Authorization and Authentication in gLite

Stefano Dal Pra (INFN)

CYCLOPS Second Training Workshop,

Chania 05 – 07 May 2008

Thanks to
Emidio Giorgio (INFN Catania),
D. Cesini (INFN CNAF)
V. Ciaschini (INFN CNAF) et. al.

Overview

- Glossary
- Encryption
  - Symmetric algorithms
  - Asymmetric algorithms: PKI
- Certificates
  - Digital Signatures
  - Certification Authorities
  - X509 certificates

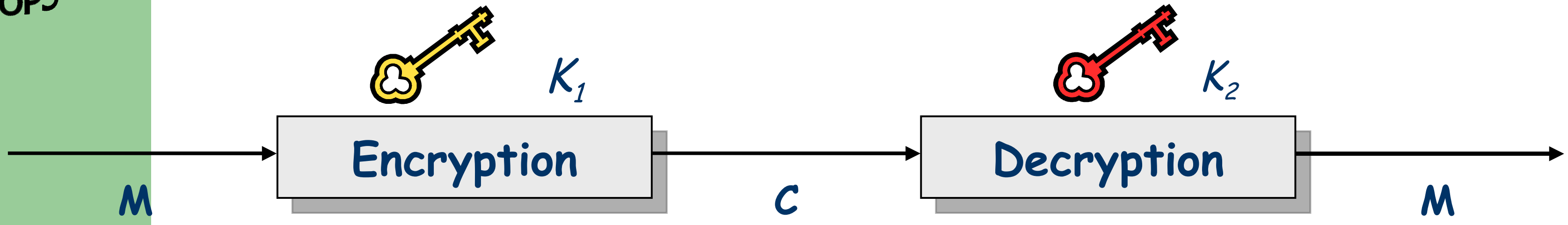
Overview

- Grid security
- Basic concepts
  - Grid Security Infrastructure
  - Proxy certificates
    - single sign-on
    - delegation

Glossary

- Principal
  - An entity: an user, a program, or a machine
- Credentials
  - Set of data identitying a principal
- Authentication
  - Identity verification of a principal
- Authorization
  - Granting a set of privileges to an Principal
- Confidentiality
  - Ensuring that a clear message is receivable only to a given Principal
- Integrity
  - Ensuring that a received message has not been altered.
- Non-repudiation
  - Impossibility of denying the authenticity of a digital signature

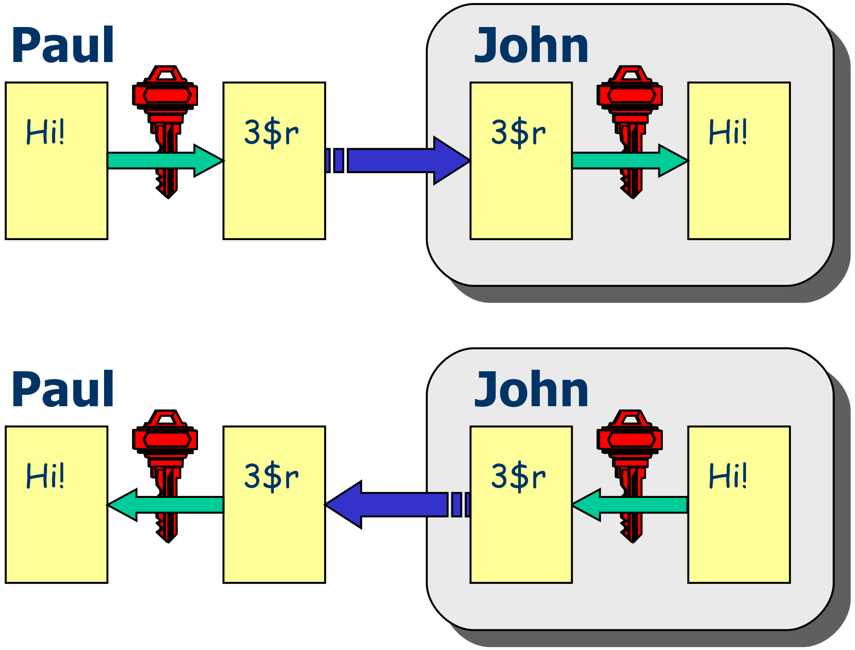
Cryptography (symmetric or asymmetric)



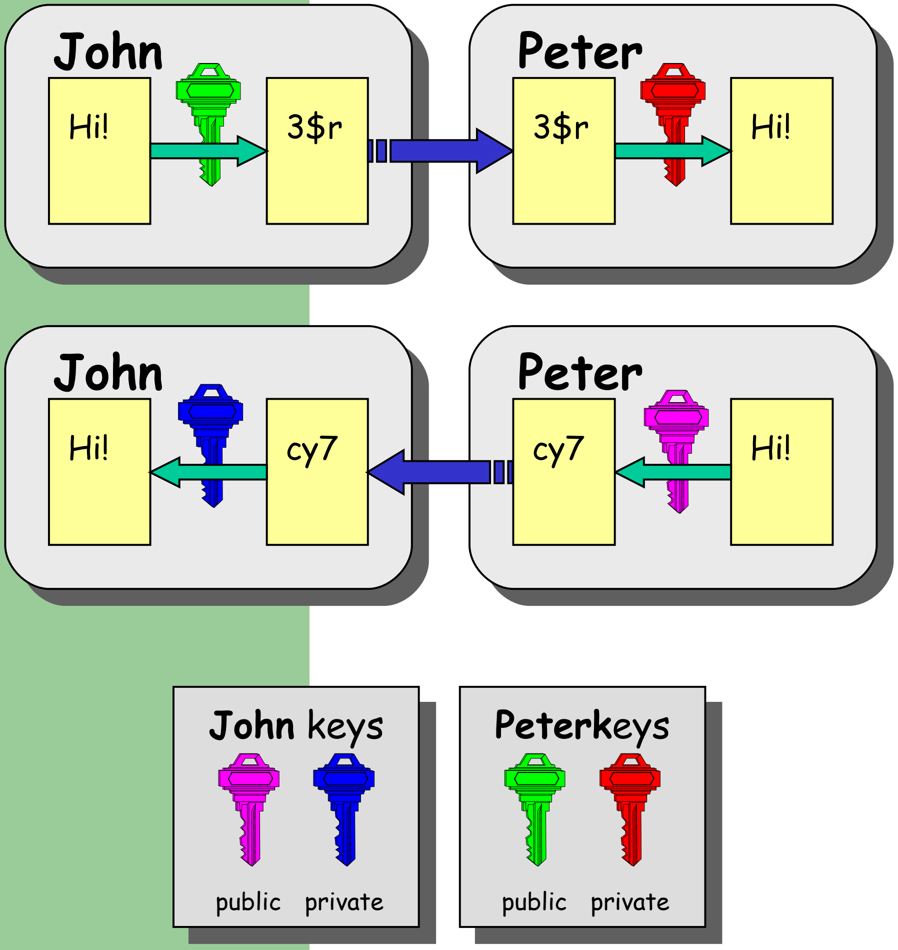
- A mathematical algorithm provides important functions for the implementation of a security infrastructure
- Symbology
  - Plaintext: *M*
  - Cyphertext: *C*
  - Encryption with key K_1: $E_{K_1}(M) = C$
  - Decryption with key K_2: $D_{K_2}(C) = M$
- Algorithms
  - **Symmetric**: $K_1 = K_2$
  - **Asymmetric**: $K_1 \neq K_2$

Symmetric Algorithms

- The **same** key is used for encryption and decryption
- Advantages:
  - Fast
- Disadvantages:
  - Keys distribution
  - keys number O(n^2)
- Examples:
  - **DES**
  - **3DES**
  - **Rijndael** (AES)
  - **Blowfish**



Asymmetric Algorithms (RSA)



- Each user owns two keys: a *private* and a *public* one:
  - it is *impossible* to calculate a private key with the public one;
  - a message encrypted by a key is **only** decryptable by the other one.
- No exchange of private key is needed.
  - the sender cyphers with the *public* key of the receiver;
  - the receiver decrypts using his own *private* key;
  - the number of keys is O(n).
- Slower than symmetric alg.
  - Idea: use RSA to safely exchange simmetric key, then use it.

One-Way Hash Functions

  - A *hash function* transform an arbitrary message (file) in a nontrivial fixed length string.
  - $H(M) = h$ **must be easy** (fast) to compute
  - $M = H^{-1}(h)$ **must be difficult** to compute
  - given M, it **must be difficult** to find M' such that $H(M) = H(M')$
- Examples:
  - **MD4/MD5**: hash of 128 bits;
  - **SHA** (Standard FIPS): hash of 160 bits.

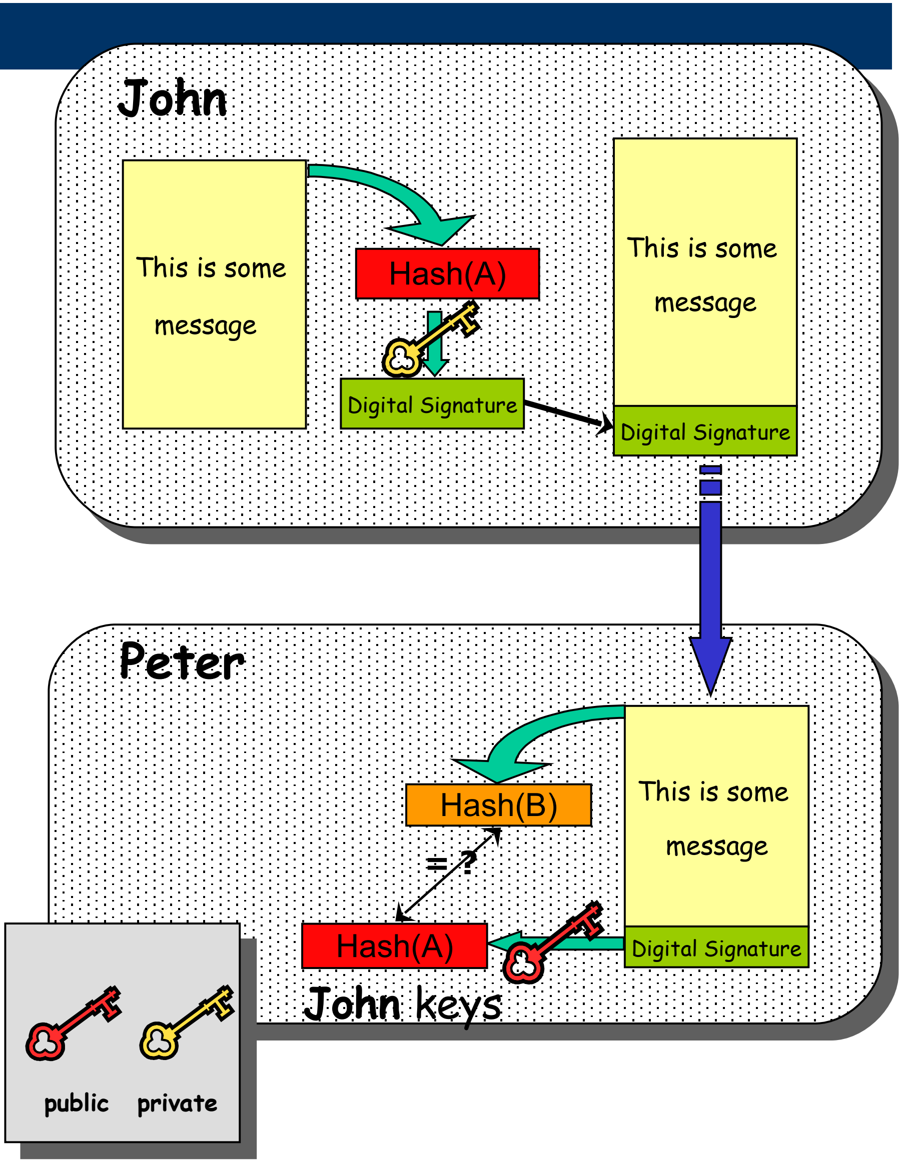
Example (md5sum)

```
[user@host]$ cat mytest
testo di prova
[user@host]$ md5sum mytest
909adc30dcc15239ac640b52d33a12b2  mytest
[user@host]$ cat mytest2
testo di prova
[user@host]$ md5sum mytest2
c89ee15b2f056edfbef2dcb62b2249aa  mytest2
[user@host]$ ls -l /bin/ls
-rwxr-xr-x    1 root     root         67700 Dec  9  2005 /bin/ls
[user@host]$ md5sum /bin/ls
2636c546ce5ca69687f5dfc74cc3175e  /bin/ls
```

Useful to check files equality!!

Digital Signature

- **John** calculates the ***hash*** of the message (with a one-way hash function)
- **John** encrypts the hash using his ***private*** key: the encrypted hash is the ***digital signature***.
- **John** sends the signed message to **Peter.**
- **Peter** calculates the hash of the message and ***verifies*** it with A, decyphered with **Peter**'s ***public*** key.
- If two hashes equal: message wasn't modified; **John** cannot repudiate it.



Digital Certificate

- **John’s digital signature is safe if:**
  1. John’s private key is not compromised
  2. Peter knows and trust John’s public key
- **How can Peter be sure that John’s public key is really John’s public key and not someone else’s?**
  - A *third party* guarantees the correspondence between public key and owner’s identity.
  - Both John and Peter must trust this third party
- **Two models proposed to build trust:**
  - X.509: hierarchical organization (**used in Grid**)
  - PGP: “web of trust”. (person to person)

X.509 and Certification Authorities

The “third party” is called ***Certification Authority*** (CA).

The CA is responsible of:

- **Issue Digital Certificates (containing public key and owner’s identity) for users, programs and machines**
- **Check identity and the personal data of the requestor**
  - Registration Authorities (RAs) do the actual validation
- **Revoke certificates in case of a compromise**
- **Renew certificates in case of expiration**
- **Periodically publish a list of revoked certificates through web repository**
  - **Certificate Revocation Lists** (CRL): contain all the revoked certificates
- **CA certificates are self-signed**

Revocation Lists

- The CAs have the obligation of issue Certificate Revocation Lists (CRL)
- The CRLs contain:
  - a list of the revoked certificates
  - the date when they were issued
  - the end date
- CRLs are signed with the CA private key
- The CRLs must be published so that the relying parties can check the validity of the certificates
  - Usually available through a web page

X.509 Certificate

An X.509 Certificate contains:

owner's public key;

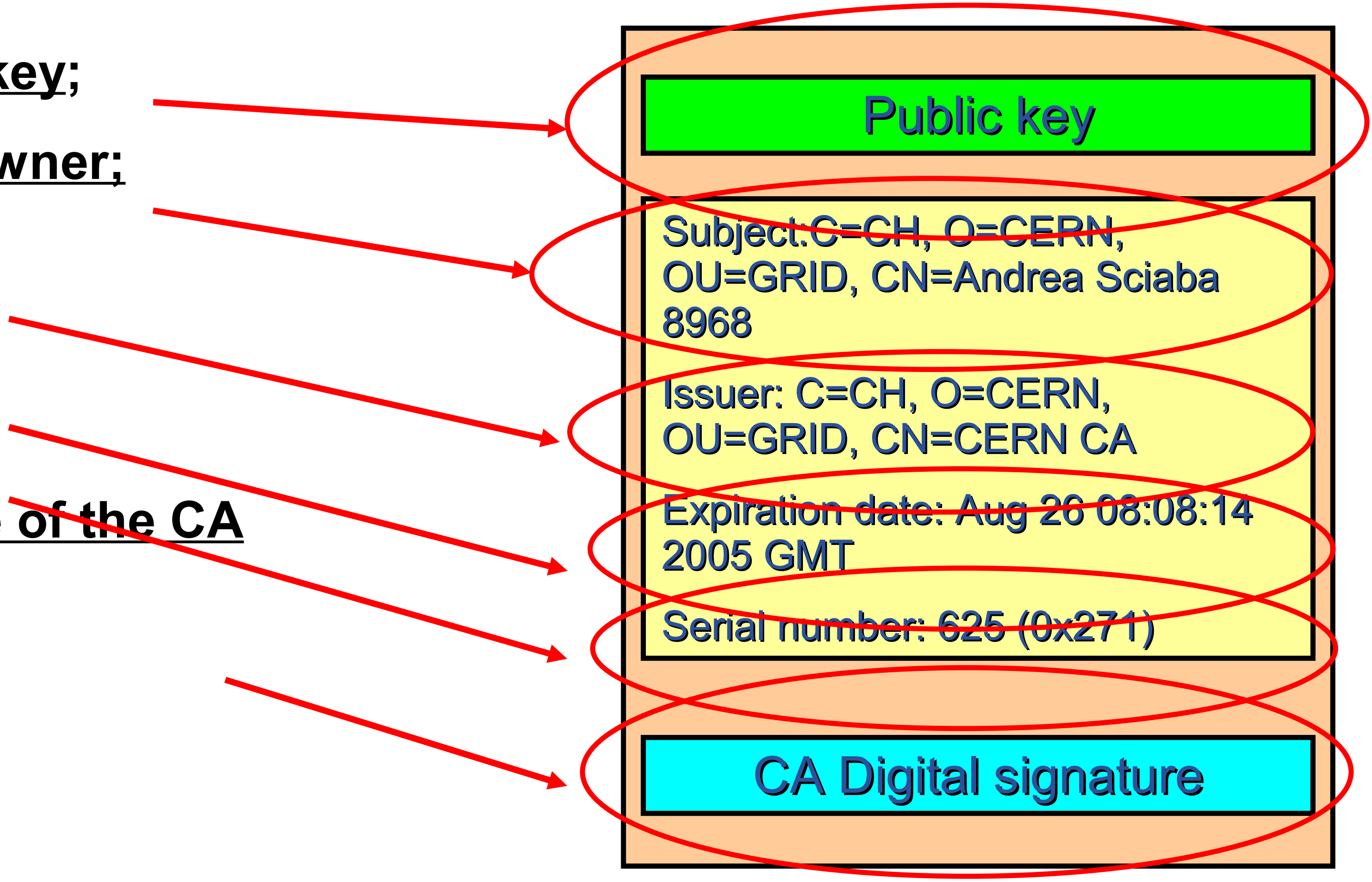
identity of the owner;

info on the CA;

time of validity;

Serial number;

digital signature of the CA



Public key

Subject:C=CH, O=CERN, OU=GRID, CN=Andrea Sciaba 8968

Issuer: C=CH, O=CERN, OU=GRID, CN=CERN CA

Expiration date: Aug 26 08:08:14 2005 GMT

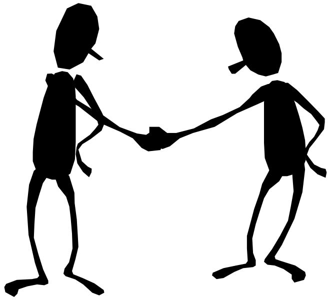
Serial number: 625 (0x271)

CA Digital signature

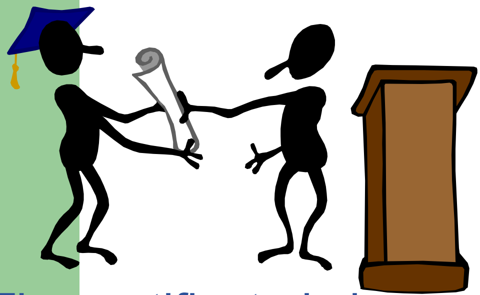
Obtaining a certificate

- **How to obtain a certificate:**

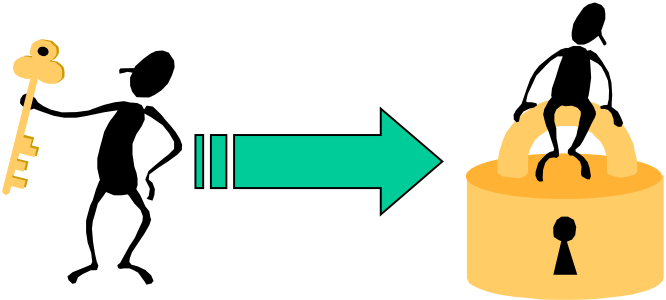
A certificate request is performed



The user identify is confirmed by the RA

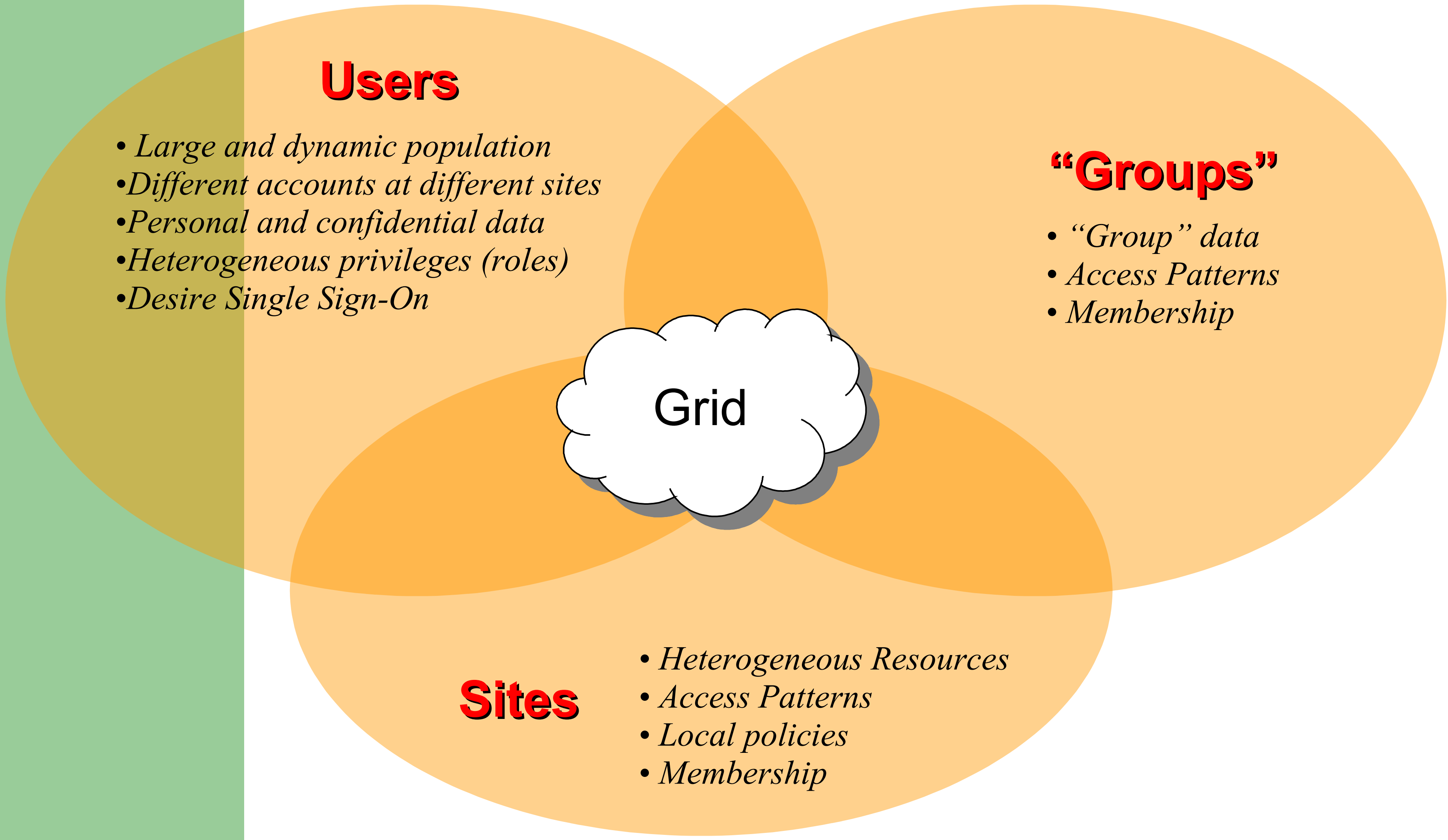


The certificate is issued by the CA



The certificate is used as a key to access the grid

GRID Security: Components



Users

- *Large and dynamic population*
- *Different accounts at different sites*
- *Personal and confidential data*
- *Heterogeneous privileges (roles)*
- *Desire Single Sign-On*

"Groups"

- *"Group" data*
- *Access Patterns*
- *Membership*

Grid

Sites

- *Heterogeneous Resources*
- *Access Patterns*
- *Local policies*
- *Membership*

The Grid Security Infrastructure (GSI)

Based on X.509 PKI:

- every user/host/service has an X.509 certificate;
- certificates a
  sites) CA's;
- every Grid t
  authenticat

1. John se
2. Peter ve
3. Peter se
4. John en
   private
5. John se
6. Peter us
   challen
7. Peter co
   the orig
8. If they match, Peter verifies John's identity
   and John can not repudiate it.

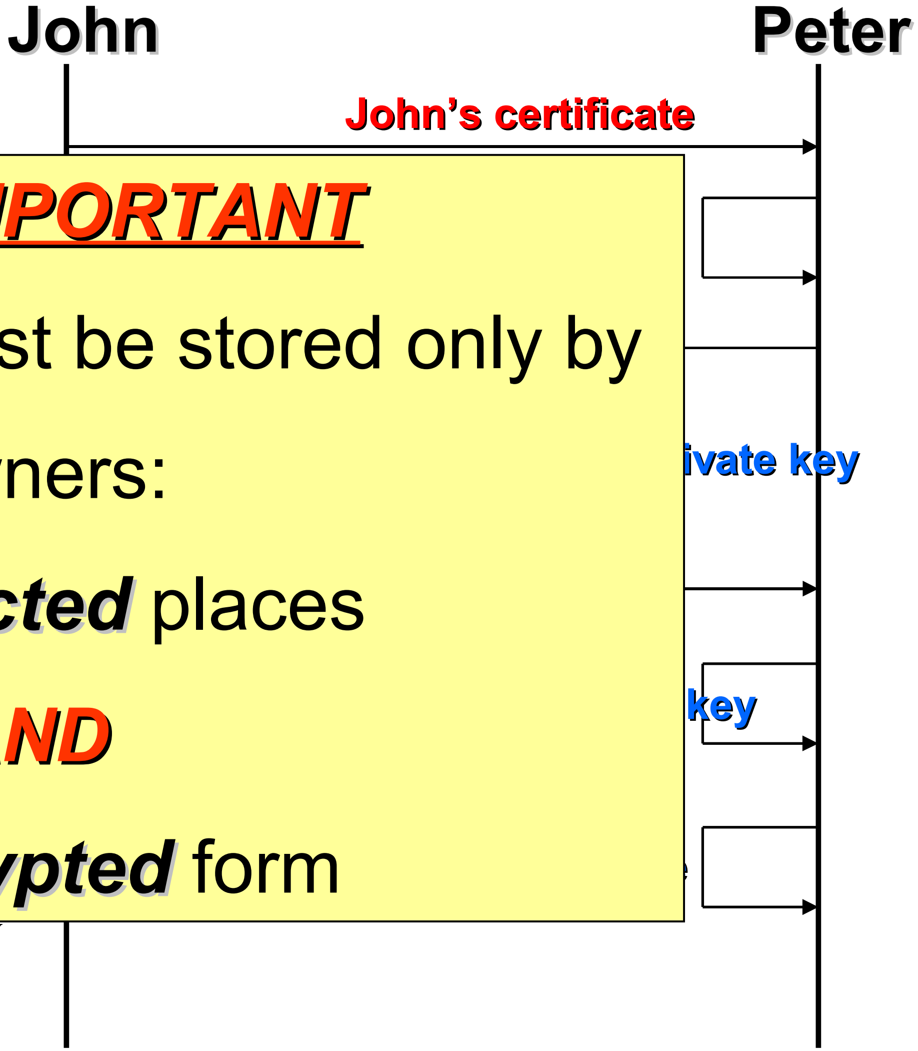
VERY IMPORTANT

Private keys must be stored only by owners:

in ***protected*** places

AND

in ***encrypted*** form



Certificate management

- **Import your certificate in your browser**
  - If you receive a .pem certificate you need to convert it to PKCS12
  - Use *openssl* command line (available in each egee/LCG UI)
    - `openssl pkcs12 –export –in usercert.pem –inkey userkey.pem –out my_cert.p12 –name 'My Name'`

- **GILDA (and other VOs):**
  - If you receive already a PKCS12 certificate, you can import it directly into the web browser.
  - For future use, you will need *usercert.pem* and *userkey.pem* in a directory ~/.globus on your UI
  - Export the PKCS12 cert to a local dir on UI and use again *openssl:*
    - `openssl pkcs12 -nocerts -in my_cert.p12 -out userkey.pem`
    - `openssl pkcs12 -clcerts -nokeys -in my_cert.p12 -out usercert.pem`

X.509 Proxy Certificate

- **Proxy: GSI extension to X.509 Identity Certificates**
  - signed by the normal end entity cert (or by another proxy).
- **It enables single sign-on.**
- **It supports some important features:**
  - Delegation
  - Mutual authentication
- **It has a limited lifetime (minimized risk of “compromised credentials”)**
- **It is created by the voms-proxy-init command:**

```
% voms-proxy-init
Enter PEM pass phrase: ******
```

  - Options for grid-proxy-init:
    - -hours <lifetime of credential>
    - -bits <length of key>
    - -help

GSI environment variables

- **User certificate files:**
  - Certificate: X509_USER_CERT (default: `$HOME/.globus/usercert.pem`)
  - Private key: X509_USER_KEY (default: `$HOME/.globus/userkey.pem`)
  - Proxy: X509_USER_PROXY (default: `/tmp/x509up_u<id>`)
- **Host certificate files:**
  - Certificate: X509_HOST_CERT (default: `/etc/grid-security/hostcert.pem`)
  - Private key: X509_HOST_KEY (default: `/etc/grid-security/hostkey.pem`)

GSI environment variables

- **Trusted certification authority certificates:**
  - X509_CERT_DIR

    (default: `/etc/grid-security/certificates`)

- **Voms server public keys**
  - X509_VOMS_DIR

    (default: `/etc/grid-security/vomsdir`)

Certificate Management

- Import your certificate in your browser
  - If you received a .pem certificate you need to convert it to PKCS12
  - Use openssl command line (available in each UI)

```
  openssl pkcs12 –export –in usercert.pem –inkey userkey.pem –
out my_cert.p12 –name ’My Name’
```

- Most of other CA’s:
  - You receive already a PKCS12 certificate (can import it directly into the web browser)
  - For future use, you will need usercert.pem and userkey.pem in a directory ~/.globus on your UI
  - Export the PKCS12 cert to a local dir on UI and use again openssl:

```
  openssl pkcs12 –nocerts –in my_cert.p12 –out userkey.pem
 openssl pkcs12 –clcerts –nokeys –in my_cert.p12 –out
usercert.pem
```

X.509 Proxy

- GSI extension to X.509 Identity Certificates
  - signed by the normal end entity cert (or by another proxy).
- Enables single sign-on and support important features:
  - Delegation, Mutual authentication
- Has a limited lifetime (minimized risk of “compromised credentials”)
- It is created by the voms-proxy-init command:

```
> voms-proxy-init --vo cyclops
Enter pass phrase: ******
```

- Options for voms-proxy-init:

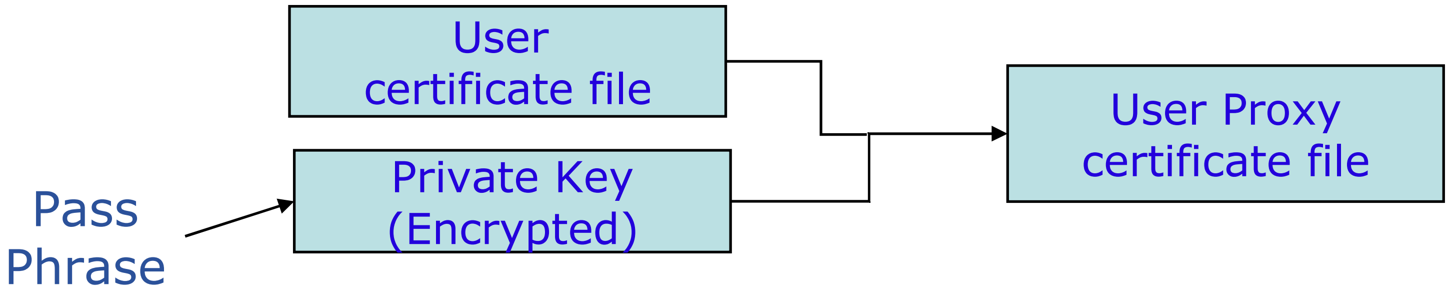
```
-hours <lifetime of credential>
-bits <length of key>
-help
```

How Proxies are created

User enters pass phrase, used to decrypt private key.

Private key is used to sign a proxy certificate with its own, new public/private key pair.

User’s private key not exposed after proxy has been signed



Proxy

- its private key is *not encrypted* and stored in local file: must be readable **only** by the owner

- its lifetime is short (typically 12 h) to minimize security risks.

Managing Proxies

- voms-proxy-init ≡ “login to the Grid”
- To “logout” you have to destroy your proxy:
  - `voms-proxy-destroy`
- To gather information about your proxy:
  - `voms-proxy-info --all`
  - Options for printing proxy information

    -subject
    -type
    -strength
    -issuer
    -timeleft
    -help

Delegation

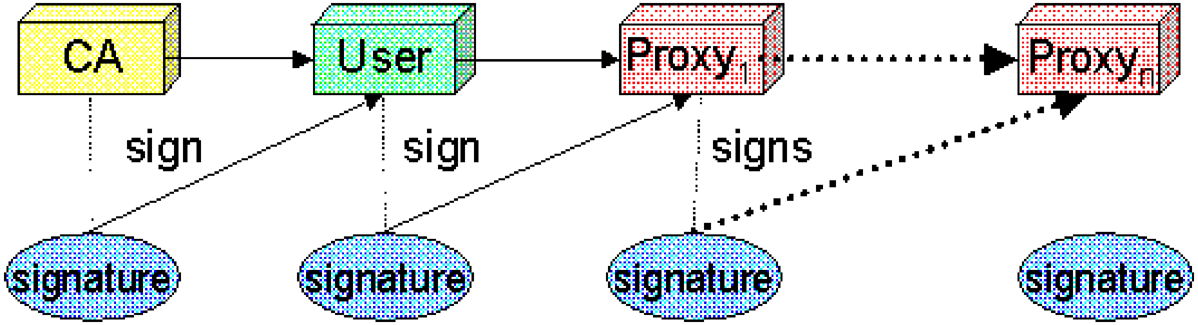
Delegation = remote creation of a (second level) proxy credential

New key pair generated remotely on server

Client signs proxy cert and returns it

Allows remote process to authenticate on behalf of the user

Remote process “impersonates” the user



VOMS Concepts

- Each Grid User **MUST** belong to a "Virtual Organization"
- VOMS offers Virtual Organization Membership Service
  - Extends the proxy with info on VO membership, group, roles
  - Fully compatible with Globus Toolkit
  - Each VO has a database containing group membership, roles and capabilities informations for each user
  - User contacts voms server requesting his authorization info
  - Server send authorization info to the client, which includes them in a proxy certificate

Voms-proxy-init

```
[sdalpra@cyclops-01 ~]$ voms-proxy-init --voms cyclops -hours 72
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Stefano Dal Pra
Creating temporary proxy ................................ Done
Contacting  voms-02.pd.infn.it:15011
    [/C=IT/O=INFN/OU=Host/L=Padova/CN=voms-02.pd.infn.it] "cyclops" Done
Creating proxy .................................... Done
Your proxy is valid until Thu May  8 10:10:19 2008
```

- voms-proxy-init -voms cyclops:/cyclops/Role=SoftwareManager
  - To enable a proxy with “Software Manager” role enabled

VOMS and Access Control

- Server creates and sign an AC containing the FQAN (Fully Qualified Attribute Name) requested by the user
- If applicable, the AC is included by the client in a well-defined, non critical, extension in a compatible manner

voms-proxy-info

```
[sdalpra@cyclops–01 ~]$ voms–proxy–info –all
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Stefano Dal Pra/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Stefano Dal Pra
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Stefano Dal Pra
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u501
timeleft  : 11:59:42
=== VO cyclops extension information ===
VO        : cyclops
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Padova/CN=Stefano Dal Pra
issuer    : /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms2.cnaf.infn.it
attribute : /cyclops/Role=SoftwareManager/Capability=NULL
attribute : /cyclops/Role=NULL/Capability=NULL
timeleft  : 11:59:42
```

Groups

- The number of users of a VO can be very high:
  - e.g.: the experiment ATLAS has 2000 member
- Make VO manageable by organizing users in groups:
  Examples:
  - VO GILDA
    - Group Catania
      - INFN
        - Group Barbera
      - University
    - Group Padua
  - VO GILDA
    - /GILDA/TUTORS can write to normal storage
    - /GILDA/STUDENT only write to volatile space
- Groups can have a hierarchical structure, indefinitely deep

Roles

- Roles are specific roles a user has and that distinguishes him from others in his group:
  - Software manager
  - VO-Administrator
- Difference between roles and groups:
  - Roles have no hierarchical structure – there is no sub-role
  - Roles are not used in ‘normal operation’
    - They are not added to the proxy by default when running voms-proxy-init
    - But they can be added to the proxy for special purposes when running voms-proxy-init

Group and roles example

- Example:
  - User Emidio has the following membership
  - VO=gilda, Group=tutors, Role=SoftwareManager
  - During normal operation the role is not taken into account (Emidio works as a normal user)
  - For special tasks he can obtain the role “Software Manager” that he has to explicitly request with the appropriate option.

LCAS and LCMAPS

- At resource level, authorization info are extracted from the proxy and processed by *LCAS* and *LCMAPS*
- Local Centre Authorization Service (LCAS)
  Checks if the user is authorized (currently using the grid-mapfile)
  Checks if the user is banned at the site
  Checks if at that time the site accepts jobs
- Local Credential Mapping Service (LCMAPS)
  Maps grid credentials to local credentials (eg. UNIX uid/gid, AFS tokens, VOMS group and roles)

```
    "/VO=cms/GROUP=/cms"                      .cms
  "/VO=cms/GROUP=/cms/prod"                   .cmsprod
  "/VO=cms/GROUP=/cms/prod/ROLE=manager"      .cmsprodman
```

References

- Cookbook
  - Quick introduction for cyclops user (Cyclops Deliverable 7)
    - http://www.cyclops-project.eu/
      - Follow “Results Documentation”
      - Pick “D07-Cyclops-EGEE_Cookbook.pdf” file
- VOMS

  http://proj-lcg-security.web.cern.ch/proj-lcg-security/
- CA

  http://proj-lcg-security.web.cern.ch/proj-lcg-security/
- PI2S2 Wiki - Authentication and Authorization

  https://grid.ct.infn.it/twiki/bin/view/PI2S2/AuthenticationAuthorization
- PI2S2 Wiki - How To Import Certificate In A Web Browser

  https://grid.ct.infn.it/twiki/bin/view/PI2S2/HowToImportCertificateInAWebBrowser