基于双基数的快速标量乘算法*)

殷新春[1,2] 侯红祥[1] 谢 立[2]
(扬州大学计算机科学与工程系 扬州 225009)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 标量乘法是整个椭圆曲线密码体制实现的瓶颈。本文在有效表示标量 k 方面,引用一个新的数域系统——双基数系统,将标量的双基数链长度限制在 $O\left(\frac{\log k}{\log\log k}\right)$ 范围内,减少标量乘法中的上层运算。在底层域快速算法研究方面,推导出直接计算 3^kP 快速算法。最后结合直接计算 2^kP,$2P\pm Q$,$3P\pm Q$ 及 3^kP 快速算法。给出基于双基数的快速标量乘新算法,新算法的效率优于 Dimitrov 算法及传统标量乘算法。

关键词 椭圆曲线密码体制,标量乘法,双基数系统,底层域运算,3^kP

Fast Scalar Multiplication Based on DBNS

YIN Xin-chun[1,2] HOU Hong-xiang[1] XIE Li[2]
(Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009, China)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[2]

Abstract Scalar multiplication is the bottleneck of elliptic curve cryptography. A new number field system——double base number system (DBNS) is employed in denoting scalar k. The length of double base chain lies in the range of $O\left(\frac{\log k}{\log\log k}\right)$. In field fast algorithm aspect, a fast algorithm of directly computing 3^kP is proposed in way of trading reversions for multiplications. The new double base scalar multiplication algorithm is integrated with fast algorithms of direct computing 2^kP, $2P\pm Q$, $3P\pm Q$ and 3^kP. The efficiency of new algorithm is superior to algorithm given by Dimitrov and traditional scalar multiplication.

Keywords Elliptic curves cryptography, Scalar multiplication, DBNS, Field operation, 3^kP

1 引言

标量乘法是椭圆曲线密码体制中最基本最耗时的运算,其运算过程分为两个层次:一是上层运算,主要是椭圆曲线上点之间的运算;另一个层次称为底层域运算,主要是在有限域中为实现椭圆曲线上点之间的运算而做的一些求逆、乘法、平方等的操作,本文中分别记作[i],[m],[s]。相应地,研究标量乘法有两条思路:一是研究标量 k 的有效表示,尽量减少上层运算,如 NAF[1],DBNS[2-5];二是底层域快速算法研究[6-11],其根本目标是减少底层域运算量,目前一种方法是适当增加底层域乘法与平方运算来减少求逆运算,如在域中直接 $3P$,$4P$,$4P\pm Q$,2^kP,$2P\pm Q$,$3P\pm Q$。

标量 k 有效表示与底层域快速算法相融合,实现"多元优化"是目前研究标量乘法快速实现的一个趋势[12]。2005 年 Dimitrov[5] 等人将底层域快速算法 $3P$,$4P$,$2P\pm Q$,$3P\pm Q$,$4P\pm Q$[8] 与双基数融合,有效地减少了底层域运算量,从而加快了标量乘法的运行速度。本文延续将求逆转化为适量乘法的思想,给出在底层域中直接计算 3^kP 的算法,然后将其与 2^kP,$2P\pm Q$ 的快速算法一起融合到双基数标量乘法中,得到更高效的双基数标量乘新算法。

2 背景知识

2.1 双基数系统

定义 1 s-整数是最大素因子不超过第 s 个素数的正整数。

定义 2 双基数系统中,每个正整数 k 都可表示成若干个 2-整数和的形式,如(1)式:

$$k=\sum_{i=1}^{m}s_i2^{b_i}3^{t_i},\text{其中 } s_i\in\{-1,1\}\text{及 } b_i,t_i\geqslant 0 \tag{1}$$

很明显在双基数系统中,数的表现形式并不唯一,但有如下性质:

定理 1[5] 每个正整数 k 最多可以表示成 $O\left(\frac{\log k}{\log\log k}\right)$ 个 2-整数的和。

定理 1 限制了(1)式中长度 m 的取值范围。

定义 3 $\forall k>0$,以及序列 $(K_1,K_2,\cdots,K_i,K_{i+1},\cdots,K_m)_{m>0}$ 满足(2)式:

$$K_1=1,K_{i+1}=2^u3^vK_i+s,\text{其中 } s\in\{-1,1\}\text{及 } u,v\geqslant 0 \tag{2}$$

如果 $\exists m$ 使得 $K_m=k$,则 $(K_1,K_2,\cdots,K_m)$ 叫做 k 的双基数链。

由于双基数链长度在定理 1 规定的范围内,因此 Dim-

*)国家自然科学基金(NSF 60473012),江苏省六大人才高峰(06-E-025)。**殷新春** 博士,教授,CCF 会员,主要研究方向为并行与分布计算、信息安全;**侯红祥** 硕士研究生,主要研究方向为信息安全;**谢 立** 博士生导师,教授,CCF 会员,主要研究方向为并行与分布处理、信息安全。

itrov 将其应用来计算椭圆曲线上的标量乘法 kP，减少了点加运算的次数。为便于求双基数链，在满足(1)式的双基数系统中找出满足(3)式的双基数形式：

$$k=\sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i}，其中\ s_i\in\{-1,1\}及\ b_i,t_i\geqslant 0\ 且递降 \quad (3)$$

2.2 底层域快速算法

仿射坐标下，已知椭圆曲线 $E: y^2=x^3+ax+b$ 上两点 $P(x_1,y_1)$，$Q(x'_2,y'_2)$，求 $2P(x_2,y_2)$及 $P+Q=(x_3,y_3)$由(4)、(5)两式给出：

$$\lambda_1=(3x_1^2+a)/(2y_1),\ x_2=\lambda_1^2-2x_1,\ y_2=\lambda_1(x_1-x_2)-y_1 \quad (4)$$

$$\lambda_2=(y'_2-y_1)/(x'_2-x_1),x_3=\lambda_2^2-x_1-x'_2,y_3=\lambda_2(x_1-x_3)-y_1 \quad (5)$$

通常公式(4)称为倍点运算，公式(5)称为两互异点相加运算，简称点加运算。如果底层域中的求逆运算记为$[i]$，平方运算记为$[s]$及乘法运算记为$[m]$，则倍点需要的运算量为 $1[i]+2[s]+2[m]$，点加需要的运算量为 $1[i]+1[s]+2[m]$。

由于求逆运算其运算量一般是乘法运算的 10 倍左右[13]，因此计算标量乘法可以将求逆运算转化为适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[8]中给出 $3P,4P,2P\pm Q,3P\pm Q,4P\pm Q$ 的快速算法。

本文只介绍 $3P$ 算法，其他请参考文献[8]。一般计算 $3P(x_3,y_3)$可以转化首先计算 $2P(x_2,y_2)$，然后将 $P(x_1,y_1)$加到 $2P$ 上，这样 $3P(x_3,y_3)=P(x_1,y_1)+2P(x_2,y_2)$，如果令 $Q=2P$，则上面两步可由(4)，(5)两式来计算。将(4)式中的 y_2 代入(5)式中的 λ_2：

$$\lambda_2=\frac{2y_1}{x_1-x_2}-\lambda_1 \quad (6)$$

不需计算 y_2；将(4)式中 $x_2=\lambda_1^2-2x_1$ 变形得 $\lambda_1^2=x_2+2x_1$ 代入(5)式 x_3：

$$x_3=(\lambda_1+\lambda_2)(\lambda_1-\lambda_2)+x_1 \quad (7)$$

所以只要求出 λ_1,λ_2 就可求出 x_3,y_3；令：

$$d=(2y_1)^2(3x_1)-(3x_1^2+a)^2 \quad (8)$$

由(4)式中的 λ_1 变形：

$$(3x_1^2+a)^2=\lambda_1^2(2y_1)^2 \quad (9)$$

将(9)式代入(8)式，同时 $\lambda_1^2=x_1+2x_1$，所以 $d=(2y_1)^2(x_1-x_2)$；再令 $D=(2y_1)d$，$I=D^{-1}$，则 $\frac{1}{2y_2}=dI$，$\frac{1}{x_1-x_2}=(2y_1)^3I$，所以 $\lambda_1=(3x_1^2+a)dI$，$\lambda_2=((2y_1)^4-(3x_1^2+a)d)I$，将 λ_1,λ_2 代入(7)，(4)分别求得 x_3,y_3。

上述计算 $3P$ 的计算复杂性是 $1[i]+4[s]+7[m]$，相对于原来的 $2[i]+3[s]+4[m]$，提高了计算效率。下面延续将求逆转化为乘法的思想，推导出直接计算 3^kP 的一般算法。

3 直接计算 3^kP 算法

3.1 直接计算 $3P$

为了推导直接计算 3^kP 的一般算法，从计算 $3P$ 开始，这里引入一些中间变量：

$A_1=3x_1^2+a$；

$B_1=2y_1$；

$d_1=3x_1B_1^2-A_1^2$；

$C_1=B_1^4-A_1d_1$；

$D_1=B_1d_1$；

$E_1=(C_1+A_1d_1)(C_1-A_1d_1)+x_1D_1^2$

$F_1=2C_1(x_1D_1^2-E_1)-B_1D_1^3$

$3P(x_3,y_3)$就可用以下的式子来计算：

$$x_3=\frac{E_1}{D_1^2};y_3=\frac{F_1}{2D_1^3}$$

计算复杂性为 $1[i]+5[s]+11[m]$（一次求逆是计算 $2D_1^3$ 的逆），此算法在保证求逆操作不增加的情况下，增加了一些乘法或平方操作，这样做的目的是为了接下来推导 3^kP 算法。

3.2 直接计算 $9P$

假设已知 $3P(x_3,y_3)$，计算 $9P(x_9,y_9)$可继续用上面的方法，由附录中的推导，得到由 $3P(x_3,y_3)$的 x 及 y 坐标，即由 E_1,F_1 和 D_1 计算 $9P(x_9,y_9)$的过程：

$A_2=3E_1^2+aD_1^4$；

$B_2=F_1$；

$d_2=3E_1B_2^2-A_2^2$；

$C_2=B_2^4-A_2d_2$；

$D_2=B_2d_2$；

$E_2=(C_2+A_2d_2)(C_2-A_2d_2)+E_1D_2^2$；

$F_2=2C_2(E_1D_2^2-E_2)-B_2D_2^3$

$9P(x_9,y_9)$就可用以下的式子来计算：

$$x_9=\frac{E_2}{(D_1,D_2)^2};y_9=\frac{F_2}{2(D_1D_2)^3}$$

计算复杂性为 $1[i]+6[s]+15[m]$，由于 $D_1^4=D_1^3D_1$，而 3.1 已经计算过 D_1^3，因此这里只算一次乘法，加上计算 E_1,F_1 和 D_1 所需的 $5[s]+8[m]$，从 $P(x_1,y_1)$计算 $9P(x_9,y_9)$的复杂性为 $1[i]+11[s]+23[m]$，如果使用文献[8]的两次三倍点运算，计算复杂性是 $2(1[i]+4[s]+7[m])=2[i]+8[s]+14[m]$，所以直接计算 $9P$ 的方法优于两次三倍点的临界点是 $1[i]=3[s]+9[m]$，当 $1[s]\approx 0.8[m]$时，临界点是 $1[i]=11.4[m]$。

3.3 直接计算 $27P$

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

$A_3=3E_2^2+a(D_1D_2)^4$；

$B_3=F_2$；

$d_3=3E_2B_3^2-A_3^2$；

$C_3=B_3^4-A_3d_3$；

$D_3=B_3d_3$；

$E_3=(C_3+A_3d_3)(C_3-A_3d_3)+E_2D_3^2$；

$F_3=2C_3(E_2D_3^2-E_3)-B_3D_3^3$

$27P(x_{27},y_{27})$就可用以下的式子来计算：

$$x_{27}=\frac{E_3}{(D_1D_2D_3)^2};y_{27}=\frac{F_3}{2(D_1D_2D_3)^3}$$

计算复杂性为 $1[i]+6[s]+17[m]$，由于 $(D_1D_2)^4=D_1^4D_2^3D_2$，而 3.2 已经计算过 D_1^4,D_2^3，因此这里只算两次乘法；加上计算 E_2,F_2 和 D_2 所需的 $10[s]+18[m]$，那么从 $P(x_1,y_1)$计算 $27P(x_{27},y_{27})$的复杂性为 $1[i]+16[s]+35[m]$，如果使用文献[8]的三次三倍点运算，计算复杂性是 $3(1[i]+4[s]+7[m])=3[i]+12[s]+21[m]$，所以直接计算 $27P$ 的方法优于三次三倍点的临界点是 $2[i]=4[s]+14[m]$，当 $1[s]\approx 0.8[m]$时，临界点是 $1[i]=8.6[m]$。

3.4 直接计算 3^kP 算法

从上面给出的计算 $3P,9P$ 和 $27P$ 的过程，可发现从 P 直接计算 3^kP 有一种普遍的方法，下面就给出由 P 直接计算

3^kP 的算法：

算法 1 直接计算 3^kP 形式的算法

输入 基点 $P_1=(x_1,y_1)$；k；
输出 椭圆曲线上的点 $3^kP=(x_{3^k},y_{3^k})$；
Step 1 计算 $A_1=3x_1^2+a$；$B_1=2y_1$；
$d_1=3x_1B_1^2-A_1^3$；$C_1=B_1^4-A_1d_1$；
$D_1=B_1d_1$；
$E_1=(C_1+A_1d_1)(C_1-A_1d_1)+x_1D_1^2$；
$F_1=2C_1(x_1D_1^2-E_1)-B_1D_1^3$；
Step 2 for i from 2 to k do
$A_i=3E_{i-1}^2+a(\prod_{j=1}^{i}D_j)^4$；
$B_i=F_{i-1}$；$d_i=3E_{i-1}B_i^2-A_i^3$；
$C_i=B_i^4-A_id_i$；$D_i=B_id_i$；
$E_i=(C_i+A_id_i)(C_i-A_id_i)+E_{i-1}D_i^2$；
$F_i=2C_i(E_{i-1}D_i^2-E_i)-B_iD_i^3$；
Step 3 compute: $x_{3^k}=\dfrac{E_k}{(\prod_{j=1}^{k}D_j)^2}$；$y_{3^k}=\dfrac{F_k}{2(\prod_{j=1}^{k}D_j)^3}$；
Step 4 Output (x_{3^k},y_{3^k})；

由给定椭圆曲线上一点 P 计算 3^kP，用上面的算法，其计算复杂性为 $1[i]+(5k+1)[s]+12k[m]$。具体计算过程如下：第一步中须计算 x_1^2，B_1^2，A_1^2，B_1^4 和 D_1^2 共 5 次平方，以及须计算 $x_1B_1^2$，A_1d_1，B_1d_1，$x_1D_1^2$，$(C_1+A_1d_1)(C_1-A_1d_1)$，$C_1(x_1D_1^2-E_1)$ 和 $B_1D_1^3$ 共 8 次乘法；第二步中须计算 E_{i-1}^2，B_i^2，A_i^2，B_i^4 和 D_i^2 共 $5(k-1)$ 次平方，以及还须计算 $a(\prod_{j=1}^{i}D_j)^4$，A_id_i，$E_{i-1}B_i^2$，B_id_i，$(C_i+A_id_i)(C_i-A_id_i)$，$E_{i-1}D_i^2$，$C_i(E_{i-1}D_i^2-E_i)$ 和 $B_iD_i^3$ 共 $11(k-1)$ 乘法（其中 $(\prod_{j=1}^{i}D_j)^4$ 可以转化为 $(\prod_{j=1}^{i-2}D_j)^4D_{i-1}^3D_{i-1}$ 两次乘法）；第三步中共须一次求逆、一次平方以及 $k+3$ 次乘法。如果使用文献[8]的 k 次三倍点运算，计算复杂性是 $k(1[i]+4[s]+7[m])=k[i]+4k[s]+7k[m]$，所以直接计算 3^kP 的算法优于 k 次三倍点的临界点是 $(k-1)[i]=(k+1)[s]+5k[m]$，当 $1[s]\approx0.8[m]$ 时，临界点是 $1[i]=\dfrac{5.8k+0.8}{k-1}[m]\approx5.8[m]$。

表 1 底层域快速算法

算法	运算量
P±Q	1[i]+1[s]+2[m]
2P±Q	1[i]+2[s]+9[m]
3P	1[i]+4[s]+7[m]
2^kP	1[i]+(4k+1)[s]+(4k+1)[m]
3^kP	1[i]+(5k+1)[s]+12k[m]

下面给出基于双基数标量乘新算法，新算法中涉及到的底层域快速算法及其运算量如下表 1 所示，为简单起见，其中 $P\pm Q$，$2P\pm Q$，$3P$，2^kP 和 3^kP 等快速算法，以后分别标记为 A，DA，T，DD 和 DT。

4 新的双基数标量乘算法

首先将给定的整数 k 表示成(3)式，然后求出其双基数链 $(K_1,K_2,\cdots,K_i,K_{i+1},\cdots,K_m)$，那么计算标量乘法 kP 便可依照该条双基数链求得：$K_1P\to K_2P\to\cdots\to K_iP\to K_{i+1}P\to\cdots\to K_mP$，每个 $K_iP\to K_{i+1}P$ 都是计算形如 $2^u3^vQ\pm P$ 的过程。

下面讨论计算形如 $2^u3^vQ\pm P$ 运算的最优路径：

Ⅰ. 指数 v 的变化有三种情况：$v=0$，$v=1$ 和 $v\geqslant2$，所有的 $\pm$P 的操作将在Ⅱ，Ⅲ，Ⅳ中讨论，此处只讨论 3^vQ 部分的最优方法，因此当 $v=0$，不做任何操作，$v=1$ 时，应用 $3P$ 快速算法，$v\geqslant2$ 时用 3^kP 算法；

Ⅱ. 当 $u=0$ 时，$2^u3^vQ\pm P$ 转化为 $3^vQ\pm P$ 的计算，计算路径为 $Q\to3^vQ\to3^vQ\pm P$，运算量为 $2[i]+(5v+2)[s]+(12v+2)[m]$；

Ⅲ. 当 $u=1$ 时，$2^u3^vQ\pm P$ 转化为 $2\cdot3^vQP\pm$ 的计算，存在两条计算路径，分别为 $Q\to3^vQ\to2\cdot3^vQ\pm P$ 和 $Q\to2Q\to3^v(2Q)\to3^v(2Q)\pm P$，由表 1 可知路径 1 需要 $2[i]+(5v+3)[s]+(12v+9)[m]$，路径 2 需要 $3[i]+(5v+4)[s]+(12v+4)[m]$，路径 1 优于路径 2 的临界点是 $1[i]=1[s]+5[m]$，在目前计算能力下路径 1 表现更出色；

Ⅳ. 当 $u\geqslant2$ 时，首先讨论 $2^uQ\pm P$ 的计算，存在两条计算路径，分别为 $Q\to2^uQ\to2^uQ\pm P$ 和 $Q\to2^{u-1}Q\to2(2^{u-1}Q)\pm P$，由表 1 可知前者需要 $2[i]+(4u+2)[s]+(4u+3)[m]$，后者需要 $2[i]+(4u-1)[s]+(4u+6)[m]$，由于 $1[s]<1[m]$，因此路径 1 被选中；那么 $2^u3^vQ\pm$ 的计算将遵循以下路径 $Q\to3^vQ\to2^u3^vQ\to2^u3^vQ\pm P$；而当 $u=2$ 时，符合 DA 快速算法的应用情形，所以路径 $Q\to3^vQ\to2\cdot3^vQ\to2(2\cdot3^vQ)\pm P$ 的运算量更低。

综合以上讨论，可列出从 K_iP 到 $K_{i+1}P$ 所有计算路径及其所用的快速算法和运算量：

表 2 从 K_iP 到 $K_{i+1}P$ 所有运算

u	v	运算路径	快速算法	运算量
0	0	$Q\to Q\pm P$	A	$1[i]+1[s]+2[m]$
0	1	$Q\to3Q\to3Q\pm P$	T, A	$2[i]+5[s]+9[m]$
0	>=2	$Q\to3vQ\to3vQ\pm P$	DT, A	$2[i]+(5v+2)[s]+(12v+2)[m]$
1	0	$Q\to2Q\pm P$	DA	$1[i]+2[s]+9[m]$
1	1	$Q\to3Q\to2(3Q)\pm P$	T, DA	$2[i]+6[s]+16[m]$
1	>=2	$Q\to3vQ\to2(3vQ)\pm P$	DT, DA	$2[i]+(5v+3)[s]+(12v+9)[m]$
2	0	$Q\to2Q\to2(2Q)\pm P$	D, DA	$2[i]+4[s]+11[m]$
2	1	$Q\to3Q\to2(3Q)\to2(2(3Q))\pm P$	T, D, DA	$3[i]+8[s]+18[m]$
2	>=2	$Q\to3vQ\to2(3vQ)\to2(2(3vQ))\pm P$	DT, D, DA	$3[i]+(5v+5)[s]+(12v+11)[m]$
>2	0	$Q\to2uQ\to2uQ\pm P$	DD, A	$2[i]+(4u+2)[s]+(4u+3)[m]$
>2	1	$Q\to3Q\to2u(3Q)\to2u(3Q)\pm P$	T, DD, A	$3[i]+(4u+6)[s]+(4u+10)[m]$
>2	>=2	$Q\to3vQ\to2u(3vQ)\to2u(3vQ)\pm P$	DT, DD, A	$3[i]+(4u+5v+3)[s]+(4u+12v+3)[m]$

从表 2 可以看到，计算标量乘法 kP 的双基数链上每一步 $K_iP\to K_{i+1}P$ 最多需要底层域三次求逆，那么依据该双基数链来计算标量乘法 kP 最多需要约 $3\,O\left(\dfrac{\log k}{\log\log k}\right)$ 次求逆。

基于上面的思想，从正整数 k 的双基数形式和椭圆曲线上一点 P 计算标量乘法 kP 的算法由算法 2 给出：

算法 2 新的双基数标量乘算法

输入 $k=\sum_{i=1}^{m}s_i2^{b_i}3^{t_i}$，$s_i\in\{-1,1\}$ 并且 $b_1\geqslant b_2\geqslant\cdots\geqslant b_m\geqslant0$，$t_1\geqslant t_2\geqslant\cdots\geqslant t_m\geqslant0$，$P\in E(F_q)$
输出 曲线上另外一点 $kP\in E(F_q)$
Step1. $Q\leftarrow s_1P$；
Step2. for i from 1 to m−1 do
2.1 $u\leftarrow b_i-b_{i+1}$；
2.2 $v\leftarrow t_i-t_{i+1}$；

```
    2.3 if v=0 then Q←Q;
      else if v=1 then Q←3Q;
           else Q←3^vQ;
    2.4 if u=0 then Q←Q+ s_{i+1}P;
      else if u=1 then Q←2Q+s_{i+1}P;
           else if u=2 then Q←2Q; Q←2Q+s_{i+1}P;
                else Q←2^uQ; Q←Q+s_{i+1}P;
Step3. if t_m=0 then Q←Q;
      else if t_m=1 then Q←3Q;
           else Q←Q;
Step4. if b_m=0 then Q←Q+s_{i+1}P;
      else if b_m=1 then Q←2Q+s_{i+1}P;
           else if b_m=2 then Q←2Q; Q←2Q+s_{i+1}P;
                else Q←2^{b_m}Q; Q←Q+s_{i+1}P;
Step5. Return Q;
```

5 算法性能分析

为了衡量算法 2 的复杂度，定义一个计算从 K_iP 到 $K_{i+1}P$ 的运算量的公式 W_{i+1}：

$$W_{i+1}=\delta_{v,1}T+(1-\delta_{v,1})DT(v)+\delta_{u,0}A+\delta_{u,1}DA+(1-\delta_{v,0}-\delta_{v,1})(DD(u)+A) \quad (6)$$

(6)式中 $a=b$ 时 $\delta_{a,b}=1$；$a\neq b$ 时 $\delta_{a,b}=0$，则整个 DBNS 算法的复杂度为：

$$W=\sum_{i=1}^{m}W_i,\text{其中 } W_1=0 \quad (7)$$

例如当 $k=314159$ 时，其双基数形式为 $314159=2^{12}3^4-2^{11}3^2+2^83^1+2^43^1-2^03^0$，双基数链为(1，17，409，6545，314159)，则算法 2 的执行过程如表 3 所示。

表 3 中的 Q 和 QA 分别代表底层域快速计算 $4P$ 和 $4P+Q$，算法 2 用 DD 和 D+DA 代替这两运算，新增了 DT 运算，来达到减少求逆运算的目的。从中可知，用算法2计算 $314159P$ 的运算量 $10[i]+71[s]+100[m]$，优于 Dimitrov 算法 ($13[i]+55[s]+95[m]$)的临界点是 $3[i]=16[s]+5[m]$，以及优于 Ciet 算法[8] ($15[i]+42[s]+95[m]$) 的临界点是 $5[i]=29[s]+5[m]$。

表 3 计算 $314159P$ 过程

i	K_i	s	u	v	算法 2 的 W_i	Dimitrov 算法的 W_i
1	1	1	0	0	0	0
2	$18K_1-1=17$	−1	12−11=1	4-2=2	DT(2)+DA	2T+DA
3	$24K_2+1=409$	1	11−8=3	2−1=1	T+DD(3)+A	T+Q+DA
4	$16K_3+1=6545$	1	8−4=4	1−1=0	DD(4)+A	Q+QA
5	$48K_4-1=314159$	−1	4−0=4	1−0=1	T+DD(4)+A	T+Q+QA
总运算量					2T+2DD(4)+DD(3)+DT(2)+DA+3A	2DA+4T+3Q+2QA

根据文献[5]中表 3 的分析知，160bits 的 k 的双基数链的平均长度为 38，192bits，224bits，256bits 分别为 43，52，64。相同条件下，算法 2 最坏情况下 $W_i=DT(v)+DD(u)+A$ 需要 $3[i]$，例如，160bits 条件下，算法 2 最坏情况下需要 $114[i]$，对应于 Dimitrov 算法的 $119[i]$以及 Ciet 算法的 $127[i]$。由此可见对于任意的 k，算法 2 也能减少求逆运算。

随机选取大素数域上不同长度的 k，求其双基数链，并根据该双基数链分析出用算法 2 和 Dimitrov 算法计算标量乘法 kP 所用的快速算法和运算量的平均值，得到表 4，如下所示。

表 4 算法平均运算量比较

$\lvert k\rvert$	b_1，t_1	快速算法		运算量	
		算法 2	Dimitrov 算法	算法 2	Dimitrov 算法
160	76，53	7D+9T+15DT+14DD+11DA+22A	14DA+45T+8TA+20Q+11QA	$77[i]+575[s]+994[m]$	$117[i]+465[s]+813[m]$
192	108，53	5D+8T+17DT+18DD+12DA+23A	17DA+47T+6TA+33Q+12QA	$83[i]+710[s]+1138[m]$	$133[i]+593[s]+969[m]$
224	140，53	8D+8T+16DT+19DD+13DA+24A	16DA+47T+6TA+46Q+16QA	$88[i]+804[s]+1272[m]$	$153[i]+722[s]+1115[m]$
256	153，65	7D+10T+21DT+23DD+21DA+31A	18DA+57T+8TA+48Q+19QA	$106[i]+958[s]+1483[m]$	$178[i]+808[s]+1280[m]$

假设 k 的二进制长度($|k|$)为 n 位，用二元法计算标量乘法平均需要 n 次倍加，$n/2$ 次点加；NAF 法需要计算点加的次数将降低至 $n/3$ 次；w-NAF 法在不考虑预计算的情况下，平均需要 $n/(w+1)$次点加。特殊地，取$|k|=160$，在使用 DA 快速运算的前提下，用二元法平均需要 $160[i]+320[s]+880[m]$；NAF 法需要 $160[i]+320[s]+691[m]$；4-NAF 需要 $160[i]+320[s]+544[m]$。从表 4 中可以看出 Dimtrov 算法需要较少的$[i]$，而算法 2 进一步减少了$[i]$的个数，稍微增加一些乘法操作。此外，算法 2 不需要任何预计算。当标量长度为 160bits 及$[i]/[m]=10$ 时，算法 2 比传统二元法效率提高 18.71%，比 2-NAF 提高 12.68%，比 4-NAF 提高 7.34%，比 Dimitrov 算法提高 5.56%，192bits 时提高 8.56%，224bits 时 13.26%，256bits 时 10.71%。

结束语 本文将底层域快速算法与双基数系统融合，对双基数标量乘法进行改进，延续将求逆转化为乘法的思想，推导出直接计算 3^kP 快速算法，再结合直接计算 $2P+Q$，2^kP 快速算法的优势，给出了新的基于双基数标量乘算法，算法的效率比以往的标量乘算法效率都要高。接下来的工作是将其与窗口算法、NAF 算法结合起来，以及考虑其他坐标下的算法变形。

参 考 文 献

[1] Solinas J. An improved algorithm for arithmetic on a family of elliptic curves[J]. Advances in Crytology-Crypto'97, LNCS, Springer-Verlag, 1997,1294:357-371

[2] Dimitrov V S, Jullien G A, Miller W C. An algorithm for modular exponentiation[J]. Information Processing Letters, 1998, 66(3):155-159

[3] Gordon D M. A survey of fast exponentiation methods[J]. Journal of algorithms, 1998,27(1):129-146

[4] Dimitrov V S, Jullien G A. A new number representation with applications[J]. IEEE Circuits and Systems Magazine, 2003, 3(2):6-23

[5] Dimitrov V S, Imbert L, Mishra P K. Fast elliptic curve point multiplication using double-base chains [EB/OL]. http://eprint.iacr.org/, 2005

[6] Guajardo J, Paar C. Efficient algorithms for elliptic curve cryptosystems [A]. crypt'97, LNCS[C], Springer-Verlag, 1997,294:343-356

(下转第 195 页)

$J(\alpha_0,\alpha_1)=\frac{1}{18+19}=0.027$

这说明 t_1 时刻的粗集在结构特性与 t_0 时刻完全相似，可以采用同一评价体系进行分析。

经反干扰后有可能达到的希望状态：

$[\alpha]/s_1=\{U_{31},U_{32},U_{33},U_{34},U_{35},U_{41},U_{42},U_{43},U_{51},U_{52},U_{53}\}$

$[\alpha]/s_2=\{U_{11},U_{14},U_{21},U_{22},U_{23},U_{32},U_{33},U_{41},U_{51}\}$

$[\alpha]/s_3=\{U_{23},U_{44}\}$

希望状态下属性类$[\alpha]_{hope}$对于属性类$[\alpha]$的粒度贴近度为〈2，1.5，2/7〉。

若采取方案 F_{11}，经反干扰后仍有部分雷达设备不能发挥正常效能。而属性集合 α_1 关于状态 S 的等价类化分为

$[\alpha]_{F_{11}}/s_1=\{U_{31},U_{34},U_{35},U_{42},U_{43},U_{51},U_{52},U_{53}\}$

$[\alpha]_{F_{11}}/s_2=\{U_{11},U_{14},U_{21},U_{32},U_{41},U_{54}\}$

$[\alpha]_{F_{11}}/s_3=\{U_{22},U_{23},U_{24},U_{33},U_{44}\}$

此时对应的状态转移矩阵为

$$P_{11}=\begin{bmatrix}1 & 0 & 0\\ 2/6 & 4/6 & 0\\ 0 & 2/7 & 5/7\end{bmatrix}$$

属性类$[\alpha]_{F_{11}}$对于属性类$[\alpha]$的粒度贴近度为〈8/5，1，5/7〉。

若采取方案 F_{12}，经反干扰后仍有部分雷达设备不能发挥正常效能。而属性集合 α_1 关于状态 S 的等价类化分为

$[\alpha]/s_1=\{U_{31},U_{32},U_{34},U_{35},U_{41},U_{42},U_{43},U_{51},U_{52},U_{53}\}$

$[\alpha]/s_2=\{U_{11},U_{14},U_{21},U_{22},U_{54}\}$

$[\alpha]/s_3=\{U_{23},U_{24},U_{33},U_{44}\}$

此时对应的状态转移矩阵为

$$P_{12}=\begin{bmatrix}1 & 0 & 0\\ 4/6 & 2/6 & 0\\ 0 & 3/7 & 4/7\end{bmatrix}$$

属性类$[\alpha]_{F_{12}}$对于属性类$[\alpha]$的粒度贴近度为〈2，5/6，4/7〉。

当无法获悉属性集中属性权重时，令报酬函数

$r(r_i,F)=\sum_S|GRD([\alpha]_F)-GRD([\alpha]_{hope})|$

对于本例可得：

$r(t_1,F_{11})=|2-8/5|+|1.5-1|+|2/7-5/7|=1.3286$

$r(t_1,F_{12})=|2-2|+|5/6-1/5|+|4/7-2/7|=0.9524$

因为 $r(t_1,F_{12})<r(t_1,F_{11})$，所以在 t_1 时刻，红方采取反干扰方案 F_{12} 后出现的状态更贴近希望状态，此时$[\alpha]/s_1$ 膨胀趋势最大，$[\alpha]/s_3$ 收缩趋势最大。

4.3 评估过程

报酬函数 $r(t_i,F)$ 是从属性类粒度贴近度的角度定义的，没有考虑属性的权重信息。有时候虽然$[\alpha]/s_1$ 呈现膨胀趋势，可是由于属性权值的不同可能会引起$[\alpha]/s_1$ 整体权值的下降，所以报酬函数还应该考虑属性的权值影响。在本例中，采取方案 F_{11} 时$[\alpha]/s_1$ 整体权值为 0.2478，采取方案 F_{12} 时$[\alpha]/s_1$ 整体权值为 0.3132，所以并不会对方案的选择带来影响。

结束语 本文针对战场态势评估中的状态转移模型，提出了基于变异 S-粗集的策略选择方法。首先，提出用 S-粗集理论来描绘战场态势评估中的状态转移模型。其次，建立了 S-粗集、变异 S-粗集的结构差离度的概念。只有在结构差离度小于某个阈值的时候，S-粗集在 t 时刻与 $t+1$ 时刻才具有可比性。给出了变异 S-粗集的属性类的粒度贴近度概念，用粒度贴近度来构建报酬函数。最后，采用序贯决策方法，根据当前战场态势和给定的一系列的被选方案，选取策略，使得战场态势向希望状态转移。

参 考 文 献

[1] 刘同明，夏祖勋，解洪成. 数据融合技术及其应用[M]. 北京：国防工业出版社，1998

[2] 康耀红. 数据融合理论与应用[M]. 西安：西安电子科技大学出版社，1997

[3] Haussman W. Sequential decision problems: a model to exploit existing forecasters[J]. Management Science, 1969, 16(2): 93-111

[4] 崔玉泉，史开泉. 粗集的动态特性分析及应用[J]. 中国管理科学，2003，11(6)：66-70

[5] 雷英杰. 基于直觉模糊推理的态势与威胁评估研究[D]. 学位论文. 西安电子科技大学，2005

（上接第 189 页）

[7] Han Y, Tan P C. Direct computation for elliptic curve cryptosystem[A]. CHES'99[C], Springer-Verlag, 1999: 328-340

[8] Ciet M, Joye M, Lauter K, et al. Trading inversions for multiplications in elliptic curve cryptography [EB/OL]. Cryptology ePrint Archive, Report, 2003: 257-277

[9] Eisenträger K, Lauter K, Montgomery P L. Faste elliptic curve arithmetic and improved weil pairing evaluation[J]// Joye M, ed. Topics in cryptology-CT-RSA 2003, 2003, 2612: 343-354

[10] Sakai Y, Sakurai K. Efficient scalar multiplications on elliptic curves without repeated doublings and their practical performance[A]. ACISP 2000[C], LNCS, Springer-verlag, 2000, 1841: 59-73

[11] Sakai Y, Sakurai K. Efficient scalar multiplications on elliptic curves with direct computations of several doublings [J]. IEICE Trans. Fundamentals, 2001, E84-A(1): 120-129

[12] Adachi D, Hirata T. Combination of mixed coordinates strategy and direct computations for efficient scalar multiplications[J]. Communications, Computers and signal Processing, 2005: 117-120

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

[14] Hankerson D, Lòpez J, Menezes A. Software implementation of elliptic curve cryptography over binary fields[C]. Cryptographic Hardware and Embedded Systemns-CHES 2000, LNCS, Springer-Verlag, 2000, 1965: 1-24

附录

假设已知 $3P(x_3,y_3)$，那么计算 $9P(x_9,y_9)$ 就可以继续 3.1 的方法，则有：

$3x_3^2+a=\frac{3E_1^2+aD_1^4}{D_1^4}$，令 $A_2=3E_1^2+aD_1^4$；

$2y_3=\frac{F_1}{D_1^3}$，令 $B_2=F_1$；

$d=(2y_3)^2(3x_3)-(3x_3^3+a)^2=\frac{F_1^2}{D_1^6}\frac{3E_1}{D_1^2}-\frac{A_2^2}{D_1^8}=\frac{3E_1B_2^2-A_2^2}{D_1^8}$，令 $d_2=3E_1B_2^2-A_2^2$；$D=(2y_3)d_2=\frac{F_1}{D_1^3}\frac{d_2}{D_1^8}=\frac{B_2d_2}{D_1^{11}}$，令 $D_2=B_2d_2$；

则 $\lambda_1'=\frac{3x_3^2+a}{2y_3}=\frac{A_2}{D_1F_1}=\frac{A_2d_2}{D_1D_2}$；$\lambda_2'=\frac{(2y_3)^4}{D}-\lambda_1'=\frac{F_1^4D_1^{11}}{D_1^{12}D_2}-\frac{A_2d_2}{D_1D_2}=\frac{B_2^4-A_2d_2}{D_1D_2}$，令 $C_2=B_2^4-A_2d_2$；

则 $x_9=\frac{B_2^4}{D_1D_2}\frac{B_2^4-2A_2d_2}{D_1D_2}+\frac{E_1}{D_1^2}=\frac{(C_2+A_2d_2)(C_2-A_2d_2)+E_1D_2^2}{(D_1D_2)^2}$，令 $E_2=(C_2+A_2d_2)(C_2-A_2d_2)+E_1D_2^2$；

$y_9=\frac{C_2}{D_1D_2}\left(\frac{E_1}{D_1^2}-\frac{E_2}{(D_1D_2)^2}\right)-\frac{F_1}{2D_1^3}=\frac{2C_2(E_1D_2^2-E_3)-B_2D_2^3}{2(D_1D_2)^3}$，令 $F_2=2C_2(E_1D_2^2-E_2)-B_2D_2^3$。