Computer Science and Application 计算机科学与应用, 2014, 4, 76-84
Published Online April 2014 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2014.44013

Hans汉斯

Research on Fast Algorithms for Scalar Multiplication of Elliptic Curve over $GF\left(2^m\right)$

Li Sun, Meng Zhou

School of Mathematics and System Science, LMIB, Beihang University, Beijing
Email: sunlibuaa@126.com, zm1613@sina.com

Received: Mar. 6th, 2014; revised: Apr. 4th, 2014; accepted: Apr. 18th, 2014

Abstract

Elliptic curve cryptography finds numerous applications because of its excellent and unique properties. This paper focused on the scalar multiplication of Elliptic curve. We proposed the recursion formula to compute $3^k P$ over $GF\left(2^m\right)$ based on the idea of trading inversions for multiplications, which reduced the inversion to only once. At the same time, this paper also gave an accelerated algorithm for computing $3P+Q$, which saved two inversions compared to computing it directly. The result suggests that the proposed algorithms are more efficient than the normal algorithms when the ratios are more than 7.4 and 5.9 respectively.

Keywords

Elliptic Curve, Scalar Multiplication, Accelerated Algorithm, Inversion

$GF\left(2^m\right)$上椭圆曲线标量乘快速算法研究

孙　俐，周　梦

北京航空航天大学数学与系统科学学院LMIB，北京
Email: sunlibuaa@126.com, zm1613@sina.com

收稿日期：2014年3月6日；修回日期：2014年4月4日；录用日期：2014年4月18日

摘　要

椭圆曲线密码自提出以来便因其优良的性质而得到了广泛的应用。本文针对椭圆曲线上关键的标量乘运算，根据将耗时较多的求逆转换为乘法的思想，推导出了$GF(2^m)$上计算$3^k P$的递推公式，将求逆次数减少到一次。同时提出了计算$3P+Q$的加速算法，比直接计算节省了2次求逆。分析表明，在逆乘率分别大于7.4和5.9时，改进算法的效率优于逐次计算。

关键词

椭圆曲线，标量乘，加速算法，求逆

1. 引言

1985 年，椭圆曲线密码(ECC)由 Miller[1]和 Koblitz[2]等人分别独立提出。该密码体制一经提出便得到众多密码学研究专家的关注。ECC 基于有限域中的离散对数难解问题(DL)，这使得在同样的安全性能下，ECC 需要的密钥长度比 RSA 要短很多。

随着椭圆曲线密码的发展，快速实现成为学着们研究的重点。椭圆曲线的运算分为上层算法和底层算法，其中，上层加速算法是针对椭圆曲线上点的运算，即群的计算；底层算法是针对底层有限域上为实现点之间的运算而做的乘法、求逆、平方等运算。很明显，为了达到更好的计算效率，需要结合上层域以及底层域计算展开改进。这也是椭圆曲线加速算法的一个研究趋势。

在椭圆曲线密码体制中，核心运算就是椭圆曲线的标量乘kP的运算，它是椭圆曲线密码体制快速实现的关键。在标量乘运算中求逆的耗时是乘法的 8 倍[3]，用适量的乘法代替求逆可以明显的提高计算的效率。利用此思想，Guajardo 等人[4]提出了$GF(2^m)$上直接计算$4P$、$8P$、$16P$的算法。Sakai 等人[5]在 Guajardo 的基础上推导了$GF(p)$上$2^k P$的递推公式。Ciet 等人[6]提出了仿射坐标下直接计算$3P+Q$的算法，将求逆次数减少到 2 次。刘连浩等人[7]在 Ciet 等人的基础上提出了另一种在仿射坐标系下计算$3P+Q$的方法，将求逆次数减少到了 1 次。

本文首先介绍了椭圆曲线的背景知识，包括定义分类以及椭圆曲线上点的运算法则；其次采取了用乘法替代求逆的方法推导了仿射坐标系下$GF(2^m)$上$3^k P$的递推公式；最后结合 Ciet 以及刘连浩等人的方法，提出了$GF(2^m)$上另一种计算$3P+Q$的方法。

2. 背景知识介绍

2.1. 椭圆曲线的定义及分类[1] [2]

由代数几何学可知，椭圆曲线(Elliptic Curve)在解析平面中表现为一条非奇异的三次平面曲线。一般来讲，椭圆曲线有如下形式的 Weierstrass 方程：

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

(1) 式中的域K为有限域，$a_1, a_2, a_3, a_4, a_6 \in K$。该方程的所有解$(x, y) \in K \times K$连同无穷远点组成的集合$E$称为椭圆曲线，记为$E(K) \cup \{\infty\}$。

对于任意给定的一条椭圆曲线$E(K)$，总可以选取一组适当的变量代换式，使曲线在仿射坐标下的 Weierstrass 方程式具有更加简单的形式。本文讨论的是$Char(K) = 2$时的椭圆曲线上的算法，此时(1)式

可被简化成

$$y^2+xy=x^3+ax^2+b \tag{2}$$

其中$a,b\in GF(2^m)$且$b\neq 0$。

在椭圆曲线底层域运算中涉及有加法、减法、乘法、平方、求逆五中运算，通常用I、M和S分别表示求逆、乘法和平方，用I/M表示逆乘率。其中，加法和减法相对其他三种运算来说，耗时可以忽略不计，所以通常不考虑加减法。求逆是最耗时的，一般$I/M\geq 8$，通常计$1S=0.8M$ [8]。

2.2. 椭圆曲线上点的运算法则

设点$P=(x_1,y_1)$和点$Q=(x_2,y_2)$是$E(K)$上的任意两个非零点，且$P\neq\pm Q$，则点加公式$P+Q=(x_3,y_3)$以及倍点公式$2P=(x_4,y_4)$分别由(3)、(4)给出：

1) 点加

$$\begin{cases}\lambda_1=(y_1+y_2)/x_1+x_2\\ x_3=\lambda_2^2+\lambda_2+x_1+x_2+a\\ y_3=\lambda_2(x_1+x_3)+x_3+y_1\end{cases} \tag{3}$$

2) 倍点

$$\begin{cases}\lambda_2=x_1+y_1/x_1\\ x_4=\lambda_1{}^2+\lambda_1+a\\ y_4=x_1^2+\lambda_1x_4+x_4\end{cases} \tag{4}$$

其中点加的运算量为$I+2M+S$，倍点的运算量为$I+2M+2S$。

3. 推导仿射坐标系下$3P$、3^kP的算法

3.1. $3P$的推导过程

先算$2P=(x_2,y_2)$

$$\lambda_1=x_1+y_1/x_1,\ x_2=\lambda_1{}^2+\lambda_1+a,\ y_2=x_1^2+\lambda_1x_2+x_2$$

再算$2P+P=(x_3,y_3)$

$$\lambda_2=(y_1+y_2)/(x_1+x_2),\ x_3=\lambda_2^2+\lambda_2+x_1+x_2+a,\ y_3=\lambda_2(x_1+x_3)+x_3+y_1$$

$$A_1=x_1$$

$$B_1=x_1^2+y_1$$

$$C_1'=x_1+x_2=x_1+\lambda_1^2+\lambda_1+a=\left(B_1(A_1+B_1)+A_1^2(a+x_1)\right)/A_1^2$$

令$C_1=B_1(A_1+B_1)+A_1^2(a+x_1)$

$$D_1'=y_1+y_2=y_1+x_1^2+\lambda_1x_2+x_2=y_1+x_1^2+(\lambda_1+1)(\lambda_1^2+\lambda_1+\mathrm{a})$$

$$=\left(A_1^3B_1+(A_1+B_1)\left(B_1(A_1+B_1)+A_1^2a\right)\right)/A_1^3$$

令$D_1=A_1^3B_1+(A_1+B_1)\left(B_1(A_1+B_1)+A_1^2a\right)$，$E_1=A_1C_1$则

$$\lambda_2=D_1/(A_1C_1)=D_1/E_1$$

令$F_1=A_1E_1$

$$x_3=\lambda_2^2+\lambda_2+x_1+x_2+a=\frac{(A_1D_1+B_1E_1)(A_1D_1+B_1E_1+1)+x_1F_1^2}{F_1^2}$$

令 $G_1=(A_1D_1+B_1E_1)(A_1D_1+B_1E_1+1)+x_1F_1^2$

$$x_3=G_1/F_1^2$$

$$y_3=\lambda_2(x_1+x_3)+x_3+y_1=\frac{A_1D_1(x_1F_1^2+G_1)+G_1F_1+y_1F_1^3}{F_1^3}$$

令 $H_1=A_1D_1(x_1F_1^2+G_1)+G_1F_1+y_1F_1^3$

$$y_3=H_1/F_1^3$$

总共需要计算的量如下：

$$A_1=x_1$$

$$B_1=x_1^2+y_1$$

$$C_1=B_1(A_1+B_1)+A_1^2(a+x_1)$$

$$D_1=A_1^3B_1+(A_1+B_1)(B_1(A_1+B_1)+A_1^2a)$$

$$E_1=A_1C_1$$

$$F_1=A_1E_1$$

$$G_1=(A_1D_1+B_1E_1)(A_1D_1+B_1E_1+1)+x_1F_1^2$$

$$H_1=A_1D_1(x_1F_1^2+G_1)+G_1F_1+y_1F_1^3$$

$$x_3=G_1/F_1^2$$

$$y_3=H_1/F_1^3$$

此过程将计算 2 次求逆减少为只计算 1 次求逆，即计算 F_1^3 的逆。其计算复杂度为 $I+18M+3S$，相比于直接计算的 $2I+4M+3S$，增加了 $14M$，这样是为了之后 3^kP 的推导。

3.2. $9P$ 的推导过程

上一部分我们已经计算出了 $3P$，下面我们计算 $9P=3\times3P=(x_9,y_9)$

$$A_2'=x_3=G_1/F_1^2$$

令 $A_2=G_1$

$$B_2'=x_3^2+y_3=(G_1^2+H_1F_1)/F_1^4$$

令 $B_2=G_1^2+H_1F_1$

$$C_2'=x_3+x_6=\left(B_2(A_2F_1^2+B_2)+(A_2F_1^2)^2a+A_2^3F_1^2\right)\Big/(A_2F_1^2)^2$$

令 $C_2=B_2(A_2F_1^2+B_2)+(A_2F_1^2)^2a+A_2^3F_1^2$

$$D_2'=\left(A_2^3F_1^2B_2+(A_2F_1^2+B_2)\left(B_2(A_2F_1^2+B_2)+(A_2F_1^2)^2a\right)\right)\Big/(A_2F_1^2)^3$$

令 $D_2=A_2^3F_1^2B_2+(A_2F_1^2+B_2)\left(B_2(A_2F_1^2+B_2)+(A_2F_1^2)^2a\right)$

$$E_2 = A_2C_2$$

$$F_2 = A_2E_2$$

$$G_2 = (A_2D_2 + B_2E_2)(A_2D_2 + B_2E_2 + F_2F_1^2) + A_2F_2^2F_1^2$$

$$H_2 = A_2D_2\left(A_2F_1^2F_2^2 + G_2\right) + G_2F_2F_1^2 + \left(A_2^2 + B_2\right)F_1^2F_2^3$$

$$x_9 = G_2 \Big/ \left(F_2F_1^2\right)^2$$

$$y_9 = H_2 \Big/ \left(F_2F_1^2\right)^3$$

在上述计算过程中，我们抵消了x_3和y_3，这样是为了在之后的递推计算中减少计算量。其计算复杂度为$I+36M+8S$，比直接计算所需的$4I+8M+6S$，增加了$28M+2S$，但减少了 3 次求逆，在逆乘率$I/M \geq 9.9$时，此算法比直接计算有效。

3.3. 27*P* 的推导过程

下面用同样的方法计算$27P = (x_{27}, y_{27})$，需要计算的量如下：

$$A_3 = G_2$$

$$B_3 = G_2^2 + H_2F_2F_1^2$$

$$C_3 = B_3\left(A_3\left(F_2F_1^2\right)^2 + B_3\right) + \left(A_3\left(F_2F_1^2\right)^2\right)^2 a + A_3^3\left(F_2F_1^2\right)^2$$

$$D_3 = A_3^3\left(F_2F_1^2\right)^2 B_3 + \left(A_3\left(F_2F_1^2\right)^2 + B_3\right)\left(B_3\left(A_3\left(F_2F_1^2\right)^2 + B_3\right) + \left(A_3\left(F_2F_1^2\right)^2\right)^2 a\right)$$

$$E_3 = A_3C_3$$

$$F_3 = A_3E_3$$

$$G_3 = (A_3D_3 + B_3E_3)\left(A_3D_3 + B_3E_3 + F_3\left(F_2F_1^2\right)^2\right) + A_3F_3^2\left(F_2F_1^2\right)^2$$

$$H_3 = A_3D_3\left(A_3F_3^2\left(F_2F_1^2\right)^2 + G_3\right) + G_3F_3\left(F_2F_1^2\right)^2 + \left(A_3^2 + B_3\right)F_3^3\left(F_2F_1^2\right)^2$$

$$x_{27} = \frac{G_3}{\left(F_3\left(F_2F_1^2\right)^2\right)^2}$$

$$y_{27} = \frac{H_3}{\left(F_3\left(F_2F_1^2\right)^2\right)^3}$$

其计算复杂度为$I+55M+12S$，比直接计算所需的$6I+12M+9S$，增加了$43M+3S$，但减少了 5 次求逆，在逆乘率$I/M \geq 9.1$时，此算法比直接计算有效。

3.4. 3^k *P* 的推导过程及算法性能比较

由之前的推导，我们可以得出计算$3^k P = \left(x_{3^k}, y_{3^k}\right)$的递推公式如下：

$$A_k = G_{k-1};\ G_0 = x_1$$

$$B_k = G_{k-1}^2 + H_{k-1}M_{k-1};\ H_0 = y_1$$

$$C_k = B_k\left(A_k M_{k-1}^2 + B_k\right) + \left(A_k M_{k-1}^2\right)^2 a + A_k^3 M_{k-1}^2$$

$$D_k = A_k^3 M_{k-1}^2 B_k + \left(A_k M_{k-1}^2 + B_k\right)\left(B_k\left(A_k M_{k-1}^2 + B_k\right) + \left(A_k M_{k-1}^2\right)^2 a\right)$$

$$E_k = A_k C_k$$

$$F_k = A_k E_k$$

$$M_k = F_k \cdots \left(F_3\left(F_2 F_1^2\right)^2\right)^2 = F_k M_{k-1}^2;\ M_0 = 1$$

$$G_k = \left(A_k D_k + B_k E_k\right)\left(A_k D_k + B_k E_k + M_k\right) + A_k F_k M_k$$

$$H_k = A_k D_k\left(A_k F_k M_k + G_k\right) + G_k M_k + \left(A_k^2 + B_k\right) F_k^2 M_k$$

$$x_{3^k} = G_k / M_k^2$$

$$y_{3^k} = H_k / M_k^3$$

在此递归过程中，先计算当$k=1$时的$A_k \sim H_k$，计算复杂度为$15M+3S$。此后$k-1$步每次需增加$18M+4S$，最后还需计算x_{3^k}, y_{3^k}，需要$1I+4M+1S$，故总的计算复杂度为：

$$(15M+3S)+(k-1)(18M+4S)+(1I+4M+1S) = I+(18k+1)M+(4k)S$$

与此同时，直接计算需要$(2k)I+(4k)M+(3k)S$。具体递归过程计算量可见表 1。我们可以看到当k趋于无穷大且$I/M \geq 7.4$时，此算法比直接计算有效。

4. 改进的$3P+Q$算法

文献[7]中利用求最小公倍数的方法将计算$3P+Q$时的 3 次求逆减少到 1 次。本文通过同样的方法推导$GF(2^m)$上$3P+Q$的改进算法。

先给出一般的计算方法。设点$P=(x_1,y_1)$和点$Q=(x_2,y_2)$是$E(K)$上的任意两个非零点，计算$3P+Q$。

先算$2P=(x_3,y_3)$，运算量为$I+2M+2S$

$$\begin{cases} \lambda_1 = x_1 + \dfrac{y_1}{x_1} \\ x_3 = \lambda_1^2 + \lambda_1 + a \\ y_3 = x_1^2 + \lambda_1 x_3 + x_3 \end{cases}$$

再算$P+Q=(x_4,y_4)$，运算量为$I+2M+S$

Table 1. The comparison of the improved algorithm and the direct calculation
表 1. 改进算法与直接计算的效率比较

	一般方法运算量	改进方法运算量	相交处I/M
$3P$	$2I+4M+3S$	$I+18M+3S$	14
$9P$	$4I+8M+6S$	$I+36M+8S$	9.9
$27P$	$6I+12M+9S$	$I+55M+12S$	9.1
$3^k P$	$(2k)I+(4k)M+(3k)S$	$I+(18k+1)M+(4k)S$	7.4

$$\begin{cases}\lambda_2=\dfrac{y_1+y_2}{x_1+x_2}\\ x_4=\lambda_2^2+\lambda_2+x_1+x_2+a\\ y_4=\lambda_2(x_1+x_4)+x_4+y_1\end{cases}$$

再算 $3P+Q=2P+(P+Q)=(x_5,y_5)$，运算量为 $I+2M+S$

$$\begin{cases}\lambda_3=\dfrac{y_3+y_4}{x_3+x_4}\\ x_5=\lambda_3^2+\lambda_3+x_3+x_4+a\\ y_5=\lambda_3(x_3+x_5)+x_5+y_3\end{cases}$$

改进方法如下：

设 $\lambda_1=B_1/A_1$， $\lambda_2=B_2/A_2$， $\lambda_3=B_3/A_3$

则 $A_1=x_1$， $B_1=x_1^2+y_1$， $A_2=x_1+x_2$， $B_2=y_1+y_2$ 以及 $\lambda_1=B_1/A_1$， $\lambda_2=B_2/A_2$ 已知，下面计算 A_3，B_3 和 λ_3。

$$\begin{aligned}A_3=x_3+x_4&=\lambda_1^2+\lambda_1+a+\lambda_2^2+\lambda_2+x_1+x_2+a=(\lambda_1+\lambda_2)^2+\lambda_1+\lambda_2+A_2\\&=\left((A_2B_1+A_1B_2+A_1A_2)(A_2B_1+A_1B_2)+(A_1A_2)^2A_2\right)\Big/(A_1A_2)^2\end{aligned}$$

设 $D=(A_2B_1+A_1B_2+A_1A_2)(A_2B_1+A_1B_2)+(A_1A_2)^2A_2$

将上式代入 $\lambda_3=B_3/A_3$ 得 $\lambda_3=B_3(A_1A_2)^2D$

为了减少求逆次数，我们取 λ_1、 λ_2、 λ_3 的最小公倍数，设 $\lambda_c=(DA_1A_2)^{-1}$，则

$$\lambda_1=D\lambda_cA_2B_1，\quad \lambda_2=D\lambda_cA_1B_2，\quad \lambda_3=B_3(A_1A_2)^3\lambda_c$$

由于 B_3 还是未知，接下来计算 B_3。

$$\begin{aligned}B_3=y_3+y_4&=x_1^2+\lambda_1x_3+x_3+\lambda_2(x_1+x_4)+x_4+y_1\\&=(\lambda_1+\lambda_2)(x_1+x_3)+(\lambda_1+1)(x_3+x_4)+\lambda_1x_1+x_1^2+y_1\\&=(\lambda_1+\lambda_2)(x_1+x_3)+(\lambda_1+1)A_3\end{aligned}$$

将 B_3 代入 $\lambda_3=B_3(A_1A_2)^3\lambda_c$ 得：

$$\lambda_3=B_3(A_1A_2)^3\lambda_c=(A_1A_2)^3\lambda_c(\lambda_1+\lambda_2)(x_1+x_3)+(A_1A_2)D(\lambda_1+1)$$

最后根据 $x_5=\lambda_3^2+\lambda_3+x_3+x_4+a$ 和 $y_5=\lambda_3(x_3+x_5)+x_5+y_3$ 计算出 (x_5,y_5)。

由于 x_5 的计算中还含有 x_4，将 $x_4=\lambda_2^2+\lambda_2+x_1+x_2+a$ 代入得

$$\begin{aligned}x_5&=\lambda_3^2+\lambda_3+x_3+\lambda_2^2+\lambda_2+x_1+x_2+a+a\\&=\lambda_3^2+\lambda_2^2+\lambda_3+\lambda_2+x_1+x_2+x_3\\&=(\lambda_3+\lambda_2)^2+\lambda_3+\lambda_2+x_1+x_2+x_3\end{aligned}$$

故在整个计算过程中所有需要计算的量有：

$$A_1=x_1$$

$$B_1=x_1^2+y_1$$

$$A_2 = x_1 + x_2$$

$$B_2 = y_1 + y_2$$

$$D = (A_2B_1 + A_1B_2 + A_1A_2)(A_2B_1 + A_1B_2) + (A_1A_2)^2 A_2$$

$$\lambda_c = (DA_1A_2)^{-1}$$

$$\lambda_1 = D\lambda_c A_2B_1$$

$$\lambda_2 = D\lambda_c A_1B_2$$

$$\lambda_3 = (A_1A_2)^3 \lambda_c (\lambda_1 + \lambda_2)(x_1 + x_3) + (A_1A_2)D(\lambda_1 + 1)$$

$$x_3 = \lambda_1^2 + \lambda_1 + a$$

$$y_3 = x_1^2 + \lambda_1 x_3 + x_3$$

$$x_5 = (\lambda_3 + \lambda_2)^2 + \lambda_3 + \lambda_2 + x_1 + x_2 + x_3$$

$$y_5 = \lambda_3(x_3 + x_5) + x_5 + y_3$$

直接按公式计算 $3P+Q = 2P+(P+Q)$，需要的计算量为 $3I+6M+4S$，经过算法改进后，将 3 次求逆转换为 1 次求逆，所需要的计算量为 $I+17M+5S$，虽然增加了 11 次乘法和 1 次平方，但减少了 2 次求逆，在 $I/M \geq 5.9$ 时，此算法比直接计算有效。

5. 结束语

本文利用了转换求逆为乘法和求取最小公倍数的方法，研究了特征为 2 的域上计算 $3^k P$ 的递推公式，当 k 趋于无穷大且 $I/M \geq 7.4$ 时改进的算法比直接计算效率更高。利用同样的原理，结合 Ciet 以及刘连浩等人的方法，推导了计算 $3P+Q$ 的方法，在 $I/M \geq 5.9$ 时，改进的算法比直接计算有效。由于在椭圆曲线标量乘中，I/M 通常大于 8，所以本文的改进算法是有效的。

另外，多基链 MBNS 作为 k 的有效表示是高度冗余的，可以减少标量乘中的上层运算。将 MBNS 和本文提出的改进算法相结合将是我们进一步研究的方向。

致　谢

在本论文的写作过程中，我的导师周梦老师倾注了大量的心血，从构思选题到开题报告，再到一遍又一遍地修改每稿中的具体问题，他都给了我极大的支持与帮助，在此我表示衷心感谢。同时我还要感谢国家自然科学基金 NSFC 11271040 资助项目的支持，以及在我学习期间给我极大关心和支持的各位老师和朋友。

项目基金

国家自然科学基金 NSFC 11271040 资助项目。

参考文献 (References)

[1] Miller, V. (1986) Use of elliptic curves in cryptography. *Lecture Notes in Computer Science*, **218**, 417-426.

[2] Koblitz, N. (1987) Elliptic curve crystosyestems. *Mathematics of Computation*, **48**, 203-209

[3] Fong, K., et al. (2004) Field inversion and point halving revisited. *IEEE Transactions on Computers*, **53**, 1047-1059.

[4] Guajardo, J. and Christof, P. (1997) Efficient algorithms for elliptic curve cryptosystems. Springer Berlin Heidelberg, Berlin.

[5] Sakai, Y. and Kouichi, S. (2001) Efficient scalar multiplications on elliptic curves with direct computations of several doublings. IEICE Transactions on Fundamentals of Electronics. *Communications and Computer Sciences*, **84**, 120-129.

[6] Ciet, M., et al. (2006) Trading inversions for multiplications in elliptic curve cryptography. *Designs, Codes and Cryptography*, **39**, 189-206.

[7] 刘连浩, 申勇 (2009) 椭圆曲线密码体制中标量乘法的快速算法. *计算机应用研究*, **3**, 1104-1108.

[8] Hankerson, D., Julio, L.H. and Alfred, M. (2000) Software implementation of elliptic curve cryptography over binary fields. *Lecture Notes in Computer Science*, **1965**, 1-24.