



ShibboLEAP Project

Final Report:

Birkbeck, University of London

Ken Brown

May 2006

Overview of Shibboleth implementation at Birkbeck College

This report includes background information intended for an audience inside and possibly outside Birkbeck, and also enough detail to enable people to reproduce what was done to install our pilot Shibboleth and OpenLDAP servers at Birkbeck.

Contents

Background to Shibboleth	2
How Shibboleth works Potential importance and benefits of Shibboleth ShibboLEAP	
Reasons for Birkbeck to participate	4
Shibboleth Implementation decisions at Birkbeck	5
Decisions on directory Decisions on content and organisation of LDAP database Decisions on platform Decisions on the provision of attributes to the authentication database	
Existing directories at Birkbeck	13
Implementation experience at Birkbeck	16
Documentation of existing CCS directories Installation of SuSE Linux 9.2 with Apache 2, OpenLDAP et.c etc. Configuration of OpenLDAP with EduServe and Posix schemas Population of LDAP database with modified copy of Birkbeck live directory OpenSSL Firewall changes Globalsign certificate Configuration of Apache, Tomcat, etc Install Shibboleth id server software. Further Configuration of Apache & Tomcat The current situation	
Appendix 1, Software installed	20
Appendix 2, Populating OpenLDAP directory	26
Appendix 3, glossary	28

Background to Shibboleth

Not so long ago each online resource, such as a database or a journal archive or a library catalogue, had its own authentication system and each user was issued with a different username and password for every application they needed to use. This obviously causes problems for end users who have to remember many passwords, and for support staff who have to help them when they can't.

Many organisations (including corporations and employers, universities, and service providers) have attempted to set up single sign-on environments (SSO) so that members or customers have one set of credentials issued by the organisation for access to all resources owned by or subscribed to by the organisation. For example, within Birkbeck the Central Computing Services (CCS) user name and password provides most (but not quite all) members of the college with access to most (but not quite all) of the online resources they need. This includes access to resources outside the Colleges – for example JSTOR or Science Citation Index, access to which are controlled through the Athens system.

Web applications are continually becoming more complex and varied. As well as resources provided centrally by commercial or academic service providers, educational and research institutions are increasingly using ICT to co-operate across organisational boundaries and to set up virtual organisations and virtual computing resources (for example in Grid computing). Such collaborations often include commercial organisations who need to protect their intellectual property, or (particularly in fields such as medical research) involve personal data which must be protected for legal and ethical reasons. The number of online roles assumed by individuals and organisations continues to increase.

As there is no real possibility of a single worldwide authentication and access system (and these applications are now worldwide, not national) the only realistic way to provide simple access to digital resources for large and diverse communities is for the resource providers to trust such organisational SSOs as identity providers for those users who are members of that organisation.

The use of attributes from an authoritative source within an information provider (IdP) makes it possible to make resources available to a subset of an organisation, for instance to research staff alone, or to members of some but not all departments, or to students on particular courses, or to members of a group of collaborators. To allow this it is vital to have a common set of attributes provided by all IdP and used by all services, which allows easily scaleable systems to be built. If each service provider (SP) requires tailored attributes or each IdP returns locally designed ones, then we quickly get a many-to-many combinatorial explosion and the system is utterly unmanageable.

Shibboleth is just such a protocol for exchanging information about users though the web.

Shibboleth originated with the US Internet2 initiative, and does some of the same things as the existing UK Athens system. It is being widely implemented in USA, has already been adopted by national higher education systems in Australia, Switzerland, Finland and the Netherlands. (The word Shibboleth, which means an ear of corn, comes from the Book of Judges. It was pronounced differently by different tribes in Israel and was used as a language test to identify potential victims during a massacre.)

Shibboleth does not carry out authentication itself. Instead, Shibboleth defines a set of protocols for the secure passing of authorisation information between institutions and service providers. It relies on the home institutions (the identity providers) to establish identity and on the service providers to confirm access rights once given information about institutional affiliation. All responsibility for user authentication is devolved to the user's home institution. So trust is vital.

How Shibboleth works

Shibboleth is based on service provider software (SP) at the site that controls access to the resources a user wishes to use (also known as the "target system"), and identity provider software (IdP) in a site which knows about the user. It decouples the management of directories by the IdP from access control to online resources by service providers so that different kinds of directory can be used to provide information to the access control software.

Briefly, when a user tries to access a Shibboleth protected resource, they are redirected, to their home organisation. After successful authentication, by whatever system is in use there, a one-time "handle" or session identifier is generated for this user session, and they return to the resource.

It can implement an authentication scheme that both certifies the identity of the user and provides other attributes about them. The authorisation aspect is more important than the authentication - i.e. the server doesn't care "who is this person?" but needs to know "is this person allowed to do what they are trying to do?". Because of this the information passed back to the service provider is about status and role rather than personal identity – it asserts that "whoever supplied me with those credentials is entitled to do this" rather than "this is such-and-such a real person". It is – or can be – limited, so that certain attributes are kept secret, or only revealed to certain SPs.

The resource provider can use the handle to request attribute information from the Identity Provider for this user, and based on the attribute information made available, the resource then allows or denies the user access to the resource.

See the glossary below, (Appendix 3) for more detail.

Potential importance and benefits of Shibboleth

Shibboleth is likely to be very significant in the near future. It is a central part of the new JISC Access Management Infrastructure for the UK educational sector intended to be available for full production use by September 2006.

Shibboleth doesn't so much replace as build on the existing UK academic Athens system. AthensDA already has similar functionality to Shibboleth, though it is put together very differently and does not use international standards. There will be an Athens/Shibboleth gateway to allow Athens users to access Shibboleth-protected resources, and Shibboleth users to access Athens-protected resources. The current Athens contract with JISC will be renewed until July 2008, and will run alongside the Athens/Shibboleth gateway for at least two years allowing institutions the time to migrate.

Shibboleth is intended to provide various benefits:

- better management of access to our institutional systems for our own users and for outsiders.
- students and staff can access a wide-range of resources by using their single institutional username and password.
- librarians, academics and departmental administrators freed from the burden of username/password management.
- a common mechanism for access to internal resources, including administrative systems where role-based authorisation may be particularly appropriate.
- online access to third-party resources provided by publishers and other institutions, at least as extensive as the current Athens system.
- support for resource sharing between institutions.
- support for dynamic, ad-hoc collaborations such as the "virtual organisations" of e-research and Grid computing.
- trustworthy control over access to services we provide outside the college
- Shibboleth also needs little central infrastructure – almost all processing is, or can be, devolved to member institutions.

ShibboLEAP

ShibboLEAP (<http://www.angel.ac.uk/ShibboLEAP/>) is a partly JISC-funded early adopters programme initiated at the LSE to Shibbolise access to the eprints institutional self-archive of academic publications established as part of the SHERPA-LEAP project. (See appendix 1 below for more details) The SP is an eprints document repository at UCL, the IdPs are systems run separately by each participating institution.

Reasons for Birkbeck to participate

Birkbeck was in some ways well-placed to participate in ShibboLEAP. The Library was already involved with LEAP, and there is a history of good co-operation between Library systems staff and the Central Computing Services (CCS). Via the London Knowledge Lab (a collaboration with the Institute of Education), Birkbeck is the technical lead in the Lifelong London Learning for All portal development project (<http://www.lkl.ac.uk/research/l4all/>) which also involves LSE staff as well as representatives of other London HE and FE colleges.

Some other features of Birkbeck are perhaps a little more ambiguous. CCS is quite a small systems and administration team with perhaps a relative lack of specialisation. One team member (the present author) was in a position to do almost all the work needed – which has both advantages and disadvantages, some of which are described below.

The experience was also of immediate benefit to us. Much of the work needed to implement Shibboleth was work we needed to do anyway - especially the investigation of a new LDAP directory. Birkbeck has something of a history of complexity in directory structures and in student and staff registration for computer use. Looking at them hard in order to think about how to introduce Shibboleth perhaps exposed some pre-existing problems. Whether or not this can be called a good thing for the ShibboLEAP programme it probably was for us.

Early experience of Shibboleth deployment has probably a good thing for us (if only by giving us a clue about what to avoid!) There are likely to be many similar requirements for cross-institutional access in the near future.

Shibboleth Implementation decisions at Birkbeck

Decisions on directory

The most urgent choice facing us was the location of the directory information that Shibboleth could access. The main possible choices were:

1) Use our existing database

We didn't seriously consider using the existing database of registered computer users directly. Its just too messy and can only be accessed by SQL queries.

We hope (one day!) either to replace it with third-party supported software, or with a more portable solution based on LDAP, or possibly by using Microsoft Windows Active Directory throughout, but that is a long way off.

2) Unix password files

Apache authentication through Unix password files is familiar territory to us, as is user logon directly to Unix systems via telnet, samba, IMAP, and other protocols.

However, the password and group files only contain limited information about each user and there is no standard method of storing extended attributes.

Also we would either have to set up scripts moving big files around in batch (as we do now for Unix and some Apache authentication) or else rely on a directory-based single-sign on system such as Sun's NIS, which has serious scalability problems, and has caused performance difficulties for us in the past.

3) LDAP directories

A centralised LDAP directory could be used to concentrate information derived from our large number of different directories. This seemed to be the most "future-proof" option.

Not only is the Apache, Tomcat, Shibboleth combination already set up for LDAP so it should work "out of the box", but the effort put into developing an LDAP system might well pay off later in allowing us to simplify, extend, or replace, our existing infrastructure.

We needed to make a strategic choice of LDAP system. Two were considered:

3.1) Microsoft Active Directory

advantages:

- we already have one
- (almost) everyone in College is already in it
- It is likely that 3rd-party software tools will be available to allow easy batch administration of WAD – though this is not yet the case

disadvantages:

- difficult management, complex Windows tools tuned for small numbers of users
- no good 3rd-party tools yet (as of summer 2005)
- everything needed to operate Windows depends on directory, even minor troubles with it can be tricky
- require making some local modifications to the schema
- would need to back out of local changes before any Windows upgrade and reinstall - this was the clincher

3.2) Special-purpose database - we install an OpenLDAP server on Unix somewhere and arrange for that to mirror our user database.

advantages:

- simpler
- can experiment
- trivial to make schema changes

disadvantages:

- yet another server,
- yet another skill requirement
- syntax of commands is not intuitive to most system administrators
- some performance and scalability problems (to a certain extent these can be alleviated by appropriate indexing)

OpenLDAP was chosen mainly because it was the best-known open-source stand-alone LDAP implementation and already has a significant user community.

That may well not have been the best solution. There are other approaches to directory provision, such as Novell Netware, or the Fedora open system, which have real advantages which we did not consider.

Decisions on content and organisation of LDAP database

This is in a way the most critical of the tasks because it commits us to things we won't easily be able to back out of when the system is live.

1) Our LDAP structure cannot be based on existing Windows Active Directory structure

Windows Active Directory (WAD) contains a large number of objects and attributes that are purely internal to Windows and would be of no direct relevance to other uses for the directory.

Our implementation has a rather confusing organisational structure. Although only four or five years old the WAD used at Birkbeck has been modified on an ad-hoc basis with little obviously remaining from its original plan. Some confusion has been introduced by inconsistent attempts to track changes in the management structure of the College that have been made over the years.

Some elements that perhaps ought to be at the same level in the hierarchy are at different levels. Some departments are at the second level, others at the third.

For example, immediately below our top-level container `DC=birkbeck` we have a number of OUs representing departments, as well as Students, Trade Unions, Students Union and so on. But we also have non-OU containers based on the first character of student usernames such as `Jusers` and `Kusers`; as well as an OU called "Disabled Staff and Postgraduates" (used as a container for usernames whose owners have left – it is the computer access rights, not the users, which are disabled)

Most Library users are in the Library Users OU, others are in a quite different Library Users container. Some CCS staff are inside the CCS OU, others aren't. Many computers and servers are in that OU as well. Some but not all users are in the Users container. `OU=Foundation Degrees` mysteriously contains two objects which seem to be student workstation PCs.

This did not seem a good basis to start planning the new LDAP from.

2) Different data sources have different unique identifiers

Our staff database has a unique staff number, our student database a unique student number. However the LDAP directory has to be able to contain information about people other than members of staff or students, as well as objects which do not represent a human being at all.

So another identifier is needed as a unique name within the directory.

3) Personal names are not adequate unique identifiers

There is no central record of people as such kept at Birkbeck. As described above the CCS user registration is based on records from diverse sources. We cannot reliably use personal names to automatically merge data from the different sources because they keep names in different formats. Some record titles, others don't. Some have full names, others only initials.

For example one person has the following names in different lists:

Mr Robert Kenneth John Brown

Mr K Brown

Mr R K J Brown

ROBERT BROWN

MR K BROWN

However: Robert Brown and Mr R Brown are someone else

Names are sometimes recorded in different spellings – sometimes by mistake, sometimes because people write their own name differently at different times. There is no consistent capitalisation or punctuation of names containing patronymics such as De, Du, Mac/Mc, O, Van, Von. We cannot guarantee, without human intervention, that names like ONEIL, O'NEIL, ONEILL, O'NEILL, NIELD, and NEIL, in fact refer to different people.

There are also well-known problems in matching names from different cultures naming systems. Some have no concept of surname, others have more than one surname. Some people use a parents name as a surname in some circumstances, not in others. Some write the family name first, others second, so there can be different results from different forms. Some non-English-speakers Anglicise their names in an English language context, others don't, and some do it inconsistently.

4) The CCS username is to be used as a unique key field.

This was the most significant choice, and one which constrains all others.

Some people have more than one username – unavoidable in a part-time adult education environment. More rarely (we have almost eliminated it, but not for some visitors and temporary and casual staff) more than one person sometimes shares the same username.

The CCS username is forced to be unique in practice because it is the unique key of the Unix password files in which every member of college who has any computer access is listed.

5) there is in our database at the moment no positive evidence of staff status.

Most, but not quite all, staff have usernames of a different format from most students. There is an inevitable blurring of roles especially among postgraduate students some of whom appear to be paid staff (well, of course some of them are) and others appear to be students. When undergraduates move onto a postgraduate course they typically retain their student account, but new joiners in some departments are issued with ones that look like staff accounts. If postgraduate students become full-time employees they sometimes retain their previous student usernames, or sometimes are issued with new ones. As things are it is a few staff might leak through as "other" and research students with no course code can come out as staff.

At a later date we might modify the new database to include membership of the departmental groups of staff that are automatically derived from the HR database and are the basis of some mailing lists. Also we could include all the data from the "access code" attributes held on our computer user registration database.

6) Attributes required by Shibboleth

LDAP attributes were discussed by the Shibboleth team and it was decided that the IdP should always send:

- eduPersonScopedAffiliation (HTTP_SHIB_EP_AFFILIATION): values required are MEMBER (for all) and [something else undecided] (for academic staff).
- eduPersonPrincipalName (REMOTE_USER): should take form login@domain.ac.uk not an email address. This was easy for us as this is exactly what we are using as our unique id anyway
- mail (HTTP_SHIB_INETORGPERSO_MAIL): email address. Everyone has an email address, though it isn't always the one they actually use
- ou (HTTP_SHIB_INETORGPERSO_OU): Department (for detection of members of library staff)

eduPersonPrincipalName is intended to be the minimum standard for the JISC Shibboleth Federation. Other Federations make use of free-form content in eduPersonEntitlement. Also "department" as such is not used in Birkbeck-speak but we can fill in that field with School/Faculty etc - all that we require here is that library staff should be recognisable.

7) OU structure in the LDAP directory

It is very important to establish a useable hierarchy of containers in the LDAP directory right at the beginning, especially the names and levels of organisational units. It is not always possible, or necessary, to exactly reflect the actual departmental structure of the organisation – reorganisations happen more often than major software implementations.

We decided on a simple directory structure. The directory is mainly intended for authentication and access and at the moment all objects (apart from a couple of administrative objects) are inside `ou=people,dc=bbk,dc=ac,dc=uk`. However, having "people" as a top-level OU allows space to extend the directory to other categories of object if needed without having to fit them into the structure of departments or schools

The following OUs have been set up under `ou=people`:

```
ou=external
ou=otherpeople
ou=staff
ou=student
ou=temp
ou=unknown
```

This forces us to make a decision as to staff or student status when adding a name to the directory, something that is required for this project

Decisions on platform

1) We needed a **local authentication system** for the web server capable of querying LDAP server.

We might have this anyway - apparently mod_authz_ldap on Apache works. Various implementations of WebISO are also OK. If not set up yet should be easy once certificate infrastructure in place. So in practice this decision defaulted once it was decided to use Apache.

2) **Server certificate.** (in fact two if web server & LDAP on different machines) We were strongly recommended to use Globalsign.

3) **Location of servers.** There is no reason that Shibboleth, Apache, and the LDAP directory cannot run on different servers. In an operational environment there may be good reasons for separating them, however as this was a test project, and for simplicity, one single computer was used for all three.

4) **Server software**

4.1) **Java**

Shibboleth has multiple dependencies on the level of Java installed and on classes available. Not all of these were understood at the start of the project and some choices made caused problems later.

4.2) **Apache on top of Tomcat**

Mainly chosen because it is by far the most common way of providing a web application infrastructure and Shibboleth already works on it.

4.3) The operating system chosen was **SuSE Linux.**

Most of our infrastructure is based on either Windows or Sun Solaris Unix.

Unix was chosen rather than Windows because system administration is simpler, the OS is more robust (though this is far less of a distinction than it used to be) and because server software availability better (still important though also less so)

Linux rather than Solaris mainly because it was easier to re-use an existing Windows/Intel machine than to source a new Solaris server. Also software packages tend to be more up-to-date on Linux than on Solaris.

There was no particular reason for the choice of SuSE over other Linux other than that the most recent Linux machine installed by our team had also been SuSE - and that was because at that time it was being extensively used by the Crystallography department at Birkbeck so there was some local knowledge.

NB The current machine we have LDAP & Shibboleth on is not suitable for critical production work. If Shibboleth and/or OpenLDAP were to become vital parts of our infrastructure it should be reinstalled on a more robust server in the computer room, and have an alternative machine for backup.

5) **Other software**

See Appendix 1 below for a more complete list of installed software

Decisions on the provision of attributes to the authentication database

There are (conceptually) two stages to providing access and authentication information. First we need to populate the database with information about people (or groups or other entities), then we need some way of returning recording which entities are allowed to do what.

We already have this information about access rights in the access codes implemented on our user registration database (user_reg) which can be stored as group memberships on Windows, or in various ways on other systems. So whatever software copies from this to the database accessed by Shibboleth needs to be able to record these groups as attributes of a user.

As part of the development of our LDAP strategy we looked into ways of obtaining the information about users and groups and access codes we need to populate LDAP. It was decided to at least start with our registration database & user database, and perhaps take information from Windows Active Directory to supplement it.

We already maintain a reporting database containing a synthesis of the user data, for the purposes of programs called "whois" and "userlist" which can be used to look up individual details through webpages. It is a separate database partly for speed, partly because of objections to giving end users any web-based contact with the live registration database (even when buffered by CGI code). It was decided to use that database as the primary source for populating LDAP.

The main source of data for these report-only tables behind the whois program is the "raw" copy of the Unix password file (i.e. still containing suspended users) generated by the existing user registration database. This was chosen because the username has to be the unique key (as explained above) and this list contains one and only one record for every possible username. The data is supplemented by other queries to the registration system database (user_reg) which collect student coursecodes and also fuller details of their names (for example to enable the web page to display these full names rather than just the short forms kept in the password file).

One of the problems with doing this was that there was little or no documentation on the database. In order to be sure the data is what it was thought to be we had to spend some days looking through user_reg as was at the time and attempting to document the tables.

Attributes to be stored and returned.

Its worth noting that there are (at least) two places in which we build the attributes Shibboleth returns for each user:

One is to have them permanently in the live directory, the database in which Shibboleth looks up details of the user. This is what we're doing at Birkbeck – an LDAP directory has been set up containing EduPerson and Posix and other attributes for every one of our users, and written code to construct them from the information we already have on each user using LDIF format to import initial user data into the OpenLDAP directory.

It is hoped to develop this code into something that can be used "real time" when a user name is created or modified, so that the LDAP is always up to date (and can perhaps be used for Unix & website authentication some time in the future & possibly as the data source for our Windows Active Directory). Our entire existing registration system is updated in batch at the moment – password changes and other minor alterations run once every half hour or so, but new users are added, or email addresses changed, overnight. So, although desirable in the medium term there did not seem any point in setting up a system to update the OpenLDAP directory in real time yet.

Another option is to build attributes dynamically in the IdP when the user logs in. This can be achieved by configuration options in the IdP (such as the format resolver.xml and idp.xml configuration files). Apparently there are also ways to alter attribute values with regular expressions, though none of us are doing that yet. Some places that use the Microsoft Windows Active Directory are doing this because they are wary of making permanent changes to the WAD schema.

At least one attribute must be built by the IdP in any case - sometimes Shibboleth sets up a unique but anonymous identifier for the combination of a particular username, origin, and target, which of course depends on context at authentication time. (Apparently there is some problem with this in older versions of Java which means it might need to be reinstalled for it to work fully)

It is not a large problem for the ShibboLEAP project because the initial implementation of eprints is reasonably simple in that eprints is public-access - at least at first the only distinction is between those allowed to post to the eprints server and everyone else in the world. So authentication in effect is access control, as long as library staff can be distinguished from anyone else, but future use of Shibboleth might require much more granularity of attributes.

Existing directories at Birkbeck

Birkbeck doesn't have just a single repository of information on staff or students. The computer and network user registration and authentication systems used at Birkbeck have grown up over many years of mostly in-house ad-hoc development.

This is a very brief overview of the registration process which enables users to get accounts on our computer systems. The details relevant here only insofar as they show where we had to look to get data to populate our new directory.

The present CCS user registration system is based round an SQL database (user_reg) containing over 80 tables which are used for a wide variety of different functions which do not all obviously belong together. It is partly a copy of a locally written database system, which was itself partly based on user databases held on a VAX. It does not, as it is, provide a very robust platform for further development and is very much in need of documentation, and perhaps simplification (if not replacement)

The initial population of the user_reg database is done differently for different classes of users. Once a username is set up its details can be examined or changed by authorised staff through web pages written for the purpose.

Most student records are added automatically, based on files exported from the student registration system. There is a new student database system that should include all students. The situation is complicated by increasingly large numbers of odd categories of students with different requirements and access permissions. For example FCE students ("Faculty of Continuing Education" previously "Further and Continuing Education" – a Birkbeck faculty providing a variety of short courses that is the successor to the old University of London Extramural system) automatically get usernames but these are not activated until they do it themselves via a web page, or get us to do it.

Staff records are automated rather differently – details of staff joining and leaving arrive at CCS based on the HR system, but the username is not set up until someone goes online and initiates it by typing in the staff number, which then causes the username to be set up with data from HR. It very frequently happens that staff join before these records are available and their usernames have to be set up by hand – which is relatively easy to do through our webpages, but can introduce confusion if names or titles are spelled differently on our system and the HR system.

Quite a few users fall into the gaps between these systems and are still set up by hand based on paper forms filled in by members of staff.

Perl programs run overnight to create database tables corresponding to data needed by Unix, Windows, and email systems. The live systems are updated in batch by files pulled down from the registration database through a tcpip service resembling a cut-down FTP daemon that was written at Birkbeck. I think it is fair to say that we would not do it this were we re-implementing the service this month. Or even this century.

The Windows users are compared to a list of existing users on the WAD, new ones added and old ones deleted, by Windows batch files. Unix authentication is via password files downloaded by scripts that run every half hour or so. Further scripts then copy version of the password file into Apache authentication directories and also generate versions for Oracle, email, and other servers.

The daemon is capable of generating files containing different subsets of the user population depending on the settings of a number of binary "access codes" associated with each user. These are in fact very useful in the current situation as they map rather well onto attributes that might be useful in an LDAP database. There are over twenty of them, coding for such things as "Unix interactive access" or "Exchange email" or "Access to staff intranet" or "NT workstation login (library)"

There are a number of different version of the password files, containing different subsets of users based on these access codes, that are loaded to different machines. For example, some servers have password files which only contain users who are members of groups with systems administration rights on those machines.

The registration system in fact works rather well for our Unix and email servers – it allows very flexible control over user groups and is quite robust. The main problems is that changes do not take effect in real time, there is an in-built delay of anything from a few minutes to an hour. The Windows part of the registration system is adequate but limited. There is a certain amount of manual intervention required in order to set up staff users on the administration systems, and some changes have to be made to the Active Directory using Windows tools rather than being controlled centrally.

One day we want to have one authentication system for people accessing computer resources at the college (one directory would be good too, but that's a bit like the end of a rainbow and I suspect we will never quite get there - or if we do we'll find that the problems we solved weren't the ones we really had) In the near future a common authentication system would almost certainly either be Windows Active Directory (or its successor) with various bits of proprietary software to force everything else to use it; or else some other sort of LDAP directory, probably open-source, probably running on Unix, used by both Unix and ever-increasing number of web-based systems, and with some method of sharing the data in real time with Windows.

As already mentioned, Birkbeck has a wide variety of directories and authentication systems and an even wider variety of databases behind them. They include, but are not limited to:

College administration data:

- Student information databases (BSIS/SITS - we are just about to complete the migration from one database system to another and have all the students on the same system - though we did not when ShibboLEAP was first considered)
- records held, sometimes more or less informally, by various departments of students who are about to join a course and already in contact with the department, or who have completed a course but are still in some relationship with them, or who are on distance learning courses which do not involve registering as a Birkbeck student
- HR records for staff employed centrally
- HR records for staff employed as sessional lecturers in FCE (different data is held on, as far as I know, a different system)
- Records for staff employed as contractors
- Records for staff employed by departments, or casually (sometimes on a relatively informal basis)
- payroll and pension databases
- Library user systems
- College printed phone book (partially compiled by hand, partially automatically set up from data held by CCS)
- Online phone book (a different branch off the same semi-manual system as sets up the printed book – basically a large spreadsheet)

Physical access systems:

- Student ID cards
- Staff ID cards (a different system)
- Card access to main college buildings through staff cards
- Card access to Library through both student and staff cards
- Card access to Crystallography (a different system again)

CCS-managed computer registration systems & directories:

- user registration database (locally designed MySQL database "user_reg" running on a Solaris machine)
- Windows Active Directory
- Windows NT registration for admin computer users not in WAD
- email tables for mail hub
- email tables for Mirapoint mail appliance
- Unix password files
- Athens (Most users now using AthensDA, though there are some individual usernames still)
- WebCT
- Many cut-down password lists used for Apache authentication, some generated automatically, some by hand
- Zope/CCS "Intranet"/etc which do not share the main username & password

Other computer registration systems:

- Crystallography
- Computer Science
- Economics
- local Unix machines and web servers in a number of departments
- a number of local systems in labs and offices all over the place we might not even know about

Implementation experience at Birkbeck

Documentation of existing CCS directories

Documentation of existing CCS user registration databases and other sources of user name data and user attributes was a necessary pre-requisite to populating the LDAP database. This was largely done during July & August, but is still ongoing.

Due to the many and varied sources of directory information this was quite a large amount of work. One would hope that it would be less in most other places. Some of that results of that work are included in this document.

Among the points that emerged:

- There are many tables that have nothing to do with user registration. For example there are tables to do with DNS, and with other network management tools
- Some of the tables are not normalised. This probably isn't really a problem.
- Some tables are obsolete. It was very confusing to find a large number (in one case twelve) of apparently similar tables containing similar data and no clear guidance as to which if any were the correct ones to use.
- Some tables look like they were set up to implement things that were never finished.

Installation of SuSE Linux 9.2 with Apache 2, OpenLDAP et.c etc.

Quicker and easier than planned for, mainly because SuSE Linux administration program, YaST, works really well these days - or did at first. Installation and upgrade of well-known packages online is almost entirely automated and simply a matter of choosing (or approving) a repository to download from and letting it go.

Configuration of OpenLDAP with EduServe and Posix schemas

This was simpler than at first expected. It took some hours to reading the documentation and look at the EduServe Schema to try to understand how it worked, but only a few minutes to actually install the schemas – a matter of copying text files to the correct place, and restarting the service.

Population of LDAP database with modified copy of Birkbeck live directory

This involved considerably more work than had been assumed. It took about three weeks worth of work spread over four or five weeks with a break of a month or so in the middle. It was done in one large batch for the initial loading of the directory, using Perl programs to write LDIF files which were imported into OpenLDAP. This stage was completed in November.

The data source is the CCS user registration database (necessarily so because that is the only place that both contains staff and student information and has all email addresses) Some, but not all, information from this database is copied overnight to a reporting database that is used as an information source for our "whois" program & some other tools we have. I modified (and documented) both the code that loads the database (mkpasswddb.pl, mknodedb.pl) and the program that reads the database (whois.test.pl).

A program (userlist2ldap.pl) was written to convert this to ldif format and generate the appropriate attributes.

Later it can be modified to run in "real time" allowing dynamic creation & update of users. At some future time we will need to write code to copy some of the other attributes we keep (for example access codes).

The LDAP database is fully populated with Birkbeck users (as of sometime last year) At the moment we can currently only authenticate a small number of test users whose passwords are defined individually on the LDAP server. For reasons of security and consistency current password information has not been added to the database, apart from those test users/

OpenSSL

Installation of OpenSSL itself is trivial. Making it work is harder. The tools supplied were used to install certificates onto Apache and test that they were working.

Firewall changes

The institutional firewall was reconfigured to allow external access to ports 80 (http), 443 (https), and 8443 (locally defined for Shibboleth) on the Tomcat server.

The local firewall on the server was reconfigured to allow access from those ports, and also 22 (ssh), 389 (ldap), 636 (ldaps), with the intention of permitting only internal users to see those services.

Globalsign certificate

This was the most problematic part of the install, and the main cause was a local lack of understanding of what was required. The first attempt to obtain a Globalsign certificate failed because the wrong information was supplied, due to a misinterpretation of the "openssl req" command. An email sent to Birkbeck by Globalsign was also misunderstood, and the end result was that we obtained a certificate for " bbk.ac.uk " which is not useful in this context – it should have been for shibboleth.bbk.ac.uk or ldap.bbk.ac.uk

The current situation is that there is a self-signed certificate in place for our "internal" domain name sambucus.ccs.bbk.ac.uk, and we are obtaining a new Globalsign certificate with the correct domain name (either shibboleth.bbk.ac.uk or ldap.bbk.ac.uk)

Integrating the certificate with Apache was the hardest and least "intuitive" part of the whole operation, took days rather than hours, and there may still be problems.

Configuration of Apache, Tomcat, etc

This is the first part of the process that involves actually thinking about and planning for Shibboleth rather than directory structures.

It took slightly longer than hoped. YaST (SuSE system management tool) broke due to a SuSE security upgrade which went wrong. It deleted an old version of the Curl library used by YaST to fetch things from the Net, which had to be manually reinstalled. (or semi-manually - curl was fetched in RPM format (not native to SuSE) from <http://curl.haxx.se/download.html> as the file curl2-7.11.0-4.1.src.rpm - YaST handled the RPM correctly)

However **that** caused automatic configurations of Apache to fail until some of the files in the Apache configuration directories had been reorganised.

Suse installed libcurl3 but YaST et.c still wanted libcurl2 This caused us to have to downgrade some software, including Java 1.5 back to 1.4.2 and also Tomcat to 5.0, because they used the new version. It also downgraded acroread from 7.0.1-2.2 to version 5.010-4.1

I decided to go ahead with the older Java and Tomcat versions until Shibboleth is working & then consider upgrade or install at a higher level on a new machine. In hindsight this was possibly the wrong decision.

Install Shibboleth id server software.

1) documentation for the then most recent release of Shibboleth, version 1.3 (released Jul 26, 2005) was downloaded from <http://shibboleth.internet2.edu/>
We also downloaded the Shibboleth Identity Provider Deployment Guide (v1.3 / 2005-08) and the Origin 1.2.1 deployment guide (older but more complete) <http://shibboleth.internet2.edu/guides/idp/> and <http://shibboleth.internet2.edu/guides/deploy-guide-origin1.2.1.html>)

2) IdP was installed from shibboleth-idp-1.3c.tar.gz downloaded from <http://wayf.internet2.edu/shibboleth/>

3) Configuration started by running the `ant` program (Apache's Java/XML build tool) Its mostly automatic and nearly everything defaults.

We were presented with a few choices:

- install on file system not via tomcat (default)
- home = `/usr/local/shibboleth-idp`
- tomcat home = `/usr/share/tomcat5`

It makes loads of files including:

`/usr/local/shibboleth-idp-1.3c-install/build.properties`

4) Set up XML files

- As we were using LDAP, the supplied file `Resolver ldap.xml` was used to over-write `resolver.xml`
- There were changes made by hand to `idp.xml` and `resolver.xml` (details attached)

Further Configuration of Apache & Tomcat

Some Java libraries supplied with Shibboleth were copied to Tomcat's endorsed jar file directory (code in an "endorsed" directory over-rides the default installation code)

```
cp /usr/local/shibboleth-idp-1.3c-install/endorsed/*.jar  
/usr/share/tomcat5/common/endorsed/
```

Apache was configured to use mod_jk to redirect queries for Shibboleth components to Tomcat.

During this stage there was a series of errors perhaps caused by a lack of understanding of Java or Tomcat. The default installation seemed to have two or three versions of most things and it was not always obvious which was in use.

Current situation

The IdP still does not work and some problems remain.

It is possible that the main problem is that Shibboleth seems to depend critically on versions of Java libraries which are "endorsed" by Tomcat and which override classes included with the default Java installations. In particular the XML parsing software needs checking - things like Xerces, xercesImpl.jar, xml-apis.jar, and xmlParserAPIs.jar.

Appendix 1 - Software installed

These programs were installed or configured on the server:

Software that was configured separately – the Linux distribution of course includes a full set of Gnu/Linux/Unix tools and other server software.

- SuSE Linux 9.2
- Apache 2.0 (semi-automatically installed with Linux distribution)
- mod_jk
- Tomcat 5
- OpenSSL 0.9.7d
- OpenLDAP 2.2.26
- Java 1.4.2 (backlevel due to problem with automatic updates)
- Shibboleth IdP 1.3
- Perl 5.8.5

Programs installed or configured on PC workstation:

Software that is not a normal part of Windows that was found to be useful for configuring or testing the server.

- PuTTY 0.55
- Perl 5.8.7 (used extensively to write programs to manipulate data)
- Softerra LDAP Browser (<http://www.softerra.com/products.htm>) very useful for checking contents of OpenLDAP directory and comparing it to WAD
- PSFTP & SSH tools
- wget (very useful for setting up repeatable tests of web services and storing responses)

Programs written at Birkbeck

- mknodedb.pl
- mkpasswddb.pl
- mkuserdb.sh
- mkuserdb.sql
- mkaddressb.sql
- userlist.test.pl
- userlist2ldap.pl
- whois.test.pl

Most of these are Perl programs involved in constructing a database table and making ldif files to load into LDAP. There are over 2,000 lines of code, but the programs are mostly modified versions of pre-existing code which produced reports on the database.

Modifications to Config files

Most of the changes are obvious ones, and not described further here, though the files that were changed are listed to give some idea of the scope of the work.

Some alterations specific to this Shibboleth installation are detailed below. NB at the time of writing the Birkbeck Shibboleth IdP was not working. So it is almost certain that at least one of the following files is incorrect!

Config files written or modified or separately installed:

```
/etc/apache2/conf.d/mod_jk.conf
/etc/apache2/local.files/user.db
/etc/apache2/localconf.d/shib.conf
/etc/apache2/localconf.d/ssl.conf
/etc/apache2/ssl.crt/ca.crt
/etc/apache2/ssl.crt/sambucus.ccs.bbk.ac.uk.crt
/etc/apache2/ssl.crt/server.crt
/etc/apache2/ssl.key/ca.key
/etc/apache2/ssl.key/ldap.bbk.ac.uk.key
/etc/apache2/ssl.key/sambucus.ccs.bbk.ac.uk.key
/etc/apache2/ssl.key/server.key
/etc/apache2/sysconfig.d/global.conf
/etc/apache2/sysconfig.d/include.conf
/etc/ntp.conf
/etc/openldap/ldap.conf
/etc/openldap/schema/eduperson.schema
/etc/openldap/schema/rfc2307bis.schema
/etc/openldap/slapd.conf
/etc/sysconfig/apache2
/etc/sysconfig/j2ee
/etc/sysconfig/openldap
/etc/sysconfig/SuSEfirewall2
/etc/syslog.conf
/usr/local/shibboleth-idp/etc/idp.xml
/usr/local/shibboleth-idp/etc/resolver.xml
/usr/local/shibboleth-idp/etc/shibboleap.xml
/usr/local/shibboleth-idp-1.3c-install/conf/log4j.properties
/usr/share/tomcat5/conf/server.xml
/usr/share/tomcat5/conf/workers.properties
```

Modifications made to Shibboleth IdP XML files

shibboleap.xml was added as supplied to us by LSE

idp.xml was changed to include the following:

```
<!-- Shibboleth Identity Provider configuration -->
<!-- edited RKJB December 2005 -->
  <IdPConfig
    xmlns="urn:mace:shibboleth:idp:config:1.0"
    xmlns:cred="urn:mace:shibboleth:credentials:1.0"
    xmlns:name="urn:mace:shibboleth:namemapper:1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="urn:mace:shibboleth:idp:config:1.0 ../schemas/shibboleth
-idpconfig-1.0.xsd"
    AAUrl="https://sambucus.ccs.bbk.ac.uk:8443/shibboleth-idp/AA"
    resolverConfig="file:/usr/local/shibboleth-idp/etc/resolver.xml"
    defaultRelyingParty="urn:mace:shibboleth:examples"
    providerId="https://sambucus.ccs.bbk.ac.uk/shibboleth">
```

[...]

```

<!-- This configuration section determines the keys/certs to be used when
signing SAML assertions -->
<!-- The credentials listed here are used when referenced within
<RelyingParty/> elements above -->
<Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
  <FileResolver Id="example_cred">
    <Key>
      <Path>file:/etc/apache2/ssl.key/sambucus.ccs.bbk.ac.uk.key</Path>
    </Key>
    <Certificate>
      <Path>file:/etc/apache2/ssl.crt/sambucus.ccs.bbk.ac.uk.crt</Path>
    </Certificate>
  </FileResolver>
</Credentials>

```

Only changed lines and their immediate surroundings are included here

resolver.xml was copied from the supplied resolver.ldap.xml and modified:

```

<JNDIDirectoryDataConnector id="directory">
  <Search filter="(commonName=%PRINCIPAL%)">
    <Controls searchScope="SUBTREE_SCOPE" returningObjects="false" />
  </Search>
  <Property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
  <Property name="java.naming.provider.url"
value="ldap://sambucus.ccs.bbk.ac.uk/dc=bbk,dc=ac,dc=uk"/>
  <Property name="java.naming.security.principal"
value="cn=ldapadm,dc=bbk,dc=ac,dc=uk" />
  <Property name="java.naming.security.credentials" value="XXXXXXXX" />
</JNDIDirectoryDataConnector>

```

(XXXXXXXX is ldapadm's password)

[...]

```

<SimpleAttributeDefinition id="urn:mace:dir:attribute-
def:eduPersonPrincipalName" smartScope="bbk.ac.uk">
  <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>

```

[...]

```

<!-- This section configures the loading of SAML2 metadata, which contains
information about system entities and
how to authenticate them. The metadatatool utility can be used to keep
federation metadata files in synch.
Metadata can also be placed directly within this these elements. -->
<MetadataProvider
type="edu.internet2.middleware.shibboleth.metadata.provider.XMLMetadata"
uri="file:/usr/local/shibboleth-idp/etc/shibboleap.xml"/>

```

Some modifications made to Apache configuration files

In the SuSE implementation of Apache 2, the main configuration file `/etc/apache2/httpd.conf` is mostly a stub that contains links to other configuration files in subdirectories. The intention is that local configuration files can be modified by the system administrator without clashing with automatic updates supplied by SuSE or Apache.

The installation discussed here includes the following subdirectories and files:

conf.d: `apache2-manual.conf` `mod_jk.conf` `php4.conf` `mod_fastcgi.conf`
`mod_perl.conf` `web2ldap.conf`

localconf.d: `shib.conf` `ssl.conf`

sysconfig.d: `global.conf` `include.conf` `loadmodule.conf`

vhosts.d: `vhost-ssl.template` `vhost.template`

`/etc/apache2/conf.d/mod_jk.conf`

```
# mod_jk.conf
# added: Ken Brown, November 2005
# manually downloaded from
http://shibboleth.internet2.edu/guides/idp/installati
on.html
<IfModule !mod_jk.c>
  LoadModule jk_module libexec/mod_jk.so
</IfModule>
JkWorkersFile "/usr/share/tomcat5/conf/workers.properties"
JkLogFile "/var/log/apache2/mod_jk.log"
JkLogLevel emerg
JkMount /shibboleth-idp/* ajp13
JkMount /jsp-examples/* ajp13
```

An include file `/etc/apache2/localconf.d/shib.conf` was set up to hold data for testing Shibboleth. Among other entries it contains:

```
[...]
<Location /shibboleth-idp/SSO>
  AuthName "Birkbeck Shibboleth"
  AuthType Basic
  AuthLDAPEnabled On
# AuthLDAPAuthoritative On
  AuthLDAPUrl ldap://sambucus.ccs.bbk.ac.uk/ou=people,dc=bbk,dc=ac,dc=uk?uid
  AuthUserFile /etc/apache2/local.files/user.db
  require valid-user
</Location>
```

```
[...]
```

```
<Directory "/srv/testldap">
  AllowOverride None
  Order allow,deny
  Allow from all
  Options None
  Authname "Elder Test 2"
  AuthType Basic
  AuthLDAPEnabled On
  AuthLDAPAuthoritative On
  AuthLDAPUrl ldap://sambucus.ccs.bbk.ac.uk/ou=people,dc=bbk,dc=ac,dc=uk?uid
  Require valid-user
</Directory>
```

That `testldap` directory was added after the initial installation of IdP failed, to show that the LDAP database can be used by Apache to authenticate users of static web pages. It works, eliminating OpenLDAP and (almost all of) Apache from suspicion.

/etc/apache2/localconf.d/ssl.conf

```
<IfModule mod_ssl.c>
<VirtualHost _default_:8443>
  SSLEngine On
  SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
  SSLCertificateFile /etc/apache2/ssl.crt/sambucus.ccs.bbk.ac.uk.crt
  SSLCertificateKeyFile /etc/apache2/ssl.key/sambucus.ccs.bbk.ac.uk.key
  SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt
  SSLVerifyClient optional_no_ca
  SSLVerifyDepth 10
  SSLOptions +StdEnvVars +ExportCertData
  ErrorLog /var/log/apache2/ssl_error_log
  TransferLog /var/log/apache2/ssl_access_log
  HostnameLookups Off
  UseCanonicalName Off
</VirtualHost>
</IfModule>
```

NB these files are referring to the server by its primary DNS name, sambucus.ccs.bbk.ac.uk

The top-level Apache config calls the server by an externally-known alias, ldap.bbk.ac.uk

This might not in fact be correct.

OpenLDAP configuration**/etc/openldap/ldap.conf**

```
# LDAP Defaults

[...]

#BASE dc=example,dc=com
BASE dc=bbk,dc=ac,dc=uk'

[...]

URI ldap:193.61.22.84 ldap://ldap.bbk.ac.uk ldap://sambucus.ccs.bbk.ac.uk
```

/etc/openldap/slapd.conf

```
[...]

include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/eduperson.schema
include /etc/openldap/schema/rfc2307bis.schema
include /etc/openldap/schema/yast.schema
```

[... security policies also changd but not listed here! ...]

Tomcat configuration

/usr/share/tomcat5/conf/workers.properties

```
# Attempt to get the thing to work with SuSE
# Based on hints on various websites

workers.tomcat_home=/usr/share/tomcat5
workers.java_home=/usr/lib/jvm/java

ps=/
worker.list=ajp12, ajp13
worker.ajp12.port=8007
worker.ajp12.host=localhost
worker.ajp12.type=ajp12
worker.ajp12.lbfactor=1
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
worker.ajp13.lbfactor=1
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=ajp12, ajp13
worker.inprocess.type=jni
worker.inprocess.class_path=$(workers.tomcat_home)$(ps)lib$(ps)tomcat.jar
worker.inprocess.cmd_line=start

worker.inprocess.stdout=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stdout
worker.inprocess.stderr=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stderr
```

Globalsign certificate and OpenSSL

OpenDDL was installed and used to request a key:

```
openssl req -new -nodes -keyout ldap.bbk.ac.uk.key -
```

When prompted for further information the following choices were made:

```
-Country Name (2 letter code) [AU]:uk
-Locality Name (eg, city) []:London
-Organization Name (eg, company) [Internet Widgits Pty Ltd]:Birkbeck College
-Organizational Unit Name (eg, section) []:CCS
-Common Name (eg, YOUR name) []:Ken Brown
-Email Address []:k.brown@bbk.ac.uk
- challenge password []:*****
An optional company name []:Birkbeck College
```

"Common Name" in fact should have been the name of the server - this wasted a month. So it was tried it again setting "Common Name" to domain name bbk.ac.uk - which of course was also entirely wrong. The name should be a DNS name of the web server.

It was also not entirely clear at first where the certificates should be installed. There was some local confusion between the roles of Apache, Tomcat, SSL, and LDAP (all of which can have certificates installed in them). Our Apache directory layout does not closely resemble that implied in either Globalsign or Shibboleth docs.

Appendix 2 - Populating OpenLDAP directory

This was another lengthy process. Most of the work was writing scripts to convert the results of database queries into LDIF format for importing into the new directory, but there was also some considerable trial-and-error involved in getting the first objects installed into the directory, and in finding a minimum set of attributes that could be used to define a test user. In this case the person doing the work was familiar with LDAP and LDIF - different implementations require slightly different formats.

These are some commands that actually worked (pasted from screen)

A command that lists many objects:

```
ldapsearch -x -LLL -b 'dc=bbk,dc=ac,dc=uk' -v -s sub '*' '+'
```

And this lists an OU:

```
ldapsearch -x -LLL -b 'ou=unknown;ou=people,dc=bbk,dc=ac,dc=uk' -s sub '*'
```

This lists one user:

```
ldapsearch -x -LLL -b 'dc=bbk,dc=ac,dc=uk' -v -s sub '(cn=ckend01)'
```

This lists everybody and then extracts cn names containing the string "test":

```
ldapsearch -x -LLL -b 'ou=people,dc=bbk,dc=ac,dc=uk' -s sub '*' | grep 'cn:'  
| grep test
```

Command to add an object by importing an LDIF file:

```
ldapadd -x -W -c -S /tmp/ldstuff -D "cn=ldapadm,dc=bbk,dc=ac,dc=uk" -f  
/etc/openldap/ldif/someOU.ldif
```

LDIF file used to add a number of OUs:

```
dn: ou=student,ou=people,dc=bbk,dc=ac,dc=uk  
objectClass: organizationalUnit  
ou: student
```

```
dn: ou=staff,ou=people,dc=bbk,dc=ac,dc=uk  
objectClass: organizationalUnit  
ou: staff
```

```
dn: ou=temp,ou=people,dc=bbk,dc=ac,dc=uk  
objectClass: organizationalUnit  
ou: temp
```

```
dn: ou=external;ou=people,dc=bbk,dc=ac,dc=uk  
objectClass: organizationalUnit  
ou: external
```

```
dn: ou=unknown;ou=people,dc=bbk,dc=ac,dc=uk  
objectClass: organizationalUnit  
ou: unknown
```

LDIF to add a user who conforms to both eduPerson and Posix schemas:

```
dn: cn=aandel16,OU=student,OU=people,dc=bbk,dc=ac,dc=uk
cn: aandel16
displayname: Mr Allan Alexander Anderson
description: test ldif to add aandel16
gn: Allan
initials: A
sn: Anderson
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: eduPerson
objectClass: InetOrgPerson
objectClass: posixAccount
mail: aandel16@students.bbk.ac.uk
eduPersonEntitlement: http://shibboleth.bbk.ac.uk
eduPersonNickname: AA Anderson
eduPersonOrgDN: dc=bbk,dc=ac,dc=uk
eduPersonOrgUnitDN: OU=student,OU=people,dc=bbk,dc=ac,dc=uk
eduPersonPrincipalName: aandel16@bbk.ac.uk
eduPersonPrimaryAffiliation: student
eduPersonAffiliation: student
uid: aandel16
gecos: AA Anderson
uidNumber: 999999
gidNumber: 9999
homeDirectory: /home/DDDDDD/aandel16
```

NB the LDIF standard (see RFC2849) allows for Base64 encoding, and it is automatically imposed if there is any doubt at all about the representation of a string. A trailing space can cause it. The database from which the student records were extracted can include a trailing space on strings. In the case of the user above it had a trailing space on the displayname. So an attempt to list that entry produced:

```
displayName:: TXIqQWxsYW4gQWxleGFuZGVyIEFuZGVyc29uIA==
```

The double colon ":" in "displayName::" is an indication that the string is Base64 encoded. To avoid Base64 encoding all characters should be in SAFE-UTF8-CHAR and all initial characters to be in SAFE-INIT-UTF8-CHAR as defined in the RFC, no value should end with a space, there should be no values with linebreak before any nonspace character, and no leading ":" or "<" characters.

Command to modify an entry by importing lines from an LDIF file:

```
ldapmodify -x -w XXXXXXXXX -c -S /tmp/ldstuff -D
"cn=ldapadm,dc=bbk,dc=ac,dc=uk" -f /etc/openldap/ldif/testmod.ldif
```

NB this form of command allows the password to be specified in clear in a script - XXXXXXXXX is the login password of user ldapadm, which has admin rights.

LDIF to modify an object

```
dn: cn=sntttest10,ou=staff,ou=people,dc=bbk,dc=ac,dc=uk
changetype: modify
description: test modification from ldif
```

LDIF for adding a password:

An encrypted password (ZZZZZZZZZZZZ) is taken from the Unix shadow file and added to LDAP. This method could be used in batch to import large numbers passwords from old Unix user passwd entries:

```
dn: cn=sntttest10,ou=staff,ou=people,dc=bbk,dc=ac,dc=uk
changetype: modify
userPassword: {crypt}ZZZZZZZZZZZZ
```

Appendix 3 - glossary

AAA - Authentication, Authorisation and Accounting

Access Management - the process of permitting access to protected online information.

Access management system (AMS) software to help create, manage and issue credentials and associated attributes for access to online resources. Ideally a distributed access management system will provide:

- trust - information about users should come from a reliable source with an auditable data path, with no obsolete or unauthorised users
- security - only authorised parties should be able to read or amend the information
- privacy - the AMS should reveal the minimum amount of information about a user to only enable them to gain access to the resources they need
- neutrality - will be used by a variety of service providers and resource owners who may be in competition so should not favour one over an other

Accounting - keeping track of usage of online resources by individuals or the organisation as a whole, providing an audit trail of usage and unusual behaviour.

Ant - open-source Java build tool, intended to provide a cross-platform replacement for make and similar programs. Originally written at Sun as part of Tomcat, then passed to Apache Jakarta. It has become the default tool used to build and manage Tomcat and Java webapps, including Shibboleth, so an installation of Shibboleth at least partly consists of running the Ant a number of times.

Apache – the most commonly used web server, maintained by the Apache Foundation

Athens - a UK-wide access management system, managed by EduServe and mainly used within higher education and the NHS, offering access to a wide variety of online resources.

AthensDA (Athens Devolved Authentication) variation on Athens system which trusts the local authentication mechanisms of participating organisations (including Birkbeck) enabling single-sign-on to a wide range of online services. Conceptually very similar to Shibboleth, though implemented differently. AthensDA will either be replaced by Shibboleth or (perhaps more likely) be merged with it, so that AthensDA users can access Shibbolised SPs, and Shibboleth IdP users access Athens-protected resources.

Attributes - information about an individual in defined formats. Attributes useful in higher education might include things like institutional affiliation, status within the organisation, whether or not fees are paid.

Authentication - verifying who is requesting access to a resource.

Authorisation - determining whether access should be granted to that individual based on information about that individual.

Certificate Authorities (CAs) – providers of x.509 certificates pre-installed in the browser software. If a server is using a certificate that doesn't have a known root at a CA then the user sees pop-up windows in their web browser

Credentials - information provided by a person or by software who needs to be authenticated - for example username and password, smart cards, digital certificates.

Directory - in this context "Directory" is like a phone directory - a way to look up information ("attributes") about people or things ("objects")

eduPerson schema - some extensions to LDAP that describe personal attributes widely used by educational institutions <http://www.educause.edu/content.asp> For example, "eduPersonPrincipalName", one of the LDAP attributes of a person defined in the eduPerson schema, is meant to uniquely identify a computer user. It looks like userID@domain and is guaranteed to be globally unique. Not the same as email address, which has a separate attribute. But in practice it might be convenient to use email addresses as eduPersonPrincipalName.

EduServe - a UK organisation that provides a number of central IT services to (mostly) public sector and higher education, most relevantly here Athens and MATU.

endorsed – an up-to-date standard library of Java code is said to be “endorsed”. If newer code is needed to over-ride the installation defaults for an application, it can be packaged in jar files and copied to an endorsed directory.

eprints is an online archive of scientific or scholarly literature which allows authors to self-archive or index their own work. People upload their own work, and it is made available to those who wish to read it. A UK academic implementation is being developed at Southampton. <http://www.eprints.org/> It is part of the "Open Archives Initiative" which (among other things) suggests standard tags & keywords that will help indexing & searching - originally for scholarly papers, but with designs on bigger things <http://www.openarchives.org/documents/FAQ.html>

Federation - a group of organisations with agreed policies and common infrastructure for access management. There are many are being established at a national level. US higher education has established a federation known as InCommon. The equivalent in Switzerland is known as SWITCHaai and in Finland HAKA. There are also local federations, special-purpose federations (like Shibboleth), and federations set up by SPs, BBK & other UK HEs are likely to become members of a small number of Shibboleth federations.

HAKA - Finnish national higher education Shibboleth Federation

Identity Provider (IDP) - member institutions of a Shibboleth federation which represent the end users. Also software running at an IdP site that knows about the user, and can reply to requests for attributes or authentication from the SP (in older documentation known rather misleadingly as Shibboleth Origin) For instance a University could inform third parties whether an individual was a registered student at the University, a mobile phone company could verify that a given phone number belongs to John Smith who has paid all his bills.

InCommon - US national higher education Shibboleth Federation

Jakarta – a variety of Java tools supplied by Apache (<http://jakarta.apache.org/>) Tomcat and Ant were once developed as part of the Jakarta project.

jar files – Java Archive files. Libraries of Java code and other objects, in a format similar to zip files.

Java – computer language in which Shibboleth and most related software is written. (seeing as this is a list of jargon, Java is C-family, statically-typed, object-orientated, single-inheritance, multi-threaded, semi-compiled, virtual-machine, computer language)

JAXP - Java API for XML Processing.

JDBC "Java Database Connectivity" Java API to databases

JISC (Joint Information Systems Committee) UK government-funded organisation that provides advice about and support R&D for, ICT in further & higher education institutions. The JISC Core Middleware Initiative "aims to improve the way in which users access resources throughout the UK educational sector" and funds projects in infrastructure s related to inter-institutional co-operation, such as access and authentication.

JNDI "Java Naming and Directory Interface" Java API that provides common programming tools for use in various directories (including LDAP)

JSP (Java Server Pages) Specification for producing dynamic web pages in Java

JSTOR - US-based organisation that provides access to online archives of images of many scholarly journals. Birkbeck subscribes to JSTOR using AthensDA authentication, so Birkbeck users who have signed on to the local network at College, or to our AthensDA login page, can freely access JSTOR.

J2EE – (Java 2 Enterprise Edition) a standard environment for provision of Java web services. Tomcat is the most common instantiation of J2EE.

LDAP - "Lightweight Directory Access Protocol" - standard methods for accessing directory information over the Internet, by far the most common Internet version of the OSI x.500 family of directory standards. Windows Active Directory is an implementation of LDAP.

LEAP is a consortium of London colleges (Birkbeck, Imperial, King's, LSE, Royal Holloway, SOAS, UCL) which implements a shared infrastructure for eprints online archives.

MATU - Middleware Assisted Take-Up Service - provides support for early adopters of Shibboleth and related software, managed by EduServe and funded by JISC. See <http://www.matu.ac.uk/> for useful information on Shibboleth and access management.

Metadata - in Shib its basically what your computer needs to know to be part of a federation - things like the URLs of IdPs, SPs, WAYFs.

OASIS (Organization for the Advancement of Structured Information Standards) - an international consortium that promotes e-business standards. Once known as SGML Open.

Realm – a set of usernames and passwords identifying valid users of a web application, together with a list of roles associated with each valid user.

Relying Party - for the IdP, the application is the "relying party", i.e. something that needs our input to get its job done. For the SP, the IdP is the RP.

SAML (Security Assertion Markup Language) - OASIS standard XML scheme for messages containing information about access control and authorisation.

SCI (Science Citation Index) Bibliographic data, abstracts of scientific publications. One of many commercial information services provided by Thomson's. An example of an academic resource protected by Athens and likely soon to be available with Shibboleth.

SDSS - development UK national Shibboleth Federation managed by EDINA

Self-archiving - authors depositing their own work to some online repository

Service Provider (SP) an institution providing resources and services which they wish to offer to the member organisations; or software running at the site that provides access to the resources. SP uses Shibboleth to get authentication information from an IdP.

Servlet - Java application that runs in a Web server

Servlet container - application server that provides an environment in which Java servlets can run. In this case Tomcat

SHERPA - consortium of UK universities establishing an open access institutional repositories. "Securing a Hybrid Environment for Research Preservation and Access". LEAP, the White Rose Partnership (Leeds, Sheffield, and York), and the Arts and Humanities Data Service, are co-participants in SHERPA with universities of Birmingham, Bristol, Cambridge, Durham, Edinburgh, Glasgow, Newcastle, Nottingham and Oxford.

Shibboleth - a JISC-funded project initiated by LSE to implement Shibboleth as the authentication protocol for LEAP, and potentially to leverage the experience gained to "Shibbolethise" some other online resources. The original SHERPA-LEAP and Shibboleth partners were:

- Birkbeck College
- Imperial College London
- King's College London
- London School of Economics & Political Science (LSE)
- Royal Holloway
- School of Oriental & African Studies (SOAS)
- UCL

Shibboleth (<http://shibboleth.internet2.edu/http://shibboleth.internet2.edu/shib-intro.html>) - a web-based protocol for exchanging authentication and access management information about users - in this case we are using it to support a distributed authorisation system for a version of eprints.

Shibboleth Origin - old name for Identity Provider

Shibboleth Target - old name for Service Provider

Shibbolisation - converting existing web applications to Shibboleth

SOAP (Simple Object Access Protocol) - XML standard sending requests to and from web services. http://ws.apache.org/soap/faq/faq_chawke.html

SPIE (Shibboleth-aware Portals and Information Environments) JISC-funded project at Oxford and Bristol intended to demonstrate integration of local and other Shibboleth environments including Athens and the Permis authorisation software.

SSL (Secure Socket Layer) protocols to allow web browsers and web servers to communicate over a secured connection. The server authenticates itself to the user by presenting the browser with credentials such as a digital certificate. Client authentication is also possible but rare. Both ends encrypt all traffic before sending out data.

SSO (Single Sign-On) allow a user to sign on once using a set of credentials and then be permitted access to a variety of computer & network resources without the need to authenticate themselves again. On the web SSO is often provided by an LDAP directory with a web front end with cookies to record information about the signed-in user.

SWITCHaai - Swiss national higher education Shibboleth Federation

Tomcat - web application server ("servlet container") that provides an environment for Java applications ("servlets") and can itself run under the Apache web server (<http://tomcat.apache.org/>)

Virtual organisations (VO) groups of individuals from multiple organisations who want to collaborate in some way and set up a shared computing environment of some sort.. Often used when talking about Grid computing.

WAD (Windows Active Directory) Microsoft's implementation of LDAP, containing the user directory and software and network configuration objects for Windows domains.

WAYF ("Where are you from?") software at the service provider site that finds out which IdP to use for a user. For example a Birkbeck eprints user would choose Birkbeck, and Shibboleth would pass the user to Birkbeck's IdP for authentication.

WebAPP - a web application is an application delivered to users from a web. Exploits web browsers as ubiquitous thin clients.

WebISO "web initial sign-on" systems allowing web browsers to authenticate to web-based services across many web servers, using a standard, typically username/password-based central authentication service.

WSDL (Web Service Description Language) XML format for specifying the interface to a web service, such as methods and SOAP endpoints

X.509 certificates - unique data object which uses strong cryptography to associate an individual, a server, a company, or some other entity, and to associate that identity with a public key. When held within web servers they are often use to authenticate a computer or program to the user (typically a web-browser establishing an SSL connection), for example