Grid security

Robert Lovas
MTA SZTAKI
rlovas@sztaki.hu

…with thanks to colleagues in EGEE, Globus and ICEAGE for many of these slides.

www.eu-egee.org

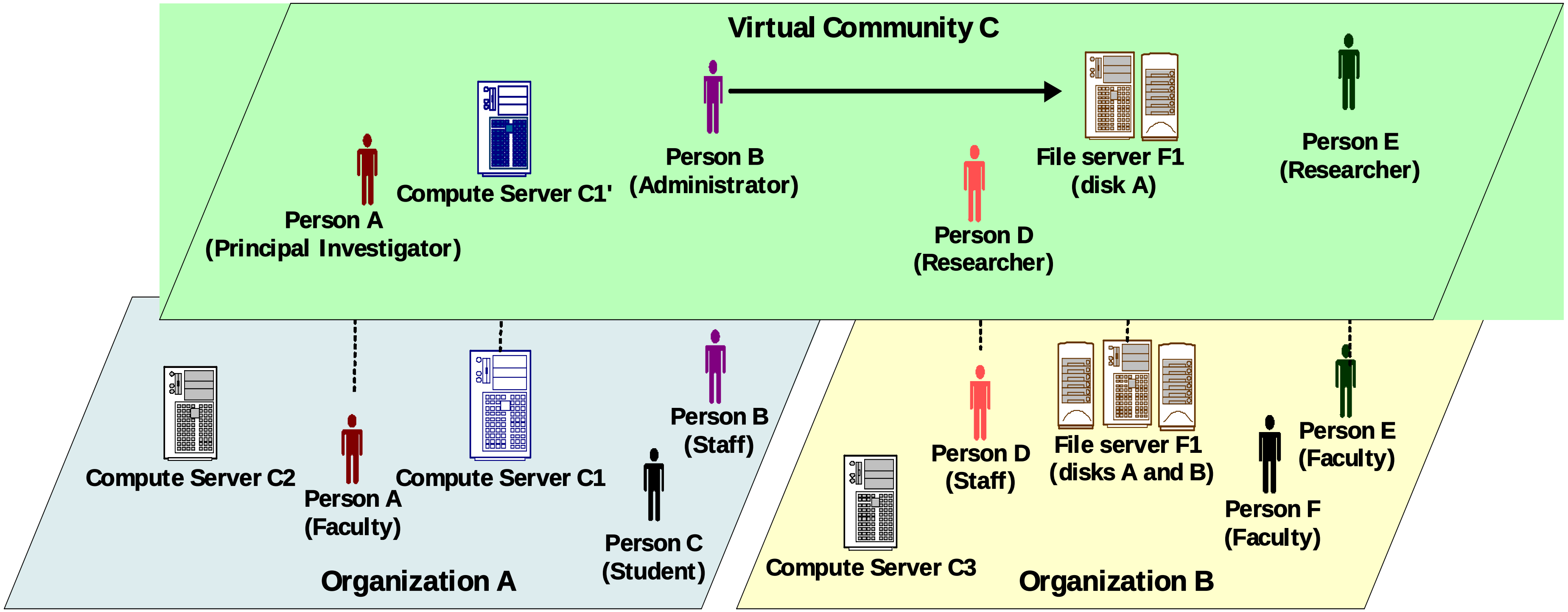
What is Grid security?

The Grid problem is to enable "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations."

From **"The Anatomy of the Grid"** by Ian Foster at. al

- So Grid Security is security to enable VOs
- What is needed in terms of security for a VO?

Virtual Organization (VO) Concept

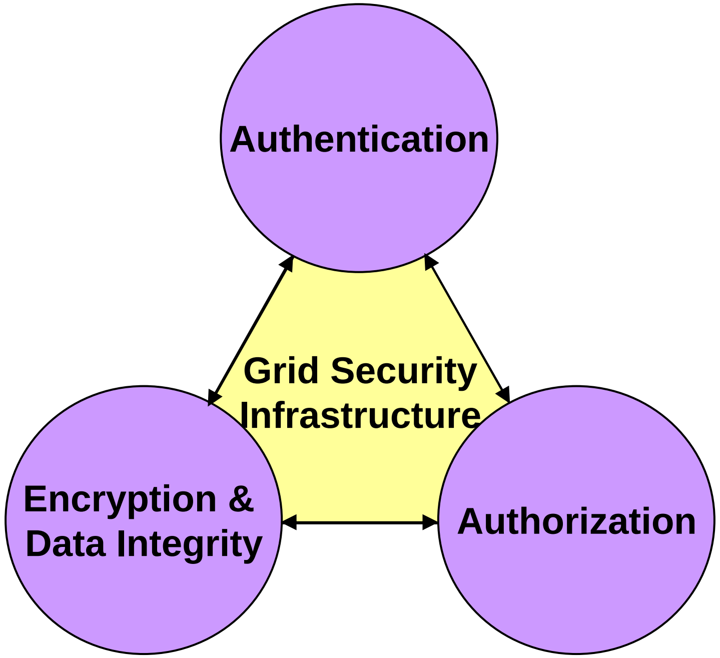


- VO for each application or workload
- Carve out and configure resources for a particular use and set of users

Overview

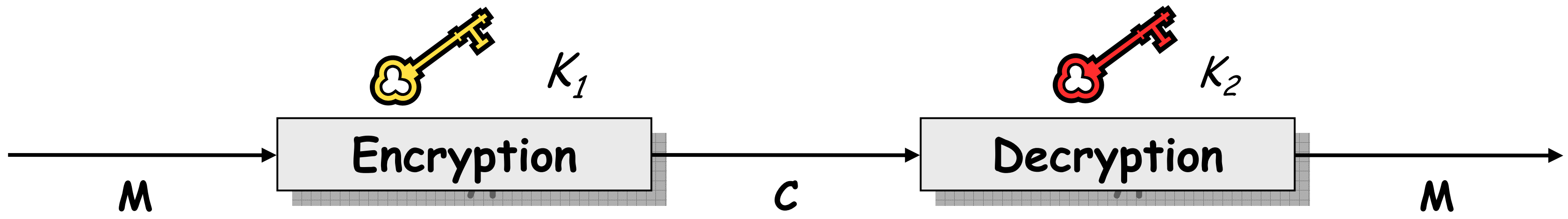
- **Security background: encryption mechanisms**
  - Symmetric algorithms
  - Asymmetric algorithms
- **Certificates: a way to authenticate users and services**
  - Certificate Authorities
  - X509 certificates
- **Grid Security Infrastructure (GSI)**
  - X.509 mechanisms in GSI
  - Delegation, proxy certificates
- **Virtual Organizations**
  - Globus, LCG, gLite
- **Summary**

Focus areas



1. **Authentication – communication of identity**
   - Message confidentiality - so only sender and receiver can understand the message
   - Non-repudiation: knowing who did what when – can’t deny it
   - Message integrity - so tampering is recognised
2. **Authorisation – once identity is known, what can a user do?**
3. **Delegation – A allows B to act on behalf of A**

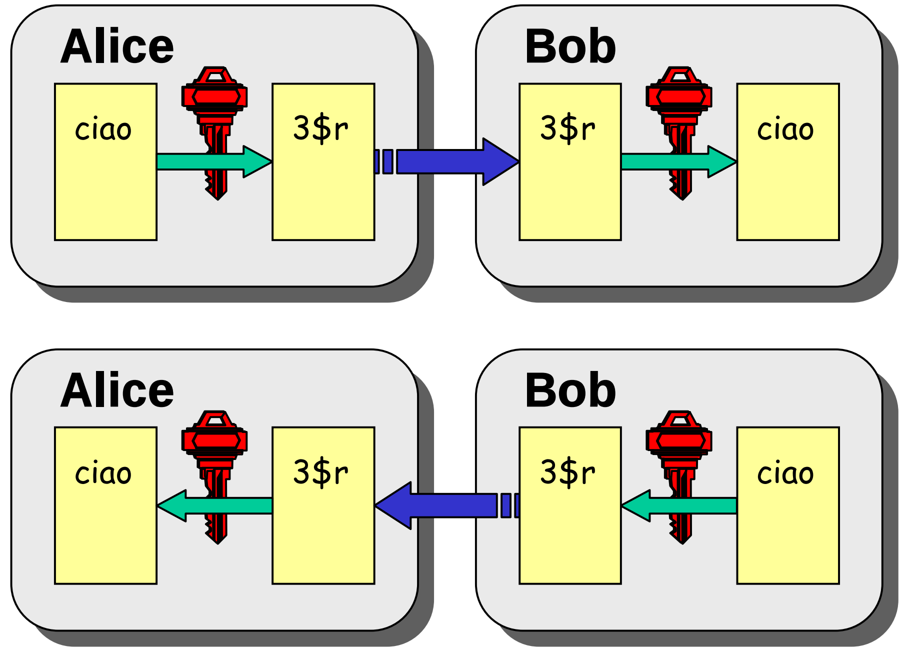
Network security basics: Cryptography ("hidden writing")



- **Mathematical algorithms that provide important building blocks for the implementation of a security infrastructure**
- **Symbology**
  - Plain text: M
  - Encrypted text: 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**: $K_1 \neq K_2$

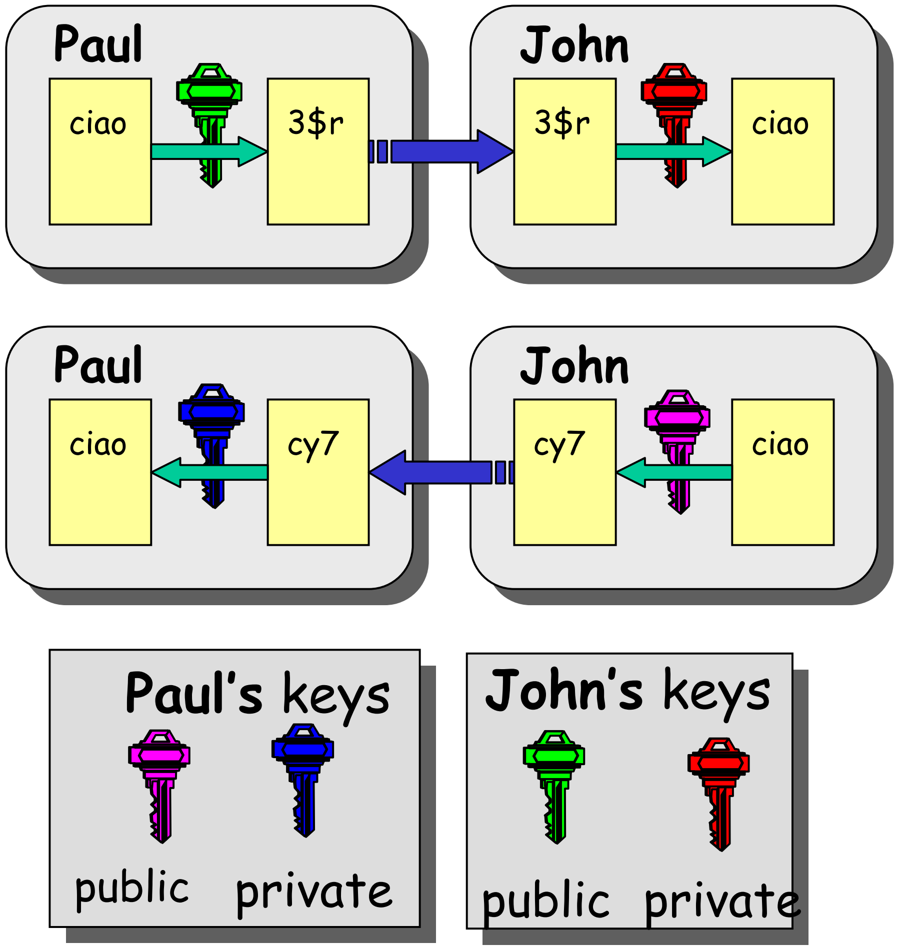
Symmetric Algorithms

- **The** same **key is used for encryption and decryption**



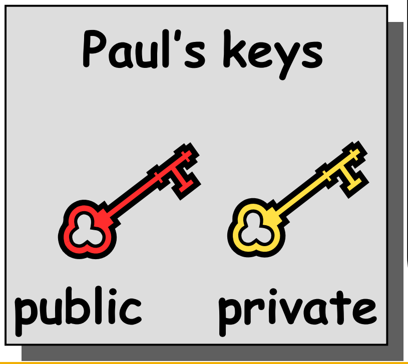
- **Disadvantages:**
  - how to distribute the keys?
  - the number of keys is $O(n^2)$
  - n: number of people

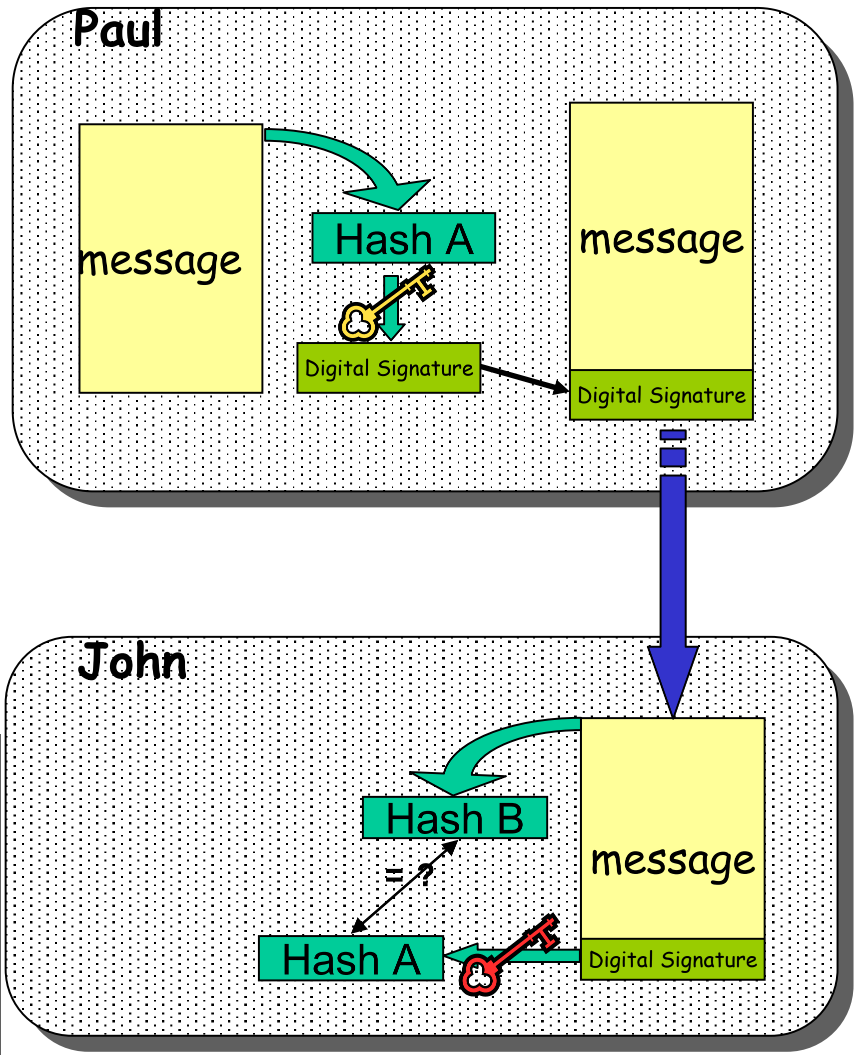
- **Every user has two keys: one *private* and one *public*:**
  - it is *impossible* to derive the private key from the public one;
  - a message encrypted by one key can be decrypted **only** by the other one.
- **Public keys are exchanged**
- **The sender encrypts using the *public* key of the receiver**
- **The receiver decrypts using his *private* key;**
- **The number of keys is O(n)**
- ***What about non-repudiation?***



Digital Signatures - Authentication 2, 3: non repudiation and integrity

- Paul **calculates the *hash* of the message:** *a 128 bit value based on the content of the message*
- Paul **encrypts the hash using his** *private* **key: the encrypted hash is the** *digital signature*.
- Paul **sends the signed message to** John.
- John **calculates the hash of the message**
- **Decrypts A with Paul's *public* key.**
- **If hashes equal:**
  1. hash B is from Paul's private key;
- **2. message wasn't modified;**





One-way hashing: used for digital signatures

- **A hash function (H) is a function that given as input a variable-length message (*M*) produce as output a string of fixed length (*h*)**
  - the length of *h* must be at least 128 bits (to avoid *birthday attacks*)
  - given *M*, it **must be easy** to calculate $H(M) = h$
  - given *h*, it **must be difficult** to calculate $M = H^{-1}(h)$
  - given *M*, it **must be difficult** to find *M'* such that $H(M) = H(M')$
- **Examples:**
  - **SNEFRU**: hash of 128 or 256 bits;
  - **MD4/MD5**: hash of 128 bits;
  - **SHA** (Standard FIPS): hash of 160 bits.

Digital Certificates

- **Paul's digital signature is useful to John if:**
  1. Paul's private key is not compromised – keep these safe!!!
  2. John has Paul's public key

- **How can John be sure that Paul's public key is really <u>Paul's</u> public key and not someone else's?**
  – A *third party* establishes the correspondence between public key and owner's identity.
  – Both John and Paul trust this third party

The "third party" is called a *<u>Certification Authority</u>* (CA).

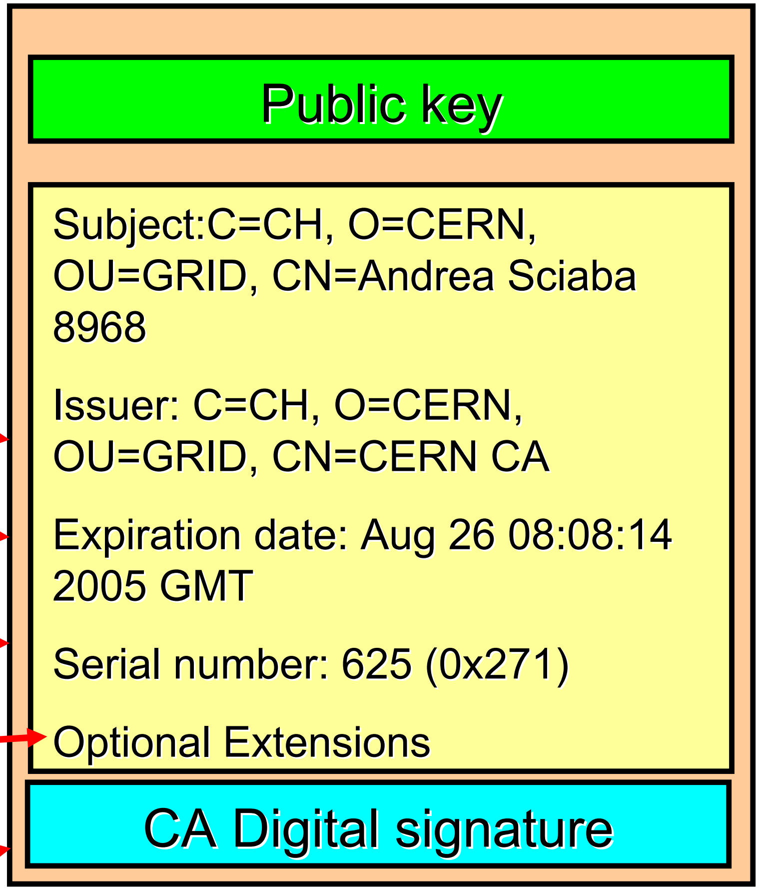
Certificate Authority

- **Issues** Digital Certificates **for users, programs and machines**
  - Combines public key + owner information
  - Signed by CA using its private certificate
  - Can use the CA's public certificate to check integrity of certificates
- **CA's check the identity and the personal data of the requestor of a certificate**
  - Registration Authorities (RAs) do the actual validation
- **CA's periodically publish a list of compromised certificates**
  - **Certificate Revocation Lists** (CRL): contain all the revoked certificates yet to expire
- **CA's own certificates are self-signed**

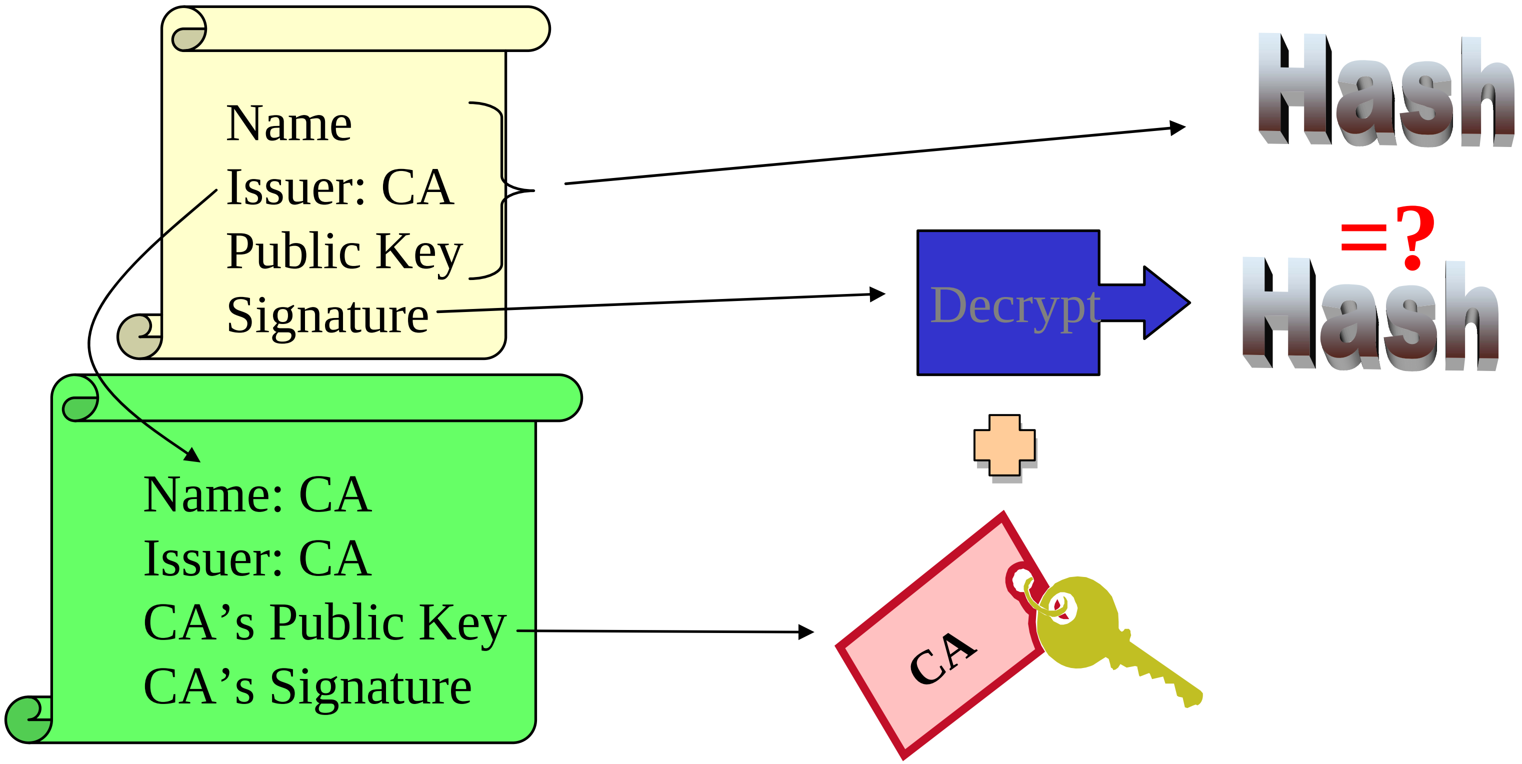
X.509 Certificates

- **An X.509 Certificate contains:**
  - owner's public key;
  - identity of the owner;
  - info on the CA;
  - time of validity;
  - Serial number;
  - Optional extensions
  - digital signature of the CA

Structure of a X.509 certificate

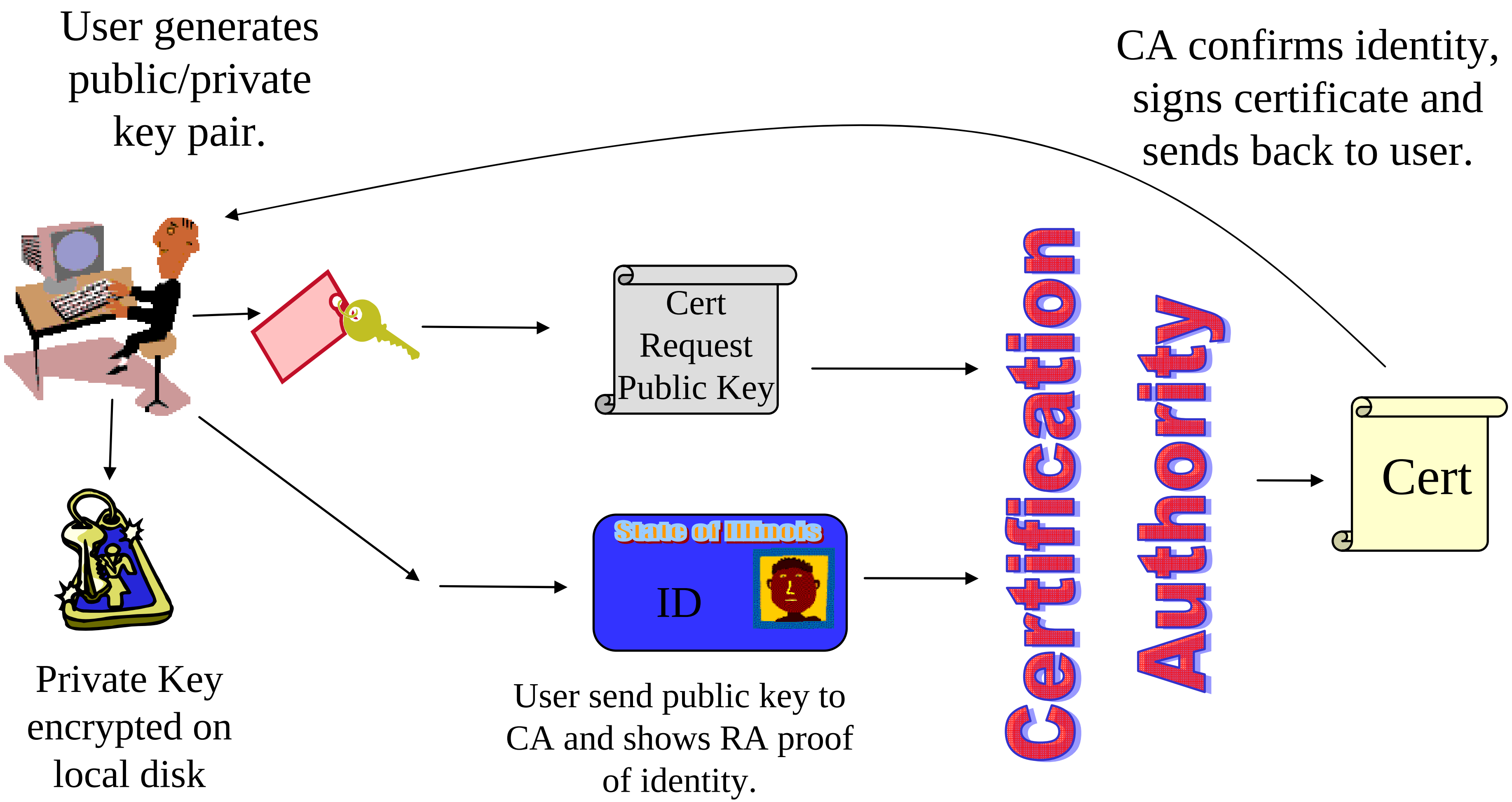


- The public key from the CA certificate can be used to verify the certificate.

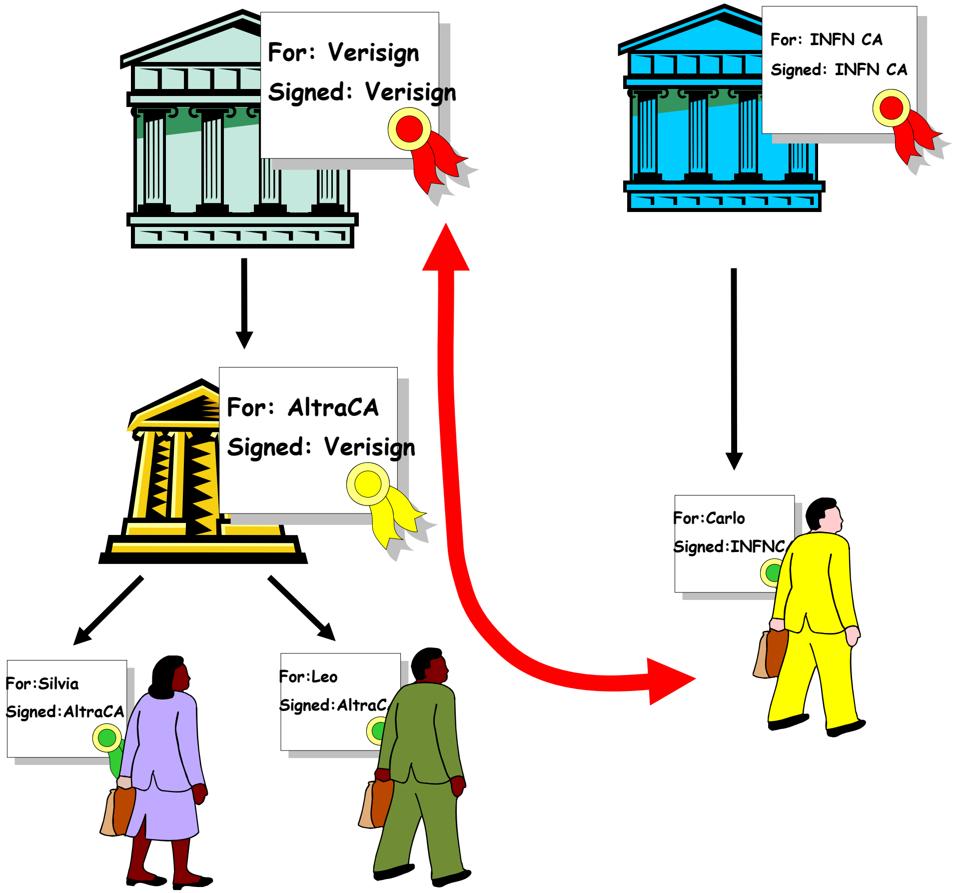


slide based on presentation given by Carl Kesselman at GGF Summer School 2004

Certificate Request



Certificate Chains

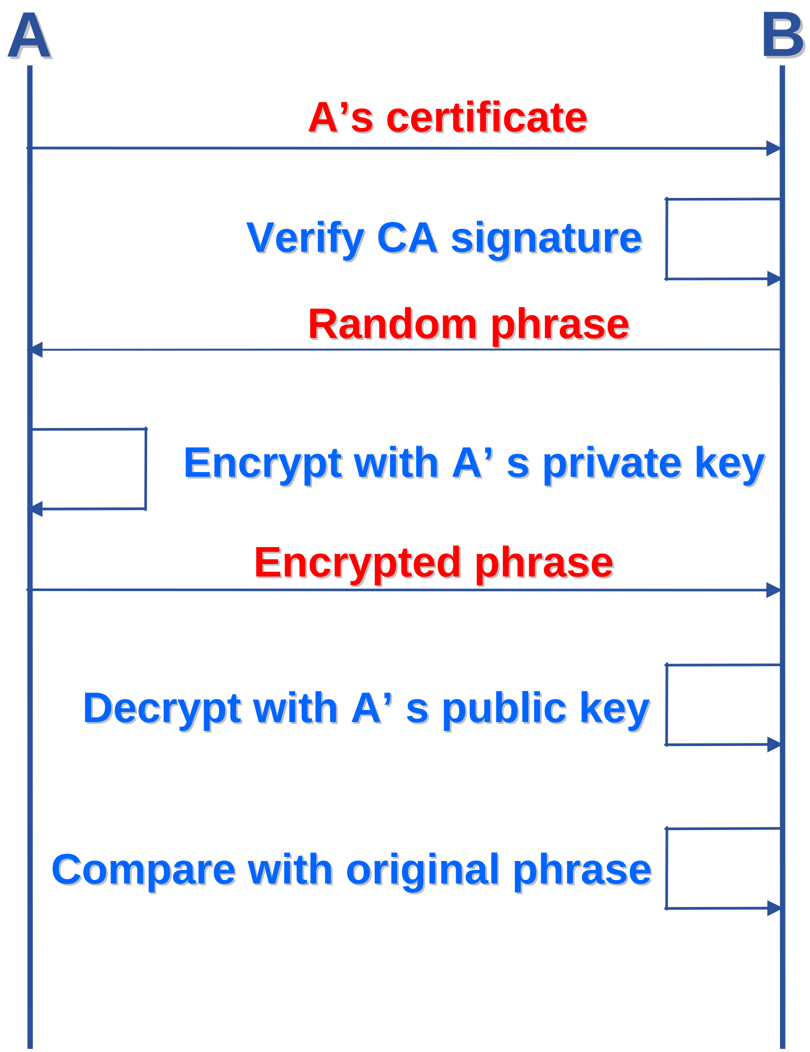


- CA's have their own certificates, too.
- A CA can guarantee for other CA's by signing their certificates
- At the top there is a self-signed certificate (**root certificate**).
- CA certificates are widely published and thus difficult to forge.

Authentication in Grid Security Infrastructure (GSI)

Based on X.509 PKI:

- **every user/host/service has an X.509 certificate;**
- **certificates are signed by trusted (by the local sites) CA's;**
- **every Grid transaction is mutually authenticated:**
  1. A sends his certificate;
  2. B verifies signature in A's certificate using CA public certificate;
  3. B sends to A a challenge string;
  4. A encrypts the challenge string with his private key;
  5. A sends encrypted challenge to B
  6. B uses A's public key to decrypt the challenge.
  7. B compares the decrypted string with the original challenge
  8. If they match, B verified A's identity and A can not repudiate it.



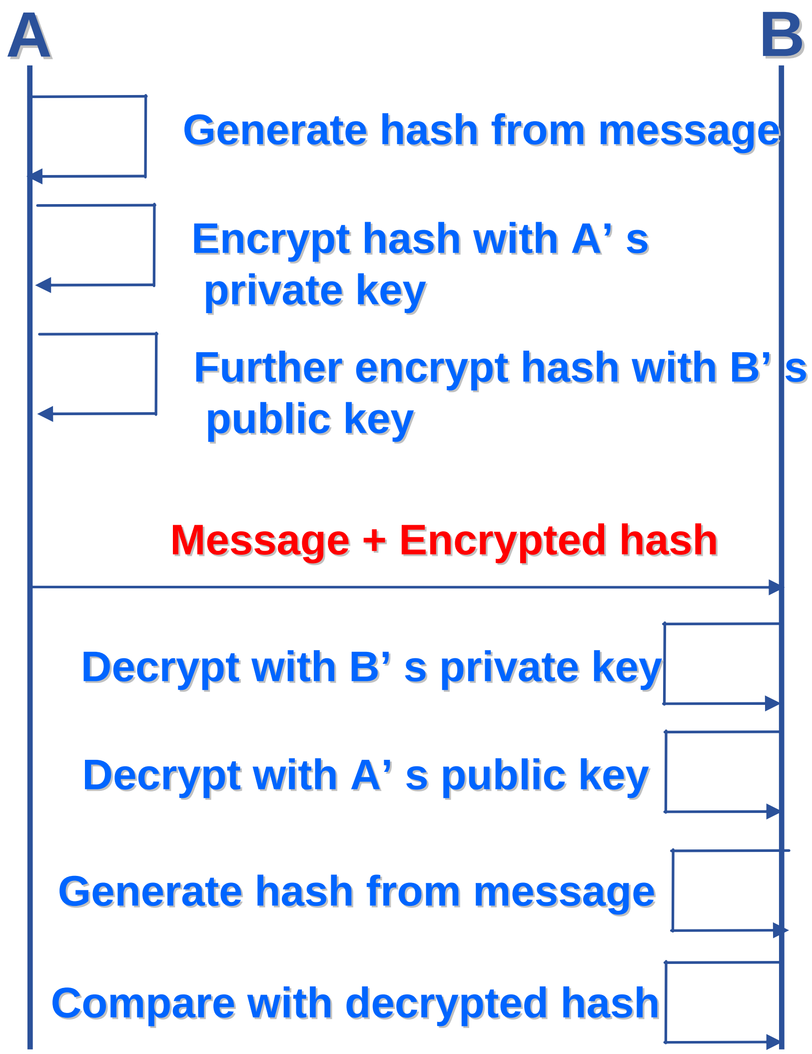
GSI - continued

After A and B authenticated each other,
for A to send a message to B:

- **Default: message integrity checking**
  - Not private – a test for tampering

.

- **For private communication:**
  - Encrypt all the message (not just hash) - Slower



International agreement

- **X.509 Digital certificate is the basis of Authentication in major Grids including EGEE, OSG, Nordugrid, Teragrid**

- ***Certification Authorities*** **(CAs)**
  - ~one per country:
  - each builds network of “Registration Authorities” who issue certificates

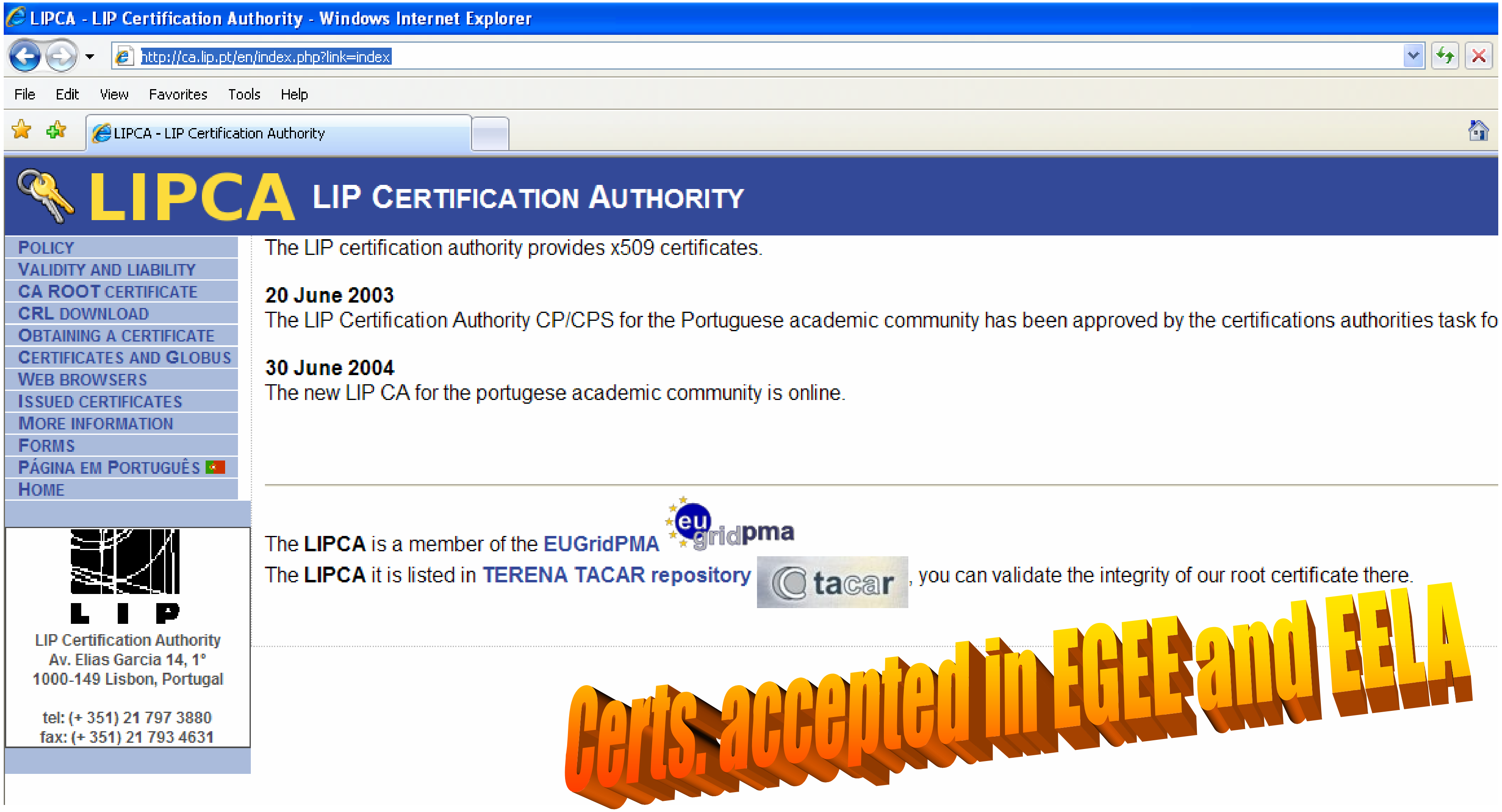
- ***<u>CAs are mutually recognized</u>*** **– to enable international collaboration**

- **International Grid Trust Federation http://www.gridpma.org/**

Grid CA in Portugal

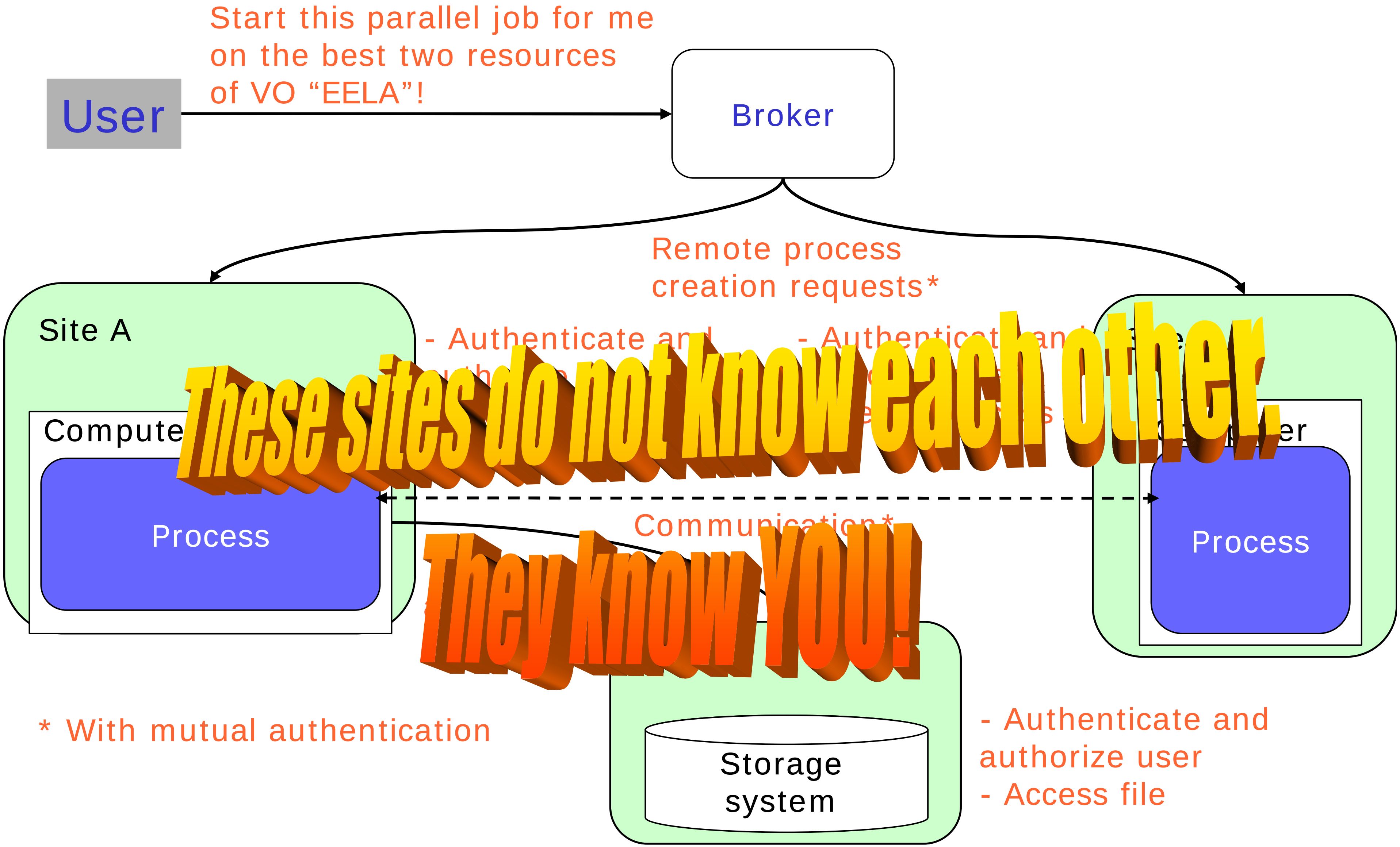
LIP Certification Authority
Av. Elias Garcia 14, 1º
1000-149 Lisbon, Portugal

http://ca.lip.pt



Certs. accepted in EGEE and EELA

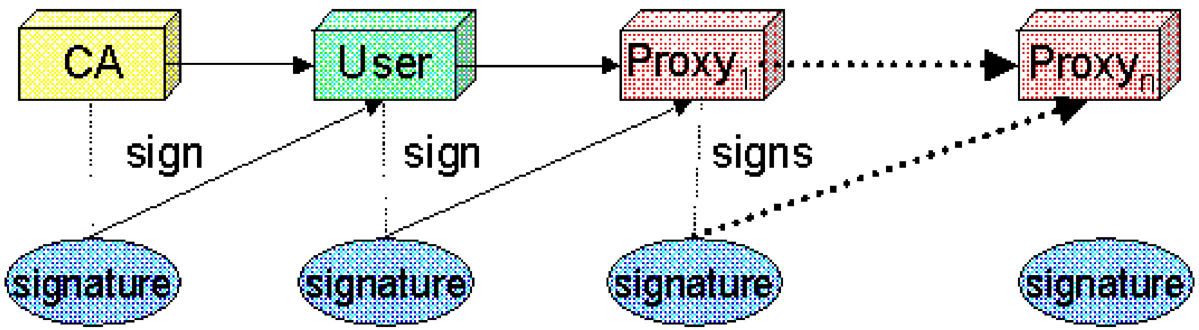
Need for delegation



* With mutual authentication

Delegation by limited proxies

- **Delegation - allows remote process and services to authenticate on behalf of the user**
  - Remote process/service **“impersonates”** the user
- **Achieved by creation of next-level key-pair from a user key-pair: proxy**
  - Proxy has limited lifetime
  - Proxy may be valid for limited operations
- **The client can delegate the proxy to processes**
  - Each service decides whether it accepts proxies for authentication



GSI Proxy Certificate

- **It is created usually by the** grid-proxy-init **command:**

  % grid-proxy-init → login to the Grid

  Enter PEM pass phrase: ****** → private key is protected by a password

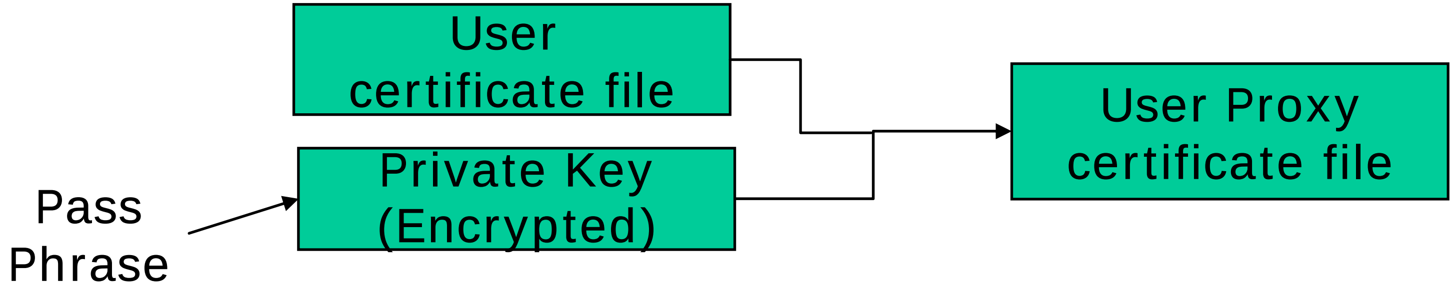
  – Options for grid-proxy-init:
    - -hours <lifetime of credential>
    - -bits <length of key>
    - -help

```
sipos@node40:~/MATRIX> grid-proxy-init
Your identity:
/C=UK/O=eScience/OU=Westminster/L=ComputerScience/CN=gergely sipos
Enter GRID pass phrase for this identity:
Creating proxy ........................................ Done
Your proxy is valid until: Tue Jan  9 23:58:27 2007
```

grid-proxy-init

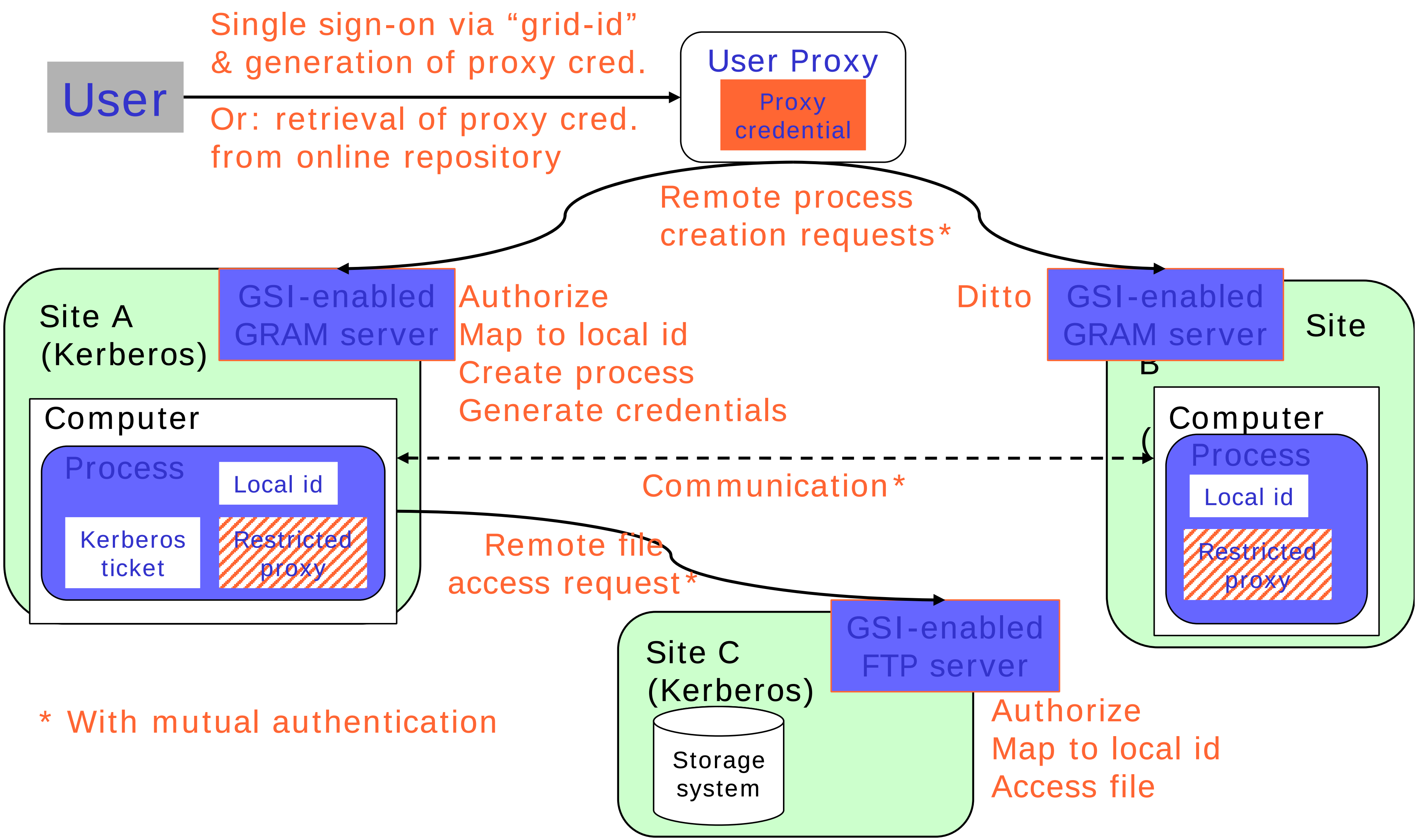
- **User enters pass phrase, which is used to decrypt private key.**

- **Private key is used to sign a proxy certificate with <u>its own</u>, new public/private key pair.**
  - User's private key not exposed after proxy has been signed



- Proxy placed in /tmp
  - the private key of the Proxy is *not* encrypted:
  - stored in local file: must be readable **only** by the owner;
  - proxy lifetime is short (typically 12 h) to minimize security risks.
- NOTE: *No* network traffic during proxy creation!

Proxies in action



Proxy again …

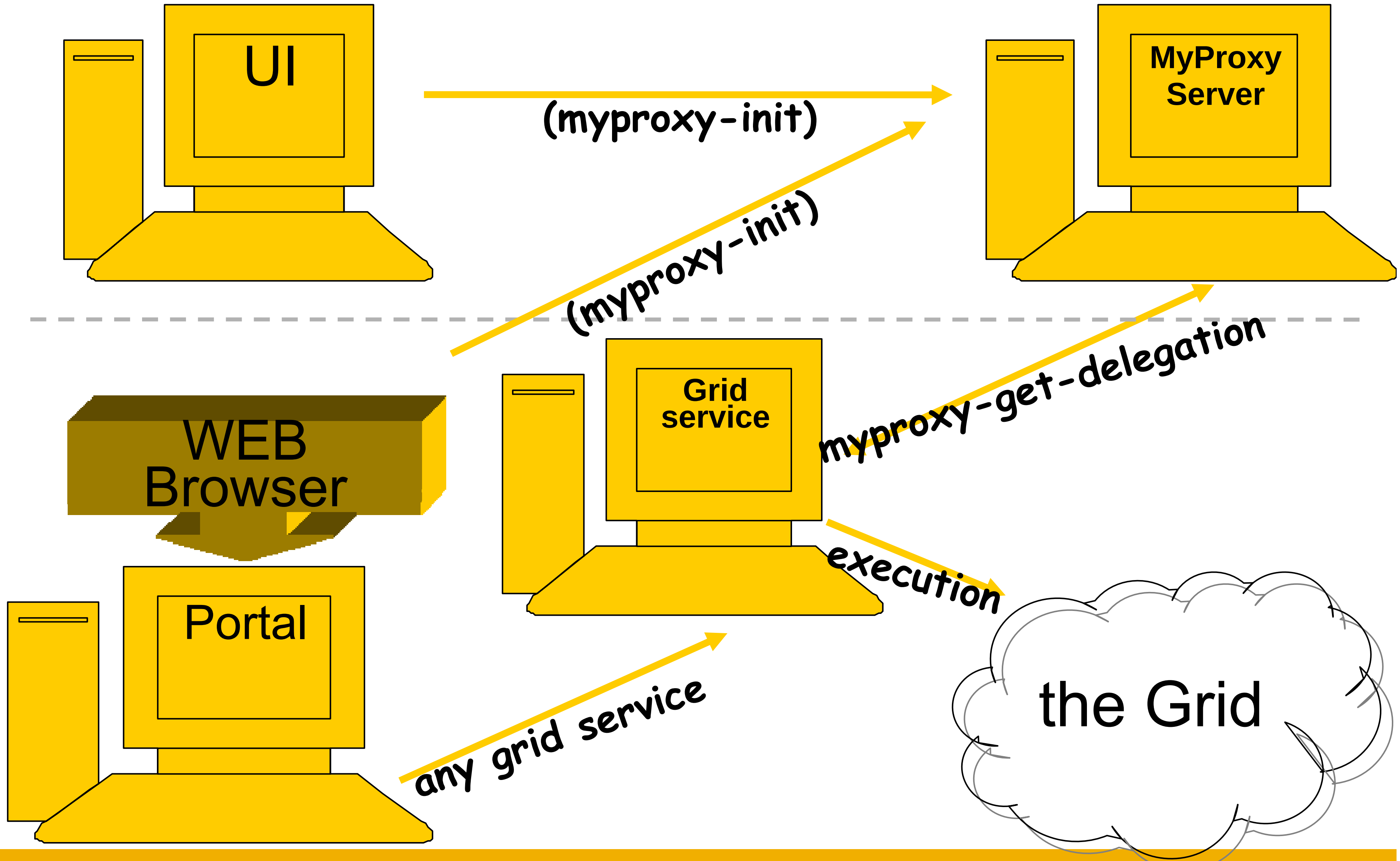
- **grid-proxy-init ≡ “login to the Grid”**
- **To “logout” you have to destroy your proxy:**
  - `grid-proxy-destroy`
  - This does *NOT* destroy any proxies that were delegated from this proxy.
  - You cannot revoke a remote proxy
  - Usually create proxies with short lifetimes
- **To gather information about your proxy:**
  - `grid-proxy-info`
  - Options for printing proxy information

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

MyProxy server

- **You may need:**
  - To interact with a grid from many machines
    - And you realise that you must NOT, EVER leave your certificate where anyone can find and use it….
  - To use a portal, and delegate to the portal the right to act on your behalf (First step is for the portal to make a proxy certificate for you)
  - To run jobs that might last longer than the lifetime of a short-lived proxy
- **Solution: you can store a proxy in a "MyProxy server" and derive a proxy certificate when needed.**
- **Most often used commands:**
  - myproxy-init -s <host_name>
    - *create and store a long term proxy certificate*
  - myproxy-info
    - get information about stored long living proxy
  - myproxy-get-delegation
    - get a new proxy from the MyProxy server
  - myproxy-destroy
    - Remove the proxy from MyProxy

Grid authentication with MyProxy



Managing VOs

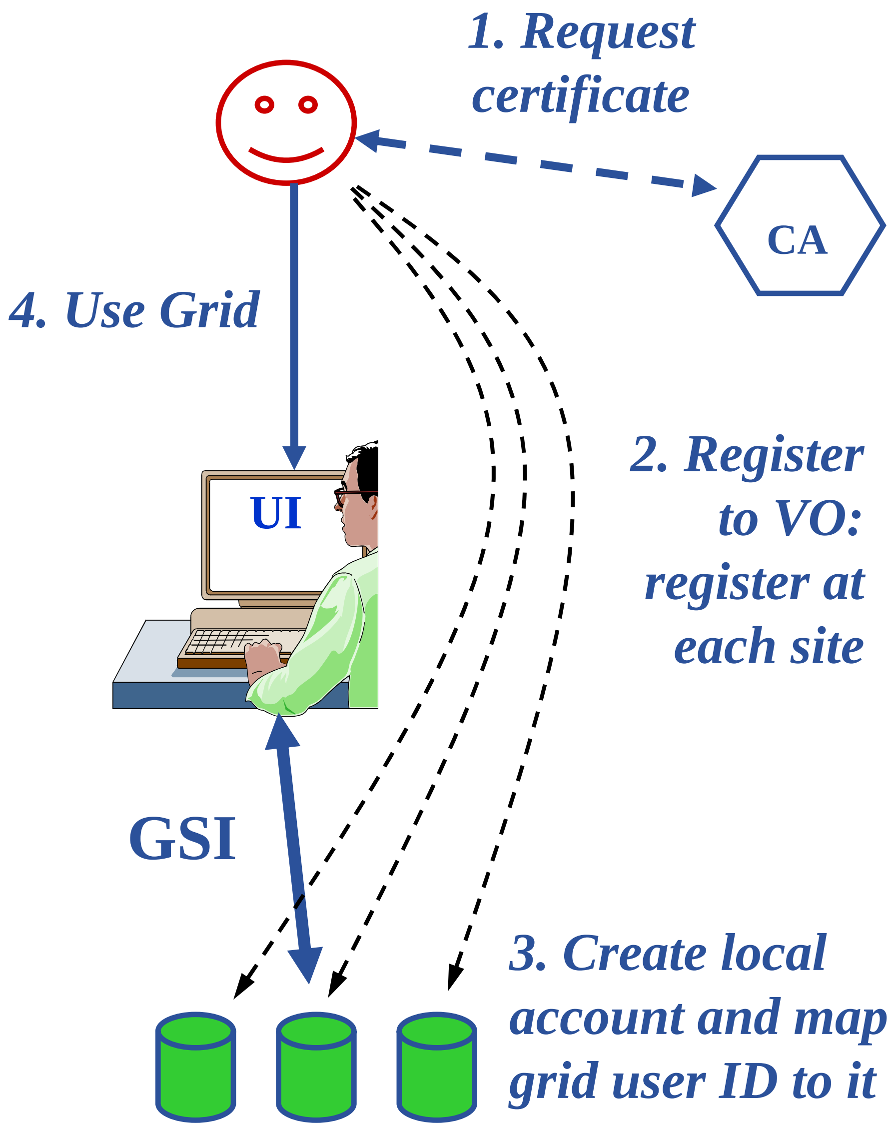
- **Grid activities happen in VOs → users MUST belong to virtual organizations**
  - Users belonging to a collaboration
  - Resources can be accessed by members of the collaboration

- **Authorisation**
  - What are you allowed to do as a VO member?
  - … and how is this controlled??

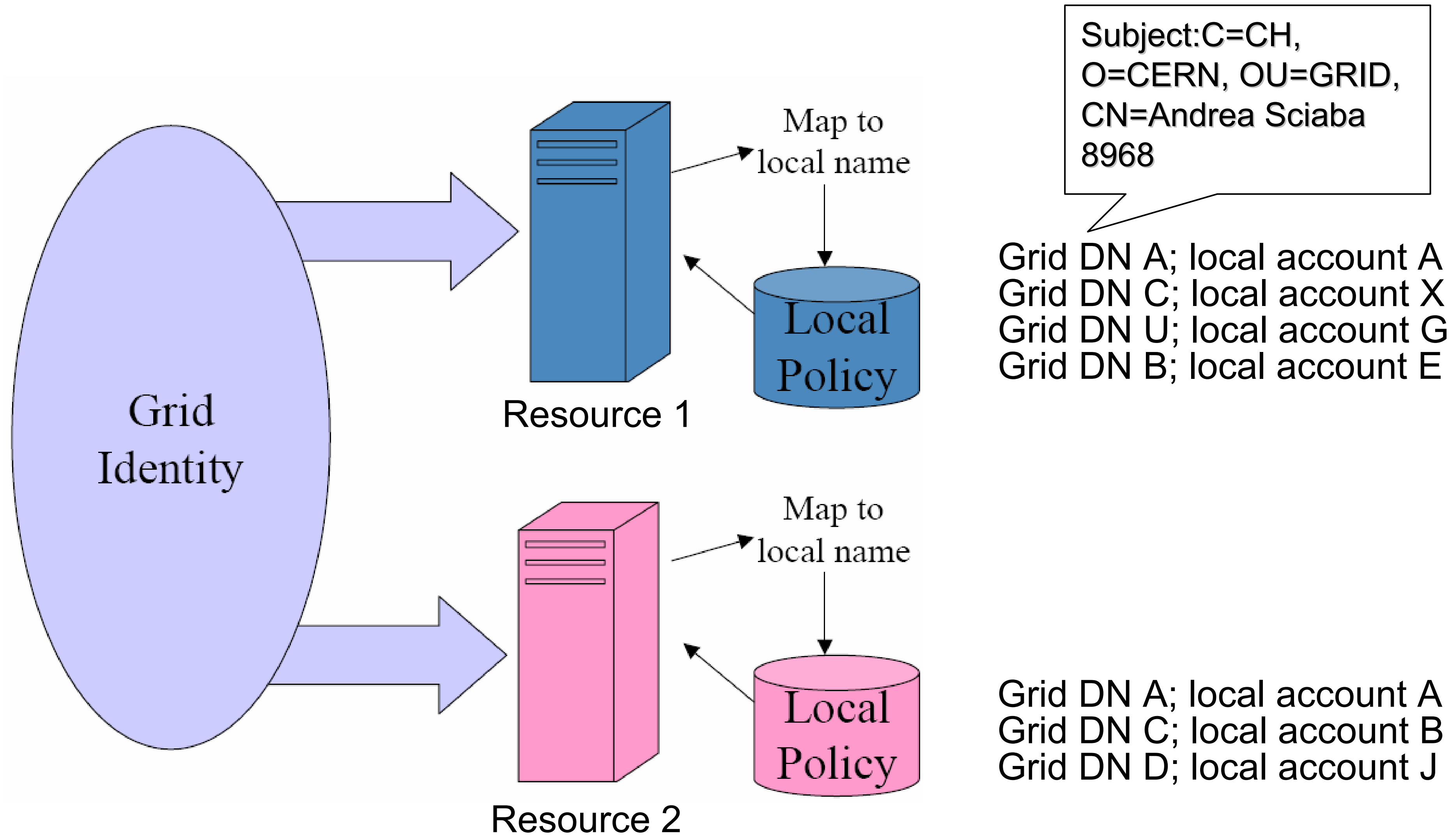
- **Concepts**
  - Globus 2: GridMap files
  - LCG-2: GridMap files with centralised LDAP servers
  - EGEE (gLite): VOMS
  - Globus 4: CAS

Authentication and authorisation: Globus 2 – Gridmap

- **Concept**
  - User receives certificate signed by CA
  - User joins VO at each site
  - A local account is created for the grid user (mapfile)
  - User connects to UI (portal or SSH)
  - Single logon to Grid (create proxy)
  - Grid Security Infrastructure identifies user on the machines

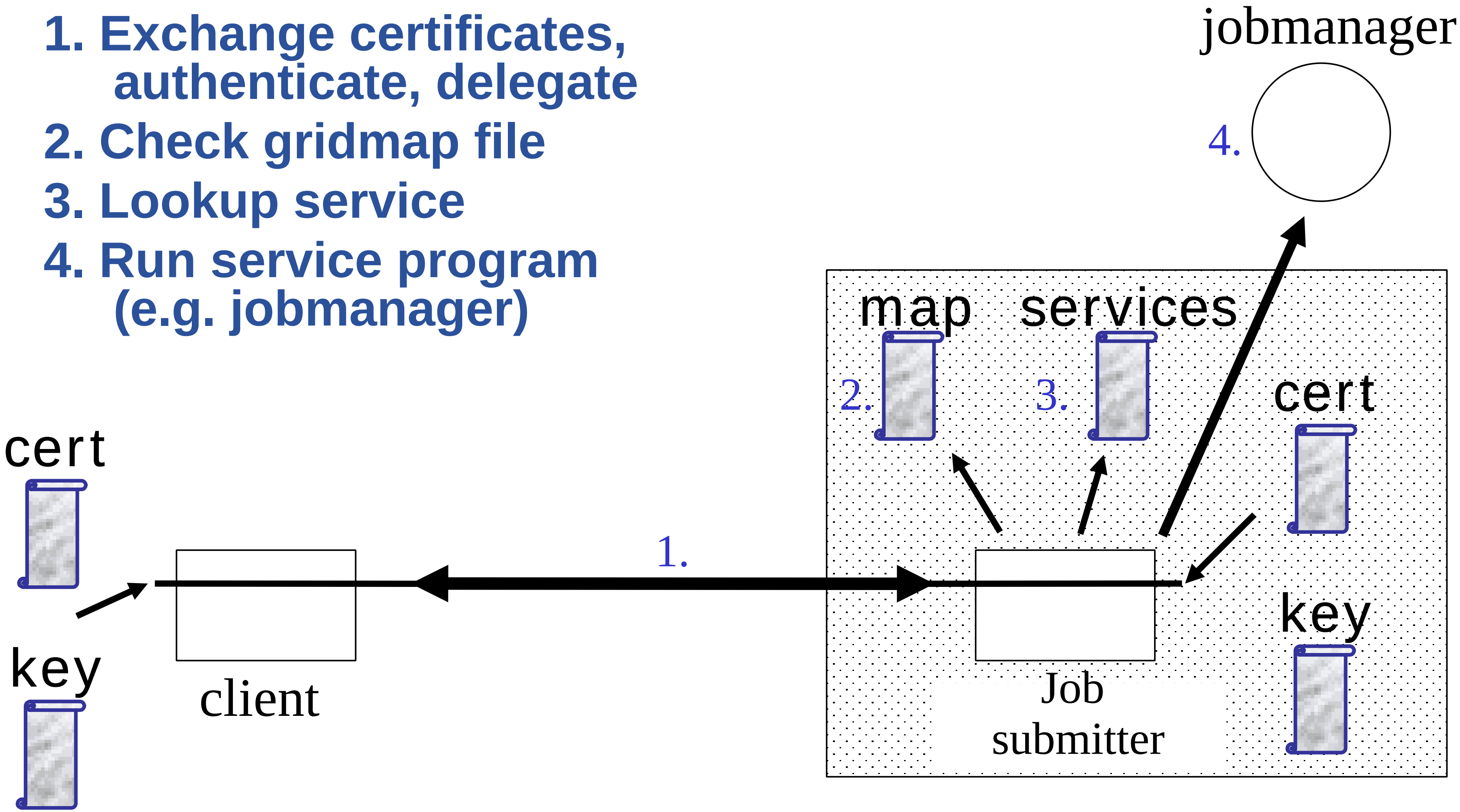


Gridmap files



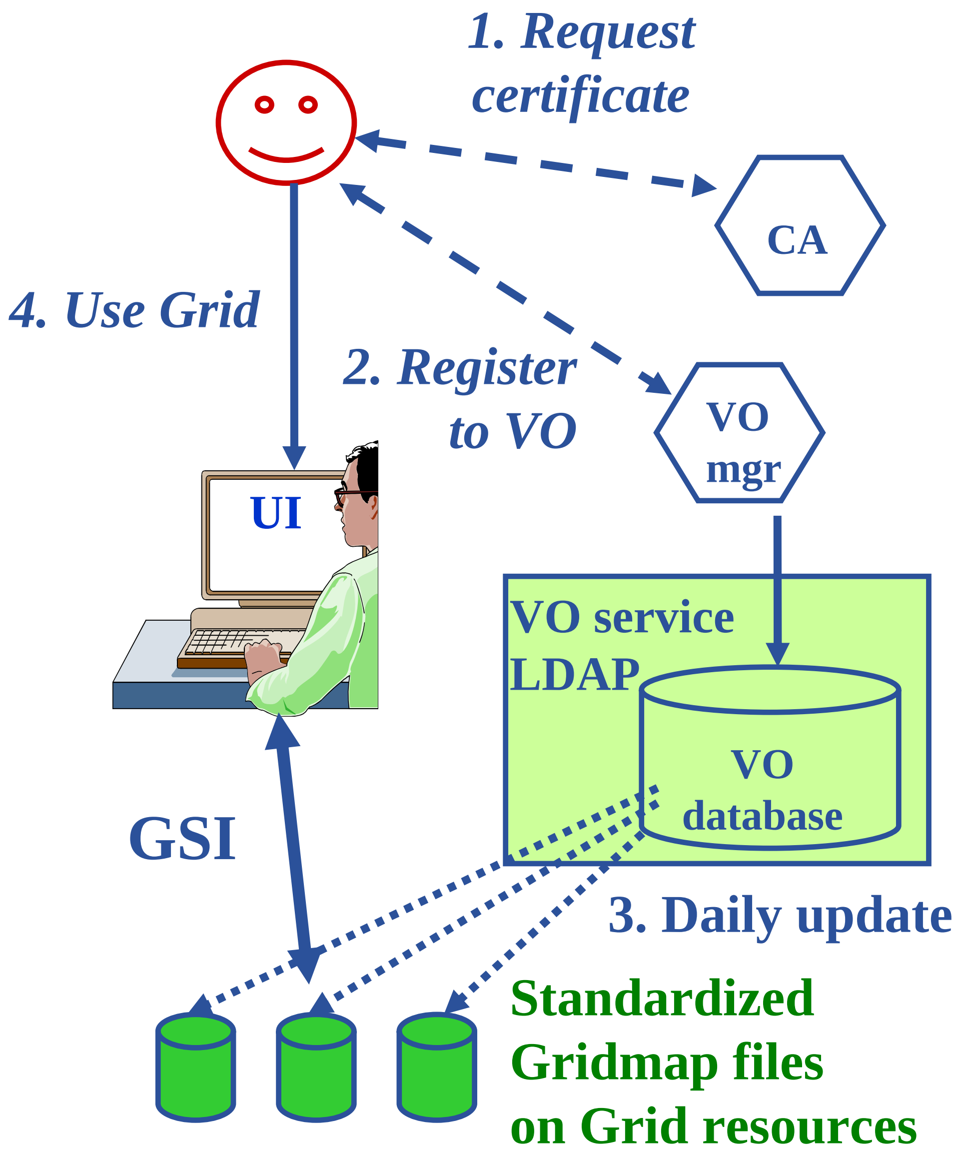
Globus 2 security example

1. Exchange certificates, authenticate, delegate
2. Check gridmap file
3. Lookup service
4. Run service program (e.g. jobmanager)



Authentication and authorisation: EGEE LCG – Gridmap with LDAP

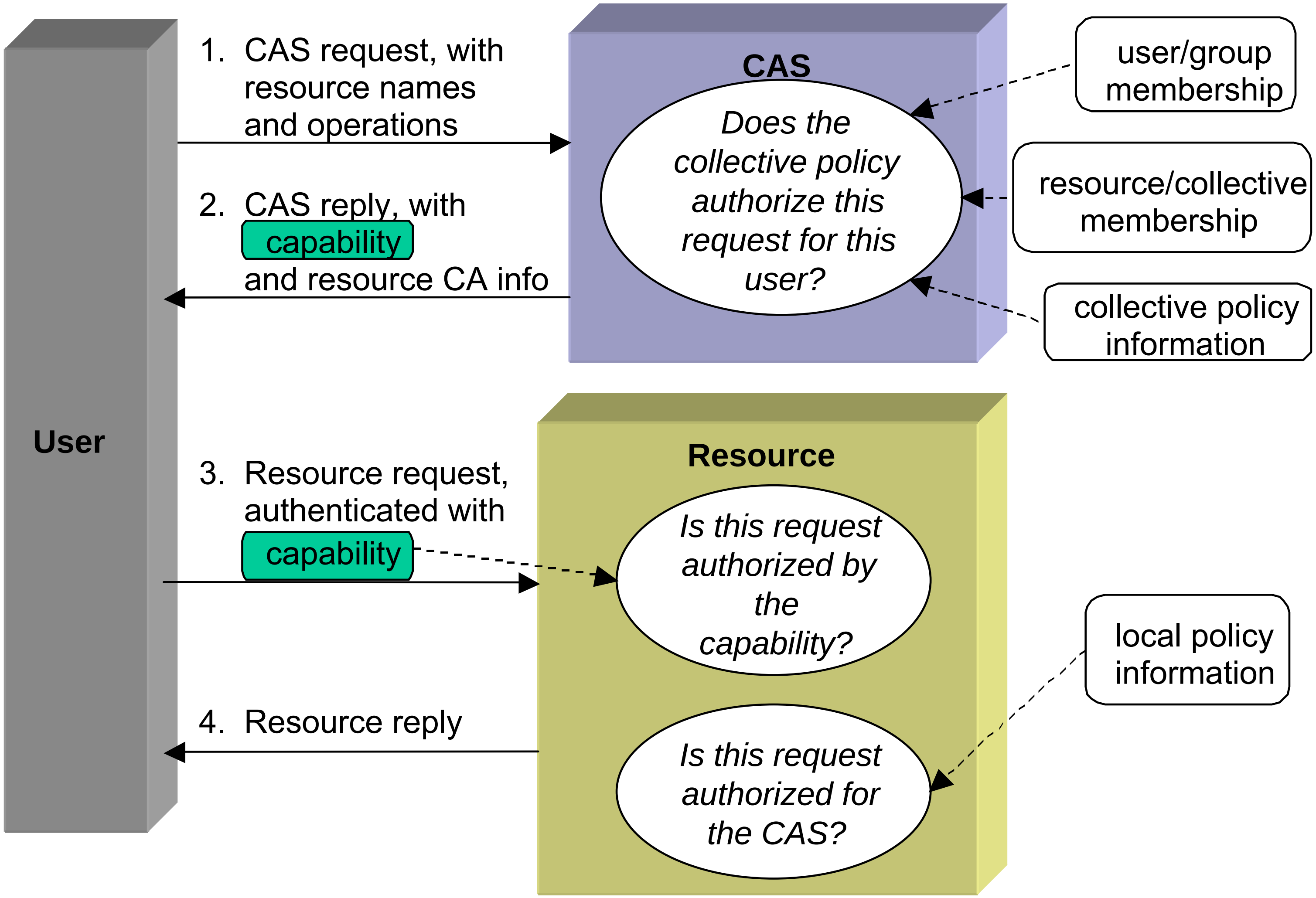
- **Concept**
  - User receives certificate signed by CA
  - User joins VO at a central place
  - VO membership information replicated onto resources
  - User connects to UI (portal or SSH)
  - Single logon to Grid (create proxy)
  - Grid Security Infrastructure identifies user on the machines
  - User identity mapped onto a **pool account**



Problems of Globus 2 gridmap files:

- The grid-mapfile doesn't scale well
- Works only at the resource level, not the collective level
- Large communities that share resources exacerbates authorization issues, which has led to CAS…

Community Authorization service



Evolution of VO management in EGEE

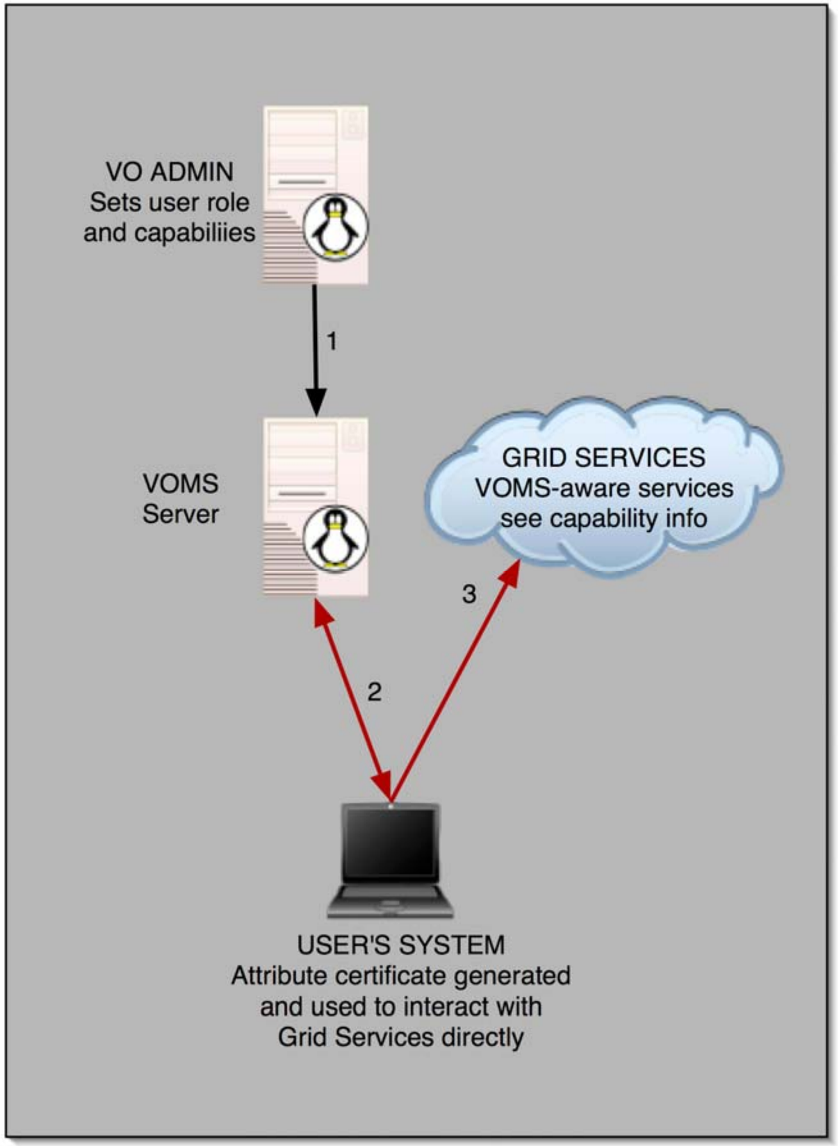
Before VOMS

- **User is authorised as a member of a single VO**
- **All members if a VO have same rights**
- **Gridmapfiles are updated by VO management software: map the user's subject to a local account**
- **grid-proxy-init – derives proxy from certificate – the "sign-on to the grid"**

VOMS

- **VO can have groups**
  - Different rights for each
    - Different groups of experimentalists
    - …
  - Nested groups
- **User can be in multiple VOs**
  - Aggregate rights
- **VO has roles**
  - Assigned to specific purposes
    - E,g. system admin
    - When assume this role
- **Proxy certificate carries the additional attributes**
- **voms-proxy-init**

VOMS



- A community-level group membership system
- Database of user roles
  - Administrative tools
  - Client interface
- voms-proxy-init
  - Uses client interface to produce an attribute certificate (instead of proxy) that includes roles & capabilities signed by VOMS server
  - Works with non-VOMS services, but gives more info to VOMS-aware services
- Allows VOs to centrally manage user roles

Summary

- **Authentication - communication of identity**
  - Grids use X509 certificate based authentication mechanism
    - Private and public key pair
    - Do not let your private key compromised! If it happens let the CA know!
- **GSI = X.509 + delegation**
    - Delegation - A allows B to act on behalf of A
    - Short term proxy: a new public + private key signed by You
    - MyProxy server: proxy storage for portals and long-running jobs
- **Authorisation and VO management: who can do what?**
    - Gridmap: map grid ID to local user
    - Gridmap with LDAP: central user management
    - CAS, VOMS: fine grained VO policies
      - *VOMS – gLite*
      - *CAS – Globus*