Montgomery 逆算法的改进和应用*)

邓 锐 周玉洁

(解放军信息工程大学信息研究系 郑州 450002)

摘 要 本文通过对 Montgomery 逆算法核心部分的改进,得到两种分别以 4 为基和 8 为基的优化算法。其中以 4 为基的算法,在基本不增加算法实现复杂度的情况下,使迭代次数的平均上限从 $2n$ 降到 $\frac{7}{6}n$,平均迭代次数也从 $\frac{3n}{2}$ 降到了 $\frac{7}{8}n$。而 8 基算法则相应分别下降到 $\frac{25}{24}n$ 和 $\frac{25}{32}n$,但算法内部的比较和跳转稍有增多。由于新算法只要求两个关键操作数中有一个变成 1,就可以结束操作(原算法要求两个都变为 1),因此实际迭代次数可能还要少。本文提出的算法也可以运用在文[1,2,7]中求基本模逆的算法中。本文算法主要适用于软件实现,在 RSA 和 ECC 等公钥体制实现中有广泛的应用。

关键词 Montgomery 逆算法,模逆,公钥加密算法

Improvement to Montgomery Modular Inverse Algorithm

DENG Rui ZHOU Yu-Jie

(Information Research Department of PLA Information Project University, Zhengzhou 450002)

Abstract After a comprehensive investigation of the Montgomery modular inverse algorithm and its refined versions, we present two modified high radix algorithms. The 4-radix algorithm can reduce the average upper limit of the number of iterations from $2n$ to $\frac{7}{6}n$, and the average number of iterations from $\frac{3n}{2}$ to $\frac{7}{8}n$ while adding little complexity. And the 8-radix one can reduce the related number of iterations to $\frac{25}{24}n$ and $\frac{25}{32}n$, but there are more comparisons and branches. For the new algorithms terminate when one of the two key operands, rather than both, is reduced to 1, the actual number of iterations may be even smaller than the ones above. The new algorithms can also be used in algorithms in [1,2,7] to get classical modular inverse. The proposed algorithms are suitable for software implementations on general-purpose microprocessors, and can be used in public key cryptographic applications such as RSA and ECC.

Keywords Montgomery modular inverse, Modular inverse, Public key cryptographic algorithm

1 引言

Montgomery 算法在公钥密码体制(如 RSA,DSA,椭圆曲线密码体制等)的实现中,都有广泛的应用。但是这种算法对于求逆运算却不适用,因此 Kaliski[1] 在 Montgomery 算法思想的基础之上,提出了 Montgomery 逆的概念,并给出了一个有效的算法。在他的工作基础之上,文[2]对这个算法的后处理部分进行改进,并把它和 Montgomery 乘算法结合起来,使得 Montgomery 数可以进行快速的除法,从而使得许多高层算法的应用成为可能。文[7]又在文[2]的基础上进行了改进,节省了一次 Montgomery 模乘运算。本文将对他们的工作再进行改进,主要是对算法当中处于核心地位的前半部分算法进行改进,使得算法效率有较大提高。本文提出的算法也可以直接运用于文[1,2,7]中所提出的求基本模逆的算法中。

2 Montgomery 逆及其算法

对于 $a,b\in[0,p-1]$,定义 $c=ab2^{-n}\pmod p$ 为 a,b 的 Montgomery 乘积,其中 $n=\lceil \log_2 p \rceil$。考虑到密码学中的实际应用以及方便讨论,这里规定 p 是素数。相似地,对于 $a\in[0,p-1]$,定义 $x=a^{-1}2^n\pmod p$ 为 a 的 Montgomery 逆。这样,如果能够有效地计算出 x,再配合 Montgomery 模乘的综合应用,以 Montgomery 数和 Montgomery 算法为基础的密码应用实现的性能将会大大提高。下面给出原始的 Montgomery 逆算法[1]:

算法 A:

Phase I

Input: $a\in[1,p-1]$, p
Output: $r\in[1,p-1]$ and k, where $r=a^{-1}2^k\pmod p$, $n\leqslant k\leqslant 2n$
1. $u=p, v=a, r=0, s=1$
2. $k=0$
3. while($v>0$)
4. if u is even then $u=u/2, s=2s$
5. else if v is even then $v=v/2, r=2r$
6. else if $u>v$ then $u=(u-v)/2, r=r+s, s=2s$
7. else if $v\geqslant u$ then $v=(v-u)/2, s=s+r, r=2r$
8. $k=k+1$
9. if $r\geqslant p$ then $r=r-p$
10. return $r=p-r$ and k

Phase II

Input: $r\in[1,p-1]$, p, and k from Phase I.
Output: $x\in[1,p-1]$, where $x=a^{-1}2^n\pmod p$
11. for $i=1$ to $k-n$ do
12. if r is even then $r=r/2$

*)863 基金项目 2003AAIZ1270。邓 锐 硕士研究生;周玉洁 教授,博士生导师。

13. else then $r=(r+p)/2$
14. return $x=r$

这个算法的正确性是基于这样 3 个事实[1]：

1) Phase I 的每一轮循环中如下等式始终成立：

$$p=us+vr,\ s\geqslant 1,u\geqslant 1,0\leqslant v\leqslant a \tag{1}$$

2) Phase I 至少进行 n 轮循环，至多进行 $2n$ 轮循环，因此 $n\leqslant k\leqslant 2n$。

3) 在 Phase I 的每一轮循环中，下面等式始终成立：

$$ar=u2^k \pmod p \tag{2}$$

$$as=v2^k \pmod p \tag{3}$$

文[2]对这个算法的 PhaseII 进行了改进。由于 PhaseII 的算法是基于模 2 基的，迭代次数太多，比较慢。因此文[2]把 PhaseII 的计算用 Montgomery 乘来完成，在计算机软件实现中用高的基数来实现是比较快的。考虑到计算机字长的特点，文[2]对 Montgomery 模乘中的 R 也做了修改，使得算法对参数的限制有所放宽。将 $R=2^n,n=\lceil \log_2 p \rceil$，修改为 $R=2^m$，其中 m 是计算机中字长 w 的整数倍。设 $m=iw$，则 $(i-1)w<n\leqslant iw$。经过修改后，Phase I 的输入 $a\in[1,2^m-1]$，而输出中 $n\leqslant k\leqslant m+n$。定义修改过的 Phase I 为 AlmMonInv 算法。文[2]利用这个算法结合 Montgomery 模乘，能够快速地算出一个 Montgomery 数在其相应的 Montgomery 数域中的逆。

算法 B：

Input: $a2^m \pmod p$, p, n, and m
Output: $x=a^{-1}2^m \pmod p$, where $x\in[1,p-1]$
1. $(r,k)=AlmMonInv(a2^m),r=a^{-1}2^{-m}2^k \pmod p$ and $n\leqslant k\leqslant m+n$
2. If $n\leqslant k\leqslant m$ then
2.1 $r=MonPro(r,R^2)=(a^{-1}2^{-m}2^k)(2^{2m})(2^{-m})=a^{-1}2^k \pmod p$
2.2 $k=k+m>m$
3. $r=MonPro(r,R^2)=(a^{-1}2^{-m}2^k)(2^{2m})(2^{-m})=a^{-1}2^k \pmod p$
4. $r=MonPro(r,2^{2m-k})=(a^{-1}2^k)(2^{2m-k})(2^{-m})=a^{-1}2^m \pmod p$
5. return $x=r$, where $x=a^{-1}2^m \pmod p$

而文[7]只是利用 $m=n$ 的特殊性把上面的算法重写，可以减少一次 Montgomery 模乘。

3 对 Montgomery 逆算法的改进

本文在算法 B 的基础上主要对 AlmMonInv 算法做改进。我们以算法 A 的 Phase I 来阐述问题。首先注意到在最后一轮循环之前 u 和 v 的值都是 1，由 (3)式可知：

$as=2^k \pmod p$，因此 $s=a^{-1}2^k \pmod p$，s 即为所求。这样，我们就不必进行最后一轮的循环了。另外，我们还注意到，u,v 的值都是单调递减的，并且在最后一轮之前它们都不可能相等，否则由于 $\gcd(a,p)=\gcd(u,v)$，将导致 a,p 不互素的矛盾结果。所以 while 循环的判断条件可以改为 while $(u,v\neq 1)$。更进一步，其实只要 u,v 中有一个是 1，循环就可以结束了，我们可以通过(2)、(3)式得到结果。这样，while 循环的条件进一步改成 while$(u\neq 1,v\neq 1)$，于是就在 AlmMonInv 算法基础上得到一个初步改进的算法 C。

算法 C：

Input: $a\in[1,2^m,-1],p$
Output: $x\in[1,p-1]$ and k, where $x=a^{-1}2^k \pmod p$, $0\leqslant k\leqslant m+n-1$
1. $u=p,v=a,r=0,s=1$
2. $k=0$
3. while$(u\neq 1,v\neq 1)$
4. if u is even then $u=u/2,s=2s$
5. else if v is even then $v=v/2,r=2r$
6. else if $u>v$ then $u(u-v)/2,r=r+s,s=2s$
7. else if $v\geqslant u$ then $v=(v-u)/2,s=s+r,r=2r$
8. $k=k+1$
9. if $v=1$ then $x=s$
10. else $x=p-r$
11. return x and k

这样算法 C 就比原来的算法至少要少做最后一次循环和最后两次减法，并且也减少了中间的许多循环，当 u,v 有一为 1 时循环就结束了。

但是算法 C 依然是以 2 为基数来进行计算的，迭代次数还是比较多。因此，我们可以尝试用 4 为基或 8 为基来进行计算。考虑 u,v 模 4 的情况，分类如下：

1) $u\equiv 0 \pmod 4$ 或 $v\equiv 0 \pmod 4$

2) $u\equiv v \pmod 4$ 不含 1)的情况。

3) $u\equiv -v \pmod 4$，不含 1)的情况。

4) u,v 模 4 的其他组合，$(1,2),(-1,2),(2,-1),(2,1)$

通过上面的分类，我们提出如下的 4 为基的改进算法 D。

算法 D：

Input: $a\in[1,2^m-1],p$
Output: $x\in[1,p-1]$ and k, where $x=a^{-1}2^k \pmod p$, $0\leqslant k\leqslant m+n-1$
1. $u=p,v=a,r=0,s=1,signr=-1$
2. $k=0$
3. while$(u\neq 1,v\neq 1)$
4. $c=u \bmod 4,d=v \bmod 4$.
5. if $c=0$ then $u=u/4,s=4s$
6. else if $d=0$ then $v=v/4,r=4r$
7. else if $c=d$ then
if $u>v$ then $u=(u-v)/4,r=r+s,s=4s$
else $v=(v-u)/4$, $s=s+r,r=4r$
8. else if $c=-d$ if then
if $u>v$ then
if $u>2v$ then $u=(u-3v)/4,r=r+3s,s=4s$
if $u<0$, then $s=-s,u=-u,signr=-signr$
else $u=(u-v)/2,r=r+s,s=2s,k=k+1$, goto step3.
else
if $v>2u$ then $v=(v-3u)/4,s=s+3r,r=4r$
if $v<0$, then $r=-r,v=-v,signr=-signr$
else $v=(v-u)/2,s=s+r,r=2r,k=k+1$, goto step3.
9. else if $c=2$
if $>v+v/2$ then $u=(u-2v)/4,r=r+2s,s=4s$
if $u<0,s=-s,u=-u,signr=-signr$
else $u=u/2,s=2s,k=k+1$, goto step 3.
10. else if $d=2$
if $v>u+u/2$ then $v=(v-2u)/4,s=s+2r,r=4r$
if $v<0,r=-r,v=-v,signr=-signr$
else $v=v/2,r=2r,k=k+1$, goto step3
11. $k=k+2$
循环体至此结束。
12. if $u=1$ then
if $signr=-1$ then
if $r>0$ then $x=p-r$
else $x=-r$
else
if $r>0$ then $x=p+r$
else $x=r$
13. else $(v=1)$
if $signr=-1$ then
if $s>0$ then $x=s$
else $x=p+s$
else
if $s>0$ then $x=p-s$
else $x=-s$
14. return x and k

下面我们来证明算法 D 的正确性。为此，我们只须考虑算法 A 所依赖的 3 个事实就可以了。对应算法 D，我们将(2)、(3)式相应改写成如下(4)、(5)式：

$$ar=gu2^k \pmod p \tag{4}$$

$$as=-gv2^k \pmod p \tag{5}$$

由于在算法中 u,v 的符号可能会变，因此我们加入 g 来调整符号，这里的 g 就对应算法当中的 $signr$。容易验证，算法始终能保证(4)、(5)两式成立。

其次，算法也能始终保证(1)式的成立，并且 u,v 一直都是正数，但是 r,s 的符号可能为负。但显然它们不能同时为负，否则和(1)式矛盾。下面我们证明：

定理 1 算法 D 循环结束后给出的结果 $r(u=1$ 时)或 $s(v=1$ 时)的值在 $(-p,p)$ 里。

由于算法里只有步骤 8,9,10 会产生符号问题。我们以

步骤 8 为例来证明。9,10 的情况是类似的,并且更简单。

由于 u,v 的对称性,我们不妨假设在某一轮中 $2v_0<u_0<3v_0, r_0, ss_0\geqslant 0$(这和算法之初的情况符合)并在步骤 8 里出现了 $u_1=(u_0-3v_0)/4<0$ 的情况。这样,随后符号被纠正,则 $u_1=(3v_0-u_0)/4, s_1=-4s_0, r_1=r_0+3s_0>0, v_1=v_0$ 不变。此时,我们有 $u_1<v_1$。注意到在随后的下一轮计算中,只有步骤 5,7,8,9 可能执行,由于 $u_1<v_1$,因此步骤 5 和 9 并不改变 us 的绝对值。

考虑步骤 8,由于 $2u_1<v_1$,因此 $v_2(v_1-3u_1)/4=(3u_0-5v_0)/16>0, u_2=u_1, s_2=s_1+3r_1=3r_0+5s_0, r_2=4r_0+12s_0>0$。则要么所有的参数都纠正到[1,$p-1$]的范围,要么由于 $s_2<0$,us 绝对值减小。另外,如果算法是经过步骤 5 或 9 之后才到步骤 8 的,由于 s 绝对值的增加,新的 s 值未必是正数,但是 us 的绝对值却减小了。

考虑步骤 7,由于 $u_1<v_1$,因此 $v_2(v_1-u_1)/4>0, r_2=4r_1>0, s_2=s_1+r_1=r_0-s_0$ 符号未定。但是,如果 $s_2>0$,则由于(1)式的约束,所有参数都回到[1,$p-1$]范围。如果 $s_2<0$,则 us 的绝对值变小了。同样,如果算法是经过步骤 5 或 9 才到步骤 7 的,则计算后至少 us 的绝对值变小了。

在随后的计算中,要么 s 保持负号但 us 绝对值变小,要么 s 变成正数。所以,结合上面的讨论,最坏的情况就是:

I:s 保持负号,$|u_1s_1|\geqslant p$,并且 $u_1=1$。此时,$u_0-3v_0=-4$。因此,不等式转化为 $4s_0\geqslant(3v_0-4)s_0+r_0v_0$。这样,$v_0$ 只能取 2,从而 $u_0=2$,这和 a,p 互素是矛盾的,所以不等式不成立。并且,此时由于(1)的约束,vr 的值也在$(0,2p)$内,由于 $u_1=1$,因此 v_1 至少是 3,此时的 r 也在$(0,p)$之内,结果符合要求。

II:s 变为正数,但同时 r 变为负数。这种情况只会发生在步骤 8 或 10 中,我们在这里证明 8 的情况,10 的情况是类似的。设在某一次循环中,$2u_i<v_i<3u_i, s_i<0, r_i>0$。并且 $|u_is_i|<p, v_ir_i<2p$。这个假设是合理的。在经过步骤 8 计算后,$v_{i+1}=(3u_i-v)/4, r_{i+1}=4r_i$。但是由于 $v_i>2u_i$,因此 $v_{i+1}r_{i+1}$ 的绝对值至多是原来的一半,$|v_{i+1}r_{i+1}|<p$,因此 $u_{i+1}s_{i+1}<2p$,即使 $v_{i+1}=1$。但由于 $u_{i+1}\geqslant 3$,因此 $s_{i+1}\in(0,p)$。

这样,我们就证明了在任意一次循环之后,如果 u_is_i, v_ir_i 中有一为负,不妨设 $u_is_i<0$,则 $|u_is_i|<p, v_ir_i<2p$。这样,不论 $u_i=1$ 还是 $v_i=1, s_i$ 和 r_i 都在$(-p,p)$内。

接下来我们证明:

定理 2 算法 D 的输出 k 值满足 $0\leqslant k\leqslant n+m-1$。

从整个算法可以看出,中心思想就是不断地削减 uv 的值,直到 u,v 中有一方变成 1。因此 $2^k\leqslant ap/2$,所以 $0\leqslant k\leqslant n+m-1$。

这样,我们就证明完了算法 D 的正确性。另外,算法 B 可以直接利用算法 D,这是因为算法 B 的步骤 2 中,$k\geqslant n$ 并不是必须的,所以可以直接改成"if $k\leqslant m$ then"即可。在算法 D 中取 $m=n$,文[7]也可以直接利用本算法。算法 D 也可以运用于文[1,2,7]中的基本模逆算法。

4 算法 D 的效率估计

下面我们来粗略估计一下算法 D 的效率。算法 D 有 6 个大的判断分支,比算法 A 多两个。而算法 D 中的分支 8 内部还有再一次的比较分支,但都是 u,v 之间的比较。我们这里并不考虑 c,d 之间的比较。另外,$u/2, v/2$ 都是移位运算。从整个运算来看,都是加减法,像 $3u, 4s$ 这样的运算都可以用加法和移位来代替。

因此整体看来,算法 D 平均每次循环比算法 A 要多大约一次加法或比较的运算量,复杂度基本没有上升。但是,算法 D 所迭代的次数是相当少的。我们来估计一下其迭代次数的平均上限。为了估计方便,我们在这里取 $m=n$。由算法可以看出:

定理 3 除了分支 9,10 的第二个子分支外,其他分支都可以使 uv 的值至少降为原来的 1/4。

分支 5,6,7 满足这个条件是明显的。下面我们证明分支 8 的情况。由于 u,v 的对称性,我们只须证明 $u>v$ 的情况。当 $u-3v>0$ 时,显然是满足的。如果 $2v<u<3v$,设 $u=2v+l, 0<l<v$,则新的 uv 值为 $(v-l)v/4<(2v+l)v/4=uv/4$,因此也满足。当 $v<u<2v$ 时,$u-2v<0$,所以 $2u-2v<u$,则 $(u-v)v/2<uv/4$,所以满足要求。这样,我们就证明了分支 8 的情形。

下面我们具体看分支 9,10 的情形。由于分支 9,10 的对称性,我们只须看分支 9 的情形即可。当 $u<3v/2$ 时,显然只能将 uv 的值缩小为原来的一半。当 $u>3v/2$ 时,如果 $u>2v$,则可以满足要求。当 $3v/2<u<2v$ 时,由于 $(2v-u)v/4<uv/4$,因此也满足要求。综合以上的讨论,只有 9,10 里面的第二个分支会造成只能将 uv 减半的情形。而这种情况出现的概率是 1/6,因为 $(c,d)=(2,2),(2,0),(0,0),(0,2)$ 的情形是不可能出现的。因此,平均迭代次数上限是 $\frac{5}{6}*n\times\frac{1}{6}*2n=\frac{7}{6}n$。由于 a 平均有 $n/2$ 个比特,因此平均迭代次数是 $\frac{5}{6}*\frac{3}{4}n+\frac{1}{6}*\frac{3}{2}n=\frac{7}{8}n$。而实际运算时所用的迭代次数可能比这个要小得多,因为我们只要求 u,v 中有 1 个被削减为 1 就可以结束了。

5 以 8 为基的算法

以 8 为基的算法和以 4 为基的算法是类似的,并且有些部分将用以 4 为基的方法来计算。此时,情况分支也多了。

1)$u\equiv 0(\mathrm{mod}\ 8)$或 $v\equiv 0(\mathrm{mod}\ 8)$,除以 8 即可。

2)$u\equiv 4(\mathrm{mod}\ 8)$或 $v\equiv 4(\mathrm{mod}\ 8)$,除以 4 即可。

3)$u\equiv v(\mathrm{mod}\ 8)$,作差之后除以 8 即可。

4)$(\pm1,\pm2),(\pm2,\pm1),(\pm2,\pm3),(\pm3,\pm2)$的情形,借用算法 D 的分支 9,10 的方法来处理。

5)$(1,-3),(-1,3),(-3,1),(3,-1)$的情形,作差除以 4 即可。

6)$(1,3),(-1,-3),(3,1),(-3,-1),(1,-1),(-1,1)$的情形,借用算法 D 的分支 8 的方法即可。

由于以 8 为基的算法分支比较多,和以 4 为基的算法在原理和流程上也没有大的区别,这里就不具体列出 8 基算法的伪代码了。但是,我们这里将具体给出其效率方面的数据估计。

通过枚举所有 64 种情况,发现只有 48 种可能出现(除去 u,v 不互素的情形)。而在这 48 种情况中,能将 uv 降为原来的 1/8 的概率为 9/24,降为原来的 1/4 的概率为 11/24,降为原来的 1/2 的概率为 1/6,所以迭代次数的平均上限是 $\frac{9}{24}*\frac{2n}{3}+\frac{11}{24}n+\frac{1}{6}*2n=\frac{25}{24}n$,而平均的迭代次数是 $\frac{25}{32}n$。这个结

果当然比4基算法要好，但同时整个程序的分支也变得比较多，程序也显得比较复杂。但对于大素数的情形，这还是值得的。

6 算法的应用

在RSA公钥算法中大量用到的模幂运算，可以用Montgomery乘和指数加减链的方法来实现[3]。要用加减链，就要求底数的Montgomery逆。如果加减链的单位是多比特的话，所需要算的Montgomery逆就更多了，因此这个算法可以用来加速以加减链为基础的模幂运算。虽然在讨论中我们假定 p 是素数，其实我们只要求输入参数互素即可。这在RSA实际应用是可以保证的，因此算法是可以运用在RSA中的。

另外，在现在广为应用的椭圆曲线密码体制中，倍点运算一直是运算的瓶颈。考虑素域 F_p 上的椭圆曲线（p 是奇素数）$F(x,y)$，对于上面的点 p,Q，要计算 $p+Q,2p$，都将要计算 $\lambda=\frac{y_2-y_1}{x_2-x_1}$ 和 $\lambda=\frac{3x_1^2+a}{2y_1}$。一般，我们是将平面坐标转换成射影坐标，然后将所有参数都转化到对应的Montgomery数，用Montgomery模乘来完成计算，这是以乘法来换取求逆运算。现在我们可以直接利用Montgomery逆来进行计算，因为整个Montgomery逆算法要比普通的求逆算法要快得多。

同时，正如我们前面提到的，本文提出的算法也可以运用于求基本的模逆。

结论 本文通过对Montgomery逆算法核心部分进行两次改进，得到了两种分别以4为基和以8为基的快速算法。其中算法D是以4为基的算法，在基本没有增加算法实现复杂度的基础上，将迭代次数的平均上限从 $2n$ 降到了 $\frac{7}{6}n$，平均迭代次数也从 $\frac{3n}{2}$ 降到了 $\frac{7}{8}n$。而以8为基的算法，其复杂度会稍有增高，主要是内部分支增多，但体现在程序上也只是增加了一些判断和跳转的运算量，其迭代次数的平均上限则降到了 $\frac{25}{24}n$，平均迭代次数也降到了 $\frac{25}{32}n$。另外，由于新算法只要求 u,v 中有一个变为1就可以结束操作，因此实际的迭代次数可能比上述的还要少。同时，这两种算法都可以运用于求基本的模逆。两种算法比较而言，一般情况下建议用4基算法，它效率较高，算法实现也是比较轻量级的。但对于特别大的素数的应用场合，在8基算法的多分支比较跳转所造成的开销相对可以接受的情况下，还是建议用8基算法，它的效率更高。本文所提出的算法在RSA公钥体制以及ECC公钥体制实现中有广泛的应用。

参考文献

1 Kaliski B S Jr. The Montgomery Inverse and its Applications [J]. IEEE Trans on. Computers, 1995, 44(8): 1064～1065

2 Savas E, Koc C K. The Montgomery Modular Inverse-Revisited [J]. IEEE Trans on Computers, 2000, 49(7): 763～766

3 Gutub A. A scalable hardware for montgomery modular inverse computation: [Tech Report]. Corvallis, Oregon 97331: Information Security Laboratory, Electrical and Computer Engineering Department, Oregon State University, 2001

4 Gutub A, Tenca A F, Koc C K. Scalable VLSI architecture for GF(p) Montgomery modular inverse computation [C]. In: Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002. 53～58

5 L'orencz R'. New Algorithm for Classical Modular Inverse [C]. In: 4th International Workshop on Cryptographic Hardware and Embedded Systems, 2002. 57～70

6 Gutub A A A, Tenca A F. Efficient scalable hardware architecture for montgomery inverse computation in GF(p). In: Signal Processing Systems, 2003, SIPS 27～29

7 McIvor C, McLoone M, McCanny J V. Improved Montgomery modular inverse algorithm [J]. IEEE Electronics Letters, 2004, 40(18)

8 Menezes A, van Oorschot P, Vanstone S. Handbook of Applied Cryptography. CRC Press, 1996

9 Knuth D E. The Art of computer Programming, vol 2. Addison-Wesley, 1981

（上接第120页）

结束语 本文提出了将CBR用于IDS中减少漏报率的思想，也提出了在基于规则的IDS中应用CBR时借助特征可视化手段确定案例中各特征的权值的思路；介绍了实现CBR的步骤、CBR的推理过程和所采用的算法；给出了在Snort上的实验结果。实验证明：CBR方法中对于度量特征明晰的案例的推理效果较之基于规则的方法有了长足的进步，可以有效地检测出Snort无法检测的由已知攻击变异出的攻击，效果良好。

需要指出的是，在案例不多的情况下，CBR的案例并不一定用数据库存储。另外，如果在CBR的推理过程中，案例修改或产生的新案例涉及到特征权值的重确定，那就需要在线估算特征权值，这是本文要进一步完善的工作。

参考文献

1 Dutta S, Wierenga B, Dalebout A. Case-based reasoning systems: from automation to decision-aiding and stimulation. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(6): 911～922

2 王万森. 人工智能原理及其应用[M]. 北京：电子工业出版社，2000

3 Han Jiawei, Kamber M. Data mining: Concepts and Techniques. San Francisco: Morgan Kaufmann Publishers, Inc, 2001

4 Wettschereck D, Aha D W, Mohri T. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review, 1997, 11: 273～314

5 http://www.ll.mit.edu/SST/ideral/1998/1998data/index.html