计算机科学
COMPUTER SCIENCE

国密算法SM9的性能优化方法

谢振杰，刘奕明，蔡瑞杰，罗友强

引用本文

谢振杰，刘奕明，蔡瑞杰，罗友强. 国密算法SM9的性能优化方法[J]. 计算机科学，2025，52(6)：390-396.

XIE Zhenjie, LIU Yiming, CAI Ruijie, LUO Youqiang. Performance Optimization Method for Domestic Cryptographic Algorithm SM9 [J]. Computer Science, 2025, 52(6): 390-396.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240300141

国密算法 SM9 的性能优化方法

谢振杰[1,2]　刘奕明[3]　蔡瑞杰[1]　罗友强[1,4]
1 信息工程大学网络空间安全教育部重点实验室　郑州 450001
2 中国人民解放军 78156 部队　重庆 400039
3 中国人民解放军 92330 部队　山东 青岛 266000
4 中国人民解放军 32158 部队　新疆 喀什 844000
(jsonxie@126.com)

摘　要　针对国密算法 SM9 的计算性能优化问题,提出椭圆曲线固定点标量乘预计算、采用预计算的 Miller 算法、最终模幂困难部分构造、分圆子群上的模幂运算、基于 Comb 固定基的模幂运算等性能优化方法,有效提升了 SM9 算法中椭圆曲线标量乘、双线性对、12 次扩域上的模幂等耗时步骤的计算性能。通过 Python 编程实现 SM9 数字签名的生成与验证、密钥交换、密钥封装与解封装、加密与解密 7 项算法。测试表明,综合运用上述优化方法后,各项 SM9 算法的性能提升幅度为 32%～352%。

关键词:国密算法;SM9;性能优化;椭圆曲线;双线性对;Python
中图分类号　TP309

Performance Optimization Method for Domestic Cryptographic Algorithm SM9

XIE Zhenjie[1,2],LIU Yiming[3],CAI Ruijie[1] and LUO Youqiang[1,4]
1 Key Laboratory of Cyberspace Security,Ministry of Education,Information Engineering University,Zhengzhou 450001,China
2 Troop 78156 of PLA,Chongqing 400039,China
3 Troop 92330 of PLA,Qingdao,Shandong 266000,China
4 Troop 32158 of PLA,Kashi,Xinjiang 844000,China

Abstract　To address the challenge of computational performance optimization in the domestic cryptographic algorithm SM9,a suite of performance enhancement techniques has been developed and applied. These methods include fixed-point scalar multiplication precomputation on elliptic curves,an improved Miller algorithm with precomputation,an optimized construction for the hard part of final exponentiation,modular exponentiation within the cyclotomic subgroup,and modular exponentiation employing a Comb-based fixed-base strategy. Through these tailored approaches,significant enhancements have been achieved in the computation of the SM9 algorithm,especially in the time-consuming steps,such as scalar multiplication on elliptic curves,bilinear pairing,and modular exponentiation in the 12th extension field. The seven fundamental SM9 algorithms,encompassing digital signature generation and verification,key exchange,key encapsulation and decapsulation,as well as encryption and decryption,have been effectively implemented in Python. Comprehensive testing reveals that the integration of these optimization techniques yields performance improvements ranging from 32% to 352% for the SM9 algorithms,marking a substantial advance in their computational efficiency.

Keywords　Domestic cryptographic algorithm,SM9,Performance optimization,Elliptic curve,Bilinear pairing,Python

1　引言

国密算法 SM9 是我国自主设计的基于标识的密码体制(Identity-based Cryptography,IBC),包含数字签名算法、密钥交换协议、密钥封装机制和加密算法。2008 年,国家密码管理局正式颁布了商用标识密码算法型号——SM9(商密九号算法);2016 年,国家密码管理局发布密码行业标准“GM/T 0044-2016《SM9 标识密码算法》”;2018 年,SM9 被纳入国际标准;2020 年,由全国信息安全标准化技术委员会提出并归口的国家标准“GB/T 38635《信息安全技术 SM9 标识密码算法》”[1-2]正式发布。基于标识的密码以用户的身份信息作为公钥,简化了复杂的公钥证书管理机制和公钥基础设施,缓解了传统公钥密码体制中高频认证带来的信息堵塞和单点失效问题,对于 5G 时代万物互联场景下的安全

到稿日期:2024-03-20　返修日期:2024-08-06
基金项目:科技委基础加强项目(2019-JCJQ-ZD-113)
This work was supported by the Foundation Strengthening Key Project of Science & Technology Commission(2019-JCJQ-ZD-113).
通信作者:蔡瑞杰(wsxcrj@163.com)

通信有巨大的应用潜力。

然而，SM9 算法涉及椭圆曲线标量乘、双线性对、有限域的 12 次扩域（下文简记为 F_{q12}）模幂等耗时计算，实现性能的问题在一定程度上制约了其落地应用。文献[3]优化了 F_{q12} 上的平方运算，提出了一种面向椭圆曲线标量乘的稀疏乘法，优化了双线性对中的 Miller 算法与幂指数计算，并使用 C 语言编程达到了较好的 R-ate 双线性对软件实现效果。文献[4]指出可通过改变 Miller 算法中同构映射的作用顺序，将 F_{q12} 上的椭圆曲线运算迁移到 2 次扩域（下文简记为 F_{q2}）上。文献[5]对 R-ate 双线性对中模幂运算的指数进行分解，通过分析其数学性质阐述了简单部分和困难部分的计算方法。文献[6]阐述了基于 Comb 固定基的高次幂算法和分圆子群的快速平方算法，以及 NAF 编码算法。文献[7]提出了完成最终模幂困难部分计算的“加法链”方法。文献[8-10]聚焦 SM9 双线性对计算，系统分析了各耗时步骤的计算复杂度，提出了一些优化方法，并通过软件实现或硬件平台进行验证。文献[11-12]对椭圆曲线标量乘以及双线性对的快速实现算法进行了细致的数学分析。文献[13]提出了 4 种针对 BN 曲线上 T-ate 对及其变体的最终模幂困难部分的新计算方法，其相较于此前方法约减少了 37％的内存消耗。文献[14]比较了使用仿射坐标和标准射影坐标实现 O-ate 对的计算效率，并在多个平台实现了 BN 曲线上的 O-ate 对。文献[15]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％。文献[16]基于 SM9 算法参数对 R-ate 双线性对进行优化，并通过 Frobenius 映射来降低硬件实现成本。文献[17]使用 Arria10 软件平台实现双线性对，运用了降低乘法复杂度的 Karatsuba 算法。文献[18]提出的分圆子群上的压缩平方算法，可提高最终模幂困难部分平方运算的效率。文献[19]研究了 BN 曲线上 O-ate 对的硬件实现，对可并行执行的部分进行了分析。文献[20]提出了 SM9 数字签名及验证算法的快速实现方法，其优化方法主要是基于 Comb 固定基的指数运算。文献[21]提出了除 SM9 以外的国密算法的软件实现性能优化方法，其中椭圆曲线标量乘预计算等优化思路也适用于 SM9。此外，国内也有研究团队开源了以 C 语言实现为主的完整的国密算法代码[22]，代码中体现了比大部分文献描述更加详尽的实现细节，也包含了不少实用的优化技巧。近年来，学者陆续提出了基于 SM9 设计的可搜索加密[23]、分层标识加密[24-26]、广播加密[26-27]、环签名[28]和容错加密[29]等标识密码应用方案，但其执行效率都将受到基础 SM9 算法运算效率的制约。因此，研究 SM9 算法的高效实现方法，持续优化其耗时步骤的计算性能，对于 SM9 算法应用推广以及基于 SM9 的各类密码方案设计都具有重要的价值。

本文针对 SM9 算法的计算性能优化问题，提出了椭圆曲线固定点标量乘预计算、固定 G_2 点的 Miller 算法、最终模幂困难部分构造、分圆子群上的模幂运算、基于 Comb 固定基的模幂运算等优化设计，有效提升了 SM9 算法相应步骤的计算性能。其中，固定 G_2 点的 Miller 算法、最终模幂困难部分构造和分圆子群上的模幂运算算法暂未见于其他文献，而本文对椭圆曲线固定点标量乘预计算和基于 Comb 固定基的模幂运算的贡献主要在于提出了高效的预计算数据生成算法。理论分析和实验测试表明，综合运用本文优化方法能明显提升 SM9 算法的整体计算效率。

2 椭圆曲线固定点标量乘预计算优化

预计算技术采用“以空间换时间”的思想，可显著加速椭圆曲线固定点的标量乘运算。文献[21]描述了国密算法 SM2 中基点标量乘的预计算方法，此技术对 SM9 同样适用。令 G_1 和 G_2 为 SM9 算法中的椭圆曲线加法循环群，P_1 和 P_2 分别为 G_1 和 G_2 的生成元，用户签名私钥为 ds，用户加密私钥为 de（本文符号表示同 SM9 国标[1-2]）。在 SM9 算法中，密钥生成中心（Key Generation Center，KGC）计算加密主公钥或 ds，运行密钥交换协议的双方以及密钥封装过程中，均需要计算 1 次与 P_1 的标量乘；KGC 计算签名主公钥或 de，以及数字签名的验证过程中，均需要计算 1 次与 P_2 的标量乘；数字签名的生成需要计算 1 次与 ds 的标量乘[2]。因此，采用预计算技术可提升 SM9 中除解封装算法和解密算法外其余算法的运行效率。

2.1 预计算实现

设参与标量乘运算的乘数 k 为 256b 的整数，椭圆曲线点为 P。预计算技术将 k 视为 32 个字节，对于 k 的每个字节 B_i（$i\in[0,31]$），计算当 B_i 分别为 1，2，…，255 而其余字节全为 0 时的 $k\cdot P$ 结果并保存，由此产生的 $32\times255=8160$ 个椭圆曲线点坐标即为预计算数据。此后，将 k 的每个字节在预计算数据中对应的椭圆曲线点相加，可得标量乘运算 $k\cdot P$ 的结果。

2.2 复杂度分析

对于普通标量乘运算，假设 256b 的乘数 k 在二进制表示下有一半的数位为 1（最高位为 1），则采用二进制展开法计算 $k\cdot P$ 需要 255 次倍点运算和 127 次点加运算；而当 k 按字节划分时，执行 1 次预计算标量乘仅需 31 次点加运算。

不考虑压缩表示时，G_1 上的点坐标大小为 $256\,\text{b}\times2=64\,\text{B}$，针对 1 个 G_1 上的固定点的预计算数据大小为 $8160\times64\,\text{B}=510\,\text{kB}$；而 G_2 上的点坐标大小为 $256\,\text{b}\times2\times2=128\,\text{B}$，针对 1 个 G_2 上的固定点的预计算数据大小为 $8160\times128\,\text{B}=1020\,\text{kB}$。在 SM9 中，$P_1$ 和 ds 在 G_1 上，P_2 在 G_2 上，因此，一个 SM9 实体（假设其可能运行数字签名、密钥交换和加解密）需要加载的固定点标量乘预计算数据大小为 $510\,\text{kB}\times2+1020\,\text{kB}=2040\,\text{kB}$。

上述讨论基于乘数 k 按字节划分，这在大多数通用计算设备中能较好地平衡存储空间和加速效果。但对于计算能力更强或更弱的设备，可选择对乘数 k 更大或更小的划分来进行预计算，以追求最佳的时空效率。

2.3 固定点标量乘预计算数据生成算法

在 SM2 中，需要预计算的椭圆曲线点只有公开且固定的基点 G，其预计算数据可视为全局常量并作为程序的一部分，即 SM2 实体无需执行预计算数据生成算法。例如，文献[21]提供的代码将 SM2 基点 G 的预计算数据作为二进制文件，由程序在启动时加载。类似地，SM9 中 P_1 和 P_2 也是参数确定的椭圆曲线点，其预计算数据可作为全局常量提供给 SM9

实体。然而，同样需要预计算的用户签名私钥 ds 则因用户而异，需要由用户自行产生 ds 的预计算数据。因此，也应关注预计算数据生成算法的效率。文献[21]提供的 SM2 实现代码包含了预计算数据生成算法，但实现方法较为朴素，对用户而言执行耗时过长。

本文重新设计了预计算数据生成算法，其伪代码描述如算法 1 所示。其中，$list.append(x)$ 表示向列表 $list$ 添加元素 x，$double(T)$ 表示 T 点的倍点。

算法 1 固定点标量乘预计算数据生成算法

输入：椭圆曲线点 P
输出：二维列表 kP_list
1. T ← P, kP_list ← []
2. for i from 0 to 31 do
3. t_list ← [T]
4. for j from 0 to 126 do
5. t_list.append(double(t_list[j]))
6. t_list.append(t_list[2 * j+1]+T)
7. end for
8. kP_list.append(t_list)
9. T ← double(t_list[127])
10. end for

在乘数 k 为 256 b 且按字节划分的情形下，算法 1 总计需执行 $32\times128=4096$ 次倍点运算和 $32\times127=4064$ 次点加运算。每个椭圆曲线点都由已有点通过 1 次倍点或点加而产生，最大限度地避免了冗余计算。

3 固定 G_2 点的 Miller 算法优化

SM9 算法所采用的双线性对是 R-ate 对，其输入为 1 个 G_2 上的点（记为 Q）和 1 个 G_1 上的点（记为 P），计算过程包含 2 个核心运算——Miller 算法与最终模幂（R-ate 对的具体计算过程可参考文献[1]附录 C）。在 SM9 算法中，运行密钥交换协议的双方以及解封装过程中，均需要计算 1 次以用户加密私钥 de（G_2 上的点，即 $Q=de$）为输入的 R-ate 对[2]。如果参与 R-ate 对计算的 Q 点是固定的，则可预先计算出该点在 Miller 算法执行过程中产生的一系列中间参数，以提高计算效率。

3.1 Miller 算法预计算数据生成算法

Miller 算法中的线函数的输出结果为 $F_{q^{12}}$ 上的稀疏元素（记为 $\boldsymbol{A}$），令 $P=(x_p, y_p)$，$\boldsymbol{A}$ 用 F_{q^2} 上的元素可表示为（该表示方法可参考文献[1]附录 A.2）：

$$\boldsymbol{A}=(\boldsymbol{0}, a_4\cdot x_p, \boldsymbol{0}, \boldsymbol{0}, a_1\cdot y_p, a_0) \tag{1}$$

要生成的预计算数据为线函数中不涉及 P 点坐标的中间参数（即 a_0，a_1 和 a_4），该不完整的“线函数”记为 g'，其输入为 T 和 Q（当 $T=Q$ 时表示求切线），输出为 a_0，a_1，a_4。

SM9 用户在收到 KGC 提供的 de 后，即可生成 de 的预计算数据，生成算法如算法 2 所示。算法 2 中体现的二进制非相邻表示型（2-NAF）优化的详细描述可参考文献[3]。

算法 2 Miller 算法预计算数据生成算法

输入：G_2 上的点 Q，a 的 2-NAF 数组 a_list，a_list 的长度 l
输出：列表 M_list
1. T ← Q, nQ ← −Q, M_list ← []
2. for i from l−2 to 0 do
3. M_list.append(g′(T, T))
4. T ← double(T)
5. if a_list[i]=1 then
6. M_list.append(g′(T, Q))
7. T ← T+Q
8. end if
9. if a_list[i]=$\bar{1}$ then
10. M_list.append(g′(T, nQ))
11. T ← T+nQ
12. end if
13. end for
14. Q1 ← π_q(Q), nQ2 ← π_{q^2}(nQ)
15. M_list.append(g′(T, Q1))
16. M_list.append(g′(T+Q1, nQ2))

算法 2 主要包含了 R-ate 对 Miller 算法中的线函数（实际为函数 g'）、倍点、点加和 Frobenius 映射（π_q 和 π_{q^2}），这些运算只与 Q 点相关。预计算数据的每一组 a_0，a_1 和 a_4 均为 F_{q^2} 上的元素，故每组大小为 256 b×2×3=192 B。对于 SM9 国标约定的参数（见文献[1]附录 A.1），g' 一共需执行 77 次，故 de 的预计算数据大小为 77×192 B≈14.4 kB。

3.2 采用预计算的 Miller 算法

对于有上述预计算数据的 Q 点，优化后的 Miller 算法如算法 3 所示。其中，$sqr(x)$ 表示 F_{q12} 元素 x 的平方；g'' 表示线函数中 g' 未执行的部分，即按式(1)计算 F_{q12} 上的元素 $\boldsymbol{A}$。

算法 3 采用预计算的 Miller 算法

输入：G_1 上的点 P，a 的 2-NAF 数组 a_list，a_list 的长度 l，点 Q 的预计算列表 M_list
输出：$F_{q^{12}}$ 上的元素 f
1. f ← 1, j ← 0
2. for i from l−2 to 0 do
3. f ← sqr(f) * g″(M_list[j], P)
4. j←j+1
5. if a_list[i] !=0 then
6. f ← f * g″(M_list[j], P)
7. j←j+1
8. end if
9. end for
10. f←f * g″(M_list[j], P) * g″(M_list[j+1], P)

可见，对于 de 参与的 Miller 算法，使用算法 3 只需取出预计算数据的每一项，由 g'' 根据 P 点坐标输出 F_{q12} 上的对应元素，再执行 $F_{q^{12}}$ 上的平方和乘法运算。经定量分析，相比原始 Miller 算法，算法 3 减少了 G_2 上的 65 次倍点运算、11 次点加运算和 2 次 Frobenius 映射。

4 最终模幂优化

Miller 算法的输出结果为 $F_{q^{12}}$ 上的元素（记为 f），模幂运算 $f^{(q^{12}-1)/r}$ 称为最终模幂。使用 Python 编程实现 R-ate 对计算，最终模幂耗时约占整个 R-ate 对耗时的 58%，所以对最终模幂的优化将有效提升 R-ate 对的计算效率。

4.1 最终模幂运算的分解

根据文献[7]，最终模幂的指数 $(q^{12}-1)/r$ 可分解为以下 3 个因子：

$$(q^{12}-1)/r=(q^6-1)\cdot(q^2+1)\cdot((q^4-q^2+1)/r) \quad (2)$$

最终模幂运算可分为简单部分 $(f^{p^6-1})^{p^2+1}$ 和困难部分 $f^{(q^4-q^2+1)/r}$。对于简单部分的计算，可参考文献[5]。简单部分的运算结果，已经是分圆子群 $G_{\phi 6}(F_{q^2})$（下文将该乘法群简记为 G_T）上的元素，在困难部分中的求逆只需 1 次共轭，几乎没有计算开销。困难部分的指数可进一步分解：

$$(q^4-q^2+1)/r=q^3+\lambda_2 q^2+\lambda_1 q+\lambda_0 \quad (3)$$

其中：

$$\lambda_2=6t^2+1 \quad (4)$$

$$\lambda_1=-36t^3-18t^2-12t+1 \quad (5)$$

$$\lambda_0=-36t^3-30t^2-18t-2 \quad (6)$$

通过 Python 编程测试得出，困难部分的计算约占最终模幂运算耗时的 97%，因此本文着重考虑困难部分的优化。

4.2 困难部分的构造优化

文献[3,5]按照文献[7]描述的加法链完成困难部分的模幂运算，计算开销为 F_{q12} 上的 43 次乘法、197 次平方和 12 次 Frobenius 映射。本文提出更加高效的困难部分模幂构造途径，如图 1 所示。

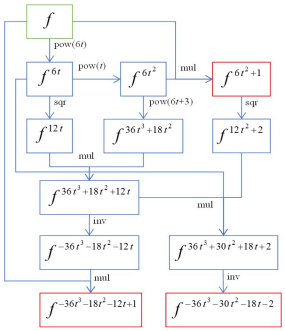


图 1 困难部分模幂的构造途径（电子版为彩图）

Fig. 1 Construction approach of modular exponentiation in hard part

在图 1 中，绿色方框表示简单部分的运算结果（即困难部分的输入），蓝色和红色方框分别表示中间值和输出，pow(x)，mul 和 sqr 分别表示求 x 次幂、乘法和平方，inv 表示求逆（实际为共轭运算）。通过该构造途径，以尽可能少的运算次数生成 $f^{\lambda 0}$，$f^{\lambda 1}$，$f^{\lambda 2}$，再结合 Frobenius 映射，按式(7)完成困难部分模幂运算：

$$f^{(q^4-q^2+1)/r}=f^{\lambda 0}\cdot\pi_q(f^{\lambda 1})\cdot\pi_{q^2}(f^{\lambda 2})\cdot\pi_{q^3}(f) \quad (7)$$

经定量分析，上述方法的计算开销为 $F_{q^{12}}$ 上的 38 次乘法、195 次平方和 8 次 Frobenius 映射，相比文献[3,5]方法在 $F_{q^{12}}$ 乘法、平方和 Frobenius 映射的运算次数上分别减少了约 11.6%，1.0%和 33.3%。这一方面是因为 3 次模幂的指数分别是 t，$6t$，$6t+3$，整体汉明重量减少更多（汉明重量每减少 1，就可减少 1 次 $F_{q^{12}}$ 上的乘法）；另一方面则是充分挖掘了式(4)—式(6)数项之间存在的因子分解或平方关系，将相同或相近的数项尽早合并计算。

4.3 分圆子群上的模幂运算优化

文献[3]为了计算困难部分中 f^t，f^{t^2}，f^{t^3} 这 3 个值，基于 2-NAF 设计了 G_T 上指数为 t 的模幂算法。在本文的困难部分构造途径中，3 次模幂的指数分别是 t，$6t$，$6t+3$，且运行密钥交换协议的双方均需要计算 1 次指数更大、底数和指数均不确定的 G_T 上的模幂，无法直接套用文献[3]中指数固定为 t 的模幂算法。因此，本文设计了 G_T 上适用于任意指数的模幂算法，其伪代码描述如算法 4 所示。

算法 4 G_T 上的模幂算法

输入：G_T 上的底数 x，指数 e

输出：幂值 $y=x^e$

1. 设 3e 的二进制表示为 $h_r h_{r-1}\cdots h_1 h_0$，最高位 h_r 为 1
2. 设 e 的二进制表示为 $u_r u_{r-1}\cdots u_1 u_0$，最高位 u_r 可为 0
3. y ← x，nx ← inv(x)
4. for i from r−1 to 1 do
5. y ← sqr(y)
6. if $h_i=1$ and $k_i=0$ then
7. y ← y * x
8. end if
9. else if $h_i=0$ and $k_i=1$ then
10. y ← y * nx
11. end if
12. end for

需要注意的是，SM9 算法中涉及模幂运算的 F_{q12} 元素，均属于 G_T，故算法 4 中的平方运算 sqr 应采用 G_T 上的平方，其相比 F_{q12} 上的平方运算开销更小。文献[6]对此有详细论述。

5 基于 Comb 固定基的模幂运算优化

在 SM9 算法中，数字签名的生成与验证、运行密钥交换协议的双方以及密钥封装过程中，均需要计算 1 次 F_{q12} 上底数固定的模幂运算[2]。预先计算出该固定底数的若干模幂值，基于 Comb 固定基的方式进行查表完成模幂运算，可显著提高计算效率[6,20]。

5.1 Comb 固定基预计算数据生成算法

虽然 Comb 固定基预计算数据取决于 KGC 的主公钥，可由 KGC 将预计算数据分发给用户，但考虑到预计算数据较大，为节省通信开销，也可由用户自行计算。本文提出高效的 Comb 固定基预计算数据生成算法，其伪代码描述如算法 5 所示。

算法 5 Comb 固定基预计算数据生成算法

输入：F_{q12} 上的元素 x

输出：列表 comb_list

1. comb_list ← [1,x]，t ← x，l ← 2
2. for i from 0 to 6 do
3. for j from 0 to 31 do
4. t ← sqr(t)
5. end for
6. for j from 0 to l−1 do
7. comb_list.append(comb_list[j] * t)
8. end for
9. l←l * 2
10. end for

算法 5 将 256 b 的指数均分成 8 段，生成的预计算数据则包含 $2^8=256$ 个 F_{q12} 上的元素，即每项 Comb 固定基预计算数据的大小为 256 b×12×256=96 kB。如果 SM9 用户既要执行数字签名又要执行加解密，则需要 96 kB×2=192 kB 预计算数据。

5.2 基于 Comb 固定基的模幂算法

基于 Comb 固定基的模幂算法如算法 6 所示。

算法 6 基于 Comb 固定基的模幂算法

输入：x 的预计算列表 comb_list，256b 的指数 e

输出：幂值 $y=x^e$

1. 设 e 的二进制表示为 $e_{255}e_{254}\cdots e_1e_0$
2. y ← 1
3. for i from 31 to 0 do
4. t ← 0
5. for j from 7 to 0 do
6. r←32 * j+i
7. t←t+e_r * 2^j
8. end for
9. y ← sqr(y) * comb_list[t]
10. end for

假设 256 b 的指数在二进制表示下有一半的数位为 1(最高位为 1)，则 $F_{q^{12}}$ 上的普通模幂需要 255 次平方和 127 次乘法；而使用算法 6 计算 $F_{q^{12}}$ 上的固定底数模幂仅需 31 次平方和 31 次乘法。

需要注意的是，类似于算法 4，算法 5 和算法 6 中的 F_{q12} 元素均属于 G_T，故算法 5 和算法 6 中的平方运算也应采用 G_T 上的平方。

6 实验与分析

本文使用 Python 语言编程，完整实现了 SM9 国标[2]规定的算法，以及上述所有优化算法。本章通过对比实验，验证各项优化算法的效果，以及 SM9 各项算法(数字签名的生成与验证、密钥交换、密钥封装与解封装、加密与解密)的实际性能提升幅度。

6.1 实验环境与参数

实验计算机的配置如表 1 所列。

表 1 实验计算机配置

Table 1 Configuration of experimental computer

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

对于各项测试，单次执行耗时不足 100 ms 的，执行 1 000 次；单次执行耗时在 100～1 000 ms 的，执行 100 次；单次执行耗时超过 1 000 ms 的，执行 10 次。取历次执行耗时的平均值为有效数据。经不同平台测试，由于存在硬件性能和软件环境的差异，算法运行速率的绝对值会有所浮动，但相对值是基本稳定的，所以本文的实验结果对于在其他环境运行 SM9 程序或使用其他编程语言实现 SM9 算法也具有参考意义。

6.2 优化算法测试

表 2 列出了本文描述的所有优化算法的单次执行耗时，并引入对应的朴素算法或其他文献提出的算法作为对照。其中，实验组与对照组的输入是完全一致的，标量乘所需的乘数和模幂所需的指数均为 256b 的随机整数。

表 2 各项算法的测试结果

Table 2 Test results of algorithms

算法	耗时/ms
生成 G_1 标量乘预计算数据(本文)	163.36
生成 G_1 标量乘预计算数据(文献[21])	3722.12
生成 G_2 标量乘预计算数据(本文)	385.98
生成 G_2 标量乘预计算数据(文献[21])	7286.31
采用预计算的 G_1 标量乘	0.71
普通 G_1 标量乘	4.43
采用预计算的 G_2 标量乘	1.63
普通 G_2 标量乘	8.75
生成 Miller 算法预计算数据	3.77
采用预计算的 Miller 算法	11.28
普通 Miller 算法	14.86
最终模幂困难部分(本文)	14.40
最终模幂困难部分(文献[3])	15.57
G_T 上的 2-NAF 模幂	23.13
G_T 上的普通模幂	28.50
$F_{q^{12}}$ 各项上的普通模幂	35.58
生成 Comb 固定基预计算数据	39.32
基于 Comb 固定基的模幂	5.37

为了更直观地呈现优化效果，图 2 给出了部分实验组与对照组的相对速率对比。

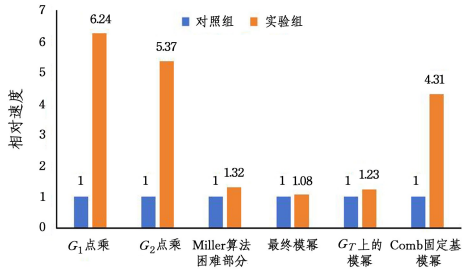


图 2 部分算法优化前后的相对速率

Fig. 2 Relative rates before and after optimization of algorithms

在图 2 中，“G_T 上的模幂”的实验组为表 2 中的“G_T 上的 2-NAF 模幂”，对照组为“G_T 上的普通模幂”；“Comb 固定基模幂”的对照组为“G_T 上的 2-NAF 模幂”。

6.3 SM9 算法综合测试

综合运用本文描述的所有优化算法，测试其在数字签名的生成与验证、密钥交换、密钥封装与解封装、加密与解密这 7 项 SM9 算法中的实际效果。作为对比，优化前的对照组不包含椭圆曲线固定点标量乘预计算、固定 G_2 点的 Miller 算法和基于 Comb 固定基的模幂运算这 3 项优化，其余实现与优化后的实验组一致。密钥交换耗时为密钥交换双方的纯数据计算(包含可选项的计算)耗时之和，不包含网络传输等其他开销。表 3 列出了各项 SM9 算法优化前后的单次执行耗时和执行速率，图 3 呈现了优化前后的相对速率。

表 3 SM9 算法优化前后的测试结果

Table 3 Test results before and after optimization of SM9 algorithms

算法	优化前耗时/ms	优化前速率/(次/s)	优化后耗时/ms	优化后速率/(次/s)
数字签名生成	28.09	35.60	6.21	161.03
数字签名验证	60.89	16.42	33.49	29.86
密钥交换	248.82	4.02	113.38	8.82
密钥封装	32.16	31.09	10.20	98.04
解封装	29.89	33.46	21.61	46.27
加密	35.26	28.36	11.65	85.84
解密	32.00	31.25	24.33	41.10

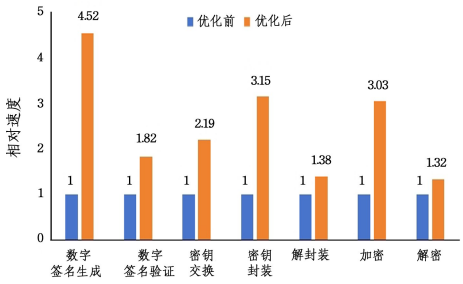


图 3　SM9 算法优化前后的相对速率

Fig. 3　Relative rates before and after optimization of SM9 algorithms

可见，在本文的实验环境下，将所述算法优化综合运用于各项 SM9 算法后，获得了 32%～352%的性能提升，证明了这些优化设计的有效性。

结束语　国密算法 SM9 是我国自主设计的一款安全性高、计算性能良好的标识密码，对于缓解传统公钥基础设施压力、提升自主可控水平有重要意义。本文综合了近些年国内外学者在 SM9 及其数学工具优化方面的研究成果，主要创新点是固定 G_2 点的 Miller 算法、最终模幂困难部分构造和分圆子群上的模幂运算算法，有效提升了 SM9 算法中椭圆曲线标量乘、双线性对、F_{q12} 上的模幂等耗时步骤的计算性能，对于提升 SM9 算法的整体计算效率具有理论价值和实践意义。除了 SM9 算法，本文研究成果对于涉及椭圆曲线、双线性对或有限域扩域等数学工具的其他算法同样具有参考价值。下一步的工作重点是借用二进制程序分析方法中的数据流分析思想，尝试进一步优化最终模幂困难部分构造途径；并设计二维 Comb 固定基模幂算法，适当扩展预计算数据，以换取更快的模幂运算速度。

此外，目前互联网上难找到以 Python 语言原生实现的 SM9 算法以及上述数学工具的开源代码，或其存在实现错误、性能不足等缺陷。而本文实现的一整套 SM9 算法高效 Python 代码已在“码云”平台开源[1)]，一定程度上解决了以上问题。

参 考 文 献

[1] SM9 标识密码算法 第 1 部分：总则：GB/T 38635.1—2020[S]. 北京：全国信息安全标准化技术委员会，2020-04-28.

[2] SM9 标识密码算法 第 2 部分：算法：GB/T 38635.2—2020[S]. 北京：全国信息安全标准化技术委员会，2020-04-28.

[3] HU X Y，HE D B，PENG C，et al. A fast implementation of R-ate pairing in SM9 algorithm[J]. Journal of Cryptologic Reseatch，2022，9(5)：936-948.

[4] GAN Z W，LIAOF Y. Rapid calculation of R-ate bilinear pairing in China state cryptography standard SM9[J]. Computer Engineering，2019，45(6)：171-174.

[5] WANG M D，HE W G，LI J，et al. Optimal design of R-ate pair in SM9 algorithm [J]. Communications Technology，2020，53(9)：2241-2244.

[6] WANG J T，FAN R，HUANG Z. Fast implementation of high power operation in SM9 [J]. Computer Engineering，2023，49(9)：118-124，136.

[7] SCOTT M，BENGER N，CHARLEMAGNE M，et al. On the final exponentiation for calculating pairings on ordinary elliptic curves [C]// Proceedings of the 3rd International Conference on Pairing-Based Cryptography. 2009：78-88.

[8] FU Z. Efficient implementation of Rate bilinear pairing algorithm[D]. Tianjin：Tianjin University，2017.

[9] SUN M W. Research on key technologies of SM9 identification cipher algorithm[D]. Harbin：Harbin University of Science and Technology，2022.

[10] LI J F. Research on SM9 algorithm and FPGA implementation [D]. Xi'an：Xidian University，2021.

[11] WANG B. Research on fast calculations of scalar multiplication and bilinear pairings on elliptic curves[D]. Hefei：University of Science and Technology of China，2021.

[12] YANG G Q. Study on fast implementation algorithms and key techniques for elliptic curve and pairing-based crytography[D]. Jinan：Shandong University，2021.

[13] DUQUESNE S，GHAMMAM L. Memory-saving computation of the pairing final exponentiation on BN curves [J]. Groups Complexity Cryptology，2016，8(1)：75-90.

[14] AZARDERAKHSH R，FISHBEIN D，GREWAL G，et al. Fast software implementations of bilinear pairings[J]. IEEE Transactions on Dependable and Secure Computing，2015，14(6)：605-619.

[15] ZHEN P，HU X，YUY，et al. Research on the optimization computation of SM9 bilinear pairings [C]// Proceedings of the 2nd ACM International Conference on Communication and Information Systems. 2017：256-261.

[16] 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. 2019：78-82.

[17] WU Y，BAI G Q，WUX J. A karatsuba algorithm based accelerator for pairing computation [C]// Proc of the 15th IEEE International Conference on Electron Devices and Solid-State Circuits. 2019：1-3.

[18] KARABINAK. Squaring in cyclotomic subgroups [J]. Mathematics of Computation，2013，82(281)：555-579.

[19] XIE Y，WANG B，ZHANG L，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.

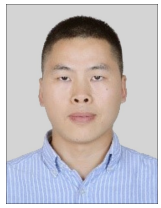
[20] WANG S，FANG L G，HAN L B，et al. Fast implementation of SM9 digital signature and verification algorithms[J]. Communications Technology，2019，52(10)：2524-2527.

[21] XIE Z J，FU W，LUO F. Performance optimization method of Python toolkit for domestic cryptographic algorithm[J]. Journal of Information Securyity Research，2023，9(10)：1001-1007.

1) https://gitee.com/basddsa/hggm

[22] Guanzhi. 支持国密 SM2/SM3/SM4/SM9/SSL 的密码工具箱[EB/OL]. (2023-10-16)[2023-10-18]. https://github.com/guanzhi/GmSSL.
[23] PU L, LIN C, WU W, et al. A public-key encryption with keyword search schemefrom SM9[J]. Journal of Cyber Security, 2023, 8(1): 108-118.
[24] LAI J C, HUANG X Y, HED B, et al. An efficient hierarchical identity-based encryption based on SM9[J]. SCIIENTIA SINICA Informmations, 2023, 53(5): 918-930.
[25] LIU K, NING J T, WU W, et al. Multi-ciphertext batch auditable decryption outsourcing SM9-HIBE key encapsulation mechanism[J]. Journal on Communications, 2023, 44(12): 158-170.
[26] LI C, LIANG J K, DINGY J, et al. Hierarchical identity-based broadcast inner product functional encryption based on SM9[J]. SCIIENTIA SINICA Informmations, 2024, 54(6): 1400-1418.
[27] CUI Y, HUANG X Y, LAIJ C, et al. Anonymous broadcast encryption based on SM9[J]. Journal of Cyber Security, 2023, 8(6): 15-27.
[28] AN H Y, HE D B, BAO Z J, et al. Ring signature based on the SM9 digital signature and its application in blockchain privacy protection[J]. Journal of Computer Research and Development, 2023, 60(11): 2545-2554.
[29] 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): 104-117.

XIE Zhenjie, born in 1995, Ph.D candidate. His main research interests include cloud security and cryptography applications.



CAI Ruijie, born in 1990, Ph.D candidate, lecturer. His main research interests include network security, binary code analysis and vulnerability discovery.

(责任编辑:柯颖)

项目发布|2025 年度 CCF-腾讯犀牛鸟基金申报工作正式启动

CCF-腾讯犀牛鸟基金于 2013 年由 CCF 和腾讯共同发起。13 年来,犀牛鸟基金致力于为海内外青年学者搭建产学合作创新的平台,推动科技在产业创新和社会发展中持续发挥价值。本年度犀牛鸟基金申报工作于即日起正式启动,申报截止时间为 2025 年 6 月 15 日 24:00(北京时间)。

一、基金简介

本年度犀牛鸟基金设立 6 个技术领域,共 39 项研究命题,关注人工智能核心技术突破,涵盖医疗健康、金融服务、智慧能源、网络安全等诸多方向,涉及包括大模型及其应用研究、强化学习及 AI Agent、生成式人工智能技术、安全与量子计算、数据库及数据挖掘技术等。每位申请者可选择其中一项(且仅一项)研究课题进行项目申报。

基金入选者还将有机会参与"犀牛鸟学者奖励计划":该计划将为合作过程中展现出优秀学术能力或合作成果具有较高应用价值的项目,提供个人或项目奖励,支持学者的个人发展,鼓励合作双方进行更加深入的实践探索。

二、申报条件

本基金将面向符合如下条件的海内外高校及科研院所青年学者展开:

男性申请人是 1989 年 1 月 1 日(含),女性申请人是 1984 年 1 月 1 日(含)之后出生的高校/科研院所在职的全职教师或研究人员。博士后需在出站后参与申报;

硕士/博士毕业后在高校/科研院所累计任职时间(博士后期间不计算在内):男性不超过 5 年,女性不超过 10 年;

能独立进行研究工作,并带领学生团队共同参与课题研究与实践。

三、基金流程

年份	时间	事项
2025 年	5 月 15 日	指南发布,申请启动
	6 月 15 日	申请截止,评审启动
	7 月 31 日	评审截止,结果发布
	8 月	企业调研,项目立项
	10 月	CNCC2025,项目颁奖
2026 年	3 月	中期检查,报告提交
	9 月 30 日	项目结束,成果提交
	10 月	CNCC2026,结题答辩
项目进行过程中的具体时间节点,请关注 CCF-腾讯联合项目组通知。		

四、基金申报

欢迎海内外优秀青年学者申报,申报截止时间为 2025 年 6 月 15 日 24:00(北京时间)。

移动端点击下方小程序阅读项目指南,如需提交申请,请在 PC 端打开链接进行申报:https://www.wizsci.com/project/detail/1748。基金项目负责人:王晓然(karliewang@tencent.com)

据 CCF 微信公众号