

ARCANE-Mithril

算法规范与支撑文档

本提交材料来自归藏（ARCANE）团队，成员按字母顺序列出如下：

- Jose Maria Bermudo Mera, 比利时 PQShield 公司
- Joppe W. Bos, 比利时恩智浦半导体 (NXP Semiconductors)
- Angshuman Karmakar, 印度坎普尔理工学院 (Indian Institute of Technology Kanpur)
- Joost Renes, 美国恩智浦半导体 (NXP Semiconductors)
- Sujoy Sinha Roy, 奥地利格拉茨工业大学 (Graz University of Technology)
- Michiel Van Beirendonck, 比利时贝尔福特实验室 (Belfort Labs)
- Frederik Vercauteren, 比利时鲁汶大学 (KU Leuven)
- 王鹏 (Peng Wang), 中国科学院大学密码学院
- 王跃武 (Yuewu Wang), 中国科学院大学密码学院
- 郑昉昱 (Fangyu Zheng), 中国科学院大学密码学院

联系人： 王鹏 (Peng Wang)

电子邮箱： p-wang@ucas.ac.cn

网站： <https://soc.ucas.ac.cn/index.php/zh/ARCANE>
<https://www.pq-arcane.org>

Contents

1	归藏 (ARCANE) 算法套件	2
2	引言	5
2.1	基本方法概述	5
2.2	与现有工作的关系	6
3	预备知识	10
3.1	符号约定	10
3.2	格密码学 (Lattice-based Cryptography)	11
3.2.1	误差学习 (Learning With Errors)	11
3.2.2	舍入学习 (Learning With Rounding)	11
3.2.3	舍入学习的安全性	12
3.3	作为结构化模的环	13
3.4	根式环舍入学习 (Radical Ring-LWR)	14
4	算法描述	15
5	安全声明与分析	21
5.1	理论安全性	21
5.1.1	软猬 (Mithril) .PKE 的 IND-CPA 安全性	22
5.1.2	软猬 (Mithril) .KEM 的 IND-CCA 安全性	22
5.1.3	RR-LWR 实例的安全性	23
5.1.4	具体攻击估计方法	24
5.2	实际安全性	26
5.2.1	解密失败攻击	26
5.2.2	侧信道攻击	26
6	性能评估	27
6.1	算术实现	27
6.2	性能分析	27
6.3	性能测试	28
A	SageMath 脚本	31
A.1	安全性估计	31
A.2	失败概率	33
B	检查清单	38

勘误与修订

版本 1 (2026 年 6 月 30 日) 更新 ARCANE-Mithril 算法规范与支撑文档的提交版本.

版本 1.1 (2026 年 8 月 11 日)

- 更正算法 1 中 软猬 (Mithril) .PKE.Enc 函数第 2 行的笔误: $(\frac{p}{2.t} - \frac{q}{2.p})$ 应为 $(\frac{q}{2.p} - \frac{p}{2.t})$, 以符合标准 Ring-LWR 和 Module-LWR 定义. 该错误影响解密失败率, 已相应更新实现.
- ARCANE-软猬 (Mithril) 规范 v1.0 曾说明: “与先前 *Saber* 型估计的做法相同, 我们启发式地将共享秘密各比特的失败事件建模为相互独立; 本文不声称严格证明了所有系数相关性。”此外, 还曾说明: “根式环结构中, $\mathbf{b}$ 与 $\mathbf{s}'$ 的第 i 行内积包含 $k-1-i$ 次乘以 $y+2$ 的操作, 见式 (6). 这会增大系数范数, 故在失败概率计算中计入最坏情况因子 3.”经进一步建模发现, $y+2$ 的贡献会在共享秘密比特间产生相关性, 使方差超过已计入的因子 3. 因此, 已更新第 5.2.1 节的失败概率模型, 以计算 $(y+2)\mathbf{e}\mathbf{s}'$ 的完整分布, 不再依赖标准 Ring-LWR 之外的启发式假设. 由于这会提高 软猬 (Mithril) -256 和 软猬 (Mithril) -512 的失败概率, 已相应更新表 2 中的参数集及实现.

版本 1.2 (2026 年 9 月 22 日)

- 更新新版 软猬 (Mithril) -256 和 软猬 (Mithril) -512 的性能测试及结果.

1 归藏 (ARCANE) 算法套件

后量子密码学 (post-quantum cryptography) 要求算法能够抵抗量子敌手, 并可在密钥交换 (或封装)、数字签名等应用中取代 RSA、椭圆曲线密码学 (Elliptic Curve Cryptography, ECC) 等传统方案. 归藏 (ARCANE) 算法套件包含两个算法:

- 归藏-软猬 (ARCANE-Mithril) : 一种后量子密钥封装机制 (Key Encapsulation Mechanism, KEM);
- 归藏-玄色 (ARCANE-Octarine) : 一种后量子数字签名 (Digital Signature, DS) 方案.

下面重点介绍 软猬 (Mithril) 与 玄色 (Octarine) 的共有特性.

创新性. 其安全性基于根式环舍入学习 (Radical Ring Learning-With-Rounding, RR-LWR) 问题 [BRVW26], 该问题是根式数域上的环舍入学习 (Ring-LWR) 变体. 在这一变体中, 舍入是在迁移后的单项式基中逐系数进行的; 该基即根式环单项式基在定理 4 所形式化的数域同构下的像. 它可作为模舍入学习 (Module Learning-With-Rounding, Module-LWR) 的替代方案, 在保留 Ring-LWR 大部分性能优势的同时, 继承 Module-LWR 参数选择灵活的优点. KEM 和 DS 方案均在如下形式的根式环上运算:

$$\mathbf{S}_{q,n,k} = (\mathbb{Z}_q[y]/(y^n + 1))[x]/(x^k - y - 2), \quad (1)$$

其中, q 与 n 均为二的幂, k 为任意正整数. 它与数域表示 $\mathbf{S}_{q,n,k} \cong \mathbb{Z}_q[x]/((x^k - 2)^n + 1)$ 之间存在直接的同构 [BRVW26, 第 4 节], 由此得到本方案迁移基 Ring-LWR 变体所采用的根式环表示. 环 $\mathbf{S}_{q,n,k}$ 是第 $2n$ 个分圆域的 k 次根式扩张. 因此, 可按需采用数论变换 (Number Theoretic Transform, NTT) 等标准实现技术或其他多项式乘法例程; 与模结构相比, 根式扩张还能更高效地生成矩阵.

我们选取二的幂作为 q , 而非素数. 这使 软猬 (Mithril) 在设计上与 Saber [DKR⁺20, 第 A.2 节] 相近, 但二者选用的理想格不同 (本方案采用 RR-LWR, 而非 Module-LWR). 我们还定义了数字签名方案 玄色 (Octarine); 该方案遵循成熟的带中止 Fiat-Shamir (Fiat-Shamir with Aborts) 范式 [Lyu09, Lyu12], 并将其适配到 RR-LWR 场景. Okada 等人此前提出过基于 LWR 的签名方案 MLWRSign [OTF⁺20]. 我们的做法有所不同: 将 $\mathbf{s}_2$ 定义为由公钥确定的确定性误差 $\mathbf{s}_2 = \mathbf{A}\mathbf{s}_1 - \frac{q}{p} \lfloor \frac{p}{q} \mathbf{A}\mathbf{s}_1 \rfloor$, 而不是 MLWRSign 中的有理数值而 MLWRSign 使用 $\lfloor \frac{p}{q} \mathbf{A}\mathbf{s}_1 \rfloor - \frac{p}{q} \mathbf{A}\mathbf{s}_1$. 此外, 我们在密钥生成阶段而非签名生成阶段计算 $\mathbf{s}_2$. 这简化了设计, 使整体构造较 MLWRSign 更接近 CRYSTALS-Dilithium [DKL⁺18].

简洁性. 我们采用根式数域上的 Ring-LWR 变体, 并在上述迁移后的单项式基中逐系数舍入; 具体估计计入了相对于通常典范嵌入视角所产生的基失真. 所得 KEM 和数字签名方案以较为保守的安全概念为目标: KEM 达到自适应选择密文攻击下的不可区分性 (IND-CCA2) [BDPR98], 数字签名则在随机预言机模型 (Random Oracle Model, ROM) 中达到适应性选择消息攻击下的强不可伪造性 (SUF-CMA) [ADR02].

我们从多个方面简化现有设计. 首先, 采用 (根式) 环结构而非模结构. 其次, 通过使用二的幂模数 q , 尽量降低各种参数选择的复杂性. 所有采样均使用模二的幂 α 的区间 $[-\alpha, \alpha - 1]$ 上的均匀分布, 并尽可能在各参数集中统一 α 的取值. 只需生成 $\log_2(2\alpha)$ 个随机比特, 即可产生 $[-\alpha, \alpha - 1]$ 中的随机元素. 因此, 在对秘密值、公有矩阵及掩蔽向量的系数采样时无需拒绝采样, 也无需丢弃任何 (伪) 随机生成的比特. 我们选择的参数使失败概率始终可忽略不计. 尤

其对于数字签名方案，分解参数和 Fiat-Shamir 中止拒绝循环的参数均基于二的幂，因此只需检查是否发生进位即可轻松完成边界检查。我们还减少了签名生成中的拒绝次数，以降低签名耗时的方差，从而可能更便于应用协议采用。例如，玄色 (Octarine) -128 完成一次签名平均需要 2.13 轮，标准差为 1.55；相比之下，Dilithium2 平均需要 4.25 轮，标准差为 3.72。

灵活性. 本方案继承了格密码系统的若干优点。选择 13 比特或 24 比特的二的幂作为 q ，可使模 q 的值容纳在大多数现代 CPU 字中，从而易于实现。我们作出以下设计选择，以便在不同体系结构上实现安全而灵活的实现：

- 尽管素数模数 q 的 LWR 可借助渐近高效的 NTT 获得快速实现优势，已有研究表明，二的幂 q 更适合掩蔽 [CJRR99, PR13]。为防御侧信道攻击和故障攻击，通常需要采用掩蔽，这对嵌入式系统及高保障实现尤为重要。构造掩蔽实现时常用的算术到布尔转换（及其逆转换）[CGV14, BBE⁺18] 在模数为二的幂时复杂度更低；无需拒绝采样的采样例程亦是如此。嵌入式系统本就受到资源约束，简化侧信道与故障防护对策有助于降低实现成本。
- 使用根式环可缩小公有矩阵，直接降低存储需求。例如，与 ML-DSA-87 的矩阵相比，本方案矩阵的大小几乎缩小到七分之一：若每个系数使用 32 比特，ML-DSA-87 的矩阵需要超过 57.3 kB；采用紧凑编码存储时仍需超过 41.2 kB。相比之下，本方案公有矩阵在每系数 32 比特时约需 8.2 kB，紧凑编码后仅需 6.1 kB。这不仅降低了矩阵生成成本，也减少了矩阵在多次使用期间（尤其是生成签名时）的存储开销，从而为受限系统提供更好的时间-存储权衡。
- 我们在标准域中表示所有变量，以避免算法描述依赖特定实现或算法技术。以特定形式（如 NTT 域）表示元素虽可能带来性能优势，却可能限制算法面向不同平台时的可移植性。归藏算法套件的算术可使用 NTT [CT65, Pol71]、Toom-Cook [Too63, Coo66]、Karatsuba [KO62] 高效实现，也可利用现有硬件执行 Kronecker 代换 [Kro82, BRvV22]。我们的结果表明，其性能相对于先进格密码系统仍具竞争力。特别地，将软猬 (Mithril) 私钥多项式 s 表示在普通域而非 NTT 域中时，每个系数只需 $\log_2(2\eta)$ 比特；Kyber 则需 $\log_2(q)$ 比特，因而本方案可显著缩小私钥。

兼容性. 该 RR-LWR 族的安全级别主要由格维数决定，而格维数取决于多项式次数 n 和根式扩张次数 k 。增大取二的幂的 n 可提高算术运算的总体效率；改变 k 则可像 Module-LWR 一样，更细粒度地控制安全级别。虽然 $n = 256$ 是 KEM 的常见选择，但此时格维数只能选为 512 或 768，因而难以恰当地达到 128 比特安全级别：维数 512 可能安全性不足（或安全余量极小），而维数 768 通常又会提供过高的安全级别。因此，我们对 KEM 采取保守设计，选择 $n = 128$ 、 $k = 5$ ，使用维数为 640 的格。具体而言，在精化攻击模型下，这将比特安全性从 Kyber 和 Saber 的 140 比特提高到 153 比特。基于根式扩张的理想格可由大小随 k 线性增长的元素表示，而 Module-LWE/LWR 则随 k 二次增长，因而我们减轻了矩阵生成成本所受的部分影响。与使用较小格相比，这一设计策略会略微增大密钥和密文，并影响总体性能；但为提高对所达到的 128 比特安全级别的信心，我们认为这是可以接受的。

我们在所有 KEM 参数集中统一使用 $n = 128$ ，以简化软硬件实现。对数字签名，我们在所有参数集中选择 $n = 1024$ ，并令 $k = 1, 2, 5$ ，以最大限度发挥分圆子环结构的性能。此外还有许多可调参数，会影响性能、密钥大小、拒绝率及签名耗时方差等。我们的参数选择使公钥和签名大小与其他格方案相近，同时提高若干常用参数集的性能并降低签名生成方差。例如，与 CRYSTALS-Dilithium-2 [DKL⁺18] 相比，归藏-玄色 (ARCANE-Octarine) -128 生成签名的期望重复次数约减半（分别为 2.1 次与 4.3 次）。

为最大限度复用数字签名与 KEM 的实现，我们将所有采样分布选为模二的幂 α 的区间 $[-\alpha, \alpha - 1]$ 上的均匀分布。数字签名和 KEM 均默认选择 $\alpha = 2$ ，仅有一个参数集例外，因为继续采用该值会造成过大的安全性影响。基于二的幂的打包和舍入操作效率高，并能最优利用所需比特。两种算法都选取适合 NTT 的次数 n （即二的幂），从而可在 NTT 对目标平台最优时采用 NTT 算术。尽管 KEM 与数字签名的 n 不同，仍可将 $n = 128$ 的乘法用作 1024 次乘法的子例程，得到共享实现。例如，KEM 取 $q = 2^{13}$ ，数字签名取 $q = 2^{24}$ 时，可选择适合 NTT 的 32 比特素数 p_1 。该素数既可用于 KEM，也可借助环 $\mathbb{Z}_{p_1 p_2} \cong \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2}$ 用于数字签名。实现时，可用单个模块在任意素数 p 对应的 $\mathbb{Z}_p$ 上运算，复用绝大部分代码；其上只需增加一个小型双素数中国剩余定理（Chinese Remainder Theorem, CRT）计算层，即可从 $\mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2}$ 转换回 $\mathbb{Z}_p$ 。参考实现和优化实现展示了这种方法及其效率。

应注意，素数的选取以及是否使用 NTT 均不属于本规范，而纯属实现选择；若其他实现方法性能更优，也可采用。

可扩展性。 这些构造不仅用于本文规定的具体实例，还旨在充当可复用的构件。在外围协议提供所需的认证、密钥确认和组合安全保证的前提下，KEM 可集成到更高层协议中，用于临时密钥建立、KEM-数据封装机制（Data Encapsulation Mechanism, DEM）加密以及认证密钥建立机制。它还可按混合模式与传统或其他后量子密钥建立机制结合，以支持迁移期间的算法敏捷性。

在套件层面，方案的模块化组织方式也为未来变体留出了空间。具体而言，可在不改变底层根式环运算的情况下考虑其他参数配置、哈希/可扩展输出函数（Extendable-Output Function, XOF）/密钥派生函数（Key Derivation Function, KDF）实例或实现配置。不过，此类变体仍应附有更新后的正确性界、具体安全性估计和实现验证。

玄色（Octarine）支持通用消息签名、代码签名、数字证书和安全证明等标准签名应用。此外，由于该方案遵循 Fiat-Shamir 范式，相同的代数环境还可作为更高级 Fiat-Shamir 类签名构造（包括门限签名 [PN25]）的起点。

因此，RR-LWR 框架不仅应被视为一组固定的 KEM 和签名实例，还应被视为一个平台，用于未来的协议集成以及复用相同根式环结构、经过审慎分析的变体。

不同平台上的性能优势。 归藏（ARCANE）套件采用保守的设计选择，因而可针对广泛的目标平台优化实现。特别地，本规范有意将多项式算术方法的选择与算法本身解耦：实现可按需采用基于 NTT、Toom-Cook、Karatsuba、朴素乘法或体系结构专用的技术。

为与成熟格方案进行有意义的比较，性能章节除报告基于指定 `pseudoXOF` 实例的提交测量结果外，还在适用时报告基于 SHAKE 的基准配置。在 KEM 场景下，基于 SHAKE 的 AVX2 比较涵盖 软猬（Mithril）-128 和 软猬（Mithril）-256；软猬（Mithril）-128 的 AVX2 优化实现比 ML-KEM-512 慢约 1.4 倍。造成这一差异的部分原因是，软猬（Mithril）-128 的估计安全余量高于 ML-KEM-512。在更高安全级别下，软猬（Mithril）-256 与 ML-KEM-1024 的性能差距缩小到约 1.1 倍，同时 软猬（Mithril）仍提供更大的估计安全余量。

在数字签名场景下，玄色（Octarine）-128 的密钥生成和签名速度快于 ML-DSA-44，而 ML-DSA-44 的验证速度仍更快。在更高安全级别下，玄色（Octarine）-256 在所测签名与验证操作上均优于 ML-DSA-87，签名和验证加速比分别为 1.13 和 1.23。这些结果表明，归藏（ARCANE）的设计在保留保守安全余量的同时实现了有竞争力的性能，并为平台专用优化留下了充分的灵活性。

应注意，归藏（ARCANE）方案在降低存储消耗及减少公钥矩阵伪随机数生成等方面的优

势，对于 XOF 成本占主导的较小型 Arm 或 RISC-V 平台内核会愈发显著。此外，简洁的设计选择应能显著便利侧信道与故障防护实现；Saber 相对 Kyber 的优势即是一个例证。同时，它也为可基于各类乘法器的硬件设计保留了灵活性。我们计划在本规范的后续版本中量化这些改进。

2 引言

本文以下部分重点给出 KEM 归藏-软猬 (ARCANE-Mithril) 的规范（亦使用简称软猬 (Mithril)）。

2.1 基本方法概述

软猬 (Mithril) 的安全性基于根式环舍入学习 (Radical Ring Learning With Rounding, RR-LWR) 问题的困难性。非形式化地说，RR-LWR 是在根式环 $\mathbf{S}_{q,n,k}$ 上实例化的 LWR 问题，其中算术在一个扩张二的幂分圆环的结构化环中进行。给定由 $\mathbf{S}_{q,n,k}$ 导出的均匀随机公开数据，RR-LWR 问题要求区分含秘密元素的舍入线性乘积与均匀随机值，或恢复秘密本身。舍入在迁移后的单项式基中逐系数进行，该基是根式环单项式基在定理 4 的数域同构下的像。因此，该假设是根式数域上的 Ring-LWR 变体，而非采用典范嵌入形式的 Ring-LWR。环 $\mathbf{S}_{q,n,k}$ 包含分圆子环 $\mathbf{R}_{q,n}$ ，既支持高效算术，又保留 LWR 类构造所需的代数性质。从格的角度看，RR-LWR 可与维数正比于 nk 的格上的 LWR 实例比较；参数 k 为调节安全级别提供了额外自由度，而具体估计则计入由此产生的基失真。

软猬 (Mithril) 的构造未引入 RR-LWR 之外的新密码学假设。密钥封装机制通过 Fujisaki-Okamoto 型变换由公钥加密方案得到，其安全性依赖于 RR-LWR 的假定困难性以及所用哈希函数的标准性质。

环结构的选择与参数选择准则。 软猬 (Mithril) 的设计目标是在保留 LWR 类构造简洁性和效率的同时，实现灵活的参数选择。为此，方案在根式环 $\mathbf{S}_{q,n,k}$ 上实例化，其中 n 为基环次数， k 为根式扩张次数。

在固定二的幂基环次数 n 的同时改变 k ，会改变总代数次数 nk ；该值是通用攻击估计所用的有效格维数。由此，无需改变底层分圆次数即可细粒度地调节安全性。子环 $\mathbf{R}_{q,n}$ 是二的幂分圆环，支持高效多项式算术，并与成熟的 LWR 类方案实现技术兼容。

模数 p 、 q 和 t 均选为二的幂，其比特长度分别记为 ϵ_p 、 ϵ_q 和 ϵ_t 。这一选择简化了 LWR 构造中的舍入和缩放操作，因为这些操作均可实现为逐系数移位与截断。各模数的相对大小经过选择，在以高概率保证正确解密的同时，使噪声分量与消息分量充分分离。

最后，参数 ℓ 决定加密与解密期间从环元素中提取的系数个数。方案仅对关于 x 的 ℓ 个最高次系数进行操作，在提供所需大小消息空间的同时避免不必要的计算。 ℓ 的选择与 n 、 k 及舍入参数相协调，以同时保证正确性与安全性。

完整参数集概览见表 2。

失败概率。 软猬 (Mithril) 底层公钥加密方案的正确性依赖于底层 LWR 构造的正确性。由于使用舍入和系数截断，解密并非完全正确，而会以较小概率失败。

加密时, 通过向舍入后的线性乘积中加入经缩放的消息分量, 将消息嵌入密文. 解密通过逆转缩放和舍入操作来恢复消息. 当累积舍入误差使某个系数越过判决边界, 导致比特解码错误时, 即发生解密失败.

解密误差有两个来源. 首先, LWR 舍入操作会引入近似误差, 其大小取决于模数 p 与 q 之比. 其次, 消息编解码仅使用关于 x 的 ℓ 个最高次系数, 这限制了可用误差裕量. 这些影响通过选择参数 ϵ_p 、 ϵ_q 、 ϵ_t 和 ℓ 加以控制.

对于本文规定的参数集, 估计解密失败概率可忽略不计, 通常对安全参数 λ 形如 $2^{-\lambda}$. 密钥封装机制的设计计入了该概率: 在 Fujisaki–Okamoto 变换 [FO99] 中, 解密失败以加性项计入安全界.

因此, 尽管该加密方案并非完全正确, 但对于预期安全级别, 其估计失败概率足够小; 在指定 KEM 构造中使用 软猬 (Mithril) 时, 该概率已计入其总体安全性核算.

失败概率的更详细分析见第 5.2.1 节.

采样. 本方案大量使用从二进制种子进行的确定性采样来生成公有矩阵和秘密值. 这一设计简化了密钥存储与传输, 支持公开参数的紧凑表示, 并使不同实现的行为可复现. 方案所用的全部随机性均通过明确定义的扩展函数由种子显式导出.

部分环元素. 通过参数 ℓ 使用部分环元素, 使方案能够避免不必要的计算. 将消息编解码限制在关于 x 的 ℓ 个最高次系数, 可在保留所需消息空间和正确性保证的同时降低计算开销. 该设计选择不改变底层安全假设, 同时便于高效实现; 与使用完整环元素相比, 还可降低失败概率.

Fujisaki–Okamoto 变换. 该公钥加密方案的结构与从 IND-CPA 安全加密到 IND-CCA 安全密钥封装机制的标准变换兼容. 特别地, 加密算法可表述为显式接收随机性, 因而无需修改方案的核心算术即可应用 Fujisaki–Okamoto 型变换 [FO99].

高保障实现. 本算法规范旨在支持常数时间实现, 并简化针对侧信道攻击和故障攻击的安全实现. 所有算术操作均以逐系数方式定义, 密钥封装机制中的条件行为可用常数时间选择原语实现. 这些考虑有助于降低实际部署中的侧信道泄漏风险.

2.2 与现有工作的关系

下面将 软猬 (Mithril) 与最相关的现有格 KEM 进行比较. 我们的设计旨在改善概念简洁性、实现效率、存储用量与加固实现稳健性之间的总体平衡. 具体而言, 目标不仅是获得具有竞争力的渐近安全性与性能, 还要降低在受限环境及侧信道敏感环境中部署方案的工程复杂度. 与 LWR 类方案的详细比较见表 1.

CRYSTALS-Kyber / ML-KEM. 软猬 (Mithril) 与 ML-KEM (原名 CRYSTALS-Kyber) 最显著的区别在于, 软猬 (Mithril) 基于 Ring-LWR, 而 ML-KEM 基于 Module-LWE. 底层困难问题之间关系的详细回顾见第 3 节.

与许多可将 q 选为素数的 LWE 类设计相比, 一个显著区别是: 由于 $\mathbf{Z}_q$ 中缺少适当的单位根, 无法原生应用 NTT. 这会影响 $\mathbf{R}_q$ 中多项式算术的效率, 使 LWR 实现通常慢于 LWE. 不过, 文献 [CHK⁺21] 首次表明, 通过将 $\mathbf{R}_q$ 嵌入模数为素数 q' 的更大 NTT 友好环, 仍可使用渐近更快的 NTT. 尽管这会使模数的比特长度至少增大一倍, 但在某些平台上, 其算术仍可能

方案	$f(x)$	k	ℓ	ϵ_q	ϵ_p	ϵ_t	D_A	D_s	B
软猬 (Mithril) -128	$(x^k - 2)^{128} + 1$	5	1	13	11	3	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
软猬 (Mithril) -256	$(x^k - 2)^{128} + 1$	9	2	13	11	3	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
软猬 (Mithril) -512	$(x^k - 2)^{128} + 1$	17	4	13	11	8	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
LightSaber [DKRV18]	$x^{256} + 1$	2	1	13	10	3	$\mathcal{U}_{\epsilon_q}$	β_5	1
Saber [DKRV18]	$x^{256} + 1$	3	1	13	10	4	$\mathcal{U}_{\epsilon_q}$	β_4	1
FireSaber [DKRV18]	$x^{256} + 1$	4	1	13	10	6	$\mathcal{U}_{\epsilon_q}$	β_3	1
uLightSaber [DKR ⁺ 20, Sec. A.2]	$x^{256} + 1$	2	1	12	10	3	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
uSaber [DKR ⁺ 20, Sec. A.2]	$x^{256} + 1$	3	1	12	10	4	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
uFireSaber [DKR ⁺ 20, Sec. A.2]	$x^{256} + 1$	4	1	12	10	6	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_2$	1
Florete [MKKV21, KNK ⁺ 25]	$x^{768} - x^{384} + 1$	1	1	10	9	3	$\mathcal{U}_{\epsilon_q}$	β_4	1
Espada [MKKV21, KNK ⁺ 25]	$x^{64} + 1$	12	1	15	13	3	$\mathcal{U}_{\epsilon_q}$	β_4	4
LightSable [MKKV21, KNK ⁺ 25]	$x^{256} + 1$	2	1	11	9	2	$\mathcal{U}_{\epsilon_q}$	β_1	1
Sable [MKKV21, KNK ⁺ 25]	$x^{256} + 1$	3	1	11	9	4	$\mathcal{U}_{\epsilon_q}$	β_1	1
FireSable [MKKV21, KNK ⁺ 25]	$x^{256} + 1$	4	1	11	9	2	$\mathcal{U}_{\epsilon_q}$	β_1	1
TiMER* [HCCSS24]	$x^{256} + 1$	2	1	10	8	3	$\mathcal{U}_{\epsilon_q}$	spCBD	2
SMAUG-T128* [HCCSS24]	$x^{256} + 1$	2	1	10	8	5	$\mathcal{U}_{\epsilon_q}$	spCBD	1
SMAUG-T192* [HCCSS24]	$x^{256} + 1$	3	1	10	8	8	$\mathcal{U}_{\epsilon_q}$	spCBD	1
SMAUG-T256* [HCCSS24]	$x^{256} + 1$	5	1	10	8	6	$\mathcal{U}_{\epsilon_q}$	spCBD	1

Table 1: 基于 Ring/Module-LWR 的后量子密码方案参数比较. 除 软猬 (Mithril) 外, 所有方案均采用 Module-LWR 参数, 其环为 $\mathbf{R} = \mathbf{Z}[x]/(f(x))$, 矩阵分布为 D_A , $\mathbf{s}$ 和/或 $\mathbf{s}'$ 的 (临时) 私钥分布为 D_s . 列 B 给出每个输入消息系数的比特数. 此处, ϵ_q 、 ϵ_p 和 ϵ_t 分别表示 $\log_2 q$ 、 $\log_2 p$ 和 $\log_2 t$. 对 Module-LWR 方案, k 表示模秩; 对 软猬 (Mithril), 同一列表示根式扩张次数.

* 密钥生成使用 LWE, 封装/解封装使用 LWR.

优于 Toom-Cook 或 Karatsuba. 例如, 文献 [CHK⁺21] 对 Saber 与 FireSaber 参数集使用 6 层大小为 256 的负循环 NTT, 随后进行 4×4 朴素乘法, 并取

$$q' = 25166081 = 196610 \cdot 128 + 1$$

所得乘法比 $\mathbf{R}_{8192}$ 中的乘法快约 60%. 总的来说, 尽管 LWR 与 LWE 的效率确有可测差异, 但二者性能足够接近, 对许多实际应用并不显著.

此外, 虽然 软猬 (Mithril) 基于 Ring-LWR 而非模问题, 但通过调节参数 k , 环 $\mathbf{S}_{q,n,k}$ 在保留 Ring-LWR 大部分性能优势的同时, 继承了 Module-LWR 参数选择灵活的优点. 使用根式环可缩小公有矩阵, 直接降低采样成本和存储需求. 全部采样均采用模二的幂 α 的区间 $[-\alpha, \alpha - 1]$ 上的均匀分布, 并尽可能在各参数集中统一 α . 只需生成 $\log_2(2\alpha)$ 个随机比特即可产生区间中的随机元素, 因此无需拒绝采样, 也无需丢弃任何 (伪) 随机生成的比特.

对于高保障实现, 软猬 (Mithril) 等基于 LWR 的方案有两项关键优势:

- 二的幂模数简化掩蔽转换;
- 舍入比噪声采样更容易掩蔽.

这些性质促成了一阶掩蔽 Saber 框架：其在 ARM Cortex-M4 上的开销为 2.5 倍，而 LWE 类方案为 5 倍。Saber 与 uSaber 的高阶掩蔽实现 [KDvB⁺22] 表明，随着 t 增大，LWR 方案的扩展性优于 LWE 方案。均匀秘密分布可简化采样，因而效率最高。Scabbard 套件中基于 LWR 的 KEM Florete、Espada 和 Sable 的掩蔽实现也因使用二的幂模数而优于 ML-KEM；文献 [KKV23] 将这些方案描述为掩蔽友好方案。得益于高效参数选择，无论掩蔽阶数如何，掩蔽 Florete 和掩蔽 Sable 甚至快于掩蔽 Saber。尽管文献 [KDvB⁺22] 将 uSaber 视为 Saber 的掩蔽友好版本，Sable 的性能仍优于 uSaber，且该优势在掩蔽后依然保持。软猬 (Mithril) 的设计同样受益于这些优势。

在硬件方面，文献 [FVR⁺22] 提出了用于掩蔽 Kyber (ML-KEM) 和 Saber 的 RISC-V 指令集扩展，结果确认了 LWR 相对于 LWE 方案的掩蔽效率优势。文献 [ABC⁺23] 进一步优化了二的幂模数下的掩蔽多项式乘法；其 Saber 硬件仅以 1.7 倍面积开销实现一阶抗性，并只需 340 个周期完成乘法，显著优于通用掩蔽方法。

最后，我们对 128 比特参数集采取更保守的设计，选择 $n = 128$ 、 $k = 5$ ，使用维数为 640 的格。在精化攻击模型下，这具体将比特安全性从 ML-KEM 的 140 比特提高到 153 比特。基于根式扩张的理想格可由大小随 k 线性增长的元素表示，而 M-LWE/LWR 随 k 二次增长，因而可减轻矩阵生成成本所受的部分影响。与较小格相比，密钥与密文仍会增大、总体性能也会受到影响，但为提高对所达到 128 比特安全级别的信心，我们认为这是可以接受的。

(u)Saber. Saber 方案 [DKRV18] 是首个完全基于 (模) LWR 的 KEM。它于 2016 年提交 NIST PQC 竞赛，并进入第三轮 [DKR⁺20]。由于完全依赖 LWR，无需从误差分布采样。Saber 以简洁和高效为设计目标，在所有参数集中固定使用环 $\mathbf{Z}_q[x]/(x^{256} + 1)$ 和 $q = 2^{13}$ ，仅改变模维数、密文模数及私钥分布。模二的幂 q 的算术易于实现，无需模约减或拒绝采样。NIST 提案定义了 LightSaber、Saber 和 FireSaber 三个参数集，分别对应 NIST 安全级别 1、3 和 5。私钥从参数 $\eta = 3, 4, 5$ 的二项分布 β_η 中选取。选择该 q 的缺点是 $\mathbf{Z}_q$ 不原生支持 NTT，因此早期实现依赖 Toom-Cook。不过，将其嵌入模数经过适当选择（比特长度约为 q 的两倍）的更大环后，仍可用 NTT 实现多项式乘法，只是效率不及 ML-KEM 等方案。

均匀 Saber (Uniform Saber, uSaber) [DKR⁺20, 第 A.2 节] 是 Saber 第三轮提交材料中提出、但不属于主提交方案的变体。主要区别是秘密向量 $\mathbf{s}$ 和 $\mathbf{s}'$ 从均匀分布 $\mathcal{U}_2$ 而非二项分布 β_η 中采样。因此，秘密系数位于区间 $[-2, 1]$ ，而非 Saber 的 $[-\eta, \eta]$ ，每个系数恰好可用 2 比特表示。

uSaber 的主要优点是简化了秘密生成过程；该过程用于密钥生成和加密，进而用于 KEM 的封装与解封装。关键收益是：要达到相近秘密方差和安全级别，所需 (伪) 随机比特显著减少。例如，uSaber 每个秘密系数仅需 2 个随机比特，而采用二项采样的标准 Saber 需要 6 至 10 个。这会加快哈希并降低计算开销和攻击面。由于哈希高度非线性，这种缩减对掩蔽实现尤其有价值。

此外，uSaber 不再需要计算二项分布。这一简化使实现更直接，尤其是在掩蔽场景下，非线性的二项采样会增加大量复杂度。去除该步骤不仅提高效率，也缩小攻击面，使设计更安全、更实用。

软猬 (Mithril) 遵循 uSaber 的设计原则，并将其扩展到 RR-LWR 场景。我们认可 (小) 均匀分布对简洁、安全实现 (包括侧信道防护) 的益处。采用 RR-LWR 可缩小矩阵 (更准确地说，是以环元素替代矩阵)，从而提高相对于 uSaber 的效率，并更细粒度地提升安全级别。将 $n = 128$ (而非 $n = 256$) 与 $k = 5$ 结合，可达到严格高于 uSaber 的 128 比特安全级别。

Scabbard 套件: Florete、Espada 与 Sable. 在 LWR 类方案的乘法例程、密码分析和参数选择等多方面改进基础上, NIST 标准化期间提出了由三个 KEM 组成的 Scabbard 套件 [MKKV21, KNK+25]. 三个 KEM 分别针对速度、面积需求、存储占用及并行实现等方面改进当时的先进水平.

第一个方案 Florete 是 Ring-LWR 方案. Saber 在 $q = 2^{13}$ 的 $\mathbf{Z}_q$ 上使用模秩 $k = 3$ 和多项式次数 $n = 256$; 与之不同, Florete 依赖由环 $\mathbf{Z}_q[x]/(x^{768} - x^{384} + 1)$ ($q = 2^{10}$) 定义的理想格, 并固定模秩 $k = 1$. 以 Saber 的 192 比特安全级别为目标, 这将 $\mathbf{A}$ 的矩阵维数从 3×3 降为单个环元素, 省去矩阵-向量乘法. 该方法生成 $\mathbf{A}$ 所需的伪随机字节更少, 而该步骤可能是性能瓶颈 (尤其在软件中). 设计还可完全复用 Saber 的乘法例程. 其缺点是安全性比 Saber 低约 15 比特 [MKKV21, 表 2]. 后续工作 [KNK+25] 以类似方法分别使用多项式次数为 512 和 1024 的环, 以匹配 LightSaber 和 FireSaber 的安全级别. Florete 与 软猬 (Mithril) 一样也使用环, 但其环无法扩展到不同安全级别, 因此只以 192 比特安全级别为目标.

第二个方案 Espada 通过提高算法并行度来改善效率. 其主要思想是将含 256 个系数的大多项式乘法化为多个各含 64 个系数的小乘法. 尽管 Toom-Cook 或 Karatsuba 也可将 256×256 多项式乘法化为 64×64 乘法, 但无法提供同等并行度, 且其求值和插值阶段会增大乘法器. 将 $n = 64$ 作为底层格的原生多项式次数, 可降低基础乘法例程的实现成本, 尤其是在硬件 [MTK+20] 和微控制器上, 同时提高参数选择粒度. 为达到 128 比特安全级别, 模秩 k 因而从 3 增至 12 (其他安全级别类似). 尽管所需 64×64 多项式乘法总数多于 Saber, 但各乘法相互独立, 更易并行实现. 将 n 从 256 降至 64 要求每个系数选取 $B = 4$ 比特, 这会影响失败概率, 并最终要求将 q 增至 15 比特. 软猬 (Mithril) 与 Espada 的区别在于前者是基于环的 KEM. 软猬 (Mithril) 同样通过将 n 从 256 降至 128 来获得类似优势, 却无需为每个系数选择多个比特; 取 $B = 1$ 可避免影响失败概率.

最后一个方案 Sable 最接近 Saber 的设计, 但改变了私钥分布. LightSaber、Saber 和 FireSaber 分别取 $\eta = 5, 4, 3$, 从区间 $[-\eta, \eta]$ 上的中心二项分布 β_η 选取私钥系数; Sable 则在所有参数集中固定 $\eta = 1$, 使不同方案之间的切换更顺畅. 此时每个私钥系数仅需 2 比特存储, 与 uSaber 相同, 也有利于掩蔽实现. 总体而言, 该变化降低了二项乘法器的复杂度; 若按 p 和 q 的最大值实现, 还可在各参数集间统一乘法器设计. 为补偿安全性损失, 相应减小 p 与 q . 论文指出, Saber 对 p 和 q 的选择使用了错误估计, 因为舍入误差采用的是连续均匀分布而非离散均匀分布. LightSable、Sable 和 FireSable 参数集均假定离散均匀分布, 从而可更准确地选择参数和估计安全性. 软猬 (Mithril) 同样以小私钥系数为目标, 但允许比 Sable 更大的私钥区间, 并仅使用均匀分布. 这简化了采样过程, 也更高效地利用每个采样比特.

SMAUG 及相关方案. SMAUG-T [HCCSS24] 由 SMAUG [CCHY24] 与 TiGER [PJP+22] 合并而成, 是较新的 LWR 类提案, 并入选韩国 PQC 标准化. 其密钥生成归约到 LWE, 而封装/解封装依赖 LWR. 生成临时封装秘密时使用稀疏中心二项分布: 更准确地说, 对 $0 \leq r < 1/2$, 所得三元值以概率 $1 - 2r$ 为 0, 以概率 r 分别为 -1 和 1 . r 取 $1/8$ 或 $3/16$, 分母特意选为二的幂以便实现. 应注意, 由于部分采样器并非常数时间实现, 该设计从 v2.0 到 v4.0 之间曾作更新, 因此必须谨慎实现. 稀疏秘密可减少解密失败并提高 KEM 效率. v3.0 在规范中加入新参数集 TiMER, 通过 D2 误差协调机制进一步降低解密失败概率. 它将每个消息比特编码到两个多项式系数中, 以额外编解码为代价降低失败概率.

软猬 (Mithril) 的密钥生成不采用 LWE 假设, 私钥也不使用稀疏分布. 我们的目标是降低实现复杂度, 尤其是与计时攻击及其他侧信道攻击有关的复杂度. 稀疏秘密虽可能带来性能优势, 软猬 (Mithril) 仍采取保守方案, 以安全性为先.

3 预备知识

本节给出理解本规范所需的全部背景知识.

3.1 符号约定

环元素. 整数环记为 $\mathbf{Z}$; 对于正整数 q , $\mathbf{Z}_q$ 表示商环 $\mathbf{Z}/q\mathbf{Z}$. 给定正整数 m , 第 m 个分圆多项式 (cyclotomic polynomial) 记为 $\Phi_m(x)$, 其定义为以所有 m 次本原单位根为根的唯一首一多项式, 即

$$\Phi_m(x) = \prod_{\substack{k=1 \\ \gcd(k,m)=1}}^m \left(x - e^{2\pi i \frac{k}{m}} \right),$$

其中满足 $\gcd(k, m) = 1$ 的 $e^{2\pi i \frac{k}{m}}$ 为 m 次本原单位根. 可以证明 $\Phi_m(x)$ 的系数均为整数, 并由定义可知 $\deg(\Phi_m(x)) = \varphi(m)$, 其中 $\varphi(m)$ 是 Euler 欧拉函数, 用于统计小于 m 且与 m 互素的正整数个数. 众所周知, 当 $m = 2n$ 且 n 本身为二的幂时, $\Phi_m(x) = x^n + 1$. 多项式商环记为 $\mathbf{R}_q = \mathbf{Z}_q[x]/(f(x))$, 其中 $f(x) \in \mathbf{Z}[x]$ 为首一多项式.

设 n 为二的幂, k 为正整数. 定义分圆环 $\mathbf{R}_{q,n} = \mathbf{Z}_q[y]/(y^n + 1)$ 以及

$$\mathbf{S}_{q,n,k} = \mathbf{R}_{q,n}[x] / (x^k - y - 2)$$

作为 $\mathbf{R}_{q,n}$ 的 k 次根式扩张. $\mathbf{S}_{q,n,k}$ 中的元素表示为多项式 $g = \sum_{i=0}^{k-1} g_i x^i$, 其系数 $g_i \in \mathbf{R}_{q,n}$. 普通小写字母表示 $\mathbf{Z}$ 或 $\mathbf{Z}_q$ 中的元素; 粗体小写字母表示 $\mathbf{R}_{q,n}$ 或 $\mathbf{S}_{q,n,k}$ 中的元素 (即多项式或环元素); 粗体大写字母表示矩阵.

模表示. 对于偶整数 α 和 $r \in \mathbf{Z}$, 记 $r' = r \bmod \alpha$ 为区间 $[-\alpha/2, \alpha/2)$ 中满足 $r' \equiv r \pmod{\alpha}$ 的唯一代表元; 该区间包含 $-\alpha/2$, 不包含 $\alpha/2$. 当 α 为二的幂时, 这等价于提取 r 的最低 $\log_2(\alpha)$ 个有效比特. 除非另有说明, 本文假定所有变量均在有符号区间内表示. 在使用 32 位或 64 位字的现代 CPU 上实现该操作时, 若丢弃高位比特, 须特别注意符号扩展. 该操作以逐系数方式扩展到 $\mathbf{S}_{q,n,k}$ 中的元素.

范数. 对于环元素 $g = \sum_{i=0}^{k-1} g_i x^i \in \mathbf{S}_{q,n,k}$ 以及 $g_i = \sum_{j=0}^{n-1} g_{i,j} y^j \in \mathbf{R}_{q,n}$, 定义 $\|g\|_\infty = \max_{i,j} |g_{i,j} \bmod q|$.

采样分布. 用 $D_x(R)$ 表示从环 R 中采样变量 x 所用的分布. 若依据种子 σ 进行确定性采样, 则记为 $D_x(R; \sigma)$. 若使用多个种子 $\sigma_1, \dots, \sigma_m$, 则记为 $D_x(R; \sigma_1, \dots, \sigma_m)$, 并在采样前将它们串接为 $\sigma = \sigma_1 \mid \dots \mid \sigma_m$. 例如, 可用这种方式在各个种子后附加环索引.

哈希函数. 算法使用输出长度可变 (可扩展) 的哈希函数 $H(\text{input}, \text{outlen})$, 其中 outlen 以字节为单位. 每次调用 H 时均单独规定其输出长度. 安全性证明要求 H 表现为随机预言机, 并具有 $\lambda/2$ 比特碰撞抗性和 λ 比特原像抗性, 其中 λ 为目标安全级别. H 的具体实例将与最终参数集一并规定.

3.2 格密码学 (Lattice-based Cryptography)

理解舍入学习 (Learning With Rounding, LWR) 的理论与实践需要先介绍误差学习 (Learning With Errors, LWE) 问题 [Reg05] 及其较常用的结构化变体: 环误差学习 (Ring-LWE) [LPR10] 和模误差学习 (Module-LWE) [BGV12, LS15]. 感兴趣的读者可参阅有关 LWE 的综合性综述 [Reg10, Pei16].

3.2.1 误差学习 (Learning With Errors)

设模秩 d 为正整数, χ 表示 $\mathbf{R}$ 上的一个分布. 令 $s \in \mathbf{R}_q^d$ 为秘密向量, 定义分布 $A_{s,\chi}$ 为

$$A_{s,\chi} = (a, b = \langle a, s \rangle + e \bmod q) \in \mathbf{R}_q^d \times \mathbf{R}_q$$

其中, $a \leftarrow \mathcal{U}(\mathbf{R}_q^d)$ 为均匀随机采样值, $e \leftarrow \chi$ 为误差项.

搜索型 LWE 问题 (*search LWE problem*) 要求在获得来自 $A_{s,\chi}$ 的多项式数量的独立样本后恢复秘密 s ; 判定型 LWE 问题 (*decision LWE problem*) 则要求区分来自 $A_{s,\chi}$ 的样本与从 $\mathbf{R}_q^d \times \mathbf{R}_q$ 中均匀抽取的样本. 实践中常取 $\mathbf{R}_{q,n} = \mathbf{Z}_q[x]/(f(x))$, 其中 $f(x)$ 为分圆多项式

$$f(x) = \Phi_{2n}(x) = x^n + 1$$

其中 n 为二的幂. 在此 Ring-LWE 变体中, 所有运算均在环 $\mathbf{R}_q$ 内完成. 单个 Ring-LWE 样本可嵌入相当于 n 个普通 LWE (Plain-LWE) 样本的信息, 同时显著缩小密码密钥. 此外, 分圆多项式的选择允许使用数论变换 (NTT) [Pol71] 高效计算多项式乘法; NTT 是一种仅涉及整数算术的 Fourier 变换变体.

Module-LWE 问题 [BGV12, LS15] 同时推广了 Plain-LWE 和 Ring-LWE, 提供一种灵活的中间结构, 可在安全性与效率之间权衡. 在 Module-LWE 中, 秘密是多项式向量, 样本为 $(\mathbf{a}, b)$, 其中 $b = \langle \mathbf{a}, s \rangle + e$. 因此, Module-LWE 是 Plain-LWE 与 Ring-LWE 的共同推广. 令秩 $d = 1$ 即得到 Ring-LWE 问题; 令环为 $\mathbf{Z}$ 、秩 $d = n$ 则得到 Plain-LWE 问题. 此外, 对于给定的 Module-LWE 参数, 只需调整模秩即可改变问题难度, 也即方案安全性.

3.2.2 舍入学习 (Learning With Rounding)

舍入学习由文献 [BPR12] 提出, 可视为 LWE 的一种去随机化技术. 在 LWE 中计算 $\langle a, s \rangle + e \bmod q$, 即将误差项 e 加到内积上 (模 q). LWR 不采用加法, 而是将该内积缩放并舍入到较小的群 $\mathbf{Z}_p$. 具体而言, 舍入把 $\mathbf{Z}_q$ 中的元素划分为 p 个相邻区间, 并将每个 $\mathbf{Z}_q$ 元素映射到其在 $\mathbf{Z}_p$ 中对应的索引. 必须满足 $p < q$, 否则不会引入误差. 舍入后的内积 $\left\lfloor \frac{p}{q} \langle a, s \rangle \right\rfloor$ 记为

$$\lfloor \langle a, s \rangle \rfloor_p = \left\lfloor \frac{p}{q} \langle a, s \rangle \right\rfloor \in \mathbf{Z}_p.$$

此处 $\mathbf{Z}_q$ (相应地, $\mathbf{Z}_p$) 用 $[0, q[$ (相应地, $[0, p[$) 表示. 注意, 当 $p \mid q$ 时, $\mathbf{Z}_q$ 上均匀分布经舍入后在 $\mathbf{Z}_p$ 上仍为均匀分布. 逐系数应用即可将舍入操作扩展到向量或多项式. 与误差学习的环变体和模变体类似, 也可定义 LWR 的环版本与模版本.

舍入学习问题的判定版本定义如下.

定义 1 (判定型舍入学习, 基于文献 [BPR12, 定义 3.1]). 设 $n \geq 1$, 模数 $q > p \geq 2$ 均为整数, $\mathbf{R}_q = \mathbf{Z}_q[x]/f(x)$ 如上定义.

- (**Plain-LWR**): 对向量 $s \in \mathbf{Z}_q^n$, 定义 LWR 分布 L_s 为 $\mathbf{Z}_q^n \times \mathbf{Z}_p$ 上的如下分布: 均匀随机选取向量 $a \leftarrow \mathbf{Z}_q^n$, 输出 $(a, b = \lfloor \langle a, s \rangle \rfloor_p)$.
- (**Ring-LWR**): 对多项式 $s \in \mathbf{R}_q$, 定义环 LWR 分布 L_s 为 $\mathbf{R}_q \times \mathbf{R}_p$ 上的如下分布: 均匀随机选取 $a \leftarrow \mathbf{R}_q$, 输出 $(a, b = \lfloor a \cdot s \rfloor_p)$.
- (**Module-LWR**): 固定模秩 $d > 1$. 对 $s \in \mathbf{R}_q^d$, 定义模 LWR 分布 L_s 为 $\mathbf{R}_q^d \times \mathbf{R}_p$ 上的如下分布: 均匀随机选取 $a \leftarrow \mathbf{R}_q^d$, 输出 $(a, \lfloor \langle a, s \rangle \rfloor_p)$.

判定型 (环/模) LWR_{n,q,p} 问题要求以关于 n 不可忽略的优势, 区分多项式数量的独立样本 $(a_i, b_i) \leftarrow L_s$ 与分别从 $\mathbf{Z}_q^n \times \mathbf{Z}_p$ 、 $\mathbf{R}_q \times \mathbf{R}_p$ 或 $\mathbf{R}_q^d \times \mathbf{R}_p$ 中均匀、独立抽取的同等数量样本.

3.2.3 舍入学习的安全性

论证 LWR 安全性的总体思路, 是证明其安全性与求解判定型 LWE (即与随机量 $\mathbf{u}$ 区分) 的安全性在统计意义上接近 [AKPW13]:

$$(a, \lfloor \langle a, s \rangle \rfloor_p) \stackrel{\text{stat}}{\approx} (a, \langle a, s \rangle + e) \stackrel{\text{comp}}{\approx} (a, \mathbf{u}).$$

文献 [BPR12] 最早研究了 LWR 的安全性, 并给出如下结果.

定理 1 (到 LWE 的归约 [BPR12, 定理 3.2]). 设 χ_n 为整数环 $\mathbf{Z}$ 上任意可高效采样的概率分布族, 以安全参数 n 为索引, 且为 $\beta(n)$ -有界 (即 $\Pr_{x \leftarrow \chi} [|x| > \beta] \leq \text{negl}(n)$). 令 $q \geq p \cdot \beta \cdot n^{\omega(1)}$. 则对于秘密 $s \in \mathbf{Z}_q^n$ 上的任意分布, 求解判定型 LWR_{n,q,p} 至少与在相同秘密分布下求解判定型 LWE_{n,q,\chi} 一样困难.

定理 1 要求模数 $q \geq p \cdot \beta \cdot n^{\omega(1)}$ 为超多项式量级. 文献 [AKPW13] 对该结果进行了改进, 给出适用于更广参数范围的安全归约: 所用模数 q 只需为多项式量级, 而非超多项式量级 [AKPW13, 定理 4.1 与推论 4.2]. 具体而言, 若参数为 n', m, q, β 的 LWE 是困难的, 则只要

$$n \geq \frac{\log(q)}{\log(2\gamma)} \cdot n' \quad \text{且} \quad q \geq \gamma(nm\beta p)$$

对某个参数 $\gamma \geq 1$ 成立, 作者便证明参数为 n, m (样本数)、 q, p 的 LWR 是困难的. 为使归约成立, 模数 q 必须远大于 LWR 样本数 m . 因此, 当攻击者可自行选择 LWR 样本数 (即样本数并非预先固定) 时, 模数 q 又会变为超多项式量级. 模数 q 必须为素数; 否则, 上述不等式须对 q 的最大素因子 $q_{\max}$ 成立. 此外, 该结果要求 q 与 $q/q_{\max}$ 互素, 这意味着当模数 q 为二的幂时该结果不成立.

文献 [BGM⁺16] 的结果消除了这一限制. 首先, 模数 q 的界改进为

$$q \geq 2m\beta p$$

其次, 该结果成立时无需对 q 施加任何数论限制; 特别是, 当模数 q 为二的幂时也成立 [BGM⁺16, 定理 1 与定理 3].

定理 2 (LWR 的单向性 [BGM⁺16, 定理 1]). 设 p, q, n, m, β 为满足 $q > 2p\beta$ 的整数. 对于任意算法 Learn ,

$$\Pr_{A,s,e} [\text{Learn}(A, \lfloor As + e \rfloor_p) = s] \geq \frac{\Pr_{A,s} [\text{Learn}(A, \lfloor As \rfloor_p) = s]^2}{(1 + 2p\beta/q)^m},$$

其中 $A \leftarrow \mathbf{Z}_q^{m \times n}$; 噪声 e 的全部 m 个坐标相互独立, 并且每个坐标均为 β -有界且平衡; s 从支撑集位于 $\mathbf{Z}_q^{n*}$ 上的任意分布中选取.

该结果证明了样本数 m 有界时 LWR 的困难性. 较近期的文献 [NR23] 研究了无界情形, 并发现通过从 LWE 到 LWR 的“逐点”归约来证明安全性存在障碍. 该工作的主要结果及相关猜想要求模数 q 为素数 [NR23, 定理 1 与猜想 1], 而我们的设计并非如此. 需要强调的是, 文献 [NR23] 的结果仅表明证明无界样本情形的安全性需要新思路, 并不意味着 LWR 容易求解.

3.3 作为结构化模的环

在 Module-LWR 场景中, 单个样本形如 $[\langle \mathbf{A}, \mathbf{s} \rangle]_p$, 其中 $\mathbf{A}, \mathbf{s} \in \mathbf{R}_{q,n}^k$, k 为模秩. 样本的选取方式使 $\mathbf{A}$ 均匀分布, 而 $\mathbf{s}$ 与 $\mathbf{e}$ 从“小”分布中生成. 若选取 m 个样本, 并假定环结构与模结构无法被利用, 则其安全级别等价于矩阵大小为 $mn \times kn$ 的 Plain-LWR 问题. 取 $m = k$ 即得到典型 KEM 的情形, 此时舍入学习问题由方阵定义. 为简化叙述, 以下均假定 $m = k$.

Module-LWR 的主要优点是可以任意选择 k . 例如, Saber 选择 $k \in \{2, 3, 4\}$. 相比之下, 要达到相同安全级别, 基于分圆环的 Ring-LWR 需从 $R_{q,kn}$ 中采样, 因此必须把 k 限制为二的幂, 无法选择 $k = 3$. 另一方面, Ring-LWR 的优点是单个样本元素 $\mathbf{A}$ 仅含 kn 个不同的 $\mathbf{Z}_q$ 元素 (即一个 $\mathbf{A} \in R_{q,kn}$ 元素), 而 Module-LWR 则含 k^2n 个 (即 $\mathbf{R}_{q,n}^k$ 中元素的 k 个样本). Module-LWR 的二次增长使其效率低于 Ring-LWR, 但 k 的更高灵活性使具体参数集更容易实例化.

众所周知, Ring-LWR 可对应于 (或视为) 一个结构化 Module-LWR 实例. 对于每个满足 $k \mid n$ 的二的幂 k , 存在同构

$$\mathbf{R}_{q,n} = \mathbf{Z}_q[x]/(x^n + 1) \cong \left(\mathbf{Z}_q[y]/(y^{\frac{n}{k}} + 1) \right) [x]/(x^k - y) \quad (2)$$

该同构映射为

$$s(x) = \sum_{i=0}^n s_i x^i \mapsto \sum_{j=0}^{k-1} s_j(y) \cdot x^j, \quad \text{其中} \quad s_j(y) = \sum_{i=0}^{n/k-1} s_{ki+j} \cdot y^i.$$

这只是将多项式 s 划分为 k 个部分 $s_j(y)$, 每个 $s_j(y)$ 包含次数模 k 等于 j 的系数. 例如, 当 $k = 2$ 时, $s_0(y)$ 包含偶数次项的系数, $s_1(y)$ 包含奇数次项的系数. 取两个元素 $\mathbf{A}, \mathbf{s} \in \mathbf{R}_{q,n}$, 在此同构下有

$$\mathbf{A} \cdot \mathbf{s} = \begin{bmatrix} a_0(y) & ya_1(y) \\ a_1(y) & a_0(y) \end{bmatrix} \begin{bmatrix} s_0(y) \\ s_1(y) \end{bmatrix},$$

其中每项运算均在环 $\mathbf{Z}_q[y]/(y^{\frac{n}{2}} + 1)$ 中进行; 特别地, ya_1 对应于 $a_1(y)$ 系数的负循环移位. 这是因为

$$\begin{aligned} \mathbf{A} \cdot \mathbf{s} &= (a_0(y) + a_1(y)x) \cdot (s_0(y) + s_1(y)x) \\ &= a_0s_0(y) + (a_0s_1(y) + a_1s_0(y))x + a_1s_1(y)x^2 \\ &= a_0s_0(y) + ya_1s_1(y) + (a_0s_1(y) + a_1s_0(y))x \bmod (x^2 - y). \end{aligned}$$

当 $k = 4$ 时, 则得到

$$\mathbf{A} \cdot \mathbf{s} = \begin{bmatrix} a_0(y) & y \cdot a_3(y) & y \cdot a_2(y) & y \cdot a_1(y) \\ a_1(y) & a_0(y) & y \cdot a_3(y) & y \cdot a_2(y) \\ a_2(y) & a_1(y) & a_0(y) & y \cdot a_3(y) \\ a_3(y) & a_2(y) & a_1(y) & a_0(y) \end{bmatrix} \begin{bmatrix} s_0(y) \\ s_1(y) \\ s_2(y) \\ s_3(y) \end{bmatrix}, \quad (3)$$

此时运算在 $\mathbf{Z}_q[y]/(y^{\frac{n}{4}} + 1)$ 中进行. 更一般地, 容易看出 $\mathbf{A} \cdot \mathbf{s}$ 对应于一个 $k \times k$ 循环矩阵与向量的乘法, 其中矩阵右上三角部分乘以 y . 作为比较, 当 $k = 4$ 时, Module-LWR 矩阵乘法的结构为

$$\begin{bmatrix} a_{0,0}(y) & a_{0,1}(y) & a_{0,2}(y) & a_{0,3}(y) \\ a_{1,0}(y) & a_{1,1}(y) & a_{1,2}(y) & a_{1,3}(y) \\ a_{2,0}(y) & a_{2,1}(y) & a_{2,2}(y) & a_{2,3}(y) \\ a_{3,0}(y) & a_{3,1}(y) & a_{3,2}(y) & a_{3,3}(y) \end{bmatrix} \begin{bmatrix} s_0(y) \\ s_1(y) \\ s_2(y) \\ s_3(y) \end{bmatrix},$$

其中算术同样在环 $\mathbf{Z}_q[y]/(y^{\frac{n}{4}} + 1)$ 中进行.

综上, 对于任意 $k \mid n$, 均可将分圆 Ring-LWR 样本视为秩为 k 的结构化 Module-LWR 样本, 其中

$$a_{i,j}(y) = \begin{cases} \text{均匀随机}, & j = 0, \\ y \cdot a_{k+i-j,0}(y), & 0 \leq i < j < k, \\ a_{i-j,0}(y), & 0 < j \leq i < k. \end{cases} \quad (4)$$

上述重新解释纯属语法层面, 不影响安全性. 在式 (2) 的环同构下, 分圆 Ring-LWR 样本与式 (4) 所述秩为 k 的结构化 Module-LWR 实例构成双射. 将结构化 Module-LWR 的任意区分器或秘密恢复算法与逆同构复合, 便立即得到 Ring-LWR 的相应算法, 其优势完全相同、运行时间也基本相同. 因此, 将分圆 Ring-LWR 样本视为秩为 k 的结构化 Module-LWR 样本不会改变底层困难性假设或具体估计; 这只是同一分布的不同表示 (相同论证也适用于 LWE).

3.4 根式环舍入学习 (Radical Ring-LWR)

第 3.3 节的观察是以不同方式解释并定义 Ring-LWR 与 Module-LWR 类方案的新环和新参数的关键.

我们不使用二的幂分圆多项式 $f(x) = x^n + 1$, 而是对二的幂 n 和正整数 k 选择多项式 $f(x) = (x^k - 2)^n + 1$. 可以证明 f 不可约 [BRVW26, 引理 1], 因此 $\mathbf{Q}[x]/(f(x))$ 是数域, 从而可针对该 $f(x)$ 定义 Ring-LWR 型问题. 在归藏实例中, 舍入在迁移后的单项式基中逐系数进行; 该基即根式环单项式基 $\{y^i x^j\}$ 在数域同构下的像. 安全性章节通过界定相应的基失真, 处理它与通常典范嵌入视角之间的联系. 乍看之下, 改变多项式似乎会丧失二的幂分圆多项式所带来的优势; 然而, 存在如下简单环同构:

$$S_{q,n,k} := \mathbf{Z}_q[x]/((x^k - 2)^n + 1) \cong (\mathbf{Z}_q[y]/(y^n + 1)[x])/(x^k - y - 2), \quad (5)$$

$$x^k \mapsto y + 2.$$

虽然 $S_{q,n,k}$ 本身不是分圆环, 但它包含一个大的二的幂分圆子环, 仍可在其中采用 NTT 等算术优化. 因此, $f(x)$ 所定义的域是第 $2n$ 个分圆域的 k 次扩张; 扩张次数 k 的作用类似于模秩, 使底层格问题的困难性比必须取二的幂次数 n 的分圆环更易调节. 由于多项式 $x^k - y - 2$ 定义了第 $2n$ 个分圆域的 k 次根式扩张, 在舍入学习场景中将此类环称为根式 Ring-LWR (RR-LWR).

) .

注意, 即便不局限于二的幂 n , 而考虑更一般的第 m 个分圆域, 也仍无法构造既含大型二的幂分圆子域、又具有给定次数的分圆域. 事实上, 第 m 个分圆域的次数为 Euler 函数 $\varphi(m)$, 因此可能的扩张次数受限于此类整数; 若出于效率需要还要求大型二的幂子域, 限制会更严格. 例如, 不存在满足 $\varphi(m) = 3$ 或 $\varphi(m) = 5$ 的整数 m , 因而不存在对应于二的幂分圆域 3 次或 5 次扩张的分圆域. 采用上述 $f(x)$ 后, 只需令 $k = 3$ 或 $k = 5$ 即可直接得到此类域.

与通用模结构相比, 上述构造通过施加环结构降低了相对于 Module-LWR 的计算复杂度和随机性需求. 也就是说, 矩阵 $\mathbf{A}$ 由分圆环中的 k 个而非 k^2 个均匀随机元素组成. 更具体地, 对元素 $\mathbf{A}, \mathbf{s} \in S_{q,n,k}$, 写作 $\mathbf{A} = \sum_{i=0}^{k-1} a_i(y)x^i$ 且 $\mathbf{s} = \sum_{i=0}^{k-1} s_i(y)x^i$, 则多项式乘法 $\mathbf{A} \cdot \mathbf{s}$ 对应于矩阵乘法

$$\begin{bmatrix} a_0(y) & (y+2) \cdot a_{k-1}(y) & (y+2) \cdot a_{k-2}(y) & \cdots & (y+2) \cdot a_1(y) \\ a_1(y) & a_0(y) & (y+2) \cdot a_{k-1}(y) & \cdots & (y+2) \cdot a_2(y) \\ \vdots & \vdots & \vdots & & \vdots \\ a_{k-1}(y) & a_{k-2}(y) & a_{k-3}(y) & \cdots & a_0(y) \end{bmatrix} \begin{bmatrix} s_0(y) \\ s_1(y) \\ s_2(y) \\ \vdots \\ s_{k-1}(y) \end{bmatrix}. \quad (6)$$

该矩阵乘法的全部运算均在分圆环 $\mathbf{Z}_q[y]/(y^n + 1)$ 中进行. 与式 (3) 的 Ring-LWR 矩阵结构相比, 可以看出它实质上只是将乘法因子 y 替换为 $y + 2$.

在不同于分圆环的数环上实例化 Ring-LWR 时必须谨慎. 安全性章节将本实例形式化为根式数域上的迁移基 Ring-LWR 变体, 并在具体估计中计入所用的基变换.

4 算法描述

本节描述构造 软猬 (Mithril) 密钥封装机制 (KEM) 所需的全部组件. 软猬 (Mithril) .KEM 通过带隐式拒绝的 Fujisaki-Okamoto 变换由 软猬 (Mithril) .PKE 导出. 软猬 (Mithril) .PKE 包含 KeyGen、Enc 和 Dec 三个算法, 软猬 (Mithril) .KEM 包含 KeyGen、Encaps 和 Decaps 三个算法, 完整规范见算法 1.

参数集与安全级别. 按照提交要求, 算法采用参数化设计以支持多个安全级别. 规定三个参数集, 分别以 128、256 和 512 比特经典安全强度为目标; 估计相应抗量子安全强度分别不低于 80、128 和 256 比特.

各参数集实例化相同算法结构, 仅在决定底层格维数和模数大小的参数选择上有所不同. 该统一设计使所有安全级别可使用一致的实现框架, 同时通过参数选择细粒度调节安全性. 完整参数列表见表 2.

	软猬 (Mithril) -128	软猬 (Mithril) -256	软猬 (Mithril) -512
n	128	128	128
k	5	9	18
ℓ	1	2	4
ϵ_q	13	13	14
ϵ_p	11	11	12
ϵ_t	3	4	6
D_A	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_{\epsilon_q}$	$\mathcal{U}_{\epsilon_q}$
D_s	$\mathcal{U}_2$	$\mathcal{U}_2$	$\mathcal{U}_2$
KEM 私钥 (字节)	1 136	2 000	4 224
KEM 公钥 (字节)	944	1 648	3 520
KEM 密文 (字节)	928	1 712	3 840
共享秘密 (字节)	16	32	64
失败概率 ($\log_2$)	-603	-278	-569
Core-SVP (经典, 单位: 比特)	129	257	524
Core-SVP (量子, 单位: 比特)	117	233	476

Table 2: 软猬 (Mithril) 三个参数集概览.

Algorithm 1 软猬 (Mithril) 公钥加密 (PKE) 和密钥封装机制 (KEM) 算法.

$(pk, sk) \leftarrow \text{软猬 (Mithril) .PKE.KeyGen}()$

- 1: $\text{seed}_A, \text{seed}_s \leftarrow \{0, 1\}^{512}$
- 2: $A \leftarrow D_A(S_{q,n,k}; \text{seed}_A)$
- 3: $s \leftarrow D_s(S_{q,n,k}; \text{seed}_s)$
- 4: $b = \lfloor \frac{p}{q} As \rfloor \in S_{p,n,k}$
- 5: **返回** $(pk = (\text{seed}_A, b), sk = s)$

$c \leftarrow \text{软猬 (Mithril) .PKE.Enc}(pk = (\text{seed}_A, b), m)$

- 1: $\text{seed}_{s'} \leftarrow \{0, 1\}^{512}$
- 2: $s' \leftarrow D_s(S_{q,n,k}; \text{seed}_{s'})$
- 3: $A \leftarrow D_A(S_{q,n,k}; \text{seed}_A)$
- 4: $b' = \lfloor \frac{p}{q} \cdot (As') \rfloor \in S_{p,n,k}$
- 5: $v' = (bs')^{[k-\ell:k-1]} + \frac{q}{2 \cdot p} \in \mathbf{R}_{p,n}^\ell$ ▷ 选择环元素的 ℓ 个最高次系数
- 6: $c_m = \lfloor \frac{t}{p} \cdot ((v' + \frac{p}{2} \cdot m) \bmod p) \rfloor \in \mathbf{R}_{t,n}^\ell$
- 7: **返回** $c = (c_m, b')$

$m' \leftarrow \text{软猬 (Mithril) .PKE.Dec}(sk = s, c = (c_m, b'))$

- 1: $v = (b's)^{[k-\ell:k-1]} \in \mathbf{R}_{p,n}^\ell$ ▷ 选择环元素的 ℓ 个最高次系数
- 2: $m' = \lfloor \frac{2}{p} \cdot ((v - \frac{p}{t} \cdot c_m + (\frac{q}{2 \cdot p} - \frac{p}{2 \cdot t})) \bmod p) \rfloor \in \mathbf{R}_{2,n}^\ell$
- 3: **返回** m'

$(pk, sk) \leftarrow \text{软猬 (Mithril) .KEM.KeyGen}()$

- 1: $(pk_{\text{pke}}, sk_{\text{pke}}) \leftarrow \text{软猬 (Mithril) .PKE.KeyGen}$
- 2: $z \leftarrow \{0, 1\}^{\ell n}$ ▷ 秘密备用值
- 3: $hpk \leftarrow \mathcal{F}_{H_F}(pk_{\text{pke}}, \lambda/8)$ ▷ 公钥哈希值
- 4: **返回** $(pk = pk_{\text{pke}}, sk = (sk_{\text{pke}}, pk_{\text{pke}}, hpk, z))$

$(c, ss) \leftarrow \text{软猬 (Mithril) .KEM.Encaps}(pk)$

- 1: $m \leftarrow \{0, 1\}^{\ell n}$ ▷ 编码/映射为 $\mathbf{R}_{2,n}^\ell$ 中的消息
- 2: $hpk \leftarrow \mathcal{F}_{H_F}(pk, \lambda/8)$
- 3: $(ss, \text{seed}_{s'}) \leftarrow \mathcal{G}_{H_G}(hpk, m, \lambda/8 + 64)$ ▷ 导出密钥材料和加密种子
- 4: $c \leftarrow \text{软猬 (Mithril) .PKE.Enc}(pk, m; \text{seed}_{s'})$ ▷ 使用显式随机性 $\text{seed}_{s'}$ 加密
- 5: **返回** (c, ss)

$ss \leftarrow \text{软猬 (Mithril) .KEM.Decaps}(sk, c)$

- 1: $m' \leftarrow \text{软猬 (Mithril) .PKE.Dec}(sk_{\text{pke}}, c)$
 - 2: $(ss', \text{seed}'_{s'}) \leftarrow \mathcal{G}_{H_G}(hpk, m', \lambda/8 + 64)$ ▷ 重新计算密钥材料和加密种子
 - 3: $c' \leftarrow \text{软猬 (Mithril) .PKE.Enc}(pk, m'; \text{seed}'_{s'})$ ▷ 确定性重加密
 - 4: $\bar{ss} \leftarrow \mathcal{H}_{H_H}(c, z, \lambda/8)$
 - 5: **if** $c = c'$ **then** ▷ 以常数时间实现
 - 6: $ss \leftarrow ss'$
 - 7: **else**
 - 8: $ss \leftarrow \bar{ss}$
 - 9: **end if**
 - 10: **返回** ss
-

环的选择. 本方案在如下形式的根式环上运算：

$$\mathbf{S}_{q,n,k} = (\mathbb{Z}_q[y]/(y^n + 1))[x]/(x^k - y - 2), \quad (7)$$

其中 q 和 n 为二的幂, k 为任意正整数. 该环与 玄色 (Octarine) 所用环相同, 并与 $\mathbf{S}_{q,n,k} \cong \mathbb{Z}_q[x]/((x^k - 2)^n + 1)$ 存在直接同构 [BRVW26, 第 4 节]. 方案在根式环单项式基中使用逐系数舍入的 RR-LWR; 在该同构下, 这对应关联数域上的迁移基 Ring-LWR 实例.

环 $\mathbf{S}_{q,n,k}$ 是 $2n$ 次分圆基环的 k 次根式扩张, 因此可按需使用 NTT 或其他多项式乘法例程等标准实现技术. 由 k 个基环分量构成的紧凑表示, 取代了秩为 k 的模构造中二次规模的公有矩阵表示.

所有参数集统一取环模数 $q = 2^{13}$ 和基环次数 $n = 128$, 改变根式扩张次数 k , 进而改变总代数次数 nk , 以获得所需安全级别. 虽然该环兼容 NTT 型实现, 但 NTT 并非必需. RR-LWR 舍入步骤将系数从模数 q 映射到舍入模数 p . 缩放因子为 q/p , 产生的确定性量化误差绝对值不超过 $q/(2p)$. 选择 p 使 s 和 s' 均服从区间 $[-\eta, \eta - 1]$ 上的相同均匀分布, 其中 $\eta = q/(2p) = 2$. 因而 s 和 s' 的每个系数恰可用 $\log_2(2\eta) = 2$ 比特表示.

消息空间. 在 Module-LWE 或 Module-LWR 场景中, 消息嵌入为 $\mathbf{R}_{2,n}$ 中的元素. 该元素由 n 个单比特系数构成, 因而自然编码 n 比特消息. 对 Ring-LWR, 消息则编码为 $\mathbf{S}_{2,n,k}$ 中的元素, 可编码 nk 比特消息. 由于无需完整的 kn 比特消息空间, 本文将消息嵌入一个仅有 ℓ 个系数非零、其余 $k - \ell$ 个系数为 0 的子空间. 因 $n = 128$, 相应参数集取 $\ell = 1, 2, 4$, 以保证 $n\ell = \lambda$.

为优化方案性能, 选择环元素的 ℓ 个最高次系数. 这可最小化模 $x^k - (y + 2)$ 约减次数, 从而节省部分乘法; 实现中可省略所有与零系数相乘的运算. 更准确地, 记号 $\mathbf{g}^{[k-\ell:k-1]}$ 表示多项式 $\mathbf{g} = \sum_{i=0}^{k-1} \mathbf{g}_i(y)x^i \in \mathbf{S}_{2,n,k}$, 其中 $\mathbf{g}_0(y), \dots, \mathbf{g}_{k-\ell-1}(y) = 0$. 将其解释为向量 $\mathbf{v} \in \mathbf{R}_{2,n}^\ell$, 满足 $\mathbf{v}_i = \mathbf{g}_{k-1-i}$. 因此, 打包的消息和密文分量按 x 次数降序序列化: 第一个 $\mathbf{R}_{2,n}$ 分量为 $\mathbf{g}_{k-1}(y)$, 随后依次为 $\mathbf{g}_{k-2}(y), \dots, \mathbf{g}_{k-\ell}(y)$.

哈希. 软猬 (Mithril) 需要哈希函数, 更准确地说是可扩展输出函数 (XOF), 以实例化随机预言机并为密钥材料和临时秘密生成伪随机比特.

首先, 使用函数 $\mathcal{F}$ 、 $\mathcal{G}$ 和 $\mathcal{H}$ 实例化随机预言机. 按惯例, 哈希函数调用中包含输出的字节长度. 在调用底层 XOF 前, 实现以显式域编码实现这些功能. 编码输入为下列头字段的串接, 后接 nf 个变长字段 $|X_1| \| X_1 \| \dots \| |X_{\text{nf}}| \| X_{\text{nf}}$, 其中每个字段长度 $|X_i|$ 编码为 8 字节小端整数; 所有头字段中的整数也采用小端编码.

字段	值	字节数
协议前缀	ARCANE-Mithril-v1	17
标签长度	$ \text{label} $	1
标签	label	$ \text{label} $
安全级别	λ	2
模秩	k	1
输出长度	outlen	4
字段数	nf	1

等价地, 标签长度、模秩和字段数各编码为一个字节; 安全级别编码为 16 位小端整数, 输出长度编码为 32 位小端整数, 每个变长字段的长度编码为 64 位小端整数. 此处 `outlen` 为请求的输出字节长度. 三种 KEM 随机预言机功能所用标签分别为 `H_F`、`H_G` 和 `H_H`. 具体为:

- 公钥摘要: $hpk \leftarrow \mathcal{F}_{H_F}(pk_{pke}, \lambda/8)$, 生成可作为私钥一部分存储的公钥摘要.
- 确定性加密: $(ss, seed_{s'}) \leftarrow \mathcal{G}_{H_G}(hpk, m, \lambda/8 + 64)$, 基于公钥和消息生成共享秘密及公钥加密所用的确定性随机币.
- 共享秘密生成: $\bar{s}s \leftarrow \mathcal{H}_{H_H}(c, z, \lambda/8)$, 生成密文被隐式拒绝时使用的共享秘密.

本规范所提交的参考实现和优化实现均以 `pseudoXOF` 实例化所有哈希函数. 为与现有最优实现进行公平性能比较, 另提供基于 SHAKE 的基准配置, 其中相关哈希函数以 `SHAKE256` 实例化. 由于 `SHAKE256` 的容量为 512 比特, 至多提供 256 比特经典安全性, 故该 SHAKE 配置不用于 512 比特参数集.

其次, 还需要 XOF 从种子扩展私钥和公钥材料时生成伪随机比特; 这用于生成 s 、 A 和 s' . 对公开数据 A 使用 XOF_A , 对秘密数据使用 XOF_S . 这些大容量 XOF 输入采用紧凑的长度定界编码:

`AMX1 || id || IntegerToBytes( $\lambda$ , 2) || IntegerToBytes( $k$ , 1) || IntegerToBytes(outlen, 2)`
`|| IntegerToBytes( $|s|$ , 1) ||  $s$  || IntegerToBytes( $i$ , 1) || IntegerToBytes(lane, 1).`

其中, `XOF_A` 取 `id = 1`, `XOF_S` 取 `id = 2`; s 为传入 XOF 调用的种子, i 为多项式或流索引, `lane` 为通道索引. 对 XOF_A , 字段依次为种子、单字节多项式索引和单字节通道索引. 对 XOF_S , 字段为种子、索引字节 0 和通道字节 0; 一条流生成全部 k 个秘密多项式. 种子长度编码为一个字节, 输出长度编码为 16 位小端整数; 这些上界覆盖所有已实现参数集. 实现中, XOF_A 和 XOF_S 均以 `pseudoXOF` 实例化. 为与现有最优实现进行公平性能比较, 优化实现使用 `SHAKE128` 实例化 XOF_A , 使用 `SHAKE256` 实例化 XOF_S .

采样分布. 方案在不同均匀分布中采样秘密值和公开值. 除非另有明确说明, 比特和字节寻址均使用小端序. 对较大系数, 例如 $\log_2(q) = 13$ 比特系数, 使用域分离独立生成 k 个各含 n 个系数的多项式, 以支持环元素多项式的并行生成. 对系数较小的环元素, 例如系数为 2 比特的 s , 每个多项式仅需 $2 \cdot 128/8 = 32$ 字节, 故在此类情形从单一字节流生成完整环元素.

- $D_A(S_{q,n,k}, seed_A)$: 使用可扩展输出函数将种子 $seed_A$ 确定性扩展为 $S_{q,n,k}$ 上均匀随机元素. 生成 A 共需 k 个各含 n 个系数的多项式, 每个系数为 $\epsilon_q = 13$ 比特. 写作 $A = \sum_{i=0}^{k-1} A_i(y)$ 及 $A_i(y) = \sum_{j=0}^{n-1} \alpha_{i,j} y^j$. 采样分两步进行: 先由 $seed_A$ 生成字节流; 随后对每个多项式使用索引 i 对 $seed_A$ 做域分离, 即从种子 $seed_A$ 生成 $13 \cdot 128/8 = 208$ 字节流 $B_i = XOF_A(seed_A, i, 0)$, 并令 $\alpha_{i,j} = B_i[13j : 13(j+1)]$. 其中 $B_i[a : b]$ 表示从比特索引 a 至 b 的区间, 包含 a 而不包含 b .
- $D_s(S_{q,n,k}, seed_s)$: 采样一个环元素, 其 nk 个整数系数独立且均匀地取自 $[-\eta, \eta - 1]$. 生成 s 共需 k 个各含 n 个系数的多项式, 每个系数为 $\theta = \log_2(2\eta)$ 比特. 写作 $s = \sum_{i=0}^{k-1} s_i(y)$ 及 $s_i(y) = \sum_{j=0}^{n-1} \sigma_{i,j} y^j$. 首先, 种子 $seed_s$ 经 XOF 函数 XOF_S 输出字节流 $B = XOF_S(seed_s, 0, 0)$; 随后令 $\sigma_{i,j} = B[\theta(in+j) : \theta(in+j+1)]$. 其中 $B[a : b]$ 表示从比特索引 a 至 b 的区间, 包含 a 而不包含 b . 对种子 $seed_{s'}$ 使用相同采样例程生成 s' .

舍入与除法. 按惯例, 模二的幂 r 的整数用区间 $[-r/2, r/2-1]$ 中的代表元表示. 对任意二的幂因子 $d \mid r$ 及任意 $x \in [-r/2, r/2-1]$, 可唯一写成 $x = x_0 + x_1 d \bmod r$, 其中 $x_0 \in [-d/2, d/2-1]$, $x_1 \in [-r/(2d), r/(2d)-1]$. 可计算为 $x_0 = x \bmod d$ 和 $x_1 = (x - x_0)/d \bmod r/d$. $\mathbf{b} = \lfloor \frac{\alpha}{\beta} \mathbf{w} \rfloor$ 返回舍入后的环元素, 对应于 $\mathbf{b} = (\mathbf{w} + \frac{\beta}{2\alpha}) \gg (\log_2 \beta - \log_2 \alpha)$, 其中 $\gg$ 表示右移, 且每个操作均逐系数执行.

通过常数因子平移舍入误差, 使误差均值为零并简化安全性分析. 详见 D'Anvers 的博士论文 [D'VV21, 第 3.2 节].

密文压缩. 为减小密文大小, 应用有损函数 $\lfloor \frac{t}{p} \cdot \rfloor$, 其中 t 的取值依参数集而定. 此举虽能减小密文大小, 却会提高失败概率. 失败概率的详细分析见第 5.2.1 节. 公钥不进行压缩.

密钥、密文和消息编码. 密钥、密文和消息均以小端序打包字节数组序列化. 注意, 说明中将元素表示为有符号整数, 而序列化时打包和解包的是无符号整数. 字节串存储无符号代表元, 解包时将其映射回中心有符号代表元, 从而避免符号扩展带来的实现问题. 对 b 比特系数序列 $(t_i)_{0 \leq i < N}$, 定义

$$\text{Enc}_b(a) = (2^{b-1} - 1 - a) \bmod 2^b \in \{0, \dots, 2^b - 1\}$$

and

$$\text{Pack}((t_i)_{0 \leq i < N}, b) = \text{Enc}_b(t_0) \parallel \dots \parallel \text{Enc}_b(t_{N-1})$$

为其序列化, 其中串接方式规定 t_0 存于最低有效位置, 且 $\text{Enc}_b(t_0) \bmod 2$ 对应比特 0. 解包使用逆映射 $\text{Dec}_b(u) = 2^{b-1} - 1 - u$, 返回区间 $[-2^{b-1}, 2^{b-1} - 1]$ 中的系数.

对系数为 b 比特的环元素 $\mathbf{w} = \sum_{i=0}^{k-1} (\sum_{j=0}^{n-1} w_{i,j} y^j) x^i \in \mathbf{S}_{q,n,k}$, 定义其序列化为

$$\text{Pack}(\mathbf{w}, b) = \text{Pack}((w_{0,j})_{0 \leq j < n}, b) \parallel \dots \parallel \text{Pack}((w_{k-1,j})_{0 \leq j < n}, b),$$

其中同样将 $\text{Pack}((w_{0,j})_{0 \leq j < n}, b)$ 存于最低有效地址.

- PKE 公钥 pk_{pke} 的序列化定义为

$$\text{pkEncode}(\text{seed}_A, \mathbf{b}) = \text{seed}_A \parallel \text{Pack}(\mathbf{b}, \epsilon_p).$$

对应的反序列化函数为 $\text{pkDecode}(pk_{\text{pke}})$. PKE 公钥总大小为 $64 + \epsilon_p kn/8$ 字节.

- PKE 私钥 sk_{pke} 的序列化定义为

$$\text{skEncode}(\mathbf{s}) = \text{Pack}(\mathbf{s}, \log_2(2\eta)).$$

对应的反序列化函数为 $\text{skDecode}(sk_{\text{pke}})$. PKE 私钥总大小为 $\log_2(2\eta)kn/8$ 字节.

- PKE 密文 ct_{pke} 的序列化定义为

$$\text{ctEncode}(c_m, \mathbf{b}') = \text{Pack}((\zeta_{0,i})_{0 \leq i < n}, \epsilon_t) \parallel \dots \parallel \text{Pack}((\zeta_{\ell-1,i})_{0 \leq i < n}, \epsilon_t) \parallel \text{Pack}(\mathbf{b}', \epsilon_p),$$

其中 $c_m = (\zeta_0, \dots, \zeta_{\ell-1}) \in \mathbf{R}_{t,n}^\ell$. 对应的反序列化函数为 $\text{ctDecode}(ct_{\text{pke}})$. PKE 密文总大小为 $\epsilon_t \ell n/8 + \epsilon_p kn/8$ 字节.

- PKE 消息 m 的序列化定义为

$$\text{mEncode}(m) = \text{Pack}((\mu_{0,i})_{0 \leq i < n}, 1) \parallel \dots \parallel \text{Pack}((\mu_{\ell-1,i})_{0 \leq i < n}, 1),$$

其中 $m = (\mu_0, \dots, \mu_{\ell-1}) \in \mathbf{R}_{2,n}^\ell$. 因 $b = 1$, 解码返回 $\{-1, 0\}$ 中的中心代表元, 并将其模 2 解释为 $\mathbf{R}_{2,n}$ 中元素. 对应的反序列化函数为 $\text{mDecode}(m)$. PKE 消息总大小为 $\ell n/8$ 字节.

- KEM 公钥的序列化与 PKE 公钥相同.
- KEM 私钥 sk_{KEM} 的序列化定义为 $sk_{\text{pke}} \parallel pk_{\text{pke}} \parallel hpk \parallel z$.
- KEM 密文的序列化与 PKE 密文相同: $ct_{\text{KEM}} = ct_{\text{pke}}$.
- KEM 共享秘密作为 ss 单独序列化; 它是 Encaps 的输出, 不作为 KEM 密文的一部分传输.

Fujisaki–Okamoto 变换. 本构造将带隐式拒绝的 Hofheinz–Hövelmanns–Kiltz 模块化 Fujisaki–Okamoto 变换 U_m^ℓ [HHK17] 应用于 软猬 (Mithril) .PKE 的确定性重加密形式. 在该变换中, 封装阶段导出候选共享秘密时不纳入密文. IND-CCA 安全性讨论及所引用的 QROM 界见第 5.1.2 节.

5 安全声明与分析

5.1 理论安全性

软猬 (Mithril) 的主要底层困难问题是环 $\mathbf{R}_q$ 上的 Ring-LWR, 定义 2 对其作了回顾. 对软猬 (Mithril) (以及 玄色 (Octarine)), 环 $\mathbf{R}_q$ 实例化为根式环 $\mathbf{S}_{q,n,k}$, 该实例称为 RR-LWR 问题. 理论安全性讨论基本独立于具体环, 故按如下方式组织:

1. 将本 PKE 方案的安全性归约到一个类似的 Ring-LWR 类 PKE 设计. 具体证明软猬 (Mithril) $:=$ 软猬 (Mithril) $_\ell$ 的安全性可归约到 软猬 (Mithril) $_k$ 的安全性, 且其失败概率不大于 软猬 (Mithril) $_k$; 其中 k 为根式环扩张次数. 见定理 3.
2. 使用把 软猬 (Mithril) $_k$ 与 Ring-LWR 联系起来的归约与分析, 并为 KEM 实例化标准 Fujisaki–Okamoto 型变换. 见第 5.1.1 节和第 5.1.2 节.
3. 最后, 第 5.1.3 节证明根式环选择 $\mathbf{R}_q := \mathbf{S}_{q,n,k}$ 是安全的 Ring-LWR 实例.

定义 2 (判定型 Ring-LWR). 设 $q > p \geq 2$, $\mathbf{R}_\nu := \mathbf{Z}_\nu[x]/(f(x))$ 为环, 其中 $\nu \in \{p, q\}$, D_s 为 $\mathbf{R}_q$ 上的秘密分布. 对于从 D_s 中采样并固定的秘密 $s \in \mathbf{R}_q$, 判定型 Ring-LWR 问题要求区分如下两个分布:

Real: $(\mathbf{A}_i, \lfloor \frac{p}{q} \mathbf{A}_i s \rfloor)$, 其中 $\mathbf{A}_i \leftarrow \mathbf{R}_q$ 对 $i = 1, \dots, N_{\text{samp}}$ 均匀随机;

Rand: $(\mathbf{A}_i, \mathbf{b}_i)$, 其中 $\mathbf{A}_i \leftarrow \mathbf{R}_q, \mathbf{b}_i \leftarrow \mathbf{R}_p$ 对 $i = 1, \dots, N_{\text{samp}}$ 均匀随机.

区分器 $\mathcal{A}$ 的优势定义为

$$\text{Adv}_{q,p,D_s,N_{\text{samp}}}^{\text{R-LWR}}(\mathcal{A}) = |\Pr[\mathcal{A}(\text{Real}) = 1] - \Pr[\mathcal{A}(\text{Rand}) = 1]|.$$

判定型 Ring-LWR 假设称: 对于所有高效区分器 $\mathcal{A}$, 该优势均可忽略.

5.1.1 软猬 (Mithril) .PKE 的 IND-CPA 安全性

按标准含义定义 IND-CPA 实验. 敌手接收公钥 pk , 输出两个等长消息 $m_0, m_1 \in \mathbf{R}_{2,n}^\ell$, 再接收隐藏比特 b 所选消息 m_b 的密文; 若猜中 b 则获胜. 优势记为

$$\text{Adv}_{\text{软猬 (Mithril)}_\ell.\text{PKE}}^{\text{IND-CPA}}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

首先证明 软猬 (Mithril) $_\ell$ 的安全性可归约到 软猬 (Mithril) $_k$ 的安全性.

定理 3. 对于运行 软猬 (Mithril) $_\ell$ 的 IND-CPA 实验的任意敌手 $\mathcal{A}$, 存在运行 软猬 (Mithril) $_k$ 的 IND-CPA 实验的敌手 $\mathcal{B}$, 使得

$$\text{Adv}_{\text{软猬 (Mithril)}_\ell.\text{PKE}}^{\text{IND-CPA}}(\mathcal{A}) = \text{Adv}_{\text{软猬 (Mithril)}_k.\text{PKE}}^{\text{IND-CPA}}(\mathcal{B}).$$

此外, 以 δ_ℓ 表示 软猬 (Mithril) $_\ell$ 的失败概率, 则 $\delta_\ell \leq \delta_k$.

Proof. 设 $pk = (\text{seed}_A, \mathbf{b})$ 为 $\mathcal{B}$ 的挑战者生成的公钥. 敌手 $\mathcal{B}$ 将其转发给 $\mathcal{A}$. $\mathcal{A}$ 返回两个 (ℓn) 比特挑战消息 m_0, m_1 . $\mathcal{B}$ 将二者嵌入较大 (kn) 比特消息的最高 ℓn 比特, 得到 $\mu_0 = 0^{(k-\ell)n} \| m_0$ 和 $\mu_1 = 0^{(k-\ell)n} \| m_1$, 并发送其挑战者. 挑战者返回密文 $c = (c_\mu, \mathbf{b}')$, $\mathcal{B}$ 将其转换为 $c^* = (c_\mu^{[k-\ell:k-1]}, \mathbf{b}')$, 这是 软猬 (Mithril) $_\ell$ 的有效密文. 最后, $\mathcal{A}$ 收到 c^* 后返回比特 b' , $\mathcal{B}$ 将其答复给挑战者. 由于 c^* 对应 软猬 (Mithril) $_\ell$ 对 m_b 的加密, 而 c 对应 软猬 (Mithril) $_k$ 对 μ_b 的加密, $\mathcal{A}$ 与 $\mathcal{B}$ 的优势完全相同.

由 PKE 设计可知 $\ell \leq k$, 且 软猬 (Mithril) $_\ell$ 的密文是 软猬 (Mithril) $_k$ 密文的子集. 因此, 若 软猬 (Mithril) $_\ell$ 解密失败, 则 软猬 (Mithril) $_k$ 也会失败, 从而 $\delta_\ell \leq \delta_k$. $\square$

由于 软猬 (Mithril) $_k$ 是常规 Ring-LWR PKE, 可应用到 Ring-LWR 问题的标准归约. 关于 Saber 场景下 Ring-LWR 类方案归约与分析的概览, 参见 D'Anvers 的论文 [D'V21, 定理 2–3]. D'Anvers 处理的是 Module-LWR, 但令 $\ell = 1$ 即可得到 Ring-LWR. 沿用该分析, 软猬 (Mithril) 的 IND-CPA 安全性可联系到 $\text{Adv}_{q,p,\mathcal{U}_2}^{\text{R-LWR}}$. 该 PKE 归约并非随机预言机模型中的论述; 随机预言机稍后进入 KEM 所用的 FO 变换. 舍入与正确性估计使用方案消息模数 t , 所得约束属于具体参数分析. 特别地, 所有参数集均满足舍入分离条件

$$\frac{q}{p} \leq \frac{p}{t}.$$

与 Saber 相同, 我们以此作为参数选择的依据.

5.1.2 软猬 (Mithril) .KEM 的 IND-CCA 安全性

对 软猬 (Mithril) $_\ell$ 使用标准 IND-CCA KEM 实验. 本构造遵循 Hofheinz–Hövelmanns–Kiltz 的带隐式拒绝模块化 FO 变换 U_m^ℓ [HHK17], 将其应用于 软猬 (Mithril) .PKE 的确定性重加密形式. 对该变换引用文献中的标准 QROM 界. Hofheinz–Hövelmanns–Kiltz [HHK17] 首先建立了 U_m^ℓ 的 QROM 处理, Bindel 等人 [BBH⁺19] 以及 Don–Fehr–Majenz–Schaffner [DFMS19, DFMS21] 通过测量–重编程框架加以改进, Kuchta 等人 [KSS⁺20] 又利用测量–倒带–测量进一步收紧.

将这些结果应用于本 PKE，在量子查询界 q_G 和 q_H 下得到如下形式的界：

$$\text{Adv}_{\text{软狷 (Mithril)}}^{\text{IND-CCA, QROM}}(\mathcal{A}) \leq C \cdot q_G \cdot \sqrt{\text{Adv}_{\text{软狷 (Mithril)}}^{\text{OW-CPA, QROM}}(\mathcal{B})} + \mathcal{O}\left(\frac{q_G^2 + q_H^2}{2^{\ell n}}\right) + q_{\text{dec}} \delta_{\text{dec}} + \Delta_{\text{coll}}^q,$$

其中 C 为绝对常数；由文献 [HHK17] 可直接得到不含平方根、带乘法因子 q_G^2 的较粗界。

5.1.3 RR-LWR 实例的安全性

设 n 为二的幂， k 为正整数。可以证明多项式 $f(x) = (x^k - 2)^n + 1$ 不可约 [BRVW26, 引理 1]，故 $\mathbf{Q}[x]/(f(x))$ 是数域，可在该数域上定义 Ring-LWR 问题。这是标准 LWR [BPR12] 的数域对应形式，并以 软狷 (Mithril) 所用的根式环表示实例化。

由多项式 $f(x) = (x^k - 2)^n + 1$ ，在数域 $\mathbf{K} = \mathbf{Q}[x]/(f(x)) = \mathbf{Q}[\alpha]$ 的整数环 $\mathcal{O}_{\mathbf{K}}$ 上定义 RR-LWR 问题。与环误差学习不同，LWR 样本不含独立采样的 Gaussian 误差项；其噪声是下述逐系数映射引入的确定性量化误差：

$$z \in \mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}} \mapsto \left\lfloor \frac{p}{q}z \right\rfloor \in \mathcal{O}_{\mathbf{K}}/p\mathcal{O}_{\mathbf{K}}.$$

通过显式计算（例如使用 Dedekind 判别准则）可验证 $\mathcal{O}_{\mathbf{K}} = \mathbf{Z}[\alpha]$ 。因此，环

$$S_{q,n,k} = (\mathbf{Z}_q[y]/(y^n + 1)[x]) / (x^k - y - 2)$$

只是 $\mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}}$ 的具体系数表示。下面的定理形式化陈述该表示归约。

定理 4 (作为根式数域上迁移基 Ring-LWR 的 RR-LWR)。设 $q > p \geq 2$ 且 $p \mid q$ ， $f(x) = (x^k - 2)^n + 1$ 不可约，并令 $\mathbf{K} = \mathbf{Q}[x]/(f(x)) = \mathbf{Q}[\alpha]$ 、 $\mathcal{O}_{\mathbf{K}} = \mathbf{Z}[\alpha]$ 。令

$$\varphi_q : S_{q,n,k} \longrightarrow \mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}}$$

为由 $x \mapsto \alpha$ 、 $y \mapsto \alpha^k - 2$ 诱导的环同构，并在模 p 下类似定义 φ_p 。令 $\mathcal{B} = \{(\alpha^k - 2)^i \alpha^j : 0 \leq i < n, 0 \leq j < k\}$ 为 $\mathcal{O}_{\mathbf{K}}$ 的 $\mathbf{Z}$ -基，即单项式基 $\{y^i x^j\}$ 在 φ_q 下的像；令 $\lfloor \cdot \rfloor_{\mathcal{B}}$ 表示在基 $\mathcal{B}$ 中逐系数应用的、从 $\mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}}$ 到 $\mathcal{O}_{\mathbf{K}}/p\mathcal{O}_{\mathbf{K}}$ 的舍入。对 $S_{q,n,k}$ 上任意秘密分布 D_s ，若 $\mathcal{A}$ 是如下 RR-LWR 样本的区分器：

$$\left(\mathbf{A}, \left\lfloor \frac{p}{q} \mathbf{A} \mathbf{s} \right\rfloor \right) \in S_{q,n,k} \times S_{p,n,k}, \quad \mathbf{A} \leftarrow S_{q,n,k}, \quad \mathbf{s} \leftarrow D_s,$$

则相对于均匀样本，它可导出 $\mathcal{O}_{\mathbf{K}}$ 上相应判定型 Ring-LWR 问题的区分器 $\mathcal{B}'$ 。该问题使用舍入 $\lfloor \cdot \rfloor_{\mathcal{B}}$ 和秘密分布 $\varphi_q(D_s)$ ，样本数相同，运行时间基本相同，优势也相同。

Proof. 由于 $S_{q,n,k} \cong \mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}}$ 且 $S_{p,n,k} \cong \mathcal{O}_{\mathbf{K}}/p\mathcal{O}_{\mathbf{K}}$ ，映射 φ_q 与 φ_p 均为可高效计算的双射。RR-LWR 的舍入在基 $\{y^i x^j\}$ 中逐系数进行，而 φ_q 将该基映射到 $\mathcal{B}$ ；故由 $\lfloor \cdot \rfloor_{\mathcal{B}}$ 的定义，对每个 $z \in S_{q,n,k}$ 有

$$\varphi_p\left(\left\lfloor \frac{p}{q}z \right\rfloor\right) = \left\lfloor \frac{p}{q}\varphi_q(z) \right\rfloor_{\mathcal{B}}.$$

归约 $\mathcal{B}'$ 接收 Ring-LWR 挑战样本 $(\tilde{\mathbf{A}}_i, \tilde{\mathbf{b}}_i) \in (\mathcal{O}_{\mathbf{K}}/q\mathcal{O}_{\mathbf{K}}) \times (\mathcal{O}_{\mathbf{K}}/p\mathcal{O}_{\mathbf{K}})$ ，并将其映射为

$$(\mathbf{A}_i, \mathbf{b}_i) = (\varphi_q^{-1}(\tilde{\mathbf{A}}_i), \varphi_p^{-1}(\tilde{\mathbf{b}}_i)).$$

若挑战样本是秘密 $\tilde{s} = \varphi_q(s)$ 对应的 Ring-LWR 样本, 则映射后的二元组恰为秘密 s 对应的 RR-LWR 样本; 若挑战均匀, 则因 φ_q, φ_p 为双射, 映射后二元组仍均匀. 因此 $\mathcal{B}'$ 与 $\mathcal{A}$ 的优势相同. $\square$

需要强调, 定理 4 将 RR-LWR 归约到一个 Ring-LWR 变体, 其舍入在迁移后的单项式基 $\mathcal{B}$ 中逐系数进行, 而非在典范嵌入中进行. 它与典范嵌入版 Ring-LWR 的关系由下述线性换基 L 刻画: 奇异值 $\sigma_{\min}(L) = \sqrt{nk}$ 和 $\sigma_{\max}(L) \leq \beta\sqrt{nk}$ 量化二者转换所引入的失真. 具体攻击估计使用下表常数 β 计入该迁移基失真.

由 f 可将 nk 个根显式写为

$$r_{i,j} = (\xi_{2n}^{(2j+1)} + 2)^{(1/k)} \xi_k^i$$

其中 $i = 0, \dots, k-1, j = 0, \dots, n-1$, ξ_m 为本原 m 次单位根. 规定这些根的顺序, 使 $m = 1, \dots, nk/2$ 时 $r_{m+nk/2} = \overline{r_m}$. 于是, 元素 $a(x) \in \mathbf{K}$ 的典范嵌入由 $i = 1, \dots, nk$ 的 nk 个求值 $a(r_i)$ 构成.

为将 RR-LWR 实例与通常的典范嵌入视角比较, 还需计入: 在环表示 $(\mathbf{Z}_q[y]/(y^n+1)[x])/(x^k-y-2)$ 中按基 $y^i x^j$ ($i = 0, \dots, n-1, j = 0, \dots, k-1$) 逐系数采样, 与 $\mathbf{K}$ 的典范嵌入之间的差异. 该差异是线性变换 L , 可表示为两个线性变换的复合:

- 第一个变换将基 $y^i x^j$ 映射到相应元素 $b(x) = (x^k - 2)^i x^j \bmod f(x)$ (本环中确有 $y = x^k - 2$);
- 第二个变换是标准典范嵌入, 将 $b(x)$ 映射为 $[b(r_i)]_{i=1, \dots, nk}$. 它由 Vandermonde 矩阵给出, 其第 i 行由 $j = 0, \dots, nk-1$ 的 r_i^j 构成.

要界定在基 $y^i x^j$ 中采样相对于典范基采样所引入的最大失真, 只需考察 L 的最小与最大奇异值. 在二的幂分圆多项式 x^n+1 的“理想情形”下, 二者相同, 均为 $\sqrt{n}$. 我们计算了 $k = 2, \dots, 15$ 及不同 n 下 L 的奇异值, 发现最小奇异值为 $\sqrt{nk}$ (nk 是扩张次数, 故为最优), 最大奇异值为依赖 k 的小倍数 $\beta\sqrt{nk}$: $k = 2, 3, 5, 7, 15$ 时分别有 $\beta < 1.74, 2.08, 2.41, 2.57, 2.79$. 这表明系数基采样相对于典范基采样仅产生很小失真. 因此, 软猬 (Mithril) 所用迁移单项式基中表示的秘密分布, 可在仅有该有界基失真的情况下, 与次数为 kn 的数域上通常的典范嵌入 Ring-LWR 视角比较; 继而可用维数 kn 上的通用格攻击估计所得 RR-LWR 实例的困难性.

最后, 由于 $f(x)$ 不可约, 本环结构不同于 m-RLWE 文献 [PUTPPG15, PUTPPG16, APUPG17, PUTPG+21] 所研究的多变量环. 那些工作考虑形如 $R_q[x, y] = \mathbf{Z}_q[x, y]/(g(x), h(y))$ 的环, 其中 g, h 是定义在 $\mathbf{Z}$ 上的单变量多项式. Bootland、Castrick 和 Vercauteren [BCV20] 破解了某些 g, h 选择下的 m-RLWE, 尤其是当二者的根均位于扩张次数小于 $\deg(g)\deg(h)$ 的数域中时. 式 (5) 的环模数 $f(x) = (x^k - 2)^n + 1$ 确实定义次数为 kn 的数域, 故这些攻击不适用. 虽然利用式 (5) 的同构高效使用环结构, 底层格问题仍是单变量环 $S_{q,n,k}$ 上的迁移基 Ring-LWR 问题.

5.1.4 具体攻击估计方法

评估 软猬 (Mithril) 的安全性时, 首先采用广泛使用的核心 SVP 困难度 (Core-SVP hardness) 方法 [ADPS16]. 该方法与具体方案无关, 可比较不同格密码系统的参数强度. Core-SVP 实质上衡量在所选参数下求解底层格问题 (例如由 RR-LWR 导出的唯一最短向量问题 (unique-SVP) 实例) 的困难程度, 并假定最佳攻击使用格约减 (例如块 Korkine–Zolotarev (BKZ) 算法 [SE94, CN11]) 及块维数 β 上的最优 SVP 预言机. 忽略格约减的多项式时间开销和存储成

方案	Type	C	Q	β	Refined attack	fail prob.
软猬 (Mithril) -128	primal	134	121	458	153	-603
	dual-hybrid	129	117	443		
软猬 (Mithril) -256	primal	270	245	925	277	-280
	dual-hybrid	257	233	879		
软猬 (Mithril) -512	primal	559	507	1913	539	-230
	dual-hybrid	526	478	1802		
ML-KEM-512	primal	119	108	406	140	-139
	dual-hybrid	115	105	395		
ML-KEM-768	primal	182	165	624	196	-165
	dual-hybrid	174	158	597		
ML-KEM-1024	primal	255	232	874	262	-175
	dual-hybrid	241	219	826		
LightSaber	primal	118	107	405	140	-120
	dual-hybrid	122	111	419		
Saber	primal	190	172	649	205	-136
	dual-hybrid	182	165	624		
FireSaber	primal	260	237	891	267	-165
	dual-hybrid	248	225	848		

Table 3: 软猬 (Mithril) 与相关格方案的安全性估计、精化攻击成本及失败概率比较.

本等低阶因素后, 攻击成本可估计为随 BKZ 块大小 β 指数增长. 假设调用一次理想的长度 β SVP 求解器, 在经典场景下约需 $2^{0.292\beta}$ 次基本操作; 对于量子攻击者约需 $2^{0.265\beta}$ 次操作, 其中计入了基于筛法的 SVP 算法目前最佳的 Grover 型加速 [Laa15, LMvdP15]. 该粗粒度指标主要依赖格维数和噪声率, 而非方案的具体代数结构, 便于直接比较不同格提案; 同时它通常偏保守, 促使参数即使在有利于敌手的成本模型下也能抵御攻击.

对于可进一步归约到 LWE 的 RR-LWR 类方案, 两类主要格攻击为原始攻击 (*primal attack*) 和对偶混合攻击 (*dual-hybrid attack*). 原始攻击将 LWE 实例化为 uSVP 实例并尝试通过格约减直接恢复秘密; 对偶混合攻击则把对偶格区分器与对部分秘密的组合猜测步骤结合. 在 Core-SVP 框架中, 通常先确定使相应区分或解码步骤成功所需的 BKZ 块大小 β , 再换算为估计成本.

Core-SVP 指标提供简洁、稳健的基线, 但有意抽象掉实际攻击复杂度的许多方面. 实践中, 还使用 Albrecht 等人的 lattice-estimator 框架 [APS15] 给出精化成本估计. 其中 `estimate.rough` 主要由 BKZ 块大小 β 推导安全级别, 与 Core-SVP 方法密切对应, 因而提供一致且与方案无关的参照点.

为得到更现实的估计, 另使用完整 `estimate` 函数, 显式计入完整攻击流程所需的环操作数 (rop) 以精化攻击成本. 除底层 SVP 预言机的渐近成本外, 还包括 BKZ 约减、免费维数技术、混合策略及其他算法优化的开销. 因此, 精化估计可反映 Core-SVP 所忽略的重复 SVP 调用成本、不可忽略的多项式因子, 以及实际时间-存储权衡.

从概念上说, 精化估计能更准确地反映具体安全性, 尤其是经典攻击; 近年来格约减和筛法的进展已显著提高其实际性能. 不过, 更高现实性以额外建模假设为代价: 结果依赖于对 BKZ 行为、筛法实现效果及底层操作成本模型有效性的启发式假设. 因此, 精化估计随时间的稳定性较低, 对未来算法改进也更敏感.

因此同时报告两类指标: 以 Core-SVP 困难度作为保守、可比的基线, 以基于 rop 的精化

估计表示当前对具体攻击成本的最佳认识. 这种双重视角符合合格方案分析的标准做法, 即用更详细但依赖假设的评估补充粗粒度而稳健的指标.

表 2 列出 软猬 (Mithril) 各参数集的安全级别. 表 3 将其与基于 LWE 的 KEM ML-KEM 及基于 LWR 的 KEM Saber 进行安全性比较. 这些数值的获取方法详见附录 A.

5.2 实际安全性

5.2.1 解密失败攻击

与大多数 LWE/LWR 类方案一样, 解密并非始终完全正确. 当确定性舍入误差与私钥和密文向量间的相互作用累积到足以使解密系数越过解码边界时, 即发生解密失败. 此类错误可被检测, 并可能泄漏私钥信息.

解密失败攻击主动利用该泄漏. 敌手特意构造可人为放大误差项的密文 [DGJ⁺19, DRV20, DB22], 通过观察解密成功与否获知关于私钥系数的精确线性不等式. 系统地积累足够多诱发失败的密文后, 敌手可建立不等式组, 并用统计技术或格约减算法求解, 从而完整恢复私钥.

软猬 (Mithril) 的解密失败与 Saber 等其他 Ring/Module-LWR 方案的产生方式相似. 依照文献 [DKR⁺20, 定理 6.2] 的计算框架, 使用 Saber 型启发式估计器计算失败概率. 换成本方案记号, 该估计器基于事件

$$\delta = \Pr[\|(\mathbf{b}\mathbf{s}')^{[k-\ell:k-1]} - (\mathbf{b}'\mathbf{s})^{[k-\ell:k-1]} + \mathbf{e}_r, \text{ mod } p\|_\infty > p/4]$$

其中 $\mathbf{e}_r$ 表示为 $\mathbf{R}_p^\ell$ 中的元素, 其中心系数位于 $[-p/(2t), p/(2t) - 1]$. 与以往 Saber 型估计相同, 启发式地将共享秘密各比特的失败事件建模为相互独立; 本文不声称严格证明了所有系数相关性.

由于本方案在根式环上运算, 应注意 $(\mathbf{b}\mathbf{s}')^{[k-\ell:k-1]}$ 与 $(\mathbf{b}'\mathbf{s})^{[k-\ell:k-1]}$ 是 $\mathbf{R}_q^\ell$ 中的元素, 而相应 Module-LWR 表达式通常是 $\mathbf{R}_q$ 中的元素. 换言之, 私钥比特由 ℓ 个各为 n 次的多项式系数导出, 而 Module-LWR 由含 n 个系数的单个多项式导出. 由于选择 $n = 128$ 而非 $n = 256$, 在 $\ell = 1$ 的 128 比特安全级别下可得到更低的失败概率.

不过, 根式环结构中, $\mathbf{b}$ 与 $\mathbf{s}'$ 的第 i 行内积包含 $k-1-i$ 次乘以 $y+2$ 的操作, 见式 (6). 这会增大系数范数, 故在失败概率计算中计入最坏情况因子 3. 由于计算的是底部各行, 可尽量减小乘以 $y+2$ 的影响. 例如, 软猬 (Mithril) -128 取 $\ell = 1$ 时只使用一行, 而底行不涉及 $y+2$, 实际不会发生此类乘法, 故该乘法等价于 Module-LWR 中的常规内积.

附录 A.2 的脚本为表 2 和表 3 所列 归藏-软猬 (ARCANE-Mithril) 各参数集实现了该启发式解密失败概率估计器. 由于估计失败概率的对数远低于实际可能发生的水平, 解密失败攻击对 软猬 (Mithril) 安全性的实际影响可视为不显著.

5.2.2 侧信道攻击

算法不含任何依赖秘密的分支, 故为常数时间. 此外, 二的幂模数简化了若干计算, 例如生成随机元素时可避免非常数时间的拒绝采样. 因此, 只要底层硬件指令也是常数时间, 正确的归藏-软猬 (ARCANE-Mithril) 实现应为常数时间.

然而, 未防护的 软猬 (Mithril) 实现可能在解封装时通过功耗或电磁辐射泄漏 [RRCB20, XPR⁺22, RCDB24], 帮助敌手猜测长期私钥. 因此, 针对已知侧信道攻击采取对策至关重要.

软猬 (Mithril) 计算的确定性执行方式及其采用二的幂模数的 LWR 基础, 使掩蔽对策能够高效实现. Saber KEM 文献 [BDK⁺21] 中的对策技术可应用于 软猬 (Mithril) .

- **二的幂模数简化掩蔽.** 算术掩蔽、模约减、舍入、压缩及布尔到算术掩码转换均可通过移位和掩码低开销实现.
- **确定性 LWR 舍入避免显式误差采样.** 这减少了必须掩蔽的随机量, 并使舍入比 LWE 型噪声加法更易防护.
- **无拒绝的固定时间采样.** 每个系数使用固定数量的比特即可生成随机元素, 避免使常数时间和掩蔽实现复杂化的拒绝循环.
- **简单秘密分布支持掩蔽乘法.** 小型有符号系数集合 $\{-2, -1, 0, 1\}$ 简化了秘密系数的掩蔽处理, 并降低受保护算术的复杂度 [ABC⁺23].

6 性能评估

软猬 (Mithril) 的算术设计有意不依赖所采用的多项式乘法算法. 规范定义所需的根式环运算, 但不限定负循环多项式乘积所用的具体算法或元素表示. 因而实现可根据目标平台或优化目标选择最适合的乘法策略. 为展示这一灵活性, 参考实现采用 Toom–Cook 乘法 [Too63, Coo66], AVX2 实现则在辅助素数上使用数论变换 [CT65, Pol71].

6.1 算术实现

参考实现. 参考实现在二的幂系数环上以 16 比特存储系数并累加最终结果. 次数为 128 的负循环子环乘法采用四路 Toom–Cook 乘法. 对 软猬 (Mithril) -512, Toom–Cook 在折叠至 16 比特累加器前使用 32 比特求值、64 比特精确乘积和插值. 关系 $x^k = y + 2$ 通过预计算乘法窗口纳入: 根式基中每个所需系数均为负循环子环上长度 k 的内积. 因而实现既可计算完整根式环乘积, 也可仅计算其 ℓ 个最高次系数. 在可能时, 舍入和模二的幂约减与最终累加和序列化合并执行.

AVX2 实现. AVX2 实现对 软猬 (Mithril) -128 和 软猬 (Mithril) -256 在长度为 128 的数论变换 (NTT) 中使用 26 比特辅助素数

$$53\,501\,952 + 1 = 2^{13} \cdot 3 \cdot 7 \cdot 311 + 1$$

, 并对 软猬 (Mithril) -512 使用 28 比特素数 268 432 897. 系数以 Montgomery 域表示; 公开操作数仅变换一次, 并为该操作所需的全部根式基内积作预处理. 对仅需消息承载根式系数的公钥和密文分量, 预处理仅限于所需的 ℓ 个输出系数. 正变换、逐点乘法、累加、逆变换以及最后的二的幂舍入或约减均以 AVX2 向量化. 该辅助素数仅为实现层面的选择: 其足够大以容纳未经约减的中间乘积, 不会改变任何 软猬 (Mithril) 参数或分布.

6.2 性能分析

采样与扩展. 三个参数集均采用 $n = 128$ 和 2 比特秘密系数, ϵ_q 分别为 13, 13, 14; 根式扩张次数 $k \in \{5, 9, 18\}$ 、消息维数 $\ell \in \{1, 2, 4\}$ 和密文压缩模数 t 则各不相同. 种子采样器产生比特

打包的 XOF 输出，可直接解包，无需拒绝采样. 扩展 **A** 需要

$$\frac{\epsilon_q n k}{8}$$

字节的 XOF 输出；对 软猬 (Mithril) -128、软猬 (Mithril) -256 和 软猬 (Mithril) -512，这分别为 1040、1872 和 4032 字节. 采样一个秘密环元素需要

$$\frac{2nk}{8} = 32k$$

字节，即分别为 160、288 和 576 字节. 因此，采样成本主要由 XOF 吞吐量、比特解包以及 AVX2 实现中的变换域表示预处理决定.

根式环乘积. 对完整乘积， k 个输出系数中的每一个均为长度 k 的内积，因此需要

$$N_{\text{full}} = k^2$$

个基础负循环多项式乘积. 仅计算 ℓ 个消息承载系数则需要

$$N_{\text{sel}} = k\ell$$

个基础乘积. 因此，对 软猬 (Mithril) -128、软猬 (Mithril) -256 和 软猬 (Mithril) -512，完整乘积分别使用 25、81 和 324 个基础乘积；而共享秘密计算中的多项式乘积（仅需计算部分输出，见第 4 节“消息空间”段）分别使用 5、18 和 72 个基础乘积. 结合每个基础乘积的成本，这些计数解释了不同参数集之间的主要扩展关系.

操作成本. 软猬 (Mithril) .PKE 密钥生成扩展 **A**、采样 **s**、计算完整乘积 **As**、从 q 舍入至 p ，并序列化密钥对. 加密扩展 **A**、采样 **s'**、计算完整乘积 **As'**，再在消息相加和密文压缩前仅计算 **bs'** 的 ℓ 个最高次系数. 解密不扩展 **A**，仅需计算相应的 ℓ 系数乘积 **b's**，随后恢复消息.

KEM 操作还包括第 4 节规定的 Fujisaki-Okamoto 哈希和随机性生成. 解封装执行 PKE 解密，导出候选密钥和加密种子，对恢复的消息确定性重加密，由密文和 z 导出备用密钥，并以常数时间选择输出. 解封装同时包含 PKE 解密和确定性重加密，因此在最小参数集下其成本与其他 KEM 操作相当，在两个较大参数集中则成为成本最高的 KEM 操作.

通信和存储成本见表 2. KEM 公钥即 PKE 公钥，KEM 密文即 PKE 密文；KEM 私钥包含 PKE 私钥、PKE 公钥、公钥摘要和备用秘密 z .

6.3 性能测试

实验设置. 所有测量均在标称频率为 3.80 GHz、支持 AVX2、BMI2 和 POPCNT 的 Intel Core i7-10700K CPU 上采集，操作系统为 Linux 6.8.0-138-generic，编译器为 GCC 11.4.0. 参考实现以 C 编写，AVX2 实现结合 C 与 x86-64 汇编. 参考实现使用 -O3 编译；AVX2 实现使用 -O3、-fomit-frame-pointer、-mavx2、-mbmi2、-mpopcnt、-march=native 和 -mtune=native 编译. 表 4 和表 5 所报告的基于 pseudoXOF 的提交版本测量中，受测二进制程序采用提交包内基于 pseudoXOF 的构造实例化 XOF 和哈希函数.

每个基准二进制程序在 10 次预热调用后，对每项操作内部执行 1000 次测量重复，并使用 `taskset -c 2` 在 CPU 2 上串行运行. 计时使用处理器性能监控单元 (PMU) 统计用户态和内

核态 CPU 周期，并扣除了测得的计数器读取开销。表中报告 PMU 周期中位数，数值越低表示越快。全函数测量包含相应算法调用所执行的采样、哈希、算术、打包及其他全部工作。CPU 频率调控器和睿频设置保持不变。下文 ML-KEM 比较所用实现取自本地源代码树的 Git 修订版 3edd5af59919。

端到端性能。 表 4 和表 5 给出提交包中基于 `pseudoXOF` 的参考实现和 AVX2 实现的端到端 PKE 与 KEM 操作成本。

参数集	实现	KeyGen	Enc	Dec
软猬 (Mithril) -128	参考实现	261 229	285 001.5	29 459.5
	优化实现	130 985	133 616	5 259
软猬 (Mithril) -256	参考实现	639 487.5	733 868	98 042.5
	优化实现	236 764.5	242 918.5	10 031
软猬 (Mithril) -512	参考实现	5 028 955.5	6 075 621	1 044 501.5
	优化实现	480 143	493 836	22 930.5

Table 4: 参考实现和优化实现中 软猬 (Mithril) .PKE 基于 `pseudoXOF` 的端到端性能。数值为 PMU 周期中位数。

参数集	实现	KeyGen	Encaps	Decaps
软猬 (Mithril) -128	参考实现	302 651	324 044	351 341
	优化实现	171 523	171 332	172 333
软猬 (Mithril) -256	参考实现	698 618.5	790 580.5	888 295.5
	优化实现	294 582.5	297 396	305 749
软猬 (Mithril) -512	参考实现	5 232 025	6 277 211.5	7 364 776
	优化实现	659 789.5	679 568	715 160

Table 5: 参考实现和优化实现中 软猬 (Mithril) .KEM 基于 `pseudoXOF` 的端到端性能。数值为 PMU 周期中位数。

AVX2 加速比随 k 增大而提高，但由于基于 `pseudoXOF` 的采样和哈希仍占测量路径的重要部分，端到端增益小于纯算术增益。三个参数集的 PKE 密钥生成加速比分别为 $1.99\times$ 、 $2.70\times$ 和 $10.47\times$ ，PKE 加密加速比分别为 $2.13\times$ 、 $3.02\times$ 和 $12.30\times$ 。由于算术占比更高，PKE 解密加速比分别达到 $5.60\times$ 、 $9.77\times$ 和 $45.55\times$ 。两种实现均只计算选定的 ℓ 个系数。KEM 加速比介于 $1.76\times$ 至 $10.30\times$ ；较小的加速比源于密钥生成、封装和解封装中的额外 `pseudoXOF` 哈希。

算术分解。 表 6 单独列出参考实现中的主要算术成本。

次数为 128 的 Toom–Cook 乘法在 软猬 (Mithril) -128 和 软猬 (Mithril) -256 中成本近似相同，而 软猬 (Mithril) -512 使用更宽的精确中间算术，基础乘积成本较高。完整乘积和单个系数测量分别反映 k^2 和 k 的计数以及该基础乘积成本。端到端解密路径需要 $k\ell$ 个基础乘积，另包括解包、约减、舍入和消息序列化。

表 7 给出 AVX2 实现的相应微基准测试结果。

参数集	基础 Toom-Cook	完整乘积	单个选定系数	PKE Dec
软猬 (Mithril) -128	4990	123 136	24 632	29 459.5
软猬 (Mithril) -256	4966.5	396 510	44 083	98 042.5
软猬 (Mithril) -512	14 137.5	4 527 987	250 985.5	1 044 501.5

Table 6: 参考实现中的选定算术成本. “单个选定系数”列测量一个最高次根式系数; PKE Dec 包含全部所需的 ℓ 个系数和消息恢复. 数值为 PMU 周期中位数.

参数集	扩展并 预处理 A	采样 秘密	变换 秘密	完整乘积 并舍入	ℓ 系数乘积 并约减
软猬 (Mithril) -128	108 869.5	15 505	1 878	5 214	2 594
软猬 (Mithril) -256	195 859.5	27 673	3 350	10 955.5	5 087
软猬 (Mithril) -512	391 975	54 940	7 126	27 581	11 595

Table 7: AVX2 实现中的选定采样与算术成本. 两个乘积列均为端到端内核, 已包含秘密变换, 故各列不可相加. 数值为 PMU 周期中位数.

扩展和预处理 **A** 以及变换秘密的成本主要随 k 增长; 完整乘积内核则反映变换域预处理后的 k^2 根式基结构. 选定系数乘积仅计算 ℓ 个输出系数, 增长较慢. 特别地, 每个操作数的变换域表示仅准备一次, 并复用于同一操作中的全部内积. 乘积微基准使用预处理后的公开操作数.

与 ML-KEM 的比较. 表 8 在同一平台上, 将附加实现目录中基于 SHAKE 的 AVX2 软猬 (Mithril) 实现与 ML-KEM-512 AVX2 和 ML-KEM-1024 AVX2 进行比较. 软猬 (Mithril) -128 参数集对应 ML-KEM-512, 软猬 (Mithril) -256 参数集对应 ML-KEM-1024. 由于 ML-KEM 没有以相应 512 比特经典安全级别为目标的参数集, 未列入软猬 (Mithril) -512. 这些均为端到端比较: 每项测量均包含受测算法调用中的哈希套件、序列化和随机性生成.

操作	软猬 (Mithril) -128 AVX2	ML-KEM-512 AVX2	软猬 (Mithril) -256 AVX2	ML-KEM-1024 AVX2
PKE KeyGen	20 555	15 100	35 013	39 798
PKE Enc	23 765	16 142	41 709	41 259
PKE Dec	5 220	1 240	9 999	2 133
KEM KeyGen	36 371	25 987.5	57 685	59 232
KEM Encaps	37 904	27 458	62 997	60 411
KEM Decaps	42 075	29 421.5	73 418	65 375

Table 8: 基于 SHAKE 的 AVX2 与 ML-KEM AVX2 端到端比较. 数值为 PMU 周期中位数.

纳入 ML-KEM 测量结果, 是为了与成熟且高度优化的格 KEM 实现进行外部标定, 而非作严格的同等安全性或一一对应比较. 所比较的参数集具有不同的安全裕度、编码数据大小和代数假设. 在第一个 AVX2 标定点, ML-KEM-512 在六项测量操作中的周期中位数均低于软猬 (Mithril) -128. 这与软猬 (Mithril) -128 的目标估计格安全裕度高于 ML-KEM-512 一致: 对软猬 (Mithril) -128, 估计的经典和量子 Core-SVP 成本分别为 129 和 117 比特, 精化门数估计为 153 比特; ML-KEM-512 相应为 115、105 和 140 比特.

在 ML-KEM-1024 级别的标定点, AVX2 结果更依赖于具体操作. 软猬 (Mithril) -256 的 PKE 密钥生成约快 12.0%, 而 KEM 密钥生成的中位数相差不足 2.7%, 表明该操作的性能实质上相当. ML-KEM-1024 在 PKE 加密、PKE 解密、KEM 封装和 KEM 解封装中具有更低中

位数；不过仍应结合不同安全裕度理解这一比较，因为 软猬 (Mithril) -256 的估计经典和量子 Core-SVP 成本为 257 和 233 比特，精化估计为 277 比特，而 ML-KEM-1024 相应为 241、219 和 262 比特。

这些差异也与方案不同的算术结构一致。ML-KEM 在固定次数为 256、模秩较小的分圆环上使用 Module-LWE，而 软猬 (Mithril) 在 $n = 128$ 、 $k \in \{5, 9, 18\}$ 的根式环表示上使用 RR-LWR 算术，消息提取由 ℓ 控制。AVX2 实现的选定系数优化减少了解密和解封装所需的根式环输出量，但不能消除后端的扩展、变换和预处理成本。因此，所测性能应视为当前 AVX2 实现在较强 RR-LWR 参数裕度下的实用基线，而非与 ML-KEM 的直接等成本比较。

致谢

感谢三位博士研究生张世聪、钟晨欣和黄浩宸参与算法设计的技术讨论，并为提交材料的准备作出贡献。其中，张世聪参与算法描述，钟晨欣参与安全性分析，黄浩宸参与软件实现。

附录

A SageMath 脚本

A.1 安全性估计

我们使用 Albrecht 等人的 lattice-estimator 框架 [APS15] 估计 RR-LWR 实例的具体困难度。具体采用以下格估计器实现：malb/lattice-estimator，提交版本为 be81b18。

为报告可与 Core-SVP 比较的安全性数值（格密码学文献中的常见做法），我们使用估计器的“粗略”接口 `LWE.estimate.rough`。对更精细的指标，则调用 `LWE.estimate`。

表 2 和表 3 所报告的经典与量子 Core-SVP 比特安全性数值及精化攻击结果，可在 lattice-estimator 仓库中运行以下脚本复现。为获得 软猬 (Mithril) -512 的安全性估计，需要在 estimator/conf.py 中将块大小上界调整为

```

1 max_beta = 3500

1 from estimator import *
2 from estimator.nd import UniformMod
3 from estimator.lwe_parameters import LWEParameters
4
5
6 def print_core_svp_table(results):
7     schemes = list(results.keys())
8
9     rows = [("bits", [f"{results[s]['bits']:.1f}" for s in schemes]),
10            ("beta", [str(results[s]["beta"]) for s in schemes]),
11            ("classical core-SVP", [f"{results[s]['classical_svp']:.1f}"
12                                   for s in schemes]),
13            ("quantum core-SVP", [f"{results[s]['quantum_svp']:.1f}" for s
14                                  in schemes]),
15            ("beta-extended", [str(results[s]["beta_ext"]) for s in schemes
16                               ]),

```

```

14         ("rop-extended", [str(results[s]["bits_ext"]) for s in schemes
15                             ]), ]
16
17     header = [""] + schemes
18     table = [header] + [[name] + vals for name, vals in rows]
19
20     col_widths = [max(len(str(cell)) for cell in col)
21                   for col in zip(*table)]
22
23     for row in table:
24         print(" ".join(str(cell).rjust(w)
25                         for cell, w in zip(row, col_widths)))
26
27 def best_attack(estimate):
28     return min(((results["beta"], N(log(results["rop"], 2)))
29               for attack, results in estimate.items()
30               if "beta" in results),
31               key=lambda x: x[1])
32
33 # Define our parameter sets
34 rrlwr5_params = LWEParameters(
35     n=5*128,
36     q=2**13,
37     Xs=UniformMod(4),
38     Xe=UniformMod(2**(13-11)), # p = 2^11
39     m=5*128,
40     tag="RR-LWR-5"
41 )
42 rrlwr9_params = LWEParameters(
43     n=9*128,
44     q=2**13,
45     Xs=UniformMod(4),
46     Xe=UniformMod(2**(13-11)), # p = 2^11
47     m=9*128,
48     tag="RR-LWR-9"
49 )
50 rrlwr17_params = LWEParameters(
51     n=17*128,
52     q=2**13,
53     Xs=UniformMod(4),
54     Xe=UniformMod(2**(13-11)), # p = 2^11
55     m=17*128,
56     tag="RR-LWR-17"
57 )
58
59 params = {
60     # The proposed RR-LWR parameter sets
61     "rrlwr5": rrlwr5_params,
62     "rrlwr9": rrlwr9_params,
63     "rrlwr17": rrlwr17_params,

```

```

63     # Kyber* for comparison
64     "kyber512":    schemes.Kyber512,
65     "kyber768":    schemes.Kyber768,
66     "kyber1024":   schemes.Kyber1024,
67     # *Saber for comparison
68     "lightsaber":  schemes.LightSaber,
69     "saber":       schemes.Saber,
70     "firesaber":   schemes.FireSaber,
71 }
72
73 results = {}
74 for name, p in params.items():
75     estimate = LWE.estimate.rough(p, quiet=True, jobs=4)
76     beta, bits = best_attack(estimate)
77     results[name] = {"beta": beta, "bits": bits }
78     estimate_extended = LWE.estimate(p, quiet=True, jobs=4)
79     beta_ext, bits_ext = best_attack(estimate_extended)
80     results[name] = {"beta": beta, "bits": bits,
81                     "beta_ext": beta_ext, "bits_ext": bits_ext }
82
83 for r in results.values():
84     beta = r["beta"]
85     r["classical_svp"] = 0.292 * beta
86     r["quantum_svp"] = 0.265 * beta
87
88 print_core_svp_table(results)

```

A.2 失败概率

我们使用以下 SABER 实现中的失败概率脚本: <https://github.com/KULEuven-COSIC/SABER>, 提交版本为 `f7f39e4`, 用于估计失败概率. 我们针对根式环结构作了若干调整, 见第 5.2.1 节. 表 2 和表 3 报告的失败概率可通过运行下述脚本复现.

```

1     #
2     # This file has been adapted from the implementation
3     # (available at, Public Domain https://github.com/KULEuven-COSIC/SABER/
4     #   blob/master/Scripts/Python_Scripts/select_params/proba_util.py)
5     #
6     from math import factorial as fac
7     from math import log, ceil, erf, sqrt, floor
8     import decimal
9
10    decimal.getcontext().prec = 1000
11
12    def mod_switch(x, q, rq):
13        """ Modulus switching (rounding to a different discretization of
14            the Torus)
15        :param x: value to round (integer)
16        :param q: input modulus (integer)

```

```

16     :param rq: output modulus (integer)
17     """
18
19     z = int(floor(1.* rq * x / q + rq/(2*q)) % rq) # Switched from '
        round' to 'floor' to round 0.5 upwards
20
21     if (z >= rq/2):
22         z -= rq
23
24     return z
25
26 def mod_centered(x, q):
27     """ reduction mod q, centered (ie represented in -q/2 .. q/2)
28     :param x: value to round (integer)
29     :param q: input modulus (integer)
30     """
31     a = x % q
32     if a < q/2:
33         return a
34     return a - q
35
36 def build_mod_switching_error_law(q, rq):
37     """ Construct Error law: law of the difference introduced by
        switching from and back a uniform value mod q
38     :param q: original modulus (integer)
39     :param rq: intermediate modulus (integer)
40     """
41     D = {}
42     V = {}
43     for x in range(-int(q/2), int(q/2)):
44         y = mod_switch(x, q, rq)
45         z = mod_switch(y, rq, q)
46         d = mod_centered(x - z, q)
47         D[d] = D.get(d, 0) + 1./q
48         V[y] = V.get(y, 0) + 1
49
50     return D
51
52 def law_convolution(A, B):
53     """ Construct the convolution of two laws (sum of independent
        variables from two input laws)
54     :param A: first input law (dictionnary)
55     :param B: second input law (dictionnary)
56     """
57
58     C = {}
59     for a in A:
60         for b in B:
61             c = a+b
62             C[c] = C.get(c, 0) + A[a] * B[b]

```

```

63         return C
64
65     def law_convolution_neg(A, B):
66         """ Construct the convolution of two laws (sum of independent
67             variables from two input laws)
68         :param A: first input law (dictionnary)
69         :param B: second input law (dictionnary)
70         """
71         C = {}
72         for a in A:
73             for b in B:
74                 c = a-b
75                 C[c] = C.get(c, 0) + A[a] * B[b]
76         return C
77
78     def law_product(A, B):
79         """ Construct the law of the product of independent variables from
80             two input laws
81         :param A: first input law (dictionnary)
82         :param B: second input law (dictionnary)
83         """
84         C = {}
85         for a in A:
86             for b in B:
87                 c = a*b
88                 C[c] = C.get(c, 0) + A[a] * B[b]
89         return C
90
91     dist_rate = -100000 # Add this to stop pruning too early
92     def clean_dist(A):
93         """ Clean a distribution to accelerate further computation (drop
94             element of the support with proba less than 2^dist_rate)
95         :param A: input law (dictionnary)
96         """
97         B = {}
98         for (x, y) in A.items():
99             if y>2**(dist_rate):
100                 B[x] = y
101         return B
102
103     def iter_law_convolution(A, i):
104         """ compute the -ith forld convolution of a distribution (using
105             double-and-add)
106         :param A: first input law (dictionnary)
107         :param i: (integer)
108         """
109         D = {0: 1.0}
110         i_bin = bin(i)[2:] # binary representation of n
111         for ch in i_bin:

```

```

109         D = law_convolution(D, D)
110         D = clean_dist(D)
111         if ch == '1':
112             D = law_convolution(D, A)
113             D = clean_dist(D)
114     return D
115
116 def convolution_remove_dependency(A, B, q, p):
117     normalizer={}
118     maxa=int(q/p)
119     for a in A:
120         normalizer[a%maxa]=normalizer.get(a%maxa,0)+A[a]
121
122     C={}
123     for a in A:
124         for b in B:
125             c=a-b
126             if (c%maxa==0):
127                 C[c] = C.get(c, 0) + A[a] * B[b]/normalizer[a%maxa]
128     return C
129
130 def tail_probability(D, t):
131     '''
132     Probability that an drawn from D is strictly greater than t in
133     absolute value
134     :param D: Law (Dictionnary)
135     :param t: tail parameter (integer)
136     '''
137     s = 0
138     ma = max(D.keys())
139     if t >= ma:
140         return 0
141     for i in reversed(range(int(ceil(t)), ma)): # Summing in reverse
142         for better numerical precision (assuming tails are decreasing)
143             s += D.get(i, 0) + D.get(-i, 0)
144     return s
145
146 def build_uniform_law(k):
147     """ Construct the binomial law as a dictionnary
148     :param k: (integer)
149     :param x: (integer)
150     :returns: A dictionnary {x:p_k(x) for x in {-k..k}}
151     """
152     D = {}
153     for i in range(-k, k):
154         D[i] = 1/(2*k)
155     return D
156
157 def get_failure_prob(D_s, q, p, T, k, ell, n):

```

```

157     # Compute rounding error distribution
158     D_e = build_mod_switching_error_law(q, p)
159
160     assert D_e == D_s # True for Mithril parameter sets
161
162     # Compute shared secret error distribution
163     se = law_product(D_s, D_e) # Dist on single product of input
        polynomial coefficients
164
165     prob_secret_success_polys = ell*[0]
166     for i in range(ell):
167
168         # For row i we get k-i polynomials without factor (y+2), i
        polynomials with (y+2) that get an additional factor 3
169         # Total (k-i) + 3*i = k+2*i
170         se2 = iter_law_convolution(se, (k+2*i)*n) # Dist on accumulated
        polynomial coefficient of s'*e, assume each coefficient is
        independent
171         se2 = law_convolution_neg(se2, se2) # Dist on accumulated
        polynomial coefficient of s'*e - e'*s
172         e2 = build_mod_switching_error_law(q, T) # Dist on ciphertext
        error
173         D = law_convolution(se2, e2) # Dist in shared secret
        coefficient
174
175         prob_coeff_fail = tail_probability(D, q/4.) # Failure
        probability of a single shared secret bit
176         a = decimal.Decimal(prob_coeff_fail)
177         prob_secret_success = (1-a)**n # Success probability for all
        bits in this polynomial
178         prob_secret_success_polys[i] = prob_secret_success
179
180     # Compute success probability across all secret key polynomials
181     prob_success = decimal.Decimal(1)
182     for i in range(ell):
183         prob_success *= prob_secret_success_polys[i]
184
185     prob_fail = log(1-prob_success, 2)
186
187     return prob_fail
188
189 def main():
190     D_s = build_uniform_law(2)
191     prob128 = get_failure_prob(D_s, 2**13, 2**11, 2**3, 5, 1, 128)
192     print('Failure probability 128-bit security level: ', prob128)
193     prob256 = get_failure_prob(D_s, 2**13, 2**11, 2**3, 9, 2, 128)
194     print('Failure probability 256-bit security level: ', prob256)
195     prob512 = get_failure_prob(D_s, 2**13, 2**11, 2**8, 17, 4, 128)
196     print('Failure probability 512-bit security level: ', prob512)
197

```

B 检查清单

以下检查清单依据提交要求文件编制，便于定位相关内容。

• 1. 算法描述

- 完整算法运行流程：第 4 节，尤其是算法 1.
- PKE 的 KeyGen、Encrypt 和 Decrypt 算法：第 4 节，算法 1.
- KEM 的 KeyGen、Encaps 和 Decaps 算法：第 4 节，算法 1.
- 数学记号与基本定义：第 3.1 节.
- LWE/LWR 背景与公式：第 3.2 节.
- 根式环结构与公式：第 3.3 和 3.4 节.
- 根式环上的结构化乘法：第 3.4 节，尤其是式 (6).
- 所有安全强度级别的算法实例参数：第 4 节，表 2.
- 软猬-128、软猬-256 和软猬-512 的参数：第 4 节，表 2.
- 密钥、密文和共享秘密大小：第 4 节，表 2.
- 参数表中的失败概率和安全性估计：第 4 节，表 2.
- 采样、随机性、种子、哈希/XOF 使用及输入输出处理：第 3.1 和 4 节.
- 支持算法描述的表格：第 2.2 节表 1；第 4 节表 2；第 5 节表 3；以及第 6 节表 4-8.
- 算法规范：第 4 节，算法 1.

• 2. 设计原理

- 主要设计思想与原理：第 1 和 2.1 节.
- 数学困难问题：第 2.1、3.4 和 5.1.3 节.
- 以 RR-LWR 为底层假设：第 1、2.1、3.4 和 5.1.3 节.
- 参数选择准则：第 1 和 2.1 节及表 2.
- 根式环结构的选择：第 1、2.1 和 3.4 节.
- n 、 k 、 p 、 q 、 t 和 ℓ 的选择：第 2.1 节及表 2.
- 二的幂模数与均匀采样的使用：第 1 和 2.1 节.
- 失败概率及其安全影响的详细分析：第 2.1 和 5.2.1 节，以及附录 A.2.
- 其他设计考虑：第 2.1 节，包括采样、部分环元素、Fujisaki–Okamoto 变换和高保障实现.
- 与现有工作的关系及比较依据：第 2.2 节.

• 3. 安全性声明与分析

- 算法安全性声明：第 5 节.
- 理论安全性：第 5.1 节.
- RR-LWR 困难问题：第 5.1.3 节.
- 软猬.PKE 的 IND-CPA 安全性：第 5.1.1 节，定理 3.
- 软猬.KEM 的 IND-CCA 安全性：第 5.1.2 节.

- 软猬.KEM 的 QROM 界讨论: 第 5.1.2 节.
- 安全模型、规约与界: 第 5.1.1 节、定理 3 及第 5.1.2 节.
- 量子模型安全性分析: 第 5.1 节.
- 实践安全性方法: 第 5.1.4 节.
- 各实例已知经典与量子攻击的复杂度: 第 5.1.4 节表 3; 第 4 节表 2; 以及附录 A.1.
- 可复现安全估计脚本: 附录 A.1.
- 解密失败攻击: 第 5.2.1 节.
- 侧信道攻击及常数时间/掩蔽考虑: 第 5.2.2 节.
- 4. 性能评估
 - 性能分析: 第 6.1 和 6.2 节.
 - 算术实现分析: 第 6.1 节.
 - 参考实现算术: 第 6.1 节.
 - AVX2 实现算术: 第 6.1 节.
 - 采样与扩展成本分析: 第 6.2 节.
 - 根式环乘积成本分析: 第 6.2 节.
 - 操作级性能分析: 第 6.2 节.
 - 通信与存储成本分析: 第 6.2 节及表 2.
 - 所有算法实例在主流 64 位 PC 处理器上的性能测试结果: 第 6.3 节.
 - 实现平台的详细配置: 第 6.3 节“实验设置”. 文档给出 CPU、时钟频率、指令集、操作系统、编译器、实现语言和基准方法.
 - PKE 性能结果: 第 6.3 节, 表 4.
 - KEM 性能结果: 第 6.3 节, 表 5.
 - 算术分解结果: 第 6.3 节, 表 6 和 7.
 - 与 Kyber AVX2 实现的比较: 第 6.3 节, 表 8.
- 5. 特性声明
 - 特性声明, 包括创新性、简洁性、灵活性、兼容性、可扩展性及不同平台上的性能优势: 第 1 节.

References

- [ABC⁺23] Aikata, Andrea Basso, Gaëtan Cassiers, Ahmet Can Mert, and Sujoy Sinha Roy. Kavach: Lightweight masking techniques for polynomial arithmetic in lattice-based cryptography. *IACR TCHES*, 2023(3):366–390, 2023.
[doi:10.46586/tches.v2023.i3.366-390](https://doi.org/10.46586/tches.v2023.i3.366-390).
- [ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange – a new hope. In *Proceedings of the 25th USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.
- [ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332

- of *LNCS*, pages 83–107. Springer, Berlin, Heidelberg, April / May 2002.
doi:[10.1007/3-540-46035-7_6](https://doi.org/10.1007/3-540-46035-7_6).
- [AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Berlin, Heidelberg, August 2013.
doi:[10.1007/978-3-642-40041-4_4](https://doi.org/10.1007/978-3-642-40041-4_4).
- [APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
doi:[doi:10.1515/jmc-2015-0016](https://doi.org/10.1515/jmc-2015-0016).
- [APUPG17] Troncoso-Pastoriza Alberto Pedrouzo-Ulloa, Juan Ramón and Fernando Pérez-González. Multivariate cryptosystems for secure processing of multidimensional signals. *ArXiv*, abs/1712.00848, 2017.
- [BBE⁺18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 354–384. Springer, Cham, April / May 2018. doi:[10.1007/978-3-319-78375-8_12](https://doi.org/10.1007/978-3-319-78375-8_12).
- [BCV20] Carl Bootland, Wouter Castryck, and Frederik Vercauteren. On the security of the multivariate ring learning with errors problem. In *ANTS-XIV*, volume 4, pages 57–71, 2020. doi:[10.2140/obs.2020.4.57](https://doi.org/10.2140/obs.2020.4.57).
- [BDK⁺21] Michiel Van Beirendonck, Jan-Pieter D’ anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede. A side-channel-resistant implementation of SABER. *J. Emerg. Technol. Comput. Syst.*, 17(2), 2021.
doi:[10.1145/3429983](https://doi.org/10.1145/3429983).
- [BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO’98*, volume 1462 of *LNCS*, pages 26–45. Springer, Berlin, Heidelberg, August 1998. doi:[10.1007/BFb0055718](https://doi.org/10.1007/BFb0055718).
- [BGM⁺16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 209–224. Springer, Berlin, Heidelberg, January 2016.
doi:[10.1007/978-3-662-49096-9_9](https://doi.org/10.1007/978-3-662-49096-9_9).
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.
doi:[10.1145/2090236.2090262](https://doi.org/10.1145/2090236.2090262).

- [BHH⁺19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Cham, December 2019. doi:[10.1007/978-3-030-36033-7_3](https://doi.org/10.1007/978-3-030-36033-7_3).
- [BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012. doi:[10.1007/978-3-642-29011-4_42](https://doi.org/10.1007/978-3-642-29011-4_42).
- [BRvV22] Joppe W. Bos, Joost Renes, and Christine van Vredendaal. Post-quantum cryptography with contemporary co-processors: Beyond Kronecker, Schönhage-Strassen & Nussbaumer. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3683–3697, Boston, MA, 2022. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity22/presentation/bos>.
- [BRVW26] Joppe W. Bos, Joost Renes, Frederik Vercauteren, and Peng Wang. Structured module lattice-based cryptography: Fine-grained parameter selection without compromising performance. *IACR Transactions on Cryptographic Hardware and Embedded Systems (to appear)*, 2026(3), 2026. URL: <https://eprint.iacr.org/2026/098>.
- [CCHY24] Jung Hee Cheon, Hyeongmin Choe, Dongyeon Hong, and MinJune Yi. SMAUG: Pushing lattice-based key encapsulation mechanisms to the limits. In Claude Carlet, Kalikinkar Mandal, and Vincent Rijmen, editors, *SAC 2023*, volume 14201 of *LNCS*, pages 127–146. Springer, Cham, August 2024. doi:[10.1007/978-3-031-53368-6_7](https://doi.org/10.1007/978-3-031-53368-6_7).
- [CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between Boolean and arithmetic masking of any order. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 188–205. Springer, Berlin, Heidelberg, September 2014. doi:[10.1007/978-3-662-44709-3_11](https://doi.org/10.1007/978-3-662-44709-3_11).
- [CHK⁺21] Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. NTT multiplication for NTT-unfriendly rings. *IACR TCHES*, 2021(2):159–188, 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8791>, doi:[10.46586/tches.v2021.i2.159-188](https://doi.org/10.46586/tches.v2021.i2.159-188).
- [CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO’99*, volume 1666 of *LNCS*, pages 398–412. Springer, Berlin, Heidelberg, August 1999. doi:[10.1007/3-540-48405-1_26](https://doi.org/10.1007/3-540-48405-1_26).

- [CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Berlin, Heidelberg, December 2011. doi:[10.1007/978-3-642-25385-0_1](https://doi.org/10.1007/978-3-642-25385-0_1).
- [Coo66] S. Cook. *On the minimum computation time of functions*. PhD thesis, Harvard University, 1966.
- [CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.
- [DB22] Jan-Pieter D’Anvers and Senne Batsleer. Multitarget decryption failure attacks and their application to Saber and Kyber. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 3–33. Springer, Cham, March 2022. doi:[10.1007/978-3-030-97121-2_1](https://doi.org/10.1007/978-3-030-97121-2_1).
- [DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Cham, August 2019. doi:[10.1007/978-3-030-26951-7_13](https://doi.org/10.1007/978-3-030-26951-7_13).
- [DFMS21] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. Cryptology ePrint Archive, Report 2021/280, 2021. URL: <https://eprint.iacr.org/2021/280>.
- [DGJ⁺19] Jan-Pieter D’Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. Decryption failure attacks on IND-CCA secure lattice-based schemes. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 565–598. Springer, Cham, April 2019. doi:[10.1007/978-3-030-17259-6_19](https://doi.org/10.1007/978-3-030-17259-6_19).
- [DKL⁺18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018. URL: <https://tches.iacr.org/index.php/TCHES/article/view/839>, doi:[10.13154/tches.v2018.i1.238-268](https://doi.org/10.13154/tches.v2018.i1.238-268).
- [DKR⁺20] Jan-Pieter D’Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>.
- [DKRV18] Jan-Pieter D’Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine

- Rachidi, editors, *AFRICACRYPT 18*, volume 10831 of *LNCS*, pages 282–305. Springer, Cham, May 2018. doi:[10.1007/978-3-319-89339-6_16](https://doi.org/10.1007/978-3-319-89339-6_16).
- [DRV20] Jan-Pieter D’Anvers, Mélissa Rossi, and Fernando Virdia. (One) failure is not an option: Bootstrapping the search for failures in lattice-based encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 3–33. Springer, Cham, May 2020. doi:[10.1007/978-3-030-45727-3_1](https://doi.org/10.1007/978-3-030-45727-3_1).
- [DVV21] Jan-Pieter D’Anvers, Ingrid Verbauwhede, and Frédérik Vercauteren. Design and security analysis of lattice-based post-quantum encryption, 2021.
- [FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO’99*, volume 1666 of *LNCS*, pages 537–554. Springer, Berlin, Heidelberg, August 1999. doi:[10.1007/3-540-48405-1_34](https://doi.org/10.1007/3-540-48405-1_34).
- [FVR⁺22] Tim Fritzmann, Michiel Van Beirendonck, Debapriya Basu Roy, Patrick Karl, Thomas Schamberger, Ingrid Verbauwhede, and Georg Sigl. Masked accelerators and instruction set extensions for post-quantum cryptography. *IACR TCHES*, 2022(1):414–460, 2022. doi:[10.46586/tches.v2022.i1.414-460](https://doi.org/10.46586/tches.v2022.i1.414-460).
- [HCCSS24] Jung Hee Cheon, Hyeongmin Choe, Jungjoo Seo, and Hyeon Seong. SMAUG(-T), revisited: Timing-secure, more compact, less failure. *IEEE Access*, 12:188386–188397, 2024. doi:[10.1109/ACCESS.2024.3511346](https://doi.org/10.1109/ACCESS.2024.3511346).
- [HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017. doi:[10.1007/978-3-319-70500-2_12](https://doi.org/10.1007/978-3-319-70500-2_12).
- [KDvB⁺22] Suparna Kundu, Jan-Pieter D’Anvers, Michiel van Beirendonck, Angshuman Karmakar, and Ingrid Verbauwhede. Higher-order masked Saber. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 93–116. Springer, Cham, September 2022. doi:[10.1007/978-3-031-14791-3_5](https://doi.org/10.1007/978-3-031-14791-3_5).
- [KKV23] Suparna Kundu, Angshuman Karmakar, and Ingrid Verbauwhede. On the masking-friendly designs for post-quantum cryptography. In Francesco Regazzoni, Bodhisatwa Mazumdar, and Sri Parameswaran, editors, *Security, Privacy, and Applied Cryptography Engineering - 13th International Conference, SPACE 2023, Roorkee, India, December 14-17, 2023, Proceedings*, volume 14412 of *Lecture Notes in Computer Science*, pages 162–184. Springer, 2023. doi:[10.1007/978-3-031-51583-5_10](https://doi.org/10.1007/978-3-031-51583-5_10).
- [KNK⁺25] Suparna Kundu, Quinten Norga, Angshuman Karmakar, Shreya Gangopadhyay, Jose Maria Bermudo Mera, and Ingrid Verbauwhede. Scabbard:

- An exploratory study on hardware aware design choices of learning with rounding-based key encapsulation mechanisms. *ACM Trans. Embed. Comput. Syst.*, 24(1):10:1–10:40, 2025. doi:[10.1145/3696208](https://doi.org/10.1145/3696208).
- [KO62] Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. *Doklady Akademii Nauk SSSR*, 145(2):293–294, 1962. URL: <https://www.mathnet.ru/eng/dan26729>.
- [Kro82] L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. *Journal für die reine und angewandte Mathematik*, 92:1–122, 1882.
- [KSS⁺20] Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 703–728. Springer, Cham, May 2020. doi:[10.1007/978-3-030-45727-3_24](https://doi.org/10.1007/978-3-030-45727-3_24).
- [Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22. Springer, Berlin, Heidelberg, August 2015. doi:[10.1007/978-3-662-47989-6_1](https://doi.org/10.1007/978-3-662-47989-6_1).
- [LMvdP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. *DCC*, 77(2-3):375–400, 2015. doi:[10.1007/s10623-015-0067-5](https://doi.org/10.1007/s10623-015-0067-5).
- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010. doi:[10.1007/978-3-642-13190-5_1](https://doi.org/10.1007/978-3-642-13190-5_1).
- [LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015. doi:[10.1007/s10623-014-9938-4](https://doi.org/10.1007/s10623-014-9938-4).
- [Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009. doi:[10.1007/978-3-642-10366-7_35](https://doi.org/10.1007/978-3-642-10366-7_35).
- [Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Berlin, Heidelberg, April 2012. doi:[10.1007/978-3-642-29011-4_43](https://doi.org/10.1007/978-3-642-29011-4_43).
- [MKKV21] Jose Maria Bermudo Mera, Angshuman Karmakar, Suparna Kundu, and Ingrid Verbauwhede. Scabbard: a suite of efficient learning with rounding key-encapsulation mechanisms. *IACR TCHES*, 2021(4):474–509, 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/9073>, doi:[10.46586/tches.v2021.i4.474-509](https://doi.org/10.46586/tches.v2021.i4.474-509).

- [MTK⁺20] Jose Maria Bermudo Mera, Furkan Turan, Angshuman Karmakar, Sujoy Sinha Roy, and Ingrid Verbauwhede. Compact domain-specific co-processor for accelerating module lattice-based KEM. In *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*, pages 1–6. IEEE, 2020. doi:[10.1109/DAC18072.2020.9218727](https://doi.org/10.1109/DAC18072.2020.9218727).
- [NR23] Parker Newton and Silas Richelson. A lower bound for proving hardness of learning with rounding with polynomial modulus. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 805–835. Springer, Cham, August 2023. doi:[10.1007/978-3-031-38554-4_26](https://doi.org/10.1007/978-3-031-38554-4_26).
- [OTF⁺20] Hiroki Okada, Atsushi Takayasu, Kazuhide Fukushima, Shinsaku Kiyomoto, and Tsuyoshi Takagi. A compact digital signature scheme based on the module-LWR problem. In Weizhi Meng, Dieter Gollmann, Christian Damsgaard Jensen, and Jianying Zhou, editors, *ICICS 20*, volume 11999 of *LNCS*, pages 73–90. Springer, Cham, August 2020. doi:[10.1007/978-3-030-61078-4_5](https://doi.org/10.1007/978-3-030-61078-4_5).
- [Pei16] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016. doi:[10.1561/04000000074](https://doi.org/10.1561/04000000074).
- [PJP⁺22] Seunghwan Park, Chi-Gon Jung, Aesun Park, Joongeun Choi, and Honggoo Kang. TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R). Cryptology ePrint Archive, Report 2022/1651, 2022. URL: <https://eprint.iacr.org/2022/1651>.
- [PN25] Rafaël Del Pino and Guilhem Niot. Finally! A compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 169–199. Springer, Cham, May 2025. doi:[10.1007/978-3-031-91826-1_6](https://doi.org/10.1007/978-3-031-91826-1_6).
- [Pol71] John M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.
- [PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, Berlin, Heidelberg, May 2013. doi:[10.1007/978-3-642-38348-9_9](https://doi.org/10.1007/978-3-642-38348-9_9).
- [PUTPG⁺21] Alberto Pedrouzo-Ulloa, Juan Ramón Troncoso-Pastoriza, Nicolas Gama, Mariya Georgieva, and Fernando Pérez-González. Revisiting multivariate ring learning with errors and its applications on lattice-based cryptography. *Mathematics*, 9(8), 2021. URL: <https://www.mdpi.com/2227-7390/9/8/858>, doi:[10.3390/math9080858](https://doi.org/10.3390/math9080858).
- [PUTPPG15] Alberto Pedrouzo-Ulloa, Juan Ramón Troncoso-Pastoriza, and Fernando Pérez-González. Multivariate lattices for encrypted image processing. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1707–1711, 2015. doi:[10.1109/ICASSP.2015.7178262](https://doi.org/10.1109/ICASSP.2015.7178262).

- [PUTPPG16] Alberto Pedrouzo-Ulloa, Troncoso-Pastoriza, and Fernando Pérez-González. On ring learning with errors over the tensor product of number fields, 2016. URL: <https://arxiv.org/abs/1607.05244>.
- [RCDB24] Prasanna Ravi, Anupam Chattopadhyay, Jan Pieter D’ Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (Kyber, Dilithium): Survey and new results. *ACM Trans. Embed. Comput. Syst.*, 23(2), March 2024. doi:10.1145/3603170.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603.
- [Reg10] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204, 2010. doi:10.1109/CCC.2010.26.
- [RRCB20] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. Generic side-channel attacks on cca-secure lattice-based PKE and kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):307–335, 2020. URL: <https://doi.org/10.13154/tches.v2020.i3.307-335>, doi:10.13154/TCHES.V2020.I3.307-335.
- [SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994. doi:10.1007/BF01581144.
- [Too63] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers, published in soviet math (translations of dokl. adad. nauk. sssr), 4, 1963.
- [XPR⁺22] Zhuang Xu, Owen Pemberton, Sujoy Sinha Roy, David Oswald, Wang Yao, and Zhiming Zheng. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of Kyber. *IEEE Transactions on Computers*, 71(9):2163–2176, 2022. doi:10.1109/TC.2021.3122997.