Karlsruhe Institute of Technology

Tokens

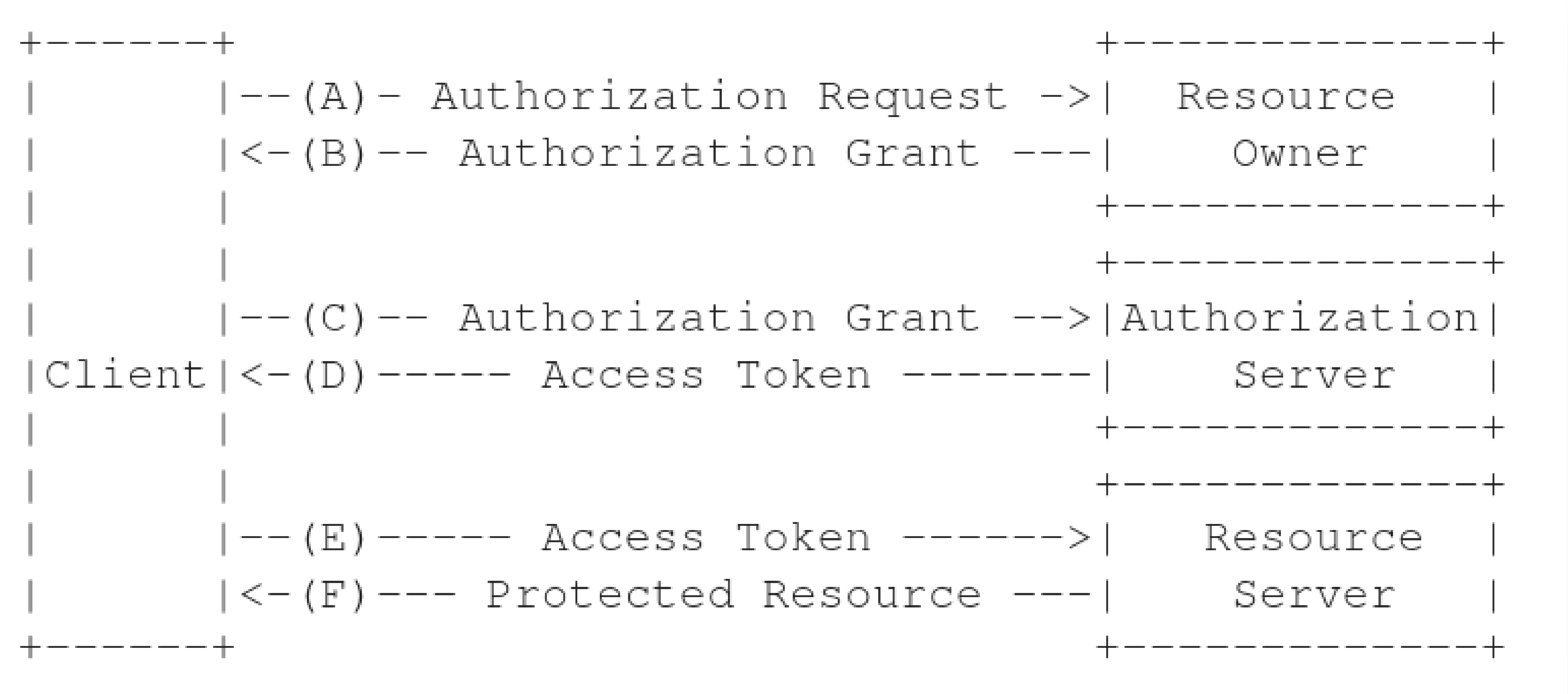
Marcus Hardt

March 2025

OpenID Connect

First: OAuth2

- Involved Parties:
  - Authorisation Server (creates the token)
  - Client (User / Browser that has the token)
  - Relying Party (Service that gets the token)

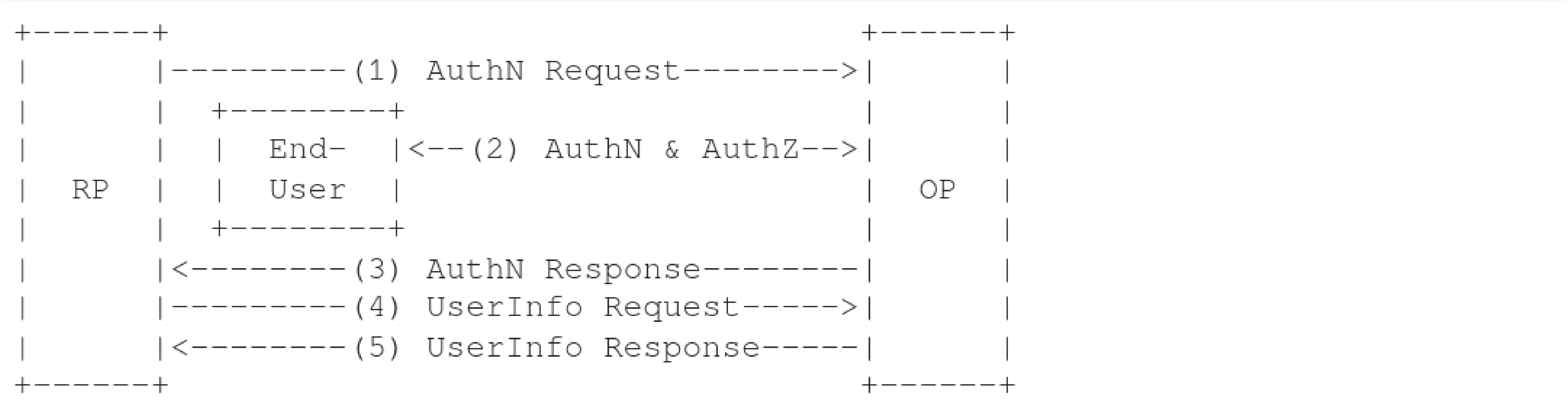


```
+------+                               +-------------+
|      |--(A)- Authorization Request ->|  Resource   |
|      |<-(B)-- Authorization Grant ---|    Owner    |
|      |                               +-------------+
|      |                               +-------------+
|      |--(C)-- Authorization Grant -->|Authorization|
|Client|<-(D)----- Access Token -------|    Server   |
|      |                               +-------------+
|      |                               +-------------+
|      |--(E)----- Access Token ------>|   Resource  |
|      |<-(F)--- Protected Resource ---|    Server   |
+------+                               +-------------+
```

https://datatracker.ietf.org/doc/html/rfc6749

OpenID Connect

- OIDC adds information about the user
  - `userinfo` endpoint is an OAuth2 protected resource
- And renames some things:
  - RP: Relying Party => “The Service” (RS in Oauth2), also client somehow...
  - OP: OpenID Provider => “Where tokens are made” (AS in Oauth2)



```
+------+                                   +------+
|      |---------(1) AuthN Request-------->|      |
|      |  +--------+                       |      |
|      |  |  End-  |<--(2) AuthN & AuthZ-->|      |
|  RP  |  |  User  |                       |  OP  |
|      |  +--------+                       |      |
|      |<--------(3) AuthN Response--------|      |
|      |---------(4) UserInfo Request----->|      |
|      |<--------(5) UserInfo Response-----|      |
+------+                                   +------+
```

Flows

i.e. ways to get “tokens”

- Authorization Code Flow
    - User points web-browser to service
    - If no session: Service redirects to OP
    - OP checks for password
        - then redirects back to service (with the authorization `code` parameter
    - Service can use `code` to get “tokens” from AS
        - Sounds complicated, but keeps tokens out of user-browser
        - Some tools (`oidc-agent`) provide the AT to the user
    - User gets a cookie, web session established, everything fine
- Device Code Flow
    - Same thing in green
- Refresh flow
    - Use one token to get a new token (Yep, this is fuzzy.... on purpose)

Token types

- Access Token (AT) (lives 5min - 24h)
  - Initial spec: “opaque string”
    - example: `0889f11abc8049e2bc70c3504207a445`
  - With RFC9068 AT contain information:
    - `sub`, `iss`, `assurance`, `iat`, `exp`, **signature**
  - Can be used to query `userinfo` endpoint at the OP (by anyone)
  - Can be used to verify the token: `token_introspection` endpoint at the OP (only clients)
- ID Token
  - JWT that contains Claims about the Authentication of an End-User by an Authorization Server
  - Bound to the `client_id` to which it was issued
- Reffresh Token (RT) (lives 1d - 1a)
  - Opaque String
  - Bound to the `client_id` to which it was issued
  - Can be used to obtain a new AT and RT from the OP

userinfo_endpoint

- How to find it?
  - You need the OP's URL https://aai.egi.eu/auth/realms/egi/.well-known/openid-configuration
  - Query it's /.well-known/openid-configuration:

    ```
    curl -s \$OP_URL/.well-known/openid_configuration
    ```
- Use it with the AT:
  - `curl -s $OP_URL/$UIEP`
  - Find much more stuff (maybe): `name`, `email`, `family_name`, ...
  - Only while AT is valid
- I guess sec folks would like some special client or endpoint, that allows them to see information about expired AT, if available. (AT typically deleted on OP after expiry for performance reasons).

Examples of token stuff

- `oidc-agent`
- `flaat-userinfo`

Who’s not confused yet?

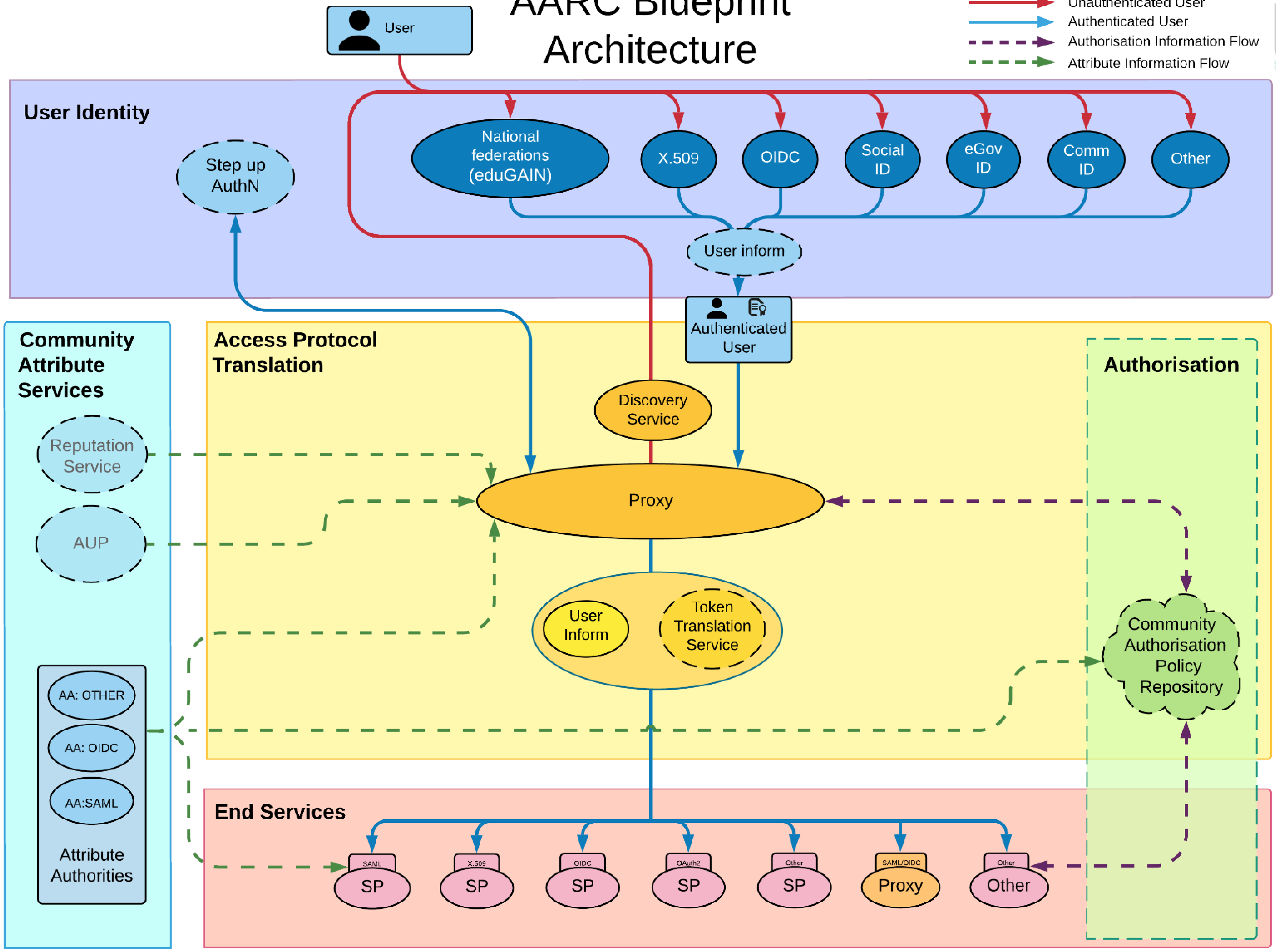
Proxies!!

Security Nightmares are made out of Proxies?

- “Proxy”: SP-IdP Proxy
  - User logs in on SP side
  - Proxy provides user data via IdP side
  - OIDC Terms: RP, OP
- Why on earth did you do this to us?
  - Address the attribute release problem
  - Manage the Community
  - Structure the field
- (Actually…)
  - We simply named what we found in the field

AARC Blueprint Architecture

Unauthenticated User
Authenticated User
Authorisation Information Flow
Attribute Information Flow



User

User Identity

Step up AuthN

National federations (eduGAIN)

X.509

OIDC

Social ID

eGov ID

Comm ID

Other

User inform

Authenticated User

Community Attribute Services

Reputation Service

AUP

AA: OTHER

AA: OIDC

AA:SAML

Attribute Authorities

Access Protocol Translation

Discovery Service

Proxy

User Inform

Token Translation Service

Authorisation

Community Authorisation Policy Repository

End Services

SAML SP

X.509 SP

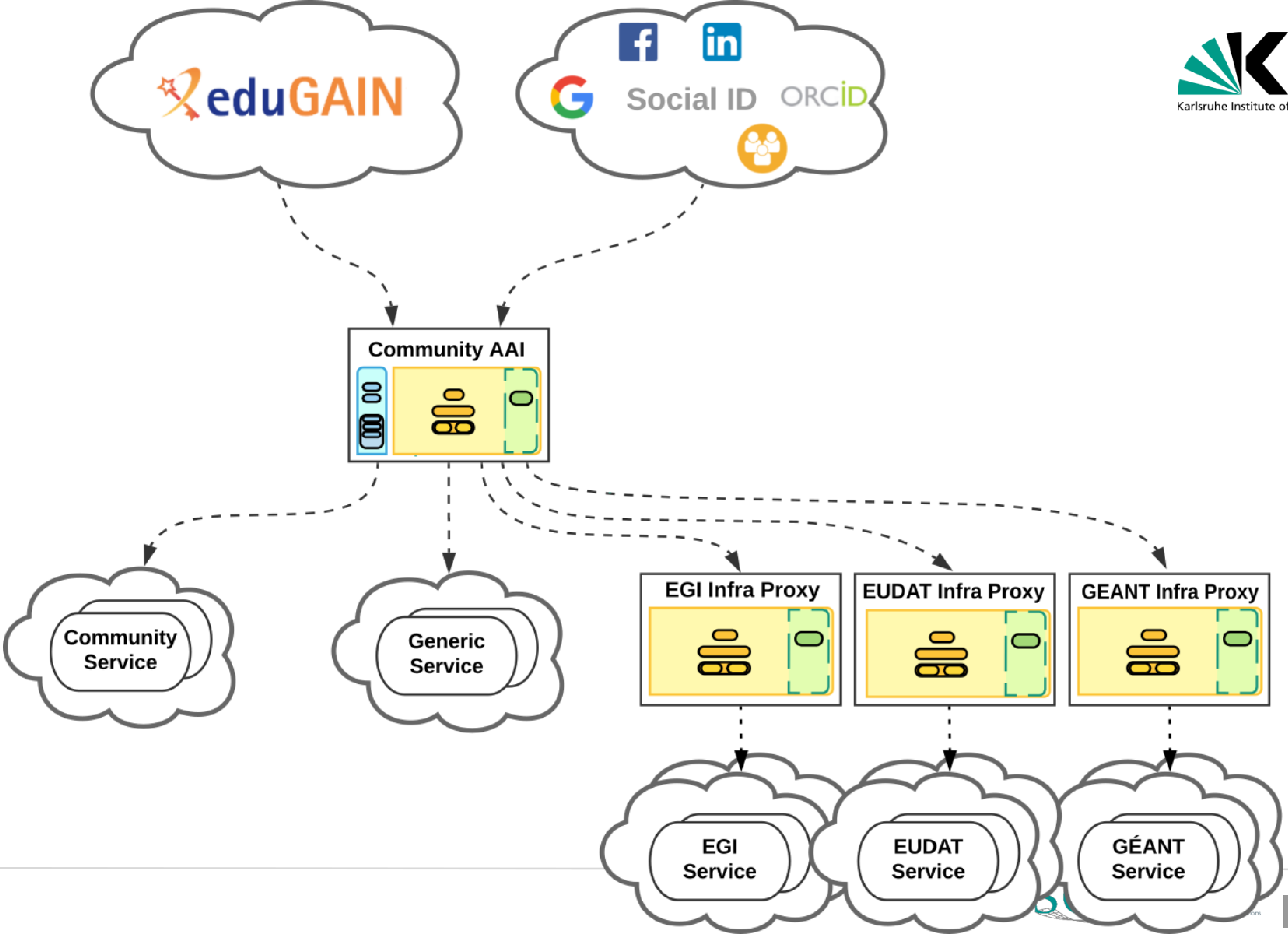
OIDC SP

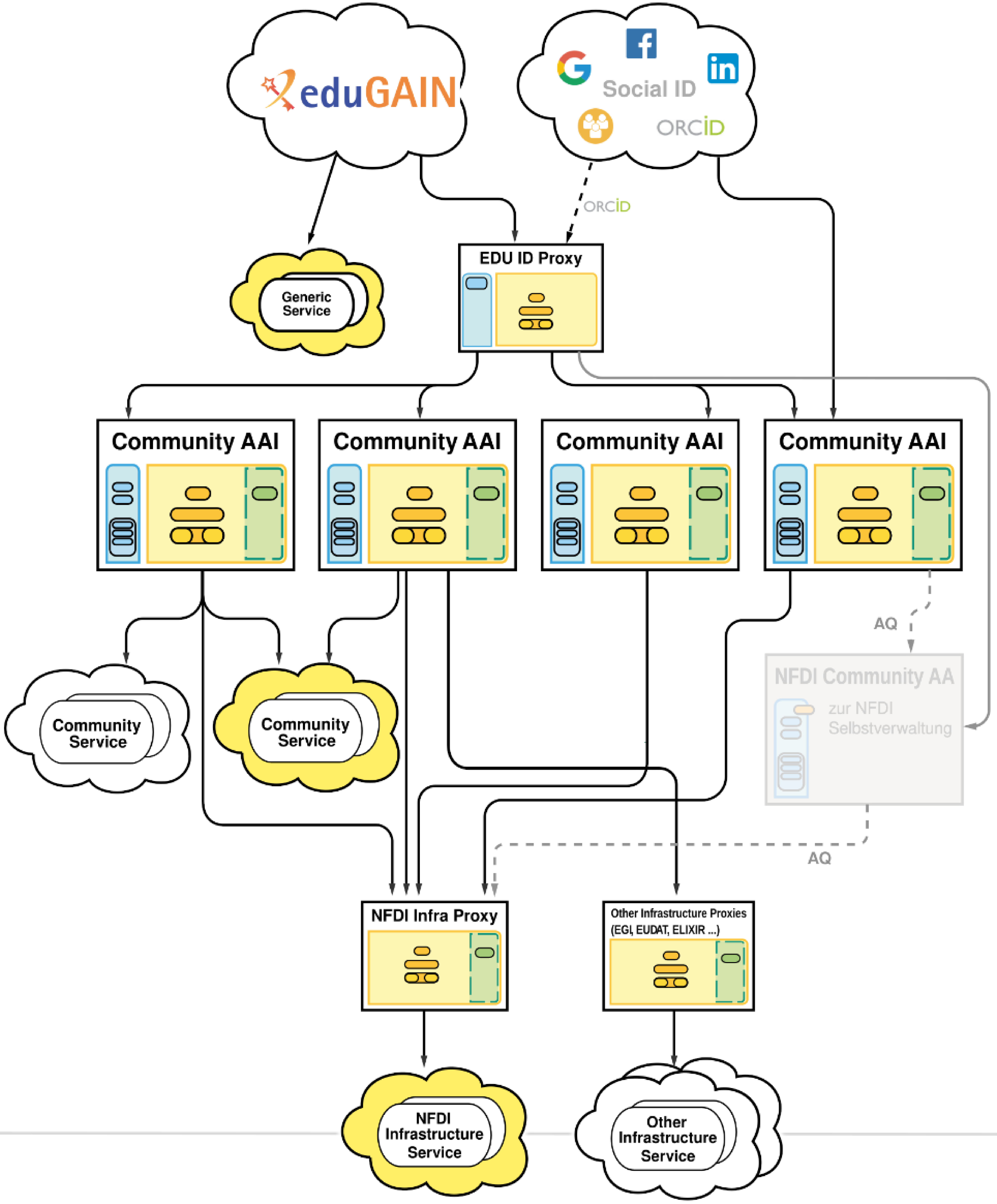
OAuth2 SP

Other SP

SAML/OIDC Proxy

Other Other


eduGAIN
Social ID
ORCID
Community AAI
Community Service
Generic Service
EGI Infra Proxy
EUDAT Infra Proxy
GEANT Infra Proxy
EGI Service
EUDAT Service
GÉANT Service


eduGAIN
Social ID
ORCID
ORCID
Generic Service
EDU ID Proxy
Community AAI
Community AAI
Community AAI
Community AAI
AQ
NFDI Community AA
zur NFDI Selbstverwaltung
Community Service
Community Service
AQ
NFDI Infra Proxy
Other Infrastructure Proxies (EGI, EUDAT, ELIXIR ...)
NFDI Infrastructure Service
Other Infrastructure Service

FAQ: How to find the user?

- My service is being abused with a given (set of) Access Tokens
  - Analyse AT:

```
for T in $(echo $TOKEN | tr '.' '\n' ); do
# echo $T \
#     | base64 -di 2>/dev/null
echo $T \
    | base64 -di 2>/dev/null \
    | jq --indent 4 2>/dev/null
done
```

```
{
  "exp": 1738950014,
  "iat": 1738946414,
  "auth_time": 1733921033,
  "jti": "a2b13273-5178-4539-a9a7-c82015a2b3f6",
  "iss": "https://aai.egi.eu/auth/realms/egi",
  "sub": "d7a53cbe3e966c53ac64fde7355956560282158ecac8f3d2c770b474862f4756@egi.eu",
  "typ": "Bearer",
  "azp": "oidc-agent",
  "session_state": "7a8df8ec-e80b-44c8-b488-673e5a5524ce",
  "scope": "openid eduperson_unique_id offline_access eduperson_scoped_affiliation
eduperson_entitlement profile email",
  "sid": "7a8df8ec-e80b-44c8-b488-673e5a5524ce",
  "eduperson_assurance": [
      "https://refeds.org/assurance/ATP/ePA-1d",
      ...
      "https://aai.egi.eu/LoA#Substantial"
  ],
  "authenticating_authorities": "[{\"name\":\"EGI Check-in\"}]",
  "authenticating_authority": "https://idp.scc.kit.edu/idp/shibboleth"
}
{
```

Important fields

- **`sub`**: the OP-Local ID of the user
- **`iss`**: the ussuer of the token: They **MUST** know more about the user

Try userinfo

- `http https://auth.didmos.nfdi-aai.de/OIDC/userinfo "Authorization: Bearer 0889f11abc8049e2bc70c3504207a445"`
- `sub`, `iss`, `email`, `name`, ...
- **`voperson_unique_id`**, **`eduperson_unique_id`**:
  - the unique identifier of the user
  - issued by the VO (or even higher up)
- **`voperson_external_id`** (optional)
  - All other IDs given to the user in this login chain
- **`eduperson_scoped_affiliation`**: might contain info about where the user is from (e.g. `faculty@kit.edu` for me)

Open Points

- Where do security folks find contact information of OP?

Fun Stuff

OMG data everywhere

https://www.youtube.com/watch?v=NAUKPq5QjL0