More Efficient Protocols for Post-Quantum Secure Messaging

Keitaro Hashimoto
AIST

Shuichi Katsumata
PQShield & AIST

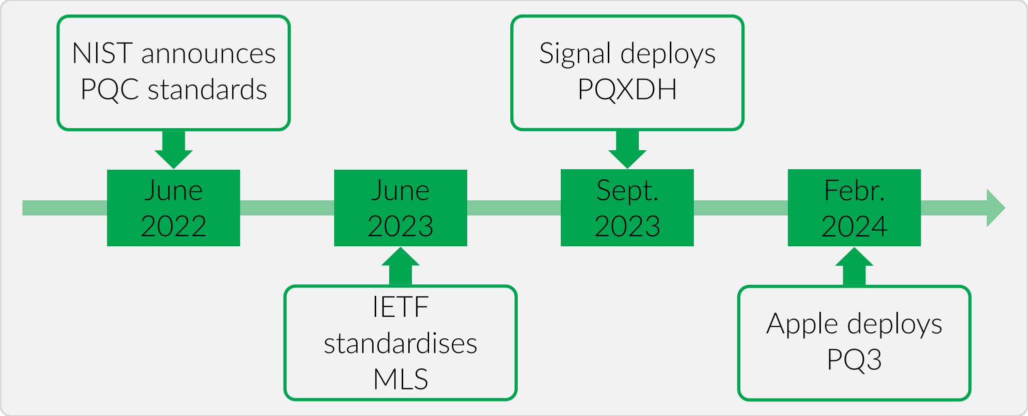
Eamonn W. Postlethwaite
King's College London

Thomas Prest
PQShield

Bas Westerbaan
Cloudflare

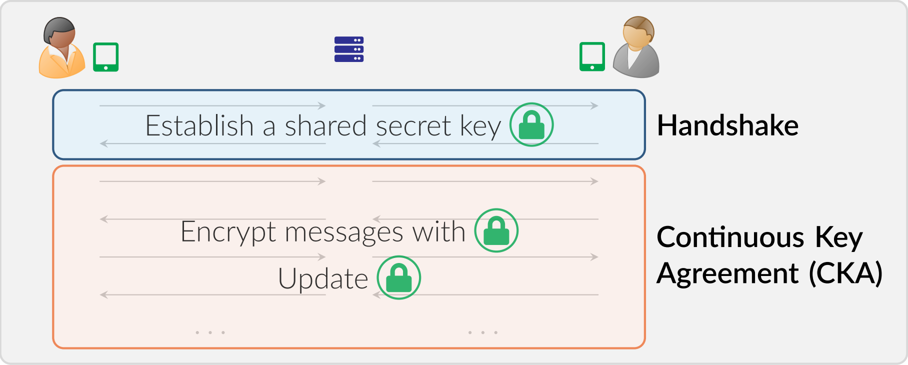
Real World Crypto 2024

Post-quantum and secure messaging



- **MLS:** post-quantum *ready*
- **PQXDH:** post-quantum handshake, classical double ratchet
- **PQ3:** post-quantum handshake, post-quantum double ratchet*
- **Next step:** scalability

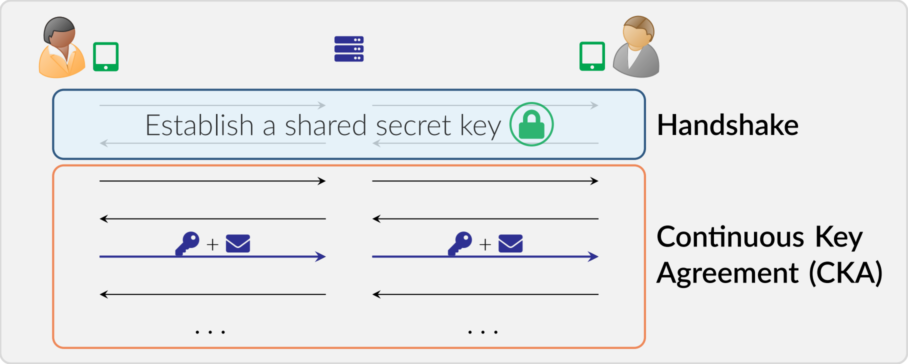
High-level view (two-users case)



Post-quantum instantiations:

- **Handshake:** KEM + (ring) signatures + symmetric crypto [HKKP21, BFG+22]
- **Continuous Key Agreement (CKA):** KEM + symmetric crypto [ACD19]

PQ continuous key agreement



- Each user has a KEM keypair
- updates her cryptographic material as follows:
  1. Generate a new KEM keypair and randomness
  2. Update with randomness
  3. Send new encryption key ( ) + encrypted randomness ( ) to

Both and are able to derive the updated

The Group Setting

This talk: bandwidth

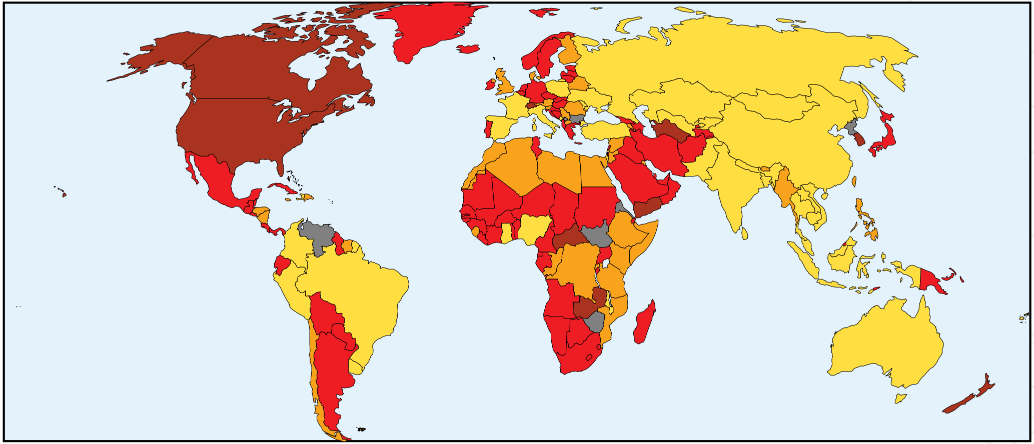
1. **Bandwidth** likely to be a bottleneck of PQ messaging, due to three factors:
   1. Mobile data plans
   2. Post-quantum primitives
   3. Continuous group key agreement (CGKA) protocols
2. Existing CGKAs can incur high bandwidth consumption
   - The bottleneck is in the public-key cryptography
3. Propose a bandwidth-efficient CGKA

How much does 1 GB of mobile data cost?

PQ SHIELD

Median cost:

- ≤ $0.50
- ≤ $1.00
- ≤ $5.00
- ≥ $5.00



Data extracted from a Cable.co.uk study [Cab23]. Notes:

- Small data plans are common in many countries.
- Reaching data caps significantly impacts UX.

Further observations

These observations will guide our design choices:

- $ Uploading and downloading data typically have the **same monetary cost**
- We expect **speed** to impact UX for application messages but not CGKA:
  - Application messages are visible
  - CGKA is invisible (ideally)

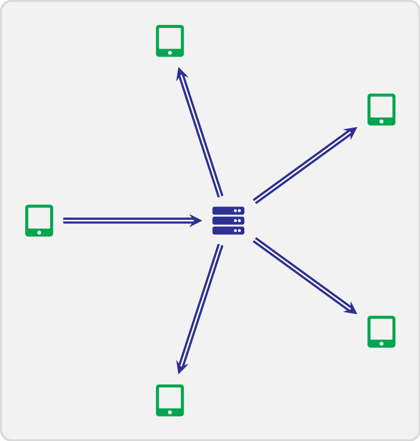
  See [Spe23] for complete data on worldwide mobile speed
- **Large groups** require more frequent key updates
  - Over 1 day, suppose each user gets compromised with probability ε.
    Over T days, a group with N users remains uncompromised with probability

$$(1-\varepsilon)^{N \cdot T} \leq \exp\left(-\varepsilon \cdot N \cdot T\right)$$

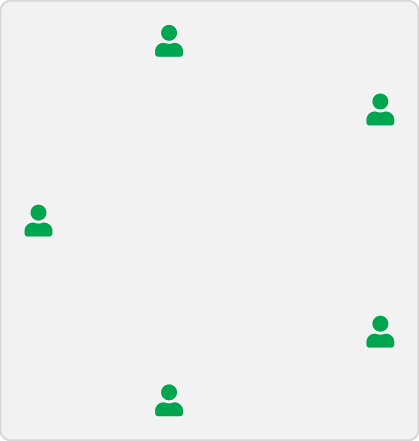
  - But existing CGKA may require high bandwidth (next slides)

Naive CGKA – pairwise channels

Physical layer

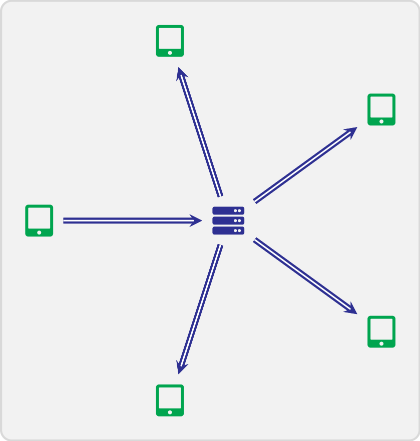


Insider view

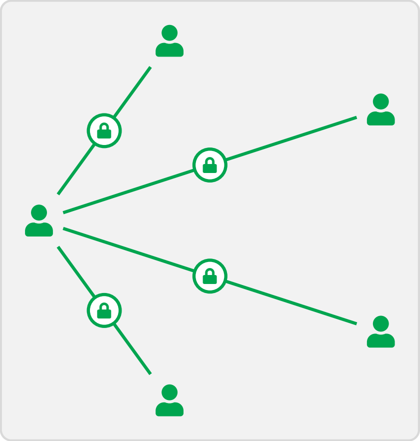


Naive CGKA – pairwise channels

Physical layer

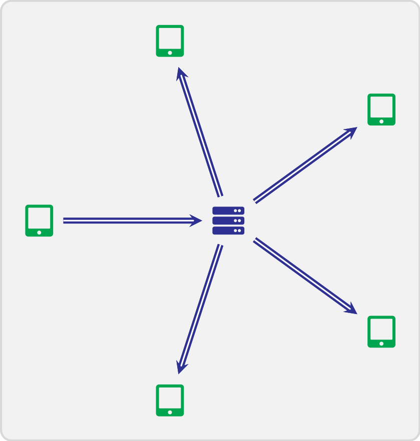


Insider view

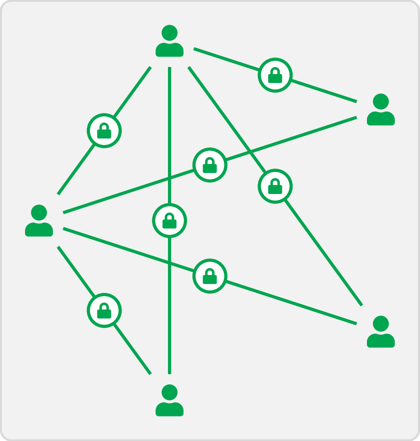


Naive CGKA – pairwise channels

Physical layer

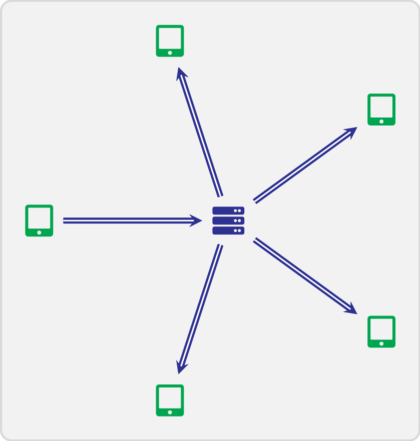


Insider view

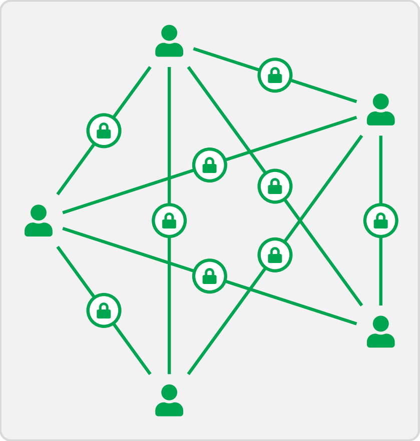


Naive CGKA – pairwise channels

Physical layer

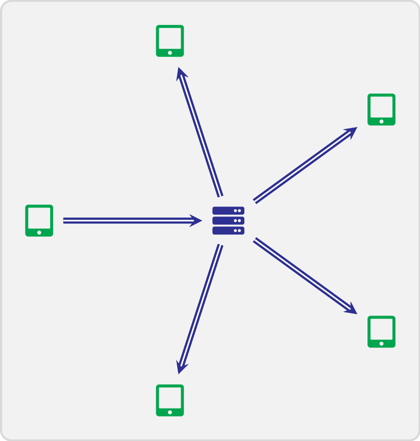


Insider view

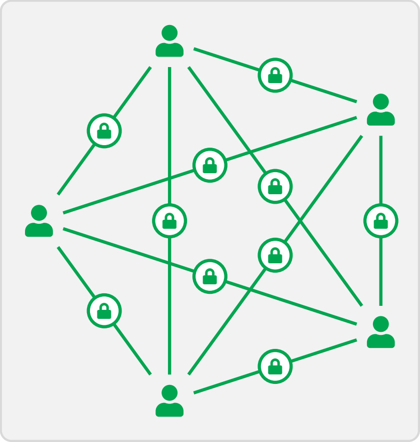


Naive CGKA – pairwise channels

Physical layer

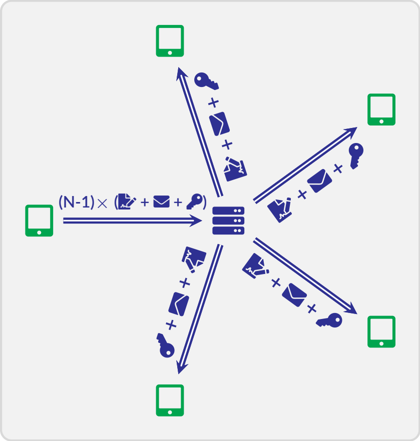


Insider view

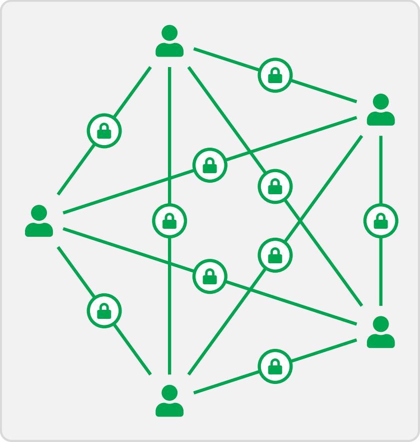


Naive CGKA – pairwise channels

Physical layer



Insider view



Cost of one update with $N = 256$, Kyber-512 and Dilithium-2:

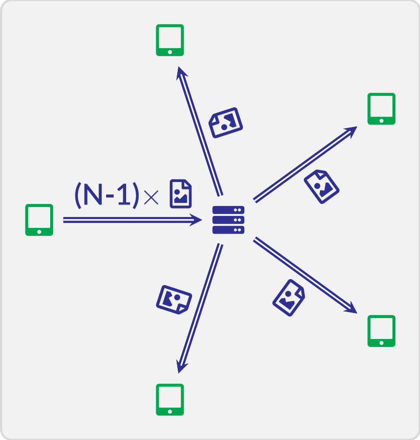
1 MB for the sender, 4 kB for each downloader

(= encryption key, = ciphertext, = signature)

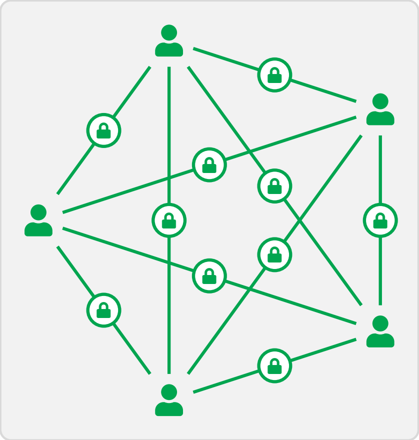
Naive CGKA – pairwise channels

PQ SHIELD

Physical layer



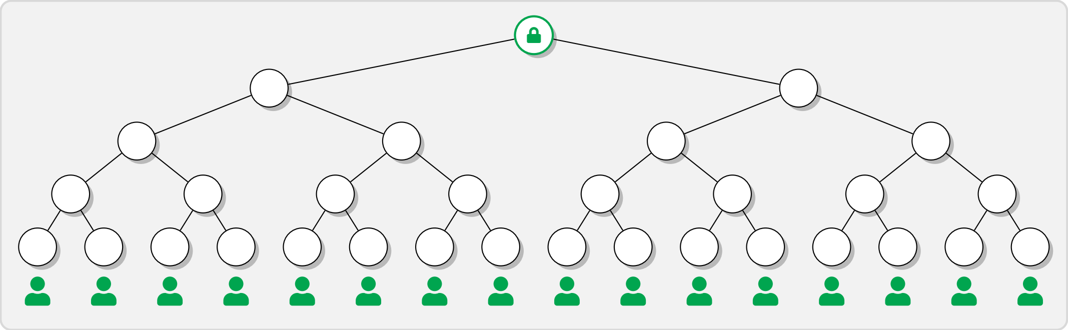
Insider view



Sending a single picture (🖼) of 100 Kilobytes with $N = 256$:

25.5 Megabytes for the sender, 100 kB for each downloader

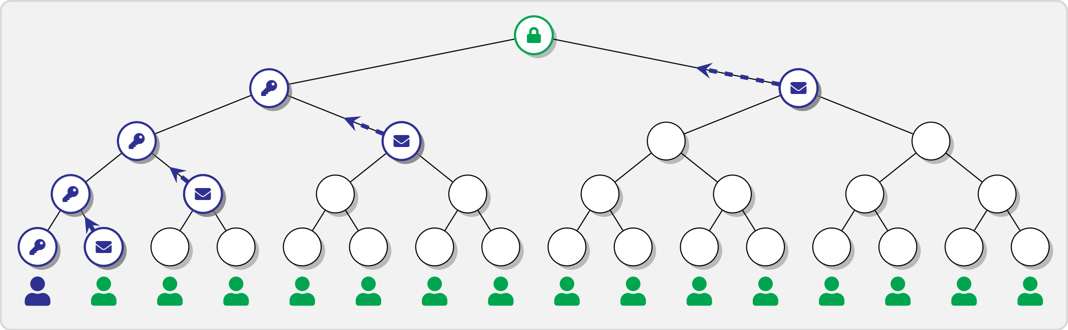
MLS' CGKA – TreeKEM



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

- **Tree invariant:** (*user* knows the private key of a *node*) ⇔ (*node* is in the path of *user*)
- **Application messages:** One key 🔒 for all users

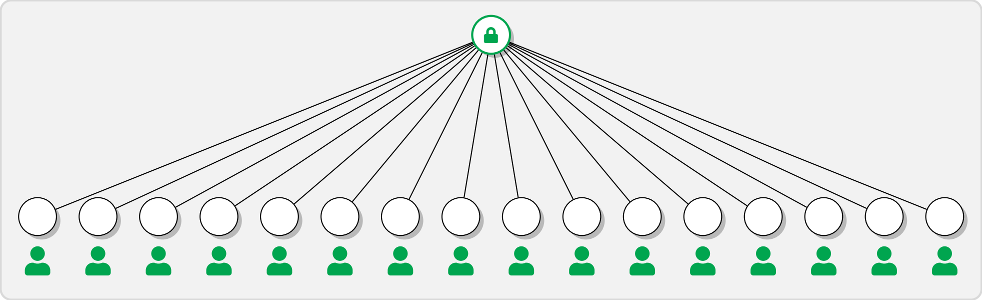
MLS' CGKA – TreeKEM



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

- **Tree invariant:** (*user* knows the private key of a *node*) ⇔ (*node* is in the path of *user*)
- **Application messages:** One key for all users
- When a user (here) updates their key, they broadcast:
  - log N encryption keys ()
  - log N ciphertexts ()
  - 2 signatures () – one for encryption keys, one for ciphertexts

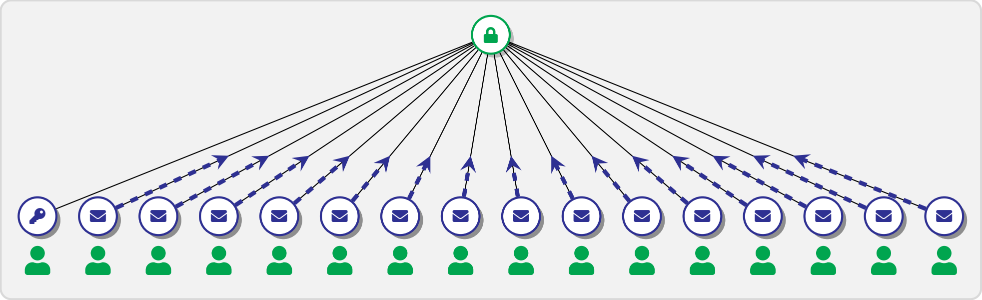
What if we use a flat tree?



This is essentially Chained mKEM [BBN19]

The tree invariant remains identical (and simpler)

What if we use a flat tree?



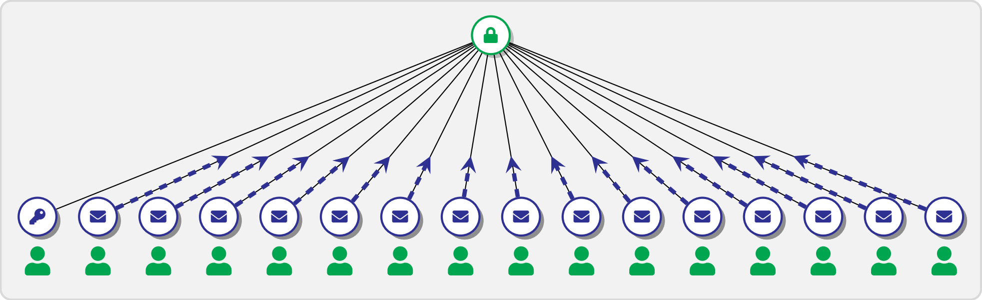
This is essentially Chained mKEM [BBN19]

- The tree invariant remains identical (and simpler)
- When a user (here 👤) updates their key, they broadcast:
  - 1 encryption key (🔑)
  - $N - 1$ ciphertexts (✉)
  - 2 signatures (✍)
- At first glance, less efficient than TreeKEM!

Can we improve efficiency?

What if we use a flat tree & lazy downloading?

PQ SHIELD

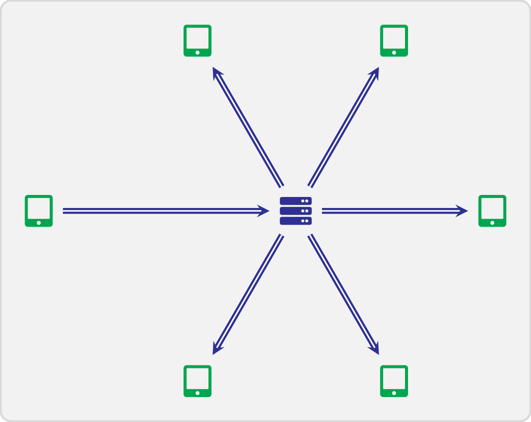


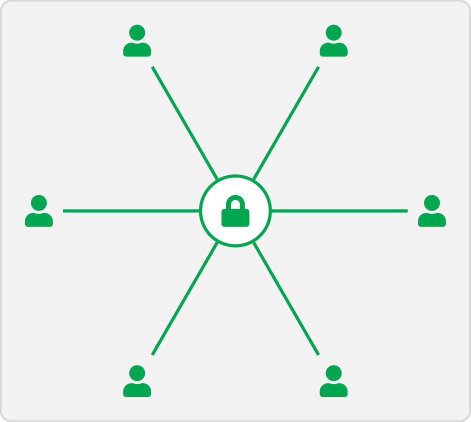
Lazy downloading:

- Users only download what they need, i.e. user *j* only need the *j*-th ciphertext
- How do we keep compatibility with the signatures?
  - One signature per ciphertext → costly
  - Merkle tree → better but same asymptotic cost as TreeKEM
  - We sign the **epoch's confirmation tag** (derived from 🔒 and the public view)
    - Idea implicit in [HKP$^+$21, Footnote 5], explicit in [AHKM22]
    - [HKP$^+$21] also used committing mPKE, but this is not necessary

Our proposed protocol

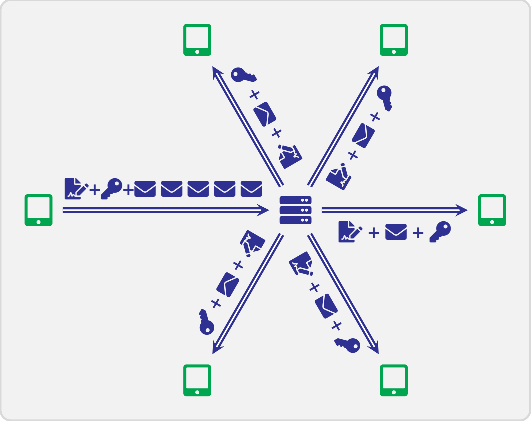
- **One channel:** a single shared secret for the whole group
  - Sending application messages is cheap
- **One signature:**
  - A single signature authenticates the encryption key & all ciphertexts
  - Compatible with lazy downloading

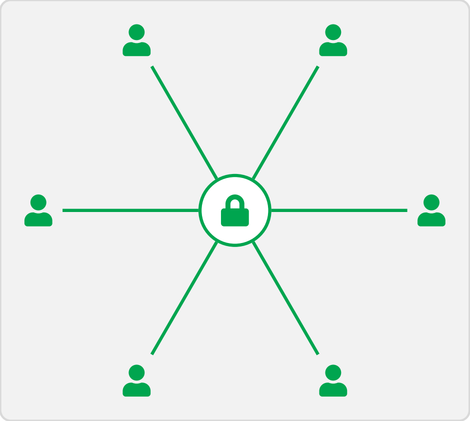




Our proposed protocol

- **One channel:** a single shared secret for the whole group
  - Sending application messages is cheap
- **One signature:**
  - A single signature authenticates the encryption key & all ciphertexts
  - Compatible with lazy downloading





Optimization: multi-recipient KEMs

Main idea: with lattice-based encryption:

{encrypt **1** message to **N** parties} $\ll$ {encrypt **N** messages to **N** parties}

Optimization: multi-recipient KEMs

Main idea: with lattice-based encryption:

{encrypt **1** message to **N** parties} $\ll$ {encrypt **N** messages to **N** parties}

Example:

- 1 Kyber ciphertext:

Optimization: multi-recipient KEMs

PQ SHIELD

Main idea: with lattice-based encryption:

{encrypt **1** message to **N** parties} ≪ {encrypt **N** messages to **N** parties}

Example:

- 1 Kyber ciphertext:

- N Kyber ciphertexts:

Optimization: multi-recipient KEMs

Main idea: with lattice-based encryption:

$$\{\text{encrypt } \mathbf{1} \text{ message to } \mathbf{N} \text{ parties}\} \ll \{\text{encrypt } \mathbf{N} \text{ messages to } \mathbf{N} \text{ parties}\}$$

Example:

- 🙂 1 Kyber ciphertext:

- 😱 N Kyber ciphertexts:

- 🙂 1 "multi-recipient" Kyber ciphertext for N parties:

Optimization: multi-recipient KEMs

Main idea: with lattice-based encryption:

{encrypt **1** message to **N** parties} $\ll$ {encrypt **N** messages to **N** parties}

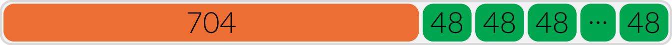
Example:

- 1 Kyber ciphertext:

- N Kyber ciphertexts:

- 1 "multi-recipient" Kyber ciphertext for N parties:

- 1 Ilum/mKyber [HKP+21] ciphertext for N parties:



Optimization: multi-recipient KEMs

Main idea: with lattice-based encryption:

{encrypt **1** message to **N** parties} $\ll$ {encrypt **N** messages to **N** parties}

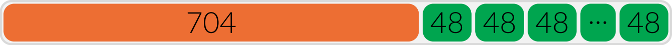
Example:

1 Kyber ciphertext:

N Kyber ciphertexts:

1 "multi-recipient" Kyber ciphertext for N parties:

1 Ilum/mKyber [HKP+21] ciphertext for N parties:



More details at the Fifth NIST PQC conference (April 10-12, 2024, Rockville, USA)!

Bandwidth costs for a group of size N

Scheme	**Application message**	Update (upload)	Update (download)	**Update (total)**
Pairwise channels	**O(N)**	O(N)	O(1)	**O(N)**
TreeKEM (MLS)	**O(1)**	O(log N)*	O(log N)*	**O(N log N)***
Our protocol	**O(1)**	O(N)†	O(1)	**O(N)**

* Best-case complexity

† With multi-recipient KEMs, we gain an order of magnitude in the *O*( ) constant.

Further reading

Full paper:

- Hashimoto, Katsumata, Postlethwaite, Prest and Westerbaan: *A Concrete Treatment of Efficient Continuous Group Key Agreement via Multi-Recipient PKEs* [HKP+21]

- See also:
  - Kwiatkowski, Katsumata, Pintore and Prest: *Scalable Ciphertext Compression Techniques for Post-Quantum KEMs and their Applications* [KKPP20]
  - Alwen, Hartmann, Kiltz and Mularczyk: *Server-Aided Continuous Group Key Agreement* [AHKM22]

Note: we are hiring post-docs on secure messaging!

Questions?

Joël Alwen, Sandro Coretti, and Yevgeniy Dodis.
The double ratchet: Security notions, proofs, and modularization for the Signal protocol.
In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019*, *Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019.

Joël Alwen, Dominik Hartmann, Eike Kiltz, and Marta Mularczyk.
Server-aided continuous group key agreement.
In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 69–82. ACM Press, November 2022.

Karthikeyan Bhargavan, Benjamin Beurdouche, and Prasad Naldurg.
Formal Models and Verified Protocols for Group Messaging: Attacks and Proofs for IETF MLS.
Research report, Inria Paris, December 2019.

Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila.
Post-quantum asynchronous deniable key exchange and the Signal handshake.
In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022*, *Part II*, volume 13178 of *LNCS*, pages 3–34. Springer, Heidelberg, March 2022.

Cable.co.uk.
Worldwide Mobile Data Pricing 2023 | 1GB Cost in 230 Countries, 2023.
https://www.cable.co.uk/mobiles/worldwide-data-pricing/.

Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest.

An efficient and generic construction for Signal's handshake (X3DH): Post-quantum, state leakage secure, and deniable.
In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 410–440. Springer, Heidelberg, May 2021.

Keitaro Hashimoto, Shuichi Katsumata, Eamonn Postlethwaite, Thomas Prest, and Bas Westerbaan.
A concrete treatment of efficient continuous group key agreement via multi-recipient PKEs.
In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1441–1462. ACM Press, November 2021.

Shuichi Katsumata, Kris Kwiatkowski, Federico Pintore, and Thomas Prest.
Scalable ciphertext compression techniques for post-quantum KEMs and their applications.
In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 289–320. Springer, Heidelberg, December 2020.

Speedtest.
Speedtest global index – internet speed around the world, July 2023.
https://www.speedtest.net/global-index.