Practically-exploitable Cryptographic Vulnerabilities in Matrix [a]

IETF 115 London, PEARG, 2022-11-09

Martin Albrecht (Royal Holloway) *martin.albrecht@rhul.ac.uk*
Sofía Celi (Brave Software) *cherenkov@riseup.net*
Benjamin Dowling (University of Sheffield) *b.dowling@sheffield.ac.uk*
Dan Jones (Royal Holloway) *dan.jones@rhul.ac.uk*

[a] Full paper available at `https://nebuchadnezzar-megolm.github.io/`.

Matrix

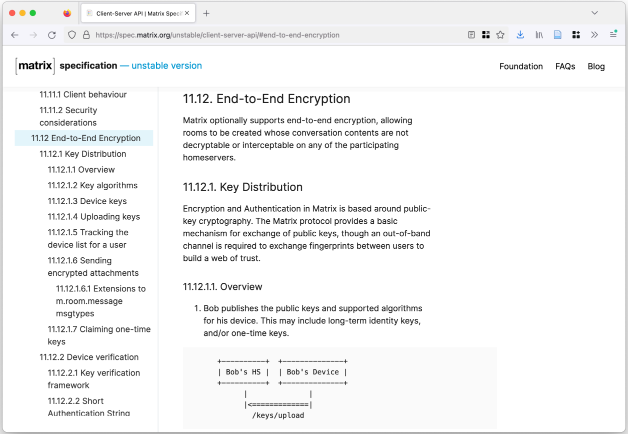
Matrix is a standard for secure, decentralised, real-time messaging.

Mission[1]:

- **Short-term** Interoperable messaging and calls.
- **Long-term** Underlying messaging and data synchronisation for applications.

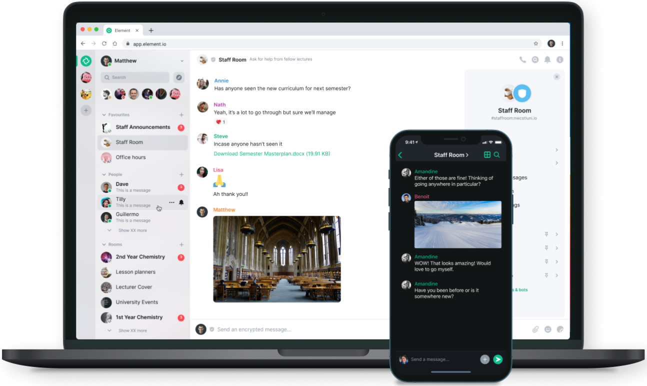
[1] `https://matrix.org/faq/#what-is-matrix's-mission%3F`

Matrix



[https://spec.matrix.org/unstable/]

Element (flagship client)

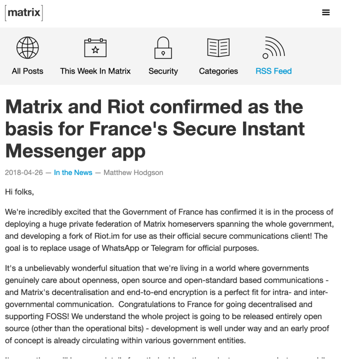


[https://element.io/images/Element-Home-Hero_1.png]

Why should we care?

Element has over 60 million users[2]

Matrix has users including...



[matrix]

All Posts This Week In Matrix Security Categories RSS Feed

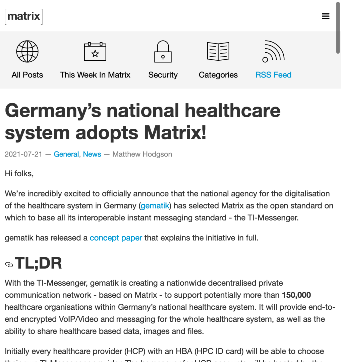
Matrix and Riot confirmed as the basis for France's Secure Instant Messenger app

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.



[matrix]

All Posts This Week In Matrix Security Categories RSS Feed

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

[2] https://archive.ph/2022.08.11-121218/https://element.io/

Secure messenger

End-to-End Encryption

- Confidentiality
- Integrity
- Authentication

- *Partial* Forward Secrecy?
- Post-compromise Security?
- Some form of Deniability?

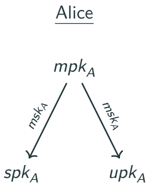
Matrix Overview

Parties

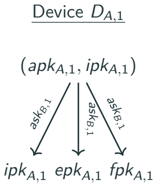
- User, device and homeserver.
- A homeserver is a server that stores the communication history and account information for a user.
- A user represents a user/client and has many devices (phone, desktop, etc).

Parties

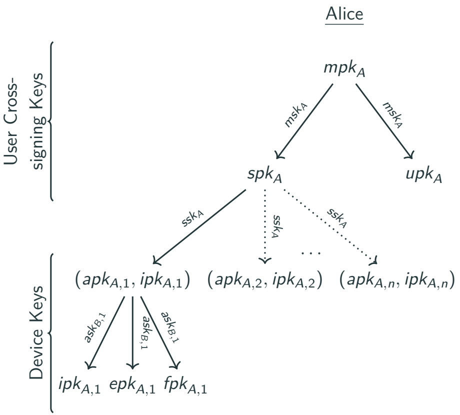
- Each user has a cryptographic identity used to:
  1. achieve trust between user's set of devices, and
  2. record the result of out-of-band verification between users.



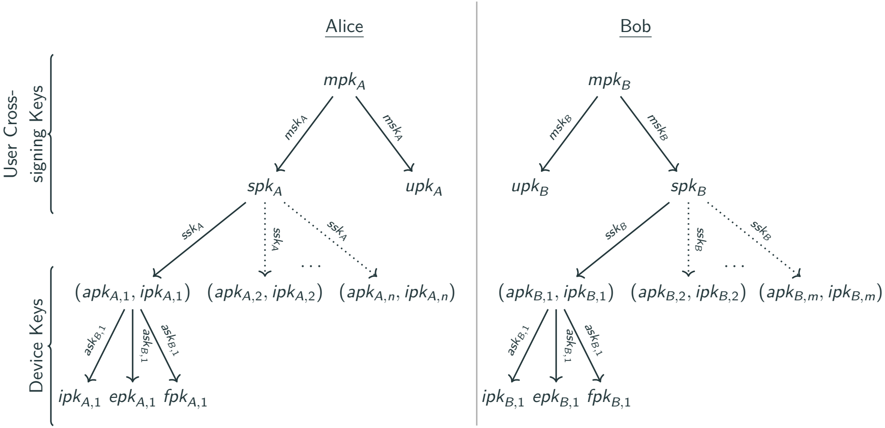
- Each device has a cryptographic identity used to communicate through the Olm and Megolm protocols.



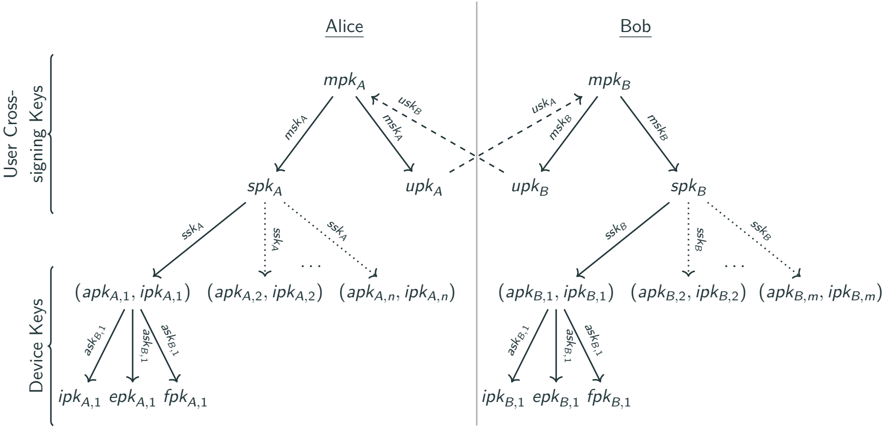
Cross-signing: linking everything together



Cross-signing: linking everything together



Cross-signing: linking everything together



Olm

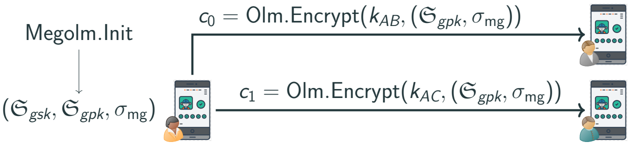
- Olm: pairwise secure channels between devices.
- Modified Signal's 3DH plus Double Ratchet algorithm.
- Connected to device's cryptographic identity.

Megolm

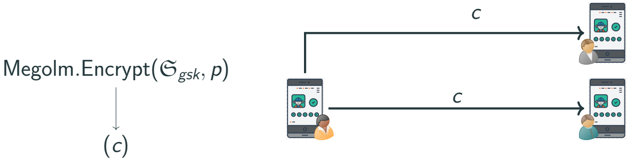
- Megolm: group messaging through composition of unidirectional channels.
- Effectively, Signal "Sender keys".
- Olm channels are used as a signalling layer to distribute the *inbound session*/sender keys (i.e. key material).

Megolm

Session setup and key distribution:



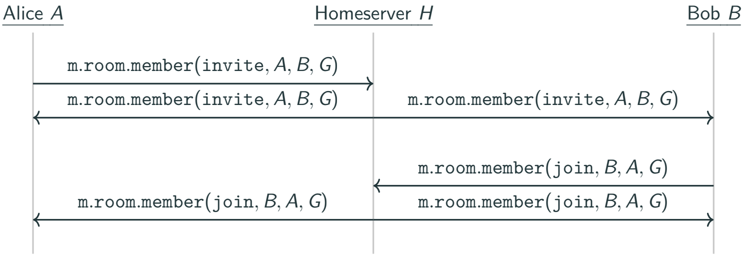
Messaging:



Attacks

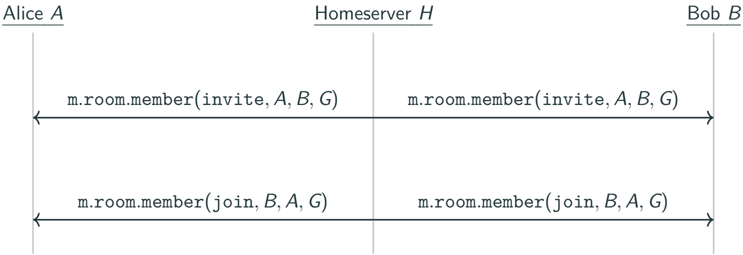
Attack 1a – Membership events are unsigned

Group membership is managed through events:



Attack 1a – Membership events are unsigned

Group membership is managed through unsigned events:



Attack 1a – Membership events are unsigned

What caused this attack?

- Assumption that only 'user messages' should be encrypted.
- Practical issues

Attack 1b – Server controls the list of user's devices

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

Attack 1b – Server controls the list of user's devices

What caused this attack?

- To stop this attack, users need a way to advertise a list of their trusted devices (that the homeserver cannot modify).
  $\rightarrow$ Cross-signing provides such a list!
- Is this too impractical? Too high a user burden?
- Arguably, yes. *For now!*

Attack 2 – Out-of-band Verification

How do two parties ensure their 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).

This attack targets step 3. The homeserver tricks device's into sharing a homeserver-controlled identity (rather than their own).

Attack 2 – Out-of-band Verification

How does the homeserver trick device's into sending a homeserver-controlled identity (rather than their own)*?*

- Two types of verification:
  1. Between two users
  2. Between two devices (of the same user)
- For step 3, each party sends the other an `m.key.verification.mac` 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 2 – Out-of-band Verification

How does the homeserver trick device's into sending a homeserver-controlled identity (rather than their own)?

Attack:

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

Attack 2 – Out-of-band Verification

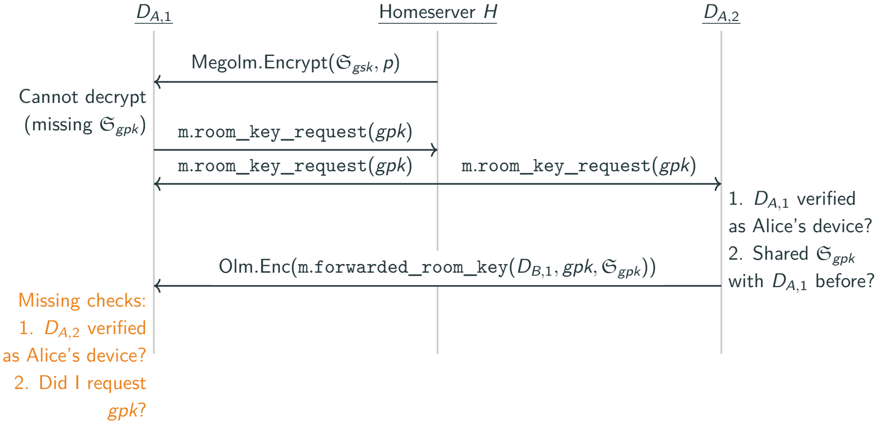
What caused this attack?

- Lack of domain separation between cross-signing key identifiers and device identifiers.
  $\rightarrow$ avoid using server-controlled inputs in the out-of-band verification process.

Attack 3 – Semi-trusted Impersonation

When a user adds a new device, they'd like that device to be able to decrypt messages *previously* sent to that user.

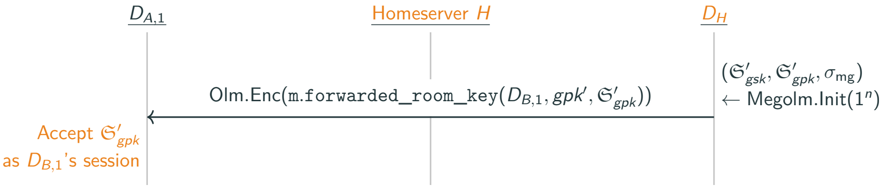
Key Request Protocol ≈



Attack 3 – Semi-trusted Impersonation

When a user adds a new device, they'd like that device to be able to decrypt messages *previously* sent to that user.

Attack:



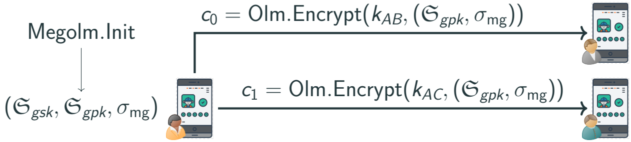
Attack 3 – Semi-trusted Impersonation

What caused this attack?

- Implementation mistake!
- Key Request Protocol was *underspecified*

Attack 4 – Trusted Impersonation

Recall Megolm session setup:

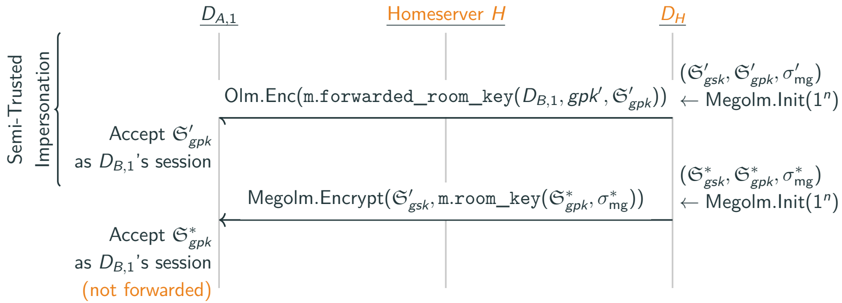


What if we could send $(\mathfrak{S}_{gpk}, \sigma_{\mathrm{mg}})$ over Megolm instead of Olm?

Could we send it over a Megolm session placed via the semi-trusted impersonation attack?

Attack 4 – Trusted Impersonation

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



Attack 5 – Confidentiality Break

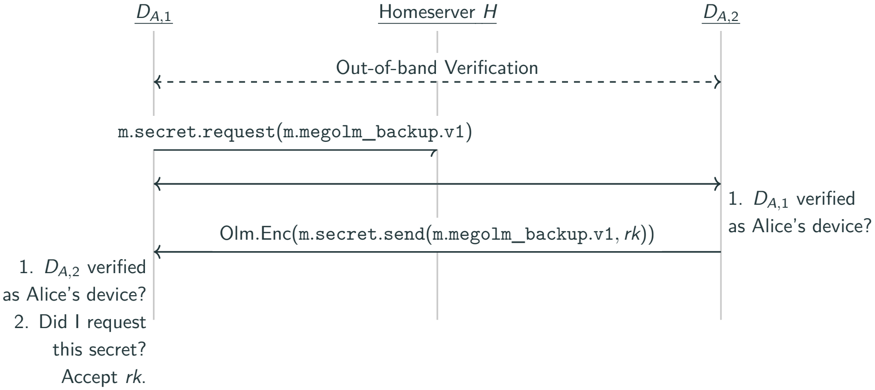
Introduce two new sub-protocols:

1. *Megolm Key Backups*
   - Inbound Megolm sessions $\mathfrak{S}_{gpk}$ are encrypted then backed up on the server.
   - A *recovery key* (shared between a user's devices) is used to decrypt them.
2. *Secure Storage and Secret Sharing (SSSS)*
   - Provides functionality to backup and share account-level secrets. E.g. cross-signing keys and the Megolm backups *recovery key*.
   - "Secret Sharing" between devices through synchronous request-response protocol
   - "Secure Storage" through backups on the homeserver (through a shared symmetric key).

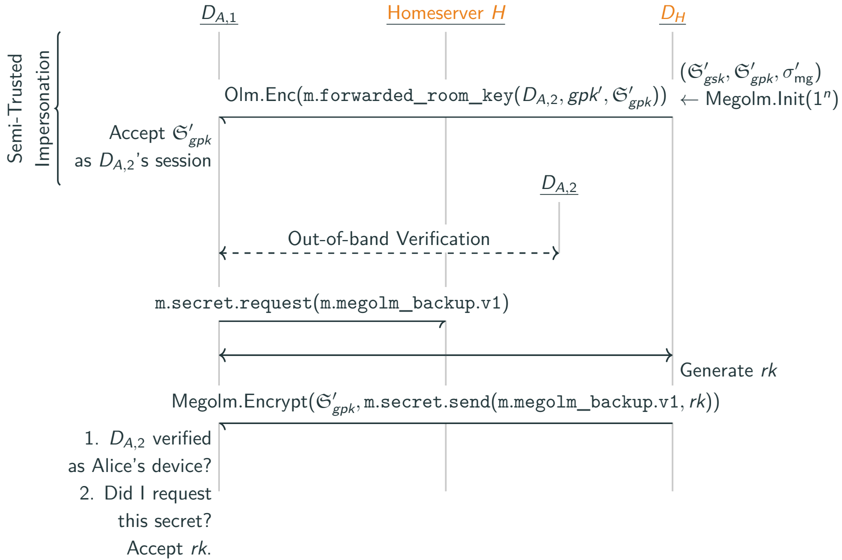
Attack 5 – Confidentiality Break

When a user verifies their new device, it will use SSSS to request account-level secrets from the user's existing devices.

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



Attack 5 – Confidentiality Break



Attacks 4 & 5 – Protocol Confusion

What caused these attack?

- Implementation mistake!
- *Looking deeper...* how could the specification discourage similar bugs in the future?

Lessons Learned

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.

Revisited: Secure messenger

End-to-End Encryption? Yes

- Confidentiality? Yes
- Integrity? Yes
- Authentication? Yes

- *Partial* Forward Secrecy? Maybe?
  (Forward Secrecy? No)
- Post-compromise Security? Maybe?
- Some form of Deniability? Maybe?

Managing Complexity

Formal proofs!
(and security analysis)

Why?

- Require clear and consistent thinking about threat models.
- Identify gaps in the specification.
  Requires a more detailed and prescriptive specification.
- Encourage a more compact, provable design.

Other thoughts

To think

Matrix homeservers accumulate a wealth of metadata

We need design that is generated by inputs of several places: formal analysis, research, standardization...