black hat EUROPE 2022

Practically-exploitable Cryptographic Vulnerabilities in Matrix

`https://nebuchadnezzar-megolm.github.io/`

Martin Albrecht martin.albrecht@kcl.ac.uk
Sofía Celi cherenkov@riseup.net
Benjamin Dowling b.dowling@sheffield.ac.uk
Dan Jones dan.jones.2020@live.rhul.ac.uk

Outline

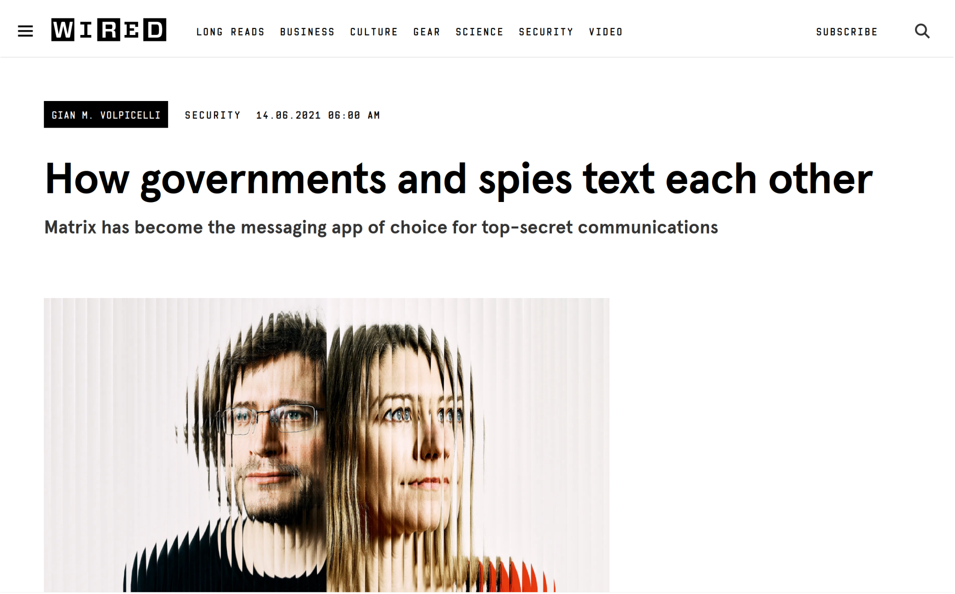
New phone, who dis?

Cryptography in Matrix

Attacks

Take Home Message

Matrix?

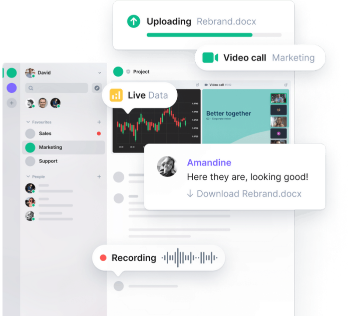


GIAN M. VOLPICELLI SECURITY 14.06.2021 06:00 AM

How governments and spies text each other

Matrix has become the messaging app of choice for top-secret communications

Matrix!

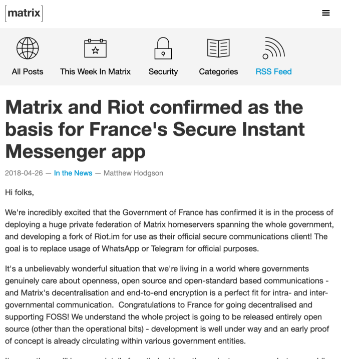


Amandine
Here they are, looking good!
↓ Download Rebrand.docx

- **Matrix** = standard for federated, decentralised, real-time group messaging
- **Element** = glossy flagship client
- End-to-end encryption is enabled by default
  - Threat model: servers are the adversary
  - Contrasts with Slack, MS Teams, Zulip, Mattermost, . . .

Matrix!

Element has over 60 million users. **Matrix**' users include



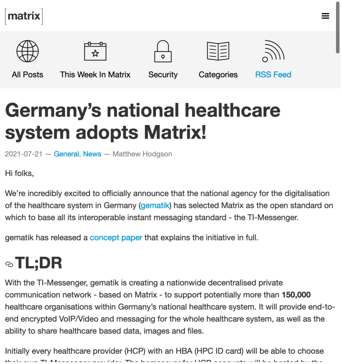
Matrix and Riot confirmed as the basis for France's Secure Instant Messenger app

2018-04-26 — In the News — Matthew Hodgson

Hi folks,

We're incredibly excited that the Government of France has confirmed it is in the process of deploying a huge private federation of Matrix homeservers spanning the whole government, and developing a fork of Riot.im for use as their official secure communications client! The goal is to replace usage of WhatsApp or Telegram for official purposes.

It's a unbelievably wonderful situation that we're living in a world where governments genuinely care about openness, open source and open-standard based communications - and Matrix's decentralisation and end-to-end encryption is a perfect fit for intra- and inter-governmental communication. Congratulations to France for going decentralised and supporting FOSS! We understand the whole project is going to be released entirely open source (other than the operational bits) - development is well under way and an early proof of concept is already circulating within various government entities.



Germany's national healthcare system adopts Matrix!

2021-07-21 — General, News — Matthew Hodgson

Hi folks,

We're incredibly excited to officially announce that the national agency for the digitalisation of the healthcare system in Germany (gematik) has selected Matrix as the open standard on which to base all its interoperable instant messaging standard - the TI-Messenger.

gematik has released a concept paper that explains the initiative in full.

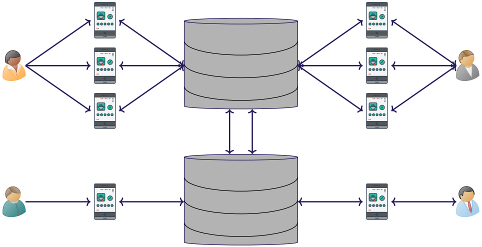
TL;DR

With the TI-Messenger, gematik is creating a nationwide decentralised private communication network - based on Matrix - to support potentially more than **150,000** healthcare organisations within Germany's national healthcare system. It will provide end-to-end encrypted VoIP/Video and messaging for the whole healthcare system, as well as the ability to share healthcare based data, images and files.

Initially every healthcare provider (HCP) with an HBA (HPC ID card) will be able to choose their own TI-Messenger provider. The homeserver for HCP accounts will be hosted by the

Architecture

- In **Matrix**, each User account can have many Devices.
- Each User has an account on a particular Homeserver.
- Homeservers maintain the link between a User account and its Devices.
- Messages are distributed by the Homeservers.
- A Room is a collection of Devices that communicate in a single conversation.



Outline

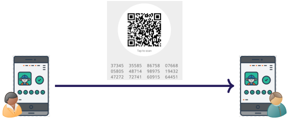
New phone, who dis?

Cryptography in Matrix

Attacks

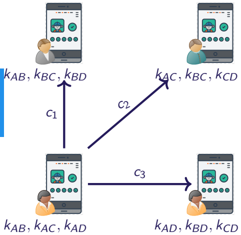
Take Home Message

Core Functionalities



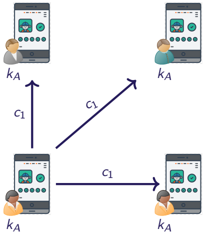
Device/Entity Authentication

(**Cross-Signing Framework**)



Session Establishment

(pairwise **Olm** channels)



Session Communication

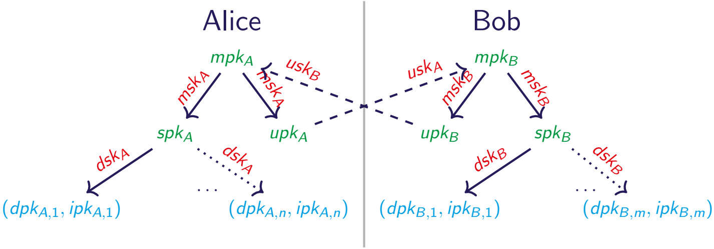
(group **Megolm** channels)

Entity Authentication via Cross-Signing Framework

Each User sets up an account with a particular Homeserver, which allocates a User identifier, A.

The User, then, generates their User Secrets, used to establish $\approx$ web-of-trust.

- The master key (mpk_A) serves as their long-term identity.
- The user-signing key (upk_A) signs other User's master keys.
- The self-signing key (spk_A) signs a User's own Device keys.



Device Authentication via Cross-Signing Framework

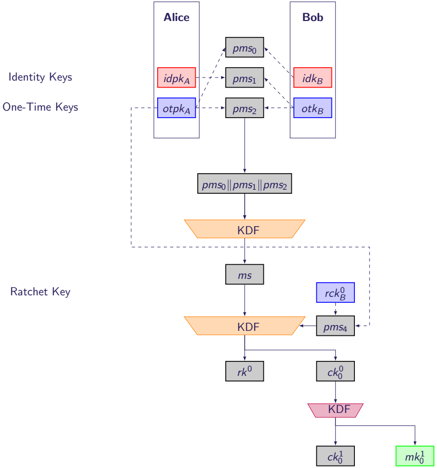
When a new Device logs in with account credentials, Homeserver allocates a device identifier $D^{A,i}$.

The Device then generates keys for this Device and registers it with the Homeserver:

1. Long-term Device Keys, authenticates **Olm Key Bundle**.
2. Olm Key Bundle, used to establish the pairwise channel, **Olm**.

Session Establishment via Olm

- Bob gets Alice's public key from Homeserver
- Bob does triple **D**iffie-**H**ellman (**3DH**) to produce a symmetric **m**aster **s**ecret.
- Bob uses **Double Ratchet** protocol to derive message keys.
- Bob encrypts **Megolm** Session State under these keys, and sends Session State to Alice.



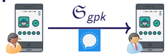
Megolm Session

Megolm Session State allows the Sender to encrypt messages to the **Megolm** channel (resp. a Receiver to decrypt).

$(gsk, gpk) \leftarrow \mathsf{SIG}.\mathbf{Gen}$

$R \xleftarrow{\$} \mathcal{R}$

$\mathfrak{S}_{gpk} \leftarrow (0, R, gpk)$



A **Megolm** session consists of the current *message index*, the *internal ratchet state*, and the *group signing key*.

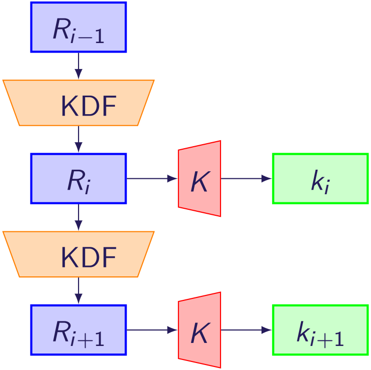
outbound session $\mathfrak{S}_{gsk} = (j, R, gsk)$ is kept in the device and used to encrypt messages for the room.

inbound session $\mathfrak{S}_{gpk} = (j, R, gpk)$ allows other devices in the room to authenticate and decrypt these messages.

Megolm Ratchet

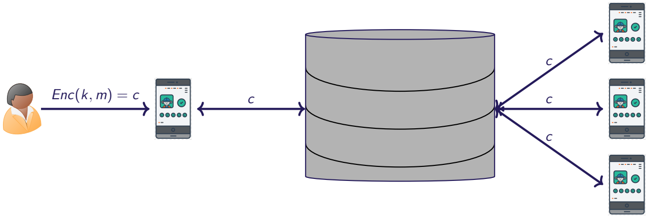
At its core, **Megolm** is a symmetric ratcheting scheme:

- it derives a new key for each message
- so that compromise of the current state cannot be used to recover previous encryption state



Megolm Encryption

1. Sender generates a fresh symmetric key from R,
2. encrypts the message under this key, and
3. signs it to provide authentication.

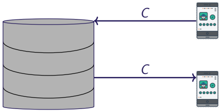


This ciphertext is distributed by the Homeserver to other devices in the Group.

“Pursue your dreams but have a backup plan”

Backup Functionalities:

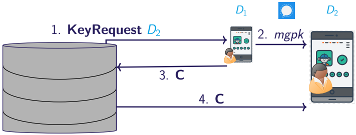
backup and recover User and Megolm secret values via Homeservers.



User Secret Backups

(Secure Secret Storage and Sharing (SSSS))

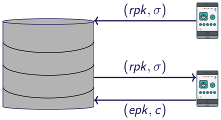
- backup master (cross-signing) secret keys to server



Online Session Recovery

(**KeyRequest** protocol)

- allows a user’s devices can share Megolm session information with each other



Offline Session Recovery

(**Server-Side Megolm Backups**)

- as a hybrid of both, backup Megolm sessions to server

Outline

New phone, who dis?

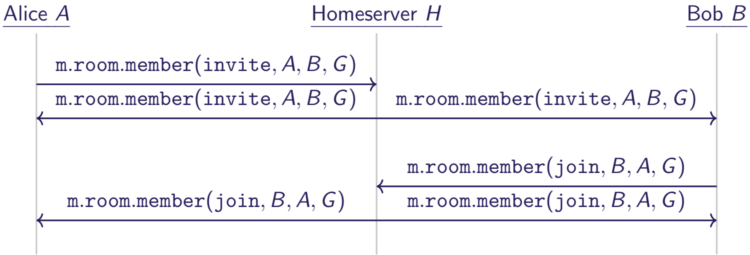
Cryptography in Matrix

Attacks

Take Home Message

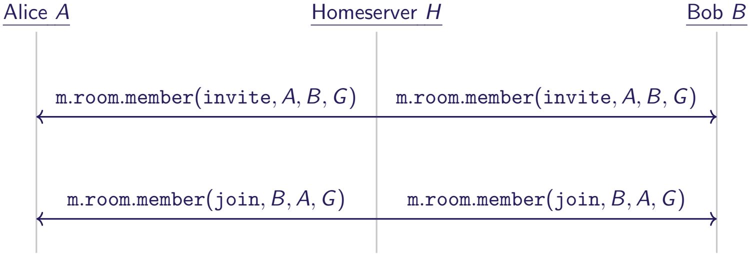
Q: “Who to encrypt to?”

Group membership is managed through events:


Alice A
Homeserver H
Bob B
m.room.member(invite, A, B, G)
m.room.member(invite, A, B, G)
m.room.member(invite, A, B, G)
m.room.member(join, B, A, G)
m.room.member(join, B, A, G)
m.room.member(join, B, A, G)

A: “Don’t worry, the server will let you know.”

Group membership is managed through unauthenticated events:



Alice A
Homeserver H
Bob B
m.room.member(invite, A, B, G)
m.room.member(invite, A, B, G)
m.room.member(join, B, A, G)
m.room.member(join, B, A, G)

Q: “What are Alice’s devices?” A: “Don’t worry . . . ”

- To send a message to a user, clients need a list of their devices.
- This list is provided by the homeserver and, hence, can be forged.

Damage

Breaks confidentiality: Attackers can eavesdrop on conversations

with some indication in (Element's) user interface.

Status

Neither of these two are fixed, but a remediation (signed group membership messages) is in the planning stage.

- Matrix' previous rational: Element client shows list of users for a room, so users can inspect, i.e. burden on users.
- Matrix post-disclosure: "many in the cryptography community consider this a serious misdesign. Eitherway, it's avoidable behaviour and we're ramping up work now to address it by signing room memberships so the clients control membership rather than the server."

Take Home Message

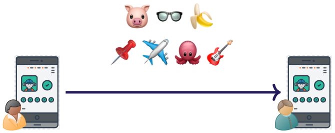
There is no confidentiality without authentication.

Attack on Out-of-Band Verification

How to ensure connection is not being MITM-ed? Out-of-band verification!

Short Authentication String (SAS) protocol ≈

1. Key exchange to generate a shared secret.
2. Compare the shared secret out-of-band
   (using short strings of emojis).
   If they don't match, then abort!
3. Send correct cryptographic identities to each other over a secure channel (constructed using the shared secret).



The homeserver tricks devices into sharing a homeserver-controlled identity.

Attack on Out-of-Band Verification

- Two types of verification:
  1. Between two users
  2. Between two devices of the same user
- Each party sends the other a message containing a "key identifier" field:
  1. For two users, this field contains the fingerprint of their master cross-signing key, *mpk*.
  2. For two devices, this field contains their device identifier.

Attack:

- Homeserver assigns the target a device identifier that is also a master cross-signing key fingerprint that the homeserver generated.
- When the target sends a verification request message with their device identifier, the receiving device interprets it as a cross-signing key fingerprint and signs it!

Damage

Breaks confidentiality: Attackers can eavesdrop on conversations

and authentication: Attackers can impersonate users

with no indication in (Element's) user interface!

Take Home Message

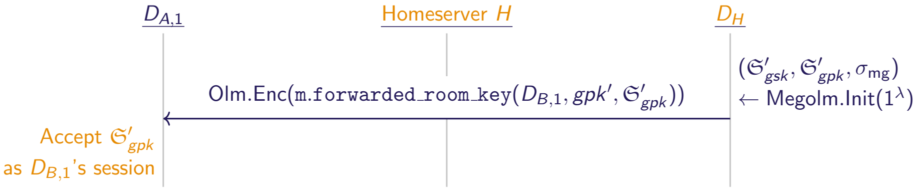
Domain separate all the things!

Alice: . . . , Bob: “Here are the keys for Charley”, Alice: “Ta!”

When a user adds a new device, they’d like that device to be able to decrypt messages previously sent to that user via the **KeyRequest** protocol.

Element and other clients limited who they sent secrets to but not who they accepted secrets from.

Attack:



$D_{A,1}$

Homeserver H

D_H

$(\mathfrak{S}'_{gsk}, \mathfrak{S}'_{gpk}, \sigma_{\text{mg}}) \leftarrow \text{Megolm.Init}(1^\lambda)$

$\text{Olm.Enc}(\texttt{m.forwarded\_room\_key}(D_{B,1}, gpk', \mathfrak{S}'_{gpk}))$

Accept $\mathfrak{S}'_{gpk}$ as $D_{B,1}$'s session

Damage

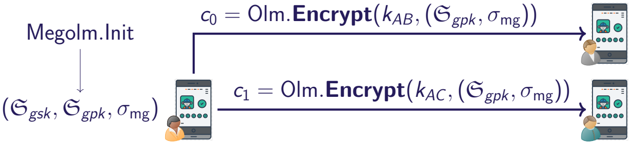
Semi-trusted Impersonation Attack

Breaks authentication: Attackers can impersonate users

with some indication in (Element's) user interface.

Layering Attacks for Full Impersonation

Megolm session setup:



Megolm.Init

$(\mathfrak{S}_{gsk}, \mathfrak{S}_{gpk}, \sigma_{\text{mg}})$

$c_0 = \text{Olm}.\textbf{Encrypt}(k_{AB}, (\mathfrak{S}_{gpk}, \sigma_{\text{mg}}))$

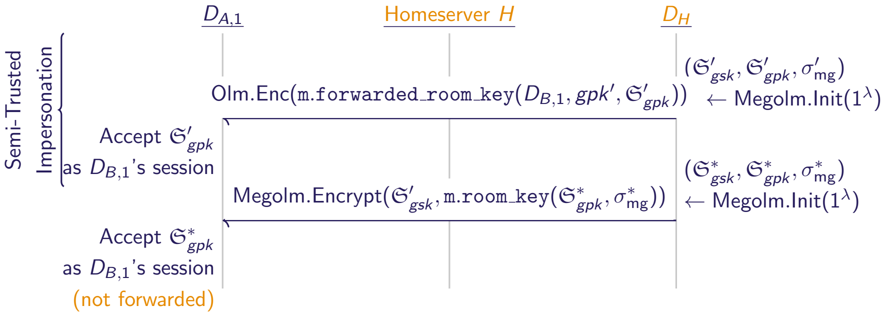
$c_1 = \text{Olm}.\textbf{Encrypt}(k_{AC}, (\mathfrak{S}_{gpk}, \sigma_{\text{mg}}))$

What if we could send $(\mathfrak{S}_{gpk}, \sigma_{\text{mg}})$ over Megolm instead of Olm?

Could we send it over a Megolm session placed via previous impersonation attack?

Layering Attacks for Full Impersonation

Device D_H impersonates $D_{B,1}$ to $D_{A,1}$:



$D_{A,1}$

Homeserver H

D_H

Semi-Trusted Impersonation

$(\mathfrak{S}'_{gsk}, \mathfrak{S}'_{gpk}, \sigma'_{\mathrm{mg}}) \leftarrow \mathsf{Megolm.Init}(1^\lambda)$

$\mathsf{Olm.Enc}(\mathtt{m.forwarded\_room\_key}(D_{B,1}, gpk', \mathfrak{S}'_{gpk}))$

Accept $\mathfrak{S}'_{gpk}$ as $D_{B,1}$'s session

$(\mathfrak{S}^*_{gsk}, \mathfrak{S}^*_{gpk}, \sigma^*_{\mathrm{mg}}) \leftarrow \mathsf{Megolm.Init}(1^\lambda)$

$\mathsf{Megolm.Encrypt}(\mathfrak{S}'_{gsk}, \mathtt{m.room\_key}(\mathfrak{S}^*_{gpk}, \sigma^*_{\mathrm{mg}}))$

Accept $\mathfrak{S}^*_{gpk}$ as $D_{B,1}$'s session (not forwarded)

Damage

Semi-trusted Impersonation Attack

Breaks authentication: Attackers can impersonate users

with some indication in (Element's) user interface.

Fully-trusted Impersonation Attack

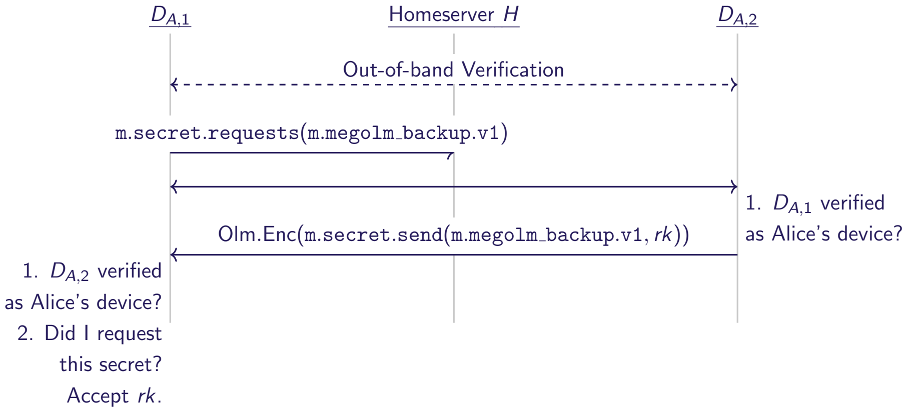
Breaks authentication: Attackers can impersonate users

with no indication in (Element's) user interface.

More Layers: Authentication to Confidentiality Break

When a user verifies their new device, it will use **SSSS** to request User Secrets from the user's existing devices.

This includes the **recovery key** used for **Megolm Backups**, i.e.



$D_{A,1}$

Homeserver H

$D_{A,2}$

Out-of-band Verification

`m.secret.requests(m.megolm_backup.v1)`

Olm.Enc(`m.secret.send(m.megolm_backup.v1,` rk`)`)

1. $D_{A,1}$ verified as Alice's device?

1. $D_{A,2}$ verified as Alice's device?
2. Did I request this secret?

Accept rk.

Damage

Semi-trusted Impersonation Attack

Breaks authentication: Attackers can impersonate users

with some indication in (Element's) user interface.

Fully-trusted Impersonation Attack

Breaks authentication: Attackers can impersonate users

with no indication in (Element's) user interface.

Authentication to Confidentiality Break

Breaks confidentiality: Attackers can eavesdrop on conversations

with no indication in (Element's) user interface.

Together: complete break of confidentiality and authentication!

Take Home Messages

- There is no confidentiality without authentication.
- Put all cryptographic code in one small core.[1]

[1] Element checked authentication at display time, rather than at receipt time and thus those checks were not run for messages that are not displayed.

Theoretical Confidentiality Break

Attack:

- Bug where the “initialisation vector” used in encryption is not integrity protected.
- Enables theoretical confidentiality break

Take home message:

- There is no confidentiality without integrity.[2]

[2] Corollary: The CIA triad – confidentiality, integrity, availability – is nonsense.

Recap & Status

1. Trivial confidentiality breaks not yet fixed
2. Attack on out-of-band verification CVE-2022-39250; reportedly mitigated
3. Impersonation CVE-2022-39246, CVE-2022-39249 and CVE-2022-39257;[3] reportedly mitigated
4. Full impersonation CVE-2022-39248, CVE-2022-39251 and CVE-2022-39255; reportedly mitigated
5. Impersonation to confidentiality break same CVEs as above; reportedly mitigated
6. Theoretical confidentiality attack not yet fixed

[3] In their review of the ecosystem the Matrix developers discovered further clients vulnerable to variants of our attack and assigned CVE-2022-39252, CVE-2022-39254 and CVE-2022-39264.

Outline

New phone, who dis?

Cryptography in Matrix

Attacks

Take Home Message

Difficult Problems!

Matrix aims to solve some difficult problems:

1. Secure (Group) Messaging

   . . . in a multi-device setting,

   . . . that is scalable to thousands of devices in a single group.
2. Backups and history sharing.
3. Authentication and identity verification

   . . . cross-signing to reduce user burden of out-of-band verification.
4. Federation.
5. Supporting a variety of clients across many platforms.

Cryptography is not a dark art

“Crypto is hard!”

Cryptography is not a dark art

"Crypto is hard!"

Of course, cryptography is hard, so is any other science.

Cryptography is not a dark art

"Crypto is hard!"

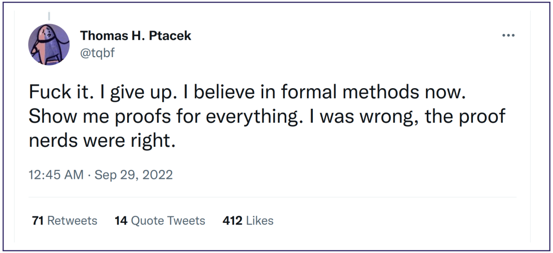
Modern cryptography gives us the tools to reason about cryptographic protocols to rule out the sort of issues we found here.

Cryptography is not a dark art

“Crypto is hard!”

“Cryptography needs security models and proofs!”

Cryptography is not a dark art



Thomas H. Ptacek
@tqbf

Fuck it. I give up. I believe in formal methods now. Show me proofs for everything. I was wrong, the proof nerds were right.

12:45 AM · Sep 29, 2022

71 Retweets **14** Quote Tweets **412** Likes

Fin

Thank you! Questions?

Never trust a cryptographic protocol without a formal proof of security.[4]

`https://nebuchadnezzar-megolm.github.io/`

[4] Yes, these have limitations. No, whitepapers and audits do not suffice.