

# **Pseudonymity Service – First Prototype Implementation**

Henri Mikkonen / HIP MWSG meeting 6.12.2007 Berkeley, CA, USA





www.eu-egee.org





- Some background (Budapest-recap)
  - Pseudo-system requirements
  - Components
  - Interaction of the components
- Implementation description
  - Design
  - Message sequence
- Short demonstration
  - How to obtain pseudonymous credentials?



- Must interoperate with the existing gLite middleware with minimal modifications
  - The pseudonym identities must be based on X.509 certificates
- The relationships between the pseudonym and real identities must be kept secret
  - The pseudonym identities must be unique and short-live
- The relationships must be possible to reveal in the case of misuse
  - Law enforcement or a similar legitimate body may require it as a part of their investigations



- Pseudonymity Service
  - The pseudonym identity provider and accountant

Enabling Grids for E-sciencE

- Online CA
  - Issues certificates with a pseudonymous subject DN
  - Standards like CMC (RFC 2797) and CMP (RFC 4210) exist for the communication
- Attribute Authority
  - Provides the attributes for the pseudonym identities

#### Client Software

Command line tool for accessing the pseudonymity service

# **CGCC**

### **Interaction of the Components**



**egee** 

Enabling Grids for E-sciencE

# Implementation





- Pseudonymity Service seems to have some similarities with the already implemented SLCS server
  - It is used for obtaining short-live certificates from an online CA
- SLCS server is used as a basis for the implementation, with the following modifications
  - User authentication is based on proxy certificate instead of Shibboleth IDP
  - User authorization is based on VOMS attributes instead of Shibboleth attributes
  - Certificate subject DN builder must build unique pseudonymous DNs
    - The DNs must be registered as to the VOMS server



- The user starts the sequence by running the client tool
  - 'pseudo-cert-request –url <server>'
  - The client tool must have have an access to a valid VOMS proxy
- The client tool contacts the pseudonymity service's login servlet
  - Https connection using an embedded HttpClient
  - gLlite Trustmanager is used for the mutual authentication on the server side



- The login servlet bases the user authorization to VOMS attributes
  - Pseudonymous DN and an authorization token are built after a successful authorization
- An example message:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SLCSLoginResponse>
<Status>Success</Status>
<AuthorizationToken>ED434BBAAAAB44F458DA9AA1A802DF520ABF7
503A15E45596AE5F1E3AF2837D6</AuthorizationToken>
<CertificateRequest
url="https://service.com/pseudoserv/certificate">
<Subject>
<Subject>
<Subject>
</Subject>
</CertificateRequest>
</SLCSLoginResponse>
```



- After receiving the <SLCSLoginResponse>, the client
  - generates a new public/private keypair
  - attaches the public key, subject DN and certificate extensions to a PKCS #10 certification request
- The following information is sent to the pseudonymity service's certificate servlet as POST parameters:
  - AuthorizationToken
    - The token is obtained from the <SLCSLoginResponse>
  - CertificateSigningRequest
    - PEM-encoded PKCS#10



- The certificate servlet first checks the validity of the authorization token and the certificate signing request
- The certificate request is then forwarded to the online CA
  - Depending on the online CA, the certificate signing request may need to be converted to a specific format, e.g. CMP CertReqMsg
- The online CA responds with the signed certificate
  - Again, the certificate may need conversion, as the response to the client must contain PEM-encoded PKCS#7
- The certificate servlet registers the new (pseudonymous) certificate as an alias to the original user certificate used in the mutual authentication process
  - Not yet implemented



- Finally, the certificate servlet sends the certificate back to the client
- An example message:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SLCSCertificateResponse>
<Status>Success</Status>
<Subject>
c=FI,O=HIP,OU=Tech,DC=Pseudotest,CN=42471505B592E5
</Subject>
<Certificate format="PEM">
-----BEGIN CERTIFICATE-----
<PEM-encoded certificate>
-----END CERTIFICATE-----
</Certificate>
</SLCSCertificateResponse>
```

• The client stores the certificate and the corresponding password-protected private key to the filesystem





- In a nutshell, the following steps are required before submitting a pseudo job
  - voms-proxy-init
    - Using the "normal" user certificate and the private key for obtaining the VOMS proxy
  - pseudo-cert-request
    - Using the "normal" VOMS proxy for obtaining the pseudonymous credentials
  - voms-proxy-init
    - Using pseudo credentials for obtaining the pseudonymous VOMS proxy

**egee** 

Enabling Grids for E-sciencE

## **Demonstration**



### **Demonstration Scenario**

- Client
  - Java 1.5
- Pseudonymity Service
  - Virtual host running Ubuntu

- Tomcat 5.5.23 and Java 1.5
- OnlineCA
  - Virtual host running Ubuntu
  - EJBCA 3.4.5 on JBoss 4.0.5 and Java 1.6