

一个新的基于 MOF 从左到右编码的多标量乘算法

程一飞

(安庆师范学院 计算机系, 安徽 安庆 246011)

摘要:很多基于椭圆曲线的密码协议都需要计算多标量乘法 $kP + lQ$ 。目前常见的多标量乘算法的效率主要取决于标量的(联合)海明权值。JSF 表示的平均联合海明权密度为 $1/2$, 是所有带符号二进制表示中最优的, 但 JSF 编码只能从右到左实现。提出一个新的从左到右的基于 MOF 的编码方法, 该方法的平均联合海明权密度与基于 JSF 表示的相同, 并提出一个新的多标量乘算法, 该算法对标量从左到右进行编码, 并将编码合并到多标量乘的主计算中, 从而节省了存储标量的新编码的内存空间, 提高了实现效率。

关键词:椭圆曲线密码系统; 标量乘; 多标量乘; MOF; JSF

中图分类号: TP309.7

文献标识码: A

文章编号: 1673-629X(2007)11-0157-03

A New Mutual Opposite Form Based Left - to - Right Multi - Scalar Multiplication Algorithm

CHENG Yi-fei

(Computer Science Dept. of Anqing Teachers College, Anqing 246011, China)

Abstract: Many elliptic curve based cryptographic protocols require computation of multiple scalar multiplications such as $kP + lQ$. Common methods to compute it are the Shamir method and the interleaving method whereas their speed mainly depends on the (joint) Hamming weight of the scalars. The joint sparse form of two L -bit integers has an average joint Hamming weight of $L/2$, which is an optimal-weight signed-binary representation, but it can be implemented only from right to left. In this paper, a new recoding method based on the mutual opposite form is proposed. This form has the same average Joint Hamming weight as the JSF. And a new MOF representation based multiple scalar multiplication algorithm is given. This method examines the integers from left to right. This results in the merging of recoding and evaluation stages. So the proposed algorithm can improve the performance and reduce the memory consumption of scalar multiplication operation.

Key words: elliptic curve cryptography; scalar multiplication; multi-scalar multiplication; mutual opposite form; joint sparse form

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。很多基于椭圆曲线的密码协议

比如 ECDSA 签名验证, 都需要计算多标量乘法 $kP + lQ$ 。目前常见的多标量乘算法有: Shamir 多标量乘算法, interleaving 多标量乘算法等, 它们的效率主要取决于标量的(联合)海明权值。JSF 表示的平均联合海明权密度为 $1/2$, 是所有带符号二进制表示中最优的, 但 JSF 编码只能从右到左(即从最低位到最高位)实现。

提出一个新的从左到右的基于 MOF 的编码方法, 该编码方法的平均联合海明权密度与基于 JSF 表示的相同, 并提出一个新的多标量乘算法, 该算法对标量从左到右进行编码, 并将编码合并到多标量乘的主计算中, 从而节省了存储标量的新编码的内存空间, 并提高了实现效率。

1 相关概念

在有限域 $K = F_2^n$ 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 $a, b \in$

收稿日期: 2007-01-11

基金项目: 安徽省教育厅自然科学基金项目(2006KJ079B)

作者简介: 程一飞(1976-), 男, 安徽怀宁人, 硕士, 讲师, 研究方向为计算机密码学。

F_2^n , 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^n 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。

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

多点乘(多标量乘): 形如 $k_1P_1 + k_2P_2 + \dots + k_tP_t$, 其中 k_i 是整数, P_i 是椭圆曲线上的点。可以理解为 t 个点乘所得到的点相加, 称为多点乘或多标量乘。

2 新的从左到右的基于 MOF 编码

定义 1: 设 d 是一个整数, 以 $a_{n-1} \dots a_1 a_0$ 表示, 即 $k = a_{n-1}2^{n-1} + \dots + a_12^1 + a_02^0$, 则称 $a_{n-1} \dots a_1 a_0$ 为 d 的 radix-2 表示, 通常写成 $(a_{n-1} \dots a_1 a_0)_2$ 。

定义 2: T 是整数的集合, $d = \sum d_i 2^i$, 若任意 $d_i \in T \cup \{0\}$, 则称 d 为 T 表示。

若 T 中包含有负数, 则称 d 为带符号的 T 表示; 若 $T = \{\pm 1\}$, 则称 k 为带符号的二元表示。

在 Shamir 算法中, 点加运算的次数由标量的联合海明权值决定, 倍点运算的次数由整数的长度决定。因此减小标量的联合海明权值可以提高标量乘运算的效率。Solinas^[3] 提出了一个对整数从右到左编码一对整数的 JSF (Joint Sparse Form), 并证明 JSF 表示唯一且最优。 L 位的整数的 JSF 表示的联合海明权值为 $L/2$, 但 JSF 表示只能从右向左进行。这就意味着需要先计算标量的 JSF 表示并存储, 然后才能进行标量乘运算。 Okeya^[4] 等人提出了一个新的算法 MOF, 同时 Avanzi^[5]、Heuberger et al^[6]、Muir 和 Stinson^[7] 等人也提出同样的表示形式, 只是以不同的形式描述。在文中, 姑且以 MOF 来表示。

定义 3: n 位 mutual opposite form (MOF) 是一个 n 位带符号的满足以下二个属性的二元串。

(1) 相邻的非零数(不考虑零)的符号互反;

(2) 最高非零位为 1, 最低非零位为 -1, 除非所有位都为 0。

因为 $2^x = 2^{x+1} - 2^x$, 所以通过 $\mu = 2d \ominus d$, 可以将 n 位的二元串转换成 $(n+1)$ 位的 MOF 串, 其中 \ominus 为位减。如图 1 所示。

$$\begin{array}{ccccccccccc} 2d = & d_{n-1} & & d_{n-2} & \cdots & d_{i-1} & & \cdots & d_1 & & d_0 & & 0 \\ \ominus d = & 0 & & d_{n-1} & \cdots & d_i & & \cdots & d_2 & & d_1 & & d_0 \\ \hline \mu = & d_{n-1} & & d_{n-2} - d_{n-1} & \cdots & d_{i-1} - d_i & & \cdots & d_2 - d_1 & & d_2 - d_0 & & -d_0 \end{array}$$

图 1 MOF 编码图

算法 1: 二元表示的正整数转换为带符号表示二元 MOF 表示

输入: n 位二元表示 $(a_{n-1} \dots a_1 a_0)_2$, 其中 $a_i \in \{0, 1\}$

输出: $n+1$ 位二元表示 $(\mu_n \dots \mu_1 \mu_0)_2$, 其中 $\mu_i \in \{0, \pm 1\}$

1) $a_{-1} = 0; a_n = 0;$

2) for i from n downto 1 step -1 do

$\mu_i = a_{i-1} - a_i$

3) return $(\mu_n \dots \mu_1 \mu_0)_2$

由 MOF 表示的定义可知, 相邻的两位 $\mu_i \mu_{i-1}$ 可能是 $00, 0x, x0, \bar{x}\bar{x}$ 。因为 $\bar{x}\bar{x} = 2x - x = x$, 所以 $\bar{x}\bar{x}$ 可以转换成 $0x$, 减小非零数的个数。其中 $x \in \{\pm 1\}$, $\bar{x} = -x$ 。而对于一对整数, $\mu_0, \mu_1, \mu_2, \dots, \mu_{i-1}, \mu_i, \mu_{i+1}, \mu_{i+2}, \dots$ 中某一个为 $x0\bar{x}$ 的形式, 另一个不为 $y0z$ 的形式, 都可以将 $x0\bar{x}$ 转换成 $0xx$ 的形式, 另一个不变, 从而减小非零数的个数。 $x, y \in \{\pm 1\}, \bar{x} = -x, z \in \{0, \pm 1\}$ 。具体算法见算法 2。

算法 2: 一对二元表示的正整数转换为带符号表示二元表示

输入: 一对 n 位二元表示 $(a_{0,n-1} \dots a_{0,1} a_{0,0})_2$ 和 $(a_{1,n-1} \dots a_{1,1} a_{1,0})_2$, 其中 $a_{i,j} \in \{0, 1\}$

输出: $n+1$ 位二元表示 $(\mu_{0,n} \dots \mu_{0,1} \mu_{0,0})_2$

$(\mu_{1,n} \dots \mu_{1,1} \mu_{1,0})_2$, 其中 $\mu_{i,j} \in \{0, \pm 1\}$

1) $a_{0,-1} = 0; a_{0,n} = 0; a_{1,-1} = 0; a_{1,n} = 0;$

2) $j = n; \text{temp}_{0,0} = \text{temp}_{0,1} = \text{temp}_{1,0} = \text{temp}_{1,1} = 0;$

3) while $j > 1$

{ for i from 0 to 1 do

$u_{i,j} = a_{i,j-1} - a_{i,j};$

$u_{i,j-1} = a_{i,j-2} - a_{i,j-1};$

$u_{i,j-2} = a_{i,j-3} - a_{i,j-2};$

if $(j < n \text{ and } \text{temp}_{i,0} \neq u_{i,j}) u_{i,j} = \text{temp}_{i,0};$

if $(j < n \text{ and } \text{temp}_{i,1} \neq u_{i,j-1}) u_{i,j-1} = \text{temp}_{i,1};$

if $u_{i,j} \mu_{i,j-1} = -1$ then $\{ u_{i,j-1} = u_{i,j}; u_{i,j} = 0; \text{temp}_{i,0} = u_{i,j-1}; \text{temp}_{i,1} = u_{i,j-2}; \}$ // $x\bar{x}$ 转换成 $0x$

else if $((u_{i,j} u_{i,j-2} = -1 \text{ and } u_{i,j-1} = 0)) \{$

if $\{ (a_{1-i,j-1} \neq a_{1-i,j} \text{ and } a_{1-i,j-1} = a_{1-i,j-2}) \text{ then } \{ u_{i,j-2} = u_{i,j-1}; u_{i,j} = 0; \text{temp}_{i,0} = u_{i,j-1}; \text{temp}_{i,1} = u_{i,j-2}; \}$

// $x0\bar{x}$ 转换成 $0xx$

$j = j - 1;$

```

}
4) if (j = 1)
    {for i from 0 to 1 do
         $u_{i,j} = a_{i,j-1} - a_{i,j};$ 
         $u_{i,j-1} = a_{i,j-2} - a_{i,j-1};$ 
        if  $u_{i,j} u_{i,j-1} = -1$  then  $\{u_{i,j-1} = u_{i,j}; u_{i,j} = 0;\}$ 
    }
5) return( $\mu_{0,n} \cdots \mu_{0,1} \mu_{0,0}$ )2
    ( $\mu_{1,n} \cdots \mu_{1,1} \mu_{1,0}$ )2

```

算法 2 的思路每次循环对三列进行处理,若有 $\bar{x}x$ 形式,将其转换成 $0x$,若一列为 $x0\bar{x}$,且对应另一列不为 $y0z$ 形式,则将 $x0\bar{x}$ 转换成 $0xx$,增加零列的个数,从而减小海明权值。循环变量每循环一次减一,即实际上每次仅处理一列。

3 新的基于 MOF 表示的标量乘法

3.1 算法

算法 3: 基于改进 MOF 的多标量乘法

输入: 一对 n 位二元表示 $a: (a_{0,n-1} \cdots a_{0,1} a_{0,0})_2$ 和 $b: (a_{1,n-1} \cdots a_{1,1} a_{1,0})_2$, 其中 $a_{i,j} \in \{0,1\}$, P 点, Q 点。

输出: $aP + bQ$

预计算:

计算 $P + Q, P - Q$;

主计算:

```

1)  $R = O$  (无穷远点);
2)  $a_{0,-1} = 0; a_{1,-1} = 0$ ;
3)  $j = n; u_{0,0} = a_{0,n-1}; u_{0,1} = a_{0,n-2} - a_{0,n-1}; u_{1,0} = a_{1,n-1};$ 
    $u_{1,1} = a_{1,n-2} - a_{1,n-1};$ 
4) while  $j > 1$ 
    { for i from 0 to 1 do
         $u_{i,2} = a_{i,j-3} - a_{i,j-2};$ 
        if  $u_{i,0} u_{i,1} = -1$  then  $\{u_{i,1} = u_{i,0}; u_{i,0} = 0;\}$  //  $\bar{x}x$  转换成  $0x$ 
            else if  $(u_{i,2} u_{i,0} = -1 \text{ and } u_{i,1} = 0)$ 
                if  $!(a_{1-i,j-1} \neq a_{1-i,j} \text{ and } a_{1-i,j-1} = a_{1-i,j-2})$  then  $\{u_{i,2} = u_{i,1}$ 
                     $= u_{i,0}; u_{i,0} = 0;\}$ 
                //  $x0\bar{x}$  转换  $0xx$ 
         $R = 2R + u_{0,0}P + u_{1,0}Q;$ 
         $j = j - 1;$ 
    } for i from 0 to 1 do
         $\{u_{i,0} = u_{i,1};$ 
         $u_{i,1} = u_{i,2};\}$ 
    }
5) if ( $j = 1$ ) {for i from 0 to 1 do
     $u_{i,0} = a_{i,1} - a_{i,0};$ 
     $u_{i,1} = -a_{i,0};$ 
    if  $u_{i,0} u_{i,1} = -1$  then  $\{u_{i,1} = u_{i,0}; u_{i,0} = 0;\}$ 
     $R = 2R + u_{0,0}P + u_{1,0}Q;$ 

```

$R = 2R + u_{0,1}P + u_{1,1}Q;$

6) return R ;

3.2 算法分析

算法 3 共需进行 $n + 1$ 次循环,且算法 2 中 $R = 2R + tP + sQ$ 运算可以分解为 $R = 2R; R = R + tP + sQ$ 。因此共需进行 $n + 1$ 次倍点运算 $R = R + tP + sQ$ 。表面上看每执行一次循环就要进行一次,即要进行二次点加运算,但实际上 t, s 有可能同为 0,而当 t, s 同为 0 是不进行点加运算的,而且由于预先进行了预计算 $P + Q, P - Q$,所以即使 t, s 同时不为 0,也仅需进行一次点加运算。因为 $t, s \in \{0, \pm 1\}$ 。由实验可知非零列约为 $1/2$,也即 n 位的整数平均海明权值约为 $n/2$ 。

4 结论

多点乘算法不仅在 ECDSA 签名验证等密码协议中应用,还可以通过同构等方法将单点乘运算转换成多点乘运算,因此也可以运用于点乘运算中。文中提出了一个新的改进的 MOF 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,并以此为基础提出了一个新的基于 Shamir 算法的多标量乘法。在新的标量乘法中,由于编码方法是从左到右进行,主计算阶段合并,这样可以节省内存空间(JSF 表示需要预先于主计算计算并存储,这样每个 L 位的整数就需要 $L + 1$ 位带符号的二进制位来存储,而文中的算法仅需要临时存储三位带符号的二进制位)。实验结果表明,该算法的效率能够达到 Solinas 提出的 JSF 多标量乘法效率。因此非常适用内存空间受限制的设备,如 Smart 卡等。

参考文献:

- [1] Koblitz N. Elliptic Curve Cryptosystems[J]. Math Computation, 1987, 48: 203 - 209.
- [2] Miller V S. Uses of Elliptic Curve in Cryptography[J]. Springer - Verlag Lecture Notes in Computer Science, 1987, 218: 417 - 426.
- [3] Solinas J A. Low - Weight Binary Representations for Pairs of Integers[R]. Canada: Center for Applied Cryptographic Research, 2001.
- [4] Okeya K, Schmidt - Samoa K, Spahn C, et al. Signed Binary Representations Revisited[C]// In Advances in Cryptology - CRYPTO 2004, LNCS. [s. l.]: Springer - Verlag, 2004: 123 - 139.
- [5] Avanzi R M. A Note on the Signed Sliding Window Integer Recoding and a Left - to - Right Analogue[C]// In Selected

(下转第 163 页)

$$\text{个人组个人生产率: } \frac{1}{100\%} = 1$$

$$\text{结对组个人生产率: } \frac{\frac{1}{60\%}}{2} = 0.83$$

可以看出,结对组的生产率还有很大的改进空间,虽然随着结对编程的实践深入结对者的生产率会进一步提升,但仍然要寻求一定的改进。

任何事物都有两面性,结对编程设定两人共用一台电脑工作、编程,造成资源的使用冲突,这是发现的一个造成个人生产率下降的主要原因。图 3 显示了一个典型的编程工作区的改进和安排示意图。

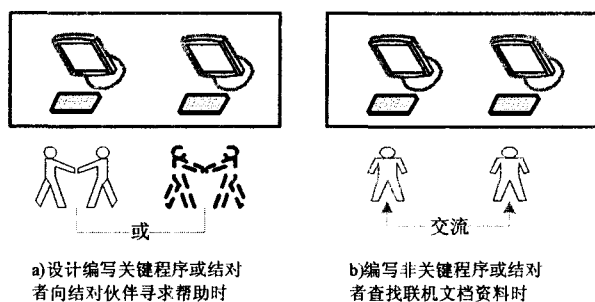


图 3 结对编程工作区改进示意图

3.2 重构二:结对编程扩展性改进

在某些情况下,尤其是在大型项目中,一个模块可能需要两人以上的开发成员来实现,而结对编程只强调了两个开发人员要结为一对。这时引入一个重构:将两人结为一对的强结对,根据情况转变为松结对,即 2~5 人组成一组,结对时根据情况寻找结对伙伴并不断交换。由松结对组成的团队称为松结对团队,如图 4 所示。一个基于结对编程组成的松结对团队必须具备以下两个关键特征:

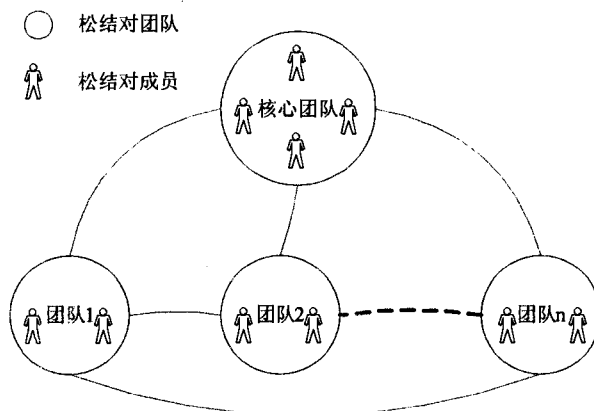
1) 一个松结对团队中编写的代码对该松结对团队所有成员透明,即代码松结对团队集体所有。

2) 松结对团队是一个利益责任共同体,所有的责任和利益在整个松结对团队中人人平等。

4 结论和下一步工作

结对编程由于注重团队合作的建立,在近十年中逐渐获得了新兴的敏捷组织和团队的欢迎。结对编程不仅提高了软件的可靠性,同时也获得了较高的工作

效率,并且有利于学习型组织的建立。在软件开发中充分实践结对编程,并结合自身的条件作一定的重构,每个企业都可以获得理想的结对编程环境,进而提升自身在现代软件开发业中的竞争力。



未来的主要研究工作是,如何让结对编程适应各种过程不同、团队大小不同、地域分布集中或分散的开发环境,并全面分析结对编程在各种环境中的作用和这些环境因素反过来对结对编程的影响。另一项重要任务是构建一定的工具和平台来支撑各种基于结对编程的软件开发工作。此外,可以在学校里开展学生结对编程的教学和科研活动,收集相关的数据,作进一步的研究工作。

参考文献:

- [1] 徐琛,杨宗源. 轻载软件开发方法[J]. 计算机工程, 2003,29(1):268-271.
- [2] Beck K. Extreme Programming Explained: Embrace Change [M]. 2nd ed. Boston: Addison - Wesley, 2004.
- [3] Cockburn A, Williams L. The Costs and Benefits of Pair Programming [C]//First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000). Italy: [s. n.], 2000.
- [4] Williams L, Kessler R, Cunningham W, et al. Strengthening the Case for Pair Programming[J]. IEEE Software, 2000, 17(4):19-25.
- [5] Fowler M. Refactoring: Improving the Design of Existing Code [M]. Boston: Addison - Wesley, 1999.

(上接第 159 页)

- Areas in Cryptography - SAC 2004, LNCS. [s. l.]: Springer - Verlag, 2004:130-143.
- [6] Heuberger C, Karti R, Proding H, et al. The Alternating Greedy Expansion and Applications to Left - to - Right Algorithms in Cryptography[J]. Theoretical Computer Science A,

2005, 341:55-72.

- [7] Muir J A, Stinson D R. New Minimal Weight Representations for Left - to - Right Window Methods [C]// In Proc. Cryptographers Track - RSA Conf (CT - RSA 2005), LNCS. [s. l.]: Springer - Verlag, 2005:366-383.