Computer Science and Application 计算机科学与应用, 2025, 15(4), 134-144
Published Online April 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.154086

Hans汉斯

基于FPGA的SM9数字签名算法的快速实现

梅　峰，李子臣*

北京印刷学院信息工程学院，北京

收稿日期：2025年3月8日；录用日期：2025年4月7日；发布日期：2025年4月15日

摘　要

SM9算法是由我国推出的基于双线性对的标识密码算法，其中使用的R-ate双线性对在计算效率上优于Weil对和Tate对，但在实际应用中仍有不足。针对上述问题，为了提升SM9中双线性对的计算效率，提出了一种在FPGA平台上使用滑动窗口改进R-ate对运算的方法。该方法以蒙哥马利模乘为基础改进扩域模逆运算并通过滑动窗口算法优化R-ate双线性对运算中的Miller循环部分。同时根据蒙哥马利模乘特性减少椭圆曲线上点运算计算量提升签名算法运算效率。仿真结果表明，采用该方法可以将SM9中R-ate对的运算效率提升约18.46%，对SM9签名算法效率提升约为13.55%。

关键词

SM9，R-ate对，蒙哥马利模乘，滑动窗口算法

Fast Implementation of SM9 Digital Signature Algorithm Based on FPGA

Feng Mei, Zichen Li*

School of Information Engineering, Beijing Institute of Graphic Communication, Beijing

Received: Mar. 8th, 2025; accepted: Apr. 7th, 2025; published: Apr. 15th, 2025

Abstract

SM9 algorithm is an identity-based cryptographic algorithm based on bilinear pairing introduced by our country, in which the R-ate pairing used is better than Weil pairing and Tate pairing in terms of computational efficiency, but it still has deficiencies in practical applications. To improve the computational efficiency of the bilinear pairing in SM9, a method was proposed to improve the R-ate pairing operation by using a sliding window on the FPGA platform. The method was based on the Montgomery modular multiplication to enhance the modular inverse operation in the twelfth

*通讯作者。

文章引用: 梅峰, 李子臣. 基于 FPGA 的 SM9 数字签名算法的快速实现[J]. 计算机科学与应用, 2025, 15(4): 134-144.
DOI: 10.12677/csa.2025.154086

extension field and optimize the Miller loop in the R-ate pairing operation through the sliding window algorithm. At the same time, the computational efficiency of the signature algorithm was improved by reducing the number of point operations on elliptic curves according to the Montgomery modular multiplication property. Simulation results showed that this method could improve the computational efficiency of R-ate pairing in SM9 by approximately 18.46%, and the efficiency of the SM9 signature algorithm by approximately 13.55%.

Keywords

SM9, R-ate Pairing, Montgomery Modular Multiplication, Sliding Window Algorithm

1. 引言

在现代密码学中，双线性对(Bilinear Pairing)技术因其广泛的应用场景和独特的数学特性，成为众多密码学协议的核心组成部分。双线性对的应用涵盖了身份基加密[1] (Identity-based cryptography, IBC)、签名方案[2]-[6]和各种基于配对的密码协议[7] [8]。其中，R-ate 对[9]作为一种优化的双线性配对，逐渐引起了研究者们的广泛关注。R-ate 对相较于传统的 Weil 对[10]和 Tate 对[11]，具有更高的计算效率和更优的性能表现。这使得 R-ate 对在高性能需求的密码学应用中显示出巨大潜力。然而，尽管 R-ate 对已经在理论和实现方面取得了显著进展，其在实际应用中的计算效率仍不能完全满足实际需求。因此，如何进一步优化双线性对的计算过程，成为当前研究的重要方向。

研究者们为了提升双线性对的计算效率，探索并提出了多种创新策略。其中包括：通过将最终模幂的指数进行分解[12] [13]，利用模乘及 Frobenius 映射进行计算，降低了计算复杂度。通过采用不同坐标系[14] (如仿射坐标系和雅克比坐标系)，改进椭圆曲线上点的运算以优化配对运算。提出一种利用分母约化技术[15]来提升运算效率的方法。从算法角度考虑，将双线性对运算中参数采用非相邻表示型[16] (non-adjacent form, NAF)算法展开，降低参数汉明重量以减少计算量。此外，有学者突破传统计算框架，设计出一种椭圆网算法[17]替代 Miller 算法[18]用于双线性对的计算。也有学者通过将双基数链[19]应用到双线性对的计算中，有效降低了 Tate 对运算的计算复杂度。

自 2016 年国家密码管理局发布《SM9 标识密码算法》[20]以来，为了促进 SM9 算法的普及与实际应用，国内学者提出了多种针对 SM9 中 R-ate 对的运算优化方案。在 SM9 算法流程中，算法的安全性依赖于 R-ate 双线性对的计算，相应的 R-ate 对的计算量占据整个算法中很大一部分。因此许多学者都在探索如何能够在不损失双线性对安全性的同时提升其计算的效率。为此有学者提出在算法底层通过改变 R-ate 对计算中同构映射的操作顺序[21]，将大部分逆元运算从大特征域转换到小特征域，减少了逆元求解的计算消耗。对于双线性对映射中群的运算提出了一系列有效的优化方法[22]以加速双线性算法的执行，包括 NAF 加速、蒙哥马利域下运算和十二次扩域稀疏乘法等。而文献[23]设计了一种十二次扩域的等价转换方法，提出了一种针对椭圆曲线点乘的稀疏乘法，从而减少了有限域乘法的数量。文献[24]则对运算中 Miller 循环、最终模幂等步骤进行优化并在硬件上进行实现。尽管已有多种针对 SM9 算法的优化方案[21]-[25]，但距离算法的普及使用仍有一定差距，仍需要更多工作以实现 SM9 算法的效率提升。

本文以 Jacobian 加重射影坐标系为基础，底层运算采用 Montgomery 模乘以及辗转相除法，利用滑

动窗口算法改进 R-ate 对运算中 Miller 循环部分计算，有效降低 R-ate 对运算过程中线函数运算以及椭圆曲线上点加运算的数量。在此基础上利用 Montgomery 特性优化椭圆曲线上点乘计算，减少计算开销，最终得到一种在 FPGA 平台上快速计算 R-ate 对以及 SM9 数字签名的方法，并通过仿真实验证明其正确性。最后，给出算法的计算量分析以及效率评估。

2. 基础知识

2.1. Miller 算法

目前计算配对的算法主要有 Miller 算法以及椭圆网算法。椭圆网算法优势在于计算过程中无需求逆操作，其计算同样适用于 Weil 对、Tate 对，但是运算效率相对于 Miller 算法仍然偏低。并且 Miller 算法的使用范围要比椭圆网算法更加广泛。Miller 函数 $f_{T,Q}$ 满足的迭代关系如下：

$$f_{m+n,Q} = f_{m,Q} f_{n,Q} \frac{g_{[m]Q,[n]Q}}{g_{[m+n]Q}} \tag{1}$$

其中 $g_{[m]Q,[n]Q}$ 表示过点 $[m]Q$ 和 $[n]Q$ 的直线方程，$g_{[m+n]Q}$ 表示过点 $[m+n]Q$ 的垂线方程。Miller 算法则是采用迭代关系中 $n = m$ 以及 $n = 1$ 的两种特殊情况，将 Miller 函数按照二进制展开的形式求解。

2.2. R-ate 对

R-ate 对作为 SM9 中最常用的双线性对，可以看作是在 Tate 对基础上优化幂次的配对。在计算过程中大多采用 Miller 算法进行运算，R-ate 对具体计算如算法 1 所示。

算法 1：R-ate 对的计算

输入：$P \in E(F_q)[r],\ Q \in E'(F_{q^2})[r],\ a = 6t+2$

输出：$R_a(Q, P)$

1) 设 $a = \sum_{i=0}^{L-1} a_i 2^i,\ a_{L-1} = 1$

2) 置 $f = 1$，$T = Q$

3) For i from L-2 to 0 do：

4) 计算 $f = f^2 \cdot g_{T,T}(P)$，$T = [2]T$

5) if $a_i = 1$

6) 计算 $f = f \cdot g_{T,Q}(P)$，$T = T + Q$

7) end if

8) end for

9) 计算 $Q_1 = \pi_q(Q),\ Q_2 = \pi_{q^2}(Q)$

10) 计算 $f = f \cdot g_{T,Q_1}(P),\ T = T + Q_1$

11) 计算 $f = f \cdot g_{T,-Q_2}(P), T = T - Q_2$

12) 计算 $f = f^{(q^{12}-1)/r}$

13) 输出 f

2.3. SM9 数字签名算法

根据 GM/T 0044.2-2016 中 SM9 签名算法如算法 2 所示，其中待签名的消息为比特串 M，消息 M 的数字签名为(h, S)，签名主公钥为 $P_{pub\text{-}s}$，签名密钥为 ds_A，群 G_1 生成元为 P_1，密码杂凑函数一般使用 SM3

算法。

算法 2：SM9 数字签名算法

输入：$M, P_{pub\text{-}s}, dsA$

输出：(h, S)

1) 计算群 G_T 中的元素 $g = e(P_1, P_{pub\text{-}s})$

2) while $l = 0$ do

3) 产生随机数 $r \in [1, N-1]$

4) 计算群 G_T 中的元素 $w = g^r$

5) 计算整数 $h = H_2(M \| w, N)$

6) 计算整数 $l = (r - h) \bmod N$

7) end while

8) 计算 $S = [l]\, dsA$

9) 确定数字签名(h, S)

3. 二次扩域的优化模逆算法

SM9 算法作为基于椭圆曲线的密码算法，其底层运算主要为有限域模运算，在此基础上按照 1-2-4-12 的塔式扩张规则构成扩域运算。有限域模运算主要包括模乘运算、模加减运算以及模逆运算。其中模乘运算可以采用 Montgomery 模乘算法进行运算；模逆运算可以采用辗转相除法进行运算。扩域上的运算可以基于 Karatsuba 思想，将运算中的乘积项转换为运算中间项的加减组合，从而减少模乘运算的使用次数，变相转换为模加减运算，从而减少运算量。

在有限域运算的基础上构建扩域运算可以根据有限域算法特性以改进运算过程，以二次扩域模逆运算为例，假设存在二次扩域元素 $a = a_0 + a_1u$，其中 $a_0, a_1 \in F_q$，其对应的逆为 $c = c_0 + c_1u$，其中 $c_0, c_1 \in F_q$。根据塔式扩张运算规则 $c_0 = a_0/(a_0^2 + 2a_1^2)$，$c_1 = -a_1/(a_0^2 + 2a_1^2)$。传统计算方式一般先计算分母，对分母求逆后乘上各元素对应分子得到结果。但由于 Montgomery 模乘算法相对比传统模乘运算计算过程可以分为模乘和形式转换两个部分，即对于待乘数 a 和 b，第一步运算得到结果 abR^{-1}，第二步将结果与 R^2 再运算得到模乘结果。因此可以将二次扩域模逆运算中涉及到有限域模乘的运算步骤进行拆分以减少计算量。具体运算流程如算法 3 所示。

算法 3：二次扩域模逆算法

输入：$a = a_0 + a_1u$，其中 $a_0, a_1 \in F_q$

输出：$c = c_0 + c_1u$

1) $m = a_0^2R^{-1}$

2) $n = a_1^2R^{-1}$

3) $n = n + n$

4) $m = m + n$

5) $m = m^{-1}$

6) $c_0 = a_0 \cdot m$

7) $c_1 = a_1 \cdot m$

8) $c_1 = -c_1$

通过算法 3 在计算分母过程中使用素数域元素直接进行 Montgomery 运算得到素数域数值带 R^{-1} 的结果，然后在分母求逆的过程将数值转换为 Montgomery 形式之后再与对应分子相乘得到素数域结果，这样

利用 Montgomery 模乘的特性，可以减少常规运算流程中进行 Montgomery 形式转换的步骤。假设以 A, M, I 分别代表有限域模加减运算、模乘运算以及模逆运算所需计算量，其中 M 为两次 Montgomery 运算，则有限域及扩域各运算所需计算量如表 1 所示。其中二次扩域模逆运算的计算量仅需 2M + 3A + I，相比于文献[23]中的 2M + 2S + I + 3A 以及文献[24]中的 4M + 2I + 3A 在二次扩域模逆上的运算效率都要更高。

Table 1. Computation analysis
表 1. 计算量分析

	模加减	模乘	模平方	模逆
有限域	A	M	M	I
二次扩域	2A	3M + 3A	2M + 4A	2M + 3A + I
四次扩域	4A	9M + 21A	60M + 20A	14M + 19A + I
十二次扩域	~	54M + 190A	36M + 172A	113M + 196 + I

4. SM9 数字签名算法快速实现

对于椭圆曲线上倍点运算，在传统仿射坐标系下运算简便但涉及模逆运算，因此计算效率低，而在雅克比坐标系下则可以节省模逆运算，将坐标点的求解转换为模加减运算以及模乘运算，计算效率更高。对于椭圆曲线上点加运算，在传统仿射坐标与雅克比坐标混合下计算效率要高于仅使用其中一种坐标系，这是由于传统仿射系仍旧需要进行模逆运算，而仅使用雅克比坐标系要比使用混合坐标系所需计算的参数更多。

除此之外，在双线性对的运算过程中椭圆曲线上点的运算总是与直线函数的运算成对出现，因此可以将椭圆曲线上点的运算与直线函数运算结合为一个整体进行计算。表 2 中总结了椭圆曲线上点的运算以及直线函数运算在优化前后所需运算数量。

Table 2. Computational analysis of points on elliptic curves
表 2. 椭圆曲线上点计算量分析

	F_{p^2} 平方	F_{p^2} 乘法	F_{p^2} 加减	F_p 乘法
$T=[2]T$	2	6	4	10
$T=T+Q$	4	15	6	0
$g_{T,T}(P)$	2	5	5	8
$g_{T,Q}(P)$	2	12	5	4
优化后倍点	5	6	15	2
优化后点加	3	11	8	2

4.1. 扩域元素乘法设计

R-ate 对的计算在 Miller 循环部分涉及大量直线函数与十二次扩域上元素乘法运算，但是直线函数运算结果在六个维度上的具体数值均为 0。因此根据直线函数的稀疏特点，采用稀疏乘法算法以降低十二次扩域元素与直线函数之间计算的复杂度，减少不必要计算，可以提升计算效率。

根据稀疏乘法，假设存在十二次扩域元素 $f=f_0+f_1\cdot w+f_2\cdot w_2$ 以及一个直线函数运算结果 $g=l_0+l_1\cdot v+l_2\cdot w_2$ 其中 $f_i\in F_{q4}$，$l_i\in F_{q2}$，元素的乘法结果如下：

$$\begin{aligned} f\cdot g &= \left(f_0+f_1\cdot w+f_2\cdot w^2\right)\cdot\left(l_0+l_1\cdot v+l_2\cdot w^2\right) \\ &= \left[f_0l_0+\left(f_0l_1+f_1l_2\right)v\right]+\left[f_1l_0+\left(f_1l_1+f_2l_2\right)v\right]w+\left(f_0l_2+f_2l_0+f_2l_1v\right)w^2 \end{aligned} \tag{2}$$

根据上式计算十二次扩域元素以及直线函数的乘法运算要相对原有十二次扩域乘法运算涉及维度更低，拆解到二次扩域上所需运算量更少，运算后结果同样为十二次扩域元素，不影响后续运算。相比于传统十二次扩域乘法运算需要 55M + 190A， $f\cdot g$ 稀疏乘法运算可以将计算量节省至 39M + 99A。

4.2. 基于滑动窗口的 R-ate 对快速实现

传统 R-ate 对计算采用 Miller 算法，即通过算法 1 中步骤 4 以及步骤 6 迭代计算实现，进而可以转换为一种类似标量乘的运算。然而在 R-ate 对计算过程中部分参数可以省略以节省运算时间，如采用 NAF 算法降低参数的汉明重量以减少点加的次数。假设参数为 a，则其对应的 NAF 形式 a'如下所示。

$$a = 1001000000000000000000000000000000000000000100001010111011001001111 10$$

$$a' = 1001000000000000000000000000000000000000000100010\,\bar{1}0\,\bar{1}000\,\bar{1}0\,\bar{1}001010000\,\bar{1}0$$

在计算 R-ate 对的过程中遍历参数，当前遍历位为零则进行步骤 4，为非零位时多进行一次步骤 6。在此基础之上，可以采用滑动窗口算法改进 R-ate 对计算方法，通过预计算存储迭代过程中所涉及到的椭圆曲线上点的坐标以及对应的 Miller 函数值 f，进一步降低 Miller 循环过程中所涉及点加运算的计算次数，从而提升 R-ate 对的计算效率，则可以将参数 a 改进为适用于滑动窗口的形式 a''，以窗口长度 r = 3 为例，参数形式可以理解为如下所示：

$$a'' = 1001000000000000000000000000000000000000000100000050007003001000070 30$$

由此遍历参数，将遍历非零位的计算改进为与对应数值所存储参数之间的计算，进而得到一种基于滑动窗口改进的计算 R-ate 双线性对的快速实现方法，具体实现算法如算法 4 所示。其中步骤 24 中模幂部分采用现有方法拆分为困难部分以及简单部分，通过加法链的方法来计算困难部分。

算法 4：基于滑动窗口的 R-ate 对的计算

输入： $P\in E\left(F_q\right)[r]$, $Q\in E'\left(F_{q^2}\right)[r]$, $a=6t+2$

输出： $R_a(Q,P)$

1) 设窗口长度 $r>1$

2) 预计算 $f_1=1$, $T_1=Q$; $f_2=g_{Q,Q}(P)$, $T_2=[2]Q$

3) For i from 1 to $2^{r-1}-1$ do:

4) $f_{2i+1}=f_{2i-1}f_2g_{[2i-1]Q,[2]Q}(P)$; $T_{2i+1}=T_{2i-1}+T_2$

5) end for

6) 设 $a=\sum_{i=0}^{L-1}a_i2^i$, $a_{L-1}=1$

7) 置 $f=1, T=Q, j=L-2$

8) While $j\geq 0$ do:

9) if $a_i=0$

10) 计算 $f=f^2\cdot g_{T,T}(P)$， $T=[2]T$

11) else

12) 令 t 为 $j-t+1\geq r$ 且 $a_t=1$ 的最小整数

13) $h_j=\sum_{i=0}^{j-t}a_{t+i}2^i$

14) For i from $j-t+1$ to 0 do

15) $f=f^2\cdot g_{T,T}(P)$，$T=[2]T$

16) end for

17) $f\leftarrow f\cdot f_{T_{h_j}}g_{T,T_{h_j}}(P);\ T\leftarrow T+T_{h_j}$

18) 置 $j=t-1$

19) end if

20) end while

21) 计算 $Q_1=\pi_q(Q),\ Q_2=\pi_{q^2}(Q)$

22) 计算 $f=f\cdot g_{T,Q_1}(P),\ T=T+Q_1$

23) 计算 $f=f\cdot g_{T,-Q_2}(P),\ T=T-Q_2$

24) 计算 $f=f^{\left(q^{12}-1\right)/r}$

25) 输出 f

采用算法 4 在窗口长度为 3 的情况下可以将 R-ate 对运算过程中点加运算以及直线函数的运算数量从传统 Miller 算法中的 16 次降低至 7 次，相对比 NAF 算法进一步减少 4 次点加运算以及稀疏乘法运算，相对地只增加了 2 次模乘运算。具体算法所对应运算数量如表 3 所示。

Table 3. The Miller loop uses the number of operations
表 3. Miller 循环调用模块量

方法	点加－直线函数	fg	ff
传统方法	16	16	0
NAF 方法	11	11	0
滑动窗口方法	7	7	2

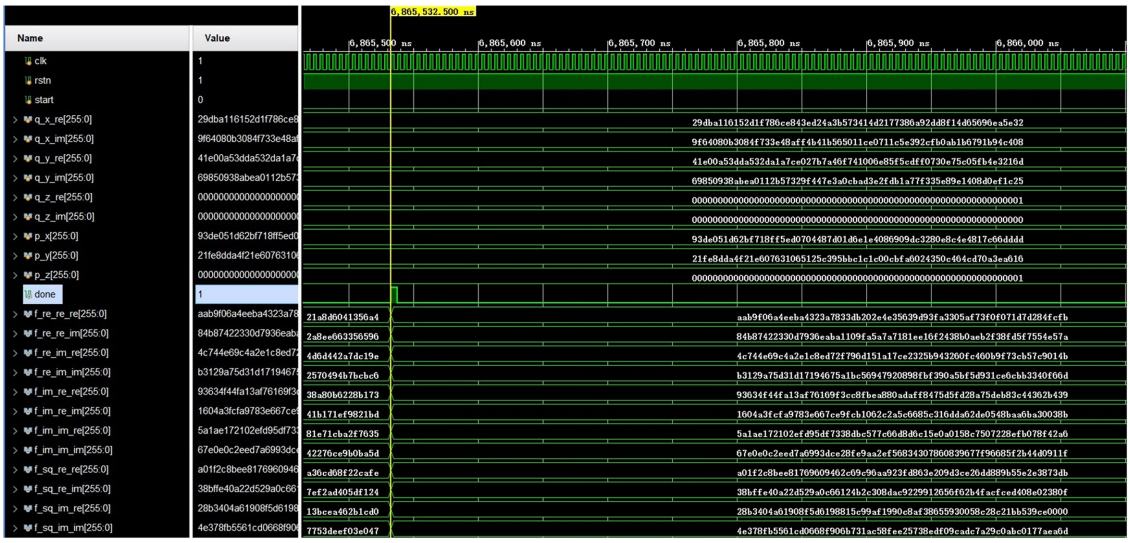

Figure 1. Graph of the results of the R-ate pairing simulation
图 1. R-ate 对仿真模拟结果图

根据表 1 对比计算量分析可知，滑动窗口算法相比于传统算法节省 549M + 1159A，相比于 NAF 算

法节省 184M + 304A。由此说明本文提出的基于滑动窗口的 R-ate 对计算方法在优化效果上要优于使用 NAF 改进的 R-ate 对计算方法。为了验证提出的实现方法的正确性，本文在 Vivado 2018.3 平台，使用 verilog 语言进行编译对实现方法进行仿真验证。在底层运算均相同的情况下，采用串行运算条件下，本文算法实现后仿真模拟结果如图 1 所示，进行一次双线性对运算所需时间为 6.865 ms，相比于优化前运算时间 8.42 ms 计算效率提升约为 18.46%。

4.3. 基于 Montgomery 的点乘算法快速实现

根据第 2 章提到的 Montgomery 模乘的特性，可以采用相同方法改进 SM9 算法中 G_1 群中的倍点运算以及点加运算。将椭圆曲线上点的运算转换为蒙哥马利形式进行运算，减少采用模乘运算过程中参数形式反复转换的步骤，进而提升椭圆曲线上倍点运算以及点加运算的计算效率。具体改进后倍点运算计算过程如算法 5 所示，点加运算计算过程如算法 6 所示。

算法 5：G_1 群倍点运算

输入：$P_1 = (X_1, Y_1, Z_1)$

输出：$P_3 = (X_3, Y_3, Z_3)$

1) 将坐标点 X_1、Y_1 分别转换为 Montgomery 形式 X_1R、Y_1R

2) $T_1 = (Y_1R)^2$

3) $T_2 = T_1 \cdot X_1\mathrm{R}$

4) $T_2 = 4 \cdot T_2$

5) $T_3 = (T_1)^2$

6) $T_3 = 8 \cdot T_3$

7) $T_1 = X_1R \cdot X_1R$

8) $T_1 = 3 \cdot T_1$

9) $X_3 = (T_1)^2$

10) $X_3 = X_3 - T_2$

11) $X_3 = X_3 - T_2$

12) $Y_3 = T_2 - X_3$

13) $Y_3 = Y_3 \cdot T_1$

14) $Y_3 = Y_3 - T_3$，将 X_3、Y_3 转换为素数域形式

15) $Z_3 = Y_1R \cdot Z_1$

16) $Z_3 = Z_3 + Z_3$

算法 5 中，在不改变变量之间映射关系的条件下，将原有的有限域模乘算法拆分为更小的 Montgomery 模乘基础运算，通过将变量在素数域以及 Montgomery 域之间来回转换以减少原来算法中的冗余计算，进而提高倍点运算的计算效率。对比分析可知，通过算法 5 可以将原有倍点运算的计算量由 7M + 12A 减少至 5.5M + 12A。

算法 6：G_1 群点加运算

输入：$P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2)$

输出：$P_3 = (X_3, Y_3, Z_3)$

1) 将坐标点 Z_1 转换为 Montgomery 形式 Z_1R

2) $T_1 = (Z_1R)^2$

3) $T_2 = T_1 \cdot Z_1\mathrm{R}$

4) $T_1 = T_1 \cdot X_2$

5) $T_2 = T_2 \cdot Y_2$

6) $T_1 = T_1 - X_1$

7) $T_2 = T_2 - Y_1$

8) $Z_3 = T_1 \cdot Z_1 R$

9) 将 T_1 转换为 Montgomery 形式赋值给 T_5

10) $T_3 = (T_5)^2$

11) $T_4 = T_3 \cdot T_5$

12) $T_3 = T_3 \cdot X_1$

13) $T_1 = T_3 + T_3$

14) 将 T_2 转换为 Montgomery 形式赋值给 T_2

15) $X_3 = (T_2)^2$

16) $X_3 = X_3 - T_4$

17) 将 X_3 转换为 Montgomery 形式赋值给 X_3

18) $X_3 = X_3 - T_1$

19) $T_3 = T_3 - X_3$

20) $T_3 = T_3 \cdot T_2$

21) $T_4 = Y_1 \cdot T_4$

22) $Y_3 = T_3 - T_4$

算法 6 采用算法 5 相同原理，通过两种形式的转换，可以将点加运算的计算量由 11M + 7A 下降至 7.5M + 7A。在此基础上，通过 NAF 算法对进行点乘运算的参数进行改进可以进一步减少点乘过程中调用点加运算的次数，进而提升计算效率。

5. 实验结果

本文提出的快速实现通过 Vivado 2018.3 在 Xilinx FPGA 上基于 verilog 语言完成综合实现。通过第 3 章对比可知，采用文本提出的算法在串行条件下进行 R-ate 双线性对的计算可以将一次 R-ate 对运算时间由 8.42 ms 下降至 6.865 ms，整体效率提升约为 18.46%，该方法同样可以在其他串行运算条件上使用，可以将 R-ate 对 Miller 循环部分中进行点加运算、直线函数运算以及稀疏乘法的次数由 16 次下降到 7 次，提升效果相对于原有 NAF 改进方法更高。

在此基础上，提出根据 Montgomery 模乘特性改进有限域模逆运算、G_1 群中点加运算以及倍点运算，同时采用 NAF 方法进一步减少点乘运算参数的汉明重量，以提升计算效率。通过在 Vivado 平台采用相同底层算法分别实现现有文献以及本文的快速实现方法，各模块运算所需时间如表 4 所示。

Table 4. Operation time of each module (μs)
表 4. 各模块运算所需时间(μs)

	二次扩域模逆	G_1 群倍点运算	G_1 群点加运算	G_1 群点乘运算
改进前	4.2145	2.6625	4.0525	1156.535
改进后	2.4775	2.0675	2.7175	731.058

最后，本文根据《SM9 标识密码算法参数定义》使用 verilog 语言实现 SM9 数字签名算法。实验结

果表明，采用本文优化后的椭圆曲线运算方法，可以将 G_1 群倍点运算和点加运算效率分别提升 22.35%以及 32.94%，结合 NAF 算法优化点乘运算后，可以将 SM9 数字签名过程中的点乘运算效率提升约 36.79%。在此基础上结合前文提出的基于滑动窗口改进的 R-ate 对计算方法，可以将串行条件下 SM9 数字签名算法的运算效率提升约 13.55%。其中签名中各运算所需时间如表 5 所示，数字签名实验仿真结果如图 2 所示。

Table 5. Each operation time in SM9 digital signature (ms)
表 5. SM9 数字签名中各运算所需时间(ms)

	R-ate 对	模幂运算	点乘运算	SM9 数字签名
改进前	8.421	5.484	1.156	15.144
改进后	6.865	5.484	0.732	13.092

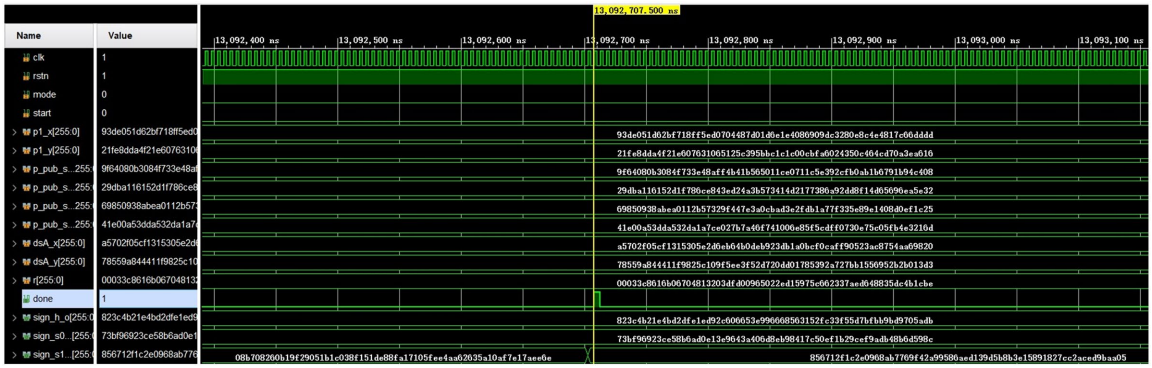

Figure 2. SM9 digital signature simulation result
图 2. SM9 数字签名仿真模拟结果图

6. 结论

R-ate 对的计算效率对 SM9 算法的广泛普及以及配对在密码学中的应用具有重要意义，本文通过对 R-ate 对的运算过程进行分析，提出一种基于滑动窗口的 R-ate 对快速实现方法，其中包括基于滑动窗口算法优化 R-ate 对计算中 Miller 循环部分以及使用 Montgomery 特性改进底层椭圆曲线上点的计算和扩域运算。通过将原有运算复杂度与改进后算法进行分析，证明本文采用实现方法对 R-ate 对运算效率的提升有效。实验结果证明，本文所提出的实现方法可以将 R-ate 对以及 SM9 数字签名算法的计算效率分别提升约 18.46%以及 13.55%。

基金项目

国家自然科学基金(62472040)；中国版权保护中心版权研究课题(BQ2024017)；北京市教委科研计划资助(No. KM202110015004)；北京市高等教育学会 2022 年立项面上课题(MS2022093)；北京市教育委员会科学研究计划项目资助(KM202310015002)。

参考文献

[1] Shamir, A. (2000) Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R. and Chaum, D., Eds., *Advances in Cryptology*, Springer, 47-53. https://doi.org/10.1007/3-540-39568-7_5

[2] Boneh, D., Lynn, B. and Shacham, H. (2001) Short Signatures from the Weil Pairing. In: Boyd, C., Ed., *Advances in Cryptology—ASIACRYPT* 2001, Springer, 514-532. https://doi.org/10.1007/3-540-45682-1_30

[3] Chen, X., Zhang, F. and Kim, K. (2006) New ID-Based Group Signature from Pairings. *Journal of Electronics* (*China*), **23**, 892-900. https://doi.org/10.1007/s11767-005-0065-2

[4] Zhang, F. and Kim, K. (2002) ID-Based Blind Signature and Ring Signature from Pairings. In: Zheng, Y.L., Ed., *Advances in Cryptology—ASIACRYPT* 2002, Springer, 533-547. https://doi.org/10.1007/3-540-36178-2_33

[5] Lin, C.Y. and Wu, T.C. (2004) An Identity-Based Ring Signature Scheme from Bilinear Pairings. 18*th International Conference on Advanced Information Networking and Applications*, Fukuoka, 29-31 March 2004, 182-185. https://doi.org/10.1109/aina.2004.1283782

[6] Boneh, D., Gentry, C., Lynn, B. and Shacham, H. (2003) Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: Biham, E., Ed., *Advances in Cryptology—EUROCRYPT* 2003, Springer, 416-432. https://doi.org/10.1007/3-540-39200-9_26

[7] Joux, A. (2000) A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W., Ed., *Algorithmic Number Theory*, Springer, 385-393. https://doi.org/10.1007/10722028_23

[8] Chen, L., Cheng, Z. and Smart, N.P. (2007) Identity-Based Key Agreement Protocols from Pairings. *International Journal of Information Security*, **6**, 213-241. https://doi.org/10.1007/s10207-006-0011-9

[9] Lee, E., Lee, H. and Park, C. (2009) Efficient and Generalized Pairing Computation on Abelian Varieties. *IEEE Transactions on Information Theory*, **55**, 1793-1803. https://doi.org/10.1109/tit.2009.2013048

[10] Menezes, A., Vanstone, S. and Okamoto, T. (1991) Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *Proceedings of the* 23*rd Annual ACM Symposium on Theory of Computing*, New Orleans, 5-8 May 1991, 80-89. https://doi.org/10.1145/103418.103434

[11] Frey, G., Muller, M. and Ruck, H. (1999) The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Transactions on Information Theory*, **45**, 1717-1719. https://doi.org/10.1109/18.771254

[12] Kim, T., Kim, S. and Cheon, J.H. (2013) On the Final Exponentiation in Tate Pairing Computations. *IEEE Transactions on Information Theory*, **59**, 4033-4041. https://doi.org/10.1109/tit.2013.2240763

[13] Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J. and Kachisa, E.J. (2009) On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H. and Waters, B., Eds., *Pairing-Based Cryptography—Pairing* 2009, Springer, 78-88. https://doi.org/10.1007/978-3-642-03298-1_6

[14] Cheng, Z.H. and Nistazakis, M. (2005) Implementing Pairing-Based Cryptosystems. *Proceedings of* 3*rd International Workshop in Wireless Security Technologies* (*IWWST* 2005), London, 4-5 April 2005.

[15] Barreto, P.S.L.M., Kim, H.Y., Lynn, B. and Scott, M. (2002) Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M., Ed., *Advances in Cryptology—CRYPTO* 2002, Springer, 354-369. https://doi.org/10.1007/3-540-45708-9_23

[16] Ezzouak, S., El Amrani, M. and Azizi, A. (2014) Optimizing the Computing of Pairing with Miller's Algorithm. *International Journal of Security and Its Applications*, **8**, 171-182. https://doi.org/10.14257/ijsia.2014.8.4.16

[17] Stange, K.E. (2007) The Tate Pairing via Elliptic Nets. In: Takagi, T., *et al*., Eds., *Pairing-Based Cryptography—Pairing* 2007, Springer, 329-348. https://doi.org/10.1007/978-3-540-73489-5_19

[18] Miller, V.S. (2004) The Weil Pairing, and Its Efficient Calculation. *Journal of Cryptology*, **17**, 235-261. https://doi.org/10.1007/s00145-004-0315-8

[19] Zhao, C., Zhang, F. and Huang, J. (2008) Efficient Tate Pairing Computation Using Double-Base Chains. *Science in China Series F*: *Information Sciences*, **51**, 1096-1105. https://doi.org/10.1007/s11432-008-0070-9

[20] 密码行业标准化技术委员会. SM9 标识密码算法 第 1 部分: 总则: GM/T 0044.1-2016 [S]. 北京: 中国标准出版社, 2016.

[21] 甘植旺, 廖方圆. 国密 SM9 中 R-ate 双线性对快速计算[J]. 计算机工程, 2019, 45(6): 171-174.

[22] 付柱. R-ate 双线性对密码算法的高效实现[D]: [硕士学位论文]. 天津: 天津大学, 2018.

[23] 胡芯忆, 何德彪, 彭聪, 等. 一种 SM9 算法 R-ate 对的快速实现方法[J]. 密码学报, 2022, 9(5): 936-948.

[24] 李江峰. SM9 算法的研究与 FPGA 实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2021.

[25] Dong, X., Gao, M., Ma, X., Xiao, C. and Zhang, L. (2024) An Implementation of R-ate Pairing Based on FPGA. 2024 3*rd International Conference on Big Data*, *Information and Computer Network* (*BDICN*), Sanya, 12-14 January 2024, 167-173. https://doi.org/10.1109/bdicn62775.2024.00041