椭圆曲线标量乘法中标量的有效表示

李 忠[1,2], 彭代渊[1]

(1. 西南交通大学信息科学与技术学院, 四川 成都 610031)
(2. 宜宾学院计算机与信息工程学院, 四川 宜宾 644000)

[摘要] 标量乘法是椭圆曲线密码(ECC)的基本运算, 也是最耗时的运算, 其运算效率直接决定着 ECC的性能. 在标量乘法运算中, 标量 k的表示起着至关重要的作用, 其长度决定了所需倍点运算量, 其汉明重量决定了所需点加运算量. 本文提出了一种新的标量表示方法, 与目前流行的方法相比, 该表示方法具有编码方式简单, 汉明重量轻等优点. 使用新的标量表示方法, 能有效提高 ECC的实现效率, 尤其对于 $\{10\}^m$ 及 $\{10\}^m \| 1$型的标量, 效率提高明显.

[关键词] 椭圆曲线密码, 标量乘法, 标量表示, 汉明重量
[中图分类号] TP309.7 [文献标识码] A [文章编号] 1001-4616(2010)03-0135-06

Efficient Scalar Representation in *kP* of Elliptic Curve Cryptosystems

LiZhong[1,2], Peng Daiyuan[1]

(1. School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China)
(2. School of Computer & Information Engineering, Yibin University, Yibin 644000, China)

Abstract Scalar multiplication is the fundamental and time-consuming operation in elliptic curve cryptosystems; the performance of the elliptic curve cryptosysytem deeply depends on the efficiency of scalar multiplication. In scalar multiplication, the scalar k's representation plays an important role; it's length decide the number of point addition operations; it's hamming weight decide the number of point double operations. In this paper, a new scalar representation method was presented, it can reduce the hamming weight efficiently. The analysis results show that the new method is more efficient than existing scalar representation methods; and is particularly useful for the scalar such as $\{10\}^m$ and $\{10\}^m \| 1$.

Key words elliptic curve cryptosysytem; scalar multiplication; scalar representation; hamming weight

1985年, Neal Koblitz和 Victor Miller分别独立地提出了椭圆曲线密码体制(ECC). 椭圆曲线密码体制的安全性依赖于椭圆曲线离散对数问题(ECDLP), 它具有安全性高、密钥尺度小、灵活性好等特点. 在目前已知的公钥体制中, 椭圆曲线密码体制是对每一比特所提供加密强度最高的一种体制, 被认为是最值得关注的密码体制, 极有可能成为现存公钥密码体制的替代者[1].

在 ECC中, 标量乘法是最耗时的运算, 其运算效率直接决定着 ECC的性能, 如何高效实现标量乘法运算, 一直是 ECC研究的一个重点. 许多研究人员在这方面做了大量的工作, 取得了大量的成果. 目前对标量乘法的研究主要还是基于"倍-加"运算的. 在这些算法中, 标量 k的表示起着至关重要的作用, 其长度决定了所需倍点运算量, 其汉明重量(表示中非零数字个数)决定了点加运算量. 在不同的坐标系下, 点加运算和倍点运算的时间消耗差别是较大的, 点加运算的时间消耗可能为倍点运算的 2~3倍[2], 因此, 如何减少标量表示的汉明重量, 成为提高标量乘法运算效率的关键之一. 本文在分析目前几种典型标量表示方法的基础上, 提出了一种新的标量表示方法(下文称其为 LZNAF表示法), 与目前流行的表示方法相比, 该方法具有编码方式简单, 能有效地减少表示中的汉明重量, 尤其对于 $\{10\}^m$ 及 $\{10\}^m \| 1$型的标量, 效率提高明显.

收稿日期: 2010-06-10
基金项目: 四川省教育厅重点科研项目(07ZA145).
通讯联系人: 李 忠, 博士生, 副教授, 研究方向: 密码学、信息安全. E-mail: lz8056859@163.com

1　基于“倍 – 加”运算的标量乘法的一般模型

对于基域 K 上的非超奇异椭圆曲线 E, $k \in K$, $P = (x, y) \in E(K)$, 称 $kP = \underbrace{P + \cdots + P}_{k}$ 为椭圆曲线 E 上的标量乘法运算. 若基域 K 的特征 $\mathrm{char}(K) = 2$, 则 $-P = (x, x + y)$, 若 $\mathrm{char}(K) > 3$, 则 $-P = (x, -y)$, 所以, 椭圆曲线点的减法与点的加法运算同样有效. 令

$$k = \sum_{i=0}^{l-1} k_i 2^i, \tag{1}$$

其中, $k_i \in D_s$, $0 \leqslant i \leqslant l-1$, $k_{l-1} \neq 0$, 称(1)式为标量 k 的 D_s 表示, D_s 为数字集, l 为标量 k 的 D_s 表示的长度.

由(1)式有:

$$kP = \left(\sum_{i=0}^{l-1} k_i 2^i\right) P = \sum_{i=0}^{l-1} (k_i P) 2^i = 2(2\cdots 2(2(2k_{l-1}P + k_{l-2}P) + k_{l-3}P) + \cdots + k_1 P) + k_0 P. \tag{2}$$

由(2)式, 可得从左向右计算 kP 的一般模型如下:

Algorithm 1: Left-to-Right ScalarMultiplication

Input: An integer $k \in K$ and a point $P \in E(K)$.

Output: kP.

Recoding Stage:

1. Compute the D_s-representation of the scalar $k = \sum_{i=0}^{l-1} k_i 2^i$, $k_i \in D_s$;

Precomputation Stage:

2. Compute and store dP for all integers $d \in D_s$, $d > 1$;

Evaluation Stage:

3. $R \leftarrow O$;

4. for i from $l-1$ down to 0 do

   4.1 $R \leftarrow 2R$;

   4.2 if $k_i > 0$ then $R \leftarrow R + k_i P$;

   4.3 else if $k_i < 0$ then $R \leftarrow R - (-k_i P)$;

5. return R.

Algorithm 1分为三个阶段: 重编码阶段(Recoding Stage)解决标量 k 的表示, 预计算阶段(Precomputation Stage)对数字集 D_s 中的每个元 $d > 1$ 计算并存储 dP, 赋值阶段(Evaluation Stage)计算 D_s 表示下的标量积 kP.

预计算阶段的时间消耗依赖于数字集 D_s, 赋值阶段的运行时间近似为 $lD + h_w A$, 而重编码阶段的时间消耗相对于赋值阶段来说可以忽略不计. 其中 D 表示倍点运算, A 表示点加运算, l 和 h_w 分别表示 k 的表示中的长度和汉明重量.

2　标量的表示法

由 Algorithm 1可知, 在基于“倍 – 加”运算的标量乘法算法中, 标量 k 的表示起着至关重要的作用, 其长度 l 决定了所需倍点运算量, 其汉明重量 h_w 决定了点加运算量. 目前流行的标量表示法有: 二进制表示法、NAF表示法、MOF表示法、wNAF表示法、wMOF表示法等, 不同的表示法的汉明重量不相同, 从而导致标量乘法的效率也不一样.

2.1　二进制表示法

ECC标量乘法运算中标量 k 的最基本表示方法为二进制表示法, 即 $k = \sum_{i=0}^{l-1} k_i 2^i$, 其中 $k_i \in D_s = \{0, 1\}$, 在此表示下, 标量 k 的长度 $l = \lfloor \log_2 k \rfloor + 1$, 汉明重量 h_w 的期望值为 $h_w = l/2$, 标量乘法 kP 运算不需要进行重编码及预计算, 即不进行 Algorithm 1的第 1、2步.

2.2 *NAF* 表示法

1951年 Booth[3] 提出了标量的带符号二进制表示, 后来 Rietweisner[4] 证明了对于每一个非负整数 k 均可以惟一地表示为 $k=\sum_{i=0}^{l-1}k_i2^i$, 其中 $k_i\in D_s=\{0,\ \pm1\}$. 在此表示下, 任意连续的两位数字中至多只有一位是非零的, 且是数字集 $D_s=\{0,\ \pm1\}$ 的所有带符号表示中具有最少非零位的表示, 称其为标量 k 的 NAF表示 (non-adjacent form). Morain和 Olivos[5] 证明了 NAF表示的非零数字的平均密度为 1/3. 对于正整数 k, 其 NAF表示转换算法如下:

Algorithm 2 Computation the NAF of a positive integer[6]

Input A positive integer $k\in K$.

Output NAF(k) (NAF representation of k).

1 $i=0$;

2 while $k\geqslant 1$ do

2.1 if k is odd then $k_i\leftarrow 2-(k \bmod 4)$, $k\leftarrow k-k_i$;

2.2 else $k_i=0$;

2.3 $k\leftarrow k/2$, $i\leftarrow i+1$;

3 return$(k_{l-1}, k_{l-2}, \cdots, k_1, k_0)$.

在 NAF表示下, 标量 k的长度 $l\leqslant\lceil\log_2k\rceil+2$, 汉明重量 h_w 的期望值为 $h_w=l/3$. 标量乘法 kP 运算不需要预计算, 由此得到的椭圆曲线标量乘法称为"加 - 减"方法[7], 而用 Algorithm 2计算标量 k的 NAF表示相对于后续运算来说, 其时间耗费可忽略不计.

2.3 *MOF* 表示法

2004年 Okeya[8] 基于 $k=2k-k$ 提出了标量的 MOF(mutual opposite form) 表示法, 对每一个正整数 k均可惟一地表示成MOF形式, 且两相邻非零位 (不计中间的 0比特) 的符号相反, 最高非零位和最低非零位分别为 1和 -1. MOF表示的平均汉明密度为 1/2, 但MOF表示更具有弹性, 既可以从左向右进行, 也可以从右向左进行, 从左向右的 MOF表示转换算法如下:

Algorithm 3 Left-to-Right computation the MOF of a positive integer

Input non-zero l-bit binary string $k=(k_{l-1}\cdots k_lk_0)_2$.

Output MOF(k) (MOF representation of k).

1 $u_l=k_{l-1}$;

2 for $i=l-1$ down to 1 do

2.1 $u_i=k_{i-1}-k_i$;

3 $u_0=-k_0$;

4 return $(u_l, \cdots, u_1, u_0)$.

如: $k=2927=(101101101111)_2=(1\bar{1}10\bar{1}10\bar{1}1000\bar{1})_2$. 其中 $\bar{1}$代表 -1.

MOF表示也属于带符号的二进制表示的范畴, 其数字集 $D_s=\{0,\ \pm1\}$, 其转换过程中只需基本的减法运算, 避免了取模运算, 其转换效率高于 NAF表示. 在 MOF表示下的标量乘法中也不需要预计算, 由于其平均汉明密度为 1/2, 导致赋值阶段的时间耗费较多.

2.4 *CR* 表示法

2007年 Balasubramaniam 与 Karthikeyan[9] 提出了 CR(complementary recoding) 表示法:

假设标量 k的二进制表示为 $(k_{l-1}\cdots k_1k_0)_2$, 有:

$$k=\sum_{i=0}^{l-1}k_i2^i=(100\cdots0)_{(l+1)\text{ bits}}-\bar{k}-1$$

其中 $\bar{k}=\bar{k}_{l-1}\bar{k}_{l-2}\cdots\bar{k}_0$, $\bar{k}_i=\begin{cases}0, & k_i=1\\ 1, & k_i=0\end{cases}$ $0\leqslant i<l$

如: $k=2927=(101101101111)_2=(100\cdots0)_{(12+1)\text{bits}}-\bar{k}-1=(1000000000000)_2-(010010010000)_2$

$-1 = (10\bar{1}00\bar{1}00\bar{1}000\bar{1})_2$.

在 CR表示下,标量 k的汉明重量 h_w由二进制表示下的 9减少到 5,从而减少了 4次点加运算. CR表示法也是属于带符号的二进制表示(当标量为偶数时, CR表示的最后 1位为 $\bar{2}$),但它与 NAF表示法相比,将二进制表示转换为 CR表示,避免了 NAF表示的取模运算,其转换效率高于 NAF表示,但其平均汉明密度也为 1/2,大于 NAF表示.

2.5　*w*NAF 表示法

标量 k在前述表示下,标量乘法运算均不需要进行预计算,不需要额外的存储空间存储预计算结果,不同的表示,其汉明重量 h_w不一样,从而其所需的点加运算量也不一样,如果有额外的存储单元可用,可扩大数字集 D_s,标量 k的表示的汉明重量 h_w将进一步减少,从而提高赋值阶段的运算效率.

2000年 Solinas[10]提出了标量 k的窗口 NAF表示法(wNAF):对于正整数 k,有 $k = \sum_{i=0}^{l-1} k_i 2^i$,其中每一个非零数 k_i都是奇数, $|k_i| < 2^{w-1}$, $k_{l-1} \neq 0$,并且任何连续 w个数字中最多 1位为非零.

在该表示方法中, $w \geqslant 2$称为窗口宽度. Avanzi[11]证明了这种表示方法是数字集 $D_s = \{-(2^{w-1}-1), -(2^{w-1}-3), \cdots, -1, 0, 1, \cdots, 2^{w-1}-3, 2^{w-1}-1\}$的具有最小平均汉明重量的表示.其转换算法如下:

Algorithm 4　Computation the wNAF of a positive integer

Input　Window width w, positive integer k.

Output　NAF$_w(k)$ (wNAF representation of k).

1　$i = 0$;

2　while $k \geqslant 1$ do

　2.1　if k is odd then $k_i \leftarrow k \bmod 2^w$, $k \leftarrow k - k_i$;

　2.2　else $k_i = 0$;

　2.3　$k \leftarrow k/2$, $i \leftarrow i + 1$;

3　return $(k_{i-1}, k_{i-2}, \cdots, k_1, k_0)$.

标量 k在 wNAF表示下,其长度 $l \leqslant \lceil \log_2 k \rceil + 2$,汉明重量 h_w的期望值为 $h_w = l/(w+1)$.在该表示下,标量乘法 kP运算需要对标量进行重编码、且对数字集 D_s中的每个元 $d > 1$计算并存储 dP,预计算的运行时间为: $1D + (2^{w-2}-1)A$.

文献[8]在 MOF表示的基础上提出了 wMOF表示法, wMOF表示法与 wNAF表示法有相同的平均汉明密度和相同的预计算量,但 wMOF表示法在转换的过程中更具有弹性,既可以从右向左进行,也可以从左向右进行.

3　*LZNAF* 表示法

为了减少标量 $k = (k_{l-1}, k_{l-2}, \cdots, k_1, k_0)_2$表示的汉明重量,令 $k_l = 0$、$k_{l+1} = 0$,从右向左跳过连续的‘0’,遇非‘0’按以下规则进行转换:

(1) $0101 \rightarrow 0005$,汉明重量减少 1;

(2) $0101101 \rightarrow 0005005$,汉明重量减少 2;

(3) $(01\cdots1)_{(s+1)\text{bits}} \rightarrow (10\cdots0\bar{1})_{(s+1)\text{bits}}$ ($s > 2$),汉明重量减少 $s-2$;

(4) $0\bar{1}01 \rightarrow 000\bar{3}$,汉明重量减少 1;

(5) $011 \rightarrow 003$,汉明重量减少 1.

其中规则(2)、(3)优先于规则(5).称按上述转换规则对标量 k进行编码的表示法为标量 k的 LZNAF表示法.其中 $\bar{c}$代表 $-c$.根据上述转换规则,可得如下的转换算法:

Algorithm 5　Computation the LZNAF of a positive integer

Input　A positive integer $k = (k_{l-1}\cdots k_1 k_0)_2$.

Output　LZNAF(k)　(LZNAF representation of k).

1　$u_i = 0$　($0 \leqslant i \leqslant l$)

2　$i=0$;

3　while $i<l$ do

3.1 while $k_i=0$ do $i++$;

3.2 if $(k_{i+3}k_{i+2}k_{i+1}k_i)=(0101)_2$ then $u_i=5$; $i=i+4$;

3.3 else if $(k_{i+6}\cdots k_{i+1}k_i)=(0101101)_2$ then $u_i=5$; $u_{i+3}=5$; $i=i+7$;

3.4 else if $(k_{i+s+1}\cdots k_{i+1}k_i)=(01\cdots1)_2$ and $s>2$ then

3.4.1 $k_{i+s+1}=1$;

3.4.2 if $(u_{i-1}u_{i-2})=(01)_2$ then $u_{i-2}=-3$; else $u_i=-1$;

3.4.3 $i=i+s+1$;

3.5 if $(k_{i+2}k_{i+1}k_i)=(011)_2$ then $u_i=3$; $i=i+3$;

4　return $(u_l, \cdots, u_1, u_0)$.

4　效率分析

LZNAF表示法也属于带符号表示的范畴, 其数字集 $D_s=\{0, \pm1, \pm3, 5\}$, 扫描标量 k 的二进制表示一遍便可以得到 LZNAF表示, 转换方法简单, 且不需要取模运算, 转换的时间消耗为 $O(\log_2 k)$. 只有当标量 k 的二进制表示的最左端为 $(1\cdots1)_{s\text{ bits}}$ 或 $(1101\cdots10)_{(s+4)\text{bits}}$ $(s>2)$ 时, LZNAF表示的长度比其二进制表示的长度大 1, 所以, 标量 k 的 LZNAF表示的长度至多比其二进制表示的长度大 1.

表 1　标量 $k=826033841434$ 的表示

Table 1　Encoding representations of $k=826033841434$

表示法	表示结果	h_w
二进制	1100000001010011011101000010010100011010	16
NAF	$10\bar{1}0000000\bar{1}0\bar{1}0\bar{1}00\bar{1}00\bar{1}0\bar{1}0000\bar{1}00\bar{1}0\bar{1}00\bar{1}0\bar{1}0\bar{1}0$	14
MOF	$10\bar{1}0000000\bar{1}\bar{1}\bar{1}0\bar{1}0\bar{1}100\bar{1}\bar{1}\bar{1}000\bar{1}\bar{1}0\bar{1}\bar{1}\bar{1}\bar{1}00\bar{1}0\bar{1}\bar{1}\bar{1}0$	22
CR	$100\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}0\bar{1}0\bar{1}\bar{1}00\bar{1}000\bar{1}0\bar{1}\bar{1}\bar{1}\bar{1}0\bar{1}\bar{1}0\bar{1}0\bar{1}0\bar{1}\bar{1}\bar{1}00\bar{1}0\bar{2}$	25
4NAF	$300000000050000700003000010003000\bar{7}000\bar{3}0$	8
LZNAF	$3000000010005001000030000100005000030\bar{1}0$	9

对于 $k=826033841434$ 在各种表示法下的结果如表 1所示.

与二进制、NAF、MOF、CR表示相比较, 标量 k 的 LZNAF表示的汉明重量有明显减少, 进而可明显减少赋值阶段所需的点加运算量. 尽管 LZNAF表示的汉明重量比 4NAF表示大 1, 但基于 4NAF表示的标量乘法在预计算阶段需计算并存储三个点 $3P$、$5P$ 和 $7P$, 需 1次倍点运算和 3次点加运算, 而基于 LZNAF表示的标量乘法在预计算阶段只需计算并存储两个点 $3P$ 和 $5P$, 只需 1次倍点运算和 2次点加运算, 且对于未知点的标量乘法, 预计算是一定要执行的, 从而, 基于 LZNAF表示的标量乘法的开销并不比基于 4NAF表示的标量乘法的开销大, 但可以节省一点存储空间, 这对于存储空间受限的应用来说是有吸引力的.

对于 $\{10\}^m=\underbrace{10\cdots10}_{m}$ 及 $\{10\}^m\|1=\underbrace{10\cdots10}_{m}\|1$ 型的标量的表示如表 2、3所示.

从表 2、表 3可知, 对于 $\{10\}^m$ 及 $\{10\}^m\|1$ 型的标量, 就汉明重量而言, MOF表示是最差的, NAF表示和 CR表示次之, LZNAF表示达到了 4NAF表示的效果.

表 2　长度为 $l=2m$ 的标量 $k=\{10\}^{l/2}$ 的表示

Table 2　Encoding representations of $k=\{10\}^{l/2}$

表示法	$l\%4=0$			$l\%4=2$		
	表示结果	长度	h_w	表示结果	长度	h_w
NAF	$\{10\}^{l/2}$	l	$l/2$	$\{10\}^{l/2}$	l	$l/2$
MOF	$\{1\bar{1}\}^{l/2}\|0$	$l+1$	l	$\{1\bar{1}\}^{l/2}\|0$	$l+1$	l
CR	$10\|\{\bar{1}0\}^{l/2-1}\|\bar{2}$	$l+1$	$l/2+1$	$10\|\{\bar{1}0\}^{l/2-1}\|\bar{2}$	$l+1$	$l/2+1$
4NAF	$\{5000\}^{l/4-1}\|50$	$l-2$	$l/4$	$1000\|\{5000\}^{l/4-1}\|50$	l	$l/4+1$
LZNAF	$\{5000\}^{l/4-1}\|50$	$l-2$	$l/4$	$1000\|\{5000\}^{l/4-1}\|50$	l	$l/4+1$

表 3　长度为 $l = 2m + 1$的标量 $k = \{10\}^{l/2} \| 1$的表示

Table 3　Encoding representations of $k = \{10\}^{l/2} \| 1$

表示法	$l\% 4 = 1$			$l\% 4 = 3$		
	表示结果	长度	h_w	表示结果	长度	h_w
NAF	$\{10\}^{l/2} \| 1$	l	$l/2 + 1$	$\{10\}^{l/2} \| 1$	l	$l/2 + 1$
MOF	$\{1\bar{1}\}^{l/2+1}$	$l + 1$	$l + 1$	$\{1\bar{1}\}^{l/2+1}$	$l + 1$	$l + 1$
CR	$10 \| \{\bar{1}0\}^{l/2-1} \| \bar{1}1$	$l + 1$	$l/2 + 2$	$10 \| \{\bar{1}0\}^{l/2-1} \| \bar{1}1$	$l + 1$	$l/2 + 2$
4NAF	$1000 \| \{5000\}^{l/4-1} \| 5$	l	$l/4 + 1$	$\{5000\}^{l/4} \| 5$	$l - 2$	$l/4 + 1$
LZNAF	$1000 \| \{5000\}^{l/4-1} \| 5$	l	$l/4 + 1$	$\{5000\}^{l/4} \| 5$	$l - 2$	$l/4 + 1$

5　结论

标量乘法是 ECC的基本运算,同时,也是最耗时的运算,其运算效率直接决定着 ECC的性能,本文在分析目前几种典型表示法的基础上,提出了一种新的表示方法,所提出的表示方法与目前流行的方式相比,具有编码方式简单,能有效地减少表示中的汉明重量,尤其是 $\{10\}^m$ 及 $\{10\}^m \| 1$型的标量,有较好的效率.

[参考文献]

[1] Lopez J, Dahab R. An overview of elliptic curve cryptography[R]. Brazil: Institute of Computing, State University of Combinas, 2000.

[2] Al-Somani T, Ibrahim M. High performance elliptic curve GF(2^n) cryptoprocessor secure against timing attacks[J]. International Journal of Computer Science and Network Security (IJCSNS), 2006, 6(1B): 177-183.

[3] Booth A D. A signed binary multiplication technique[J]. Journal of Applied Mathematics, 1951, 4(2): 236-240.

[4] Reitwiesner G W. Binary arithmetic[C]// Advances in Computers. New York: Academic Press, 1960(1): 231-308.

[5] Morain F, Olivos J. Speeding up the computations on an elliptic curve using addition-subtraction chains[J]. RAIRO Theoretical Informatics and Applications, 1990, 24(6): 531-543.

[6] Hankerson D, Menezes A, Vanstone S. Guide to Elliptic Curve Cryptography[M]. New York: Springer-Verlag, 2004.

[7] IEEE P1363-2000. Standard Specifications for Public-Key Cryptography[S]. New York: IEEE Standard Associations, 2000.

[8] Okeya K. Signed binary representations revisited[C]// Franklin M. Advances in Cryptology-CRYPTO 2004, Volume 3152 of LNCS. New York: Springer-Verlag, 2004: 123-139.

[9] Balasubramaniam P, Karthikeyan E. Elliptic curve scalar multiplication algorithm using complementary recoding[J]. Applied Mathematics and Computation, 2007, 190: 51-56.

[10] Solinas J A. Efficient arithmetic on Koblitz curves[J]. Designs, Codes and Cryptography, 2000, 19(2/3): 195-249.

[11] Avanir R M. A note on the sliding window integer recoding and its left-to-right analogue[C]// Proceedings of SAC 2004. Waterloo: University of Waterloo, 2004.

[责任编辑: 丁　蓉]