Linearized Matrix & Channel State

MIMI Interim - June 7, 2023

Travis Ralston
@travis:t2l.io | travisr@matrix.org

Architecture

Hub servers are responsible for linear conversation history on behalf of other servers in the room.

Participant servers are every other non-hub server. Does not need to hold history.

Rooms hold **events** that carry meaning in-context. Events over the wire between servers are called **PDUs** (Persistent Data Units).

Architecture

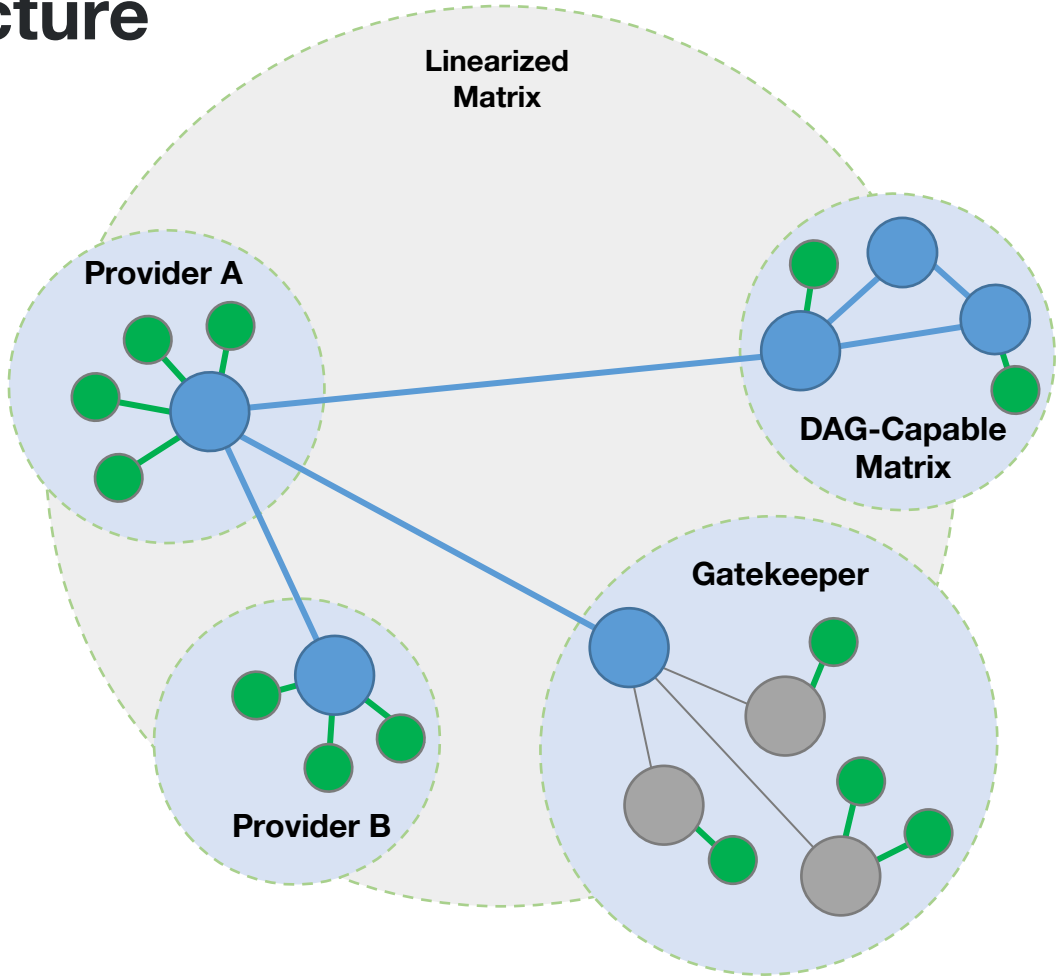
All of a room's behaviour is defined under a **room version**.

Users are identified like @alice:example.org and can have zero or more devices.

Devices are individual sessions/clients belonging to a user. MLS tracks devices in the group.

Architecture

For a given room:



Events

All changes within a room are sent as events. The event ID is the reference hash of the event itself, protecting against fraudulent events.

State events track the metadata for the room, such as *user* membership, name, join rules, power levels, etc.

MLS commits (and possibly proposals) would be sent as **room events**, or non-state events.

Definition of ACLs

The room version defines the event authorization rules. The auth rules use **power levels** alongside other semantics to determine what is allowed to be sent.

All participants need to support the entire room version. The room version is expected to be specified in an I-D (then RFC, etc). No need for a domain-specific language.

ACLs are therefore known well before a user/server can join the room.

“Channel State”

Last discussed at interim-2023-mimi-01 (May 3rd) in ekr’s slides, #14.

Matrix traditionally handles state server-side, keeping keys/devices in sync with the MLS group on clients.

MLS is capable of handling (nearly) everything within its state client-side.

Linearized Matrix can support either.

Server-side room model

MLS *only* handles device (key) membership in the group, informed by what's going on in the room.

Rules need to be created for how to engage new devices in the MLS group (who does the join? the kick?)

Servers reject events based on permissions/legitimacy.

Without cross-signing, servers can inject malicious devices to a group.

Client-side room model

Servers *only* check to ensure the event format is okay.

Within MLS, events are run against the auth rules/ACLs. Servers don't know what was accepted or rejected.

Membership model changes from per-user to per-device, likely.

Client implementation complexity goes up: conflict resolution needed.

Server-side or client-side?

To recap:

Server-side requires rules for how a user's devices are added to MLS groups.

Client-side gives better trust guarantees, but is more complex to implement.

If server-side, does device membership need to be cryptographically bound to user (room) membership?