# Certificate Token Translation Service

Mischa Sallé

msalle@nikhef.nl

*EGI Conference, Amsterdam*

7 April 2016

# Introduction: Token Translation Service

Why do we need a Token Translation Service?

- many services (Grid and Cloud) use certificate authentication: *reliable, fast and secure when done correctly*
- certificates are complicated for end-users
- Single-Sign-on is widely available and used

→ want certificates "behind the scenes"

# Introduction: Token Translation Service

- Further prerequisites:
  - VO integration should be simple: no security experts
  - only one online CA in Europe

  → Need extra step for handling complexity and caching: *Master Portal*

- Distributed architecture:
  - $O(1)$ onlineCA, "infra" (Elixir, EGI, ...) agnostic.
  - $O(10)$ Master Portals: one per "infra", VO agnostic
  - $O(\mathrm{many})$ VO portals

# Delegation

Delegation of Authorization:

- portals need to be able to obtain (proxy)credentials: *delegation of authorization to portals*
- SAML delegation is tricky: would need at least ECP support
- OpenID-Connect (e.g. Google) much more suitable for that: easy to implement standard.

→ chosen solution by CILogon

# CILogon

- US project for federated access to online CA
- provides and runs software, using MyProxy CA as backend
- multiple scenarios
- interested in "portal delegation" scenario:

$$\textit{user login} \quad \rightarrow \quad \textit{end-entity cert on portal}$$

- Uses OpenID Connect flow
  - online CA is both OIDC provider and protected resource
  - login there via "normal" SSO (IdP or WAYF)
  - portal is an OpenID Connect client

# Master Portal

Additional requirements:

- token translation should result in *proxy* not *end-entity cert*
- integration with VOMS
- easy integration within VO portals
- need caching (prevent overloading of onlineCA)

→ extra heavy service in between: *Master Portal*

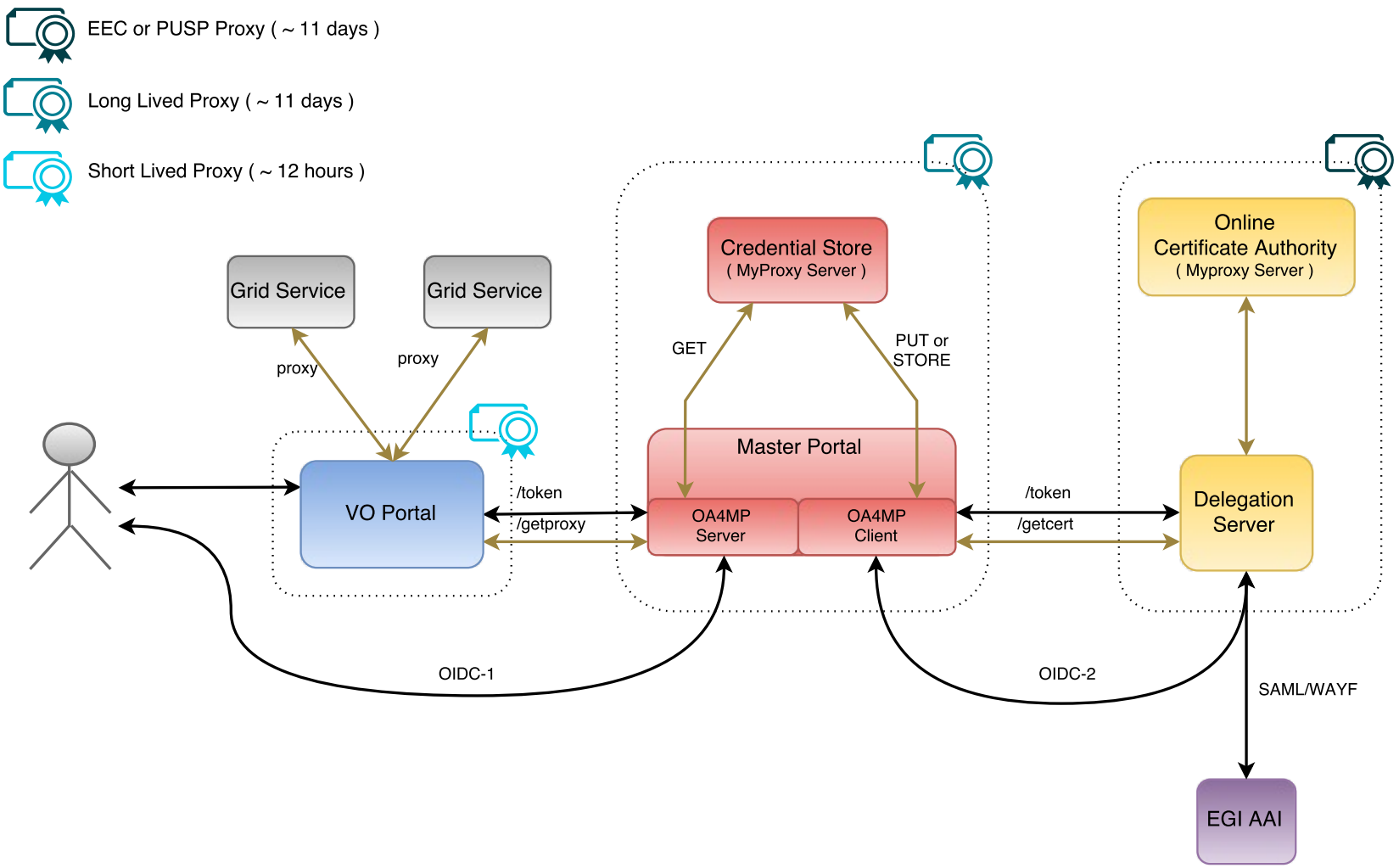
# Master Portal (cont.)

Implementation:

- implements both sides of same OA4MP protocol
- *client* to online CA
- *server* for VO portal
- VOMS pass-through: VO portal requests
- caches long-term proxy in MyProxy credential store

- Can also use as a pure OpenID Connect provider

# Flow


EEC or PUSP Proxy ( ~ 11 days )
Long Lived Proxy ( ~ 11 days )
Short Lived Proxy ( ~ 12 hours )
Grid Service
Grid Service
proxy
proxy
VO Portal
/token
/getproxy
Credential Store
( MyProxy Server )
GET
PUT or
STORE
Master Portal
OA4MP
Server
OA4MP
Client
/token
/getcert
Online
Certificate Authority
( Myproxy Server )
Delegation
Server
OIDC-1
OIDC-2
SAML/WAYF
EGI AAI

# Status

- Three (preliminary) masterportals running:
  - ELIXIR – talking to AARC demo CA
  - EGI – talking to EGI demo CA
  - EGI – talking to EGI demo "CA" with Robot cert for PUSP
- busy with a proper IGTF-approved CA (rcauth.eu, AARC)
- OIDC currently very basic:
  - only sub claim
- see demo in a minute...

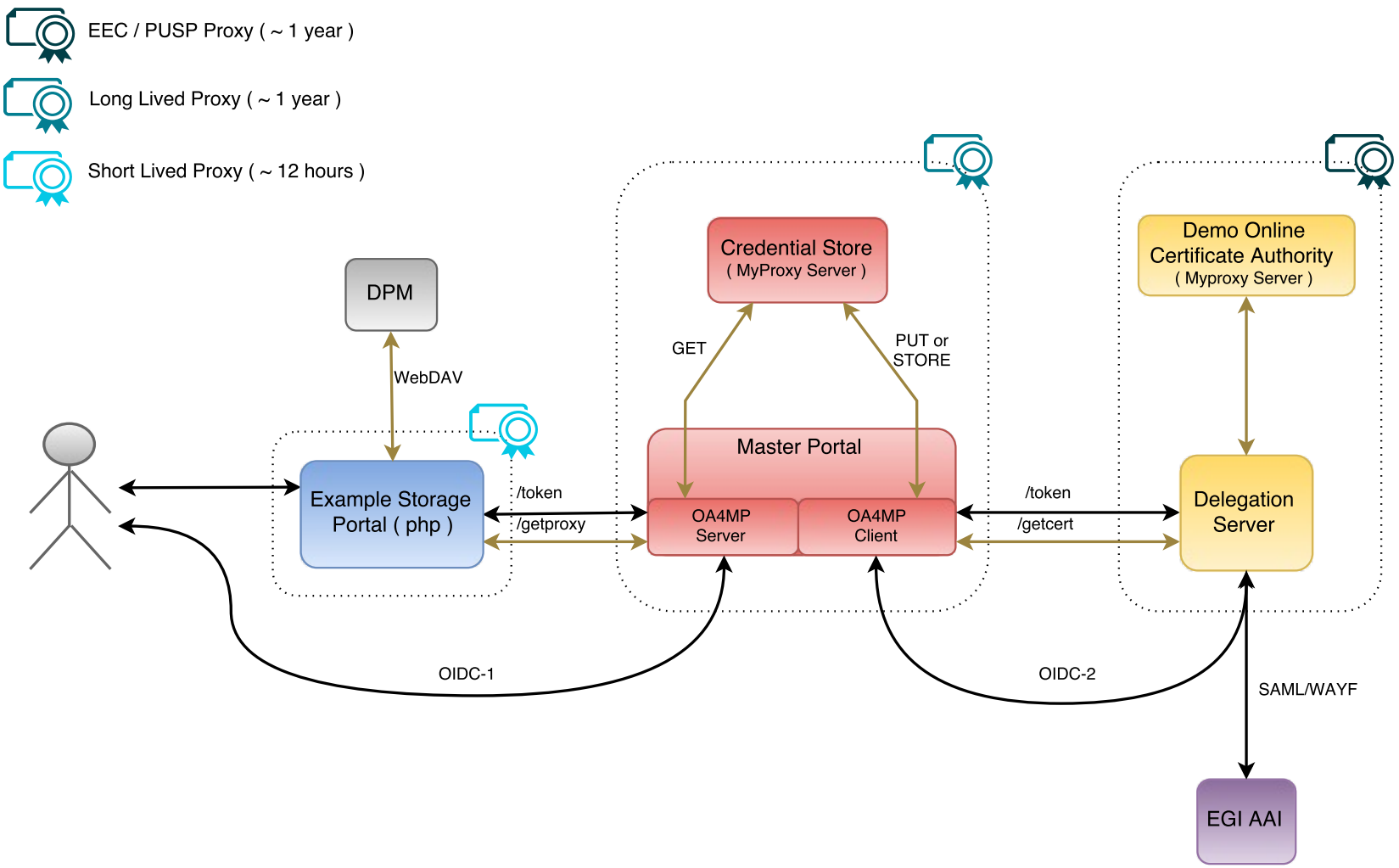
# Future plans

- improve traceability and data protection (EU $\neq$ US)
- extend OIDC support
  - more claims (email, name, ePPN, . . . )
  - reproduce Delegation Server OIDC claims in Master Portal
- for commandline access to resources:
  - cmdline tool for downloading proxy: à la github, svn . . .

  $\rightarrow$ upload ssh pubkey to Master Portal

  $\rightarrow$ ssh-server with authorized keys command for downloading

# Demo


EEC / PUSP Proxy ( ~ 1 year )
Long Lived Proxy ( ~ 1 year )
Short Lived Proxy ( ~ 12 hours )
DPM
WebDAV
Example Storage Portal ( php )
/token
/getproxy
Credential Store
( MyProxy Server )
GET
PUT or STORE
Master Portal
OA4MP Server
OA4MP Client
/token
/getcert
Demo Online Certificate Authority
( Myproxy Server )
Delegation Server
OIDC-1
OIDC-2
SAML/WAYF
EGI AAI

# Some References

- Our setup: `https://wiki.nikhef.nl/grid/CILogon_Pre-Pilot_Work`
- OpenID Connect for MyProxy: `http://goo.gl/VnMKXS`
- CILogon docs: `http://www.cilogon.org/portal-delegation`
- MyProxy: `http://grid.ncsa.illinois.edu/myproxy/`
  - OA4MP: `http://grid.ncsa.illinois.edu/myproxy/oauth/`
  - protocol: `http://grid.ncsa.illinois.edu/myproxy/protocol/`
- VOMS: e.g. `http://italiangrid.github.io/voms/`
- ssh `authorized_keys`: `man sshd`