CRYPTREC
Cryptography Research and Evaluation Committees

# NIST標準暗号 格子ベース暗号FIPS 203, 204の 数学的構成の解説

2025年7月25日（金）

立教大学・理学部数学科
安田雅哉

# FIPS 203, 204に関する基本情報

## FIPS 203

- **M**odule-**L**attice-Based **K**ey-**E**ncapsulation **M**echanism Standard **(ML-KEM)**
  - Module-Lattice-Based Key-Encapsulation Mechanism Standard
  - Module格子上のLWE暗号ベースの鍵カプセル化メカニズム標準
  - CRYSTALS-Kyberに基づく[1]

## FIPS 204

- **M**odule-**L**attice-Based **D**igital **S**ignature Standard **(ML-DSA)**
  - Module-Lattice-Based Digital Signature Standard
  - Module格子上のLWE暗号ベースのデジタル署名標準
  - CRYSTALS-Dilithiumに基づく[2]

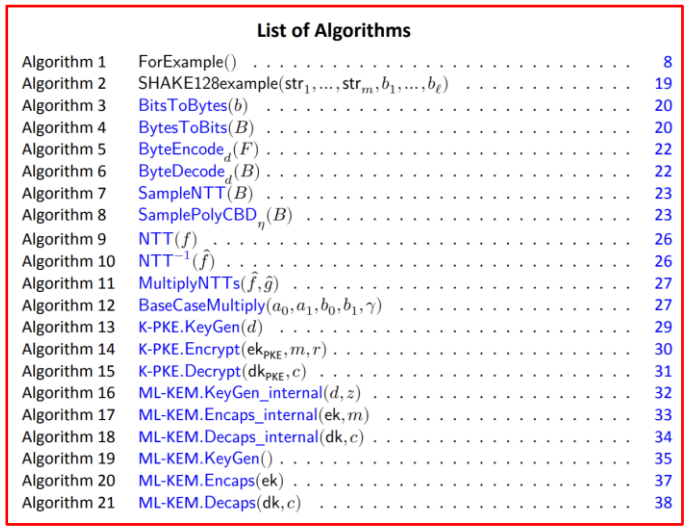
**[1] CRYSTALS-Kyber (version 3.02) – Submission to round 3 of the NIST post-quantum project**
Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé.
Specification document (update from August 2021). 2021-08-04. kyber-specification-round3-20210804.pdf
**[2] CRYSTALS-Dilithium – Algorithm Specifications and Supporting Documentation (Version 3.1)**
Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé.
Specification document (update from February 2021). 2021-02-08. CRYSTALS-Dilithium

# FIPS標準化文書と本講演の目的

## ■ FIPS 203, 204標準化文書

- アルゴリズムレベルの記述が多い
  - ➢ サブ関数含めて，アルゴリズム通りに実装すれば良い
- 一方，全体の構成の理解が困難
  - ➢ 主アルゴリズムがどれか分かりにくく
  - ➢ 暗号方式の基本構成や，その仕組み・原理の理解が困難

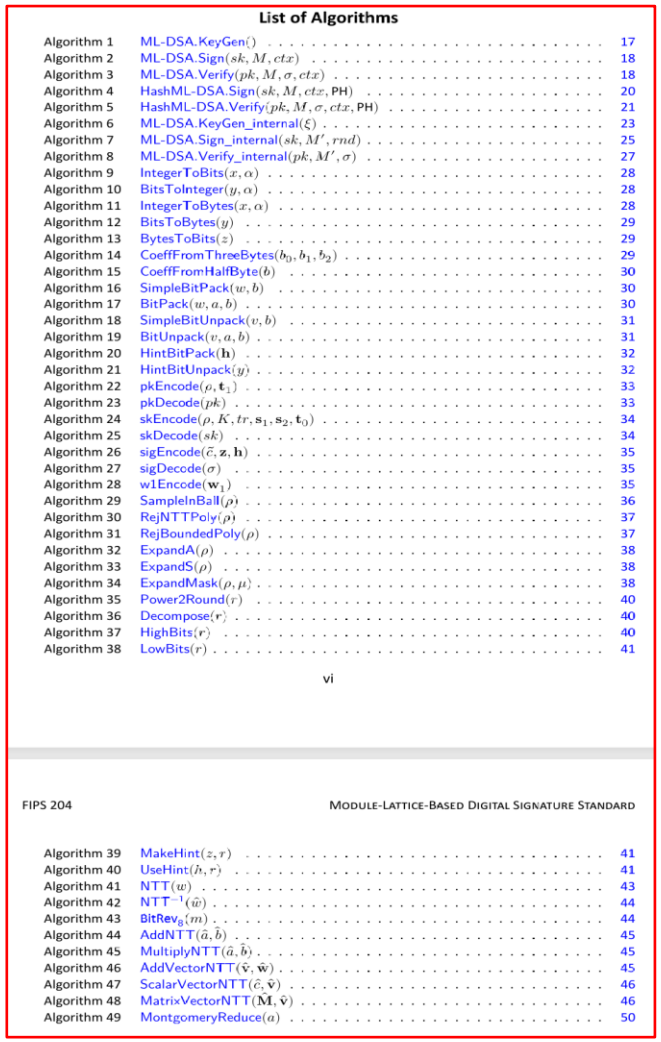
FIPS 203（ML-KEM）における
アルゴリズムのリスト（21個）



List of Algorithms



FIPS 204（ML-DSA）における
アルゴリズムのリスト（49個）



List of Algorithms



vi

FIPS 204 MODULE-LATTICE-BASED DIGITAL SIGNATURE STANDARD



## ■ 本講演の目的

- **FIPS 203（ML-KEM），204（ML-DSA）の数学的構成を解説**
- 具体的には
  - ➢ ML-KEM，ML-DSAそれぞれの基本構成と処理概要
  - ➢ 数論変換（Number-Theoretic Transform，NTT）の仕組み
  - ➢ NTT処理を含めたML-KEM，ML-DSAの処理手順

# 0. ML-KEM/DSAの安全性を支える計算問題

## ■ LWE（Learning With Errors）問題

- $\mathbb{Z}_q$上の近似連立線形方程式の求解問題
  - ➢ $\mathbb{Z}_q$：法$q$による整数剰余環（整数を$q$で割った余りの集合）
- LWE関係式：$\mathbf{t} \equiv \mathbf{A}^{\mathsf{T}} \cdot \mathbf{s} + \mathbf{e} \pmod{q}$
  - ➢ $\mathbf{A} = (a_{ij}), \mathbf{t} = (t_i)$：公開
  - ➢ $\mathbf{s} = (s_i)$，$\mathbf{e} = (e_i)$：秘密（特に，$\|\boldsymbol{e}\| \approx \sigma$は非常に小さい）
    - ✓ $(n, q, \sigma)$のバランスで，計算量困難性が決まる

$$\begin{cases} 14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 & \pmod{17} \\ 13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 & \pmod{17} \\ 6s_1 + 10s_2 + 13s_3 + s_4 \approx 12 & \pmod{17} \\ 10s_1 + 4s_2 + 12s_3 + 16s_4 \approx 12 & \pmod{17} \\ 9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 & \pmod{17} \\ 3s_1 + 6s_2 + 4s_3 + 5s_4 \approx 16 & \pmod{17} \\ \vdots & \\ 6s_1 + 7s_2 + 16s_3 + 2s_4 \approx 3 & \pmod{17} \end{cases}$$

定式化 ⇕

**解：** $\mathbf{s} = (0, 13, 9, 11) \in \mathbb{Z}_{17}$
（ただし，等号の誤差は$\{0, \pm 1\}$）

$$\begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \equiv \begin{pmatrix} a_{11} & \cdots & a_{n1} \\ \vdots & \ddots & \vdots \\ a_{1m} & \cdots & a_{nm} \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ \vdots \\ e_m \end{pmatrix} \pmod{q}$$

## ■ Module-LWE（Module格子上のLWE）

- 環$R_q$上の階数$k$の加群におけるLWE問題
  - ➢ $R_q = \mathbb{Z}_q[x]/(x^n + 1)$: **NTT乗算**で高速な乗算が可能
  - ➢ $R_q \ni a(x) = a_0 + \cdots + a_{n-1}x^{n-1} \leftrightarrow \boldsymbol{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$
  - ➢ 安全性に応じて，$k = 2, 3, 4$を利用
    - ✓ ML-KEM: $n = 256, q = 3329$
    - ✓ ML-DSA: $n = 256, q = 8380417$

Module-LWE化 ⇓
各成分は$R_q$の元で，$R_q$上で演算

$$\begin{pmatrix} t_1(x) \\ \vdots \\ t_m(x) \end{pmatrix} = \begin{pmatrix} a_{11}(x) & \cdots & a_{k1}(x) \\ \vdots & \ddots & \vdots \\ a_{1m}(x) & \cdots & a_{km}(x) \end{pmatrix} \begin{pmatrix} s_1(x) \\ \vdots \\ s_k(x) \end{pmatrix} + \begin{pmatrix} e_1(x) \\ \vdots \\ e_m(x) \end{pmatrix} \text{ in } R_q$$

# 1. ML-KEMの基本構成と処理概要

## ■ ML-KEMの基本構成

① **K-PKE**: 公開鍵暗号アルゴリズム群
- ➢ Regevの暗号方式が基（cf. $\mathbb{Z}_q \Rightarrow R_q^k$）
- ➢ $R_q$の元で，**十分小さい$\mathbb{Z}_q$係数**は中心二項分布からサンプリング

i. $(x_1, \ldots, x_\eta, y_1, \ldots, y_\eta) \in \{0,1\}^{2\eta}$を一様ランダムにサンプリング（$\eta = 2, 3$）
ii. $\sum(x_i - y_i) \bmod q \in \mathbb{Z}_q$を出力

② **ML-KEM**: 藤崎-岡本（FO）変換でK-PKEをKEMに変換

### K-PKEの処理概要

送信者

受信者

**(1) 鍵生成**
- ランダムに行列 $\mathbf{A} \in R_q^{k \times k}$ を生成
- すべての係数が十分小さい $\mathbf{s}, \mathbf{e} \in R_q^k$ をサンプリング
- $\mathrm{dk} \leftarrow \mathbf{s}$, $\mathrm{ek} \leftarrow (\mathbf{A}, \mathbf{t})$（ただし，$\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \in R_q^k$）

Module-LWE関係式

暗号鍵ek=$(\mathbf{A}, \mathbf{t})$

**(2) 暗号化**
- すべての係数が十分小さい $\mathbf{y}, \mathbf{e}_1 \in R_q^k$ と $e_2 \in R_q$ をサンプリング
- 暗号鍵 $(\mathbf{A}, \mathbf{t})$ と平文 $m \in R_q$ に対し，次を計算：
  $\mathbf{u} = \mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1 \in R_q^k$,
  $v = \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mathrm{Decompress}(m) \in R_q$
- $(\mathbf{u}, v)$: 暗号文

暗号文$(\mathbf{u}, v)$

**(3) 復号**
- 復号鍵 $\mathbf{s}$ を用いて，$w = v - \mathbf{s}^\top \cdot \mathbf{u} \in R_q$
- 圧縮関数で補正：$z = \mathrm{Compress}(w) \in R_q$ (復号文)

※圧縮関数Compressと解凍関数Decompressについては，次スライドで説明

# 1. ML-KEMにおける復号処理の仕組み

## ■ 復号原理

- $\mathbf{s}, \mathbf{e}, \mathbf{y}, \mathbf{e}_1 \in R_q^k$ の成分，および $e_2 \in R_q$ はすべての$\mathbb{Z}_q$係数が小さい多項式

$$
\begin{aligned}
w = v - \mathbf{s}^\top \cdot \mathbf{u} &= \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mathrm{Decompress}(m) - \mathbf{s}^\top \cdot (\mathbf{A}^\top \cdot \mathbf{y} + \mathbf{e}_1) \\
&= \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mathrm{Decompress}(m) - (\mathbf{A}\mathbf{s})^\top \cdot \mathbf{y} - \mathbf{s}^\top \cdot \mathbf{e}_1 \\
&= \mathbf{t}^\top \cdot \mathbf{y} + e_2 + \mathrm{Decompress}(m) - (\mathbf{t} - \mathbf{e})^\top \cdot \mathbf{y} - \mathbf{s}^\top \cdot \mathbf{e}_1 \\
&= \mathrm{Decompress}(m) + \underbrace{e_2 + \mathbf{e}^\top \cdot \mathbf{y} - \mathbf{s}^\top \cdot \mathbf{e}_1}_{r \in R_q\text{: すべての係数が小さい多項式}} \approx \mathrm{Decompress}(m) \in R_q
\end{aligned}
$$

Module-LWE関係式
$\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$

## ■ 復号文から平文を抽出（ゆらぎ補正）

- 平文 $m = m_0 + m_1 x + \cdots + m_{n-1} x^{n-1}$ $(m_i \in \{0,1\})$ に対して
  - ➢ 解凍: $\mathrm{Decompress}(m) = \mu_0 + \mu_1 x + \cdots + \mu_{n-1} x^{n-1}$ $\left(\mu_i = \left\lceil \frac{q}{2} \cdot m_i \right\rfloor \in \mathbb{Z}_q\right)$
- 上記 $w = w_0 + w_1 x + \cdots + w_{n-1} x^{n-1}$ に対して
  - ➢ 圧縮: $\mathrm{z} = \mathrm{Compress}(w) = z_0 + z_1 x + \cdots + z_{n-1} x^{n-1}$ $\left(z_i = \left\lceil \frac{2}{q} \cdot w_i \right\rfloor \bmod 2 \in \{0,1\}\right)$

$\Rightarrow \mathrm{Compress}(w) = \mathrm{Compress}(\mathrm{Decompress}(m) + r) = m$（**平文抽出**）

ゆらぎ分（すべての$\mathbb{Z}_q$係数は十分小さい）

# 1. ML-KEMにおけるNTT処理（1/2）

## ■ NTT：Number-Theoretic Transform（数論変換）

- $R_q = \mathbb{Z}_q[x]/(x^n+1)$ 上の高速乗算を可能とする変換（パラメータ依存）
  - ➢ $\mathbb{C}$上の高速フーリエ変換と同じアイデア，NTTはその$\mathbb{Z}_q$上版
- ML-KEMのパラメータ設定：$n = 256 = 2^8,\ \ q = 3329$
  - ➢ $q-1 = 3328 = \mathbf{2^8}\cdot 13$ より，$\mathbb{Z}_q$は1の原始 $n$ 乗根 $\zeta$ を含む $(\zeta = 17 \bmod q)$
  - ➢ $N = {}^{n}/_{2} = 128$ とおくと，$\zeta^{(2i+1)N} \equiv -1 \bmod q\ (i = 0, 1, \dots, N-1)$
  - ➢ $x^n+1$ は $N$個の**2次式の積に分解**：$x^n+1 = \prod_{i=0}^{N-1}(x^2-\zeta^{2i+1})$ $(\because (x^2)^N = \left(\zeta^{2i+1}\right)^N \equiv -1)$

$$R_q = \mathbb{Z}_q[x]/(x^n+1) \quad \cong \quad T_q = \bigoplus_{i=0}^{N-1} \mathbb{Z}_q[x]/\left(x^2-\zeta^{2i+1}\right)$$

**NTT空間**

各$i$において $x^2 = \zeta^{2i+1}$ を代入（1次式で表せる）

$$f = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} = f_e + f_o x$$

**NTT変換** →

$$\hat{f} = \left(f \bmod (x^2-\zeta^{2i+1})\right)_{i=0}^{N-1} = \left(\hat{f}_e + \hat{f}_o x\right)_{i=0}^{N-1}$$

ただし

$$f_e = f_0 + f_2 x^2 + \cdots + f_{n-2}x^{n-2} \text{（偶）}$$
$$f_o = f_1 + f_3 x^2 + \cdots + f_{n-1}x^{n-2} \text{（奇）}$$

$x^2 = \zeta^{2i+1}$ を代入 →

係数ベクトルで表すと行列 **B** との乗算

ただし

$$\hat{f}_e = \sum f_{2j}\zeta^{(2i+1)j}$$
$$\hat{f}_o = \sum f_{2j+1}\zeta^{(2i+1)j}$$

$$\mathbf{B} = \begin{pmatrix} 1 & \zeta & \zeta^2 & \cdots & \zeta^{N-1} \\ 1 & \zeta^3 & \zeta^6 & \cdots & \zeta^{3(N-1)} \\ 1 & \zeta^5 & \zeta^{10} & \cdots & \zeta^{5(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{2N-1} & \zeta^{2(2N-1)} & \cdots & \zeta^{(N-1)(2N-1)} \end{pmatrix}$$

FIPS 203, Appendix Aで事前計算済み

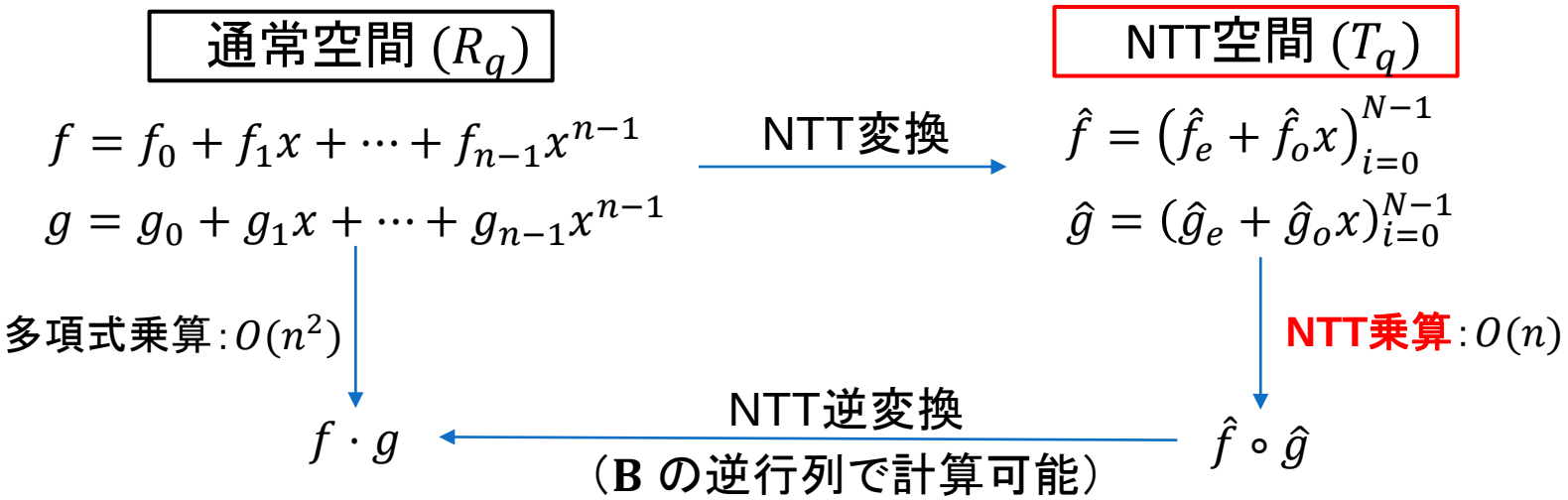
# 1. ML-KEMにおけるNTT処理（2/2）

## ■ NTT乗算

- NTT空間の2つの元 $\hat{f} = \left(\hat{f}_e + \hat{f}_o x\right)_{i=0}^{N-1}, \hat{g} = (\hat{g}_e + \hat{g}_o x)_{i=0}^{N-1}$ に対して
- $\hat{f} \circ \hat{g} = \left((\hat{f}_e + \hat{f}_o x) \cdot (\hat{g}_e + \hat{g}_o x) \bmod (x^2 - \zeta^{2i+1})\right)_{i=0}^{N-1} = \left(\left(\hat{f}_e \hat{g}_e + \hat{f}_o \hat{g}_o \zeta^{2i+1}\right) + \left(\hat{f}_e \hat{g}_o + \hat{f}_o \hat{g}_e\right) x\right)_{i=0}^{N-1}$

$x^2 = \zeta^{2i+1}$ に注意

  - ➢ 成分ごとの演算で，効率的に計算可能




## ■ NTT加算

- NTT乗算と同様，成分ごとの加算で計算

# 1. ML-KEMにおける主な内部処理(1/3)

## ■ K-PKEの鍵生成アルゴリズム(FIPS 203, Algorithm 13)の処理概要

- 入力:乱数 $d$
- 出力:暗号鍵 ek と復号鍵 dk
  - ➢ $\mathbb{Z}_q$係数が十分小さい$R_q$の元はすべて通常空間上で生成
  - ➢ 一方，鍵ペアはNTT空間上で構成

| | 処理説明 | 通常空間($R_q$) | NTT空間($T_q$) |
|---|---|---|---|
| 1 | NTT空間上で公開鍵行列を生成<br>(乱数 $d$ から一意に生成) | | $\hat{\mathbf{A}} = \left(\hat{\mathbf{A}}[i,j]\right)_{0\le i,j<k}$ |
| 2 | 秘密と誤差をサンプリング<br>(各$\mathbb{Z}_q$係数は中心二項分布から) | $\mathbf{s} = (\mathbf{s}[i])_{0\le i<k}$<br>$\mathbf{e} = (\mathbf{e}[i])_{0\le i<k}$ | |
| 3 | NTT変換 | | $\hat{\mathbf{s}} = \left(\mathrm{NTT}(\mathbf{s}[i])\right)_{0\le i<k}$<br>$\hat{\mathbf{e}} = \left(\mathrm{NTT}(\mathbf{e}[i])\right)_{0\le i<k}$ |
| 4 | NTT空間上でModule-LWE関係式<br>を計算(**NTT乗算**を利用) | | $\hat{\mathbf{t}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}} = \left(\sum_{j=0}^{k-1} \hat{\mathbf{A}}[i,j] \circ \hat{\mathbf{s}}[j] + \hat{\mathbf{e}}[i]\right)_{0\le i<k}$ |
| 5 | 鍵ペアを構成 | | $\mathrm{ek} = \left(\hat{\mathbf{A}}, \hat{\mathbf{t}}\right),\ \ \mathrm{dk} = \hat{\mathbf{s}}$ |

# 1. ML-KEMにおける主な内部処理(2/3)

## ■ K-PKEの暗号化アルゴリズム(FIPS 203, Algorithm 14)の処理概要

- 入力:暗号鍵 $\mathrm{ek} = (\hat{\mathbf{A}}, \hat{\mathbf{t}})$, 平文 $m \in R_q$
- 出力:暗号文 $c$

| | 処理説明 | 通常空間($R_q$) | NTT空間($T_q$) |
|---|---|---|---|
| 1 | $\mathbb{Z}_q$係数が小さいベクトルを生成 | $\mathbf{y} = (\mathbf{y}[i])_{0 \le i < k}, \mathbf{e}_1 = (\mathbf{e}_1[i])_{0 \le i < k}$ | |
| 2 | $\mathbb{Z}_q$係数が小さい元を生成 | $e_2 \in R_q$ | |
| 3 | NTT変換 | | $\hat{\mathbf{y}} = \left(\mathrm{NTT}(\mathbf{y}[i])\right)_{0 \le i < k}$ |
| 4 | NTT空間上で演算<br>(**NTT乗算**を利用) | | $\hat{\mathbf{w}} = \hat{\mathbf{A}}^{\mathsf{T}} \circ \hat{\mathbf{y}} = \left(\sum_{j=0}^{k-1} \hat{\mathbf{A}}[j,i] \circ \hat{\mathbf{y}}[j]\right)_{0 \le i < k}$<br>$\hat{h} = \hat{\mathbf{t}}^{\mathsf{T}} \circ \hat{\mathbf{y}} = \sum_{j=0}^{k-1} \hat{\mathbf{t}}[j] \circ \hat{\mathbf{y}}[j] \in T_q$ |
| 5 | NTT逆変換 | $\mathbf{w} = \left(\mathrm{NTT}^{-1}(\hat{\mathbf{w}}[i])\right)_{0 \le i < k}$<br>$h = \mathrm{NTT}^{-1}(\hat{h}) \in R_q$ | |
| 6 | 暗号文 $c = (\mathbf{u}, v)$ を計算<br>(ただし, $\mu = \mathrm{Decompress}(m)$) | $\mathbf{u} = \mathbf{w} + \mathbf{e}_1 \quad (= \mathbf{A}^{\mathsf{T}} \cdot \mathbf{y} + \mathbf{e}_1)$<br>$v = h + e_2 + \mu \quad (= \mathbf{t}^{\mathsf{T}} \cdot \mathbf{y} + e_2 + \mu)$ | |

# 1. ML-KEMにおける主な内部処理(3/3)

## ■ K-PKEの復号アルゴリズム(FIPS 203, Algorithm 15)の処理概要

- 入力:復号鍵 $\mathrm{dk} = \hat{\mathbf{s}}$, 暗号文 $c = (\mathbf{u}, v)$
- 出力:復号文 $z$

| | 処理説明 | 通常空間($R_q$) | NTT空間($T_q$) |
|---|---|---|---|
| 1 | $\mathbf{u}$ をNTT変換 | | $\hat{\mathbf{u}} = \mathrm{NTT}(\mathbf{u})$ |
| 2 | **NTT乗算** | | $\hat{r} = \hat{\mathbf{s}}^{\mathsf{T}} \circ \hat{\mathbf{u}} \in T_q$ |
| 3 | NTT逆変換 | $r = \mathrm{NTT}^{-1}(\hat{r}) \in R_q$ | |
| 4 | 通常空間で演算<br>圧縮関数でゆらぎ補正 | $w = v - r\ (= v - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{u})$<br>$z = \mathrm{Compress}(w)$ | |

## ■ 復号文の(再)確認

- $w = v - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{u} = v - \mathbf{s}^{\mathsf{T}} \cdot (\mathbf{A}^{\mathsf{T}} \cdot \mathbf{y} + \mathbf{e}_1) = v - (\mathbf{A}\mathbf{s})^{\mathsf{T}} \cdot \mathbf{y} - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{e}_1$
  $= (\mathbf{t}^{\mathsf{T}} \cdot \mathbf{y} + e_2 + \mu) - (\mathbf{t} - \mathbf{e})^{\mathsf{T}} \cdot \mathbf{y} - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{e}_1 = \mu + (e_2 + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{y} - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{e}_1)$
- $z = \mathrm{Compress}(w) = \mathrm{Compress}\big(\mu + (e_2 + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{y} - \mathbf{s}^{\mathsf{T}} \cdot \mathbf{e}_1)\big) = \mathrm{Compress}(\mu) = m$ (**平文に一致**)

ゆらぎ分

# 2. ML-DSAの処理概要

## ■ Fiat-Shamir型の署名方式

- $R_q = \mathbb{Z}_q[x]/(x^n+1),\ n = 256,\ q = 8380417 = 2^{23} - 2^{13} + 1$（素数）
- $c$: チャレンジ，$\mathbf{z}$:レスポンス，$\mathbf{w}_1$:コミットメント

署名者

**(1) 鍵生成**

- ランダムに行列 $\mathbf{A} \in R_q^{k \times \ell}$ を生成
- すべての$\mathbb{Z}_q$係数が十分小さい $\mathbf{s}_1 \in R_q^{\ell}, \mathbf{s}_2 \in R_q^{k}$ をサンプリング
- $\mathrm{sk} \leftarrow (\mathbf{s}_1, \mathbf{s}_2),\ \mathrm{pk} \leftarrow (\mathbf{A}, \mathbf{t})$（ただし，$\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in R_q^k$）

Module-LWE関係式

公開鍵$(\mathbf{A}, \mathbf{t})$を送信

検証者

**(2) 署名生成**

- すべての$\mathbb{Z}_q$係数が十分小さい $\mathbf{y} \in R_q^{\ell}$ をサンプリング
- $\mathbf{w}_1 \leftarrow \mathrm{HighBits}\ (\mathbf{w})$: $\mathbf{w} = \mathbf{A}\mathbf{y} \in R_q^k$ の各成分の上位ビット
- $c \leftarrow H(m||\mathbf{w}_1) \in R$: 平文 $m$ と $\mathbf{w}_1$ の連結データのハッシュ値
- $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1 \in R_q^{\ell}$
  （すべての$\mathbb{Z}_q$係数が十分小さくなるまで，$\mathbf{y}$ を取り直す）
- $\sigma \leftarrow (\mathbf{z}, c)$: 署名

署名付きの平文 $(m, \sigma)$

**(3) 署名検証**

- $\mathbf{w}_1{}' \leftarrow \mathrm{HighBits}\ (\mathbf{A}\mathbf{z} - c\mathbf{t})$
- $\mathbf{z}$ が十分短い，かつ $c = H(m||\mathbf{w}_1{}')$ が成立 ⇒ 検証成功
  （そうでなければ，検証失敗）

# 2. ML-DSAにおける署名検証の正当性

■ 正当な署名 $(\mathbf{z}, c)$ に対して

- $\mathbf{z} = \boldsymbol{y} + c\mathbf{s}_1 \in R_q^{\ell}$ のすべての$\mathbb{Z}_q$係数は十分短いのでOK
- コミットメント $\mathbf{w}_1$ について
  - ➢ Module-LWE関係式 $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ に注意

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

⇩ 上位ビットが一致

$$\mathbf{w}_1' = \mathrm{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}) = \mathrm{HighBits}(\mathbf{w}) = \mathbf{w}_1$$

⇩ 平文$m' = m$ であれば

$$c' = H(m' || \mathbf{w}_1') = H(m || \boldsymbol{w_1}) = c \text{（検証成功）}$$

署名偽造の可能性
- $\mathbf{s}_1$を知っていると偽造可能
- 一方，Module-LWE問題が困難なので，$\mathbf{s}_1$は分からない

# 2. ML-DSAにおけるNTT処理

## ■ ML-DSAのパラメータ設定：$n = 256 = 2^8,\ \ q = 2^{23} - 2^{13} + 1$

- $q - 1 = 2^{23} - 2^{13} = \mathbf{2^{13}} \cdot 1023$ より，$\mathbb{Z}_q$は1の原始 $2n$ 乗根 $\zeta$ を含む $(\zeta = 1753 \bmod q)$
- $x^n + 1$ は $n$個の**1次式の積に分解**：$x^n + 1 = \prod_{i=0}^{n-1}\left(x - \zeta^{2i+1}\right)$ （cf. ML-KEMでは，2次式の積に分解）

$$R_q = \mathbb{Z}_q[x]/(x^n + 1) \quad \cong \quad T_q = \bigoplus_{i=0}^{n-1} \mathbb{Z}_q[x]/\left(x - \zeta^{2i+1}\right)$$

**NTT空間**

$$f = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} \xrightarrow{\text{NTT変換}} \hat{f} = \left(f \bmod (x - \zeta^{2i+1})\right)_{i=0}^{n-1} = \left(f\left(\zeta^{2i+1}\right)\right)_{i=0}^{n-1}$$

各$i$において $x = \zeta^{2i+1}$ を代入

係数ベクトル表現

$$\boldsymbol{f} = (f_0, f_1, \ldots, f_{n-1}) \xrightarrow{\text{行列Bとの乗算}} \hat{\boldsymbol{f}} = \mathbf{B} \cdot \boldsymbol{f}^{\top}$$

$$\mathbf{B} = \begin{pmatrix} 1 & \zeta & \zeta^2 & \cdots & \zeta^{n-1} \\ 1 & \zeta^3 & \zeta^6 & \cdots & \zeta^{3(n-1)} \\ 1 & \zeta^5 & \zeta^{10} & \cdots & \zeta^{5(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{2n-1} & \zeta^{2(2n-1)} & \cdots & \zeta^{(n-1)(2n-1)} \end{pmatrix}$$

各成分はFIPS 204, Appendix Bで事前計算済み

## ■ NTT演算

- NTT乗算・加算ともに，成分ごとの演算
- 成分ごとの演算で，効率的に計算可能

# 2. ML-DSAにおける鍵生成

## ■ 鍵生成アルゴリズム（FIPS 204, Algorithm 6）の処理概要

- 入力：乱数 $\xi$
- 出力：公開鍵 pk と秘密鍵 sk

| | 処理説明 | 通常空間（$R_q$） | NTT空間（$T_q$） |
|---|---|---|---|
| 1 | NTT表現の公開鍵行列を生成<br>（乱数 $\rho$ ($\leftarrow \xi$) から一意的に生成） | | $\widehat{\mathbf{A}} \in \left(T_q\right)^{k \times \ell}$ |
| 2 | すべての係数が$[-\eta, \eta]$内の$R$の元の組<br>（棄却サンプリングを利用） | $(\mathbf{s}_1, \mathbf{s}_2) \in S_\eta^\ell \times S_\eta^k$ | |
| 3 | NTT変換・NTT乗算 | | $\hat{\mathbf{z}} = \widehat{\mathbf{A}} \circ \mathrm{NTT}(\boldsymbol{s}_1)$ |
| 4 | NTT逆変換・$R_q$上の加算<br>（Module-LWE関係式を計算） | $\mathbf{t} = \mathrm{NTT}^{-1}(\hat{\mathbf{z}}) + \mathbf{s}_2 \ (= \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2)$ | |
| 5 | 各成分を上位と下位ビットに分割 | $(\mathbf{t}_1, \mathbf{t}_0) = \mathrm{Power2Round}(\mathbf{t})$ | |
| 6 | 公開鍵を構成し，そのハッシュ値を計算 | $\mathrm{pk} = (\rho, \mathbf{t}_1),\ tr = H(\mathrm{pk})$ | |
| 7 | 秘密鍵を構成 | $\mathrm{sk} = (\rho, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$ | |

# 2. ML-DSAにおける署名生成(1/2)

## ■ 署名生成アルゴリズム(FIPS 204, Algorithm 7)の処理概要

- 入力:秘密鍵 $\mathrm{sk} = (\rho, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$ と平文 $m$
- 出力:署名 $\sigma$
  - $\tilde{c}$:チャレンジ, $\mathbf{z}$:レスポンス, $\mathbf{w}_1$:コミットメント

| | 処理説明 | 通常空間($R_q$) | NTT空間($T_q$) |
|---|---|---|---|
| 1 | NTT変換 | | $\hat{\mathbf{s}}_1 = \mathrm{NTT}(\mathbf{s}_1), \hat{\mathbf{s}}_2 = \mathrm{NTT}(\mathbf{s}_2), \hat{\mathbf{t}}_0 = \mathrm{NTT}(\mathbf{t}_0)$ |
| 2 | NTT表現の公開鍵行列を生成<br>(乱数 $\rho$ から同じものを復元) | | $\hat{\mathbf{A}} \in (T_q)^{k \times \ell}$ |
| 3 | ハッシュ値を計算 | $\mu = H(tr \| \| m)$ | |
| 4 | $\mathbb{Z}_q$係数が小さいベクトルを生成+NTT変換 | $\mathbf{y} = (\mathbf{y}[i])_{0 \le i < \ell}$ → | $\hat{\mathbf{y}} = (\mathrm{NTT}(\mathbf{y}[i]))_{0 \le i < \ell}$ |
| 5 | NTT乗算+NTT逆変換 | $\mathbf{w} = \mathrm{NTT}^{-1}(\hat{\mathbf{w}}) (= \mathbf{A}\mathbf{y})$ ← | $\hat{\mathbf{w}} = \hat{\mathbf{A}} \circ \hat{\mathbf{y}} = \left( \sum_{j=0}^{\ell-1} \hat{\mathbf{A}}[i,j] \circ \hat{\mathbf{y}}[j] \right)_{0 \le i < k}$ |
| 6 | 上位ビットの抽出 | $\mathbf{w}_1 = \mathrm{HighBits}(\mathbf{w})$ | |
| 7 | ハッシュ値の計算 | $\tilde{c} = H(\mu \| \| \mathbf{w}_1)$ | |
| 8 | $\mathbb{Z}_q$係数が$\{-1, 0, 1\}$の元を生成+NTT変換 | $c = \mathrm{SampleInBall}(\tilde{c})$ → | $\hat{c} = \mathrm{NTT}(c) \in T_q$ |

※ SampleInBall() は, $\tilde{c}$ を引数とする$\mathbb{Z}_q$係数を$\{-1, 0, 1\}$からサンプルした多項式 $c \in R_q$ を生成(FIPS 204, Algorithm 29)

# 2. ML-DSAにおける署名生成(2/2)

## ■ 処理の続き

- ステップ10の $\mathbf{z} \in R_q^\ell$ の各成分の$\mathbb{Z}_q$係数が十分小さくなるまで，ステップ4の $\mathbf{y} \in R_q^\ell$ などを取りなおす

| | 処理説明 | 通常空間 | NTT空間($T_q$) |
|---|---|---|---|
| 9 | NTT乗算＋NTT逆変換 | $c\mathbf{s}_1 = \mathrm{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_1), c\mathbf{s}_2 = \mathrm{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_2)$ ← | $\hat{c} \circ \hat{\mathbf{s}}_1, \hat{c} \circ \hat{\mathbf{s}}_2$ |
| 10 | $R_q$上での加算 | $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1 \in R_q^\ell$ | |
| 11 | 下位ビットの抽出 | $\mathbf{r}_0 = \mathrm{LowBits}(\mathbf{w} - c\mathbf{s}_2) \in R_q^k$ | |
| 12 | NTT乗算＋NTT逆変換 | $c\mathbf{t}_0 = \mathrm{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{t}}_0) \in R_q^k$ ← | $\hat{c} \circ \hat{\mathbf{t}}_0$ |
| 13 | 不一致真理値ベクトルの計算 | $\mathbf{h} = \mathrm{MakeHint}(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0) \in \{0,1\}^k$ | |
| 14 | 署名の構成 | $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ | |

## ■ ヒント $\mathbf{h}$ の生成(FIPS 204, Algorithm 39)

- $\mathbf{w}_1 = \mathrm{HighBits}(\mathbf{w})$のヒント ⇒ 署名サイズの削減
- $\mathrm{HighBits}(\mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0)$ と $\mathrm{HighBits}(\mathbf{w} - c\mathbf{s}_2)$ の各成分の不一致真理値を計算(ベクトル長は$k$)
  - $\mathbf{h}$ は $\mathbf{w} - c\mathbf{s}_2$ に対する $c\mathbf{t}_0$ による桁上がり情報(ヒント情報)

# 2. ML-DSAにおける署名検証

## ■ 署名検証アルゴリズム（FIPS 204, Algorithm 8）の処理概要

- 入力：公開鍵 $\mathrm{pk} = (\rho, \mathbf{t}_1)$，署名 $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ 付きの平文 $m'$
- 出力：真偽値

$d$:上位と下位ビットを分割する閾値

| | 処理説明 | 通常空間（$R_q$） | NTT空間（$T_q$） |
|---|---|---|---|
| 1 | ハッシュ値の計算 | $tr = H(\mathrm{pk}), \mu = H(tr \| \| m')$ | |
| 2 | | $c' = \mathrm{SampleInBall}(\tilde{c}) \in R_q$ | |
| 3 | NTT変換 | | $\hat{\mathbf{z}} = \mathrm{NTT}(\mathbf{z}), \hat{c}' = \mathrm{NTT}(c'), \hat{\mathbf{u}} = \mathrm{NTT}(\mathbf{t}_1 \cdot 2^d)$ |
| 4 | NTT演算＋NTT逆変換 | $\mathbf{w}'_{\mathrm{App}} = \mathrm{NTT}^{-1}(\hat{\mathbf{v}})\ (= \mathbf{Az} - c'\mathbf{t}_1 \cdot 2^d)$ | $\hat{\mathbf{v}} = \hat{\mathbf{A}} \circ \hat{\mathbf{z}} - \hat{c}' \circ \hat{\mathbf{u}}$　（$\hat{\mathbf{A}}$ は $\rho$ から復元） |
| 5 | ヒントからコミットメントを復元 | $\mathbf{w}'_1 = \mathrm{UseHint}(\mathbf{h}, \mathbf{w}'_{\mathrm{App}})$ | |
| 6 | ハッシュ値の計算＋判定 | $\tilde{c}' = H(\mu \| \| \mathbf{w}'_1)$<br>$\tilde{c} = \tilde{c}'$なら「真」，それ以外なら「偽」 | |

## ■ ヒント利用（FIPS 204, Algorithm 40）と判定

- 署名が正当なら，$\mathbf{w}'_{\mathrm{App}} = \mathbf{A}(\mathbf{y} + c\mathbf{s}_1) - c\mathbf{t}_1 \cdot 2^d = \mathbf{w} + c(\mathbf{t} - \mathbf{s}_2) - c\mathbf{t}_1 \cdot 2^d = \mathbf{w} - c\mathbf{s}_2 - c\mathbf{t}_0$
  - $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1, c = c', \mathbf{w} = \mathbf{Ay}, \mathbf{t} = \mathbf{As}_1 + \mathbf{s}_2, \mathbf{t} = \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$ を利用
  - ヒント情報 $\mathbf{h}$ を利用すると，$\mathbf{w}'_1 = \mathrm{UseHint}(\mathbf{h}, \mathbf{w}'_{\mathrm{App}}) = \mathrm{HighBits}(\mathbf{w} - c\mathbf{s}_2) = \mathrm{HighBits}(\mathbf{w}) = \mathbf{w}_1$

  ⇒ さらに $m = m'$ であれば，$\tilde{c}' = H(\mu \| \| \mathbf{w}'_1) = H(\mu \| \| \mathbf{w}_1) = \tilde{c}$（**検証成功**）


https://www.cryptrec.go.jp/

# 3. パラメータセットと安全性レベル

## ■ ML-KEMのパラメータセット

- $n = 256, q = 3329$は共通，階数$k$のサイズで安全性レベルが異なる

| | 暗号パラメータ | | | サイズ（単位：バイト） | | | | 安全性レベル |
|---|---|---|---|---|---|---|---|---|
| | $n$ | $q$ | $k$ | カプセル化鍵 | デカプセル化鍵 | 暗号文 | 共有の秘密鍵 | |
| ML-KEM-512 | 256 | 3329 | 2 | 800 | 1,632 | 768 | 32 | レベル1 |
| ML-KEM-768 | 256 | 3329 | 3 | 1,184 | 2,400 | 1,088 | 32 | レベル3 |
| ML-KEM-1024 | 256 | 3329 | 4 | 1,568 | 3,168 | 1,568 | 32 | レベル5 |

## ■ ML-DSAのパラメータセット

- $n = 256, q = 8380417$は共通，$(k, \ell)$のサイズで安全性レベルが異なる

| | 暗号パラメータ | | | サイズ（単位：バイト） | | | 安全性レベル |
|---|---|---|---|---|---|---|---|
| | $n$ | $q$ | $(k, \ell)$ | 秘密鍵 | 公開鍵 | 署名 | |
| ML-DSA-44 | 256 | 8380417 | (4, 4) | 2,560 | 1,312 | 2,420 | レベル2 |
| ML-DSA-65 | 256 | 8380417 | (6, 5) | 4,032 | 1,952 | 3,309 | レベル3 |
| ML-DSA-87 | 256 | 8380417 | (8, 7) | 4,896 | 2,592 | 4,627 | レベル5 |