软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(2):747−775 [doi: 10.13328/j.cnki.jos.007161] [CSTR: 32375.14.jos.007161]

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

素阶数域上的高效紧凑 NTRU 密钥封装方案*

梁志闯[1], 赵旭阳[1], 方博越[1], 赵运磊[1,2]

[1](复旦大学 计算机科学技术学院, 上海 200433)

[2](密码科学技术国家重点实验室, 北京 100036)

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

摘　要: 基于格 (特别是 NTRU 格) 设计后量子密钥封装方案是格密码领域的主流方向之一. 现有多数格密码方案基于分圆环构造, 但分圆环饱含丰富的代数结构导致这些方案容易遭受相关攻击. 一个可选的且更安全的代数结构是大 Galois 群、素数阶、基于素理想的数域 (简称为素阶数域). NTRU-Prime 是一个基于素阶数域的备受青睐的 NTRU 密钥封装方案, 且早已经在国际标准 OpenSSH 中默认应用. 旨在设计出比 NTRU-Prime 性能更优的素阶数域上 NTRU 密钥封装方案. 首先, 梳理分圆环的安全隐患, 特别是针对 2 次幂分圆环的系列攻击, 同时展示出素阶数域在抵御这些攻击方面的安全优势. 接着, 基于素阶数域提出 NTRU 密钥封装方案 CNTR-Prime, 并给出详细的相关分析和参数集. 然后, 提出一种伪梅森数不完整 NTT, 它能有效计算 CNTR-Prime 中关于素阶数域的多项式乘法. 此外, 还提出一种改进的伪梅森数约减算法, 并将它应用在伪梅森数不完整 NTT 中. 它在软件实现方面比 Barrett 约减快 2.6%, 在硬件实现方面比 Montgomery 约减和 Barrett 约减快 2–6 倍. 最后, 提供 CNTR-Prime 的 C 语言实现, 并与其他同类方案进行全面对比. 结果表明, 与 SNTRU-Prime 相比, CNTR-Prime 在安全强度、带宽和实现效率上有优势, 其中 CNTR-Prime-761 的经典和量子安全强度都比 SNTRU-Prime-761 的高 19 bit, 密文尺寸降低 8.3%, 密钥生成算法、密钥封装算法和解封装算法分别快 25.3 倍、10.8 倍和 2.0 倍. 实际上, CNTR-Prime-653 的经典和量子安全强度已可与 SNTRU-Prime-761 相媲美, 且 CNTR-Prime-653 的带宽降低 13.8%, 密钥生成算法、密钥封装算法和解封装算法分别快 33.9 倍、12.6 倍和 2.3 倍. 所提工作可为后续同类型的格密码方案的设计、分析和优化实现提供重要参考.

关键词: 格密码; 后量子密码; 数论研究单元 (NTRU); 素阶数域; 密钥封装方案; 数论变换; 模约减; 软件实现

中图法分类号: TP309

中文引用格式: 梁志闯, 赵旭阳, 方博越, 赵运磊. 素阶数域上的高效紧凑NTRU密钥封装方案. 软件学报, 2025, 36(2): 747–775. http://www.jos.org.cn/1000-9825/7161.htm

英文引用格式: Liang ZC, Zhao XY, Fang BY, Zhao YL. Efficient and Compact NTRU-based Key Encapsulation Mechanism in Large-Galois-group Prime-degree Prime-ideal Number Field. Ruan Jian Xue Bao/Journal of Software, 2025, 36(2): 747–775 (in Chinese). http://www.jos.org.cn/1000-9825/7161.htm

Efficient and Compact NTRU-based Key Encapsulation Mechanism in Large-Galois-group Prime-degree Prime-ideal Number Field

LIANG Zhi-Chuang[1], ZHAO Xu-Yang[1], FANG Bo-Yue[1], ZHAO Yun-Lei[1,2]

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

[2](State Key Laboratory of Cryptology, Beijing 100036, China)

Abstract: Constructing post-quantum key encapsulation mechanisms based on Lattice (especially NTRU Lattice) is one of the popular research fields in Lattice-based cryptography. Commonly, most Lattice-based cryptographic schemes are constructed over cyclotomic rings,

* 基金项目: 国家自然科学基金 (61877011); 国家重点研发计划 (2022YFB2701600); 上海市科学技术发展基金 (21DZ2200500); 山东省重点研发计划 (2017CXG0701, 2018CXGC0701)
收稿时间: 2023-10-19; 修改时间: 2023-12-19; 采用时间: 2024-01-08; jos 在线出版时间: 2024-07-10
CNKI 网络首发时间: 2024-07-15

which, however, are vulnerable to some attacks due to their abundant algebraic structures. An optional and more secure underlying algebraic structure is the large-Galois-group prime-degree prime-ideal number field. NTRU-Prime is an excellent NTRU-based key encapsulation mechanism over the large-Galois-group prime-degree prime-ideal number field and has been widely adopted as the default in the OpenSSH standard. This study aims to construct a key encapsulation mechanism over the same algebraic structure but with better performance than NTRU-Prime. Firstly, this work studies the security risks of cyclotomic rings, especially the attacks on quadratic power cyclotomic rings, and demonstrates the security advantages of a large-Galois-group prime-degree prime-ideal number field in resisting these attacks. Next, an NTRU-based key encapsulation mechanism named CNTR-Prime over a large-Galois-group prime-degree prime-ideal number field is proposed, along with the corresponding detailed analysis and parameter sets. Then, a pseudo-Mersenne incomplete number theoretic transform (NTT) is provided, which can compute polynomial multiplication efficiently over a large-Galois-group prime-degree prime-ideal number field. In addition, an improved pseudo-Mersenne modular reduction algorithm is proposed, which is utilized in pseudo-Mersenne incomplete NTT. It is faster than Barrett reduction by 2.6% in software implementation and is 2 to 6 times faster than both Montgomery reduction and Barrett reduction in hardware implementation. Finally, a C-language implementation of CNTR-Prime is presented. When compared to SNTRU-Prime, CNTR-Prime has advantages in security, bandwidth, and implementation efficiency. For example, CNTR-Prime-761 has an 8.3% smaller ciphertext size, and its security is strengthened by 19 bits for both classical and quantum security. CNTR-Prime-761 is faster in key generation, encapsulation, and decapsulation algorithms by 25.3×, 10.8×, and 2.0×, respectively. The classical and quantum security of CNTR-Prime-653 is already comparable to that of SNTRU-Prime-761, with a 13.8% reduction in bandwidth, and it is faster in key generation, encapsulation, and decapsulation by 33.9×, 12.6×, and 2.3×, respectively. This study provides an important reference for subsequent research, analysis, and optimization of similar Lattice-based cryptographic schemes.

Key words: Lattice-based cryptography; post-quantum cryptography; number theory research unit (NTRU); large-Galois-group prime-degree prime-ideal number field (LPPNF); key encapsulation mechanism; number theoretic transform; modular reduction; software implementation

现有公钥密码体制多数基于大整数分解和 (椭圆曲线) 离散对数问题等经典的困难问题. 随着量子计算的飞速发展, 量子敌手能够利用 Shor 量子算法[1]在多项式时间内求解这些困难问题, 这将对现有公钥密码体制产生颠覆性的威胁. 在此背景下, 密码界开始研究能够抵抗量子攻击的密码学——后量子密码学 (post-quantum cryptography, PQC).

在 2016 年美国国家标准技术研究所 (national institute of standards and technology, NIST) 举办的后量子密码方案征集项目[2]和 2019 年中国密码学会举办的后量子密码算法竞赛[3]中, 主要对公钥加密方案 (public key encryption, PKE)、密钥封装方案 (key encapsulation mechanism, KEM) 和数字签名这 3 种密钥原语进行标准征集. 提交的后量子密码提案主要分为: 基于格的、基于编码的、基于哈希的、基于多变量的和基于同源的, 其中基于格的后量子密码因其在安全性、带宽和实现效率上表现均衡而备受青睐. 基于格设计 PKE 和 KEM 主要分为以下 2 种技术路线: 第 1 种是基于{R, M}LWE/LWR 问题[4–7]; 第 2 种是基于 NTRU 问题[8].

Hoffstein 等人[8]于 1998 年正式提出 NTRU 加密方案, 如今 NTRU 已经成为构造各种密码原语的基础构件之一. 特别是 NIST 后量子密码征集项目中, 基于 NTRU 格构造的方案有: 拟标准化的数字签名方案之一的 Falcon; 第 3 轮中 4 个决赛 KEM 方案之一的 NTRU KEM (包含 NTRU-HRSS 和 NTRUEncrypt)[9]和 5 个候选方案之一的 NTRU-Prime (包含 SNTRU-Prime 和 NTRU-LPRime)[10]. 尽管 NIST 最终选择基于模格的 Kyber[11]作为唯一的标准化 KEM (官方名字为 ML-KEM), 而基于 NTRU 格的 NTRU KEM 和 NTRU-Prime 均落选, 但事实上, 基于 NTRU 格的方案早已开启了标准化和商业化之路. 早在 2008 年, IEEE Std 1363.1 标准已经将包括 NTRUEncrypt 在内的格密码方案纳入其中[12]. 在 2011 年, X9.98 标准采用了 NTRUEncrypt 并应用在金融服务中[13]. 在 2016 年, Schanck 等人[14]便尝试在 Tor 协议产生临时 ECDH 密钥之余, 使用 NTRUEncrypt 产生额外的临时密钥, 以实现协议在量子时代的前向安全性. 欧盟开启的 PQCRYPTO 项目 (Horizon 2020 ICT-645622) 也考虑 NTRU 类型方案[15]. 国际标准 OpenSSH 在 2022 年 4 月公布的 9.0 版本之后的版本中采用了 NTRU-Prime 结合 X25519 ECDH 的混合模式, 用于抵抗“先捕获后解密”攻击[16].

基于 NTRU 格设计 KEM 是目前主流的技术路线之一. 这主要得益于 NTRU 自提出至今从没实质上被攻破; 有关 NTRU 的核心专利也于 2017 年过期, 便于开展商业化. 但是以往的基于 NTRU 格的 KEM 难以在安全性、带宽、错误率和实现效率上达到性能均衡, 这也是基于 NTRU 格的 KEM 难以普及的原因之一. 学术界对此进行

了深入研究, 并基于分圆环设计了多个 NTRU 类型 KEM, 力求在安全性、带宽、错误率和实现效率上达到均衡的性能, 比如 NTRU-A[17]、BAT[18]、NTTRU[19]、LTRU[20]和 CNTR/CTRU[21].

事实上, 现有的基于代数格 (如理想格、模格和 NTRU 格) 构造的密码方案多数使用分圆环作为底层代数结构, 其中常见的是 2 次幂分圆环, 因为它们的结构简单且允许使用高效的数论变换 (number theoretic transform, NTT) 计算多项式乘法. 但是, 由于分圆环饱含丰富的代数结构 (如存在大量子域、自同构和环同态), 导致基于分圆环构造的密码方案容易遭受相关攻击, 如文献 [22–26] 提出的攻击.

Bernstein 等人[10,27]指出, 在密码学的发展史上为密码方案去除不必要的代数结构是广泛共识. 例如, 小特征有限域上的椭圆曲线离散对数问题 (elliptic curve discrete logarithms, ECDL) 比最小域上的 ECDL 更容易受到特殊的数学攻击, 因为敌手攻击后者时可用操作大幅减少 (特别是基于自同构攻击). 即使尚没有攻击对现有小特征有限域上的椭圆曲线方案产生实质性影响, 但将小特征有限域迁移到最小域能够防患于未然.

因此, 鉴于对分圆环安全性潜在的担忧, Bernstein 等人[10,27]提出一种具有“高安全性、素数阶、大 Galois 群和惰性模数 (high-security prime-degree large-Galois-group inert-modulus)”的底层代数结构, 即是: 多项式环 $\mathbb{Z}_q[x]/(x^n-x-1)$, 其中 n 为素数, 使得它的 Galois 群足够大且同构于阶数为 $n!$ 的置换群 S_n; q 为素数, 且使得 x^n-x-1 在 $\mathbb{Z}_q[x]$ 中不可约, 可知 (x^n-x-1) 构成一个素理想, 同时 $\mathbb{Z}_q[x]/(x^n-x-1)$ 同构于一个代数数域.

本文将这类大 Galois 群、素数阶、基于素理想的数域简称为素阶数域 (large-Galois-group prime-degree prime-ideal number field, LPPNF). Bernstein 等人[10,27]还推荐将现有底层代数结构从分圆环迁移到素阶数域, 以抵抗针对分圆环的已知和潜在的攻击, 使得方案具有更保守的安全性.

基于素阶数域设计各种格密码方案是一条安全可靠、高效实用、值得深入研究的格密码技术路线. 基于素阶数域构造的 SNTRU-Prime 方案便是在这样的背景下被提出的[27]. 况且, SNTRU-Prime 早已经在 OpenSSH 中默认应用, 已经成为事实上的 KEM 标准. 鉴于 SNTRU-Prime 是一个素阶数域上 NTRU 类型的出色的 KEM, 能否基于素阶数域设计一个比 SNTRU-Prime 性能更优的 KEM 是目前学术界一个高关注度的问题.

注意到基于分圆环构造的 CNTR[21]是首个连接高维 NTRU 格密码和底层格编码, 且在安全性、带宽、错误率和实现效率上性能均衡的 NTRU 类型 KEM. 基于 CNTR 框架构造素阶数域上 NTRU 类型 KEM 有望具备比事实标准 NTRU-Prime 更优的性能. 然而, 倘若将 CNTR 从分圆环直接拓展到素阶数域, 这将会带来以下几方面的变化: 参数集、多项式尺度化 E_8 格编码、错误率分析、多项式乘法, 因为原先关于这几方面的分析均是依赖于 CNTR 使用的分圆环, 但不能直接应用在素阶数域. 所以, 基于 CNTR 框架构造素阶数域上 NTRU 类型 KEM 需要重新选择性能均衡的参数集、重新设计多项式尺度化 E_8 格编码以及重新进行错误率分析. 至于实现效率, 它一直是基于素阶数域的格密码方案面临的障碍之一, 因为在计算素阶数域上多项式乘法方面缺乏高效的算法, 毕竟素阶数域并不像 2 次幂分圆环一样能够直接应用 NTT. 后来文献 [28] 表明, 素阶数域也能通过拓展多项式环后使用 NTT 计算多项式乘法, 其性能接近分圆环上多项式乘法性能.

这便构成了本文的研究动机. 为此, 本文提出了 CNTR 在素阶数域上的变体方案 CNTR-Prime. 与 SNTRU-Prime 相比, CNTR-Prime 在安全性、带宽、实现效率方面有优势. 同时, CNTR-Prime 解决了直接将 CNTR 从分圆环拓展到素阶数域导致的参数选取、多项式尺度化 E_8 格编码、错误率分析和实现效率等问题. 特别地, 针对实现效率, 本文提出了伪梅森数不完整 NTT 计算素阶数域上多项式乘法.

详细地, 本文的贡献总结为以下 5 点.

1) 本文梳理了格密码常见底层分圆环存在的安全隐患, 特别是针对 2 次幂分圆环的系列攻击, 同时简要总结了素阶数域在抵抗相关攻击的安全优势, 特别子域攻击和自同构攻击.

2) 本文提出了 CNTR[21]在素阶数域上的变体方案 CNTR-Prime, 它基于 NTRU 假设和 RLWR 假设. 本文还对 CNTR-Prime 进行了安全性分析和错误率分析, 同时给出了素阶数域上多项式乘积系数的具体形式, 并通过大量测试最终选取了性能均衡的 3 组参数集, 其中每组参数集都具有高安全性和可忽略的错误率. 本文推荐使用 CNTR-Prime-761.

3) 本文提出了高效的伪梅森数不完整 NTT, 它能够高效计算 CNTR-Prime 中 3 组参数集对应的素阶数域上

多项式乘法. 其中, 通过调整 FFT trick 顺序, 本文在素阶数域 $\mathbb{Z}_{4621}[x]/(x^{653}-x-1)$ 上的伪梅森数不完整 NTT 能够降低 33.3% 的本原单位根存储开销. 对于 CNTR-Prime 使用到的 3 种素阶数域, 伪梅森数不完整 NTT 比 4-way Toom-Cook 快了 13.2–28.8 倍.

4) 本文提出了一种改进的伪梅森数约减, 同时将它应用在本文的伪梅森数不完整 NTT 中. 它能利用伪梅森数独特的构造来进行高效的模约减, 并且支持有符号整型的输入输出, 不但输入范围更大, 而且无需纠正数值步骤, 适合格密码实现. 实验表明, 对于 CNTR-Prime 使用到的伪梅森数 16 777 153 和 33 550 337, 在软件实现方面, 改进的伪梅森数约减 (单轮) 比 Barrett 约减快了 2.6%; 在硬件上实现方面, 改进的伪梅森数约减 (单轮、双轮) 比 Montgomery 约减和 Barrett 约减快了 2–6 倍.

5) 本文给出了 CNTR-Prime 的高效实现, 还将 CNTR-Prime 和其他方案进行全面对比. 结果表明, 与应用在国际标准 OpenSSH 的 SNTRU-Prime 相比, CNTR-Prime 的 3 组参数集在安全强度、带宽和实现效率上有优势, 其中 CNTR-Prime-761 的经典和量子安全强度都比 SNTRU-Prime-761 的高 19 bit, 密文尺寸降低 8.3%, 密钥生成算法、密钥封装算法和解封装算法分别快了 25.3 倍、10.8 倍和 2.0 倍. 实际上, CNTR-Prime-653 的经典和量子安全强度已可与 SNTRU-Prime-761 相媲美, 且 CNTR-Prime-653 的公钥尺寸降低 14.1%, 密文尺寸降低 13.5%, 总带宽降低 13.8%, 密钥生成算法、密钥封装算法和解封装算法分别快了 33.9 倍、12.6 倍和 2.3 倍. 与基于其他多项式环的同类方案相比, CNTR-Prime 在安全强度、带宽和错误率上性能均衡, 且底层代数结构具有更保守的安全性.

1 相关工作

基于 NTRU 格构造密码方案是近些年热门研究领域之一. Stehlé等人[29]率先基于 2 次幂分圆环设计基于 NTRU 格的 PKE, 并首次给出了严格的安全规约证明, 但是该方案由于公钥尺寸过大导致实用性较差. Jarvis 等人[30]将 NTRU 类型 PKE 推广至 Eisenstein 整数环 $\mathbb{Z}[\omega]/(x^n-1)$, 其中 $\omega=\exp(2\pi i/3)$, 并获得密钥尺寸和性能方面的优势. Hülsing 等人[31]将经典 NTRU 加密方案进行了优化, 并提升其尺寸和效率, 基于此提出的 NTRU-HRSS 是 NIST 后量子密码征集项目第 3 轮决赛方案之一. Bernstein 等人[27]基于代数结构更少的素阶数域构造基于 NTRU 格的方案 NTRU-Prime, 它曾经是 NIST 后量子密码征集项目第 3 轮候选方案之一.

Lyubashevsky 等人[19]基于三项分圆环 $\mathbb{Z}_{7681}[x]/(x^{768}-x^{384}+1)$ 构造了高效的 NTTRU 方案. Duman 等人[17]提出的 NTRU-A 将该框架推广至更一般的三项分圆环 $\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$. 然而, 上述 NTRU 方案均是基于传统 NTRU 框架构造, 并不能压缩密文. Fouque 等人[18]基于 2 次幂分圆环 $\mathbb{Z}_q[x]/(x^n+1)$ 和改进的 Babai 纠错码算法提出了一个名为 BAT 的 NTRU 类型 KEM, 但其主要针对 $n=512$ 和 1024 的参数集, 缺乏格密码常用的 $n=768$ 和 761 参数集. 另外, BAT 所基于的二元秘密的 RLWR 假设是比较新的问题, 依然需要更深入的研究. Liang 等人[21]基于三项分圆环 $\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$ 和尺度化 E_8 格编码给出了一个新的 NTRU 构造框架, 并设计了 CNTR 方案. BAT 和 CNTR 均能对密文进行压缩, 并利用纠错码来消除计算过程和压缩过程产生的错误. 梁志闯等人[20]提出了三项分圆环上无需纠错码便能压缩密文的 NTRU 类型密钥封装方案 LTRU. 但是这些 NTRU 类型密钥封装方案均是基于分圆环构造, 容易遭受针对分圆环的相关攻击.

2 预备知识

2.1 符号和定义

记 $\mathbb{Z}$ 为整数集, $\mathbb{R}$ 为实数集, n 和 q 为某些正整数. 记符号 $\mathbb{Z}_q=\mathbb{Z}/q\mathbb{Z}\cong\{0,1,\ldots,q-1\}$, $\mathbb{Z}_q$ 中可逆元素集合用 $\mathbb{Z}_q^\times$ 表示. 对于 $x\in\mathbb{R}$, 记 $\lfloor x\rceil$ 表示 x 的四舍五入的值. 本文使用多项式环 $\mathcal{R}:=\mathbb{Z}[x]/(x^n-x-1)$ 和 $\mathcal{R}_q:=\mathcal{R}/q\mathcal{R}=\mathbb{Z}_q[x]/(x^n-x-1)$, 它们中的元素是 n 维多项式且系数均在 $\mathbb{Z}$ 或 $\mathbb{Z}_q$ 中. 本文将这些多项式使用小写字母表示, 如 f 或者 $f(x)$, 那么, f 可以表示为幂级数形式 $f=\sum_{i=0}^{n-1}f_ix^i$, 其中, $f_i\in\mathbb{Z}$ 或 $f_i\in\mathbb{Z}_q$. 一个函数 $\varepsilon:\mathbb{N}\to[0,1]$ 如果对任意的正数 c 和充分大的 λ 都成立 $\varepsilon(\lambda)<1/\lambda^c$, 则称 ε 是可忽略的. 本文使用第 m 个分圆多项式: 当 $m=2^{k+1}$, $k\geqslant 0$ 为 2 次幂时, 得到 2 次幂

分圆多项式 $\phi_m(x)=x^{2^k}+1$. 当 $m=2^k3^l$, $k\geqslant 1$, $l\geqslant 1$ 时, 得到三项分圆多项式 $\phi_m(x)=x^{2^k3^{l-1}}-x^{2^{k-1}3^{l-1}}+1$.

本文遵循文献 [21] 的符号系统. 对于正数 q, $r'=r \bmod^{\pm} q$ 表示 r 的绝对最小完全剩余, 即 r 落在 $\left[-\frac{q}{2},\frac{q}{2}\right)$ 中的代表元为 r'; $r'=r \bmod q$ 表示 r 的非负最小完全剩余, 即 r 落在 $[0,q)$ 中的代表元为 r'. 对于 $w\in\mathbb{R}$, 它的 ℓ_∞ 范数为 $\|w\|_{q,\infty}=\left|w \bmod^{\pm} q\right|$. 若 w 是 n 维向量, 那么它的 ℓ_2 范数为 $\|w\|_{q,2}=\sqrt{\|w_0\|_{q,\infty}^2+\ldots+\|w_{n-1}\|_{q,\infty}^2}$.

如果 D 是一个集合, $x\leftarrow \$D$ 表示从 D 中随机均匀选取 x. 如果 D 是一个概率分布, $x\leftarrow D$ 表示根据分布 D 选取 x. 以整数 η 为参数的中心二项分布 B_η 的定义如下: 从 $\{0,1\}^{2\eta}$ 中随机均匀采样 $(a_1,a_2,\ldots,a_\eta,b_1,b_2,\ldots,b_\eta)$, 输出 $\sum_{i=1}^{\eta}(a_i-b_i)$. 采样多项式 $f\leftarrow B_\eta$ 指的是根据分布 B_η 分别对 f 的每个系数进行采样.

2.2 密码原语

一个公钥加密方案 PKE 含有 3 个概率多项式时间 (probabilistic polynomial time, PPT) 算法: *KeyGen*, *Enc* 和 *Dec*, 以及明文空间 $\mathcal{M}$. 密钥生成算法 *KeyGen* 输出公私钥对 (pk,sk). 加密算法 *Enc* 以公钥 pk 和明文 $m\in\mathcal{M}$ 为输入, 然后输出密文 c. 本文在必要时会显式写出加密算法所使用的随机数 *coin*, 即 $Enc(pk,m;coin)$. 确定性的解密算法 *Dec* 以私钥 sk 和密文 c 为输入, 然后输出明文 $m\in\mathcal{M}$, 但解密失败时输出符号 $\perp$. 公钥加密方案 PKE 的解密错误率 δ 指的是 $E[\max_{m\in\mathcal{M}}\Pr[Dec(sk,Enc(pk,m))\neq m]]<\delta$, 其中期望的计算是基于 $(pk,sk)\leftarrow KeyGen$, 概率来自 *Enc* 所使用的随机数. 敌手 $\mathcal{A}$ 关于公钥加密方案 PKE 的不可区分意义下选择明文攻击 (indistinguishability under chosen-plaintext attacks, IND-CPA) 的优势定义为:

$$Adv_{\mathrm{PKE}}^{\text{IND-CPA}}(\mathcal{A})=\left|\Pr\left[b'=b:\begin{array}{c}(pk,sk)\leftarrow KeyGen();(m_0,m_1)\leftarrow\mathcal{A}(pk);\\ b\leftarrow\$\{0,1\};c*\leftarrow Enc(pk,m_b);b'\leftarrow\mathcal{A}(c*)\end{array}\right]-\frac{1}{2}\right|.$$

如果 $Adv_{\mathrm{PKE}}^{\text{IND-CPA}}(\mathcal{A})$ 是可忽略的, 则称公钥加密方案 PKE 是 IND-CPA 安全的.

一个密钥封装方案 KEM 含有 3 个概率多项式时间算法: *KeyGen*, *Encaps* 和 *Decaps*, 以及导出共享密钥空间 $\mathcal{K}$. 密钥生成算法 *KeyGen* 输出公私钥对 (pk,sk). 封装算法 *Encaps* 以公钥 pk 为输入, 然后输出密文 c 和共享密钥 $K\in\mathcal{K}$. 确定性的解封装算法 *Decaps* 以私钥 sk 和密文 c 为输入, 然后共享密钥 $K\in\mathcal{K}$, 但解封装失败时输出符号 $\perp$. 密钥封装方案 KEM 的解封装错误率 δ 指的是 $\Pr[Decaps(sk,ct)\neq K:(ct,K)\leftarrow Encaps(pk)]<\delta$, 其中, 概率来自 $(pk,sk)\leftarrow KeyGen$ 以及 *Encaps* 所使用的随机数. 敌手 $\mathcal{A}$ 关于密钥封装方案 KEM 的不可区分意义下选择密文攻击 (indistinguishability under chosen-ciphertext attacks, IND-CCA) 的优势定义为:

$$Adv_{\mathrm{KEM}}^{\text{IND-CCA}}(\mathcal{A})=\left|\Pr\left[b'=b:\begin{array}{c}(pk,sk)\leftarrow KeyGen();b\leftarrow\$\{0,1\};\\ (c*,K_0^*)\leftarrow Encaps(pk);K_1^*\leftarrow\$\mathcal{K};b'\leftarrow\mathcal{A}^{Decaps(\cdot)}(pk,c*,K_b^*)\end{array}\right]-\frac{1}{2}\right|.$$

如果 $Adv_{\mathrm{KEM}}^{\text{IND-CCA}}(\mathcal{A})$ 是可忽略的, 则称密钥封装方案 KEM 是 IND-CCA 安全的.

2.3 困难性假设

本文方案主要基于 NTRU 困难性假设[8]以及 RLWR 困难性假设[5], 它们的定义如下所示.

定义 1. NTRU 假设[8]. 记 R 为多项式环, Ψ 为 R 上的分布. 根据 Ψ 采样多项式 f 和 g 且需要 f 在 R 中可逆, 并记 $h=g/f$. 判定型 NTRU 问题指的是区分 h 与 R 中随机均匀元素. 判定型 NTRU 问题是困难的, 指的是任意概率多项式时间敌手 $\mathcal{A}$ 的优势 $Adv_{R,\Psi}^{\mathrm{NTRU}}(\mathcal{A})$ 是可忽略的, 其中,

$$Adv_{R,\Psi}^{\mathrm{NTRU}}(\mathcal{A})=\left|\Pr\left[b'=1:\begin{array}{c}f,g\leftarrow\Psi;f^{-1}\in R;\\ h=g/f\in R;b'\leftarrow\mathcal{A}(h)\end{array}\right]-\Pr\left[b'=1:\begin{array}{c}h\leftarrow\$R;\\ b'\leftarrow\mathcal{A}(h)\end{array}\right]\right|.$$

定义 2. RLWR 假设[5]. 记 $q>p\geqslant 2$ 为正整数. 记 Ψ 为多项式环 R 上的分布. 记 $R_q=R/qR$ 和 $R_p=R/pR$ 为商环. 判定型 RLWR 问题指的是区分二元组 $(h,c)\leftarrow\$R_q\times R_p$ 和二元组 $(h,c)\in R_q\times R_p$, 其中, $h\leftarrow\$R_q$, $r\leftarrow\Psi$, $c=\left\lfloor\frac{p}{q}hr\right\rceil \bmod p$. 判定型 RLWR 问题是困难的, 指的是任意概率多项式时间敌手 $\mathcal{A}$ 的优势 $Adv_{R,\Psi}^{\mathrm{RLWR}}(\mathcal{A})$ 是可忽略的, 其中,

$$Adv_{R,\Psi}^{\mathrm{RLWR}}(\mathcal{A})=\left|\Pr\left[b'=1:\begin{array}{c}h\leftarrow\$R_q;r\leftarrow\Psi;\\ c=\left\lfloor\frac{p}{q}hr\right\rceil \bmod p;b'\leftarrow\mathcal{A}(h,c)\end{array}\right]-\Pr\left[b'=1:\begin{array}{c}h\leftarrow\$R_q;c\leftarrow\$R_p\\ b'\leftarrow\mathcal{A}(h,c)\end{array}\right]\right|.$$

2.4 尺度化 E_8 格编码

本文方案使用与文献 [21] 相同的尺度化 E_8 格编码. 遵循文献 [21] 的描述, 尺度化 E_8 格是由 8 维的扩展汉明码 $H_8=\left\{\mathbf{c}\in\{0,1\}^8|\mathbf{c}=\mathbf{zH} \bmod 2,\mathbf{z}\in\{0,1\}^4\right\}$ 构造, 能够满足 $\mathbb{R}^8$ 中最优密度装球方式, 以此构造的编码算法具有最优的纠错能力, 并且相关的编码算法和解码算法均能通过简单且常数时间的算术实现. 记 $\mathbf{c}=(0,1,0,1,0,1,0,1)$ 为 $\mathbf{H}$ 的最后一行向量, $C=\left\{(x_1,x_1,x_2,x_2,x_3,x_3,x_4,x_4)\in\{0,1\}^8\middle|\sum x_i\equiv 0 \bmod 2\right\}$ 张成 $\mathbf{H}$ 剩下的 3 行向量. 尺度化 E_8 格的构造为 $E_8:=\lambda\cdot[C\cup(C+\mathbf{c})]$, 其中 $\lambda\in\mathbb{R}^+$ 是尺度化因子. 尺度化 E_8 格的编码算法 $Encode_{E_8}$ 以 4 位的比特串 $\mathbf{k}$ 为输入, 然后通过 $\lambda\cdot(\mathbf{kH} \bmod 2)$ 计算得到尺度化 E_8 格的格点. 尺度化 E_8 格的解码算法 $Decode_{E_8}$ 以 8 维向量 $\mathbf{x}$ 为输入, 通过求解尺度化 E_8 格上的最近向量问题 (closest vector problem, CVP) 并转化为 4 位的比特串 $\mathbf{k}$, 以此作为输出. 尺度化 E_8 格的编码算法和解码算法的详细伪代码见文献 [21].

2.5 数论变换

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. 使用 NTT 计算多项式乘法 $h=f\cdot g$ 的步骤为 $h=INTT(NTT(f)\circ NTT(g))$, 其中 NTT 为正向变换, $INTT$ 为逆向变换, "$\circ$"为点乘. 本文沿用文献 [32] 的符号和术语来描述多项式乘法和 NTT. 文献 [32] 概括了一种计算 NTT 的快速算法 FFT trick. 本质上 FFT trick 是基于多项式环形式的中国剩余定理 (Chinese remainder theorem, CRT), 即存在 CRT 同构 $\mathbb{Z}_q[x]/(g_1g_2\cdots g_k)\cong\prod_{i=1}^{k}\mathbb{Z}_q[x]/(g_i)$ 以及 $f\mapsto(f \bmod g_1,f \bmod g_2,\ldots,f \bmod g_k)$, 其中 $g_1,g_2,\ldots,g_k$ 为两两互素的多项式. 记 ζ 是 $\mathbb{Z}_q$ 中可逆元素. 对于 $N\geqslant 2$ 的情况, 经典的基-N FFT trick 有以下同构: $\mathbb{Z}_q[x]/(x^{Nm}-\zeta^N)\cong\prod_{i=0}^{N-1}\mathbb{Z}_q[x]/(x^m-\rho^i\zeta)$, 其中 ρ 为 N 次单位根, 正向 FFT trick 可使用 CT (Cooley-Tukey) 蝴蝶操作, 而逆向 FFT trick 可使用 GS (Gentleman-Sande) 蝴蝶操作. 当 $\zeta^N=1$ 时得到的 NTT 称为循环卷积 NTT. 当 $m=1$ 时得到的 NTT 称为完整的 NTT; 否则称为不完整的 NTT.

2.6 Karatsuba 算法

定义 3. 1-迭代 Karatsuba 算法[33]. 记 a,b,c,d 为 4 个数或者多项式. 1-迭代 Karatsuba 算法通过以下方式计算 $t_1=ac$, $t_2=ad+bc$, $t_3=bd$: 首先计算 t_1 和 t_3, 然后计算 $t_2=(a+b)(c+d)-t_1-t_3$. 它能够将 4 个乘法减少至 3 个乘法.

3 底层代数结构的分析

本节将基于文献 [10,27] 梳理分圆环存在的安全隐患, 包括分圆环存在的丰富代数结构和针对分圆环的系列攻击, 同时介绍素阶数域在抵抗子域攻击和自同构攻击等相关攻击的安全优势.

3.1 分圆环的安全隐患

分圆环是代数格密码方案常用的底层代数结构, 例如基于模格的 Kyber[11]使用 $\mathbb{Z}_{3329}[x]/(x^{256}+1)$, 基于理想格的 NewHope[34]使用 $\mathbb{Z}_{12289}[x]/(x^{1024}+1)$. 基于分圆环构造的密码方案在安全性方面并非坚不可摧. 依赖于 2 次幂分圆环的 Smart-Vercauteren 全同态加密方案[22]已经遭受了多项式时间的量子攻击和亚指数时间的经典攻击[10,27]. 事实上, 分圆环以下的 3 点内在特性导致它易受相关攻击: 第 1 点是分圆环具有丰富的子域; 第 2 点是分圆环具有小 Galois 群; 第 3 点是分圆环具有大量的环同态.

1) CGS14 攻击和 BS16 攻击

Campbell 等人[23]提出一个名为"Soliloquy"的基于分圆环的理想格密码系统, 并提出了对该密码系统的一种有效的量子攻击方法. 本文记它为 Campbell-Groves-Shepherd 攻击, 简称 CGS14 攻击. 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, 它们跟主理想问题密切相关. 在这些方案中, 接收者所持有的公钥能够生成环上的主理想. 接收者的私钥是该主理想上的短生成元. 文献 [27] 指出, 这种类型的密钥恢复问题多见于同态加密方案和多线性映射系统中.

CGS14 攻击中的一个关键步骤可视为"log-unit 格"中的解码问题. 单位 (unit) 指的是交换环中具有乘法逆元的非零元素. 为保证密码系统的安全性, 其所依赖的底层解码问题理应在指数时间才能求解成功. 但对于分圆环的

情况, 存在更有效的算法寻找 log-unit 格中一个非常短的基, 并且这个短基包含了各种分圆 unit 的对数, 这使得此时解码问题变得更加容易. 文献 [27] 给出了一个有关分圆环的 unit 简单的例子. 在 2 次幂分圆环 $\mathbb{Z}[x]/(x^{1024}+1)$ 中能够容易找到一个 unit: $(1-x^3)/(1-x)$, 这是因为 $(1-x^3)/(1-x)=1+x+x^2$ 和它的乘法逆元 $(1-x)/(1-x^3)=(1-x^{2049})/(1-x^3)=1+x^3+\ldots+x^{2046}$ 均在 $\mathbb{Z}[x]/(x^{1024}+1)$ 中.

Biasse 等人[24,35]在文献 [36] 和 CGS14 攻击思想的基础上提出了一种多项式时间量子算法用于求解任意次数的数域上的主理想问题, 并指出它能够直接导出一种针对依赖于主理想问题的密码方案的多项式时间量子攻击. 本文将 Biasse 等人提出的攻击称为 Biasse-Song 攻击, 简称 BS16 攻击. 事实上, BS16 攻击能够成功攻击 Smart-Vercauteren 全同态加密方案[22]、分圆环情形下的 Gentry 全同态加密方案[37]和分圆环情形下的 Garg-Gentry-Halevi 多线性映射[38], 恰是因为这些方案的安全性依赖于分圆环上的主理想问题.

2) CDW21 攻击

Cramer 等人[25]提出了针对理想格困难问题的多项式时间量子攻击算法, 本文记它为 Cramer-Ducas-Wesolowski 攻击. 简称 CDW21 攻击. 这种攻击的基础是求解最短向量问题 (short vector problem, SVP). 对于一般的环 $\mathbb{Z}[x]/(\phi(x))$, 其中 $\phi(x)$ 是 $\mathbb{Z}[x]$ 上首一不可约多项式, 给定任意非零理想, CDW21 攻击能够找到一个非零向量, 其长度与最短非零向量相差不超过 $2^{\tilde{O}(\sqrt{\deg\phi(x)})}$ 倍. 该攻击的效果比以往攻击的效果更好, 因为以往攻击最好的结果为 $2^{\tilde{O}(\deg\phi(x))}$ 倍.

对于分圆环来说, CDW21 攻击能够取得更好的效果, 它能够进一步求解出主理想的最短生成元. 具体而言, 它利用分圆环的代数结构, 找到输入理想附近的主理想, 然后应用原来的攻击找到该主理想的最短生成元. 后续它有助于恢复私钥或者完成解密过程.

3) 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一个子域, 那么在该子域上所依赖的可能是一个更简单的格问题, 并且在这个子域上该问题的解能够简单地用于构造原问题的解. 文献 [39] 指出, 对于大模数的情况, 子域攻击的优势更为明显. 例如, 多数同态加密方案 (如 LTV 和 YASHE) 的渐进安全性都依赖的假设为: 敌手的攻击需要指数时间, 而子域攻击允许敌手能够在亚指数时间完成攻击. 这种攻击在 Garg-Gentry-Halevi 多线性映射[38]上能够取得更为显著的效果.

另外, 具有大量子域的密码系统, 容易遭受子域攻击. 例如, 文献 [40] 提出了一种不需量子计算机辅助的拟多项式时间的子域攻击, 它利用多重二次环存在大量子域的特点, 在某个类型的多重二次环中成功寻找到理想的短生成元.

实际上, 大多数子域攻击都需要依赖于原始域 $\mathbb{Q}[x]/(\phi(x))$ 的大次数子域: 即次数远大于 1 但小于 $\deg\phi(x)$ 的子域. 每个子域的次数是原始域的次数的因子. 对于分圆域特别是 2 次幂分圆域的情况, 此时 $\phi(x)=x^{2^k}+1$.

可见 $\deg\phi(x)$ 具有大量的因子, 导致 2 次幂分圆域存在大量子域. 这也是基于 2 次幂分圆环容易遭受子域攻击的重要原因之一.

4) 自同构攻击和环同态攻击

文献 [27] 将多项式对应的 Galois 群定义为包含该多项式所有复数根的最小域的自同构群. 对于 2 次幂分圆多项式 $x^{2^k}+1$ 的情况, $\mathbb{Q}(\zeta)$ 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 $\mathbb{Q}(\zeta)$ 具有 2^k 个自同构, 且不难得到这些自同构的具体形式: $\zeta\mapsto\zeta^i$, $i\in\mathbb{Z}_{2^{k+1}}^{\times}$. 文献 [27] 建议为了使方案更安全, 不应该让敌手掌握太多有关方案的信息, 比如这些自同构. 因为敌手能够将这些自同构用于构建 unit, 包括分圆 unit.

另外, 代数格密码方案为了能够使用高效的 NTT 计算多项式乘法, 往往使用满足 $q\equiv 1 \bmod 2^{k+1}$ 的素数 q 作为分圆环的模数, 即 $\mathbb{Z}_q[x]/(x^{2^k}+1)$. 此时 $x^{2^k}+1$ 能够分解为 $\mathbb{Z}_q[x]$ 中的线性多项式, 同时存在从 $\mathbb{Z}_q[x]/(x^{2^k}+1)$ 到更小的非零环 (例如 $\mathbb{Z}_q[x]/(x^{2^{k-1}}-\zeta^{2i})$ 和 $\mathbb{Z}_q[x]/(x^{2^{k-2}}-\zeta^i)$) 的环同态. 文献 [41,42] 给出了基于环同态实施攻击某些方案的例子.

5) S-unit 攻击

unit 攻击的主要思想是借助环中的 unit 来寻找理想中的短生成元. 在实际操作中, unit 攻击是通过约减 unit 的模数来缩短生成元. S-unit 攻击是 unit 攻击的一般化. 有关 unit 攻击和 S-unit 攻击的更多细节见文献 [26]. 近些

年来的研究表明, 基于分圆环的格密码方案更容易遭受 S-unit 攻击.

实际上, unit 攻击和 S-unit 攻击早已出现在以往文献中, 只不过不同文献使用不同的名称和术语. Campbell 等人[23]提出的 CGS14 攻击便是研究了 unit 群在分圆结构中的应用. Biasse 等人[24,35]提出的 BS16 攻击便是与计算 unit 群和 S-unit 群有关. Pellet-Mary 等人[43]提出一种改进的算法并将其应用于 S-unit 攻击. 并且指出, 比起现存的方法, S-unit 攻击能够计算得到更短的向量, 但对于一个指数级大的近似因子仍需要指数时间.

随后, Bernstein 等人[26]进一步改进了 S-unit 攻击, 并指出 S-unit 格存在大量更短的向量且能够更有效地进行约减. 计算数域中元素的范数是 S-unit 攻击中核心任务之一, 计算范数的时间会影响攻击的效果. Bernstein[44]给出了一种针对某些 Abelian 数域的新型快速计算元素范数方法, 并对 2 次幂分圆域的情形利用其自同构作进一步的优化, 使得在 2 次幂分圆域上的范数计算速度比一般数域的要快 10 万倍[45]. 这极大地提高了针对 2 次幂分圆环的 S-unit 攻击的效率, 并显著提高了对相关方案的安全性的影响.

3.2 素阶数域的安全优势

本文使用的底层代数结构是素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$. CGS14 攻击、BS16 攻击、CDW21 攻击和 S-unit 攻击均在分圆环上取得了比一般多项式环上更显著的优势, 特别是 2 次幂分圆域上计算 S-unit 攻击某些范数速度比一般数域的要快 10 万倍, 但这些优势均不能扩展到素阶数域, 因为这些攻击依赖分圆环中的特定结构, 而素阶数域中不存在这些结构. 另外, 素阶数域还有以下几点优势.

1) 抵御子域攻击

由于子域的次数只能是原始域次数的因子, 当 n 为素数时, $\mathbb{Q}[x]/(x^n-x-1)$ 的子域显然只有 2 个: $\mathbb{Q}[x]$ 和整个域 $\mathbb{Q}[x]/(x^n-x-1)$. 缺乏有效的子域能够减少敌手的可用信息. 敌手无法通过将原始域上的问题规约为子域上更简单的问题来求解, 同时也能阻止文献 [40,46] 提出的子域攻击.

值得注意的是, 文献 [39] 提出了一个针对 NTRU 格的子域攻击. 当模数 q 较大时, 该子域攻击能够从 NTRU 格的某些稠密子格中有效地恢复私钥. 在文献 [39] 中, 该子域攻击比常见格攻击更有效的模数 q 的临界值称为疲劳点, 并且对于秘密值为 O(1) 时的疲劳点 q 值为 $q=n^{2.484+O(1)}$, 其中 n 是维度. 只要素阶数域选取远小于疲劳点的 q 值, 文献 [39] 的子域攻击便无效. 值得一提的是, 文献 [39] 同样指出它的攻击并不会影响 NIST 后量子密码方案征集项目的所有 NTRU 类型方案 (包括 SNTRU-Prime) 的安全性; 相反, 文献 [39] 给出的疲劳点恰能说明这些小模数的 NTRU 类型方案不存在这样潜在的安全漏洞.

2) 抵御自同构攻击

多项式 x^n-x-1 对应的 Galois 群足够大, 因为它的 Galois 群同构于阶数为 $n!$ 的置换群 S_n. 这远远大于相同次数的分圆环的 Galois 群 (它的阶只有 n). 文献 [27] 指出, 拥有一个更大的 Galois 群意味着 x^n-x-1 在任何合理次数的域中至多有少量的根. 这能够消除了所有已知的使用大量自同构执行有效计算的攻击.

3) 抵御环同态攻击

本文使用的素数 q 满足 x^n-x-1 在 $\mathbb{Z}_q[x]$ 中不可约, 使得 x^n-x-1 无法分解为 $\mathbb{Z}_q[x]$ 中的线性多项式, 也不存在从 $\mathbb{Z}_q[x]/(x^n-x-1)$ 到更小的非零环的环同态. 显然, 不存在环同态的素阶数域能够避免文献 [41,42] 提出的环同态攻击. 尽管在某些密码方案 (如同态加密) 中, 模转换 (modulus switching) 技术能够弱化原始模数 q 的影响, 但是使用模转换技术将 q 转换为另一个模数时, 不可避免地引入额外的噪音, 这明显大幅度降低了攻击算法的性能, 同时也增加了攻击的难度.

3.3 小 结

素阶数域和分圆环在抵御相关攻击的能力比较见表 1. 综合来看, 分圆环在结构上存在的安全隐患使得它容易遭受相关攻击, 导致基于分圆环的密码方案的实际安全性偏低. 不过, 这些攻击对素阶数域的影响极为有限. 相比于分圆环, 素阶数域可以使敌手的可用操作 (特别是自同构和环同态) 大幅减少, 能够显著增加敌手的攻击难度. 这可以增强基于素阶数域的密码方案抵抗自同构攻击和子域攻击等系列攻击的能力, 大幅度提高方案的安全性, 使得方案具有更保守、更长久的安全保障.

表 1 抵御攻击能力的对比

攻击类型	素阶数域	分圆环
CGS14攻击	●	○
BS16攻击	●	○
CDW21攻击	●	○
S-unit攻击	●	○
子域攻击	✔	○
自同构攻击	✔	○
环同态攻击	✔	○

注: 抵御攻击能力的分类为: ●表示抵御能力强, ○表示抵御能力弱, ✔表示完全抵御

4 本文方案设计与分析

本节将介绍本文提出的基于素阶数域构造的 CNTR-Prime 方案. 它包括了一个 IND-CPA 安全的公钥加密方案, 记为 CNTR-Prime.PKE; 以及一个 IND-CCA 安全的密钥封装方案, 记为 CNTR-Prime.KEM.

4.1 公钥加密方案

CNTR-Prime 的公钥加密方案如算法 1–算法 3 所示. 素阶数域 $\mathcal{R}_q=\mathbb{Z}_q[x]/(x^n-x-1)$ 中的 n 和 q 均为素数, 且 x^n-x-1 在 $\mathbb{Z}_q[x]$ 中不可分解. 记 q_2 是密文模数, 通常设为 2 的某次幂且 q_2 小于 q. 记 p 是明文空间模数, 且满足 $\gcd(q,p)=1$. 本文重点关注 q 是一个奇素数和 $p=2$ 的情况. 设 Ψ_1 和 Ψ_2 是 $\mathcal{R}$ 上的两个给定的分布. 为了简单起见, 下文中 f',g 取自 Ψ_1, r 取自 Ψ_2. 实际上, Ψ_1 和 Ψ_2 可以是不同的分布. Ψ_1 和 Ψ_2 两个分布的具体选择见表 2. 设 $\mathcal{M}=\{0,1\}^{n'}$ 表示明文空间, 其中 $m\in\mathcal{M}$ 可以被视为一个 n' 维多项式且每个系数属于 $\{0,1\}$.

表 2 CNTR-Prime 参数集

参数集	n	q	q_2	n'	(Ψ_1,Ψ_2)	$\lvert pk\rvert$	$\lvert ct\rvert$	B.W.	NTRU(C, Q)	RLWR(C, Q)	δ
CNTR-Prime-653	653	4621	2^{11}	320	(B_3,B_3)	994	898	1892	(152, 137)	(151, 136)	2^{-166}
CNTR-Prime-761	761	4591	2^{10}	376	(B_2,B_2)	1158	952	2110	(173, 157)	(172, 156)	2^{-167}
CNTR-Prime-1277	1277	7879	2^{10}	632	(B_2,B_2)	2067	1597	3664	(299, 271)	(297, 269)	2^{-279}

算法 1. 密钥生成算法 CNTR-Prime.PKE.KeyGen.

输入: 安全参数 1^κ;

输出: 公钥加密公私钥对 (pk,sk).

① $f',g\leftarrow\Psi_1$

② $f:=pf'+1$

③ $h:=g/f$ //在 $\mathcal{R}_q$ 中恒存在 f 的逆

④ return $(pk:=h,sk:=f)$

算法 2. 加密算法 CNTR-Prime.PKE.Enc.

输入: 公钥 pk, 明文 m;

输出: 密文 c.

① $r\leftarrow\Psi_2$

② $\sigma:=hr$

③ $c := \left\lfloor \frac{q_2}{q}(\sigma + \text{PolyEncode}(m)) \right\rceil \bmod q_2$

④ return c

算法 3. 解密算法 CNTR-Prime.PKE.Dec.

输入: 私钥 sk, 密文 c;

输出: 明文 m.

① $m' := cf \bmod^{\pm} q_2$

② $m := \text{PolyDecode}(m')$

③ return m

在算法 1 的密钥生成算法中, 一般情况下需要判断 f 的可逆性, 若不可逆则需要多次选取 f' 并计算 f, 直到 f 可逆. 不过在本文中, $\mathcal{R}_q$ 构成一个域, 而且 f 恒为非零元, 所以 f 在 $\mathcal{R}_q$ 中恒存在它的逆. 于是, 算法 1 第 3 行计算 $h = g/f$ 完全可行.

CNTR-Prime 的密文压缩操作出现在算法 2 的第 3 行. 在数学形式上它们的每个分量可以写为 $y = \left\lfloor \frac{q_2}{q}x \right\rceil$, $x \in \mathbb{Z}_q$. 只有当 $\lceil \log(q_2) \rceil < \lceil \log(q) \rceil$ 时, 密钥压缩才有意义, 因为此时 y 的位宽比 x 的更小. 压缩参数 q_2 可根据压缩需求进行调整, q_2 越小, 压缩程度越高, 密文尺寸越小. CNTR-Prime 的 3 组参数集中 q_2 的值分别为 2^{11}、2^{10} 和 2^{10}, 而它们的 q 均为 13 bit 的数值, 故 3 组参数集的密文尺寸分别能够降低 164、285 和 479 字节, 降低的比例分别为 15.4%、23.0% 和 23.0%.

4.2 多项式尺度化 E_8 格编码

下面将文献 [21] 的多项式尺度化 E_8 格编码算法拓展到素阶数域上. 算法 2 用到的多项式编码算法 PolyEncode 见算法 4, 算法 3 用到的多项式解码算法 PolyDecode 见算法 5. 值得注意的是, 在算法 4 的多项式编码算法中使用到的尺度化 E_8 格 (记为 E_8' 格) 是基于尺度化因子 $\frac{q}{2}$ 构造, 即 $E_8' := \frac{q}{2} \cdot [C \cup (C + \mathbf{c})]$. 在算法 5 的多项式解码算法中使用到的尺度化 E_8 格 (记为 E_8'' 格) 是基于尺度化因子 $\frac{q_2}{2}$ 构造, 即 $E_8'' := \frac{q_2}{2} \cdot [C \cup (C + \mathbf{c})]$.

算法 4. 多项式编码算法 PolyEncode.

输入: 明文 $m := \sum_{i=0}^{n'-1} m_i x^i \in \mathcal{M}$;

输出: 编码后的多项式 $v := \sum_{i=0}^{n-1} v_i x^i$.

① for $i = 0, \ldots, n'/4 - 1$ do

② $\quad \mathbf{k}_i := (m_{4i}, m_{4i+1}, m_{4i+2}, m_{4i+3}) \in \{0,1\}^4$

③ $\quad (v_{8i}, v_{8i+1}, \ldots, v_{8i+7}) := Encode_{E_8'}(\mathbf{k}_i)$

④ end for

⑤ for $i = 2n', \ldots, n-1$ do

⑥ $\quad v_i := 0$

⑦ end for

⑧ $v := \sum_{i=0}^{n-1} v_i x^i$

⑨ return v

算法 5. 多项式解码算法 PolyDecode.

输入: 多项式 $v := \sum_{i=0}^{n-1} v_i x^i$;

输出: 明文 m.

① for $i = 0, \ldots, n'/4 - 1$ do

② $\quad \mathbf{x}_i := (v_{8i}, v_{8i+1}, \ldots, v_{8i+7})$

③ $\quad (m_{4i}, m_{4i+1}, m_{4i+2}, m_{4i+3}) := Decode_{E_8''}(\mathbf{x}_i)$

④ end for

⑤ $m := \sum_{i=0}^{n'-1} m_i x^i \in \mathcal{M}$

⑥ return m

注意到, 在算法 4 的第 5–7 行中, 需要对多项式 v 的剩下的系数赋值为 0. 下面给出解释. 尽管计算机存储信息的最小单位是 bit, 但对于现实的密码方案来说, 计算机处理数据的基本单位是字节. 所以, 对于明文 m 来说, 它的存储和处理均是以字节为单位. 对于本文的 n' 个 bit 的明文来说, 这里的 n' 是 8 的整数倍. 另外, E_8' 格编码算法每次输入 m 的 4 bit 的信息, 而输出 v 的 8 个系数. 于是, 每个明文 m 生成 v 的系数共有 $2n'$ 个. 但是本文使用的 n 为奇素数, 每个明文 m 生成 v 的系数数量不足以构建一个完整的多项式 v. 为此, 本文将剩下的 $n - 2n'$ 个系数均赋值为 0. 但事实上, 由于这 $n - 2n'$ 个系数不包含明文的任何信息, 它们可以被赋值为任何的值, 但是为了计算方便同时为了不影响错误率, 在此选择将它们赋值为 0.

在算法 5 中, 由于多项式 v 只有前面的 $2n'$ 个系数包含明文 m 的信息, 所以只需要处理 $2n'$ 个系数. 解密方式为: 每次提取 v 的 8 个系数作为 E_8'' 格解码算法的输入, 它便会输出 4 bit. 将 $n'/4$ 组 4 bit 信息合并便得到明文 m.

4.3 密钥封装方案

与 CNTR[21]一样, 本文使用了文献 [47] 提出的 Fujisaki-Okamoto (FO) 转换的变体: $\mathrm{FO}^{\not\perp}_{ID(pk),m}$ 转换, 它在不降低安全性的前提下能够有效降低哈希函数的调用次数, 降低哈希运算的耗时, 从而能够提升方案的运行效率. 值得注意的是, 其他 FO 转换同样适用于本文方案. 本文通过 $\mathrm{FO}^{\not\perp}_{ID(pk),m}$ 转换构造 CNTR-Prime 的密钥封装方案, 记为 CNTR-Prime.KEM=(*KeyGen*, *Encaps*, *Decaps*). CNTR-Prime.KEM 的 3 个算法的伪代码见算法 6–算法 8.

算法 6. 密钥生成算法 CNTR-Prime.KEM.KeyGen.

输入: 安全参数 1^κ;

输出: 密钥封装公私钥对 (pk', sk').

① $(pk, sk) \leftarrow$ CNTR-Prime.PKE.KeyGen(1^κ)

② $z \leftarrow \$\{0,1\}^\iota$

③ return $(pk' := pk, sk' := (sk, z))$

算法 7. 封装算法 CNTR-Prime.KEM.Encaps.

输入: 公钥 pk';

输出: 密文 c, 共享密钥 K.

① $m \leftarrow \$\mathcal{M}$

② $(K, coin) := \mathcal{H}(ID(pk), m)$

③ $c :=$ CNTR-Prime.PKE.Enc($pk, m; coin$)

④ return (c,K)

算法 8. 解封装算法 CNTR-Prime.KEM.Decaps.

输入: 私钥 $sk':=(sk,z)$, 密文 c;
输出: 共享密钥 K.

① $m':=\text{CNTR-Prime.PKE.Dec}(sk,c)$
② $(K',coin'):=\mathcal{H}(ID(pk),m')$
③ $\tilde{K}:=\mathcal{H}_1(ID(pk),z,c)$
④ if $m'\neq\perp$ and $c=\text{CNTR-Prime.PKE.Enc}(pk,m';coin')$ then
⑤ return K'
⑥ else
⑦ return $\tilde{K}$
⑧ end if

设 ι,γ 为正整数. 在本文中为了增强安全性, 设置 $\iota=\gamma=256$. 记 $\mathcal{H}:\{0,1\}^*\to\mathcal{K}\times\mathcal{COINS}$ 为哈希函数, 其中 $\mathcal{K}$ 是 CNTR-Prime.KEM 导出的共享密钥空间, $\mathcal{COINS}$ 是 CNTR-Prime.PKE 的加密算法使用的随机数空间. 值得注意的是, 在这里显式描述了 CNTR-Prime.PKE 的加密算法使用的随机数. $\mathcal{H}(\cdot)$ 可分为 2 部分: 记 $\mathcal{H}_1(\cdot)$ 为 $\mathcal{H}(\cdot)$ 映射到 $\mathcal{K}$ 中的那部分输出, $\mathcal{H}_2(\cdot)$ 为 $\mathcal{H}(\cdot)$ 映射到 $\mathcal{COINS}$ 中的那部分输出. 在算法 7 的第 2 行中可先使用 $\mathcal{H}_2(\cdot)$ 生成加密所需的 $coin$, 在第 3 行计算得到密文 c 之后再使用 $\mathcal{H}_1(\cdot)$ 导出共享密钥 K. 设 $\mathcal{PK}$ 是 CNTR-Prime.PKE 的公钥空间. 设 $ID:\mathcal{PK}\to\{0,1\}^\gamma$ 为某个输出固定长度函数.

4.4 错误率分析

本节将计算 CNTR-Prime 的错误率, 同时还给出素阶数域上多项式乘积系数的具体形式, 它能用于计算错误率过程中多项式乘积的系数分布.

定理 1. CNTR-Prime 的正确性. 设 Ψ_1 和 Ψ_2 是环 $\mathcal{R}$ 上的分布. 记 n,n',q,q_2 是方案参数. 设 $f',g\leftarrow\Psi_1$, $r\leftarrow\Psi_2$. 设 $\varepsilon\leftarrow\chi$, 其中 χ 是环 $\mathcal{R}$ 上的分布, 且定义为: 采样 $h\leftarrow\$\mathcal{R}_q$, $r\leftarrow\Psi_2$, 输出 $\left\lfloor\left\lfloor\frac{q_2}{q}hr\right\rceil-\frac{q_2}{q}hr\right\rceil \bmod^{\pm} q_2$. 设 Err_i 为 $gr+\frac{q}{q_2}\varepsilon f$ 的第 i 个八元组. 记 $1-\delta=\Pr\left[\|Err_i\|_{q,2}<\frac{q}{2},1\leqslant i\leqslant\frac{n'}{4}\right]$, 那么, CNTR-Prime 的错误率是 δ.

本文定理 1 的证明跟文献 [21] 的定理 3 的证明基本相同, 唯一区别在于, 本文定理 1 只需要处理前 $\frac{n'}{4}$ 个八元组 Err_i 便能恢复出明文 m, 这是因为明文 m 的信息只编码在前 $\frac{n'}{4}$ 个八元组之中, 但文献 [21] 的定理 3 需要处理所有的八元组. 本文定理 1 的证明可参见文献 [21] 的定理 3 的证明.

值得注意的是, 由于多项式乘积的系数分布由被乘多项式的系数分布和系数个数决定, 所以还需提供素阶数域上多项式乘积系数的具体项个数. 对于 CNTR-Prime 中多项式 $f=\sum_{i=0}^{n-1}f_ix^i,g=\sum_{i=0}^{n-1}g_ix^i\in\mathcal{R}_q$, 记它们相乘的结果为 $h=\sum_{k=0}^{n-1}h_kx^k=f\cdot g\in\mathcal{R}_q$. 那么,

1) 对于 $k=0$, 有 $h_k=f_0g_0+\sum_{i=1}^{n-1}f_ig_{n-i}$;
2) 对于 $1\leqslant k\leqslant n-2$, 有 $h_k=\sum_{i=0}^{k}f_ig_{k-i}+\sum_{i=k}^{n-1}f_ig_{n-1+k-i}+\sum_{i=k+1}^{n-1}f_ig_{n+k-i}$;
3) 对于 $k=n-1$, 有 $h_k=\sum_{i=0}^{n-1}f_ig_{n-1-i}+f_{n-1}g_{n-1}$.

因此, 对于 $k=0$, h_k 共有 n 个形如 f_ig_j 的项; 对于 $1\leqslant k\leqslant n-2$, h_k 共有 $2n-k$ 个形如 f_ig_j 的项; 对于 $k=n-1$, h_k 共有 $n+1$ 个形如 f_ig_j 的项. 已知 f_i 和 g_j 的分布, 便可计算 h_k 的分布.

本文计算 CNTR-Prime 的错误率的方法论和 Python 脚本源自文献 [11,21]. 相关的错误率计算结果见表 2.

4.5 安全性分析

1) 安全规约

下面将基于 NTRU 困难性假设和 RLWR 困难性假设, 说明 CNTR-Prime.PKE 满足 IND-CPA 安全性.

定理 2. CNTR-Prime.PKE 的 IND-CPA 安全性. 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 $\mathcal{B}$ 和 $\mathcal{C}$, 使得 $Adv^{\text{IND-CPA}}_{\text{CNTR-Prime.PKE}}(\mathcal{A}) \leqslant Adv^{\text{NTRU}}_{\mathcal{R}_q,\Psi_1}(\mathcal{B}) + Adv^{\text{RLWR}}_{\mathcal{R},\Psi_2}(\mathcal{C})$.

定理 2 的证明跟文献 [21] 的定理 5 的证明相同, 详情参见文献 [21] 的定理 5 的证明. 由于 CNTR-Prime.KEM 是从 CNTR-Prime.PKE 通过文献 [47] 提出的 $\text{FO}^{\not\perp}_{ID(pk),m}$ 转换得到, 所以根据文献 [47] 的结论, 得到定理 3 中 CNTR-Prime.KEM 在经典随机预言机模型和量子随机预言机模型下的 IND-CCA 安全性.

定理 3. CNTR-Prime.KEM 的 IND-CCA 安全性[47]. 设 ℓ 是 $ID(pk)$ 的最小熵, 其中 $(pk, sk) \leftarrow$ CNTR-Prime.PKE.KeyGen. 对任意 (量子) 敌手 $\mathcal{A}$, 如果它最多可以进行 q_D 次的解封装查询, q_H 次的 (量子) 随机预言机查询, 那么存在 (量子) 敌手 $\mathcal{B}$, 它的运行时间与 $\mathcal{A}$ 的相当, 使得:

i) 在经典随机预言机模型下, 有:

$$Adv^{\text{IND-CCA}}_{\text{CNTR-Prime.KEM}}(\mathcal{A}) \leqslant 2\left(Adv^{\text{IND-CPA}}_{\text{CNTR-Prime.PKE}}(\mathcal{B}) + \frac{q_H+1}{|\mathcal{M}|}\right) + \frac{q_H}{2^t} + (q_D+q_H)\cdot\delta + \frac{1}{2^\ell}.$$

ii) 在量子随机预言机模型下, 记 $q_{HD} := q_D + q_H + 1$, 则有:

$$Adv^{\text{IND-CCA}}_{\text{CNTR-Prime.KEM}}(\mathcal{A}) \leqslant 2\sqrt{q_{HD}\cdot Adv^{\text{IND-CPA}}_{\text{CNTR-Prime.PKE}}(\mathcal{B})} + \frac{4q_{HD}}{\sqrt{|\mathcal{M}|}} + \frac{4(q_H+1)}{\sqrt{2^t}} + 16q^2_{HD}\delta + \frac{1}{2^\ell}.$$

2) 具体安全强度

下面将分析 CNTR-Prime 所依赖的 NTRU 和 RLWR 困难问题的安全强度. 事实上, 对于 CNTR-Prime 来说, 格攻击中的原始攻击是目前为止最有效的攻击手段. 原始攻击将求解原问题转化为构造一种整数嵌入格, 之后求解该格中唯一最短向量问题 (unique-short vector problem, u-SVP). NTRU 困难问题能转化为求解 NTRU 格中的 u-SVP 问题. RLWR 问题能转化为求解某个嵌入格中的 u-SVP 问题. 为求解 u-SVP 问题, BKZ 算法[48]是目前常用的格基约化算法. 给定格的一组基, BKZ 算法在运行过程中将格划分为若干个更低维度的子格, 其中这些子格的维度记为 b. 之后, 它会在这些 b 维格上逐一求解最短向量问题 SVP. 本文使用文献 [34] 提出的 core-SVP 方法论对 b 维格上求解 SVP 问题的复杂度进行保守的估算. 文献 [34] 指出, 对于 b 维格上 SVP 问题的求解器, 目前最优的经典算法和量子算法的复杂度分别为 $2^{0.292b}$ 和 $2^{0.265b}$. 文献 [34] 的 core-SVP 方法论将这些复杂度视为 b 维格上求解 SVP 问题的复杂度, 同时将基于此运行的 BKZ 算法的复杂度作为原始攻击的复杂度. 文献 [34] 的 core-SVP 方法论能够给 CNTR-Prime 提供保守的安全强度估计. 本文使用文献 [34] 提供的计算 core-SVP 方法论安全强度的 Python 脚本来计算本文方案的经典安全强度和量子安全强度, 相关的计算结果见表 2.

4.6 参数集

表 2 给出 CNTR-Prime 的 3 组参数集: CNTR-Prime-653、CNTR-Prime-761 和 CNTR-Prime-1277. 素阶数域 $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n - x - 1)$, 其中 n 是维度, q 是模数, 且 n 和 q 满足: $x^n - x - 1$ 在 $\mathbb{Z}_q[x]$ 不可约. q_2 为密文模数, 同时也是 CNTR-Prime 的 RLWR 模数. 本文使用小的 q 值, 满足 $q < n^{2.484+o(1)}$, 即 q 值小于文献 [39] 给出的 NTRU 格的疲劳点, 避免了文献 [39] 的子域攻击对 CNTR-Prime 的影响. 给定明文空间 $\mathcal{M} = \{0,1\}^{n'}$, 但是考虑到计算机处理数据的基本单位是字节并且需要尽可能大的明文空间, 同时 n' 也是密钥封装方案能够导出的共享密钥的最大比特数, 所以本文选取 $n' = 8\cdot\lfloor n/16\rfloor$. 当 $n = 653$、761 和 1277 时对应 $n' = 320$、376 和 632. 另外, 本文固定明文空间模数恒为 $p = 2$. Ψ_1 和 Ψ_2 是概率分布, 本文主要考虑分布 B_η, 即以整数 η 为参数的中心二项分布. $|pk|$ 为公钥尺寸, $|ct|$ 为密文尺寸, B.W.为带宽, 即 $|pk|+|ct|$, 它们均是以字节为单位. 本文考虑了 NTRU 攻击和 RLWR 攻击, 并且根据第 4.5 节描述的 core-SVP 方法论计算得到, 其中“C”和“Q”分别是在经典和量子情形下的安全强度, 它们的数值用 bit 表示. 最后一列的 δ 表示该参数集的错误率, 它的计算细节见第 4.4 节.

4.7 比较和讨论

1) 和 CNTR 的比较

CNTR-Prime 和 CNTR 在以下几方面存在显著差异.

i) 底层代数结构. CNTR-Prime 使用素阶数域 $\mathbb{Z}_q[x]/(x^n-x-1)$, 要求 n 和 q 为素数, 且 x^n-x-1 在 $\mathbb{Z}_q[x]$ 中不可约, 使得方案能够抵御子域攻击、自同构攻击等, 所以 CNTR-Prime 具有更保守的安全性. CNTR 使用三项分圆环 $\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$, 旨在使用高效的 NTT, 但是它易受分圆环系列攻击, 详情见第 3.1 节.

ii) 错误率的计算方式. 由于计算错误率时多项式乘积的系数分布由被乘多项式的系数分布和系数个数决定, CNTR-Prime 和 CNTR 采用不同底层代数结构导致了它们的错误率的计算方式不相同. 比如, CNTR 使用的三项分圆环的多项式乘积系数至多有 $3n/2$ 个形如 f_ig_j 的项, 而 CNTR-Prime 使用的素阶数域情况则更加复杂. 由于 CNTR 计算错误率的方法不能应用在 CNTR-Prime 中, 为此本文给出了素阶数域的多项式乘积系数的具体形式, 见第 4.4 节. 并可知, 素阶数域的多项式乘积系数至多有 $2n-1$ 个形如 f_ig_j 的项, 其中 n 是多项式维度, 这跟三项分圆环的情况完全不同. 随后, 本文基于素阶数域的多项式乘积系数的具体形式计算了 CNTR-Prime 的错误率, 同时给出了用于计算错误率的 Python 脚本, 它同样能够用来计算素阶数域上其他 KEM 的错误率.

iii) 多项式运算的快速算法. CNTR 能够直接使用高效的 (混合基) NTT 计算多项式乘法和求逆. 因为 CNTR-Prime 的环参数均为素数, 所以不能直接应用任何的 NTT 算法. 且在计算多项式求逆时 CNTR-Prime 也只能使用一般的求逆算法, 所以 CNTR-Prime 的密钥生成算法会更耗时. 为了加速 CNTR-Prime 的多项式乘法, 本文将原来的多项式环拓展到更大的 NTT 友好环, 再使用本文提出的伪梅森数不完整 NTT, 使得 CNTR-Prime 的实现效率优于 SNTRU-Prime, 甚至能够媲美基于分圆环的方案. 伪梅森数不完整 NTT 的更多细节将在第 5 节介绍.

2) 和其他 NTRU 类型 KEM 的比较

在此将给出 CNTR-Prime 和其他 NTRU 类型 KEM 在构造方式上的比较情况, 至于它们之间的性能比较和效率比较将集中在第 8.4 节和第 8.5 节中展示.

CNTR-Prime 和 SNTRU-Prime[27]使用素阶数域, 而 NTRU-HRSS[9]使用截断多项式环 $\mathbb{Z}_q[x]/(x^n-1)$, 其他 NTRU 类型 KEM 比如、NTRU-A[17]、BAT[18]、NTTRU[19]和 LTRU[20]使用分圆环 (旨在追求高效地实现).

CNTR-Prime 和 LTRU[20]使用的明文空间模数均为 $p=2$, 且都只出现在私钥 f 的计算过程中, 而不需要在生成公钥和加/解密过程中使用到. 但是 NTRU 类型 KEM 像 NTRU-HRSS、SNTRU-Prime、NTRU-A[17]和 NTTRU 均使用 $p=3$. 特别是 NTRU-HRSS, 由于它使用 2 次幂模数 q, 而不得不使用 $p=3$ 使得 p 和 q 互素. 更小的 p 产生的错误率更低, 更大的 p 能够加密更多的信息. 但由于 CNTR-Prime 的明文为比特串, 故适合使用 $p=2$, 同时错误率更低.

CNTR-Prime 和 LTRU 都将明文信息编码在密文的高位, 而其他 NTRU 类型 KEM 都将明文信息编码在密文的低位. 后者的优势是实现简单, 因为它们的密文通常为 $c=phr+m \bmod q$ 这种简单的形式, 但此时密文不能压缩. 前者的优势是能够支持密文压缩, 使得压缩带来的误差对恢复明文的影响在可控范围之内. CNTR-Prime 使用素阶数域上的多项式尺度化 E_8 格编码来纠错所以错误率足够低, LTRU 不使用纠错码所以错误率只能跟安全性匹配而没法做到更低.

5 伪梅森数不完整 NTT

本节将介绍本文提出的伪梅森数不完整 NTT, 它能高效地计算 CNTR-Prime 中素阶数域上的多项式乘法.

5.1 总体流程

在算法 1 中, 公钥 $h=g/f$ 的计算通过素阶数域 $\mathcal{R}_q$ 中多项式求逆和多项式乘法来完成, 即 $h=g\cdot f_{inv}\in\mathcal{R}_q$, 其中 $f_{inv}:=f^{-1} \bmod x^n-x-1$ 为 f 的逆. 本文使用文献 [49] 的常数时间实现的多项式求逆算法计算 f_{inv}, SNTRU-Prime 同样使用了该求逆算法, 具体伪代码可见文献 [49].

可知, 在 CNTR-Prime 中需要计算的多项式乘法包含 2 种: 第 1 种是 $\mathcal{R}_q$ 中多项式乘法 $h=g\cdot f_{inv} \bmod q$ 和

$\sigma = hr \bmod q$; 第 2 种是 $\mathcal{R}_{q_2}$ 中多项式乘法 $m' = cf \bmod^{\pm} q_2$. 但是, 由于 CNTR-Prime 中选择的参数 n 和 q 均为素数以及 q_2 为 2 次幂, 不能直接使用长度为 n 的循环卷积 NTT 计算这 2 种多项式乘法.

本文将基于文献 [28] 的思想, 结合 64 位处理器的特性, 对 CNTR-Prime 的 3 组参数集分别给出高效的伪梅森数不完整 NTT 来计算多项式乘法. 伪梅森数指的是具有 $2^x - 2^y + 1$ 形式的数值, 其中 x 和 y 为正整数. 本文使用满足 $y < x/2$ 的伪梅森素数作为 NTT 的模数, 因为它能支持高效的模约减算法 (见第 6 节). 下面以 $\mathcal{R}_q$ 上的多项式乘法 $h = f \cdot g \in \mathcal{R}_q$ 为例进行介绍, 相同的计算流程可以用于计算 $\mathcal{R}_{q_2}$ 中多项式乘法. 主要的流程如下所示, 相关的示意图如图 1 所示.

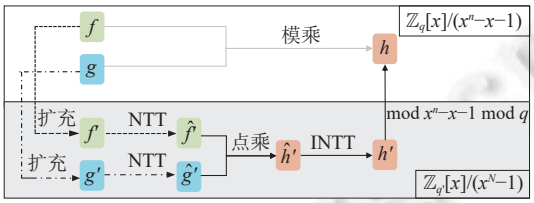

图 1　本文 NTT 计算素阶数域上的多项式乘法流程

1) 将 n 维的多项式 f 和 g 分别在高次项补 0 扩充至 N 维, 得到拓展环上的多项式 $f', g' \in \mathbb{Z}_{q'}[x]/(x^N - 1)$, 其中 $N \geqslant 2n$, q' 为某个足够大的素数使得它大于计算过程中多项式系数在 $\mathbb{Z}$ 上的最大值;

2) 使用 N 长的伪梅森数不完整 NTT 依次计算 f' 和 g' 的正向变换 (NTT)、点乘和逆向变换 (INTT), 得到 $h' = f' \cdot g' \in \mathbb{Z}_{q'}[x]/(x^N - 1)$;

3) 计算 $h = (h' \bmod x^n - x - 1) \bmod q$, 它是 f 和 g 在 $\mathcal{R}_q$ 中相乘的结果.

注意到, 选取这样的 N 和 q' 是为了在第 1 步和第 2 步中计算得到的 h' 跟 $f \cdot g$ 在 $\mathbb{Z}[x]$ 上的结果恒相等, 接着进行 $\bmod x^n - x - 1$ 和 $\bmod q$ 运算后便得到 $f \cdot g \in \mathcal{R}_q$ 中的结果.

5.2　CNTR-Prime-653 的多项式乘法

对于 N 和 q' 的选取, 其选取准则是让 N 包含尽量多的因子 2, q' 为伪梅森素数. 对于 CNTR-Prime-653 的情况, N 和 q' 的选取如下所示.

1) N 的选取

对于 $n = 653$, 备选的 N 有 $N = 1344$、1440 和 1536. 考虑到 $N = 1344 = 2^6 \cdot 3 \cdot 7$ 最接近 $2n$ 且支持快速的基-2 FFT trick 和基-3 FFT trick, 同时避免多项式维度扩充过多造成计算量激增, 所以本文选定 $N = 1344$.

2) 伪梅森数 q' 的选取

下面将详细讨论伪梅森数 q' 的选取, 目标是能够在拓展环 $\mathbb{Z}_{q'}[x]/(x^N - 1)$ 中使用 N 长的高效 NTT 计算多项式乘法.

首先需要注意的是 q' 的选取范围. CNTR-Prime-653 中需要计算的多项式乘法均为大系数多项式和小系数多项式的乘积. 为了更有效地选取合适的伪梅森数 q', 本文使用绝对最小完全剩余系表示 f 和 g 的系数. 下面对 CNTR-Prime-653 中 2 种多项式乘法进行详细讨论. 对于 $\mathcal{R}_q$ 中多项式乘法, 系数范围能够概括为: $|f_i| \leqslant \eta$ 和 $|g_i| \leqslant \frac{q}{2}$. 进一步地, 得到 h' 的系数范围为 $|h'_i| \leqslant n \cdot \frac{q}{2} \cdot \eta = \frac{1}{2} nq\eta$. 因此, 只需满足 $\frac{q'}{2} > \frac{1}{2} nq\eta$, 换句话说, $q' > nq\eta$. 对于 $\mathcal{R}_{q_2}$ 中多项式乘法, f 的系数范围除了常数项为 $|f_0| \leqslant 2\eta + 1$, 其余均为 $|f_i| \leqslant 2\eta$, g 的系数范围为 $|g_i| \leqslant \frac{q_2}{2}$. 则 h' 的系数范围为 $|h'_i| \leqslant (n-1) \cdot \frac{q_2}{2} \cdot 2\eta + \frac{q_2}{2} \cdot (2\eta + 1) = nq_2\eta + \frac{q_2}{2}$. 因此, 只需满足 $\frac{q'}{2} > nq_2\eta + \frac{q_2}{2}$, 即 $q' > (2n\eta + 1)q_2$. 综上, 选择的 q' 需要满足 $q' > \max\{nq\eta, (2n\eta + 1)q_2\}$.

其次, 需要注意的是 N 和 q' 之间的关系. 对于 $N = 1344 = 2^6 \cdot 3 \cdot 7$, 为避免涉及复杂的基-7 FFT trick, 本文在此使用不完整的 NTT, 其 CRT 同构为 $\mathbb{Z}_{q'}[x]/(x^N - 1) \cong \prod_{i=0}^{N/7-1} \mathbb{Z}_{q'}[x]/(x^7 - \zeta^{\tau(i)})$, 其中 ζ 是 $\mathbb{Z}_{q'}$ 中 $N/7$ 次本原单位根, $\tau(i)$

表示第 i 个环中 ζ 的幂, 且它的序号从 0 开始. 所以, q' 只需要满足 $q' \equiv 1 \bmod N/7$.

综上, 满足 $q' > \max\{nq\eta, (2n\eta+1)q_2\}$ 和 $q' \equiv 1 \bmod N/7$ 且具有 $2^x - 2^y + 1$, $y < x/2$ 形式的最小的伪梅森素数 q' 为 $q' = 16777153$. 此时, 需要 6 层的基-2 FFT trick 和 1 层的基-3 FFT trick, 点乘由 $\mathbb{Z}_{q'}[x]/(x^7 - \zeta^{\tau(i)})$ 中 6 次多项式的乘法构成. 本文计算 FFT trick 的顺序为: 先计算 5 层基-2 FFT trick, 再计算 1 层基-3 FFT trick, 最后计算 1 层基-2 FFT trick. 这样使得在最后 1 层中 $\mathbb{Z}_{q'}[x]/(x^7 - \zeta^j)$ 和 $\mathbb{Z}_{q'}[x]/(x^7 + \zeta^j)$ 成对出现, 以便能够只使用最后 1 层基-2 FFT trick 的 $N/14$ 个本原单位根来计算点乘中 $N/7$ 个 6 次多项式乘法. 从表 3 可知, 与直接使用 6 层基-2 FFT trick 加上 1 层基-3 FFT trick 相比, 5 层基-2 FFT trick、1 层基-3 FFT trick、1 层基-2 FFT trick 的计算顺序在保持乘法复杂度不变的情况下, 能够降低 33.3% 本原单位根的存储开销.

表 3 $N = 1344$ 时 FFT trick 顺序的对比

FFT trick顺序	乘法的数量	本原单位根存储 (B)
6层基-2、1层基-3	$\frac{83}{7}N$	2 304
5层基-2、1层基-3、1层基-2	$\frac{83}{7}N$	1 536

注意到, 在 CNTR-Prime-653 的伪梅森数不完整 NTT 的 CRT 同构中, 第 1 层均使用基-2 FFT trick. 对于这种情况, 本文利用本原单位根的性质将第 1 层基-2 FFT trick 中的乘法转换为加减法. 记 ζ 是 $\mathbb{Z}_{q'}$ 中 N/μ 次本原单位根, 其中 $\mu = 7$. 首先可知 $\zeta^{N/\mu} = 1 \bmod q'$ 和 $\zeta^{N/2\mu} = -1 \bmod q'$. 进一步地, 在第 1 层中有:

$$\mathbb{Z}_{q'}[x]/(x^N - 1) \cong \mathbb{Z}_{q'}[x]/(x^{N/2} - \zeta^{N/2\mu}) \times \mathbb{Z}_{q'}[x]/(x^{N/2} + \zeta^{N/2\mu}) = \mathbb{Z}_{q'}[x]/(x^{N/2} + 1) \times \mathbb{Z}_{q'}[x]/(x^{N/2} - 1).$$

所以, 第 1 层基-2 FFT trick 中系数乘以 $\zeta^{N/2\mu}$ 均能转化为系数乘以 -1 (即, 减去该系数). 于是, 第 1 层基-2 FFT trick 只需要 $N/2$ 个加法和 $N/2$ 个减法, 而无需乘法.

针对 CNTR-Prime-653 的伪梅森数不完整 NTT 的参数选取情况见表 4. 为了方便理解, 图 2 展示了针对 CNTR-Prime-653 的伪梅森数不完整 NTT 对应的 CRT 同构树形图.

表 4 伪梅森数不完整 NTT 的参数表

参数集	参数 (n, q, q_2, η)	拓展环 $\mathbb{Z}_{q'}[x]/(x^N - 1)$ 参数 (N, q') 的约束条件	(N, q')的取值	FFT trick顺序
CNTR-Prime-653	(653, 4621, 2^{11}, 3)	$N \geqslant 2n$, $q' \equiv 1 \bmod N/7$, $q' > \max\{nq\eta, (2n\eta+1)q_2\}$	(1344, 16777153=2^{24}–2^6+1)	5层基-2、1层基-3、1层基-2
CNTR-Prime-761	(761, 4591, 2^{10}, 2)	$N \geqslant 2n$, $q' \equiv 1 \bmod N/3$, $q' > \max\{nq\eta, (2n\eta+1)q_2\}$	(1536, 33550337=2^{25}–2^{12}+1)	9层基-2
CNTR-Prime-1277	(1277, 7879, 2^{10}, 2)	$N \geqslant 2n$, $q' \equiv 1 \bmod N/5$, $q' > \max\{nq\eta, (2n\eta+1)q_2\}$	(2560, 33550337=2^{25}–2^{12}+1)	9层基-2

5.3 CNTR-Prime-761 的多项式乘法

对于 CNTR-Prime-761 的情况, 当 $n = 761$ 时, 类似于第 5.2 节的分析, N 包含尽量多的因子 2. 本文选择 $N = 1536 = 2^9 \cdot 3$, 因为它接近 $2n$ 且包含足够多的因子 2. 对于 $N = 1536$, 本文使用的不完整的 NTT 的 CRT 同构 $\mathbb{Z}_{q'}[x]/(x^N - 1) \cong \prod_{i=0}^{N/3-1} \mathbb{Z}_{q'}[x]/(x^3 - \zeta^{\tau(i)})$, 其中 ζ 是 $\mathbb{Z}_{q'}$ 中 $N/3$ 次本原单位根. 所以, 素数 q' 只需要满足 $q' \equiv 1 \bmod N/3$. 所以, 满足 $q' > \max\{nq\eta, (2n\eta+1)q_2\}$ 和 $q' \equiv 1 \bmod N/3$ 且具有 $2^x - 2^y + 1$, $y < x/2$ 形式的最小伪梅森素数为 $q' = 33550337$. 此时, 只需要 9 层基-2 FFT trick, 点乘由 $\mathbb{Z}_{q'}[x]/(x^3 - \zeta^{\tau(i)})$ 中 2 次多项式乘法构成. 类似地, CNTR-Prime-761 的伪梅森数不完整 NTT 的 CRT 同构第 1 层利用 $\zeta^{N/2\mu} = -1 \bmod q'$, $\mu = 3$ 的性质, 可仅需 $N/2$ 个加法和 $N/2$ 个减法, 而无需乘法来计算首层基-2 FFT trick.

表 4 中同样给出了 CNTR-Prime-761 的伪梅森数不完整 NTT 的参数选取情况. CNTR-Prime-761 的伪梅森数

不完整 NTT 对应的 CRT 同构树形图与图 2 的类似. 该 NTT 的正向变换的伪代码可参考算法 9, 只不过 CNTR-Prime-761 的伪梅森数不完整 NTT 仅使用基-2 FFT trick.

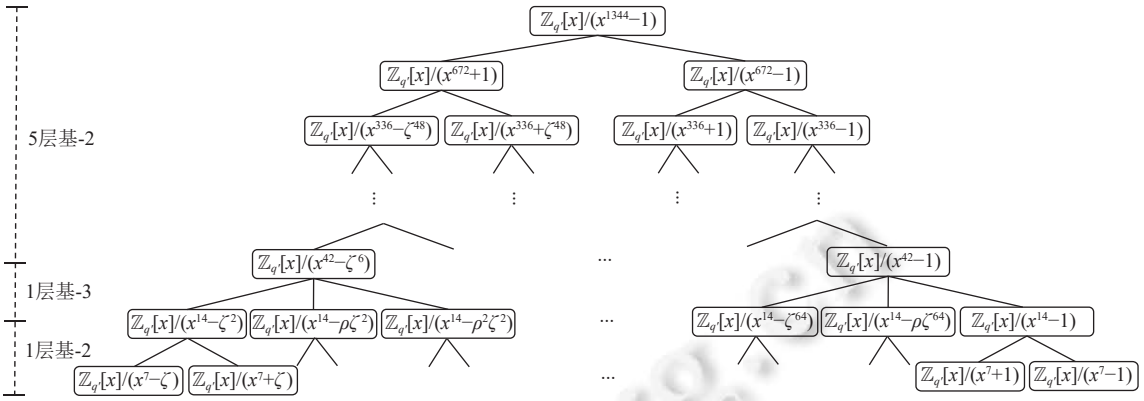

图 2　$N=1344$ 时 CRT 同构的树形图

5.4 CNTR-Prime-1277 的多项式乘法

对于 CNTR-Prime-1277 的情况, 当 $n=1277$ 时, 类似地, 本文选择 $N=2560=2^9\cdot 5$. 为避免使用基-5 FFT trick, 本文使用的不完整的 NTT 的 CRT 同构 $\mathbb{Z}_{q'}[x]/(x^N-1)\cong\prod_{i=0}^{N/5-1}\mathbb{Z}_{q'}[x]/(x^5-\zeta^{\tau(i)})$, 其中 ζ 是 $\mathbb{Z}_{q'}$ 中 $N/5$ 次本原单位根. 所以, 素数 q' 只需要满足 $q'\equiv 1 \bmod N/5$. 所以, 满足 $q'>\max\{nq\eta,(2n\eta+1)q_2\}$ 和 $q'\equiv 1 \bmod N/5$ 的最小伪梅森素数为 $q'=33550337$. 此时, 同样需要 9 层的基-2 FFT trick, 点乘由 $\mathbb{Z}_{q'}[x]/(x^5-\zeta^{\tau(i)})$ 中 4 次多项式乘法构成. 同样地, CNTR-Prime-1277 的伪梅森数不完整 NTT 的首层 CRT 同构仅使用 $N/2$ 个加法和 $N/2$ 个减法来计算基-2 FFT trick.

同样地, 表 4 中给出了 CNTR-Prime-1277 的伪梅森数不完整 NTT 的参数选取情况. CNTR-Prime-1277 的伪梅森数不完整 NTT 对应的 CRT 同构树形图同样与图 2 的类似. 该 NTT 的正向变换的伪代码同样可参考算法 9.

5.5 进一步的优化

1) 点乘的优化

本文使用 1-迭代 Karatsuba 算法加速点乘中 $\mu-1$ 次多项式乘法, 其中 $\mu=7$、3 和 5, 对应于 $n=653$、761 和 1277 的点乘在每个分量的维度. 不失一般性, 以 $\mathbb{Z}_{q'}[x]/(x^\mu-\zeta)$ 为例, 点乘涉及的多项式乘法的形式写为 $\left(\sum_{i=0}^{\mu-1}a_ix^i\right)\left(\sum_{i=0}^{\mu-1}b_ix^i\right) \bmod x^\mu-\zeta$.

使用 1-迭代 Karatsuba 算法计算点乘分量的步骤为: 首先计算并存储 $t_i=a_ib_i, 0\leqslant i\leqslant\mu-1$; 接着对于满足 $0\leqslant i<j\leqslant\mu-1$ 的 (i,j) 对, 通过 $(a_i+a_j)(b_i+b_j)-t_i-t_j$ 来计算 $a_ib_j+a_jb_i$. 1-迭代 Karatsuba 算法每次能够将 4 个乘法减少至 3 个乘法.

2) 计算顺序的优化

类似于文献 [28], 在计算逆向 NTT 变换时, 本文在基-2 逆向 FFT trick 和基-3 逆向 FFT trick 中分别延迟处理因子 1/2 和 1/3, 所以在逆向 NTT 变换的末尾需要乘以 $(N/\mu)^{-1} \bmod q'$ 才能得到图 1 中的 N 维多项式 h', 其中 $\mu=7$、3 和 5, 对应于 $n=653$、761 和 1277. 接着, 计算 $h=(h' \bmod x^n-x-1) \bmod q$. 这种计算 h 的方式需要计算 N 次 $(N/\mu)^{-1} \bmod q'$ 乘法.

为进一步降低乘法的数量, 类似于文献 [28] 的做法, 本文在逆向 NTT 变换时并不马上乘以 $(N/\mu)^{-1} \bmod q'$, 而是先得到 N 维多项式 h'; 接着计算 $h' \bmod x^n-x-1$, 得到 n 维多项式 h''; 然后才对 h'' 的每个系数乘以 $(N/\mu)^{-1} \bmod q'$; 最后, 对每个系数计算 $\bmod q$, 得到 h. 这种计算 h 的方式仅需要计算 n 次 $(N/\mu)^{-1} \bmod q'$ 乘法. 由于 $N>n$, 所以通过调整计算顺序, CNTR-Prime-653、CNTR-Prime-761、CNTR-Prime-1277 分别减少 691、775 和 1283 个

$\bmod q'$ 乘法.

6 改进的伪梅森数约减

在格密码实现中, 多项式系数往往使用绝对最小完全剩余系表示, 计算过程的中间变量值允许使用有符号整型变量 (比如有符号 16、32 位整型). 为了防止系数的数值溢出整型变量, 需要模约减算法将系数约减到合适的范围.

对伪梅森数 $q'=2^x-2^y+1$ 来说, 经典的伪梅森数约减算法[50]以 $0\leqslant a\leqslant (q')^2$ 为输入, 经过两次利用 $2^x\equiv 2^y-1 \bmod q'$, 还需额外的纠正步骤 (即: 若所得值不小于 q', 则需要额外减去 q' 值), 最后以 $r=a \bmod q'$, $0\leqslant r<q'$ 作为输出. 可见, 经典伪梅森数约减算法不仅输入输出范围均为正整数, 还需要额外的纠正步骤.

本节将介绍本文提出的改进的伪梅森数约减算法, 它能解决上述两个痛点. 另外, 它十分适用于格密码实现, 特别是第 5 节提出的伪梅森数不完整 NTT 的实现. 针对格密码实现中的模约减算法所需的不同输入输出范围, 本文提出的改进的伪梅森数约减分为单轮和双轮, 具体伪代码分别见算法 9 和算法 10.

对于任意整数 a 和正整数 x, 本节使用符号 $[a]_x$ 表示 $a \bmod 2^x$, 即 a 的二进制表示的低 x 位; 使用 $[a]^x$ 表示 $\lfloor a/2^x \rfloor$, 即 a 的二进制表示右移 x 位. 它们能够使用高效位运算来计算.

算法 9. 改进的伪梅森数约减 (单轮).

输入: 伪梅森数 $q'=2^x-2^y+1$, $y<\dfrac{x}{2}$, 整数 a, $-2^{2x-y}<a<2^{2x-y}$;
输出: $r=a \bmod q', -q'\leqslant r<q'$.

① $r:=[a]_x-[a]^x+[a]^x \ll y-q'$
② return r

算法 10. 改进的伪梅森数约减 (双轮).

输入: 伪梅森数 $q'=2^x-2^y+1$, $y<\dfrac{x}{2}$, 整数 a, $-2^{3x-2y}<a<2^{3x-2y}$;
输出: $r=a \bmod q', -q'\leqslant r<q'$.

① $r:=[a]_x-[a]^x+[a]^x \ll y$
② $r:=[r]_x-[r]^x+[r]^x \ll y-q'$
③ return r

6.1 改进的伪梅森数约减 (单轮)

改进的伪梅森数约减 (单轮) 的伪代码见算法 9. 它以 $a\in(-2^{2x-y}, 2^{2x-y})$ 作为输入, 计算得到 $r=a \bmod q'$, $r\in[-q', q')$. 单轮的意思是仅基于一轮 $2^x\equiv 2^y-1 \bmod q'$ 性质来计算模约减. 可见, 算法 9 能以有符号整型作为输入, 且计算得到的输出值位于 $[-q', q')$ 区间之中, 适用于后续 FFT trick 的计算.

注意到, 算法 9 的计算仅需简单的位运算 (逻辑与、移位) 和加减法, 无需多余的乘法或除法. 算法 9 的正确性由下面的定理 4 给出, 其证明的主要基础是 q' 的伪梅森数形式 $q'=2^x-2^y+1$ 和输入值 a 的表示方式 $a=[a]_x+2^x\cdot[a]^x$.

定理 4. 改进的伪梅森数约减 (单轮) 的正确性. 对于满足算法 9 输入要求的伪梅森数 q' 和整数 a, 算法 9 是正确的.

证明: 下面通过 2 步来证明定理 4.

第 1 步, 先证明 $r\equiv a \bmod q'$. 由于 $q'=2^x-2^y+1$, 于是 $2^x\equiv 2^y-1 \bmod q'$, 那么有:

$$r \bmod q'=[a]_x-[a]^x+[a]^x \ll y-q' \bmod q'\equiv [a]_x+(2^y-1)\cdot[a]^x \bmod q'\equiv [a]_x+2^x\cdot[a]^x \bmod q'=a \bmod q',$$

其中, 最后的等号成立是因为 $a=[a]_x+2^x\cdot[a]^x$. 第 1 步得证.

第 2 步, 再证明 r 的范围满足 $-q' \leqslant r < q'$. 易知 $2^x - q' = 2^y - 1$. 类似于第 1 步的推导, 有:

$$r = [a]_x - [a]^x + [a]^x \ll y - q' = [a]_x + (2^y - 1)\cdot[a]^x - q' = [a]_x + (2^x - q')\cdot[a]^x - q'$$
$$= ([a]_x + 2^x \cdot [a]^x) - q' \cdot [a]^x - q' = a - ([a]^x + 1)\cdot q'.$$

由于 $a = \left\lfloor \frac{a}{q'} \right\rfloor \cdot q' + a \bmod q'$, 于是 $r = \left(\left\lfloor \frac{a}{q'} \right\rfloor \cdot q' + a \bmod q'\right) - \left(\left\lfloor \frac{a}{2^x} \right\rfloor + 1\right)\cdot q' = \left(\left\lfloor \frac{a}{q'} \right\rfloor - \left\lfloor \frac{a}{2^x} \right\rfloor\right)\cdot q' + (-q' + a \bmod q')$. 当 $-2^{2x-y} < a < 2^{2x-y}$ 时, 有: $0 \leqslant \left\lfloor \frac{a}{q'} \right\rfloor - \left\lfloor \frac{a}{2^x} \right\rfloor \leqslant 1$, 所以得到 $0 \leqslant \left(\left\lfloor \frac{a}{q'} \right\rfloor - \left\lfloor \frac{a}{2^x} \right\rfloor\right)\cdot q' \leqslant q'$. 另外, 由于 $-q' \leqslant -q' + a \bmod q' < 0$. 结合两个不等式可得 $-q' \leqslant r < q'$. 第 2 步得证. 于是, 定理 4 得证. 证毕.

6.2 改进的伪梅森数约减 (双轮)

为满足更大的输入范围, 本文在此提出改进的伪梅森数约减 (双轮), 它的伪代码见算法 10, 它以 $a \in (-2^{3x-2y}, 2^{3x-2y})$ 作为输入, 计算得到 $r = a \bmod q'$, $r \in [-q', q')$. 双轮的意思指的是 2 次利用 $2^x \equiv 2^y - 1 \bmod q'$ 性质来计算模约减. 改进的伪梅森数约减 (双轮) 适用于计算模乘 $r = a \cdot b \bmod q'$, 其中 $a, b \in \mathbb{Z}_{q'}$ 以及 $q' = 2^x - 2^y + 1$. 此时 $a \cdot b$ 的数值大小往往超过了当前整型变量的有效表示范围, 比如在格密码 NTT 的基-2 FFT trick 中需要计算 $a \pm \zeta \cdot b$, 其中 $a, b, \zeta \in \mathbb{Z}_{q'}$.

相比于第 6.1 节的改进的伪梅森数约减 (单轮), 双轮的约减算法具有更大的输入范围, 但同时需要多一轮的计算 (即: 算法 10 还需要第 2 行的计算量). 算法 10 的正确性由下面的定理 5 给出. 定理 5 的与定理 4 只是在输入范围和轮数上存在差异. 定理 5 的证明可以从定理 4 的直接延伸过来, 在此不再赘述.

定理 5. 改进的伪梅森数约减 (双轮) 的正确性. 对于满足算法 10 输入要求的伪梅森数 q' 和整数 a, 算法 10 是正确的.

6.3 比较和优缺点分析

本节对比改进的伪梅森数约减和之前的伪梅森数约减[50]、Barrett 约减算法[51]和 Montgomery 约减算法[52]. 总的来说, 相比于 Montgomery 约减和 Barrett 约减, 本文的改进的伪梅森数约减最重要的优势是无需任何乘法运算, 仅需简单的位运算和加减法.

1) 和之前的伪梅森数约减的比较

尽管本文提出的改进的伪梅森数约减 (单轮、双轮) 与之前的伪梅森数约减都利用了 $2^x \equiv 2^y - 1 \bmod q'$ 性质对输入值进行约减, 但改进的伪梅森数约减 (单轮、双轮) 还具有以下 3 点优势.

i) 支持有符号整型的输入输出, 适合格密码实现. 现有的格密码实现普遍使用绝对最小完全剩余系表示系数值且使用有符号整型变量. 之前的伪梅森数约减仅支持无符号整型输入输出值, 即输入范围 $[0, (q')^2]$ 和输出范围 $[0, q')$, 不便于格密码实现, 因为需要把有符号整型加上 q' 将系数转换为无符号整型. 改进的伪梅森数约减 (单轮、双轮) 支持有符号整型输入输出值, 并无需额外的加 q' 操作.

ii) 更大的输入范围. 之前的伪梅森数约减仅支持输入范围 $[0, (q')^2]$, 而改进的伪梅森数约减 (单轮、双轮) 分别支持 $(-2^{2x-y}, 2^{2x-y})$ 和 $(-2^{3x-2y}, 2^{3x-2y})$ 的输入范围. 在 $q' = 2^x - 2^y + 1$, $y < x/2$ 的情况下, 改进的伪梅森数约减的输入范围更灵活, 同时双轮情况的输入范围比之前的伪梅森数约减的输入范围更大.

iii) 无需纠正数值步骤. 如上文所述, 在之前的伪梅森数约减中, 若所得值不小于 q', 则需要额外减去 q' 值. 本文的改进的伪梅森数约减 (单轮、双轮) 无需额外的纠正数值步骤. 而且它的输出值位于 $[-q', q')$ 之中, 满足格密码运算的数值范围要求.

2) 和 Montgomery 约减的比较

本文在此仅将改进的伪梅森数约减 (双轮) 和 Montgomery 约减进行对比, 因为相比于改进的伪梅森数约减 (单轮), 改进的伪梅森数约减 (双轮) 的输入输出范围与 Montgomery 约减的接近, 它们之间的对比更公平. 相比于 Montgomery 约减, 改进的伪梅森数约减 (双轮) 无需任何乘法运算, 无需预存任何数值, 同时输出值位于正常域而无需使用 Montgomery 域表示. 不过, Montgomery 约减的输入范围可能大于改进的伪梅森数约减 (双轮) 的输入范

围. 比如, 在本文使用的伪梅森数 33 550 337 的情况下, Montgomery 约减的输入范围更大.

另外, 在软件实现方面, 改进的伪梅森数约减 (双轮) 由于进行位移操作, 多数情况下它不是字长友好的. 而 Montgomery 约减是字长友好的, 它可以使用类似于 AVX2 指令集高效的 vpmulhw、vpmullw、vpmulld 和 vpmuldq 等指令计算字长级别数值乘法, 同时便于 gcc 编译器对 Montgomery 约减进行深度优化, 使得它在处理字长级别数值的约减时, 能够更加高效, 在 Intel 平台中更具优势. 但在硬件实现方面, 由于硬件设计语言可自定义中间变量数据位宽, 改进的伪梅森数约减 (双轮) 的位运算更有优势. 更多的分析和具体对比实验数据见第 8.1 节.

3) 和 Barrett 约减的比较

同样地, 本文在此将改进的伪梅森数约减 (单轮) 和 Barrett 约减进行对比, 因为两者的输入输出范围接近. 相比于 Barrett 约减, 改进的伪梅森数约减 (单轮) 无需任何乘法运算和预存任何数值, 同时输入范围更大. 第 8.1 节同样给出了它们的实验数据比较.

7 实现细节

本节将给出 CNTR-Prime 的一些实现细节. 本文主要关注 CNTR-Prime 的 C 语言实现, 并且为了抵抗侧信道攻击 (如计时攻击[53]), 本文所有实现均采用常数时间的实现技巧. 其中, 针对尺度化 E_8 格的编码算法和解码算法, 本文提供类似于文献 [21] 的常数时间实现技巧.

7.1 数据结构

本文主要使用长度为 n 的有符号 16 位整型数组来存储多项式. 但是在进行 NTT 计算时, 本文暂时转向使用长度为 N 的有符号 32 位整型, 这是因为本文使用了 32 位整型的伪梅森数 q'. 在完成 NTT 计算的同时, 通过模约减算法得到 $[0,q)$ 之内的数值.

7.2 哈希函数

本文的 CNTR-Prime 均使用 SHA-3 系列对哈希函数进行实例化. 具体而言, 以秘密种子作为 SHAKE-256 的输入, 然后输出秘密多项式 f',g,r 采样所需要的随机数. 基于 SHA3-512 来实例化哈希函数 $\mathcal{H}$, 并将公钥前缀 $ID(pk)$ 和明文 m 作为输入, 然后输出 64 字节, 其中前 32 字节作为共享密钥使用, 后 32 字节作为加密算法的秘密种子使用.

7.3 公私钥和密文

公私钥均是以正常域中多项式的形式进行存储和传输. 本文使用文献 [10] 的公钥压缩技术对本文方案的公钥多项式 h 进行紧凑地压缩, 使得 3 组参数集的公钥尺寸分别降低 68、79、9 字节, 降低的比例分别为 6.4%、6.3%、0.4%. NTRU-Prime 同样使用了这种公钥压缩技术. 在此以公钥多项式 $h=\sum_{i=0}^{n-1}h_ix^i$ 为例来简要说明, 其中 $h_i\in\mathbb{Z}_q$, $i=0,1,\ldots,n-1$. 压缩过程是将公钥向量 $(h_0,h_1,\ldots,h_{n-1})$ 无损地转换到一个字节向量 S, 其中 S 的数值表达更加紧凑. 核心思想是将每个整数对 $(h_i,h_{i+1})\bmod q$ 按照 $h_i+qh_{i+1}\bmod q^2$ 的方式合并成一个新的更大的整数. 解压缩过程是其逆过程, 核心思想是通过带余除法从 $h_i+qh_{i+1}\bmod q^2$ 恢复出整数对 $(h_i,h_{i+1})\bmod q$.

接着解释上述压缩过程能够降低公钥尺寸的原因. CNTR-Prime 的模数 q 的二进制表示需要 13 位, 但 13 位整数能够表示的最大值为 $2^{13}-1$, 而直接使用 13 位存储空间来存储多项式系数值会导致 $\{q,q+1,\ldots,2^{13}-1\}$ 的数值空间未被充分利用. 通过 $h_i+qh_{i+1}\bmod q^2$ 的方式将模数从 q 转换到 q^2, 便可以利用这部分数值空间, 使得数值空间更加紧凑. 例如, 对于 CNTR-Prime-761, 模数 $q=4591$ 小于 2^{13} 且两个数值差距大, 这种情况下公钥压缩可以显著提升模数转换带来的数值空间利用率, 压缩效果明显, 使得公钥尺寸降低 6.3%. 然而, 对于 CNTR-Prime-1277, 模数 $q=7879$ 小于 2^{13} 但两个数值差距小, 这意味着压缩过程可利用的 q 至 $2^{13}-1$ 范围内的数值空间不大, 因此压缩效果不明显, 导致公钥尺寸只降低 0.4%.

与文献 [47] 相同, 本文使用压缩后公钥的前 33 字节作为公钥前缀 $ID(pk)$. 这是因为压缩前公钥多项式 h 跟 $\mathcal{R}_q$ 中均匀随机元素是计算不可区分的, 且 h 的前 20 个系数具有超过 256 bit 的最小熵, 由于 h 每个系数均为 13 bit,

在压缩 h 的情况下, 它们共占压缩 h 后的字节数组前面至多 33 字节.

公钥加密方案的私钥所包含的秘密多项式 f 的系数位于 $[-p\eta, p\eta+1]$ 之中, 本文将其转化为正数再进行传输. 密文只包含正常域中一个压缩后的多项式, 且它的每个系数占 $\lceil\log(q_2)\rceil$ 比特, 所以密文共需要 $\lceil n\lceil\log(q_2)\rceil/8\rceil$ 字节. 由于密文模数 q_2 已经是 2 次幂整数, 此时能够直接使用 $\lceil\log(q_2)\rceil$ 位存储空间来存储多项式系数值, 并且从上面分析可知使用文献 [10] 的压缩技术对其系数的数值空间利用率并无提升.

8 实验结果和比较

本节将给出本文提出的改进的伪梅森数约减、伪梅森数不完整 NTT、CNTR-Prime 方案的测试数据, 以及它们和同类方案的对比实验数据. 软件实现的测试设备为: 硬件配置为 2.3 GHz 的 Intel(R) Core(TM) i7-10510U CPU 和 16 GB 内存的笔记本电脑, 且关闭 Turbo Boost 和 Hyperthreading, 核为 Linux Kernel 4.4.0 的 Ubuntu 20.04 LTS 操作系统, 且 gcc 版本为 9.4.0. 编译选项为-O3. 所有实验均运算 10 000 次并取 CPU 周期数的中位数. 部分实验需要额外测量运算 10 000 次 CPU 周期数的均值. 由于系统中断原因, CPU 周期数的中位数更能体现方案真实的运行耗时. 对于部分实验所涉及的硬件实现, 本文采用 Verilog HDL 语言进行了纯硬件实现, 开发环境为 Vivado 2017.4, 针对 28 nm 工艺 FPGA 芯片 Xilinx Artix-7 系列 XC7A200TFBG484-2 型号做了仿真、综合、布局布线.

8.1 模约减的比较

本节对改进的伪梅森数约减、Barrett 约减算法[51]和 Montgomery 约减算法[52]的软硬件实现进行比较.

1) 软件实现的比较

图 3 给出了这 3 种约减算法运行 10 000 次的总时间 CPU 周期数 (单位: cycle). 下面按照输入范围接近的原则进行对比. 改进的伪梅森数约减 (单轮) 和 Barrett 约减输入范围接近, 改进的伪梅森数约减 (双轮) 和 Montgomery 约减输入范围接近, 故对它们两两进行对比. 从图 3 中可以看出, 对于 CNTR-Prime 使用到的伪梅森数 16 777 153 和 33 550 337, 改进的伪梅森数约减 (单轮) 都比 Barrett 约减快了 2.6%. 但是, 改进的伪梅森数约减 (双轮) 慢于 Montgomery 约减. 从第 6.3 节分析可知, 主要是因为 Montgomery 约减经过 gcc 编译器优化之后, 更擅长处理字长级别数值的约减.

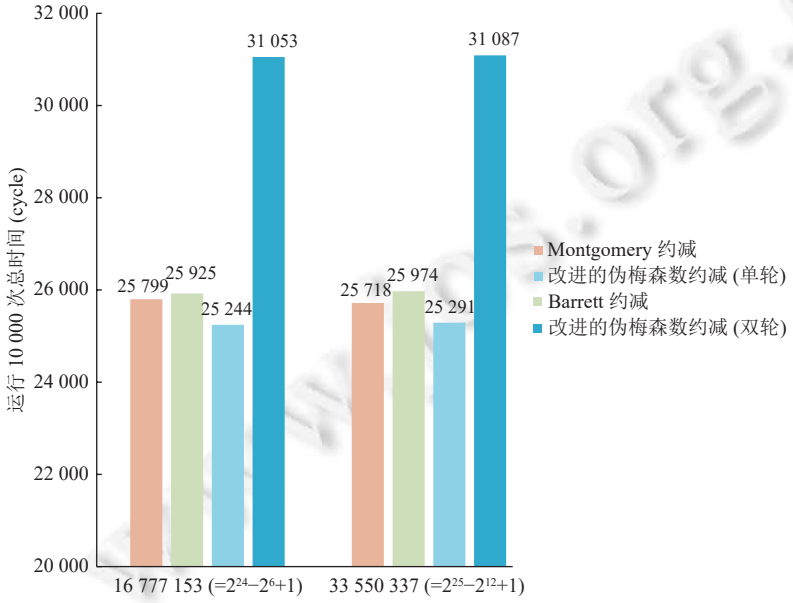

图 3 不同模约减算法的软件实现比较

结合第 6.3 节分析, 本文在计算伪梅森数不完整 NTT 过程中, 计算系数乘法情况下使用 Montgomery 约减, 计算加法后避免溢出情况下使用改进的伪梅森数约减 (单轮).

2) 硬件实现的比较

表 5 给出了这 3 种约减算法硬件实现的测试数据. 相对于 Barrett 约减和 Montgomery 约减, 改进的伪梅森数约减模块硬件实现的计算步骤更少, 且无分支判断, 因此其时钟周期数更少, 最大时钟频率更高. 具体而言, 改进的伪梅森数约减 (单轮) 比 Barrett 约减和 Montgomery 约减分别快了 4 倍和 6 倍; 改进的伪梅森数约减 (双轮) 比 Barrett 约减和 Montgomery 约减分别快了 2 倍和 3 倍. 其中, Barrett 约减模块采用移位操作代替乘法计算, 两个模数 16 777 153 和 33 550 337 的计算步骤相同, 消耗时钟周期数一致, 模数位宽的增加导致硬件资源消耗增多. Montgomery 约减模块采用字长级别 (32 bit) 的设计思路, 使用片内 DSP 资源实现乘法计算, 因此两个模数的时钟周期及硬件消耗一致.

表 5 单个模约减的硬件实现比较

比较项目	$16777153=2^{24}-2^{6}+1$				$33550337=2^{25}-2^{12}+1$			
	Montgomery	Barrett	改进的伪梅森数约减		Montgomery	Barrett	改进的伪梅森数约减	
			单轮	双轮			单轮	双轮
时钟周期数 (cycle)	6	4	1	2	6	4	1	2
最大时钟频率 (MHz)	256.1	310.4	357.1	353.4	256.1	309.1	356.2	350.4
LUT/FF/DSP	229/122/4	63/79/—	25/25/—	94/74/—	229/122/4	114/123/—	26/26/—	64/64/—

注: LUT为查找表 (look-up-table), FF为触发器 (flip-flop), DSP为乘法器 (digital-signal-processor). “—”表示无数据

与软件实现相比, 改进的伪梅森数约减 (单轮、双轮) 的硬件实现都有着更明显的速度优势. 其原因在于: 1) FPGA 采用并行架构, 数据计算过程不依赖于指令集, 而是通过不同电路单元的组合实现相应功能, 所以 Montgomery 约减没有指令集优化和编译器优化的优势; 2) 硬件设计语言可自定义中间变量数据位宽, 优化后的数据冗余较少, 且相比于 Barrett 约减和 Montgomery 约减的乘法运算, 改进的伪梅森数约减的位运算相对简单, 从而降低了改进的伪梅森数约减模块的硬件资源消耗.

8.2 素阶数域上的多项式乘法比较

本节将给出本文提出的伪梅森数不完整 NTT 和文献 [28,54] 的 NTT、4-way Toom-Cook、Schoolbook (朴素教科书) 等算法在计算素阶数域上多项式乘法的比较.

文献 [28,54] 只针对 NTRU-Prime 在素阶数域 $\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$ 上多项式乘法进行 NTT 优化, 缺少针对素阶数域 $\mathbb{Z}_{4621}[x]/(x^{653}-x-1)$ 和 $\mathbb{Z}_{7879}[x]/(x^{1277}-x-1)$ 的 NTT 优化策略. 另外, 文献 [28,54] 主要针对 ARM Cortex-M4 或 AVX2 实现, 并未给出 C 实现开源代码. 为了公平比较, 表 6 仅给出了 $\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$ 上伪梅森数不完整 NTT、文献 [28,54] 的 NTT、4-way Toom-Cook 和 Schoolbook 算法分别计算单个多项式乘法时的整数乘法数量, 其中文献 [28,54] 的 NTT 的数据是根据文献 [28,54] 给出的优化技巧计算所得.

对于 CNTR-Prime 所使用的 $n=653$、761 和 1277 时共 6 种多项式环上的多项式乘法, 表 7 给出了伪梅森数不完整 NTT、4-way Toom-Cook 和 Schoolbook 算法分别计算单个多项式乘法的 CPU 周期数.

从表 6 可知, 相比其他算法, 本文伪梅森数不完整 NTT 在计算素阶数域 $\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$ 上单个多项式乘法时更具优势. 具体而言, 伪梅森数不完整 NTT 比文献 [28] 的 Good’s trick+NTT、混合基 NTT (策略 1)、混合基 NTT (策略 2) 乘法数量少了 2.0–6.0 倍. 尽管 Good’s trick+NTT 的拓展环的维度和伪梅森数不完整 NTT 的相同, 但是前者的乘法策略消耗更多乘法. 文献 [28] 的混合基 NTT (策略 1、策略 2) 由于需要计算复杂的基-5 FFT trick 和基-17 FFT trick, 故乘法总数更多. 另外, 伪梅森数不完整 NTT 比文献 [54] 的 CRT+NTT 快了 3.5 倍, 因为 CRT+NTT 本质上使用两套 NTT, 所以它的乘法数量比伪梅森数不完整 NTT 的更多.

从表 7 可知, 在 CNTR-Prime 所使用的 $n=653$、761 和 1277 时共 6 种多项式环上, 伪梅森数不完整 NTT 比其他多项式乘法算法更高效. 具体而言, 对于素阶数域 $\mathbb{Z}_{4621}[x]/(x^{653}-x-1)$、$\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$ 和 $\mathbb{Z}_{7879}[x]/(x^{1277}-x-1)$,

本文的伪梅森数不完整 NTT 比 4-way Toom-Cook 分别快了 13.2 倍、18.4 倍、28.8 倍; 比 Schoolbook 算法分别快了 62.5 倍、76.7 倍、121.1 倍. 对于解密算法中的 2 次幂模数的多项式环乘法, 2 次幂模数的模约减能够利用位运算高效计算, 故此时 3 种多项式乘法算法的差距不大. 具体而言, 对于 3 种多项式环 $\mathbb{Z}_{2^{11}}[x]/(x^{653}-x-1)$、$\mathbb{Z}_{2^{10}}[x]/(x^{761}-x-1)$ 和 $\mathbb{Z}_{2^{10}}[x]/(x^{1277}-x-1)$, 伪梅森数不完整 NTT 比 4-way Toom-Cook 分别快了 4.6%、23.5%、30.9%; 比 Schoolbook 算法分别快了 21.4%、36.9%、44.1%.

表 6 $\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$ 中多项式乘法的乘法数量比较

多项式乘法算法	拓展环	(2个) 正向变换 (k)	逆向变换 (k)	点乘 (k)	总计 (k)
伪梅森数不完整NTT (本文)	$\mathbb{Z}_{33550337}[x]/(x^{1536}-1)$	10.8	6.1	3.6	20.5
Good's trick+NTT[28]	$\mathbb{Z}_{6984193}[x]/(x^{1536}-1)$	23.0	17.7	1.5	42.2
混合基NTT (策略1)[28]	$\mathbb{Z}_{4591}[x]/(x^{1620}-1)$	24.3	6.5	11.3	42.1
混合基NTT (策略2)[28]	$\mathbb{Z}_{4591}[x]/(x^{1530}-1)$	52.7	52.7	16.8	122.2
CRT+NTT[54]	$\mathbb{Z}_{7681}[x]/(x^{1536}-1)\times\mathbb{Z}_{10753}[x]/(x^{1536}-1)$	—	—	—	72.2
4-way Toom-Cook	—	—	—	—	140.7
Schoolbook	—	—	—	—	579.1

注: CRT+NTT[54]、4-way Toom-Cook和Schoolbook都不能进一步细分为正向(逆向)变换和点乘, 故只给出总计的数据. "—"表示无数据

表 7 CNTR-Prime 的多项式乘法比较 (kcycle)

参数集	底层多项式环	伪梅森数不完整NTT (本文)	4-way Toom-Cook	Schoolbook
CNTR-Prime-653	$\mathbb{Z}_{4621}[x]/(x^{653}-x-1)$	46.7	617.1	2920.7
	$\mathbb{Z}_{2^{11}}[x]/(x^{653}-x-1)$	43.5	45.6	55.4
CNTR-Prime-761	$\mathbb{Z}_{4591}[x]/(x^{761}-x-1)$	53.1	977.2	4075.1
	$\mathbb{Z}_{2^{10}}[x]/(x^{761}-x-1)$	51.7	67.6	82.0
CNTR-Prime-1277	$\mathbb{Z}_{7879}[x]/(x^{1277}-x-1)$	95.2	2744.5	11532.0
	$\mathbb{Z}_{2^{10}}[x]/(x^{1277}-x-1)$	92.9	134.5	166.2

8.3 素阶数域和分圆环的多项式乘法比较

后文图 4 中给出了素阶数域、2 次幂分圆环和三项分圆环这 3 种代数结构上使用 NTT 计算单个多项式乘法的 CPU 周期数, 其中素阶数域上的所有多项式乘法均使用本文提出的伪梅森数不完整 NTT 计算的, 2 次幂分圆环和三项分圆环使用基-2 FFT trick 计算. 为了公平比较, 这 3 种代数结构的模数均选取 13 bit 的素数. 下面对于维度 n 接近的情况, 对比 3 种代数结构上使用 NTT 计算多项式乘法性能.

当 $n=653$ 时素阶数域的多项式乘法性能跟 2 次幂分圆环和三项分圆环 $n=512$ 的多项式乘法性能差别较小, 仅为±0.4 kcycle. 但当 $n=761$ 时素阶数域的伪梅森数不完整 NTT 计算的多项式乘法略快于三项分圆环 $n=768$ 情形. 主要原因是三项分圆环 $\mathbb{Z}_{7681}[x]/(x^{768}-x^{384}+1)$ 计算 NTT 过程中, 多项式系数在进行 4 次加减法后需要模约减到 $\mathbb{Z}_{7681}$ 中, 以防止数值溢出, 使得该 NTT 需要大量额外的模约减操作, 总体耗时更多. 当 $n=1277$ 时素阶数域的伪梅森数不完整 NTT 稍慢于 2 次幂和三项分圆环 $n=1024$ 情形的 NTT, CPU 周期数分别增加 8.0% 和 0.5%. 这主要是因为此时素阶数域维度更大, 需要计算的系数个数更多, 使得总体速度下降.

总的来说, 使用本文提出的伪梅森数不完整 NTT 之后, 素阶数域上多项式乘法性能接近甚至优于 2 次幂分圆环和三项分圆环的多项式乘法性能.

8.4 方案的性能比较

由于本文研究动机主要是设计一个比 SNTRU-Prime[10]更优的方案, 故在此主要跟 SNTRU-Prime 比较, 同时也给出本文方案跟其他方案的比较, 且主要比较它们安全强度接近的参数集, 如表 8 所示.

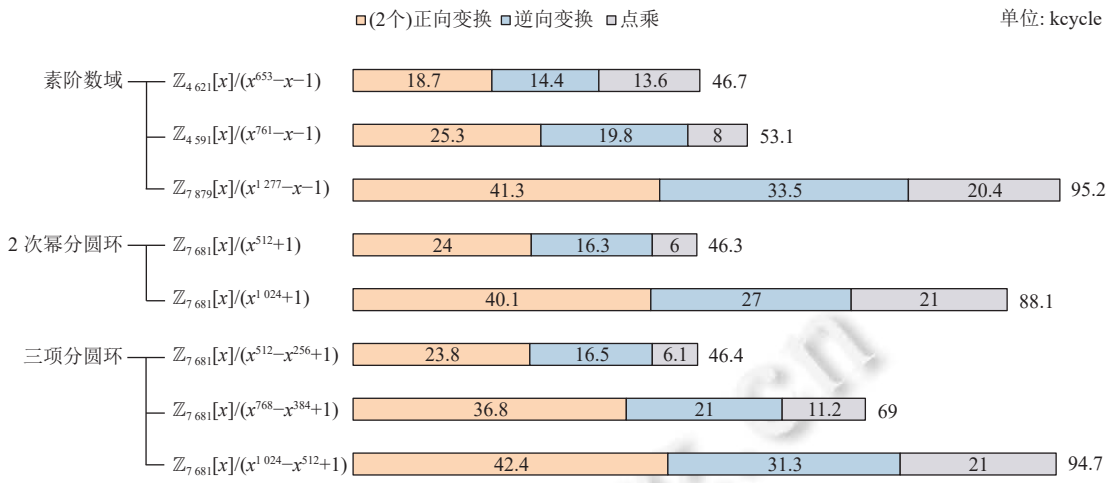

图 4　不同代数结构的单个多项式乘法比较

表 8　方案的性能比较

方案	困难性假设	底层代数结构	$\lvert pk\rvert$	$\lvert ct\rvert$	B.W.	(C, Q)	δ
CNTR-Prime-653 (本文)	NTRU+RLWR	素阶数域	994	898	1 892	(151, 136)	2^{-166}
CNTR-Prime-761 (本文)	NTRU+RLWR	素阶数域	1 158	952	2 110	(172, 156)	2^{-167}
CNTR-Prime-1277 (本文)	NTRU+RLWR	素阶数域	2 067	1 597	3 664	(297, 269)	2^{-279}
SNTRU-Prime-653[10]	NTRU-OW	素阶数域	994	897	1 891	(129, 117)	0
SNTRU-Prime-761[10]	NTRU-OW	素阶数域	1 158	1 039	2 197	(153, 137)	0
SNTRU-Prime-1277[10]	NTRU-OW	素阶数域	2 067	1 847	3 914	(270, 244)	0
NTRU-HRSS[9]	NTRU-OW	截断多项式环	1 138	1 138	2 276	(136, 124)	0
NTRU-A^{648}_{2917}[17]	NTRU+RLWE	三项分圆环	972	972	1 944	(152, 137)	2^{-175}
NTTRU[19]	NTRU-OW	三项分圆环	1 248	1 248	2 496	(153, 140)	$2^{-1\,217}$
BAT-512[18]	NTRU+RLWR	2次幂分圆环	521	473	994	(140, 124)	2^{-146}
LTRU[20]	NTRU-OW	三项分圆环	972	842	1 814	(142, 129)	2^{-154}
CNTR-768[21]	NTRU+RLWR	三项分圆环	1 152	960	2 112	(191, 173)	2^{-230}
CTRU-768[21]	NTRU+RLWE	三项分圆环	1 152	960	2 112	(181, 164)	2^{-184}
Kyber-768[11]	MLWE	2次幂分圆环	1 184	1 088	2 272	(183, 166)	2^{-164}
BAT-1024[18]	NTRU+RLWR	2次幂分圆环	1 230	1 006	2 236	(283, 250)	2^{-166}
CNTR-1024[21]	NTRU+RLWR	三项分圆环	1 536	1 280	2 816	(253, 230)	2^{-291}
CTRU-1024[21]	NTRU+RLWE	三项分圆环	1 536	1 408	2 944	(255, 231)	2^{-195}
Kyber-1024[11]	MLWE	2次幂分圆环	1 568	1 568	3 136	(256, 232)	2^{-174}

表 8 给出了 CNTR-Prime 跟其他方案的性能, 包括 NTRU 类型的密钥封装方案如 SNTRU-Prime、NTRU-HRSS[9]、NTRU-A[17]、BAT[18]、NTTRU[19]、LTRU[20]、CNTR/CTRU[21], 以及基于模格的密钥封装方案如 NIST 标准化方案 Kyber[11]. SNTRU-Prime、NTRU-HRSS 和 Kyber 的数据来自它们 NIST 第 3 轮算法文档. NTRU-A、BAT、NTTRU、LTRU 和 CNTR/CTRU 的数据来自它们的论文. 困难性假设中“NTRU-OW”指 NTRU 单向困难性假设[9], “{R, M}LWE”指{环, 模}带误差学习. $\lvert pk\rvert$ 为公钥尺寸, $\lvert ct\rvert$ 为密文尺寸, B.W.为带宽, 即 $\lvert pk\rvert+\lvert ct\rvert$, 它们均是以字节为单位. (C, Q) 给出了方案的安全强度, 其中“C”和“Q”分别指的是经典和量子情形下的安全强度, 单位为 bit. 多种困难性假设的情况下, 选取不同困难性假设下安全强度的最小值作为方案的安全强度. δ 表示方案的错误率.

8.5 方案的效率比较

表 9 比较了 CNTR-Prime 跟其他方案的实现效率. *KeyGen*, *Encaps* 和 *Decaps* 分别给出这些方案在密钥生成算法、密钥封装算法和解封装算法运行 10 000 次的 CPU 周期的中位数和均值. 本文主要比较它们的中位数, 除非在部分方案只有均值数据的情况下才比较均值数据.

表 9 方案的实现效率比较 (kcycle)

方案/参数集	*KeyGen*		*Encaps*		*Decaps*	
	中位数	均值	中位数	均值	中位数	均值
CNTR-Prime-653 (本文)	4 333.4	4 427.1	71.2	76.2	127.8	133.6
CNTR-Prime-761 (本文)	5 789.5	5 930.1	82.8	87.9	148.8	155.7
CNTR-Prime-1277 (本文)	16 611.9	17 121.1	140.2	147.1	254.6	269.9
SNTRU-Prime-653*	111 323.8	112 414.9	782.4	825.0	268.5	273.3
SNTRU-Prime-761*	146 986.5	148 486.2	898.4	940.7	301.0	311.2
SNTRU-Prime-1277*	427 548.3	424 661.1	1 776.2	1 807.4	517.6	528.0
SNTRU-Prime-653[10]	127 920.5	129 787.3	11 358.1	11 517.3	33 376.9	35 883.8
SNTRU-Prime-761[10]	165 880.6	166 342.3	15 048.9	15 228.5	53 343.5	51 366.2
SNTRU-Prime-1277[10]	491 281.7	492 688.6	51 193.7	51 199.9	154 108.3	158 737.6
NTRU-HRSS*	63 171.6	65 098.6	410.8	417.7	1 082.1	1 105.6
NTRU-HRSS[9]	86 563.7	88 227.6	2 448.7	2 560.3	8 120.6	8 360.4
NTTRU-SHA3[19]	121.0	124.9	78.4	81.6	110.9	117.7
LTRU[20]	—	88.9	—	61.5	—	108.8
CNTR-768[21]	—	118.4	—	64.9	—	133.1
CTRU-768[21]	—	117.5	—	63.4	—	134.6
Kyber-768[11]	128.7	131.1	127.0	131.2	151.3	154.7
Kyber-1024[11]	189.4	195.1	185.5	190.1	268.3	269.6

注: SNTRU-Prime*和NTRU-HRSS*使用本文的伪梅森数不完整NTT计算多项式乘法, 但求逆算法保持不变. “—”表示无数据

SNTRU-Prime 和 NTRU-HRSS 的实现测试数据是将它们开源 C 代码在跟 CNTR-Prime 同一平台运行所得到的. 为了尽量减少不同多项式乘法算法差异导致的影响, 本文将伪梅森数不完整 NTT 同样应用在 SNTRU-Prime 和 NTRU-HRSS 之中, 得到它们的实现变体 SNTRU-Prime*和 NTRU-HRSS*, 其中本文在 NTRU-HRSS 中应用 $\mathbb{Z}_{33550337}[x]/(x^{1536}-1)$ 上的伪梅森数不完整 NTT 并在最后进行 $\bmod x^{701}-1$. 并从表 9 中可知, 应用了本文提出的伪梅森数不完整 NTT 之后, SNTRU-Prime 的 3 组参数集在密钥生成、密钥封装和解封装算法的实现效率方面均有所提升. 具体而言, SNTRU-Prime*密钥生成算法快了 1.1–1.3 倍, 密钥封装算法快了 14.5–28.8 倍, 解封装算法快了 138.7–308.8 倍. 同时, NTRU-HRSS*在 3 种算法分别快了 1.3 倍、5.9 倍、7.5 倍. 在下文中 CNTR-Prime 跟 SNTRU-Prime 和 NTRU-HRSS 进行实现效率比较时, 均使用 SNTRU-Prime*和 NTRU-HRSS*的测试数据. 需要注意的是, 由于 NTTRU、LTRU、CNTR、CTRU 和 Kyber 的底层代数结构已经是 NTT 友好环, 它们能够直接使用 NTT 计算三项分圆环或 2 次幂分圆环中的多项式算法, 且从图 4 可知, 直接在 NTT 友好环上应用 NTT 计算多项式乘法的速度更快. 可见, 它们可以使用自身的 NTT, 而无需更换为本文的伪梅森数不完整 NTT.

另外, 为方便比较, 本文基于 Kyber 的开源代码, 按照文献 [47] 的方式将它的 FO 转换修改为 $\mathrm{FO}^{\perp}_{ID(pk),m}$, 即跟 CNTR-Prime 相同的 FO 转换, 并重测其 C 实现数据. 另外, 本文将 NTTRU 底层伪随机数生成器和哈希函数替换为跟 CNTR-Prime 相同的 SHA-3 系列函数, 得到它的实现变体 NTTRU-SHA3. 除此之外, LTRU 和 CTRU/CNTR-768 只给出实现测试数据而并没公开其源代码, 故表 9 中 LTRU 的实现测试数据取自文献 [20], CNTR/CTRU-768 的实现测试数据取自文献 [21] 由于 NTRU-A 和 BAT 缺少开源 C 代码和测试数据, 故表 9 中不展示它们的数据.

8.6 分析和结论

下面将基于表 8 和表 9 的数据, 对 CNTR-Prime 和其他方案展开比较和分析.

1) 和素阶数域方案 SNTRU-Prime 的比较

在带宽方面, CNTR-Prime 和 SNTRU-Prime 的公钥尺寸相同的情况下, CNTR-Prime-761/1277 的密文尺寸均小于 SNTRU-Prime-761/1277 的 (分别降低 8.3%、13.5%), 而 CNTR-Prime-653 的密文尺寸只比 SNTRU-Prime-653 的多 1 字节, 但 1 字节的差别对实际应用几乎没影响.

在安全强度方面, CNTR-Prime 的 3 组参数集的经典/量子安全强度都高于 SNTRU-Prime 的. 具体而言, CNTR-Prime-653 的经典和量子安全强度比 SNTRU-Prime-653 分别高 22 bit 和 19 bit; CNTR-Prime-761 的经典和量子安全强度比 SNTRU-Prime-761 都高 19 bit; CNTR-Prime-1277 的经典和量子安全强度比 SNTRU-Prime-1277 分别高 27 bit 和 25 bit. 尽管 SNTRU-Prime 具有零错误率, 但 CNTR-Prime 的错误率相对于它的量子安全强度而言是匹配的和可忽略的, 因此不影响安全性.

在实现效率方面, CNTR-Prime 比 SNTRU-Prime 更加高效. 具体而言, 对于密钥生成算法、密钥封装算法和解封装算法, CNTR-Prime-653 比 SNTRU-Prime-653*分别快了 25.6 倍、10.9 倍和 2.1 倍; CNTR-Prime-761 比 SNTRU-Prime-761*分别快了 25.3 倍、10.8 倍和 2.0 倍; CNTR-Prime-1277 比 SNTRU-Prime-1277*分别快了 25.7 倍、12.6 倍和 2.0 倍. 其中, CNTR-Prime 的密钥生成算法比 SNTRU-Prime 快了超过 25 倍的原因是, CNTR-Prime 只需计算 $\mathcal{R}_q$ 中一次多项式求逆, 而 SNTRU-Prime 需要计算 $\mathcal{R}_q$ 和 $\mathcal{R}_3$ 中总共 2 次求逆, 并且 $\mathcal{R}_3$ 中元素的逆元并不是恒存在的, 需要判断元素的可逆性, 若不可逆则需要重新执行密钥生成算法, 使得 SNTRU-Prime 密钥生成算法更加耗时. 另外, 尽管 CNTR-Prime 和 SNTRU-Prime*的密钥封装算法和解封装算法均使用伪梅森数不完整 NTT, 但 CNTR-Prime 的构造更加简单, 计算量更少, 所以耗时更少.

实际上, 与 SNTRU-Prime-761 相比, CNTR-Prime-653 的经典和量子安全强度已可与 SNTRU-Prime-761 相媲美, 且 CNTR-Prime-653 的公钥尺寸降低 14.1%, 密文尺寸降低 13.5%, 总带宽降低 13.8%, 密钥生成算法、密钥封装算法和解封装算法分别快了 33.9 倍、12.6 倍和 2.3 倍.

2) 和其他 NTRU 类型 KEM 的比较

从表 8 和表 9 可知, CNTR-Prime-653 的安全强度和 NTRU-HRSS、NTRU-A_{2917}^{648}、NTTRU、BAT-512、LTRU 的较为接近; 而 CNTR-Prime-761 的安全强度和 CNTR-768、CTRU-768 的较为接近; CNTR-Prime-1277 和 BAT-1024、CNTR-1024、CTRU-1024 的较为接近, 所以本文分别将 CNTR-Prime-653/761/1277 跟这些方案进行比较. 除了 NTRU-HRSS 使用截断多项式环 $\mathbb{Z}_{2^{13}}[x]/(x^{701}-1)$, 其余 NTRU 类型 KEM 均使用分圆环. 尽管基于分圆环的 NTRU 类型 KEM 能够直接使用高效 NTT 计算多项式乘法从而运算速度更快, 但从第 3 节的分析可知, 基于素阶数域的 CNTR-Prime 比基于截断多项式环和分圆环的 NTRU 类型 KEM 具有更保守的安全性.

与 NTRU-HRSS 相比, CNTR-Prime-653 的经典和量子安全强度分别高 15 bit 和 12 bit; 带宽降低 16.8%; 密钥生成算法、密钥封装算法和解封装算法比 NTRU-HRSS*分别快了 14.5 倍、5.7 倍、8.4 倍.

与 NTRU-A_{2917}^{648} 相比, CNTR-Prime-653 的安全强度、带宽、错误率都跟 NTRU-A_{2917}^{648} 的接近, 然而在底层代数结构方面, CNTR-Prime-653 的底层代数结构的安全性更保守稳健.

与 NTTRU 相比, CNTR-Prime-653 的带宽降低 24.1%, 密钥封装算法耗时降低 8.9%. 不过 NTTRU 的密钥生成算法比 CNTR-Prime-653 的密钥生成算法快 35.8 倍.

与 BAT-512 和 LTRU 相比, 尽管它们的带宽更低, 但是 CNTR-Prime-653 的量子安全强度比它们分别高 12 bit 和 7 bit.

与 CNTR-768 和 CTRU-768 相比, CNTR-Prime-761 的带宽、密钥封装算法和解封装算法耗时跟它们的接近, 但是 CNTR-Prime-761 的安全强度和错误率稍逊色于 CNTR-768 和 CTRU-768 的.

与 BAT-1024、CNTR-1024 和 CTRU-1024 相比, CNTR-Prime-1024 具有更高的安全强度 (高于 19–39 bit 量子安全强度), 且错误率比 BAT-1024 和 CTRU-1024 的低.

3) 和模格方案 Kyber 的比较

与 Kyber-768 相比, CNTR-Prime-761 的错误率和带宽均和 Kyber-768 的接近, 量子安全强度低 10 bit, 但是 Kyber-768 基于 2 次幂分圆环, 易受针对分圆环的攻击 (如子域攻击、基于自同构和环同态的攻击); 而 CNTR-

Prime-761 基于素阶数域能够抵御这些攻击, 具体情况可见第 3 节. 至于实现效率方面, CNTR-Prime-761 在密钥封装算法快了 1.7 倍, 主要是因为 CNTR-Prime-761 的加密算法仅有一个多项式乘法而 Kyber-768 的加密算法需要计算更复杂的多项式矩阵-向量乘法; CNTR-Prime-761 在密钥生成算法中需要额外的非常耗时的多项式求逆运算, 因此 Kyber-768 比 CNTR-Prime-761 的密钥生成算法快 44.9 倍.

与 Kyber-1024 相比, CNTR-Prime-1277 的安全强度远高于 Kyber-1024 的. 具体而言, 经典和量子安全强度分别高 41 bit 和 37 bit. 另外, CNTR-Prime-1277 的错误率更低, 在密钥封装算法和解封装算法的实现效率分别快了 24.4% 和 5.1%.

总的来说, 与基于素阶数域的同类方案 SNTRU-Prime 相比, CNTR-Prime 在安全强度、带宽和实现效率上有优势. 与基于截断多项式环和分圆环的 NTRU 类型 KEM 相比, CNTR-Prime 在安全强度、带宽和错误率上性能均衡, 且底层代数结构具有更保守的安全性. 尽管 CNTR-Prime 的密钥生成算法更加耗时, 但由于公私钥对生成一次可多次使用, 因此现实中密钥生成算法的耗时影响较小, CNTR-Prime 的密钥生成算法完全能够适应多数的现实场景.

9 总结与展望

本文梳理了针对分圆环的相关攻击以及素阶数域的安全优势, 然后提出了 CNTR 在素阶数域上的变体方案 CNTR-Prime, 并给出推荐参数集. 针对 CNTR-Prime 中素阶数域上多项式乘法, 本文提出了伪梅森数不完整 NTT 和改进的伪梅森数约减. 未来工作有以下 3 个方向: 一是在多平台上对 CNTR-Prime 方案进行高效实现, 如 FPGA 实现; 二是基于素阶数域构造更多的密码方案; 三是使用伪梅森数不完整 NTT 加速其他基于非 NTT 友好环的方案.

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] 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

[3] 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

[4] 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]

[5] 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]

[6] 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]

[7] 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]

[8] Hoffstein J, Pipher J, Silverman JH. NTRU: A ring-based public key cryptosystem. In: Proc. of the 3rd Int'l Symp. on Algorithmic Number Theory. Portland: Springer, 1998. 267–288. [doi: 10.5555/648184.749737]

[9] Chen C, Danba O, Hoffstein J, Hülsing A, Rijneveld J, Schanck JM, Schwabe P, Whyte W, Zhang ZF. NTRU: Algorithm specifications and supporting documentation. 2019. https://www.ntru.org/f/ntru-20190330.pdf

[10] Bernstein DJ, Brumley BB, 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/ntruprime-20201007.pdf

[11] 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

[12] IEEE. IEEE 1363-2000 IEEE standard specifications for public-key cryptography. 2000. https://standards.ieee.org/ieee/1363/2049/

[13] Business Wire. Security innovation's NTRUEncrypt adopted as X9 standard for data protection. 2011. https://www.businesswire.com/news/home/20110411005309/en/Security-Innovations-NTRUEncrypt-Adopted-X9-Standard-Data

[14] Schanck JM, Whyte W, Zhang ZF. Circuit-extension handshakes for Tor achieving forward secrecy in a quantum world. Proc. on Privacy Enhancing Technologies, 2016, 2016(4): 219–236. [doi: 10.1515/popets-2016-0037]

[15] Augot D, Batina L, Bernstein DJ, Bos J, Buchmann J, Castryck W, Dunkelman O, Güneysu T, Gueron S, Hülsing A, Lange T, Mohamed MSE, Rechberger C, Schwabe P, Sendrier N, Vercauteren F, Yang BY. Initial recommendations of long-term secure post-quantum systems. 2015. https://pqcrypto.eu.org/slides/recommendations-20150907.pdf

[16] OpenSSH. OpenSSH release notes. 2022. https://www.openssh.com/releasenotes.html

[17] Duman J, Hövelmanns K, Kiltz E, Lyubashevsky V, Seiler G, Unruh D. A thorough treatment of highly-efficient NTRU instantiations. In: Proc. of the 26th IACR Int'l Conf. on Practice and Theory of Public-key Cryptography. Atlanta: Springer, 2023. 65–94. [doi: 10.1007/978-3-031-31368-4_3]

[18] Fouque PA, Kirchner P, Pornin T, Yu Y. BAT: Small and fast KEM over NTRU lattices. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2022, 2022(2): 240–265. [doi: 10.46586/tches.v2022.i2.240-265]

[19] Lyubashevsky V, Seiler G. NTTRU: Truly fast NTRU using NTT. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2019, 2019(3): 180–201. [doi: 10.13154/tches.v2019.i3.180-201]

[20] Liang ZC, Zheng JY, Zhao YL. An efficient and compact key encapsulation mechanism based on NTRU lattice. Journal of Computer Research and Development, 2024, 61(4): 1049–1069 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.202220980]

[21] Liang ZC, Fang BY, Zheng JY, Zhao YL. Compact and efficient KEMs over NTRU lattices. Computer Standards & Interfaces, 2024, 89: 103828. [doi: 10.1016/J.CSI.2023.103828]

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

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

[24] 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: ACM, 2016. 893–902. [doi: 10.5555/2884435.2884499]

[25] 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]

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

[27] Bernstein DJ, Chuengsatiansup C, Lange T, Vredendaa 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]

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

[29] Stehlé D, Steinfeld R. Making NTRU as secure as worst-case problems over ideal lattices. In: Proc. of the 30th Annual Int'l Conf. on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. Tallinn: Springer, 2011. 27–47. [doi: 10.5555/2008684.2008690]

[30] Jarvis K, Nevins M. ETRU: NTRU over the eisenstein integers. Designs, Codes and Cryptography, 2015, 74(1): 219–242. [doi: 10.1007/s10623-013-9850-3]

[31] Hülsing A, Rijneveld J, Schanck J, Schwabe P. High-speed key encapsulation from NTRU. In: Proc. of the 19th Int'l Conf. on Cryptographic Hardware and Embedded Systems. Taipei: Springer, 2017. 232–252. [doi: 10.1007/978-3-319-66787-4_12]

[32] Bernstein DJ. Multidigit multiplication for mathematicians. 2001. http://cr.yp.to/papers/m3-20010811-retypeset-20220327.pdf

[33] Weimerskirch A, Paar C. Generalizations of the Karatsuba algorithm for efficient implementations. 2006. https://eprint.iacr.org/2006/224

[34] Alkim E, Ducas L, Pöppelmann T, Schwabe P. Post-quantum key exchange: A new hope. In: Proc. of the 25th USENIX Conf. on Security Symp. Austin: USENIX Association, 2016. 327–343. [doi: 10.5555/3241094.3241120]

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

[36] 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]

[37] Gentry C. Fully homomorphic encryption using ideal lattices. In: Proc. of the 41st ACM Symp. on Theory of Computing. Bethesda: ACM, 2009. 169–178. [doi: 10.1145/1536414.1536440]

[38] Garg S, Gentry C, Halevi S. Candidate multilinear maps from ideal lattices. In: Proc. of the 32nd Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Athens: Springer, 2013. 1–17. [doi: 10.1007/978-3-642-38348-9_1]

[39] Ducas L, Van Woerden W. NTRU fatigue: How stretched is overstretched? In: Proc. of the 27th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Singapore: Springer, 2021. 3–32. [doi: 10.1007/978-3-030-92068-5_1]

[40] 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]

[41] 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]

[42] Chen H, Lauter K, Stange KE. Vulnerable Galois RLWE families and improved attacks. 2016. https://eprint.iacr.org/archive/2016/193/20160224:182837

[43] 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]

[44] Bernstein DJ. Fast norm computation in smooth-degree Abelian number fields. Research in Number Theory, 2023, 9(4): 82. [doi: 10.1007/s40993-022-00402-0]

[45] S-unit attacks. S-unit attacks: Norms. 2022. https://s-unit.attacks.cr.yp.to/norms.html

[46] Albrecht M, Bai S, Ducas L. A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes. In: Proc. of the 36th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2016. 153–178. [doi: 10.1007/978-3-662-53018-4_6]

[47] Duman J, Hövelmanns K, Kiltz E, Lyubashevsky V, Seiler G. Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix hashing. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security. New York: ACM, 2021. 2722–2737. [doi: 10.1145/3460120.3484819]

[48] 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]

[49] Bernstein DJ, Yang BY. Fast constant-time gcd computation and modular inversion. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2019, 2019(3): 340–398. [doi: 10.13154/tches.v2019.i3.340-398]

[50] Solinas J. Generalized Mersenne numbers. Research Report, CORR-99-39, Waterloo: University of Waterloo, 1999.

[51] Barrett P. Implementing the rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: Proc. of the 1987 Conf. on the Theory and Application of Cryptographic Techniques. Santa Barbara: Springer, 1987. 311–323. [doi: 10.1007/3-540-47721-7_24]

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

[53] 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]

[54] Bernstein DJ, Brumley BB, Chen MS, Tuveri N. OpenSSLNTRU: Faster post-quantum TLS key exchange. In: Proc. of the 31st USENIX Security Symp. Berkeley: USENIX Association, 2022. 845–862.

附中文参考文献:

[3] 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

[20] 梁志闯, 郑婕妤, 赵运磊. NTRU 格上高效紧凑密钥封装方案. 计算机研究与发展, 2024, 61(4): 1049–1069. [doi: 10.7544/issn1000-1239.202220980]

梁志闯(1997－), 男, 博士生, 主要研究领域为格密码.

方博越(1997－), 男, 学士, 主要研究领域为格密码.

赵旭阳(1994－), 男, 博士生, CCF 学生会员, 主要研究领域为后量子密码, 密码工程, 软硬件协同设计.

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, 计算理论.