Software Engineering and Applications 软件工程与应用, 2025, 14(5), 1056-1066
Published Online October 2025 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2025.145094

Hans汉斯

一种并行动态滑动窗口算法研究与实现

田　田[1]，徐　军[2]，李子臣[1*]

[1]北京印刷学院信息工程学院，北京
[2]香港城市大学建造与土木工程学系，香港

收稿日期：2025年8月8日；录用日期：2025年9月30日；发布日期：2025年10月13日

摘　要

椭圆曲线倍点运算在密码学中有广泛应用，其效率直接影响密码算法与协议的性能。本文提出一种并行动态滑动窗口算法，将椭圆曲线倍点运算的标量k分为高位k_1和低位k_2，分别并行计算k_1P和k_2P后，再合并计算kP，且各自内部采用滑动窗口法。根据高位k_1和低位k_2的具体情况，设计不同的窗口大小，提升了倍点运算的速度。实验表明，该方法在保证正确性的前提下，较传统滑动窗口效率提升了约36.5%。最后，基于本文提出并行动态滑动窗口算法进一步优化了SM2公钥密码算法。

关键词

椭圆曲线倍点运算，动态滑动窗口，SM2公钥密码算法

Research and Application of a Parallel Dynamic Sliding Window Algorithms

Tian Tian[1], Jun Xu[2], Zichen Li[1*]

[1]School of Information Engineering, Beijing Institute of Graphic Communication, Beijing
[2]Department of Architecture and Civil Engineering, City University of Hong Kong, Hong Kong

Received: August 8, 2025; accepted: September 30, 2025; published: October 13, 2025

Abstract

Elliptic curve point multiplication is widely used in cryptography, and its efficiency directly impacts the performance of cryptographic algorithms and protocols. This paper proposes a parallel dynamic sliding window algorithm to accelerate point multiplication. The scalar k is split into a high-order part k_1 and a low-order part k_2, which are processed in parallel to compute k_1P and k_2P separately before merging them to obtain the final result kP. Internally, each computation employs a sliding

*通讯作者。

文章引用: 田田, 徐军, 李子臣. 一种并行动态滑动窗口算法研究与实现[J]. 软件工程与应用, 2025, 14(5): 1056-1066.
DOI: 10.12677/sea.2025.145094

window method to reduce the number of point additions. Additionally, different window sizes are dynamically selected based on the values of k_1 and k_2 to further optimize performance. Experimental results demonstrate that the proposed method improves computational efficiency by approximately 36.5% compared to the traditional sliding window approach while ensuring correctness. Finally, the parallel dynamic sliding window algorithm is applied to optimize the SM2 public-key cryptographic algorithm, further enhancing its performance.

Keywords

Elliptic Curve Point Multiplication, Dynamic Sliding Window, SM2 Public Key Cryptography Algorithm

1. 引言

椭圆曲线密码学(Elliptic Curve Cryptography, ECC)因其在相同安全强度下密钥长度更短、计算效率更高的优势，已成为现代公钥密码体系的重要组成部分[1]。作为我国自主设计的商用密码标准，SM2 算法在电子签名、密钥交换等领域得到广泛应用。然而，SM2 算法的核心运算——椭圆曲线倍点运算(kP)因其计算复杂度高，往往成为整个密码协议的效率瓶颈[2] [3]。在典型的 256 位 SM2 实现中，一次倍点运算需要执行数千次有限域运算，这使得倍点优化成为提升 SM2 算法性能的关键[4]。

为优化倍点运算，研究者们提出了多种算法。现有工作主要沿着串行优化和并行优化两个方向展开。在串行优化方面，文献[5]提到的三种 NAF 编码方法，通过减少非零比特数，降低点加次数，将计算效率提升约了 31%~35%，以及一些其他的 NAF 改进方案[6] [7]。同时，国家密码管理局标准文件[8]也在附录给出了滑动窗口法的基本实现框架，也进一步优化了计算效率。此外，文献[9]提出，采用混合坐标系可以显著减少 30%~40%的有限域运算。

然而，随着密码运算场景对实时性要求的不断提升，研究者们逐步探索出多层次的并行优化技术路径。在并行计算方向，文献[10]提出的 Montgomery 阶梯方法及其改进方法[11] [12]，开创性地实现了双线程并行计算框架，通过交替计算标量乘法的相邻步骤，为后续并行化研究提供了重要范式。并且，近期研究[13]通过 FPGA 上的流水线技术和并行乘法器设计，显著提升了倍点运算效率。文献[14]进一步提出了动态分块并行策略，通过标量的自适应分解实现多子任务并行计算，显著拓展了理论并行度。这些工作为并行椭圆曲线密码学奠定了重要基础。

本文提出一种并行滑动窗口方法，首先，该算法实行确定性任务划分，将 256 位标量 k 的高位(k_1)与低位(k_2)作为分块，既保留 SM2 变量随机性优势，又避免动态分块的调度开销。其次，算法过程中的计算资源协同，使得每个子任务内部采用滑动窗口算法优化，与文献[8]标准框架完全兼容，实现串行优化与并行计算的深度融合。实验结果表明，在 visual studio 平台上，本文方法可将 SM2 倍点运算效率提升约 36.5%。

2. 椭圆曲线上点的运算

椭圆曲线的方程形式为 $y^2 = x^3 + ax + b$，其是一种特殊的代数曲线，a 和 b 是定义曲线特征的参数。椭圆曲线密码学中的点运算主要包括点加运算和倍点运算，这些运算是实现椭圆曲线数字签名、密钥交换等密码协议的核心操作。

2.1. 点加运算

点加运算用于计算椭圆曲线上两个不同点的和，其计算复杂度与所采用的坐标系密切相关。

在仿射坐标系下，点加运算 $P_3(x_3,y_3)=P_1(x_1,y_1)+P_2(x_2,y_2)$ 的具体计算过程如公式(1)所示，其中涉及 1 次求逆运算和 3 次模乘运算。由于求逆运算在有限域中的计算开销较大，仿射坐标系下的点加运算效率较低。

$$\begin{cases} x_3=\lambda^2-x_1-x_2 \\ y_3=\lambda(x_1-x_3)-y_1 \end{cases},\quad \lambda=\frac{y_2-y_1}{x_2-x_1}. \tag{1}$$

为了提高计算效率，Jacobian 加重射影坐标系被广泛应用于椭圆曲线密码算法的实现中。在该坐标系下，点加运算 $P_3(x_3,y_3,z_3)=P_1(x_1,y_1,z_1)+P_2(x_2,y_2,z_2)$ 的计算过程如公式(2)所示，其完全避免了求逆运算。这种优化在软件实现中尤为重要，可以通过多线程技术有效提升模乘运算的执行效率。

$$\begin{cases} x_3=\lambda_5^2-\lambda_9 \\ y_3=\left(\lambda_{10}\lambda_6-\lambda_8\lambda_3^3\right)/2 \\ z_3=z_1z_2\lambda_3 \end{cases} \quad \begin{cases} \lambda_1=x_1z_2^2, & \lambda_2=x_2z_1^2 \\ \lambda_3=\lambda_1-\lambda_2, & \lambda_7=\lambda_1+\lambda_2 \\ \lambda_4=y_1z_2^2, & \lambda_5=y_2z_1^2 \\ \lambda_6=\lambda_4-\lambda_5, & \lambda_8=\lambda_4+\lambda_5 \\ \lambda_9=\lambda_7\lambda_3^2 & \\ \lambda_{10}=\lambda_9^2-2x_3 & \end{cases} \tag{2}$$

2.2. 倍点运算

倍点运算用于计算曲线上同一个点的自加操作。在仿射坐标系下，倍点运算 $P_3(x_3,y_3)=[2]P_1(x_1,y_1)$ 的实现过程如公式(3)所示。该运算需要完成 1 次有限域求逆运算和 4 次模乘运算，其中求逆运算的计算复杂度较高，往往成为性能优化的重点。

$$\begin{cases} x_3=\lambda^2-x_1-x_2 \\ y_3=\lambda(x_1-x_3)-y_1 \end{cases},\quad \lambda=\frac{3x_1^2+a}{2y_1} \tag{3}$$

而在 Jacobian 加重射影坐标系下，计算倍点运算 $P_3(x_3,y_3,z_3)=[2]P_1(x_1,y_1,z_1)$ 的计算过程如公式(4)所示。

$$\begin{cases} x_3=\lambda_1^2-2\lambda_2 \\ y_3=\lambda_1(\lambda_2-x_3)-\lambda_3 \\ z_3=2y_1z_1 \end{cases} \quad \begin{cases} \lambda_1=3x_1^2+ax_1^4 \\ \lambda_2=4x_1-y_1^2 \\ \lambda_3=8y_1^4 \end{cases} \tag{4}$$

3. 基于滑动窗口椭圆曲线倍点运算

3.1. 滑动窗口法

为了优化标量乘法运算，研究者们提出了多种算法改进方案，其中滑动窗口法因其在计算效率和存储开销之间的良好平衡而备受关注，文献[8]给出了下面的基本算法。

如算法 1 所示，该方法通过引入固定长度的处理窗口，将连续的标量位组合处理，有效减少了点加运算的次数。其核心创新在于预计算阶段构建了窗口范围内的所有奇数倍点，在主循环阶段通过窗口滑动机制实现了对标量的高效扫描和处理。这种设计使得算法在保持合理存储需求的同时，能够显著提升计算性能，特别是对于较长的标量运算具有明显优势，其算法如下所示。

Algorithm 1. Sliding window method
算法 1. 滑动窗口法

算法 1：滑动窗口法
输入：点 P，l 比特的整数 $k-\sum_{j=0}^{l-1}k_j2^j,k_j\in\{0,1\}$。
输出：$Q=[k]P$
设窗口长度 $r>1$。
预计算
a) $P_1=P,P_2=[2]P$
b) i 从 1 到 $2^{r-1}-1$ 计算 $P_{2i+1}=P_{2i-1}+P_2$。
c) 置 $j=l-1,Q=O$。
主循环
d) 当 $j\geq 0$ 执行。
若 $k_j=0$，则 $Q=[2]Q,j=j-1$。
e) 否则
令 t 是使 $j-t+l<r$ 且 k_i-1 的最小整数；
$h_j=\sum_{i=0}^{j-t}k_{t+i}2^i$；
$Q=\left[2^{j-t+1}\right]Q+P_{h_j}$；
置 $j=t-l$。
f) 输出 Q。

3.2. NAF 算法

非相邻形式(NAF)算法通过优化标量的编码方式实现高效的点运算策略。如算法 2 所示，该方法采用特殊的符号表示系统，通过消除标量二进制表示中连续的非零位来减少点加运算次数。关于 NAF 编码算法，文献[15]和[16]给出了下面的基本算法：

Algorithm 2. NAF algorithm
算法 2. NAF 算法

算法 2：NAF 算法
输入：整数 k
输出：NAF 表达式 $k=\sum_{j=0}^{l}s_j2^j,s_j\in\{-1,0,1\}$
a) $j=0$
b) 当 $k>0$ 时
若 k 是奇数，则 $s_j=2-(k \bmod 4)$
否则，
$s_j=0$。
$k=(k-s_j)/2$；
$j=j+1$。
输出 $(s_ls_{l-1}\cdots s_0)$

本章介绍了两种经典的标量乘法优化算法：滑动窗口法和非相邻形式(NAF)算法。滑动窗口法通过固定长度的处理窗口组合连续的标量位，显著减少了点加运算的次数。NAF 算法则通过优化标量的编码方式，消除二进制表示中连续的非零位，降低了运算复杂度。在后续章节中，将通过实验对比分析这两种算法与本文所设计算法的性能差异。

4. 并行动态滑动窗口椭圆曲线倍点运算

4.1. 整体架构设计

FBPSW (Fixed-Block Parallel Sliding Window)方法的整体架构设计如图 1 所示，采用分层模块化设计，主要包括标量预处理层、并行计算层和结果合并层三个核心模块，显著提升了 SM2 标量倍点效率，算法框架如图 1 所示：

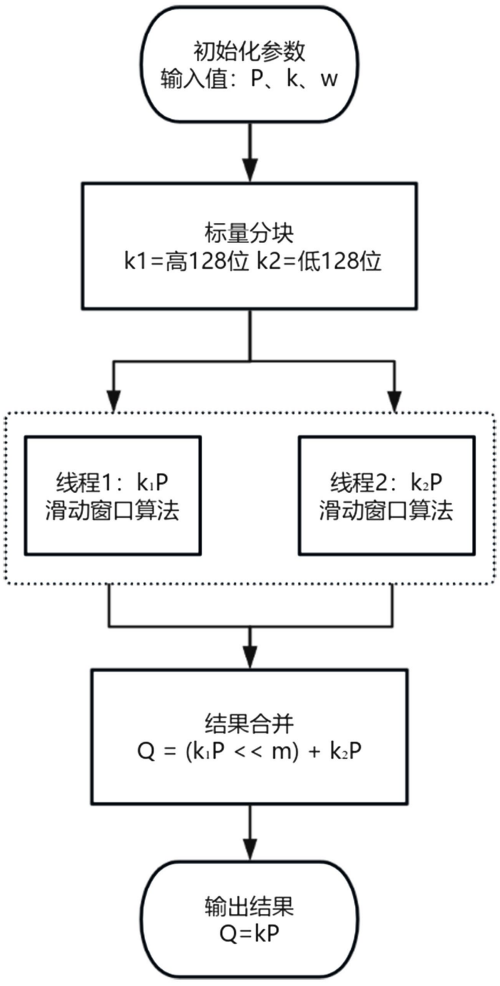

Figure 1. Framework diagram of the algorithm (FBPSW) described in this paper
图 1. 本文算法(FBPSW)框架图

4.2. 标量 k 均分策略

标量均分策略是 FBPSW 方法的并行化基础。给定 SM2 标准 256 位标量 k，采用高位优先的固定分块原则：首先将 k 的二进制表示 $\left(k_{255}k_{254}\cdots k_0\right)_2$ 划分为高 128 位和低 128 位：

$$\begin{cases} k_1 = \left(k_{255}k_{254}\cdots k_{128}\right)_2 \\ k_2 = \left(k_{127}k_{126}\cdots k_0\right)_2 \end{cases} \tag{5}$$

后续再根据分块数学关系 $k = k_1 \times 2^{128} + k_2$ 将倍点结果进行合并。均分策略通过将单一倍点任务 kP 拆分为完全独立的子任务 k_1P 和 k_2P，实现了真正的任务级并行，相较于动态分块，固定均分避免了任务调度开销，并且确定性划分方式不会引入新的侧信道漏洞。

4.3. 子任务滑动窗口

本文算法将标量 k 划分为 k_1 和 k_2 后，将总任务划分为两个子任务，每个子任务(k_1P 和 k_2P)内部采用滑动窗口算法(即算法 1)进行加速。具体而言，先开展预计算表生成操作,其中窗口宽度为 w，预计算点数目为 $2^{w-1}-1$。经过对比发现，当 $w=5$ 时，计算效率最高，故基点 P 构建窗口宽度 $w=5$ 的奇数点集 $T=\{P,3P,5P,7P,9P,11P,13P,15P\}$，该表用 Jacobian 加重射影坐标表示。点加运算和倍点运算也应用 Jacobian 加重射影坐标，以避免模逆运算，提升倍点效率。在计算过程中，算法从标量的最高有效位开始扫描，当检测到首尾两位均为非零位，且间隔 w 位时，通过查表获取对应的预计算点，并执行累加操作。

4.4. 算法

Algorithm 3. FBPSW algorithm
算法 3. FBPSW 算法

算法 3：FBPSW 算法
输入：基点 P，标量 k，窗口宽度 w
输出：倍点结果 $Q=[k]P$
a) 标量分块：
$k_1=\left(k_{255}k_{254}\cdots k_{128}\right)_2$；
$k_2=\left(k_{127}k_{126}\cdots k_0\right)_2$。
b) 并行计算：
线程 1：计算 $Q_1=[k_1]P$
执行算法 1;
线程 2：计算 $Q_2=[k_2]P$
执行算法 1。
c) 结果合并：
$m=bit\_length(k_2)$；
i 从 1 到 m 计算
$Q_2=[2]Q_1$
$Q=Q_1+Q_2$
d) 输出 Q

如算法 3 所示，本文提出了一种基于滑动窗口的并行标量乘法算法(FBPSW)。该并行算法的设计，其优势在于巧妙地规避了传统串行计算中固有的顺序依赖问题。通过将规模较大的原始计算任务分解为

两个规模显著减小的子任务，并使其能够同时执行，算法有效地将计算负载分散开来。这种任务分配策略使得在多核处理器环境下，原本可能被闲置的计算单元得以充分利用，从而从整体上缩短了运算时间。尽管在计算的最后阶段需要一个合并步骤来整合来自不同线程的中间结果，但该步骤主要由轻量级的椭圆曲线基本运算构成，其时间开销远小于一次完整的标量乘法。因此，合并过程所带来的额外成本，完全能够被并行计算所带来的巨大性能提升所抵消，最终实现了可观的综合加速效果。

5. 实验与结果分析

本章旨在通过详尽的实验评估本文提出的 FBPSW 算法的性能。实验环境为 CPU 为 Intel Core i7-9750H、频率 2.60 GHz (2.59 GHz)、RAM 为 16 GB 的计算机。开发环境为 Visual Studio，使用 CMake 构建系统并集成 BNPoint 库进行椭圆曲线运算。

5.1. 基准算法性能对比

为全面评估本文算法的性能优势，我们首先选取了五种倍点算法作为性能比较的基线，包括：1) 标准二进制展开法(即最基本的逐比特倍点算法)；2) 三种基于 NAF (非相邻形式)编码的优化算法(分别记为 NAF1、NAF2、NAF3，均来自参考文献[1])；3) 标准滑动窗口算法(基于参考文献[2])。这些算法代表了当前椭圆曲线倍点运算的主流优化技术，其中 NAF 算法通过减少非零比特数降低点加次数，而滑动窗口算法则通过窗口优化进一步减少计算量。

实验分别在 10 次和 100 次倍点运算下进行，以考察不同计算规模下的性能表现。测试数据取运行时间的平均值，以减少系统调度和背景进程的干扰。表详细列出了各算法的平均耗时(单位：毫秒，ms)及其相对于标准算法的加速比例。

Table 1. Benchmark algorithm performance time comparison
表 1. 基准算法性能时间对比

次数	单次运行时间(ms)				
	标准	NAF1	NAF2	NAF3	滑动窗口
10 次	314	219.8	223.8	236.1	164
100 次	316.5	207	203.4	216.8	154.6

从表 1 可以看出，三种 NAF 算法相较于标准算法均表现出显著的性能提升。其中，NAF2 在 100 次运算中表现最优，耗时仅 203.4 ms，提速 35.7%，而 NAF1 和 NAF3 的提升比例分别为 34%和 31.5%。这一结果与 NAF 编码的理论预期一致，即通过减少非零比特数，降低点加运算次数，从而缩短整体计算时间。然而，NAF 算法的优化效果受限于其编码方式，无法完全消除冗余计算。

相比之下，标准滑动窗口算法展现出更显著的性能优势。在 10 次和 100 次运算中，其耗时分别为 164 ms 和 154.64 ms，相较于标准算法提升比例高达 51%，成为当前对比组中性能最优的算法。这一现象可归因于滑动窗口技术的高效性——它通过预计算固定窗口内的点值，减少整体点加和倍点操作次数，从而在计算效率上超越 NAF 算法。这一结果也表明，在现有研究中，滑动窗口算法是比 NAF 更优的倍点优化方案，为后续进一步优化提供了更高的基准。

此外，观察不同运算次数下的耗时变化可以发现，标准算法和 NAF 算法的耗时随运算次数增加变化较小(如标准算法：10 次耗时 314 ms，100 次耗时 316.5 ms)，说明其计算时间主要取决于单次倍点的固定开销。而滑动窗口算法在 100 次运算时表现出更优的相对加速效果(10 次提速 48.4%，100 次提速 51%)，

这表明其优化效果在批量计算中更为显著，可能与预计算机制的摊销效应有关。这一现象也为本文后续的并行优化提供了实验依据。

5.2. 本文算法性能分析

为进一步提升倍点运算效率，本文在标准滑动窗口算法的基础上，提出了 FBPSW 算法。本节将本文算法和标准的滑动窗口算法以及预计算后的滑动窗口算法时间进行对比(如表 2)，在 10 次和 100 次倍点运算中，优化算法均展现出显著的性能优势。其中，100 次运算耗时仅 57.9 ms，较标准滑动窗口算法提速 62.5%。这一结果验证了预计算与并行化结合的协同效应——通过提前生成窗口点并分配多线程任务，有效降低了重复计算和 CPU 空闲等待时间。

Table 2. Parallel vs. serial algorithm performance time comparison
表 2. 并行串行算法性能时间对比

次数	单次运行时间(ms)		
	串行滑动窗口	并行滑动窗口(FBPSW)	提升比例
10 次	164	88	46%
100 次	154.6	57.9	62.5%

值得注意的是，随着运算次数的增加，优化算法的加速比显著提升(如预计算滑窗在 10 次和 100 次运算中分别提速 24.4%和 32.1%)，表明其更适合大规模批量计算。与基准算法相比，本文提出的优化方案在减少点加次数的同时，通过并行化进一步挖掘硬件潜力，为 SM2 签名等高频场景提供了更高效的实现路径。如图 2 和图 3，为本文算法分别在 10 次运行和 100 次运行下的计算结果及计算耗时。

Figure 2. SM2 digital signature simulation test results (10 times)
图 2. SM2 数字签名仿真测试结果(10 次)

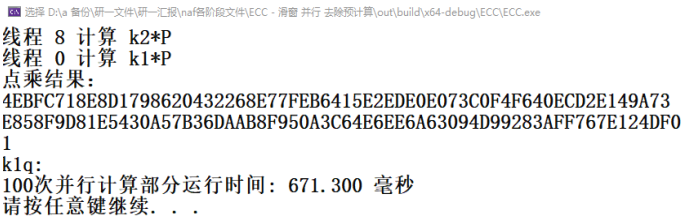

Figure 3. SM2 digital signature simulation test results (100 times)
图 3. SM2 数字签名仿真测试结果(100 次)

此外，除了时间性能的对比，基于实验观测的内存使用数据，本算法展现出优异的空间效率特性(如表 3)。如性能监测结果所示，算法执行前的工作集内存占用为 3.36 MB，页面文件使用量为 0.64 MB；执行后工作集增长至 3.96 MB，页面文件使用量增至 1.11 MB。经计算，工作集内存增量仅为 0.61 MB，页

面文件增量仅为 0.47 MB。

Table 3. Space complexity analysis of the proposed algorithm
表 3. 本文算法空间复杂度分析

各项指标	占容大小(MB)		
	算法执行前	算法执行后	变化量
工作集大小	3.35 MB	3.96 MB	+0.61 MB
峰值工作集	3.35 MB	3.97 MB	+0.62 MB
页面文件使用	0.64 MB	1.11 MB	+0.47 MB
峰值页面文件	0.68 MB	1.11 MB	+0.43 MB

实验数据表明，本算法具有常数级别的空间复杂度(O(1))，这一特征通过以下三个方面(见图 4)得到充分体现：首先，在算法执行过程中，内存使用量保持稳定，未随输入规模的增大呈现线性或多项式级增长，表明算法避免了依赖输入规模的大规模动态内存分配。其次，尽管采用了多线程并行架构，但线程创建与同步所带来的额外内存开销得到了有效控制，工作内存集的增长幅度极小，展现了优异的并行内存管理机制。

特别值得指出的是，算法运行结束后监测显示临时内存资源得到立即释放，体现了高效的内存资源管理策略。这种常数级别的空间复杂度特征使算法特别适用于内存受限的嵌入式系统环境，以及需要处理大规模输入数据的应用场景，有效避免了因内存不足导致的运行中断，为算法在实际工程中的应用奠定了坚实基础。

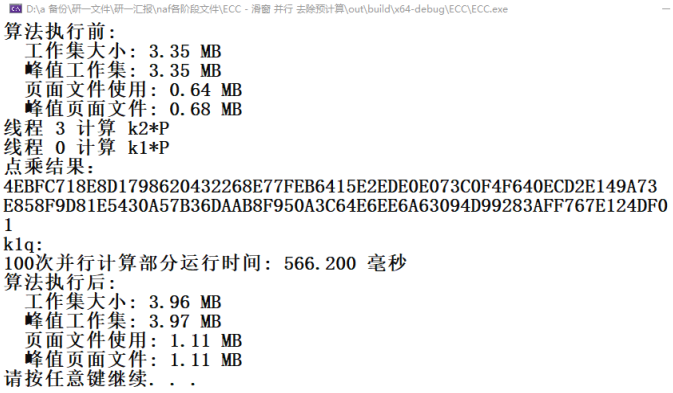

Figure 4. Pre- vs. post-algorithm data comparison
图 4. 算法执行前后数据对比

经过上述实验，证明本文算法在时间和空间复杂度上相比传统算法具有一定优势。另外，为了验证本文算法的可移植性，本研究工作配置了多个实验环境加以验证。其实验结果如表 4 所示。

Table 4. Algorithm performance under different experimental conditions
表 4. 不同实验环境下的本文算法性能

	单次运行时间(ms)			
型号 / 次数	(第 9 代)i7-9750H CPU @ 2.60GHZ	(第 11 代)i5-11400H @ 2.70GHz	(第 11 代)i5-11400H @ 2.70GHz	(第 14 代)i5-14600KF
10 次	88.00	63.95	61.06	23.92
100 次	57.90	61.20	57.76	26.80

如表所示，该算法在第 9 代、第 11 代、第 14 代的 CPU 上均具有良好性能。第 9 代和第 14 代的配置上存在一定差异，故该算法在第 14 代的 CPU 上性能表现更好。而同样是第 11 代的 CPU 上，由于内存读取速度以及其他硬件配置的差异，该算法在两个机型的性能会略有不同，但整体差距较小。

总的来说，该算法在不同 CPU 上的性能较为稳定，且算法性能与 CPU 性能配置正相关。充分说明了该算法在不同硬件配置下的可移植性较强。

6. 总结与展望

随着新一代信息技术与数字经济的高速发展，椭圆曲线密码体制正在保障信息安全方面发挥着越来越重要的作用。SM2 公钥密码算法是我国自主研究开发的商用密码标准算法，被国际标准化组织 ISO 采纳，成为国际标准 ISO/IEC 14888-3 的重要组成部分，在电子政务、金融科技、通信、医疗等领域有广泛应用。其核心运算是椭圆曲线倍点运算，也一直是密码学研究的热点[17]。

本文针对如何提升 SM2 椭圆曲线倍点运算(kP)的计算效率，提出了一种基于固定分块并行滑动窗口(FBPSW)的高性能优化方法。该方法在完全遵循 SM2 算法安全规范的前提下，通过系统性的架构优化，显著提升了倍点运算的执行效率，将 256 位标量 k 均分为高 128 位 k_1 与低 128 位 k_2，通过负载均衡优化实现并行计算[18]。在子任务内部采用滑动窗口优化(窗口宽度 $w=5$) [19]将倍点效率提高了 36.5%。

基于本文的 FBPSW 方法，未来研究工作可以从以下几个方面展开：

行业应用拓展：适配金融级密码模块(如 HSM)的安全要求，探索 FBPSW 算法与隐私计算技术的创新结合，研发适用于多方安全计算场景的优化方案，进一步拓展算法在车联网、智能电网等新兴领域的应用前景。

综上所述，本文提出的 FBPSW 方法为 SM2 倍点运算的高效实现提供了创新解决方案，通过上述研究方向的持续推进，将进一步提升算法性能、增强安全性并扩大应用范围。特别是在当前信息技术应用创新和密码算法国产化的战略背景下，这些研究工作将为构建自主可控的密码技术生态提供更好条件。

后量子安全增强：研究基于格密码的并行倍点算法，构建后量子时代的国密算法体系，并深入研究在格密码框架下保持算法并行效率的安全证明方法，为即将到来的量子计算时代做好密码技术储备[20]。

基金项目

国家自然科学基金(62472040)；北京市自然科学基金(L251065)；中国版权保护中心版权研究课题(BQ2024017)；2025 年京港澳高校交流项目(11000025T000003319739)。

参考文献

[1] Ullah, S., Zheng, J., Din, N., Hussain, M.T., Ullah, F. and Yousaf, M. (2023) Elliptic Curve Cryptography; Applications, Challenges, Recent Advances, and Future Trends: A Comprehensive Survey. *Computer Science Review*, **47**, Article ID: 100530. https://doi.org/10.1016/j.cosrev.2022.100530

[2] 胡晓辉, 巩俊辉, 徐宁, 等. 跨层优化的 WSN 能耗均衡拓扑博弈算法[J]. 计算机工程与应用, 2019, 55(14): 69-75.

[3] 杨晓秋. 高性能椭圆曲线标量乘算法的研究[D]: [硕士学位论文]. 哈尔滨: 哈尔滨理工大学, 2023.

[4] 兰修文. ECC 计算算法的优化及其在 SM2 实现中的运用[D]: [硕士学位论文]. 成都: 电子科技大学, 2019.

[5] 黄世中, 羊红光. NAF 编码方法的分析与应用[J]. 信息网络安全, 2012(5): 4-6+35.

[6] 赵石磊, 杨晓秋, 刘志伟. 一种低复杂度的改进 wNAF 标量乘算法[J]. 电子学报, 2022, 50(4): 977-983.

[7] Hai, H., Ning, N., Lin, X., Zhiwei, L., Bin, Y. and Shilei, Z. (2021) An Improved wNAF Scalar-Multiplication Algorithm with Low Computational Complexity by Using Prime Precomputation. *IEEE Access*, **9**, 31546-31552. https://doi.org/10.1109/access.2021.3061124

[8] 国家密码管理局. GB/T 32918.1-2016.SM2 椭圆曲线公钥密码算法 第 1 部分: 总则[S]. 北京: 中国标准出版社,

2016.

[9] Cohen, H., Miyaji, A. and Ono, T. (1998) Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In: Ohta, K. and Pei, D.Y., Eds., *Advances in Cryptology—ASIACRYPT*'98, Springer, 51-65. https://doi.org/10.1007/3-540-49649-1_6

[10] Okeya, K. and Sakurai, K. (2001) Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the Y-Coordinate on a Montgomery-Form Elliptic Curve. In: *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 126-141. https://doi.org/10.1007/3-540-44709-1_12

[11] 于伟, 李宝, 王鲲鹏, 等. 特 3 有限域上椭圆曲线的 co-Z Montgomery 算法[J]. 计算机学报, 2017, 40(5): 1121-1133.

[12] 刘双根, 王蓉蓉, 李圣雨. GF(3m)上 Hessian 曲线的三进制 Montgomery 算法[J]. 山东大学学报(理学版), 2019, 54(1): 96-102.

[13] 谭光昭, 周骅. 面向物联网安全的素域 SM2 倍点硬件优化[J]. 运筹与模糊学, 2023, 13(6): 7247-7255.

[14] Longa, P. and Sica, F. (2012) Four-Dimensional Gallant-Lambert-Vanstone Scalar Multiplication. In: *Advances in Cryptology—ASIACRYPT* 2011, Springer, 718-739. https://doi.org/10.1007/978-3-642-34961-4_43

[15] 王学理, 斐定一. 椭圆与超椭圆曲线公钥密码的理论与实现[M]. 北京: 科学出版社, 2006: 462-463.

[16] Solinas, J.A. (2000) Efficient Arithmetic on Koblitz Curves. *Designs*, *Codes and Cryptography*, **19**, 195-249. https://doi.org/10.1023/a:1008306223194

[17] Cui, Y., Liu, Q., Yao, Y., Xu, X., Wu, W. and Xu, X. (2023) An Area-Efficient and Low-Latency Elliptic Curve Scalar Multiplication Accelerator over Prime Field. *Microprocessors and Microsystems*, **103**, Article ID: 104944. https://doi.org/10.1016/j.micpro.2023.104944

[18] Joye, M. and Yen, S.M. (2000) Optimal Left-to-Right Binary Signed-Digit Recoding. *IEEE Transactions on Computers*, **49**, 740-748. https://doi.org/10.1109/12.863044

[19] Li, M., Li, N., Liu, H., Cheng, S., Hu, X. and Li, J. (2023). Implementation of a Safe and Efficient Point Multiplication for SM2 Algorithm. 2023 *IEEE 6th Information Technology*, *Networking*, *Electronic and Automation Control Conference* (*ITNEC*), Chongqing, 24-26 February 2023, 137-141. https://doi.org/10.1109/itnec56291.2023.10082311

[20] Alkim, E., Ducas, L., Pöppelmann, T., *et al*. (2016) Post-Quantum Key {Exchange-A} New Hope. 25*th USENIX Security Symposium* (*USENIX Security* 16), Austin, 10-12 August 2016, 327-343.