J Grid Computing (2011) 9:441–453
DOI 10.1007/s10723-011-9195-y

GridCertLib: A Single Sign-on Solution for Grid Web Applications and Portals

Riccardo Murri · Peter Z. Kunszt · Sergio Maffioletti · Valery Tschopp

Received: 27 April 2011 / Accepted: 16 September 2011 / Published online: 4 November 2011

Abstract This paper describes the design and implementation of *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 *GridCertLib*, is to provide seamless and secure access to Grid X.509 certificates and proxies in web applications and portals: when a user logs in to the portal using SAML-based Shibboleth authentication, *GridCertLib* uses the SAML assertion to 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 *GridCertLib* and briefly describe its programming model. Its application to some deployment scenarios is outlined, as well as a report on practical experience integrating *GridCertLib* into portals for Bioinformatics and Computational Chemistry applications, based on the popular P-GRADE and Django softwares.

R. Murri · S. Maffioletti
Grid Computing Competence Centre,
Organisch-Chemisches Institut, University of Zürich,
Winterthurerstrasse 190, 8057 Zürich, Switzerland

R. Murri
e-mail: riccardo.murri@gmail.com

S. Maffioletti
e-mail: sergio.maffioletti@gc3.uzh.ch

P. Z. Kunszt (✉)
SystemsX, ETH Zürich, Clausiusstrasse 45,
8092 Zürich, Switzerland
e-mail: peter.kunszt@systemsx.ch

V. Tschopp
SWITCH, Werdstrasse 2, 8004 Zürich, Switzerland
e-mail: valery.tschopp@switch.ch

1 Introduction

Most Grid computing middleware in production use today relies on X.509 certificate proxies [43] 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 2), but none of them can be considered entirely satisfactory: either because they do not fully address the security concerns, 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 federated authentication and authorization infrastructure

SWITCHaai and the Short-Lived Credential Service (SLCS). SWITCHaai is a federated authentication and authorization infrastructure [39], based on Shibboleth2 [18]; SWITCHaai federates all Universities in Switzerland, plus major research centers and educational institutions. Similar nationwide Authentication and Authorization Infrastructures (AAIs) exist in Germany, Denmark, Belgium, the Netherlands and other countries [7]. A web service provider (e.g., a portal) requiring SWITCHaai authentication will delegate the authentication step to the user's home institution Identity Provider (IdP). Users will be prompted with the familiar login page of the home institution; after successful logon, the service provider will receive a set of parameters and additional metadata (about the user) to proceed with authorization [38].

SWITCH also provides the Short-Lived Credential Service (SLCS) and makes it available to all SWITCHaai users [36]. 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 (CA) that signs SLCS certificates is included in the International Grid Trust Federation (IGTF) bundle [17], 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 s (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 [42].

The last key enabler for our project is the Shibboleth delegation feature, developed in the Shibboleth uPortal project [5]. The delegation is based on the Liberty ID-WSF ECP Single Sign-On (SSO) profile [16], and allows SAML-based authentication for Shibboleth-protected web service providers.[1] Based on the assertion resulting 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 Enhanced Client or Proxy (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 is a Java library providing programming interfaces to create a SLCS certificate and Grid proxy (optionally VOMS-enabled), given the Security Assertion Markup Language (SAML) 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* uses the SAML assertion to obtain a Grid X.509 certificate from the SLCS service and generate a Virtual Organisation Membership Service (VOMS) proxy from it. None of these steps requires user interaction (after the initial Shibboleth authentication), making Grid resources as easy to use as any single-sign-on enabled web service while retaining the full security stack.

The outline 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 a Bionformatics portal (based on P-GRADE, [10]) and within a Computational Chemistry portal (based on Django, [8]).

2 An Overview of Existing Solutions

Distributed authentication and authorization is a difficult problem to solve in a standard and integrated manner. In past Grid projects, proprietary services have often been developed to address this issue (e.g. CAS [33], PRIMA [27], ROAM [6]) but none of them established itself as a widely accepted community standard as they were too

[1] The interested reader can find readable introductions to Shibboleth and SAML in [15, 19].

tightly coupled with the middleware and the local resources. A standardization on SAML/XACML profiles to be used by all middlewares is available [11] but has not been widely adopted yet.

However, we can see two technologies that are widely accepted also outside of the Grid community: SAML-based authentication using Shibboleth and X.509 certificates to authenticate local resources, using short-lived proxy certificates. In most Grids, also the VOMS cervice [1] is used to enrich the proxy certificate with usage attributes for fine-grained authorization. We are using the SLCS service as developed by SWITCH for the Enabling Grids for E-sciencE (EGEE) consortium to generate certificates from the users' Shibboleth login. A similar but now defunct project was the U.S. GridShib effort [4] to create a certificate based on a Shibboleth login also as a Certificate Authority.

In order to generate a certificate 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, called "robot" certificates.

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 a few 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 [3]. Indeed, guidelines [9] have been issued by the IGTF on the generation and storage of private keys, and permissible key usage of automated clients (robots) that can hold credentials issued by IGTF-accredited Certification Authorities, so this specific issue is likely to become less relevant in the future.
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 Distinguished Name (DN).

The solution adopted in the P-GRADE portal [10, 23] is to have users upload a long-lived proxy to a MyProxy server [25, 31, 32] and authorize the portal software for automated retrieval of short-lived proxies for job submission and data movement. 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 to be a real barrier to Grid adoption in less tech-savvy user communities. In the newer WS-PGRADE portal [24] the interaction with the MyProxy service has been streamlined so no command-line interaction is necessary, user certificate can be directly uploaded. However, the user still needs to apply for and manage a certificate that expires every year. For the end-user it is a 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 (the certificate 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 in [14]. Clicking a button on the MyProxy web details page redirects the user to a web service (Shibboleth-protected), 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 publicly available from http://gridcertlib.googlecode.com/ under the Apache License 2.0 [2].

3 Design and Implementation

3.1 Architecture Overview

GridCertLib was designed to bridge Shibboleth-based and Grid X.509-based authentication services for web applications and portals.[2] Its design goals were to allow easy integration into any Java portal, and to minimize interaction with the user while retaining the full security stack for Grid authentication and authorization.

The flow of interaction with the Java portal code and the SWITCHaai services illustrated in Fig. 1 was devised in order to accomplish the design objectives (numbers in parentheses correspond to arrows in Fig. 1):

(1) Users initiates log in to the web portal using Shibboleth single sign-on (i.e., request a Shibboleth-protected URL).
(2) They are authenticated by their home organization's identity provider IdP; this is handled transparently by the Shibboleth software.[3]
(3) *GridCertLib* queries Apache's `mod_shib` to get the SAML2 assertion. The assertion is exported, together with other authentication parameters, to any proxied web service; portals may make use of these Shibboleth attributes to restrict certain services or map users into user groups.
(4) 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 Identity Domain—Web Service Framework (ID-WSF) ECP Web Service Client, developed by SWITCH [41].
(5) 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 [42].
(6) The portal calls GridCertLib to create a Grid proxy (with or without VOMS extensions). Here *GridCertLib* is just a thin wrapper around the regular VOMS libraries, mainly providing simpler façade calls for commonly-used cases.
(7) The certificate, private key and proxies are 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 backends (e.g., databases), although it should be noted that most Grid middlewares require the proxy to be in a known location on the filesystem.

As soon as the *GridCertLib* code completes successfully, a valid certificate and proxy are available on the filesystem and Grid operations can proceed.

The foreseen usage of *GridCertLib* in web portals (see Section 4) called for implementation of additional features in *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

[2] Henceforth, we shall briefly write "portal" to mean any web-based interactive application or service.

[3] A detailed understanding of the Shibboleth authentication process is not needed for working with *GridCertLib*: it suffices to know that the outcome of a successful Shibboleth logon is a Security Assertion Markup Language version 2 (SAML2) assertion stored in the web server on the Portal machine. The interested reader is referred to [15, 19, 38] for an introduction to Shibboleth and SAML2.

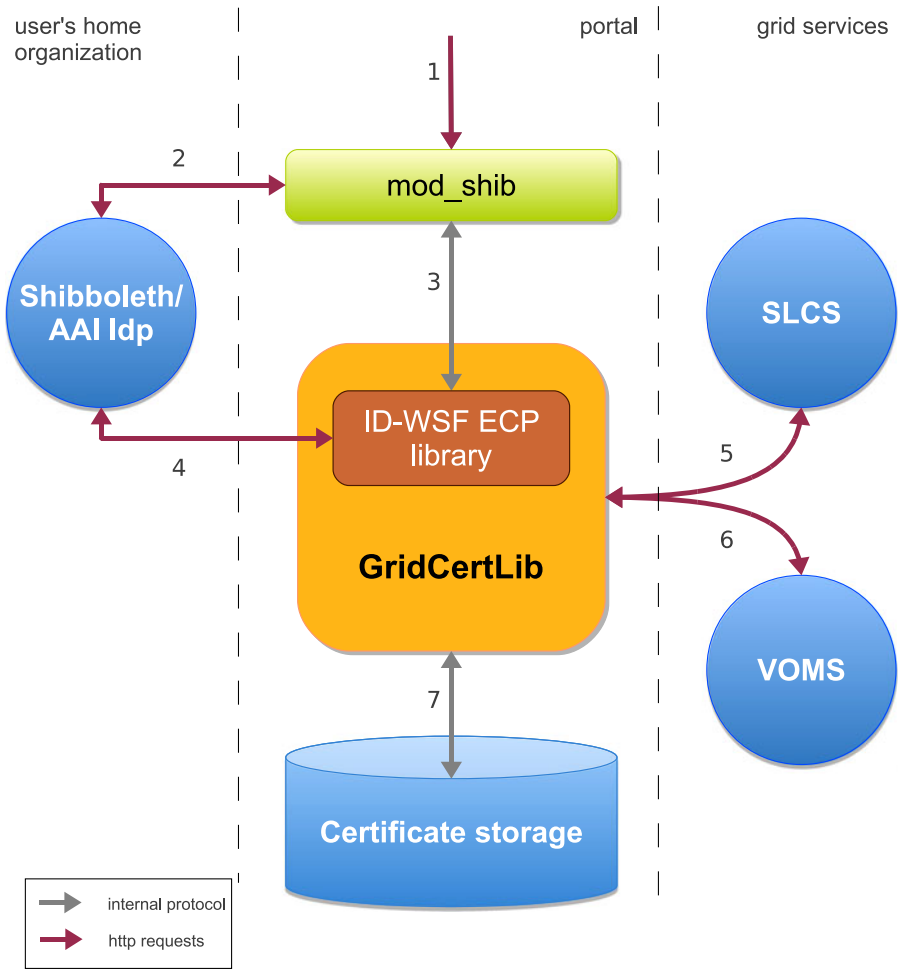

Fig. 1 Interactions of *GridCertLib* with the other involved services. Components in the center column are all part of the same web application (but they could be part of different processes: for instance, the Shibboleth web login is usually handled by an Apache module rather than by Java servlet code). Round boxes on the sides represent services running on remote hosts, that *GridCertLib* interacts with over HTTP. A detailed description of interactions is given in Section 3.1

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 (from a user interface perspective), 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 and Grid proxies, respectively. 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 shortened 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 3.2.1). To solve this problem, *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 Application Programming Interface (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 [42]; 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 data (like the password), which is 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 min in the default configuration) but SP and IdP sessions last much longer (8 hours by default, see [37] for details), 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 HyperText Transfer Protocol (HTTP) redirections, so that the whole procedure does not require any user intervention. We implemented this workaround in the `RenewAssertion` servlet, described in Section 3.2.3.

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 Virtual Organisation (VO) endpoints.

3.2.3 Example Servlets

The provided sample servlets can run in any Java servlet container. They have been successfully tested with the Jetty and Tomcat Java application servers (with an Apache proxy front-end for managing the Shibboleth session).

Three servlets are distributed with the GridCertLib source code:

- *SlcsInit:* This servlet generates an X.509 certificate and private key and uses the SLCS service to sign it. Upon successful completion, the certificate and private key are stored in the filesystem.
- *VomsProxyInit:* This servlet creates a VOMS proxy and stores it in the filesystem.
- *RenewAssertion:* This servlet ensures that a fresh SAML assertion is stored in the SP Shibboleth session cache.

A detailed description of each of these servlets follows.

SlcsInit The *ch.swing.gridcertlib.servlet.SlcsInit* servlet extracts the SAML2 assertion Uniform Resource Locator (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 Shibboleth 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, may be lost before 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. However, this delay was not a problem on the P-GRADE Bioinformatics portal.
- 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, e.g., VO name or Fully-Qualified Attribute Name (FQAN) list. In the P-GRADE implementation, the users can set their VOs 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 Integration into the P-GRADE Portal

The P-GRADE portal comes with full Grid X.509 proxy support, which in this 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 [29] for the Gridsphere portal [13] (provided by the Australian MAMS project [28]) by storing all Shibboleth attributes including the assertion and other attributes that were not previously requested 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 and whether it is valid for longer than 24 h. If not, *GridCertLib* is used to create a new SLCS certificate. 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 VOMS configuration is the same for all users of the portal in our current implementation, which is set to the "life" VO of the national Grid computing infrastructure Swiss Multi-Science Computing Grid (SMSCG) [35], using a portal-wide configuration of *GridCertLib*.

An issue remains: the delegation feature used by *GridCertLib* is not yet deployed as a standard feature in the Swiss SWITCHaai federation, therefore we currently can only make use of this whole mechanism through a special home organization, the Virtual Home Organisation (VHO), 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 SWITCHaai federation has upgraded to a version of Shibboleth that supports delegation, which should happen sometime in late 2011 or 2012.

For now, in the optimal case a user can log in through AAI by selecting the VHO as the "home organization", and is ready to submit Grid jobs to the Swiss Multi-Science Computing Grid. Clicking on the "Certificates" tab will show the details of the current certificates and their validity.

Expiration of the certificate is not an issue, as P-GRADE requests the download of the results only when the user asks for it through the portal. The portal makes sure that a new proxy is generated automatically in the background from the SLCS certificate (if the existing proxy is not valid anymore). Should the SLCS certificate expire, a new one is requested automatically at the next login, so unless the user is actively using the portal browser window for ten days with no interruption, this will not happen.

4.2 Integration into Django-Based Web Applications

Django [8] is a high-level Python Web framework, providing reusable components to build any sort of web application. We have used it to build a simple portal for users of the computational chemistry application GAMESS-US [12, 34]. The portal uses the *Django-Shibboleth*

application[4] [30] to enable users to log in using their SWITCHaai/Shibboleth credentials; new users will have their account created automatically when they log in for the first time.

Django support in *GridCertLib* thus comprises two (inter-dependent) parts:

- A Python package, containing the access-control decorators[5] *certificate_required* and *gridproxy_required*. By using these decorators, a Django programmer can easily mark some URLs as requiring the use of a valid SLCS certificate and/or proxy.
- A set of Java servlets, which should be deployed alongside the Django site, that interface with *GridCertLib* to provide the SLCS- and proxy-generation functionality.

All communication between the Django decorators and the corresponding servlets happens by means of HTTP redirects through the users' web browser.

The *GridCertLib* Django decorators will first ensure that the HTTP request is authenticated with the standard Django login system; when the *Django-Shibboleth* application is installed, this automatically ensures that the HTTP request is part of a valid Shibboleth session.

Next, *GridCertLib* Django decorators check that the certificate (resp. proxy certificate) exists and is valid. For the sake of processing speed (no response can be sent to the web browser until the decorator has passed control to the view function), the decorators assume that no other actor can modify the certificate/proxy files they have created: thus a simple "modification time" check suffices to prove that a certificate/proxy is still in its validity period. Note that, in contrast to what happens in the P-GRADE portal, the certificate/proxy check happens each time the Django view function is invoked, and it is thus essential to keep it performant.

If the certificate/proxy exists and is valid, environment variables are set to the filesystem path of the relevant files to communicate the location to the Grid middleware, and control is passed to the view function.

Otherwise, an HTTP redirect response is issued, channeling the web browser to the URL corresponding to a Django-specific version *SlcsInit* or *VomsProxyInit* servlets. As mentioned for the Example Servlets (see Section 3.2.3), only the *SlcsInit* URL needs to be Shibboleth-protected.

HTTP session cookies are used to tell the servlets to store the certificate/proxy in a certain filesystem location;[6] however this poses a mild security threat: since the servlets URLs must be public (so that the users' web browsers can visit them), then an HTTP request could be crafted to make the servlets read/write the certificate/proxy file in an arbitrary location on the filesystem. Security is enforced with the following procedure:

- Before starting the redirect to the a servlet, the Django access decorator creates an empty directory L, creates a "marker" file in it, and writes a random string K into this "marker" file.
- The decorator redirects the web browser to the servlet URL, passing along L and K (as HTTP cookies).
- The servlet verifies that the "marker" file exists in L and that it has the expected content K, then it deletes the "marker" file and proceeds. For added security, it can optionally verify that L is a filesystem path starting with a configured prefix (e.g., `/var/www/portal`), so that possible damage is confined to a portion of the filesystem.

It is clear that the above procedure guarantees that hypothetical attackers can only trick the

[4] Django structures a web site as an set of web applications, each of which is attached to specific URLs in the web site URL space. Applications can be packaged and deployed separately, and can be thus re-used in different combinations to build a site.

[5] Django routes HTTP requests to Python functions ("view" functions), that are responsible for returning content to the user. Access control is most easily done through Python function decorators: if a view function is marked with the *login_required* decorator, then Django ensures that HTTP requests to that URL come from logged-in users, and will redirect any unauthorized request to the site login page.

[6] Since the *SlcsInit* and *VomsProxyInit* servlets run in a Java server, completely separated by the server running Django, an issue arises as how to communicate certificate/proxy location and passphrases back and forth from the Django decorator to the Java servlets.

GridCertLib servlets into writing into a location *L* if and only if they can already write to *L*.

The added security layer is basically the only difference between the Django-support servlets and the *GridCertLib* example servlets (see Section 3.2.3). After successful creation of the certificate or proxy, the servlet redirects the web browser back to the initial requesting page with no output.

As in the P-GRADE integration, two issues remain that might need special attention in the future:

- The VOMS configuration is the same for all users: while it is possible to extend the Django user object model to include individual VOMS information, this is not necessary at present since all users of the GAMESS portal belong to the same Virtual Organization.
- Until the delegation feature becomes a standard feature of SWITCHaai, users have to select the special home organization VHO in order to use the portal.

Django support for *GridCertLib* provides an example of how *GridCertLib* can be integrated into an existing web framework with little coding and only small edits to tune the example servlet behavior to the interface expected by other portal components.

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, potentially 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 License version 2.0 [2].

The current implementation of *GridCertLib* relies on three key features of the SWITCHaai 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 [35] thus enabling Grid use by virtually any Swiss scientific community.

An interesting future development could be to adapt GridCertLib to draw certificates from the (recently created) Trans-European Research and Education Networking Association (TERENA) on-line CA; this would lift the dependency on the Swiss infrastructure and potentially allow usage of GridCertLib on any European Grid infrastructure.

More generally, one could investigate whether *GridCertLib* could be ported to provide its functionality on top of equivalent base technologies (e.g., substitute Shibboleth with a different SAML-based federated authentication infrastructure). Developments in this area could turn *GridCertLib* into a modular system capable of providing its functionality for almost all Grid users today. No investigation has been carried out by us in this area: the project that funded *GridCertLib* development had a practical scope of producing a simple single sign-on solution for the selected portals; we are anyway open to collaborations in this respect.

GridCertLib has already been successfully deployed and integrated into a Bioinformatics portal based on P-GRADE, and into a Django-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 [21] and the new WS-PGRADE [22]. 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 SWITCHaai 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 [40] 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 [26] 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.

Acknowledgements This work was carried out in the context of the "Swiss Grid Portal" project, funded through the SWITCH-AAA track and through the SyBIT project of SystemsX.ch. We would like to thank all our collaborators in the Swiss Grid Portal project—Cesare Pautasso, Frédérique 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.

List of Acronyms

AAI Authentication and Authorization Infrastructure
AC Attribute Certificate (VOMS, X.509)
API Application Programming Interface
CA Certification Authority
CSR Certificate Signing Request
DN Distinguished Name
ECP Enhanced Client or Proxy
EGEE Enabling Grids for E-sciencE
FQAN Fully-Qualified Attribute Name (VOMS)
GAMESS General Atomic and Molecular Electronic Structure System, a Computational Chemistry application (see [12, 34])
GC3 Grid Computing Competence Center, University of Zurich
HTTP HyperText Transfer Protocol
ID-WSF Identity Domain—Web Service Framework (Shibboleth)
IGTF International Grid Trust Federation
IdP Identity Provider (Shibboleth)
MAMS Meta Access Management System, an Australian development project (see [28])
PKI Private Key Infrastructure
SAML Security Assertion Markup Language
SAML2 Security Assertion Markup Language version 2
SARoNGS UK development project (see [20])
SLCS Short-Lived Credential Service
SMSCG Swiss Multi-Science Computing Grid
SP Service Provider (Shibboleth)
SSO Single Sign-On
SWITCH Swiss Academic Network Provider
TERENA Trans-European Research and Education Networking Association
URL Uniform Resource Locator
VHO Virtual Home Organisation
VOMS Virtual Organisation Membership Service
VO Virtual Organisation
XACML eXtensible Access Control Markup Language

References

1. Alfieri, R., Cecchini, R., Ciaschini, V., dell'Agnello, L., Frohner, Á., Lőrentey, K., Spataro, F.: From gridmap-file to VOMS: managing authorization in a Grid environment. Future Gener. Comput. Syst. **21**(4), 549–558 (2005)
2. The Apache Software Foundation: Apache License, version 2.0 http://www.apache.org/licenses/LICENSE-2.0. Cited 17 Apr 2011
3. Barbera, R., Donvito, G., Falzone, A., La Rocca, G., Milanesi, L., Maggi, G.P., Vicario, S.: The GENIUS Grid Portal and robot certificates: a new tool for e-Science. BMC Bioinformatics **10**(Suppl 6), S21 (2009). http://www.biomedcentral.com/1471-2105/10/S6/S21
4. Barton, T., Basney, J., Freeman, T., Scavo, T., Siebenlist, F., Welch, V., Ananthakrishnan, R., Baker, B., Goode, M., Keahey, K.: Identity federation and attribute-based authorization through the Globus toolkit, Shibboleth, Gridshib, and MyProxy. In: 5th Annual PKI R&D Workshop (2006)
5. Barton, T., Cantor, S., Olshansky, S.: Shib-uPortal Home. https://wiki.shibboleth.net/confluence/display/ShibuPortal/Home. Cited 17 Apr 2011
6. Burruss, J.R., Fredian, T.W., Thompson, M.R.: ROAM: an authorization manager for Grids. J. Grid Computing **4**(4), 413–423 (2006)
7. Cantor, S., et al.: Shibboleth Federations. https://wiki.shibboleth.net/confluence/display/SHIB/ShibbolethFederations. Cited 17 Apr 2011
8. Django: http://www.djangoproject.com. Cited 17 Apr 2011
9. The European Grid Authentication Policy Management Authority in e-Science: Guideline on IGTF Approved Robots, version 1.0. http://www.eugridpma.org/guidelines/robot/. Cited 17 Apr 2011
10. Farkas, Z., Kacsuk, P.: P-GRADE portal: a generic workflow system to support user communities. Future Gener. Comput. Syst. **27**(5), 454–465 (2011). doi:10.1016/j.future.2010.12.001
11. Garzoglio, G., et al.: Definition and implementation of a SAML-XACML profile for authorization interoperability across Grid middleware in OSG and EGEE. J. Grid Computing **7**(3), 297–307 (2009). doi:10.1007/s10723-009-9117-4

12. Gordon, M.S., Schmidt, M.W.: Advances in electronic structure theory: GAMESS a decade later. In: Dykstra, C.E., Frenking, G., Kim, K.S., Scuseria, G.E. (eds.) Theory and Applications of Computational Chemistry: The First Forty Years, pp. 1167–1189. Elsevier, Amsterdam (2005)
13. Grisphere Portal: http://www.gridsphere.org/gridsphere/gridsphere. Cited 30 Dec 2010
14. Hewitt, M., Kaushal, S.: Experiences of P-GRADE at the White Rose Grid e-Science Centre. Presentation held at the 2010 P-GRADE User Communities Workshop (PUCoWo), ETH Zürich, 10 Jun 2010. http://portal.p-grade.hu/pucowo/download/slides/WRG-Mark-Shiv.ppt (2010)
15. Hodges, J.: How to Study and Learn SAML. http://identitymeme.org/doc/draft-hodges-learning-saml-00.html. Cited 9 Sept 2011
16. Hodges, J., Aarts, R., Madsen, P., Cantor, S., Cahill, C., Champagne, D., Ellison, G., Lockhart, R., Whitehead, G.: Liberty ID-WSF Authentication, Single Sign-On, and Identity Mapping Services Specification. http://www.projectliberty.org/liberty/content/download/3441/22949/file/liberty-idwsf-authn-svc-2.0-diff-v1.0.pdf. Cited 17 Apr 2011
17. IGTF: The International Grid Trust Federation Charter. http://www.igtf.net/charter.html. Cited 17 Apr 2011
18. Internet2 Middleware Initiative: Shibboleth. http://www.internet2.edu/shibboleth. Cited 17 Apr 2011
19. Internet2 Middleware Initiative: High Level Introduction to Shibboleth. http://shibboleth.internet2.edu/HighLevelIntro.html. Cited 9 Sept 2011
20. JISC: Shibboleth Access to Resources on the National Grid Service. http://www.jisc.ac.uk/whatwedo/programmes/einfrastructure/sarongs.aspx. Cited 17 Apr 2011
21. JOpera. http://www.jopera.org/. Cited 17 Apr 2011
22. Kacsuk, P.: P-GRADE portal family for Grid infrastructures. Concurr. Comput.: Practice and Experience **23**(3), 235–245 (2011)
23. Kacsuk, P., Sipos, G.: Multi-Grid, multi-user workflows in the P-GRADE portal. J. Grid Computing **3**(3–4), 221–238 (2005)
24. Kacsuk, P., et al.: WS-PGRADE: supporting parameter sweep applications in workflows. In: 3rd Workshop on Workflows in Support of Large-Scale Science. In conjunction with SC 2008, pp. 1–10, IEEE, Austin, TX, USA (2008). doi:10.1109/WORKS.2008.4723955
25. Kouril, D., Basney, J.: A Credential Renewal Service for Long-Running Jobs. In: 6th IEEE/ACM International Workshop on Grid Computing (Grid 2005), Seattle, WA, 13–14 Nov 2005
26. Liferay portal: http://www.liferay.com/. Cited 17 Apr 2011
27. Lorch, M., Kafura, D.: The PRIMA Grid Authorization System. J. Grid Computing **2**(3), 279–298 (2005)
28. MAMS project: Meta Access Management System. https://mams.melcoe.mq.edu.au/zope/mams. Cited 17 Apr 2011
29. MAMS project: Shibbolizing GridSphere demo. https://mams.melcoe.mq.edu.au/zope/mams/kb/all/GridSphere%20Wink%20demo.zip/view. Cited 17 Apr 2011
30. Morrison, S.: Django-Shibboleth: Shibboleth login/registration integration. http://code.arcs.org.au/gitorious/django/django-shibboleth. Cited 17 Apr 2011
31. NCSA: MyProxy Credential Management Service. http://grid.ncsa.illinois.edu/myproxy/. Cited 30 Dec 2010
32. Novotny, J., Tuecke, S., Welch, V.: An online credential repository for the Grid: MyProxy. In: Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), pp. 104–111. IEEE Press (2001)
33. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: Community authorization service for group collaboration. In: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, pp. 50–59 (2002)
34. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., Windus, T.L., Dupuis, M., Montgomery, J.A.: General atomic and molecular electronic structure system. J. Comput. Chem. **14**, 1347–1363 (1993)
35. SMSCG project: Swiss Multi-Science Computing Grid. http://www.smscg.ch/. Cited 17 Apr 2011
36. SWITCH: Description of the SLCS. http://www.switch.ch/grid/slcs/about/about_long.html. Cited 17 Apr 2011
37. SWITCH: Expert Session Add-On. http://www.switch.ch/aai/demo/2/expert_session.html. Cited 17 Apr 2011
38. SWITCH: Simple Demo http://www.switch.ch/aai/demo/2/simple.html. Cited 17 Apr 2011
39. SWITCH: SWITCHaai—The Key That Connects Students and the University. http://www.switch.ch/aai. Cited 17 Apr 2011
40. SystemsX.ch: SyBIT: Systems Biology IT. http://www.systemsx.ch/projects/systemsxch-projects/sybit/. Cited 17 Apr 2011
41. Tschopp, V.: ID-WSF ECP Web Service Client. https://forge.switch.ch/redmine/projects/idwsfecp. Cited 30 Dec 2010
42. Tschopp, V., Witzig, Ch.: Short-lived Credential Service—User Guide. EGEE-II project. https://edms.cern.ch/document/788604/1 (2006)
43. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, RFC3820. The Internet Society. http://www.ietf.org/rfc/rfc3820.txt (2004)