Opening up comms silos with Matrix 2.0 and the EU Digital Markets Act

FOSDEM 2024

matthew@matrix.org
@matthew:matrix.org
Matrix is an open network for secure, decentralised real-time communication.

- Interoperable chat
- Interoperable VoIP
- Open comms for VR/AR
- Real-time IoT data fabric
Our mission: To build the real-time communication layer of the open Web.
No single party owns your conversations.

Conversations are shared over all participants.
Visible Monthly Active Users (unbridged!)
“The most admired synchronous comms tool”  
(...and the most desired open source one)

Some Big Public Sector Matrix Deployments

- US Govt
- Sweden
- NATO
- UK Govt
- Poland MOD
- gematik
- Ukraine MOD
- Polska MOD
- BundesMessenger
- openDesk
- Phoenix Suite
- NRW Schools
- Bavaria Schools
- Hessen Administration
- LuxChat
- France (Tchap)

Size of financial support to the Matrix core team in 2023
2023 was a really rough year.

- COVID funding evaporated + general macroeconomic slowdown
- Lots and lots of large deployments not helping funding underlying dev.
- “Public Money For Public Code” ⇒ Govts only want to fund new features.
- This has forced focus - on Matrix 2.0, Synapse, matrix-rust-sdk (Element X) and matrix-js-sdk (Element Web & Element Call) - and nothing else.
- Everything else is paused: P2P Matrix, Pseudo IDs, Crypto IDs, Account Portability, Low Bandwidth Matrix, Element-funded Dendrite work
- …critical bugfixes only: matrix-ios-sdk & matrix-android-sdk (Element iOS/Android); libolm (replaced by vodozemac)
- …or gone: Third Room.
- ⇒ Element ended up switching its development on Synapse to AGPL in order to sell AGPL exceptions to those who need them.
The Foundation now runs entirely independently, with Josh Simmons as MD!

Setting up a Governance Board from across the ecosystem to steer the direction of the project - elections in April 2024.

Fundraising right now to support core spec work, trust & safety work, bridging improvements, running the matrix.org infrastructure and governance work - target £900K.

To support, join the Foundation: [https://matrix.org/membership/](https://matrix.org/membership/)

Meanwhile, much of [github.com/matrix-org](https://github.com/matrix-org) is written and maintained by the core team hired by Element, who donates their time to the project - please support them by buying enterprise Matrix deployments from Element if you’re a Government or Enterprise.

For Matrix to prevail, we need your support.
Current members and supporters...

...and >716 individual donors!
Matrix 2.0 Status
Matrix 2.0

● We announced the idea of Matrix 2.0 at FOSDEM 2023

● Mission: to make Matrix as **fast** and usable as the mainstream alternatives:
  ○ Sliding Sync (MSC3575)
  ○ Faster Joins (MSC3902)
  ○ Native VoIP (MSC3401)
  ○ OpenID Connect (MSC3861)

● **Not a new spec release** (yet).

● Showcased in matrix-rust-sdk, as used in Element X and GNOME Fractal 5

● Back at FOSDEM 23 last year, this was distinctly alpha :D

● In Sept 2023, it became available to everyone with Element X Ignition!
Demo!
# Sliding Sync

- Thesis: the server should only tell the client about the rooms the client needs to display - $O(1)$ with number of rooms, not $O(N)$.
- It’s been a bit of a journey :D
- Reworked the matrix-rust-sdk implementation; added unread rooms state!
- What is the right balance between server-side and client-side ordering?
- Original idea: order rooms serverside, clients get a sliding window & receive ops to update it; and fix up ordering clientside $\Rightarrow$ optimal solution! 🎉
- Problem: only clients know the right order for E2EE rooms, and E2EE rooms are pretty common these days. Also, the “fix up” is horribly fragile 😱
- Solution: sort primarily on the client (and use coarse heuristics on the server to incrementally send most relevant rooms first) $\Rightarrow$ “Pragmatic Sync”.
- This is basically a subset of Sliding Sync, without the *cough* sliding bit.
**Sliding Sync**

- Client-side ordering is in flight:
  - https://github.com/matrix-org/matrix-rust-sdk/pull/3068 (opened a few days ago)
- We’re not thinking about native Sliding Sync implementations serverside until we’ve finished iterating (i.e. simplifying) the current API.
- Meanwhile, it’s **really easy** to run your own Sliding Sync proxy:
  
  ```
git clone https://github.com/matrix-org/sliding-sync && cd sliding-sync
go build ./cmd/syncv3
createdb syncv3

echo -n "$(openssl rand -hex 32)" > .secret
SYNCV3_SECRET=$(cat .secret) SYNCV3_SERVER="https://matrix.example.com" SYNCV3_DB="user=$(whoami) dbname=syncv3 sslmode=disable password='hunter42'" SYNCV3_BINDADDR=127.0.0.1:8009 ./syncv3
  # route /_matrix/client/unstable/org.matrix.msc3575/sync to your 127.0.0.1:8009
  # add "org.matrix.msc3575.proxy": { "url": "https://matrix.example.com" } to https://example.com/.well-known/matrix/server
  ```
Native E2EE Group VoIP

- We finally have stable, end-to-end encrypted, scalable, VoIP!
- Calling and E2EE signalled over Matrix
- Uses LiveKit as an Selective Forwarding Unit
- Built on matrix-js-sdk
- Runs on [https://call.element.io](https://call.element.io) as a Single Page App
- Also embedded in Element X and Element Web (if you turn it on in labs, replacing Jitsi) - using the host client for encryption!
- When embedded, streams are encrypted per-sender!
- Interoperates with FluffyChat - see the talk in the devroom later today!
- Next up: finalising the spec and turning on by default everywhere!
Native OpenID Connect

- The great transition to native OpenID Connect is in full swing!
- So many benefits:
  - Support 2FA, MFA, Passkeys etc via an OIDC IdP!
  - Login via QR code, complete with E2EE identity! (almost)
  - No more implementing Matrix auth flows on every client (and homeserver)!
  - Users only ever send their password to their server, not random clients
  - Consistent auth and account management experience across apps
  - Integrates seamlessly with password managers
  - Lets users share authentication between apps (SSO)
  - Finally gives access-token refresh by default
  - OIDC scopes let users control what features an app can access.
Native OpenID Connect

Allow access to your account?

Device
IP address: 83.167.152.9
Access requested: Today 9:08 PM
Code: DELTV6

Another device wants to access your account. This will allow FOSEDM Example to:

- See your profile info and contact details
- View your existing messages and data
- Send new messages on your behalf

Make sure that you trust FOSEDM Example. You may be sharing sensitive information with this site or app. Find out how FOSEDM Example will handle your data by reviewing its privacy policy and terms of service.

Continue
Cancel

Not logged in? Sign out

Privacy Policy • Terms & Conditions
All Rights Reserved. The Super Chat name, logo and device are registered trade marks of Ring Corp Ltd.
Native OpenID Connect

- You can run matrix-authentication-service (MAS) today alongside Synapse as a small Matrix-aware OIDC IdP (written in Rust!)
- MAS provides UI for login, permissions, Matrix account management in a standard OIDC form factor.
- Hooks into Synapse to wrangle accounts and devices.
- Migration is now available in **syn2mas**:
- Provides some backwards compatibility for Matrix auth, but right now missing account deactivation, device kick-out and email bindings.
- Requires a Native OpenID capable client: Element X, Element Web available in Labs.
matrix-rust-sdk

- Clientside Matrix 2.0 implementation has been happening in matrix-rust-sdk and Element X
  - Sliding sync + OIDC support
- All new matrix-sdk-ui crate for providing higher level UI abstractions:
  - Lazyloaded ordered roomlist
  - Lazyloaded precomputed timelines
  - Sync spinner
  - Filters…
matrix-sdk-crypto in matrix-js-sdk

- At last, matrix-js-sdk and matrix-rust-sdk have converged on the same E2EE implementation: matrix-sdk-crypto from matrix-rust-sdk.

- **Merged in matrix-react-sdk on Friday!!!!**
  - [https://github.com/element-hq/element-web/pull/26939](https://github.com/element-hq/element-web/pull/26939)

- Fix E2EE bugs in one place, and get a single reference codebase audited.

- Also landed in matrix-android-sdk2 and matrix-ios-sdk in 2023.

- No more libolm! vodozemac ftw! 🦀🦀🦀
Crypto reliability

- Now that everyone is finally speaking matrix-rust-sdk for crypto, we can fix the remaining reliability issues in one place.

- **complement-crypto** is one of our main weapons in the fight.
  - Tests matrix-rust-sdk and matrix-js-sdk against real homeserver federations (running in docker)
  - Written in Golang, built on complement
  - Includes torture tests and unhappy-path tests
  - Failing tests for all remaining known issues.

- The race is on!

- (And then… as if by magic…)
Now that everyone is finally speaking matrix-rust-sdk for crypto, we can fix the remaining reliability issues in one place.

**complement-crypto** is one of our main weapons in the fight.

- Tests matrix-rust-sdk and matrix-js-sdk against real homeserver federations (running in docker)
- Written in Golang, built on complement
- Includes torture tests and unhappy-path tests
- Failing tests for all remaining known issues.

The race is on!

(And then… as if by magic… a draft PQXDH PR appeared)

- [https://github.com/matrix-org/vodozemac/pull/120](https://github.com/matrix-org/vodozemac/pull/120)
What’s next?

● Actually releasing this all as Matrix 2.0
● …and get it all audited!
● Native Sliding Sync
● Replacing matrix-{ios,android}-sdk with matrix-rust-sdk entirely
● Get matrix-react-sdk talking Sliding Sync
  ○ …or think about replacing matrix-js-sdk with matrix-rust-sdk? 😈
● Foundation-funded Trust & Safety work
● Foundation-funded bridging work.
● …and DMA.
The Digital Markets Act
The Digital Markets Act

- The EU Digital Markets Act mandates that communication services from big tech companies must interoperate together.
- Lets users pick their preferred service without sacrificing interoperability.
- Forces gatekeepers to differentiate based on quality, rather than relying on the network effects of a silo.

![Timeline diagram showing key dates and steps involved in the Digital Markets Act process.]

We are here!
The Digital Markets Act

● Once in a lifetime opportunity to see if we can use Matrix as a common language to talk to the large messaging providers!

● DMA requires gatekeepers to provide same level of E2EE for interoperability as for their existing service. Three options:

1. **Open APIs** + polyglot (aka multihead) messengers - e.g. Beeper Mini
2. **Client-side bridging** - install a “Gatekeeper<>Matrix app” to copy traffic back & forth between the gatekeeper service & Matrix.
   - We demoed this to the EC using WhatsApp and Google Chat in Feb 2023: https://matrix.org/blog/2023/03/15/the-dma-stakeholder-workshop-interoperability-between-messaging-services/
3. **Everyone talks the same protocol** (i.e. the gatekeeper protocol gets converted into Matrix or similar).

● However, over the last year we’ve been experimenting with **Option 3**.
DMA Challenges

● Two big challenges:
  ○ The gatekeeper has to speak the same end-to-end encryption protocol (but doesn’t have to speak the same signalling protocol)
  ○ Everyone has to use the same content format within the E2EE payloads.
● Good news: we picked The Double Ratchet for Matrix’s encryption back in 2015 because it was best of breed and everyone was converging on it:
  ○ Signal, WhatsApp, Google Allo + Messages, Skype, Viber, Wire, Wickr...
● ⇒ Pretty much everyone (other than Apple) uses libsignal (or libolm/vodozemac) under the hood for their E2EE today.
● Bad news: Matrix’s dialect (Olm) isn’t interoperable with libsignal.
A brief history of Olm

- Matrix’s version of the Double Ratchet is called **Olm** (a type of salamander)
- Clean-room implementation of the Double Ratchet.
- We specced it back in 2015: [https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/olm.md](https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/olm.md)
- Unlike libsignal, Olm normally uses separate keys for identity (Curve25519) and signing (Ed25519). It doesn’t use X3DH and X25519.
- There are two Apache-licensed implementations:
  - 2016: libolm (C++11 with a C API)
  - 2022: vodozemac (Rust)
- **However, while working on DMA experiments, we have now added X3DH support to vodozemac, and so it interoperates with libsignal.** We’ve calling the new dialect “interolm”: [https://github.com/matrix-org/vodozemac/pull/124](https://github.com/matrix-org/vodozemac/pull/124)
Hypothetical Matrix-for-DMA architecture

Matrix client (e.g. Element X Android)
- Normal Matrix client stack (e.g. matrix-rust-sdk)
- Normal Matrix events
- vodozemac in X3DH mode
- common content format (e.g. Matrix events as protobuf)

Matrix homeserver (e.g. Synapse)
- Client Server API
- Application Service API
- MSC3983 - Sending OTK claims to ASes
- MSC3984 - Sending E2EE key reqs to ASes

Protocol Converter Application Service (i.e. E2EE-capable Bridge)

DMA Gatekeeper Service

DMA gatekeeper client
- Existing client stack
- Existing signalling protocol
- libsignal
- common content format (e.g. Matrix events as protobuf)
Hypothetical Matrix-for-DMA architecture

Gatekeeper 1

Gatekeeper clients

Matrix<>GK1 Protocol Converter AS

Matrix clients

Matrix<>GK2 Protocol Converter AS

Gatekeeper 2
Yes, this could work.

We’ve now done experimental implementations with WhatsApp as a not-so-hypothetical gatekeeper & it seems viable, complete with E2EE.

However, we don’t yet know what will happen come March.

There are some challenges:

⊙ What sort of permissions would be needed for someone on Matrix to use such a protocol converter? (Would the organisation running the Matrix server have to Request access to the gatekeeper under DMA Article 7?)

⊙ Would the end-user device have to expose a stable identifier (e.g. an obfuscated IP address) to the gatekeeper to help with anti-spam?

⊙ Group chat is unsolved (but not in scope of DMA until 2026). As a first cut, one could just fan out DR session (like Matrix did before Megolm).
Linearized Matrix

- DMA doesn’t mandate decentralised conversation history, so gatekeepers might see Matrix as overkill when implementing it natively.
- Is there a lighter architecture that could work?
- What if we had a protocol which was compatible with Matrix, but skipped the complexities of state resolution and full decentralization (knowing that it can be upgraded to full decentralized Matrix when needed)?
  - Same old Matrix events and power levels, etc - but stored in a linked list rather than a DAG, with a hub and spoke server topology.
  - When linked to normal Matrix, the server which does the linking handles the decentralisation.
  - Example implementation in https://github.com/matrix-org/eigen-server
Hypothetical Linearized Matrix concept
(N.B. not real gatekeepers!)

Matrix

Matrix clients

Matrix Homeservers

Hub server
(for the conversation)

DMA interoperating providers

Network view for a given conversation
Overall connectivity
Introducing MIMI

- MIMI is the More Instant Messaging Interoperability Working Group in IETF
- Started by folks from the MLS (Messaging Layer Security) working group
- Seeks to define a long-term protocol specifically for DMA interoperability.
- We’ve been participating since the outset (IETF114 in Philadelphia, Jun 2022)
- At first proposed Matrix - rejected as decentralised rooms seen as overkill.
- Then proposed Linearized Matrix - rejected due to concerns about it still being too rich, e.g:
  - Does DMA even need message history? does it need state events?
  - Why have an event DAG at all, linearized or otherwise?
  - Why do you need auth events?
- Big debate over whether MIMI should support interoperability with today’s protocols, or hard-code the design to use MLS for encryption.
MIMI drafts

- Formed a Design Team (Matrix, Cisco, Google, Wire, Phoenix, Wickr) to try to build something from the ground up which would provide an “on-ramp” from Double Ratchet (DR) to MLS
  - To provide interop with today’s real-world DR platforms (e.g. Matrix!)
  - To act as a low-friction way to steer everyone to talk MLS long-term.
- If you have MLS, it uses it to synchronise state over the servers.
- If you don’t have MLS, you’d need something like a Matrix room graph.
- The layering ended up being over-complex, though, and there’s now a new proposal from Wire at [https://github.com/bifurcation/mimi-protocol](https://github.com/bifurcation/mimi-protocol)
- Currently trying to merge the two drafts together (yay, teamwork.)
- Meanwhile, to solve today’s DMA challenges, we’re using plain old Matrix.
What comes next?

- No idea :D
- Let’s see what DMA APIs Meta ships on March 7th
- Element looks to have been the first organisation to implement against them, so whatever happens, hopefully it’ll involve Matrix!
We need help!!
FRIENDS DON’T LET FRIENDS
USE PROPRIETARY CHAT SERVICES

• If you benefit commercially from Matrix - **PLEASE** financially support the Foundation
• Run a server (or get an enterprise one from Element)
• Build bridges and bots to your services!
• Build your cool new project on Matrix!
• Follow @matrix@mastodon.matrix.org & spread the word

https://matrix.org/membership/
Thank you!

@matthew:matrix.org
matthew@matrix.org
https://matrix.org
@matrixdotorg
@matrix@mastodon.matrix.org
@matrix.org