第3章 公钥密码学

公钥密码学的基本思想是公开密钥。这里，每一个用户的密钥分为两个部分：一个是任何人都可以使用的，作为加密用的公开密钥；一个是只有用户本人使用且严格保密的解密用的私钥。本章先介绍公钥密码学的基本概念，接着讨论公钥密码学中最重要的一些例子，比如 RSA、ELGamal 和 Rabin 等密码体制，它们都可以用于加密和签名。

3.1 公钥密码学基本概念

经典的对称密码学为每一对用户提供了一个安全的信道。为建立这个信道，用户必须对某一个公用的密钥达成共识。这个安全信道的建立保证了消息的秘密性。对称密码学也包括了消息篡改检测和消息源认证的一些方法。因此，使用私钥技术就可以获得保密性和完整性。

但在秘密钥的安全分配中，必须用到公钥技术，一些重要的认证和不可否认性体制中也要求使用公钥技术，比如数字签名。数字签名是手写签名的数字形式，依赖于被签消息和签名人的一些个人秘密特征。一个可信第三方可以在不知道签名人秘密特征的情况下验证签名的合法性。

在公钥加密体制中，一对用户之间没有共享的秘密钥，每个用户有一对密钥：秘密密钥(secret key) sk(简称私钥)——只有自己知道，公开密钥(public key) pk(简称公钥)——体制中的每个用户都知道。以这些公钥为参数，确定一个加密函数族 $(E_{pk})_{pk\in PK}$，其中 PK 表示这些公钥的集合，这个加密函数族 $(E_{pk})_{pk\in PK}$ 也是公开的。

假设 Bob 参与了这个公钥加密系统，Alice 希望传给 Bob 一个加密的消息 m。Bob 有一个私钥 sk 和一个公钥 pk。和系统中其他人一样，Alice 知道 Bob 的公钥 pk。她只需要使用 Bob 的加密函数 E_{pk} 计算 $c=E_{pk}(m)$ 就可以了。显然，这个系统仅在通过 $c=E_{pk}(m)$ 计算 m 不可行的情况下才是安全的。而 Bob 又如何从 c 中恢复出明文 m 呢？这就用到了 Bob 的私钥。加密函数 E_{pk} 必须具有这样的性质——运用 Bob 的私钥可以十分简单的计算出 E_{pk} 的原像 m，因为只有 Bob 知道秘密密钥 sk，也就只有他能解密消息。即使消息的加密者 Alice 在丢失了原始消息 m 的情况下，她也不能通过 $E_{pk}(m)$ 恢复出明文

m。当然,有效算法是以上加、解密操作的基础。

我们来总结一下公钥密码系统的要求。我们必须找到这样的一个函数族,其中的每一个函数 f 都有有效的算法来计算它,而计算函数 f 的原像是不可行的。这样的函数族称为单向函数族(one-way function)(确切定义见定义 6.12)。对这个族中的每一个函数,如果存在一些保密信息,这些保密信息使得计算 f 的逆有效可行,那么这些秘密信息称为陷门信息(trapdoor information),有这种性质的函数称为陷门函数(trapdoor function)。

1976 年,W. Diffie 和 M. E. Hellman 在他们著名的论文《密码发展新方向》(参见[DifHel76])中提出了公钥密码学的思想,给出了一种密钥协商的方法,这种方法在公钥密码学中使用至今。除此之外他们还描述了数字签名的作用,并以公开问题的形式提出了寻找单向函数的课题。第一个既可作为密钥协商又可作为数字签名的密码学体制是 1978 年提出的 RSA 密码体制(参见[RivShaAdl78])。RSA 是以其 3 个开发者:R. Rivest、A. Shamir 和 L. Adleman 命名的。RSA 是现今使用最广泛的公钥密码体制,它提供了加密以及数字签名的功能。我们将在 3.3 节对它进行详细的讨论。RSA 密码体制的安全性是基于大整数分解的困难性,这个问题引入了一个带有陷门的单向函数。另一类单向函数是基于求解离散对数的困难性。这两个数论上的难题是现在大多数公钥密码体制的基础。

每一个公钥密码体制的用户都拥有一对个人密钥:$k=(pk,sk)$,包括公开密钥 pk 和秘密钥 sk。为了保证密码系统的安全,必须确保从公钥 pk 计算私钥 sk 是不可行的,密钥空间也必须足够大,还要存在有效的算法实现随机选择密钥。如果一个用户 Bob 希望参与到这个密码系统,他随机地选择密钥 $k=(pk,sk)$,sk 保密,pk 公开,那么现在每个人都可以使用 pk 为 Bob 加密消息。

为讨论数字签名的基本概念,假定陷门函数组成一个双射函数的族。这样的一个陷门置换族可以用来实现数字签名。令 f 是与 Alice 的公钥相关的函数。而计算 f 的逆 f^{-1}需要 Alice 的私钥。因此 Alice 是唯一可以执行求逆操作的人。如果 Alice 希望对消息 m 签字,她计算 $f^{-1}(m)$,并将此值 s 作为 m 的签字,任何人都可以使用 Alice 的公开加密函数 f 对此签名进行验证。如果 $f(s)=m$,Bob 就可以确认 Alice 确实签署了这个消息,因为只有她才能计算 $f^{-1}(m)$。

公钥密码系统的一个重要的直接应用是分配会话密钥。会话密钥是指在经典对称加密体制中,一次加密会话中使用的密钥。如果 Alice 知道 Bob 的公钥(并且信任它),她生成一个会话密钥,用 Bob 的公钥加密后传送给 Bob。认证中心使用数字签名来确定公钥的可靠性,用每个用户的私钥签发他们的公钥,这个签字可以用每个用户的可信公钥来验证。密码协议也有了很大的发展,可以用来进行用户认证和更高级的协议设计,比如,比特承诺系统和不经意传输以及交互式零知识证明。今天,它们都是网上通信和电子商务的基础。

公钥密码学对理论计算机科学也有很重要的意义：安全性理论已有了很大的进展，应当提及的是它推动了复杂性理论的研究。

3.2 模算术

本节对模算术给出一个简要的介绍，为讨论本章的公钥密码学做一些必要的理论准备。详见附录 A。

3.2.1 整数

令**Z** 表示有序集{…，−3，−2，−1，0，1，2，3，…}，**Z** 中元素称为整数或数字。大于 0 的整数称为自然数，用符号**N** 来表示。定义 $n+m$ 为和，$n \cdot m$ 为积，加法和乘法满足有单位元的交换环的性质，我们称**Z** 为整数环。

加法，乘法和指数运算。在整数[①]运算中，加法和乘法都有可行有效的计算方法。我们所说的有效算法是指以输入长度的多项式时间运行的算法。这里一个数的长度是指它的二进制编码长度。即对于 $n \in \mathbf{N}$，其长度等于 $\lfloor \log_2(n) \rfloor + 1$[②]，用 $|n|$ 来表示。令 $a, b \in \mathbf{N}$，$a, b \leqslant n$ 且 $k := \lfloor \log_2(n) \rfloor + 1$，计算 $a+b$ 的比特操作次数为 $O(k)$，计算 $a \cdot b$ 乘法的操作次数是 $O(k^2)$。如果使用一种快速算法，乘法的操作次数可以降低到 $O(k \log_2(k))$。

指数运算也是常用到的一种运算，利用重复乘方法（算法 A.26）可以更加有效地计算 a^n，最多需要 $2 \cdot |n|$ 次模乘法运算。例如计算 a^{16}，用 $(((a^2)^2)^2)^2$ 来计算，进行 4 次平方运算，而原始的算法需要进行 15 次乘法运算。如果指数不是 2 的幂次，那么这种运算需要做些修正。例如，计算 $a^{14} = ((a^2 a)^2 a)^2$，用 3 次平方运算和两次乘法运算取代 13 次乘法运算。

带余除法。如果 m 和 n 为整数，且 $m \neq 0$，可以用 m 除 n，进行带余数的除法。如果 $0 \leqslant r < \mathrm{abs}(m)$[③]，可唯一地写成，$n = q \cdot m + r$。在这种除法中，$q$ 称为商（quotient），r 称为余数（remainder）。它们都是唯一的，我们常用 $a \bmod b$ 来表示 r。

当整数 n 是整数 m 的倍数时，即存在整数 q，使得 $n = mq$，在这里我们说整数 m 整除整数 n，m 是 n 的一个除数（divisor）或因子（factor）。m 和 n（$m, n \neq 0$）的最大公因子（great common divisor）$\gcd(m, n)$，表示整除 m 和 n 的最大正整数，$\gcd(0, 0)$ 定义为 0。

① 简单起见，在这里我们只考虑非负整数，对于本书这已经足够了。

② $\lfloor x \rfloor$ 表示不超过 x 的最大整数。

③ $\mathrm{abs}(m)$ 表示 m 的绝对值。

如果 $\gcd(m,n)=1$，则说 m 是 n 的相对素数(relatively prime to)，简单地说就是 m 与 n 互素(prime to)。

计算两个数最大公因子的欧几里得算法(Euclidean algorithm)是数学领域中最古老的算法之一。

算法 3.1

```
int gcd(int a,b)
1  while b≠0 do
2     r←a mod b
3     a←b
4     b←r
5  return abs(a)
```

这个算法用来计算 $a\neq 0$ 和 $b\neq 0$ 情况下的 $\gcd(a,b)$。因非负数 r 的逐步减小而终止，要注意在每一轮中 $\gcd(a,b)$ 是不变的，因为 $\gcd(a,b)=\gcd(b,a \bmod b)$。最后一步，余数 r 减为 0，我们就得到 $\gcd(a,b)=\gcd(a,0)=\mathrm{abs}(a)$。

素数与分解。如果一个自然数 $p\neq 1$ 只有两个因子 1 和 p，我们说 p 为素整数(prime number)简称素数(prime)。如果一个数 $n\in\mathbf{N}$ 不是素数，则称之为合数(composite)。素数是本章中我们构建公钥密码系统的基础。幸运的是，对于判断一个数是否为素数，存在十分快速的算法(即所谓的概率素性检测)以很高的概率(虽然不是数学上的“确定”)做出判断(见附录 A.7)。素数是数的基本构件，这一点可通过算术基本定理(Fundamental Theorem of Arithmetic)来精确陈述。

定理 3.2(算术基本定理) 令 $n\in\mathbf{N}, n\geqslant 2$，存在不同的素数 $p_1,\cdots,p_k$ 及其与之对应的指数 $e_1,\cdots,e_k\in\mathbf{N}, e_i\geqslant 1, i=1,\cdots,k$，使得

$$n=\prod_{i=1}^{k} p_i^{e_i}$$

其中素数 $p_1,\cdots,p_k$ 和 $e_1,\cdots,e_k\in\mathbf{N}$ 唯一确定。

两数的相乘很好计算，但是构造求解一个数的素因子的算法却是一个古老的数学难题。例如在二百多年前，著名数学家 C. F. Gauβ 就曾经研究过这个问题(例：详见[Riesel94]Gauβ 分解方法)。但是，直到今天，我们都没有一个切实可行的算法分解极大的数。

3.2.2 整数模 n

模 n 剩余类环。假设 n 为正整数，a,b 为整数。如果 a 和 b 除以 n 得到相同的余数，也就是 n 可以整除 $a-b$，那么就说 a 同余(congruent)于 b 模 n，记为 $a\equiv b \bmod n$。这样我

们就获得了一个等价关系。a 的等价类为所有与 a 同余的数的集合，用$[a]$来表示，称为 a 模 n 的剩余类(residue class)。剩余类集合$\{[a] \mid a \in \mathbf{Z}\}$称为整数模 n(integers modulo n)集合，用$\mathbf{Z}_n$表示。

每个数都与一个唯一的 $r(0 \leqslant r \leqslant n-1)$同余，因此，数 $0, \cdots, n-1$ 构成了一个$\mathbf{Z}_n$中元素代表的集合，称它们为$\mathbf{Z}_n$的自然表示(natural representative)。

在$\mathbf{Z}$ 中，等价关系与加法和乘法具有结合律，即，如果 $a \equiv a' \bmod n$，$b \equiv b' \bmod n$，那么有 $a+b \equiv (a'+b') \bmod n$ 以及 $a \cdot b \equiv (a' \cdot b') \bmod n$，因此，$\mathbf{Z}$ 中加法和乘法演化出$\mathbf{Z}_n$中加法和乘法：

$$[a]+[b] := [a+b]$$
$$[a] \cdot [b] := [a \cdot b]$$

加法和乘法满足有单位元的交换环中的运算法则。我们称$\mathbf{Z}_n$为模 n 剩余类环(residue class ring modulo n)。

虽然我们可以像在$\mathbf{Z}$ 中一样在$\mathbf{Z}_n$中进行运算，但是这两者有一些很重要的差异。首先，我们不能像在$\mathbf{Z}$ 中一样，根据加法和乘法对$\mathbf{Z}_n$中元素排序。例如，假设在$\mathbf{Z}_5$中我们有这样的排序，$[0]<[1]$，那么就有$[0]<[1]+[1]+[1]+[1]+[1]=[0]$，显然是矛盾的。类似计算可以证明假设$[1]<[0]$也推出矛盾。

另外一个事实是，对于$[a] \neq 0$ 和$[b] \neq 0$，$[a] \cdot [b]$可以为 0，例如，在$\mathbf{Z}_6$中$[2] \cdot [3]=[0]$。这样的元素——$[a]$和$[b]$——称为零因子(zero divisor)。

模 n 的素剩余类群。在$\mathbf{Z}$ 中，当且仅当 a 和 b 都为 1 时或都为 -1 时，才能满足 $a \cdot b=1$。我们说 1 和 -1 有乘法逆元素。而在$\mathbf{Z}_n$中，这种情况经常发生。比如，在$\mathbf{Z}_5$中，不是$[0]$的任何一个剩余类中元素都有乘法逆元。环中具有乘法逆元的元素，都称为单元(unit)，它们构成一个乘法群。

模 n 的素剩余类群$\mathbf{Z}_n$中元素$[a]$，如果存在元素$[b]$，满足$[a] \cdot [b]=1 \bmod n$，或者等价地，n 整除 $1-a \cdot b$，那么就说$[a]$有逆元。这意味着对合适的 m，存在一个等式 $nm+ab=1$。这个等式暗示 $\gcd(a,n)=1$。另一方面，如果给定数字 a，n 满足 $\gcd(a,n)=1$，那么根据扩展欧几里得算法(算法 A.5)，可以从 a 和 n 推导出合适的 b 和 m，满足等式 $nm+ab=1$。因此，$[a]$是$\mathbf{Z}_n$中单元，逆元为$[b]$。$\mathbf{Z}_n$单元群中的元素用与 n 互素的数来表示：

$$\mathbf{Z}_n^* := \{[a] \mid 1 \leqslant a \leqslant n-1 \text{ 且 } \gcd(a,n) = 1\}$$

称为模 n 的素剩余类群。$\mathbf{Z}_n^*$ 中元素的个数(又称为$\mathbf{Z}_n^*$ 的阶)是在$[1,n-1]$间与 n 互素的整数的个数。一般用 $\varphi(n)$来表示，函数 φ 称为欧拉函数(Euler totient function)。

对于有限群 G 中任意元素 a 而言，我们有 $a^{|G|}=e$，e 为 G 的单位元[①]，这是有限域中基本的性质，易于证明。这样，对于与 n 互素的数 a 有

$$a^{\varphi(n)} \equiv 1 \bmod n$$

称此命题为欧拉定理(Euler's Theorem)，或者，如果 n 为素数，称为费马定理(Fermat's Theorem)。

如果 $\prod_{i=1}^{k} p_i^{e_i}$ 是 n 的素分解，那么欧拉函数可以用以下公式计算(见推论 A.30)

$$\varphi(n)=n\prod_{i=1}^{k}\left(1-\frac{1}{p_i}\right)$$

如果 p 为素数，那么$\{1,\cdots,p-1\}$中每一个整数都与 p 互素。因此，$\mathbf{Z}_p\backslash\{0\}$中每一个元素都是可逆的，$\mathbf{Z}_p$是一个域。$\mathbf{Z}_p$中单元构成的群是一个有 $p-1$ 个元素的循环群，即对于 $g\in\mathbf{Z}_{p-1}$，有$\mathbf{Z}_p^*=\{g,g^2,\cdots,g^{p-1}=1\}$，这样的 g 称为$\mathbf{Z}_p^*$的生成元，生成元又称为模 p 本原元或模 p 本原根(见定义 A.37)。

下面介绍 3 个函数，这 3 个函数都可以用作为单向函数，因此在密码学中有很重要的意义。

离散指数。令 p 表示一个素数，g 为$\mathbf{Z}_p$中本原根，称

$$\mathrm{Exp}:\mathbf{Z}_{p-1}\to\mathbf{Z}_p^*,\quad x\mapsto g^x$$

为离散指数函数。Exp 是加法群$\mathbf{Z}_{p-1}$到乘法群$\mathbf{Z}_p^*$的同态映射，即 $\mathrm{Exp}(x+y)=\mathrm{Exp}(x)\cdot\mathrm{Exp}(y)$，Exp 还是一个双射。换句话说，Exp 是一个群间的同构映射，这与本原根的定义相符。其逆函数定义如下，称

$$\mathrm{Log}:\mathbf{Z}_p^*\to\mathbf{Z}_{p-1}$$

为离散对数函数(discrete logarithm function)。在这里我们使用形容词“离散”来区别实数域中经典定义的指数和对数函数来区别 Exp 和 Log 这两个有限域中的函数。

Exp 函数可有效计算，比如利用重复平方法(见 3.2.1 节)。但是，普遍认为，在 p 为大素数的情况下，对于计算其逆函数，Log 函数，不存在有效的算法。离散对数假设(见定义 6.1)精确地给出了以上的论述。

模幂。令 n 表示两个不同素数 p 和 q 的乘积，e 与 $\varphi(n)$互素，称

$$\mathrm{RSA}_e:\mathbf{Z}_n\to\mathbf{Z}_n,\quad x\mapsto x^e$$

为 RSA 函数(RSA function)。

命题 3.3 与上面的标示符号相同，d 为 e 模 $\varphi(n)$的乘法逆元(注意在这里 d 也与 $\varphi(n)$互素，定义函数RSA_d)。那么有

① $|G|$表示 G 中元素的个数(称为 G 的基数(cardinality)或阶数(order))。

$$\mathrm{RSA}_d \circ \mathrm{RSA}_e = \mathrm{RSA}_e \circ \mathrm{RSA}_d = \mathrm{id}_{\mathbf{Z}_n}$$

证明：对于 $x \in \mathbf{Z}_n$，我们来说明 $x^{ed} = x$。首先令 $x \in \mathbf{Z}_n^*$，群 $\mathbf{Z}_n^*$ 的阶为 $\varphi(n)$，因此有，$x^{\varphi(n)} = 1$ 进而有 $x^{ed} = x^{ed \bmod \varphi(n)} = x$。当 $x \notin \mathbf{Z}_n^*$ 时，p 或 q 为 x 的因子。如果它们都整除 x，则有 $x = 0$ 和 $x^{ed} = 0$，这样等式成立。注意到 $\varphi(n) = (p-1)(q-1)$（见推论 A.30）。如果 p 整除 x 而 q 不整除 x，则有 $(x^e)^d \bmod p = 0$ 和 $x \bmod p = 0$，因为 $ed \equiv 1 \bmod (q-1)$，有 $(x^e)^d \equiv x^{ed \bmod (q-1)} \equiv x \bmod q$，这说明 $(x^e)^d \equiv x \bmod n$。当 q 整除 x 而 p 不整除 x 时，情况与之类似。综上，对于所有的 $x \in \mathbf{Z}_n$，有 $(x^e)^d = x^{ed} = x$，这样就证明了我们的论断。

可以看出 RSA_e（易于计算）是 $\mathbf{Z}_n$ 中的一个置换。在知道 d 的情况下求逆也很容易。但是，如果 d 保密，则普遍认为 RSA_e 函数是不可逆的（如果 p 和 q 很大）。

模平方。令 p 和 q 表示不同的素数，$n = pq$：

$$\mathrm{Square}: \mathbf{Z}_n \to \mathbf{Z}_n, \quad x \mapsto x^2$$

称为平方函数（square function）。对于每一个 $y \in \mathbf{Z}_n$，$y \neq 0$，有 0 个，2 个或是 4 个原像。如果 p 和 q 都是模 4 同余于 3 的素数，则 -1 就不是模 p 和模 q 的平方根，那么根据简单的推断，在 $\mathbf{Z}_n^*$ 中，限制在一定的部分和平方的子集范围，Square 函数可以成为一个单射（详见附录 A.5）。如果知道 n 的分解，那么平方函数的原像（即平方根）可以有效地计算（见命题 A.57）。同样的，在不知道因子 p 和 q 的情况下，当 p 和 q 足够大的时候，计算平方根实际上是不可行的。

关于开方运算和求解离散对数的困难性。令 $g, k \in \mathbf{N}, g, k \geqslant 2$，令

$$F: \mathbf{Z} \to \mathbf{Z}$$

表示映射 $x \mapsto x^2$、$x \mapsto x^k$、$x \mapsto g^x$ 中的一个。一般而言，这些映射都是在实数域中的，已经有了计算函数值和原像的有效算法。这些算法的存在主要依赖于实数的有序性和 F 的单调性，至少也是部分单调。利用整数算术中的快速指数算法（见 3.2.1 节），映射 F 可以有效地计算，如果使用这样的算法计算 F，我们得到了以下计算 $y = F(x)$ 原像 x 的算法。

简单起见，将值域局限于正整数集上，这样这 3 个函数就都是单调的，因此是单射。容易找到整数 a 和 b 满足 $a \leqslant x \leqslant b$。如果 $F(a) \neq y$ 且 $F(b) \neq y$，有以下 FInvers(y, a, b)：

算法 3.4

```
int FInvers(y, a, b)
1  repeat
2     c←(a+b)div2
3     if F(c)<y
4        then a←c
5        else b←c
```

6 until $F(c)=y$

7 return c

F 可以有效地计算。以上算法经过 $O(\log_2|b-a|)$步重复后，直到循环终止。因此，Finvers 也可以有效地计算，这样当 F 作为从$\mathbf{Z}$ 到$\mathbf{Z}$ 的函数时，可以有效地计算其逆函数。

现在考虑模 n 时相同的映射。函数 F 还是可以有效地计算(见上或算法 A.26)，但是算法 Finvers 在模算术中就不可行了。其原因在于第三步的问题 $F(c)<y$ 就没有意义了。在环$\mathbf{Z}_n$中，就没有与算术操作相对应的排序。当上面的算法在模算术情况下使用时，我们所能做的最有效的方法就是逐个的检查$\mathbf{Z}_n$中的c，直到得到 $F(c)=y$。但是，这就成了一个指数时间的算法(n 是 n 的二进制长度$|n|$的幂)。

如果 n 的因子保密，迄今没有有效地计算 RSA 和 Square 函数的逆的算法，Exp 函数也一样，普遍认为没有有效的算法计算其原像。但是在过去一直没有人证明这一点。这些假设在第 6 章有详细的定义。它们是公钥密码学安全性证明的基础(第 9 章和第 10 章)。

3.3 RSA

RSA 密码系统基于 250 年来广为所知的初等数论知识。建立 RSA 密码系统，我们需要将两个大素数相乘，公开得到的积 n。n 作为公钥的一部分，而 n 的因子保密作为私钥。基本思想是 n 的因子不能从 n 求得。事实上，RSA 加密函数的安全性基于分解的困难性，但是这个等价关系还未得到证明。

下面我们详细地描述 RSA 工作的情况，讨论密钥生成、加密解密以及数字签名。

3.3.1 密钥生成与加密

密钥生成。RSA 密码系统的每一个用户 Alice 都有自己的公钥和私钥。密钥生成算法进行以下 3 个步骤(也可见于 6.4 节)：

1. 选择不同的大素数 p 和 q，计算 $n=p\cdot q$。
2. 选择 e 与 $\varphi(n)$互素，(n,e)公开作为公钥。
3. 通过 $ed\equiv 1 \bmod \varphi(n)$计算 d，(n,d)作为私钥。

注意这里 $\varphi(n)=(p-1)(q-1)$(推论 A.30)。数 n,e,d 分别指模数(modulus)，加密(encryption)和解密指数(decryption exponents)。为了解密一个密文或是为了生成一个数字签名，Alice 只需用到她的解密指数 d，而不需要知道素数 p 和 q。但是知道 p 和 q 将有利于她进行操作(加速解密，见下)。任何时候，Alice 都可以用有效算法，以很高的概

率，通过 n,e,d 推导出这个素数(见习题 4)。

攻击者应该对 Alice 的素数毫不知情。我们按以下的方式获取素数 p 和 q。首先，随机地选择一个大整数 x，如果 x 是偶数，将 $x+1$ 赋予 x 进行概率素性检测(见附录 A.70)，检验其是否为素数，如果 x 不是素数，就将 $x+2$ 赋予 x，继续检测，直到得到第一个素数。在我们得到第一个素数之前，可能进行 $O(\ln(x))$ 次素性检测(见推论 A.64)。以上所叙述的方法并不能肯定地产生素数(在此使用了一种概率得素性检测法)，但是实际上这已经足够了。现在，建议使用 512b 的素数。因为很难对分解算法以及计算机技术的发展做出准确的估计，所以现在没有人可以预测出多长的数才是安全的。

数字 e 也可以随机选择。利用欧几里得算法(算法A.4)检查 e 是否与 $\varphi(n)$ 互素。另一种获取 e 的方法是从 $\max(p,q)$ 和 $\varphi(n)$ 之间选取一个素数，这样即可保证 e 与 $\varphi(n)$ 互素。选择 p 和 q 时也可以用同样的方法。数 d 可以通过扩展欧几里得算法求得(算法 A.5)。

为了随机选择一个数，可以用一个伪随机序列发生器。这是一种生成看起来像随机数字的数字序列的算法。对于有效而安全地产生伪随机数已经有了大量的研究文献(参见[MenOorVan96]中例子)。我们也将在第 8 章中讨论这个课题。

加密与解密。我们对 $\{0,\cdots,n-1\}$ 范围内的消息进行加密，考虑其为 $\mathbf{Z}_n$ 中的一个元素。

1. 加密函数定义如下

$$\mathrm{E}:\mathbf{Z}_n\to\mathbf{Z}_n,\quad x\mapsto x^e$$

2. 类似地，解密函数定义如下

$$\mathrm{D}:\mathbf{Z}_n\to\mathbf{Z}_n,\quad x\mapsto x^d$$

E 和 D 都为双射且彼此互逆，即 $\mathrm{E}\circ\mathrm{D}=\mathrm{D}\circ\mathrm{E}=\mathrm{id}_{\mathbf{Z}_n}$(见命题 3.3)。加密和解密都可以使用一种更为有效的算法(见算法 A.26)。

按照基本的加密程序，可以对长度最大为 $k:=\lfloor\log_2(n)\rfloor$ 的比特序列进行加密。如果消息更长的话，可以将其分解为长度为 k 的块并使用 3.3.4 节描述的方案，或者是别的合适的操作模式。比如，电码本模式或密码分组连接模式(见 2.2.2 节)，这里密码反馈模式和输出反馈模式不能直接使用。显然，如果初始信息没有保密，那么任何人都可以解密密文，第 9 章有一个对 RSA 使用输出反馈的例子。

安全。如果一个敌手知道 n 的因子 p 和 q，那么他必然也知道 $\varphi(n)=(p-1)(q-1)$，利用公开的加密密钥 e，使用扩展欧几里得算法(算法 A.5)即可求出解密密钥 d。因此，如果 p 和 q 是大素数，RSA 的安全性依赖于求解 n 的因子 p 和 q 的困难性。一般我们认为，如果 p 和 q 足够大的话，现在还不存在分解 n 的有效算法。这种说法称为分解

假设(详细定义,见定义 6.9)。也就是说有效的分解算法即可攻破 RSA 体制。解密 RSA 的密文是否必须要分解 n 还未曾得到证明,但是人们也普遍认为对 RSA 函数求逆是不可行的。这种说法称为 RSA 假设(见定义 6.7)。

在选择 RSA 的密钥时,计算指数 e 和 d 唯一所需要输入的就是 $\varphi(n)$ 和 n。因为 $\varphi(n)=(p-1)(q-1)$,有

$$p+q=n-\varphi(n)+1 \text{ 和 } p-q=\sqrt{(p+q)^2-4n} \quad (\text{如果 } p>q)$$

由此,如果已知 $\varphi(n)$,分解 n 就十分容易。

n 的分解可以归约为一种算法 A,通过 n 和 e 计算出 d(见习题 4)。这样得来的算法 A'是一种概率算法,与算法 A 有同样的复杂度(这个事实在参考文献 [RivShaAdl78]中已经提到过了)。

现在,能否从有效的求 RSA 逆函数的算法推导出有效的分解算法,即在输入 n,e 和 x^e 的情况下,输出为 x 的有效算法,还是个公开的难题。

解密指数 d 应该大于 $n^{\frac{1}{4}}$。对于 $d<n^{\frac{1}{4}}$,存在一种多项式时间的算法求 d(参见 [Wiener90][①])。

对于特殊类型的素数 p 和 q,存在有效的分解算法。为了使这种算法无可乘之机,应避免使用这样的素数。首先,要求绝对值 $|p-q|$ 要大。这样可以防止以下的攻击:我们有 $\frac{(p+q)^2}{4}-n=\frac{(p+q)^2}{4}-pq=\frac{(p-q)^2}{4}$,如果 $|p-q|$ 的值小,那么 $\frac{(p-q)^2}{4}$ 必然也小,这样的话,$\frac{(p+q)^2}{4}$ 就只比 n 大一点,那么 $\frac{p+q}{2}$ 只比 $\sqrt{n}$ 大一点,从而以下分解方法有效:

1. 选择一系列的 $x>\sqrt{n}$,检验 x^2-n 是否为平方数。

2. 在这种情况下,有 $x^2-n=y^2$,得 $x^2-y^2=(x-y)(x+y)=n$,这样就得到了 n 的一个分解。

这种分解方法可以归结为费马的想法。

为了防止对 RSA 密码系统的其他攻击,定义了一种强素数(strong prime)概念。一个素数称为强素数必须满足以下条件:

1. $p-1$ 有一个大素数因子,记为 r。

2. $p+1$ 有一个大素数因子。

3. $r-1$ 有一个大素数因子。

"大"的含义可以从攻击的角度来看(见后)。强素数可以通过 Gordon 算法获得(参见[Gordon84])。如果与概率素性检测方法联合起来使用,Gordon 算法只比前面提到的

① 这个算法使用了 e/n 的连分式展开式。