FAME: Adding Multi-Level Authentication to Shibboleth

Aleksandra Nenadic, Ning Zhang, Jay Chin, Carole Goble
School of Computer Science, University of Manchester
{anenadic, nzhang, jchin, carole}@cs.man.ac.uk

Abstract

The paper describes the design of FAME (Flexible Access Middleware Extension) architecture aimed at providing multi-level user authentication service for Shibboleth, which is endorsed by the Joint Information Systems Committee (JISC) as the next generation authentication and authorisation infrastructure for the UK Higher Education community. FAME derives authentication assurance level based upon the strength of the authentication token and protocol used by the user when authenticating and feeds it to the PERMIS authorisation decision engine via Shibboleth to enable more fine-grained access control. In this way, access to resources is controlled according to the strength of the authentication method so that more sensitive resources may require users to identify themselves using a higher level of authentication.

1. Introduction

Authentication methods and tokens provide varying levels of authentication assurance. Similarly, data and resources may have varying levels of sensitivity - library catalogue is less security-sensitive than highly confidential medical patients’ records. To provide a fine-grained access control to such resources, there is a need to link access privileges to the authentication assurance level of the method or token used to identify a user. For instance, an IP address-based authorisation service would grant the access privilege to anybody who can access a machine with a correct IP address. Authentication via username and password establishes the identity of a user through proving the knowledge of username and password. Smart-card authentication identifies a user who has a cryptographic key embedded within a physical token (smart-card) and can also demonstrate the knowledge of a secret (a password or a PIN) or prove the possession of biometrics (such as a fingerprint) that can be used to unlock the smart-card token. Clearly, the IP-based authentication method provides the weakest, whereas the smart-card based method provides the strongest level of authentication assurance. This fact should be taken into consideration when controlling access to data with different levels of sensitivity and the FAME-PERMIS project is aimed at realising this vision.

FAME provides support for Single-Sign-On (SSO), a facility which enables a user to authenticate once and repeatedly gain access to resources on multiple sites without re-authentication for the duration of the SSO session. It is designed for the easy integration with authentication services utilising a Web-based front-end, such as Kerberos, LDAP-, MySQL-based services, etc., or custom built enterprise authentication systems, and supports the use of various authentication tokens including username and password pairs, soft and hard certificate tokens, IP addresses, etc. Based upon the token used in an authentication instance and the cryptographic strength of the authentication protocol, FAME derives one of the four defined Levels of Assurance (called LoA hereafter), specified by the NIST Electronic Authentication Guideline (NIST Special Publication 800-63, [1]). The derived LoA is subsequently passed via Shibboleth infrastructure [5] to the authorisation decision engine provided by PERMIS, together with the user’s other attributes.

PERMIS [3] is an existing X.509 role-based Privilege Management Infrastructure, extended to include LoA in its decision making process. So, instead of using (Subject, Target, Action), PERMIS authorisation decisions are now made based on (Subject, Target, Action, LoA), where Subject is the user accessing the resource, Target is what the Subject is trying to access, Action is the action the Subject is allowed to perform on the Target, while LoA specifies the minimal level of authentication assurance required to access the specified Target.

Shibboleth is an open framework for passing user’s attributes from an institution that has performed user authentication (Identity Provider or IdP) to an institution that is providing resources (Service Provider or SP) the user is trying to access. The IdPs and SPs jointly form a trust federation and Shibboleth enables
such organizations to exchange information about the users in a secure and privacy-preserving manner. Shibboleth itself does not provide authentication nor authorisation services; it leaves the task and the means of identifying a user to the user’s home institution (IdP) and the task of authorising the user’s access to a resource by the resource provider (SP). As IdPs and SPs belong to the same federation and have predefined trust relationships, an SP trusts an IdP to authenticate its users properly and assert correct attributes about them. Shibboleth uses SAML (Security Assertions Markup Language, [2]) for the secure assertion of the user’s attributes between IdPs and SPs. The SP makes authorisation decisions for its resources based on the attributes asserted by the IdP. These attributes are generally used to refer to the characteristics of a user and his affiliation, e.g. employee@IBM, student@kent.ac.uk, or “currently enrolled in physics”@manchester.ac.uk, and need not contain user’s real identity (which is only revealed to the user’s IdP during authentication). In this way, Shibboleth preserves the user’s identity privacy.

FAME and PERMIS components are to be integrated within the Shibboleth infrastructure. FAME plays the role of an IdP’s local authentication system - it authenticates the users and derives their LoAs based on the strength of the selected authentication method/token used. The LoA value is subsequently passed to the SP via Shibboleth infrastructure as an user attribute. The SP feeds this attribute into its authorisation decision engine provided by PERMIS in order to reach a decision whether to grant or deny the user the access to a particular resource.

The rest of the paper is dedicated to the FAME part of the project. Details about PERMIS can be found elsewhere in the literature (cf. [3]). The paper describes the Shibboleth infrastructure, the design of FAME, and how FAME and PERMIS can be linked via Shibboleth. In detail, Section 2 describes the Shibboleth architecture. Section 3 explains how FAME and PERMIS fit inside the Shibboleth infrastructure. Section 4 gives details of the FAME design, architecture and components, and outlines FAME’s implementation. Section 6 discusses future challenges, and Section 7 draws conclusions.

2. The Shibboleth Framework

This section gives the overview of the Shibboleth [4] infrastructure, its architecture and components.

2.1 Shibboleth Overview

Shibboleth is an infrastructure for enabling inter-institutional sharing of Web-accessible resources which are subject to authentication and access control [5]. It defines two types of architectural entities: IdPs, sometimes also referred to as origin sites, and SPs, also referred to as target sites. The origin and target sites jointly form a federation, whose members are assumed to trust each other and have agreed to operate under certain rules. Inside a federation, a target site trusts an origin site to identify its users and provide correct attributes about the users. Based on the attributes that are asserted by the origin, the target makes a decision as whether the user is should be granted the access to the requested resource.

A user’s home institution is responsible for identifying and authenticating the user, and providing attribute assertions (encoded in SAML) about the user to a target site from which the user has requested resources. Shibboleth does not mandate how user authentication should be performed at the origin – the only requirement is that the origin has a Web-based authentication mechanism with preferably a SSO facility.

Authorisation is performed at the target and is based on the attributes asserted by the origin. The access control mechanism matches the obtained attributes against policy rules associated with the requested resource in order to determine whether the requester is permitted to access it. Similarly, Shibboleth does not mandate which authorisation engine should be used by the target. In other words, the authentication and authentication decisions and their implementation are left to and managed locally at the origin and the target, respectively. The only requirement is that the origin and the target agree and understand the meaning of the attributes and implement the Shibboleth-defined protocols for their request and release. In this way, Shibboleth detaches the management of users at cooperating institutions.

Shibboleth has an embedded mechanism to protect users’ identity privacy, i.e. to prevent the revelation of the user’s identity to the SP. To achieve this, attributes released from an IdP to an SP via Shibboleth need not contain information about the user’s identity. From the SP’s point of view, most of the time for it will suffice for the IdP to release an assertion that the user belongs to a certain group, i.e. that the user is a “student” or a “member of staff”. In this way, the user’s identity is withheld. Furthermore, the user has the option of specifying exactly which of their attributes should be released to a specific SP, allowing user greater control over its private data. On the other hand, SPs have a mechanism to ask IdPs for specific attributes, so they can explicitly ask for the user’s identity, should they require it.
2.2 Shibboleth Architecture

Shibboleth architectural components are illustrated in Figure 1. On the IdP’s side, Shibboleth has four components: the Attribute Authority (AA), the Handle Service (HS), an attribute repository (such as LDAP directory service [6]) containing user attributes and a Local Authentication System (LAS). The HS is a Web-based service responsible for determining the identity of the user by interacting with the LAS and creating an opaque handle (a reference) for the user. The handle will be used as a reference for the Shibboleth Attribute Requester (SHAR) from the SP’s site to later fetch the user’s attributes from the AA. The AA maps the user’s reference number from the handle with the user’s real identity and fetches the user’s attributes from the local attribute repository. Shibboleth does not specify how the HS knows the identity of the user, but it is assumed that the user is logged to the LAS prior to being allowed to hit the HS. The AA manages the users’ attributes at the origin site and releases them to SHAR when requested.

On the SP’s side, Shibboleth has three components: the Shibboleth Indexical Reference Estimator (SHIRE), the Shibboleth Attribute Requester (SHAR), and the Resource Manager (RM). SHIRE monitors the users’ requests for the resource on the target: if the user has not been authenticated yet by the IdP (as determined by the absence of a handle), the user is asked to go back to his IdP and do so. Once the user is authenticated and returns back to the SP with the handle obtained from his IdP, SHAR uses the handle to ask for the user’s attributes from the IdP. Once user’s attributes are supplied by the AA, the RM makes access control decisions by interacting with some sort of an authorisation engine.

In addition, there is a WAYF (Where Are You From) service in the infrastructure. This service is used by an SP to find out the address of an IdP for a particular user. The WAYF service may determine the address of an IdP by allowing the user to select his/her origin institution from a list of institutions that have been registered with the federation. The WAYF service may be provided by a third party or directly by the SP.

The following detailed steps explain the process when a user tries to access a protected resource at an SP where his institution has a subscription.

1. Using a Web browser, the user navigates to the protected resource that requires proof that he is eligible to access it.
2. The Shibboleth components on the SP detects that the user has not been authenticated yet (due to the absence of a handle in her request), so it redirects the browser to the WAYF service.
3. The WAYF service displays a page from which the user selects his home institution (i.e. his IdP).
4. The browser returns the selection to the WAYF.
5. The WAYF redirects the user to the IdP’s HS.
6. The LAS authenticates the user with whichever method is endorsed by the institution.
7. The user enters his credentials.
8. Provided the entered credentials are correct, the HS generates a digitally signed handle on behalf of the user. A handle is basically a session identifier. The handle is sent to the SHIRE component of the SP, which passes it to SHAR.
9. The SHAR sends the attribute request with the handle back to the AA of the IdP.
10. The AA verifies the handle, translates it back to the user’s identity and checks which attributes it may release to the asking SP. Then it sends the digitally signed attributes to the SHAR, packed in a SAML message.
11. Finally, the SHAR passes the received attributes to the RM, which decides whether Alice should be granted the access.

Steps (7)-(10) are completely invisible to the user, as it is server-to-server communication. Alice is only aware of steps (1)-(6) and (11).
3. FAME-Shibboleth-PERMIS Integration

As Shibboleth does not specify all components of its architecture, institutions have the freedom to implement them as they find suit. FAME and PERMIS respectively implement two of these components, namely the IdP’s Local Authentication System and the SP’s access control engine that interacts with the Resource Manager.

3.1 Shibboleth-PERMIS Integration

PERMIS is an X.509 Privilege Management Infrastructure implementation that employs the Role Based Access Control paradigm. Details on how PERMIS operates are outside the scope of this paper; more information about PERMIS can be found in [3, 7]. PERMIS is integrated within Shibboleth infrastructure to provide the SPs with a policy-based access control decision engine to enable controlled use of their resources. Attributes obtained from the IdP’s AA are fed into PERMIS to make access-control decisions. One of the attributes is LoA, which is derived by the FAME component with respect to the authentication method and token used by the user in the identification process.

PERMIS has developed an Apache module called mod_permis [8] for the use with target sites. It takes the control of the user’s request for the resource on the target after user attributes have been obtained from the AA, and interacts with the PERMIS access control engine in order to make an access control decision for the request.

3.2 FAME-Shibboleth Integration

FAME is a part of the IdP and plays the role of the LAS – it offers users a variety of authentication methods supported by the IdP, calculates the LoA based upon the selected method, and adds LoA to the set of other user’s attributes ready to be fetched by Shibboleth SPs. The rest of the paper explains the design of FAME.

4. The FAME Component

The architecture of the Shibboleth IdP is shown in Figure 2. The HS and AA are Java servlets running inside a Tomcat server assembled together with the Apache Web server. The HS creates the Shibboleth session, while the AA deals with attribute requests from SPs. Several Apache modules are used in order to carry out a Shibboleth session: mod_ssl protects the authenticity, confidentiality and integrity of the connection and checks the certificate presented by the SP. Requests for HS are accepted at the standard SSL port 443, while requests for the AA work on SSL port 8443. Any Apache AuthN module can be used to protect access to the HS, and this is where FAME component fits. Mod_jk Apache module is used to translate and pass requests from Apache to Tomcat, such that requests for the HS and AA are passed to the respective servlets. When dealing with attribute requests, the AA pulls the attributes from the LDAP server based on the IdP’s Attribute Release Policy (ARP). Only those attributes are permitted by the ARP are released back to the SP.

FAME is an Apache-Perl module with a purpose of guarding access to the Shibboleth’s HS: its role is to ensure that only authenticated users can pass to the HS. The functionalities of FAME are summarised here:

1. Facilitate controlled access to the HS;
2. Allow a user to choose among a list of authentication methods provided by the IdP;
3. Invoke the appropriate Authentication Server based upon the user’s choice;
4. Calculate the LoA based on the cryptographic strength of the selected authentication method;
5. Add the user’s current LoA to his other attributes to be retrieved by the Shibboleth SP later on;
6. Provide the Single Sign-On (SSO) functionality.

4.1 Deriving LoA

One of the main tasks of FAME is to assign a LoA attribute to users based on the authentication method they selected, and pass this LoA further to PERMIS via Shibboleth. NIST Electronic Authentication Guideline (NIST Special Publication 800-63) [1] identifies four levels of authentication and is used as a basis for the LoA values assigned by FAME. Level 1 provides the lowest level of assurance, while Level 4 guarantees the highest. These levels are defined in terms of the likely consequences of an authentication error - as the errors become more serious, the required level of assurance
increases. The following is a short summary of the technical requirements for each of the four levels.

**Level 1** – Successful authentication requires the claimant to prove that he/she controls the authentication token through a secure authentication protocol. Any of the authentication tokens of Level 2, 3, or 4 can be used, including PINs. Plaintext tokens (e.g. passwords) are never transmitted across the network. However, this level does not require the use of cryptographic protocols that can block off-line attacks. For example, password challenge-response protocols are allowed at this level, and an eavesdropper, having intercepted such a protocol exchange, can launch an off-line dictionary attack in order to work out the password. Common protocol that meets the Level 1 requirements is Kerberos [9].

**Level 2** - Successful authentication requires the claimant to prove that he/she controls the authentication token through a secure authentication protocol. Any of the authentication tokens of Level 3, or 4 can be used, as well as passwords (but not PINs). The use of cryptographic protocols that can prevent off-line, replay and on-line guessing attacks is required. To satisfy the Level 2 requirements, users can be authenticated, e.g., using passwords through encrypted TLS protocol session (so called “tunneled” password protocols).

**Level 3** - Successful authentication requires the claimant to prove the possession of the authentication token through a secure authentication protocol. Three kinds of tokens can be used at this level: cryptographic soft and hard tokens, and one-time password device tokens. Level 3 also requires two-factor authentication; the authentication token must be used in conjunction with a password or biometric to activate (unlock) the token. The use of cryptographic protocols that can prevent off-line, replay, on-line guessing, verifier impersonation and man-in-the-middle attacks is required. Example implementations include authentication with public key certificates via TLS, or tunneling the output of the one-time password device and a Level 1 personal password (used to unlock the device) through a TLS session.

**Level 4** – This level is intended to provide the highest practical remote network authentication assurance. Successful authentication requires the claimant to prove the possession of a cryptographic key through a secure authentication protocol. This level is similar to Level 3, except that only hard cryptographic tokens are allowed. These tokens are hardware devices that cannot be easily copied and have also to be unlocked with a password or a piece of biometric, and, thus, provide two-factor authentication. Either public or symmetric key technology may be used.

Table 1 shows relations between authentication tokens and their corresponding LoAs, while Table 2 does the same for various authentication protocols.

### Table 1. Authentication tokens vs. LoAs

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard token</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Soft token</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>One-time password device</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strong passwords</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passwords &amp; PINs</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

### Table 2. Authentication protocols vs. LoAs

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private key proof-of-possession</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Symmetric key proof-of-possession</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zero knowledge password protocol</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tunneled password protocol</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Challenge-response password protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.2 FAME Architecture

The FAME system architecture (shown Figure 3) consists of two components:

1. FAME SSO Checker (F-SSO), and
2. FAME Login Server (F-LS).

Figure 3. FAME architecture

F-SSO and F-LS jointly provide the SSO facility. The AuthN Servers are Web-based authentication services external to FAME. They can be any existing authentication services with Web-based interface, such as Kerberos, NIS, LDAP directory- or MySQL database-based authentication, SSL with PKI soft and hard tokens, etc. FAME does not re-implement these.
existing authentication services, but rather integrates itself with them and invokes them as necessary.

4.2.1 FAME SSO Checker. FAME SSO Checker (F-SSO) plays the role of a gate-keeper by controlling the access to the Shibboleth’s HS. If a user has not been authenticated yet in the current session (as determined by the absence of an \texttt{ss0} cookie in the user’s Web browser), he/she will be redirected to the FAME Login Server (F-LS) and forced to go through an authentication process. Otherwise, the access is granted and the current Shibboleth session continues without re-authentication. The address of the originally requested resource (i.e. the Shibboleth’s HS) is passed from the F-SSO to the F-LS in a \texttt{request-url} cookie. This is in order for the F-LS to know where to redirect the user upon successful authentication.

4.2.2 FAME Login Server. The role of the FAME Login Server (F-LS) is to receive users directed from the F-SSO and redirect them to AuthN Server of their choice. The number of AuthN Servers supported depends on the origin’s authentication policy and needs; the FAME interface allows a user to choose one from the list. For example, a user may normally authenticate using his smart card when accessing resources on a day-to-day basis from his office or lab. However, when the user is located in an environment that does not have a smart card reader present (e.g. in a Internet cafe), he/she may authenticate using username and password. The price the user pays for his mobility is lower LoA and restricted access to certain resources.

If authentication with the AuthN Server is not successful, it is up to the AuthN Server to display the error message to the user and the user will not be able to continue. Otherwise, the user is returned back to the F-LS and the AuthN Server also passes back the user’s identity to the F-LS. This ID varies with different authentication methods, e.g. for username-password based authentication ID is the user’s \texttt{username} with Kerberos ID is the principal’s name in the form of \texttt{<Fully Qualified Domain Name>@<Realm Name>}; for LDAP authentication ID is user’s \texttt{Distinguished Name (DN)} formed as a sequence of user’s \texttt{Relative Distinguished Names (RDNs)}; for certificate-based authentication ID is the \texttt{Distinguished Name extracted from the user’s certificate, i.e. a sequence of C=<Country>, ST=<State>, L=<Locality>, O=<Organisation>, OU=<Organisational Unit>, CN=<Common Name>; etc. A user can have multiple IDs as he/she can be authenticated using various authentication methods. All these IDs are mapped by FAME to the same entity as Shibboleth is not aware of the multi-level authentication.

Upon the return of the user to the F-LS from the AuthN Server, the user is issued with an \texttt{ss0} cookie by the F-LS and scoped for the use by the F-SSO. It contains the user ID received from the AuthN Server and the LoA derived by the F-LS. Subsequently, the user is redirected to the originally requested URL. This time, when the user tries to access the HS, the request will again be intercepted by the F-SSO. However, since the user’s request now contains the \texttt{ss0} cookie, the F-SSO will let the user through to the HS. In addition, the F-SSO will pass to the HS the user’s ID (obtained from the \texttt{ss0} cookie) through the environment variable \texttt{REMOTE_USER} in order to supply the HS with the identity of the browser user. This information is further used by the HS to produce a handle for the user.

4.2.3 AuthN Server. An AuthN Server is responsible for authenticating users. It receives an \texttt{auth-request} token from the F-LS, packaged as an URL parameter, encrypted with a symmetric key shared between the F-LS and the AuthN Server. This token contains a random challenge \texttt{RC} to which the AuthN Server has to produce a response in order to authenticate itself to the F-LS. The AuthN Server also has to pass the ID of the authenticated user back to the F-LS. These two pieces of information are packaged in an \texttt{auth-response} token, encrypted with the same shared and passed back to the F-LS via URL.

4.3 FAME Authentication Scenarios

In the following, we illustrate the working of the FAME system by describing the steps of authentication procedures. On diagrams shown in Figures 4 and 5, redirection between the components is represented using dashed lines.

\textit{Scenario 1: Access to the Shibboleth’s HS without an \texttt{ss0} cookie (initial sign-on).} A user initiates a new session to access the FAME-protected Shibboleth’s HS. In this case, no \texttt{ss0} cookie has been generated by the F-LS yet. In detail,

(1) A user makes a request for the url of the Shibboleth HS that uses FAME for authentication.

(2) The F-SSO intercepts the request and verifies that it is not associated with a current valid authenticated session for the requested resource. This is achieved by checking whether the request contains a valid \texttt{ss0} cookie that has been previously issued by the F-LS. Consequently, the F-SSO generates the \texttt{request-url} cookie that contains the address of the originally requested
resource (i.e. that of the HS) and redirects the user to the F-LS to perform the authentication.

(3) The F-LS determines a list of the available AuthN Servers (each with an associated LoA), and sends a form page with the list back to the user.

(4) The user is presented with the page containing the AuthN Servers and chooses his/her preferred method of authentication.

(5) The user sends his choice back to the F-LS by submitting the form.

(6) The F-LS redirects the user to the selected AuthN Server after generating the auth-control cookie (containing the random challenge RC and scoped for the F-LS itself) and auth-request token (containing the same random challenge RC and scoped for the AuthN Server) encrypted with the shared secret key.

(7) The user is presented with the login page by the AuthN Server where they enter their credentials. Note that some authentication methods, such as SSL client certificate authentication, do not require the user to enter any credentials. Authentication in these cases is performed without any interaction between the user and the AuthN Server, as the user’s authentication credentials are sent to the AuthN Server automatically by the user’s browser.

(8) The user’s credentials are sent by the browser to the AuthN Server. If the authentication is not successful, the user is prompted to re-enter their credentials until the authentication is completed successfully. Otherwise, the AuthN Server generates an auth-reply token containing the ID of the authenticated user and the random challenge incremented by one, i.e. RC+1, and redirects the user back to the F-LS.

(9) The F-LS verifies the auth-reply token by confirming that the random number contained inside it (RC+1) is equal to the random number from the auth-control cookie (RC) incremented by 1. This verification confirms the authenticity of the AuthN Server to the F-LS as the tokens exchanged between them (namely auth-request and auth-reply tokens) are encrypted with the secret key shared by these two components. If the verification is positive, the F-LS generates an SSO cookie scoped for the F-SSO. The F-LS then redirects the user to the originally requested url.

(10) The redirect will cause the browser to re-request the original resource. This time, when the request is intercepted by the F-SSO, it contains the SSO cookie issued by the F-LS.

(11) The F-SSO verifies the SSO cookie, and if the verification is successful the F-SSO lets the user through to the HS.

---

**Scenario 2: Access to the Shibboleth’s HS with an SSO cookie (Single Sign-On).** In this scenario, a user requests access to the HS and the user has already got a valid SSO cookie for this session.

(1) A user makes a request to the Shibboleth’s HS.

(2) The F-SSO detects a valid SSO cookie (which the user has obtained earlier in the session) and grants the user the access. If the SSO cookie has expired or is invalid, the user will be redirected to the F-LS for re-authentication as in the previous case.

All interactions between the user’s browser, the F-SSO, the F-LS, and the AuthN Server are SSL-protected to prevent cookies and other data from being disclosed to an eavesdropper that could use them to impersonate the user.

---

**4.5. Configuring FAME**

FAME is implemented as an Apache-Perl module Fame.pm in package called MyAp. In order to configure FAME to work with a Shibboleth IdP, the following steps need to be undertaken. It is assumed that Apache Web Server and Tomcat servlet container are made to interoperate and Shibboleth IdP components have been installed. The path to the FAME module
(MyAp::Fame) has to be configured in the Apache configuration files so that the Server can find and load the module.

Let us assume that the Shibboleth’s HS (which is a JAVA servlet served by Tomcat) is running at the address https://my-host/shibboleth/HS. To set up FAME to protect this location, an entry has to be added in Apache’s configuration file for my-host, similar to the following:

```
LoadModule MyAp::Fame

<Location /shibboleth/HS>
  AuthType Fame
  PerlResponseHandler MyAp::Fame::sso_checker ...
</Location>
```

The command `LoadModule` instructs Apache to load the FAME module and the `<Location>` block sets up the F-SSO component of FAME to handle authentication for requests for the URL /shibboleth/HS on my-host before they are allowed to proceed to HS.

To set up the F-LS, a snippet of code similar to the one below should be inserted in the Apache’s configuration file for my-host.

```
<Location /fls>
  PerlResponseHandler MyAp::Fame::login_server
</Location>
```

Although FAME is developed with integration with Shibboleth in mind, it is a module that can clearly be used completely independently from Shibboleth to protect any Web-based resource with its multiple authentication services.

### 6. Future Work

So far, both FAME and PERMIS components have independently been successfully integrated into the Shibboleth infrastructure (i.e. FAME-Shibboleth and Shibboleth-PERMIS integrations have been made to work). We are now working on making the integrated solution FAME-Shibboleth-PERMIS. In order to pass the LoA attribute derived by FAME to PERMIS, we have defined an LDAP attribute LoA (OID urn:oid:1.2.826.0.1.3344810.1.1.104) which will hold this value and set up the Shibboleth on the IdP’s side to pick up this attribute and automatically pass it to the Shibboleth components on the SP. Our immediate future work includes testing this integration.

FAME will also be integrated via Shibboleth with GridSite [10] in order to provide their SP, hosting HTTP(S) Fileservers, Websites and Web Services, with LoA values if its users, and it will be used without Shibboleth to provide multi-factor authentication for the Clinical e-Science Framework (CLEF) [11].

### 7. Conclusions

In this paper, our work on the design and implementation of a multi-level and multi-factor authentication framework FAME in the context of Shibboleth is presented. The work has demonstrated the feasibility of such an authentication framework and enabled us to understand the challenges involved in the use of various authentication methods for securing Web-based resources. Our future work involves continued integration of other authentication systems into FAME and inking authentication strengths to authorisation decisions and fine-grained access control provided by PERMIS. In addition, we will also be looking at taking into account other authentication methods, such as using dynamic parameters (location, time, proximity to sensors, etc.) to make access control decisions, into our FAME system design.

### 8. References