·学术研究·

# 抗 SPA 攻击的固定基 comb 标量乘算法*

李 杨[1,2] 王劲林[1] 曾学文[1] 叶晓舟[1]

([1] 中国科学院声学研究所 国家网络新媒体工程技术研究中心 北京 100190 [2] 中国科学院大学 北京 100049)

**摘要**:简单能量分析攻击(SPA)是椭圆曲线密码体制中比较常见的一种攻击方式。基于 Mohamed 提出的优化固定基 comb 标量乘算法,提出一种优化的标量编码算法,使得编码后标量中所有的项均不为零;并在此基础上提出一种能够抵抗 SPA 攻击的固定基 comb 算法,在保证算法性能的基础上提高算法的安全性。最终 OCTEON 平台上实验结果表明,对比能够抗 SPA 攻击的 Joye – Tunstall 算法,本文提出的算法在需要存储预计算点数量相同的情况下,计算性能得到了大幅度的提高。

**关键词**:**SPA 攻击,固定基 comb 算法,标量乘算法,椭圆曲线密码**

# Securing Fixed – Base Comb Method for Scalar Multiplication Against SPA Attacks

LI Yang[1,2], WANG Jinlin[1], ZENG Xuewen[1], YE Xiaozhou[1]

([1]National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing, 100190, China,[2]University of Chinese Academy of Sciences, Beijing, 100049, China)

**Abstract**: Simple Power Analysis attacks are particular efficient on elliptic curve cryptosystems. Based on the Mohamed's idea of computing scalar multiplication with fixed – based comb method, we proposed a new Improved scalar Encoding algorithm to make sure there is no zero item. And on the basis of it, we propose a new fixed – base comb method for scalar multiplication against SPA attacks to keep both the efficiency and security of algorithm. The experiment result on OCTEON platform shows that with the same number of precomputation points, the proposed algorithm is more efficient than the Joye – Tunstall's algorithm.

**Keywords**: SPA attack, fixed – base comb algorithm, scalar multiplication, elliptic curve cryptosystems

简单能量分析攻击(Simple Power Analysis Attack,SPA)是根据测量到单次密码操作的功率消耗轨迹,判断加密设备在某一时刻执行的指令以及所用的操作数从而恢复出使用的密钥信息。在椭圆曲线密码体系中,标量乘算法中点加和倍点运算的时间和消耗能量不同,比较容易受到 SPA 攻击。现有几种固定基 comb 算法都不能很好的抵抗 SPA 攻击,因此保证标量乘算法安全性也是需要重点解决的问题。

本文基于 Mohamed 算法[1,2],对标量的表示方式进行改进,去掉其中的零项,在此基础上提出一种可以抵抗 SPA 攻击的标量乘算法。这种算法对比 Mohamed 算法增加了少量的计算量,达到保证算法安全性的目的。对比现有的抵抗 SPA 攻击的算法,本文提出的算法具有预计算点存储数目灵活多变,存储相同与计算点效率更高等优点。

本文于 2017 – 04 – 10 收到,2017 – 04 – 25 收到修改稿。

* 中国科学院战略性先导专项课题资助项目(XDA06010302),中国科学院声学研究所知识创新工程资助项目(Y154191601)。

## 1 基础知识介绍

### 1.1 固定基 comb 方法

固定基 comb 方法的基本思想是将标量 $k$ 的二进制表示,分成 $w*v$ 块的矩阵形式。通过预计算矩阵中每块的值并进行存储,最终减少计算标量乘的时间。

1994 年 Lim 和 Lee[3]提出了固定基的 comb 方法,令 $d=[t/w]$,首先在 $k$ 的二进制表示 $k=(k_{t-1},\cdots,k_1,k_0)_2$ 左边填充 $(dw-t)$ 个 0,然后按照从右至左,从上至下的顺序将 $k$ 分成了 $w*v$ 块,$e=[d/v]$,按图 1 表示将 $k$ 分成 $w$ 行和 $d$ 列,再将每 $e$ 列组成一块则共分成 $v$ 块。经分析可知,此种算法共需计算:$U(2^w-1)$ 个平均计算点,总的计算复杂度为 $[(1-(1/2)^w)d-1]A+(e-1)D$。

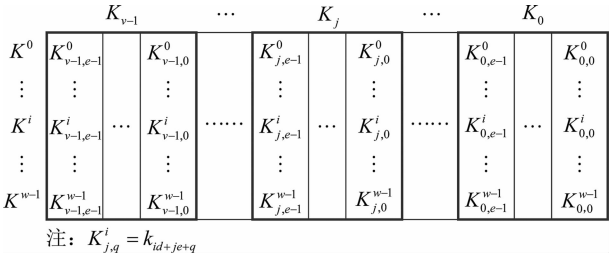


图 1 Lim – Lee 固定基 comb 标量乘算法的计算结构图

2005 年,Tsaur[4]结合标量的 NAF 表示方法和 Sakai[5,6]提出的直接计算多倍点的算法优化 Lim – Lee 算法。将 $k$ 使用 NAF 方法表示 $k=(k_{t-1},\cdots,k_1,k_0)_{NAF}$,令 $d=[t/w]$,左边填充 $(dw-t)$ 个 0,然后按照从上至下,从右至左的顺序将 $k$ 分成了 $w*v$ 块,$e=[d/v]$。如式(2)。

$$k=[\{K_{d-1}\ \cdots\ K_{(v-1)e}\}\ \cdots\ \{K_{je+(e-1)}\ \cdots\ K_{je}\}\ \cdots\ \{K_{e-1}\ \cdots\ K_0\}]$$

$$=\begin{bmatrix} k_{(d-1)w} & & k_{(v-1)ew} & & k_{(je+e-1)w} & & k_{jew} & & k_{(e-1)w} & & k_0 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ k_{(d-1)w+i} & \cdots & k_{(v-1)ew+i} & \cdots & k_{(je+e-1)w+i} & \cdots & k_{jew+i} & \cdots & k_{(e-1)w+i} & \cdots & k_i \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ k_{(d-1)w+w-1} & & k_{(v-1)ew+w-1} & & k_{(je+e-1)w+w-1} & & k_{jew+w-1} & & k_{(e-1)w+w-1} & & k_{w-1} \end{bmatrix} \tag{2}$$

因此可将计算 $kP$ 转化为如式(3)形式。

$$kP=\sum_{n=0}^{d-1}K_i2^{iw}P=\sum_{j=0}^{v-1}\sum_{i=0}^{e-1}(K_{je+i}2^{iw})2^{jew}P=\sum_{i=0}^{e-1}2^{iw}\sum_{j=0}^{v-1}\underbrace{K_{je+i}2^{jew}P}_{G[j][K_{i,j}]} \tag{3}$$

令 $G[j][K_{i,j}]=K_{je+i}2^{jew}P$,则:$kP=\sum_{i=0}^{e-1}2^{iw}\sum_{j=0}^{v-1}G[j][K_{i,j}]$。将 $K_{je+i}$ 使用 NAF 编码方式得:$K_{je+i}=(k_{(je+i)w+w-1}\cdots k_{(je+i)w})$。我们将所有的 $G[j][K_{i,j}]=K_{je+i}2^{jew}P(j\in[0,v-1],i\in[0,e-1])$ 都进行预计算存入预计算表中,供后续计算查表使用。由于 $K_{je+i}$ 是使用 NAF 编码 $w$ 比特数值,所以 $-2(2^w-1)/3\leqslant K_{je+i}\leqslant 2(2^w-1)/3$。则我们仅需预计算 $G[j][u]=u\,2^{jew}P,j\in[0,v-1],0<u\leqslant 2(2^w-1)/3$。则:

$$G[0][u]=uP$$

$$G[j][u]=2^{ew}G[j-1][u]=\cdots=2^{jew}G[0][u]=2^{jew}uP$$

由于 $0<u\leqslant 2(2^w-1)/3$,故 $G[0][u]$ 共需预计算:$2(2^w-1)/3$ 个预计算点,$G[j][u]$,$j\in[0,v-1]$ 共需计算:$2v(2^w-1)/3$ 个预计算点。

$kP=\sum_{i=0}^{e-1}2^{iw}\sum_{j=0}^{v-1}G[j][K_{i,j}]$,由于 $K_{i,j}=(k_{(je+i)w+w-1}\cdots k_{(je+i)w})$ 为零的概率是 $(2/3)^w$,因此循环中需要进行点加运算的次数为 $[(1-(2/3)^w)d-1]$。总的计算复杂度为 $[(1-(2/3)^w)d-1]A+(e-1)D^*$,其中 $D^*$ 表示使用 sakai – sakurai[5] 算法计算 $2^wP$。

2012 年 Mohamed 算法对 Tsaur 提出的方法进一步进行改进,将 $k$ 使用 $w$NAF 的方式进行表示,其他步骤不变,这里不再赘述。由于 $w$NAF 表示的标量中相邻的 $w$ 位中仅有一位为非零数,且非零项 $k_i\in\{\pm1,\pm3,\cdots,\pm(2^{w-1}-1)\}$。因此我们可以把所有需要预计算的点用 $G[j][sr]$,$j\in[0,v-1]$,$s=2^u\in\{1,2,$

$2^2,\cdots,2^{w-1}\}$，$r\in\{1,3,\cdots,(2^{w-1}-1)\}$表示。

$$G[0][sr]=rsP$$

$$G[j][sr]=2^{ew}G[j-1][sr]=\cdots=2^{jew}G[0][sr]=2^{jew}rsP$$

由于 $s=2^u\in\{1,2,2^2,\cdots,2^{w-1}\}$，$r\in\{1,3,\cdots,(2^{w-1}-1)\}$，故 $G[0][sr]$共需预计算：$w2^{w-2}$个预计算点。$G[j][sr]$，$j\in[0,v-1]$共需计算：$vw2^{w-2}$个预计算点。

由于 $K_{i,j}=(k_{(je+i)w+w-1}\cdots k_{(je+i)w})$使用 wNAF 的方式表示，因此 $K_{i,j}$为零的概率是$(w/(w+1))^w$，因此循环中需要进行点加运算的次数为：$[(1-(w/(w+1))^w)d-1]$。总的计算复杂度为$[(1-(w/(w+1))^w)d-1]A+(e-1)D^*$，其中 $D^*$ 表示使用 sakai－sakurai 算法[5]计算 $2^wP$。

### 1.2　椭圆曲线上的 SPA 攻进介绍

简单能量分析攻击(SPA)是根据测量到的功率消耗轨迹，判断加密设备在某一时刻执行的指令以及所用的操作数，从而恢复出使用的密钥信息。在椭圆曲线密码体系中，点加和倍点运算所有的时间和消耗能量不同，因此比较容易受到 SPA 攻击。

目前抵抗 SPA 攻击主要有以下几种方法。第一种是通过加一些多余的操作将算法循环中每次计算都设置成固定的模式[7]；第二种是使用一些特殊的曲线形式，比如说 Montgomery 曲线形式等[8]；另外可以使用 Weierstrass 曲线上统一的点加和倍点公式[9]。以上几种方法的基本思想都是将算法设置成固定模式，不能通过区分点加和倍点的不同来获取标量的相关信息。

针对如何使固定基 comb 算法能够抵抗 SPA 攻击，已有的文献中已经提出了很多方法。Möller[10]基于 $m$ $(m=2^w)$进制，利用$(0,0)=(1,\overline{2^w})$将标量表示中的零元素去掉，使得循环中每次都进行一次点加和倍点运算，从而能够抵抗 SPA 攻击；Okeya[11]将 $k$ 的 wNAF 表示分成大小为 $w$ 的块，通过$(0,0,\cdots,1)=(1,\overline{1},\cdots,\overline{1})$的基本思想将标量表示中的为零和偶数的块都转化为奇数块；Joye[12]提出了新的标量编码方法，使得标量表示总不包含零元素。已有文献中抵抗 SPA 攻击的大部分方法的基本思想是通过重新编码，去掉标量表示中的零项。

## 2　本文提出的算法

Mohamed 算法中，当 $K_i=(K_i^{w-1},\cdots,K_i^1,K_i^0)=(0,\cdots,0,0)$时，只需要做一次多倍点$(D^*)$运算即可；而当 $K_i=(K_i^{w-1},\cdots,K_i^1,K_i^0)\neq(0,\cdots,0,0)$时，需要做一次多倍点$(D^*)$和一次点加$(A)$运算。攻击者可以通过运算时的能量信息分析当前的 $K_i$ 是否为零，即 Mohamed 算法不能抵抗 SPA 攻击。

本节提出一种优化的标量编码算法，使得编码后标量中所有的 $K_i$ 均不为零。并在此基础上提出一种能够抵抗 SPA 攻击的固定基 comb 算法，在保证算法性能的基础上提高算法的安全性。

### 2.1　优化的标量编码方法

本节是依据窗口算法进行改进，进行了两次优化。用优化后的编码方式代替原始 Mohamed 算法中宽度为 $w$ 的 NAF 表达方式，使得优化后编码被分成 $w$ 大小的块后，每块均不为零。

长度为 $t$ 的标量 $k$ 的二进制表示为 $k=\sum_{i=0}^{t-1}k_i2^i,k_i\in\{0,1\}$，令窗口大小为 $w$，则整个标量 $k$ 可被分成 $d=[t/w]$块。原始的窗口方法是将 $k$ 的二进制表示转化为 $k'=\sum_{i=0}^{t-1}k_i'2^i,k_i'\in\{0,1,2,\cdots,(2^w-1)\}$。具体做法是将 $k$ 的二进制表示分成大小为 $w$ 的块，即 $k=[K_{d-1},K_{d-2},\cdots,K_0]$。将其中的每一块 $K_i=(k_{iw+w-1},\cdots,k_{iw+1},k_{iw})$，转化为$(0,\cdots,0,k_{iw}')$，其中：$k_{iw}'\in\{0,1,\cdots,(2^w-1)\}$。

例如：$k=(10000000110000000011111010)_2$，$t=24$，$w=4$，$d=[t/w]=6$。

将标量 $k$ 的二进制表示，$k=\sum_{i=0}^{t-1}k_i2^i,k_i\in\{0,1\}$，转化为 $m=\sum_{i=0}^{t-1}m_i2^i,m_i\in\{0,1,3,5,\cdots,(2^w-1)\}$。

$K_i$ 中若 $k_{iw} = \cdots = k_{iw+j-1} = 0$，且 $k_{iw+j} \neq 0$，则：$K_i = (k_{iw+w-1}, \cdots, k_{iw+1}, k_{iw})$ 转化为 $(0, \cdots, 0, m_{iw+j}, 0, \cdots, 0)$，$m_{iw+j} \in \{1, 3, \cdots, (2^{w-j} - 1)\}$。

例如：$k = (1000000011000000001111010)_2$，$t = 24$，$w = 4$，$d = [t/w] = 6$。

| | $K_5$ | $K_4$ | $K_3$ | $K_2$ | $K_1$ | $K_0$ |
|---|---|---|---|---|---|---|
| $k$： | 1000 | 0000 | 1100 | 0000 | 0111 | 1010 |
| $k'$： | 0008 | 0000 | 000C | 0000 | 0007 | 000A |

| | $K_5$ | $K_4$ | $K_3$ | $K_2$ | $K_1$ | $K_0$ |
|---|---|---|---|---|---|---|
| $k$： | 1000 | 0000 | 1100 | 0000 | 0111 | 1010 |
| $m$： | 1000 | 0000 | 0300 | 0000 | 0007 | 0050 |

若 $K_i = (k_{iw+w-1}, \cdots, k_{iw}) = (0, \cdots, 0) = 0$，则转化后 $K_i = (m_{iw+w-1}, \cdots, m_{iw}) = 0$。为了避免零元素的出现，则进行进一步的优化，对于 $0 < i < d$，对 $(K_i, K_{i-1})$ 做如式(4)变化。

$$(K_i, K_{i-1}) = \begin{cases} (1, K_{i-1} - 2^w), & \text{若 } K_i = 0 \\ (K_i, K_{i-1}), & \text{否则} \end{cases} \tag{4}$$

其中 $K_{i-1} = K_{i-1} - 2^w$ 运算时，若 $K_{i-1} = (0, \cdots, m_{(i-1)w+j}, \cdots, 0)$，则 $K_{i-1} - 2^w = (0, \cdots, m_{(i-1)w+j} - 2^{w-j}, \cdots, 0))$。由于 $m_{(i-1)w+j} \in \{1, 3, \cdots, (2^{w-j} - 1)\}$，则 $m_{(i-1)w+j} - 2^{w-j} \in \{-1, -3, \cdots, -(2^{w-j} - 1)\}$。注意上述算法中 $i > 0$ 时，对 $K_i$ 进行变换，若 $K_0 = 0$ 是不能通过上述方法去掉零项的。因此，此种方法不适用于满足 $k \bmod 2^w = 0$ 的标量 $k$。

例如：$k = (1000000011000000001111010)_2$，$t = 24$，$w = 4$，$d = [t/w] = 6$。

| | $K_5$ | $K_4$ | $K_3$ | $K_2$ | $K_1$ | $K_0$ |
|---|---|---|---|---|---|---|
| $m$： | 1000 | 0000 | 0300 | 0000 | 0007 | 0050 |
| $m'$： | 1000 | 0001 | $0\,\bar{1}00$ | 0001 | $000\,\bar{9}$ | 0050 |

证明上述变换的正确性，若 $K_i = 0$，则 $(K_i, K_{i-1}) = (1, K_{i-1} - 2^w)$，其中 $K_{i-1} = (0, \cdots, m_{(i-1)w+j}, \cdots, 0)$，$K_{i-1} - 2^w = (0, \cdots, m_{(i-1)w+j} - 2^{w-j}, \cdots, 0)$。即上述证明可转化为：$\{(0, \cdots, 0)(0, \cdots, m_{(i-1)w+j}, \cdots, 0)\} = \{(0, \cdots, 1)(0, \cdots, m_{(i-1)w+j} - 2^{w-j}, \cdots, 0)\}$ 证明。

证明：由 $\{(0, \cdots, 0)(0, \cdots, m_{(i-1)w+j}, \cdots, 0)\} = m_{(i-1)w+j} \times 2^{(i-1)w+j}$；

$\{(0, \cdots, 1)(0, \cdots, m_{(i-1)w+j} - 2^{w-j}, \cdots, 0)\} = 1 \times 2^{iw} + (m_{(i-1)w+j} - 2^{w-j}) \times 2^{(i-1)w+j}$

$= 2^i w + m_{(i-1)w+j} \times 2^{(i-1)w+j} - 2^{w-j+(i-1)w+j} = m_{(i-1)w+j} \times 2^{(i-1)w+j} = \{(0, \cdots, 0)(0, \cdots, m_{(i-1)w+j}, \cdots, 0)\}$

则 $\{(0, \cdots, 0)(0, \cdots, m_{(i-1)w+j}, \cdots, 0)\} = \{(0, \cdots, 1)(0, \cdots, m_{(i-1)w+j} - 2^{w-j}, \cdots, 0)\}$

因此证明了上述变换的正确性。

至此我们定义了一种新的编码方式 $k = \sum_{i=0}^{t-1} m_i 2^i, m_i \in \{0, \pm 1, \pm 3, , \cdots, \pm (2^w - 1)\}$。当将 $k$ 分成大小为 $w$ 的块，即 $k = [K_{d-1}, K_{d-2}, \cdots, K_0]$，$K_i = (m_{iw+w-1}, \cdots, m_{iw})$，$K_i$ 中有且仅有一位 $m_{iw+j}$ 非零，且 $m_{iw+j} \in \{\pm 1, \pm 3, \cdots, \pm (2^{w-j} - 1)\}$。

算法 1 详细描述了此种编码算法。其中：$m_{iw+j} = |k_{iw+(w-1)} \cdots k_{iw+j}|$ 表示将 $(k_{iw+(w-1)} \cdots k_{iw+j})$ 的 $2^w$ 进制值赋给 $m_{iw+j}$。例如：$m_{iw+j} = |1001| = 9$；$m_{iw+j} = |1011| = B$，则实现代码如下。

算法 1：优化的标量编码算法(Improve recoding method – IR)

输入：正整数 $k$ 的二进制编码：$k=(k_{t-1},\cdots,k_1,k_0)_2$，$k_i\in\{0,1\}$，且满足 $k \bmod 2^w\neq 0$；

窗口宽度：$w$。

输出：$m=(m_{t-1},\cdots,m_1,m_0)_{IR}$，$m_i\in\{0,\pm1,\pm3,\cdots,\pm(2^w-1)\}$

$d=[t/w]$；$j=0$；

while($k_j=0$)

$j++$；

$m_0=m_1=\cdots=m_{j-1}=0$；$m_j=|k_{w-1}\cdots k_j|$；$m_{j+1}=\cdots=m_{w-1}=0$；

for i = 1 to d – 1 do

*if* $K_i=(k_{iw+w-1},\cdots,k_{iw})=0$ *then* //若 $K_i=0$，则 $(k_i,k_{i-1})=(1,k_{i-1}-2^w)$

$m_{iw+w-1}=\cdots=m_{iw+1}=0$；

$m_{iw}=1$；

$m_{(i-1)w+j}=m_{(i-1)w+j}-2^{w-j}$；

*else then*//若 $K_i\neq 0$，则 $K_i=(k_{iw+w-1},\cdots,k_{iw})$ 转化为 $(0,\cdots,0,m_{jw+g},0,\cdots,0)$

$j=0$；

*while*($k_{iw+j}=0$)

$j++$；

$m_{iw}=\cdots=m_{iw+(j-1)}=0$；

$m_{iw+j}=|k_{iw+(w-1)}\cdots k_{iw+j}|$；

$m_{iw+(j+1)}=\cdots=m_{iw+(w-1)}=0$；

*return* $m=(m_{t-1},\cdots,m_1,m_0)$；

## 2.2　抗 SPA 攻击的固定基 comb 标量乘算法

基于 Mohamed 算法提出固定基 comb 标量乘算法的基本思想，将原算法中标量表示使用宽度为 $w$ 的 NAF 算法，改为使用 3.1 节提出的优化编码算法。在算法性能不受影响的基础上，保证算法的安全性使其能够抵抗 SPA 攻击。

首先按照从上至下，从右至左的顺序，将长度为 $t$ 的的标量 $k$ 分为 $d=[t/w]$ 块，每块大小为 $w$。即将 $k$ 表示为如式(5)形式，其中 $K_i=(K_i^{w-1},\cdots,K_i^1,K_i^0)=(k_{iw+w-1},\cdots,k_{iw+1},k_{iw})$。

$$k=[K_{d-1}K_{d-2}\cdots K_1K_0]=\sum_{i=0}^{d-1}K_i2^{iw} \tag{5}$$

然后将 $d$ 块 $[K_{d-1},\cdots,K_{d-2},\cdots,K_0]$ 分成 $v$ 大块，每块包含 $e=[d/v]$ 个 $K_i$，即将 $k$ 表示为如公式(2)形式。

$$k=[\{K_{d-1}\cdots K_{(v-1)e}\}\cdots\cdots\{K_{je+(e-1)}\cdots K_{je}\}\cdots\cdots\{K_{e-1}\cdots K_0\}] \tag{6}$$

本文提出算法的计算顺序与 Tsaur – Chou 算法基本相同，图 2 描述了抗 SPA 攻击的固定基 comb 标量乘算法的计算结构图。因此计算 $kP$ 可以转化为公式(7)的形式。

$$kP=\sum_{n=0}^{d-1}K_i2^{iw}P=\sum_{j=0}^{v-1}\sum_{i=0}^{e-1}(K_{je+i}2^{iw})2^{jew}P=\sum_{i=0}^{e-1}2^{iw}\sum_{j=0}^{v-1}\underbrace{K_{je+i}2^{jew}P}_{G[j][I_{i,j}]} \tag{7}$$

其中，$K_{je+i}=(k_{(je+i)w+w-1}\cdots k_{(je+i)w})$，使用 3.1 节中提到的优化编码方式。我们将所有的 $G[j][I_{i,j}]=K_{je+i}2^{jew}P(j\in[0,v-1],i\in[0,e-1])$ 都进行预计算存入预计算表中，供后续计算查表使用。对于 $K_iP=(K_i^{w-1},\cdots,K_i^1,K_i^0)P=K_i^{w-1}2^{w-1}P+\cdots+K_i^12P+K_i^0P$，由于优化的编码方式中 $w$ 位内有且仅有一项为非零值，假设第 $u$ 项($u\in[0,w-1]$)为非零值，则 $K_iP=K_i^u2^uP=rsP$，$s=2^u\in\{1,2,2^2,\cdots,2^{w-1}\}$ $r\in\{\pm1,\pm3,\cdots,\pm(2^{w-u}-1)\}$。由于在椭圆曲线中计算 $-P$ 的运算量很小基本可以忽略。因此，我们可以将所有需要预计算的点用 $G[j][sr]$，$j=[0,v-1]$，$s=2^u\in\{1,2,2^2,\cdots,2^{w-1}\}$，$r\in\{1,3,\cdots,(2^{w-u}-1)\}$ 表示，则：

$$G[0][sr]=K_i^u2^uP=rsP$$

$$G[j][sr]=2^{ew}G[j-1][sr]=\cdots=2^{jew}G[0][sr]=2^{jew}rsP$$

由于 $r\in\{1,3,\cdots,(2^{w-u}-1)\}$ 共包含 $2^{w-u-1}$ 个点，且 $u\in[0,w-1]$，故 $G[0][sr]$ 共需预计算 $\sum_{s=0}^{w-1}2^{w-s-1}$

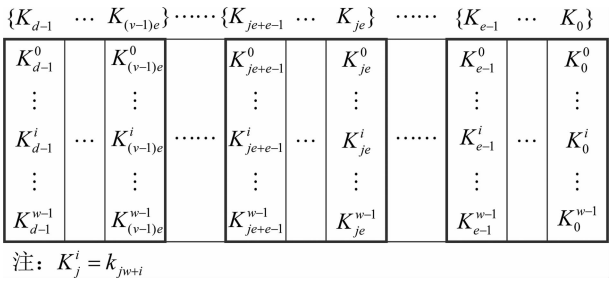


图 2　抗 SPA 攻击的固定基 comb 标量乘算法的计算结构图

$=2^0+2^1+\cdots+2^{w-1}=2^w-1$ 个预计算点。则 $G[j][sr], j\in[0,v-1]$ 共需计算:$v(2^w-1)$ 个预计算点。

算法 2 详细描述了抗 SPA 攻击的固定基 comb 标量乘算法。由于 3.1 节中提到的优化编码方式要求满足 $k\ mod\ 2^w\neq 0$。$k\ mod\ 2^w=0$ 时,令 $k=k+1$,计算完 $kP$ 后再在结尾处减去 $P$,即计算$((k+1)P-P)$。为了能够抵抗 SPA 攻击,我们必须使计算模式完全相同,当 $k\ mod\ 2^w\neq 0$ 时,令 $k=k+2$,再在结尾处减去 $2P$,即计算$((k+2)P-2P)$。

算法 2:抗 SPA 攻击的固定基 comb 标量乘算法

输入:正整数 $k$ 的二进制表示:$k=(k_{t-1},\cdots,k_1,k_0)_2$;正整数 $w,v$;椭圆曲线点 $P\in E(F_q)$

输出:$Q=kP$

$d=[t/w]$; $e=[d/v]$; $j=0$;

*if* $k\ mod\ 2^w=0$ *then*

$k=k+1$;

*elsethen*

$k=k+2$;

使用算法 1 对 $k$ 进行编码,得到 $k=(m_{t-1},\cdots,m_1,m_0)_{IR}$;

对所有:$j\in[0,v-1]$, $s=2^u\in\{1,2,2^2,\cdots,2^{w-1}\}$, $r\in\{1,3,\cdots,(2^{w-u}-1)\}$,

预计算 $G[0][sr]=rsP$ 和 $G[j][sr]=2^{jew}G[0][sr]$;

$Q=0$;

*for* $i=e-1$ *to* 0 *do*

*if* $w=1$ *then*

$Q=2Q$;

*else then*

使用 sakai - sakurai 算法计算 $Q=2^wQ$;

*for* $j=v-1$ *to* 0 *do*

$I_{i,j}=(m_{(je+i)w+w-1}\cdots m_{(je+i)w})_{IR}$;

*if* $I_{i,j}>0$ *then*

$Q=Q+G[j][I_{i,j}]$;

*else then*

$Q=Q-G[j][-I_{i,j}]$;

*if* $k\ mod\ 2^w=0$ *then*

$Q=Q-P$;

*elsethen*

$Q=Q-G[0][2]$; ## 相当于计算 $Q=Q-2P$

*return* $Q$;

由算法 2 分析可知,主要循环阶段算法复杂度是:$(e-1)D^*+(d-1)A$,其中 $D*$ 表示使用 sakai - sakurai 算法计算 $2^wP$。另外在最后一步恢复 $kP$ 时使用了一次 $A$。因此此种方法计算标量乘的算法复杂度是 $(e-1)D^*+dA$。无论 $k$ 取值如何,标量乘算法的运算都是一样的,因此此种算法能够有效的抵抗 SPA 攻击。

## 3 仿真实验与分析

本节中我们对本文提出的算法与现阶段比较典型的固定基 comb 算法在算法计算性能和安全性方面进行分析。主要针对需存储预计算点个数,运算时算法的复杂度以及是否能够抵抗 SPA 攻击等方面进行比较。如表 1 所示。表 1 中比较了各种算法需要预计算存储点的个数,算法复杂度和是否能够抵抗 SPA 攻击。标量 $k$ 的二进制表示长度为 $t$,将其分为 $w*v$ 块,$d=[t/w]$, $e=[d/v]$。$A$ 表示点加运算;$D$ 表示倍点运算;$D^*$ 表示使用 sakai - sakurai 算法计算 $2^wP$。

**表 1 几种固定基 comb 算法性能和安全性对比**

| 算法名称 | 算法复杂度 | 预计算存储点的个数 | 是否抗 SPA 攻击 |
|---|---|---|---|
| Lim - Lee 算法 | $\left[\left(1-\left(\frac{1}{2}\right)^w\right)d-1\right]A+(e-1)D$ | $v(2^w-1)$ | 否 |
| Tsaur - Chou 算法 | $\left[\left(1-\left(\frac{2}{3}\right)^w\right)d-1\right]A+(e-1)D*$ | $\frac{2}{3}v(2^w-1)$ | 否 |
| NHH 算法 | $\left[\left(1-\left(\frac{w}{w+1}\right)^w\right)d-1\right]A+(e-1)D*$ | $w2^{w-2}v$ | 否 |
| 本文提出算法 | $dA+(e-1)D*$ | $v(2^w-1)$ | 是 |
| Möller 算法 | $(d-1)A+w(d-1)D$ | $2^{w-1}+1$ | 是 |
| Okey - Takagi 算法 | $dA+[w(d-1)+1]D$ | $2^{w-1}$ | 是 |
| Joye - Tunstall 算法 | $(d-1)A+w(d-1)D$ | $2^{w-1}$ | 是 |

从 1 表中可以看出，表中前三种算法与本文提出的算法相比较算法复杂度较低，但是其不能抵抗 SPA 攻击。本文提出的算法虽然少量提高了计算的复杂度，但是安全性更高，能够很好的抵抗 SPA 攻击。目前比较典型的抵抗 SPA 攻击的固定基 comb 算法有 Möller 算法，Okey－Takagi 算法和 Joye－Tunstall 算法。通过比较可以看出在预计算点相同时，Joye－Tunstall 算法的算法复杂度最低。下面重点针对本文提出的算法和 Joye－Tunstall 算法在预计算点个数相同时，运算性能进行分析比较。

在 CAVIUM 公司的 OCTEON CN6645 平台上[14,16]基于开源软件库 OpenSSL，针对 160bit、192bit、224bit 和 256bit 标量乘，对本文提出的算法和 Joye－Tunstall 算法进行对比。在预计算存储点个数相同的情况下，比较两种算法的性能。测试 OpenSSL 软件包中的函数 BN_mod_mul(M)，BN_mod_sqr(S)和 BN_mod_inverse(I)的性能可知，OCTEON 平台上 M，S，I 三种操作消耗时间比约为 S＝0.9M，I＝35M。因此仿射坐标系下倍点算法复杂度为 D＝I＋2M＋2S＝38.8M；点加的算法复杂度为 A＝I＋2M＋S＝37.9M＝0.977D，sakai－sakurai 算法计算 $2^w P$ 复杂度为 $D^* = I + (4w+1)M + (4w+1)S = (7.6w+36.9)M = (0.1959w+0.951)D = hD$，$(w>1)$；当 $w=1$ 时，$D^*$ 按照基本倍点方式计算，因此 $D^* = D$。表 2 列出当 $w$ 取不同值时，$w$ 和 $h$ 的关系。

**表 2　$w$ 与 $h$ 关系表格**

| $w$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $h$ | 1 | 1.343 | 1.539 | 1.735 | 1.930 | 2.126 | 2.322 | 2.518 |

由表 1 可知，Joye－Tunstall 算法，当取定 $w$ 值时，预计算点个数为 $2^{w-1}$，计算复杂度为 $(d-1)A + w(d-1)D$，$d = [t/w]$。使用 Joye－Tunstall 算法分别在 160bit，192bit，224bit 和 256bit 的曲线上进行标量乘，将 $w$ 取不同值，得出需要存储预计算点个数和计算复杂度的关系如图 3 所示。其中预计算点单位是个；计算复杂度通过 $A = 0.977D$，将其转化为求取需要进行多少次 $D$ 运算，因此单位是 $D$。此外，使用本文提出的算法分别在 160bit，192bit，224bit 和 256bit 的曲线上进行标量乘，$w$ 和 $v$ 取不同值，预计算点个数为 $v(2^w-1)$，计算复杂度为：$dA + (e-1)D^*$，$d = [t/w]$，$e = [d/v]$，得出需要存储预计算点个数和计算复杂度的关系如图 3 所示。其中预计算点单位是个；通过 $A = 0.977D$，$D^* = (0.1959w + 0.951)D$，将其转化为 $D$，计算复杂度的单位是 $D$。在获得图 3 曲线上的点时，遵循下面原则：①$(w,v)$ 取不同值时，得到的预计算点个数相同，选取计算复杂度较小的情况，画入曲线中。例如：$(w,v) = (2,5)$ 和 $(w,v) = (4,1)$ 时需要存储预计算点个数都为 15 个，但 $(w,v) = (2,5)$ 时所需的计算复杂度小，因此取此种情况画入曲线中。②当需要存储预计算点个数增加时，算法的计算复杂度却没有减小，这种情况的点也舍弃不画入曲线中。

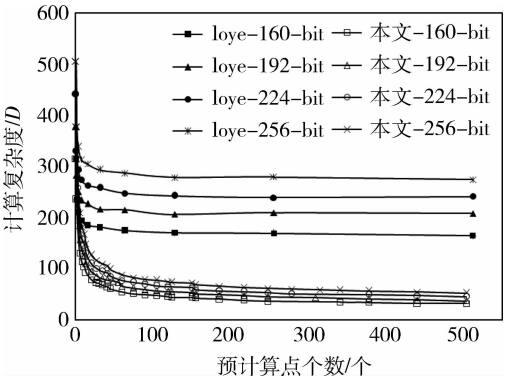


图 3　两种算法计算复杂度对比

图 3 对比了本文提出的算法和 Joye－Tunstall 算法，预计算点个数相同时，所需的计算复杂度，即需要计算多少个倍点(D)运算。由图可以看出在常用位数的椭圆曲线上，存储预计算点个数相同时，本文提出算法所需进行的运算量小于 Joye－Tunstall 算法，性能也远远优于此种算法。综上所述，本文提出的算法在保证性能的基础上也保证算法的安全性，使其能够抵抗 SPA 攻击。

## 4　结束语

本文提出一种能够抵抗 SPA 攻击的优化固定基 comb 算法，此种算法对 Mohamed 算法进行优化改进，外部使用 Mohamed 算法的结构，对内部标量表示方式进行改进，在保证算法性能的基础上使其能够抗 SPA 攻

击。最终在 OCTEON 平台上,分别对 160/192/224/256 bit 的标量乘算法进行评估,对比本文提出算法和能够抗 SPA 攻击的 Joye – Tunstall 算法,在存储预计算点个数相同的情况下,本文提出算法所用的计算时间远远小于 Joye – Tunstall 算法。

## 参 考 文 献

[1] Mohamed N A F, Hashim M H A, Hutter M. Improved Fixed – Base Comb Method for Fast Scalar Multiplication[C]//International Conference on Cryptology in Africa. 2012.342 – 359

[2] Mohamed N A F. New Fixed – base Comb Methods for Point Multiplication in Elliptic Curve Cryptosystems[D]. University of Khartoum, 2012.

[3] Lim C H, Lee P J. More Flexible Exponentiation with Precomputation[C]//International Cryptology Conference on Advances in Cryptology. Springer – Verlag, 1994.95 – 107

[4] Tsaur W J, Chou C H. Efficient algorithms for speeding up the computations of elliptic curve cryptosystems[J]. Applied Mathematics & Computation, 2005, 168(2):1045 – 1064

[5] Sakai Y. Speeding Up Elliptic Scalar Multiplication Using Multidoubling[J]. Ieice Transactions on Fundamentals of Electronics Communications & Computer Sciences, 2002, 85(5): 1075 – 1083

[6] Sakai Y, Sakurai K. On the Power of Multidoubling in Speeding Up Elliptic Scalar Multiplication[J]. Ieice Transactions on Fundamentals of Electronics Communications & Computer Sciences, 2001, E85A(5): 268 – 283

[7] Yen S M, Kim S, Lim S, et al. A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack[C]//International Conference Seoul on Information Security and Cryptology. Springer – Verlag, 2001.414 – 427

[8] Okeya K, Sakurai K. Power Analysis Breaks Elliptic Curve Cryptosystems Even Secure against the Timing Attack[C]//Progress in Cryptology – INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10 – 13, 2000, Proceedings. DBLP, 2000. 178 – 190

[9] Izu T, Takagi T. A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks[M]. Public Key Cryptography. Springer Berlin Heidelberg, 2002. 280 – 296

[10] Möller B. Securing Elliptic Curve Point Multiplication Against Side – Channel Attacks[C]//International Conference on Information Security. Springer – Verlag, 2001. 324 – 334

[11] Okeya K, Takagi T. The Width – w NAF Method Provides Small Memory and Fast EllipticScalar Multiplications Secure against Side Channel Attacks[C]//Topics in Cryptology – CT – RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13 – 17, 2003, Proceedings. DBLP, 2003: 328 – 342

[12] Joye M, Tunstall M. Exponent Recoding and Regular Exponentiation Algorithms[C]//Progress in Cryptology – AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21 – 25, 2009. Proceedings. DBLP, 2009: 334 – 349

[13] Cavium. OCTEON II CN66XX Multi – Core MIPS64 Processors[J/OL]. [2011 – 07].

[14] 李军, 陈君, 倪宏,等. 基于多核协作的流媒体内容缓存算法[J]. 网络新媒体技术, 2014, 3(4):12 – 18

## 作者简介

李杨,女,(1990 – ),博士研究生,研究方向:网络安全,国密算法。