REESSE 2公开密钥密码体制

苏盛辉[1] 杨炳儒[2]

(北京石油化工学院信息管理系 北京102617)[1] (北京科技大学信息工程学院 北京100083)[2]

摘 要 作者综合利用超递增序列、杠杆函数和HASH函数的特性提出了能有效地抵御"极小点"攻击和"LOB-L3归约基"攻击的REESSE2公开密钥密码体制,详细描述了该体制的数学基础、密钥生成算法、加密算法和解密算法。文章对REESSE2体制的安全性和优越性做了分析,并归纳了提高公钥密码体制安全性的两条途径。

关键词 杠杆函数,超递增数列,冗余加密,公开密钥密码体制

The REESSE 2 Public Key Cryptosystem

SU Sheng-Hui[1] YANG Bing-Ru[2]

(Beijing Institute of Petrochemical Technology, Beijing 102617)[1] (Beijing University of Science and Technology, Beijing 100083)[2]

Abstract On the basis of the super increasing sequence, lever function and HASH function, the authors put forward the REESSE2 public key cryptosystem which can defend efficiently attacks by the extreme small dot and LOB-L[3] reduced base, expound the math foundation and three algorithms for generating-key, encryption and decryption of the crytosystem, analyze algorithmic security and advantage, and induce the two approaches to enhancing securities of public key crytosystems.

Keywords Lever function, Super increasing sequence, Redundancy encryption, Public key crytosystem

1 引言

1978年,在Whitfield Diffie和Martin Hellman提出公开密钥密码思想两年后,首先由Ralph Merkle和Martin Hellman设计了一个基于背包问题的公钥密码体制。尽管背包问题被证明属于NP完全类,但是,由于MH变换(模乘变换)本身的缺陷,该体制先后被以"极小点"方法和"LOB-L[3]归约基"方法破译[1~3]。

MH背包体制可以被扩展到伽罗瓦域的多项式环上。此时,虽然超递增序列的元素由整数变成了多项式,但是从私钥到公钥的变换仍是模乘变换。由于多项式模乘变换与整数模乘变换在本质上是一致的,因此这种改变只是原有体制的扩展,而不是新体制的发明[1,4])。显然,多项式环上的MH公钥体制与椭圆曲线上的ElGamal公钥体制有异曲同工之妙:它们都提高了体制的安全性。这是因为,在多项式环上或椭圆曲线上,对基本运算符进行了组合,形成了一种新的更复杂的群的运算符。

本文介绍的REESSE 2公钥密码体制综合利用了超递增序列、杠杆函数和HASH单向散列函数的特点,对从私钥到公钥的变换进行了创新,即密钥的变换从单群扩展到了双群,同时在域的两个群上实施变换运算。注意,这与代数系统升级了但密钥变换仍在单群上进行有本质的不同。经分析,REESSE 2体制在安全性和速度方面有其优势。

2 REESSE2体制的数学基础

2.1 三个有关函数

2.1.1 杠杆函数 在REESSE1公钥体制中[5],私有密钥为互素序列$\{A_1,A_2,\cdots,A_n\}$,公开密钥$\{C_1,C_2,\cdots,C_n\}$经公式$C_i=(A_i*W^{f(i)})\bmod M$变换后得到。其中,$f(i)$为整数到整数的单射函数,且$5\leqslant f(i)\leqslant(n+4),(i=1,2,\cdots,n)$。

在上述变换中,$f(i)$具有这样的特点:从公开密钥破译私有密钥时,需考虑$f(i)$的全排列数$n!$,因此当n足够大时,$f(i)$的排列在有效时间内不可被穷举;但从私有密钥解开密文时只需考虑$f(i)$的累加和,时间复杂度与n多项式相关,故解密是可行的。若把密文当作支点,则$f(i)$是"公开"一头计算量大,"私有"一头计算量小,因此,称具有这种特征的$f(i)$为杠杆函数。

2.1.2 超递增序列 设$A_1,A_2,\cdots,A_n$为n个互不相同的正整数,且满足:$A_i>\Sigma A_j(i=2,3,\cdots,n;j=1,2,\cdots,i-1)$,则称这样的正整数序列为超递增序列,记为$\{A_1,A_2,\cdots,A_n\}$或$\{A_i\}$。

超递增序列有如下性质:对于任意的正整数$m(1\leqslant m\leqslant$

苏盛辉 副教授,博士生,主要从事信息安全、柔性建模与集成技术的研究和开发。**杨炳儒** 教授,博士生导师,主要从事知识发现与智能系统、柔性建模与集成技术的研究。

7 Pan Y, Li K, Zheng S Q. Fast Nearest Neighbor Algorithms on a Linear Array with a Reconfigurable Pipelined Bus System: [Technical Report # 97-002]. Department of computer Science, Louisiana State Unibersity, Baton Rouge, LA 70803(1997)

8 Preparata F P, Hong S J. Convex hulls of finite sets of points in two and three dimensions. Comm. ACM, 1977, 2(20): 87~93

9 Atallah M J, Goodrich M T. Parallel algorithms for some functions of two convex polygons. algorithmica, 1988, 3: 535~548

10 Datta A, Owens R, Soundaralakshmi S. Fast Sorting Algorithms on a Linear Array with a Reconfigurable Pipelined Bus System. IEEE Transactions on Parallel and Distributed Systems 2002, 13(3): 202~222

n)，从超递增序列$\{A_i\}$中任选m个元素作为子集，则子集和：$E_S=(A_{i_1}+A_{i_2}+\cdots+A_{i_m})$唯一确定了原子集的元素。下标$i_1$，$i_2$，…，$i_m$代表子集元素在原超递增序列中的序号。

2.1.3 HASH函数 也称单向散列函数，它把一个二进制明文文件或数据块转换为n比特的输出，该输出称为明文摘要，n通常为64、128或256等。HASH函数具有这些特点：(1)给定明文，计算明文摘要很容易；(2)给定明文摘要，逆向计算明文很困难；(3)从两个不同的明文得到同样的明文摘要几乎是不可能的[2]。

2.2 从超递增序列得到非超递增序列

设$\{A_1,A_2,\cdots,A_n\}$为超递增序列，令：$C_i=((A_i+Z*f(i))*W) \bmod M(i=1,2,\cdots,n)$，等式右端称为REESSE2变换。在上式中$M$为正素数，并且$M>\Sigma A_i$，$W$、$Z$为小于$M$的工作参数，$f(i)$为单射的杠杆函数，且$5\leqslant f(i)\leqslant(n+4)$。从上式可以看到，密钥的变换是在模$M$同余类域上进行的，并同时在域的两个群上实施运算。显然，MH变换是REESSE2变换$Z=0$时的特例。

不难验证，$\{C_1,C_2,\cdots,C_n\}$已不具有超递增序列的性质，因此它是非超递增序列。

2.3 从非超递增序列子集和得到超递增序列子集和

2.3.1 非超递增序列子集和的计算 设$\{C_1,C_2,\cdots,C_n\}$为n项非超递增序列(可以理解为公钥)，$b_1b_2\cdots b_n$为n位二进制数，则非超递增序列子集和为：$E_N=\Sigma(C_i*b_i) \bmod M$ (i from 1 to n)，如果$b_1b_2\cdots b_n$是明文分组，则E_N是相应的密文。

2.3.2 超递增序列子集和的导出 从近世代数知，倘若W^{-1}为W的乘法逆元，则有$(W*W^{-1}) \bmod M=1$。

倘若$-Z$为Z的加法逆元，则有$(Z+(-Z)) \bmod M=0$。且对任意正整数$K<M$，有：

$(K*Z+K*(-Z)) \bmod M=(K*Z+(-(K*Z))) \bmod M=0$

那么，如何在超递增序列、工作参数和杠杆函数的基础上从E_N求出超递增序列子集和呢？即如何求出$E_S=\Sigma(A_i*b_i)$，(i from 1 to n)。如果能，则可以从E_S中恢复出原二进制明文$b_1b_2\cdots b_n$。

理论上，我们可以这样来推算，首先，令：$K=\Sigma(f(i)*b_i)$，则有：

$E_N=(\Sigma(C_i*b_i)) \bmod M=(\Sigma((A_i+Z*f(i))\cdot W*b_i)) \bmod M=(W*\Sigma((A_i+Z*f(i))*b_i)) \bmod M$

因此，$(E_N*W^{-1}) \bmod M=(\Sigma((A_i+Z*f(i))*b_i)) \bmod M$

$=(\Sigma(A_i*b_i+Z*f(i)*b_i)) \bmod M=(\Sigma(A_i*b_i)+\Sigma(Z*f(i)*b_i)) \bmod M$

$=(\Sigma(A_i*b_i)+Z*\Sigma(f(i)*b_i)) \bmod M=(\Sigma(A_i*b_i)+Z*K) \bmod M$

进而，$(E_N*W^{-1}+(-Z)*K) \bmod M=(E_N*W^{-1}+(-(Z*K))) \bmod M$

$=(\Sigma A_ib_i+Z*K+(-(Z*K))) \bmod M=(\Sigma(A_i*b_i)) \bmod M=\Sigma(A_i*b_i)=E_S$

所以，$E_S=\Sigma(A_i*b_i)=(E_N*W^{-1}+(-Z)*K) \bmod M$

然而，在实际情况中，我们并不事先知道二进制明文($b_1b_2\cdots b_n$)，因而就无法直接确定K，但由于K的取值范围有限，因此可以搜索K，并根据E_S能否被$\{A_i\}$的某些项相减为0来验证K，在K得到验证的同时，也就推导出了原二进制明文。搜索K的次数不会超过$\Sigma f(i)$，其与n是多项式时间复杂度相关，故求解是可行的。

2.3.3 解的唯一性问题 由于$\{C_i\}$是非超递增序列，其子集与子集和之间不存在一一对应关系，因此，采用边搜索边验证的方法来求取E_S时，符合条件的E_S值可能不止一个，从而会导致明文解$b_1b_2\cdots b_n$的不唯一性。

设密文E_N可由$\{C_i\}$的两个不同子集的累加和得到，因此：

$E_N\equiv(Cx_1+Cx_2+\cdots+Cx_m)\equiv(Cy_1+Cy_2+\cdots+Cy_k) \bmod M$

即：$((Ax_1+f(x_1)*Z)+(Ax_2+f(x_2)*Z)+\cdots+(Ax_m+f(x_m)*Z))*W$

$\equiv((Ay_1+f(y_1)*Z)+(Ay_2+f(y_2)*Z)+\cdots+(Ay_k+f(y_k)*Z))*W \bmod M$

$(Ax_1+Ax_2+\cdots+Ax_m+k1*Z)\equiv(Ay_1+Ay_2+\cdots+Ay_k+k2*Z) \bmod M$

其中，$k1=f(x_1)+f(x_2)+\cdots+f(x_m)$，$k2=f(y_1)+f(y_2)+\cdots+f(y_k)$

不妨令$k1\geqslant k2$，由于上述变换是在模M加法交换群上进行，故有：

$Z*(k1-k2)\equiv(Ay_1+Ay_2+\cdots+Ay_k)+(-(Ax_1+Ax_2+\cdots+Ax_m)) \bmod M$

简记为：$Z*(k1-k2)\equiv((\Sigma Ay_i)+(-\Sigma Ax_j)) \bmod M$

上式表明，当E_S的值不唯一时，必有Z的值与$((\Sigma Ay_i)+(-\Sigma Ax_j))$相关。其逆否含义是，如果$Z$的值与$((\Sigma Ay_i)+(-\Sigma Ax_j))$无关，则$E_S$的值将会唯一。

当某个元素A_i同时出现在(ΣAy_i)与$(-\Sigma Ax_j)$中时，则它们会相互抵消，对Z的取值没有影响。因此，考虑到那些不同时出现在(ΣAy_i)与$(-\Sigma Ax_j)$中的元素，当序列长度为n时，$((\Sigma Ay_i)+(-\Sigma Ax_j))$型值的个数最多为$2^n$。又由于$Z$在$M$范围内取值，如果让$M>2^{(n+32)}$，即$(\log_2M-n)>32$，那么，$Z$取得$((\Sigma Ay_i)+(-\Sigma Ax_j))$相关值的概率会低于$1/2^{32}$。所以，选取适当的$M$，就可以使$E_S$不唯一的概率降至非常非常低，且同时满足序列$\{A_i\}$的密度大于0.645。另外，通过事先排查一些$Z$值，可以使$E_S$不唯一的概率进一步降低。

3 REESSE 2体制的算法描述

目前，REESSE 2公钥体制包括密钥生成、加密和解密等三个算法。从效率方面考虑，公钥算法一般只用来加密对称算法的会话密钥，而用对称算法去加密明文。为防止"LOB-L3归约基"方法攻击，本体制应用了"冗余加密"的思想，即会话密钥来源于一个随机数据块的HASH输出，而加密与解密是针对随机数据块进行的。

3.1 密钥生成算法

(1)产生项数为n的超递增序列$\{A_1,A_2,\cdots,A_n\}$

(2)选取适当的正素数M，使$M>\Sigma A_i$和$(n/\log_2M)>0.645$以及$(\log_2M-n)>32$

(3)随机产生两两不同的杠杆函数值$f(1)$、$f(2)$、…、$f(n)$，$5\leqslant f(i)\leqslant(n+4)$

(4)选取适当的正素数W、Z，其满足$W<M$、$Z<M$

(5)计算非超递增序列$C_i=((A_i+Z*f(i))*W) \bmod M$，$(i=1,2,\cdots,n)$

(6)从$(Z+(-Z)) \bmod M=0$求出$-Z$，从$(W*W^{-1}) \bmod M=1$递归求出W^{-1}[6]。

然后，以($\{A_i\}$、W^{-1}、$-Z$、M)作为私有密钥，($\{C_i\}$、M)作为公开密钥，$f(i)$可舍弃。

3.3 加密算法

设($\{C_i\}$、M)为公开密钥，h为随机数据块分组数，$1<h\leqslant32$。

(1)随机产生$(h*n)$比特的数据块$b_{11}b_{12}\cdots b_{1n}b_{21}b_{22}\cdots b_{2n}$

$\cdots\cdots b_{h1}b_{h2}\cdots b_{hn}$，其 n 比特的 HASH($b_{11}b_{12}\cdots b_{1n}$，$b_{21}b_{22}\cdots b_{2n}$，$\cdots\cdots$，$b_{h1}b_{h2}\cdots b_{hn}$)输出作为会话密钥 K；

(2)令 $i=1$；

(3)令 $E_{Ni}=0$，$j=1$；

(4)如果 $b_{ij}=1$，则 $E_{Ni}=(E_{Ni}+C_j)\bmod M$

(5)做 $j=j+1$，如果 $j\leqslant n$，则转至(4)；

(6)做 $i=i+1$，如果 $i\leqslant h$，则转至(3)，否则，结束。

最后，得到密文(子集和)序列{E_{N1}、E_{N2}、…、E_{Nh}}。

3.4 解密算法

设({A_i}、W^{-1}、$-Z$、M)为私有密钥，{E_{N1}、E_{N2}、…、E_{Nh}}为密文(子集和)序列。

(1)令 $i=1$；

(2)做 $E_{Ni}=(E_{Ni}*W^{-1})\bmod M$；

(3)做 $E_{Ni}=(E_{Ni}+(-Z))\bmod M$；

(4)令 $b_{i1}b_{i2}\cdots b_{in}$各位皆为0，$E_S=E_{Ni}$，$j=n$；

(5)如果 $E_S\geqslant A_j$，则 $b_{ij}=1$并且 $E_S=E_S-A_j$；

(6)做 $j=j-1$，如果 $j\geqslant 1$并且 $E_S\neq 0$，则转至(5)；

(7)如果 $E_S\neq 0$，则转至(3)；

(8)做 $i=i+1$，如果 $i\leqslant h$，则转至(2)，否则，结束。

最后，以 n 比特的 HASH($b_{11}b_{12}\cdots b_{1n}$，$b_{21}b_{22}\cdots b_{2n}$，…，$b_{h1}b_{h2}\cdots b_{hn}$)输出作为会话密钥 K。

4 REESSE2体制的安全性分析

4.1 有效抵御极小点攻击方法

在 MH 变换中，$C_i=(A_i*W)\bmod M$。由于 $C_i=(A_i*W)$在不求余时仍是超递增序列，因此，这种变换并没有改变超递增序列的特性，Adi Shamir 正是抓住了这个“尾巴”以极小点方法从公开密钥中恢复出私有密钥，一举将 MH 背包体制破译。如果企图通过多次 MH 变换将超递增序列变得彻底一些，则随着模数的增大，背包密度会降低，因而，可利用 LOB-L^{3}归约基方法将密文破译[1,3]。这正是 Merkle 和 Hellman 的两难困惑所在。

在 REESSE2变换中，$C_i=((A_i+Z*f(i))*W)\bmod M$。当 $W\neq 1$时，由于$\{A'_i\}=\{A_i+Z*f(i)\}$不是超递增序列，$\{A'_i\}$不满足“序列的每一项大致是前面一项的两倍”的假设，因此，极小点方法将失效。当 $W=1$时，$C_i=(A_i+Z*f(i))$，由于 $f(i)$的存在，锯齿曲线极小点的聚点不可能存在，因此，极小点方法同样将失效。

4.2 有效抵御 LOB-L^{3}归约基攻击方法

对于背包序列{C_1，C_2，…，C_n}，其背包密度 $D=n/\log_2(\max C_i)$，其中 n 为序列长度(序列项数)。考虑到模数 $M>(\Sigma A_i)$和 $M>(\max C_i)$，所以，背包密度也可以表示成 $D=n/\log_2 M$，即序列长度与模数的比特长度之比。

Lagarias-Odlyzko 和 Brickell 都已证明了对于所有 D<0.645的背包公钥体制皆可以破译其密文[1]。也就是说，不管是 MH 变换还是 REESSE2变换，只要背包密度 D<0.645，就可以通过 LOB-L^{3}归约基方法从密文子集和 $E_N=(\Sigma C_i*b_i)\bmod M$ 中恢复出明文($b_1b_2\cdots b_n$)。

那么，这是否意味着只要取 $D>0.645$，就可以使 LOB-L^{3}归约基方法失效呢？令人遗憾的是 Lagarias-Odlyzko 和 Brickell 的否命题并不总是成立。

作者参照 LOB 方法编制了利用 L^3算法破解密文子集和问题的程序，经过对不同长度和模数构成的不同密度的背包所做之实验分析，发现：(1)即使当 $D>0.645$时，有些密文也能被破译，但并不是每个密文都能被破译；(2)随着背包密度的提高，密文被破译的可能性越小；(3)当背包密度相同时，随着背包长度的增加，密文被破译的可能性越小。关于第(3)点，作者认为主要是由于背包的攻击密度降低了。所谓攻击密度是指单位时间内所能攻击的最大次数与背包可能的密文数 M 之比。

从上述第(3)点知，当密度相同时(例如0.75)，随着序列长度的增加，每次攻击的成功率 p 会越来越少，如果以 h 次连续攻击作为一轮，则每轮攻击的成功率为$1/p^h$。据实验当 $n\geqslant 96$时，h 的值在32范围之内，并可以使$1/p^h<1/2^{64}$，这个概率已经完全达到实用的程度。

在 REESSE2体制中，加密是针对随机数据块($b_{11}b_{12}\cdots b_{1n}$ $b_{21}b_{22}\cdots b_{2n}\cdots\cdots b_{h1}b_{h2}\cdots b_{hn}$)进行的，并生成密文序列{$E_{N1}$、$E_{N2}$、…、$E_{Nh}$}。当 $n\geqslant 96$时，可以选取适当的模数 M，使得 LOB-L^{3}归约基方法不可能连续攻破{E_{N1}、E_{N2}、…、E_{Nh}}，即不可能完整破译出($b_{11}b_{12}\cdots b_{1n}b_{21}b_{22}\cdots b_{2n}\cdots\cdots b_{h1}b_{h2}\cdots b_{hn}$)，从而，依据 HASH 函数的特点，不可能破译出 n 比特的会话密钥 HASH($b_{11}b_{12}\cdots b_{1n}$，$b_{21}b_{22}\cdots b_{2n}$，……，$b_{h1}b_{h2}\cdots b_{hn}$)。注意，该 M 还应该使明文分组 $b_{i1}b_{i2}\cdots b_{in}$的不唯一性概率降至几乎为零。

4.3 杠杆函数 $f(i)$的不确定性

首先，如果攻击者企图猜测 $f(i)$的排列，则当 $n>80$时，它的全排列数将达 P_{80}^{80}，远远大于2^{80}，根据目前的万亿次计算机速度，$f(i)$的排列数目不可能有效穷举。其次，令 $Z_i=(Z*f(i))\bmod M$，$C_i=((A_i+Z_i)*W)\bmod M$，根据群的知识，当 Z_i 确定时，Z 与 $f(i)$两者不能唯一确定。当 Z 确定时，由于 W 的存在，则 C_i 与 $f(i)$之间没有一一对应关系。这些充分说明，在某个具体的公钥序列中，$f(i)$的排列是不确定的。另外，还可以通过一个置放函数来打破{A_i}的排列顺序。

4.4 密钥映射的不充分性和不可逆性

在 REESSE 2体制中，私有密钥虽然包括{A_i}、$f(i)$、Z、W 四部分($f(i)$在解密时未用到)，但却只有 $C_i=((A_i+Z*f(i))*W)\bmod M$ 的显性映射，映射是不充分的。该表达式实际包含了 n 个等式和$(2n+2)$个未知变量，所以，通过数学方法求解{A_i}、W 和 Z 是困难的。同时这种映射的不充分性，使得公钥与私钥之间不存在相对确定的函数可逆性。

5 REESSE2体制的特点与优越性

5.1 同时利用了计算机的数字运算与逻辑运算功能

传统的公钥密码体制一般只利用了计算机的数字运算功能，而 REESSE2体制不仅利用了计算机的数字运算功能，也利用了计算机的逻辑运算功能，是数学知识与软件知识相结合的产物。

5.2 公钥与私钥之间不存在相对确定的函数可逆性

正如4.4节所言，由于公钥与私钥之间映射的不充分性，公钥与私钥之间不存在相对确定的函数可逆性，且大量私钥被隐藏。而传统的公钥密码体制中，公钥与私钥之间一般存在相对确定的函数可逆性，且只有少数几个或一个工作变量作为私钥被隐藏。

5.3 揭示了提高公钥体制安全性的另一条途径

正如引言所述，多项式环上的 MH 背包体制及椭圆曲线上的 ElGamal 体制的安全性与其在简单代数系统上相比都得到了提高。这是一条对运算符进行组合的途径，是一条纯数学的途径。但是，由于简单代数系统与复杂代数系统之间(包括群之间、环之间、域之间)存在同构或同态现象，因此，简单

代数系统中的安全隐患在复杂代数系统中应该同样存在，并且在复杂代数中，公钥体制的安全性仍较大地依赖于计算机的运算速度。

另一方面，我们从REESSE1和REESSE2体制中看到，对运算数进行组合是提高公钥体制安全性的另一条途径，这是一条数学与软件相结合的途径。这种途径要求公钥体制必须是序列密码体制。在REESSE1和REESSE2体制中，运算数的组合与排列是由杠杆函数 $f(i)$ 实现的，使得体制的安全性对计算机运算速度的依赖程度非常低。通过运算符组合来提高安全性有如对单个CPU提高速度，这种提高是有极限的。通过运算数组合来提高安全性有如对多个CPU并联提速，这种提高是无极限的。

5.4 易于硬件实现且运算速度快

REESSE2体制的加密算法与解密算法中大部分是模加运算和逻辑判断，所以，易于硬件，特别是简单硬件与CPU实现。

与流行的RSA体制、ECC(椭圆曲线上的ElGamal)体制相比，REESSE2也有速度上的优势(即使考虑到冗余加密和 $O(n^3)$ 时间复杂度解密等因素)。这是因为：(1)REESSE2的模数可以不超过192比特，而RSA至少为1024比特，ECC也常为192比特；(2)REESSE2的运算符主要是基本的带模加法、而RSA是带模乘法，ECC是复合运算符，由于乘法、复合运算符是由基本运算符(加法、减法等)构成的，因此，完成一个复合运算所需的时间要远远多于一个基本运算所需的时间。

结束语 在本文，我们虽然提出了REESSE2公钥体制的理论算法，但是在技术实现上，还应该对REESSE2的解密算法进行进一步优化，争取使其时间复杂度降至 $O(n^2)$ 的水平。还应在模数和序列长度确定的情况下，测定最少冗余加密次数。最后，我们期待同仁对REESSE2公钥体制做出更深入的安全性分析。

参考文献

1 卢开澄. 计算机密码学(第2版). 北京：清华大学出版社，1998

2 Schneier B(美). Applied Cryptography(应用密码学，吴世忠，祝世雄等译). 北京：机械工业出版社，2000

3 冯登国. 密码分析学. 北京：清华大学出版社，2000

4 冯克勤，李尚志. 近世代数引论. 合肥：中国科学技术大学出版社，2002

5 苏盛辉. REESSE1公开密钥密码体制. 计算机工程与科学，2003(5)

6 苏盛辉，李国华，王其文. 模运算的新性质及求模逆元的递归算法. 信息安全与通信保密，2001(11)

(上接第136页)

总之，行为抽象可用于以下几种场合：

·需要从类中分离出行为，以便可以交换、保存、修改、共享或者调用相互独立的行为对象；

·需要在运行时选择行为；

·一些类的区别仅仅在于一个或者几个方法的不同。这样，增加新的类，实现这些不同的行为，可以以统一的方式使用具有不同行为的类。

·在选择不同的行为时，需要使用大量的条件判断代码。

下面使用2中的方法，以bridge模式为例，说明设计模式与行为抽象的关系：

Objectifier (Implementor, ConcreteImplementor$_j^{j\in 1..m}$, Abstraction) (1)

AbstractInterface(Abstraction,Operation) (2)

r:Abstraction (3)

C:ConcreteImplementor$_j^{j\in 1..m}$ (4)

r.operation$\ll_m c$.operationImp (5)

Bridge (Abstraction, Implementor, concreteImplementor$_j^{j\in 1..m}$,operation,operationImp) (6)

可见，Bridge模式中使用了行为抽象方法，在此基础上在客户(这里是Abstraction)端增加了一个抽象接口。

总结 面向对象中封装性不仅仅体现在对象的状态和行为的封装，行为抽象中也体现了一种封装性：抽象类封装了各个具体子类。通过委托，客户不需了解子类的细节，就可以具有不同的行为。

行为抽象方法所产生的结构被认为是在众多设计模式中使用的基本设计模式。我们认为，在设计模式中存在着一些基本的设计方法，它们只使用类、对象、方法、方法调用和数据成员的概念，但能反映基本的OO设计原则。我们可以从这些简单的基本设计方法入手学习面向对象设计方法。另外，研究这些基本设计模式，可以更好地学习和掌握更加复杂的设计模式，发现设计模式之间的关系以及发掘新的设计模式。

参考文献

1 Gamma E,Helm R,Johnson R,Vlissides J. Design Patterns. Addison Wesley,1995

2 Shalloway A,Trott J R. Design Patterns Explained:A New perspective on Object-Oriented Design. Addison-Wesley,2002

3 熊兵舫，张志祥. 设计模式的软件度量分析. 青岛大学学报，2003，增刊

4 Abadi M,Cardelli L. A Theory of Objects. Springer-Verlag New York,Inc.,1996

5 Smith J M,Stotts D. Elemental design patterns:A link between architecture and object semantics:[Technical Report TR-02-011]. Univ. of North Carolina,2002

6 Zimmer W. Relationships between design patterns. In pattern Languages of Program Design. Addison-Wesley,1994