第 3 章　公钥密码

在古典密码时代，人们对密码算法的初始实现主要是通过手工计算完成的。随着轮转加密和解密机器的出现，传统密码学有了很大进展，通过电子机械转轮可以开发出很复杂的加密系统。轮转密码机（Enigma）和 DES 是传统密码学发展的重要成果，但是它们仍然基于替换和置换的初等方法之上。在公钥密码（Public Key Cryptography）提出以前，我们了解到的密码都基于替换和置换这些初等方法。

公钥密码的发展是整个密码学发展历史上最伟大的一次革命，也可以说是唯一的一次革命。

3.1　对称密码的密钥交换问题

在对称密码模型中，一个密码系统由 5 个要素组成，包括明文、密文、加密算法、解密算法和通信双方共享的秘密——密钥，对称密码模型解决的主要是保密通信的问题。

图 3.1 所示为对称密码模型，发送方 Alice 要发一段明文给接收方 Bob，Alice 用对称密钥进行加密，Bob 用同样的对称密钥解密。对称密码又称单密钥密码、秘密密钥密码、传统密码。其特点是：发送方加密和接收方解密使用同一个密钥，该密钥需要事先由发送方和接收方实现共享，是发送方和接收方共同的秘密。如果密钥泄露，则不安全，无法提供保密通信的服务。对称的含义是指通信双方的地位是对等的，或者说通信双方的“秘密”是相同的。

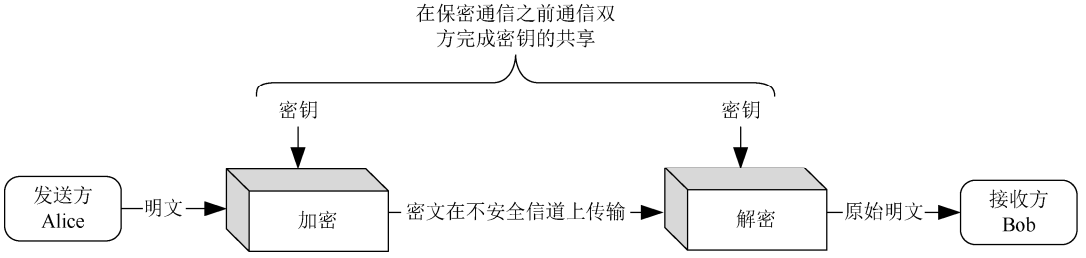


图 3.1　对称密码模型

对称密码模型要求在任何通信开始之前，双方需要事先共享密钥，并且密钥不能通过不安全信道进行传输，即没有办法通过传输密文信道进行传输。在历史上，人们创建使馆或情报机构，为了交换密钥，把密钥放在特殊的“信封”里进行传输。而传输密文信道采用电子的、无线电波的形式，这样就造成了一个很大的问题：通信开始之前需要进行密钥的协商，协商完成之后加密和解密是很容易的，但密钥交换本身成了最大的问题。

3.2　公钥密码模型的提出

对称密码模型要求在任何通信开始之前，发送方和接收方需要完成密钥的共享，这是个难以实现的问题。为了解决这个问题，提出了公钥密码模型。

3.2.1　公钥密码模型

在公钥密码模型里，发送方和接收方各有两个密钥：公钥和私钥。公钥可以公开，可以放在公告牌或个人主页上，任何人都可以通过这个公钥和公钥所有者进行通信。

图 3.2 所示为公钥密码模型，为了实现保密通信，发送方 Alice 通过接收方 Bob 的公钥加密信息，然后发送给 Bob，Bob 再用自己的私钥解密，这样就可以省去交换密钥的环节。

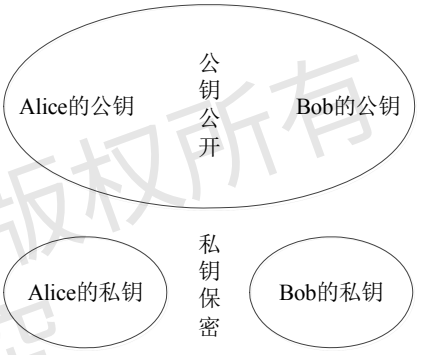


图 3.2　公钥密码模型

在公钥密码模型里，公钥和私钥是成对出现的，参与通信的每个人都有两个密钥，即公钥和私钥。公钥是完全公开的，理论上来说通信系统中的任何人都可以获取到其他用户的公钥，而每个用户的私钥仅用户自己知道，其他任何人都不知道。

有了这样一对密钥，如何实现保密通信呢？参考图 3.3 的示例，Bob 想要和 Alice 通信，那么他会使用 Alice 的公钥对明文进行加密。加密后的明文通过不安全信道传输给 Alice，Alice 收到密文后，用自己的私钥通过解密算法将密文还原成明文。由此可以得出以下结论。

（1）Alice 的公钥和私钥是成对出现的，加密的时候使用 Alice 的公钥，解密的时候使用 Alice 的私钥。

（2）即使攻击者知道了 Alice 的公钥，以及加密算法和解密算法，也不知道且不能推导出 Alice 的私钥。

（3）假设攻击者获取到了密文和 Alice 的公钥，而且知道了加密算法和解密算法，攻击者同样不能还原出明文。

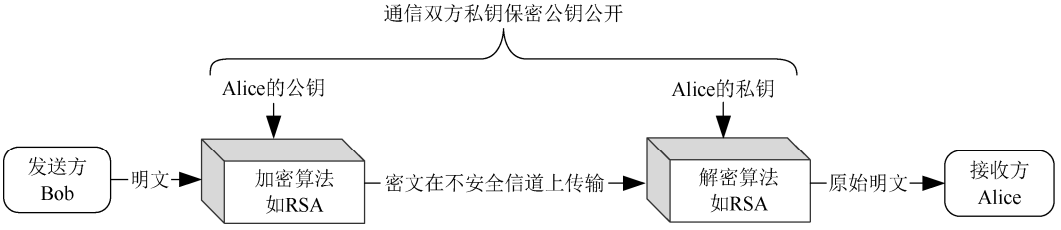


图 3.3　提供保密通信的用法

通过上面的分析，我们可以得出结论：公钥密码模型很好地解决了对称密码模型交换密钥的问题。在通信开始之前，只要发送方获取到接收方的公钥（例如，从公告牌或接收方的个人主页中获取），就可以和接收方进行保密通信。

3.2.2 密码学的两种密码

（1）对称的、单密钥、秘密密钥、传统密码技术：发送方和接收方使用同一个密钥，主要解决保密通信的问题，通信双方所知道的内容是一样的。

（2）非对称的、双密钥、公钥密码技术：发送方和接收方使用不同的密钥，加密密钥和解密密钥分开，且无法由一个推出另一个，不仅可以公开密码算法，而且可以公开加密密钥（公告牌、号码簿）。

人类文明早期几乎都是基于对称密码模型的密码，这些对称密码是为了实现保密通信的功能。随着公钥密码模型的提出，密码技术进入了一个新的篇章。

公钥密码模型是针对现实应用需求而提出的，互联网的高速发展使人们逐渐产生了数字签名、报文鉴别等方面的新的安全需求。例如，在互联网通信场景中，你可能要和从未谋面的人进行保密通信，而公钥密码可以很好地满足这样的需求。每个参与公钥密码通信的用户都有一对密钥。

考虑这一场景，假设有 n 个用户，要实现其中任意两个用户之间的保密通信。如果使用对称密码，由于任意两个用户之间通信都要使用不同的密钥，总共需要用到 $n(n-1)/2$ 个不同的对称密钥；如果这 n 个用户采用公钥密码通信，则只需给每个用户设置一对密钥，总共需要 $2n$ 个密钥。

公钥密码的密钥往往是应用数论的一些函数精心构造的，由于公钥密码的加密和解密速度慢，因此公钥密码模型只是补充而非取代对称密码模型。

3.2.3 公钥密码的历史

1976 年，Diffie 和 Hellman 在论文 *New Direction in Cryptography* 中首次提出了公钥密码的思想；Diffie 和 Hellman 提出了第 1 个基于公钥密码思想的算法，称为 DH 算法，此算法仅可以用于实现密钥交换。

1977 年，Rivest、Shamir 和 Adleman 实现了公钥密码，现在称为 RSA 算法，它是第一个既能用于密钥交换，又能用于数据加密和数字签名的算法。

3.3 设计公钥密码的基本要求

一个公钥密码模型由 6 个要素组成：明文、公钥（记作 PU 或 KU）、私钥（记作 PR 或 KR）、加密算法、密文、解密算法。公钥算法的基本要求如下。

（1）参与方 B 容易通过计算产生一对密钥（公钥 PU_B、私钥 PR_B）。

（2）参与方 A 很容易通过计算产生密文 $\left[C = E_{PU_B}(M)\right]$。

（3）参与方 B 通过计算解密密文$\left[M = D_{\mathrm{PR_B}}\left(C\right) = D_{\mathrm{PR_B}}\left(E_{\mathrm{PU_B}}\left(M\right)\right)\right]$。

（4）攻击者即使知道公钥$\mathrm{PU_B}$，要确定私钥$\mathrm{PR_B}$在计算上是不可行的。

（5）攻击者即使知道公钥$\mathrm{PU_B}$和密文 C，要确定明文 M 在计算上是不可行的。

（6）密钥对之间可以交换使用$\left[M = D_{\mathrm{PR_B}}\left(E_{\mathrm{PU_B}}\left(M\right)\right) = D_{\mathrm{PU_B}}\left(E_{\mathrm{PR_B}}\left(M\right)\right)\right]$。

总体来说，公钥算法建立在计算复杂性理论基础之上。其基本思想是让使用公钥密码的发送方和接收方各自只需多项式时间算法即可完成明文的加密和解密过程，然而对攻击者而言却是一个未能找到多项式时间算法的 NP 完全问题。

图 3.4 展示了不同类型算法的时间复杂度，随着求解目标问题规模（n 值）的增大，多项式时间算法的时间复杂度按多项式时间复杂度增长，然而对于 NP 完全问题的时间复杂度按指数型增长。换言之，公钥密码就是通过精心构造，让密码算法的使用者自己求解一个多项式时间的问题，而让攻击者求解一个难题（NP 完全问题），二者在计算成本上形成巨大的差距。

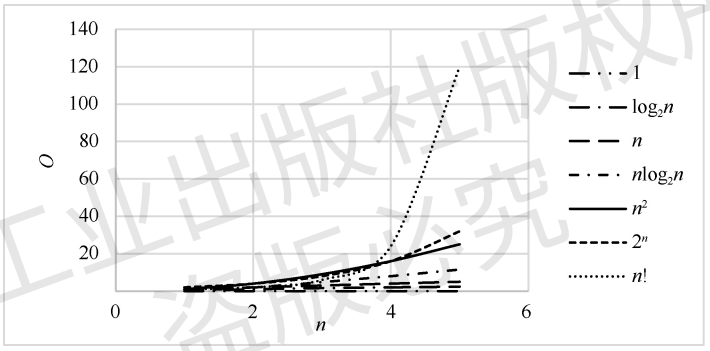


图 3.4　不同类型算法的时间复杂度

3.4　数字签名

在前面的叙述中我们了解到，密钥对之间是可以交换使用的。在一对密钥中，用公钥加密的报文可以由私钥解密；用私钥加密的报文可以由公钥解密。公钥和私钥是成对出现的，这样一种交换使用的方式可以提供很多功能，数字签名（又称公钥数字签名）就是其中一种。

广义的数字签名包含两种安全需求：身份验证和抗抵赖。

数字签名是只有信息的发送方才能产生而别人无法伪造的一段数字串，这段数字串是对信息的发送方发送信息的真实性和身份合法性的一个有效证明；也可以作为“防止发送方事后抵赖其发送过这个信息”的一个证据。它是一种类似写在纸上的普通的物理签名，但是使用了公钥加密领域的技术来实现，用于鉴别数字信息的方法。一套数字签名通常定义两种互补运算，一个用于签名（使用私钥），另一个用于验证签名（使用公钥）。

图 3.5 所示为数字签名的工作流程，发送方 Bob 用自己的私钥加密想要发送的明文，加密后的密文在不安全信道上传输，接收方 Alice 用 Bob 的公钥进行解密。

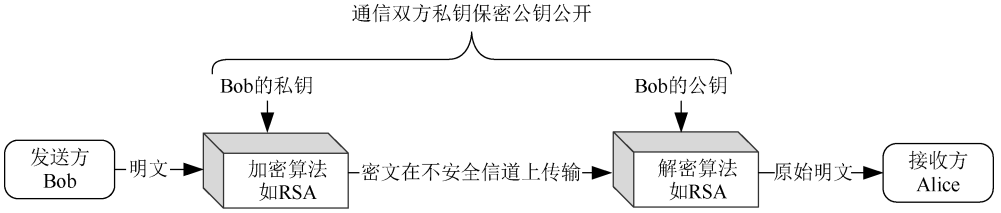


图 3.5　数字签名的工作流程

（1）身份验证：解密的过程也就是身份验证的过程，如果说 Alice 用 Bob 的公钥解开了密文，那么可以说明这段密文是 Bob 发送过来的，因为 Bob 的私钥只有 Bob 知道。

（2）抗抵赖：假设 Bob 加密的明文是“Bob 欠 Alice 一万块钱”，不久之后 Bob 想抵赖没有发过。这时候只要 Alice 保存 Bob 发送过来的密文，作为证据提供给第三方，并且用 Bob 的公钥解密还原为 Bob 的原始明文，就可以证明这段明文是 Bob 发出的。因为只有 Bob 自己才知道自己的私钥，而每个人的私钥只会对应自己的公钥。

前面章节主要讨论对称密码模型，主要关注的是用密码实现保密通信的安全服务。随着公钥密码模型的提出，我们将面临更多的安全需求。例如，公钥加密和私钥解密可以实现保密通信和密钥交换，私钥加密和公钥解密可以实现数字签名。不同公钥密码算法实现的功能如表 3.1 所示。

表 3.1　不同公钥密码算法实现的功能

功能	RSA 算法	DH 算法	DSA 算法
保密通信	能	不能	不能
密钥交换	能	能	不能
数字签名	能	不能	能

3.5　RSA 算法

RSA 算法在 1977 年由 MIT（麻省理工学院）的 Ron Rivest、Adi Shamir 和 Len Adleman 提出，是最早的既可实现加密，又可用于数字签名的公钥密码算法。RSA 算法是目前被广泛接受且被实现的通用公钥加密算法。

RSA 算法的理论基础是数论中的下述论断：求两个大质数的乘积是容易的，但分解一个合数为两个大质数的乘积，在计算上几乎是不可能的。

3.5.1　数论知识

假设 a 和 b 均为整数，m 为正整数。若 m 整除 $b-a$，则可将其表示为 $a\equiv b(\mathrm{mod}\ m)$。式 $a\equiv b(\mathrm{mod}\ m)$读作“$a$ 与 b 模 m 同余”，m 称为模数。

定义模 m 上的算术运算：令 Z_m 表示集合$\{0,1,\cdots,m-1\}$，在其上定义两个运算，即加法和乘法，类似普通实数域上的加法和乘法，不同的是所得的值是取模以后的余数，在 Z_m 上计算 $ab=ab(\mathrm{mod}\ n)$。设 $a\in Z_m$，对于任意的 $b\in Z_m$，同余方程 $ax\equiv b(\mathrm{mod}\ m)$有唯一解，$x\in Z_m$

的充要条件是 gcd(*a*,*m*)=1。

设 $a \geqslant 1$，$m \geqslant 2$，且均为整数。如果 gcd(*a*,*m*)=1，则称 *a* 与 *m* 互质。Z_m 中所有与 *m* 互质的数的个数用 $\phi(m)$ 表示（这个函数称为欧拉函数）。假定 $m=\prod_{i=1}^{n} p_i^{e_i}$，$p_i$ 均为质数且互不相同，$e_i>0\left(1 \leqslant i \leqslant n\right)$，则

$$\phi(n)=\prod_{i=1}^{n}\left(p_i^{e_i}-p_i^{e_i-1}\right) \tag{3.1}$$

模 *n* 的余数与 *n* 互质的全体记为 Z_n^*。可以看到 Z_n^* 在乘法运算下形成一个阿贝尔群。模 *n* 的乘法运算是可结合和可交换的，1 为乘法单位元。Z_n^* 中的任一元素都有一个乘法逆元（也在 Z_n^* 中）。最后，Z_n^* 是乘法封闭的，这是因为当 *x* 和 *y* 都与 *n* 互质时，*xy* 与 *n* 互质。

现在我们知道任意的 $b \in Z_n^*$ 都有一个乘法逆 *b*–1，可以采用扩展 Euclidean 算法（扩展的欧几里得算法）。

3.5.2　算法原理

任何一个密码都是满足以下条件的五元组(*P*,*C*,*K*,*ε*,*D*)。

（1）*P* 表示由所有可能的明文组成的有限集。

（2）*C* 表示由所有可能的密文组成的有限集。

（3）*K* 代表密钥空间，由所有可能的密钥组成的有限集。

（4）对于每个 $K \in \mathcal{K}$，都存在一个加密规则 $e_K \in \varepsilon$ 和相应的解密规则 $d_K \in D$。并且对于每对 $e_K: P \to C$，$d_K: C \to P$，都满足条件：对于每个明文 $x \in P$，均有 $d_K\left(e_K(x)\right)=x$。

对于 RSA 算法：利用 Z_n 中的计算，其中 *n* 是两个不同的奇质数 *p* 和 *q* 的乘积。对于这样一个整数 *n*，定义 $\phi(n)=(p-1)(q-1)$。设 $P=C=Z_n$，且定义

$$K=\left\{(n,p,q,a,b): ab \equiv 1\left(\bmod \phi(n)\right)\right\} \tag{3.2}$$

对于 $K=(n,p,q,a,b)$ 定义 $e_K(x)=x^b \pmod n$ 和 $d_K(y)=y^a \pmod n\left(x,y \in Z_n\right)$。*n* 和 *b* 组成了公钥，*p*、*q* 和 *a* 组成了私钥。

验证加密和解密是逆运算：由于 $ab \equiv 1\left(\bmod \phi(n)\right)$，$ab=t\phi(n)+1$ 对于某个正整数 $t \geqslant 1$。假定 $x \in Z_n$，那么

$$\begin{aligned}\left(x^b\right)^a &\equiv x^{t\phi(n)+1} \pmod n \\ &\equiv \left(x^{\phi(n)}\right)^t x \pmod n \\ &\equiv 1^t x \pmod n \\ &\equiv x \pmod n\end{aligned} \tag{3.3}$$

整数 *b* 可以选为加密指数，当且仅当 *b* 不能被 2、5、7 整除时。

3.5.3 RSA 密钥生成

执行以下程序。

（1）生成两个大质数 p 和 q（p 和 q 要求至少是十进制百位级别的数）。

（2）计算两个质数的乘积 $n=pq$。

（3）计算小于 n 并且与 n 互质的整数的个数，即欧拉函数 $\phi(n)=(p-1)(q-1)$。

（4）选择一个随机数 b 满足 $1<b<\phi(n)$，并且 b 和 $\phi(n)$ 互质，即 $\gcd(b,\phi(n))=1$。

（5）$ba\left(\bmod \phi(n)\right)=1$，求出 a。

（6）保密 a，销毁 p 和 q，公开 n 和 b。

公钥公开：PU=$\{b,n\}$

私钥保密：PR=$\{a,n\}$

对密钥生成程序的说明。

（1）得到足够大质数 p 和 q 的方法：在实际应用中，首先产生一个足够大的随机数，然后通过一个概率多项式时间算法检测该随机数是否为质数。常用的两个素性测试算法：Solovay-Strassen 素性测试、Miller-Rabin 素性测试。

（2）上述步骤（4）中的 b 可以在质数中选择，用 $\phi(n)$ 除以 b。若除不尽，则认为 b 和 $\phi(n)$ 互质。

（3）在上述步骤（5）求解方程的时候，计算 a 可以使用辗转相除法（扩展的欧几里得算法）。

具体过程举例如下。

假设 $b=1001$，$\phi(n)=3837$，方程为 $x\times1001=1\left(\bmod 3837\right)$。

求解过程：

$$3837=3\times1001+834$$
$$1001=1\times834+167$$
$$834=4\times167+166$$
$$167=166+1$$

由上面的推导可以得到

$$\begin{aligned}1&=167-166\\&=167-(834-4\times167)\\&=5\times167-834\\&=5\times(1001-834)-834\\&=5\times1001-6\times834\\&=5\times1001-6\times(3837-3\times1001)\\&=23\times1001-6\times3837\end{aligned}$$

求得 a=23。

由上面的分析可以得到，在上述密钥生成的过程中，每步都是有对应算法的，而且算法都是多项式时间算法，即密钥生成只需多项式数量级的时间复杂度。

3.5.4　RSA 算法的加密和解密过程

1．RSA 算法的加密过程

为了加密一个报文 M，发送方需要完成以下操作。

（1）获取接收方的公钥 $\mathrm{PU}=\{b,n\}$。

（2）计算 $C=M^b \pmod n$，$0 \leqslant M < n$。

（3）把 C 作为密文发送给接收方。

加密算法的说明：步骤（2）中要求 $M<n$，当 $M>n$ 的时候，可以将 M 分组，其中每个分组都视为一个小于 n 的正整数。

2．RSA 算法的解密过程

为了解密一个密文 C，接收方需要完成以下操作。

（1）使用自己的私钥 $\mathrm{PR}=\{a,n\}$。

（2）计算 $M=C^a \pmod n$。

（3）计算得到的 M 为发送方想发送的明文。

RSA 的加密算法和解密算法是一致的，都是先计算一个数的多次方，然后 mod n 求余数，这意味着我们在针对加密和解密过程设计硬件和软件的时候，只需设计一套。

3．RSA 算法的验证实例

密钥生成如下。

挑选质数：$p=17$，$q=11$。

计算 $n=pq=17\times11=187$。

计算 $\phi(n)=(p-1)(q-1)=16\times10=160$。

选择 b：$\gcd(b,160)=1$；不妨选 $b=7$。

求解 a：$ba=1 \pmod{160}$ 且 $a<160$，$a=23$，显然 $23\times7=161=1\times160+1$。

$\mathrm{PU}=\{7,187\}$，$\mathrm{PR}=\{23,187\}$。

加密和解密过程验证如下。

选择 $M=88$（注意：$88<187$）。

加密：$C=88^7 \pmod{187}=11$。

解密：$M=11^{23} \pmod{187}=88$。

3.5.5　模指数算法

在 RSA 算法的加密和解密过程中，$C=M^b \pmod n$ 为模指数运算。可以通过 $b-1$ 次模乘实现计算，然而，如果 b 非常大，则其效率会很低下，平方-乘算法可以把计算所需的模乘的次数降低。

求模指数实例如下。

$$11^{23} \pmod{187}=\left[11^1 \pmod{187}\times11^2 \pmod{187}\times11^4 \pmod{187}\times11^8 \pmod{187}\times11^8 \pmod{187}\right]$$

$(\mathrm{mod}\ 187)$；

$11^1 (\mathrm{mod}\ 187) = 11$；

$11^2 (\mathrm{mod}\ 187) = 121$；

$11^4 (\mathrm{mod}\ 187) = 14641 (\mathrm{mod}\ 187) = 55$；

$11^8 (\mathrm{mod}\ 187) = 214358881 (\mathrm{mod}\ 187) = 33$；

$11^{23} (\mathrm{mod}\ 187) = 11 \times 121 \times 55 \times 33 \times 33 (\mathrm{mod}\ 187) = 79720245 (\mathrm{mod}\ 187) = 88$。

模指数算法如下。

假设 x, y, N 为 3 个正整数（y 可能为 0），计算 $x^y (\mathrm{mod}\ N)$。

$$x^y = \begin{cases} \left(x^{\lfloor y/2 \rfloor}\right)^2, & y\text{为偶数} \\ x\left(x^{\lfloor y/2 \rfloor}\right)^2, & y\text{为奇数} \end{cases}$$

当 $y = 0$ 时，$x^y (\mathrm{mod}\ N) = 1$；当 $y! = 0$ 时，设 $z = x^{\lfloor y/2 \rfloor} (\mathrm{mod}\ N)$，分为以下两种情况。

当 y 为偶数时，有

$$x^y (\mathrm{mod}\ N) = \left(x^{\lfloor y/2 \rfloor}\right)^2 (\mathrm{mod}\ N) = z^2 (\mathrm{mod}\ N)$$

当 y 为奇数时，有

$$x^y (\mathrm{mod}\ N) = x\left(x^{\lfloor y/2 \rfloor}\right)^2 (\mathrm{mod}\ N) = xz^2 (\mathrm{mod}\ N)$$

由上面的讨论我们可以得到以下递归算法。

function $\mathrm{modexp}(x, y, N)$，输入 x, y, N，输出 $x^y (\mathrm{mod}\ N)$

```
if y == 0 : return 1
z = modexp(x, ⌊y / 2⌋, N)
if y is even :
    return z^2 (mod n)
else:
    return x · z^2 (mod n)
```

RSA 算法的注意事项：RSA 算法加密时，明文以分组的方式加密，每个分组的比特数都应该小于 $\log_2 n$ 比特，即 $M < n$；选取的质数 p 和 q 要足够大，从而乘积 n 足够大，在事先不知道 p 和 q 的情况下，分解 n 在计算上是不可行的。

3.5.6 RSA 算法的可逆性证明

要证明 $D\left(E\left(M\right)\right) = M$（$E$ 表示加密 Encode，D 表示解密 Decode），即证明

$$M = C^a (\mathrm{mod}\ n) = \left(M^b\right)^a (\mathrm{mod}\ n) = M^{ba} (\mathrm{mod}\ n)$$

因为 $ba = 1 \left(\mathrm{mod} \phi(n)\right)$，说明 $ba = t\phi(n) + 1$，其中 t 为某整数。所以

$$M^{ba} (\mathrm{mod}\ n) = M^{t\phi(n)+1} (\mathrm{mod}\ n)$$

因此，要证明 $M^{ba}\pmod n = M\pmod n$，只需证明

$$M^{t\phi(n)+1}\pmod n = M\pmod n$$

在 $(M,n)=1$ 的情况下，根据数论（Euler 定理）

$$M^{t\phi(n)}\pmod n = 1\pmod n$$

有

$$M^{t\phi(n)+1}\pmod n = M\pmod n$$

在 $(M,n)\neq 1$ 的情况下，分为以下两种情况。

第 1 种情况：$M\in\{1,2,3,\cdots,n-1\}$。

因为 $n=pq$，p 和 q 为质数，$M\in\{1,2,3,\cdots,n-1\}$，且 $(M,n)\neq 1$，说明 M 必包含 p 或 q 之一作为因子，而且不能同时包含二者，否则 $M\geqslant n$，与 $M\in\{1,2,3,\cdots,n-1\}$ 矛盾。不妨设 $M=kp$，又因为 q 为质数，且 M 不包含 q，故有 $(M,q)=1$，于是

$$M^{\phi(q)}\pmod q = 1\pmod q$$

进一步有

$$M^{t(p-1)\phi(q)}\pmod q = 1\pmod q$$

因为 q 是质数，$\phi(q)=q-1$，$t(p-1)\phi(q)=t\phi(n)$，所以

$$M^{t\phi(n)}\pmod q = 1\pmod q$$

于是，$M^{t\phi(n)}=lq+1$，其中 l 为整数。

两边同乘 M 得

$$M^{t\phi(n)+1}=lqM+M$$

因为 $M=kp$，故

$$M^{t\phi(n)+1}=lqkp+M=lkn+M$$

取模 n 得

$$M^{t\phi(n)+1}\pmod n = M\pmod n$$

第 2 种情况：当 M=0 时，直接验证。

3.5.7　RSA 算法的安全性

在理论上，RSA 算法的安全性取决于模 n 分解的困难性。若 n 被分解成功，则 RSA 算法被攻破。但并没有证明大数分解就是 NP 问题，并不能排除存在尚未发现的多项式时间算法。也没有证明对 RSA 算法的攻击的难度与模 n 分解等价，因此对 RSA 算法的攻击的难度不比大数分解难。

目前，RSA 算法的一些变种已被证明等价于大数分解。不管怎样，模 n 分解仍是最显然的攻击方法。现在人们已能分解多个十进制的大质数，因此，模 n 必须选得大一些。

由于都是大数计算，RSA 算法最快的情况也比 DES 算法慢许多，无论是软件还是硬件实现，速度一直都是 RSA 算法的缺陷，一般只用于少量数据加密。RSA 算法是被研究得最

广泛的公钥密码算法，从提出到现在已有 40 多年，经历了各种攻击的考验，逐渐为人们所接受，普遍认为是目前最优秀的公钥密码方案之一。

1．对 RSA 算法的攻击：分解因子算法

攻击 RSA 算法的最明显方式就是试图分解公开模数，下面讨论当今比较好的分解因子算法，以及在实际中的应用。对于大整数，最有效的 3 种算法是二次筛法（Quadratic Sieve）、椭圆曲线分解算法（Elliptic Curve Factorization Algorithm）和数域筛法（Number Field Sieve）。其他作为先驱的著名算法包括 Pollard-Rho 算法和 p−1 算法，William 的 p+1 算法、连分式算法（Continued Fraction）和试除法（Trial Division）。

假定要分解的整数 n 为奇数，如果 n 是合数，则容易看出 n 有一个质因子 $p \leqslant \lfloor \sqrt{n} \rfloor$。因此作为最简单的试除法，就是用直到 $\lfloor \sqrt{n} \rfloor$ 的每个奇数去除 n，这足以判断 n 是质数还是合数。如果 $n < 10^{12}$，那么它还是不错的算法，但对于非常大的 n，我们还需要更多复杂的技巧。

当我们要分解 n 时，完全分解为质因子之积或找到非平凡因子（Non-Trivial Factor）即可。

2．Pollard p−1 算法

两个输入：要分解的奇整数 n 和一个预先指定的界 B。

```
Pollard p-1 Factoring Algorithm (n,B):
a = 2
for(j=2 to B)
do a=a^j(mod n)
d = gcd(a-1,n)
if 1<d<n
then return(d)
else return("failure")
```

在 Pollard p−1 算法中，假定 p 是 n 的一个质因子，又假定对于每个质数幂 $q \mid (p-1)$，都有 $q \leqslant B$。在这种情形下必有

$$(p-1) \mid B!$$

在 for 循环结束时，有

$$a \equiv 2^{B!} \pmod{n}$$

由于 $p \mid n$，一定有

$$a \equiv 2^{B!} \pmod{p}$$

由费马（Fermat）引理可知

$$2^{p-1} \equiv 1 \pmod{p}$$

由于 $(p-1) \mid B!$，于是

$$a \equiv 1 \pmod{p}$$

因此有 $p \mid (a-1)$。由于我们已经有了 $p \mid n$，因此可以看到 $p \mid d$，其中 $d = \gcd(a-1, n)$。整数 d 就是 n 的一个非平凡因子（除非 $a = 1$）。一旦找到 n 的一个非平凡因子 d，就可以对 d

和 n/d 继续分解（如果 d 和 n/d 还是合数）。

另外，还有 Pollard-Rho 算法和 Dixon 的随机平方算法。

3．实际中的分解因子算法

一个著名的并在实际中广泛应用的分解因子算法是 Pomerance 提出的二次筛法，二次筛法源于判定 $z^2 \pmod n$ 在 β 上分解的一个筛选过程。数域筛法是 20 世纪 80 年代末发展起来的算法，通过构造同余方程 $x^2 \equiv y^2 \pmod n$ 分解 n，是在代数整数环中进行计算的。

分解因子算法的时间复杂度如表 3.2 所示。

表 3.2　分解因子算法的时间复杂度

二次筛法	$O\left(\mathrm{e}^{(1+o(1))\sqrt{\ln n \ln(\ln n)}}\right)$
椭圆曲线分解算法	$O\left(\mathrm{e}^{(1+o(1))\sqrt{2\ln p \ln(\ln p)}}\right)$
数域筛法	$O\left(\mathrm{e}^{(1.92+o(1))(\ln n)^{\frac{1}{3}}(\ln(\ln n))^{\frac{2}{3}}}\right)$

$o(1)$ 表示当 $n \to \infty$ 时趋向于 0 的函数，p 表示 n 的最小质因子。在最坏的情况下，$p \approx \sqrt{n}$，二次筛法和椭圆曲线分解算法的渐进运行时间本质上是一样的。但在这种情况下，二次筛法一般快于椭圆曲线分解算法。如果 n 的素因子具有不同的长度，那么椭圆曲线分解算法是更有效的。

直到 20 世纪 90 年代，二次筛法都是分解 RSA 模（RSA 模指 $n = pq$，p 和 q 是两个不同的质数，p 和 q 的长度大致相等）的最常用算法。数域筛法是 3 种算法中最近发展起来的算法，该算法和其他算法相比的优点是，渐进时间复杂度（在问题规模趋于无穷大时算法时间复杂度的数量级）低。已证明数域筛法对于大于 125～130 比特的十进制数是较快的算法。

3.5.8　对 RSA 算法的其他攻击

1．计算 $\phi(n)$

首先看到计算 $\phi(n)$ 不比分解 n 容易。假设 n 和 $\phi(n)$ 已知，n 为两个质数 p 和 q 的乘积，那么可以容易地分解，通过求解以下两个方程：

$$n = pq$$
$$\phi(n) = (p-1)(q-1) \tag{3.4}$$

得到两个未知数 p 和 q。如果将 $q = n/p$ 代入式（3.4），我们可以得到一个关于未知数 p 的二次方程：

$$p^2 - (n - \phi(n) + 1)p + n = 0 \tag{3.5}$$

这个方程的两个根就是 p 和 q，即 n 的因子。因此，如果一个密码分析者能够求出 $\phi(n)$，他就能分解 n，进而攻破系统。也就是说，计算 $\phi(n)$ 不比分解 n 容易。

2．小指数攻击法

RSA 算法主要基于软件实现，其加密速度远不如 DES 算法。所以一些使用 RSA 算法进

行加密的机构采用了一种提升 RSA 算法速度并且能使加密易于实现的解决方案——令公钥 b 取较小的值，这样做会使该算法的安全强度降低。从理论上曾有人证明过，若采用不同的模 n 与相同的 b，则对 $b\times(b+1)/2$ 个线性相关的信息存在一种攻破方法。若私钥 a 为 n 的四分之一且 $b<n$，则使用该方法能解密。

应对措施：b 和 a 都应取较大的值，并且使用独立填充信息（随机数填充），使每个信息的大小都与 n 相同。这种做法保证了密钥长度在 2048 比特以上，会给推导私钥 a 的过程增加很大的难度。

3．群体暴力破解法

群体暴力破解法是一种猜测攻击。尽管从 RSA 算法的内在特性分析可知 RSA 算法对硬件要求较高，但群体暴力破解法在理论上和实践中仍是可行的。在数论中，一个不小于 6 的偶数总可以分解为两个质数之和的形式。对于某一个偶数，若分解为两个奇数之和，则有 $n/2$ 个分解式。自然数中的质数总数大约占奇数总数的 1/3，当 n 较大时，很有可能在这 $n/2$ 个分解式中存在某个分解式，使得该分解式分解出来的两个奇数全是质数。

应对措施：比较有效的措施是使用 IDS（入侵检测系统）发现攻击者的暴力破解行为，通过安全策略的制定限制攻击者的暴力破解行为——最简单的设定就是限制某账号几次登录不成功后进行账号锁定且不解锁。

4．选择密文攻击法

选择密文攻击法是一种绕开 RSA 算法，直接攻击协议的攻击方式。算法安全性和协议安全性是两个基本安全组件，二者组合到一起之后的安全性究竟是否增加的界限并不是太好判定，选择密文攻击就是一个安全性没有增加的实例。在通信过程中，贸然签名的一方容易被攻击者进行选择密文攻击。攻击者只需将某信息进行伪装（Blind），让拥有私钥的实体签名。

应对措施如下。

（1）采用好的公钥协议，保证工作过程中实体不对其他实体产生的任意信息进行解密，不对自己未见过的信息签名。

（2）不对陌生人的随机文档签名，签名时首先使用单向函数（One-Way Hash Function）对文档进行哈希处理，同时使用不同的签名算法。

（3）适当加强信息的冗余度，这样可以增加协议的安全性。

5．公共模数攻击法

如果网络中使用同样的模 n，则容易被攻击者进行公共模数攻击。尽管公钥不同，但采用同一个模 n，最明显的问题是如果同一报文用两个不同的、互质的密钥加密，则无须解密就可恢复明文。

应对措施：不要让所有人都使用同一个模 n，这样使得只有通过推导密钥对才能恢复明文，对攻击者而言几乎是不可能完成的任务。

6．计时攻击法

计时攻击是一种测信道攻击，在 RSA 解密或签名时通过统计的方法获得私钥 d，这种方法通常用于有较弱计算能力的设备，如智能卡。设私钥 d 的二进制表示为 $d_0d_1\cdots d_n$，$d_0=1$，

在模指数运算普遍使用的平方-乘算法中，当 d 的位为 1 时，整个运算过程多了一个乘法操作，即先平方操作后乘法操作；而当 d 的位为 0 时，仅有平方操作。又因为在硬件操作时，乘法操作需要附加的寄存器参与，所以比平方操作耗时长。因此，攻击者可以在解密过程中观测执行时间的差异来确定 d_i 为 1 或为 0。例如，在攻击过程中，选择一系列的密文让智能卡签名并测量耗时，这里还需要一个 e 值（e 值是根据经验确定的），通过样本分析得出，如果记录的时间超过 e，则表示 d_i 为 1，否则为 0，通过求 d_i 的期望值统计分析 d_i 是 1 还是 0，这样用到统计学的方法，通过大量的样本分析，最终逐位获取整个私钥 d。

应对措施如下。

（1）随机延时：通过模指数运算增加一个随机延时迷惑计时攻击者。

（2）盲化：执行模指数运算之前先将一个随机数与密文相乘，使得计时攻击者不知计算机正在处理哪几位密文，防止计时攻击者一位一位地进行分析。

7．低熵攻击法

如果明文空间熵太小或是小明文空间，则 RSA 算法容易受到低熵攻击。密码分析者预先构造出某个完整的解密表，对所有可能的明文进行加密，将所得到的密文按一定的规则排列。攻击者只需查表，无须解密即可获得明文。

应对措施：加密前对明文系统加以筛选，去掉熵太小的明文空间或小明文空间。这样就能确保密钥的模 n 的有效长度大于 2048，使得解密表的范围增加，构造成本足够高。

8．公共指数攻击法

公共指数攻击法利用使用者的使用习惯进行攻击。RSA 算法原理中曾经提到过，最常用作公钥 b 的数有 3 个：31、765、537，如果许多使用 RSA 算法的机构使用的公钥 b 为这 3 个数中的任何一个，则攻击者使用最为简单的猜测法即可猜测出来。

应对措施：将 b 的取值范围设得更宽些，使得攻击者无法完成对 b 取值的猜测。

3.6　公钥密码的特征总结

3.6.1　公钥密码的原理总结

公钥密码涉及两个密钥，通信的参与者拥有公钥和私钥成对的密钥。公钥公开，理论上任何人都可以知道，用于加密明文和验证签名。私钥仅拥有者自己知道，用于解密或构造签名。公钥密码又称非对称密码，所谓“非对称”包括两方面：一是密钥的不对称，即用于加密和解密的密钥是不同的，用于加密的密钥不能解密，用于解密的密钥不能加密；二是通信双方的地位不对等，一方拥有自己独有的秘密，而另一方无法知道这个秘密。

3.6.2　公钥密码算法的基础

公钥密码算法设计有以下基本要求。

（1）加密和解密由不同的密钥完成。

（2）已知加密算法，从加密密钥中得到解密密钥在计算上是不可行的。

（3）两个密钥中任何一个都可以用作加密，而另一个用作解密。

3.6.3 单向函数

1976 年，Diffie 和 Hellman 在论文 *New Direction in Cryptography* 中提出了公钥密码的概念。1978 年， Rivest RL 、 Shamir A 和 Adleman LM 在《实现数字签名和公钥密码体制的一种方法》中提出了一种可行的实现方法，这就是我们广泛使用的 RSA 密码体制。RSA 密码体制的提出真正使得互不相识的通信双方在一个不安全信道上进行安全通信成为可能，其背后的基础是单向函数。

给定任意两个集合 X 和 Y。函数 $f:X\to Y$ 称为单向的，对于每个 x 属于 X ，很容易计算出函数 $f(x)$，而对于大多数 y 属于 Y，要确定满足 $y=f(x)$ 的 x 在计算上是很困难的（假设至少有这样一个 x 存在）。不能将单向函数的概念与数学意义上的不可逆函数的概念混同，因为单向函数可能是数学意义上可逆或一对一的函数，而不可逆函数不一定是单向函数。

1．公式定义

函数 $y=f:\{0,1\}^*\to\{0,1\}^*$ 是一个单向函数，当且仅当 f 可以用一个多项式时间算法计算时，但是对于任意一个以 x 为输入的随机化多项式时间算法 A，任意一个多项式 $p(n)$ 和足够大 n，使得概率

$$\Pr_{x\in\{0,1\}^n}\left[f\left(A\left(f(x)\right)\right)=f(x)\right]<\frac{1}{p(n)} \tag{3.6}$$

2．陷门单向函数

单向函数不能直接用于构造公钥密码算法，因为如果用单向函数对明文进行加密，即使是合法的接收方也不能还原出明文，因为单向函数的逆运算是困难的。与公钥密码算法关系更为密切的概念是陷门单向函数。一个函数 $f:X\to Y$ 称为是陷门单向的，如果该函数及其逆函数的计算都存在有效的算法，而且可以将计算 f 的方法公开，那么即使由计算 f 的完整方法也不能推导出其逆运算的有效算法。其中，使得双向都能有效计算的秘密信息称为陷门（Trap Door）。需要注意的是，不能顾名思义地认为陷门单向函数是单向函数。事实上，陷门单向函数不是单向函数，它只是对于那些不知道陷门的人表现出了单向函数的特性。

3．陷门单向函数与公钥密码算法

公钥密码算法的设计就是寻找陷门单向函数。1976 年， Diffie 和 Hellman 提出了公钥密码和陷门单向函数的概念时，并没能给出一个陷门单向函数的实例。第 1 个陷门单向函数和第 1 个公钥密码算法在 1977 年才被提出。此后，人们又尝试过很多种单向函数的设计方法，如背包问题、纠错码问题、因子分解问题、离散对数问题、有限自动机合成问题等，但当前除了因子分解问题和离散对数问题，其他陷门单向函数都被证明存在安全缺陷或因为其复杂性不能归约到某个问题而无法得到广泛的认可。

基于因子分解问题的公钥密码算法除了大家熟知的 RSA 算法，还有 Rabin 算法、Feige-Fiat-Shamir 算法等。后两个算法不被大家熟悉的主要原因是，一个只能用于身份认证，另一

个虽然被证明其破译难度与大整数因子分解一样，但不能抵抗选择密文攻击。

基于离散对数问题的公钥密码算法主要有数字签名标准（DSS）、椭圆曲线密码（ECC）和 DH 算法等。DH 算法并不能用于加密和解密，只能用于在不安全信道上进行密钥分配，真正第 1 个可以用于加密和解密的公钥密码算法是 RSA 算法。

基于有限自动机合成问题的公钥密码算法 FAPKC 是我国著名学者陶仁骥教授于 1985 年首次提出的，该算法在 1995 年被攻破。后来有一些变种，但因为其所基于的自动机理论不是广泛熟悉的理论，以及前身被破解等原因，没有能够从安全性上得到人们广泛的认可。

背包密码算法基于一般背包问题求解是 NP 难，而递增背包的求解基于快速算法的原理进行设计。该算法提出后不久就被破解，而且破解方法同样适用于其所有变种。

基于纠错码问题的第 1 个公钥密码算法是 McEliece 算法，它是基于 Goppa 的解码存在快速算法而一般线性码的解码是 NP 难的原理进行设计的，该算法在 1992 年被破解。基于类似的方法，国内知名学者王新梅等也提出了一系列基于纠错码问题的公钥密码算法，但在 1992 年前后都被破解。

单向函数的另一个应用是用于数字签名时产生信息摘要的单向散列函数（哈希函数），将在第 4 章进行讨论。由于公钥密码的运算量往往比较大，为了避免对文件进行全文签名，一般在签名运算前使用单向散列函数对待签文件进行摘要处理，将待签文件压缩成一个分组之内的定长位串，以提高签名的效率。MD5 和 SHA-1 就是两个曾被广泛使用的、具有单向函数性质的散列函数。有些学者把密码算法分成三类，单向散列函数就是其中很重要的一类，另外两类分别是公钥密码和对称密码。

构造公钥密码系统的关键是如何在求解某个单向函数的逆函数的 NP 完全问题中设置合理的陷门，提供陷门单向函数的数学难题主要有三类：大整数分解问题（简称 IFP ）、离散对数问题（简称 DLP ）、椭圆曲线离散对数问题（简称 ECDLP ）。

建立在不同计算难题上的其他公钥密码算法还有基于有限域中离散对数问题的 ElGamal 公钥密码算法、基于“子集和”难题的 Merkle-Hellman Knapsack（背包）公钥密码算法等。

3.7　DH 算法

DH 算法是第 1 个公钥密码算法，使用在一些常用安全协议或产品（如 SSH 等）中。它是一种密钥交换方案，不能用于加密传输任意信息，仅允许两个参与方可以安全地建立一个共享的秘密信息，用于后续的保密通信，该秘密信息（可称会话密钥）仅为两个参与方所知。

3.7.1　DH 算法的数学基础

DH 算法的有效性建立在计算离散对数这一问题的基础上。简单地说，离散对数的定义如下。首先定义质数 p 的本原根。质数 p 的本原根是一个整数，且其幂可以产生 1 到 $p-1$ 之间的所有整数。也就是说，若 a 是质数 p 的本原根，则

$$a \pmod p, a^2 \pmod p, \cdots, a^{p-1} \pmod p$$

各不相同，它是 1 到 $p-1$ 的一个置换。

对于任意整数 b 和质数 p 的本原根 a，我们可以找到唯一的指数，使得

$$b \equiv a^i \pmod{p} \quad (0 \leqslant i \leqslant p-1)$$

指数 i 称为 b 的以 a 为底的模 p 的离散对数，记为 $d\log_a p(b)$。

3.7.2 DH 算法的具体内容

图 3.6 所示为 DH 算法。在 DH 算法中，质数 q 和其本原根 a 是两个公开的整数。假定用户 Alice 和 Bob 希望交换密钥，那么 Alice 选择一个随机整数 $X_A < q$，并计算 $Y_A = a^{X_A} \pmod{q}$。类似地，Bob 也独立地选择一个随机整数 $X_B < q$，并计算 $Y_B = a^{X_B} \pmod{q}$。Alice 和 Bob 确保各自的私钥 X 是私有的，公钥 Y 是公开可访问的。Alice 计算 $K = (Y_B)^{X_A} \pmod{q}$ 并将其作为密钥，Bob 计算 $K = (Y_A)^{X_B} \pmod{q}$ 并将其作为密钥。这两种计算所得的结果是相同的。

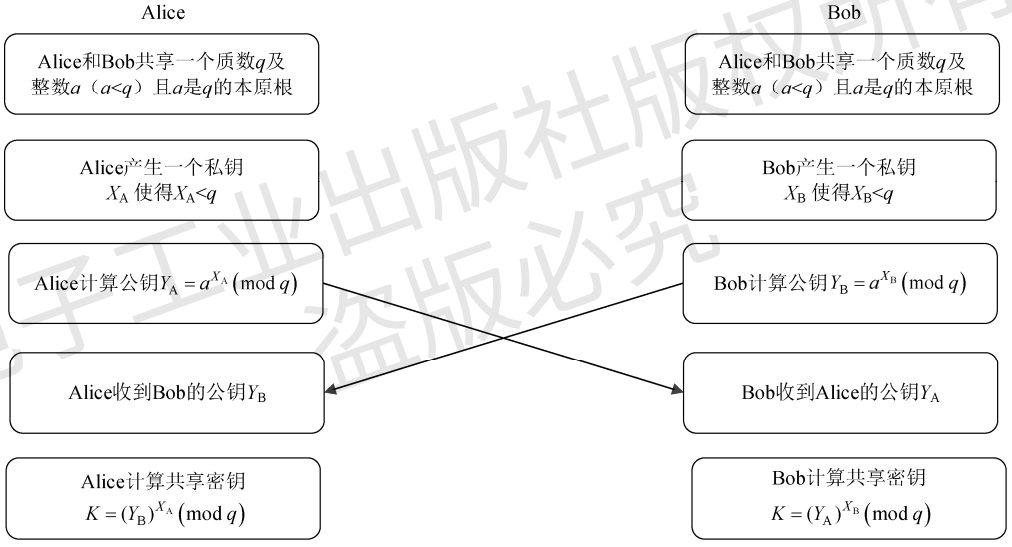


图 3.6　DH 算法

根据模算术的运算规律：

$$\begin{aligned}
K &= \left(Y_B\right)^{X_A} \pmod{q} \\
&= \left(a^{X_B} \pmod{q}\right)^{X_A} \pmod{q} \\
&= \left(a^{X_B}\right)^{X_A} \pmod{q} \\
&= a^{X_B X_A} \pmod{q} \\
&= \left(a^{X_A}\right)^{X_B} \pmod{q} \\
&= \left(a^{X_A} \pmod{q}\right)^{X_B} \pmod{q} \\
&= \left(Y_A\right)^{X_B} \pmod{q}
\end{aligned}$$

（1）通信双方选择质数 q，以及 q 的一个本原根 a。

（2）用户 A 选择一个随机数 $X_A < q$，计算 $Y_A = a^{X_A} \pmod q$。

（3）用户 B 选择一个随机数 $X_B < q$，计算 $Y_B = a^{X_B} \pmod q$。

（4）双方保密 X，而将 Y 交换给对方，即 X 是私钥，Y 是公钥。

双方获得一个共享密钥 $K = a^{X_A X_B} \pmod q$。

对于用户 A，计算出 $K = Y_B{}^{X_A} \pmod q$。

对于用户 B，计算出 $K = Y_A{}^{X_B} \pmod q$。

攻击者要获得 K，需要求解离散对数问题。在实际使用中，质数 q 及其本原根 a 可由一方选择后发送给另一方。

DH 算法的安全性建立在下述事实之上：求关于质数的模质数幂运算相对容易，而求解离散对数问题却非常困难；对于大质数，求解离散对数问题被认为在计算上是不可行的。

3.8　椭圆曲线密码算法

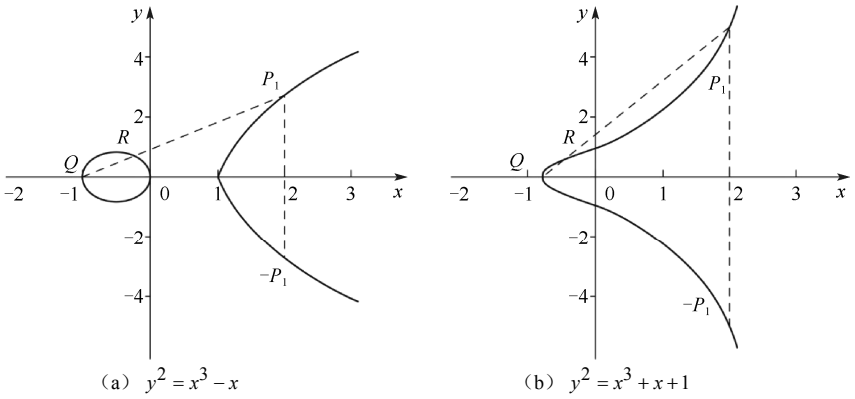
为保证 RSA 算法的安全性与足够的破译难度，在应用过程中它的密钥长度需要一再增大，使得运算负担越来越大。相比之下，椭圆曲线密码（Elliptic Curve Cryptography，ECC）可用短得多的密钥获得同样的安全性，因此具有广泛的应用前景。ECC 算法已被 IEEE 公钥密码标准 P1363 采用。

3.8.1　椭圆曲线

椭圆曲线并非椭圆，之所以称为椭圆曲线，是因为它的曲线方程与计算椭圆周长的方程类似。一般地，椭圆曲线的曲线方程是以下形式的三次方程：

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

式中，a,b,c,d,e 是满足某些简单条件的实数。定义中包括一个称为无穷远点的元素，记为 O。图 3.7 所示为椭圆曲线的举例。



（a）$y^2 = x^3 - x$　　（b）$y^2 = x^3 + x + 1$

图 3.7　椭圆曲线的举例

由图 3.7 可见，椭圆曲线关于 x 轴对称。

椭圆曲线上的加法运算定义如下：如果其上的 3 个点位于同一条直线上，那么它们的和为 O。进一步可定义椭圆曲线上的加法律（加法法则）。

（1）O 为加法单位元，即对于椭圆曲线上的任一点 P，都有 $P+O=P$。

（2）设 $P_1=(x,y)$ 是椭圆曲线上的一点（见图 3.7），它的加法逆元定义为 $P_2=-P_1=(x,-y)$。

这是因为 P_1、P_2 点的连线延长到无穷远时，得到椭圆曲线上的另一点 O，即椭圆曲线上的 3 个点 P_1、P_2、O 共线，所以 $P_1+P_2+O=O$，$P_1+P_2=O$，即 $P_2=-P_1$。

由 $O+O=O$ 可得 $O=-O$。

（3）设 Q 和 R 是椭圆曲线上 x 坐标不同的两点，$Q+R$ 的定义如下：画一条通过 Q、R 点的直线，与椭圆曲线交于 P_1 点（这一交点是唯一的，除非所画的直线是 Q 点或 R 点的切线，此时分别取 $P_1=Q$ 和 $P_1=R$）。由 $Q+R+P_1=O$ 可得 $Q+R=-P_1$。

（4）Q 点的倍数定义如下：在 Q 点做椭圆曲线的一条切线，设切线与椭圆曲线交于 S 点，定义 $2Q=Q+Q=-S$。类似地，可定义 $3Q=Q+Q+Q$ 等。

以上定义的加法律具有加法运算的一般性质，如交换律、结合律等。

3.8.2 有限域上的椭圆曲线

密码算法中普遍采用的是有限域上的椭圆曲线，有限域上的椭圆曲线是指在曲线方程定义式中，所有系数都是某一有限域 $\mathrm{GF}(p)$ 中的元素（p 为一个大质数）。其中最为常用的是由方程

$$y^2=x^3+ax+b\left(a,b\in \mathrm{GF}(p),4a^3+27b^2\neq 0\right)$$

定义的曲线。

因为 $\Delta=\left(\frac{a}{3}\right)^3+\left(\frac{b}{2}\right)^2=\frac{1}{108}\left(4a^3+27b^2\right)$ 是方程 $x^3+ax+b=0$ 的判别式，当 $4a^3+27b^2=0$ 时，方程 $x^3+ax+b=0$ 有重根，设为 x_0，则 $Q_0=(x_0,0)$ 点是方程 $y^2=x^3+ax+b$ 的重根。令 $F(x,y)=y^2-x^3-ax-b$，则 $\left.\frac{\partial F}{\partial x}\right|_{Q_0}=\left.\frac{\partial F}{\partial y}\right|_{Q_0}=0$，所以 $\frac{\mathrm{d}y}{\mathrm{d}x}=-\frac{\partial F}{\partial x}/\frac{\partial F}{\partial y}$ 在 Q_0 点无定义，即曲线 $y^2\equiv x^3+ax+b$ 在 Q_0 点的切线无定义，因此 Q_0 点的倍点运算无定义。

例如，$p=23,a=b=1,4a^3+27b^2=8\neq 0$，方程为 $y^2\equiv x^3+x+1$，其图形是连续曲线，如图 3.7（b）所示。然而我们感兴趣的是曲线在第Ⅰ象限中的整数点。设 $E_p(a,b)$ 表示方程定义的椭圆曲线上的点集 $\{(x,y)|0\leqslant x<p,0\leqslant y<p$ 且 x,y 均为整数$\}$ 并上无穷远点 O。本例中的 $E_{23}(1,1)$ 由表 3.3 给出，表中未给出 O。

表 3.3　椭圆曲线上的点集 $E_{23}(1,1)$

(0,1)	(0,22)	(1,7)	(1,16)	(3,10)	(3,13)	(4,0)	(5,4)	(5,19)
(6,4)	(6,19)	(7,11)	(7,12)	(9,7)	(9,16)	(11,3)	(11,20)	(12,4)
(12,19)	(13,7)	(13,16)	(17,3)	(17,20)	(18,3)	(18,20)	(19,5)	(19,18)

一般来说，$E_p(a,b)$ 由以下方式产生。

（1）对每个 $x\left(0 \leqslant x < p \text{且} x \text{为整数}\right)$ 都计算 $x^3 + ax + b \pmod p$。

（2）决定（1）中求得的值在模 p 下是否有平方根，如果没有，则曲线上没有与 x 相对应的点；如果有，则求出两个平方根（$y=0$ 时只有一个平方根）。

$E_p(a,b)$ 上的加法定义如下（设 $P,Q \in E_p(a,b)$）。

（1）$P+O=P$。

（2）如果 $P=(x,y)$，那么 $(x,y)+(x,-y)=O$，即 $(x,-y)$ 是 P 点的加法逆元，表示为 $-P$。

由 $E_p(a,b)$ 的产生方式可知，$-P$ 也是 $E_p(a,b)$ 中的点，如上例，$P=(13,7) \in E_{23}(1,1)$，$-P=(13,-7)$，而 $-7 \pmod{23} \equiv 16$，所以 $-P=(13,16)$，也在 $E_{23}(1,1)$ 中。

（3）设 $P=(x_1,y_1), Q=(x_2,y_2), P \neq -Q$，则 $P+Q=(x_3,y_3)$ 由以下规则确定：

$$\begin{aligned} x_3 &\equiv \lambda^2 - x_1 - x_2 \pmod p \\ y_3 &\equiv \lambda(x_1 - x_3) - y_1 \pmod p \end{aligned} \tag{3.7}$$

式中

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, & P = Q \end{cases} \tag{3.8}$$

【例 3-1】仍以 $E_{23}(1,1)$ 为例，设 $P=(3,10), Q=(9,7)$，则

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} \equiv 11 \pmod{23}$$

$$x_3 \equiv 11^2 - 3 - 9 \pmod{23}$$

$$y_3 \equiv 11(3-17) - 10 \pmod{23}$$

所以 $P+Q=(17,20)$，仍为 $E_{23}(1,1)$ 中的点。

若求 $2P$，则

$$\lambda = \frac{3 \times 3^2 + 1}{2 \times 10} = \frac{28}{20} = \frac{7}{5} \equiv 6 \pmod{23}$$

$$x_3 \equiv 6^2 - 3 - 3 \pmod{23}$$

$$y_3 \equiv 6(3-7) - 10 \pmod{23}$$

所以 $2P=(7,12)$。

倍点运算仍定义为重复加法，如 $4P=P+P+P+P$。

从例 3-1 中可以看出，加法运算在 $E_{23}(1,1)$ 中是封闭的，且能验证其满足交换律。对于

一般 $E_p(a,b)$，可证明其上的加法运算是封闭的、满足交换律，还能证明其上的加法逆元运算也是封闭的，所以 $E_p(a,b)$ 是一个阿贝尔群。

【例 3-2】已知 $y^2 \equiv x^3 - 2x - 3$ 是系数在 $\mathrm{GF}(7)$ 上的椭圆曲线，$P=(3,2)$ 是其上一点，求 $10P$。

解：

$$2P = P + P = (3,2) + (3,2) = (2,6)$$
$$3P = P + 2P = (3,2) + (2,6) = (4,2)$$
$$4P = P + 3P = (3,2) + (4,2) = (0,5)$$
$$5P = P + 4P = (3,2) + (0,5) = (5,0)$$
$$6P = P + 5P = (3,2) + (5,0) = (0,2)$$
$$7P = P + 6P = (3,2) + (0,2) = (4,5)$$
$$8P = P + 7P = (3,2) + (4,5) = (2,1)$$
$$9P = P + 8P = (3,2) + (2,1) = (3,5)$$
$$10P = P + 9P = (3,2) + (3,5) = O$$

3.8.3 椭圆曲线上的点数

在 3.8.2 节的例子中，$\mathrm{GF}(23)$ 上的椭圆曲线 $y^2 \equiv x^3 + x + 1$ 在第Ⅰ象限中的整数点加无穷远点 O 共有 28 个，一般有以下定理。

$\mathrm{GF}(p)$ 上的椭圆曲线 $y^2 = x^3 + ax + b\left(a,b \in \mathrm{GF}(p), 4a^3 + 27b^2 \neq 0\right)$ 在第Ⅰ象限中的整数点加无穷远点 O 共有

$$1 + p + \sum_{x \in \mathrm{GF}(p)} \left(\frac{x^3 + ax + b}{p} \right) = 1 + p + \varepsilon$$

个，其中 $\left(\frac{x^3 + ax + b}{p} \right)$ 是勒让德（Legendre）符号。ε 由以下定理给出：Hasse 定理，即 $|\varepsilon| \leqslant 2\sqrt{p}$。

【例 3-3】若 $p=5$，则 $|\varepsilon| \leqslant 4$。因此 $\mathrm{GF}(5)$ 上的椭圆曲线 $y^2 = x^3 + ax + b$ 上的点数为 2～10。

3.8.4 明文嵌入椭圆曲线

在使用椭圆曲线构造密码算法之前，需要将明文嵌入椭圆曲线，作为椭圆曲线上的点。设明文是 $m(0 \leqslant m \leqslant M)$，椭圆曲线由 3.8.2 节中有限域上的常用方程给出，k 是一个足够大的整数，使得将明文嵌入椭圆曲线时，错误概率是 2^{-k}。实际中，k 可在 30～50 之间取值。下面取 $k=30$，对于明文 m，如下计算一系列 x：

$$x = \{mk + j, j = 0,1,2,\cdots\} = \{30m, 30m+1, 30m+2, \cdots\} \tag{3.9}$$

直到 $x^3+ax+b\pmod p$ 是平方根，即得到椭圆曲线上的点 $\left(x,\sqrt{x^3+ax+b}\right)$。因为在 0～$p$ 的整数中，有一半是模 p 的平方剩余，另一半是模 p 的非平方剩余。所以 k 次找到 x，使得 $x^3+ax+b\pmod p$ 是平方根的概率不小于 $1-2^{-k}$。

反过来，为了从椭圆曲线上的点 (x,y) 上得到明文 m，只需求 $m=\left\lfloor\frac{x}{30}\right\rfloor$。

【例 3-4】设椭圆曲线为 $y^2=x^3+3x, p=4177, m=2174$，则 $x=\{30\times2174+j, j=0,1,2,\cdots\}$。当 $j=15$ 时，$x=30\times2174+15=65235, x^3+3x=65235^3+3\times65235+1444\pmod{4177}=38^2$，得到椭圆曲线上的点为 $(65235,38)$。若已知椭圆曲线上的点 $(65235,38)$，则明文 $m=\left\lfloor\frac{65235}{30}\right\rfloor=\lfloor2174.5\rfloor=2174$。

3.8.5　椭圆曲线上的密码算法

为使用椭圆曲线构造密码算法，需要找出椭圆曲线上的数学难题。

在椭圆曲线构成的阿贝尔群 $E_p(a,b)$ 上考虑方程 $Q=kP$，其中，$P,Q\in E_p(a,b), k<p$，则由 k 和 P 易求 Q，但由 P、Q 求 k 则是困难的，这就是椭圆曲线上的离散对数问题，可应用于公钥密码算法。DH 算法和 ElGamal 密码算法是基于有限域上离散对数问题的公钥密码算法，下面考虑如何用椭圆曲线实现这两种密码算法。

1. DH 算法

首先取一个质数 $p\approx2^{180}$ 和两个参数 a、b，得到方程表达的椭圆曲线及其上面的点构成的阿贝尔群 $E_p(a,b)$。然后取 $E_p(a,b)$ 的一个生成元 $G(x_1,y_1)$，要求 G 的阶是一个非常大的质数，是满足 $nG=O$ 的最小正整数 n。$E_p(a,b)$ 和 G 作为公开参数。

用户 A 和 B 之间的密钥交换如下。

（1）用户 A 选取一小于 n 的整数 n_A 作为私钥，并由 $P_A=n_AG$ 产生 $E_p(a,b)$ 上的一点作为公钥。

（2）用户 B 类似地选取自己的私钥 n_B 和公钥 P_B。

（3）用户 A 和 B 分别由 $K=n_AP_B$ 和 $K=n_BP_A$ 产生双方共享的密钥。

这是因为 $K=n_AP_B=n_A(n_BG)=n_B(n_AG)=n_BP_A$。攻击者若想获取 K，则必须由 P_A 和 G 求出 n_A，或者由 P_B 和 G 求出 n_B，即需要求解椭圆曲线上的离散对数问题，而这是不可行的。

【例 3-5】$p=211, E_p(0,-4)$，即椭圆曲线为 $y^2\equiv x^3-4, G=(2,2)$ 是 $E_{211}(0,-4)$ 的阶为 241 的一个生成元，即 $241G=O$。用户 A 的私钥为 $n_A=121$，公钥为 $P_A=121(2,2)=(115,48)$。用户 B 的私钥为 $n_B=203$，公钥为 $P_B=203(2,2)=(130,203)$。由此得到的共享密钥为 $121(130,203)=203(115,48)=(161,169)$，即共享密钥是一对数。如果将这一密钥用作单密钥加密的会话密钥，则可简单地取其中的一个，如取 x 坐标，或者取 x 坐标的某一简单函数。

2．ElGamal 密码算法

密钥产生过程如下：首先选择一个质数 p 及两个小于 p 的随机数 g 和 x，计算 $y \equiv g^x \pmod p$。以 (y,g,p) 为公钥，x 为私钥。

加密过程如下：设待加密明文为 M，随机选择一个与 $p-1$ 互质的整数 k，计算 $C_1 \equiv g^k \pmod p, C_2 \equiv y^k M \pmod p$，密文为 $C=(C_1,C_2)$。

解密过程如下：

$$M = \frac{C_2}{C_1^x} \pmod p \tag{3.10}$$

这是因为

$$\frac{C_2}{C_1^x} \pmod p = \frac{y^k M}{g^{kx}} \pmod p = \frac{y^k M}{y^k} \pmod p = M \pmod p \tag{3.11}$$

下面讨论利用椭圆曲线实现 ElGamal 密码算法。

首先选取一条椭圆曲线，并得 $E_p(a,b)$，将明文 m 嵌入椭圆曲线上的点 P_m，再对点 P_m 做加密变换。

取 $E_p(a,b)$ 的一个生成元 G，$E_p(a,b)$ 和 G 作为公开参数。

用户 A 选取 n_A 作为私钥，并以 $P_A = n_A G$ 为公钥。用户 B 若想向用户 A 发送信息 P_m，则可选取一个随机正整数 k，产生以下点对作为密文：

$$C_m = \{kG, P_m + kP_A\} \tag{3.12}$$

用户 A 解密时，以密文点对中的第 2 个点减去用户自己的私钥与第 1 个点的倍乘，即

$$P_m + kP_A - n_A kG = P_m + k(n_A G) - n_A kG = P_m \tag{3.13}$$

攻击者若想由 C_m 得到 P_m，就必须知道 k。而要得到 k，只有通过椭圆曲线上的两个已知点 G 和 kG，这意味着必须求解椭圆曲线上的离散对数问题，因此不可行。

【例 3-6】取 $p=751, E_p(-1,188)$，即椭圆曲线为 $y^2 \equiv x^3 - x + 188, E_p(-1,188)$ 的一个生成元是 $G=(0,376)$，用户 A 的公钥为 $P_A=(201,5)$。假定用户 B 已将欲发往用户 A 的明文嵌入椭圆曲线上的点 $P_m=(562,201)$，用户 B 选取随机数 $k=386$，由 $kG=386(0,376)=(676,558)$ 和 $P_m + kP_A = (562,201) + 386(201,5) = (385,328)$ 得密文为 $\{(676,558),(385,328)\}$。

与基于有限域上的离散对数问题的公钥密码算法（如 DH 密钥交换和 ElGamal 密码算法）相比，ECC 算法具有以下优点。

1）安全性高

攻击有限域上的离散对数问题有指数积分法，其运算复杂度为 $O\left(\exp\sqrt[3]{(\log_2 p)(\log\log_2 p)^2}\right)$，其中，$p$ 是模数，为质数。而它对椭圆曲线上的离散对数问题并不是有效的。目前，攻击椭圆曲线上的离散对数问题只适合攻击任何循环群上的离散对数问题的大步小步法，其运算复杂度为 $O\left(\exp\left(\log_2\sqrt{p_{\max}}\right)\right)$，其中，$p_{\max}$ 是椭圆曲线所形成的阿贝尔群的阶的最大质因子。因此，ECC 算法比基于有限域上的离散对数问题的公钥密码算法更安全。

2）密钥量小

由攻击二者的运算复杂度可知，在实现相同安全性的条件下，ECC 算法所需的密钥量远比基于有限域上的离散对数问题的公钥密码算法的密钥量小。

3）灵活性好

在有限域$\mathrm{GF}(q)$一定的情况下，其上的循环群$\left(\mathrm{GF}(q)-\{0\}\right)$就确定了。而$\mathrm{GF}(q)$上的椭圆曲线可以通过改变曲线参数得到不同的曲线，形成不同的循环群。因此，椭圆曲线具有丰富的群结构和多选择性。

正是由于椭圆曲线具有丰富的群结构和多选择性，并可在保持和 RSA/DSA 算法相同安全性的前提下大大缩短密钥长度（目前 160 比特足以保证安全性），因而在密码领域有着广阔的应用前景。表 3.4 给出了 ECC 算法和 RSA/DSA 算法在保持相同安全性的条件下所需的密钥长度。

表 3.4　ECC 算法和 RSA/DSA 算法在保持相同安全性的条件下所需的密钥长度（单位：比特）

RSA/DSA 算法	512	768	1024	2048	21000
ECC 算法	106	132	160	211	600

3.9　SM2 算法

SM2 算法是中国国家密码管理局颁布的中国商用公钥密码标准算法，它是一组 ECC 算法，其中包含加密算法、解密算法、数字签名算法。

SM2 算法与国际标准的 ECC 算法相比具有以下特点。

（1）ECC 算法通常采用 NIST 等国际机构建议的曲线及参数，而 SM2 算法的参数需要利用一定的算法产生。由于 SM2 算法中加入了用户特异性的曲线参数、基点、用户的公钥点信息，故其安全性得到了明显提高。

（2）在 ECC 算法中，用户可以选择 MD5 或 SHA-1 等国际通用的哈希算法。而在 SM2 算法中则使用 SM3 哈希算法，输出为 256 比特，其安全性与 SHA-256 算法基本相当。

SM2 算法分为基于质数域和基于二元扩域两种。本节仅介绍基于质数域的 SM2 算法。

3.9.1　基本参数

基于质数域F_p的 SM2 算法的基本参数如下。

（1）F_p的特征 p 为 m 比特长的质数，p 要尽可能大，但太大会影响计算速度。

（2）长度不小于 192 比特的比特串 SEED。

（3）F_p上的两个元素 a、b 满足$4a^3+27b^2\neq 0$，定义曲线$E\left(F_p\right)$：$y^2=x^2+ax+b$。

（4）基点$G=\left(x_G,y_G\right)\in E\left(F_p\right),G\neq O$。

（5）G 的阶 n 为 m 比特长的质数，满足$n>2^{191}$且$n>4\sqrt{p}$。

（6）$h=\dfrac{\left|E\left(F_p\right)\right|}{n}$ 称为余因子，其中，$\left|E\left(F_p\right)\right|$ 是曲线 $E\left(F_p\right)$ 的点数。

SEED 和 a、b 的产生算法如下。

（1）任意选取长度不小于 192 比特的比特串 SEED。

（2）计算 $H=H_{256}(\text{SEED})$，记 $H=\left(h_{255},h_{254},\cdots,h_0\right)$，其中，$H_{256}$ 表示 256 比特输出的 SM3 算法。

（3）取 $R=\sum_{i=0}^{255}h_i 2^i$。

（4）取 $r=R(\bmod p)$。

（5）在 F_p 上任意选择两个元素 a、b，满足 $rb^2=a^3(\bmod p)$。

（6）若 $4a^3+27b^2=0(\bmod p)$，则转向（1）。

（7）所选择的 F_p 上的曲线是 $E\left(F_p\right)$：$y^2=x^2+ax+b$。

（8）输出 (SEED,a,b)。

3.9.2 密钥产生

设接收方 B 的私钥取为 $\{1,2,\cdots,n-1\}$ 中的一个随机数 d_{B}，记为 $d_{\text{B}}\leftarrow_R\{1,2,\cdots,n-1\}$，其中，$n$ 是基点 G 的阶。

接收方 B 的公钥取为椭圆曲线上的点：

$$P_{\text{B}}=d_{\text{B}}G \tag{3.14}$$

式中，$G=G(x,y)$ 是基点。

3.9.3 加密算法

设发送方 A 要发送的消息为比特串 M，M 的长度为 klen。加密运算如下。

（1）选择随机数 $k\leftarrow_R\{1,2,\cdots,n-1\}$。

（2）计算椭圆曲线上的点 $C_1=kG=(x_1,y_1)$，将 (x_1,y_1) 表示为比特串。

（3）计算椭圆曲线上的点 $S=hP_{\text{B}}$，若 S 是无穷远点，则报错退出。

（4）计算椭圆曲线上的点 $kP_{\text{B}}=(x_2,y_2)$，将 (x_2,y_2) 表示为比特串。

（5）计算 $t=\text{KDF}(x_2\parallel y_2,\text{klen})$，若 t 为全 0 的比特串，则返回（1）。

（6）计算 $C_2=M\oplus t$。

（7）计算 $C_3=\text{Hash}(x_2\parallel M\parallel y_2)$。

（8）输出密文 $C=(C_1,C_2,C_3)$。

其中，（5）中的 $\text{KDF}(\cdot)$ 是密钥派生函数，其本质就是一个伪随机数产生函数，用来产生密钥，取为 SM3 算法。（7）中的 $\text{Hash}(\cdot)$ 函数也取为 SM3 算法。

图 3.8 所示为 SM2 加密算法的流程图。

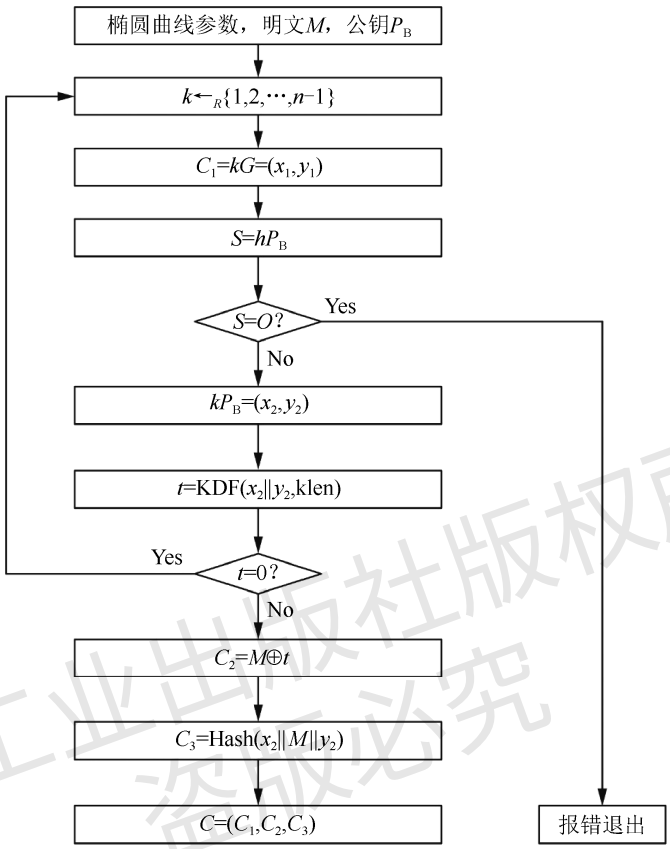


图 3.8　SM2 加密算法的流程图

3.9.4　解密算法

接收方 B 收到密文 L 后，执行以下解密运算。

（1）从密文 C 中取出比特串 C_1，将 C_1 表示为椭圆曲线上的点，验证 C_1 是否满足椭圆曲线方程，若不满足，则报错退出。

（2）计算椭圆曲线上的点 $S=hC_1$，若 S 是无穷远点，则报错退出。

（3）计算 $d_BC_1=(x_2,y_2)$，将 (x_2,y_2) 表示为比特串。

（4）计算 $t=\mathrm{KDF}(x_2\|y_2,\mathrm{klen})$，若 t 为全 0 比特串，则报错退出。

（5）从 C 中取出比特串 C_2，计算 $M=C_2\oplus t$。

（6）计算 $u=\mathrm{Hash}(x_2\|M\|y_2)$，从 C 中取出 C_3，若 $u\neq C_3$，则报错退出。

（7）输出明文 M。

图 3.9 所示为 SM2 解密算法的流程图。

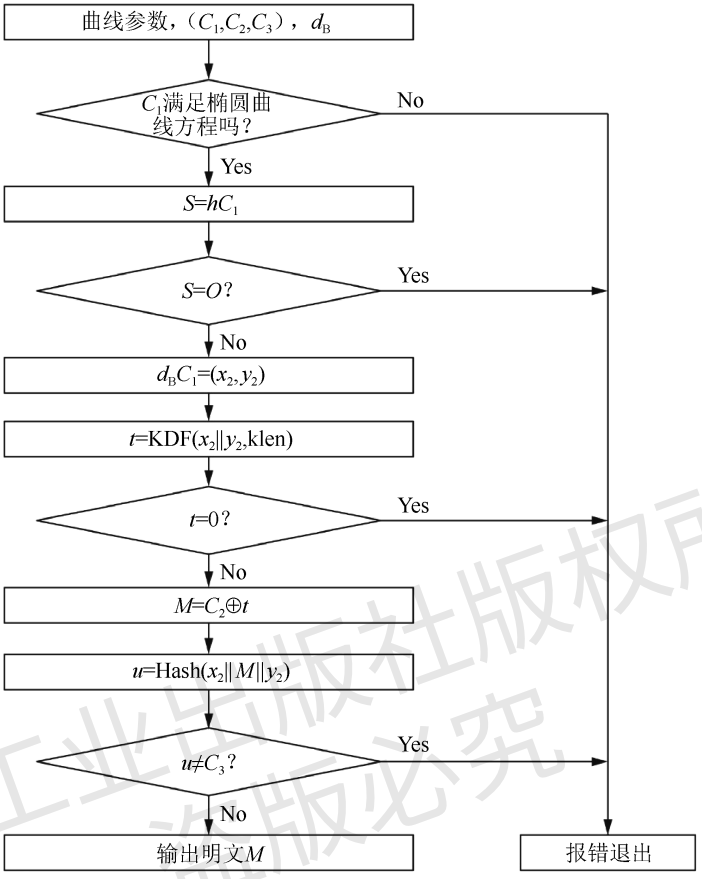


图 3.9　SM2 解密算法的流程图

解密的正确性：因为 $P_B = d_B G, C_1 = kG = (x_1, y_1)$，由解密算法的（3）可得

$$d_B C_1 = d_B kG = k(d_B G) = kP_B = (x_2, y_2) \tag{3.15}$$

所以解密算法（4）中得到的 t 与加密算法（5）中得到的 t 相等，由 $C_2 \oplus t$ 得到明文 M。

习　题

1．试讨论对称加密与非对称加密的概念、二者的区别及它们的优缺点？

2．请阐述使用 DH 算法产生一个会话密钥的具体过程。

3．试简述对称密码和公钥密码的区别，并指出它们各自所能提供的安全服务。

4．给出至少 3 种对 RSA 密码的攻击方法，并简述其原理。

5．（　　）算法只能用于实现密钥交换，算法的安全性依赖有限域上的离散对数问题。

A．ECC　　B．AES　　C．RSA　　D．DH

6．下面关于公钥密码体制的说法，不正确的是（　　）。

A．一般情况下，公钥密码的加密和解密速度较对称密码慢

B．若知道加密算法，则从加密密钥中得到解密密钥在计算上是不可行的

C．加密和解密由不同的密钥完成，并且可以交换使用

D．通信双方掌握的秘密信息（密钥）是一样的

7．1976 年，Diffie 和 Hellman 在论文 *New Direction in Cryptography* 中首次提出了公钥密码的思想，（　　）。

A．将公钥公开，将私钥保密　　B．将私钥公开，将公钥保密

C．将公钥和私钥都保密　　D．将公钥和私钥都公开

8．（　　）算法不是分组加密算法。

A．AES　　B．RSA　　C．DES　　D．RC4

9．（　　）不属于 RSA 算法的应用。

A．密钥交换　　B．数据加密　　C．数字签名　　D．可靠传输