[matrix]

pacman

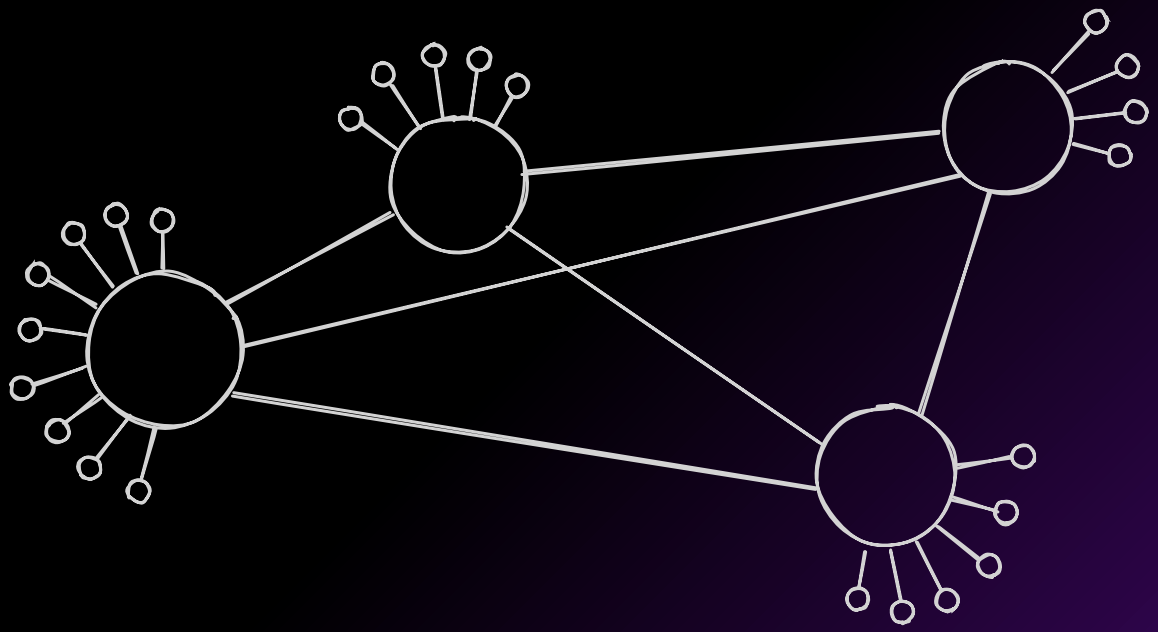
EUPLv1.2

slides

https://computerbetrug.jetzt/haecksen/matrix.pdf

pacman

what is matrix



events

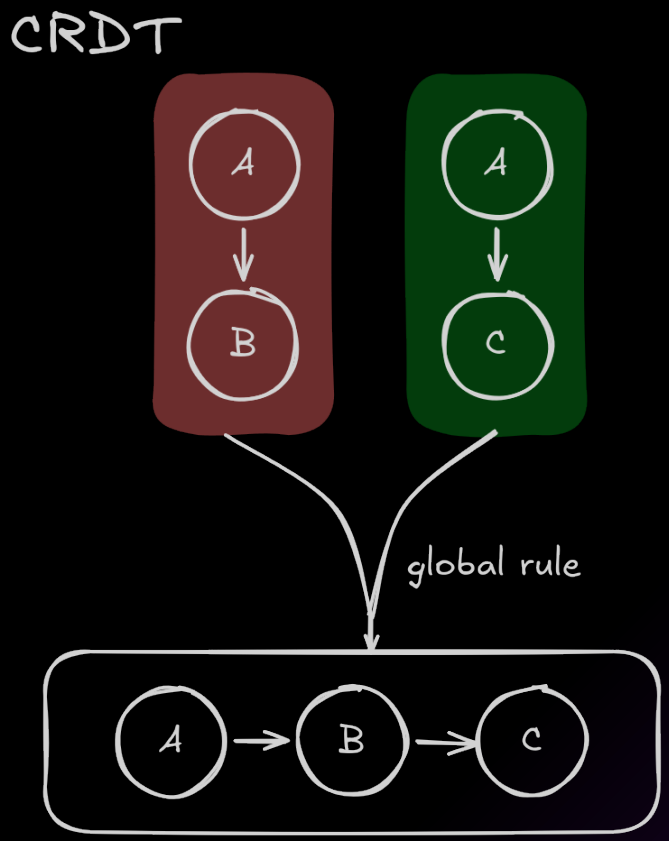
A JSON object that describes an interaction between users and their homeserver.

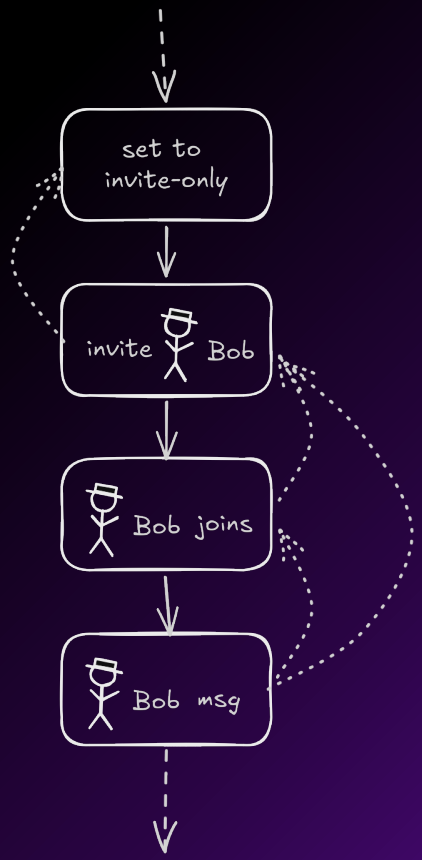
Message types follow the java package naming convention. This allows for easy transfer of arbitrary data (when both clients have access to the namespace). "`m.`" is for events defined in the matrix spec. For example `m.room.message`.

A server sends events into rooms. So a room is a series of events (JSON objects).


```
{
  "content": {
    "body": "example text message",
    "format": "org.matrix.custom.html",
    "formatted_body": "<b>example</b>",
    "msgtype": "m.text"
  },
  "event_id": "$8244...rSn:example.org",
  "origin_server_ts": 1432735824653,
  "room_id": "!DJdhl...yVU:example.org",
  "sender": "@example:example.org",
  "type": "m.room.message",
  "unsigned": {
    "age": 1234
  }
}
```

events | synchronization

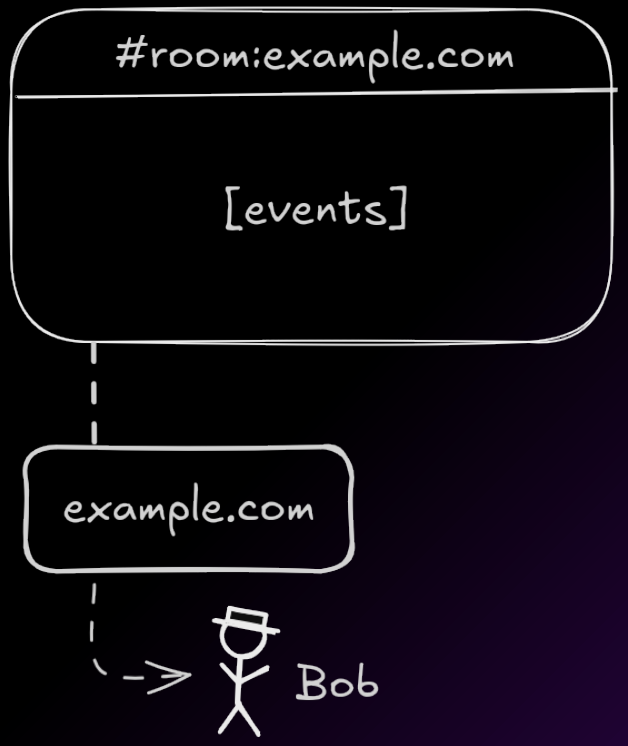




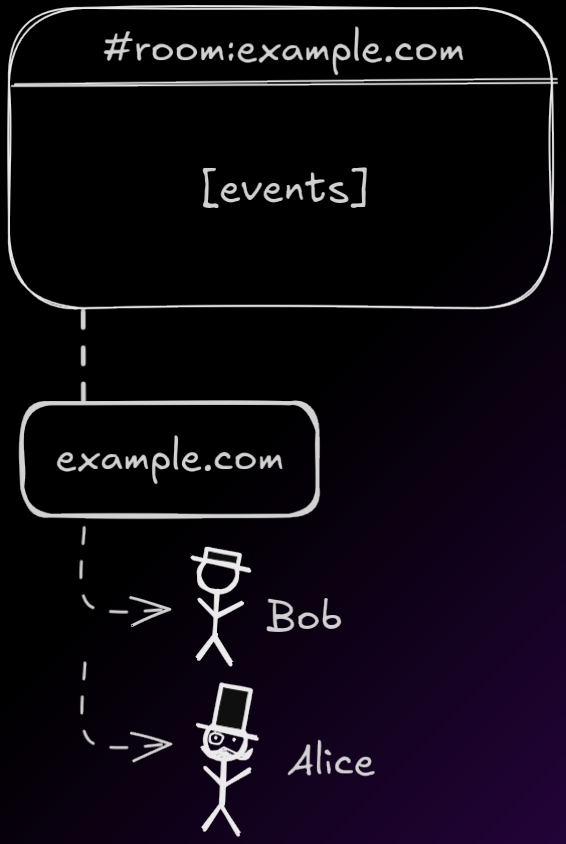
pacman

EUPLv1.2

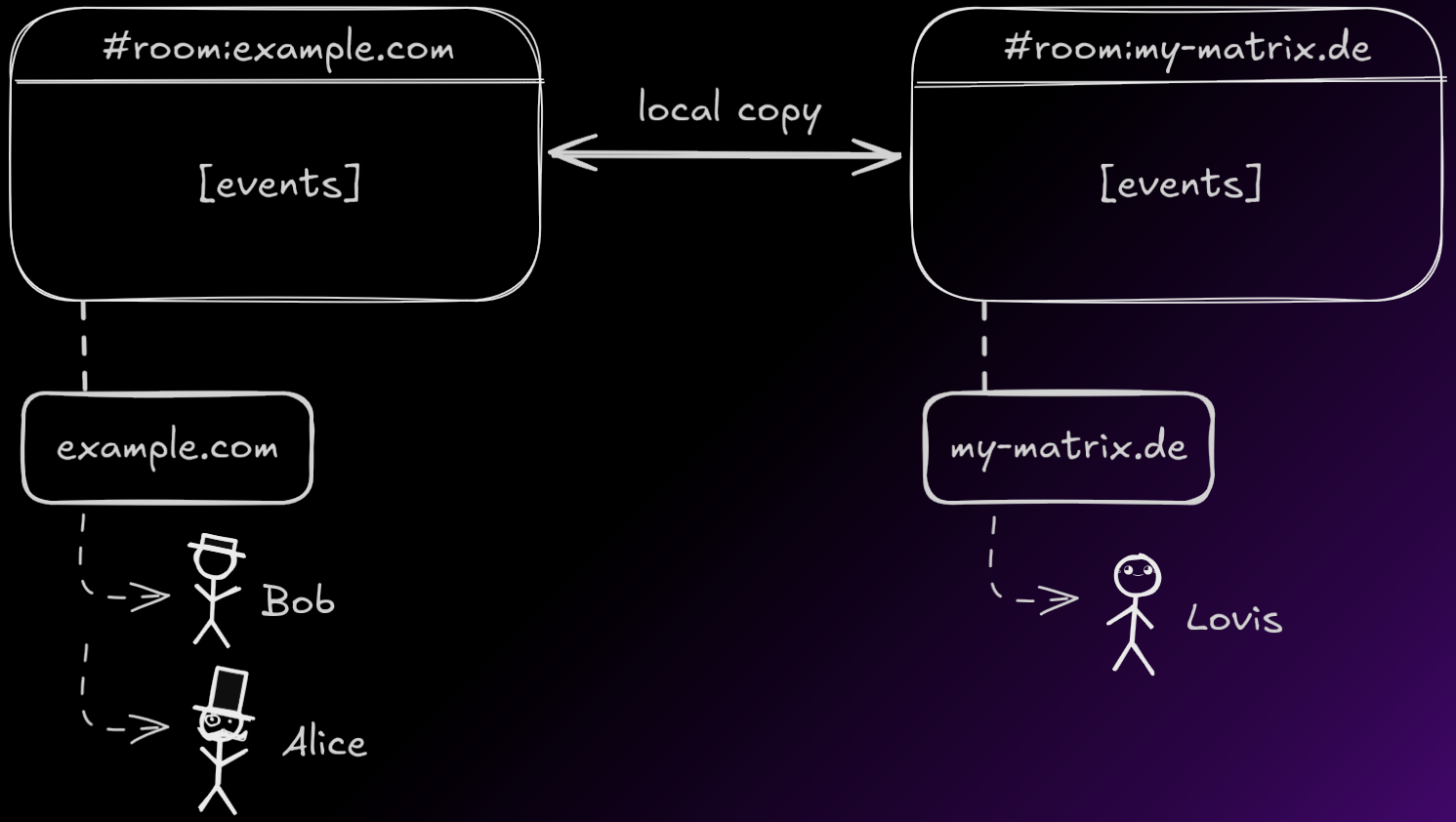
federation | rooms



federation | rooms



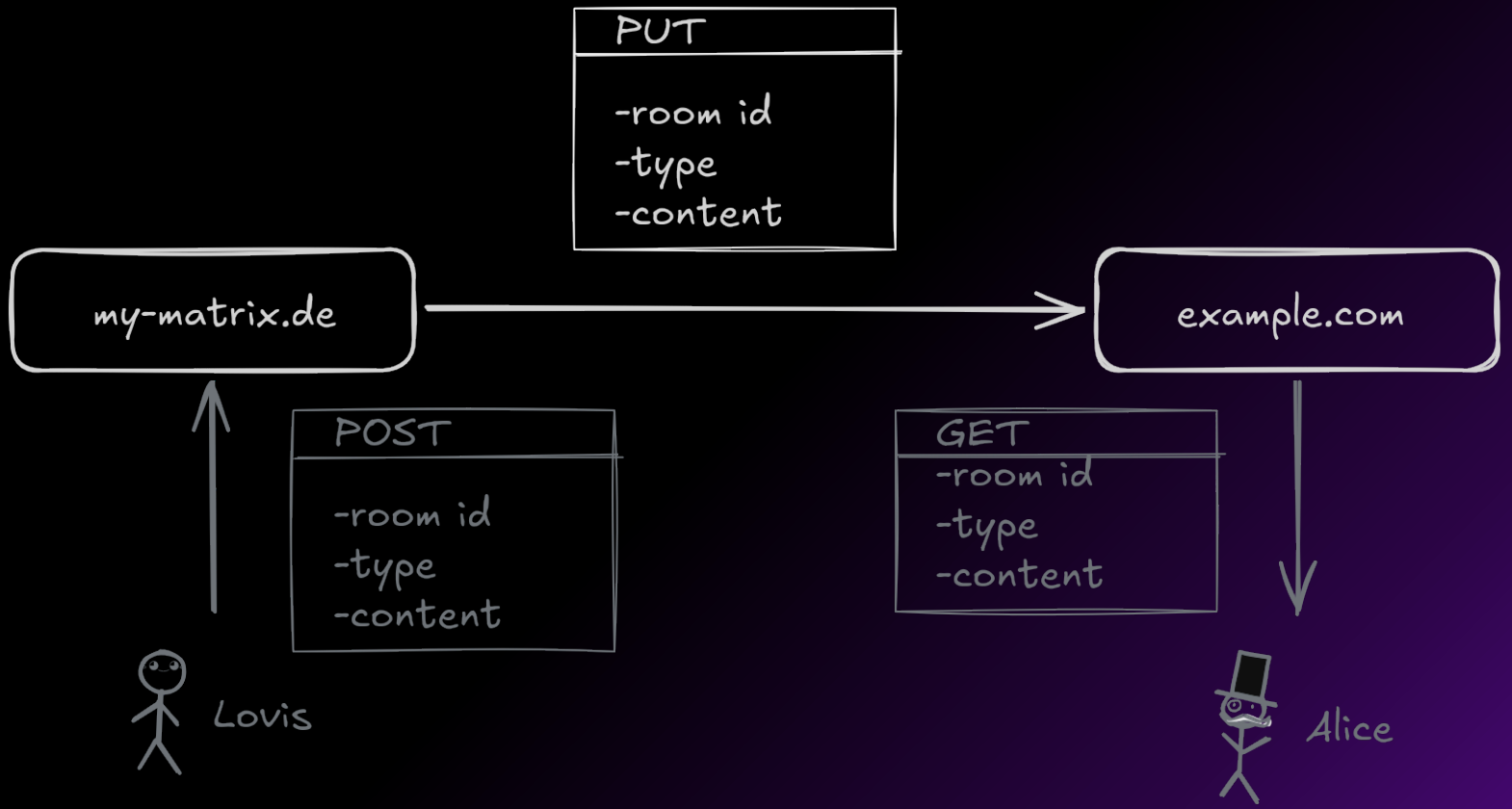
federation | rooms



pacman

EUPLv1.2

federation | rooms



e2ee

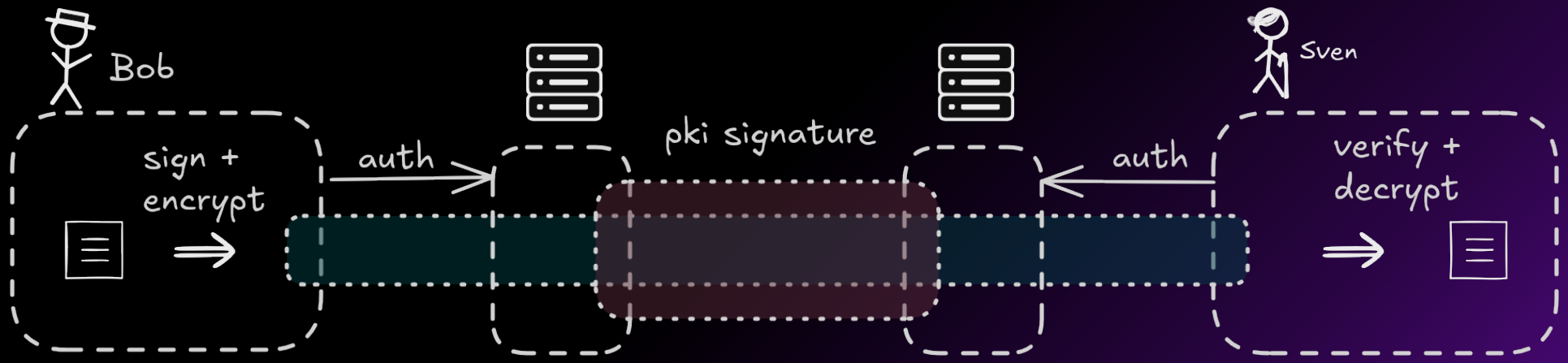
- message security (can only be read by sender and receiver)
- message integrity (cannot be edited)
- identity (verifying that a user/device is who they claim they are)

Matrix uses multiple keypairs to achieve this.
These keys are encrypted with a symmetric AES key and saved on the server. With the recovery Key they can be decrypted and used on the devices.

e2ee | identity

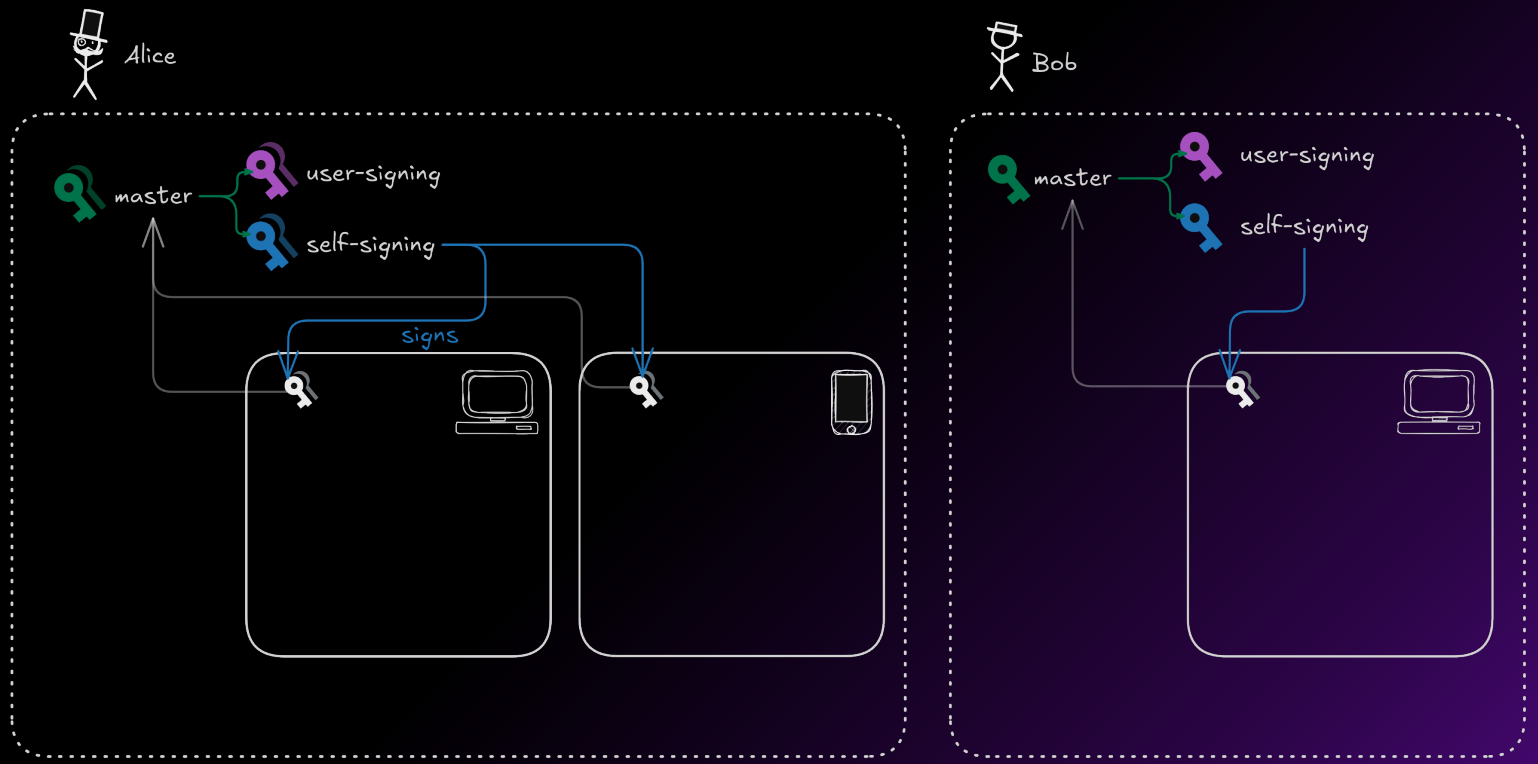
Your matrix identity is a public-private-key-pair. Other profile data is just metadata.

The identity is **not** the account; that is why you have 2 “authentication” steps. One to authenticate yourself to the server and one to prove your identity to other users.

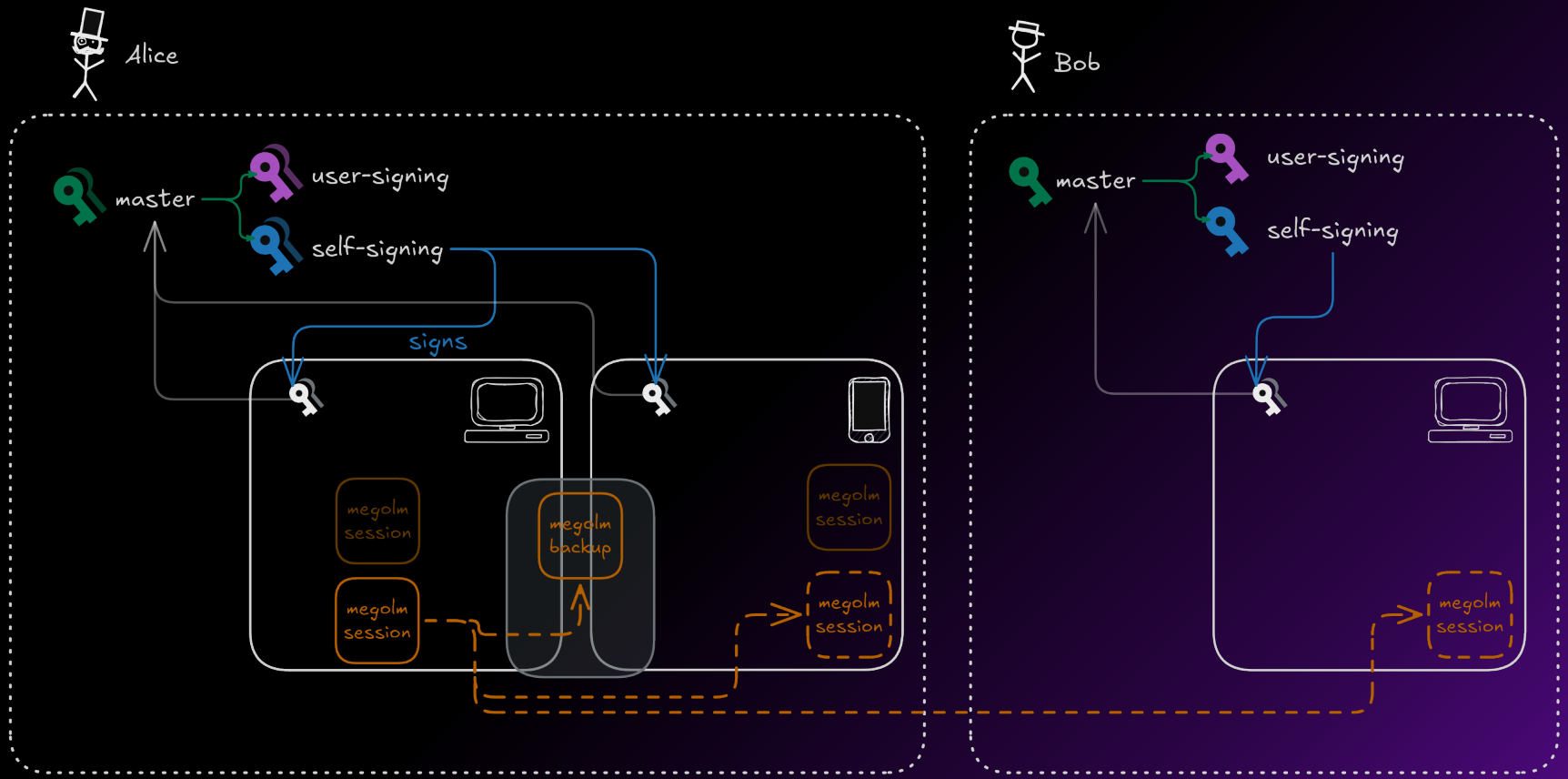


e2ee | device identity

Each device also has a public-private-key-pair.



e2ee | megolm session

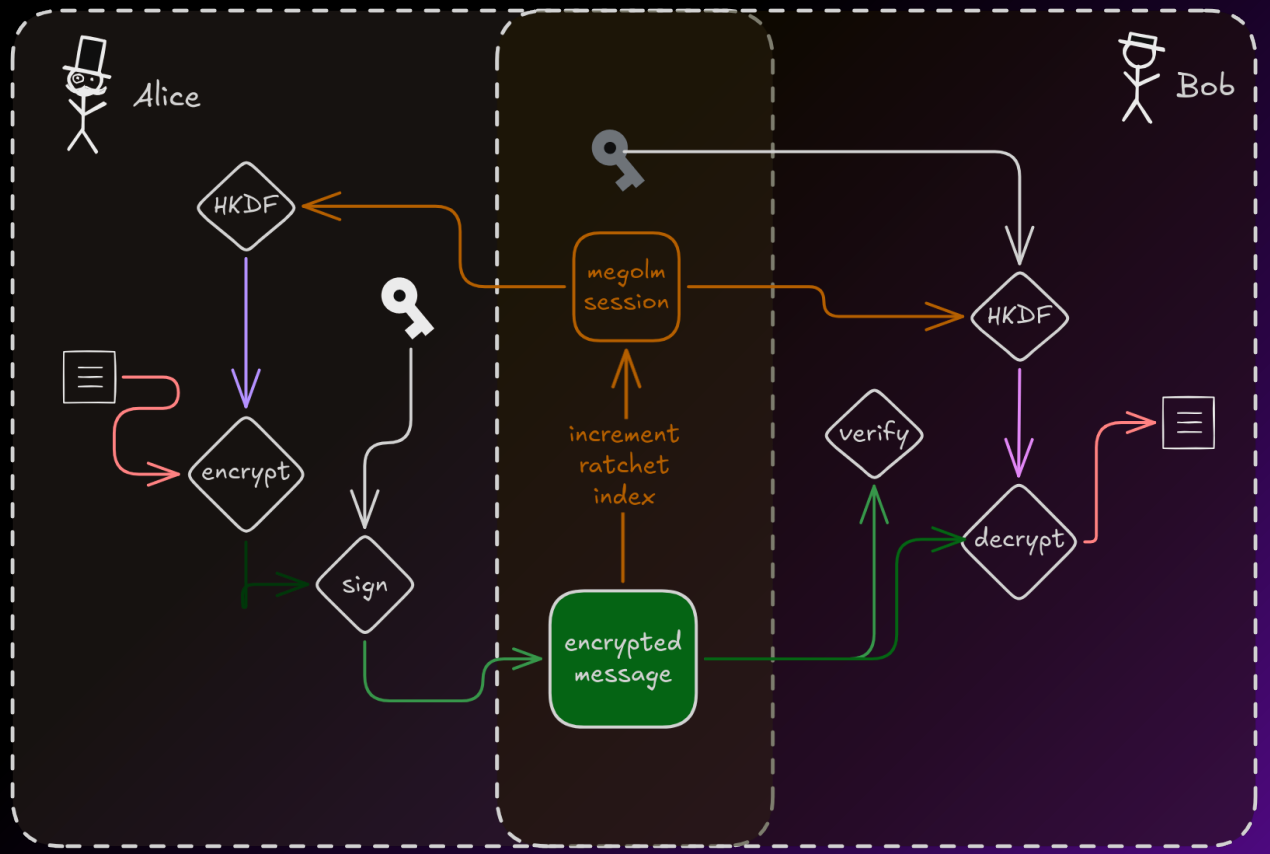


pacman

EUPLv1.2

e2ee | megolm session

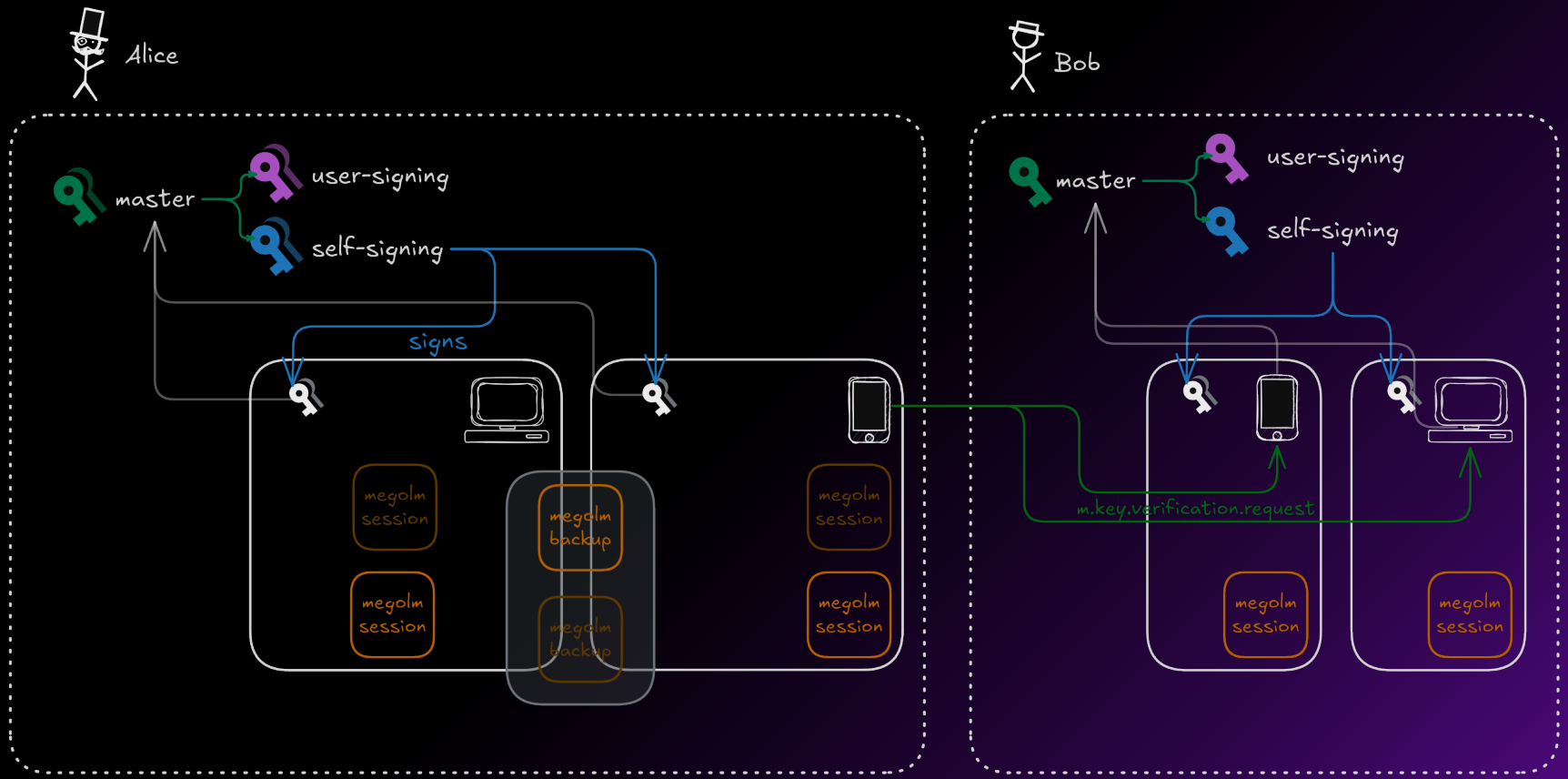
- long lived identity keys
- short lived encryption keys
- perfect forward secrecy
- closely modeled after signal
- post quantum
- small payload overhead



pacman

EUPLv1.2

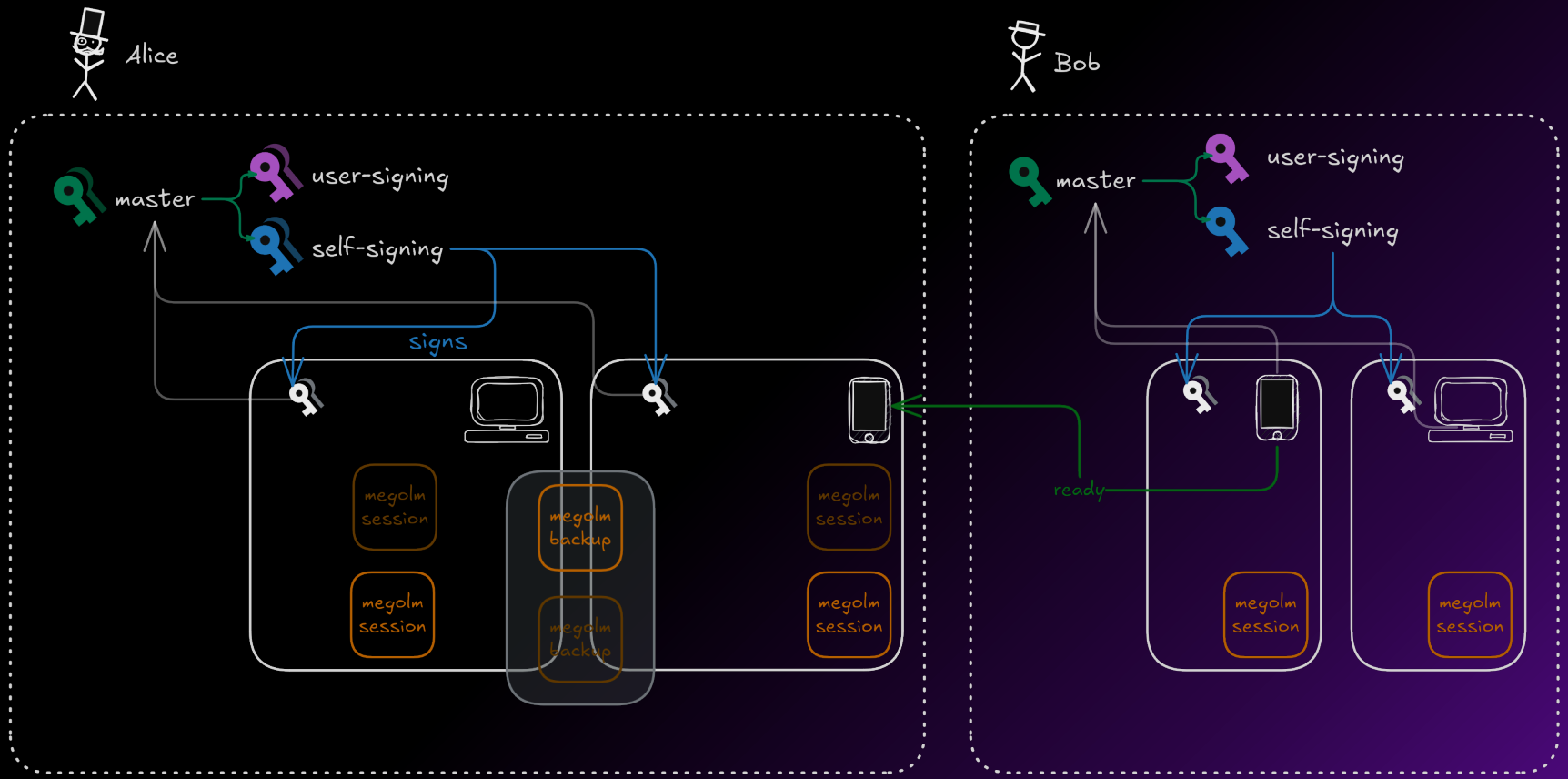
e2ee | user verification



pacman

EUPLv1.2

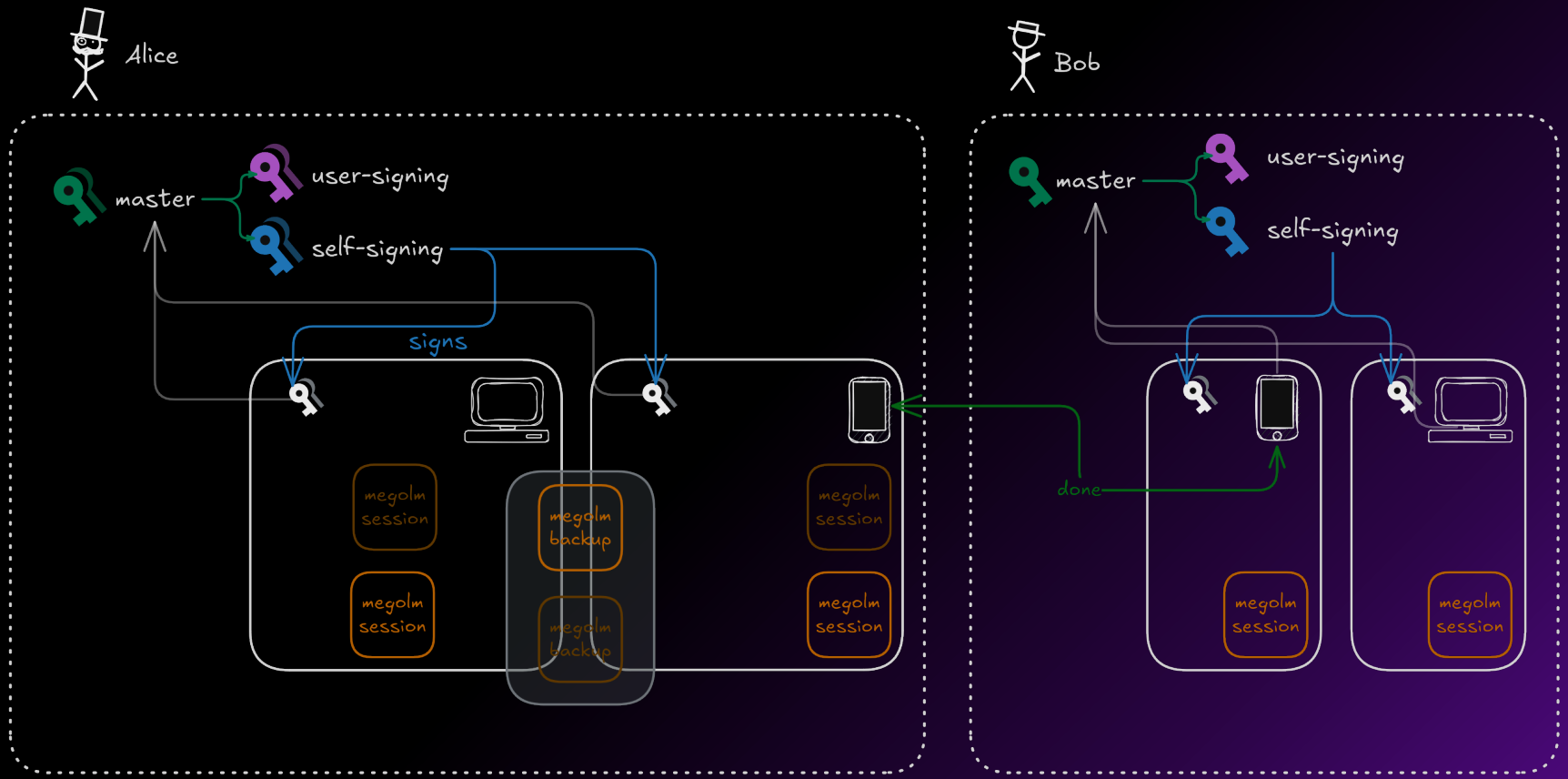
e2ee | user verification



pacman

EUPLv1.2

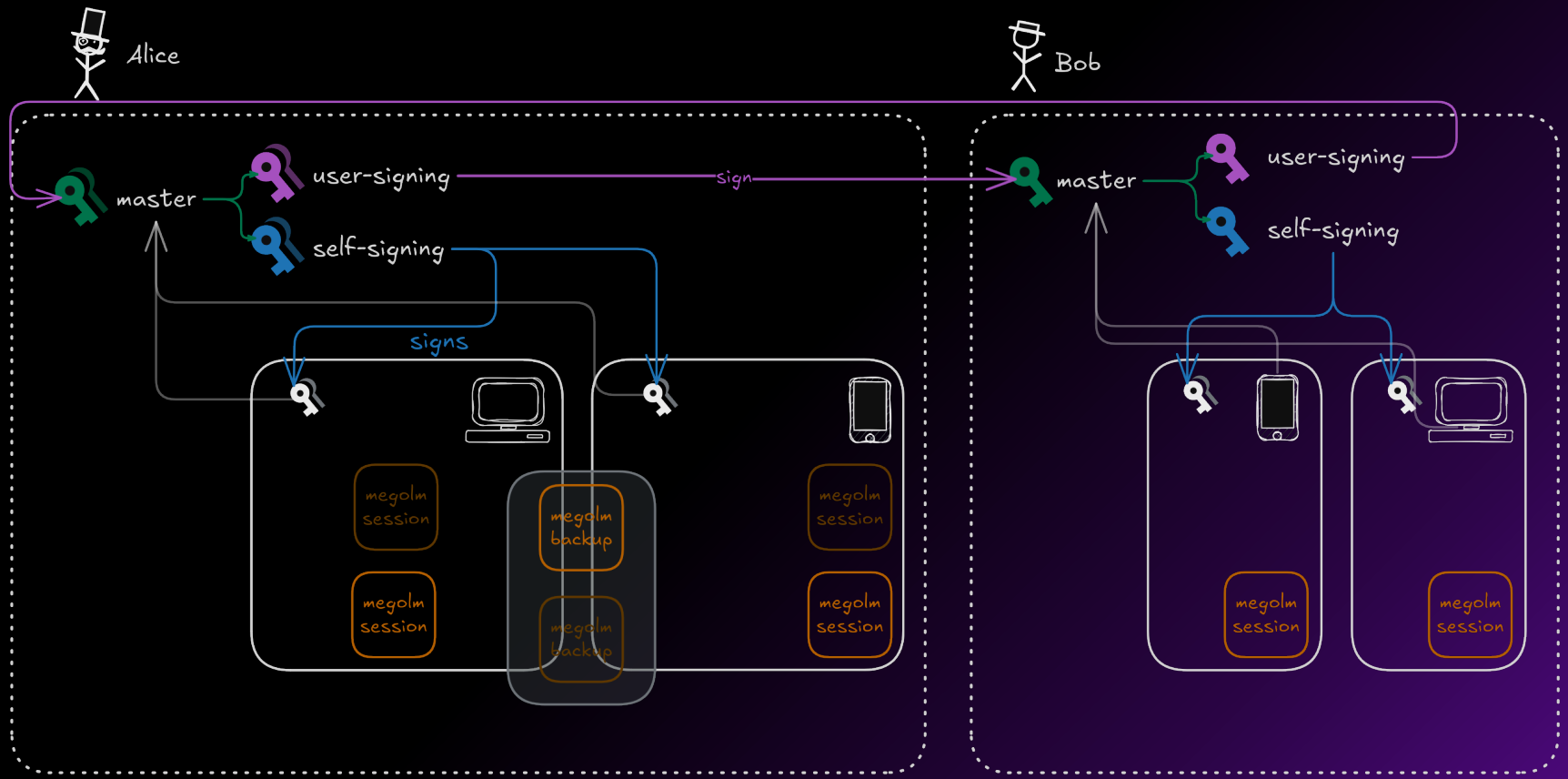
e2ee | user verification



pacman

EUPLv1.2

e2ee | user verification



pacman

EUPLv1.2

danke :)

https://computerbetrug.jetzt/haecksen/matrix.pdf

- https://spec.matrix.org/v1.18/client-server-api/
- https://matrix.org/docs/matrix-concepts/rooms_and_events/
- https://spec.matrix.org/v1.18/client-server-api/#types-of-room-events
- https://spec.matrix.org/latest/#event-graphs
- https://crdt.tech/
- https://spec.matrix.org/unstable/rooms/v10/#state-resolution
- https://spec.matrix.org/v1.18/olm-megolm/megolm/#session-setup
- https://spec.matrix.org/v1.18/client-server-api/#key-verification-framework