软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(10):4542−4557 [doi: 10.13328/j.cnki.jos.007301] [CSTR: 32375.14.jos.007301]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于两种新标量表示的安全高效标量乘算法*

程 石, 胡 志, 陶 铮

(中南大学 数学与统计学院, 湖南 长沙 410083)

通信作者: 胡志, E-mail: huzhi_math@csu.edu.cn

摘 要: 标量乘法是传统椭圆曲线密码 (ECC) 的核心运算. 标量表示决定了标量乘法算法中的迭代方式, 进而直接影响算法的安全性和效率. 提出两种新的标量表示算法: 一种称为规则窗口非相邻算法 (ordered window width non-adjacent form, OWNAF), 它将传统的窗口非相邻算法与随机密钥分割处理相结合, 在提升计算效率的同时可以抵抗能量分析攻击; 另一种称为窗口联合正则形式 (window joint regular form, wJRF), 它由传统的联合正则形式改进而来, 适用于多标量乘算法, 与已有算法相比, 在减少基础计算量的同时有着更好的安全性.

关键词: 标量乘算法; 侧信道攻击; 窗口非相邻形式; 联合正则形式

中图法分类号: TP301

中文引用格式: 程石, 胡志, 陶铮. 基于两种新标量表示的安全高效标量乘算法. 软件学报, 2025, 36(10): 4542–4557. http://www.jos.org.cn/1000-9825/7301.htm

英文引用格式: Cheng S, Hu Z, Tao Z. Safe and Efficient Scalar Multiplication Algorithms Based on Two New Scalar Representations. Ruan Jian Xue Bao/Journal of Software, 2025, 36(10): 4542–4557 (in Chinese). http://www.jos.org.cn/1000-9825/7301.htm

Safe and Efficient Scalar Multiplication Algorithms Based on Two New Scalar Representations

CHENG Shi, HU Zhi, TAO Zheng

(School of Mathematics and Statistics, Central South University, Changsha 410083, China)

Abstract: Scalar multiplication is the core operation in traditional elliptic curve cryptography (ECC). Scalar representations determine the iterations in scalar multiplication algorithms, which directly affect the security and efficiency of the algorithms. This study proposes two new scalar representation algorithms. One algorithm is ordered window width non-adjacent form (OWNAF) which combines traditional window non-adjacent form with random key segmentation and can resist energy analysis attacks while yielding better efficiency. The other is called window joint regular form (wJRF), which is improved from the traditional joint regular form. The wJRF algorithm is applicable to multi-scalar multiplication algorithms, which can reduce computational costs and ensure sound security compared with the existing algorithms.

Key words: scalar multiplication algorithm; side channel attack; window non-adjacent form; joint regular form

椭圆曲线密码 (ECC) 算法作为一种非对称密码体制算法, 自 1985 年由 Millerl[1]和 Koblitz[2]分别独立地提出以来, 已被广泛地应用于保护网络空间安全. 特别地, 由于 ECC 具有较高的安全等级和较小的密钥长度等特点, 十分适合应用在智能卡等存储资源受限的环境. 椭圆曲线标量乘算法是椭圆曲线密码的核心运算, 其实现直接影响密码系统的效率和安全性. 随着侧信道攻击技术的发展, 传统的标量乘算法如 wNAF 算法[3]等逐渐显露出安全隐患, 会在使用过程中暴露一定的侧信道信息, 被攻击者加以利用. 因此, 实现高效安全的椭圆曲线标量乘算法成为椭圆曲线密码中的重要研究问题.

椭圆曲线标量乘运算通常表示为 $[k]P = P + P + \ldots + P$, 其中 k 是一个整数, P 是椭圆曲线上的点. 标量乘运算又被称为点乘运算, 由点加和倍点两种运算交替实现. 而点加、倍点运算最终由有限域的各种模运算完成. 因此,

* 基金项目: 国家自然科学基金 (61972420); 湖南省自然科学基金 (2020JJ3050)
收稿时间: 2024-06-06; 修改时间: 2024-08-27; 采用时间: 2024-10-08; jos 在线出版时间: 2025-01-24
CNKI 网络首发时间: 2025-01-26

椭圆曲线点乘运算的效率和安全性被专门研究.

由于非相邻型表示 (non adjacent form, NAF) 相较二进制表示能够减少标量的汉明重量, 因此基于窗口的 NAF 算法能够显著提高标量乘算法的效率. Okeya 等人[4], Anagreh 等人[5]和 Hu 等人[6]通过对标量 k 进行了 NAF 编码, 降低了 k 的汉明重量, 进一步提高了算法效率.

随着侧信道攻击的不断发展, 诸如简单功耗分析 (simple power analysis, SPA) 攻击[7], 差分能量分析 (differential power analysis, DPA) 攻击[7]等对密码算法造成了巨大的威胁, wNAF 算法的安全隐患也日渐显露, 使用时可能存在侧信道泄漏并被攻击者加以利用. 2022 年, Cao 等人[8]通过一种扩展的密钥恢复攻击了基于 wNAF 的 SM2 签名算法; 2023 年, Ma 等人[9]通过 Flush+Reload 攻击来破解基于 wNAF 的 ECDSA 算法.

因此, 许多学者从兼顾安全与性能两方面对椭圆曲线标量乘算法进行研究, 其中针对 NAF 编码的安全高效算法有着较多方案. 张涛等人[10]提出了一种基于改进 wNAF 的标量乘算法 RWNAF (refined width-w NAF), 将原先 wNAF 算法生成的 NAF 序列变得规则, 并借助掩码, 实现了抗 SPA, DPA 等攻击的安全标量乘算法; Yao 等人[11]在文献 [10] 的基础上通过采用碎片窗口 NAF 的处理, 所实现的 FWNAF (fractional width-w NAF) 算法相较于 RWNAF 算法对存储资源的利用率更高; 史量等人[12]在此基础上通过采用带门限的动态窗口方法, 所实现的 DWNAF (dynamic width-w NAF) 算法大大减少了预计算开销, 但生成算法的时间消耗相较 RWNAF 和 FWNAF 算法更大, 适用于水下声传感器等信道时延较大的应用场景. 目前已有的工作中, 改进 NAF 编码的安全高效算法大部分与 RWNAF 算法相关, RWNAF 算法中采用随机掩码的方式来规避 DPA 攻击, 存在被 Flush+Reload 攻击进而获取密钥等隐私信息的可能. 如果在 RWNAF 算法的基础上进行适当的优化, 就能得到更高效, 更安全的算法.

多标量乘算法也是椭圆曲线密码体制的重要组成部分, 通常表示为 $[k]P+[l]Q$, 其中 k 和 l 表示两个整数, P, Q 表示椭圆曲线上两点. 同样由于侧信道攻击的影响, 许多密码学者研究安全高效的多标量乘法. Lee[13]首先提出了一种抗 SPA 的多标量乘法, Ciet 等人[14]和 Liu 等人[15]也给出了不同格式的抗 SPA 的多标量乘法. 陈厚友等人[16]在此基础上提出了一种改进算法, 有效提升了多标量乘算法的效率. Akishita 等人[17]提出了一种名为联合正则形式 (joint regular form, JRF) 的算法, 并基于此实现了抗 SPA 的多标量乘法. 但是目前的主流多标量乘算法大多只针对 SPA 攻击而没有考虑 DPA, RPA, ZPA 攻击等, 在安全性上还有所欠缺.

本文主要工作如下.

(1) 提出了一种新的 NAF 生成算法 OWNAF (ordered window width non-adjacent form), 并通过生成的 OWNAF 序列和随机密钥分割的方法实现了新的高效标量乘算法, 在增加一定存储开销的基础上提升了计算效率, 同时能够有效地抵抗 SPA, DPA 等侧信道攻击, 并对 Flush+Reload 攻击方法产生了一定的干扰.

(2) 从 JRF 算法生成序列的特点出发, 通过对标量乘算法的适当变换, 给出基于 JRF 的安全标量乘算法, 在保证安全的前提下存储开销较小, 非常适合智能卡等存储资源受限的环境, 并且更进一步将 JRF 推广为 wJRF, 并基于 wJRF 提出一种安全高效多标量乘算法, 在减少计算量的同时能有效抵抗 DPA, RPA, ZPA 等侧信道攻击, 提高了安全性.

本文第 1 节主要介绍所需要的预备知识. 第 2 节主要介绍 OWNAF 的生成算法以及相应的标量乘算法. 第 3 节描述 wJRF 标量形式和基于 wJRF 的多标量乘法. 最后总结全文.

1 基础知识

本文所提方法主要基于 RWNAF 算法和联合正则形式算法进行优化, 下面就相关概念和基本知识予以介绍. 相关符号说明见后文表 1.

1.1 RWNAF 标量乘法

非相邻型表示 (non-adjacent form, NAF) 是带符号数的一种表示方法, 顾名思义, NAF 表示中的非 0 值不能相邻, 取值为 $\{0,1,-1\}$, 引入窗口算法即为 wNAF 形式, 任何连续的 w 个位中最多有 1 位非 0. 将一个标量 k 表示为 $NAF_w(k)=\sum_{i=0}^{len-1}k_i2^i$ 的形式, 其中非 0 的 k_i 均为奇数且 $|k_i|<2^{w-1}$.

表 1 符号说明

符号	含义
k	标量乘算法中的标量
P,Q	椭圆曲线上的点
$[k]P$	椭圆曲线上点 P 的 k 倍点
w	窗口宽度
$NAF_{\mathrm{w}}(k)$	标量 k 的wNAF表示
$NAF_{\mathrm{rw}}(k)$	标量 k 的RWNAF表示
$NAF_{\mathrm{ow}}(k)$	标量 k 的OWNAF表示
A	点加运算
D	倍点运算
$sgn()$	符号函数
$MSB_w(k)$	取标量 k 的二进制表示的第 w 位
$\widetilde{\lambda}$	$\widetilde{\lambda}=\lambda-2^{w-1} \bmod 2^{w-1}$

计算数 k 的 wNAF 形式的算法如算法 1 所示.

算法 1. 标量 k 的 wNAF 编码算法[3].

输入: 标量 k, 窗口宽度 w;
输出: $NAF_{\mathrm{w}}(k)$.

1. $i \leftarrow 0$;
2. **while** $k \geqslant 0$ **do**
3. **if** k 是奇数 **then**
4. $k_i \leftarrow k \bmod 2^w$;
5. **if** $k_i \geqslant 2^{w-1}$ **then**
6. $k_i = k_i - 2^w$;
7. **end if**
8. $k \leftarrow k - k_i$;
9. **else**
10. $k_i \leftarrow 0$;
11. **end if**
12. $k \leftarrow k/2$, $i \leftarrow i+1$;
13. **end while**
14. **return** $NAF_{\mathrm{w}}(k) = (k_{i-1}, k_{i-2}, \ldots, k_1, k_0)$.

由算法 1 计算得到的 $NAF_{\mathrm{w}}(k)$ 的位数最大为 $n+1$, n 为 k 的二进制表示位数. 在宽度为 w 的 $NAF_{\mathrm{w}}(k)$ 中非 0 数字的密度平均为 $1/(w+1)$. 计算 wNAF 形式 k 的标量乘算法如算法 2 所示.

算法 2. wNAF 标量乘算法[3].

输入: 标量 k, 椭圆曲线上一点 P, 窗口宽度 w;
输出: $[k]P$.

1. 计算 $NAF_{\mathrm{w}}(k) = \sum_{i=0}^{len-1} k_i 2^i$; /* 算法 1 */
2. 预计算 $P_i = [i]P$, $i \in \{1,3,\ldots,2^{w-1}-1\}$;

3. $Q \leftarrow O$;
4. **for** $i = len - 1$ **down to** 0 **do**
5. $\quad Q \leftarrow [2]Q$;
6. $\quad Q \leftarrow Q + [sgn(k_i)]P_{k_i}$;
7. **end for**
8. **return** Q.

算法 2 中存在以下安全隐患: 当第 6 步中 $k_i = 0$ 时, 不进行点加运算; 而当 $k_i \neq 0$ 时, 进行一次点加运算, 这就导致了 $k_i = 0$ 和 $k_i \neq 0$ 时运算出现了不同的分支, 耗时不同. 因此, 攻击者可以通过诸如 Flush+Reload 攻击等, 对时间进行检测, 判断 k_i 是否等于 0. 记点加运算为 A, 倍点运算为 D, 则对 wNAF 算法进行一次攻击后就可以获得类似 "$DADDADDDA\ldots$"的点加和倍点信息链, 通过对这种不规则的信息链进行分析, 就能获取标量 k 的 wNAF 表示法中所有的非 0 元素 $\{k_i\}_{i=1}^{n}$ 以及 k_i 对应的原来角标值 i, 这样便可通过文献 [8,9] 中的格方法将其转换成 EHNP 问题进行求解, 多次攻击后即可获取密钥相关的私密信息.

张涛等人在文献 [10] 中提出了一种基于改进 wNAF 的标量乘算法 RWNAF, 将原先 wNAF 算法生成的 NAF 序列变得规则, 并借助随机掩码, 实现了抗 SPA, DPA 等攻击的安全标量乘算法, 其中 RWNAF 编码算法如算法 3 所示.

算法 3. 标量 k 的 RWNAF 编码算法[10].

输入: 标量 k, 窗口宽度 w;
输出: $NAF_{\mathrm{rw}}(k)$.

1. $r = 0$, $i = 0$, $r_0 = w$;
2. **if** k 为偶数 **then**
3. $\quad k = k + 1$;
4. **end if**
5. **while** $k > 1$ **do**
6. $\quad u[i] = (k \bmod 2^{w+1}) - 2^w$;
7. $\quad k = (k - u[i])/2^{r_i}$;
8. $\quad k_w[r + r_i - 1] = 0, \ldots, k_w[r + 1] = 0, k_w[r] = u[i]$;
9. $\quad r = r + r_i, i = i + 1, r_i = w$;
10. **end while**
11. $k_w[n] = 0, \ldots, k_w[r + 1] = 0, k_w[r] = 1$;
12. **return** $k_w[n], k_w[n-1], \ldots, k_w[0]$.

注意到对于点 R 可以构造其关于 RWNAF 标量表示的恒等式, 即:

$$R = (0\ldots1|0\ldots b|\ldots|0\ldots b)R \tag{1}$$

其中, $b = -(2^w - 1)$, 因此令 $R' = [b]R = [-(2^w - 1)]R$ 用于预计算, 预计算开销为 $t(R') = wD + A$. 此时, 预计算表的内容为 $E' = \{P + R', [3]P + R', \ldots, [(2^w - 1)]P + R'\}$, 预计算开销为 $t(E') = 2^{w-1}A$, 标量乘算法如算法 4 所示.

算法 4. RWNAF 标量乘算法[10].

输入: $NAF_{\mathrm{rw}}(k)$, 椭圆曲线上一点 P;
输出: $[k]P$.

1. $R = RandomPoint()$;

2. $Q=[k_w[c]]P+R$ /* 其中 c 为 $NAF_{rw}(k)$ 序列中不为 0 的比特的最大序号 */
3. **for** $i=c-1$ **down to** 0 **do**
4. $Q=[2]Q$;
5. **if** $k_w[i]\neq 0$ **then**
6. $Q=Q+P'[i]$; /*其中 $P'[i]=[k_w[i]]P+R'$ 为预计算的点*/
7. **end if**
8. **end for**
9. **return** $Q-R$.

1.2 联合正则形式算法

设 $(k_{n-1},\ldots,k_0)$ 和 $(l_{n-1},\ldots,l_0)$ 分别为 k 和 l 的有符号二进制表示, 满足 $k+l\equiv 1(\bmod 2)$, 并且如果对于任意的 i, k_i 和 l_i 均满足 $(k_i,l_i)=(0,\pm 1)$ 或 $(k_i,l_i)=(\pm 1,0)$, 则 $(k_{n-1},\ldots,k_0)$ 和 $(l_{n-1},\ldots,l_0)$ 称为联合正则形式 (JRF)[17]. JRF 的生成算法如算法 5.

算法 5. JRF 生成算法[17].

输入: 一对整数 (k,l) 满足 $k+l\equiv 1(\bmod 2)$;
输出: $JRF(k,l)$: $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$.

1. $i\leftarrow 0$, $s\leftarrow k$, $t\leftarrow l$;
2. **while** $s>0$ **or** $k>0$ **do**
3. $k_i=s \bmod 2$, $l_i=t \bmod 2$;
4. **if** $(k_i,l_i)=(0,0)$ **then**
5. $k_i\leftarrow k_{i-1}$, $k_{i-1}\leftarrow -k_{i-1}$, $l_i\leftarrow l_{i-1}$, $l_{i-1}\leftarrow -l_{i-1}$, $s\leftarrow s/2$, $t\leftarrow t/2$;
6. **else if** $(k_i,l_i)=(1,1)$ **then**
7. $k_i\leftarrow 1-k_{i-1}$, $k_{i-1}\leftarrow -k_{i-1}$, $l_i\leftarrow 1-l_{i-1}$, $l_{i-1}\leftarrow -l_{i-1}$, $s\leftarrow (s-2k_i+1)/2$, $t\leftarrow (t-2l_i+1)/2$;
8. **else**
9. $s\leftarrow (s-k_i)/2$, $t\leftarrow (t-l_i)/2$;
10. **end if**
11. $i\leftarrow i+1$;
12. **end while**
13. $n\leftarrow i$;
14. **return** $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$.

基于 JRF 生成算法的多标量乘算法如算法 6.

算法 6. JRF 多标量乘算法[17].

输入: 椭圆曲线上两点 P,Q, $JRF(k,l)$: $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$;
输出: $[k]P+[l]Q$.

1. $R\leftarrow O$;
2. **for** $i=n-1$ **down to** 0 **do**
3. $R=[2]R$;
4. **if** $k_i\neq 0$ **then**

5. $R = R + [k_i]P$;
6. **else**
7. $R = R + [l_i]Q$;
8. **end if**
9. **end for**
10. **return** R.

从算法 6 可知, 生成 k 和 l 的 JRF 格式后, 计算多标量乘法时, 可始终保证 k_i 和 l_i 其中一个为 ±1 而另一个为 0, 这样在第 2 步后的每轮迭代中都必然要计算一次点加运算和倍点运算, 实现了一致的点加-倍点 (double-and-add always) 操作, 达到了抗 SPA 攻击的效果.

1.3 常见侧信道攻击及其防御

在 1999 年, Kocher 等人[7]指出一般的二进制标量乘算法是将乘法和加法转化为移位操作, 攻击者可以在每一轮迭代中分析标量, 获取私密信息, 从而该计算方法不能抵抗侧信道攻击. 这是一种通过分析密码设备泄露的边信道信息来推测秘钥的密码分析方法, 同时是目前最常见的密码算法攻击方式, 能通过加密设备在运行过程中的边信道信息比如电压, 功耗等的泄露来破解密码算法. 对于椭圆曲线标量乘算法, 常见的攻击如下.

简单能量分析 (SPA)[7]攻击: 通过分析功耗曲线来获得密钥等私密信息. 例如, 当使用 double-and-add 算法进行标量乘时, 如果攻击者能够从功耗曲线中区分点加运算和倍点运算, 就可以获取标量的值. 主要防护措施有: 原子块处理, 一致的点加-倍点操作等.

差分能量分析 (DPA)[7]攻击: 使用统计技术从测量中获取私密信息. 攻击者通过为密码设备设置多个输入点, 记录并存储随时间变化的侧信道测量值, 通过分析大量功耗曲线的相对差异进而推测算法密钥, 主要防护措施有: 基点掩码, 随机化投影坐标, 随机密钥分割等.

修正能量分析 (refined power analysis, RPA)[18]攻击: 利用特殊点 $(x,0)$ 或 $(0,y)$ 攻击随机化投影坐标的方法, 进而使攻击者可以通过 DPA 攻击获取密钥, 相较 DPA 对设备的要求也更高更复杂. 主要防护措施有: 基点掩码, 随机密钥分割等.

零值点能量分析 (zero power analysis, ZPA)[19]攻击: 这种攻击不依赖于功耗分析, 而是通过分析目标设备在不同操作状态下的电磁辐射或其他物理特性来获取敏感信息. ZPA 可以在设备处于低功耗或零功耗状态时进行攻击, 相对于 DPA 更具挑战性, 但相较 DPA 对设备的要求也更高更复杂. 主要防护措施有: 基点掩码, 随机密钥分割等.

Flush+Reload 攻击[20]: 该方法由 Yarom 等人在 2014 年首次提出. 其原理是利用现代处理器的缓存机制. 攻击者首先通过特定的指令将目标设备中的某些数据从缓存中清除 (Flush), 然后监视缓存中的访问情况, 当被攻击者再次访问这些数据时, 攻击者可以通过观察访问时间的差异来推断出数据的敏感信息 (Reload). 通过不断重复 Flush 和 Reload 的过程, 攻击者可以逐渐还原目标设备中的数据, 包括密钥等敏感信息, 需要针对具体的攻击方式决定防御措施.

2 基于 OWNAF 的安全高效标量乘算法方案

考虑在 wNAF 和 RWNAF 算法现有的思路及效率基础上, 通过调整 NAF 序列的生成方式来抵御侧信道攻击, 避免直接计算 k 的 NAF 形式来直接得出标量乘运算结果, 实现安全高效的标量乘算法, 该算法分为以下几个阶段.

2.1 预计算阶段

首先, 在 RWNAF 算法的基础上, 对算法中非 0 值的取值范围进行调整, 同时通过一定的处理规避可能存在子序列中全为 0 的情况. 新的 OWNAF 算法如算法 7 所示.

算法 7. 标量 k 的 OWNAF(k) 生成算法.

输入: 标量 k;

输出: $NAF_{\mathrm{ow}}(k)$.

1. $a \leftarrow 0$, $i \leftarrow 0$;
2. **while** $k \geqslant 1$ **do**
3. $\quad T[i] = (k \bmod 2^{w+1})$;
4. $\quad T[i] = (T[i] - 2^w \cdot MSB_w(T[i])) \bmod 2^w$; /* $MSB_w(k)$ 表示取标量 k 的二进制表示的第 w 位 */
5. $\quad T[0] = T[0]\text{^}((-((T[0] == 0)\&1))\&(2^w\text{^}T[0]))$;/* $T[0] = (T[0] == 0)\ ?\ 2^w : T[0]$ */
6. $\quad t = (T[i] == 0)$;
7. $\quad T[i-1] += 2^w \cdot sgn(T[i-1]) \cdot t$, $T[i] = (1-t) \cdot T[i] - sgn(T[i-1]) \cdot t$; /* sgn 为符号函数 */
8. $\quad k \leftarrow (k - T[i])/2^w$;
9. $\quad k_{\mathrm{ow}}[a+w-1] = 0, \ldots, k_{\mathrm{ow}}[a+1] = 0$, $k_{\mathrm{ow}}[a] = T[i]$;
10. $\quad a \leftarrow a + w$, $i \leftarrow i + 1$;
11. **end while**
12. **return** $(k_{\mathrm{ow}}[a+w-1], \ldots, k_{\mathrm{ow}}[1], k_{\mathrm{ow}}[0])$.

通过 OWNAF 算法得到了标量 k 的 $NAF_{\mathrm{ow}}(k)$ 表示, 在每个宽度为 w 的子序列中都有形如 $0\ldots0a$ 的形式, 其中 $0 < |a| \leqslant 2^w$, 假设生成的序列长度为 n, 最终得到的序列形如:

$$\overbrace{0\ldots0a|0\ldots0a|\ldots|0\ldots0a}^{\lceil n/w \rceil \text{个子序列}} \tag{2}$$

由于在 RWNAF 算法的基础上调整了 $T[i]$ 的取值范围, OWNAF 算法相较 RWNAF 算法的预计算开销差值为 $(2^{w-1}-1-w)A$, 预计算开销有所增加. OWNAF 算法则规避了 RWNAF 在循环结束后补充一个 $0\ldots01$ 的子序列的处理, 因而生成的序列 OWNAF(k) 的平均长度小于同样窗口大小的 RWNAF 算法生成的序列 $RWNAF(k)$ 的平均长度. 下面给出数学证明.

定理 1. OWNAF 生成的序列长度小于等于 RWNAF.

证明: 不失一般性, 设标量 $k > 0$, 如果 OWNAF 生成的序列长度大于 RWNAF 生成的序列长度, 由算法结构, 假设 OWNAF 的序列多一个窗口长度, 即:

OWNAF 生成的序列 $NAF_{\mathrm{ow}}(k) = (0\ldots x[n+1]|0\ldots x[n]|\ldots|0\ldots x[0])$.

RWNAF 生成的序列 $NAF_{\mathrm{rw}}(k) = (0\ldots y[n]|0\ldots y[n-1]|\ldots|0\ldots y[0])$.

对于 k 的两种标量表示, 显然可以互相转换.

(1) 假设 $x[n+1] = 1$, 由 $1 \cdot 2^{(n+1)w} = (2^w-1)2^{nw} + (2^w-1)2^{(n-1)w} + \ldots + (2^w-1)2^0 + 1$, 有:

$$k = \sum_{i=1}^{n}(x[i] + (2^w-1)) \cdot 2^{iw} + (x[0] + 2^w) = \sum_{i=0}^{n}(y[i]) \cdot 2^{iw} \tag{3}$$

由于 RWNAF 算法构造的序列唯一, 转换后有 $x[i] + (2^w-1) = y[i], 1 \leqslant i \leqslant n$ 且 $x[0] + 2^w = y[0]$, 由 RWNAF 序列中 $y[n] = 1$, 因此 $x[n] = -2^w$, 此时 OWNAF 序列中有 $x[n+1] = 1$ 和 $x[n] = -2^w$, 显然可以将第 $n+1$ 个窗口和第 n 个窗口的子序列合并缩短, 即 OWNAF(k) $= (0\ldots x[n-1]|\ldots|0\ldots x[0])$, 此时 OWNAF 生成的序列长度显然小于 RWNAF 序列生成的长度, 矛盾; $x[n+1] = -1$ 同理.

(2) 假设 $1 < |x[n+1]| \leqslant 2^w$, 在相互转换时, 有 $x[i] + x[n+1] \cdot (2^w-1) = y[i]$, 此时 $y[i]$ 的取值范围显然超过了 RWNAF 算法的取值范围 $[-(2^w-1), 2^w-1]$, 矛盾.

综上, 可知假设错误, 因此 OWNAF 生成的序列长度小于等于 RWNAF. 证毕.

我们基于 OpenSSL 实现了 OWNAF 和 RWNAF 序列生成算法, 针对不同的大数长度和窗口宽度进行 10^6 次

测试. 生成 NAF 序列的长度比对结果如图 1 所示. OWNAF 算法相较于 RWNAF 算法在寄存器相对充足的情况下能够在保证安全的前提下减少标量乘算法的计算开销. 数值实验的结果显示 OWNAF 算法的生成序列平均长度约比 RWNAF 算法生成的序列的平均长度小 w.

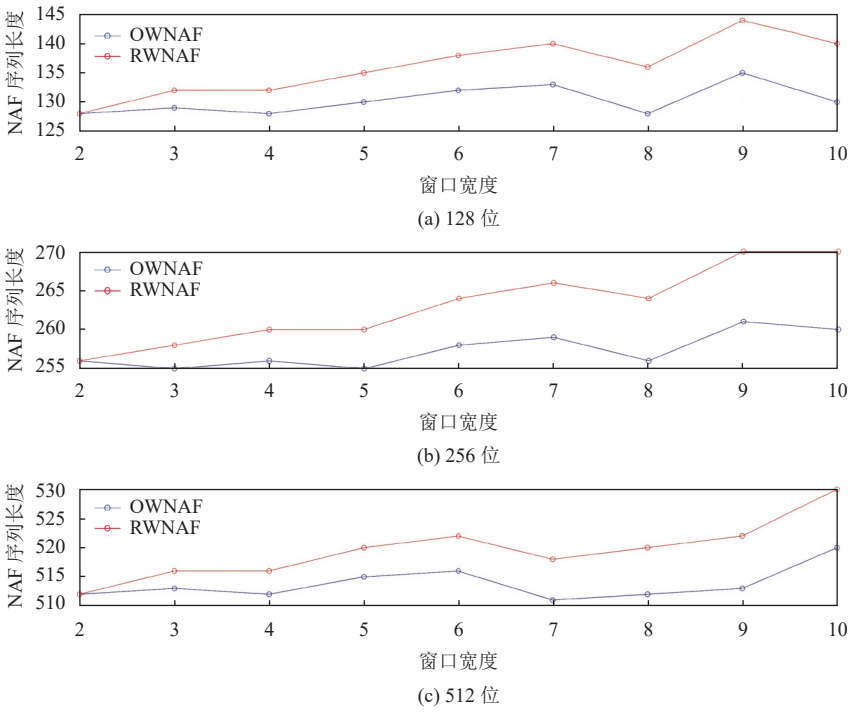

图 1 NAF 序列长度比较

2.2 随机密钥分割阶段

在各种 DPA 对策中, 随机密钥分割是抵抗 DPA 的高效方法[21]. 下面考虑在 OWNAF 算法计算 kP 的过程中引入随机数 r, 考虑 k 的随机密钥分割方式为:

$$k = \lfloor k/r \rfloor r + k \bmod r \tag{4}$$

$$[k]P = [\lfloor k/r \rfloor r]P + [k \bmod r]P \tag{5}$$

在算法 7 的基础上进行调整, 输入标量 k 后生成等长的两组 NAF 随机分割序列 K_1, K_2, 随后通过这个序列组合实现标量乘算法, 如算法 8.

算法 8. 标量 k 的 OWNAF 随机分割算法.

输入: 标量 k, 窗口值 w;

输出: 长度为 n 的两组 NAF 序列 $NAF_{ow}(K_1)$, $NAF_{ow}(K_2)$.

1. $a \leftarrow 0$, $i \leftarrow 0$, $r \leftarrow RandomInt()$;
2. $x \leftarrow \lfloor k/r \rfloor r$, $y = k \bmod r$;
3. **while** $x \geqslant 1$ **or** $y \geqslant 1$ **do**
4. $T_1[i] = (x \bmod 2^{w+1})$;$T_2[i] = (y \bmod 2^{w+1})$;
5. **for** $j = 1,2$ **do**
6. $T_j[i] = (T_j[i] - 2^w \cdot MSB_w(T_j[i])) \bmod 2^w$; /* $MSB_w(k)$ 表示取标量 k 的二进制表示的第 w 位 */

7. $T_j[0]=T_j[0]\hat{}((-((T_j[0]==0)\&1))\&(2^w\hat{}T_j[0]))$; /* $T_j[0]=(T_j[0]==0)\ ?\ 2^w:T_j[0]$ */
8. **for** $j=1,2$ **do**
9. $t_j=(T_j[i]==0)$;
10. $T_j[i-1]+=2^w\cdot sgn(T_j[i-1])\cdot t_j$, $T_j[i]=(1-t_j)\cdot T_j[i]-sgn(T_j[i-1])\cdot t_j$; /* sgn 为符号函数 */
11. **end for**
12. **end for**
13. $x\leftarrow(x-T_1[i])/2^w$, $y\leftarrow(y-T_1[i])/2^w$;
14. $x_{ow}[a+w-1]=0,\ \ldots,x_{ow}[a+1]=0,x_{ow}[a]=T_1[i];y_{ow}[a+w-1]=0,\ \ldots,y_{ow}[a+1]=0,y_{ow}[a]=T_2[i]$;
15. $a\leftarrow a+w$, $i\leftarrow i+1$;
16. **end while**
17. **return** $NAF_{ow}(K_1)=(x_{ow}[n],x_{ow}[n-1],\ \ldots,x_{ow}[1],x_{ow}[0])$, $NAF_{ow}(K_2)=(y_{ow}[n],y_{ow}[n-1],\ \ldots,y_{ow}[1],y_{ow}[0])$.

2.3 标量乘阶段

在算法 8 对密钥随机分割的基础上, 给出如算法 9 的标量乘算法.

算法 9. OWNAF 标量乘算法.

输入: 标量 k, 椭圆曲线上一点 P, 窗口 w;
输出: $[k]P$.

1. 计算 $NAF_{ow}(K_1)$, $NAF_{ow}(K_2)$; /* 算法 8 */
2. 预计算 $P_c=[c]P$, $i\in\{1,2,\ldots,2^w\}$;
3. $Q\leftarrow O$;
4. **for** $i=\lceil n/w\rceil-1$ **down to** 0 **do**
5. $Q\leftarrow[2^w]Q$;
6. $c\leftarrow x_{ow}[i\cdot w]+y_{ow}[i\cdot w]$;
7. $Q\leftarrow Q+sgn(c)P_{|c|}$; /* sgn 为符号函数 */
8. **end for**
9. **return** Q.

2.4 算法分析

(1) 安全性分析

在 OWNAF 算法的第 4 步中的每次计算中点加和倍点运算都是 $D\ldots DA$ 的均匀运算, 攻击者无法通过 SPA 攻击获取任何信息; 并且由于随机地引入 r 来分割序列, 两序列的 $NAF_{ow}(k)$ 本身并不相关, 因此, 攻击者无法通过 DPA 获取任何信息; 攻击者不能在每次执行一个标量乘法时推断出 0 值的特殊点, 因此可以抵御 RPA 和 ZPA; 进一步的, 由于算法 8 生成的序列 $NAF_{ow}(K_j)(j=1,2)$ 中的非 0 元素由于有随机值混淆, 与 wNAF 算法对应的非 0 元素不同, 会给文献 [8,9] 中的破解方法造成干扰, 能在一定程度上抵御这种针对 ECC 签名算法的攻击.

(2) 计算开销与存储开销分析

RWNAF 算法的计算开销分为 3 部分: 预计算随机掩码 $R'=-(2^w-1)R$, 预计算表 E' 和标量乘法运算 s', 它们的计算开销分别为 $t(R')=wD+A$, $t(E')=2^{w-1}A$, $t(s')=n'D+\lceil n'/w\rceil A$, 其中 n' 为 RWNAF 算法生成的序列长度. 而 OWNAF 算法的计算开销分为两部分: 计算 $NAF_{ow}(k)$ 时的预计算表 E 和标量乘法运算 s, 它们的计算开销分别为 $t(E)=wD+(2^w-1-w)A$, $t(s)=nD+\lceil n/w\rceil A$, 其中 n 为 OWNAF 算法生成的序列长度. 它们的计算开销比较如表 2 所示.

表 2 NAF 标量乘算法计算开销比较

算法	预计算开销 $t(E)$	标量乘开销 $t(s)$	总开销
RWNAF[10]	$wD+\left(2^{w-1}+1\right)A$	$n'D+\lceil n'/w\rceil A$	$t(E)+t(s)$
OWNAF (本文)	$wD+(2^w-1-w)A$	$nD+\lceil n/w\rceil A$	

可以看出, 在预计算阶段 OWNAF 算法无需额外对随机掩码进行处理, 但由于需要对 k 进行分割, 需要额外的预计算点用于后续计算. 在标量乘开销方面, 由于 OWNAF 算法生成的 NAF 序列的平均长度 n 小于 RWNAF 算法生成的 NAF 序列的平均长度 n', 因此在标量乘方面的开销有了一定的优势.

为了进一步验证本文算法的性能, 本文在 64 位 Windows 11 操作系统上使用 C++语言和 OpenSSL 库实现上述算法, 实验环境中处理器型号为 Intel(R) Core(TM) i9-13900H 处理器, 主频为 2.60 GHz. 实验环境如表 3 所示.

表 3 实验环境

实验环境	配置
操作系统	Windows 11
CPU型号	i9-13900H (2.60 GHz)
使用语言	C++
运行内存	16 GB
代码库版本	OpenSSL 3.2.0

我们使用随机生成器生成 128, 256 和 512 位的随机整数, OpenSSL 中椭圆曲线的参数分别设置为 NID_secp128r1, NID_sm2 和 NID_secp521r1 (3 类不同安全级别的椭圆曲线). 针对不同的窗口长度, 分别使用 RWNAF 和 OWNAF 算法进行 10^6 次标量乘运算, 最终标量乘效率比对结果如表 4 所示. 数值实验结果表明, 基于 OWNAF 的标量乘法相较基于 RWNAF 的标量乘法实现效率提升了 2%–5% 左右, 在保证安全的基础上提升了标量乘算法的效率.

表 4 NAF 标量乘算法效率比较

大数长度 (bit)	窗口宽度	RWNAF算法耗时 (ms)	OWNAF算法耗时 (ms)	加速比 (%)
128	2	0.332 667	0.315 723	5.09
128	3	0.287 894	0.276 295	4.03
128	4	0.264 948	0.254 496	3.95
128	5	0.259 858	0.247 743	4.66
128	6	0.240 851	0.228 241	5.24
256	2	0.668 457	0.645 516	3.43
256	3	0.572 802	0.553 764	3.32
256	4	0.524 114	0.509 962	2.70
256	5	0.496 005	0.479 385	3.35
256	6	0.486 888	0.468 379	3.80
512	2	1.376 3	1.334 23	3.06
512	3	1.151 38	1.127 11	2.11
512	4	1.059 36	1.029 46	2.82
512	5	0.995 034	0.968 719	2.64
512	6	0.945 202	0.921 907	2.46

3 基于 wJRF 的多标量乘算法方案

3.1 JRF 标量乘算法方案

注意到当 $l=k+1$ 时, 假设 $k=\sum_{i=0}^{n-1}k_i$, 有 $k+1=2^n+\sum_{i=0}^{n-1}(k_i-1)2^i$, 计算 k 和 l 的 JRF 格式将非常简单, 此时有算法 10.

算法 10. simple-JRF.

输入: 一对整数 (k,l) 满足 $l=k+1$;

输出: $JRF(k,l)$: $(k_n,\ldots,k_0)$, $(l_n,\ldots,l_0)$.

1. $k=(k_{n-1},\ldots,k_0)_2$;
2. $l_i=k_i-1$ $(0\leqslant i\leqslant n-1)$;
3. $k_n=0$, $l_n=1$;
4. **return** $(k_n,\ldots,k_0)$, $(l_n,\ldots,l_0)$.

假设 $k=53$, 有 $JRF(53,54)$: $53=(0110101)$ 和 $54=\left(100\bar{1}0\bar{1}0\right)$, 其中 $\bar{1}=-1$. 因此, 基于算法 10, 可以很容易地得到 k 和 $k+1$ 的 JRF 表示, 这将为构造出一种一致的点加-倍点标量乘算法提供很大的便利.

随机掩码是保证标量乘法不受 DPA 影响的有效方法[21], 考虑用 $P+R$ 代替了计算中的 P, 保证在计算过程中随机遮蔽了基点 P, 考虑如下转换:

$$[k]P=[k](P+R)-[k+1]R+R \tag{6}$$

构造标量乘算法如算法 11.

算法 11. 基于 JRF 的抗 DPA 标量乘算法.

输入: 标量 k, 椭圆曲线上一基点 P;

输出: $[k]P$.

1. 计算 $JRF(k,k+1)$: $(k_n,\ldots,k_0)$, $(l_n,\ldots,l_0)$; /* 算法 10 */
2. $R\leftarrow RandomPoint()$;
3. $T=O$, $T(1)=-R$, $T(2)=P+R$, $T(-1)=R$, $T(-2)=-P-R$;
4. **for** $i=n$ **down to** 0 **do**
5. $\quad T=[2]T$;
6. $\quad c\leftarrow 2k_i+l_i$, $T=T+T(c)$;
7. **end for**
8. **return** $T+T(-1)$.

仍旧以 $k=53$ 为例, 算法 11 的计算过程如下:

i	6	5	4	3	2	1	0
c	1	2	2	−1	2	−1	2
$T(c)$	$-R$	$P+R$	$P+R$	R	$P+R$	R	$P+R$
R	$-R$	$P-R$	$[3]P-R$	$[6]P-R$	$[13]P-R$	$[26]P-R$	$[53]P-R$

最终返回 $R+T(-1)=[53]P-R+R=[53]P$, 与预期结果一致.

3.2 JRF 标量乘算法分析

(1) 安全性分析

算法 11 中, 由于 k 和 $k+1$ 满足一奇一偶的条件, 通过 simple-JRF 算法构造了它们的联合正则表示, 由于 k_i 和 l_i 满足其中一方等于 ±1 而另一方等于 0, 因此 $c=2k_i+l_i\neq 0$, $T(c)\neq O$, 这样在得到 $[k]P$ 的过程中的每次循环都必然会执行一次倍点运算和点加运算, 能够抵御 SPA 攻击; 由于引入了随机掩码 R, 预计算点在每次执行期间都会发生变化, 因此能在一定程度上抵御 DPA 及其变种 RPA 和 ZPA.

(2) 计算开销与存储开销分析

算法 11 基于 JRF, 只需预计算 4 个点便实现了抗 DPA 攻击的安全标量乘算法, 但是由于 JRF 的特性影响, 每次循环都需要一次点加运算和倍点运算, 计算开销为 $nD+nA$, 因此, 更适合密码卡这种存储资源受限的应用场景.

3.3 wJRF 标量形式

注意到 JRF 算法只是调整了 k 和 l 的表示方式, 从而使得在 JRF 的相同位置上都满足一个为 0, 另一个为 ±1. 此基础上, 将窗口方法的思想应用到 JRF 形式上, 使得算法中出现的非 0 值不局限于 ±1, 定义 wJRF 如下: 设 $(k_{n-1},\ldots,k_0)$ 和 $(l_{n-1},\ldots,l_0)$ 分别为 k 和 l 的有符号二进制表示, 窗口宽度为 w, 如果 k 和 l 满足 $k \bmod 2^{w-1} \equiv 0$ 且 $l \bmod 2^{w-1} \not\equiv 0$. 且对于任意 $i \in [1, n-2]$, k_i 和 l_i 满足 $(k_i, l_i) = (0, \pm 2^{w-2})$ 或 $(k_i, l_i) = (\pm 2^{w-2}, 0)$, 此时将 $(k_{n-1},\ldots,k_0)$ 和 $(l_{n-1},\ldots,l_0)$ 称为窗口联合正则形式 (wJRF). 关于多标量的 wJRF 有如下重要性质.

定理 2. wJRF 存在且唯一.

证明: 首先证明 wJRF 的存在性, 最直接的方法是给出一个计算 wJRF 结构的算法.

对于一对整数 (k,l) 满足 $k \bmod 2^{w-1} \equiv 0$ 且 $l \bmod 2^{w-1} \not\equiv 0$, 在 JRF 算法的基础上给出类似的 wJRF 算法: 首先, 由于整数 (k,l) 满足 $k \bmod 2^{w-1} \equiv 0$ 且 $l \bmod 2^{w-1} \not\equiv 0$, 显然 k_0 和 l_0 一个为 0 而另一个非 0, 假设非 0 值为 λ, 满足 $0 \leqslant |\lambda| \leqslant 2^{w-2}$. 这是由于给出的 wJRF 只要求 $k \bmod 2^{w-1} \equiv 0$ 并且 $l \bmod 2^{w-1} \not\equiv 0$, 对于 wJRF 形式的其他位置的非 0 值, 显然只有 $\pm 2^{w-1}$, 因此接下来, 首先来观察 (k_1, l_1) 的情况.

(1) 假设 $(k_1, l_1) = (0,0)$, 为了保证在调整表示后的结果与原先一致, 令 $\widetilde{\lambda} = \lambda - 2^{w-1} \bmod 2^{w-1}$, 此时只会出现下面两种情况之一:

$$\begin{matrix} k_1 & k_0 \\ 0 & \lambda \\ 0 & 0 \\ l_1 & l_0 \end{matrix} \Rightarrow \begin{matrix} k_1 & k_0 \\ 2^{w-1} & \widetilde{\lambda} \\ 0 & 0 \\ l_1 & l_0 \end{matrix}, \begin{matrix} k_1 & k_0 \\ 0 & 0 \\ 0 & \lambda \\ l_1 & l_0 \end{matrix} \Rightarrow \begin{matrix} k_1 & k_0 \\ 0 & 0 \\ 2^{w-1} & \widetilde{\lambda} \\ l_1 & l_0 \end{matrix}$$

此时显然 $2 \cdot 2^{w-1} + \lambda - 2^w = \lambda$, 前后表示结果一致, 此时计算过程中的 s 和 t 无需调整;

(2) $(k_1, l_1) = (2^{w-1}, 2^{w-1})$, 令 $\widetilde{\lambda} = \lambda - 2^{w-1} \bmod 2^{w-1}$, 同样地, 为了满足 w-JRF 格式, 必然会导致下面两种情况之一:

$$\begin{matrix} k_1 & k_0 \\ 2^{w-1} & \lambda \\ 2^{w-1} & 0 \\ l_1 & l_0 \end{matrix} \Rightarrow \begin{matrix} k_1 & k_0 \\ 0 & \widetilde{\lambda} \\ 2^{w-1} & 0 \\ l_1 & l_0 \end{matrix}, \begin{matrix} k_1 & k_0 \\ 2^{w-1} & 0 \\ 2^{w-1} & \lambda \\ l_1 & l_0 \end{matrix} \Rightarrow \begin{matrix} k_1 & k_0 \\ 2^{w-1} & 0 \\ 0 & \widetilde{\lambda} \\ l_1 & l_0 \end{matrix}$$

此时 k 和 l 满足了 wJRF 格式的要求, 但是显然 s 和 t 需要补充变动后缺少的值 2^{w-1}, 因此 $s \leftarrow (s - 2k_1 + 2^{w-1})/2$, $t \leftarrow (t - 2l_1 + 2^{w-1})/2$.

(3) k_1 和 l_1 一个为 0 而另一个非 0, 满足格式, 正常进行.

(4) 对于任意的 $(k_i, l_i)(1 < i \leqslant n-1)$, 由于此时任意位数可能存在的非 0 值均为 $\pm 2^{w-1}$, 因此上述对 $\widetilde{\lambda}$ 的处理实际上等价于 $\widetilde{\lambda} = -\lambda \bmod 2^{w-1}$, 与 JRF 算法计算过程近似, 证毕.

下面证明 wJRF 的唯一性.

假设有两个长度为 n 的不同 wJRF 表示:

$$\begin{cases} k = (k_{n-1},\ldots,k_0) = (k'_{n-1}, \ldots, k'_0) \\ l = (l_{n-1},\ldots,l_0) = (l'_{n-1}, \ldots, l'_0) \end{cases} \tag{7}$$

其中, j 是满足 $k_i \neq k'_i$ 或 $l_i \neq l'_i$ 的最小值并且:

$$\begin{cases} s = (k_{n-1},\ldots,k_j) = \left(k'_{n-1}, \ldots, k'_j\right) \\ t = (l_{n-1},\ldots,l_j) = \left(l'_{n-1}, \ldots, l'_j\right) \end{cases} \tag{8}$$

由于 k 和 l 可交换, 不失一般性, 假设 $k_i \neq k'_i$. 此时必然有 k_j 和 k'_j 非 0, 其中一个为 2^{w-1}, 另一个为 -2^{w-1}. 不失一般性, 假设 $k_j = 2^{w-1}$, $k'_j = -2^{w-1}$. 这样通过 wJRF 的定义可知有 $l_j = l'_j = 0$.

此时, 假设 $s \equiv 2^{w-1} \bmod 2^{w+1}$, 根据 wJRF 的定义有 $k_{j+1} = 0$ 且 $k'_{j+1} = \pm 2^{w-1}$, 这样显然可以得出 $l_{j+1} = \pm 2^{w-1}$ 且 $l'_{j+1} = 0$, 这样前者说明 $t \equiv 2^w \bmod 2^{w+1}$ 而后者说明 $t \equiv 0 \bmod 2^{w+1}$, 该矛盾表明最初的假设错误. 假设 $s \equiv 2^w + 2^{w-1} \bmod 2^{w+1}$, 同样可以得到类似的结论. 唯一性得证.

3.4 wJRF 多标量乘算法方案

定理 2 证明了 wJRF 存在唯一, 这样从最小有效位出发的变换, 可以得到相应的 wJRF 格式, 算法过程只需要计算 $[\pm 2^{w-1}]P$, $[\lambda]P$ 和 $[\widetilde{\lambda}]P$ 这 4 个点 ($\lambda = k + l \bmod 2^{w-1}$, $\widetilde{\lambda} = \lambda - 2^{w-1} \bmod 2^{w-1}$), wJRF 的生成算法如算法 12.

算法 12. wJRF 生成算法.

输入: 一对整数 (k,l) 满足 $k \bmod 2^{w-1} \equiv 0$ 且 $l \bmod 2^{w-1} \not\equiv 0$;

输出: $wJRF(k,l)$: $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$.

1. $i \leftarrow 0$, $s \leftarrow k$, $t \leftarrow l$;
2. **while** $s > 0$ **and** $t > 0$ **do**
3. $\quad k_i = s \bmod 2^{w-1}$, $l_i = t \bmod 2^{w-1}$;
4. $\quad a = sgn(k_i)$, $b = sgn(l_i)$, $c = ((-((a == b)\&1))\&1)$; /* $c = (a == b)?1 : 0$ */
5. $\quad k_i \leftarrow c \cdot (a \cdot 2^w + (1-2a) \cdot k_{i-1}) + (1-c) \cdot k_i$, $k_{i-1} \leftarrow \widetilde{k_{i-1}}$;
6. $\quad l_i \leftarrow c \cdot (b \cdot 2^w + (1-2b) \cdot l_{i-1}) + (1-c) \cdot l_i$, $l_{i-1} \leftarrow \widetilde{l_{i-1}}$;
7. $\quad s \leftarrow (s - (1-c-2a \cdot c) \cdot k_i + a \cdot c \cdot 2^{w-1})/2$, $t \leftarrow (t - (1-c-2b \cdot c) \cdot k_i + b \cdot c \cdot 2^{w-1})/2$;
8. $\quad i \leftarrow i + 1$;
9. **end while**
12. $k_{i+1} = sgn(s)$, $l_{i+1} = 1 - sgn(s)$; /* sgn 为符号函数 */
10. $n \leftarrow i + 1$;
11. **return** $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$.

由于多标量乘法对标量随机分割整体开销较大, 因此通过随机掩码的方式来抵抗 DPA 等攻击: 计算 $[k]P + [l]Q + R$, 其中 R 是椭圆曲线上的随机点, 先计算 $wJRF(k,l)$: $(k_{n-1},\ldots,k_0)$, $(l_{n-1},\ldots,l_0)$, 此时对 R 做如下变换:

$$R = (1\bar{1}\bar{1}\ldots\bar{1})R \tag{9}$$

其中, $\bar{1} = -1$, 令 $P_i = [i]P - R$, $Q_i = [i]Q - R$, $i \in \{\pm 1, \pm 2, \ldots, \pm 2^{w-1}\}$. 将每个窗口的标量信息提前预计算, 进而减少标量乘过程中的点加运算次数. 设 $wJRF(k,l)$ 中第 i 个非 0 下标对应的标量乘法值为 T_i^j, 即当 $k_i \neq 0$ 时, $j = 0$, $T_i^0 = P_{k_i}$, 反之 $j = 1$, $T_i^1 = Q_{l_i}$, 令:

$$T_{i_1,i_2,\ldots,i_w}^{j_1,j_2,\ldots,j_w} = [2^{w-1}]T_{i_1}^{j_1} + [2^{w-2}]T_{i_2}^{j_2} + \ldots + T_{i_w}^{j_w} \tag{10}$$

其中, $i_1, i_2, \ldots, i_w$ 对应窗口长度为 w 的窗口中每列的非 0 值, 此时可通过对应的下标找到对应窗口的标量值, 其中非 0 值等于 $\pm 2^{w-1}$, 因此预计算表需要的存储大小为 2^{w+1}. 基于 wJRF 的抗 DPA 多标量乘算法如算法 13.

算法 13. 基于 wJRF 的抗 DPA 多标量乘算法.

输入: 椭圆曲线上两点 P,Q, 窗口宽度 w, 一对整数 (k,l);

输出: $[k]P + [l]Q$.

1. $t \leftarrow 0$;
2. **while** $k \bmod 2^{w-1} \equiv 0$ **and** $l \bmod 2^{w-1} \equiv 0$ **do**
3. $\quad l \leftarrow l + 1$, $t \leftarrow t + 1$;
4. **end while**
5. $R = RandomPoint()$;
6. $T_i^0 = [i]P - R$, $T_i^1 = [i]Q - R$, $i \in \{\pm 1, \pm 2, \ldots, \pm 2^{w-1}\}$;
7. 预计算 $T_{i_1,i_2,\ldots,i_w}^{j_1,j_2,\ldots,j_w} = [2^{w-1}]T_{i_1}^{j_1} + [2^{w-2}]T_{i_2}^{j_2} + \ldots + T_{i_w}^{j_w}$, $i_s \in [\pm 2^{w-1}]$, $j_s \in [0,1]$, $s \in [1,2,\ldots,w]$;

8. 计算 $wJRF(k,l)$; /* 算法 12 */
9. $T \leftarrow R$;
10. **for** $i = n-1$ **down to** 0 **by** w **do**
11. $T = [2^w]T$;
12. 根据窗口内的信息确定 $i_1, i_2, \ldots, i_w$ 和 $j_1, j_2, \ldots, j_w$ 的取值;
13. $T = T + T^{j_1, j_2, \ldots, j_w}_{i_1, i_2, \ldots, i_w}$;
14. **end for**
15. **return** $T - [t]Q - R$.

以 52 和 39, $w = 2$ 为例, 此时它们的 wJRF 表示为 (102020) 和 $\left(020201\overline{1}\right)$, 此时的计算过程为:

i	5	4	3	2	1	0
k_i	1	0	2	0	2	0
l_i	0	2	0	2	0	−1
T	O	$[2]P+[2]Q+R$	O	$[12]P+[10]Q+R$	O	$[52]P+[39]Q+R$

最终返回 $T-[0]Q-R=[52]P+[39]Q$, 与预期结果一致. 且通过窗口算法只需 $6D+3A$ 便完成了计算, 有效提高了多标量乘运算的效率.

3.5 wJRF 多标量乘算法分析

(1) 安全性分析

通过算法 12 生成了 k 和 l 的 wJRF 格式后, 计算多标量乘法 $[k]P+[l]Q$ 时, 每次都只需要计算一次点加运算和倍点运算, 实现了一致的点加-倍点标量乘算法, 达到了抗 SPA 攻击的效果. 除此之外, 由于每次算法中都引入了一个随机掩码 R, 使得 DPA 攻击无法获取有效的能量曲线, 进而防御 DPA 攻击. 引入随机掩码 R 也使得多标量乘的计算过程中不存在坐标为 0 值的特殊点, 不使用零值寄存器, 进而防御 RPA 和 ZPA 攻击.

(2) 计算开销与存储开销分析

我们将算法 13 与已有的算法进行横向对比, 比对结果如表 5 所示, 算法 13 在与其他算法相比除了抗 SPA 攻击的能力外, 还增加了对 DPA, RPA 和 ZPA 攻击的抵抗能力. 计算开销为 $nD+\lceil n/w\rceil A$, 相较已有算法的标量乘效率有效提升, 有效提高了多标量乘运算的效率.

表 5 多标量乘算法横向对比

算法	预计算开销	多标量乘开销	抗SPA	抗DPA	抗RPA	抗ZPA
文献[13]	5	$nD+nA$	√	×	×	×
文献[14]	3	$nD+nA$	√	×	×	×
文献[15]	5	$1.384nD+0.692nA$	√	×	×	×
文献[17]	2	$nD+nA$	√	×	×	×
算法13	2^{w+1}	$nD+\lceil n/w\rceil A$	√	√	√	√

4 总 结

本文研究了基于 wNAF 的标量乘算法 OWNAF, 基于 JRF 的标量乘算法和基于 wJRF 的多标量乘算法: OWNAF 算法相较于 RWNAF 算法在存储开销上有所增加, 但计算效率更高, 在抵御 SPA, DPA 等攻击的同时, 能够一定程度抵御 Flush+Reload 攻击; 基于 wJRF 的多标量乘算法在付出一定预计算量的代价下能够有效提升多标量乘算法的效率, 且能抵御 SPA, DPA 等攻击. 未来期待在这些算法的基础上更进一步地减少存储开销和提升计算效率, 推动椭圆曲线标量乘和多标量乘算法的改进.

References:

[1] Miller VS. Use of elliptic curves in cryptography. In: Williams HC, ed. Advances in Cryptology—CRYPTO 1985 Proc. Berlin: Springer, 1986. 417–426. [doi: 10.1007/3-540-39799-X_31]

[2] Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation, 1987, 48(177): 203–209. [doi: 10.1090/S0025-5718-1987-0866109-5]

[3] Hankerson D, Menezes A, Vanstone S. Guide to Elliptic Curve Cryptography. New York: Springer, 2004. [doi: 10.1007/b97644]

[4] Okeya K, Takagi T. The width-*w* NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks. In: Joye M, ed. Topics in Cryptology—CT-RSA 2003. San Francisco: Springer, 2003. 328–343. [doi: 10.1007/3-540-36563-x_23]

[5] Anagreh M, Vainikko E, Laud P. Accelerate performance for elliptic curve scalar multiplication based on NAF by parallel computing. In: Proc. of the 5th Int'l Conf. on Information Systems Security and Privacy. Prague: SciTePress, 2019. 238–245. [doi: 10.5220/0007312702380245]

[6] Hu XH, Zheng X, Zhang SS, Li WJ, Cai ST, Xiong XM. A high-performance elliptic curve cryptographic processor of SM2 over GF(*P*). Electronics, 2019, 8(4): 431. [doi: 10.3390/electronics8040431]

[7] Kocher P, Jaffe J, Jun B. Differential power analysis. In: Proc. of the 19th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 1999. 388–397. [doi: 10.1007/3-540-48405-1_25]

[8] Cao JZ, Cheng QF, Weng J. EHNP strikes back: Analyzing SM2 implementations. In: Proc. of the 13th Int'l Conf. on Cryptology in Africa. Fes: Springer, 2022. 576–600. [doi: 10.1007/978-3-031-17433-9_25]

[9] Ma ZQ, Li SG, Lin JQ, Cai QW, Fan SQ, Zhang F, Luo B. Another lattice attack against ECDSA with the wNAF to recover more bits per signature. In: Proc. of the 18th EAI Int'l Conf. on Security and Privacy in Communication Networks. Springer, 2022. 111–129. [doi: 10.1007/978-3-031-25538-0_7]

[10] Zhang T, Fan MY, Wang GW, Lu XJ. Protection against power analysis attack for ECC on smartcard. Computer Engineering, 2007, 33(14): 125–127. (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-3428.2007.14.044]

[11] Yao JB, Yan CQ, Zhang T. Elliptic curve cryptography algorithm against energy attack. In: Proc. of the 2021 IEEE Conf. on Telecommunications, Optics and Computer Science (TOCS). Shenyang: IEEE, 2021. 224–227. [doi: 10.1109/tocs53301.2021.9688886]

[12] Shi L, Xu M. DWNAF: A dynamic window NAF scalar multiplication with threshold. Computer Science, 2017, 44(10): 159–164. (in Chinese with English abstract). [doi: 10.11896/j.issn.1002-137X.2017.10.030]

[13] Lee MK. SPA-resistant simultaneous scalar multiplication. In: Proc. of the 2005 Int'l Conf. on Computational Science and Its Applications. Singapore: Springer, 2005. 314–321. [doi: 10.1007/11424826_33]

[14] Ciet M, Joye M. (Virtually) free randomization techniques for elliptic curve cryptography. In: Proc. of the 5th Int'l Conf. on Information and Communications Security. Huhehaote: Springer, 2003. 348–359. [doi: 10.1007/978-3-540-39927-8_32]

[15] Liu D, Tan ZY, Dai YQ. New elliptic curve multi-scalar multiplication algorithm for a pair of integers to resist SPA. In: Proc. of the 4th Int'l Conf. on Information Security and Cryptology. Beijing: Springer, 2008. 253–264. [doi: 10.1007/978-3-642-01440-6_20]

[16] Chen HY, Ma CG. A multiple scalar multiplications algorithm in the elliptic curve cryptosystem. Ruan Jian Xue Bao/Journal of Software, 2011, 22(4): 782–788. (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3730.htm [doi: 10.3724/SP.J.1001.2011.03730]

[17] Akishita T, Katagi M, Kitamura I. SPA-resistant scalar multiplication on hyperelliptic curve cryptosystems combining divisor decomposition technique and joint regular form. In: Proc. of the 8th Int'l Workshop on Cryptographic Hardware and Embedded Systems. Yokohama: Springer, 2006. 148–159. [doi: 10.1007/11894063_12]

[18] Goubin L. A refined power-analysis attack on elliptic curve cryptosystems. In: Proc. of the 6th Int'l Workshop on Public Key Cryptography. Miami: Springer, 2003. 199–211. [doi: 10.1007/3-540-36288-6_15]

[19] Akishita T, Takagi T. Zero-value point attacks on elliptic curve cryptosystem. In: Proc. of the 6th Int'l Conf. on Information Security. Bristol: Springer, 2003. 218–233. [doi: 10.1007/10958513_17]

[20] Yarom Y, Falkner K. Flush+Reload: A high resolution, low noise, L3 cache side-channel attack. In: Proc. of the 23rd USENIX Security Symp. San Diego: USENIX Association, 2014. 719–732.

[21] Fan JF, Verbauwhede I. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In: Naccache D, ed. Cryptography and Security: From Theory to Applications: Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday. Berlin: Springer, 2012. 265–282. [doi: 10.1007/978-3-642-28368-0_18]

附中文参考文献:

[10] 张涛, 范明钰, 王光卫, 鲁晓军. Smartcard 上椭圆曲线密码算法的能量攻击和防御. 计算机工程, 2007, 33(14): 125–127. [doi: 10.3969/j.issn.1000-3428.2007.14.044]

[12] 史量, 徐明. DWNAF: 带门限的动态窗口的 NAF 标量乘法. 计算机科学, 2017, 44(10): 159–164. [doi: 10.11896/j.issn.1002-137X.2017.10.030]

[16] 陈厚友, 马传贵. 椭圆曲线密码中一种多标量乘算法. 软件学报, 2011, 22(4): 782–788. http://www.jos.org.cn/1000-9825/3730.htm [doi: 10.3724/SP.J.1001.2011.03730]

程石(1999－), 男, 硕士生, 主要研究领域为国密 SM2 算法.

陶铮(1996－), 男, 博士生, 主要研究领域为椭圆曲线密码, 同源密码.

胡志(1985－), 男, 博士, 副教授, 博士生导师, 主要研究领域为密码学, 信息安全, 算法设计与分析.