

# XtreemOS



*Enabling Linux  
for the Grid*

## **Virtual Organisation Management and Security in Grid-based Operating Systems**

**Alvaro Arenas  
STFC Rutherford Appleton Laboratory, UK**

**XtreemOS Summer School, Oxford, September 2009**

*XtreemOS IP project*

*is funded by the European Commission under contract IST-FP6-033576*



Information Society  
Technologies





- **Basics Security Concepts**
- **Public-Key Encryption**
  
- **Grid Security Infrastructure**
- **Grid Authorisation Systems**
- **Example of Security in Grid Systems**
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





- What is computer security?
  - Computer security deals with the **prevention and detection of unauthorised actions** by user of a computer system
  
- Why is security important in Grids?
  - Grids are open distributed systems
  - Opening our systems to others implies **security risks**



# Basic Security Concepts

- **Confidentiality:** Protection from disclosure to unauthorised persons
- **Privacy:** an aspect of confidentiality, dealing with personal data
- **Authentication:** Assurance of identity of person or originator of data (who?)
- **Authorisation:** Being allowed to perform a particular action
- **Integrity:** Preventing tampering of data
- **Availability:** Legitimate users have access when they need it
- **Non-repudiation:** Originator of communications can't deny it later





# Basic Security Concepts

- **Accountability:** Ability to trace actions to the responsible entity
- **Anonymity:** Inability to trace actions to the responsible entity
- **Auditing:** Provide information for post-mortem analysis of security related events
- **Recoverability:** Ability to recover from security failures
- **Trust:** The degree to which another entity needs to be depended upon to act correctly





- **Policies specifying security**
  - What is it supposed to do?
  
- **Mechanisms implementing security**
  - How does it do it?
  
- **Assurance correctness of security**
  - Does it really work?





# Defensive Strategies

- **Keep the bad guy out**
  - Authentication; firewalls; ...
- **Let him in, but keep him from doing damage**
  - Access control; sandboxing; ...
- **Keep everybody out**
  - Isolation; ...
- **Catch him and prosecute him**
  - Monitoring; auditing
  
- **Morris' rule of computer security**
  - The three golden rules to ensure computer security are:
    - Do not own a computer; do not power it on; and do not use it





# Security Mechanisms

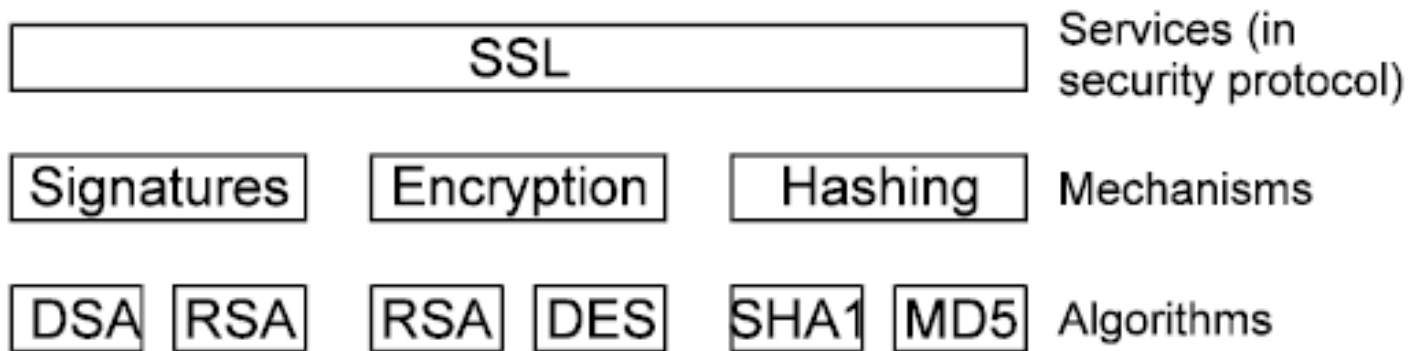
- **Three basic building blocks are used:**
  - **Encryption** is used to provide confidentiality, can also provide authentication and integrity protection
  - **Digital signatures** are used to provide authentication, integrity protection, and non-repudiation
  - **Checksums/hash algorithms** are used to provide integrity protection, can provide authentication
- **One or more security mechanisms are combined to provide a security service**





## Security Services and Mechanisms

- A typical security protocol provides one or more services



- Services are built from mechanisms
- Mechanisms are implemented using algorithms



- Basics Security Concepts
- **Public-Key Encryption**
  
- **Grid Security Infrastructure**
- **Grid Authorisation Systems**
- **Example of Security in Grid Systems**
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





- **Cryptography is a method for hiding the meaning of a message, so that only intended recipients can understand the message**
- **Historical Ciphers**
  - Non-standard hieroglyphics, 1900BC
  - Caesar cipher: letter = letter + 3
    - 'fish' → 'ilvk'





# Cryptography Concepts

- **Plaintext:** original message that we are trying to protect
- **Ciphertext:** message that is actually transmitted
- **Encrypt:** function that converts plaintext into ciphertext
- **Decrypt:** function that converts ciphertext into plaintext
  - Encrypt: Plaintext X Key  $\rightarrow$  Ciphertext
  - Decrypt: Ciphertext X Key  $\rightarrow$  Plaintext
  - Decrypt( Encrypt(p,k), k) = p
  - Computationally infeasible to calculate either function without the key





# Symmetric Encryption

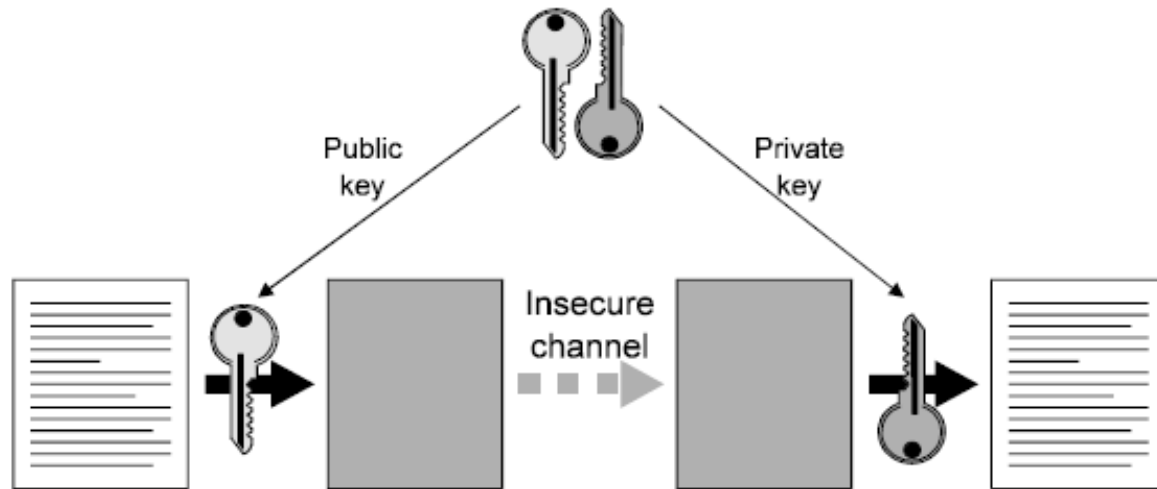
- The same key is used for both encryption and decryption
- If key **k** is known only to Alice and Bob, then Alice can send Bob a secret message **m** by encrypting it with **k**
- This also ensures **authentication** and **integrity**





# Asymmetric Encryption

- Users possess public/private key pairs



- Anyone can encrypt with the public key, only one person can decrypt with the private key



## Key Management Problems

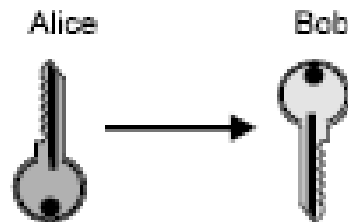
- Key certification
- Distributing keys
  - Obtaining someone else's public key
  - Distributing your own public key
- Establishing a shared key with another party
  - Confidentiality: Is it really known only to the other party?
  - Authentication: Is it really shared with the intended party?
- Key storage
  - Secure storage of keys
- Revocation
  - Revoking published keys
  - Determining whether a published key is still valid



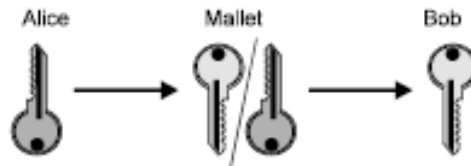


# Key Distribution Problems

- Alice retains the private key and sends the public key to Bob



- Mallet intercepts the key and substitutes his own key

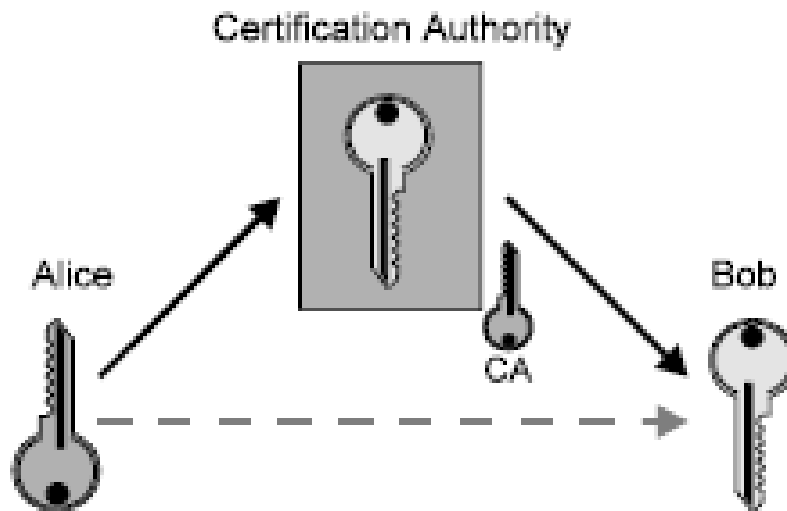


- Mallet can decrypt all traffic and generate fake signed message



# Certification Authority

- A Certification Authority (CA) solves this problem



- CA signs Alice's key to guarantee its authenticity to Bob
  - Mallet can't substitute his key since the CA won't sign it



## Certification Authorities (CAs)

- CAs are entities that are trusted by different systems
- The CAs are responsible for certifying the public keys of different users who subscribe to the CA
  - **Guarantee the connection between a key and an end entity**
- An end entity is
  - **Person, role (“Director of marketing”), organisation, pseudonym, a piece of hardware or software, an account (bank or credit card)**
- CA manages key lifecycle: creation, store, delete, renew





## Obtaining a Certificate (1)

1. Alice generates a key pair and signs the public key and identification information with the private key
  - Proves that Alice holds the private key corresponding to the public key
  - Protects the public key and ID information while in transit to the CA
- § CA verifies Alice's signature on the key and ID information
- § Optional: CA verifies Alice's ID through out-of-band means
  - email/phone callback
  - Business/credit bureau records, in-house records



## Obtaining a Certificate (2)

1. CA signs the public key and ID with the CA key, creating a certificate
  - **CA has certified the binding between the key and ID**
2. Alice verifies the key, ID, and CA's signature
  - **Ensures the CA didn't alter the key or ID**
  - **Protects the certificate in transit**
3. Alice and/or the CA publish the certificate





# Public Key Infrastructure (PKI)

- PKI allows one to know that a given key belongs to a given user
  - **Based on asymmetric encryption**
- The public key is given to the world encapsulated in a **X.509 certificate**
- Certificates: Similar to passport or driver license
  - **Identity signed by a trusted party (a CA)**





- Basics Security Concepts
- Public-Key Encryption
  
- **Grid Security Infrastructure**
- **Grid Authorisation Systems**
- **Example of Security in Grid Systems**
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





- **Grids concern with ...**
- “Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations.”
  - From *The Anatomy of the Grid*
- **So Grid Security is security to enable VOs**
  - Access to shared services
    - Cross-domain authentication, authorisation, accounting, billing
  - Support multi-user collaboration
    - Organised in one or more ‘**Virtual Organisations**’
    - May contain individuals acting alone – their home organisation administration need not necessarily know about all activities
  - Leave resource owner always in control





- **A VO is**
  - a temporary or permanent coalition of geographically **dispersed** and **autonomous** participants
    - including individual and/or organisations,
  - who agree to **share resources** in the system in order to fulfill their tasks
    - e.g. running jobs, sharing applications, accessing data
  
- **Properties**
  - Geographically distributed
  - Autonomously governed
  - Short-termed or long-term
  - Static or dynamics





# Issues in making Grid security work

- **Resources may be valuable & the problems being solved sensitive**
  - Both users and resources need to be careful
- **Resources & users often located in distinct administrative domains**
  - Can't assume cross-organizational trust agreements
  - Different mechanisms & credentials
- **Dynamic formation and management of communities (VOs)**
  - Large, dynamic, unpredictable, self-managed ...
- **Interactions are not just client-server, but service-to-service on behalf of the user**
  - Requires delegation of rights by user to service
- **Policy from sites, VO, users need to be combined**
  - Varying formats
- **Want to hide as much as possible from applications!**

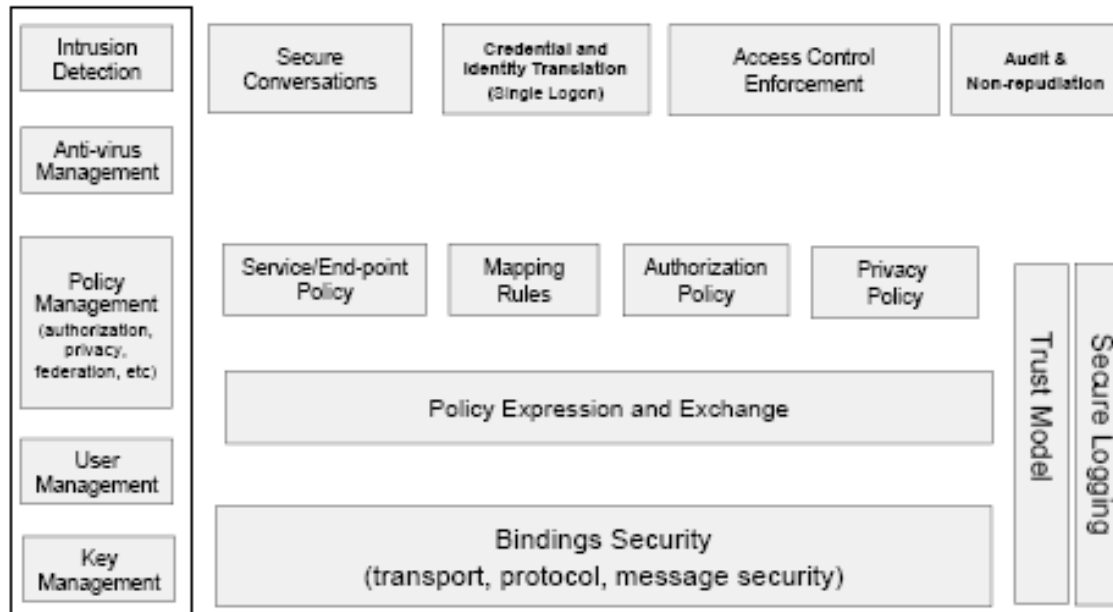




- **Secure functionality should be cast as services**
  - allowing applications to locate and use the particular functionality they need
  
- **Leverage on existing and emerging WS security standards**
  - Authentication service;
  - Identity mapping service
  - Authorisation service;
  - VO Policy service;
  - Credential conversion service;
  - Audit service; etc



## Components of the Grid Security Model



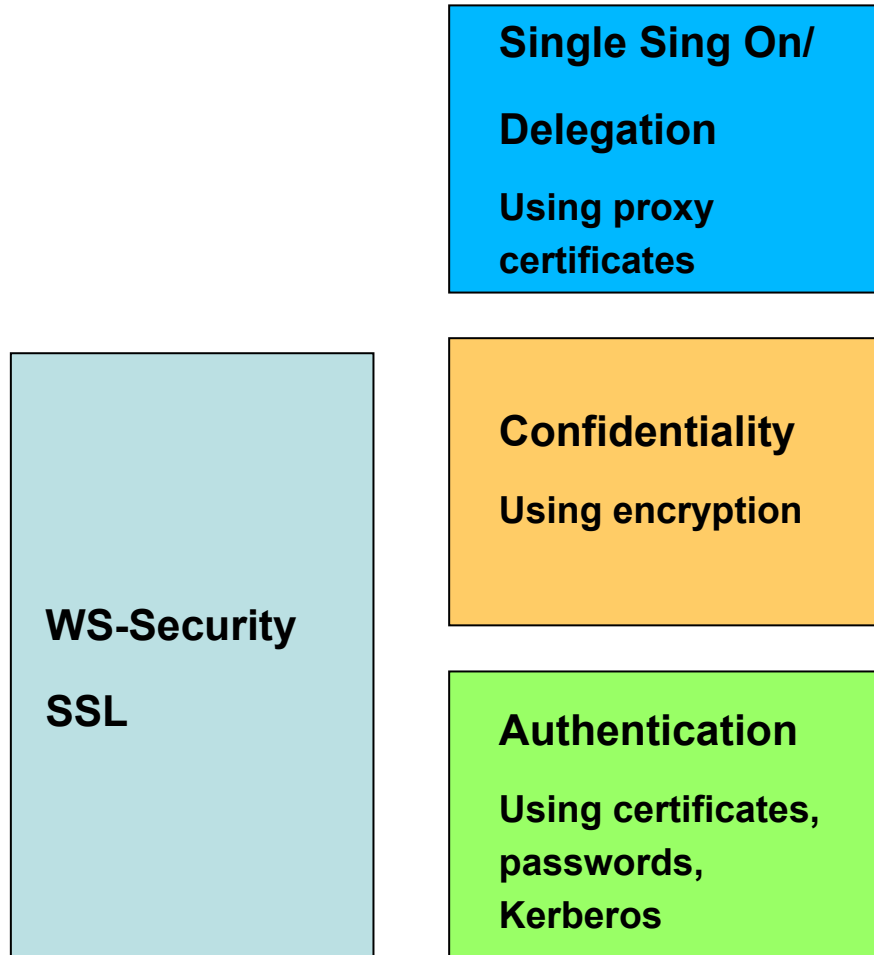


- **A reference specification for Grid security architectures**
- **Protocols and APIs to address Grid security needs**
- **Based on public-key encryption technology**
  - SSL protocol for authentication, message protection
  - X.509 certificates
- **Each user as a Grid id, a private key, and a certificate signed by a CA**
  
- **First implementation – in the Globus Toolkit**





# High-Level View of GSI





- **Certificate-based authentication (PKI)**
- **GSI certificate includes the following information**
  - Subject name;
  - public key belonging to the subject;
  - Identity of the CA; and
  - Digital signature of the named CA
- **Certificates are obtained as shown before**



# Mutual Authentication

- **Two hosts mutually authenticate each other is both of them trust the CA (or any other third party)**
  1. Alice sets up a connection with Bob
  2. Alice then sends her certificate over to Bob for authentication
  3. After receiving the information from Alice, Bob verifies the message
  4. Bob create a random number or message and sends it to Alice
  5. Alice receives the message, encrypts it with her private key and send it back to Bob
  6. Bob decrypts the message and check it
  7. Bob trusts the identity of Alice





# Single Sign On and Delegation

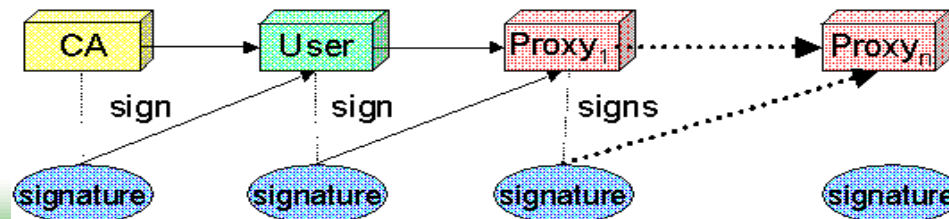
- **Jobs require access to multiple resources**
  - To authenticate with your certificate directly you would have to type a passphrase every time
- **Need to automate access to other resources: Authenticate Once**
  - Important for complex applications that need to use Grid resources
  - Allows remote processes and resources to act on user's behalf - also known as **delegation**
  - Also you need a way to send you VO details (Groups membership, roles and capabilities) across
- **Solution adopted in the GSI: proxy certificates**
  - A temporary key pair
  - in a temporary certificate signed by your 'long term' private key
  - valid for a limited time (default: 12 hours), but can be renewed





# Delegation and limited proxy

- **Delegation = remote creation of a (second level) proxy credential**
  - Agents and brokers act on behalf of users, with (a subset of) their rights
  - you don't know beforehand where your task will end up
  - definition of attribute release policies to these 'unknown' entities is virtually impossible
  - need to support restricted delegation
- **Allows remote process to authenticate on behalf of the user**
- **The client can elect to delegate a "limited proxy"**
  - Each service decides whether it will allow authentication with a limited proxy
  - The proxy can also be used as a container for other elements (e.g. extensions that contain user credentials)





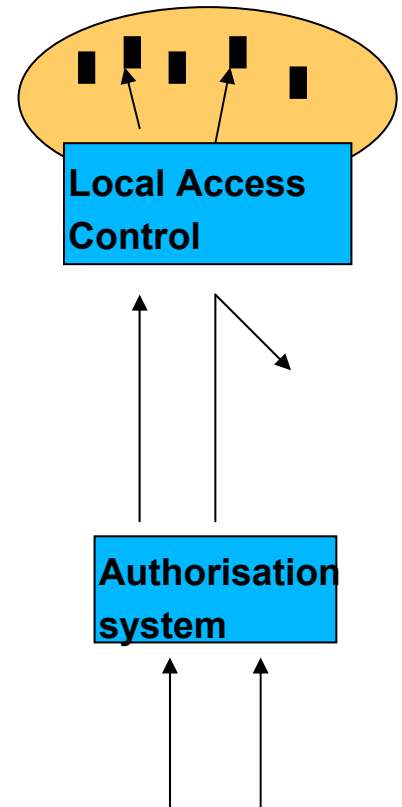
- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- **Grid Authorisation Systems**
- **Example of Security in Grid Systems**
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





# Authorisation vs Access Control

- We want to ensure that only authorised subjects (users or jobs) have access to certain objects (resources)
- **Authorisation:** act of providing and checking the authority of a subject on a specific set of objects
- **Access control:** way of controlling the access to a set of objects





- **Discretionary Access Control**
  - The owner of each resource decides who has access to it;
  
- **Mandatory Access Control**
  - A system-wide policy decrees who has access to each object
  
- **Role-Based Access Control**
  - Permissions are associated with roles; and users are made members of appropriate roles thereby acquiring roles' permissions

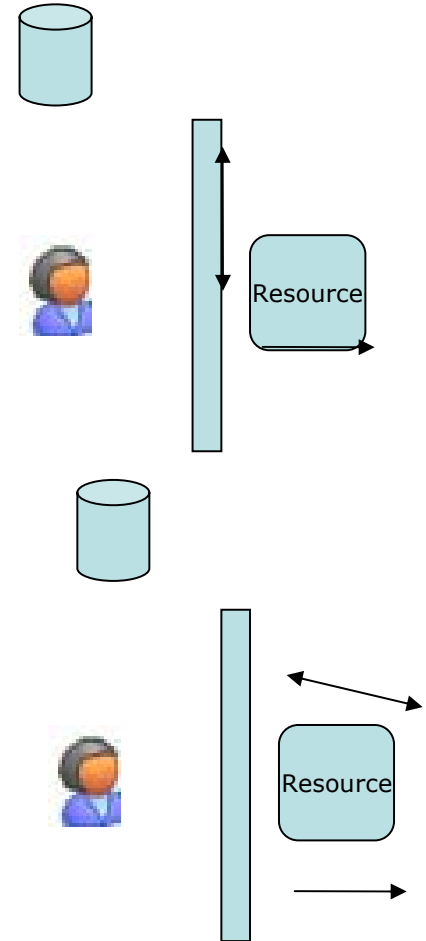


## ■ Push Authorization

- Produce a proof (proxy certificate) that you are authorized to use the requested resource
- Bring (push) this proof to an access control point, who will make sure the proof is valid

## ■ Pull Authorization

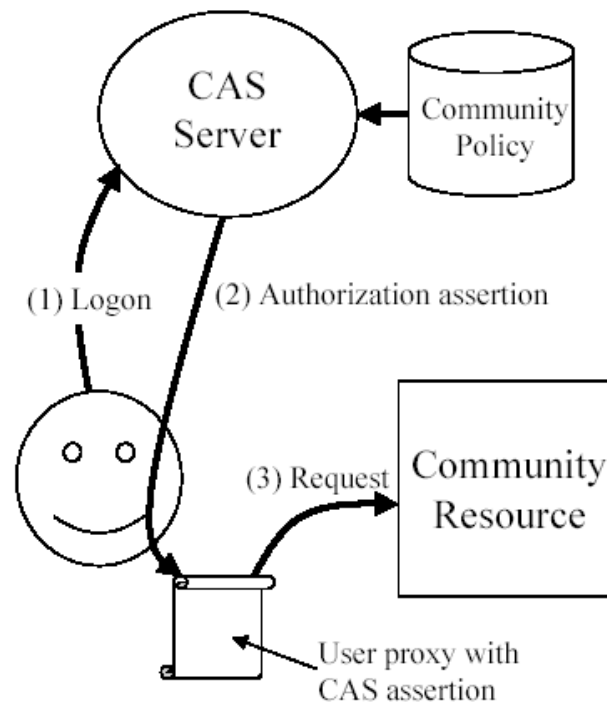
- Go to the access control point and ask for access (just showing who you are, showing your ID, nothing about what you're authorized to do).
- The access controller uses your ID to pull the access policies from a database.
- Depending on the access policies, you're authorized to run your program on the resources, or parts of the resources, or not at all.





# CAS – Community Authorization Service

- **CAS manages a data base of VO policies**
  - What each grid user can do as VO member
- **A Grid user contacts CAS**
  - Proxy cert. is exploited for authentication on CAS
  - CAS returns a signed policy assertion for the user
- **Grid user creates a new proxy that embeds the CAS assertion**
- **Exploits this proxy certificate to access services**





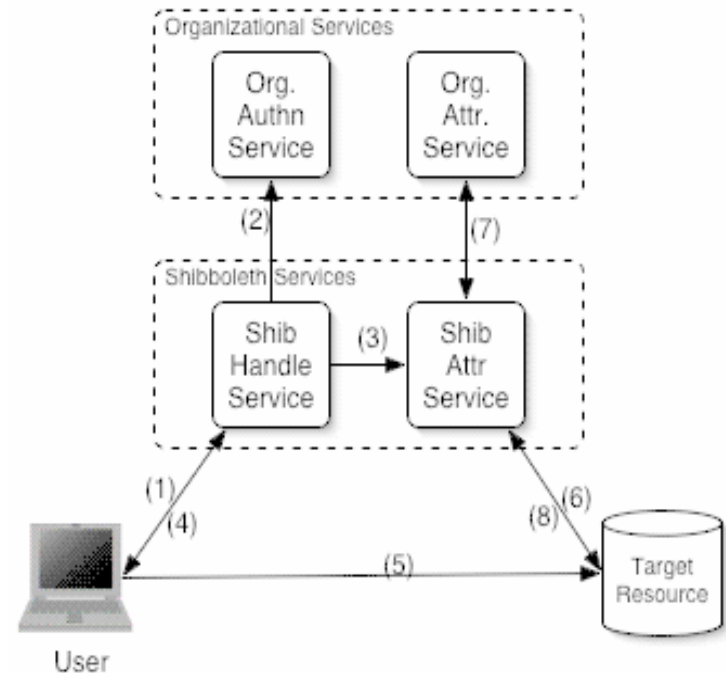
- **VOMS = Virtual Organization Membership Service**
  - Developed by the EU DataGrid and DataTag projects
  
- **Provides a way to delegate the authorization of users to VO managers:**
  - The user credentials are associated with a set of membership information (VO name, group, roles, generic attributes)
  - The information is stored in an account database
  - The VOMS service can provide signed assertions containing these attributes
  
- **VOMS allows for dynamic & “fine-grained” access control on Grid resources**





## Attribute Authority Service for distributed cross domain environments

- User authentication is done on a local Shibboleth server that returns a handle to the user
- Users use the handle to access remote services
- Remote services use the user handle to retrieve user's attributes from a Shibboleth Attribute Server
- Remote Service determines user access rights exploiting his attributes





- **Distributed authorization system where certificates are created independently by distinct stakeholders**
  
- **Certificates type**
  - Authentication
  - Policy certificates
    - Specifies the Source of Authority for resources
  - Use condition certificates
    - Constraints to access resources
  - Attribute certificates
  - Assign attribute to users
  
- **For each access Akenti finds (pulls) all the relevant authorization policy on LDAP/Akenti servers**



- **Role Based Access Control Auth. Infrastructure**
- **Runs as Globus Service**
  - SAML Callout Authz Service
- **X.509 Attribute certificates to assign roles to users**
  - Issued by an Attribute Authority
- **X.509 Attribute certificates to store the policy**
  - Definition of roles and permissions (XML)
  - Issued by the Source of Authority
- **LDAP server(s) to store Attribute Certificates**



- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- Grid Authorisation Systems
- **Example of Security in Grid Systems**
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





- **Open source middleware for computing grids**
- **It has evolved to an implementation based on web services**
  - implements the Open Grid Services Architecture (OGSA) and the Web Services Resource Framework (WSRF)
  - includes components that provide resource management, data management, security, information infrastructure, communication, fault detection etc.
- **Probably the most widely used Grid middleware**





- **Implements the Grid Security Infrastructure (GSI)**
- **X.509 proxy certificates**
  - Enable single sign-on
  - The users can dynamically assign rights to services
- **MyProxy – storing and retrieving GSI credentials**
  - “convert” from username/passphrase to a GSI certificates
  - Renewing credentials for long-running tasks
  - Support for One Time Password



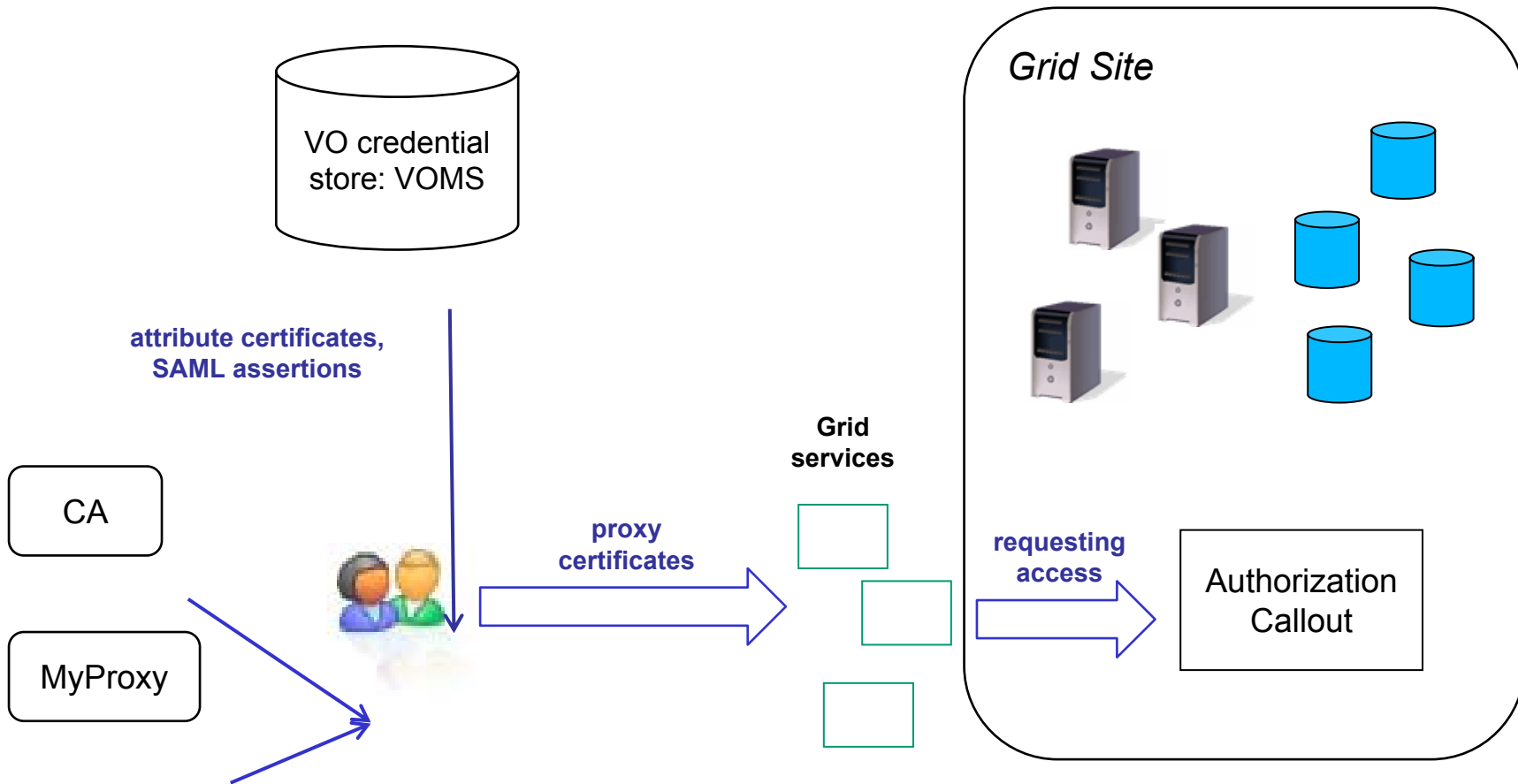


- **GridShib – GT integration with Shibboleth**
  - Policy controlled attribute service
  - Interactions through WS protocols
- **Authorization – many types of policy information:**
  - Attribute assertions: VOMS, X509, Permis, Shibboleth, SAML, Kerberos, ...
  - Authorization assertions: XACML, SAML, CAS, XCAP, Permis, ...
- **Authorization processing**
  - Policy Decision Point (PDP) abstraction
  - after validation, all attribute assertions are mapped to XACML Request Context Attribute format
  - mechanism-specific PDP instances are created for each authorization assertion and call-out service





## Globus Toolkit – Security flow





- **gLite: Grid middleware developed at CERN, in the context of the LHC experiments**
  
- **Used by more than 15000 researchers around the world**
  
- **gLite components:**
  - User Interface (UI)
  - Computing Element (CE)
  - Storage Element (SE)
  - Resource Broker (RB)
  - Information Service (IS)





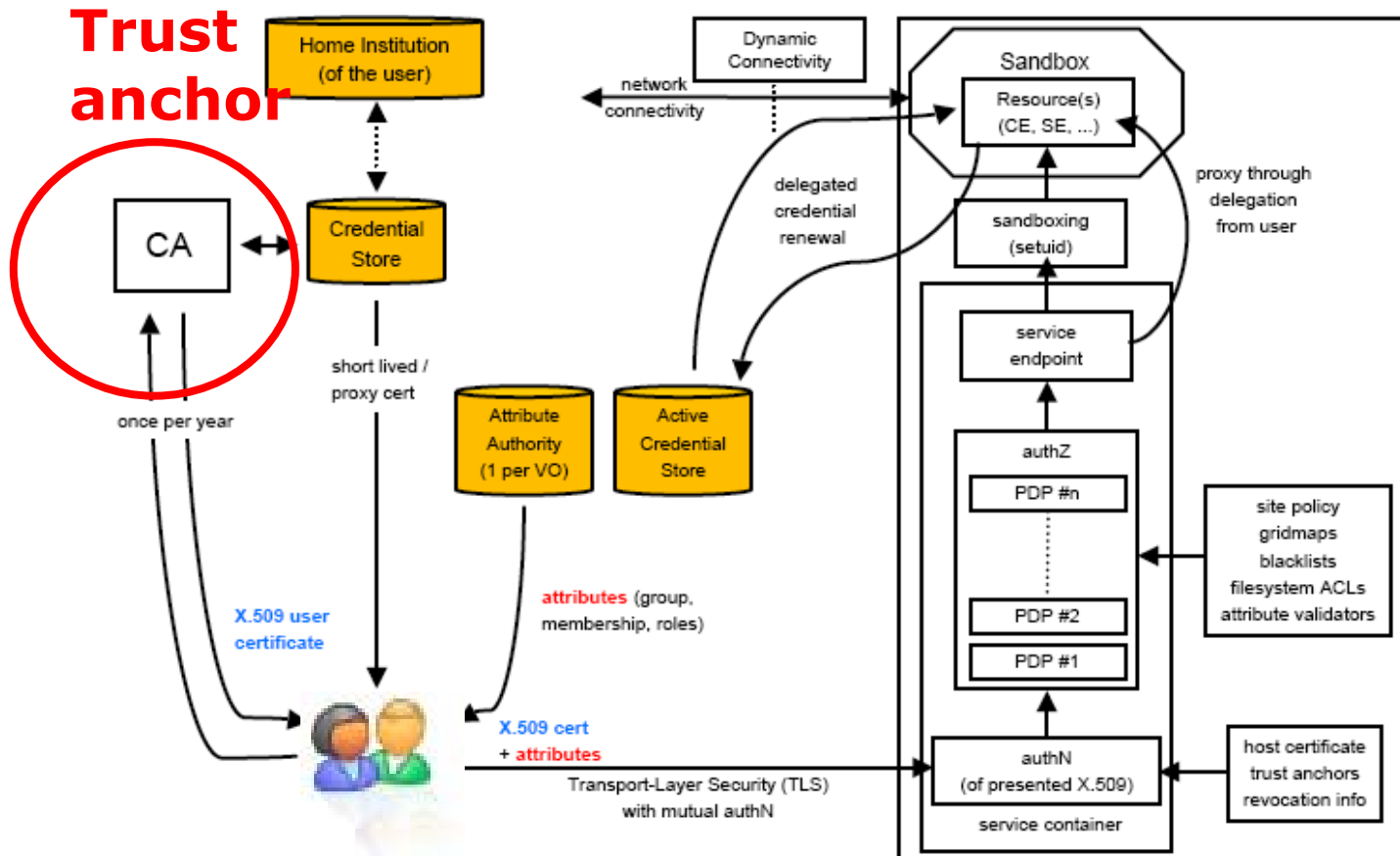
# Security in gLite - Overview

- **Security system based on X.509 certificates**
- **Single sign-on enabled by proxy certificates**
- **VOMS service used to stored information about groups, roles and capabilities for the users**
- **Local Centre Authorization Service (LCAS)**
  - Checks if the user is authorized or banned at the site
  - And if the site can currently accept jobs
- **Local Credential Mapping Service (LCMAPS)**
  - Maps the Grid credentials (including groups, roles etc.) to local credentials





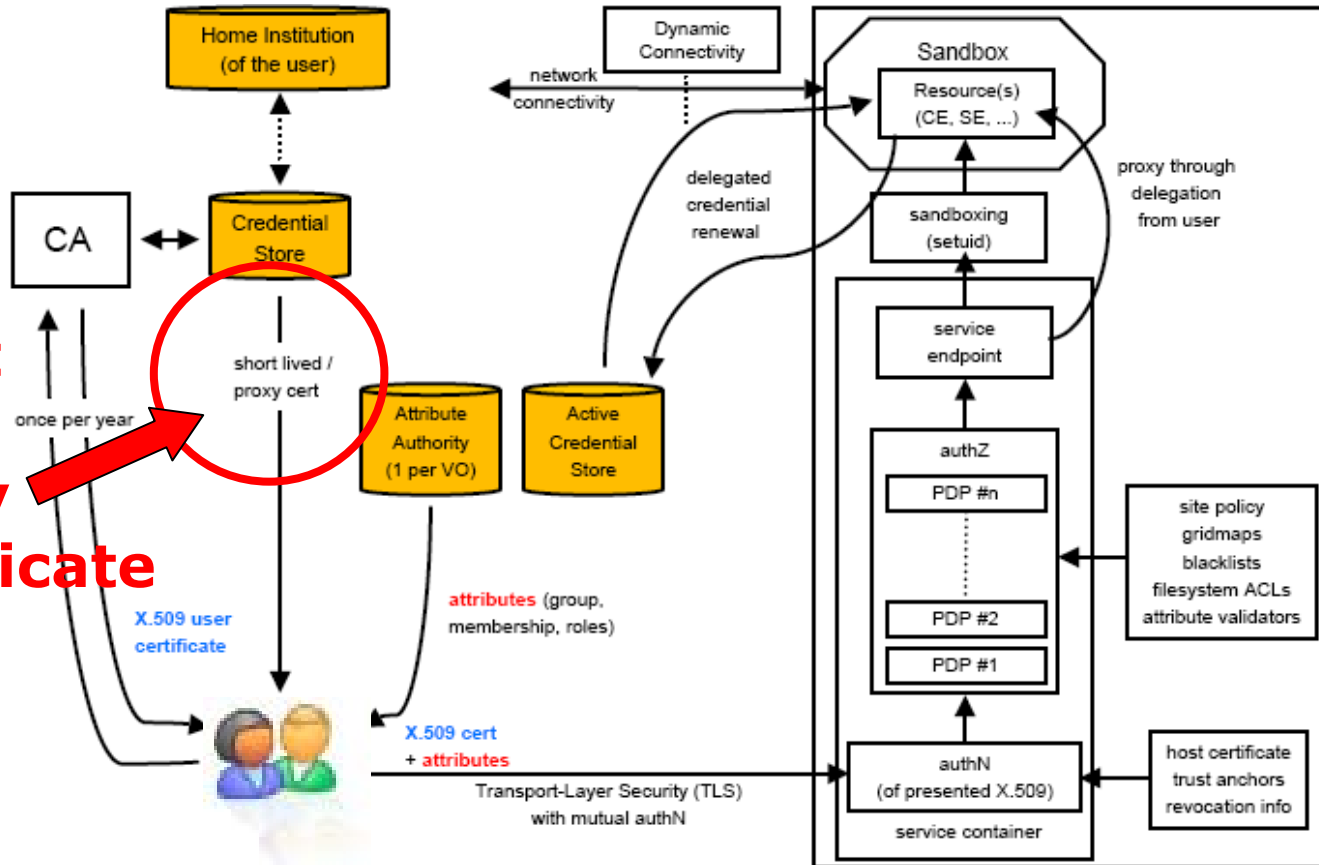
# gLite - Security flow (1)





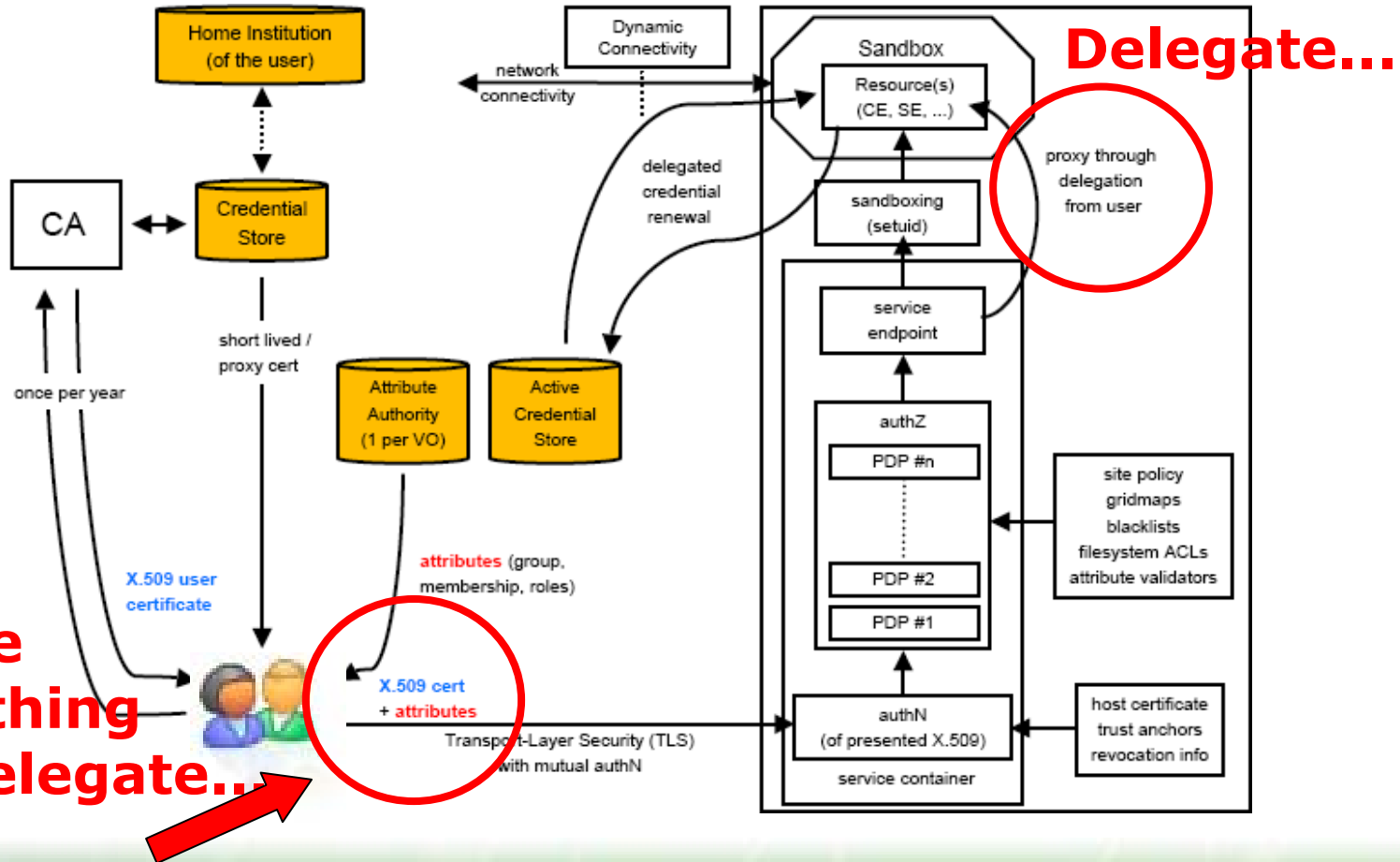
# gLite - Security flow (2)

**Short lived proxy certificate**





## gLite - Security flow (3)

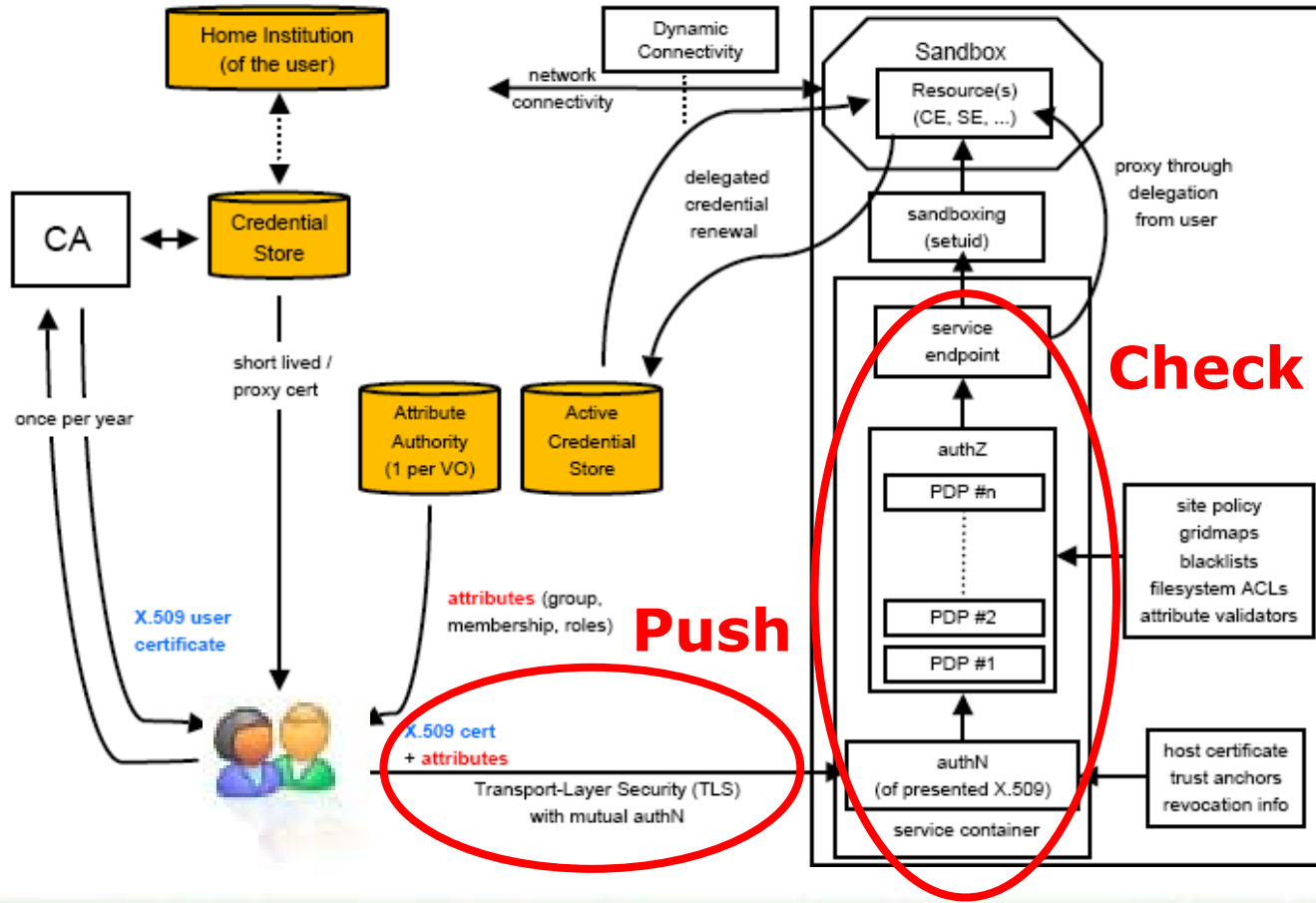


**Bundle everything and delegate...**



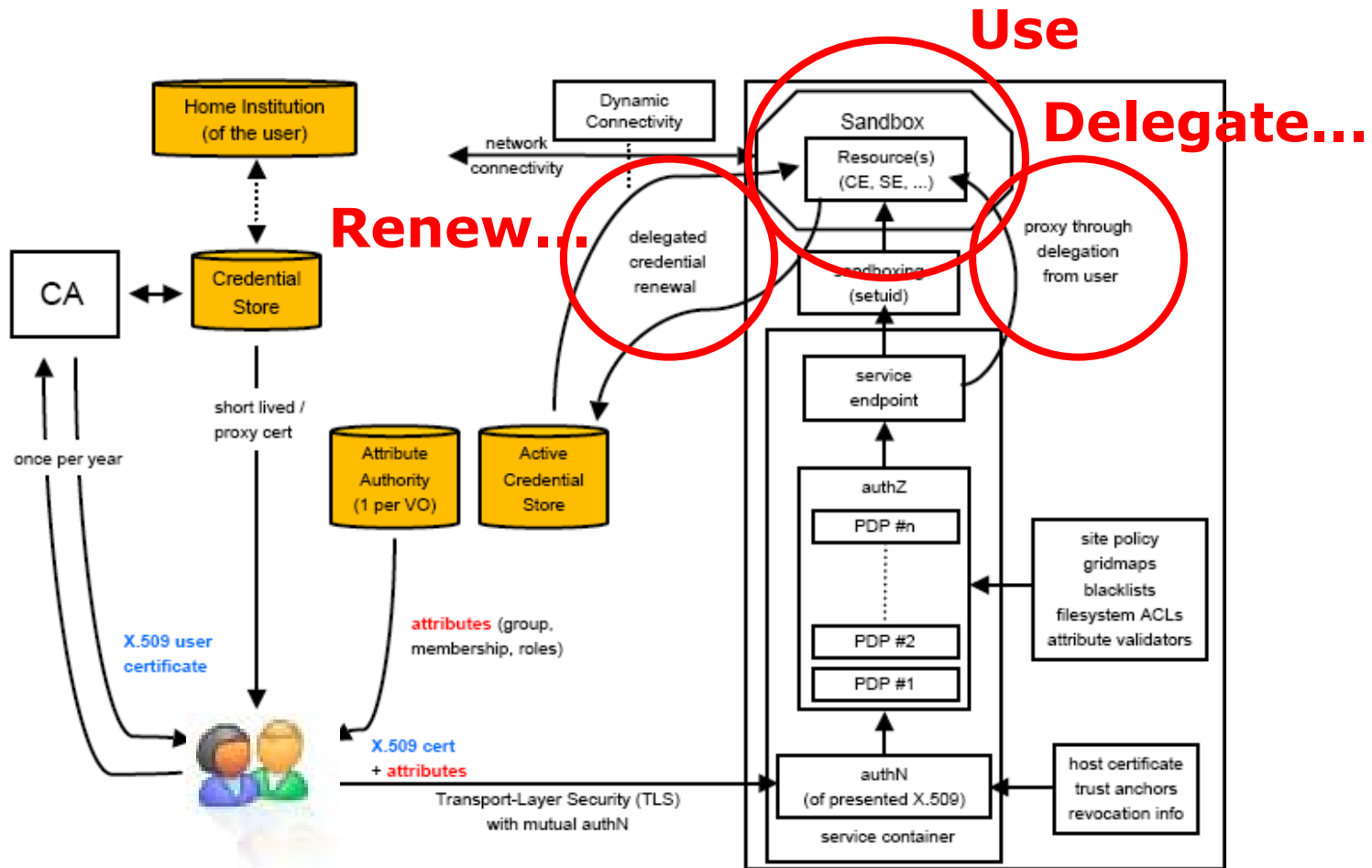


# gLite - Security flow (4)





## gLite - Security flow (5)





- **Grid middleware used by many European research projects**
  - DEISA (Distributed European Infrastructure for Scientific Applications) uses the UNICORE technology
  
- **UNICORE layers:**
  - Client: graphical interfaces, command line, APIs
    - *The UNICORE services can also be accessed through portals (e.g. GridSphere)*
  - Service: components of the Unicore Service Oriented Architecture
    - *Gateway – entry point to a Unicore site*
    - *NJS – job management & execution engine*
    - *Global service registry*
    - ...
  - System: interface between Unicore and the local resource management systems / operating systems





# UNICORE security – overview

- **Mutual authentication (between Gateway/NJS and User) using X509 Certificates**
- **No proxy certificates, no generalized delegation**
- **Authorization:**
  - Performed by NJS (and thus moved away from the target system)
  - Using UUDB (Unicore User Database)
  - More recent extensions to support both role and attribute based authorization (VOMS, Shibboleth)
- **Separation of consigner and endorser: only a user can endorse a job; an NJS or a user can consign a job**



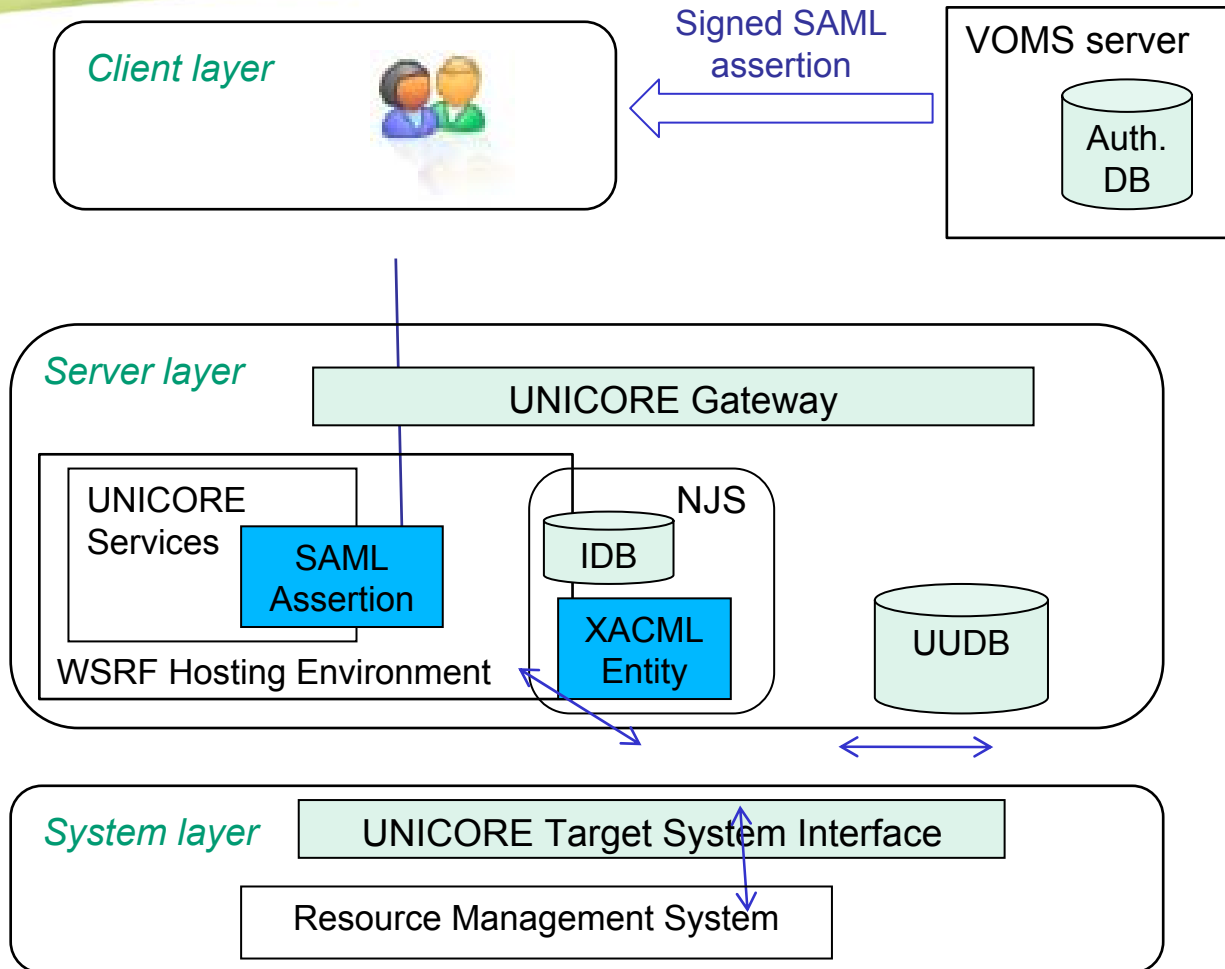


- **VOMS releases SAML assertions containing user attributes**
  - The assertions are included in the SOAP headers,
  - and signed with the VOMS server's certificate
  
- **The authorization decisions are taken in the service tier**
  - PDP – Policy Decision Point
  - uses XACML policies,
  - and information obtained from the UNICORE User Database





# UNICORE – Security flow (with SAML based VOMS)





- **Can services that are hosted in different environments (with different security mechanism) interoperate?**
  
- **This is a difficult problem**
  - We cannot expect all the organizations to adopt a single security technology
  - Or to share their user registries with other organizations
  
- **Ongoing work in many of the current Grid projects**
  - standardized protocol to communicate authorization assertions across OSG, EGEE, Globus and Condor
  - XtreemOS: interoperability solution based on SAGA (Simple API for Grid Applications)





- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- Grid Authorisation Systems
- Example of Security in Grid Systems
  
- **XtreemOS Security Infrastructure**
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**



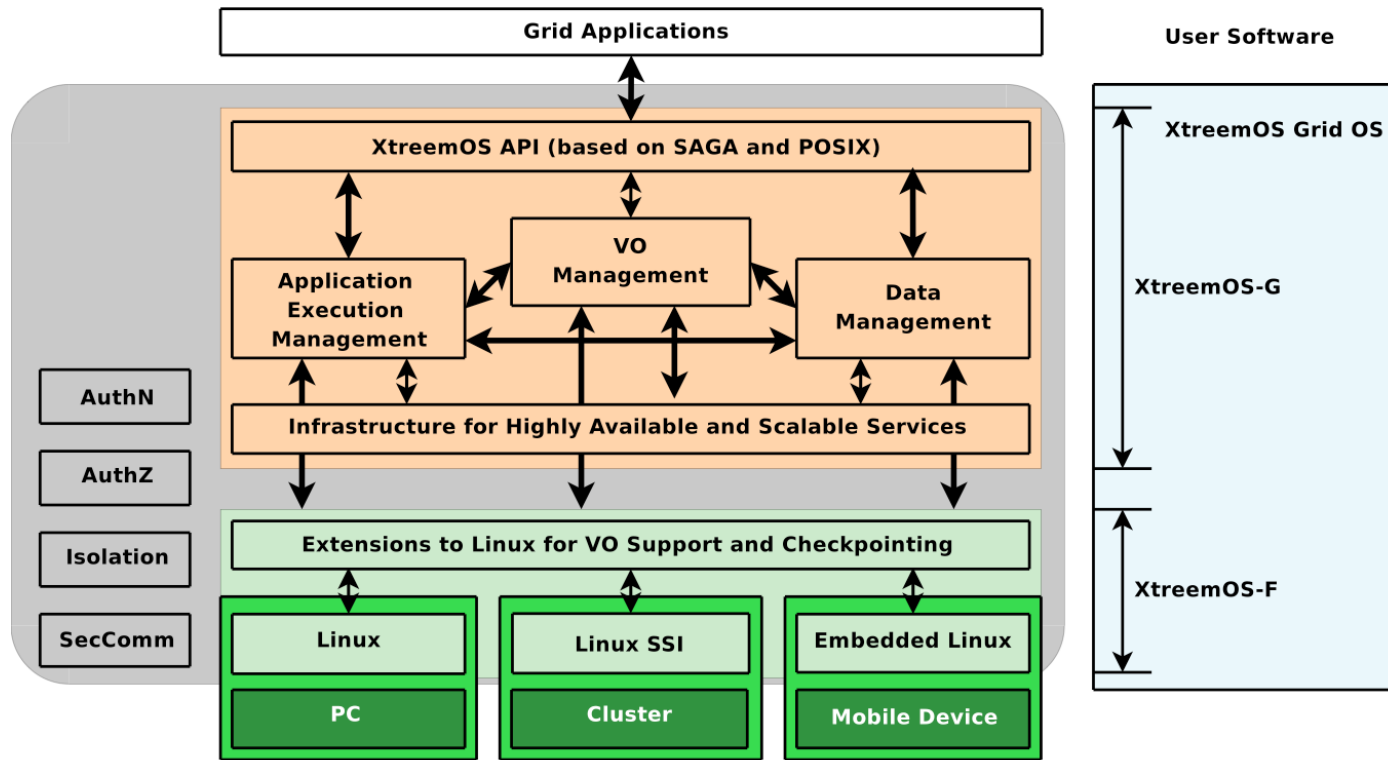


- **A XtreemOS system consists of**
  - A set of **resource machines** from one or more participants
    - Offering resources through a set of foundation-level node services
  - A set of **Grid-wide system services**
  - A set of **VOs** to support cross-machine resource sharing and logical isolation of resource usage within the system
  
- **A user of a XtreemOS system is defined as another system**
  - Including humans or separated autonomous software systems
  - Interacts with the current system through a set of well-defined interfaces.





## XtreemOS Software Architecture





## ▪ X-VOMS Database

- VO database management system, containing details of users' VO membership and VO attributes (e.g. VO groups they belong to)

## ▪ CDA: Credential Distribution Authority

- provides users with XOS-Certificates
- contain their public key and VO attributes,
- used to authenticate user and check their authorisation.

## ▪ RCA: Resource Certification Authority

- provides machines with a certified list of the resources (CPU, RAM etc) they can provide for use in AEM resource allocation

## ▪ VOPS: VO Policy Service

- allows flexible selection of nodes during AEM resource matching according to VO-defined policies

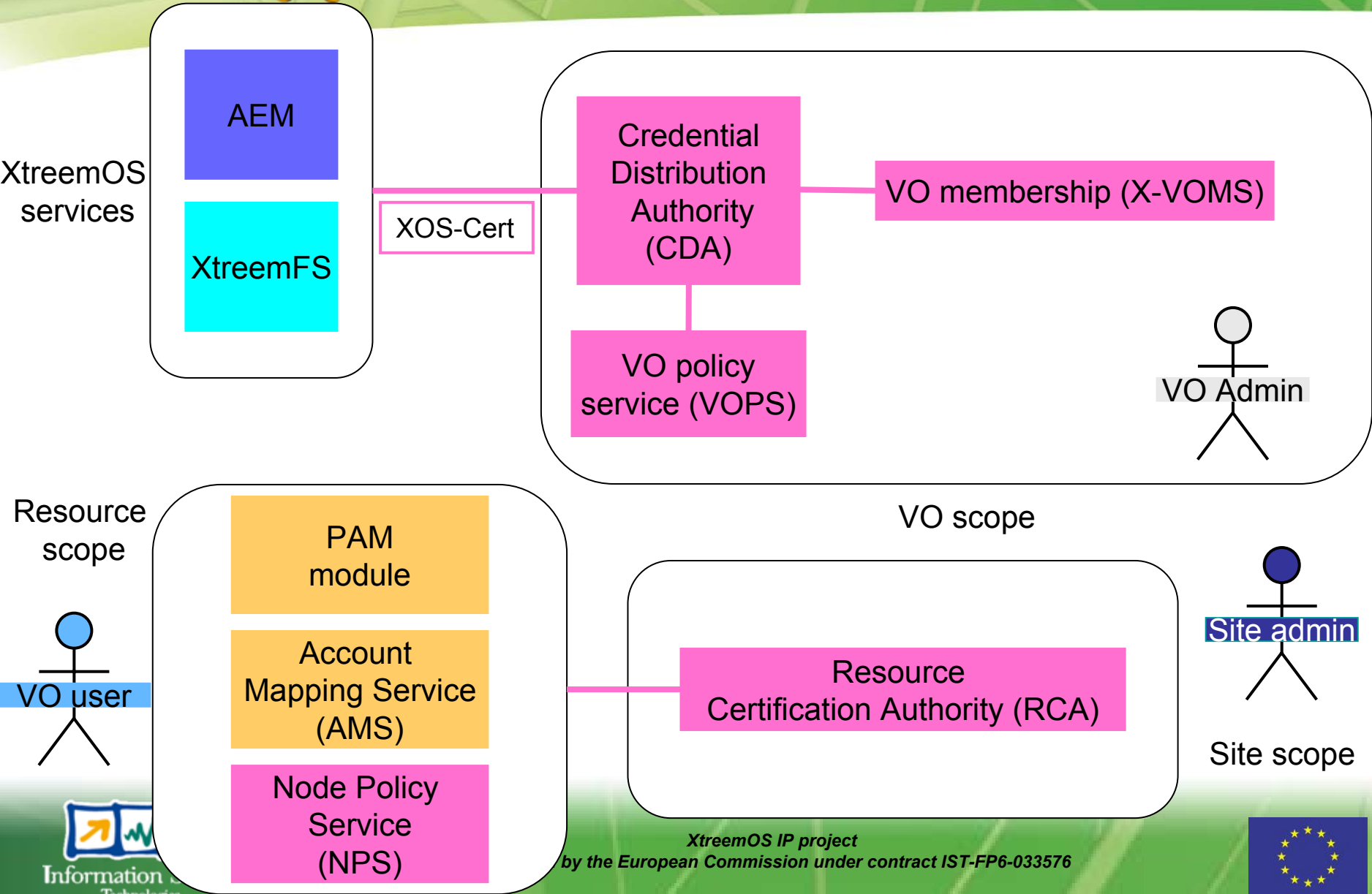
## ▪ VOFrontEnd and RCAFrontEnd

- Web applications providing a convenient front-end to the services above,
- allowing initial registration of users, creating of VOs and allocation of users to VOS





## VO-centric Security Architecture





## Virtual Organizations in Action

Welcome to VoLifeCycle

Home Manage My VOs Manage My Resources

Login  
Create an account

**User Login**

**Account Information**

Login id:

Password:

**Help Hints**



Enabling Linux  
for the Grid

This is the demo site for Virtual Organizations(VOs) managed by XtreamOS.

**No Account? [Create One](#)**  
The registration is open to XtreamOS consortium only.

```
paradeux : ~  
File Edit View Terminal  
/XtreamOS/Barcelona$ m  
kdir Volife  
yjegou@paradeux:~/Save  
/XtreamOS/Barcelona$ .  
/shot.sh
```





- The XtreemOS security infrastructure (XSI) includes the support for the following security functionalities:
  - **Secure communication**
    - i.e. mutually authenticated and encrypted channels
      - SRP, SSL, X.509
  - **Authorisation**
    - i.e. node-level and VO-level policy management
  - **Single Sign-On and Delegation**
  - **Isolation**
    - User-level through account mapping
    - VO-level through VO tokens





## ■ Global Entities

- Persist in a global namespace
- Identified by a global id (attribute of a global entity)
- User; nodes (machines); services; VOs

## ■ Local (OS-level) Entities -> Linux entities

- Exist in an OS (local) namespace
- OS Users, identified by user id
- OS Resources
  - Files, identified by an inode number
  - Processes, identified by a process id





- **Some terminology**
  - **Credential:** Piece of information about a subject
  - **Certificate:** Mechanism for carrying credentials
  - **Identity:** Human-readable information identifying an entity
  - **Identifier:** Machine-processable information identifying an entity
  
- **Type of Identifiers in XtreemOS**
  - Global user identifier (GUID)
  - Global virtual organisation identifier (GVID)
  - Global group identifier (GGID)
  - Global node identifier (GNID)
  
- **Only users and nodes are associated with X.509 certificates**
  - Their identity is represented by the Distinguished Name (DN)





- **Identity and Attribute Credentials**
  - Both are held as attribute certificates
  
- **XtreemOS Identity Certificates (XOS-Cert)**
  - No difference with standard X.509 identity certificate
  
- **XtreemOS Attribute Certificates**
  - Associated to an identity certificate through sharing a common DN field
  - Stores a ser of attributes of the same validity period of a global entity



## Type of Attributes

- **Users can have the following attributes**

- GUID
  - GVID
  - GGID
  - Group
  - Role
  - Capabilities
- Identifiers
- VO Attributes

- **Nodes can have the following attributes**

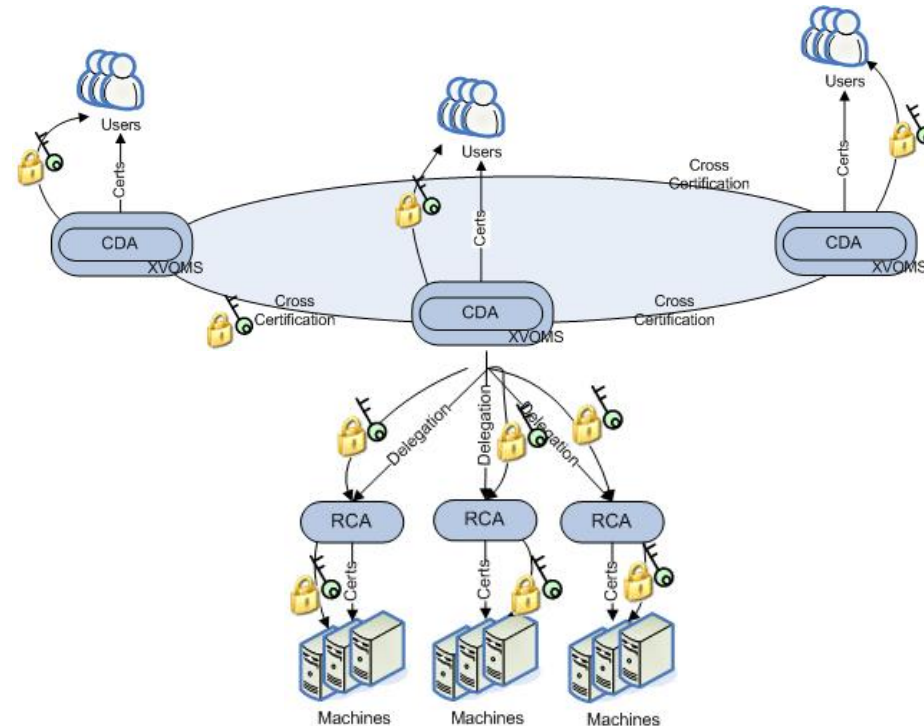
- GNID
- GVID
- Service





# XtreemOS Trust Model

- A cross-certified hierarchical PKI-based trust model
  - At the top level, there is a list of cross-certified root CA
    - Implemented by the CDA component of XVOMS
  - Underneath each root CA there is a list of subordinate CAs (RCAs)





# Building Up Trust in XtreemOS

- **XVOMS Certificate**
  - XVOMS has a self-signed certificate representing the root certificate of the system
  - The private counterpart is used by CDA to sign end-entity certificates for users and subordinated RCAs
  
- **User registration with XVOMS**
  - Each user shares a secret (i.e. password) with XVOMS
  - User obtain XVOMS public key certificate through established password-based mutual authentication protocols
  - **There is not need of pre-installed certificate**
  
- **RCA registration with XVOMS**
  - Each RCA is registered with a XVOMS and is given a shared secret with XVOMS
  - **Mutually authenticate with XVOMS with any pre-installed certificate**
  
- **Machine registration with RCA**





# Advantages of XtreemOS Trust Model

- **User management is separated from resources management**
- **Scalability in resource management**
- **Main difference with classical PKI trust models resides in the set up of trust**
  - In classical PKI models, trusted root CA certificates are distributed through **offline** means
  - In XtreemOS, certificates could be created on-the-fly and disseminated through **online** protocols
- **SSO and Delegation in the next release**
  - Not depending on proxy certificates



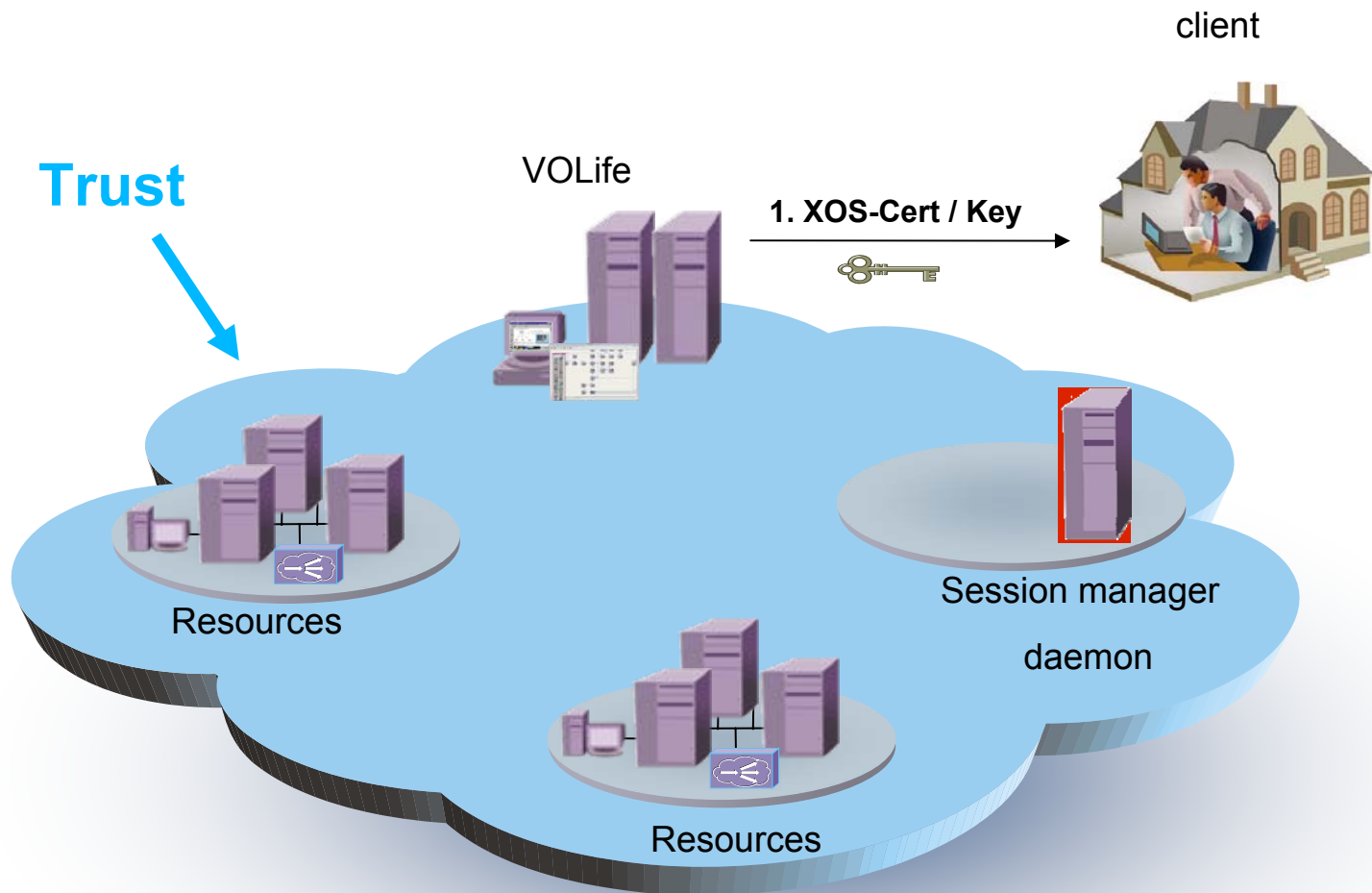


# Single Sign-On and Delegation

- **Single Sing-On**
  - As a distributed OS, XtreemOS services trust each other
  - Once user credentials are validated by a XtreemOS service, they can be used by other XtreemOS services without additional validation
- **Underpinning technology**
  - A trusted credential store service is associate to each user session.
    - Authenticate the user when he opens a session,
    - Collect and validate all user credentials,
    - Forward all grid requests (xsub, xps, etc.) from the user to XtreemOS services
  - **There is not need of proxy certificates**
- **Delegation, exploiting similar technology**
  - A credential store services is associated to jobs on the same resource node
  - Once job credentials are validated, they can be used in other XtreemOS services
  - Key technology for interactive jobs

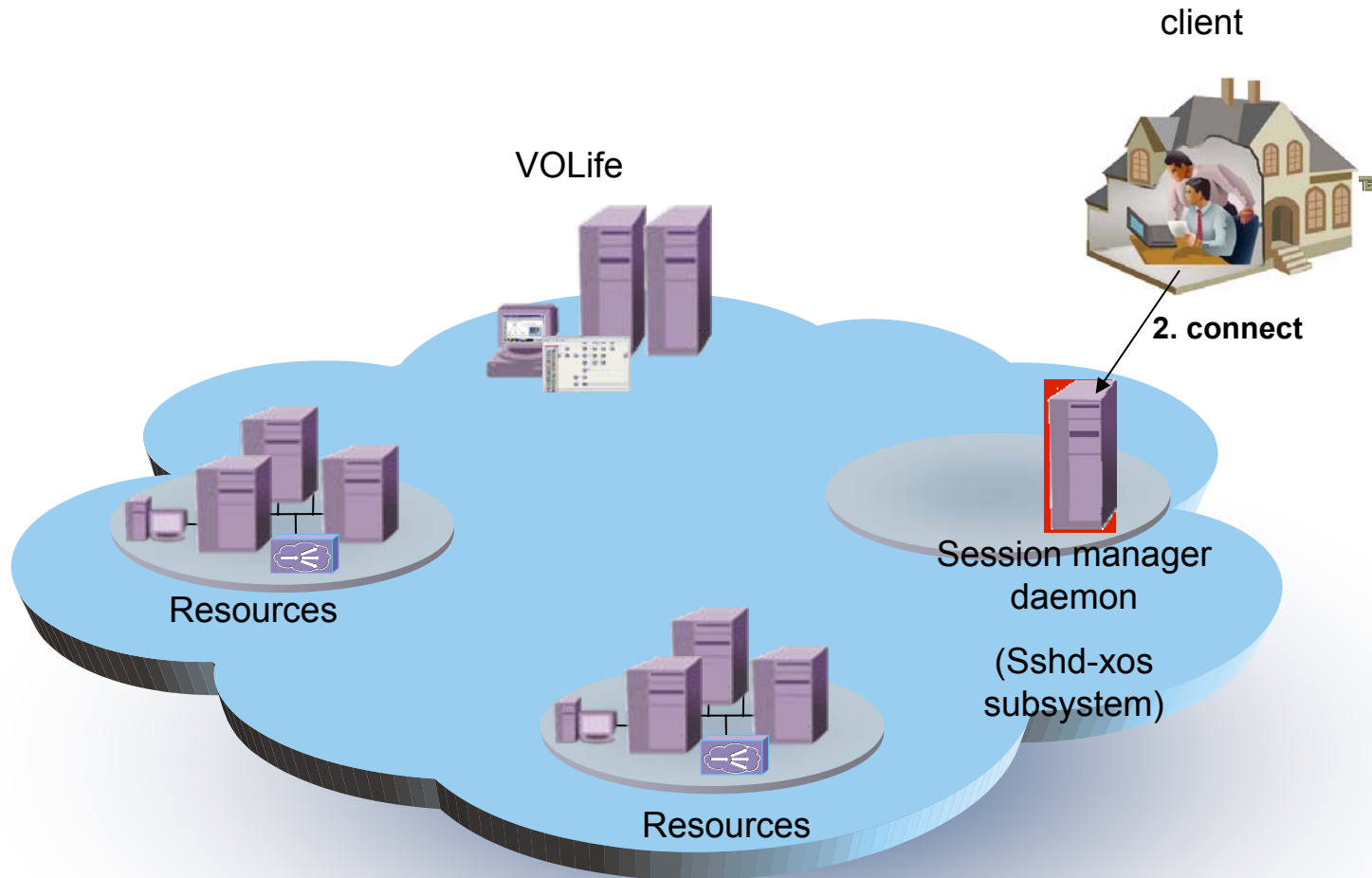


# Single-sign-on for grid requests



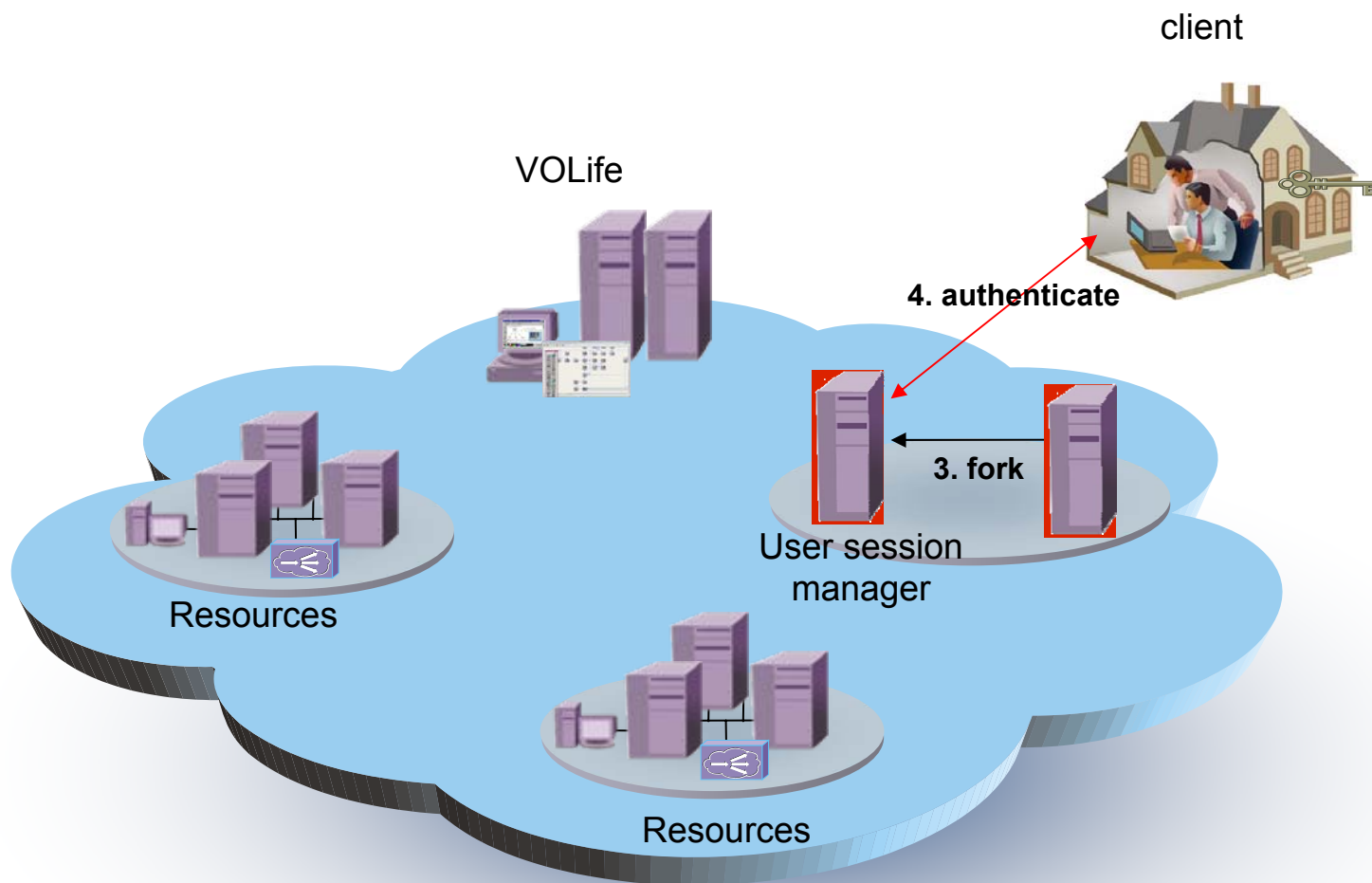


# Session managers



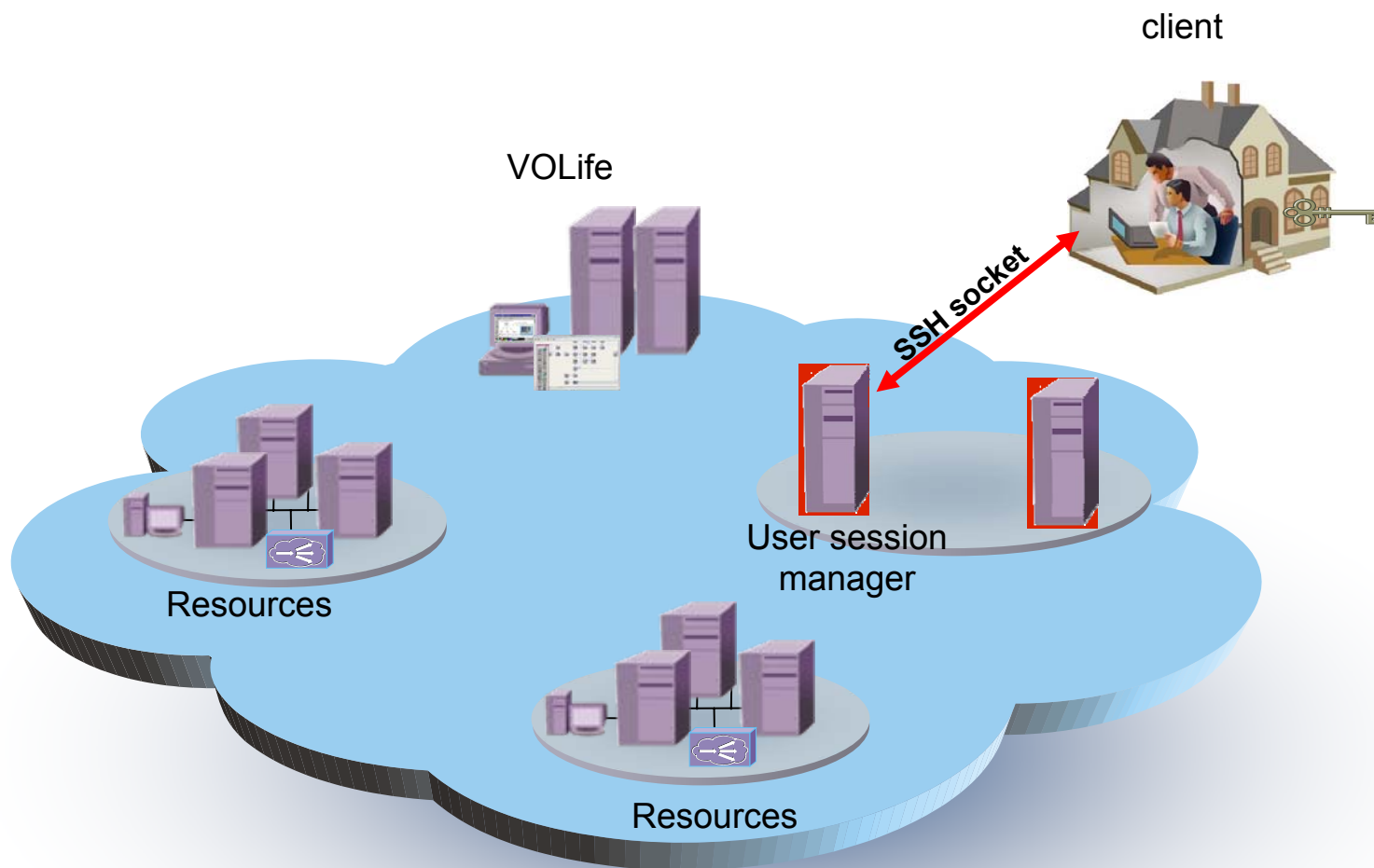


# Login



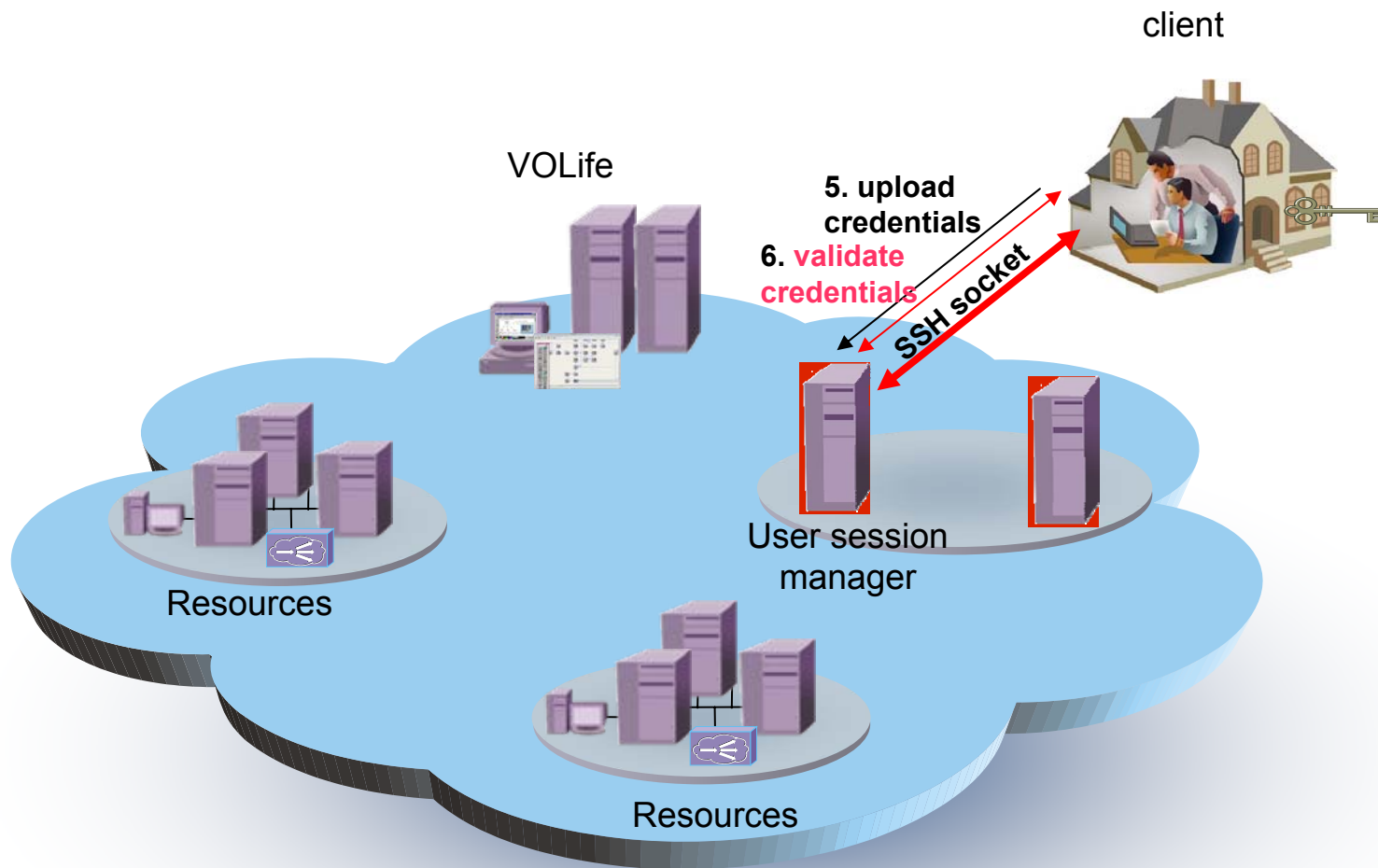


# SSH-XOS control master



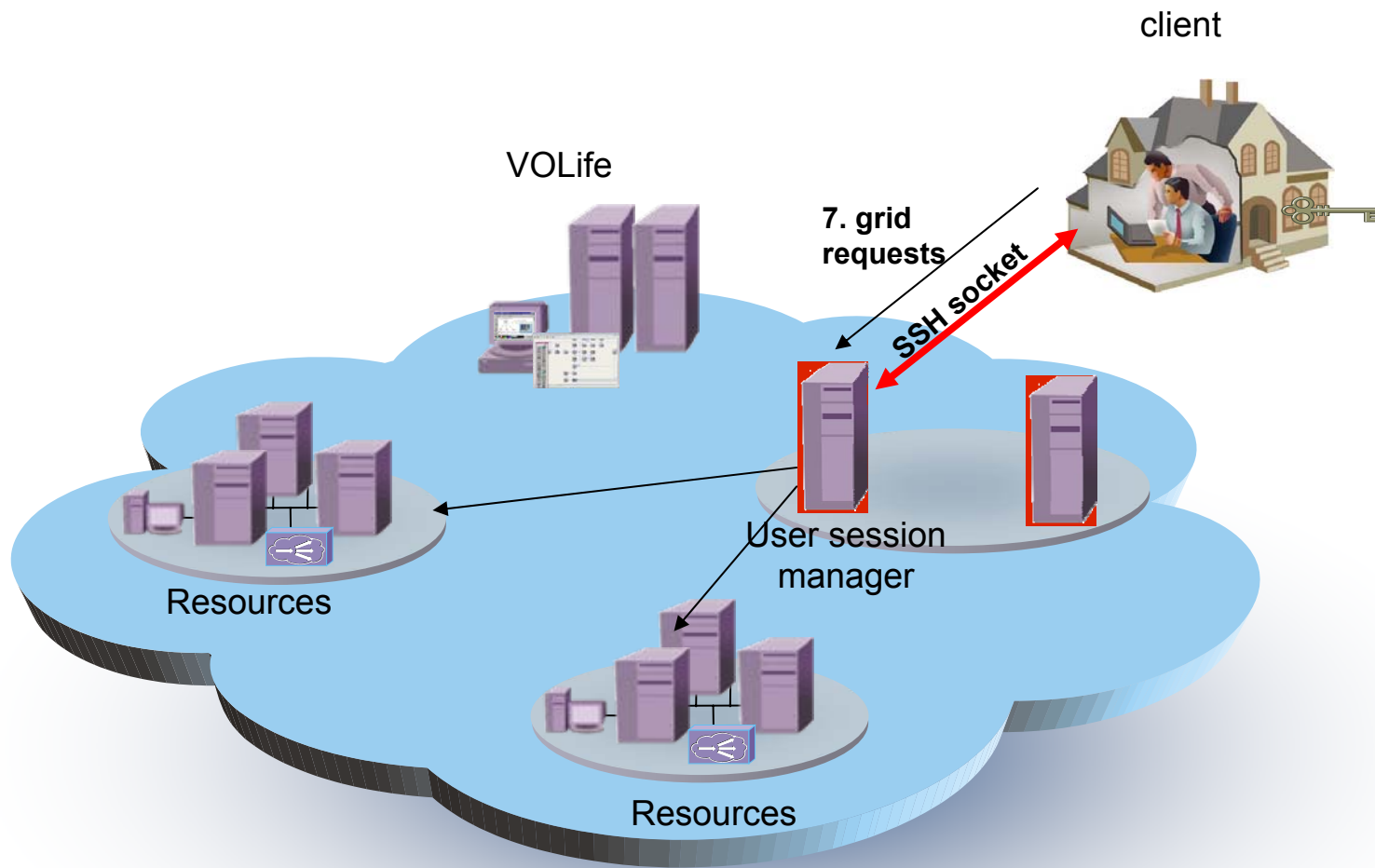


# Credential management





# Grid requests submission





- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- Grid Authorisation Systems
- Example of Security in Grid Systems
  
- XtreemOS Security Infrastructure
- **Isolation in XtreemOS**
  
- **Challenges and Open Issues**
- **Concluding Remarks**





# Isolation in XtreemOS

- **Basic idea: Put each job ( PAM session) into a resource container**
  - A resource container could be seen as either lightweight or heavy-weight virtual machines in a local OS instance
  - A resource container allows **fine-grained**, **isolated** and **strong** control of resource usage of a job (could be a hierarchy of processes )
- **Features: Full-fledged control of resource usage by VOs**
  - CPU: Assignment of cores, bandwidth/percentage/ priority/walltime allocation
  - Memory: virtual/physical/swap memory limitation
  - Disk I/O: disk i/o bandwidth limitation
  - Network: network bandwidth/traffic limitation





## Isolation

Job request with XOS credentials

PAM-aware applications  
( AEM ExecMgr, XOS-SSHD...)

XOS-NSS-PAM extension module

**Account Mapping**

VO Users → Unix accounts: uid/gid(s)

**Resource Container Management**

Put VO User's Job into containers/VMs

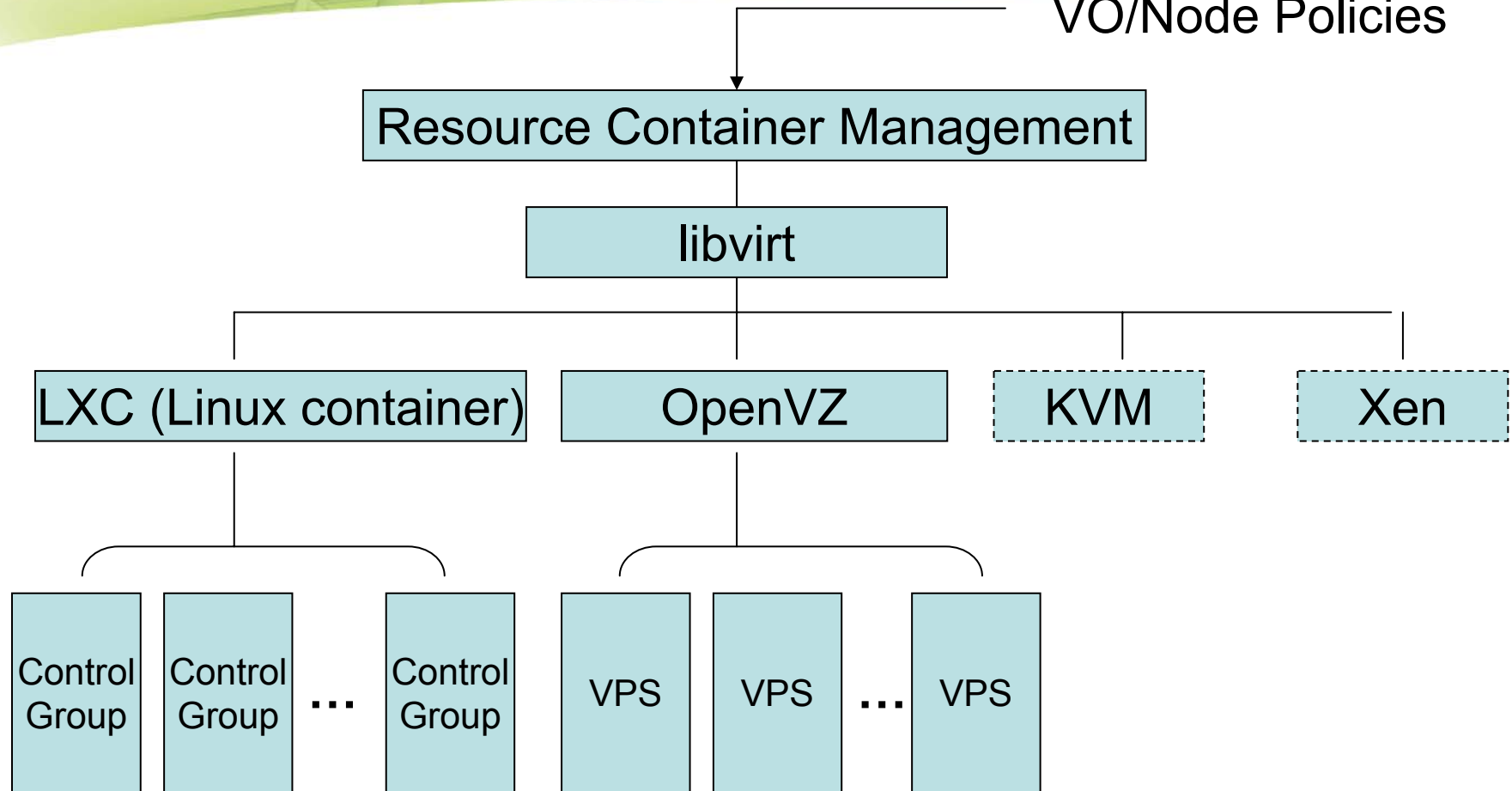
Advanced VO Support  
-Strong Isolation  
-Policy enforcement





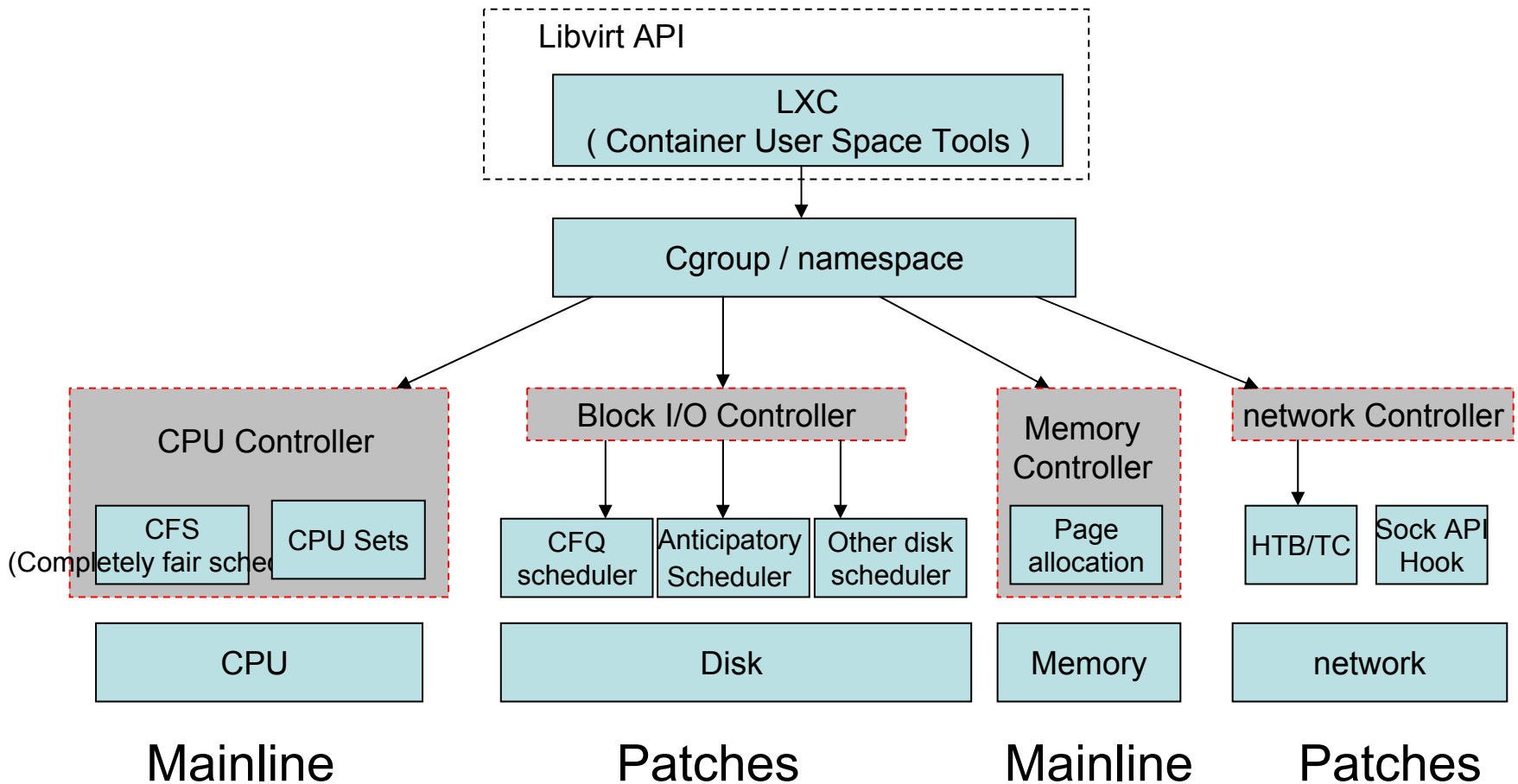
# How to do it ?

VO/Node Policies





# LXC (cgroup) approach





# What we achieve now?

- In advanced version of VO-support, what new features have been embedded in based on cgroup mechanism?

```

root@testbed0: ~/workspace/linux-2.6-lxc
root@uml:~# ls /mnt/cgrp/
cpu.rt_period_us          disk.bandwidth_debug      memory.force_empty
cpu.rt_runtime_us        disk.bandwidth_per_sec    memory.limit_in_bytes
cpu.shares                 disk.bandwidth_stat       memory.max_usage_in_bytes
cpuacct.usage            disk.limit_in_block       memory.stat
debug.cgroup_refcount     disk.limit_in_inode       memory.usage_in_bytes
debug.current_css_set     disk.max_usage_in_block   net.raw
debug.current_css_set_refcount disk.max_usage_in_inode   net.tcp
debug.releasable         disk.stat                  net.tot
debug.taskcount          disk.usage_in_block       net_udp
devices.allow            disk.usage_in_inode       notify_on_release
devices.deny             freezer.state              release_agent
devices.list             memory.failcnt             tasks
root@uml:~#
root@uml:~#
    
```

**CPU subsystem** (points to `cpu.*`)

**Disk subsystem** (points to `disk.*`)

**Memory subsystem** (points to `memory.*`)

**Network subsystem** (points to `net.*`)





## ■ Isolation of task group

- Userspace helps strong isolation for task group

\* Before enable cgroup mechanism, 'ps aux' show same behaviors as normal.

```
anqin@anqin-desktop:~/workspace/cntprj/cgroup/src/examples$ sudo ./pam_app_conv2 -pem testcerts/testbed1_usercert.pem
/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8@anqin-desktop:/tmp$ id
uid=60000 (/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8) gid=60138(xosuser_g60138) groups=60138(xosuser_g60138)
/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8@anqin-desktop:/tmp$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.4	0.7	2948	1816	?	Ss	04:15	0:06	/sbin/init
root	2	0.0	0.0	0	0	?	S<	04:15	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S<	04:15	0:00	[migration/0]
root	4	0.0	0.0	0	0	?	S<	04:15	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	04:15	0:00	[watchdog/0]
root	6	0.0	0.0	0	0	?	S<	04:15	0:00	[events/0]
root	7	0.0	0.0	0	0	?	S<	04:15	0:00	[khelper]

\* After enable cgroup mechanism, processes can only see self-space. So, 'ps aux' works differently.

```
anqin@anqin-desktop:~/workspace/cntprj/cgroup/src/examples$ sudo ./pam_app_conv2 -pem testcerts/testbed1_usercert.pem
/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8@anqin-desktop:/tmp$ id
uid=60000 (/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8) gid=60138(xosuser_g60138) groups=60138(xosuser_g60138)
/CN=62a57c32-9c7f-4da2-a08d-1093f5e6bee8@anqin-desktop:/tmp$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.2	5960	692	?	Ss	04:44	0:00	./pam_app_conv2 -pem testcerts/testbed1_usercert.pem
60000	2	0.6	1.6	9168	4164	pts/2	S	04:44	0:00	bash
60000	29	0.0	0.6	5680	1680	pts/2	R+	04:45	0:00	ps aux





## ▪ Disk quota limitation

- Record the usage of allocated file inode
- Record the usage of allocated disk block

- Limit created file number

```
# echo 4 > disk.max_usage_in_inode
```

```
root@testbed0:/tmp
[root@testbed0 test]#
[root@testbed0 test]# cd /tmp/
[root@testbed0 tmp]# touch test1
[root@testbed0 tmp]# touch test2
[root@testbed0 tmp]# touch test3
[root@testbed0 tmp]# touch test4
[root@testbed0 tmp]# touch test5
touch: cannot touch `test5': Disk quota exceeded
[root@testbed0 tmp]#
```

- Limit allocated file block (3\*4096)

```
# echo 1288 > disk.max_usage_in_block
```

```
root@testbed0:/tmp
[root@testbed0 tmp]#
[root@testbed0 tmp]#
[root@testbed0 tmp]# echo "test file 1" >> test1
[root@testbed0 tmp]# echo "test file 2" >> test2
[root@testbed0 tmp]# echo "test file 3" >> test3
[root@testbed0 tmp]# echo "test file 4" >> test4
bash: echo: write error: Disk quota exceeded
[root@testbed0 tmp]#
[root@testbed0 tmp]#
```





- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- Grid Authorisation Systems
- Example of Security in Grid Systems
  
- XtreemOS Security Infrastructure
- Isolation in XtreemOS
  
- **Challenges and Open Issues**
- **Concluding Remarks**





- **New technology = new security risks**
  
- **Adaptation**
  - Incorporate existing know-how in a uniform and single way
  
- **Virtual Machine Management**
  - Secure migration of components across multiple domains
  - IP spoofing

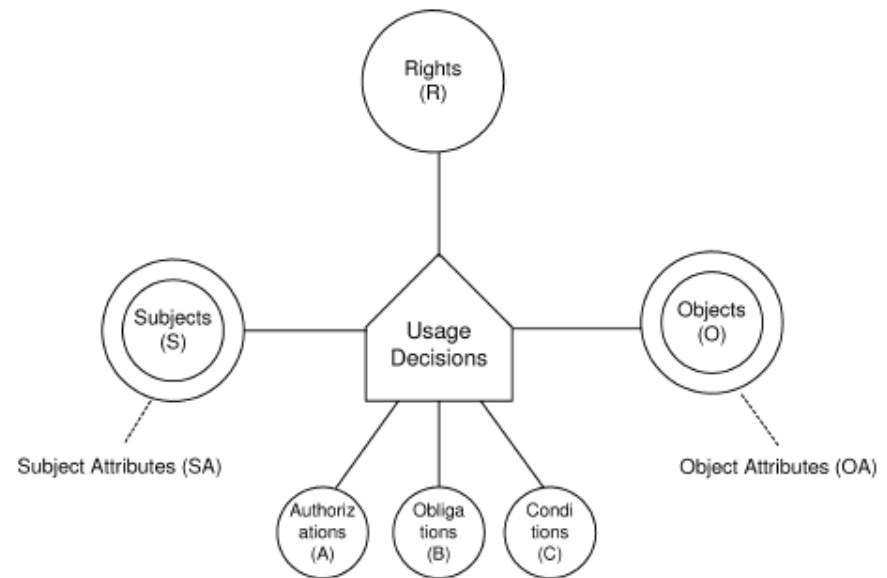


- **Fine-grained access control**
  - Usage control in Grids
  
- **Privileged access to data in Clouds**
  - Sensitive data processed outside enterprise boundaries needs the assurance that they are only accessible and propagated to privileged users
  - Usage control in Clouds
  
- **Policy composition**
  - Service composition should include policy composition



# Usage Control in Grids

- Encompass traditional access control, trust management and digital right management
- Usage control based on
  - Authorisations
  - Obligations
  - Conditions
- Mutability of attributes
- Continuity of decision





- **Recommendation systems and reputation in Clouds**
  - Reputation do not virtualise well – one customer's bad behaviour can affect the reputation of Cloud as a whole
  - E.g. Blacklisting of EC2 IP addresses by spam-prevention services may limit which applications can be hosted
- **Frameworks for trust calculation, composition and propagation in services**
  - Strategies to determine the level of trust for composite components based on the integration of individually secure, insecure or malicious components



- **We need to “measure” the security implications (risks) of putting my data/computation in the Cloud**
  - Risk management/utility computing in Clouds
- **Monitoring security**
- **Security performance**





## ▪ **Auditability**

- Audit access and usage of information to trace inappropriate activities

## ▪ **Privacy and Legal Issues**

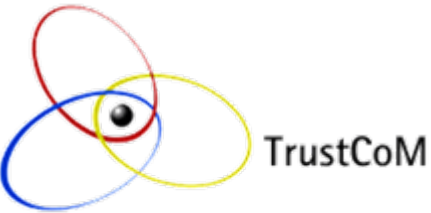
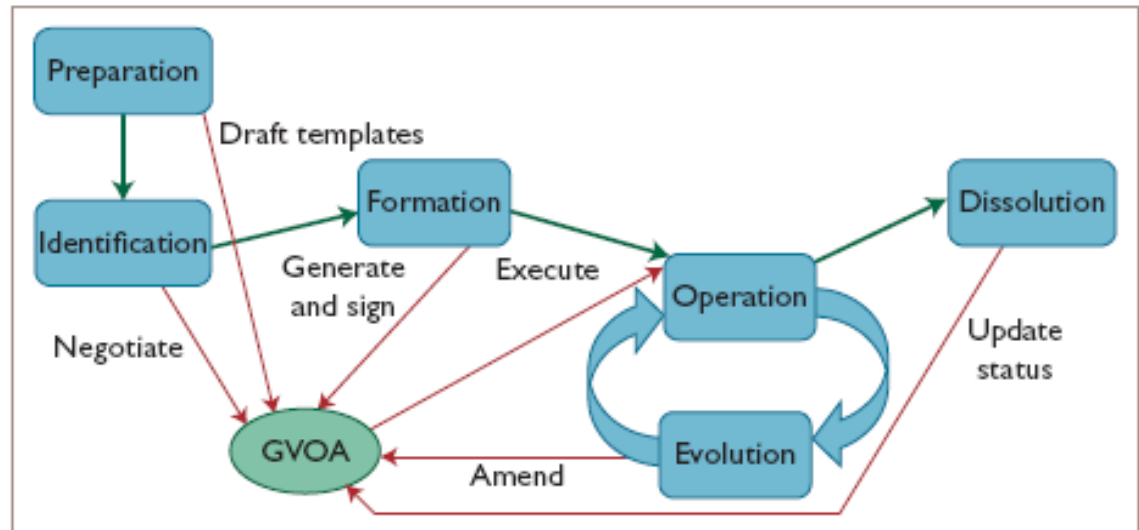
- Cloud providers must commit to storing and processing data in specific jurisdictions and to obey local privacy requirements on behalf of the customer
- Cloud providers may require security certifications that their infrastructure complies with some regulatory security requirements





# Contract Management in Collaborations

- **Business Contract:** human-readable representation of the contract that addresses the main risks related to participating in the collaboration
- **Operational Contract:** machine-readable representation of the contract that structures the document for automated processing





- Basics Security Concepts
- Public-Key Encryption
  
- Grid Security Infrastructure
- Grid Authorisation Systems
- Example of Security in Grid Systems
  
- XtreemOS Security Infrastructure
- Isolation in XtreemOS
  
- Challenges and Open Issues
- **Concluding Remarks**





- **Improving usability**
  - **Local resource administrator: autonomous management of local resources**
  - **VO administrator: flexibility management of credential and VO policies**
  - **End user: login as a Grid user into a VO; the Grid should be as much as possible invisible**
  - **Posix interface as far as possible**
  
- **Secure and reliable application execution**
  - **Fine-grained control of resource usage**





- **Scalable VO management**
  - Independent user and resource management
  - Interoperability with VO management frameworks and security models
  - Customizable isolation, access control and auditing
  
- **Very Dynamic VOs**
  - Short-lived VOs created automatically for the duration of an application/workflow
    - Multi-users
  - Lightweight configuration of resources
  - Predefined policies (VO-based)





# On-going and Future Work

- **Delegation**
  - Following similar approach to SSO
  
- **Traceability**
  - Exploiting tokens for traceability in SSO
  
- **Security monitoring and auditing**
  - Rule-based monitoring systems; including aggregation of events and logs for auditing purpose
  
- **Interoperability**
  - Shibboleth; Kerberos; myProxy
  - Exploiting SAGA for accessing at application level XtreemOS and gLite Grids.
  
- **CDA distribution through Distributed Servers**





- Some slides are based on presentations given by:
  - **Ake Edlund's security course at ISSGC'07**
  - **Peter Gutmann's tutorial on Security**
  - **Syed Naqvi's Grid security tutorial at CGW 2006**
  - **Philippe Massonet, CETIC, presentation on Grid security requirements, OGF 25, March 2009**
  - **Matej Artac's presentation on XtreemOS VOPS**
  - **XtreemOS Security Tutorial at ICS'09 (Christine Morin, Yvon Jegou and Corina Stratan)**



- **This work is a summary of the work carried out in XtreamOS WP2.1 and WP3.5 work packages**
  - **INRIA:** Christine Morin, Yvon Jegou
  - **ICT:** Haiyan Yu
  - **SAP:** Philip Robinson
  - **STFC:** Benjamin Aziz, Ian Johnson, Brian Matthews, Erica Yang, Adam Barker
  - **XLAB:** Matej Artac



## References

- A. Qin, H. Yu, C. Shu, X. Yu, "Operating System-level Virtual Organization Support in XtreemOS", 9th International Conference on Parallel and Distributed Computing, Applications and Technologies (**PDCAT'08**)
- A. Qin, H. Yu, C. Shu, B. Xu, "XOS-SSH: A Lightweight User-Centric Tool to Support Remote Execution in Virtual Organizations". 1st USENIX workshop on Large Scale Computing Systems, 2008 (**LASCO'08**)
- E.Y. Yang, I. Johnson, B. Matthews, A.E. Arenas. "VOHost: A Secure and Flexible VO Hosting System for Grids and Beyond". UK e-Science 2008 All Hands Meeting (poster) (**UK AHM'08**)
- C. Shu, E.Y. Yang, A.E. Arenas. "Detecting Conflicts in ABAC Policies with Rule-Reduction and Binary-Search Techniques". Proc. IEEE International Symposium on Policies for Distributed Systems and Networks (**Policy 2009**)
- E.Y. Yang, B. Matthews. "DToken: a Lightweight and Traceable Delegation Architecture for Distributed Systems". 28th IEEE Symposium on Reliable Distributed Systems (**SRDS 2009**)



XtreemOS



Enabling Linux  
for the Grid

**Thank you !**

Questions ?



Information Society  
Technologies

*XtreemOS IP project  
is funded by the European Commission under contract IST-FP6-033576*

