

## 基于基域表示的SM9高次扩域运算优化

谢振杰<sup>1,2</sup>, 刘胜利<sup>1</sup>, 赵方方<sup>1</sup>, 王智超<sup>1</sup>

(1. 信息工程大学网络空间安全教育部重点实验室, 河南 郑州 450001;

2. 中国人民解放军78156部队, 重庆 400039)

**摘要:** 国密算法SM9是我国自主研发的基于标识的密码体制, 其性能瓶颈主要存在于底层的高次扩域运算。传统的塔式扩张法在实现高次扩域上的乘法、平方、求逆和Frobenius映射等运算时, 存在多层逐级计算带来的冗余开销, 限制了算法整体效率的提升。针对这一问题, 提出一种基于基域表示的高次扩域运算优化方法。通过理论推导, 建立了高次扩域上各项运算的基域表达式, 将高次扩域运算转化为基域上的大整数运算, 避免了塔式扩张过程的大量中间层级开销。理论分析表明, 该方法通过分配律优化、合并同类项, 使模运算量减少2/3以上, 其他运算同步简化。测试结果显示, 采用基域表示法的各项高次扩域运算速率达到塔式扩张法的2~3倍, 内存开销峰值降低13.30%~63.60%。进一步地, 在SM9数字签名、密钥交换、加密等7项算法中, 性能提升幅度达95.39%~158.36%。基域表示法在保持算法数学等价性与安全性的前提下, 能够显著提升SM9及其衍生密码方案的执行效率与资源利用率, 且该方法具有通用性, 可推广至SM9以外的高次扩域运算场景。

**关键词:** 高次扩域; 基域表示; SM9算法; 塔式扩张; 双线性对

**中图分类号:** TP309.7

**文献标志码:** A

**DOI:** 10.11959/j.issn.2096-109x.AQ25187

## Optimization of high-degree extension field operations in SM9 based on base field representation

Xie Zhenjie<sup>1,2</sup>, Liu Shengli<sup>1</sup>, Zhao Fangfang<sup>1</sup>, Wang Zhichao<sup>1</sup>

1. Key Laboratory of Cyberspace Security, Ministry of Education, Information Engineering University, Zhengzhou 450001, China

2. Troop 78156 of People's Liberation Army, Chongqing 400039, China

**Abstract:** The domestic cryptographic algorithm SM9, which is an independently innovated identity-based cryptographic system in China, was limited by performance bottlenecks in its underlying high-degree extension field operations. In the traditional tower extension method, operations such as multiplication, squaring, inversion, and Frobenius mapping on high-degree extension fields involved redundant computational overhead due to multi-level hierarchical processing, which constrained the overall efficiency improvement of the algorithm. To address this issue, a base field representation method was proposed for optimizing high-degree extension field operations. Through theoretical derivation, base field expressions for various operations on high-degree extension fields were established, transforming these operations directly into large integer arithmetic on the base field and thereby avoiding the substantial intermediate layer overhead inherent in the tower extension process. Theoretical analysis demonstrated that, through distributive law optimization and combination of like terms, modular operations were reduced by over two-thirds, with other operations simplified accordingly. Experimental results show that the proposed base field representation achieves a

收稿日期: 2025-07-19; 修回日期: 2026-02-21

通信作者: 赵方方, fangfangzhaowlc@163.com

基金项目: 装备预先研究项目 (No.30603010601)

**Foundation Item:** The Equipment Pre Research Project (No.30603010601)

speedup of 2~3 times for various high-degree extension field operations compared to the tower extension method, with peak memory consumption reduced by 13.30%~63.60%. Furthermore, performance improvements of 95.39%~158.36% were observed in seven SM9 algorithms including digital signature, key exchange, and encryption. The base field representation method significantly enhances the execution efficiency and resource utilization of SM9 and its derived cryptographic schemes while preserving mathematical equivalence and security. Moreover, this method exhibits generality and can be extended to other high-degree extension field scenarios beyond SM9.

**Key words:** high-degree extension field, base field representation, SM9 algorithm, tower extension, bilinear pairing

## 0 引言

国密算法 SM9 是我国自主研发的基于标识的密码 (identity-based cryptography, IBC) 体制, 其涵盖数字签名算法、密钥交换协议、密钥封装机制和加密算法<sup>[1-2]</sup>。在 IBC 中, 用户的身份信息 (如证件号、手机号、电子邮件地址等) 可以直接作为公钥使用, 无须依赖第三方颁发的证书来验证公钥的真实性。这种设计避免了公钥基础设施 (public key infrastructure, PKI) 的构建, 显著减少了密码系统的部署和运维成本。此外, SM9 算法基于椭圆曲线密码学, 在提供与 RSA 等传统算法同等安全强度的同时, 具备更短的密钥长度与更高的计算效率, 从而增强了系统的整体安全性。近年来, 基于国密算法 SM9 设计的环签名<sup>[3]</sup>、可搜索加密<sup>[4]</sup>、分层标识加密<sup>[5-6]</sup>、广播加密<sup>[7-8]</sup>、容错加密<sup>[9]</sup>和聚合签名<sup>[10]</sup>等标识密码应用方案被陆续提出, 体现出 SM9 算法优异的性能和广泛的应用适配能力。然而, SM9 及其衍生方案的执行效率仍受限于部分高耗时运算 (如椭圆曲线标量乘法、双线性对计算和在群  $\mathcal{G}_T$  上的幂运算等), 对这些底层算法进行优化将有助于释放 SM9 的应用潜力。

本文将高次扩域  $\mathcal{F}_{q^{12}}$  (包括群  $\mathcal{G}_T$ ) 上的乘法、平方、求逆和 Frobenius 映射等 SM9 基础运算 (统称为“高次扩域运算”) 直接转化为基域  $\mathcal{F}_q$  上的运算, 提出高次扩域运算的基域表示法。与传统的塔式扩张法相比, 所提方法可以大幅减少总体计算量, 显著提升 SM9 算法的性能。

## 1 相关工作

近年来, SM9 算法优化研究沿两条主线并行发展, 即算法层面的改进与实现层面的创新。在算法层面, 研究者聚焦核心运算组件的效率提升。

胡芯忆等<sup>[11]</sup>通过优化有限域 12 次扩域  $\mathcal{F}_{q^{12}}$  上的平方运算, 提出面向椭圆曲线标量乘的稀疏乘法, 并对双线性对中的 Miller 算法与幂运算进行了优化, 达到了较好的 SM9 双线性对软件实现效果。Zhen 等<sup>[12]</sup>分析了 SM9 双线性对的计算过程和优化方法, 将 Miller 循环和最终模幂的效率分别提高了约 5.2% 和 0.91%。Cheng 等<sup>[13]</sup>基于 SM9 算法参数对 R-ate 双线性对进行优化, 并通过 Frobenius 映射降低了硬件实现成本。Karabina<sup>[14]</sup>提出了群  $\mathcal{G}_T$  上的压缩平方算法, 提高了最终模幂困难部分中平方运算的效率。王江涛等<sup>[15]</sup>阐述了基于 Comb 固定基的高次幂算法、分圆子群的快速平方算法及非相邻形式 (NAF) 编码算法。谢振杰等<sup>[16]</sup>通过预计算提高了椭圆曲线固定点标量乘的性能, 并采用预计算的 Miller 算法、最终模幂困难部分的构造优化和群  $\mathcal{G}_T$  上的幂运算优化, 提高了双线性对的性能, 同时借助 Comb 固定基技术提高了群  $\mathcal{G}_T$  上幂的计算效率。谢振杰等<sup>[17]</sup>使用二维 Comb 固定基改进  $\mathcal{G}_T$  上幂运算的效率, 并将预计算标量乘推广到 SM9 的 3 个常见步骤, 还针对常用标识 (ID) 设计了专项优化方案。

在实现层面, 研究者从软、硬件方向不断挖掘算法性能的潜力。Wu 等<sup>[18]</sup>在 Arria10 软件平台上实现双线性对, 并将 Karatsuba 算法应用于扩域乘法, 降低了计算复杂度。Xie 等<sup>[19]</sup>设计了 BN (Barreto-Naehrig) 曲线上 O-ate 对的高性能硬件实现, 分析了其可并行计算部分。此外, 开源社区贡献了具有工程实用价值的实现代码 (如国内研究团队开源了以 C、JavaScript 和 Python 等语言实现的完整国密算法代码<sup>[20-21]</sup>), 这些实现代码不仅包含比文献描述更详尽的技术细节, 也融入了诸多实用的优化技巧。

综上所述, 对于大整数运算、群  $\mathcal{G}_T$  上的幂和双线性对等 SM9 底层数学组件的优化实现, 现

有文献和代码已有不少成果。然而，SM9算法涉及大量高次扩域运算，其数学表示和代码实现始终受限于塔式扩张的固有框架。虽然塔式扩张法形式简洁，但多层逐级计算会引入额外的计算和存储开销。目前，未见公开文献或代码采用塔式扩张法以外的方法实现高次扩域运算。若能在运用现有优化技术的同时，彻底消除塔式扩张的中间层级，实现高次扩域的“扁平化”计算，有望进一步释放SM9算法的性能潜力。

## 2 高次扩域运算的塔式扩张

本节简要介绍扩域 $\mathcal{F}_{q^{12}}$ 和群 $\mathcal{G}_T$ 的基本含义及其元素与运算的塔式扩张表示法。

### 2.1 扩域 $\mathcal{F}_{q^{12}}$ 和群 $\mathcal{G}_T$ 的含义与元素表示

设 $\mathcal{F}_q$ 是包含 $q$ 个元素的有限域， $\mathcal{F}_{q^d}$ 是有限域 $\mathcal{F}_q$ 的 $d$ 次扩域， $\mathcal{F}_q$ 是扩域 $\mathcal{F}_{q^d}$ 的基域；设 $\mathcal{G}$ 为椭圆曲线 $E(\mathcal{F}_q)$ 的 $N$ 阶子群，使 $N|q^k-1$ 成立的最小正整数 $k$ 称为子群 $\mathcal{G}$ 相对于 $N$ 的嵌入次数。 $\mathcal{G}_1$ 为 $E(\mathcal{F}_{q^{d_1}})$ 的 $N$ 阶子群（其中 $d_1|k$ ）， $\mathcal{G}_2$ 为 $E(\mathcal{F}_{q^{d_2}})$ 的 $N$ 阶子群（其中 $d_2|k$ ）。在SM9算法的系统参数中<sup>[1]</sup>，嵌入次数 $k=12$ ， $k$ 的因子 $d_1=1$ 、 $d_2=2$ 。椭圆曲线双线性对 $e$ 表示从 $\mathcal{G}_1, \mathcal{G}_2$ 到分圆子群 $\mathcal{G}_{\phi_6}(\mathcal{F}_{q^2})$ 的映射， $\mathcal{G}_{\phi_6}(\mathcal{F}_{q^2})$ 是 $\mathcal{F}_{q^{12}}^*$ 的子群（ $\mathcal{F}_{q^{12}}^*$ 是由扩域 $\mathcal{F}_{q^{12}}$ 中所有非零元组成的乘法群），简记为 $\mathcal{G}_T$ ，即 $\mathcal{G}_T$ 是 $e$ 的值域。为方便对照，表1列出了本文使用的主要符号及其含义，其余符号将在首次出现时予以说明。

表1 主要符号及其含义

| 符号                     | 含义                               |
|------------------------|----------------------------------|
| $\mathcal{F}_q$        | 包含 $q$ 个元素的有限域                   |
| $\mathcal{F}_{q^2}$    | 有限域 $\mathcal{F}_q$ 的2次扩域        |
| $\mathcal{F}_{q^4}$    | 有限域 $\mathcal{F}_q$ 的4次扩域        |
| $\mathcal{F}_{q^{12}}$ | 有限域 $\mathcal{F}_q$ 的12次扩域       |
| $\mathcal{G}_T$        | 乘法循环群（双线性对的值域）                   |
| $q$                    | 大素数，表示有限域 $\mathcal{F}_q$ 中元素的个数 |
| $\% q$                 | 模 $q$ 运算                         |

扩域 $\mathcal{F}_{q^{12}}$ 上的元素 $a$ 通常以1-2-4-12的塔式扩张方式表示，即：

$$\begin{aligned} a &= (a_2, a_1, a_0) \\ &= ((a_{2,1}, a_{2,0}), (a_{1,1}, a_{1,0}), (a_{0,1}, a_{0,0})) \\ &= (((a_{2,1,1}, a_{2,1,0}), (a_{2,0,1}, a_{2,0,0})), ((a_{1,1,1}, a_{1,1,0}), (a_{1,0,1}, a_{1,0,0})), \\ &\quad ((a_{0,1,1}, a_{0,1,0}), (a_{0,0,1}, a_{0,0,0}))) \end{aligned} \quad (1)$$

其中， $a_2, a_1, a_0 \in \mathcal{F}_{q^4}$ ， $a_{2,1}, a_{2,0}, a_{1,1}, a_{1,0}, a_{0,1}, a_{0,0} \in \mathcal{F}_{q^2}$ ， $a_{2,1,1}, a_{2,1,0}, a_{2,0,1}, a_{2,0,0}, a_{1,1,1}, a_{1,1,0}, a_{1,0,1}, a_{1,0,0}, a_{0,1,1}, a_{0,1,0}, a_{0,0,1}, a_{0,0,0} \in \mathcal{F}_q$ 。在本文的扩域表达式中，高维元素在左，低维元素在右。

### 2.2 扩域 $\mathcal{F}_{q^{12}}$ 和群 $\mathcal{G}_T$ 上的运算

令 $a, b \in \mathcal{F}_{q^{12}}$ ， $\mathcal{F}_{q^{12}}$ 上的乘法和平方运算可由 $\mathcal{F}_{q^4}$ 上的运算表示，分别如式(2)和式(3)所示。

$$\begin{aligned} a \times b &= (a_0 \times b_2 + a_1 \times b_1 + a_2 \times b_0, a_0 \times b_1 + a_1 \times b_0 + a_2 \times b_2 \times v, \\ &\quad a_0 \times b_0 + a_1 \times b_2 \times v + a_2 \times b_1 \times v) \quad (2) \\ a^2 &= (a_0 \times a_2 \times 2 + a_1^2, a_0 \times a_1 \times 2 + a_2^2 \times v, a_0^2 + a_1 \times a_2 \times v \times 2) \quad (3) \end{aligned}$$

其中， $v = \sqrt{\sqrt{-2}} \in \mathcal{F}_{q^4}$ 。因 $\mathcal{G}_T \subset \mathcal{F}_{q^{12}}$ ，故 $\mathcal{F}_{q^{12}}$ 元素的表示和计算方法同样适用于 $\mathcal{G}_T$ 元素。而 $\mathcal{G}_T$ 上的平方运算较 $\mathcal{F}_{q^{12}}$ 更加简化<sup>[14-15]</sup>。若 $a \in \mathcal{G}_T$ ，则：

$$\begin{aligned} a^2 &= (a_1^2 + (a_1^2 - \text{conj}(a_2)) \times 2, a_2^2 \times v + \\ &\quad (a_2^2 \times v - \text{conj}(a_1)) \times 2, a_0^2 + (a_0^2 - \text{conj}(a_0)) \times 2) \quad (4) \end{aligned}$$

其中， $\text{conj}(a)$ 表示 $a$ 的共轭。对于 $a = (a_1, a_0)$ ，且 $a \in \mathcal{F}_{q^4}$ 或 $a \in \mathcal{F}_{q^2}$ ，则有 $\text{conj}(a) = (-a_1, a_0)$ 。

### 2.3 扩域 $\mathcal{F}_{q^4}$ 上的运算

令 $a, b \in \mathcal{F}_{q^4}$ ， $\mathcal{F}_{q^4}$ 上的运算可由 $\mathcal{F}_{q^2}$ 上的运算表示，分别如式(5)~式(8)所示。

$$a \times b = (a_0 \times b_1 + a_1 \times b_0 \times u, a_0 \times b_0 + a_1 \times b_1 \times u) \quad (5)$$

$$a \times b \times v = (a_0 \times b_0 + a_1 \times b_1 \times u, a_0 \times b_1 \times u + a_1 \times b_0 \times u) \quad (6)$$

$$a^2 = (a_0 \times a_1 \times 2, a_0^2 + a_1^2 \times u) \quad (7)$$

$$a^2 \times v = (a_0^2 + a_1^2 \times u, a_0 \times a_1 \times u \times 2) \quad (8)$$

其中， $u = \sqrt{-2} \in \mathcal{F}_{q^2}$ 。

### 2.4 扩域 $\mathcal{F}_{q^2}$ 上的运算

令 $a, b \in \mathcal{F}_{q^2}$ ， $\mathcal{F}_{q^2}$ 上的运算可由 $\mathcal{F}_q$ 上的运算表示，分别如式(9)~式(12)所示。

$$a \times b = ((a_0 \times b_1 + a_1 \times b_0) \% q, (a_0 \times b_0 - a_1 \times b_1 \times 2) \% q) \quad (9)$$

$$\begin{aligned} a \times b \times u &= ((a_0 \times b_0 - a_1 \times b_1 \times 2) \% q, \\ &\quad -2 \times (a_0 \times b_1 + a_1 \times b_0) \% q) \quad (10) \end{aligned}$$

$$a^2 = (a_0 \times a_1 \times 2 \% q, (a_0^2 - a_1^2 \times 2) \% q) \quad (11)$$

$$a^2 \times u = ((a_0^2 - a_1^2 \times 2) \% q, -4 \times a_0 \times a_1 \% q) \quad (12)$$

通过上述计算式, 高次扩域运算经过逐级塔式扩张 ( $\mathcal{F}_{q^{12}} \rightarrow \mathcal{F}_{q^4} \rightarrow \mathcal{F}_{q^2} \rightarrow \mathcal{F}_q$ ), 最终转化为基域上的加、减、乘和模运算, 而基域上的运算实际上是大整数运算。

### 3 高次扩域运算的基域表示

为简化高次扩域运算, 推导直接用基域元素进行计算的表示方法。

#### 3.1 $\mathcal{F}_{q^{12}}$ 上的乘法和平方运算

对于  $a \in \mathcal{F}_{q^{12}}$ ,  $a$  由  $\mathcal{F}_q$  上的元素表示为:

$$a = (a_{11}, a_{10}, a_9, a_8, a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0) \quad (13)$$

其中,  $a_{11}, a_{10}, \dots, a_0 \in \mathcal{F}_q$ 。

若  $a, b, c, d \in \mathcal{F}_{q^{12}}$  且  $c = a \times b, d = a^2$ , 则  $\mathcal{F}_{q^{12}}$  上的乘法运算由  $\mathcal{F}_q$  上的运算表示如下:

$$c_0 = (a_0 \times b_0 - 2 \times (a_1 \times b_1 + a_2 \times b_3 + a_3 \times b_2 + a_4 \times b_{11} + a_5 \times b_{10} + a_6 \times b_9 + a_7 \times b_8 + a_8 \times b_7 + a_9 \times b_6 + a_{10} \times b_5 + a_{11} \times b_4)) \% q$$

$$c_1 = (a_0 \times b_1 + a_1 \times b_0 + a_2 \times b_2 + a_4 \times b_{10} + a_6 \times b_8 + a_8 \times b_6 + a_{10} \times b_4 - 2 \times (a_3 \times b_3 + a_5 \times b_{11} + a_7 \times b_9 + a_9 \times b_7 + a_{11} \times b_5)) \% q$$

$$c_2 = (a_0 \times b_2 + a_2 \times b_0 + a_4 \times b_8 + a_8 \times b_4 - 2 \times (a_1 \times b_3 + a_3 \times b_1 + a_5 \times b_9 + a_6 \times b_{11} + a_7 \times b_{10} + a_9 \times b_5 + a_{10} \times b_7 + a_{11} \times b_6)) \% q$$

$$c_3 = (a_0 \times b_3 + a_1 \times b_2 + a_2 \times b_1 + a_3 \times b_0 + a_4 \times b_9 + a_5 \times b_8 + a_6 \times b_{10} + a_8 \times b_5 + a_9 \times b_4 + a_{10} \times b_6 - 2 \times (a_7 \times b_{11} + a_{11} \times b_7)) \% q$$

$$c_4 = (a_0 \times b_4 + a_4 \times b_0 - 2 \times (a_1 \times b_5 + a_2 \times b_7 + a_3 \times b_6 + a_5 \times b_1 + a_6 \times b_3 + a_7 \times b_2 + a_8 \times b_{11} + a_9 \times b_{10} + a_{10} \times b_9 + a_{11} \times b_8)) \% q$$

$$c_5 = (a_0 \times b_5 + a_1 \times b_4 + a_2 \times b_6 + a_4 \times b_1 + a_5 \times b_0 + a_6 \times b_2 + a_8 \times b_{10} + a_{10} \times b_8 - 2 \times (a_3 \times b_7 + a_7 \times b_3 + a_9 \times b_{11} + a_{11} \times b_9)) \% q$$

$$c_6 = (a_0 \times b_6 + a_2 \times b_4 + a_4 \times b_2 + a_6 \times b_0 + a_8 \times b_8 - 2 \times (a_1 \times b_7 + a_3 \times b_5 + a_5 \times b_3 + a_7 \times b_1 + a_9 \times b_9 + a_{10} \times b_{11} + a_{11} \times b_{10})) \% q$$

$$c_7 = (a_0 \times b_7 + a_1 \times b_6 + a_2 \times b_5 + a_3 \times b_4 + a_4 \times b_3 + a_5 \times b_2 + a_6 \times b_1 + a_7 \times b_0 + a_8 \times b_9 + a_9 \times b_8 + a_{10} \times b_{10} - 2 \times (a_{11} \times b_{11})) \% q$$

$$c_8 = (a_0 \times b_8 + a_4 \times b_4 + a_8 \times b_0 - 2 \times (a_1 \times b_9 +$$

$$a_2 \times b_{11} + a_3 \times b_{10} + a_5 \times b_5 + a_6 \times b_7 + a_7 \times b_6 + a_9 \times b_1 + a_{10} \times b_3 + a_{11} \times b_2)) \% q$$

$$c_9 = (a_0 \times b_9 + a_1 \times b_8 + a_2 \times b_{10} + a_4 \times b_5 + a_5 \times b_4 + a_6 \times b_6 + a_8 \times b_1 + a_9 \times b_0 + a_{10} \times b_2 - 2 \times (a_3 \times b_{11} + a_7 \times b_7 + a_{11} \times b_3)) \% q$$

$$c_{10} = (a_0 \times b_{10} + a_2 \times b_8 + a_4 \times b_6 + a_6 \times b_4 + a_8 \times b_2 + a_{10} \times b_0 - 2 \times (a_1 \times b_{11} + a_3 \times b_9 + a_5 \times b_7 + a_7 \times b_5 + a_9 \times b_3 + a_{11} \times b_1)) \% q$$

$$c_{11} = (a_0 \times b_{11} + a_1 \times b_{10} + a_2 \times b_9 + a_3 \times b_8 + a_4 \times b_7 + a_5 \times b_6 + a_6 \times b_5 + a_7 \times b_4 + a_8 \times b_3 + a_9 \times b_2 + a_{10} \times b_1 + a_{11} \times b_0) \% q$$

$$c = (c_{11}, c_{10}, c_9, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$$

$\mathcal{F}_{q^{12}}$  上的平方运算表示如下:

$$d_0 = (a_0 \times a_0 - 2 \times a_1 \times a_1 - 4 \times (a_2 \times a_3 + a_4 \times a_{11} + a_5 \times a_{10} + a_6 \times a_9 + a_7 \times a_8)) \% q$$

$$d_1 = (a_2 \times a_2 + 2 \times (a_0 \times a_1 - a_3 \times a_3 + a_4 \times a_{10} + a_6 \times a_8) - 4 \times (a_5 \times a_{11} + a_7 \times a_9)) \% q$$

$$d_2 = (2 \times (a_0 \times a_2 + a_4 \times a_8) - 4 \times (a_1 \times a_3 + a_5 \times a_9 + a_6 \times a_{11} + a_7 \times a_{10})) \% q$$

$$d_3 = (2 \times (a_0 \times a_3 + a_1 \times a_2 + a_4 \times a_9 + a_5 \times a_8 + a_6 \times a_{10}) - 4 \times a_7 \times a_{11}) \% q$$

$$d_4 = (2 \times a_0 \times a_4 - 4 \times (a_1 \times a_5 + a_2 \times a_7 + a_3 \times a_6 + a_8 \times a_{11} + a_9 \times a_{10})) \% q$$

$$d_5 = (2 \times (a_0 \times a_5 + a_1 \times a_4 + a_2 \times a_6 + a_8 \times a_{10}) - 4 \times (a_3 \times a_7 + a_9 \times a_{11})) \% q$$

$$d_6 = (a_8 \times a_8 + 2 \times (a_0 \times a_6 + a_2 \times a_4 - a_9 \times a_9) - 4 \times (a_1 \times a_7 + a_3 \times a_5 + a_{10} \times a_{11})) \% q$$

$$d_7 = (a_{10} \times a_{10} + 2 \times (a_0 \times a_7 + a_1 \times a_6 + a_2 \times a_5 + a_3 \times a_4 + a_8 \times a_9 - a_{11} \times a_{11})) \% q$$

$$d_8 = (a_4 \times a_4 + 2 \times (a_0 \times a_8 - a_5 \times a_5) - 4 \times (a_1 \times a_9 + a_2 \times a_{11} + a_3 \times a_{10} + a_6 \times a_7)) \% q$$

$$d_9 = (a_6 \times a_6 + 2 \times (a_0 \times a_9 + a_1 \times a_8 + a_2 \times a_{10} + a_4 \times a_5 - a_7 \times a_7) - 4 \times a_3 \times a_{11}) \% q$$

$$d_{10} = (2 \times (a_0 \times a_{10} + a_2 \times a_8 + a_4 \times a_6) - 4 \times (a_1 \times a_{11} + a_3 \times a_9 + a_5 \times a_7)) \% q$$

$$d_{11} = 2 \times (a_0 \times a_{11} + a_1 \times a_{10} + a_2 \times a_9 + a_3 \times a_8 + a_4 \times a_7 + a_5 \times a_6) \% q$$

$$d = (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, d_4, d_3, d_2, d_1, d_0)$$

#### 3.2 $\mathcal{G}_T$ 上的平方运算

若  $a, d \in \mathcal{G}_T$  且  $d = a^2$ , 则  $\mathcal{G}_T$  上的平方运算表示如下:

$$\begin{aligned}
d_0 &= ((3 \times a_0 - 2) \times a_0 - 6 \times a_1 \times a_1 - 12 \times a_2 \times a_3) \% q \\
d_1 &= (3 \times a_2 \times a_2 + 6 \times (a_0 \times a_1 - a_3 \times a_3) - 2 \times a_1) \% q \\
d_2 &= ((6 \times a_0 + 2) \times a_2 - 12 \times a_1 \times a_3) \% q \\
d_3 &= (6 \times (a_0 \times a_3 + a_1 \times a_2) + 2 \times a_3) \% q \\
d_4 &= (2 \times a_4 - 12 \times (a_8 \times a_{11} + a_9 \times a_{10})) \% q \\
d_5 &= (6 \times a_8 \times a_{10} - 12 \times a_9 \times a_{11} + 2 \times a_5) \% q \\
d_6 &= (3 \times a_8 \times a_8 - 6 \times a_9 \times a_9 - 12 \times a_{10} \times a_{11} - 2 \times a_6) \% q \\
d_7 &= (3 \times a_{10} \times a_{10} + 6 \times (a_8 \times a_9 - a_{11} \times a_{11}) - 2 \times a_7) \% q \\
d_8 &= (3 \times a_4 \times a_4 - 6 \times a_5 \times a_5 - 12 \times a_6 \times a_7 - 2 \times a_8) \% q \\
d_9 &= (3 \times a_6 \times a_6 + 6 \times (a_4 \times a_5 - a_7 \times a_7) - 2 \times a_9) \% q \\
d_{10} &= (6 \times a_4 \times a_6 - 12 \times a_5 \times a_7 + 2 \times a_{10}) \% q \\
d_{11} &= (6 \times (a_4 \times a_7 + a_5 \times a_6) + 2 \times a_{11}) \% q \\
\mathbf{d} &= (d_{11}, d_{10}, d_9, d_8, d_7, d_6, d_5, d_4, d_3, d_2, d_1, d_0)
\end{aligned}$$

以上表达式展示了部分高次扩域运算的基域表示法，即由输入元素的12个基域分量直接计算目标元素的12个基域分量。此方法通过分配律合并同类计算，不再途径 $\mathcal{F}_{q^4}$ 和 $\mathcal{F}_{q^2}$ ，为塔式扩张的数学等价变形，其正确性由代数一致性保证，且二者的计算结果完全一致。 $\mathcal{F}_{q^{12}}$ 上的求逆和Frobenius映射也可以采用基域表示法，限于篇幅，本文不再赘述。

## 4 性能分析与实验

本节对塔式扩张法和基域表示法两种计算方法进行理论分析与实验测试，并考察其对SM9算法性能的影响。

### 4.1 算法复杂度分析

分析 $\mathcal{F}_{q^{12}}$ 上的乘法和平方、 $\mathcal{G}_T$ 上的平方3项运算在两种计算方法下各项基域运算的次数，结果见表2。其中，R、M、A、S分别表示 $\mathcal{F}_q$ 上的模运算、乘法、加/减法和移位，乘数之一为2或4的乘法用移位代替，表2最后一列为基域表示法相比塔式扩张法节省的计算开销。

表2数据表明，基域表示法大幅减少了 $\mathcal{F}_q$ 上的计算开销，特别是相对耗时的模运算减少了2/3以上。模运算次数的减少，主要是因为塔式扩

张中，计算目标 $\mathcal{F}_{q^{12}}$ 元素的每一个分量都需要逐级调用若干次 $\mathcal{F}_{q^2}$ 运算，即需要进行若干次 $\mathcal{F}_q$ 上的模运算；而在基域表示法中，计算目标 $\mathcal{F}_{q^{12}}$ 元素的每一个分量只需进行一次 $\mathcal{F}_q$ 上的模运算。

表2 各项运算在两种计算方法下的计算复杂度

| 运算                        | 塔式扩张法             | 基域表示法             | 差额              |
|---------------------------|-------------------|-------------------|-----------------|
| $\mathcal{F}_{q^{12}}$ 乘法 | 72R+144M+132A+51S | 12R+144M+132A+11S | 60R+40S         |
| $\mathcal{F}_{q^{12}}$ 平方 | 60R+114M+84A+42S  | 12R+78M+66A+22S   | 48R+36M+18A+20S |
| $\mathcal{G}_T$ 平方        | 36R+66M+60A+25S   | 12R+54M+30A+10S   | 24R+12M+30A+15S |

### 4.2 实验测试

本节基于国密算法开源Python库hggm<sup>[21]</sup>，采用SM9国家标准规定的参数设置<sup>[1]</sup>，通过Python编程实现了两种计算方法下的各项运算。测试中使用tracemalloc工具<sup>[22]</sup>对内存开销进行监测。各项测试均执行10 000次，取其平均值作为最终的有效数据。实验所用计算机的配置见表3。

表3 计算机配置

| 项目       | 配置                            |
|----------|-------------------------------|
| 设备类型     | PC                            |
| 操作系统     | Windows10 64位                 |
| CPU      | Intel Core i3-10110U (2核心4线程) |
| 内存       | 8GB LPDDR3 2133 MHz           |
| 硬盘       | SAMSUNG MZVLB512HBJQ-000L7    |
| Python版本 | 3.7.1                         |

表4展示了各项运算在两种计算方法下的测试结果。其中，第4列为基域表示法相对于塔式扩张法的运算速率加速比（即塔式扩张法耗时与基域表示法耗时之比），第7列为基域表示法内存开销峰值相对于塔式扩张法内存开销峰值的降幅。除乘法和平方运算外，表4还包含了 $\mathcal{F}_{q^{12}}$ 上的求逆运算和4种Frobenius映射运算，以及 $\mathcal{G}_T$ 上的幂运算与双线性对运算的测试结果。 $\mathcal{G}_T$ 上的幂运算由大量 $\mathcal{F}_{q^{12}}$ 上的乘法和 $\mathcal{G}_T$ 上的平方运算组合而成，而双线性对运算则涉及表4中除其本身以外的其余9种运算，这两类运算均为SM9算法的关键步骤<sup>[2]</sup>，且在实际计算中耗时显著<sup>[16]</sup>。

表 4 各项运算在两种计算方法下的测试结果

| 运算                        | 塔式扩张法耗时              | 基域表示法耗时              | 加速比  | 塔式扩张法<br>内存开销峰值 | 基域表示法<br>内存开销峰值 | 内存开销<br>峰值降幅 |
|---------------------------|----------------------|----------------------|------|-----------------|-----------------|--------------|
| $\mathcal{F}_{q^{12}}$ 乘法 | 110.11 $\mu\text{s}$ | 39.63 $\mu\text{s}$  | 2.78 | 3 424 B         | 1 796 B         | 47.55%       |
| $\mathcal{F}_{q^{12}}$ 平方 | 66.98 $\mu\text{s}$  | 24.90 $\mu\text{s}$  | 2.69 | 2 720 B         | 1 684 B         | 38.09%       |
| $\mathcal{G}_T$ 平方        | 46.05 $\mu\text{s}$  | 18.75 $\mu\text{s}$  | 2.46 | 4 604 B         | 1 676 B         | 63.60%       |
| $\mathcal{F}_{q^{12}}$ 求逆 | 214.86 $\mu\text{s}$ | 109.97 $\mu\text{s}$ | 1.95 | 6 696 B         | 4 832 B         | 27.84%       |
| $q$ 次Frobenius映射          | 16.58 $\mu\text{s}$  | 7.65 $\mu\text{s}$   | 2.17 | 1 744 B         | 1 512 B         | 13.30%       |
| $q^2$ 次Frobenius映射        | 16.79 $\mu\text{s}$  | 6.08 $\mu\text{s}$   | 2.76 | 1 716 B         | 1 428 B         | 16.78%       |
| $q^3$ 次Frobenius映射        | 15.47 $\mu\text{s}$  | 4.73 $\mu\text{s}$   | 3.27 | 1 684 B         | 1 392 B         | 17.34%       |
| $q^6$ 次Frobenius映射        | 7.58 $\mu\text{s}$   | 1.08 $\mu\text{s}$   | 7.03 | 2 264 B         | 1 064 B         | 53.00%       |
| $\mathcal{G}_T$ 上的幂       | 21.08 ms             | 7.84 ms              | 2.69 | 9 390 B         | 3 690 B         | 60.70%       |
| 双线性对                      | 24.82 ms             | 11.53 ms             | 2.15 | 42 280 B        | 24 572 B        | 41.88%       |

从表 4 中的数据可以看出, 对于大部分运算, 基域表示法的计算速率约为塔式扩张法的 2~3 倍, 内存开销峰值降幅达 13.30%~63.60%, 表明基域表示法具有更高的时空效率。这一优势主要得益于两个方面: 一方面是基域表示法减少了  $\mathcal{F}_q$  上的运算次数; 另一方面是避免了塔式扩张过程中频繁创建  $\mathcal{F}_{q^4}$  和  $\mathcal{F}_{q^2}$  对象、函数调用的时空开销。

本文对比了两种计算方法在 7 项 SM9 算法中的实际耗时表现, 结果如图 1 所示。除高次扩域运算分别采用塔式扩张法和基域表示法外, 其余实现细节均相同。测试数据显示, 基域表示法相较于塔式扩张法使 SM9 各项算法获得 95.39%~158.36% 的性能提升。

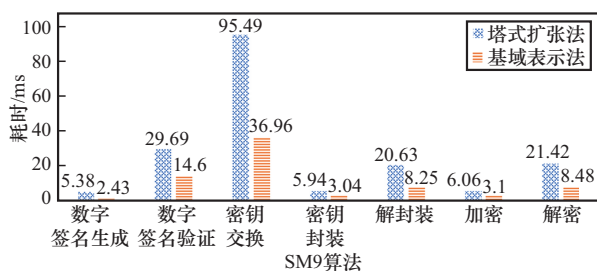


图 1 SM9 算法在两种计算方法下的耗时

## 5 结束语

SM9 基础运算的效率优化, 对提升 SM9 算法及其衍生密码方案的整体性能和实用性具有重要的意义。针对采用塔式扩张法进行高次扩域运算时存在效率偏低的问题, 本文推导了基域元素直接参与各项高次扩域运算的表达式, 从而省略了塔式扩张中的多层中间计算过程。理论分析和

实验测试均表明, 基域表示法能够有效减少计算量和内存开销, 显著提升 SM9 各项算法的运行效率。由于基域表示法仅为数学上的等价变换, 并未改变 SM9 算法的数学基础, 因此不会影响其安全性。本文聚焦的扩域运算优化位于底层大整数运算 (如大整数乘法、模运算) 与上层 SM9 算法步骤 (包括双线性对运算、群  $\mathcal{G}_T$  上的幂运算等) 之间, 与这两层的现有优化方法相互正交, 因此可以无缝集成到现有 SM9 优化体系中, 从而减少 SM9 算法在实际应用中的时延与内存开销。另外, 该方法具有通用性, 不仅适用于 SM9 相关的扩域运算, 也适用于其他高次扩域运算。后续研究将探索基域表示法与多核/图形处理单元 (GPU) 并行计算相结合的方案, 进一步提升 SM9 算法的执行效率。

本文涉及的 Python 实现代码及其他未尽细节 (如  $\mathcal{F}_{q^{12}}$  上的求逆和 Frobenius 映射等) 已发布在“码云”平台, 该方法为国密算法开源社区提供了一个性能更强的 SM9 实现参考。

## 参考文献:

- [1] 全国信息安全标准化技术委员会. 信息安全技术 SM9 标识密码算法 第 1 部分: 总则: GB/T 38635.1—2020[S]. 北京: 中国标准出版社, 2020.  
National Information Security Standardization Technical Committee. Information security technology—Identity-based cryptographic algorithms SM9—Part 1: General: GB/T 38635.1—2020[S]. Beijing: Standards Press of China, 2020.
- [2] 全国信息安全标准化技术委员会. 信息安全技术 SM9 标识密码算法 第 2 部分: 算法: GB/T 38635.2—2020[S]. 北京: 中国标准出版社, 2020.

- National Information Security Standardization Technical Committee. Information security technology—Identity-based cryptographic algorithms SM9—Part 2: Algorithms: GB/T 38635.2—2020[S]. Beijing: Standards Press of China, 2020.
- [3] 谢振杰, 尹小康, 蔡瑞杰, 等. 基于国密算法SM9的可追踪环签名方案[J]. 通信学报, 2025, 46(3): 199-211.  
Xie Z J, Yin X K, Cai R J, et al. Traceable ring signature scheme based on domestic cryptographic algorithm SM9[J]. Journal on Communications, 2025, 46(3): 199-211.
- [4] 蒲浪, 林超, 伍玮, 等. 基于国密SM9的公钥认证可搜索加密方案[J]. 软件学报, 2025, 36(9): 4271-4284.  
Pu L, Lin C, Wu W, et al. Public-key authenticated encryption scheme with keyword search from Chinese cryptographic SM9[J]. Journal of Software, 2025, 36(9): 4271-4284.
- [5] 赖建昌, 黄欣沂, 何德彪, 等. 基于商用密码SM9的高效分层标识加密[J]. 中国科学(信息科学), 2023, 53(5): 918-930.  
Lai J C, Huang X Y, He D B, et al. Anefficient hierarchical identity-based encryption based on SM9[J]. Science in China (Information Sciences), 2023, 53(5): 918-930.
- [6] 李聪, 梁俊凯, 丁煜甲, 等. 基于SM9的分层标识广播内积函数加密[J]. 中国科学(信息科学), 2024, 54(6): 1400-1418.  
Li C, Liang J K, Ding Y J, et al. Hierarchical identity-based broadcast inner product functional encryption based on SM9[J]. Science in China (Information Sciences), 2024, 54(6): 1400-1418.
- [7] 赖建昌, 黄欣沂, 何德彪, 等. 基于SM9的CCA安全广播加密方案[J]. 软件学报, 2023, 34(7): 3354-3364.  
Lai J C, Huang X Y, He D B, et al. CCA secure broadcast encryption based on SM9[J]. Journal of Software, 2023, 34(7): 3354-3364.
- [8] 崔岩, 黄欣沂, 赖建昌, 等. 基于SM9的匿名广播加密方案[J]. 信息安全学报, 2023, 8(6): 15-27.  
Cui Y, Huang X Y, Lai J C, et al. Anonymous broadcast encryption based on SM9[J]. Journal of Cyber Security, 2023, 8(6): 15-27.
- [9] Liu X H, Huang X Y, Cheng Z H, et al. Fault-tolerant identity-based encryption from SM9[J]. Science China Information Sciences, 2024, 67(2): 122101.
- [10] 李继国, 方淳. 基于SM9的指定验证者聚合签名方案[J]. 网络与信息安全学报, 2024, 10(4): 63-71.  
Li J G, Fang C. Designated verifier aggregate signature scheme based on SM9[J]. Chinese Journal of Network and Information Security, 2024, 10(4): 63-71.
- [11] 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.  
Hu X Y, He D B, Peng C, et al. A fast implementation of R-ate pairing in SM9 algorithm[J]. Journal of Cryptologic Research, 2022, 9(5): 936-948.
- [12] Zhen P, Hu X B, Yu Y Y, et al. Research on the optimization computation of SM9 bilinear pairings[C]//Proceedings of the 2nd International Conference on Communication and Information Systems. New York: ACM, 2017: 256-261.
- [13] Cheng X S, Zhang Y Z, Wang Y W. Simplification and hardware parallel design of frobenius mapping algorithm based on SM9[C]//Proceedings of the 3rd IEEE International Conference on Circuits, Systems and Devices (ICCS). Piscataway: IEEE Press, 2019: 78-82.
- [14] Karabina K. Squaring in cyclotomic subgroups[J]. Mathematics of Computation, 2013, 82(281): 555-579.
- [15] 王江涛, 樊荣, 黄哲. SM9中高次幂运算的快速实现方法[J]. 计算机工程, 2023, 49(9): 118-124, 136.  
Wang J T, Fan R, Huang Z. Fast implementation of high power operation in SM9[J]. Computer Engineering, 2023, 49(9): 118-124, 136.
- [16] 谢振杰, 刘奕明, 蔡瑞杰, 等. 国密算法SM9的性能优化方法[J]. 计算机科学, 2025, 52(6): 390-396.  
Xie Z J, Liu Y M, Cai R J, et al. Performance optimization method for domestic cryptographic algorithm SM9[J]. Computer Science, 2025, 52(6): 390-396.
- [17] 谢振杰, 刘奕明, 罗友强, 等. 国密算法SM9的计算性能改进方法[J]. 信息安全研究, 2025, 11(1): 5-11.  
Xie Z J, Liu Y M, Luo Y Q, et al. The methods to improve the computational performance of domestic cryptographic algorithm SM9[J]. Journal of Information Security Research, 2025, 11(1): 5-11.
- [18] Wu Y, Bai G Q, Wu X J. A karatsuba algorithm based accelerator for pairing computation[C]//Proceedings of the 2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC). Piscataway: IEEE Press, 2019: 1-3.
- [19] Xie Y J, Wang B, Zhang L J, et al. A high-performance processor for optimal ate pairing computation over Barreto-Naehrig curves[J]. IET Circuits, Devices & Systems, 2022, 16(5): 427-436.
- [20] Guan Z. GmSSL-Cipher toolbox supporting domestic cryptographic algorithms SM2/SM3/SM4/SM9/SSL[EB]. 2023.
- [21] BASDDSA. Hggm-Python implementation of domestic cryptographic algorithm SM2/SM3/SM4/SM9/ZUC[EB]. 2025.
- [22] Python Software Foundation. Tracemalloc-Trace memory allocations[EB]. 2025.

#### [作者简介]



谢振杰 (1995-), 男, 信息工程大学博士生, 主要研究方向为云安全、密码学应用。



刘胜利 (1973-), 男, 博士, 信息工程大学教授、博士生导师, 主要研究方向为网络设备安全、网络攻击检测。



赵方方 (1990-), 女, 博士, 信息工程大学讲师, 主要研究方向为网络安全、网络流量异常检测。



王智超 (2001-), 男, 信息工程大学博士生, 主要研究方向为网络设备安全、二进制代码分析。