GSI-based Security for Web Services

Sriram Krishnan, Ph.D.

sriram@sdsc.edu

Topics Covered

- **High-level Overview**
  - Message and Transport Level Security
  - Authentication and Authorization
- **Implementation details of NBCR's initial prototype**
  - Authentication: Transport-level security using GSI-based certificates
  - Authorization: Basic Grid-map based authorization to restrict Web service access

Modeling and Analysis Across Scales

Organisms

Organ

Tissue

Cell

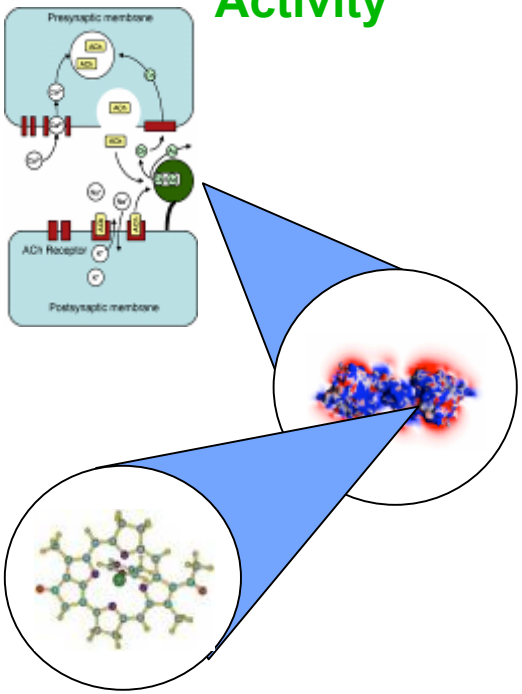
Subcellular

Macromolecular

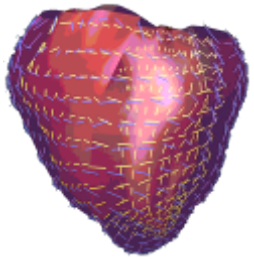
Molecule

Atom

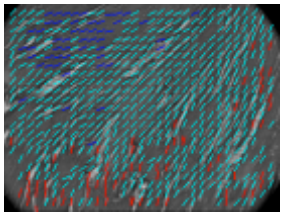
Modeling Synaptic Activity



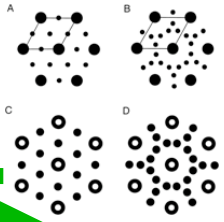
Modeling the Heart



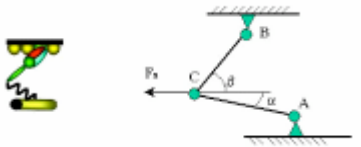
ventricles



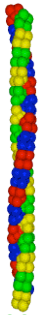
multicellular



lattice



crossbridge

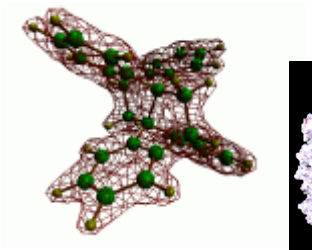


filament

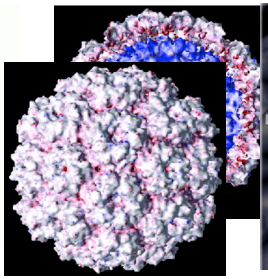
NBCR Tools Integrate Data, Construct Models and Perform Analysis across Scales

Computational Infrastructure for Multiscale Modeling

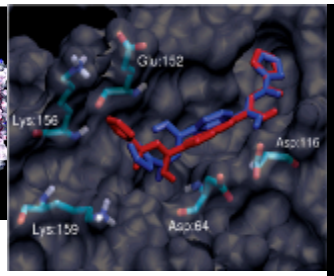
Set of Biomedical Applications



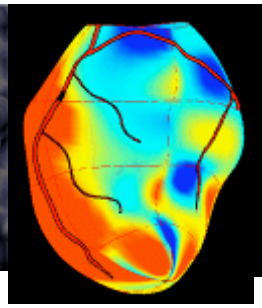
QMView
GAMESS



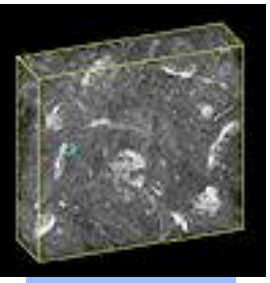
APBS



Autodock



Continuity

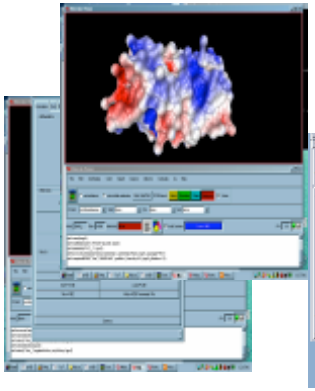


Gtomo2
TxBR

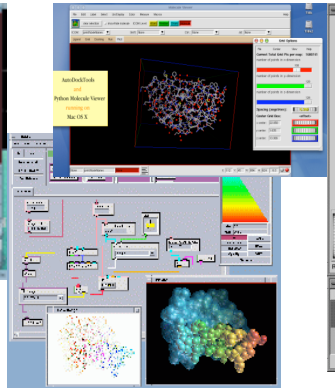
Infrastructure

Computational Grid

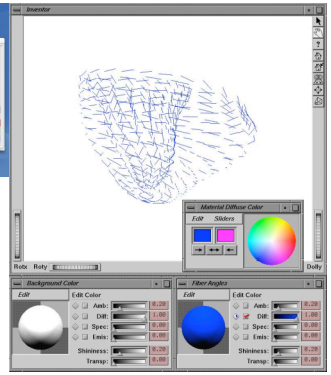
Rich Clients



APBSCommand

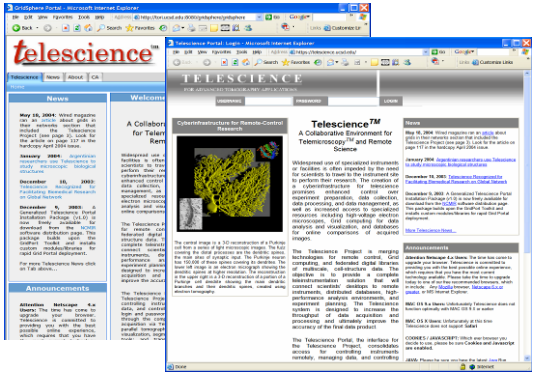


PMV ADT
Vision



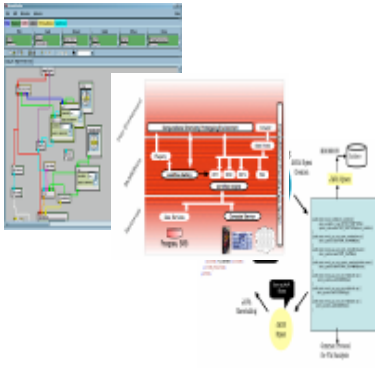
Continuity

Web Portals



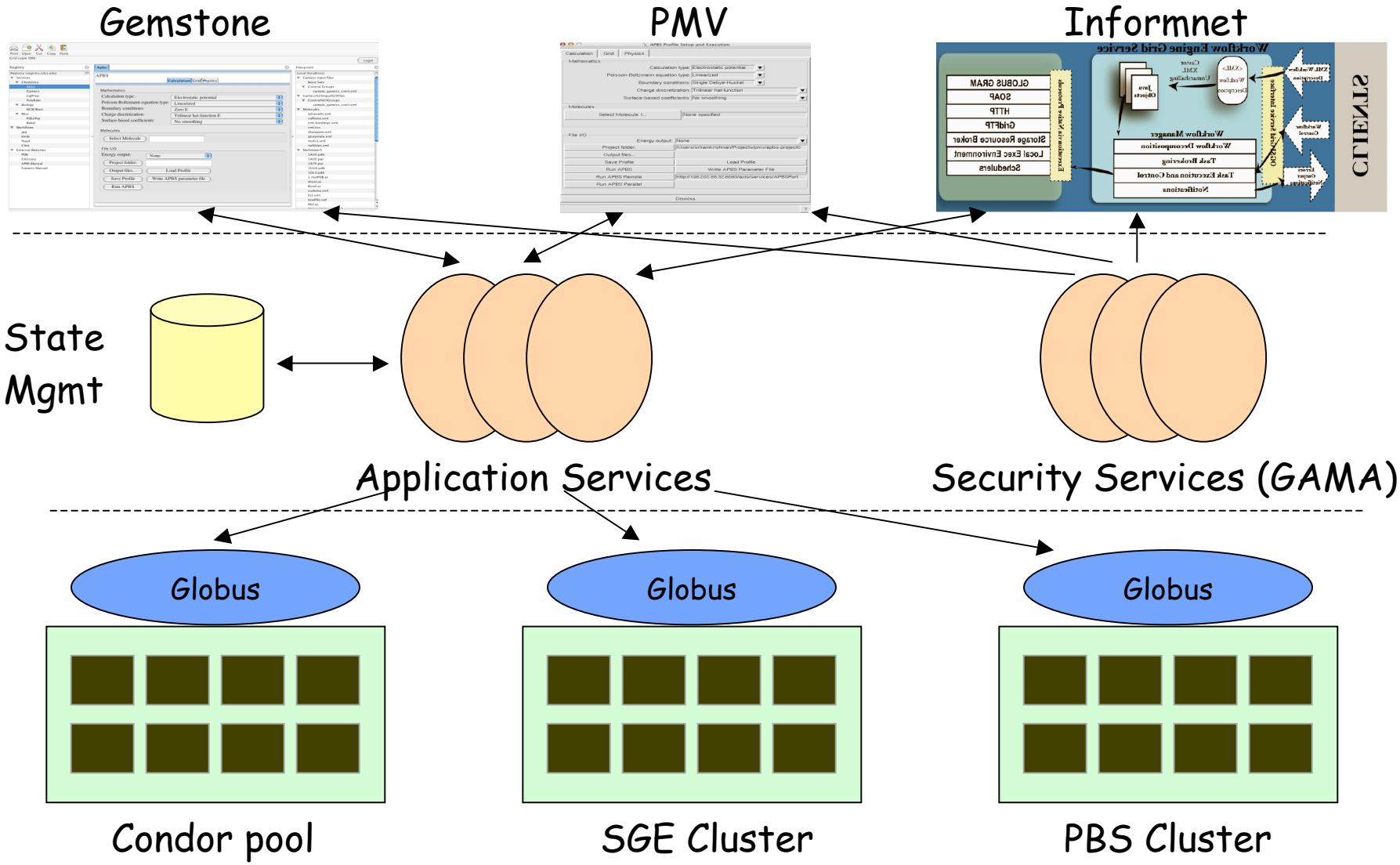
Telescience Portal

Web Services



Workflow
Middleware

Architecture Overview



End-to-end Security: Steps

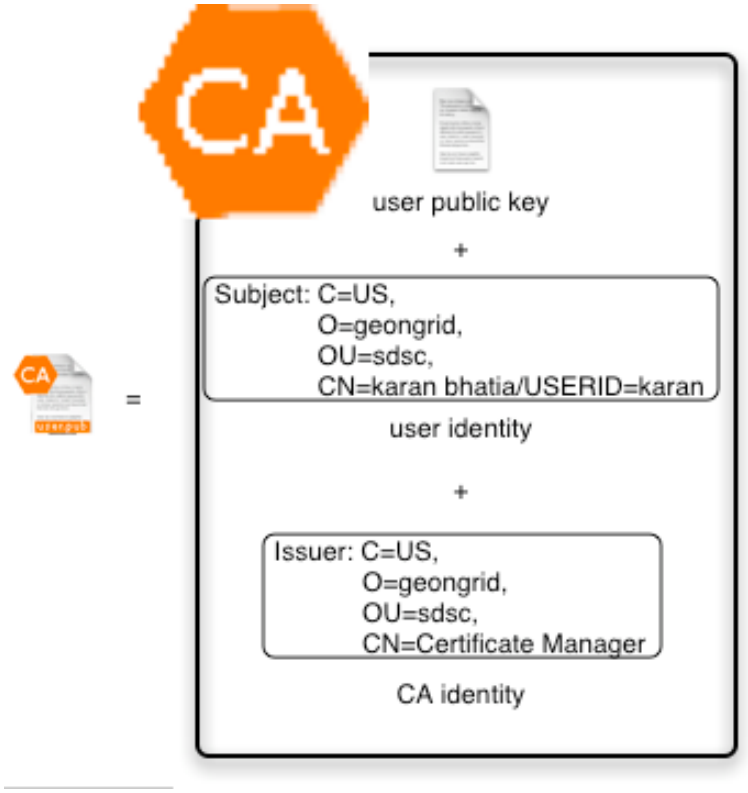
- **Authentication**
  - An entity identifies itself as a particular user
- **Privacy**
  - Messages sent on the wire are kept secret from anyone other than the intended recipient
- **Integrity**
  - Messages sent on the wire are not tampered with in any form
- **Authorization**
  - A user is given permissions to access a particular resource

Authentication, Privacy, Integrity: Alternatives

- **Public Key Cryptography**
  - X.509 certificates to identify entities, and corresponding private keys so sign/encrypt messages
  - SSL is a *de facto* standard for internet applications
- **Private (Secret) Key Cryptography**
  - Use of a shared secret key for encryption/decryption
  - Kerberos is the most widely used implementation

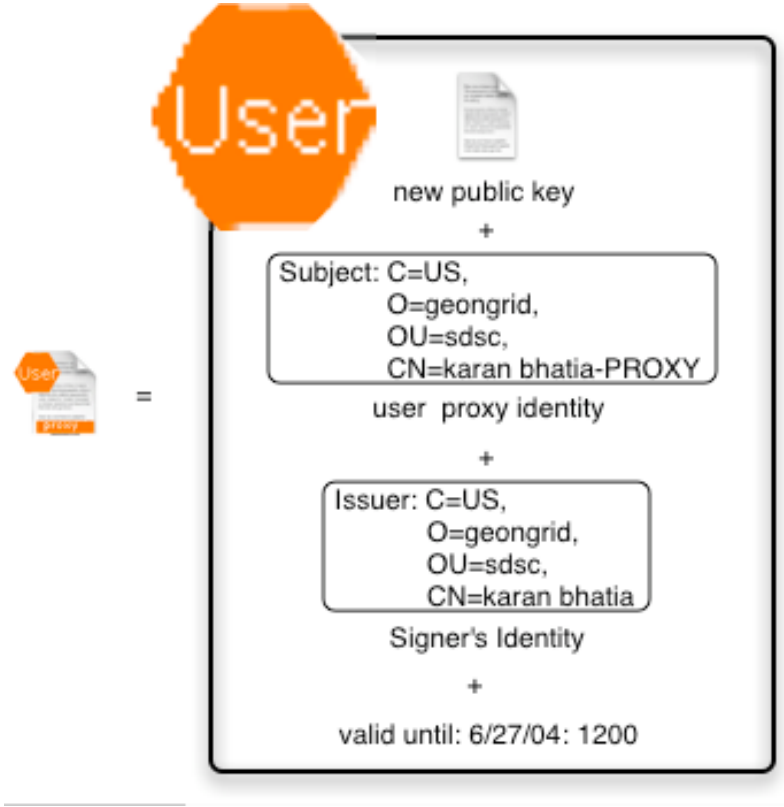
Grid Security Infrastructure (GSI)

- Every user and service on the Grid is identified via a X.509 certificate, a text file containing the following information:
  - A subject name identifying the person or object that the certificate represents
  - The public key belonging to the subject
  - 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
  - The digital signature of the named CA.



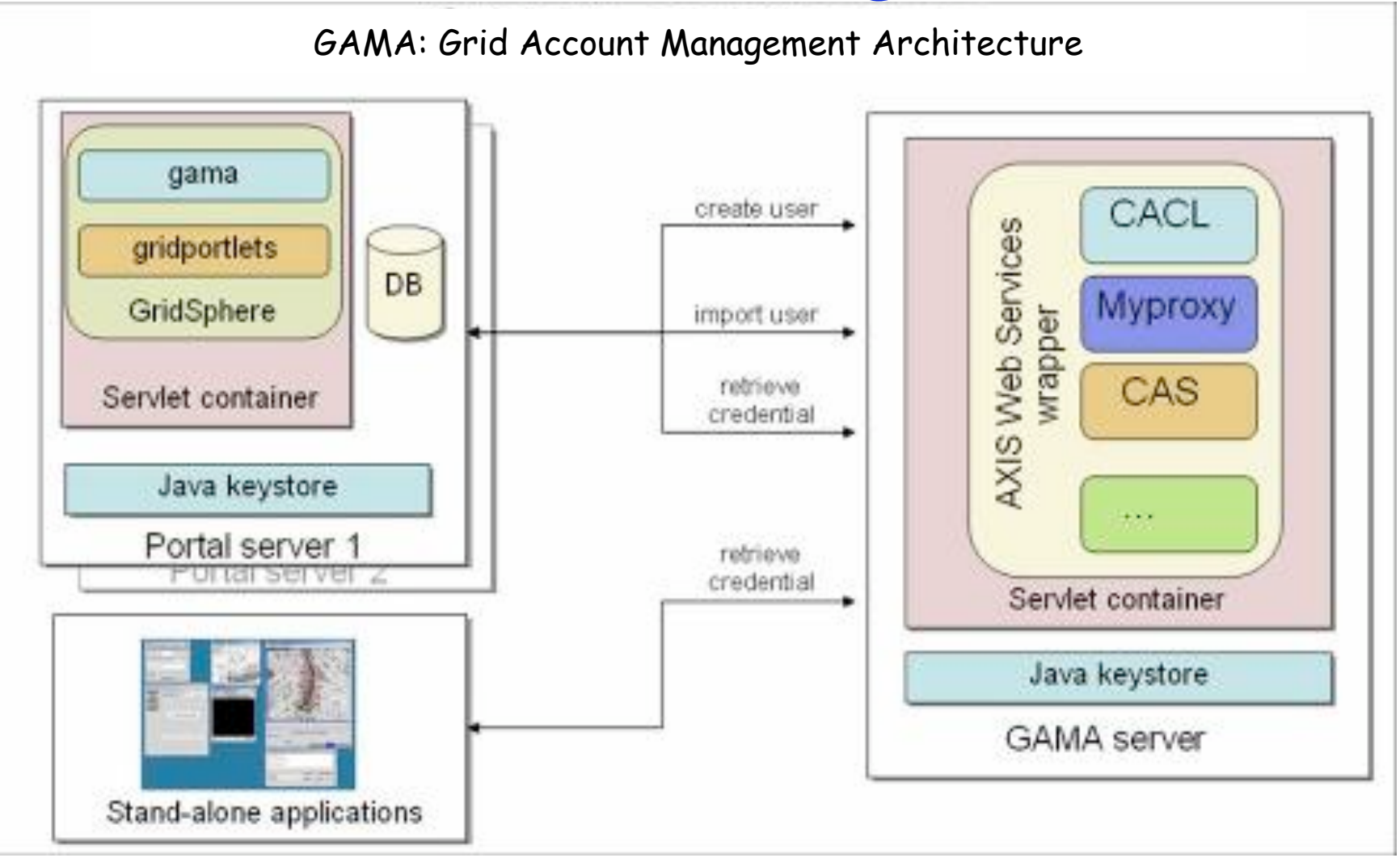
Proxy Certificate

- A *proxy* consists of a new certificate with a new public and private key
- The new certificate contains the owner's identity modified slightly to indicate that it is a proxy
- The new certificate is signed by the owner rather than a CA
- The certificate also includes a time notation after which the proxy should no longer be accepted by others
- Proxies have limited lifetimes in order to minimize the security vulnerability
- Proxies can be delegated to other entities to act on behalf of a particular user



Certificate Management

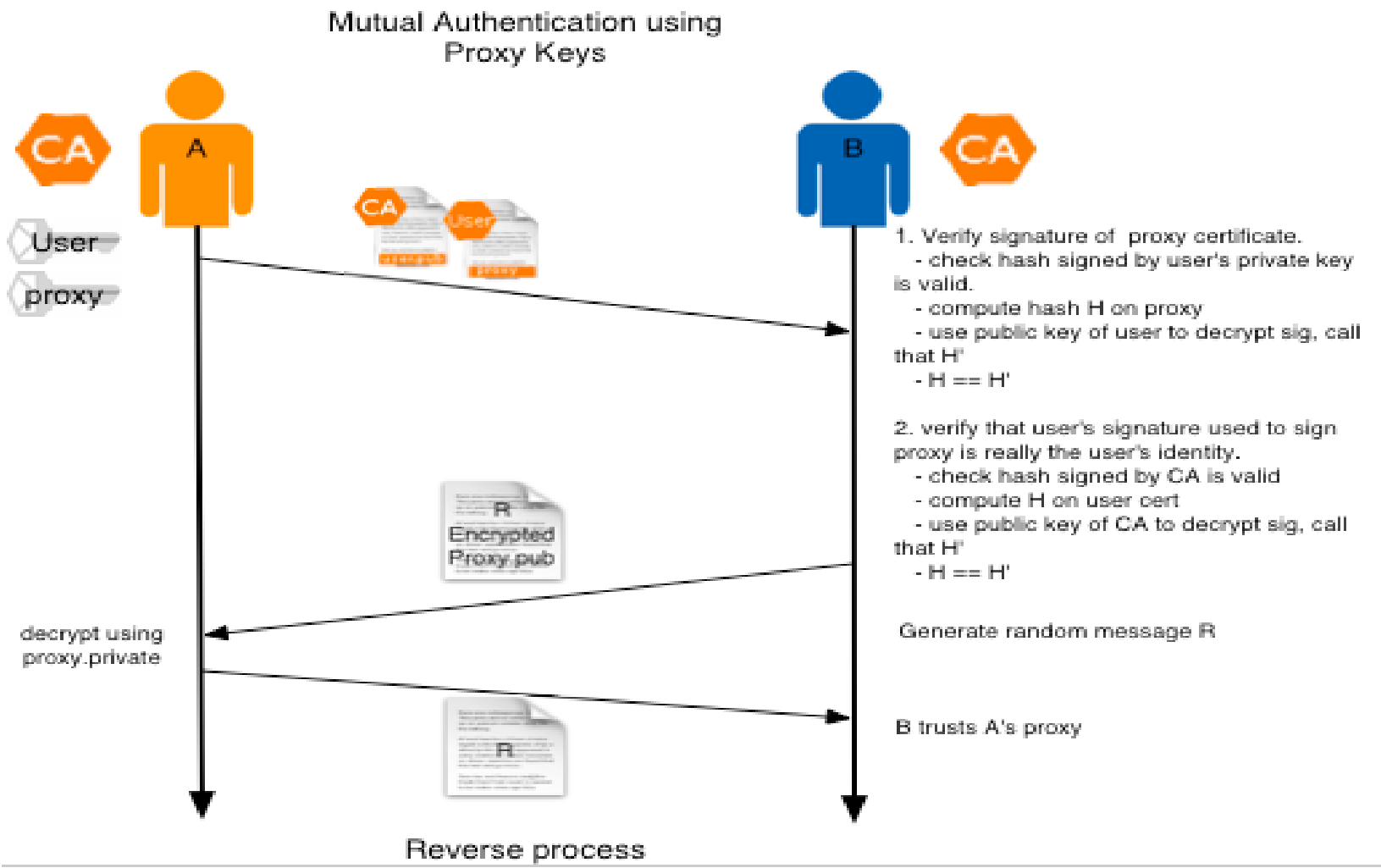
GAMA: Grid Account Management Architecture



Security: Techniques

- **Transport Level Security (TLS)**
  - Creation of a secure point-to-point connection between the client and server
  - Use of a Secure Sockets Layer (SSL) implementation
- **Message Level Security (MLS)**
  - SOAP messages are signed/encrypted over a non-secure socket connection
  - Use of emerging WS standards such as WS-Security, WS-Secure Conversation, XML Signatures, etc.

GSI TLS: Mutual Authentication



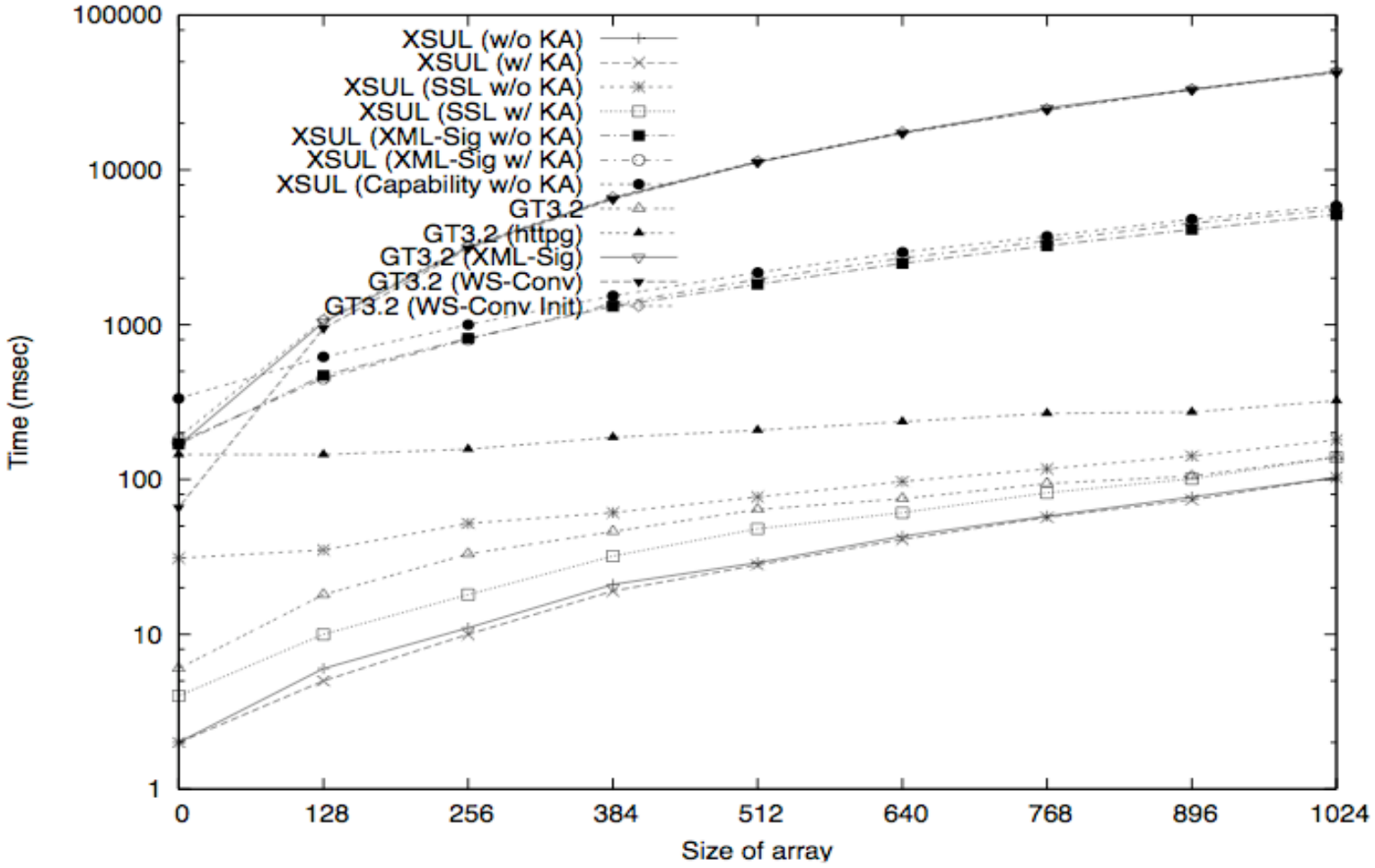
TLS: Pros and Cons

- **Pros**
  - SSL has been an internet standard for years
  - Fast implementations available
- **Cons**
  - Implemented at the socket layer - difficult to propagate security related information (e.g. client's DN, security assertions, etc) to higher levels in the software stack
  - Due to the secure point-to-point nature of the socket connection, it doesn't work for multi-hop connections, e.g. in the presence of firewalls, intermediaries, etc.

MLS: Pros and Cons

- **Pros**
  - No need for a secure point-to-point connection - works well for multi-hop connections
  - Since it is done at the message level, portions of messages can be encrypted - useful if messages can contain a mixture of sensitive and non-sensitive information
  - Authorization information (e.g. assertions) can be propagated easily to higher levels in the software stack
- **Cons**
  - Performance

MLS Performance: Shirasuna, et al (Grid2004)



MLS Performance: Bottleneck

- **XML manipulations are expensive (surprise, surprise!!)**
  - XML has to be canonicalized before signing or verification - this is very expensive (and becomes worse with larger data sizes)
  - Need for XML Canonicalization
    - Different SOAP toolkits may represent XML differently (e.g. namespaces, prefixes, order of attributes, etc) - the SOAP message can look different when it reaches the server
    - Logical equivalence of XML documents doesn't mean physical equivalence - however, physical equivalence is required to verify signatures, and decrypt messages

Authorization: Alternatives

- **Grid-map based**
  - Access Control List that maps client's DN to a user on a physical resource
  - Most basic, and commonly used technique
- **Community Authorization Service (CAS)**
  - User capabilities embedded inside generated proxy
  - Central authorization service responsible for creation of user roles, and access rights
  - Server grants access to user on the basis of the generated role
  - Most recent implementation based on SAML
- **Server-side call-outs**
  - Server makes call-outs to authorization services using the client's DN
  - Can be implemented in a variety of ways, including SAML

Implementation Details

Separating the facts from science fiction

Experience with GT4

- **GT4: Globus Toolkit 4, first implementation of the WSRF Framework**
- **Security**
  - Default is transport-level security
  - Message-level security can be used optionally
- **Authorization**
  - Authorization implemented as server-side call-outs (Users can implement their own call-outs, if need be)
  - Push-based mechanisms (e.g. CAS assertions) currently not supported

GT4 Security: Issues

- **The security implementations out of the box work only with WSRF services**
  - NBCR services are simple Web services, and couldn't simply be dropped into their GSI-enabled containers
- **Had to reverse-engineer their security implementation to use with plain Web services**
  - Didn't have any particular need for Message-level security
    - No multi-hop connections, or need to sign portions of messages
    - Performance of MLS was a big concern
  - Decided on using GSI-enabled TLS, and simple Grid-map based authorization for now

GSI-enabled TLS Setup

- **Use of the Java CoG Kit 4.0a1 implementation of GSI-based HTTPS**
- **Server setup**

```
<Connector className="org.globus.tomcat.coyote.net.HTTPSConnector"
        port="8443" maxThreads="150" minSpareThreads="25"
       maxSpareThreads="75"
        enableLookups="false" disableUploadTimeout="true"
        acceptCount="100" clientAuth="true"
        debug="3" scheme="https"
        cert="/Users/sriramkrishnan/certs/apbs_service.cert.pem"
        key="/Users/sriramkrishnan/certs/apbs_service.privkey"
        cacertdir="/Users/sriramkrishnan/.globus/certificates" />
```

GSI-enabled TLS: Client setup

- **Add a GSI-HTTPS Provider, if need be:**

```
if (httpsInUse) {
  SimpleProvider provider = new SimpleProvider();
  SimpleTargetedChain c= new SimpleTargetedChain(new HTTPSSender());
  provider.deployTransport("https", c);
  asl.setEngine(new AxisClient(provider));
  Util.registerTransport();
}
```

- **Set Web service client Stub properties**

```
IdentityAuthorization auth =
  new IdentityAuthorization("/C=US/O=nbcr/OU=sdsc/CN=apbs_service");
if (httpsInUse) {
  ((Stub) apbsPort)._setProperty(GSIConstants.GSI_AUTHORIZATION,
                                 auth);
  ((Stub) apbsPort)._setProperty(GSIConstants.GSI_CREDENTIALS,
                                 proxy);
}
```

Authorization

- **Simple Grid-map authorization implemented as an Axis Handler**
  - Axis uses a Handler-chain model - a message passages through a chain of handlers before it is processed by a *Pivot Handler*, that invokes the target service
  - Users can write their own Axis Handlers if they wish to process the message before/after a service is invoked
- **Grid-map Authorization Handler**
  - Retrieves the client's DN from inside the *HttpServletRequest* (which can be retrieved from the *MessageContext*)
  - Verifies that the client DN is found inside the service grid-map

Authorization: Setup

- **Add the Grid-map Authorization Handler to the *requestFlow* inside the server-config.wsdd**

```
<requestFlow>
  …
  <handler type="java:edu.sdsc.nbcr.common.GridMapAuthHandler">
    <parameter name="gridmap"
               value="/Users/sriramkrishnan/.globus/grid-mapfile"/>
  </handler>
</requestFlow>
```

Summary

- **Use of the Java CoG Kit to provide GSI-based transport-level security (via HTTPS) for Web services**
- **Provision of a simple Grid-map based Authorization service (implemented as an Axis Handler) to restrict service access**

Limitations & Future Work

- **Push-based authorization mechanisms (e.g. CAS) not supported**
  - Can be somewhat alleviated with the use of call-outs to authorization services
- **Support in different languages**
  - Currently, most SSL implementations do not support full-path proxy validation
  - OpenSSL version 0.9.8 (currently Beta 6) will support the above
    - Can be used by clients written in C, C++, Python, etc.

Questions & Discussions