

求解双基链的改进型贪心算法

李 明, 王鲲鹏

(中科院研究生院 信息安全国家重点实验室, 北京 100049)

摘 要: 为了提升贪心算法的表现, 更有效地求解整数的双基展开, 文中采用了 2-Line Search 方法, 将树形分支结构引入了贪心算法. 改进加速了贪心算法的收敛, 使得在相同输入下, 新算法返回的双基表达式的长度比普通的贪心算法减少了 8% 以上, 在椭圆曲线标量乘法的计算代价方面也获得了至少 2% 的收益.

关键词: 双基链; 标量乘法; 贪心算法; 椭圆曲线密码

中图分类号: TP309

文献标识码: A

文章编号: 1000-7180(2010)03-0177-04

Greedy Algorithm of Convert Integers into Double-Base Chains

LI Ming, WANG Kun-peng

(State Key Lab of Information Security, GUCAS, Beijing 100049, China)

Abstract: In order to improve greedy algorithm so that it could convert integers into Double-Base Chains more efficiently, we introduce tree structure into greedy algorithm by taking advantage of 2-Line Search method. This improvement accelerates the convergence of greedy algorithm. Compared with the greedy algorithm, new algorithm gains more than 8% in the average length of double-base expansions and 2% in the cost of scalar multiplication.

Key words: double-base chains; scalar multiplication; greedy algorithm; elliptic curves cryptography

1 引言

由于椭圆曲线密码具有密钥短、安全性高等其他公钥密码体制所无法比拟的特点, 近年来已经成为密码学领域研究的热点之一^[1-2]. 在许多基于椭圆曲线的密码方案中, 计算点乘 $[k]P$ (又称标量乘法) 是最耗时的操作, 其中 P 是椭圆曲线上一点, k 为整数. 可以采用多种方法来加快速度: (1) 通过对 k 的展开形式的优化来提高运算效率, 代表算法有 NAF 表示、 τ -adic 表示、双基表示等^[3-6]; (2) 引入预计算的窗口算法和梳子算法等^[3]; (3) 根据椭圆曲线的特点使用不同的坐标表示, 例如仿射坐标、射影坐标和 Inverted Edwards 坐标^[7]等.

贪心算法常用来求解整数的双基表达式. 在文献^[8]中提出了基于 Line Search 方法的贪心算法, 该算法在效率方面有着良好的表现. 但是, 贪心算法并不能保证多轮迭代后的结果仍然是最优的. 受

Tree-based 算法^[9]的启发, 文中将树形分支结构引入贪心算法, 有效地改善了上述问题, 提升了贪心算法的表现. 已知参数 B , 新算法为每个目标值创建 2 个分支, 分别对应不同的近似值; 并在每一层仅保留收敛速度最快的 B 个分支作为下一层的目标, 如此反复直至某一支为 0. 这种改进加速了贪心算法的收敛, 使得在相同输入下, 新算法求出的双基链长度相比普通的贪心算法有明显的降低, 而且在标量乘法的代价方面, 新算法相比 NAF 减少了 6.6% 以上.

2 双基数系统

Dimitrov 和 Cooklev 在文献^[10]中首次提出了双基数系统 (DBNS) 的概念, 并在随后将其应用到了椭圆曲线密码学^[11].

在 DBNS 中, 一个整数 n 被表示为:

收稿日期: 2009-03-30; 修回日期: 2009-06-05

基金项目: 国家“八六三”计划项目(2006AA01Z427); 国家“九七三”计划项目(2007CB311201)

$$n = \sum_{i=1}^l d_i 2^{a_i} 3^{b_i}, \quad d_i \in S \quad (1)$$

当 $S = \{1\}$ 时, 式(1) 被称为无符号的 DBNS 表示; 当 $S = \{-1, 1\}$ 时, 式(1) 为有符号的 DBNS 表示. 显而易见, 整数 n 的 DBNS 表达式并不唯一, 准确地说是高度冗余的. 要找到一个项数尽可能少的表达式, 贪心算法是解决这个问题的有效算法.

例 1 令 $n = 841232$, 利用贪心算法可以求出 n 的 DBNS 表示:

$$841232 = 2^7 3^8 + 1424,$$

$$1424 = 2^1 3^6 - 34,$$

$$34 = 2^2 3^2 - 2.$$

$$\text{综上, } 841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2^1.$$

Dimitrov 等人给出了贪心算法求出的 n 的 DBNS 表达式的项数上界: 对于任意整数 n , 贪心算法都可以在至多 $O(\frac{\log n}{\log \log n})$ 项内求出其 DBNS 展开式^[12].

然而, n 的 DBNS 表达式往往并不适合用来计算标量乘法, 因为无序的指数使得必须计算 $Q/2$ 或者 $Q/3$, 而这样的操作是相当耗时的. 为了避免上述情况的发生, 双基表达式的指数序列必须满足同时递减, 即 $a_1 \geq a_2 \geq \dots \geq a_l$ 且 $b_1 \geq b_2 \geq \dots \geq b_l$. 满足这一性质的双基展开式叫做双基链(DBC)^[11]. 只要在普通贪心算法的每轮迭代中将目标值对应的上轮近似值的指数作为本轮该目标值搜索的指数上界就得到了一个用来求解 DBC 的贪心算法.

例 2 将上例中的 n 展开为 DBC 形式如下:

$$841232 = 2^7 3^8 + 1424,$$

$$1424 = 2^1 3^6 - 34,$$

$$34 = 3^3 + 7,$$

$$7 = 3^2 - 2,$$

$$2 = 3^1 - 1.$$

即 $841232 = 2^7 3^8 + 2^1 3^6 - 3^3 - 3^2 + 3^1 - 1$. 一般而言, 一个整数的 DBC 长度要大于其对应的 DBNS 表达式.

3 (2,B)-Greedy 算法

(2, B)-Greedy 算法的核心是 2-Line Search 方法, 该方法是 Line Search 方法的改进, 二者的差别在于为每个目标值所创建的分支数. 为了阐明 2-Line Search 方法的执行过程, 以 127 为例, 并设 $a_{\max} = 6, b_{\max} = 4$ (见图 1), 其中 $a_{\max}$ 和 $b_{\max}$ 分别对应基数 2 和 3 的指数上界. 2-Line Search 方法将搜索每

一个 $z = 2^a 3^b$, 对应于图中直线 $a \log 2 + b \log 3 = \log 127$ 附近的某点 (a, b) . 根据 n 和 z 的大小关系, 只需要乘以 3 或者除以 2 来更新 z 值, 并判断 $|n - z|$ 是否应当被暂存下来. 图 1 中折线上的点按照从右下到左上的顺序就是 2-Line Search 方法的搜索轨迹. 与 Line Search 方法不同的是, 文中将折线上的点分成两个互不相交的集合 $I_1: \{(a, b) | 2^a 3^b \geq n\}$ 和 $I_2: \{(a, b) | 2^a 3^b < n\}$, 在两个集合内分别寻找目标 n 的最佳近似值, 并将对应的 $|n - z|$ 作为下一轮目标的候选.

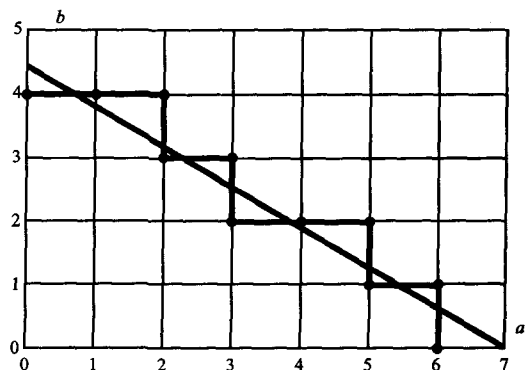


图 1 2-Line Search 方法的搜索轨迹

算法 1 2-Line Search

输入: 正整数 n ; $a, b > 0$, 分别对应基数 2 和 3 的指数上界; 属于集合 $\{\pm 1\}$ 的系数 c .

输出: 4 个 1×2 数组 Num, Ea, Eb 和 Co , 用来存储集合 I_1 和 I_2 内的最佳近似值及其对应的指数和系数.

1. $Num[0] \leftarrow n; Num[1] \leftarrow n$
2. $i \leftarrow a; j \leftarrow 0$
3. $z \leftarrow 2^a$
4. while $z < n$ do
5. $z \leftarrow z * 3$
6. $j \leftarrow j + 1$
7. if $z > n$ and $j > 0$ then
8. $z \leftarrow z / 3$
9. $j \leftarrow j - 1$
10. while $i \geq 0$ do
11. if $z \geq n$ then $k \leftarrow 0$
12. else $k \leftarrow 1$
13. if $|n - z| < Num[k]$ then
14. $Num[k] \leftarrow |n - z|$
15. $Ea[k] \leftarrow i$
16. $Eb[k] \leftarrow j$
17. $Co[k] \leftarrow c * (2k - 1)$

```

18. if  $z < n$  then
19.  $z \leftarrow z * 3$ 
20.  $j \leftarrow j + 1$ 
21. if  $j > b$  then break
22. else
23.  $z \leftarrow z / 2$ 
24.  $i \leftarrow i - 1$ 
25. return  $Num, Ea, Eb, Co$ .

```

已知参数 B , $(2, B)$ -Greedy 算法的执行过程如下: 在 $(2, B)$ -Greedy 算法的每一层, 2-Line Search 方法将针对 B 个不同的目标值各运行一次 (除了第一步仅有 1 个目标外); 然后 $(2, B)$ -Greedy 算法选择最小的 B 个候选值作为下一步的目标. 算法将重复上述操作直到目标值中出现 0, 便可得到初始输入 N 的 DBC 表达式. 需要强调的是 $(2, B)$ -Greedy 算法每一层的目标值都应该是两两不同的.

算法 2 $(2, B)$ -Greedy 算法

输入: 正整数 N ; $a_{\max}$ 和 $b_{\max}$, 初始指数上界; 界 B .

输出: 序列 $(d_i; a_i; b_i)$ 满足 $N = \sum_{i=1}^l d_i 2^{a_i} 3^{b_i}$, $d_i \in \{\pm 1\}$, 其中 $a_1 \geq a_2 \geq \dots \geq a_l \geq 0$ 且 $b_1 \geq b_2 \geq \dots \geq b_l \geq 0$.

```

1. for  $i = 0, \dots, B - 1$  do
2.  $Num[i] \leftarrow N$ 
3.  $Ema[i] \leftarrow a_{\max}$ 
4.  $Emb[i] \leftarrow b_{\max}$ 
5.  $Co[i] \leftarrow 1$ 
6. while  $Num[0] > 0$  do
7. for  $i = 0, \dots, B - 1$  do
8. 运行 2-Line Search 并将  $Num[i]$ ,
    $Ema[i]$ ,  $Emb[i]$ ,  $Co[i]$  作为输入
9. 保存最优的  $B$  个候选值并将对应的结果存入
    $Num$ ,  $Ema$ ,  $Emb$ ,  $Co$ 
10. return  $(d_i; a_i; b_i)$ .

```

例 3 设 $n = 841232$, $B = 2$, 由 $(2, 2)$ -Greedy 算法求出的 DBC 展开式为

$$841232 = 2^7 3^8 + 2^4 3^4 + 2^4 3^2 - 2^4$$

尽管文献[12]已经给出了 DBNS 表达式的项数上界, 但是该结论是否同样适用于 DBC 表达式仍是一个开放问题. 因为任意整数 n 的 DBC 长度不小于其 DBNS 表达式, 且由于 2-Line Search 方法中指数上界的限定和分支结构的出现, 对于 $(2, B)$ -

Greedy 算法的效率分析非常困难. 因此, 将在下一节通过实验和数据来间接地加以分析.

4 实验和结果

测试平台为 Intel(R) Core(TM)2 Duo CPU E7200 2.53GHz, RAM 1.96GB; 操作系统是 32 位 Microsoft Windows XP professional 2002 SP3; 编译器 Visual Studio C++ 6.0; 大整数运算函数库 miracl 5.3.3.

首先, 分析了 $(2, B)$ -Greedy 算法中参数 B 对 DBC 表达式平均长度的影响. 针对不同的比特长度和 B 值, 各选取 1000 个随机整数进行新算法的测试, 见图 2. 实验表明, 在所有的比特长度下, 随着 B 的增大 DBC 表达式的平均长度是递减的. 例如, 和 $(2, 1)$ -Greedy 算法相比, 使用 $(2, 4)$ -Greedy 算法可以将平均长度减少大约 8%; 如果使用 $(2, 128)$ -Greedy 算法则能将平均长度减少 15% 以上. 但是从图 2 中可以看出这种减小的趋势随着 B 的增大而逐步放缓, 当 B 大于 64 的时候, 线条趋于水平. 与此同时算法的平均运行时间也受到参数 B 的影响: 随着 B 的增大而迅速增加, 见图 3. 为了保证算法求出的 DBC 表达式是相对较短的, 同时又能有效限定算法的运行时间, 可以设置 $B = 4, 8, 16$. 本节的实验数据除特别说明外均来自于 $(2, 4)$ -Greedy 算法.

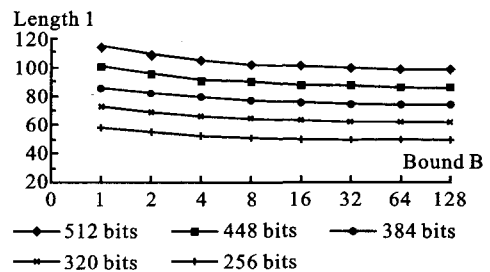


图 2 参数 B 对 DBC 平均长度的影响

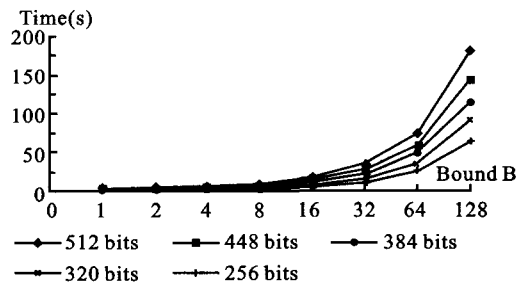


图 3 参数 B 对运行时间的影响 (1000 个随机数)

计算点乘 $[k]P$ 的复杂度不仅取决于 DBC 表达式的长度, 同时也受到表达式首项对应指数 a_1 和 b_1

的影响. a_1 和 b_1 分别等于该表达式在计算点乘时所必需的二倍点和三倍点运算的次数. 文中为每个比特长度各选择了 1000 个随机整数来测试不同算法在这三个参数上的表现, 见表 1. 显而易见, B 值的

变化所带来的影响只作用在展开式的长度 l 上, 对 a_1, b_1 并不起作用; 由于双基表示的引入, $(2, B)$ -Greedy 算法返回的表达式长度相比 NAF 表示大大降低.

表 1 不同方法求解的 DBC 表达式的参数

Size(bits)	256			320			384			448			512		
Method	l	a_1	b_1	l	a_1	b_1	l	a_1	b_1	l	a_1	b_1	l	a_1	b_1
NAF	85.3	256	0	106.7	320	0	128.0	384	0	149.3	448	0	170.7	512	0
(2,1)-Greedy	57.9	149.4	67.0	72.1	187.0	83.6	86.4	221.6	102.2	100.7	260.9	117.8	114.9	298.8	134.2
(2,4)-Greedy	52.8	148.2	67.7	66.2	187.0	83.6	79.2	224.1	100.6	92.1	259.3	118.8	105.3	298.0	134.8

文中选取 Inverted Edwards 坐标上的椭圆曲线, 通过定义在其上的标量乘法来测试不同算法的平均代价. 在该类椭圆曲线中一个加法需要 $9M + S$ (M 表示域上的乘法, S 表示平方), 一个二倍点需要 $3M + 4S$, 一个三倍点需要 $9M + 4S$. 同时为了简化比较, 不失一般性地假设 $S = 0.8M$. 在表 2 中, N_M

表示经过换算后计算点乘 $[k]P$ 所需的乘法数量; Gain 表示算法相对于 NAF 表示所带来的收益. 从表 2 可以看出 $(2, 4)$ -Greedy 算法求出的表达式相比 NAF, 其计算代价降低了 6.6% 以上. 即使和 $(2, 1)$ -Greedy 算法比较, $(2, 4)$ -Greedy 也有至少 2% 的提升.

表 2 不同算法的计算点乘 $[k]P$ 的代价

Size(bits)	256		320		384		448		512	
Method	N_M	Gain	N_M	Gain	N_M	Gain	N_M	Gain	N_M	Gain
NAF	2423.14	—	3029.66	—	3635.20	—	4240.74	—	4847.26	—
(2, 1)-Greedy	2311.10	4.62%	2885.90	4.75%	3467.48	4.61%	4041.60	4.70%	4615.82	4.77%
(2, 4)-Greedy	2262.22	6.64%	2828.08	6.65%	3392.90	6.67%	3959.60	6.63%	4524.10	6.67%

最后, 文中分析了 $a_{\max}$ 和 $b_{\max}$ 的选取对 DBC 表达式的长度及性能的影响 (见表 3). 通过在不同的比特长度, 各自 1000 个随机整数的测试, 发现当 $a_{\max} = \lceil 2\log_2 n / 3 \rceil$, $b_{\max} = \lceil \log_3 n / 2 \rceil$ 时, $(2, 4)$ -

Greedy 算法的结果无论在表达式长度还是标量乘法的代价等方面都是 5 种参考取值里最优的, 建议在实际应用中选取. 其中 $a_{\max} = x * \log_2 n$, $b_{\max} = y * \log_3 n$.

表 3 $(2, 4)$ -Greedy 算法在不同的 $a_{\max}$ 和 $b_{\max}$ 下的表现

Size(bits)	256		320		384		448		512	
(x, y)	N_M	l	N_M	l	N_M	l	N_M	l	N_M	l
(1/2, 1/2)	2301.16	53.5	2878.32	67	3448.54	79.8	4020.20	92.8	4590.24	105.7
(1/2, 2/3)	2382.92	58.7	2986.50	73.9	3586.16	89.0	4193.80	104.8	4789.46	119.6
(2/3, 1/2)	2260.32	52.6	2827.70	66.1	3397.16	79.4	3961.00	92.6	4524.24	105.5
(2/3, 1/3)	2275.12	57.5	2845.62	71.9	3417.18	86.4	3992.68	101.4	4567.66	116.4
(1/3, 2/3)	2502.60	67.7	3137.50	85.4	3774.10	103.4	4410.02	121.2	5043.72	138.9

5 结束语

通过引入树形结构, 文中将普通的贪心算法扩展成了 $(2, B)$ -Greedy 算法. 实验证明, 这种改进加速了贪心算法的收敛, 并有效降低了椭圆曲线上标量乘法的计算代价. 通过实验数据, 建议选取 $B = 4, 8, 16$ 以在算法收益和时间代价之间取得良好的平衡. 下一步将主要研究新算法在扩展 DBNS 中的应用.

参考文献:

- [1] 傅鹏, 程艳, 陈承源. 基于椭圆曲线的 XML 数字签名研究与实现[J]. 微电子学与计算机, 2006, 23(9): 144 - 146.
- [2] 肖立国, 钟诚, 陈国良. 基于椭圆曲线密码体制的动态秘密共享方案[J]. 微电子学与计算机, 2002, 19(1): 30 - 31.

(下转第 184 页)

4 结束语

融合能量谱融合可有效避免单通道数据的信息不完整、不精确和不准确等不足. 文中通过对转子试验台的典型故障实验, 并对比分析了基于单通道数据和融合数据的诊断结论. 结果表明, 将融合能量谱应用于旋转机械故障诊断, 能提高故障诊断的准确率, 具有非常重要的工程应用价值.

参考文献:

- [1] 韩捷, 石来德. 旋转机械的全信息能量谱分析方法研究[J]. 机械强度, 2003, 25(4): 364 - 368.
- [2] Qu L S, Chen Y D, Liu X. The holo-spectrum: a new method for rotor surveillance and diagnosis[J]. Mechanical System and Signal Processing, 1989, 3(3): 255 - 267.
- [3] Agnes Muszynska, Donald Bentley, Alex Pechiney. Vibration response of rotor supported in one rigid and one poorly lubricated fluid film bring[J]. ORBIT, 1996, 17(3): 7 -

9.

- [4] 韩捷, 石来德, 董辛旻. 旋转机械虚拟探头融合检测技术研究[J]. 中国工程机械学报, 2007(2): 204 - 207.
- [5] 符向前, 刘光临, 蒋劲. BP 神经网络在水轮发电机组状态监测与诊断系统中的应用[J]. 武汉大学学报: 工学版, 2002, 35(1): 24 - 28.
- [6] 张斌, 严骏, 张兴. BP 神经网络在挖掘机液压系统故障诊断中的应用[J]. 机床与液压, 2008, 36(2): 194 - 195.
- [7] 韩捷, 张瑞林. 旋转机械故障机理及诊断技术[M]. 北京: 机械工业出版社, 1991.

作者简介:

董辛旻 男, (1968 -), 博士研究生. 研究方向为机械设备信息融合及故障诊断.

韩捷 男, (1957 -), 教授, 博士生导师. 研究方向为机械设备信息融合及故障诊断等.

石来德 男, (1938 -), 教授, 博士生导师. 研究方向为机械设备信息融合及故障诊断等.

(上接第 180 页)

- [3] Hankerson D, Menezes A, Vanstone S. Guide to elliptic curve cryptography[M]. New York: Springer - Verlag, 2003.
- [4] Solinas J. Efficient arithmetic on koblitz curves[C]// Designs, Codes and Cryptography. Netherlands, Springer, 2000; 19(3): 195 - 249.
- [5] Okeya K, Samoa K S, Spahn C, et al. Signed binary representations revisited[C]// CRYPTO 2004. Heidelberg, Springer, 2004(3152): 123 - 139.
- [6] Avanzi R, Dimitrov V, Doche C, et al. Extending scalar multiplication using double bases[C]// Advances in Cryptology ASIACRYPT '06. Heidelberg, Springer, 2006 (4284): 130 - 144.
- [7] Bernstein D J, Lange T. Inverted edwards coordinates [C]// AAECC 2007. Heidelberg, Springer, 2007 (4851): 20 - 27.
- [8] Yu Wei, Wang Kunpeng, Li Bao. Fast algorithm converting integer to double base chain[C]// ChinaCrypt' 2008. China: Chendu, 2008: 355 - 366.
- [9] Doche C, Habsieger L. A tree-based approach for com-

puting double-base chains[C]// ACISP 2008. Heidelberg, Springer, 2008(5107): 433 - 446.

- [10] Dimitrov V S, Cooklev T. Hybrid algorithm for the computation of the matrix polynomial $I + A + \dots + AN - 1$ [J]. IEEE Trans. on Circuits and Systems, 1995, 42(7): 377 - 380.
- [11] Dimitrov V S, Imbert L, Mishra P K. Efficient and secure elliptic curve point multiplication using double-base chains[C]// ASIACRYPT 2005. Heidelberg, Springer, 2005(3788): 59 - 78.
- [12] Dimitrov V S, Jullien G A, Miller W C. An algorithm for modular exponentiation[J]. Information Processing Letters, 1998, 66(3): 155 - 159.

作者简介:

李明 男, (1983 -), 硕士研究生. 研究方向为椭圆曲线密码学.

王鲲鹏 男, (1971 -), 副教授. 研究方向为密码理论与技术.