应用密码学（第八讲）
— 公钥密码

林东岱

信息安全国家重点实验室

2022年9月

第 8 讲　公钥密码

8.1　公钥密码的基本概念

本世纪70年代，美国学者Diffie和Hellman，以及以色列学者Merkle分别独立地提出了一种全新的密码体制的概念。Diffie和Hellman首先将这个概念公布在1976年美国国家计算机会议上，几个月后，他们这篇开创性的论文《密码学的新方向》（“New Directions in Cryptography”）发表在IEEE杂志信息论卷上，由于印刷原因，Merk1e对这一领域的首次贡献直到1978年才出版。他们所创造的新的密码学理论，突破了传统密码体制对称密钥的概念，树起了近代密码学的又一大里程碑。

不同于以前采用相同加密和解密密钥的对称密码体制，Diffie和Hellman提出采用双钥体制的每个用户都有一对选定的密钥：一个可以是公开的，另一个则是秘密的。公开的密钥可以象电话号码一样公布，因此称为公钥密码体制或双钥体制（two-key cryptosystem）。

8.1.1 公钥密码体制的原理

公开密钥密码的基本思想是将传统密码的密钥一分为二，分为加密钥 K_e 和解密钥 K_d，用加密钥 K_e 控制加密，用解密钥 K_d 控制解密。而且由计算复杂性确保由加密钥 K_e 在计算上不能推出解密钥 K_d。这样，即使是将 K_e 公开也不会暴露 K_d，也不会损害密码的安全。于是便可将 K_e 公开，而只对 K_d 保密。由于 K_e 是公开的，只有 K_d 是保密的，所以从根本上克服了传统密码在密钥分配上的困难。

根据公开密钥密码的基本思想，可知一个公开密钥密码应当满足下面三个条件：

① 解密算法 D 与加密算法 E 互逆，即对于所有明文 M 都有，$D(E(M, K_e), K_d)=M$；

② 在计算上不能由 K_e 求出 K_d；

③ 算法 E 和 D 都是高效的。

条件①是构成密码的基本条件，是传统密码和公开密钥密码都必须具备的起码条件。

条件②是公开密钥密码的安全条件，是公开密钥密码的安全基础，而且这一条件是最难满足的。由于数学水平的限制，目前尚不能从数学上证明一个公开密钥密码完全满足这一条件，而只能证明它不满足这一条件。这就是这一条件困难的根本原因。

条件③是公开密钥密码的工程实用条件。因为只有算法 E 和 D 都是高效的，密码才能实际应用。否则，可能只有理论意义，而不能实际应用。

公开密钥密码从根本上克服了传统密码在密钥分配上的困难，利用公开密钥密码进行保密通信需要成立一个密钥管理结构（KMC），每个用户都将自己的姓名、地址和公开的加密钥等信息在 KMC 登记注册，将公钥记入共享的公开密钥数据库 PKDD。KMC 负责密钥的管

理，并且对用户是可信赖的。这样，用户利用公开密钥密码进行保密通信就像查电话号码簿打电话一样方便，再无通信双方预约密钥之苦，因此特别适合计算机网络应用。而且公开密钥密码实现数字签名容易，所以特别受欢迎。

公钥密码算法的最大特点是采用两个相关密钥将加密和解密分开，其中一个密钥是公开的，称为公开密钥，简称公开钥，用于加密；另一个密钥是为用户专用，因而是保密的，称为秘密密钥，简称秘密钥，用于解密。因此公钥密码体制也称为双钥密码体制，它有一个很重要的特性，即已知密码算法和加密密钥，求解密密钥在计算上是不可行的。

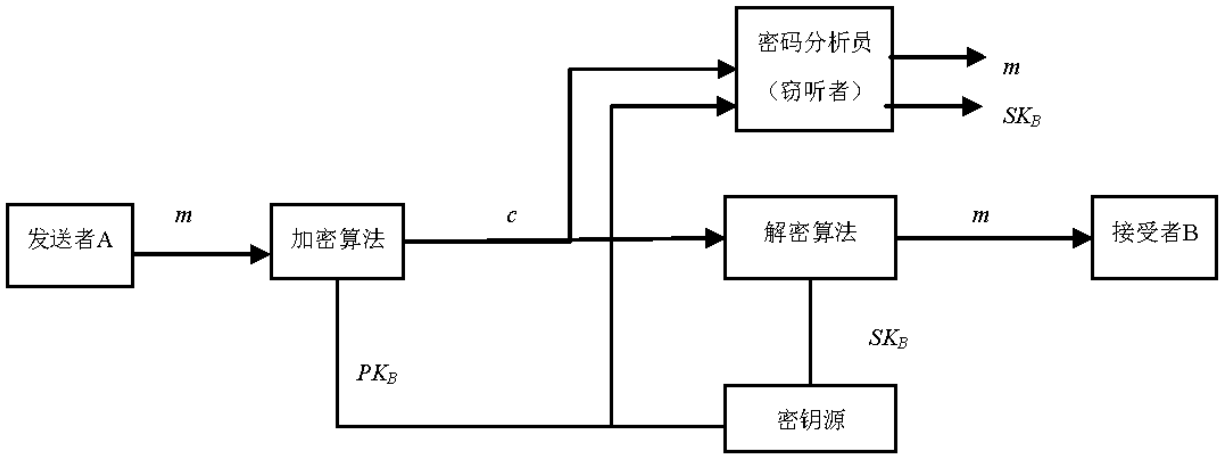

图 8.1　公钥密码算法

图 8.1 是公钥体制的框图，主要分为以下几步：

① 网络中要求接收消息的端系统，产生一对用来加密和解密的密钥，如图中的接收者 B，产生一对密钥 PK_B ，SK_B，其中 PK_B 是公开钥，SK_B 是秘密钥。

② 端系统 B 将加密密钥（图中的 PK_B）在一个公开的寄存器或文件中，另一密钥则被保密（图中的 SK_B）。

③ A 要想向 B 发送消息 m，则使用 B 的公开钥加密 m，表示为

$$c = E_{PK_B}[m],$$

其中 c 是密文，E 是加密算法。

④ B 收到密文 c 后，用自己的秘密钥 SK_B 解密，表示为

$$m = D_{SK_B}[c],$$

其中 D 是解密算法。

因为只有 B 知道 SK_B，所以其他人无法对 c 解密。

8.1.2 公钥密码算法应满足的要求

公钥密码算法应满足以下要求：

① 接收方 B 产生密钥对（公开钥 PK_B 和秘密钥 SK_B）是计算上容易的。

② 发方 A 用收方的公开钥对消息 m 加密以产生密文 c，即 $c = E_{PK_B}[m]$在计算上是容易的。

③ 收方 B 用自己的秘密钥对 c 解密，即 $m = D_{SK_B}[c]$在计算上是容易的。

④ 敌手由 B 的公开钥 PK_B 求秘密钥 SK_B 在计算上是不可行的。

⑤ 敌手由密文 c 和 B 的公开钥 PK_B 恢复明文则在计算上是不可行的。

⑥ 加、解密次序可换，即

$$E_{PK_B}[D_{SK_B}(m)]=D_{SK_B}[E_{PK_B}(m)]。$$

这一点虽然非常有用，但不是对所有的算法都作要求。

以上要求的本质在于要求一个单向陷门函数。单向陷门函数是两个集合 X、Y 之间的一个映射，使得 Y 中每一元素 y 都有惟一的一个原像 $x\in X$，且由 x 易于计算它的像 y，由 y 计算它的原像 x 是不可行的。

我们说一个函数是单向陷门函数，是指该函数是易于计算的，但求它的逆是不可行的，除非再已知某些附加信息。当附加信息给定后，求逆可在多项式时间完成。

8.1.3 对公钥密码的攻击

和单钥密码体制一样，如果密钥太短，公钥密码也易受到穷举搜索攻击。因此密钥必须足够长才能抗击穷举搜索攻击。然而又由于公钥密码所使用的可逆函数的计算复杂性与密钥长度常常不是呈线性关系，而是增大得更快。所以密钥长度太大又会使得加解密运算太慢而不实用。因此公钥密码目前主要用于密钥管理和数字签名。

对公钥密码算法的第二种攻击按是从寻找公开钥计算秘密钥的方法。目前为止，对常用公钥算法还都未能够证明这种攻击是不可行的。

还有一种仅适用于对公钥密码算法的攻击法，称为可能字攻击。例如对 56 比特的 DES

密钥用公钥密码算法加密后发送，敌手用算法的公开钥对所有可能的密钥加密后与截获的密文相比较。如果一样，则相应的明文即 DES 密钥就被找出。因此不管公钥算法的密钥多长，这种攻击的本质是对 56 比特 DES 密钥的穷举搜索攻击。抵抗方法是在欲发送的明文消息后添加一些随机比特。

8.2 RSA 密码体制

RSA 算法是 1978 年由美国麻省理工学院的三名密码学者 R.Rjvest，A.Shamir 和 L.Adleman 提出的一种用数论构造的、基于大合数因子分解困难性的公开密钥密码，也是迄今为止理论上最为成熟完善的公钥密码，简称为 RSA 密码，该体制已得到广泛应用。由于 RSA 密码既可用于加密，又可用于数字签名，安全、易懂，因此 RSA 密码已成为目前应用最广泛的公开密钥密码。

8.4.1 加密算法描述

1. 密钥的产生

① 选两个保密的大素数 p 和 q；
② 计算 $n=p\times q$，$\varphi(n)=(p-1)(q-1)$，其中 $\varphi(n)$ 是 n 的欧拉函数值；
② 选一整数 e，满足 $1<e<\varphi(n)$，且 $\gcd(\varphi(n), e)=1$；

④ 计算 d，满足 $d\cdot e=1 \bmod \varphi(n)$。即 d 是 e 在模$\varphi(n)$下的乘法逆元，因 e 与$\varphi(n)$互素，由模运算可知，它的乘法逆元一定存在；

⑤ 以{e，n}为公开钥，{d，n}为秘密钥。

2. 加密

加密时首先将明文比特串分组，使得每个分组对应的十进制数小于 n，即分组长度小于 $\log_2 n$。然后对每个明文分组 m，作加密运算：$c=m^e \bmod n$。

3. 解密

对密文分组的解密运算为：$m=c^d \bmod n$。

下面对 RSA 算法中解密过程的正确性进行证明。

证明：由加密过程知 $c=m^e \bmod n$，所以

$$c^d \bmod n \equiv m^{ed} \bmod n \equiv m^{1 \bmod \varphi(n)} \bmod n \equiv m^{k\varphi(n)+1} \bmod n。$$

不妨分两种情况讨论：

（1）m 与 n 互素，则由 Euler 定理：

$$m^{\varphi(n)} \equiv 1 \bmod n，\ m^{k\varphi(n)} \equiv 1 \bmod n，\ m^{k\varphi(n)+1} \equiv m \bmod n，$$

即 $c^d \bmod n \equiv m$。

（2）$\gcd(m，n)\neq 1$，我们先看 $\gcd(m，n)=1$ 的含义，由于 $n=p\times q$，所以 $\gcd(m，M)=1$ 意味着 m 不是 p 的倍数也不是 q 的倍数。因此 $\gcd(m，n)\neq 1$ 意味着 m 是 p 的倍数或 q 的倍数，不妨设 $m=cp$，其中 c 为一正整数。此时必有 $\gcd(m，q)=1$，否则 m 也是 q 的

倍数，从而是 $p\,q$ 的倍数，与 $m<n=p\,q$ 矛盾。

根据 $\gcd(m,q)=1$ 以及 Euler 定理可以得到 $m^{\varphi(q)}\equiv 1 \bmod q$，所以 $m^{k\varphi(q)}\equiv 1 \bmod q$，$[m^{k\varphi(q)}]^{\varphi(p)}\equiv 1 \bmod q$，$m^{k\varphi(n)}\equiv 1 \bmod q$，因此存在一整数 r，使得 $m^{k\varphi(n)}=1+r\,q$，两边同乘以 $m=c\,p$ 得 $m^{k\varphi(n)}=m+r\,c\,p\,q=m+r\,c\,n$，即 $m^{k\varphi(n)+1}\equiv m \bmod n$，所以 $c^{d} \bmod n\equiv m$。

例：设 $p=7$，$q=17$，$n=p\times q$，$\varphi(n)=(p-1)(q-1)=96$，验证 RSA 解密算法。

解：取 $e=5$，满足 $1<e<\varphi(n)$，且 $\gcd(\varphi(n),e)=1$。确定 $d\cdot e=1 \bmod 96$ 且小于 96 的 d，因为 $77\times 5=385=4\times 96+1$，所以 d 为 77，因此公开钥为{5，119}，秘密钥为{77，119}。设明文 $m=19$，则由加密过程得密文为：

$$c=19^{5} \bmod 119=2476099 \bmod 119=66$$

解密为：

$$66^{77} \bmod 119=19$$

例：设 $p=43$，$q=59$，$N=p\,q=43\times 59=2537$，$\varphi(N)=42\times 58=2436$，取 $e=13$，求 e 的逆元 d。

解：解方程 $d\,e=1 \bmod 2436$

$\because 2436=13\times 187+5$，　$13=2\times 5+3$

$5=3+2$ ，$3=2+1$

$\therefore 1=3-2,\quad 2=5-3\ ,\ 3=13-2\times 5$

$5=2436-13\times 187$

$\therefore 1=3-2=3-(5-3)=2\times 3-5$

$=2\,(13-2\times 5\,)-5=2\times 13-5\times 5$
$=2\times 13-5\times 2436$

即$937\times 13\equiv 1 \bmod 2436$，当$e=13$时，$d=937$。

若有明文public key encryptions，将明文分块为

pu bl ic ke nc ry pt io ns

再将明文数字化（令a，b，…，z 分别为00，01，…，25）得

1520 0111 0802 1004 2404
1302 1724 1519 0814 1418

利用加密可得密文

0095 1648 1410 1299 1365
1379 2333 2132 1751 1289

RSA体制不仅能实现保密通信还能实现数字签名，因而特别适用于现代密码通信的要求。

在众多的公钥密码中，RSA公钥密码被认为是比较完善的一个。自从它问世至今的20余年中，尽管有许多密码学家对它进行了长期而深入的分析，但是一直没有发现它有明显的脆弱性，所以RSA公钥密码至少有以下一些优点：

① 数学表达式简单，在公钥密码中是最容易理解和实现的一个，这个体制也是目前国际上比较流行的公钥密码之一。

② RSA的安全性基于大数分解的困难性，到目前为止除了大数分解之外，人们还没有发现一种其它的方法能够对RSA进行有效的密码分析。虽然RSA也有一些弱点，但是只要设计密码参数时仔细一点这些弱点是可以避免的，所以说RSA公钥密码的安全性比较高。

③ RSA公钥密码具有一些传统密码体制不能实现的一些功能，如认证鉴别和数字签名等，特别适合于现代密码通信。

8.4.2 RSA 算法中的计算问题

1. 加、解密

RSA 的加、解密过程都为求一个整数的整数次幂，再取模。如果按其含义直接计算，则中间结果非常大，有可能超出计算机所允许的整数取值范围。如上例中解密运算 $66^{77} \bmod 119$，先求 66^{77} 再取模，则中间结果就已远远超出了计算机允许的整数取值范围。而用模运算的性质：

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

就可减小中间结果。

再者，考虑如何提高加、解密运算中指数运算的有效性。例如求 x^{16}，直接计算的话需做 15 次乘法，即 $x^{16}=x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x\times x$。然而如果重复对每个部分结果做平方运算即求 x, x^2，x^4，x^8，x^{16} 则只需 4 次乘法。

一般，求 a^m 可按照下面的算法进行，其中 a、m 是正整数。

将 m 表示为二进制形式 $b_k b_{k-1} \ldots b_0$，即

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

因此

$$a^m = a^{\sum_{b_i\neq 0} 2^i} = \prod_{b_i\neq 0} a^{2^i} ,$$

$$a^m \bmod n = [\prod_{b_i\neq 0} a^{2^i}] \bmod n = \prod_{b_i\neq 0} [a^{2^i} \bmod n] 。$$

从而可得以下快速指数算法：

```
c=0;d=1
for i=k dowto 0 do {
        c=2×c;
        d=(d×d) mod n;
        if bi=1 then    {
                c=c+1;
```

```
            d＝(d×a) mod n
                }
        }
    return d
```

其中 d 是中间结果，d 的终值即为所求结果。c 在这里的作用是表示指数的部分结果，终值即为指数 m，c 对计算结果无任何贡献，算法中完全可将之去掉。

例：求 $7^{560} \bmod 561$

解：将 560 表示为 1000110000，算法的中间结果如表 8.1 所示。

表 8.1　快速指数算法示例

i		9	8	7	6	5	4	3	2	1	0
b_i		1	0	0	0	1	1	0	0	0	0
c		0	1	2	4	8	17	35	70	140	
d	280	560									
		1	7	49	157	527	526	160	241	298	
	166	1									

所以 $7^{560} \bmod 561 = 1$。

2. 密钥的产生

产生密钥时，需要考虑两个大素数 p、q 的选取，以及 e 的选取和 d 的计算。因为 $n = p\,q$

在体制中是公开的，因此为了防止敌手通过穷搜索发现 p、q 这两个素数，应在一个足够大的整数集合中选取大数。因此如何有效地寻找大素数是第一个需要解决的问题。

寻找大素数时一般是先随机选取一个大的奇数（例如用伪随机数产生器），然后用素性检验算法检验这一奇数是否为素数，如果不是则选取另一大奇数，重复这一过程，直到找到素数为止。素性检验算法通常都是概率性的，但如果算法被多次重复执行，每次执行时输入不同的参数，算法的检验结果都认为被检验的数是素数，那么我们就可以比较有把握地认为被检验的数是素数。例如 Miller-Rabin 算法。可见寻找大素数是个比较繁琐的工作。然而在 RSA 体制中，只有在产生新密钥时才需要执行这一工作。

p 和 q 决定出后，下一个需要解决的问题是如何选取满足 $1<e<\varphi(n)$和 $\gcd(\varphi(n), e)=1$ 的 e，并计算满足 $d\cdot e=1 \bmod \varphi(n)$的 d。这一问题可由推广的 Euclid 算法完成。

8.4.3 对 RSA 的攻击

对密码分析者来说攻击RSA体制最显而易见的方法就是分解模数N ，因为一旦求出N的两个素因子p 和q，N 的欧拉数$\varphi(N)=(p-1)(q-1)$立即可得，再求出以$\varphi(N)$为模的公钥e的乘逆d，从而破译RSA的秘密密钥。

但是分解大合数是众所周知的数学难题。近300多年来许多著名的数学家对这个问题进行了大量研究。虽然近代科学家借助电子计算机和先进的科学技术在这方面取得了很大的进展，数学家已经成功分解了150位十进制的大合数，但是至今为止，要分解一个200位的大合数还是无能为力的。

1. 公共模数攻击

如果有一种方法能够不分解N而方便地计算出$\varphi(N)$，则RSA体制可轻而易举地被攻破。不过目前还没有什么人能够证明直接计算$\varphi(N)$比分解N更容易。从另一个方面也说明为什么RSA密码体制的模数N必须选择合数而不能选择素数。因为如果模N选择素数那么公布了N 也就公布了$\varphi(N)$，从$\varphi(N)$和公开钥e可以很容易地推导出秘密密钥d。

既然秘密密钥d不能推导出来，那么它能不能直接计算出来呢？最显而易见的方法当然是穷举搜索法。但是只要密码设计者仔细选取d，并使它本身足够大就可以挫败密码分析者的穷举搜索攻击。

如果密码分析者知道了d，可以方便地按下述方法实现对N的分解。首先计算出$(ed-1)$，由RSA体制的协议可知，$(ed-1)$必然是$\varphi(N)$的一个倍数。密码学家Miller指出，利用$\varphi(N)$的任何倍数都能成功实现对N的分解。但是到目前为止，还没有人能够提出这类对N进行因子分解的实际方法。因此可以断言，密码分析者不可能用比分解N更容易的方法直接计算秘密密钥，否则大合数分解的难题就不存在了。

但是这并不意味着密码分析者不能通过其它途径来破译RSA体制。美国学者Simmons 指出RSA公钥密码存在着一种潜在的弱点，这就是幂剩余变换的周期性。利用这种特殊的周期性，可以不必破译秘密密钥而直接得到明文。其方法如下：

设m是待加密的明文，(N, e)是RSA密码体制的公开密钥，则密文为$c=m^e \bmod N$。密码分析者从不保密信道上得到密文c后，并不设法获取秘密密钥d，而是利用公开密钥对密文c依次进行如下变换：

$$c^{e}=c_1 \bmod N$$

$$c_1^{e}=c_2 \bmod N$$

$$\cdots$$

$$c_{k-1}^{e}=c_k \bmod N$$

$$c_k^{e}=c_{k+1} \bmod N$$

上述一组变换经过$k+1$步迭代之后，幂剩余变换的结果恰好等于密文c。对照式$c=m^e \bmod N$可知第k步的变换结果c_k就是明文m。这样利用众所周知的公开密钥(N, e)和截获的密文成功地破译了RSA公开密钥体制。

设RSA公开密钥体制的一组参数为$p=383$，$q=563$，$N=pq=215629$，$e=49$，$d=56957$，加密和解密变换式分别为：

$$m^{49}=c \bmod 215629$$
$$c^{56957}=m \bmod 215629$$

如果明文信息$m=123456$，则相应的密文为

$$123456^{49}=1603 \bmod 215629$$

密码分析者利用公钥(215629，49)和密文1603按所示的一组变换进行运算得到$c_1=180661$，$c_2=109265$，$c_3=131172$，$c_4=98178$，$c_5=56372$，…，$c_8=85978$，$c_9=123456$，$c_{10}=1603 \bmod 215629$。由此可见仅仅经过10步变换就得到$c_{10}=c$，于是立即推知$c_9=m=123456$。

这是由于幂剩余函数满足下述周期性定理，即若N为素数或不同素数之积，且$\gcd(e,\varphi(N))=1$，则

$$a^{ek}=a \bmod N$$

式中k为某一正整数。当上述例子中模数的欧拉数为$\varphi(N)=214684$时，49的指数是10。也就是说周期k仅为10，这固然是加密指数e的特定选择所决定的。这种结果也很容易理解，因为$\gcd(e,\varphi(N))=1$，所以每次变换均为一一映射。由于信息空间是有限的，连续映射最终一定会恢复到初始状态。

要避免密码分析者利用幂剩余的周期性破译RSA体制，就要使k足够大以至于密码分析者在合理的时间内不能够破译RSA密文。事实上幂剩余函数的周期k与模数N的两个素因子的欧拉数$\varphi(p)$和$\varphi(q)$的最小公倍数l有关，l越大幂剩余函数的周期就越长，反之l越小，周期就越小。显然，为了使RSA公钥密码实际上是不可能破译的，理当选择l较大的素因子p和q。

还有一个值得注意的问题是在以RSA公钥密码作为加密标准的通信网中，每个用户的密钥不能有相同的模值。这里最显而易见的问题是同一明文用相同的模数但用不同的指数加密，而且这两个指数是互素的。那么无需任何一个解密密钥就可恢复出明文，一般情况下这很有可能。

假设m是明文信息，有两个不同用户的加密密钥分别是e_1和e_2，他们有共同的模数N，两个密文信息分别为：

$$c_1 = m^{e_1} \bmod N,$$
$$c_2 = m^{e_2} \bmod N。$$

密码分析者知道N，e_1，e_2，c_1和c_2，他们用下述方法恢复出明文m。

由于e_1和e_2和互素，由欧几里德算法能够找出r和s使之满足$r\,e_1 + s\,e_2 = 1$，在r和s中，有一个是负数。假定r是负数，那么用欧几里德算法计算出c_1^{-1}，然后有$(c_1^{-1})^{-r} c_2^{s} = m^{e_1 r + e_2 s} \equiv m \bmod N$。这样，无需破译秘密密钥$d$，就可以得到明文$m$。这种方法叫做RSA的共模攻击。避免这种攻击的措施是不在一组用户之间共享N。

2. 选择密文攻击

RSA的模幂运算有保持输入的乘法结构的特性，即有$E_K(a\,b) \equiv E_K(a)\,E_K(b) \pmod n$，这将影响RSA算法协议的安全性。

（1）破译密文

用户甲发布了他的公钥(e, n)，攻击者监听到一段发给甲的密文c，$c = m^e \bmod n$为了解析出明文m，攻击者随机选取了一个小于n的数r，计算$y = r^e \bmod n$，$t = y\,c \pmod n$，并且令$k = r^{-1} \pmod n$，则$k = y^{-d} \bmod n$ 。现在攻击者拿t 给用户甲签名（甲得用私钥签名，而非计算hash 值），当甲把签名返回给攻击者时,攻击者就得到了$s = t^d \bmod n$ 。这样攻击者就可以计算：

$$K\,s \pmod n = y^{-d} \times t^d \pmod n = y^{-d} \times y^d \times c^d \pmod n = c^d \bmod n = m$$

于是攻击者获得了明文m。

（2）骗取签名

同样对于用户甲，攻击者有非法消息m_i要获得甲的签名，但m_i直接给甲不能被接受，于是攻击者选择一个r，算得$y=r^e \pmod n$，再计算$m=y\,m_i \pmod n$，将m给甲签字，获得$s=m^d \pmod n$，这样就可由计算

$$sr^{-1}=y^d m_i{}^d r^{-1} \pmod n = r\times r^{-1}\times m_i{}^d \pmod n = m_i{}^d \pmod n$$

于是攻击者得到了m_i的签名。所以，安全起见，不要给陌生人提交的随机性文件签名，并且最好先采用单向的散列函数。ISO9796的分组格式可用来防止这样的攻击。

3. 低加密指数攻击

在RSA系统中，使用小的加密公钥可以加快加密和验证签名的速度，但过小的公钥易受到攻击。如果用户组中3个用户都使用3作为公钥，当用这种公钥对同一个明文m加密，则$c_1=m^3 \pmod{n_1}$，$c_2=m^3 \pmod{n_2}$，$c_3=m^3 \pmod{n_3}$，n_1、n_2、n_3各不相同，且$m<n_1$，$m<n_2$，$m<n_3$。一般安全性考虑，n_1、n_2、n_3是互素的，这样由中国余数定理可从c_1，c_2，c_3计算出c，且$c=m^3 \bmod (n_1 n_2 n_3)$，显然$m^3<n_1 n_2 n_3$，所以$m=c^{1/3}$。

Hastad证明如果采用不同的模n，相同的公钥e，则对$e(e+1)/2$个线性相关的消息加密，系统就会受到威胁。所以一般选取16位以上素数，速度也快，并且可防止这样的攻击。而对短的消息，使用独立随机数填充以保证$m^e \pmod n \neq m^e$，从而杜绝低加密指数攻击。

4. 定时攻击

定时攻击主要针对于RSA核心运算是非常耗时间的模乘，只要能够精确监视RSA的解密过程，获得解密的时间，从而估算出私有密钥d。模幂是通过一位一位来计算的，每次迭代执

行一次模乘，并且如果当前位是1，则还需要进行一次模乘。对于有些密码，后一次模乘执行速度会极慢，攻击者就可以在观测数据解密时，根据执行时间判断当前位是1或者0。不过这种方法只是理论上可以考虑，实际操作很困难。如果在加密前对数据做盲化处理，再进行加密，使得加密时间具有随机性，最后进行去盲，这样可以抵抗定时攻击，不过增加了数据处理步骤。

综上所述，在使用RSA公钥密码时必须注意以下问题：

（1）选择素数p和q时，应使这两个素数的欧拉数$\varphi(p)$和$\varphi(q)$的最小公倍数l尽可能大。l越大，幂剩余函数的周期就越长。这样就可以避免密码分析者利用剩余函数的周期性破译该体制。

（2）密钥中的各项参数应选得足够大，以避免密码分析者利用穷举搜索来破译该体制。

（3）在同一个通信网络中，不同的用户不应该使用共同的模数。

8.4.4 RSA-OAEP 加密标准

最佳非对称加密填充（OAEP）是对消息编码的一种方法，由 Mihir Bellare 和 Phil Rogaway 提出。首先对消息使用 OAEP 编码，然后再使用 RSA 加密，该方法是可证明安全的。

1. 加密操作

① 长度检查

a. 如果附加在消息上的标签 L 的长度大于 Hash 函数（SHA-1 为 $2^{61}-1$ 字节）的输入限制，那么输出“标签太长”并停止。

b. 如果消息 M 的长度 $mLen$ 大于 $k-2hLen-2$，其中 k 是密文长度，$hLen$ 表示 Hash 函数输出的长度，那么输出“消息太长”并停止。

② EME-OAEP 编码（见下图 8.2）

a. 如果没有提供标签 L，则让 L 为空字符串，且 $lHash = hash(L)$，$hLen$ 表示其长度。

b. 生成包括 $k-mLen-2hLen-2$ 个 0 的字节串 PS，PS 的长度可能为 0。

c. 连接 $lHash$、PS、十六进制值 0x01 以及消息 M，得到长度为 $k-hLen-1$ 的数据块 DB：$DB = lHash \,||\, PS \,||\, 0x01 \,||\, M$

d. 生成随机字节串 $seed$，其长度为 $hLen$。

e. 使用掩码生成函数 MGF 得到 $dbMask = \mathrm{MGF}(seed, k-hLen-1)$。

f. $maskedDB = DB \oplus dbMask$。

g. $seedMask = \mathrm{MGF}(maskedDB, hLen)$。

h. $maskedSeed = seed \oplus seedMask$。

i. 连接十六进制值为 0x00 的字节串、$maskedSeed$ 和 $maskedDB$，得到长度为 k 的编码消息 EM：$EM = 0x00 \,||\, maskedSeed \,||\, maskedDB$。

③ RSA 加密

a. 把编码消息 EM 转换成整数消息 m：$m = \mathrm{OS2IP}(EM)$。

b. 对 RSA 公钥(n, e)和消息 m 使用 RSAEP 加密得到整数密文 c：c=RSAEP $((n, e), m)$。
c. 把 c 转换成长度为 k 字节的密文 C：C=I2OSP (c, k)。
④ 输出密文 C

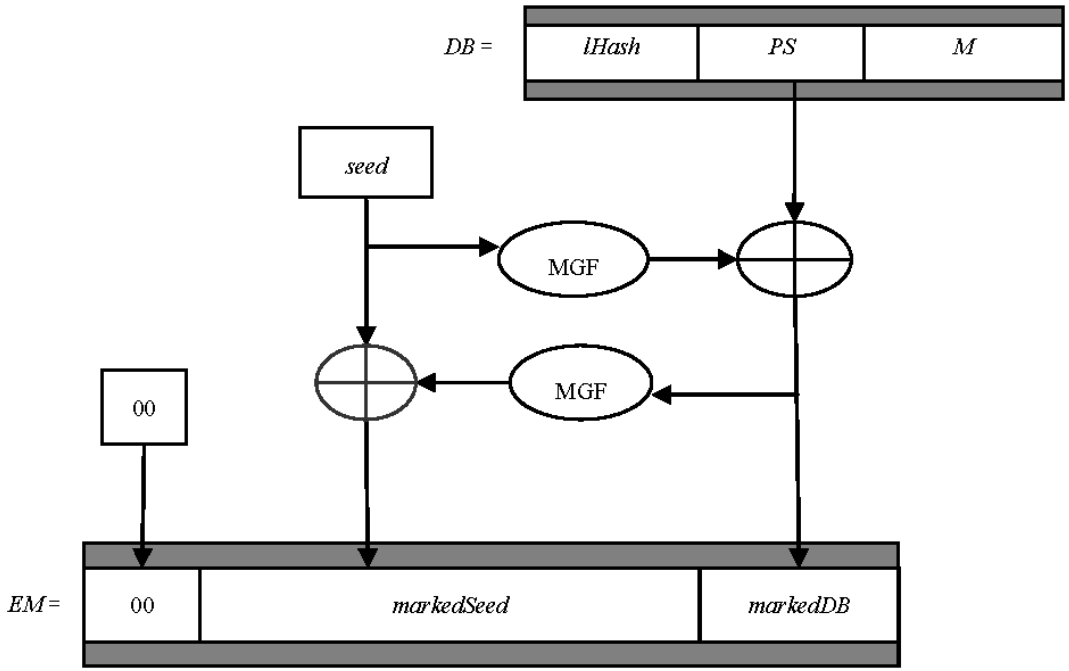

图 8.2　EME-OAEP 编码操作

2. 解密操作

hLen 表示 Hash 函数输出的字节长度，MGF 表示掩码生成函数，K 表示 RSA 私钥，C 待解密的密文，其长度为 k，且 $k \geqslant 2hLen+2$。消息 M 的长度为 *mLen*，且 $mLen \leqslant k-2hLen-2$。

① 长度检查

a. 如果附加在消息上的标签 L 的长度大于 Hash 函数（SHA-1 为 $2^{61}-1$ 字节）的输入限制，那么输出“解密错误”并停止。

b. 如果密文 C 的长度不是 k，那么输出“解密错误”并停止。

c. 如果 $k<2hLen+2$，其中 *hLen* 表示 Hash 函数输出的字节长度，则停止。

② RSA 解密

a. 把密文 C 转换成整数密文 c：$c=\mathrm{OS2IP}(C)$。

b. 对 RSA 私钥 K 和密文 c 使用 RSADP 解密得到整数消息 m：$m=\mathrm{RSADP}(K, c)$。

如果 RSADP 输出“密文超出长度”则输出“解密错误”并停止。

c. 把消息 m 转换成长度为 k 字节的编码消息 EM：$EM=\mathrm{I2OSP}(m, k)$。

③ EME-OAEP 解码

a. 如果没有提供标签 L，则让 L 为空字符串，且 $lHash=hash(L)$，*hLen* 表示其长度。

b. 把编码消息 EM 分离为字节 Y，长度为 *hLen* 的 *maskedSeed* 以及长度为 $k-hLen-1$ 的 *maskedDB*：$EM=Y \| maskedSeed \| maskedDB$。

c. $seedMask=\mathrm{MGF}(maskedDB, hLen)$。

d. $seed = maskedSeed \oplus seedMask$。

e. $dbMask = \mathrm{MGF}(seed, k - hLen - 1)$。

f. $DB = maskedDB \oplus dbMask$。

g. 分离 DB 得到长度为 $hLen$ 的 $lHash$（可能为空的）填充字节 PS、十六进制值 0x00 以及消息 M：$DB = lHash' \| PS \| 0x01 \| M$。

如果分离 M 和 PS 得到的十六进制值不是 0x01，或者 $lHash$ 不等于 $lHash'$，或者 Y 非零，输出“解密错误”并停止。

④ 输出消息 M

8.3 ElGamal 密码体制

ElGamal 密码体制是 T.ElGamal 于 1984 年提出的，它至今仍然是一个安全性良好的公钥密码体制。

8.3.1 ElGamal 算法

ElGamal 算法的安全性是建立在有限域上的离散对数的难解性这一数学难题的基础上的。

（1）密钥产生过程

选择一素数 p 以及两个小于 p 的随机数 g 和 x，计算 $y \equiv g^x \bmod p$。以(y，g，p)作为公开密钥，x 作为秘密密钥。

（2）加密过程

设加密明文消息M，随机选取一个与$p-1$互素的整数k，计算$C_1 \equiv g^k \bmod p$，$C_2 \equiv y^k M \bmod p$，密文为$C=(C_1, C_2)$。

（3）解密过程

计算$C_2 C_1^{-x} \bmod p$就可以得到正确的解密结果。

解密过程的正确性证明如下：

$$\begin{aligned} C_2 C_1^{-x} \bmod p &= (y^k M)(g^k)^{-x} \bmod p \\ &= (g^{xk} M)(g^k)^{-x} \bmod p \\ &= M \bmod p \end{aligned}$$

下面的这个例子可以简单说明在ElGamal密码体制中所进行的计算。

例：设$p=11$，$g=7$，在$GF(11)$上有$7^0=1$，$7^1=7$，$7^2=5$，$7^3=2$，$7^4=3$，$7^5=10$，$7^6=4$，$7^7=6$，$7^8=9$，$7^9=1$，$7^{10}=1$，g是$GF(11)$上的本原元素。

设A的私钥$x_A=3$，公钥$y_A=2$；B的私钥$x_B=5$，公钥$y_B=10$。

假定A欲将消息$m=6$保密寄给B。A取$x=7$，A计算：

$$C_1 \equiv g^7 \bmod 11 = 6$$

$$K \equiv (10)^7 \bmod 11 = 10$$

$$C_2 \equiv K m \bmod 11 = 60 \equiv 5 \bmod 11$$

A将$(6, 5)$作为密文寄给B。B收到后计算

$$K \equiv (g)^5 \bmod 11 = 10$$

$$K^{-1} \equiv g^{10-5} \equiv g^5 = 10$$

$$M \equiv K^{-1} C_2 \equiv 50 \bmod 11 \equiv 6$$

故恢复 $m=6$。

8.3.2 ElGamal 公钥密码体制的安全性

为了确保 ElGamal 算法的安全性，通常 p 至少应该具有 150 位以上的十进制数字，大约 512 位的二进制数。而且，$p-1$ 至少有一个大的素因子。在满足上述条件下，根据 p、g、b 来计算离散对数是相当困难的。

设 p 是一个素数，$\alpha \in Z_p^*$，α是一个本原元，$\beta \in Z_p^*$。已知α和β，求满足$\alpha^n \equiv \beta \pmod{n}$的唯一整数 n，$0 \leqslant n \leqslant p-2$，称为有限域上的离散对数问题，常将 n 记为 $\log_\alpha \beta$。

ElGamal 公钥密码体制可以在计算离散对数困难的任何群中实现，不过通常使用的是有限域，但不局限于有限域。

关于有限域上的离散对数问题，已进行了许多深入的研究，取得了许多重要成果。但到目前为止，还没有找到一个非常有效的多项式时间算法来计算有限域上的离散对数。一般而言，只要素数 p 选取适当，有限域 Z_p 上的离散对数问题是难解的。反过来，已知α和 n，计算 $\beta = \alpha^n \bmod p$ 是容易的。因此，对于适当的素数 p，模 p 指数运算是一个单向函数。

在 ElGamal 公钥密码体制中，$\beta = \alpha^d \bmod p$，从公开的α和β，求保密的解密密钥 d，就是计算一个离散对数。因此，ElGamal 公钥密码体制的安全性主要是基于有限域 Z_p 上离散对数问题的难解性。

为了抵抗目前已知的一些对 ElGamal 公钥密码体制的攻击，素数 p 按十进制表示至少应该有 150 位数字，并且 $p-1$ 至少应该有一个大的素因子。

Thank you!

(To be continued....)