一种低复杂度的改进wNAF标量乘算法

赵石磊,杨晓秋,刘志伟,于 斌,黄 海
(哈尔滨理工大学计算机科学与技术学院,黑龙江哈尔滨 150080)

摘 要: 在物联网等资源受限的环境中,低计算复杂度、存储空间占用少的标量乘算法尤为重要.为了降低标量乘的计算复杂度,本文采用有符号的窗口非相邻算法(window width-Non-Adjacent Form,wNAF)生成标量k的wNAF链;用2^n替换wNAF链中的奇数,替换后的差值则通过构造微小的加法链进行弥补.该算法能降低预计算点的个数,解决wNAF标量乘不适用于窗口宽度较大的问题.相较于wNAF算法、swNAF算法和基于素数预计算的算法,当窗口宽度为11时,该算法只需要12个预计算点,预计算点分别减少了98.83%,96.49%和94.42%,标量乘计算复杂度优化了78.23%,68.94%和43.63%.

关键词: 标量乘;窗口非相邻算法;加法链
中图分类号: TN918.1 **文献标识码:** A **文章编号:** 0372-2112(2022)04-0977-07
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20211016

An Improved wNAF Scalar-Multiplication Algorithm with Low Computational Complexity

ZHAO Shi-lei, YANG Xiao-qiu, LIU Zhi-wei, YU Bin, HUANG Hai
(*School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, Heilongjiang* 150080, *China*)

Abstract: It is very important that scalar multiplication algorithm has low computational complexity and less storage space in the resource constrained environment such as the Internet of things. In order to reduce the complexity of scalar multiplication, wNAF is used to generate the wNAF chain of scalar k; it replaces odd number with power of 2, and makes up the difference between power of 2 and odd by building the addition chain. At the same time, the algorithm reduces the number of precomputed points, and solves the problem that wNAF scalar multiplication is not suitable for large window width. Compared the wNAF, swNAF, and precomputation algorithm based prime, only 12 are needed to store the precomputed points, and the number of precomputed points are optimized by 98.83%, 96.49%, and 94.42%, and the scalar multiplication calculation stage are optimized by 78.23%, 68.94%, and 43.63% when the window width is 11.

Key words: scalar-multiplication; windowed non-adjacent form; addition chains

1 引言

椭圆曲线密码体制的核心运算主要是椭圆曲线群上的标量乘法,它的运算速度决定着密码系统整体执行效率[1].一般来说,标量乘算法可分为两种:无预计算算法和有预计算算法.无预计算算法包括二进制算法(Double and Add, D&A)[2]、非邻接表示算法(Non-Adjacent Form, NAF)[3]、多基链表算法(Double-Base Chain, DBC)[4]和相反形式算法(Mutual Opposite Form, MOF)[5]等,这些算法大都存在着计算复杂度过高的问题.为降低计算复杂度,学者们提出了有预计算算法,如滑动窗口算法[6]、窗口非相邻算法(window width-Non-Adjacent Form, wNAF)[7]、加法链算法[8]、利用素数代替奇数进行预计算并通过构建多基链来弥补素数与奇数之间的差值算法[9]、带门限的动态窗口的NAF标量乘法[10]等.文献[8]所提加法链算法相比较其他加法链算法计算复杂度降低了4%~18%.文献[9]的计算复杂度相比于wNAF优化了28.37%.在文献[10]中,带门限的动态窗口的NAF标量乘法的预计算量仅为基于Moller碎片窗口技术的标量乘法的30%、基于固定窗口改进后的NAF标量乘法的25%,其预计算利用率比基

收稿日期:2021-08-01;修回日期:2022-01-23;责任编辑:孙瑶
基金项目:国家重点研发计划"光电子与微电子器件及集成"重点专项子课题(No.2018YFB2202100);黑龙江省自然科学基金优秀青年项目(No.YQ2019F010);黑龙江省普通高校基本科研业务费专项资金资助(No.2019KYYWF0214)

于Moller碎片窗口技术的标量乘法提高了15%左右，比基于固定窗口改进后的NAF标量乘法提高了12%左右. 文献[11]则提出了Alternate-Zeckendorf表示算法，可以用加法链序列表示任何标量k，此算法比其他算法的成本至少降低12.7%. 文献[12]提出了基于窗口的NAF算法，当$n=\{192,224,256,384\}$时，预计算效率提高了26.35%，当$n=521$时，预计算效率提高33.59%.

以上算法虽然取得了较好的结果，能够有效降低标量k的汉明重量，减少点加操作的次数，降低标量乘的计算复杂度，但是仍然存在着不适用于窗口宽度较大、使用寄存器较多、预计算量过大等问题. 针对以上问题，本文设计了一种低复杂度的改进wNAF算法，首先在预计算阶段用2^nP替换$(2h-1)P$（wNAF算法中奇数与基点P的乘积），再用2^nP构造加法链来补偿2^nP与$(2h-1)P$之间的差值. 然后，为解决随着窗口宽度的增大，2^nP与$(2h-1)P$之间差值过大的问题，采用有符号的wNAF算法将标量k的二进制链转换成有符号的wNAF链，能将该差值范围缩小为原来的50%，极大地减少了点加次数. 最后，实验结果证明，所设计的算法具有所需寄存器数量少以及计算复杂度低等优点.

2 椭圆曲线标量乘算法分析

2.1 窗口非相邻(wNAF)标量乘

椭圆曲线标量乘法是椭圆曲线密码系统中最关键（也是消耗资源和能量最多）的步骤，是椭圆曲线上一个点P与一个随机的整数k的乘积，即$Q=kP=P+P+\cdots+P$. 由于在雅可比坐标下，倍点的运算消耗小于点加，因此通常做法是将k进行转换以减少其中点加次数，如wNAF标量乘算法[13]. wNAF标量乘算法通过将二进制形式的标量k转化为wNAF形式，构建一条wNAF链，即$k_i=\sum_1^m k_i 2^{\lambda_i}$，其中$k_i\in\{\pm1,\pm3,\cdots,\pm(2^w-1)\}$，$m$表示wNAF链中的非零数字个数，$\lambda_i$是wNAF链中每个非零数$k_i$所在的位置. 实现标量乘时，通过调用已预计算并存储的点$\{\pm P,\pm3P,\cdots,\pm(2^w-1)P\}$，其中$w$是窗口宽度，可极大地减少运算时的计算量. wNAF的具体实现如算法1所示.

该算法整个过程分成3个阶段：wNAF序列生成阶段（步骤1到步骤12）、预计算阶段（步骤13）和算数运算阶段（算数运算包括点加运算和倍点运算，步骤14到步骤20）. wNAF是通过降低标量k的汉明权重、减少点加运算来降低算法的复杂度. 假设在wNAF链$\{k_i\}$中有非零数字m个，第i个非零数字是k_i，λ_i是wNAF链中每个k_i所在的位置，则有$\lambda_i-\lambda_{i-1}\geqslant w+1$，显然，wNAF的非零概率为$1/(w+1)$，预计算阶段和算数运算阶段的成本

算法1 带符号的wNAF标量乘算法

输入:标量k,基点P,窗口宽度w
输出:标量乘结果Q
1)$i=0$
2) While $k>0$ do
3) If $k \bmod 2=1$ then $e_i=k \bmod 2^{w+1}$;
4) If $e_i\geqslant 2^w$ then $e_i\Leftarrow e_i-2^{w+1}$;
5) End if
6) $k\Leftarrow k-e_i$;
7) Else
8) $e_i=0$;
9) End if
10) $k\Leftarrow k/2$; $i\Leftarrow i+1$;
11)End while
12)return $\{k_{i-1},k_{i-2},\cdots,k_1,k_0\}$
13)预计算:$P_i=iP, i\in\{1,3,5,\cdots,2^w-1\}$
14)for i from $b-1$ to 0 do
15) $Q\Leftarrow 2Q$;
16) If $k_i\neq0$ then
17) If $k_i>0$ then $Q\Leftarrow Q+P_{ki}$;
18) If $k_i<0$ then $Q\Leftarrow Q-P_{ki}$;
19) End
20)End

分别为$2^{w-1}\mathrm{A}+\mathrm{D}$和$n\mathrm{D}+n/(w+1)\mathrm{A}$，所以标量乘计算量为$(n+1)\mathrm{D}+\left(2^{w-1}+n/(w+1)\right)\mathrm{A}$，其中A是点加运算，D是倍点运算，且有$1\mathrm{A}=12\mathrm{M}+4\mathrm{S}$，$1\mathrm{D}=4\mathrm{M}+6\mathrm{S}$[14]以及$0.8\mathrm{M}=1\mathrm{S}$[15]（M是模乘运算，S是模平方运算）. 然而，wNAF标量乘算法也存在一定的问题：当窗口宽度增大的时候，预计算点的个数呈指数增长，大大增加了预计算量和所需的寄存器数量.

2.2 加法链标量乘算法

文献[16]进行加法链标量乘运算时，首先构造一条加法链，定义为一个序列$v=(v_1,v_2,\cdots,v_l)$，其中，$v_1=1,v_2=2,v_i=v_{i-1}+v_{i-2}(3\leqslant i\leqslant l)$，$l$是加法链的长度，标量$k$由加法链序列中若干个$v_i$组成，表示成$k_i=\sum_1^l v_i$；其次，将$v_i$与基点$P$相乘并存储；最后，只需点加操作即可得到标量乘的最终结果. 加法链标量乘算法如算法2所示.

算法2 加法链标量乘算法

输入:标量k,基点P
输出:标量乘结果Q
1)通过贪心算法得到k的加法链$(v_1,v_2,\cdots,v_l)$
2)$Q\Leftarrow0$;
3)for i from 1 to l do
4) $Q\Leftarrow Q+v_i$;
5)end
6)return Q

如算法2所示,步骤1是得到k的加法链的过程,贪心算法用于每次在加法链中寻找最接近k_i的元素v_i,然后再利用k_i和v_i的差值在加法链中找到最接近该差值的v_i,进而可以得到标量k的加法链;步骤2~6是计算标量乘的过程,这部分只需要进行点加操作而不需要进行倍点操作即可得到标量乘的结果. 此外,由于生成k的加法链需要用到贪心算法,随着k位数的增加,创建加法链所用的时间也会越长.

3　低计算复杂度的wNAF算法设计

3.1　基于$2^n p$预计算

众所周知,wNAF算法在预计算阶段将$\{\pm P, \pm 3P, \cdots, \pm(2^w-1)P\}$存储起来用于加速标量乘的计算,但随着窗口宽度$w$的增大,预计算点的个数呈指数增长,使得该算法不适用于窗口宽度较大的情况. 本部分针对该问题,对wNAF算法进行改进,减少预计算点以降低wNAF算法的标量乘计算复杂度.

由于倍点运算相较于点加运算、3倍点运算和5倍点运算需要较少的模乘次数,同时由于2^n构成的集合中元素比3^n,5^n构成的集合中元素更加密集,2^n与奇数之间的差值更小,有利于构造差值的加法链. 此外,考虑到在进行算数运算时也需要进行倍点运算,倍点运算结构可以通过控制器控制,反复利用,节省资源. 本文的思路是:在预计算阶段用2^n代替生成的wNAF链中的奇数,只预计算并存储$2^n P$. 这种替换的优势在于能够大大减少预计算点的个数且预计算点只需要通过倍点运算即可得到. 例如:当窗口为5时,wNAF算法需要存储$\{P, 3P, \cdots, 31P\}$,共计16个点;而本文算法只需要存储$\{2^0P, 2^1P, \cdots, 2^5P\}$,共计6个点. 表1列出了在不同窗口宽度下所需预计算的点及个数.

表1　预计算点及个数

窗口宽度	预计算点	预计算点的个数
5	$\{2^0P, 2^1P, 2^2P, \cdots, 2^5P\}$	6
6	$\{2^0P, 2^1P, 2^2P, \cdots, 2^6P\}$	7
…	…	…
n	$\{2^0P, 2^1P, 2^2P, \cdots, 2^{n+1}P\}$	n+1

3.2　基于$2^n p$的加法链的差值补偿

由3.1节中可知,用$\{2^0P, 2^1P, \cdots, 2^nP\}$代替$\{P, 3P, \cdots, (2^w-1)P\}$会产生差值. 例如当窗口宽度为$w$=5时,本文预计算点为$\{2^0P, 2^1P, \cdots, 2^5P\}$,wNAF算法预计算点为$\{P, 3P, \cdots, 31P\}$,若wNAF链中存在非零数17时,$2^4P$与$17P$最接近,但存在差值$P$;当wNAF链中存在非零数21时,$2^4P$与$21P$最接近,存在差值$5P$;当wNAF链中存在非零数23时,$2^4P$与$23P$最接近,存在差值$7P$. 不同窗口宽度下的预计算点与wNAF预计算点之间可能存在的差值如表2所示.

由表2可知,本文预计算与wNAF预计算之间的差值为奇数,且随着窗口宽度的增加,该差值的最大值也不断增加,当窗口宽度在5~11范围内时,差值的最大值为$511P$.

表2　本文预计算点与wNAF预计算点之间的差值

窗口宽度	预计算点之间的差值
5	$\{1P, 3P, 5P, 7P\}$
6	$\{1P, 3P, 5P, 7P, 9P, \cdots, 15P\}$
…	…
n	$\{1P, 3P, 5P, 7P, 9P, \cdots, (2^{n-2}-1)P\}$

由于任意一个标量k都可以用二进制表示并且在预计算阶段已经完成了对2^nP的计算和存储,因此2^nP与$(2h-1)P$之间的差值Δ可以通过构造2^nP加法链来补偿. 由第2.2节可知,加法链是将一个大数拆分成若干个小数,在计算时,通过若干个小数相加减得到结果. 因此,差值Δ可以基于加法链的思想由多个2^nP相加减得到,即$\Delta=\sum 2^nP$. 例如:在窗口宽度为w=5时,已知wNAF链中存在非零数13,且已有预计算点$\{2^0P, 2^1P, 2^2P, 2^3P, 2^4P, 2^5P\}$找到与$13P$最接近的$2^4P$,$2^4P$与$13P$之间的差值为$3P$,$3P$则可以表示为$2^2P-2^0P$,最终$13P=2^4P-2^2P+2^0P$. 此外,根据表2,当窗口宽度为11时,构造的加法链最长,需要4次点加运算,例如:差值为$299P$时,$299P$的加法链可以构造为$299P=2^8P+2^5P+2^3P+2^1P+2^0P$.

3.3　改进的wNAF标量乘算法

根据第3.1节和第3.2节,可以总结出本文的算法:通过算法1生成wNAF链,用2^n代替wNAF链中的奇数,在预计算阶段,将2^nP预计算出来,在算数运算阶段,通过搜索k链中的非零数值k_i,2^nP与k_iP之间的差值通过构建微小的2^nP加法链来实现,最后再通过一系列的点加运算和倍点运算,得到标量乘的最终结果. 改进的wNAF标量乘算法如算法3所示.

算法3中,步骤1~2是预计算部分,根据窗口的大小预计算2^nP;步骤3~37是算数运算阶段. 步骤8~10用于寻找与wNAF链中非零数值最接近的2^nP,此时,在查找k链时会分成两种情况,一种是k链中的非零数值小于0的情况,执行步骤11~22;另一种是k链中的非零数值大于0的情况,执行步骤23~34. 此外,步骤11~12以及步骤23~24用于得到2^nP和$(2h-1)P$之间的差值Δ;步骤14~19以及步骤26~31是构造差值Δ的加法链的过程,先找到与Δ接近的2^nP,计算它们的差值,再找到与该差值接近的2^nP,以此类推,迭代找到构成Δ的2^nP加法链add1,最终得到标量乘结果.

算法3 改进的wNAF的标量乘算法

输入:标量k,基点P,窗口宽度w
输出:标量乘结果Q
1)预计算:
2) $P_i=2^iP, i\in\{0,1,2,\cdots,w\}$
3)标量乘计算:
4) $Q\Leftarrow 0$; add1$\Leftarrow 0$;
5) for i from $b-1$ to 0 do
6) $Q\Leftarrow 2Q$;
7) add1$\Leftarrow 0$;
8) If $k_i\neq 0$ then
9) $s\Leftarrow$ Findnearst($|k_iP|$) //找到最接近$|k_iP|$的2^nP;
10) add$\Leftarrow$ Findnearst($|k_iP|$);
11) If $k_i<0$
12) $a\Leftarrow k_iP+s$;
13) $j\Leftarrow 0$;
14) While($a\neq 0$)
15) $t_j\Leftarrow$ Findnearst($|a|$);
16) $a\Leftarrow |a|-t_j$
17) $j\Leftarrow j+1$;
18) End
19) add1$\Leftarrow\sum t_j$;
20) If $a\geqslant 0$ then add$\Leftarrow$ add1 − add;
21) Else add$\Leftarrow -$add − add1;
22) End
23) If $k_i>0$
24) $a\Leftarrow k_iP-s$;
25) $j\Leftarrow 0$;
26) While($a\neq 0$)
27) $t_j\Leftarrow$ Findnearst($|a|$);
28) $a\Leftarrow |a|-t_j$
29) $j\Leftarrow j+1$;
30) End
31) add1$\Leftarrow\sum t_j$;
32) If $a\geqslant 0$ then add$\Leftarrow$ add − add1;
33) Else add$\Leftarrow -$add − add1
34) End
35) $Q\Leftarrow Q+$add;
36) End
37) End

3.4 算法计算复杂度分析

结合第3.1节、第3.2节、第3.3节得到的本文算法,下面对算法的复杂度进行分析.

预计算部分:计算2^nP时,窗口宽度为w,则需要进行w次倍点,例如,当$w=5$时,预计算点为$\{2^0P,2^1P,\cdots,2^5P\}$,需要进行5次倍点,所以预计算的成本为

$$\text{预计算成本}=wD \tag{1}$$

算数运算部分:设wNAF链长为n,则需要进行n次倍点操作,由文献[12]可知wNAF的汉明重量为$1/(w+1)$,则k链中一共有$\lceil n/(w+1)\rceil$个非零数字,需要进行$\lceil n/(w+1)\rceil$次点加,当窗口宽度$w=5$时,在构建2^nP与$(2h-1)P$差值的加法链时,需要额外进行一次点加运算的点的个数占k链中非零数字的1/4,需要额外进行两次点加运算的点的个数占k链中非零数字的3/4,因此需要额外进行$\left\lceil 7/4\lceil n/(w+1)\rceil\right\rceil$次点加运算,一共需要进行$\left\lceil 11/4\lceil n/(w+1)\rceil\right\rceil$次点加运算,并且窗口宽度每增加1,点加次数多增加$\left\lceil 3/8\lceil n/(w+1)\rceil\right\rceil$,算数运算阶段一共所需的成本为

$$\text{算数运算成本}=\left\lceil\frac{22+3(w-5)}{8}\left\lceil\frac{n}{w+1}\right\rceil\right\rceil\text{A}+n\text{D} \tag{2}$$

标量乘部分:标量乘成本=预计算成本+算数运算成本,即

$$\text{标量乘成本}=(w+n)\text{D}+\left\lceil\frac{22+3(w-5)}{8}\left\lceil\frac{n}{w+1}\right\rceil\right\rceil\text{A} \tag{3}$$

通过将1A=12M+4S,1D=4M+6S[14]以及0.8M=1S[15]代入到式(1)、式(2)和式(3)可以得到以模乘次数为标准的预计算计算复杂度、算数运算计算复杂度和标量乘计算复杂度,结果如图1所示.

图1表示了以模乘次数为标准的本文算法在不同窗口宽度、不同曲线下的计算复杂度曲线. 计算复杂度包括预计算计算复杂度、算数运算计算复杂度和标量乘计算复杂度;曲线包括P_{256},P_{384}和P_{521},窗口宽度范围为2~12. 由图1可以看出,对于所有曲线,窗口宽度为11时,标量乘计算复杂度最小;窗口宽度小于5时,标量乘计算复杂度要高于窗口宽度为5时. 而当窗口宽度大于11时,由于2^nP与$(2h-1)P$之间的差值过大,不利于构建基于2^nP的加法链. 根据以上结果,本文后面对算法的比较与分析都将窗口宽度限制在5~11的范围内.

4 算法计算复杂度比较

4.1 预计算计算复杂度比较

为了更直观、更清晰地观察本文算法在预计算方面有优势,将提出的算法与目前研究比较多的wNAF算法[17]、滑动窗口非相邻形式算法(swNAF算法)[18]和基于素数预计算的算法[9]进行了比较,比较结果如表3所示.

由表3可以看出,相较于wNAF算法、swNAF算法和基于素数预计算的算法,本文算法在窗口宽度为5~11时预计算点的个数减少,而在窗口宽度为11时预计算点个数减少最多,分别减少了1 013个、330个和204个,减少的百分比分别为98.83%,96.49%和94.42%. 此外,当窗口宽度增加时,本文算法的预计算点的数量减少的百分比也随之增加,说明本算法比其他算法更适用于窗口宽度较大的情况.

表4显示了当窗口宽度为5~11时预计算点所需的

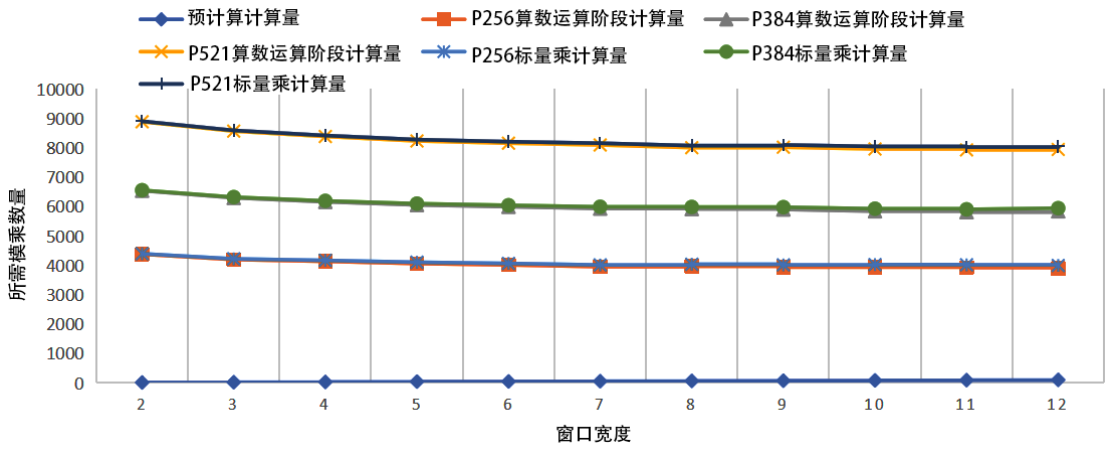

图1　低复杂度的改进wNAF标量乘算法在不同窗口下的计算复杂度分析

模乘次数的比较,其中,倍点运算(用D表示)根据1D=4M+6S以及0.8M=1S进行转换. 从表4可以看出,相较于wNAF算法、swNAF算法和基于素数预计算的算法,在窗口宽度为5时,本文算法在预计算的模乘次数分别减少了81.95%,70.19%和74.19%,在窗口宽度为11时,预计算的模乘次数分别减少了99.38%,99.07%和97.83%. 由此可知,在窗口宽度5~11时,本文算法的预计算复杂度最低. 此外,随着窗口宽度的增加,本文算法预计算所需模乘次数减少的百分比也随之增加,也说明了本文算法更适用于较大窗口.

表3　预计算点比较

窗口宽度	本文算法	wNAF算法	swNAF算法	基于素数预计算算法	与wNAF算法相比减少的百分比	与swNAF算法相比减少的百分比	与素数预计算相比减少的百分比
5	6	16	6	11	62.5%	0.0%	45.45%
6	7	32	11	18	78.13%	36.36%	61.11%
7	8	64	22	31	87.5%	63.63%	74.19%
8	9	128	43	54	92.97%	79.1%	83.33%
9	10	256	86	97	96.09%	88.37%	89.69%
10	11	512	171	140	97.85%	93.57%	92.14%
11	12	1 024	342	215	98.83%	96.49%	94.42%

表4　预计算点所需的模乘次数比较

窗口宽度	预计算所需模乘次数				模乘次数减少比率/%		
	本文算法	wNAF算法	swNAF算法	基于素数预计算算法	wNAF算法	swNAF算法	基于素数预计算算法
5	48	266	161	186	81.95	70.19	74.19
6	53	522	313	266	89.85	83.07	80.08
7	62	1 304	648	522	95.25	90.43	88.12
8	71	2 058	1 286	842	96.55	94.48	91.57
9	80	4 106	2 593	1 386	98.05	96.91	94.23
10	88	7 792	5 177	2 608	98.87	98.30	96.63
11	97	15 574	10 376	4 463	99.38	99.07	97.83

4.2　标量乘计算复杂度比较

标量乘计算复杂度包括了预计算计算复杂度以及算数运算计算复杂度两部分,在第4.1节已经对预计算计算复杂度进行了对比,表5显示了当n为256,384,521时,4种算法的算数运算计算复杂度和标量乘计算复杂度对比. 由表5可以看出,本文算法在算数运算部分的计算复杂度与其他3种算法相当,但窗口宽度增加时,本文算法的预计算优势越明显. 并且随着窗口宽度增大,在相同位数下,有符号wNAF链中非零个数相比无符号wNAF链中更少,这也进一步减少点加次数,从而降低标量乘的计算复杂度. 此外,在窗口宽度较大时$w=11$时,本文算法的标量乘计算复杂度相较于wNAF算法、swNAF算法和基于素数预计算算法降低了最多,分别为78.23%,68.94%和43.63%.

表5 4种算法总体计算复杂度对比

算法	标量k的长度	窗口宽度	算数运算阶段所需模乘次数	标量乘所需模乘次数	标量乘减少比率/%	算法	标量k的长度	窗口宽度	算术运算阶段所需模乘次数	标量乘所需模乘次数	标量乘减少比率/%
wNAF算法	256	8	3 015	5 073	20.34	基于素数预计算算法	256	8	3 110	3 952	−2.25
	384		4 523	6 581	9.0%		384		4 660	5 502	−8.83
	521		6 136	8 194	1.44		521		6 320	7 162	−11.37
	256	9	2 764	6 870	41.48		256	9	3 025	4 411	8.86
	384		4 249	8 355	28.40		384		4 607	5 993	0.18
	521		5 838	9 944	18.64		521		6 317	7 703	−5.02
	256	10	2 113	9 905	59.33		256	10	2 688	5 296	23.94
	384		3 579	11 371	47.85		384		4 032	6 640	10.69
	521		5 148	12 940	37.81		521		5 466	8 074	0.32
	256	11	2 901	18 475	78.23		256	11	2 672	7 135	43.63
	384		4 352	19 926	70.35		384		4 000	8 463	30.19
	521		5 905	21 479	62.63		521		5 418	9 881	18.77
swNAF算法	256	8	2 678	3 964	−1.91	本文算法	256	8	3 971	4 041	
	384		4 018	5 304	−11.42		384		5 918	5 988	
	521		5 436	6 722	−16.77		521		8 005	8 076	
	256	9	2 633	5 226	23.08		256	9	3 940	4 020	
	384		3 957	6 550	8.67		384		5 903	5 982	
	521		5 360	7 953	−1.72		521		8 020	8 090	
	256	10	2 603	7 780	48.23		256	10	3 940	4 028	
	384		3 896	9 073	34.64		384		5 842	5 930	
	521		5 284	10 461	23.07		521		7 960	8 048	
	256	11	2 572	12 948	68.94		256	11	3 925	4 022	
	384		3 865	14 241	58.51		384		5 812	5 908	
	521		5 238	15 614	48.60		521		7 929	8 026	

5 结论

本文提出了在有符号wNAF算法的基础上，在预计算阶段采用2^nP替换$(2h-1)P$，替换后的差值采用2^nP构造的加法链进行补偿，该方法有效降低了预计算复杂度，并且只需要少量的寄存器即可完成预计算点的存储，进而解决了有预计算算法不适用于窗口很大的问题. 与现有的算法相比，预计算所需模乘数相较于wNAF算法、swNAF算法和基于素数预计算算法最多减少了99.38%，99.07%和97.83%，标量乘的计算复杂度分别降低了78.23%，68.94%和43.63%.

参考文献

[1] 张亮. 改进的基于整数拆分形式标量乘快速算法[J]. 中国电子科学研究院学报, 2016, 11(5): 490-494.
ZHANG L. Improved fast scalar multiplication algorithm based on signed integer splitting form[J]. Journal of China Academy of Electronics and Information Technology, 2016, 11(5): 490-494. (in Chinese)

[2] ISLAM M M, HOSSAIN M S, HASAN M K, et al. FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field[J]. IEEE Access, 2019, 7: 178811-178826.

[3] KHLEBORODOV D. Fast elliptic curve point multiplication based on binary and binary non-adjacent scalar form methods[J]. Advances in Computational Mathematics, 2018, 44(4): 1275-1293.

[4] 徐明, 史量. 基于伪四维投射坐标的多基链标量乘法[J]. 通信学报, 2018, 39(5): 74-84.
XU M, SHI L. Pseudo 4D projective coordinate-based multi-base scalar multiplication[J]. Journal on Communications, 2018, 39(5): 74-84. (in Chinese)

[5] OKEYA K, SCHMIDT-SAMOA K, SPAHN C, et al. Signed binary representations revisited[C]//Annual International Cryptology Conference. Berlin: Springer, 2004: 123-139.

[6] 李忠, 彭代渊. 基于滑动窗口技术的快速标量乘法[J]. 计

算机科学, 2012, 39(6A): 54-56, 64.
LI Z, PENG D Y. Fast scalar multiplication based on sliding window technology[J]. Computer Science, 2012, 39 (6A): 54-56, 64. (in Chinese)

[7] ALIMORADI R, ARKIAN H R, RAZAVIAN S M J, et al. Scalar multiplication in elliptic curve libraries[J]. Journal of Discrete Mathematical Sciences and Cryptography, 2021, 24(3): 657-666.

[8] LIU S G, QI G L, WANG X A. Fast and secure elliptic curve scalar multiplication algorithm based on a kind of deformed fibonacci-type series[C]//2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). Xi´an: IEEE, 2015: 398-402.

[9] HUANG H, NA N, XING L, et al. An improved wNAF scalar-multiplication algorithm with low computational complexity by using prime precomputation[J]. IEEE Access, 2021, 9: 31546-31552.

[10] 史量, 徐明 . DWNAF: 带门限的动态窗口的NAF标量乘法[J]. 计算机科学, 2017, 44(10): 159-164.
SHI L, XU M. DWNAF: A dynamic window NAF scalar multiplication with threshold[J]. Computer Science, 2017, 44(10): 159-164. (in Chinese)

[11] LIU S G, SUN X J. A fast scalar multiplication algorithm based on alternate-zeckendorf representation[J]. International Journal of Network Security, 2018, 20(5): 931-937.

[12] Khleborodov D. Fast elliptic curve point multiplication based on window Non-Adjacent Form method[J]. Applied Mathematics and Computation, 2018, 334: 41-59.

[13] ZHANG Z B, WU L J, MU Z L, et al. A novel template attack on wNAF algorithm of ECC[C]//2014 Tenth International Conference on Computational Intelligence and Security. Beijing: IEEE, 2014: 671-675.

[14] DANGER J L, GUILLEY S, HOOGVORST P, et al. Improving the Big Mac Attack on Elliptic Curve Cryptography[M]. Berlin: Springer, 2016: 374- 386.

[15] DOU Y Q, WENG J, MA C G, et al. Fast scalar multiplication algorithm using constrained triple-base number system and its applications[C]//2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications(BWCCA). Zhengzhou: IEEE, 2015: 426-431.

[16] CHUDNOVSKY D V, CHUDNOVSKY G V. Sequences of numbers generated by addition in formal groups and new primality and factorization tests[J]. Advances in Applied Mathematics, 1986, 7(4): 385-434.

[17] WANG W B, FAN S Q. Attacking OpenSSL ECDSA with a small amount of side-channel information[J]. Science China Information Sciences, 2017, 61(3): 1-14.

[18] RASMI M, SOKHON A A, SH M, et al. A survey on single scalar point multiplication algorithms for Elliptic curves over Prime fields[J]. IOSR Journal of Computer Enginnering, 2016, 18(2): 31-47.

作者简介

赵石磊　男,1979年出生,黑龙江肇源人. 博士,硕士生导师. 主要研究方向为信息安全、可重构计算、IC设计、VLSI数字信号处理.
E-mail: zhaosl@hrbust.edu.cn

杨晓秋　女,1997年出生,黑龙江省鹤岗人. 现为哈尔滨理工大学计算机科学与技术学院硕士生. 主要研究方向为信息安全、密码算法.
E-mail: yangxq2022@163.com

刘志伟　男,1987年出生,黑龙江哈尔滨人. 哈尔滨理工大学讲师、博士生. 主要研究方向为可重构计算、高速密码算法、并行加密技术、密码芯片的安全设计等.
E-mail: zwliu@hrbust.edu.cn

于　斌　男,1984年出生,黑龙江饶河人. 硕士,哈尔滨理工大学讲师. 主要研究方向为密码算法、密码芯片设计和数字集成电路设计等.
E-mail: yubin@hrbust.edu.cn

黄　海(通讯作者)　男,1982年出生,内蒙古巴彦淖尔人. 博士,教授,博士生导师,CCF高级会员. 主要研究方向为信息安全、可重构计算、集成电路设计.
E-mail: ic@hrbust.edu.cn