第 5 章　公钥密码技术

本章学习目标

公钥密码体制加密密钥的公开，解决了密钥管理与分发的问题。那么如何实现公钥密码呢？本章介绍几个典型的公钥密钥算法，包括基于大数分解难题的 RSA 算法，基于有限域上求解离散对数难解问题的 ElGamal 算法，以及基于椭圆曲线上求解离散对数难解问题的 ECC 算法。通过本章的学习，读者应该掌握以下内容:

- RSA 公钥密码算法及其用于加密和签名的实现。
- ElGamal 公钥密码算法及其用于加密和签名的实现。
- ECC 公钥密码算法及其用于加密和签名的实现。
- 公钥密码算法的设计基本方法和安全性原理。

5.1　RSA 公钥密码算法

如何实现加密密钥公开的加密算法?

RSA 是一个典型公钥密码（Public-key Cryptography），以当时在 MIT 的提出者 Rivest、Shamir 和 Adleman 三个人的名字命名，于 1978 年公开描述。RSA 既可以用于加密也可以用于签名，广泛用于电子商务安全系统中，当给定足够长度的密钥时，RSA 被认为是安全的。

5.1.1　RSA 基本算法

RSA 包括公钥和私钥两个密钥，公钥可以让任何人知道并用于加密消息，使用公钥加密的消息只能使用对应的私钥解密，使用 RSA 加密算法实现完整的公钥加密系统包括密钥产生、加密和解密三个步骤，以用户 A 与 B 两个参与者为例，加密系统如表 5-1 所示。

表 5-1　RSA 加密系统

RSA 加密系统一般描述	RSA 加密示例
密钥生成 用户 A 做如下工作： ● 选择 2 个素数 p 和 q。为了安全考虑，p 和 q 应该一致地随机选择，并具有相似的比特长度 ● 计算 $n=pq$。n 用于公钥和私钥的模运算 ● 计算欧拉函数 $\varphi(n)=(p-1)(q-1)$ ● 选择一个整数 e，满足 $1<e<\varphi(n)$，且 e 与 $\varphi(n)$互素，即 e 与 $\varphi(n)$除了 1 没有其他公因子。e 作为公钥指数发布	**密钥生成** $p=61$，$q=53$ $n=p\times q$，$n=61\times 53=3233$ $\varphi(n)=(61-1)\times(53-1)=3120$ 选择 $\varphi(n)$互素 e，可以选择一个素数，如 $e=17$，并判断是否整除 3120

续表

RSA 加密系统一般描述	RSA 加密示例
● 计算 d，满足全等关系 $de \equiv 1 \bmod \varphi(n)$，换句话讲，$ed-1$ 只能被 $\varphi(n)$整除。通常使用扩展欧几里得算法，可以快速计算 d。d 作为私钥指数保密 通常公钥记为(n,e)，私钥记为(n,d) ● A 发布公钥(n,e)	计算 d，实际上计算模 $\varphi(n)$的 e 的乘法逆元因为 17×2753=46801，46801%3120=1，即 46801 除 3120 余 1。d=2753 即公钥为(n=3233, e=17)， 私钥为(n=3233,d=2753)
加密 用户 B 加密消息 M 发送给用户 A，做如下工作： ● 将消息 M 转换为一个整数 m（0<m<n） ● 计算 $c \equiv m^e \pmod n$	**加密** m=123 c= $123^{17} \bmod 3233 = 855$
解密 用户 A 收到消息 c，做如下工作： ● 计算 $m \equiv c^d \pmod n$ 给定 m，A 能够恢复原来格式明文 M	**解密** $m = 855^{2753} \bmod 3233 = 123$

从下列关系可以看出上述过程可以正确解密密文：

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod n$$

又因为 $ed = 1 + k\varphi(n)$，当 m 与 n 互素时，直接使用欧拉定理，有：

$$m^{ed} \equiv m^{1+k\varphi(n)} \equiv m(m^{\varphi(n)})^k \equiv m \pmod n$$

因此可以正确解密。

5.1.2　RSA 加密算法的数论基础

这里我们需要弄清两个问题：第一个问题是 RSA 算法看似非常简单，它为什么能正确加解密？第二个问题是为什么 RSA 是安全的，即没有私钥，解密消息是困难的？

为了解答第一个问题，首先了解 RSA 用到数论中的一些知识。

定义 1（同余）：设 a、b、m 是正整数，如果 $m|(a-b)$，即 m 整除$(a-b)$，则称 a 和 b 模 m 同余，记为 $a \equiv b(\bmod m)$。

定理 1（素数分解定理）：对任意正整数 n，存在唯一的正素数序列 $p_1 < p_2 < ..... < p_m$，以及正整数 $\alpha_1, \alpha_2, ... \alpha_m$，使得 $n = p_1^{\alpha_1} p_2^{\alpha_2} ..... p_m^{\alpha_m}$。

定义 2（欧拉数）：设 n 是一个正整数，$\varphi(n) = |\{x \mid 1 \leqslant x \leqslant n-1, \gcd(x,n) = 1\}|$，即小于 n 的与 n 互素的正整数，称为欧拉数。

特别地，当 p 是一个素数时，则 $\varphi(p) = p - 1$。

定理 2：如果 n_1 和 n_2 互素，则 $\varphi(n_1 \times n_2) = \varphi(n_1)\varphi(n_2)$。

定理 3：如果一个正整数 n 按定理 1 分解并表示为 $n = p_1^{\alpha_1} p_2^{\alpha_2} ..... p_m^{\alpha_m}$，则 $\varphi(n) = n(1-1/p_1)(1-1/p_2).....(1-1/p_m)$。

例 1　求 $\varphi(30)$。

解：30=2×3×5

方法 1：∵ 2、3、5 互素

∴ $\varphi(30) = \varphi(2)\varphi(3)\varphi(30) = (2-1)(3-1)(5-1) = 8$

方法 2：$\varphi(30) = 30(1-1/2)(1-1/3)(1-1/5) = 8$

与 30 互素的数构成一个模 30 乘法群 $Z_{30}^{*}=\{1,7,11,13,17,19,23,29\}$，即 $|Z_{30}^{*}|=\varphi(30)=8$。

定理 4（Euler 定理，欧拉定理）：设 x 和 n 都是正整数，如果 $\gcd(x,n)=1$，则 $x^{\varphi(n)}\equiv 1(\bmod n)$。（证明略）。

例 2 使用 Z_{30}^{*} 验证欧拉定理。

解：$Z_{30}^{*}=\{1,7,11,13,17,19,23,29\}$，$|Z_{30}^{*}|=\varphi(30)=8$

有：$7^8 \bmod 30=5764801 \bmod 30=1$

实际上利用模运算的性质，有 $7^8 \bmod 30=(7^4)^2 \bmod 30=(2401)^2 \bmod 30=1$。

$11^8 \bmod 30=(11^2)^4 \bmod 30=(121)^4 \bmod 30=1$

$23^8 \bmod 30=(23^2)^4 \bmod 30=(19)^4 \bmod 30=1$

Z_{30}^{*} 中的其他元素读者可自行计算验证。

利用 Euler 定理可以计算乘法群中元素的逆元，如求 x 的逆。因为 $x^{\varphi(n)}\equiv 1(\bmod n)$，所以 $x\times x^{\varphi(n)-1}\equiv 1(\bmod n)$，因此有 $x^{-1}\equiv x^{\varphi(n)-1}(\bmod n)$。

例 3 求 Z_{30}^{*} 中元素的逆。

解：$Z_{30}^{*}=\{1,7,11,13,17,19,23,29\}$，我们以 7、13 和 19 为例计算它们的逆。

$7^{-1}\equiv 7^{\varphi(30)-1} \bmod 30=7^7 \bmod 30=13$

$13^{-1}\equiv 13^{\varphi(30)-1} \bmod 30=13^7 \bmod 30=7$

$19^{-1}\equiv 19^{\varphi(30)-1} \bmod 30=19^7 \bmod 30=19$

可见 7 与 13 是互为逆，而 19 的逆是它自身。其他的元素逆读者可自行计算。

推论 1（Fermat 定理，费马定理，Euler 定理推论）：设 x 和 p 都是正整数。如果 p 是素数，且 $\gcd(x,p)=1$，则 $x^{p-1}\equiv 1(\bmod p)$。

定理 5（Fermat 小定理）：设 x 和 p 都是正整数。如果 p 是素数，则 $x^p\equiv x(\bmod p)$。

易得，如果 p 是素数，则 $x^{-1} \bmod p\equiv x^{p-2}(\bmod p)$。

上述欧拉定理及费马小定理是RSA加密体制的数学依据，下面看一下RSA的正确性证明。

RSA 解密过程正确性证明：

证明：解密过程是加密过程的逆变换，即 $(m^e)^d=m(\bmod n)$。

因为 $de\equiv 1(\bmod \varphi(n))$，故存在整数 $t\geqslant 1$，使得 $de=t\times\varphi(n)+1$。

那么，对于任意明文 m，$n>m\geqslant 1$，有：

（1）当 $\gcd(m,n)=1$ 时，直接根据 Euler 定理，有：

$$(m^e)^d\equiv m^{t\varphi(n)+1}(\bmod n)\equiv(m^{t\varphi(n)}m)(\bmod n)\equiv 1^t m(\bmod n)\equiv m(\bmod n)$$

（2）当 $\gcd(m,n)\neq 1$ 时，

因为 $n=pq$，且 p 和 q 都是素数，所以 $\gcd(m,n)$ 一定为 p 或 q。

不妨设 $\gcd(m,n)=p$，即 m 是 p 的倍数，设 $m=cp$，$1\leqslant c\leqslant q$。

根据 Fermat 定理，$m^{q-1}\equiv 1(\bmod q)$，

所以有，$(m^{q-1})^{t(p-1)}\equiv 1(\bmod q)$，即 $m^{t\varphi(n)}\equiv 1(\bmod q)$。

于是存在一个整数 s，使得 $m^{t\varphi(n)}=sq+1$，

上式两端同时乘以 $m=cp$，有 $m^{t\varphi(n)+1}=msq+m=cpsq+m=csn+m$，

因此，$(m^e)^d \equiv m(\mathrm{mod}\, n)$。

综上所述，对于任意 $m \in Z_n$，都有 $(m^e)^d \equiv m(\mathrm{mod}\, n)$。

因此，RSA 能够正确解密加密的明文。

对于前面提到的第二个问题，即找出 RSA 对应的数学难解问题。RSA 密码系统的安全性依赖两个数学问题：大数分解问题和 RSA 问题。由于目前尚无有效的算法解决这两个问题的假设，已知公钥解密 RSA 密文是不容易的。

RSA 问题定义为：给定一个模 n 运算下某个根的 e 次方，计算一个值 m，使得 $c=m^e \bmod n$，其中(n,e)是 RSA 公钥，c 是 RSA 密文。当前认为可以解决 RSA 问题的方法是分解 n 的因子，若可以分解出素因子 p 和 q，则攻击者可以计算 $\varphi(n)$，进而从公钥(n,e)计算秘密的指数 d。目前尚未发现在传统计算机上可以在多项式时间内分解大整数的有效方法。2008 年报道，应用分布式系统，使用通用因子分解算法（General-purpose Factoring）能够分解的最大整数为 663 比特。目前普遍认为 RSA 的 n 长度至少取 1024 比特，建议取 2048 比特，因此普遍相信当 n 足够大时 RSA 算法是安全的。

5.1.3　RSA 算法实现中的计算问题

1. 素数产生

RSA 算法实现过程中需要随机选取 2 个大素数，一般随机选择 2 个数再判断它们是否为素数。最基本的方法是给定一个数 n，判断是否存在 2～(n−1)（实际上测试到 $\sqrt{n}$ 即可）之间的数能够整除 n，存在，则 n 不是素数，否则 n 是素数。显然这种素性判断效率低下。

实际中采用概率测试方法，选取特定比特长度的随机数，通过多次迭代进行概率素性测试。最简单的方法如 Fermat 素性测试。其过程如下：

给定整数 n，选择与 n 互素的整数 a，计算 $a^{n-1} \bmod n$，如果结果不是 1，则 n 是合数；若结果是 1，n 可能是素数，也可能不是素数（回顾 Fermet 定理）。Fermat 素数测试是一种启发式测试，无论如何选取证据，总有一些合数（也称 Carmichael 数，如 561、1105、1729 等）将被作为“概率素数”。不过，该算法在 RSA 密钥生成阶段有时仍作为快速产生素数的方法使用。

更复杂的还有 Miller-Rabin 素性测试和 Solovay-Stassen 素性测试算法，对于任意合数 n，至少 3/4（Miller-Rabin）、1/2（Solovay-Stassen）的 a 可以作为 n 是合数的证据。因此，也称这些算法为合数测试。

Miller-Rabin 素性测试算法基本方法：给定整数 n，选择一些整数 $a<n$，令 $2^s d = n-1$，d 为奇数。对于所有的 $0 \leqslant r \leqslant s-1$，如果 $a^d \not\equiv 1 \ (\mathrm{mod}\, n)$，而且 $a^{2rd} \not\equiv -1 \ (\mathrm{mod}\, n)$，则 n 是合数；否则 n 可能是素数，也可能不是素数。通过多次迭代，提高判定 n 是素数的概率。

注：上述算法用到二次探测定理，如果 p 是一个素数，0<x<p，则方程 x^2≡1(mod p)的解为 x=1 和 p-1。

Solovay-Stassen 素性测试算法：给定一个奇数 n，选择一些整数 $a<n$，如果 $a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \ (\mathrm{mod}\, n)$，其中 $\left(\frac{a}{n}\right)$ 是亚克比符号（Jacobi symbol），则 n 是合数，而 a 是证据。否则 n 可能是素数，也可能不是素数。

此外，还有一些已经证明了的确定素性测试方法，但一般计算量较大。实际应用中，可

以先采用概率素数测试算法去除合数，再通过确定素性测试方法进一步证明所选整数是素数。当然，通过若干次迭代概率素性测试方法，可以使通过测试的整数是合数的概率降到很小（可忽略概率），从而实现快速选择素数的目的。

选择 p 和 q 应该尽量接近，防止 Fermat 因数分解算法成功分解 n。进一步地，若 $p-1$ 和 $q-1$ 仅有小素因子，通过 Pollard 算法也可以快速分解 n，因此选取素数时应测试这一属性，丢弃满足这一条件的素数。

此外，私钥 d 应该足够大，Michael 等指出，若 p 在 q 和 $2q$ 之间，且 $d<n^{1/4}/3$，则从 n 和 e 可以有效计算 d。当使用适当的填充机制（参见 5.1.5）时，即使选择小公钥指数，如 $e=3$，也没有对 RSA 的有效攻击；但不采用填充机制或填充不当，会导致遭受较大攻击风险。通常 e 会选取 65537，这一数值认为是避免小指数攻击和实现高效加密（即高效签名验证）的合理折中。NIST 在 2007 年发布的《计算机安全专刊 SP 800-78 Rev 1》中要求 e 选择要大于 65537。

此外，考虑解密效率，RSA 密钥对产生后，通常可以作为私钥的一部分保存下列内容：p 和 q、$d \bmod (p-1)$和 $d \bmod (q-1)$，以及 $q^{-1} \bmod (p)$。

2. 模指数运算

RSA 加密和解密算法均是在模 n 下进行指数运算，下面性质可以提高算法实现的效率：

（1）模幂运算满足分配律。

$$[(a \bmod n)\times(b \bmod n)] \bmod n = (a\times b) \bmod n$$

利用中间结果对 n 取模，既降低了存储要求，又可实现高效算法。

（2）递进式指数计算。

例如，计算 $a^d \pmod n$，为了便于计算机实现，其中指数 d 可以表示为 k 比特二进制 $(b_{k-1}b_{k-2}...b_1b_0)_2$，因此，$d$ 可以记为：

$$d = \sum_{b_i \neq 0} 2^i$$

因此，有：

$$a^d \bmod n = [\prod_{b_i \neq 0} a^{(2^i)}] \bmod n = [\prod_{b_i \neq 0} a^{(2^i)} \bmod n]$$

例如，计算 13^{19}，可以表示为 $13^{19}=13^{16}\times13^2\times13=((((13)^2)^2)^2)^2\times13^2\times13$。

而 19 的二进制表示为$(10011)_2$，从指数 19 的二进制低比特位到高比特位扫描，累乘对应比特位为 1 的指数值即可，可以快速完成指数计算。根据这一性质，可以采用迭代或递归算法实现快速指数计算。迭代指数计算算法描述如下：

```
int exp( int a, int d, int p )
{   /*计算 a ^ d mod p   */
   int s=1;
   while ( d ) {
      if ( d % 2 ) {
         s = (s * a) mod p ;
   }
   d = d / 2 ;
      a = a*a mod p ;
   }
   return ( s ) ;
}
```

3．私钥求解

在 RSA 算法中选择一个公钥指数 e，然后需要计算私钥指数 d，使得 $ed \equiv 1 \bmod \varphi(n)$，即找到一个最小整数 d，使得 ed 除以 $\varphi(n)$余数等于 1，或者表示为 $ed=1+k\varphi(n)$。通常采用扩展欧几里得算法快速计算 d。扩展欧几里得算法可以表示为：$ax+by = \gcd(a,b)$，$\gcd(a,b)$表示 a 和 b 的最大公约数。

即求解 x 和 y，使得上面等式成立。对应地，求解式子 $ed+(-k)\varphi(n) =1$ 中的 d 和$-k$。

扩展欧几里得算法的递归形式（假定 a>b）：

```
int ExGCD(int a,int b,int &x,int &y)
{
  if (b==0) {
     x=1;
     y=0;
     return a;
  }
  int r=ExGCD(b,a%b,x,y)
  int temp=x;
  x=y;
  y=temp-(a/b)*y;
  return r;
}
//用扩展欧几里得算法解线性方程 ax+by=c
bool linearEquation(int a,int b,int c,int& x,int &y)
{
     int d=Extended_Euclid(a,b,x,y);
     if(c%d) return false;
     int k=c/d;
     x*=k;   y*=k;     //求的只是其中一个解
     return true;
}
```


5.1.4　RSA 体制安全性分析

对 RSA 加密体制的一些典型攻击方法如下：

（1）穷举攻击。即列出所有可能的私钥，显然这是缺乏效率和困难的。

（2）因数分解攻击。给定某个整数 $c \equiv m^e \pmod n$，求 c 的模 n 的 e 次方根 $m \equiv c^{1/e} \pmod n$ 是一个困难问题，但如果已知整数 n 的素数分解，则上述问题易解。因此，因数分解攻击 RSA 的途径包括：

- 分解 n 为 p 和 q。
- 直接确定 $\varphi(n)$，而不确定 p 和 q。
- 直接确定 d，而不确定 $\varphi(n)$。

可以证明，从 e 和 n 确定 $\varphi(n)$ 或者 d 的算法至少和因数分解一样费时。因此，目前将因

数分解作为评价 RSA 安全性的基础。

（3）参数选取不当造成的攻击。选取 p 和 q 时应该是随机的且不应太接近。因为，$n=pq=(p+q)^2/4-(p-q)^2/4$，当$(p-q)/2$很小时，那么$(p+q)/2$只比$\sqrt{n}$大一点，因此逐个检查大于$\sqrt{n}$的整数 x，使得x^2-n是一个完全平方数，记为 y^2，那么就有 $p=x+y$ 和 $q=x-y$。

（4）选择密文攻击。攻击者得到两个明密文对(m_0,c_0)、(m_1,c_1)，则可以获得$m_0\times m_1$的密文结果，因为$m_0^e m_1^e\equiv(m_0m_1)^e \pmod n\equiv c_0c_1(\bmod n)$。

由明密文对(m_0,c_0)，可以获得对m_0^r的加密结果，因为$(m_0^r)^e\equiv c_0^r(\bmod n)$。

此外，能够获得$(c\times 2^e)(\bmod n)$的解密结果，就可以恢复出 c 对应的明文。

（5）共模攻击。通信系统中使用相同的 n，且存在两个用户的公钥 e_1 和 e_2 互素，则可以由两个用户对同一个明文的不同加密结果恢复出原始明文。

设$c_1\equiv m^{e_1}(\bmod n)$、$c_2\equiv m^{e_2}(\bmod n)$，若攻击者获得这两个密文，根据中国剩余定理推论：

存在 s、t，使得$t\times e_1+s\times e_2=1$，使得$c_1^t\times c_2^s\equiv m(\bmod n)$，从而恢复出明文。

（6）小 e 攻击。采用很小的加密指数，如 e=3，加密值很小的消息 m，如 $m<n^{1/e}$，即 m^e 远小于模数 n，这种情况下，直接计算 e 的指数，密文很容易被破解。

此外，若多个人使用相同的 e=3，但彼此 n 不同，设有 3 个人分别使用 n_1、n_2、n_3，若他们加密相同消息 m，即有：

$$c_1\equiv m^3(\bmod n_1)、\ c_2\equiv m^3(\bmod n_2)、\ c_3\equiv m^3(\bmod n_3)，$$

因为一般情况下 n_1、n_2、n_3 是互素的，使用中国剩余定理可得：

$$c\equiv m^3(\bmod n_1\times n_2\times n_3)，$$

又因为 m< n_1,n_2,n_3，所以 $m^3<n_1\times n_2\times n_3$，则 $m=\sqrt[3]{c}$。

此外还有计时攻击，即利用指数中某一比特位为 1 或 0 时，硬件加密速度不同反映出来的差异进行分析。

5.1.5 RSA 填充加密机制

分析上面列举的一些 RSA 攻击方法可见，在有些情况下，基本的 RSA 加密是不安全的。这些攻击方法使用已知明/密文解密其他未知的但相关联的密文（选择密文攻击），或者利用对相关明文加密的密文分析明文（如小指数攻击、共模攻击）。

分析基本 RSA 算法，相同明文使用相同密钥加密，得到相同密文，相同明文使用不同密钥加密，保持了模乘的一些性质。这就是所谓“确定加密算法”，加密过程中没有随机成分，相同的明文加密后得到相同的密文，这样通过尝试加密某些明文，测试是否与特定密文一致，攻击者可以成功实施选择明文攻击。

那么是否能够实现即使使用同一个密钥，相同明文在不同时候加密得到不同密文？即能否对明文混入随机性。这就是密码学中给出的强安全定义——语义安全（Semantically Secure），即使知道明文，攻击者仍无法区分出对应的密文。基本的 RSA 算法不是语义安全的。

为了避免上述攻击的发生，RSA 才采用随机填充机制加密消息，避免消息 m 落入不安全明文区域，并且给定一个 m，不同的随机填充产生不同的输出。

RSA 实施标准如 PKCS#1 精心设计了安全填充，由于填充需要额外加入信息，要求填充

前消息 M 要小一些（保证填充后的 m<n）。PKCS#1v1.5 定义了优化非对称加密填充 OAEP（Optimal Asymmetric Encryption Padding）机制，该机制由 Bellare 和 Rogaway 提出。OAEP 算法也采用 Feistel 网络结构形式，使用一对随机数 G 和 H，结合单向陷门函数 f，该处理模式被证明在随机预示模型下是选择明文攻击语义安全的 IND-CPA（Indistinguishable Under Chosen Plaintext Attack），选择特定陷门函数 f，如 RSA，OAEP 被证明是抗选择密文攻击的。OAEP 满足以下目标：

（1）添加随机元素将确定密码机制（如基本 RSA）转换为一个概率机制。

（2）部分密文解密（或其他信息泄露），只要不能翻转单向陷门函数，攻击者仍不能解密任何密文部分。

OAEP 工作过程如图 5-1 所示。其中 n 是 RSA 的模数，k_0 和 k_1 是 OAEP 协议定义的两个表示比特长度的整数，m 表示明文消息，长度为 $(n-k_0-k_1)$ 比特，G 和 H 为 OAEP 协议定义的两个密码学哈希函数。

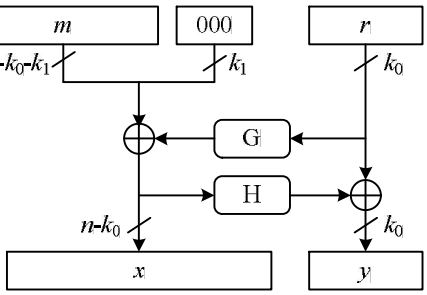


图 5-1　OAEP 工作流程

加密过程如下：

（1）明文 m 后面填充 k_1 个 0。

（2）产生 k_0 比特随机数 r。

（3）使用 G 将 k_0 比特随机数 r 扩展为 $(n-k_0)$ 长度比特串。

（4）计算 $x=m00\ldots0 \oplus G(r)$。

（5）使用 H 将 $(n-k_0)$ 长度 x 压缩为长度 k_0 比特串。

（6）计算 $y=r \oplus H(x)$。

（7）最后输出 $x||y$。

实际上从 RSA 加密的明文块即$(x||y)$，可以看出，由于填充时使用了随机数 r，因此，即使两次加密相同 m，选择不同 r，填充后得到不同比特串$(x||y)$，加密后密文也不同。

从图中可以看出，变换结构是可逆的，解密操作如下：

（1）恢复随机串 $r=y \oplus H(x)$。

（2）恢复消息 $m00\ldots0 = x \oplus G(r)$。

5.1.6　RSA 签名算法

RSA 算法可以直接用于数字签名，密钥对持有者使用自己的私钥对消息摘要签名，验证者使用签名者的公钥进行验证。过程如下：

签名者密钥参数包括 e、d、n。做以下操作：

（1）对于待签名消息 m，使用 Hash 函数 h 生成消息摘要，即 h(m)。

（2）使用私钥对消息摘要签名，得到 $s \equiv (h(m))^d \pmod{n}$ 。

（3）验证方接收到消息 m 和签名 s，计算消息摘要 $h(m)$；并使用签名者公钥恢复 $v \equiv s^e \pmod{n}$，比较 v 与 h(m)，若二者相等，签名验证通过；不相等，则签名验证不通过。

5.2 Diffie-Hellman 密钥协商机制

能否在不安全的通信信道上传输一些公开信息，最终使得双方获得秘密信息？实际中存在这样的例子，两个好朋友通过网络可以聊一些公开的话题，但其中隐含着只有他们两个人知道的信息，如两个人上一周共同逛商场买的一件衣服，两个人共同使用一位好朋友的姓名（他们使用外号称呼），这样的秘密信息可以作为两个人共享的秘密加密其他消息。

Whitfield Diffie 和 Martin Hellman 于 1976 年提出了一种密钥协商机制（也称 D-H 协议），可以实现在不安全信道中为两个实体建立一个共享秘密，协商的秘密可以作为后续对称密码体制的密钥使用。D-H 密钥协商协议如图 5-2 所示。

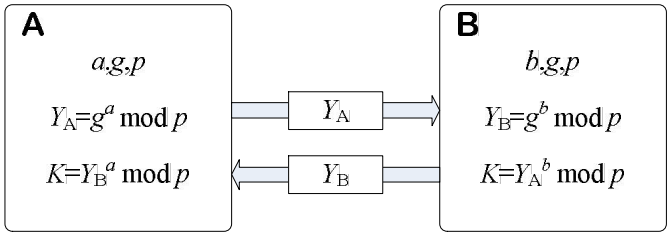


图 5-2 D-H 密钥协商协议

如图所示，用户 A 与 B 通过公开信道协商一个秘密密钥 K，协议的一般描述及示例描述如表 5-2 所示。

表 5-2 D-H 密钥协商协议描述及实例

D-H 密钥协商协议一般描述	D-H 密钥协商协议示例
● A 与 B 同意一个有限循环群 G 及其一个生成元 g。群的阶为素数 p	模素数 p=23，原根 g=5
● A 选择一个随机自然数 a，计算 ($Y_A=g^a \bmod p$)并发送给 B	a=6，$Y_A=g^a \bmod p=5^6 \bmod 23=8$
● B 选择一个随机自然数 b，计算 ($Y_B=g^b \bmod p$)并发送给 A	b=15，$Y_B=g^b \bmod p=5^{15} \bmod 23=19$
● A 计算会话密钥 $K=Y_B{}^a \bmod p$	$K=Y_B{}^a \bmod p=19^6 \bmod 23=2$
● B 计算会话密钥 $K=Y_A{}^b \bmod p$	$K=Y_A{}^b \bmod p=8^{15} \bmod 23=2$

A 与 B 能够计算一个共享秘密值 K，基本原理如下式所示：

$$K = Y_A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = Y_B^a \bmod p$$

协议中使用了模 p 的整数乘法群，其中 p 是素数，g 是模 p 的一个原根。p 和 g 可以事先

在用户 A 和 B 间协商共享或由一方产生告知另一方。

从协议中可以看出，攻击者为了计算秘密值 K，必须知道 a 或 b（有时也称为 A 和 B 的私钥），从公开传递的值 Y_A 和 Y_B 求解 a 和 b 是困难的，即求解离散对数问题目前仍然是数学难解问题。因此，当 G 和 g 选取适当时，D-H 协议被认为是安全的。攻击者只有解决“D-H 问题”才能获得秘密值 g^{ab}，目前认为这是困难的。必须存在有效的求解离散对数算法，才能够容易求解 a 和 b，从而求解 D-H 问题，但目前没有有效方法。从安全角度考虑，G 的阶 p 应该选取 $p-1$ 具有大素因子的素数。

当然基本的 D-H 协议存在典型的中间人攻击，攻击实例如表 5-3 所示，其中 E 代表攻击者。

表 5-3　D-H 协议中间人攻击

（1）A 选取随机数 a，计算 Y_A 并发送给 B。
（2）E 截获 Y_A，选取整数 e，计算 $Y_E=g^e \bmod p$，冒充 A 将 Y_E 发送给 B。
（3）B 选取随机数 b，计算 Y_B 并发送给 A。
（4）E 截获 Y_B，冒充 B 将 Y_E 发送给 A。
（5）A 计算：$K1=(Y_E)^a \bmod p=g^{ae} \bmod p$。
（6）B 计算：$K2=(Y_E)^b \bmod p=g^{be} \bmod p$。
（7）E 计算：$K1=(Y_A)^e \bmod p=g^{ae} \bmod p$，$K2=(Y_B)^e \bmod p=g^{be} \bmod p$。

至此，E 获得与 A 共享的密钥 $K1$，以及与 B 共享的密钥 $K2$。A 则认为与 B 共享了秘密密钥 $K1$，而 B 认为与 A 共享了秘密密钥 $K2$。之后 E 在 A 与 B 之间截获他们互发的消息，解密并转发，从而获得 A 与 B 之间的保密通信的消息。

需要注意的是，攻击者 E 并不能计算 $g^{ab} \bmod p$，E 只是分别与 A、B 共享了不同密钥。当然，实际应用中，E 必须能够插入到 A 与 B 之间，并阻止 A 与 B 发送的消息直接到达对方，这在实现上是需要一定条件的。如 E 能够控制 A 与 B 之间通信的一台路由器，能够实时监控、分析并转发流量，从而适时地实现中间人攻击。

之所以 D-H 存在典型的中间人攻击，是因为基本的 D-H 协议缺乏实体认证（或称鉴别），因此，实际应用 D-H 协议时，一般混合鉴别机制。

5.3　ElGamal 公钥密码体制

基于离散对数难解问题设计的公钥密码算法。

5.3.1　ElGamal 加密算法

ElGamal 公钥加密算法是类似于 D-H 密钥协商机制的一种公钥密码，是于 1985 年由 Taher Elgamal 提出的，在 PGP 等密码系统得到实际应用。后来也有许多变形算法，统称 ElGamal 类公钥加密体制。

ElGamal 公钥加密算法定义在任意循环群 G 上，其安全性依赖 G 上特定问题相关的计算

离散对数问题。ElGamal 公钥加密系统包括密钥产生、加密和解密三个算法，以用户 A 与 B 两个参与者为例，加密体制如表 5-4 所示。

表 5-4　ElGamal 加密体制

ElGamal 加密体制一般描述	示例
密钥生成	**密钥生成**
用户 A 做如下工作：	
● 产生一个阶为 p、生成元为 g 的循环群 G。p 和 g 可以在一组用户中共享	p=2357，g=2
● 从集合{1,2,…,p-2}中选择一个随机数 x	x=1751（私钥）
● 计算 $y=g^x \pmod p$	$y=2^{1751} \bmod 2357=1185$
● 公布公钥(G,p,g,y)，对应私钥为 x	公钥(p=2357, g=2, y=1185)
加密	**加密**
用户 B 加密消息 M 发送给用户 A，做如下工作：	
● 将 M 表示为{0,1,…,p-1}中的一个整数 m	m=2035
● 从集合{1,2,…,p-2}中选择一个随机数 k	k=1520
● 计算 $a=g^k \pmod p$	$a=2^{1520} \bmod 2357=1430$
● 计算 $b=m\cdot y^k \pmod p$	$b=2035\times 1430^{1520} \bmod 2357=697$
● 发送密文(a,b)给用户 A	密文(a=1430, b=697)
解密	**解密**
用户 A 收到消息(a,b)，做如下工作：	
● 计算 $m=b/a^k \pmod p$	$1430^{2357-1-1751} \bmod 2357=872$
● 恢复明文 M。	$872\times 697 \bmod 2357=2035$

从下列关系可以看出如何正确解密密文：

$$b / a^{-k} \equiv b \cdot a^{-k} \equiv m \cdot g^{ak} \cdot g^{-ak} \equiv m \bmod p$$

5.3.2　ElGamal 公钥密码体制的安全性

ElGamal 加密算法安全性基于离散对数难解问题，即给定 g^a，求解 a 是困难的。首先看相关概念。

定义 1：一个群 G，如果存在一个元素 $\alpha \in G$，使得任意 $b \in G$，存在整数 i，满足 $b=\alpha^i$，则 G 是循环群，其中 α 称为 G 的生成元（本原元）。

定义 2：设 G 是一个阶为 n 的循环群，α 是 G 的生成元，$\beta \in G$，β 相对于基数 α 的离散对数是满足 $\beta=\alpha^x \pmod n$ 的唯一整数 x，$0 \leqslant x \leqslant n-1$，常将 x 记为 $\log_\alpha \beta$。

例：p=97，Z_{97}^* 是阶为 n=96 的循环群，$\alpha=5$ 为 Z_{97}^* 的一个生成元。

因为 $5^{32} \equiv 35 \pmod{97}$，所以 $\log_5 35=32$。

定义 3：设 p 是一个素数，$\alpha \in Z_p^*$，α 是一个生成元，$\beta \in Z_p^*$。已知 α 和 β，求满足 $\beta=\alpha^x \pmod p$ 的唯一整数 x，$0 \leqslant x \leqslant p-2$，称为有限域上的离散对数问题（DLP）。

关于有限域上的离散对数问题，目前还没有找到一个非常有效的多项式时间算法计算有限域上的离散对数。

一般而言，只要素数 p 选择适当，有限域 Z_p 上的离散对数问题是难解的。ElGamal 密码体制的安全性依赖于求解有限域上的离散对数问题的困难。

5.3.3　ElGamal 签名算法

同样，ElGamal 密码体制也可以用于数字签名。经典 ElGamal 签名算法如下：

（1）密钥产生。选择一个素数 p，两个小于 p 的随机数 g 和 x，计算 $y = g^x \pmod p$，则其公钥为(y,g,p)，私钥为 x，g 和 p 可以由一组用户共享。

（2）签名生成。设被签名消息为 m（可以是原始消息摘要，并表示为小于 p 的数值），首先选择一个随机数 k，k 与 $p-1$ 互质，计算：

$$a = g^k \pmod p$$

再利用扩展 Euclidean 算法求解下列方程中的 b：

$$m = ax + kb \pmod{p-1}$$

签名就是(a,b)。

（3）签名验证。验证者计算 $y^a a^b \pmod p$ 和 $g^m \pmod p$，若二者相等，签名验证通过，否则签名无效。当然也要验证$1 \leqslant a < p$。

容易看出在正确签名和传递下，上述两个式子应该是相等的：

$$y^a a^b \equiv g^{xa} g^{kb} \pmod p \equiv g^{ax+kb \pmod{p-1}} \equiv g^m \pmod p$$

ElGamal 的安全性主要依赖于 p 和 g，若选取不当则签名容易伪造，应保证 g 对于 $p-1$ 的大素数因子不可约。ElGamal 的一个不足之处是它的密文成倍扩张。

美国政府的数字签名标准 DSS（Digital Signature Standard）采用的数字签名算法 DSA（Digital Signature Algorithm）是 ElGamal 算法的演变。DSA 的主要过程：

（1）密钥产生。选取一个素数 q，选取一个素数模 p，使得 $p-1$ 是 q 的倍数（标准中对 p、q 长度有要求）。选取一个模 p 阶为 q 的生成元 g，一般采用下列方法构造：任意选取 h $(1 < h < p-1)$，计算 $g = h^{(p-1)/q} \pmod p$，若 g 等于 1，重新选取 h，直到 g 不等于 1。参数(p,q,g)可以在系统中被多用户共享。

接下来计算公私钥，随机选取 x，$0 < x < q$，计算 $y = g^x \pmod p$，则私钥为 x，而公钥为(p,q,g,y)。

（2）签名生成。令 H 是哈希函数，对消息 m 签名：产生一个消息的随机数 k，$0 < k < q$，计算 $r = (g^k \bmod p) \bmod q$，及 $s = (k^{-1}(H(m) + xr)) \bmod q$，若 r 或 s 为 0 重新计算签名，则消息签名为(r,s)。

（3）签名验证。若不满足 $0 < r < q$ 或 $0 < s < q$，则拒绝签名；否则计算 $w = (s)^{-1} \bmod q$、$u1 = (H(m)w) \bmod q$、$u2 = (rw) \bmod q$，以及 $v = ((g^{u1} y^{u2}) \bmod p) \bmod q$，如果 $v=r$ 则签名有效。

5.4 椭圆曲线密码体制

如何发现可用于公钥密码体制的代数系统?

椭圆曲线密码学（Elliptic Curve Cryptography，ECC）是一种基于有限域上的椭圆曲线代数结构的公钥密码方法，1985 年 Neal Koblitz 和 Victor Miller 分别独立提出椭圆曲线可以作为公钥密码体制的基础，其依据就是定义在椭圆曲线点构成的 Abel 加法群构造的离散对数计算困难性。美国国家安全局已将 ECC 技术包含在推荐密码算法套件中，允许使用 384 比特密钥，保护分类为最高机密级信息。

我们知道，公钥密码算法都是基于特定数学问题的难解性。由于椭圆曲线上的离散对数问题更加困难，在其密码系统中允许使用较小的密钥，如相对应 RSA 就可以得到同样强度的安全性。使用较小的密钥意味着加密或解密速度更快，同时也节省了带宽，并且在某些情况下对内存的需求也更低。

5.4.1 椭圆曲线基本概念

椭圆曲线是如下韦尔斯特拉斯（Weierstrass）方程所确定一个平面曲线：

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + b$$

其中系数 a_i（i=1,2,…,6）定义在某个域上，可以是有理数域、实数域、复数域，还可以是有限域。当系数 a_i 取不同有理数时，椭圆曲线几何图形也不同，如图 5-3 所示。

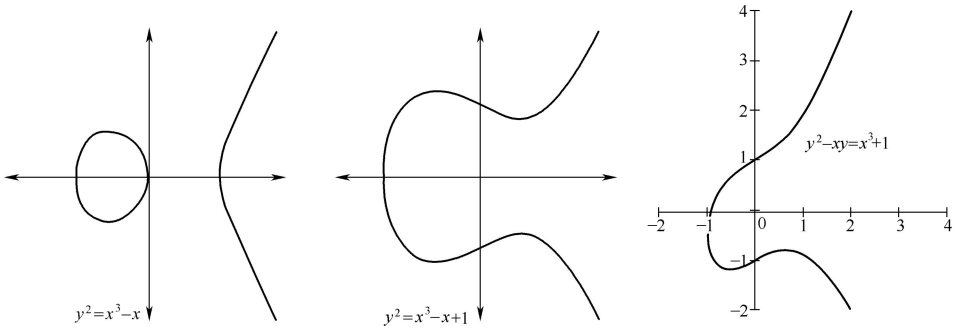


图 5-3 椭圆曲线实例

椭圆曲线可以定义在任意的有限域上，但用于公钥密码的椭圆曲线方案主要是基于 Z_p 和特征为 2 的有限域 F_{2^m}（其中，$m \geqslant 1$）。

1. 域 Z_p 上的椭圆曲线

Z_p 表示 p 个元素有限域，元素为 0～(p−1)有理数。使用一组参数确定域 Z_p 上的椭圆曲线，这也是将椭圆曲线用于密码体制所必要的。现在选择一个素域 Z_p，p 为素数（p>3），椭圆曲线参数表示为一个六元组 T：

$$T = (p, a, b, Q, n, h)$$

其中，$a, b \in Z_p$，并且满足 $4a^3 + 27b^2 \neq 0 (\mathrm{mod}\ p)$，由参数 a 和 b 定义的域 Z_p 上的一个椭

圆曲线方程为：

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

该方程所有的解（或称曲线的点）$P=(x,y)$（$x,y \in Z_p$）再加上一个无穷远点（记为 O）所构成的集合，记为 $E(Z_p)$，也可记为 $E_p(a,b)$。参数 T 确定了椭圆曲线 $E(Z_p)$ 及其上的基点。

上面定义的椭圆曲线要求是非退化的，从几何意义上讲曲线不存在尖端（奇异点）或自交叉点。

注：并不是所有的椭圆曲线都适合加密。$y^2 = x^3 + ax + b$ 是一类有限域 Z_p 上可以用来加密的椭圆曲线，也是最为简单的一类。

椭圆曲线 $E(Z_p)$ 上所有点的个数记为 $\#E(Z_p)$，根据 Hasse 定理有：

$$p+1-2\sqrt{p} \leqslant \#E(Z_p) \leqslant p+1+2\sqrt{p}$$

素数 n 是基点 Q 的阶，整数 h 是余因子，$h=\#E(Z_p)/n$。

集合 $E(Z_p)$ 定义如下的加法规则，构成一个 Abel 群，无穷远点 O 为单位元。

（1）$O+O=O$。

（2）对于所有点 $P \in E(Z_p)$，有 $P+O=O+P=P$。

（3）对于所有点 $P \in E(Z_p)$，$P=(x,y)$ 的加法逆元 $-P=(x,\text{-}y)$，即

$$P+(-P)=(x,y)+(x,-y)=O$$

（4）椭圆曲线 $E(Z_p)$ 上的两个非互为加法逆元的点 $P=(x_1,y_1)$ 和 $Q=(x_2,y_2)$，$P+Q=R=(x_3,y_3)$。

其中，

$$x_3 \equiv \lambda^2 - x_1 - x_2(\text{mod } p)$$
$$y_3 \equiv \lambda(x_1 - x_3) - y_1(\text{mod } p)$$

并且有：

如果 $P \neq Q$，$\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$

如果 $P=Q$，$\lambda = \dfrac{3x_1^2 + a}{2y_1}$（倍点运算法则，即 $2P=P+P=R$）

注：若 $\#E(Z_p)=p+1$，曲线 $E(Z_p)$ 称为超奇异的，否则称为非超奇异的。

当上述椭圆曲线 $E(Z_p)$ 用于密码体制时，为了避免一些已知攻击，选取的 p 不应该等于椭圆曲线上所有点的个数，即 $p \neq \#E(Z_p)$；并且对于任意的 $1 \leqslant m \leqslant 20$，$p^m \neq 1(\text{mod } p)$。类似地，基点 $Q=(x_q,y_q)$ 的选取应使其阶数 n 满足 $h \leqslant 4$。

例 1　在素域 Z_5 上，由方程为 $y^2 = x^3 + x + 1$ 确定的椭圆曲线 $E(Z_5)$，求曲线上的点。

解：共有 9 个点：分别是方程的解(0,1)、(0,4)、(2,1)、(2,4)、(3,1)、(3,4)、(4,3)、(4,2)，以及一个无穷远点。

例 2　在素域 Z_{23} 上，由方程 $y^2 = x^3 + x + 1$ 确定的椭圆曲线 $E_{23}(1,1)$ 上的两个点 P=(3,10)，Q=(9,7)，计算(−P)、P+Q、2P。

解：

（1）−P=(3,-10)。

（2）计算 P+Q，因为是两个不同的点，有：

斜率 λ=(y_P-y_Q)/(x_P-x_Q)=(7-10)/(9-3)=−1/2。

因为 2×12≡1 (mod 23)，即 2 的乘法逆元为 12。

λ≡−1×12 (mod 23)，即 λ=11。

$x=11^2-3-9=109 \equiv 17 \pmod{23}$

$y=11\times(3-17)-10 \equiv 11\times[3-(-6)]-10 \pmod{23} \equiv 20 \pmod{23}$

故 P+Q 的点坐标为(17,20)。

（3）计算 2P，即 P+P，有：

$\lambda=[3\times(3^2)+1]/(2\times10) \pmod{23} \equiv 1/4 \pmod{23} \equiv 6 \pmod{23}$

$x=6^2-3-3 = 30 \equiv 7 \pmod{23}$

$y=6\times(3-7)-10 = -34 \equiv 12 \pmod{23}$

故 2P 点坐标为(7,12)。

给定有限域及椭圆曲线方程，确定了一个曲线，若给出点坐标 x，可以计算有效 y 值，从而获得曲线上的点。如上面例子中的 $E(Z_{23})$ 点集合为{(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)}。有兴趣的读者可以在平面坐标系中画出这些点，看看是什么样子。此时会发现，离散的点与前面曲线样子截然不同，即椭圆曲线在不同的数域中会呈现出不同的表现形式。

2. 域 F_{2^m} 上的椭圆曲线

首先定义有限域 F_{2^m}，特征为 2 的有限域 F_{2^m} 包含有 2^m 个元素（$m \geqslant 1$），它们可以表示为：

$$a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1x + a_0$$

其中系数 $a_i \in \{0,1\}$，即或者为 0，或者为 1。

域 F_{2^m} 上的加法按如下方法定义：

设 $a,b \in F_{2^m}$，其中 a 和 b 表示为：

$$a = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1x + a_0$$

$$b = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + ... + b_1x + b_0$$

则有，a+b=c，其中 $c \in F_{2^m}$，c 表示为：

$$c = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + ... + c_1x + c_0$$

满足 $c_i \equiv a_i + b_i \pmod 2$。

域 F_{2^m} 上的乘法是按一个 m 次的不可约多项式 $f(x)$ 的形式定义的。如上述点 a 和 b，$a \cdot b = c$，则其中 c 是多项式 $a \cdot b$ 除以不可约多项式 $f(x)$ 的余项，所有的系数做模 2 运算。

元素 $a \in F_{2^m}$ 的加法逆元记为 $-a$，它是方程 $a + x = 0$ 在 F_{2^m} 中的唯一解。

元素 $a \in F_{2^m}$ 的乘法逆元记为 a^{-1}，它是方程 $a \cdot x = 1$ 在 F_{2^m} 中的唯一解。

下面看定义在有限域 F_{2^m} 的椭圆曲线，域 F_{2^m} 上的椭圆曲线域参数定义为一个七元组

$T=(m,f(x),a,b,Q,n,h)$，具体说明了椭圆曲线及其上的基点$Q=(x_q,y_q)$。其中 m 是定义域 F_{2^m} 的整数，$f(x)\in F_{2^m}$ 是一个 m 次的不可约多项式，它决定了 F_{2^m} 的表示；$a,b\in F_{2^m},b\neq 0$ 决定了由下面的方程定义的椭圆曲线 $E(F_{2^m})$：

$$y^2+xy=x^3+ax^2+b$$

且 $b\neq 0$，素数 n 是基点 Q 的阶，整数 h 是余因子 $h=\#E(F_{2^m})/n$。

上面选取的方程是域 F_{2^m} 上的非超奇异（超奇异椭圆容易受到某些攻击）椭圆曲线，是由上述方程的解（即点）$P=(x,y)$（$x,y\in F_{2^m}$）连同一个无穷远点 O 所构成的集合，记为 $E(F_{2^m})$。

$E(F_{2^m})$ 上加法运算定义如下：

（1）$O+O=O$。

（2）对于所有点 $P\in E(F_{2^m})$，有 $P+O=O+P=P$。

（3）对于所有点 $P\in E(F_{2^m})$，$P=(x,y)$ 的加法逆元 $-P=(x,x+y)$，即

$$P+(-P)=(x,y)+(x,x+y)=O$$

（4）对于 P，$Q\in E(F_{2^m})$ 是两个相异且互不为加法逆元的点，$P=(x_1,y_1)$，$Q=(x_2,y_2)$，且有 $x_1\neq x_2$，则 $P+Q=(x_3,y_3)$，其中：

$$x_3=\lambda^2+\lambda+x_1+x_2+a$$
$$y_3=\lambda(x_1+x_2)+x_3+y_1$$

如果 $x_1\neq x_2$，$\lambda=\dfrac{y_1+y_2}{x_1+x_2}$

如果 $x_1=x_2$，$\lambda=\dfrac{x_1^2+y_1}{x_1}$

$x_3,y_3,\lambda\in F_{2^m}$

（5）（倍点运算法则）对任意满足 $x\neq 0$ 的点 $P=(x,y)\in F_{2^m}$，有 $2P=P+P=(x_3,y_3)$，其中

$$x_3=\lambda^2+\lambda+a$$
$$y_3=x^2+(\lambda+1)x_3$$
$$\lambda=x_1+\frac{y_1}{x_1}$$
$$x_3,y_3,\lambda\in F_{2^m}$$

数 n 与椭圆曲线上的点 P 的乘法运算就是把点 P 和它自身相加 n 次，与前面介绍的 Z_p 上的椭圆曲线的情形一样。

椭圆曲线 $E(F_{2^m})$ 上的所有点的个数记为 $\#E(F_{2^m})$，Hasse 定理表明其满足不等式：

$$2^m+1-2\sqrt{2^m}\leqslant \#E(F_{2^m})\leqslant 2^m+1+2\sqrt{2^m}$$

为了避免对 ECC 的一些已知攻击，选取的 2^m 不应该等于椭圆曲线上的所有点的个数，即 $2^m\neq E(Z_p)$；并且对于任意的 $1\leqslant B\leqslant 20$，$2^{mB}\neq 1(\mathrm{mod}\,n)$。类似地，基点的选取应使其阶数 n 满足 $h\leqslant 4$。

3．椭圆曲线上加法运算的几何含义

上面定义了在有限域Z_p和F_{2^m}上的椭圆曲线，以及基于曲线构造 Abel 群，椭圆曲线密码体制是基于群上加法实现的，下面看一下椭圆曲线上加法运算的几何含义。

椭圆曲线上点群法则规定如下：设 P、Q 是 E 上的两个点，连接两个点得到一条直线，如果直线与曲线交叉，则得到第 3 个点（如图 5-4 所示得到点 R）；如果该直线在其中一个点与曲线相切，则该点计两次；如果直线与 y 轴平行，则定义第 3 个点为无穷远点。曲线上任何一对点都属于上述情况中一种，如图 5-4 所示。

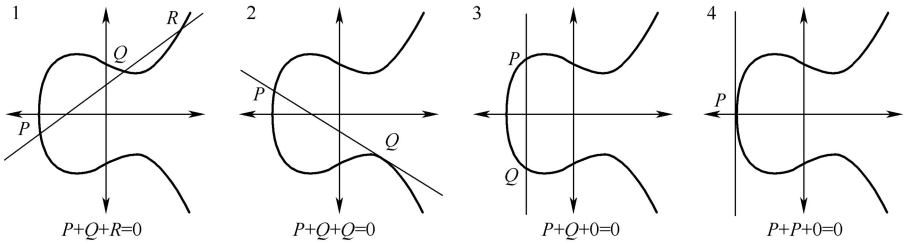


图 5-4　椭圆曲线上群法则

曲线群上加法运算法则定义如下：如果直线与曲线交于 P、Q、R 三点，则要求在群中有 $P+Q+R=O$，并有 $O=-O$，$P+O=P$。这时曲线转换为一个阿贝尔群，所有有理数点（包括无穷远点，数量记为 K）构成了上述群的一个子群；如果曲线记为 E，则子群常被记为 $E(K)$。

上述群既可以描述为代数形式，也可以描述为几何形式。从上述定义可以看出以下性质：

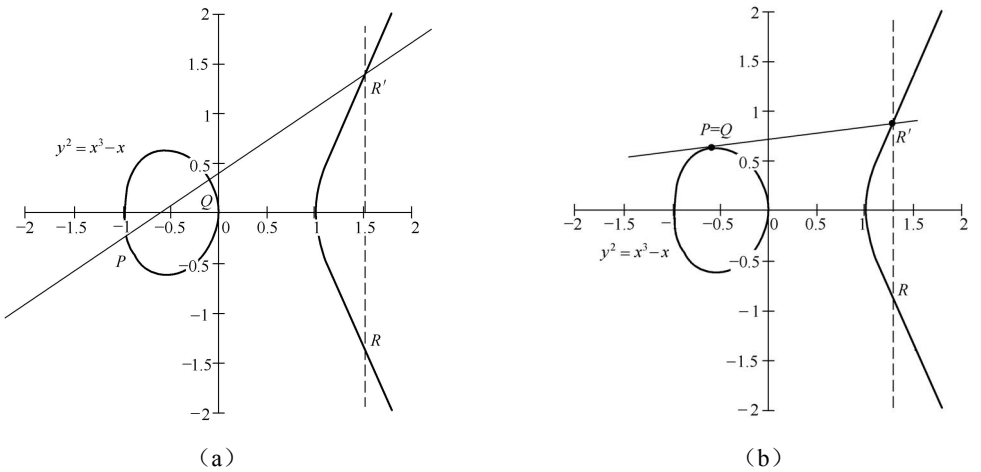
（1）曲线上三个点在一条直线上，则它们的和等于 O（无穷远点）。

（2）若曲线上有点 P，则存在一个负点，记为$-P$，$P+(-P)=O$。

（3）若一条垂直 x 轴的竖线交于曲线上两点 P、Q，则 $P+Q+O=O$，于是有 $P=-Q$。

（4）如果曲线上两个点 P 和 Q 的 x 坐标不同，则连接 P 和 Q 得到一条直线与曲线交于 R' 点，则 $P+Q+R'=O$，若 R 是 R' 的负点，则 $P+Q=-R'=R$，如图 5-5（a）所示。

（5）倍数运算，定义一个点 P 的两倍是它的切线与曲线的另一个交点 R'，则 $P+P=2P=-R'=R$，如图 5-5（b）所示。



（a）　　（b）

图 5-5　椭圆曲线上点加

从几何含义上可以看出，曲线上加法计算，即给定一个域上曲线，以及曲线上两个点 P、Q，连接 P 和 Q 得到一条直线，与曲线交于点 R'，该点相对于 x 轴对称点，即其加法逆元 R 即为 $P+Q$ 的和值。上面在有限域 Z_p 和 F_{2^m} 上定义特定的椭圆曲线都给出了计算法则。

5.4.2　基于椭圆曲线的加密体制

椭圆曲线密码体制 ECC 的依据就是定义在椭圆曲线点群上的离散对数问题的难解性。ECC 点加法等同于离散对数中的模乘，因此存在下列难解数学问题：若 P 是曲线 $E_p(a,b)$ 上的点，$Q=kP$ 也是曲线上的点，$k<p$，给定 k 和 P，容易计算出 Q，但给出 Q 和 P，计算 k 是困难的，这个问题称为椭圆曲线上点群的离散对数问题。

因此，椭圆曲线密码体制的实现依赖于椭圆曲线点的数乘。数 n 与椭圆曲线点 P 的乘法就是把点 P 与它自身叠加 n 次，$mP=P+P+\cdots+P=Q$。例如 $3P=P+P+P$，即利用椭圆曲线上点加法规则实现数乘运算。

建立椭圆曲线密码体制，给定某些椭圆曲线域参数：

$$T=(p,a,b,P.n,h) \text{ 或 } T=(m,f(x),a,b,P,n,h)$$

关于 T 椭圆曲线密钥对 (k,Q)，其中私钥为 k（k 是满足 $1\leqslant k\leqslant n-1$ 的整数），公钥 $Q=(x_Q,y_Q)$（即点 $Q=kP$）。

下面以基域 Z_p 为例，椭圆曲线方程选择为 $y^2\equiv x^3+ax+b(\text{mod } p)$，基于椭圆曲线 $E_p(a,b)$ 构造公钥密码体制。设用户 A 产生密钥对，用户 B 发送消息给 A。基于椭圆曲线的加密体制的一个实例过程：

密钥生成

用户 A 做如下工作：

- 选择 $E_p(a,b)$ 的元素 G，使得 G 的阶 n 是一个大素数，即满足 $nG=O$ 的最小 n 值。
- 选择整数 r，计算 $P=rG$。
- 公布公钥(p,a,b,G,P)，对应私钥为 r。

加密

用户 B 将加密消息 M 发送给用户 A，做如下工作：

- 将要传递的消息 M 变换为 $E_p(a,b)$ 上的一个点 P_m。
- 选择一个随机数 k，$kG\neq O$ 且 $kP\neq O$。k 可以看作为一个临时密钥。
- 计算 $Q=kG$，$R=P_m+kP$。
- 发送密文$\{Q,R\}$给用户 A。

解密

用户 A 收到消息$\{Q,R\}$，做如下工作：

- 计算 $R-rQ$，恢复明文 P_m。

从下列关系可以看出如何正确解密密文：

$$R-rQ=(P_m+kP)-r(kG)=(P_m+krG)-r(kG)=P_m$$

上述加密机制信息有扩张，即明文表示为一个点，而密文由两个点表示。

当然，具体加密、解密的实现方式不一定与上述完全一样，只要能够完成隐藏临时密钥

即可。如加密过程中，将消息 M 表示为 Z_p 域上一个元素 m，而不是曲线上一个点；这时，R 计算公式 $R=kP=(x_2,y_2)$，如果 $x_2\neq 0$，计算 $c=m\cdot x_2$，这时密文为(Q,c)。A 接收到密文，首先计算出 $R=rQ=(x_2,y_2)$，再计算 x_2 的逆元 x_2^{-1}，最后恢复明文 $m=c\cdot x_2^{-1}$。这时密文长度为一个点加一个域上元素的长度。

在密码学应用中，构造一条有限域 Z_p 上的椭圆曲线 $E_p(a,b)$，参数取值直接影响加密的安全性，一般要求参数满足以下几个条件：

（1）p 应该足够大，当然越大越安全，但越大计算速度会越慢，200 位左右可以满足一般安全要求。

（2）$p\neq n\times h$。

（3）$p^t\neq 1 \pmod n$，$1\leqslant t<20$。

（4）$4a^3+27b^2\neq 0 \pmod p$。

（5）n 为素数。

（6）$h\leqslant 4$。

5.4.3 椭圆曲线 D-H 密钥协商

椭圆曲线 D-H（Elliptic Curve Diffie-Hellman，ECDH）密钥协商协议，采用 D-H 基本方法，在两个实体间秘密协商一个会话密钥，两个实体各自拥有一个椭圆曲线公私钥对。具体过程如下：

设有两个通信实体 A 和 B，双方拥有公共参数 (p,a,b,G,n,h)。

A 的密钥对(d_A,Q_A)，B 的密钥对(d_B,Q_B)，其中 $Q_A=d_AG$，$Q_B=d_BG$，双方交换各自的公钥 Q。之后，A 计算 $d_AQ_B=(x,y)$，B 计算 d_BQ_A，则 A 和 B 获得相同值。因为：

$$d_AQ_B=d_Ad_BG=d_Bd_AG=d_BQ_A$$

5.4.4 基于椭圆曲线的数字签名算法

假设两个用户 A 和 B，用户 A 签名一个消息发送给 B。首先假设双方已协商同意公共的参数 (p,a,b,G,n,h)，A 有私钥 d_A（在[1..n−1]之间随机选取）以及相应的公钥 Q_A（$Q_A=d_AG$）。令 L_n 为群阶 n 的比特长度。用户 A 的签名消息 m，签名算法如下：

（1）计算消息散列 e=HASH(m)，令 z 是 e 的最左边 L_n 位。

（2）在[1..n−1]之间选择一个随机整数 k。计算(x,y)=kG。

（3）计算 $r=x \pmod n$，检验，如果 r=0，则返回执行步骤（2）。

（4）计算 $s=k^{-1}(z+rd_A) \pmod n$，如果 s=0，返回执行步骤（2）。

（5）签名为(r,s)。

其中，z 是消息散列的一部分，计算过程中需要转换为整数，需要注意的是，z 可以比 n 大，但不能比 n 长。

用户 B 收到签名，签名验证算法如下（设 B 已经拥有 A 的公钥 Q_A）：

（1）检测 A 公钥有效性：Q_A 不等于 O，相关参数有效；Q_A 取决于曲线；$Q_An=O$。

（2）验证 r 和 s 是[1..n−1]之间的整数，否则签名无效。

（3）计算 $w=s^{-1} \pmod n$。

（4）计算 $u=zw \pmod n$，$v=rw \pmod n$。

（5）计算$(x,y)=uG+vQ_A$。

（6）如果 $r=x \pmod n$，则签名有效，否则无效。

5.4.5　ECC 安全强度分析

一般认为，ECDSA 算法的公钥长度应该是同样安全级别比特长度的两倍。例如，安全强度为 80 比特，意味着攻击者需要 2^{80} 个签名生成发现私钥，因此 ECDSA 公钥长度应为 160 比特，而达到此安全级别，DSA 算法公钥长度至少需要 1024 比特。

RSA 算法的特点之一是数学原理简单、在工程应用中比较易于实现，但它的单位安全强度相对较低。目前用国际上公认的对于 RSA 算法最有效的攻击方法——一般数域筛（NFS）方法去破译和攻击 RSA 算法，它的破译或求解难度是亚指数级的。而现有研究结果认为，ECC 的破译或求解难度是指数级的，RSA 和 ECC 密钥体制的安全强度对比如表 5-5 所示。

表 5-5　RSA 和 ECC 安全模长比较

攻破时间（MIPS 年）	RSA/DSA 密钥长度	ECC 密钥长度	RSA/ECC 密钥长度比
10^4	512	106	5:1
10^8	768	132	6:1
10^{11}	1024	160	7:1
10^{20}	2048	210	10:1
10^{78}	21000	600	35:1

本章小结

本章介绍公钥密码体制的工作原理和具体实现方法。介绍了典型公钥密码算法，如 RSA、ElGamal，以及强度更高的椭圆曲线密码体制。描述了相关公钥密码算法的实现机理和对应密码体制的工作过程，讨论了 RSA 算法具体实现过程中素数产生、模数运算等实际问题。对各种公钥密码体制安全性进行了分析。

习题五

1．使用 RSA 加密消息，设选取的两个素数为 71 和 83，加密明文消息 1234，请选择密钥并计算密文，验证是否能正确解密。

2．解释说明 RSA 加密算法为什么能够正确解密。

3．编写一个 RSA 加密算法实现程序。

4．编写一个基于 Miller-Rabin 算法的素数产生程序。

5．在 RSA 中加密需要做模指数运算，如形式 $c=m^e \pmod n$，密码分析者获得 c 后，为什么不能尝试求出 e，即不断模乘 m?

6．编写一个程序，给定一个公钥选择 e，求私钥 d。

7．什么是 RSA 的共模攻击？如何避免这种攻击？

8．简述 RSA 签名机制的工作过程。

9．简述 Diffie-Hellman 密钥协商机制过程，设 p=37，请选择一个原根，并给出描述 DH 密钥协商机制的实现实例。

10．什么是 DH 密钥协商机制的中间人攻击？如何避免中间人攻击？

11．使用 ElGamal 加密体制，设选择 p=371，g=2，私钥 x=36。现在加密消息 m=81，试给出加密和解密过程的实例运算及过程。

12．采用 11 题的参数，现使用基本的 ElGamal 签名体制，对消息 m=81 签名，试给出签名和签名验证过程的实例运算及过程。

13．请验证 DSA 算法的正确性，即对于合法签名，签名验证算法能够验证签名的有效性。

14．在素域 Z_{23} 上由方程 $y^2 = x^3 + x + 1$ 确定的椭圆曲线 $E(Z_{23})$，求出 $E(Z_{23})$ 上所有的点。

15．已知在素域 Z_{11} 由方程 $y^2 = x^3 + x + 6$ 确定的椭圆曲线上一点 G(2,7)，求 2G 到 13G 所有的值。

16．描述椭圆曲线密码体制是如何实现加密和解密的。