**Krystal-Kyber**

Rana Barua, IAI, TCG CREST, Kolkata

# 1 Introduction

**Kyber** is a quantam-safe Key Encapsulation Mechnism(KEM) that has been standardized by NIST in FIPS 203. There it is called Module-Lattice-based Key Encapsulation Mechanism (ML-KEM)

# 2 Preliminaries

We denote by $\mathcal{B}$ the set $\{0, 1, \ldots, 255\}$ , i.e., the set of 8-bit unsigned integers (bytes). For two byte arrays $a$ and $b$ we denote by $a||b$ the concatenation of a and b. For a byte array $a$ we denote by $a + k$ the byte array starting at byte k of a ,with indexing starting at zero. **BytesToBits** is a function that takes as input an array of $\ell$ bytes and produces as output an array of $8\ell$ bits.

## 2.1 Modular Reduction

Given an even(odd) positive integer $\alpha$, let $r'$ be the unique integer in the range $-\alpha/2 < r' \leq \alpha/2$(resp.$(\alpha - 1)/2 \leq r' \leq (\alpha + 1)/2$) such that $r' \equiv r \bmod \alpha$. In this case we write

$$r' = r \bmod {}^{\pm}\alpha.$$

We call it a centred reduction modulo $\alpha$.

## 2.2 Norm

For an element $a \in \mathbb{Z}_q$, $||a||_\infty$ denotes $|a \bmod {}^{\pm}q|$. The $\ell_\infty$ and $\ell_2$ norms of an element $w = w_0 + w_1X + \ldots + w_{n-1}X^{n-1} \in \mathcal{R}$ are as follows.

$$||w||_\infty = Max_i||w_i||_\infty; \ ||w|| = \sqrt{||w_0||_\infty^2 + \ldots + ||w_{n-1}||_\infty^2}.$$

For a vector $\mathbf{w} = (w_1, \ldots, w_k) \in \mathcal{R}^k$ the norms are similarly defined. We define $S_\eta$ by

$$S_\eta = \{w \in \mathcal{R} : ||w||_\infty \leq \eta\}.$$

We also define $\tilde{S}_\eta$ by

$$\tilde{S}_\eta = \{w \bmod {}^{\pm}2\eta : w \in \mathcal{R}\}.$$

## 2.3 Compression and Decompression

For $d < \lceil \log q \rceil$, define a function

$$\mathbf{Compress}_q(., d) : \mathbb{Z}_q \to \{0, 1, \ldots, 2^d - 1\}$$

as follows

$$\mathbf{Compress}_q(x, d) = \lceil (2^d/q)x \rfloor \bmod 2^d.$$

We also define $\mathbf{Decompress}_q(x, d)$ by

$$\mathbf{Decompress}_q(x, d) = \lceil (q/2^d).x \rfloor.$$

One can check that if

$$x' = \mathbf{Decompress}_q(\mathbf{Compress}_q(x, d), d),$$

then

$$|x' - x \bmod {}^{\pm}q| \leq \lceil (q/2^{d+1}) \rfloor.$$

## 2.4 Symmetric primitives

Kyber uses a pseudorandom function $\mathbf{PRF} : \mathcal{B}^{32} \times \mathcal{B} \to \mathcal{B}^*$ and an extendable output function $\mathbf{XOF} : \mathcal{B}^* \times \mathcal{B} \times \mathcal{B} \to \mathcal{B}^*$ . Kyber also uses two hash functions $\mathbf{H} : \mathcal{B}^* \to \mathcal{B}^{32}$ and $\mathbf{G} : \mathcal{B}^* \to \mathcal{B}^{32} \times \mathcal{B}^{32}$ and a key-derivation function $\mathbf{KDF} : \mathcal{B}^* \to \mathcal{B}^*$.

## 2.5 Uniform sampling in $\mathcal{R}_q$.

Kyber uses a deterministic algorithm to sample elements in $\mathcal{R}_q$ that are statistically close to a uniformly random distribution. Kyber uses a function $\mathbf{Parse} : \mathcal{B}^* \to \mathcal{R}_q$ which receives as input a byte stream $B = b_0 b_1 b_2 \ldots$ and outputs the NTT-representation $\hat{\mathbf{a}} = \hat{a}_0 + \hat{a}_1 X + \ldots + \hat{a}_{n-1} X^{n-1} \in \mathcal{R}_q$ of $\mathbf{a} \in \mathcal{R}_q$.

## 2.6 Sampling from a binomial distribution.

Kyber uses a central binomial distribution (CBD) $B_\eta$, for $\eta = 2$ or $\eta = 3$, as follows.
Choose uniformly at random $(a_1, \ldots, a_\eta, b_1, \ldots, b_\eta) \in \{0, 1\}^{2\eta}$ and output

$c = \sum_{i=1}^{\eta}(a_i - b_i)$. One can check that $c \in [-\eta, \eta]$ and that for any $j \in [-\eta, \eta]$, $\text{Prob}[c = j] = \begin{pmatrix} 2\eta \\ \eta + j \end{pmatrix} / 2^{2\eta}$.

We say that an element $f \in \mathcal{R}_q$ is sampled according to $B_\eta$, we mean that each coefficient is sampled according to $B_\eta$.

Kyber also defines a function $CBD_\eta : \mathcal{B}^{64\eta} \to \mathcal{R}_q$, which takes as input a $64\eta$ length byte array and output a polynomial in $\mathcal{R}_q$. This is done as follows. Convert the byte array, using **BytesToBits**, into a bit array of length $512\eta$, say $\beta_0, \ldots, \beta_{512\eta-1}$. Take the first $2\eta$ bits $\beta_0, \ldots, \beta_{2\eta-1}$ and apply $B_\eta$ to obtain $f_0$. Takes the next $2\eta$ bits $\beta_{2\eta}, \ldots, \beta_{4\eta-1}$ and apply $B_\eta$ to obtain $f_1$ and so on . Finally, output the polynomial $\mathbf{f} = f_0 + f_1 X + \ldots + f_{255} X^{255} \in \mathcal{R}_q$.

## 2.7 Encoding and decoding

The function $\mathbf{Decode}_\ell$ takes as input an array of $32\ell$ bytes and outputs a polynomial $f_0 + F_1 X + \ldots + f_{255} X^{255} \in \mathcal{R}_q$, where each $f_i \in \{0, \ldots, 2^\ell - 1\}$. Using **BytesToBits**, obtain a bit array of length $256\ell$ *viz.* $\beta_0, \ldots, \beta_{256\ell-1}$. The first $\ell$ bits $\beta_0, \ldots, \beta_{\ell-1}$ represents $f_0$. The next $\ell$ bits $\beta_\ell, \ldots, \beta_{2\ell-1}$ represents $f_1$ and so on. This yields a polynomial

$$\mathbf{f} = f_0 + f_1 X + \ldots + f_{255} X^{255} \in \mathcal{R}_q,$$

where each $f_i \in \{0, \ldots, 2^\ell - 1\}$.

$\mathbf{Encode}_\ell$ is just the inverse of $\mathbf{Decode}_\ell$.

# 3 NTT and Inverse NTT

We will be considering multiplication in the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $n = 2^k$ is a power of 2 and $q$ is a prime such that $q \equiv 1 \bmod n$. This ensures that a primitive $n$-th root of unity exists in $\mathbb{Z}_q$ i.e. an element $\zeta \in \mathbb{Z}_q$ such that $\zeta^n = 1 \bmod q$ but for $0 < k < n, \zeta^k \neq 1 \bmod q$. Note that a typical element $\mathbf{a} \in \mathcal{R}_q$ is a polynomial of degree at most $n - 1$. If

$$\mathbf{a} = \sum_{i=0}^{n-1} a_i X^i,$$

then we identify $\mathbf{a}$ with the vector of coefficients $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$ and we write $\mathbf{a} = (a_0, \ldots, a_{n-1})(X)$. Recall that multiplication in $\mathcal{R}_q$ is defined as follows.

$$X.X^i = \begin{cases} X^{i+1} & \text{if } \ i+1 < n \\ -1 & \text{if } \ i+1 = n \end{cases}.$$

**In Kyber** $n = 2^8 = 256$ **and** $q = 3329$ **so that** $q - 1 = 2^8.13$**. Hence** $256|q-1$ **but** $512$ **does not divide** $q-1$**. These are fixed throughout the notes.** Fix a 256th root of unity $\zeta$ modulo $q$. Concretely, let $\zeta = 17$ be the smallest primitive root of unity.. Then $\zeta, \zeta^3, \zeta^5, \ldots, \zeta^{255}$ are all the roots of $X^{128}+1$ Hence, $X^{128}+1$ completely splits as

$$X^{128} + 1 = \prod_{i=0}^{127}(X - \zeta^{2i+1}).$$

Consequently,

$$X^{256} + 1 = \prod_{i=0}^{127}(X^2 - \zeta^{2i+1}).$$

We now show

**Lemma 3.1.** For every $i, 0 \le i \le 127, (X^2 - \zeta^{2i+1})$ is irreducible over $\mathbb{Z}_q$.

*Proof.* If not, then $X^2 - \zeta^{2i+1}$ has a root $c \in \mathbb{Z}_q$. Hence, in $\mathbb{Z}_q$,

$$(c^2)^{128} = (\zeta^{128})^{2i+1} = (-1)^{2i+1} = -1.$$

Hence the order of $c \in \mathbb{Z}_q$ does not divide 256. On the other hand

$$(c^2)^{256} = (\zeta^{256})^{2i+1} = 1^{2i+1} = 1.$$

Hence the order of $c$ divides 512. Hence the order of $c$ is 512. This is not possible, since 512 does not divide $q-1$. □

Now let $\zeta_i = \zeta^{2br(i)+1}$, where $br(i)$ denotes the bit reversal of the unsigned 7-bit integer $i$. From above, we have

$$X^{256} + 1 = \prod_{i=0}^{127}(X^2 - \zeta^{2i+1}) = \prod_{i=0}^{127}(X^2 - \zeta_i). \tag{3.1}$$

**Definition 3.1.** Define $\mathcal{Q}_i = \mathbb{Z}_q[X]/(X^2 - \zeta_i)$ and $T_q = \mathcal{Q}_0 \times \mathcal{Q}_1 \times \ldots \times \mathcal{Q}_{127}$. Then the Number-Theoretic Transform is the map NTT:$\mathcal{R}_q \to T_q$ given by

$$\hat{\mathbf{a}} = NTT(\mathbf{a}) = (\mathbf{a} \bmod (X^2 - \zeta_0), \mathbf{a} \bmod (X^2 - \zeta_1), \ldots . \mathbf{a} \bmod (X^2 - \zeta_{127})) \tag{3.2}$$

One can check that NTT is a ring isomorphism and hence its inverse $\text{NTT}^{-1}$ exists.

## 3.1 Multiplication in $\mathcal{R}_q$

Let $\mathbf{a}(X), \mathbf{b}(X) \in \mathcal{R}_q$. Let $\mathbf{c}(X) = \mathbf{a}(X).\mathbf{b}(X) \bmod (X^n + 1)$. Then

$$\mathbf{c}(X) = \mathbf{a}(X).\mathbf{b}(X) + \mathbf{p}(X)(X^n + 1).$$

Hence

$$\mathbf{c}(X) \bmod (X^2 - \zeta_i) = \mathbf{a}(X).\mathbf{b}(X) \bmod (X^2 - \zeta_i),$$

since $X^n + 1 \bmod (X^2 - \zeta_i) = \zeta^{n/2^{(2br(i)+1)}} + 1 = (-1)^{2br(i)+1} + 1 = 0$. Thus

$$\hat{\mathbf{c}} = \hat{\mathbf{a}} \odot \hat{\mathbf{b}},$$

where $\odot$ is component-wise multiplication in $T_q$. Consequently

$$\mathbf{c} = NTT^{-1}(\hat{\mathbf{a}} \odot \hat{\mathbf{b}}).$$

## 3.2 Multiplication in $\mathcal{Q}_i$

. Let $a_0 + a_1X, b_0 + b_1X \in \mathcal{Q}_i$. Then

$$(a_0+b_1X)(b_0+b_1X) \bmod (X^2-\zeta_i) = a_0b_0+(a_0b_1+a_1b_0)X+a_1b_1 \bmod (X^2-\zeta_i)$$

$$= (a_0b_0 + a_1b_1\zeta_i) + (a_0b_1 + a_1b_0)X.$$

## 3.3 Computing Kyber NTTs

Recall that $q = 3328$ and $q - 1 = 2^8.13$ Hence a primitive $256th$ root of unity exists but $512th$ root does not exist. Fix $\zeta = 17$ a primitive $256th$ root of unity Let $\mathbf{f}(X) = \sum_{i=0}^{255} f_iX^i$ be an element of $\mathcal{R}_q$. We identify $\mathbf{f}$ with the vector of coefficients $(f_0, \dots, f_{255}) \in \mathbb{Z}_q^{256}$. Define $\mathbf{f}^0 = (f_0, f_2, \dots, f_{254})$ and $\mathbf{f}^1 = (f_1, f_3, \dots, f_{255})$. Then

$$\mathbf{f}(X) = \mathbf{f}^0(X^2) + X\mathbf{f}^1(X^2).$$

Consequently

$$\mathbf{f} \bmod (X^2 - \zeta_i) = \mathbf{f}^0(\zeta_i) + \mathbf{f}^1(\zeta_i)X.$$

Now define

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j}\zeta_i^j \tag{3.3}$$

$$\hat{f}_{2i+1} \sum_{j=0}^{127} f_{2j+1}\zeta_i^j \tag{3.4}$$

Then from (3.2) we have

$$\hat{\mathbf{f}} = (\hat{f}_0 + \hat{f}_1 X, \hat{f}_2 + \hat{f}_3 X, \ldots, \hat{f}_{254} + \hat{f}_{255} X). \tag{3.5}$$

Let $\mathbf{A}$ be the following $128 \times 128$ matrix over $\mathbb{Z}_q$.

$$\mathbf{A} = \begin{pmatrix} 1 & \zeta_0 & \zeta_0^2 & \ldots & \zeta_0^{127} \\ 1 & \zeta_1 & \zeta_1^2 & \ldots & \zeta_1^{127} \\ 1 & \zeta_2 & \zeta_2^2 & \ldots & \zeta_2^{127} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & \zeta_{127} & \zeta_{127}^2 & \ldots & \zeta_{127}^{127} \end{pmatrix}$$

Then (3.3) and (3.4) can be re-written as

$$(\hat{\mathbf{f}}^0)^T = \mathbf{A}(\mathbf{f}^0)^T, \tag{3.6}$$

$$(\hat{\mathbf{f}}^1)^T = \mathbf{A}(\mathbf{f}^1)^T, \tag{3.7}$$

Hence

$$(\mathbf{f}^0)^T = \mathbf{A}^{-1}(\hat{\mathbf{f}}^0)^T, \tag{3.8}$$

$$(\mathbf{f}^1)^T = \mathbf{A}^{-1}(\hat{\mathbf{f}}^1)^T, \tag{3.9}$$

We now show that

$$\mathbf{A}^{-1} = 1/128 \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ \zeta_0^{-1} & \zeta_1^{-1} & \zeta_2^{-1} & \ldots & \zeta_{127}^{-1} \\ \zeta_0^{-2} & \zeta_1^{-2} & \zeta_2^{-2} & \ldots & \zeta_{127}^{-2} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \zeta_0^{-127} & \zeta_1^{-127} & \zeta_2^{-127} & \ldots & \zeta_{127}^{-127} \end{pmatrix}$$

Denote the matrix on RHS by $\mathbf{C}$. Then the (i,j)$th$ entry of $\mathbf{A} \times \mathbf{C}$ is

$$1/128 \sum_{k=0}^{127} \zeta_i^k . \zeta_j^{-k} = 1/128 \sum_{k=0}^{127} \zeta^{2(br(i)-br(j))k}.$$

When $i = j$ this sum is 1. When $i \neq j$ the sum is

$$\frac{1}{128} \frac{\zeta^{2(br(i)-br(j))128} - 1}{\zeta^{2(br(i)-br(j))} - 1} = 1/128 \frac{1-1}{\zeta^{2(br(i)-br(j))} - 1} = 0,$$

since $\zeta$ is a primitive $256th$ root of unity. Thus $\mathbf{C}$ is the inverse of $\mathbf{A}$. Thus (3.8) and (3.9) yield

$$f_{2i} = 1/128 \sum_{j=0}^{127} \hat{f}_{2j} \zeta_j^{-i}, \tag{3.10}$$

$$f_{2i+1} = 1/128 \sum_{j=0}^{127} \hat{f}_{2j+1} \zeta_j^{-i}. \tag{3.11}$$

## 3.4 Faster NTT(Cooley-Tukey)

Let $n = 2^7 = 128$ and $q = 3329$. Recall that $\zeta$ is a primitive $2nth$ root of unity in $\mathbb{Z}_q$ and $\zeta^n = -1$. Let $\zeta' = \zeta^2$. Then $\zeta'$ is a primitive $nth$ root of unity. Fro m (3.3) we have

$$\hat{f}_{2i} = \sum_{j=0}^{n-1} f_{2j} \zeta_i^j = \sum_{j=0}^{n-1} f_j^0 \zeta_i^j$$

$$= \sum_{j=0}^{n/2-1} f_{2j}^0 \zeta_i^{2j} + \sum_{j=0}^{n/2-1} f_{2j+1}^0 \zeta_i^{2j+1}$$

Thus

$$\hat{f}_{2i} = \sum_{j=0}^{n/2-1} f_{2j}^0 \zeta_i'^j + \zeta_i \sum_{j=0}^{n/2-1} f_{2j+1}^0 \zeta_i'^j, \ 0 \le i < n/2. \tag{3.12}$$

Replacing $i$ by $n/2 + i$, we have

$$\hat{f}_{n+2i} = \sum_{j=0}^{n/2-1} f_{2j}^0 \zeta_{n/2+i}^{2j} + \sum_{j=0}^{n/2-1} f_{2j+1}^0 \zeta_{n/2+i}^{2j+1}, \ 0 \le i < n/2.$$

Now, observe that
$\zeta_{n/2+i}^{2j} = \zeta^{(2br(n/2+i)+1)(2j)} = \zeta^{(n+2br(i)+1)(2j)} = \zeta^{(2br(i)+1)(2j)} = \zeta_i^{2j}$,
since $\zeta^{n.2j} = 1$. Also, since $\zeta^{n.(2j+1)} = -1$, we have $\zeta_{n/2+i}^{2j+1} = -\zeta_i^{2j+1}$. Hence, it follows that

$$\hat{f}_{n+2i} = \sum_{j=0}^{n/2-1} f_{2j}^0 \zeta_i^{2j} - \sum_{j=0}^{n/2-1} f_{2j+1}^0 \zeta_i^{2j+1}, \ 0 \le i < n/2,$$

which we write as

$$\hat{f}_{n+2i} = \sum_{j=0}^{n/2-1} f^0_{2j}\zeta_i'^j - \zeta_i \sum_{j=0}^{n/2-1} f^0_{2j+1}\zeta_i'^j,\ 0 \le i < n/2, \tag{3.13}$$

Equations (3.12) and (3.13) yield two sub-problems over a smaller ring $\mathbb{Z}[X]/(x^{n/2}+1)$. This will give rise to a recursive algorithm.. Similar expresions can be obtained for $\hat{f}_{2i+1}$.

## 3.5 Parameter sets for Kyber

Kyber is parameterized by integers $n, k, q, \eta_1, \eta_2, d_u$ and $d_v$ as given below.

| | n | k | q | $\eta_1$ | $\eta_2$ | $(d_u, d_v)$ | $\delta$ |
|---|---|---|---|---|---|---|---|
| Kyber 512 | 256 | 2 | 3329 | 3 | 2 | (10,4) | $2^{-139}$ |
| Kyber768 | 256 | 3 | 3329 | 2 | 2 | (10,4) | $2^{-164}$ |
| Kyber1024 | 256 | 4 | 3329 | 2 | 2 | (11,5) | $2^{-174}$ |

Here $\delta$ denotes the failure probability.

## 3.6 Instantiation of PRF, XOF, H,G and KDF

Tese primitives are instatiated with functions from the FIPS-202 standard as follows:

- Instantiate **XOF** with **SHAKE**-128;
- instantiate **H** with **SHA**3-256;
- instantiate **G** with **SHA**3-512;
- instantiate **PRF**(s,b) with **SHAKE**-256($s||b$); and
- instantiate **KDF** with **SHAKE**-256

# 4 Kyber CPA-PKE

Kyber CPA-PKE is parametrized by $n, k, q, \eta_1, \eta_2, d_u$ and $d_v$. As stated above $n$ is always 256 and $q$ is alwya 3329.

==========================================

**Kyber.CPAPKE.KeyGen()**

==========================================

**Output**: Secret key $sk \in \mathcal{B}^{12.k.n/8}$; Public key $pk \in \mathcal{B}^{12.k.n/8+32}$

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathbf{G}(d)$
$N := 0$
**for** $i = 0$ **to** $k-1$ **do**
  **for** $j = 0$ **to** $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathbf{Parse}(\mathbf{XOF}(\rho, j, i)$ ◊ generate matrix $\hat{\mathbf{A}} \in \mathcal{R}_q^{k\times k}$ in NTT domain
  **end for**
**end for**
**for** $i = 0$ **to** $k-1$ **do**
  $s[i] := CBD_{\eta_1}(\mathbf{PRF}(\sigma, N))$ ◊ sample $\mathbf{s} \in \mathcal{R}_q^k$ from $B_{\eta_1}$
  $N \leftarrow N + 1$
**end for**
**for** $i = 0$ **to** $k-1$ **do**
  $e[i] := CBD_{\eta_1}(\mathbf{PRF}(\sigma, N))$ ◊ sample $\mathbf{e} \in \mathcal{R}_q^k$ from $B_{\eta_1}$
  $N \leftarrow N + 1$
**end for**
$\hat{\mathbf{s}} := NTT(\mathbf{s})$
$\hat{\mathbf{e}} := NTT(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \odot \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$pk := \mathbf{Encode}_{12}(\hat{\mathbf{t}} \bmod q) || \rho$ ◊ $pk := \mathbf{As} + \mathbf{e}$
$sk := \mathbf{Encode}_{12}(\hat{\mathbf{s}} \bmod q)$ ◊ $sk := \mathbf{s}$
$\mathbf{Return}(pk, sk)$.

==========================================

**Kyber.CPAPKE.Enc**$(pk, m, r)$

==========================================

**INPUT:** Publik Key $pk \in \mathcal{B}^{12.k.n/8+32}$, message $m \in \mathcal{B}^{32}$;
random coins $r \in \mathcal{B}^{32}$

**OUTPUT**: ciphertext $c \in \mathcal{B}^{d_u.k.n/8+d_v.n/8}$

1. $N \leftarrow 0$
2. $\hat{\mathbf{t}} := \mathbf{Decode}_{12}(pk)$
3. $\rho := pk + 12.k.n/8$ $\Diamond$ extract the seed $\rho$ from $pk$
4. **for**$i = 0$ **to** $k - 1$ **do**
5. **for** $j = 0$ **to** $k - 1$ **do**
6. $\mathbf{A}^T[i][j] := \mathbf{PARSE}(\mathbf{XOF}(\rho, i, j))$ $\Diamond$ genetrate the matrix
. $\mathbf{A} \in \mathcal{R}_q^{k \times k}$ in NTT domain
7. **end for**
8. **end for**
9. **for** $i = 0$ **to** $k - 1$ **do**
10. $\mathbf{r}[i] := CBD_{\eta_1}(\mathbf{PRF}(r, N))$ $\Diamond$ sample $\mathbf{r} \in \mathcal{R}_q^k$ according to $B_{\eta_1}$
11. $N \leftarrow N + 1$
12 **end for**
13. **for** $i = 0$ **to** $k - 1$ **do**
14. $\mathbf{e}_1[i] := CBD_{\eta_2}(\mathbf{PRF}(r, N))$ $\Diamond$ sample $\mathbf{e}_1 \in \mathcal{R}_q^k$ according to $B_{\eta_2}$
15. $N \leftarrow N + 1$
16. **end for**
17. $e_2 := CBD_{\eta_2}(\mathbf{PRF}(r, N))$ $\Diamond$ sample $e_2 \in \mathcal{R}_q$ according to $B_{\eta_2}$
18. $\hat{\mathbf{r}} := NTT(\mathbf{r})$
19. $\mathbf{u} := NTT^{-1}(\hat{\mathbf{A}}^T \odot \hat{\mathbf{r}}) + \mathbf{e}_1$ $\Diamond \mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
20. $v := NTT^{-1}(\hat{\mathbf{t}}^T \odot \hat{\mathbf{r}}) + e_2 + \mathbf{Decompress}_q(\mathbf{Decode}_1(m), 1)$
. $\Diamond v := \mathbf{t}^T\mathbf{r} + e_2 + \mathbf{Decompress}_q(m, 1)$
21. $c_1 := \mathbf{Encode}_{d_u}(\mathbf{Compress}_q(\mathbf{u}, d_u))$
22. $c_2 := \mathbf{Encode}_{d_v}(\mathbf{Compress}_q(v, d_v))$
23. **return** $c := c_1||c_2$ $\Diamond c = (\mathbf{Compress}_q(\mathbf{u}, d_u), \mathbf{Compress}_q(v, d_v))$

**Remark:** Note that in Line 20 of the encryption algorithm, for each bit $b$ of the message $m$, the decompression function adds. $b.\lceil q/2 \rfloor$ .

The decryption algorithm is given below.

==================================================

**Kyber.CPAPKE.Dec**$(sk, c)$

==================================================

**INPUT**: secret key $sk \in \mathcal{B}^{12.k.n/8}$, ciphertext $c \in \mathcal{B}^{d_u.k.n/8+d_v.n/8}$
**OUTPUT**: message $m \in \mathcal{B}^{32}$

1. $\mathbf{u} := \mathbf{Deccompress}_q(\mathbf{Decode}_{d_u}(c), d_u)$
2. $v := \mathbf{Deccompress}_q(\mathbf{Decode}_{d_v}(c + d_u.k.n/8), d_v)$
3. $\hat{\mathbf{s}} := \mathbf{Decode}_{12}(sk)$
4. $m := \mathbf{Encode}_1(\mathbf{Compress}_q(v - NTT^{-1}(\hat{\mathbf{s}}^T \odot NTT(\mathbf{u})), 1))$

. $\Diamond m := \mathbf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

5. **return** $m$

==================================================

**Correctness**: In line 4 of the decryption algorithm, the compression function decrypts to a 1 if $v - \mathbf{s}^T\mathbf{u}$ is closer to $\lceil q/2 \rfloor$ than to 0, and decrypts to 0 otherwise. Now, let us compute $v - \mathbf{s}^T\mathbf{u}$. By line 19 of the encryption algorithm

$$\mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e}_1,$$

and by line 20 we have

$$v := \mathbf{t}^T\mathbf{r} + e_2 + \lceil q/2 \rfloor m.$$

Also we have $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$. Hence

$$v - \mathbf{s}^T\mathbf{u} = \mathbf{t}^T\mathbf{r} + e_2 + \lceil q/2 \rfloor m - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}_1$$

$$= (\mathbf{s}^T\mathbf{A}^T + \mathbf{e}^T)\mathbf{r} + e_2 + \lceil q/2 \rfloor - \mathbf{s}^T\mathbf{A}^T\mathbf{r} - \mathbf{s}^T\mathbf{e}^1$$

$$= \mathbf{e}^T\mathbf{r} + e_2 - \mathbf{s}^T\mathbf{e}_1 + \lceil q/2 \rfloor m.$$

Now, iif $||\mathbf{e}^T\mathbf{r} + e_2 - \mathbf{s}^T\mathbf{e}_1||_\infty < q/4$, then we can write

$$v - \mathbf{s}^T\mathbf{u} = w + \lceil q/2 \rfloor m,$$

where, $||w||_\infty < q/4$. Let $m' = \mathbf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$. Then we know that

$$q/4 \geq ||v - \mathbf{s}^T\mathbf{u} - \lceil q/2 \rfloor m'||_\infty$$

$$= ||w + \lceil q/2 \rfloor (m - m')||_\infty.$$

Hence

$$\lceil q/2 \rfloor ||(m - m'||_\infty = ||w + \lceil q/2 \rfloor (m - m') - w||_\infty \leq$$
$$||w + \lceil q/2 \rfloor (m - m')||_\infty + ||w||_\infty < 2(q/4) = q/2.$$

For odd $q$, this is possible only when $m = m'$. □

**Remark:** One can show that $||\mathbf{e}^T \mathbf{r} + e_2 - \mathbf{s}^T \mathbf{e}_1||_\infty < q/4$ with overwhelming probability. Hence, decryption will almost certainly yield the correct message.

## 4.1 Security

: By M-LWE, adversary $\mathcal{A}$ can not distinguish $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$ from random. Again by M-LWE, $\mathcal{A}$ can not distinguish $\mathbf{t}^T \mathbf{r} + e_2$ from random. Thus to an adversary, $v$ appears to be a sum of a random element in $\mathcal{R}_q$ and $\lceil q/2 \rfloor m$. Thus adversary $\mathcal{A}$ can learn nothing about the message $m$. □

# 5 Kyber CCAKEM

One constructs IND-CCA2- secure Kyber CCAKEM, from the IND-CPA - secure public- key encryption scheme Kyber CPAPKE via a tweaked Fujisali-Okamoto transform. Key generation, encapsulation, and decapsulation of Kyber.CCAKEM are described below.

==================================================

Kyber.CCAKEM.KeyGen()

==================================================

**Output**: Public key $pk \in \mathcal{B}^{12.k.n/8+32}$; secret key $sk \in \mathcal{B}^{24.k.n/8+96}$

1. $z \leftarrow \mathcal{B}^{32}$
2. $(pk, sk') \leftarrow Kyber.CPAPKE.KeyGen()$
3. $sk := (sk'||pk||\mathbf{H}(pk)||z)$
6. **return** $(pk, sk)$

==================================================

==================================================

Kyber.CCAKEM.Enc($pk$)

==================================================

**INPUT**: Public key $pk \in \mathcal{B}^{12.k.n/8+32}$

**OUTPUT:** Ciphertext $c \in \mathcal{B}^{d_u.k.n/8+d_v.n/8}$; shared key $K \in \mathcal{B}^*$

1. $m \leftarrow \mathcal{B}^{32}$
2. $m \leftarrow \mathbf{H}(m)$
3. $(\bar{K}, r) := \mathbf{G}(m||\mathbf{H}(pk))$
4. $c := Enc.CPAPKE.Enc(pk, m, r)$
5. $K := \mathbf{KDF}(\bar{K}||\mathbf{H}(c))$
6. **return**$(c, K)$.

==================================================

==================================================

Kyber.CCAKEM.Dec($c, sk$)

==================================================

**INPUT**: Ciphertext $c \in \mathcal{B}^{d_u.k.n/8+d_v.n/8}$; secret key $sk \in \mathcal{B}^{24.k.n/8+96}$

**OUTPUT**: Shared key $K \in \mathcal{B}^*$

1. $pk := sk + 12.k.n/8$
2. $h := sk + 24.k.n/8 + 32 \in \mathcal{B}^{32}$
3. $z := sk + 24.k.n/8 + 64$
4. $m' := Kyber.\text{CPAPKE.Dec(sk, c)}$
5. $(\bar{K}', r') := \mathbf{G}(m'||h)$
6. $c' := Kyber.\text{CPAPKE.Enc(pk, m', r')}$
7. **If** $c = c'$ **then**
8.     **return** $K := \mathbf{KDF}(\bar{K}'||\mathbf{H}(c))$
9. **else**
10.     **return** $K := \mathbf{KDF}(z||\mathbf{H}(c))$
11. **end if**
12. **return** $K$

# References


[CD] L.Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler and D. Stehlé. *CRYSTALS- Kyber: a CCA-secure module-lattice-based KEM*,
https://eprint.iacr.org/2017/634.pdf

[CK3.02] R. Avanzi, J. Bos, L.Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler and D. Stehlé. *CRYSTALS-Kyber(version 3.02)*
https://pq-crystals.org/kyber/

[AM] A. Menezes: Cryptography101 with Alfred Menezes
https://cryptography101.ca/kyber-dilithium/