Diving into Decentralised Communication with 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
Mission: to create a global decentralised encrypted comms network that provides an open platform for real-time communication.
No single party owns your conversations.

Conversations are shared over all participants.
Matrix Architecture

- Clients
- Home Servers
- Application Servers
- Identity Servers
What do you get in the spec?

- Decentralised conversation history
- Group Messaging (and 1:1)
- End-to-end Encryption
- VoIP signalling for WebRTC
- Server-side push notification rules
- Server-side search
- Read receipts, Typing Notifs, Presence
- Synchronised read state and unread counts
- Decentralised content repository
- “Account data” for users per room
Matrix Ecosystem

The Matrix Specification (Client/Server API)

client-side

- Matrix Web Console
- Matrix iOS Console
- MatrixKit (iOS)
- “Riot X”
- matrix-react-sdk
- matrix-angular-sdk
- matrix-ios-sdk
- matrix-js-sdk
- matrix-sdk-android-rx
- matrix-sdk-android
- matrix-sdk-android
- matrix-sdk-android
- matrix-sdk-android

server-side

- Synapse (1st gen Matrix Server)
- Dendrite (2nd gen Server)
- Matrix Application Services and Bridges
- Other Servers: Ruma (Rust), jeon (Java)...
- Other Services: Bridges, Bots, Integs...

Other Clients:
- Quaternion (Qt/C++)
- gomuks (CLI/go)
- Fractal (Gtk+/Rust)
- Seaglass (macOS)
- nheko-reborn
- weechat-matrix
- ...and many many more

Other Services:
- Dendrite (2nd gen Server)
- Matrix Application Services and Bridges
- Other Servers: Ruma (Rust), jeon (Java)...
- Other Services: Bridges, Bots, Integs…
Daily active users on Matrix
Matrix as compared to XMPP

- **Completely** different philosophy & architecture:
  - A single, monolithic, consistent, spec.
  - Different primitives:
    - Syncing **decentralised conversation history** (not message passing / pubsub)
    - Group conversation as a first class citizen
    - E2E crypto as a first class citizen
  - HTTP+JSON as the baseline API (**but supporting other efficient transports too**)!
  - Core focus on defragmentation and bridging (hence the name “matrix”).
Community Status

- ~13.5M global visible accounts
- ~5.0M messages per day
- ~4.3M chatrooms
- ~40,000 federated servers
- ~3500 msgs/s out, ~35 msgs/s in on Matrix.org
- ~500 projects building on Matrix
- ~100 companies building on Matrix
- ~4 governments deploying Matrix
Demo
The Client-Server API

To send a message:

curl -XPOST -d '{"msgtype":"m.text", "body":"hello"}' "https://alice.com:8448/_matrix/client/api/r0/rooms/$ROOMID/send/m.room.message?access_token=$TOKEN"

{
    "event_id": "YUwRidLecu"
}
The Client-Server API

To receive a message:

```json
curl "https://alice.com:8448/_matrix/client/api/r0/sync?access_token=$TOKEN"
{
    "rooms": {
        "join": {
            "!SjPTfpm1qqzPmECFi:bellerophon": {
                "timeline": {
                    "events": [
                        {
                            "type": "m.room.message",
                            "sender": "@matthew:bellerophon",
                            "content": {
                                "msgtype": "m.text",
                                "body": "test"
                            },
                            "event_id": "$15582798620qPlCO:bellerophon",
                            "origin_server_ts": 1558279862446,
                            "unsigned": {
                                "age": 5006654475
                            }
                        }
                    ]
                }
            }
        }
    }
}
"next_batch": "s281_8540_0_144_235_1_155_103_1"
}``
The Server-Server API

curl -XPOST -H ‘Authorization: X-Matrix origin=matrix.org,key="898be4..",sig="j7JXfIcPFdW11p
  "ts": 1413414391521,
  "origin": "matrix.org",
  "destination": "alice.com",
  "prev_ids": [ 
    "e1da392e61898be4d2009b9fece5325"
  ],
  "pdus": [ 
    { 
      "age": 314,
      "content": { 
        "body": "hello world",
        "msgtype": "m.text"
      },
      "context": "!fkILCfBTHtNfgkP:matrix.org",
      "depth": 26,
      "hashes": { 
        "sha256": "MqVORjmjauxBDBzSyN2+Yu+KJxw8oxrrJyuPBNpELs"
      },
      "is_state": false,
      "origin": "matrix.org",
      "pdu_id": "rKQFuZQawa",
      "pdu_type": "m.room.message",
      "prev_pdus": [ 
        { 
          "PaBNREEuZj",
          "matrix.org"
        } ]
    },
    "signatures": { 
      "matrix.org": { 
        "ed25519:auto": "j2XTwAH/7E2bzHf1fG8xjoHGoS1+j7JXfIcPFdW11dp2c+JJPnHDIzRha75oJ71g7UM+CnhNAayHwZsUY3Ag"
      } }},
  "origin_server_ts": 1413414391521,
  "user_id": "@matthew:matrix.org"
}]
' https://alice.com:8448/_matrix/federation/v1/send/916d630ea616342b42e98a3be0b74113
Rooms as Directed Acyclic Graphs

- Rooms in Matrix are expressed as a Directed Acyclic Graph (DAG) of messages called ‘events’.
- Very heavily inspired by Git and serves the same purpose.
- Every room begins with an “m.room.create” event.
- Each new event points to the preceding event(s) in the room.
- Each event includes a cryptographic signature which covers the preceding events in the DAG, thus asserting integrity (like a blockchain).
- All events update the timeline of the room.
- Some events can also update key-value state in the room (“state event”) – e.g. room name, membership, topic, etc.
Rooms as Directed Acyclic Graphs

• Rooms DAGs are replicated over all participating servers with eventual consistency
State resolution

• Every time we send a state event in a room (e.g. a membership change), we have to reconcile our copy of the room with everyone else’s.

• This is Matrix’s main technical novelty

• Black arrows point to parent events

• Red arrows are the “auth DAG” – used to show the ordering of semantically important state events.
MSC1442 State resolution

• State resolution is the fundamental algorithm for merging two copies of a DAG together while preserving the room’s integrity and security.

• Current version was finalized in Feb 2019, and has no known defects.

• Relies on ordering events by the “auth DAG” with appropriate tie-breaking rules.

• Deliberately avoids servers having to possess a full copy of the DAG (for scalability)
E2E Encryption Goals

• Configurable *trade-off* between privacy and *usability* per room.
  • Sometimes you want Perfect Forward Secrecy…
  • ...but sometimes you want to replay history.
• Encrypt & trust *per-device*, not per-user.
• Support *big rooms* (thousands of devices)
• Encrypt non-public rooms by default
• Be supported on all Matrix clients.
High level overview

• Two mechanisms at work:
  
  • **Olm** – a Double Ratchet implementation
    • provides a secure channel between two devices
    • used mainly for syncing key data
  
  • **Megolm** - a new ratchet that encrypts a sender’s messages for a **group** of receivers
    • Ratchet state is shared to receivers 1:1 over Olm
    • Ratchets can be replaced to seal history
    • Ratchets can be fast-forwarded to share selective history
E2EE Device Verification

• All this E2EE is useless if you don’t know whether you’re talking to the right user or an attacker.

• The initial stop-gap solution was to ask all users to manually compare all the key fingerprints of each user’s device:

![Verify device dialog box]

- To verify that this device can be trusted, please contact its owner using some other means (e.g., in person or a phone call) and ask them whether the key they see in their User Settings for this device matches the key below:

  - **Device name:** https://riot.im/develop/ via Chrome on Mac OS
  - **Device ID:** CETUCRADOM
  - **Device key:** yGQ0 ox0u bReC weGe uGJ m0j ywEA T1+00 BHUK yGPK kyc

  If it matches, press the verify button below. If it doesn’t, then someone else is intercepting this device and you probably want to press the blacklist button instead.

  In future this verification process will be more sophisticated.
MSC1267 – Interactive key verification using short authentication strings

• Comparing long fingerprints is very tedious
• Replaced in 2019 by calculating a short shared secret which is compared via emoji or numbers:
MSC1756 – Cross-signing

We are currently introducing cross-signing:

• The ability to publish attestations of your own devices so users only need to verify you once.

• A trusted user could also be used to vouch for others to simplify trust.

• Persists your trust decisions serverside.

• Switches to scanning QR codes by default at login to validate.

• Aiming to land in Matrix in Jan 2020.
Remaining challenges

• In a decentralized ecosystem it’s possible to not know what devices (or users!) are actually in a room when you encrypt a message.
• We have “keyshare requests” to let users request missing keys to solve this edge case.
• Currently improving error messages when this happens to better educate the user.
• Rare, but possible – particularly if the network is unreliable.
MLS (Messaging Layer Security)

- Olm/Megolm are good, but setting up new encryption sessions is $O(N)$ complexity over the devices in the room.
- MLS is IETF’s next generation E2E standard, which offers $O(\log N)$ complexity by grouping devices into a tree for key exchange rather than full mesh.
- However, it requires a centralized approach to order the devices consistently into a tree.
- We are currently working on “Decentralised MLS”.
- [https://matrix.uhoreg.ca/mls/ordering.html](https://matrix.uhoreg.ca/mls/ordering.html)
Metadata Privacy

- Matrix does not protect metadata currently; server admins can see who you talk to & when (but not what). If you need this today, look at Ricochet or Cwtch or Vuvuzela etc.

- Protecting metadata is incompatible with bridging.

- However, peer-to-peer homeservers could run clientside, tunnelling traffic over overlay servers with dead-drops for store-and-forward.

- We started on P2P seriously in Dec 2019
Peer-to-Peer Matrix

After P2P...

BS API (HTTPS+JSON)

SS API (HTTPS+JSON over libp2p WebRTC)
Matrix in DIL Networks
Compression & Encryption

We tried several combinations of encryption and compression protocols before finding one achieving the target size:

<table>
<thead>
<tr>
<th>Protocol Combination</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.1 + JSON + GZIP</td>
<td>~ 8kB</td>
</tr>
<tr>
<td>HTTP/2 + JSON + GZIP</td>
<td>~ 7kB</td>
</tr>
<tr>
<td>HTTP/3 (QUIC) + JSON</td>
<td>~ 6kB</td>
</tr>
<tr>
<td>COAP + CBOR + DTLS 1.2</td>
<td>~ 500B</td>
</tr>
<tr>
<td>COAP + CBOR + Flate + Noise</td>
<td>~ 100B</td>
</tr>
</tbody>
</table>
Matrix in DIL Networks
Default Architecture

Client

Client

Server

Server

Client

JSON
HTTPS
TCP
IP
Ethernet

~ 7kB
Matrix in DIL Networks
Low-bandwidth Architecture

Client

CoAP Proxy

JSON
HTTPS
TCP
IP
Ethernet

Server

CoAP Proxy

7kB

Server

CoAP Proxy

Client

CoAP Proxy

CBOR
CoAP
FLATE
NOISE
UDP
IP
Ethernet

50B
108B

65x improvement!
Matrix in DIL Networks
Meshsim

• We model Disconnected / Intermittent / Limited Bandwidth networks using “meshsim” – a network simulator environment which lets you directly model and manipulate custom network topologies and conditions on the fly.

• Uses real Matrix homeservers running under Docker.

• Directly observe the performance and behaviour of Matrix clients during network partitions and adversarial conditions.
What’s next?

• End-to-end Encryption by default (Feb 2020)
• E2EE Cross-signing support
  • E2EE Search
  • Key loss mitigations
• RiotX 1.0 (Feb 2020)
• Synapse sharding per room
• Reworking communities
• Extensible profiles
• Decentralised accounts (MSC1228) and P2P matrix
• Decentralised reputation
Thank you!

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