第 11 章　公钥密码学中的数学问题

密码学（密码技术）是一门应用的科学，是信息安全的核心技术．它的知识涉及多个领域，如数学、计算机等学科．密码学也可以认为是主要以数学学科为理论基础，以计算机科学为实现工具而形成的一门具有自己特色的理论体系的一门科学．本节的主要目的是综合介绍密码学所用到的数论、代数的一些数学问题．这些问题将构成公钥密码学所依赖的数学理论的核心内容．通过对这些问题的进一步深入理解，将很容易地了解与学习公钥密码学的分析技术与设计技术．另外，本章所介绍的数学问题的重要性主要体现在这些问题计算的有效性或者是计算的复杂性，因此，在本章中，我们还将对一些问题的计算复杂性进行讨论．

§1　时间估计与算法复杂性

我们必须首先介绍复杂性理论的一些概念，这有助于理解密码技术和算法的安全性，而且有助于我们讨论各种数学问题的算法时间估计．

复杂性既是研究算法所需的计算量，也针对问题的内在复杂性进行分类．

算法的复杂性常常用两个概念来度量：时间复杂性和空间复杂性，这两个概念通常表示为输入长度 n 的函数，而这两个概念从某种程度上也是可以相互转化的．比如要用试除的方法求一个大数的因子分解，可以在一台计算机上串行的搜索素数表；也可以把素数表分配给多台计算机，进行并行计算．前一种方法牺牲时间，后一种则牺牲了空间．

定义 1　在计算机的二进制整数的运算中，两个比特的一次加法、减法、乘法或一个二位整数被一位整数整除的运算时间相同，都称为一次**比特运算**.

另外，一个算法在计算机上运行时，通常还有移位运算或存取运算，由于这些运算较快，在算法的时间估计中，通常忽略这些运算时间．

在一个算法的运行中，比特运算次数与完成计算的时间基本上成正比，所以通常我们对于一个算法的时间估计总是指计算机运行该算法所需要的比特次数．

定理 1　若 a,b 是两个二进制长度为 k 的整数，则计算这两个整数的和或差的计算需要 $O(k)$ 次比特运算．

定理 1 显然成立．

定理 2　若 a,b 分别为 k 位与 l 位二进制数 $(l \le k)$，则计算它们的积或商需要 $O(kl)$（$O(\log_2 a \cdot \log_2 b)$）次比特运算．

证明　先来看看乘法的实现．令

$$a=(a_k\cdots a_1a_0)_2=\sum_{i=0}^{k}a_i2^i,\quad b=(b_l\cdots b_1b_0)_2=\sum_{j=0}^{l}b_j2^j.$$

由

$$ab=(\sum_{i=0}^{k}a_i2^i)(\sum_{i=0}^{l}b_j2^j)=\sum_{j=0}^{l}2^j(\sum_{i=0}^{k}a_ib_j2^i)$$

可见乘法可分以下两步完成：

（i）对于每一个$b_j\ (0\le j\le l)$，计算$b_j\sum_{i=0}^{k}a_i2^i$．由于b_j取 0 或 1，所以$b_j\sum_{i=0}^{k}a_i2^i$是 0 或$\sum_{i=0}^{k}a_i2^i$，至多需要k次比特运算.

（ii）将求得的$b_j\sum_{i=0}^{k}a_i2^i$乘以2^j，并对$j=1,2,\cdots,l$求和，得到乘积ab．其中乘以2^j是进行一次移位运算，然后将$l+1$个数求和，这些数至多是$k+l$位的二进制数．因此完成乘法所需要的比特运算次数是$O(l(l+k))$，假定$l\le k$，于是$O(l(l+k))=O(kl)$．

关于除法的证明类似，作为练习证明留给读者．定理 2 得证.

推论 1 计算$n!$需要$O(n^2\log_2{}^2 n)$次比特运算.

下面我们考虑的一个问题是初等数论中介绍 Euclid 算法的时间估计.

定理 3 若a,b分别为n位数与m位数，$n>m$，利用 Euclid 算法计算(a,b)的时间估计为$O((\log_2 a)^2)$.

证明 给定a,b，由 Euclid 算法我们有

$$\begin{aligned}
&a=q_0b+r_0, && 0\le r_0<b,\\
&b=q_1r_0+r_1, && 0\le r_1<r_0,\\
&r_0=q_2r_1+r_2, && 0\le r_2<r_1,\\
&\cdots\ \cdots\ \cdots\ \cdots && \cdots\ \cdots\ \cdots\ \cdots\\
&r_{k-4}=q_{k-2}r_{k-3}+r_{k-2}, && 0\le r_{k-2}<r_{k-3},\\
&r_{k-3}=q_{k-1}r_{k-2}+r_{k-1}, && 0\le r_{k-1}<r_{k-2},
\end{aligned}$$

$$r_{k-2} = q_k r_{k-1}.$$

估计这个算法要用到下述事实：

1) 上述算法需要的除法次数 k 最多是 2n.；

2) 带余除法运算 $a = bq + r$ 的时间复杂度为 $O((\log_2 a)(\log_2 q))$；

3) 上述算法中的商 $q_0, q_1, \ldots, q_k$，满足

$$\sum_{i=0}^{k} \log_2 q_i = \log_2 \prod_{i=0}^{k} q_i \le \log_2 a.$$

故而完成 Euclid 算法需要的运算次数为

$$O((\log_2 a)(\sum_{i=0}^{k} \log_2 q_i)) = O((\log_2 a)^2).$$

因此利用 Euclid 算法计算 (a,b) 的时间估计为 $O((\log_2 a)^2)$.

由于 Euclid 算法可以用于计算模 m 中元素的逆元及模 m 的一元一次方程等．所以由定理 3 有以下结论．

推论 2 给定模 m，任意 $a < m, (a,m) = 1$，利用 Euclid 算法计算 $a^{-1} (\mathrm{mod} m)$ 所需要的时间为 $O(\log_2^2 m)$.

推论 3 给定模 m，$\forall a < m$，利用 Euclid 算法计算一元一次方程 $ax \equiv b (\mathrm{mod}\, m)$ 所需要的时间为 $O(\log_2^3 m)$.

在多数的公钥密码算法中，最常用的一种运算是模的幂运算．模的幂运算基本上是所有基于分解因子问题与离散对数问题的公钥密码算法的主要运算内容．在给出模幂运算的时间估计之前，首先给出一种最为常用的计算模幂运算的算法—模平方求幂算法．

给定模 m，及 $0 \le a < m$，$0 \le x < \phi(m)$，下面计算 $a^x (\mathrm{mod} m)$：

1 计算 $x = (b_l, b_{l-1}, \cdots\cdots, b_0)_2$；

2 依次计算

$$a_1 = a^2 \bmod m,$$

$$a_2 = (a_1)^2 = a^{2^2} \bmod m,$$

……,

$$a_{l-1} = (a_{l-2})^2 = a^{2^{l-1}} \bmod m;$$

3 若 $b_0 = 0$，

$$a \leftarrow a^{b_0} \bmod m;$$

对于 $i = 1,....l-1,$

$$a \leftarrow a(a_i)^{b_i} \bmod m.$$

实际上，上述算法中 2、3 步可以同时进行，减少算法的内存空间.

于是得到下列定理

定理 4 给定模 m，及 $0 \le a < m$， $0 \le x < \phi(m)$，利用模幂平方运算计算 $a^x \pmod m$ 需要 $O(\log_2^3 m)$ 次比特运算.

下面给出多项式时间算法的定义.

定义 2 设 n_i 是 k_i $(1 \le i \le r)$ 位二进制数，如果存在 $d_i > 0$ $(1 \le i \le r)$，使得完成某个关于 n_i $(1 \le i \le r)$ 的算法需要 $O(k_1^{d_1} \cdots k_r^{d_r})$ 次比特运算，则称这个算法是**多项式时间算法**. 实际上，多项式时间算法就是要求存在多项式 $P(k_1, \cdots, k_r)$，使得比特运算次数不超过 $P(k_1, \cdots, k_r)$.

显然，推论 1 的算法不是多项式时间的.

一个要求给出解答的一般性提问，称作一个**问题**. 如分解因子问题、离散对数问题、求最大公因子问题等.

复杂性理论把问题分成了若干类，分类的依据是在图灵机(**DTM**)上解决问题中最难的实例所需要的时间和空间. 所谓图灵机是一种具有无限读-写存储带的有限状态机，由一个有限状态控制器，一个读写头和一条无限的读-写存储带. 工作时有限状态机给出一个状态，读写头作出响应，记录在读-写存储器上，直到过程结束. 这实际上就是计算机的一个理想模型.

定义 3 设 M 是问题，如果存在多项式 $P(x)$，对于任意的输入长度 n（这代表了问题的每一个实例）和解决这个实例在图灵机上运算的次数 $T_M(n)$ 使得， $T_M(n) \le P(n)$，则称这个问题是 P **类**的.

显然，前面所给出的加、减、乘、除运算、Euclid 算法、求模元素逆的算法、一元一次方程的算法等，如果看作问题的话，显然是 P 类的.

我们改造一下图灵机，构成所谓非确定型图灵机（**NTM**）：加入一个猜测头，它能把最

优的状态猜测并写出来，由有限状态控制器去验证．当一个判定性问题，在非确定型图灵机上的验证是有限步的，则称这个问题是 **NTM** 可解的．

定义 4　设 M 是判定性问题，如果存在多项式 $P(x)$，对于该问题的每一个猜测（实例），对于任意的输入长度 n 和验证这个实例在非确定性图灵机上运算的步数 $T_M(n)$，使得 $T_M(n) \le P(n)$，则称这个问题是 NP 类的．

首先，P 类是属于 NP 类的（$P \subseteq NP$）．但 $NP \subseteq P$ 却没有被证明．而且人们倾向于认为 $P \neq NP$，即存在某个问题是没有多项式时间算法的，虽然这也同样还没有证明．数论中所谓的困难问题，例如大数的因子分解问题是 NP 类的．许多公钥密码学算法的安全性就是建立在 $P \neq NP$ 这一假设的基础上的．由此可见算法复杂性理论在密码学中的重要性．

若 M_1、M_2 是两个判定性问题，存在一个多项式时间算法，可以把 M_1 的任意实例变换为 M_2 的某个实例，则称 M_1 可多项式归约为 M_2，记做 $M_1 \propto M_2$．两个可以相互多项式归约的问题称为是多项式时间等价的．更实际的，如果在某一个安全概率 ρ 以上（即要求把 M_1 的实例成功变换为 M_2 的某个实例的概率至少是 ρ）．两个问题是多项式时间等价的，则称为概率多项式时间等价．

定义 5　若判定问题 $M \in NP$，而对任意其他判定问题 M'，都有 $M' \propto M$，则称 M 属于 $NP-C$ 类．

如果存在这样一个判定问题 M 属于 $NP-C$ 类，并且能证明 M 有多项式时间算法（即 $M \in P$），那么显然有 $NP = P$；反之，如果有一个 NP 问题是难解的（没有多项式时间算法），那么所有的 $NP-C$ 问题都是难解的，即有 $NP-C \in NP-P$，这是密码工作者所希望的结果．

第一个被证明的 $NP-C$ 问题是逻辑表达式可满足性问题（**SAT** 问题）．定点覆盖问题和哈密尔顿回路问题也是 $NP-C$ 类．另外，公钥密码学常用的一个 $NP-C$ 问题是分解因子问题．

§2　分解因子问题

上节我们已经提到分解因子问题是 $NP-C$ 问题．密码学中有相当多的公钥密码算法均基于分解因子是困难的基础之上，也就是说，如果分解因子问题被多项式时间内求出或者在可能的计算资源的条件下可以分解，那么基于这些问题的密码算法即可破译．也就是说被加密的密文可以被破解出明文；数字签名可以被不法者伪造，更严重的是，有些电子钱币可以被不法者重复使用而被发现不了等严重问题．

因此对于这些困难问题，我们不仅要知道它是一个困难问题，而且需要及时了解这些困

难问题最快的求解算法，才可以确定关于该问题的安全参数的长度．如关于 $n = pq$ 分解因子问题早在 1976 年就已经被用于设计著名的公钥加密算法 RSA，其中 p, q 为两个长度相当的大素数．$n = pq$ 的安全长度从早期的 256 比特增加到现在的 1024 比特，反映了计算机速度的提高以及数学家对这些困难问题的计算方法的改进对于密码算法的一种影响．

下面简单描述以下几个分解因子算法时间估计．

分解大整数 n 的因子最古老的算法是 Eratoschenes 筛法，也称为**试除法**，也就是测试所有不大于 $\sqrt{n}$ 素数，看它是不是 n 的因子．这个算法要运行

$$O(\frac{\sqrt{n}}{\ln\sqrt{n}})$$

次除法．分解因子的求解方法早已经超出初等数论的范畴．目前分解因子的最快的方法有以下两种：一个方法是曾经被广泛应用的一个算法叫做**二次筛法**（quadratic sieve, **QS**），这个算法对小于 110 位的十进制数来说，到目前为止还是最快的．其渐近运行时间是

$$e^{(1+O(1))(\ln n)^{\frac{1}{2}}(\ln\ln n)^{\frac{1}{2}}}.$$

另一个算法是数域筛法（Number field sieve, **NFS**），它对于大于 110 位的十进制数的分解是已知最快的．**NFS** 的渐近运行时间是

$$e^{(1.923+O(1))(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}}.$$

因子分解是一个迅速发展的领域，毕竟数学理论的进展是不可预见的．降低计算复杂性的一条道路是找到更好的方法将数 n 表示成小系数的多项式，这也是正在研究的课题．

虽然所有的尝试都不能从根本上解决它，但毕竟还是有了一些进展，分解因子算法的提高结合计算机速度的提高将不断影响着安全参数 n 的长度的增加．我们回顾以下这两种提高对于安全参数 n 的长度的影响．

在 70 年代，根据当时计算机的计算速度以及分解因子的计算公式，找到一个 100 比特以上素数是困难的，因此当时认为 $n = pq$ 的安全长度为 256 比特．到 90 年代初，多台联网的计算机成功分解了一个 428 位的数，而且还不是用的最快的 **NFS** 算法．一般认为，对于一个大规模的组织来说，512 位的模数是易于攻击的．现行安全的模数的长度是 1024 位．

为了跟上因子分解的现状，RSA 公司在 1991 年设立了 RSA 因子分解难题，包含一系列从 100 位（十进制）到 500 位较难分解的数．在这一问题上的进展也从一个方面体现了分析人员的能力．

§3 素检测

在密码学中，在多数的公钥密码算法中，即使是一些非公钥密码算法，往往需要选取一些大的素数．给定一个随机数，判断一个数是否为素数简记为素判定问题．２００２年

印度的计算机科学家证明素判定问题是一个 P 问题.

本节的主要目的是描述两种素判定的概率算法.

引理 1 设 n 是素数，对于任意 $b \in Z_n^*$. 若

$$b^{\frac{n-1}{2}} \equiv (\frac{b}{n}) \pmod{n},$$

其中 $(\frac{b}{n})$ 是 Jacobi 符号.

定义 1 设 n 是奇合数，$(b,n)=1$. $(\frac{b}{n})$ 是 Jacobi 符号. 若

$$b^{\frac{n-1}{2}} \equiv (\frac{b}{n}) \pmod{n}, \qquad (1)$$

则称 n 是对基 b 的 Euler 型伪素数.

定理 1 设 n 是奇合数，在 Z_n^* 中至少有一半的 b 使得（2）式不成立.

证明 设 $b_1,\cdots,b_k$ 是 Z_n^* 中所有使得（1）式成立的不同数，$b \in Z_n^*$ 且使得（2）式不成立（这样的 b 肯定存在）. 由 Jacobi 符号的性质知，对于 $1 \le i \le k$，由于

$$b^{\frac{n-1}{2}} \not\equiv (\frac{b}{n}) \pmod{n},$$

$$b_i^{\frac{n-1}{2}} \equiv (\frac{b_i}{n}) \pmod{n},$$

故

$$(\frac{b}{n})(\frac{b_i}{n}) = (\frac{bb_i}{n}) \not\equiv (bb_i)^{\frac{n-1}{2}} \pmod{n}.$$

即 $bb_1,\cdots,bb_k$ 都不使（1）成立，而 $bb_1,\cdots,bb_k$ 是互不相同的，所以使得（1）式不成立的数也至少有 k 个. 由此得出结论.

定义 2 设 n 是奇合数，$n-1=2^s t$，$2 \nmid t$，又设 $b \in Z_n^*$，若

$$b^t \equiv 1 \pmod{n},$$

或

$$b^{2^r t} \equiv -1 \pmod{n},$$

对于某个r，$0 \le r < s$成立，则称n是对基b的强伪素数.

关于强伪素数，我们有

定理 2 设n是奇合数，$b \in Z_n^*$，则n是对基b的强伪素数的概率不大于$\frac{1}{4}$.

证明比较繁琐，有兴趣者可参考有关参考书. 性质 2 将用于下面 Rabin-Miller 素检测的概率估计.

下面介绍两种素检测算法：

1. Solovag-Strassen 算法

这个算法用 Jacobi 函数的性质测试p是否为素数，步骤如下：

（1）选择一个小于p的随机数a；

（2）求两数的最大公因数，若$(a, p) \neq 1$，则p是合数，不能通过测试；

（3）计算$j = a^{\frac{p-1}{2}} \pmod p$；

（4）计算 Jacobi 符号$(\frac{a}{p})$；

（5）如果$j \neq (\frac{a}{p})$，则p肯定不是素数；如果$j = (\frac{a}{p})$那么p通过测试，由性质 1，它不是素数的可能性不超过$\frac{1}{2}$；

（6）如果p通过测试，重新选取随机的a，重复以上测试t次，若p都通过，则p是合数的概率最大为$\frac{1}{2^t}$.

我们可以根据所需要的安全度确定t的大小.

2. **Rabin-Miller** 算法

这是一个简单而广泛使用的算法. 首先选择一个待测的p，计算b和奇数m，使得$p = 1 + 2^b m$. 然后进行

（1）选择小于p的随机数a；

（2）设定步数初值$j = 0$，并令$z \equiv a^m \bmod p$；

（3）若$z = 1$或$z = p - 1$，则p可能是素数，通过测试；

（4）步数值加 1，若$j < b$，赋值$z^2 \to z$；

（5）若$z = 1$，则p不是素数. 若$z = p - 1$则p可能是素数，通过测试. 否则回

到第（4）步；

（6）如果 $j=b$ 且 $z\neq p-1$，则 p 不是素数.

这个算法比上一个更快，因为由性质 2，a 指证 p 可能是素数的概率是 $\frac{3}{4}$.

最后，我们通过一个定理说明在广义 Riemann 成立的条件下，素判定是多项式的.

定理 3 若广义 Riemann 猜想成立，则存在一个常数 $C>0$ 使得，使得对于大于 1 的任何奇数 n，下列结论是等价的：

（1）n 是素数；

（2）对所有的 $a\in Z_n^*$，且 $a<C(\log n)^2$，（1）式成立.

§3. RSA 问题与强 RSA 问题

RSA 问题是针对著名的公钥加密算法 RSA 提出的. 它标志着 RSA 加密算法（签名算法）的安全强度. 虽然没有人能够证明 RSA 问题的难度是否等价于大整数分解的难度，但是人们相信是 RSA 问题是一个困难问题，并且将 RSA 问题作为一个判断许多密码算法安全的依据. 也就是说，如果一个密码算法破译的难度等价于 RSA 问题，则该算法被公认为是安全的. RSA 问题概念是由 Rivest, Shamir 和 Adleman 于 1978 年提出的（这也是“RSA”的由来）.

设 p,q 是两个二进制长度相当的大素数，$n=pq$ 并且满足 n 的二进制长度不小于 1024 比特，而且 $p-1,q-1$ 都有大的素因子，则称 n 为 RIPE 合数.

定义 1（RSA 问题） 给定一个 RIPE 合数 $n=pq$ 和以及一个满足 $(e,\varphi(n))=1$ 的正奇数 e，对于任给的随机整数 $c\in Z_n^*$，求满足

$$m^e\equiv c(\mathrm{mod}\, n)$$

的整数 m.

由定义知，RSA 问题即为求 Z_n^* 中的 e-次根，与一般的 e-次根不同的是 $n=pq$，且 e 为正奇数. RSA 假设就是认为上面的 RSA 问题没有概率多项式时间算法. 而相应的大数分解问题假设则是说对 n 不存在多项式时间算法可以分解出 $n=pq$.

在 RSA 问题的基础上，Baric, Pfitzmann 和 Fujisaki,Okamoto 两组于 1997 年又各自引进了强 RSA 问题和假设.

定义 2（强 RSA 问题） 设 $n=pq$ 是一个 RSA 问题的模，G 是 Z_n^* 的循环子群. 给

定 n 对于任意随机选取的 $z \in G$，求 $(u,e) \in G \times Z_n$，满足

$$z \equiv u^e \pmod{n},$$

其中 l 是安全参数，一般取为 n 的二进制长度.

实际上强 RSA 问题就是求模 n 的任意次方根问题. 而强 RSA 假设是说，对任意的多项式 $P(l)$，使得能够在多项式时间内找到

$$z \equiv u^e \pmod{n}$$

的解 $(u,e) \in G \times Z_n$ 的概率小于 $\dfrac{1}{P(l)}$.

1998 年，J.Camenisch 和 M.Michels 在强 RSA 假设的基础上提出了一个安全有效的群签名方案.

一般的，基于 RSA 问题及其派生的 RSA 问题（如强 RSA 问题）的密码算法通常被划为基于分解因子系列的密码算法之列. RSA-类问题的提出，使设计基于分解因子问题的密码算法更加灵活，密码算法的范围更加广阔. 使密码算法从加密算法、一般的签名方案扩展到各种数字签名方案、群签名方案乃至电子钱币等应用性很强的密码算法，并且能够保证密码算法的安全性.

§4 二次剩余

二次剩余问题在初等数论第 3 章第 5 节我们已经较详细的讨论过，由于二次剩余问题在密码学中的诸多应用，我们在此就二次剩余在密码学中涉及到的诸多特性进行讨论.

不加特别说明，本节所讨论的二次剩余均是 Z_n^* 中的二次剩余问题，且 n 满足如下条件：

$$n = pq, \quad p \equiv 3 \pmod{4}, \quad q \equiv 3 \pmod{4} \qquad (1)$$

我们称满足（1）的数为 **Blum 数**. 从而有下列结论：

定理 1 若 n 为 Blum 数，则 -1 为 Z_n^* 中的非二次剩余并且 $(\dfrac{-1}{n}) = 1$.

证明 由于 -1 为 Z_n^* 中的二次剩余的充要条件是 -1 分别为模 p、q 的二次剩余. 所以 -1 为 Z_n^* 中的非二次剩余. 由 Jacobi 符号的性质有

$$(\frac{-1}{n}) = (\frac{-1}{p})(\frac{-1}{q}) = (-1)(-1) = 1.$$

得证.

在一些密码算法中，-1的以上两个特性是非常重要的，这也正是给出 Blum 数定义的一个原因.

定义 1（模 n 二次剩余问题）设 $n=pq$，是两个大素数的乘积，随机选取 $a \in QR_n$，求 x 使得

$$x^2 \equiv a(\mathrm{mod}\, n)$$

在本节中，我们将证明二次剩余假设概率多项式价于 n 的分解因子问题.

关于模 $n=pq$ 的二次剩余问题，我们有进一步的结论：

定理 1 任给 $a \in QR_n$，则 a 有 4 个模 n 的平方根.

证明 由中国剩余定理知，

$$x^2 \equiv a(\mathrm{mod}\, n)$$

的解等价于方程组

$$\begin{cases} x^2 \equiv a(\mathrm{mod}\, p) \\ x^2 \equiv a(\mathrm{mod}\, q) \end{cases},$$

的解.

设

$$x^2 \equiv a(\mathrm{mod}\, p)$$

有两个根 $\pm x_0$，$x^2 \equiv a(\mathrm{mod}\, q)$ 的两个根 $\pm x_1$. 则

$$x^2 \equiv a(\mathrm{mod}\, n)$$

有以下四个根

$$\begin{cases} x \equiv \pm x_0(\mathrm{mod}\, p), \\ x \equiv \pm x_1(\mathrm{mod}\, q). \end{cases}$$

定理 2 每一个二次剩余的四个平方根中仅有一个平方根是模 n 的二次剩余.

证明 设 a 是模 n 的二次剩余，由定理 1 可进一步证明 a 的 4 个平方根，可以设为 $x, -x, y, -y$ 并且

$$x \equiv y(\mathrm{mod}\, p)，\quad x \equiv -y(\mathrm{mod}\, q).$$

于是有 Jacobi 符号 $(\frac{x}{n}) = -(\frac{y}{n})$. 又不妨设 x 是使得 $(\frac{x}{n}) = 1$ 的那个平方根，从而要么

$$(\frac{x}{p})=(\frac{x}{q})=1,$$

要么

$$(\frac{-x}{p})=(\frac{-x}{q})=1,$$

这样 x，$-x$ 中一定有一个是模 n 的二次剩余.

从上面两个定理进一步可以知道，每个二次剩余 a 有 4 个平方根 x_1,x_2,x_3,x_4，它们满足

$$(\frac{x_1}{p})=(\frac{x_1}{q})=1,\quad (\frac{x_2}{p})=(\frac{x_2}{q})=-1,$$

$$(\frac{x_3}{p})=-(\frac{x_3}{q})=1,\quad (\frac{x_4}{p})=-(\frac{x_4}{q})=-1.$$

其中 x_1 也是模 n 的二次剩余.

二次剩余在密码学中的重要性还体现在以下结论.

定理 3　n 的因子分解问题的算法复杂性概率多项式等价于求 n 的二次剩余问题.

证明　首先证明如果存在一个多项式时间算法 A，对所有输入的 n，输出 n 的一个素因子，那么一定存在多项式时间算法 B，对所有输入为 (n,x)，x 是 n 的二次剩余，输出 x 的一个平方根.

定义算法 B 为：

（1）　设 $A(n)=p$，$q=\frac{n}{p}$.

（2）　根据定理 2 即可求出模 n 的四个平方根

其次证明如果存在多项式时间算法 B，对所有输入为 (n,x)，x 是 n 的二次剩余，输出 x 的一个平方根，那么存在一个多项式时间算法 A，对所有输入的 n，以 1/2 的概率输出 n 的一个素因子.

我们如下定义算法 A：

（1）　选择一个随机数 a 使得 $(\frac{a}{n})=-1$．输入 $x\equiv a^2(\mathrm{mod}\,n)$，$B$ 输出 x 的一个平方根 b.

（2）　若 $(\frac{b}{n})=1$，计算 $g.c.d(a-b,n)$ 或 $g.c.d(a+b,n)$ 即得到 n 的一个素因子（为

什么？）．得证．

最后我们给出一个假设：

定义 2（判断模 n 二次剩余问题）设 $n = pq$，是两个大素数的乘积，随机选取 $a \in Z_n^*$，判断是否存在 x 使得 $x^2 \equiv a(\mathrm{mod}\, n)$．

关于判断模 n 的二次剩余问题的难度，到目前为止还没有人给出证明．但大家认为它是困难的．该问题可以用于设计概率加密算法等密码算法．

§4 离散对数问题

应用密码学中经常用到的另一个困难问题是离散对数问题．离散对数问题与分解因子问题构成了公钥密码学两大数学难题，因此基本上将公钥密码学分成两大系列．如同 RSA 问题派生出强 RSA 问题一样，离散对数问题也派生几个困难问题及其假设．

定义 1 设 g 为 Z_p^* 的原根（也可以说成有限循环群 Z_p^* 的生成元），任给元素 $y \in Z_p^*$，求唯一的 x，$1 \le x < p-1$ 使得

$$g^x \equiv y(\mathrm{mod}\, p),$$

称为以 p 为模，以 g 为底 y 的**离散对数**．

由第 4 章第 3 节知，求离散对数问题即为求指标问题．到目前为止还不存在多项式时间算法，已知的最好的计算法 NFS（数域筛选法），它的渐进时间估计为

$$e^{(1.923+O(1))(\ln(p))^{(1/3)}(\ln(\ln(p)))^{(2/3)}}.$$

从 Z_p^* 离散对数问题与 n 的分解因子的数域筛法估计式看，两者的难度似乎相当．但到目前为止，没有人能够证明这两个问题当中，谁更困难．

对于 Z_n^* 中同样可以定义离散对数问题，我们举一个 $n = pq$（p, q 为两个大素数）的例子．

例 1 设 $n = pq$，p, q 为两个大素数且 $(\frac{p-1}{2}, \frac{q-1}{2}) = 1$，存在 $a \in Z_n^*$，使得

$$\delta_n(a) = \lambda(n) = [(p-1),(q-1)] = \frac{(p-1)(q-1)}{2}$$

证明 设 g, h 分别为 p, q 的原根并且

$$\begin{cases} a \equiv g(\mathrm{mod}\ p) \\ a \equiv h(\mathrm{mod}\ q) \end{cases}$$

则根据孙子定理

$$a \equiv q^{p-1} g + p^{q-1} h(\mathrm{mod}\, n)\ .$$

$$\delta_p(a) = p-1,\ \ \delta_q(a) = q-1,$$

由指数的性质得

$$\delta_n(a) = [\delta_p(a), \delta_q(a)] = [(p-1),(q-1)] = \frac{(p-1)(q-1)}{2}\ .$$

定义 2 有限循环群 Z_p^*， g,h 分别为 p 的不相关原根（即 g 对 h 的离散对数未知），对于任何 $(a, p) = 1$，将 a 表示为

$$a \equiv g^{\alpha} h^{\beta} (\mathrm{mod}\ p)\ ,$$

(α, β) 称为以 p 为模以 g,h 为底 a 的一个表示.

上面的定义同样可以扩充到多个原根的情况.

定义 3 有限循环群 Z_p^*， $g_1, g_2, \cdots, g_s$ 分别为 p 的不相关原根，对于任何 $(a,n) = 1$，将 a 表示为

$$a \equiv g_1^{\alpha_1} g_2^{\alpha_2} \cdots g_s^{\alpha_s} (\mathrm{mod}\ p)\ ,$$

$(\alpha_1, \alpha_2, ..., \alpha_s)$ 称为以 p 为模以 $g_1, g_2, \cdots, g_s$ 为底的 a 的一个表示.

a 的这种表示形式在群签名、可以追踪身份的盲签名有非常重要的应用．特别是可以追踪身份的盲签名可以用于设计电子钱币方案.

另外，我们还要介绍一个与离散对数问题相关的困难问题假设.

定义 3（**Diffie-Hellman** 问题）设 g 为有限循环群 Z_p^* 的生成元，对任意的 $a,b \in Z_P^*$，

$$a \equiv g^x (\mathrm{mod}\ p),\ \ b \equiv g^y (\mathrm{mod}\ p)\ ,$$

在 x, y 未知情况下，求

$$c \equiv g^{xy} (\mathrm{mod}\ p)\ .$$

如同 RSA 问题是 RSA 加密算法的安全衡量标准一样，Diffie-Hellman 问题（简记 D-H 问题）是 Diffie-Hellman 密钥交换体制的安全衡量标准．D-H 问题的难度至今没有人能够证明，但是一般大家都认为 D-H 问题是困难的．D-H 问题与离散对数问题有以下关系：如果离散对

数问题是多项式时间可计算的，则 D-H 问题也是多项式时间可计算的．反过来未必成立，但是可以证明，p 满足一定的条件时，两者是等价的，当然这些情况下的离散对数问题仍是困难的，否则就失去了两个问题在密码学中的作用．

最后，我们描述一下椭圆曲线上的离散对数问题．在群论中作为有限群的例子，我们已经引入了椭圆曲线的一般定义，并介绍了椭圆曲线上的有理点群．这种群的结构及特性，特别是定义在群上的离散对数问题在密码学中是非常重要的，它是基于椭圆曲线一系列密码算法的理论根据。而基于椭圆曲线的密码算法目前仍是密码学中研究的热点之一．

密码算法中所用到的椭圆曲线通常是如下形式．

定义 4 设 p 为大于 3 的素数，有限域 Z_p^* 上的椭圆曲线 $E_P(a,b)$ 是同余方程

$$y^2 \equiv x^3 + ax + b(\mathrm{mod}\ p) \qquad (1)$$

的解 $(x,y) \in Z_p^* \times Z_p^*$ 的集合，及一个无穷远点 O 组成．

定义 2 设 G 是有限域 Z_p^* 的椭圆曲线群的循环子群，点 P 是 G 的生成元 $G =< P >$，对任给的点 $Q \in G$，求正整数 l，使得 $Q = lP$，这就是椭圆曲线上的离散对数问题．

单从代数结构来看，椭圆曲线上的离散对数问题对应着相应的整数群中的离散对数问题．所不同的是，两者的具体运算不同，一个是具有几何特性的加运算，一个是整数模乘运算．