

17.11.08

Deliverable DJ5.3.4: Evaluation of a Prototype for unified SSO



Deliverable DJ5.3.4

Contractual Date: 30/09/08
Actual Date: 17/11/08
Contract Number: 511082
Instrument type: Integrated Infrastructure Initiative (I3)
Activity: JRA5
Work Item: WI3
Nature of Deliverable: R (Report)
Dissemination Level: PU (Public)
Lead Partner: USTUTT
Document Code: GN2-08-208v2

Authors: S. Neinert (USTUTT), D. Lopez (RedIRIS), M. S. Cuenca (UMU), Ó. C. Reverte (UMU), I. Thomson (DANTE)

Abstract

This Deliverable evaluates the development of a prototype for unified Single Sign On for eduroam and eduGAIN.

Table of Contents

0	Executive Summary	vii
1	Introduction	1
2	GÉANT2 uSSO Requirements	2
2.1	Security	2
2.2	Standards and Integration	3
2.3	Operational	3
2.4	Organisational	3
2.5	Authentication and Authorisation Mechanisms	3
3	DAMe uSSO Architecture	5
3.1	Network Authentication and Token Delivery	6
3.2	uSSO Client Middleware	8
3.3	uSSO Profiles	8
3.4	Token-based Authentication to Application-Level Services	9
4	Evaluation of the Architecture	11
4.1	Functional and Performance analysis	11
4.1.1	Testbed overview	11
4.1.2	Performance analysis	12
4.2	Security Considerations	15
4.3	Standards, Integration and Operational Aspects	16
5	Prototype Implementation and Deployment	18
5.1	Overview of the deployment	18
5.2	Software and hardware requirements	19
5.3	User	20
5.3.1	Xsupplicant	20
5.3.2	Token Manager	21
5.4	Remote Institution (RI)	22
5.4.1	RADIUS Server	22
5.4.2	Bridging Element (BE)	24

5.4.3	Policy Decision Point	25
5.4.4	Protected Service	25
5.5	Home Institution (HI)	25
5.5.1	Metadata publication	26
5.5.2	RADIUS Server	26
5.5.3	Identity Provider	27
5.5.4	Authentication Bridging Element	29
5.5.5	Attribute Bridging Element	30
5.6	Further steps	31
5.6.1	Use of FreeRADIUS 2.0	31
5.6.2	Use of XSupplicant 2.0	31
5.6.3	RadSec support	31
6	Conclusions	33
7	References	34
8	Acronyms	35

Table of Figures

Figure 3.1:	unified Single Sign On	5
Figure 3.2:	Network Authentication	7
Figure 3.3:	Token-based Authentication to Applications	9
Figure 4.1:	uSSO testbed	12
Figure 5.1:	Deployment of the testbed; authorised network access (left) and combined network and service access (right)	18
Figure 5.2:	User	20
Figure 5.3:	Remote Institution	22

Table of Tables

Table 4.1: Median of the values (in milliseconds)	13
Table 4.2: Medians of the network authz values (in milliseconds)	14
Table 4.3: Average of the values (in milliseconds)	15
Table 5.1: Software versions and download URLs	19

0 Executive Summary

This deliverable is based upon a previous deliverable, GN2-08-080 DJ5.3.2 “Architecture for Unified SSO” [DJ5.3.2], which describes the requirements, architecture and potential for a unified Single Sign On (uSSO) solution based on eduroam and eduGAIN. Such a solution uses the SSO state information from the lower network layer (eduroam) to bootstrap the SSO state for a higher web and application layer (eduGAIN). Additionally it allows fine-grained attribute-based access to networks and authorisation to services, including the eduroam service. This means that authorisation could be based on groups and roles, rather than individuals, so that users can obtain access to resources specific to their individual preferences and entitlements based on the attributes assigned to their sign on (please see [DJ5.3.2] for detailed examples of this type of authorisation).

The Single Sign On element of uSSO is particularly attractive to end users. This means that they need enter their details once only, and yet still be granted access to the resources they are eligible to use. For example, this could mean: wireless network access anywhere in Europe and beyond using eduroam, access to web based services (for example eLearning and library systems), or even access to Grid based services (such as high performance computing resources or a distributed medical database).

This deliverable:

- Summarises the requirements for uSSO (as defined in [DJ5.3.1]).
- Gives an overview on the uSSO architecture that was developed in the sub-project DAME (Deploying Authorization Mechanisms for Federated Services in the eduroam Architecture), which has been documented in detail in [DJ5.3.2].
- Compares the architecture against all the defined requirements to check for compliance,
- Provides a description of the complete deployment of the prototypical implementation in a federated testbed.
- Lists and describes the evaluation tests that have been carried out on the uSSO architecture and deployment, and gives the conclusions derived from these tests.
- Concludes with an assessment of the evaluation and a reference to future work that will be undertaken for further evaluation, development and deployment.

Note that no uSSO service is planned for GN2, although evaluation is ongoing for employing the uSSO architecture in other GN2 related projects.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

1 Introduction

eduroam gives roaming users wireless network access all across Europe, and beyond. Its basic principle is that authentication is carried out by the home institution and authorisation by the visited institution, thereby forming a federated Authentication and Authorisation Infrastructure (AAI).

Currently, authorisation is limited. eduroam Service Providers do not get much information about their users (usually only their name, sometimes not even that, and home domain. In contrast, Service Providers for web-based AAIs (for example Shibboleth or PAPI) can request user attributes as expressed in a schema as eduPerson or SCHAC (for example the user's home organisation, the home organisation's type and the user's affiliation to that organisation). Using such information the Service Provider can perform authorisation based on groups and roles, rather than individuals. Also, certain grid-based service infrastructures also make use of these fine-grained authorisation capabilities and define their own, sometimes application-specific, attributes.

Such fine-grained attribute-based authorisation can be beneficial for both the users and the Service Providers. Users can receive personalised access to extended services, possibly not offered to all "anonymous" users. The Service Providers can better protect their resources compared to letting every "anonymous" get access, but still can apply a much more scalable authorisation mechanism compared to individual access policies.

As described in GN2-08-080 DJ5.3.2 "Architecture for Unified SSO", such authorisation allows for the development of a unified Single Sign On (uSSO) solution. This strategy not only offers an improved user experience, but could also improve security as it would reduce the transmission of sensitive information.

This Deliverable evaluates the development of a prototype for uSSO based on eduroam and eduGAIN.

2 GÉANT2 uSSO Requirements

The most fundamental requirement of Single Sign On (SSO) is that the user must login once only. The credentials entered would then be authenticated and authorised for accessing multiple services and resources, provided by multiple Service Providers (SPs), without the need for repeated input of those credentials.

The term unified SSO means that the service “network access” is seen as one of these services. This leads to a number of further requirements and challenges, as although both the AAI for the network and the higher layer services share many common concepts, the specific implementations have been developed using different technologies.

Within GÉANT2 there are two federated AAI technologies, eduroam (for wireless and wired network access) and eduGAIN (for access to web and other resources, e.g. Grid resources). Both were established before the uSSO architecture was finalised. Therefore, eduroam is deployed and offered as a production service, and eduGAIN is deployed in an operational testbed. The uSSO solution has to make use of these available infrastructures and also must fit into this setup of diverse components. It is seen as a proof-of-concept development with no plans for transition to service in GN2.

The following requirements have been defined in [DJ5.3.1], which in turn is based on DJ5.2.1 “Documentation on AAI Requirements” and DJ5.1.2 “Documentation on GÉANT2 Roaming Requirements”. They are repeated here in a summarised form. Please see the above deliverables for more details. The generic uSSO scenario is also described in these deliverables.

2.1 Security

An infrastructure for controlling access to resources and services across the different security domains must take appropriate measures to achieve a reasonable level of security (which is a trade-off between maintaining a high security level while still maintaining acceptable usability). The preservation of data integrity is essential in the federation context. Furthermore the integrity of security-related data must be ensured, and related data like CRLs must be regularly refreshed and kept up to date.

As digital user identities and attributes are exchanged inside the uSSO architecture, privacy protection issues arise. Therefore, users must be able to get to know and control what information about them is exchanged, and to what entities.

Mutual authentication should be performed between end-user and authentication server, and the relevant processes within the infrastructure should be verifiable, using records of data related to service access when necessary.

2.2 Standards and Integration

As already mentioned, eduroam is an established service and eduGAIN is a pilot service. Therefore, the uSSO solution must be able to be **integrated** into these infrastructures, interfacing on one side with 802.1X and RADIUS and on the other side with SAML (Security Assertion Markup Language) over HTTP.

The uSSO infrastructure should be based on **open standards**, such as those used in eduroam and eduGAIN already. It should also be **neutral** with regards to the local AAI, as well as the type of application supported.

2.3 Operational

An important requirement for an infrastructure meant to build a European confederation is **scalability**; not only concerning the number of geographical regions but also concerning the number of services and users. The system must be sufficiently **robust** to cope with the expected load.

The uSSO system must be **easy to use**, not only for the end-user but also for the administrators of the services and identity providers (IdPs).

2.4 Organisational

As the envisaged federated authorisation infrastructure will be a federation of federations (a confederation), one organisation needs to be the **federation service provider** and connect it's federation to the confederation. **Federation agreements** shall define the purpose, technical and business terms of every federation. These agreements need to be available for all other participants of the confederation.

2.5 Authentication and Authorisation Mechanisms

According to the principle of federation, home institutions authenticate their users with the **authentication mechanism of their choice**, and in order to **protect its users' credentials** it shall be the only component that sees those credentials.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

A **harmonised user identity** should be created by agreeing to a common schema that describes a minimum set of supported attributes, including the syntax of the **user id format**.

Participating federations should be able to handle and support different **Levels of Assurance (LoA)**.

Except for the support of different LoAs (which is still being developed in most of the participating federations), all of these requirements are met in this uSSO solution.

3 DAME uSSO Architecture

The uSSO architecture defined within the DAME project, and fully described in [DJ5.3.2], extends the existing eduoam infrastructure to create a seamless link between the network-layer authentication mechanism and other authentication steps enforced by application-level service providers.

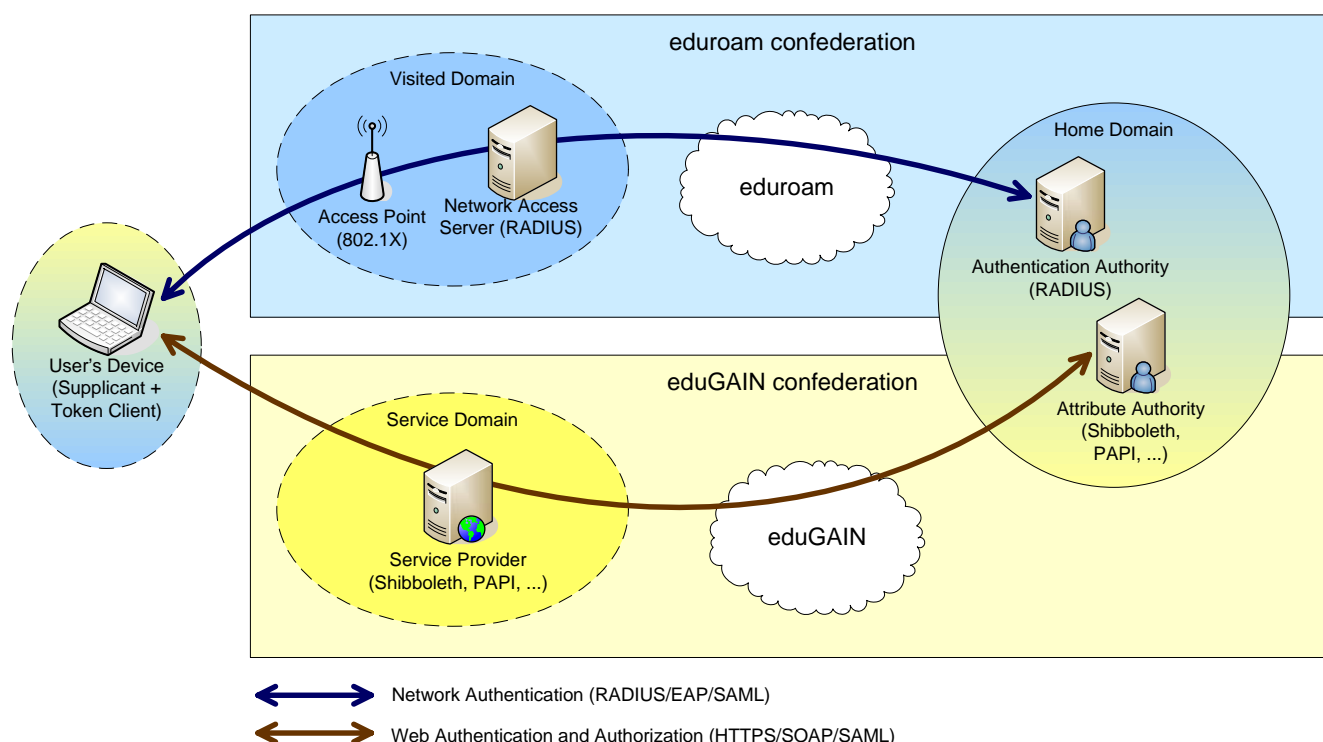


Figure 3.1: unified Single Sign On

Basically, this architecture involves the delivery of additional authentication information to end users (**authentication tokens**) during the network access process, and the definition of how an application-level service is able to validate the information contained in the token. The architecture follows the guidelines defined in [DJ5.3.1] and makes use of eduGAIN [eduGAIN] as the core AAI technology. This is shown in Figure 3.1. The authentication for eduoam (and the network) bootstraps the authentication for eduGAIN (and thereby higher-layer resources and services).

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

From a global point of view, users belong to their Home Institution (HI), which is responsible for performing the authentication process and for releasing the appropriate user attributes when required by other institutions. The Remote Institution (RI) is where the roaming user is trying to access the network. It has to determine the properties related to the network connection of the visiting users, according to the described attributes. Additionally, once users gain access to the network, they might want to access other high-level protected resources (for example the Web or Grid computing tools). Note: Visited Domain, Home Domain and/or Service Domain need not be in geographically different locations; for example could simply be different networks (virtual or “real”) on the same campus or even within the same building.

3.1 Network Authentication and Token Delivery

To provide one uSSO, authentication for all kinds of services is bootstrapped from the network authentication. This is done by delivering a token to the client during the network authentication phase. The token is generated by the IdP responsible for replying to additional attribute requests that might be received later. As we can see in Figure 3.2, the token (a SAML Authentication Statement following eduGAIN recommendations) is generated by the Bridging Element (BE) of the HI, based on the native authentication response provided by the corresponding IdP. Finally, the token is sent to the user through the Protected Extensible Authentication Protocol (PEAP) [PEAP]. Network authentication remains unaltered from the original eduroam, which guarantees a minimal impact during the deployment of the uSSO architecture.

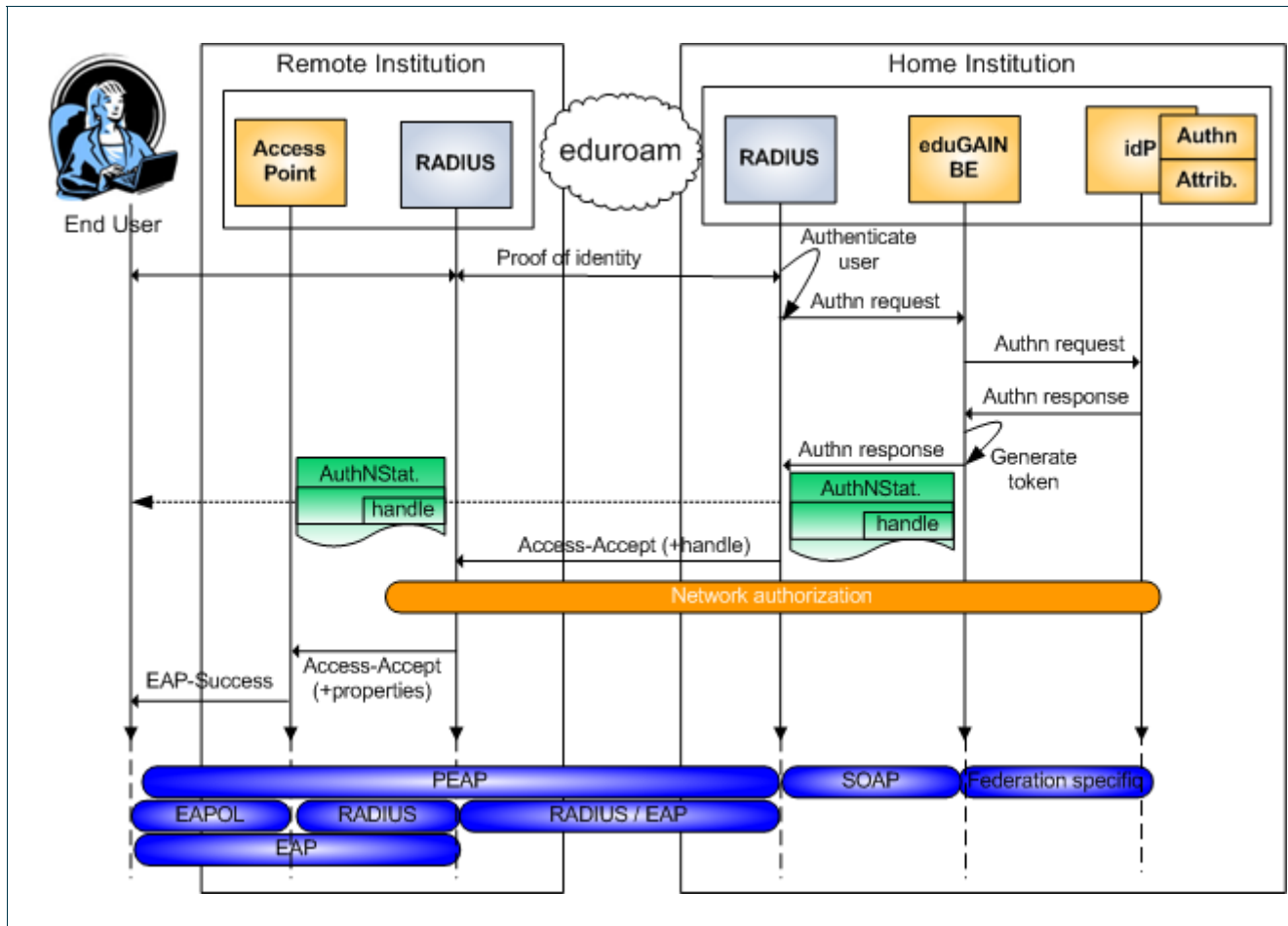


Figure 3.2: Network Authentication

The token indicates that this user has been authenticated by a trusted entity inside the federation. The information contained in this statement includes:

- Data about the subject.
- The entity performing the authentication.
- A validity period for the token.

The uSSO token is implemented by means of an eduGAIN **AuthNResponse**, and it is called **eduToken**. Several mappings have been defined in order to connect specific federation technologies to the uSSO architecture. These mappings requires conversions from their specific authentication statements to and from the **eduToken**, as described in [DJ5.3.2] for Shibboleth [Shib] and PAPI [PAPI].

3.2 uSSO Client Middleware

The uSSO architecture implementation uses specific middleware characteristics on the client. To begin with, a SSO-enabled supplicant is required for end users. This supplicant implements the 802.1X [802.1X] protocol for the connection to the wireless network. After a complete analysis of the different existing alternatives and the requirements imposed by the architecture, Xsupplicant [XSUPPLICANT] was selected. This software had to be modified to adapt it to some of the requirements of the uSSO architecture, as described in [DJ5.3.2].

Also, a dedicated client for managing eduTokens is required. This Token Manager is an application that is installed on end-user devices. Tokens, which are digitally signed, are always encrypted before storing them on the user's device. The primary purpose of the Token Manager is to link network authentication to Web authentication. It takes the eduToken from the supplicant and provides an interface for web components to request tokens when authentication is required.

3.3 uSSO Profiles

Together with the architecture for uSSO, several profiles (or instantiations) of this architecture have been defined to incorporate SSO mechanisms into specific access control scenarios. These profiles refer to different OSI levels; one profile for authorisation of network connections and their related properties, and another focused on web-level resources.

In terms of network authorisation, one of the targets for uSSO is the integration of authorisation decisions into the eduroam infrastructure. As previously discussed, the current eduroam authentication mechanism is preserved in order to minimise the impact on the institutions willing to continue using the standard process. Thus, a new authorisation infrastructure has been defined to extend the RADIUS servers with NAS-SAML [NAS-SAML] Policy Decision Points (PDPs). Once a roaming user is authenticated following the standard eduroam mechanism, the remote RADIUS server uses NAS-SAML to query the user's attributes and, once this information is recovered, an authorisation decision is obtained using the PDP. Also, a set of obligations can be returned, along with the authorisation decision, containing some specific network properties to be enforced by the access point. As is described in [DJ5.3.2], the remote RADIUS server can make use of the LDAP interface to connect with the eduGAIN Bridging Element, which acts as context handler and guides all the authorisation process. This LDAP interface is usually included in the most common implementations of RADIUS servers, which facilitates the deployment of this profile.

Regarding Web authentication, the profile was defined considering that Service Providers are not yet supporting the use of eduTokens, and therefore the remote Bridging Elements have to be modified. These elements are eduGAIN-aware and act as IdP to SPs. They can request user attributes for authorisation and they are extended to request an eduToken from the uSSO client and to validate that token.

3.4 Token-based Authentication to Application-Level Services

After a user has signed on to the network the eduToken is available in the Token Manager. Figure 3.3 shows how an eduToken is used for authenticating to an application-level service (which can be a simple web resource or a complex Grid Computing service). When requesting access to a service offered by an SP, the request is redirected to an eduGAIN remote Bridging Element (r-BE) that has been extended to be token-enabled. Instead of redirecting it further to the user's HI, an active component is sent back in the response to the user. This component (implemented as a signed Java Applet in the current prototype) fetches the eduToken from the Token Manager and POSTs it to the extended r-BE. The r-BE validates the token and, if successful, replies with an assertion to the SP and the user is granted access to the service.

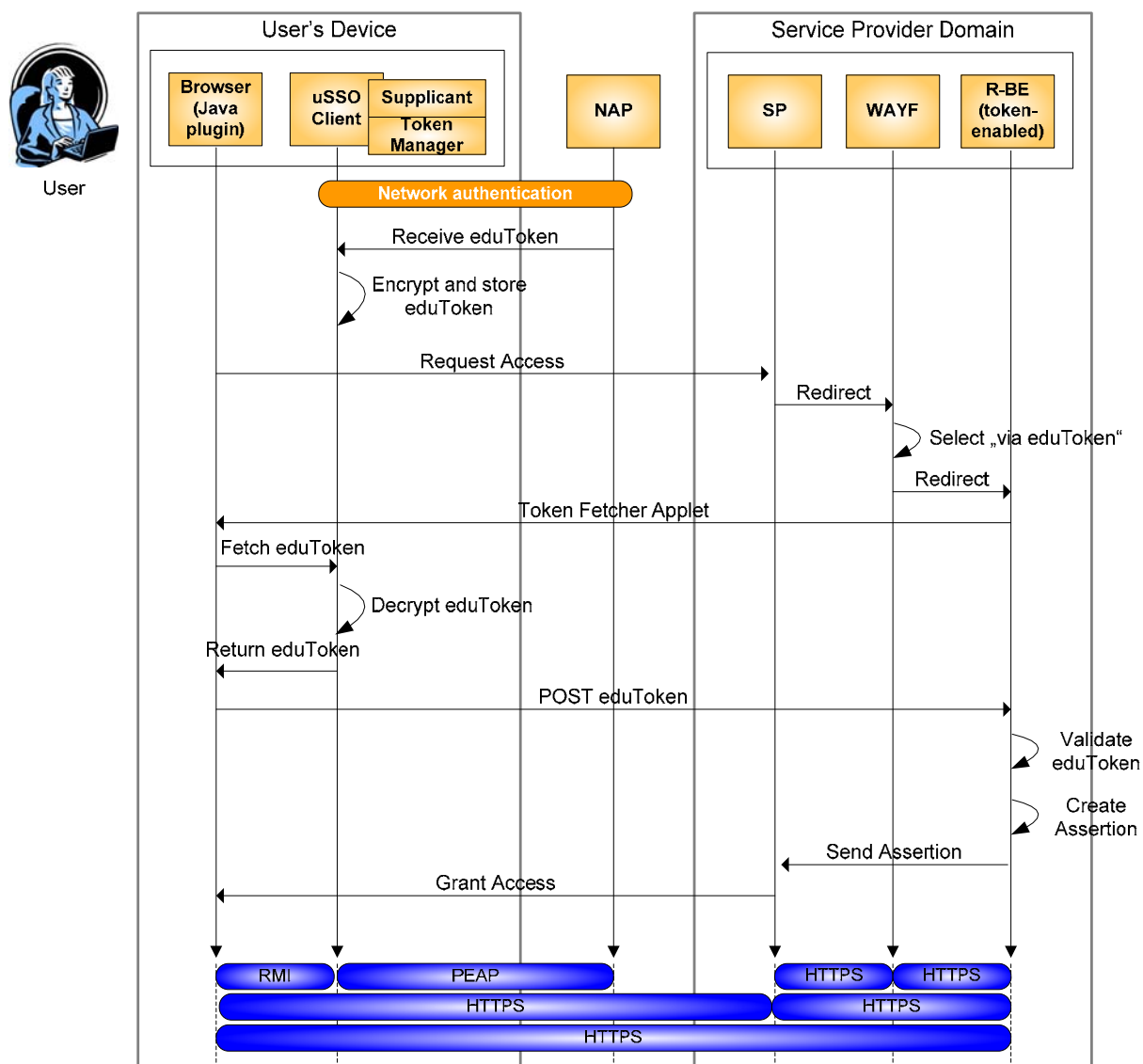


Figure 3.3: Token-based Authentication to Applications

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

If the service to be accessed is a more complex Grid service, the message flow is very similar to that for accessing web resources. A number of Grid architectures require a dedicated Grid client on the user's side, but many support a user-friendly access via a web browser. In that case a Grid portal provides the simplified interface to the complex Grid service(s) that are operated via servlets. Authentication to the Grid portal is done via the collocated SP using the procedure described above.

Authorisation for the Grid service(s) is supported by passing the user handle from the SP to the Grid portal and then to the service, which can request user attributes and delegate the authorisation to a Policy Decision Point according to the NAS-SAML profile as described in section 3.3.

4 Evaluation of the Architecture

This section describes the testbed used to carry out the functional and performance analysis of the architecture and profiles described in this document. This section is an overview of the more detailed analysis in DJ5.3.2 [DJ5.3.2].

4.1 Functional and Performance analysis

4.1.1 Testbed overview

The testbed deployed for performance analysis implements the architecture and profiles described in this document. The aim of this testbed is to check the different steps that compose the interaction between components and to get an overview of the efficiency of the proposed architecture. This includes from the initial network authentication based on the eduroam network, to the use of public key certificates and MDS components of the eduGAIN framework, going through the generation and delivery of the eduToken and the management of user attributes and authorisation policies.

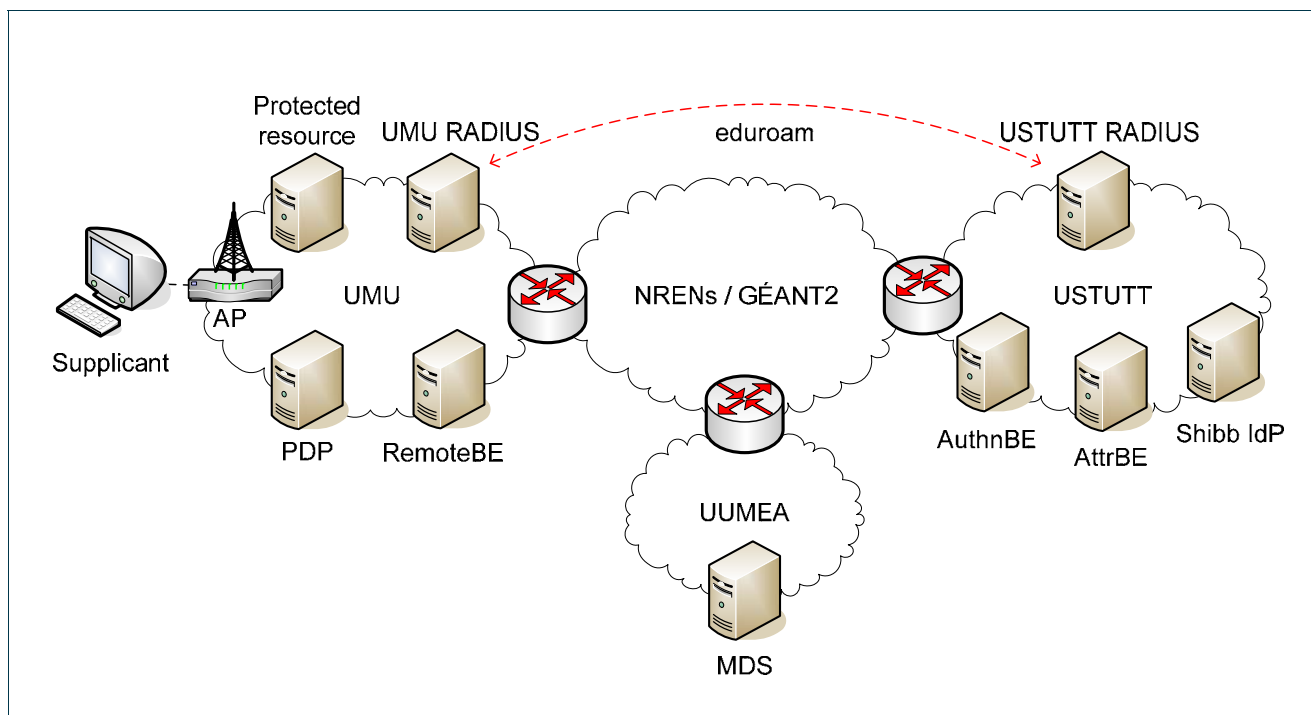


Figure 4.1: uSSO testbed

There is a direct translation between the architecture developed and the elements of the testbed, shown in Figure 4.1. The scenario for the testbed is made up by three institutions: University of Murcia (UMU), University of Stuttgart (USTUTT) and University of Umea (UUMEA). UMU acts as the RI where the roaming user is trying to access to the network. USTUTT acts as the HI the user belongs to. Finally, UUMEA is the institution that holds the Metadata Service (MDS) for the federation. Furthermore, the RADIUS servers from UMU and USTUTT are connected through the eduroam hierarchy. A detailed description of the testbed components can be found in [DJ5.3.2].

To deploy the testbed without modifying the current eduroam infrastructure, a new level of RADIUS servers has been added to the hierarchy and they are located under the institutional RADIUS in UMU and USTUTT. Therefore, we must take into account that the time measurements obtained for eduroam include an additional level of RADIUS servers.

4.1.2 Performance analysis

4.1.2.1 Initial authentication and uSSO token delivery

To demonstrate that the architecture developed is feasible, several tests were performed over the testbed described above. In these tests, time measurements were taken to control the time that each part of the process consumes. Then, the time results obtained in the uSSO testbed were compared with the time that eduroam takes to authenticate a user. The specific measurements recorded in the testbed are:

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

- **Eduroam:** The time that the authentication process takes in the standard eduroam. The new level added in the eduroam hierarchy must be taken into account.
- **Extended authentication:** The time from when the user runs the client to connect to the network until the connection is available. Apart from these final results, the influence of the different phases during the extended authentication has also been analysed:
 - **Shibboleth authentication.** This is the time that the IdP needs to validate the user and to generate the authentication assertion needed to build the uSSO token. It is measured in the AuthnBE.
 - **Token delivery.** This is the time that the basic eduroam authentication process is incremented due to the delivery of the uSSO token.

Tests were executed 105 times sequentially. The time results [DJ5.3.2] show how approximately 95% of the values for each time measurement are uniform. Only 5% of the results are different, and could be due to an unnoticed problem in the network or in the computer where the entity is running.

Because the majority of the values are uniform, a median average is best suited as a representative measure. The medians of all the time measures taken are summarised in the following table. These values will be used as the representative value of each time measurement.

eduroam	Extended authentication	Shibboleth authentication	Token delivery
1568	2817	500	736

Table 4.1: Median of the values (in milliseconds)

These figures indicate that the token delivery and the Shibboleth authentication represent an increase of around 78% over the basic eduroam authentication process. However, this step is carried out only once at the start of the process. Taking into account the benefit derived from the SSO mechanisms, this seems a reasonable time for the user to wait for the network connection.

4.1.2.2 Profile for network authorisation

This profile allows the remote institution to enforce specific network properties for the user that has authenticated using the previously described architecture. The increase in the time that the user has to wait until the connection is available due to the network authorisation phase is studied next.

The time measurements taken in the testbed cover from the query times to obtain BEs location (MDS), attribute requests and authorisation decisions query. Based on these measured values, the following times were calculated:

- **Network authorisation:** The time necessary to obtain the user's attributes and, based on them, determine the proper network properties. This is calculated as the sum of the *MDS query* time, the *Attribute request* time and the *PDP query* time (*MDS query + Attribute request + PDP query*).
- **eduGAIN request:** The time that the eduGAIN protocol needs to transport the attribute request and attribute response between the remote and home BEs. This is calculated as *Attribute request* time minus the *Shibboleth attribute request* time (*Attribute request - Shibboleth attribute request*).

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

- **Authn & authz:** The total time needed to establish the connection for a user in a remote institution that is also enforcing specific network properties for the user. This time includes the *Extended authentication* time and the *Network authorisation* time (*Extended authentication* + *Network authorisation*).

The medians of these values, calculated after the execution of the test 105 times sequentially, are summarised in the following table. The time to gain access to the network goes up to 3376 ms. This time represents an increase of 18.9% over the extended authentication, and 112.6% over the basic eduroam authentication. However, this process needs to be carried out only once, and therefore less than 3.5 seconds is not a long time for the user to wait to get initial network access.

MDS query	eduGAIN request	Shibboleth attr. request	Attribute request	PDP query	Network authorisation	Authn & authz
388	87	31	110	30	531	3376

Table 4.2: Medians of the network authz values (in milliseconds)

4.1.2.3 Token-based Web Authentication

This workflow starts with visiting a protected website, using the eduToken for authentication and finishing with granting the user access to that website. It is composed of automated and manual sections, the latter requiring user interaction.

Two phases were measured according to their performance:

- **Client Side:** The Token Fetcher Applet retrieves the eduToken and returns it (RMI call + read from file + decrypt + encode the eduToken, no marshalling).
- **Server side:** The Token Servlet processes the eduToken and creates a Shibboleth assertion for the SP (decode, unmarshall, validate, create Shibboleth Response).

As a normal browser was used as a client to perform realistic tests, they were started manually and each was executed ten times as follows:

- **Client Side:** To ensure true measurements, before each test the browser cache was emptied, all cookies were deleted, the Java VM cache was emptied and the browser was restarted. Therefore the results are a worst-case estimation only, as caches would strongly shorten the times. Times for redirections were not included in the measurements. Tests were performed using Firefox 2.0 and Java 1.5 on a Windows XP system.
- **Server Side:** The server hosting the Token Servlet (= the token-enabled r-BE) was restarted before each test execution, and the browser state was reset as described above. Therefore these measurements are also a worst-case estimation. Tests without restart resulted in noticeably shorter times, so it is possible that some time was consumed by one-time only operations for initialisation inside the used libraries. The test server used was an Apache Tomcat 5.5 with Java 1.5 on a Windows 2003 system.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

Client Side	Server Side	Server Side: Unmarshalling	Client & Server Side
906	6359	4896	7265

Table 4.3: Average of the values (in milliseconds)

An initial login using the eduToken would take approximately seven seconds. This shows that it would probably be useful to take further performance optimisation into account when developing the current prototype towards a production release version. But still this time is only required once, when signing on, which is one of the main benefits of the unified SSO architecture.

Summarizing the results of this performance analysis from initial authentication for the network until accessing a web resource shows a small additional overhead for establishing the context for uSSO, which was expected. Anyhow the results clearly show the feasibility of this approach for uSSO (at least performance wise), although it is still only a prototype implementation that could be optimized to further reduce this overhead. The advantage for the user by having to sign on only once definitely outbalances the slightly longer time for this one login process.

Finally it shall be noted that the Identity Providers (IdP) will in fact enable or not the unified SSO functionality for their users, as it is them who decide to connect their RADIUS to the IdP that can produce the uSSO token (see figure 4.3), thereby allowing for a more unified user experience of the federated AAI they are part of.

4.2 Security Considerations

For unified SSO the eduroam and eduGAIN architectures are extended at three places: the Home Institution (HI), the user, and at the Service Provider (SP).

At the HI the EAP-tunnel ends, the user credentials are validated and also the eduToken is generated by the IdP at the HI. Both HI RADIUS and IdP are supposed to be within the same security domain, so they are statically connected, and an encrypted connection via HTTPS is strongly advised. Therefore no additional threat is expected at the HI.

The eduToken is sent in an EAP TLV inside the encrypted tunnel from the HI RADIUS to the user's supplicant. The security level is as high as in usual eduroam with PEAP or EAP-TTLS for the transport of the eduToken.

The user's device can store eduTokens permanently, and these can be used to log in at any site supporting this uSSO approach. As with any SSO solution the loss or theft of credentials poses a greater risk as they are valid for a larger set of resources and services, and also in any SSO a "signed on" state is established on the user's device that enables him or anyone else using his device to sign on to the respective resources and services without entering any credentials. This higher risk is the price of the increased usability of the system.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

In the current prototype the tokens are stored only encrypted to the disk. They can be made to have a limited lifetime to increase security, and more options for deleting tokens (e.g. automatically when shutting down the token manager) and for prompting for credentials (e.g. periodically) could easily be implemented. General measures for system security and integrity are strongly advised, but out of scope of the uSSO architecture. A unified Single Sign Out would leverage the security further, although no design was finalized for this feature yet.

A dedicated token store outside the browser requires an active component that can be sent with an HTTP reply but still read from that token store. A number of alternatives are possible for this; in the current implementation a signed JavaApplet approach was chosen.

The third extension is at the SP: As in eduGAIN it does not rely on the authentication decision of an IdP directly, but refers to the remote Bridging Element (r-BE). The eduToken that the r-BE receives is not specific to the SP currently accessed, but the audience restriction is contained in the assertion that the r-BE produces and sends to the SP. This again is similar to the usual web profile of eduGAIN. It is up to the SP to decide if this security level (and usability level) is appropriate for its resources and services or not.

As a summary it can be stated the confidentiality is gained by the transport only over encrypted channels (PEAP and HTTPS), and mutual authentication that provides peer entity authentication. There is only one step with only unidirectional authentication; when the eduToken is retrieved, it cannot be detected if it comes from the same token store it was originally sent to. Data integrity of the eduToken is ensured by the HI that digitally signs it, so any modification can be detected.

Finally it should be noted that no analysis on the robustness of the system against denial of service (DoS) attacks exists at present. It can be expected that the vulnerability against DoS is slightly higher than with eduroam/eduGAIN simply because of the increased computation that need to be performed at the HI and the SP.

4.3 Integration and Operational Aspects

The uSSO architecture is designed by extending the eduroam and eduGAIN infrastructures and links both together. It mostly makes use of existing components of eduroam and eduGAIN, and only adds a few additional ones. A home institution that wants to provide uSSO to its users needs to have a FreeRADIUS installation with a modified PEAP module, which connects via a new Bridging Element (for Shibboleth or for PAPI) to the existing Identity Provider. Further types of AAI would need their specific BE “flavour”. A remote institution does not need to change any installations at all. A Service Provider that wants to offer uSSO and accept eduTokens needs to install the modified, token-enabled r-BE and provide the Token Fetcher. All required software as well as related technologies (802.1X, RADIUS, SAML) should already be known if the institution is participating in eduroam and in eduGAIN, therefore the additional burden on the administrators is expected to be rather low.

An important aspect is that the uSSO architecture requires specific client middleware, which has to be distributed, installed and configured correctly on the end-users’ devices. A client for eduroam is required in any case, but it additionally needs to support receiving eduTokens and providing them to the Token Manager. The Token Manager needs to be installed as well, but both clients could be offered in one package.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

The uSSO specific components can be deployed “on top” of existing eduroam and eduGAIN deployments, without disturbing either. Therefore each institution can decide individually if and when it wants to support a unified SSO and migrate at it’s own pace. In addition to the description in [DJ5.3.2], the deployed testbed recently was extended by installations of RadSec servers demonstrating the compatibility with this new eduroam profile.

From the end-user’s perspective the uSSO is designed to simplify their life, but they are expected be familiar already with the usage patterns of a federated login.

Performance issues are already analysed in section 4.1.2, and in the design of the uSSO architecture care was taken not to introduce any central components that could become a bottle-neck and thereby hamper scalability of the overall system.

The developed software fully implements the required functionality, but it was developed as a prototype for demonstrating the architecture. Improvements concerning documentation, logging, error handling and integration into existing middleware should be considered when planning to deploy this software for productive use. Extensive testing in a production-like testbed would be strongly advised, to gather further experiences on operating the uSSO related components.

5 Prototype Implementation and Deployment

5.1 Overview of the deployment

The testbed was implemented in three different institutions; one acts as the RI and the other two act as the HI for the user. The first provides access to the network and a protected service to users who belong to the federation, while the others are responsible for managing the user information based on PAPI and Shibboleth respectively. In this way, we can identify two different scenarios; one where the user accesses the network in the RI, shown in Figure 5.1(left) and one where the user accesses a protected service, shown in Figure 5.1 (right).

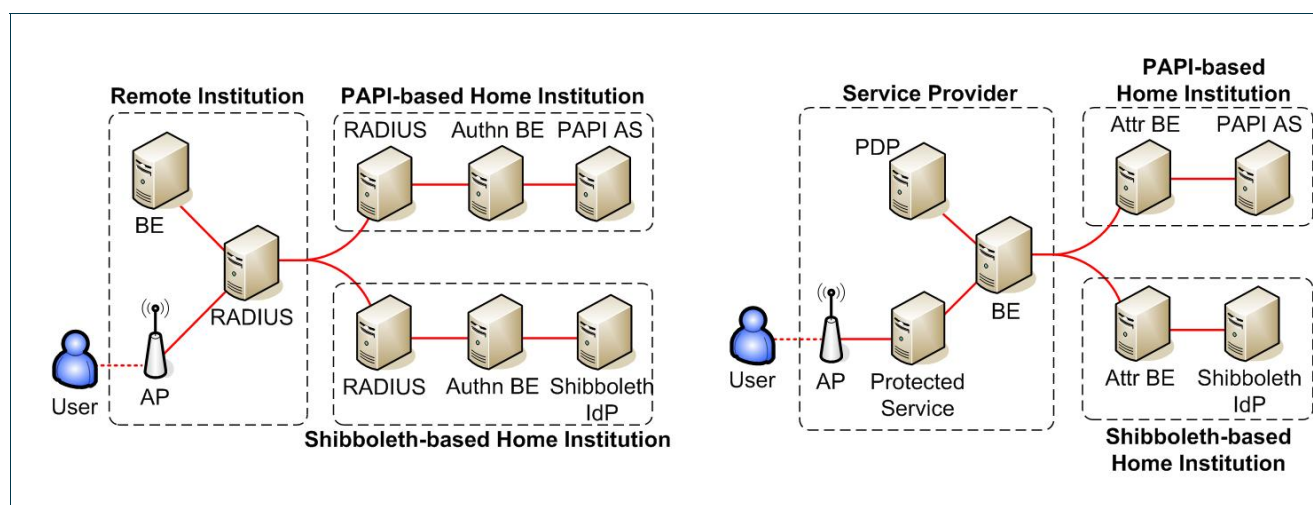


Figure 5.1: Deployment of the testbed; authorised network access (left) and combined network and service access (right)

The user belongs to any HI, based on PAPI or Shibboleth, and at any given moment moves to the RI. They first access the wireless network and later to a protected resource using the SSO Token obtained during the network access, or eduToken. The network access process is managed by means of the 802.1X client or Supplicant. On the other side, to manage the eduToken properly, the user makes use of the Token Manager.

The RI needs to deploy an access point (AP) to offer wireless network access to the user. The authentication process is carried out by the RADIUS server, which contacts the analogous entity in the HI. This institution also offers a Web Service granting access to users based on the validity of the eduToken.

Finally, since Home Institutions are responsible for managing the user attributes and providing authentication statements and attributes about their users, they need to make use of an IP. With this purpose, two different systems are used in the testbed: the Shibboleth Identity Provider and the PAPI Authentication Service. Independently of the specific system, the access to this entity is carried out via an eduGAIN BE. Specifically, the Attribute BE is used to recover the user attributes and the Authentication BE is used to obtain authentication statements about users. Moreover, a RADIUS server is needed to receive authentication requests about the users that belong to the institution and are accessing the network in the RI.

5.2 Software and hardware requirements

All the software described in this deliverable can be downloaded from their respective web pages. Table 5.1 shows the pieces of software used, the version and the web page where to download them. New or modified programs can be obtained from the FTP of the DAME project¹. Some other libraries and tools (wireless-tools, libiw-dev, libssl-dev, bison, flex, Make) are also necessary, but are usually included in current Linux distributions.

Name	Version	URL
Sun Java SE	1.5.0	http://java.sun.com/javase/downloads/index_jdk5.jsp
FreeRADIUS	1.1.4	http://freeradius.org/download.html
Apache Tomcat	5.5.23	http://tomcat.apache.org/download-55.cgi
Shibboleth IdP	1.3	http://shibboleth.internet2.edu/downloads/shibboleth/idp/1.3.3
Apache HTTP Server	2.0.55	http://httpd.apache.org/download.cgi
Apache HTTP Server	1.3.34	http://httpd.apache.org/download.cgi
PAPI AS	1.4.0	http://papi.rediris.es/software.html
Penrose Server	1.2.1	http://docs.safehaus.org/display/PENROSE/Download
libcURL	7.15	http://curl.haxx.se/download.html

Table 5.1: Software versions and download URLs

¹ <ftp://dame.inf.um.es>, contact contact.dame@dif.um.es for login information

From the user's perspective, it is necessary to have a PC with wireless capabilities. The Supplicant and the Token Manager must be installed on it. For the RI, it is necessary to have a standard access point able to enforce some network parameters (such as the session timeout or maximum bandwidth) and four computers, one for the Remote RADIUS server, one for the Protected Service, one for the Remote BE and the last one for the PDP. These four components can be installed on the same computer provided the standard ports used by each component are properly configured to avoid conflicting situations. In the HI, four computers are necessary: one for the specific identity provider, one for the Home RADIUS server and one for each of the BEs. Again, only one computer with the proper configuration can be used to install all the components.

5.3 User

As Figure 5.2 shows, the user needs to install the Supplicant to access to the wireless network and the Token Manager to store, protect, and send the eduToken received during the network authentication phase to the protected service. The Supplicant is a modified version of the XSupplicant application, which belongs to the Open1X project². The Token Manager is a java application developed internally in DAME. Both applications can be downloaded from the FTP of the project.



Figure 5.2: User

The sections that follow describe the process for installing the Xsupplicant and Token Manager, and provide sample scripts for use. Interested parties are encouraged to carry out these procedures themselves for evaluation purposes.

5.3.1 Xsupplicant

Download the package *xsupplicant-DAME-1.2.8.tgz* from the FTP and extract its contents. Then, compile and install the application:

```
# configure  
# make  
# make install
```

² <http://open1x.sourceforge.net>

An example configuration file, *xapplicant-example.conf*, is included in the package to help to configure this application. The main part of this file is the PEAP configuration section, where the appropriate CA certificate and user identifier are specified. Additionally, it is necessary to set the *token_path* field, which defines where to store the eduToken received during the authentication.

Finally, the command to execute this application is the following:

```
# xapplicant -i <iface> -c <conf-file> -d4 -f
```

Where *iface* is the network interface being used and *conf-file* is the path to the configuration file.

5.3.2 Token Manager

The Token Manager is a java application, so the Sun Java SE (JDK) 1.5.0 or higher must be installed. Moreover, it relies on the eduGAIN infrastructure to manage the eduToken. Therefore it is necessary to configure it appropriately. With this in mind, the package *eduGAIN-conf-client.tgz* is included in the FTP as an example. It should be decompressed in the */etc* directory of the system. As an alternative (e.g. when installing on Windows) the location of these files can be specified using the option:

```
-Dnet.geant.edugain.base.configuration = <PATH>/edugain_client.properties
```

Then, the file *edugain_client.properties* should be configured with the proper keystore and truststore information. Detailed information about how to configure eduGAIN can be found on its website³.

After the installation of the JDK and the configuration of eduGAIN, download the package *DAMeTokenManager.tgz* from the FTP server and extract its contents. This package contains all the libraries and java classes needed to run this application. It also includes a script to run the Token Manager in the *bin* directory. It is necessary to adjust some of the parameters of this script: the path to save the eduToken; the password that should be used to decrypt it.

```
# . run_tokenmanager.sh
```

The interaction between the Remote BE and the Token Manager is carried out by means of a Java applet and RMI. Thus, it is necessary to configure the security level in the Java Virtual Machine to allow it. This can be done by writing in the file *<java_path>/jre/lib/security/java.policy*, the following policy:

```
grant {  
    // Allow everything for now  
    permission java.security.AllPermission;  
};
```

³ <http://www.edugain.org>

5.4 Remote Institution (RI)

The RI offers two types of services:

- Access to the network by means of an eduroam-based infrastructure. In this case, an authorisation process is performed to determine some specific network parameters to enforce in the network connection, such as the session timeout or maximum bandwidth. Therefore, the RADIUS server needs to interact with the BE deployed in the institution.
- Access to a protected website via eduToken.

Figure 5.3 shows the entities that need to be deployed in the Remote Institution and the specific software elements that each entity requires.

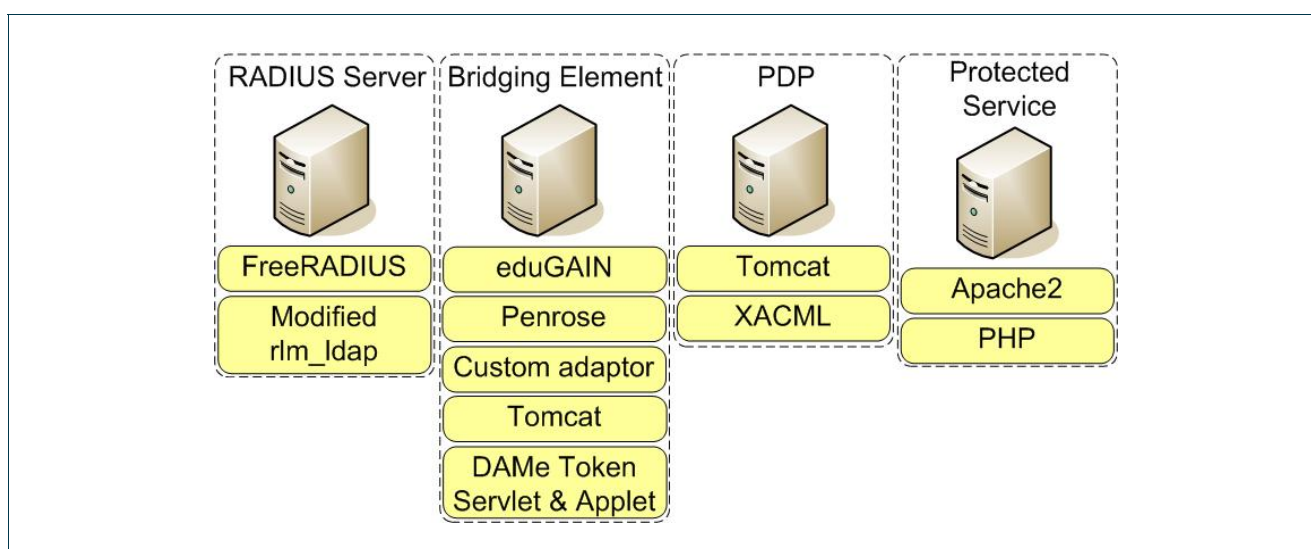


Figure 5.3. Remote Institution

The sections that follow describe the process for configuring the RI, and provide sample scripts for use. Interested parties are encouraged to carry out these procedures themselves for evaluation purposes.

5.4.1 RADIUS Server

This RADIUS server is the one that initially receives the authentication request for the user from the access point. Then, based on its realm, this request is forwarded to the appropriate Home RADIUS server. If the authentication response is *Success*, this server should contact the BE to get the network parameters for the user identified by the user's handle received from the Home RADIUS.

This server is based on FreeRADIUS, but before compiling the source code, it is necessary to replace the module `rlm_ldap` with the contents of the package `rlm_ldap.tgz`, which can be downloaded from the FTP of the

project. This modified module is necessary due to some restrictions in the original module to be used in the post-auth section of the RADIUS server. After this step, FreeRADIUS can be compiled and installed using the following commands:

```
# ./configure --with-openssl-includes=<path_to_openssl_includes>  
              --with-openssl-libraries=<path_to_openssl_libraries> --prefix=<path_to_freeradius_installation>  
# make  
# make install
```

Once the server is installed, the first configuration step is to define a new attribute in the RADIUS dictionary. This attribute will be included by the Home RADIUS server to specify a handle that is necessary to requests the user's attributes in the authorization process. The new attribute is defined by creating the file `<freeradius_directory>/share/freeradius/dictionary.dame` with the following contents:

```
VENDOR DAME 7777  
BEGIN-VENDOR DAME  
ATTRIBUTE DAME-Handle 1 string  
END-VENDOR DAME
```

Next, the new dictionary file must be added to the RADIUS general one. This is done adding the following line to the file `<freeradius_directory>/share/freeradius/dictionary`:

```
$INCLUDE dictionary.dame.
```

Regarding the interaction with the BE, to avoid introducing a new interface in the FreeRADIUS implementation, the communication between these two entities is done through an LDAP interface. To set up this communication, the LDAP section of the file `<freeradius_directory>/etc/raddb/radiusd.conf` is configured in the following way:

```
ldap {  
    server = "<ldap_server>"  
    identity = "<root_identity>"  
    password = <password>  
    basedn = "<base_DN>"  
    filter = "(uid=%{reply:DAME-Handle})"  
    start_tls = no  
    dictionary_mapping = ${raddbdir}/ldap.attrmap  
    ldap_connections_number = 5  
}
```

Where `ldap_server` is the IP address of the LDAP server, `root_identity` is the name of the root in X.500 format, `password` is the root password and `base_DN` is the base DN configured in the LDAP server.

This communication is carried out after the RADIUS server knows that the user is successfully authenticated, therefore it is necessary to configure the RADIUS server to contact the LDAP server in the post-authentication section of the file `radiusd.conf`.

```
post-auth {  
    ...  
    ldap  
    ...  
}
```

5.4.2 Bridging Element (BE)

As explained in the previous section, the BE is accessed from the RADIUS server by means of LDAP. This interface is implemented using an LDAP front-end (such as the Penrose server), which can be extended by implementing custom partitions. Detailed information about how to deploy this virtual directory can be found on its web site.

The file *DAMe-testbed-penrose.zip* from the FTP of the project includes all the necessary configuration files, java libraries and the custom partition implemented in DAME. All the jar files must be copied into the *<penrose>/lib/ext* directory.

The partition must be configured by means of the file *<penrose>/conf/server.xml*. In this file, it is specified the adapter class for the new partition, and the configuration files for the new implementation:

```
<adapter name="DAMe">  
    <adapter-class>org.dame.penrose.adapter.DAMeAdapter</adapter-class>  
</adapter>  
<partition name="adapter" path="partitions/dame"/>
```

The configuration of the partition includes the following four files:

- *mapping.xml*: specifies the mapping between the base DN used in the LDAP search, and the attributes to return.
- *modules.xml*: not used.
- *sources.xml*: specifies the attributes that can be returned, in our case, the appropriate radius attributes.
- *connections.xml*: this file is used to specify the parameters for the implementation of the custom partition. Specifically, the parameters used in our custom partition are the following:
 - *MDS*: URL to the MDS.
 - *producer*: eduGAIN identifier.
 - *consumer*: eduGAIN identifier.
 - *trustStorePath*: path to truststore.
 - *trustStorePw*: password of the truststore.
 - *PDPURL*: URL of the Policy Decision Point.

This BE is also used in the service access scenario to request the eduToken to the user. This part of the BE is implemented as a Java servlet, which runs on a Tomcat server, and a signed applet used to contact the user.

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

The servlet, signed applet and web pages that implement this functionality are included in the file *DameTokenServlet.war* from the FTP of the project, which should be deployed on the Tomcat server.

5.4.3 Policy Decision Point

The PDP is the entity responsible for taking the authorisation decisions based on the policies defined in the institution and the attributes associated with the user. To deploy it, the file *DAMePDP.war* (which can be downloaded from the FTP of the project) must be copied to the *webapps* directory of the Tomcat server. This entity is configured using the *web.xml* file. The parameters specified in this file are the following:

- *keystoreType*: format of the keystore (JKS/PKCS12)
- *keystorePath*: path to the keystore.
- *keystorePw*: password of the keystore.
- *XACMLpath*: path to the directory containing the XACML policies.
- *XACMLpolicy*: name of the file containing the main XACML target access policy.

Some example XACML policies are included in the war file. These policies are automatically deployed in the directory `<tomcat>/webapps/DAMePDP/xacml`.

5.4.4 Protected Service

The Policy Enforcement Point that protects the access to the services is implemented independently of Shibboleth and PAPI. That is, neither the PAPI PoA nor the Shibboleth SP is used at this point. Instead, the protected service is based on a PHP page that checks the presence of a specific authentication parameter. The file *_protected_service_index.php* (which can be downloaded from the FTP of the project) shows an example of this PHP page.

In this way, access is granted only if the required parameter is included in the URL used to access the web page. If not, the user is redirected to the BE responsible for requesting the token. After the validation process, if it is successful, the user is redirected to the target service including the proper parameter to enable the access.

5.5 Home Institution (HI)

The main function of the HI in this testbed is to authenticate the roaming user through the RADIUS infrastructure and to provide the user attributes when requested by any other institution from the federation. Moreover, the eduToken is generated in the Authentication BE from their HI. In this testbed, two different Home Institutions have been deployed. The first is based on the use of Shibboleth and the other is based on PAPI. Therefore, specific BEs for each kind of institution have been developed. Figure 5.4 shows the elements of the HI and the software used to implement them.

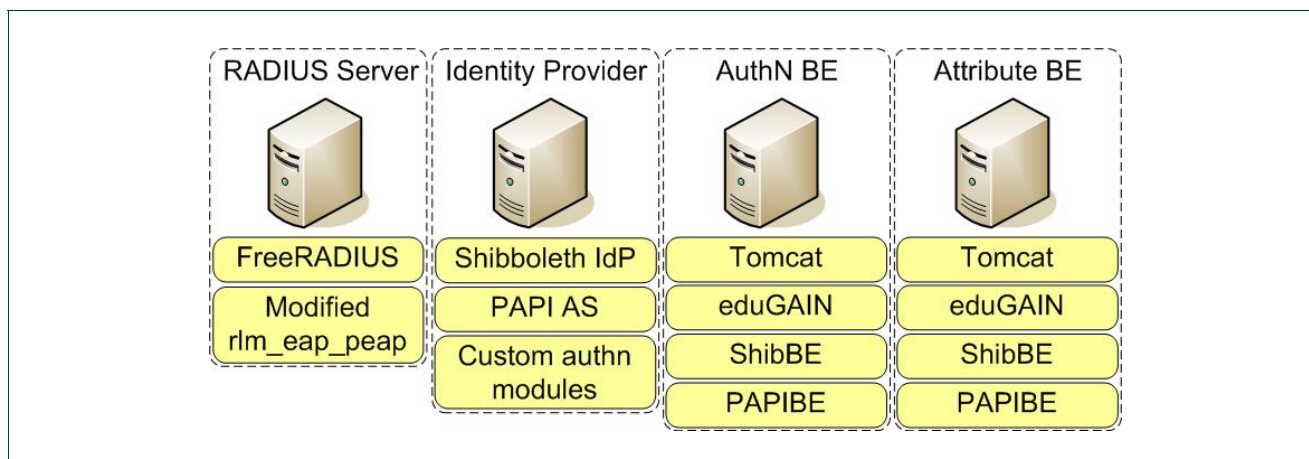


Figure 5.4: Home Institution

The sections that follow describe the process for configuring the HI, and provide sample scripts for use. Interested parties are encouraged to carry out these procedures themselves for evaluation purposes.

5.5.1 Metadata publication

Before starting to describe the different elements of this institution, it is important to remember that metadata must be used to manage and configure the interactions between the different institutions of the federation dynamically. The publication of this information in the MDS of the federation⁴ is the responsibility of a BE of the HI. However, currently this process is done manually, by means of a java application. This application is included in the file *edugainpublisher.jar* (included in the FTP of the project) and can be executed in the following way:

```
# java -jar edugainpublisher.jar <MDS URL / Federation Name> <Metadata>
```

Where the first parameter is an URL like <https://mds.ladok.umu.se:8443/DAMe>, and <Metadata> is an XML file with the metadata of the institution. Some examples are included in the file *DAMe-testbed-MDS-metadata.zip*, which can be obtained from the MDS of the project.

5.5.2 RADIUS Server

The Home RADIUS server is in charge of authenticating the users from the institution that are accessing the network in a different institution. Therefore, it will receive authentication requests through the eduroam-based infrastructure. After authenticating a user, this server should obtain the eduToken from the Authentication BE and send it to the user.

⁴ The MDS <https://mds.ladok.umu.se:8443> can be used for testing

This server is based on a standard FreeRADIUS installation, but the PEAP module has been modified to request the eduToken and include it in the response to the user. Therefore it is necessary to replace the *rlm_eap_peap* directory in the FreeRADIUS sources with the contents of the package *rlm_eap_peap.tgz*, which can be downloaded from the FTP of the project. Also, this modified PEAP module requires the *libcURL* library to be installed. After the replacement of this module, the compilation and installation instructions are the same as those described for the Remote RADIUS Server.

This modification in the PEAP module implies some changes in the *eap.conf* configuration file. Two new parameters are needed: *AuthBE* to specify the URL of the Authentication BE where to request for the eduToken, and *BESecret* to specify a shared secret for mutual authentication with the BE, as a security mechanism. The same secret should also be specified in the configuration file of the Authentication BE.

```
peap {
    ...
    # DAME attributes
    AuthBE = <BE URL>
    BESecret = <secret>
}
```

Finally, the PEAP module includes in the response to the Remote RADIUS server a new RADIUS attribute with the user's handle, therefore the definition of the new attribute must be added to the radius dictionary in the same way as described for the Remote RADIUS configuration.

5.5.3 Identity Provider

The Identity Provider (IdP) is the system responsible for managing the identity of the users that belong to the institution. The IP is accessed directly within the limits of each individual institution, but through the federation it is accessed via the proper Bridging Elements (the Attribute BE to obtain the user attributes and the Authentication BE to obtain authentication sentences about a specific user).

5.5.3.1 Shibboleth IdP

This is a typical Shibboleth IdP installation, although it is configured to use a custom authentication module. Therefore, the first step is to install and configure Shibboleth in the usual way. Detailed information about this process can be found on the Shibboleth web site⁵.

After the Shibboleth IdP is working properly, it must be configured to use a specific authentication module. The module *DAMeAuthn.pm*, which is included in the package *DAMe-testbed-ShibIdP.zip* from the FTP of the project, must be copied to some directory of the PERL path, for example */etc/perl/Apache2*. The reason for developing a new module is that users will be authenticated by the RADIUS server, not by the IdP. Therefore, the function of this element is only to provide the authentication statement about the users specified by the Authentication BE, and not to verify any kind of credentials. But as a security mechanism, the IdP can only trust

⁵ <https://spaces.internet2.edu/display/SHIB/WebHome>

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

the proper BE to provide these statements, therefore it is necessary that this BE presents its digital certificate, and that the authentication module verifies its identity. In this way, it is necessary to set the value of the accepted distinguished name (DN) in the *DAMeAuthn.pm* module to the DN of the trusted Authentication BE.

After the authentication module is properly included, the Apache server must be configured for when Shibboleth authentication is required. This is done by including the following lines in the file *httpd.conf*:

```
<Location /shibboleth-idp/SSO>
  AuthType DAME
  AuthName "DAMe Verification Service (VVS)"
  PerlAuthenHandler Apache2::DAMeAuthn
  require valid-user
</Location>
```

Finally, in the SSL configuration section of the Apache2 web server where the IdP runs, it is necessary to specify that the client need to present a certificate to access this server. Also, Apache should export the client certificate information to make it available to the PERL modules. This is specified using the following options:

```
SSLVerifyClient require
SSLOptions +ExportCertData +StdEnvVars
```

5.5.3.2 PAPI AS

Similar to the Shibboleth IdP, this is a standard PAPI AS, but configured to use a new authentication module. This provides the authentication cookie for the users based only on their identity. Detailed information about how to install it can be found on the PAPI web site⁶.

The new authentication method is based on the BasicAuth authentication, but it authenticates the user only by its username. The file *DAMeAuth.pm* that is included in the package *PAPI-AS-update.tgz*, which can be downloaded from the FTP of the project, defines this method. It must to be copied in one of the PERL directories (for example */usr/local/lib/perl/5.8.7/PAPI*). Again, this IdP can only trust in the specific BE that is requesting the user's authentication cookies, therefore it is necessary to configure the DN of this BE using the method *VerifyUser* included in this file.

The *AuthServer* has been slightly modified, because the current PAPI AS regenerates the *PAPIAuthNcook* each time a request is made, but we need to maintain this cookie constant to allow SSO. Therefore it is necessary to replace it with the file included in the package *PAPI-AS-update.tgz*.

The *DAMeAuth* authentication method has been configured in the configuration file of the AS (*AuthServer.cf*). Also, the set of user attributes that must be released using the configuration variable *defAssertion* are specified.

```
my $authType = "DAMe";
...
$$cfg{defAssertion} = 'uid=<papi var="PAPLuid"/>
```

⁶ <http://papi.rediris.es/documents.html>

```
schacPersonalPosition=<papi var="schacPersonalPosition"/>,  
preferredLanguage=<papi var="preferredLanguage"/>,  
eduPersonPrincipalName=<papi var="eduPersonPrincipalName"/>,  
eduPersonEntitlement=<papi var="eduPersonEntitlement"/>;  
...  
} elseif ($authType eq "DAME") {  
    $$cfg{authenticationHook} = \&PAPI::DAMEAuth::VerifyUser;  
    $$cfg{credentialHook} = \&PAPI::DAMEAuth::UserCredentials;  
    $$cfg{attrRequestHook} = \&PAPI::DAMEAuth::UserAttributes;  
    $$cfg{basicAuthDB} = "test.database";  
}
```

Finally, in the users' database (*database.source*), the attributes for the user are specified. An example of this file is included in the package *PAPI-AS-update.tgz*, as shown:

```
# Users  
<class>::<id>::<crypt'd_passwd>::<alt_name>::<grp_ids>::<list_of_site_ids>::<attributes>  
user::lolo::SDm2sZ2B6gPQw::Manolo::group1::dame.site:  
eduPersonPrincipalName=lolo@dif.um.es  
schacPersonalPosition=urn:mace:terena.org:schac:personalPosition:um.es:student  
preferredLanguage=en  
eduPersonEntitlement=student  
  
# Groups  
# <class>::<id>::<alt_name>::<list_of_site_ids>  
group::group1::Group One::dame.site  
  
# Sites  
#<class>::<id>::<description>:  
:<PoA URL>::<PoA token request URI>::TimeToLive::<service_id>::<PoA URI>  
site::dame.site::PAPI protected site:  
:https://rook.inf.um.es::cookie_handler.cgi::1800::UMUDAMEPoA::/protected
```

Similarly to Shibboleth, the Apache web server (where the AS runs) must be configured to request the client certificate to access this server. Also, the client certificate information must be exported to make it available to the PERL modules. This is specified with the following options:

```
SSLVerifyClient require  
SSLOptions +ExportCertData +StdEnvVars
```

5.5.4 Authentication Bridging Element

The Authentication BE is responsible for creating the eduToken for a user that has been previously authenticated by the RADIUS server, based on the information provided by the IdP of the institution. Specifically, two different Authentication BEs have been developed; one for the Shibboleth IdP and another one for the PAPI AS.

The war file for the first BE is *ShibBE.war*, while the other is *PAPIBE.war*. Both files can be downloaded from the FTP of the project, and should be copied to the *webapps* directory of Tomcat to deploy them. These entities

Project:	GN2
Deliverable Number:	DJ5.3.4
Date of Issue:	17/11/08
EC Contract No.:	511082
Document Code:	GN2-08-208v2

read parameters from their respective *web.xml* file. This file needs to be adapted to each specific site. Specifically, the parameters for the Shibboleth Authentication BE are:

- *idpURL*: URL of the Shibboleth Identity Provider SSO service.
- *signingPropsFile*: path to the eduGAIN signing properties file.
- *truststorePath*: path to the truststore.
- *truststorePw*: password of the truststore.
- *keystoreFormat*: format of the keystore (JKS/PKCS12).
- *keystorePath*: path of the keystore.
- *keystorePw*: password of the keystore.
- *sharedSecret*: value of the secret shared with the RADIUS server.

The configuration of the PAPI Authentication BE is analogous to the previous one. The only difference is that the *idpURL* should specify the URL of the PAPI Authentication Server.

Because both elements rely on the eduGAIN infrastructure, it is necessary to set up the configuration files of eduGAIN correctly.

5.5.5 Attribute Bridging Element

The Attribute BE is the element responsible for receiving attribute requests through eduGAIN, translate them into the specific protocol used in the institution, and send them to the IdP. Subsequently, when it gets the attributes from the IdP, this BE should build an eduGAIN attribute statement to send back the requested attributes. Again, two different BEs have been developed; one to be used in an institution based on PAPI, and another one to be used in an institution based on Shibboleth.

The attribute BEs are included in the same war files than the authentication BEs. These elements are configured by means of the *web.xml* file. The specific parameters for the Shibboleth Attribute BE are:

- *componentId*: urn of the component.
- *truststorePath*: path to the truststore.
- *truststorePw*: password of the truststore.
- *keystoreFormat*: format of the keystore (JKS/PKCS12).
- *keystorePath*: path of the keystore.
- *keystorePw*: password of the keystore.
- *aaurl*: url to the Shibboleth Attribute Authority.

The configuration of the PAPI Attribute BE is similar to the previous one, but instead of the *aaurl* parameter the *idp* parameter specifies the url to the PAPI AuthServer. Also, a new parameter is included; *askeypath*. It is necessary to specify the path to the public key of the AuthServer, which should be used to decrypt the user's attributes.

As before, it is necessary to configure the eduGAIN configuration files.

5.6 Further steps

There are some improvements that can be done in the testbed described above. Some of them are only related to software upgrades (for example the use of FreeRADIUS 2.0 and XSupplicant 2.0), while others are related to security and new functionalities (for example the use of RadSec protocol instead of RADIUS).

5.6.1 Use of FreeRADIUS 2.0

One of the elements of the testbed that can be upgraded is the FreeRADIUS software. Currently, version 1.1.4 is used but version 2.0 is available. The latest version includes several new functionalities, such as IPv6 support, more EAP types and the use of virtual servers.

One of the advantages of this software is its modular design. This allowed the implementation of the DAME features needed for the testbed in very specific areas. In particular, only the PEAP and LDAP modules were modified. Therefore, to upgrade the current FreeRADIUS version to the latest version, it is only necessary to adapt the currently used modules or to modify the modules of that version to include the required functionality. In fact, there is only a minor difference between FreeRADIUS 1.1.4 and FreeRADIUS 2.0 modules, so it is very simple to adapt the current PEAP and LDAP modules to the new release.

5.6.2 Use of XSupplicant 2.0

Since the XSupplicant was selected as the base supplicant for the OpenSEA Alliance⁷, its main objective has been to provide support for WindowsXP and to develop an easy to use graphical interface. The result of this is that the latest release from XSupplicant, version 2.0, is not available on Linux. The Open1X project web page recommends to their Linux users to continue using version 1.2.8 because the latest release is not fully adapted. Therefore it is not possible to upgrade the current version of XSupplicant used in the testbed (1.2.8) to the latest release easily.

5.6.3 RadSec support

Because this testbed is based on the current *eduroam* deployment, the underlying AAA infrastructure is based on RADIUS servers. Therefore, communication between Home and Remote RADIUS servers is insecure. Some mechanism (such as IPSec) can be used to protect this communications channel, but another option is to use the RadSec protocol, where communication between the AAA servers is protected by TLS channels. Some preliminary tests have been done in the testbed using RadSec.

⁷ <http://www.openseaalliance.org>

The easiest way to extend the current testbed and include RadSec support is to use the radsecproxy⁸ software. It is a generic RADIUS proxy that, in addition to the usual UDP transport, also supports TCP (and TLS). With this method it is only necessary to collocate a RadSec proxy with each RADIUS server to enable secure communications and also the other benefits from RadSec (such as dynamic peer discovery). The main advantage of this option is that the general processing remains in the RADIUS servers, while the proxies only protect communications, so the current FreeRADIUS servers can be used without modifications in the testbed.

An alternative method is to use a full RADIUS server with RadSec support, such as Radiator⁹. This option implies the extension of the Radiator PEAP module to include DAME features (the recovery of the eduToken and sending it to the user). Also, it would be necessary to find out if the LDAP interface in Radiator can be used as required by DAME, or whether it would have to be adapted. However, this alternative means it is not necessary to maintain two different servers (the RADIUS server and the proxy) in each institution.

⁸ <http://software.uninett.no/radsecproxy>

⁹ <http://www.open.com.au/radiator>

6 Conclusions

This document describes how a prototype for unified Single Sign On has been implemented and tested. The results demonstrate the technological feasibility of such an approach, and thus imply the possibility of a new service deployment by NRENs.

The fact that uSSO is a direct service to end-users makes its deployment extremely sensitive to client availability: it will only achieve widespread usage if several clients in several platforms support it. Both the integration with the application access part (typically, a Web browser) and the network access part (typically, a 802.1x supplicant) require specific modules to support the uSSO solution presented here. But while the usual design of Web browsers make this achievable by means of specific add-ons (applets or browser plugins, for example), common 802.1x supplicants do not incorporate a similar extension mechanism. It is therefore necessary to include an additional task of dissemination of the technology and its potential, presenting the results at the standard bodies and seeking collaboration with open-source development communities and industry initiatives. In this respect, the possibility of aligning the eduToken concept with the CardSpace paradigm looks very promising.

Furthermore, the concepts demonstrated by the uSSO prototype development have opened new possibilities for extending federated identity to network services beyond the current applications in Web access. The results of this work have inspired new identity profiles to be used in areas such as network measurement, dynamic network resource allocation and presence protocols, extending the idea of uSSO in directions that were not even considered at the beginning of the project. This is incredibly valuable.

It should be noted that further work is in progress. RedIRIS plans to make a complete operational deployment of a uSSO service as part of its experimental PASITO network framework. This deployment will provide additional insight into the reliability, scalability, and (most important) usability from the network manager's perspective. The usability report will address aspects related to installation, configuration and operation of a multi-institutional uSSO infrastructure. The results of this work will be made available in due course.

7 References

- [MJ5.1.5]** M. Stanica, T. Wiberg, K. Wierenga, S. Winter, J. Rauschenbach. Milestone MJ5.1.5: JRA5 Glossary of Terms - Second Edition. January 2007.
- [DJ5.3.1]** B. Kerver, M. Stanica, J. Rauschenbach, K. Wierenga. Deliverable DJ5.3.1: Documentation on GÉANT2 unified Single Sign-On (uSSO) Requirements. February 2007.
- [DJ5.3.2]** Ó. Cánovas, M. Sánchez, G. López, R. del Campo, S. Neinert, J. Rauschenbach, I. Thomson. Deliverable DJ5.3.2: Architecture for Unified SSO. Mai 2008.

8 Acronyms

AAI	Authentication and Authorisation Infrastructure
AP	Access Point
AS	Authentication Service
BE	Bridging Element
CRL	Certificate Revocation List
DAMe	Deploying Authorization Mechanisms for Federated Services in the eduroam Architecture
DN	Distinguished Name
DoS	Denial of Service
HI	Home Institution
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	identity Provider
JDK	Java Development Kit
LoA	Level of Assurance
MDS	Metadata Service
PDP	Policy Decision Point
PEAP	Protected Extensible Authentication Protocol
r-BE	Remote Bridging Element
RI	Remote Institution
SAML	Security Assertion Markup Language
SP	Service Provider
SSO	Single Sign On
UMU	University of Murcia
USTUTT	University of Stuttgart
UUMEA	University of Umea
uSSO	unified Single Sign On