Comparison of Advanced Authorisation Infrastructures for Grid Computing

A.J. Stell, Dr R.O. Sinnott, Dr J.P. Watt
National e-Science Centre
University of Glasgow, UK
ajstell@dcs.gla.ac.uk
ros@dcs.gla.ac.uk
jwatt@dcs.gla.ac.uk

Abstract

The widespread use of Grid technology and distributed compute power, with all its inherent benefits, will only be established if the use of that technology can be guaranteed efficient and secure. The predominant method for currently enforcing security is through the use of public key infrastructures (PKI) to support authentication and the use of access control lists (ACL) to support authorisation. These systems alone do not provide enough fine-grained control over the restriction of user rights, necessary in a dynamic Grid environment. This paper compares the implementation and experiences of using the current standard for Grid authorisation with Globus - the Grid Security Infrastructure (GSI) - with the Role-Based Access Control (RBAC) authorisation infrastructure PERMIS. The suitability of these security infrastructures for integration with regard to existing Grid technology is presented based upon experiences within the JISC-funded DyVOSE project.

1. Introduction

Institutions in science and industry are increasingly turning to distributed computer technology to achieve higher efficiency and greater production. Grid technologies allow distributed resources such as data storage or CPU compute power to be made available to a much wider user base beyond the original domain. To gain optimum use of this resource sharing, institutions form collaborative communities known as Virtual Organisations (VOs). Within these VOs, a flexible approach to resource use and acquisition must be adopted but as the degree of trust between participants varies, it must not be at the expense of security. To enforce security in such an open and dynamic environment presents many challenges and any solution must allow for a variety of fine-grained security policies to be realised. At the same time, this infrastructure needs to be simple to use, set up and deploy.

The most common approach for security is based upon authentication, whereby a user makes an action request and they are challenged to prove their identity. This is commonly realised by means of a Public Key Infrastructure (PKI). In a PKI, a root of trust issues certificates and keys to trusted users; these are, upon request, presented to a gate-keeper that protects the resource. To enforce authorisation, the process of allowing a user certain access privileges based on who they are, the most basic (and currently most widespread) method is to use an Access Control List (ACL). The ACL simply lists which users are allocated given privileges. These privileges are often achieved through mapping user requests to specific accounts on those protected resources, e.g. through a grid-mapfile which maps a distinguished name (DN) to a local user account.

Both these methods of security are very coarse-grained and static in their ability to ascertain the privileges of a user, and hence their ability to provide a decision on a resource request. Grid technology requires dynamic and quick authorisation decisions, once again emphasising a balance between flexibility and security.

Many solutions to this have been proposed in the Grid community and currently no one standard has been widely adopted. PERMIS [5, 6, 7], CAS [2], VOMS [8], Cardea [3] and Akenti [1] are all examples of authorisation infrastructures. In this paper we present the implementation effort involved in setting up and using the Grid Security Infrastructure

(GSI) [4], which exemplifies the use of an ACL, and PERMIS, which uses an advanced infrastructure based on Role-Based Access Control (RBAC). The implementation and application of these infrastructures is discussed along with an outline of the performance overheads in applying these technologies.

2. Authorisation Background

Authorisation is closely linked to authentication. Once a user has had their identity validated at a remote resource, it is essential that users actions are restricted based on who they are, what they are trying to do, and in what context etc. There are various methods of enforcing this restriction, the simplest method being the use of an Access Control List (ACL), which lists what users have access to a privilege.

Essentially, a user presents their credentials at the gate-keeper to a resource, which consults a list of users. This basic authorisation structure extends the concept of authentication and no more. If the user cannot authenticate to the satisfaction of the gate-keeper then the resource request will be denied. A problem that arises when trying to apply this method to a dynamic Grid environment is that only one list exists, where there could be many privileges that require different ACLs. For example, a user might need access to a given resource for different purposes within a given VO. Having a single list with a predefined set of accounts and user DNs does not support this multi-role approach. This is a solution that would not scale well in a large VO.

A more sophisticated method of applying authorisation controls is through use of Role-Based Access Control (RBAC) mechanisms, which allow Privilege Management Infrastructures (PMI).

2.1 PMI & Role-Based Access Control

The relationship between a PMI and authorisation is similar to the relationship between a PKI and authentication. Consequently, there are many similar concepts in the two types of infrastructure.

Central to a PMI is the idea of the attribute certificate (AC), which maintains a binding between the user and their privilege attributes. It is similar in notion to the public key certificate in a PKI. The entity that signs a public key certificate is a Certification Authority (CA); the entity that signs attribute certificates is called an Attribute Authority (AA). The root of trust of a PKI is often called the root CA, which can delegate this trust to a subordinate CA; the root of trust of a PMI is called the Source of Authority (SOA). The SOA may have subordinate authorities to which it can delegate powers of authorisation. Certificate Revocation Lists (CRLs), which show a list of certificates that should not longer be accepted as valid, exist in a PKI; Attribute Certificate Revocation Lists (ACRLs) exist in a PMI [15].

The critical idea in a PMI is that the access rights of a user are not held in an ACL but in the privilege attributes of the ACs that are issued to the users. This is the central idea behind RBAC – the privilege attribute will describe one or more of the user's rights and the target resource will then read a user's AC to see if they are allowed to perform the action being requested. This de-couples the user's privileges from their local identity and allows a more dynamic and flexible approach to access control.

The X.812 | ISO 10181-3 Access Control Framework standard [19] defines a generic framework to support this type of authorisation, depicted in figure 1.

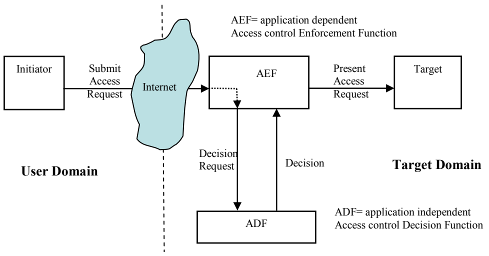
AEF= application dependent
Access control Enforcement Function
Initiator
Submit Access Request
Internet
AEF
Present Access Request
Target
Decision Request
Decision
User Domain
Target Domain
ADF
ADF= application independent
Access control Decision Function

Figure 1: Overview of X.812 Access Control Function

In this model, the initiator attempts to access a target in a remote domain. Two key components support authorised access to the target: a Policy Enforcement Point (PEP), described in the figure as the Access control Enforcement Point (AEF), and a Policy Decision Point (PDP), described as the Access control Decision Function (ADF). The PEP ensures that all requests to access the target are run through the PDP and the PDP casts the authorisation decision on the request based on a collection of rules (policies). To provide a generic interface between this framework and grid-enabled applications, an API has been proposed and created.

2.2 GGF SAML AuthZ API & PERMIS

The GGF (Global Grid Forum) have put forward an API that provides a generic PEP, which can be associated with an arbitrary authorisation infrastructure. The specification for Grid technologies is an enhanced profile of the OASIS (Organisation for the Advancement of Structured Information Standards) Security Assertion Markup Language (SAML) v1.1 [21].

The OASIS SAML AuthZ specification defines a message exchange between a PEP and PDP consisting of an *AuthorizationDecisionQuery* (which contains a *subject*, a *resource* and an *action*) going from PEP to PDP, and an assertion returned containing a number of *AuthorizationDecisionStatements*.

The GGF SAML AuthZ specification [20] defines a *SimpleAuthorizationDecisionStatement* (a boolean stating "granted/denied") and an *ExtendedAuthorisationDecisionQuery* that allows the PEP to specify whether the simple or full authorisation decision is to be returned. Figure 2 shows the interactions supported by this API.

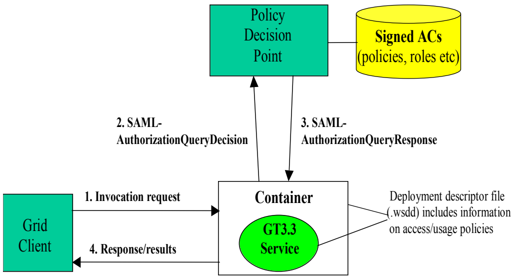
Policy Decision Point
Signed ACs
(policies, roles etc)
2. SAML-AuthorizationQueryDecision
3. SAML-AuthorizationQueryResponse
Grid Client
1. Invocation request
4. Response/results
Container
GT3.3 Service
Deployment descriptor file (.wsdd) includes information on access/usage policies

Figure 2: GGF SAML AuthZ API

By using this API, a generic policy enforcement engine can be constructed, that can be used by arbitrary grid services. Instead of having to explicitly create new policy engines for every application, the information can be incorporated into the deployment descriptor of the service and together with the policy identifier, the policy repository and the user DN, authorisation checks can be made for every method that is accessed on a service.

The PDP in the model above has been realised in the form of the Privilege and Role Management Infrastructure Standards Validation (PERMIS) initiative. This is an EC project that has built an authorisation infrastructure to realise a scalable X.509 Attribute Certificate based PMI.

The PERMIS software provides an RBAC authorisation infrastructure that uses XML based policies, specifying the access control decisions to be made for given resources within a VO. The rules that the policy covers include:

- subject definitions
- source of authority definitions
- roles and their hierarchies
- target resources
- which roles are allowed to perform which actions

Through implementing PERMIS, a dynamic and flexible authorisation infrastructure is established.

3. Establishing and Using Grid Security Infrastructures

To explore Grid security requires Grid services to have been implemented. Within the BRIDGES [29] and DyVOSE [11] projects various Globus Toolkit (v3.3) services have been prototyped. The steps for creating these services are similar:

- Create a schema file in GWSDL (Grid Web Services Description Language – a temporary version of WSDL for use with the OGSI specification).
- Implement the service operations
- Construct a deployment descriptor for the service
- Generate and compile the necessary stub classes.
- Package these into a Grid ARchive (GAR) file, and deploy it into the Globus container. The service URI is published upon starting the container.

The client that uses this basic grid service must have access to two classes specifically generated for this service:

- The first implements the *GridServiceLocator* interface and provides the handle on the service instance.
- The second implements the *GridServicePortType* interface and is the stub, which interacts with the service handle. This is the class instance upon which the service operations are performed.

To provide access to the authorisation infrastructures requires that the infrastructures are set up and that the services and clients developed above are modified to use the infrastructure.

3.1 Grid Security Infrastructure (GSI)

Some general security features must be put in place before GSI can be used. A host certificate and key must be available to allow the Globus container to be started up securely. Also the root certificate and signing policy of the CA that issued these certificates must be available – in this implementation, the UK e-Science Certificate Authority [18] (based in the Rutherford Appleton Laboratory) was used, as this provides trust on a national scale.

To set up the authorisation infrastructure in GSI requires a *grid-map* file, which provides the ACL. The *grid-map* file is traditionally placed in the */etc/grid-security/* folder. The file is a list that maps Distinguished Names (DNs) to local usernames. This provides local authorisation control, with access to requested (secured) resources being permitted or denied depending on whether the DN produced by creating a proxy certificate corresponds to an entry on that list. To demand additional security requirements such as encryption or signatures, a customised security configuration file can be written. (The default is *gsi-security-config.xml*.)

To make use of this infrastructure requires modification of the deployment descriptor to point to the ACL. The service is pointed to the gridmap file by adding the following parameters to the deployment descriptor (*server-deploy.wsdd*) before building and deployment:

```
<parameter = "authorisation",
        value = "gridmap"/>

<parameter = "gridmap",
        value = "/etc/grid-security/
                        grid-mapfile"/>
```

Modifying the deployment descriptor to use a security configuration file is done by adding a similar parameter called "securityConfig". To enable strong authentication on the client requires setting properties on the stub. To enable the client to "shake hands" with the service in a secure manner, properties on the relevant stubs must be set so that an action will only take place if strong authentication has taken place and the requesting action call has been found to be valid. In order to do this the following code is inserted into the client class to set these properties:

```
((Stub)stubname)._setProperty(
        Constants.GSI_SEC_CONV,
        Constants.ENCRYPTION);
```

This property demands that a secure conversation is set up by requiring that the stub has method calls encrypted.

```
((Stub)stubname)._setProperty(
        Constants.AUTHORIZATION,
        HostAuthorization.getInstance())
```

This property allows the client call to be authorised if a hostname is returned. This is an example of client-side authorisation that is performed in addition to the grid-map authorisation set up on the server side.

Once these modifications have been made, all services that have these security features enabled, can be accessed by only those users with DNs present on the ACL. (This is essentially security at the container level.) The users identify themselves by creating proxy certificates from their own certificate, located in their home directory.

3.2 PERMIS

In the context of securing grid services, PERMIS is provided in the form of a grid service itself, deployed into the same container as the service to be restricted. This PERMIS service acts as the PEP between the target and the PDP. To implement the service, a GAR file is downloaded from the PERMIS development pages [28] and deployed into the container.

The infrastructure requires a *Lightweight Directory Access Protocol (LDAP) server* to store the roles and policies in the form of attribute certificates. The server is set up so that the DNs on the proxy certificates of the domains, users and central managers correspond to the location of the user certificates. This allows the client to be identified when making the service call. The version used is OpenLDAP v2.1, obtained by CVS from the OpenLDAP software repository [26]. The LDAP server process can reside on a separate machine as long as this is visible from the machine running the Globus container, by means of an IP address.

In the DyVOSE project [10], to provide users with certificates that corresponded to the LDAP structure, it was necessary to create our own local certificate authority (CA). This involved creating a root certificate using OpenSSL [22], signing this certificate and then using it to create and sign all subsequent user certificates. This root certificate, originally created in Privacy Enhanced Mail (PEM) format, was converted to Direct Encoding Rules (DER) format and was imported into the

Source of Authority (SOA) node on the LDAP server. The p12 (Personal Information Exchange) file created for the SOA user was then used to sign all the attribute certificates created using the privilege allocator (see below). As is standard when using Globus in a security context, the user certificate and key must be extracted from the p12 file that has been distributed from their certificate authority and must be placed in a folder called *.globus/* beneath their home directory. The signing policy and root certificate associated with the certificate authority must be placed in a folder called *certificates/* beneath the aforementioned *.globus/* folder, for each user. UK e-Science [18] certificates were not used for DyVOSE, but will be in future project implementations. This will allow a much wider user base as UK e-Science certificates are trusted on a national scale, whereas a local CA is only trusted by the certificates that it issues and has local control over. To attempt to scale the local CA model up would involve complicated issues such as creating CA "bridges". [23]

Two important user tools exist that allow the necessary XML security policies and attribute certificates to be created. The policy editor is a graphical user interface that allows XML policies to be created using English semantics. By presenting the concept of domains and roles in terms that are understandable by non-computer scientists, the policy editor takes the input from the policy writer and generates a policy that fits the necessary XML syntax. The critical parts of the policy are the users, targets and actions specified. The other important graphical tool is the privilege allocator, which allows attribute certificates to be created. These attribute certificates, in DER format, can comprise either the XML policy or the role that a user can take.

To allow the grid service to be authorised using the PERMIS Authorisation Service, three parameters must be added to the deployment descriptor, either in *server-deploy.wsdd* before the service is deployed or under the relevant service name in *server-config.wsdd*, located in the Globus installation directory. These parameters are:

```
<parameter = "authorization",
value = "custom"/>

<parameter = "authzClass",
value = "org.globus.ogsa.impl
        .security.authorization
        .SAMLAuthorisationCallout" />

<parameter = "authzService",
value = "http://localhost:8080/
        ogsa/services/decider/
        PermisAuthorizationService"/>
```

These parameters indicate the customised nature of the authorisation, the class that will be used for implementing the authorisation service and the URI that will actually provide the authorisation service.

Additionally, the PERMIS service must be pointed towards the LDAP server, the policy that you wish to use and the source of authority that manages the policy and roles within this domain. In order to do this, the following parameters must be added (or modified) within *server-config.wsdd* under the "*decider/PermisAuthorisationService*":

```
<parameter = "LDAP",
        value = "ldap://cassini.
                nesc.gla.ac.uk:389"/>

<parameter = "OID",
        value = "1.0.0.1"/>

<parameter = "SOA",
        value = "cn=Administrator,
        o=University of Glasgow, c=GB"/>
```

The DN given by the client user's proxy certificate provides the identification necessary for the PERMIS engine to recognise what user is making the service call. To pick the DN up and use it in this context requires extra Globus security code to be inserted into the client, allowing strong authentication to take place between client and server. The necessary lines are as follows:

```
((Stub)stubname)
        ._setProperty(
        Constants.GSI_SEC_CONV,
        Constants.SIGNATURE)

((Stub)stubname)
        ._setProperty(
        GSIConstants.GSI_MODE,
        GSIConstants.GSI_MODE_NO_DELEG)

((Stub)stubname)
        ._setProperty(
        Constants.AUTHORIZATION,
        HostAuthorization.getInstance())

((Stub)stubname)
        ._setProperty(
        Constants.GRIM_POLICY_HANDLER,
        new IgnoreProxyPolicyHandler())
```

These properties require that the credentials are signed, that they cannot be from a delegated party, that the container must be authorised using the host credentials and that any policies created and maintained using the Globus GRIM (Grid Resource Identity Mapper) facility are ignored.

4. Experiences and Performance Analysis of Security Infrastructures

The JISC-funded DyVOSE project is investigating advanced RBAC infrastructures (PERMIS) for dynamic establishment of VOs within a teaching environment, specifically as part of the Advanced MSc Grid Computing module at the University of Glasgow. Students at Glasgow were asked to develop a Globus service (version 3.3 of the toolkit) that wraps a Condor based application, which itself offers two methods to search and sort a large text file (the complete works of Shakespeare – 5MB). The students were split into two groups with the PERMIS authorisation policy to ensure that the sort method could only be invoked by members of their own student group and the lecturing staff, and that the search method could be invoked by everyone. Students were also asked to ensure (through GSI) that the service itself could only be invoked by themselves individually and the lecturing staff. They were requested in particular to undertake performance benchmarking of the search/sort application on a single PC; on a Condor pool; as a grid service on that pool; and to compare the respective speeds of PERMIS RBAC authorisation and GSI-based authorisation, on the service.

Their experiences in developing these services have offered numerous insights into the benefits and pitfalls of the Grid. Table 1 shows the various statistics gathered from the students - arrived at by averaging the results reported. The results were most comprehensive for jobs run on four nodes in the pool, so these are shown for comparison between security infrastructures.

Table 1: The job completion times for the different scenarios.

	Search (s)	Sort (s)
Single Processor	1.7 ± 0.4	5.7 ± 3.3
Condor Pool (16 nodes)	62.2 ± 4.4	60.7 ± 0.1
Condor Pool (4 nodes)	29.5 ± 6.9	35.2 ± 1.8
Grid Service (4 nodes)	31.8 ± 5.9	37.6 ± 11.2
GSI (4 nodes)	39.9 ± 8.6	48.3 ± 15.3
PERMIS (4 nodes)	34.5 ± 8.6	38.5 ± 9.8

The time taken to search and sort the given file typically took, on a single PC, around 2 seconds for the search and 6 seconds for the sort. Distributing the application across a Condor pool required that subsets of the data were distributed and Condor jobs submitted to the various (16 nodes) of the Condor pool. The overheads in distributing the sort/search were significant and typically resulted in taking around 62 seconds to search the file and 60 seconds to sort it, using all the nodes in the pool.

The reasons for this are primarily due to the overheads involved in farming out the jobs across a network. The time taken to split the text files, traverse the local network, prepare the Condor jobs, process them, come back to the original machine and concatenate the final results gave a significant time overhead. A further key factor in the performance is due to the job being completed when *all* distributed Condor jobs have completed, i.e. one queued or delayed job delays the overall time. Other issues that contributed was the high network latency and the non-deterministic nature of benchmarking on a multi-user system. Possible solutions to this include the use of NFS to provide the platform for the Condor pool and also to increase the size of the data sets to be analysed.

The GSI-based authorisation of the application required an increase of around 8-11 seconds to complete the jobs, compared to the unsecured service. The PERMIS based authorisation of the search/sort application took approximately 2-3 seconds more than the unsecured service. The reasons for these increases were due to the time overhead in consulting the grid-map file and the LDAP repository, respectively, then proceeding through the necessary stages of credential validation. These results suggest PERMIS to be more efficient, however the error margins are relatively large so more testing must be undertaken before stronger conclusions can be drawn.

5. Conclusions and Future Plans

Based upon the experiences within the DyVOSE project, both the PERMIS and GSI technologies incur considerable overheads, however in comparison with the overheads incurred in distributed processing via Condor these were not so significant. For Grid security infrastructures such as PERMIS and GSI to be accepted by the wider Grid community, it is clear that performance aspects need to be addressed and developed significantly. This is especially true when real time high throughput Grid applications require fine grained security, e.g. for secure visualisation. The next stage in DyVOSE is to use PERMIS in conjunction with Shibboleth [25] to establish a dynamic mapping of local PMI's to a wider infrastructure involving institutions beyond the local domain. Future uses of PERMIS also include the *MRC*-funded VOTES (Virtual Organisations for Trials and Epidemiological Studies) [24] project, which will explore PERMIS suitability to secure bio-medical data sets as part of conducting clinical trials.

5.1 Acknowledgements

The authors would like to thank collaborators from the PERMIS team including Prof David Chadwick and Dr Sassa Otenko, the other Grid Computing lecturer at Glasgow, Dr Colin Perkins, and also the students that provided the benchmarking data.

6. References

[1] Johnston, W., Mudumbai, S., Thompson, M. Authorization and Attribute Certificates for Widely Distributed Access Control, IEEE 7th Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Stanford, CA, June, 1998, p340-345 (http://www-itg.lbl.gov/security/Akenti/)

[2] L Pearlman, et al., A Community Authorisation Service for Group Collaboration, in Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks. 2002.

[3] Lepro, R., Cardea: Dynamic Access Control in Distributed Systems, NASA Technical Report NAS-03-020, November 2003

[4] Globus Grid Security Infrastructure (GSI), http://www-unix.globus.org/toolkit/docs/3.2/gsi/index.html

[5] D.W.Chadwick, A. Otenko, E.Ball, Role-based Access Control with X.509 Attribute Certificates, IEEE Internet Computing, March-April 2003, pp. 62-69.

[6] D.W.Chadwick, A. Otenko, The PERMIS X.509 Role Based Privilege Management Infrastructure, Future Generation Computer Systems, 936 (2002) 1–13, December 2002. Elsevier Science BV.

[7] Privilege and Role Management Infrastructure Standards Validation project www.permis.org

[8] VOMS Architecture, European Datagrid Authorization Working group, 5 September 2002.

[9] Steven Newhouse, Virtual Organisation Management, The London E-Science centre, http://www.lesc.ic.ac.uk/projects/oscar-g.html

[10] Dynamic Virtual Organisations in e-Science Education project (DyVOSE), www.nesc.ac.uk/hub/projects/dyvose

[11] Globus, http://www.globus.org

[12] R. Housley, T. Polk, Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructures, Wiley Computer Publishing, 2001.

[13] ITU-T Recommendation X.509 (2001) | ISO/IEC 9594-8: 2001, Information technology – Open Systems Interconnection – Public-Key and Attribute Certificate Frameworks.

[14] JISC Authentication, Authorisation and Accounting (AAA) Programme Technologies for Information Environment Security (TIES), http://www.edina.ac.uk/projects/ties/ties_23-9.pdf.

[15] D. Chadwick, O. Otenko, A Comparison of the Akenti and PERMIS Authorization Infrastructures, in Ensuring Security in IT Infrastructures, Proceedings of ITI First International Conference on Information and Communications Technology (ICICT 2003) Cairo University, Ed. Mahmoud T El-Hadidi, p5-26, 2003

[16] A.J. Stell, Grid Security: An Evaluation of Authorisation Infrastructures for Grid Computing, MSc Dissertation, University of Glasgow, 2004.

[17] ITU-T Rec. X.509 (2000) | ISO/IEC 9594-8. The Directory: Authentication Framework.

[18] UK e-Science Certification Authority, http://www.grid-support.ac.uk

[19] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996, Security Frameworks for open systems: Access control framework

[20] V. Welch, F. Siebenlist, D. Chadwick, S. Meder, L. Pearlman, Use of SAML for OGSA Authorization, June 2004, https://forge.gridforum.org/projects/ogsa-authz

[21] OASIS. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) v1.1,. 2 September 2003, http://www.oasis-open.org/committees/security/

[22] OpenSSL to create certificates, http://www.flatmtn.com/computer/Linux-SSLCertificates.html

[23] J. Jokl, J. Basney and M. Humphrey, Experiences using Bridge CAs for Grids, Proceedings of UK Workshop on Grid Security Practice - Oxford, July 2004

[24] Virtual Organisations for Trials and Epidemiological Studies project (VOTES), www.nesc.ac.uk/hub/projects/votes

[25] Shibboleth, http://shibboleth.internet2.edu

[26] OpenLDAP, http://www.openldap.org

[27] A. Otenko, personal communications

[28] PERMIS development pages, http://sec.isi.salford.ac.uk

[29] BRIDGES (Biomedical Research Informatics Delivered by Grid Enabled Services) – http://www.brc.dcs.gla.ac.uk/projects/bridges