一个新的基于radix－8的标量乘算法

程一飞，陈文莉
（安庆师范学院 计算机系，安徽 安庆 246011）

摘 要：椭圆曲线标量乘是椭圆曲线密码系统中最关键、最耗时的运算，因此如何快速高效实现标量乘运算是研究的重点。目前常见的标量乘算法有：double－and－add算法，NAF算法，MOF算法等，但它们都是基于radix－2编码表示的，无论采用何种编码，倍点运算的次数都不变，减少的只是点加（或点减）运算的次数。提出一个基于radix－8表示的新的编码方法，及一个基于radix－8表示的标量乘算法，通过用八倍点运算代替倍点运算，且编码是从左到右（即从最高位向最低位）进行，编码和主计算可以合并，提高实现效率并节省内存空间。实验结果表明，该算法较经典的double－and－add算法能够提高效率30％以上。

关键词：椭圆曲线密码系统；标量乘；radix－8表示；改进Booth算法；编码
中图分类号：TP309.7　　**文献标识码**：A　　**文章编号**：1673－629X(2007)10－0155－03

A New Radix－8 Representation Based Scalar Multiplication Algorithm

CHENG Yi-fei, CHEN Wen-li
(Computer Science Dept. of Anqing Teachers College, Anqing 246011, China)

Abstract: The scalar multiplication dominates the execution time of elliptic curve cryptographic schemes, so various methods have been studied to enhance the performance of this operation. The double－and－add algorithm, the NAF algorithm and the MOF algorithm etc are the frequently used methods implementing this operation, but the common drawback of these algorithms is that they are based on the radix－2 representations. So no matter what coding is used, only the number of point addition (or subtraction) can be diminished, but the number of point doubling can not be diminished. In this paper, a new coding method based on the radix－8 representation is proposed. A new radix－8 representation based scalar multiplication algorithm is given. This method adopts point octuple instead of point doubling, and examines the integer from left to right (from the most significant digit to the least significant digit). This results in the merging of coding and evaluation stages. So the proposed algorithm can improve the performance and reduce the memory consumption of scalar multiplication operation. The result of the experiment shows that more than 30% performance enhances over the algorithm using double－and－add method.

Key words: elliptic curve cryptography; scalar multiplication; radix－8 representation; improved Booth's algorithm; coding

0 引 言

椭圆曲线密码系统（ECC）是1985年分别由Neal Koblitz[1]和V.S. Miller[2]独立提出的，相对于其它公钥密码系统（如RSA，ElGamal），其具有计算速度快、存储空间小、带宽要求低等优点，特别适用于Smart卡和无线应用环境，受到了人们的广泛关注，成为最有希望的公钥密码系统。如何快速高效地实现椭圆曲线密码系统一直是人们的一个研究重点。而其中最关键、最耗时的运算是椭圆曲线标量乘运算，即计算kP，其中P是椭圆曲线上的一个点，k是一个大整数且$k \in [1, n-1]$，n是P点的阶，很多学者都致力于如何快速实现标量乘运算，并提出了多种方法[3,4]。标量k的整数表示形式在这些方法中起着至关重要的作用，象那些具有最小平均海明权值的表示（如ωNAF表示）更具吸引力。这是因为最小平均海明权值就意味着最少的点加或点减运算[5,6]，但是所有这些方法都没有减少倍点运算的次数。

笔者提出一个基于radix－8表示的新的编码方法，并提出一个基于改进Booth算法的radix－8表示的标量乘算法，用八倍点运算代替倍点运算，且编码是从左到右（即从最高位向最低位）进行，编码和主计算

收稿日期：2006－12－25
基金项目：安徽省教育厅自然科学研究项目（2006KJ079B）
作者简介：程一飞（1976－），男，安徽怀宁人，讲师，硕士，研究方向为计算机密码学。

可以合并。这样提高实现效率并节省内存空间。

1 椭圆曲线和点乘

在有限域 $K = F_2^m$ 上 Weierstrass 方程可以转换成形如 $E: y^2 + xy = x^3 + ax^2 + b$ 的形式，其中 $a, b \in F_2^m$，且 $b \neq 0$。通过特定的加法运算，$E(K) = \{(x,y) \in K \times K \mid y^2 + xy = x^3 + ax^2 + b\} \cup \{O\}$，其中 O 表示无穷远点，构成一个交换群。

设 $E: y^2 + xy = x^3 + ax^2 + b(b \neq 0)$ 是 F_2^m 中给定的一条曲线，$P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ 是 E 上的两个点，且 $P_1 \neq -P_2$，那么 P_3 点的坐标可以如下计算：

当 $P_1 \neq P_2$ 时，$x_3 = \lambda^2 + \lambda + a + x_1 + x_2, y_3 = y_1 + \lambda(x_1 + x_3) + x_3, \lambda = \frac{y_2 + y_1}{x_2 + x_1}$，此运算称为点加运算，而当 $P_1 = P_2$ 时 $\lambda = x_1 + \frac{y_1}{x_1}, x_3 = \lambda^2 + \lambda + a$，$y_3 = x_1^2 + (\lambda + 1)x_3$，此运算称为倍点运算。

$R(x_3, y_3) = 8P_1(x_1, y_1)$[7]，其中 $x_3 = \frac{\omega^2 + \omega\rho}{\rho^2} + a, y_3 = \frac{(\upsilon^2)^2 + \omega\rho x_3}{\rho^2} + x_3, \gamma = x_1^2, \eta = \gamma + y_1,$

$\delta = \eta^2 + \eta x_1 + a\gamma, \xi = \eta x_1 + \gamma, \zeta = \delta(\delta + \xi) + \gamma^2\gamma, \tau = \delta\gamma, \upsilon = \zeta^2 + \tau\zeta + \tau^2\zeta, \rho = \upsilon\tau^2,$

$\omega = \upsilon(\upsilon + \zeta\tau) + (\tau\delta^2)^2 + \rho$

点乘(标量乘)：kP 表示点 P 与自身相加 k 次，即 $kP = P + P + \cdots + P$，共 k 个 P 相加，称为点乘或标量乘。

2 新的编码

定义 1：设 k 是一个整数，以 $a_{n-1}\cdots a_1\ a_0$ 表示，即 $k = a_{n-1}2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$，则称 $a_{n-1}\cdots a_1\ a_0$ 为 k 的 radix - 2 表示，通常写成 $(a_{n-1}\cdots a_1\ a_0)_2$。

定义 2[8]：若 $(a_{n-1}\cdots a_1\ a_0)_2$ 满足以下三个条件，则称其为 ωNAF 表示($\omega \geqslant 2$)。

(1) 零位为正数；(2) 任意连续 ω 位数中，最多有一位非零；(3) 任意一个非零数均为绝对值小于 $2^{\omega-1}$ 的奇数。

在所有基于绝对值小于 $2^{\omega-1}$ 的奇数和 0 作为数码的表示形式中，ωNAF 表示具有最小海明权值(即非零个数最少)。但是 n 位二进制数，其 ωNAF 表示的长度至少为 n，这就意味着 ωNAF 表示仅能减少点加或点减运算的次数，而不能减少倍点运算的次数。

定义 3：设 k 是一个整数，以 $a_{m-1}\cdots a_1\ a_0$ 表示，即 $k = a_{m-1}8^{m-1} + \cdots + a_1 8^1 + a_0 8^0$，则称 $a_{m-1}\cdots a_1\ a_0$ 为 k 的 radix - 8 表示，通常写成 $(a_{m-1}\cdots a_1\ a_0)_8$。

将整数 k 的 radix - 2 表示 $(a_{n-1}\cdots a_1\ a_0)_2$ 从右到左每三位一组很容易生成整数 k 的 radix - 8 表示形式 $b_{m-1}\cdots b_1\ b_0, b_i \in \{0,1,2,3,4,5,6,7\}$。当然也可以从左到右生成 radix - 8 表示形式，这时就要考虑 radix - 2 表示形式的长度 n 为 3 的倍数，如果不是 3 的倍数，则需要在最高位前补充相应的 0 使得整个长度是 3 的倍数，然后每三位一组转换成 radix - 8 表示形式。不管是从左到右还是从右到左，都需要预计算 $2P, 3P, 4P, 5P, 6P, 7P$，存储 $P, 2P, 3P, 4P, 5P, 6P, 7P$ 七个点。为了减少预存储的点的个数，采用另一种新的编码方法。

k 是 radix - 2 表示的 n 位二进制数 $(a_{n-1}\cdots a_1\ a_0)_2$，即 $k = a_{n-1}2^{n-1} + \cdots + a_1 2^1 + a_0 2^0$，因为 $2^x = 2^{x+1} - 2^x$，所以 k 可以表示成：$k = (a_{n-1} - 0)2^n + (a_{n-2} - a_{n-1})2^{n-1} + \cdots + (a_0 - a_1)2^1 + (0 - a_0)2^0 = (a_{n-1} - 0)2^n + (a_{n-2} - a_{n-1})2^{n-1} + \cdots + [(a_4 - a_5)2^2 + (a_3 - a_4)2^1 + (a_2 - a_3)2^0]8^1 + [(a_1 - a_2)2^2 + (a_0 - a_1)2^1 + (0 - a_0)2^0]8^0$。通过此方法编码即可将 $(a_{n-1}\cdots a_1\ a_0)_2$ 从右到左转换成 radix - 8 编码。即位 a_i, a_{i-1}, a_{i-2} 以 a_{i-3} 作为参考位编码成 $b_i, b_{i-1}, b_{i-2}, b_i = a_{i-1} - a_i, b_{i-1} = a_{i-2} - a_{i-1}, b_{i-2} = a_{i-3} - a_{i-2}$，相应的，位 $a_{i-3}, a_{i-4}, a_{i-5}$ 以 a_{i-6} 作为参考位编码成 $b_{i-3}, b_{i-4}, b_{i-5}, b_{i-3} = a_{i-4} - a_{i-3}, b_{i-4} = a_{i-5} - a_{i-4}, b_{i-5} = a_{i-6} - a_{i-5}$ 四位一组，从右到左进行，i 初值为 2，每循环一次，i 加 3，并且置 $a_{-1} = a_n = 0$。例如 $k = 228 = (11100100)_2$，则 k 的 radix - 8 表示可以由表 1 从上向下运算得到。

表 1 228 的 radix - 8 表示

a_i	a_{i-1}	a_{i-2}	a_{i-3}	b_i	b_{i-1}	b_{i-2}	c_i
1	0	0	0	-1	0	0	-4
1	0	0	1	-1	0	1	-3
0	1	1	1	1	0	0	4

从上面分析可知，i 初值为 2，每循环一次，i 加 3，则若对整数 k 从左到右进行编码，必须保证 i 的终止值为 2，因为步长为 3，故要求 $i + 1$ 的初值必须为 3 的倍数，即要求 k 的二进制表示的长度为 3 的倍数加 2。若长度不为 3 的倍数加 2，则可以通过置 a_{n+1}(或 a_{n+2}, a_{n+1}) 位为 0，i 的初值为 $n + 1$(或 $n + 2$) 来实现。

算法 1：二进制表示的正整数转换为八进制的带符号表示

输入：n 位二进制数 $(a_{n-1}\cdots a_1\ a_0)_2$，其中 $a_i \in \{0, 1\}$

输出：m 位八进制表示 $(c_{m-1}\cdots c_1\ c_0)_8$，其中 $c_i \in$

$\{0, \pm 1, \pm 2, \pm 3, \pm 4\}$

1)if n 为 3 的倍数加 2 then{start = n;$a_{-1} = 0$;$a_n = 0$;}

else if n 为 3 的倍数加 1 then{start = $n+1$;$a_{-1} = 0$;$a_n = 0$;$a_{n+1} = 0$}

else {start = $n+2$;$a_{-1} = 0$;$a_n = 0$;$a_{n+1} = 0$;$a_{n+2} = 0$}

2)m = (start + 1)/3;

3)for i from start downto 1 step − 3 do

$c_i = (a_{i-1} - a_i) * 4 + (a_{i-2} - a_{i-1}) * 2 + (a_{i-3} - a_{i-2})$;

4) return$(c_{m-1} \cdots c_1 c_0)_8$

3　新的基于 radix－8 表示的标量乘算法

3.1　算　法

算法 2:基于 radix－8 表示的标量乘算法

输入: n 位二进制数$(a_{n-1} \cdots a_1 a_0)_2$,其中 $a_i \in \{0,1\}$,P 点

输出:kP

预计算:

计算 $2P$,$3P$,$4P$;

主计算:

1)$R = O$(无穷远点);

2)if n 为 3 的倍数加 2 then{start = n;$a_{-1} = 0$;$a_n = 0$;}

else if n 为 3 的倍数加 1 then{start = $n+1$;$a_{-1} = 0$;$a_n = 0$;$a_{n+1} = 0$}

else {start = $n+2$;$a_{-1} = 0$;$a_n = 0$;$a_{n+1} = 0$;$a_{n+2} = 0$}

3)for i from start downto 1 step − 3 do

(1) $R = 8R$;

(2) $t = (a_{i-1} - a_i) * 4 + (a_{i-2} - a_{i-1}) * 2 + (a_{i-3} - a_{i-2})$;

(3) $R = R + tP$;

4) return R

3.2　算法分析

算法 2 共需进行(start + 1)/3 次循环,且算法 2 中 $R = R + tP$ 运算表面上看每执行一次循环就要进行一次,即要进行一次点加运算,但实际上 t 有可能为 0,而当 $t = 0$ 时是不进行点加运算的,因此点加运算的次数小于循环的次数。由于 t 由相邻的四位二进制数决定,t 取值见表 2。

由表 2 可知 t 为零的概率为 1/8,也就意味着仅需要进行循环次数的 7/8 次点加运算,故需要进行 7(start + 1)/24 次点加运算。八倍点运算共需进行(start + 1)/3 次,倍点运算共需进行(start + 1)/3 次。

表 2　radix－2 表示对应 radix－8 表示转换表

a_i	a_{i-1}	a_{i-2}	a_{i-3}	b_i	b_{i-1}	b_{i-2}	t
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	1	−1	1
0	0	1	1	0	1	0	2
0	1	0	0	1	−1	0	−2
0	1	0	1	1	−1	1	3
0	1	1	0	1	0	−1	3
0	1	1	1	1	0	0	4
1	0	0	0	−1	0	0	−4
1	0	0	1	−1	0	1	−3
1	0	1	0	−1	1	−1	−3
1	0	1	1	−1	1	0	−2
1	1	0	0	0	−1	0	−2
1	1	0	1	0	−1	1	−1
1	1	1	0	0	0	−1	−1
1	1	1	1	0	0	0	0

因此本算法对 n 位二进制标量共需要进行 7(n + 1)/24 次点加运算和(n + 1)/3 次八倍点运算。下面通过分析点加运算、倍点运算以及八倍点运算分别需要的基本域运算(加、减、乘、除、平方)等,来具体比较新的算法和经典的 double－and－add 算法。分别以 A 表示加法、M 表示乘法、D 表示除法、S 表示平方。表 3 列出三种运算所需基本运算的次数。

表 3　点加、倍点以及八倍点运算所需基本运算次数

	加	乘	平方	除
点加	9	1	1	1
倍点	5	1	1	1
八倍点	17	14	7	1

利用表 3 数据来具体比较新的算法和经典的 double－and－add 算法,见表 4。

表 4　新的算法和经典的 double－and－add 算法性能比较

	Double－and－add	新算法
M	$\frac{3}{2}n$	$\frac{119}{24}(n+1)$
D	$\frac{3}{2}n$	$\frac{1}{3}(n+1)$
S	$\frac{5}{2}n$	$\frac{21}{8}(n+1)$

4　结　论

提出了一个新的基于 radix－8 的编码方法,该编码方法既可以从右到左又可以从左到右进行编码,并提出了一个新的标量乘算法,在新的标量乘算法中,通过采用八倍点运算代替倍点运算来提高运算效率,并

(下转第 161 页)

进,同一暂停点失效节点的个数少,增加速度比 DSR 慢。

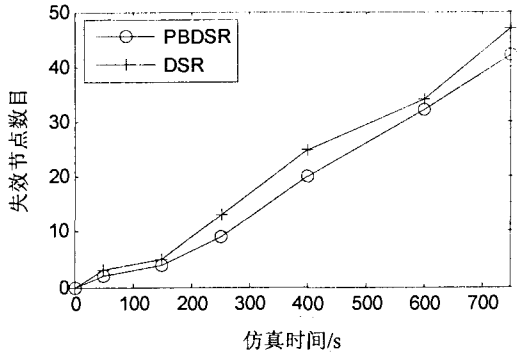

图 5　采用 281.8mW 发射功率仿真结果

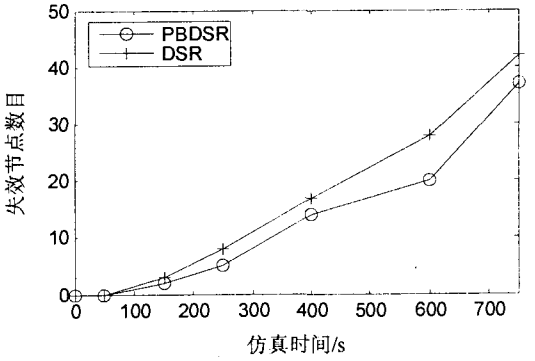

图 6　采用 115.44mW 发射功率仿真结果

②该算法继承了 DSR 路由协议按需和源路由方式,仅在需要通信的节点间维护路由,减少了路由维护的代价,所采用的路由缓冲技术,可以减少发起路由发现过程的次数。

③引入了节点的位置信息,节点建立了基于目的节点的路由选择区域,因此节点在路由发现的过程中可以缩小路由请求帧的发送范围(如果整个网络是均匀分布理论上则可以减少 3/4),这样就大大减少了参与到路由发现过程中的节点的数量,降低了整个网络的能量消耗。

④在路由选择时采用链路总体距离和移动性参数作为路由选择的性能指标,充分的考虑了链路的总体能量消耗以及节点的移动性,选择的路由是节能的、相对稳定的。

4　结　论

PBDSR 利用节点的地理位置信息,构建了一个路由选择区域,减少了路由发现过程中参与的节点数目,选择路由时充分考虑了节点及链路整体的移动性,总体上减少了路由发现过程启动的次数以及每次参与路由发现过程节点数量,从仿真结果看 PBDSR 是比 DSR 协议具有更长网络生存时间和稳定性的算法。

参考文献:

[1] 郑少仁,王海涛,赵志峰. Ad Hoc 网络技术[M]. 北京:人民邮电出版社,2005:71－73.

[2] Malizd J J. Dynamic source routing in ad hoc wireless networks [M]. [s.l.]:Kluwer Academic,1996:135－138.

[3] 王申涛,杨　浩,周　熙. Ad Hoc 网络 DSR 路由协议仿真性能分析[J]. 无线通信技术,2006,32(5):10－12.

[4] 李明峰,冯宝红,刘三枝. GPS 定位技术及其应用[M]. 北京:国防工业出版社,2006:63－72.

[5] 沈长星. 基于地理位置的移动 Ad Hoc 网络路由协议研究[D]. 北京:北京邮电大学,2006.

(上接第 157 页)

且由于编码方法可以从左到右进行,故可以和主计算阶段合并,这样可以节省内存空间,因此非常适用内存空间受限制的设备,如 smart 卡等。实验结果表明,该算法较经典的 double－and－add 算法能够提高效率 30%以上。

参考文献:

[1] Koblitz N. Elliptic Curve Cryptosystems[J]. Math Computation,1987,48:203－209.

[2] Miller V. Uses of Elliptic Curve in Cryptography[J]. Springer－Verlag Lecture Notes in Computer Science,1987,218:417－426.

[3] Gordon D M. A Survey of Fast Exponentiation Methods[J]. Journal of Algorithms,1998,27:129－146.

[4] Hankerson D, Menezes A, Vanstone S. Guide to Elliptic Curve Cryptography[M]. [s.l.]:Springer, 2004.

[5] Avanzi R M. A Note on the Signed Sliding Window Integer Recoding and its Left－to－Right Analogue[C]// In Selected Areas in Cryptography 2004, LNCS Vol 3357. [s. l.]: Springer－Verlag,2004:130－143.

[6] Muir J A, Stinson D R. New Minimal Weight Representations for Left－to－Right Window Methods[C]// In Cryptographers' Track at the RSA Conference 2005, LNCS Vol 3376. [s.l.]:Springer－Verlag,2005:366－383.

[7] Guajardo J, Paar C. Efficient algorithms for elliptic curve cryptosystems[C]// In Advances in Cryptology－CRYPTO'97,LNCS Vol 1294. [s.l.]:Springer－Verlag,1997:342－356.

[8] Okeya K, Schmidt－Samoa K, Spahn C, et al. Signed Binary Representations Revisited[C]// In Advances in Cryptology－CRYPTO 2004, LNCS Vol 3152. [s.l.]:Springer－Verlag, 2004:123－139.