素数域上亏格为3的超椭圆曲线快速算法

郝艳华　谭示崇　王育民
（西安电子科技大学 ISN 国家重点实验室　西安 710071）

摘　要　本文给出了素数域上亏格为3的超椭圆曲线退化除子加法和倍点运算的确定性公式，这些公式在有固定基点的超椭圆曲线密码算法，如 ElGamal 型加密算法、Diffie-Hellman 协议的发送方及 HECDSA 的标量乘算法中都有应用。与标准除子标量乘算法相比，给出的1次和2次退化除子标量乘算法可分别获得 33.4%和 16.7%的加速，同时基点的表示长度可压缩至标准除子表示长度的 1/3 或 2/3。

关键词　亏格为3的超椭圆曲线，确定性公式，标量乘，退化除子

Fast Arithmetic of Genus 3 Hyperelliptic Curves over Prime Fields

HAO Yan-Hua　TAN Shi-Chong　WANG Yu-Min
(National Key Laboratory on ISN, Xidian University, Xi'an710071)

Abstract　Explicit formulae for addition and doubling on genus 3 hyperelliptic curve over prime fields using degenerate divisors are presented, which can be applied to scalar multiplications of hyperelliptic curve cryptosystems with a fixed base point, e. g. , ElGamal-type encryption, the sender of Diffie-Hellman and HECDSA. Compared with scalar multiplications using standard divisors, the proposed scheme using degenerate divisors of degree 1 or 2 can attain a speed-up of approximately 33.4% and 16.7%, respectively. At the same time, the representation of the base point can be compressed to one third or two thirds of standard divisors.

Keywords　Genus 3 hyperelliptic curves, Explicit formula, Scalar multiplication, Degenerate divisor

1　引言

近年来，由于具有密钥长度短、每比特安全性高及签名速度快等这些其他公钥密码体制所无法比拟的优点，椭圆曲线密码体制(ECC)受到了越来越多的理论密码学家、应用密码学家及数学家们的青睐。而作为 ECC($g=1$)推广的具有更高亏格($g=2,3$)的超椭圆曲线密码体制(HECC)与 ECC 相比，具有密钥长度更短、每比特安全性更高的特点，如 160 比特 ECC 所能达到的安全强度与 80 比特亏格为2的 HECC 所能达到的安全强度相当，而只相当于 56 比特亏格为3的 HECC 所能达到的安全强度，因此在 64 比特 CPU 上实现亏格为3的 HECC 具有不需要多精度计算的好处，这是亏格为2的 HECC 无法做到的。但是，由于超椭圆曲线(HEC)的结构更加复杂(HEC 上有理点不再构成可换群)，HECC 的实现速度成为了阻碍它走向实用的最重要的因素，这是目前许多密码学家所竭力要解决的一个问题。

第一个在这方面作出突出贡献的人是 Cantor[1]，他给出了第一个可用于实现的 HEC 上 Jacobian 群的加法算法。该算法适用于任何特征域上任意亏格的 HEC，因此不是一个确定性的算法，结果表明它的运算效率很低。13 年后，Harley[2,3]给出了素数域上亏格为2的 HEC 上 Jacobian 群加法和倍点算法的第一个确定性公式。与 Cantor 算法相比，Harley 算法是 Cantor 算法的一种特殊情况，但是由于多加了一些限定条件，如由于在素数域中考虑，因此不失一般性可认为 $h(x)=0$、由 $g=2$ 带来的计算过程中的一些便利及为简化运算所作的一些假定等。Harley 算法的效率要比同样亏格的 Cantor 算法高得多。它更重要的意义在于为那些致力于改进 HECC 实现速度的人指明了方向。随后在 2004 年，GMA[4]给出了素数域上亏格为3的 HEC 上 Jacobian 群加法和倍点算法的确定性公式，经过估算，所需运算量分别为 $1I+70M$ 和 $1I+71M$，其中 I 和 M 分别表示域中一次求逆运算和一次乘法运算所需要的时间。如果按照 $1I=5.2M$[5]计算，这个运算量只是使用最初的 Cantor 算法所需运算量的 1/3 左右。以上算法中考虑的输入除子均为 Jacobian 群中最常见的 g 次除子。

在 HECC 的 ElGamal 型加密算法、Diffie-Hellman 协议的发送方及签名算法 HECDSA 中都用到了固定基点的标量乘法。标量乘法包括两个最基本的运算：除子加法运算和除子倍点运算。因此，对除子加法和倍点运算速度的改进有助于提高这些密码算法的实现效率。KKA[6]证明了使用退化除子作为基点的超椭圆曲线离散对数问题与使用任意除子作为基点的超椭圆曲线离散对数问题同样困难。由于退化除子的结构性质使得它的标量乘法比标准除子的标量乘法要快，同时根据 KKA[6]的证明，取退化除子作为基点又不影响算法的安全性。因此，如何给出超椭圆曲线上退化除子更高效的加法和倍点算法的确定性公式，就成为了当前一些密码学家所要研究解决的问题。

2　背景知识

有限域 F 上亏格为 g 的超椭圆曲线是指所有满足方程

郝艳华　博士生，研究方向为超椭圆曲线密码体制和信息安全；**谭示崇**　博士生，研究方向为信息安全与密码学；**王育民**　教授，博士生导师，长期从事信息论、编码与密码学的教学和科研工作。

$y^2+h(x)y=f(x)$，但不同时满足两个偏导数方程 $2y+h(x)=0$ 和 $h'(x)y-f'(x)=0$ 的解 $(x,y)\in\bar{F}\times\bar{F}$ 的集合，其中 $f(x)$ 是 $F[x]$ 上 $2g+1$ 次首一多项式，$h(x)$ 是 $F[x]$ 上至多 g 次的多项式，$\bar{F}$ 是 F 的代数闭包。当 F 为奇特征域时，不失一般性，有 $h(x)=0$。

有限域 F_P（p 为奇素数）上亏格为 3 的超椭圆曲线方程可写为

$$C:\ y^2=x^7+f_6x^6+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$$

当 $p\neq 7$ 时，通过双有理变换 $(x,y)\to(x+\frac{f_6}{7},y)$，$C$ 能够变换为 $f_6=0$ 的超椭圆曲线方程，二者 F_P 在上是同构的，因此本文中我们总假定 $f_6=0$。

与 EC 不同，HEC 上的有理点并不构成群。构成 HEC 上 Abel 群的是 C 上的 Jacobian 簇，记为 $J_c(F_P)$。Jacobian 群中的每一个元素都可由一个约化除子唯一表示。约化除子 D 由 $F_P[x]$ 上满足下列条件的两个多项式 $u(x)$，$v(x)$ 表示：(1) $u(x)$ 是首一多项式；(2) $\deg v(x)<\deg u(x)\leqslant g$；(3) $u(x)\mid v^2(x)-f(x)$。记 $D=(u,v)$，本文以下给出的除子均为约化除子。定义 D 的次数等于 $u(x)$ 的次数，记为 degD。$\deg D=g$ 时，称 D 为标准除子；$\deg D<g$ 时，称 D 为退化除子。

超椭圆曲线离散对数问题（HECDLP）是指给定 $P,Q\in J_c(F_P)$，确定出整数 m，使得 $Q=mP$（如果这样的 m 存在）。

HECC 是 ECC 的推广，因此一些对 ECC 有效的攻击算法同样对 HECC 有效，比如过去最经典的计算标量乘法的倍点加算法就容易遭受简单能量分析攻（SPA）或时间攻击（TA），从而容易暴露出对应标量的比特 Hamming 重量。目前计算标量乘法的标准算法——例行倍点加算法（double-and-add always method）则免除了这方面的困扰[7]。

例行倍点加算法（double-and-add always method）

输入：$d=(d_{n-1}\cdots d_0)_2$，$D\in J_c(F_P)$，$(d_{n-1}=1)$；

输出：dD。

(1) $D[0]=D$；

(2) for i from n−2 to 0 do

(3) $D[0]=2D[0]$，$D[1]=D[0]+D$，$D[0]=D[d_i]$；

(4) 输出 $D[0]$

对超椭圆曲线密码体制感兴趣的读者可以参见文[8]。

3 退化除子的标量乘法[5]

这里计算标量乘法所采用的是标准的例行倍点加算法。取 D 为退化除子，以下采用符号 ADD($i+j\to k$) 表示一个 i 次除子与一个 j 次除子相加生成一个 k 次除子的除子加法算法，DBL($i\to j$) 表示一个 i 次除子与自身相加生成一个 j 次除子的除子倍点算法，DBL 表示标准除子的倍点算法。记 $D_0=\mathrm{DBL}$，$D_1=\mathrm{DBL}(1\to 2)$，$D_2=\mathrm{DBL}(2\to 3)$，$A_1=\mathrm{ADD}(3+1\to 3)$，$A_2=\mathrm{ADD}(1+2\to 3)$，$A_3=\mathrm{ADD}(3+2\to 3)$。

当 $d_{n-2}=0$ 时，例行倍点加算法的加法链中最初几个为

$$D\to 2D\to 3D\to 4D\to 5D$$

当 $d_{n-2}=1$ 时，例行倍点加算法的加法链中最初几个为

$$D\to 2D\to 3D\to 6D\to 7D$$

取 $\deg D=1$，当 $d_{n-2}=0$ 时，完成一次标量乘法所需要的基本运算依次为 $D_1A_2D_2A_1\underbrace{D_0A_1\cdots D_0A_1}_{n-3\text{个}}$；当 $d_{n-2}=1$ 时，完成一次标量乘法所需要的基本运算依次为 $D_1A_2D_0A_1\underbrace{D_0A_1\cdots D_0A_1}_{n-3\text{个}}$。尽管当 d_{n-2} 取值不同时，第二轮循环中的除子倍点算法不同，但由于其他运算均相同，攻击者并不能得到关于 d 的有关其他比特的任何信息，因此该标量乘算法仍是抗 SPA 或 TA 的。

取 $\deg D=2$，不管 d_{n-2} 取值如何，完成一次标量乘法所需要的基本运算总是 $D_2A_3\underbrace{D_0A_3\cdots D_0A_3}_{n-2\text{个}}$。

4 退化除子的加法和倍点运算的确定性公式

为了实现退化除子的标量乘法，现在我们只需要给出 A_1，A_2，A_3，D_1，D_2 的确定性公式。下面我们首先给出进行这些算法所需要的限定条件，其中有一些是在 Harley 算法中已经给出的，还有一些是简化公式的过程中需要添加的假定条件。当这些条件不满足时，需要调用 Cantor 算法或是换一个退化除子做基点。但是由于我们的限定条件包括了绝大多数的情况，因此其他情形基本上可以忽略不计。记 $D_1=(u_1,v_1)$，$D_2=(u_2,v_2)$，$D_3=D_1+D_2$。

A_1：$\deg D_2=1$，$\deg D_1=\deg D_3=3$，$\gcd(u_1,u_2)=1$；

A_2：$\deg D_1=1$，$\deg D_2=2$，$\deg D_3=3$，$D_2=2D_1$，$2v_1\neq 0$；注意 $D_2=2D_1$ 是上节算法要求的；

A_3：$\deg D_2=2$，$\deg D_1=\deg D_3=3$，$\gcd(u_1,u_2)=1$；

D_1：$\deg D_1=1$，$\deg D3=2$，$D_1=D_2$，$2v_1\neq 0$；

D_2：$\deg D_1=2$，$\deg D_3=3$，$D_1=D_2$，$\gcd(u_1,2v_1)=1$。

由于篇幅所限，我们将这些算法的确定性公式列在本文的附录中，下面仅简单给出我们估算的这些算法所需要的运算量。注意，由于域中加、减法运算比乘法和求逆运算快得多，因此在下面的估算中这些运算均没有考虑在内。

算法	运算量
A_1	$1I+20M$
A_2	$1I+15M$
A_3	$1I+45M$
D_1	$1I+7M$
D_2	$1I+34M$

下面我们将退化除子标量乘法的运算量和标准除子标量乘法的运算量进行比较。DBL 运算量的估计值我们采用的是文[4]中给出的结果，即 $1I+71M$。为达到最低的安全水平，d 取 160 比特的大整数，此时标准除子标量乘法的运算量为 $159[(1I+70M)+(1I+71M)]=318I+22419M$。而使用我们给出的算法，当 $\deg D=1$ 时，由于 d_{n-2} 取值不同时 dD 会有不用的运算量，因此取平均值 $(1I+7M)+(1I+15M)+1/2[(1I+71M)+(1I+34M)]+(1I+20M)+157[(1I+71M)+(1I+20M)]=318I+14381.5M$，按照 $1I=5.2M$[5]，经过计算，一次除子标量乘法要比标准除子标量乘法大约快 33.4%，当 $\deg D=2$ 时，dD 的运算量为 $(1I+34M)+(1I+45M)+158[(1I+71M)+(1I+45M)]=318I+18407M$。经过计算，二次除子标量乘法要比标准除子标量乘法大约快16.7%。

5 公式优化过程中用到的技巧及说明

5.1 利用结式将多项式求逆化为常数求逆

在算法 3 的第 3 步中要求计算 $s=s_1x+s_0=\frac{v_2-v_1}{u_1}\bmod u_2$，我们首先计算 u_1 和 u_2 的结式 r，给出 $I=\frac{r}{u_1}\bmod u_2$ 的表

达式。注意到 r 是 u_1 和 u_2 的结式，所以这一步并不需要真的求逆。然后我们再计算 $s'(x)=rs\equiv(v_2-v_1)I \bmod u_2$。至此，为了计算 s，我们只需要计算 r^{-1} 就可以了，注意到 r 只是一个常数。这个技巧在算法 1 和算法 5 的第 3 步中都有应用。

5.2 Montgomery 同时求逆技巧

Montgomery 同时求逆的思想就是当求两个逆 a^{-1} 和 b^{-1} 时，可以通过计算 $t=(ab)^{-1}$，$a^{-1}=bt$，$b^{-1}=at$，只需要计算 1 个逆、3 个乘法运算就可以了。由于求逆运算通常比乘法运算慢得多，所以这一技巧经常被用来提高运算速度。算法 3、算法 5 都用到了这一技巧。

5.3 利用输入除子的结构简化运算

在 Cantor 算法[1]中，我们注意到一个很重要的步骤，就是要将 u_3 化为首一多项式。但是在算法 1 的第 5 步和算法 5 的第 6 步中，这一步骤可以省略。这是因为此时 v 是 3 次多项式，v^2 是 6 次多项式，而 f 是 7 次首一多项式，所以 $f-v^2$ 是首一的，而 u_1u_2 首一，所以求出的 $u_3(x)$ 本身就已经是首一多项式了。

5.4 巧求商

当求两个多项式 $a(x)$ 和 $b(x)$($\deg a(x)>\deg b(x)$)的商 $t(x)$ 的时候，$t(x)$ 各项的系数只和 $a(x)$ 和 $b(x)$ 的前 $\deg a(x)-\deg b(x)+1$ 个最高系数有关。当 $a(x)$ 和 $b(x)$ 都是首一多项式的时候，$t(x)$ 的各项系数就只和除最高系数之外的前 $\deg a(x)-\deg b(x)$ 个最高系数有关。利用这一点可以减少无谓的计算，算法 1、3、5 均利用了这一点。

5.5 利用性质求 v_3

当 $u_3=(x+u)^b$ 时，v_3 满足性质：$\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^j[v_3^2-f(x)]_{|x=-u}=0$，$0\leqslant j\leqslant b-1$，因此通过待定系数解方程组就可以求出 v_3 了，不需要再套用一般的方法。这一特点在算法 2 和 4 中都有应用。

5.6 利用 Karatsuba 公式[9]

利用 Karatsuba 公式计算 $(ax+b)(cx+d)=acx^2+(ad+bc)x+bd=acx^2+[(a+b)(c+d)-ac-bd]x+bd$，本来需要 4 个乘法运算，现在只要 3 个就可以了。Karatsuba 用在了算法 2 和算法 3 中。

5.7 利用 Toom 乘法[10]

利用 Toom 乘法给出一个三次多项式和一个一次多项式相乘的算法：

$(s_1x+s_0)(x^3+u_{12}x^2+u_{11}x+u_{10})=s_1x^4+(s_1u_{12}+s_0)x^3+(s_1u_{11}+s_0u_{12})x^2+(s_1u_{10}+s_0u_{11})x+s_0u_{10}=s_1x^4+k_3x^3+k_2x^2+k_1x+k_0$

$t_1=s_1u_{12}$，$k_3=t_1+s_0$，$k_0=s_0u_{10}$，$t_2=(u_{12}+u_{11}+u_{10})(s_0+s_1)$，$t_3=(u_{12}-u_{11}+u_{10})(s_0-s_1)$，$k_2=(-2k_0+t_2+t_3)/2k_1=(-2t_1+t_2-t_3)/2$，因此本来需要 6 个乘法的运算，现在只要 4 个就可以了。我们在算法 3 中用到了这个公式。

5.8 利用一些变形

利用一些变形也可以达到减少运算量的目的，如 $b^2+bc+c^2=(b+c)^2-bc$，$a^2\pm2a(b+c)+b^2+bc+c^2=a^2\pm2a(b+c)+(b+c)^2-bc=[a\pm(b+c)]^2-bc$。另外，如果前面的运算过程中已经求出了 $(a+b)^2$ 的值，那么利用公式 $a^3+b^3=(a+b)^3-3ab(a+b)=(a+b)[(a+b)^2-3ab]$ 可以简化运算。

结论 本文给出了素数域上亏格为 3 的超椭圆曲线退化除子加法和倍点运算的确定性公式，分析了公式优化中的一些技巧并对所需的运算量进行了估算。这些公式可用于超椭圆曲线密码体制中带有固定基点的标量乘算法(如 ElGamal 型加密算法、Diffie-Hellman 协议的发送方及签名算法 HECDSA 等)中。对我们的退化除子标量乘算法进行估算，结果表明当 d 取 160 比特长的大整数时，1 次和 2 次除子标量乘算法大约比标准除子标量乘算法分别快 33.4%和 16.7%。而且由于退化除子更低的次数，我们天然得到了一个基点的压缩表示，压缩量为标准除子的 1/3 或 2/3。

由于退化除子更简单的结构使得更多使用退化除子进行运算的密码算法能够获得更快的运算速度，同时也正由于退化除子特殊的结构使得一些密码学家担心使用退化除子的密码算法是否更易遭受某些目前未知的攻击。对退化除子的研究仍是目前研究的一个热点问题。

参考文献

1 Cantor D G. Computing in the Jacobian of A hyperelliptic curve. Math Comp ,1987,48: 95～101

2 Harley R. Adding. text. 2000. http://cristal. inria. fr/～harley/hyper/

3 Harley R. Doubling. c. 2000. http://cristal. inria. fr/～harley/hyper/

4 Gonda M, Matsuo K, Aoki K, et al. Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementations. The 2004 Symposium on Cryptography and Information Security, Sendai, Japan, 2004

5 Katagi M, Akishita T, Kitamura I, et al. Some improved algorithms for hyperelliptic curve cryptosystems using degenerate divisors. In: ICISC'04, 2005, LNCS 3506: 296～312

6 Katagi M, Kitamura I, Akishita T, et al. Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. WISA '04, 2004, LNCS 3325: 345～359

7 Coron J S. Resistance against differential power analysis for elliptic curve cryptosystems. CHES'99, 1999, LNCS1717: 292～302

8 Menezes A J, Wu Yi-Hong, Zuccherato R J. An elementary introduction to hyperelliptic curves: [Technical Report]. CORR96-19, 1996. http://www. cacr. math. uwaterloo. ca/

9 Karatsuba A, Ofman Y. Multiplication of multidigit numbers on automata. Soviet Physics Doklady, 1963, 7: 595～596

10 Toom A L. The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady, 1963, 3: 714～716

附录

算法 1　ADD(3+1→3)

输入：$y^2=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$; $D_1=(u_1, v_1)$, $D_2=(u_2, v_2)$; $u_1=x^3+u_{12}x^2+u_{11}x+u_{10}$, $v_1=v_{12}x^2+v_{11}x+v_{10}$; $u_2=x+u_{20}$, $v_2=v_{20}$;		
输出：$D_3=(u_3, v_3)$, $u_3=x^3+u_{32}x^2+u_{31}x+u_{30}$, $v_3=v_{32}x^2+v_{31}x+v_{30}$.		
步骤	过程	计算量
1	计算 $r=\mathrm{res}(u_1, u_2)$: $t_1=u_{20}-u_{12}$　$t_2=u_{20}t_1+u_{11}$　$t_3=u_{20}t_2$　$r=t_3-u_{10}$	$2M$
2	计算 $inv=r^{-1}$	$1I$
3	计算 $s_0=inv(v_2-v_1) \bmod u_2$: $t_1=v_{12}u_{20}$　$t_2=v_{11}-t_1$　$t_3=u_{20}t_2$　$t_4=t_3+v_{20}-v_{10}$　$s_0=t_4\ inv$	$3M$
4	计算 $v=s_0u_1+v_1=s_0x^3+k_2x^2+k_1x+k_0$: $t_1=s_0u_{10}$　$t_2=s_0u_{11}$　$t_3=s_0u_{12}$　$k_2=t_3+v_{12}$　$k_1=t_2+v_{11}$　$k_0=t_1+v_{10}$	$3M$
5	计算 $u_3=\dfrac{(f-v^2)}{(u_1u_2)}=x^3+u_{32}x^2+u_{31}x+u_{30}$: $t_1=s_0^2$　$t_2=u_{20}+u_{12}$　$u_{32}=-t_1-t_2$　$t_3=2s_0k_2+u_{11}$　$t_4=u_{12}u_{20}$　$t_5=t_2(t_1+t_2)$ $u_{31}=f_5-t_3+t_5-t_4$　$t_6=s_0k_1$　$t_7=k_2^2$　$t_8=t_2(t_4-u_{31})$　$t_9=u_{11}u_{12}$　$t_{10}=t_1(t_4+u_{11})$ $u_{30}=f_4-2t_6-t_7+t_8+t_9+t_{10}-u_{10}$	$9M$
6	计算 $v_3=(-v) \bmod u_3 \equiv v_{32}x^2+v_{31}x+v_{30}$: $t_1=s_0u_{30}$　$t_2=s_0u_{31}$　$t_3=s_0u_{32}$　$v_{30}=t_1-k_0$　$v_{31}=t_2-k_1$　$v_{32}=t_3-k_2$	$3M$
总计算量		$1I+20M$

算法 2　ADD(1+2→3)

输入：$y^2=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$; $D_1=(u_1, v_1)$, $D_2=(u_2, v_2)$, $D_2=2D_1$; $u_1=x+u_{10}$, $v_1=v_{10}$; $u_2=x^2+u_{21}x+u_{20}$, $v_2=v_{21}x+v_{20}$;		
输出：$D_3=(u_3, v_3)$, $u_3=x^3+u_{32}x^2+u_{31}x+u_{30}$, $v_3=v_{32}x^2+v_{31}x+v_{30}$.		
步骤	过程	计算量
1	计算 $u_3=(x+u_{10})^3=x^3+u_{32}x^2+u_{31}x+u_{30}$: $t_1=u_{10}u_{21}$　$u_{32}=u_{10}+u_{21}$　$u_{31}=u_{10}+t_1$　$u_{30}=u_{10}u_{20}$	$2M$
2	计算 $v_{32}=\dfrac{f''(-u_{10})}{4v_{10}}-\dfrac{(f'(-u_{10}))^2}{8v_{10}^3}$: $t_1=u_{10}^2$　$t_2=7t_1+5f_5$　$t_3=-u_{10}t_2+4f_4$　$t_4=-u_{10}t_3+3f_3$　$t_5=-u_{10}t_4+2f_2$ $t_6=-u_{10}t_5+f_1$　$t_7=(2v_{10})^{-1}$　$t_8=t_6t_7$　$t_9=21t_1+10f_5$　$t_{10}=-u_{10}t_9+6f_4$ $t_{11}=-u_{10}t_{10}+3f_3$　$t_{12}=-u_{10}t_{11}+f_2$　$t_{13}=t_8^2$　$v_{32}=t_7(t_{12}-t_{13})$	$1I+11M$
3	计算 $v_{31}=\dfrac{f'(-u_{10})}{2v_{10}}+2v_{32}u_{10}$: $t_1=v_{32}u_{10}$　$v_{31}=t_8+2t_1$	$1M$
4	计算 $v_{30}=v_{10}-u_{10}(v_{32}u_{10}-v_{31})$: $t_2=t_1-v_{31}$　$t_3=-u_{10}t_2$　$v_{30}=v_{10}+t_3$	$1M$
总计算量		$1I+15M$

算法 3　ADD(3+2→3)

输入：$y^2=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$; $D_1=(u_1, v_1)$, $D_2=(u_2, v_2)$; $u_1=x^3+u_{12}x^2+u_{11}x+u_{10}$, $v_1=v_{12}x^2+v_{11}x+v_{10}$, $u_2=x^2+u_{21}x+u_{20}$, $v_2=v_{21}x+v_{20}$;		
输出：$D_3=(u_3, v_3)$, $u_3=x^3+u_{32}x^2+u_{31}x+u_{30}$, $v_3=v_{32}x^2+v_{31}x+v_{30}$.		
步骤	过程	计算量
1	计算 $r=\mathrm{res}(u_1, u_2)$: $t_1=u_{20}-u_{11}$　$t_2=u_{12}-u_{21}$　$t_3=u_{21}t_1$　$t_4=u_{10}(t_3+u_{10})$　$t_5=u_{21}^2$　$t_6=u_{10}(t_5-2u_{20})$ $t_7=u_{12}u_{20}$　$t_8=u_{11}u_{21}$　$t_9=u_{20}(t_7-t_8)$　$t_{10}=t_2(t_6+t_9)$　$t_{11}=u_{20}t_1^2$　$r=t_4+t_{10}+t_{11}$	10*M*
2	计算 $I=i_1x+i_0\equiv r u_1^{-1} \bmod u_2$: $t_4=u_{12}u_{21}$　$i_1=t_4-t_5+t_1$　$t_6=u_{21}(t_4-t_5-u_{11})$　$t_7=u_{20}(2u_{21}-u_{12})$　$i_0=t_6+t_7+u_{10}$	3*M*
3	计算 $s'=s_1'x+s_0'=rs\equiv(v_2-v_1)I \bmod u_2$: $t_1=u_{21}v_{12}$　$t_2=v_{11}-v_{21}-t_1$　$t_5=i_1t_2$　$t_6=u_{20}t_5$　$t_7=u_{20}v_{12}$　$t_8=v_{20}-v_{10}+t_7$ $t_9=i_0t_8$　$s_0'=t_6+t_9$　$t_{10}=u_{21}t_5$　$t_{11}=(i_1+i_0)(t_8-t_2)$　$s_1'=t_{10}+t_{11}+t_5-t_9$	7*M*
4	计算 $s=s'/r=s_1x+s_0$: $t_1=r s_1'$　$t_2=(t_1)^{-1}$　$t_5=t_2r$　$t_6=t_2 s_1'$　$t_7=t_5r$　$s_1=t_6 s_1'$　$s_0=t_6 s_0'$	1*I*+6*M*
5	计算 $v=su_1+v_1=s_1x^4+k_3x^3+k_2x^2+k_1x+k_0$: $t_1=s_1u_{12}$　$k_3=t_1+s_0$　$t_2=s_0u_{10}$　$t_5=(u_{12}+u_{11}+u_{10})(s_1+s_0)$ $t_6=(u_{12}-u_{11}+u_{10})(s_0-s_1)$　$k_2=(-2t_2+t_5+t_6)/2+v_{12}$　$k_1=(-2t_1+t_5-t_6)/2+v_{11}$ $k_0=t_2+v_{10}$	4*M*
6	计算 $u_3=s_1^{-2}(-f+v^2)/(u_1u_2)=x^3+u_{32}x^2+u_{31}x+u_{30}$: $t_1=k_3t_7$　$t_2=t_7^2$　$t_5=u_{21}+u_{12}$　$u_{32}=2t_1-t_2-t_5$　$t_6=k_2t_7$　$t_8=(t_1-t_5)^2$　$t_9=t_2t_5$ $t_{10}=u_{11}+u_{20}$　$u_{31}=2t_6+t_8+t_9-t_4-t_{10}$　$t_{11}=-t_5(t_8-3t_4+t_9-2t_6-t_{10}-u_{11})$ $t_{12}=2t_1(t_4+t_{10}-t_6)$　$t_{13}=(t_4+t_{10}-f_5)t_2$　$t_{14}=k_1t_7$　$u_{30}=t_{11}-t_{12}+t_{13}+2t_{14}+t_3-u_{10}$	9*M*
7	计算 $v_3=(-v) \bmod u_3\equiv v_{32}x^2+v_{31}x+v_{30}$: $t_1=s_1u_{32}$　$t_2=k_3-t_1$　$t_3=u_{30}t_2$　$t_4=u_{31}t_2$　$t_5=u_{32}t_2$　$t_6=s_1u_{30}$　$t_7=s_1u_{31}$ $v_{30}=t_3-k_0$　$v_{31}=t_4+t_6-k_1$　$v_{32}=t_5+t_7-k_2$	6*M*
总计算量		1*I*+45*M*

算法 4　DBL(1→2)

输入：$y^2=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$; $D_1=(u_1, v_1)$, $D_2=D_1$; $u_1=x+u_{10}$, $v_1=v_{10}$;		
输出：$D_3=(u_3, v_3)$, $u_3=x^2+u_{31}x+u_{30}$, $v_3=v_{31}x+v_{30}$		
步骤	过程	计算量
1	计算 $u_3=(x+u_{10})^2=x^2+u_{31}x+u_{30}$: $u_{31}=2u_{10}$　$u_{30}=u_{10}^2$	1*M*
2	计算 $v_{31}=\dfrac{f'(-u_{10})}{2v_{10}}$: $t_1=7u_{30}+5f_5$　$t_2=-u_{10}t_1+4f_4$　$t_3=-u_{10}t_2+3f_3$　$t_4=-u_{10}t_3+2f_2$　$t_5=-u_{10}t_4+f_1$ $t_6=(2v_{10})^{-1}$　$v_{31}=t_5t_6$	1*I*+5*M*
3	计算 $v_{30}=v_{31}u_{10}+v_{10}$: $t_1=v_{31}u_{10}$　$v_{30}=t_1+v_{10}$	1*M*
总计算量		1*I*+7*M*

算法 5　DBL(2→3)

输入：$y^2=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$; $D_1=(u_1, v_1)$, $D_2=D_1$; $u_1=x^2+u_{11}x+u_{10}$, $v_1=v_{11}x+v_{10}$;		
输出：$D_3=(u_3, v_3)$, $u_3=x^3+u_{32}x^2+u_{31}x+u_{30}$, $v_3=v_{32}x^2+v_{31}x+v_{30}$.		
步骤	过程	计算量
1	计算 $r=\mathrm{res}(u_1, v_1)$: $t_1=u_{11}v_{11}$　$t_2=v_{10}-t_1$　$t_3=v_{10}t_2$　$t_4=v_{11}^2$　$t_5=u_{10}t_4$　$r=t_3+t_5$	$4M$
2	计算 $I=i_1x+i_0\equiv r v_1^{-1} \bmod u_1$: $i_1=-v_{11}$　$i_0=t_2$	$0M$
3	计算 $s'=s_1'x+s_0'=2rs\equiv I(f-v_1^2)/u_1 \bmod u_1$: $t_1=u_{11}^2$　$t_2=2f_5-3u_{10}+12t_1$　$t_3=u_{10}t_2$　$t_5=f_4u_{11}$　$t_6=3f_5+5t_1$ $t_7=i_1t_6$　$t_8=f_3-t_3-2t_5+t_7$　$t_9=6u_{10}-3f_5-10t_1$　$t_{10}=u_{11}t_9$　$t_{11}=f_4+t_{10}$ $t_{12}=u_{10}t_{11}$　$t_{13}=4f_5+6t_1$　$t_{14}=t_1t_{13}$　$t_{15}=2f_3-3t_5+t_{14}$　$t_{16}=u_{11}t_{15}$ $t_{17}=f_2-t_4-2t_{12}-t_{16}$　$t_{18}=t_8i_0$　$t_{19}=t_{17}i_1$　$t_{20}=u_{11}t_8$　$t_{21}=t_{17}+t_{20}$　$t_{22}=t_{21}i_0$ $t_{23}=t_8i_1$　$t_{24}=u_{10}t_{23}$　$s_1'=t_{18}+t_{19}$　$s_0'=t_{22}-t_{24}$	$14M$
4	计算 $s=s'/(2r)=s_1x+s_0$: $t_2=2rs_1'$　$t_3=(t_2)^{-1}$　$t_4=t_3s_1'$　$s_1=t_4s_1'$　$s_0=t_4s_0'$	$1I+4M$
5	计算 $v=su_1+v_1=s_1x^3+k_2x^2+k_1x+k_0$: $t_2=s_1u_{11}$　$k_2=t_2+s_0$　$t_3=s_0+s_1$　$t_4=u_{11}+u_{10}$　$t_5=s_0u_{10}$　$t_6=t_3t_4-t_2-t_5$ $k_1=t_6+v_{11}$　$k_0=t_5+v_{10}$	$3M$
6	计算 $u_3=(f-v^2)/u_1^2=x^3+u_{32}x^2+u_{31}x+u_{30}$: $t_3=s_1^2$　$u_{32}=-t_3-2u_{11}$　$t_4=t_2-k_2$　$t_5=2s_1t_4$　$t_6=s_1k_1$　$t_7=t_2+t_4$　$t_8=k_2t_7$ $t_9=3u_{10}-2t_1-f_5$　$t_{10}=2u_{11}t_9$　$t_{11}=2u_{10}-3t_1$　$u_{31}=f_5+t_5-t_{11}$　$t_{12}=t_3t_{11}$ $u_{30}=f_4-2t_6+t_8+t_{10}+t_{12}$	$6M$
7	计算 $v_3=(-v) \bmod u_3\equiv v_{32}x^2+v_{31}x+v_{30}$: $t_1=s_1u_{32}$　$t_2=s_1u_{31}$　$t_3=s_1u_{30}$　$v_{32}=t_1-k_2$　$v_{31}=t_2-k_1$　$v_{30}=t_3-k_0$	$3M$
总计算量		$1I+34M$

（上接第 33 页）

的多跳接入方式。其组网方式为：边缘接入采用 Ad hoc 技术，选择具有发展潜力的 AODV 按需路由协议；主干接入采用移动 IP 技术，选择信令开销较低且切换较快的分层移动 IPv6。

支持 IP 漫游的多跳接入方式扩展了已有的分层移动 IPv6 协议，对部分消息进行了补充修改，并增加了一些消息类型和移动选项，实现了主干网络与边缘 Ad hoc 网络的有机结合。移动用户不但能以较低的信令开销实现全局的 IP 漫游，还能保证进入同一 MANET 网络中的移动节点直接通信，降低了主干网络的传输负担，也保证了突发情况下节点仍能通过 MANET 保持链路畅通。边缘接入部分的多跳网络大大增加了移动 IP 网络的覆盖范围，能有效降低网络铺设成本。且由于多跳网络链路层可采用 802.11 系列标准，因此至少可拥有与无线局域网同样的带宽。

参 考 文 献

1　Eklund C, Marks R B, Stanwood K L, et al. IEEE Standard 802.16: A Technical Overview of the WirelessMAN Air Interface for Broadband Wireless Access. IEEE Communications Magazine, 2002, 40(6):98～107

2　IETF MIP6 Working Group. Mobile IPv6. http://www.ietf.org/html.charters/mip6-charter.htm

3　IETF MANET Working Group. Mobile Ad-hoc Networks. http://www.ietf.org/html.charters/manet-charter.htm

4　Perkins C E, Royer E M, Das S R. Ad Hoc On-Demand Distance Vector(AODV)Routing. IETF RFC 3561, Jul. 2003

5　Soliman H, Catelluccia C, Malki K E. et al. Hierarchical Mobile IPv6 Mobility Management(HMIPv6). IETF RFC 4140, Jun. 2004

6　Johnson D B, Perkins C E, Arkko J. Mobility Support in IPv6. IETF RFC 3775, Jun. 2004

7　Narten T, Nordmark E, Simpson W. Neighbor Discovery for IP Version 6(IPv6). IETF RFC 2461, Dec. 1998