

Invisible Cryptography

Crypto usability for Matrix clients
20.09.2024

Valère Fédronic // @valere35:matrix.org
Patrick Maier // @pmaier:element.io

Agenda

1. Matrix crypto today
2. A proposal to make crypto invisible
3. So now I always have to verify my devices?
4. Summary

1. Matrix crypto today

Matrix has very high standards

- Real multi-device support right from the beginning
- Device lifecycle with frequent log in / log out (Web)
- Access to encrypted history on new devices
- Threat model: Decentralization and homeserver trust

Matrix crypto is very complicated and wants to cover a lot more challenges than centralized, single-device, or even unencrypted messaging services.

Up until now these requirements made it very challenging to provide good usability...

Terminology is technical and inconsistent



Notifications

Preferences

Keyboard

Sidebar

Voice & Video

Security & Privacy

Labs

Encryption

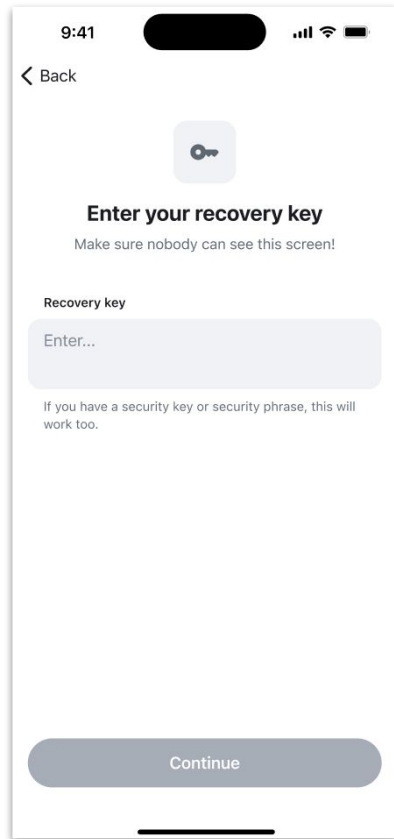
Secure Backup

Back up your encryption keys with your account data in case you lose access to your sessions. Your keys will be secured with a unique Security Key.

✔ This session is backing up your keys.


▶ [Advanced](#)

[Restore from Backup](#) [Delete Backup](#) [Reset](#)



9:41

< Back



Enter your recovery key

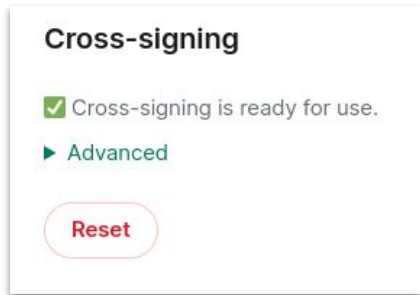
Make sure nobody can see this screen!

Recovery key

Enter...

If you have a security key or security phrase, this will work too.

[Continue](#)

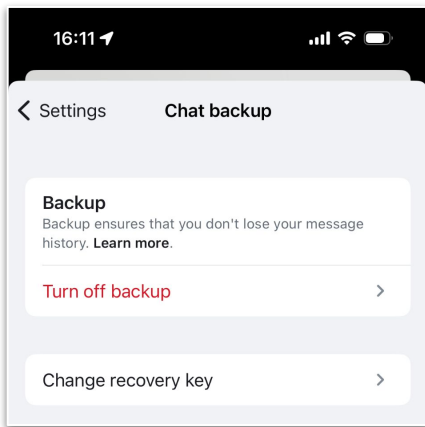


Cross-signing

✔ Cross-signing is ready for use.

▶ [Advanced](#)

[Reset](#)



16:11

< Settings Chat backup

Backup

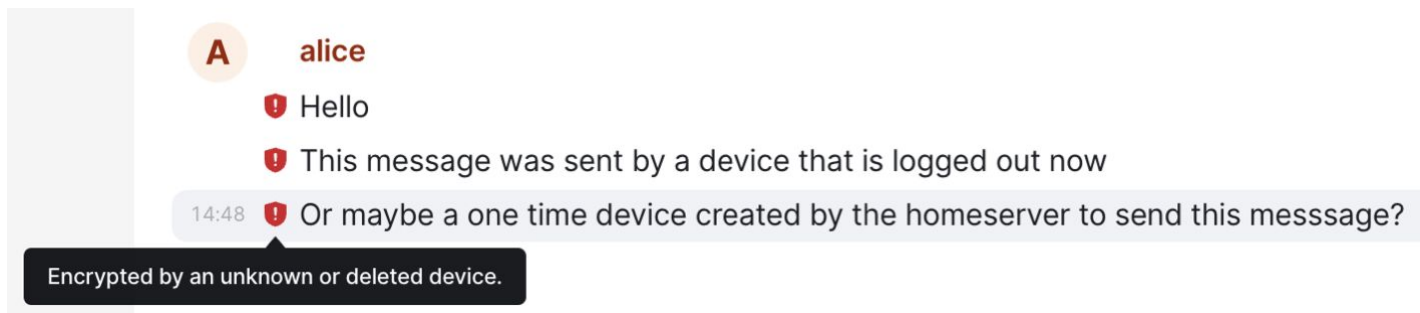
Backup ensures that you don't lose your message history. [Learn more.](#)

[Turn off backup](#) >

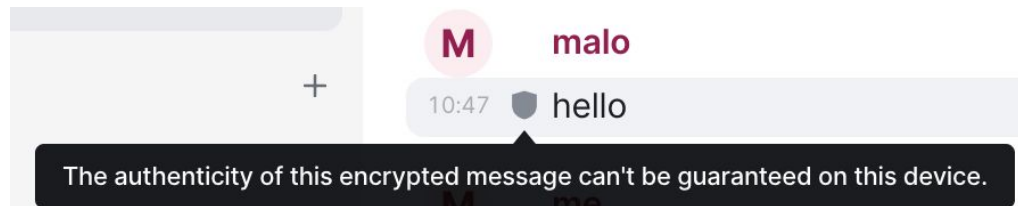
[Change recovery key](#) >

Attack risks are deferred to users

- A user has deleted a device they sent a message from

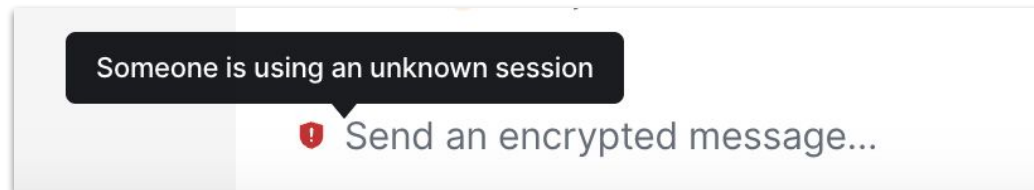
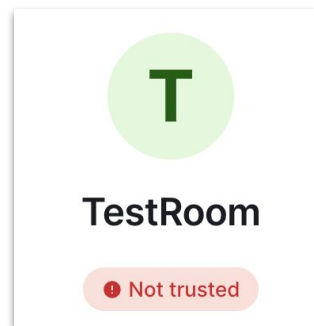
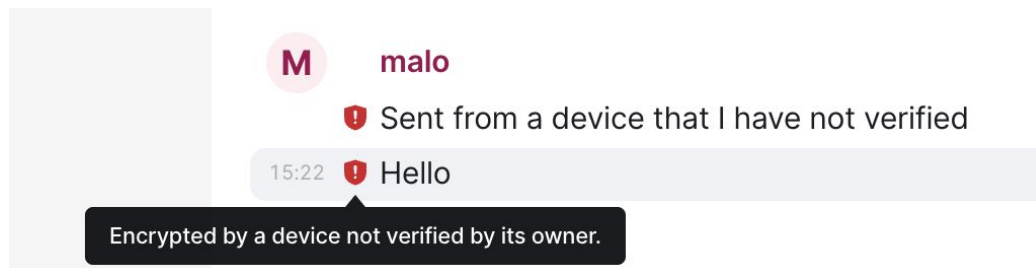


- You recover message history on a new device



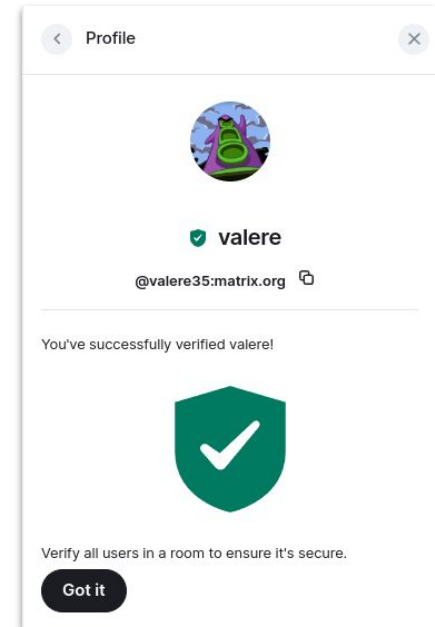
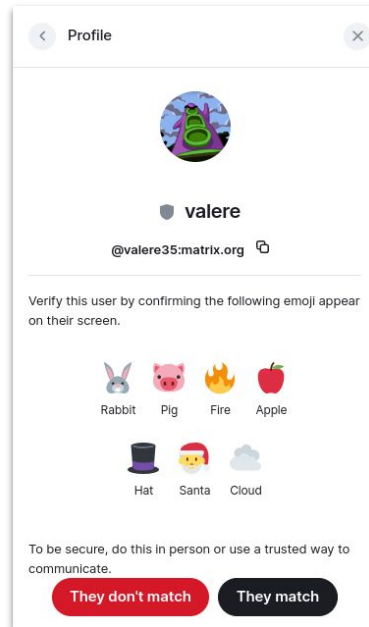
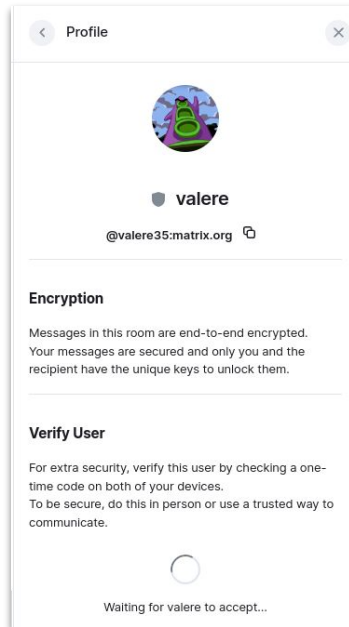
Attack risks are deferred to users

- Users send messages from a unverified devices or change their identity



Trust between users is manual

- Trust between users is crucial to prevent MITM attacks
- Currently it requires users to consciously establish trust, otherwise there is no protection



We have powerful tools not used fully

- Cross-signing
- Secret Storage + Key backup
- Dehydrated device
- QR verification
- Matrix historically gave users a lot of options/freedom to use the product in different ways, which makes it complicated sometimes
 - Clients with or without encryption support
 - Encrypted vs. unencrypted rooms
 - Verified vs. unverified devices
 - Cross-signing or not, backup but no secret storage...

Good crypto is invisible.

2. A proposal to make crypto invisible

Crypto terminology for non-technical users element

- To deal with the inconsistencies and technical language, we propose an MSC!
- **MSC4161: Crypto terminology for non-technical users**
 - Suggests conventions for user-facing crypto-related features in Matrix clients
 - *Devices*
 - *Verified person*
 - *Identity*
 - *Recovery key*
 - *etc.*
 - Provides usage examples and usage to avoid

Key storage

Key storage means keeping cryptographic information on the server. This includes the user's cryptographic identity, and/or the message keys needed to decrypt messages.

If a user loses their recovery key, they may **reset** their key storage. Unless they have old devices, they will not be able to access old encrypted messages because the message keys are stored in key storage, and their cryptographic identity will change, because it too is stored in key storage.

| "Allow key storage"

| "Key storage holds your cryptographic identity on the server along with the keys that allow you to read your message history."

| "Message history is unavailable because key storage is disabled."

⚠ Avoid distinguishing between "secret storage" and "key backup" - these are both part of key storage.

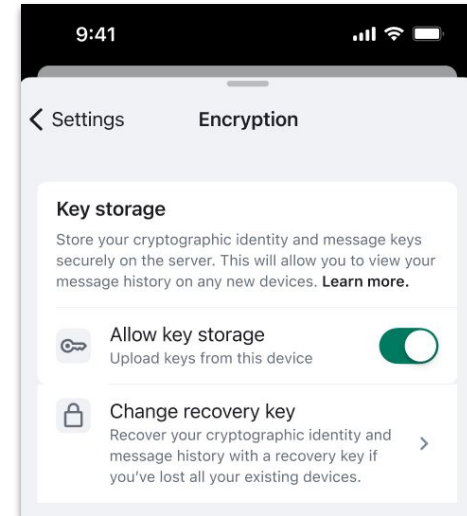
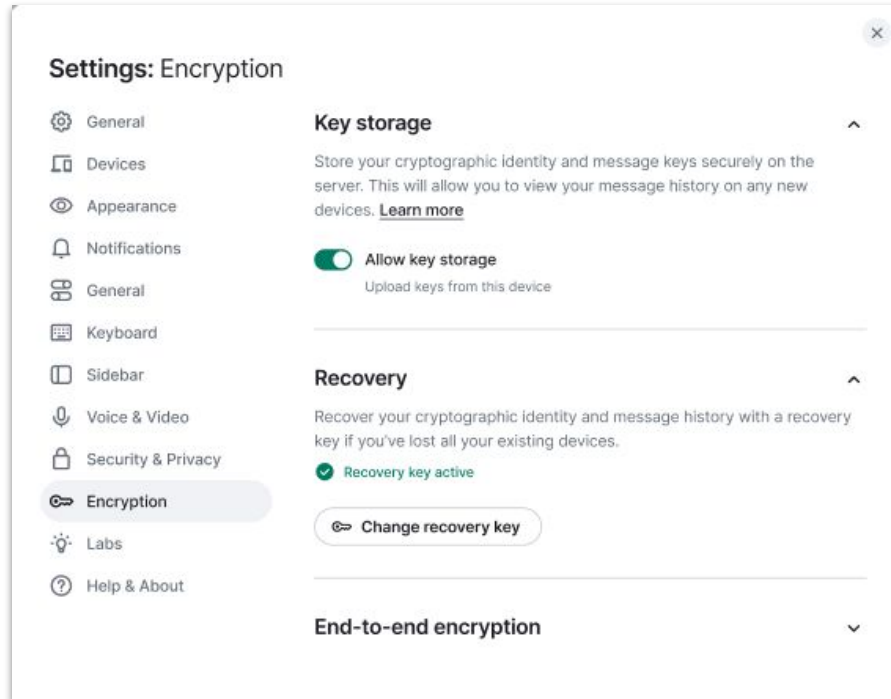
⚠ Avoid talking about more keys: "the backup key is stored in the secret storage, and this allows us to decrypt the messages keys from key backup". Instead, we simply say that both cryptographic identity and message keys are stored in key storage.

⚠ Avoid using "key backup" to talk about storing message keys: keeping things on the server is not a "backup", but a reliable, cross-device place where this information is stored. The word "backup" implies a redundant way to recover lost information, but if the user loses their recovery key, this information is lost. Clients and servers may wish to offer additional backup services that provide true redundancy and disaster recovery, but key storage is not this.

⚠ Avoid "4S" or "quad-S" - these are not descriptive terms.

⚠ Avoid "private key" - this is an implementation detail and a term with specific meaning from cryptography.

New concepts based on MSC4161



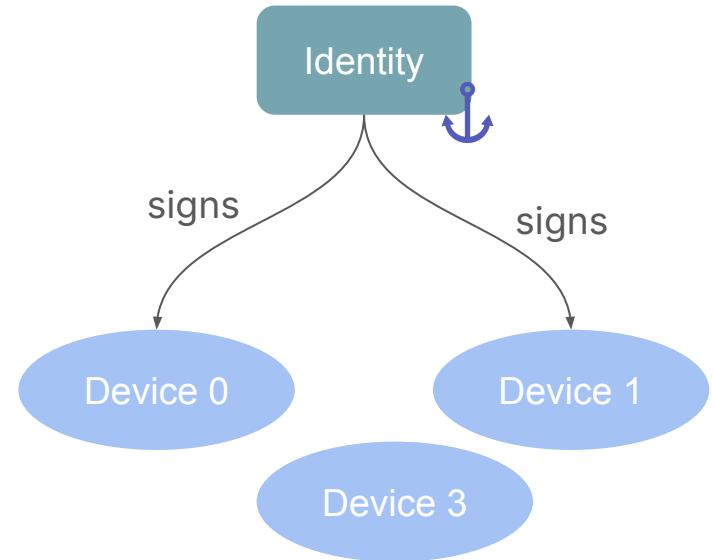
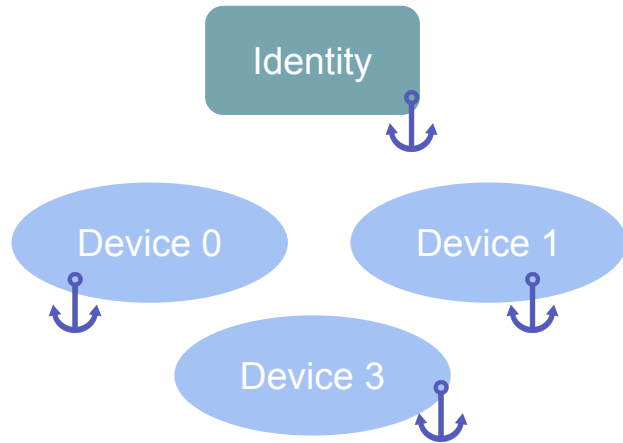
Trust & decorations

In a nutshell:

- Cross-Signing and device verification becomes mandatory
- You only send/accept keys to/from verified devices
- Identity Pinning

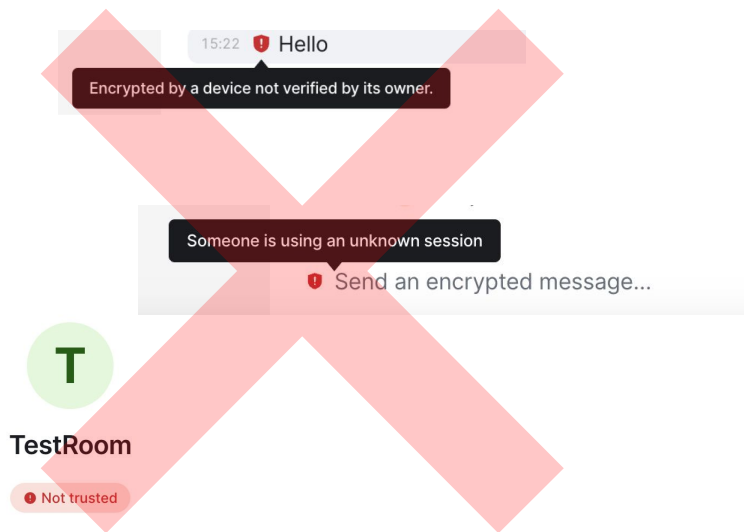
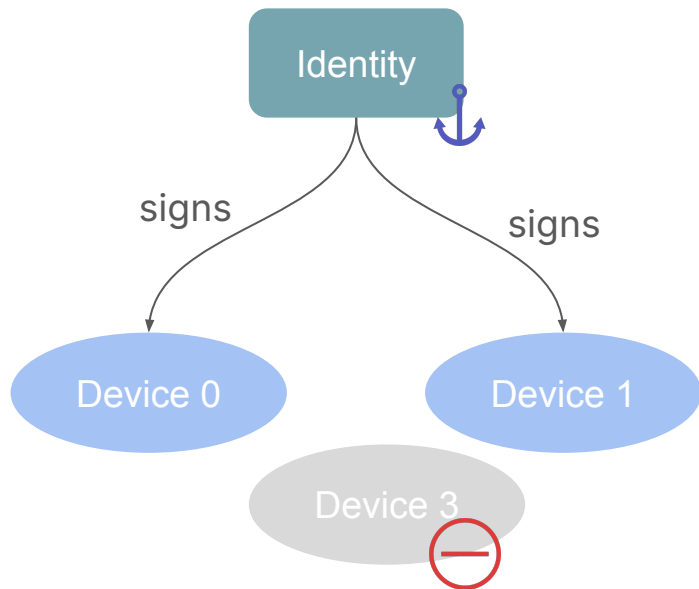
Device verification becomes mandatory

- We are moving away from device granularity to identity granularity. Right now, we usually have $N+1$ unauthenticated nodes where N is the number of devices and that extra 1 is for your user identity. We want at most one now.



Devices not cross-signed are ignored

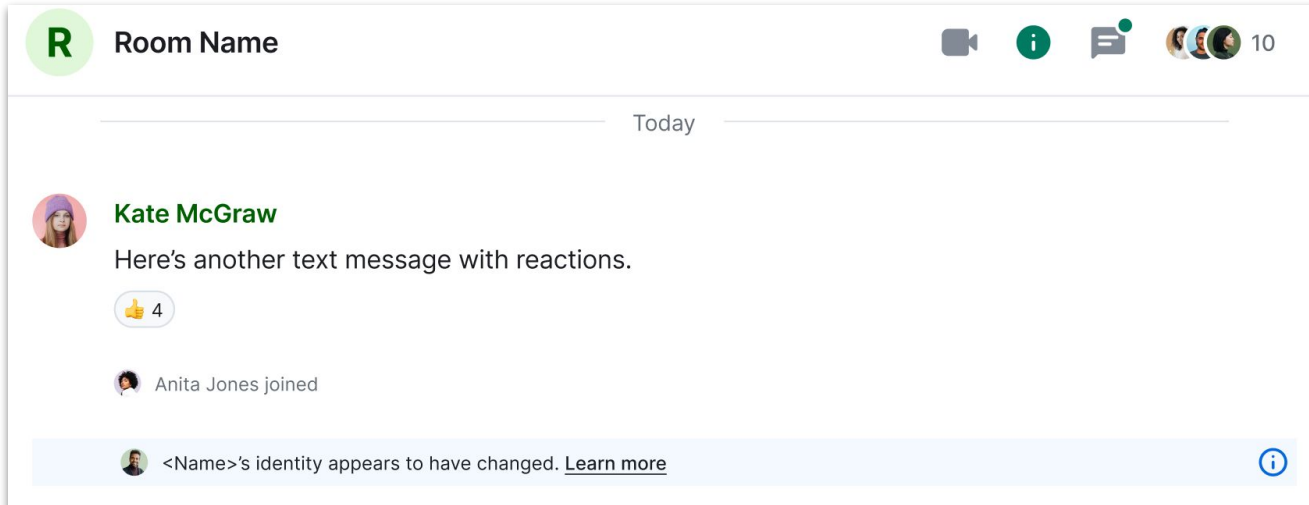
Devices not signed by their owners do not exist for the crypto layer. They won't be able to read messages, and the messages they sent will be ignored.



Key Pinning - Better default security

Initially blindly trust the identity, and display non-blocking warnings when identity changes.

It will not be considered as high a level of trust. Nevertheless, this raises the bar for Mallory-in-the-middle attacks by malicious homeservers, as they can no longer quietly falsify signing keys for users who are already in contact with each other.



The screenshot shows a chat room interface with a header bar containing a room name, video call icon, info icon, message icon, and a group of 10 users. A horizontal line separates the header from the chat history, with the word "Today" centered. A message from "Kate McGraw" is shown with a thumbs-up reaction from 4 users. Below it, a system message states "Anita Jones joined". At the bottom, a light blue warning banner displays a user icon, the text "<Name>'s identity appears to have changed. [Learn more](#)", and an info icon.

Identity mismatch for previously verified users

For more security, you can still verify users.

In that case the identity change situation will be handled differently, moving from non-blocking warnings to blocking warnings, e.g require a manual action from the user to fix it (re-verify, or withdraw verification).



Bob

Message goes here

12:36

 Verified identity has changed



<Name>'s verified identity has changed.

[Learn more](#)

Go to profile

Withdraw verification

Authenticated backup

- To provide the ability of verifying the authenticity of message keys stored in key storage, we propose an MSC!
- **MSC4048: Authenticated backup**
 - Message keys uploaded to the key storage will have an authentication tag (MAC)
 - When a user recovers history on a new device, the MAC will be used to determine the authenticity/integrity of the keys
 - Unauthenticated keys will just be *dropped* in the future and users won't be bothered anymore!

MSC4048: Authenticated key backup

The [server-side key backups](#) allows clients to store event decryption keys so that they can read old messages. The current algorithm encrypts the event keys using an asymmetric algorithm, which necessarily gives them the ability to read from the backup. For example, this allows clients to read old messages (the keys for) current messages, but not read old messages.

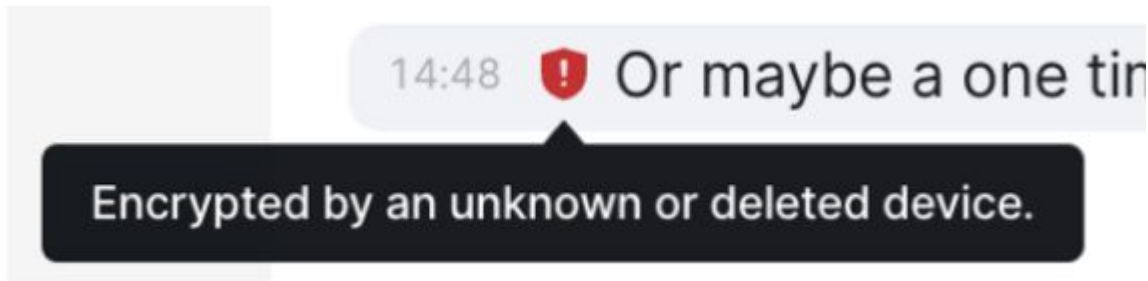
However, since the event decryption keys are encrypted using an asymmetric algorithm, they are not readable to the backup. As a result, keys loaded from the backup must be marked as unauthenticated.

[MSC3270](#) tries to fix this issue by using a symmetric, authenticated encryption algorithm. However, a secret key can write to the backup. However this removes the ability for a client to read old messages from it.

We propose to continue using an asymmetric encryption algorithm in the backup, but to mark keys derived from the backup's decryption key.

Technical Challenge: Unknown/Deleted devices

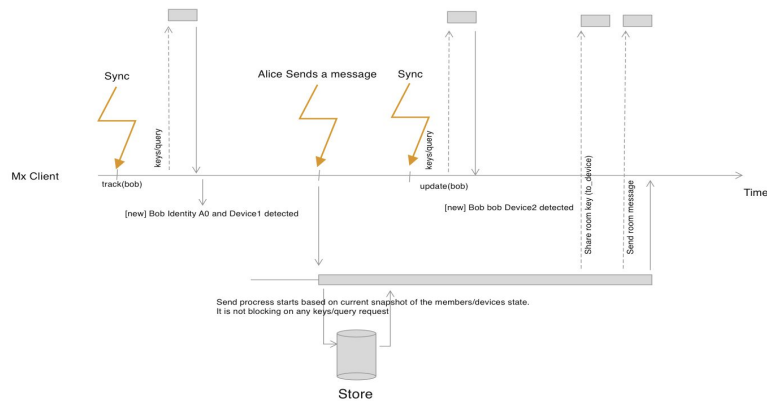
- [MSC4147: Including device keys with Olm-encrypted events](#)
- Ignoring “unsigned” devices means that we need to know at time of reception of a message what is the status of the sending device.
 - Establishing communication with olm is asynchronous, you will receive messages from devices before you know they existed.
 - And it can be a message sent by a device that is deleted now.



All the heavy lifting is done in rust-sdk

```
pub fn identity_needs_user_approval(&self) -> bool {...}
```

```
impl<'a> SenderDataFinder<'a> {  
    /// Find the device associated with the to-device message used to  
    /// create the InboundGroupSession we are about to create, and decide  
    /// whether we trust the sender.  
}
```



3. So now I always have to verify my devices?

...even if I only use public rooms?

- Yes, Element clients will force users to **verify their own devices** in the future (not verifying other users!)
- Less choice for users but
 - Security by default
 - Ensure clients have everything they need to operate properly
 - ... without bothering users with shields and other weird indicators

But isn't this all going to be very cumbersome?

...even if I only use public rooms?

- Yes, Element clients will force users to **verify their own devices** in the future (not verifying other users!)
- Less choice for users but
 - Security by default
 - Ensure clients have everything they need to operate properly
 - ... without bothering users with shields and other weird indicators

QR code login is here to save you!

Just scan a QR code from a signed-in device and everything else will happen automagically

- Transfers connection information (e.g., which homeserver)
- Signs the device in (using OAuth 2.0 device code grant)
- Verifies the device and transfers crypto secrets

QR code login (existing Web device)

Patrick Maier
@pmaier:element.io

- Link new device
- Notifications
- Security & Privacy
- All settings
- Feedback
- Sign out




Settings: Devices

- Account
- Devices**
- Appearance
- Notifications
- General
- Keyboard
- Sidebar
- Voice & Video
- Security & Privacy
- Encryption
- Labs
- Help & About

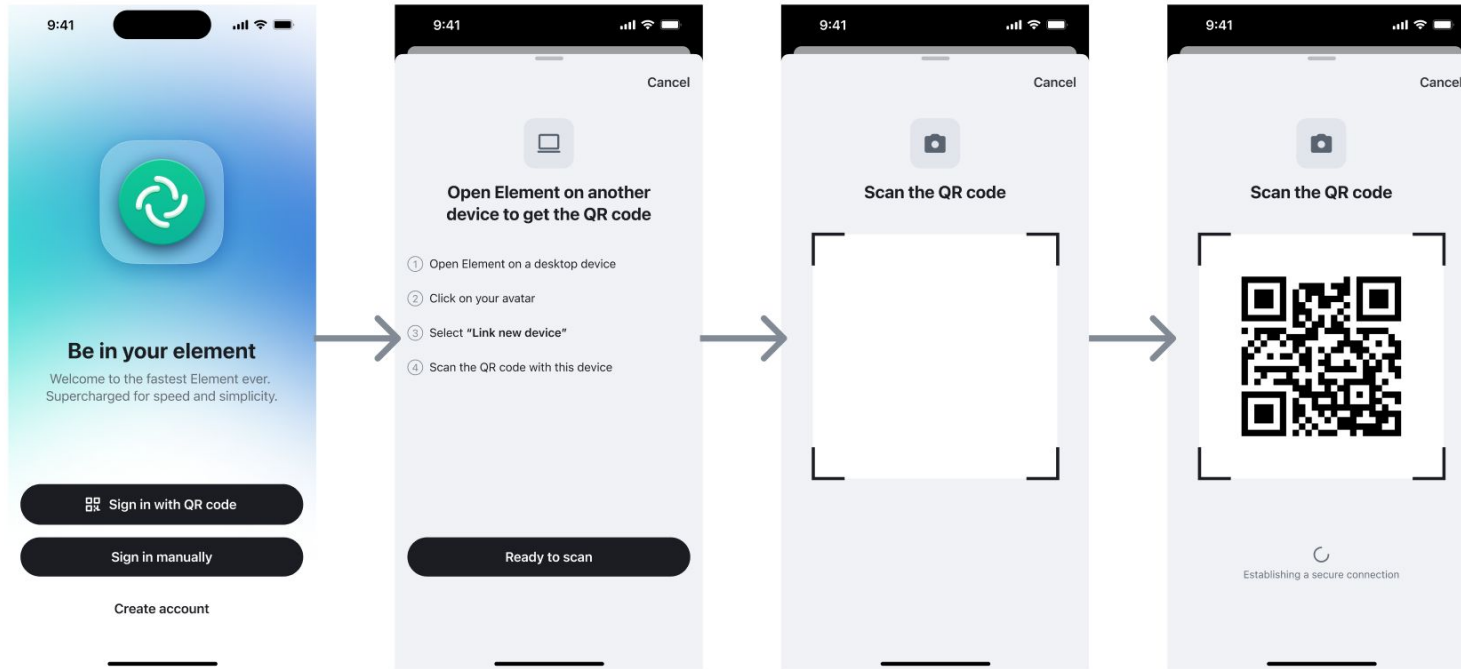
< [Devices](#) / Link new device

Scan the QR code with another device



- 1 Open Element on your other device
- 2 Select "Sign in with QR code"
- 3 Scan the QR code shown here
- 4 Follow the remaining instructions

QR code login (new EX device)



4. Summary

Invisible Crypto needs your support!

- Invisible Crypto is a collection of concepts to improve crypto usability in Matrix
 - Crypto terminology for non-technical users (MSC4161)
 - Identity pinning, exclude unsigned devices & mandatory device verification (MSC4153)
 - Authenticated backup (MSC4048)
 - Including device keys with Olm-encrypted events (MSC4147)
- *Client developers*, please engage with MSC4161 and provide feedback
- *Client developers*, please engage with the idea of excluding unsigned devices (MSC4153)
 - Element is proposing this change to simplify and to keep users secure by default
 - It's crucial that we establish a convention here
 - There will be a transition phase for Element clients

Questions?