GridCertLib: Use Shibboleth to Access the Grid from Web Portals

P. Kunszt\textsuperscript{1}, S. Maffioletti\textsuperscript{2}, R. Murri\textsuperscript{2}, and V. Tschopp\textsuperscript{3}

\textsuperscript{1}SystemsX, ETH Zürich, Clausiusstr. 45, CH-8092 Zürich, Switzerland
\textsuperscript{2}Grid Computing Competence Centre, Organisch-Chemisches Institut, University of Zürich, Winterthurerstr. 190, CH-8057 Zürich, Switzerland
\textsuperscript{3}SWITCH, Werdstrasse 2, CH-8004 Zürich, Switzerland

Abstract

This paper describes the design and implementation of \textit{GridCertLib}, a Java library leveraging a Shibboleth-based authentication infrastructure and the SLCS online certificate signing service, to provide short-lived X.509 certificates and Grid proxies.

The main use case envisioned for \textit{GridCertLib}, is to provide seamless and secure access to Grid/X.509 certificates and proxies in web portals: when a user logs in to the portal using SWITCHaai Shibboleth authentication, \textit{GridCertLib} can automatically obtain a Grid/X.509 certificate from the SLCS service and generate a VOMS proxy from it.

We give an overview of the architecture of \textit{GridCertLib} and briefly describe its programming model. Application to common deployment scenarios are outlined, and we report on our practical experience integrating \textit{GridCertLib} into the a portal for Bioinformatics applications, based on the popular P-GRADE software.

1 Introduction

Most Grid computing middleware in production use today relies on X.509 certificate proxies \cite{16} for user authentication. This has been an issue when implementing web-based interfaces to Grid computing facilities: in order to generate a proxy, a copy of the X.509 private key is needed together with the passphrase used to encrypt it. However, uploading the public/private key pair to a web portal is undesirable on security grounds. Several solutions and workarounds have been implemented (see Section \textit{An overview of existing solutions} below), but none of them can be considered entirely satisfactory: either on security grounds, or because they require end users to take multiple steps, possibly through different and unrelated user interfaces (e.g. a web portal and UNIX shell commands).

The solution we developed leverages two features offered by SWITCH, the Swiss National Academic Network: SWITCHaai and SLCS. SWITCH has deployed the
SWITCHaai federated authentication infrastructure [24], based on Shibboleth2 [20]. SWITCHaai federates all Universities in Switzerland, plus major research centres and educational institutions. A web service provider (e.g., a portal) requiring SWITCHaai authentication will prompt its users for their home institution and rely on the home institution's identity provider (IdP) for the authentication step. The users will see the familiar login page of the home institution, the service provider will receive a set of parameters and additional metadata on the user to proceed with authentication and authorization [25].

SWITCH also provides the Short-Lived Credential Service (SLCS) and made it available to all SWITCHaai users [22]. SLCS is a web-service that can sign an X.509 certificate online; authentication and authorization to the SLCS service are based on the SWITCHaai Shibboleth system. The online Certification Authority signing SLCS certificate is included in the IGTF bundle [8], so SLCS certificates can be used for any legitimate Grid purpose. This enables any user from a Swiss institution participating in the SWITCHaai Federation to request a Grid-enabled X.509 certificate. It is valid up to 1’000’000 seconds, corresponding to almost 11 days, which is short-lived in comparison to regular one-year certificates issued by other CAs. A SLCS command-line client is also part of the gLite middleware distribution [4].

The last key enabler for our project is the Shibboleth delegation feature, developed in the Shibboleth uPortal project [21]. The delegation is based on the Liberty ID-WSF Enhanced Client or Proxy (ECP) SSO Profile [3], and allows SAML-based authentication for Shibboleth-protected web service providers. Based on the assertion from the web authentication through Shibboleth on the user portal, we are able to call the SLCS service on the user’s behalf using the ECP profile. Delegation, however, is still an experimental feature in Shibboleth, and is expected to become standard in the next Shibboleth version 3.0. For our project, SWITCH has upgraded their Shibboleth Virtual Home Organization identity provider and the SLCS service provider with ECP delegation features.

GridCertLib, described here, is a Java library providing programming interfaces to create a SLCS certificate and a (possibly) VOMS-enabled Grid proxy, given the SAML2 assertion resulting from a successful previous Shibboleth authentication. The main use case envisioned for GridCertLib, is to provide seamless and secure access to Grid/X.509 certificates and proxies in web portals: when a user logs in to the portal using the regular SWITCHaai Shibboleth authentication, GridCertLib can automatically obtain a Grid/X.509 certificate from the SLCS service and generate a VOMS proxy from it. None of these steps requires user interaction (after the initial Shibboleth authentication).

The plan of the paper is as follows: first we provide an overview of similar solutions already implemented in production-grade Grid web portals. In the next Section, we review the requirements that were set for GridCertLib, its actual design and discuss some implementation details. Finally, we report on some deployment scenarios and particularly on the integration of GridCertLib within the P-GRADE portal.

2 An overview of existing solutions

In order to generate a proxy, a copy of the X.509 private key is needed together with the passphrase used to encrypt it. This poses a basic problem in web portals: having direct access to the public/private certificate key pair of a user, although technically feasible, is undesirable on security grounds: intruders getting access to the portal machine would
gain unrestricted access to all of the portal users’ credentials.

Some projects have worked around this issue by submitting to the Grid as a single portal superuser, using credentials of a single entity for all Grid jobs issued through the portal or through special-purpose certificates for automation, like the robot certificates used by the GILDA portal [17]. Robot certificates are X.509 certificates granted to a Portal service or application, rather than a human; users interested in running a certain application on the Grid can log in to the portal, and the portal will operate on the Grid using the robot certificate. This approach has drawbacks:

1. The certificate private key is available on the portal machine, although this can be prevented by using hardware-based protection (e.g., smartcards), as done in the GILDA/GENIUS portal.

2. The use of robot certificates moves the responsibility of user authentication and logging from the CA to the Portal, thus implicitly introducing an additional trust step in the Grid authentication infrastructure. Not all Grid sites and resource providers might be happy with delegating trust this way.

3. It is difficult to provide per-user accounting of computational resource usage: jobs submitted through different interfaces (e.g., portal and command-line) by the same user will be accounted to different end-entities, since all popular Grid middlewares group usage records by certificate subject DN.

The solution adopted in the P-GRADE portal [15] is to have users upload a long-lived proxy to a MyProxy server [12], [14] and authorize the portal software for automated retrieval of short-lived proxies for job submission and data movement [13]. However, this still requires users to deal with many of the complexities of managing X.509-based certificates and command-line tools, which has been found an actual barrier to Grid adoption in less tech-savvy user communities. For the end-user it is a cumbersome complication to use an authentication infrastructure that does not blend with the native web portal authentication system. It interrupts the natural flow of operations in the web user interface, requiring either an additional password (to generate the proxy) or additional command-line operations in order to proceed with Grid job submission and control.

An extension to this mechanism that blends more seamlessly with P-GRADE’s web-based interface has been developed by the UK project SARoNGS (see [18]) in [19]. Clicking a button on the MyProxy web details page redirects the user to a Shibboleth-protected web service, which in turn loads a long-lived proxy into a specific MyProxy server, and fills in the details in the P-GRADE configuration page.

The approach taken in GridCertLib, instead, requires no user interaction: once the web-based Shibboleth login is successfully completed, the GridCertLib code can generate an X.509 certificate through the SLCS service using the web service based ECP delegation, and an accompanying Grid proxy. Details of this process are given in the following sections.

The source code of GridCertLib is available from http://gridcertlib.googlecode.com/ under the Apache Licence 2.0 [1].
3 Design and Implementation

3.1 Architecture overview

GridCertLib was designed to bridge Shibboleth-based and Grid/X.509-based authentication services for Java web-portals; stated design goals were to allow easy integration into any Java portal, and to minimize interaction with the user.

The flow of interaction with the Java portal code and the SWITCHaai services illustrated in Figure 1 was devised in order to accomplish the design objectives:

1. **The portal users log in to the web portal using Shibboleth single-sign-on.** They are authenticated by their home organization’s identity provider IdP; this is all handled transparently by the Shibboleth software. Among the parameters that are returned from this process, GridCertLib only needs the SAML2 assertion, which is exported by Apache’s mod_shib to any proxied web service.

2. **The portal application code calls GridCertLib to obtain a short-lived Grid/X.509 certificate signed by the SLCS CA.** This step requires delegation of the Shibboleth credentials (SAML2 assertion) to the SLCS login service, which is done through the generic ID-WSF ECP Web Service Client developed by SWITCH [2]. After logging in to the SLCS service, GridCertLib proceeds to generate an X.509 certificate and have it signed by the SLCS online CA, using code similar to the one used by gLite’s slcs-init command-
line client [4].

3. The Portal calls GridCertLib to create a regular grid proxy or a VOMS proxy certificate. Here GridCertLib is just a thin wrapper around the regular VOMS libraries, mainly providing simpler facade calls for commonly-used cases.

4. The certificate, private key and proxies are securely stored. Currently, GridCertLib provides methods for persisting certificates and proxies to the filesystem; it is up to the portal to move the files to different storage back-ends (e.g., databases), although it should be noted that most Grid middlewares require the proxy to be in a known location of the filesystem.

As soon as the GridCertLib code completes successfully, valid certificate and proxy are available on the filesystem and Grid operations can proceed. The foreseen usage of GridCertLib into web portals (see Section Deployment Experiences) called for implementation of additional features into GridCertLib:

a. The SLCS and the Grid/VOMS proxy generation functions can be called independently. In particular, proxy generation does not rely on the SLCS generation feature being called first. As a consequence, Grid proxy code does not need Shibboleth authentication to run, it only expects a valid user certificate to be available.

This is a portal user interface requirement: all that is necessary for generating an X.509 certificate is known after SLCS login, but proxy generation requires additional data; namely, names of the VOs a user belongs to. Portals might need to gather this additional data after a user has logged in.

b. The SLCS generation function can be called at any time, as long as the SAML2 assertion resulting from the Shibboleth login process is available.

This is another portal user interface requirement: SLCS login and generation of X.509 certificates can take long times (in a user interface context), so they can be delayed at a later stage or started in an asynchronous thread, in order not to delay the login process.

c. GridCertLib has a generic interface that can be used with any web application or portal. In particular, GridCertLib does not make any assumption on how user data (like the certificates) are represented and/or stored within the portal, except that they can be stored on the filesystem.

Feature a. led us to provide GridCertLib with two main independent entry points, SLCSFactory and GridProxyFactory. An instance of each class is responsible for generating SLCS certificate, resp. Grid proxies. To achieve portal independence, each class constructor takes an explicit list of all the parameters needed for instantiation, although they can also be conveniently provided by a single Java Properties object.

Similarly, for the same goal c. of portal technology independence, GridCertLib certificate and proxy generation functions accept an explicit list of all the required parameters, but provide abridged forms that have common defaults.

Feature b. implies that the SLCS-generation methods in GridCertLib only require the Shibboleth SAML assertion as input. However, the SAML2 assertion can expire long before the Shibboleth session itself does (see Section Creating SLCS Certificates). GridCertLib provides a re-usable servlet RenewAssertion; which can also be used as a model for implementing assertion renewal in the portal code.
3.2 Core Library Implementation

GridCertLib core functions reside in the single Java package ch.swing.gridcertlib; an additional package ch.swing.gridcertlib.servlet provides example servlets (with fully-commented code) that show how the library can be used.

The main package ch.swing.gridcertlib has two public entry points:

- The SLCSFactory class provides the SLCS certificate generation functionality and can store the certificate and its associated private key on the filesystem.

- The GridProxyFactory class creates Globus Toolkit proxy certificates with or without VOMS extensions from available user certificates and stores them to a temporary location on the filesystem.

A single instance of each of these classes can generate multiple SLCS certificates or proxies, possibly for different Portal users, via repeated invocation of the certificate creation methods.

Since the parameters used to configure the factory objects are portal-wide global variables and their values are fixed while the portal application is running, each factory class can be configured (at construction time) through a java.util.Properties object, which can be conveniently loaded from a file with standard Java API calls. Alternatively, a constructor that allows to specify all instance parameters explicitly is also provided.

However, GridCertLib does not enforce that only a single instance of these factory objects exists. Different factory objects can be created to cater for different classes of users (e.g., users coming from different Shibboleth federations). It is up to the web application/portal code to route requests to the correct factory.

3.2.1 Creating SLCS Certificates

Upon calling the newSLCS method of the SLCSFactory class, a SLCSRequestor object is created to carry out generation of the certificate and actual interaction with the SLCS server. The reason for this split is twofold:

- SLCSFactory handles system-wide defaults, and thus a single instance is needed to serve the whole portal, whereas a new SLCSRequestor object is created for every certificate request.

- The SLCSRequestor corresponds to the slcs-init command-line tool provided in the gLite middleware distribution [4]; this eases porting of fixes from the official SLCS client on to GridCertLib.

SLCS certificates are created following these steps:

1. Login to the SLCS server using ECP delegation: if successful, this returns the subject DN to use in the X.509 certificate to be generated, and an authorization token to validate the final certificate signing request to the same SLCS server.

2. Locally generate an X.509 public/private key pair.

3. Locally generate an X.509 certificate signing request (CSR), using the subject DN and other X.509 constraints returned by step 1.
4. Submit the CSR to the SLCS server and get the signed certificate back.

All of the above steps keep relevant data like the password necessary for the private key in memory. Only after a certificate has been successfully generated by the SLCSRequestor will SLCSFactory save the result in a file and return the certificate file path, private key path and private key password to the caller. Any of these three can be defined by the client code by passing an optional argument to the newSLCS method; by default, SLCSFactory uses a random password and stores the certificate and private key files in a configurable directory (using a random file name, which is also returned as a result of the call).

The SLCS service is a properly authorized Shibboleth Service Provider (SP). Since GridCertLib is contacting SLCS on behalf of the user, and with no user intervention at all, delegation of SAML credentials is needed. Shibboleth delegation is an experimental feature available in Shibboleth2, by which the SAML2 assertion that initiates a Shibboleth session on an SP, may be re-used to authenticate towards other web service based SPs. Shibboleth delegation must be supported both on the IdP granting the SAML2 assertion and the target SP receiving the delegated assertion (the SLCS server in this case).

The requirement that SLCS generation may happen at any time after login to the Portal creates an additional complication: SAML2 assertions have a short time validity (5 minutes in the default configuration) but SP and IdP sessions last much longer (8 hours by default), i.e., users are authenticated with the Shibboleth SP even after the SAML2 assertion is long expired. Therefore, by the time SLCSFactory.newSLCS is called, the SAML2 assertion might be unusable for delegated authentication with the SLCS server. There is no support in the Shibboleth API to get a fresh assertion in the IdP and store it in the SP session, but this can be worked around by forcing an SP session logout, followed by a redirection to a Shibboleth-protected URL: the SP will start a new session and request a fresh assertion from the IdP. This can be implemented by a chain of HTTP redirections, so that the whole procedure does not require any user intervention. We implemented this workaround in the RenewAssertion servlet, described in the Example servlets section.

3.2.2 Creating Grid and VOMS proxies

The GridProxyFactory class is a interface wrapper on top of the VOMS Java API. GridProxyFactory implements a simplified interface to create a Grid proxy in the use case most frequently needed in web applications and portals: its newProxy method creates a proxy (with optional VOMS extensions) given an X.509 certificate and private key, and a (possibly empty) list of VOs to contact for VOMS ACs.

A single instance of the class can generate multiple proxies (possibly for different users) via repeated invocation of the newProxy method. Since the org.glite.voms.contact.VOMSProxyInit class uses system properties to determine part of its configuration, it is not possible to create different instances of this class, each using its own configuration. This is not a limit in practice, as the org.glite.voms library has native support for multiple servers and VO endpoints.

3.2.3 Example servlets

The provided sample servlets can run in any Java servlet container. They have been successfully tested with Jetty and Tomcat (with an Apache proxy front-end for managing the Shibboleth session).
**SlcsInit**  The ch.swing.gridcertlib.servlet.SlcsInit servlet extracts the SAML2 assertion URL from the Shibboleth HTTP headers, downloads the assertion into memory, and uses it to authenticate to a remote SLCS service and get a new certificate/private key pair. The key is encrypted with a random password, and the certificate and private key locations (on the filesystem) are printed in the response text.

If SLCSFactory detects an expired assertion in the SP session, it will raise an exception. The SlcsInit code catches the error and redirects the user’s browser to the RenewAssertion servlet, setting the return address to the current page: when the user browser is sent back to the return URL, a new SAML2 assertion will be in the SP cache.

**VomsProxyInit**  The ch.swing.gridcertlib.servlet.VomsProxyInit servlet creates a VOMS proxy and stores it on the filesystem, in the default store directory. HTTP query parameters can set arguments that are passed to the GridProxyFactory.newProxy method, thus making this servlet a generic front-end to the GridProxyFactory class functionality.

This servlet does not require any interaction with the Shibboleth subsystem, and can be deployed unprotected. It requires, however, that the certificate and private key are available on the filesystem.

**RenewAssertion**  The ch.swing.gridcertlib.servlet.RenewAssertion servlet ensures that a fresh assertion is stored in the SP session cache. It implements the workaround described in a previous section for the “expired assertion” problem:

1. The user’s browser is redirected to the SP session logout URL.
2. The logout function allows setting a “return address” via a URL query parameter, to which the browser will be redirected after the logout is done; this “return address” is set to the RenewAssertion URL plus a trailing component (URL “path information” part) that encodes the referring page URL.
3. The Shibboleth SP logs the user out of the session and destroys the cached data, then redirects the user browser to the RenewAssertion URL.
4. The RenewAssertion page is Shibboleth-protected, so a new Shibboleth authentication procedure begins. As long as the user session in the IdP is still valid, this will not require user interaction, and the IdP will just send a new SAML2 assertion to the requesting SP.
5. The RenewAssertion servlet detects that the browser is returning after the initial visit (from the trailing portion of the URL), and redirects the user to the initial requesting page (by decoding the URL embedded in the “path information” component).

Note that none of the above steps requires any user interaction (unless the Shibboleth session on the IdP is expired).

A request URL to RenewAssertion must be properly formatted; the convenience method RenewAssertion.getRenewalUrl is provided to this purpose. However, the URL encoding system in the RenewAssertion servlet imposes a limit on the length of return URLs; more importantly, it cannot be used with HTTP POST requests, as there is no way of encoding the POST data into a single URL. This is a technical issue which we have not been able to work around so far: due to the large number of HTTP redirects...
taking place, session cookies, query parameters, and other commonly-used ways of associating state data with HTTP requests do not survive until the final visit to the RenewAssertion servlet.

4 Deployment experiences

The following points need to be taken into consideration by the portal providers:

- Since certificate generation can be time-consuming (relative to user interface reaction times), it could be delayed to a later stage or executed asynchronously in a separate thread.

- The validity of the Shibboleth assertion is usually limited to a few minutes, so the SLCS certificate request should not be delayed for too long. Of course if a valid SLCS certificate for the user is already available from a previous login of the user, the request can simply be omitted.

- When a VOMS-enabled proxy is needed, it is the Portal’s responsibility to prompt the user for the relevant information (VO name or FQAN list). In the P-GRADE implementation, the users can set their VO(s) in their settings page. There is a global default configuration option for the administrator if every user is expected to be always member of the same VO.

4.1 Generic deployment using the provided servlets

The example servlet can be used in a generic deployment scenario, that requires no modification to the portal code and minimal edits to tune the servlet output to the interface expected by other portal components.

Assuming the target portal has already a Shibboleth-enabled login procedure, the following steps would provide Grid certificate functionality using GridCertLib:

1. Deploy the SlcsInit and VomsProxyInit servlet at a publicly-accessible URL. These URLs should be Shibboleth-protected as well.

2. Modify the user status page to issue a visit to the SlcsInit servlet; HTTP query parameters should be used to tell SlcsInit to store the certificates in a known location.

   In addition, the SlcsInit code needs to set a password on the X.509 private key. The password can either be set by the supporting Portal code through HTTP query parameters or cookies, or generated by some JavaScript code embedded in the calling page, or chosen at random by SlcsInit. In the latter case, SlcsInit must have a way to communicate this password to the VomsProxyInit servlet when it’s called later on; the simplest approach to solve this problem is to use HTTP cookies.

   Depending on the actual mechanism chosen to have the user browser visit the SlcsInit URL (e.g. JavaScript’s XMLHttpRequest, HTML IMG tag, etc.), the output of SlcsInit might need modification in order to be further processed or displayed directly on the web page: the source code of SlcsInit has been kept simple enough to allow easy adaptation.
3. Similarly, modify the “job submission page” to visit the VomsProxyInit servlet: HTTP query parameters or cookies should be used to inform the servlet code about location of the certificate and private key, which where obtained in the SlcsInit step, and to set a known location for the proxy file.

Parameter passing through cookies is recommended, as it allows the VomsProxyInit code to directly get the password from the SlcsInit servlet without any supporting code in the portal.

The GC3 Computational Chemistry portal provides an example of integration of the GridCertLib servlets into a portal based on the Python framework Django. Work is currently underway as of this writing and code has not yet been released to the general public.

4.2 Integration into the P-GRADE portal

The P-GRADE portal comes already with full Grid X.509 proxy support, which in our case is a mixed blessing as many of the certificate management features need to be modified in various places of the portal code. Out of the box, P-GRADE supports proxy certificate upload or the usage of a MyProxy server to which the user has to upload the certificate outside of P-GRADE.

In its standard form, P-GRADE provides no facilities for the creation of the certificates; this is a new feature we add using GridCertLib. We extended the Shibboleth-enabled login [7] for the Gridsphere portal [6] (provided by the Australian MAMS project [11]) by storing all Shibboleth attributes including the assertion and other attributes that were not requested previously into a singleton object.

In the MAMS implementation, on first-time login using Shibboleth, the user is presented with a registration request portlet which simply displays the attributes of the user as received through the Shibboleth login by the server. Users can then simply press a button “Send registration request”, which triggers an email to the portal administrator, who can decide whether to enable the user account, and optionally assign it certain roles in Gridsphere.

Users can simply reload the page or re-login once the admin has enabled them. At the same time it is checked whether an SLCS certificate still exists for the given user that is valid for longer than 24 hours. If not, a separate thread is spawned and GridCertLib is used to create a new SLCS certificate using the singleton object. The certificate location and other related information is stored together with all other user attributes in the user table, which has been extended accordingly.

The singleton is used again when the user clicks on the “Security” tab. A button has been added, to renew the Grid/X.509 certificate based on an existing SLCS certificate, whose validity period is also displayed. The functionality to upload a proxy or to get it from a myproxy server has been retained. The VOMS configuration is the same for all users of the portal in our current implementation.

An issue remains that the delegation feature we make use of is not yet deployed as a standard feature in the Swiss AAI federation, therefore we currently can only make use of this whole mechanism through a special home organization, namely the “Virtual Home Organization” (VHO) as provided by SWITCH for collaboration purposes. We have a dedicated group in the VHO where we can administer our own users. This should not be necessary anymore after the Swiss AAI federation has upgraded to a version of Shibboleth that supports delegation, which should happen sometime in 2011.
For now, in the optimal case a user can log in through AAI by selecting the VHO as the “home organization”, click on the “Security” tab, request a Grid Proxy (if not there already) and then click on any of the other tabs to start using the Grid.

5 Conclusions and Future Developments

GridCertLib is an easy to use Java library that enables automatic creation of SLCS certificates and/or Grid proxies from SAML2 assertions obtained from successful Shibboleth authentication. It can be integrated into real-world Grid portals, hiding the complexities of X.509 certificate usage from the portal user. This considerably lowers the barrier to Grid usage, allowing much larger communities to profit from Grid resources securely. Source code for GridCertLib is publicly available from http://gridcertlib.googlecode.com/ under the Apache Licence 2.0 [1].

The current implementation of GridCertLib relies on three key features of the SWITCH AAI infrastructure: Shibboleth authentication, ID-WSF ECP delegation, and the SLCS online CA service. The integration of these three components together with a valid access to a VOMS server, allow the creation of any community specific web portal that can leverage the national grid computing infrastructure SMSCG [23] thus enabling Grid use by virtually any Swiss scientific community.

GridCertLib has already been successfully deployed and integrated into a Systems Biology portal based on P-GRADE portal, and into a Python-based Computational Chemistry portal, proving the flexibility and re-usability of the library and its design.

We will assist in the integration of GridCertLib into portals that are in use in Switzerland, like JOpera [9] and the new WS-PGRADE [29]. We will consider requests for extensions in functionality of the GridCertLib based on the experience with these new portals.

Looking further into the future, GridCertLib will greatly profit from the upgrade of the Swiss AAI federation to the next version of Shibboleth: this will enable true single-sign on and Grid usage in one portal, without the need to use a special VHO account. The SystemsX project SyBIT [27] also plans to upgrade its P-GRADE portal from the current Gridsphere-based implementation to the more modern WS-PGRADE, which makes use of the Liferay portal technology: besides many portal-related improvements, this will allow the users to freely choose the VOMS attributes they wish to associate with their proxy. However, due to the entirely new portal code base, a new programming effort will be needed to integrate GridCertLib into the Liferay framework.

6 Acknowledgements

This work was carried out in the context of the Swiss Grid Portal project, funded through the SWITCH-AAA track. We would like to thank all our collaborators in that project —Cesare Pautasso, Frederique Lisacek, Heinz Stockinger— and also all the help we received from the Hungarian Academy of Sciences SZTAKI for the integration with P-GRADE, especially Ákos Balasko.

References


[29] http://www.guse.hu/?m=architecture&s=0