Estudo e Análise: Kyber

Gabriel Costa Kinder - 234720

O que é o Kyber?

O que é o Kyber?

- Algoritmo de Criptografia pública pós-quântico

O que é o Kyber?

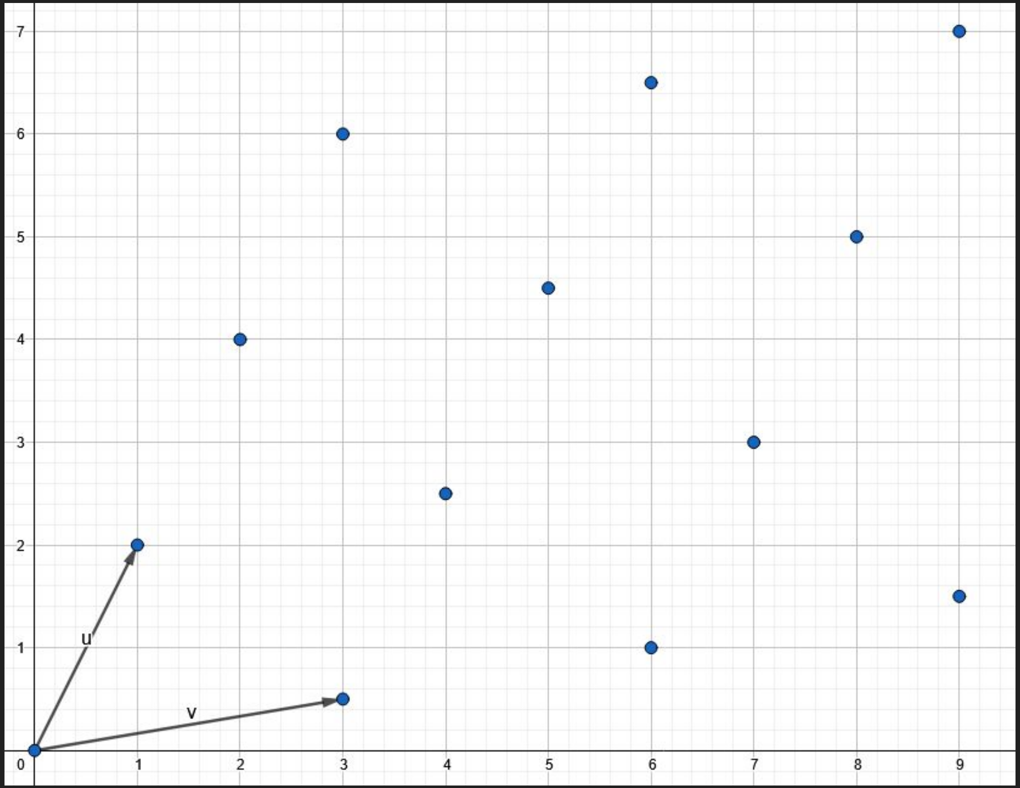
- Algoritmo de Criptografia pública pós-quântico
- Padronizado (05/07/2022)

O que é o Kyber?

- Algoritmo de Criptografia pública pós-quântico
- Padronizado (05/07/2022)
- Baseado em problemas de Reticulados

O que é um reticulado?

O que é um reticulado?



O algoritmo simplificado: Geração de Chaves

O algoritmo simplificado: Geração de Chaves

$$A*s + e = t$$

O algoritmo simplificado: Geração de Chaves

$$A * s + e = t$$

private key

public key

O algoritmo simplificado: A mensagem

O algoritmo simplificado: A mensagem

C

O algoritmo simplificado: A mensagem

C

01000011

O algoritmo simplificado: A mensagem

C

01000011

$$x^6 + x + 1$$

O algoritmo simplificado: A mensagem

C

01000011

$x^6 + x + 1$

$m = 1665x^6 + 1665x + 1665$

O algoritmo simplificado: Encriptação

O algoritmo simplificado: Encriptação

$$u = A * r + e_1$$

O algoritmo simplificado: Encriptação

$$u = A * r + e_1$$

$$v = t * r + e_2 + m$$

O algoritmo simplificado: Encriptação

$$u = A * r + e_1$$

$$v = t * r + e_2 + m$$

$$c = (u, v)$$

O algoritmo simplificado: Decriptação

O algoritmo simplificado: Decriptação

$d = v - s * u$

O algoritmo simplificado: Decriptação

$$d = v - s * u = t * r + e_2 + m - s(A * r + e_1)$$

O algoritmo simplificado: Decriptação

$$d = v - s * u = t * r + e_2 + m - s(A * r + e_1)$$

$$d = r(t - A * s) + e_2 + m - s * e_1$$

O algoritmo simplificado: Decriptação

$$d = v - s * u = t * r + e_2 + m - s(A * r + e_1)$$

$$d = r(t - A * s) + e_2 + m - s * e_1$$

$$d = r * e + e_2 + m - s * e_1$$

O algoritmo simplificado: Decriptação

$$d = v - s * u = t * r + e_2 + m - s(A * r + e_1)$$

$$d = r(t - A * s) + e_2 + m - s * e_1$$

$$d = r * e + e_2 + m - s * e_1$$

$$d = 35x^7 + 1458x^6 + 87x^5 + 24x^4 + 46x^3 + 110x^2 + 1892x + 1555$$

O algoritmo simplificado: Decriptação

$$d = v - s * u = t * r + e_2 + m - s(A * r + e_1)$$

$$d = r(t - A * s) + e_2 + m - s * e_1$$

$$d = r * e + e_2 + m - s * e_1$$

$$d = 35x^7 + 1458x^6 + 87x^5 + 24x^4 + 46x^3 + 110x^2 + 1892x + 1555$$

$$d = 1665x^6 + 1665x + 1665 = m$$

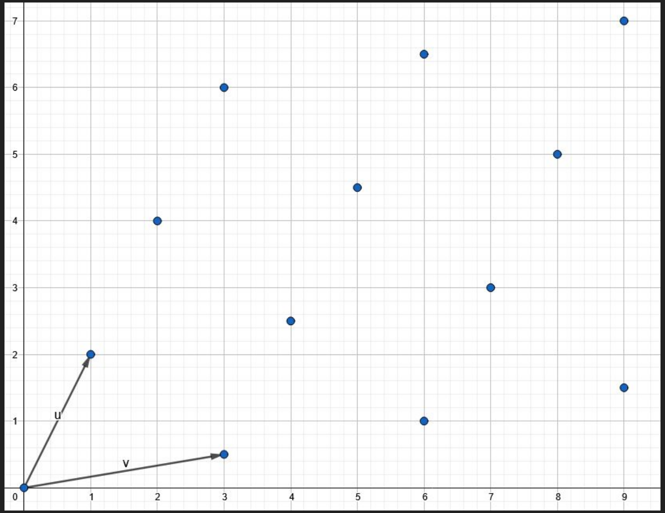
Reticulados?

Reticulados?

$$A*s + e = t$$

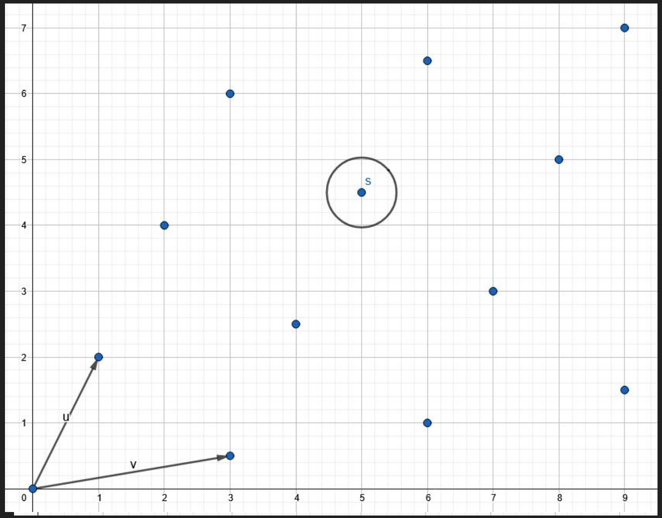
Reticulados?

$$A * s + e = t$$



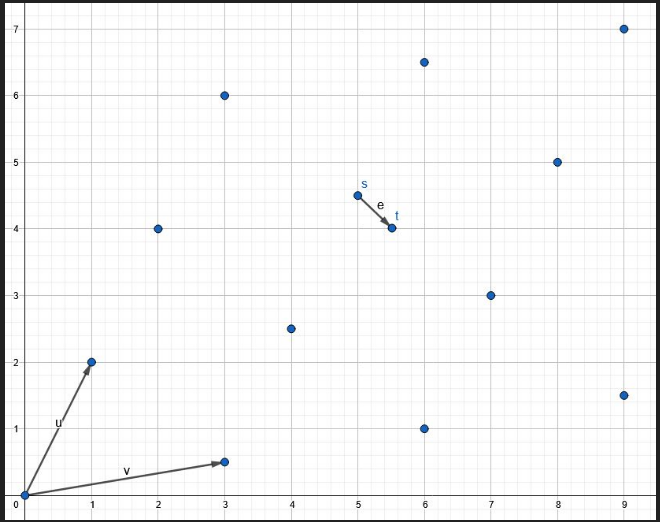
Reticulados?

$$A * s + e = t$$



Reticulados?

$$A * s + e = t$$



Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Transformada Teórica Numérica

Transformada Teórica Numérica

- Converte polinômio de grau 255 para 128 polinômios de grau 1

Transformada Teórica Numérica

- Converte polinômio de grau 255 para 128 polinômios de grau 1
- Auxilia nas multiplicações em $Z_q[X]/(X^n + 1)$

Transformada Teórica Numérica

- Converte polinômio de grau 255 para 128 polinômios de grau 1
- Auxilia nas multiplicações em $Z_q\,[X]/(X^n + 1)$
  - Condição: $q - 1 = c * n$

Transformada Teórica Numérica

- Converte polinômio de grau 255 para 128 polinômios de grau 1
- Auxilia nas multiplicações em $Z_q [X]/(X^n +1)$
  - Condição: q - 1 = c * n
  - n = 256 -> q = {257, 769, 3329, …}

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Geração de “noise”

- Distribuição Binomial Centrada

Geração de “noise”

- Distribuição Binomial Centrada
- “Seed” gerada por uma função pseudo-aleatória

Geração de “noise”

- Distribuição Binomial Centrada
- “Seed” gerada por uma função pseudo-aleatória
- Gera: s, e, r, e_1, e_2

Geração de “noise”

- Distribuição Binomial Centrada
- “Seed” gerada por uma função pseudo-aleatória
- Gera: s, e, r, e_1, e_2
- **Input:** Byte array $B = (b_0, b_1, \ldots, b_{64\eta-1}) \in \mathcal{B}^{64\eta}$
  Output: Polynomial $f \in R_q$
  $(\beta_0, \ldots, \beta_{512\eta-1}) := \mathsf{BytesToBits}(B)$
  for i from 0 to 255 **do**
  $a := \sum_{j=0}^{\eta-1} \beta_{2i\eta+j}$
  $b := \sum_{j=0}^{\eta-1} \beta_{2i\eta+\eta+j}$
  $f_i := a - b$
  end for
  return $f_0 + f_1X + f_2X^2 + \cdots + f_{255}X^{255}$

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Compressão e Descompressão

- Remover bits de baixa ordem do texto cifrado

Compressão e Descompressão

- Remover bits de baixa ordem do texto cifrado
- Converter a mensagem

Compressão e Descompressão

- Remover bits de baixa ordem do texto cifrado
- Converter a mensagem
- $$\mathsf{Compress}_q(x, d) = \lceil (2^d/q) \cdot x \rfloor \bmod^+ 2^d\,,$$
  $$\mathsf{Decompress}_q(x, d) = \lceil (q/2^d) \cdot x \rfloor\,.$$

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Classificação de Segurança

Classificação de Segurança

- Algoritmo Inicial: IND-CPA, 32 bytes

Classificação de Segurança

- Algoritmo Inicial: IND-CPA, 32 bytes
- Classificação desejada: IND-CCA2

Classificação de Segurança

- Algoritmo Inicial: IND-CPA, 32 bytes
- Classificação desejada: IND-CCA2
- Transformada Fujisaki-Okamoto

Classificação de Segurança

- Algoritmo Inicial: IND-CPA, 32 bytes
- Classificação desejada: IND-CCA2
- Transformada Fujisaki-Okamoto
- Chance de falha de decriptação

Parâmetros

	n	k	q	η_1	η_2	d_u	d_v	δ
Kyber512	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}

Geração de A

Geração de A

- Distribuição Uniforme

Geração de A

- Distribuição Uniforme
- Direto no domínio NTT

Geração de A

- Distribuição Uniforme
- Direto no domínio NTT
- **Input:** Byte stream $B = b_0, b_1, b_2 \cdots \in \mathcal{B}^*$
  Output: NTT-representation $\hat{a} \in R_q$ of $a \in R_q$
  $i := 0$
  $j := 0$
  while $j < n$ **do**
  $\quad d_1 := b_i + 256 \cdot (b_{i+1} \bmod^+ 16)$
  $\quad d_2 := \lfloor b_{i+1}/16 \rfloor + 16 \cdot b_{i+2}$
  $\quad$ **if** $d_1 < q$ **then**
  $\quad\quad \hat{a}_j := d_1$
  $\quad\quad j := j + 1$
  $\quad$ **end if**
  $\quad$ **if** $d_2 < q$ **and** $j < n$ **then**
  $\quad\quad \hat{a}_j := d_2$
  $\quad\quad j := j + 1$
  $\quad$ **end if**
  $\quad i := i + 3$
  end while
  return $\hat{a}_0 + \hat{a}_1 X + \cdots + \hat{a}_{n-1} X^{n-1}$

Funções Simétricas Auxiliares

Funções Simétricas Auxiliares

Kyber	Kyber 90s

Funções Simétricas Auxiliares

Kyber

- XOF: SHAKE-128

Kyber 90s

- XOF: AES-256, CTR mode

Funções Simétricas Auxiliares

Kyber

- XOF: SHAKE-128
- H: SHA3-256

Kyber 90s

- XOF: AES-256, CTR mode
- H: SHA2-256

Funções Simétricas Auxiliares

Kyber

- XOF: SHAKE-128
- H: SHA3-256
- G: SHA3-512

Kyber 90s

- XOF: AES-256, CTR mode
- H: SHA2-256
- G: SHA2-512

Funções Simétricas Auxiliares

Kyber

- XOF: SHAKE-128
- H: SHA3-256
- G: SHA3-512
- PRF: SHAKE-256

Kyber 90s

- XOF: AES-256, CTR mode
- H: SHA2-256
- G: SHA2-512
- PRF: AES-256, CTR mode

Funções Simétricas Auxiliares

Kyber

- XOF: SHAKE-128
- H: SHA3-256
- G: SHA3-512
- PRF: SHAKE-256
- KDF: SHAKE-256

Kyber 90s

- XOF: AES-256, CTR mode
- H: SHA2-256
- G: SHA2-512
- PRF: AES-256, CTR mode
- KDF: SHAKE-256

O algoritmo completo: Kyber.CPAPKE

O algoritmo completo: Kyber.CPAPKE

Geração de Chaves

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

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathrm{G}(d)$
$N := 0$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, j, i))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{s} \in R_q^k$ from B_{η_1}
  $\mathbf{s}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e} \in R_q^k$ from B_{η_1}
  $\mathbf{e}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
  $N := N + 1$
end for
$\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$pk := (\mathsf{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q) \| \rho)$ ▷ $pk := \mathbf{As} + \mathbf{e}$
$sk := \mathsf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$ ▷ $sk := \mathbf{s}$
return (pk, sk)

O algoritmo completo: Kyber.CPAPKE

Encriptação

Input: Public key $pk \in \mathcal{B}^{12\cdot k\cdot n/8+32}$
Input: Message $m \in \mathcal{B}^{32}$
Input: Random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u\cdot k\cdot n/8+d_v\cdot n/8}$

$N := 0$
$\hat{\mathbf{t}} := \mathsf{Decode}_{12}(pk)$
$\rho := pk + 12\cdot k\cdot n/8$
for i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k\times k}$ in NTT domain
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, i, j))$
  end for
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{r} \in R_q^k$ from B_{η_1}
  $\mathbf{r}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(r, N))$
  $N := N+1$
end for
for i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e}_1 \in R_q^k$ from B_{η_2}
  $\mathbf{e}_1[i] := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$
  $N := N+1$
end for
$e_2 := \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(r, N))$ ▷ Sample $e_2 \in R_q$ from B_{η_2}
$\hat{\mathbf{r}} := \mathsf{NTT}(\mathbf{r})$
$\mathbf{u} := \mathsf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ ▷ $\mathbf{u} := \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
$v := \mathsf{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + \mathsf{Decompress}_q(\mathsf{Decode}_1(m), 1)$ ▷ $v := \mathbf{t}^T\mathbf{r} + e_2 + \mathsf{Decompress}_q(m, 1)$
$c_1 := \mathsf{Encode}_{d_u}(\mathsf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathsf{Encode}_{d_v}(\mathsf{Compress}_q(v, d_v))$
return $c = (c_1 \| c_2)$ ▷ $c := (\mathsf{Compress}_q(\mathbf{u}, d_u), \mathsf{Compress}_q(v, d_v))$

O algoritmo completo: Kyber.CPAPKE

Decriptação

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

1: $\mathbf{u} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_u}(c), d_u)$
2: $v := \mathsf{Decompress}_q(\mathsf{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \mathsf{Decode}_{12}(sk)$
4: $m := \mathsf{Encode}_1(\mathsf{Compress}_q(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u})), 1))$ $\quad \triangleright\ m := \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1))$
5: **return** m

O algoritmo completo: Kyber.CCAKEM

O algoritmo completo: Kyber.CCAKEM

Geração de Chave

Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$

```
z ← B^32
(pk, sk′) := KYBER.CPAPKE.KeyGen()
sk := (sk′‖pk‖H(pk)‖z)
return (pk, sk)
```

Encriptação

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Output: Shared key $K \in \mathcal{B}^*$

```
m ← B^32
m ← H(m)
(K̄, r) := G(m‖H(pk))
c := KYBER.CPAPKE.Enc(pk, m, r)
K := KDF(K̄‖H(c))
return (c, K)
```

Decriptação

Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8+d_v \cdot n/8}$
Input: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$
Output: Shared key $K \in \mathcal{B}^*$

```
pk := sk + 12 · k · n/8
h := sk + 24 · k · n/8 + 32 ∈ B^32
z := sk + 24 · k · n/8 + 64
m′ := KYBER.CPAPKE.Dec(s, (u, v))
(K̄′, r′) := G(m′‖h)
c′ := KYBER.CPAPKE.Enc(pk, m′, r′)
if c = c′ then
    return K := KDF(K̄′‖H(c))
else
    return K := KDF(z‖H(c))
end if
return K
```

Performance

Performance - Espaço (bytes)

	Kyber512	Kyber512 90s	Kyber768	Kyber768 90s	Kyber1024	Kyber1024 90s
sk	1632	1632	2400	2400	3168	3168
pk	800	800	1184	1184	1568	1568
ct	768	768	1088	1088	1568	1568

Performance - Espaço (bytes)

	Kyber512	Kyber512 90s	Kyber768	Kyber768 90s	Kyber1024	Kyber1024 90s
sk	1632	1632	2400	2400	3168	3168
pk	800	800	1184	1184	1568	1568
ct	768	768	1088	1088	1568	1568

Segurança semelhante	RSA (pk)	ECC (pk)
Kyber512	384	32
Kyber768	960	48
Kyber1024	1920	64

Performance - Ciclos, referência

Intel Core-i7 4770K (Haswell)

	Kyber512	Kyber512 90s	Kyber768	Kyber768 90s	Kyber1024	Kyber1024 90s
gen	122684	213156	199408	389760	307148	636380
enc	154524	213156	235260	432764	346648	672644
dec	187960	277612	274900	473984	396584	724144

Performance - Ciclos, avx2

Intel Core-i7 4770K (Haswell)

	Kyber512	Kyber512 90s	Kyber768	Kyber768 90s	Kyber1024	Kyber1024 90s
gen	33856	21880	52732	30460	73544	43212
enc	45200	28592	67624	40140	97324	56556
dec	34572	20980	53156	30108	79128	44328

Referências

Official Kyber Website: https://pq-crystals.org/kyber/index.shtml

Kyber Specification Documentation: https://pq-crystals.org/kyber/data/kyber-specification-round3.pdf

rC3 Conference Talk About Kyber: https://media.ccc.de/v/rc3-2021-cwtv-230-kyber-and-post-quantum