Grid Security and Accounting for eScience User Environments

1st Year PhD Report

Shiv Kaushal
High Energy Physics Group
Department of Physics and Astronomy
Victoria University of Manchester
May 2004

1. Introduction 3
What is the Grid? 3
Security 4
Accounting 4
2. GridSite 6
Basic file handling 6
Site Management 8
Group management 9
Delegation of control 10
Other Features 11
3. Security – Authentication, Authorisation and Encryption 12
RSA Encryption Algorithm 12
X.509 Certificates - Authentication 13
GridSite Security and Authorization 16
4. Extensions to GridSite 19
GACL Policy Editor 19
Migration to XACML 22
5. Further Work 25
Continued Security Work 25
Investigation of Accounting Requirements 25
Implementation of Accounting Requirements 25
Acronyms and Glossary 27
Expansion of Acronyms: 27
Glossary 28

1. Introduction

This report details the work I have done over the past seven months (Section 4) and covers the relevant background material (Sections 2 and 3). The title of my PhD is "Grid Security and Accounting for eScience User Environments". The following sections aim to explain the term "Grid" and to introduce the context of my research.

What is the Grid?

The Grid is a method of implementing large scale distributed computing. The term "Grid" can be linked to the analogy with electricity grids: power is supplied via the national grid and may have been generated at any of a number of power stations but the end user is unaffected by the original source and does not need to know. In computing Grids, the resource is not electricity but usually disk space or processor time on large farms across the world but can also be specialised software or hardware. The use of Grids is of importance to the High Energy Physics (HEP) community as they may be the only plausible way of dealing with the large volumes of data storage and data analysis which will be required once the Large Hadron Collider (LHC) is up and running. The LCG (LHC Computing Grid) Project estimates that

> *"12-14 PetaBytes of data will be generated each year, the equivalent of more than 20 million CDs. Analysing this will require the equivalent of 70,000 of today's fastest PC processors."*[1]

However, simply having the hardware capable of dealing with these needs is not enough. No site is large enough to accommodate all of the hardware so there must be a way of combining it into a useful whole while its constituents could potentially be distributed across the globe. This is the aim of the Grid: to supply a framework so that all of this computing capacity is tied together such that users could be unaware of

[1]The LCG Grid Operations Centre, http://goc.grid-support.ac.uk

where their data is stored or where it is being analysed without any detriment to their work.

Security

Possibly the biggest issue concerning potential and even current users of the Grid is the issue of security. With data potentially distributed amongst multiple sites across the world, how can a user's data be kept secure? How can access controls be implemented in a secure way? There are a variety of methods which can be used to determine whether someone is authorized to access a resource, some of which will be discussed in detail later in this report.

Security and authorization are often overlooked by the majority of Grid projects and is sometimes though of as something that "someone else will take care of". It can not be overlooked however, especially if plans for eScience projects to use Grids to store medical information or even complete medical records are to come to fruition. The same applies to the expansion of Grid technologies into non-scientific areas, such as banking, where security is a high priority.

Accounting

Another, perhaps less obvious issue is one of accounting. Accounting is concerned with the allocation of resources to a user, such as disk space or CPU time, taking into account the various organisations that the user is a member of. Accounting is required to ensure that a single user or group does not monopolise the storage capacity or CPU time of a remote site but equally to ensure that they get their "fair share" of resources. It could also be of interest to a site that wishes to sell its resources to other organisations.

These controls could be implemented on a group by group basis but this raises some problems. What is the best way to deal with a user who is a member of multiple groups? How to deal with allocations for "Virtual Organisations" (VOs) - temporary

constructs set up for collaborations? Implementing this on a per-user basis could be a solution, but this defeats the point of using VOs.

Allowing the site hosting the resource to select the credential that gives most/least resources to the user may be a solution, however users may want to be able to explicitly specify which group membership to favour. All of these factors need to be considered in order to produce a system acceptable to all.

2. GridSite

The majority of the work I have done is based around GridSite, a set of tools produced by Dr. Andrew McNab, Department of Physics and Astronomy, The Victoria University of Manchester. This section presents the “main features” of GridSite.

GridSite is an extension to the Apache web server, which combines Grid technology with existing web based services. In its simplest form it can be used as a simple file server that implements access control to files using Grid credentials. It can also be used to provide a greatly enhanced website, allowing many functions for advanced management of the site. It is the latter use that will be described in more detail. It should be noted that the features described below are not accessible to any user who visits the site but are implemented securely using access control methods described in the Section 4.

Basic file handling

GridSite allows authorized users to edit pages “on the fly” (Figure 1): while visiting a page, users can correct mistakes or add further information to a document from within their web browser. The ability to rename or delete files is also presented through a simple interface (Figure 2).

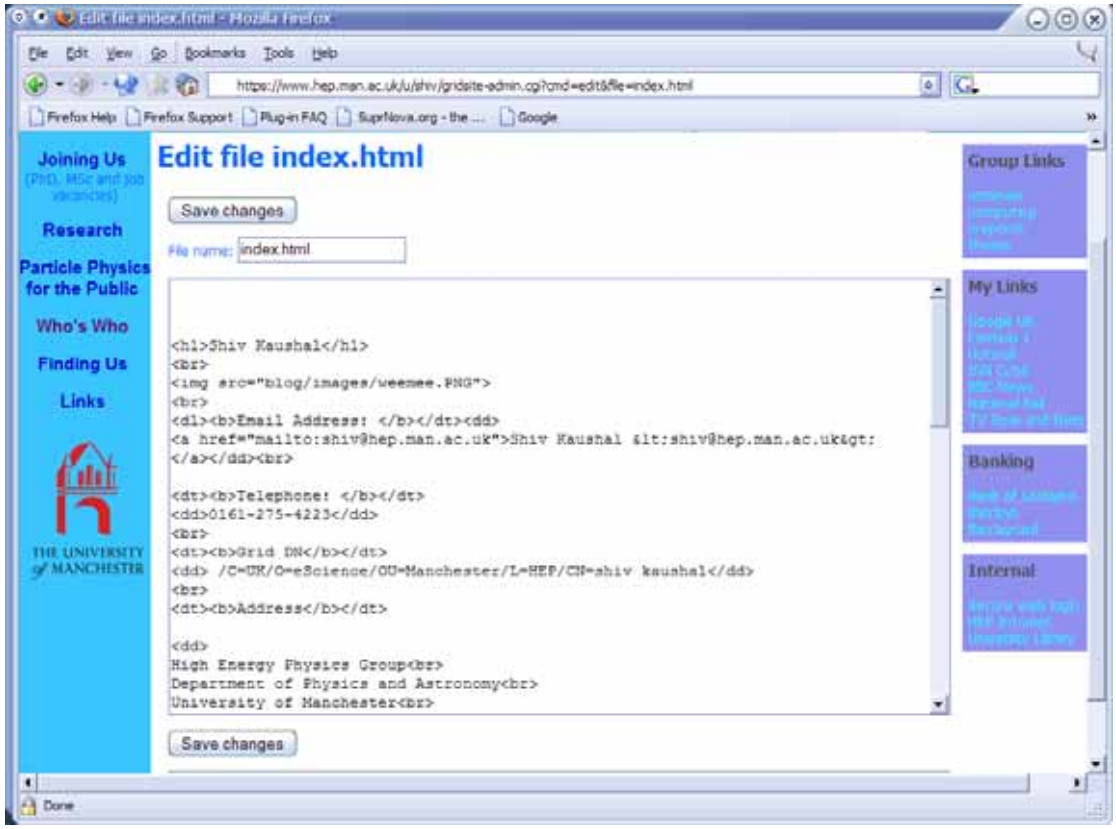

Figure 1: GridSite allows editing of HTML and other documents with a web browser.

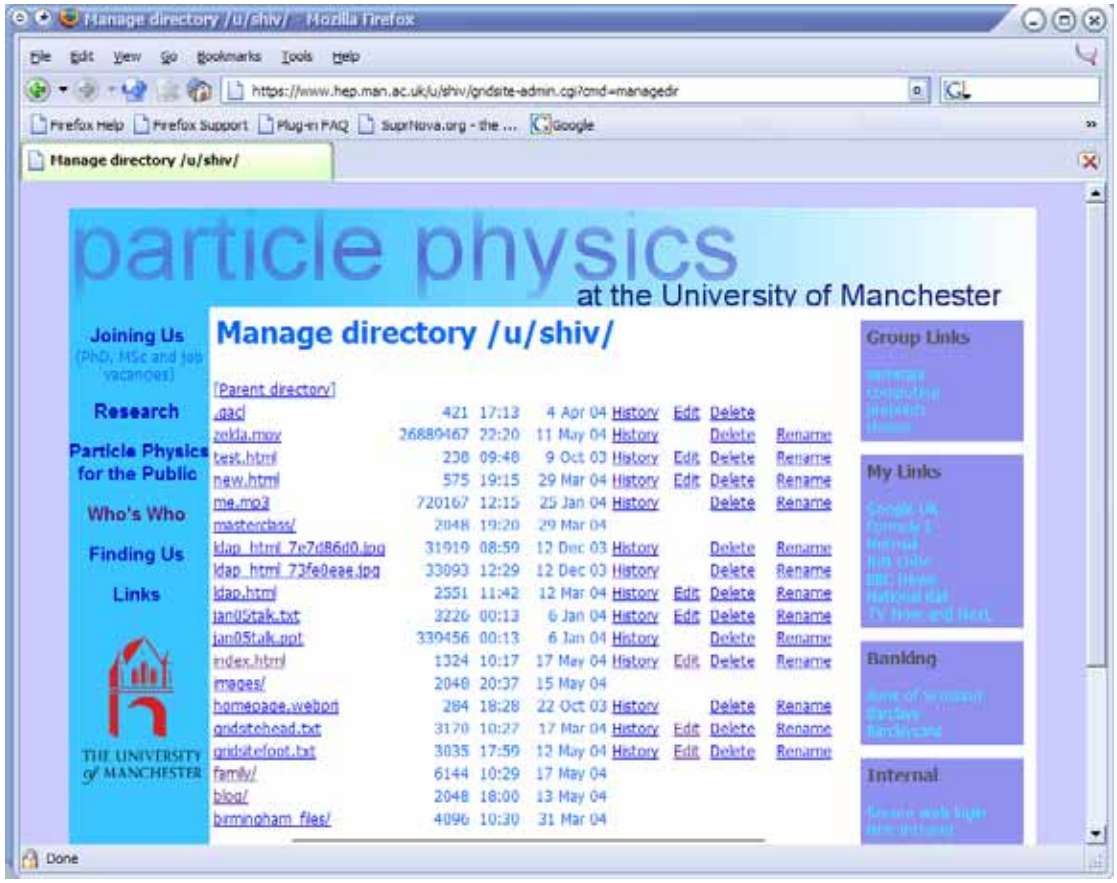

Figure 2: GridSite offers various file management options.

It also allows users to view a file's history. This keeps track of all of the changes made to a file, which user made the changes and allows users to view previous versions of the file (Figure 3).

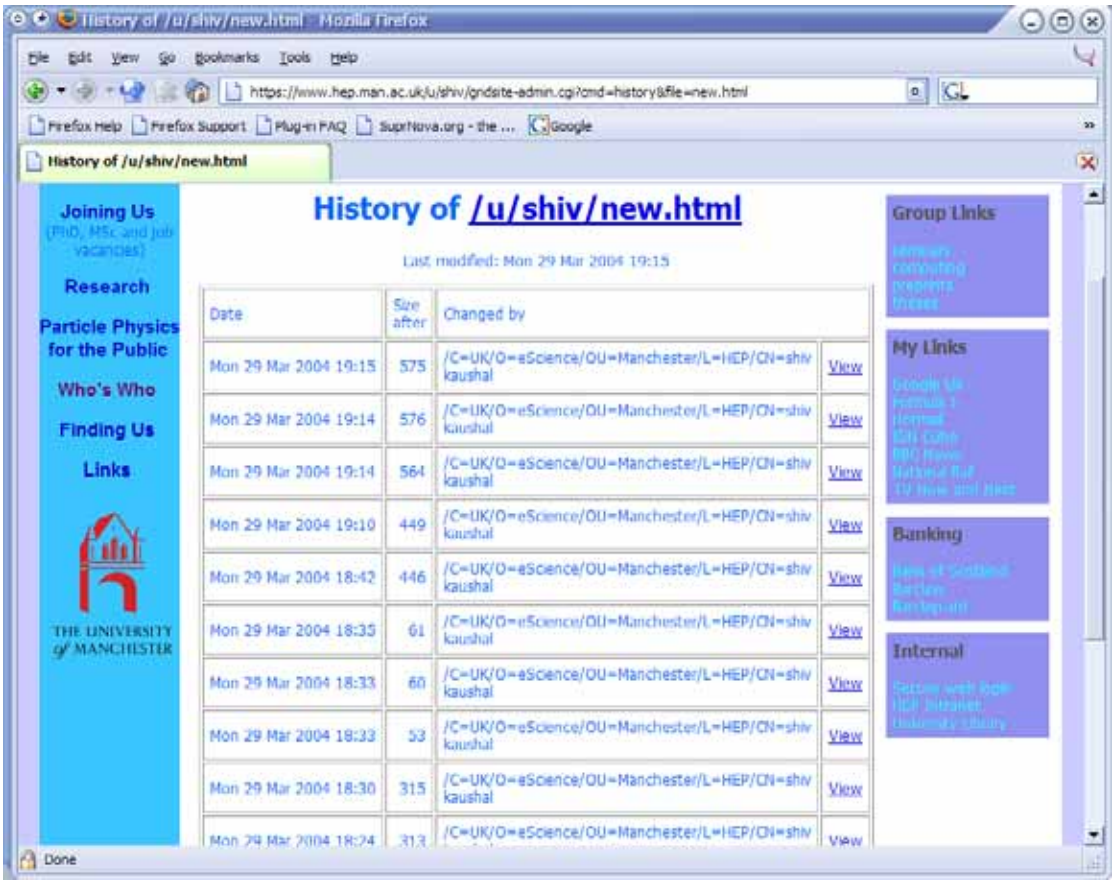

Figure 3: The history function keeps track of changes made to files and allows viewing of previous versions.

Site Management

GridSite also offers more advanced management features (Figure 4), such as uploading files and creation of new files/directories. Entire directory structures can also be uploaded in a single "zipped" file and then expanded with a single click. This allows almost total site management from within a web browser – there is no need to log in to the web server directly.

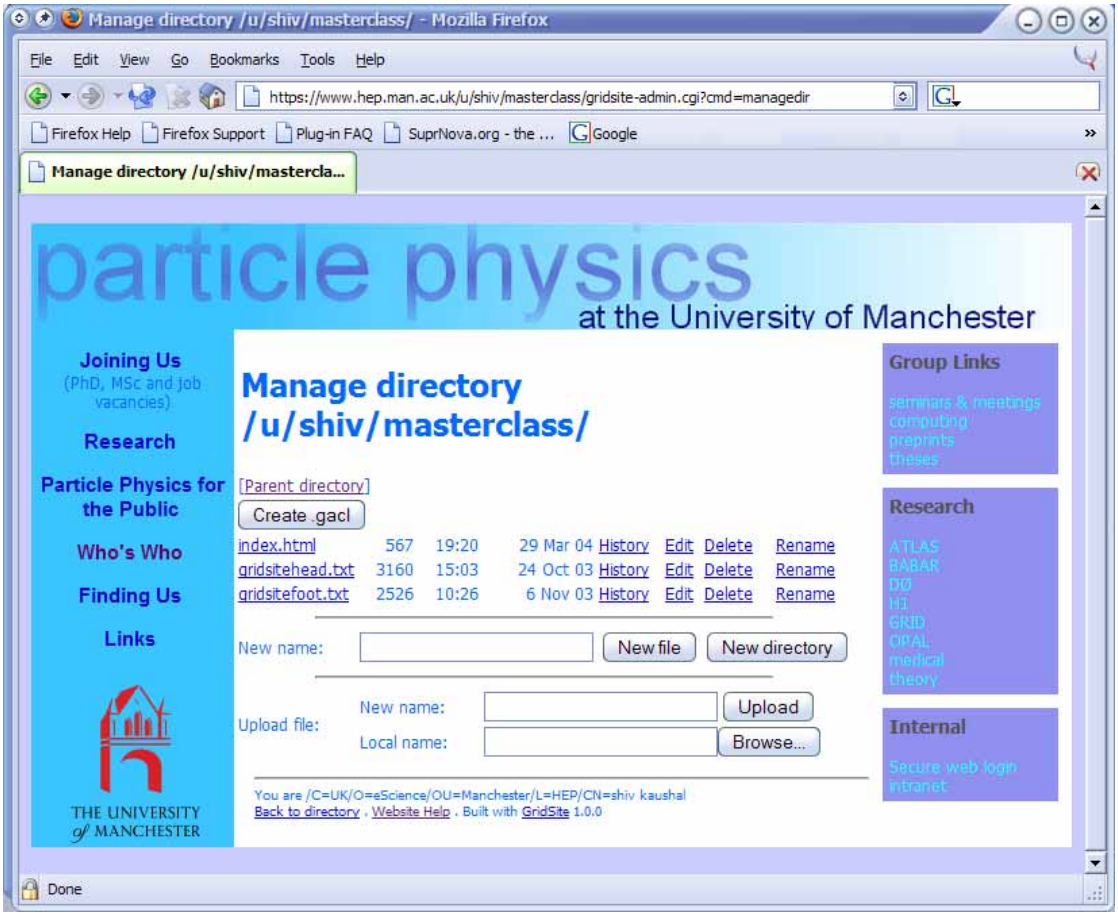

Figure 4: GridSite presents a simple interface for creating new files and directories.

Group management

GridSite also has the ability to define and manage simple VOs. It can be used to define groups of users, which can be used not only by the local site but by any other website or Grid application. A group is defined by a list of unique identifiers called DNs (Distinguished Names - explained in more detail in section 3). These groups can then be used to control access to areas of the website and which features are available to users.

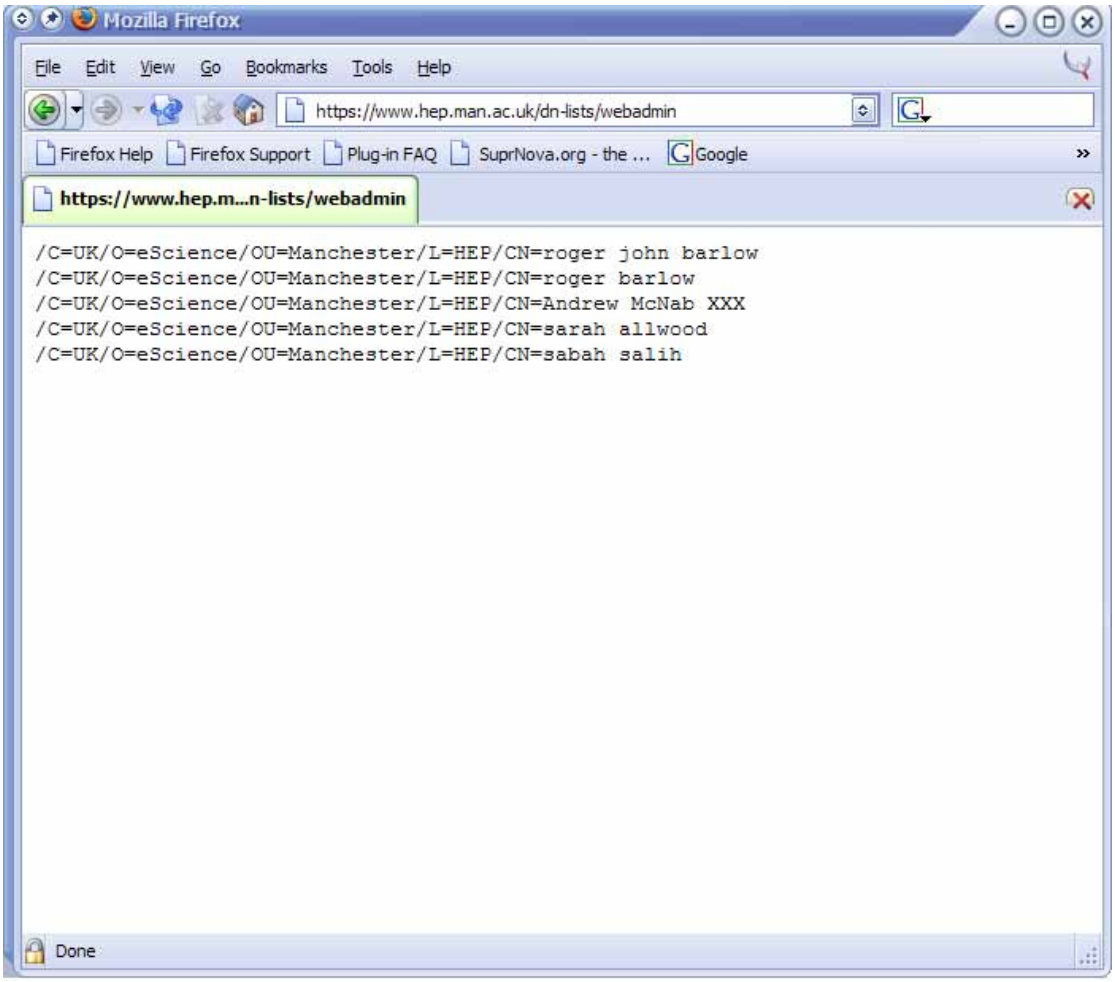

Figure 5: This example of a VO/DN-list is the Manchester HEP website admin group

Delegation of control

One immediate advantage of using DN-lists is that the site can be easily configured to have different sections of the website administered by different groups of people. This makes the overall management of a website much simpler for the webmaster. Users can have complete control over their own area without need for administrator access or even a standard account on the web server without any risk of unauthorized access to other areas of the site. Again, all of these functions can be performed through a web browser, giving users and administrators great flexibility.

Each of the above features can be useful individually but the combination of them into one package gives a powerful set of tools for site management. It is important

however to ensure that only the intended users have access to these features. GridSite uses Grid security features to authenticate users and decide which users can access which features. This is covered in more detail in the Section 4.

Other Features

While all of the previously mentioned features are useful, a user will not always want to simply have access to their data through a user friendly management interface. Usually the aim is to perform some kind of analysis on the data of interest. GridSite provides a number of useful features to aid in this process.

Firstly, HTTP(S) (the standard protocol/secure protocol used for web pages) is a good protocol for data analysis batch jobs as often only parts of the data files are needed for analysis. The HTTP(S) protocol allows software to specify which parts of a file are to be retrieved, thus reducing network traffic to a minimum and increasing the speed of the batch job. Also provided are a set of command line utilities, *htcp* and *htls*, which imitate the functions of the standard Unix/Linux *cp* and *ls* commands, but operate on website directories much in the way that the *scp* (secure copy) command works on remote PCs. These tools will also work on non GridSite enabled web servers (if directory listings are enabled), so providing the option to use a wide range of web resources in batch jobs. The tools will also obey the access control policies implemented on a GridSite enabled server.

3. Security – Authentication, Authorisation and Encryption

The security methods used in GridSite are based on the HTTPS protocol and X.509 certificates (Grid certificates). These in turn allow the secure transfer of information between user and site and the enforcement of access controls. These methods are explained in more detail below.

RSA Encryption Algorithm

The HTTPS protocol is one of the most widely distributed secure protocols, used by most (if not all) websites that allow online purchases. HTTPS is based upon the RSA encryption algorithm, developed by and named after Rivest, Shamir and Adleman in the 1970s, a form of public key cryptography. This is also the encryption method used in standard *ssh* package. The mathematics of the algorithm is shown below[1].

- Choose two large (e.g. 1024-bit) prime numbers, *P* and *Q*,
- Choose *E* such that, $E > 1$, $E < PQ$, and *E* and $(P-1)(Q-1)$ are relatively prime (i.e. they have no prime factors in common). Note: *E* does not have to be prime, but it must be odd and $(P-1)(Q-1)$ can not be prime as it is even.
- Compute *D* such that $(DE - 1)$ is evenly divisible by $(P-1)(Q-1)$, this can be written

$$DE = 1(\mathrm{mod}(P-1)(Q-1)) \tag{3.1}$$

- The encryption function is

$$C = (T^E)\,\mathrm{mod}(PQ), \tag{3.2}$$

[1] Eric W. Weisstein. "RSA Encryption." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/RSAEncryption.html

where T is the plaintext (the message to be encrypted converted to a positive integer) and C is the ciphertext (the encrypted message in the form of a positive integer). The message being encrypted, T, must be less than the modulus, PQ.

- The decryption function is

$$T = (C^D) \bmod (PQ), \tag{3.3}$$

where all variables are as defined in (3.2) .

- The public key is the pair of values (PQ, E) – used for encryption. The private key is the number D – needed to decrypt the message.

Using functions (3.2) and (3.3) secure communication is possible between two points. At first glance this is not immediately obvious, but there currently exists no method for decrypting a message, C, without the private key, D, other than using brute force and trying all possible values of D (of which there are an enormous number of possibilities). A user can publish their public key in some way and (as long as their private key is kept secret) only they will be able to decrypt messages which are created using their public key. If two users (or a user and a website) send each other their public keys they can then communicate completely securely.

X.509 Certificates - Authentication

Authentication is the process of identifying yourself to a remote site or user. Using RSA encryption alone is not enough to achieve this. It allows you to verify that you are always "talking to the same person", provided that their private key is not compromised, but does not offer a way to prove that a public key belongs to a specific person or server. In the cases of *ssh* clients and e-commerce sites this is usually accomplished by using usernames and passwords to verify the identity of the user at the time that the secure connection is created. Another solution to this problem is to

use a trusted third party, a Certificate Authority (CA), to link an identity to a public key – this is done using certificates, known as X.509 certificates or Grid certificates

Grid certificates are based on your public key and are "signed" by the CA. This still leaves the issue of being able to trust the CA, but this is rarely an issue as there is usually a well defined method of obtaining the CA's public key and their private key is often stored on an unusually highly secure machine. Along with simply signing a user's public key, the CA will also attach a unique identifier, a Distinguished Name (DN), for the user. This should be unique across all CAs so usually contains a reference to the issuing CA as well as the user's organisation and their name. The UK eSicence CA uses DNs of the following form:

/C=UK/O=eScience/OU=Manchester/L=HEP/CN=shiv kaushal

In order to receive a certificate from a CA there is usually an identification process that users are required to go through. The UK eScience CA, for example, requires photographic identification to be provided to the user's local administrator before a certificate will be issued.

Since certificates can be issued to sites as well as users, certificates can identify users to a site but also provides a method for users to verify the identity a website (Figure 6). This method is well established and used by e-commerce sites as well as sites offering software downloads – this allows users to verify that files have originated from a trusted source and should, for example, not contain any malicious code.

As with the private key, a user's certificate must be kept secure and is usually password protected. In the event that a user's certificate is compromised there is a procedure to combat any possible consequences. Most CAs publish a list of certificates that are no longer valid, known as a Certificate Revocation List (CRL). Currently it is each site's responsibility to keep abreast of any changes to the CRL. After the CA is informed that a particular certificate is no longer secure, that certificate would become invalid at all sites which would normally accept it as soon as the updated CRL is retrieved by those sites. There are currently proposals for a system where the validity of a certificate would be checked in real time with the CA every

time a secure connection is established. If this is implemented then the compromised certificate would become invalid almost instantaneously. This kind of rapid response would be much more difficult to achieve without using CAs as a central third party.

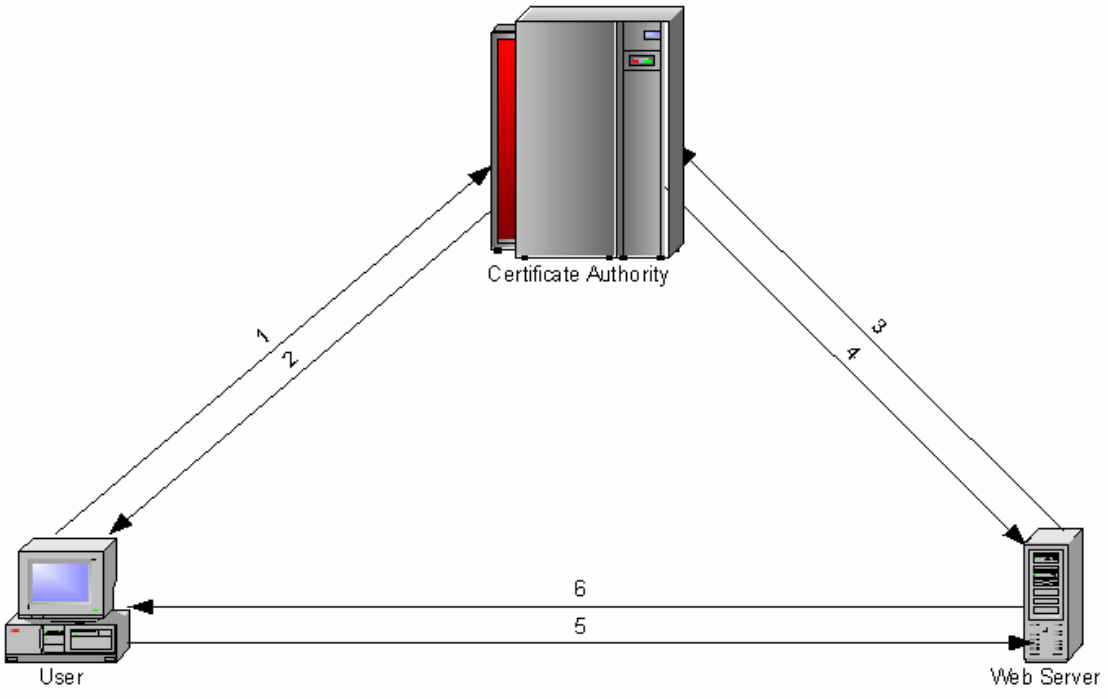

Figure 6: Before secure communications can take place between server and user, the user must first send identification to the CA (1), and then receive the CA's public key and a Grid certificate (2). The web server must go through a similar process (3 & 4). Only after this can the user prove their identity to the web server (5) and verify the identity of the web server (6).

Most modern web browsers allow X.509 certificates to be loaded into them for site authentication purposes. GridSite takes advantage of this for its all of its web based features. The X.509 standard is an extensible specification so that other attributes, such as VO membership information, can also be attached to the certificate and verified. The main attribute used by GridSite is the DN but, as its security is based on X.509 certificates, access control can be implemented based on a number of attributes. Another advantage of using X.509 certificated for authorisation purposes is that any Grid application or GridSite server can define what resources a user should have access to simply by using their DN. There is no need for local user accounts on different sites and no need to remember multiple username and password combinations – one certificate does it all.

Another common practice, usually used in batch jobs, is the use of Grid proxies. This is a temporary certificate, with a limited lifetime, which can be used without having to enter the protecting password. This is essential if users want to run non-interactive batch jobs that require access to Grid resources, such as files on a GridSite file server. GridSite accepts these proxies and treats them as it would a standard Grid certificate making the command line tools mentioned on page 11 suitable for non-interactive batch jobs. Using these tools GridSite enables a server to act both as an enhanced website and as a secure file server.

GridSite Security and Authorization

Authorization is the process of deciding which users are allowed access to a particular resource this occurs after the authentication stage, so users can be described purely by their Grid certificate DN. GridSite defines access control using an XML based access control language called GACL (Grid Access Control Language). GACL was defined by Dr. Andrew McNab, Victoria University of Manchester, when there was need for such a language for projects within the European Data Grid (EDG) but no widely accepted standard existed. An example of an entry in a GACL Access Control List (ACL) is shown in Figure 7, below.

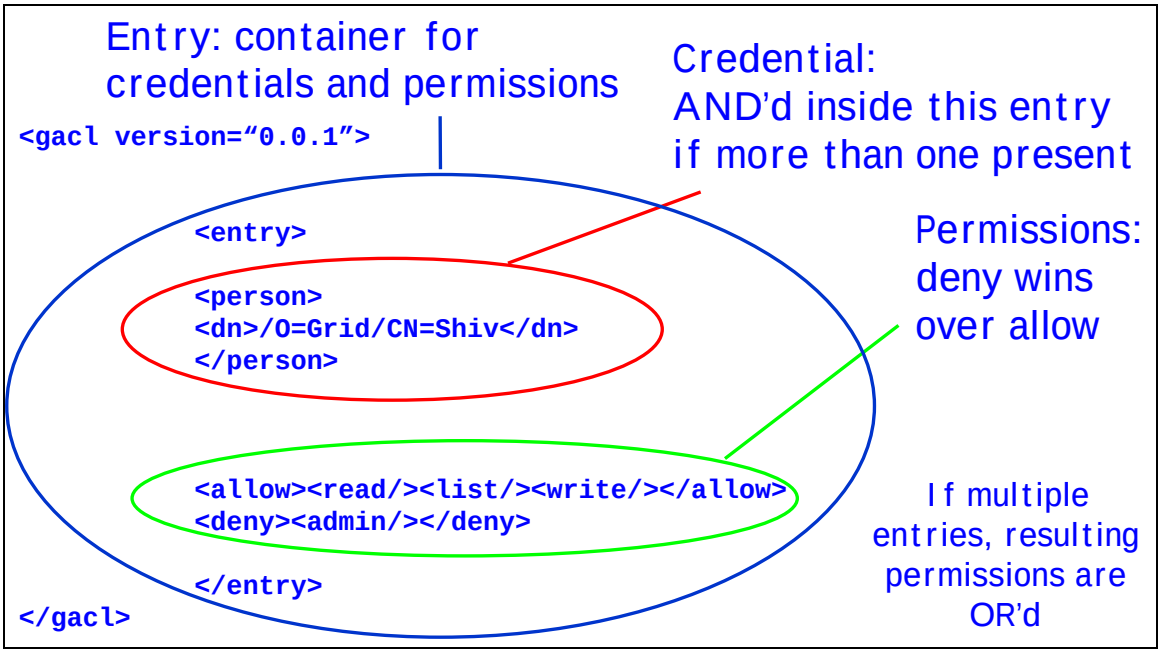

Figure 7: A GACL access control list is a list of entries: each entry contains credentials and associated permissions. The above access control list simply states that the user whose Grid certificate DN is "/O=Grid/CN=Shiv" is allows read, list and write access permissions but denied admin permission.

A GACL ACL is simply a list of entries. Each entry consists of a list of one or more credentials and a set of permissions. If there are multiple credentials within an entry then they are AND'd (i.e. a user must have both credentials in order for this entry to apply to them). Types of credentials that GACL can use to define access control include the following:

- DN, obtained from the users Grid certificate.
- DN-lists, in the form of a URL to the location of the DN-list e.g. the Manchester HEP web administrator group shown in Figure 5.
- DNS hostnames, (with wildcard matching), either in the form of a hostname (*.hep.man.ac.uk) or an IP address (194.36.2.*).
- VOMS attributes – group membership information relating to the Virtual Organisation Management System defined by work done for the EDG Project.
- Any User – a generic credential which any person trying to access the site will have.

Permissions available are read, execute, list, write and admin. The admin permission defines if a user is permitted to edit the ACL. If the ACL does not define whether a particular permission is allowed or denied, it is assumed that it is denied. If a permission is both denied and allowed by the same entry then the permission will always be denied. If multiple entries apply to a user the resulting permissions are OR'd. All of these features should be kept in mind when writing ACLs as unexpected results can occur e.g. if the "any user" credential is explicitly denied write access, then **all** users will be denied write access even if a subsequent entry allows write access.

The ACLs are usually defined on a per-directory basis and stored in the directory to which they apply. If no ACL exists then GridSite will look in the parent directory for an ACL and continue to do so up the directory tree and will stop when an ACL is found.

4. Extensions to GridSite

This section details the work I have done in extending and improving the security functions in GridSite. Learning how GridSite works was the first step towards adding extra functionality. This involved my gaining an understanding of the material covered in the previous sections, an in depth examination of the source code and testing all of the functionality.

Having gained an understanding of GridSite's internals, I was able to work on extending its functionality. In order to contribute new features to the GridSite source code, the changes had to be committed to the publicly readable source code repository using the Concurrent Versioning System (CVS). CVS is an open source code management tool designed to allow multiple users to work on the same files at the same time using a shared directory.

CVS was used so that any changes I made were combined with those made by Andrew McNab, who was working on the development of GridSite simultaneously. It also allowed the very latest versions of GridSite to be downloaded and tested by users who preferred the "bleeding edge" version of the software or wanted to use a new feature not yet in an official release. These users could then provide feedback on the changes made in the latest version. I was also involved in fixes for bugs found by users after initial release in December 2003. The changes I have made have been deployed on multiple sites; the software is being used by real users and is functioning as intended.

GACL Policy Editor

As a site gains more and more users, the ACLs can become increasingly complex. This will inevitably lead to errors when creating and editing ACLs, which can be difficult to locate. In order to combat this problem, I have designed and implemented a web based editor which allows users to edit the ACLs through GridSite without having to resort to editing plain text.

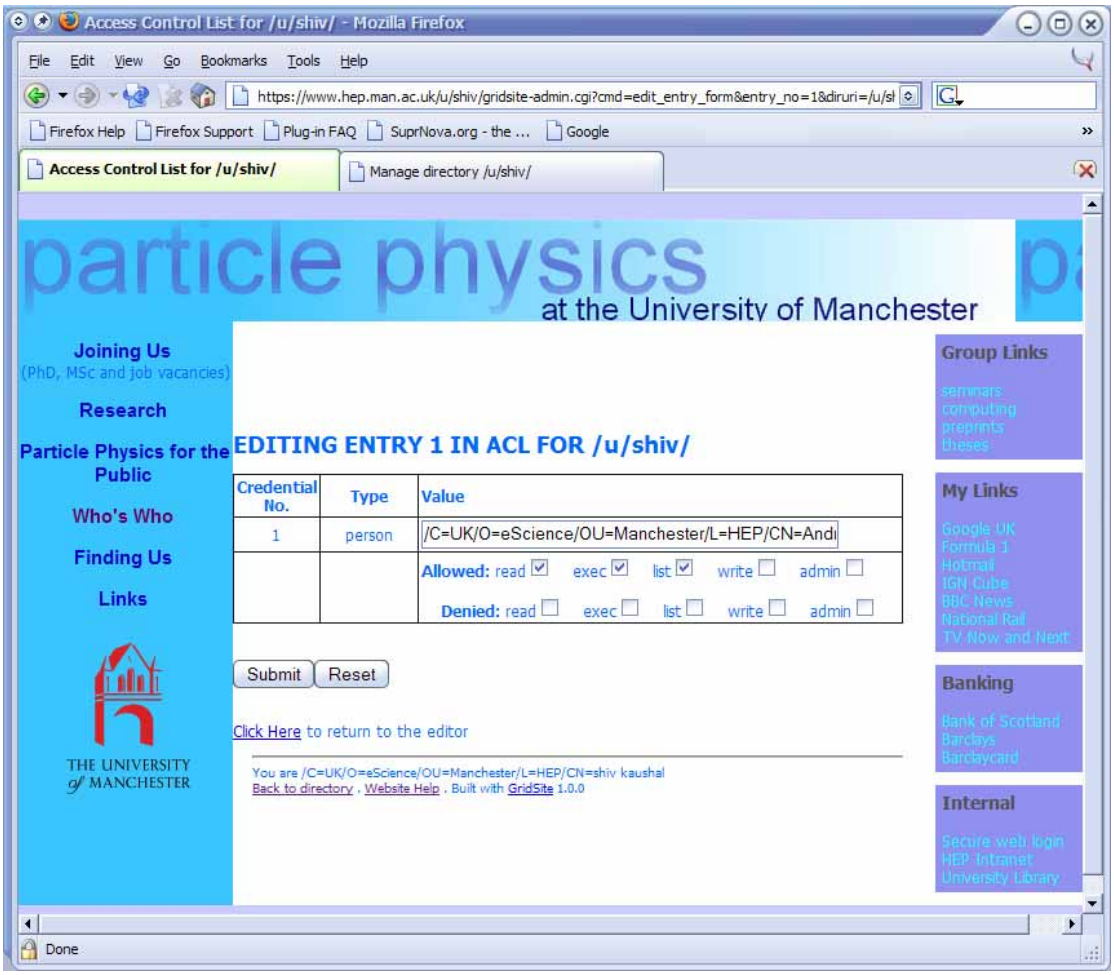

Figure 8: Editing entries in GACL ACLs is greatly simplified by the GACL policy editor

When designing the policy editor, as well as examining existing interfaces for setting file/directory permissions I conferred with potential users from within the Manchester HEP group and eventually settled on the design shown in Figure 8 and Figure 9. As the design uses only basic HTML features it is easily displayed within text based web browsers such as *lynx*, essentially allowing the policy editor (as well as the other features of GridSite) to be used form the command line.

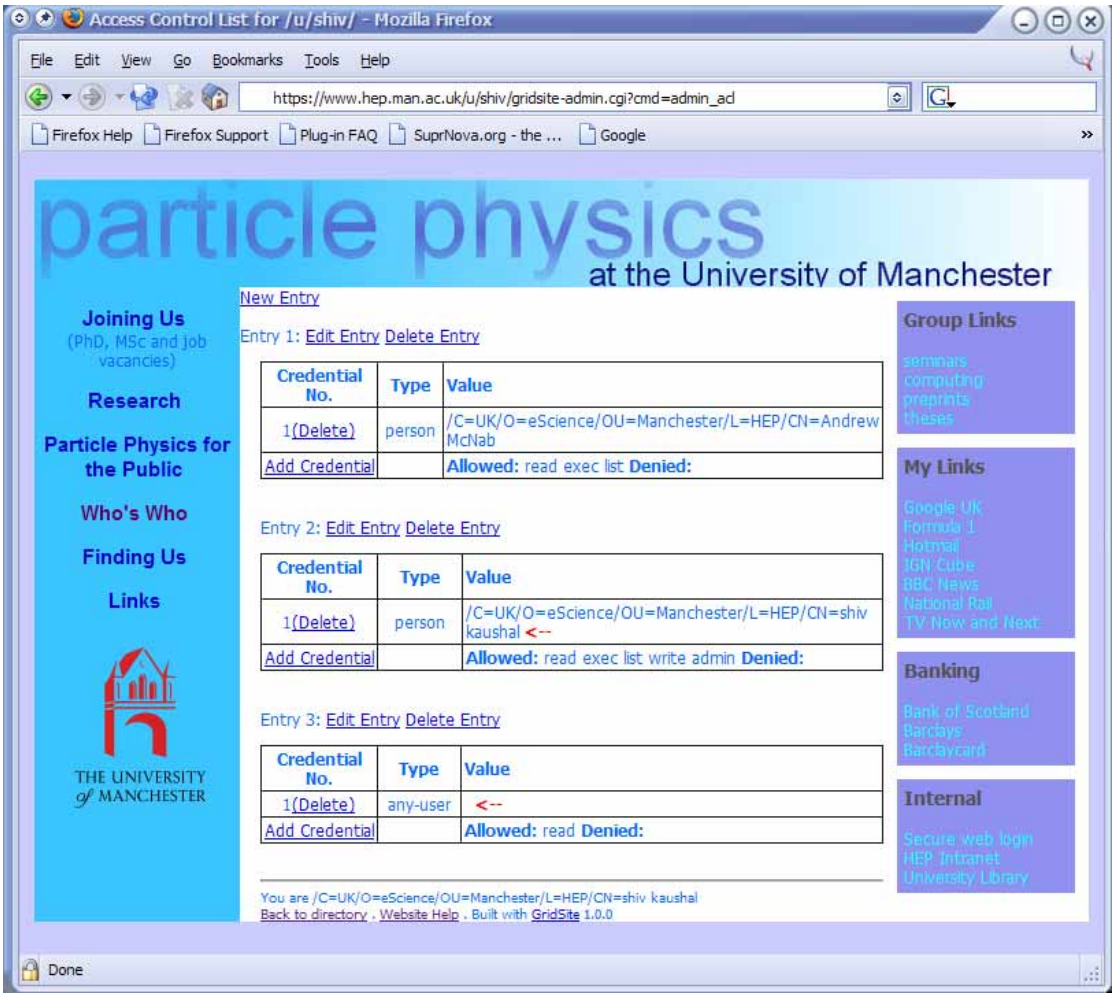

Figure 9: The GACL policy editor breaks up entries into separate "sections" and marks any entries affecting the current user with a red arrow (<- -).

While the GACL policy editor ensures that the formatting of the ACL is correct it also provides various other features to help a user who is editing ACLs. It will prevent a user from performing certain actions (e.g. denying admin rights to themselves) and highlights any entries that apply to them (Figure 9). It also warns if the ACL has been changed by some other process and warns a user if their permissions have changed by their actions.

After the initial implementation was completed and functioning I added a history function, similar to that implanted for standard files which, in addition to showing previous versions of the ACL, also lets a user revert to that version. This is useful if editing an ACL causes unexpected results. The user can revert to a previous (working) version of the ACL very quickly and then try to locate the problem.

Migration to XACML

XACML (eXtensible Access Control Markup Language) is a generic access control language standard defined by OASIS (Organisation for the Advancement of Structured Information Standards) that is gaining acceptance in the Grid community. Users will eventually want to use this standard either exclusively or in parallel with GACL in order to be compliant with the other Grid applications. Due to the generic nature of XACML it is inevitably more complex than GACL, as shown in the comparison on the following page (Figure 10). This increased complexity makes need for easy editing even greater. Currently the only open source implementation is by Sun Microsystems, written in Java. I investigated the features of Sun's XACML implementation in order to determine whether it would be useful to extend GridSite to use the features it provides. There were a number of points to consider:

- Java is generally considered to be slower than complied C/C++ code so is not ideal for the dynamic content nature of GridSite.
- Would need to build a "bridge" between Sun's Java implementation and GridSite which would need to be updated to match any changes to Sun's package.
- GridSite would no longer be a standalone application: currently GridSite has no dependencies on other packages (other than Apache – packaged with most Linux distributions), changing this would make installation more complex and reduce its appeal to potential users.
- It would be simpler to create partial implementation within GridSite to create a link between GACL and XACML.

I have modified the GridSite functionality so that it now has a partial XACML implementation in C. Using these extensions GridSite can now output XACML policy files using the existing GACL based policy editor.

XACML

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy  xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy cs-xacml-schema-policy-01.xsd"
  PolicyId="GridSitePolicy"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/path/to/dir</AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>

  <Rule RuleId="Entry1A" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/C=UK/CN=shiv</AttributeValue>
            <SubjectAttributeDesignator
              AttributeId="person"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">list</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
  <Rule RuleId="Entry1D" Effect="Deny">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/C=UK/CN=shiv</AttributeValue>
            <SubjectAttributeDesignator
              AttributeId="person"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">exec</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
```

GACL

```
<?xml version="1.0"?>
<gacl version="0.0.1">
<entry>
<person>
<dn>/C=UK/CN=shiv</dn>
</person>
<allow><read/><list/></allow>
<deny><exec/><write/></deny>
</entry>
</gacl>
```

Figure 10: Comparison of XACML and GACL: The two polices shown above express identical access control rules, the only extra information in the XACML policy is the directory for which the policy applies.

I am currently working on modifying GridSite to read in XACML policy files. This is more difficult than outputting policies since the nature of XACML allows a set of rules to be expressed in a number of different ways. The aim is to keep functionality for both GACL and XACML and to allow site administrators to choose which access control language to use. This should make migration to XACML relatively simple for current users of GridSite.

Another reason that XACML is an area of interest is that it can be used to define usage control policies for accounting purposes as well as access control policies. There are built-in data types defined within the XACML standard which could be of use but it is likely that the extensible features of XACML will allow suitable data types to be defined as they are needed.

5. Further Work

Continued Security Work

As stated in the previous section, the XACML implementation is not yet fully functional. Further work is required to allow XACML to become the default access control language used in GridSite, so currently GACL is still the default. Support must also be provided for GridSite following its 1.0 (non beta) release and any bugs found by users must be corrected. GridSite is being used for various sites including the Manchester University's HEP Group, the Manchester eSecurity centre and the LCG Grid Operations Centre (responsible for coordinating the overall operation of the LHC Computing Grid).

Investigation of Accounting Requirements

As stated in Section 1, there are many possible requirements and solutions to the problem of implementing accounting for Grid environments. It is likely that the HEP and eSciecne communities will have a specific set of requirements in this area. It may be useful to collaborate with the Site Authentication, Authorization, and Accounting Requirements (SAAAR) Research Group of the Global Grid Forum (GGF) as well as the newly formed Particle and Nuclear Physics Applications (PNPA) Research Group in order to try and discover what these requirements are. I attended the GGF 10 meeting in March 2004 and hope to attend further meetings in order gain as much input from these groups as possible.

Implementation of Accounting Requirements

Having found what the requirements of the relevant communities are with regards to accounting it may be possible to produce an implementation of the specifications. This may involve extending the GridSite policy editor or even the creation of a new

editor. It may be necessary to produce separate software which can link XACML policy files to local file systems and their access control mechanisms. If GridSite were to be place “on top” of such a system then the usage control (for disk space) would come about “for free” through the GridSite interface. This still leaves open the issue of how CPU time quotas can be implemented and the opportunity for even further research.

Acronyms and Glossary

Expansion of Acronyms:

ACL - Access Control List
CA - Certificate Authority
CPU - Central Processing Unit
CRL - Certificate Revocation List
CVS - Concurrent Versioning System
DN - Distinguished Name
EDG - European Data Grid
GACL - Grid Access Control List
HEP - High Energy Physics
HTML - Hyper Text Markup Language
HTTP - Hyper Text Transfer Protocol
LCG - LHC Computing Grid
LHC - Large Hadron Collider
OASIS - Organisation for the Advancement of Structured Information Standards
SSH - Secure Shell or Secure Socket Shell
SSL - Secure Socket Layer
VO - Virtual Organization
VOMS - Virtual Organization Management System
XACML - eXtensible Access Control Markup Language
XML - eXtensible Markup Language

Glossary

Access Control List (ACL)
: A list of rules in a particular expression language which govern whether or not requests for access to a resource will be approved. Also called Security Policy.

Accounting
: The process of monitoring the use of resources to a user, and enforcing limitations on allocation and usage.

Apache
: A commonly used web server software package. GridSite is based upon Apache.

Attribute
: A named property (type and value) associated with an entity. The most commonly used attributes are those associated with subjects, e.g., roles, group membership. Attributes can also be associated with resources, such as the clearance level required to access the resource.

Attribute Certificate
: A structured document containing attributes used for authorization which is digitally signed using public key cryptography. See X.509 Certificate.

Authentication
: The act of establishing the identity of a subject or user based on a credential that the user presents.

Authorization
: Either the act of authorizing a subject to access a resource, the issuing of a token that proves such a right, or the token itself.

Certificate Authority (CA)
: A trusted entity which signs X.509 public key certificates upon request that bind a public key to a distinguished name. Possession of an X.509 public key certificate signed by a trusted CA is a start to establishing trust between two parties. See X.509 certificate.

Certificate Revocation List (CRL)
: A list maintained by a Certificate Authority which of X.509 certificates which have been revoked and are not longer valid.

Concurrent Versioning System
: A code management system which provides the ability to track (and potentially revert) incremental changes to files stored in a central repository and can be used concurrently by many developers.

Credential
: Those pieces of information necessary for some entity to authenticate as a given identity. This usually includes an identifier (e.g. a username) and some secret (e.g. a password or private key). The authentication process typically involves the proof of possession of the secret component without the need to make the secret component available to the other party.

Distinguished Name (DN)
: A unique identifier associated with a user, usually part of an X.509 Certificate.

DN-list
: A list of Distinguished Names, used to define a group of people.

Extensible Access Control Markup Language (XACML)
: An XML based access control expression language produced by OASIS.

Extensible Markup Language (XML)
: A text based markup language for interchange of structured data.

Grid Certificate
: See X.509 Certificate

Grid Access Control Language (GACL)
: An XML based access control expression language produced by Andrew McNab (University of Manchester)

GridSite
: An extension to the Apache web server, which combines Grid security with existing web based services.

Hypertext Markup Language (HTML)
: A text based markup language used to produce and control the appearance of web pages.

Hypertext Transfer Protocol (HTTP)
: A protocol used to transfer data over the internet.

Policy
: A policy is generally a set of rules that describe how a system should behave. Policy in terms of security is typically limited to information and rules about resource access. See Access Control List

Private Key
: The secret key in a Public Key Cryptography system, used to decrypt incoming messages and sign outgoing ones.

Public Key
: The publicly available key in a Public Key Cryptography system, used to encrypt messages bound for its owner and to decrypt signatures made by its owner.

Public Key Cryptography
: A coding system in which encryption and decryption are done with public and private keys, allowing subjects who do not know each other to send secure or verifiable messages.

Resource
: A component of a system that provides or hosts services and may enforce access to these services based on a set of rules and policies defined by entities that are authoritative for the particular resource.

RSA Encryption
: A popular encryption and authentication standard that uses Public Key Cryptography. Developed by Rivest, Sharmir, and Adelman.

Secure Socket Layer (SSL)
: A protocol that transmits communications over the internet in an encrypted form.

Subject
: See User.

User
: The entity seeking authorization to use a resource or a service.

Virtual Organization (VO)
: A set of users, resources and services from different home organizations that have set up common authorities and operational rules in order to share the resources for a common purpose.

X.509
: ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997. Defines a structured name for users and services called a distinguished name that defines a unique name for a person by combining a common name with organizational and other components.

X.509 Certificate
: A certificate that binds a X.509 distinguished name with a public key. When presented via a protocol that confirms that the presenter knows the private key associated with the public key in the certificate, can be used to authenticate the presenter of the certificate.