第 2 章
数 学 基 础

现代密码学是建立在数学、计算机科学等基础学科之上的。《老子·德经》有云：合抱之木，生于毫末；九层之台，起于累土；千里之行，始于足下。本章介绍现代密码学中的一些常用数学基础知识。

2.1 预 备 知 识

2.1.1 素数

整数集合用 $\mathbb{Z}$ 表示，对于整数集合中的元素 $a,b\in\mathbb{Z}$，如果存在一个整数 c，使得 $a*c=b$ 成立，则称 a 整除 b，记作 $a|b$。如果 $a|b$ 并且 a 是正整数，则称 a 是 b 的一个除数。如果加上条件 $a\notin\{1,b\}$，则称 a 是 b 的一个非平凡除数或称为因子。正整数 $p>1$ 如果没有因子，则为素数；也就是说，素数只有 1 和自身两个除数。一个大于 1 的正整数如果不是素数则为合数。我们约定 1 既不是素数也不是合数。

很多密码学协议中都需要生成素数。生成一个小的素数比较容易，然而基于小素数的密码学协议往往是不安全的。如何生成一个大素数？我们可以选择一个较大的数，然后检查这个数是否为素数，如果不是，则重新选择并检查，直到产生一个大素数为止。下面介绍著名的 RM 素数检测算法。

【算法 2-1】 RM 素数检测算法

步骤 1：对于待检测的随机数 p，计算 b。b 是 2 整除 $p-1$ 的次数，即 2^b 是能整除 $p-1$ 的 2 的最大幂数。然后计算 m，使得 $n=1+2^b m$。

步骤 2：选择一个小于 p 的随机数 a。

步骤 3：设 $j=0$ 且 $z=a^m \bmod p$。

步骤 4：如果 $z=1$ 或 $z=p-1$，那么 p 通过测试，p 可能是素数。

步骤 5：如果 $j>0$ 且 $z=1$，那么 p 不是素数。

步骤 6：设 $j=j+1$。如果 $j<6$ 且 $z\neq p-1$，则计算 $z=z^2 \bmod p$，然后回到步骤 5。如果 $z=p-1$，那么 p 通过测试，p 可能是素数。

步骤 7：如果 $j=b$ 且 $z\neq p-1$，那么 p 不是素数。

2.1.2 模运算

【定义 2-1】（同余的模）

设 a,b 是整数，如果 $a=b+kn$ 对某些 k 成立，那么就说 a,b 模 n 同余，记作 $a\equiv b(\mathrm{mod}\,n)$，$n$ 称为同余的模。

模 n 运算的结果一定是 0 到 $n-1$ 之间的一个数，从 0 到 $n-1$ 的整数组成的集合包括了模 n 的所有结果，或者说模 n 运算的结果一定落在这个集合中，称这个集合为模 n 运算的最小剩余集，记作 $Z_n=\{0,1,2,\cdots,n-1\}$。

2.1.3 群

【定义 2-2】（群的定义）

一个非空集合 G 对于一个二元运算“*”来说作为一个群，假如

Ⅰ. G 对于“*”来说是闭的；

Ⅱ. 结合律成立，即对于 G 的任意三个元 a、b、c 满足

$$a*(b*c)=(a*b)*c$$

Ⅲ. G 中至少存在一个左单位元 e，使得

$$e*a=a$$

Ⅳ. 对于 G 的每一个元 a，在 G 中至少存在一个左逆元 a^{-1}，使得

$$a^{-1}*a=e$$

例如，假设 G 是全体整数的集合，G 对于普通加法来说作成一个群。假设 G 是所有不等于零的整数的集合，G 对于普通乘法来说不作成一个群。

一个群 G 中元素的个数可以是有限的，也可以是无限的。如果一个群中元素的个数是一个有限整数，则称这个群为有限群。否则的话，这个群称为无限群。一个有限群的元的个数称为这个群的阶。

由于在一个群里结合律是成立的，因此 $a_1*a_2*\cdots*a_n$ 有意义。又由于群对于“*”来说是闭的，因此 $a_1*a_2*\cdots*a_n$ 是 G 的某一个元。这样，可以把 n 个相同的元 a 来相乘。因为用普通乘法的符号来表示群的运算，所以可以使用符号 a^n 来表示 n 个相同的元 a 的乘法，即

$$a^n=a*a*\cdots*a\quad（n\text{ 是正整数}）$$

并且也把 a 称为 a 的 n 次乘方（简称 n 次方）。

【定义 2-3】（交换群）

一个群 G 称为交换群，假如

$$a*b=b*a$$

对于 G 的任何两个元 a、b 都成立。

【定义 2-4】（单位元）

一个群 G 中唯一能使

$$e*a=a*e=a\qquad（a\text{ 是 }G\text{ 的任意元}）$$

的元 e 称为群 G 的单位元。

【定义 2-5】（逆元）

唯一能使

$$a^{-1} * a = a * a^{-1} = e$$

的元 a^{-1} 称为元 a 的逆元，有时也简称逆。

【定义 2-6】（阶）

对于群 G 的一个元 a，能够使得

$$a^m = e$$

的最小的正整数 m 称为 a 的阶。

【定义 2-7】（循环群）

如果一个群 G 的每一个元都是 G 的某一个固定元 a 的乘方，就把 G 称为循环群；也就是说，G 是由元 a 所生成的，并且用下面符号来表示

$$G = (a)$$

其中，a 称为 G 的一个生成元。

2.2 密码学困难性假设

现代密码学方案尤其是公钥密码学方案的安全性是建立在解决某些问题的困难性假设基础上的。例如，RSA 公开密钥算法是基于大数分解困难性假设设计的，ElGamal 加密算法是基于离散对数困难性假设设计的，Paillier 加密算法的安全性依赖于合数剩余判定困难性假设。那么，在密码学中什么样的问题是困难的呢？困难并不是无法计算或无法攻破，很多学者致力于研究当前密码学中常用困难性假设问题的解决算法并已经取得了一系列进展。例如，对于满足如下假设的大数分解困难性问题，已经陆续有更短运行时间的算法被提出。假设 $N = pq$，p 和 q 是两个长度相等但大小不同的素数，大数分解问题要求对 N 进行分解，即求出 N 的素因子。Pollard 提出的 RHO 方法是一种通用因子分解方法，针对上述大数分解问题，该算法的时间复杂度是 n 的指数函数，其中 n 是大数 N 的长度。Pomerance 提出的二次筛算法也是一种通用因子分解方法，该算法的时间复杂度是 n 的亚指数函数。尽管解决这些困难性假设问题已经取得了一些成果，但是当前没有找到多项式时间算法或概率多项式时间算法来解决这些问题。因此，当合理选择参数时，人们认为攻破基于这些困难性问题的密码学方案在时间上是不可接受的，从而保证了这些密码学方案在一段时间之内的安全性。当然，随着信息技术的不断进步，如果有一天这些困难性假设不再成立，那么这些假设所对应的密码学方案的安全性也将荡然无存。

本节介绍现代密码学方案中常用的困难性假设问题，深入理解这些问题可以帮助人们更好地设计密码学方案。

2.2.1 大数分解困难性假设

在数论中，对一个数进行因子分解是一个古老的问题。分解一个小的数相对比较容易，例如下面使用试除法得到一个数的因子分解，其中 p_i 是互不相等的素数并且 $x_i \geq 1$。

$$12 = 2^2 \times 3$$

$$88 = 2^3 \times 11$$

$$\vdots$$

$$N = p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$$

然而，分解一个较大的数就不是这么容易了。尽管当前已经有二次筛算法、连分式算法、普通数域筛选法等一系列研究成果，但是正如上文中提到的，这些算法无法在多项式时间或概率多项式时间内解决问题。因此，在当前的计算能力下，解决大数分解问题是困难的。这就是大数分解困难性假设，其形式化定义如下。

给定一个大数 N，N 满足 $N = pq$，其中 p 和 q 是两个长度相等但大小不等的素数，即 $|p| = |q| = l$。对于任意的多项式时间算法 $A(N) = (p', q')$，存在一个可忽略的函数 $\mathrm{neg}(n)$ 满足

$$P_r\left[(A(N) = (p', q')) \wedge ((p', q') = (p, q))\right] \leqslant \mathrm{neg}(n)$$

2.2.2 离散对数困难性假设

首先解释什么是离散对数。给定素数 p，假设 α 是 Z_p^* 上的生成元，β 是 Z_p^* 上的元素，如果整数 x 满足 $\alpha^x \bmod p = \beta$，则称 x 是关于 α 底 β 的对数，记作 $x = \log_\alpha \beta$。下面讨论定义在任意循环群上的离散对数。假设 $\mathbb{G}$ 是 n 阶循环群，g 是 $\mathbb{G}$ 上的生成元，则对于任意的 $h \in \mathbb{G}$，存在唯一 $x \in Z_n$ 使得 $g^x = h$。当循环群 $\mathbb{G}$ 已知时，称 x 是关于 g 底 h 的对数，记作 $x = \log_g h$。可以看出，对于任意整数 x'，如果 $g^{x'} = h$，则 $x = x' \bmod n$。从这个角度来讲，离散对数的值在“有限”范围内，而传统对数值的范围是无穷集合。尽管存在这种差别，但是传统对数的很多规则仍然适用于离散对数。例如，假设 e 是循环群 $\mathbb{G}$ 的单位元，则 $\log_g e = 0$。

严格的离散对数困难性假设如下：假设 p 是一个大素数且 $|p| = l$，$\alpha \in Z_p^*$ 是一个生成元，$\beta \in Z_p^*$ 且满足 $\alpha^x \bmod p = \beta$。对于任意的多项式时间算法 $A(\alpha, \beta, p)$，存在一个可忽略的函数 $\mathrm{neg}(n)$ 满足

$$P_r\left[(A(\alpha, \beta, p) = x) \wedge (\alpha^x \bmod p = \beta)\right] \leqslant \mathrm{neg}(n)$$

循环群上的离散对数困难性假设如下：给定 n 阶循环群 $\mathbb{G}$，g 是 $\mathbb{G}$ 上的生成元，h 是 $\mathbb{G}$ 上的元素。对于任意的多项式时间算法 $A(\mathbb{G}, g, h)$，存在一个可忽略的函数 $\mathrm{neg}(n)$ 满足

$$P_r\left[(A(\mathbb{G}, g, h) = x) \wedge (x \in Z_n) \wedge (g^x = h)\right] \leqslant \mathrm{neg}(n)$$

2.2.3 Diffie-Hellman 问题

Diffie-Hellman 问题与上节介绍的离散对数困难性假设具有一定的相关性。常用的 Diffie-Hellman 问题分为两类，一类是计算 Diffie-Hellman 问题，简称 CDH；另一类是判定 Diffie-Hellman 问题，简称 DDH。

给定 n 阶循环群 $\mathbb{G}$，g 是 $\mathbb{G}$ 上的生成元，h_1 和 h_2 都是 $\mathbb{G}$ 上的元素。计算 Diffie-Hellman 问题是指计算 $g^{\log_g h_1 \cdot \log_g h_2}$。判定 Diffie-Hellman 问题是指判定 $\mathbb{G}$ 上的元素 h' 是否满足 $h' = g^{\log_g h_1 \cdot \log_g h_2}$。