

# **AKCN-E8：新型格编码 及基于理想格的强安全、高效、灵活的密钥封装 算法说明书**

赵运磊（复旦大学）

刘哲（南京航空航天大学）

金正中（复旦大学）

巩博儒（复旦大学）

隋光烨（上海扈民区块链科技有限公司）

刘伟超（银联商务股份有限公司）

文黎明（银联商务股份有限公司）

2019 年 10 月

## 摘要

近年来数学上一项重要大突破是证明了一个长期存在的猜想：8 维 E8 格中的装球问题具有最优的密度 [29]。在本提案中，我们设计了基于 E8 的格编码，称为 AKCN-E8，用于纠错和非对称密钥共识（AKC）。AKCN-E8 编码具有独立的价值和意义，亦可应用至诸如 IEEE 802.11a WLAN 标准等通信协议。

基于非对称密钥共识构建密钥封装机制的模块化通用化框架，作为 AKCN-E8 编码的一个直接的应用，我们提出了一种基于理想格 RLWE 的实用的密钥封装机制（KEM）。与 NIST 后量子密码竞赛第二轮的基于 RLWE 的 NewHope-KEM 相比，我们的 AKCN-E8-KEM 有以下优势：

- (1) 共享密钥的长度增加了一倍，这对于后量子时代抵抗 Grover 类型量子算法和更高级的量子密码分析具有重要意义。
- (2) 可以在安全性、共享密钥尺寸、带宽、错误率等所有方面同时超越 NIST 第二轮的 NewHope-KEM 密钥封装算法。可以选取参数同时达到：双倍的共享密钥大小，更小的密文大小，更低的错误概率、和更强的安全性。
- (3) 更灵活的参数选择，用于安全性、密文大小和错误率之间的权衡。可以根据安全性优先、带宽优先等不同应用场景，灵活配置参数。
- (4) NewHope-1024 的错误率并不能与其标准安全性匹配，而 AKCN-E8 的所有参数对应的错误率均既匹配后量子安全性也匹配经典安全性。

更一般地，AKCN-E8-KEM 密钥封装机制还具有如下特色

- (1) 计算高效。
- (2) 安全冗余度高：无论是所基于的 RLWE 问题难度还是共享密钥尺寸都有很高的安全冗余度。
- (3) 模块化设计与分析。

AKCN-E8-KEM 的参数和性能以及与 NewHope-KEM 的比较如下表所示，其中  $|K|$  指的是共享密钥的长度， $\text{pk}(B)$  和  $\text{cipher}(B)$  对应公钥和密文的字节数， $\text{err}$  是错误率， $\text{pq-sec}$  是以比特数衡量的 RLWE 后量子难度值， $\text{c-sec}$  对应经典难度。

	$ K $	$n$	$q$	$\eta$	$g$	$t$	c-sec	pq-sec	err	pk (B)	cipher (B)
NewHope-512-CPA	128	512	12289	8	$2^3$	0	112	101	$2^{-213}$	928	1088
AKCN-E8-512-S-CPA	256	512	12289	14	$2^4$	3	121	110	$2^{-224}$	928	960
AKCN-E8-512-E-CPA	256	512	12289	8	$2^4$	4	112	101	$2^{-256}$	928	896
AKCN-E8-512-C-CPA	256	512	12289	8	$2^3$	4	112	101	$2^{-150}$	928	832
NewHope-512-CCA	128	512	12289	8	$2^3$	0	112	101	$2^{-213}$	928	1120
AKCN-E8-512-S-CCA	256	512	12289	14	$2^4$	3	121	110	$2^{-224}$	928	992
AKCN-E8-512-E-CCA	256	512	12289	8	$2^4$	4	112	101	$2^{-256}$	928	928
AKCN-E8-512-C-CCA	256	512	12289	8	$2^3$	4	112	101	$2^{-150}$	928	864
NewHope-1024-CPA	256	1024	12289	8	$2^3$	0	257	233	$2^{-216}$	1824	2176
AKCN-E8-1024-S-CPA	512	1024	12289	10	$2^4$	2	265	240	$2^{-274}$	1824	2048
AKCN-E8-1024-E-CPA	512	1024	12289	8	$2^4$	3	257	233	$2^{-280}$	1824	1920
AKCN-E8-1024-C-CPA	512	1024	12289	4	$2^3$	3	236	214	$2^{-500}$	1824	1792
NewHope-1024-CCA	256	1024	12289	8	$2^3$	0	257	233	$2^{-216}$	1824	2208
AKCN-E8-1024-S-CCA	512	1024	12289	10	$2^4$	2	265	240	$2^{-274}$	1824	2080
AKCN-E8-1024-E-CCA	512	1024	12289	8	$2^4$	3	257	233	$2^{-280}$	1824	1952
AKCN-E8-1024-C-CCA	512	1024	12289	4	$2^3$	3	236	214	$2^{-500}$	1824	1824

理想格 RLWE 相对于一般格 LWE 由于具有更丰富的代数结构，而这些代数结构不排除未来会被（特别是量子算法）利用从而带来相对较多的安全隐患。这个不足我们通过充足的安全冗余和翻倍的密钥长度来加以克服。相对于目标为封装 266 比特 128 比特后量子安全级别的格密码方案，AKCN-E8 充足的安全冗余和 512 比特的封装密钥导致带宽适度增大（但是相对于 NewHope-KEM，带宽显著降低）。但是，后量子密码主要是针对未来网络环境（5G 甚至 6G），在新一代网络通讯环境中，适度增加的带宽的影响会降低，反而充足的安全冗余和翻倍的密钥封装能力在未来网络和计算环境带来的优势更明显。

# 目 录

1	引言 .....	1
2	预备知识.....	2
2.1	密钥封装机制 (KEM) .....	3
2.2	公钥加密 (PKE) .....	3
2.3	LWE, Ring-LWE 问题 .....	4
3	设计理念：基于 RLWE 构建 PKE/KEM 的模块化和通用性框架 .....	6
3.1	基础构件：非对称密钥共识 AKC.....	6
3.2	基于 AKC 构造 CPA 安全的 PKE.....	7
3.3	量子预言机 QROM 模型下 CPA-PKE 转化为 CCA-KEM.....	8
4	核心工具：基于 E8 格编码的 AKCN-E8 非对称密钥共识机制的设计与分析 .....	9
4.1	错误率分析.....	13
5	算法模块化完整描述、参数选取及性能分析.....	13
5.1	参数选取、性能分析和测试.....	14
5.2	算法模块化完整描述、与实现说明.....	17
5.2.1	CPA-安全的 AKCN-E8-KEM 算法描述和实现 .....	17
5.2.2	CCA-安全的 AKCN-E8-KEM 算法描述和实现 .....	19
6	算法创新性和特色、优缺点说明.....	21
6.1	AKCN-E8-KEM 创新性 .....	21
6.2	AKCN-E8-KEM 特色 .....	21
6.3	AKCN-E8-KEM 优缺点 .....	22
7	适配性说明.....	22
8	算法第二轮修改说明.....	23
9	支持文档.....	23
10	参考文献.....	23

# 1 引言

量子计算的发展刺激了新的抗量子分析公钥密码系统的发展,统称为后量子密码。其中一类很有前途的后量子密码系统是基于格的密码,这引出了基于LWE(Learning with Errors)问题的密钥封装机制(KEM)[20]。在密码学的环境下,和其他古典的格困难问题(例如SVP和CVP)相比,LWE问题已经被证明功能更加全面[25]。然而,基于LWE的密码系统通常效率不高,后来通过从理想格引入RLWE来解决[18]。在基于RLWE的非对称密码系统中,NewHope-KEM[23]是领先的KEM方案之一,它是NewHope-Usenix[1](2016年互联网防御奖获得者)的变体,现在已经进入NIST后量子密码竞赛第二轮。

在本提案中,我们回顾了[15,16]中明确提出的用于实现和分析基于LWE及其变体(包括RLWE)的KEM机制的模块化和通用化的框架。这个模块化和通用化的框架使我们把重点放在从LWE及其变体实现KEMs的一个关键构建块上,这就是所谓的非对称密钥协商(AKC)。在这个框架中,NewHope-KEM的底层AKC机制是 $\mathbf{Z}_4$ 中的一个格编码,它将一个密钥位编码成四个多项式系数。在本提案中,我们设计了基于E8的格编码,称为AKCN-E8,用于纠错和非对称密钥共识,它将四个密钥位编码为八个系数。作为AKCN-E8码的直接应用,我们提出了基于RLWE的高度实用的KEM方案,称为AKCN-E-KEM。与NewHope KEM[23]相比,我们的AKCN-E8-KEM具有以下优势:

- (1) 共享密钥的长度增加了一倍,这从长远来看对于针对Grover搜索算法的目标安全级别和更复杂的量子密码分析的可能性都很重要。
- (2) 在相同甚至更高的安全级别下有更紧凑的密文。
- (3) 更灵活的参数选择,以便安全性、密文大小和错误率之间的权衡。
- (4) 特别是,相比于NewHope-KEM,AKCN-E8可以同时具有如下性能优势:双倍的共享密钥大小,更小的密文大小,更低的错误概率、和更强的安全性。

AKCN-E8的性能优势以及参数选择的灵活性主要来自于基于E8的格编码,其比NewHope-KEM使用的基础 $\mathbf{Z}_4$ 一般格编码密度更高、纠错能力更强[23],

[24]。实际上，近年来数学上的一个重大突破是证明了长期存在的猜想：基于 E8 的装球问题被证明是最优的[29]。

Leech 格在 24 维装球问题中被证明也是密度最高的[4]，并且已经用于通信协议中的纠错 [5, 28]，例如在 IEEE 802.11a WLAN 标准（<https://standards.ieee.org/standard/80211-2016.html>）中。一方面，它的编码和解码比我们的 AKCN-E8 代码更复杂，效率更低。另一方面，更重要的是，Leech 格编码不适应基于 RLWE 的密码系统构造。具体而言，很难找到 RLWE 的参数[22]，因为 Leech 格是 24 维的格。对于基于 RLWE 的密码系统，我们通常使用 NTT 算法来加速多项式乘法。当 RLWE 的维数为 2 的幂时，NTT 算法可以充分优化计算资源。但是，我们并不能将参数  $n$  同时设置为 2 的幂和 24 的倍数。设置 Leech 格的密钥长度时也会出现同样的问题，因为密钥大小通常是 12 的倍数。相比之下，E8 格没有上述问题。从这个意义上讲，我们相信我们的 AKCN-E8 代码不仅可应用于基于 RLWE 的公钥加密，亦可以在更广泛的通信协议中找到应用。

## 2 预备知识

在本文，字符串或者值  $\alpha$  都以二进制表示， $|\alpha|$  表示  $\alpha$  二进制的长度。对于任意实数  $x$ ， $\lfloor x \rfloor$  表示小于等于  $x$  的最大整数， $\lfloor x \rfloor = \lfloor x + 1/2 \rfloor$ 。对于任意的正整数  $a$  和  $b$ ，用  $\text{lcm}(a, b)$  表示  $a$  和  $b$  的最小公倍数。对于任意的  $i, j \in \mathbb{Z}$ ，并且  $i < j$ ，用  $[i, j]$  表示整数集合  $\{i, i+1, \dots, j-1, j\}$ 。对于任意的正整数  $t$ ，令  $\mathbb{Z}_t$  表示  $\mathbb{Z} / t\mathbb{Z}$ 。 $\mathbb{Z}_t$  中的元素默认表示为  $[0, t-1]$ ，但有时  $\mathbb{Z}_t$  会明确表示为  $[-\lfloor (t-1)/2 \rfloor, \lfloor t/2 \rfloor]$ 。

如果  $S$  是一个有限集合，那么  $|S|$  表示它的基数，并且  $x \leftarrow S$  表示均匀随机的从  $S$  中取一个元素。对于两个集合  $A, B \subseteq \mathbb{Z}_q$ ，我们定义  $A + B \triangleq \{a + b \mid a \in A, b \in B\}$ 。对于一个加法群  $(G, +)$ ，元素  $x \in G$  并且子集  $S \subseteq G$ ， $x + S$  表示将  $S$  中每一个元素都和  $x$  相加结果的集合。对于一个集合  $S$ ，用  $\mathcal{U}(S)$  表示  $S$  的一个均匀分布。对于任意的  $\mathbb{R}$  中的离散随机变量  $X$ ， $\text{Supp}(X) = \{x \in \mathbb{R} \mid \Pr[X = x] > 0\}$ 。

在后面的概率相关的算法、实验和交互协议当中，我们使用传统的符号和概念。如果  $\mathcal{D}$  表示一个概率分布，那么  $x \leftarrow \mathcal{D}$  表示根据  $\mathcal{D}$  选择一个元素并赋值给  $x$ 。

如果 $\alpha$ 既不是一个算法也不是一个集合,那么 $x \leftarrow \alpha$ 就表示简单的赋值操作。如果 $A$ 是一个概率算法,那么 $A(x_1, x_2, \dots; r)$ 表示将 $x_1, x_2, \dots$ 作为输入,  $r$ 为随机种子 $A$ 的运算结果。我们用 $y \leftarrow A(x_1, x_2, \dots)$ 表示随机选取 $r$ 并令 $y$ 为 $A(x_1, x_2, \dots; r)$ 的实验。用 $\Pr[R_1; \dots; R_n: E]$ 表示事件 $E$ 在一连串有序的随机过程 $R_1, \dots, R_n$ 之后发生的概率。

如果对于任意的 $c > 0$ ,对于所有的 $\lambda > \lambda_c$ ,都存在一个 $\lambda_c$ 使得 $f(\lambda) < 1/\lambda^c$ ,那么函数 $f(\lambda)$ 是可忽略的。

## 2.1 密钥封装机制 (KEM)

我们回顾一下文献[7, 12]中关于 KEM 的定义。一个密钥封装机制 $\text{KEM} = (\text{KeyGen}, \text{Encaps}, \text{Decaps})$ 包括了三种算法。输入安全参数 $\kappa$ , 密钥生成算法 $\text{KeyGen}$ 输出密钥对 $(pk, sk)$ ,  $pk$ 也定义了一个有限的密钥空间 $\mathcal{K}$ 。而封装算法 $\text{Encaps}$ 在输入 $pk$ 时输出二元组 $(K, c)$ , 其中 $c$ 是密钥 $K$ 的一个封装, 而 $K$ 则包含在密钥空间 $\mathcal{K}$ 中。确定的解封装算法 $\text{Decaps}$ 在输入 $sk$ 和一个密钥封装 $c$ 后, 输出密钥 $K := \text{Decaps}(sk, c) \in \mathcal{K}$ , 或符号 $\perp \notin \mathcal{K}$ 以表示 $c$ 不是一个有效的密钥封装。

如果  $\Pr[\text{Decaps}(sk, c) \neq K | (pk, sk) \leftarrow \text{KeyGen}(1^\kappa); (K, c) \leftarrow \text{Encaps}(pk)] \leq \delta$ , 那么我们称KEM是 $\delta$ 近似正确的。

<b>GAME</b>	<b>IND-CCA</b>	<b>DECAPS(<math>c \neq c^*</math>)</b>
$(pk, sk) \leftarrow \text{Gen}$		$K := \text{Decaps}(sk, c)$
$b \xleftarrow{\$} \{0, 1\}$		<b>return</b> $K$
$(K_0^*, c^*) \leftarrow \text{Encaps}(pk)$		
$K_1^* \xleftarrow{\$} \mathcal{K}$		
$b' \leftarrow \text{A}^{\text{DECAPS}}(c^*, K_b^*)$		
<b>return</b> $[b' = b]$		

图 2.1 KEM 的 CCA 游戏

在选择密文攻击 (CCA) 下的安全概念不可区分性定义如图 2.1。对任意概率多项式时间的敌手  $\text{A} =$ , 定义它的 CCA 优势为  $\text{Adv}_{\text{KEM}}^{\text{CCA}}(\text{A}) := |\Pr[\text{GAME CCA outputs } 1] - 1/2|$ 。我们说KEM机制是 CCA 安全的, 如果对任意的安全参数和任意概率多项式时间的敌手  $\text{A} =$ ,  $\text{Adv}_{\text{KEM}}^{\text{CCA}}(\text{A})$ 是可忽略的。

## 2.2 公钥加密 (PKE)

我们回顾文献[11,12]中关于 PKE 的定义，一个公钥加密机制可由算法构成的三元组来给出， $PKE = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ ，对任意大的  $\kappa \in \mathbb{N}$ 。

- (1) **KeyGen**: 密钥生成算法，是一个概率多项式时间（在  $\kappa$  中）算法，输入  $1^\kappa$  并输出一对字符串  $(pk, sk)$ ， $(pk, sk)$  分别被称为公钥和私钥。这个实验可以写作  $(pk, sk) \leftarrow \text{KeyGen}(1^\kappa)$ 。
- (2)  $\mathcal{E}$ : 加密算法，是一个概率多项式时间（在  $\kappa$  中）算法，其从信息空间  $MSP$  取公钥  $pk$  和消息  $M$ ，从硬币空间  $COIN$  均匀取硬币  $r$ ，并产生密文  $C := \mathcal{E}_{pk}(M; r)$ 。这个实验被写作为  $C \leftarrow \mathcal{E}_{pk}(x)$ 。
- (3)  $\mathcal{D}$ , 解密算法，是一个确定性的多项式时间（在  $\kappa$  中）算法，输入密钥  $sk$  和密文  $C \leftarrow \{0,1\}^*$ ，并返回消息  $M \in MSP$ 。

我们认为一个 PKE 机制是  $\delta$  正确的，如果对任意大的  $\kappa \in \mathbb{N}$ ，每对  $(pk, sk)$  都由  $\text{KeyGen}(1^\kappa)$  生成，且每个  $M \in MSP$ ，我们总是有  $E[\max_{M \in MSP} \Pr[\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) \neq M]] \leq \delta$ 。

**定义 2.1 (CCA 安全)** 令  $PKE = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  是一个非对称加密机制， $A = (A_1, A_2)$  是对 PKE 方案的一个敌手。对任意  $\kappa \in \mathbb{N}$ ，定义如下 CCA 优势：

$$\begin{aligned} Adv_A^{CCA}(\kappa) &= 2 \cdot \Pr[(pk, sk) \leftarrow \text{KeyGen}(1^\kappa); (M_0, M_1, st) \leftarrow A_1^{\mathcal{D}_{sk}}(pk); \\ &\quad b \leftarrow \{0,1\}; C^* \leftarrow \mathcal{E}_{pk}(M_b); A_2^{\mathcal{D}_{sk}}(C^*, st) = b] - 1 \end{aligned}$$

我们就说这个 PKE 机制是 CCA 安全的，如果对任意足够大的安全参数  $\kappa$  和概率多项式时间敌手  $A$ ，它的 CCA 优势  $Adv_A^{CCA}$  在  $\kappa$  上是可忽略的。同时如果当  $A$  在无法访问解密预言机  $\mathcal{D}_{sk}$  的情况下优势可以忽略，那么称这个 PKE 方案是抗选择明文攻击（简称 CPA 安全）。

## 2.3 LWE, Ring-LWE 问题

给定正连续数  $\sigma > 0$ ，对于  $x \in \mathbb{R}$ ，定义高斯函数  $\rho_\sigma(x) \triangleq \exp(-x^2/2\sigma^2)/\sqrt{2\pi\sigma^2}$ 。令  $D_{\mathbb{Z},\sigma}$  表示在  $\mathbb{Z}$  上的一维离散高斯分布，此由其概率密度函数  $D_{\mathbb{Z},\sigma}(x) \propto \rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ ， $x \in \mathbb{Z}$  决定。最后，令  $D_{\mathbb{Z}^n,\sigma}$  表示在  $\mathbb{Z}^n$  上的  $n$  维球面离散高

斯分布，其中每个坐标都独立于  $D_{\square, \sigma}$ 。

给定正整数  $n$  和  $q$ ，它们都是安全参数  $\lambda$  中的多项式中的参数，并给定整数向量  $\mathbf{s} \in \square_q^n$  和一个定义在  $\mathbb{Z}_q$  上的概率分布  $\chi$ ，通过随机均匀选择  $\mathbf{a} \in \square_q^n$ ，令  $A_{q, \mathbf{s}, \chi}$  是  $\mathbb{Z}_q^n \times \mathbb{Z}_q$  上的分布，误差  $e \leftarrow \chi$ ，并输出  $(\mathbf{a}, b = \mathbf{a}^T \mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ 。误差分布  $\chi$  通常被认为是离散高斯概率分布  $D_{\mathbb{Z}, \sigma}$ ；但是，如文献[3]中所述，也可以采用其他的  $\chi$  分布。简而言之，在判定型 LWE [25] 假设中，对足够大的安全参数  $\lambda$ ，概率多项式时间算法无法以不可忽略的概率来区分  $A_{q, \mathbf{s}, \chi}$  和  $\mathbb{Z}_q^n \times \mathbb{Z}_q$  上的均匀分布。即使  $A$  看到多项式多个样本，且秘密向量  $\mathbf{s}$  是从  $\chi^n$  随机选取的，这也是成立的[2]。

对于安全参数  $\lambda$  中为多项式的正整数  $m$ ，定义  $n \triangleq \varphi(m)$  为欧拉函数值。同时定义  $K \subseteq \square(\zeta_m)$  为数域上通过毗邻一个抽象元素  $\zeta_m$  满足  $\Phi_m(\zeta_m) = 0$ ，其中  $\Phi_m(x) \in \square[x]$  是  $m$  阶  $n$  次环多项式。并且，定义  $R \subseteq O_K$  为  $K$  上的整数环。最后，给定一个正素数  $q = \text{poly}(\lambda)$ ，满足  $q \equiv 1 \pmod{m}$ ，定义商环  $R_q \subseteq R / qR$ 。

我们简要回顾 RLWE 问题以及它的困难性[8], [18, 19]。在这项工作中我们关注了定义在文献[18]的 RLWE 问题。给定  $n \geq 16$  且为 2 的幂， $q = \text{poly}(\lambda)$  为正素数，满足  $q \equiv 1 \pmod{2n}$ 。给定  $s \leftarrow R_q$ ，一个定义在  $R_q \times R_q$  上的 RLWE 分布  $A_{n, q, \sigma, s}$  且由  $\mathbf{a} \leftarrow R_q, e \leftarrow D_{\square^n, \sigma}$  选定产生，输出  $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e) \in R_q \times R_q$ 。大致上说，（决定性）RLWE 假设描述了，对于足够大的安全参数  $\lambda$ ，不存在概率多项式算法  $A$  可以在不可忽略概率下区分基于  $R_q \times R_q$  上的均匀分布和  $A_{n, q, \sigma, s}$ 。即使在  $A$  观察多项次数量的样本且密文  $s$  从同样的误差多项式  $e$  随机选取，这个结论仍旧正确 [2, 8]。并且，正如文献[1]所说，误差多项式的替代分布可以在不削弱安全性的情况下保持效率。

最近，文献[21]提出了一种将最坏理想格点问题直接归结为环-LWE 决策问题的多项式时间（量子）约化方法。特别地，在任意模和任意数域约化方法同样有效。除了 RLWE 问题的特殊版本[18]，另一个 RLWE 问题版本在多项式环



$R_n = \mathbb{Z}[x]/\Phi_{n+1}(x)$  上定义，其中  $n+1$  是安全素数， $\Phi_{n+1}(x) = x^n + x^{n-1} + \dots + x + 1$  是  $(n+1)$  阶环多项式。对于这个环， $n$  有一个大的取值范围从中选择。

### 3 设计理念：基于 RLWE 构建 PKE/KEM 的模块化和通用性框架

#### 3.1 基础构件：非对称密钥共识 AKC

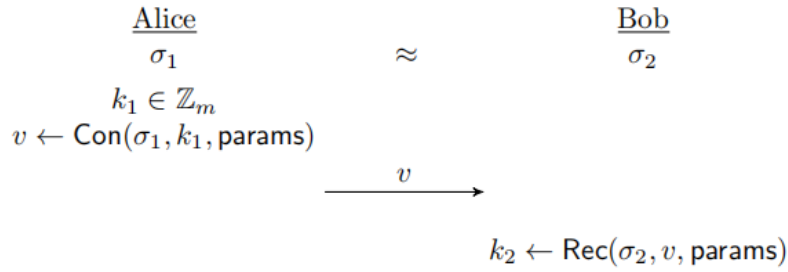


图 2 AKC 的描述

在介绍非对称密钥共识（AKC）方案的定义之前，我们首先介绍一个相对于正整数  $q \geq 1$  新的函数  $|\cdot|_q$ ：  $|x|_q = \min\{x \bmod q, q - x \bmod q\}$ ，  $\forall x \in \mathbb{Z}$ ，其中模块化操作的结果在  $\{0, \dots, (q-1)\}$  中。比如，  $|-1|_q = \min\{-1 \bmod q, (q+1) \bmod q\} = \min\{q-1, 1\} = 1$ 。对于任意的  $X = (x_0, x_1, x_2, \dots, x_{\mu-1})^T \in \mathbb{Z}_q^\mu$ ，其中  $\mu$  是正整数。用  $\|X\|_{q,1}$  表示  $|x_0|_q + |x_1|_q + \dots + |x_{\mu-1}|_q$  的和。

**定义 3.1** 一个非对称密钥共识算法  $AKC = (\text{params}, \text{Con}, \text{Rec})$ ，具体描述如下：

- (1)  $\text{params} = (q, m, g, d, \text{aux})$  表示系统参数，其中  $q, 2 \leq m, g \leq q, 1 \leq d \leq \left\lfloor \frac{q}{2} \right\rfloor$  均

为正整数，其中  $\text{aux}$  表示辅助信息，通常由  $(q, m, g, d)$  确定，其值可以设定为空；

- (2)  $v \leftarrow \text{Con}(\sigma_1, k_1, \text{params})$ ：在输入为  $(\sigma_1 \in \mathbb{Z}_q^u, k_1 \in \mathbb{Z}_m^{u'}, \text{params})$  的条件下，

其中  $\mu$  是正整数，多项式时间调和算法  $\text{Con}$  的输出为  $v \in \square_g$ ， $v$  为公共提示信息；

- (3)  $k_2 \leftarrow \text{Rec}(\sigma_2, v, \text{params})$ ：在输入为  $(\sigma_2 \in Z_q^u, v \in Z_g^u, \text{params})$  的情况下，确定的多项式时间算法  $\text{Rec}$  的输出为  $k_2 \in \square_m^{u'}$ 。

**正确性：**如果一个  $\text{AKC}$  算法对于任意的  $\sigma_1, \sigma_2 \in \square_q^u$  且  $\|\sigma_1 - \sigma_2\|_{q,1} \leq d$ ，都有  $k_1 = k_2$ ，那么其满足正确性

**安全性：**如果  $k_1$  和  $v$  是相互独立的，并且  $\sigma_1$  在  $\square_q^u$  上是均匀分布的，那么  $\text{AKC}$  算法满足安全性。具体来说，对于任意的  $\tilde{v} \in \square_g^u$  和任意的  $\tilde{k}_1, \tilde{k}_1' \in \square_m^{u'}$  满足  $\Pr[v = \tilde{v} | k_1 = \tilde{k}_1] = \Pr[v = \tilde{v} | k_1 = \tilde{k}_1']$ ，其中的概率来源于  $\sigma_1 \in \square_q^u$  和  $\text{Con}$  中使用的随机种子。

### 3.2 基于 $\text{AKC}$ 构造 $\text{CPA}$ 安全的 $\text{PKE}$

令  $(\lambda, n, q, \sigma, \text{AKC})$  表示系统参数，其中  $\lambda$  是安全参数， $q \geq 2$  是一个正素数， $\sigma$  是离散高斯分布  $D_{\square^n, \sigma}$  的参数， $n$  表示  $R_q$  上多项式的次数，为简单起见，我们假设  $\mu | n$ ，并且  $\text{Gen}$  是一个由小种子  $\text{seed} \leftarrow \{0,1\}^K$  生成  $a \in R_q$  的伪随机生成器 (PRG)。假设  $\text{AKC} = (\text{params}, \text{Con}, \text{Rec})$  是一个正确且安全的  $\text{AKC}$  方案，其中  $\text{params} = (q, g, m, d)$ 。在此文章中，我们主要考虑  $m = 2$  的情况。RLWE 基于  $\text{AKC}$  的  $\text{PKE}$  于图 3 中进行描述。这里， $(\text{seed}, y_1)$  是公钥， $(y_2, v)$  是密文。在协议的描述中，为了简化表示，将  $\text{Con}$  和  $\text{Rec}$  函数应用于多项式，表示它们分别应用于每组  $\mu$  个系数。NewHope 中  $\mu = 4$ ， $u' = 1$ ，而 AKCN-E8 中  $\mu = 8$ ， $u' = 4$ 。为简便表示，我们仍将  $k_1 = k_2$  记作为共享密钥：在  $\text{PKE}$  中它们对应任意明文空间的明文，在  $\text{KEM}$  中则对应封装的随机密钥。

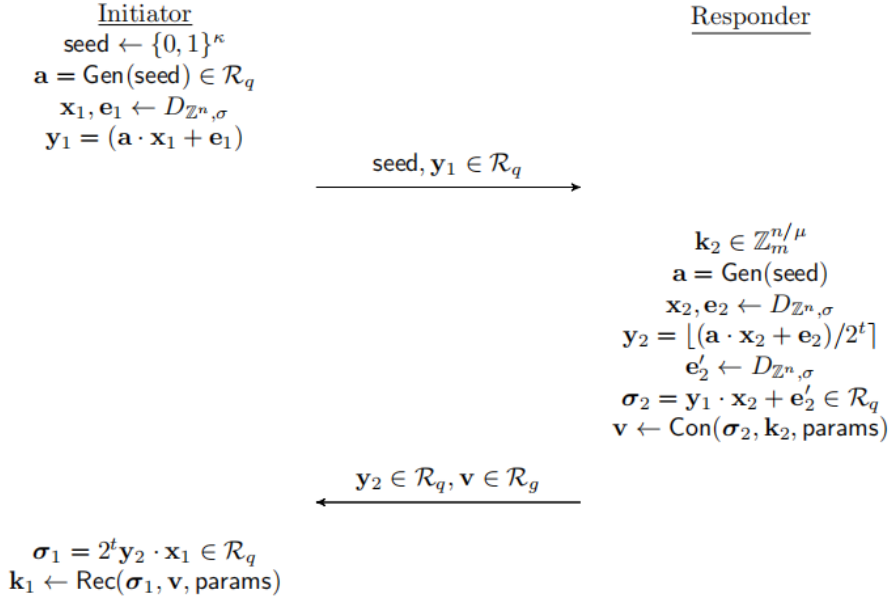


图 3 AKC 的基于 RLWE 的 CPA 安全的 PKE 描述

众所周知，假设（1）基础 AKC 方案是正确又安全的（2）（判定）RLWE 是困难的，那么上述 PKE 的模块化结构是 CPA 安全的[14-18, 3, 25]。通过明确定义和研究基础构建工具 AKC，Jin 和 Zhao 在[15]明确提出了来自 LWE 及其变体的 CPA 安全 PKE 的上述模块化和通用化框架。以前所有的工作都以非黑盒方式隐式使用 AKC。通常，抽象和泛化是自然科学（如数学、物理）的基础，对于密码学尤其重要。例如，在签名领域中，Schnorr 是通过 Fiat-Shamir 变换进行推广的[9]，并采用  $\Sigma$  协议的抽象[6]。类似的抽象和泛化也在 CCA 安全的 PKE 以及现代密码学的许多领域中起着重要作用。抽象和泛化对基于格的密码学非常有用，因为它们通常不易理解，且与正在进行的 NIST 后量子密码标准化有关。

### 3.3 量子预言机 QROM 模型下 CPA-PKE 转化为 CCA-KEM

目前已经提出了许多由 CPA 安全的 PKE 转化为 CCA 安全的 KEM 的方法[10-14, 27]，且在量子随机预言模型（QROM）中进行了具体的安全评估。在此文中，为了简化表示和便于比较，我们使用与 NewHope KEM 相同的 CCA 转换方法，有关详细信息，请参阅[23]。

## 4 核心工具：基于 E8 格编码的 AKCN-E8 非对称密钥共识机制的设计与分析

根据上述基于 RLWE 的 CPA 和 CCA 安全的 KEM 的 AKC 上的模块化和通用框架，剩下的就是开发一个实用的 AKC 方案，即本节将要开发和分析的 AKCN-E8。AKCN-E8 的核心是 E8 中一个新的格编码。

我们将多项式 $\sigma_1$ 和 $\sigma_2$ 的系数分为 $\hat{n} = n/8$ 组，每组由 8 个系数组成。具体而言，令 $R = \mathbb{Z}[x]/(x^8 + 1)$ ,  $R_q = R/qR$ ,  $K = \mathbb{Q}[x]/(x^8 + 1)$ 和 $K_{\mathbb{R}} = K \otimes \mathbb{R} \simeq \mathbb{R}[x]/(x^8 + 1)$ 。那么多项式 $\sigma_1$ 可以表示为 $\sigma_1(x) = \sigma_0(x^{\hat{n}}) + \sigma_1(x^{\hat{n}})x + \dots + \sigma_{\hat{n}-1}(x^{\hat{n}})x^{\hat{n}+1}$ ，其中， $\sigma_i(x) \in R_q$ 对于 $i = 0, 1, \dots, \hat{n}$ 。 $\sigma_2$ 可以用同样的方法划分。然后我们只需要为每个 $\sigma_i(x)$ 构建调节机制 Con，并最终将密钥组合在一起。为此，我们需要首先介绍格 $E_8$ ，及其编码和解码。

我们从维度为 8 的扩展汉明代码构造了格 $E_8$ ，为了表示简单，记为 $H_8$ 。 $H_8$ 是指 8 维线性空间 $\mathbb{Z}_2^8$ 的 4 维线性子空间。

$$H_8 = \{c \in \mathbb{Z}_2^8 \mid c = \mathbf{zH} \bmod 2, z \in \mathbb{Z}^4\}$$

其中，

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

编码算法很简单：给定一个 4 比特的字符串 $k_1$ ，计算 $k_1\mathbf{H}$ 。这个操作可以通过按位操作很有效地完成。完整的算法如算法 1 所示。

算法 1: AKCN-E8: E8 中的编码函数 Con

- 1: 子算法:  $\text{Con}(\sigma_1 \in \mathbb{Z}_q^8, k_1 \in \mathbb{Z}_2^4, \text{params})$
- 2:     子算法:  $v = \lfloor \frac{g}{q}(\sigma_1 + \frac{q-1}{2}(k_1\mathbf{H} \bmod 2)) \rfloor \bmod g$
- 3:     返回 v
- 4: 结束子算法

解码算法确定了格 $E_8$ 的最近向量问题（CVP）的解。对于任意给定的 $x \in \mathbb{R}^8$ ，CVP 求解 $E_8$ 中哪个格点最接近 $x$ 。基于 $E_8$ 的结构，我们提出了一种有效的解码算

法。

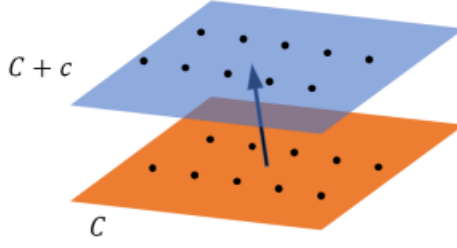


图 4:  $E_8$  的结构

设  $C = \{(x_1, x_1, x_2, x_2, x_3, x_3, x_4, x_4) \in \mathbb{Z}_2^8 \mid x_1 + x_2 + x_3 + x_4 = 0 \pmod{2}\}$ 。实际上， $C$  的跨度是  $H$  的最上面三行。因此， $E_8 = C \cup (C + c)$ ，其中  $c = (0, 1, 0, 1, 0, 1, 0, 1)$  是  $H$  的最后一行。对于给定的  $x \in \mathbb{R}^8$ ，为了求解  $E_8$  中  $x$  的 CVP，我们先求解  $x$  和  $x - c$  在  $C$  中的 CVP，然后再选择距离较小的那个作为输出。对于  $E_8$  的图形表示，参考图 4。

---

算法 2:  $AKCN-E_8$  基于  $E_8$  解码的  $Rec$

---

- 1: 子算法  $Rec(\sigma_2 \in \mathbb{Z}_g^8, v \in \mathbb{Z}_g^8, params)$
  - 2:  $k_2 = Decode_{E_8}(\lfloor \frac{q}{g} v \rfloor - \sigma_2)$
  - 3: 返回  $k_2$
  - 4: 结束程序
- 

接下来，我们考虑如何求解  $C$  中的 CVP。对于  $x \in \mathbb{R}^8$ ，我们选择  $(x_1, x_2, x_3, x_4) \in \mathbb{Z}_2^4$ ，使得  $(x_1, x_1, x_2, x_2, x_3, x_3, x_4, x_4)$  最接近  $x$ ，但是  $x_1 + x_2 + x_3 + x_4 = 0 \pmod{2}$  可能等于 1。在这种情况下，我们选择 4 位字符串  $(x'_1, x'_2, x'_3, x'_4)$ ，使得  $(x'_1, x'_1, x'_2, x'_2, x'_3, x'_3, x'_4, x'_4)$  第二接近  $x$ 。注意  $(x'_1, x'_2, x'_3, x'_4)$  与  $(x_1, x_2, x_3, x_4)$  最多有一个位差异。具体算法见算法 IV。考虑到可能的定时攻击，所有“if”条件语句都可以通过恒定的时间位操作实现。在实践中， $Decode_C^{00}$  和  $Decode_C^{01}$  被实现为两个子线程。

对于算法 3，在  $Decode_{E_8}$  中，我们计算  $cost_{i,b}$ ，其中  $i = 0, 1, \dots, 7, b \in \{0, 1\}$ ，这是指当  $x_i = b$  时对总的 2 范数的贡献，而  $Decode_C^{00}$  解决格  $C$  中的 CVP，而  $Decode_C^{01}$  解决格  $C + c$  中的 CVP。然后我们选择具有较小距离的 CVP。 $Decode_C^{b_0 b_1}$

计算  $k_i$ ,  $i = 0,1,2,3$  使得  $\frac{q-1}{2}(k_0 \oplus b_0, k_0 \oplus b_1, k_1 \oplus b_0, k_1 \oplus b_1, k_2 \oplus b_0, k_2 \oplus b_1, k_3 \oplus b_0, k_3 \oplus b_1)$  最接近  $x$ 。我们使用  $\min_d$  和  $\min_i$  查找第二个最接近的向量。最后, 我们检查奇偶性以决定应该返回哪个。

---

**算法 3**  $E_8$  和  $C$  中的解码

---

```

1: 子算法  $\text{Decode}_{E_8}(\mathbf{x} \in \mathbb{F}_q^8)$ 
2:   当  $i = 0 \dots 7$  时执行:
3:      $\text{cost}_{i,0} = |x_i|_q^2$ 
4:      $\text{cost}_{i,1} = \left| x_i - \frac{q-1}{2} \right|_q^2$ 
5:   结束循环
6:    $(\mathbf{k}^{00}, \text{TotalCost}^{00}) \leftarrow \text{Decode}_C^{00}(\text{cost}_{i \in 0 \dots 7, b \in \{0,1\}})$ 
7:    $(\mathbf{k}^{01}, \text{TotalCost}^{01}) \leftarrow \text{Decode}_C^{01}(\text{cost}_{i \in 0 \dots 7, b \in \{0,1\}})$ 
8:   如果  $\text{TotalCost}^{00} < \text{TotalCost}^{01}$  则
9:      $b = 0$ 
10:   否则
11:      $b = 1$ 
12:   结束判断
13:    $(k_0, k_1, k_2, k_3) \leftarrow \mathbf{k}^{0b}$ 
14:    $\mathbf{k}_2 = (k_0, k_1 \oplus k_0, k_3, b)$ 
15:   返回  $\mathbf{k}_2$ 
16: 子算法结束
17: 子算法  $\text{Decode}_C^{b_0 b_1}(\text{cost}_{i \in 0 \dots 7, b \in \{0,1\}} \in \mathbb{F}_q^{8 \times 2})$ 
18:    $\min_d = +\infty$ 
19:    $\min_i = 0$ 
20:    $\text{TotalCost} = 0$ 
21:   当  $j = 0 \dots 3$  时执行:
22:      $c_0 \leftarrow \text{cost}_{2j, b_0} + \text{cost}_{2j+1, b_1}$ 
23:      $c_1 \leftarrow \text{cost}_{2j, 1-b_0} + \text{cost}_{2j+1, 1-b_1}$ 
24:     如果  $c_0 < c_1$  则
25:        $k_i \leftarrow 0$ 
26:     否则
27:        $k_i \leftarrow 1$ 
28:     结束判断
29:      $\text{TotalCost} \leftarrow \text{TotalCost} + c_{k_i}$ 
30:     如果  $c_{1-k_i} - c_{k_i} < \min_d$  则
31:        $\min_d \leftarrow c_{1-k_i} - c_{k_i}$ 
32:        $\min_i \leftarrow i$ 

```

```

33:          结束判断
34:      结束循环
35:      如果  $k_0 + k_1 + k_2 + k_3 \bmod 2 = 1$  则
36:           $k_{\min_i} \leftarrow 1 - k_{\min_i}$ 
37:          TotalCost  $\leftarrow$  TotalCost +  $\min_d$ 
38:      结束判断
39:       $\mathbf{k} = (k_0, k_1, k_2, k_3)$ 
40:      返回 ( $\mathbf{k}$ , TotalCost)
41: 子算法结束

```

---

接下来的定理给出了算法 1 和算法 2 中的编码和解码算法的一个成功条件。

为了简单起见，对于任意  $\sigma = (x_0, x_1, \dots, x_7) \in \mathbb{Z}_q^8$ ，我们定义  $\|\sigma\|_{q,2}^2 = \sum_{i=0}^7 |x_i|_q^2$ 。

**定理 4.1** 如果  $\|\sigma_1 - \sigma_2\|_{q,2} \leq \frac{q-1}{2} - \sqrt{2}(\frac{q}{g} + 1)$ ，那么由 *Con* 和 *Rec* 计算出的  $k_1$  和  $k_2$  相等。

证明：可扩展汉明码的最小汉明距离是 4，因此，我们使用的格的最小距离

$$\text{是 } \frac{1}{2} \sqrt{\left(\frac{q-1}{2}\right)^2 \times 4} = (q-1)/2。$$

我们可以找到  $\varepsilon, \varepsilon_1 \in [-1/2, 1/2]^8, \theta \in \mathbb{Z}^8$ ，使得：

$$\begin{aligned}
 \lfloor \frac{q}{g} v \rfloor - \sigma_2 &= \frac{q}{g} v + \varepsilon - \sigma_2 \\
 &= \frac{q}{g} \left( \frac{q}{g} (\sigma_1 + \frac{q-1}{2} k_1 H) + \varepsilon + \theta g \right) \\
 &= (\sigma_1 - \sigma_2) + \frac{q-1}{2} k_1 H + \frac{q}{g} \varepsilon + \varepsilon_1 + \theta q
 \end{aligned}$$

因此，由  $\frac{q-1}{2} K_1 H$  产生的偏差不大于

$$\|\sigma_1 - \sigma_2\|_{q,2} + \frac{q}{g} \|\varepsilon\| + \sqrt{2} \leq \|\sigma_1 - \sigma_2\|_{q,2} + \frac{q}{g} \|\varepsilon\| + \sqrt{2}(\frac{q}{g} + 1)。$$

如果其值小于最小距离  $(q-1)/2$ ，解码将是正确的，这意味着  $k_1 = k_2$ 。

**命题 4.1.** *AKCN-E8* 是安全的。特别地，如果  $\sigma_1$  在  $\mathbb{Z}_q^8$  服从均匀分布，那么  $v$  和  $k_1$  是独立。

证明：对于任意固定的  $k_1$ ， $k_1 H \bmod 2$  是固定的。由于  $\sigma_1$  是均匀分布的，

$\sigma_1 + \frac{q}{2}(k_1 H \bmod 2)$  在  $\mathbf{Z}_q$  上是均匀随机的。因此,  $v$  服从分布  $\lfloor \frac{g}{q}u \rfloor$ , 其中  $u$  在  $\mathbf{Z}_q$  上是均匀随机的, 因此,  $v$  独立于  $k_1$ 。

## 4.1 错误率分析

现在, 对于图 3 中描述的基于 RLWE 难题和 AKC 机制的 CPA 安全的 PKE 一般性方案, 我们将其中得 AKC 替换为 AKCN-E8, 得到得 KEM 方案称为 CPA-安全得 AKCN-E8-KEM。我们接下来分析其错误率。

令  $\varepsilon = ax_2 + e_2 - 2^t \lfloor (ax_2 + e_2) / 2^t \rfloor$ , 则

$$\begin{aligned}\sigma_1 - \sigma_2 &= x_1(2^t y_2) - (y_1 x_2 + e'_2) \\ &= 2^t x_1 \lfloor (ax_2 + e_2) / 2^t \rfloor - ((ax_1 + e_1)x_2 + e'_2) \\ &= x_1(ax_2 + e_2 - \varepsilon) - (ax_1 x_2 + e_1 x_2 + e'_2) \\ &= x_1(e_2 - \varepsilon) - (e_1 x_2 + e'_2)\end{aligned}$$

根据 RLWE 假设,  $(a, ax_2 + e_2)$  与  $(a, u)$  计算不可区分, 其中  $u$  服从均匀分布。然后,  $\varepsilon$  接近  $u - 2^t \lfloor u/2^t \rfloor$ 。我们可以粗略地把  $u - 2^t \lfloor u/2^t \rfloor$  的多项式的每个系数看作在  $[-2^{t-1}, 2^{t-1}]^n$  上的均匀分布。 $\sigma_t$  为  $[-2^{t-1}, 2^{t-1}]^n$  均匀分布的标准差。然后我们可以计算出  $\sigma_2 - \sigma_1$  中每个多项式的系数的标准差, 表示为  $s$ 。因此有  $s^2 = n\sigma^2(2\sigma^2 + \sigma_t^2) + \sigma^2 = n\sigma^2\left(2\sigma^2 + \frac{(1+2^t)^2-1}{12}\right) + \sigma^2$ 。

由中心极限定理可知,  $\sigma_2 - \sigma_1$  中的多项式的每个系数接近高斯分布。由定理 4.1 可知, AKCN-E8 正确的概率为  $\Pr\left[d' \leftarrow \chi^2(8): \sqrt{d'} \leq \left(\frac{q-1}{2} - \sqrt{2}\left(\frac{q}{g} - 1\right)\right)/s\right]$ 。

作为算法实现得一部分, 我们提供了一个脚本来计算具体的失败率; 该脚本也可以从 <http://github.com/AKCN-E8> 获得。

## 5 算法模块化完整描述、参数选取及性能分析

AKCN-E8-KEM 方案是由第三章中描述的模块和通用框架产生的, 其底层的



AKC 机制被第四章中提出的 AKCN-E8 方案所取代，该方案在 RLWE 问题的任何困难的实例上都有效。但如果  $n$  是 2 的幂，并且质数  $q$  满足  $q \bmod 2n = 1$ ，则可以使用数论变换(NTT)来加速多项式的乘法。使用 Montgomery 算法和 AVX2 指令集可以进一步提高性能[1, 23]。在[23]中，底层的噪声分布是中心二项分布  $S_\eta$ ：对于一个正整数  $\eta$ ，取样  $(a_1, \dots, a_\eta, b_1, \dots, b_\eta) \leftarrow \{0, 1\}^{2\eta}$ ，然后输出  $\sum_{i=1}^{\eta} (a_i - b_i)$ 。中心二项分布  $S_\eta$  的标准差为  $\sigma = \sqrt{\eta/2}$ 。在 NEWHOPE [23] 中， $q = 12289, n = 512$  或  $n = 1024, \eta = 8$ 。为了便于比较，我们使用与 NewHope [23] 相同的 CCA 转换和相同的值  $(q, n)$  来构建和实现 AKCN-E8-KEM。

## 5.1 参数选取、性能分析和测试

我们使用与 NewHope-KEM [23] 相同的 RLWE 难度评测脚本，这是现在对 LWE 及其变体难度通行的测评方法。具体安全评估的方法和脚本请参考[23]，也可以从 <https://newhopecrypto.org/> 获得。为了便于测评，我们的算法提交包也包含了该 RLWE 难度测评脚本。

NewHope-1024(或 NewHope-512)的目标是 233 位(或 101 位)后量子安全(简称，pq-sec)，但仅封装了 256 (或 128)位的共享密钥  $k_1 = k_2$ 。考虑到 Grover 搜索算法的平方加速，以及从长远来看更复杂和更先进的量子密码分析的可能性，我们认为 NewHope-KEM 共享密钥的大小可能与后量子时代的安全级别目标不匹配。事实上，人们普遍认为在后量子时代对称密钥体制（如 AES）需要更大的密钥大小。在增加共享密钥大小方面，NewHopeKEM 的灵活性较差。例如，我们如果想要一个共享密钥 512 位的 NewHopeKEM，那就必须使用一个 2048 次的多项式，使得效率明显降低。而由于  $E_8$  格编码的优良性质，AKCN-E8-1024 可以达到 512 比特的密钥长度，AKCN-E8-512 也可以达到 256 比特的密钥长度。

AKCN-E8-KEM 的参数和性能如下表所示，其中  $|K|$  指的是共享密钥的长度， $pk(B)$  和  $cipher(B)$  对应公钥和密文的字节数， $err$  是错误率，pq-sec 是以比特数衡量的对应参数的 RLWE 难度值，c-sec 是经典安全难度。对于 AKCN-E8- 512 和 AKCN-E8-1024，我们给出了三组参数：“S”代表更高的安全级别，“E”代表比较平衡的性能，“C”代表密文大小优先。与 NewHopeKEM [23] 相比，AKCN-

E8 总是能将共享密钥的大小增加一倍，这对于确保后量子时代的目标安全级别、抵御 Grover 算法等先进的量子攻击来说具有较大的实际意义。对于我们给出的每一组参数，AKCN-E8 密文长度也比相应的 NewHope-KEM 小。相比于 NewHopeKEM，除了密钥长度扩展至两倍的优势，AKCN-E8-512-S 和 AKCN-E8-1024-S 还同时具有更强的安全性、更低的错误概率和更小的密文大小。另外，NewHope-1024 的错误率并不能与其标准安全性匹配，而 AKCN-E8 的所有参数对应的错误率均既匹配后量子安全性也匹配经典安全性。

	$ K $	$n$	$q$	$\eta$	$g$	$t$	c-sec	pq-sec	err	pk (B)	cipher (B)
NewHope-512-CPA	128	512	12289	8	$2^3$	0	112	101	$2^{-213}$	928	1088
AKCN-E8-512-S-CPA	256	512	12289	14	$2^4$	3	121	110	$2^{-224}$	928	960
AKCN-E8-512-E-CPA	256	512	12289	8	$2^4$	4	112	101	$2^{-256}$	928	896
AKCN-E8-512-C-CPA	256	512	12289	8	$2^3$	4	112	101	$2^{-150}$	928	832
NewHope-512-CCA	128	512	12289	8	$2^3$	0	112	101	$2^{-213}$	928	1120
AKCN-E8-512-S-CCA	256	512	12289	14	$2^4$	3	121	110	$2^{-224}$	928	992
AKCN-E8-512-E-CCA	256	512	12289	8	$2^4$	4	112	101	$2^{-256}$	928	928
AKCN-E8-512-C-CCA	256	512	12289	8	$2^3$	4	112	101	$2^{-150}$	928	864
NewHope-1024-CPA	256	1024	12289	8	$2^3$	0	257	233	$2^{-216}$	1824	2176
AKCN-E8-1024-S-CPA	512	1024	12289	10	$2^4$	2	265	240	$2^{-274}$	1824	2048
AKCN-E8-1024-E-CPA	512	1024	12289	8	$2^4$	3	257	233	$2^{-280}$	1824	1920
AKCN-E8-1024-C-CPA	512	1024	12289	4	$2^3$	3	236	214	$2^{-500}$	1824	1792
NewHope-1024-CCA	256	1024	12289	8	$2^3$	0	257	233	$2^{-216}$	1824	2208
AKCN-E8-1024-S-CCA	512	1024	12289	10	$2^4$	2	265	240	$2^{-274}$	1824	2080
AKCN-E8-1024-E-CCA	512	1024	12289	8	$2^4$	3	257	233	$2^{-280}$	1824	1952
AKCN-E8-1024-C-CCA	512	1024	12289	4	$2^3$	3	236	214	$2^{-500}$	1824	1824

AKCN-E8 的性能优势以及在参数选择上的灵活性，很大程度上是由底层的  $E_8$  格编码实现的，它比 NewHope-KEM [23, 24]使用的底层  $Z_4$  格编码密集得多。实际上，近年来数学上取得了一个显著突破，即证明填充球单元问题， $E_8$  中的填充可以达到最佳密度，所以  $E_8$  格中的球体填充是最优的[29]。

#### 指定参数：

一、困难问题：	RLWE
(1) 秘密向量维度 $n$	1024
(2) 模数 $q$	12289
(3) 秘密分布	中心二项分布
(4) 分布标准差 $sd$	$\sqrt{8}$
(5) 声称的经典安全强度	257
(6) 声称的量子安全强度	233

算法实现自测结果如下：

KEM 方案	平均运行效率(ms/次)	测试平台
密钥生成算法	0.065	<b>设备类型：</b> 微型计算机 <b>型号：</b> HP Pavilion Gaming Desktop PC <b>CPU：</b> Intel® Core™ i7-8700 CPU @ 3.20GHz <b>内存：</b> RAM 8GB(1*8GB) DDR4 2666 NECC <b>硬盘：</b> HDD 1TB 7200RPM SATA 3.5 cDT 2 <sup>nd</sup> SSD 128G 2280 PCIe NVMe Value <b>软件环境：</b> Windows 10。 Microsoft Visual Studio Community2017
封装算法	0.101	
解封装算法	0.122	

KEM 方案	规模(字节)	测试平台
公钥	1824	<b>设备类型：</b> 微型计算机 <b>型号：</b> HP Pavilion Gaming Desktop PC <b>CPU：</b> Intel® Core™ i7-8700 CPU @ 3.20GHz <b>内存：</b> RAM 8GB(1*8GB) DDR4 2666 NECC <b>硬盘：</b> HDD 1TB 7200RPM SATA 3.5 cDT 2 <sup>nd</sup> SSD 128G 2280 PCIe NVMe Value <b>软件环境：</b> Windows 10。 Microsoft Visual Studio Community2017
私钥	3680	
密文	1824	

## 5.2 算法模块化完整描述、与实现说明

### 5.2.1 CPA-安全的 AKCN-E8-KEM 算法描述和实现

本节给出 CPA 安全 AKCN-E8-PKE 和 CCA 安全的 AKCN-E8-KEM 的描述。CPA-PKE 见算法 4、5、8，与 NewHopeKEM [23]类似，我们也使用 NTT 来加快多项式的乘法。我们实现的是推荐参数：AKCN-E8-1024-C。

在算法 4 中，密钥生成算法随机抽取一个种子样本，然后使用该种子来确定地生成 seedPublic 和 seedPrivate。使用 seedPublic 可以生成与诚实方相同的  $\hat{\mathbf{a}}$ 。种子 seedPublic 是公钥 pk 的一部分。算法 Encode( $\hat{\mathbf{y}}_1$ ) 将  $\hat{\mathbf{y}}_1$  中 14bit 的系数聚集在一起。

我们使用下面的算法 6 对密文进行编码和压缩。对于  $\mathbf{y}_2$  中的每个系数，我们将其四舍五入至区间  $[0, 2^{11} - 1]$ 。对于  $\mathbf{v}$  中的每个系数，我们将其四舍五入至区间  $[0, 2^3 - 1]$ 。然后我们把调整后的  $\mathbf{y}_2$  的 11bit 系数放在较高位，把  $\mathbf{v}$  的 3bit 系数放在较低位，组成 14bit 的整数。为了加快四舍五入和其他操作，我们使用位运算。最后，我们调用了编码算法来聚集 14 位整数。在算法 7 中，我们使用了类似的算法来解压缩和解码  $\mathbf{y}_2$  和  $\mathbf{v}$ 。

---

#### 算法 4 密钥生成算法

---

```
1  函数 KEYGEN
2      /* 生成种子，处理后将其分为两部分，分别用于公私钥的生成 */
3      seed  $\leftarrow \{0,1\}^{256}$ 
4      (seedPublic, seedPrivate) = H(seed)
5      /* 根据种子产生相应矩阵与向量参数，得到公私钥 */
6       $\hat{\mathbf{a}} = \text{GenA}(\text{seedPublic})$ 
7       $\mathbf{x}_1 \leftarrow \text{SampleNoise}(\text{seedPrivate}, 0)$ 
8       $\hat{\mathbf{x}}_1 \leftarrow \text{NTT}(\mathbf{x}_1)$ 
9       $\mathbf{e}_1 \leftarrow \text{SampleNoise}(\text{seedPrivate}, 1)$ 
10      $\hat{\mathbf{e}}_1 \leftarrow \text{NTT}(\mathbf{e}_1)$ 
11      $\hat{\mathbf{y}}_1 \leftarrow \hat{\mathbf{a}} \circ \hat{\mathbf{x}}_1 + \hat{\mathbf{e}}_1$ 
12     返回 pk=(Encode( $\hat{\mathbf{y}}_1$ ), seedPublic), sk=Encode( $\hat{\mathbf{x}}_1$ )
13  函数结束
```

---

---

#### 算法 5 加密算法

---

```
1  函数 ENCRYPT(pk, msg)
2      /* 根据公钥生成相应种子与向量参数 */
```

```

3      ( $\hat{y}_1, \text{seedPublic}$ ) = Decode(pk)
4       $\hat{\mathbf{a}} = \text{GenA}(\text{seedPublic})$ 
5      /* 取样噪音, 并进行 NTT 多项式快乘运算, 得出  $\sigma_2$  */
6       $\mathbf{x}_1, \mathbf{e}_2, \mathbf{e}_2' \leftarrow \text{SampleNoise}()$ 
7       $\hat{\mathbf{x}}_2 \leftarrow \text{NTT}(\mathbf{x}_2)$ 
8       $\hat{\mathbf{e}}_2 \leftarrow \text{NTT}(\mathbf{e}_2)$ 
9       $\hat{\mathbf{e}}_2' \leftarrow \text{NTT}(\mathbf{e}_2')$ 
10      $\mathbf{y}_2 = \text{NTT}^{-1}(\hat{\mathbf{a}} \circ \hat{\mathbf{x}}_2 + \hat{\mathbf{e}}_2)$ 
11      $\sigma_2 = \text{NTT}^{-1}(\hat{\mathbf{y}}_1 \circ \hat{\mathbf{x}}_2 + \hat{\mathbf{e}}_2')$ 
12     /* 压缩消息, 得到信号向量  $\mathbf{v}$  */
13      $\mathbf{v} \leftarrow \text{Con}(\sigma_2, \text{msg})$ 
14     返回  $\text{ct} = \text{CompressAndEncode}(\mathbf{y}_2, \mathbf{v})$ 
15  函数结束

```

---



---

#### 算法 6 压缩和编码

---

```

1:  函数 COMPRESSANDENCODE ( $y_2, v$ )
2:   $c = 0$ 
3:  for  $i = 1 \dots 1024$  do
4:   $hi = ((y_2(i) \ll 11) + 6144) / 12289$ 
5:   $lo = ((v[i] \ll 3) + 6144) / 12289$ 
6:   $c[i] = (hi \ll 3) + lo$ 
7:  end for
8:  返回 Encode(c)
9:  结束函数

```

---



---

#### 算法 7 解码和解压缩

---

```

1:  函数 DECODEANDDECOMPRESS (ct)
2:   $c = \text{Decode}(\text{ct})$ 
3:  for  $i = 1 \dots 1024$  do
4:   $hi = (c[i] \ll 3) \& 0x7FF$ 
5:   $lo = c[i] \& 3$ 
6:   $y_2'[i] = (hi * 12289 + 0x400) \ll 11$ 
7:   $v'[i] = (lo * 12289 + 0x4) \ll 3$ 
8:  end for
9:  返回 ( $y_2', v'$ )
10: 结束函数

```

---



---

#### 算法 8 解密

---

```

1:  函数 DECRYPT ( $sk, ct$ )
2:   $\hat{\mathbf{x}}_1 = \text{Decode}(sk)$ 
3:  ( $y_2', v'$ ) = DecodeAndDecompress(ct)
4:   $\hat{y}_2' \leftarrow \text{NTT}(\hat{y}_2')$ 

```

---

---

```

5:       $\sigma_1 \leftarrow \text{NTT}^{-1}(\hat{y}'_2 \circ \hat{x}_1)$ 
6:      返回  $\text{Rec}(\sigma_1, v')$ 
7:      结束函数

```

---

### 5.2.2 CCA-安全的 AKCN-E8-KEM 算法描述和实现

AKCN-E8-CCA-KEM 是利用 FO 转换对 AKCN-E8-CPA-PKE 进行转换得来的。Fujisaki-Okamoto 变换[11]从一个传统随机预言机模型下单向安全的公钥加密方案构造出一个 IND-CCA2 安全的公钥加密方案（假设每个明文的密文分布都足够接近均匀分布）。Taeghi 和 Unruh [27] 给出了一个 Fujisaki-Okamoto 变换的变体，并证明了相同假设下量子预言机模型中面对量子敌手它是具有 IND-CCA2 安全性的。FO 和 TU 的结果都是在公钥加密方案完全正确性（即：错误率为 0）的假设下进行的，而基于 LWE 的 KEM 方案一般都有错误率。Hofheinz, Hövelmanns, and Kiltz [12]给出了一个模块化的从 CPA-PKE 到 CCA-KEM 的转换构造，这也是 NewHope-KEM 和我们 CCA-安全 AKCN-E8-KEM 所使用的转换技术。我们应用[12]的  $FO_m^\chi$  转换：利用一个 IND-CPA 安全的公钥加密方案和三个哈希函数构造一个 IND-CCA 安全的密钥封装机制。

$FO_m^\chi$  转换：令  $\text{PKE} = (\text{KeyGen}, \text{Encrypt}, \text{Decrypt})$  表示一个明文空间为  $M$  密文空间为  $C$  的公钥加密方案，其中  $\text{Encrypt}$  的随机空间是  $\mathbb{R}^E$ 。令  $\text{len}_s, \text{len}_k, \text{len}_d, \text{len}_{ss}$  表示如下参数。  $G: \{0, \dots, 255\}^* \rightarrow \{0, \dots, 255\}^{\text{len}_k} \times \mathbb{R}^E \times \{0, \dots, 255\}^{\text{len}_d}$ ,  $F: \{0, \dots, 255\}^* \rightarrow \{0, \dots, 255\}^{\text{len}_{ss}}$  为两哈希函数。定义  $\text{KEM}^{\text{FO}} = FO_m^\chi[\text{PKE}, G, F]$  为一个密钥封装机制，如图 5 所示由  $\text{KEM}^{\text{FO}}.\text{KeyGen}()$ 、 $\text{KEM}^{\text{FO}}.\text{Encaps}()$ 、 $\text{KEM}^{\text{FO}}.\text{Decaps}()$  组成。

<p><u><math>\text{KEM}^{\text{FO}}.\text{KeyGen}()</math>:</u></p> <pre> 1: (<math>pk, sk</math>) <math>\leftarrow</math> <math>\\$</math> <math>\text{PKE}.\text{KeyGen}()</math> 2: <math>s \leftarrow \\$ \{0, \dots, 255\}^{\text{len}_s}</math> 3: <math>\overline{sk} \leftarrow (sk, s, pk)</math> 4: 返回 (<math>pk, \overline{sk}</math>)  <u><math>\text{KEM}^{\text{FO}}.\text{Encaps}(pk)</math>:</u> 1: <math>\mathbf{K}_2 \leftarrow \\$ \square_m^{l_A \times l_B}</math> 2: (<math>\mathbf{k}, \text{coin}', d</math>) <math>\leftarrow G(pk \parallel \mu)</math> 3: <math>c \leftarrow \text{PKE}.\text{Enc}(\mu, pk; \text{coin}')</math> </pre>	<p><u><math>\text{KEM}^{\text{FO}}.\text{Decaps}((c, d), (sk, s, pk))</math>:</u></p> <pre> 1: <math>\mathbf{K}_1 \leftarrow \text{PKE}.\text{Dec}(c, sk)</math> 2: (<math>\mathbf{k}', \text{coin}'', d'</math>) <math>\leftarrow G(pk \parallel \mathbf{K}_1)</math> 3: 如果 <math>c = \text{PKE}.\text{Enc}(\mathbf{K}_1, pk; \text{coin}'')</math> 且 <math>d = d'</math> 4:     返回 <math>ss' \leftarrow F(\mathbf{k}' \parallel c \parallel d)</math> 5: 否则 6:     返回 <math>ss' \leftarrow F(s \parallel c \parallel d)</math> </pre>
---	---

```

4:  $\mathbf{ss} \leftarrow F(\mathbf{k} \parallel c \parallel d)$ 
5:  $\bar{c} \leftarrow (c, d)$ 
6: 返回  $(\bar{c}, \mathbf{ss})$ 

```

图 5: 利用 PKE 方案和哈希函数  $G$  和  $F$  通过 FO 转换构造 IND-CCA 的密钥封装机制

AKCN-E8-KEM 是从 AKCN-CPA-PKE 方案通过  $FO_m^\mathcal{L}$  转换得到的。哈希函数在实现时都是采用的 SHAKE256。参数的长度取为  $\text{len}_s = \text{len}_k = \text{len}_d = \text{len}_{ss} = 32$ 。随机空间为  $\{0, \dots, 255\}^{32}$ ，消息空间为  $\{0, \dots, 255\}^{32}$ 。详细算法由密钥生成算法（算法 9）、密钥封装算法（算法 10）、解封装算法（算法 11）三个算法构成。

---

**算法 9 KeyGen: AKCN-E8-CCA-KEM 密钥生成算法**

---

```

1: 函数 KeyGen()
2:    $(\mathbf{pk}, \mathbf{sk}) \xleftarrow{\$} \text{AKCN-E8-CPA-PKE.KEYGEN}$ 
3:    $s \xleftarrow{\$} \{0, \dots, 255\}^{32}$ 
4:   返回  $(\mathbf{pk}, \mathbf{sk} = \mathbf{sk} \parallel \text{SHAKE256}(32, \mathbf{pk}) \parallel s)$ 

```

---



---

**算法 10 Encaps: AKCN-E8-CCA-KEM 密钥封装算法**

---

```

1: 函数 Encaps(pk)
2:    $\text{coin} \xleftarrow{\$} \{0, \dots, 255\}^{32}$ 
3:    $K_2 \leftarrow \text{SHAKE256}(32, \text{coin}) \in \{0, \dots, 255\}^{32}$ 
4:    $k \parallel \text{coin}' \parallel d \leftarrow \text{SHAKE256}(96, K_2 \parallel \text{SHAKE256}(32, \mathbf{pk})) \in \{0, \dots, 255\}^{32+32+32}$ 
5:    $c \leftarrow \text{AKCN-E8-CPA-PKE.Encrypt}(\mathbf{pk}, K_2; \text{coin}')$ 
6:    $\mathbf{SS} \leftarrow \text{SHAKE256}(32, k \parallel \text{SHAKE256}(32, c \parallel d))$ 
7:   返回  $(\bar{c} = c \parallel d, \mathbf{SS})$ 

```

---



---

**算法 11 Decaps: AKCN-E8-CCA-KEM 解封装算法**

---

```

1: 函数 Decaps( $(\bar{c}, \bar{sk})$ )
2:    $c \parallel d \leftarrow \bar{c} \in \{0, \dots, 255\}^{3n/8+7n/4+32}$ 
3:    $sk \parallel \mathbf{pk} \parallel h \parallel s \leftarrow \bar{sk} \in \{0, \dots, 255\}^{7n/4+7n/4+32+32+32}$ 
4:    $K_1 \leftarrow \text{AKCN-E8-CPA-PKE.Decrypt}(c, \mathbf{ct})$ 
5:    $k' \parallel \text{coin}'' \parallel d' \leftarrow \text{SHAKE256}(96, K_1 \parallel h) \in \{0, \dots, 255\}^{32|32|32}$ 
6:   如果  $c = \text{AKCN-E8-CPA-PKE.Encrypt}(\mathbf{pk}, K_1; \text{coin}')$  且  $d = d'$ ，那么
7:      $\text{fail} \leftarrow 0$ 
8:   否则
9:      $\text{fail} \leftarrow 1$ 
10:    $k_0 \leftarrow k'$ 
11:    $k_1 \leftarrow s$ 

```

---

---

12: 返回  $SS = \text{SHAKE256}(32, K_{\text{fail}} \parallel \text{SHAKE256}(32, c \parallel d))$

---

## 6 算法创新性和特色、优缺点说明

NewHope-KEM 作为 NIST 后量子密码标准竞赛第二轮的一个重要提案，其特色是高效、并且具有非常高的安全冗余度。由于现在对实际的量子计算我们还知之甚少，尽可能在保持高效的基础上提供充足的安全冗余是密码标准，特别是面向未来几十年量子计算发展，的明智选择。

但是，NewHope-KEM 所基于的 D4 格编码存在如下不足：（1）只能封装多项式维度的四分之一长度的密钥（具体而言，1024 维多项式只能封装 256 比特密钥），这在后量子世界长远发展的角度，可能与其底层 RLWE 难度不匹配。

（2）纠错能力偏弱（由于所基于的底层四维格装球密度不是最优），这拖累了其整体的综合性能并限制了其参数选取的灵活度。

### 6.1 AKCN-E8-KEM 创新性

AKCN-E8-KEM 方案的主要创新主要是编码机制的创新，具体而言是基于 E8 格编码的非对称密钥共识机制 AKCN-E8 的设计与分析。由于 E8 上的装球问题有密度最优解，使得我们的 AKCN-E8 编码机制可以显著提升纠错能力，并将共识出的密钥长度翻倍（相对于 D4 格编码）。

### 6.2 AKCN-E8-KEM 特色

1. 可以在安全性、共享密钥尺寸、带宽、错误率等所有方面同时超越 NIST 第二轮重要算法提案 NewHope-KEM 密钥封装算法。

2. 安全冗余度高：无论是所基于的 RLWE 问题难度还是共享密钥尺寸都有很高的安全冗余度。

3. 参数选取灵活：根据安全性优先、带宽优先等不同应用场景，可以灵活



配置参数。

#### 4. 模块化设计与分析。

## 6.3 AKCN-E8-KEM 优缺点

1. 相对于 NewHope-KEM, AKCN-E8-KEM 可以采用参数使得以下优势同时满足: (1) 共享密钥长度翻倍; (2) 更高的安全性; (3) 更短的公钥和密文尺寸; (4) 更低的错误率。

2. 是目前已知的同时满足如下条件的综合性能最优的 RLWE-KEM 方案之一: (1) 512 比特共享密钥; (2) 后量子安全 210 比特以上; (3) 错误率低于  $2^{-256}$ 。

3. 计算高效。

4. 理想格 RLWE 相对于一般格 LWE 由于具有更丰富的代数结构, 而这些代数结构不排除未来会被(特别是量子算法)利用从而带来相对较多的安全隐患。这个不足我们通过充足的安全冗余和翻倍的密钥长度来加以克服。

5. 相对于目标封装 266 比特 128 比特后量子安全级别的格密码方案, AKCN-E8 充足的安全冗余和 512 比特的封装密钥导致带宽适度增大(但是相对于 NewHope-KEM, 带宽显著降低)。但是, 后量子密码主要是针对未来网络环境(5G 甚至 6G), 在新一代网络通讯环境中, 适度增加的带宽的影响会降低, 反而充足的安全冗余和翻倍的密钥封装能力在未来网络和计算环境带来的优势更明显。

## 7 适配性说明

1. AKCN-E8-KEM 与 NIST 后量子密码竞赛第二轮的 NIST 高度匹配: 可以使用相同的参数, 并执行相同的基础操作。由于 NewHope-KEM 是 NIST 第二轮的重要候选算法, 不排除最终被 NIST 标准化的可能(在 NIST 第二轮算法中可能性还是相对较高的)。如果 NewHope-KEM 被 NIST 选为标准, 则 AKCN-E8 将具有国际标准的高度适配性。

2. RLWE 是 MLWE 的一个特殊形式。在我国算法竞赛第二轮的算法提案中，基于 RLWE 和 MLWE 的算法是最多的一类，涵盖密钥封装、数字签名、和密钥交换三大类。因此，AKCN-E8 与这些算法在数学基础和多项式运算基本操作上具有很好的匹配性。

3. 由于 24 维的 Leech 格编码已经在 IEEE 802.11 无线通讯标准中应用，作为我们方案核心工具所发展的 E8 格编码和 Leech 格编码具有同等的纠错能力，但更为高效、更为灵活，因此在无线通讯中也有良好应用前景从而在更大的应用领域获得良好适配性。

## 8 算法第二轮修改说明

基于已有成熟的且 NewHope 相同的 CCA-KEM 转换技术，增加了 CCA 安全 KEM 描述和实现；测试和优化了参数选取。

## 9 支持文档

[AKCN-E8] AKCN-E8: Compact and Flexible Key Encapsulation Mechanism from Ideal Lattice.

[ACNS19] Generic and Practical Key Establishment from Lattice. ACNS 2019, 最佳学生论文。

[ArXiv16] Optimal Key Consensus in Presence of Noise. CoRRabs/1611.06150(2016)

## 10 参考文献

1. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum Key Exchange — A New Hope. 25th USENIX Security Symposium (USENIX Security 16), pages 327–343. Winner of the 2016 Internet Defense Prize (<https://internetdefenseprize.org/>)
2. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. CRYPTO 2009: 595-618.
3. J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A.

- Raghuathan, and D. Stebila. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. ACM CCS 2016: 1006-1018.
4. H. Cohn, A. Kumar, S. D. Miller, D. Radchenko, M. Viazovska. The Sphere Packing Problem in Dimension 24. *Annals of Mathematics*, 185 (3): 1017-1033, 2017.
  5. J. H. Conway and N. Sloane. *Sphere Packings, Lattices, and Groups*. Springer-Verlag, New York, 1993.
  6. R. Cramer, I. Damgrd and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. CRYPTO 1994: 174187.
  7. A. W. Dent. A Designers Guide to KEMs. Cryptology ePrint Archive, Report 2002/174, 2002.
  8. L. Ducas and A. Durmus. Ring-LWE in Polynomial Rings. PKC 2012: 34-51.
  9. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986: 186194.
  10. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* Volume 83, Issue 1, pages 24-32, 1999.
  11. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, Volume 26, Issue 1, pages 80-101, 2013.
  12. D. Hofheinz, K. H"ovelmanns, and E. Kiltz. A Modular Analysis of the Fujisaki-Okamoto Transformation. TCC (1) 2017: 341-371.
  13. K. H"ovelmanns, E. Kiltz, S. Sch"age, and D. Unruh. Generic Authenticated Key Exchange in the Quantum Random Oracle Model. Cryptology ePrint Archive, Report 2018/928.
  14. H. Jiang, Z. Zhang, and Z. Ma. Tighter Security Proofs for Generic Key Encapsulation Mechanism in the Quantum Random Oracle Model. PQCrypto 2019: 227-248.
  15. Z. Jin, and Y. Zhao. Optimal Key Consensus in Presence of Noise. CoRR, abs/1611.06150 (2016) <https://arxiv.org/abs/1611.06150>
  16. Z. Jin, and Y. Zhao. Generic and Practical Key Establishment from Lattice. ACNS 2019: 302-322. (Best Student Paper)
  17. R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. CT-RSA 2011: 319-339.
  18. V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with

- Errors over Rings. EUROCRYPT 2010: 1-23.
19. V. Lyubashevsky, C. Peikert, and O. Regev. A Toolkit for Ring-LWE Cryptography. EUROCRYPT 2013: 35-54
  20. NIST. Post-Quantum Cryptography Standardization. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization>
  21. C. Peikert, O. Regev and N. Stephens-Davidowitz. Pseudorandomness of Ring-LWE for Any Ring and Modulus. STOC 2017: 461-473.
  22. A.V. Poppelen, Cryptographic Decoding of the Leech Lattice. Cryptology ePrint Archive, Report 2016/1050, 2016.
  23. T. Pöppelmann, E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. Piedra, P. Schwabe, D. Stebila, M. Albrecht, E. Orsini, V. Osheter, K. Paterson, G. Peer, and N. Smart. Supporting documentation: Newhope. Technical report, National Institute of Standards and Technology. <https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/round-2-submissions>
  24. T. Pöppelmann and T. Güneysu. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. SAC 2013: 68-85.
  25. O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Journal of the ACM (JACM), Volume 56, Issue 6, pages 34, 2009.
  26. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3.
  27. E. E. Targhi and D. Unruh. Post-Quantum Security of the Fujisaki-Okamoto and OAEP Transforms. TCC 2016-B: 192-216.
  28. A. Vardy, and Y. Be'ery. Maximum Likelihood Decoding of the Leech Lattice. IEEE Transactions on Information Theory, 39(4):1435-1444, 1993.
  29. M. S. Viazovska. The Sphere Packing Problem in Dimension 8. Annals of Mathematics, 185(3): 991-1015, 2017.