

比特托管技术与遗忘传递协议是设计 3.3 节零知识证明协议的基础，一些高级安全协议的设计都依赖这两个基础协议。

1. 比特托管技术

所谓的比特托管技术就是满足以下要求的一种安全协议的设计技术。

(1)A 通过一种技术手段将其选择的一比特信息托管起来，好比 A 将一枚硬币投入一个井中，A 不能改变这一比特。

(2)A 将托管的比特交给 B，B 仍不知其内容。

(3)在 B 需要打开时，A 将其打开，好比 A 让 B 猜井中硬币的正反面。若 B 猜测后，A 和 B 同时到井中看结果，判断 B 的猜测是否正确。

用对称密码体制实现比特托管：

(1)B 产生一个随机比特串 R，发送 R 给 A。

(2)A 将其需要托管的比特 b 与 R 一起，用密钥 K 将它们加密，$C = E_k(R, b)$，将 C 送给 B，K 为一随机密钥。

当 B 需要 A 解开 b 时，A 提交 k，可再现 R 和 b。

利用单向函数的比特托管协议：

设 $f(x)$是一单向散列函数。

(1)A 产生随机串 R_1，R_2 及托管比特 b。

(2)A 计算 $f(R_1, R_2, b)$，将计算结果和 R_1 给 B，当 B 需要 A 解开 b 时，A 向 B 提交 R_1，R_2 和 b，A 无法改变 R_2 和 b，这是由 $f(x)$的单向性保证的，通常 $f(x)$为一个哈希(Hash)函数。

2. 遗忘传递协议

遗忘传递协议是另一种形式的基础协议。协议要求 A 发给 B 两条消息中的一条，B 将收到其中一条消息，A 不能控制 B 的选择，也不知道 B 收到了哪一条消息。一个一般性的协议可以按以下方式进行：

(1)A 产生两个公开密钥密码系统，公开其两个公开密钥，A 和 B 共同使用一对称密码系统，如 DES。

(2)B 选择 DES 算法的一个密钥，B 用 A 的公开密钥来加密这个密钥，并将两个结果传给 A。

(3)A 用其两个秘密密钥解开两个结果，一个是真正的密钥，而另一个是一些乱七八糟的随机数，但 A 是无法区分的。

(4)A 用这两个数使用 DES 加密两个消息给 B。

(5)B 只能用真正的密钥解开其中一条消息，而 A 无法判断是哪一个。

(6)协议完成后，A 将其私钥交给 B，以证明 A 没有欺骗 B。

当然，A 可以在第(4)步加密同一消息。

下面我们来看勒宾(Rabin)的协议：

A 将 p，q 两个素数发送给 B。

(1)A 将 $n = p \cdot q$ 给 B。

(2)B 随机选择 $x<n$，$(x, n) = 1$，将 $a = x^2 \bmod n$ 给 A。

(3)A 知道 p，q，A 计算 a 的 4 个平方根，x，$n-x$，y，$n-y$，A 随机选择其中一个发给 B。

(4)B 收到 y 或 $n-y$，B 就可以求 $x+y$ 和 n 的最大公因子，计算出 p 和 q。

(5)如果 B 收到 x 或 $n-x$，B 什么也得不到。

还有一些其它形式的遗忘传递协议：有非交互式的，也有交互式的；这些协议可以用来设计更复杂的安全协议。

3.2.4 随机掷币

A 和 B 想通过随机掷硬币的方式来决定一件事情(如赌博)，为了防止掷币人的欺骗行为和避免引起纠纷，A 和 B 使用以下方法来完成协议。

首先，B 想一个比特，但 B 不立即宣布，只是将它写入信封中；接着，A 公布自己的比特；最后，B 取出比特与 A 的异或，得到一随机比特。只要 A 和 B 中有一人是诚实地遵守协议，最后的随机比特就是真正随机的。利用比特托管技术可以实现以上协议，代替了上面的信封。

(1)A 使用比特托管技术托管一个比特。

(2)B 来猜这个比特。

(3)A 出示这个比特给 B，如果 B 猜对了，B 就赢了。

一般情况是，A 必须在 B 猜测之前抛币，在 B 猜测之后 A 不能改变抛币结果。

1. 使用单向函数的掷币协议

A 与 B 共同选用一单向函数 $f(x)$。

(1)A 随机选择 x，计算 $y=f(x)$，将 y 给 B。

(2)B 猜测 x 的奇偶性，将猜测结果给 A。

(3)若 B 猜测正确，结果为 1，否则为 0，A 公布 x。

(4)B 验证 $y=f(x)$，以确信 A 没有欺骗。

协议要求：$f(x)$必须是安全的 Hash 函数，并且 $f(x)$必须以相等的概率产生奇数和偶数，B 的猜测就不会产生优势。

2. 使用公开密钥的抛币协议。

一般要求公开密码体制满足交换律，即：

$$D_{K1}(E_{K2}(E_{K1}(m))) = E_{K2}(m)$$

比如 RSA 就满足这个条件。

使用随机掷币协议可以用来产生随机密钥，这个密钥来自几个人，而任何人不能控制产生结果，即使有人偷听或欺骗，这样产生的密钥也是真正随机的。

3.2.5 智力扑克

在网络中，玩扑克存在不可信因素，谁也不会轻信一副扑克是由某人随机洗牌的结果；为了避免发牌出现欺骗，必须有必要的安全协议。我们以两人发牌和三人发牌协议为例，说明这种协议的重要性。当然，这些协议同样适用于设计更复杂的安全协议。

1. 两人协议

(1)A 产生 52 张牌 m_1，m_2，…，m_{52}，加密发给 B。

(2)B 随机取 5 个，并用 B 的公开密钥加密，送回给 A。

(3)A 用其私钥对这 5 个密文加密，送给 B。

(4)B 用私钥和 A 的公钥解密，B 获得自己的牌。

A 以同样方式获得自己的 5 张牌。此协议可以一直执行下去，直至打完牌，A、B

公开其牌和密钥，以确保双方没有欺骗。

2. 三人协议

(1)设 A、B、C 三人使用公开密钥体制。

(2)A 产生 52 张牌，对它们加密，发给 B。

(3)B 随机选取 5 个，并加密给 A。

(4)B 将剩下的 47 张牌给 C。

(5)C 随机选 5 个，用其密钥加密给 A。

(6)A 将他们送回的密文解密，再分别送回 B、C。

(7)B、C 用他们各自的密解解密，获得自己的牌。

(8)C 随机选 5 个给 A，A 解密获得自己的 5 张牌。

上面不仅可以使用公开密钥算法，也可使用对称密钥体制。当牌打完后，所有人公开自己的密钥和牌，以确保没人欺骗。

使用某些公开密钥算法可能造成信息泄露。例如：使用 RSA 算法，当一张牌的二进制数是二次剩余时，该牌加密后仍为二次剩余，这样 A 就可以给某些牌做记号，如所有的“A”，这样对玩扑克牌是不公平的，使用零知识证明等高级协议可以防止这类事情的发生。

同样，智力扑克协议可以用来完成某些安全性更高的密钥分配协议。比如，密钥分配中心分配给用户的密钥，要求中心也不知道具体用户所用的密钥。

3.3 高级安全协议

3.3.1 零知识证明协议

假设 A 知道两个大素数 p，q，将它们乘起来后，$n = p \cdot q$，A 告诉 B，说她知道 n 的两个素因子，B 怎样才能相信呢?

如果 A 将 p，q 告诉 B，那么 B 就可以验证 n 是否等于 $p \cdot q$。但是，B 除了相信 A 以外，他还得到了额外的信息，那就是 p，q，他甚至可以向第三方 C 证明他知道 n 的两个因子，甚至将 n 的因子公开。

为了防止 B 的这种行为，必须设计一种协议，除了让 B 相信 A 的结论以外，B 得不到其它任何额外的信息，这就是零知识证明的基本要求。通常用 P 来表示 A，即证明者；用 V 来表示 B，即验证者。对一个问题的证明往往采用交互式方法，一般要求满足以下条件：

(1)P 不能欺骗 V，如果 P 不知道她所宣称的证明，她要使 V 相信她的概率是任意小的。

(2)如果 P 知道证明，在 V、P 都遵守协议时，V 以概率 1 接受 P 的结论。

(3)V 即使采用欺骗手段，违反协议，他从 P 那里得不到任何额外信息。

通常将(1)称为完备性；(2)称为有效性；(3)称为零知识性。

根据 V、P 的计算能力和对知识的定义，可以分为完美零知识证明、统计零知识证明和计算零知识证明。这些理论都需要太多的数学基础和计算复杂性理论，在此就不作深入介绍了，这里仅介绍一些零知识证明协议的设计问题。

假设 P 知道的知识是一个困难问题的解法，通常的零知识证明协议由以下步骤来完成：

(1)P 用她的信息和一些随机数将此问题转化为另一困难问题，新问题和原问题同构，她利用原问题的知识，同样可以解开新的问题。

(2)P 使用比特托管技术，向 V 提交新问题的比特托管。

(3)P 告诉 V 这个新问题，V 不能利用该新问题得到原问题的任何信息。

(4)V 要求 P 打开比特托管：

1)要么公布第(2)步中新问题的解法，并验证它的确是新问题的正确解法；

2)要么给他证明新问题和老问题是同构的。

V 只能从这两个选择中任选一个要求 P 做。选择权在 V，各占 50% 的概率。

(5)P 按 V 的要求回答，如果每步都正确，重复(1)至(5)步 n 遍，那么，P 欺骗 V 的可能性是 $1/2^n$。并不是所有的“困难”问题都有以上形式的零知识证明协议，但许多问题都有，下面几个例子是很有代表性的。

1. 哈密尔顿(Hamilton)回路问题的零知识证明协议

设 G 是一个图，G 中存在一个哈密尔顿(Hamilton)回路，P 知道这条回路，而 V 不知道；P 向 V 证明她知道这条回路，但又使 V 不知道这条回路到底是什么，并且要 V 相信 P 确实知道 G 的一条哈密尔顿回路。

(1)P 对图 G 作随机置换，得一新图 H，设置换为 π，H 和 G 同构，P 就很容易知道 H 的哈密尔顿回路，如果 P 知道 G 的哈密尔顿回路的话。

(2)P 将 H 发给 V。

(3)V 发给 P 0 或 1，0 要求 P 向 V 证明 G 和 H 是同构的，1 要求 P 公布 H 的哈密尔顿回路。

(4)若 P 收到 0，P 将 π 发给 V；

若 P 收到 1，P 发给 V 图 H 的哈密尔顿回路。

(5)V 对收到的信息进行验证。

以上过程可以重复 n 遍，P 欺骗的可能性为 $1/2^n$，

下面，我们来看 P 对 V 欺骗的可能性：

首先，P 根本不知道 G 的一条哈密尔顿回路，她同样可以做 G 的同构 H，但她同样无法找到 H 的一条哈密尔顿回路，当 V 发送 1 时，对第(4)步，P 将无法回答。

其次，P 可能知道另一个图 H 的一条哈密尔顿回路，但要 P 证明 H 和 G 是同构的，并给出同构置换 π 同样是困难问题。

所以，P 无法对 V 进行欺骗；由于协议仅要求 V 发送 0 和 1，V 无法欺骗 P。

2. 图同构问题的零知识证明协议