MONASH University
Information Technology

FIT3142 Distributed Computing

Topic 6: Grid Middleware and Security Infrastructure

Dr Carlo Kopp
Dr Asad I. Khan
Clayton School of Information Technology
Monash University

© 2008 - 2016 Monash University

Why Study Grid Middleware?

- Middleware is the software layer which sits in between Distributed Applications and the Operating Systems and Network Protocol Stacks – hence the term “Middleware”;
- Foundation knowledge: The intent behind middleware is to provide Grid, Cloud, or Web Services functionality while hiding most of the details from the programmer – you need this understanding to design applications;
- Practical skills: By necessity middleware involves the use of standard interfaces, so this topic deals with numerous standards, that you will need to know to code applications;
- Warning*: Standards are like “death and taxes” – unavoidable, but usually unpleasant and tedious!*

Why Study Grid Security?

- Security is critical in a globally networked computing environment;
- Weaknesses in security can and will be exploited by third parties with malicious agendas;
- Examples can include criminal organisations and individuals, corporations attempting theft of trade secrets and commercial data, pranksters, and foreign governments seeking a competitive advantage;
- Foundation knowledge: The Grid security infrastructure is the most mature distributed security model and presents the best case study;
- Practical skills: Future distributed environments will employ similar models and basic technology as employed in existing Grid security infrastructure.

GRID MIDDLEWARE AND WEB SERVICES

References / Reading

- Grid Computing by Pawel Plaszczak and Richard Wellner, Jr.
- Globus Tool Kit Documentation, http://www.globus.org/toolkit/docs/4.0/
- A Globus Primer, http://www.globus.org/toolkit/docs/4.0/key/
- Grid Computing: A Practical Guide to Technology and Applications by Ahmar Abbas;
- SOAP Version 1.2 Part 0: Primer (Second Edition)/W3C Recommendation 27 April 2007

Overview of Topic - Grid Middleware?

- Utility Computing concepts;
- Explore Web Services (WS) and the Simple Object Access Protocol (SOAP) – WS is widely used outside grids, so it is useful even if tedious material to study;
- Explore Open Grid Services Architecture (OGSA) and associated concepts – this is how the various layers fit together and why;
- Explore the Globus Toolkit which will be used in the Labs extensively.

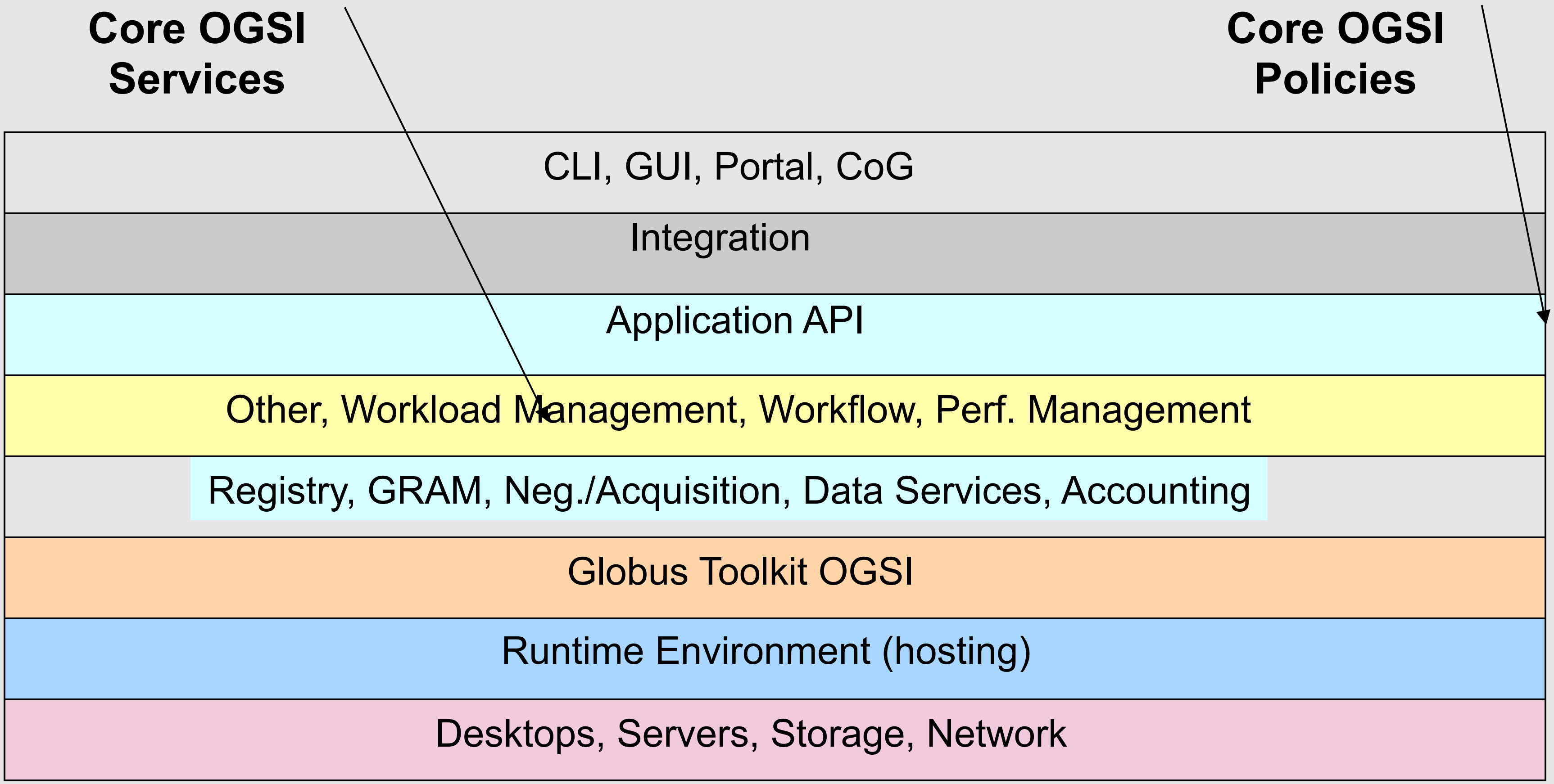
Grid Terminology Abbreviations

- OGSA - Open Grid Services Architecture
- WSRF – Web Services Resource Framework
- WSDL - Web Services Definition Language
- SOAP – Simple Object Access Protocol
- GT – Globus Toolkit
- XML – eXtensible Markup Language
- VO – Virtual Organisation
- GSI - Grid Security Infrastructure
- WS GRAM - Web Services Grid Resource Allocation and Management

Utility Computing

- **Utility computing or on-demand computing would require the dynamic entity management features of WSRF**
- **Current system infrastructure is static, heterogeneous, and relatively coarse grained**
- **It is also often tailored for the in-house requirements of the owning department**
- **Move to utility computing would mean use of infrastructure on per-use-pay basis.**
  - Compute grids are therefore similar to power grids and phone networks

OGSA Platform (Discussed Further)



Core OGSI Services

Core OGSI Policies

CLI, GUI, Portal, CoG

Integration

Application API

Other, Workload Management, Workflow, Perf. Management

Registry, GRAM, Neg./Acquisition, Data Services, Accounting

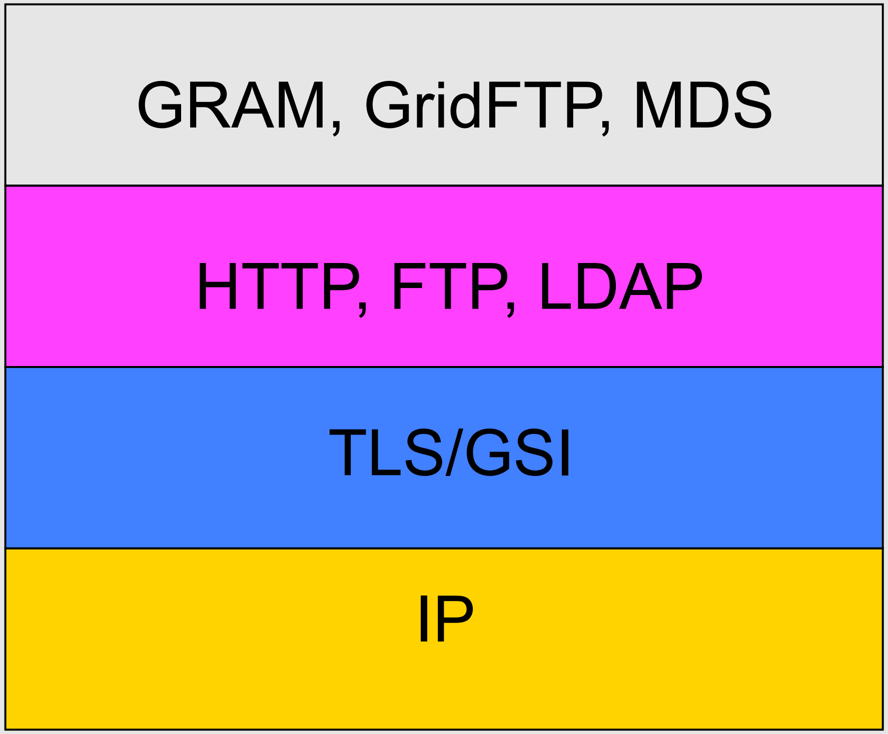
Globus Toolkit OGSI

Runtime Environment (hosting)

Desktops, Servers, Storage, Network

OGSA Platform: A Functional Representation

Simplified GT Protocol Stack (Discussed Further)



GRAM, GridFTP, MDS

HTTP, FTP, LDAP

TLS/GSI

IP

GT 2.x Protocol Stack

Grid Developments

- **GT 2.x supported GRAM, GridFTP, and MDS.**
  - These services were not integrated
  - Used their own protocols
- **SOAP and Web Services provide the uniformity of access and communication protocol**
- **Web Services are not sufficient for supporting Utility computing requirements**
- **Hence a combined approach has been adopted**

Grid and WS

- **XML and SOAP provide the generic and flexible communication fabric which is easy to implement**
- **The SOA can provide the Grid service layer functionality**
- **Web Services Definition Language (WSDL) provides standardisation of interfaces**
  - Enabling dynamic discovery of services

Global Grid Forum

- **First meeting of GGF was held in 2001**
- **To closely align WS with Grid services**
- **Effectively interconnect scattered resources**
  - By defining protocol level standards
- **GGF working groups: Architecture, Data, Security, Scheduling and Resource Management, Information System and Performance, Applications and Programming Models Environments, and Peer-to-Peer.**

WS-Resource Framework

- **Globus and IBM proposed WSRF as a new set of standards**
- **WSRF proposes a standard way of associating resources within Web services**
- **Extra abstraction added to Web services for supporting**
  - Virtualisation
  - On-demand computing

WSRF and SOA

- **WSRF endorses**
  - SOA where,
  - WSDL is for web service definition
  - SOAP is for communications
- **Web services act as gateways to IT resources within the Grid network**
- **A resource associated with a Web service is called a WS-Resource**
- **The associations can be one-to-many or many-to-many**
- **One Web service may be associated with several WS-Resource entities OR**
- **One resource may be accessible through many services.**

WS-Resource Characteristics

- **WS-Resources are dynamic**
  - Created and destroyed on demand
- **These are stateful entities**
  - State and properties can be defined
  - Known as WS-ResourceProperties
- **Actors of the Grid can**
  - Discover,
  - Inspect, and
  - Modify,

these properties by contacting the respective Web services

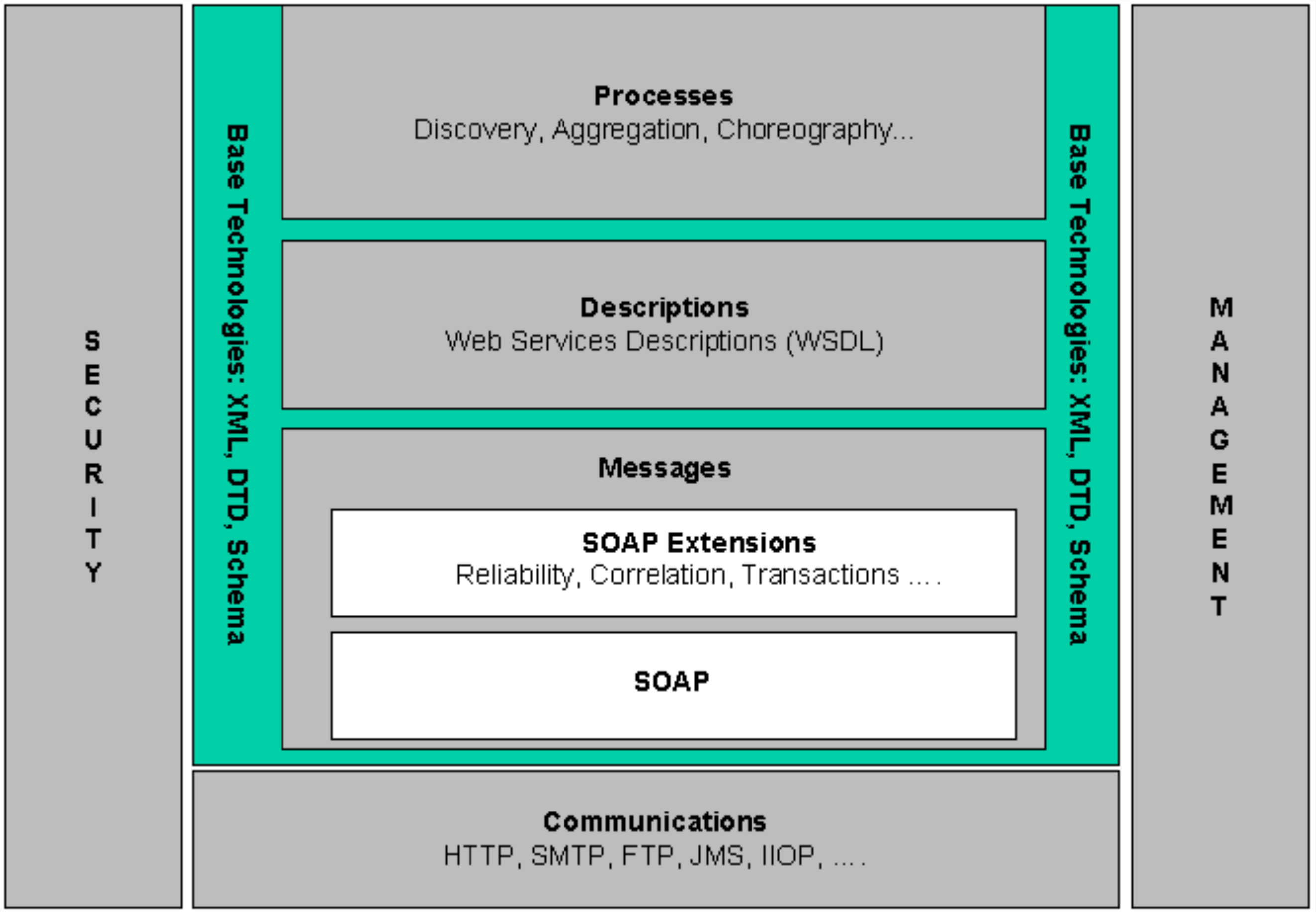
WS-Resource Addressing

- **No explicit address, WS-Resources can only be reached through their associated services**
- **These are identified by**
  - End point reference i.e. address of the WS
  - Followed by the relative identifier of the particular resource

Web Services (W3C 2004)

- "Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.";
- The intent of "Web Services" is to provide a standard technique for running distributed applications across the W3, regardless of the platforms being used for servers and clients;
- The W3C *Web Services Architecture* defines a framework for application and protocol designers intending to provide web services;
- The Web Services model relies heavily on the use of XML (eXtensible Markup Language), a derivative of SGML and the intended long term replacement for HTML;

Web Services (W3C 2004)



SECURITY

Base Technologies: XML, DTD, Schema

Processes
Discovery, Aggregation, Choreography...

Descriptions
Web Services Descriptions (WSDL)

Messages

SOAP Extensions
Reliability, Correlation, Transactions .... .

SOAP

Base Technologies: XML, DTD, Schema

MANAGEMENT

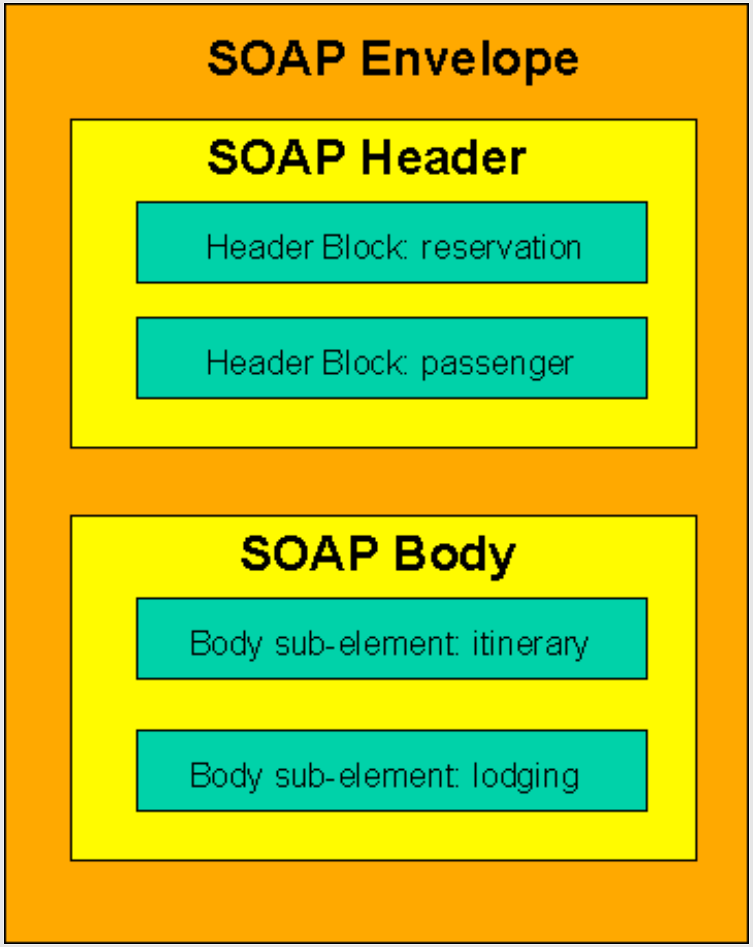
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Web Services - Simple Object Access Protocol

- **Communications between distributed applications in WS employ primarily HTTP, or other "basic" well established protocols such as FTP, SMTP etc;**
- **Messaging is based on SOAP (Simple Object Access Protocol), which is most commonly transmitted over HTTP or HTTPS in secure environments;**
- **SOAP provides similar functionality to CORBA, but is inherently more verbose due to the use of XML;**
- **W3C*: "A SOAP message is fundamentally a one-way transmission between SOAP nodes, from a SOAP sender to a SOAP receiver, but SOAP messages are expected to be combined by applications to implement more complex interaction patterns ranging from request/response to multiple, back-and-forth 'conversational' exchanges."***

W3C Simple Object Access Protocol

- Example SOAP message (W3C) for airline reservation application in WS environment;
- SOAP message has "Envelope" containing a SOAP "Header" with identifying information and "Body" with actual message payload;
- The SOAP message is formatted in XML language.


SOAP Envelope
SOAP Header
Header Block: reservation
Header Block: passenger
SOAP Body
Body sub-element: itinerary
Body sub-element: lodging

Example SOAP Message

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
<m:reservation xmlns:m="http://travelcompany.example.org/reservation" env:role="http://www.w3.org/2003/05/soap-
        envelope/role/next" env:mustUnderstand="true">
<m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
<m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime> </m:reservation>
<n:passenger xmlns:n="http://mycompany.example.com/employees" env:role="http://www.w3.org/2003/05/soap-
        envelope/role/next" env:mustUnderstand="true">
<n:name>Åke Jógvan Øyvind</n:name> </n:passenger> </env:Header>
 <env:Body> <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
<p:departure>
 <p:departing>New York</p:departing>
<p:arriving>Los Angeles</p:arriving>
<p:departureDate>2001-12-14</p:departureDate>
<p:departureTime>late afternoon</p:departureTime>
 <p:seatPreference>aisle</p:seatPreference> </p:departure>
 <p:return> <p:departing>Los Angeles</p:departing> <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
<p:departureTime>mid-morning</p:departureTime> <p:seatPreference/> </p:return> </p:itinerary> <q:lodging
        xmlns:q="http://travelcompany.example.org/reservation/hotels"> <q:preference>none</q:preference> </
        q:lodging> </env:Body> </env:Envelope>
```

W3C: Simple Object Access Protocol RPC

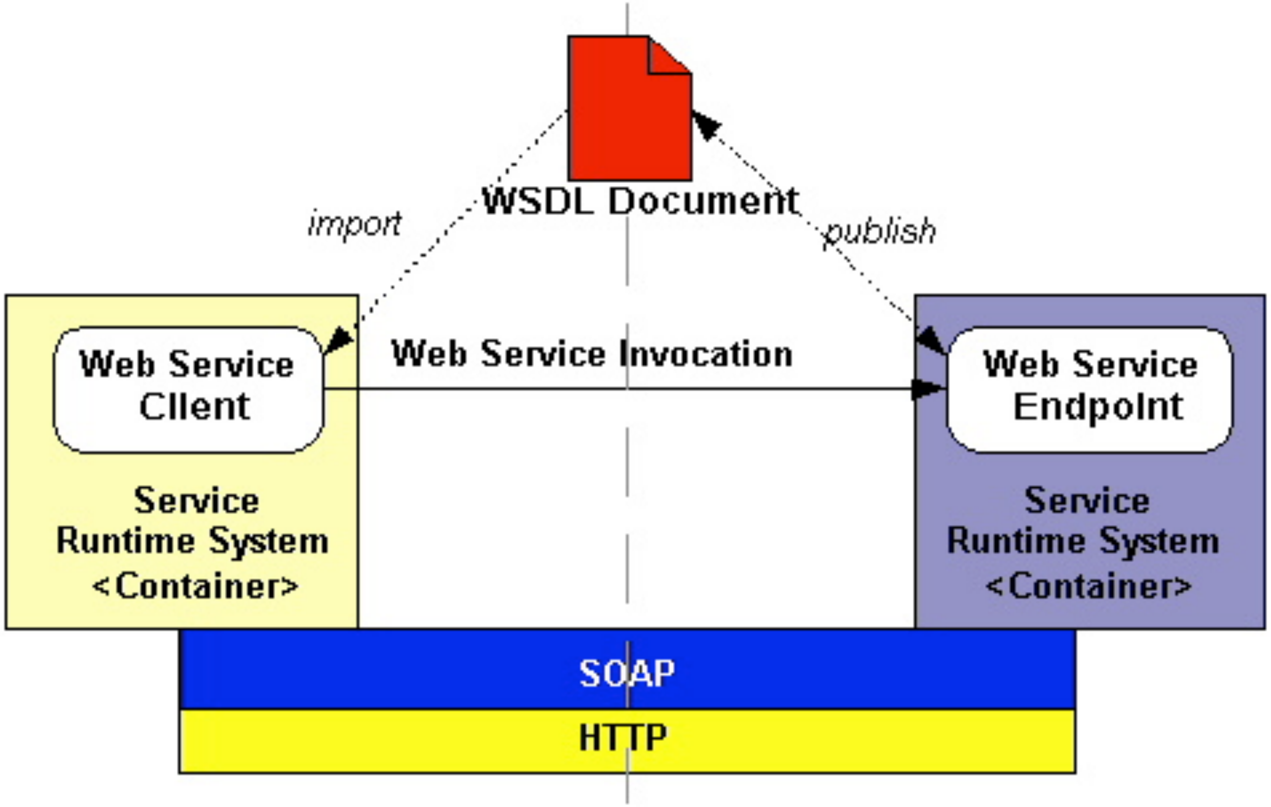
- To invoke a SOAP RPC, the following information is needed:

1. The address of the target SOAP node.
2. The procedure or method name.
3. The identities and values of any arguments to be passed to the procedure or method together with any output parameters and return value.
4. A clear separation of the arguments used to identify the Web resource which is the actual target for the RPC, as contrasted with those that convey data or control information used for processing the call by the target resource.
5. The message exchange pattern which will be employed to convey the RPC, together with an identification of the so-called "Web Method" (on which more later) to be used.
6. Optionally, data which may be carried as a part of SOAP header blocks.

Web Services in Java

- Java has a defined interface for Web Services, released initially in the 2002 JAX-RPC document;
- The model is based on the RPC concept, where a client can communicate with a remote process on a server host, and invoke procedure calls as required;
- The JAX-RPC scheme is based on the SOAP interface, and is intended to be standards complaint;
- The intent of the JAX-RPC scheme is to provide Java developers with an interface which is easily used and well validated.

Web Service Endpoints (JAX-RPC)



Grid Service Nomenclature

- WSRF restricts Grid service to services conforming to WSRF, However
- The term Grid service can also be used for any service which conforms to the specific interface conventions accepted throughout a grid.

Virtual Service Layer

- Grids operate at a virtual service layer
- Users of Grid services know only of the resources in terms of CPU-hours and storage bytes
- The virtual service layer enables sharing of services and resources across organisational boundaries transparently to the users

Virtual Organisation

- **The technological challenge of presenting scattered heterogenous IT resources seamlessly through uniform gateway services is solved by WSRF**
- **Management functions are still required for who gets access and to how much of the resource.**
  - Membership to various categories
  - Access privileges

PUBLIC KEY INFRASTRUCTURE AND GRID SECURITY

References/Reading

- Handbook of Applied Cryptography, by A. Menezes, P. van Oorschot, and S. Vanstone, 1996
- http://www-unix.globus.org/toolkit/docs/3.2/security.html
- http://myproxy.ncsa.uiuc.edu
- http://shibboleth.internet2.edu/
- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Crypto-Part-1-0798.htm
- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Crypto-Part-2-0898.htm

Grid Security

- The Grid Security Infrastructure (GSI)
- Public Key Infrastructure
- Digital Signatures
- Mutual Authentication
- Single Sign-On
- MyProxy
- GSI Authorisation
- Shibboleth

Overview

- **GSI is a set of libraries and tools developed by Globus; It is an overlay on the transport-layer security protocol - SSL**
- **Uses the public key infrastructure (PKI) to achieve**
  - Authentication
  - Data Integrity Verification
  - Single Sign-on
  - Inter-organisation decentralised security system
- **All Grid entities (users and processes) must have a public key certificate**

PKI

- Under a Public Key Infrastructure, each grid user has a key pair
  - Key Pair = (public key + private key)
- Data Encrypted with one of the two keys can ‘only’ be decrypted by the other key, and vice versa
- Public Key Certificate of the subject (Bob) = {Bob’s public key + Bob’s personal details}
- Bob must not reveal his private key to anyone, and hence the name ‘private’
- Bob’s private key is pass-phrase protected

PKI Certificates

- **A GSI certificate follows the X.509 standard**
- **It includes four primary pieces of information:**

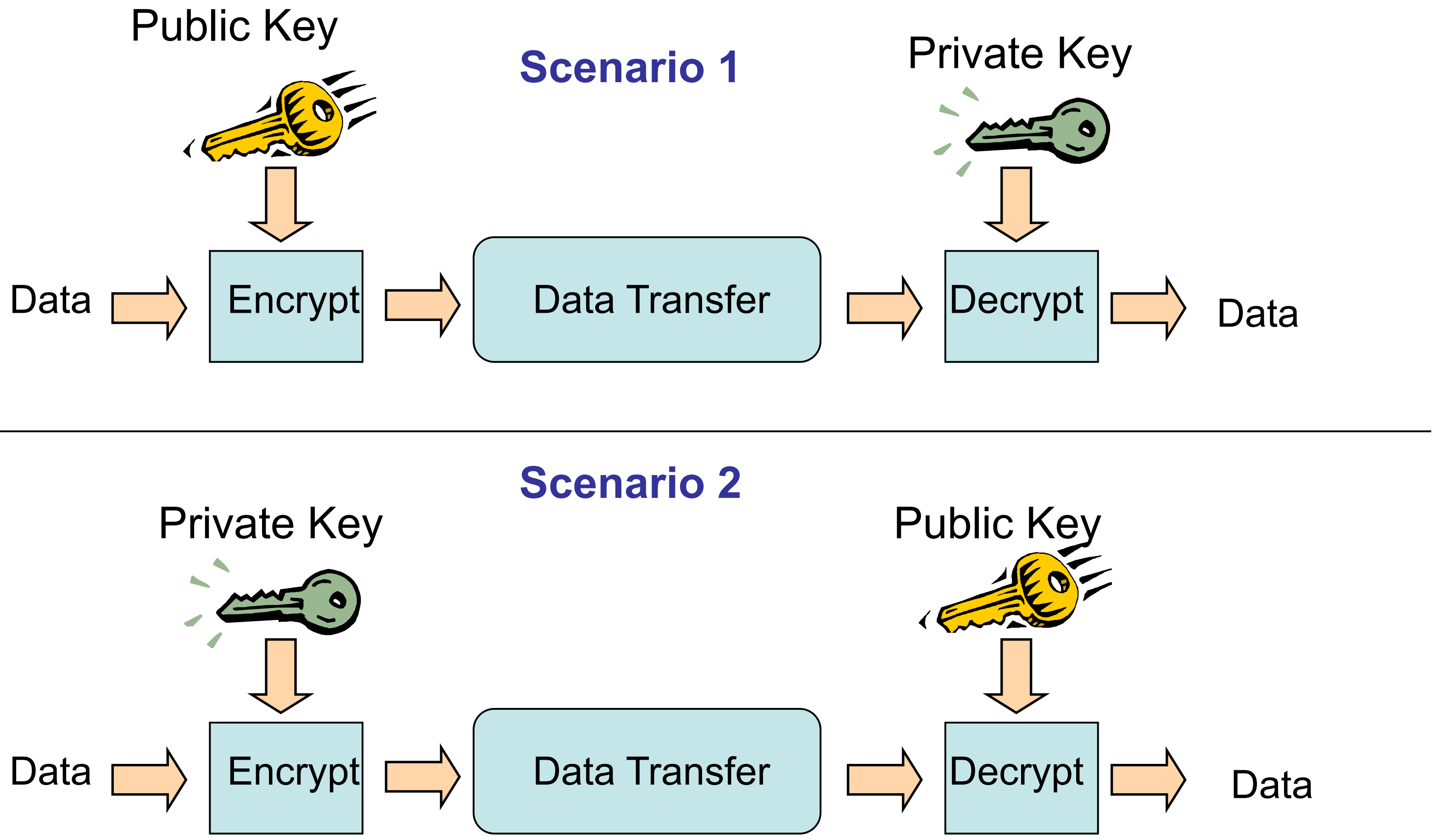
a. **Subject name**, which identifies the person or object that the certificate represents.

b. The **public key** belonging to the subject.

c. The **identity** of a Certificate Authority (CA) that has signed the certificate to certify that the public key and the identity both belong to the subject.

d. The **digital signature** of the named CA.

PKI - Illustrated



Scenario 1

Public Key

Private Key

Data → Encrypt → Data Transfer → Decrypt → Data

Scenario 2

Private Key

Public Key

Data → Encrypt → Data Transfer → Decrypt → Data

PKI Scenarios

- Scenario 1 - all users wishing to communicate with Bob encrypt data using his public key, only Bob can decrypt the data using his private key – *privacy*

- Scenario 2 – Bob wishes to communicate with all users, encrypts data using his private key, users subsequently use Bob’s public key for decryption - *authentication*

PKI in Detail

- A Certification Authority (CA) signs user-generated certificates
- Bob generates a public, private key pair, and a public key certificate using a tool like Gnu Privacy Guard (GPG)
- Bob approaches a CA to have his certificate signed
- Bob publishes his public key globally
- Alice wishes to communicate securely with Bob - uses Bob's public key to encrypt her message
- Bob receives and decrypts the message using his private key

Note: The CA which signed Bob's certificate must also be present in Alice's list of trusted CAs to allow Alice to verify the validity of Bob's certificate

Digital Signatures

- Digital Signatures allow verification of data integrity as well as the data origin
- A hash function (*H*) is defined as a function which takes an arbitrary length message as input, and generates a fixed size output, called the hash of the message
- A hash function is generally a one-way function i.e. a message cannot be retrieved back from its hash value
- Message Digest-5 (MD5) and Secure Hash Algorithm (SHA) are two examples of hash functions

Digital Signatures

- **A *Digital Signature* is defined as follows:**
  - For a message *m* originating from Bob's machine, with Bob's public key being Bob_{pub} ,the digital signature of *m* is given by:

    $$Signature = Encrypt(Hash(m))\ with\ Bob_{private}$$

  - This signature is appended to message *m* when Bob transfers it to a recipient.
  - A Digital signature and a digital certificate are two entirely different things.

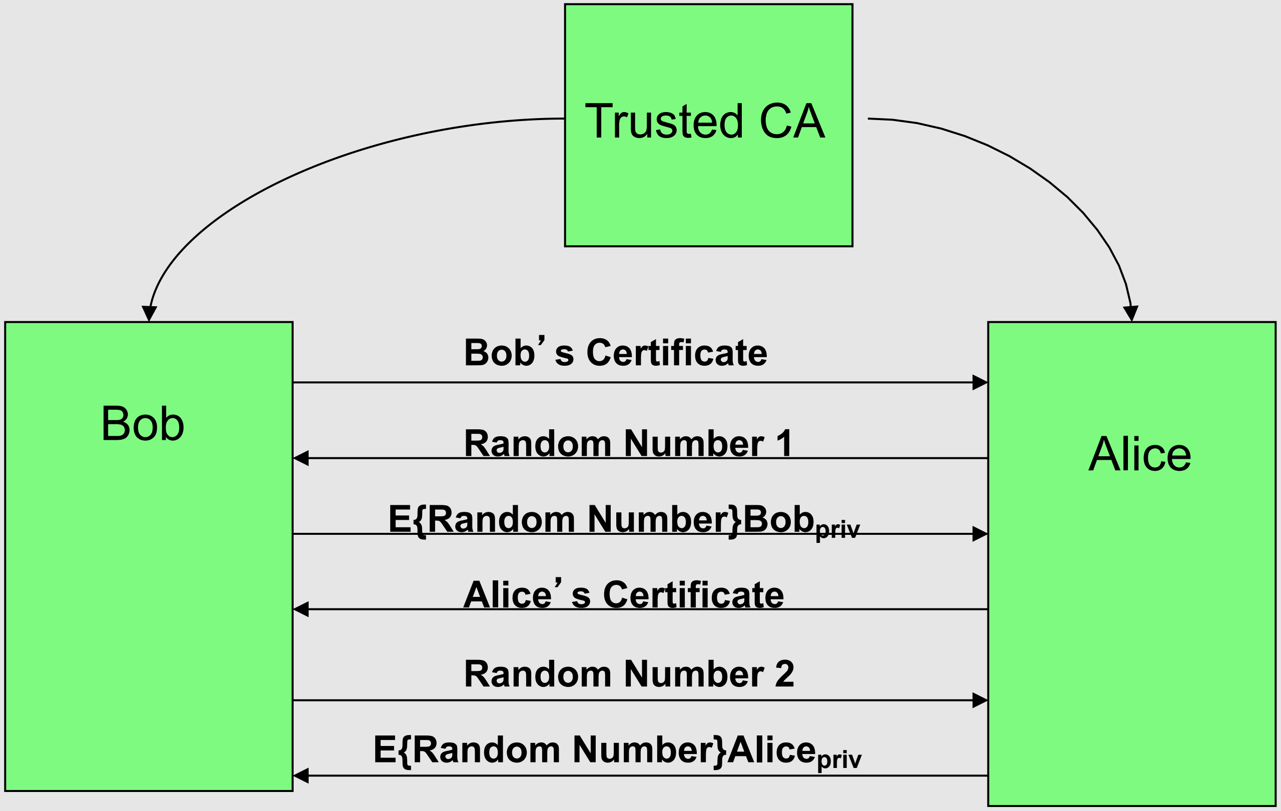
Digital Signatures

- – The recipient verifies the message origin as well as the message integrity by recalculating the hash of the received message, and comparing it with the received hash value
- – If the message is tampered with enroute to the destination, the computed hash will be different from the received value
- – If the two values are the same, the message integrity is verified, and the message is defined as being intact

Mutual Authentication

- Alice and Bob are two grid users with certificates, and trust a common CA
- Mutual authentication implies that Alice and Bob prove to each other their respective identities
- Prerequisite for this process is for Alice and Bob to have a signed copy of the CA's certificate

Mutual Authentication



Bob_{priv} = Bob’s Private Key

E{ } = Encryption Function

Mutual Authentication

Alice authenticating Bob:

- Alice requests for Bob's certificate
- Alice checks validity of certificate by checking the CA's signature
- Alice needs to make sure Bob is who he is claiming to be
- Alice sends a random number to Bob
- Bob encrypts it using his private key and sends it to Alice
- Alice decrypts the message using Bob's public key
- If the result is the same as the random number, Bob has successfully proven his identity to Alice

The above steps are repeated in the opposite direction for Bob to verify Alice's identity

Proxy Certificates

- Single Sign-On is the ability to access several grid resources without the need to have the Grid user type a passphrase again and again
- Proxy certificates provide the SSO feature to GSI users
- A proxy certificate is a new public-private key pair created with a shortened lifetime – usually 12 hours

Proxy Certificates

- The long term private key of the user is used to sign the proxy certificate
- The private key of the proxy certificate is not passphrase protected
- The short lifetime of a proxy certificate reduces the extent of damage that can be caused due to certificate theft/ compromise

GSI Delegation

- GSI enables users to delegate their proxy credentials to processes running on remote resources
- The remote processes and resources can thus act on a user’s behalf
- Used for complex applications such as: batch jobs that need access to Grid data storage using a particular user’s credentials

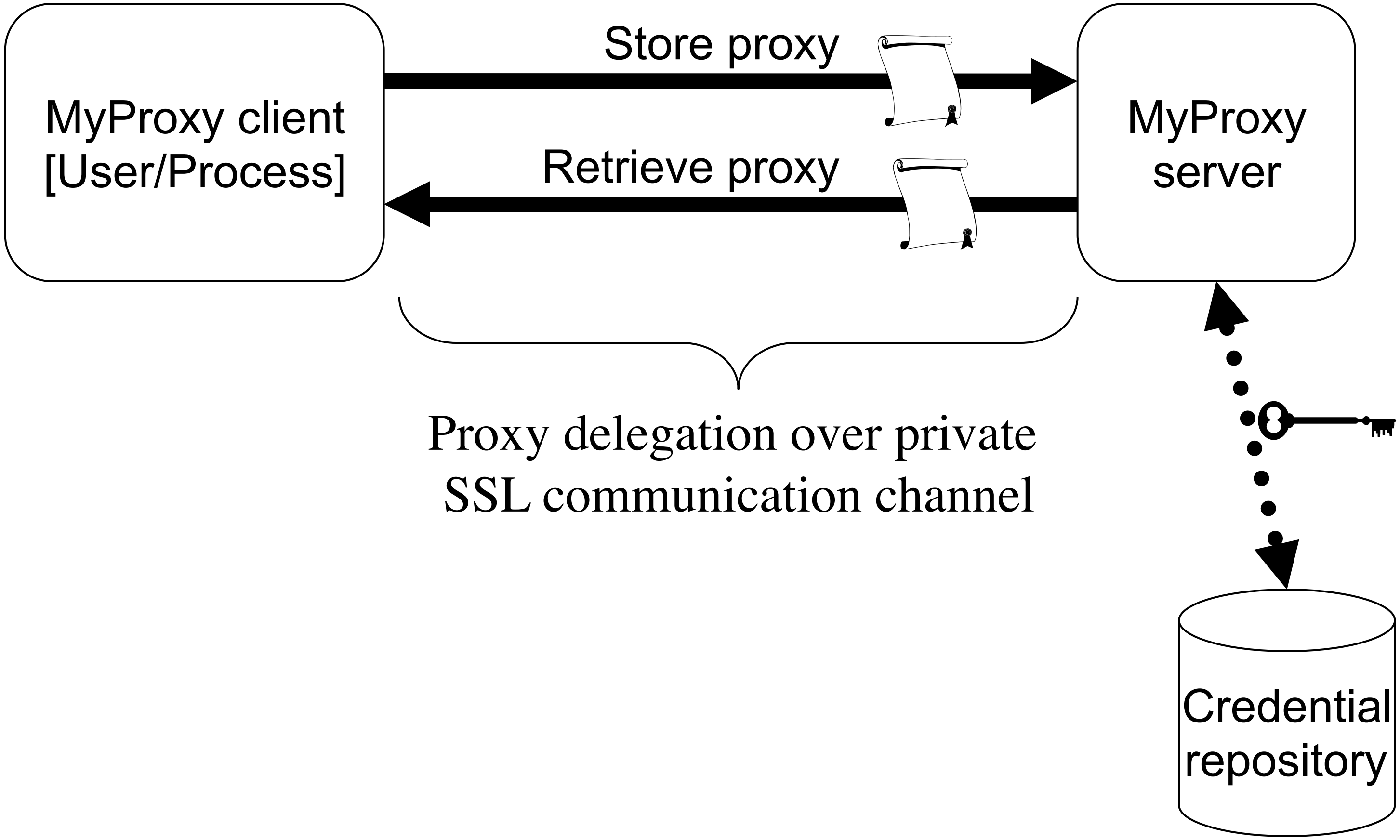
MyProxy

- MyProxy is a credential management service for Grid users
- It can also perform the functions of a CA if need be
- A traveling Grid user can delegate a proxy to the MyProxy server before leaving home, as well as access to his long-term credential
- Could then delegate short-term (12-hrs) certificates from MyProxy to other hosts to enable initiation of Globus jobs.

MyProxy

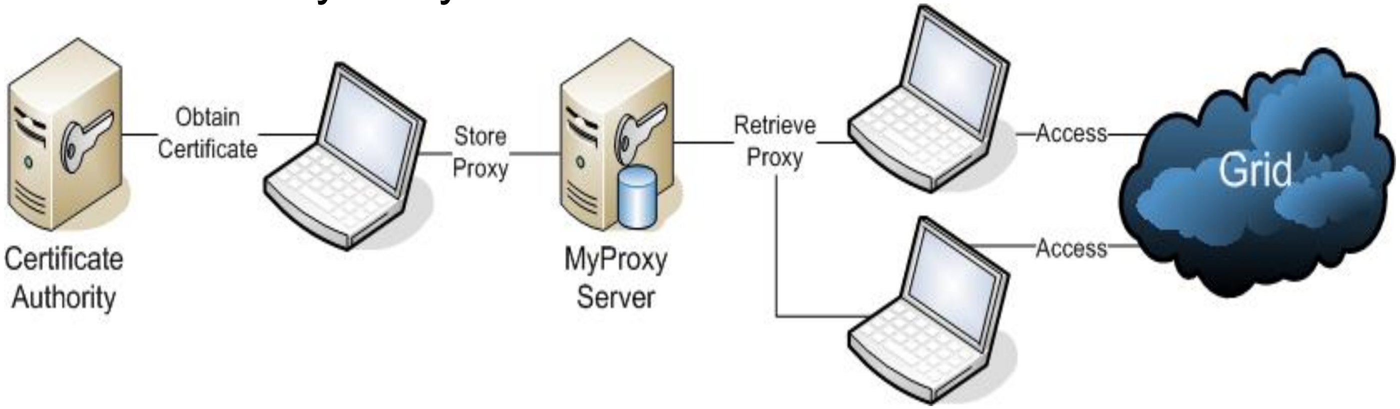
- **MyProxy repository can help users manage their credentials by:**
  - Securely storing the long-term private keys of users
  - Obtaining credentials from remote processes/users when needed
  - Using the credentials to act on a user’ s behalf

MyProxy Illustrated

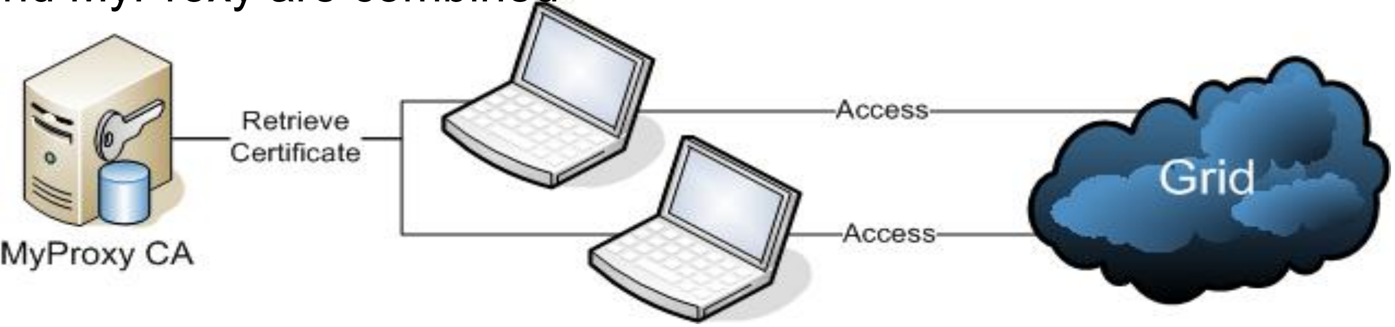


MyProxy Scenarios

CA different from MyProxy Server

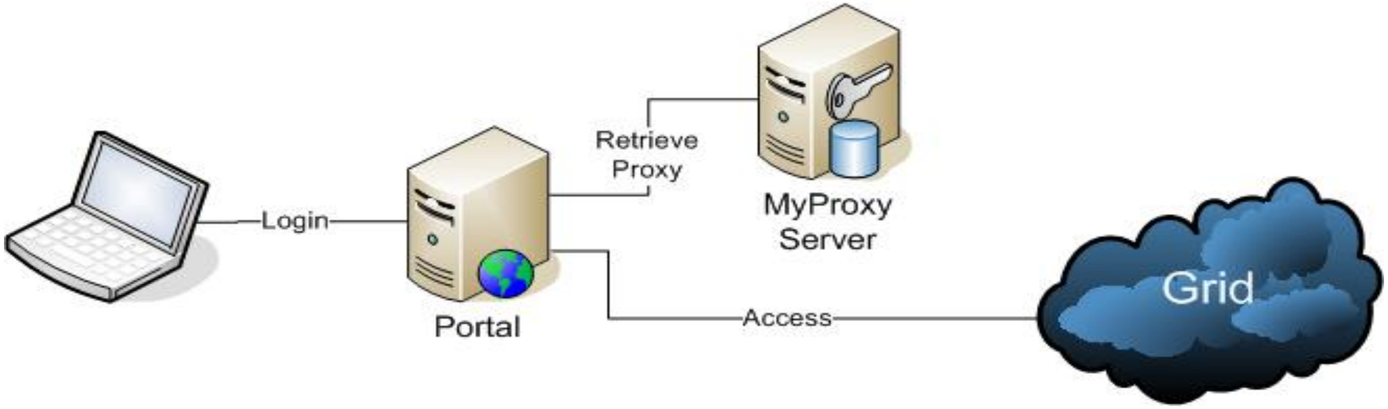


CA and MyProxy are combined



MyProxy Scenarios

A web-portal based MyProxy Access Mechanism



GSI Authorisation

- Community Authorisation Service (CAS) defines access control policies for Grid entities
- A CAS server is initiated for a community, where a community representative acquires a GSI credential to represent that community as a whole, and then runs a CAS server using that community identity.

GSI Authorisation

- Resource providers grant privileges to the community.
- Each resource provider verifies compatibility between CAS server policy and global policies
- Once a trust relationship has been established, the resource provider then grants rights to the community identity

GSI Authorisation

- New users and resources are enrolled based on policies defined in the CAS server
- The CAS server also performs fine-grained access control to Grid resources.
- When a user wants to access resources served by the CAS, that user makes a request to the CAS server.
- If the CAS server's database indicates that the user has the appropriate privileges, the CAS issues the user a GSI proxy credential with an embedded policy giving the user the right to perform the requested actions.

GSI Authorisation

- The user then uses the credentials from the CAS to connect to the resource.
- The resource then applies its local policy to determine the amount of access granted to the community, and further restricts that access based on the policy in the CAS credentials.
- As a result the user's privileges are limited to the intersection of those granted by the CAS to the user and those granted by the resource provider to the community.

Shibboleth – InterRealm Authentication

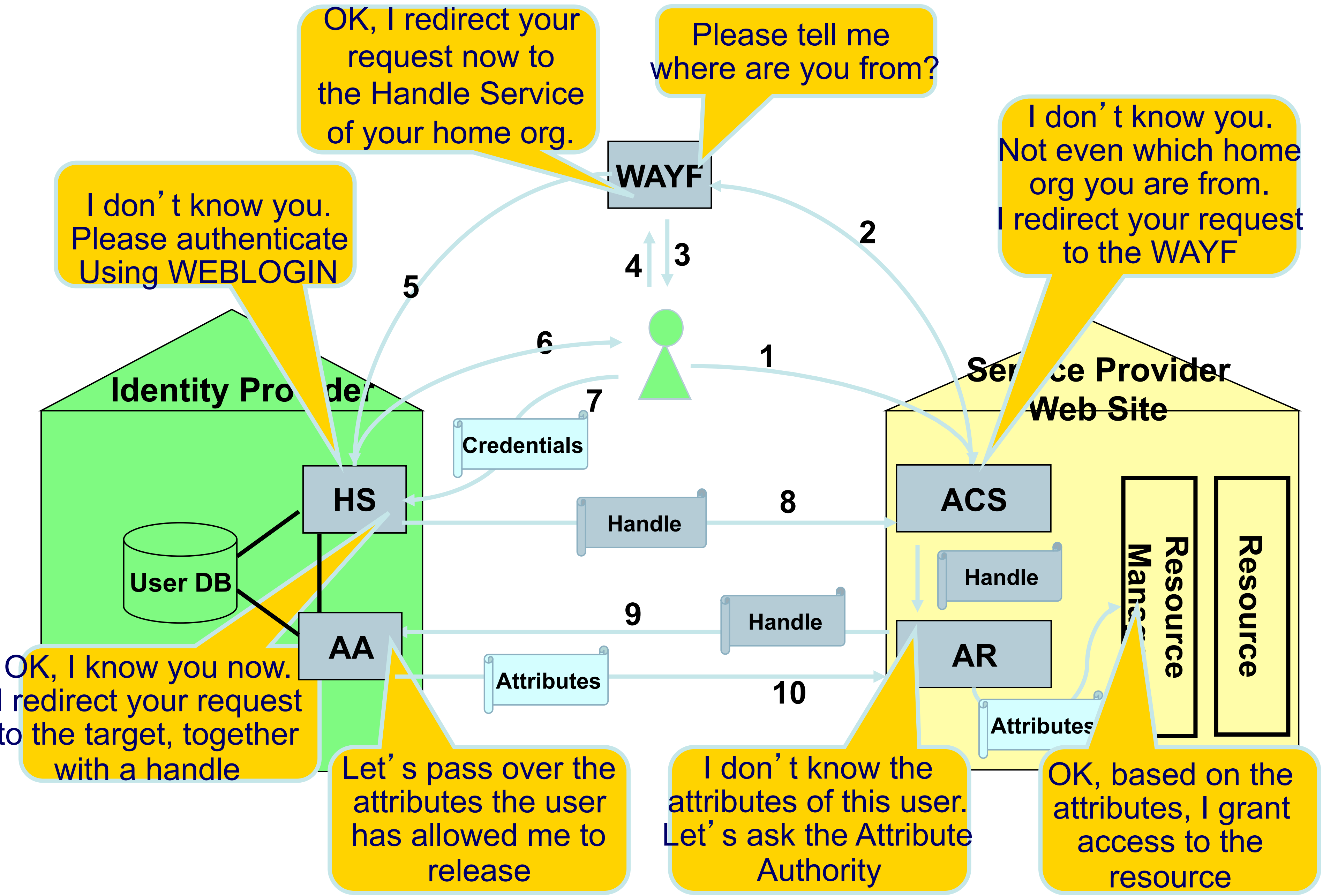
- **Internet2 project**
- **Allows for inter-institutional sharing of web resources (via browsers)**
  - Provides attributes for authorisation between institutions
- **Being extended to non-web resources**

Shibboleth

- **A Shibboleth Identity Provider is composed of single sign-on (SSO) and attribute authority (AA) services**
- **SSO: authenticates users locally and issues authentication assertions with a Handle**
  - An assertion is a short-lived bearer assertion
  - Handle is also short-lived and non-identifying
  - Handle is registered with AA
- **Attribute Authority responds to queries regarding handle**

Shibboleth

- A service Provider is composed of an Assertion Consumer and an Attribute Requestor
- Assertion Consumer parses authentication assertions
- Attribute Requestor: request attributes from AA
  - Attributes used for authorization
- A ‘Where Are You From’ (WAYF) service determines a user’s Identity Provider



I don’t know you. Not even which home org you are from. I redirect your request to the WAYF

Please tell me where are you from?

OK, I redirect your request now to the Handle Service of your home org.

I don’t know you. Please authenticate Using WEBLOGIN

OK, I know you now. I redirect your request to the target, together with a handle

I don’t know the attributes of this user. Let’s ask the Attribute Authority

Let’s pass over the attributes the user has allowed me to release

OK, based on the attributes, I grant access to the resource

From: Michael R Gettes
Slides