Post-Quantum Secure Messaging

Thomas Prest

PQ SHIELD

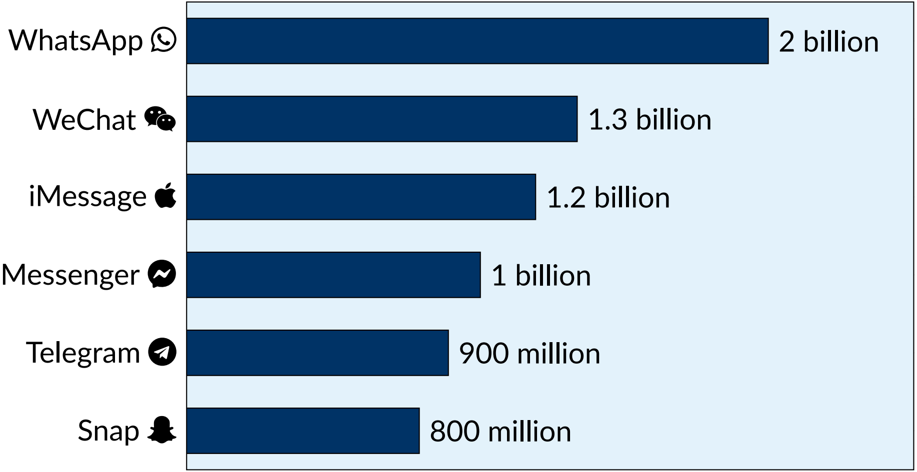
Graz Security Week 2024

Why secure messaging?

- Everyone uses it
- Many people try to break it
- Fun research topic!

Some popular messaging apps (2024)[1]

PQ SHIELD



- **This talk:** protocol security
- **First question:** which attack model should we consider?

[1] **Source:** Statista. Sources are often private, difficult to check, and exclude certain applications.

Threat Model

Compromising servers via subpoenas (1/2)

PQ SHIELD

RollingStone

GOT A TIP? LOG IN SUBSCRIBE

POLITICS

FBI Document Says the Feds Can Get Your WhatsApp Data – in Real Time

A previously unreported FBI document obtained by *Rolling Stone* reveals that "private" messaging apps WhatsApp and iMessage are deeply vulnerable to law-enforcement searches

BY ANDY KROLL

NOVEMBER 29, 2021

See also the FBI infographic[2] reproduced in the next slide.

[2] Link: https://propertyofthepeople.org/document-detail/?doc-id=21114562

(U//FOUO) FBI's Ability to Legally Access Secure Messaging App Content and Metadata

(U//LES) As of November 2020, the FBI's ability to legally access secure content on leading messaging applications is depicted below, including accessible information based on the applicable legal process. Return data provided by the companies listed below, with the exception of WhatsApp, are actually logs of latent data that are provided to law enforcement in a non-real-time manner and may impact investigation due to delivery delays.

UNCLASSIFIED // LAW ENFORCEMENT SENSITIVE

App	Information accessed	Legal Process & Additional Details
iMessage	Subscriber data; Message Sender Receiver Data; Device backup; Encryption key(s); Data/time information; Registration time data; User's contacts	• **Message Content: Limited** • **Subpoena:** can render basic subscriber information • **18 U.S.C. §2709(d):** can render 25 days of iMessage lookups to and from a target number[1] • **Pen Register:** no capability[1] • **Search Warrant:** can render backups of a target device; if target uses iCloud backup, the encryption keys should also be provided with content return; can also acquire iMessages from iCloud returns if target has enabled Messages in iCloud
Line	Subscriber data; Message Sender Receiver Data; Data/time information; Registration time data; User's contacts	• **Message Content: Limited*** • Suspect and/or victim's registered information (profile image, display name, email address, phone number, LINE ID, date of registration, etc.) • Information on usage * Maximum of seven days' worth of specified users' text chats (Only when E2EE has not been elected and applied and only when receiving an effective warrant; however video, picture, files, location, phone call audio and other such data will not be disclosed)
Signal	Data/time information; Registration time data	• **No Message Content** • Date and time a user registered • Last date of a user's connectivity to the service
Telegram	Registration time data	• **No Message Content** • No contact information provided for law enforcement to pursue a court order. As per Telegran's privacy statement, for confirmed terrorist invectigations, Telegram now discloses IP address and phone number to relevant authorities
Threema	Subscriber data; Encryption key(s); Data/time information; Registration time data	• **No Message Content** • Hash of phone number and email address, if provided by user • Push/Token, if push service is used • Public Key • Date (no time) of Threema ID creation • Date (no time) of last login
Viber	Subscriber data; IP address; Encryption key(s); Data/time information; Registration time data	• **No Message Content** • Provides account (i.e. phone number) registration data and IP address at time of creation • Message History: time, date, source number and destination number
WeChat	Subscriber data; IP address; Registration time data	• **No Message Content** • Accepts preservation letters and subpoenas, but cannot provide records for accounts created in China • For non-China accounts, they can provide basic information (name, phone number, email, IP address) which is retained for as long as the account is active
WhatsApp	Subscriber data; Message Sender Receiver Data; Data/time information; Registration time data; User's contacts	• **Message Content: Limited*** • Subpoena: can render basic subscriber records • Court Order: Subpoena return as well as information like blocked users • Search Warrant: Provides address book contacts and WhatsApp users who have the target in their address book contacts • Pen Register: Sent every 15 minutes, provides source and destination for each message * If target is using an iPhone and iCloud backups enabled, iCloud returns may contain WhatsApp data, to include message content
Wickr	Subscriber data; Data/time information; Registration time data	• **No Message Content** • Date and time account created • Type of decide(s) add installed in • Date of last use • Total number of messages • Number of external IDs (email addresses and phone numbers) connected to the account, but not plaintext external IDs themselves • Avatar image • Limited records of recent changes to account setting such as adding or suspending a device (does not include message content or routing and delivery information) • Wickr Version Number

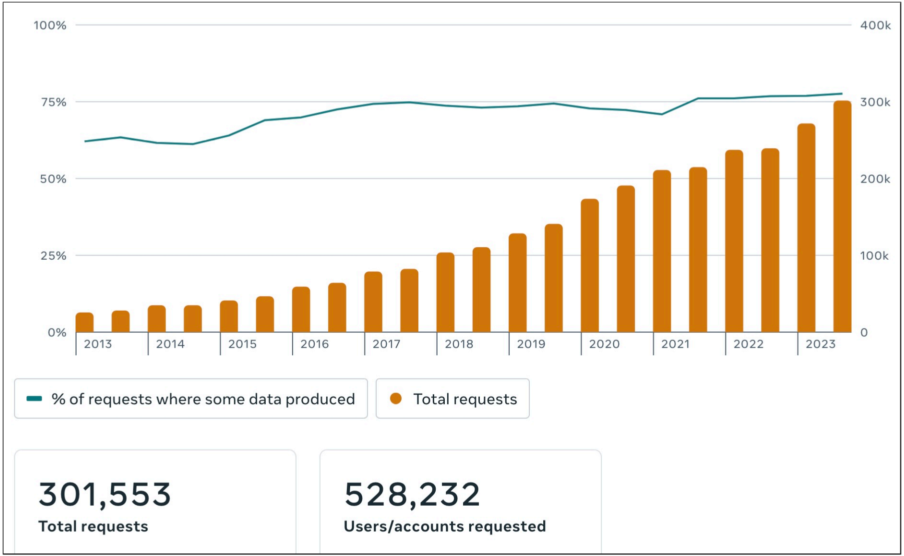
SUBSCRIBER DATA | MESSAGE SENDER RECEIVER DATA | DEVICE BACKUP | IP ADDRESS | ENCRYPTION KEY(S) | DATA/TIME INFORMATION | REGISTRATION TIME DATA | USER'S CONTACTS

(U) Prepared by Science and Technology Branch and Operational Technology Division 7 January 2021

[1](U//LES) Apple provided logs only identify if a lookup occurred. Apple returns include a disclaimer that a log entry between parties does not indicate a conversation took place. These query logs have also contained errors.

Compromising servers via subpoeanas (2/2)

Meta’s transparency report up to December 2023



Meta’s transparency center: `https://transparency.meta.com/`

Compromising servers via legal *pressure*

PQ SHIELD

How it started (source: Le Monde):

Telegram CEO Pavel Durov arrested in France in world-first case

The founder of the messaging service was arrested on Saturday evening at Le Bourget airport outside of Paris. He is the subject of an investigation for the lack of moderation on his platform.

How it's going (source: Le Monde): *"After the arrest of Pavel Durov, Telegram's surge of cooperation with the justice system in France and Belgium"*

Après l'arrestation de Pavel Durov, le sursaut de coopération de Telegram avec la justice de France et de Belgique

La justice des deux pays a confirmé que l'entreprise sise aux Emirats arabes unis, généralement muette face aux réquisitions judiciaires, avait changé de pied depuis l'arrestation, le 24 août, en France, de son cofondateur Pavel Durov.

Compromising users via legal means

USA: Searching electronic devices at ports of entry without a warrant is legal:

- See "Border Search of Electronic Devices at Ports of Entry"[3]
- Legality is contested, see "United States v. Sultanov" ruling (July 2024)

Russia: "*Yarovaya law*":

- Requires phone operators to store SMS, calls and internet traffic for 6 months
- Feasibility and status of deployment is unclear

Europe: Routinely proposes to backdoor end-to-end encryption or undermine it ("client-side scanning")

- See "*Proposal for a regulation of the european parliament and of the council laying down rules to prevent and combat child sexual abuse*"[4]

[3] https://www.cbp.gov/travel/cbp-search-authority/border-search-electronic-devices

[4] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2022:209:FIN

Compromising users via technical means

Spyware sold off-the-shelf by companies and hackers

PEGASUS: THE NEW GLOBAL WEAPON FOR SILENCING JOURNALISTS

At least 180 journalists around the world have been selected as targets by clients of the cybersurveillance company NSO Group, according to a new Forbidden Stories investigation, published today.

Takeaways and next steps

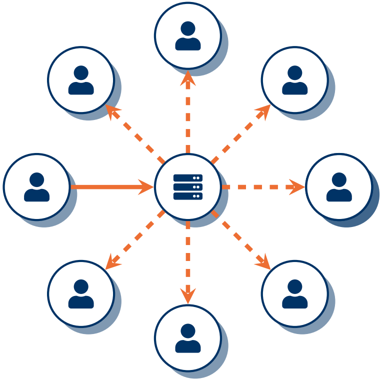
What we observe

- Asynchrony

PQ SHIELD

What we observe

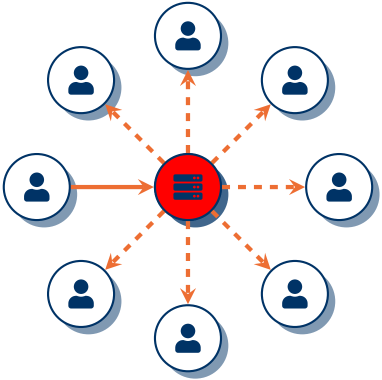
- Asynchrony
- Conversations can have many users (dozens or more)



Takeaways and next steps

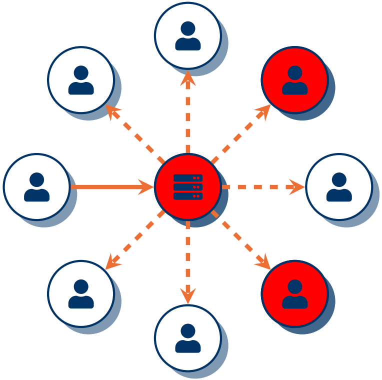
What we observe

- Asynchrony
- Conversations can have many users (dozens or more)
- The server should not be trusted ($\Longrightarrow$ end-to-end encryption)



What we observe

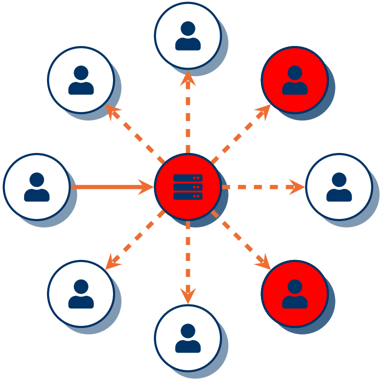
- Asynchrony
- Conversations can have many users (dozens or more)
- The server should not be trusted ($\Longrightarrow$ end-to-end encryption)
- Users can be compromised



Takeaways and next steps

What we observe

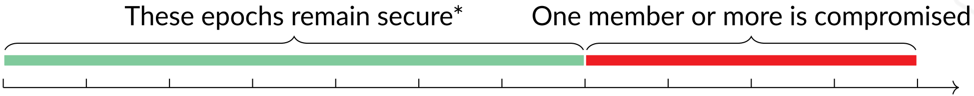
- Asynchrony
- Conversations can have many users (dozens or more)
- The server should not be trusted ($\Longrightarrow$ end-to-end encryption)
- Users can be compromised
- Very long sessions (years) ($\Longrightarrow$ next slide)



Some security notions

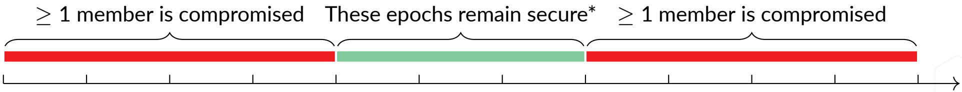
PQ SHIELD

Forward secrecy (FS) [CCG16, CCD^{+}17, ACD19]:



Post-Compromise Security (PCS) [CCG16, CCD^{+}17, ACD19]:

Post-Compromise Forward Security (PCFS) [ACDT20, ACJM20, AJM20]:



How do we obtain a secure messaging protocol that is simultaneously...

$$\underbrace{\text{post-quantum}}_{\text{Part I}} + \underbrace{\text{scalable}}_{\text{Part II}} + \underbrace{\text{metadata-hiding}}_{\text{Part III}}\ ?$$

Post-Quantum Security

Achieving post-quantum security

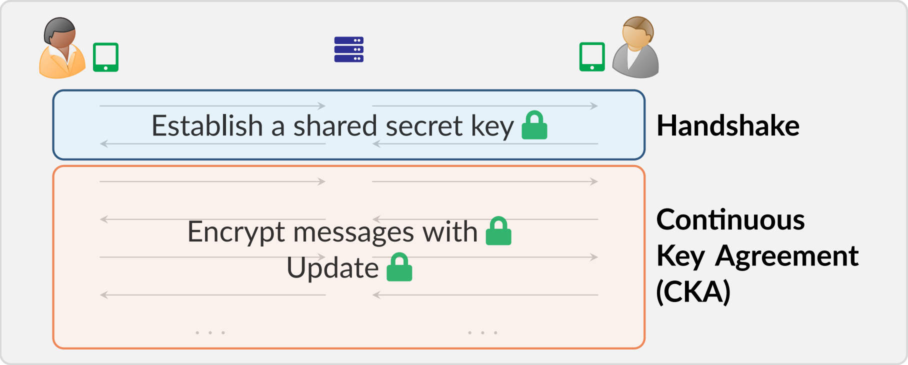
There are two approaches in building a post-quantum protocol:

- **Black-box:** provide a generic construction assuming secure building blocks
  - Symmetric crypto (hash functions, AEAD, etc.)
  - Key encapsulation mechanisms (KEMs)
  - Signatures
  - etc.
- **White-box:** open and optimize the underlying primitives

In my experience, the best protocols take advantage of both approaches.

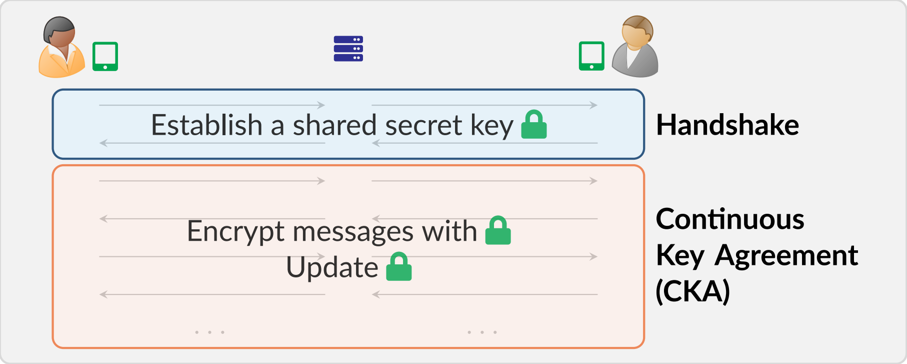
Secure messaging, high-level view (2 users)

PQ SHIELD



Post-quantum instantiations:

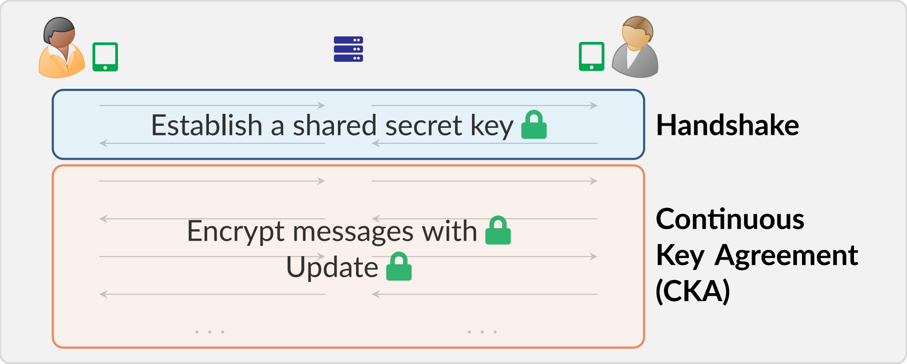
- **Handshake:** KEM + (ring) signatures + symmetric crypto [HKKP21, BFG+22]
- **Continuous Key Agreement (CKA):** KEM + symmetric crypto [ACD19]



CKA: sending application messages

Assume both parties share a secret symmetric key

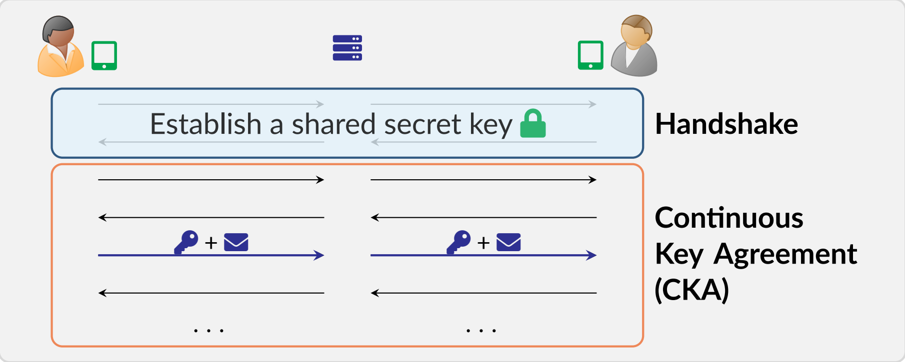
- → Application messages may be sent using an AEAD
- → More advanced functionalities (abuse reporting aka *message franking*) may require more specific properties (context committing [DGRW18])



CKA: achieving forward secrecy (“symmetric ratchet”)

A compromised 🔒 shall not allow to recover prior messages

→ After each message, 🔒 is locally updated by feeding it into a PRF



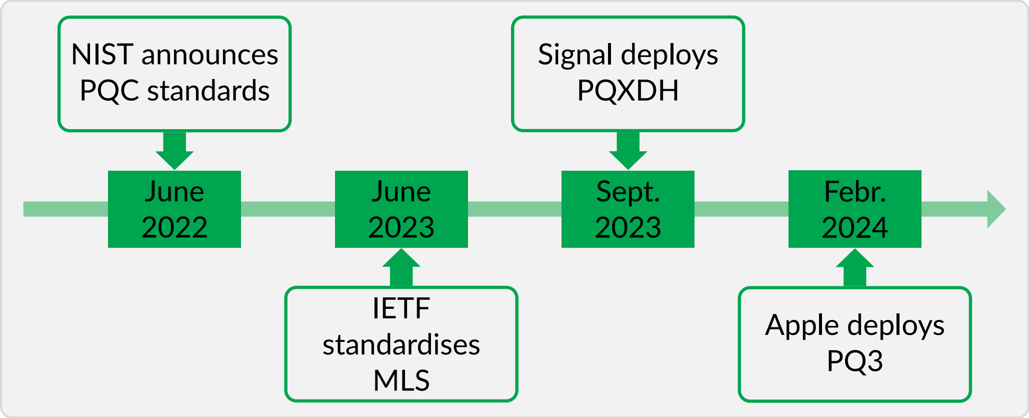
CKA: achieving post-compromise security (“asymmetric ratchet”)

A compromised shall not allow to recover future messages

- 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

Is it deployed? Yes!



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

Scalability

This section: bandwidth efficiency

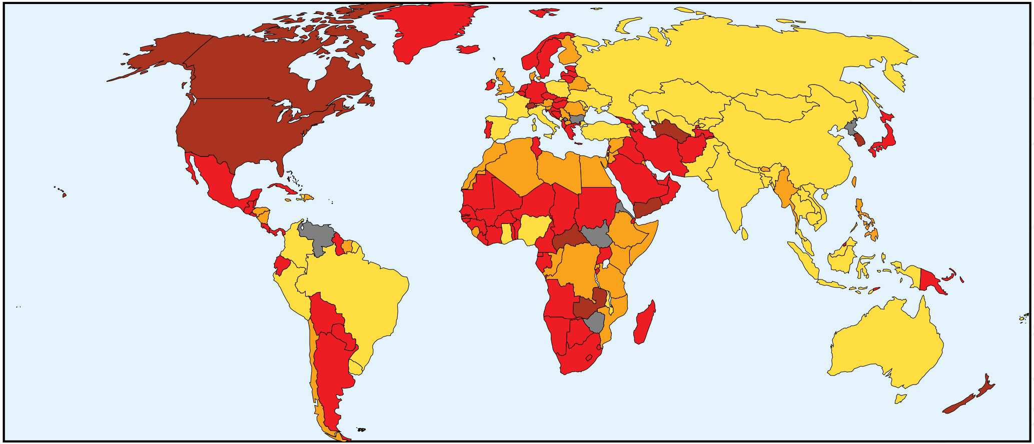
1. **Bandwidth** likely to be a bottleneck of PQ messaging, due to three factors:
   1. Mobile data plans
   2. Post-quantum primitives
   3. Continuous group key agreement (CGKA) protocols
2. Existing CGKAs can incur high bandwidth consumption
   - The bottleneck is in the public-key cryptography
3.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

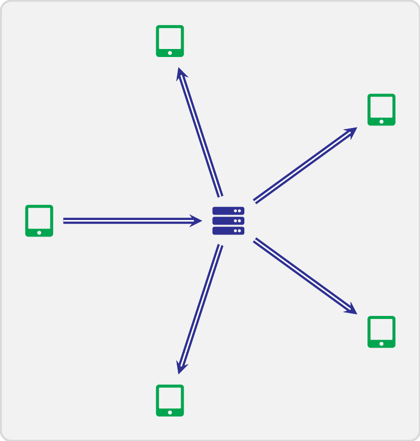
  Complete data on worldwide mobile speed:
  `https://www.speedtest.net/global-index`
- **Large groups** require more frequent key updates
  - Over 1 day, suppose each user gets compromised with probability ε.
    Over T days, a group with N users remains uncompromised with probability

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

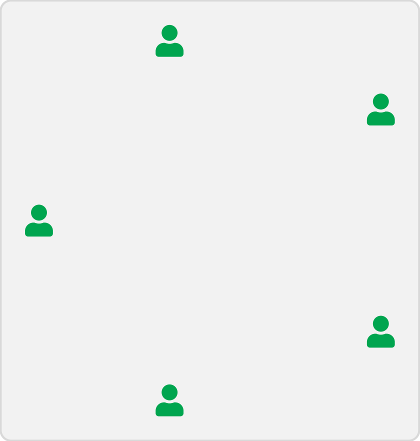
  - But existing CGKA may require high bandwidth (next slides)

Naive CGKA – pairwise channels

Physical layer

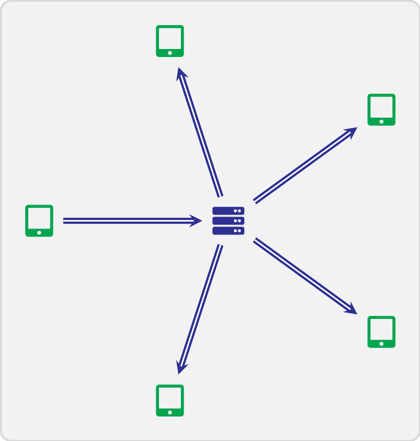


Insider view

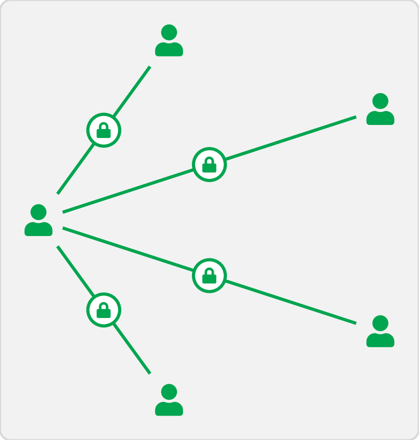


Naive CGKA – pairwise channels

Physical layer

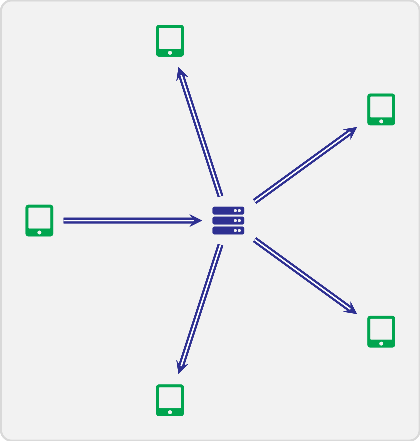


Insider view

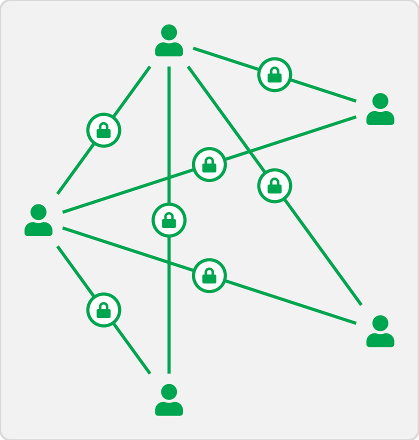


Naive CGKA – pairwise channels

Physical layer



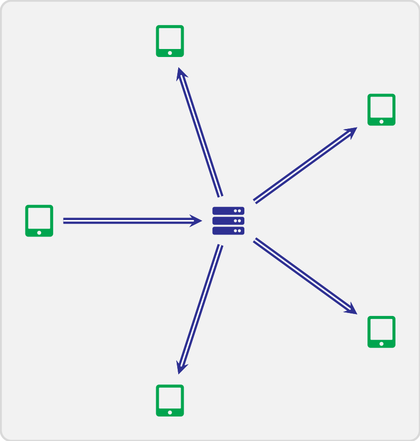
Insider view



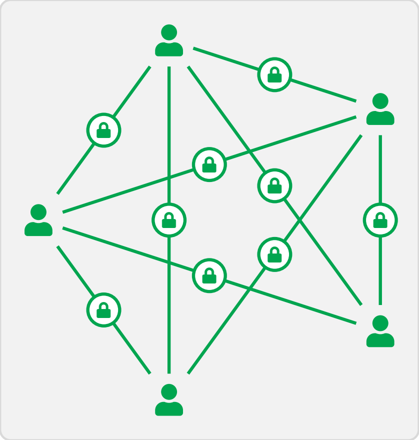
Naive CGKA – pairwise channels

PQ SHIELD

Physical layer

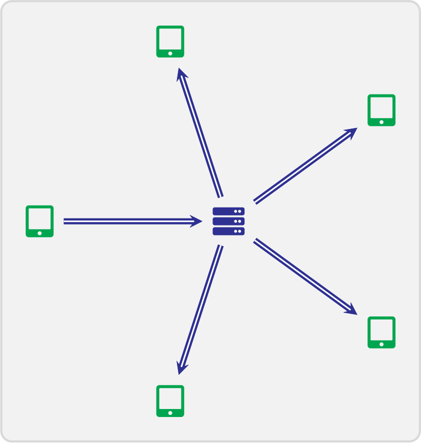


Insider view

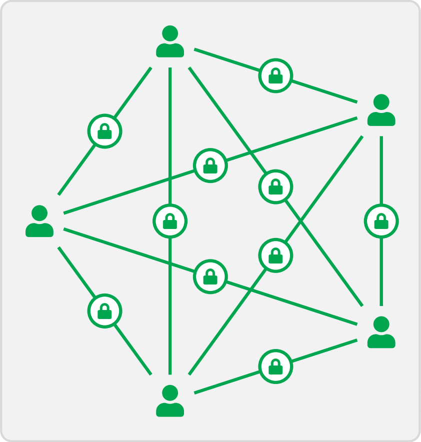


Naive CGKA – pairwise channels

Physical layer



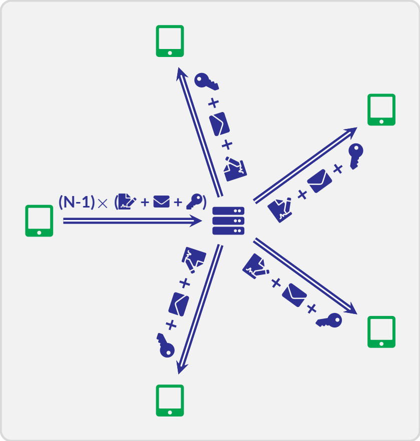
Insider view



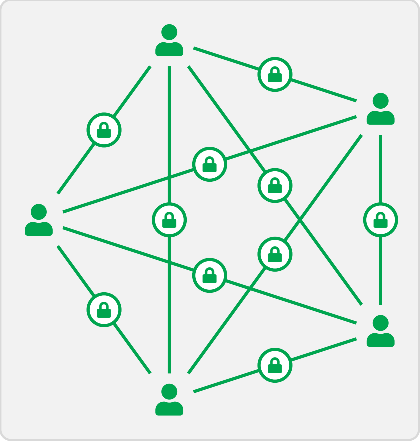
Naive CGKA – pairwise channels

PQ SHIELD

Physical layer



Insider view



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

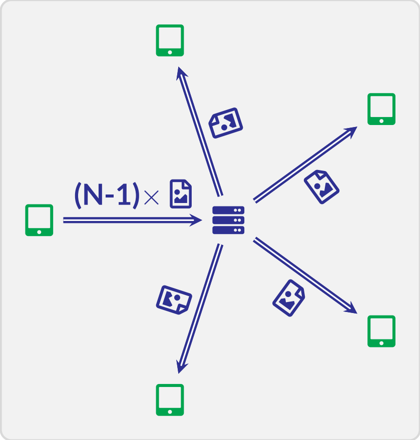
1 MB for the sender, 4 kB for each downloader

(= encryption key, = ciphertext, = signature)

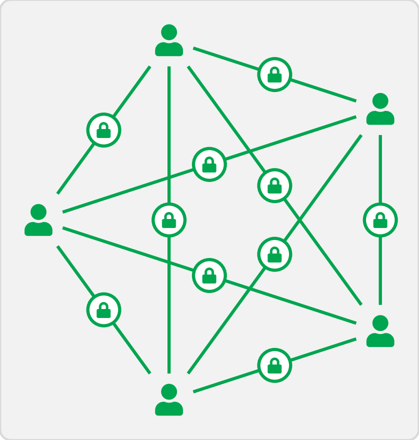
Naive CGKA – pairwise channels

PQ SHIELD

Physical layer

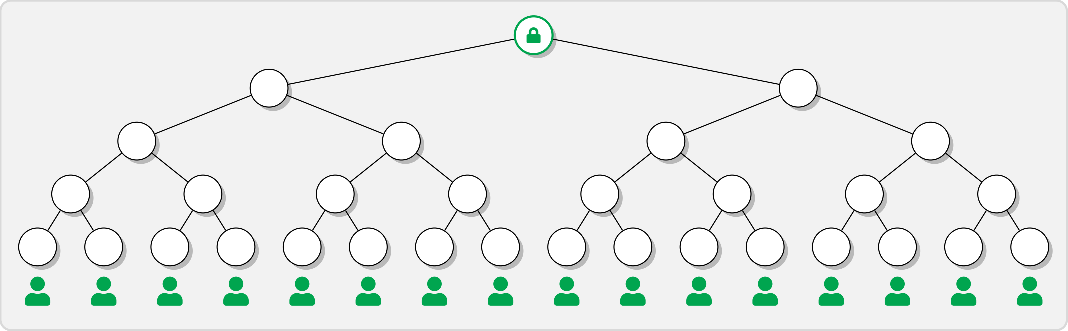


Insider view



Sending a single picture () of 100 Kilobytes with $N = 256$:
25.5 Megabytes for the sender, 100 kB for each downloader

MLS' CGKA – TreeKEM



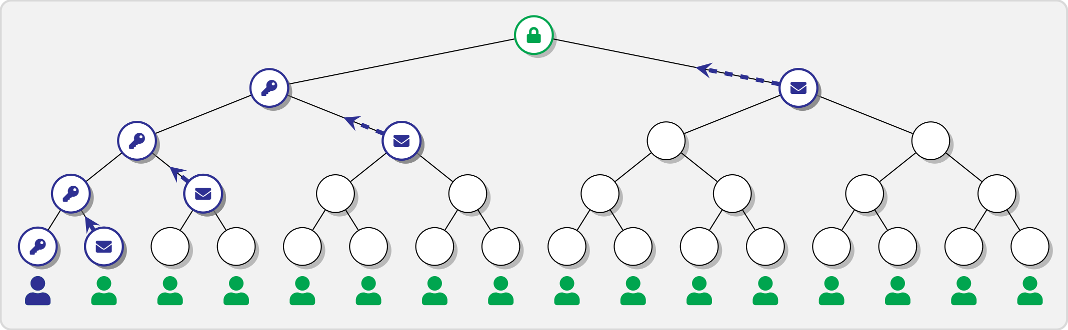
The N users are arranged as the leaves of a (binary) tree

Tree invariants:

1. All users know the public keys of all nodes in the tree
2. (*user* knows the private key of *node*) ⇔ (*node* is in the path of *user*)

Application messages: All users use the root private key

MLS' CGKA – TreeKEM

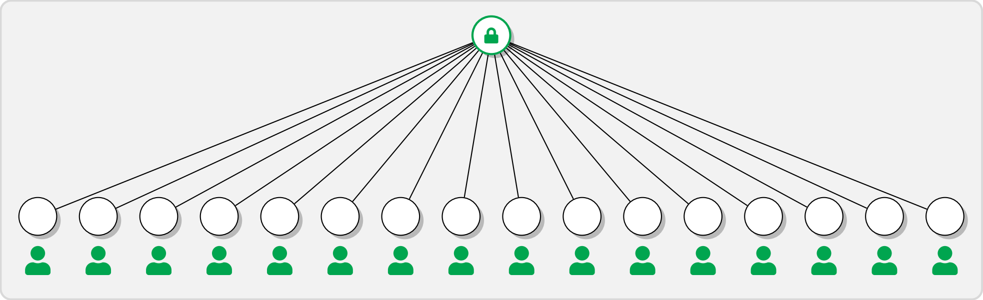


The *N* users are arranged as the leaves of a (binary) tree

When a user (here 👤) updates their key, they broadcast:

- log *N* encryption keys (🔑)
- log *N* ciphertexts (✉)
  - Each ciphertext encrypts to its sibling node the private key of their parent node
- 2 signatures (✍) – one for encryption keys, one for ciphertexts

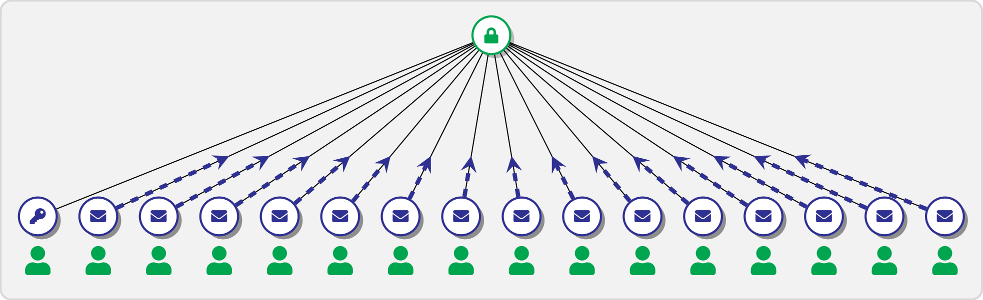
What if we use a flat tree?



This is essentially Chained mKEM [BBN19]

The tree invariant remains identical (and simpler)

What if we use a flat tree?

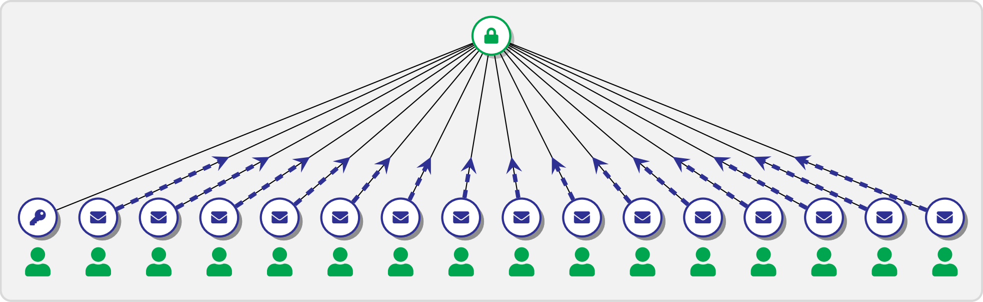


This is essentially Chained mKEM [BBN19]

- 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?

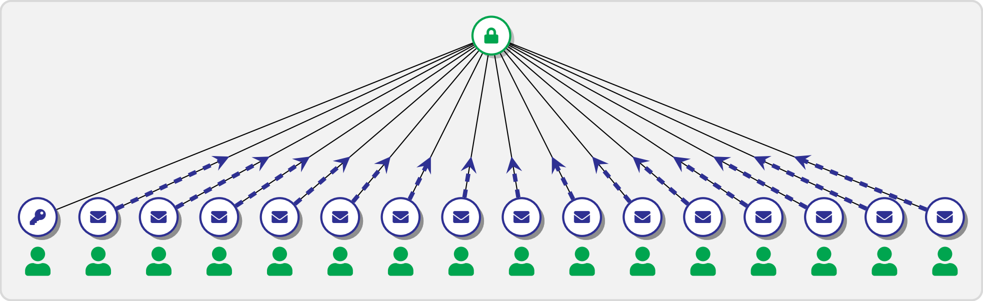
Flat tree + lazy download



Lazy downloading:

- Users only download what they need, i.e. user *j* only need the *j*-th ciphertext
- How do we keep signatures consistent with only partial information?
- Imperfect solutions
  - One signature per ciphertext → costly
  - Merkle tree → better but same asymptotic cost as TreeKEM

Flat tree + lazy download + implicit consistency:

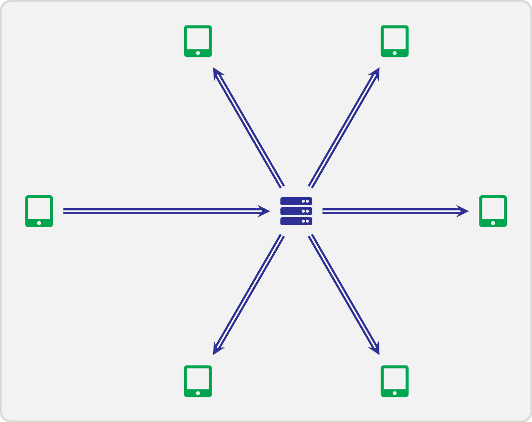


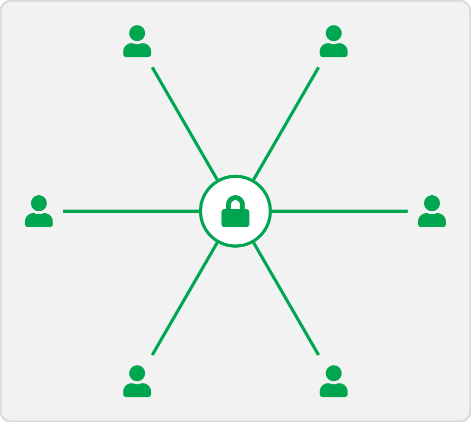
Lazy downloading:

- Users only download what they need, i.e. user *j* only need the *j*-th ciphertext
- How do we keep signatures consistent with only partial information?
- **Solution:** sign the **epoch's confirmation tag** (derived from 🔒 and 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

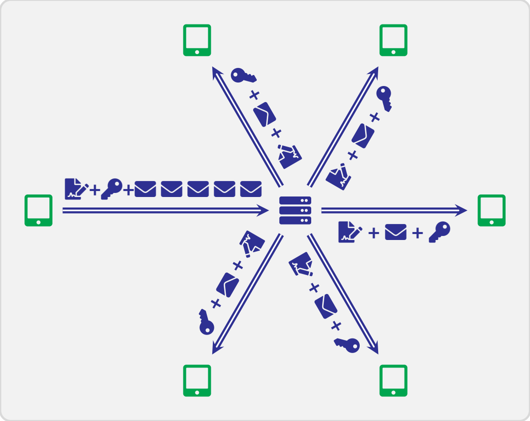


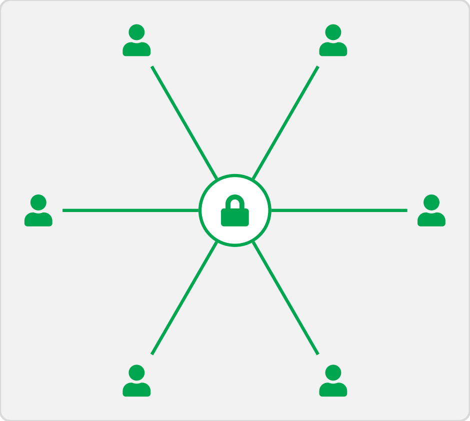


Our proposed protocol

PQ SHIELD

- **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:

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

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:

640	128

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:

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:

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:

  640 | 128

- N Kyber ciphertexts:

  640 | 128 | ··· | 640 | 128

- 1 “multi-recipient” Kyber ciphertext for N parties:

  640 | 128 | 128 | 128 | ··· | 128

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

  704 | 48 | 48 | 48 | ··· | 48

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 a factor **16** in the O( ) constant.

Metadata Protection

Metadata collection is systemic

PQ SHIELD

“ Metadata, however, showing how a WhatsApp account was used and which numbers were contacting one another and when, can be tracked with a surveillance technology known as a pen-register. PenLink provides that tool as a service. ”

Forbes

CYBERSECURITY • EDITORS' PICK

Meet The Secretive Surveillance Wizards Helping The FBI And ICE Wiretap Facebook And Google Users

Thomas Brewster Forbes Staff

Associate editor at Forbes, covering cybercrime, privacy, security and surveillance.

Follow

Feb 23, 2022, 01:53pm EST

2

A small Nebraska company is helping law enforcement around the world spy on users of Google, Facebook and other tech giants. A secretly recorded presentation to police reveals how deeply embedded in the U.S. surveillance machine PenLink has become.

Metadata collection, in more details

RollingStone

GOT A TIP? | LOG IN | SUBSCRIBE

POLITICS

FBI Document Says the Feds Can Get Your WhatsApp Data – in Real Time

A previously unreported FBI document obtained by *Rolling Stone* reveals that "private" messaging apps WhatsApp and iMessage are deeply vulnerable to law-enforcement searches

BY ANDY KROLL

NOVEMBER 29, 2021

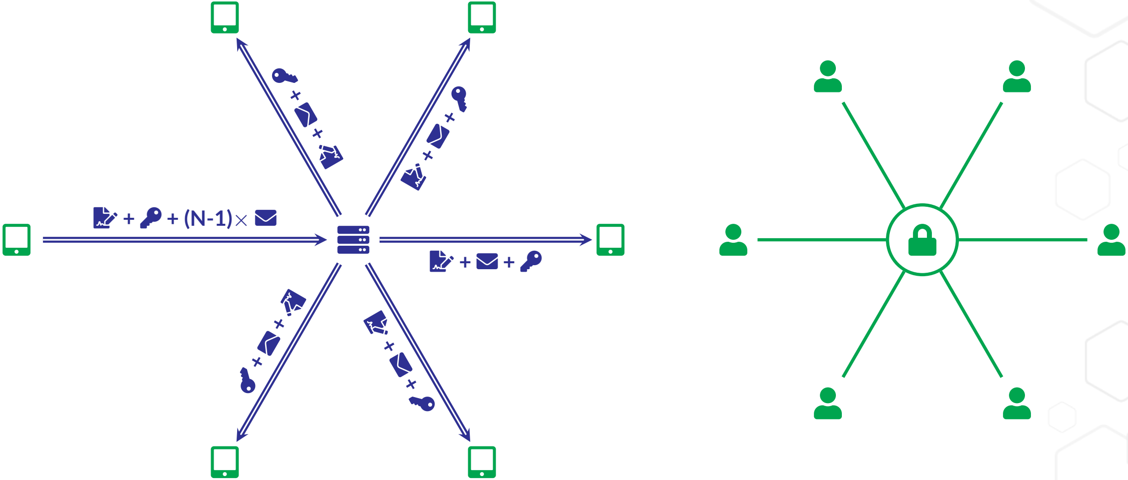
	iMessage	Line	Signal	Telegram	Threema	Viber	WeChat	WhatsApp
Subscriber data	☠	☠			☠	☠	☠	☠
Message sender, receiver data	☠	☠					☠	
IP address						☠	☠	
Date/time information	☠	☠	☠		☠	☠		☠
User contacts	☠	☠			☠	☠	☠	☠

Scope of this section

We want to hide that user X is sending information to $G \ni X$

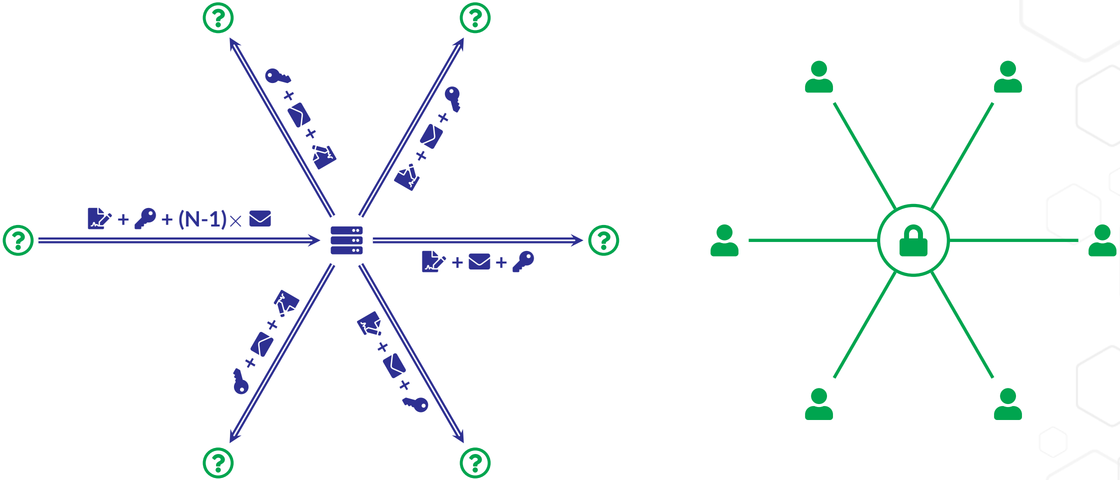
- **Assumption:** X shares a secure (user-side) anonymous connection with the server
- Solutions exist (Signal's Private Groups System) but they are not post-quantum
- **Outside the scope:** Server-side inference based on relations between groups

Initial protocol
(🔑 = encryption key, ✉ = ciphertext, 📝 = signature)



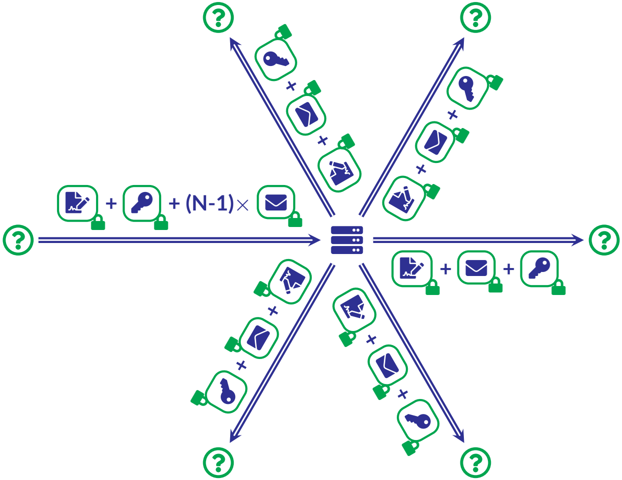
⚠ **Messages are confidential, but not metadata (📱)**

Use client-anonymous authenticated channels



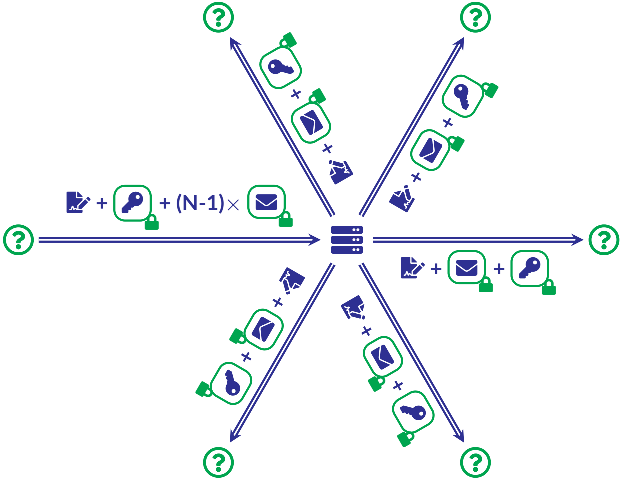
The packages , , leak the identities

OK, then let's also encrypt all the packages with



Now anyone can upload garbage messages to the group!

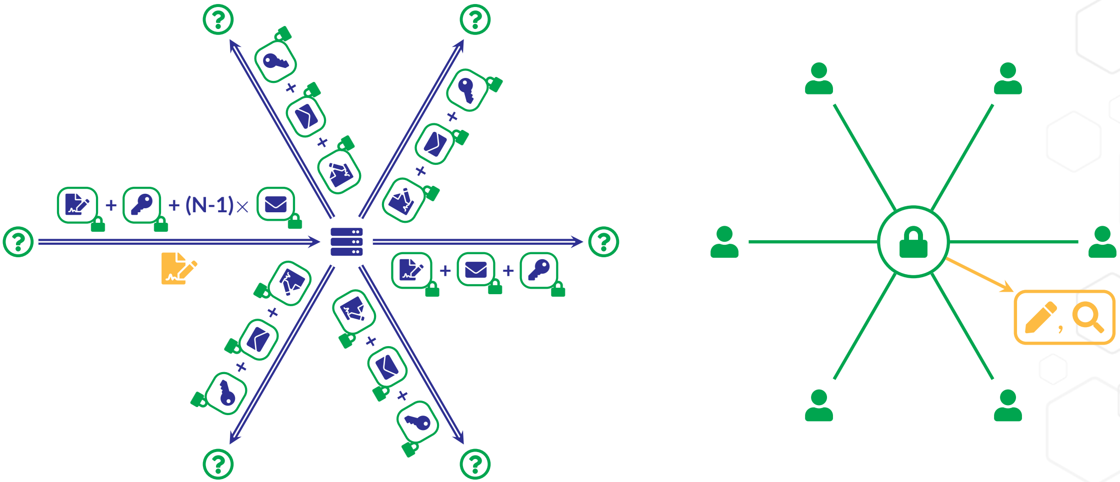
Let's encrypt all packages *except the signature*



But each signature is linked to its sender

Solution: derive a signature keypair (✏️, 🔍) from 🔒

- The verification key 🔍 is public, but only users 👤 know the signing key ✏️
- Group members can authenticate themselves anonymously



Further reading

Scalability:

- Kwiatkowski, Katsumata, Pintore, Prest: *Scalable Ciphertext Compression Techniques for Post-Quantum KEMs and their Applications.* ASIACRYPT 2020. [KKPP20]
- Hashimoto, Katsumata, Postlethwaite, Prest, Westerbaan: *A Concrete Treatment of Efficient Continuous Group Key Agreement via Multi-Recipient PKEs.* CCS 2021. [HKP+21]

Metadata protection:

- Hashimoto, Katsumata, Prest: *How to Hide MetaData in MLS-Like Secure Group Messaging: Simple, Modular, and Post-Quantum.* CCS 2022. [HKP22]

Other:

- The presentation is on my website: `https://tprest.github.io`
- White paper "*Secure Messaging in a Post-Quantum World*", written by Shu and me: `https://content.pqshield.com/secure-messaging-in-a-post-quantum-world`
- Please come say hi if you are interested in research projects!

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, Cham, May 2019.

Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis.
Security analysis and improvements for the IETF MLS standard for group messaging.
In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 248–277. Springer, Cham, August 2020.

Joël Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk.
Continuous group key agreement with active security.
In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 261–290. Springer, Cham, November 2020.

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.

Joël Alwen, Daniel Jost, and Marta Mularczyk.
On the insider security of MLS.
Cryptology ePrint Archive, Report 2020/1327, 2020.

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, Cham, 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/.

K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila.
A formal security analysis of the signal messaging protocol.
In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, 2017.

Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt.
On post-compromise security.
In Michael Hicks and Boris Köpf, editors, *CSF 2016 Computer Security Foundations Symposium*, pages 164–178. IEEE Computer Society Press, 2016.

Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage.
Fast message franking: From invisible salamanders to encryptment.
In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 155–186. Springer, Cham, August 2018.

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, Cham, 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.

Keitaro Hashimoto, Shuichi Katsumata, and Thomas Prest.
How to hide MetaData in MLS-like secure group messaging: Simple, modular, and post-quantum.
In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1399–1412. ACM Press, November 2022.

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, Cham, December 2020.