31-03-2015

Open Call Deliverable OCD-D1.2
Final Report (CLASSe)

Open Call Deliverable OCD-D1.2

Grant Agreement No.:	605243
Activity:	NA1
Task Item:	10
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	UMU
Document Code:	GN3PLUS14-1286-90
Authors:	A. Perez-Mendez , R. Marin-Lopez, G. Lopez-Millan, D. W. Chadwick

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 605243 (GN3plus).

Abstract
Deliverable 1.2 (D1.2) represents the Final Deliverable or Final Report of the CLASSe project. It includes chapters for each WP, namely: WP1 Management, WP2 results of test and deployment of OpenStack/Moonshot prototype, WP3 report on traditional and real SSO, WP4 report of design and testing of CoIs/VOs, and WP5 report on dissemination and exploitation activities.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This document represents the Final Deliverable (D1.2) for the GÉANT3 Open Call project CLASSe. It includes all the relevant information regarding the project management, outcomes, dissemination and exploitation. This document is organized as follow:

- Section 1 briefly introduces the project outcomes, impact and potential continuation.

- Section 2 focuses on WP1 Management, highlighting the status of the different Deliverables and Milestones worked out during the project. The detailed report about the management work done in CLASSe can be found in Deliverable 1.1 [D1.1].

- Section 3 focuses on WP2 Analysis and Integration of ABFAB in Cloud Services. This section presents a gap analysis of the selected implementations for both technologies: Moonshot implementation of ABFAB protocols and OpenStack like cloud solution. This gap analysis focuses on the missing points required for the correct integration of both technologies. Then, Section 3 presents how these gaps have been solved (see Deliverable 2.1 [D2.1]) and then proposes solution of the integration of OpenStack and Moonshot. It also presents the results of the tests done to this integration in order to demonstrate its feasibility.

- Section 4 covers WP3 Research on Federated Single Sign On (SSO) for Cloud Services. It includes an analysis of the SSO solutions provided by Moonshot (traditional SSO) and presents some improvements for two main uses cases: cloud-to-cloud SSO and network-to-cloud SSO. Then, it includes an analysis of how to implement real SSO in Moonshot and OpenStack for federated environments. In this WP, we propose to make use of the EAP Re-authentication Protocol (ERP) in order to provide real SSO. We have implemented a proof-of-concept of the ERP integration in Moonshot for OpenStack services whose details are in Deliverable 3.1 [D3.1]; Moreover, we have carried out a detailed performance analysis and have analysed how ERP can improve the typical use of cloud services over a federated environment, like the one presented in GÉANT.

- Section 5 introduces the work done in WP4: Research on Communities of Interest (CoIs) for Cloud Services. CLASSe reoriented this topic, moving from the concept of CoIs to the concept of Virtual Organizations (VO) which fits better with the original proposal. It also introduces the concepts of small and large VO for cloud services, and presents the solutions designed and implemented for the OpenStack solution. Tests done for these implementations are also described.

- Section 6 summarizes the work done in WP5: Dissemination and Exploitation. It includes activities such as participating in standardizations groups, publications in highly ranked journals and book chapters, conferences and workshops and several IETF Internet-Drafts. It also includes the exploitations works done by the partners of the project. Finally, Section 7 presents some conclusions and future work of the CLASSe project.

1 Introduction

This document presents the work done under the GÉANT3plus Open Call CLASSe project for the last 18 months of the GÉANT3 project.

This introduction presents the main outcomes of the project, taking into account the main objectives defined in the original proposal. It also highlights the project impact and the future work. Then, it describes the work done in the different Work Packages (one per section), providing a summary of the main results in each one of them. Finally some conclusions and future work are provided.

1.1 Project outcomes

CLASSe covers call topic 17 in the GN3plus Open Call by allowing GÉANT users to access cloud services with their home institution credentials. For that, two main objectives were defined in the original proposal. Namely:

- *Objective 1: Ubiquitous access to cloud services with ABFAB.*

"The aim of this project is to investigate and to define the deployment of the Application Bridging for Federated Access Beyond web (ABFAB) standard, as implemented in the project Moonshot [MOONSHOT] technology, over the GÉANT network for cloud services (although they can potentially be used for any kind of application protocol or service)."

- *Objective 2: Improvements for the federated access to cloud services with ABFAB.*

"For the second objective, this proposal will focus on two related concepts: SSO (Single Sign-On) and CoI (Communities of Interest). In the first case, it will test and analyse traditional SSO and research into the design and implementation a generic and effective real SSO solution for federated cloud services over the GÉANT network. In the second case, it will also research into the fine-grained access control to cloud services for different groups of users, based on the Communities of Interest concept from ABFAB. We will label these objectives 2.1 and 2.2.":

    - *Objective 2.1: Single Sign On (SSO) and ABFAB*
    - *Objective 2.2: Community of Interests (CoIs) in ABFAB*

To simplify the explanation, we have re-numbered these objectives as Objective 1 (original Objective 1), Objective 2 (original Objective 2.1) and Objective 3 (Original 2.3). Thus, they define three main aspects were defined in CLASSe at the beginning of the project.

As explained, the **Objective 1** (addressed in WP2) was related to the integration of the advanced authentication and authorization mechanisms proposed by the IETF ABFAB WG [ABFAB], and implemented under the Moonshot Project [MOONSHOT], into the OpenStack Cloud software [OPENSTACK].

In order to achieve this integration we developed a gap analysis [D2.1] of the prototype OpenStack/Moonshot implementation available at the start of the project, in order to find its deficiencies. As a result of this gap analysis new functionality was added to OpenStack (i.e. support for protocol independent federated identity management), and to Moonshot, (i.e. end user interface, bug fixes, etc.). [D2.1] describes this gap analysis and the associated actions. One of the most important outcomes of CLASSe is that the OpenStack community has incorporated the protocol independent federation mechanism into the main branch code, ensuring that future support and maintenance of the ABFAB integration will continue in future releases. To test this integration a CLASSe testbed (Section 4) has been deployed between University of Kent, University of Murcia and QALAB [QALAB].

The **Objective 2** (addressed in WP3) was related to the improvement of the end user experience through the Moonshot and OpenStack integration, and was focused on the improvement of the Single Sign On (SSO) mechanisms defined by ABFAB and how these mechanisms could be applied in scenarios were multiple service providers are involved (cloud-to-cloud SSO) and where an authentication to the network access service (in eduroam) is leveraged to access different cloud services (network-to-cloud SSO). A more detailed description of the uses cases can be found in Section 4. To achieve the second objective we started analysing the use of the existing ABFAB SSO mechanism deployed in Moonshot in the two target scenarios (cloud-to-cloud and network-to-cloud). It is important to note that the existing ABFAB SSO mechanism implies a full EAP-TLS authentication between the end user and the Identity Provider for each access to the service. Here two improvements have been carried out: first, to improve the Moonshot's end user interface helping the user to pre-select the right identity for SSO when accessing the service; second, to improve the SSO overload by adding the support of EAP-TTLS resumption mechanism [EAP-TTLS] into Moonshot (Section 4). The second action in this objective was to provide a real SSO mechanism for ABFAB, allowing the reduction of both the end user time to access the service and the used bandwidth in the GEANT network. It is important to mention that a real SSO mechanism means the one that takes advantage of a previous end user authentication in order to provide further access to the end user without the need of a new full authentication process. The solution proposed in CLASSe is to integrate the EAP Re-authentication Protocol (ERP) [ERP] into ABFAB/Moonshot by means of ERP extensions to the existing GSS-EAP mechanism (Section 4). Regarding Moonshot, the ERP extensions to GSS-EAP have been implemented and integrated in the project. Regarding the standardization effort (ABFAB), these extensions has been published as an IETF Internet draft and presented to the IETF community [GSS-ERP]. WP3 outcomes have been implemented and tested over the CLASSe testbed network described before and the results can be found in Section 4.

The **Objective 3** (addressed in WP4) was to support the dynamic formation of communities of interest (CoIs), so that any users will be able to form their own CoIs for access to their own private cloud services. Unfortunately CoIs cannot be created without IdP involvement, which makes it very difficult for users to form their own CoIs. Consequently, we decided that the Virtual Organisation concept was a much better group

collaboration mechanism to allow users to manage access to their own cloud services. With project management agreement, we therefore concentrated on providing support for VO role management within OpenStack. The end result, Section 5, is that an OpenStack Keystone administrator can now set up VO role management functions that will allow him/her to invite individual users from individual IdPs within a federation, to join his/her VO (or community of interest), and to give them appropriate privileges within his/her OpenStack service. Both small scale and large scale VOs are supported. For small scale VOs, the OpenStack Keystone administrator administers all the users him/herself. For large scale VOs, the administrator invites key individuals to become VO administrators, and then delegates to them permission to manage the various VOs. A working prototype is available at the University of Kent [VODEMO]. There was insufficient time in the CLASSe project to get this code incorporated into the main branch code, so that at present, it only exists as an add-on to the Icehouse release of OpenStack. A future project could be to integrate this into the main branch of OpenStack, and to add additional functions to Keystone to allow VO roles to be shared between federated clouds and other services.

1.2 Project Impact

The expected impacts from the GN3plus Open Call topic 17 were defined as follow:

- *Expected Impact 1: "The results will contribute to the use of applications and network functions beyond the web and introduce new ways to access applications beyond those commonly used today, i.e. VPN, federated access, etc."*

- *Expected Impact 2: "The proposal will provide middleware layers that can be used to support different types of cloud services, spanning from those offering storage to those offering network virtualisation capabilities."*

The proposal defined a set of actions to address each expected impact. In the following, we provide an explanation how these actions have been finally carried out in the project.

Expected Impact 1:

- Integration of Cloud services into the eduroam's RADIUS network identity federation by using Moonshot technology.

OpenStack has been extended with a generic authentication and authorization framework allowing the easy integration of advanced authentication mechanisms such as the one proposed by Moonshot. Thanks to the integration of OpenStack and Moonshot, cloud services providers deployed over the GEANT network will be able to allow the access to federated users in a seamless and easy way, by delegating the authentication and authorisation process to the underlying RADIUS infrastructure.

- Collaboration with existing standardization bodies and GÉANT3plus

CLASSe project has been actively collaborating with the different standardization bodies around the project objectives. On one hand, University of Kent has been collaboration with the OpenStack development group in order to standardize the Moonshot integration into the OpenStack core code; on the other hand, University of

Murcia has been collaborating with the IETF working groups RADEXT and ABFAB where we have presented our proposal for integration of ERP into Moonshot (see [D3.1]) and a proposal for RADIUS fragmentation for large authorization attributes [RADFRAG] (i.e. for transporting SAML-based authorization information), accepted as RFC[1] (D2.1). Besides, CLASSe partners have published project results in conferences, book chapters, and high ranked journals. Moreover, Dr. Alejandro Pérez Méndez's PhD has concluded during the life of CLASSe project and contains research outcomes developed during the project. Some of the result from the PhD form part of the CLASSe results and came from the close relationship with the Moonshot project (Section 6).

- Future Research Collaborations and Initiatives

University of Murcia will continue collaborating with other institutions in order to improve the use of ABFAB and Moonshot for services, including cloud ones. One of these collaborations has been materialized in the form of a contract between University of Murcia and Janet (UK). It is also expected to continue the collaboration with University of Kent and GEANT in this area of research during GN4.

Expected Impact 2:

- Integration of Moonshot into OpenStack Core Software Release

This new authentication and authorization framework has been included in the main OpenStack branch ensuring the support of those advanced authorization mechanisms beyond the end of the CLASSe project by the OpenStack community (Section 3).

- To improve GÉANT end user's experience by using the eduroam's RADIUS network and cloud services with Single Sign On procedures

Real SSO functionality with the ERP extensions to GSS-EAP will improve end user experience allowing a better and faster secure access to cloud services over the GÉANT network. There will be a bigger impact in those services required a number of connection to the cloud services, such as those based on HTTP (Section 4).

1.3 Potential Project Continuation

Some CLASSe results are expected to continue in GN4. The deployment of ABFAB like a security service over the GEANT network infrastructure and the deployment of dynamic trust relationships for those services (Trust Router) planned for GN4 SA5-T2 are related with CLASSe area of work. Beside, the work defined in JRA3-T1 about the management of groups of users for cloud services is directly related with the work in VOs (CLASSe WP4).

CLASSe partners will aim to apply for new Open Calls initiatives in GN4 in order to both work in the deployment of federated cloud services over the GÉANT network, and to analyse the use of technologies such as the Trust Router to improve this deployment. Regarding VOs, a future GN4 project could be to integrate the VO code into

[1] RFC number not assigned yet at the end of the project

the main branch of OpenStack, and to add additional functions to Keystone to allow VO roles to be shared between federated clouds and other services.

In the following we describe the work done in the project per Working Package (Section 2-5). Finally some conclusions and future work are presented (Section 6).

2 WP1: Management

Deliverable 1.1 [D1.1] provides a short overview of the evolution of the CLASSe project, the management tasks and how the involved partners have been coordinated. It also describes the contingences during the project and the resolutions adopted. Regarding RAG reports, these can be found in GN3plus Intranet > JR0: Open Call Projects > CLASSe > Documents > DoW > WP1, and there are not significant issues regarding the budget.

A summary of the Milestone and Deliverables is described next. For more detail, please refer to [D1.1]

Milestone/ deliverable code	Milestone/deliverable name	Due Date	Delivery Date	Achieved Yes/No	% complete	Comments
D1.1	Management Reports	M1-M18	M1-M18	Yes	100	
MS1	ABFAB for cloud services: Gap Analysis	M3	M3	Yes	100	
MS2	Dissemination plan	M6	M6	Yes	100	
D2.1	Supporting tools for the Moonshot/OpenStack implementation	M9	M11	Yes	100	Delayed by delay in T2.2 due to Kent's RA's contract finishing. Notified to Technical coordinator in RAG report.
MS3	Analysis and Design of SSO for cloud services	M9	M9	Yes	100	
MS4	Research activities for ABFAB/OpenStack. Test	M12	M18	Yes	100	Delayed by delay in T2.2

	and deployment					due to Kent's RA's contract finishing. Notified to Technical coordinator in RAG report.
MS5	Cloud-to-cloud and network-to-cloud SSO	M12	M12	Yes	100	
MS6	User friendly administrator tools for managing small ABFAB cloud communities of interest	M12	M17	Yes	100	Delayed by Kent's RA's contract finishing. Notified to Technical coordinator in RAG report
MS7	User friendly administrator tools for managing large ABFAB cloud communities of interest	M15	M18	Yes	100	Delayed by Kent's RA's contract finishing. Notified to Technical coordinator in RAG report
MS8	Publications	M15	M15	Yes	100	No Milestone document generated. Expressed in RAG report.
D3.1	Proof-of-Concept	M18	M18	Yes	100	
D1.2	Final Report	M18	M18	Yes	100	

3 WP2: Analysis and Integration of ABFAB in Cloud Services

The overall objective for WP2 is to cover the first of the goals (Objective 1) of this project:

"…to investigate and to define the deployment of the Application Bridging for Federated Access Beyond web (ABFAB)/Moonshot [...] technology over the GÉANT network for cloud services …".

This goal has been firstly approached by the performing of a gap analysis of the current OpenStack/Moonshot implementation developed by the University of Kent. This analysis is described in section 3.1. After that, we have made the necessary improvements to the existing OpenStack/Moonshot prototype to implement the missing functionality discovered during the gap analysis, as detailed in section 3.2. Finally, we have tested the prototype again, verifying the new functionality, and created detailed installation guides so that other partners in GÉANT can easily deploy and test the prototype, as described in section 3.3.

3.1 GAP analysis

This section provides a gap analysis derived from UMU's experience of setting up and executing the OpenStack/Moonshot code for providing federated access to cloud services in GÉANT (T2.1), developed by KENT before the start of the project. This gap analysis aims to ensure the usability and manageability of this implementation in the GÉANT network, serving as input for Task 2.2, where they will be addressed.

To gather all the gaps we have set up a testbed that simulates what we consider will be the most typical real scenario: an organization member of the GÉANT network deploys a cloud service. This service is connected to the RADIUS infrastructure to allow federated authentication and authorization of any end user that comes from another member organization. Hence, three different machines comprise the testbed:

1. End user. This machine deploys the OpenStack/Moonshot client side of the software. It consists on a Ubuntu 13.04 (Raring Ringtail) [UBUNTU13], with the federated OpenStackSwift client [FEDSWIFTCLIENT] installed. It also has the Moonshot [MOONSHOT] software installed.

2. OpenStack server. This machine deploys the OpenStack/Moonshot server side of the software. It consists on an Ubuntu Server 12.04.3 LTS (Precise Pangolin) [UBUNTU12], with the federated OpenStack Keystone [FEDKEYSTONE] installed. It also has installed the Moonshot software.

3. RADIUS server. This machine deploys the FreeRADIUS 2.2.0 [FREERAD] server. It connects the testbed with the RADIUS infrastructure, to allow a federated operation. It also generates the SAML assertion required by the OpenStack server to perform authorization, as current RADIUS servers in the eduroam infrastructure do not implement the Moonshot's IdP functionality.

Some of the gaps detected during this analysis led to a block situation, where a fix was required in order to allow the process to continue. Hence, the OpenStack/Moonshot software has evolved through this gap analysis process. These gaps are marked in the analysis results as *Critical* and *Solved*.

3.1.1 List of gaps

To make this analysis more comprehensive the gaps have been grouped into two different categories:

1. **Setting-up**. This category includes the gaps found when setting up an OpenStack deployment including support for the Moonshot federated authentication and authorization technology. Specifically, it comprises gaps related to the installation and configuration process. These gaps have a direct impact on the deployment potentiality of the solution, as any difficulties found in the setting up process will preclude systems administrators from installing it.

2. **User experience**. This category includes the gaps found during the utilisation of the software. These gaps have a direct impact on the end user experience, as if it does not fulfil end user expectations, she will not use it and will prefer other authentication mechanisms.

Each gap is described in an individual card, which provides all the information related to it. The fields of the card are explained in the following example:

CODE. This field provides a unique name to the GAP for reference within the project	
Summary	**This field provides a brief (1-2 lines) description of the gap**
Description	This field provides a complete description of the gap, providing information of why it is a gap, and what are the actions that would be required to overcome the issue.
Source of the gap	This field indicates which module is the source of the issue and, therefore, who would be responsible for its solution. It can be any of: • **OpenStack**. The source of the issue is located on the OpenStack code. • **Moonshot**. The source of the issue is located on the Moonshot code. • **OpenStack/Moonshot**. The source of the issue is located on the integration code provided to us.
Expected solution	This field describes what is expected to address the gap.
Severity	This field indicates the severity of the gap. Those gaps with higher severity should be fixed first. It can be one of: • **Critical**. The gap completely precludes the setting up or execution of the

	software. Its addressing is mandatory, as the code would be useless otherwise. • **High**. The gap generates great difficulties either on the setting up or the execution of the software. It requires a high level of expertise to go around it to continue with the testing process. Its addressing is mandatory, as they preclude their deployment in organizations that do not participate in the project. • **Medium**. The gap makes the software less useful or interesting than it should be, due to missing functionality or bad user interface. Addressing it is recommended in order to maximize its success possibilities. • **Low**. The gap is a minor inconvenience to the user, for which workarounds already exist. It does not interfere significantly with the normal functionality of the software but fixing it would improve the user experience.
Relevance	This field indicates the relevance of the gap for the CLASSe project to achieve its objectives. Those gaps with higher relevance will have a greater likelihood to be addressed in Task 2.2. It can be any of: • **High**. This gap has a high relevance and should be addressed in Task 2.2. • **Medium**. This gap has a medium relevance and might be addressed in Task 2.2. • **Low**. This gap has a low relevance and probably will not be addressed in Task 2.2.
Status	This field indicates the current status of the gap. Some progress has already been made during the gap gathering process, thus some gaps are already solved or being worked on. It can be any of: • **Solved**. The gap has been solved and no longer applies to the current status of the code. This is typically applicable to critical gaps that were required to be solved to continue with the analysis process. • **Pending.** The gap is pending to be addressed by Task 2.2. • **Queued.** The gap is either a OpenStack/Moonshot issue that is not expected to be addressed by the CLASSe project, or is not within the remit of the CLASSe project to fix (e.g.it is a Moonshot issue).

3.1.1.1 *Setting-up*

This subsection lists the gaps we have found when trying to install the OpenStack server and client with the support for Moonshot-based federated authentication.

GAP-SU-01	
Summary	**The software is based on the OpenStack *Grizzly* release (April 2013), but should be based on the *Havana* release (Oct 2013)**

Description	The OpenStack/Moonshot implementation was initially developed using the OpenStack *Grizzly* release. However, before the CLASSe project started, the OpenStack *Havana* release was published in Oct 2013. This release included changes to the Keystone protocol, which was upgraded from v2 to v3, thus making it incompatible with the existing code. Working with a deprecated/old release makes little sense, as one of the objectives of the project is to try get this federated and SSO authentication support accepted by the OpenStack community. Thus, it should be done using the latest version of their APIs.
Source of the gap	OpenStack/Moonshot
Expected solution	Existing integration code needs be ported to support the new keystone protocol (v3) included with the *Havana* release.
Severity	Critical
Relevance	High
Status	Solved

GAP-SU-02	
Summary	**There is not an installation guide for the server side**
Description	While an installation guide of the OpenStack/Moonshot software exists for the client side, its counterpart for the server side does not exist. Without a good quality installation and configuration guide, it is almost impossible that any organization considers installing this software, as the process is far from being trivial.
Source of the gap	OpenStack/Moonshot
Solution	Create an installation guide for the server side.
Severity	High
Relevance	High
Status	Solved

GAP-SU-03	
Summary	**Initial installation guide for the server is incomplete and contains several errors**
Description	The first version of the installation guide for the server should be considered as a draft, as it contains a number of mistakes and omissions. Besides, it is not well organised, making it difficult to follow through the installation process. The following is a list of the errors found: • The command line to include Moonshot's repository is incorrect. It misses some spaces and several "&&" symbols to concatenate commands. It should be: `echo "deb http://repository.project-moonshot.org/debian-moonshot sid main" > /etc/apt/sources.list.d/moonshot.list&&wget -O -`

	`http://repository.project-moonshot.org/key.gpg | apt-key add - && apt-get update` • The installation of the specified Linux kernel image, as well as of the moonshot-ui,is not required. • The installation of libradsec [LIBRADSEC] must be done after adding the Moonshotsoftware repository. Otherwise it will not be available. • No information about how to set up the FreeRADIUS server is provided. • The example *keystone.conf* entry does not specify that *federated* must be also added to the *method* list, as it appears in the example-keystone.conf file in the federated-docs folder. • The command for the creation of the ABFAB service is incorrect. Parameters should be prepended with `--`, and not with `-`. • The command for the creation of the end point is incorrect. Should be `keystone endpoint-create --service-id=<service uuid> --publicurl=http://localhost --internalurl=http://localhost --adminurl=http://localhost` • The mapping file example is incorrect. Internal attributes *role* and *project* require UUIDs as their value, not a textual name. • Documentation does not follow a temporal line. Section 7 dealing with devstack [DEVSTACK] installation should be presented before. Readers may find out that they have downloaded Keystone code manually unnecessarily if they wanted to use devstack. • Regarding the devstack installation, the Keystone repository and branch variables are not well specified. Besides, the branch name is incorrect. They should be: `KEYSTONE_REPO=${KEYSTONE_REPO:-https://github.com/kwss/keystone.git}` `KEYSTONE_BRANCH=${KEYSTONE_BRANCH:-fed-plugin-moonshot}` • The Keystone requires some fields to be present on the *SAML assertion*received from the IdP(i.e. *Subject* and *NotBefore* and *NotAfterConditions* are required). If they are not present, authorization fails with no clear indications. The installation guide does not include information about them.
Source of the gap	OpenStack/Moonshot
Expected solution	The installation guide should be corrected, completed, and re-published.
Severity	High
Relevance	High
Status	Pending

GAP-SU-04

Summary	**Client repository is incorrect in the installation guide**
Description	The current installation guide for the client indicates *kent-federated-client* branch, when it should specify *havana-moonshot-client* instead.
Source of the gap	OpenStack/Moonshot
Expected solution	The installation guide for the client side should be updated to correct this mistake.
Severity	Low
Relevance	High
Status	Pending

GAP-SU-05	
Summary	**Swift volume is not mounted after rejoining the stack**
Description	When the stack is rejoined after a reboot, the Swift volume is not mounted automatically. Therefore, you should do that manually by executing: `sudo mount -t xfs -o loop,noatime,nodiratime,nobarrier /opt/stack/data/swift/drives/images/swift.img /opt/stack/data/swift/drives/sdb1`
Source of the gap	OpenStack
Expected solution	The *rejoin_stack* script should be updated so that the swift volume is mounted when needed.
Severity	Low
Relevance	Low
Status	Queued

GAP-SU-06	
Summary	**Client installation guide does not specify how to create a container to be used by the user**
Description	In order to operate with Swift, a container needs to be created. It can be done by executing the following in the client: `swift -F -A http://155.54.95.75:5000/v3 post name_of_container`
Source of the gap	OpenStack/Moonshot
Expected solution	The installation guide for the client side should be update to include this information.
Severity	Low
Relevance	Medium

Status	Pending

GAP-SU-07	
Summary	**SAML assertion should be generated dynamically for each federated authentication**
Description	The current software assumes the *SAML assertion* to be generated statically as a copy from a template assertion hardcoded on the FreeRADIUS configuration files. However, this is not realistic, especially on federated environments, as assertions should be generated dynamically, reflecting the specific attributes of the end user being authenticated. Moreover, dedicated SAML-based IdP (e.g. Shibboleth or SimpleSAMLphp) usually generates SAML assertions, so that the RADIUS server should contact them to obtain the assertion rather than generating it by itself. Without this ability, federated authorization is neither flexible nor realistic.
Source of the gap	Moonshot
Expected solution	The Moonshot solution should be updated to allow the dynamic generation of the SAML assertion, either by its generation on the RADIUS server or by its retrieval from an external IdP.
Severity	High
Relevance	Low
Status	Queued

GAP-SU-08	
Summary	**Current solution only works on Linux**
Description	The current software and installation guide only cover the deployment of the federated authentication and authorization mechanism for OpenStack on Linux. Specifically, the client and the server should be installed on specific versions of Ubuntu, and may not work on others. Many organizations run their services on other platforms (such as Microsoft Windows). It would be interesting to allow this federated cloud solution on those systems as well.
Source of the gap	OpenStack/Moonshot and Moonshot
Expected solution	The software should work on different platforms and operating systems.
Severity	Low
Relevance	Low
Status	Queued

3.1.1.2 *Execution*

This subsection lists the gaps we have found when trying to execute the OpenStack client and server with the support for Moonshot-based federated authentication.

GAP-UE-01	
Summary	**Only Swift client is supported**
Description	On the server side, the integration with Moonshot has been implemented on the *keystone* module. In principle, this will allow the use of federated access to any OpenStack service that uses keystone as its means of authentication. However, on the client side, the integration code has only been implemented on the Swift service client. This limits the usability of the solution, as OpenStack defines a lot of different services and has different clients for each service. Consequently only the Swift service can currently be used. One of these omitted services is *Horizon*, the dashboard of OpenStack, which is expected to be one of the most accessed by the end users and administrators.
Source of the gap	OpenStack/Moonshot
Expected solution	As the OpenStack community is working to integrate the entire authentication processing into two single types of client applications (one for the command line applications and the other for the web applications), this gap will be addressed if the integration code with Moonshot is included into these two client applications.
Severity	Medium
Relevance	Medium
Status	Pending

GAP-UE-02	
Summary	**OpenStack SSO support is not used**
Description	When the Swift client is used repeatedly to perform different actions over the same storage device on the same Swift server, the application does not re-utilise either the *scoped* nor *unscoped* tokens provided by the Keystone server during the authentication process. This leads to a complete execution of the EAP authentication process for each action the end user wants to access any cloud service.
Source of the gap	OpenStack/Moonshot
Expected solution	The client application should be improved so that it makes use of the SSO tokens provided by Keystone. This should be done either for the Swift client, or for the integrated authentication client applications that will replace it in the future (see GAP-UE-01).
Severity	Medium
Relevance	High

Status	Pending

GAP-UE-03	
Summary	**Scoped token size is too big for the swift service**
Description	The scoped token provided by the Keystone service is too big to be accepted by the Swift service. A workaround consists on switching the token format from PKI to UUID, but this leads to a less secure solution. PKI tokens should be supported as they are expected to be the most desirable alternative for most deployments.
Source of the gap	OpenStack
Expected solution	The Swift server should be updated so it allows receiving scoped tokens of a big size.
Severity	High
Relevance	High
Status	Queued

GAP-UE-04	
Summary	**Client does not receive proper information when authentication or authorization fails**
Description	When the end user tries to access the service, and the authentication or the authorization process fails (e.g. password incorrect, invalid user, invalid assertion, not authorized...), the client application does not provide proper information of what happened. Instead, the client dumps a trace back output with requires a high knowledge of the code to understand the situation. Often it is even necessary to check the log output on the server to determine the specific problem.
Source of the gap	OpenStack/Moonshot and OpenStack
Expected solution	The client application should provide human intelligible information of the reasons behind a failed authentication process so that the end user can take the pertinent actions to solve it (e.g. try another password or contact his home domain administrator to supply the required authorization information).However, the client is limited by the error messages that Keystone returns and some of these may come from core OpenStack code rather than the federated plugin.
Severity	Medium
Relevance	High
Status	Pending

GAP-UE-05

Summary	**Swift client always prompts the end user to select the authentication method**
Description	When the end user executes the Swift client application, it always prompts her to select the preferred federated authentication method (e.g. ABFAB, SAML...). This is annoying, as for each action the end user performs over a storage volume, she has to select the method in an interactive way, degrading the quality of the user experience.
Source of the gap	OpenStack/Moonshot
Expected solution	The Swift client application should allow the user to provide a command line parameter to specify the preferred federated authentication method (e.g. -f abfab).
Severity	Medium
Relevance	High
Status	Pending

GAP-UE-06	
Summary	**Swift client always prompts the end user to select the tenant/project**
Description	After a successful authentication the Swift client application always prompts the end user to select the tenant/project value. This is annoying, as for each action the end user performs over a storage volume, she needs to introduce this in an interactive way, degrading the quality of the user experience. There exists a -T parameter that does not work.
Source of the gap	OpenStack/Moonshot
Expected solution	The Swift client application should allow using the -T parameter to select a tenant without introducing it in an interactive way.
Severity	Medium
Relevance	High
Status	Solved

GAP-UE-07	
Summary	**End users should have control over what attributes are released to the cloud service**
Description	When the end user authenticates to access a cloud service, a SAML assertion containing part of her identity information is sent from her home organization to the cloud provider. However, the end user does not have any control over which information is actually being sent. Having such a control would greatly improve the level of control the end user has over her privacy.
Source of the gap	Moonshot

Expected solution	The IdP should allow the end user to dynamically determine which attributes are to be released to OpenStack (preferred) or alternatively to define privacy policies, to determine which information can be released to which services.
Severity	Medium
Relevance	Low
Status	Queued

GAP-UE-08	
Summary	**There is no user-friendly administrative tool to allow the end user to manage how her SAML attributes are mapped**
Description	The federated Keystone plugin allows the cloud administrator to map the federated SAML attributes received during authentication into attributes that OpenStack can understand (e.g. role, tenant...). However, currently there is no way to let the end user know how this mapping is being performed, or to allow her to control how it is done.
Source of the gap	OpenStack/Moonshot
Expected solution	A user-friendly administrative tool (e.g. through a Horizon module) should be developed to allow the end user to manage attribute mapping
Severity	Low
Relevance	High
Status	Pending

3.1.2 Conclusions

After performing this gap analysis we concluded that the current state of the software (after having addressing the most critical gaps) is enough as a proof of concept, showing its potential for a federated authentication and authorization functionally on OpenStack based on the Moonshot technology.

However, the gaps found in this analysis reveal that further work is required in order to make this software actually useful for production environments. It lacks of proper installation and configuration guides, something that increases the effort required for setting it up. Besides, its applicability is reduced to a single use case and service, which makes more difficult to evaluate its potential as a means of access control for all the available could services.

3.2 Implementation of usability and management tools for OpenStack/Moonshot

This section is the result of Task 2.2, which has the purpose of fixing the gaps identified in T2.1 and documented in section 3.1. It should be noted that this gap analysis was based on the gaps derived from setting up and executing the initial OpenStack/Moonshot federated authentication and authorization plugin code developed for the Havana release by the University of Kent at the start of the CLASSe project.

Several new usability tools have been created in Task 2.2 as a result of the gap analysis. These are:

1. Adding federated access to the Keystone client. (The Keystone client, from Openstack release Icehouse onwards, is now the fundamental way for connecting all client software to the OpenStack servers. Not only is it a standalone client for connecting to the Keystone server, but it also forms the communication's library for all the other OpenStack clients);

2. Adding federated access to Horizon, the web server front end to OpenStack. (This entailed embedding the new Keystone client into Horizon (Icehouse onwards) as well as building a new federated login screen);

3. Adding the ability to manage the mapping rules and Identity Providers in the Keystone server (Icehouse onwards) via the Horizon server.

The resulting software and documentation has been published on the CLASSe web site [CLASSEWEB]. It is important to note that none of this preliminary software has been integrated into the main OpenStack branch, although discussions have been had about how this might be done. Consequently, the OpenStack community will not support this initial version of the software. .Further work would be needed to achieve this. Nevertheless, as further mentioned in Section 3.3, the core release of OpenStack (starting from Juno release) now supports the protocol independent federated identity management designed within CLASSe by KENT, and in particular, one of the protocols is ABFAB.

3.2.1 Fixing the Gaps

It is important to note that the CLASSe project is only responsible for addressing those gaps sourced in the OpenStack/Moonshot integration, and not sourced in either the OpenStack or Moonshot source codes. Nevertheless the project has used considerable effort to persuade and work with the OpenStack developers in particular to help to fix the gaps identified in their libraries.

There were no critical gaps remaining in the implementation by the end of Task 2.1.

Table 3.1 lists those gaps that were sourced as OpenStack/Moonshot and were still outstanding at the end of Task 2.1, i.e. those with a status of Pending or Queued. As Table 3.1 shows, all these gaps have now been successfully filled.

Gap Identifier	Summary	Resolution
GAP-SU-03	Initial installation guide for the server is incomplete and contains several errors	Documentation revised [GUIDEHAVANASERVER] and published on the documents page of the CLASSe web site [CLASSEWEBDOC]
GAP-SU-04	Client repository is incorrect in the installation guide	Fixed in the first public release of the documentation [GUIDEHAVANACLIENT] on the CLASSe web site [CLASSEWEBDOC]
GAP-SU-06	Client installation guide does not specify how to create a container to be used by the user	This is a feature of using the Swift client after installation, and now it is fully documented in it, so no action needs to be taken in the client installation guide.
GAP-SU-08	Current solution only works on Linux	The solution works for OpenStack and Python and therefore should work on any platform for which the Moonshot code is available. At the time of testing, the most stable version was Linux, so this is the version we tested. Now that the Horizon code has federated access added to it, then users will be able to use any browsers for which the Moonshot plugin is available in order to access federated OpenStack.
GAP-UE-01	Only Swift client is supported	The latest release of OpenStack (Icehouse) uses the Keystone client to access OpenStack, and all other clients use this. The Keystone client now supports federated login and this is described in [FEDKEYSTONECLIENT] and [GUIDEICEHOUSECLIENT].
GAP-UE-02	OpenStack SSO support is not used	Federated SSO is now supported in the latest Keystone client.
GAP-UE-04	Client does not receive proper information when authentication or authorization fails	Improved error diagnostics have been built into the Keystone client with a new Federation class of errors being added; see [FEDKEYSTONECLIENT] for a full description.
GAP-UE-05	Swift client always prompts the end user to select the authentication method	This is related to UE-02 and the lack of support for SSO. Adding support for SSO fixes this gap as well.
GAP-UE-08	There is no user-friendly administrative tool to allow the end user to manage how her SAML attributes are mapped	An attribute mapping GUI, using a web browser and Horizon, has now been developed for the Icehouse release [GUIDEJUNOSERVER]

Table 3.1: Gaps sourced in the OpenStack/Moonshot implementation

The status of the gaps that were identified as originating from either the OpenStack or Moonshot code is given in Table 3.2 below. As can be seen, it is not known when most of these will be fixed.

Gap Identifier	Summary	Resolution
GAP-SU-05	Swift volume is not mounted after rejoining the stack	The user can execute his own mount command
GAP-SU-07	SAML assertion should be generated dynamically for each federated authentication	Still awaiting update from the Moonshot development team regarding SAML generation component for the IdP
GAP-SU-08	Current solution only works on Linux	Implementing a web interface will fix this, providing ABFAB plugins are available for all browsers
GAP-UE-03	Scoped token size is too big for the swift service	Modifications in progress for the next release to allow tokens to be requested without a service catalogue, and the service catalogue retrieved in a separate transaction to

		reduce token size when using PKI. Alternatively use UUID token provider in Keystone.

Table 3.2: Gaps sourced in the OpenStack or Moonshot libraries

3.2.2 Conclusion

The gaps that were identified as originating from the initial OpenStack/Moonshot (Grizzly) implementation have been fixed, and three new usability tools/interfaces have been released to the public as a result, namely:

- A modified Horizon which allows a web browser to undertake federated login to Keystone (Icehouse release)
- New administrative pages to Horizon that allow mapping rules and Identity Providers to be managed in Keystone (Icehouse release)
- A modified Keystone command line client (Icehouse release)

3.3 OpenStack/Moonshot software integration

The original protocol independent federated OpenStack developed by the University of Kent for the Grizzly and Havana releases, before the start of the CLASSe project, is described here [JGC]. Kent modified Keystone in two ways. Firstly they developed a set of protocol independent trust management functions, and secondly they developed a set of protocol handling modules that returned identity information in a standard way. When this implementation was taken to the OpenStack Keystone group for incorporation into the OpenStack core release, members of this group argued that they did not want to burden Keystone with protocol handling code, as this would place a significant maintenance burden on the group. Rather they preferred to use federated Keystone as a back end service of Apache (see Figure 3.1), and use Apache plugins to do the federation protocol handling e.g. using mod_shib, mod_mellon, mod_oidc, mod_auth_kerb, etc. In this way, the Keystone group would not need to maintain any federation protocol handling code. This caused the CLASSe project an immediate short-term problem, as there was no ABFAB plugin module available for Apache. Project Moonshot was developing one, but it was not available at the start of the CLASSe project. Nevertheless we took the high-risk strategy of assuming the Apache plugin would be available before the end of the CLASSe project, and that it would work with Keystone in a similar way to mod_shib. In the end, this decision turned out to be the correct one, as the Moonshot plugin did become available about 9 months into the project, and we did effectively use it in the end deliverables.

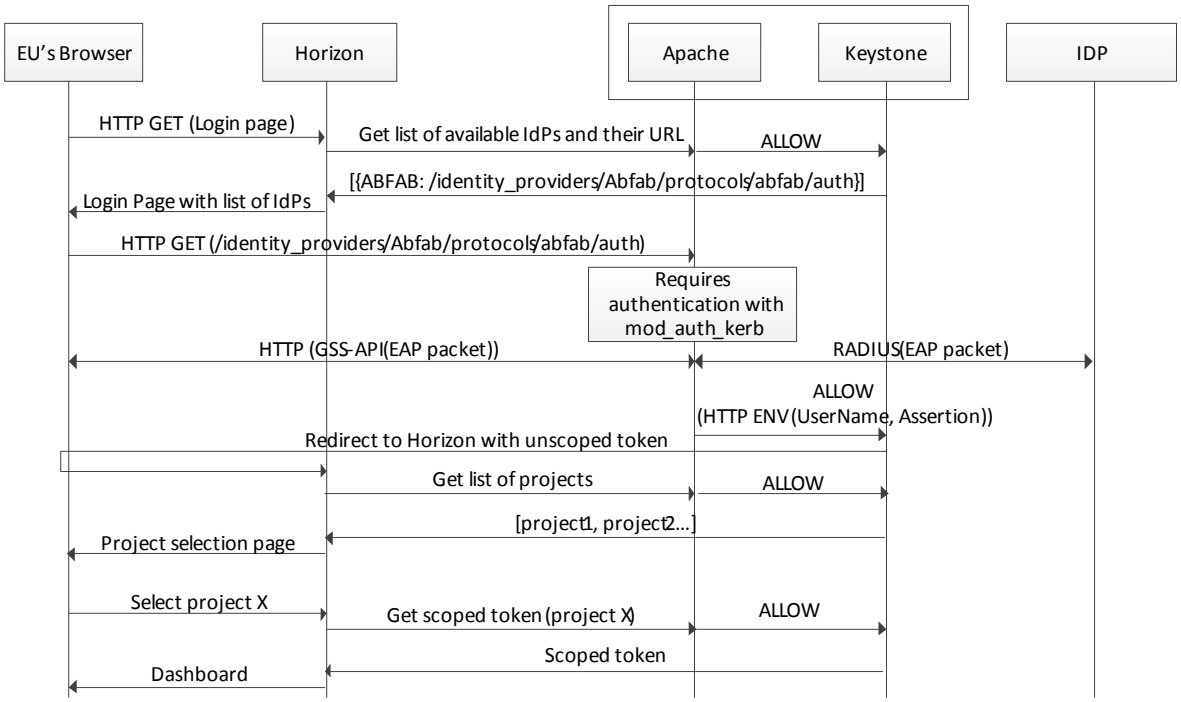

Figure 3.1: Horizon ABFAB/Moonshot Integration

Concerning the trust management functions, the Keystone group decided to adopt 2 of the 3 functions originally implemented by Kent, namely: trusted IdPs, and attribute mapping, albeit in a slightly different way. They did not adopt the trusted attribute functionality. Furthermore, due to the fact that OpenStack has 6-month release cycles, this means that only 3 months (approximately) are available for the design of any new functions, and 3 months for their implementation and testing. Consequently the first core release of federated Keystone (the Icehouse release, April 2014) only supported SAMLv2, and a lot of the support for this was hardcoded in. During the next release cycle (for the Juno release, Oct 2014) Kent spent a lot of effort making sure the code was protocol independent so that we could easily plug in ABFAB once the Moonshot plugin became available for Apache. We tested this during the Juno release cycle, and we are happy to say that the core release of OpenStack now supports protocol independent federated identity management, with different protocol handling being done by Apache plugins. Configuring the trusted IdPs, along with their protocols and mapping rules is simply an issue of configuring Keystone correctly, without any code changes being necessary.

As Figure 3.1 shows, a browser user accesses OpenStack via the Horizon dashboard, which retrieves the public list of trusted IdPs configured into Keystone and displays them to the user on the login page. The user selects his/her IdP and tries to access the corresponding Keystone URL. Since this is protected by Apache, Apache activates the correct federation protocol, which requires the user to send his/her authentication credentials (typically a username and password) directly to the IdP for validation (in the case of ABFAB, this is to a RADIUS Server). After the IdP validates the user's authentication credentials, it sends the user's identity attributes to Apache, formatted according to the federation protocol. Apache unpacks these and sends them to Keystone as environmental attributes, the most important of these being Remote_User, which holds the user's identifier. The mapping rules inspect the environmental attributes and map them into the configured OpenStack ones (projects, domains, roles etc.). Keystone returns an unscoped token and the user's projects to Horizon, which displays the projects to the user. The user chooses one and Horizon sends this to Keystone, which

returns a token scoped for the chosen project. From now on, the federated user can access OpenStack in exactly the same way as a locally authenticated one.

The Apache federated identity management plugins invariably return the requester's identity attributes to Keystone (see Figure 3.1) as environmental attributes of the HTTP message. Keystone picks up these identity attributes from the HTTP message, then passes them to the attribute mapping function in order to convert the user's identity into its own local attributes (roles, projects, domains). Apache is told which endpoints of Keystone to protect with which federation protocol plugin – a current limitation of Apache with the Moonshot/ABFAB plug in seems to be that it will not work with another federation protocol plugins simultaneously, but this is outside the scope of the CLASSe project to fix. Keystone is told which protocol and IdP is being used to protect its endpoints – these names are built into the endpoint URLs, which is a bad design in our opinion, since the endpoints of Keystone should not be tied to specific IdPs and protocols. However, this is the design that was chosen by the Keystone core group since the mapping rules are written for each protocol/IdP combination, and the URL is effectively the way of saying which mapping rules should be used. In Kent's original design, the trusted attributes function was used to discard all untrusted attributes from each IdP, then a common set of mapping rules was used for the remaining trusted attributes, regardless of the IdP or protocol. However we can use the current design to our advantage when using the ABFAB protocol, by calling the IDP a wildcard as in

```
http://openstack.server.com:5000/v3/OS-FEDERATION/identity_providers/.*?/protocols/abfab/auth
```

so that all ABFAB IdPs are treated in the same way and not differentiated between. Keystone uses the URL to determine which mapping rules to use, since these are specified for an IDP/protocol combination.

3.4 Test and deployment of the prototype

This section is the result of T2.3, where we have developed the corresponding installation guides and install tools that are needed so that other partners in GÉANT can easily deploy and test the prototype. Besides, we have tested the different components created in Task 2.2 and by proving that they actually work as intended.

3.4.1 Installation guides

In order to help anyone to test and install the software developed within WP2, we have created a set of detailed guides that provide step-by-step installation instructions. These guides are available in the CLASSe web page (www.um.es/classe/docs.html).

Due to the quick rate of changes introduced in the OpenStack source code (a stable release every six months), and the many discussions on how federated authentication should be implemented, these have significantly affected the authentication modules in OpenStack, and the way that federation is supported. The initial Kent prototype was implemented using the Havana release, whereas the CLASSe releases have been implemented using the Icehouse and Juno versions. For this reasons, we have developed the following set of installation guides:

- Installation Guide for Federated Keystone Server (Havana Version). This guide describes how to install a Moonshot enabled Keystone Identity Server and the supporting technologies required for this [GUIDEHAVANASERVER].

- Installation Guide for Federated Swift Client (Havana Version). This guide details how to install the Moonshot enabled Swift client under a Debian or Ubuntu Operating system [GUIDEHAVANACLIENT].

- Installation Guide for the Juno Server software. This guide details how to install the latest Juno Server software for Keystone supporting federated authentication and authorization, including the Horizon front end with Identity Provider and Attribute Mapping Management interfaces [GUIDEJUNOSERVER].

- Installation and Usage of the Federated OpenStack Client — Icehouse Version. This guide details how to install Keystone Client with WebSSO federated authentication [GUIDEICEHOUSECLIENT].

- Adding Federated Identity Management to OpenStacks's Keystone Client (Grizzly, Havana, and Icehouse versions). This describes adding federated identity management to the standalone Keystone client that is compatible with versions 2 and 3 of the Keystone Identity API. It also says how the universal client has been adapted for the latest Icehouse release. Finally, an overview of the authentication procedures for the Grizzly, Havana and Icehouse releases are presented along with a discussion of their current limitations [FEDKEYSTONECLIENT].

3.4.2 Testing the software

We have tested the latest software produced as a result of tasks T2.2. In particular, we have focused on verifying the most important gaps from T2.1 have been solved, and that the new functionality introduce in T2.2 works as expected. For the testing process we have followed this process:

1. To create two completely new VMs at the University of Murcia, one for the Keystone server and Horizon, and the other for the client software.

2. To follow the steps detailed in the installation guide for the Juno release ([GUIDEJUNOSERVER].) to create a new instance of the Keystone server (Juno version) and Horizon with support for federation using ABFAB technologies (Moonshot).

3. To connect the Keystone server to the existing FreeRADIUS server in the University of Murcia (described in section 3.1 of this document), in order to perform the Moonshot-based authentication process.

4. To follow the steps detailed in the installation guide ([GUIDEICEHOUSECLIENT]) to create a new instance of the client with support for federation with Moonshot technologies.

5. To verify that we can authenticate to access to the Horizon service (OpenStack's dashboard) using an Moonshot authentication and some of the identities we have in our local FreeRADIUS server.

The following snapshots depict the process of performing a Moonshot authentication to the Horizon service. First, the Horizon login page is presented. This page includes the option to select an ABFAB identity provider. We select it and proceed.

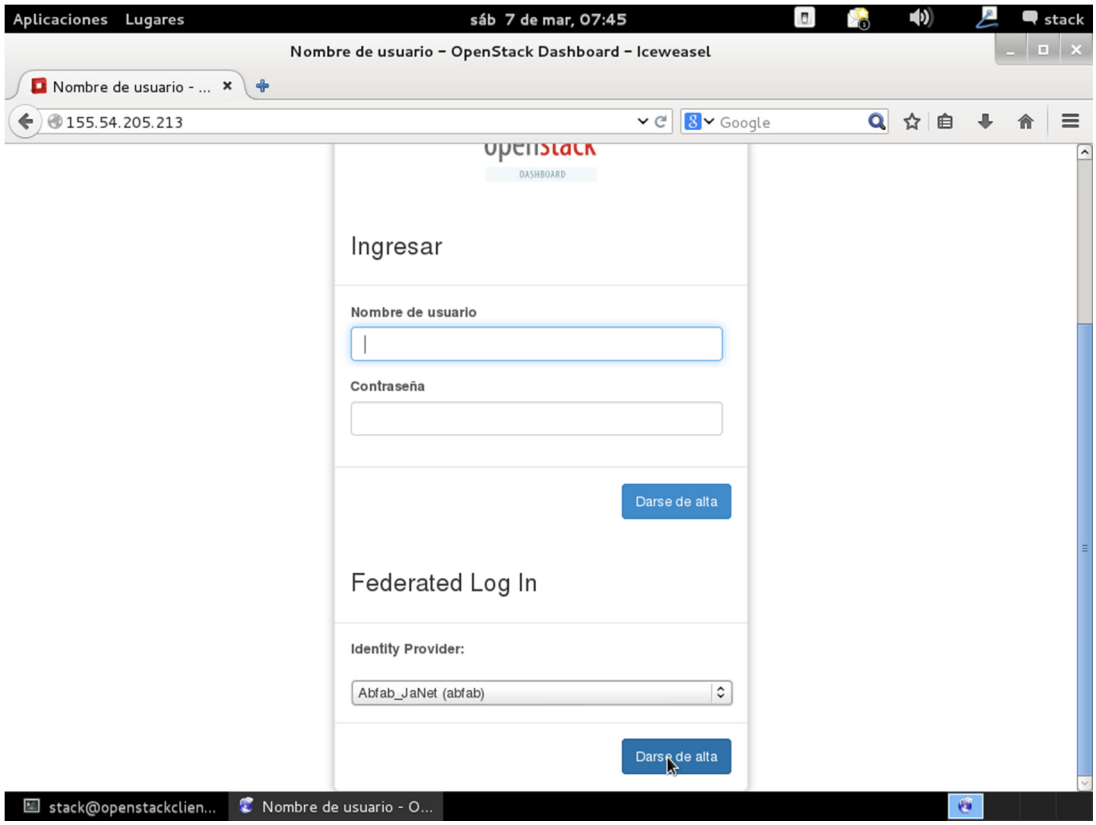

Figure 3.2: Horizon/ABFAB. Login page

At this point, the Moonshot Identity manager pops up, prompting us to select the identity we want to use. We select *alex@um.es* and start the authentication process.

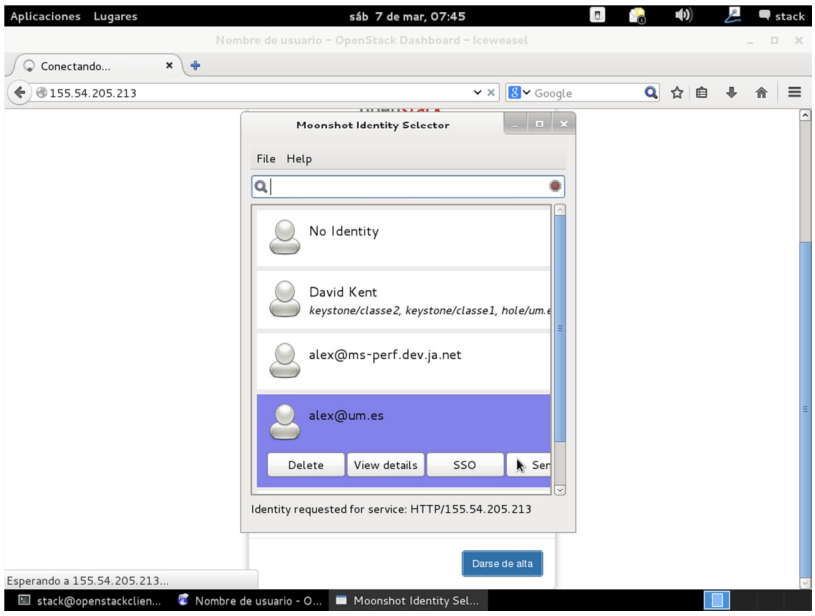

Figure 3.3: Horizon/ABFAB. Moonshot Identity selector

The authentication is performed between our browser, the Keystone, and the home IDP. When it finalizes, Horizon asks us which one of our projects we want to authenticate to.

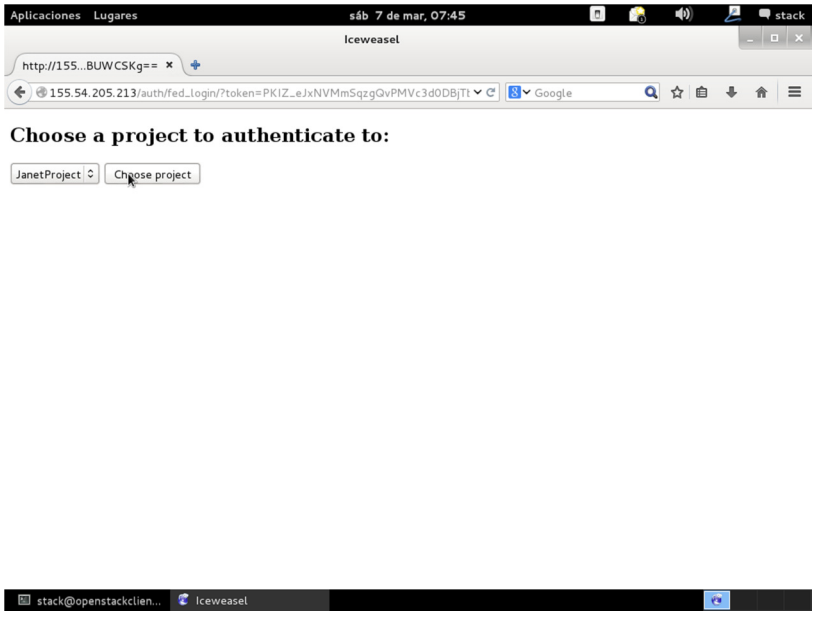

Figure 3.4: Horizon/ABFAB. Selecting project.

And finally Horizon presents us the dashboard. Note the pseudonym provided by ABFAB that protects our privacy in the visited cloud service.

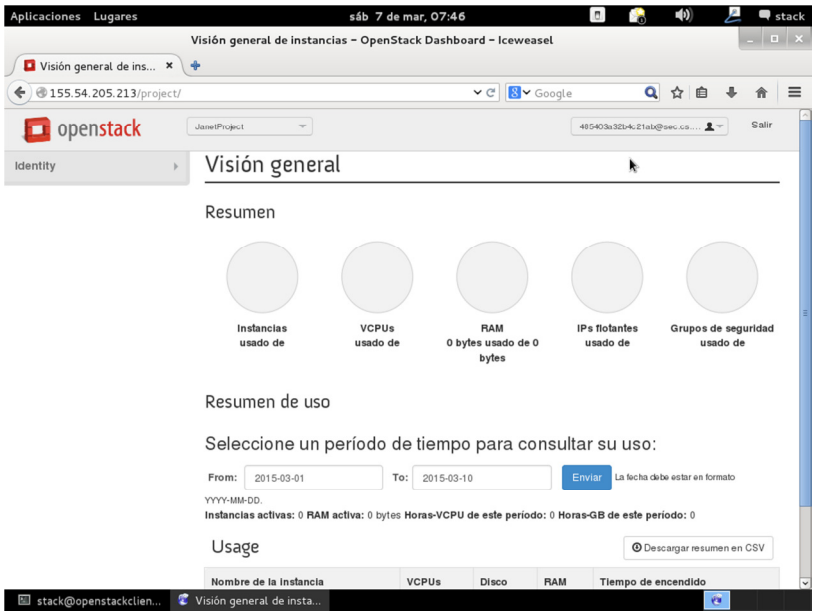

Figure 3.5: Horizon/ABFAB. Dashboard.

The conclusions extracted after this testing process are the following:

- The installation guides are very complete, clear and correct. It has been easy to follow the installation process with no mistakes or omissions. The process almost consists on copying commands from the guide and pasting them into the server terminal session, with just a minor number of customization to accommodate IP addresses and keys.

- The software works as expected. We were able to access to Horizon by performing an ABFAB authentication process, and Horizon showed our id correctly. We obtained access to the tenant and role defined in the configured mapping rule for that identity provider.

- The installation process is not trivial. It requires the system administrator to perform a considerable amount of steps before completing the process. Nevertheless, that is somewhat expected when dealing with proof-of-concept implementations, which typically require a number of small hacks to make them work before their base code is accepted into the mainline branch.

We therefore consider the software and the installation guides are ready to be used by the partners of the GÉANT community that are willing to test this software. Besides, we will provide them with support for performing the installation and configuration process if needed.

4 WP3: Research on Federated Single Sign On (SSO) for Cloud Services

In this project we distinguish between two different approaches to implement SSO: *traditional SSO* and *real SSO*. In the context of this project, the term *traditional SSO* refers to those mechanisms that minimize the involvement of the end user in the authentication process, giving her the impression that she only performed the authentication process once at the beginning of the session. However, what the mechanism actually does is to "remember" which credentials should be used for each individual service and provide them in an automatic way on behalf of the end user. The main drawback of *traditional SSO* solutions is that they are just focused on improving the user experience, not the actual performance of the procedure.

A way to improve the performance of *traditional SSO* is implementing what we have denominated *real SSO*. This kind of SSO is based on the provision of some sort of security token to the end user. This token can be used later on to re-authenticate with the authentication server in just a single round-trip. These tokens usually have a limited lifetime to avoid the harm of a theft.

4.1 Analysis and design of traditional solutions for SSO between Federated Cloud Services

This section describes the work performed in task T3.1. It starts with an analysis of how the current Moonshot's Identity Selector software handles traditional SSO in two different scenarios: when the end user moves from one cloud provider to another (*cloud-to-cloud SSO*), and when the end user accesses a cloud provider after having being authenticated to access the network (*network-to-cloud SSO*).

Then, this task provides solutions aiming to improve this traditional SSO support, either reducing the complexity of the process or increasing the quality of the user experience, or both.

A preliminary evaluation of the expected impact that these solutions might have on the overall performance of Moonshot is provided in [MS3]. That document defines a testbed, and measures the time required to complete the access to the OpenStack Swift service using Moonshot with and without the modifications. However, they

have not been included in this document as more updated results are provided in section 4.4, where they are also compared with the real SSO solution.

4.1.1 Analysis of the OpenStack/Moonshot integration for traditional SSO

This section analyses the support for traditional SSO as provided by Moonshot, and the main drawbacks that it presents. In particular, it focuses on two specific use cases, described in the DoW of the project: *cloud-to-cloud* and *network-to-cloud* [CLASSEP].

4.1.1.1 *Cloud-to-cloud*

This subsection analyzes the costs associated with the traditional SSO as provided by Moonshot, when the client moves from one cloud provider to another. In particular, it describes the process and analyzes the costs incurred when an end user (UA) from the University of Kent (*kristy@cs.kent.ac.uk*) accesses to the Swift service on a particular server (*classe1.qalab.geant.net)*, and then tries to access to the Swift service on a different machine (*classe2.qalab.geant.net)* afterwards. In this description it is assumed the end user starts with a Moonshot User Interface (UI) where no identity has still been configured.

Description of the process

The whole process can be split in the following steps:

Identity selection for classe1.qalab.geant.net

The first time the UA (*kristy@cs.kent.ac.uk*) tries to access to the Swift server (*classe1.qalab.geant.net*), the Moonshot UI prompts her to select one of her configured identities. As the end user has not any configured identity yet, she needs to add one at that precise moment. This process requires the manual intervention of the UA to introduce several parameters. Specifically, she needs to introduce a friendly-name identity, the IdP name, the username, and the password. In particular, she adds the *kristy@cs.kent.ca.uk* identity. Figure 4.1 depicts how this addition is actually done.

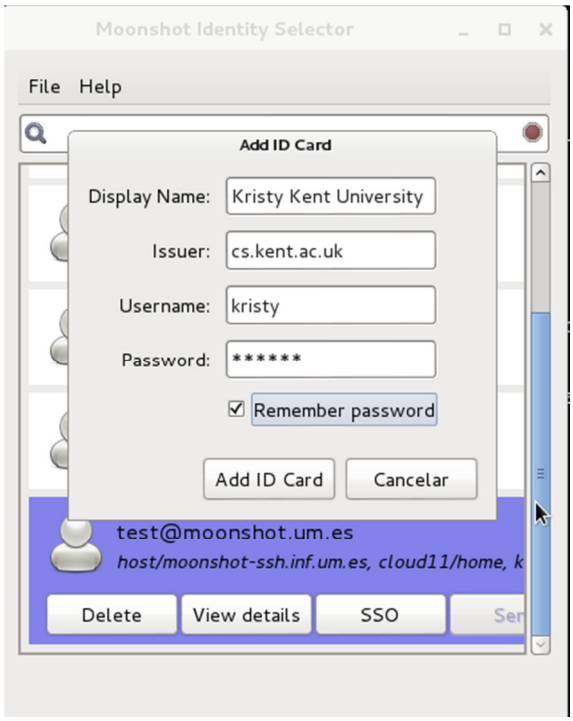

Figure 4.1: Adding an ID card to Moonshot.

Authentication and authorization for classe1.qalab.geant.net

Once the identity has been selected, the authentication and authorization processes with the Keystone server at *classe1.qalab.geant.net* begin. The UA starts the authentication process, consisting on several HTTP exchanges. They involve the exchange of GSS-API tokens containing EAP packets, as well as the distribution of different tokens [OPENSTACK] to the UA. Figure 4.2 depicts the exchange of information required to perform the federated authentication and authorization.

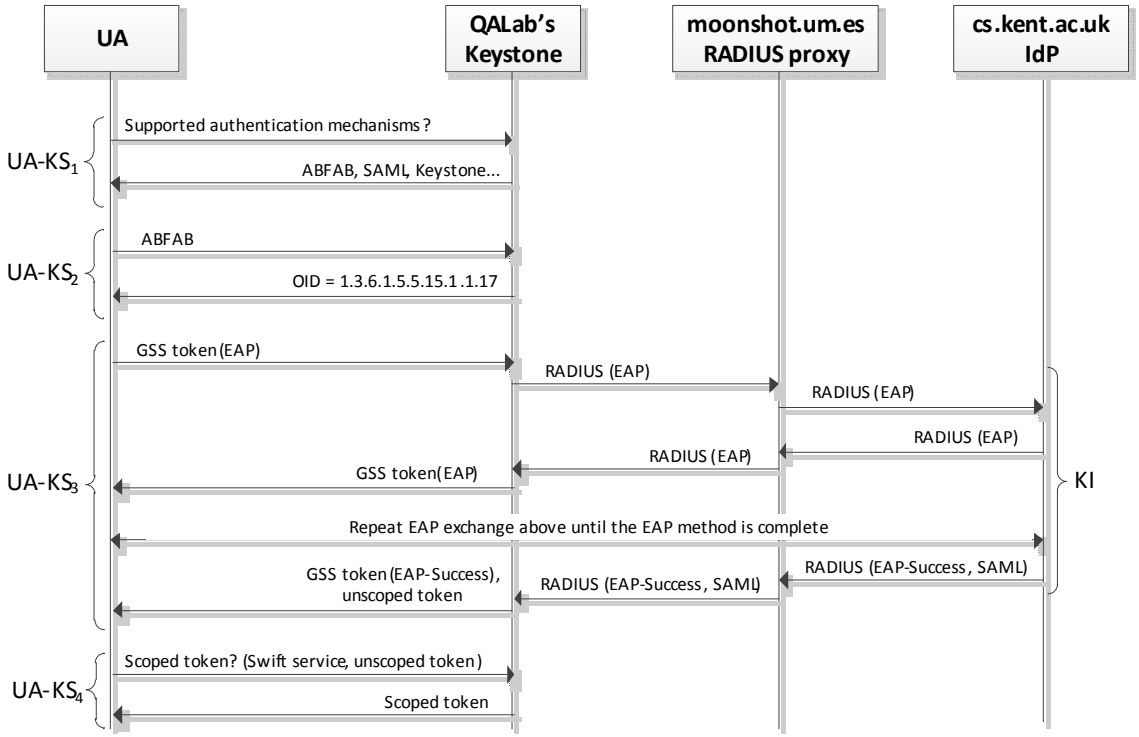

Figure 4.2: Authentication and authorization exchange

In particular, we can distinguish five different conversations. On the one hand, conversations UA-KS_X occur between the UA and the Keystone and are based on the HTTP protocol. On the other hand, conversation KS-I occurs between the Keystone and the IdP, and it is based on RADIUS. At the end of this conversation the IdP provides a SAML assertion (Figure 4.3) to the Swift service.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                ID="moonshot" IssueInstant="2011-03-19T08:30:00Z" Version="2.0">
  <saml:Conditions NotOnOrAfter="2015-03-19T08:30:00Z" />
  <saml:Issuer>urn:mace:incommon:osu.edu</saml:Issuer>
  <saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
      kristy@cs.kent.ac.uk
    </saml:NameID>
  </saml:Subject>
  <saml:AttributeStatement>
    <saml:Attribute Name="studentcard"
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
      <saml:AttributeValue>Student</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute Name="uid"
                    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
      <saml:AttributeValue>kristy@cs.kent.ac.uk</saml:AttributeValue>
```

Figure 4.3: SAML assertion.

Access to the Swift service at classe1.qalab.geant.net

After the UA-KS_4 conversation, the UA starts an HTTP conversation with the Swift server, denoted as UA-S, in order to provide it with the received scoped token, and get authorized to access the Swift service. Figure 4.4 depicts this exchange.

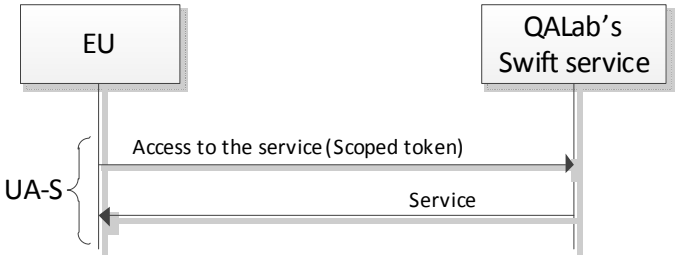

Figure 4.4: Access to the Swift service exchange

Identity selection for classe2.qalab.geant.net

When the UA tries to access *classe2.qalab.geant.net*, the identity *kristy@cs.kent.ac.uk* is already configured in the Moonshot UI. However, as the service name is unknown, the end user is prompted to select an identity to be used for the service. Although this selection process is faster than the addition of a complete new identity, it still requires the UA to take an active part of the process.

Authentication and authorization for classe2.qalab.geant.net

Moonshot bases its SSO support on the storage of associations between identities and services. However, this traditional SSO does not prevent nor modifies any of the exchanges of data that are required to perform access control. For this reason, this second authentication and authorization process will be identical in terms of number of messages, amount of data exchanged, and time spent to the first one.

Access to the Swift service at classe2.qalab.geant.net

As for the authentication and authorization process, the access is identical to the one performed for the classe1.qalab.geant.net server.

Analysis of the process

As described above, the costs associated to the authentication are repeated every time the end user accesses a Swift cloud service; regardless the end user was or not previously authenticated. Besides, whenever the end user accesses to the Swift service for the first time, she needs to manually select the identity to be used in the Moonshot UI. Moreover, if a suitable identity is not listed, it has to be added at that moment. Two major conclusions can be extracted:

- The EAP authentication the most expensive part of the access control process, as it requires a high number of message exchanges.

- The identity selection process requires additional time, as the UA needs to iterate through the identity list to determine which one of them is the most suitable to be used. Besides, it may impact the user since it interrupts the automatic nature of the access control process.

4.1.1.2 *Network-to-cloud*

This subsection analyzes the costs associated to the traditional SSO as provided by Moonshot, when the client is required to perform an authenticate access to the network, and then she tries to access to a cloud provider. In particular, it describes the process and analyzes the costs incurred when an end user (UA) from the University of Kent (*kristy@cs.kent.ac.uk*) accesses to the wireless network access service provided at the University of Murcia, and then she tries to access to the Swift service on *classe1.qalab.geant.net* afterwards. In this description it is assumed the end user knows how to connect to the network and has a configured 802.11 supplicant.

Description of the process

The whole process can be split in the following steps:

Identity selection for accessing the network

This step is out of the scope of this project, and it is performed using the standard mechanisms envisioned for the access to the network (e.g. using a network supplicant).

Authentication and authorization for the network

Once the identity has been selected, the authentication and authorization processes with the AP begin. The UA starts the authentication process for the network services, which implies several 802.11 exchanges. They involve the exchange of EAP packets and the execution of the 4-way handshake. This is a standard access to the network, without any kind of change to the process.

Identity selection for accessing cloud1.qalab.geant.net

Although the UA has configured an identity for accessing the network, this selection does not apply for Moonshot. Therefore, this step is identical to the one described for the first access in the cloud-to-cloud scenario.

Authentication and authorization for cloud1.qalab.geant.net

This step is identical to the one described for the first access in the cloud-to-cloud scenario.

Access to the Swift service at classe1.qalab.geant.net

This step is identical to the one described for the first access in the cloud-to-cloud scenario.

Analysis of the process

As it can be observed, the costs of accessing the cloud server are identical to those described for the cloud-to-cloud scenario.

4.1.2 Design of improvements for the traditional SSO

In order to improve the traditional SSO provided by the OpenStack/Moonshot integration, we have decided to focus on the two major costs that have been detected during the analysis: EAP authentication and identity selection. In particular, this section provides the design of different approaches aiming to reduce the total amount of time required to perform cloud-to-cloud and network-to-cloud traditional SSO. These approaches focus on:

- **Reducing the number of times the user needs to add or select her identity from the Moonshot ID selector.** It consists on implementing some kind of heuristic that enables taking the decision on what identity to use for the next access in an automatic way. In this way, the end user is less disrupted and the user experience is improved.

- **Reducing the time required to perform EAP authentication**. As many current IdPs use TLS-based EAP methods (i.e. EAP-TTLS or PEAP), a great possibility to reduce the complexity of the EAP authentication comes from the use of TLS-resumption [TLS]. TLS-resumption allows re-using the cryptographic material derived for a previous TLS session to establish a new TLS channel with the same server. The establishment of this second session uses the previous session key as PSK, avoiding the requirement of exchanging certificates and of performing an expensive Diffie-Hellman [DH] exchange. Besides, when applied to the EAP-TTLS and PEAP methods, it avoids the execution of the phase 2 [EAP-TTLS], as the user is directly considered authenticated. TLS-resumption is only possible when the previous TLS session has not yet been expired. In this way, the use of TLS-resumption has a great benefit on the number of round-trips required to perform the EAP authentication. As these round-trips are performed inter-domain, this reduction will also have a high impact on the overall authentication time. Besides, the computational complexity of the method is also reduced, as there is no need to use asymmetric cryptography. This is something much appreciable when using low capacity devices, such as mobile devices, tablets, sensors, etc.

Following sections provide concrete design and implementations of these approaches, for each one of the use cases.

4.1.2.1 *Cloud-to-cloud*

Reducing the number of times the user needs to add or select her identity

Whenever the UA requests access to a particular service, the Moonshot software asks the Moonshot UI to provide the identity to be used. Indeed, the standard Moonshot UI process to select a suitable identity is as follows:

1. It checks whether a previous association for that service already exists. In such a case, it returns the associated identity and exists.

2. If an association does not exist, then the UI checks whether the requested service name matches any of the rules the existing identities. If so, return the matching identity and associate it with the requested service.

3. Otherwise, prompt the UA to manually select or add an identity, and associate the selected identity with the requested service.

For this scenario, we have decided to modify the way the Moonshot UI selects the identity to be used for a particular service. In particular, the proposal consists on adding a new field to the Moonshot UI's identity database. This field, called "SSO", indicates whether the UA desires that particular identity to be selected when there is neither association nor rule that matches with the requested service. This field can only be set to TRUE for one of the identities on the Moonshot UI. Therefore, a new step is added between steps 2 and 3.

- If none of the rules match the requested service, then check whether there is an identity marked as SSO. If so, it returns the marked identity and associates it with the requested service.

Besides defining how this field is used, it has been also required to define how it is set to TRUE, in order to mark an identity as the one to be used for SSO operation. In particular, we have implemented the following heuristic:

- Whenever an identity is selected by any of the 4 steps described above (original 3 plus the new one), that identity is marked as SSO, unmarking the one previously marked (if any).

In this way, following the same example as in 4.1.1.1, when the UA accesses to *classe1.qalab.geant.net,* the identity *kristy@cs.kent.ac.uk* is marked as SSO. Then, when trying to access to *classe2.qalab.geant.net,* the same identity is selected without prompting the UA.

Besides, we have modified the UI implementation in order to indicate which identity is currently selected for SSO. This identity also includes an indicative suffix "*(SSO enabled)*". Moreover, we have added a new toggle button that represents the SSO status of each identity. In this way, the UA can easily remove the SSO field of a specific identity, or set it to a different one.

Although we have selected the indicated heuristic, others are possible. For example, one may think that automatic selection based on the last used identity would incur into a lot of erroneous selections. Instead, one could only rely on the manual selection of the SSO identity, falling back into the same SSO identity (e.g. the working identity) for every new service we try to access, regardless which identity was used for the last access.

Figure 4.5 depicts the modified Moonshot UI, the suffix and the SSO toggle button indicating that *kristy@cs.kent.ac.uk* is marked as SSO.

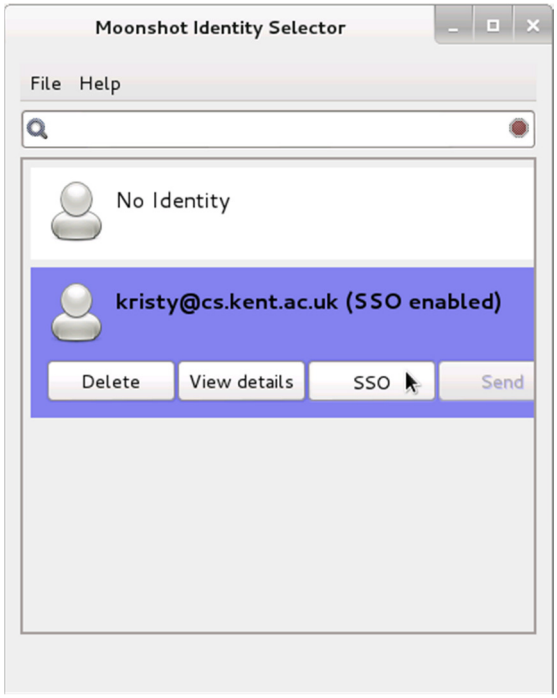

Figure 4.5: Moonshot UI and SSO indication.

Reducing the time required to perform EAP authentication

FreeRADIUS supports TLS resumption for the EAP-TTLS and PEAP methods. All it needs to be done is to set the *eap→tls→cache→enable* variable to *true* in the *eap.conf* configuration file.

On the end user's side, Moonshot's implementation ships an EAP library called *libeap*. This library is based on *wpa_supplicant's* code, which supports TLS-resumption for 802.11 connections. However, this implementation has two issues that preclude its usage in Moonshot:

1. wpa_supplicant is only able to resume TLS connections within a single execution of the application, that is, TLS session information is stored in memory, but it disappears when the program is finalized. However, the Moonshot UI is not a daemon, but a single process that is executed and finalized each time the end user tries to access an application service. Therefore, a persistent storage for TLS sessions is required to make them available to be used in subsequent executions.

2. TLS resumption functionality in wpa_supplicant is activated by EAPOL [802.1X], whenever it determines there is a re-authentication in place. However, EAPOL code has been pruned from *libeap*, as it makes no sense in Moonshot for application authentication. Therefore, it is required that the GSS-EAP layer forces TLS resumption whenever there is a TLS session available to be used.

Therefore, for this scenario we have decided to extend *libeap* functionality to support persistent storage of TLS sessions (1), and the correct activation of this support (2). The authentication process would be as follows:

1. When the EAP-TTLS or PEAP methods are started, *libeap* looks for a persistent TLS session file for the current end user's identifier. TLS sessions are stored in a file in the /tmp folder (this is just for proof-

of-concept purposes; see section 4.1.2.3 for security considerations). The filename has the following structure: *ssl-session-{ID}*, where {ID} is the end user's identifier (e.g. ssl-session-kristy@cs.kent.ac.uk). TLS session duration is established by the TLS server (i.e. IdP) and, if the server considers the session is expired, the process will continue as a regular TLS connection.

2. If the file is found, the session is imported into the current TLS object, and the TLS resumption functionality is executed.

3. Regardless the session was imported or not, when the TLS session is considered established, it is stored into a persistent file in the /tmp folder, following the same naming rules as explained before. This will make it available for future executions.

This functionality has been implemented for both, EAP-TTLS and PEAP, when using the OpenSSL cryptographic backend (the one used by default).

The following snapshots depict the log output of the *gss-client* application and Moonshot. Figure 4.6 shows the log output for a first authentication where not TLS session file is found (1) and therefore the X.509 public key certificate exchange is required (2).

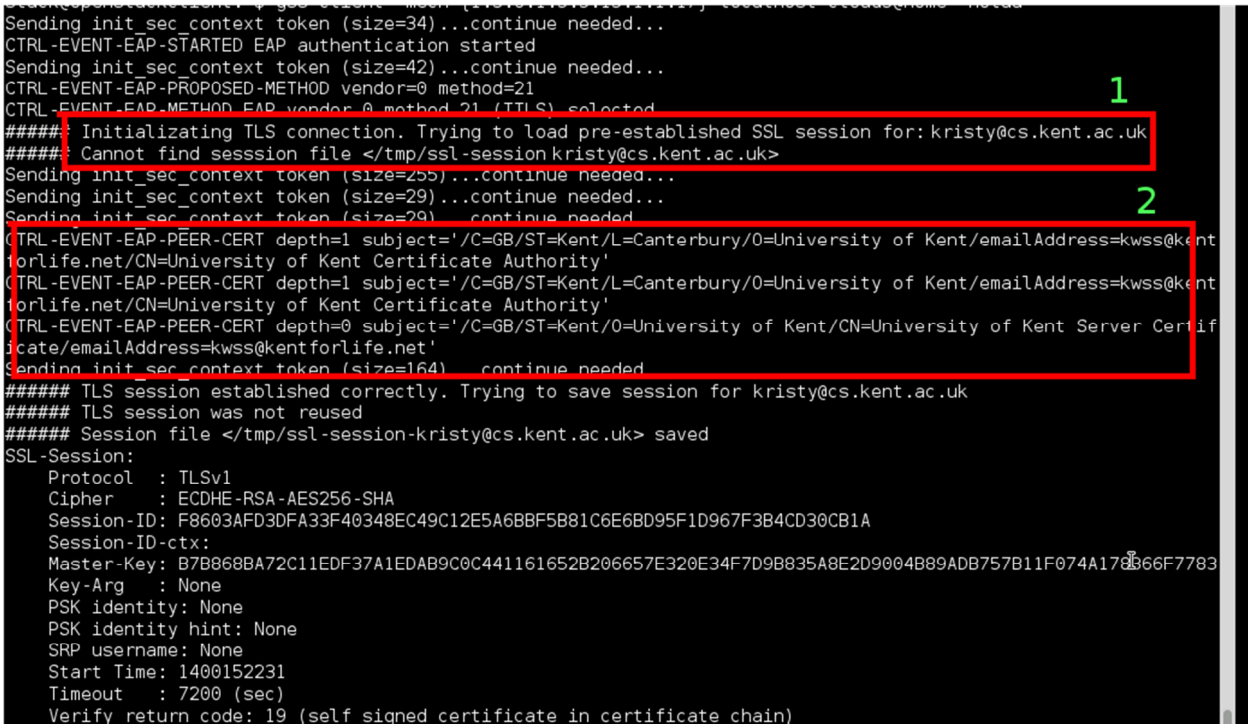

Figure 4.6: Moonshot authentication without TLS resumption

Figure 4.7 shows the log output of a second authentication, where the previous TLS session is found and used. It can be observed how, among other things, it does not require the exchange of X.509 certificates.

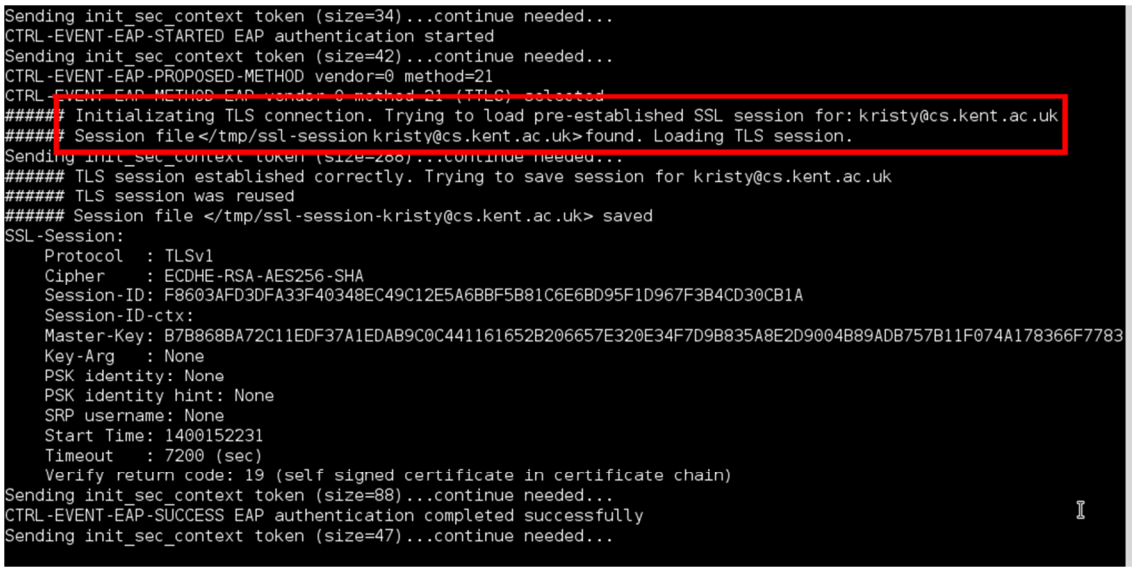

Figure 4.7: Moonshot authentication with TLS resumption

Following the examples provided in section 4.1.1, the use of TLS-resumption would have impact in the *Authentication and authorization for classe2.qalab.geant.net* step. In particular, it affects UA-KS$_3$ and KS-I conversations.

4.1.2.2 *Network-to-cloud*

Reducing the number of times the user needs to add or select her identity

The Moonshot UI provides a way to add identities from the command line, using an XML description file and the *moonshot-webp* [MOONSHOT] tool. Therefore, our approach was to modify the 802.11 supplicant to, upon a successful authentication on the network using an identity, generating an XML file describing that identity and introducing it into the Moonshot UI configuration. Besides, this identity would be marked as SSO, in such a way that during subsequent accesses to cloud services, the UI would not prompt the end user to select one.

In particular, the most used supplicant in GNU/Linux nowadays is *wpa_supplicant*. The problem with performing this functionality on this program is that it is executed as root user, whereas the *moonshot-webp* application should be executed as the user running the desktop (not necessarily the root user). Although possible, this was a bothersome process.

Therefore, we have decided to take advantage of the widespread use of the NetworkManager software [NM]. This software provides a DBUS interface [NM-DBUS] that allows controlling its behavior, as well as being notified of certain aspects, including the establishment of new connections. In particular, we have created a python script that monitors NetworkManager's events. Whenever the UA connects to a new network, it looks if there are 802.1X credentials configured (see section 4.1.2.3 for some considerations on this). If so, an XML file describing the identity is created and introduced in the Moonshot UI. This ID card is represented in Figure 4.8 (for the *kristy@cs.kent.ac.uk* identity):

```
<identities>
        <identity>
                <display-name>Auto N2C cs.kent.ac.uk </display-name>
                <user>kristy</user>
                <password>kristyspassword</password>
                <realm>cs.kent.ac.uk</realm>
        </identity>
</identities>
```

Figure 4.8: Moonshot's ID card XML description.

If the identity card already exists in the Moonshot UI, it is modified to mark it is as SSO.

In order to make this process working flawlessly, we also needed to modify the *moonshot-webp* application to avoid it prompting the UA to ask whether she wanted to add/modify the ID card. We disabled this dialog and assumed a YES response (see consideration in section 4.1.2.3). We also modified the Moonshot UI to automatically mark any new identities received via *moonshot-webp* as SSO.

Reducing the time required to perform EAP authentication

As described for the cloud-to-cloud scenario, Moonshot library has been extended to load a previous TLS session associated with an identifier whenever a file with a specific name exists on the /tmp folder.

As Moonshot EAP source code is based on wpa_supplicant's one, the modifications carried out in Moonshot to export the TLS session to a file should be easily applied to wpa_supplicant's code. In this way, when Moonshot is executed, the TLS session file would be found. As it has been commented before, this file-based storage of the session is only for proof-of-concept purposes. Secure keyrings or encrypted files should be used instead.

Although this modification to wpa_supplicant has not been implemented, it can be easily extrapolated that a subsequent access to a Cloud service using Moonshot will be identical to the one described for the TLS-resumption in the Cloud-to-Cloud scenario, as the session file will already be there when the Moonshot process is executed.

4.1.2.3 *Considerations*

1. Storing TLS session in plaintext files is far from being a secure alternative. Although this option has been implemented for simplicity, in production implementations secure storage such as Gnome-Keyring should be used instead.

2. The Moonshot identity selector does not behave as a full-blown authentication agent. Instead, it just limits itself to provide the credentials to be used with the GSS-EAP mechanism. In this way, the Moonshot UI is called whenever a GSS-EAP authentication starts, but it is closed right after the identity has been selected. Therefore, the Moonshot UI is unable to receive any kind of authentication feedback. Whether the authentication is successful or not is something that must be handled by the application service itself. However, most of them would need modifications to do this in a user-friendly

way. This would be required to allow the UA to know what happened; and try to solve the issue (e.g. updating password, disabling SSO, removing previous associations, etc.).

An alternative would be integrating part of the Moonshot UI's functionality into the GSS-EAP mechanism implementation. In this way, the mechanism would be able to try to authenticate first with the identity marked as SSO and, if the authentication process fails, it would be able to disable SSO and prompt the UA to manually select the proper identity to be used. Besides, it would also be able to provide the UA with richer information in case of authentication failure. However, this integration would require a big re-writing of the code, being out of the scope of this project.

3. The NetworkManager DBUS interface does not provide the end user 802.1x credentials if the *remember password* checkbox has not been set on the UI. This is indeed the expected behavior, as if the UA did not want the password to be remembered for subsequent accesses to the network, it is reasonable to consider that she does not want it to be automatically used for anything else.

4. The moonshot-webp executable [MOONSHOT] should include a command-line option to determine whether the adding/modification should be done quietly or not.

4.1.3 Conclusions

Current support for traditional SSO in Moonshot is reduced to offer an identity manager that remembers passwords and associations between services and identities. This implies that every time the end user accesses to any cloud service, a complete EAP authentication method is executed. Besides, the UA needs to manually select an identity whenever an association for the current service does not exist, impacting the user experience.

This task has analysed the traditional SSO as provided by Moonshot, and has designed approaches to improve Moonshot's SSO handling for both, cloud-to-cloud and network-to-cloud scenarios. These approaches focus on the one hand on the reduction of the involvement of the UA by the use of heuristics to select the identity to be used for unknown services, and on the other hand, on the reduction of the amount of data exchanged with the home domain, by the use of TLS-resumption techniques.

4.2 Analysis and design solutions for real SSO in Cloud Services between federated Cloud services: cloud-to-cloud and network to cloud

One way to improve the performance of *traditional SSO* is implementing what we have denominated *real SSO*. This kind of SSO is based on the provision of some sort of security token to the end user. This token can be used later on to re-authenticate with the authentication server in just a single round-trip. These tokens use to have a limited lifetime to avoid the harm of a theft.

In this context, Task 3.2: *Analysis and design of solutions for real SSO in Cloud Services between federated Cloud services: cloud-to-cloud and network to cloud*, aims to analyse and design solutions that allow an end user to obtain such a token after an initial ABFAB-based access to a cloud service, in order to use it to access further cloud services afterwards, without the need of executing a full ABFAB authentication again. Besides, this task also aims to provide a solution to allow the end user to obtain such a token after an initial access to the network service.

Therefore, during this task we have, on the first place, made a thorough analysis of the problem, the associated technologies, and the applicability scenarios. Then, taking all this information into account, we have come to the conclusion that, since ABFAB authentication is based on EAP, the most natural and appropriate solution is to apply the standard mechanism defined in the IETF to reduce the number of round-trips in EAP: the *EAP Extensions for the EAP Re-authentication Protocol (ERP)* [ERP]. Extending the GSS-EAP [GSS-EAP] mechanism to provide support for the ERP extensions would provide one round-trip authentications in a straightforward way, providing an improved performance and resource utilisation. Besides, we have also analysed the implications that the usage of ERP has on the authorization mechanism already defined for GSS-EAP. Additionally, we have also analysed other non ABFAB-based alternatives, such as FedKERB [CSI] and SASL-SAML-EC [SASL-SAML-EC]. They are described in Section 4.2.3.

The remainder of this section is structured as follows. Section 4.2.1 provides an overview of ERP, its features and its different exchanges, using a descriptive example scenario to motivate its utilisation. Section 4.2.2 describes the updates and modification required at the GSS-EAP layer in order to incorporate the ERP functionality. Section 4.2.3 provides the analysis of the applicability of another ABFAB-based alternative called FedKERB.

4.2.1 ERP overview

The core component of the ABFAB authentication [ABFAB] is the GSS-EAP mechanism [GSS-EAP]. This mechanism allows any application service supporting the GSS-API to perform an EAP-based authentication for any federated end user (UA), and to obtain a SAML assertion generated in the home domain, containing authorization attributes about that UA. From these two processes, the former is the one consuming most of the round-trips, as some EAP-methods based on public key cryptography may exchange more than 10 messages (e.g. EAP-TTLS [EAP-TTLS]) with the end-user's home AAA/IdP server.

Hence, the more straight and natural way to optimize the ABFAB authentication process is to target the EAP authentication. For this purpose, the IETF Handover Keying (HOKEY) WG defined the *EAP Extensions for EAP Re-authentication protocol* (ERP) [ERP], in order to avoid repeating the entire EAP exchange when the UA is moving from one EAP authenticator to another. In particular, ERP allows the UA and an AAA server (either in the home domain or in the visited domain) to mutually verify proof of possession of key material from an earlier EAP method run. In this sense, this key material (and the protected ERP message) represent the security token that must be verified to allow the access to the service. That is, by using this material the UA is able to establish a security association with the targeted service. In particular, ERP consists of a pair of new EAP packet codes allowing the derivation and verification of this shared key material.

One important aspect to notice is that EAP, as thus ERP, was originally conceived for the network access context. However, the ABFAB WG has extended the EAP applicability statement [EAP-APPLICABILITY] to

allow the use of EAP for authentication in generic applications. This fact is what enables ERP to be an interesting approach for real SSO in generic applications, as it is combined with the GSS-EAP mechanism defined by the IETF ABFAB WG. Besides, the fact that it can be used for both, network access and application access automatically makes it provide support for the two target scenarios of this task: cloud-to-cloud and network-to-cloud. Nevertheless, it is worth mentioning that nowadays ERP has not been widely deployed for network access authentication.

In order to provide an overview of how ERP works, we will use an example scenario. Let's assume that Alice (UA) has a subscription with a university, called *Home University*. This means that Alice has some long-term credentials allowing her to authenticate with this university. Let's assume there are also other universities, called *University A* and *University B*, which have different services deployed. These three universities are interconnected through an AAA infrastructure (e.g. RADIUS, trust router [TRUST-ROUTER], etc.), and the services they deploy support the use of EAP for performing end user authentication (e.g. 802.11 or GSS-EAP). Figure 4.9 depicts this scenario.

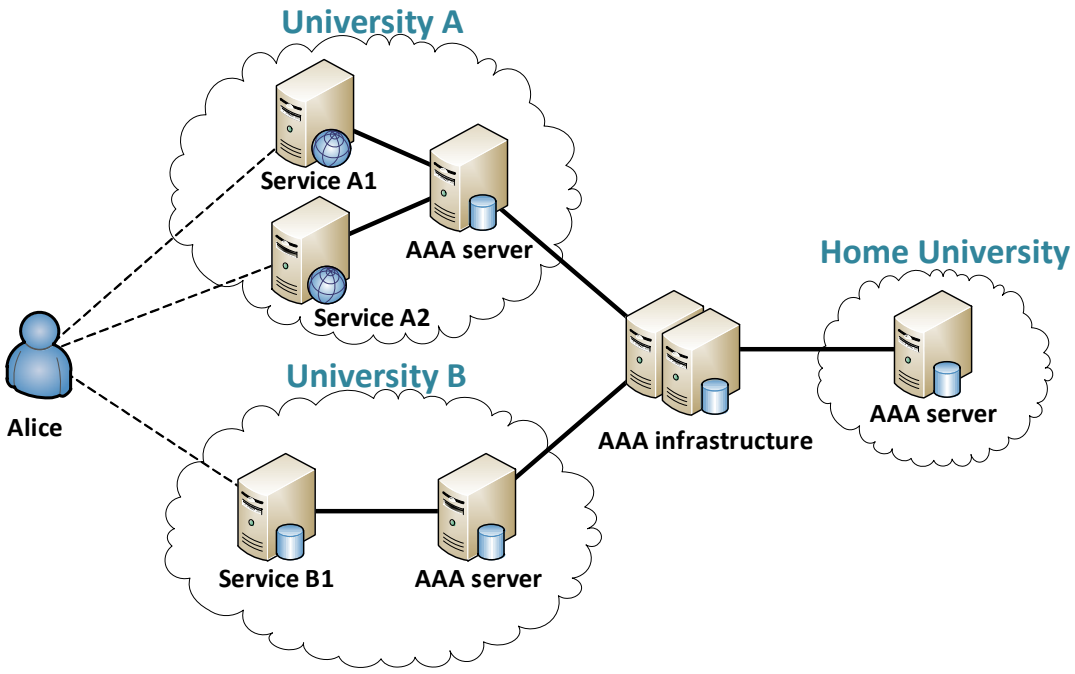

Figure 4.9: Scenario example

What Alice and the universities want is to allow her the access to any of the services deployed by the different universities without requiring her to create a subscription with each of them. This is called a federated access. Moreover, they also want to perform this process in the most efficient possible way. On the one hand, Alice desires to avoid being prompted to introduce her long-term credentials once and again, and to pass through the process as fast as possible to start enjoying the service. On the other hand, the universities want to reduce the amount of network traffic exchanged between them, as well as the computational cost of performing this process, in order to improve scalability and reduce costs. For this reason, let us assume that they all agree to

support ERP in addition to EAP for authentication. ERP defines three possible situations that can happen during an access request, that are explained following.

Let's imagine Alice wants to access a service deployed in *University A, Service A1*. To make use ERP she needs to have an ERP key established with *University A's* AAA server (ERP key_A). Since she does not have any yet, she needs to perform a process called ***ERP implicit bootstrapping***. This process consists on a standard EAP authentication process that derives an ERP key from the EAP method. The key is then delivered to the *University A's* AAA server through the AAA infrastructure, by using extensions in RADIUS [RADIUS-ERP]. In addition, Alice and *Home University* derive another ERP key (ERP key_H) that will serve to speed up future bootstrapping processes (as detailed below).

The ERP process is limited to provide the authentication functionality. That means that the execution of an authorization process is left to the GSS-EAP mechanism and the AAA infrastructure in ABFAB scenarios. As mentioned earlier, GSS-EAP already provides an authorization mechanism based on the distribution of a SAML assertion after the EAP authentication process with the home AAA server through the service. Hence, after performing any of the ERP exchanges with GSS-EAP, the application will end up with an assertion to perform the authorization. More details about this distribution as well as some considerations are provided in the next section. After the EAP authentication process, the EAP derivation keys and the authorization process, Alice gets access to Service A1 in University A.

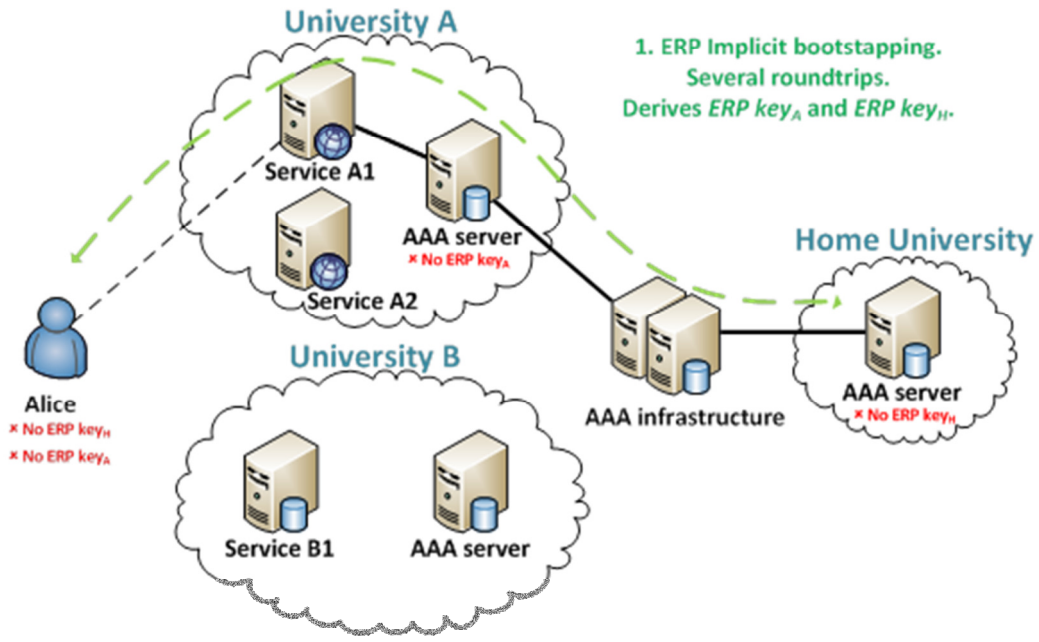

Figure 4.10: ERP implicit bootstrapping

After a while, Alice decides to access to a different service deployed on *University A, Service A2.* As she shares an ERP key with the AAA server in that university (i.e. ERP key_A), she just needs to perform a ***local ERP exchange***, which consists of a single round-trip EAP exchange that is completely performed within the visited organization.

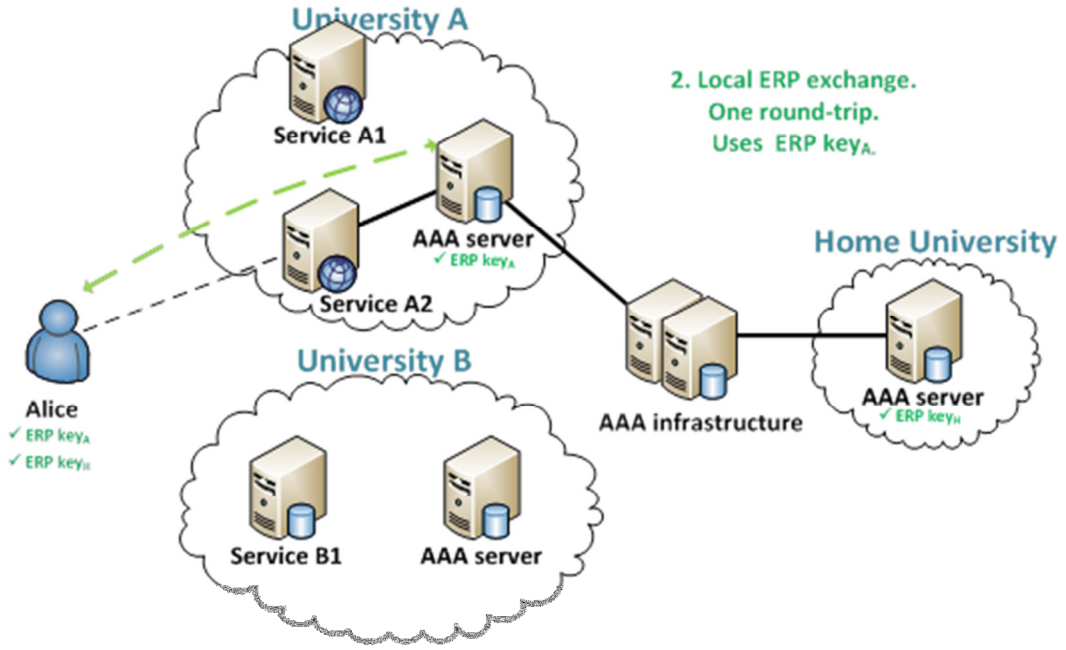

Figure 4.11: Local ERP exchange

Finally, Alice decides to access to a service deployed on *University B, Service B1*. Since she does not share any ERP key with the AAA server on *University* B (ERP Key_B), she needs to perform a bootstrapping process to derive it. Nevertheless, Alice can now use the ERP key derived during the previous implicit bootstrapping process (ERP key_H) to perform an ***ERP explicit bootstrapping*** process. This process consists of a single round-trip EAP exchange performed with the home organization. After this process, Alice can perform additional ***local ERP exchanges*** within *University B*, as explained before. Therefore, the ERP implicit bootstrapping process described above is only performed once during the whole lifetime of the EAP session.

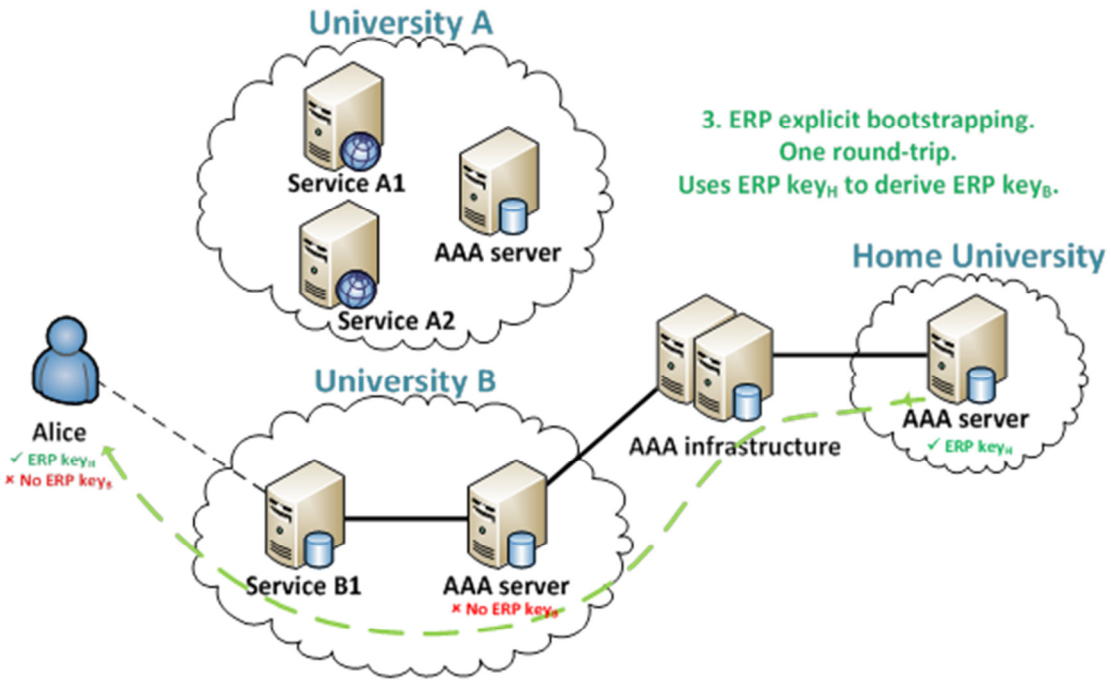

Figure 4.12: ERP explicit bootstrapping.

From the point of view of the authorization process, EAP and ERP are intended to perform authentication only. Authorization must be implemented at the EAP lower layer level. For instance, GSS-EAP already defines an authorization mechanism to allow the transport of a SAML assertion from the home organization to the service. Section 4.2.2.2 provides some considerations on authorization when using ERP.

4.2.2 ERP extensions for the GSS-EAP mechanism

4.2.2.1 *Overview*

Although ERP was defined to be a standard mechanism to perform fast re-authentication with EAP, it requires the EAP peer, authenticator and servers state machines to be updated accordingly. This also implies updating EAP lower layer specifications. This fact has precluded its introduction in the existing standard wireless technologies, as it would have required substantial standardization effort. Besides, ERP follows a different exchange scheme than EAP. While a typical EAP exchange follows an authenticator-initiated scheme, ERP typically follows a client-initiated one. This different operation has made its introduction in current lower layers more difficult, as the latter have been usually designed following a server-initiated scheme, in order to be more compliant with what EAP does.

However, the GSS-API (and thus the GSS-EAP mechanism) operation follows a client-driven scheme. This aspect favours the integration of ERP into its message flow, minimizing the number of adaptations that are required to incorporate ERP capabilities into GSS-EAP. Specifically, it allows an easier negotiation of ERP

capabilities and parameters between GSS-EAP initiator (UA) and the GSS-EAP acceptor (application), and a simpler handling of error conditions (e.g. retransmissions). This integration process has been implemented as part of the T3.2 in CLASSe.

In particular, the changes required in the GSS-EAP mechanism in order to support ERP are the following:

1. The definition of a new subtoken for the GSS-EAP initial state, in order to allow the negotiation of ERP capabilities, notify the name of the ERP realm, and request the type of ERP exchange to be performed (i.e. implicit/explicit bootstrapping or local ERP operation). We have called this subtoken *ERP-Supported*.

2. The modification of the EAP state machines for the initiator (peer) and acceptor (authenticator), in order to incorporate ERP processing as described in [ERP] during the Initial and Authentication states. This includes the generation and parsing of ERP messages as well as the derivation, storage and usage of the ERP keys.

3. The modification of the acceptor behaviour, as it must notify its AAA server whenever the initiator requests implicit bootstrapping, in order to request an ERP key from the home AAA server.

Besides the change required to the GSS-EAP mechanism, the involved AAA servers also require some adaptations in order to support ERP. In particular the following changes are required:

1. The modification of the home AAA server (EAP server) to incorporate ERP processing as described in [ERP]. This includes the generation and parsing of ERP messages as well as the derivation, storage and usage of the ERP keys.

2. The modification of the AAA server in the visited domain in order to request the ERP key from the home AAA server during either the implicit or the explicit bootstrapping, and to store it for future use.

3. The modification of the AAA server in the visited domain in order to make use of the stored ERP keys to perform *local ERP exchanges* with the UA.

Figure 4.13 depicts a simplified overview of the GSS-EAP mechanism with the ERP extensions. Note that steps 2a, 2b, and 2c are mutually exclusive. The selection of which one will be executed is performed during the first step. For a more detailed description of the message flow, please refer to the Appendix in [D31].

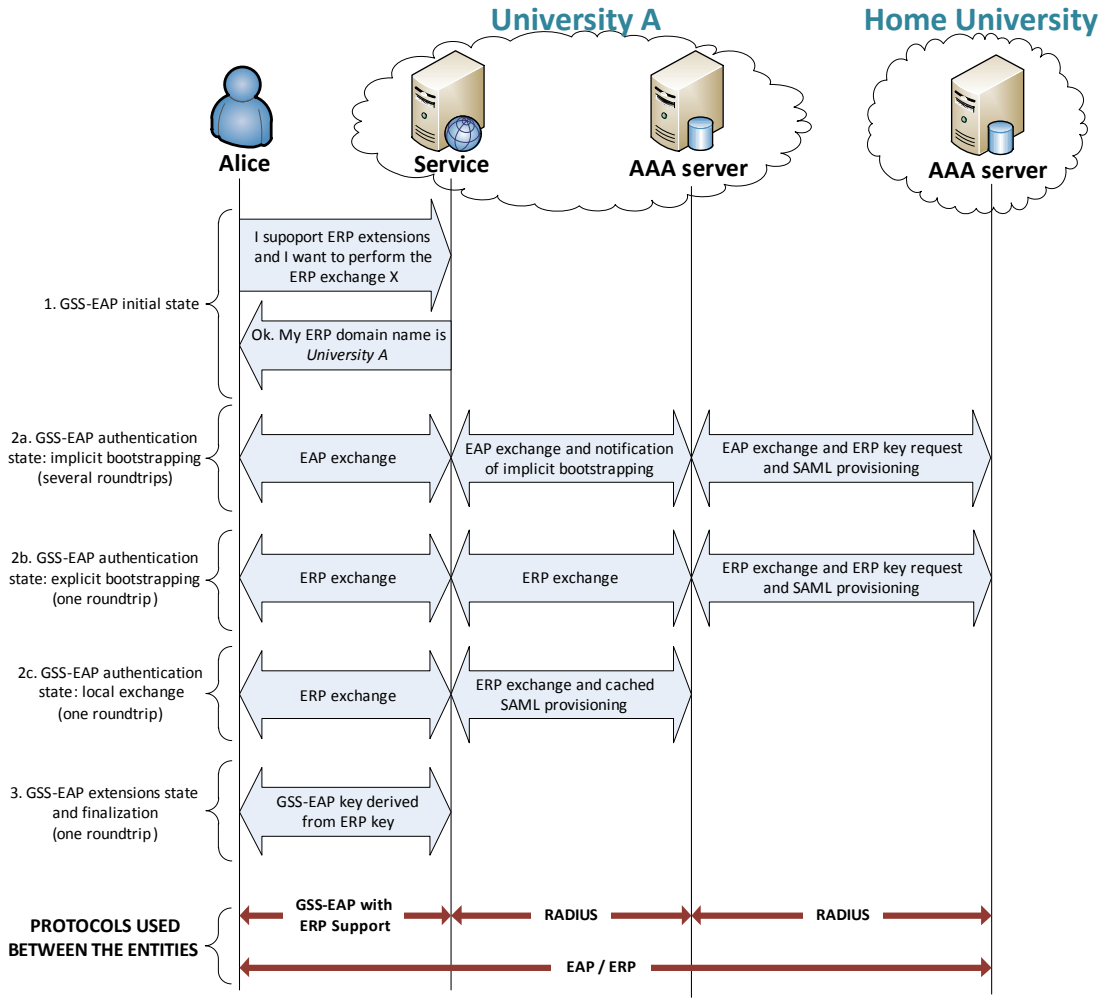

Figure 4.13: Overview of the ERP extensions to GSS-EAP

4.2.2.2 *Authorization considerations*

An important aspect has to do with how authorization is handled in this solution. As mentioned in the previous section, the GSS-EAP mechanism determines that the home AAA server generates a SAML assertion for the authenticated UA after a successful EAP authentication. In principle, this process does not need of any change for the ERP implicit bootstrapping and the ERP explicit bootstrapping scenarios. At the end of the ERP process, the home AAA server generates the SAML assertion and delivers it to the application in the same way as it does during a regular GSS-EAP exchange.

However, there is an important consideration when it comes to the local ERP exchange. Since the home AAA server is not involved in the process, there is no way for it to generate and deliver an SAML assertion.

To overcome this limitation, and to take advantage of the ERP local operation, we consider that the visited AAA server may store the SAML assertion received during any of the bootstrapping phases (either implicit or explicit)

previously executed. In this way, it could just use that cached SAML assertion to deliver the authorization information to the application during the local ERP exchange. However, further work is still required in order to evaluate the pros and cons of such an approach.

Another aspect worth to be noted is that the SAML assertion generated by the IdP might be larger than the maximum length for a RADIUS packet (4096 bytes). This limitation might preclude the execution of ABFAB when the AAA infrastructure is based on RADIUS. Within CLASSe we have been working, in collaboration with Telefonica I+D and Network RADIUS, on a proposal to allow RADIUS peers to exchange large amounts of authorization data exceeding this limit, by fragmenting RADIUS packets across several exchanges. The proposed solution does not impose any additional requirements to the RADIUS system administrators (e.g. need to modify firewall rules, set up web servers, configure routers, or modify any application server). It maintains compatibility with intra-packet fragmentation mechanisms (like those defined in [RFC3579] or in [RFC6929]). It is also transparent to existing RADIUS proxies that do not implement this specification. The only systems needing to implement it are the ones that either generate, or consume the fragmented data being transmitted. This effort is being standardized within the RADEXT WG of the IETF, where it has counted with great support. Indeed, it has already been accepted for publication as an IETF Experimental RFC [RADFRAG].

Note that if the SAML assertion is big enough to require RADIUS fragmentation [RADFRAG], the number of round-trips might be increased between the service and the IdP. This would be especially noticeable during the explicit bootstrapping, which otherwise would be performed in just one round-trip.

It must be also noted that the use of the Trust Router [TRUST-ROUTER] infrastructure does not affect to how this proposal works. In fact, it would be beneficial as the home domain would be sending the cryptographic material and the SAML assertion to the visited domain through a secured channel, so no intermediary entity (trusted or not) will see the data.

4.2.3 Other alternatives

Besides ERP, another ABFAB-based alternative has been analyzed and evaluated within the context of this task: FedKERB [CSI, SURVEY].

Kerberos is a widely extended key distribution protocol that allows the establishment of dynamic trust relationships between the UA and an application service by means of the distribution of the so-called Kerberos tickets. Kerberos processing is composed of three steps. On the first one, the UA authenticates with the KDC using some long-term credentials, and obtains a ticket as a result (called TGT). On the second one, the UA request a ticket for a particular service, using the TGT as a credential. As a result, the UA obtains another ticket (called ST). Finally, on the third step, the UA requests access to the service, using the ST as a credential.

The advantage of Kerberos is that subsequent accesses to services deployed within the same domain only require to perform steps 2 (obtain a ST) and 3 (access the service). Step 1 (authenticate and obtain a TGT), which is typically the most expensive one, is omitted. Moreover, if the UA already has a ST for the service, only step 3 needs to be performed.

FedKERB defines a mechanism that allows performing the first step of the Kerberos process by means of a federated EAP authentication. In particular, this EAP authentication process is defined as a GSS-EAP execution between the UA and the KDC.

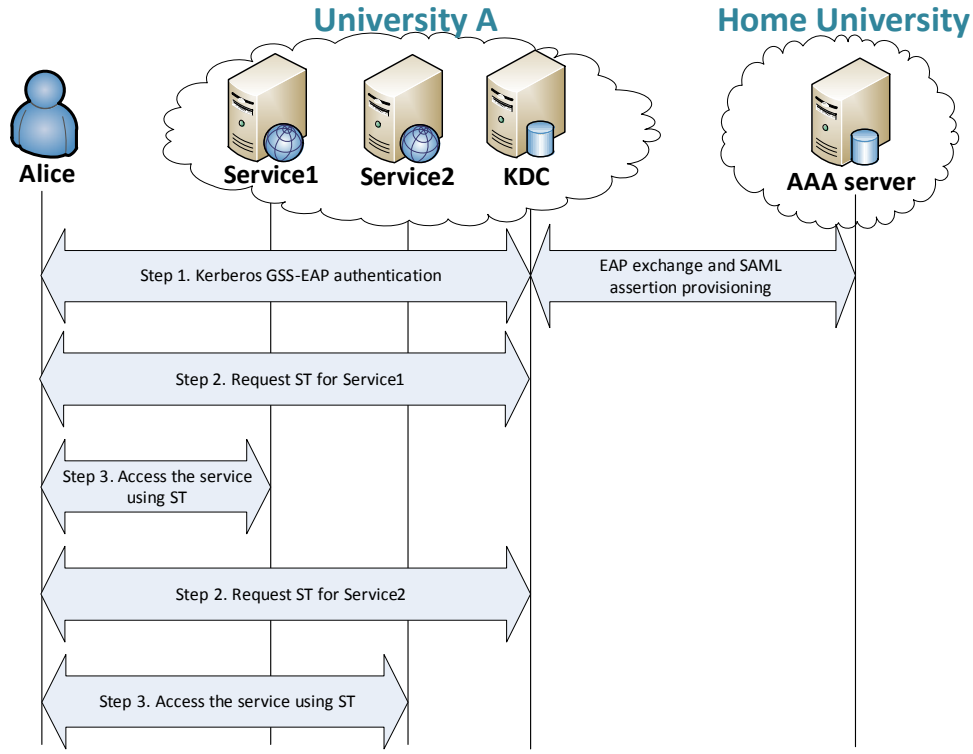

Figure 4.14: FedKERB operation.

The main advantage of FedKERB over ERP is the direct application of the ABFAB/Moonshot procedure, without requiring the modification of the GSS-EAP mechanism. On the downside, it requires the inclusion of a KDC in the architecture and the modification of the Kerberos authentication process. Besides, the provided SSO only works within the visited domain. That is, whenever the UA moves to a different visited domain, a complete EAP authentication needs to be performed. Finally, FedKERB does not provide a solution for the network-to-cloud scenario.

In the context of this task we also explored solutions that were not based on ABFAB, but have some common aspects. In particular, the IETF is currently working on a SASL and GSS-API mechanism for SAML 2.0 that leverages the capabilities of an *enhanced client* to address federated authentication reusing existing SAML bindings and profiles [SASL-SAML-EC]. In this way, SAML can be used for arbitrary applications as long as they support either the SASL framework or the GSS-API (as it happens with ABFAB/Moonshot). The main advantage of this approach is that it is under standardization on the IETF WG, and it requires little modifications on existing SAML capable RPs and IdPs. Its main disadvantage is that is does not provide a solution for the network-to-cloud scenario, and that it always requires contacting the IdP, typically using a TLS/SSL channel. This undermines part of the objectives of this task, which aims to minimize the amount of network traffic required.

Therefore, for the requirements of this task and the context of CLASSe project, the solution based on ERP shows better features than the rest and therefore, FedKERB and the SASL SAML EC mechanism has been discarded.

4.3 Description of the proof-of-concept implementation

The purpose of this proof-of-concept implementation is to provide a functional demonstrator of the design for real SSO described in section 4.2. In particular, this prototype implements the functionality associated to the Cloud-to-cloud scenario. To do that, we must modify the source code of the Moonshot and FreeRADIUS software components of our initial testbed [MS3], in order to provide them with support for ERP. The following subsections describe these modifications with a higher level of detail. The Network-to-cloud scenario would require modifying the wpa_supplicant software in a similar way as we have done for Moonshot. However, it has not been implemented in this proof-of-concept since the objective was not to provide the full functionality, but to demonstrate and measure the potential and actual reduction that can be achieved using the design of section 4.2.2.

4.3.1 Moonshot

Moonshot [MOONSHOT] is subdivided into several modules, being the most important the one that implements the GSS-EAP [GSS-EAP] mechanism. This module consists of two different libraries. Both of them have required modifications, as described in the following subsections.

4.3.1.1 *Libeap*

It provides the implementation of the EAP [EAP] peer and EAP authenticator state machines. It is based on the source code of the wpa_supplicant [WPASUP] solution, with some basic modifications to obviate the absence of the EAP-over-LAN (EAPoL) layer.

We have modified this library in order to extend the EAP state machine to incorporate the states, transitions, and functionality associated to ERP. A patch including the modifications to this library that we have performed is available in the CLASSe web page [GSSERPCODE]. A summary is provided in the following list:

- Extended the definition of the EAP codes enumeration to include *EAP_CODE_INITIATE* and *EAP_CODE_FINISH*.
- Extended the definition of the EAP types enumeration to include *EAP_TYPE_RE_AUTH_START* and *EAP_TYPE_RE_AUTH*.
- Defined the new states for the ERP processing (e.g. *ERP_INITIALIZE*, *ERP_IDLE, ERP_RECEIVED*…), and extend the state machine main loop to accept transitions to them.
- Defined the procedures associated to the transition to those states (e.g. *eap_sm_buildInitiateReAuth, eap_sm_processInitiateReauthStart*…).

- Defined several helper functions to derive, store, and load the ERP keying material (e.g. *prfplus, eap_sm_setDsrk, eap_sm_deriveEmskName, eap_sm_deriveErpKeys*…). Keys are stored and load from files in the /tmp folder for the proof-of-concept

- Extend the library to enable the implementation of the EAP-TTLS method to derive an EMSK, and defined a method to get it (i.e. *get_emsk*).

- Defined a new configuration parameter (*acceptorRealm*), which defines the ERP realm of the acceptor.

- Defined a persistent storage of ERP keys. In this proof-of-concept implementation, this storage is based on files that are located in the /tmp folder. This is just for testing purposes, as storing unencrypted secret keys in the filesystem implies a high security risk. For production environments, this should be substituted by a secure storage such as system keyrings. These files are structured as follows:

    - A /tmp/emsk file stores the value of the EMSK derived during the ERP implicit bootstrapping.

    - A /tmp/dsrk-{domainname} file stores the value of the DSRK derived during the ERP implicit or explicit bootstrapping

- Defined a new flag that indicates the lower layer which type of ERP exchange can be performed based on the available keying material.

4.3.1.2 *Mech_eap*

It provides the actual implementation of the GSS-API Mechanism for the Extensible Authentication Protocol. This library makes use of libeap for all the EAP-related functionality. mech_eap is called by the operating system's GSS layer when the GSS-EAP mechanism is selected by an application.

We have modified this library in order to extend the GSS-EAP state machine to allow the use of ERP messages. A patch including the modifications to this library that we have performed is available in the CLASSe web page [GSSERPCODE]. A summary is provided in the following list:

- Creation and exchange of the ERP-Supported subtokens, where ERP parameters are negotiated between GSS initiator and acceptor.

- Determine the kind of EAP exchange that will be performed based on the information received from the ERP state machine and the realm of the GSS acceptor.

- Allow including EAP-Initiate/Re-auth-start instead of the EAP-Identity/Request when required.

- Allow processing EAP-Initiate/Re-auth as a source of identity instead of an EAP-Identity/Response when required.

4.3.2 FreeRADIUS

FreeRADIUS [FREERAD] is the most relevant RADIUS open-source implementation nowadays. Moreover, according to its web page, it is the most widely deployed RADIUS server in the world (including also the private solutions). It contains an EAP module (called *rlm_eap*), which implements the EAP layer.

FreeRADIUS is used with two different purposes in the proof-of-concept. First, it is used to implement the RADIUS server (also called IdP in Moonshot), which is in charge of the EAP authentication of the UA. Second, it is also used to implement the RADIUS proxy, which interconnects the cloud service with the IdP. We have modified this module in order to extend the EAP state machine to incorporate the states, transitions, and functionality associated to ERP in the IdP, and to manage the ERP keys in both, the IdP and the RADIUS proxy. A patch including the modifications to this library that we have performed is available in the CLASSe web page [GSSERPCODE]. A summary is provided in the following list:

- Extend the definition of the EAP codes enumeration to include *PW_EAP_INITIATE* and *PW_EAP_FINISH*.
- Create a state in the IdP to store the EMSK after the authentication of the Client during an implicit bootstrapping.
- Generate and distribute the DSRK to the cloud service's RADIUS proxy during implicit and explicit bootstrapping.
- Create a state in the RADIUS proxy to store the DSRK associated to a particular Client after receiving it from the IdP after a bootstrapping process (either implicit or explicit).
- Use stored ERP keys (EMSK, DSRK) to authenticate the Client in the explicit bootstrapping and local ERP operation respectively.
- Generate the rMSK accordingly to the used ERP key material (both IdP and proxy).
- Add a parameter to the EAP module to set the ERP realm of the IdP.

4.3.3 Execution of the proof-of-concept

The execution of this proof-of-concept implementation differs nothing from a normal Moonshot execution from a point of view of the services execution. As described in section 4.2.2, the modified mech_eap code will start exchanging ERP-Supported subtokens, in order to determine whether both the initiator and the acceptor support ERP, and to provide the initiator with the ERP realm of the acceptor in the latter case. If ERP is not supported, a standard EAP exchange is performed. If ERP is supported, the mech_eap implementation decides, based on the available keying material, the kind of ERP exchange that will be performed (implicit bootstrapping, explicit bootstrapping or local operation).

4.4 Performance analysis

The implementation of real SSO during the access to cloud services is envisioned to provide an improved performance against the use of traditional SSO. This improvement must be noticeable in terms of both, the time required to complete the authentication and authorization processes, and the amount of traffic that is delivered through the network.

In order to demonstrate this, we have carried out a performance analysis that compares the execution of the both traditional SSO and real SSO strategies that have been defined in this project, during a complete process of accessing to the Swift OpenStack service. In particular, we have made use of the unmodified Moonshot implementation, which provides a basic traditional SSO approach, and that is used as a reference for the performance analysis. Secondly, we have measured the performance of the proof-of-concept implementation produced as a result of task T3.1. This approach includes support for an optimized traditional SSO based on the use of TLS-resumption. Finally, we included in the comparison the results of the execution of proof-of-concept implementation described in this document, which provides a real SSO approach based on the use of ERP.

This analysis includes the following steps:

1. The setup of a testbed where these three implementations will be executed (section 4.4.1).

2. The measurement of different performance indicators (computational time, network time, data traffic…) that have relevance for this analysis (section 4.4.2).

3. The calculation of statistical results and its analysis, in order to extract some conclusions (section4.4.3).

4.4.1 Testbed description

In order to analyse these implementations we need a realistic scenario that simulates the actual conditions that one might find on a Moonshot deployment. A preliminary tested was defined in [MS3] to measure the time of traditional SSO. However, we decided to evolve it in order to have a simpler one without prejudice to the validity of the obtained results. For this reason, we have defined the following testbed depicted in Figure 4.15.

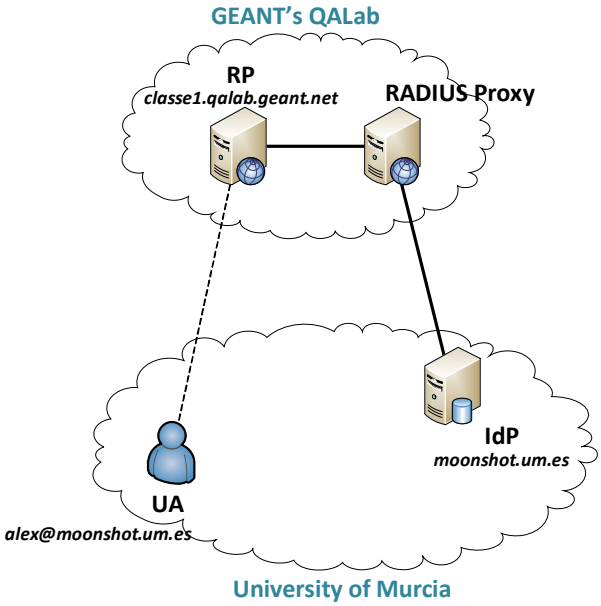

Figure 4.15: Testbed configuration

The testbed involves two different locations: University of Murcia and GÉANT's QALab [QALAB]. In particular, it consists of:

- GÉANT's QALab
    - One RP (cloud server), deployed on a Virtual Machine (VM), and hosted at the GÉANT QALab (classe1.qalab.geant.net). It is provisioned with an Intel Xeon CPU (2.67GHz) and 2GB of RAM. It runs OpenStack [OPENSTACK] (Havana version) instance with support for Moonshot based authentication [FEDKEYSTONE]. In particular, it deploys the OpenStack Swift service (file storage), as well as the Keystone (authentication/authorization) one. This RP is connected to a RADIUS proxy deployed also at QALab.
    - A RADIUS proxy, deployed on a Virtual Machine (VM), and hosted at the GÉANT QALab (classe2.qalab.geant.net). It is provisioned with an Intel Xeon CPU (2.67GHz) and 2GB of RAM. It runs FreeRADIUS 3.x instance and interconnects the cloud service with the IdP located at the University of Murcia.
- University of Murcia
    - An IdP (RADIUS server), deployed on a dedicated VM, and hosted at the University of Murcia. It is provisioned with an AMP Opteron Processor (2GHz) and 128 MB of RAM. It handles the moonshot.um.es test RADIUS realm.

- A UA (User Agent), deployed on a VM. It is provisioned with an Intel Core i3 CPU (3.3GHz) and 1GB of RAM. It has an OpenStack Swift service client installed with Moonshot support [FEDSWIFTCLIENT]. The UA has a testing identity on the UMU's IdP: alex@moonshot.um.es.

4.4.2 Measurement of performance indicators

For the performance analysis we have measured the following indicators:

- Computational time (COMP). The amount of time spent by each one of the involved entities to perform the required functionality. This indicator specifically excludes network delays.
- Network time (NETW). The amount of time spent delivering messages between each pair of the involved entities.
- Total time (TOTAL_TIME). The amount of time that is required at the UA to complete the access process. It is the result of the addition of all the computational times (COMPs) and network times (NETWs).
- Network data (DATA). The amount of data (bytes) that is transmitted between each pair of the involved entities.
- Total traffic (TOTAL_TRAF). The total amount of data that is transmitted in the network due to the execution of an access process.

In order to measure these indicators we have made use of the Wireshark [WIRESHARK]. Using this software, we have captured the network traffic on each one of these four components (UA, RP, PROXY, and IDP) during a complete process of accessing to the cloud service. Using the captured data, we have calculated, with the help of a set of python scripts, the values of these indicators as follows:

- COMP(X). We have calculated the value of this indicator as the sum of the time elapsed in component X between the reception of a message (either a request or a response) until the emission of a new message.
- WAIT(X). We have calculated this intermediate value as the sum of the time elapsed in component X between the emissions of a request message until the reception of the corresponding response message.
- NETW(X, Y). We have calculated the value of this indicator as WAIT(X) – COMP(Y) – WAIT(Y).
- TOTAL_TIME. We have calculated the value of this indicator as COMP(UA) + NETW(UA,RP) + COMP(RP) + NETW(RP, PROXY) + COMP(PROXY) + NETW(PROXY, IDP) + COMP(IDP). It should be equivalent to COMP(UA) + WAIT(UA).
- DATA(X, Y). We have calculated the value of this indicator as the sum of the size of the messages (either request or responses) transmitted between components X and Y.

TOTAL_TRAF. We have calculated the value of this indicator as DATA(UA, RP) + DATA(RP, PROXY) + DATA(PROXY, IDP)

Figure 4.16 depicts the distribution of the indicators we have measured.

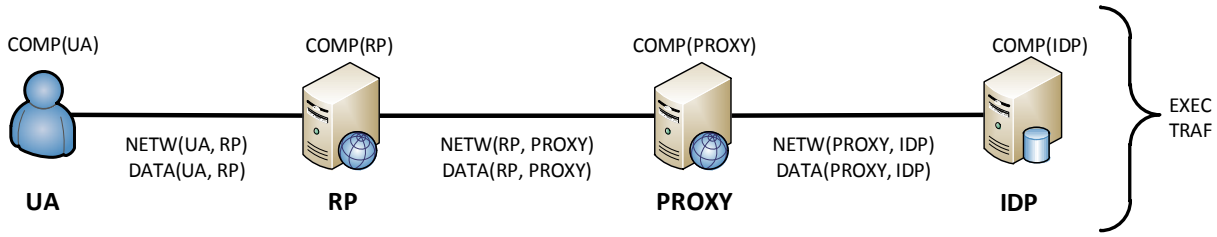

Figure 4.16: Distribution of indicators

4.4.3 Results

Once we have defined the way we have measured the performance indicators, we can obtain representative results by executing several times each one of these five configurations over our testbed:

- Access to a cloud server using the unmodified Moonshot implementation.
- Access to a cloud server using Moonshot with TTLS-resumption support (T3.1).
- Access to a cloud server using Moonshot with ERP support (T3.2). This configuration has three different configurations:
    - ERP implicit bootstrapping.
    - ERP explicit bootstrapping.
    - Local ERP operation.

In particular, we have performed 200 executions per configuration. From these executions we have calculated the 95% confidence interval for each one of the performance indicators described above.

Before analysing the results, it is important to note that the EAP authentication is just a part of the process of accessing to a cloud service. Other tasks take also part of this process, such as the generation of OpenStack tokens, the encapsulation of GSS-API packets within Keystone messages, etc. Hence, following the Amdahl's law [AMDAHL], the overall time to perform the whole process of accessing to the cloud service will not be proportionally affected by the reductions we introduce into the EAP authentication process. Conversely, the overall reduction will be limited by the amount of time actually spent in the EAP authentication.

Table 4.1 shows the obtained results for the time-based performance indicators (COMP, NETW, and EXEC). Conversely, Table 4.2 shows the results for the data-based performance indicators (DATA and TRAFFIC).

As it can be observed in Table 4.1, both the unmodified *Moonshot* and the *Moonshot ERP Implicit Bootstrapping* configurations show a similar overall execution time (EXEC) of around 2700 ms. These times are the highest of the five configurations. This is something we expected, since both require the execution of a complete EAP authentication process with the home AAA server.

On the other hand, the use of TTLS-resumption reduces the overall execution time (EXEC) to around 2050 ms, meaning a reduction of around a 24%. Conversely, the use of an ERP explicit bootstrapping process reduces this time even further: up to around 1770 ms., resulting into a reduction of around a 35% when compared to the unmodified Moonshot configuration. Finally, the execution of an ERP local operation requires around 1660 ms. of execution time (EXEC), implying a reduction of around a 39% over the unmodified Moonshot configuration. If we focus on the usage of the AAA infrastructure (NETW(PROXY, IDP) and COMP(IDP)), we can see reduction between the 86% (explicit bootstrapping) and 100% (local operation).

Table 4.1 also tells us where these reductions come from. If we look at the computational times (COMP), only small differences among the five configurations can be observed. This means that the reduction on the overall time (EXEC) comes primarily from the reduction of the network times (NETW), as can be observed from the table below.

CONFIGURATION	Moonshot	Moonshot TTLS-Resump	Moonshot ERP Impl. Boost.	Moonshot ERP Expl. Boost.	Moonshot ERP Local operation
COMP(UA)	406.74	402.23	409.47	415.85	405.14
milliseconds	±1.11	±1.03	±1.18	±1.94	±1.11
NETW(UA, RP)	1163.89	863.11	1162.72	711.12	706.54
milliseconds	±8.38	±18.57	±9.95	±13.13	±12.37
COMP(RP)	589.72	560.63	588.72	567.84	546.94
milliseconds	±5.40	±3.90	±4.44	±7.43	±4.10
NETW(RP, PROXY)	1.26	0.55	1.29	0.17	0.21
milliseconds	±0.38	±0.04	±0.21	±0.01	±0.01
COMP(PROXY)	2.01	0.89	2.07	0.38	0.44
milliseconds	±0.03	±0.02	±0.03	±0.01	±0.00
NETW(PROXY, IDP)	556.52	227.10	504.00	76.19	0.00
milliseconds	±0.37	±0.03	±0.21	±0.29	±0.00
COMP(IDP)	6.99	1.84	7.06	0.53	0.00
milliseconds	±0.04	±0.01	±0.11	±0.01	±0.00
TOTAL_TIME	**2727.13**	**2056.35**	**2675.34**	**1778.98**	**1659.28**
milliseconds	**±8.70**	**±18.65**	**±10.23**	**±19.75**	**±12.53**

Table 4.1: Obtained results for the COMP, NETW, and TOTAL_TIME performance indicators

The reduction in the network time is a direct consequence of the reduction of the amount of network traffic exchanged between each pair of entities (DATA). This relation is more relevant between the UA and the RP and between the RP and the PROXY, since the distance between them make that transmitting a larger amount of bytes takes a higher time.

As can be observed in Table 4.2, the use of TTLS resumption has resulted into a reduction of around a 25% on the overall amount of data traffic (TRAF), whereas the use of ERP explicit bootstrapping and ERP local operation has resulted into a reduction of a 32% and a 34% respectively. If we focus on the usage of the AAA

infrastructure (DATA(PROXY, IDP)), we can see reductions between the 75% (explicit bootstrapping) and 100% (local operation).

CONFIGURATION	Moonshot	Moonshot TTLS-Resump	Moonshot ERP Impl. Boost.	Moonshot ERP Expl. Boost.	Moonshot ERP Local operation
DATA(UA, RP) *bytes*	51578.95 ±45.87	43154.40 ±25.84	51663.03 ±21.41	40532.37 ±34.38	40444.51 ±10.37
DATA(RP, PROXY) *bytes*	6366.00 ±0.00	2384.00 ±0.00	6438.00 ±0.00	1604.98 ±15.73	1485.00 ±0.00
DATA(PROXY, IDP) *bytes*	6450.00 ±0.00	2420.00 ±0.00	6662.00 ±0.00	1645.00 ±0.00	0.00 ±0.00
TOTAL_TRAF *bytes*	**64394.95 ±45.87**	**47958.40 ±25.84**	**64763.03 ±21.41**	**43682.35 ±50.11**	**41929.51 ±10.37**

Table 4.2: Obtained results for the DATA, and TOTAL_TRAF performance indicators.

4.5 Applying the results to a practical scenario

The results obtained from our performance analysis (described in section 4.4.3) give raw information about the reductions that are expected when using the real SSO proposal that we have designed in this project. However, they need to be further analysed and interpreted in order to evaluate the actual impact that these solutions might have in the cloud-to-cloud and network-to-cloud scenarios, when applied in a more realistic environment (e.g. GÉANT network).

The first thing to take into account is that one cannot select the kind of ERP exchange that will be performed in a particular moment. This type of exchange will be selected automatically based on the keying material available to the UA and the RP or PROXY. So, in order to obtain an accurate approximation of the expected performance improvement that can be obtained, it is required to put this proposal in context with a typical cloud-to-cloud and network-to-cloud use cases. The following subsections provide an overview to these processes.

4.5.1 Cloud-to-cloud

In this scenario it is assumed that the authentication to the network is out of the scope. That is, the UA will start already connected to the network, and how that process was performed does not affect in any way to the process of accessing to the cloud service performed by means of Moonshot.

A typical cloud-to-cloud scenario consists on an end user accessing to several cloud servers during a working day or session. These cloud servers may be located at the same or different organizations. The end user may access several times to the same cloud server as well. Hence, we can group the accesses to these servers as follows:

- **First access to a cloud server**. This is the first access to a cloud server during a working day or session. It happens once per session.

- **Access to a cloud server located in a different organization**. The end user accesses to a cloud server located in an organization different to those already visited during the previous session. It might happen from 0 to M times per session.

- **Access to a cloud server located in the same organization**. The end user accesses to a cloud server located in the same organization where the end user has already authenticated within the current session (it might be the same server). It can happen from 0 to N times per session.

Therefore, we can conclude that the time required for each one of these kind of access will be the following, depending whether unmodified Moonshot or Moonshot with support for ERP is being used:

	Moonshot	Moonshot with ERP support
First access	$TOTAL\_TIME_{moonshot}$	$TOTAL\_TIME_{implicit}$
Access in a different organization	$TOTAL\_TIME_{moonshot}$	$TOTAL\_TIME_{explicit}$
Access in the same organization	$TOTAL\_TIME_{moonshot}$	$TOTAL\_TIME_{local}$

Table 4.3: Time required to perform each kind of access in the cloud-to-cloud scenario

Based on that, we can calculate that a typical cloud-to-cloud session will require the following time to complete the different access control processes:

$$T_{moonshot} = TOTAL\_TIME_{moonshot} + M * TOTAL\_TIME_{moonshot} + N * TOTAL\_TIME_{moonshot}$$

$$T_{erp} = TOTAL\_TIME_{implicit} + M * TOTAL\_TIME_{explicit} + N * TOTAL\_TIME_{local}$$

And we can calculate the amount of time of that session that is spent using the AAA infrastructure as the following:

$$AAA_{moonshot} = NETW(PROXY,IDP)_{moonshot} + COMP(IDP)_{moonshot} + M * (NETW(PROXY,IDP)_{moonshot} + COMP(IDP)_{moonshot}) + N * (NETW(PROXY,IDP)_{moonshot} + COMP(IDP)_{moonshot})$$

$$AAA_{erp} = NETW(PROXY,IDP)_{implicit} + COMP(IDP)_{implicit} + M * (NETW(PROXY,IDP)_{explicit} + COMP(IDP)_{explicit}) + N * (NETW(PROXY,IDP)_{local} + COMP(IDP)_{local})$$

Table 4.4 represents the value of these four variables ($T_{moonshot}$, T_{erp}, $AAA_{moonshot}$, and AAA_{erp}), as well as the % of reduction achieved. For simplicity, we have assumed N and M to take the same value, comprised between 0 and 20 (which we consider a reasonable top limit for the number of authentications per session). As it can be observed, as long as there are a few of different accesses to cloud servers (either in the same or different organization), it quickly tends to reach the 35-39% of total reduction, and the 86-100% of reduction in the AAA usage, stated in the previous section.

N	M	$T_{moonshot}$ *milliseconds*	T_{erp} *milliseconds*	% of total reduction	$AAA_{moonshot}$	AAA_{erp}	% of AAA reduction
0	0	2727	2675	1.91	564	511	9.40
1	1	8181	6113	25.28	1692	646	61.82

2	2	13635	9551	29.95	2820	722	74.40
3	3	19089	12989	31.96	3948	798	79.79
4	4	24543	16427	33.07	5076	874	82.78
5	5	29997	19865	33.78	6204	950	84.69
6	6	35451	23303	34.27	7332	1026	86.01
7	7	40905	26741	34.63	8460	1102	86.97
8	8	46359	30179	34.90	9588	1178	87.71
9	9	51813	33617	35.12	10716	1254	88.30
10	10	57267	37055	35.29	11844	1330	88.77
11	11	62721	40493	35.44	12972	1406	89.16
12	12	68175	43931	35.56	14100	1482	89.49
13	13	73629	47369	35.67	15228	1558	89.77
14	14	79083	50807	35.75	16356	1634	90.01
15	15	84537	54245	35.83	17484	1710	90.22
16	16	89991	57683	35.90	18612	1786	90.40
17	17	95445	61121	35.96	19740	1862	90.57
18	18	100899	64559	36.02	20868	1938	90.71
19	19	106353	67997	36.06	21996	2014	90.84
20	20	111807	71435	36.11	23124	2090	90.96

Table 4.4: Values for $T_{moonshot}$, T_{erp}, $AAA_{moonshot}$, and AAA_{erp} with different combinations of N and M for the cloud-to-cloud scenario for the case of N=M.

In terms of data exchanged, and based on the results obtained in the previous section, we can define the following variables:

$$DATA_{moonshot} = TOTAL\_TRAF_{moonshot} + M * TOTAL\_TRAF_{moonshot} + N * TOTAL\_TRAF_{moonshot}$$

$$DATA_{erp} = TOTAL\_TRAF_{implicit} + M * TOTAL\_TRAF_{explicit} + N * TOTAL\_TRAF_{local}$$

And the amount of data transferred over the AAA infrastructure will be the following:

$$DAAA_{moonshot} = DATA(PROXY,IDP)_{moonshot} + M * DATA(PROXY,IDP)_{moonshot} + N * DATA(PROXY,IDP)_{moonshot}$$

$$DAAA_{erp} = DATA(PROXY,IDP)_{implicit} + M * DATA\ (PROXY,IDP)_{explicit} + N * DATA(PROXY,IDP)_{local} +$$

Table 4.5 represents the value of these variables in a similar way as done above in Table 4.4. As it can be observed, as long as there are a few different accesses to cloud servers (either in the same or different organization), it quickly reaches a 32-35% of total improvement and 75-100% of improvement in the AAA usage, as stated in the previous section.

N	M	Dmoonshot *bytes*	Derp *bytes*	% of total reduction	DAAAmoonshot *bytes*	DAAAerp *bytes*	% of AAA reduction
0	0	64395	64395	0.00	6450	6450	0.00
1	1	193185	150375	22.16	19350	8307	57.07
2	2	321975	235987	26.71	32250	9952	69.14

N	M	Dmoonshot *bytes*	Derp *bytes*	% of total reduction	DAAAmoonshot *bytes*	DAAAerp *bytes*	% of AAA reduction
3	3	450765	321599	28.65	45150	11597	74.31
4	4	579555	407211	29.74	58050	13242	77.19
5	5	708345	492823	30.43	70950	14887	79.02
6	6	837135	578435	30.90	83850	16532	80.28
7	7	965925	664047	31.25	96750	18177	81.21
8	8	1094715	749659	31.52	109650	19822	81.92
9	9	1223505	835271	31.73	122550	21467	82.48
10	10	1352295	920883	31.90	135450	23112	82.94
11	11	1481085	1006495	32.04	148350	24757	83.31
12	12	1609875	1092107	32.16	161250	26402	83.63
13	13	1738665	1177719	32.26	174150	28047	83.89
14	14	1867455	1263331	32.35	187050	29692	84.13
15	15	1996245	1348943	32.43	199950	31337	84.33
16	16	2125035	1434555	32.49	212850	32982	84.50
17	17	2253825	1520167	32.55	225750	34627	84.66
18	18	2382615	1605779	32.60	238650	36272	84.80
19	19	2511405	1691391	32.65	251550	37917	84.93
20	20	2640195	1777003	32.69	264450	39562	85.04

Table 4.5: Values for $D_{moonshot}$, D_{erp}, $DAAA_{moonshot}$, and $DAAA_{erp}$ with different combinations of N and M for the cloud-to-cloud scenario for the case of N=M.

4.5.2 Network-to-cloud

In this scenario it is assumed that the end user is required to perform a federated access to the network prior accessing any of the available services. In our solution, we have leveraged this EAP authentication to allow real SSO for the cloud services.

A typical network-to-cloud scenario consists on an end user accessing to the network in a federated way at the beginning of a session, and then accessing to several cloud servers afterwards. These cloud servers may be located at the same or different organizations. Hence, we can group the accesses to these servers as follows:

- **Access to a cloud server located in the same organization**. The end user accesses to a cloud server located in the same organization than a cloud server where the end user has already authenticated within the current session (it might be the same server), or where the end user has authentication to access to the network. It can happen from 0 to N times per session.

- **Access to a cloud server located in a different organization**. The end user accesses to a cloud server located in an organization different to those already visited during the current session. It might happen from 0 to M times per session.

Therefore, we can conclude that the time required for each one of these kind of access will be the following, depending whether unmodified Moonshot or Moonshot with support for ERP is being used:

	Moonshot	Moonshot with ERP support
Access in a different organization	$TOTAL\_TIME_{moonshot}$	$TOTAL\_TIME_{explicit}$
Access in the same organization	$TOTAL\_TIME_{moonshot}$	$TOTAL\_TIME_{local}$

Table 4.6: Time required to perform each kind of access for the network-to-cloud scenario.

Based on that, we can calculate that a typical cloud-to-cloud session will require the following time to complete the different access control processes:

$$T_{moonshot} = M * TOTAL\_TIME_{moonshot} + N * TOTAL\_TIME_{moonshot}$$

$$T_{erp} = M * TOTAL\_TIME_{explicit} + N * TOTAL\_TIME_{local}$$

And we can calculate the amount of time of that session that is spent using the AAA infrastructure as the following:

$$AAA_{moonshot} = M * (NETW(PROXY,IDP)_{moonshot} + COMP(IDP)_{moonshot}) + N * (NETW(PROXY,IDP)_{moonshot} + COMP(IDP)_{moonshot})$$

$$AAA_{erp} = M * (NETW(PROXY,IDP)_{explicit} + COMP(IDP)_{explicit}) + N * (NETW(PROXY,IDP)_{local} + COMP(IDP)_{local})$$

As it can be observed, these formulas are similar to the ones in the previous section. The difference is that the first operand, which corresponded to the first access to a cloud service, is absent. The reason is that the first EAP authentication will be performed during the access to the network. This fact makes that both, the % of total reduction and the % of AAA reduction, to be constant for any value of M=N, as described in the following formula:

$$REDUCTION = \frac{T_{erp}}{T_{moonshot}} = \frac{N*(TOTAL\_TIME_{explicit} + TOTAL\_TIME_{local})}{N*2*TOTAL\_TIME_{moonshot}}$$

$$\frac{TOTAL\_TIME_{explicit} + TOTAL\_TIME_{local}}{2*TOTAL\_TIME_{moonshot}} = CONSTANT\_VALUE$$

In particular, it results into a constant total percentage of reduction of a 36.96%, and a reduction in the AAA usage of a 93.26%.

In terms of data exchanged, we can define the following variables:

$$DATA_{moonshot} = M * TOTAL\_TRAF_{moonshot} + N * TOTAL\_TRAF_{moonshot}$$

$$DATA_{erp} = M * TOTAL\_TRAF_{explicit} + N * TOTAL\_TRAF_{local}$$

And the amount of data transferred over the AAA infrastructure will be the following:

$$DAAA_{moonshot} = M * DATA(PROXY,IDP)_{moonshot} + N * DATA(PROXY,IDP)_{moonshot}$$

$$DAAA_{erp} = M * DATA\ (PROXY,IDP)_{explicit} + N * DATA(PROXY,IDP)_{local} +$$

In a similar way as for the amount of time, the percentages of reduction are constant in this scenario when M=N. In particular, the % of total reduction is a 33.53%, whereas the reduction in the AAA usage is an 87.25%.

4.6 Conclusions

In task T3.3 we have developed a proof-of-concept implementation of the proposal for real SSO for cloud services and ABFAB, based on the extensions of the ABFAB architecture to include support for the *EAP Extensions for the EAP Re-authentication Protocol (ERP)* that were defined in T3.2. This section has provided an overview of this software, explaining how Moonshot and FreeRADIUS source code have been modified to include the required functionality. With this implementation we have demonstrated the feasibility of the proposal.

Using this proof-of-concept implementation, we have carried out a performance analysis in order to compare the unmodified Moonshot with the traditional SSO mechanism based on TTLS resumption defined in T3.1, and with the real SSO mechanism based on ERP defined in T3.2 and T3.3. For this analysis we have calculated 95% confidence intervals for each one of a series of performance indicators. Among these indicators, the more representative are TOTAL_TIME (total time required by the end user to successfully complete the access to a cloud service) and TOTAL_TRAF (total amount of data required to be exchanged between the different elements of the architecture during the access process).

The results of this performance analysis show how the use of the real SSO proposal based on ERP, and designed and implemented in this project, provides significant reductions to the process of accessing a cloud service when using ABFAB technologies. In particular, we have obtained that using ERP can reduce the overall time required to access to the cloud service in around a 35-39%. This reduction is much higher if we look at the savings in the AAA infrastructure, being able to reach reductions of about 86-100%. In terms of total amount of network traffic generated, the obtained reductions are similar. Specifically, the use of ERP can provide an overall reduction of around a 32-35%. If we focus on the AAA infrastructure, these reductions can reach the 75-100%.

However, these calculations are only valid when the end user is able to select the kind of ERP exchange to be executed. This is something that does not happen in real interactions, where the keying material available on the UA and on the visited domain conditions the selection. Therefore, these values should be considered as the maximum reduction we could reach using ERP. In order to obtain a more realistic estimation of the expected level of reduction in real conditions, we have described a typical cloud-to-cloud and network-to-cloud scenario, indicating the amount of the different ERP exchanges that would be executed. The obtained results have verified that the overall reduction that can be obtained in those scenarios tends very quickly to the maximum values obtained in our performance analysis, reaching values very close to the upper limit when the end user performs more than four accesses to cloud services (either different or the same).

Hence, the use of ERP to incorporate real SSO to the access to cloud services significantly reduces the overall time required to complete the process, improving the user experience and usability of the system. Besides, it drastically reduces the utilization of the AAA infrastructure. This improves its scalability, as the same architecture will be able to handle an exponential boost of the number of authentication processes without requiring of any upgrade in the equipment

5 WP4: Research on Communities of Interest (CoIs) for Cloud Services

Our project proposal said that we wanted to support the dynamic formation of communities of interest (CoIs), so that any users would be able to form their own CoIs for access to their own private cloud services. Once the project started, and we became more familiar with the CoI concept (which we admit was rather vague before then) we realised that CoIs were not the most appropriate group collaboration mechanism to achieve our objective. Specifically, CoIs are very similar to federations, in that they are a group of SPs and IdPs that have agreed to cooperate to provide access to a specific set of services to those users within a particular community. Consequently CoIs cannot be created without IdP involvement, which makes it very difficult, if not impossible, for users to form their own CoIs. Virtual Organisations, on the other hand, are a set of users from different organisations that co-operate to share resources for a common purpose. We therefore decided that the Virtual Organisation concept was a much better group collaboration mechanism to allow users to manage access to their own cloud services, as this was being used effectively in the grid world. With project management agreement, we therefore concentrated on providing support for VO role management within OpenStack, instead of support for CoIs.

Whilst a number of VO-like solutions are being investigated in the academic community, and a number of other tools and products also exist for managing user identity attributes[2], one of the primary differences between the CLASSe solution and the other ones, is that our VO role management is built directly into the OpenStack application, rather than being separate from it. There are a number of advantages and disadvantages of this approach. VO systems that are separate from applications tend not be used, since they require the applications to be modified to access the VO system. Since the CLASSe VO solution is an integral part of Keystone, it has to be used for federated access to the OpenStack cloud, so gaining users is not an issue. Once a critical mass of users start to use the OpenStack VO tools, we believe it could become the VO of choice for other applications as well. The disadvantage of the current implementation, is that other applications currently cannot use it, since there was insufficient time to develop the APIs for this in CLASSe. However, this could be the topic for a follow-on project.

Keystone uses a variant of role based access control (RBAC) to authorise users to access OpenStack resources. Users are assigned to roles in projects. When a user logs into Keystone, the first thing he/she must

[2] A comparison of all the existing solutions can be found here: https://docs.google.com/a/kent.ac.uk/spreadsheets/d/1SZGw3Nl9OOTToiuHibv3b0zKteU-xlaMj1QNm225J78/

do is choose the project they wish to work on. The correct roles are then assigned to the user. Roles are globally named in OpenStack, so it is not possible for different VOs to have the same role names within an OpenStack installation, since all role names must be globally unique. However, Keystone also has the concepts of groups and domains. Groups are simply sets of users who can be assigned the same role by assigning the role to the group. Domains are administrative boundaries for the management of entities within Keystone. A domain can represent an individual, a company, or an operator owned set of OpenStack resources. Whilst domains are globally named, groups and projects are named on a per domain basis. We can therefore map the VO concept to Keystone concepts by equating a VO to a domain, and a VO role to a group. In this way, each VO in Keystone will be uniquely named (by having the domain name), and different VOs can have the same or different VO role names (by having group names). By assigning roles in projects to groups, we can give the various VO roles the different privileges they need in OpenStack. To summarise:

- A VO becomes a Keystone domain. Each VO/domain must be uniquely named.
- A VO role becomes a Keystone group. Each VO role/group is named per domain.
- Groups are assigned to roles on projects, in order to gain the necessary privileges in OpenStack. In this way each VO role member will inherit the correct set of privileges in OpenStack.

The VO administrator must set up the appropriate mapping rules for federated users, so that their identity attributes are mapped into the appropriate group(s) and domain(s). He/she must also assign the correct roles and projects to each group. A full guide to setting up Keystone to support federated access can be found here:

```
http://docs.openstack.org/developer/keystone/configure_federation.html
```

Details about how this is done are described below.

5.1 Mapping Rules for VO Roles

There are two aspects to setting up mapping rules

i) Adding a mapping rule

ii) Adding an IdP and linking it to a protocol and mapping rule

The above are traditionally set up via the Keystone API and are part of the core release. Consult [KEYSTONEFED] for more details.

However, we have added code to Horizon to make most of the above process significantly easier via GUI screens. Note however, our modified Horizon is not part of the core release, and the work was not fully completed during the CLASSe project, so you may have to revert to the Keystone API in order to complete some of the tasks.

The mapping rules are a JSON structure comprising a list of a subset of the remote (identity) attributes provided by the Apache plugin, and the local Keystone attributes that they should map into (in our case this is a group).

If the members of a VO have well-known sets of identity attributes that are issued by their respective IdPs e.g. they are members of particular groups in particular organisations, then it is relatively easy for the VO administrator to create the appropriate mapping rules in Keystone.

For example, the following mapping rule says that users from the Computer Science department of the University of Kent are members of the group `0a4182fc920e4cc699445ac66ffeca41`.

```
[{"remote": [{"type": "organisation", "any_one_of": ["University of Kent"]},
{"type": "organisationalUnit", "any_one_of": ["Computer Science"]},],
"local": [{"group": {"id": "0a4182fc920e4cc699445ac66ffeca41"}}]}]
```

and the following mapping rule says that every remote user that is authenticated by a particular IdP (to which this mapping rules is assigned), regardless of his user ID, is to be given the user name 0 and be a member of the group 097d1d0dff9c46beaaf36293b0175f21

```
[{"local": [{"user": {"name": "{0}"}}, {"group": {"id":
"097d1d0dff9c46beaaf36293b0175f21"}}], "remote": [{"type": "REMOTE_USER"}]}]
```

Note that Keystone automatically assigns unique IDs to groups (i.e. VO roles) when they are created, and it is the unique group ID, rather than the group name, that is used in the mapping rules. This is because group names are not unique within Keystone, but are domain specific.

A problem arises when the members of a VO role are not all the members of a specific IdP group e.g. not everyone from the Computer Science department of the University of Kent is a member of a particular VO, or the IdP does not assert the group memberships of the various VO role holders. This is typically the case. Consequently another way must be found of identifying the VO role holders. Since each user who logs in via an IdP is typically assigned a unique user ID by the IdP, and this value is sent by the Apache plugin to Keystone in the REMOTE_USER environmental attribute, then the REMOTE_USER attribute can be used to identify each user. In the case of ABFAB, the user ID is of the form userID@domain e.g. dwc@kent.ac.uk. So this could be used in the mapping rules as follows:

```
{"local": [{"group": {"id": "3cad8560cdd14877aefa9df842feb3b9"}}], "remote":
[{"type": "REMOTE_USER", "any_one_of": ["dwc@kent.ac.uk"]}]}
```

However, the user ID is not always known to the user or to the OpenStack administrator, for example, the persistent ID assigned by a Shibboleth IdP is algorithmically generated and appears to be a random number. Consequently another way must be found to map these users into the correct VO roles (groups). For this, we have introduced VO Roles by Invitation.

5.2 VO Roles by Invitation

The VO administrator creates a VO role (i.e. domain and group) and gives this a secret PIN or password. He then passes the name of the VO (domain name), the name of the VO role (group name) and PIN/password to the VO role holder out of band. This could be via a telephone call, Skype message, secure email or anything else. The invited user then logs into Keystone via his IdP, and asks to become a member of the VO role, quoting the PIN/password with his request. The CLASSe software checks that the PIN/password is valid for this VO role, and if it is, does one of two things. If the VO administrator has set this VO role membership to automatic join, then the CLASSe software automatically creates a mapping rule for this IdP, in which it maps the value of the REMOTE_USER attribute to the group ID (as in the example above). If the VO role membership is not automatic join, then the user's request is put in a pending queue, so that the VO administrator can check who has requested to join the VO role, and then either accept or reject the request. If the PIN/password is wrong, this is recorded against the user's ID, and if the user attempts three (a configurable number) false attempts to join a VO role, he is blacklisted. The user can then no longer manage his VO role memberships until the VO administrator removes him from the blacklist.

5.3 Assigning Roles and Projects to Groups

This is done via the Keystone API [KEYSTONEAPI]. The administrator needs to run curl, or jmeter, or write a bespoke program that will send an HTTP request to Keystone to add/check/delete a role or project assignment to a group.

In order to add or remove a role to a group, the API is

```
/v3/domains/{domain_id}/groups/{group_id}/roles/{role_id}
```

The HTTP PUT command will assign a role to the specified domain and group. HEAD will check whether a role is assigned to a domain/group and DELETE will remove the role from the domain/group.

In order to add or remove a project and role to a group, the API is

```
/v3/projects/{project_id}/groups/{group_id}/roles/{role_id}
```

The HTTP PUT command will assign the project and role to the group. HEAD will check whether a role and project are assigned to a group and DELETE will remove the role and project from the group.

5.4 Dynamically Managing Small VOs

We define small VOs as those VOs in an OpenStack cloud implementation that has one designated Keystone administrator (or a set of users who have been assigned the admin role in Keystone). The Keystone administrator(s) is (are) able to manage all the VOs and their roles. If the task of managing the VOs and their

roles is too big for the Keystone administrator(s), or you wish to have separate administrators for each VO, then we define this as a large VO, and it should be managed as described in the next section.

1. The Keystone administrator creates the domains and groups (VOs and VO roles) using the Horizon interface (Identity -> VO Admin -> Create VO Role) see Figure 5.1, and assigns roles and projects to them as described above.

2. The Keystone administrator creates mapping rules for the groups as described above.

3. The Keystone administrator adds the IdP(s) and links it (them) to the mapping rule(s) and protocol(s).

In the case where all the VO members will be invited to join by invitation, the only mapping rule that needs to be created is the default rule that gives every user from an IdP a username of 0 and a default group that has no privileges assigned to it i.e.

```
[{"local": [{"user": {"name": "{0}"}}, {"group": {"id":
"097d1d0dff9c46beaaf36293b0175f21"}}], "remote": [{"type": "REMOTE_USER"}]}]
```

This allows the user, when logging in via Horizon, to see the VO Role by Invitation GUI. The CLASSe code will then automatically create all the new mapping rules that are needed when each user requests to join a VO role.

5.5 Scalable Distributed Management of Large VOs

Large scale VOs are managed in a two stage process. Stage 1 involves the Keystone administrator(s) creating a new role, say VOadmin, assigning the appropriate permissions to it, and inviting or mapping the members to this role. Stage 2 involves these new VOadmins managing their own VOs (in a similar way to the Keystone admin does for small scale VOs).

Stage 1.

1. Ensure you have a full set of roles, and that the OpenStack services such as Glance have assigned the appropriate privileges to these roles (this is part of the OpenStack administrator's normal duties).

2. Create a new role, say VOadmin. The easiest way to do this is to use the Horizon GUI.

3. Assign the new role privileges to manage his/her own VO. This requires the creation of a new rule:

   "admin_or_VOadmin": "rule:admin_required or (role:VOadmin and domain_id=%(domainID)s)"

   a. access all the VO APIs

      Edit the Keystone policy.json file to allow the VOadmin role to access the VO APIs for his own VO (i.e. domain) e.g.

"identity:create_vo_role": ”rule:admin_or_VOadmin”

The full set of VO APIs that need modifying by replacing the existing rule with rule:admin_or_VOadmin: `"identity:vo_admin", "identity:list_vo_roles", "identity:get_vo_role", "identity:create_vo_role", "identity:delete_vo_role", "identity:update_vo_role", "identity:add_user_to_vo_role", "identity:list_vo_roles_members", "identity:get_vo_role_member", "identity:remove_vo_role_membership_from_user", "identity:switch_vo_role_for_user", "identity:list_vo_requests", "identity:decline_vo_request", "identity:approve_vo_request", "identity:get_vo_blacklist", "identity:remove_user_from_blacklist”.`

The following VO APIs are already access controlled so that all users can access them, so the rules do not need modifying: `"identity:join_vo_role”, "identity:list_my_vo_roles", "identity:check_vo_membership_status", "identity:resign_from_role”.`

b. manage groups (in your domain)

Edit the Keystone policy.json file to add the rule:admin_or_VOadmin to the group APIs i.e.
`"identity:get_group", "identity:list_groups", "identity:list_groups_for_user", "identity:create_group", "identity:update_group", "identity:delete_group”, "identity:list_users_in_group", "identity:remove_user_from_group", "identity:check_user_in_group", "identity:add_user_to_group".`

c. assign roles to your groups

Edit the Keystone policy.json file to add the rule:admin_or_VOadmin to the grant APIs i.e.

`"identity:check_grant", "identity:list_grants", "identity:create_grant", "identity:revoke_grant”.`

4. Create a VO i.e. domain e.g. MyFirstVO. Perform a POST operation on /v3/domains, with the following json body:

```
{
   "domain": {
        "description": "-my first VO-",
        "enabled": "true",
        "name": "MyFirstVO"
    }
}
```

5. Create a group for the administrators of this VO e.g. FirstVOAdmins using the Keystone API to POST to /v3/groups

```
{
    "group": {
        "description": "--optional--",
        "domain_id": "--optional--",
        "name": "..."
    }
}
```

Note. Steps 4 and 5 can easily be performed using the create VO Role GUI under the VO Admin tab of Horizon.

6. Assign the VOadmin role to this group as described in Section 5.3

7. Add mapping rules to add the members of the VOadmin as described in Section 5.1. This can either be done directly, if the unique identity attributes of the members are known, or via the Invitation mechanism if their unique IDs are not known. If using VO Roles by Invitation, and the IdP is already configured, then no new mapping rule will be needed as the default rule is sufficient to allow the VO admin members to subscribe themselves.

The above grants the VOadmin role permission to manage VOs and VO roles (create, delete, list members etc.), with each VOadmin member only being able to manage his/her own VO, because he/she has to be in the same domain as the VO roles they are managing.

Stage 2.

The VO administrator creates the VO roles. This is most easily done via Horizon (Identity -> VO Admin -> Create VO Role). See Figure 5.1 below:

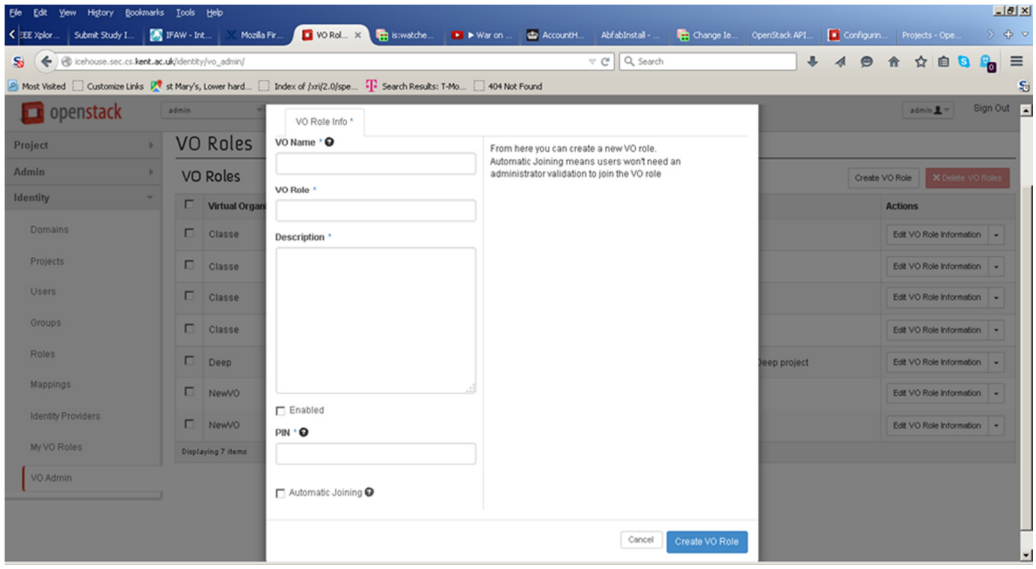

Figure 5.1: Creating a VO Role.

5.6 Trials of Administering Dynamic VOs

In order to help anyone to test and install the software developed within WP4, we have created a guide that provides step-by-step installation instructions [GUIDEJUNOSERVER]. This guide is available in the CLASSe web page (www.um.es/classe/docs.html).

In task T4.3 we have tested the latest software produced as a result of tasks T4.1 and T4.2. In particular, we have focused on the creation and management of dynamic VOs, verifying that the functionality works as expected. For the testing process we have followed this process:

1. To create a completely new VMs at the University of Murcia for the Keystone server and horizon.

2. To follow the steps detailed in the installation guide to create a new instance of the Keystone server (Juno version) and Horizon with support for VO management.

3. To follow the steps detailed in the installation guide to perform the initial configuration and permissions for the VO management.

4. To verify that we can create and administer VOs from the dashboard of Horizon.

The following snapshots depict the process of how the cloud administrator creates a new VO role, and how a demo user joins that VO.

1. First, the admin logs in Horizon, goes to the VO Admin section, and clicks on Create VO role.

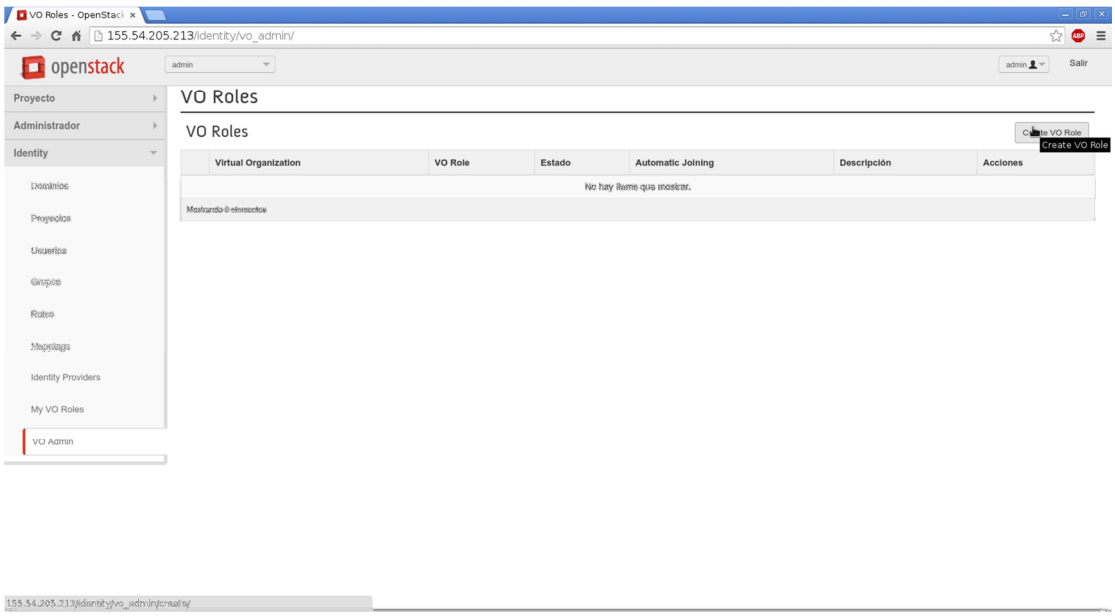

Figure 5.2: VO management. Admin enters VO Admin section.

2. Horizon shows a form where the admin sets the new VO role parameters. In this case, the VO is called VOTEST, the role is called member, and the PIN is set to 1234. The VO role is activated, but automatic joining is not (meaning that the administrator will have to approve each user joining request).

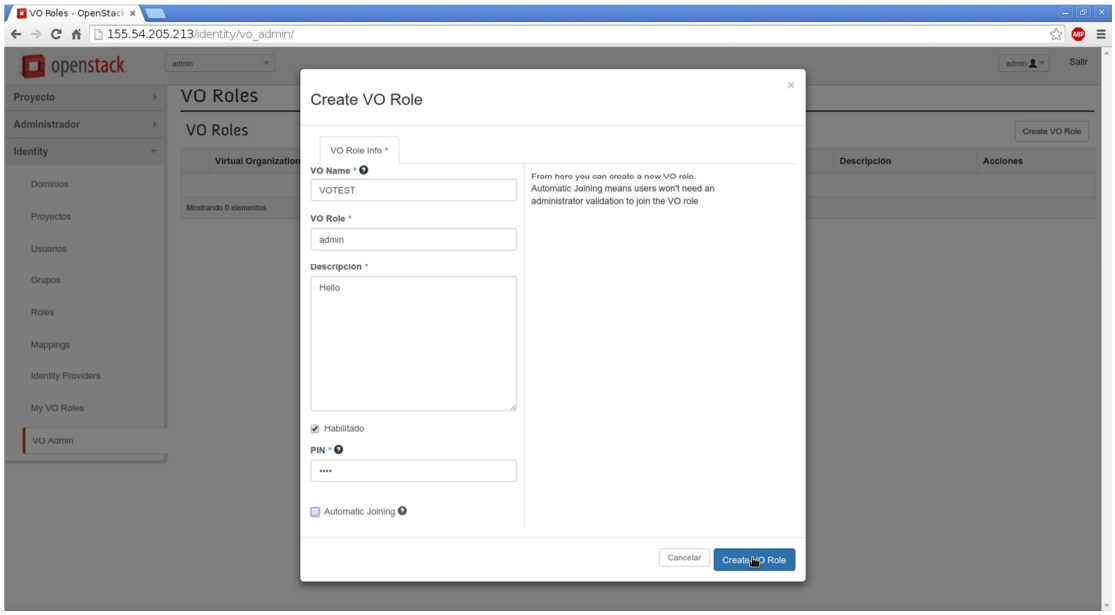

Figure 5.3: VO management. Admin creates VO role.

3. Then, the demo user logs in (e.g. through an ABFAB authentication in Horizon thanks to the work in WP2) and goes to the My VO roles section. In that section, he clicks on Join Request button placed on the top-right corner.

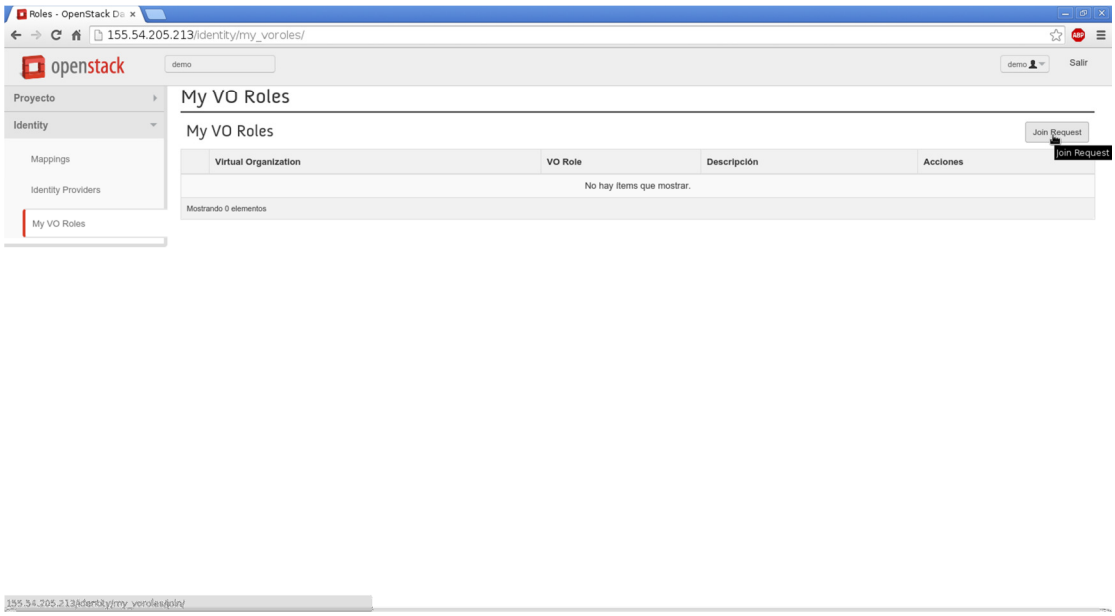

Figure 5.4: VO management. Demo user enters My VO Roles section.

4. Upon this request, Horizon shows a form where the demo user introduces the parameters of the VO role he wants to join.

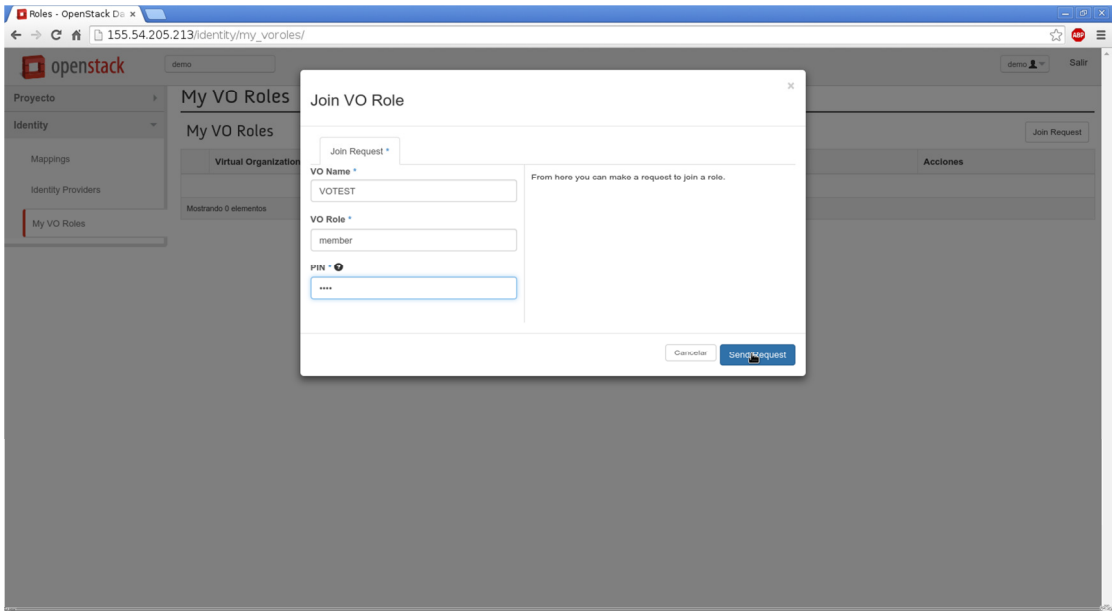

Figure 5.5: VO management. Demo user requests joining the VO role.

5. The demo user cannot see the VO role listed until the admin user approves the join request. The admin can do so in the VO Admin section.

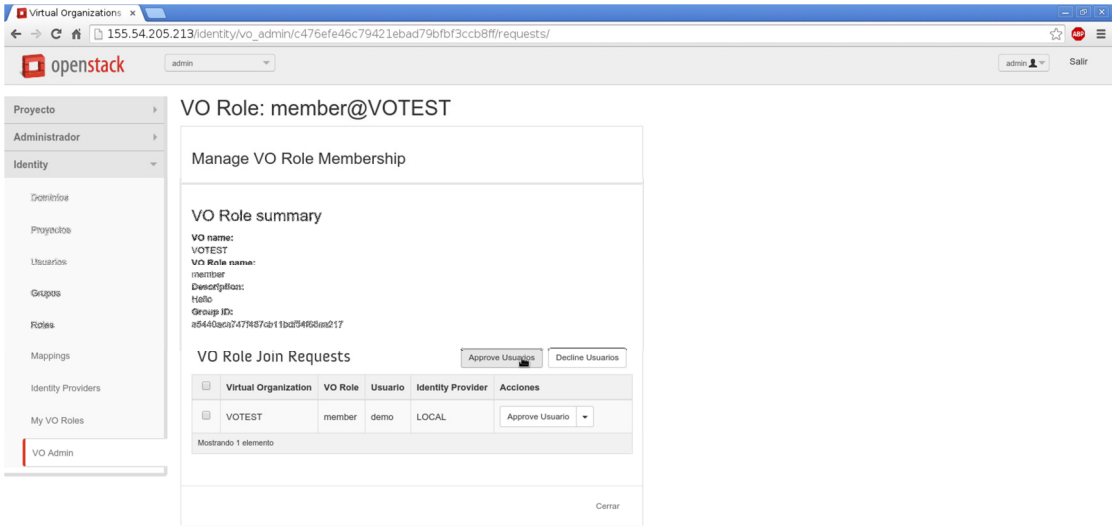

Figure 5.6: VO management. Admin approves user's request.

6. Finally, the demo user will see the VO role listed under the My VO section.

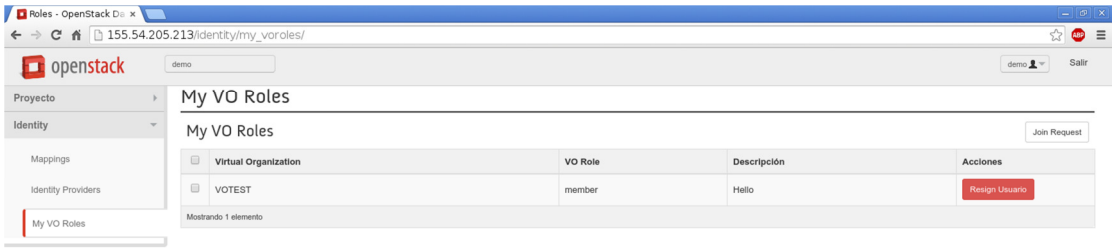

Figure 5.7: VO management. Demo user sees the VO role in the My VO Roles section.

The following enumeration describes how to create a new VO called *VOUMU*, a VO role called *administrator*, and how to grant that role with permissions to manage the VO (i.e. create new VO roles, approve join request…):

1. First, we assume the installation guide [GUIDEJUNOSERVER] has been properly followed, and the VoAdmin OpenStack role exists, and it has been assigned with the proper permissions.

2. Then, the admin starts the process by logging in Horizon, going to the *VO Admin* section, and clicking on *Create VO role*.

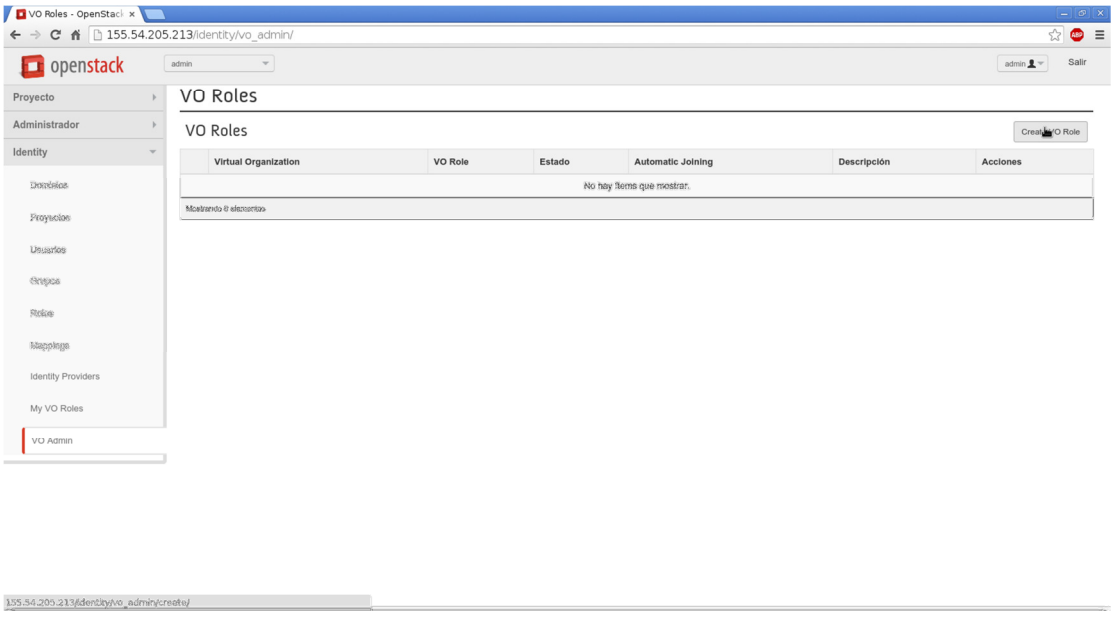

Figure 5.8: Large scale VO management. Admin selects creating a new VO role.

3. The admin creates a new VO role called *administrator,* within a new VO called *VOUMU.* We will refer to this VO role as *administrator@VOUMU*.

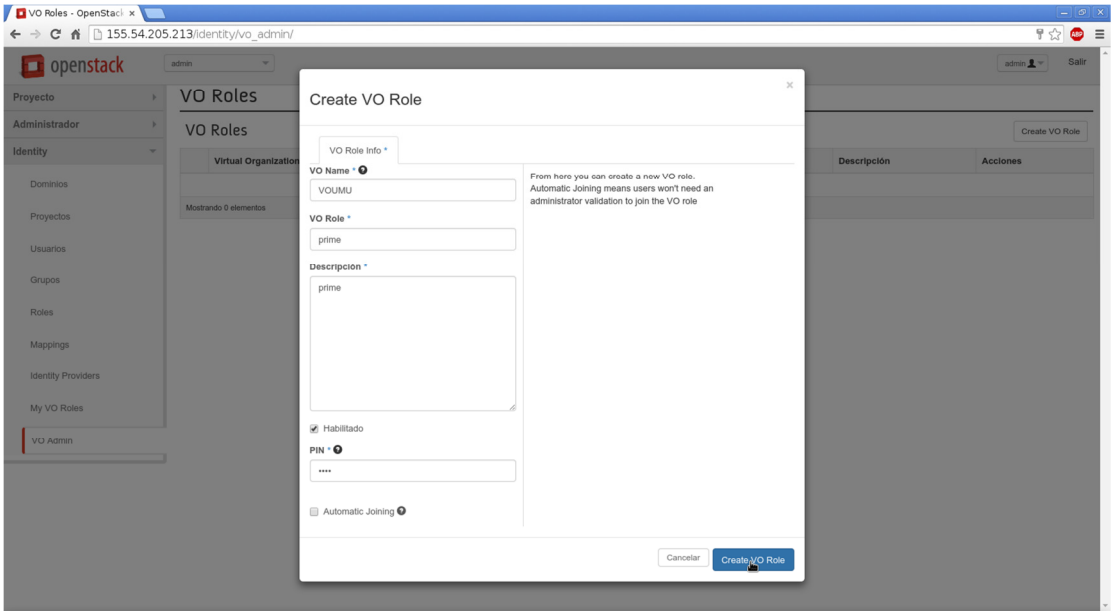

Figure 5.9: Large scale VO management. Admin creates new VO role.

4. Once the VO role has been created, the OpenStack admin needs to follow the instructions on section 4.1 of [GUIDEJUNOSERVER] in order to assign the *VoAdmin* OpenStack role to the recently created *administrator@VOUMU* VO role. This process consists on the execution of a set of SQL commands described in the guide.

5. The admin provides the VO role name and the PIN code to the demo user. For that he can use any means of communication he wants (e.g. email, phone call…).

6. The demo user logs in Horizon, goes to the *My VO Roles* section, and clicks on *Join Request*.

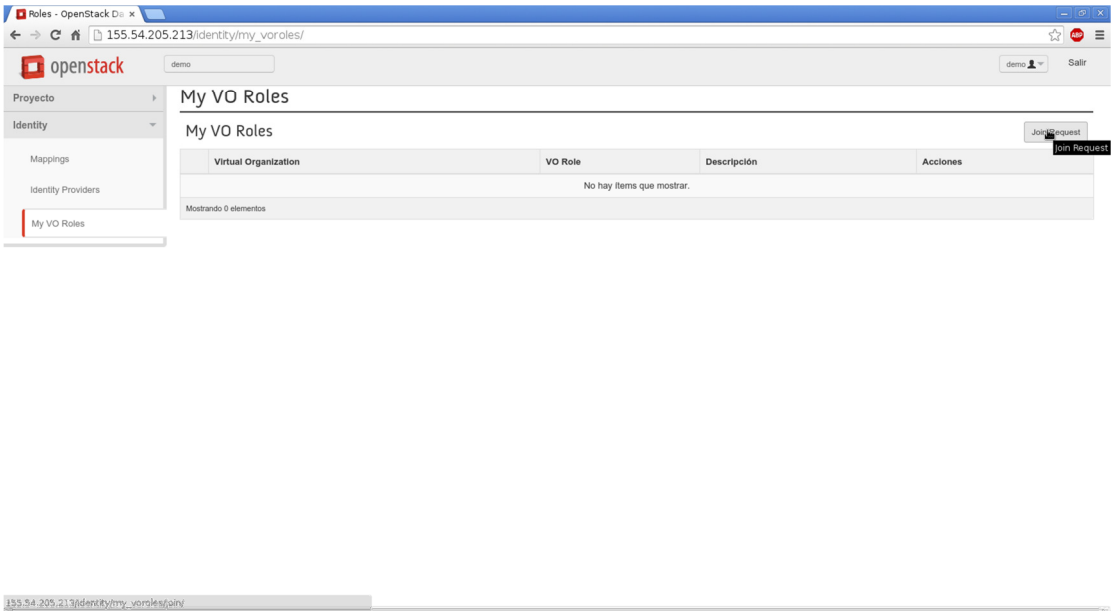

Figure 5.10: Large scale VO management. Demo user selects joining a VO role.

7. Then he introduces the VO role information and sends the join request.

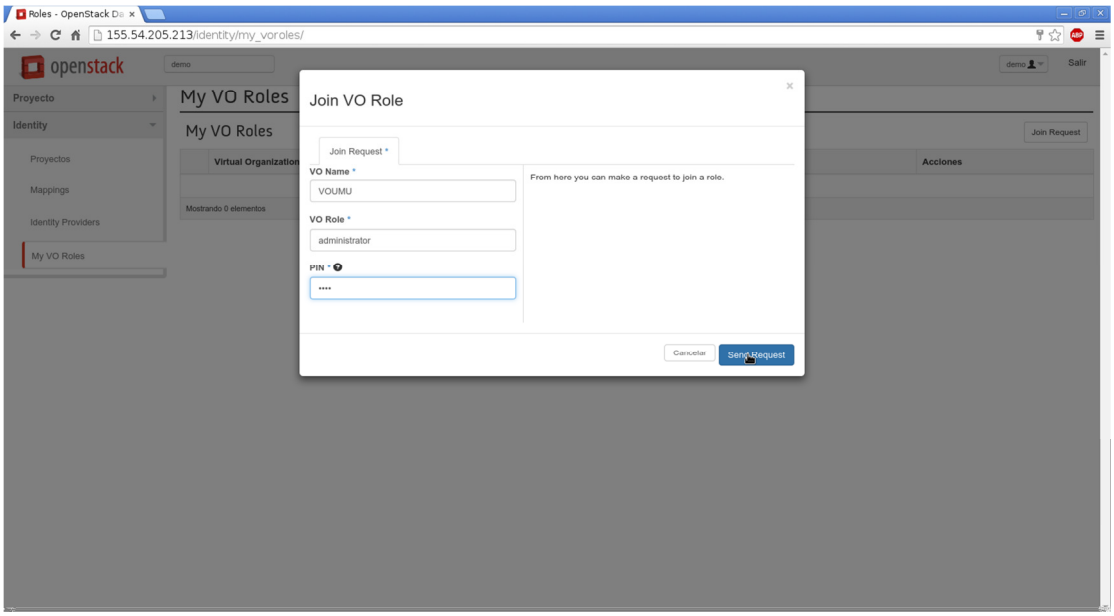

Figure 5.11: Large scale VO management. Demo user requests joining the VO role.

8. At this point, the admin can approve the request.

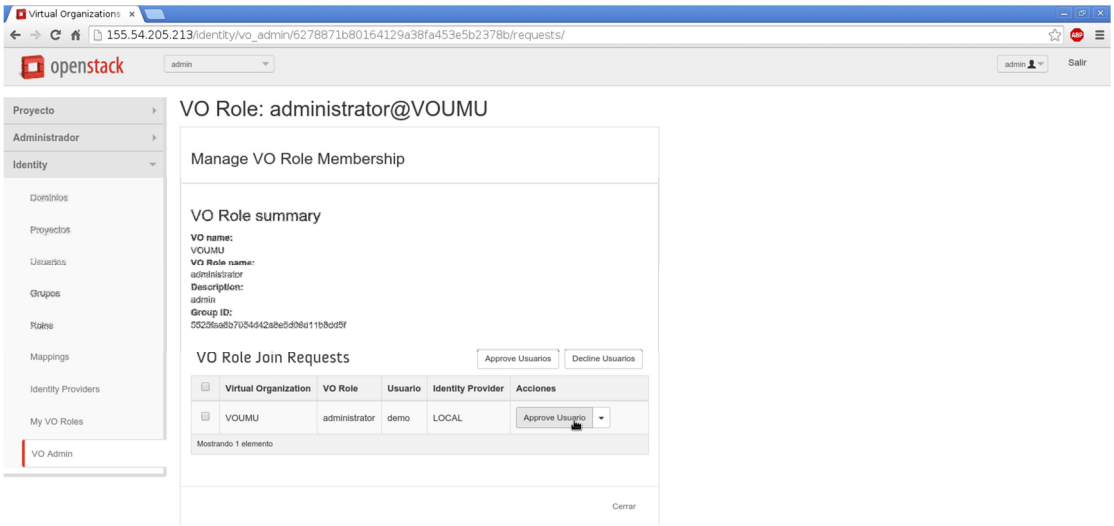

Figure 5.12: Large scale VO management. Admin approves join request.

9. Finally, the demo user can log in Horizon and the VO Admin menu option will be available. Using this menu entry he can administer the VO (create new roles, approve join requests, etc.) as described at the beginning of this section.

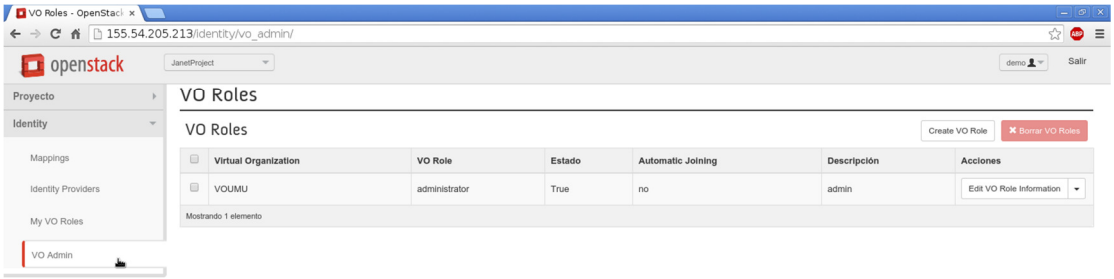

Figure 5.13: Large scale VO management. Demo user has the VO Admin menu entry.

The conclusions extracted after this testing process are the following:

- The installation guides are very complete, clear and correct. It has not been difficult to follow most of the installation process with no mistakes or omissions. The process almost consists on copying commands from the guide and pasting them into the server terminal session, with just a minor number of customization to accommodate IP addresses and keys.

- The software works as expected. We were able to create, join and manage dynamic VO Roles, as well as assign the VoAdmin role to other users so that they can manage the VOs.

- The installation process is not trivial. It requires the system administrator to perform a considerable amount of steps before completing the process. Nevertheless, that is somewhat expected when dealing with proof-of-concept implementations, which typically require a number of small hacks to make them work before their base code is accepted into the mainline branch.

- The process of assigning the VoAdmin role to a VO role requires of writing a couple of SQL commands in the terminal. This functionality requires further work so it can be done from the Horizon web interface.

We therefore consider the software and the installation guides are ready to be used by the partners of the GÉANT community that are willing to test this software, but only as a proof-of-concept.

5.7 Conclusion

The end result of our VO management is that an OpenStack Keystone administrator can now set up VO role management functions that will allow him/her to invite individual users from individual IdPs within a federation, to join his/her VO (or community of interest), and to give them appropriate privileges within his/her OpenStack service. Both small scale and large scale VOs are supported. For small scale VOs, the OpenStack Keystone administrator administers all the users him/herself. For large scale VOs, the administrator invites key individuals to become VO administrators, and then delegates to them permission to manage the various VOs. A working prototype is available at the University of Kent (http://icehouse.sec.cs.kent.ac.uk). There was insufficient time in the CLASSe project to get this code incorporated into the main branch code, so that at present, it only exists as an add-on to the Icehouse release of OpenStack. Furthermore not all functionality has user friendly Horizon GUIs to support it, and the Keystone API must still be used for these features. However ongoing MSc student projects at Kent are continuing to add user-friendly GUI functions to Horizon. A future GN4 project could be to integrate the VO code into the main branch of OpenStack, and to add additional functions to Keystone to allow VO roles to be shared between federated clouds and other services.

6 WP5: Dissemination & Exploitation

6.1 Dissemination Plan

The dissemination plan (preliminary version can found in milestone MS2 [MS2]), devised during the formulation of the project, comprised several components:

1. Work closely with the Moonshot community to ensure that traditional and real SSO designs, deployments and tests are aligned with the Moonshot results. This will ensure CLASSe results will be easily integrated in Moonshot code during the CLASSe project or in future work.

2. Work closely with the OpenStack community to ensure that federation is integrated into the core Keystone code in a protocol independent way. In this way, all future ongoing support for federated OpenStack, including support for multiple protocols including ABFAB and SAML, will be carried out by the OpenStack community at large rather than the University of Kent. This will ensure the sustainability of the research carried out in CLASSe, and will ensure that the deliverables of the CLASSe project will far outlive the lifetime of the project.

3. Ensure that the research from the project is published in quality academic journals and research conferences so that the world at large can learn about our accomplishments.

4. Build a project web site to inform the public about the CLASSe project, its progress and its deliverables.

5. Present the results of the project to the academic community via venues such as Networkshop and Terena Networking Conference.

6. Publish the implementation as open source software, preferably via the Moonshot developer's site and OpenStack releases, or failing that, via the CLASSe web site.

7. Participate in the standardisation processes in the ABFAB and RADEXT IETF WGs, and contribute to the Internet Drafts and RFCs that it produces.

8. Participate in concertation meetings with the other GEANT partners.

6.1.1 Dissemination Activities

6.1.1.1 *Moonshot participation*

CLASSe partners participated in the Moonshot developer meetings, providing feedback of the tests being carried out in UMU and Kent's premises, where they are deploying Moonshot's IdP and RP services. These meetings were held every two months approximately.

CLASSe partners also actively participated in the Moonshot mailing list.

6.1.1.2 *OpenStack Participation*

Prof Chadwick and Kristy Siu actively participated in the development of federated Keystone from the start of the project until October 2014 when Siu's contract finished. After that, Ioram Sette, a PhD student from Brazil, on a one year internship with Professor Chadwick, took over from Kristy Siu. Together, they wrote several OpenStack blueprints and API specifications, as well as contributed code towards the Icehouse, Juno and Kilo releases.

Prof Chadwick attended the OpenStack Summit in Hong Kong, 5-8 November 2013, during which the design of federation for the next OpenStack release, codenamed Icehouse, was discussed. He also attended the OpenStack Summit in Atlanta, May 2014, where moving from SAMLv2 federation to protocol independent federation was discussed. Finally both Prof Chadwick and Ioram Sette attended the OpenStack Summit in Paris in October 2014 where VOs were discussed.

6.1.1.3 *IETF meetings participation*

CLASSe partners are attending IETF meetings, either in person or remotely. Prof Chadwick attended the London meeting (IETF 89). The partners plan to attend some of the next meetings as well: IETF 90, 91 and 92. CLASSe participation is focused on ABFAB and RADEXT WGs.

6.1.1.4 *Publications*

Journals and Book Chapters

Please, note that all mentioned journals are ISI JCR indexed.

Already published:

1. David W. Chadwick, Kristy Siu, Craig Lee, YannFouillat, Damien Germonville. "Adding Federated Identity Management to OpenStack". Journal of Grid Computing. Available online Dec 2013 at http://dx.doi.org/10.1007/s10723-013-9283-2.

2. A. Perez-Mendez, F. Pereniguez-Garcia, R. Marin-Lopez, G. Lopez-Millan, J. Howlett. "Identity Federations Beyond the Web: A Survey " IEEE Communications Surveys and Tutorials Vol 16(4) pp 2125-2141.

Tentative publications:

- Future Generation Computer Systems[3]: UMU plans to submit in the next months, a new paper to this journal. This paper will include the main results from WP3.

- Another paper has been planned with the details of the integration of ABFAB in OpenStack. Journal has not been determined yet.

- A book chapter on Federation and Authorisation in Openstack, to be written by Kent, has been accepted by the editors for submission in May 2015.

Conferences and Workshops

Already happened

- Prof Chadwick from the University of Kent attended the 2 day Moonshot workshop at the University of Warwick, 14-15 October 2013. This meeting was hosted by JANET, and was attended by a dozen or so UK universities who are implementing Moonshot, as well as the Moonshot software developers from Painless Security, USA.

- UMU was invited to participate in the "Grupos de Trabajo de RedIris"[4], June 3rd and 4th 2014, Madrid. For 18 years RedIris (Spanish NREN) organizes a national workshop where staff from research and academic organizations present the latest advances in the deployment of Internet services, communication networks, etc. UMU will present CLASSe in this workshop, together with latest results in Moonshot deployment services.

- UMU submitted the paper entitled: "Cross-layer Architecture for Accessing Federated Services After Network Authentication" to the IEEE Cloud Federation Management Workshop as part of the 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014).

- Prof Chadwick gave a presentation entitled "Adding FIM to Openstack" in the Federation Security session at the Open Grid Forum meeting hosted by the University of Oxford, UK, on 16 January 2014.

- Prof Chadwick gave a Masterclass on Federated Identity Management to members of Kent Connect on 19 February 2014. (Kent Connect is an association of all public service organisations in Kent, including the county council, police, fire service etc.) Part of the talk included a description of ABFAB, Moonshot and CLASSe.

- Prof Chadwick gave a presentation entitled "Adding federated identity management to Openstack" at the EPSRC Cloud Computing Security and Identity Workshop, held on 3rd and 4th April 2014, at the National Museum of Computing, Milton Keynes, UK

[3] http://www.journals.elsevier.com/future-generation-computer-systems/
[4] http://www.rediris.es/gt/

- Prof Chadwick gave a presentation entitled “ABFAB + OpenStack (in the Cloud)” as part of the session “Establishing Virtual Organisations and Collaboration Infrastructures” which took place on Monday, April 7, at 3:00pm-4:00pm in Plaza Court 1, at the Internet2 Global Summit in Denver, Colorado.

- Prof Chadwick gave a talk at the Cloud plugfest event in London on Friday 12 December 2014 entitled “OpenStack ABFAB Federation & VO Management (Geant CLASSe project)”.

Planned Conferences

- “CLASSe Project: Improving SSO in the Cloud” by Alejandro Pérez, Rafael Marín, Gabriel López, accepted for presentation at TNC 2015, Porto on 15-18 June 2015.

- “Opening Up OpenStack’s Identity Service” by David W Chadwick, Ioram S Sette, Kristy W Siu accepted for presentation at TNC 2015, Porto on 15-18 June 2015.

IETF Internet-Drafts

- UMU is currently working (in collaboration with Telefónica I+D and Network Radius) in the standardization of a mechanism that allows RADIUS peers to exchange large amounts of authorization data exceeding the RADIUS limit (4096 octets), by fragmenting it across several client/server exchanges. This mechanism is a key point for CLASSe when large authorization data attributes must be exchanged between the RP (i.e. OpenStack) and the IdP. This effort will be an IETF Experimental RFC soon. (https://tools.ietf.org/html/draft-ietf-radext-radius-fragmentation-12)

- UMU has published a new IETF draft describing the CLASSe solution for the real SSO problem with the ERP extension for GSS-EAP. It has been submitted. in the context of the IETF ABFAB WG (https://tools.ietf.org/html/draft-perez-abfab-wg-arch-erp-00).

PhD thesis

Perez-Mendez, A. "Providing Federated Access to Internet Services by means of Kerberos and AAA Infrastructures". Published in January 2015 URL: http://www.um.es/classe/files/A.Perez.PhD.pdf

6.1.1.5 *Project Web Site*

The CLASSe web site is currently accessible at the following link: http://www.um.es/classe. It contains a summary of the main deliverables and project outputs including the publications, demos, presentations, documentation and software downloads.

6.1.1.6 *Standardisation Activities*

As described before, UMU has carried out the following standardization activities:

- IETF Internet Draft: “Support of fragmentation of RADIUS packets”, https://datatracker.ietf.org/doc/draft-ietf-radext-radius-fragmentation.

- IETF Internet Draft describing the CLASSe solution for the real SSO problem entitled “ERP extensions for the ABFAB architecture” , https://tools.ietf.org/html/draft-perez-abfab-wg-arch-erp-00. This I-D is in the context of IETF ABFAB WG.

6.1.1.7 *GEANT Concertation Meetings*

- The partners attended the GEANT meeting in Vienna, 7-9 October 2013.
- UMU attended the GEANT Symposium in Athens, February 2015.

6.1.1.8 *Open Source Software*

Software developed within the CLASSe project has been published as Open Source Software at a variety of locations. The integration of protocol independent federation, including ABFAB, into OpenStack Keystone, is now part of the core OpenStack release and can be downloaded from the OpenStack web site (https://wiki.openstack.org/wiki/Getting_The_Code). Additions to OpenStack that did not make the core release can be downloaded from the CLASSe web site (http://www.um.es/classe/download.html). This includes modifications to various OpenStack command line clients and to Horizon, the web based GUI, as well as modifications to Moonshot GUI and TLS resumption implementation support for traditional SSO and the ERP extensions for GSS-EAP to support real SSO.

6.2 Exploitation

As described in the original proposal, we defined two main activities for the exploitation plan:

"In particular, partners will work with the OpenStack open source project to ensure that the results are included in the core release of OpenStack, so that the thousands of people who download OpenStack regularly, and use it for their operational services, will have federation and ABFAB distributed to them as part of the standard release. This will maximise the impact of this project."

In addition, UMU and KENT are universities that represent a real use case for the deployment of the results of the project. Both partners will promote the exploitation in their respective institutions by trying to establish the first bridge to federate access to cloud services between both institutions.

Regarding the first, as we have introduced in section 1.2 and described elsewhere in the document, the core release of OpenStack (starting from the Juno release, October 2014) now supports the protocol independent federated identity management designed within CLASSe by KENT, and in particular, one of the protocols that is supported is ABFAB. The Kilo release, scheduled for April 2015, is planned to contain the installation instructions for ABFAB federation. At the time of writing, we are currently working on this. Thanks to this activity in the OpenStack community, we consider that activity 1 has been successfully achieved.

Regarding the second, it is always difficult to convince IT departments at the Universities to deploy some experimental software or new research ideas. Nevertheless, during the different tests carried out between KENT and UMU (see [MS3]) we contacted our respective IT departments so they could configure and support the experiments we were carrying out in the context of CLASSe. The configuration tasks carried out in the respective IT departments were not difficult, and we successfully demonstrated access to cloud services between UMU and Kent using the eduroam infrastructure. This has led to an important conclusion: in technical terms, the deployment of the results of CLASSe should not be difficult with more stable and mature implementations. In the UK, Janet has recently announced the first operational ABFAB service, based on the Moonshot software. We have reviewed the documentation for this, and have pointed out to Janet that as it stands there is no incentive for any UK university to migrate from eduroam to ABFAB. This is because the transfer of the all-important SAML assertion from the IdP to the SP for application authorisation, was only marked as "should be supported" in the documentation, rather than mandatory to support. If an application does not receive this SAML assertion, then it cannot make any better authorisation decisions with ABFAB than it can with today's eduroam. Thus in our opinion support for the SAML assertion must be made mandatory. Unfortunately, the University of Kent's IT department (called Information Services) has made the policy decision that it will not be supporting ABFAB in the near future.

7 Conclusions

This section presents the main conclusions derived from the work done in the CLASSe project and the tentative future work for each of the topics covered for the different working packages. We present now these conclusions per working package.

Regarding project management (WP1), all the administrative work has been done regarding the instructions given by the project coordinator and GÉANT administrative staff. The reader may notice that some Milestones were delayed, but all the objectives have been finally achieved after taking the adequate measures. The collaboration between University of Murcia and University of Kent has been fluent and very constructive. In fact, once the project is finished, we plan to continue collaborating in the near future in the publication of some results from the projects.

Regarding the integration of OpenStack and ABFAB/Moonshot to provide federated access to cloud services (WP2), and thanks to the gap analysis carried out at the beginning of the project, some important gaps were fixed from the initial OpenStack/Moonshot (Grizzly), and new usability tools/interfaces have been released to the public. Beside, OpenStack has been extended with a generic authentication and authorization framework allowing the easy integration of advanced authentication mechanisms such as the one proposed by ABFAB. With this solution end users trying to access cloud services provided by visited institutions will be able to gain access to the service making use of their home credentials, in the same way they currently do accessing the eduroam network service. The most important outcome here is that the OpenStack community has incorporated this new protocol independent federation mechanism into the main branch code, ensuring that future support and maintenance of the ABFAB integration will continue in future releases. The solution proposed in CLASSe has demonstrated to work properly and to achieve the original objectives.

Regarding the analysis of traditional and real SSO (WP3), we have analysed and improved the traditional SSO mechanisms proposed by ABFAB/Moonshot. We have also improved the performance time (in terms of computational time and data transfer) of federated authentication and authorisations processes carried out during the access to federated Moonshot services. We have achieved this from different alternatives: improving the traditional SSO by means of the modification of the Moonshot Identity Selector and the use of TLS-resumption; and providing real SSO by means of extending GSS-EAP mechanism with the EAP Re-authentication Protocol (ERP). The results of these improvements show how the use of both TLS-resumption and (especially) ERP provide significant reductions to the process of accessing a cloud service when using ABFAB technologies, and when the use moves from one service to another: cloud-to-cloud and network-to-cloud. In particular, real SSO based on ERP can reduce the overall time required to access to the cloud service in around a 35-39% (even more if we look at the savings in the AAA infrastructure, being able to reach reductions of about 86-100%). In terms of total amount of network traffic generated, the obtained reductions are similar. ERP can provide an overall reduction of around a 32-35%. If we focus on the AAA infrastructure, these reductions can reach the 75-100%.

Regarding the management of Virtual Organization in Cloud Services (WP4), CLASSe has defined how the OpenStack administrator can manage VO roles and to invite end users from different IdPs to join some specific VOs. CLASSe has defined how to deal with VOs in OpenStack for both small and large organizations., University of Kent has been able to implement a proof-of-concept code for OpenStack that would need to be improved to be included as part of the main OpenStack branch code, in the same way as the framework for

external authentication and authorizations mechanisms used by Moonshot. In despite of the end of the project, University of Kent is continuing improving VO management in OpenStack/Horizon.

Regarding dissemination and exploitation (WP5), we have successfully achieve the main objectives defined at the beginning of the project: we have been actively participating in standardizations groups, such as the OpenStack development group and the IETF working groups ABFAB and RADEXT; several Internet-Drafts and one approved-to-RFC proposal have been worked out during the project, several publications in impact journals, book chapters, conferences and workshops have been accepted from both University of Kent and University of Murcia; and the PdD of Dr. Alejandro Pérez Méndez (RA from the University of Murcia) has been read in January 2015. Regarding the exploitation of the results, different testbeds have been defined and interconnected between the premises of both universities, allowing testing the prototypes of each WPs by means of a real network infrastructure. We have also make use of the virtualization services provided by GÉANT through the QALab infrastructure.

The future work regarding WP2 and WP3 is to continue testing the integration of OpenStack and Moonshot and the use of technologies such as ERP, through the GN4 SA5 task, where UMU will participate and where different federated application services will be tested, among them, OpenStack. In the mid-term future, we plan to apply for new GN4 Open Calls in order to both work in the deployment of federated cloud services over the GÉANT network, and to analyse the use of technologies such as the Trust Router to improve this deployment. Regarding WP4, a future GN4 project could be to integrate the VO code into the main branch of OpenStack, and to add additional functions to Keystone to allow VO roles to be shared between federated clouds and other services.

The results of CLASSe have been demonstrated to be feasible for the deployment over a federation infrastructure like the one provided by GÉANT, directly over the current RADIUS infrastructure, providing the authentication of high level applications like another service (in the same way eduroam currently works) or by means of a parallel infrastructure making use of technologies such as the Trust Router. However, at this point we have proof-of-concepts and prototypes, that can be and will be tested by other GÉANT members, but we need to deeply test and deploy these results in order to be able to provide more stable services to the GÉANT community like a ready-to-use service. We hope we had the opportunity to work in the maturity of these proposals somewhere and sometime in the near future of the GN4 project, for example, by means of future Open Calls.

References

This section should come after the Appendices (if present), but before the Glossary. Use SHIFT+RETURN to put a line break in a URL while retaining the correct indentation alignment.

[ABFAB]	https://tools.ietf.org/wg/abfab/ Application Bridging for Federated Access Beyond web
[AMDAHL]	Gene Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", AFIPS Conference Proceedings, (30), pp. 483-485. 1967.
[CLASSEP]	https://intranet.geant.net/JRA0/CLASSe/Shared%20Documents/Original%20proposal/CLASSe-PartB-FINAL-v2.2.pdf CLASSe project proposal.
[CLASSEWEB]	http://www.um.es/classe CLASSe project webpage.
[CLASSEWEBDOC]	http://www.um.es/classe/doc.html CLASSe project webpage documentation section.
[CSI]	R. Marin-Lopez, F. Perñiguez, G. Lopez, and A. Perez-Mendez. Providing EAP-based Kerberos pre-authentication and advanced authorization for network federations. Computer Standards & Interfaces, 33(5):494–504, 2011.
[D1.1]	Management Reports. CLASSe deliverable. March 2015.
[D2.1]	Supporting tools for the Moonshot/OpenStack implementation. CLASSe deliverable. June 2014.
[D3.1]	Proof-of-Concept. CLASSe deliverable. March 2015.
[DEVSTACK]	http://devstack.org/ devstack
[DH]	W. Diffie and M.E. Hellman. New directions in cryptography. Information Theory, IEEE Transactions on, 22(6):644–654, Nov 1976.
[EAP-APPLICABILITY]	S. Winter and J. Salowey. Update to the EAP Applicability Statement. IETF Internet Draft, draft-winter-abfab-eapapplicability-02, October 2011.
[EAP-TTLS]	P. Funk and S. Blake-Wilson. EAP Tunneled TLS Authentication Protocol (EAP-TTLS). IETF Internet Draft, draft-ietf-pppext-eap-ttls-05, July 2004.
[802.1X]	IEEE 802.1X Std., Standards for Local and Metropolitan Area Networks: Port based Network Access Control. IEEE Standards, 2004.
[ERP]	Z. Cao, B. He, Y. Shi, Q. Wu, and G. Zorn. EAP Extensions for EAP Re-authentication Protocol (ERP). IETF RFC 6696, July 2012.
[FEDKEYSTONE]	https://github.com/kwss/keystone/tree/fed-plugin-moonshot Federated keystone

[FEDKEYSTONECLIENT] http://www.um.es/classe/keystoneclient.pdf
Adding Federated Identity Management to OpenStacks's Keystone Client

[FEDSWIFTCLIENT] https://github.com/kwss/python-swiftclient/tree/havana-moonshot-client
Federated swift client.

[FREERAD] http://freeradius.org/
FreeRADIUS

[GSSERPCODE] http://www.um.es/classe/download.html
Patches for the libeap, mech_eap and freeradius in order to introduce ERP support

[GSS-EAP] S. Hartman and J. Howlett. A GSS-API Mechanism for the Extensible Authentication Protocol. IETF RFC 7055, August 2012.

[GSS-ERP] https://tools.ietf.org/html/draft-perez-abfab-wg-arch-erp-00
A. Perez-Mendez, R. Marin-Lopez, G. Lopez-Millan, F. Pereniguez-Garcia. ERP extensions for the ABFAB architecture. IETF Internet Draft, October 27, 2014.

[GSS-KRB] L. Zhu, K. Jaganathan, and S. Hartman. The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2. IETF RFC 4121, July 2005.

[GUIDEHAVANACLIENT] http://www.um.es/classe/swiftclient.pdf
Installation Guide for Federated Swift Client (Havana Version)

[GUIDEHAVANASERVER] http://www.um.es/classe/server_install_guide_1.2.pdf
Installation Guide for Federated Keystone Server (Havana Version)

[GUIDEICEHOUSECLIENT] http://www.um.es/classe/installationkeystoneclient.pdf
Installation and Usage of the Federated OpenStack Client — Icehouse Version

[GUIDEJUNOSERVER] http://www.um.es/classe/juno-server-federation-vo-abfab_20150224.pdf
Installation Guide For the Juno Server software

[JGC] http://dx.doi.org/10.1007/s10723-013-9283-2
David W. Chadwick, Kristy Siu, Craig Lee, Yann Fouillat, Damien Germonville. “Adding Federated Identity Management to OpenStack”. Journal of Grid Computing: ISSN: 1570-7873 (Print) 1572-9184 (Online). Volume 12, Issue 1 (2014), Page 3-27.

[KERBEROS] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). IETF RFC 4120, July 2005.P. Funk and S. Blake-Wilson.

[KEYSTONEAPI] http://developer.openstack.org/api-ref-identity-v3.html
Keystone API.

[KEYSTONEFED] http://docs.openstack.org/developer/keystone/configure_federation.html
Configuring federation in Keystone.

[LIBRADSEC] http://git.nordu.net/?p=radsecproxy.git;a=summary
libradsec

[MOONSHOT] https://community.ja.net/groups/moonshot
Moonshot

[MS3] Analysis and Design of SSO for cloud service
CLASSe milestone. June 2014.

[NM] https://wiki.gnome.org/NetworkManager
NetworkManager.

[NM-DBUS] https://developer.gnome.org/NetworkManager/unstable/spec.html
NetworkManager D-Bus Interface.

[OPENSTACK] http://www.openstack.org/
OpenStack

[PAP] B. Lloyd and W. Simpson. PPP Authentication Protocols.
IETF RFC 1334, October 1992.

[PEAP] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. IETF Internet Draft, draft-josefsson-pppext-eap-10, Oct. 2004.

[QALAB] https://issues.geant.net/jira/browse/QATB
GÉANT QA testbed

[RADFRAG] A. Perez-Mendez, R. Marın-Lopez, F. Pereñiguez-Garcia, G. Lopez-Millan, A. DeKok, and D. Lopez. Support of fragmentation of RADIUS packets. IETF Internet Draft (in WGLC), draft-perez-radext-radius-fragmentation-12, February 2015

[RADIUS-ERP] K. Gaonkar, L. Dondeti, G. Zorn. RADIUS Support for EAP Re-authentication Protocol. IETF Internet Draft, draft-gaonkar-radext-erp-attrs-03, February 2008.

[RADSAML] J. Howlett and S. Hartman. A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for SAML. IETF Internet Draft, draft-ietf-abfab-aaa-saml-09, February 2014.

[RFC3579] B. Aboba, P. Calhoun. RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP). RFC 3579. September 2003.

[RFC6929] A. DeKok, A. Lior. Remote Authentication Dial-In User Service (RADIUS) Protocol Extensions. RFC 6969. April 2013.

[SAML] S. Cantor, J. Kemp, R. Philpott, and E. Maler (Eds.). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0, March 2005.

[SASL-SAML-EC] S. Cantor, S. Josefsson. SAML Enhanced Client SASL and GSS-API Mechanisms. IETF Internet Draft, draft-ietf-kitten-sasl-saml-ec-11.txt. January 2014.

[SURVEY] A. Perez-Mendez, F. Pereñiguez-Garcıa, R. Marın-Lopez, G. Lopez-Millan, and J. Howlett. Identity Federations Beyond the Web: A survey, Communications Surveys & Tutorials, IEEE, vol.PP, no.99, pp.1,1.

[TLS] T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5248, August 2008.

[TRUST-ROUTER] M. Wasserman, S. Hartman. Application Bridging for Federation Beyond the Web (ABFAB) Trust Router Protocol, IETF Internet Draft, draft-mrw-abfab-trust-router-02, February, 2013.

[UBUNTU12] http://releases.ubuntu.com/12.04/
Ubuntu 12.04

[UBUNTU13] http://cdimage.ubuntu.com/xubuntu/releases/13.04/release/
Xubuntu 13.04.

[VODEMO] http://icehouse.sec.cs.kent.ac.uk
Live VO demo.

[WIRESHARK] http://www.wireshark.org
Wireshark

[WPASUP] http://w1.fi/wpa_supplicant/
WPA Supplicant.

Glossary

AAA	Authentication, Authorization and Accounting
ABFAB	Application Bridging for Federated Access Beyond web
API	Application Programming Interface
CLASSe	Cloud-ABFAB Federation Services in eduroam
DSRK	Domain Specific Root Key
EAP	Extensible Authentication Protocol
EAPoL	EAP over LAN
EMSK	Extended Master Session Key
ERP	EAP Re-authentication Protocol
GUI	Graphical User Interface
HOKEY	HandOver KEYing
IdP	Identity Provider
IDaaS	Identity as a Service
IETF	Internet Engineering Task Force
JRA	Joint Research Activity
KDC	Key Distribution Center
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PSK	Pre-shared Key
RADEXT	RADIUS EXTensions
RAG	Red Amber Green
RBAC	Role based access control
RFC	Request For Comments
rMSK	Re-authentication Master Session Key
RP	Relaying Party
SA	Service Activity
SAML	Security Assertion Markup Language
SSO	Single Sign-On
ST	Service Ticket
TGT	Ticket Granting Ticket
UA	User Agent / End user
UI	User Interface
UMU	University of Murcia
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VM	Virtual Machine
VO	Virtual Organisation
VPN	Virtual Private Network
WG	Working Group
WP	Working Package
XML	eXtensible Markup Language