CRYSTALS–Kyber

A CCA-secure module-lattice-based KEM

Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, **John M. Schanck**, Peter Schwabe, Gregor Seiler, Damien Stehlé

authors@pq-crystals.org

https://pq-crystals.org/kyber

April 26, 2018

Overview

CRYSTALS is a suite of public key cryptographic algorithms.

Kyber is a general purpose key encapsulation mechanism (KEM).

Overview

CRYSTALS is a suite of public key cryptographic algorithms.

Kyber is a general purpose key encapsulation mechanism (KEM).

Overview

CRYSTALS is a suite of public key cryptographic algorithms.

Kyber is a general purpose key encapsulation mechanism (KEM).

- Think of it as a successor to the NewHope key exchange from 2016.

Overview

CRYSTALS is a suite of public key cryptographic algorithms.

Kyber is a general purpose key encapsulation mechanism (KEM).

- Think of it as a successor to the NewHope key exchange from 2016.
    - Comparable computational efficiency (can be used in an ephemeral setting).
    - Smaller public keys and ciphertexts.
    - CCA secure (keys can be reused).

Overview

CRYSTALS is a suite of public key cryptographic algorithms.

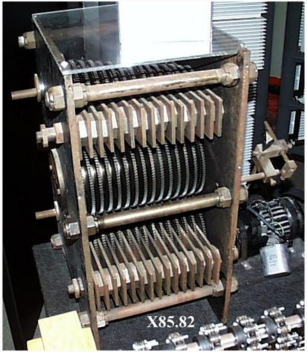
Kyber is a general purpose key encapsulation mechanism (KEM).

- Think of it as a successor to the NewHope key exchange from 2016.
  - Comparable computational efficiency (can be used in an ephemeral setting).
  - Smaller public keys and ciphertexts.
  - CCA secure (keys can be reused).

Why do we need new cryptographic primitives?

A (possible) look at 100 years of factoring machines

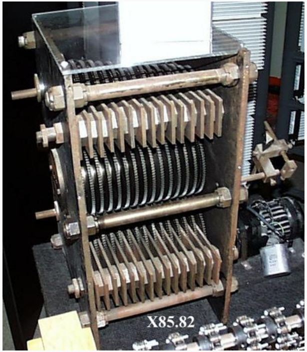
1931



Factors 60-bit numbers in < 1 hour.

A (possible) look at 100 years of factoring machines

1931



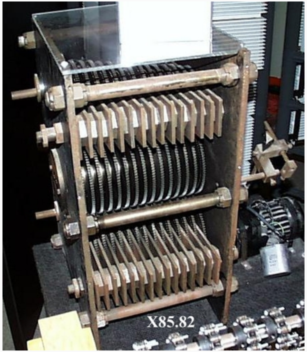
Factors 60-bit numbers
in < 1 hour.

1981

Factors 160-bit numbers
in < 24 hours.

A (possible) look at 100 years of factoring machines

1931

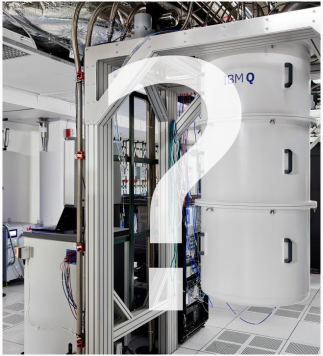


Factors 60-bit numbers in < 1 hour.

1981

Factors 160-bit numbers in < 24 hours.

2031



Factors 2048-bit numbers in < 30 hours.

The cryptocalypse?!?

That depends.

The cryptocalypse?!?

That depends.

- Factoring and discrete log are not fundamentally difficult.
- Large quantum computers may be built soon.

The cryptocalypse?!?

That depends.

- Factoring and discrete log are not fundamentally difficult.
- Large quantum computers may be built soon.

Ask yourself:

- How are you using cryptography now?
    - No real threat to symmetric crypto.
- How strong is your adversary?
    - Willing to wait 10+ years?
    - Willing to spend 30+ hours of compute, per key, on a $1bn+ machine?

The cryptocalypse?!?

That depends.

- Factoring and discrete log are not fundamentally difficult.
- Large quantum computers may be built soon.

Ask yourself:

- How are you using cryptography now?
    - No real threat to symmetric crypto.
- How strong is your adversary?
    - Willing to wait 10+ years?
    - Willing to spend 30+ hours of compute, per key, on a $1bn+ machine?

And at least think about upgrade path.

NIST standardization process

US National Institute of Standards and Technology put out a call for

- Key Encapsulation Mechanisms (KEMs),
- Public key encryption schemes,
- Digital signature schemes.

Timeline:

- [x] Nov. 2017: First round submission deadline.
- [x] Apr. 2018: First workshop.
- [ ] Late 2018/Early 2019: Second round candidate announcement.
- [ ] Aug. 2019: Second workshop.
- [ ] 2020/2021: Third round?
- [ ] 2022/2024: Draft standards.

NIST standardization process

45 KEM submissions. 21 are "lattice based."

Of these:

- 12 are built on same "chassis" as Kyber.
    - Approximate key transport via noisy dot products.
    - Syntactically similar to Lindner–Peikert 2011 (based on LWE [Regev, 2005]).

NIST standardization process

45 KEM submissions. 21 are “lattice based.”

Of these:

- 12 are built on same “chassis” as Kyber.
  - Approximate key transport via noisy dot products.
  - Syntactically similar to Lindner–Peikert 2011 (based on LWE [Regev, 2005]).
- 1 is syntactically similar to original LWE system.
- 3 are based on NTRU [Hoffstein–Pipher–Silverman, 1998].
- Remaining 5 are harder to classify.

NIST standardization process

45 KEM submissions. 21 are “lattice based.”

Of these:

- 12 are built on same “chassis” as Kyber.
  - Approximate key transport via **noisy dot products**.
  - Syntactically similar to Lindner–Peikert 2011 (based on LWE [Regev, 2005]).

Assumption: One-wayness and/or indistinguishability of “noisy dot products”

Suppose

- $\mathbf{a}$ is a known vector of *scalars* chosen uniformly at random.
- $\mathbf{s}$ is a secret vector of scalars of known distribution.
- e is a secret scalar of known distribution.

Then, with appropriate restrictions on

1. the definition of “scalar” and
2. and distribution of $\mathbf{s}$ and e,

it is hard to distinguish the *noisy dot product*

$$\mathbf{a}^{\mathsf{T}} \cdot \mathbf{s} + e$$

from a uniform scalar. Even when same $\mathbf{s}$ is used with many $\mathbf{a}$'s and e's.

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same “chassis” as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}\right) \cdot \mathbf{s}$$
$$= \quad \mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 - \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}\right) \cdot \mathbf{s}$$

$$= \quad \mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 - \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same “chassis” as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}\right) \cdot \mathbf{s}$$

$$= \quad \mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 - \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - **Syntactically similar** to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$

$$= \quad \mathbf{r}^{\mathsf{T}} \cdot \mathbf{e}_1 - \mathbf{e}_2^{\mathsf{T}} \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate key transport via noisy dot products.
  - Syntactically similar to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}\right) \cdot \mathbf{s}$$
$$= \quad \mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 - \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

45 KEMs. 21 based on lattices. Of these:

- 12 are built on same "chassis" as Kyber.
  - Approximate **key transport** via noisy dot products.
  - Syntactically similar to Lindner–Peikert 2011.

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^T \cdot \mathbf{A} + \mathbf{e}_2^T$.
- A scalar $\mathbf{r}^T \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^T \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^T \cdot \mathbf{A} + \mathbf{e}_2^T\right) \cdot \mathbf{s}$$
$$= \quad \mathbf{r}^T \cdot \mathbf{e}_1 - \mathbf{e}_2^T \cdot \mathbf{s} + e_3$$

Chassis: Approximate key transport using noisy dot products

…Approximate **key transport**…

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$
$$= \quad \mathbf{r}^{\mathsf{T}} \cdot \mathbf{e}_1 - \mathbf{e}_2^{\mathsf{T}} \cdot \mathbf{s} + e_3.$$

Chassis: Approximate key transport using noisy dot products

...Approximate **key transport**...

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}$.
- A scalar $\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3 + m$.

Alice can compute:

$$\left(\mathbf{r}^\mathsf{T} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1 + m) + e_3\right) - \left(\mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_2^\mathsf{T}\right) \cdot \mathbf{s}$$

$$= \quad \mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 - \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3 + m.$$

Chassis: Approximate key transport using noisy dot products

...Approximate **key transport**...

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3 + m$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1 + m) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$
$$= \quad \mathbf{r}^{\mathsf{T}} \cdot \mathbf{e}_1 - \mathbf{e}_2^{\mathsf{T}} \cdot \mathbf{s} + e_3 + m.$$

Chassis: Approximate key transport using noisy dot products

...**Approximate** key transport...

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^T \cdot \mathbf{A} + \mathbf{e}_2^T$.
- A scalar $\mathbf{r}^T \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3 + m$.

Alice can compute:

$$\left(\mathbf{r}^T \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1 + m) + e_3\right) - \left(\mathbf{r}^T \cdot \mathbf{A} + \mathbf{e}_2^T\right) \cdot \mathbf{s}$$

$$= \underbrace{\mathbf{r}^T \cdot \mathbf{e}_1 - \mathbf{e}_2^T \cdot \mathbf{s} + e_3}_{noise} + m.$$

Chassis: Approximate key transport using noisy dot products

...Approximate key transport...

Alice contributes:

- A matrix $\mathbf{A}$
- A column vector $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1$.

Bob contributes:

- A row vector $\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}$.
- A scalar $\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1) + e_3 + m$.

Alice can compute:

$$\left(\mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}_1 + m) + e_3\right) - \left(\mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_2^{\mathsf{T}}\right) \cdot \mathbf{s}$$

$$= \underbrace{\mathbf{r}^{\mathsf{T}} \cdot \mathbf{e}_1 - \mathbf{e}_2^{\mathsf{T}} \cdot \mathbf{s} + e_3}_{\textit{noise}} + m.$$

$\Rightarrow$ Bob transmits *one noisy scalar* to Alice.

Details

I've omitted several crucial details:

- the definition of "scalar" and the dimensions of $\mathbf{A}$,
- distributions for $\mathbf{s}, \mathbf{e}_1, \mathbf{r}, \mathbf{e}_2, e_3$,
- encoding of key material into m,
- how to go from approximate to exact key transport.

These are the attributes that distinguish the 12 syntactically similar KEM submissions.

Scalars – the main innovation of Kyber

Most schemes go to one of two extremes.

Scalars – the main innovation of Kyber

Most schemes go to one of two extremes.

FrodoKEM-640 "LWE"	• **A** is 640×640. • Scalars are $\mathbb{Z}_q$.	• Alice decodes 2 bits from each of 64 noisy scalars (from 8 parallel exchanges). 128 total.

Scalars – the main innovation of Kyber

Most schemes go to one of two extremes.

FrodoKEM-640 "LWE"	• **A** is 640×640. • Scalars are $\mathbb{Z}_q$.	• Alice decodes 2 bits from each of 64 noisy scalars (from 8 parallel exchanges). 128 total.
NewHope1024 "RLWE"	• **A** is 1×1. • Scalars are $\mathbb{Z}_q^{1024}$ with the multiplication of $$\mathbb{Z}_q[x]/(x^{1024}+1).$$	• Alice decodes 1 bit from each $\mathbb{Z}_q$-coefficient of the noisy scalar. 1024 total. 4−to−1 bit error correction.

Scalars – the main innovation of Kyber

Kyber strikes a balance.

Scalars – the main innovation of Kyber

Kyber strikes a balance.

Kyber768 “MLWE”	• **A** is 3×3. • Scalars are $\mathbb{Z}_q^{256}$ with the multiplication of $$\mathbb{Z}_q[x]/(x^{256}+1).$$	• Alice decodes 1 bit from each $\mathbb{Z}_q$-coefficient of the noisy scalar. 256 total.

Scalars – the main innovation of Kyber

Kyber strikes a balance.

Kyber768 "MLWE"	• $\mathbf{A}$ is 3×3. • Scalars are $\mathbb{Z}_q^{256}$ with the multiplication of $$\mathbb{Z}_q[x]/(x^{256}+1).$$	• Alice decodes 1 bit from each $\mathbb{Z}_q$-coefficient of the noisy scalar. 256 total.

- Dimension 768 is sweet spot for lattice security.
- 256-bit symmetric keys are standard.
- For Kyber512 and Kyber1024: change the size of $\mathbf{A}$.

Exact key transport

Sketch:

- $m = \lfloor q/2 \rceil m'$, where m' is key to encapsulate.
- Ensure that the coefficients of $\mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 + \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$ have magnitude less than $q/4$.
- Recover m' by "rounding" noisy scalar.

Exact key transport

Sketch:

- $m = \lfloor q/2 \rfloor m'$, where m' is key to encapsulate.
- Ensure that the coefficients of $\mathbf{r}^\mathsf{T} \cdot \mathbf{e}_1 + \mathbf{e}_2^\mathsf{T} \cdot \mathbf{s} + e_3$ have magnitude less than $q/4$.
- Recover m' by "rounding" noisy scalar.

This is not *guaranteed* to succeed.

We fix distributions for $\mathbf{s}, \mathbf{e}_1, \mathbf{r}, \mathbf{e}_2$, and e_3 so that it fails with negligible probability.

CCA transform – slightly modified Fujisaki-Okamoto

High level idea:

- Bob expands all the random bits he needs for encryption from a seed.
- He takes the seed to be a hash of Alice's public key and m.
- After decryption, Alice recovers the seed and checks that the ciphertext was generated correctly.

CCA transform – slightly modified Fujisaki-Okamoto

High level idea:

- Bob expands all the random bits he needs for encryption from a seed.
- He takes the seed to be a hash of Alice's public key and m.
- After decryption, Alice recovers the seed and checks that the ciphertext was generated correctly.

Including Alice's public key in seed is a defense against multi-target attacks.

Efficiency enhancements

- Choose q to support a length 256 number theoretic transform (think: FFT).

$$\mathbb{Z}_q^{256} \text{ with } \mathbb{Z}_q[x]/(x^{256}+1) \text{ mult.} \xleftrightarrow{\text{NTT}} \mathbb{Z}_q^{256} \text{ with coefficient-wise mult.}$$

Efficiency enhancements

- Choose q to support a length 256 number theoretic transform (think: FFT).

$$\mathbb{Z}_q^{256} \text{ with } \mathbb{Z}_q[x]/(x^{256}+1) \text{ mult.} \xleftrightarrow{\text{NTT}} \mathbb{Z}_q^{256} \text{ with coefficient-wise mult.}$$

- Sample entries of $\mathbf{A}$ in "NTT domain".

Efficiency enhancements

- Choose q to support a length 256 number theoretic transform (think: FFT).

$$\mathbb{Z}_q^{256} \text{ with } \mathbb{Z}_q[x]/(x^{256}+1) \text{ mult.} \xleftrightarrow{\text{NTT}} \mathbb{Z}_q^{256} \text{ with coefficient-wise mult.}$$

- Sample entries of $\mathbf{A}$ in “NTT domain”.
- Expand $\mathbf{A}$ from a short seed.

Efficiency enhancements

- Choose q to support a length 256 number theoretic transform (think: FFT).

$$\mathbb{Z}_q^{256} \text{ with } \mathbb{Z}_q[x]/(x^{256}+1) \text{ mult.} \xleftrightarrow{\text{NTT}} \mathbb{Z}_q^{256} \text{ with coefficient-wise mult.}$$

- Sample entries of $\mathbf{A}$ in "NTT domain".
- Expand $\mathbf{A}$ from a short seed.
- Compress Alice's vector, Bob's vector, and Bob's scalar.

Efficiency enhancements

- Choose q to support a length 256 number theoretic transform (think: FFT).

$$\mathbb{Z}_q^{256} \text{ with } \mathbb{Z}_q[x]/(x^{256}+1) \text{ mult.} \xleftrightarrow{\text{NTT}} \mathbb{Z}_q^{256} \text{ with coefficient-wise mult.}$$

- Sample entries of $\mathbf{A}$ in "NTT domain".
- Expand $\mathbf{A}$ from a short seed.
- Compress Alice's vector, Bob's vector, and Bob's scalar.
    - Careful! Not just an efficiency tweak.
    - Changes distribution of $\mathbf{e}_1$, $\mathbf{e}_2$ and e_3.
    - Affects correctness and security proofs
      (Thanks to Jan Pieter D'Anvers for pointing out an error in earlier version).

Parameter sets and performance

		Kyber512	Kyber768	Kyber1024
Size (in bytes)	**pk:**	736	1088	1440
	ct:	800	1152	1504
Haswell Cycles (Ref)	**gen:**	141 872	243 004	368 564
	enc:	205 468	332 616	481 042
	dec:	246 040	394 424	558 740
Haswell Cycles (AVX2)	**gen:**	55 160	85 472	121 056
	enc:	75 680	112 660	157 964
	dec:	74 428	108 904	154 952

X25519: gen: 90668 cycles, enc/dec: 138963

Security

	Kyber512	Kyber768	Kyber1024
Best quantum attack cost	2^{103}	2^{161}	2^{221}

Note: units of “cost” are $\gg$ bit operations.

Correctness

	Kyber512	Kyber768	Kyber1024
Decryption failure probability	2^{-145}	2^{-142}	2^{-169}

Takeaway: think about your upgrade path

- How hard is it for you to "drop-in" new crypto?
- Is there anything you can do now to make that process easier?
- Can you tolerate $\approx$ 1kB public keys and ciphertexts.

More info online

https://pq-crystals.org/kyber

Thanks!