计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202220980
61(4): 1049−1069, 2024

NTRU 格上高效紧凑密钥封装方案

梁志闯[1] 郑婕妤[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)
(zcliang21@m.fudan.edu.cn)

An Efficient and Compact Key Encapsulation Mechanism Based on NTRU Lattice

Liang Zhichuang[1], Zheng Jieyu[1], and Zhao Yunlei[1,2]
[1](*School of Computer Science*, *Fudan University*, *Shanghai* 200433)
[2](*State Key Laboratory of Cryptology*, *Beijing* 100036)

Abstract Constructing post-quantum key encapsulation mechanism based on NTRU lattice is one of the popular research fields in lattice-based cryptography. To reduce the ciphertext size, some current schemes compress the ciphertext with the aid of extra hardness assumptions and error correction codes, which leads to idealistic underlying assumption and complicated implementation. To address the issues, an efficient and compact key encapsulation mechanism, named LTRU, is proposed. LTRU is only based on NTRU one-wayness assumption and enables ciphertext compression without using any error correction codes. The performance-balanced parameter set of LTRU is provided, featuring 128 b quantum security level along with the matching and negligible error probability, and smaller public key size and ciphertext size. LTRU is based on the NTT-friendly polynomial ring. To compute the polynomial operations of LTRU, an efficient mixed-radix NTT is presented. At last, both C implementation and AVX2 implementation of LTRU are provided. When compared with NIST Round 3 finalist NTRU-HRSS, the classical and quantum security of LTRU are strengthened by 6 b and 5 b, respectively. LTRU reduces the public key size, ciphertext size and total bandwidth by 14.6%, 26.0% and 20.3%, respectively. LTRU is 10.9 times faster in key generation and 1.7 faster in decapsulation with respect to AVX2 implementation, respectively.

Key words post-quantum cryptography; NTRU; key encapsulation mechanism; ciphertext compression; number theoretic transform; AVX2 implementation

摘 要 基于NTRU格设计后量子密钥封装方案是格密码领域主流方向之一.为降低密文尺寸,现有方案会引入额外的困难性假设和使用纠错码来辅助压缩密文,但这会导致方案的假设过强和实现更复杂.为克服这些障碍,提出了一个仅基于NTRU单向困难性假设、不使用纠错码也能压缩密文的高效紧凑的密钥封装方案LTRU.给出一套性能均衡的LTRU参数集:具有128 b量子安全强度、与之匹配且可忽略的错误率、较小的公钥尺寸和密文尺寸.LTRU基于NTT友好环构造,给出一种高效的混合基数论变换算法来计算该环上多项式运算还给出了LTRU的C实现和AVX2实现.与NIST第3轮决赛方案NTRU-HRSS相

收稿日期: 2022-11-30; **修回日期**: 2023-05-15
基金项目: 国家自然科学基金项目(61877011); 国家重点研发计划项目(2022YFB2701600); 上海市科学技术发展基金项目(21DZ2200500); 山东省重点研发计划项目(2017CXG0701, 2018CXGC0701)
This work was supported by the National Natural Science Foundation of China (61877011), the National Key Research and Development Program of China (2022YFB2701600), the Shanghai Science and Technology Innovation Action Plan (21DZ2200500), and the Shandong Provincial Key Research and Development Program (2017CXG0701, 2018CXGC0701).
通信作者: 赵运磊(ylzhao@fudan.edu.cn)

比, LTRU 的经典安全强度和量子安全强度分别增强 6 b 和 5 b, LTRU 的公钥尺寸降低 14.6%, 密文尺寸降低 26.0%, 总带宽降低 20.3%; 在 AVX2 实现的密钥生成和解封装算法上分别快了 10.9 倍和 1.7 倍.

关键词 后量子密码; NTRU; 密钥封装方案; 密文压缩; 数论变换; AVX2 实现

中图法分类号 TP309

密码学是现代网络安全的基石. 现阶段广泛使用的公钥密码体制, 如公钥加密、密钥封装、数字签名等, 多数是基于经典的困难问题构造的, 如求解大整数分解、(椭圆)离散对数等困难问题. 这些密码方案在当今互联网协议, 如 SSL/TLS, IPSec 等中扮演了无可替代的角色, 如保证通信内容的机密性、通信双方的身份认证、消息的完整性等. 尽管针对这些数论问题目前最好的经典求解算法的时间复杂度是指数或亚指数级别, 至少是超多项式级别的, 但近些年来随着量子计算的飞速发展, 已有相关研究表明存在多项式时间的量子算法可以完全求解它们. 例如, 美国科学家 Shor[1] 提出可在量子计算机中以多项式时间求解大整数分解和(椭圆)离散对数问题的量子算法. 量子计算相关技术的发展日新月异. 考虑到量子计算技术对现有公钥密码的颠覆性威胁, 目前很多国家、公司和学术界已提前研究和布局能够抵抗量子攻击的密码学——后量子密码学(post-quantum cryptography, PQC).

事实上, 早在 2016 年, 美国国家标准技术研究所(National Institute of Standards and Technology, NIST)已经正式开启了对公钥加密(public key encryption, PKE)方案、密钥封装方案(key encapsulation mechanism, KEM)和数字签名这 3 种密钥原语的后量子密码方案标准征集活动, 且目前已经评选出第 3 轮结果并选定拟标准化的方案[2]. 我国也在 2019 年举行后量子密码算法竞赛, 并于 2020 年评选出获奖方案[3].

目前, 后量子密码学的类型主要分为: 基于哈希的、基于多变量的、基于同源的、基于编码的以及基于格的. 其中基于格构造的后量子密码方案不仅具有平均情形到最糟情形的困难性规约, 还在安全性、通信带宽和计算效率等方面表现均衡. 在美国 NIST 和我国举办的后量子密码方案征集活动中, 基于格的密码方案占据多数. 例如, NIST 第 1 轮 64 个方案中的 26 个[4]、第 2 轮 26 个方案中的 12 个[5]、第 3 轮 15 个方案中的 7 个[6] 以及拟标准化 4 个方案中的 3 个[2] 均为基于格构造的密码方案. 我国后量子密码算法设计竞赛获奖方案共 14 个, 其中有 11 个均为基于格构造的. 基于格的方案在设计 PKE 和 KEM 时主要基于一般格、理想格、模格和 NTRU 格. 常用的格困难问题主要分为 2 类: 第 1 类是 {R,M}LWE/LWR 问题[7−10]; 第 2 类是 NTRU 格相关问题[11−13].

NTRU 最早由 Hoffstein 等人[14] 于 1996 年美密会讨论环节中提出, 但它在 1997 年遭遇格攻击[15]. 随后 Hoffstein 等人对方案安全性方面进行补救, 并于 1998 年正式提出 NTRU 加密方案(NTRU-HPS)[11]. 该方案是第 1 个基于格困难问题构造的现实的公钥加密方案. NTRU 相关问题目前已经是许多密码方案中的基础元件[16−18]. 在 NIST 的后量子密码方案征集项目中, 基于 NTRU 格的方案具有举足轻重的地位. 具体而言, Falcon 数字签名[19] 基于 NTRU 格构造, 且是 NIST 后量子密码方案标准征集拟标准化的数字签名之一. 另外, 在 NIST 第 3 轮中, NTRU KEM[20] (包含 NTRU-HRSS 和 NTRUEncrypt)是 4 个决赛 KEM 方案之一, NTRU Prime KEM[21](包含 SNTRU Prime 和 NTRU LPRime)是 5 个候选方案之一. 尽管目前 NIST 选择了基于模格的 Kyber[22] 作为唯一的拟标准化 KEM 方案[2], 而基于 NTRU 格的方案没能被 NIST 选为拟标准化 KEM 方案, 但 NIST 官方报告声称如果 Kyber 相关专利问题未能得到很好地解决, 仍可能会放弃 Kyber 而启用 NTRU KEM[23]. 另外, 在 2011 年 NTRU-Encrypt 方案便已入选 X9.98 标准并应用于金融服务企业之中[24]. 自 2022 年 4 月起, 国际标准 OpenSSH 更新了版本 9.0 之后, OpenSSH 便采用了 NTRU Prime KEM 方案结合 X25519 ECDH 方案的混合模式, 以抵抗“先捕获后解密”攻击[25]. 另外, 有关基于 NTRU 格的方案的研究并不应因此而止步不前, 对基于 NTRU 格的方案的设计和分析理应得到更充分、深入和全面的研究.

基于 NTRU 格设计的 KEM 备受青睐, 主要原因有 3 点: 1)基于 NTRU 格的 KEM 具有坚固的安全保障. 自 NTRU 被提出至今, 有关 NTRU 的攻击和密码分析对基于 NTRU 格的 KEM 安全性产生的影响非常有限. 事实上, 大多数基于 NTRU 相关困难问题设计的密码方案至今没被攻破. 2)有关 NTRU 的专利已失效. 在 1997 年, NTRU 加密系统获得了专利, 但在 2017 年 NTRU 相关核心专利已到期. 3)大多数基于 NTRU 格的 KEM 结构简单, 便于实现, 同时加解

密算法的运算速度快[20−21].

近些年来，针对基于 NTRU 格的 KEM 有 2 点可优化的方向：

1）计算效率方面. 基于 NTRU 格的 KEM 如 NTRU-HRSS[20] 和 SNTRU Prime[21] 在计算效率方面逊色于基于{R,M}LWE 的方案. 主要是因为后者能够使用高效的数论变换（number theoretic transform, NTT）来计算多项式乘法. 近些年来，文献 [26−27] 基于 NTT 友好的多项式环来构造出基于 NTRU 格的 KEM，使得方案中的多项式运算得到高效计算.

2）密文尺寸方面. 降低密文尺寸对网络协议，如 TLS 和 IoT 资源受限设备通信都具有积极的作用. 尽管密文压缩在基于 LWE 及其变体的 KEM 中是一项成熟的技术，但对基于 NTRU 格的 KEM 来说，目前相关研究极少. 常见的 NTRU 的加密方案的密文形式一般为 $c = phr + m \bmod q$，其中 h 为公钥（一般 $h = g/f$），q 为方案模数，g、f 为秘密多项式；r 为采样得到的秘密多项式，m 为待加密的明文，p 为明文空间模数. 这可视为 m 通过编码到密文 c 的低位. 解密算法将 c 乘以私钥 f，得到 $cf \bmod q = pgr + mf$. 随后通过乘 f 的逆，或当 $f = pf' + 1$ 时可通过模 p 消除杂项来恢复明文. 这种解密机制要求被消除的杂项为 p 的倍数. 所以一旦密文 c 被压缩，压缩带来的误差往往集中在 c 的低位，从而破坏 m 的有效信息，此时便无法正确解密得到原始的 m.

文献 [28−29] 对基于 NTRU 格的 KEM 的密文压缩进行了探索，但是文献 [28−29] 的方案有 2 点不足. 第 1 点不足是这些方案都基于 2 种困难性假设来构建 KEM.其中文献 [28] 是基于 NTRU 假设[11] 和 RLWE 假设[9]，文献 [29] 是基于 NTRU 假设和 RLWR 假设[8]. 引入过多的困难性假设会导致方案建立在更理想化的假设上. 同时，分析这些方案的安全强度则需要分析 2 种困难性问题的安全强度，并取 2 个安全强度的最小值. 所以，方案的安全性很大程度上取决于安全强度更低的困难性问题.NTRU 相关困难问题的安全性经过 20 多年的密码分析现今仍然安全；RLWE 和 RLWR 是较之更新的问题而需要进一步密码分析和验证. 为得到更为稳定的安全保障，仅基于 NTRU 相关困难问题设计的 KEM 值得深入地研究和探索. 第 2 点不足是文献 [28−29] 的方案均使用纠错码和复杂的纠错机制来恢复明文信息. 其中文献 [28] 使用可尺度 E_8 格纠错码；文献 [29] 使用改进的 Babai 算法. 尽管纠错码具有良好的纠错能力从而有效恢复信息，但是 KEM 使用纠错码会带来 2 个缺点：1）纠错码增加了方案遭遇侧信道攻击的风险[30]，特别是计时攻击[31]；2）纠错码导致实现更复杂、更耗时，且更容易出错. 为了设计更安全且便于实现的 KEM，设计者应该避免使用纠错码. 所以，对基于 NTRU 格的 KEM 来说，在不使用纠错码的情况下，如何得到可压缩密文的构造依然需要进一步探索.

在我国，后量子密码被我国科协列为 60 个我国亟待解决的重大科技难题之一. 尽管我国 2019 年密码算法设计竞赛的获奖方案大多数是基于格构造的，且包含基于 NTRU 格的 FatSeal 签名[3]，但是在它们之中并没有基于 NTRU 格的 PKE 和 KEM.在当前国内外网络空间安全愈发重要的大背景下，我国研究团队理应自主研制高性能、多样化的后量子密码方案. 考虑到基于 NTRU 格的方案在国际上（特别是 NIST 后量子密码标准征集项目）的重要地位，本文主要关注基于 NTRU 格构造高效的 PKE 和 KEM，并针对目前可优化方向，如计算效率方面和密文尺寸方面，提出一套基于 NTRU 格构造的可压缩密文的实用化公钥密码系统. 本文旨在为后续同类后量子密码方案的研究和优化提供重要参考. 同时，这方面的研究工作对我国未来的后量子密码标准的选拔、制定、推广和实际应用都具有重要的现实意义.

本文的贡献主要有 4 点：

1）基于 NTT 友好环 $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ 构造了一个基于 NTRU 格的可压缩密文的密钥封装方案 LTRU. 该方案包含了 IND-CPA 安全的公钥加密方案 LTRU. PKE 和 IND-CCA 安全的密钥封装方案 LTRU.KEM. 本文的 LTRU 仅基于 NTRU 单向困难性假设构建. LTRU 能够避免引入过多假设，从而避免方案假设过强和过于理想化. 同时 LTRU 不使用纠错码，从而实现更简单，且降低了遭遇侧信道攻击的风险. 已知 LTRU 是第 1 个仅基于 NTRU 类型假设且不使用纠错码也能压缩密文的密钥封装方案.

2）提供了针对 128 b 安全强度的 LTRU 参数集. 这使得 LTRU 不仅达到了目标安全强度（实际上达到 129 b 量子安全强度），还具有与安全性匹配的、可忽略的错误率（2^{-154}），同时通信带宽小于其他 NTRU 方案. 具体而言，与 NIST 第 3 轮决赛方案 NTRU-HRSS 相比，LTRU 的经典安全强度和量子安全强度分别增强 6 b 和 5 b，并且 LTRU 的公钥尺寸降低 14.6%，密文尺寸降低 26.0%，总带宽降低 20.3%.

3）提出了一种高效的混合基 NTT 算法. 该算法能够有效计算环 $\mathbb{Z}_{2\,917}[x]/(x^{648} - x^{324} + 1)$ 上的多项式运算，利用 3 次单位根的性质来有效减少基-3 FFT trick

步骤中一半的乘法数量，并利用 1-迭代 Karatsuba 算法来有效减少点乘中的乘法数量. 通过具体实现的效率分析可知，跟未使用 1-迭代 Karatsuba 算法和单位根的性质优化乘法数量的 NTT 算法相比，本文 NTT 在正向变换、点乘和逆向变换的 CPU 周期数分别降低 24.9%, 20.6%, 26.8%, 总共降低 25.3%.

4) 提供 LTRU 的 C 实现和 AVX2 实现. 实验结果表明，与 NTRU-HRSS 的 AVX2 实现相比，LTRU 在密钥生成和解封装算法上分别快了 10.9 倍和 1.7 倍，封装算法的速度则与 NTRU-HRSS 的速度相当.

1 相关工作

近些年来，陆续有基于 NTRU 格的方案被提出. Stehlé等人[32] 基于 2 次幂分圆环$\mathbb{Z}[x]/(x^n+1)$(其中n为 2 次幂)来设计基于 NTRU 变体的 PKE，并证明了公钥的安全性. 随后，Wang 等人[33] 将该方案推广至任意的分圆环. 但是这 2 个方案因为公钥尺寸过大而无法应用于实际的场景中. Jarvis 等人[34] 将 NTRU-HPS 推广至 Eisenstein 整数环$\mathbb{Z}[\omega]/(x^n-1)$(其中$\omega=\exp(2\pi\,\mathrm{i}/3)$). 该方案在密钥尺寸和性能方面比 NTRU-HPS 更有优势.Hülsing 等人[35] 全面提升 NTRU-HPS 的密钥尺寸、密文尺寸和效率，并提出了 NTRU-HRSS. NTRU-HRSS 曾是 NIST 后量子方案征集项目第 3 轮决赛方案之一[20]. Bernstein 等人[36] 考虑到 NTRU 类型方案使用的多项式环或因丰富的代数结构而遭受相关攻击，提出了使用代数结构更少的环$\mathbb{Z}_q[x]/(x^n-x-1)$的 NTRU 变体方案：NTRU Prime(其中n,q均为素数). NTRU-prime 曾是 NIST 后量子方案征集项目第 3 轮候选方案之一[20].

为了追求更高的实现效率，Lyubashevsky 等人[26] 在多项式环$\mathbb{Z}_{7681}[x]/(x^{768}-x^{384}+1)$上构造了针对 128 b 安全强度的 NTTRU 方案. 随后，Duman 等人[27] 将该方案推广至更一般的n并选择对应的q. 但是这些方案均是遵循传统 NTRU 的框架，并不能对密文进行压缩.

Fouque 等人[29] 提出 BAT 方案. BAT 与传统 NTRU 类型的 KEM 的构造有所区别，但与 Falcon 数字签名方案[19] 有众多相似之处. BAT 的私钥是一组陷门基，这一点与 Falcon 相似，但区别于其他 NTRU 类型 KEM 的私钥f. 直观地理解，BAT 使用的纠错码机制相当于对含有 2 个未知数的方程求解出秘密多项式和噪音多项式，并用来恢复明文. BAT 通过将密文构造为 RLWR 实例的方式来降低密文的尺寸. 然而，为生成陷门基作为私钥，BAT 的密钥生成算法速度比常见的 NTRU 类型 KEM 的速度慢约 1 000 倍. 其次，为降低密文尺寸而构造的 RLWR 实例使用二元的秘密分布(即秘密多项式的系数选自{0,1})，而关于该问题的安全性目前还是公开问题且还需要进一步研究. 另外，BAT 缺少匹配 128 b 安全强度的参数集，因为它使用 2 次幂分圆环$\mathbb{Z}[x]/(x^n+1)$，且$n=512$和$n=1\,024$，而前者的安全强度略低于 128 b，后者的安全强度则远高于 128 b.

2 预备知识

本节主要介绍一些符号说明、基本定义和密码原语等.

2.1 符号说明和基本定义

1) 本文记$\mathbb{Z}$为整数环，n和q为某些整数. 定义集合$\mathbb{Z}_q=\mathbb{Z}/q\mathbb{Z}\cong\{0,1,\cdots,q-1\}$. 对于实数$x\in\mathbb{R}$，符号$\lfloor x\rceil$表示对$x$四舍五入后的值. 本文主要使用分圆多项式环$\mathcal{R}:=\mathbb{Z}[x]/(x^n-x^{n/2}+1)$及其商环$\mathcal{R}_q:=\mathcal{R}/q\mathcal{R}=\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，其中$n=2^i3^j$，$i\geqslant 0$，$j>1$，且此时$x^n-x^{n/2}+1$是$3n$阶分圆多项式. 环$\mathcal{R}$(或$\mathcal{R}_q$)中的元素均是多项式，本文使用符号$f$的幂级数形式$f=\sum_{i=0}^{n-1}f_ix^i$，$f_i\in\mathbb{Z}$(或$f_i\in\mathbb{Z}_q$). 一个函数$\varepsilon:\mathbb{N}\to[0,1]$是可忽略的，指的是对任意的正数$c$以及充分大的$\lambda$，$\varepsilon(\lambda)<1/\lambda^c$成立.

2) 模约减. 对于某正整数q，符号$r'=r \bmod q$表示r落在$[0,q)\cap\mathbb{Z}$中的代表元r'. 符号$r'=r \bmod^{\pm} q$表示r落在$\left[-\frac{q}{2},\frac{q}{2}\right)\cap\mathbb{Z}$中的代表元$r'$. 另外，符号$r'\equiv r \bmod q$表示$r'$和$r$模$q$同余.

3) 元素的范数[37]. 对于元素$v\in\mathbb{Z}_q$，定义它的ℓ_∞范数$\|v\|_\infty$为$\left|v \bmod^{\pm} q\right|$. 对于环$\mathcal{R}$(或$\mathcal{R}_q$)中的元素$w$，它的$\ell_\infty$范数为$\|w\|_\infty=\max_i\|w_i\|_\infty$.

4) 压缩函数和解压缩函数[37]. 对于正整数q和d，定义压缩函数

$$Compress_q(x,d)=\left\lfloor\frac{2^d}{q}x\right\rceil \bmod 2^d,\ x\in\mathbb{Z}_q,$$

以及解压缩函数

$$Decompress_q(y,d)=\left\lfloor\frac{q}{2^d}y\right\rceil \bmod q,\ y\in\mathbb{Z}_{2^d},$$

且对于任意的$x\in\mathbb{Z}_q$，有

$$\left\|x-Decompress_q(Compress_q(x,d),d)\right\|_\infty\leqslant\left\lfloor\frac{q}{2^{d+1}}\right\rceil.$$

5) 集合和分布. 对于集合D，记$x\leftarrow\$D$表示从$D$中均匀随机选取$x$. 如果$D$是一个概率分布，$x\leftarrow D$表示根据分布$D$采样得到$x$. 本文主要使用的分布$\bar{B}_\eta$为：

采样$(a_1,a_2,\cdots,a_\eta,b_1,b_2,\cdots,b_\eta)\leftarrow\$\{0,1\}^{2\eta}$，然后输出$\left[\sum_{i=1}^{\eta}(a_i-b_i)\right] \bmod^{\pm}3$. 根据分布$D$，采样多项式$f$指的是$f$的每一个系数均根据$D$采样得到.

2.2 密码原语

1）公钥加密方案

一个公钥加密方案 PKE 包含 *KeyGen, Enc, Dec* 3 个概率多项式时间（probabilistic polynomial time, PPT）算法，记其明文空间为$\mathcal{M}$. 密钥生成算法 *KeyGen* 输出公私钥对(pk,sk). 加密算法 *Enc* 输入公钥pk和明文$m\in\mathcal{M}$，输出密文ct. 本文在必要时使用$Enc(pk,m;coin)$显式表示加密算法使用的随机数为$coin$. 确定性的解密算法 *Dec* 输入密文ct和私钥sk，输出$m\in\mathcal{M}$，或者$\perp$表示解密失败. 公钥加密方案的解密错误率δ定义为

$$E[\max_{m\in\mathcal{M}}Pr[Dec(sk,Enc(pk,m))\neq m]]<\delta, \tag{1}$$

其中期望作用于$(pk,sk)\leftarrow KeyGen$上，式（1）概率取自加密算法 *Enc* 使用的随机数. 一个公钥加密方案是抗原像恢复意义下的选择明文 PRE-CPA（preimage resistance under chosen-plaintext attacks）安全的[27]，指的是对于任意 PPT 敌手$\mathcal{A}$，其优势是可忽略的：

$$Adv_{\mathrm{PKE}}^{\mathrm{PRE\text{-}CPA}}(\mathcal{A})=Pr\left[c=c^*:\begin{array}{c}(pk,sk)\leftarrow KeyGen;\\ m^*\leftarrow\$\mathcal{M};\\ c^*\leftarrow Enc(pk,m^*);\\ (m,coin)\leftarrow\mathcal{A}(pk,c^*);\\ c:=Enc(pk,m;coin)\end{array}\right].$$

一个公钥加密方案是不可区分意义下的选择明文 IND-CPA（indistinguishability under chosen-plaintext attacks）安全的，指的是对于任意 PPT 敌手$\mathcal{A}$，其优势是可忽略的：

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

2）密钥封装方案

一个密钥封装方案 KEM 包含 *KeyGen, Encaps, Decaps* 3 个 PPT 算法. 记其导出密钥空间为$\mathcal{K}$. 密钥生成算法 *KeyGen* 输出公私钥对(pk,sk). 密钥封装算法 *Encaps* 输入公钥pk，输出密文ct和密钥$K\in\mathcal{K}$. 确定性的解封装算法 *Decaps* 输入密文ct和私钥sk，输出$K\in\mathcal{K}$，或者$\perp$表示解封装失败. 密钥封装方案的错误率δ定义为

$$Pr\left[Decaps(sk,ct)\neq K:(ct,K)\leftarrow Encaps(pk)\right]<\delta,$$

其中概率取自$(pk,sk)\leftarrow KeyGen$和封装算法 *Encaps* 使用的随机数. 一个密钥封装方案是不可区分意义下的选择密文 IND-CCA（indistinguishability under chosen-ciphertext attacks）安全的，指的是对于任意 PPT 敌手$\mathcal{A}$，其优势是可忽略的：

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

2.3 NTRU 单向困难性假设

定义 1. NTRU 单向困难性假设[12-13]. 给定 NTRU 公钥加密方案的公钥h以及 PPT 加密算法 *Enc*，则 NTRU 单向函数是指

$$Enc:\mathcal{L}_r\times\mathcal{L}_e\rightarrow\mathcal{L}_c,$$
$$(r,e)\mapsto c=Enc(h,(r,e)),$$

其中$Enc(h,(r,e))$指的是以h，e，r作为加密算法的输入，在 NTRU 公钥加密方案中$\mathcal{L}_e$一般是它的明文空间，$\mathcal{L}_r$一般是它的随机数空间.

NTRU 单向问题 NTRU-OW（NTRU one-wayness）指的是给定公钥h和密文多项式c，计算得到(r,e). NTRU 单向问题是困难的，指的是对于任意 PPT 敌手$\mathcal{A}$，其优势是可忽略的：

$$Adv^{\mathrm{NTRU\text{-}OW}}(\mathcal{A})=\left|Pr\left[Enc(pk,(r,e))=c^*:\begin{array}{c}(pk,sk)\leftarrow KeyGen;\\ (r^*,e^*)\leftarrow\mathcal{L}_r\times\mathcal{L}_e;\\ c^*:=Enc(pk,(r^*,e^*));\\ (r,e)\leftarrow\mathcal{A}(pk,c^*)\end{array}\right]\right|.$$

通俗来讲，NTRU 单向困难性假设是指没有 PPT 敌手能够从公钥h和密文c中恢复出(r,e). 原始的 NTRU-HPS 方案[11]便蕴含了 NTRU 单向困难性假设，直到文献 [12−13] 将 NTRU 单向困难性假设提炼出来，并应用在基于 NTRU 格的方案（如 NAEP 方案）的安全性证明之中.

2.4 ACWC$_0$ 转换

定义 2. ACWC$_0$ 转换[27]. 记PKE_1=($KeyGen_1$, Enc_1, Dec_1)为公钥加密方案，其明文空间记为$\mathcal{M}_1$. 令$\mathcal{F}:\{0,1\}^*\rightarrow\{0,1\}^\lambda$为哈希函数，$\lambda$为某正整数.ACWC$_0$ 通过算法 1~3 将PKE_1=($KeyGen_1$, Enc_1, Dec_1)转换得到另一个公钥加密方案PKE_2=($KeyGen_2$, Enc_2, Dec_2)，其明文空间为$\mathcal{M}_2=\{0,1\}^\lambda$.

算法 1. 密钥生成算法 $KeyGen_2$.

输入：安全参数κ；

输出：公私钥对(pk,sk).

① $(pk,sk)\leftarrow KeyGen_1$.

算法 2. 加密算法 Enc_2.

输入：公钥pk，明文$m\in\mathcal{M}_2$；

输出：密文$ct:=(c,u)$.

① $m'\leftarrow \$\mathcal{M}_1$；

② $c:=Enc_1(pk,m')$；

③ $u:=\mathcal{F}(m')\oplus m$.

算法 3. 解密算法 Dec_2.

输入：私钥sk，密文$ct=(c,u)$；

输出：明文$m\in\mathcal{M}_2$.

① $m':=Dec_1(sk,c)$；

② $m:=\mathcal{F}(m')\oplus u$.

根据文献 [27] 中的引理 2.2 和定理 3.3，PKE_2 方案在随机预言机模型（random oracle model, ROM）[38] 下的 IND-CPA 安全性由定理 1 给出.

定理 1. 在随机预言机模型下，对于任意的攻击 PKE_2 的 IND-CPA 安全性的敌手$\mathcal{A}$，存在攻击 PKE_1 的 PRE-CPA 安全性的敌手$\mathcal{B}$，其运行时间与$\mathcal{A}$相当，使得

$$Adv^{\text{IND-CPA}}_{PKE_2}(\mathcal{A})\leqslant Adv^{\text{PRE-CPA}}_{PKE_1}(\mathcal{B}).$$

另外，根据文献 [27] 中的引理 2.2 和定理 3.4，可得定理 2 关于 PKE_2 方案在量子随机预言机模型（quantum random oracle model, QROM）[39] 下的 IND-CPA 安全性.

定理 2. 在量子随机预言机模型下，对于任意的攻击 PKE_2 的 IND-CPA 安全性的量子敌手$\mathcal{A}$，存在攻击 PKE_1 的 PRE-CPA 安全性的量子敌手$\mathcal{B}$，其运行时间与$\mathcal{A}$相当，使得

$$Adv^{\text{IND-CPA}}_{PKE_2}(\mathcal{A})\leqslant 2d_{\mathcal{F}}\sqrt{Adv^{\text{PRE-CPA}}_{PKE_1}(\mathcal{B})},$$

其中$d_{\mathcal{F}}$是$\mathcal{A}$查询$\mathcal{F}$的深度.

可见，ACWC_0 转换能够将一个 PRE-CPA 安全的公钥加密方案转换得到另一个 IND-CPA 安全的公钥加密方案. 本文将在 3.1 节中提出的公钥加密方案便是通过 ACWC_0 转换得到的.

2.5 数论变换（NTT）

NTT 是快速傅里叶变换（fast Fourier transform, FFT）在有限域上的特殊形式.NTT 是计算高次多项式乘法最高效的算法，其复杂度为$O(n\log n)$，其中n为被乘多项式的维度. 简单地说，基于 NTT 计算多项式乘法$h=fg$的过程为$h=\mathcal{INTT}(\mathcal{NTT}(f)\circ\mathcal{NTT}(g))$，其中$\mathcal{NTT}$表示正向变换，$\mathcal{INTT}$表示逆向变换，“$\circ$”表示点乘. 本文沿用文献 [40] 有关多项式乘法的符号和术语，FFT trick 是计算 NTT 的一种快速算法，其计算过程本质是基于多项式环形式的中国剩余定理（Chinese remainder theorem, CRT），即给定两两互素的多项式$g_1,g_2,\cdots,g_k$，有 CRT 同构

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

并且$\varphi(f)=(f \bmod g_1, f \bmod g_2,\cdots,f \bmod g_k)$. 记$\zeta$在$\mathbb{Z}_q$中可逆. 对于经典的基-2 FFT trick 的情况，有 CRT 同构：

$$\mathbb{Z}_q[x]/(x^{2m}-\zeta^2)\cong\mathbb{Z}_q[x]/(x^m-\zeta)\times\mathbb{Z}_q[x]/(x^m+\zeta).$$

正向 FFT trick 基于 CT（Cooley-Tukey）蝴蝶操作[41] 从(f_i,f_j)得到$(f_i+\zeta f_j,f_i-\zeta f_j)$，而逆向 FFT trick 基于 GS（Gentleman-Sande）蝴蝶操作[42] 从(f_i',f_j')得到$((f_i'+f_j'),(f_i'-f_j')\zeta^{-1})$. 对于经典的基-3 FFT trick，有 CRT 同构：

$$\mathbb{Z}_q[x]/(x^{3m}-\zeta^3)\cong\mathbb{Z}_q[x]/(x^m-\zeta)\times\mathbb{Z}_q[x]/(x^m-\rho\zeta)\times\mathbb{Z}_q[x]/(x^m-\rho^2\zeta),$$

其中ρ是 3 次单位根. 混合基 NTT 指的是在计算 NTT 过程含有多种 FFT trick.

2.6 Karatsuba 算法

定义 3. 1-迭代 Karatsuba 算法[43]. 设a,b,c,d为某些数或多项式.1-迭代 Karatsuba 算法指的是为了计算$t_1=ac$，$t_2=ad+bc$和$t_3=bd$，可以先计算t_1和t_3，然后通过$t_2=(a+b)(c+d)-t_1-t_3$计算t_2. 该算法能在增加 3 个加（减）法的代价上，减少 1 个乘法.

3 本文方案及其分析

下面介绍本文提出的基于 NTRU 格构造的可压缩密文的密钥封装方案 LTRU. LTRU 包括一个 IND-CPA 安全的公钥加密方案 LTRU.PKE，以及一个 IND-CCA 安全的密钥封装方案 LTRU.KEM，其中该 KEM 通过 FO（Fujisaki-Okamoto）转换[44-45] 得到. LTRU.PKE 是由底层公钥加密方案$PKE'=(KeyGen, Enc', Dec')$结合文献 [27] 的 ACWC_0 转换得到的. 底层公钥加密方案PKE'将在 3.4 节中展示，但它只能满足 PRE-CPA 安全性，并且目前还没有相关文献论证 FO 转换将 PRE-CPA 安全的公钥加密方案转换得到 IND-CCA 安全的 KEM 的规约上界. 本文通过增加 ACWC_0 转换，将底层公钥加密方案PKE'转换得到 IND-CPA 安全的 LTRU.PKE，然后便能通过 FO 转换得到 IND-CCA 安全的 KEM.

3.1 LTRU 的构造

LTRU.PKE 的具体构造见算法 4~6.记其明文空间为$\mathcal{M}=\{0,1\}^{\lambda}$，其中$\lambda$为正整数. 记多项式环为$\mathcal{R}_q=\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，其中$n$和$q$是环参数. 记$d$和$p$为正

整数，且p和q互素. 需要说明的是，在算法 4~6 中，为了给出一般的构造框架，本文使用的分布ψ泛指$\mathcal{R}$上的某分布，但事实上，在 3.3 节的具体参数选择方面，本文实例化ψ为参数$\eta=2$的$\bar{B}_\eta$分布（定义见 2.1 节）. 令$\mathcal{F}:\{0,1\}^*\to\{0,1\}^\lambda$为哈希函数. 本文主要考虑$p=2$以及$\lambda=256$的情况. 可视$\{0,1\}^n$中的元素是维度为$n$、系数为 0 或 1 的多项式. 压缩函数 *Compress* 和解压缩函数 *Decompress* 的具体计算过程见 2.1 节.

算法 4. 密钥生成算法 *LTRU.PKE.KeyGen*.

输入：安全参数κ；

输出：公私钥对(pk,sk).

① repeat

② $f',g\leftarrow\psi$；

③ $f:=pf'+1$；

④ until f^{-1} exists in $\mathcal{R}_q$；

⑤ $h:=g/f$；

⑥ $pk:=h$, $sk:=f$.

算法 5. 加密算法 *LTRU.PKE.Enc*.

输入：公钥pk, 明文$m\in\mathcal{M}$；

输出：密文$ct:=(c,u)$.

① $r\leftarrow\psi$, $e\leftarrow\$\{0,1\}^n$；

② $c:=Compress_q(hr+\left\lfloor\frac{q}{2}\right\rceil e,d)$；

③ $u:=m\oplus\mathcal{F}(e)$；

④ $ct:=(c,u)$.

算法 6. 解密算法 *LTRU.PKE.Dec*.

输入：私钥sk, 密文$ct:=(c,u)$；

输出：明文$m\in\mathcal{M}$.

① $c':=Decompress_q(c,d)$；

② $e':=Compress_q(c'f \bmod^{\pm}q,1)$；

③ $m:=u\oplus\mathcal{F}(e')$.

可见，LTRU.PKE 应用了 ACWC_0 转换[27]，见算法 5 行③以及算法 6 行③. LTRU.KEM 是由 LTRU.PKE 通过$\mathrm{FO}_m^{\perp}$转换得到，其中$\mathrm{FO}_m^{\perp}$是文献 [45] 提出的 FO 转换的一个变体. LTRU.KEM 的密钥生成算法 *KeyGen* 与 LTRU.PKE 的 *KeyGen* 相同. LTRU.KEM 的封装算法和解封装算法分别见算法 7 和算法 8.记$\mathcal{K}$为 LTRU.KEM 的导出密钥空间，为方便起见，本文令$\mathcal{K}=\{0,1\}^{256}$. 记$\mathcal{COINS}$为 LTRU 公钥加密方案中加密算法所使用的随机数空间. 记$\mathcal{H}:\{0,1\}^*\to\mathcal{K}\times\mathcal{COINS}$为哈希函数. 实际上，$\mathcal{H}$能够拆分为 2 部分$\mathcal{H}_1$和$\mathcal{H}_2$，记$\mathcal{H}_1$为$\mathcal{H}$映射到$\mathcal{COINS}$的部分，$\mathcal{H}_2$为$\mathcal{H}$映射到$\mathcal{K}$的部分. 在算法 7 的行②中可以先使用$\mathcal{H}_1$生成$coin$，在行③中计算得到密文$c$之后，再使用$\mathcal{H}_2$生成共享密钥$K$.

算法 7. 封装算法 *LTRU.KEM.Encaps*.

输入：公钥pk；

输出：密钥K, 密文c.

① $m\leftarrow\$\mathcal{M}$；

② $(K,coin):=\mathcal{H}(m)$；

③ $c:=LTRU.PKE.Enc(pk,m;coin)$.

算法 8. 解封装算法 *LTRU.KEM.Decaps*.

输入：私钥sk, 密文c；

输出：密钥K.

① $m':=LTRU.PKE.Dec(sk,c)$；

② $(K',coin'):=\mathcal{H}(m')$；

③ if $c=LTRU.PKE.Enc(pk,m';coin')$and $m'\neq\perp$

④ return K'；

⑤ else

⑥ return $\perp$；

⑦ end if

3.2 错误率分析

定理 3. 对于$\mathcal{R}$上的分布ψ、正整数d和q, 令$f',g,r\leftarrow\psi$, 以及$\varepsilon\leftarrow\chi$, 其中$\chi$为$\mathcal{R}$上的分布且定义为：采样$h\leftarrow\$\mathcal{R}_q,r\leftarrow\psi,e\leftarrow\$\{0,1\}^n$,输出$\left(\left\lfloor\frac{q}{2^d}\left\lfloor\frac{2^d}{q}Z\right\rceil\right\rceil-Z\right)\bmod^{\pm}q$, 其中$Z=hr+\left\lfloor\frac{q}{2}\right\rceil e$. 记

$$1-\delta=Pr\left[\left\|gr+\left(\frac{e}{2}+\varepsilon\right)f\right\|_\infty<\frac{q}{4}\right],$$

则 LTRU 的错误率为δ.

证明.

从 2.1 节的压缩函数 *Compress* 和解压缩函数 *Decompress* 的定义可知，算法 6 行①的c'可以表示为

$$c'=Decompress_q(Compress_q(Z,d),d)=Z+\varepsilon,$$

其中$Z=hr+\left\lfloor\frac{q}{2}\right\rceil e$, $\varepsilon\in\mathcal{R}$. 所以，算法 6 行②的$e'$为

$$\begin{aligned}e'=&Compress_q(c'f\bmod^{\pm}q,1)=\\&\left\lfloor\frac{2}{q}[(Z+\varepsilon)f\bmod^{\pm}q]\right\rceil\bmod 2=\\&\left\lfloor\frac{2}{q}\left[\left(hr+\frac{q+1}{2}e+\varepsilon\right)f\bmod^{\pm}q\right]\right\rceil\bmod 2.\end{aligned}$$

由于 LTRU 中$h=g/f$和$f=2f'+1$, 则可得

$$\begin{aligned}e'=&\frac{2}{q}\left\lfloor\left[\left(hrf+\frac{q}{2}ef+\frac{1}{2}ef+\varepsilon f\right)\bmod^{\pm}q\right]\right\rceil\bmod 2=\\&\left\lfloor\frac{2}{q}\left[hrf+\frac{q}{2}e(2f'+1)+\frac{1}{2}ef+\varepsilon f\bmod^{\pm}q\right]\right\rceil\bmod 2=\\&\left\lfloor e+\frac{2}{q}\left(gr+\frac{1}{2}ef+\varepsilon f\right)\right\rceil\bmod 2.\end{aligned}$$

于是，当$\left\|\frac{2}{q}\left(gr+\frac{1}{2}ef+\varepsilon f\right)\right\|_\infty<\frac{1}{2}$时，可得$e'=e$

恒成立，继而解密算法能够成功解密以及解封装算法能够正确地解封装. 等价地，正确解密的条件为

$$\left\| gr+(\frac{e}{2}+\varepsilon)f \right\|_\infty < \frac{q}{4}.$$

证毕.

本文计算错误率的方法论和 Python 脚本源自文献 [22, 26, 37]. 值得注意的是，文献 [26] 将环$\mathbb{Z}[x]/(x^n-x^{n/2}+1)$上的多项式的乘积的系数视为由$n/2$个形如$ab+b'(a+a')$的元组构成，其中$a$, a', b, b'的分布由被乘多项式的分布决定. 尽管该环上的多项式的乘积另外还包含形如$ab+a'b'$的元组，文献 [26] 建议全部使用$ab+b'(a+a')$的形式，因为该形式具有更宽泛的分布，且能得到更保守估计的错误率结果.

3.3 参数选择

本节给出 LTRU 的推荐参数集，其主要针对 128 b 安全强度，如表 1 所示. $\mathcal{R}_q$是多项式环，维数n和模数q是环参数. 本文选择 NTT 友好的q，旨在q允许在$\mathcal{R}_q$上执行高效的多项式乘法和除法，具体细节见第 4 节. p是明文空间模数；d是压缩参数；ψ是概率分布，本文主要考虑分布$\bar{B}_2$，即参数为$\eta=2$的$\bar{B}_\eta$分布，$\bar{B}_\eta$的定义见 2.1 节；$|pk|$, $|ct|$, *B.W.*分别是公钥尺寸、密文尺寸和带宽(公钥尺寸+密文尺寸)，单位均为 B；Sec-(C, Q)是该参数集的 NTRU 安全强度(单位为 b)，C 表示经典安全强度，Q 表示量子安全强度，安全强度的分析见 3.5 节；δ是该参数集的错误率，其细节见 3.2 节.

Table 1 Recommended Parameter Set of LTRU

表 1 LTRU 的推荐参数集

$\mathcal{R}_q$	n	q	p	d	ψ	$\|pk\|$/B	$\|ct\|$/B	*B.W.*/B	Sec-(C, Q)/b	δ
$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$	648	2 917	2	10	$\bar{B}_2$	972	842	1 814	(142, 129)	2^{-154}

3.4 安全规约

下面将基于 NTRU 单向困难性假设，证明 LTRU.PKE 是 IND-CPA 安全的.

定理 4. IND-CPA 安全性. 对于任意的概率多项式时间敌手$\mathcal{A}$，存在概率多项式时间敌手$\mathcal{B}$，其运行时间与$\mathcal{A}$相当，使得：

1) $Adv^{\text{IND-CPA}}_{\text{LTRU.PKE}}(\mathcal{A}) \leqslant Adv^{\text{NTRU-OW}}(\mathcal{B})$，其中$\mathcal{A}$为经典敌手，哈希函数$\mathcal{F}$被建模为经典随机预言机；

2) $Adv^{\text{IND-CPA}}_{\text{LTRU.PKE}}(\mathcal{A}) \leqslant 2d_{\mathcal{F}}\sqrt{Adv^{\text{NTRU-OW}}(\mathcal{B})}$，其中$\mathcal{A}$为量子敌手，哈希函数$\mathcal{F}$被建模为量子随机预言机，且$d_{\mathcal{F}}$为$\mathcal{A}$查询$\mathcal{F}$的深度.

证明. LTRU.PKE 方案由底层公钥加密方案$PKE'=(KeyGen, Enc', Dec')$结合文献 [27] 的 ACWC$_0$ 转换得到，其中底层公钥加密方案PKE'的$KeyGen$算法与 LTRU.PKE 方案的一致，Enc'算法和Dec'算法分别见算法 9 和算法 10.

算法 9. 加密算法$PKE'.Enc'$.

输入：公钥pk，明文$e\in\{0,1\}^n$；

输出：密文c.

① $r\leftarrow\psi$；

② $c:=Compress_q(hr+\left\lfloor\frac{q}{2}\right\rfloor e, d)$.

算法 10. 解密算法$PKE'.Dec'$.

输入：私钥sk，密文c；

输出：明文e'.

① $c':=Decompress_q(c,d)$；

② $e':=Compress_q(c'f \bmod^{\pm} q, 1)$.

由定理 1 和定理 2 可知，当底层公钥加密方案PKE'满足 PRE-CPA 安全性时，经过 ACWC$_0$ 转换得到的 LTRU.PKE 方案满足 IND-CPA 安全性. 下面将通过 Game-Hopping 技术[46]来证明底层公钥加密方案PKE'的 PRE-CPA 安全性.

记$\mathcal{A}$是攻击PKE'方案的 PRE-CPA 安全性的概率多项式时间敌手. 考虑游戏G_0和G_1. 记事件$Succ_i$为敌手$\mathcal{A}$在游戏G_i中获胜，即游戏G_i的输出满足$c=c^*$.

1)游戏G_0

① repeat

② $f', g\leftarrow\psi$；

③ $f:=pf'+1$；

④ until f^{-1} exists in$\mathcal{R}_q$；

⑤ $h:=g/f$；

⑥ $r^*\leftarrow\psi$, $e^*\leftarrow\$\{0,1\}^n$；

⑦ $c^*:=Compress_q\left(hr^*+\left\lfloor\frac{q}{2}\right\rfloor e^*, d\right)$；

⑧ $(e,r)\leftarrow\mathcal{A}(pk,c^*)$；

⑨ $c:=Enc'(pk,e;r)=Compress_q\left(hr+\left\lfloor\frac{q}{2}\right\rfloor e, d\right)$；

⑩ if $c=c^*$ then

⑪ return 1；

⑫ else

⑬ return 0；

⑭ end if

游戏G_0是关于PKE'方案原始的 PRE-CPA 游戏.

根据 PRE-CPA 安全性的定义，可知

$$Adv^{\text{PRE-CPA}}_{\text{PKE}'}(\mathcal{A}) = Pr[Succ_0].$$

2）游戏G_1

① repeat

② $f', g \leftarrow \psi$;

③ $f := pf' + 1$;

④ until f^{-1} exists in $\mathcal{R}_q$;

⑤ $h := g/f$;

⑥ $r^* \leftarrow \psi$, $e^* \leftarrow \$\{0,1\}^n$;

⑦ $c^* := hr^* + \left\lfloor \frac{q}{2} \right\rceil e^*$;

⑧ $(e, r) \leftarrow \mathcal{A}(pk, c^*)$;

⑨ $c := Enc'(pk, e; r) = hr + \left\lfloor \frac{q}{2} \right\rceil e$;

⑩ if $c = c^*$ then

⑪ return 1;

⑫ else

⑬ return 0;

⑭ end if

游戏G_1是在游戏G_0的基础上，取消对行⑦挑战密文c^*的压缩操作. 所以，敌手$\mathcal{A}$在游戏G_1中拥有的挑战密文的信息至少比在游戏G_0中的多. 可见，游戏G_0的困难性至少与游戏G_1的相同，则

$$Pr[Succ_0] \leqslant Pr[Succ_1].$$

如果存在一个概率多项式时间敌手$\mathcal{A}$赢得游戏G_1，则可以构造一个概率多项式时间敌手$\mathcal{B}$解决 NTRU 单向困难性问题. 具体地，运行游戏G_1中的行①~⑦，将得到的h和c^*给敌手$\mathcal{B}$. 则敌手$\mathcal{B}$的目标是在给定公钥h和挑战密文c^*的情况下输出一对(e,r)，使得$c^* = Enc'(pk,(e,r)) = hr + \left\lfloor \frac{q}{2} \right\rceil e$成立. 敌手$\mathcal{B}$将$h$和$c^*$作为$\mathcal{A}$的输入并运行$\mathcal{A}$. 敌手$\mathcal{A}$输出一对$(e,r)$，然后敌手$\mathcal{B}$以$(e,r)$作为自己的输出. 可见，敌手$\mathcal{B}$完美模拟了游戏$G_1$. 如果发生事件$Succ_1$，说明敌手$\mathcal{A}$输出的$(e,r)$能够使得$c^* = hr + \left\lfloor \frac{q}{2} \right\rceil e$成立，即敌手$\mathcal{A}$赢得游戏$G_1$. 于是敌手$\mathcal{B}$以该$(e,r)$作为输出能够成功求解 NTRU 单向困难性问题. 否则，如果不发生事件$Succ_1$，说明敌手$\mathcal{A}$输出的(e,r)不能让$c^* = hr + \left\lfloor \frac{q}{2} \right\rceil e$成立. 于是，敌手$\mathcal{B}$以该$(e,r)$作为输出，不能成功求解 NTRU 单向困难性问题. 所以，敌手$\mathcal{A}$赢得游戏G_1的优势等于敌手$\mathcal{B}$求解该 NTRU 单向困难性问题的优势，即

$$Pr[Succ_1] = Adv^{\text{NTRU-OW}}(\mathcal{B}).$$

整合游戏G_0和G_1中的概率可知，对于概率多项式时间敌手$\mathcal{A}$，存在概率多项式时间敌手$\mathcal{B}$，使得

$$Adv^{\text{PRE-CPA}}_{PKE'}(\mathcal{A}) = Pr[Succ_0] \leqslant Pr[Succ_1] \leqslant Adv^{\text{NTRU-OW}}(\mathcal{B}).$$

再根据定理 1 和定理 2，由PKE'方案的 PRE-CPA 安全性可得到 LTRU.PKE 的 IND-CPA 安全性，其对应的规约上界正如定理 4 所示. 综上，命题得证. 证毕.

从定理 4 可知，若 NTRU 单向困难性问题是困难的，那么 LTRU.PKE 是 IND-CPA 安全的. 由于 LTRU.KEM 是从 IND-CPA 安全的 LTRU.PKE 通过$\text{FO}^{\perp}_m$变换得到，根据文献 [45, 47]，LTRU.KEM 在经典随机预言机和量子随机预言机下的 IND-CCA 安全性由定理 5 给出.

定理 5. IND-CCA 安全性[45,47]. 在记γ和δ分别是 LTRU.PKE 的弱 spread 参数[45,47]和解密错误率. 对于任意攻击 LTRU.KEM 的 IND-CCA 安全性的概率多项式时间敌手$\mathcal{A}$，存在攻击 LTRU.PKE 的 IND-CPA 安全性的概率多项式时间敌手$\mathcal{B}$，使得：

1）在经典随机预言机模型下，将$\mathcal{H}$建模为经典随机预言机，$\mathcal{A}$查询至多$q_{\mathcal{H}}$次$\mathcal{H}$，查询至多q_{D}次解封装预言机，$\mathcal{B}$的运行时间和$\mathcal{A}$的相当，则有

$$Adv^{\text{IND-CCA}}_{\text{LTRU.KEM}}(\mathcal{A}) \leqslant 2\left(Adv^{\text{IND-CPA}}_{\text{LTRU.PKE}}(\mathcal{B}) + \frac{q_{\mathcal{H}}}{|\mathcal{M}|}\right) + \frac{q_{\text{D}}}{2^{\gamma}} + q_{\mathcal{H}}\delta.$$

2）在量子随机预言机模型下，$\mathcal{A}$和$\mathcal{B}$为量子敌手，且将$\mathcal{H}$建模为量子随机预言机，$\mathcal{A}$以量子方式查询至多$q_{\mathcal{H}}$次$\mathcal{H}$，以经典方式查询至多q_{D}次解封装预言机，则有

$$Adv^{\text{IND-CCA}}_{\text{LTRU.KEM}}(\mathcal{A}) \leqslant 2q_{\text{T}}\sqrt{Adv^{\text{IND-CPA}}_{\text{LTRU.PKE}}(\mathcal{B})} + 24q_{\text{T}}^2\sqrt{\delta} + 24q_{\text{T}}\sqrt{q_{\text{T}}q_{\text{D}}} \times 2^{-\gamma/4},$$

其中$q_{\text{T}} := 2(q_{\mathcal{H}} + q_{\text{D}})$，并且

$$Time(\mathcal{B}) \approx Time(\mathcal{A}) + O(q_{\mathcal{H}}q_{\text{D}}Time(LTRU.PKE.Enc) + q_{\text{T}}^2).$$

3.5 安全强度

对于基于 NTRU 格构造的 KEM 来说，格攻击是目前最有效的攻击. 本节主要从常用的格攻击来分析 LTRU 的安全强度.

1）NTRU 格

起初，基于 NTRU 格构造的方案均是依赖于多项式环上的困难性问题. 然而，Coppersmith 等人[15]提出针对基于 NTRU 格构造方案的格攻击，它的核心思想是构造 NTRU 格. 本文方案的公钥对应的 NTRU 格$\mathcal{L}(\boldsymbol{B}) \subset \mathbb{Z}^{2n}$的基矩阵为

$$\boldsymbol{B} = \begin{pmatrix} q\boldsymbol{I}_n & \boldsymbol{Rot}(h) \\ \boldsymbol{0} & \boldsymbol{I}_n \end{pmatrix},$$

其中$\boldsymbol{I}_n$是n维单位矩阵，$\boldsymbol{Rot}(h)$是h的循环移位矩阵，即它的第i行向量对应$x^i h \bmod x^n - x^{n/2} + 1$. 文献 [15] 提出，秘密$(f,g)$及其循环移位向量是格$\mathcal{L}(\boldsymbol{B})$的最短

向量. 事实上，NTRU 类型方案的格攻击流程为：通过格基约化算法求解格$\mathcal{L}(\boldsymbol{B})$的唯一最短向量问题（unique-short vector problem, u-SVP），找到该最短向量之后转化得到方案的私钥[15].

2）原始攻击

原始攻击通过构造一个整数嵌入格，如 Kannan 嵌入[48]和 Bai-Galbraith 嵌入[49]等来求解 u-SVP 问题. 目前主要的格基约化算法为 BKZ 算法[50-51]. 给定格$\mathcal{L}(\boldsymbol{B})$的一组基，BKZ 算法运行时将格划分为多个低维度格，并把这些低维度格的维度记为b. 然后在b维格中求解最短向量问题（short vector problem, SVP），所使用的方法有筛法和枚举. 根据文献 [52] 的 core-SVP 方法论，对于b维格上求解 SVP 问题，目前最优的经典算法的复杂度为$2^{0.292b}$，最优的量子算法的复杂度为$2^{0.265b}$. core-SVP 方法论将这些复杂度视为求解b维格 SVP 问题的复杂度，并将得到的 BKZ 算法的复杂度作为原始攻击的复杂度估计. 本文将基于 core-SVP 方法论给本文方案提供一个保守估计的安全强度，并基于文献 [22, 37, 52] 提供的计算安全强度的 Python 脚本来计算 LTRU 在原始攻击下的经典安全强度和量子安全强度，其详细结果如表 1 所示.

3）其他攻击

对偶攻击主要通过将判定型 LWE 问题规约为短整数解问题（short integer solution problem, SIS）的方式来求解判定型 LWE 问题. 由于对偶攻击不适用于 NTRU 类型的方案[53]，故本文不考虑对偶攻击对 LTRU 的影响. 过度拉伸 NTRU 攻击[54]的适用范围为模数q远大于秘密多项式系数的情况. 由于 LTRU 的模数q数值较小，不满足该情况，故本文不考虑该攻击. 混合攻击[55]是格攻击和中间相遇攻击的提升版本，但混合攻击对于有着稀疏分布的秘密多项式系数和噪音多项式系数的方案能取得显著效果. 然而对于 LTRU，混合攻击不如原始攻击有效，因为 LTRU 的秘密(f,g)并非稀疏分布，不符合混合攻击的条件.

3.6 分析和讨论

1）多项式环

本文 LTRU 方案使用第$3n$个分圆多项式环$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，其中$n$的形式为$3^l\times 2^e$，$q$为某些素数. 主要原因是，当选择合适的$q$，如 NTT 友好的$q$时，存在高效的 NTT 算法计算该环上的多项式乘法和除法. 同时，n的选择具有更大的灵活性，因为n能够选择 576, 768, 864, 972, 1 152, 1 296 等数值. 当选择其他的n值时，LTRU 能够达到其他的安全强度（如 192 b 和 256 b），而不再局限于 128 b 的安全强度. 但是，本文主要针对 128 b 的安全强度，故选择n=648. 相比而言，NTRU-HRSS[20]使用多项式环$\mathbb{Z}_q[x]/(x^n-1)$，其中n为素数，q为 2 次幂；SNTRU Prime[21]使用多项式环$\mathbb{Z}_q[x]/(x^n-x-1)$，其中$n$，$q$均为素数. 这些多项式环不是 NTT 友好环，所以这些多项式环上的多项式乘法和除法更复杂，且耗时更多.

2）明文空间模数p

与其他 NTRU 方案如文献 [20−21, 26−27] 中的方案不同，本文 LTRU 使用的明文空间模数$p=2$，且只需出现在私钥f中，即$f=pf'+1$，而无需在公钥h以及解密算法中出现. 但是，NTRU 方案[20-21,26-27]使用模数$p=3$. 甚至，由于 NTRU 类型方案要求q和p互素，而 NTRU-HRSS[20]由于使用 2 次幂的模数q，所以它使用了与q互素的最小整数$p=3$.

3）纠错机制

常见的 NTRU 类型方案[20-21,26-27]的密文形式为$c=phr+m \bmod q$，这可以视为明文m编码在密文c的低位. 其恢复明文m的方式主要通过计算$cf \bmod q$之后再模p来消去杂项. 这种纠错机制依赖于杂项是p的倍数. 与之不同的是，LTRU 在底层公钥加密方案 PKE′中首先将明文m提升$q/2$倍，相当于把明文m编码到密文c的高位，随后在解密算法中只要求杂项的范数小于$1/2$，而不必要求其为p的倍数. 同时可以看到，LTRU 不使用纠错码来恢复明文. 纠错码虽然能够高效地实现纠错来恢复明文，但是纠错码使得代码实现更复杂且增加遭遇侧信道攻击的风险，如文献 [30] 针对纠错码提出了有效的侧信道攻击.

4）密文压缩

LTRU 的密文压缩操作见算法 5 的行②. 本文的 LTRU 是第 1 个仅基于 NTRU 单向困难性假设构造的、不使用纠错码也能够压缩密文的密钥封装方案. LTRU 的密钥压缩参数d可根据需求自行调整. 对于其他 NTRU 类型方案[20-21,26-27]，如果它们压缩密文，那么压缩过程会在密文的低位带来误差，而编码在密文低位的明文m的有效信息会被这些误差破坏，使得正确的明文m难以恢复出来；同时在解密算法中杂项也不再是p的倍数，所以也无法通过模p来消去杂项以恢复正确的明文m.

4 高效的混合基数论变换

本节将介绍本文提出的一种混合基 NTT 算法，其能高效地计算本文推荐参数集 LTRU-648 所使用的环$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，$n=648$，$q=2\,917$上的多项式

乘法和除法，且利用了 1-迭代 Karatsuba 算法和单位根的性质减少乘法的数量，以提高计算效率.

对于$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，$n=648$，$q=2\,917$，可知$\mathbb{Z}_q$中存在$3n/2$次本原单位根$\zeta=2$，其中$\zeta$为$k$次本原单位根是指$\zeta^k\equiv 1 \bmod q$且$\zeta^i\neq 1 \bmod q$，$0<i<k$.

4.1 正向变换

为方便理解，图 1 展示了混合基 NTT 对应的 CRT 同构树形图. 本文的混合基 NTT 的计算过程主要依靠逐层 CRT 分解.

类似于文献 [26] 提出的第 1 层 CRT 同构，对于$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，有

$$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)\cong \mathbb{Z}_q[x]/(x^{n/2}-\zeta_1)\times\mathbb{Z}_q[x]/(x^{n/2}-\zeta_2), \tag{2}$$

其中$\zeta_1+\zeta_2=1$和$\zeta_1\zeta_2=1$. 容易求得，$\zeta_2=1-\zeta_1$和$\zeta_2=\zeta_1^5$. 取$\zeta_1=\zeta^{n/4} \bmod q$以及$\zeta_2=\zeta^{5n/4} \bmod q$，对于任意的$f\in\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，可得

$$\begin{cases} f \bmod x^{n/2}-\zeta_1=(f_0+\zeta_1 f_{n/2})+\cdots+(f_{n/2-1}+\zeta_1 f_{n-1})x^{n/2-1},\\ f \bmod x^{n/2}-\zeta_2=(f_0+f_{n/2}-\zeta_1 f_{n/2})+\cdots+\\ (f_{n/2-1}+f_{n-1}-\zeta_1 f_{n-1})x^{n/2-1}. \end{cases} \tag{3}$$

由于$\zeta_1 f_{n/2}$只需计算 1 次而可使用 2 次，故式(2)(3)总计只需$n/2$个乘法和n个加(减)法.

1) 基-2 FFT trick 步骤

可利用 CRT 对$\mathbb{Z}_q[x]/(x^{n/2}-\zeta_1)$和$\mathbb{Z}_q[x]/(x^{n/2}-\zeta_2)$进行基-2 FFT trick 步骤的分解. 对于每个基-2 FFT trick 步骤形如

$$\mathbb{Z}_q[x]/(x^{2m}-r^2)\cong\mathbb{Z}_q[x]/(x^m-r)\times\mathbb{Z}_q[x]/(x^m+r),$$

其中r为ζ的某次幂，对于$f'\in\mathbb{Z}_q[x]/(x^{2m}-r^2)$，可得

$$\begin{cases} f_l=f' \bmod x^m-r=(f'_0+rf'_m)+\cdots+(f'_{m-1}+rf'_{2m-1})x^{m-1},\\ f_r=f' \bmod x^m+r=(f'_0-rf'_m)+\cdots+(f'_{m-1}-rf'_{2m-1})x^{m-1}, \end{cases}$$

其中每个rf'_i只需计算 1 次而可使用 2 次. 这使得每层的基-2 FFT trick 步骤所需乘法数量从n个锐减到$n/2$个.

2) 基-3 FFT trick 步骤

对于$n=648$，基-2 FFT trick 步骤可进行 1 层，直至得到 4 个形如$\mathbb{Z}_q[x]/(x^{162}-\zeta^{81})$的项. 此时本文采用基-3 FFT trick. 每个基-3 FFT trick 步骤均形如

$$\mathbb{Z}_q[x]/(x^{3m}-r^3)\cong\mathbb{Z}_q[x]/(x^m-r)\times\mathbb{Z}_q[x]/(x^m-\rho r)\times \mathbb{Z}_q[x]/(x^m-\rho^2 r),$$

其中r为ζ的某次幂，$\rho=\zeta^{n/3} \bmod q$为 3 次单位根. $\mathbb{Z}_q[x]/(x^{3m}-r^3)$中的多项式$f''$可以表示为

$$f''=f_a+f_b x^m+f_c x^{2m},$$

其中f_a,f_b,f_c均为$m-1$次多项式. 由于$\rho^3=1 \bmod q$，可得

$$\begin{cases} f_x=f'' \bmod x^m-r=f_a+rf_b+r^2 f_c,\\ f_y=f'' \bmod x^m-\rho r=f_a+\rho r f_b+\rho^2 r^2 f_c,\\ f_z=f'' \bmod x^m-\rho^2 r=f_a+\rho^2 r f_b+\rho r^2 f_c. \end{cases}$$

注意到$\rho^2+\rho+1=0 \bmod q$，则$\rho^2=-\rho-1 \bmod q$. 类似于文献 [56]，此时有

$$\begin{aligned} f_y&=f_a+\rho r f_b+\rho^2 r^2 f_c=\\ &f_a+\rho r f_b+(-\rho-1)r^2 f_c=\\ &f_a-r^2 f_c+\rho(r f_b-r^2 f_c), \end{aligned}$$

以及

$$\begin{aligned} f_z&=f_a+\rho^2 r f_b+\rho r^2 f_c=\\ &f_a+(-\rho-1)r f_b+\rho r^2 f_c=\\ &f_a-r f_b-\rho(r f_b-r^2 f_c). \end{aligned}$$

于是可得

$$\begin{cases} f_x=f_a+rf_b+r^2 f_c,\\ f_y=f_a-r^2 f_c+\rho(rf_b-r^2 f_c),\\ f_z=f_a-rf_b-\rho(rf_b-r^2 f_c). \end{cases}$$

注意到，$rf_b, r^2 f_c, \rho(rf_b-r^2 f_c)$都只需要计算 1 次而能够多次使用. 并且只需存储和使用本原单位根r和r^2以及 3 次单位根ρ，而无需存储其他值如ρr，$\rho^2 r^2$，$\rho^2 r$，ρr^2. 易知，使用 3 次单位根的性质之后，每层的基-3 FFT trick 步骤所需乘法数量从$2n$个锐减到n个.

正向变换中基-3 FFT trick 步骤可进行 4 层，直到将$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$分解得到$n/2$个模 2 次多项式的环. 换句话说，即有

$$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)\cong\prod_{i=0}^{n/2-1}\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)}),$$

其中$\tau(i)$表示第i个环中ζ的幂且序号从 0 开始. 对于$f\in\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，其正向变换的结果记为

$$\hat{f}=(\hat{f}_0,\hat{f}_1,\cdots,\hat{f}_{n/2-1}),$$

其中$\hat{f}_i\in\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)})$为线性多项式.

为方便后文书写，本文使用符号$\mathcal{NTT}$表示上述的正向变换过程. 正向变换$\mathcal{NTT}$的伪代码见算法 11.

算法 11. 正向变换$\mathcal{NTT}$.

输入：多项式$f\in\mathcal{R}_q$的系数数组$\{f[i]\}_{i=0}^{n-1}$，按照树形图 1 从上而下排列的本原单位根$\{\zeta[k]\}_{k=1}^{n/2}$，3 次单位根ρ；

输出：$\hat{f}=\mathcal{NTT}(f)$的系数数组$\{\hat{f}[i]\}_{i=0}^{n-1}$.

① $k:=1$;

② for $j:=0$, $j<n/2$, $j:=j+1$ do

/*进行首层 CRT 分解*/

③ $t:=\zeta[k]\times f[j+n/2]$;

④ $f[j+n/2]:=f[j]+f[j+n/2]-t$;

⑤ $f[j]:=f[j]+t$;

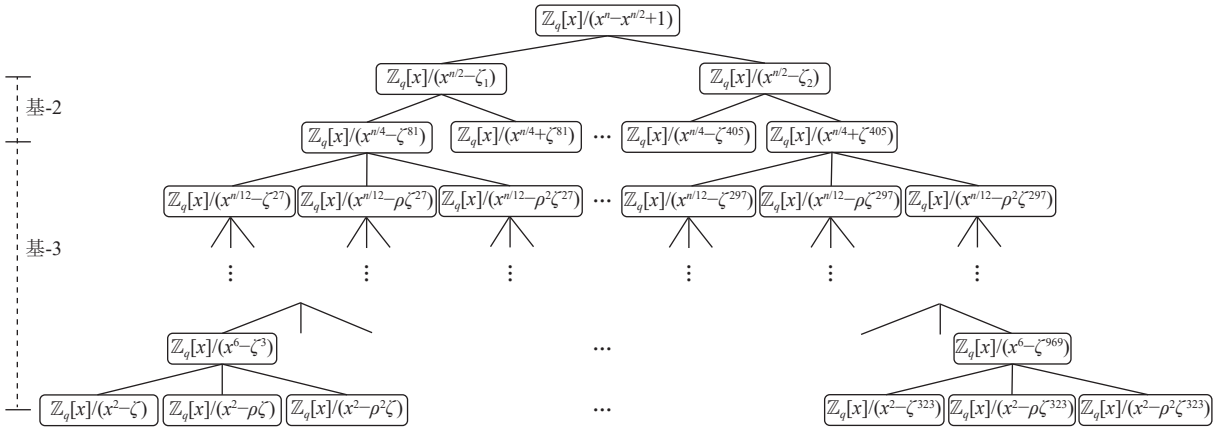


Fig. 1 The tree map of CRT isomorphism

图 1 CRT 同构的树形图

⑥ end for

⑦ $k := k+1$;

⑧ for $s := 0$, $s < n$, $s := j + n/4$ do

/*进行 1 层基-2 正向 FFT trick*/

⑨ for $j := s$, $j < s + n/4$, $j := j+1$ do

⑩ $t := \zeta[k] \times f[j+n/4]$;

⑪ $f[j+n/4] := f[j] - t$; /*CT 蝴蝶*/

⑫ $f[j] := f[j] + t$;

⑬ end for

⑭ $k := k+1$;

⑮ end for

⑯ for $l := n/12$, $l \geqslant 2$, $l := l/3$ do

/*进行 4 层基-3 正向 FFT trick*/

⑰ for $s := 0$, $s < n$, $s := s + 3l$ do

⑱ for $j := s$, $j < s + l$, $j := j+1$ do

⑲ $u := \zeta[k] \times f[j+l]$;

⑳ $v := \zeta[k+1] \times f[j+2l]$;

㉑ $w := \rho(u - v)$;

㉒ $f[j+2l] := f[j] - u - w$;

㉓ $f[j+l] := f[j] - v + w$;

㉔ $f[j] := f[j] + u + v$;

㉕ end for

㉖ $k := k+2$;

㉗ end for

㉘ end for

4.2 逆向变换

逆向变换可由正向变换的逆过程得到. 本文使用符号$\mathcal{INTT}$表示逆向变换. 值得注意的是, 在计算基-3 逆向 FFT trick 时, 同样可利用 3 次单位根的性质减少乘法的数量. 沿用 4.1 节的符号和$\mathbb{Z}_q[x]/(x^{3m} - r^3)$示例, 在基-3 逆向 FFT trick 中, 从f_x, f_y, f_z计算得到f''. 由$f'' = f_a + f_b x^m + f_c x^{2m}$可知, 这等价于得到$f_a, f_b, f_c$. 先计算

$$\begin{cases} \tilde{f_a} = f_x + f_y + f_z, \\ \tilde{f_b} = f_x + \rho^2 f_y + \rho f_z, \\ \tilde{f_c} = f_x + \rho f_y + \rho^2 f_z. \end{cases}$$

利用$\rho^2 = -\rho - 1 \bmod q$, 进一步可得

$$\begin{aligned} \tilde{f_b} =& f_x + \rho^2 f_y + \rho f_z = \\ & f_x + (-\rho - 1) f_y + \rho f_z = \\ & f_x - f_y - \rho(f_y - f_z), \end{aligned}$$

以及

$$\begin{aligned} \tilde{f_c} =& f_x + \rho f_y + \rho^2 f_z = \\ & f_x + \rho f_y + (-\rho - 1) f_z = \\ & f_x - f_z + \rho(f_y - f_z). \end{aligned}$$

于是可得

$$\begin{cases} \tilde{f_a} = f_x + f_y + f_z, \\ \tilde{f_b} = f_x - f_y - \rho(f_y - f_z), \\ \tilde{f_c} = f_x - f_z + \rho(f_y - f_z). \end{cases}$$

注意到, $\rho(f_y - f_z)$只需要计算 1 次而使用 2 次. 进一步可从$\tilde{f_a}, \tilde{f_b}, \tilde{f_c}$分别得到$f_a, f_b, f_c$为

$$\begin{cases} f_a = 3^{-1} \times \tilde{f_a}, \\ f_b = 3^{-1} \times r^{-1} \times \tilde{f_b}, \\ f_c = 3^{-1} \times r^{-2} \times \tilde{f_c}, \end{cases}$$

其中倍数3^{-1}可延迟处理. 故易知, 基于 3 次单位根的性质, 每层的基-3 逆向 FFT trick 步骤所需乘法数量从$2n$个锐减到n个.

另外, 对于基-2 逆向 FFT trick, 在此沿用 4.1 节的示例, 从$f_l = f' \bmod x^m - r$和$f_r = f' \bmod x^m + r$计算得到$f' \in \mathbb{Z}_q[x]/(x^{2m} - r^2)$的步骤为

$$f'=\sum_{i=0}^{m-1}\frac{f_{l,i}+f_{r,i}}{2}x^{i}+\sum_{i=0}^{m-1}\frac{f_{l,i}-f_{r,i}}{2r}x^{i+m},$$

其中$f_{l,i}$和$f_{r,i}$分别表示f_l和f_r的第i个系数，倍数2^{-1}同样可延迟处理. 易知，每层的基-2 逆向 FFT trick 步骤共需要$n/2$个乘法. 再合计上基-3 逆向 FFT trick 步骤延迟处理的倍数，在整个逆向变换结束之时，需要乘以总的倍数$(n/2)^{-1} \bmod q$.

逆向变换$\mathcal{INTT}$的伪代码见算法 12.

算法 12. 逆向变换$\mathcal{INTT}$.

输入：多项式$\hat{f}$的系数数组$\{\hat{f}[i]\}_{i=0}^{n-1}$，按照树形图 1 从下而上排列的本原单位根的逆$\{\zeta[k]\}_{k=1}^{n/2}$，3 次单位根$\rho$；

输出：$f=\mathcal{INTT}(\hat{f})$的系数数组$\{f[i]\}_{i=0}^{n-1}$.

① $k:=1$;
② for $l:=2$, $l\leqslant n/12$, $l:=3l$ do
/*进行 4 层基-3 逆向 FFT trick*/
③ for $s:=0$, $s<n$, $s:=s+3l$ do
④ for $j:=s$, $j<s+l$, $j:=j+1$ do
⑤ $u:=\rho(\hat{f}[j+l]-\hat{f}[j+2l])$;
⑥ $v:=\hat{f}[j]-\hat{f}[j+l]-u$;
⑦ $w:=\hat{f}[j]-\hat{f}[j+2l]+u$;
⑧ $\hat{f}[j]:=\hat{f}[j]+\hat{f}[j+l]+\hat{f}[j+2l]$;
⑨ $\hat{f}[j+l]:=\zeta[k]\times v$;
⑩ $\hat{f}[j+2l]:=\zeta[k+1]\times w$;
⑪ end for
⑫ $k:=k+2$;
⑬ end for
⑭ end for
⑮ for $s:=0$, $s<n$, $s:=j+n/4$ do
/*进行 1 层基-2 逆向 FFT trick*/
⑯ for $j:=s$, $j<s+n/4$, $j:=j+1$ do
⑰ $t:=\hat{f}[j]$;
⑱ $\hat{f}[j]:=t+\hat{f}[j+n/4]$; /*GS 蝴蝶*/
⑲ $\hat{f}[j+n/4]:=\zeta[k]\times(t-\hat{f}[j+n/4])$;
⑳ end for
㉑ $k:=k+1$;
㉒ end for
㉓ $\zeta':=(2\zeta_1-1)^{-1}$; /*$\zeta_1$如图 1 所示*/
㉔ for $j:=0$, $j<n/2$, $j:=j+1$ do
/*进行首层 CRT 同构*/
㉕ $t:=\zeta'\times(\hat{f}[j]-\hat{f}[j+n/2])$;
㉖ $\hat{f}[j]:=(\hat{f}[j]+\hat{f}[j+n/2]-t)\times(n/2)^{-1}$;
㉗ $\hat{f}[j+n/2]:=t\times(n/4)^{-1}$;
㉘ end for

4.3 点 乘

点乘“$\circ$”由$\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)})$中的乘法构成. 对于$\hat{f}=(\hat{f}_0,\hat{f}_1,\cdots,\hat{f}_{n/2-1})$和$\hat{g}=(\hat{g}_0,\hat{g}_1,\cdots,\hat{g}_{n/2-1})$有

$$\hat{f}\circ\hat{g}=(\hat{f}_0\hat{g}_0,\hat{f}_1\hat{g}_1,\cdots,\hat{f}_{n/2-1}\hat{g}_{n/2-1}),$$

其中$\hat{f}_i,\hat{g}_i\in\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)})$, $i=0,1,\cdots,n/2-1$.

本文使用 1-迭代 Karatsuba 算法来减少点乘中的乘法数量. 具体而言，为了在$\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)})$中计算$\hat{f}_i$和$\hat{g}_i$的积，即是

$$\hat{f}_i\hat{g}_i \bmod x^2-\zeta^{\tau(i)}=(\hat{f}_{i,0}+\hat{f}_{i.1}x)(\hat{g}_{i,0}+\hat{g}_{i,1}x) \bmod x^2-\zeta^{\tau(i)}=(\hat{f}_{i,0}\hat{g}_{i,0}+\zeta^{\tau(i)}\hat{f}_{i.1}\hat{g}_{i,1})+(\hat{f}_{i,0}\hat{g}_{i,1}+\hat{f}_{i.1}\hat{g}_{i,0})x,$$

可使用 1-迭代 Karatsuba 算法首先计算$t_0=\hat{f}_{i,0}\hat{g}_{i,0}$和$t_1=\hat{f}_{i.1}\hat{g}_{i,1}$，然后计算$\zeta^{\tau(i)}t_1$，以及

$$\hat{f}_{i,0}\hat{g}_{i,1}+\hat{f}_{i.1}\hat{g}_{i,0}=(\hat{f}_{i,0}+\hat{f}_{i.1})(\hat{g}_{i,0}+\hat{g}_{i,1})-t_0-t_1.$$

原本需要 5 个乘法，现在只需要 4 个乘法. 使用 1-迭代 Karatsuba 算法计算点乘的伪代码见算法 13.

算法 13. 点乘.

输入：$\hat{f}$的系数数组$\{\hat{f}[i]\}_{i=0}^{n-1}$，$\hat{g}$的系数数组$\{\hat{g}[i]\}_{i=0}^{n-1}$，本原单位根$\{\zeta^{\tau(k)}\}_{k=0}^{n/2-1}$；

输出：点乘结果$\hat{h}$的系数数组$\{\hat{h}[i]\}_{i=0}^{n-1}$.

① for $j:=0$, $j<n/2$, $j:=j+1$ do
② $u:=\hat{f}[2j]\times\hat{g}[2j]$;
③ $v:=\hat{f}[2j+1]\times\hat{g}[2j+1]$;
④ $w:=(\hat{f}[2j]+\hat{f}[2j+1])(\hat{g}[2j]+\hat{g}[2j+1])$;
⑤ $\hat{h}[2j]:=u+\zeta^{\tau(j)}\times v$;
⑥ $\hat{h}[2j+1]:=w-u-v$;
⑦ end for

4.4 基于混合基 NTT 的多项式运算

基于混合基 NTT 计算多项式乘法$h=fg\in\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$的过程为

$$h=\mathcal{INTT}(\mathcal{NTT}(f)\circ\mathcal{NTT}(g)).$$

另外，本文同样使用该混合基 NTT 计算多项式除法$h=g/f\in\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$，即计算 LTRU 方案中的公钥$h$. 具体而言，其计算过程为

$$h=\mathcal{INTT}(\hat{g}\circ\hat{f}^{-1}),$$

其中$\hat{g}=\mathcal{NTT}(g)$, $\hat{f}=\mathcal{NTT}(f)$以及

$$\hat{f}^{-1}=(\hat{f}_0^{-1},\hat{f}_1^{-1},\cdots,\hat{f}_{n/2-1}^{-1}).$$

本文使用$\hat{f}_i^{-1}$表示$\hat{f}_i$在$\mathbb{Z}_q[x]/(x^2-\zeta^{\tau(i)})$中的逆（此时需要$\hat{f}_i^{-1}$均存在）. 实际上，$\hat{f}_i^{-1}$能够由$\hat{f}_i=\hat{f}_{i,0}+\hat{f}_{i,1}x$显式求得

$$\hat{f}_i^{-1}=\frac{1}{\hat{f}_{i,0}^2-\zeta^{\tau(i)}\hat{f}_{i,1}^2}(\hat{f}_{i,0}-\hat{f}_{i,1}x).$$

同时可知，当且仅当$\hat{f}_{i,0}^2-\zeta^{\tau(i)}\hat{f}_{i,1}^2\neq 0$时$\hat{f}_i^{-1}$存在；另一方面，如果存在某个$i$使得$\hat{f}_{i,0}^2-\zeta^{\tau(i)}\hat{f}_{i,1}^2=0$，那么$f$的逆不存在. 此时公钥$h$不存在，需要重新采样$f$直至$f$的逆存在. 给定$f$，计算$\hat{f}^{-1}$的伪代码见算法 14.

算法 14. 计算低次数多项式的逆.

输入：$\hat{f}$的系数数组$\{\hat{f}[i]\}_{i=0}^{n-1}$，本原单位根$\{\zeta^{\tau(k)}\}_{k=0}^{n/2-1}$；

输出：$\hat{f}^{-1}$的系数数组$\{\hat{f}^{-1}[i]\}_{i=0}^{n-1}$.

① for $j:=0;\ j<n/2;\ j:=j+1$ do

② $\quad t:=\hat{f}[2j]\times\hat{f}[2j]$;

③ $\quad t:=t-\zeta^{\tau(j)}\times\hat{f}[2j+1]\times\hat{f}[2j+1]$;

④ $\quad t:=t^{-1}$;

⑤ $\quad \hat{f}^{-1}[2j]:=\hat{f}[2j]\times t$;

⑥ $\quad \hat{f}^{-1}[2j+1]:=-\hat{f}[2j+1]\times t$;

⑦ end for

5 实现细节

本文给出了 LTRU 的便携 C 实现和优化 AVX2 实现. 下面将简要介绍 LTRU 的一些实现细节. 值得注意的是，LTRU 的所有实现均采用了常数时间实现技巧，以抵抗一些潜在的侧信道攻击，特别是计时攻击[31].

5.1 基本构件

本文使用的模数$q=2\,917$是低于 16 b 的素数，所以$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$中的多项式能够使用长度为$n$、类型为 16 b 有符号整型的数组来存储.

1）哈希函数

本文中使用的哈希函数均使用 SHA3 系列函数来实例化. 具体而言，使用 SHA3-256 实例化哈希函数$\mathcal{F}$，使用 SHA3-512 实例化哈希函数$\mathcal{H}$. 其中$\mathcal{H}$以明文消息m作为输入，得到 64 B 的输出，其中前 32 B 作为导出的共享密钥，后 32 B 作为种子并使用 SHAKE128 来生成加密算法所需的随机数.

2）秘密多项式

本文使用的秘密多项式如f', g, r都是根据分布$\bar{B}_2$采样得到. 每采样一个系数需要 4 个随机比特，所以采样一个多项式需要$4n$个比特，即$n/2$字节.

3）约减算法

在计算多项式系数的加法和乘法时，计算结果存在超出 16 b 有符号整型的有效表示范围的可能性. 此时需要使用约减算法将计算结果约减到$[0,q)$. 本文使用 Barrett 约减算法[57] 和 Montgomery 约减算法[58]. 其中 Barrett 约减算法主要用于系数加法后的约减；而 Montgomery 约减算法主要用于系数和系数相乘，或者系数和本原单位根相乘时的约减.

4）消息的存储形式

注意到在 LTRU 加密算法中，多项式e是系数范围取自$\{0,1\}$的多项式，所以可以将e视为大小为n的比特串. 实际上，本文使用$n/8$个字节（即n位）来存储并表示e，这样能够减少将e在多项式和比特串之间转换的过程，提高效率.

5）公私钥和密文

公钥和私钥均以 NTT 域中多项式的形式存储和传输. 具体而言，计算得到$\hat{h}$和$\hat{f}$之后，以$\hat{h}$作为公钥，且以$\hat{f}$作为私钥. 这样能在密钥生成算法中省去对$\hat{h}$的逆向 NTT 变换，以及在加密算法中对h的正向 NTT 变换. 同时，在解密算法中亦省去对私钥f的正向 NTT 变换. 所以，公钥$\hat{h}$和私钥$\hat{f}$的每个系数均为 12 b，故共需$12n$位，即是$3n/2$个字节的存储空间. 另外，由于解封装算法中需要重加密步骤，故封装方案的私钥还需要包含公钥$\hat{h}$. 密文的第 1 项c以正常域中多项式的形式存储和传输，需要$nd/8$个字节的存储空间，这是因为压缩密文操作需要在正常域中进行；第 2 项u是比特串，需要$n/8$个字节的存储空间. 在实际传输过程中，密文的第 1 项和第 2 项拼接之后总计$nd/8+n/8$个字节.

5.2 C 实现

本节提供 LTRU 的 C 实现的细节. 本文使用第 4 节的高效的混合基 NTT 来计算 LTRU-648 的多项式乘法和除法（即计算公钥h）. 测试设备为：硬件配置为 2.3 GHz 的 Intel® Core™ i7-10510U CPU 和 16 GB 内存的笔记本电脑，且关闭 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，且 gcc 版本为 9.4.0.编译参数为：*-Wall-march=native-mtune=native-O3-fomit-framepointer-Wno-unknown-pragmas*.

本文 LTRU 的 C 实现只涉及到整型算术操作，特别是 16 b 有符号整型算术. 在 NTT 的实现中，由于 C 语言的“%”运算符不是常数时间实现，其执行时间和输入数据长度密切相关，存在泄露秘密数据的可能性，故本文转向使用常数时间实现的 Barrett 约减算法和 Montgomery 约减算法. 对于正向 NTT 变换，本文使用延迟规约策略[59] 来减少约减的次数. 具体而言，对于乘法中使用的 Montgomery 约减，其输出值范围为$[-q,q]$. 本文使用的模数q为 12 b，所以在 6 层 FFT trick 之后，正向 NTT 变换输出的多项式范围

在$[-7q,7q]$之间，这显然在 16 b 有符号整型的有效表示范围之内. 故中间计算过程无需进行额外的 Barrett 约减，而只需要在正向 NTT 变换结束时进行 1 次 Barrett 约减.

5.3 AVX2 实现

本节提供 LTRU 的 AVX2 实现细节.AVX2 实现主要优化的运算包括：多项式加法/乘法/除法、模约减算法等. 但是，SHA3 系列哈希函数并没采用 AVX2 优化实现，而是继续采用 C 语言的实现. 这是因为 SHA3 系列哈希函数难以向量化实现，而目前最高效的实现是基于 C 语言的[52]. 另外，NTT 运算非常耗时，所以 NTT 运算是 AVX2 优化的主要目标. 下面主要介绍本文提出的混合基 NTT 的 AVX2 实现细节.

AVX2 指令集的载入/存储指令一次性载入/存储的数据长度为 256 b. 换句话说，指令能够一次性载入/存储 16 个多项式系数，因为每个系数使用 16 b 有符号整型表示. 但是，每个多项式共有 648 个系数，其中 648 并非 16 的整数倍. 为了能够使用 AVX2 指令集的载入/存储指令，将 648 个系数对相邻的 16 个系数进行划分，得到 40 个划分单位以及剩下的 8 个系数，然后对每个划分单位进行数据对齐. 这种方法虽然便于使用载入/存储指令，但不便于在 NTT 运算中计算基-2 FFT trick 步骤和基-3 FFT trick 步骤.

为了方便使用 AVX2 指令集的载入/存储指令，同时方便计算基-2 FFT trick 步骤和基-3 FFT trick 步骤，本文采用 2 个系数填充过程：

1) 将 648 个系数对相邻的 12 个系数进行划分，得到 54 个划分单位；

2)把每一个划分单位里面的 12 个系数拓展到 16 个系数，即保留原 12 个系数位置不变，在它们末尾的位置填充 4 个 0.

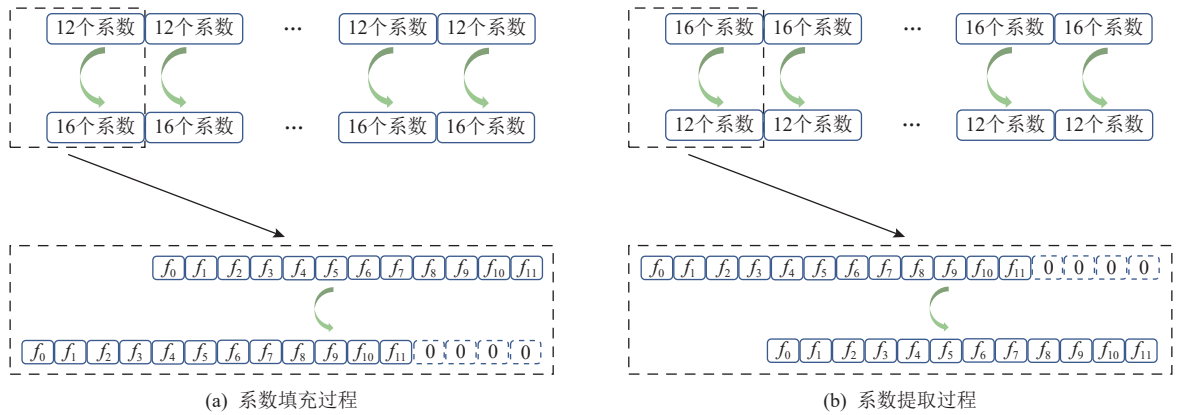
通过这 2 个系数填充过程，得到 864 个系数，以及新的 54 个划分单位，每个划分单位包含 16 个系数. 此时每个划分单位能够一次性被载入到寄存器之中. 在完成寄存器一系列计算、存储回内存位置后，多项式系数提取的过程为：

1)将 864 个系数对相邻的 16 个系数进行划分，得到 54 个划分单位；

2) 在每一个划分单位中提取前面的 12 个系数，舍弃末尾的 4 个 0.

通过这 2 个系数提取过程，得到原始长度的多项式数组. 系数填充过程和系数提取过程如图 2 所示.

这种填充过程应用在多项式的 NTT 运算，如正向变换、逆向变换、点乘和求逆之中. 另外，这种填充过程使得 NTT 运算以 12 倍的并行度进行计算，区别于通常的 16 倍的并行度计算. 这是因为此时每个寄存器只能一次性处理原始多项式的 12 个系数，而非原始多项式的 16 个系数. 由于 AVX2 指令集的载入/存储指令耗时较多，所以应该减少内存访问的次数. 常用的方法是合并处理 NTT 运算中的若干层. 本文合并正向变换的第 2 层和第 3 层的计算. 在此期间，仅需载入一次数据，无需额外的访问内存和数据载入/存储便能够完成第 2 层和第 3 层的计算，这能够降低因载入/存储指令带来的 CPU 周期数的消耗. 本文通过 *vmovdqa* 指令完成数据的载入/存储. 在分别载入多项式系数和本原单位根之后，通过 *vpmulhw* 指令和 *vpmullw* 指令分别计算它们乘积的高位和低位，这样能够高效地计算它们的 Montgomery 约减的值. 本文的 AVX2 实现将一些常用的代码段通过 *macro* 封装起来，这使得整体代码简洁和可读性强.



(a) 系数填充过程 (b) 系数提取过程

Fig. 2 The padding and extraction of coefficients

图 2 系数的填充与提取

5.4 常数时间实现

计时攻击[31]是一种常见的侧信道攻击，它指的是敌手可以根据密码方案在软硬件环境中的执行时间等信息来推断所执行的运算操作和内容访问情况，继而攻击者可以根据所获取的侧信道信息来推算出该操作所涉及的一些秘密信息，如私钥或者一些秘密值. 目前来说，密码方案的实现几乎都需要考虑计时攻击，并应该采用常数时间实现策略. 因此，本文的 LTRU 实现过程采用了常数时间实现策略. 最重要的是，本文并没使用非常数时间的指令来操作秘密数据，以及没使用依赖于秘密数据的判断分支语句，也没访问依赖于秘密数据寻址的内存地址. 计算 NTT 过程中取模使用到的 Barrett 约减算法和 Montgomery 约减算法均是常数时间实现[59]，且算法本身对任意合适的模数均成立. 至于采样秘密多项式 f',g,r，本文均以常数时间的方式来采样. 每次固定使用 4 b 来生成 1 个多项式系数. 显然，生成秘密多项式的时间也是恒定的.

6 比较和分析

本节将比较混合基 NTT 算法与未使用 1-迭代 Karatsuba 算法和单位根的性质优化乘法数量的 NTT 算法，以及比较本文 LTRU 和其他 NTRU 类型密钥封装方案如 NTRU-HRSS[20], SNTRU Prime[21], NTTRU[26], NTRU-A[27], CTRU[28], BAT[29]，还有基于模格的密钥封装方案如 NIST 标准化方案 Kyber[22].

表 2 给出了混合基 NTT 算法与未使用 1-迭代 Karatsuba 算法和单位根的性质优化乘法数量的 NTT 算法在理论乘法复杂度分析和具体实现的 CPU 周期数的比较.

表 3 和表 4 展示了本文 LTRU 和其他方案对比的详细情况. 为方便比较，在此选取与这些方案安全强度接近的参数集，但 CTRU 和 Kyber 并没有接近 128 b 量子安全强度的参数集，所以在此选取它们的推荐参数集. SNTRU Prime 选取 SNTRU Prime-761 参数集，NTRU-A 选取 NTRU - $A^{648}_{2\,917}$ 参数集，CTRU 选取 CTRU-768 参数集，BAT 选取 BAT-512 参数集，Kyber 选取 Kyber-768 参数集. 其中 NTRU-HRSS, SNTRU Prime-761, Kyber-768 的数据和开源代码来自它们的 NIST 第 3 轮材料. NTTRU 的数据和开源代码来自文献 [26]. NTRU - $A^{648}_{2\,917}$ 的数据来自文献 [27], CTRU-768 的数据来自文献 [28], BAT-512 的数据来自文献 [29].

6.1 NTT 算法的比较

值得注意的是，本文的混合基 NTT 和未优化的混合基 NTT 在多项式环 $\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$ 首层 CRT 同构具有相同的乘法复杂度：正向变换时需要 $n/2$ 个乘法，逆向变换时需要 $3n/2$ 个乘法. 为方便比较乘法复杂度，在此记基-2 FFT trick 和基-3 FFT trick 的层数分别为 l_2 和 l_3. 在本文的混合基 NTT 中，每层基-2 FFT trick 的正向变换和逆向变换均需要 $n/2$ 个乘法. 每层基-3 FFT trick 的正向变换和逆向变换均需要 n 个乘法. 然而，未优化的 NTT 每层基-3 FFT trick 的正向变换和逆向变换均需要 $2n$ 个乘法. 因为 2 种 NTT 算法的求逆运算相同，于是求逆运算具有相同的乘法复杂度，故可以省去对它们的求逆运算的比较.

从表 2 可以看出，在比较多项式乘法复杂度时，本文 NTT 比未优化的 NTT 在正向变换、点乘和逆向

Table 2 Comparison Between Mixed-Radix NTT and Unoptimized NTT

表 2 混合基 NTT 和未优化的 NTT 的比较

比较的项目	类别	混合基 NTT（本文）	未优化的 NTT
乘法复杂度理论分析	多项式环	$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$	$\mathbb{Z}_q[x]/(x^n-x^{n/2}+1)$
	正向变换	$\frac{1}{2}nl_2+nl_3+\frac{1}{2}n$	$\frac{1}{2}nl_2+2nl_3+\frac{1}{2}n$
	点乘	$2n$	$\frac{5}{2}n$
	逆向变换	$\frac{1}{2}nl_2+nl_3+\frac{3}{2}n$	$\frac{1}{2}nl_2+2nl_3+\frac{3}{2}n$
	求逆矩阵	2×2 方阵	2×2 方阵
	总计	nl_2+2nl_3+4n	$nl_2+4nl_3+\frac{9}{2}n$
具体实现耗时/kcycle（选取 $n=648,q=2\,917$）	正向变换	21 588	28 756
	点乘	6 874	8 657
	逆向变换	25 119	34 326
	总计	53 581	71 739

Table 3 Comparison Between LTRU and Other Schemes

表 3 LTRU 和其他方案的对比

方案	困难性假设	底层 PKE	$\|pk\|$/B	$\|ct\|$/B	*B.W.* /B	Sec-(C, Q)/b	δ
LTRU（本文）	NTRU-OW	IND-CPA RPKE	972	842	1814	(142, 129)	2^{-154}
NTRU-HRSS	NTRU-OW	OW-CPA DPKE	1 138	1 138	2 276	(136, 124)	$2^{-\infty}$
SNTRU Prime-761	NTRU-OW	OW-CPA DPKE	1 158	1 039	2 197	(153, 137)	$2^{-\infty}$
NTTRU	NTRU-OW	OW-CPA RPKE	1 248	1 248	2 496	(153, 140)	2^{-1217}
NTRU - $A_{2\,917}^{648}$	NTRU+RLWE	OW-CPA RPKE	972	972	1 944	(152, 137)	2^{-175}
CTRU-768	NTRU+RLWE	IND-CPA RPKE	1 152	960	2 112	(181, 164)	2^{-184}
BAT-512	NTRU+RLWR	IND-CPA RPKE	521	473	994	(140, 124)	2^{-146}
Kyber-768	MLWE	IND-CPA RPKE	1 184	1 088	2 272	(183, 166)	2^{-164}

Table 4 Comparison of Implementation Efficiency of Schemes

表 4 方案的实现效率对比 kcycle

方案	C 实现			AVX2 实现		
	KeyGen	*Encaps*	*Decaps*	*KeyGen*	*Encaps*	*Decaps*
LTRU（本文）	88.9	61.5	108.8	23.1	24.1	33.3
NTRU-HRSS	130.0×10^3	3.6×10^3	9.8×10^3	254.0	24.9	59.2
SNTRU Prime-761	16.9×10^3	9.1×10^3	24.5×10^3	715.1	41.7	49.0
CTRU-768	117.6	63.3	134.7	10.6	11.8	35.6
BAT-512				29.4×10^3	11.1	59.7
Kyber-768	165.9	188.6	210.2	41.5	56.8	44.3
LTRU-SHA2（本文）	77.3	52.4	93.8	16.6	16.2	23.8
NTTRU	160.5	101.1	141.6	6.4	6.1	7.9
NTRU - $A_{2\,917}^{648}$				6.2	5.6	7.3

变换方面分别减少了nl_3, $n/2$, nl_3个乘法，总计减少$2nl_3+n/2$个乘法. 具体而言，选取参数$n=648$和$q=2\,917$时，此时层数分别为$l_2=1, l_3=4$. 测试数据均由 C 语言实现，并采用 O3 优化指令. 从具体实现的 CPU 周期数来看，相比未优化的 NTT，本文 NTT 在正向变换、点乘和逆向变换的 CPU 周期数分别降低 24.9%, 20.6%, 26.8%，总共降低 25.3%.

6.2 方案的性能比较

在表 3 中，困难性假设指的是该方案所基于的困难性问题. 底层 PKE 指的是该密钥封装方案所蕴含的底层公钥加密方案的属性，其中 IND-CPA 指不可区分意义下的选择明文安全性，OW-CPA 指单向意义下的选择明文安全性，DPKE 指确定性的公钥加密方案，RPKE 指随机性的公钥加密方案. $|pk|$, $|ct|$, *B.W.* 分别指它们的公钥尺寸、密文尺寸和带宽（公钥尺寸+密文尺寸），单位均是 B. Sec-(C, Q)是方案的安全强度，C 表示经典安全强度，Q 表示量子安全强度. δ 指方案的错误率.

6.3 方案的效率比较

在表 4 中，*KeyGen*, *Encaps*, *Decaps* 对应的列分别给出这些密钥封装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 次的平均 CPU 周期，单位是 kcycle.

需要说明的是，NTRU-HRSS, SNTRU Prime-761, NTTRU, Kyber-768 的 C 实现数据均是将它们的开源 C 代码在同一平台上运行得到的，旨在为它们的 C 实现效率提供直观的对比. 至于 NTRU-HRSS, SNTRU Prime-761, Kyber-768 的 AVX2 实现的测试数据则是取自 SUPERCUP（代号为 supercop-20220506, 测试设备为 3.0GHz Intel Xeon E3-1 220 v6）[60]; SUPERCUP 能够提供密码方案的 AVX2 实现最新最快的测试数据. NTTRU 的 AVX2 测试数据取自文献 [26]. NTRU-$A_{2\,917}^{648}$的 AVX2 测试数据则取自文献 [27], BAT-512 的 AVX2 测试数据则取自文献 [29], CTRU-768 的 C 实现和 AVX2 实现的测试数据均是取自文献 [28]. 需要注意的是，文献 [27−28] 没有提供开源 C 代码和 AVX2

代码，文献 [29] 只提供 AVX2 代码. 且文献 [27, 29] 只提供 AVX2 实现的测试数据，故为公平比较，表 4 只展示NTRU-$A_{2\,917}^{648}$和 BAT-512 的 AVX2 实现的测试数据，而不再展示它们的 C 实现的测试数据.

6.4 分析和结论

经表 3 和表 4 的数据分析，与其他基于 NTRU 格的密钥封装方案 NTRU-HRSS[20]，SNTRU Prime[21]，NTTRU[26]，NTRU-A[27]，CTRU[28]，BAT[29]，还有基于模格的密钥封装方案 Kyber[22] 相比，本文的 LTRU 具有 4 点优势：

1）小的公钥尺寸、密文尺寸和带宽

与 NTRU-HRSS，SNTRU Prime-761，NTTRU，NTRU - $A_{2\,917}^{648}$，CTRU-768，Kyber-768 对比，LTRU 具有更小的密文尺寸和更小的通信带宽. 具体地，与 NTRU-HRSS 对比，LTRU 的公钥尺寸降低 14.6%，密文尺寸降低 26.0%，带宽降低 20.3%. 与 SNTRU Prime-761 相比，LTRU 的公钥尺寸降低 16.1%，密文尺寸降低 19.0%，带宽降低 17.4%. 与 Kyber-768 相比，LTRU 的公钥尺寸降低 17.9%，密文尺寸降低 22.6%，带宽降低 20.2%.LTRU 和NTRU - $A_{2\,917}^{648}$具有相同长度的公钥尺寸，但是 LTRU 的密文尺寸更小，降低了 13.4%. 由于 BAT-512 使用复杂的纠错码使得它能使用更小的模数，从而它的带宽比 LTRU 的更有优势，但复杂的纠错码导致它的密钥生成比 LTRU 的密钥生成慢了 3 个数量级.

2）均衡的安全强度、错误率和带宽

由于 LTRU 的目标安全强度为 128 b，所以它的错误率2^{-154}可视为可忽略. NTRU-HRSS 和 BAT-512 的量子安全强度只有 124 b，并没达到 128 b.而 Kyber-768 具有 166 b 量子安全强度. CTRU-768 具有 164 b 量子安全强度，与 128 b 的间距过大，造成过剩的安全性.SNTRU Prime-761，NTTRU，NTRU - $A_{2\,917}^{648}$虽然达到了 128 b 安全强度，但其带宽都比 LTRU 的带宽大. 注意到 NTRU-HRSS 和 SNTRU Prime-761 均为零错误率，但是它们需要使用更大的模数q，这无疑增加了通信带宽. 所以，与其他方案对比，LTRU 达到了 128 b 安全强度，且具有与安全强度匹配的、可忽略的错误率，同时能够节省通信带宽.

3）强的底层 PKE 方案安全性

由于 Kyber-768 是基于 MLWE 困难性假设的，而其他方案都是基于 NTRU 相关的困难性假设，所以为了公平对比，在此只比较LTRU，NTRU-HRSS，SNTRU Prime-761，NTTRU，NTRU - $A_{2\,917}^{648}$，CTRU-768，BAT-512 的底层 PKE 方案安全性. 从表 3 可以看到，LTRU，CTRU-768，BAT-512 的底层 PKE 方案能够达到 IND-CPA 安全性，而 NTRU-HRSS，SNTRU Prime-761，NTTRU，NTRU - $A_{2\,917}^{648}$的底层 PKE 方案均是 OW-CPA 安全的. 其中 CTRU-768 和 BAT-512 均基于 2 个困难性假设，唯有 LTRU 仅基于 NTRU-OW 假设而达到 IND-CPA 安全性. 不容忽视的是，对底层 PKE 方案来说，IND-CPA 安全性比 OW-CPA 安全性更强.

4）高效的实现

从表 4 可知，LTRU 具有出色的实现效率. 具体而言，与 NTRU-HRSS 的 C 实现相比，LTRU 在密钥生成、封装和解封装等算法上分别快了 1 462.3 倍、58.5 倍和 90.0 倍；与 NTRU-HRSS 的 AVX2 实现相比，LTRU 在密钥生成算法和解封装算法上分别快了 10.9 倍和 1.7 倍，其封装算法的速度与 NTRU-HRSS 的相当. 另外，与 SNTU Prime-761 的 AVX2 实现相比，LTRU 在密钥生成、封装和解封装等算法上分别快了 30.9 倍、1.7 倍和 1.4 倍. 主要原因是 LTRU 使用了本文提出的高效的混合基 NTT 计算多项式运算，而 NTRU-HRSS 和 SNTU Prime-761 使用的多项式环不是 NTT 友好的，它们只能使用效率更低的算法计算多项式运算. 与 CTRU-768 相比，LTRU 的 C 实现在密钥生成、封装和解封装等算法都更有优势，AVX2 实现在解封装算法的速度与 CTRU-768 的相当. 与 BAT-512 的 AVX2 实现相比，LTRU 在密钥生成算法上快了 3 个数量级，在解封装算法上快了 1.7 倍.

我们注意到，NTTRU 和NTRU - $A_{2\,917}^{648}$使用 SHA2 系列函数来实例化哈希函数，且使用 AES 算法作为伪随机函数来生成所需的随机数. 相比于本文的 LTRU 使用的 SHA3 系列函数，SHA2 系列函数和 AES 算法更易于向量化实现（如 AVX2 实现）. 为方便与 NTTRU 和NTRU-$A_{2\,917}^{648}$比较，表 4 给出了 LTRU-SHA2 变体的 AVX2 实现效率的测试数据.LTRU-SHA2 变体使用了与 NTTRU 和NTRU-$A_{2\,917}^{648}$相同的 SHA2 哈希函数以及 AES 算法. 尽管 LTRU（LTRU-SHA2 的变体）的 AVX2 实现效率稍逊于 NTTRU 和 NTRU-$A_{2\,917}^{648}$，事实上，从表 3 和表 4 中 LTRU（LTRU-SHA2 的变体）的安全性、带宽和实现效率等数据来看，它们已经能够适用于现实多数场景. 另外，与 NTTRU 相比，LTRU 在生成公钥h时只需要对底层的线性多项式求逆，而 NTTRU 需要计算底层的 2 次多项式的逆，这明显更复杂. 同时 LTRU 的模数$q=2\,917$的 NTT 比 NTTRU 的模数$q=7\,681$的 NTT 在计算资源开销和效率上都更有优势.

与 Kyber-768 相比，LTRU 的实现效率全面占优. 具体而言，与 Kyber-768 的 AVX2 实现相比，LTRU 在密钥生成、封装和解封装等算法上分别快了 1.7 倍、2.3 倍和 1.3 倍. 这得益于 LTRU 在密钥生成算法中使用高效混合基 NTT 以及在加密和解密算法中只需要计算 1 个多项式乘法. 然而，Kyber 在密钥生成算法和加密算法中均需要使用拒绝采样生成矩阵且需要计算多项式矩阵-向量乘法，其中解封装算法中还有重加密和重新计算多项式矩阵-向量乘法的过程，这些操作明显更加耗时. 与之对比，LTRU 方案具有运算简单、实现高效等特点.

7 总　　结

基于 NTRU 格的密码系统具有结构简洁、强安全保障、不受专利影响等优势，有利于设计实用化高效的后量子密码方案. 本文提出了一个仅基于 NTRU 单向困难性假设构造的、不使用纠错码也能够压缩密文的密钥封装方案 LTRU. LTRU 包含一个 IND-CPA 安全的公钥加密方案 LTRU.PKE 以及一个 IND-CCA 安全的密钥封装方案 LTRU.KEM.本文还提供了针对 128 b 安全强度的参数集，及其对应的 C 实现和 AVX2 实现. 针对 LTRU 所使用的环 $\mathbb{Z}_{2\,917}[x]/(x^{648}-x^{324}+1)$ 上的多项式乘法和除法，本文提出了一种高效的混合基 NTT 算法，该算法结合单位根的性质和 1-迭代 Karatsuba 算法来减少乘法的数量，以提高计算效率. 本文实验结果表明，与 NIST 标准化方案 Kyber-768 相比，LTRU 的公钥尺寸降低 17.9%，密文尺寸降低 22.6%，带宽降低 20.2%；与 NIST 第 3 轮决赛方案 NTRU-HRSS 相比，LTRU 的经典安全强度和量子安全强度分别增强 6 b 和 5 b，以及公钥尺寸降低 14.6%，密文尺寸降低 26.0%，带宽降低 20.3%，并且 LTRU 的实现效率优于 NTRU-HRSS.

未来工作主要包括 2 点：1）对本文提出的 LTRU 在多平台上实现，包括 ARM Cortex-M4 实现和 FPGA 实现等；2）提出更多的 LTRU 参数集，使其满足更多的应用场景，如资源受限的 IoT 设备和智能卡等.

作者贡献声明：梁志闯负责方案的设计和论文撰写；郑婕妤负责方案的实现；赵运磊负责论文修改.

参　考　文　献

[1] Shor P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer[J]. SIAM Review, 1999, 41(2): 303−332

[2] NIST. PQC standardization process: Announcing four candidates to be standardized, plus fourth round candidates [EB/OL]. (2022-07-05) [2023-04-10].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 [EB/OL](2020-01-02)[2023-04-10]. https://www.cacrnet.org.cn/site/content/854.html (in Chinese).
(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[EB/OL].(2020-01-02)[2023-04-10].https://www.cacrnet.org.cn/site/content/854.html)

[4] NIST. Post-quantum cryptography round 1 submissions [EB/OL]. (2017-01-03)[2023-04-10] https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions

[5] NIST. Post-quantum cryptography round 2 submissions [EB/OL]. (2019-01-30)[2023-04-10].https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions

[6] NIST. Post-quantum cryptography round 3 submissions [EB/OL]. (2020-07-23)[2023-04-10].https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions

[7] Regev O. On lattices, learning with errors, random linear codes, and cryptography[J]. Journal of the ACM, 2009, 56(6): 1−40

[8] Banerjee A, Peikert C, Rosen A. Pseudorandom functions and lattices [C] // Proc of the 31st Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2012: 719–737

[9] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings[C] // Proc of the 29th Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2010: 1–23

[10] Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices[J]. Designs, Codes and Cryptography, 2015, 75(3): 565−599

[11] Hoffstein J, Pipher J, Silverman J H. NTRU: A ring-based public key cryptosystem[C] //Proc of the 3rd Int Algorithmic Number Theory Symp. Berlin: Springer, 1998: 267–288

[12] Howgrave-Graham N, Silverman J H, Whyte W. Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3[C] //Proc of the Cryptographers' Track at the RSA Conf. Berlin: Springer, 2005: 118–135

[13] Howgrave-Graham N, Silverman J H, Singer A, et al. NAEP: Provable security in the presence of decryption failures [DB/OL]. IACR Cryptology ePrint Archive, 2003 [2023-04-10].https://eprint.iacr.org/2003/172

[14] Hoffstein J, Pipher J, Silverman J H. NTRU: A new high speed public key cryptosystem[C] //Proc of the Rump Session of Crypto96. Berlin: Springer, 1996: 1–11

[15] Coppersmith D, Shamir A. Lattice attacks on NTRU[C] //Proc of Int Conf on the Theory and Application of Cryptographic Techniques.

Berlin: Springer, 1997: 52–61

[16] Garg S, Gentry C, Halevi S. Candidate multilinear maps from ideal lattices[C] //Proc of the 32nd Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2013: 1–17

[17] Langlois A, Stehlé D, Steinfeld R. GGHLite: More efficient multilinear maps from ideal lattices[C] //Proc of the 33rd Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2014: 239–256

[18] Ducas L, Lyubashevsky V, Prest T. Efficient identity-based encryption over NTRU lattices[C] // Proc of the 20th Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2014: 22–41

[19] Fouque P A, Hoffstein J, Kirchner P, et al. Falcon: Fast-Fourier lattice-based compact signatures over NTRU[EB/OL]. NIST PQC Round 3 Submission, 2020 [2023-04-10].https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[20] Chen Cong, Danba O, Hoffstein J. et al. NTRU submission[EB/OL]. NIST PQC round 3 submission, 2020 [2023-04-10]. https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[21] Bernstein D J, Brumley B, Chen M S, et al. NTRU Prime: Round 3 [EB/OL]. NIST PQC Round 3 Submission, 2020 [2023-04-10].https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[22] Avanzi R, Bos J, Ducas L, et al. CRYSTALS-Kyber: Algorithm specifications and supporting documentation (version 3.01)[EB/OL]. NIST PQC Round 3 Submission, 2020 [2023-04-10].https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[23] NIST. Status report on the third round of the NIST post-quantum cryptography standardization process [EB/OL]. (2022-07-30) [2023-04-10].https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf

[24] Business Wire. Business wire security innovation's NTRUEncrypt adopted as X9 standard for data protection [EB/OL]. (2011-04-11) [2023-04-10].https://www.businesswire.com/news/home/20110411005309/en/Security-Innovations-NTRUEncrypt-Adopted-X9-Standard-Data

[25] OpenSSH. OpenSSH release notes [EB/OL]. (2022-10-04) [2023-04-10].https://www.openssh.com/releasenotes.html

[26] Lyubashevsky V, Seiler G. NTTRU: Truly fast NTRU using NTT[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019, 2019(3): 180−201

[27] Duman J, Hövelmanns K, Kiltz E, et al. A thorough treatment of highly-efficient NTRU instantiations [C] //Proc of the 26th IACR Int Conf on Practice and Theory of Public-Key Cryptography. Berlin: Springer, 2023: 65–94

[28] Liang Zhichuang, Fang Boyue, Zheng Jieyu, et al. Compact and efficient KEMs over NTRU lattices [DB/OL]. IACR Cryptology ePrint Archive, 2022 [2023-04-10].https://eprint.iacr.org/2022/579

[29] Fouque P A, Kirchner P, Pornin T, et al. BAT: Small and fast KEM over NTRU lattices[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022, 2022(2): 240−265

[30] D'Anvers J P, Tiepelt M, Vercauteren F, et al. Timing attacks on error correcting codes in post-quantum schemes[C] //Proc of ACM Workshop on Theory of Implementation Security Workshop. New York: ACM, 2019: 2–9

[31] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems[C] // Proc of the 16th Annual Int Cryptology Conf. Berlin: Springer, 1996: 104–113

[32] Stehlé D, Steinfeld R. Making NTRU as secure as worst-case problems over ideal lattices[C] //Proc of the 30th Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2011: 27–47

[33] Wang Yang, Wang Mingqiang. Provably secure NTRUEncrypt over any cyclotomic field[C] //Proc of the 25th Int Conf on Selected Areas in Cryptography. Berlin: Springer, 2018: 391–417

[34] Jarvis K, Nevins M. ETRU: NTRU over the Eisenstein integers[J]. Designs, Codes and Cryptography, 2015, 74(1): 219−242

[35] Hülsing A, Rijneveld J, Schanck J, et al. High-speed key encapsulation from NTRU[C] //Proc of the 17th Int Conf on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2017: 232–252

[36] Bernstein D J, Chuengsatiansup C, Lange T, et al. NTRU prime: Reducing attack surface at low cost[C] //Proc of the 24th Int Conf on Selected Areas in Cryptography. Berlin: Springer, 2017: 235–260

[37] Bos J, Ducas L, Kiltz E, et al. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM[C] //Proc of 2018 IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2018: 353–367

[38] Bellare M, Rogaway P. Random oracles are practical: A paradigm for designing efficient protocols[C] //Proc of the 1st ACM Conf on Computer and Communications Security. New York: ACM, 1993: 62–73

[39] Boneh D, Dagdelen Ö, Fischlin M, et al. Random oracles in a quantum world[C] //Proc of the 17th Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2011: 41–69

[40] Bernstein D J. Multidigit multiplication for mathematicians [EB/OL]. (2001-09-12) [2023-04-10]. http://cr.yp.to/papers.html#m3

[41] Cooley J W, Tukey J W. An algorithm for the machine calculation of complex Fourier series[J]. Mathematics of Computation, 1965, 19(90): 297−301

[42] Gentleman W M, Sande G. Fast Fourier transforms: For fun and profit[C] // Proc of the AFIPS'66 Fall Joint Computer Conf. New York: ACM, 1966: 563–578

[43] Weimerskirch A, Paar C. Generalizations of the Karatsuba algorithm for efficient implementations [DB/OL]. IACR Cryptology ePrint Archive, 2006 [2023-04-10].https://eprint.iacr.org/2006/224

[44] Fujisaki E, Okamoto T. Secure integration of asymmetric and symmetric encryption schemes[J]. Journal of Cryptology, 2013, 26(1): 80−101

[45] Hofheinz D, Hövelmanns K, Kiltz E. A modular analysis of the Fujisaki-Okamoto transformation[C] //Proc of the 15th Theory of Cryptography Conf. Berlin: Springer, 2017: 341–371

[46] Shoup V. Sequences of games: A tool for taming complexity in security proofs [DB/OL]. IACR Cryptology ePrint Archive, 2004 [2023-04-10].https://eprint.iacr.org/2004/332

[47] Don J, Fehr S, Majenz C, et al. Online-extractability in the quantum random-oracle model [C] //Proc of the 41st Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2022: 677–706

[48] Kannan R. Minkowski's convex body theorem and integer

programming[J]. Mathematics of Operations Research, 1987, 12(3): 415−440

[49] Bai Shi, Galbraith S D. Lattice decoding attacks on binary LWE[C] //Proc of the 19th Australasian Conf on Information Security and Privacy. Berlin: Springer, 2014: 322–337

[50] Schnorr C P, Euchner M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems[J]. Mathematical Programming, 1994, 66(1): 181−199

[51] Chen Yuanmi, Nguyen P Q. BKZ 2.0: Better lattice security estimates[C] //Proc of the 17th Int Conf on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2011: 1–20

[52] Alkim E, Ducas L, Pöppelmann T, et al. Post-quantum key exchange—A new hope[C] //Proc of the 25th USENIX Security Symp. Berkeley: USENIX Association, 2016: 327–343

[53] Albrecht M R, Curtis B R, Deo A, et al. Estimate all the {LWE, NTRU} schemes![C] //Proc of the 11th Int Conf on Security and Cryptography for Networks. Berlin: Springer, 2018: 351–367

[54] Albrecht M, Bai Shi, Ducas L. A subfield lattice attack on overstretched NTRU assumptions[C] //Proc of the 36th Annual Int Cryptology Conf. Berlin: Springer, 2016: 153–178

[55] Howgrave-Graham N. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU[C] // Proc of the 27th Annual Int Cryptology Conf. Berlin: Springer, 2007: 150–169

[56] Hassan C A, Yayla O. Radix-3 NTT-based polynomial multiplication for lattice-based cryptography [DB/OL]. IACR Cryptology ePrint Archive, 2022 [2023-04-10].https://eprint.iacr.org/2022/726

[57] Barrett P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor[C] //Proc of Conf on the Theory and Application of Cryptographic Techniques. Berlin: Springer, 1986: 311–323

[58] Montgomery P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170): 519−521

[59] Seiler G. Faster AVX2 optimized NTT multiplication for ring-LWE lattice cryptography [DB/OL]. IACR Cryptology ePrint Archive, 2018 [2023-04-10].https://eprint.iacr.org/2018/039

[60] Bernstein D J, Lange T. eBACS: ECRYPT benchmarking of cryptographic systems [EB/OL]. (2022-06-19) [2023-04-10].https://bench.cr.yp.to

Liang Zhichuang, born in 1997. PhD candidate. His main research interest includes lattice-based cryptography.

梁志闯, 1997 年生. 博士研究生. 主要研究方向为格密码.

Zheng Jieyu, born in 2000. PhD candidate. Her main research interests include post-quantum cryptography and cryptographic engineering.

郑婕妤, 2000 年生. 博士研究生. 主要研究方向为后量子密码、密码工程.

Zhao Yunlei, born in 1974. PhD, distinguished professor. His main research interests include post-quantum cryptography, cryptographic protocols, and theory of computing.

赵运磊, 1974 年生. 博士, 特聘教授. 主要研究方向为后量子密码、密码协议、计算理论.