TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Shibboleth and the challenge of authentication in multiple servers on a e-learning environment

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Humberto Gonzalez Gonzalez

DIRECTOR: J. Manuel Yúfera / Michael Praetorius

DATA: 18 de gener de 2006

Títol: Shibboleth and the challenge of authentication in multiple servers on a e-learning environment

Autor: Humberto Gonzalez Gonzalez

Director: J. Manuel Yúfera / Michael Praetorius

Data: 18 de gener de 2006

Resum

L' objectiu d'aquest treball és l'estudi, implementació i prova d'un sistema de autentificació compartida per a múltiples servidors. Encara que des d'un principi es sabia que es treballaria amb Shibboleth també s'han tingut en compte altres possibles solucions. Shibboleth és un projecte desenvolupat per els membres de les universitats que formen el consorci Internet2 amb l' objectiu de desenvolupar un nou middleware per a realitzar les funcions d'autentificació compartida en múltiples servidors i pensat específicament per facilitar la col·laboració entre institucions i l'accés a continguts digitals.

Shibboleth és una solució complerta ja que contempla des de l'autentificació , autorització i accounting, fins al sistema de login i els atributs a emprar. La qual cosa fa que es converteixi en un entorn de treball molt segur però amb l'avantatge d'aportar privacitat als usuaris.

El primer objectiu ha estat identificar les peculiaritats i requeriments dels entorns de e-learning distribuïts, per això s'ha estudiat conceptes específics de seguretat així com la manera d'adaptar-los a l'entorn requerit. Després s'ha fet una comparativa de les solucions existents al mercat amb una funcionalitat similar a Shibboleth, per tal de presentar els avantatges i desavantatges de Shibboleth vers aquests.

Posteriorment, el treball ha consistit en entendre la estructura i els principis de funcionament de Shibboleth, quin tipus de requeriments tenia, el funcionament i objectius de cada part, estudiar els requeriments de l'entorn específic per al qual ha estat dissenyat (e-learning) i donar una idea general de com s' hauria de fer la implementació. També s'han estudiat totes les tecnologies i requeriments necessaris per desenvolupar Shibboleth.

Una vegada estudiat Shibboleth i l'entorn específic en el que s'hauria d'integrar, s'ha muntat un escenari per a la posada en marxa i proves d'aquest, provant específicament cada part i entenent amb les proves reals el funcionament. Amb l'escenari en funcionament, la idea era integrar Shibboleth amb Sakai i Blackboard, els CMS (Course Management System) utilitzats a on-campus, el campus virtual de la Fachhochschule Lübeck.

Per a finalitzar i a mode de conclusions s'ha fet una petita explicació dels resultats obtinguts, una valoració de com Shibboleth resoldria les necessitats plantejades i algunes propostes de millora.

Title: Shibboleth and the challenge of authentication in multiple servers on a e-learning environment

Author: Humberto Gonzalez Gonzalez

Director: J. Manuel Yúfera / Michael Praetorius

Date: January, 18th 2006

Overview

The subject of this work is a study, implementation and test of a shared authentication in multiple servers system. Although it was thought that Shibboleth would be the final studied and deployed system, other solutions were considered too at the beginning.

Shibboleth is a project developed by members of universities participating in Internet2 partnership with the objective to develop and to deploy new middleware. This middleware facilitates the authentication functions of shared resources in multiple servers and makes easier the collaboration between institutions and access to digital contents and e-learning environments in university surroundings. Shibboleth is a complete solution because it enables the authentication, authorization, accounting, system login and the attributes to use. It makes the scenario very secure without compromising the privacy of the users.

The first part of the work explores the peculiarities of distributed requirements surroundings of e-learning environment, so specific concepts of security as well as the way to apply them were studied.

The next part of the work was a comparison among Shibboleth and other solutions available in the market and in this way the advantages of Shibboleth are described. After that, the work shows the structure and the main issues of Shibboleth and its operation and objectives. The surrounding requirements were also studied, specifically those used in e-learning, and a general concept of the work and of the basis of the technologies implied in Shibboleth were also considered.

Once the technical part was studied and documented, an scenario to test the system was constructed. Each part was proved, and when the basic example proves finished some CMS existing in the market were integrated to the project too. Basically Sakai and BlackBoard, the ones used in on on-campus (the e-learning environment at the Fachhochschule Lübeck) and Moodle as new proposal of use.

The last part is a conclusion and evaluation of how to deal with Shibboleth needs and some improvement suggestions.

Dedico aquest treball a tota la gent que ha estat al meu costat durant aquests anys de estudi, als pares, als amics, als companys i al grup de rdv.

També a tothom que m'ha ajudat en la elaboració de aquest treball tant a Lübeck com a Barcelona.

Per finalitzar, no em voldria oblidar de la gent que d'una manera o un altre col·labora en la elaboració i difusió del programari lliure.

ZUSAMMENFASSUNG

Die Zielsetzung dieser Arbeit, ist die Studie, die Implementierung und der Test einer geteilten Authentisierung im mehrfachen Bedienersystem. Obgleich vom Anfang ich weiß, dass die Zielsetzung Shibboleth sind, auch andere mögliche Lösungen haben betrachtet.

Shibboleth ist ein Projekt, das von den Mitgliedern der Universitäten von der Teilhaberschaft Internet2 mit der Zielsetzung, um neues middleware zu entwickeln entwickelt wird und zu entfalten, um Funktionen von Authentisierung geteilten Betriebsmitteln in den mehrfachen Bedienern und hauptsächlich für Universitätsumlagerungen und vom e-Lernen mit der Zielsetzung zu bilden, eine einfache Weise zu verbessern, die Zusammenarbeit zwischen Anstalten und dem Zugang zum digitalen Inhalt zu handhaben gedacht ist. Shibboleth, ist eine komplette Lösung, da es von der Authentisierung erwägt, Ermächtigung und Acounting zum System LOGON und zu den Attributen, um zu verwenden also bildet das Drehbuch, das sehr ohne das Privatleben der Benutzer sich zu vergleichen gesichert wird.

Der Erste Schritt der Arbeiten, die, die Eigenheiten und die Anforderungen der verteilten Umlagerungen des e-Lernens, für dieses zu kennzeichnen seiner sind, habe ich spezifische Konzepte der Sicherheit sowie die Weise, sie an den erforderlichen Umlagerungen anzuwenden studiert.

Der Folgende Schritt, obgleich ich freies dass die Software habe, die, alle Erwartungen zu erfüllen Shibboleth, da ein Vergleich diese Software/vorhandenen Lösungen im Markt studiert haben, haben eine Funktionalität waren, die Shibboleth und sich die diese Anzeige darzustellen/Vorteile von Shibboleth ähnlich ist. Später hat die Arbeit bestanden, die Struktur und die Grundregeln von Shibboleth zu verstehen, denen Art von Anforderungen a, den Betrieb und Zielsetzung jedes Teils haben, um die Anforderungen der Umlagerungen zu studieren, die ich spezifiziere für, welches gewesen es entwerfen lässt (e-erlernend) und ein allgemeines Konzept sollte von, wie man zu bilden arbeiten ist. Auch ist im Detail allen Technologien studiert worden, auf denen Shibboleth basiert.

Einmal studiertes Shibboleth und die Umlagerungen, in denen sein muss integtated, entfalte ich ein scenerio für den Anfang und die Tests von Shibboleth und prüfe spezifisch jedes Teil und das Verstehen auf realen Tests des Betriebes, prüfen einmal in einer realen Szene, ist der Folgende Schritt gewesen, wie die Integrierung irgendeines CMS von das Bestehen im Markt zu studieren, damit es mit Shibboleth funktionierte, im Allgemeinen ich studieren Sakai und Tafel, welches sie die sind, das in on-campus des Fachhochschule Lübeck verwendet wird.

Das Letzte ist Zusammenfassungen ist geworden eine Auswertung von, wie man die Shibboleth Notwendigkeiten und einige Anträge der Verbesserung löst.

INDEX

LIST OF FIGURES

INTRODUCTION

The objective of this work is the study and implementation of a shared authentication system for Web servers. The aim is to fulfill the on-campus (e-learning environment of the Fachhochschule Lübeck) requirements, the objective is to design a system that avoids the multiple actions of login for the student when access to content distributed across different servers.

With this objective, a first study of the specific requirements of e-learning environments, and also of some necessary concepts of security, needed for the Shibboleth correct operation, was made.

First of all, and although Shibboleth had already been chosen as the definitive solution, a comparative among different likely systems has to be made. After that, a study of the individual components of Shibboleth, the ways to integrate them, and the software needed to make it work, has to begin.

Once finished the Shibboleth theoretical knowledge and before start with the practical work, Is also necessary to study the requirements that Shibboleth needs to work, the software under which works and some concepts and technologies in which is based.

The next step in the project is to implement, in a practical way, the knowledge acquired in the early stages, materializing it all in a physical scenario to develop each part of Shibboleth. When a simple example of authentication has been implemented, and works properly, the study of the CMS (Course Management System) would begun. With the idea to integrate Shibboleth with it.
A study will become of some CMS in particular and the way of Shibbolize (integrate the authentication system so that it works with Shibboleth) Sakai and Blackboard, the CMS used in on-campus (The e-learning environment on the Fachhochschule Lübeck)

The next part of the introduction talks about how this paper is organized.

The first chapter is an introduction to the e-learning environments, to their peculiarities and requirements. It also explains basic subject security concepts, like federation. It also shows the main concept to work at: the authentication based on attributes.

The second chapter is an overview of the existing solutions for the exposed requirements and a comparison of Shibboleth with other systems.

In the third chapter there is an explanation and a complete perspective of Shibboleth: basic concepts, development facts and procedures to explain every part of the scenario in depth. And also some of the goals that Shibboleth has to fit in the proposed solution are explained.

The fourth chapter presents a perspective of all software and concepts based on Shibboleth. Mainly it is an introduction to the concept and later how Shibboleth is used .

An exhaustive explanation of the implementation in the chosen scenario can be found in the fifth chapter.

The sixth chapter explains how to use Shibboleth as a system of authentication in CMS (Course Management System) and a guide for the implementation.

The last chapter, the seventh, shows the general conclusions of the work as well as some improvement proposals for development.

In the appendixes, there is a glossary with some of the abbreviations used in this paper and a short description of some concepts to use as reference. As well there are a explanation about an example of Shibboleth in action and concluding the appendixes there are some explanations about how to install the software.

CHAPTER 1: THE FACT. SHARED AUTHENTICATION

1.1 Introduction

Shared authentication is the fact of once the user is authenticated in the system and verified the privileges, that although this user access to another resources and the new server not require again him to authenticate. The idea is the servers interchange some attributes and based on some polices grants the access to the resources.

Authorization mechanisms require an authorizing agent to query some type of database to check the privileges of the concerned user. Nevertheless, in a panorama of the cross-domain, this requirement can be heavy. In the first place, given the number of users in the several domains, the great data bases would be unmanageably. Secondly, the users and the administrators of system would be equally reluctant to have sensible information of the user stored in so many diverse locations. This makes a new mechanism necessary to handle with effectiveness the authentication and the authorization of the cross-dominion.
Authorization based in attributes, whereas not a new concept, are winning quickly in reputation.

Attribute based authorization essentially considers more granularity in the process of the authorization, because one can choose what information to assert, and considerably simplifies the tasks of the administrators of the network since they don't have to maintain accounts for foreign users. Basing decisions of the authorization on the qualities or the “papers” of the users, the lists of complex control of access are needed not more for each resource. Everything is required that it's a system of mappings simple that they contain the several respective papers and their privileges. Thus, also primary targets are to avoid copies of the file of passwords between different servants and to guarantee the safe access to the resources without invading the privacy of the user.

1.2 The new scenarios (e-learning, course management systems, libraries, etc.)

Every time more the education, remote or face to face, it's nourished of contents on-line and these normally aren't centralized, the necessity arises to regulate the access to the information is of the own institution or as which they serve firstly like support to teaching because sometimes they are resources with copyright and in other occasions for audit questions. For this reason, one of the recent fields of investigation is like allowing the user to access to all the necessary resources in an easy way (for example with same login and password) without the disadvantages of having to distribute files of password between all the servers.

Scholarship and higher education students increasingly depend on digital information, and the on-line sources that provide them, for research and teaching. These sources vary greatly in size, focus, function, and scope. Valuable teaching and research materials might be found in a dataset collection on a departmental web site, in a repository of images run by a university library, or in a licensed commercial database of journal articles. Large numbers of these data sources, often known as digital repositories, now exist, and a scholar is likely to require materials drawn from multiple repositories to support research or teaching on a particular topic.

There are also, a growing interest among academic institutions in collecting, preserving, reusing and creating value-added services from digital content produced in and for research, teaching and learning.

The emphasis on research outputs and collaboration, and distance, flexible and on-line learning, together with developments in information technology, has led to an increased awareness that the digital content being created by members of the academic community is an institutional asset. At the same time many academic libraries are responding to the challenges of new technologies by taking the opportunity to redefine their fundamental role in the creation, distribution and provision of access to information. Over the past decade libraries have moved almost completely towards a digital platform for management of the information (both print and electronic) that they acquire or subscribe to. They have built significant digital collections of material published by others, and they are increasingly producing new content themselves. Often this content originates from, or is the intellectual property of, their own institutions.

Some services may be made available to restricted user groups only, such as the database provided to medical students by the scientific library. On the other hand, the networking of institutions has lead to a situation where the service provider and the user are not from the same organization.

Teaching itself is increasingly supported by software applications, both to support distance education and to supplement traditional face-to-face instruction. In particular, many colleges and universities have deployed or are developing learning management systems, also known as “courseware,” to support instruction. These systems are often used to deliver information drawn from internal or external digital repositories. Smaller, specialized learning applications also take advantage of content from digital repositories, such as electronic course reserve systems, personal bibliographical databases, digital portfolio managers, and presentation and analysis tools.

To make the most effective use of digital content in teaching, learning applications need to be able to easily interoperate with digital repositories so that teachers and students can discover, access, view, quote, adapt, and evaluate appropriate learning material. Because this, grows the necessity of improve distributed learning where the resources, the students and the teachers are located in different server/organizations.

1.2 Security Concepts and the e-learning environments.

1.2.1 AAA. Authentication, Authorization and Accounting

Any discussion regarding access to and use of information resources must, at some point, consider the primary concepts of authentication, authorization and accounting.

-*Authentication:* Authentication refers to the process by which a user's claim to an identity is checked and verified.
In a e-learning environment, authentication is needed in order to guarantee that only registered students of the participating institutes can access the on-line courses.

-*Authorization:* Authorization is the process of giving someone permission to do something.
In a e-learning environment, after a successful authentication phase the student is given access to some of the resources and based on an on-line timetable the students can reserve timeslots for accessing certain parts of the application.

-*Accounting:* Accounting is the process which measures the resources a user consumes or plans to consume during his session. Accounting is used for authorization control, billing, trend analysis and capacity planning activities. Due to the limited resources, accounting has to be used for authorization control.
In a e-learning environment, this means that students cannot access the whole application at once but are just given access to a certain module based on their on-line reservations.

For the implementation of the AAA, in an e-learning environment is needed an application to manage this action, one of the solutions is the authentication and authorization infrastructures. The Authentication and Authorization Infrastructures are middleware systems, consisting of a set of protocols that enable the delegation of authentication and authorization issues to different instances. The infrastructures provide all the necessary mechanisms that let users, organizations, and resources collaborate in the system. An authentication and authorization infrastructure requires three entities in order to provide its functionality; Home organization, resources and users.

Home Organizations are universities or other entities, where the potential users (students or staff members) are registered. The home organization is responsible for the authentication of their respective users. The users possess at least one account at a home organization. Resources exist in two forms: either they are web-based, or they act as services for authentication and authorization infrastructure users. If a user wants to access a resource, the authentication and authorization component of the resource asks the user's home organization for the authentication. Upon successful authentication, the home organization sends back a set of attributes containing the information about the user's home organization role.

It is a precondition in authentication and authorization infrastructures that user information released by home organizations is reliable and accepted by the resource providers. Based on these attributes, the resource authorization system decides if the access should be granted or denied.

Authentication and authorization infrastructures bring advantages for all the involved entities in the process of accessing protected resources. The home organizations can increase the quantity of accessible resources by their members with a minimal overhead. The users take advantage of a simplified resource access procedure. The advantage for the resources consists in a decreased administration overhead (they do not need to create accounts for their users), and the reliable user information, since the attributes sent by the home organization are trustful.

An authorization process is based on the exchange of user authorization attributes in the form of name/value pairs. A policy which allows controlling this exchange of attributes is desirable. Some authentication and authorization infrastructures provide such a functionality which enables the users to make a trade-off between access and privacy. This mechanism is enforced by the Attribute Release Policy (ARP). In order to authorize users, resources need a policy which defines a minimal amount of attributes they must receive. This policy is called Attribute Acceptance Policy (AAP). Based on users attribute release policy and resources' attribute acceptance policy, users are able to access resources. Only users providing the required attributes are granted access to the resource.

Privacy is conceptually feasible, though impractical, as it would require isolating one's self from the world. Beyond such simplistic definitions, it seems prudent to shy away from making definitive statements, in that privacy has a very subjective nature. Privacy advocates, however, often appear to treat the subject in a similarly simplistic manner. Statements are made claiming some action will result in a loss of privacy. Yet, defining that loss is often avoided or subjectively stated.

Privacy is a function of the prevailing culture, social context, political context and the individual ideals. Concerns widen from purely physical equivalents (such as unauthorized distribution of personal photos), to include more intimate details of one's self. These changes speak in a different way, for in an electronic world one's identity is captured easily in reproducible bits of information.

Given the above discussion, we see difficulty in defining privacy without tying it to a specific attributes and individuals. In the anonymity discussion that follows, we describe a means of providing this connection.

1.2.3 Anonymity

Systems that allow users to control the amount of personal information (IP address, email address, physical address, real name, etc.) revealed to different entities on the Internet can be used to reclaim an individual's on-line privacy. We consider two forms of on-line privacy, anonymity and "pseudonymity" (although in practice both are normally referred to as anonymity). A system that provides anonymity hides the user's identity unless the user chooses to reveal their identity. "Pseudonymity" is a type of anonymity; however, with "pseudonymity" the user maintains one or more distinct identities (pseudonyms) that are not connected to the user's physical identity.

Most systems use some secret that only the user who created the pseudonym knows to ensure that people with whom the user interacts using a given “nick” can be assured that, although they do not know the physical identity behind the “nick”, it's in fact the same person each time. However, more than one person may “share” a single “nick” simply by sharing the secret. In contrast, anonymous systems that provide strong or unlinkable anonymity do not leave any persistent information that lets someone link any transaction to another transaction preformed by the same user.

1.3 Federations

Federation means a group of organizations or service providers which have built trust among each other and enable sharing of user identity information amongst them. One proposal is a flexible approach to establish a Single Sign-On (SSO) ID in the federation. Then show how a user can leverage this SSO ID to establish certified and un-certified user identity attributes without the dependence on PKI for user authentication. This makes the process more usable and privacy preserving. The figure **(Fig. 1.1)** presents and overview of a Federation concept. The basic purpose is the access possibility to any resource (hosted in SP) inside the Federation using the authentication service (IdP) from any of the Institutions that belongs to the Federation.

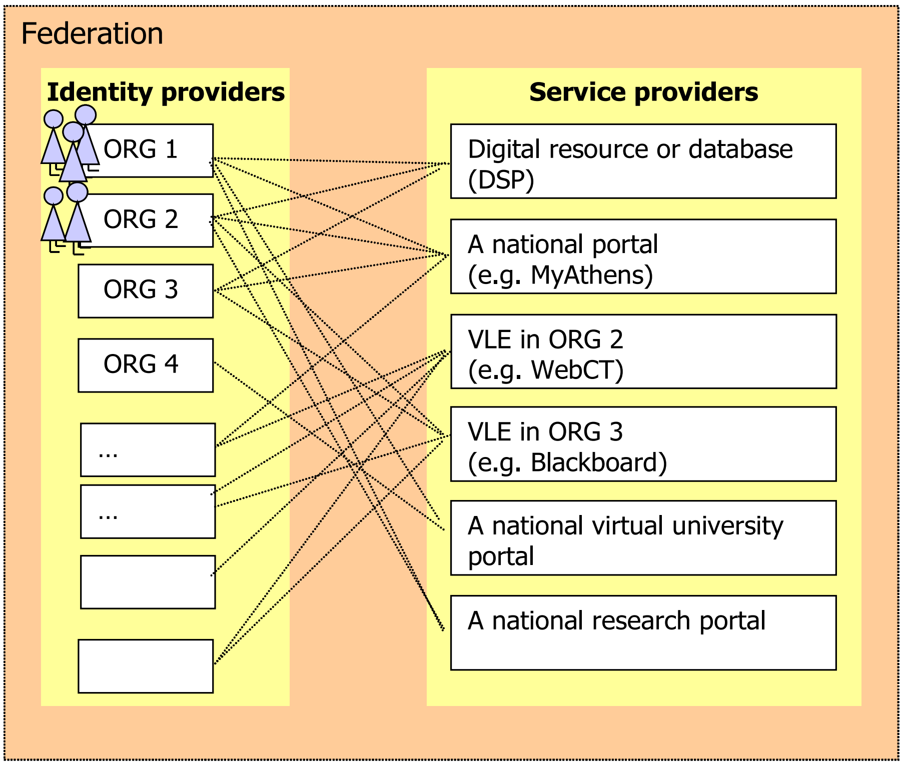

Federation
Identity providers
ORG 1
ORG 2
ORG 3
ORG 4
...
...
Service providers
Digital resource or database (DSP)
A national portal (e.g. MyAthens)
VLE in ORG 2 (e.g. WebCT)
VLE in ORG 3 (e.g. Blackboard)
A national virtual university portal
A national research portal

Fig. 1.1: Overview of the Federation Concept

Practical applications of federated identities are represented by large multinational companies which have to manage several heterogeneous systems at the same time. An effort in this sense is represented by the notion of SSO, which enables a user to login to multiple organizations or Service Providers (SP's) by using the same Username and password. This approach increases usability and adds security by reducing the number of passwords that need to be managed. Emerging standards are currently extending the notion of federated identity to other user information referred to as identity attributes. The main goal of such extensions is to enable interoperability and link together redundant user identities maintained by different SP’s. An important requirement in this context is that the federation environment should enable SP’s to exchange user data in a secure and trustworthy manner while also enforcing the original privacy preferences of the user.

Current federation solutions are built on top of Secure Assertion Markup Language (SAML) specification which depends on Public Key Infrastructure (PKI) with additional trust relationships for it's security. As such, federations have to rely on PKI for exchanging data among SP’s, and between users and SP’s authentication. However, PKI has experienced numerous implementation problems because of it's technical complexities. It is also oriented towards strong identity granted through Registration and Certification Authorities, which is not always suitable for user privacy. Hence, the assumption of relying on PKI for all types of interaction in the federation is not realistic. It's needed articulated identity solutions supporting multiple complementary options for digital identity.

A serious concern related with identity management, whatever solution is chosen, is the risk of identity theft. Despite guidelines have been provided on how to protect against identity theft, not many identity theft protection solutions have been proposed so far. Sensitive information in the Internet is currently hard to track and also consistent usage of the proposed solutions is extremely hard to achieve. In a federation environment it's possible to develop protocols able to achieve identity theft protection. As noted above, the security and privacy of the user identity information, both certified and uncertified, are of utmost importance today. Security prevents theft and impersonation when the identity attributes are used and privacy protects against the disclosure of identity when the user has the right or expectation of anonymity.

If are expected to work with federations, two criteria for trustworthy attribute assertions by Credential Providers are: that the identity management system fall under the purview of the organization’s executive or business management, and the system for issuing end-user credentials (ex. PKI certificates, user-id's/passwords, Kerberos principals, etc.) specifically have in place appropriate risk management measures (for example authentication and authorization standards, security practices, risk assessment, change management controls, audit trails, etc.).

At last but not at least, it's important to remark that one institution or institutions can be member of more than one Federation.

1.4 Why shared/attribute based authentication.

One way of forming a federation is to assert an attribute and have this attribute examined by the authority of the remote resource. Attribute based authorization entails just such an assertion by an authorization service of attributes associated with an identity. These attributes describe the 'role' or 'roles' of the identity of member at a university. An assertion for that principal identity might state that they have the ‘role’ of a faculty member. This allows role-based authorization to offer a very compelling privacy and anonymity solution. Identity becomes one more attribute of an assertion that may or may not be shared with various destinations.

CHAPTER 2: OVERVIEW OF SOLUTIONS

2.1 Introduction

Although from the beginning already was determined that Shibboleth is the technology to use, I want to compare Shibboleth with other existing solutions in the market. Thus for example Kerberos and the Single Sign-On (SSO) solutions. Kerberos is a solution based on tickets; the main idea is that when the user tries to accede to a resource, this it's turned aside to a server in whom after login and password provides ticket to him valued to accede to the resource during an assigned period. Kerberos is a solution very used and that has demonstrated it's functionality during long time. The other possibility is the Single Sign-On systems, (Shibboleth incorporates an SSO), The operation principle is easy: the authentication systems interchange information among them of a safe way so that it's not necessary that they are sent literally the login and the password.

2.2 Kerberos

Kerberos is a system for authentication in distributed computer systems. It provides strong authentication of clients and server using conventional cryptography. It also protects the confidentiality and integrity of communications over an insecure network. Kerberos operates on the application layer, providing end-to-end security.

The Kerberos system was developed at Massachusetts Institute of Technology as a part of project Athena. One implementation of Kerberos is freely available from MIT, and Kerberos has also been integrated into commercial products such as Windows 2000. Kerberos is actively developed; the current version is version 5.

Kerberos is designed to operate in an environment where the network itself and the workstations are not trustworthy. An adversary is assumed to be able to read, insert and modify packets on the network. The Kerberos architecture is designed around a data element called a ticket.

The ticket is a set of cryptographically sealed data that lets a client authenticate to a server. The Kerberos system uses two trusted on line services. These services are the authentication service and the ticket-granting service.

The Authentication Server (AS) implements the authentication service, and is responsible for authenticating users and giving them tickets usable for contacting the Ticket-Granting Server (TGS). The ticket-granting server implements the ticket-granting service, and is responsible for issuing tickets usable with servers implementing other services on the network. The combination of the AS and the TGS is called the key distribution center (KDC). This figure **(Fig. 2.2)** shows the different stages of the authentication process.

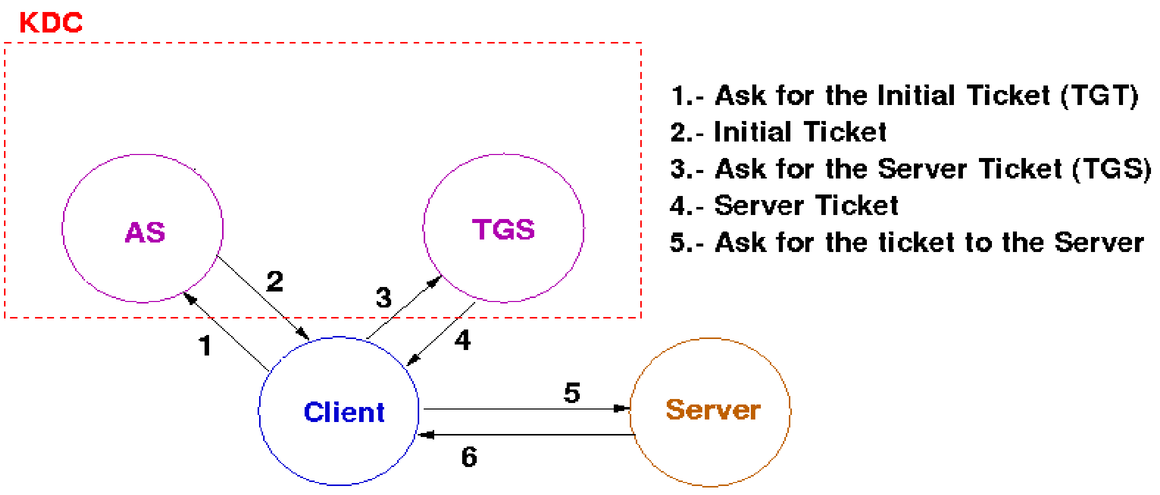

KDC
AS
TGS
Client
Server
1
2
3
4
5
6
1.- Ask for the Initial Ticket (TGT)
2.- Initial Ticket
3.- Ask for the Server Ticket (TGS)
4.- Server Ticket
5.- Ask for the ticket to the Server

Fig. 2.2: Kerberos authentication process

Both the TGS and the AS are trusted services, and are assumed to be physically secure. Each organizational unit has one TGS and one AS. Each such unit is called a realm or domain. Kerberos supports authentication of users across realm boundaries. The realm of a user belongs to is a part of the user's name. Trust between realms can be established through inter-realm keys.

One of the disadvantages of Kerberos and the main disadvantages compared to Shibboleth is that Kerberos needs a centralized organization that manages the tickets and because this, it does not fulfill the requirements.

2.3 Single Sign On Solutions

Another solution is a Single Sign-On System, this minds that the user only are asked for a login the first time that they enters in the system. Actually, most of the times when the user enter in the system are required to authenticate them, which is usually done by giving the user an account with an associated user name and password. The website could be an on-line shop where users need to be authenticated to see their current orders. Other examples of websites could be: news sites, game sites or libraries. This means a user could, potentially, have lots of accounts, each on different websites. It is tedious for a user to remember many user names and passwords, and often a user will resort to using only a single user name and password for every website.

The basic idea is the connection of the authentication mechanism of all the servers, this minds they are allowed to interchange some attributes and allow the access to the resources.

From a security point of view, this is not a wise strategy. If a single login is in some way cracked and thereby compromised, all logins are compromised. Furthermore, it has a cost to the websites to manage the many accounts for their users. Each website needs their own system to issue new accounts, remove unused accounts, handle lost passwords and other administrative tasks associated with the accounts. Also the accounts are often created ad-hoc, websites cannot truly rely on the identity information stated, when an account is created. Websites could be stricter and require valid identity information, but as it could exclude or scare some of their users this is often not an option.

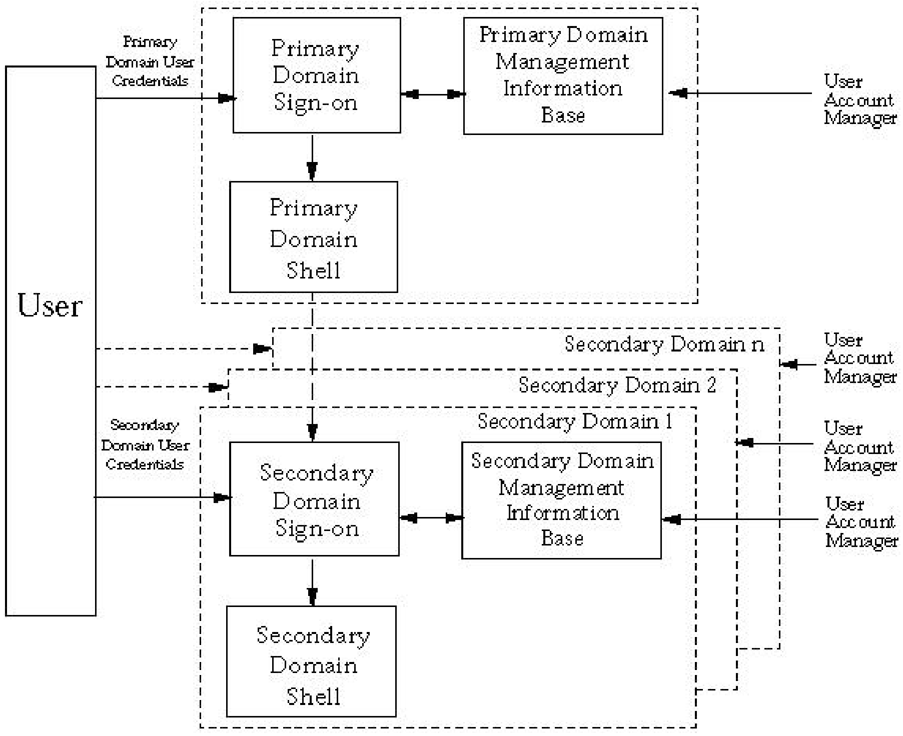


Fig. 2.3: SSO Overview

These problems have motivated the development of new ways to authenticate users, of which Single Sign-On protocols are one. The purposes of these protocols are to allow users to authenticate themselves to several websites using one account **(Fig. 2.3)**, and if visiting more websites simultaneously or in quick succession, a single log on information is passed thought the servers. At the same time, when using a Single Sign-On protocol, each website does not have to manage accounts for users. Account management can be carried out at a central authentication site. As the users only have to use one account it should be easier to use and remember the associated password. Using a Single Sign-On protocol the user only should create one account, to authenticate themselves to the authentication site used by the websites providing the services.

The Internet2 consortium, have a group to develop SSO solutions called WebISO. The WebISO Working Group is investigating the realm of "web initial sign-on" (WebISO) packages: systems designed to allow users, with standard web browsers, to authenticate to web-based services across many web servers, using a standard, typically username/password-based central authentication service. The objective is to share experience in this field and work towards in common solutions. The group output may take the form of: recommendations for best practice and architecture; recommendations for interfaces between services and WebISO infrastructure; common WebISO protocols and/or common implementations. Web ISO is not software, are only recommendations and drafts to developers like the Pubcookie project. One of these recommendations is the Shibboleth projects that include a SSO and some other features.

2.3.1 Shibboleth

Shibboleth is a project of Internet2/MACE with the aim of develops architectures, policy structures, practical technologies, and an open source implementation to support inter-institutional sharing of web resources subject to access controls. In addition, Shibboleth will develop a policy framework that will allow inter-operation within the higher education community. Key concepts within Shibboleth include:

> -*Federated Administration:* The Identity Provider (origin) campus (home to the browser user) provides attribute assertions about that user to the Service Provider (target) site. A trust fabric exists between campus, allowing each site to identify the other speaker, and assign a trust level. Identity Provider sites are responsible for authenticating their users, but can use any reliable means to do this.
>
> -*Access Control Based on Attributes:* Access control decisions are made using those assertions. The collection of assertions might include the identity, but many situations will not require this (ex. accessing a resource licensed for use by all active members of the campus community, accessing a resource available to students in a particular course).
>
> -*Active Management of Privacy:* The Identity Provider (origin) site, and the browser user, control which kind of information is released to the Service Provider (target). A typical default is merely "member of community". Individuals can manage attribute release via a web-based user interface. Users are no longer at the mercy of the target's privacy policy.
>
> -*Standards Based:* Shibboleth will use OpenSAML for the message and assertion formats, and protocol bindings which is based on Security Assertion Markup Language .
>
> -*A Framework for Multiple, Scalable Trust and Policy Sets (Federations):* Shibboleth uses Federations to specify a set of parties who have agreed to a common set of policies. (A site can be in multiple Federations.) This moves the trust framework beyond bi-lateral agreements, while providing flexibility when different situations require different policy sets.

-*A Standard Attribute Value Vocabulary:* Shibboleth has defined a standard set of attributes; the first set is based on the eduPerson object class that includes widely-used person attributes in higher education. It's not a closed list, can be possible add new attributes.

When a user from one institution tries to use a resource of another, Shibboleth sends attributes about the user to the remote destination before login the user. The destination can use the attributes to decide whether or not grants the user access. Shibboleth lets each user choose which kind of information about them can be released to every destination. In particular, a user may choose whether or not their name is sent to the remote site as an attribute, protecting privacy in scenarios where a users name is not necessary and with some other attribute, such as being affiliated with an institution, can be enough.

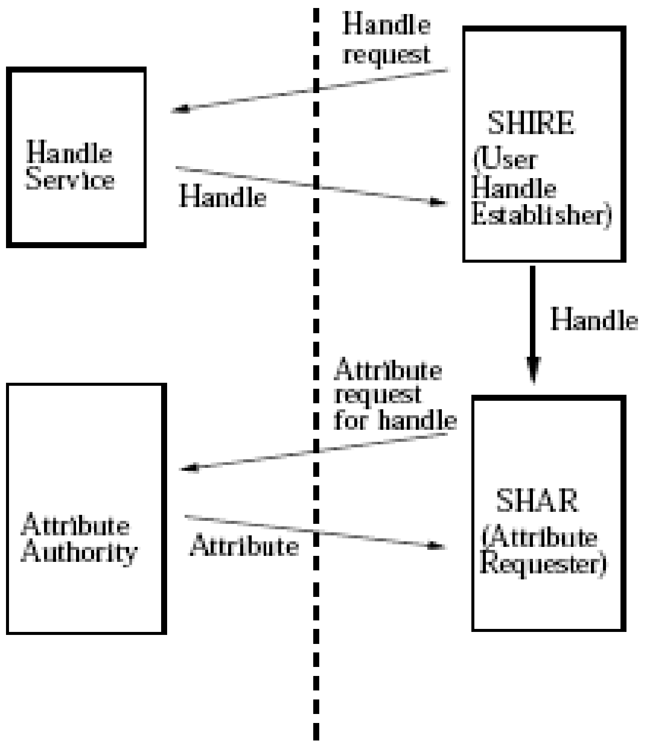

Handle request
Handle Service
Handle
SHIRE (User Handle Establisher)
Handle
Attribute request for handle
Attribute Authority
Attribute
SHAR (Attribute Requester)

Fig. 2.4: Shibboleth procedure

A simplified view of the Shibboleth authentication is shown in the figure **(Fig. 2.4).** On receiving a request, the target site establishes a handle for the user, then requests attributes for that handle, and makes an authorization decision based on the attributes passed to the Attribute Authority (AA) so it can decide which attributes to release, based on Attribute Release Policies. The attributes will likely be non-identifying, like confirmation of institution membership, or age, so the sites do not receive any personal information about the user. The AA at the SP however does see which sites (including URLs) are asking for attributes for the user, and if server at the AA wished to do so, they could log this information.

CHAPTER 3: SHIBBOLETH IN DEPTH

3.1 Introduction

Shibboleth is a system designed to exchange attributes across realms with the primary purpose of authorization. It provides a secure framework for an organization to send attributes using the web-browser across security domains to another institution. The first time, when the user attempts to access a resource in a domain outside the home institution, the user own home security domain sends certain information about that user to the service provider site in a trusted exchange. These attributes are used by the resource server to help determine if grants the user access. The user have the possibility to decide release specific attributes to certain sites by specifying personal Attribute Release Policies (ARP's), preserving the own privacy while still granting access based on trusted information. This drawing **(Fig. 3.5)** shows an overview of the Shibboleth working procedure. The Appendix C explains step by step working procedure with a real scenario.

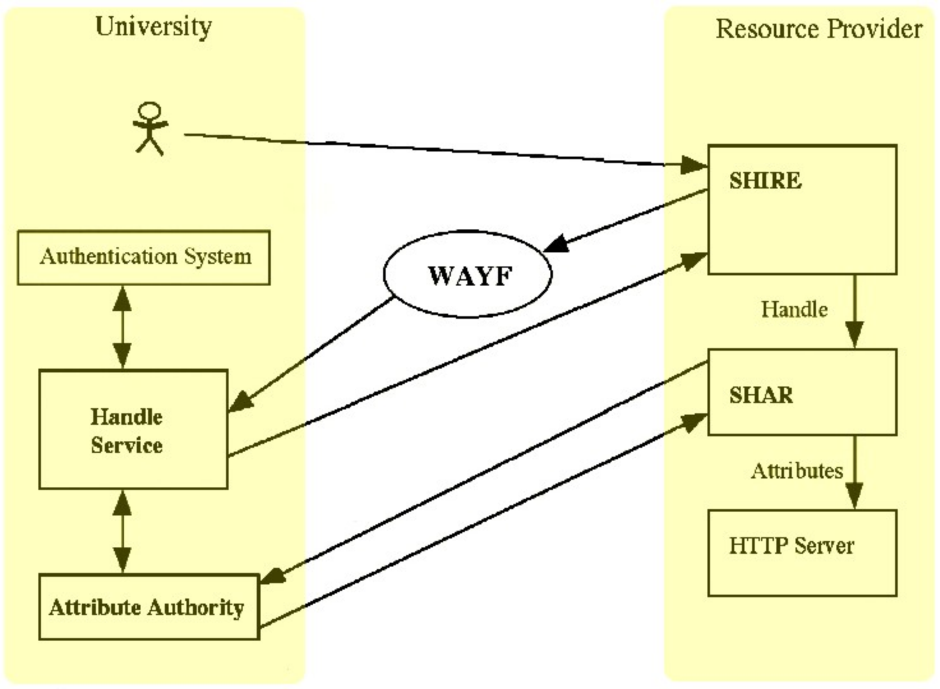


Fig. 3.5: Shibboleth working procedure

There are several controls on privacy in Shibboleth, and mechanisms are provided to allow users to determine exactly which information about them is released. A user's actual identity isn't necessary for many access control decisions, so privacy often is needlessly compromised. Instead, the resource often utilizes other attributes such as faculty member or member of a certain class. While these are commonly determined using the identity of the user, Shibboleth provides a way to mutually refer to the same principal without revealing that principal identity. Because the user is initially known to the service provider site only by a randomly generated temporary handle, if sufficient, the service provider site might know no more about the user than that the user is a member of the identity provider organization. This handle should never be used to decide whether or not to grant access, and is intended only as a temporary reference for requesting attributes.

3.2 History and development facts

Shibboleth is being developed by members of Internet2 with guidance from the Middleware Architecture Committee for Education (MACE) group of higher education architects and a major contribution of work from IBM. Alpha and beta development were spearheaded by Carnegie Mellon University, Ohio State University, and IBM.

The version 1.0 of the Shibboleth System was released in June 2003. As of this writing the current version, 1.3 (released in July, 2005) is in use or in test by more than 150 organizations, including universities, research labs, commercial service providers, and software vendors. Internationally, Shibboleth is deployed throughout Switzerland by SWITCH (the Swiss Education and Research network, EDINA in the UK, HAKA in Finland and many other university and research networks.

During the next year is expected to release the 2.0 version, the new version would be SAML 2.0-compliant and Single Sign-On logout support.

Several non Web-based projects such as instant messaging, peer-to-peer resource sharing, and grid systems are actively exploring Shibboleth integration. A joint effort with Microsoft is under way to provide interoperability with the IBM-Microsoft Web Services Security Model. Finally, Shibboleth is in the process of being certified for use with the U.S. Federal E-Authentication Initiative.

3.3 The actors in Shibboleth

3.3.1 IdP (Origin)

There are four primary components to the identity provider side **(Fig. 3.6)** in Shibboleth: the attribute query handler, the SSO handler, the directory service, and an authentication mechanism. The attribute query and SSO handlers are provided with Shibboleth, and an open-source WebISO solution, the directory is provided by the organization. Shibboleth is able to interface with a directory exporting a JDBC or JNDI interface containing user attributes, and is designed such that programming interfaces to other repositories is straightforward. Shibboleth relies on standard web server mechanisms to trigger local authentication. A <Location> block is used to trigger either an authentication mechanism such as local WebISO system, Kerberos, or the web server's own basic auth.

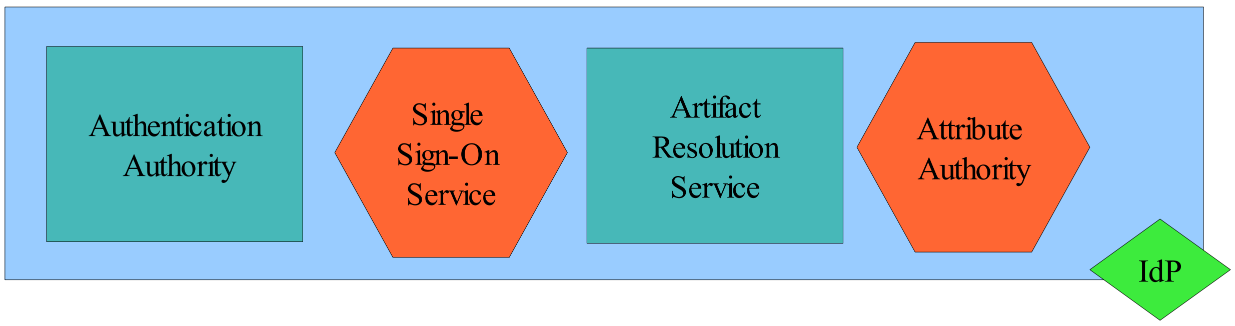


Fig. 3.6: Identity Provider block diagram

-Authentication Authority: The authentication authority issues authentication statements to other components. The authentication authority is integrated with the IdP's authentication service.

-Single Sign-On Service: A single sign-on (SSO) service is the first point of contact at the IdP. The SSO service initiates the authentication process at the IdP and ultimately redirects the client to the inter site transfer service. The inter-site transfer service issues HTTP responses conforming to the Browser/POST and Browser/Artifact profiles. The inter-site transfer service interacts with the authentication authority behind the scenes to produce the required authentication assertion.

-Artifact Resolution Service: If the Browser/Artifact profile is used, the IdP sends an artifact to the SP instead of the actual assertion. (An artifact is a reference to an authentication assertion.) The SP then sends the artifact to the artifact resolution service at the IdP via a back-channel exchange. In return, the IdP sends the required authentication assertion to the SP.

-Attribute Authority: The attribute authority processes attribute requests; that is, it issues attribute assertions. The attribute authority authenticates and authorizes any requests it receives.

From the identity provider site's point of view, the first contact will be the redirection of a user to the handle service, which will then consult the SSO handler to determine whether the user has already been authenticated. If not, then the browser user will be asked to authenticate, and then sent back to the assertion consumer service URL with a handle bundled in an attribute assertion. Next, a request from the Shibboleth daemon, shibd, will arrive at the attribute query handler which will include the previously mentioned handle. The IdP then consults the ARP's for the directory entry corresponding to the handle, queries the directory for these attributes, and releases to shibd all attributes the service provider is entitled to know about that user.

3.3.2 SP (Target)

A Service Provider (formerly a Shibboleth target) is a deployment of SAML software that validates assertions issued by Identity Providers and uses them to create a security context and assists in the enforcement of access control based on the information.

In Shibboleth, the Service Provider software is in C++ (and soon Java), and is provided as a set of libraries, services, and web server pluggins that implement the Shibboleth Specifications in a flexible way. A web server can use the software to handle user authentication and security session management for any web content and applications that it hosts.

A service provider (formerly called a target) **(Fig. 3.7)** manages secured resources. User access to resources is based on assertions received by the service provider (SP) from an identity provider. Note that the service provider has access control in place that prevents access by clients without a security context.

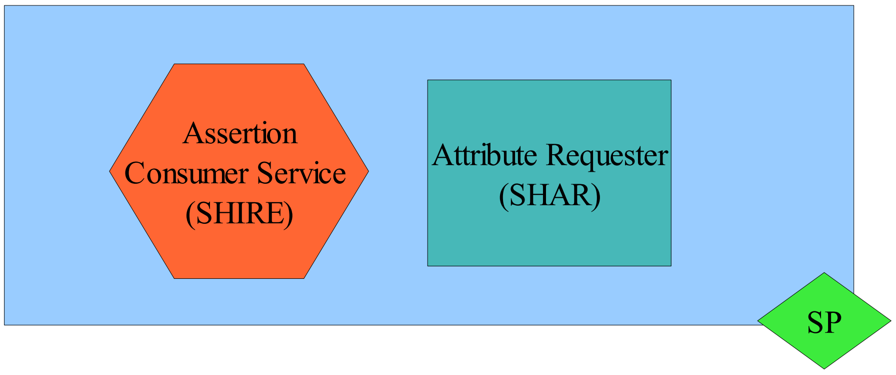

Assertion Consumer Service (SHIRE)
Attribute Requester (SHAR)
SP

Fig. 3.7: Service Provider block diagram

-Assertion Consumer Service: The assertion consumer service (formerly called a SHIRE) is the service provider endpoint of the SSO exchange. It processes the authentication assertion returned by the SSO service (or artifact resolution service, depending on the profile used), initiates an optional attribute request, establishes a security context at the SP, and redirects the client to the desired target resource.

-Attribute Requester: An attribute requester (formerly called a SHAR) at the SP and the attribute authority at the IdP may conduct a back-channel attribute exchange once a security context has been established at the SP. That is, the SP and IdP interact directly, bypassing the browser.

From the service provider's point of view, a browser initially makes a request for a Shibboleth-protected resource. The resource manager allows the service provider to step in, which will use the Were Are You From (WAYF) to acquire the name of an identity provider to ask about the user. The IdP will then reply with a SAML authentication assertion containing a handle, which the assertion consumer service then hands off to shibd. Shibd uses the handle and the supplied address of the corresponding attribute query handler to request all attributes it's allowed to know about the handle. The resource manager performs some basic validation and analysis based on attribute acceptance policies (AAP's). These attributes are then handed off to the application or used internally to decide whether to grant access.

3.3.3 WAYF and Federations

An optional WAYF service is operated independent of the SP and IdP. The WAYF can be used by the SP to determine the user's preferred IdP, with or without user interaction. The WAYF is essentially a proxy for the authentication request passed from the SP to the SSO service at the IdP.

The WAYF service can be either outsourced and operated by a federation or deployed as part of the service provider. It is responsible for allowing a user to associate themselves with an institution of their specification, then redirecting the user to the known address for the handle service of that institution.

A Shibboleth federation provides part of the underlying trust required for function of the Shibboleth architecture. A federation is a group of organizations(universities, corporations, content providers, etc.) who agree to exchange attributes using the SAML/Shibboleth protocols and abide by a common set of policies and practices. In so doing, they must implicitly or explicitly agree to a common set of guidelines. Joining a federation is not explicitly necessary for operation of Shibboleth, but it dramatically expands the number of service providers and identity providers that can interact without defining bilateral agreements between all these parties.

A federation can be created in a variety of formats and trust models, but must provide a certain set of services to federation members. It needs to supply a registry to process applications to the federation and distribute membership information to the identity provider and service provider sites. This must include distribution of the PKI components necessary for trust between identity providers and service providers. There also needs to be a set of agreements and best practices defined by the federation governing the exchange, use, and population of attributes before and after transit, and there should be a way to find information on local authentication and authorization practices for federation members.

3.4 Shibboleth goals

One of the goals of Shibboleth is protect user's privacy in a different ways. A user can choose which of their attributes can be released to any specific destination. This means that a user may choose to send only that they are a member of a certain class to a collaborational project, may choose to send their name to the research group site they belong to, and may also choose to release that they are a student to a digital library.

Frequently in current designs, identity is mapped backwards to determine attributes such as member of a particular working group, which is then used to govern access to resources. Shibboleth reverses this process, allowing attributes to be sent with an identity optionally included only if it's necessary. The target site will only know the attributes and information necessary to perform an access control decision, protecting users' anonymity in cases where their identity is not necessarily important. This gives users a large amount of control and flexibility
about how their attributes are released and known. Administration has the
capability to develop default guidelines for attribute release for users.

Another goal is that provide a way to exchange attributes in a secure environment. Shibboleth has been extensively designed to protect attributes while in transit from many potential attacks. Hosts in all communications are authenticated, and vulnerable transactions are protected using secure channels. Other techniques are employed to protect user privacy and defend against replay attacks.

It is important to note that Shibboleth does not provide any limitations on what the target can do with received attributes or what the origin can submit as a presumably accurate representation. Trust agreements are necessary to define the population, retention, and use of attributes out of band.

Also provide a useful federated administration. Federated administration is a way of making authentication, authorization, attributes, etc. useful to other domains . This allows for items of information that are established in one domain to be trusted in another domain based on the trust relationship between the two domains.

Doing so can reduce administrative burdens for all parties concerned without relying on a central authority or similar service to perform extensive operations. A registry service is often still needed to host agreements reached by the federation, trusted information about which members are in the federation, or who is authoritative for which entity.

And the last, Shibboleth emphasis on user privacy and control over information release. Shibboleth is a system for securely transferring attributes about a user from the user's origin site to a resource provider site. Again, Shibboleth allows users to determine what information is released about the user and to which site. Thus, the job of balancing access and privacy lies ultimately with the user, where it belongs. Shibboleth also keeps information local, meaning that a user does not have to worry (as much) about who has access to their user information and browsing behaviour.

CHAPTER 4: SHIBBOLETH. BASED CONCEPTS AND REQUIREMENTS

4.1 Introduction

Because Shibboleth is a Middleware, it needs layers on which base the work and facilitate that upper layers can work, it's why in this section are explained some basic concepts like HTTP, HTML Forms, SSL as well as applications on which work depend Shibboleth like SOAP, PKI, XML and SAML and finally some concepts of operation of Shibboleth like Attribute-Based Authentication and User attributes that although are not specific of Shibboleth, it's needed to explain.

4.2 HTTP

Found everywhere on the Internet, HTTP (HyperText Transfer Protocol) is a ubiquitous protocol for data connections between Web browsers and servers.
This protocol is the current standard for transferring HTML documents, although it's designed to be extensible to almost any document format like XML for example. HTTP Version 1.1 is documented in RFC 2068.

It operates over TCP connections, usually to port 80, though any other port can be used. After a successful connection, the client transmits a request message to the server, which sends a reply message back.
The simplest HTTP message is "GET url", to which the server replies by sending the named document. If the document doesn't exist, the server may send an HTML-encoded message stating this. This form of communication represents a typical request/response mechanism. A client sends a request for a specific document to the server and waits for a response. If the server does not respond with the requested document it's up to the client to wait for the time-out and request the same document again. This loosely coupled type of communication is very common in client-server architectures.

In addition to GET requests, clients can also send HEAD and POST requests, of which POST's are the most important. POST's are used for HTML forms and other operations that require the client to transmit a block of data to the server. After sending the header and the blank line, the client transmits the data.

This way Web services utilize the HTTP protocol to transmit both data payload and service request to a Web service.

4.3 HTML Forms

The main of the HTML Forms is submit a form to the destination specified by their action attribute. How the submission takes place is specified in the method and encoding type attributes. Most HTML forms use method="POST" to transmit their data, because method="GET" has a 255-byte restriction in the content size. Unfortunately most portal engines treat POST requests differently than GET requests.

GET requests are transmitted as name-value pairs encoded in URL parameters and can be extracted easily. These parameters can be forwarded to the remote site by constructing a new URL. Portals may use their own request parameters, like portal-request-counter. These portal specific request parameters must be stripped from the request sent to the remote site, as they may result in unexpected results.

POST requests usually contain the form data in the request body not in the URL, but exceptions are the norm. There are three cases of form definitions which need different treatments:

-Forms with a definition like <form action="mailto:..."> are email forms. The action URLs of these forms must not be encoded by the portal, because it's directly processed by the web browser which sends an email upon submission.

```
<form action="mailto:user@domain.com">
```

-Forms with a definition like <form method="POST" enctype="application/x-www-form-urlencoded"> must be treated like GET requests, because the browser encodes the form values into the request URL.

```
<form method="POST" enctype="application/x-www-form-urlencoded">
```

-Forms with an attribute method="POST" and without attribute enctype="application/x-www-form-urlencoded" are "regular" POST requests and require the request body to be forwarded to the remote resource by issuing a new POST request. POST requests can contain huge amounts of data that will add load to the server where the portal is running on.

Shibboleth uses the HTML Forms to transfer information between the different servers and clients.

The Secure Sockets Layer Protocol (SSL) is a protocol developed by Netscape designed to provide privacy between two communicating applications (a client and a server) by using public key cryptography. Second, the protocol is designed to authenticate the server and, optionally, the client. SSL requires a reliable transport protocol (for example TCP) for data transmission and reception.

An advantage of the SSL protocol is that it's application protocol independent **(Fig. 4.8)**. An application level protocol (for example HTTP, FTP, Telnet, etc.) can layer on top of the SSL protocol transparently. The SSL protocol negotiates an encryption algorithm and a session key as well as authenticates a server before the application protocol transmits or receives it's first byte of data.

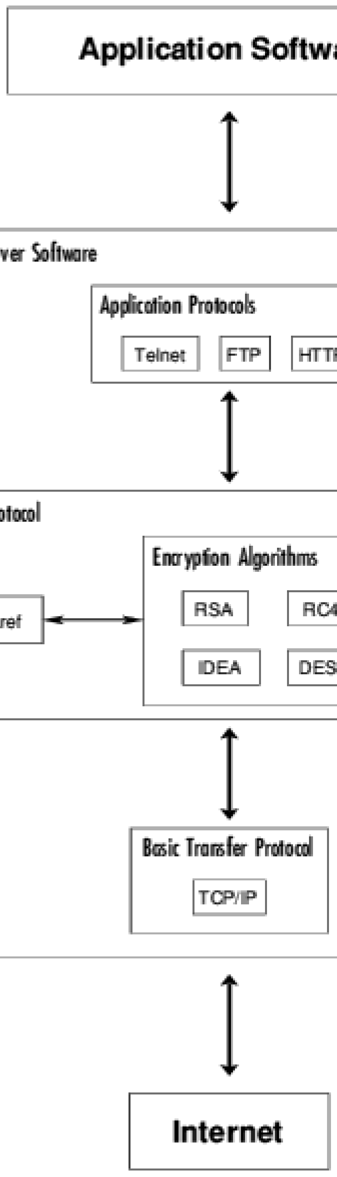


Fig. 4.8: SSL Details

All application protocol data is encrypted before transmission, ensuring privacy. The connection provided by the SSL protocol has three main properties:

-The connection is private. All messages are encrypted using secret key cryptography (for example DES, RC4, etc.) with a session key that is decided at the beginning with an initial handshake.

-The identities can be authenticated using public key cryptography (for example RSA, DSS, etc.). The server endpoint of the conversation is always authenticated, while client endpoint authentication is optionally.

-The connection is reliable. The protocol includes a message integrity check using a Message Authentication Code (MAC) ensuring that package alteration between client and server is detected. The MAC is calculated using secure one-way hash functions (for example SHA, MD5, etc.).

Transport Layer Security (TLS) is the latest enhancement of SSL. The TLS protocol is based on the SSL 3.0 protocol specification as published by Netscape. The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate. The major changes are cryptographically stronger MAC computation, larger padding (up to 256 instead of 63 bytes) and some protocol clean-up (for example ignoring unknown record types, improved alert messages, etc.).

The OpenSSL project offers an Open Source implementation of the SSL/TLS protocols. Shibboleth use SSL to transfer information between servers encrypted and protect the “sensible” information.

4.5 SOAP

SOAP is very new and there is not much history to it but I will go into how it was initiated and got the support of several big vendors in the business. SOAP began at Microsoft as XML-RPC, created by Dave Winer back in 1998. It was developed to replace the existing RPC's on the market that were not suited for use over the Internet.

Other protocols (such as DCOM) required a significant dedicated runtime support. They also had problems with several firewalls that didn’t support access via non- HTTP protocols. Later on they changed the name from XML-RPC to the more generic Simple Object Access Protocol. The first draft was heavily linked to Microsoft technology and used Microsoft’s Biztalk server software but in the later version 1.1 it supported the W3C’s XML schema standard.

SOAP is a XML based way to send and receive information over a network such as the Internet. With the right software support such as the Jakarta Tomcat server you will get the ability to do Remote Procedure Call (RPC) with your XML messages and receive the information back to you in a XML message. There are different ways to use this; in it's easiest form you use a web browser to access an on-line site that provides you with an interface to call the objects. The technical solutions are hidden behind the web interface. Another way to utilize SOAP is to incorporate it in an application and use the XML messages to send information to a server containing the objects used in the application. The application is then only an empty shell that resides on your local computer allowing the calls to be made to the server where the objects and the data are stored. In this way all the clients don’t have to be replaced when changes in the objects are done. Also it's possible to ensure that the data is stored in a safe environment and accessed only as intended.

SOAP was developed with two major design goals:

-Provide a standard object invocation protocol built on Internet standards, using HTTP as the transport and XML for data encoding.

-Create an extensible protocol and payload format that can evolve.

HTTP was chosen as the primary application layer protocol for SOAP since it works well with today's Internet infrastructure, specifically, SOAP works well with network firewall . XML was chosen as the standard message format because of it's widespread acceptance by major corporations and opensource development efforts. Additionally, a wide variety of freely available tools significantly ease the transition to a SOAP-based implementation.

The somewhat lengthy syntax of XML can be both a benefit and a drawback. it's format is easy for humans to read, but can be complex and slow down processing times. For example, GIOP and DCOM use much shorter, binary message formats. On the other hand, hardware appliances are available to accelerate processing of XML messages. Binary XML is also being explored as a means for streamlining the throughput requirements of XML.

A SOAP message is contained in an envelope. Within this envelope are two additional sections: the header and the body of the message. SOAP messages use XML namespaces. The header contains relevant information about the message. For example, a header can contain the date the message is sent, or authentication information. It is not required, but must always be included at the top of the envelope when it's present.

An example of how a client might format a SOAP message requesting product information from a fictional warehouse web service. The client needs to know which product corresponds with the ID 827635.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body> </soap:Envelope>
```

Here is how the warehouse web service might format it's reply message with the requested product information.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
   <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
     <getProductDetailsResult>
      <productName>Toptimate 3-Piece Set</productName>
       <productID>827635</productID>
       <description>3-Piece luggage set.  Black Polyester.</description>
       <price>96.50</price>
       <inStock>true</inStock>
     </getProductDetailsResult>
   </getProductDetailsResponse>
 </soap:Body> </soap:Envelope>
```

Shibboleth use SOAP as a definition of how to use XML to transfer data between the servers.

4.6 PKI. Public Key Infrastructure

In the physical world, face-to-face transactions, photo identification and even written signatures offer some protection against fraud. However, the Internet remains relatively anonymous, making it harder to know who is at the other end of the network. The challenge is to translate the trust conventions from the physical to the on-line world. A Public Key Infrastructure (PKI) **(Fig. 4.9)** has become the de facto standard for establishing this trust over electronic networks.

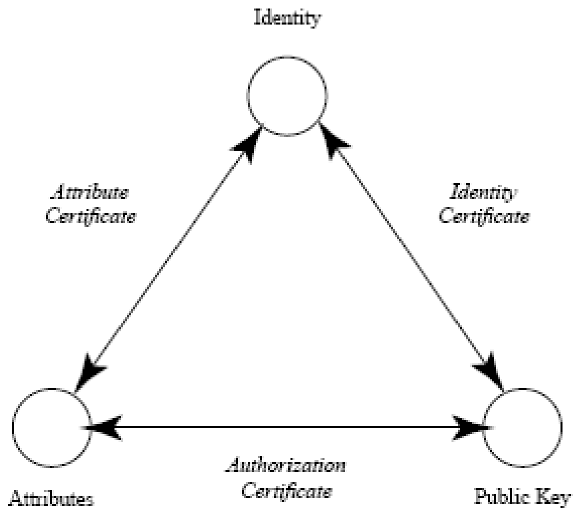


Fig. 4.9: Public Key Infrastructure components

Public Key Infrastructure (PKI) is a system of digital certificates and Certificate Authorities that verify and authenticate the validity of each involved party.

Certificate Authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption and signature verification.

A digital certificate consists of the public key and the identity of an entity, rendered unforgeable by digitally signing the entire information with the private key of the issuing Certificate Authority (CA).

The name Public Key Infrastructure is used because Certificate Authorities issue digital certificates by signing public keys.

> A public key is a value that can be used to effectively encrypt messages and verify digital signatures.
> The public key can be made publicly available, it does not contain secret information. All secret information is stored within the corresponding private key.
> A private key is a value - known only to one party - that can be used to decrypt encrypted messages, issue digital signatures and compute the corresponding public key.

The private key must be kept private and must not be made publicly available. Together, a private and a public key form a key pair.

The most important services of a Certificate Authority are:

-*Key Registration:* Issuing a new Certificate for a public key after verifying the identity of the person demanding the Certificate.

-*Certificate Revocation:* Cancelling a previously issued Certificate. This is usually done when a private key is corrupted.

-*Key Selection:* Obtaining a party's public key. The CA provides the corresponding public key if someone asks for a specific identity.

-*Trust Evaluation:* Determining whether a Certificate is valid and what operations it authorizes.

Shibboleth use a PKI infrastructure to establish a trust environment where the federations can work.

4.8 XML (eXtensible Markup Language)

XML, an acronym for eXtensible Markup Language, was derived from Standard Generalized Markup Language (SGML). A document written in XML is said to be well formed if the tag structure is equivalent to a balanced tree. Unlike HTML, the tag vocabulary of XML is unrestricted. Users can create arbitrary tags that can have arbitrary meanings.

The only syntactic constraint XML imposes on it's users is that valid XML documents must conform to a pre-defined Document Type Definition (DTD) or XML Schema. DTD's and XML Schemas describe the structure of an XML document. XML processors (or parsers) perform XML document validation using the schema to reference tag structure descriptions. XML Schemas were recently introduced by the World Wide Web consortium. The syntax is similar to XML documents in that they must represent balanced trees. However, the tag vocabulary of schemas is restricted to well defined set of tags. This thesis primarily deals with XML files that use schemas rather than DTD's.

A schema is composed of entities known as schema components. In all, there are 13 different schema components that fall into three different groups namely, primary components, secondary components and helper components.

Primary components include the following:

-*Simple Type Definitions:* determine constraints on element and attribute declarations. The content model of simple types can contain pre-defined datatypes as well as other simple types. Simple type definitions, if named, must be unique.

-Complex Type Definitions: also constrain element and attribute declarations. Unlike simple types, the content model of a complex type can include other simple and complex type definitions or declarations in addition to pre-defined datatypes. Complex types also introduce the notion of type derivation.
Every complex type definition is derived either by restricting a complex type definition or by extending a simple or complex type definition. Type derivation will be discussed shortly. Complex type definitions, if named, must be unique.

-Element Declarations: associate a name with a simple or complex type definition.

-Attribute Declarations: associate a name with a simple type definition.

Secondary components include the following:

-Notation declarations: associate a name with an identifier for a notation.

-Model group declaration: is an association between a name and a model group. Model groups constrain the order and cardinality of a list of element declarations. There are three different types of model groups, namely:

·*Sequence:* Element declarations in the XML file must follow a specific sequence defined by the sequence model group.

·*Choice:* Element declarations in the XML file can have an arbitrary order and cardinality. This is the equivalent of a logical disjunction.

·*All:* Element declarations in the XML file must have all elements specified in this model group. This is the equivalent of a logical intersection.

-Attribute group definitions: associate names with attribute groups. An attribute group contains a set of attribute declarations that are included in complex type definitions as a group.

-Identity-constraint definitions: associate names with uniqueness and reference constraints.

Helper components are the following:

-Particles: include an element, wildcard or group declaration followed by occurrence constraints. They impose cardinality constraints on content model declarations.

-Wildcard: is a special case of a particle definition that includes an additional namespace constraint.

-Attribute Uses: used in an attribute declaration to specify whether that particular attribute is required or allowed.

-Model Groups: provide for a more refined definition of complex content models.

XML Schemas use a restricted tag set to describe the structure of an XML document. XML schemas are syntactically constrained to form a balanced tree (for example, every starting tag must have a corresponding closing tag). XML schemas encompass a wide range of features.

> *-Type Derivation:* This is similar to class-subclass relationships. A type derivation that has the same schema components as another type definition and restricts the range values of these entities is said to be a restriction. Restriction applies to both simple and complex types. A type derivation that includes additional content in addition to the content model of an existing type is said to be an extension. As simple types cannot have additional simple or complex content types, only complex types can be derived by extension. An element of type A that has been derived from a type B, by restriction is also an element of type B. This mirrors class-subclass relationships where a class A is said to be a subclass of class B if every member of class A is also a member of class B. However, this similarity is limited to type derivations by restrictions. An element of a type A, derived by extension of type B, is not necessarily an element of type B.
>
> *-Namespaces:* XML schemas require that the set of type definitions and elements are said to be locally unique in a domain name known as a namespace. Namespace is known as a qualified name and must be globally. Such namespaces are known as target namespaces. If no namespaces are defined, then a default namespace (www.w3.org/2001/XMLSchema) is assumed by a schema validator. A namespace identifies the location of the schema document that contains the precise definition of the element declaration or type definition. As XML does not have any restriction on the XML tag vocabulary, it's entirely possible for different users to conjure up tags with different meanings but identical names. However, using the namespace facility, each user can define the element/type definition in a schema that serves as a namespace. Prefixing the namespace schema to element names ensures that a schema validator can access the proper definition and perform an accurate validation.

XML allows any type of data to be encoded as there are no restrictions on XML tags. The only constraint it enforces is that a well formed and valid XML document conform to an DTD or schema, for example, it requires that the XML file adhere to a pre-defined grammar. In any communication protocol, both sides must have a consistent protocol description. In the case of XML, both sides need access to the schema to process XML data. Note that the schema only provides a syntactic description of data.

The semantics provided by a schema is extremely limited. If we assume that web users all over the world will use the same terms to describe entities, then parsing a XML file will yield both syntactic and semantic information. In the context of controlled domains where users agree on common standards, this however becomes a reasonable assumption. This is a naive assumption to make for the web. As we shall see in the following section, RDF overcomes these limitations to a great extent and can represent semantics more efficiently than XML.
Shibboleth uses the XML file format in all the configuration files.

4.9 SAML (Security Assertions Markup Language)

SAML, an OASIS standard, is a standard XML-based framework for the exchange of attributes, authentication and authorization information. Prior to SAML, there was no XML based standard that enabled exchange of security information between a security system (such as an authentication authority) and an application that trusts the security system. SAML provides a standard XML schema **(Fig. 4.10)** for specifying authentication, attribute, and authorization decision statements, and it additionally specifies a WS-based request/reply protocol for exchanging these statements.

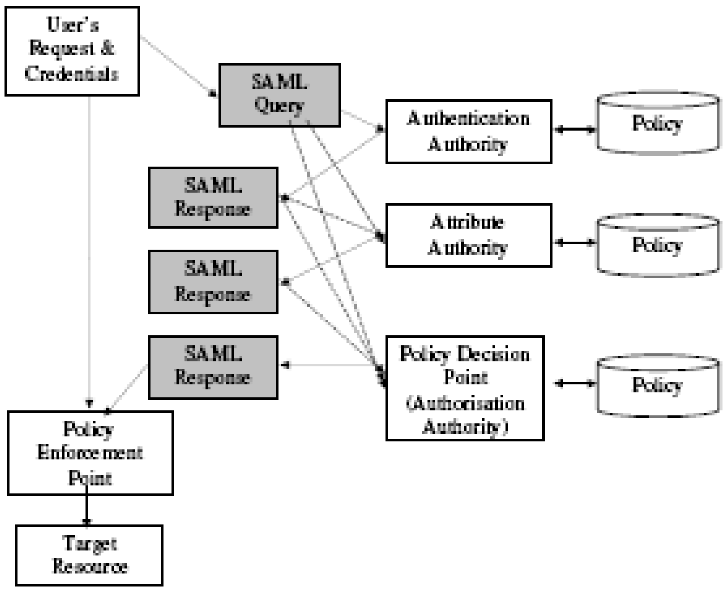


Fig. 4.10: SAML Diagram

SAML is designed to keep your information with only a few selected parties and allowing those parties to share that information with other interested parties, if required, after your explicit approval. This means that your information is safe in the hands of parties you trust, plus you can have access to a range of higher-level services offered by vendors that bring together a multitude of lower-level services.

The primary objectives of SAML are:

-Create an authentication and authorization exchange mechanism that is protocol- and platform-independent (SSO).

-This should be independent of the deployment environment and should work with centralized, decentralized, and federated deployment scenarios.

-The SAML framework should be XML-based. SAML is a mechanism for controlling access to resources for authenticated principals. Access to resources is managed based on specific policies. Two key actions are required for such a mechanism:

- Decisions about access control based on policies.
- Enforcing those decisions.

SAML provisions two roles to handle these actions: Policy Decision Points (PDP's) and Policy Enforcement Points (PEP's).

Scenario: The subject is interested in accessing some secured content from a target Web site. The subject goes to the source Web site that recognizes the subject or has already authenticated the subject. From the source site, the user tries to access secured content on the target Web site, taking the following steps:

1. Subject authenticates to the source site and requests a link to target site's secured resource.

2. Source site redirects to the subject using the authentication token.

3. Subject makes a request to the target site for the secured resource using the token.
4. Target site's PEP checks permission with the PDP.
5. PDP may internally request the source site for the SAML authentication assertion using the token.
6. The source site provides an SAML authentication assertion to the target site based on the token.
7. Target site provides the secured resource to the subject.

In all, a subject having been authenticated at the originating site gets a token from the SAML authority that it provides to the target site. The target site uses the token to request the information it needs from the originating site without having to get it explicitly from the subject.

The SAML specification is made up of the following components:

-Assertion and protocol: This specification addresses the syntax and semantics for defining XML-encoded assertions as well as the request and response protocols.

-Bindings and profiles: This specification addresses the mapping of SAML request/response message exchange onto lower layer communication protocols like SOAP or SMTP. A set of rules that govern embedding and extracting the SAML information from lower layer communication protocols is called a profile.

-Conformance specifications: Different SAML implementations may only implement part of these specifications. Conformance specifications set the basic standards to which an implementation of the SAML specification must conform before it can be called a conformant implementation. This helps with interoperability and compatibility.

-Security and privacy considerations: This specification covers the security risks in the SAML architecture specifically, the way SAML addresses those risks and the risks that are not addressed.

Assertions provide information about the authentication performed by the subjects, attributes of subjects, and authorization decisions about whether or not subjects are allowed to access certain resources.

A set of assertions makes a profile of a subject. Assertions in a profile set may be from different organizations. The three assertion types are:

-Authentication: Authentication assertion deals with authentication of a subject by a specific mechanism at a particular time.

-Attribute: Attribute assertion provides a mechanism for associating specific attributes with a given subject.

-Authorization decision: Authorization decision assertion manages a given subject's authority to access resources.

SAML authorities produce assertions. SAML authorities can be further specified as authentication authorities, attribute authorities, or PDP's. Consumers of assertions can either be clients or SAML authorities themselves. This is an example of a SAML Response. The fields are explained after.

```
<Response>
<ResponseID>dynadi:2003-02-20:5aff53b33d7a78c4</ResponseID>
<InResponseTo>access:2003-02-20:1e5638a2bbd3180a</InResponseTo>
<IssueInstant>2003-02-20T09:12:15</IssueInstant>
<Assertion>
<AssertionID>dynadi:2003-02-20:777a35f6b3ac638b</AssertionID>
<Issuer>urn:com:ibm:zurich:dynadi</Issuer>
<IssueInstant>2003-02-20T09:11:07</IssueInstant>
<SubjectStatement>
<Subject>
<NameIdentifier>john_smith@domain.org</NameIdentifier>
</Subject>
</SubjectStatement>
<AuthenticationStatement>
<AuthenticationMethod>urn:ietf:rfc:2246</AuthenticationMethod>
<AuthenticationInstant>2003-02-20T09:10:59</AuthenticationInstant>
</AuthenticationStatement>
</Assertion>
</Response>
```

IDType. This basic data type declares identifiers for assertions, requests and responses. Values of this type are supposed to be chosen uniquely such that no party accidentally assigns the same identifier to a different data object. The IDReferenceType is used to refer to instances of IDType.

Assertion. An Assertion is a package of information that supplies one or more statements made by an issuer. An Assertion contains the following major elements:

-AssertionID [IDType, required] Identifies the assertion.

-Issuer [String, required] The name of the issuer of the Assertion.

-IssueInstant [UTC, required] The time instant of issue.

-Statement [Statement, one or more] A Statement about a subject made by the issuer.

Statement. The Statement element is an abstract extension point for different kinds of concrete Statements. The most important derived concrete elements are Subject Statement and Authentication Statement.

*Subject Statemen*t. This Statement contains a Subject element that
allows an issuer to describe a subject. The Subject element itself has the following structure:

-*Name Identifier* An identification of a subject by a name and security domain.

-*Subject Confirmation* Information that allows the subject to be authenticated. If the Subject contains the Subject Confirmation in addition to a NameIdentifier, the relying party can perform the Subject Confirmation to verify that the entity presenting the assertion is the one that the issuer identifies with the Name Identifier.

A*uthentication Statement.* The issuer states that the subject was authenticated by a particular means at a particular time.

Attribute Statement. The issuer states that the specified subject is associated with the specified attributes.

Request. The requester sends a message of type Request to a SAML responder. The Request inherits different elements from Request Abstract Type.

-*RequestID [IDType, required]* Identifies the request. If present the Response corresponding to the Request must contain it's value in the InResponseTo element.

-*IssueInstant [UTC, required]* The time instant of issue of the request.

-*Query* Specifies the query to be answered by the SAML responder.

-*AssertionArtifact [String, one or more]* Contains a so-called SAML artifact, which is a String handle and represents a specific assertion.

Response. The Response element specifies the message sent back by the SAML responder.

-*ResponseID [IDType, required]* Identifies the response.

-*InResponseTo [IDReferenceType, optional]* Refers to the Request the Response corresponds to. If the RequestID of the Request can be determined, it must be present.

Shibboleth is a set of SAML profiles, is a functioning SAML instantiation for federated administration, with privacy management built into the design. The Appendix C shows the assertion schema.

4.10 Attribute-Based Authorization

Typical user authentication methods only provide the application with the permanent user identifier (user_id) of the person who has authenticated. This simple approach won't suffice in modern systems. Applications need additional information about users-user attributes to make proper authorization decisions. Providing this information as part of the sign-on process is especially useful in multi-organizational situations where an application probably won't have access to user information through other means such as a directory service. Shibboleth is designed specifically to provide user attributes to applications with the flexibility, extensibility, security, and privacy required in federated scenarios. Organizations can use Shibboleth's built-in attribute support (based on the Internet2/EDUCAUSE eduPerson directory schema or create new attributes to meet the needs of applications. For example, attributes can represent “entitlements” such as “user is authorized to access resource collection X.”

The Attribute Based Authorization is the main concept that Shibboleth is based. The Shibboleth IdP software plugs in to existing institutional identity management and user information services (typically Lightweight Directory Application Protocol, or LDAP-based, directories), extending them to work inter-organizationally.

4.11 User attributes

User attributes are the basis for how the target site makes it's authorization decisions. In Shibboleth, attributes are usually name/value pairs relevant to the user. Shibboleth also provides for hidden attributes relevant to the origin and target site, but not necessarily to the user.

Shibboleth places a large emphasis on user privacy. However, Shibboleth target sites are greedy and will try to obtain as many of the user's attributes as possible. To balance access and privacy, Shibboleth allows it's users a choice in what information gets released about them and to which site. To achieve this balance, Shibboleth lets each user have an Attribute Release Policy (ARP).

When the SHAR at the target site asks the AA at the origin site for attributes for the user with a specific handle, the AA retrieves that user's ARP and only releases attributes consistent with that policy. Shibboleth ARP's consist of a list of entries, each with three main fields: a destination SHAR name, a resource URL, and a list of attributes that can be released to this SHAR and URL (if the user has these attributes and the SHAR wants them). The SHAR is usually the Website where the URL resides and which hosts the resource. As noted earlier, the origin site might have hidden attributes, such as a contract number, and an institution-specific ARP to specify when they should be released. Thus, the AA may serve to add additional attribute values into the Attribute Response Message (ARM) sent to the SHAR.

Since is not possible for a user and her institution to be able to provide ARP's for every possible target resource on the Internet, and every SHAR and URL pairing, Shibboleth permit's default and wildcarded ARP's.

The Shibboleth standard requires that the AA must provide users at the origin side a configuration file by which they can specify their Attribute Release Policies, this is usually done using a GUI such as a web browser and enables the user to control his own privacy. The downside, of course, is that a user's choice of ARP may not be proper to be able to grant him access to a target resource. Due to this problem, it's often preferable for the user to be aware of each site's attribute requirements, possibly shown on the interface.

The Problem is that Shibboleth defines the basic structure and use of an ARP. However, how the AA retrieves a user's ARP is not part of the Shibboleth standard and is left open to interpretation. The user may have a single ARP or multiple ARP's. They may be dispersed throughout the organization or they may be collected in one place. How the user's ARP is retrieved, validated and enforced is left to the implementers. The Shibboleth draft states: "AA implementers are free to support many different kinds of ARP's with varying semantics as long as the AA can efficiently process requests and determine the effective policy to apply...Shibboleth doesn't specify or constrain how an AA can answer these kinds of questions."

In a typical Higher Education Institute (HEI), the lines of administrative control are dispersed Furthermore, the structure of this distribution will vary from user to user. In a typical HEI, it also likely that the decision procedure for attribute release will follow such administrative lines.

```
<user-attribute>
      <description>User Given Name</description>
      <name>user.name.given</name>
   </user-attribute>
   <user-attribute>
      <description>User Last Name</description>
      <name>user.name.family</name>
   </user-attribute>
   <user-attribute>
      <description>User eMail</description>
      <name>user.home-info.online.email</name>
</user-attribute>
```

CHAPTER 5: SHIBBOLETH IMPLEMENTATION

5.1 Introduction

Now we know more in depth about Shibboleth, the goal is to develop a real scenario **(Fig. 5.11)** where to be able to put the characteristics of Shibboleth on approval, so we left from a scene with 3 machines.

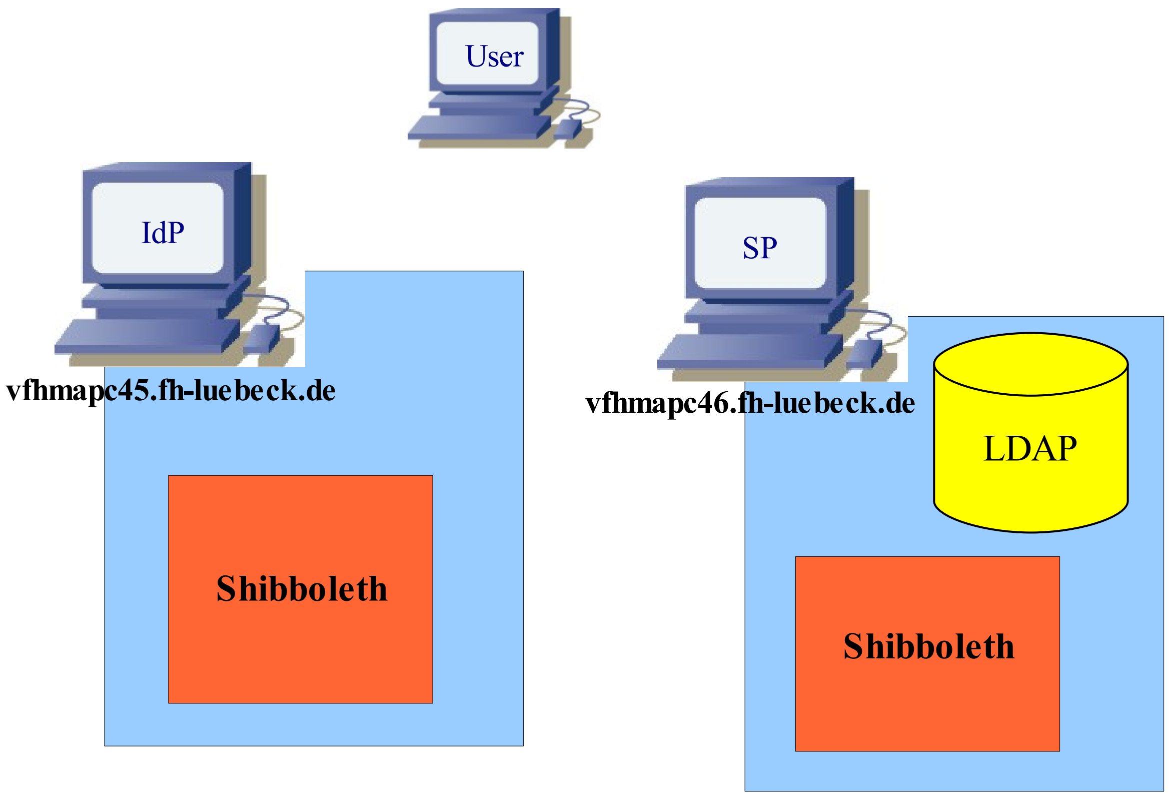


Fig. 5.11: Scene of tests

The first machine lodged the services of identification (IdP), the second machine lodged the content that we want to protect (SP) and the third machine acted properly as a client. The requirements of the last one are simply to have a navigator and connection to the network in which the rest of machines are. The machines act like IdP and SP are the one that runs properly Shibboleth, these are the machines which needs special characteristics.

These 3 machines are x86 all of them and they do not require special hardware characteristics. We left from the base that the 3 are connected through their respective networks to Internet although in this case the institution and network are common, it's not mandatory.

In this chapter, first I start to explain the pre-requirements and co-requisite to install Shibboleth in IdP and SP (I have already commented previously the client does not have special characteristics) and later the configuration of the machines to work with, in the desired scenario.

5.2 Requirements

Since Shibboleth is middleware, needs of software on which runs (O.S.) and that upon does interaction with system / client (Web server, web applications, database, another middleware, etc.). The **(Fig. 5.12)** shows the working layering for Shibboleth.

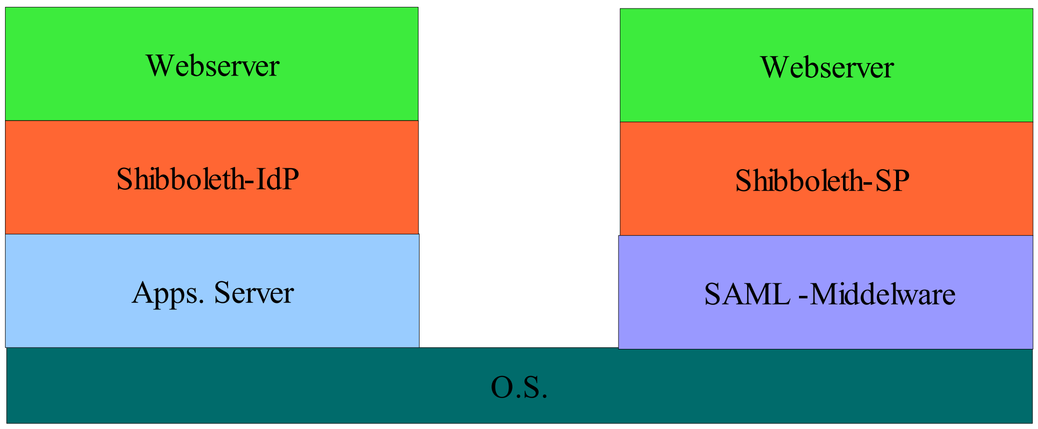


Fig. 5.12: Software Layering

Beginning by the low layer, we needed system operative where to work, normally for implementation of Shibboleth is used Freebsd or Linux, also it allows that some of their functions run on machines under Windows, Solaris and Mac-Os.

For the scenario, I use Linux (Gentoo 2005.1) for the Service Provider and BSD (Freebsd RC5.4) for the Identify Provider.

The SP runs with Linux (Gentoo) because it's the OS that I have in my machine and I'm more close to them. (For the SP it's necessary some requirements and some of them are in portage and the another's are only possible to install from scratch). I use this machine for the SP because in this machine is were are going to run the applications for the student (e-learning environment contents, etc) and for me it's more easy install and deploy this applications in a familiar environment.

The IdP runs under BSD (Freebsd) because it was a new machine and Freebsd allows me a fast installation of the system and also a good support on case of difficulties, another added reason was the fact of test Shibboleth running under Freebsd because the institution for which I'm developing the project normally works in most of her machines with them.

About the web server, I start to work with the Apache branch 1.3 because in the Shibboleth documentation recommends the use of 1.3 for best support but I quickly switch to 2.0 due to the incompatibility of 1.3 with some Apache new modules needed for the deployment and the absence or instability of some of them with the 1.3 branch (now, in the last versions of the Shibboleth docs also speak about 2.0).

In the SP the Web server is necessary to be able to offer the contents to the users (page Web, document, graph, etc.) and to allow to establish rules of access to this content, later is through these rules where grants/deny Shibboleth the access.

In the IdP the Web server is necessary for the remote authentication in the system (campus, library, etc.) and to pass the password of the SSO to Tomcat. For the rest of Web server functions Tomcat replaces Apache. If it's necessary, is possible to work without Apache in the IdP.

About Tomcat, the branch has been used is the 5.0 although according to the documentation also it's possible to use branch 5.5, I chose the first because have better support in the used systems. Tomcat, only is necessary in the IdP, and the purpose it's to manage the jar applications where Shibboleth are deployed.

Another of the requisites for the implementation is the LDAP server. In this database are located the login, password and attributes of the user of the system. It's the base of Shibboleth, is where we will be able to know who is that user, its roll, the resources who can access, etc.

I choose Openldap because have an easy implementation and are available in the portage packages of Gentoo. The ldapserver runs in the IdP machine but it's possible to use a existing ldap server of the institution or use another database like Mysql, Postgres, etc.

This is the ldif I use to fill the database:

```
dn: dc=fh-luebeck,dc=de
dc: fh-luebeck
objectClass: dcObject
objectClass: organization
o: VFH-Luebeck

dn: cn=anne, dc=fh-luebeck,dc=de
userPassword:: e1NIQX1YQzNaUk4zcDRJaUJ2dkNKVCtleUtseWNTd1k9
description: biology
objectClass: top
objectClass: person
sn: lenna
cn: anne
```

For the Shibboleth development I use 3 attributes:

-c*n and userPassword:* Used by the SSO to login in the system.

-*description:* This field contains the courses where the user are joined.

5.3 Deploying

In order to test the implementation of Shibboleth I improve a simplified solution is shown in the figure **(Fig. 5.13)**. Although it was in the beginning a bit complex, especially by the subject of the service of WAYF and the Federation, once finished the installation and for the final tests, both have been suppressed. The objective is the control of the access to the resources (document, webpage, image, etc.) hosted in the SP, the policy and the attributes for the access to the resource are defined in Apache in the *httpd.conf* configuration file.

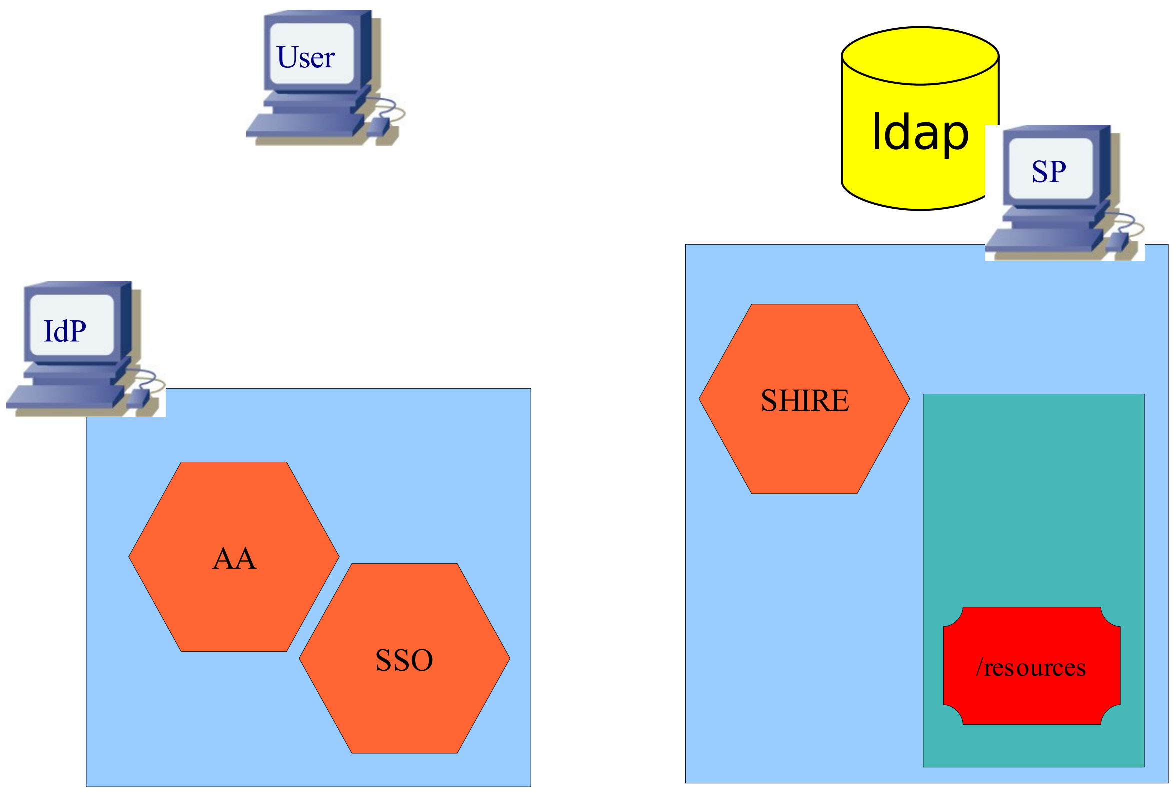


Fig. 5.13: Scenario Overview

The names of user, passwords and attributes are located in a LDAP server. This minds all the specific data about the users is located in a centralized database this allows a better control of “the sensible” data of user with which it improves the privacy. To this database, that circumstantially located in the SP machine it's possible can be defined access policies so that it could be accessible from the IdP machine as well as from the SP for his management.

5.3.1 IdP (Origin)

For the development of the IdP, after the Freebsd installation, and all the requirements (openssh, apache+modules, Tomcat), and the configuration of Tomcat and the Apache modules, the Shibboleth install program copies the .war to the directory of webapps, is when Tomcat starts when the war file is installed and appears the resource /Shibboleth-idp/. In the figure **(fig 5.14)** can be observed a scheme of the IdP installation.

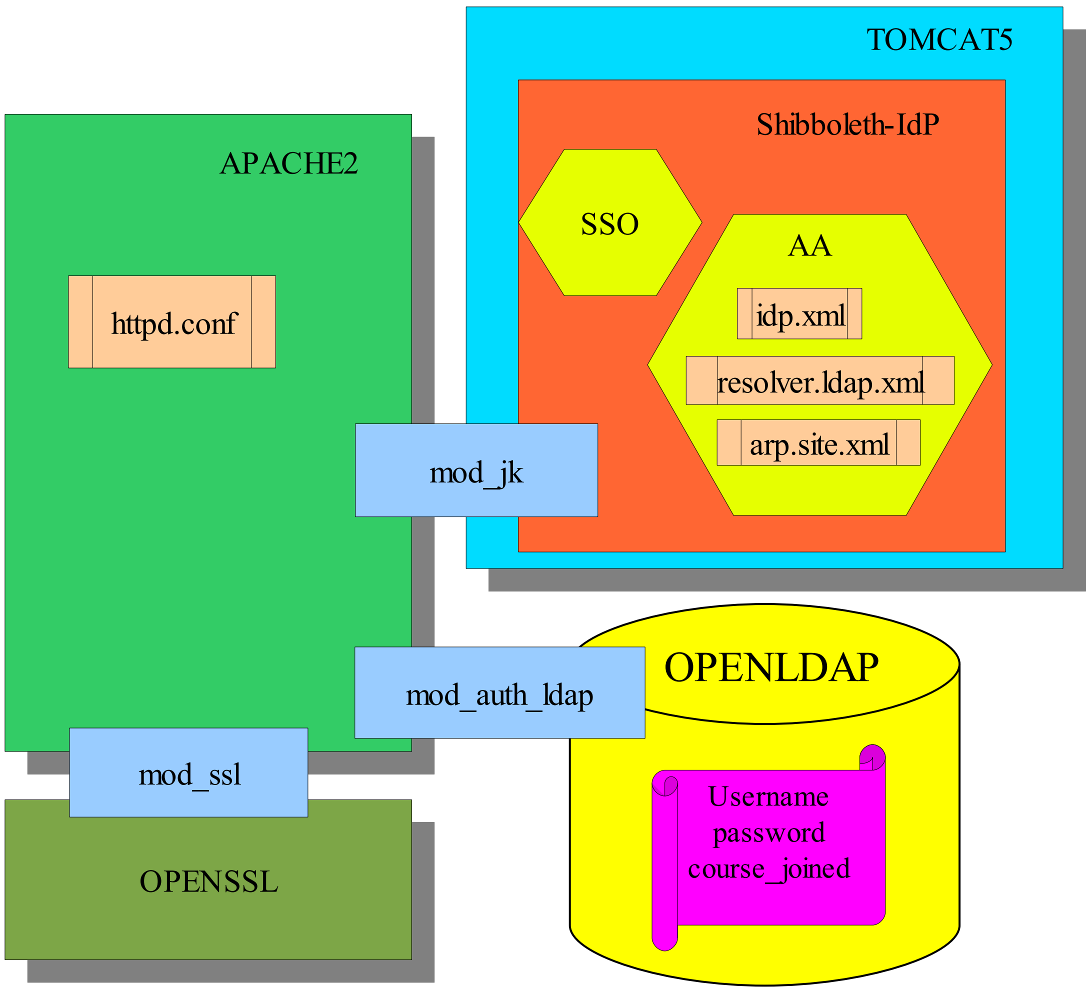


Fig. 5.14: IdP Software Structure

The following step is properly the configuration of Shibboleth that consists of 3 files:

-idp.xml: Contains the main configuration of the IdP. Aside the configuration of federation and the certificates the important points are the situation of the AA

```
AAUrl="https://vfhmapc45.fh-luebeck.de:8443/shibboleth-idp/AA"
```

and the file used to access to the attributes.

```
resolverConfig="file:/usr/local/shibboleth-idp/etc/resolver.ldap.xml"
```

-resolver.ldap.xml: In this file, is configured that I want to use an LDAP server for get the attributes and the location of the server.

```
<JNDIDirectoryDataConnector id="directory">
     <Search filter="cn=%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://vfhmapc46.fh-
luebeck.de:389/dc=fh-luebeck,dc=de" />
</JNDIDirectoryDataConnector>
```

and which attributes they must be available. In the file I ask for all the registries available but the really important ones are these (initials and description).

```
<SimpleAttributeDefinition id="urn:mace:dir:attribute-def:initials">
     <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>

<SimpleAttributeDefinition id="urn:mace:dir:attribute-def:description">
     <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>
```

-arp.site.xml: This file contains the sending policy attributes from the IdP. This allows send the attributes to any target (any SP).

```
<Target>
               <AnyTarget/>
</Target>
```

I define send 3 attributes (cn, sn and description). Its also possible to define another attributes depending the rules or the scenario.

```
<Attribute name="urn:mace:dir:attribute-def:sn">
        <AnyValue release="permit"/>
</Attribute>

<Attribute name="urn:mace:dir:attribute-def:cn">
        <AnyValue release="permit"/>
</Attribute>

<Attribute name="urn:mace:dir:attribute-def:description">
        <AnyValue release="permit"/>
</Attribute>
```

It's also possible to define policies depending on the user or SP who ask for them. For example if the SP is from the own institution its possible to release more details of the user than if the SP is a public library that only needs if this user have grants or not.

In the Appendix E there are some explanations about how to install and configure the IdP side of Shibboleth.

5.3.2 SP (Target)

For the SP deployment, the steps are a bit complicated because the SHAR of Shibboleth depends of a lot of other software:

> *-Xerces-C:* Is a validating XML parser designed to give an application the ability to read and write XML data. Provides a shared library for parsing, generating, manipulating, and validating XML documents. Shibboleth uses this to work with XML files.
>
> *-Log4cpp:* Is a library of C++ classes for flexible logging to files, syslog, and other destinations. Shibboleth uses this to log the activities.
>
> *-XML-Security:* Library developed by the Apache project with the aim of work with XML in a secure way. Shibboleth use this to manage secure XML files.
>
> *-Opensaml: I*s a set of open-source libraries in Java and C++ which can be used to build, transport, and parse SAML messages. OpenSAML is able to store the individual information fields that make up a SAML message, build the correct XML representation, and parse XML back into the individual fields before handing it off to a recipient. OpenSAML supports the SOAP binding for the exchange of SAML request and response objects (C++ supports requesting only). Shibboleth use this to support the SAML assertions.

The last step is compile the Shibboleth source code. If everything was ok, now to start the server it's necessary to start the shar always before start apache.

```
<path_shibboleth_install>/bin/shar -f &
```

A detail of the SP software structures is shown in the figure **(Fig. 5.15).**

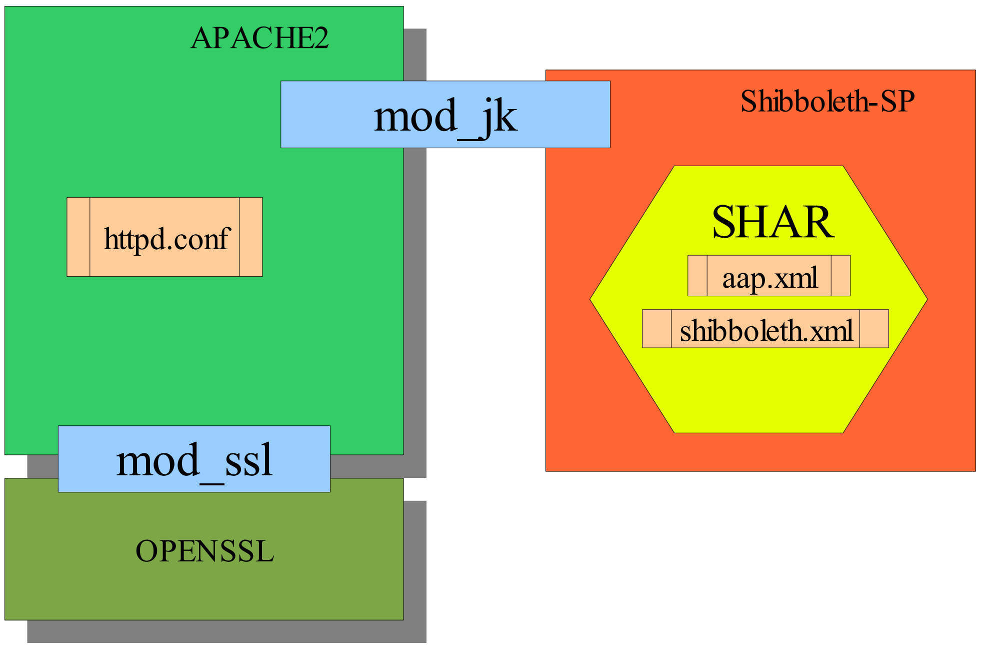


Fig. 5.15: SP Software Structure

The 3 basic files of configuration are:

-*shibboleth.xml:* This file contain the basic configuration of the SP, the session lifetime, the wayf location (In this case, there are no wayf and this points to the SP) and the location of the shire.

```
<Sessions    lifetime="7200"    timeout="3600"    checkAddress="true"
wayfURL="https://vfhmapc45.fh-luebeck.de/shibboleth-idp/SSO"
shireURL="/Shibboleth.shire" shireSSL="false"/>
```

-*aap.xml:* This file contain the configuration of the attributes to receive and the bind with the local attributes. Also this configuration allows to release this attributtes to any SP.

```
<AttributeRule        Name="urn:mace:dir:attribute-def:description"
Header="Shib-Person-enrolled" Alias="enrolled">
          <AnySite>
            <AnyValue/>
        </AnySite>
</AttributeRule>
```

In the file binds the attribute description with the alias enrolled. This binding is needed for apache. Binds is the “attribute that apache knows.”

-*http.conf:* This file, aside the configuration of the webserver contains the configuration of the folders to protect.

These are the lines concerning to Shibboleth.

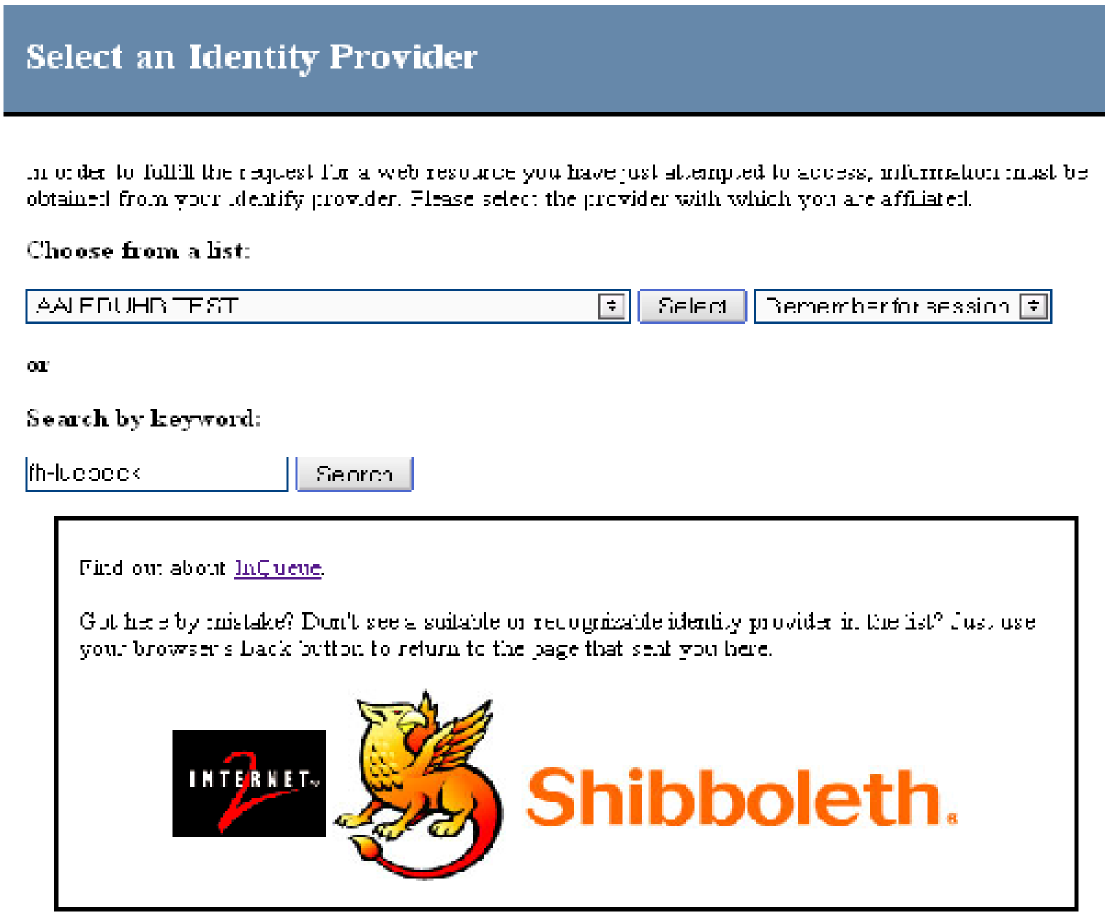
```
<Location /electronics>
  AuthType shibboleth
  ShibRequireSession On
  require enrolled electronics
</Location>
```

“enrolled” it's the bind of the description in the ldap database and electronics the attribute needed for grant the access to the electronics folder

In the Appendix F there are some explanations about how to install and configure the SP side of Shibboleth.

5.3.3 Federation and WAYF

I start the deploying joining to a federation called InQueue, InQueue it's a test federation environment and it's helps to test the deploy (provide some tools that for example show with kind of attributes are send, etc). In the beginning, every time I want to access to a resource the browser shows this screen and the user needs to select the own IdP. This drawing shown a snapshot of the wayf service for InQueue **(Fig. 5.16)**.



Select an Identity Provider

In order to fulfill the request for a web resource you have just attempted to access, information must be obtained from your identify provider. Please select the provider with which you are affiliated.

Choose from a list:

AAI FDUHR TEST | Select | Remember for session

or

Search by keyword:

fh-lueneck | Search

Find out about InQueue.

Got here by mistake? Don't see a suitable or recognizable identity provider in the list? Just use your browser's back button to return to the page that sent you here.

Fig. 5.16: InQueue WAYF Service

Because in the scenario are only one IdP, i don't use federation and the wayf check is omitted. Now the SP knows in which IdP are needed to login, because this, in the SP configuration, instead of point to a wayf service, point directly to the IdP.

The configuration files in the Appendixes are also valid for the InQueue Federation.

CHAPTER 6: E-LEARNING APPLICATIONS AND SHIBBOLETH

6.1 Introduction to the CMS (Course Management Systems)

Many universities have been using a Course Management System (CMS) for a few years now. A Course Management System is a web based application **(Fig. 6.17)** through which students and teachers can interact. Course documents such as presentation sheets and assignments are made available to the students by the teacher, and students can work on assignments in groups and take on-line tests.

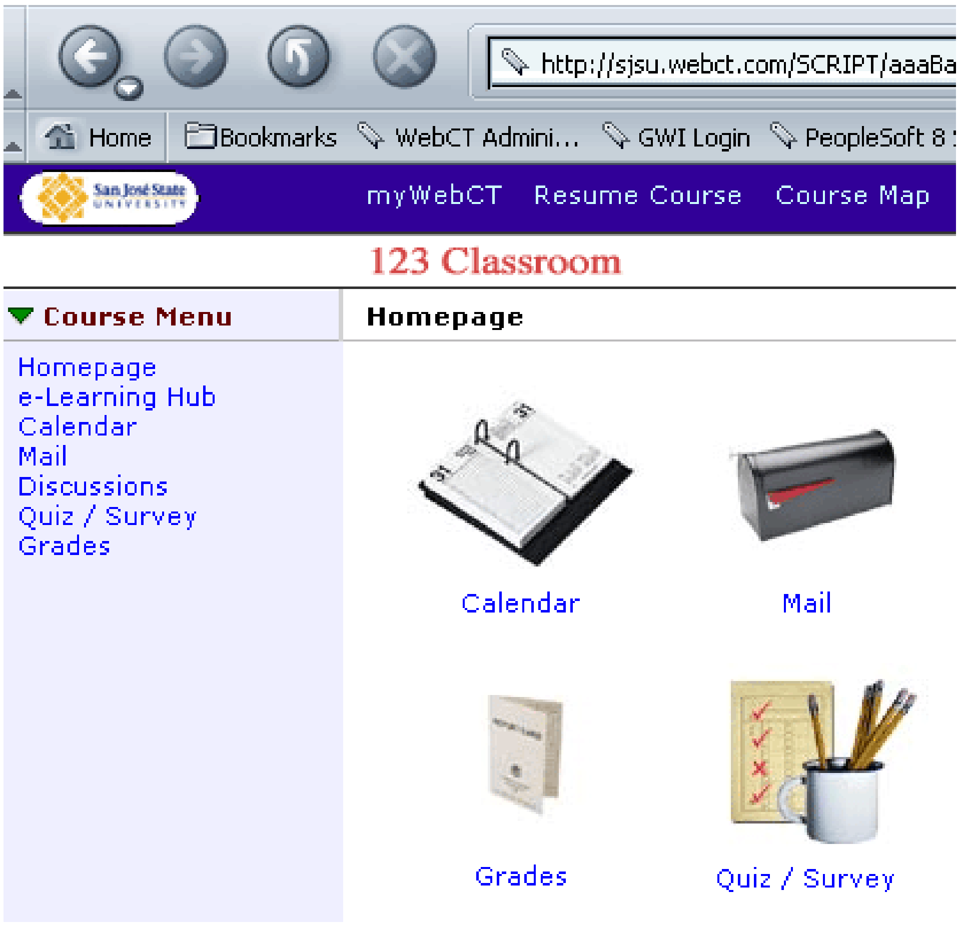


Fig. 6.17: CMS Snapshot

There is no agreed upon definition or even one single term for CMS's. Other terms used are Virtual Learning Environments (VLEs), Managed Learning Environment (MLE), Learning Management System (LMS) and Learning Support System (LSS).

6.2 Shibbolize a e-learning application

Shibbolize is the process of adding Shibboleth to an application. Primary should be possible to Shibbolize most of the web applications with the condition of knows how works inside and the availability of the source code.

Shibboleth has been mainly designed for exchanging authorisation attributes across institutions using a secure and devolved model, where users are authenticated and authorised against their own institutional access management system (IdP). Its scope is wider than an institution itself, being very suitable for scenarios where the universe of users does not necessarily belong uniquely to one institution.

Within an institutional context, integration of Shibboleth into CMS or portal (Most of them are linked with the access portal of the institution) might be done in at least two different approaches. The easiest way it's natively but most of the times it's not possible because this software exists before Shibboleth. A second option is the integration with the institutional authentication system explained in the chapter 2 (WebISO, PubCookie and authorisation system (for example role-based authorisation supported by LDAP).

The former approach uses Shibboleth in a WebISO fashion and extends the default mechanisms. Shibboleth is powerful enough to be used as a native system to provide SSO authentication and authorisation within an institution.

The latter approach typically uses a WebISO system already deployed and integrated with the institutional Portal and other resources and services. This leaves the WebISO system to provide SSO across the institution for the Portal as if it was any other SSO-enabled Web-based resource/ service. The WebISO authentication point has to be 'Shibbolized' (acting as a SP), at least for all users not coming from a security domain internal to the institution.

For users succeeding to authenticate, all the attributes associated with the user might have to be fetched and associated with the WebISO SSO session. In the case of a portal, one of the attributes has to be a unique and permanent identifier such as username, e-mail address or something more pseudonymous (like an e-mail address).

The purpose of this chapter is explore the possibilities of Shibbolize two of the CMS applications used in on-campus and others like Moodle that should be a good alternative.

6.3 Blackboard

Blackboard CMS is a software product from Blackboard Inc. Blackboard develops and licenses enterprise software applications and related services to over 2200 education institutions in more than 60 countries. These institutions use Blackboard software to manage e-learning courses.

Blackboard is used to support flexible teaching and learning in face-to-face and distance courses. It provides tools and facilities for on-line course management, content management and sharing, assessment management, and on-line collaboration and communication.

Blackboard provides instructors with the following functions:

- Announcements
- Course Content
- Tests/Quizzes
- Surveys
- Discussion Boards
- Chat Rooms
- Broadcast Email
- Grade book Management
- Assignment Collection

Some of the advantages of Blackboard are:

- Customizable Course Menu & Buttons.
- Content Types:
    - Assignments
    - Learning Units
- Copy and Move Course Content: Course content can be copied or moved to another place within the course, or to another course entirely.
- Linking Content and Tools: Content and tools can be linked to other areas within the course.
- External Link Editor. Allows URLs for external links to be placed anywhere in content areas.
- WYSIWYG (rich text) editor available for PC users.

Regarding the Blackboard documentation, Shibboleth is fully supported as a custom authentication option for Blackboard Learning System on UNIX operating systems. In the Appendix H there are some explanations about how to configure Blackboard to use Shibboleth.

6.4 Sakai

The Sakai Project is a community source software development effort to design, build and deploy a new Collaboration and Learning Environment (CLE) for higher education. The Project began in January, 2004.

The Sakai Project has it's origins at the University of Michigan and Indiana University, where both universities independently began open source efforts to replicate and enhance the functionality of their existing CMSs .Soon after, MIT and Stanford joined in and, along with the Open Knowledge Initiative (OKI) and the uPortal consortium, and a generous grant from the Mellon Foundation, they formed the Sakai Project. The Sakai Project's primary goal is to deliver the Sakai application framework and associated CMS tools and components that are designed to work together. These components are for course management, and, as an augmentation of the original CMS model, they also support research collaboration. The software is being designed to be competitive with the best CMS's available.

The tools are being built by designers, software architects and developers at different institutions, using an experimental variation of an open source development model called the community source model. To provide a support system for institutions that want to be involved in the Sakai Project, either by adopting Sakai tools or by developing tools for inter-institutional portability, the Sakai Project has also formed the Sakai Educational Partners Program (SEPP) and the Sakai Commercial Affiliates Program.

The Sakai Project follows what is called the community source model, which is an extension to the already successful, economically feasible, open source movement forged by projects such as Apache, Linux, and Mozilla. Based on the goal of addressing the common and unique needs of multiple institutions. community source relies more on defined roles, responsibilities, and funded commitments by community members, than some open source development models.

To date, the Sakai Project has put out three major software releases (1.0, 1.5 and 2.0), developed an Educational Partner's Program which now has around 80 members around the world with 14+ active discussion groups and five commercial affiliates, organized three highly successful SEPP conferences, and successfully demonstrated a model for community source software development among colleges and universities.

Some of the advantages of Sakai are:

- easy-to-use interface.
- notify users of new content via email .
- export calendar events as PDF.
- create online assignments: allow students to submit text based assignments or upload a file for review .
- subscribe to RSS feeds: allows instructors to display syndicated content from other websites.

- manage multiple files simultaneously: allows instructors to upload or delete more than one file at a time .
- display web content from within the course site: provide easy access to web resources by adding links to the navigation panel.

In the future, the Sakai Project expects to continue it's effort to build out it's interoperable framework and develop many new tools and extensions. In addition, the Partner's Program will continue to grow and the development model used for Sakai is expected to become more broadly applicable.

About Shibbolize Sakai, the only information that I found is that there are some efforts to Shibbolize but at the end of this work the results are not published.

6.5 Moodle

Moodle is a course management system, designed specifically to manage internet-based educational courses. The design is influenced strongly by progressive ideas of educational theory such asocial constructionism. Is the creation of Martin Dougiamas, PhD student at Curtin University, Perth, Australia. This research project started in 2000. The first public release of Moodle, version 1.0, was released on August 20, 2002. In 2003 the company moodle.com was launched, offering professional support and management of Moodle installations. Currently, the project is at version 1.5, with a large number of features added since the first release.

Moodle can be used in a very wide range of environments, ranging from fairly restrictive web hosting providers to institutional servers. It requires PHP 4.1.0 with GD to work properly, and is regularly tested on Linux, Unix, Windows, Mac OS X and Netware servers. It contains complete support for MySQL and PostgreSQL, and support for other databases is very easy to add. Only a single database is required, and Moodle can share this with other applications (using table prefixes). The PHP code has been designed to be readable and hackable by people with only medium programming skills. Administration is made easy with a unique self-upgrading system (no more .sql files!).

A modular authentication design allows Moodle to be "hooked up" to external authentication services such as LDAP, IMAP, POP3, NNTP, or arbitrary database tables (for example a Postnuke user table). It also supports traditional email authentication and instant guest accounts.

The site itself is very customisable. Themes allow you to customise the look using CSS and HTML. The front page can consist of a blog-style news forum, or a list of course descriptions, or a list of course categories (departments) with abbreviated course names.

Courses themselves can be one of three predesigned layouts (weekly, topics, or social) and are built up by the teacher by adding activity modules using an intuitive interface. Activities include assignments, choices, forums, journals, resources, quizzes, and surveys. Chat and peer-graded assessments are in development, with a tracker module coming soon after that. Each of these modules is very customizable and contains a range of expanding features.

Other features include comprehensive user profiles with pictures, centralised gradebook, detailed logging and displays of user activity, teacher forums, and teacher customisation of things like access policies, naming (for example instead of "teacher" and "student" a teacher might prefer using "professor" and "participant") and so on.

Lastly, Moodle features complete localization has already been translated into 20 languages including Arabic, Catalan, Chinese, Dutch, English (UK and US versions), Finnish, French, German, Indonesian, Italian, Japanese, Norwegian, Portuguese, Spanish (Spain, Mexico and Caribbean versions), Swedish, Thai and Turkish.

The project has grown substantially, with many users and developers participating in the community at Moodle.org.

Using Moodle with Shibboleth authentication has the following advantages:

-Access to Moodle can be restricted very accurate.

-User accounts are created automatically as soon as a user login the first time.

-The user profiles are set up automatically.

-The user profiles can automatically kept up-to date all the time.

-So you don't have to care anymore for user management issues because this is basically handled by the Identity Provider of the Shibboleth user.

-Once Shibboleth users are authenticated, they can access other Shibboleth-enabled resources without login in another time. Due to this single sign-on mechanism, they for example can jump from one Moodle installation to another or the can access a Shibboleth-protected library or a web shop, always being authenticated.

-Future automatic course enrollement according to Shibboleth attributes.

According to the last documentations, Moodle is Shibboleth enabled. Just needs some configurations and teach the users. There are some guidelines in the Appendix I.

CHAPTER 7: CONCLUSIONS AND IMPROVEMENT SUGGESTIONS

7.1 Discussion

During the development of this work, a first approach, reading and learning about the possible and suitable scenarios for Shibboleth, was made. Furthermore, the software requirements that could help to understand how to deploy an scenario with this type of technology were studied. Some important concepts of security in networks were studied, and it helped to understand why Shibboleth was the correct solution for the expected requirements of on-campus (The e-learning environment at the Fachhochschule Lübeck).

But to be able to have a general overview of the solutions, some alternatives to Shibboleth were studied. Even though these ones did not fulfil the requirements or were already implemented in Shibboleth.

Later, a deeper Shibboleth study was made, which included all components and the function of every one of them. Once understood Shibboleth, since is a global solution and include in its development some other technologies it was necessary to explore this other protocols and software on which it was based, is for this reason that I had to learn about each one of the technologies before start the Shibboleth deployment.

After the theoretical study of Shibboleth, the deployment and the test of each one of the components was made to evaluate their performance in a separately way. When the different parts worked separately, all the design was deployed together. This was one of the most complicated parts, which finished when a basic example work using this scenario. The main challenge now was that Shibboleth offers very powerful surroundings, it was necessary to make it work with applications of CMS and thus to be able to demonstrate all its power.

Then a learning about the CMS concept was needed and the integration with a CMS started. Later specific efforts in commercial CMS, were made, in terms of learning about how to Shibbolize (integrate with Shibboleth) a CMS. Finally, and once tested all, some captures of the traffic were made to evaluate the behaviour of Shibboleth, placing special attention to the lower levels of Shibboleth.

7.2 Conclusion

Shibboleth is a complex piece of software, and its installation is likely to stretch many institutions, particularly when it comes to set up the infrastructure needed for it. It is essential to plan the process carefully, and to ensure that enough time is allocated.

The skills required are an advanced knowledge of system admin, experience in installing software, and also in handling a Web server configuration. The difficulty of understanding Shibboleth is heightened by the out-of-date nature of the documentation of the architecture. Most of the information is in the distribution mailing-list.

The Shibboleth software generally seems to work fine, once configured correctly. The obvious area to improve with respect to make easier the software installation is the documentation, some time it was difficult to found answers because its in deployment and normally the people works and solve the problems themselves.

About the last test and the capture traffic, there is not much to explain because all the traffic captured was encrypted through the TLS protocol, the only remarkable thing would be some packets of the requests to the LDAP server.

Having worked during some months with Shibboleth, one of the conclusions is that Shibboleth it’s a very good system for big universities, groups of investigation or big users networks since it allows, for example, to work with a structure of common data base (eduPerson), the different management of IdP by means of the WAYF and the possibility of being upgradeable. It does not have limits in the number of SP and IdP. Another goal is the possibility to create federations so that networks of trust settle down.

However, and although once the system was mounted, everything worked without hardly maintenance, the implementation in small surroundings, like the one that has been created, is not so advantageous; firstly because all the potential does not take advantage of Shibboleth and also because the assembly is a laborious process.

A good point of start is if all the servers of the network in which is wanted to mount Shibboleth are administered by the same people I believe that he is not worth the trouble Shibboleth and should be better use more simple systems. The potential of Shibboleth is in administrating groups that are different, because it establishes really good bases to manage it all, and it is much more easy to reach compatibility and a mutual working.

7.3 Future Work and improvement suggestions

During the development of this project and after working for a couple of months there are some improvement proposals.

One of the things that has not been fully implemented is the integration of Shibboleth with Sakai since it's not yet finished. Although this should not be very difficult because Sakai is open software and there is a lot of people working on that.
At the other side Blackboard is already Shibbolized, maybe because there are lots of economic motivations behind them. Nevertheless, there are some other tools like Moodle that has been Shibbolized, provably in not so much time there are available the code to Shibbolize Sakai.

Another of the things to improve, is the change of version 1.2 to version 1.3 since many concepts have been deprecated and much has been simplified, one of the biggest changes has been the adaptation to complain with the standard OpenSaml v. 2.0, this has caused that many concepts of base totally must be reformulated and the documentation not yet reflects it.

REFERENCES

[1] Eve Maler. *OASIS Security Services TC.* http://www.oasisopen.org/committees/security/faq.php. Mar. 2005.

[2] Abhilasha Bhargav-Spantzel, Anna C. Squicciarini, Elisa Bertino. *Establishing and protecting digital identity in federation systems.* https://www.cerias.purdue.edu/tools_and_resources/bibtex_archive/archive/2005-48.pdf. 2005.

[3] T. Bray, J. Paoli. *Extensible Markup Language (XML) 1.0 (Third Edition).* http://www.w3.org/TR/REC-xml. Feb. 2004.

[4] M. Gudgin, M. Hadley, N. Mendelsohn, JJ. Moreau, H.F. Nielsen. *Simple Object Access Protocol(SOAP)1.2, (W3C).* http://www.w3.org/TR/soap12. Jun. 2003.

[5] Mi. Linden, J. Kanner *.Identity management as part of the infrastructure of higher education. CSC News (Vol,16, No.2).* http://www.csc.fi/lehdet/cscnews/cscnews2_2004.pdf. Apr. 2004.

[6] A.O. Freier, P. Karlton, P.C. Kocher. *The SSL Protocol Version 3.0.*http://www.netscape.com/eng/ssl3/draft302.txt. Nov. 1996.

[7] W. Yeong, T. Howes, S. Kille. *Lightweight Directory Access Protocol.* http://www.ietf.org/rfc/rfc1777.txt. Mar. 1995

[8] G. Good. *The LDAP Data Interchange Format (LDIF). Technical Specification.* http://www.ietf.org/rfc/rfc2849.txt. Jun. 2000

[9] Thomas Groß. *Context-based Access Control.* http://www.akiras.de/publications/papers/Gross03a.Context-based_Access_Control.Master_Thesis_01-31-03.pdf. Mar. 2003.

[10] A. Iliev, S.W. Smith. *Privacy-Enhanced Credential Services.* 2nd Annual PKI Resarch Workshop. NIST, Gaithersburg. http://www.cs.dartmouth.edu/~sws/papers/ilsm03.pdf. Apr. 2003.

[11] S. Nazareth, S.W. Smith.*Using SPKI/SDSI for Distributed Maintenance of Attribute Release Policies in Shibboleth.*Proceedings of the IADIS International Conference WWW/Internet 2004.Volume 1. 218--226 http://www.cs.dartmouth.edu/~sws/abstracts/nazareth04.shtml. Oct. 2004.

[12] Ian Goldberg. *A Pseudonymous Communications Infrastructure for the Internet, PhD Thesis.*http://www.isaac.cs.berkeley.edu/~iang/thesis-final.pdf. 2001

[13] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. *SPKI certificate theory.*IETF RFC 2693.http://www.faqs.org/ftp/rfc/rfc2693.txt. Sep. 1999.

[14] C.Platezer . Trust-based Security in Web Services. www.infosys.tuwien.ac.at/Staff/sd/DA/ChristianPlatzer.pdf. May. 2004.

[15] Moodle. *Moodle Documentation*
http://moodle.org/course/view.php?id=29. Apr. 2005.

[16] Karin van der Berg. *Finding Open Options*
http://www.karinvandenberg.nl/Thesis.pdf. Aug. 2005.

Shibboleth Project.
http://shibboleth.internet2.edu

Shibboleth Identity Provider Deployment Guide.
http://shibboleth.internet2.edu/guides/idp

Shibboleth Service-Provider Deployment Guide.
http://shibboleth.internet2.edu/guides/sp

shib-users mailing list: archive.
https://mail.internet2.edu/wws/arc/shibboleth-users

Shibboleth Development and Support Services Federation (UK).
http://sdss.ac.uk/

Blackboard
http://www.blackboard.com

Sakai
http://www.sakaiproject.com

Moodle
http://www.moodle.org

Pubcookie.
http://www.pubcookie.org

EDINA
http://edina.ac.uk/

OASIS Security Services TC.
http://www.oasisopen.org/committees/security/faq.php

RFC 2828 Internet Security Glossary.
http://www.faqs.org/rfcs/rfc2828.html

Apache HTTP Server Project.
http://httpd.apache.org

Apache Module Registry.
http://modules.apache.org.

OpenSSL: The Open Source Toolkit SSL/TLS.
http://www.openssl.org

mod_ssl: The Apache Interface to OpenSSL.
`http://www.modssl.org`

OpenSSH.
`http://www.openssh.org`

Open Source Implementation of the Lightweight Directory Access Protocol.
`http://www.openldap.org/.`

Kerberos.
`http://web.mit.edu/kerberos/www`

Role Based Access Control, National Institute of Standards and Technology.
`http://csrc.nist.gov/rbac/`

eduPerson Home Page.
`http://www.educause.edu/content.asp?PAGE_ID=949&bhcp=1`

InQueue Federation Home Page
`http://inqueue.internet2.edu/`

InCommon Federation Web site
`http://www.incommonfederation.org/`

Pubcookie.
`http://pubcookie.org`

Log for C++.
`http://log4cpp.sourceforge.net`

Xerces C++ parser.
`http://xml.apache.org/xerces-c`

Opensaml.
`http://www.opensaml.org`

XML-Security.
`http://xml.apache.org/security/index.html`

APPENDIXES

APPENDIX A: GLOSSARY

A.1 Abbreviations

AAA	Authentication, Authorization, Accounting
ACL	Access Control List
CMS	Course Management System
HEI	Higher Education Institute
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
OASIS	Organization for the Advancement of Structured Information Standards
PKI	Public Key Infrastructure
RFC	Request for Comments
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
SSO	Single Sign-on
TGS	Ticket Granting Service
TLS	Transport Layer Security
XML	Extensible Markup Language

A.2 Terminology

AAP The Attribute Acceptance Policy define the rules that map attributes and information out of the SAML attribute assertion into simpler forms that are usable by the application for policy decisions. These rules may also provide filters on who is allowed to assert certain information.

AA The Attribute Authority (deprecated) is the portion of the IdP responsible for issuing attributes on behalf of an organization. This term has been deprecated, and the AA is now the attribute query protocol handler function in the IdP.

Access Control Access control is the process of denning and enforcing (rules, procedures) to prevent unauthorized access to resources. Authentication and authorization are central parts of access control.

Access Control List Format for specification of an Access Matrix. The Access Control List is attached to a resource. It contains users, groups and permissions that they have on the resource.

Assertion A statement that is taken as being correct or true.

Authentication Authentication is the process of providing assurance regarding the identify of a subject or an object.

Authorization Authorization is the process of determining whether a particular subject has permission to manipulate a certain object in a certain way.

Attribute An attribute is an atom of information which is defined by the intersection of attribute name and attribute value. Attributes must be considered in terms of the subject about which they are asserted and the authority who is asserting the atom of information is true.

Attribute Assertion A SAML attribute assertion carries attribute information about a subject such as a browser user. In Shibboleth, the attribute assertion conveys attributes once a context is established from the IdP to the SP.

ARP Attribute Release Policies are rulesets regarding attribute release. These are combined and matrixed against the requester to compute an effective ARP. The effective ARP filters the set of attributes supplied by the directory released to a given relying party by the IdP.

Attribute Resolver The component of the IdP responsible for retrieving attributes from various data sources or computing advanced attributes and performing the necessary transformations for SAML transport.

Authentication Assertion SAML authentication assertions carry information regarding the act and results of the authentication of a principal by an authority. In Shibboleth, this is used to transport the handle or other form of name identifier to the SP for authentication or subsequent attribute request.

Certificate A certificate consists of the public key and the identity of an entity rendered unforgeable by digitally signing the entire information with the private key of the issuing Certificate Authority (CA).

Certificate Authority A Certificate Authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption.

Digital Certificate A digital certificate is an electronic means of establishing your credentials when doing business or other transactions on the Internet. It is issued by a certification authority (CA). It contains your name, a serial number, expiration dates, a copy of the certificate holder's public key (used for encrypting and decrypting messages and digital signatures), and the digital signature of the certificate-issuing authority so that a recipient can verify that the certificate is real.

Digital Signature A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document and, possibly, to ensure that the original content of the message or document is unchanged. Digital signatures are easily transportable and cannot be imitated by someone else. The ability to ensure that the original signed message arrived means that the sender cannot repudiate it later.

EduPerson This object class was defined by MACE-Dir to handle standard educational attributes in a way that would facilitate collaboration. Shibboleth is often used to transport eduPerson attributes, but the two are *distinct entities* and each can be used individually with full functionality.

Entitlement Entitlements form a specialized class of attributes important enough to call out separately. They can be used to identify specific group membership or eligibility to use a given resource. One method of deploying Shibboleth insulates the decision making logic used by the IdP from the SP by expressing entitlements instead of several individual attributes.

Handle A handle is one form of name identifier and is used in an authentication assertion to establish a referential identifier for attribute query in classic Shibboleth. The handle itself is completely opaque and temporary and should never be directly used for authentication purposes, as it corresponds only to a particular unknown browser user.

HS Handle Service (deprecated) is, the portion of the IdP responsible for handling single sign-on interoperability and functionality.

Federation A federation is a collection of organizations that agree to interoperate under a certain ruleset. Federations will usually define trusted roots, authorities, and attributes, along with distribution of metadata representing this information. Shibboleth treats federations -- representing multiple relying parties -- like single relying parties. Federations are not required for the use of Shibboleth but can facilitate exchange greatly.

IdP/Origin The Identity Provider is the authority responsible for generating and asserting authentication, authorization, and identity information about principals in a security domain.

LDAP Lightweight Directory Access Protocol, a protocol for accessing directory services. Set of protocols used to access a hierarchical directory of information on a directory server. LDAP is considered to be lightweight because it's based on a simplified version of X.500 directories. Directories may contain phone numbers, electronic mail addresses, Public Key's, computer names and addresses, or any other information that can be conveniently arranged hierarchically.

Metadata Shibboleth relies on metadata to identify and distribute trusted IdP, SP, and certificate authority information. Prior to 1.3, this took the form of sites.xml and trust.xml; now only sites.xml, based on the new SAML 2.0 metadata standards, is used.

Name Identifier There are several different name identifiers, each one representing a different meaning for the Subject field of the SAML authentication assertion, and often, a different set of flows as well. Handles are the name identifier in traditional Shibboleth flows, and actual identities or persistentID's may also be used either for attribute transport or as standalone sign-on assertions.

Principal The individual being authenticated and about whom assertions are being issued. Note the principal of an assertion is only the subject of the assertion in cases where identity is directly expressed.

Private Key A private key is a value - known only to one party - that can be used to decrypt encrypted messages, issue digital signatures and compute the corresponding public key. The private key must be kept private and must not be made publicly available. This term is most often used in the context of public key cryptography and not in the context of traditional cryptography.

ProviderID The atom of trust implementation for both the SP and IdP is the providerID of the corresponding partner in a transaction. Often this will be assigned by federations, but at other times individual providers will define their own. Common names may be used in addition to providerID's for UI purposes.

Public Key A public key is a value that can be used to effectively encrypt messages and verify digital signatures. The public key can be made publicly available, it does not contain secret information.

Public Key Cryptography Public key cryptography is the science of information security that uses *private key* and *public key* pairs for encryption, decryption and signature creation and verification. The problem of the key distribution is solved because the public key can be made publicly available, just the private key is kept as a secret. RSA is an example of a public key crypto system.

Relying Party The relying party is defined per-flow and is always the provider receiving and utilizing information from another entity in a given flow. Generally, this will be a particular SP.

Role The actions and activities assigned to or required or expected of a person or an entity.

Resource A resource in our sense can be a piece of data or a service provided by a system.

SAML Artifact A small piece of data that refers non-ambiguously to a SAML Assertion.

SAML Assertion A piece of data that formalizes an assertion about a subject’s identity or attributes. It regards either an act of authentication performed on a subject, attribute information about the subject, or authorization permission applying to the subject with respect to a specified resource.

SP/Target The Service Provider is an entity authorized to request attributes about IdP users on behalf of a relying organization.

SHAR The Shibboleth Attribute Requestor (deprecated) was the component of the SP responsible for requesting attributes about a browser user with whom a handle had already been associated, but is now a part of the SP package as a whole.

SHIRE The Shibboleth Indexical Reference Establisher (deprecated) is responsible for helping to associate a browser user with an identifier that the IdP and SP can both refer to. It is now part of the SP package.

Sign-on The process of authentication to a system.

Single Sign-on A user logs in at a Source Site and authenticates to that site. The Source Site confirms an identity of the user to other sites (Destination Sites). The user only needs to authenticate to the Source Site and does not need to identify at each Destination Site.

Transport Layer Security A protocol that generates secure point-to-point connections. The connections provide confidentiality and integrity as well as freshness and robustness measures. Unilateral or bilateral authentication is possible. The successor of SSL.

Web Service A Web Service is a self-describing, self-contained, modular application. It provides some functionality to other applications through an Internet connection.

APPENDIX B: Security Overview Solutions

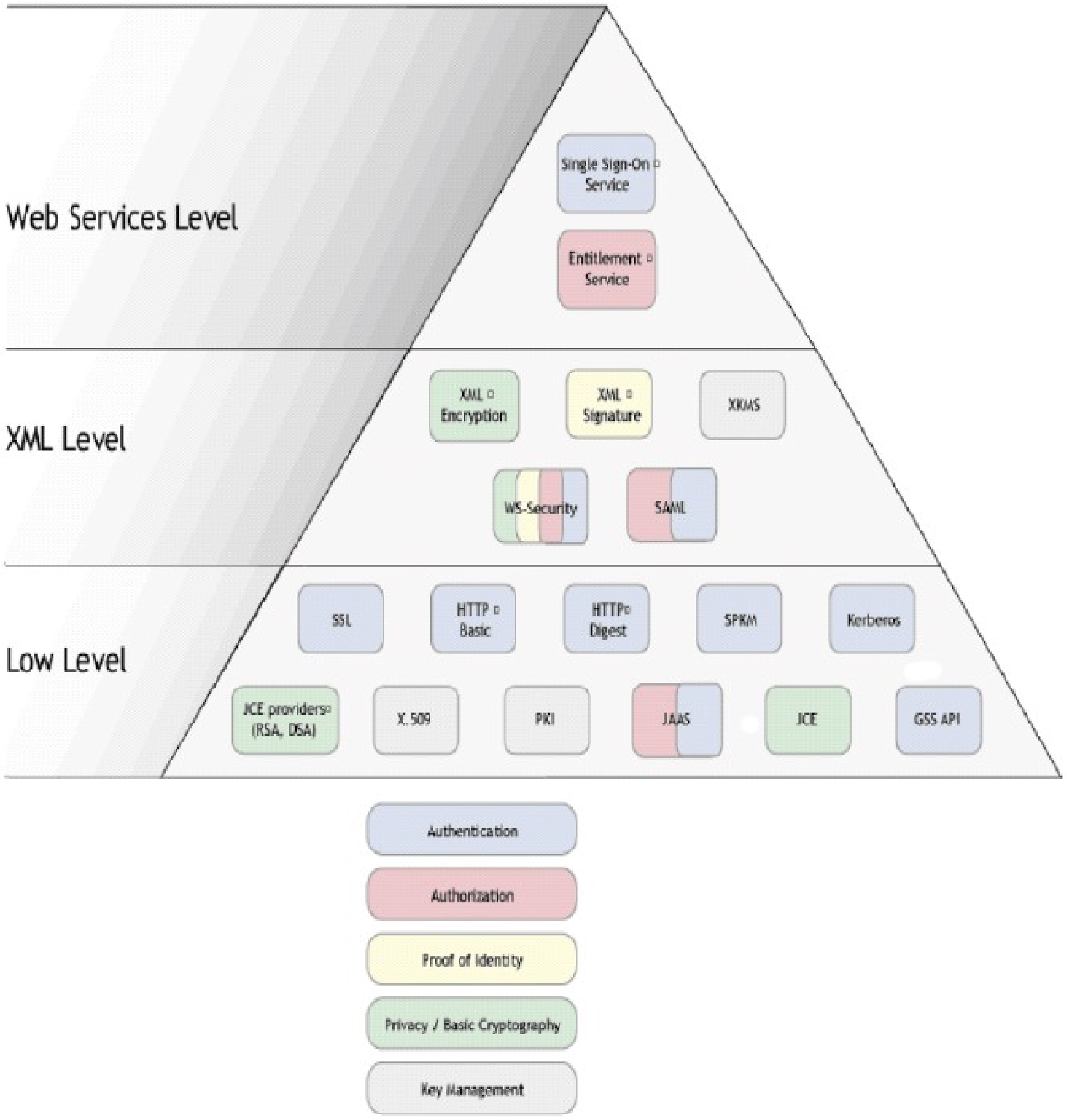


Fig. B.18: Overview of Security Solutions

APPENDIX C: SAML Assertion schema

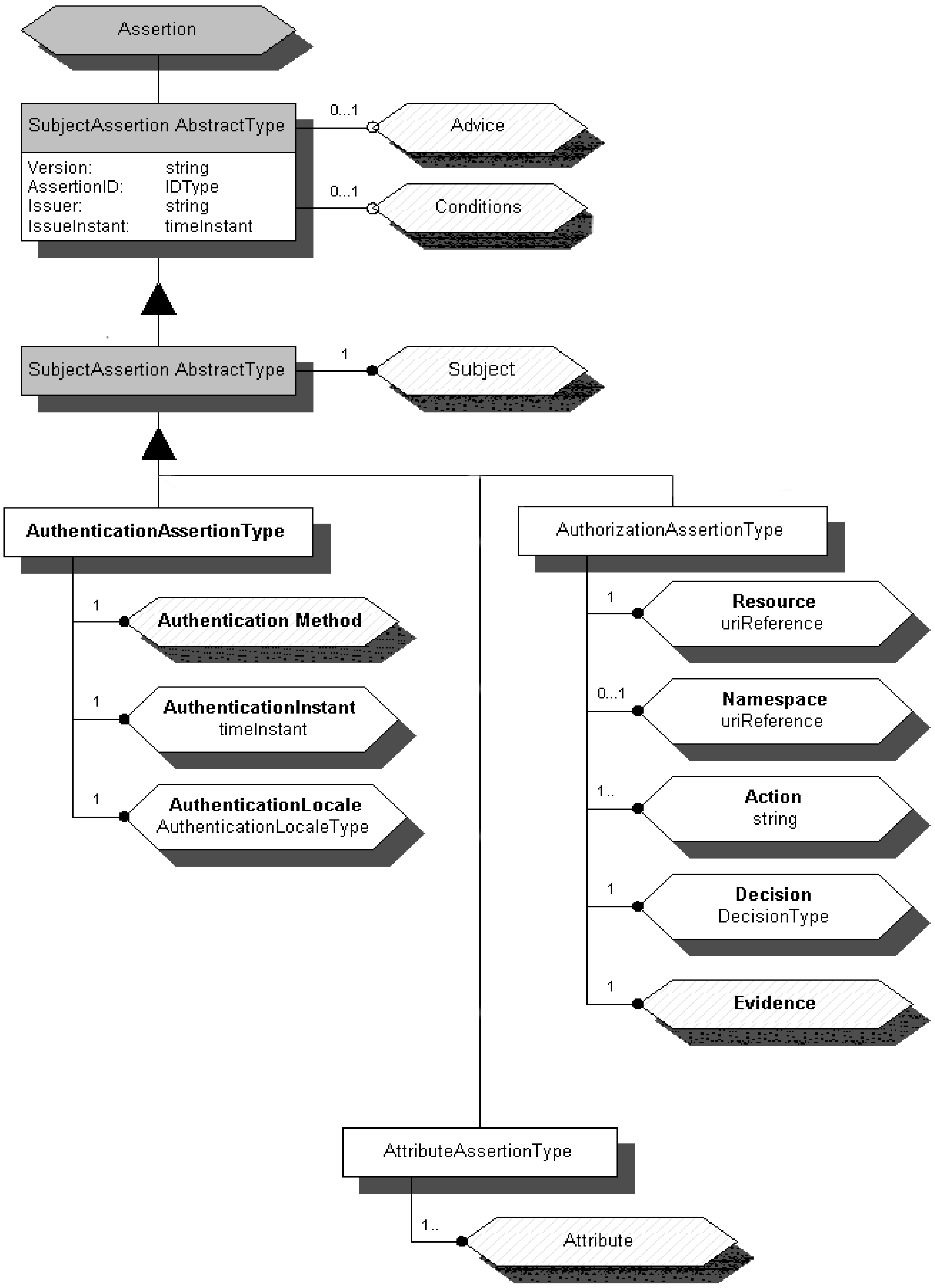


Fig. C.19: SAML Assertion Schema

APPENDIX D: Step by step Shibboleth work flow

Phase A:Connect to Resource

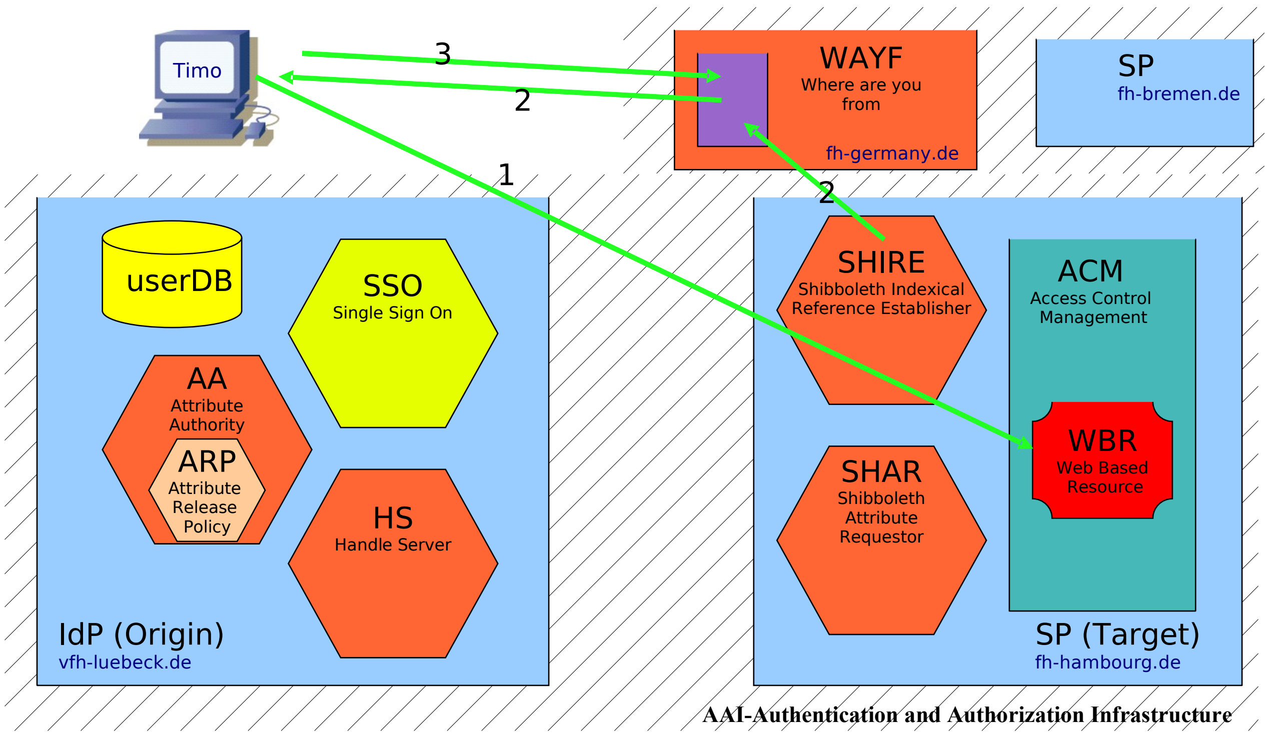


Fig. D.20: Shibboleth. Phase A

1.-The user Timo, connect her browser with a WBR (fh-hambourg.de/resource.pdf) located in another university.

2.-The server hands the request over SHIRE and redirect the webbrowser to the WAYF list (fh-germany.de/list_of_univ.html).

3.-The user selects from a list her university (vfh.luebeck.de).

Phase B:Authentication in the home University

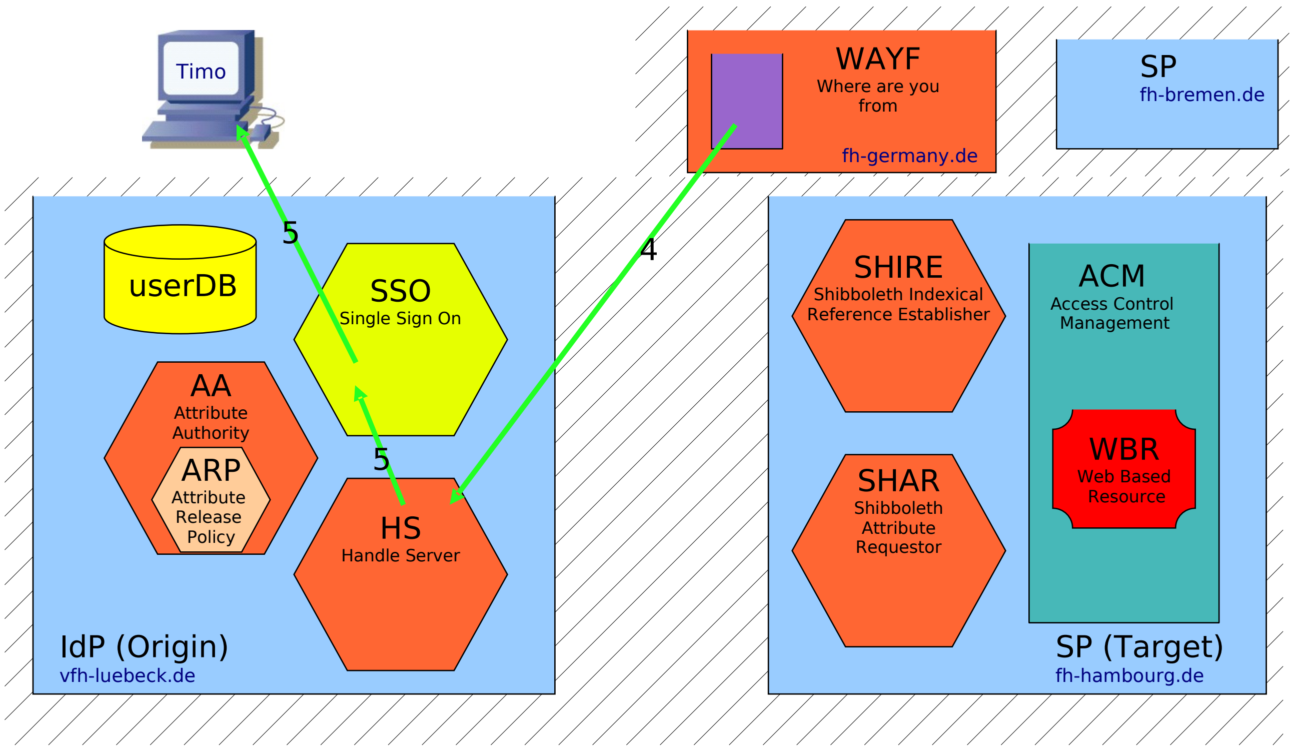


Fig. D.21: Shibboleth. Phase B

4.-WAYF redirects (depending of the option selected) the browser to the HS of the university.

5.-The HS send the login screen that Timo normally use for access to the contents of her lessons (SSO).

Phase C:Redirection to the University Resource

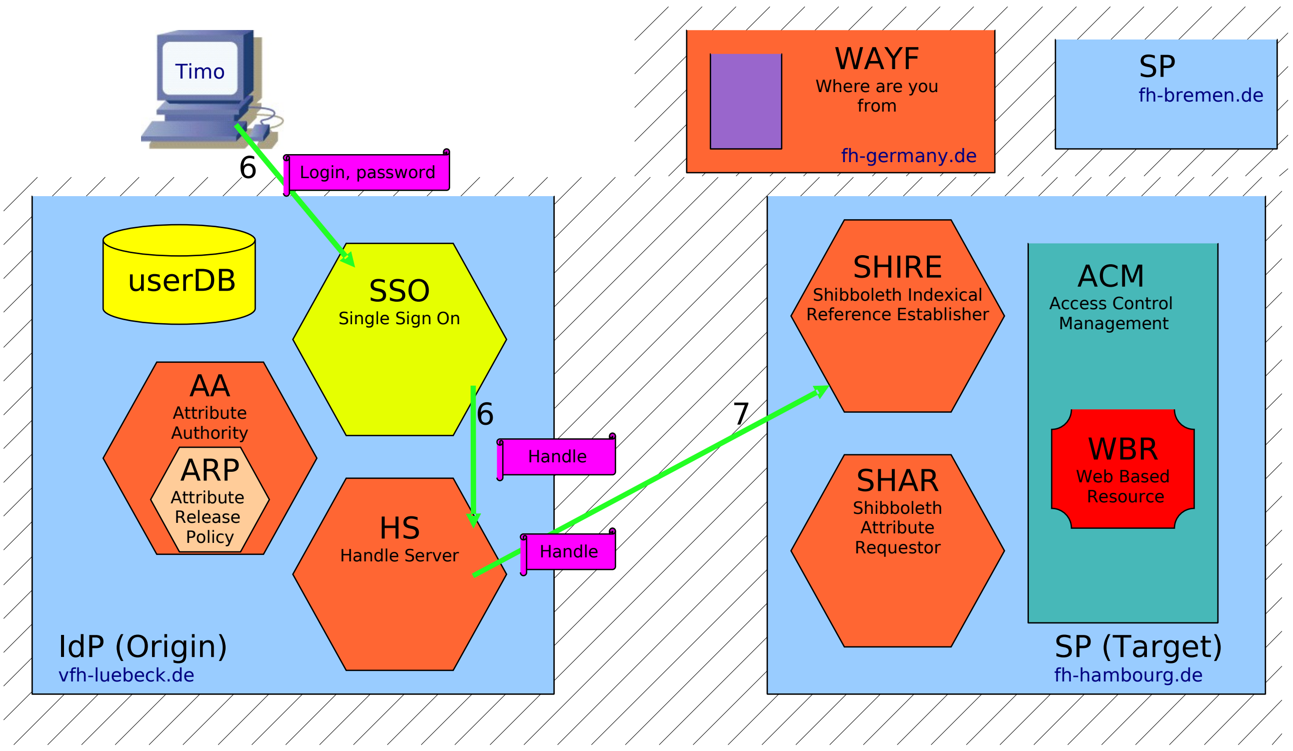


Fig. D.22: Shibboleth. Phase C

6.-Timo send to the server the credentials (login and password).

7.-If the credentials are ok the HS generate a Handle containing info about the resource (no user data) and its send to the SHIRE.

Phase D:Shibboleth Authentication

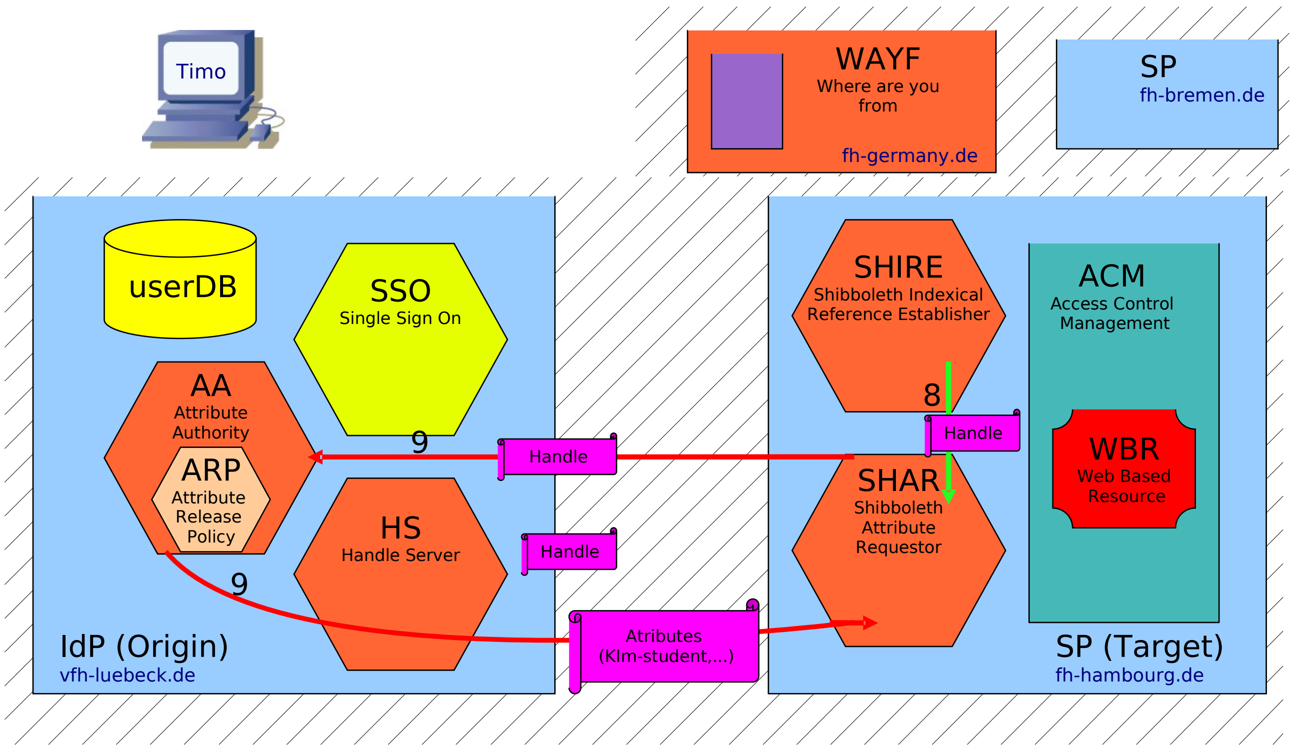


Fig. D.23: Shibboleth. Phase D

8.-The SHIRE receive the Handle and its send to the SHAR and the SHAR sends via https to the AA.

9.-The AA verifies the Handle and ask to the ARP with attributes its possible to send (are digitally signed).

Phase E:SP Authorized Access

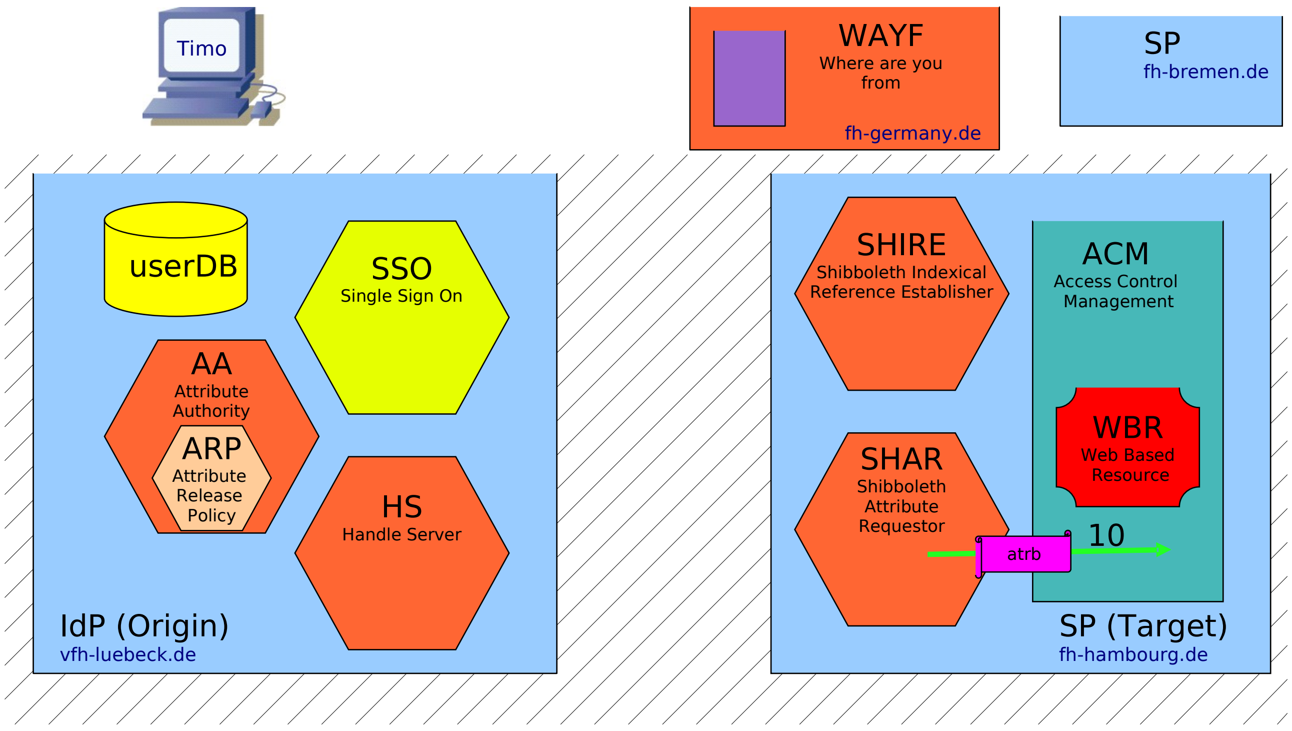


Fig. D.24: Shibboleth. Phase E

10.-The SHAR passes the attributes to the ACM ant this authorize Timo the access to the WBR (fh-hambourg.de/resource.pdf.) now, Timo can access the (authorized) resources in another university in a transparent way (until the auth. expires).

APPENDIX E: IdP Installation procedure and configuration files

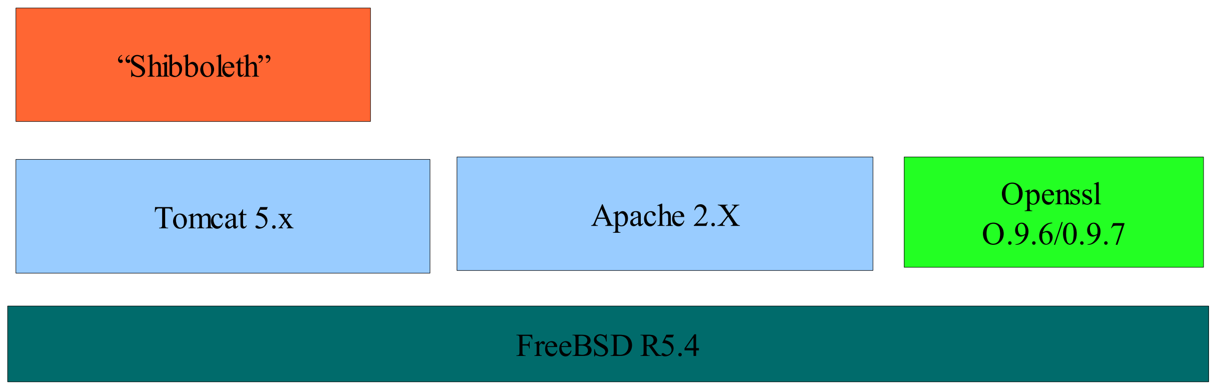


Fig. E.25: IdP Software Prerequisites

idp.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Shibboleth Identity Provider configuration -->

      <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://vfhmapc45.fh-luebeck.de:8443/shibboleth-idp/AA"
      resolverConfig="file:/usr/local/shibboleth-idp/etc/resolver.ldap.xml"
      defaultRelyingParty="urn:mace:inqueue"
      providerId="urn:mace:inqueue:fh-luebeck.de">

      <RelyingParty name="urn:mace:inqueue" signingCredential="inqueue_cred"
                  schemaHack="true">
            <NameID nameMapping="shm"/>
      </RelyingParty>


      <ReleasePolicyEngine>
            <ArpRepository
implementation="edu.internet2.middleware.shibboleth.aa.arp.provider.FileSystemAr
pRepository">
                  <Path>file:/usr/local/shibboleth-idp/etc/arps/</Path>
            </ArpRepository>
      </ReleasePolicyEngine>


      <Logging>
            <ErrorLog     level="DEBUG"     location="file:/usr/local/shibboleth-
idp/logs/shib-error.log" />
            <TransactionLog  level="INFO"  location="file:/usr/local/shibboleth-
idp/logs/shib-access.log" />
      </Logging>
      <!-- Uncomment the configuration section below and comment out the one
above if you would like to manually configure log4j -->
    <!--
      <Logging>
            <Log4JConfig location="file:///tmp/log4j.properties" />
      </Logging> -->

      <NameMapping
            xmlns="urn:mace:shibboleth:namemapper:1.0"
            id="shm"
            format="urn:mace:shibboleth:1.0:nameIdentifier"
            type="SharedMemoryShibHandle"
            handleTTL="28800"/>


      <ArtifactMapper
implementation="edu.internet2.middleware.shibboleth.artifact.provider.MemoryArti
factMapper" />

      <Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
            <FileResolver Id="inqueue_cred">
                  <Key>

      <Path>file:/usr/local/etc/apache2/ssl.certs/server.key</Path>
                  </Key>
                  <Certificate>

      <Path>file:/usr/local/etc/apache2/ssl.certs/server.crt</Path>
                  </Certificate>
            </FileResolver>

      </Credentials>

      <ProtocolHandler
implementation="edu.internet2.middleware.shibboleth.idp.provider.ShibbolethV1SSO
Handler">
            <Location>https?://[^:/]+(:(443|80))?/shibboleth-idp/SSO</Location>
      </ProtocolHandler>
      <ProtocolHandler
implementation="edu.internet2.middleware.shibboleth.idp.provider.SAMLv1_Attribut
eQueryHandler">
            <Location>.+8443/shibboleth-idp/AA</Location>
      </ProtocolHandler>
      <ProtocolHandler
implementation="edu.internet2.middleware.shibboleth.idp.provider.SAMLv1_1Artifac
tQueryHandler">
            <Location>.+8443/shibboleth-idp/Artifact</Location>
      </ProtocolHandler>
      <ProtocolHandler
implementation="edu.internet2.middleware.shibboleth.idp.provider.Shibboleth_Stat
usHandler">
            <Location>https://[^:/]+(:443)?/shibboleth-idp/Status</Location>
      </ProtocolHandler>

      <MetadataProvider
type="edu.internet2.middleware.shibboleth.metadata.provider.XMLMetadata"
                uri="file:/usr/local/shibboleth-idp/etc/IQ-metadata.xml"/>
</IdPConfig>
```

resolver.ldap.xml

```
<AttributeResolver xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mace:shibboleth:resolver:1.0"
xsi:schemaLocation="urn:mace:shibboleth:resolver:1.0 shibboleth-resolver-
1.0.xsd">
            <SimpleAttributeDefinition id="urn:mace:dir:attribute-def:cn">
            <DataConnectorDependency requires="directory"/>
      </SimpleAttributeDefinition>

      <SimpleAttributeDefinition id="urn:mace:dir:attribute-def:sn">
            <DataConnectorDependency requires="directory"/>
      </SimpleAttributeDefinition>

      <SimpleAttributeDefinition id="urn:mace:dir:attribute-def:descritption">
            <DataConnectorDependency requires="directory"/>
      </SimpleAttributeDefinition>


      <JNDIDirectoryDataConnector id="directory">
                <Search filter="cn=%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://vfhmapc46.fh-luebeck.de:389/dc=fh-
luebeck,dc=de" />
        </JNDIDirectoryDataConnector>

</AttributeResolver>
```

arp.site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<AttributeReleasePolicy    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mace:shibboleth:arp:1.0"
xsi:schemaLocation="urn:mace:shibboleth:arp:1.0 shibboleth-arp-1.0.xsd" >
      <Description>Simplest possible ARP.</Description>
      <Rule>
            <Target>
                  <AnyTarget/>
            </Target>
            <Attribute name="urn:mace:dir:attribute-def:eduPersonAffiliation">
                  <AnyValue release="permit"/>
            </Attribute>
            <Attribute                             name="urn:mace:dir:attribute-
def:eduPersonScopedAffiliation">
                  <AnyValue release="permit"/>
            </Attribute>
  <Attribute name="urn:mace:dir:attribute-def:sn">
         <AnyValue release="permit"/>
  </Attribute>
<Attribute name="urn:mace:dir:attribute-def:cn">
         <AnyValue release="permit"/>
  </Attribute>
<Attribute name="urn:mace:dir:attribute-def:description">
         <AnyValue release="permit"/>
</Attribute>
      </Rule>
</AttributeReleasePolicy>
```

APPENDIX F: SP Installation procedure and configuration files

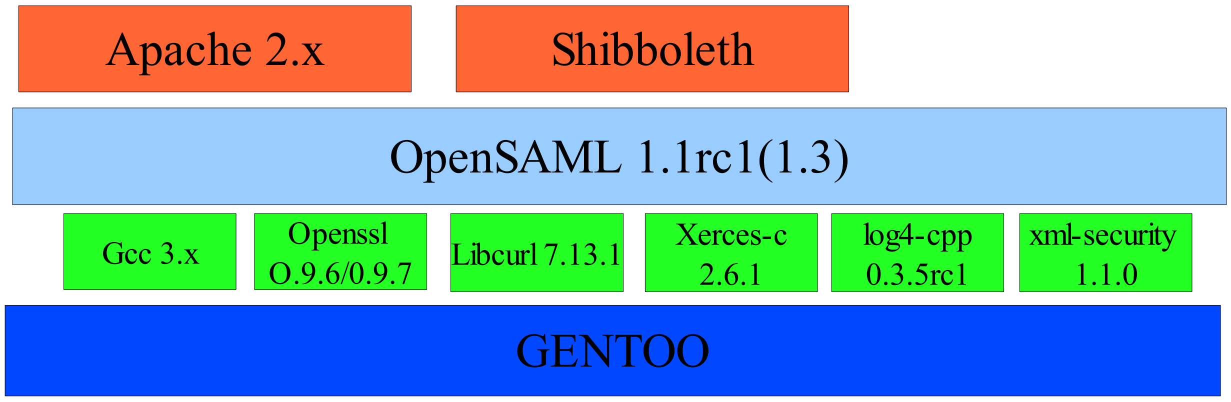


Fig. F.26: SP Software Prerequisites

Installation Procedure with gentoo

```
XERCES-C
export XERCESCROOT=<full path to xerces-c-src2_6_0>
cd $XERCESCROOT/src/xercesc
autoconf
/runConfigure -p linux -c gcc -x g++ -r pthread -b 32 -P /opt/shibboleth
make
make install

LOG4CPP
 ./configure --prefix=/opt/shibboleth --with-pthreads --disable-static
make
make check
make install

XML-Security
./configure --prefix=/opt/shibboleth --without-xalan
make
make install

OPENSAML
./configure   --prefix=/opt/shibboleth   --with-curl=/usr/lib      --with-
xerces=/usr/local/include      --with-log4cpp=/opt/shibboleth      --with-
openssl=/usr/lib -C
make
make install

SHIBBOLETH
./configure  --prefix=/opt/shibboleth  --with-log4cpp=/opt/shibboleth  --
with-xerces=/opt/shibboleth --with-saml=/opt/shibboleth  --enable-apache-
20 --with-apxs2=/usr/sbin/apxs2 -C

make
make install
```

shibboleth.xml

```
<ShibbolethTargetConfig xmlns="urn:mace:shibboleth:target:config:1.0"
    logger="/opt/shibboleth/etc/shibboleth/shibboleth.logger" clockSkew="180">

    <Extensions>
        <Library path="/opt/shibboleth/libexec/xmlproviders.so" fatal="true"/>
    </Extensions>

    <SHAR logger="/opt/shibboleth/etc/shibboleth/shar.logger">

            <!--
        <Extensions>
                    <Library  path="/opt/shibboleth/libexec/shib-mysql-ccache.so"
fatal="false"/>
        </Extensions>
        -->


        <UnixListener address="/tmp/shar-socket"/>

                   <MemorySessionCache  cleanupInterval="300"  cacheTimeout="3600"
AATimeout="30" AAConnectTimeout="15"
               defaultLifetime="1800"  retryInterval="300"  strictValidity="false"
propagateErrors="true"/>
        <!--
                  <MySQLSessionCache   cleanupInterval="300"   cacheTimeout="3600"
AATimeout="30" AAConnectTimeout="15"
               defaultLifetime="1800"  retryInterval="300"  strictValidity="false"
propagateErrors="true"
               mysqlTimeout="14400">
            <Argument>&#x2D;&#x2D;language=/opt/shibboleth/share/english</Argume
nt>
            <Argument>&#x2D;&#x2D;datadir=/opt/shibboleth/data</Argument>
        </MySQLSessionCache>
        -->
    </SHAR>

    <SHIRE logger="/opt/shibboleth/etc/shibboleth/shire.logger">
                                                                <RequestMapProvider
type="edu.internet2.middleware.shibboleth.target.provider.XMLRequestMap">
            <RequestMap applicationId="default">

                <Host name="localhost">
                                     <Path  name="secure"  requireSession="true"
exportAssertion="true">

                          <Path name="admin" applicationId="foo-admin"/>
                      </Path>
                </Host>
            </RequestMap>
        </RequestMapProvider>

        <Implementation>
            <ISAPI normalizeRequest="true">

                <Site id="1" name="localhost"/>
            </ISAPI>
        </Implementation>
    </SHIRE>
    <Applications xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
      id="default" providerId="https://vfhmapc46.fh-luebeck.de/shibboleth">


       <Sessions lifetime="7200" timeout="3600" checkAddress="true"
                wayfURL="https://vfhmapc45.fh-luebeck.de/shibboleth-idp/SSO"
            shireURL="/Shibboleth.shire" shireSSL="false"/>

<!-- WAYF      <Sessions lifetime="7200" timeout="3600" checkAddress="true"
                wayfURL="https://wayf.internet2.edu/InQueue/WAYF"
            shireURL="/Shibboleth.shire" shireSSL="false"/>
-->


       <!--
          You should customize these pages! You can add attributes with values
that can be plugged
        into your templates.
        -->
        <Errors shire="/opt/shibboleth/etc/shibboleth/shireError.html"
            rm="/opt/shibboleth/etc/shibboleth/rmError.html"
            access="/opt/shibboleth/etc/shibboleth/accessError.html"
            supportContact="humbertog.shibboleth@gmail.com"
            logoLocation="/shibtarget/logo.jpg"
            styleSheet="/shibtarget/main.css"/>


        </CredentialUse>


                                                                <AAPProvider
type="edu.internet2.middleware.shibboleth.target.provider.XMLAAP"
            <AttributeAcceptancePolicy xmlns="urn:mace:shibboleth:aap:1.0">
                                 <AttributeRule  Name="urn:mace:dir:attribute-
def:eduPersonPrincipalName" Header="REMOTE_USER" Alias="user">
                    <AnySite>
                        <AnyValue/>
                    </AnySite>
                </AttributeRule>
            </AttributeAcceptancePolicy>
        </AAPProvider>
        -->


                                                         <FederationProvider
type="edu.internet2.middleware.shibboleth.common.provider.XMLMetadata"
            uri="/opt/shibboleth/etc/shibboleth/IQ-sites.xml"/>
                                                         <FederationProvider
type="edu.internet2.middleware.shibboleth.common.provider.XMLMetadata">
                 <SiteGroup              Name="https://example.org/shibboleth"
xmlns="urn:mace:shibboleth:1.0">
                       <OriginSite
Name="https://example.org/shibboleth/origin">
                             <Alias>Localhost Test Deployment</Alias>
                             <Contact  Type="technical"  Name="Your  Name  Here"
Email="root@localhost"/>
                             <HandleService
Location="https://localhost/shibboleth/HS" Name="localhost"/>
                             <AttributeAuthority
Location="https://localhost/shibboleth/AA" Name="localhost"/>
                             <Domain>localhost</Domain>
                       </OriginSite>
                 </SiteGroup>
        </FederationProvider>

                                                                  <TrustProvider
type="edu.internet2.middleware.shibboleth.common.provider.XMLTrust"
            uri="/opt/shibboleth/etc/shibboleth/IQ-trust.xml"/>



        <saml:Audience>urn:mace:inqueue</saml:Audience>


    </Applications>


                                                            <CredentialsProvider
type="edu.internet2.middleware.shibboleth.common.Credentials">
        <Credentials xmlns="urn:mace:shibboleth:credentials:1.0">
            <FileResolver Id="defcreds">
                <Key format="PEM">
                    <Path>/opt/shibboleth/etc/shibboleth/shar.key</Path>
                </Key>
                <Certificate format="PEM">
                    <Path>/opt/shibboleth/etc/shibboleth/shar.crt</Path>
                </Certificate>
            </FileResolver>

            <!--
            <FileResolver Id="inqueuecreds">
                <Key format="PEM" password="handsoff">
                    <Path>/opt/shibboleth/etc/shibboleth/inqueue.key</Path>
                </Key>
                <Certificate format="PEM">
                    <Path>/opt/shibboleth/etc/shibboleth/inqueue.crt</Path>
                </Certificate>
            </FileResolver>
            -->
        </Credentials>
    </CredentialsProvider>

</ShibbolethTargetConfig>
```

aap.xml

```
<AttributeAcceptancePolicy xmlns="urn:mace:shibboleth:1.0">

            <AttributeRule                        Name="urn:mace:dir:attribute-
def:eduPersonScopedAffiliation"    Scoped="true"    Header="Shib-EP-Affiliation"
Alias="affiliation">
            <!--  Filtering   rule   to   limit   values   to   eduPerson-defined
enumeration. -->
        <AnySite>
            <Value Type="regexp">^[M|m][E|e][M|m][B|b][E|e][R|r]$</Value>
            <Value Type="regexp">^[F|f][A|a][C|c][U|u][L|l][T|t][Y|y]$</Value>
            <Value Type="regexp">^[S|s][T|t][U|u][D|d][E|e][N|n][T|t]$</Value>
            <Value Type="regexp">^[S|s][T|t][A|a][F|f][F|f]$</Value>
            <Value Type="regexp">^[A|a][L|l][U|u][M|m]$</Value>
               <Value Type="regexp">^[A|a][F|f][F|f][I|i][L|l][I|i][A|a][T|t][E|
e]$</Value>
                    <Value  Type="regexp">^[E|e][M|m][P|p][L|l][O|o][Y|y][E|e][E|
e]$</Value>
```

```
        </AnySite>
                        <SiteRule Name="urn:mace:inqueue:shibdev.edu">
            <Scope Accept="false">shibdev.edu</Scope>
            <Scope Type="regexp">^.+\.shibdev\.edu$</Scope>
        </SiteRule>
      </AttributeRule>

      <AttributeRule      Name="urn:mace:dir:attribute-def:eduPersonAffiliation"
Header="Shib-EP-UnscopedAffiliation" Alias="unscoped-affiliation">
        <AnySite>
            <Value Type="regexp">^[M|m][E|e][M|m][B|b][E|e][R|r]$</Value>
            <Value Type="regexp">^[F|f][A|a][C|c][U|u][L|l][T|t][Y|y]$</Value>
            <Value Type="regexp">^[S|s][T|t][U|u][D|d][E|e][N|n][T|t]$</Value>
            <Value Type="regexp">^[S|s][T|t][A|a][F|f][F|f]$</Value>
            <Value Type="regexp">^[A|a][L|l][U|u][M|m]$</Value>
               <Value Type="regexp">^[A|a][F|f][F|f][I|i][L|l][I|i][A|a][T|t][E|
e]$</Value>
                    <Value  Type="regexp">^[E|e][M|m][P|p][L|l][O|o][Y|y][E|e][E|
e]$</Value>
        </AnySite>
      </AttributeRule>

        <AttributeRule  Name="urn:mace:dir:attribute-def:eduPersonPrincipalName"
Scoped="true" Header="REMOTE_USER" Alias="user">
            <!-- Basic rule to pass through any value. -->
        <AnySite>
            <Value Type="regexp">^[^@]+$</Value>
        </AnySite>
    </AttributeRule>
      <AttributeRule      Name="urn:mace:dir:attribute-def:eduPersonEntitlement"
Header="Shib-EP-Entitlement" Alias="entitlement">
            <!-- Entitlements tend to be filtered per-site. -->


      <AttributeRule       Name="urn:mace:dir:attribute-def:eduPersonTargetedID"
Header="Shib-TargetedID" Alias="targeted_id">
        <AnySite>
            <AnyValue/>
        </AnySite>
      </AttributeRule>
      <AttributeRule  Name="urn:mace:dir:attribute-def:cn"  Header="Shib-Person-
commonName" Alias="usuari">
            <AnySite>
            <AnyValue/>
        </AnySite>
      </AttributeRule>

      <AttributeRule  Name="urn:mace:dir:attribute-def:sn"  Header="Shib-Person-
surname">
            <AnySite>
            <AnyValue/>
        </AnySite>
      </AttributeRule>
            <AttributeRule         Name="urn:mace:dir:attribute-def:description"
Header="Shib-Person-enrolled" Alias="enrolled">
            <AnySite>
            <AnyValue/>
        </AnySite>
      </AttributeRule>

</AttributeAcceptancePolicy>
```

APPENDIX G: WAYF Installation instructions

<u>To install the WAYF:</u>

```
cd /opt/src/shibboleth-origin-1.2 ant package-wayf cp dist/shibboleth-
wayf.war /usr/share/tomcat5/webapps/
```

<u>To configure it:</u>

```
cd /usr/share/tomcat5/webapps/shibboleth-wayf/WEB-INF/classes/conf/
cp IQ-sites.xml sites.xml
```

This creates a basic list of two sites taken from the test federation, which we are just putting in to make the drop-down list have more than one thing in it. To this file, add our local test origin as follows:

```
<OriginSite Name="https://example.org/shibboleth/origin">
     <Alias>Localhost Test Deployment</Alias> <Contact Type="technical"
     Name="Your Name Here" Email="root@localhost"/>

     <HandleService Location="http://morbius.iay.org.uk/shibboleth/HS"
     Name="CN=localhost, O=Shibboleth Project, C=US"/>

     <AttributeAuthority
     Location="http://morbius.iay.org.uk/shibboleth/AA"
     Name="CN=localhost, O=Shibboleth Project, C=US"/>
     <Domain>
          localhost
     </Domain>

</OriginSite>
```

At this point, it's possible to access the tomcat context directly to verify that something is happening: *http://vfhmapc48.fh-luebeck.de:8080/shibboleth-wayf/WAYF?target=a&shire=b*

To make the WAYF context live at the correct URL, add the following clause to the /etc/httpd/conf/httpd.conf in an appropriate place:

```
<Location /shibboleth-wayf>
      JkUriSet worker ajp13:localhost:8009
</Location>
```

Finally, Is needed to make a change to the configuration of the local host “federation” from the point of view of the origin, as otherwise the origin will notice a providerId discrepancy and fail requests.
To do this, edit /usr/share/tomcat5/webapps/WEB-INF/classes/conf/localhost-sites.xml and change the DestinationSite’s AssertionConsumerServiceURL. This starts out as https://localhost/Shibboleth.shire but as I was browsing to my server from another host, Is needed to include the proper machine name: http://vfhmapc48.fh-luebeck.de/Shibboleth.shire.

APPENDIX H: Shibbolize Blackboard

The following section explains how to install Shibboleth and how to set up Shibboleth with the Installation.

1. Install Blackboard Learning System (Release 6) enable OpenSSL.

2. Configure SSL for Blackboard Learning System. Save the certificate files under blackboard/apps/httpd/conf/certs/. These are formatted as .cer, .crt and .key.

3. Download the correct Shibboleth package for the operating system and install it.

4. Follow the Shibboleth v1.1 instructions to install the package. Check that the most current libraries are installed. The Shibboleth directions contain detailed instructions for updating libraries.

The institution needs a signed CA certificate, for example, from Verisign. This is the same certificate used for SSL.

CONFIGURATION:

1. Edit the blackboard/apps/httpd/conf/httpd.conf to include the /opt/shibboleth/etc/shibboleth/apache.config file. This step must be repeated when PushConfigUpdates is run. PushConfigUpdates may overwrite this setting.

2. Add the following to apache.config in the Shibboleth file system. This instructs Shibboleth to protect all files beginning with '/webapps'. The apache.confing and.ini files are located in /opt/shibboleth/etc/shibboleth

```
<Location /webapps>
AuthType shibboleth
require affiliation ~ ^member@.+$
# This rule below accepts any valid principal name passed from the
Origin.
require user ~ ^.+$
</Location>
```

3. The value of the "require" directive is dependent on the Attribute Acceptance and Attribute Release Policies for the Target and Origin, respectively. Check with the Shibboleth federation administration for details on what attributes will be released to your Target.

4. Add the following custom attributes to apache.config ShibMapAttribute urn mace dir attribute-def eduPersonPrincipalName Shib-EP-BBUSER-NAME. If you configure AJP13 as the Apache/Tomcat protocol, you may omit this value. Edit the Blackboard Tomcat server.xml to use AJP13 as the connector protocol. This should be done using the Ajp13Connector configuration. The AJP12 protocol readers in Tomcat have a bug that prevents

REMOTE_USER from being properly propagated to Tomcat from Apache. Additionally, the Coyote connectors have not been tested with Shibboleth.

For example (make sure you've disabled any other listeners that may be listening on the same port)

```
<Connector
className="org.apache.ajp.tomcat4.Ajp13Connector"
port="8009"
minProcessors="50"
maxProcessors="100"
tomcatAuthentication="false"/>
```

5. Edit /opt/shibboleth/etc/shibboleth/shibboleth.ini file to point to the correct WAYF server. Shibboleth should default to the correct location wayfURL = http //servername.blackboard.com 8080/shibboleth/HS Point to the location of the certificate file, the key file, calist and the password (omit the line breaks after the '=')

```
certfile=
/usr/local/blackboard/apps/httpd/conf/certs/server.crt
keyfile=
/usr/local/blackboard/apps/httpd/conf/certs/server.key
calist=/usr/local/blackboard/apps/httpd/conf/certs/qa-b64.cer
keypass='password'
```

6. Add PEM-encoded HS certificate to the trust.xml file in /opt/shibboleth/etc/shibboleth. This certificate is the one created as the signing certificate of the origin.

```
<KeyAuthority>
<ds: KeyInfo>
<ds: X509Data>
<ds: X509Certificate>
Add PEM-encoded HS here
..
</ds: X509Certificate>
</ds: X509Data>
</ds: KeyInfo>
<Subject>qamigl2.qa.dc.blackboard.com</Subject>
</KeyAuthority>
```

7. Change the authentication type in Blackboard the Blackboard bb-config.properties file. bbconfig.auth.type=shib

8. Uncomment all the Shibboleth Authentication Properties in the Blackboard authentication.properties file.

9. Edit site.xml file under /opt/shibboleth/etc/shibboleth to point to a valid origin server. See example below.

```
<OriginSite Name="qamigl2.qa.dc.blackboard.com">
<Alias>Blackboard QA Testing Origin</Alias>
<Contact        Type="technical"        Name="John        Doe"
Email="jdoe@blackboard.com"/>
<HandleServiceLocation="http://qamigl2.qa.dc.blackboard.com
8080/shibboleth/HS" Name="qamigl2.qa.dc.blackboard.com"/>
```

```
        <Domain>qa.dc.blackboard.com</Domain>
        </OriginSite>
```

10. Start the shar executable on the Shibboleth server `/opt/shibboleth/bin/shar -f`

11. Restart the Blackboard web services

```
/usr/local/blackboard/tools/admin/ServiceController.sh services.restart
```

Some considerations about certificates and keys:

-Certificates that are needed for Shibboleth:

-The certificate must be signed by an authority.

-If a Test Certificate is used, then the Administrator must coordinate with representatives from Shibboleth to be added to the trusted list of institutions (this is referred to as In Queue)

Users of a system that participates in Shibboleth will go through the following steps to login:

1. Click Login on the Blackboard Learning System Login page.

2. Choose the institution from the drop-down list.

3. Enter login and password information and click Login.

4. Users may enter the URL for another institution that participates in Shibboleth and enter that school’s Web site.

APPENDIX I: Shibbolize Moodle

The changes that would be necessary to extend the moodle code and file structure are (most of this is already described in moodle/auth/shibboleth/README.txt):

1. A new directory has to be created, e.g. moodle/auth/shibboleth/login (alternatively the moodle/auth/shibboleth directory could be used for that itself).

2. Within that directory there has to be a .htaccess file with

```
## Shibboleth authentication required
AuthType shibboleth
ShibRequireSession On
require valid-user
# Adapt the require statement to your needs
```

Furthermore there has to be an index.php file within that directory with the following content:

3. The moodle/login/index.php file has to be extended by:

```
if   ($CFG->shib_user_attribute   &&   $_SERVER[$CFG-
>shib_user_attribute])       {$frm->username       =
$_SERVER[$CFG >shib_user_attribute];
$frm->password = substr(base64_encode($_SERVER[$CFG-
>shib_user_attribute]),0,8);}
```

after every "$frm = data_submitted();" line. What the code actually does is to "fill" the form data with the shib_user_attribute that is used in moodle/auth/shibboleth/lib.php:auth_user_login($username, $password) to check if this user is authenticated. The password line is not really necessary, but may be useful if an admin decides to convert a shibboleth user account into a manual one

4. In the moodle/login directory there should be a .htaccess file with the following content (the statements have to be commented out per default because they may cause problems on moodle instances on webservers that don't have Shibboleth installed):

```
## Shibboleth lazy session
#AuthType shibboleth
#ShibRequireSession Off
#require shibboleth
```

5. On the login page there has to be a link to the moodle/auth/sibboleth/login directory (can be done manually by the moodle admin modifying the moodlelib strings).