Computer Science and Application 计算机科学与应用, 2012, 2, 26-31
http://dx.doi.org/10.12677/csa.2012.21006 Published Online March 2012 (http://www.hanspub.org/journal/csa)

Research on Fast Algorithms of Elliptic Curve over $GF\left(3^n\right)^*$

Jingjing Liu, Meng Zhou

School of Mathematics and System Science, LMIB, Beihang University, Beijing
Email: jingjingliu1988@163.com, zm1613@sina.com

Received: Nov. 24th, 2011; revised: Dec. 16th, 2011; accepted: Jan. 2nd, 2012

Abstract: $GF\left(3^n\right)$, as a more special type field of $GF\left(p^n\right)$, the elliptic curve cryptosystem based on which has their own advantages. As we know, reducing the operation of inverse is an important method in elliptic curve cryptography fast calculation. It requires $2k$ times inversions on elliptic curves over $GF\left(3^n\right)$ to compute scalar multiplication $3^k P$ by individual computation. This paper deduces a formula of calculating $3^k P$ directly based upon the idea of recursive induction and trading inversions for multiplication, which reduces the inversion to once. The proposed algorithm is prior to multiple tripling point algorithms when the speed ratio of field inversion to field multiplication is high. And the bigger the ratio is, the more the efficiency improves.

Keywords: Elliptic Curves; Scalar Multiplication; Field Inversion; Recursive Induction

$GF\left(3^n\right)$椭圆曲线快速算法研究*

刘景景，周　梦

北京航空航天大学数学与系统科学学院 LMIB，北京
Email: jingjingliu1988@163.com, zm1613@sina.com

收稿日期：2011 年 11 月 24 日；修回日期：2011 年 12 月 16 日；录用日期：2012 年 1 月 2 日

摘　要： $GF\left(3^n\right)$作为$GF\left(p^n\right)$更加特殊的一种类型，定义于其上的椭圆曲线密码算法有自己的优越性。快速计算椭圆曲线密码标量乘的方法之一是牺牲乘法或平方减少求逆次数，若采用逐次累加的方法计算$GF\left(3^n\right)$上椭圆曲线标量乘$3^k P$，需要$2k$次求逆运算。本文根据递推归纳、转换求逆为乘法的思想，推导了直接计算$3^k P$的公式，使求逆运算降至一次。在逆乘率 I/M 较高时，其效率要优于逐次三倍点算法，并且逆乘率越大，其效率提高的越多。

关键词： 椭圆曲线；标量乘；求逆；递推归纳

1. 引言

自 1985 年 Koblitz[1]和 Miller[2]各自分别提出椭圆曲线密码体制(ECC，Elliptic Curve Cryptography)，ECC 就一直得到众多密码学家及密码学研究者的关注。ECC 体制的安全性是基于椭圆曲线上离散对数问题求解的困难性[2]，目前还没有找到解决此问题的亚指数时间算法，因而与另一著名的公钥密码 RSA 相

*基金项目：国家自然科学基金资助项目(10871017)；北京市自然科学基金资助项目(102026)。

Copyright © 2012 Hanspub

比，ECC 密钥短，安全性高，速度快，存储空间占用少和带宽要求低。它的这些特点使得业内人士普遍认为 ECC 将成为下一代最通用的公钥加密算法标准。

目前，二元域$GF(2^n)$–ECC 和大素数域$GF(p)$–ECC 是最为常用的两个有限域，对于其上的椭圆曲线密码体制研究的也比较充分。相比而言，$GF(3^n)$–ECC 的研究工作还很少开展。随着 Weil 对和 Tate 对理论研究的不断进步以及基于此而设计的各种安全协议的广泛应用，使得小素数扩域椭圆曲线$GF(p^n)$–ECC 能够比传统有限域提供更多、更加灵活的密码选择方案。由于针对$GF(p^n)$–ECC 的攻击算法相对来说比较少，因而其安全性相比之下比较高。$GF(3^n)$作为$GF(p^n)$更加特殊的一种类型，定义于其上的椭圆曲线密码算法更加优越。$GF(3^n)$–ECC 有类似于$GF(2^n)$–ECC 计算速度快的某些特性，但也有自己的特殊性质，这使得其上算术运算效率非常高，极适合作为安全密码算法的载体。

椭圆曲线标量乘是 ECC 中最核心的运算，而求逆又是标量乘运算中最耗时的运算，所以很多学者采用牺牲乘法来减少逆运算的次数，我们延续这一思想，着重讨论在特征为 3 的椭圆曲线上直接计算$3^k P$的算法。文章首先介绍了 ECC 中相关的知识，之后提出了一种直接计算$3^k P$的递推算法。

2. 背景知识简介

2.1. 椭圆曲线的定义

定义 1 定义在域 K 上的 Weierstrass 方程：

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

所确定的平面曲线称为椭圆曲线：其中 K 为有限域，$a_1, a_2, a_3, a_4, a_6 \in K$,。满足(1)式的$(x, y)$称为域 K 上的点；此外，椭圆曲线还定义一个特殊的无穷远点 o；即域 K 上的点集和一个无穷远点 o 组成域 K 上的椭圆曲线$E(K)$。

椭圆曲线 E 的群运算是按照椭圆曲线群运算法则完成的，其中所涉及的运算有加、减、乘、求逆和平方五种基本运算，用 M, S, I 分别表示域 K 上的乘法、平方和求逆运算，I/M 表示求逆和乘法运算的复杂度比，也称逆乘率。五种基本运算中，加法和减法相对其他运算来说，所用时间可以忽略不计。求逆运算则是最耗时的，对于平方和乘法，一般设定$1[S] = 0.8[M]$。

2.2. 椭圆曲线上点的运算

$E(K)$表示定义在域 K 上的椭圆曲线 E 的所有点及一个无穷远点(称为零点)的集合。设$P = (x_1, y_1)$，$Q = (x_2, y_2)$是$E(K)$上的任意两个非零点，且$P \neq -Q$，仿射坐标下点加公式$P + Q = (x_3, y_3)$和倍点公式$2P(x_4, y_4)$分别由(2)、(3)给出：

1) 点加

$$\begin{cases} x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - a_1x_3 - y_1 - a_3 \end{cases}, \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad (2)$$

2) 倍点

$$\begin{cases} x_4 = \lambda^2 + a_1\lambda - a_2 - 2x_1 \\ y_4 = \lambda(x_1 - x_4) - a_1x_4 - y_1 - a_3 \end{cases},$$

$$\lambda = \frac{3x_1^2 + 2a_2x_1 - a_2y_1 + a_4}{2y_1 + a_1x_1 + a_3} \quad (3)$$

如果 K 的特征等于 3 且满足$a_1^2 = -a_2$时，容许的变量变换$(x, y) \to (x, y + a_1x + a_3)$可以将 E 变成：

$$y^2 = x^3 + ax + b \quad (4)$$

其中$a, b \in K$，上述曲线是非超奇异的，且$\Delta = -3a^3$，文献[3]指出若F_{3^L}上的椭圆曲线 E 由$y^2 = x^3 + ax + b\,(a, b \in F_{3^L})$,但$\chi(-b) \neq 1$)决定，则存在正整数 l，使得 E 为F_{3^L}上一类良好的椭圆曲线。下面给出当椭圆曲线 E 为形式(4)时，点加和倍点的公式：

1) 点加

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}, \text{ 其中 } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad (5)$$

2) 倍点

$$\begin{cases} x_4 = \lambda^2 + x_1 \\ y_4 = -\lambda^3 - y_1 \end{cases}, \text{ 其中 } \lambda = \frac{-a}{y_1} \quad (6)$$

其中点加的运算量为$1[I] + 1[S] + 2[M]$，倍点的运算量为$1[I] + 1[S] + 2[M]$。

3. 算法改进

由于求逆运算的运算量一般是乘法运算的 10 倍左右[4]，因此标量乘的计算可以将求逆运算转化为适

Copyright © 2012 Hanspub

量的乘法运算从而提高运算效率。基于这一思想，Ciet 等在文献[5]中给出了 $3P, 4P, 2P \pm Q, 3P \pm Q, 4P \pm Q$ 的快速算法。选取特征 3 上的椭圆曲线(4)计算 $3^k P$，若根据(5)、(6)式逐次计算，计算量为 $2k[I]+2k[S]+4k[M]$。下面推导直接计算 $3^k P$ 的一般算法公式。

3.1. $3P$ 的直接计算公式

$3^k P$ 直接计算公式的推导需从 $3P$ 开始。计算 $3P(x_3, y_3)$，一般先计算 $2P(x_2, y_2)$，然后将 $P(x_1, y_1)$ 加到 $2P$ 上，这样 $3P = 2P + P$，分别通过式(6)、式(5)计算 $2P$ 和 $3P$，即：

$$\lambda_1 = \frac{-a}{y_1},\ x_2 = \lambda_1^2 + x_1,\ y_2 = -\lambda_1^3 - y_1 \tag{7}$$

$$\lambda_2 = \frac{y_2 - y_1}{x_2 - x_1}, x_3 = \lambda_2^2 - x_1 - x_2, y_3 = \lambda_2(x_1 - x_3) - y_1 \tag{8}$$

如果按照上面的公式计算 $3P$，则需要两次求逆，总运算量为 $2[I]+2[S]+4[M]$。为了减少求逆运算，把(7)式中的 x_2, y_2 代入(8)式中的 λ_2 则 $\lambda_2 = -\lambda_1 + y_1^3/a^2$ 令 $I = 1/(a^2 y_1)$，从而 $3P$ 可以按照如下公式计算：$I = 1/(a^2 y_1), \lambda_1 = -Ia^3, \lambda_2 = -\lambda_1 + Iy_1^4$，$x_2 = \lambda_1^2 + x_1, x_3 = \lambda_2^2 - x_1 - x_2, y_3 = \lambda_2(x_1 - x_3) - y_1$。其总运算量为 $1[I]+5[S]+5[M]$，虽然比逐次计算增加了 1 次乘法运算和 3 次平方运算，但是减少了一次求逆运算。当 $I/M \geq 5.2$ 时，该算法优于直接计算。

下面推导 $3P$ 的一般形式，令 $A_1 = a$, $B_1 = -y$，则 $\lambda_1 = A_1/B_1$，将(7)式中的 y_2 代入(8)式中的 λ_2：

$$\lambda_2 = \frac{-\lambda_1^3 + y_1}{\lambda_1^2} = -\lambda_1 + \frac{y_1}{\lambda_1^2} = \frac{-(A_1^3 + B_1^4)}{A_1^2 B_1} \tag{9}$$

不需要计算 y_2，令 $d_1 = A_1^2 B_1, C_1 = -(A_1^3 + B_1^4)$ 和 $D_1 = B_1 d_1$，则 $\lambda_2 = C_1/d_1$，将(7)式中的 x_2 代入(8)式中的 x_3：

$$\begin{aligned} x_3 &= (\lambda_2 + \lambda_1)(\lambda_2 - \lambda_1) + x_1 = \left(\frac{C_1}{d_1} + \frac{A_1}{B_1}\right)\left(\frac{C_1}{d_1} - \frac{A_1}{B_1}\right) + x_1 \\ &= \frac{(B_1C_1 + A_1d_1)(B_1C_1 - A_1d_1) + x_1(B_1d_1)^2}{(B_1d_1)^2} \\ &= \frac{(B_1C_1 + A_1d_1)(B_1C_1 - A_1d_1) + x_1D_1^2}{(D_1)^2} \end{aligned} \tag{10}$$

令 $E_1 = (B_1C_1 + A_1d_1)(B_1C_1 - A_1d_1) + x_1D_1^2$，则 $x_3 = \frac{E_1}{D_1^2}$

$$\begin{aligned} y_3 &= \lambda_2(x_1 - x_3) - y_1 = \frac{C_1}{d_1}\left(x_1 - \frac{E_1}{D_1^2}\right) + B_1 \\ &= \frac{B_1C_1(x_1D_1^2 - E_1) + B_1D_1^3}{D_1^3} \end{aligned} \tag{11}$$

令 $F_1 = B_1C_1(x_1D_1^2 - E_1) + B_1D_1^3$，则 $y_3 = F_1/D_1^3$。

为了推导直接计算 $3^k P$ 的一般算法，这里我们引入了一些中间变量：

$$\begin{aligned} &A_1 = a; \\ &B_1 = -y_1; \\ &d_1 = A_1^2 B_1; \\ &C_1 = -(B_1^4 + A_1^3); \\ &D_1 = B_1 d_1; \\ &E_1 = (B_1C_1 + A_1d_1)(B_1C_1 - A_1d_1) + x_1D_1^2; \\ &F_1 = B_1C_1(x_1D_1^2 - E_1) + B_1D_1^3. \end{aligned}$$

则 $3P(x_3, y_3)$ 可以由以下式子计算得到：

$$x_3 = \frac{E_1}{D_1^2},\quad y_3 = \frac{F_1}{D_1^3} \tag{12}$$

计算复杂度为 $1[I]+4[S]+13[M]$ (一次求逆是计算 D_1^3 的逆)，此算法在保证求逆操作不增加的情况下，与以上不使用推导公式计算 $3P(1[I]+5[S]+5[M])$ 相比，增加了一些乘法操作，这样牺牲乘法的目的是为了接下来推导 $3^k P$ 的一般算法。

3.2. $9P$ 的直接计算公式

假设已知 $3P(x_3, y_3)$，下面来计算 $9P(x_9, y_9)$，这里我们利用 $3P(x_3, y_3)$ 的计算公式，直接计算[6] $9P(x_9, y_9) = 3 \times 3P(x_3, y_3)$。

依然令 $A_2 = a$，$B = -y_3 = -F_1/D_1^3$，则令 $B_2 = -F_1$，从而 $y_3 = -B_2/D_1^3$；$d = a^2(-y_3) = (A_2^2 B_2)/D_1^3$，令 $d_2 = A_2^2 B_2$；

$C = -(y_3^4 + a^3) = -\left(\frac{B_2^4}{D_1^{12}} + A_2^3\right) = \frac{-(B_2^4 + A_2^3 D_1^{12})}{D_1^{12}}$，令 $C_2 = -(B_2^4 + A_2^3 D_1^{12})$；

$D = (-y_3)a^2(-y_3) = a^2 y_3^2 = A_2^2 \times \frac{B_2^2}{D_1^6} = \frac{B_2 d_2}{D_1^6}$，令 $D_2 = B_2 d_2$；则

$$\lambda_1' = -a/y_3 = (A_2 D_1^3)/B_2 \tag{13}$$

$$\lambda_2' = -\lambda_1' + \frac{y_3}{(\lambda_1')^2} = \frac{-(B_2^4 + A_2^3 D_1^{12})}{A_2^2 B_2 D_1^9} = \frac{C_2}{d_2 D_1^9} \tag{14}$$

Copyright © 2012 Hanspub

从而：

$$\begin{aligned} x_9 &= (\lambda_2' + \lambda_1')(\lambda_2' - \lambda_1') + x_3 \\ &= \left(\frac{C_2}{d_2 D_1^9} + \frac{A_2 D_1^3}{B_2}\right)\left(\frac{C_2}{d_2 D_1^9} - \frac{A_2 D_1^3}{B_2}\right) + \frac{E_1}{D_1^2} \\ &= \frac{\left(B_2 C_2 + A_2 d_2 D_1^{12}\right)}{D_2 D_1^9} \cdot \frac{\left(B_2 C_2 - A_2 d_2 D_1^{12}\right)}{D_2 D_1^9} + \frac{E_1}{D_1^2} \\ &= \frac{\left(B_2 C_2 + A_2 d_2 D_1^{12}\right)\left(B_2 C_2 - A_2 d_2 D_1^{12}\right) + D_1^{16} D_2^2 E_1}{D_2 D_1^9} \end{aligned} \quad (15)$$

令 $E_2 = \left(B_2 C_2 + A_2 d_2 D_1^{12}\right)\left(B_2 C_2 - A_2 d_2 D_1^{12}\right) + D_1^{16} D_2^2 E_1$，则 $x_9 = \dfrac{E_2}{\left(D_1^9 D_2\right)^2}$。

$$\begin{aligned} y_3 &= \lambda_2'(x_3 - x_9) - y_3 \\ &= \frac{B_2 C_2}{D_1^9 D_2}\left(\frac{E_1}{D_1^2} - \frac{E_2}{\left(D_1^9 D_2\right)^2}\right) + \frac{B_2}{D_1^3} \\ &= \frac{B_2 C_2\left(D_1^{16} D_2^2 E_1 - E_2\right) + B_2 D_1^{24} D_2^3}{\left(D_1^9 D_2\right)^3} \end{aligned} \quad (16)$$

令 $F_3 = B_2 C_2\left(D_1^{16} D_2^2 E_1 - E_2\right) + B_2 D_1^{24} D_2^3$，则 $y_9 = \dfrac{F_2}{\left(D_1^9 D_2\right)^3}$；

综上，引入以下中间变量：

$$\begin{aligned} &A_2 = a; \\ &B_2 = -F_1; \\ &d_2 = A_2^2 B_2; \\ &C_2 = -\left(B_2^4 + A_2^3\right); \\ &D_2 = B_2 d_2; \\ &E_1 = \left(B_2 C_2 + A_2 d_2 D_1^{12}\right)\left(B_2 C_2 - A_2 d_2 D_1^{12}\right) + D_1^{16} D_2^2 E_1; \\ &F_2 = B_2 C_2\left(D_1^{16} D_2^2 E_1 - E_2\right) + B_2 D_1^{24} D_2^3. \end{aligned}$$

则 $9P(x_9, y_9)$ 可以由以下式子计算得到：

$$x_9 = \frac{E_2}{\left(D_1^9 D_2\right)^2}, \quad y_9 = \frac{F_2}{\left(D_1^9 D_2\right)^3} \quad (17)$$

其计算复杂度为 $1[I] + 12[S] + 30[M]$。与逐次计算的复杂度 $(4[I] + 4[S] + 8[M])$ 相比，该算法牺牲了 8 次平方操作和 22 次乘法操作，但是减少了 3 次求逆操作，当 $I/M \geq 9.47$ 时，将求逆运算转化为适量的乘法运算能够提高运算效率。

3.3. 27*P* 的直接计算公式

同样按照上面由 3*P* 计算 9*P* 的方法，由 9*P* 计算 $27P(x_{27}, y_{27})$，即由 E_2, F_2 和 D_2 计算 27*P*，同时保证中间过程和上面累死，便于找出计算 $3^k P$ 的一般算法，得到以下的过程：

$$\begin{aligned} &A_3 = a; \\ &B_3 = -F_2; \\ &d_3 = A_3^2 B_3; \\ &C_3 = -\left(B_3^4 + A_3^3\left(D_1^9 D_2\right)^{12}\right); \\ &D_3 = B_3 d_3; \\ &E_3 = \left(B_3 C_3 + A_3 d_3\left(D_1^9 D_2\right)^{12}\right)\left(B_3 C_3 - A_3 d_3\left(D_1^9 D_2\right)^{12}\right) \\ &\quad + \left(D_1^9 D_2\right)^{16} D_3^2 E_2; \\ &F_3 = B_3 C_3\left(\left(D_1^9 D_2\right)^{16} D_3^2 E_2 - E_3\right) + B_3\left(D_1^9 D_2\right)^{24} D_3^3. \end{aligned}$$

$27P(x_{27}, y_{27})$ 可以由以下式子得到：

$$x_{27} = \frac{E_3}{\left(\left(D_1^9 D_2\right)^9 D_3\right)^2}, \quad y_{27} = \frac{F_3}{\left(\left(D_1^9 D_2\right)^9 D_3\right)^3} \quad (18)$$

其计算复杂度为 $1[I] + 20[S] + 46[M]$。与逐次计算的复杂度 $(6[I] + 6[S] + 12[M])$ 相比，该算法牺牲了 14 次平方操作和 34 次乘法操作，但是减少了 5 次求逆操作，当 $I/M \geq 9.04$ 时，将求逆运算转化为适量的乘法运算能够提高运算效率。

3.4. $3^k P$ 的直接计算公式

根据前面对 3*P*，9*P* 和 27*P* 的计算，以此类推可以得到 $3^k P$ 的递推公式：

$$\begin{aligned} &A_k = a; \\ &B_k = -F_{k-1}; \\ &d_k = A_k^2 B_k; \\ &C_k = -\left(B_k^4 + A_k^3\left(M_k\right)^{12}\right); \\ &\text{其中} M_k = \left(\left(D_1^9 D_2\right)^9 D_3\right)^9 \cdots D_{k-1} \\ &D_k = B_k d_k; \\ &E_k = \left(B_k C_k + A_k d_k\left(M_k\right)^{12}\right)\left(B_k C_k - A_k d_k\left(M_k\right)^{12}\right) \\ &\quad + M_k^{16} D_k^2 E_{k-1}; \\ &F_k = B_k C_k\left(M_k^{16} D_k^2 E_{k-1} - E_k\right) + B_k M_k^{24} D_k^3. \end{aligned}$$

$$x_{3^k} = \frac{E_k}{\left(M_{k+1}\right)^2}, \quad y_{3^k} = \frac{F_k}{\left(M_{k+1}\right)^3}; \quad (19)$$

Copyright © 2012 Hanspub

下面给出在仿射坐标下直接计算$3^k P$的算法：

输入：基点$P(x_1, y_1)$; k; a

输出：椭圆曲线上的点$3^k P = (x_{3^k}, y_{3^k})$;

Step 1: 计算

$$A_1 = a; B_1 = -y_1; d_1 = A_1^2 B_1;$$
$$C_1 = -(B_1^4 + A_1^3); \quad D_1 = B_1 d_1;$$
$$E_1 = (B_1 C_1 + A_1 d_1)(B_1 C_1 - A_1 d_1) + x_1 D_1^2;$$
$$F_1 = B_1 C_1 (x_1 D_1^2 - E_1) + B_1 D_1^3.$$

Step 2: for i from 2 to k do

$$A_i = a; B_i = -F_{i-1}; d_i = A_i^2 B_i;$$
$$M_i = \left(\left(D_1^9 D_2 \right)^9 D_3 \right)^9 \cdots D_{i-1}$$
$$C_i = -(B_i^4 + A_i^3 (M_i)^{12}); \quad D_i = B_i d_i;$$
$$E_i = \left(B_i C_i + A_i d_i (M_i)^{12} \right)\left(B_i C_i - A_i d_i (M_i)^{12} \right) + M_i^{16} D_i^2 E_{i-1};$$
$$F_i = B_i C_i \left(M_i^{16} D_i^2 E_{i-1} - E_i \right) + B_i M_i^{24} D_i^3.$$

Step 3: compute: $x_{3^k} = \dfrac{E_k}{(M_{k+1})^2}; y_{3^k} = \dfrac{F_k}{(M_{k+1})^3};$

Step 4: 输出(x_{3^k}, y_{3^k});

在此算法中，由椭圆曲线上点P计算$3^k P$，其计算复杂性为$1[I]+(8k-4)[S]+(16k-2)[M]$，具体计算过程如下：第一步中须计算$A_1^2, B_1^4, D_1^2$共4次平方，$A_1^2 B_1$，$A_1^3$，$B_1 d_1$，$B_1 C_1$，$A_1 d_1$，$B_1 C_1 (x_1 D_1^2 - E_1), B_1 D_1^3$，$(B_1 C_1 + A_1 d_1)(B_1 C_1 - A_1 d_1)$，$x_1 D_1^2$，$D_1^3$，共10次乘法；第二步中当$i=2$时，需要7次平法，14次乘法，从$i=3$开始往后的$k-2$项，每步均需要8次平法，16次乘法，最后一步还需要计算$x_{3^k} = E_k / (M_{k+1})^2$，$y_{3^k} = F_k / (M_{k+1})^3$，它需要计算1次平方：$(M_{k+1})^2$；6次乘法：$M_K^8 \cdot M_k$，$M_K^9 D_k$，$(M_{k+1})^2 \cdot M_{k+1}$，$F_k / M_{k+1}^3$，$E_k / M_{k+1}^2$，$(E_k M_{k+1}) / M_{k+1}^2$，1次求逆：$1/M_{k+1}^3$，其中$M_{k+1} = M_k^9 D_k$，($M_k^8$在上一步中计算$M_k^{16}$时已经用到且存储$M_k^9 = M_k^8 \cdot M_k$)。综上，计算$3^k P$共需要的计算量为：

$$\begin{aligned} 1[I]+(8k-4)[S]+(16k-2)[M] &= (4[S]+10[M]) \\ &+ (7[S]+14[M]) + (k-2)(8[S]+16[M]) \\ &+ 1[I]+1[S]+6[M] \end{aligned}.$$

3.5. 算法性能比较

根据域中运算量的比较，假设1 S = 0.8 M，列表1。

通过表1比较可以发现，与逐次计算相比，虽然直接计算增加了一些乘法或平方运算，但同时也减少了一些求逆运算。而I/M的值一般大于10，将求逆运算转化为适量的乘法运算之后，由上表可以看出，随着k的增大，直接计算方法比逐次计算方法更有效。上面的临界点是$1[I] = (16.8k-5.2)[M]/(2k-1)$，当$k$趋于无穷大时，$I/M = 8.4$。故当$k$比较大且$I/M \geq 8.4$时，本文介绍的直接计算方法均优于逐次计算方法。

4. 结束语

本文根据递归思想及最小公倍数方法，研究了特征为3的有限域上的椭圆曲线在仿射坐标下直接计算$3^k P$的算法。本质上将耗时的域中求逆运算替换成了较快的平方或乘法运算，虽然牺牲了一些乘法或平方操作，但同时也减少了求逆操作，当逆乘率I/M比较大时，这种替换能加快计算速度，提高运算效率。具体地，当k很大时，与逐次计算相比，此算法的效率至少提高：1) 5.08%，当$I/M \geq 9$时；2) 12.50%，当$I/M \geq 10$时；3) 18.84%，当$I/M \geq 11$时。综上，I/M值越大，将求逆运算转化为适量的乘法运算越有效。因此，在椭圆曲线标量乘中，当I/M比较大时，该算法是一种有效的改进方法。

5. 致谢

在此，非常感谢我的导师周梦教授在论文构思和书写的过程中给予指导，感谢国家自然科学基金(10871017)、北京市自然科学基金(102026)的资助和

Table 1. The cost of different operations
表 1. 不同运算的比较

计算的值	计算方法	I	S	M	I/M
$3P$	逐次计算	2	2	4	10.6
	直接计算	1	4	13	
$9P$	逐次计算	4	4	8	9.47
	直接计算	1	12	30	
$27P$	逐次计算	6	6	12	9.04
	直接计算	1	20	46	
3^4P	逐次计算	8	8	16	8.86
	直接计算	1	28	62	
3^kP	逐次计算	$2k$	$2k$	$4k$	$(16.8k-5.2)/(2k-1)$
	直接计算	1	$8k-4$	$16k-2$	

Copyright © 2012 Hanspub

支持，同时对所有提供给我指导和帮助者、给予转载和引用权的资料、文献、研究思想和设想的所有者谨致谢意。

参考文献 (References)

[1] N. Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 1987, 48(177): 203-209.

[2] V. S. Miller. Uses of elliptic curves in cryptography. In: H. C. Williams, Ed., CRYPTO 1985. LNCS, Vol. 218. Heidelberg: Springer, 1986: 417-428.

[3] 李超. 特征为 3 的有限域上用于密码体制的椭圆曲线[J]. 通信保密, 1994, 58(2): 46-49.

[4] K. Fong, D. Hankerson, J. Lopez, et al. Field inversion and point halving revisited. IEEE Transactions on Computers, 2004, 53(8): 1047-1059.

[5] M. Ciet, K. Lauter, M. Joye and P. L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. Designs, Codes and Cryptography, 2006, 39(2): 189-206.

[6] 殷新春, 候红祥, 谢立. 基于双基数的快速标量乘算法[J]. 计算机科学, 2008, 35(6): 186-189, 195.

Copyright © 2012 Hanspub