Advances in Applied Mathematics 应用数学进展, 2015, 4(4), 390-399
Published Online November 2015 in Hans. http://www.hanspub.org/journal/aam
http://dx.doi.org/10.12677/aam.2015.44049

Hans汉斯

Research on Fast Algorithms for Scalar Multiplication of Elliptic Curve Cryptography over $GF(3^n)$

Shaofang Shen, Meng Zhou*

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

Received: Nov. 9th, 2015; accepted: Nov. 25th, 2015; published: Nov. 30th, 2015

Abstract

In this paper we investigate the fast algorithm of scalar multiplication based on Elliptic Curve Cryptography (ECC) over fields of characteristic three, and make improvements both in underlying operations and upper operations respectively. In the underlying operations, we deduce a formula of calculating 3^kP directly under the affine coordinates based upon the idea of recursion, trading inversion for multiplication and trading multiplication for cube, which reduces the inversion to once; in the upper operations, we adopt the sliding window scalar multiplication method, which reduces the length of the non-zero windows and the total computations of $3P$ effectively.

Keywords

Elliptic Curve Cryptography (ECC), Scalar Multiplication, Recursion, Sliding Window

$GF(3^n)$椭圆曲线密码体制中标量乘快速算法研究

申少芳，周　梦*

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

*通讯作者。

文章引用：申少芳，周梦. $GF(3^n)$椭圆曲线密码体制中标量乘快速算法研究[J]. 应用数学进展, 2015, 4(4): 390-399.
http://dx.doi.org/10.12677/aam.2015.44049

收稿日期：2015年11月9日；录用日期：2015年11月25日；发布日期：2015年11月30日

摘　要

本文研究了特征为3的椭圆曲线密码体制中标量乘的快速算法，并针对底层运算和上层运算分别进行改进。在底层运算中，采用递推归纳、乘法代替求逆、立方代替乘法的思想提出在仿射坐标下直接计算3^kP的算法，将求逆运算降至1次；在上层运算中，采用滑动窗口标量乘的方法，既有效减少了非零窗口的长度又减少了算法中3倍点总的运算量。

关键词

椭圆曲线密码体制，标量乘，递推归纳，滑动窗口

1. 引言

自 1985 年 Neal Koblitz 和 Victor Miller 分别独立提出椭圆曲线密码体制(ECC, Elliptic Curve Cryptography)，其安全性和有效实现性就得到了广泛的关注。ECC 属于公钥密码体制，它可以提供同 RSA 密码体制相同的功能。然而它的安全性建立在椭圆曲线离散对数问题(ECDLP)的困难性之上[1]，目前求解 ECDLP 最好的算法具有全指数时间复杂度，与 RSA 相比，ECC 具有如下优良性质：

1) ECC 安全性高。在同样的攻破时间内，ECC 需要的密钥比 RSA 要短得多，如 160 bite 的 ECC 和 1024 bite 的 RSA 有相同的安全性；

2) ECC 实现性强。密钥短、存储空间占用少、宽带要求低、运算处理时间短等特点均有利于 ECC 实现；

3) ECC 具有多重选择性。在相同基域下，ECC 可以通过改变椭圆曲线方程的系数得到不同的加密算法，拥有更多的选择性，而对于给定的素数 p 或 q 只能对应唯一的一个 RSA 算法。

因此，ECC 在信息安全中有广泛的应用。目前，$GF\left(2^n\right)-ECC$ 和 $GF\left(p\right)-ECC$ 已经研究得比较充分，而对 $GF\left(3^n\right)-ECC$ 的研究工作还很少开展。随着 Weil 对和 Tate 对理论的不断进展及基于此而设计的各种安全协议的广泛应用，人们对小素数扩域 $GF\left(p^n\right)-ECC$ 的研究产生了浓厚的兴趣。与传统有限域相比，小素数扩域 $GF\left(p^n\right)-ECC$ 能提供更多、更灵活的密码方案，且针对其的攻击算法相对较少，因此其安全性也比较高。$GF\left(3^n\right)-ECC$ 作为 $GF\left(p^n\right)-ECC$ 的一种特殊类型，其具有类似于 $GF\left(2^n\right)-ECC$ 的计算速度快的某些特性，但也有自己的特殊性质，这使得其算术运算效率非常高，极适合作为安全密码算法的载体[2]-[5]。

在 ECC 中，核心运算就是椭圆曲线的标量乘 kP 的运算，它是 ECC 快速实现的关键，而求逆运算又是标量乘中最耗时的，因此采用数学技巧减少求逆次数，对提高运算效率具有重要意义[6]。

本文首先介绍了 ECC 的基础知识，包括椭圆曲线的定义及分类以及其上点的运算法则；其次，采用递推归纳、乘法代替求逆、立方代替乘法的思想推导出了仿射坐标系下 $GF\left(3^n\right)$ 上 3^kP 的递推公式；最后基于滑动窗口标量乘对核心标量乘算法进行改进并分析其性能。

2. ECC 基础知识

2.1. 椭圆曲线的定义及分类

定义 1. 设 K 为有限域，K 上的 Weierstrass 方程为

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

其中， $a_i \in K\ (i = 1,2,3,4,6)$，称 E 是域 K 上的椭圆曲线。

有限域上椭圆曲线的分类[7]：

(1) $char(K) = 2$

$$\begin{cases} y^2 + xy = x^3 + ax^2 + b, & 若a_1 \neq 0且\ j(E) \neq 0 \\ y^2 + cy = x^3 + ax + b, & 若a_1 = 0且\ j(E) = 0 \end{cases} \tag{2.2}$$

(2) $char(K) = 3$

$$\begin{cases} y^2 = x^3 + ax^2 + b, & 若a_1^2 \neq -a_2且\ j(E) \neq 0 \\ y^2 = x^3 + ax + b, & 若a_1^2 = -a_2且\ j(E) = 0 \end{cases} \tag{2.3}$$

(3) $char(K) = p > 3$

$$y^2 = x^3 + ax + b \tag{2.4}$$

其中 j 是 E 的不变量。

2.2. 椭圆曲线上点的运算法则

定义在域 K 上的椭圆曲线 E 的所有点的集合称为 $E(K)$， $E(K)$ 并上一个无穷远点 O 构成一个交换群，群上的运算法则由“弦切律”定义。设 $P = (x_1, y_1)$，$Q = (x_2, y_2)$ 是 $E(K)$ 上的任意两个非零点，且 $P \neq -Q$，下面给出仿射坐标下点加公式 $P + Q = (x_3, y_3)$ 和倍点公式 $2P = (x_4, y_4)$：

(1) 点加

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

(2) 倍点

$$\begin{cases} x_4 = \lambda^2 + a_1\lambda - a_2 - 2x_1 \\ y_4 = \lambda(x_1 - x_4) - a_1x_4 - y_1 - a_3 \end{cases}, \quad \lambda = \frac{3x_1^2 + 2a_2x_1 - a_2y_1 + a_4}{2y_1 + a_1x_1 + a_3}$$

出于安全性考虑，人们通常选择 $j \neq 0$ 的椭圆曲线来设计椭圆曲线密码系统，故本文我们将对 $char(K) = 3$ 时， $E: y^2 = x^3 + ax^2 + b$ 的椭圆曲线进行讨论，其点加和倍点可分别简化为：

(1) 点加

$$\begin{cases} x_3 = \lambda^2 - a - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}, \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \tag{2.5}$$

(2) 倍点

$$\begin{cases} x_4 = u^2 - a + x_1 \\ y_4 = u(x_1 - x_4) - y_1 \end{cases}, \quad u = \frac{ax_1}{y_1} \tag{2.6}$$

令 M 表示域乘法， S 表示域平方， C 表示域立方， I 表示域逆， A 表示点加， $t(X)$ 表示计算一次域上 X 运算所需的时间， $\alpha = t(I)/t(M)$ 表示逆乘率。与其他运算时间相比， $t(A)$ 可以忽略不计，可知 $P + Q$ 所需的计算复杂度为 $1[I] + 2[M] + 1[S]$， $2P$ 所需的计算复杂度为 $1[I] + 2[M] + 1[S]$。

3. $GF(3^n)$上底层域快速算法综述

标量乘 kP 的运算分为两个层次来完成，一个是底层运算，主要是执行有限域 $GF(3^n)$ 中各种点加、倍点运算；另一个是上层运算，即椭圆曲线上的点加和倍点运算构成的有限阿贝尔群上的运算。故研究 $GF(3^n)$ 上底层域快速算法应是寻求实现点加、倍点的快速算法。

$GF(3^n)$ 上有限域运算[8]是其底层运算的基础，设 $f(x)$ 是定义在 $GF(3)$ 中的 n 次不可约多项式，则 $GF(3^n)$ 可表示为

$$GF(3^n)=GF(3)[x]/f(x)$$

而对 $\forall a\in GF(3^n)$ 可以唯一地表示为

$$a=a_{n-1}x^{n-1}+\cdots+a_1x+a_0,a_i\in GF(3)$$

又因为 $GF(3^n)$ 的特征为 3，故对 $\forall a\in GF(3^n)$ 有 $3a=0$，从而对 $\forall g(x)\in GF(3^n)$ 有

$$g(x)^3=g(x^3)\pmod{f(x)}$$

因而元素立方运算比较快，通常比乘法运算或平方运算快至少十倍[8]，而域逆运算又是最费时的运算，一般而言有

$$t(I)\gg t(M)>t(S)\gg t(C)$$

故本文通过引入多个中间变量，多做乘法运算以减少求逆次数，并尽可能将多项式变换成立方项相加减的形式来提高运算速度。

4. $GF(3^n)$上底层域快速算法介绍

4.1. $3P$、$9P$ 的直接计算公式

为了寻求 3^kP 的递推公式，我们先来推导 $3P$、$9P$ 的公式，然后观察、分析，进而推导出直接计算 3^kP 的公式。设椭圆曲线 $E:y^2=x^3+ax^2+b$，$P=(x_1,y_1)$ 为 E 上一点，$3P=(x_3,y_3)=2P+P$，先利用式(2.6)得到 $2P$，再利用式(2.5)得到 $3P$，即

$$u=\frac{ax_1}{y_1},x_2=u^2-a+x_1,y_2=u(x_1-x_2)-y_1\overset{\text{将}x_2\text{代入}}{=}-u(u^2-a)-y_1$$

$$\lambda=\frac{y_2-y_1}{x_2-x_1}\overset{\text{将上式得到的}x_2,y_2\text{代入}}{=}\frac{u(x_1-u^2+a-x_1)-y_1-y_1}{u^2-a+x_1-x_1}=-\frac{ax_1}{y_1}+\frac{y_1^3}{a(ax_1^2-y_1^2)}$$

$$x_3=\lambda^2-a-x_1-x_2=\lambda^2-a-x_1-(u^2-a+x_1)=\lambda^2-u^2-2x_1\overset{char(K)=3,\text{故}-2x_1=x_1}{=}\lambda^2-u^2+x_1$$

$$=\frac{x_1^9-a^2x_1^7+a^2bx_1^4-a^3bx_1^3-a^2b^2x_1+b^3}{a^2(x_1^3+b)^2}+x_1$$

$$=\frac{(x_1^3+b)^3-a^3bx_1^3-a^2x_1(x_1^3+b)^2}{a^2(x_1^3+b)^2}+x_1=\frac{(x_1^3+b)^3-b(ax_1)^3}{a^2(x_1^3+b)^2}$$

(将分子上的多项式变换成立方项相加减的形式)

$$y_3=\lambda(x_1-x_3)-y_1=\lambda\left(x_1-\lambda^2+u^2+2x_1\right)-y_1=-\lambda\left(\lambda^2-u^2+3x_1\right)-y_1$$

$$\overset{char(K)=3,\ 故3x_1=0}{=}-\lambda\left(\lambda^2-u^2\right)-y_1=\frac{\left(y_1^3\right)^3-ay_1^3\left[a\left(x_1^3+b\right)\right]^2}{a^3\left(x_1^3+b\right)^3}$$

综上所述可得，$3P=(x_3,y_3)$ 的直接计算公式为

$$\begin{cases}x_3=\dfrac{\left(x_1^3+b\right)^3-b(ax_1)^3}{a^2\left(x_1^3+b\right)^2}\\ y_3=\dfrac{\left(y_1^3\right)^3-ay_1^3\left[a\left(x_1^3+b\right)\right]^2}{a^3\left(x_1^3+b\right)^3}\end{cases}\tag{4.1}$$

若令

$$\begin{aligned}&B_1=x_1^3+b;\\ &A_1=ax_1;\\ &C_0=1,C_1=a\left(x_1^3+b\right)=aB_1;\\ &D_1=y_1^3;\\ &E_1=B_1^3-bA_1^3C_0^{12};\\ &F_1=D_1^3-aD_1C_1^2.\end{aligned}$$

则(4.1)式可化为

$$\begin{cases}x_3=\dfrac{E_1}{C_1^2}\\ y_3=\dfrac{F_1}{C_1^3}\end{cases}\tag{4.2}$$

其计算复杂度为 $1[I]+3[M]+2[S]+6[C]$，相当于不用计算 x_2,y_2 直接得到 x_3,y_3，而若先得到 x_2,y_2 再将其代入(2.5)式得到 x_3,y_3 的计算复杂度为 $2[I]+4[M]+2[S]$，相比之下，该算法虽然增加了 1 次乘法操作和 6 次立方操作，但却减少了 1 次求逆运算，不仅在逆乘率较大的情况下极大地减少了运算量，而且为下面推导 3^kP 的一般算法提供了便利。

再设 $9P=(x_9,y_9)=3(3P)=3(x_3,y_3)$，将 (x_3,y_3) 代入(2.6)式得到 $6P$，再将 $3P$ 和 $6P$ 代入(2.5)式得到 $9P$ 的直接计算公式，结果如下(为书写方便，将 x_1^3+b 记为 t)：

$$\begin{cases}x_9=\dfrac{\left[\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3\right]^3-a^3b\left[t^3-b(ax_1)^3\right]^3(at)^{12}}{\left\{a(at)^3\left[\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3\right]\right\}^2}\\ y_3=\dfrac{\left\{\left[\left(y_1^3\right)^3-ay_1^3(at)^2\right]^3\right\}^3-a\left[\left(y_1^3\right)^3-ay_1^3(at)^2\right]^3\left\{a(at)^3\left[\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3\right]\right\}^2}{\left\{a(at)^3\left[\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3\right]\right\}^3}\end{cases}\tag{4.3}$$

若令

$$B_2=\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3;$$
$$A_2=t^3-a^3bx_1^3=B_1^3-bA_1^3C_1^{12};$$

$$B_2=A_2^3+b\left(C_1^2\right)^3;$$

$$C_2=a(at)^3\left[\left(t^3-a^3bx_1^3\right)^3+b\left((at)^2\right)^3\right]=aB_2C_1^3;$$

$$D_2=\left[\left(y_1^3\right)^3-ay_1^3(at)^2\right]^3=\left(D_1^3-aD_1C_1^2\right)^3;$$
$$E_2=B_2^3-a^3bA_2^3C_1^{12};$$
$$F_2=D_2^3-aD_2C_2^2.$$

则(4.3)式可化为

$$\begin{cases}x_9=\dfrac{E_2}{C_2^2}\\ y_9=\dfrac{F_2}{C_2^3}\end{cases}\tag{4.4}$$

其计算复杂度为$1[I]+8[M]+4[S]+12[C]$，与逐次计算的复杂度$4[I]+4[S]+8[M]$相比，该算法牺牲了 12 次立方操作，但却减少了 3 次求逆运算，而$t(I)\gg t(C)$，可知该算法极大地提高了运算效率。

4.2. 3^kP 的直接计算公式

根据上一节$3P$、$9P$的计算，以此类推我们可以得到3^kP的递推公式。

设$3^kP=\left(x_{3^k},y_{3^k}\right)$，令

$$A_k=B_{k-1}^3-bA_{k-1}^3C_{k-2}^{12};$$
$$B_k=A_k^3+b\left(C_{k-1}^2\right)^3;$$
$$C_k=aB_kC_{k-1}^3;$$
$$D_k=\left(D_{k-1}^3-aD_{k-1}C_{k-1}^2\right)^3;$$
$$E_k=B_k^3-a^{3(k-1)}bA_k^3C_{k-1}^{12};$$
$$F_k=D_k^3-aD_kC_k^2,k\geq 2.$$

则有

$$\begin{cases}x_{3^k}=\dfrac{E_k}{C_k^2}\\ y_{3^k}=\dfrac{F_k}{C_k^3}\end{cases}\tag{4.5}$$

下面给出仿射坐标下直接计算3^kP的算法：

算法 1：*Direct computation of* 3^kP *in terms of affine coordinates*

$$
\begin{aligned}
&Input: P(x_1, y_1) \neq 0; k; a; b\\
&Output: T = 3^k P = (x_{3^k}, y_{3^k}).\\
&Step\ \ 1 \qquad set\ C_0, A_1, B_1, C_1, D_1\\
&\qquad C_0 \leftarrow 1;\\
&\qquad A_1 \leftarrow ax_1;\\
&\qquad B_1 \leftarrow x_1^{\,3} + b;\\
&\qquad C_1 \leftarrow a(x_1^{\,3} + b);\\
&\qquad D_1 \leftarrow y_1^{\,3};\\
&Step\ \ 2 \qquad for\ \ i\ \ from\ \ 2\ \ to\ \ k\ \ compute\\
&\qquad A_i \leftarrow B_{i-1}^{\,3} - bA_{i-1}^{\,3}C_{i-1}^{\,12};\\
&\qquad B_i \leftarrow A_i^3 + b(C_{i-1}^{\,2})^3;\\
&\qquad C_i \leftarrow aB_iC_{i-1}^{\,3};\\
&\qquad D_i \leftarrow (D_{i-1}^{\,3} - aD_{i-1}C_{i-1}^{\,2})^3;\\
&\qquad E_i \leftarrow B_i^3 - a^{3(i-1)}bA_i^3C_{i-1}^{\,12};\\
&\qquad F_i \leftarrow D_i^3 - aD_iC_i^{\,2};\\
&Step\ \ 3 \qquad compute\ \ x_{3^k}, y_{3^k}\\
&\qquad x_{3^k} \leftarrow \frac{E_k}{C_k^2};\\
&\qquad y_{3^k} \leftarrow \frac{F_k}{C_k^3}.
\end{aligned}
$$

在此递归过程中，先计算当$k=1$时的$A_k \sim F_k$，其计算复杂度为$1[M]+1[S]+5[C]$，此后$k-1$步每次需增加$5[M]+2[S]+6[C]$，最后还需计算x_{3^k}, y_{3^k}，需要$1[I]+2[M]+1[S]+1[C]$，故总的计算复杂度为：

$$
\begin{aligned}
&(1[M]+1[S]+5[C])+(k-1)(5[M]+2[S]+6[C])+(1[I]+2[M]+1[S]+1[C])\\
&=1[I]+(5k-2)[M]+(2k)[S]+(6k)[C].
\end{aligned}
$$

与此同时，直接计算需要$(2k)[I]+(4k)[M]+(2k)[S]$，虽然增加了$(k-2)[M]$，$(6k)[C]$，但却减少了$(2k-1)[I]$，当k很大时，直接计算的运算量将会极大地降低。

4.3. 算法性能比较

根据域中运算量的比较，假设$1[C] \approx 0.1[S], 1[S] \approx 0.8[M], 1[I] \approx 8[M]$，如表1。

通过表1，可以直观地看出：直接计算较逐次计算效率有显著的提升，并且随着k值的增大提升幅度也越来越大，特别是当$k \geq 4$时，提升幅度均可达到60%以上。

5. 基于滑动窗口标量乘算法改进

5.1. 算法介绍

利用底域直接计算$3^k P$的优势，对传统滑动窗口算法[9] [10]改进，得到一个新的标量乘法，新算法在赋值阶段比传统算法效率有较大提高。由于新算法预处理栈中存储的都是非零窗口，所以还能抵抗边际信道攻击。

滑动窗口算法分为三个阶段：①标量k编码阶段，计算出k的NAF_w形式，w为窗口宽度；②预计算

Table 1. Comparison of different methods in the underlying domain
表 1. 不同方法在底层域运算量比较

运算对象	计算方法	I	M	S	C	近似 M 值	减少量
$3P$	逐次计算	2	4	2		21.6	39.4%
	直接计算	1	3	2	6	13.08	
$9P$	逐次计算	4	8	4		43.2	53.3%
	直接计算	1	8	4	12	20.16	
3^3P	逐次计算	6	12	6		64.8	58.0%
	直接计算	1	13	6	18	27.24	
3^4P	逐次计算	8	16	8		86.4	60.3%
	直接计算	1	18	8	24	34.32	
3^kP	逐次计算	$2k$	$4k$	$2k$		$21.6k$	67.2%
	直接计算	1	$5k-2$	$2k$	$6k$	$7.08k+6$	

阶段，计算出 $\left\{2P,4P,\cdots,\left\lfloor\left(3^w-1\right)/2\right\rfloor P\right\}$ 的值存储在一个表中，供下一阶段查询使用；③赋值阶段，利用预计算表计算出 kP [11] [12]。

将标量 k 改写成(5.1)式：

$$k=\sum_{i=0}^{l-1}k_ir^i=k_0+k_1r+\cdots+k_{l-1}r^{l-1} \tag{5.1}$$

那么计算标量乘 kP 可改写成(5.2)式：

$$kP=r\left(\cdots r\left(rk_{l-1}P+k_{l-2}P\right)+\cdots+k_1P\right)+k_0P,k_i\in D_r,D_r\text{为系数数字集合.} \tag{5.2}$$

算法 2： 3-NAF_w 展开

$Input: a\ \ positive\ \ integer\ \ k; w>1$
$Output: k=(k_{l-1},\cdots,k_0)_{3-NAF_w}.$
$Step\ \ 1\qquad i\leftarrow 0;$
$Step\ \ 2\qquad while\ \ k\geq 1\ \ do$
$2.1\ \ k_i\leftarrow k \bmod 3^w;$
$2.2\ \ if\ (k_i>3^w/2)$
$2.2.1\ \ k_i\leftarrow k_i-3^w;$
$2.2.2\ \ k\leftarrow k-k_i;$
$else\ \ k_i\leftarrow 0;$
$2.3\ \ k\leftarrow k/3;$
$2.4\ \ i\leftarrow i+1;$
$Step\ \ 3\qquad Return((k_{l-1},\cdots,k_0)_{3-NAF_w}).$

算法 3： 3-NAF_w标量乘算法

Input: a *positive integer* $k; w>1; P\in GF(3^n)$

Output: kP.

Step 1　利用算法2，得到$k=(k_{l-1},\cdots,k_0)_{3-NAF_w}$;

Step 2　$P_i=iP$,其中$i\in\{1,2,\cdots,\lfloor(3^w-1)/2\rfloor\}$;

Step 3　$Q\leftarrow k_{l-1}P; i\leftarrow l-2$;

Step 4　*while* $i\geq 0$ *do*

4.1 *if* $(k_i=\cdots=k_0=0)$

4.1.1 $u\leftarrow(k_i,\cdots,k_0)$;

4.1.2 $t\leftarrow i+1$;

else 找最大t，使$u\leftarrow(k_i,\cdots,k_{i-t+1})$满足$k_i=\cdots=k_{i-t+2}=0, k_{i-t+1}\neq 0$;

4.2 $Q\leftarrow 3^tQ$(算法1)

4.3 *if* $(u>0)$

4.3.1 $Q\leftarrow Q+P_u$;

else $Q\leftarrow Q-P_{-u}$;

4.4 $i\leftarrow i-t$;

Step 5　*Return* Q.

5.2. 算法性能分析与结论

算法 2 是对 k 进行重新编码，3-NAF_w 展开数字集 $D_{w,r}=\left\{0,\pm1,\pm2,\cdots,\pm\left\lfloor\left(3^w-1\right)/2\right\rfloor\right\}$，且在其展开式中任意 w 个邻接数字中最多只有 1 个非零数字[10]，故任意两个非零数字之间至少有 $w-1$ 个 0，其非零数字密度为 $\frac{(r-1)}{w(r-1)+1}=\frac{2}{2w+1}$，算法平均需要 $\lfloor\log_3 k\rfloor+1$ 次 3 倍点和 $\frac{\lfloor\log_3 k\rfloor+1}{2}=\frac{2(\lfloor\log_3 k\rfloor+1)}{2w+1}$ 次点加。

算法 3 与传统滑动窗口算法的区别在于赋值阶段，对于固定基点的标量乘法，预计算表不需要频繁更换，故建立预计算表在整个标量乘中所占比例不大[9]，而赋值阶段对于提高标量乘的计算复杂度就显得格外重要。赋值阶段的复杂性与非零窗口的个数 t 有关，文献[13]给出非零窗口的密度是 $w+1$，而在 3-NAF_w 展开式中任意个非零数字之间至少有 $w-1$ 个 0，故 Step 4 最多执行 $\lfloor m/w\rfloor+1$ 次，每次执行一次 3^tQ 和一次点加，故算法 3 的计算复杂度为

$$(\lfloor m/w\rfloor+1)3^tQ+(\lfloor m/w\rfloor+1)A \tag{5.3}$$

由于滑动窗口可变，因此可以有效地减少非零窗口的长度，从而减少点加运算量；此外，由于在 Step 4 中采用直接计算 3^tQ 的递归算法，在逆乘率较高时有效地减少了算法中 3 倍点总的计算量。

6. 结束语

$GF(3^n)$ 作为 $GF(p^n)$ 的一种特殊类型，具有其他有限域不可比拟的特点，它可提供更多更加灵活的加密方案且具备较高的安全性。本文采用递归思想，研究了 $GF(3^n)$ 椭圆曲线在仿射坐标下直接计算 3^kP 算法，对底层域进行改进，当 $k\geq 4$ 时，相对于逐次计算而言效率提升到了 60% 以上；同时采用滑动窗口标量乘的方法对上层域进行改进，既有效减少了非零窗口的长度又减少了算法中 3 倍点总的运算量。

基金项目

国家自然科学基金 NSFC11271040 资助。

参考文献 (References)

[1] Koblitz, N. (1987) Elliptic Curve Cryptosystems. *Mathematics of Computation*, **48**, 203-209. http://dx.doi.org/10.1090/S0025-5718-1987-0866109-5

[2] 汪宏，李宝，于伟. 特征 3 有限域上椭圆曲线 Montgomery 算法[J]. 通信学报, 2008, 29(10): 25-29.

[3] Kim, K.H., Kim, S.I. and Choe, J.S. (2007) New Fast Algorithms for Arithmetic on Elliptic Curves over Fields of Characteristic Three. Cryptology ePrint Archive: Report 2007/179. http://eprint.iacr.org/2007/179

[4] 沈丽敏，陈恭亮，游永兴. 特征为 3 的域上的椭圆曲线点的快速计算[J]. 数学杂志, 2004, 24(5): 557-560.

[5] 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): 42-47.

[6] Kim, K.H. (2007) A Note on Point Multiplication on Supersingular Elliptic Curves over Ternary Fields. Cryptology ePrint Archive: Report 2007/310. http://eprint.iacr.org/2007/310

[7] Hankerson, D.，等著，张焕国，译. 椭圆曲线密码学导论[M]. 北京：电子工业出版社, 2005.

[8] Smart, N.P. and Westwood, E.J. (2003) Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three. *Applicable Algebra in Engineering, Communication and Computing*, **13**, 485-497. http://dx.doi.org/10.1007/s00200-002-0114-0

[9] Al-Daoud, E., Mahmod, R., Rushdan, M. and Kilicman, A. (2002) A New Addition Formula for Elliptic Curves over GF(2^n). *IEEE Transactions on Computers*, **51**, 972-975. http://dx.doi.org/10.1109/TC.2002.1024743

[10] Solinas. J.A. (2000) Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography*, **19**, 195-249. http://dx.doi.org/10.1023/A:1008306223194

[11] 侯红祥. 椭圆曲线上快速标量乘的研究[D]: [硕士学位论文]. 扬州：扬州大学, 2008.

[12] Takagi, T., Yen, S.M. and Wu, B.C. (2004) Radix-*r* Non-Adjacent Form. *Proceedings of 7th Information Security Conference, ISC* 2004, Palo Alto, 27-29 September 2004, 2004, 99-110. http://dx.doi.org/10.1007/978-3-540-30144-8_9

[13] 刘连浩，申勇. 椭圆曲线密码体制中标量乘法的快速算法[J]. 计算机应用研究, 2009, 26(3): 1104-1108.