How Multi-Recipient KEMs can help the Deployment of Post-Quantum Cryptography

Joël Alwen
AWS

Matthew Campagna
AWS

Dominik Hartmann
AWS

Shuichi Katsumata
PQShield & AIST

Eike Kiltz
Ruhr University Bochum

Jake Massimo
AWS

Marta Mularczyk
AWS

Guillermo Pascual-Perez
ISTA

Thomas Prest
PQShield

Peter Schwabe
MPI & Radboud University

Fifth PQC Standardization Conference

I want to share a key with 100 people

- → Encapsulating K to 1 party using Kyber: **768 bytes**
- → Encapsulating K to 100 parties using Kyber: **76 800 bytes**
- → Encapsulating K to 100 parties using a “multi-recipient Kyber”: **5 504 bytes**

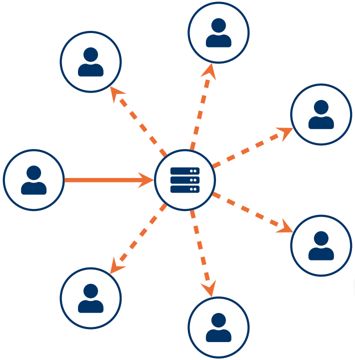
How do we gain this factor 14?

Multi-Recipient KEMs

Main question

How efficiently can we share a session key K between (N + 1) users?

- **Naive solution with El Gamal:**
  - Send $(g^{r_i}, \mathrm{pk}_i^{r_i} \cdot \mathrm{K})$ for each user *i*
- **Variant by Kurosawa, PKC 2002:**
  - Send $(g^{r}, \mathrm{pk}_1^{r} \cdot \mathrm{K}, \ldots, \mathrm{pk}_N^{r} \cdot \mathrm{K})$
  - Asymptotically, saves a factor **2**



Decomposability

Definition. In a decomposable encryption scheme, a ciphertext can be decomposed in key-dependent and key-independent parts:

$$\mathrm{Enc}(\mathrm{pk}_i, \mathtt{msg}) = \underbrace{\mathtt{ctxt}_0}_{\mathrm{Enc}^{\mathtt{ind}}(r_0)} \; \underbrace{\widehat{\mathtt{ctxt}}_i}_{\mathrm{Enc}^{\mathtt{dep}}(\mathrm{pk}_i, \mathtt{msg}, r_0, r_i)}$$

Decomposability

Definition. In a decomposable encryption scheme, a ciphertext can be decomposed in key-dependent and key-independent parts:

$$\mathsf{Enc}(\mathsf{pk}_i, \mathtt{msg}) = \underbrace{\mathtt{ctxt}_0}_{\mathsf{Enc}^{\mathtt{ind}}(r_0)} \; \underbrace{\widehat{\mathtt{ctxt}_i}}_{\mathsf{Enc}^{\mathtt{dep}}(\mathsf{pk}_i, \mathtt{msg}, r_0, r_i)}$$

El Gamal is decomposable. Let a ciphertext $\mathtt{ctxt} = (g^r, \mathsf{pk}_1^r \cdot \mathtt{msg})$ with $\mathsf{pk}_1 = g^{\mathsf{sk}_1}$.

1. $\mathtt{ctxt}_0 = g^r$.
2. $\widehat{\mathtt{ctxt}_1} = \mathsf{pk}_1^r \cdot \mathtt{msg}$.

A ciphertext with N recipients will be $\vec{\mathtt{ctxt}} = (\mathtt{ctxt}_0, \widehat{\mathtt{ctxt}_1}, \ldots, \widehat{\mathtt{ctxt}_N})$.
Key generation and decryption remain the same.

Decomposability

Definition. In a decomposable encryption scheme, a ciphertext can be decomposed in key-dependent and key-independent parts:

$$\mathsf{Enc}(\mathtt{pk}_i, \mathtt{msg}) \quad = \quad \underbrace{\mathtt{ctxt}_0}_{\mathsf{Enc}^{\mathtt{ind}}(r_0)} \; \underbrace{\widehat{\mathtt{ctxt}_i}}_{\mathsf{Enc}^{\mathtt{dep}}(\mathtt{pk}_i, \mathtt{msg}, r_0, r_i)}$$

El Gamal is decomposable. Let a ciphertext $\mathtt{ctxt} = (g^r, \mathtt{pk}_1^r \cdot \mathtt{msg})$ with $\mathtt{pk}_1 = g^{\mathtt{sk}_1}$.

1. $\mathtt{ctxt}_0 = g^r$.
2. $\widehat{\mathtt{ctxt}_1} = \mathtt{pk}_1^r \cdot \mathtt{msg}$.

A ciphertext with N recipients will be $\vec{\mathtt{ctxt}} = (\mathtt{ctxt}_0, \widehat{\mathtt{ctxt}_1}, \dots, \widehat{\mathtt{ctxt}_N})$.
Key generation and decryption remain the same.

Questions:

1. What about CCA security?
   - ✔ ($\exists$ decomposable IND-CPA mPKE) $\overset{\text{F-O}}{\Longrightarrow}$ ($\exists$ decomposable IND-CCA mKEM).
2. Is Kyber securely decomposable?

mKyber: a Kyber-based mKEM

Kyber, CPA version

PQ SHIELD

Keygen ()

1. Sample $\mathbf{A}$ and short $\mathbf{s}, \mathbf{e}$
2. $\mathbf{b} \leftarrow \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$
3. $\mathsf{dk} := (\mathbf{s}, \mathbf{E}), \mathsf{ek} := \mathbf{b}$

Enc$(\mathsf{ek}, \mathtt{msg})$

1. Sample short row vectors $\mathbf{r}, \mathbf{e}', \mathbf{e}''$
2. $\mathbf{u} \leftarrow \mathbf{r} \cdot \mathbf{A} + \mathbf{e}'$
3. $\mathbf{v} \leftarrow \mathbf{r} \cdot \mathbf{b} + \mathbf{e}'' + \mathsf{Encode}(\mathtt{msg})$
4. $\mathtt{ctxt} := (\mathbf{u}, \mathbf{v})$

Dec$(\mathsf{dk}, \mathtt{ctxt})$

1. $\mathtt{msg} \leftarrow \mathsf{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{S})$

This construction is decomposable:

- Use the same $\mathbf{A}$ for all public keys.
- $\mathbf{u}$ is then independent of $\mathtt{ek}$ and $\mathtt{msg}$.

$\mathsf{Enc}(\mathtt{ek} \quad \mathbf{b}, \mathtt{msg})$

1. Sample short matrices $\mathbf{r}, \mathbf{e}', \mathbf{e}''$
2. $\mathbf{u} \leftarrow \mathbf{rA} + \mathbf{e}'$
3. $\mathbf{v} \leftarrow \mathbf{rb} + \mathbf{e}'' + \mathsf{Encode}(\mathtt{msg})$
4. $\mathtt{ctxt} := (\mathbf{u}, \mathbf{v})$

This construction is decomposable:

- → Use the same $\mathbf{A}$ for all public keys.
- → $\mathbf{u}$ is then independent of $\mathtt{ek}$ and $\mathtt{msg}$.

Enc$(\mathtt{ek} \quad \mathbf{b}, \mathtt{msg})$

1. Sample short matrices $\mathbf{r}, \mathbf{e}', \mathbf{e}''$
2. $\mathbf{u} \leftarrow \mathbf{rA} + \mathbf{e}'$
3. $\mathbf{v} \leftarrow \mathbf{rb} + \mathbf{e}'' + \mathrm{Encode}(\mathtt{msg})$
4. $\mathtt{ctxt} := (\mathbf{u}, \mathbf{v})$

$\Longrightarrow$

MultiEnc$(\{\mathtt{ek}_1, \ldots, \mathtt{ek}_N\}, \mathtt{msg})$

1. Sample short matrices $\mathbf{r}, \mathbf{e}'$
2. $\mathbf{u} \leftarrow \mathbf{rA} + \mathbf{e}'$
3. For $i = 1, \ldots, N$:
   1. Sample a short matrix $\mathbf{e}_i''$
   2. $\mathbf{v}_i \leftarrow \mathbf{rb}_i + \mathbf{e}_i'' + \mathrm{Encode}(\mathtt{msg})$
4. $\overrightarrow{\mathtt{ctxt}} := (\mathbf{u}, \mathbf{v}_1, \ldots, \mathbf{v}_N)$

This construction is decomposable:

- → Use the same **A** for all public keys.
- → **u** is then independent of ek and msg.

Enc$(\mathtt{ek}\quad \mathbf{b}, \mathtt{msg})$

1. Sample short matrices $\mathbf{r}, \mathbf{e}', \mathbf{e}''$
2. $\mathbf{u} \leftarrow \mathbf{rA} + \mathbf{e}'$
3. $\mathbf{v} \leftarrow \mathbf{rb} + \mathbf{e}'' + \text{Encode}(\mathtt{msg})$
4. $\mathtt{ctxt} := (\mathbf{u}, \mathbf{v})$

$\Longrightarrow$

MultiEnc$(\{\mathtt{ek}_1, \ldots, \mathtt{ek}_N\}, \mathtt{msg})$

1. Sample short matrices $\mathbf{r}, \mathbf{e}'$
2. $\mathbf{u} \leftarrow \mathbf{rA} + \mathbf{e}'$
3. For $i = 1, \ldots, N$:
   1. Sample a short matrix $\mathbf{e}_i''$
   2. $\mathbf{v}_i \leftarrow \mathbf{rb}_i + \mathbf{e}_i'' + \text{Encode}(\mathtt{msg})$
4. $\vec{\mathtt{ctxt}} := (\mathbf{u}, \mathbf{v}_1, \ldots, \mathbf{v}_N)$

Are we done? No!

1. Security?
2. Efficiency?

PQ SHIELD

What assumptions do we rely on?

	Kyber	mKyber
Public key security	MLWE, $O(1)$ samples	MLWE, $O(1)$ samples
Ciphertext security	MLWE, $O(1)$ samples	MLWE, $O(N)$ samples

Which attacks are relevant against MLWE?

	Primal (Lattice)	Dual (Lattice)	Arora-Ge (Algebraic)	BKW (Combinatorial)
$O(1)$ samples	✓	✓	-	-
$O(N)$ samples	✓	✓	✓	✓

What assumptions do we rely on?

	Kyber	mKyber
Public key security	MLWE, $O(1)$ samples	MLWE, $O(1)$ samples
Ciphertext security	MLWE, $O(1)$ samples	MLWE, $O(N)$ samples

Which attacks are relevant against MLWE?

	Primal (Lattice)	Dual (Lattice)	Arora-Ge (Algebraic)	BKW (Combinatorial)
$O(1)$ samples	✔	✔	-	-
$O(N)$ samples	✔	✔	✔	✔

Are we in trouble? No.

- ✔ Bit dropping on the $\mathbf{v}_i$ makes Arora-Ge + BKW hard to the point of irrelevance

Wrapping up mKyber

	Parameters								Sizes in bytes		
	q	n	k	η_1	η_2	d_u	d_v	$\lvert\mathtt{msg}\rvert$	$\lvert\mathtt{ek}\rvert$	$\lvert\mathtt{u}\rvert$	$\lvert\mathtt{v}\rvert$
Kyber-512	3329	256	2	3	2	10	4	32	800	640	128
mKyber-512	3329	256	2	3	2	11	3	16	768	704	**48**

Wrapping up mKyber

PQ SHIELD

	Parameters								Sizes in bytes		
	q	n	k	η_1	η_2	d_u	d_v	$\lvert\mathtt{msg}\rvert$	$\lvert\mathtt{ek}\rvert$	$\lvert\mathtt{u}\rvert$	$\lvert\mathtt{v}\rvert$
Kyber-512	3329	256	2	3	2	10	4	32	800	640	128
mKyber-512	3329	256	2	3	2	11	3	16	768	704	**48**

Not covered in this talk (see paper):

- We can achieve IND-CCA security
- We can upgrade to adaptive security by doubling the ciphertext size (amKyber)
- Parameter selection differs from the KEM setting

Application 1: Broadcast

Setting

One sender sends the same keying material K to N parties

- Example application: state synchronisation in HSM fleet
- Perfect fit for mKEM!
- Also slightly simpler than naive solution (no DEM)

Setting

One sender sends the same keying material K to N parties

- → Example application: state synchronisation in HSM fleet
- → Perfect fit for mKEM!
- → Also slightly simpler than naive solution (no DEM)

Example:

- ☺ 1 Kyber ciphertext:

One sender sends the same keying material *K* to *N* parties

- Example application: state synchronisation in HSM fleet
- Perfect fit for mKEM!
- Also slightly simpler than naive solution (no DEM)

Example:

1 Kyber ciphertext:

N Kyber ciphertexts:

Setting

One sender sends the same keying material K to N parties

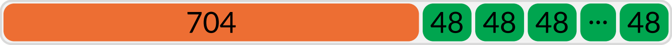
- → Example application: state synchronisation in HSM fleet
- → Perfect fit for mKEM!
- → Also slightly simpler than naive solution (no DEM)

Example:

1 Kyber ciphertext:

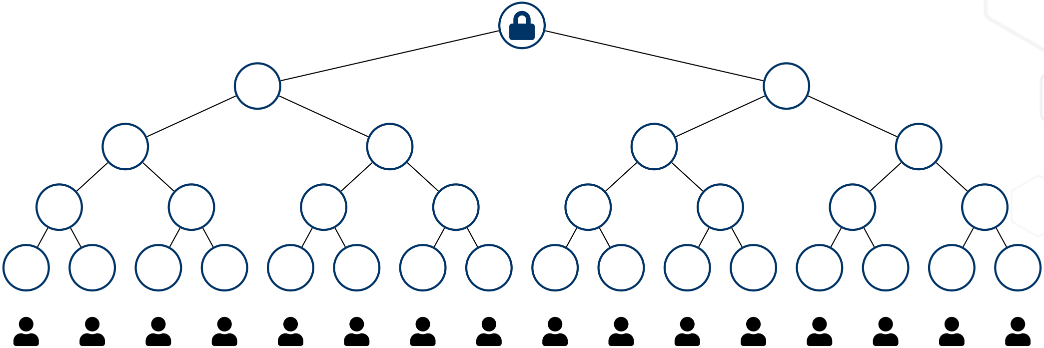
N Kyber ciphertexts:

1 mKyber ciphertext for N parties:



Application 2: MLS

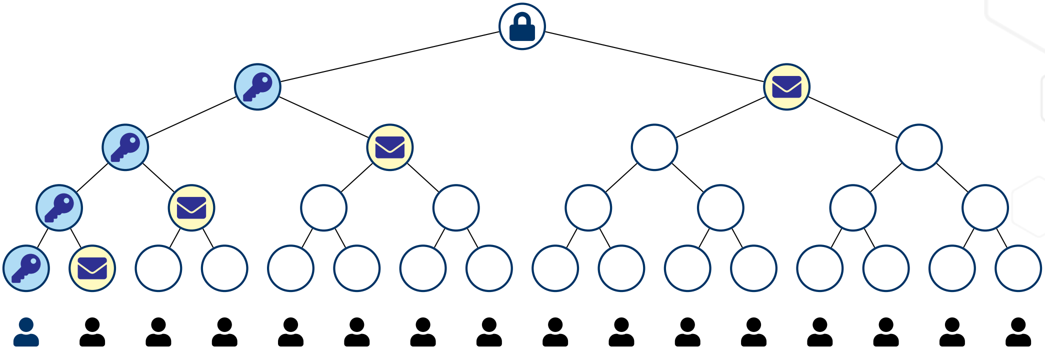
The TreeKEM construction (MLS)



The N users are arranged as the leaves of a (binary) tree

Tree invariant: (*user* knows the private key of a *node*) $\Leftrightarrow$ (*node* is in the path of *user*)

The TreeKEM construction (MLS)



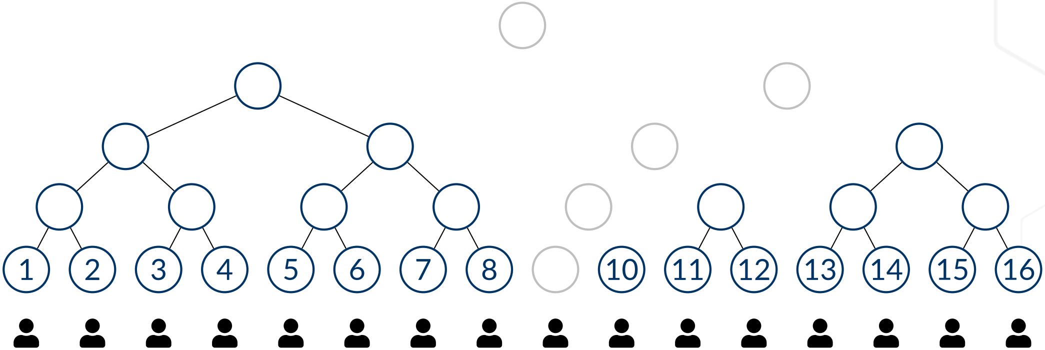
The N users are arranged as the leaves of a (binary) tree

Tree invariant: (*user* knows the private key of a *node*) $\Leftrightarrow$ (*node* is in the path of *user*)

Users routinely update their key material and broadcast:

- All $\lceil \log N \rceil$ encryption keys (🔑) in their direct path
- All $\geq \lceil \log N \rceil$ ciphertexts (✉) in their co-path
- 2 signatures (✍) – one for encryption keys, one for ciphertexts

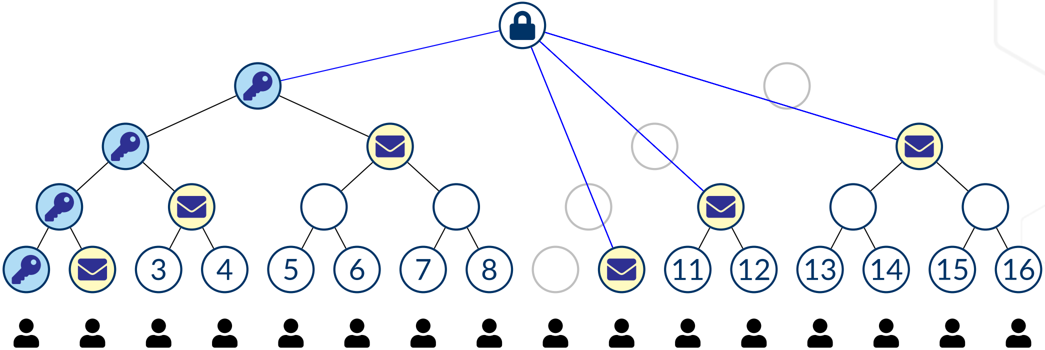
Blanking



When users are removed, their keys are removed for security.

→ This changes the topology of the tree

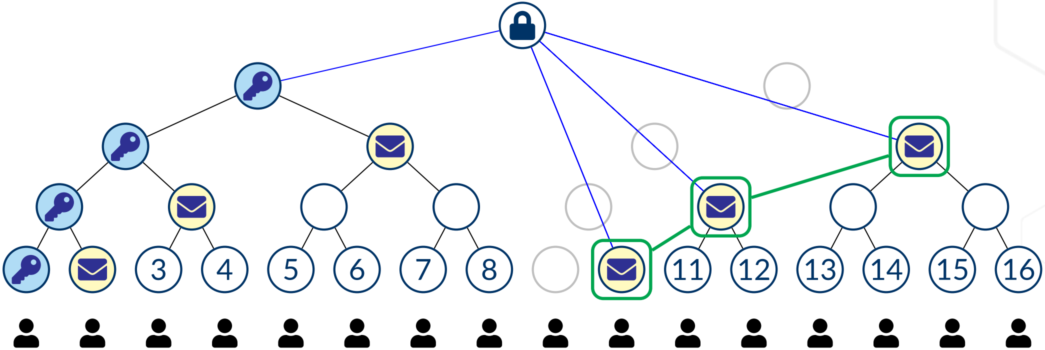
Blanking



When users are removed, their keys are removed for security.

- This changes the topology of the tree
- This increases the number of ciphertext sent (here, $4 \to 6$)

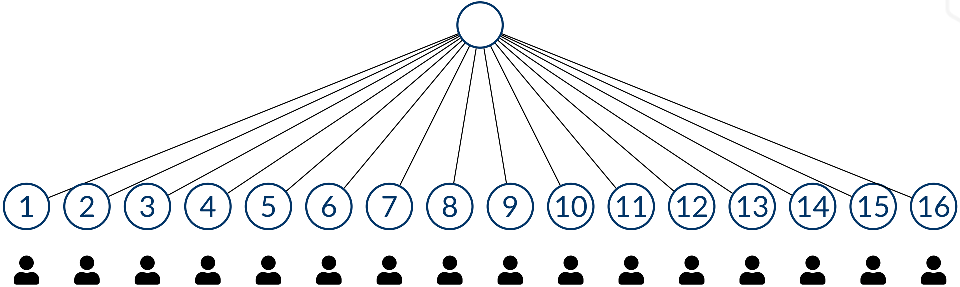
Blanking



When users are removed, their keys are removed for security.

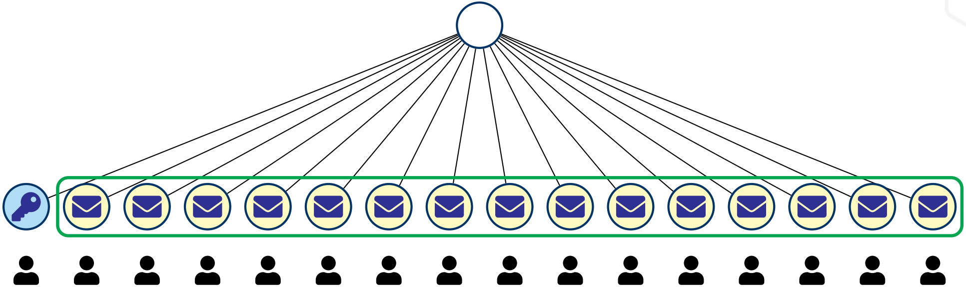
- This changes the topology of the tree
- This increases the number of ciphertext sent (here, $4 \to 6$)
- **Key observation:** Some of these ciphertexts encrypt the same value
  - We can use mKEMs!
  - Allows to always have $\approx$ the best-case behavior

Next step: mKEM-optimised designs



Suppose we replace the binary tree by a star/flat tree:

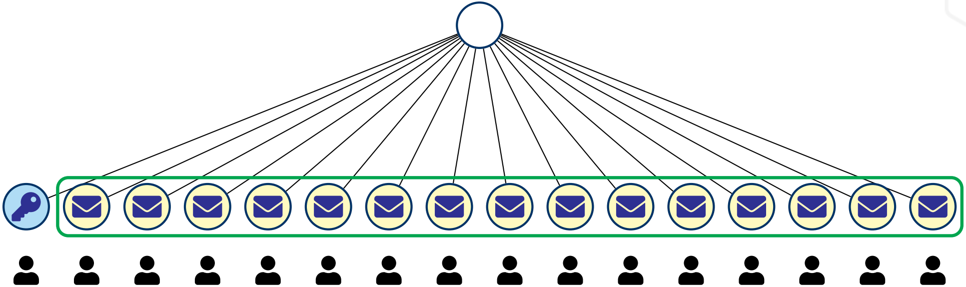
Next step: mKEM-optimised designs



Suppose we replace the binary tree by a star/flat tree:

→ The number of ciphertexts become $O(N)$, but we can compress this using mKEM!

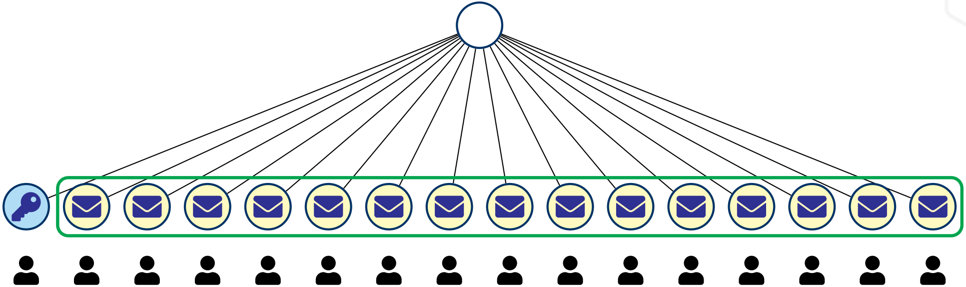
Next step: mKEM-optimised designs



Suppose we replace the binary tree by a star/flat tree:

- The number of ciphertexts become $O(N)$, but we can compress this using mKEM!
- In addition, we can exploit the decomposability and have each user only download a portion $O(1)$ of the ciphertext

Next step: mKEM-optimised designs



Suppose we replace the binary tree by a star/flat tree:

- The number of ciphertexts become $O(N)$, but we can compress this using mKEM!
- In addition, we can exploit the decomposability and have each user only download a portion $O(1)$ of the ciphertext

For more details: *More Efficient Protocols for Post-Quantum Secure Messaging*, RWC 2024. https://www.youtube.com/watch?v=0hCPbu1wrhg

Conclusion

Conclusion

- mKEMs are a **simple and powerful tool** for scalable deployment of PQC
- Many potential applications
- We believe **standardizing mKEMs** would be useful

Questions?