软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007164] [CSTR: 32375.14.jos.007164]

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

素阶数域上的高效格基数字签名方案*

董怡帆[1], 方博越[1], 梁志闯[1], 赵运磊[1,2]

[1](复旦大学 计算机科学技术学院, 上海 200433)

[2](密码科学技术全国重点实验室, 北京 100878)

通信作者: 赵运磊, E-mail: ylzhao@fudan.edu.cn

摘　要: 随着量子计算的快速发展, 特别是 Shor 量子算法及其变体的优化进步, 当前基于大整数分解和离散对数问题的经典公钥密码体制将面临颠覆性的影响. 为了应对量子攻击, 学界开始对后量子密码学的研究, 其中基于格的后量子密码方案因其在安全、效率、带宽等方面的均衡表现和良好的可扩展性而成为后量子密码的主流技术路线. 目前, 基于格的后量子密码方案大多使用分圆环, 尤其是二次幂分圆环作为底层代数结构. 但分圆环中具有丰富的子域、自同构、环同态等代数结构, 容易遭受针对性攻击. 基于具有"高安全性、素数阶、大 Galois 群和惰性模数"特点的素阶数域, 设计出后量子数字签名方案 Dilithium-Prime, 并给出推荐参数集. 然而, 素阶数域的一个显著缺点是无法直接使用快速数论变换 (NTT) 算法进行高效的多项式乘法, 导致素阶数域上的密码方案性能较差. 为此, 设计素阶数域上的 NTT 算法和小多项式乘法, 实现素阶数域上高效的多项式乘法. 最后, 为方案的关键算法设计常数时间无分支实现方法, 给出方案的 C 语言实现, 并与其他方案进行对比. 实验结果表明, 在同一安全等级下, 与分圆环上的数字签名方案 CRYSTALS-Dilithium 推荐参数相比, Dilithium-Prime 方案的公钥尺寸、私钥尺寸、签名尺寸分别降低 1.8%、10.2%、1.8%, 签名算法效率提高 11.9%, 密钥生成算法、验证算法所需时间分别为 CRYSTALS-Dilithium 方案的 2.0 倍和 2.5 倍, 但不同于 CRYSTALS-Dilithium, Dilithium-Prime 方案具有抵抗针对分圆环的密码攻击的优越特性; 与 2023 年韩国后量子密码算法竞赛中提出的基于素阶数域的签名方案 NCC-Sign 推荐参数相比, 在相同的安全等级和带宽条件下, Dilithium-Prime 方案的密钥生成算法、签名算法、验证算法的速度分别提升至 4.2 倍、35.3 倍、7.2 倍, 实现兼顾高效性和安全性的素阶数域签名算法.

关键词: 后量子密码; 格密码; 素阶数域; 数字签名方案; 快速数论变换; 小多项式乘法

中图法分类号: TP309

中文引用格式: 董怡帆, 方博越, 梁志闯, 赵运磊. 素阶数域上的高效格基数字签名方案. 软件学报. http://www.jos.org.cn/1000-9825/7164.htm

英文引用格式: Dong YF, Fang BY, Liang ZC, Zhao YL. Efficient Lattice-based Digital Signature Scheme in Large -Galois-group Prime-degree Prime-ideal Field. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7164.htm

Efficient Lattice-based Digital Signature Scheme in Large-Galois-group Prime-degree Prime-ideal Field

DONG Yi-Fan[1], FANG Bo-Yue[1], LIANG Zhi-Chuang[1], ZHAO Yun-Lei[1,2]

[1](School of Computer Science, Fudan University, Shanghai 200433, China)

[2](State Key Laboratory of Cryptology, Beijing 100878, China)

Abstract: With the rapid development of quantum computing, especially the optimization and progress of the Shor quantum algorithm and its variants, the current classical public-key cryptography based on factoring large integers and discrete logarithm problems is facing serious security threats. To cope with quantum attacks, post-quantum cryptography has been proposed, among which lattice-based

* 基金项目: 国家自然科学基金 (61877011); 国家重点研发计划 (2022YFB2701600); 上海科技创新行动计划技术标准项目 (21DZ2200500)
收稿时间: 2023-10-19; 修改时间: 2024-01-03; 采用时间: 2024-02-01; jos 在线出版时间: 2024-10-30

cryptography is commonly viewed as the most promising one due to its outstanding performance in security, bandwidth, and efficiency. Most of the existing lattice-based post-quantum cryptographic schemes use cyclotomic rings, especially power-of-two cyclotomic rings, as their underlying algebraic structures. However, targeted attacks against cyclotomic rings have been proposed, exploiting subfields, small Galois groups, and ring homomorphisms in these rings. This study uses the large-Galois-group prime-degree prime-ideal field as the new underlying algebraic structure, which has characteristics of high security, prime order, large Galois group, and inert modulus.First, this study proposes a post-quantum digital signature scheme based on the large-Galois-group prime-degree prime-ideal field, which is named Dilithium-Prime, and the recommended parameter sets are provided. Next, considering that the traditional number theory transform (NTT) algorithm cannot be used to multiply polynomials efficiently in the large-Galois-group prime-degree prime-ideal field, this study designs efficient polynomial multiplication strategies for Dilithium-Prime, including NTT for the large-Galois-group prime-degree prime-ideal field and small polynomial multiplication. Finally, this study provides a portable C language implementation of Dilithium-Prime, along with the implementation details and constant-time implementation skills, and compares Dilithium-Prime with other lattice-based digital signature schemes. The experimental results show that the public key size, secret key size, and signature size of Dilithium-Prime are reduced by 1.8%, 10.2%, and 1.8%, respectively, compared to CRYSTALS-Dilithium. The efficiency of the signature algorithm is improved by 11.9%, and the key generation algorithm and the verification algorithm are 2.0× and 2.5× slower than those of CRYSTALS-Dilithium, respectively. However, Dilithium-Prime can withstand the cryptographic attack against cyclotomic rings, which is exactly what CRYSTALS Dilithium lacks. Compared to NCC-Sign, Dilithium-Prime's key generation algorithm, signature algorithm, and verification algorithm are 4.2×, 35.3×, and 7.2× faster, respectively, than those of NCC-Sign under the same security level and bandwidth.

Key words: post-quantum cryptography; lattice-based cryptography; large-Galois-group prime-degree prime-ideal field; digital signature scheme; number theory transform; small polynomial multiplication

1 引 言

作为网络空间安全的基础与关键支撑, 密码方案的构建以数学上的计算困难性问题为基础, 其安全性依赖于这些数学困难问题的难解性. 随着量子计算技术的发展, 许多经典算法无法破解的困难问题能够被量子算法攻破, 例如 Shor 量子算法[1]及其最新进展[2]能够在多项式时间内解决大整数分解问题和离散对数问题, 这也正是现有大部分公钥密码体制依赖的困难性问题.

量子计算的发展意味着经典密码体制遭受到了颠覆性挑战, 因此密码学界开始了对后量子密码学 (post-quantum cryptography, PQC) 的研究, 致力于构建能够抵抗量子攻击的密码算法. 2016 年美国国家标准技术研究所 (National Institute of Standards and Technology, NIST) 举办了后量子密码征集项目[3]; 2019 年, 中国密码学会也面向国内的学者举办了后量子密码算法竞赛[4]. 这些项目的征集内容包括公钥加密方案 (public key encryption, PKE)、密钥封装方案 (key encapsulation mechanism, KEM)、数字签名方案 (digital signature) 这 3 个密码原语, 入选的密码方案包括基于格、基于编码、基于多变量、基于哈希、基于同源等技术路线.

基于格的后量子密码方案依赖于 SIS、LWE 等格上困难问题[5–7], 已被证实具有抗量子特性以及最差情况到一般情况的安全规约[8], 并且具有较高的计算效率和较好的可扩展性, 成为 NIST 后量子密码标准征集项目的有力竞争者. 在 NIST 第 3 轮征集中甄选出的 7 个后量子密码方案中, 有 5 个基于格的密码方案, 包含 3 个 PKE/KEM 方案和 2 个数字签名方案; 在第 1 批拟标准化的 4 个后量子密码算法中, 有 3 个基于格的密码方案.

已知的基于格的后量子密码方案大多使用分圆环, 尤其是 2 次幂分圆环作为底层代数结构. 例如, NIST 第 3 轮的 PKE/KEM 方案 CRYSTALS-Kyber[9]、Saber[10], 数字签名方案 CRYSTALS-Dilithium[11]、Falcon[12]均使用了 2 次幂分圆环 $\mathbb{Z}_q/(x^n+1)$, 其中 $n=2^k$. 这些分圆环允许使用快速数论变换 (number theory transform, NTT) 进行高效的乘法运算, 能够构建高效的密码方案. 然而另一方面, 敌手可能利用分圆环中丰富的子域、自同构、环同态等代数结构进行针对性的密码分析[13–17], 使得基于分圆环的密码方案受到相对于一般整数格的额外安全威胁.

Bernstein 等人[18,19]认为, 在设计密码方案的过程中, 应尽量避免不必要的代数结构, 以此减少敌手可能的攻击手段. 尽管目前并没有有效的算法能够攻破使用分圆环的后量子密码算法, 但对于密码分析而言, 未雨绸缪是有必要的.

Bernstein 等人在文献 [18,19] 中建议将密码方案的底层代数结构从有较多隐患的分圆环转移到具有“高安全

性、素数阶、大 Galois 群和惰性模数”特点的多项式环 $\mathbb{Z}_q[x]/(x^n-x-1)$ 上. 在多项式环 $\mathbb{Z}_q[x]/(x^n-x-1)$ 中, n 和 q 为素数, n 满足多项式 x^n-x-1 对应的 Galois 群足够大且同构于阶数为 $n!$ 的置换群 S_n; q 满足 x^n-x-1 是 $\mathbb{Z}_q[x]$ 上的不可约多项式, 此时 x^n-x-1 生成一个素理想, 进而商环 $\mathbb{Z}_q[x]/(x^n-x-1)$ 构成一个素数阶且基于素理想的域, 称之为素阶数域. 素阶数域具有代数结构单一且 Galois 群较大的特点, 能够较好地抵抗分圆环上常见的子域攻击、自同构攻击等攻击手段.

然而, 素阶数域的一个显著缺点是无法直接使用 NTT 算法进行高效的多项式乘法, 导致素阶数域上的密码方案性能较差. 文献 [20] 给出了一种将 NTT 运算转移到比素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$ 更大的 NTT 友好环上进行的方法, 使得基于素阶数域的密码方案也可以进行高效多项式乘法. 目前, 使用素阶数域的 KEM 方案 NTRU-Prime[18]已达到与传统分圆环密码方案相当的性能, 并在 OpenSSH 等国际标准中得到实际应用[21]. 针对其他密码原语, 如何基于素阶数域设计安全高效的格密码方案值得进行进一步研究.

本文贡献在于设计一种基于素阶数域的数字签名算法, 以对抗针对分圆环的潜在攻击. 同时设计素阶数域签名算法的高效多项式乘法和相关常数时间实现算法, 使得素阶数域签名方案在保障安全性的同时兼顾高效性.

本文的主要贡献包括以下 4 个部分.

(1) 本文基于 Fiat-Shamir with Aborts 技术和 CRYSTALS-Dilithium 方案的改进策略, 提出了基于素阶数域的数字签名方案, 记为 Dilithium-Prime. 本文给出了 Dilithium-Prime 方案的 3 组推荐参数集, 并提供了 Dilithium-Prime 方案的正确性分析、安全性规约和具体安全强度的分析. 除此之外, 本文简单梳理了传统分圆环的代数特点和安全隐患, 并简单介绍了素阶数域在抵抗子域攻击和自同构攻击等的安全优势.

(2) 本文针对 Dilithium-Prime 方案的 3 组参数, 设计了针对性的高效多项式乘法方案, 包括针对小系数多项式的小多项式乘法和 $\mathbb{Z}_q[x]/(x^n-x-1)$ 上通用的素阶数域 NTT 算法. 本文给出了多项式乘法方案的设计思路和具体算法流程, 并分别给出了小多项式乘法和素阶数域 NTT 算法相比于 School Book 乘法的效率提升结果.

(3) 本文设计了高低位分解算法和模约减算法的高效无分支常数时间实现算法, 它们利用位运算的特性和近似计算的思想, 确保了算法在常数时间内进行, 并且使用秘密值的计算过程中不存在判断分支, 防止在利用秘密值进行判断的过程中泄露秘密值信息, 以此抵抗计时攻击等侧信道攻击方法.

(4) 本文给出了 Dilithium-Prime 方案对应安全等级III的一组参数的 C 语言实现, 并且在相同的安全等级下, 给出了 Dilithium-Prime 方案与同类算法的性能对比. 实验结果表明, 在同一安全等级下, 与 CRYSTALS-Dilithium 方案推荐参数相比, Dilithium-Prime 方案的公钥尺寸、私钥尺寸、签名尺寸分别降低 1.8%、10.2%、1.8%, 签名算法效率提高 11.9%, 密钥生成算法、验证算法所需的时间分别为 CRYSTALS-Dilithium 方案的 2.0 倍和 2.5 倍, 但不同于 CRYSTALS-Dilithium, Dilithium-Prime 方案具有抵抗针对分圆环的密码攻击的优越特性. 与 2023 年韩国后量子密码算法竞赛中提出的基于素阶数域的签名方案 NCC-Sign 推荐参数相比, 在同一安全等级和非常接近的带宽条件下, Dilithium-Prime 方案的密钥生成算法、签名算法、验证算法的速度分别提升至 4.2 倍、35.3 倍、7.2 倍.

2 相关工作

基于格的数字签名方案研究开始于 20 世纪末, Ajtai[22]给出了格中困难问题从最坏情况到一般情况的规约, 奠定了基于格的密码方案的基础. 随后 Goldreich 等人[23]和 Hoffstein 等人[24]先后提出了早期的格签名方案 GGH 和 NTRUSign, 然而 Nguyen 等人[25]指出以上方案使用的格上陷门函数会导致秘密值的泄露, 并给出了完全攻破 GGH 类签名方案的方法.

为了应对此前格密码方案存在的缺陷, Gentry 等人[26]基于原像采样函数等格上陷门工具, 提出了 Hash-and-Sign 签名模式和对应的 GPV 数字签名方案, 该方案在随机预言机模型 (random oracle model, ROM) 下具有可证明的安全性. Lyubashevsky[27]基于 Fiat-Shamir 范式引入拒绝采样技术, 构造了一种无陷门的“Fiat-Shamir with Aborts”签名方案, 并将安全性规约到最坏情况的格困难问题上. 随后, Güneysu 等人[28]和 Bai 等人[29]先后给出了针对 Fiat-

Shamir with Aborts 签名方案的不需要高斯采样的压缩技术.

Hash-and-Sign 与 Fiat-Shamir with Aborts 是当前格密码方案的两种主要技术路线. 在 NIST 拟标准化的两个格基签名方案中, Falcon 等人[12]的方案是基于 Hash-and-Sign 范式的, CRYSTALS-Dilithium 方案[11]则基于 Fiat-Shamir with Aborts 范式. CRYSTALS-Dilithium 方案的构造基于 2 次幂分圆环 $\mathbb{Z}_q[x]/(x^n+1)$, 在 Güneysu 等人[28]和 Bai 等人[29]工作的基础上, 进一步压缩了公钥大小, 其安全性依赖于 MSIS 问题和 MLWE 问题. NIST 最终标准化的版本中, CRYSTALS-Dilithium 预计更名为 ML-DSA, 从和当前文档和参考文献保持一致的角度本文仍采用 CRYSTALS-Dilithium 这一名称. 2023 年韩国 PQC 算法竞赛第一轮征集的方案中, Shim 等人给出了第 1 个基于素阶数域的签名方案 NCC-Sign[30], 使用 RLWE 和 RSIS 作为方案的底层困难假设, 然而该方案受到素阶数域无法使用 NTT 算法这一特性的限制, 方案效率极差. 事实上, 目前基于格的高效后量子签名方案均基于分圆环构造, 存在一定的安全隐患, 而基于素阶数域的签名方案则存在不可忽视的效率缺陷.

3 预备知识

本节主要介绍在本文中使用的预备知识, 包括符号和定义、密码原语和困难性假设. 个别没有在本节介绍的术语, 将在第 1 次使用时进行介绍.

3.1 符号和定义

记 $\mathbb{Z}$ 为整数集, $\mathbb{R}$ 为实数集, 对于 $x\in\mathbb{R}$,$\lceil x\rfloor$ 代表 x 向上取整的值, 同理 $\lfloor x\rfloor$ 代表 x 向下取整的值. n 和 q 为正整数, 在本文中 $n=251, q=7681537$. 符号 $\mathbb{Z}_q$ 定义为 $\mathbb{Z}_q=\mathbb{Z}/q\mathbb{Z}\cong\{0,1,\ldots,q-1\}$,$\mathbb{Z}_q^\times$ 代表 $\mathbb{Z}_q$ 中可逆元素构成的集合.

记 $\mathcal{R}$ 和 $\mathcal{R}_q$ 分别为多项式环 $\mathcal{R}=\mathbb{Z}[x]/(x^n-x-1)$ 和 $\mathcal{R}_q=\mathbb{Z}_q[x]/(x^n-x-1)$, 它们中的元素是 n 维多项式且系数分别在 $\mathbb{Z}$ 或 $\mathbb{Z}_q$ 中. 除第 5.4.2 节和本节有说明的部分外, 本文使用小写字母表示 $\mathcal{R}$ 或 $\mathcal{R}_q$ 中的元素, 小写黑体字母表示由 $\mathcal{R}$ 或 $\mathcal{R}_q$ 中的元素构成的列向量, 大写黑体字母表示 $\mathcal{R}$ 或 $\mathcal{R}_q$ 中的元素构成的矩阵. 对于列向量 $\boldsymbol{v}$ 和矩阵 $\mathbf{A}$, 定义 $\boldsymbol{v}^{\mathrm{T}}$ 和 $\mathbf{A}^{\mathrm{T}}$ 分别为它们的转置. 第 5.4.2 节中, 上述字母代表 $\mathbb{Z}$ 或 $\mathbb{Z}_q$ 中的元素、向量或矩阵.

与文献 [11] 的符号系统类似. 对于正偶 (奇) 数 α, 定义 $r'=r \bmod^{\pm}\alpha$ 表示 r 的绝对最小完全剩余, 此时 r' 落在区间 $(-\alpha/2,\alpha/2]$ 内 (α 为奇数时落在区间 $[-(\alpha-1)/2,(\alpha-1)/2]$ 内); $r'=r \bmod^{+}\alpha$ 表示 r 的非负最小完全剩余, 即此时 r' 落在区间 $[0,\alpha)$ 内; 不需要严格区分以上两种情况时表示为 $r'=r \bmod \alpha$.

对于 $w\in\mathbb{Z}_q$, 定义 $\|w\|_\infty=\left|w \bmod^{\pm} q\right|$ 为 ω 的无穷范数 ℓ_∞, 对于 $\mathcal{R}_q$ 上的多项式 $w=\sum_{i=0}^{n-1}w_i x^i$, 它的无穷范数 ℓ_∞ 和 2-范数 ℓ_2 定义如下:

$$\|w\|_\infty=\max_i\|w_i\|_\infty,$$

$$\|w\|_2=\sqrt{\|w_0\|_\infty^2+\ldots+\|w_{n-1}\|_\infty^2}.$$

类似地, 对于多项式向量 $\boldsymbol{w}=(w_1,\ldots,w_k)\in\mathcal{R}_q^k$, 无穷范数 ℓ_∞ 和 2-范数 ℓ_2 定义为:

$$\|\boldsymbol{w}\|_\infty=\max_i\|w_i\|_\infty,$$

$$\|\boldsymbol{w}\|_2=\sqrt{\|w_1\|_2^2+\ldots+\|w_k\|_2^2}.$$

定义 $\mathcal{R}$ 中所有满足 $\|\boldsymbol{w}\|_\infty\leqslant\eta$ 的元素 w 组成的集合为 $\mathcal{S}_\eta$, 定义集合 $\tilde{\mathcal{S}}_\eta=\{w \bmod^{\pm} 2\eta : w\in\mathcal{R}\}$.

3.2 密码原语

一个数字签名方案 Π 包含 3 个多项式时间内的算法 $(KeyGen, Sign, Verify)$.

(1) 密钥生成算法 $KeyGen$: 输入安全参数 1^λ, 输出公私钥对 (pk, sk).

(2) 签名算法 $Sign$: 输入私钥 sk 和签名消息 $M\in\{0,1\}^*$, 输出签名 σ.

(3) 验证算法 $Verify$: 输入公钥 pk, 签名消息 $M\in\{0,1\}^*$ 和签名 σ, 当验证成功时输出 1, 验证失败输出 0.

数字签名方案 $\Pi=(KeyGen, Sign, Verify)$ 的正确性指, 对于任意足够大的安全参数 1^λ 和公私钥对 $(pk, sk)\leftarrow KeyGen(1^\lambda)$, 算法满足:

$$\Pr[Verify(pk,M,Sign(sk,M))=1]=1.$$

3.3 困难性假设

本文构造的方案主要基于 3 个格上困难问题: MLWE 问题、MSIS 问题和 SelfTargetMSIS 问题. 它们的具体定义如下.

定义 1. MLWE 假设[8]. 给定正整数 l, k, 和 $\mathcal{R}_q$ 上的概率分布 $\mathcal{D}:\mathcal{R}_q\to[0,1]$, 在 $\mathcal{R}_q^{k\times l}$ 上随机采样矩阵 $\mathbf{A}$, 并且根据概率分布 $\mathcal{D}$ 采样 $\boldsymbol{s}_1\leftarrow\mathcal{D}^l$ 和 $\boldsymbol{s}_2\leftarrow\mathcal{D}^k$. 判定性 MLWE 问题指的是区分随机采样的 $(\mathbf{A},\boldsymbol{t}\leftarrow\mathcal{D}^k)$ 和 LWE 采样 $(\mathbf{A},\mathbf{A}\boldsymbol{s}_1+\boldsymbol{s}_2)$, 判定性 MLWE 问题的困难性指对于任意概率多项式时间内的敌手 $\mathcal{A}$, 解决 MLWE 问题的优势 $\boldsymbol{Adv}_{l,k,\mathcal{D}}^{\mathrm{MLWE}}(\mathcal{A})$ 是可忽略的, 其中:

$$\boldsymbol{Adv}_{k,l,\mathcal{D}}^{\mathrm{MLWE}}(\mathcal{A}):=\left|\Pr\left[b=1:\begin{array}{c}\mathbf{A}\leftarrow\mathcal{R}_q^{k\times l};\\ \boldsymbol{t}\leftarrow\mathcal{D}^k;\\ b\leftarrow\mathcal{A}(\mathbf{A},\boldsymbol{t})\end{array}\right]-\Pr\left[b=1:\begin{array}{c}\mathbf{A}\leftarrow\mathcal{R}_q^{k\times l};\\ \boldsymbol{s}_1\leftarrow\mathcal{D}^l,\boldsymbol{s}_2\leftarrow\mathcal{D}^k;\\ b\leftarrow\mathcal{A}(\mathbf{A},\mathbf{A}\boldsymbol{s}_1+\boldsymbol{s}_2)\end{array}\right]\right|.$$

定义 2. MSIS 假设[8]. 给定 $\mathbf{A}\leftarrow\mathcal{R}_q^{k\times l}$, Hermite 标准形式的 MSIS 问题指找到范数足够小的向量 $\boldsymbol{y}\in\mathcal{R}_q^{k+l}$, 满足 $[\mathbf{I}\,|\,\mathbf{A}]\cdot\boldsymbol{y}=\mathbf{0}$, 其中 $\mathbf{0}$ 表示全 0 向量. MSIS 问题的困难性指对于任意概率多项式时间内的敌手 $\mathcal{A}$, 解决 MSIS 问题的可能性 $\boldsymbol{Adv}_{k,l,\gamma}^{\mathrm{MSIS}}(\mathcal{A})$ 是可忽略的, 其中:

$$\boldsymbol{Adv}_{k,l,\gamma}^{\mathrm{MSIS}}(\mathcal{A}):=Pr\left[0<\|\boldsymbol{y}\|_\infty\leqslant\gamma\wedge[\mathbf{I}\,|\,\mathbf{A}]\cdot\boldsymbol{y}=0:\begin{array}{c}\mathbf{A}\leftarrow\mathcal{R}_q^{k\times l};\\ \boldsymbol{y}\leftarrow\mathcal{A}(\mathbf{A})\end{array}\right].$$

SelfTargetMSIS 问题[31]的困难性依赖于 MSIS 问题的困难性和哈希函数 H 的安全性, 在经典随机预言机模型中, 存在从 MSIS 问题到 SelfTargetMSIS 问题的安全规约, 文献 [31] 给出了这一规约过程的概览.

定义 3. SelfTargetMSIS 假设[31]. 设 $H:\{0,1\}^*\to\mathcal{B}_\tau$ 是密码哈希函数, SelfTargetMSIS 问题指找到范数足够小的向量 $\boldsymbol{y}=\begin{bmatrix}\boldsymbol{r}\\ c\end{bmatrix}\in\mathcal{R}_q^{k+l}$ 和 μ, 使得 $H(\mu|[\mathbf{I}\,|\,\mathbf{A}]\cdot\boldsymbol{y})=c$. SelfTargetMSIS 问题的困难性指对于任意概率多项式时间内的敌手 $\mathcal{A}$, 解决 SelfTargetMSIS 问题的可能性 $\boldsymbol{Adv}_{H,l,k,\gamma}^{\mathrm{SelfTargetMSIS}}(\mathcal{A})$ 是可忽略的, 其中:

$$\boldsymbol{Adv}_{H,k,l,\gamma}^{\mathrm{SelfTargetMSIS}}(\mathcal{A}):=\Pr\left[0<\|\boldsymbol{y}\|_\infty\leqslant\gamma\wedge H(\mu||[\mathbf{I}\,|\,\mathbf{A}]\cdot\boldsymbol{y})=c\ :\ \begin{array}{c}\mathbf{A}\leftarrow\mathcal{R}_q^{k\times l};\\ \left(\boldsymbol{y}:=\begin{bmatrix}\boldsymbol{r}\\ c\end{bmatrix},\mu\right)\leftarrow\mathcal{A}^{|H(\cdot)\rangle}(\mathbf{A})\end{array}\right].$$

3.4 School Book 乘法

对于多项式环 $\mathcal{R}_q=\mathbb{Z}_q[x]/(x^n-x-1)$ 的乘法 $h=f\cdot g$, 设 $f=\sum_{i=0}^{n-1}f_ix^i,g=\sum_{j=0}^{n-1}g_jx^j$, 设 $h=\sum_{k=0}^{n-1}h_kx^k$, 此时有如下结论:

(1) 对于 $k=0$, 有:

$$h_k=f_0g_0+\sum_{i=1}^{n-1}f_ig_{n-i}(\mathrm{mod}^+q).$$

(2) 对于 $1\leqslant k\leqslant n-2$, 有:

$$h_k=\sum_{i=0}^{k}f_ig_{k-i}+\sum_{i=k}^{n-1}f_ig_{n-1+k-i}+\sum_{i=k+1}^{n-1}f_ig_{n+k-i}(\mathrm{mod}^+q).$$

(3) 对于 $k=n-1$, 有:

$$h_k=\sum_{i=0}^{n-1}f_ig_{n-1-i}+f_{n-1}g_{n-1}(\mathrm{mod}^+q).$$

事实上, School Book 算法即为公式化表示的普通多项式乘法, 根据以上公式, 当 $k=0$ 时, 需要 n 个乘法; 当 $1\leqslant k\leqslant n-2$ 时, 需要 $2n-k$ 个乘法; 当 $k=n-1$ 时, 需要 $n+1$ 个乘法. 并且整个 School Book 乘法共需要 n 个模约减.

可以看出 School Book 乘法多数用于低次多项式相乘, 若多项式次数较高, 则需要更高效的算法.

3.5 快速数论变换

快速数论变换 (NTT) 是快速傅里叶变换 (fast Fourier transform, FFT) 在有限域上的一种特殊形式, 它的本质

是利用多项式的点值表示法进行高效的多项式乘法运算.

使用 NTT 计算高次多项式乘法 $h=f\cdot g$ 的基本流程为 $h=\mathrm{INTT}(\mathrm{NTT}(f)\circ\mathrm{NTT}(g))$, 其中 NTT 代表正向变换, 将多项式的系数表达式 f,g 转换为点值表达式 $\hat{f},\hat{g}$, INTT 为逆向变换, 将多项式从点值表达式 $\hat{h}$ 转换回系数表达式 h, "$\circ$"则代表着两个点值多项式之间的点乘. 文献 [32–34] 描述了使用快速算法 FFT Trick 结合 Cooley-Tukey 蝴蝶变换和 Gentlemen-Sande 蝴蝶变换以 $\mathrm{O}(n\log n)$ 的时间复杂度计算 NTT 的正逆向操作.

FFT Trick 的本质是多项式商环上的中国剩余定理 (Chinese remainder theorem, CRT), 其同构映射为:

$$\mathbb{Z}_q[x]/(g_1g_2\ldots g_k)\cong\prod_{i=1}^{k}\mathbb{Z}_q[x]/(g_k),$$

$$f\mapsto(f \bmod g_1, f \bmod g_2,\ldots, f \bmod g_k),$$

其中, $g_1,g_2,\ldots,g_k$ 为多项式环 $\mathbb{Z}_q[x]$ 上两两互素的 k 个理想, 多项式环 $\mathbb{Z}_q[x]/(g_1g_2\ldots g_k)$ 上的多项式 f 在 CRT 同构下被映射到 $\mathbb{Z}_q[x]/(g_i)$ 上的 k 个多项式 $(f \bmod g_1, f \bmod g_2,\ldots, f \bmod g_k)$.

基-N FFT Trick 的 CRT 同构形式为:

$$\mathbb{Z}_q[x]/(x^{Nm}-\zeta^N)\cong\prod_{i=0}^{N-1}\mathbb{Z}_q[x]/(x^m-\rho^i\zeta),$$

其中, ρ 为 $\mathbb{Z}_q$ 中的 N 次单位根, ζ 是 $\mathbb{Z}_q$ 中的可逆元素. 当 $\zeta^N=1$ 时得到的 NTT 称为循环卷积 NTT, 当 $\zeta^N=-1$ 时称为负折叠卷积 NTT. 当 $m=1$ 时, NTT 称为完整的 NTT, 否则称为不完整的 NTT.

4 底层代数结构的分析

本节着眼于格密码方案的底层代数结构, 对比此前基于格的签名方案使用的分圆环和本文使用的素阶数域, 梳理了分圆环存在的安全隐患和当前针对分圆环的攻击策略, 并介绍了素阶数域在抵抗相关攻击方面的优势.

4.1 分圆环存在的安全隐患

$\phi_m(x)=\prod_{j\in\mathbb{Z}_m^\times}(x-\xi_m^j)$ 是第 m 个分圆多项式, 其中 $\xi_m=\exp(2\pi i/m)$ 是 m 次单位根. 分圆多项式是 $\mathbb{Z}[x]$ 上的首一、整系数的不可约多项式, 第 m 个分圆多项式对应的分圆环为 $\mathbb{Z}[x]/(\phi_m(x))$, 当 $m=2^{k+1}$ 时, $\phi_m(x)=x^{2^k}+1$.

分圆环是当前大多数格密码方案使用的底层代数结构, 例如在 NIST 后量子密码征集项目的第 3 轮提交方案中, 基于模格的 CRYSTALS-Kyber[9]和 CRYSTALS-Dilithium[11]方案分别使用了分圆环 $\mathbb{Z}_{3329}[x]/(x^{256}+1)$ 和 $\mathbb{Z}_{8380417}[x]/(x^{256}+1)$, Saber、Falcon 等方案也使用了对应 2^k 次分圆多项式的 2 次幂分圆环. 然而由于分圆环具有丰富的代数结构, 因此基于分圆环的密码方案易受到由此衍生出的针对性攻击, 例如, 基于 2 次幂分圆环的 Smart-Vercauteren 全同态加密方案[13]已被多项式时间的量子攻击和亚指数时间的经典攻击[18,19]攻破.

分圆环的 3 个特性导致了它容易受到针对性攻击: 一是分圆环具有大量子域; 二是分圆环具有大量环同态; 三是分圆环具有小 Galois 群.

4.1.1 Campbell-Grove-Shepherd 攻击

Campbell-Grove-Shepherd 攻击指 Campbell 等人基于理想格密码系统"Soliloquy"提出的一种多项式时间内的量子密钥恢复攻击[14]. "Soliloquy"系统的密钥恢复问题和 Smart-Vercauteren 全同态加密方案的密钥恢复问题类似, 并且两者均使用分圆环 $\mathbb{Z}[x]/(\phi(x))$ 作为底层代数结构. 同样的密钥恢复问题也常见于同态加密方案和多线性映射系统中.

在这些方案中, 公钥能够生成多项式环上的主理想, 私钥则是这个主理想的一个短生成元. Campbell-Grove-Shepherd 攻击可以分为两个阶段: 第 1 阶段是寻找主理想上的一些表示成环元素幂次的乘积形式的生成元, 即使不考虑量子计算机的应用, 通用算法也可以在亚指数时间内完成这一步骤. 第 2 阶段则可以视为"log-unit 格"上的解码问题, 其主要目的是通过对已知生成元的约减找到适合解密的短生成元, 从而完成密钥恢复攻击. 单位 (unit) 指的是交换环中具有乘法逆元的非零元素. 在通常情况下, 解码问题需要花费指数级的时间, 然而在分圆环的情况

下, 存在有效的算法能够得到 log-unit 格的一个包含各个“分圆单位 (cyclotomic units)”的对数的短基, 使得此时的解码问题变得容易. 文献 [35] 中详细分析了这一过程.

4.1.2 Biasse-Song 攻击

Biasse 等人[36]认为 Campbell-Grove-Shepherd 攻击第 1 阶段的实际效率低于 Campbell 等人[14]给出的结果, 总体算法效率无法达到多项式级别. 随后, 他们基于 Campbell-Grove-Shepherd 攻击的思想和文献 [37] 中计算主理想生成元的方法, 提出了一种多项式时间的量子算法, 能够求解任意次数数域上的主理想问题[15], 称之为 Biasse-Song 攻击.

4.1.3 Cramer-Ducas-Wesolowski 攻击

Cramer-Ducas-Wesolowski 攻击指 Cramer 等人[16]提出的针对理想格困难问题的多项式时间量子攻击算法. 对于一般的多项式环 $\mathbb{Z}[x]/(\phi(x))$, $\phi(x)$ 是 $\mathbb{Z}[x]$ 上的首一不可约多项式, 给定多项式环的任意非零理想, Cramer-Ducas-Wesolowski 攻击能够找到一个较短的向量, 其长度不超过最短向量的 $2^{\tilde{O}\left(\sqrt{\deg\phi(x)}\right)}$ 倍, 这一效果显著超越了以往攻击能够达到的 $2^{\tilde{O}(\deg\phi(x))}$ 倍.

而在分圆环上, Cramer-Ducas-Wesolowski 攻击可以利用分圆环特有的代数结构, 得到给定非零理想附近的主理想, 再调用 Campbell-Grove-Shepherd 攻击得到这一主理想的最短生成元, 取得更好的攻击效果.

4.1.4 S-unit 攻击

unit 攻击指利用环中的 unit 来寻找理想中的短生成元的一类攻击方法, 它的常见思路是通过约减 unit 格的模数来缩短生成元.S-unit 攻击则是 unit 攻击在 S-unit 格上的一般化. 详细的描述参见文献 [17].

事实上, unit 攻击和 S-unit 攻击包含了大量类似思想的攻击策略, 且相关研究表明, 基于分圆环的格密码方案更容易受到 S-unit 攻击. 例如, 上文提到的 Campbell-Grove-Shepherd 攻击[14]在 unit 攻击中应用了分圆结构的特性, Biasse-Song[15]攻击则与计算 unit 群和 S-unit 群有关.

Laarhoven 等人[38]提出了一种在 ℓ_2 范数下约减任意格上 unit 攻击短向量模数的算法, Pellet-Mary 等人[39]将这种方法应用于 S-unit 攻击; 随后, Bernstein 等人[17]进一步改进了 S-unit 攻击, 指出 S-unit 格上存在大量更短的向量, 并且能够更有效地进行约减. 另一方面, S-unit 攻击的核心步骤之一是计算数域中元素的范数, 因此计算范数的耗时会对攻击结果造成影响. Bernstein 给出了一种 Abelian 域上元素范数的快速计算方法[40], 并且利用 2 次幂分圆域上的自同构给出了针对性优化, 使得 2 次幂分圆域上元素范数的计算速度比一般数域快 10 万倍. 以上工作极大地提高了 2 次幂分圆域上 S-unit 攻击的效果, 使得基于 2 次幂分圆域的密码方案安全性受到威胁.

4.1.5 针对分圆环的对偶攻击

对偶攻击是针对 LWE 问题的常见攻击方式, 其复杂度通常也是衡量基于 LWE 问题的密码方案安全性的重要指标. 近年来的相关研究发现, 利用分圆环结构的对偶攻击能够表现出更好的性能.

对偶攻击的基本思想是将判定性 LWE 问题规约为 SIS 问题进行求解. Duc 等人[41]使用快速傅里叶变换提升了对偶攻击的效率, 并且指出在秘密值系数较小的情况下, 对偶攻击可以被进一步优化, 详细描述见文献 [42–44]. Guo 等人[45]针对分圆环的特性改进了对偶攻击, 随后 MATZOV[46]进一步给出了一种优化的对偶攻击, 并且声称该攻击的效果优于 Guo 等人给出的攻击.

4.1.6 子域攻击

子域攻击的基本思想是将原始域上的格问题规约到子域上的一个相对简单的格问题上, 然后反过来用子域上格问题的解构造原问题的解.

显然, 底层代数结构具有大量子域的密码系统更容易遭受子域攻击, 例如文献 [47] 中提出的拟多项式时间子域攻击, 它利用了多重二次环中存在大量子域的特点, 在不需要量子计算机辅助的情况下成功寻找到理想的短生成元. 实际应用中, 大多数子域攻击依赖原始域 $\mathbb{Q}[x]/(\phi(x))$ 的大次数子域, 而 2 次幂分圆域中有 $\phi(x)=x^{2^k}+1$, 此时 $\deg\phi(x)=2^k$ 具有大量远大于 1 且小于 $\deg\phi(x)$ 的因子, 进而 2 次幂分圆域中存在大量大次数子域, 容易遭受子域攻击.

4.1.7 自同构攻击和环同态攻击

文献 [19] 指出, 敌手有可能利用方案中的自同构构建 unit, 进行相应的攻击. 对于 2 次幂分圆多项式 $x^{2^k}+1$, $\mathrm{Q}(\zeta)$ 是 $x^{2^k}+1$ 对应的最小域, 其中 ζ 是 2^{k+1} 次单位根. 对应的 Galois 群同构于阶为 2^k 的乘法群 $\mathbb{Z}_{2^{k+1}}^{\times}$, 此时多项式对应的 Galois 群定义为包含多项式所有复数根的最小域的自同构群. 因此 $\mathbb{Q}(\zeta)$ 具有 2^k 个自同构: $\zeta \mapsto \zeta^i, i \in \mathbb{Z}_{2^{k+1}}^{\times}$, 大量自同构的存在可能导致安全隐患.

在格密码方案中, 分圆环的选择往往会考虑到利用 NTT 算法进行高效多项式乘法的便利性. 为了进行 NTT 运算, 通常能够利用中国剩余定理构建出分圆环 $\mathbb{Z}_q[x]/(x^{2^k}+1)$ 到一些较小非零环的同态映射, 这体现了分圆环具有大量的环同态. 文献 [48,49] 给出了利用环同态的性质攻击某些密码方案的例子, 尽管目前还没有针对分圆环的环同态攻击, 但不能排除分圆环中大量的环同态带来的潜在威胁.

4.2 素阶数域的安全优势

本文使用的底层代数结构是 Bernstein 等人在文献 [18,19] 中建议使用的素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$, 其中 n 和 q 为素数, 且 x^n-x-1 是 $\mathbb{Z}_q[x]$ 中的不可约多项式.

虽然 Campbell-Groves-Shepherd 攻击、Biasse-Song 攻击、Cramer-Ducas-Wesolowski 攻击、S-unit 攻击和对偶攻击在分圆环上取得了相比于普通情况的显著优势, 然而这些优势依赖于分圆环的特殊结构, 因此它们在素阶数域上无法得到同样的结果. 除此之外, 素阶数域在抵抗子域攻击、自同构攻击与环同态攻击上也体现了相应的安全优势.

4.2.1 抵抗子域攻击

由于素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$ 中 n 为素数, 此时原始域 $\mathbb{Q}[x]/(x^n-x-1)$ 只有两个子域: $\mathbb{Q}[x]$ 和它本身. 由于原始域的子域数量极少, 因此素阶数域可以抵抗文献 [47] 提出的基于大量子域的拟多项式时间子域攻击, 并且敌手无法将格上困难问题转化为合适子域上更简单的问题, 也在客观上抵御了子域攻击.

4.2.2 抵抗自同构攻击

素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$ 中, 不可约多项式 x^n-x-1 对应的 Galois 群同构于阶数为 $n!$ 的置换群的 S_n, 远大于分圆多项式对应的阶数为 n 的 Galois 群. 文献 [19] 指出, 当 Galois 群足够大时, 多项式 x^n-x-1 在任何合理次数的域上至多有少数根, 这一性质体现了素阶数域能够抵抗已知的自同构攻击.

4.2.3 抵抗环同态攻击

素阶数域满足 x^n-x-1 是 $\mathbb{Z}_q[x]$ 中的不可约多项式, 无法使用中国剩余定理构建环同态, 因此文献 [48,49] 提出的环同态攻击无法应用于素阶数域. 虽然在一些密码方案 (如全同态加密) 中, 模转换 (modulus switching) 技术的使用能够弱化模数 q 的影响, 然而这一过程会引入额外的噪音, 使得环同态攻击的攻击难度和算法效率受到极大的影响.

综合以上两部分介绍, 容易看出分圆环丰富的代数结构使得它容易遭到针对性攻击, 存在不可忽视的安全隐患. 相反素阶数域上可供敌手使用的代数结构 (如子域、自同构和环同态) 很少, 能够以明显的优势抵抗子域攻击、自同构攻击等在分圆环上具有显著优势的攻击策略, 保障了密码方案的安全性.

5 本文方案

本节介绍本文提出的基于素阶数域和 CRYSTALS-Dilithium 数字签名算法[11]构造的后量子数字签名方案, 记为 Dilithium-Prime.

5.1 Dilithium-Prime 算法构造

Dilithium-Prime 算法包含密钥生成算法 Dilithium-Prime.*KeyGen*, 签名算法 Dilithium-Prime.*Sign* 和验证算法 Dilithium-Prime.*Verify* 这 3 个部分, 如算法 1–算法 3 所示. 其中 $\mathcal{R}_q=\mathbb{Z}_q[x]/(x^n-x-1)$ 为算法使用的素阶数域, 其中 n 和 q 均为素数, 且 x^n-x-1 是 $\mathbb{Z}_q$ 上的不可约多项式.

算法 2 中给出了签名算法的确定性版本和随机性版本, 其主要区别在于确定性算法中, 生成多项式向量 y 时

使用的随机种子 ρ' 由哈希函数计算得到, 相反在随机性算法中, ρ' 由 512 位的随机采样生成. 注意这并不意味着确定性算法无法保证种子 ρ' 的随机性, 在确定性算法中使用哈希函数计算 ρ' 时, 其随机性来源于输入的私钥 sk 和签名消息 M 共同计算得到的哈希值. 此时, 由于算法的随机性来源于消息, 因此相同消息进行多次签名时, 签名的拒绝次数始终相同, 当签名者不希望泄露正在进行签名的消息 M 或考虑到文献 [50,51] 中利用算法确定性的侧信道攻击时, 需要考虑使用随机性签名算法.

算法 1. 密钥生成算法 Dilithium-Prime.*KeyGen*.

输入: 安全参数 1^{λ};
输出: 签名公私钥对 (pk, sk).

1. $\zeta \leftarrow \{0,1\}^{256}$
2. $(\rho,\rho',K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} := H(\zeta)$
3. $\mathbf{A} \in \mathcal{R}_q^{k\times l} := \mathrm{ExpandA}(\rho)$
4. $(\boldsymbol{s}_1, \boldsymbol{s}_2) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k := \mathrm{ExpandS}(\rho')$
5. $\boldsymbol{t} := \mathbf{A}\boldsymbol{s}_1 + \boldsymbol{s}_2$
6. $(\boldsymbol{t}_1, \boldsymbol{t}_0) := \mathrm{Power2Round}_q(\boldsymbol{t}, d)$
7. $tr \in \{0,1\}^{256} := H(\rho||\boldsymbol{t}_1)$
8. **RETURN** $(pk := (\rho, \boldsymbol{t}_1), sk := (\rho, K, tr, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_0))$

算法 2. 签名算法 Dilithium-Prime.*Sign*.

输入: 私钥 sk, 消息 M;
输出: 签名 σ.

1. $\mathbf{A} \in \mathcal{R}_q^{k\times l} := \mathrm{ExpandA}(\rho)$
2. $\mu \in \{0,1\}^{512} := H(tr||M)$
3. $\kappa := 0,\ (\boldsymbol{z}, \boldsymbol{h}) := \perp$
4. $\rho' \in \{0,1\}^{512} := H(K||\mu)$ //算法的随机性版本中 $\rho' \leftarrow \{0,1\}^{512}$
5. **WHILE** $(\boldsymbol{z}, \boldsymbol{h}) := \perp$ **DO**
6. $\quad \boldsymbol{y} \in \tilde{\mathcal{S}}_{\gamma_1}^l := \mathrm{ExpandMask}(\rho', \kappa)$
7. $\quad \boldsymbol{w} := \mathbf{A}\boldsymbol{y}$
8. $\quad \boldsymbol{w}_1 := \mathrm{HighBits}_q(\boldsymbol{w}, 2\gamma_2)$
9. $\quad \tilde{c} \in \{0,1\}^{256} := H(\mu||\boldsymbol{w}_1)$
10. $\quad c \in \mathcal{B}_\tau := \mathrm{SampleInBall}(\tilde{c})$
11. $\quad \boldsymbol{z} := \boldsymbol{y} + c\boldsymbol{s}_1$
12. $\quad \boldsymbol{r}_0 := \mathrm{LowBits}_q(\boldsymbol{w} - c\boldsymbol{s}_2, 2\gamma_2)$
13. $\quad$ **IF** $\|\boldsymbol{z}\|_\infty \geqslant \gamma_1 - \beta$ **or** $\|\boldsymbol{r}_0\|_\infty \geqslant \gamma_2 - \beta$
14. $\quad\quad (\boldsymbol{z}, \boldsymbol{h}) := \perp$
15. $\quad$ **ELSE**
16. $\quad\quad \boldsymbol{h} := \mathrm{MakeHint}_q(-c\boldsymbol{t}_0, \boldsymbol{w} - c\boldsymbol{s}_2 + c\boldsymbol{t}_0, 2\gamma_2)$
17. $\quad\quad$ **IF** $\|c\boldsymbol{t}_0\|_\infty \geqslant \gamma_2$ **or** the number of 1 in $\boldsymbol{h}$ is greater than ω
18. $\quad\quad\quad (\boldsymbol{z}, \boldsymbol{h}) := \perp$
19. $\quad \kappa := \kappa + l$

20. **RETURN** $\sigma = (\tilde{c}, \mathbf{z}, \boldsymbol{h})$

算法 3. 验证算法 Dilithium-Prime.*Verify*.

输入: 公钥 pk, 消息 M, 签名 $\sigma = (\tilde{c}, \mathbf{z}, \boldsymbol{h})$;
输出: 验证通过 (输出 1) 或验证不通过 (输出 0).

1. $\mathbf{A} \in \mathcal{R}_q^{k\times l} := \text{ExpandA}(\rho)$
2. $\mu \in \{0,1\}^{512} := H(H(\rho \| \boldsymbol{t}_1) \| M)$
3. $c := \text{SampleInBall}(\tilde{c})$
4. $\boldsymbol{w}_1' := \text{UseHint}_q(\boldsymbol{h}, \mathbf{A}\mathbf{z} - c\boldsymbol{t}_1 \cdot 2^d, 2\gamma_2)$
5. **IF** $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ **and** $\tilde{c} = H(\mu \| \boldsymbol{w}_1')$ **and** the number of 1 in $\boldsymbol{h}$ is $\leqslant \omega$
6. **RETURN** 1
7. **ELSE**
8. **RETURN** 0

为了压缩公钥尺寸, Dilithium-Prime 算法使用种子 ρ 来代替矩阵 $\mathbf{A}$ 作为公钥, 这使得 Dilithium-Prime.*Sign* 和 Dilithium-Prime.*Verify* 算法需要重复使用 ρ 生成 $\mathbf{A}$ 这一过程. 考虑到矩阵 $\mathbf{A}$ 在算法中仅用于多项式乘法运算, 因此若不考虑存储空间, 算法也可以预先计算矩阵 $\mathbf{A}$ 在 NTT 域上的对应形式 $\widehat{\mathbf{A}}$, 使用 $(\widehat{\mathbf{A}}, t_1)$ 作为公钥, 节省了重复调用 ExpandA 和 NTT 算法的时间, 提升签名算法和验证算法的效率. 由于素阶数域上的 NTT 算法效率相对较低, 且无法直接生成 NTT 域上的 $\widehat{\mathbf{A}}$, 因此相比于 CRYSTALS-Dilihium 方案, 这一策略对 Dilithium-Prime 算法效率的提升更为明显. 同理, 也可以预先计算并存储私钥中 $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_0$ 的 NTT 形式, 节省后续算法中计算 NTT 所需的时间.

而在另一种极端情况下, Dilithium-Prime 算法的私钥可以被压缩至 32 字节的随机种子 $\zeta \leftarrow \{0,1\}^{256}$, 此时签名算法需要使用 ζ 重新生成 $\rho, K, tr, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_0$, 在节省存储空间的同时显著降低算法的效率.

算法 1–算法 3 中使用了 4 种基于哈希的采样算法, 其中 ExpandA 通过哈希将 256 位的随机种子 ρ 映射到矩阵 $\mathbf{A} \in \mathcal{R}_q^{k\times l}$, 注意该算法得到的矩阵 $\mathbf{A}$ 在正常域而并非 NTT 域中, 其原因在于本方案在素阶数域上的 NTT 计算不是双射 (详细内容见本文第 6.2 节), 无法直接在 NTT 域上生成乘法计算所需的 $\widehat{\mathbf{A}}$; ExpandS 则使用 512 位的种子 ρ' 生成 $(\boldsymbol{s}_1, \boldsymbol{s}_2) \in \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$; ExpandMask 使用随机生成或根据 sk 和 M 计算得到的种子 ρ' 生成 $\boldsymbol{y} \in \tilde{\mathcal{S}}_{\gamma_1}^l$, 由于签名可能会重复多次拒绝采样的过程, 因此 ExpandMask 使用计数器 κ 来保证每次采样得到的随机多项式向量不同. 定义 $\mathcal{B}_\tau$ 为 $\mathcal{R}$ 的子集, $\mathcal{B}_\tau$ 中的多项式有且仅有 τ 个系数为 1 或−1, 其余系数均为 0, 可得 $|\mathcal{B}_\tau| = 2^\tau \cdot \binom{n}{\tau}$, SampleInBall 通过两步哈希生成 $\mathcal{B}_\tau$ 中的元素 c, 详细步骤由算法 4 给出.

算法 4. $\mathcal{B}_\tau$ 采样算法 SampleInBall.

输入: 种子 $\tilde{c} \in \{0,1\}^{256}$;
输出: $c \in \mathcal{B}_\tau$.

1. Initialize $c \in \{0,1\}^n := \{0\}^n$
2. **FOR** $i = n - \tau$ **to** $n - 1$
3. $j \leftarrow \{0, 1, \ldots, i\}$
4. $s \leftarrow \{0,1\}$
5. $c_i := c_j$
6. $c_j := (-1)^s$

7. **RETURN** c

Dilithium-Prime 算法使用了一系列算法分离并提取 $\mathbb{Z}_q$ 中元素的高位和低位, 来进一步压缩公钥尺寸. 算法 5–算法 10 给出了详细的步骤, 除去分解高低位之外, 对于给定任意 $r\in\mathbb{Z}_q$ 和一个范数较小的 $z\in\mathbb{Z}_q$, MakeHint_q 算法 (算法 9) 计算出一个比特大小的"线索" h, 使得签名验证者只需使用 r 和 h 便可计算 $r+z$ 的高位, 计算 $r+z$ 的高位的过程由 UseHint_q (算法 10) 描述.

算法 5. 高低位分解算法 Power2Round_q.

输入: $r\in\mathbb{Z}_q$, $0<d\leqslant\lfloor\log_2 r\rceil$, q;
输出: r 的低 d 位 r_0 和对应的高位 r_1.

1. $r:=r \bmod^+ q$
2. $r_0:=r \bmod^\pm 2^d$
3. $r_1:=(r-r_0)/2^d$
4. **RETURN** (r_1,r_0)

算法 6. 高低位分解算法 Decompose_q.

输入: $r\in\mathbb{Z}_q$, $0<\alpha<\dfrac{q}{2}$, q;
输出: r 的高位和低位 (r_1,r_0).

1. $r:=r \bmod^+ q$
2. $r_0:=r \bmod^\pm \alpha$
3. **IF** $r-r_0=q-1$
4. $\quad r_1:=0$
5. $\quad r_0:=r_0-1$
6. **ELSE**
7. $\quad r_1:=(r-r_0)/\alpha$
8. **RETURN** (r_1,r_0)

算法 5 和算法 6 描述了两种不同的高低位分解方法[11]. Power2Round_q 算法根据 r 的二进制形式, 直接在其低 d 位处进行分解, 得到的高位 r_1 和低位 r_0 可被分别视为商和余数. 注意到此时高位 r_1 满足 $0\leqslant r_1\leqslant\lfloor q/2^d\rceil$, 在通常情况下, 由于 $r_1\cdot 2^d$ 和 $r_1'\cdot 2^d$ 差值较大, 因此高位分别为 r_1 和 r_1' 的两个数字 r 和 r' 在模 q 意义下具有较大的差值, 向 r 添加一个范数较小的数字导致高位的变化至多为 1. 然而, 考虑边界条件 $r_1=\lfloor q/2^d\rceil$ 和 $r_1'=0,\lfloor q/2^d\rceil\cdot 2^d$ 和 0 在模 q 时可能差值很小, 此时若向 r 添加一个数字, 高位的变化值会超过 1.

为了将"线索" h 的大小限制在 1 bit 内, Decompose_q 算法选择了 $q-1$ 的一个因子 α, 对应 r 的高低位计算为 $r=r_1\cdot\alpha+r_0$, 此时 $r_1\in\{0,1,\ldots,(q-1)/\alpha\}$, 由于 $q-1$ 和 0 在模 q 意义下只相差 1, 因此当 $r_1=(q-1)/\alpha$ 时, $r=q+r_0-1$, 将 r_1 置为 0, 对应的余项大小减去 1, 就可以避免高位的变化值过大. 在实现 Decompose_q 算法的过程中, 考虑到侧信道攻击和算法效率, 本文设计了一种高效的无分支算法, 具体内容将在第 7 节中详细介绍. 在 Decompose_q 的基础上定义 MakeHint_q 算法和 UseHint_q 算法, 即可生成 h 并恢复高位.

算法 7. 取高位算法 HighBits_q.

输入: $r\in\mathbb{Z}_q$, $0<\alpha<\dfrac{q}{2}$, q;
输出: r 的高位 r_1.

1. $(r_1, r_0) := \text{Decompose}_q(r, \alpha)$
2. **RETURN** r_1

算法 8. 取低位算法 LowBits_q.

输入: $r \in \mathbb{Z}_q$, $0 < \alpha < \dfrac{q}{2}$, q;
输出: r 的低位 r_0.

1. $(r_1, r_0) := \text{Decompose}_q(r, \alpha)$
2. **RETURN** r_0

算法 9. Hint 生成算法 MakeHint_q.

输入: $r \in \mathbb{Z}_q$, $0 < \alpha < \dfrac{q}{2}$, $|z| \leqslant \dfrac{\alpha}{2}$, q;
输出: $h \in \{0,1\}$.

1. $r_1 := \text{HighBits}_q(r, \alpha)$
2. $v_1 := \text{HighBits}_q(r+z, \alpha)$
3. **IF** $r_1 \neq v_1$
4. **RETURN** 1
5. **ELSE**
6. **RETURN** 0

算法 10. Hint 还原算法 UseHint_q.

输入: $r \in \mathbb{Z}_q$, $0 < \alpha < \dfrac{q}{2}$, $h \in \{0,1\}$, q;
输出: $r+z$ 的高位 r_1.

1. $m := (q-1)/\alpha$
2. $(r_1, r_0) := \text{Decompose}_q(r, \alpha)$
3. **IF** $h = 1$ **and** $r_0 > 0$
4. **RETURN** $(r_1 + 1) \bmod^+ m$
5. **ELSEIF** $h = 1$ **and** $r_0 \leqslant 0$
6. **RETURN** $(r_1 - 1) \bmod^+ m$
7. **ELSE**
8. **RETURN** r_1

上述算法的输入定义为 $\mathbb{Z}_q$ 中的元素, 而在 Dilithium-Prime 方案中, 算法的输入为 $\mathcal{R}_q$ 上的多项式向量, 此时运算相当于将多项式向量中每个多项式的每项系数分别输入算法 5–算法 10 进行计算. 在多项式向量运算中, HighBits_q, LowBits_q, MakeHint_q 和 UseHint_q 算法的相关性质详见第 5.2 节的描述.

5.2 正确性分析

类似于文献 [11], Dilithium-Prime 的正确性定理基于以下 3 条引理, 引理的证明详见文献 [11] 的附录 A.

引理 1. 设正数 α 和 q 满足 $q > 2\alpha, q \equiv 1 \pmod{\alpha}$, 且 α 为偶数; 给定 $\boldsymbol{r}, \boldsymbol{z}$ 为 $\mathcal{R}_q$ 上的多项式向量, 其中 $\|\boldsymbol{z}\|_\infty \leqslant \alpha/2$, 此时算法 HighBits_q, MakeHint_q 和 UseHint_q 具有以下性质.

(1) 给定 $\boldsymbol{h} = \text{MakeHint}_q(\boldsymbol{z}, \boldsymbol{r}, \alpha)$, 则 $\boldsymbol{r} + \boldsymbol{z}$ 的高位可由 $\text{UseHint}_q(\boldsymbol{h}, \boldsymbol{r}, \alpha) = \text{HighBits}_q(\boldsymbol{r} + \boldsymbol{z}, \alpha)$ 恢复.

(2) 设 $\boldsymbol{v}_1=\mathrm{UseHint}_q(\boldsymbol{h},\boldsymbol{r},\alpha)$, 则 $\|\boldsymbol{r}-\boldsymbol{v}_1\cdot\alpha\|_\infty\leqslant\alpha+1$, 且若 $\boldsymbol{h}$ 中某系数为 0, 则 $\boldsymbol{r}-\boldsymbol{v}_1\cdot\alpha$ 中对应位置系数的无穷范数不超过 $\alpha/2$.

(3) 给定任意的 $\boldsymbol{h}$, $\boldsymbol{h}'$, 若满足 $\mathrm{UseHint}_q(\boldsymbol{h},\boldsymbol{r},\alpha)=\mathrm{UseHint}_q(\boldsymbol{h}',\boldsymbol{r},\alpha)$, 则 $\boldsymbol{h}=\boldsymbol{h}'$.

引理 2. 若同时满足 $\|\mathrm{LowBits}_q(\boldsymbol{r},\alpha)\|_\infty<\alpha/2-\beta$ 且 $\|\boldsymbol{s}\|_\infty\leqslant\beta$, 则下式成立:

$$\mathrm{HighBits}_q(\boldsymbol{r},\alpha)=\mathrm{HighBits}_q(\boldsymbol{r}+\boldsymbol{s},\alpha).$$

引理 3. 定义 $(\boldsymbol{r}_1,\boldsymbol{r}_0):=\mathrm{Decompose}_q(\boldsymbol{r},\alpha)$ 与 $(\boldsymbol{w}_1,\boldsymbol{w}_0):=\mathrm{Decompose}_q(\boldsymbol{r}+\boldsymbol{s},\alpha)$ 且满足 $\|\boldsymbol{s}\|_\infty\leqslant\beta$, 则以下条件等价:

$$\|\boldsymbol{s}+\boldsymbol{r}_0\|_\infty\leqslant\frac{\alpha}{2}-\beta\iff\boldsymbol{w}_1=\boldsymbol{r}_1\wedge\|\boldsymbol{w}_0\|_\infty\leqslant\frac{\alpha}{2}-\beta.$$

定理 1. Dilithium-Prime 方案的正确性. 设 Dilithium-Prime 签名算法输出的签名 $\sigma=(\tilde{c},\boldsymbol{z},\boldsymbol{h})$, 验证算法接收该签名后计算得到的结果满足 $\tilde{c}=H(\mu\|\boldsymbol{w}_1')$.

证明: 根据引理 1, 当 $c\boldsymbol{t}_0$ 满足 $\|c\boldsymbol{t}_0\|_\infty<\gamma_2$ 时, 以下公式成立:

$$\mathrm{UseHint}_q(\boldsymbol{h},\boldsymbol{w}-c\boldsymbol{s}_2+c\boldsymbol{t}_0,2\gamma_2)=\mathrm{HighBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2).$$

由于算法中 $\boldsymbol{w}=\mathbf{A}\boldsymbol{y},\boldsymbol{z}=\boldsymbol{y}+c\boldsymbol{s}_1$ 且 $\boldsymbol{t}=\mathbf{A}\boldsymbol{s}_1+\boldsymbol{s}_2$, 因此:

$$\boldsymbol{w}-c\boldsymbol{s}_2=\mathbf{A}\boldsymbol{y}-c\boldsymbol{s}_2=\mathbf{A}(\boldsymbol{z}-c\boldsymbol{s}_1)-c\boldsymbol{s}_2=\mathbf{A}\boldsymbol{z}-c\boldsymbol{t}.$$

由于 $\boldsymbol{t}=\boldsymbol{t}_1\cdot2^d+\boldsymbol{t}_0$, 上式可以进一步得到:

$$\boldsymbol{w}-c\boldsymbol{s}_2+c\boldsymbol{t}_0=\mathbf{A}\boldsymbol{z}-c\boldsymbol{t}_1\cdot2^d.$$

结合引理 1 给出的公式, 验证算法中 $\mathrm{UseHint}_q$ 计算得到:

$$\mathrm{UseHint}_q\left(\boldsymbol{h},\mathbf{A}\boldsymbol{z}-c\boldsymbol{t}_1\cdot2^d,2\gamma_2\right)=\mathrm{HighBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2).$$

另一方面, 签名算法中拒绝采样的过程保证了 $\|c\boldsymbol{s}_2\|_\infty\leqslant\beta$ 和 $\mathrm{LowBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2)<\gamma_2-\beta$, 根据引理 2 可得:

$$\begin{aligned}\boldsymbol{w}_1'&=\mathrm{UseHint}_q\left(\boldsymbol{h},\mathbf{A}\boldsymbol{z}-c\boldsymbol{t}_1\cdot2^d,2\gamma_2\right)\\&=\mathrm{HighBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2)\\&=\mathrm{HighBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2+c\boldsymbol{s}_2,2\gamma_2)\\&=\mathrm{HighBits}_q(\boldsymbol{w},2\gamma_2)\\&=\boldsymbol{w}_1.\end{aligned}$$

因此验证算法中哈希函数的输入与签名算法相同, 自然满足 $\tilde{c}=H(\mu\|\boldsymbol{w}_1')$.

5.3 采样成功率分析

类似于文献 [11], 在签名者进行签名的过程中, 算法可能重复多次拒绝采样的过程, 定理 2 给出了单次拒绝采样成功的概率, 基于此概率的期望循环次数见第 5.5 节给出的参数集.

定理 2. 拒绝采样的成功率. 在 Dilithium-Prime 签名算法中, 执行一次拒绝采样成功的概率约为 $\mathrm{e}^{-n\cdot\beta(l/\gamma_1+k/\gamma_2)}$.

证明: 已知拒绝采样成功需要同时满足 $\|\boldsymbol{z}\|_\infty<\gamma_1-\beta$, $\|\boldsymbol{r}_0\|_\infty<\gamma_2-\beta$, $\|c\boldsymbol{t}_0\|_\infty<\gamma_2$ 以及 $\boldsymbol{h}$ 中值为 1 的系数数量不超过 ω 这 4 个条件.

其中, 计算后两个条件不满足的概率十分困难, 可以通过参数设置将其概率控制在 1%–2% 之间, 此时, 拒绝采样成功的概率可以近似看作同时满足 $\|\boldsymbol{z}\|_\infty<\gamma_1-\beta$ 和 $\|\boldsymbol{r}_0\|_\infty<\gamma_2-\beta$ 的概率.

若条件 $\|\boldsymbol{z}\|_\infty<\gamma_1-\beta$ 满足, 则 $\boldsymbol{z}$ 中各多项式的系数在区间 $[-\gamma_1+\beta+1,\gamma_1-\beta-1]$ 内, 设 $c\boldsymbol{s}_1$ 中多项式的系数为 μ, 则 $\boldsymbol{y}=\boldsymbol{z}-c\boldsymbol{s}_1$ 中对应的多项式系数应落在 $[-\gamma_1+\beta+1-\mu,\gamma_1-\beta-1-\mu]$ 内, 因此单个系数满足条件的概率为 $\dfrac{2(\gamma_1-\beta)-1}{2\gamma_1-1}$.

$\boldsymbol{y}$ 中系数均满足条件的概率为:

$$\left(\frac{2(\gamma_1-\beta)-1}{2\gamma_1-1}\right)^{n\cdot l}=\left(1-\frac{\beta}{\gamma_1-1/2}\right)^{n\cdot l}\approx\mathrm{e}^{\frac{-n\beta\cdot l}{\gamma_1}}.$$

由于方案选择的 γ_1 远大于 1/2, 因此近似公式成立.

考虑满足 $\|\boldsymbol{r}_0\|_\infty<\gamma_2-\beta$ 的概率时, 不妨将 $\boldsymbol{r}_0=\mathrm{LowBits}_q(\boldsymbol{w}-c\boldsymbol{s}_2,2\gamma_2)$ 视为在区间 $(-\gamma_2,\gamma_2]$ 内均匀分布, 此时满

足条件的概率为:

$$\left(\frac{2(\gamma_2-\beta)-1}{2\gamma_2}\right)^{n\cdot k}\approx \mathrm{e}^{\frac{-n\beta k}{\gamma_2}}.$$

综上, 执行一次拒绝采样成功的概率约为 $\mathrm{e}^{-n\cdot\beta(l/\gamma_1+k/\gamma_2)}$.

5.4 安全性分析

5.4.1 安全规约

本方案基于 MLWE 困难性假设、MSIS 困难性假设、SelfTargetMSIS 困难性假设的强不可伪造性 (strong unforgeability under chosen message attacks, SUF-CMA) 安全规约与文献 [11] 相同, 文献 [31] 详细证明了此类 Fiat-Shamir 签名在 QROM 模型下的安全性. 本文将直接给出 Dilithium-Prime 方案的安全规约结论.

定理 3. Dilithium-Prime 方案的安全性. 假设 H 是量子随机预言机, 概率分布 $\mathcal{D}$ 是 $\mathcal{S}_\eta$ 上的均匀分布. Dilithium-Prime 方案满足 SUF-CMA 安全. 具体而言, 对于任何概率多项式时间敌手 $\mathcal{A}$, 都存在概率多项式时间敌手 $\mathcal{B}$、C 和 $\mathcal{D}$, 使得:

$$\boldsymbol{Adv}^{\text{SUF-CMA}}_{\text{Dilithium}}(\mathcal{A})\leqslant \boldsymbol{Adv}^{\text{MLWE}}_{k,l,\mathcal{D}}(\mathcal{B})+\boldsymbol{Adv}^{\text{SelfTargetMSIS}}_{H,k,l+1,\zeta'}(C)+\boldsymbol{Adv}^{\text{MSIS}}_{k,l,\zeta'}(\mathcal{D})+2^{-254},$$

其中,

$$\zeta=\max\left\{\gamma_1-\beta,2\gamma_2+1+2^{d-1}\cdot\tau\right\},$$

$$\zeta'=\max\{2(\gamma_1-\beta),4\gamma_2+2\}.$$

直观来讲, MLWE 假设保证密钥难以恢复, SelfTargetMSIS 假设保证新签名消息的不可伪造性 (UF-CMA), MSIS 假设保证了强不可伪造性 (SUF-CMA).

5.4.2 具体安全强度

Dilithium-Prime 方案的具体安全强度依赖 MLWE 问题、MSIS 问题和 SelfTargetMSIS 问题的计算复杂性. 本文考虑针对底层困难问题的攻击手段, 将求解 MLWE 和 MSIS 问题的过程转化为求解某个格上的 u-SVP 问题/无穷范数 SVP 问题, 使用 core-SVP 方法论[52]对格基约化算法 BKZ[53]求解 SVP 问题的复杂度进行保守的安全强度估算. 文献 [52] 指出, 目前使用块大小 β 的 BKZ 算法求解 SVP 问题时, 最优的经典算法和量子算法复杂度分别为 $2^{0.292\beta}$ 和 $2^{0.265\beta}$.

(1) MLWE 问题

在现有的求解 LWE 问题的算法中, 需要大量样本的 BKW 算法和复杂度过高的 Arora-Ge 算法在 Dilithium-Prime 方案上表现较差, 因此本文主要考虑原始攻击和对偶攻击这两种最有效的格上攻击手段. 由于这两种攻击手段在求解 RLWE/MLWE 问题时并不具有额外优势, 在分析 MLWE 实例 $\left(\mathbf{A}\in\mathcal{R}_q^{k\times l},\ \boldsymbol{t}=\mathbf{A}\boldsymbol{s}_1+\boldsymbol{s}_2\in\mathcal{R}_q^k\right)$ 的计算复杂性时, 本文将其转换为对应的标准 LWE 实例 $\left(\mathbf{A}'\in\mathbb{Z}_q^{nk\times nl},\ \boldsymbol{t}'\in\mathbb{Z}_q^{nk}\right)$ 进行分析.

1) 原始攻击

原始攻击的基本思想是将 LWE 问题规约为 u-SVP 问题[52,54], 给定 LWE 实例 $(\mathbf{A}',\boldsymbol{t}')\in\mathbb{Z}_q^{nk\times nl}\times\mathbb{Z}_q^{nk}$, 构造 $d=nl+m+1$ 维格 $\mathcal{L}(\mathbf{M})=\{x\in\mathbb{Z}_q^{nl+m+1}|\left(\mathbf{A}'_{[1:m]}\middle|\boldsymbol{I}_m\middle|-\boldsymbol{t}'_{[1:m]}\right)\boldsymbol{x}=\mathbf{0}\mod q\}$ 中, 其中 $\mathbf{0}$ 表示全 0 向量, $m\leqslant nk$, $\mathbf{A}'_{[1:m]}$ 和 $\boldsymbol{t}'_{[1:m]}$ 指对应矩阵/向量的前 m 行 (在原始攻击中使用全部实例有时并非最优, 因此需要尝试每个可能的 m 取值).

$$\mathbf{M}=\begin{pmatrix}\boldsymbol{I}_{nl} & 0 & 0\\ -\mathbf{A}'_{[1:m]} & q\boldsymbol{I}_m & \boldsymbol{t}'_{[1:m]}\\ 0 & 0 & 1\end{pmatrix}\in\mathbb{Z}_q^{(nl+m+1)\times(nl+m+1)}$$

矩阵 $\mathbf{M}$ 的列向量构成了一组格基, $vol(\mathcal{L}(\mathbf{M}))=q^m$, 显然 $\mathcal{L}(\mathbf{M})$ 上的唯一最短向量 $\boldsymbol{v}=\left(\boldsymbol{s}^{\mathrm{T}}\middle|\boldsymbol{e}^{\mathrm{T}}\middle|1\right)^{\mathrm{T}}$ 即为 LWE 问题的解. BKZ 算法逐渐增大块大小 β, 直到成功求解 u-SVP 问题. 此时 $2^{0.292\beta}$ 和 $2^{0.265\beta}$ 即为在量子和经典情况下的安全强度.

2) 对偶攻击

对偶攻击将判定性 LWE 问题转化为寻找对偶格上的短向量, 然后利用短向量将求解 LWE 问题转化成区分一定参数下的亚高斯分布和 $\mathbb{Z}_q$ 上的均匀分布的问题[52]. 给定 LWE 实例 $(\mathbf{A}', \boldsymbol{t}') \in \mathbb{Z}_q^{nk\times nl} \times \mathbb{Z}_q^{nk}$, 构造 $d = nl + m$ 维格 $\mathcal{L}(\mathbf{M}) = \{(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{nl} : \mathbf{A}'^{\mathrm{T}}_{[1:m]}\boldsymbol{x} = \boldsymbol{y} \mod q\}$ 中, 其中 $m \leqslant nk$. 设 $\mathbf{A}'^{\mathrm{T}}_{[1:m]}$ 中前 nl 列对应的子矩阵为 $\mathbf{A}_1$, $\mathbf{A}'^{\mathrm{T}}_{[1:m]} = (\mathbf{A}_1 || \mathbf{A}_2)$, 则格基矩阵为:

$$\mathbf{M} = \begin{pmatrix} q\boldsymbol{I}_{nl} & -\mathbf{A}_1^{-1}\mathbf{A}_2 & \mathbf{A}_1^{-1} \\ 0 & \boldsymbol{I}_{m-nl} & 0 \\ 0 & 0 & \boldsymbol{I}_{nl} \end{pmatrix} \in \mathbb{Z}_q^{(nl+m)\times(nl+m)}.$$

格上的短向量 $(\boldsymbol{v}, \boldsymbol{w})$ 满足 $\boldsymbol{v}^{\mathrm{T}} \cdot \boldsymbol{t} = \boldsymbol{w}^{\mathrm{T}} \cdot \boldsymbol{s} + \boldsymbol{v}^{\mathrm{T}} \cdot \mathrm{e}$. 可以假设 $(\boldsymbol{v}, \boldsymbol{w})$ 中每个分量的值大致相等, 则 $u = \boldsymbol{w}^{\mathrm{T}} \cdot \boldsymbol{s} + \boldsymbol{v}^{\mathrm{T}} \cdot \mathrm{e}$ 可看作标准差为 $\|(\boldsymbol{v}, \boldsymbol{w})\|\eta$ 的亚高斯分布.

此时存在有效算法以 $\epsilon = 4\mathrm{e}^{-2\pi^2\tau^2}$, 其中 $\tau = \|(\boldsymbol{v}, \boldsymbol{w})\|\eta/q$ 的概率区分 u 和 $\mathbb{Z}_q$ 中均匀分布的元素. 为了使最终的区分优势大于 1/2, 需要找出约 $1/\epsilon^2$ 个短向量, 而每次运行筛法能够产生大约 $2^{0.2075\beta}$ 个短向量, 因此攻击需要重复的次数大约为:

$$R = \max(1, 1/(2^{0.2075\beta}\epsilon^2)).$$

因此 BKZ 算法在经典情形和量子情形的整体消耗分别为 $R \cdot 2^{0.292\beta}$ 和 $R \cdot 2^{0.265\beta}$.

(2) MSIS 问题与 SelfTargetMSIS 问题

与 MLWE 问题类似, $k \times l$ 维 $\mathrm{MSIS}_{k,l,\zeta'}$ 实例 $\mathbf{A}$ 可以转化为 $d = nk \times nl$ 维的 SIS 实例 $\mathbf{A}'$. 本方案使用无穷范数来衡量 SIS 问题解的长度, 相比于二范数而言, BKZ 算法求解无穷范数 SIS 问题的难度将更大[11]. 将基矩阵 $\mathbf{B}$ 输入块大小为 β 的 BKZ 算法得到约化基 $\widehat{\mathbf{B}}$, 对应的 GS 正交基为 $\widehat{B}^* = (\widehat{b}_1^*, \ldots, \widehat{b}_d^*)$. 定义 $l_k = \log_2 \|\widehat{b}_k^*\|, k = 1, \ldots, d$, 这一系列 l_k 值满足前 $i-1$ 个值为 $\log_2 q$, 后 $d-j$ 个值为 0.

虽然 BKZ 算法并不能直接找到满足无穷范数 SIS 问题的解, 但可以以求解一次块大小 β 的 SVP 问题的消耗得到 $\sqrt{4/3}^{\beta}$ 个向量, 这些向量投影到前 $i-1$ 个基向量的正交补后, 其欧式范数约为 $2^{l_i} \approx q$. 由于前 $i-1$ 个基向量是乘以模数的单位向量, 因此投影到正交补上的向量的前 $i-1$ 个分量为 0, 且后 $d-j$ 个基向量对应的分量为 0, 此时中间的 $j-i+1$ 个分量总体的欧式范数约为 2^{l_i}. 可以假设中间的 $j-i+1$ 个分量都服从标准差为 $\sigma = 2^{l_i}/\sqrt{j-i+1}$ 的高斯分布, 且原向量的前 $i-1$ 个分量服从 $\mathbb{Z}_q$ 上的均匀分布. 此时在 $\sqrt{4/3}^{\beta}$ 个向量中找到一个满足 $[0, i-1]$ 和 $[i, j]$ 两个区间内分量的绝对值均小于 ζ' 的向量的概率的倒数乘以调用 SVP 求解器的复杂度即为总的安全强度.

对于 SelfTargetMSIS 问题, 敌手解决该问题相当于攻破哈希函数 H 的安全性或解决对应的 MSIS 问题, 其处理方法与 MSIS 问题一致.

以上分析的具体计算结果见第 5.5 节. 值得注意的是, BKZ 算法在运行过程中将多次调用 SVP 求解器, 具体评估调用次数较为困难, core-SVP 方法的基本思想是不考虑重复调用的次数, 只考虑一次调用的成本, 因此 core-SVP 方法的估计是保守的[52]. 另一方面, 在安全规约的过程中, 困难问题的参数选择同样是保守的[11], 这就保证了本方案的底层计算困难性问题实际上是更困难的.

5.5 参数集

后文表 1 给出 Dilithium-Prime 的推荐参数集, 其中包含了对应 NIST 3 个安全等级的 Dilithium-Prime 参数, 其中环维度 n 均为 251, 方案选取的多项式环为 $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n - x - 1)$, $q = 7681537$ 是环模数, 本文选取的 n 和 q 满足 $x^n - x - 1$ 是 $\mathbb{Z}_q[x]$ 上的不可约多项式. 表 1 中“Repetitions”代表该参数集下拒绝采样次数的期望值. 公钥、私钥和签名尺寸的具体分析详见第 7.2 节. 安全强度的分析见第 5.4 节, 其中 LWE hardness 代表在原始攻击和对偶攻击中最低的安全强度, 表中列出的前后两项数据分别表示在经典和量子情形下计算的安全强度, 它们的数值用比特表示.

6 多项式运算

Dilithium-Prime 方案中包含两类多项式乘法的使用, 一类是 $\mathcal{R}_q$ 上的多项式与 $\mathcal{B}_\tau$ 上多项式的相乘, 其特点是

其中一个多项式的系数限制在 $\{0,\pm1\}$ 上, 例如 $c\boldsymbol{s}_1, c\boldsymbol{s}_2$ 和 $c\boldsymbol{t}_0, c\boldsymbol{t}_1$ 的计算; 另一类则是 $\mathcal{R}_q$ 上两个多项式的相乘, 例如 $\mathbf{A}\boldsymbol{y}, \mathbf{A}\boldsymbol{s}_1$ 的计算.

表 1 Dilithium-Prime 推荐参数集 ($n=251$)

Security level	II	III	V
q [modulus]	7681537	7681537	7681537
d [dropped bits from $\boldsymbol{t}$]	13	13	13
τ [the number of nonzero coefficient in c]	39	49	60
Challenge entropy [$\log\binom{251}{\tau}+\tau$]	192	225	257
γ_1 [coefficient range of $\boldsymbol{y}$]	2^{18}	2^{19}	2^{19}
γ_2 [low-order rounding range]	$(q-1)/32$	$(q-1)/16$	$(q-1)/16$
(k,l) [dimensions of matrix $\mathbf{A}$]	(4, 4)	(6, 5)	(8, 7)
η [secret key range]	2	2	2
β [$2\tau\cdot\eta$]	156	196	240
ω [the maximal number of nonzero coefficient in h]	80	55	75
Repetitions [rejection sample]	3.49	2.96	6.10
pk size	1288	1916	2544
sk size	2504	3605	4801
σ size	2504	3233	4511
LWE hardness	(121, 110)	(162, 146)	(247, 224)
SIS hardness (strong)	(110, 100)	(169, 153)	(243, 220)
SIS hardness (weak)	(120, 109)	(184, 167)	(263, 238)

为了提升多项式乘法的效率, 本方案使用了小多项式乘法和素阶数域 NTT 两种技术分别计算以上两类多项式乘法. 本节将详细介绍 Dilithium-Prime 方案使用的多项式乘法的技术细节.

6.1 小多项式乘法

注意到在 Dilithium-Prime 的签名算法和验证算法中存在一类特殊的多项式乘法, 即 $\mathcal{R}_q$ 上的多项式与 $\mathcal{B}_\tau$ 上多项式 c 的相乘, 其中 $c\in\mathcal{B}_\tau$ 的系数限制在 $\{0,\pm1\}$ 上. 文献 [55] 提到, 在多项式乘法中, 若其中一个多项式的系数足够小, 就可以利用这一特点使用比 NTT 算法更加高效的小多项式乘法进行运算. 本节基于文献 [55] 的思想, 设计了适用于 Dilithium-Prime 的素阶数域小多项式算法.

对于 $\mathcal{R}_q$ 上的多项式乘法 $u=c\cdot a$, 其中 $c=\sum_{i=0}^{n-1}c_ix^i\in\mathcal{B}_\tau, a=\sum_{i=0}^{n-1}a_ix^i\in\mathcal{R}_q$, 设 $u=\sum_{i=0}^{n-1}u_ix^i$, 则 u 的计算遵循以下公式.

- 当 $i=0$ 时: $u_i=c_0\cdot a_0+\sum_{j=1}^{n-1}c_ja_{n-j}\,(\mathrm{mod}^+\,q)$.
- 当 $1\leqslant i\leqslant n-2$ 时: $u_i=\sum_{j=0}^{i}c_ja_{i-j}+\sum_{j=i}^{n-1}c_ja_{n+i-j-1}+\sum_{j=i+1}^{n-1}c_ia_{n+i-j}\,(\mathrm{mod}^+\,q)$.
- 当 $i=n-1$ 时: $u_i=c_{n-1}\cdot a_{n-1}+\sum_{j=0}^{n-1}c_ja_{n-j-1}\,(\mathrm{mod}^+\,q)$.

可以看出, 由于 $c\in\{0,\pm1\}$, 以上公式中的乘法均可被加减法替换, 算法 11 详细描述了这一过程, 称之为基于索引的小多项式乘法.

算法 11. 基于索引的小多项式乘法 SmallPoly.

输入: $c\in\mathcal{B}_\tau, a\in\mathcal{R}_q$;
输出: $u=c\cdot a$.

1. **FOR** $i\in\{0,1,2,\ldots,2n-1\}$
2. $\quad w_i:=0$
3. **FOR** $i\in\{0,1,2,\ldots,n-1\}$

4. **IF** $c_i = 1$
5. **FOR** $j \in \{0,1,2,\ldots,n-1\}$
6. $w_{i+j} := w_{i+j} + a_j$
7. **IF** $c_i = -1$
8. **FOR** $j \in \{0,1,2,\ldots,n-1\}$
9. $w_{i+j} := w_{i+j} - a_j$
10. $u_0 := w_0 + w_n \pmod^+ q$
11. **FOR** $i \in \{1,2,\ldots,n-1\}$
12. $u_i := w_i + w_{i+n-1} + w_{i+n} \pmod^+ q$
13. $u := \sum_{i=0}^{n-1} u_i x^i$
14. **RETURN** u

为了后续设计并行版本小多项式, 算法 12 给出了带有预计算的小多项式乘法. 该算法需要预计算一组数值 $\{v_{1-n}, v_{2-n}, \ldots, v_{-1}, v_0, v_1, \ldots, v_{n-2}, v_{n-1}, v_n\}$, 其中,

- 当 $i = n$ 时: $v_i := a_0$.
- 当 $1 \leqslant i \leqslant n-1$ 时: $v_i := a_i$.
- 当 $i = 0$ 时: $v_i := a_0 + a_{n-1}$.
- 当 $1-n \leqslant i < 0$ 时: $v_i := a_{i+n} + a_{i+n-1}$.

此时, 则 u 的计算遵循以下公式.

- 当 $i = 0$ 时: $u_i := \sum_{j=0}^{n-1} c_j v_{n-j} \pmod^+ q$.
- 当 $1 \leqslant i \leqslant n-1$ 时: $u_i := \sum_{j=0}^{n-1} c_j v_{i-j} \pmod^+ q$.

算法 12. 预计算小多项式乘法 Pre-SmallPoly.

输入: $c \in \mathcal{B}_\tau, a \in \mathcal{R}_q$;
输出: $u = c \cdot a$.

1. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
2. $w_i := 0$
3. $a_n := a_0$
4. **FOR** $i \in \{1,2,\ldots,n\}$
5. $v_i := a_i$
6. $v_{i-n} := a_i + a_{i-1}$
7. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
8. **IF** $c_i = 1$
9. $w_0 := w_0 + v_{n-i}$
10. **FOR** $j \in \{1,2,\ldots,n-1\}$
11. $w_j := w_j + v_{j-i}$
12. **IF** $c_i = -1$
13. $w_0 := w_0 - v_{n-i}$
14. **FOR** $j \in \{1,2,\ldots,n-1\}$
15. $w_j := w_j - v_{j-i}$

16. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
17. $u_i := w_i \pmod^{+} q$
18. $u := \sum_{i=0}^{n-1} u_i x^i$
19. **RETURN** u

可以看出, 在算法 12 的计算过程中, v_i 和加减运算的中间值 w_j 均有可能是负数, 这会增加算法实现和并行算法设计的难度. 综合考虑以上因素, 算法 13 描述了一种非负的并行版本小多项式算法, 它可以同时完成多个多项式 $a^{(j)}$ 与小系数多项式 c 的相乘.

算法 13. 并行小多项式乘法 Parallel-SmallPoly.

输入: $c \in \mathcal{B}_\tau, \boldsymbol{a} = \{a^{(j)}\} \in \mathcal{R}_q^r$, 其中 $a^{(j)} \in \mathcal{R}_q$;
输出: $\boldsymbol{u} = [u^{(0)},\ldots,u^{(r-1)}]^{\mathrm{T}} \in \mathcal{R}_q^r$, 其中 $u^{(j)} = c \cdot a^{(j)}$.

1. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
2. $w_i := 0$
3. $a_n := a_0$
4. **FOR** $i \in \{1,2,\ldots,n\}$
5. $v_i := 0$
6. $v_{i-n} := 0$
7. **FOR** $j \in \{0,1,\ldots,r-1\}$
8. $v_i := v_i \cdot M + (2U + a_i^{(j)})$
9. $v_{i-n} := v_i \cdot M + (2U + a_i^{(j)} + a_{i-1}^{(j)})$
10. $\gamma := 4U \cdot (M^r - 1)/(M-1)$
11. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
12. **IF** $c_i = 1$
13. $w_0 := w_0 + v_{n-i}$
14. **FOR** $j \in \{1,2,\ldots,n-1\}$
15. $w_j := w_j + v_{j-i}$
16. **IF** $c_i = -1$
17. $w_0 := w_0 + (\gamma - v_{n-i})$
18. **FOR** $j \in \{1,2,\ldots,n-1\}$
19. $w_j := w_j + (\gamma - v_{j-i})$
20. **FOR** $i \in \{0,1,2,\ldots,n-1\}$
21. $t := w_i$
22. **FOR** $j \in \{0,1,\ldots,r-1\}$
23. $u_i^{(r-1-j)} := (t \bmod M) - 2\tau U \pmod^{+} q$
24. $t := \lfloor t/M \rfloor$
25. **FOR** $j \in \{0,1,\ldots,r-1\}$
26. $u^{(j)} := \sum_{i=0}^{n-1} u_i^{(j)} x^i$
27. $\boldsymbol{u} := [u^{(0)},\ldots,u^{(r-1)}]^{\mathrm{T}}$
28. **RETURN** $\boldsymbol{u}$

算法 13 中预计算阶段的 $2U$ 和中间计算过程中的 γ 用于保障预计算值 v_i 和加减运算的中间值 w_j 均为正数, U 需要满足 $U \geqslant \|a^{(j)}\|_\infty$, γ 的计算公式如下:

$$\gamma = (2U \cdot 2) \cdot (1 + M + M^2 + \cdots + M^{r-1}) = 4U \cdot \frac{M^r - 1}{M - 1}.$$

预计算阶段的 M 是为了保证并行计算的正确性, 它的取值为: $M = 2^{\lceil \log_2(1+4\tau U) \rceil}$.

Dilithium-Prime 方案使用的就是并行版小多项式算法, 小多项式算法与 NTT 的速度对比见第 8.1 节, 表 2 中列出了 Dilithium-Prime-III (具体推荐参数如表 1 所示) 中并行小多项式的各项参数取值.

表 2　并行小多项式乘法的参数取值

运算	τ	U	$4\tau U$	M	r
cs_1	49	2	392	2^9	5
cs_2	49	2	392	2^9	6
ct_0	49	2^{12}	802 816	2^{20}	6
ct_1	49	2^{10}	200 704	2^{18}	6

6.2 素阶数域 NTT 算法

在计算 $\mathcal{R}_q$ 上两个多项式之间的乘法时, 由于素阶数域上的不可约多项式为 $x^n - x - 1$, 不能直接使用一般形式的 FFT Trick 进行运算. 文献 [20] 针对 NTRU-Prime 方案设计了多项式环 $\mathbb{Z}_q[x]/(x^n - x - 1)$ 上的 NTT 算法, 本文将基于文献 [20] 的思想, 设计适用于 Dilithium-Prime 算法的素阶数域 NTT 算法.

素阶数域上的 NTT 基本思想是将多项式环 $\mathbb{Z}_q[x]/(x^n - x - 1)$ 上的多项式乘法 $h = f \cdot g$ 扩展到一个更大的环 $\mathbb{Z}_Q[x]/(x^N - 1)$ 上进行运算, 使得循环卷积 NTT 能够被使用. 环 $\mathbb{Z}_Q[x]/(x^N - 1)$ 需要足够大, 使得 $f \cdot g$ 在 $\mathbb{Z}_Q[x]/(x^N - 1)$ 上计算得到的结果 h' 相当于 $f \cdot g$ 在 $\mathbb{Z}[x]$ 上的结果, 最后计算 h' 模 q 和不可约多项式 $x^n - x - 1$, 将结果映射回环 $\mathbb{Z}_q[x]/(x^n - x - 1)$.

主要的算法流程包括以下 3 个部分.

(1) 在 n 维多项式 f, g 的高位分别填充 0, 得到 N 维多项式 f' 和 g'.

(2) 在环 $\mathbb{Z}_Q[x]/(x^N - 1)$ 上使用循环卷积 NTT 计算得到 $h' = INTT(NTT(f') \circ NTT(g'))$.

(3) 计算 h' 模 q 和不可约多项式 $x^n - x - 1$.

可以看出算法计算正确的关键在于选取合适的 N 和 Q.

对于 N 的选取, 由于 $f \cdot g$ 在 $\mathbb{Z}[x]$ 上的结果是至多为 $2n-1$ 次的多项式, 所以 N 应满足 $N > 2n - 1$. 另外, 为了保障 $\mathbb{Z}_Q[x]/(x^N - 1)$ 上循环卷积 NTT 计算的效率, 选取的 N 应尽量多包含 2、3 等小素数因子.

对于 Q 的选取, 有 2 种选取方法均可以保证 h' 模 q 能够将结果正确地映射回环 $\mathbb{Z}_q[x]/(x^n - x - 1)$. 第 1 种最直接的考虑是令 $Q = q$, 此时 h' 映射回环 $\mathbb{Z}_q[x]/(x^n - x - 1)$ 的过程不需要再次模 q; 第 2 种方法则将多项式环扩大到 $\mathbb{Z}[x]$ 上, 由于在 $\mathbb{Z}[x]$ 上计算得到的 h' 系数 $h_k = \sum_{i+j=k} f_i g_j \geqslant n \cdot \|f\|_\infty \cdot \|g\|_\infty, k < 2n - 1$, 因此, 此时 Q 应满足 $Q > \|h\|_\infty \geqslant n \cdot \|f\|_\infty \cdot \|g\|_\infty$.

除此之外, 综合考虑 N 和 Q 的选取, 为了保证找到 $\mathbb{Z}_Q$ 上的 N 阶本原单位根, 使得循环卷积 NTT 能够进行, 还需要满足 $N | Q - 1$, 或 N 除以一个小整数因子的结果整除 $Q - 1$ (此时需要考虑不完整的 NTT 变体).

针对 Dilithium-Prime 的参数 $n = 251$, $q = 7681537$, N 应满足 $N > 501$. 对于 Q 的选取范围, 为了尽可能地选择出最合适的 Q, 这里我们计算出取值范围的精确下界. Dilithium-Prime 算法中有 3 处运算涉及 $\mathcal{R}_q$ 上两个多项式之间的乘法, 分别是密钥生成阶段的 $\mathbf{A}\boldsymbol{s}_1$, 签名阶段的 $\mathbf{A}\boldsymbol{y}$ 以及验证阶段的 $\mathbf{A}\boldsymbol{z}$, 其中矩阵中多项式的系数在 $\mathbb{Z}_q$ 内, y 的多项式系数范围为 $[-\eta, \eta]$, s_1 的多项式系数范围为 $(-\gamma_1, \gamma_1]$, $\boldsymbol{z}$ 的多项式系数范围为 $(-\gamma_1 + \beta, \gamma_1 - \beta)$, 综上, 对于 $\mathcal{R}_q$ 上的多项式乘法 $h = f \cdot g, \|f\|_\infty \leqslant q$ 且 $\|g\|_\infty \leqslant \gamma_1$, 从而 Q 应满足 $Q > n \cdot q \cdot \gamma_1$, 由于此时 $Q > 2^{32}$, 会在方案的实现上带来诸多不便, 因此下文在 Q 的选取中只考虑 $Q = q$ 的情况.

首先考虑到 $N = 512 = 2^9$ 满足 $N > 2n - 1$ 的参数要求, 并且只含有小素数因子 2. $N = 512$ 满足 $N | (q - 1)$, 因

此直接选取 $Q=q$ 即可, 此时 $\mathbb{Z}_Q[x]/(x^N-1)$ 上的计算结果 h' 只需模不可约多项式即可映射回原本的素阶数域, 节省了大量的模 q 计算, 理论算法效率较为理想.

另外, $N=576=2^6\cdot 3^2$ 同样大于 $2n-1$, 且满足 $N|(q-1)$ 的条件, 能够直接选取 $Q=q$ 作为模数. 然而, $N=576$ 含有小素数因子 3, 需要进行 6 层基-2 FFT Trick 和 2 层基-3 FFT Trick, 基-3 FFT Trick 的效率略低于基-2 FFT Trick; 但另一方面, 此时总计需要进行 8 层 FFT Trick, 少于 $N=512$ 时需要进行的 9 层 FFT Trick, 因此 $N=576$, $Q=q$ 也可作为备选参数.

两种 N 的选取均可构造合适的 NTT 算法. 由于在本文给出的 C 语言实现中, $N=512$ 表现出了更好的性能, 因此在最终方案中我们选择了 $N=512$ 的构造, 具体乘法速度的比较见第 8.1 节. 为了便于不同应用场景和实现平台上的灵活应用, 下面我们仍分别介绍 $N_1=512$ 或 $N_2=576$ 时的素阶数域 NTT 构造.

(1) N_1=512, Q=q

此时参数满足 $N_1|q-1$, 因此可以使用完整的循环卷积 NTT. 此时 CRT 同构为:

$$\mathbb{Z}_q[x]/(x^{N_1}-1)\cong\prod_{i=0}^{N_1-1}\mathbb{Z}_q[x]/(x-\rho^{\tau(i)}),$$

其中, ρ 是 $\mathbb{Z}_q$ 中 N_1 次本原单位根, $\tau(i)$ 表示第 i 个环中 ρ 的幂次. 算法总计执行 9 层基-2 FFT Trick.

(2) N_2=576, Q=q

此时参数满足 $N_2|q-1$, 同样可以使用完整的循环卷积 NTT, 此时 CRT 同构为:

$$\mathbb{Z}_q[x]/(x^{N_2}-1)\cong\prod_{i=0}^{N_2-1}\mathbb{Z}_q[x]/(x-\rho^{\tau(i)}),$$

其中, ρ 是 $\mathbb{Z}_q$ 中 N_2 次本原单位根, $\tau(i)$ 表示第 i 个环中 ρ 的幂次. 算法总计执行 6 层基-2 FFT Trick 和 2 层基-3 NTT.

表 3 汇总了上述素阶数域 NTT 算法的参数、构造和相关信息.

表 3 素阶数域 NTT 具体参数表

(n,q)	(N,Q)	参数条件	NTT类型	FFT Trick	点乘
(251, 7681537)	(512, 7681537)	$N\mid(Q-1)$	循环卷积NTT	9层基-2 FFT Trick	1×1
	(576, 7681537)	$N\mid(Q-1)$	循环卷积NTT	6层基-2 + 2层基-3 FFT Trick	1×1

7 实现细节

本节给出了 Dilithium-Prime 方案的 C 语言实现细节. 本文的实现方案主要使用长度为 n 的 32 位有符号整型数组存储多项式. 特别地, 由于 NTT 运算的需求, 在执行 NTT 计算的过程中, 多项式使用长度为 N 的 32 位有符号整型数组存储. 另外, 本文中 Dilithium-Prime 实现时选用的哈希函数与文献 [11] 保持一致.

7.1 多项式采样

Dilithium-Prime 方案中需要在不同的集合上采样 n 维多项式的系数. 由于本方案在素阶数域上的 NTT 计算不是双射, 因此不适用文献 [11] 中提出的, 为了节省计算时间直接在 NTT 域上采样的方法, 本方案中矩阵 $\mathbf{A}$, l 维多项式向量 $\boldsymbol{s}_1$ 和 $\boldsymbol{y}$, k 维多项式向量 $\boldsymbol{s}_2$ 均需要在正常域中采样.

对于矩阵 $\mathbf{A}$ 中任意多项式 a, 其中的系数 $a_i\in\mathbb{Z}_q$, 此时采样算法将哈希函数 SHAKE-128 或 AES256ctr 生成的 3 个字节随机比特流的最高位置为 0, 此时得到的系数 $a_i'\in[0,2^{23}-1]$. 随后进行拒绝采样, 若 $a_i'\leqslant q$, 则 $a_i:=a_i'$, 系数采样成功, 反之则重新进行采样, 直到得到 n 个符合条件的系数. 由于 $2^{23}-1$ 与 q 十分接近, 因此单次采样的拒绝率很低, 此时采样算法具有较好的效率.

对于多项式向量 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 中的多项式 s, 系数 $|s_i|\leqslant\eta$, $\eta=2$. 采样算法首先利用哈希函数生成的 4 个随机比特得到 $\{0,\ldots,15\}$ 之间的整数, 接着使用拒绝采样思想, 只保留小于 15 的数字, 保留下的数字模 5 得到 $\{0,\ldots,2\eta\}$ 中

的数字, 最后减去 η 即完成了系数 $s_{i,j}$ 的采样.

由于 $\boldsymbol{y}$ 中的多项式 $y_i \in \tilde{\mathcal{S}}_{\gamma_1}, \gamma_1$ 是 2 的幂次, 因此不需要进行拒绝采样, 采样算法生成 $\{0,\ldots,2\gamma_1-1\}$ 内的随机数后减去 γ_1 即可得到符合条件的系数 y_i. 对于 $\gamma_1=2^{18}$ 和 $\gamma_1=2^{19}$ 两种情况, 分别使用 19 个比特和 20 个比特的随机流生成 $\{0,\ldots,2\eta\}$ 内的随机数.

7.2 公私钥对和签名的传输

Dilithium-Prime 方案的公私钥对和签名中均包含了 $\mathcal{R}_q$ 上的多项式和多项式向量, 在每个算法内部, 这些多项式和多项式向量存储为 32 位有符号整型数组, 但在算法间传输时, 需要将它们按照系数顺序打包成字节流的形式.

具体来说, 公钥中 k 维多项式向量 $\boldsymbol{t}_1$ 的系数区间为 $[0,2^{10}-1]$, 因此每个系数在字节流中占据 10 比特, 每个多项式占据 $\lceil n\times 10/8\rceil=314$ 字节, $\boldsymbol{t}_1$ 共占据 $314k$ 字节; 种子 ρ 占据 32 字节, 因此公钥的存储共需 $314k$+32 字节.

私钥中, 随机种子 ρ 和 K 各占 32 字节, 被设置为 SHAKE-256 的哈希函数 $H(\rho||t_1)$ 的输出值 tr 占据 48 字节; k 维多项式向量 $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 中的多项式系数 $|s_i|\leqslant\eta$, 且有可能是负数, 因此在实现中计算 $s_i'=\eta-s_i, s_i'\in\{0,\ldots,2\eta\}$, 将 s_i' 打包存储进字节流. 打包过程中 s_i' 在字节流中占据 $\lceil\log_2(2\eta+1)\rceil=3$ 比特, $\boldsymbol{s}_1$ 和 $\boldsymbol{s}_2$ 共占据 $(k+l)\times\lceil 251\times 3/8\rceil=95(k+l)$ 字节; 同理, 多项式向量 $\boldsymbol{t}_0$ 中的多项式系数 $-2^{12}<t_i\leqslant 2^{12}$, 计算 $t_i'=2^{12}-t_i\in\{0,\ldots,2^{13}-1\}$, t_i' 在字节流中占据 13 比特, $\boldsymbol{t}_0$ 共占据 $k\times\lceil 251\times 13/8\rceil=408k$ 字节; 综上, 私钥的存储共需 $503k+95l+112$ 字节.

签名中, 被设置为 SHAKE-256 的哈希函数 $H(\rho||\boldsymbol{w}_1)$ 的输出值 $\tilde{c}$ 占据 32 字节; l 维多项式向量 $\boldsymbol{z}$ 的打包思路与上文类似, $-\gamma_1<z_i\leqslant\gamma_1$, 计算 $z_i'=\gamma_1-z_i\in\{0,\ldots,2\gamma_1-1\}$, z_i' 在字节流中占据 $\lceil\log_2(2\gamma_1)\rceil$ 比特, $\boldsymbol{z}$ 共占据 $l\times\lceil 251\times\lceil\log_2(2\gamma_1)\rceil/8\rceil$ 字节; k 维多项式向量 $\boldsymbol{h}$ 的存储较为特殊, 对于 $\boldsymbol{h}=\{h_1,\ldots,h_k\}$, 值为 1 的系数至多存在 ω 个, 因此使用 ω 字节按照 $i=1,\ldots,k$ 的顺序分别存储值为 1 的系数在 h_i 中的位置, 并额外使用 k 字节分别存储多项式 h_i 中系数 1 的数量. 综上, 签名的存储共需 $l\times\lceil 251\times\lceil\log_2(2\gamma_1)\rceil/8\rceil+\omega+k+32$ 字节.

7.3 常数时间实现

为了抵抗计时攻击[56], 本文给出的实现尽量避免使用非常数时间算法计算秘密值, 并且给出了一些算法的无分支实现, 以防在利用秘密值进行判断的过程中泄露秘密值信息. 注意在签名算法执行拒绝采样时, 多项式系数被拒绝的概率与秘密信息无关, 因此无需使用常数时间算法和无分支策略.

7.3.1 模约减算法

针对普通模数 q 的模约减, C 语言中的“%”运算符可以实现任意模数的模运算, 但是它的运行时间与输入数据的长度有关. 在对秘密值进行模运算时, 敌手能够利用这一特性, 观察算法运行时间来获得和秘密值有关的信息.

基于以上考虑, 本文在乘法约减中使用了 Montgomery 约减算法[57], Montgomery 约减算法适用于任何模数, 并且在常数时间内实现. 而对于独立的约减计算, 本文借鉴 Barrett 约减[58]的近似思想, 针对模数 q 设计了无分支的高效约减算法 Reduce, 如算法 14 所示.

算法 14. 约减算法 Reduce.

输入: $r\in[-2^{31}+1,2^{31}-1]$;

输出: $r \bmod q\in[-7181504,7181504]$.

---

1. $r':=(r\times 280+(1\ll 30))\gg 31$
2. $r:=r-r'\times q$
3. **RETURN** r

算法 14 借鉴了 Barrett 约减中的近似思想, 将 $[-2^{31}+1,2^{31}-1]$ 区间内的数约减至区间 $[-7181504,7181504]$ 内, 该区间为 $[-q,q]$ 的子区间. 算法 14 不仅针对模数 q 进行了计算上的简化, 提升约减效率, 同时不需要 IF 分支

语句进行判断, 传统的 Barrett 约减则需要进行分支判断, 容易受到侧信道攻击.

另外, 针对算法中一些特殊的 2 次幂模数, 取模运算可以使用位运算进行, 更加简洁高效.

7.3.2 无分支 Decompose_q 算法

由于方案需要使用 Decompose_q 算法对秘密值进行操作, 因此本文给出了 Decompose_q 算法的一种无分支实现方法, 避免了算法 6 中的 IF 分支语句. 算法 15 和算法 16 分别给出了这一算法在 Dilithium-Prime-II 和 Dilithium-Prime-III/Dilithium-Prime-V 中的具体实现.

算法 15. 无分支分解算法 $\text{Decompose}_{q,2}$.

输入: $r \in \mathbb{Z}_q$, $\alpha = (q-1)/16$, q;
输出: r 的高位和低位 (r_1, r_0).

1. $r := r \bmod^+ q$
2. $r_1 := \lfloor (r + (\alpha/2 - 1))/\alpha \rfloor$
3. $r_1 := r_1 \& 15$
4. $r_0 := r - r_1 \times \alpha$
5. $r_0 := r_0 - (((q-1)/2 - r_0) \gg 31) \& q$
6. **RETURN** (r_1, r_0)

算法 16. 无分支分解算法 $\text{Decompose}_{q,3,5}$.

输入: $r \in \mathbb{Z}_q$, $\alpha = (q-1)/8$, q;
输出: r 的高位和低位 (r_1, r_0).

1. $r := r \bmod^+ q$
2. $r_1 := \lfloor (r + (\alpha/2 - 1))/\alpha \rfloor$
3. $r_1 := r_1 \& 7$
4. $r_0 := r - r_1 \times \alpha$
5. $r_0 := r_0 - (((q-1)/2 - r_0) \gg 31) \& q$
6. **RETURN** (r_1, r_0)

在算法 15 和算法 16 中, "&"代表按位与运算. 算法无分支的主要思想是利用 $r_1 \in \{0, 1, \ldots, (q-1)/\alpha\}$ 的特点, 当 $r_1 = (q-1)/\alpha$ 时, 利用位运算将 r_1 置为 0, 将对应的 r_0 置为 $r_0 - 1$, 此时有 $r_1' = 0$, r_0' 满足:

$$\begin{aligned} r_0' &= r_0 - 1 = r - r_1\alpha - 1 \\ &= r - (q-1) - 1 = r - r_1'\alpha - q \end{aligned}$$

无分支算法使用了 $\lfloor (r + (\alpha/2 - 1))/\alpha \rfloor$ 计算 r_1, 此时对应的 $r_0 \in (-\alpha/2, \alpha/2]$, 文献 [59] 描述了利用乘法和移位实现除法的优化算法, 基于文献 [59] 中常数时间除法的思想, 本文给出了计算 r_1 的快速算法, 即算法 17.

算法 17. 常数时间高位计算算法 $Evaluate_{r_1}$.

输入: $r \in \mathbb{Z}_q$, $0 < \alpha < \dfrac{q}{2}$;
输出: r 的高位 r_1.

1. $N := \lceil \log_2(q + (\alpha/2 - 1)) \rceil$
2. $r := r + (\alpha/2 - 1)$
3. $L := \lfloor \log_2 \alpha \rfloor$

4. $c := \lceil 2^{N+L}/\alpha \rceil$
5. **IF** $\alpha = 2^K, K > 0$
6. **RETURN** $r_1 = r \gg K$
7. **ELSEIF** $c \times (2^N - 2^N \bmod^+ \alpha - 1) < \lfloor 2^N/d \rfloor \times 2^{N+L}$
8. **RETURN** $r_1 = cr \gg (N+L)$
9. **ELSEIF** $\alpha = 2^K d, K > 0$
10. $L := \lceil \log_2 d \rceil$
11. $c := \lceil 2^{N-K+L}/d \rceil$
12. **RETURN** $r_1 = (c \times (r \gg K)) \gg (N-K+L)$
13. **ELSE**
14. $L := \lceil \log_2 \alpha \rceil$
15. $c := c - 2^N$
16. $c := \lfloor r \times c/2^N \rfloor$
17. **RETURN** $r_1 = (c + ((r-c) \gg 1)) \gg (L-1)$

在实际计算过程中, 根据方案对应的 α 值进行预计算, 就能够将算法 17 中的具体分支和相应参数代入算法 15–算法 16, 避免 IF 语句和除法的存在.

7.4 NTT 优化实现

文献 [32–34] 描述了将 Cooley-Tukey 蝴蝶变换和 Gentlemen-Sande 蝴蝶变换分别应用于正向和逆向基-2 FFT Trick, 使得计算基-2 NTT 的时间复杂度降低至 $\mathrm{O}(n\log n)$, 这也是在高次多项式乘法中, 使得 NTT 算法效率远高于 School Book 乘法的关键技术.

在本文使用到的基-3 FFT Trick 中, 可以使用类似蝴蝶变换的思想, 利用单位根的特殊性质减少计算过程中的乘法, 加速 NTT 运算.

基-3 FFT Trick 的 CRT 分解为:

$$\mathbb{Z}_q[x]/(x^{3m}-\zeta^3) \cong \prod_{i=0}^{2} \mathbb{Z}_q[x]/(x^m-\rho^i\zeta),$$
$$f \mapsto \left\{ f \bmod (x^m-\rho^i\zeta) \right\},$$

其中, ρ 为 $\mathbb{Z}_q$ 中的 3 次单位根, $3m$ 维多项式 f 可以写为:

$$f = f_0 + f_1 x^m + f_2 x^{2m},$$

其中, f_i 均为 m 维多项式, 经过基-3 分解后, f 被映射到的 3 个多项式可以表示为:

$$\begin{cases} f \bmod (x^m-\zeta) = f_0 + f_1\zeta + f_2\zeta^2 \\ f \bmod (x^m-\rho\zeta) = f_0 + f_1\rho\zeta + f_2\rho^2\zeta^2 \\ f \bmod \left(x^m-\rho^2\zeta\right) = f_0 + f_1\rho^2\zeta + f_2\rho\zeta^2 \end{cases}$$

利用三次单位根 $\rho^2+\rho+1=0$ 的性质, 记 $a_1 = f_1\rho\zeta, a_2 = f_2\rho^2\zeta^2, a_3 = \rho(f_1\rho\zeta - f_2\rho^2\zeta^2)$, 有:

$$\begin{cases} f \bmod (x^m-\zeta) = f_0 - a_3 - a_1 \\ f \bmod (x^m-\rho\zeta) = f_0 + a_1 + a_2 \\ f \bmod \left(x^m-\rho^2\zeta\right) = f_0 + a_3 - a_2 \end{cases}$$

此时共需要 $3m$ 个乘法, 比优化前需要 $6m$ 个乘法节省了一半的消耗.

8 实验结果与比较

本节将给出 Dilithium-Prime 方案中使用的两类多项式乘法 C 语言实现的测试数据, 以及方案总体的测试数

据, 同时给出以上数据同其他方案的对比. 测试对象为安全等级III的参数集. 测试设备为: 硬件配置为 2.5 GHz 的 Intel(R) Core(TM) i5-12400F CPU 和 32 GB 内存的电脑, 测试时关闭 Turbo Boost 和 Hyperthreading. 操作系统为: 核为 Linux Kernel 4.4.0 的 Ubuntu 20.04 LTS 操作系统. GCC 版本为 9.4.0, 编译选项为-O3. 所有测试结果均以 CPU 周期数为单位, 取 10000 次运行结果的中位数.

8.1 多项式乘法比较

本节使用安全等级为 III 的 Dilithium-Prime 参数集, 针对 $\mathcal{R}_q$ 中的多项式乘法, 对比了两种素阶数域 NTT 算法和 School Book 算法的计算效率; 针对 $\mathcal{R}_q$ 中多项式与 $\mathcal{B}_\tau$ 中多项式的乘法, 对比了小多项式乘法、两种素阶数域 NTT 算法和 School Book 算法的计算效率, 具体测试数据见表 4, 单位为 CPU 周期 (cycle).

表 4 多项式乘法效率的对比 (cycle)

算法	$\mathcal{R}_q$ 中多项式乘法		$\mathcal{R}_q$ 和 $\mathcal{B}_\tau$ 中多项式乘法			
	$\mathcal{R}_q\times\mathcal{R}_q$	$\mathcal{R}_q^{k\times l}\times\mathcal{R}_q^{l}$	cs_1	cs_2	ct_0	ct_1
School Book算法	246595	7121005	1180042	1415762	1423710	1421439
素阶数域NTT算法 ($N=512$)	**25451**	**337249**	93280	108463	113152	109949
素阶数域NTT算法 ($N=576$)	26161	349323	97974	112232	113733	114088
小多项式乘法	—	—	**8132**	**8760**	**13700**	**13397**

(1) $\mathcal{R}_q$ 中多项式乘法的比较

对于 $\mathcal{R}_q$ 中的多项式乘法, 本文测试了单个多项式乘法, 以及多项式矩阵和多项式向量之间的乘法, 后者在组成 Dilithium-Prime 方案的 3 个算法中均有使用, 如 $\mathbf{Ay},\mathbf{A}\boldsymbol{s}_1$ 等计算, 是方案中耗时最久的计算之一. 可以看出, 不论是在单个多项式乘法还是多项式矩阵-向量乘法中, 两种素阶数域 NTT 算法的效率均明显高于 School Book 算法, 其中 $N=512$ 的素阶数域 NTT 效率高于 $N=576$ 的素阶数域 NTT.

具体而言, 在单个多项式相乘的情况下, $N=512$ 的素阶数域 NTT 和 $N=576$ 的素阶数域 NTT 算法的效率分别比 School Book 算法快 9.7 倍和 9.4 倍. 在多项式矩阵-向量的情况下, $N=512$ 的素阶数域 NTT 和 $N=576$ 的素阶数域 NTT 算法比 School Book 算法分别快 21.1 倍和 20.4 倍. 综上, 两种素阶数域 NTT 算法均可实现对 $\mathcal{R}_q$ 中的多项式乘法的有效加速, 并且 $N=512$ 的素阶数域 NTT 加速效果相对更好. 尤其是在多项式向量-矩阵乘法中, 多项式向量的正向 NTT 计算结果可被重复使用, 加速效果更加明显, 能够节省约 95% 的计算时间, 这一特性也体现在多项式向量-多项式乘法中, 图 1 中数据均为各运算场景下素阶数域 NTT 运行时间占 School Book 乘法运行时间的比例. 由图 1 可见正向 NTT 计算结果的复用率越高, 计算效率的提升越明显.

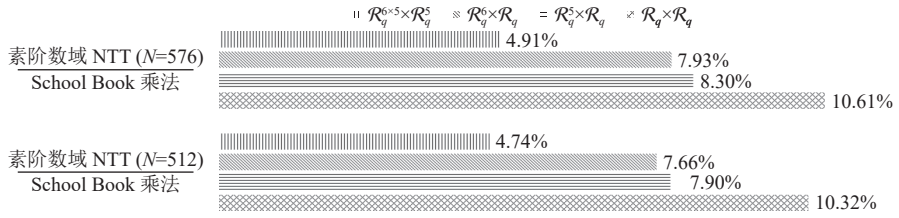

图 1 不同计算场景下素阶数域 NTT 性能提升的比较

(2) $\mathcal{R}_q$ 中多项式与 $\mathcal{B}_\tau$ 中多项式的乘法比较

对于 $\mathcal{R}_q$ 中多项式与 $\mathcal{B}_\tau$ 中多项式的乘法, 本文分别测试了 Dilithium-Prime 方案中所有的此类乘法运算, 即 cs_1, cs_2 和 ct_0, ct_1, 四者均为单独多项式与 k/l 维多项式向量的乘法. 可以看出, 在所有的计算情形中, 两类素阶数域 NTT 算法明显快于 School Book 乘法, 而小多项式乘法则显著快于素阶数域 NTT 算法.

具体而言, 在 cs_1 的计算中, $N=512$ 的素阶数域 NTT、$N=576$ 的素阶数域 NTT 算法、小多项式乘法比 School Book 算法分别快 12.6 倍、12.0 倍、145.1 倍, 其中 $N=512$ 的素阶数域 NTT 比 $N=576$ 的素阶数域 NTT

快 4.8%, 小多项式乘法则比 $N=512$ 的素阶数域 NTT 算法快 11.5 倍. 在 $c\boldsymbol{s}_2$ 的计算中, $N=512$ 的素阶数域 NTT、$N=576$ 的素阶数域 NTT 算法、小多项式乘法比 School Book 算法分别快 13.1 倍、12.6 倍、161.6 倍, $N=512$ 的素阶数域 NTT 比 $N=576$ 的素阶数域 NTT 快 3.4%, 小多项式乘法则比 $N=512$ 的素阶数域 NTT 算法快 12.4 倍. 在 $c\boldsymbol{t}_0$ 的计算中, $N=512$ 的素阶数域 NTT、$N=576$ 的素阶数域 NTT 算法、小多项式乘法比 School Book 算法分别快 12.6 倍、12.5 倍、103.9 倍, 在 $c\boldsymbol{t}_1$ 的计算中, $N=512$ 的素阶数域 NTT、$N=576$ 的素阶数域 NTT 算法、小多项式乘法比 School Book 算法分别快 12.9 倍、12.5 倍、106.1 倍, 可以看出, 此时小多项式乘法相比于 School Book 算法的提速效果略低于此前两种情形, 但仍显著优于素阶数域 NTT 算法, 具体而言, 在在 $c\boldsymbol{t}_0$ 和 $c\boldsymbol{t}_1$ 的计算中, $N=512$ 的素阶数域 NTT 平均比 $N=576$ 的素阶数域 NTT 快 2.1%, 小多项式乘法则平均比 $N=512$ 的素阶数域 NTT 算法快 8.2 倍.

可以看出, 相比于素阶数域 NTT 算法, 小多项式算法的提速更加显著. 相比于 $c\boldsymbol{s}_1$ 的计算, $c\boldsymbol{s}_2$ 中小多项式算法提速更明显的原因是, $\boldsymbol{s}_1\in\mathcal{R}_q^l$, $\boldsymbol{s}_2\in\mathcal{R}_q^k$, 且 $k>l$, 故在 $c\boldsymbol{s}_2$ 的计算中, 小多项式算法的并行程度更高. 同样在 $\boldsymbol{t}_0,\boldsymbol{t}_1\in\mathcal{R}_q^k$ 的情况下, 小多项式乘法对 $c\boldsymbol{t}_0, c\boldsymbol{t}_1$ 计算的提速效果稍差是由于 $\boldsymbol{t}_0,\boldsymbol{t}_1$ 中各项系数较大, 由于 64 位的存储限制, 需要对并行计算进行拆分, 导致此时小多项式算法的并行程度降低. 图 2 中数据均为不同并行度下小多项式乘法运行时间占对应多项式乘法运行时间的比例, 展示了小多项式乘法的并行度增加带来的加速效果提升.

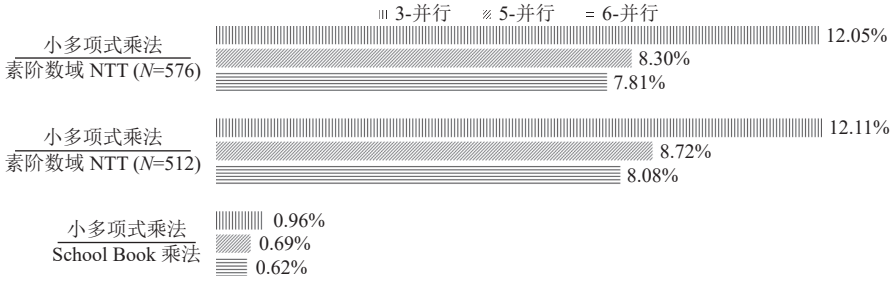

图 2 不同并行度的小多项式乘法效率比较

综合上述分析, 两类素阶数域 NTT 算法均可实现对多项式算法的有效加速, 其中 $N=512$ 的素阶数域 NTT 在各个计算场景中均比 $N=576$ 的素阶数域 NTT 快, 因此在 $\mathcal{R}_q$ 中的多项式乘法中, 方案使用 $N=512$ 的素阶数域 NTT 算法; 另一方面, 小多项式乘法相比于素阶数域 NTT 算法实现了更显著的提速, 因此在满足使用条件的情形, 即 $\mathcal{R}_q$ 中多项式与 $\mathcal{B}_\tau$ 中多项式的乘法中, 方案使用小多项式乘法进行计算.

8.2 方案性能比较

本节对比了 Dilithium-Prime 算法、CRYSTALS-Dilithium 算法[11]和同样基于 Fiat-Shamir with Aborts 的素阶数域签名方案 NCC-Sign[30].

表 5 给出了方案对比的详细数据, 其中 $|pk|$ 为公钥尺寸, $|sk|$ 为私钥尺寸, $|\sigma|$ 为签名尺寸, 三者均以字节为单位; “Exp.reps”代表签名过程中拒绝次数的期望; *KeyGen*、*Sign* 和 *Verify* 这 3 项则分别给出了表 5 中方案分别运行密钥生成算法、签名算法、验证算法时耗费的时间, 以 CPU 周期计数, 单位为千 CPU 周期 (kcycles).

表 5 方案的性能对比和实现效率对比

方案	$\|pk\|$ (byte)	$\|sk\|$ (byte)	$\|\sigma\|$ (byte)	Exp.reps	*KeyGen* (kcycles)	*Sign* (kcycles)	*Verify* (kcycles)
CRYSTALS-Dilithium[11]	1952	4016	3293	5.1	246.6	775.8	224.1
NCC-Sign[30]	1997	3312	3605	5.7	2057.7	24106.3	4032.0
Dilithium-Prime	**1916**	**3605**	**3233**	**3.0**	**484.2**	**683.8**	**563.3**

在对比时, 本文选用了第 5.5 节中给出的 Dilithium-Prime 安全等级III的推荐参数组. 基于同一安全等级下横向比较的原则, 本文同样使用 CRYSTALS-Dilithium 方案和 NCC-Sign 方案中安全等级为 III 的推荐参数组进行数

据测试. 其中 CRYSTALS-Dilithium 测试所使用的实现代码来自 NIST 官网提供的 C 语言参考实现. NCC-Sign 的参数和测试 C 语言代码均来自韩国 PQC 竞赛的第 1 轮提交[30].

(1) 与分圆环方案 Dilithium-Prime 的比较

从表 5 中可以看出, 与 CRYSTALS-Dilithium 方案相比, 本文给出的 Dilithium-Prime 方案的公钥尺寸、私钥尺寸和签名尺寸均略小, 其中私钥尺寸相差较为明显, 其余两项则相差较小. 具体而言, Dilithium-Prime 方案的公钥尺寸、私钥尺寸和签名尺寸分别比 CRYSTALS-Dilithium 方案中小 1.8%、10.2%、1.8%.

从运行效率上来看, Dilithium-Prime 方案的密钥生成速度和签名验证速度分别是 CRYSTALS-Dilithium 的 2.0 倍和 2.5 倍, 其主要原因是素阶数域上的 NTT 算法效率相对较低, 且无法从种子直接生成 NTT 域上的 $\widehat{\mathbf{A}}$, 后者导致密钥生成算法、签名算法、验证算法均需要比 CRYSTALS-Dilithium 方案多进行 $k\times l$ 次正向 NTT 运算, 从而将 $\mathbf{A}$ 映射到 NTT 域上. Dilithium-Prime 的签名算法中也存在相同的问题, 但本文通过参数选择, 控制签名算法的拒绝次数, 使得签名算法的效率比 CRYSTALS-Dilithium 快 11.9%. 总体而言, Dilithium-Prime 方案能够抵抗针对分圆环的攻击, CRYSTALS-Dilithium 则存在此类安全隐患. 相比于 CRYSTALS-Dilithium, Dilithium-Prime 的密钥生成算法和验证算法较慢, 但签名算法较快, 事实上, 一对成功生成的公私钥可被多次使用, 因此在实际应用中, 密钥生成算法和验证算法耗时的影响较小, Dilithium-Prime 方案在能够适应多数现实应用场景.

(2) 与素阶数域方案 NCC-Sign 的比较

从表 5 中可以看出, 本文给出的 Dilithium-Prime 方案的公钥尺寸、私钥尺寸、签名尺寸之和与 NCC-Sign 基本持平.

从实现效率的方面来看, 同样为基于 Fiat-Shamir with Aborts 范式的素阶数域签名方案, Dilithium-Prime 方案的密钥生成算法、签名算法、验证算法的速度提升至 4.2 倍、35.3 倍、7.2 倍, 其主要原因是 NCC-Sign 方案中使用了效率较低的 Toom-Cook 算法进行多项式乘法, 导致计算效率极低, 另一方面, NCC-Sign 方案签名过程中拒绝次数较多也是方案效率不理想的重要原因之一. 总体而言, Dilithium-Prime 方案在素阶数域上实现了高效的签名方案, 兼顾了安全性与高效性两个方面的考量.

9 结 论

本文设计了基于素阶数域和 Fiat-Shamir with Aborts 范式的数字签名方案 Dilithium-Prime, 并给出了推荐参数集. 针对素阶数域无法使用传统 NTT 算法加速多项式乘法的缺点, 本文设计了素阶数域 NTT 算法和小多项式乘法, 实现了素阶数域上高效的多项式乘法. 在实现过程中, 本文为方案关键算法设计了常数时间无分支实现. 实验结果表明, 在同一安全等级下, 与分圆环上的数字签名方案 CRYSTALS-Dilithium 的推荐参数相比, Dilithium-Prime 方案的公钥尺寸、私钥尺寸、签名尺寸分别降低 1.8%、10.2%、1.8%; 签名算法效率提高 11.9%, 密钥生成算法、验证算法所需的时间分别为 CRYSTALS-Dilithium 方案的 2.0 倍和 2.5 倍. 另一方面, Dilithium-Prime 方案能够抵抗针对分圆环的潜在密码分析, 具有更鲁棒的安全性. 与韩国后量子密码标准竞赛提案中的素阶数域签名方案 NCC-Sign 的推荐参数相比, 在同一安全等级和非常接近的带宽条件下, Dilithium-Prime 方案的密钥生成算法、签名算法、验证算法的速度分别提升至 4.2 倍、35.3 倍、7.2 倍, 实现了兼顾高效性和安全性的素阶数域签名算法.

后续工作拟考虑 Dilithium-Prime 方案在更多平台上的高效实现, 例如 AVX2 实现、ARM Cortex-M4 实现、GPU 实现和 FPGA 实现等.

References:

[1] Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review, 1999, 41(2): 303–332. [doi: 10.1137/S0036144598347011]

[2] Regev O. An efficient quantum factoring algorithm. arXiv:2308.06572, 2023.

[3] NIST. PQC standardization process: Announcing four candidates to be standardized, plus fourth round candidates. 2022. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4

[4] Chinese Association for Cryptologic Research. Announcement of the selection results of the national cryptographic algorithm

competitions. 2020 (in Chinese). https://www.cacrnet.org.cn/site/content/854.html

[5] Regev O. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 2009, 56(6): 34. [doi: 10.1145/1568318.1568324]

[6] Banerjee A, Peikert C, Rosen A. Pseudorandom functions and lattices. In: Proc. of the 31st Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Cambridge: Springer, 2012. 719–737. [doi: 10.1007/978-3-642-29011-4_42]

[7] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings. In: Proc. of the 29th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. French Riviera: Springer, 2010. 1–23. [doi: 10.1007/978-3-642-13190-5_1]

[8] Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 2015, 75(3): 565–599. [doi: 10.1007/s10623-014-9938-4]

[9] Avanzi R, Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, Schwabe P, Seiler G, Stehlé D. CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.01). 2021. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf

[10] Basso A, Mera JMB, D'Anvers JP, Karmakar A, Sinha Roy S, van Beirendonck M, Vercauteren F. SABER: Mod-LWR based KEM (round 3 submission). 2023. https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/saberspecround3.pdf

[11] Bai S, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, Stehlé D. CRYSTALS-Dilithium algorithm specifications and supporting documentation (version 3.1). 2021. https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[12] Fouque PA, Hoffstein J, Kirchner P, Lyubashevsky V, Pornin T, Prest T, Ricosset T, Seiler G, Whyte W, Zhang ZF. Falcon: Fast-Fourier lattice-based compact signatures over NTRU (specification v1.2). 2020. https://falcon-sign.info/falcon.pdf

[13] Smart NP, Vercauteren F. Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Proc. of the 13th Int'l Workshop on Public Key Cryptography. Paris: Springer, 2010. 420–443. [doi: 10.1007/978-3-642-13013-7_25]

[14] Campbell P, Groves M, Shepherd D. Soliloquy: A cautionary tale. In: Proc. of the 2nd ETSI Quantum-safe Crypto Workshop in Partnership with the IQC. Ottawa, 2014. 1–9.

[15] Biasse JF, Song F. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In: Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms. Arlington: SIAM, 2016. 893–902.

[16] Cramer R, Ducas L, Wesolowski B. Mildly short vectors in cyclotomic ideal lattices in quantum polynomial time. Journal of the ACM, 2021, 68(2): 8. [doi: 10.1145/3431725]

[17] Bernstein DJ, Lange T. Non-randomness of S-unit lattices. 2021. https://eprint.iacr.org/2021/1428

[18] Bernstein DJ, Brumley B, Chen MS, Chuengsatiansup C, Lange T, Marotzke A, Peng BY, Tuveri N, van Vredendaal C, Yang BY. NTRU Prime: Round 3. 2020. https://ntruprime.cr.yp.to/nist.html

[19] Bernstein DJ, Chuengsatiansup C, Lange T, van Vredendaal C. NTRU prime: Reducing attack surface at low cost. In: Proc. of the 24th Int'l Conf. on Selected Areas in Cryptography. Ottawa: Springer, 2018. 235–260. [doi: 10.1007/978-3-319-72565-9_12]

[20] Alkim E, Cheng DYL, Chung CMM, Evkan H, Huang LWL, Hwang V, Li CLT, Niederhagen R, Shih CJ, Wälde J, Yang BY. Polynomial multiplication in NTRU prime: Comparison of optimization strategies on Cortex-M4. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2020, 2021(1): 217–238. [doi: 10.46586/tches.v2021.i1.217-238]

[21] OpenSSH Release Notes. OpenSSH 9.0/9.0p1. 2022. https://www.openssh.com/releasenotes.html

[22] Ajtai M. Generating hard instances of lattice problems. In: Proc. of the 28th Annual ACM Symp. on Theory of Computing. Philadelphia: ACM, 1996. 99–108. [doi: 10.1145/237814.237838]

[23] Goldreich O, Goldwasser S, Halevi S. Public-key cryptosystems from lattice reduction problems. In: Proc. of the 17th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 1997. 112–131. [doi: 10.1007/BFb0052231]

[24] Hoffstein J, Howgrave-Graham N, Pipher J, Silverman JH, Whyte W. NTRUSign: Digital signatures using the NTRU lattice. In: Cryptographers' Track at the RSA Conf. on Topics in Cryptology. San Francisco: Springer, 2003. 122–140. [doi: 10.1007/3-540-36563-X_9]

[25] Nguyen PQ, Regev O. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In: Proc. of the 25th Int'l Conf. on the Theory and Applications of Cryptographic Techniques. St. Petersburg: Springer, 2006. 271–288. [doi: 10.1007/11761679_17]

[26] Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: Proc. of the 40th Annual ACM Symp. on Theory of Computing. Victoria: ACM, 2008. 197–206. [doi: 10.1145/1374376.1374407]

[27] Lyubashevsky V. Lattice signatures without trapdoors. In: Proc. of the 31st Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Cambridge: Springer, 2012. 738–755. [doi: 10.1007/978-3-642-29011-4_43]

[28] Güneysu T, Lyubashevsky V, Pöppelmann T. Practical lattice-based cryptography: A signature scheme for embedded systems. In: Proc. of the 14th Int'l Workshop on Cryptographic Hardware and Embedded Systems. Leuven: Springer, 2012. 530–547. [doi: 10.1007/978-3-

642-33027-8_31]

[29] Bai S, Galbraith SD. An improved compression technique for signatures based on learning with errors. In: Proc. of the 2014 Cryptographer's Track at the RSA Conf. on Topics in Cryptology. San Francisco: Springer, 2014. 28–47. [doi: 10.1007/978-3-319-04852-9_2]

[30] Shim KA, Kim J, An Y. NCC-Sign: A new lattice-based signature scheme using non-cyclotomic polynomials. 2023. https://www.kpqc.or.kr/images/pdf/NCC-Sign.pdf

[31] Kiltz E, Lyubashevsky V, Schaffner C. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Proc. of the 37th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Tel Aviv: Springer, 2018. 552–586. [doi: 10.1007/978-3-319-78372-7_18]

[32] Bernstein DJ. Multidigit multiplication for mathematicians. 2001. http://cr.yp.to/papers.html#m3

[33] Cooley JW, Tukey JW. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 1965, 19(90): 297–301. [doi: 10.1090/S0025-5718-1965-0178586-1]

[34] Gentleman WM, Sande G. Fast Fourier transforms: For fun and profit. In: Proc. of the 1966 AFIPS Fall Joint Computer Conf. San Francisco: ACM, 1966. 563–578. [doi: 10.1145/1464291.1464352]

[35] Cramer R, Ducas L, Peikert C, Regev O. Recovering short generators of principal ideals in cyclotomic rings. In: Proc. of the 35th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Vienna: Springer, 2016. 559–585. [doi: 10.1007/978-3-662-49896-5_20]

[36] Biasse JF, Song F. On the quantum attacks against schemes relying on the hardness of finding a short generator of an ideal in $\mathbb{Q}(\zeta_{2^r})$. Journal of Mathematical Cryptology, 2019, 13(3–4): 151–168. [doi: 10.1515/jmc-2015-0046]

[37] Eisenträger K, Hallgren S, Kitaev A, Song F. A quantum algorithm for computing the unit group of an arbitrary degree number field. In: Proc. of the 46th ACM Symp. on Theory of Computing. New York: ACM, 2014. 293–302. [doi: 10.1145/2591796.2591860]

[38] Laarhoven T. Sieving for closest lattice vectors (with preprocessing). In: Proc. of the 2016 Int'l Conf. on Selected Areas in Cryptography. Cham: Springer, 2016.

[39] Pellet-Mary A, Hanrot G, Stehlé D. Approx-SVP in ideal lattices with pre-processing. In: Proc. of the 38th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Darmstadt: Springer, 2019. 685–716. [doi: 10.1007/978-3-030-17656-3_24]

[40] Bernstein DJ. Fast norm computation in smooth-degree Abelian number fields. 2022. https://eprint.iacr.org/2022/980

[41] Duc A, Tramèr F, Vaudenay S. Better algorithms for LWE and LWR. In: Proc. of the 34th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Sofia: Springer, 2015. 173–202. [doi: 10.1007/978-3-662-46800-5_8]

[42] Buchmann J, Göpfert F, Player R, Wunderer T. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In: Proc. of the 8th Int'l Conf. on Cryptology in Africa. Fes: Springer, 2016. 24–43. [doi: 10.1007/978-3-319-31517-1_2]

[43] Albrecht MR. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In: Proc. of the 36th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Paris: Springer, 2017. 103–129. [doi: 10.1007/978-3-319-56614-6_4]

[44] Cheon JH, Hhan M, Hong S, Son Y. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE. IEEE Access, 2019, 7: 89497–89506. [doi: 10.1109/ACCESS.2019.2925425]

[45] Guo Q, Johansson T. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In: Proc. of the 27th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Singapore: Springer, 2021. 33–62. [doi: 10.1007/978-3-030-92068-5_2]

[46] MATZOV. Report on the security of LWE: Improved dual lattice. 2022. https://zenodo.org/record/6412487

[47] Bauch J, Bernstein DJ, de Valence H, Lange T, van Vredendaal C. Short generators without quantum computers: The case of multiquadratics. In: Proc. of the 36th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Paris: Springer, 2017. 27–59. [doi: 10.1007/978-3-319-56620-7_2]

[48] Eisenträger K, Hallgren S, Lauter K. Weak instances of PLWE. In: Proc. of the 21st Int'l Conf. on Selected Areas in Cryptography. Montreal: Springer, 2014. 183–194. [doi: 10.1007/978-3-319-13051-4_11]

[49] Chen H, Lauter KE, Stange KE. Security considerations for Galois non-dual RLWE families. IACR Cryptology ePrint Archive, Paper ID: 2016/193. https://eprint.iacr.org/2016/193

[50] Poddebniak D, Somorovsky J, Schinzel S, Lochter M, Rösler P. Attacking deterministic signature schemes using fault attacks. In: Proc. of the 2018 IEEE European Symp. on Security and Privacy (EuroS&P). London: IEEE, 2018. 338–352. [doi: 10.1109/EuroSP.2018.00031]

[51] Samwel N, Batina L, Bertoni G, Daemen J, Susella R. Breaking Ed25519 in WolfSSL. In: Proc. of the 2018 Cryptographers' Track at the RSA Conf. on Topics in Cryptology. San Francisco: Springer, 2018. 1–20. [doi: 10.1007/978-3-319-76953-0_1]

[52] Alkim E, Ducas L, Pöppelmann T, Schwabe P. Post-quantum key exchange: A new hope. In: Proc. of the 25th USENIX Security Symp. (USENIX Security 16). Austin: USENIX Association, 2016. 327–343.

[53] Chen YM, Nguyen PQ. BKZ 2.0: Better lattice security estimates. In: Proc. of the 17th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Seoul: Springer, 2011. 1–20. [doi: 10.1007/978-3-642-25385-0_1]

[54] Bai S, Galbraith SD. Lattice decoding attacks on binary LWE. In: Proc. of the 19th Australasian Conf. on Information Security and Privacy. Wollongong: Springer, 2014. 322–337. [doi: 10.1007/978-3-319-08344-5_21]

[55] Zheng JY, He F, Shen SY, Xue CX, Zhao YL. Parallel small polynomial multiplication for dilithium: A faster design and implementation. In: Proc. of the 38th Annual Computer Security Applications Conf. Austin: ACM, 2022. 304–317. [doi: 10.1145/3564625.3564629]

[56] Kocher PC. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Proc. of the 16th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 1996. 104–113. [doi: 10.1007/3-540-68697-5_9]

[57] Montgomery PL. Modular multiplication without trial division. Mathematics of Computation, 1985, 44(170): 519–521. [doi: 10.1090/S0025-5718-1985-0777282-X]

[58] Barrett P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM, ed. Proc. of the 1987 Conf. on the Theory and Application of Cryptographic Techniques (CRYPTO 1986). Berlin, Heidelberg: Springer, 1987. 311–323. [doi: 10.1007/3-540-47721-7_24]

[59] Granlund T, Montgomery PL. Division by invariant integers using multiplication. In: Proc. of the 1994 ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI). Orlando: ACM, 1994. 61–72. [doi: 10.1145/178243.178249]

附中文参考文献:

[4] 中国密码学会. 关于全国密码算法设计竞赛算法评选结果的公示. 2020. https://www.cacrnet.org.cn/site/content/854.html

董怡帆(2000－), 女, 硕士生, CCF 学生会员, 主要研究领域为格密码.

梁志闯(1997－), 男, 博士生, 主要研究领域为格密码.

方博越(1997－), 男, 讲师, 主要研究领域为格密码.

赵运磊(1974－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为后量子密码, 密码协议与计算理论.