

**TERENA TF-AACE**

**Deliverable B.1**

**Version 1**

**30. July 2003**

## **Definition for TF-AACE Deliverable B.1**

Investigate the different approaches to inter- and extra-institutional Authentication and Authorization, analyzing the alternatives in architecture and protocols. Produce a report on these alternatives

This document is mainly based on the three independantly written documents [A-Select], [Olsen] and [SWTCH] with updates and extenstions applied where available.

The main contributors to this document were:

- Ali Odaci, Alfa & Ariss (NL)
- Cato Olsen, UNINETT (NO)
- Diego Lopez, RedIRIS (ES)
- Ingrid Melve, UNINETT (NO)
- Maarten Koopmans, SURFnet (NL)
- Thomas Lenggenhager, SWITCH (CH)
- Torbjörn Wiberg, Umeå Universitet (SE)

Send comments to the TERENA TF-AACE mailing list: <http://www.terena.nl/tech/task-forces/tf-aace/#maillist>

# Table of Contents

1	Introduction.....	1
1.1	Overview.....	1
1.2	Problem Description for Authentication and Authorization .....	1
1.3	AAI Model.....	2
1.4	Benefits of an AAI .....	3
2	Architecture Descriptions.....	5
2.1	A-Select .....	5
2.1.1	System Architecture .....	5
2.1.2	Evaluation .....	6
2.2	FEIDE .....	6
2.2.1	Architecture / System Design .....	6
2.2.2	Evaluation .....	8
2.3	FEIDHE .....	8
2.3.1	Architecture / System Design .....	8
2.3.2	Evaluation .....	9
2.4	Grid Security Infrastructure (GSI).....	9
2.4.1	Architecture / System Design .....	10
2.4.2	Evaluation .....	10
2.5	PAPI.....	11
2.5.1	System Architecture .....	11
2.5.2	Evaluation .....	12
2.6	Shibboleth .....	13
2.6.1	Architecture / System Design .....	13
2.6.2	Evaluation .....	15
3	Comparison of AAI Components.....	16
3.1	Aspects.....	16
3.1.1	Architecture .....	16
3.1.2	Implementation.....	16
3.1.3	Security.....	16
3.1.4	Privacy .....	17
3.2	Reviewed Systems .....	17
3.2.1	Architecture .....	17
3.2.2	Implementation.....	18
3.2.3	Security.....	19
3.2.4	Privacy .....	20
4	Interoperability.....	21
5	References .....	22

# 1 Introduction

## 1.1 Overview

This document is structured in four chapters. The first introduces into the problem space of inter-organizational Authentication and Authorization (AA) concluding with a description of a generic model. Chapter 2 presents a number of AA architectures with briefly pro/cons statements. Chapter 3 is an approach to compare some of the architectures from chapter 2 according to a list of questions, which might get asked during a proper evaluation process selecting an AA system satisfying the own requirements. Finally, the last chapter looks at future interoperability of AA systems.

## 1.2 Problem Description for Authentication and Authorization

In order to authorize access to a single Resource (e.g. information on a web server, e-learning application, library catalog etc.), the following generic interactions between a User and the Resource occur:

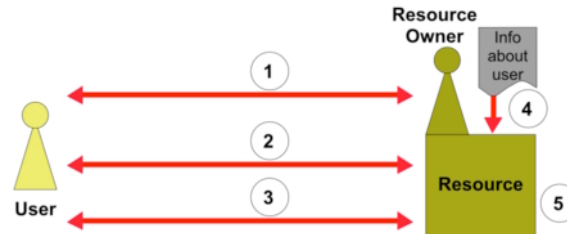


Figure 1: Authorizing access to a single Resource

- 1) A User (e.g. student, employee, library user, but also application) who wants to access a Resource has to register with the Resource Owner. The Resource Owner creates a virtual identity for this User, stores the necessary information about him/her and provides an identifier, like a login name, as well as credentials to the User. Later on, the User may use identifier and credentials to authenticate him-/herself to the Resource.
- 2) The registered User wants to access a Resource and submits an access request to the Resource, claiming the virtual identity by identifying him-/herself with the identifier.
- 3) The Resource asks the User to authenticate him-/herself (i.e. to provide the credentials belonging to that virtual identity).
- 4) After checking the credentials, the Resource retrieves previously stored information about the User and,
- 5) Based on this information, decides whether the access is authorized or not.

The disadvantage of this approach is that it does not scale if a User wants to have access to large numbers of different Resources:

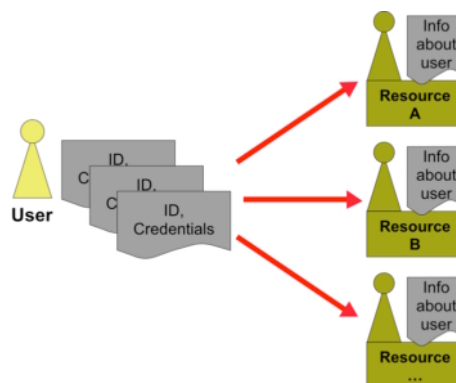


Figure 2: Accessing large numbers of Resources

Registration: The User has to register with each Resource. This is the case even if many Resource Owners do not need to know the exact identity of a User (e.g., Resources which grant access to all students of a particular university only need to know if a User belongs to this university or not).

The Resource Owner has to make sure that the information about its Users is always correct (e.g. to know if a person is still a student or not).

Authentication: The Resources themselves may use different technologies to authenticate Users. Therefore, a User has to be able to handle different authentication technologies (e.g. password-based, certificate-based, smart-card-based, etc.) and for each technology at least one identifier, but often more than one.

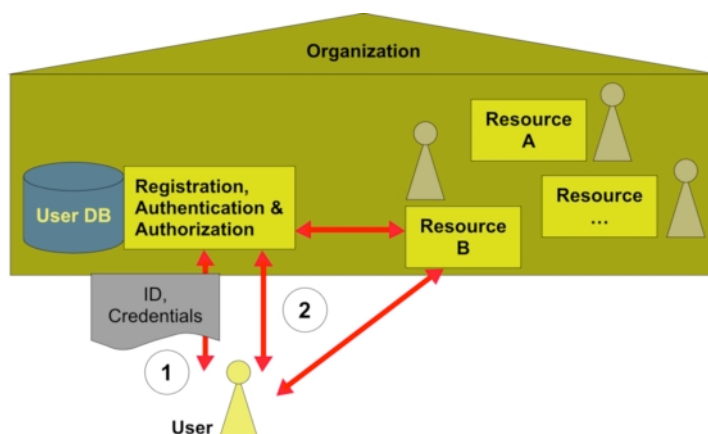


Figure 3: Granting access to Resources with centralized registration, authentication and authorization

Recently, larger organizations have started to implement local authentication and authorization infrastructures for their Users and Resources:

- 1) Centralized User registration and storage of Authorization Attributes in a central User database
- 2) Authentication and authorization interactions between User, Resource and central infrastructure of the organization

This approach solves the authentication and authorization issue for Resources belonging to the User's Home Organization, but it is no solution for authentication and authorization across organizations.

- Users of one organization would like to be authorized to use Resources of other organizations (without having to register with each of these organizations).
- Organizations would like to open their Resources to (some) registered Users of other organizations in a controlled way, without having to register all these "foreign" Users by themselves.

These issues will be dealt with in the following. It goes without saying that already existing infrastructures of organizations have to be taken into account.

### 1.3 AAI Model

A solution to the problem of inter-organizational authentication and authorization is the implementation of an AAI (Authentication and Authorization Infrastructure). The core functionality of an AAI is to tightly couple together the three basic interactions between a User, his or her Home Organization and a Resource during the authentication and authorization process. These three basic interactions are:

- 1) User authentication, which is always carried out by the User's Home Organization;
- 2) Authorization Request; and
- 3) Delivery of Authorization Attributes from the Home Organization to the Resource.  
The set of Authorization Attributes, which is transmitted to an Authorization System, has to be configurable and extendible, depending on the needs of the Resource Owner and respecting the restrictions from the data protection law(s).

In order to describe the functionality of an AAI, the following generic model will be used:

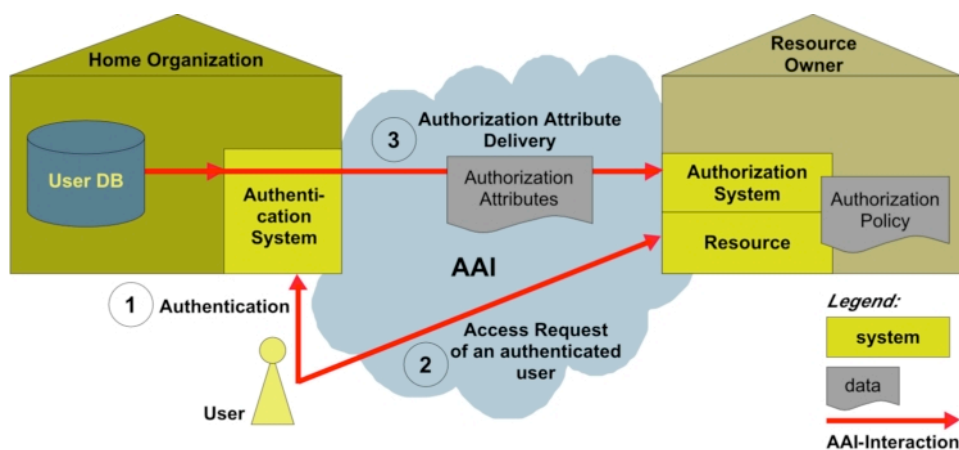


Figure 4: Generic functional model of an AA system

The terms introduced in Figure 4 as well as in this chapter are defined as follows:

<i>Home Organization</i>	Representative of a User community, e.g. university, library, university hospital etc. <ul style="list-style-type: none"> <li>• It registers Users and stores information about them</li> <li>• It is able to authenticate Users</li> </ul>
<i>User</i>	Registered member of a Home Organization
<i>User-DB</i>	Database storing information about a registered User, maintained by the Home Organization
<i>Authentication system</i>	System which can authenticate a previously registered User
<i>Resource</i>	Application, web site
<i>Resource Owner</i>	Entity owning a Resource and offering access to authorized Users
<i>Authorization System</i>	On behalf of the Resource, it takes the authorization decision based on the Authorization Policy and the Authorization Attributes received
<i>Authorization Policy</i>	Formal description of who shall get authorized by the Authorization System.
<i>Authorization Attributes</i>	Information about the User needed for the authorization decision
<i>AAI-related systems</i>	Resources, registration and authentication systems which will interact within the AAI and are a prerequisite to use the functionality of the AAI
<i>AAI core systems</i>	Systems which provide the core functionality of the AAI

After having received the authentication acknowledgement and the Authorization Attributes from the User's Home Organization, the Authorization System authorizes on behalf of the Resource Owner, based on the Authorization Policy.

## 1.4 Benefits of an AAI

By implementing an AAI, one can imagine benefits in the following areas:

- Enabling students' mobility: AAI is a required building block for all resources, which shall be shared between students of different institutions. Therefore, an AAI will enable initiatives that promote cross-organization studies, or in the future even international studies, and might be supportive implementing the European Bologna Agreement.
- Better information protection: Today, securing information access is often skipped, because it is too complicated or expensive. An AAI will offer a standardized procedure of authenticating and authorizing resource access. Therefore, Resource Owners can concentrate on protecting their assets, because they do not have to implement registration and authentication procedures themselves.
- Support for nomadic Users: Today, Users expect to be able to access from anywhere the resources they are allowed to use, not just from a workplace at their university. An AAI will enable Resource Owners to define their Authorization Policy based on personal attributes of Users and not based on IP addresses.

- Improved User convenience: After registering once, Users will be able to access many resources with a single authentication technology.
- Improved IT efficiency: IT organizations can share their security knowledge and profit from common AAI developments and from standardizations. This will make the implementation of AA functionality within their organization more efficient.

Notice that there are not only benefits that can be derived from an AAI, but there are also considerable risks involved in *not* acting:

- Growing registration overhead: increased mobility puts additional load on User registration, namely to register remote Users from outside the resource hosting organization. The inability to handle this additional load can lead to loss of potential customers, or to additional security risks due to simplified, incomplete or lacking access control to protected resources.
- Isolation: ease of access to remote resources as well as easy access from outside to local resources in a controlled way is a crucial precondition to be part of a community that increasingly relies on virtualized, distributed resources. It is also crucial to be able to form teams across organizational and international boundaries.
- Image problem: complicated access procedures to networked resources will be perceived as outdated. This can be very damaging to an organization's reputation, particularly for technology-oriented ones.
- Synchronization problems: with increasing User communities in our separate systems it is increasingly difficult to keep authentication and authorization information synchronized and updated.

## 2 Architecture Descriptions

### 2.1 A-Select

A-Select<sup>1</sup> is middleware that offers a uniform User authentication service to Web applications within an organization or for cross-organizational authentication where A-Select can authenticate Users of allied organizations. A-Select is completely web based and does not require any additional software for Users to install.

A-Select has been designed and implemented by Alfa & Ariss for SURFnet. A-Select is freely available for non-profit organizations worldwide. Other organizations need to acquire a commercial license.

#### 2.1.1 System Architecture

The A-Select systems consists of several components that communicate with each other in some way.

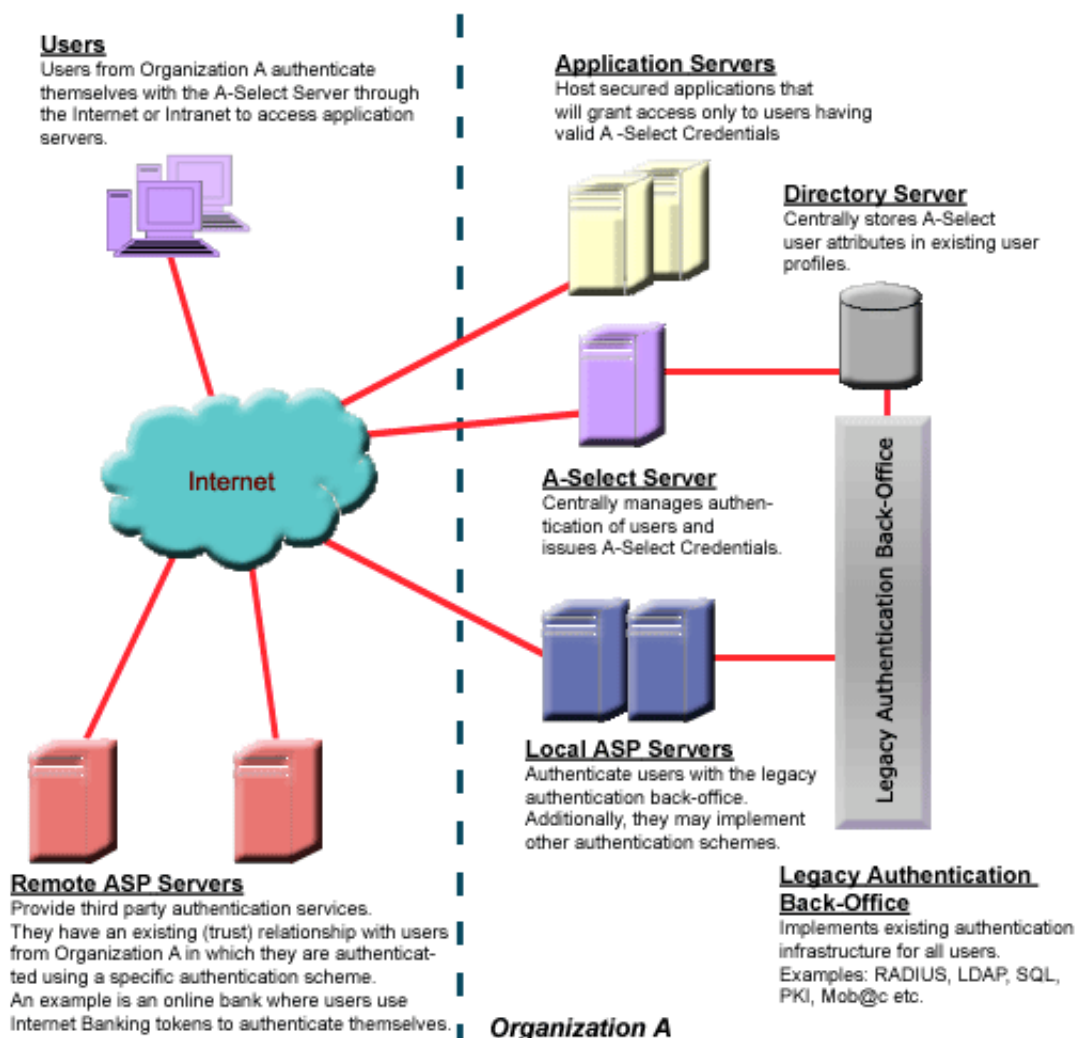


Figure 5: Architecture of A-Select

- **A-Select Aware Applications**  
They are Web applications that implement access control according to the A-Select security and trust model. Applications may with an A-Select Agent or are protected by an A-Select web server filter. In that case the application doesn't need to interact with A-Select components.
- **A-Select Agent (not included in figure 5)**  
This is a lightweight server or daemon that runs on the application server and implements convenient functionality for applications. The A-Select Agent itself offers an API. It implements session management and generation of application tickets.

<sup>1</sup> <http://a-select.surfnet.nl/>

- **A-Select Server**  
It authenticates Users in a transparent manner. The A-Select Server has a database in which information is stored about Users and how they can be authenticated. Applications redirect a User who has to get authenticated to the A-Select Server. Once the User is authenticated, the A-Select Server will issue the User credentials and redirect him/her back to application. The A-Select Server does not authenticate Users itself but rather redirects them to an A-Select Authentication Service Provider (ASP). The A-Select Server maintains a registry of all ASP servers and also knows which ASP can authenticate which Users.
- **A-Select Authentication Service Provider (ASP)**  
It is a server that knows how to authenticate a User in some manner. Typically, an ASP offers a Web front-end for traditional authentication systems like RADIUS, LDAP authentication etc. ASPs may be local or remote.

### 2.1.2 Evaluation

Pros:

- No requirements to end user software and hardware, besides a web browser with cookie support.
- Supports common authentication methods and can be extended. 6 authentication methods are supported out of the box.
- A web server filter can be used to integrate applications without any custom integration. Optionally, applications can be easily integrated using a Java or socket based API to provide SSO. Plug-ins for common applications (Citrix, web mail, Blackboard etc.) are available.
- Based on standard technology: Java, SSL, Apache and IIS.
- Easy to deploy.

Cons:

- Simple federated model.
- Requires a simple web server filter to be installed

## 2.2 FEIDE

FEIDE<sup>2</sup> is the federated electronic identity system for Norwegian education.

### 2.2.1 Architecture / System Design

FEIDE (Federated Electronic IDentity for Education) is a project with the goal of establishing a common electronic identity for Norwegian academic Users. The three main parts are

- User administration and description of inter-institutional roles
- Common access control for Internet services with well defined authentication and authorization mechanisms
- Secure electronic ID built on public key encryption

<i>System Component</i>	<i>FEIDE Tasks</i>	<i>Relationship to other system components</i>
<i>Client</i>	Software running on the User's own computer. Acquires the User's ID and accreditation. Examples: Web Client, Smart Card application	The User has credentials: he or she either knows the username/password or can prove the possession of the private key corresponding to the public key in the certificate. The system trusts that the User does not share these with others.
<i>BAS</i>	User management system for each participating institution. Collects and collates information about all Users from administrative systems. Exports information to internal systems (examples: NIS for log on to UNIX machines and ActiveDirectory for the use of Windows 2000) and AT.	BAS knows everything about the User, based on authoritative information from administrative systems. BAS exports information to AT.

<sup>2</sup> <http://www.feide.no/index.en.html>

<i>System Component</i>	<i>FEIDE Tasks</i>	<i>Relationship to other system components</i>
<i>Network Resource</i>	Requires authorization, needs role to provide access. Examples: web publishing, intra-web portal, LMS, BIBSYS, network access points	Authentication takes place outside the Network Resource, directly from client to AT (via AT-index). The client presents its signed permits to the Network Resource.
<i>Web Authentication Server (Moria)</i>	Web service for authentication and distribution of information about Users based on AT.	Network Resource redirects Client for authentication; Network Resource collects information about the User. Moria talks to AT-index and AT.
<i>Authentication Server (AT)</i>	Provides authentication information based on the User's unique ID. Example: LDAP-server	All units trust this. Updating takes place from BAS. AT exports indexing information to AT-index. AT is standard interface from BAS indicating which Users are affiliated with the institution
<i>Authentication Index (AT-index)</i>	Forwards information to proper AT based on unique ID. Example: LIMS	All units trust this. Updated from each individual AT.

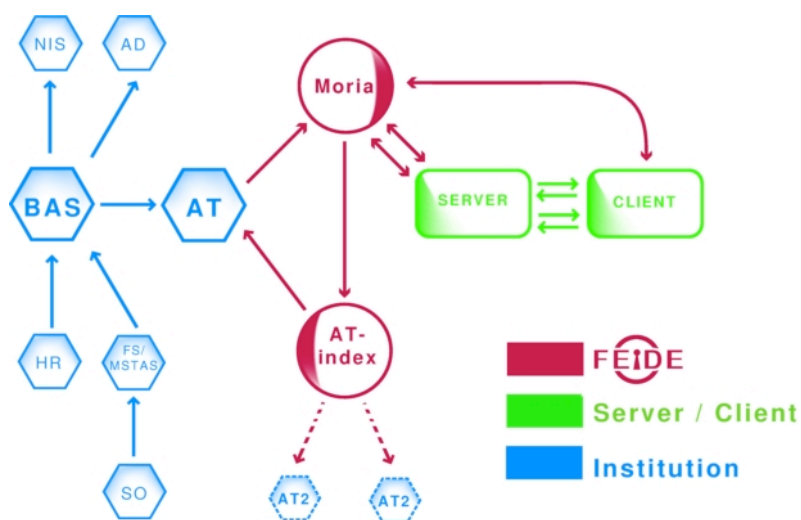


Figure 6: Architecture of FEIDE

In FEIDE each academic institution has its own User management system (BAS) for their students and employees, but services at one institution should be able to allow access to a User from another institution without importing that User into its own User management system. FEIDE requires access to an LDAP front end (AT) of the User management system.

The Moria web authentication service implementation is based on the User being redirected to the FEIDE login system when the web Resource requests authentication. The authentication consists of two steps:

- 1) Establishing an authentication session and redirect of User to the login page
- 2) Web Resource collects data about the User when s/he returns after being authenticated

<i>System Component</i>	<i>Information Accessible</i>
<i>Client</i>	The User governs the client's use of credentials. <ul style="list-style-type: none"> <li>• The User may enter username/password.</li> <li>• Certificates/keys are stored on cards or in software.</li> <li>• Biometric information resides with the User.</li> </ul>
<i>BAS</i>	Has access to all the institution's internal User information Generates information to AT.
<i>Network Resource</i>	Obtains information about whether the User was authenticated. Does not see credentials or other information about the User without explicit permission.
<i>Moria</i>	Sees information about Users when passed to Network Resource. Sees credentials when provided by the Users.

<i>System Component</i>	<i>Information Accessible</i>
<i>AT</i>	Knows whether User is valid or not. Stores information about the institution's Users.
<i>Authentication Index (AT-index)</i>	Mapping between unique ID and which AT is authoritative thereof.

### 2.2.2 Evaluation

Pros:

- Software available as open source
- User's Home Organization is responsible for the authentication.
- Designed for multiple protocols
- Persistent ID available

Cons:

- Central components required (LDAP-index and web login server)
- Short deployment period
- Persistent ID for persons

## 2.3 FEIDHE

FEIDHE (Electronic Identification in Finnish Higher Education) was a joint project for higher education in Finland. The purpose of the project was to investigate possibilities for implementing a smart card based electronic identification system in institutions of higher education. The universities and polytechnics, the national student unions of both sectors (the National Union of Finnish Students and the National Union of Polytechnic Students), and CSC, the Finnish center for high-performance computing and networking owned by the Ministry of Education were involved in the project.

What activities were involved?

Smart card based electronic identification is based on a technology that uses the public key infrastructure (PKI) by utilizing smart cards. This technology enables reliable authentication of network Users, as well as the use of electronic signatures with documents, and document security. The project involved studying and testing the applicability of the technology to existing academic systems. The most visible and extensive activities in the project were the nine pilot projects with almost 1000 pilot Users in different educational institutions. The Users tested the technology of smart card based electronic identification when they use the network services provided by the universities and polytechnics. The majority of the testing took place between September and December 2001.

What were the targets?

The target of the project was to collect data and practical experience on smart card based electronic identification technology, its functional performance, usability, and possibilities as an integrated part of the academic data network. An additional aim was to gain understanding of

- How the system and the services should be constructed
- The general implications of implementing the EID technology
- What Resources are needed to build the system

The academic community will use the information gained during the project when they make decisions regarding the electronic identification system.

Links to articles, presentations and a summary can be found on the web page<sup>3</sup>.

*Current status of FEIDHE:*

The project concluded in March 2002.

### 2.3.1 Architecture / System Design

The main goal of FEIDHE is to clarify the use of PKIs in terms of:

- Technical implementation;
- Costs: of components (commercial / open source) and of integration in each organization;
- Usability;
- Large-scale deployment.

<sup>3</sup> <http://www.csc.fi/suomi/funet/middleware/english/feidhe.phtml>

Nine pilots have been realized, all using smart cards. The smart cards are in general used for storing a certificate. One of the pilots also implemented digital signatures. In eight pilots, commercial certification authorities are used (there is not mandatory use of one specific certification authority).

The major components of FEIDHE are:

<i>Application</i>	Typically a Web Browser. But any application can make use of the infrastructure.
<i>PKI Client</i>	Provides generic access methods to applications for retrieving information from the smart card
<i>SC base components &amp; SC reader driver</i>	Smart-Card related components
<i>SC reader</i>	Terminal for reading smart cards
<i>Application server</i>	Typically a Web browser, but can be any server
<i>PKI server</i>	Module that validates User's certificates, handles Certification Revocation Lists (CRL), etc.

Figure 7 illustrates the relationships between the FEIDHE core components:

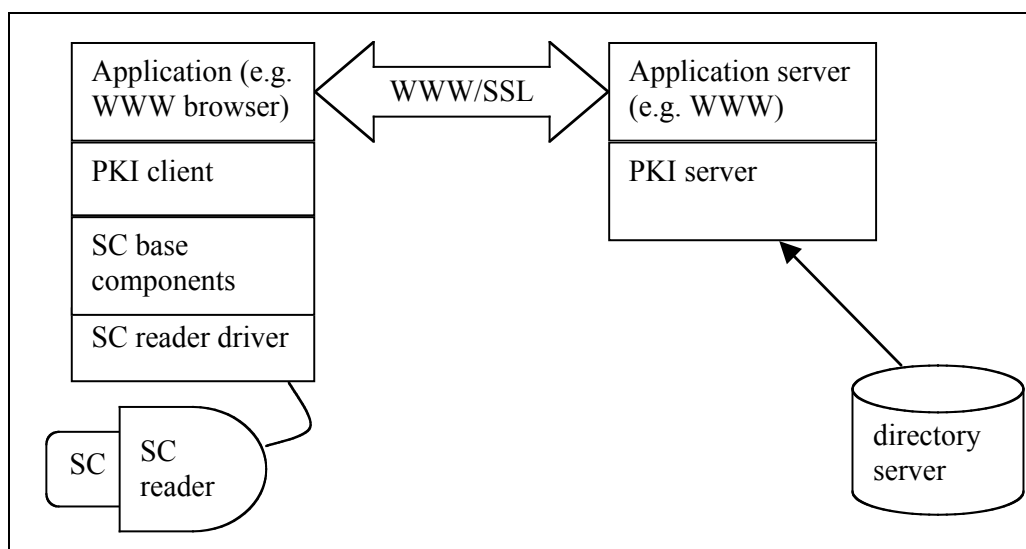


Figure 7: Architecture of FEIDHE

### 2.3.2 Evaluation

The main goal of the FEIDHE project is to evaluate the use of PKIs along with smart cards for authenticating Users. With this objective in mind, the pro and cons can be summarized as follows:

Pros:

- Interesting experience in deploying and using PKIs and smart cards.

Cons:

- Major drawback: smart cards are a mandatory requirement.
- The primary goal is User authentication only. Authorization management may be handled in some sites but is not a key element of the project.

Evaluation:

FEIDHE aims to prepare for the introduction of smart cards and digital certificates in the Finnish higher education environment. It is built around a global PKI infrastructure, which is why we will not pursue it any further.

## 2.4 Grid Security Infrastructure (GSI)

GSI<sup>4</sup> was primarily designed for the use in computational grids and is based on public key cryptography using X.509 encoded User and server certificates employing SSL/TLS transport. It is implemented in the open source Globus Toolkit<sup>TM</sup>, currently at version 3.0<sup>5</sup>.

<sup>4</sup> <http://www.globus.org/security/>

### 2.4.1 Architecture / System Design

Public key cryptography requires the availability of each party's private key during a communication. Since private keys should be kept secure, one should not leave a private key unprotected (i.e. unencrypted). For automated distributed computing, this is a showstopper – you are not able to decrypt a private key without the secret known by the owner only. Therefore, GSI makes heavy use of delegated proxies acting on behalf of Users and Resources. Such proxies do not use the original long-lived certificates, but their own short-lived ones. Because they are short-lived, their private keys do not need to be protected as vigorously as those of long-lived ones and automated processing becomes possible.

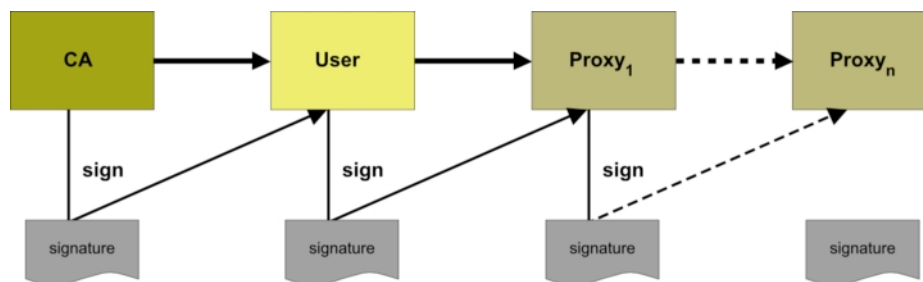


Figure 8: GSI proxy certificate chain

Proxy certificates are not signed by a CA, but by the private key of the issuing User or Resource certificate. In addition they can even generate new proxies on their own, e.g. for spawned sub-processes. That way, these proxies can be authenticated and they can securely communicate without involving the real User. Through the chain of certificates, each proxy can be followed back to the original certificate signed by a CA.

When a User wants to access a Resource, the Resource authenticates the User based on the provided certificate and applies a global-to-local mapping in order to be able to execute the User request in the local environment.

The whole delegated proxy system is rather complex since these proxy certificates include information on the time-to-live and their rights (e.g. are they allowed to spawn off new proxies; to which part of a disk do they have write access).

An article in *IEEE Computer* mentions the use of MyProxy entities needed when Users should communicate with web portals, because the standard web browsers are not able to cope with delegate proxies.<sup>6</sup> One implementation for MyProxy exists, currently at version 0.5.5<sup>7</sup>.

Example of GSI usage for web access:

A User U with certificate UC, affiliated to origin site O, wants to access a web-based Resource R located at some remote site via the web portal server WP.

- U generates a proxy UP based on his/her certificate UC and protects it with the password P. The proxy UP gets delegated to the trusted host TH that runs a myproxy-server for the portal server WP (or for more than one such server at the remote site).
- U connects via HTTPS to WP and provides the password P. WP connects to TH and by providing password P it is able to generate a new delegated proxy UP2 of UP for the use on the web portal server WP.
- U selects the Resource R in the web portal WP. WP authenticates towards R with the delegated proxy UP2 without having to further involve User U.

### 2.4.2 Evaluation

Pros:

- Wide deployment in GRID community, mainly for distributed high-performance computing.
- Based on standards (X.509).
- Software is open source.

Cons:

<sup>5</sup> <http://www.globus.org/toolkit/>

<sup>6</sup> A National-Scale Authentication Infrastructure. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch. *IEEE Computer*, 33(12):60-66, 2000. <http://www.globus.org/documentation/incoming/butler.pdf>

<sup>7</sup> <http://grid.ncsa.uiuc.edu/myproxy/>

- The architecture is primarily designed for distributed high-performance computing. It is not easily applicable to web-based Resources.
- Missing framework for data protection law compliant access to Authorization Attributes.
- GSI requires a standardized authentication infrastructure (X.509 certificates).
- Requires a trusted CA infrastructure.
- Global-to-local mappings need co-ordination, not very scalable.

Evaluation:

The many cons of GSI do not justify further investigation of this technology for AAI purposes. However, it will have its important role in the area of distributed high-performance computing

## 2.5 PAPI

PAPI (Point of Access to Providers of Information) is a system for providing access control to restricted web-based information Resources across the Internet. It intends to keep authentication as an issue local to the User's Home Organization, while leaving the information providers full control over the Resources they offer.

The authentication mechanisms are designed to be as flexible as possible, allowing each organization to use its own authentication scheme, maintaining User privacy, and offering information providers the attributes required for access control decisions. Moreover, access control mechanisms are transparent to the User and compatible with the most commonly employed Web browsers, i.e., Netscape/MSIE/Lynx, and any operating system.

PAPI has been designed and is being developed by a small team from the Spanish national research network RedIRIS. Descriptions and the product itself can be found at RedIRIS.<sup>8</sup>

*Current status of PAPI:*

The two latest versions (the current 1.2.1, was released in April 2003) introduce new features regarding privacy aspects. The AS can now be configured to send individual assertions about a User to each (G)PoA (see 2.5.1 System Architecture below). They are built based on the configuration attributes 'papiQualifiedAssertion' and/or 'papiAssertion' and can be defined User- and Resource-specific.

Therefore, not all attributes are transmitted to all Resources anymore. Filters at the PoA side can now be configured to reject or allow a request based on the received attributes, contrary to only reject a request, as was the case in version 1.1. Furthermore, a GPoA can be directed to only disclose to its child PoAs a subset of the User data sent to it by the AS: the specific data to be sent may be configured in a per-PoA basis.

The last beta release of PAPI includes the option that the AS issues lists of Resources without instant attribute exchange with all (G)PoAs. The attribute exchange will only take place for the Resource the User wants to access. This further improves scalability and eliminates a potential data protection issue. The PAPI and SPOCP development teams have also successfully integrated PAPI and the authorization engine SPOCP [SPOCP], thus enhancing attribute-based authorization at the (G)PoAs with a much richer semantics. A project proposal for integrating PAPI and the PERMIS privilege management infrastructure [PERMIS] (based on X.509 attribute certificates) has been submitted to TERENA, seeking for support from the European NRENs.

### 2.5.1 System Architecture

System components as shown in Figure 9:

- Authentication Server (AS): The User has to provide credentials to the AS which are in turn verified by the organization's authentication method (e.g. LDAP, POP, x.509-certs, etc.) [1]. Once successfully authenticated, the AS will consult a local site database and create a web page, listing all web Resources available to the User. Cryptographically signed authorization information is also coded into those URLs [2].
- User's web browser: While building up the page received from the AS, the User's browser will contact all Points of Access of those web Resources and thus provide them with authorization information [3].

---

<sup>8</sup> <http://www.rediris.es/app/papi/index.en.html>

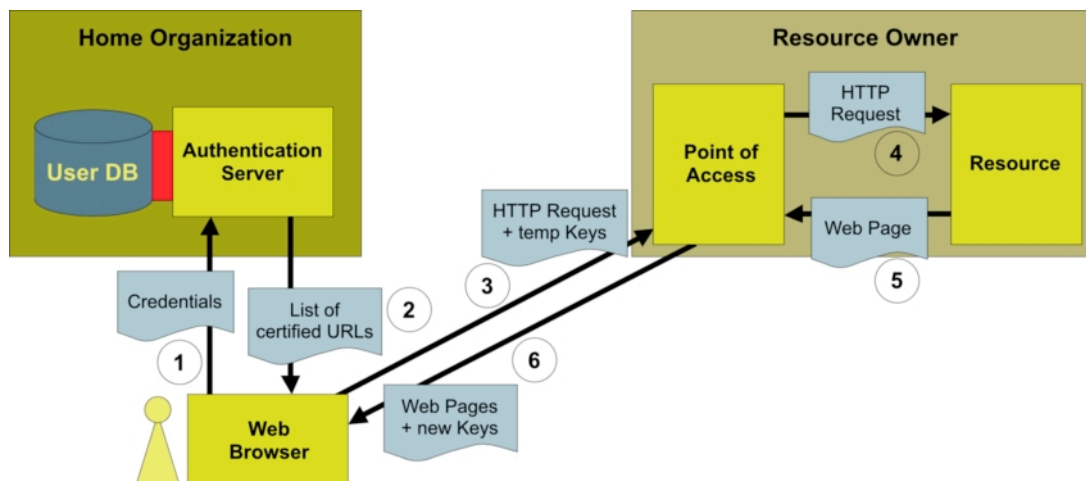


Figure 9: Point of Access to providers of information (PAPI) system

- Point of Access (PoA): This component performs actual access control to a set of protected web Resources. It verifies the authorization information it receives from the User's browser and updates cookies in the User's browser to keep track of authorized Users [6]. The PoA acts as a dedicated web gateway as shown above and accesses the Resource on behalf of the User's browser [4][5]. Alternatively, the PoA can be integrated with the Resource as an access control module of the Resource's web server.
- Protected web Resource: It trusts its PoA(s) to perform access control on its behalf.
- Version 1.1 of PAPI introduced the concept of a GPoA (Group wide PoA, not shown in Figure 9). A GPoA acts as gateway to a set of PoAs with identical access policy. The User gets access to all Resources, if the browser initially exchanged keys with the GPoA in charge of those Resources. This is an important feature to increase scalability of the PAPI model. Resource hosting organizations are advised to define a short list of different access policies and group all their Resources behind a low number of GPoAs, one for each access policy.

Example of PAPI usage:

A User, affiliated with Home Organization, wants to access a web-based Resource located at some remote site.

- 1) The User authenticates him-/herself to the AS of the User's Home Organization with whatever method the Home Organization uses for that purpose. The User gets a list of available Resources.
- 2) While the page is being built up, the browser contacts all PoAs of all Resources with attribute information embedded in the URL. Each Point of Access evaluates the received attributes and checks them against a local access policy. Graphical elements in the User's browser window will inform about the outcome of that authorization check for each entry in the list of available Resources. The PoA will at the same time send cookies to the User's browser allowing the User to access the Resource without renewed authentication.
- 3) The User can now access all listed Resources without any further authentication and will only be redirected back to the Authentication Server after expiry of the cookies.

## 2.5.2 Evaluation

Pros:

- No requirements to end user software and hardware, besides a web browser with cookie support.
- Supports common authentication methods and can be extended.
- Compatible with almost any web-based service. Existing web-based services need not be changed and can be put behind a Point of Access. Only authenticated and authorized Users will then be able to access this Resource.
- PAPI is operational in Spain to control access to libraries and other university Resources. It is also being used by the GLAM project<sup>9</sup>, an initiative of the University of London Library, funded by JISC.
- Software is freely available.
- Very responsive development team.

<sup>9</sup> <http://www.glam.lon.ac.uk/>

Cons:

- The Authentication Server of the User's Home Organization has to present an exhaustive list of services available to the User. It therefore has to know them all. All available Resources need to be registered at the Home Organization.
- All GPoAs and PoAs known to a certain PAPI AS are accessed after authentication to perform initial authorization checks. It is rather likely that the list of known Points of Access is larger than the subset the User is interested in. This is both a potential scalability problem as well as a privacy problem.
- Small development team, although its number has been doubled during last year.

Evaluation:

PAPI promises to provide the functionality required, but concerns over its scalability remain. The scalability issues are being dealt with in upcoming versions, and it is recommended to gain operational experience as soon as new versions of the software become available.

## 2.6 Shibboleth

Shibboleth<sup>10</sup> is a joint project of Internet2/MACE (Middleware Architecture Committee for Education)<sup>11</sup> and IBM. It aims to develop an architecture for standard-based vendor-independent web access control infrastructure that can operate across institutional boundaries.

The focus of Shibboleth is on supporting inter-institutional authentication and authorization for access to web-based applications. The intent is to build upon existing heterogeneous security systems in use on campuses today, rather than mandating particular schemes like Kerberos or PKI based on X.509. Project Shibboleth will produce an architectural analysis of the issues involved in providing such inter-institutional services, given current campus realities; it will also produce a pilot implementation to demonstrate the concepts.

In a February 2001, the Shibboleth working group documented *Shibboleth Overview and Requirements*<sup>12</sup>, according to which the system was designed.

*Current status of Shibboleth:*

Version 0.8 was released beginning of March 2003, version 1.0 was released in June 2003.

### 2.6.1 Architecture / System Design

The primary design principles for Shibboleth are:

- No single central piece of infrastructure required, scalable.
- Data protection and privacy are of importance for Shibboleth.
- The User is guided by 'HTTP redirect' from the Resource to the Authentication Server and back to the Resource for the authorization

A detailed description of the Shibboleth architecture can be found in *Shibboleth-Architecture Draft v0.5*<sup>13</sup>.

Shibboleth uses a federated administration, a Resource Owner leaves the administration of User identities and attributes to the User's Home Organization, which is also responsible for providing attributes about a User (possibly but not necessarily including a username) that the Resource Owner can use in making an access control decision when the User attempts to use a Resource. Users are registered only at their Home Organizations, and not at each Resource.

Shibboleth is a system for securely transferring User attributes from the User's Home Organization to the site of the Resource Owner, provided the Resources are accessible via standard web browsers. In addition, Shibboleth enables the Users to decide which information about them gets released to which site. The Users therefore have to balance access and privacy.

---

<sup>10</sup> <http://shibboleth.internet2.edu/>

<sup>11</sup> <http://middleware.internet2.edu/MACE/>

<sup>12</sup> Shibboleth Overview and Requirements, 20. February 2001, <http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html>

<sup>13</sup> Shibboleth-Architecture Draft v0.5, Marlena Erdos and Scott Cantor, 13. May 2002, <http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-v05.pdf>

The major components of Shibboleth are:

<i>WAYF</i>	Where Are You From Server Redirects the User back to the HS at his/her Home Organization. At least one WAYF Server is needed, but it may be replicated as desired.
<i>HS</i>	Handle Server Authenticates a local User according to the methods of the Home Organization and provides an opaque handle identifying the User.
<i>AA</i>	Attribute Authority Retrieves the attributes, which a User allows to be given to a Resource (according to the User's Attribute Release Policy) and passes them to the SHAR on behalf of the Resource.
<i>SHIRE</i>	Shibboleth Indexical Reference Establisher Makes sure that the Resource gets a 'pointer' (handle) back to the User without requiring more knowledge about the User. In case it is missing it refers the User via the WAYF Server back to his/her HS to get one.
<i>SHAR</i>	Shibboleth Attribute Requester Contacts the AA to fetch the available attributes describing the User and passes them on to RM.
<i>RM</i>	Resource Manager Decides on access to the Resource based on the information received and where necessary the information about earlier sessions of the same User.

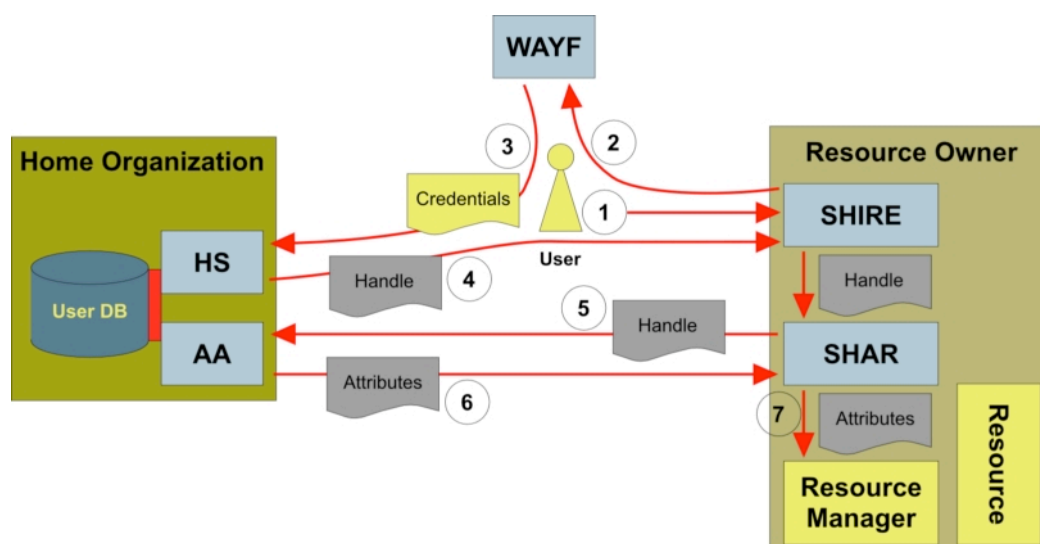


Figure 10: Shibboleth interactions

Example of Shibboleth usage:

The User U, affiliated to the Home Organization O, wants to access the web-based Resource R located at some remote site.

- U connects with his or her web browser to the web site R (1). The server R does not detect the required authorization information and redirects U (2) to the 'Where Are You From' web server W. The URL of R gets passed along.
- On W, U selects his/her Home Organization O from a list of organizations participating in Shibboleth. W redirects U (3) to the web server HS (Handle Service) located at the Home Organization O. The URL of R gets passed along again.
- HS authenticates U according the local rules and methods. Once authenticated, HS generates an opaque handle H for the User U. H is the authentication info U needs to present to R. U gets redirected to R (4). R sends handle H together with the URL of R (5) to the Attribute Authority (AA) located at the Home Organization O. AA checks which Attribute Release Policy (ARP) of User U applies to Resource R. AA returns the attributes it is

allowed to send to R (6).

Within R the attributes retrieved get passed to the Resource Manager RM (7) that decides on providing access.

- U gains access to the Resource

## 2.6.2 Evaluation

Pros:

- User privacy was from the beginning part of the architectural design. The User controls which information gets released to which Resource.
- Federative administration: User's Home Organization is responsible for the authentication, the User is responsible for selecting data to be released, and the Resource Owner is responsible for the authorization.
- Software will be available as open source.
- Supported by Internet2 – good chance for broad adoption.
- Scalable architecture. No single central infrastructure.

Cons:

- The architecture is designed for web-based Resources only. To make it easier for the User, it depends on the HTTP redirection feature.
- Few deployment experiences yet.

Evaluation:

Shibboleth promises to provide the functionality required; therefore, we should consider gaining operational experience as soon as the software becomes available. Effort will have to be put into the integration of the components AA, HS and RM into the local infrastructure at the participating organizations.

### 3 Comparison of AAI Components

This section introduces a set of aspects to be considered when analyzing AAI components or whole systems, and applies them to those elements presented in the previous sections that are suitable to be employed in the deployment of a general-purpose distributed AAI.

#### 3.1 Aspects

##### 3.1.1 Architecture

- *Centralized or distributed?*  
Is the architecture largely distributed, or does it rely on a few centralized components?
- *Inter-institutional?*  
Must be able to support authentication across institutional boundaries, using globally unique User identities.
- *Scalability*  
The architecture must be able to handle a large and growing number of Users (hundreds of thousands), resources (thousands), and authentication/authorization services (hundreds).
- *Use of existing solutions/methods?*  
Does the system rely on existing (partial) solutions for any of its individual components? For example, are the methods used for authentication widely known and implemented elsewhere?
- *Intended purpose*  
Who is expected to use or benefit from the system, and what problems does it aim to solve?
- *Complexity*  
What is the level of complexity in the individual system components, as well as the system as a whole?
- *Survivability*  
What are the most vulnerable points of failure in the system, and what are the consequences of a failure in any given component?
- *Component interaction*  
How much information must the individual component have of other components in the system, and how is this information distributed?  
For example, how much does an authentication service need to know about each relevant resource?
- *Task sharing*  
Who handles authentication? Who handles authorization?

##### 3.1.2 Implementation

- *Availability*  
Is there an implementation available today with sufficient functionality to be of practical use?
- *Distribution license*  
Open source freely available?
- *Real-life experience*  
Has the system been shown to work as intended in a realistic setting? What is the operational experience so far?
- *Implementation requirements*  
Does the system impose new requirements on existing systems/partial solutions? For example, are modifications to existing client software necessary?
- *Protocols supported*  
Support for protocols other than HTTP? Are HTTP redirects or cookies required?
- *Future support*  
Is future support and development planned and likely to happen?

##### 3.1.3 Security

- *Security levels*  
Is it possible to have several levels of security?
- *Authentication methods*  
Does it include support for different methods of authentication/different types of User credentials?
- *SSO supported?*  
Is a single sign-on solution possible?
- *PKI required?*  
Between which system components, and who manages the root CA?
- *Encryption*  
Where are encrypted tunnels required, and what methods are used?
- *Vulnerabilities*  
What kinds of attacks on the system are possible, where is the system most vulnerable, and what are the consequences in each case?

### 3.1.4 Privacy

- *User data ownership*  
Are User attributes stored centrally or at each User's Home Organization?
- *Distribution of User data*  
Who authorizes the use and distribution of User attributes to any given resource or resource group, and how?
- *Anonymization*  
How visible is the User (and the User's attributes) throughout the system? It must be possible to hide a User's globally unique identifiers from any given resource.

## 3.2 Reviewed Systems

### 3.2.1 Architecture

- *Centralized or distributed?*

A-Select	Distributed.
FEIDE	Centralized
PAPI	Distributed. The User's Home Organization handles authentication (through its AS), the resource owner handles authorization (through its handling of the (G)PoAs).
Shibboleth	Distributed.

- *Scalability*

A-Select	Promising. Simple authorization made possible by globally defined security levels. Some concern about how cross-organizational authentication will scale; no A-Select Server index described in [A-Select]. This feature is planned for implementation soon. Scalability for cross A-Select is already built-in, it requires only manual configuration to import another A-Select Server's identity.
FEIDE	Scales to millions of persons, hundreds of thousands of Users, ten thousand applications. Some concern about the centralized components (Moria for web login and LIMS for LDAP indexing)
PAPI	Potentially high number of interactions between client and PoAs. Some efforts underway to overcome this problem (GPOAs). No global index of Home Organization Authentication Servers.
Shibboleth	Promises to scale well (using an indexing server solution), but some concern about handling of User's approval of data sharing.

- *Use of existing solutions/methods?*

A-Select	Highly flexible. LDA/SQL/flat-file for User database back-end. Can integrate existing authentication schemes. Some concern about the need to modify existing applications to use the A-Select Agent but when using one of the available filters for Apache or IIS modification is reduced to a minimum
FEIDE	Flexible. LDAP for User backend, combined with SQL database for User management. Relies on extended eduPerson and eduOrg schemas.
PAPI	Highly flexible. LDAP/SQL for User database back-end. Existing web resources can be protected behind a PoA without modification.
Shibboleth	Highly flexible. LDAP/MySQL for User database back-end, SAML [SAML] for User attribute exchange. Can integrate existing authentication schemes. Relies on system-wide agreement to use eduPerson to describe the User.

- *Intended purpose*

A-Select	"...offers a uniform User authentication service to applications" [A-Select].
FEIDE	Federated Identity for education in Norway.
PAPI	"...a system for providing access control to restricted web-based information resources across the Internet" [PAPI]. Designed for and used by Spanish libraries/universities.
Shibboleth	"...designed to exchange attributes across realms for the primary purpose of authorization" [Shibboleth]. Does not handle "policy issues" such as automatic log-off.

- *Complexity*

A-Select	Moderate. Clearly defined system components/interfaces, but requires a relatively high degree of inter-component communication (only during authentication).
FEIDE	Moderate for web services. Manageable for institutions participating in the User identity solution.
PAPI	High, mostly due to how authorization is handled. Benefits from a highly modular design and implementation.
Shibboleth	Manageable. System components/interfaces are clearly defined.

- *Survivability*

A-Select	Good. No system-wide single point-of-failure. Includes a fallback mechanism in case of missing Authentication Service Providers.
FEIDE	Moderate. Some vulnerabilities to Moria and LIMS downtime, these components need to be replicated.
PAPI	Moderate to poor. Some vulnerability to GPoA downtime. Also some concern about the need to contact each PoA after authentication.
Shibboleth	Moderate. Some vulnerability to WAYF Server downtime.

- *Component interaction*

A-Select	Each A-Select Server must know about every other A-Select Server for cross-organizational A-Select; no central server index, yet.
FEIDE	Centralized through one point.
PAPI	The User's Home Organization needs to know about each (G)PoA the User can access.
Shibboleth	Each authenticator must possibly know about every available resource. Future efforts to group similar resources in "clubs" for easier handling.

- *Task sharing*

A-Select	User's Home Organization handles authentication. Resource Owner sets required security and authorization levels.
FEIDE	User's Home Organization handles authentication, for web this is proxied by Moria. Authorization is done by the resource owner, possibly on basis of information provided through Moria from the User's Home Organization.
PAPI	The User's Home Organization handles authentication (through its AS), the resource owner handles authorization (through its handling of the (G)PoAs).
Shibboleth	User's Home Organization handles authentication. The resource owner authorizes.

### 3.2.2 Implementation

- *Languages*

A-Select	Java. Web server filters in C.
FEIDE	Java.
PAPI	Perl.
Shibboleth	Origin (Home Organization) side Java, target (Resource) side C++.

- *Availability*

A-Select	Latest production release available for download.
FEIDE	Beta since April 2003. Production since June 2003.
PAPI	Latest production release available for download.
Shibboleth	0.8 available since March 2003. 1.0 planned for mid-June 2003.

- *Distribution license*

A-Select	Free for non-profit organizations worldwide.
FEIDE	Open source.
PAPI	Open source.

Shibboleth	Open source. Note a potential patent claim on SAML [RSA].
------------	---

- *Real-life experience*

A-Select	Pilots are being run at several organizations since September 2002.
FEIDE	Pilots since early 2003, beta since April 14 2003. Production June 16 2003
PAPI	Currently in use in Spanish libraries and universities.
Shibboleth	Version 0.8 out since March 2003. Test usage at Internet2 and internationally.

- *Implementation requirements*

A-Select	Requires an OS that runs Java SDK 1.4. Client browser must support redirects, SSLv3 or higher, and (non-persistent) cookies.
FEIDE	Client browser must support redirects. Server must speak SOAP or have an OS that runs Java SDK 1.4.
PAPI	Client browser must support redirects and cookies.
Shibboleth	Requires Apache v1.3.x, on the Home Organization side also Tomcat. Clock synchronization required; NTP on all servers. Client browser must support redirects and SSL, preferably JavaScript.

- *Support for protocols other than HTTP?*

A-Select	Not currently, relies on HTTP redirects. Support for non-web based environments is planned.
FEIDE	Not for Moria. May use LDAP directly to AT.
PAPI	No. Relies on HTTP redirects.
Shibboleth	No. Relies on HTTP redirects.

- *Future support*

A-Select	Backed-up by SURF and the SURF community. Future development likely.
FEIDE	Backed-up by UNINETT and the UNINETT community. Future development likely.
PAPI	Already in use; future development likely.
Shibboleth	Developed by Internet2/MACE. Future development likely.

### 3.2.3 Security

- *Security levels*

A-Select	Authentication method left up to the User's Home Organization. Explicit support for multiple security/authentication levels, with SSO possible.
FEIDE	Authentication method left up to the User's Home Organization, support for multiple types of credentials. Does not implement SSO as such
PAPI	Authentication method left up to the User's Home Organization. SSO possible.
Shibboleth	Authentication method left up to the User's Home Organization. Does not implement SSO as such, but may incorporate an existing SSO system.

- *PKI required?*

A-Select	Certificates required for system components.
FEIDE	System component certificates required. Usage of user-level certificates possible.
PAPI	Certificates required for system components (AS and PoA).
Shibboleth	System component certificates required. Usage of user-level certificates possible.

- *Encryption*

A-Select	SSL highly recommended to use.
FEIDE	Mandatory use of SSL for system component. SSL/TLS highly recommended for client to web server.
PAPI	TLS/SSL optional.
Shibboleth	SSL highly recommended to use.

- *Vulnerabilities*

A-Select	Open to attack during redirects. SSL optional.
FEIDE	Open to attack during redirects. Open to DOS attack at central components.
PAPI	Open to attack during redirects. TLS/SSL optional. Complete list of accessible resources sent to client after authentication; client contacts each resource.
Shibboleth	Open to attack during redirects. SSL optional.

### 3.2.4 Privacy

- *Ownership of User attributes*

A-Select	Only the User's Home Organization manages User data.
FEIDE	Only the User's Home Organization manages User data.
PAPI	Only the User's Home Organization manages User data.
Shibboleth	User's Home Organization manages User data; the User decides which resource gets access to which attributes.

- *Anonymization*

A-Select	User's identity is not transmitted during authentication. No User information leaves the User's Home Organization. Pseudo-identities are used.
FEIDE	User's identity hidden for target service, unless the target has explicit permission to access information about the User's identity.
PAPI	Encrypted User identity code is sent between AS and client.
Shibboleth	User's identity hidden internally behind a "handle". Only User's Home Organization can map a handle to a User.

## 4 Interoperability

The experience along the lifetime of TF-AACE has shown that many different solutions for AA infrastructures have been developed, almost one for each national environment, while still other solutions are being proposed and/or experimented with. The reasons for this high number of different initiatives include:

- Particular requirements at a national (or even more local) level: legal, cultural, User base...
- A well established (if not fully standardized) set of basic technologies.
- Availability of open source solutions, total or partially implementing these basic technologies.

So the current trend in the AA infrastructures arena is the development and deployment of local solutions, tuned to satisfy the particular requirements of the environment where they are being deployed.

Interoperability is required not only to enable trans-national collaboration, but also to integrate the different approaches followed so far to provide AA capabilities within particular applications or services, specifically to enable the inclusion of those communities whose nature implies them (like the case of Grids). The establishment of AA as a basic middle-ware infrastructure, rather than a set of (more or less) well-suited applications for a certain purpose, requires the definition of a framework to ensure interoperability. This interoperability framework must include:

- A definition of architectural concepts.
- A (extensible) set of User attributes, with well-defined syntax and semantics.
- Common protocols.
- Possibly APIs

Although interoperability requirements are not part of the objectives of this TF-AACE deliverable, some others (like B.2 and B.3) will address architectural and protocol issues. However, we think it is worth reporting the common understanding within the TF-AACE community (especially from those dedicated to develop AA software), which was reached recently.

It is not a matter of this pursued interoperability how Users are authenticated at their respective local sites (the acceptance of the corresponding procedures being subject to methods such as bi-lateral agreements or audit trails), but to consider what attributes are to be exchanged among sites, and the format for this communication. Five different ways to exchange information between AA components in different systems may be considered:

- Local - Proprietary:
  - 1) Native APIs or protocols.
  - 2) Underlying server/environment infrastructures: Apache notes/variables, Java environment, etc.
- External - Standard:
  - 3) SAML queries/responses.
  - 4) LDAP storage and access to trusted objects.
  - 5) Standard API, such as the GAA API.

With respect to SAML, it is worth noting that it provides only a framework and a lot of things related to interoperability are left to developers. Real usage and examples are still sparse and there are open issues, like those dealing with usability: how the attributes should be defined, in order to be understood, for example, by a Web server administrator. The common agreement of the developers in the TF-AACE community is to try to avoid "dialectal derive" of SAML, using OpenSAML as a reference.

A proposal, which gathered consensus and will constitute one of the central outcomes of deliverable B3, was the definition of a set of SAML schemas and rules accepted by any implementation claiming interoperability. This will be verified for at least three AA implementations (A-Select, PAPI and SPOCP).

The storage and access of trusted objects by means of LDAP is an appealing solution that could avoid part of the interoperability learning curve (since protocols to store and retrieve data are already well established), does not require a secure channel for data exchange, and implies a potential reduction of site configuration complexity. A proposal for a project implementing this approach has been presented for consideration of TERENA and the NREN community.

## 5 References

- [A-Select] A-Select V1.0 - Functional and Technical Design  
<http://a-select.surfnet.nl/>
- [FEIDE] Federated Identity for Education  
<http://www.feide.no/>
- [Olsen] Cato Olsen, UNINETT: "An evaluation of Shibboleth, PAPI and A-Select"  
<http://www.terena.nl/tech/task-forces/tf-aace/workshop/presentations/TF-AACECato.ppt>
- [OpenSAML] Open Source Security Assertion Markup Language Implementation  
<http://www.opensaml.org/>
- [PAPI] PAPI  
<http://www.rediris.es/app/papi/index.en.html>
- [PERMIS] PrivilEge and Role Management Infrastructure Standards Validation  
<http://www.permis.org/>
- [RSA] RSA Intellectual Property Rights Statement of Intent of January 20, 2003  
<http://www.oasis-open.org/committees/security/rsa-ipr-statement-SAML-OASIS-2003-01-20.shtml>
- [SAML] SAML - Security Assertion Markup Language  
[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security)
- [Shibboleth] Shibboleth  
<http://shibboleth.internet2.edu/>
- [SPOCP] Simple Policy Control Protocol  
<http://www.spopc.org/>
- [SWITCH] SWITCH Authentication and Authorization Infrastructure, Preparatory Study, May 2003  
<http://www.switch.ch/aa1/report.html>

## Appendix A Definitions and Abbreviations

<i>AAI</i>	Authentication and Authorization Infrastructure
<i>Access control manager</i>	Gatekeeper functionality of the resource which grants or denies access to the resource based on the access control definition and the Authorization Attributes retrieved
<i>Authentication</i>	Process of proving the identity of a previously registered User
<i>Authentication system</i>	System which can authenticate a previously registered User
<i>Authorization</i>	Process of granting or denying access rights for a resource to an authenticated User
<i>Authorization Attributes</i>	User data needed for access control decisions
<i>FEIDE</i>	Federated Identity for Education
<i>FEIDHE</i>	PKI Project of the Finish higher education
<i>GSI</i>	Grid Security Infrastructure
<i>Home Organization</i>	Representative of a User community, e.g. university, library, university hospital etc.
<i>NREN</i>	National Research and Education Network
<i>PAPI</i>	AAI Implementation from Spain (Point of Access to Providers of Information)
<i>Personal security environment (PSE)</i>	E.g. smart cards, passwords, certificates
<i>PKI</i>	Public Key Infrastructure
<i>PoA</i>	Point of Access (see PAPI 2.5)
<i>Registration</i>	Process of becoming an official member of a User community. During the registration, a person has to prove his/her identity
<i>Resource</i>	Application, web site, network, system, etc.
<i>Resource Owner</i>	Entity owning a resource and offering resource access to Users
<i>SAML</i>	Security Assertion Markup Language
<i>Shibboleth</i>	Joint project of Internet2/MACE (Middleware Architecture Committee for Education) and formerly IBM
<i>TF-AACE</i>	TERENA Middleware Task Force for Authentication and Authorization Coordination for Europe
<i>User</i>	Registered member of a Home Organization
<i>User-DB</i>	Database storing information about a registered User, maintained by the Home Organization