第 3 章

认证理论基础

学习目标与要求

1. 理解数字认证过程本质是一个交互证明过程这一概念。
2. 掌握交互证明过程需要满足的两个属性：完整性和完备性。
3. 理解在交互证明过程中证明者与验证者之间计算能力的差异。

3.1 引　　言

本章从复杂性理论与交互证明（interactive proof）的角度来阐述数字认证的理论基础。尽管后续将介绍各种消息认证、身份认证及不同种类的数字签名方案或系统，但是无论这些方案之间的区别多大，其本质都是交互证明系统，都需要符合交互证明系统（interactive proof system，简称证明系统）规定的基本特征和安全要求。通过学习本章，读者将了解一个重要的事实：对于一个设计合理的数字认证协议而言，即使在使用一个计算能力非常有限的设备去执行它时，也能充分防范足够聪明或具有较强计算能力的敌手对该设备进行欺骗。此外，读者也能了解到制约数字认证协议的各种因素以及数学基本模型，为学习后续内容打好基础。

3.2 交互证明系统

在通常情况下数字认证过程被认为是一个交互证明过程。在这个交互过程中包括两个实体：证明者（Prover，P）和验证者（Verifier，V），认证过程就是证明者期望通过信息交互使验证者能够接受他的某种宣称的过程。在计算机理论中这个交互证明过程可由交互证明系统所描述。交互证明系统是一类计算模型，像其他计算模型一样，其目标是对一个语言类 L 和一个给定的输入 x 进行验证，判断 x 是否在 L 中。在上述证明过程中，两个实体（验证者和证明者）都被看作是某种类型的计算机（也被称为图灵机，Turing Machine）。它的交互证明过程为：

> 给定输入 x，通过验证者和证明者之间交换信息，最终由验证者根据证明者给出的信息（或证据）判断输入 x 是不是在语言 L 中。

为了从数字认证系统的角度更好地理解这一抽象模型，不妨将语言类 L 比做所有授权标识字符串构成的集合，输入 x 是任何标识字符串，交互证明过程就是判定关系 $x \in L$ 是否成立。如果关系成立，则验证者 V 输出为真（True 或 1），否则为假（False 或 0）。再如，对于一个指纹身份认证系统，语言类 L 可以理解为所有授权用户指纹特征信息构成的集合，输入 x 是任何采集到的指纹特征，认证系统就是判定该指纹特征是否存在于集合 L 中。

为了达到上述证明之目的，也可将交互证明视为二人博弈的过程。在博弈过程中，首先要求证明者满足“忠诚性假定”，即证明者愿意向验证者进行其所宣称的证明，并能够遵循交互证明的相关规定。也就是说，即便证明者有能力在博弈中获胜（去证明某种宣称），但他不尽力导致输掉博弈通常是非常容易的，因此需要在博弈中排除掉这种情况。除此之外，证明者与验证者之间也需要满足下面规定。

① **能力假设**：验证者具有多项式时间确定图灵机的计算机能力，但是证明者具有强大的计算能力；

② **运行规则**：博弈双方共走 m 步（保证在有限时间内终止博弈），且由验证者（或证明者）先行等规定；

③ **胜负规则**：当且仅当 x 属于 L，证明者在以 x 为输入的对局中有必胜策略。

此处可以通过棋类系统更好地理解上述证明系统的规定。例如，围棋的运行规则是黑棋先行，黑先白后，交替下子，但白棋多占 3 又 3/4 棋子来弥补黑棋的先行优势。另一个例子是五子棋中先行的一方优势很大，有研究表明在双方都不犯错误情况下谁先下子谁就会赢。由此可知，运行规则中约定的先行方在博弈中占有优势。胜负规则用于规定博弈中如何确定获胜方，例如围棋按照所占空间多者为胜。

在此基础上，可以采用具有交互带的图灵机模型来描述上述系统，图 3.1 便对上述模型进行了描述：证明者和验证者分别用一台图灵机表示，每个图灵机具有一个工作带；其次，两台图灵机之间共享一条交互带，证明者可以向其中写入或读取信息，验证者也可以读写信息，通常是由一方写入而由另一方读出；进而，此处假设证明者和验证者之

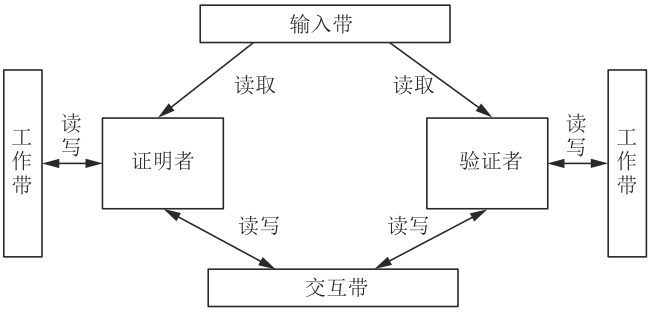

图 3.1　交互证明系统的图灵机表示

间共享一个随机输入带，两者都可以从该带中读取随机字符串用于概率选择。随机输入带的引入是为了在交互证明中引入“随机数发生器”，从而使得证明过程存在不确定性的猜测，这样的模型也被称为概率交互证明系统。

上述图灵机模型的优点就在于其可以用来衡量交互证明系统的性能，评估证明者和验证者的计算性质。在交互过程中，证明者和验证者双方每次送给对方的符号被表示为 $y_1, y_2, \cdots, y_m$，其中，y_i 可依赖于 x 和 $y_1, \cdots, y_{i-1}$ 来确定。此处将其关系表示为 $y_i = R(x, y_1, \cdots, y_{i-1})$，其中，$R(\cdot)$ 为有效可计算函数。如果经过 m 步后，验证者能够证明所有交互符号 $x, y_1, y_2, \cdots, y_m$ 均满足某种预定的条件 $R(x, y_1, y_2, \cdots, y_m)$，则证明者获胜，否则验证者获胜。这个过程可表示为：

$$x \in L \Leftrightarrow (\forall y_1, \exists y_2, \cdots, \forall y_{m-1}, \exists y_m, R(x, y_1, y_2, \cdots, y_m))$$

上述模型是建立在计算模型上的，对于一个安全协议设计而言，更需要关注的是协议的安全性。如果说对于一个有效的输入 $x \in L$，存在一个忠诚的证明者 P 能证明它属于 L，则这将被认为是交互证明系统的应有功能。而用于认证的交互证明系统更关心无效输入下的证明者行为，显然，对于无效输入 $x \notin L$，一般希望证明者即便具有很强乃至无限的计算能力也不能对验证者进行欺骗，即向验证者 V 证明错误的结果 $x \in L$。

3.3 模型与计算能力假设

数字认证研究的对象是各种安全协议，评价安全协议的标准就是看协议是否能够抵抗敌手的各种攻击。敌手攻击的种类是繁多的，但对认证协议而言，最重要的是保证敌手无法对其进行欺骗性的宣称。

根据交互证明系统的定义，协议中的敌手显然就是证明者，他要向验证者证明某种虚假的宣称。以一个简单门禁系统为例，它由进门读卡器、门禁控制器、控制门开关的电路以及射频 IC 卡构成。其中，门禁控制器由一台用于 IC 卡认证功能的小型计算机或单片机构成，而射频 IC 卡内部则集成了一个小型集成电路。可以看出这样一个门禁系统成本不高，作为验证者的门禁控制器和作为证明者的射频 IC 卡的计算能力都非常有限。此处便需要思考这样一个问题，当攻击者采用一个后端为大型机（或未来量子计算机）的设备进行攻击，这样的门禁系统是否就是不堪一击的？显然，需要设计一个好的认证协议，使拥有合法射频 IC 卡的授权用户可以快速实现身份认证，而使具有较强计算能力的攻击者通过认证变为困难的。

因此，依据证明者与验证者的计算能力对比，可以发现一个根本性的问题：

> 敌手（证明者）的能力越强，他是否就越可能进行欺骗？或者说，是否在认证协议中要求验证者的能力一定具有与证明者对等或更强的计算能力？

这是个比较有意思的问题，其内涵可以看成是：证明者的能力是否决定命题的证明结果或命题的正确与否？命题的证明是否与验证者的能力有关？证明命题与证明者和验证者之间的计算能力相关吗？是不是一个非常“弱智力”的验证者能被“强智力”的证明者所欺骗？如果一个“聪明”的证明者能够将“假命题”证明为真的，命题的证明是“绝对”还是“相对”的？等等。

令 $L \subset \{0,1\}^*$ 是一个语言类，可以将它理解为某种宣称的集合。根据上述胜负规则和对证明者不可欺骗性的要求，交互证明系统定义如下：

定义 3.1 由证明者与验证者 (P, V) 构成的一个交互证明系统，如果它满足下面两个条件，则有以下结果。

① 完整性（completeness）。如果 $x \in L$，则存在诚实的证明者 P，使得 V 与 P 的交互之后，以绝对优势（或接近概率 1）接受 x 并输出“$x \in L$”如下：

$$\Pr[(\mathrm{P},\mathrm{V})(x) = 1 | x \in L] = 1 \tag{3.1}$$

② 完备性（Soundness）。如果 $x \notin L$，则对任意的证明者 P^*，V 与 P^* 交互之后，仅以较小（可忽略）概率 ε 输出“$x \in L$”如下：

$$\Pr[(\mathrm{P}^*,\mathrm{V})(x) = 1 | x \notin L] < \varepsilon \tag{3.2}$$

上述定义中，完整性用于表明对于真命题 $x \in L$，证明系统存在一个有效的交互过程以接近 1 的概率通过证明；反之，对于一个伪命题 $x \notin L$，那么任何的攻击者 P^* 想通过交互证明的概率都是足够小（可忽略的）。因此，完整性被用于定义证明系统的正确性，而完备性则被用于定义证明系统的不可欺骗性。注意，由于在完整性与完备性定义中的前提假设不同（分别为 $x \in L$ 和 $x \notin L$），因此这两个概率相互之间无关。

下面将给一个简单例子来验证交互证明系统的存在。假设存在一个山洞，如图 3.2所示，进入洞穴后有两条道路，但是不知道这两条路径是否相通。

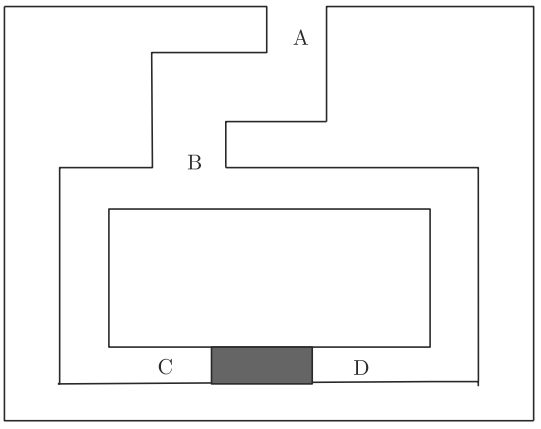

图 3.2 交互证明洞穴

假设证明者 P 是个探险家，他知道是否两条路径连通，并想向验证者 V 证明这一事实，但是验证者 V 担心证明者 P 欺骗他，因此设计了下面的游戏流程：

① P 走进洞穴，可以选择一条路径到达 C 或者 D 点，V 站在 A 点，但看不到 P 选择了哪个路径；

② 在 P 进入洞穴后，V 走进 B 并向 P 喊话，让 P 从左通道或者右通道出来；

③ P 答应了，并从指定通道走出；

④ 如果 P 从指定的通道走出，V 认可道路连通（输出 1）；否则，否认这一结论（输出 0）。

下面将分析上面游戏的有效性。上述游戏中的命题 x 为：洞穴两条路径是连通的，L 表示事实。此处分两种情况进行讨论。

① 完整性：如果洞穴两条路径是连通的，即 $x \in L$，那么对于一个忠诚的证明者 P 来说，他总能按照 V 的要求走出，因此，这种情况的交互游戏获胜（输出 1）的概率为 1，即 $\Pr[(\mathrm{P},\mathrm{V})(x)=1|x \in L]=1$。

② 完备性：如果洞穴两条路径是不连通的，那么对于任何证明者而言，他都可以事先猜测验证者的喊话，从而在第一步中选择验证者要求的方向进入通道，如果这种猜测是正确的，那么他就能获得成功；否则由于岩石的阻挡，他只能验证失败。因此，这一验证结果取决于对验证者的猜测，如果验证者的喊话是完全随机的，即发出指令“由左通道（或右通道）出来”的概率各为 1/2，即 $\Pr[(\mathrm{P},\mathrm{V})(x)=1|x \notin L]=1/2$。

注意，在交互证明系统中，哪一方先行选择是很重要的。上例中是证明者先行，即证明者先选择一条路径，但验证者不知道，因而可获得至少 1/2 的成功概率。但是，如果验证者可以先为证明进行选择，那么他可以直接指定证明者进入通道的方向，验证过程将会更为简单，且不含有随机性。

3.4 交互证明系统举例

下面将以平方剩余（也称二次剩余）的判定问题为例讨论一个实际的交互证明协议。首先，回忆二次剩余问题，二次非剩余类定义如下：

定义 3.2（二次非剩余类） 令 x 和 n 为正整数，如果存在整数 y，使得 $y^2 \equiv x \pmod{n}$，则称 x 是在模 n 下的二次剩余，记 $x \in QR_n$，而 $QNR_n=\{x|x \notin QR_n\}$ 则被称为二次非剩余类。

例如，令 $n=13$，当 $x=4$ 时，可以计算 $2^2 \equiv 4 \pmod{13}$ 和 $11^2=121 \equiv 4 \pmod{13}$。因此，4 属于二次剩余类（$4 \in QR_{13}$）。但是，当 $x=5$ 时，找不到一个数的平方模 13 后等于 5，因此 5 属于二次非剩余类（$5 \in QNR_{13}$）。如表 3.1所示，此处列出了所有模 13 的二次剩余关系。因此，可知 $QR_{13}=\{0,1,3,4,9,10,12\}$ 和

$QNR_{13}=\{2,5,6,7,8,11\}$。

表 3.1　二次剩余表（模 13）

x	y	x	y
1	1,12	4	2,11
9	3,10	3	4,9
12	5,8	10	6,7

需要说明的是，当 n 为素数时，二次剩余的判定是容易的，但是当 n 为合数时，二次剩余的判定依赖于 n 分解后的素因子。如果 n 的分解未知时，二次剩余的判定是困难的。

下面将给出一个二次剩余判定的证明系统。这个交互证明协议被描述在表 3.2中。在这个协议中，给定 n 和一个自然数 x，希望证明证明者 P 具有对任意给定数 x 进行二次剩余判定的能力。

表 3.2　基于二次非剩余判定的交互证明协议

步　骤	证　明　者	验　证　者
第一步		验证者随机选择整数 $b\in\{0,1\}$，以及一个自然数 $z\in\mathbb{Z}_n^*$，然后计算数值 w 如下：$$w\equiv\begin{cases}z^2 \pmod n & b=1\\ x\cdot z^2 \pmod n & b=0\end{cases}$$ 并将其送给证明者
第二步	证明者判定 w 是否属于 QR_n，如果是，则令 $b'=1$；否则，令 $b'=0$。并将所得结果 $b'\in\{0,1\}$ 传送给验证者	
第三步		验证者检测是否 $b=b'$，如果是，输出 1，否则输出 0

在表 3.2 中，协议由验证者 V 开始，他选择一个随机比特 b 以及一个自然数 z，然后依据 b 生成挑战问题 w，并将其发送给证明者 P；证明者 P 只需要通过对 w 的二次剩余判定即可对 b 给出一个猜测 b'；最后，验证者 V 检查是否 $b=b'$，如果等式成立，则输出 1；否则，输出 0。因此，最终输出结果给出了证明者 P 是否具有对给定数 x 进行二次非剩余判定能力的判定。

定理 3.1　上述二次非剩余判定的交互协议是一个交互证明系统。

证明　根据交互证明系统的定义，可证明该协议满足下面性质：

① **完整性**：如果 $x\in QNR_n$，那么对任何数 $z\in\mathbb{Z}_n^*$，挑战 $w=x\cdot z^2 \pmod n$ 也

属于平方非剩余类。根据这一性质，当验证者选择 $b=1$ 时，挑战 w 属于平方剩余类；但 $b=0$ 时，挑战 w 属于平方非剩余类。如果证明者能够判定 w 是否属于平方剩余类，则可根据 w 正确猜测出 b，因此，证明者能以概率 1 使得 $b=b'$，也就意味着验证者将以概率 1 接受证明者的证明，成功概率计算如下：

$$\begin{aligned}
&\Pr[(\mathrm{P},\mathrm{V})(x)=1|x\in QNR_n]\\
=&\Pr[b=b'|x\in QNR_n]\\
=&\Pr[b'=1|b=1,x\in QNR_n]\cdot\Pr[b=1|x\in QNR_n]+\\
&\Pr[b'=0|b=0,x\in QNR_n]\cdot\Pr[b=0|x\in QNR_n]\\
=&\Pr[w\in QR_n|w\equiv z^2 \pmod n, x\in QNR_n]\cdot\Pr[b=1]+\\
&\Pr[w\in QNR_n|w\equiv xz^2 \pmod n, x\in QNR_n]\cdot\Pr[b=0]\\
=&1\cdot\Pr[b=1]+1\cdot\Pr[b=0]=1
\end{aligned}$$

② **完备性**：如果 x 是二次剩余类，那么无论验证者选择何种 b，所发送的 w 都是二次剩余的，因此，证明者将无法进行判定 b，因此他只能以概率 1/2 猜测 b。因此，V 将以概率 1/2 接受证明者的证明。上述性质可由下面等式证明：

$$\begin{aligned}
&\Pr[(\mathrm{P}^*,\mathrm{V})(x)=1|x\in QR_n]\\
=&\Pr[b=b'|x\in QR_n]\\
=&\Pr[b'=1|b=1,x\in QR_n]\cdot\Pr[b=1|x\in QR_n]+\\
&\Pr[b'=0|b=0,x\in QR_n]\cdot\Pr[b=0|x\in QR_n]\\
=&\Pr[w\in QR_n|w\equiv z^2 \pmod n, x\in QR_n]\cdot\Pr[b=1]+\\
&\Pr[w\in QNR_n|w\equiv xz^2 \pmod n, \exists y, y^2=x \pmod n]\cdot\Pr[b=0]\\
=&\Pr[w\in QR_n|w\equiv z^2 \pmod n]\cdot\Pr[b=1]+\\
&\Pr[w\in QNR_n|w\equiv (yz)^2 \pmod n]\cdot\Pr[b=0]\\
=&1\cdot\Pr[b=1]+0\cdot\Pr[b=0]=\Pr[b=1]=1/2
\end{aligned}$$

为了将验证者的欺骗成功概率降低到任意小的 ε，可采用 k 次协议执行的方式，直到完备性概率满足 $1/2^k<\varepsilon$。令 $(\mathrm{P}^*,\mathrm{V})^{(i)}$ 表示第 i 次协议执行，则上面完备性概率要求每次证明都是成功的，也就是满足如下等式：

$$\begin{aligned}
&\Pr[(\mathrm{P}^*,\mathrm{V})(x)=1|x\in QR_n]\\
=&\prod_{i=1}^{k}\Pr[(\mathrm{P}^*,\mathrm{V})^{(i)}(x)=1|x\in QR_n]=\frac{1}{2^k}
\end{aligned}$$

因此，协议满足完整性和完备性，问题得证。 ■

例 3.1 在上例中，首先来看完整性例子：令 $n=13$ 且 $x=2$，

① 如果 V 选择 $b=1$ 且 $z=5$，那么 $w=12$ 是二次剩余类。

② 如果 V 选择 $b=0$ 且 $z=5$，那么 $w=11$ 是二次非剩余类。

再来看完备性的例子：令 $n=13$ 且 $x=3$，

① 如果 V 选择 $b=1$ 且 $z=5$，那么 $w=12$ 是二次剩余类；

② 如果 V 选择 $b=0$ 且 $z=5$，那么 $w=10$ 也是二次剩余类。

注意，上述证明过程对 n 没有任何前提假设，也无须采用 RSA 密码中大整数 $N=p\cdot q$ 的条件。原因在于，x 的二次剩余性质决定了协议结果的概率，这个概率是与证明者的猜测无关的，或者说 1/2 的猜测成功概率是仅由验证者对 b 的选择概率决定的；因此，只要 x 属于二次剩余类，证明者无论具有怎样（无穷）的计算能力都无法进行欺骗（改变成功概率 $1/2^k$）。

3.5 协议信息泄露

在设计证明协议时，证明者 P 所具有的计算能力有可能会被验证者 V 利用，实现对某些问题的计算或泄露出某些秘密信息。为了说明这一问题存在，下面仍然以二次剩余问题的求解为例，证明协议中的信息泄露问题。

令 $N=p\cdot q$，不妨假设 p 和 q 是两个大素数，保证 N 的分解是计算上困难的，且 $p\equiv q\equiv 3 \pmod 4$，然后将 N 作为公共参数发送给协议双方。表 3.3给出了一个交互协议，协议中证明者宣称：他能够有效地判定任意给定 $\mathbb{Z}_N$ 中的数是否在平方剩余类中。该协议只需要两次交互过程：验证者 V 传送给 P 值 y，它是一个随机选择的模 N 的二次剩余；然后，P 计算 y 的二次方根 x，并传送给 V；最后，V 验证 $y\equiv x^2 \pmod N$ 是否成立。其中，$p\equiv q\equiv 3 \pmod 4$ 是保证证明者能够有效计算 y 的二次方根。[①]

表 3.3 具有信息泄露的二次剩余判定交互证明协议

步　骤	证　明　者	验　证　者
第一步		随机选择一个正整数 $z\in\mathbb{Z}_N^*$，然后计算数值 $y\equiv z^2 \pmod N$，并将 y 送给证明者
第二步	判定 y 是否属于 QR_N，如果是，则计算 x，使得 $y\equiv x^2 \mod N$，传送 x 给验证者	
第三步		检测是否 $y\equiv x^2 \mod N$？如果是，输出 1，否则输出 0

① 令 $p\equiv q\equiv 3 \pmod 4$ 是素数，那么 $N=p\cdot q$ 被称为 Blum 整数，且 $(q+1)/4$ 和 $(p+1)/4$ 均为整数。在已知 p 和 q 时，可以通过中国剩余定理计算出平方剩余数的四个平方根。

不难证明上述协议能够对证明者是否具有平方剩余判定和求根能力进行有效判定。但是协议也存在着证明者对 N 分解的泄露：

定理 3.2（信息泄露）　如果上述协议输出 1，那么验证者能够通过该协议以 1/2 概率获得 N 的分解。

证明　在一次成功的协议执行过程中，第一步 V 先选择 z，使得 $y \equiv z^2 \mod N$；第二步中，P 返回 x，如果 $x \neq z$ 且 $y \equiv x^2 (\mathrm{mod}\, N)$。显然，有相等关系 $y \equiv x^2 \equiv z^2 (\mathrm{mod}\, N)$ 存在。因而，$N|(z^2-x^2)$，也就是 $pq|(z-x)(z+x)$。由于 $x, z \in \mathbb{Z}_N^*$，故有 $0 < z+x < 2N$ 且 $-N < z-x < N$。

根据上述分析，可知有下面四种情况：

① 当 $p|(z-x)$ 且 $q|(z+x)$ 时，可计算 $p = \gcd(N, z-x)$ 和 $q = \gcd(N, z+x)$；

② 当 $q|(z-x)$ 且 $p|(z+x)$ 时，可计算 $q = \gcd(N, z-x)$ 和 $p = \gcd(N, z+x)$；

③ 当 $N|(z+x)$ 时，$z \equiv -x \pmod N = N - x$；

④ 当 $z - x = 0$ 时，$z = x$。

上述情况分别对应于平方剩余 y 在模 N 下的四个根，但只有前两种情况下可实现大整数 N 的分解。

下面计算成功分解 N 的概率。在验证者 V 将 y 给了证明者 P 之后，P 并不能判定四个根中哪个是验证者的随机选择 z，因此，他返回 x 是 z 或 $N-z$ 的概率是 1/2，但这不能成功分解 N；而其他两种情况下都可以成功分解 N，概率是 1/2，也就是，N 的分解将以 1/2 概率被泄露。问题得证。 ■

3.6　零知识证明系统

由于交互证明系统中证明者具有较强的计算能力或拥有特殊的秘密，通过上述协议分析可知，设计不当的协议可能会使得敌手利用上述能力或透露其中的秘密。为了避免上述安全风险发生，需要协议满足零知识性（zero-knowledge）。也就是在证明过程中验证者能够验证证明者的宣称，但是不能获得证明者所掌握的知识，那么这便被称为证明者具有零知识性，该系统也被称为零知识证明系统（zero-knowledge proof system）。上述零知识性定义如下：

定义 3.3（证明的零知识性）　在交互证明系统中一方 A^* 对语言 L 被称为具有（完美或计算）零知识性，只要对于每个可能策略 B（作为协议另一方），存在一个多项式时间的模拟器 S，使得下面两个概率分布是（统计或计算上）不可区分的。

① $\{(A^*, B)(x)\}_{x \in L}$ 表示对公共输入 $x \in L$，A 与 B 交互过程中 B 所有看到的信息；

② $\{S^B(x)\}_{x \in L}$ 表示模拟器 S 对公共输入 $x \in L$ 模拟协议运行的输出。

其中，S^B 表示模拟器 S 针对策略 B 构造的多项式算法。

从原理上讲，零知识性来源于模型检验（Model Checking）思想。首先，协议的真实运行（上述定义中的第一种情况）中敌手得到的信息与不需要任何交互下的模拟运行（上述定义中的第二种情况）是“一样”的；其次，基于上述两种情况的一致性，如果模拟运行是不要特别知识的，那么真实协议运行也就没有泄露任何知识。

这里，“知识”是指具有解决某种问题的能力。在交互证明系统中，验证者通常不需要具有特定的知识，而证明者则是通过所掌握的知识（也被称为秘密）使验证者相信他的某种宣称，因此，零知识就是指证明者的秘密是否丢失。在上述定义中，第一个分布也被表示为 B 的视，即，$\mathrm{View}_B(x)=\{(A^*,B)(x)\}_{x\in L}$。如果上述两个概率分布是统计不可取分的，那么可以记作

$$\{(A^*,B)(x)\}_{x\in L}\simeq\{S^B(x)\}_{x\in L}$$

此时，可称交互证明系统具有完美零知识性。

下面仍然以平方剩余的判定为例说明零知识证明系统的构造。令 $N=p\cdot q$ 是 RSA 类型的大整数，令一个随机整数 $x\in\mathbb{Z}_N$，则系统公钥为 $pk=(x,N)$，证明者的私钥是 $sk_A=w$，且 $x\equiv w^2 \pmod N$。

表 3.4 显示了一个具有零知识性的二次剩余判定的交互证明协议，协议中证明者的宣称是：x 属于模 N 下的平方剩余类，即 $x\in QR_N$。

表 3.4　具有零知识性的二次剩余判定的交互证明协议

步　骤	证　明　者	验　证　者
第一步	随机选择一个正整数 $u\in\mathbb{Z}_N^*$，然后计算数值 $y\equiv u^2 \pmod N$，并将 y 送给验证者	
第二步		随机选择一比特 $b\in_R\{0,1\}$，并把它送给证明者
第三步	如果 $b=0$，发送 $v=u$ 到验证者； 如果 $b=1$，发送 $v\equiv w\cdot u \pmod N$ 给验证者	
第四步		如果 $b=0$，那么 $V_{pk}(y,v)=\begin{cases}1, & y\equiv v^2 \mod N\\ 0, & \text{其他}\end{cases}$ 如果 $b=1$，那么 $V_{pk}(y,v)=\begin{cases}1, & xy\equiv v^2 \mod N\\ 0, & \text{其他}\end{cases}$ 最后输出 $V_{pk}(y,v)$ 的值

下面仅就交互证明系统中的证明者零知识性进行证明，而该系统的完整性和完备性留给读者自行证明。

定理 3.3 上述二次剩余判定交互证明协议满足证明的零知识性。

证明 首先，协议的真实执行中验证者的视，以及它在协议运行成功（输出为 1）时的分布定义如下：

$$\mathrm{View}_B(x) = \{(A^*, B)(x)\}_{x\in L} = (y, b, v) \in_R \{QR_N, \{0,1\}, \mathbb{Z}_N\}$$

其次，构造一个多项式时间的协议模拟器 S^B 如下：

第一步：随机选择一比特 $b \in_R \{0,1\}$；

第二步：随机选择整数 $v \in_R \mathbb{Z}_N$；

第三步：按照如下方式计算 y 如下。

$$y \equiv \begin{cases} v^2 \pmod N & b = 0 \\ v^2/x \pmod N & b = 1 \end{cases} \tag{3.3}$$

显然由此得到的 $S^B(x) = (y, b, v)$ 是有效的协议模拟。如果 $x \in QR_N$，那么 $y = v^2$ 或者 $y = v^2/x \pmod N$ 都是有效的平方剩余，即 $y \in_R QR_N$（但如果 $x \notin QR_N$，那么 $y \equiv v^2/x \pmod N$ 也是非平方剩余）。因此，$S^B(x) \in_R \{QR_N, \{0,1\}, \mathbb{Z}_N\}$，也就是说，$\{(A^*, B)(x)\}_{x\in L} \simeq \{S^B(x)\}_{x\in L}$，因而，上述协议满足证明的零知识性要求，问题得证。■

3.7 由 NP 类问题理解交互证明系统

为了进一步讨论交互证明系统中的证明者与验证者之间计算能力的差异，本节以计算复杂性理论中的 NP 类和 P 类问题之间的转化关系为例对这种差异性进行分析，读者可选修本节。

通常上讲，P 类问题是指多项式时间内确定图灵机（deterministic turing machine）可计算问题构成的集合，而 NP 类问题则是指多项式时间内由非确定图灵机（non-deterministic turing machine）可计算问题构成的集合。就复杂性理论而言，对于任何 NP 类问题，总存在一个 P 类算法实现对问题解进行验证，下面定理为 NP 类问题的协议构造奠定了基础。

定理 3.4 如果语言 L 属于 NP 类（$L \in NP$），当且仅当存在算法 M 属于 P 类（$M \in P$）[①]和多项式 $p(\cdot)$，使得

$$x \in L \Longleftrightarrow \exists y \in \{0,1\}^{p(|x|)}, M(x, y) = 1 \tag{3.4}$$

① 确定图灵机 M 满足如下三个性质：

- 对于任意输入 x 和 y，M 在多形式时间内停机并输出 0 或 1;
- 对于所有 $x \in L$，存在一个长度为 $p(|x|)$ 的串 y，使得 $M(x, y) = 1$;
- 对于所有 $x \notin L$ 和任意长度为 $p(|x|)$ 的串 y，必然有 $M(x, y) = 0$。

其中，$p(|x|)$ 表示以串 x 长度（表示为 $|x|$）为输入的多项式值。

基于上述刻画方式，通常来说非确定型计算问题是“猜测加验证”的方式。这一定理也表明，任何 NP 类问题总存在一个确定性的算法进行验证。给定每个 $x \in L$，定理中能够使 x 通过验证的长为 $p(|x|)$ 的字符串被称为“证据”或“佐证”（Witness）。

根据上面定理，可构建一个简单的 NP 类问题的交互证明协议。

第一步：验证者 V 首先发送 x 给证明者 P；

第二步：证明者 P 返回证据 y 到验证者 V；

第三步：验证者 V 执行确定性多项式时间判定算法 M 对 (x, y) 进行判定。

上述协议中验证者只需要采用算法 M 即可实施协议，但证明者需要具备对给定的 x 找出证据 y 的能力，通常这种获得证据的能力是依靠巨大计算能力实现的或是依赖某种解决问题的“诀窍”（可理解为拥有某种秘密），这从另一个侧面说明，验证者与证明者能力的不均衡性。

对于一个交互证明系统而言，从算法复杂性角度我们希望满足如下要求：

> 一个有效的交互证明系统只需要验证者和“合法”证明者具有多项式时间的计算能力；而非法证明者或恶意攻击者即便具有巨大的运行时间和空间也不能改变证明结果。

最后，可以看出一个命题的验证过程是与证明者的计算能力无关的，即使证明者具有非常强的欺骗能力，只要无法改变待验证的事实状态（如山洞联通问题中把两个通道打通），便无法对一个能力很弱的验证者进行欺骗。通过本章的讨论，还能看出命题的证明是没有任何前提假设存在的，在真理面前人人平等，定理的证明是客观的、稳定的、不以人的认识为转移的。

习　　题

1. 简述交互证明系统的三个性质。

2. 当 $n = 21$ 和 $x = 15$ 时，对基于二次剩余问题的证明协议进行安全性分析。

3. 当 $n = 31$，如何计算 $y = 5$ 的平方根？对于素数 n，请给出计算平方根的数学公式。

4. 试用中国剩余定理分析 $N = pq$，且 $p \equiv q \equiv 3 \pmod 4$ 时，如何计算平方根？例如，$p = 19$ 和 $q = 31$ 时，计算 366 的平方根。

5. 在山洞的零知识证明协议中，如果由验证者先开始协议，结果会怎样？

6. 在表 3.3 中，试用零知识证明方法讨论具有信息泄露的二次剩余判定交互证明协议的非零知识性。

7. 请证明表 3.4 中交互证明协议的完整性与完备性。

第 4 章

数据完整性认证

学习目标与要求

1. 理解数据完整性验证的基础知识。
2. 掌握 Hash 函数构造方法和基于 Hash 的完整性验证方法。
3. 掌握消息认证码的构造。

4.1 绪　　论

消息认证就是验证消息完整性的过程，当接收方收到发送方的报文时，接收方需要验证收到的报文是真实的和未被篡改的。它包含两层含义：一是验证信息在传送过程中未被篡改或伪造；二是验证信息的发送者是真实的而不是冒充的，即数据来源认证。本章将集中于介绍上述消息认证中的第一个问题，即数据完整性验证问题。

数据完整性验证是对数据是否被改变进行的验证，任何的数据变化都应被这一验证发现。此处给出如下准确定义：

定义 4.1（数据完整性）　数据完整性验证是一种验证消息的变更状态，防止恶意攻击者对交换数据进行修改、插入、替换和删除的技术，或者说如果其被修改、插入、替换和删除，则可以被该技术检测出来。

在数据交换中，由于各种原因（包括物理上损耗、敌手攻击等），数据可能被异常改变，这里的数据可能是文件、网络数据包、数据库中的表、交互中的消息等。在网络通信环境下，数据完整性验证也被称为消息完整性验证。

消息完整性验证问题由来已久，所采用的方法也多种多样。在古代，人们已经对信件采用了带有徽章的“蜡封”的方式，通过检查信件蜡封的破损情况来保证消息不被篡改、阅读和伪造。其他方法还包括采用特定的书写方式、书写工具、书写载体等。由此可见，完整性验证的技术并不是单一的、固定的。

随着数字社会的到来，传统的通过媒介方式进行消息完整性验证的方法已不可用，因此保护计算机或网络中数字信息的完整性需要采用一些其他可行的技术。目前经常使用的技术包括纠错码、Hash 函数、数字水印等。本章将对上述方法进行简单分类，而后重点对基于 Hash 函数的完整性验证方法进行介绍，内容包括 Hash 函数的定义、多

种构建方法、长数据处理技术以及带密钥的 Hash 函数构造等。特别是，本章内容引入了基于 NP 困难问题的 Hash 构造方法，克服了传统教材中 Hash 函数构造方法只基于分组密码的单一性缺点。

4.2 数据完整性验证方法

数据完整性验证的通用方法是消息发送者在消息中加入一个与该消息相关联的验证码并经加密后发送给接受者（如果对消息无保密性要求，则只需加密验证码即可）。接受者利用约定的算法对解密后的消息再次求取验证码，并将得到的验证码与收到的验证码进行比较，若二者相等便接收，否则拒绝接收。

上述过程已经成为目前完整性验证的基本流程。然而，这里需要注意的是消息与验证码之间的关系，显然在此需要两者之间建立非常紧密的关系来保证完整性验证的安全。根据待验证数据与验证块之间的依附关系，可将完整性验证分为两类。

① **带验证块的验证方式：**在不改动待验证数据的情况下，通过增加额外的验证块来存储校验信息，实现数据完整性验证；

② **不带验证块的验证方式：**通过修改待验证数据来加入额外的校验信息，实现数据完整性验证。这种方式通常被称为数字水印或数字指纹技术。

第二种方法的前提是验证信息的加入并不影响待验证数据的使用，这意味着待验证数据中存在较大的数据冗余，这些数据冗余使得额外验证信息的加入成为可能。例如，视频和音频文件中存在信息冗余，同时人类听觉和视觉的感知能力也允许细小的更改不影响视听质量，在这种情况下，采用数字水印或数字指纹技术（如脆弱水印）实现完整性验证是必然的选择。然而，数字消息中最常见的文本文件则很难通过不被察觉的方式添加验证信息，尤其一些受法律或法规约束的文本，其任何改动都是不被允许的，因此这类数据仍然需要采用带验证块的验证方式。

如图 4.1 所示，带验证块的验证方式也包括两类技术：基于 Hash 函数的验证方法和基于纠错码的验证方法。这两者的使用场景有所不同，通常需要根据出现数据可能发生错误的性质来加以区分。如果数据错误（被称为干扰）具有确定的概率分布（如信道传输错误），那么可根据概率分布来采用编码理论进行纠错编码以设计验证码，达到发现错误和纠正错误的目的；如果数据错误不具有确定的概率分布，甚至会有人为的伪造或篡改，那么这种情况下通常就需要采用基于 Hash 函数的验证方法。

根据待验证数据在使用中潜在改变方式的不同，可将其分为两类：非攻击下的改变和恶意攻击下的改变。非攻击下所发生的数据改变通常是指数据在传输、存储、处理中所经受的异常变化，例如，信道干扰、分块存储中的数据块丢失等。这种改变通常是随机出现的，不带有明确的攻击企图。另一类是指人为对数据和验证块的攻击，其带有明

显的恶意企图，并通常伴随着专业的攻击手段，显然，对这种攻击的防范比前一种更加困难，需要采用更加严格的密码方式。根据上述两种分类方式，可将基于 Hash 函数的验证方法分为两类。

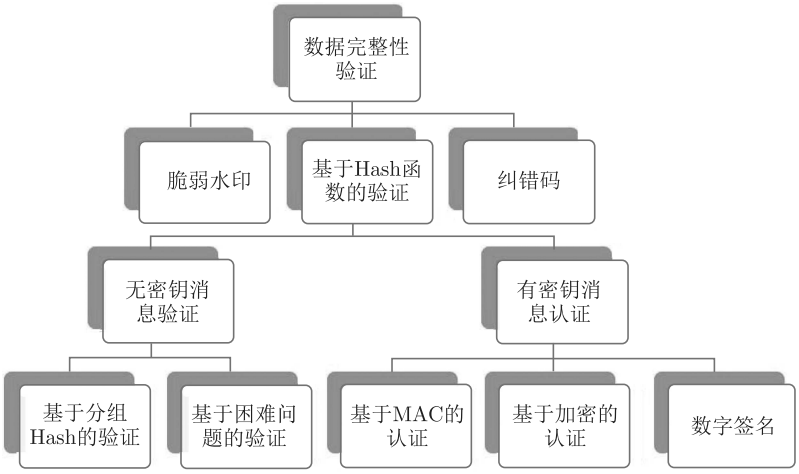

图 4.1　数据完整性验证方法分类

① **无密钥的 Hash 验证方式**：不改动待验证数据的情况下，通过增加额外的验证块来存储校验信息，实现数据完整性验证，但这种校验信息会被明文传输，因此该方式不能抵抗人为攻击；

② **有密钥的 Hash 验证方式**：不改动待验证数据的情况下，通过增加额外的验证块来存储校验信息，实现数据完整性验证，其校验信息以基于密钥处理的密文形式被传送，因此能够抵抗人为攻击。

基于纠错码的验证也属于无密钥的验证，尽管在错误检测中非常有用，但其并不可能可靠地验证数据完整性。这是因为纠错码中采用的校验多项式是线性结构，可以非常容易地被人为改变，之后被传输的数据仍能在校验中被通过，因此其安全性不足。所以除了一些特殊设计的纠错码，通常的纠错码并不适合在安全应用中被使用，本书将不对此进行介绍。

在论述密码学 Hash 函数之前，此处将进一步研究数据完整性验证中数据块与验证块之间的关系。如前所述，对于文本一类数据，通常采用尾随一个验证块的方式进行完整性验证或身份认证。虽然同样采用这种方式的纠错码并不安全，但是基于 Hash 函数的验证方法却能抵抗恶意的数据伪造攻击，即其可保证敌手难以在验证块不变的情况下伪造一个数据块。本节将说明其中的原理，并阐述一个构造密码学 Hash 函数的重要目标：如何采用较短的校验信息来发现尽可能多的数据错误。

在研究完整性验证中数据块与验证块的关系之前，此处先给出一个重要的“碰撞”

概念：在数据块后添加了一个校验块之后，如果两个不同数据块能够得到相同的验证码，便可以称其产生了一个碰撞。也就是说，对于一个验证块生成函数 f 和任意的不同数据块 m 和 m' 而言，如果 $f(m)=f(m')$，则可称它们产生了碰撞。

定理 4.1 令数据块长度为 l，验证码长度为 k 且 $k \leqslant l$，那么两个数据产生随机验证码碰撞的碰撞概率与待验证数据长度 l 无关，只与验证码长度 k 相关。

证明 假设从数据块到验证码的映射 $f:\{0,1\}^l \to \{0,1\}^k$ 是均匀分布的随机单射，也就是说，给定数据 m，函数 $f(m)$ 可能是 2^k 个验证码中任意一个数。显然，当 $l \geqslant k$ 时，数据的数目大于验证码的数目，就会出现几个数据映射到一个验证码的情况，也就是产生了碰撞。假设一个验证码对应的平均数据数目被称为验证码重复率 T。

根据上述定义，可知下面关系成立：

当 $l=k$ 时，验证码重复率为 $T=2^k/2^k=1$；

当 $l=k+1$ 时，验证码重复率为 $T=2^{k+1}/2^k=2$；

……

当 $l=k+m$ 时，验证码重复率为 $T=2^{k+m}/2^k=2^m$。

由此可知，验证重复率为 $T=2^l/2^k=2^{l-k}$，这相当于 2^l 个球中每 T 个球具有相同颜色，总共有 2^k 种颜色。

当两个数据产生相同验证码时发生的碰撞，其碰撞概率可以被理解为给定一个球 m，从 2^l 个球中取出另一只一样颜色球 m' 的概率，具体碰撞概率为

$$\Pr[f(m)=f(m')]=T/2^l=2^{l-k}/2^l=1/2^k$$

由此可知，碰撞概率只与验证码长度 k 相关，而与待验证数据长度无关，因此，问题得证。 ■

在上述证明中，验证码重复率 T 表示了具有相同验证码的消息数目，但其与碰撞概率无关。注意，上面结论必须建立在数据块到验证码的映射是均匀分布随机单射的假设下。而且，当消息长度比验证码长度每大 1 比特，则验证重复率就增加一倍。当 $l-k \gg 0$，那么 T 也必然大于 0。

上述定理阐述了下列事实：

① 只要验证码长度足够长，那么数据产生碰撞的概率便可以足够的小；

② 可设计固定长度的验证码，实现对于任意长度的待验证数据的验证。

这是两个重要的结论，它为构建后续的安全 Hash 函数提供了前提，也打下了良好的数学基础。此外，Hash 函数一个重要指标是编码效率 R，它是指 Hash 函数的输出与输入数据长度之比，例如本节 Hash 函数的编码效率为 $R=k/l$，通常 $\dfrac{1}{5} \leqslant R \leqslant \dfrac{1}{3}$。

4.3　密码学 Hash 函数

Hash 函数也被称为杂凑函数或散列函数，从严格意义上讲，其输入为一个可变长度串 x，返回一个固定长度串，该串被称为输入 x 的 Hash 值（消息摘要）。但在实际应用中，输入 x 的长度被固定到某一特定长度 l，且要求其大于输出长度（保证碰撞存在）。因为 Hash 函数是多对一的函数，所以一定要将某些不同的输入转化为相同的输出。这就要求给定一个 Hash 值，其求逆应该是比较难的，但由给定的输入计算 Hash 值必须是很容易的，因此也可称 Hash 函数为单向 Hash 函数。再考虑到前述所讨论的碰撞性，此处需要对密码学 Hash 函数提出如下两点要求：

首先，密码学 Hash 函数需要抗原像（preimage free）攻击，也就是给定任何 y，找到一个 x，使得 $f(x)=y$ 是困难的。这一性质遵循单向函数“反向困难”的定义。因而，密码学 Hash 函数的前提是单向函数存在。

其次，密码学 Hash 函数要求是抗碰撞的（也被称为抗二次原像攻击，Second Preimage Free），即给定 x，找到另一个不同的 x'，使得 $h(x)=h(x')$ 也是困难的。

综合上述两点，定义密码学 Hash 函数如下：

定义 4.2（密码学 Hash 函数）　令 $l>k$，一个函数 $h:\{0,1\}^l\to\{0,1\}^k$ 被称为密码学 Hash 函数。如果它满足下面属性：

① 有效计算：给定 x，可在多项式时间内计算 $y\leftarrow h(x)$。

② 单向性：已知 y，找出 x，使得 $h(x)=y$ 困难。

③ 弱碰撞：已知 x，找出 x' 且 $x'\neq x$，使得 $h(x')=h(x)$ 困难。

上述定义中，也可省略对单向性的要求，因为如下定理存在：

定理 4.2　弱碰撞性预示着单向性。

证明　假设存在一个有限时间内解决 Hash 函数 h 中单向性问题的预言机（Oracle），即，$\text{Oracle}(h,y)=x$，使得 $h(x)=y$ 成立。那么给定一个 x，并提交 $y=h(x)$ 给该 Oracle，如果 Oracle 返回 x'，如果 $x=x'$，那么重复上面过程；否则，输出 x' 作为弱碰撞问题的结果。在上述过程中，重复运行次数与 Hash 函数在原像集合 $\{x:h(x)=y\}$ 大小相关。如果存在 Hash 碰撞，则 $\Pr[x'\neq x]\geqslant\dfrac{1}{2}$。[①] 弱碰撞问题得到解决的概率 $\Pr[\text{Oracle}(h,y)=x'\vee x\neq x']=\Pr[\text{Oracle}(h,y)=x']\cdot\Pr[x\neq x']\geqslant\Pr[\text{Oracle}(h,y)=x']/2$，因此，如果预言机解决 Hash 函数的单向性问题的概率 $\Pr[\text{Oracle}(h,y)=x']$ 足够大，则上述碰撞问题求解概率也足够大，此时弱碰撞性不再成立。因而，逆反命题成立，问题得证。　■

由于上述定理中的蕴含关系，在密码学 Hash 函数定义中可以去掉单向性的要求，

① 当存在 Hash 碰撞时，$m=|\{x:h(x)=y\}|\geqslant 2$，进而 $\Pr[x'\neq x]\leqslant\dfrac{1}{2}=\dfrac{m-1}{m}=1-\dfrac{1}{m}\geqslant\dfrac{1}{2}$。

而仅保留弱碰撞性。上述 Hash 函数的定义满足了密码学的较低安全性，但有些情况下需要密码学 Hash 函数具有如下更强的安全性。

定义 4.3（抗碰撞 Hash 函数） 令 $l > k$，如果一个函数 $h : \{0,1\}^l \to \{0,1\}^k$ 满足下面属性，便可被称为抗碰撞 Hash 函数。

① h 是一个密码学 Hash 函数；

② 强碰撞：找出任意 $x' \neq x$，使得 $h(x) = h(x')$ 困难。

显然，强碰撞性要比弱碰撞性更加严格，此处给出如下简单证明：

定理 4.3 强碰撞性预示着弱碰撞性。

证明 假设存在一个解决 h 中弱碰撞问题的预言机 Oracle，即，$\text{Oracle}(h, x) = x'$，使得 $h(x') = h(x)$ 成立。那么对一个给定的抗碰撞 Hash 函数而言，只需要随机选择一个 x，并提交给该 Oracle，如果 Oracle 返回 x' 且 $x \neq x'$，那么输出 (x, x')。显然，(x, x') 使得强碰撞性不再成立。因而，逆反命题成立，问题得证。 ■

上述定义并没有涉及密钥，也就是说 Hash 函数的安全性不是基于某种秘密的私密性，而是 Hash 函数的数学属性本身决定的。从广义上讲，给定密钥空间 $\mathcal{K}$，密码学 Hash 函数 $H : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^k$ 是一个能将任意长度消息压缩到固定长度比特串的函数。但在实际系统中，通常会限制每一次 Hash 函数的输入长度，再通过递归多次调用实现无限长输入的输出。因此，密码学 Hash 函数被进一步定义为 $H : \mathcal{K} \times \{0,1\}^l \to \{0,1\}^k$，这里仅定义了 Hash 函数的压缩性质，其他性质与前述定义一致。

4.4 基于分组密码的 Hash 构造

采用分组密码构造是目前设计 Hash 函数最经常被使用的方法，典型的标准包括 MD4、MD5、SHA-1、SHA-2 等，其中，SHA 是标准 Hash 算法（Standard Hash Algorithm）的缩写，SHA-2 是第二代标准，它由 6 个函数构成，包括 SHA-224、SHA-256、SHA-384、SHA-512、SHA-512/224、SHA-512/256 等，后面的数字表示输出的长度，可根据不同应用需要选择使用。

分组密码是指一种将数据分解为若干分组，并以分组为单位循环进行处理的密码设计技术。通常情况下，一轮数据处理并不足以达到数据混淆的目的，因此需要采用多轮处理的方式获得最后的结果。但是与分组加密不同，Hash 函数的设计要更加简单，原因在于加密需要考虑信息加密后的恢复，因此每次运算都需要设计成为密钥下的随机置换（可理解为符号之间的换位）；而 Hash 函数只要保证输出结果的随机性，不需要考虑消息的恢复。

分组 Hash 函数的构造结构可被描述如下：假设输入数据 $\boldsymbol{W}$ 可被分解为 l 个分组，即 $\boldsymbol{W} = (w_1, w_2, \cdots, w_l)$，且 $w_i \in \mathcal{W}$（对于 $i \in [1, l]$），每次由函数 $f : \mathcal{K} \times \mathcal{W} \times \mathcal{M}^k \to \mathcal{M}$

处理一个分组，其中，$\mathcal{K}$、$\mathcal{W}$、$\mathcal{M} \in \{0,1\}^s$ 分别为 s 比特长的密钥、数据（也被视为“填料”）和状态分组。Hash 函数操作是一个迭代过程，第 i 次处理为

$$M_{i+k} \leftarrow f_{K_i,w_i}(M_i,\cdots,M_{i+k-1}), i=1,\cdots,n \tag{4.1}$$

其中，K_i 为第 i 次迭代所使用的密钥，$n>k$ 且 f 是一个非线性函数。通常情况下，k 是每次处理中的分组数目，n 是 k 的倍数，也被称为 Hash 函数的轮数。

如果把这一过程看成一个“搅拌机”，那么 Hash 函数的输入数据作为“填料”将不断地被加入到迭代过程中。由于输入数据是有限的，在这些数据输入完后，通过对其简单组合将形成新的“填料”。因此，分组 Hash 构造与通常理解的函数处理不同，后者是对输入数据进行变换，而分组 Hash 函数则是由输入数据控制对系统状态的变换，这是一种设计上的显著不同。此外，上述系统状态的变换要求是非线性的（也就是无法由线性方程对该变换进行表示，进而对上述迭代过程也不能进行表示），其通常依靠 f 中存在的一个“非线性盒”来实现。

在给定初始的 k 个状态 $M_1,\cdots,M_k$ 后，n 次处理可表示为：

$$\begin{cases} M_{k+1} & \leftarrow & f_{K_1,w_1}(M_1,M_2,\cdots,M_k) \\ M_{k+2} & \leftarrow & f_{K_2,w_2}(M_2,M_3,\cdots,M_{k+1}) \\ \vdots & & \vdots \\ M_{n+k-1} & \leftarrow & f_{K_{n-1},w_{n-1}}(M_{n-1},M_n,\cdots,M_{n+k-2}) \\ M_{n+k} & \leftarrow & f_{K_n,w_n}(M_n,M_{n+1},\cdots,M_{n+k-1}) \end{cases} \tag{4.2}$$

最终，Hash 函数的输出为最后 k 个状态，即 $(M_{n+1},M_{n+2},\cdots,M_{n+k}) \leftarrow \mathrm{Hash}(\boldsymbol{W})$。令每个分组长度为 $|M_i|=s$ 比特，则函数 f 每次只处理 $k\cdot s$ 比特。

如果每个分组占用一个寄存器，那么硬件设计仅需要 k 个寄存器，并用新产生的分组替代序号最小的分组，其他分组顺序右移，上述设计的好处是易于硬件实现。

下面针对上述 Hash 函数构造模型，简要分析 Hash 函数破解的难度。如图 4.2 显示了基于分组技术的 Hash 函数构造框图。不难看出，上述 Hash 函数可以被看成由初始状态 $(M_1,M_2,\cdots,M_k)$ 到最终状态 $(M_{n+1},M_{n+2},\cdots,M_{n+k})$ 的过程，期间经过了由消息 $(w_1,w_2,\cdots,w_l)$ 控制的 n 次状态转换。这一过程可转化为如下问题：

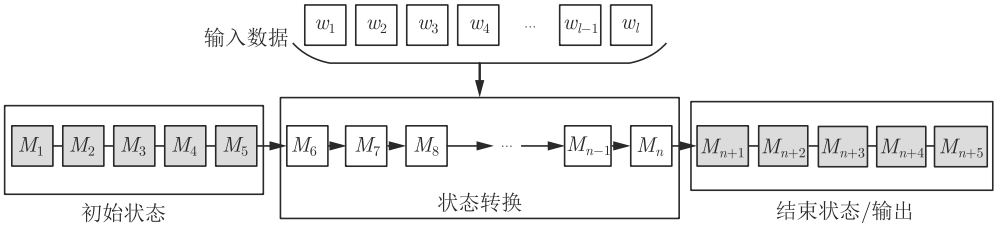

图 4.2　基于分组技术的 Hash 函数构造框图

定义 4.4（分组 Hash 函数求逆问题） 给定一个 Hash 函数输出 $(M_{n+1}, M_{n+2}, \cdots, M_{n+k})$ 和初始状态 $(M_1, M_2, \cdots, M_k)$，则分组 Hash 函数求逆（或碰撞）过程也就是在已知最终状态 $(M_{n+1}, M_{n+2}, \cdots, M_{n+k})$ 的情况下找到一组消息 $(w_1, w_2, \cdots, w_l)$，使其控制一个函数将输出数据由终止状态还原到初始状态 $(M_1, M_2, \cdots, M_k)$ 的过程。

不难发现随着轮数 n 的增大，分组 Hash 函数求逆问题难度也逐渐增大。在没有任何先验知识的情况下，只要轮数足够多，由结束状态反推结果与初始状态相同的概率等于 $1/2^{ks}$。

4.4.1 SHA-1 算法构造

SHA-1 是一个输入为 512 比特、输出为 160 比特的压缩函数，其定义如下：

$$\mathrm{SHA}-1 : \{0,1\}^{128} \times \{0,1\}^{32*16} \to \{0,1\}^{160} \tag{4.3}$$

第一个输入参数是 128 比特的密钥 K，该密钥被定义为常数，第二个输入参数是 32*16=512 比特的输入消息。

SHA-1 面向字进行处理，字长度为 32 比特，在 SHA-1 中状态分组为 M，初始状态由 5 个字组成。初始的五个状态分别为

$$\begin{aligned} &M_1 = \mathrm{C3D2E1F0}, \quad M_2 = 10325476, \quad M_3 = \mathrm{98BADCFE}, \\ &M_4 = \mathrm{EFCDAB89}, \quad M_5 = 67452301 \end{aligned} \tag{4.4}$$

如图 4.3 所示，给定初始的五个状态 $M_1, \cdots, M_5$ 后，SHA-1 处理要进行 $n = 80$ 轮迭代。对于第 $i = 1, \cdots, 80$ 次迭代的过程采用了反馈移位寄存器方式，表示为

$$\begin{aligned} M_{i+5} &= \mathrm{ROTL}^5(M_{i+4}) + f_i(M_{i+3}, M_{i+2}, M_{i+1}) + M_i + w_i + K_i \\ M_{i+3} &= \mathrm{ROTL}^{30}(M_{i+3}) \end{aligned} \tag{4.5}$$

其中，ROTL^j 表示循环左移 j 位，$+$ 表示模 2^{32} 整数加运算。

关于输入数据（填料），512 比特的输入 W 需要被进一步划分成 16 个 32 比特的字，即 $W_i = (w_1, w_2, \cdots, w_{16})$。这 16 组输入只能完成前 16 次迭代，第 16 次迭代之后用到的所有填料 w_i 由之前的填料组合产生，表示如下：

$$w_i = \mathrm{ROTL}^1(w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16}), i = 17, \cdots, 80 \tag{4.6}$$

每次迭代都使用一个非线性函数 f_t，每个函数 f_t（$1 \leqslant t \leqslant 80$）操作三个 32 位字 (X, Y, Z), 并产生一个 32 位字作为输出。函数定义如下：

$$f_t(X,Y,Z) = \begin{cases} (X \wedge Y) \vee ((\neg X) \wedge Z), & 1 \leqslant t \leqslant 20 \\ X \oplus Y \oplus Z, & 21 \leqslant t \leqslant 40 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & 41 \leqslant t \leqslant 60 \\ X \oplus Y \oplus Z, & 61 \leqslant t \leqslant 80 \end{cases} \tag{4.7}$$