J. Sens. Actuator Netw. **2013**, *2*, 109-121; doi:10.3390/jsan2010109

OPEN ACCESS

Journal of Sensor and Actuator Networks

ISSN 2224-2708

www.mdpi.com/journal/jsan

Article

User and Machine Authentication and Authorization Infrastructure for Distributed Wireless Sensor Network Testbeds

Markus Anwander, Torsten Braun *, Philipp Hurni, Thomas Staub and Gerald Wagenknecht

Institute of Computer Science and Applied Mathematics, Universität Bern, Neubrückstrasse 10, 3012 Bern, Switzerland; E-Mails: anwander@iam.unibe.ch (M.A.); hurni@iam.unibe.ch (P.H.); staub@iam.unibe.ch (T.S.); wagen@iam.unibe.ch (G.W.)

* Author to whom correspondence should be addressed; E-Mail: braun@iam.unibe.ch; Tel.: +41-31-631-4994; Fax: +41-31-631-3262.

Received: 24 December 2012; in revised form: 12 February 2013 / Accepted: 17 February 2013 / Published: 6 March 2013

Abstract: The intention of an authentication and authorization infrastructure (AAI) is to simplify and unify access to different web resources. With a single login, a user can access web applications at multiple organizations. The Shibboleth authentication and authorization infrastructure is a standards-based, open source software package for web single sign-on (SSO) across or within organizational boundaries. It allows service providers to make fine-grained authorization decisions for individual access of protected online resources. The Shibboleth system is a widely used AAI, but only supports protection of browser-based web resources. We have implemented a Shibboleth AAI extension to protect web services using Simple Object Access Protocol (SOAP). Besides user authentication for browser-based web resources, this extension also provides user and machine authentication for web service-based resources. Although implemented for a Shibboleth AAI, the architecture can be easily adapted to other AAIs.

Keywords: authentication; authorization; wireless sensor networks; experimentation; testbeds

---

1. Introduction

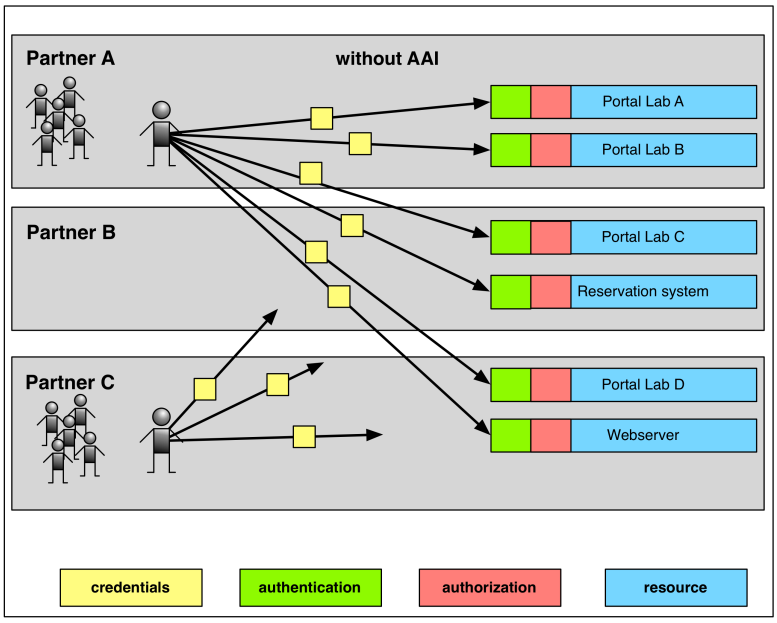
Resource providers in the Internet usually want to restrict access to their resources to certain users or provide user-specific content. Both cases require authentication and authorization. Authentication is the

process of verifying a claimed identity. It is always a prerequisite before any authorization can happen. Authorization is the process of guaranteeing that only authorized users can access a resource or a service.

Figure 1 depicts a system with a traditional non-federated authentication and authorization infrastructure (AAI). Users have to register themselves on every resource that they want to access. Usually, they receive a new username and password pair, so-called credentials, for each single resource. Obviously drawbacks are the following:

- Users have to keep track of many credentials.
- Every resource requires time-consuming user registration and administration.
- No consistent user data management exists.

Figure 1. Traditional non-federated AAI.



Those drawbacks should be avoided by using federated AAIs. Section 2 gives an introduction into principles of federated AAIs and discusses related work on existing federated AAIs. Section 3 presents the design concept and implementation of our authentication and authorization infrastructure. In Section 4 we describe the reservation system and the user interfaces for accessing our distributed Wireless Sensor Network (WSN) testbed. The paper is concluded in Section 5.

2. Background

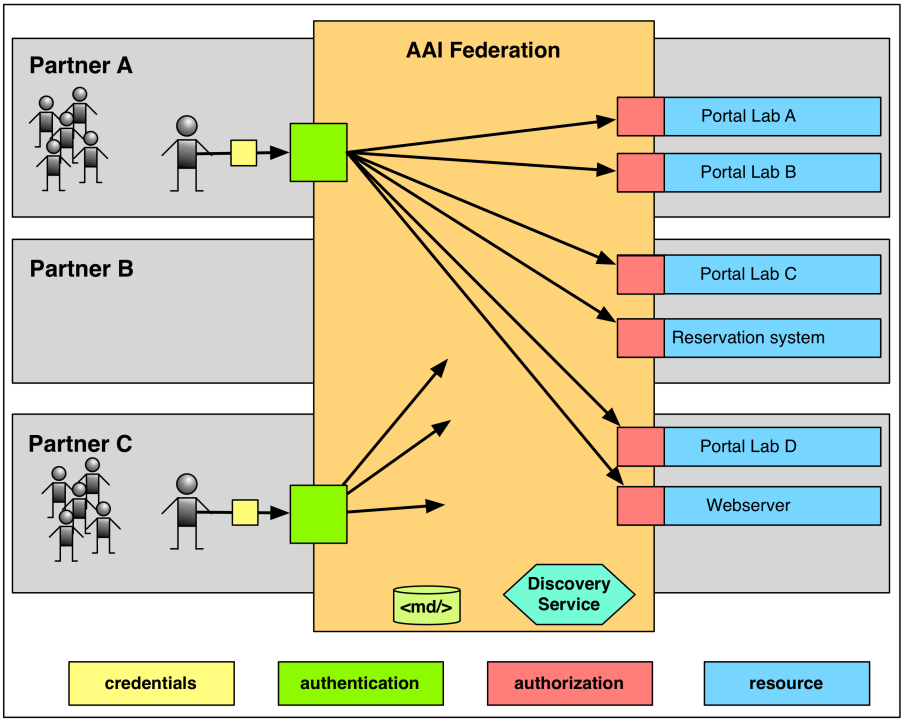
2.1. Federated Authentication and Authorization Infrastructures

A federation is a collection of organizations that agree to interoperate under a certain rule set. Federations usually define trusted roots, authorities, and attributes, along with distribution of metadata describing this information. In general, each partner participating in a federation operates at least one identity provider (IDP) for its users and a number of service providers (SP). Each partner controls

the information relevant to it. An SP integrates a resource provider into an AAI federation. Partners register, maintain, and authenticate their users by the home organization's IDP. Resource owners and administrators define resource access rules based on the user attributes received from the IDP.

A federated AAI simplifies the processes for all parties involved using the concept of federated identity management. For our implementations we used the Shibboleth system [1,2], which is an open source software package for web single sign-on (SSO). Shibboleth is used by some European National Research and Education Networks (NRENs), such as DANTE, GEANT2, DFN, SWITCH as well as several scientific publishers. A user registers himself/herself only once at the so-called home organization to which the user is affiliated. This home organization is responsible for maintaining user related data and hands out credentials to users. Home organizations are usually institutions like universities and companies. Thus, the federated identity management relies on trust between identity and service providers. After successful authentication the resources receive predefined user attributes from the identity provider of the home organization that is responsible for the user. Authorization is then performed at each individual resource based on attributes. Figure 2 shows an example.

Figure 2. Authentication with a federated AAI.



The advantages are evident:

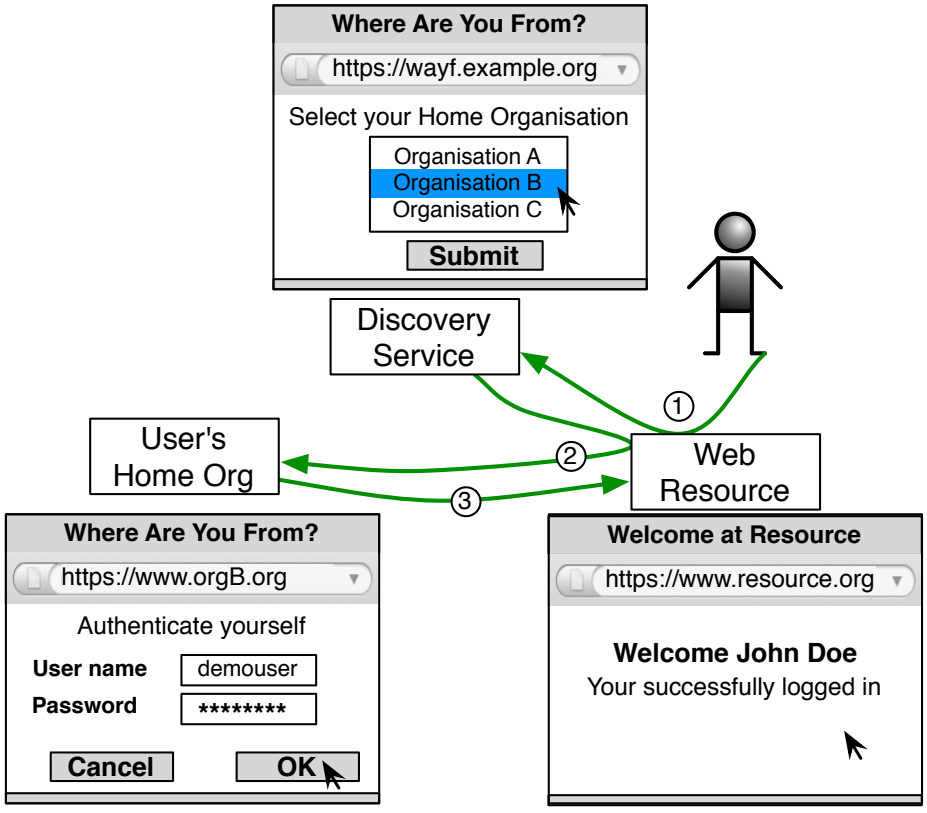
- IDs issued and managed by home organizations avoid the effort to register and administer users at each resource.
- A single authentication mechanism standardized for each home organization allows users to access all resources of the whole AAI federation.
- Resources can be easily made available to a larger number of potential users.
- User data remain at the home organization.

2.2. Authentication and Authorization in Federated AAIs

All AAI-enabled resources are available to a user with a single set of credentials. Authentication is always carried out by the IDP of the user's home organization. For authentication, the Shibboleth IDP uses credentials such as a user ID and password. The IDP can also provide additional information about the user (e.g., a unique identifier) to the SP, *i.e.*, the owner of the resource to be protected. There is no need for resource administrators to register new users, as they get the required information for authorization directly from the IDP of the user's home organization.

Figure 3 demonstrates the authentication at an SP. After connecting to the protected resource, the user is redirected to a discovery service (DS), where it selects its home organization, respectively the corresponding IDP (**1**). The user enters his/her credentials at the Single Sign-on (SSO) login interface of the IDP. After successful authentication, the IDP sends the user attributes to the SP (**2**). The user is then forwarded to the resource for authorization (**3**). Authorization is the process of verifying whether the authenticated user is permitted to access the resource. Authorization is then only based on user properties or attributes such as the home organization or the unique identifier.

Figure 3. Authentication overview.



2.3. Related Work

There are a number of projects related to Shibboleth [1], which is a standards based, open source software package for web single sign-on across or within organizational boundaries.

GridShib for Shibboleth2 [3,4] helps to bridge the gap between SAML federations based on Shibboleth and Grid federations based on the Globus Toolkit. The effort to adapt GridShib to the target

requirements go beyond the scope of this work. The D-Grid Initiative [5] has the goal to develop a distributed, integrated resource platform for high-performance computing and related services to enable processing of large amounts of scientific data and information. It is focused on a grid infrastructure. ESP-Grid [6] has the goal to evaluate the usability of Shibboleth and PKI for grids in general. The Condor Shibboleth Merger Project integrates Condor [7] with Shibboleth. Condor is a grid-computing software and the user does not require an account on the remote host. There are a number of small projects (e.g., OIOSAML [8], predic8 [9]), which replace a service provider by an own application. The provided IDP communication API is only in a prototypical state, unstable and thus does not fulfill our requirements. OAuth [10] enables a third-party application to obtain limited access to an HTTP service. In short it can be considered as delegated authorization, while Shibboleth includes both authentication and authorization. On the other hand, Web Service Security (WS-Security) [11] focuses on message integrity and confidentiality, but does not address authorization of services.

In [12] we presented a secure remote authentication, operation and management infrastructure for distributed WSN testbeds. In contrast to that work, in this paper we are focusing on the reservation of non-web-based resources with AAI.

2.4. Machine-Based Authentication Using Web Services

Shibboleth-protected web services are usually supporting access by human users via web browsers. The whole authentication and authorization process requires human interaction, e.g., selection of home organizations and entering password credentials. If a non-browser based client, e.g., a script, wants to access Shibboleth-protected web services, machine-based authentication is required. Federated AAIs such as Shibboleth cannot be used for machine-based authentication and authorization without modification, *i.e.*, when scripts or machines desire to access web services. In our WSN testbed use case, web services for reservation of wireless sensor resources had to be provided to scripts and machines such that resources can be reserved via non-web-browser-based systems.

To enable machine-based authentication as well as service access, we have implemented Simple Object Access Protocol (SOAP) based web services for authentication, authorization and resource reservation. Every IDP is enhanced by an authentication web service. This web service returns a temporary authentication key for machine-based authentication. The temporary authentication key works similar to a session key used in web browsers. It is a secret key and a time-limited validity. The authentication key is transmitted to the corresponding IDP, which verifies the validity of the key and authenticates the machine. The user attributes are then transmitted to the requesting SP.

In some cases, performing only authorization may not be sufficient. Some resources can only be used by a limited number of users at a time, e.g., a wireless sensor network (WSN) testbed or some individual sensor nodes. If all authorized users have access to all the nodes at the same time, they may produce interferences among their experiments. Therefore, we introduce a resource reservation service to reserve exclusive access to resources during a certain time interval.

Based on the user attributes mentioned above, authorization can be performed and access to the reservation web services can be granted. As a proof of concept we implemented an iPhone [13] application that makes use of the authentication web services.

3. User and Machine Authentication and Authorization Infrastructure

3.1. Design Concept

A web service running on an AAI protected resource should be able to perform authentication and authorization for the incoming web service calls. Therefore, every web service call comes with an additional *temporary authentication key*. With this *temporary authentication key* incoming web service calls can be authenticated. *Temporary authentication keys* include the issuing home organization and are valid for a certain time, e.g., 1 hour. The included home organization indicates the URL of the responsible IDP, where the key has to be validated. If the *temporary authentication key* is valid, the responsible IDP returns user attributes to the requesting resource. Based on these user attributes, the resource can then perform authorization.

Figure 4. Sequence diagram of authentication and authorization of web service calls.

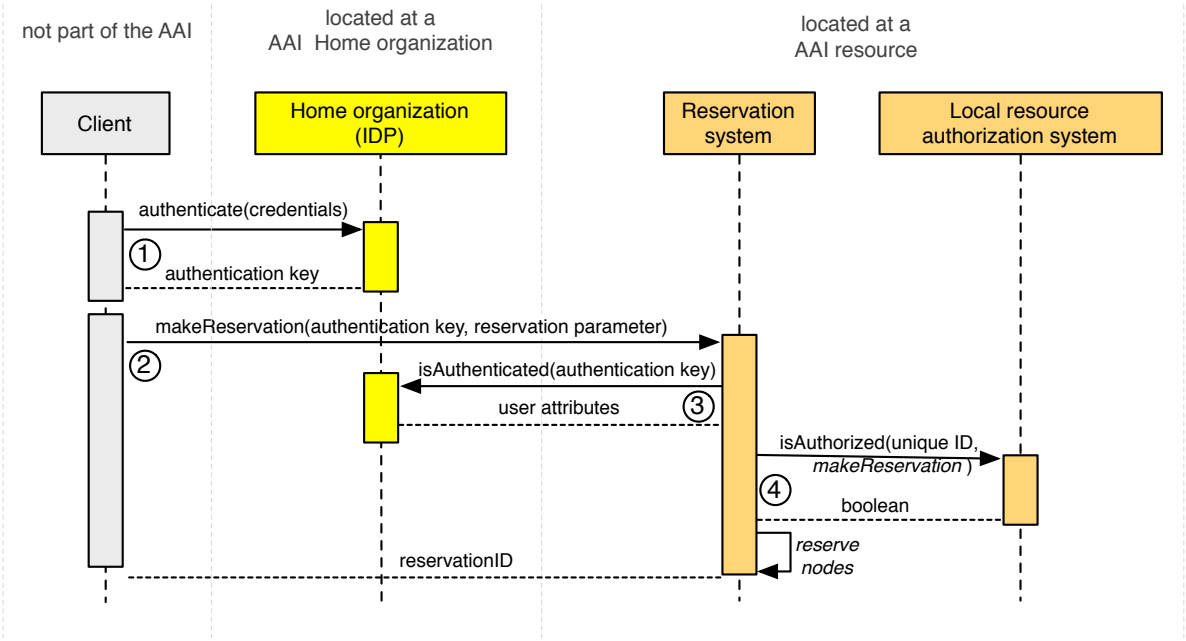


Figure 4 illustrates the web service-based authentication and authorization. In the depicted example, a client wants to reserve sensor nodes. First, the client sends the credentials to the home organization of the user. The home organization validates the authentication request and returns the secret *temporary authentication key* (**1**). In this case, the credentials are only sent to the own home organization, which is trusted by a user. Credentials are never sent to any other systems. Now, the client transmits the received *temporary authentication key* with all required attributes for reservation to the reservation server (**2**), which can be considered as SP. The reservation server verifies the *temporary authentication key* at the user's home organization. After successful verification, the IDP of the home organization returns the user's unique identifier (uniqueID), which includes the domain name of the home organization and a username. Therefore, the uniqueID is unique in the whole AAI federation (**3**). With the uniqueID, the reservation system checks the authorization for the called web service (*makeReservation*) (**4**). We implemented a local resource authorization system, which provides flexible authorization rules. This

enables fine-grained control over the web services that a user is authorized to call. In this use case, the authorization decision is based on the unique user identifier. Decision may be based on other attributes than the uniqueID. If the user is authorized, the requested operations can be executed.

3.2. Implementation

We implemented a wireless sensor network testbed federation including a web service extension for Shibboleth for the WISEBED [14] large-scale distributed wireless sensor network testbed. The sensor nodes are resources and for exclusively access these resources have to be reserved.

Figure 5 depicts the message flow to get the *temporary authentication key* in the WSN testbed federation. A SOAP web service client sends the credentials to the home organization to which the user or machine is affiliated. The web service call establishes an encrypted channel by a Secure Socket Layer (SSL) connection. A WSDL (Web Services Description Language) file describes the offered authentication service.

Figure 5. Message Flow: Authenticate.

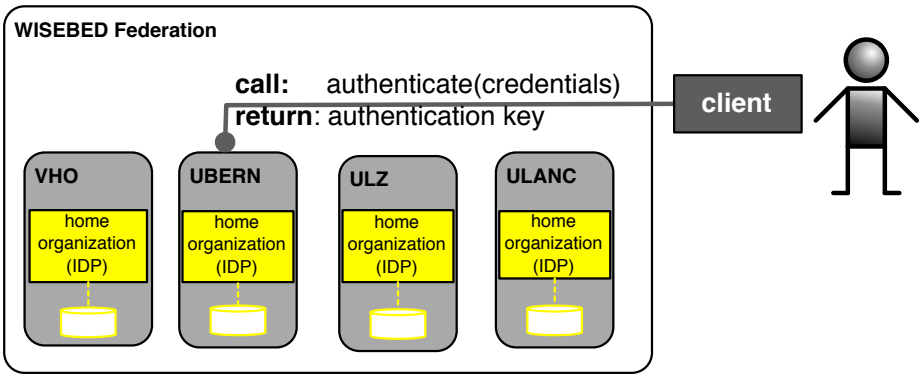
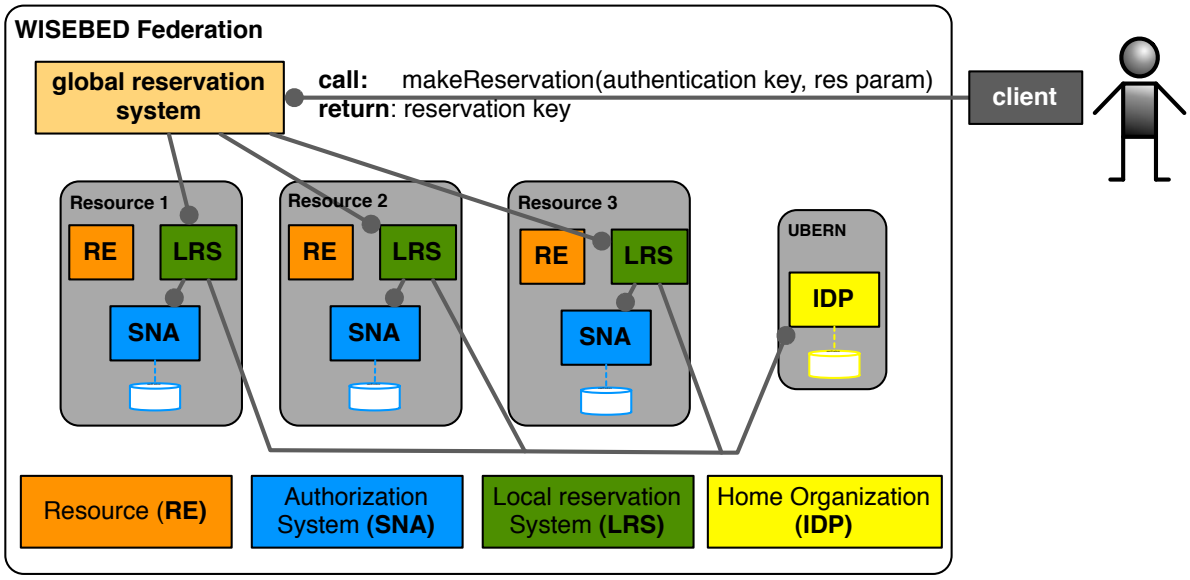


Figure 6 depicts the process to perform a reservation using the *temporary authentication key*. The client sends the *temporary authentication key* and the required parameters for a reservation to a global reservation system. The global reservation system federates several local reservation systems. The global reservation system forwards the reservation request to the responsible local reservation systems. A reservation server as part of a local reservation system verifies the received *temporary authentication key* at the corresponding home organization. After receiving the user's uniqueID and other user attributes from the IDP, the local authorization system performs the validation of the authorization for the user, *i.e.*, checks if the user is authorized to call the *makeReservation* web service. We implemented an authentication and authorization system for web services based on Shibboleth. Our extension does not make any modification to the standard Shibboleth installation. It only provides an additional web service. Applications that are not running in a web-browser can use the web service for authentication. Our implementation uses an HTTP client, which performs the same authentication operations as a normal user through a web browser. For the Shibboleth system, it makes no difference if a real user with a web browser or an HTTP client tries to access a Shibboleth-protected resource.

Figure 7 describes the implementation of the authentication web service. Steps (**2**)–(**4**) are common Shibboleth operations without any adaptation. In step (**1**), the user starts the (non-web-based) client

with the IDP URL and provides its credentials (username and password). Then the client sends these credentials to the authentication SOAP web service of the IDP. In step (**2**), the HTTP client connects to an internal website of the IDP, which is protected by Shibboleth. As the HTTP client has no valid authentication session, the request is forwarded to the discovery service and the HTTP client selects the corresponding home organization. In step (**3**), the HTTP client enters the credentials, *i.e.*, username and password into the login form of the Shibboleth central authentication service (CAS). CAS is a SSO access controller and handles the authentication process and the session management for already authenticated users. After successful authentication the CAS redirects the HTTP client to the local web resource (**4**). The HTTP client has now a valid authentication session cookie from the Shibboleth CAS. Then, the authentication web service creates the *temporary authentication key*, which consists of the value of the CAS session cookie and the name of the home organization (**5**). Moreover, all user attributes delivered by the home organization can be further processed by the HTTP client connection, if required.

Figure 6. Message Flow: Make Reservation.



By adding the received *temporary authentication key* to a web service call, every SOAP server can verify the authentication. Figure 8 shows an example for authentication and authorization of a web service call (e.g., *makeReservation*). First, the client sends the *temporary authentication key* and the required attributes for making a reservation to the reservation server (**1**). Then, the reservation server starts the HTTP client with the CAS session cookie retrieved from the *temporary authentication key*. This HTTP client connects to a local website, which is empty and provides the user attributes after the authentication process. Since the website is protected by Shibboleth, the client is automatically redirected to the discovery service (**2**). After selecting the home organization, the client is forwarded to the Shibboleth CAS interface. There, the HTTP client is authenticated by the CAS session cookie. In addition, the Shibboleth IDP sends back the user attributes to the local website. Afterwards, the HTTP client is redirected to the web resource and receives the user attributes from the web server as session variables. After successful authentication, the reservation server can read the user attributes from the

HTTP client and perform authorization. For this purpose, the authorization web service *isAuthorized* of the local resource authorization server can be used. The local authorization web service checks whether the sent uniqueID is authorized to perform the requested function. This authorization web service can only be used by the local reservation server. All other clients are blocked.

Figure 7. Implementation of authentication web service.

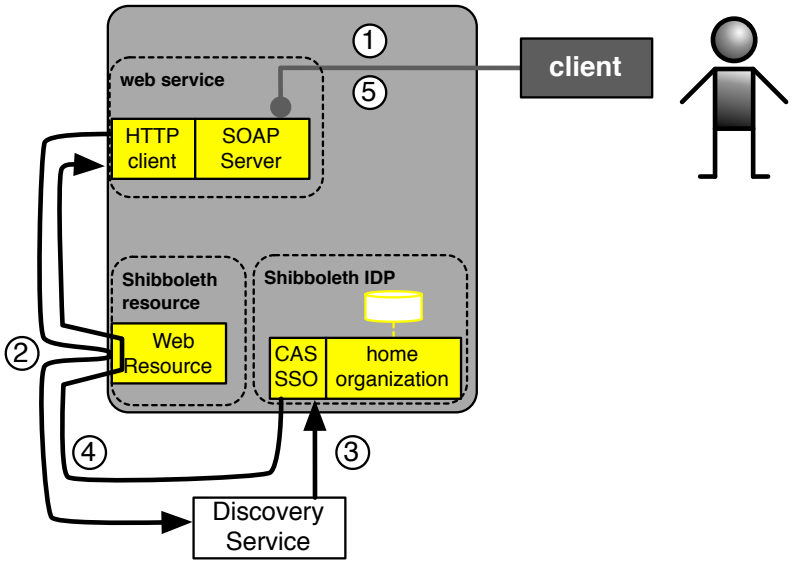
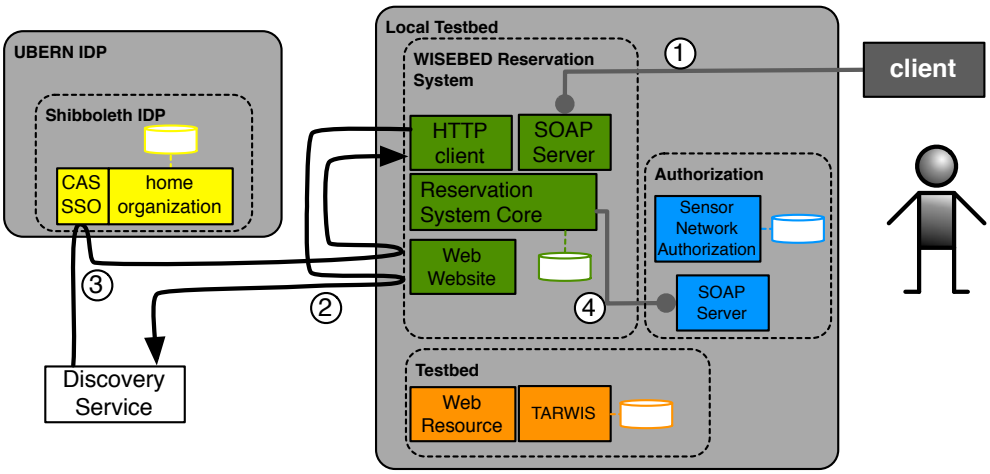


Figure 8. Example: Authentication and authorization for makeReservation.



4. Resource Reservation and User Interfaces

4.1. Reservation Web Services

In the target application scenario, several wireless sensor network testbeds are maintained by different organizations. For experimentation, a user requires time-exclusive access to sensor nodes of one or more testbeds. Therefore, a reservation service has to coordinate access to the sensor nodes. To reserve sensor nodes in the federated WSN testbed, we implemented such a reservation system. It contains a web

based graphical user interface, which can be used to perform the reservation process via a web browser. Furthermore, we implemented an iPhone application, which can be used as a non-browser client to reserve sensor nodes of selected testbeds. The non-browser based access to the reservation system uses the web service authentication presented in Section 3.2.

The reservation system provides five web service calls, which a client can use to reserve wireless sensor nodes from an individual testbed and to manage its reservations: *getReservations*, *getConfidentialReservations*, *getReservation*, *makeReservation*, and *deleteReservation* (*cf.* Figure 8).

To get all existing reservations for a given time period, the web service call *getReservations* can be used. It returns a list of the reservations including the reservation time period and the reserved nodes. It does not include any information about the users. It is used to determine occupied and free time-slots for reservations.

To get user's own reservation for a given time period, the web service call *getConfidentialReservations* can be used. It returns all reservations for a given user. This includes the user ID and the *reservation key*. To authenticate and authorize the user's request, the *temporary authentication key* is used. After user authentication, the web service call *isAuthorized* verifies whether the user is authorized to perform this action (cf. Figure 4). After successful authorization, the list of the user's reservations is delivered to the user. If the user has an administrator role, all reservations of all users are delivered. With this information the administrator can manage all reservations.

The data of a concrete reservation can be retrieved by using the web service call *getReservation* with a given reservation key. The data includes time period, node list, and user information.

The web service call *makeReservation* is used to reserve wireless sensor nodes for a certain time period. Besides the list of required nodes and the time period, the authentication data with the *temporary authentication key* has to be provided as a parameter. User authentication is done with the *temporary authentication key*. Afterwards the authorization is verified. After successful authorization, the availability of the nodes for the given time period is checked. If the reservation was successful, the nodes are reserved for the time period. The user then receives a *secret reservation key* as acknowledgment of the successful reservation. A user with an administrator role can also reserve nodes for another user.

A reservation for a given *reservation key* can be deleted using the web service call *deleteReservation*. The user authentication data has to be given as well. A normal user can only delete its own reservations. An administrator can delete all users' reservations.

4.2. Web-based Graphical User Interface

Reservations can also be managed over a web-based GUI using a normal web browser. The web-based GUI is integrated into the TARWIS GUI [15] for managing complete local wireless sensor network testbeds. Figure 9 shows this web front-end. A user selects the calendar dates and time slots for the sensor nodes to be reserved. The reservation is done using the web services provided by the reservation system.

Because the TARWIS GUI is a protected web-resource, the authentication process is done with the login form of the Shibboleth CAS. After successful authentication, the user attributes are transmitted to the GUI. The decision if a user is allowed to reserve sensor nodes is done on the basis of the

uniqueID. Because the TARWIS GUI website is Shibboleth protected, a *temporary authentication key* is not required.

Figure 9. Reservations system GUI integrated into TARWIS GUI.

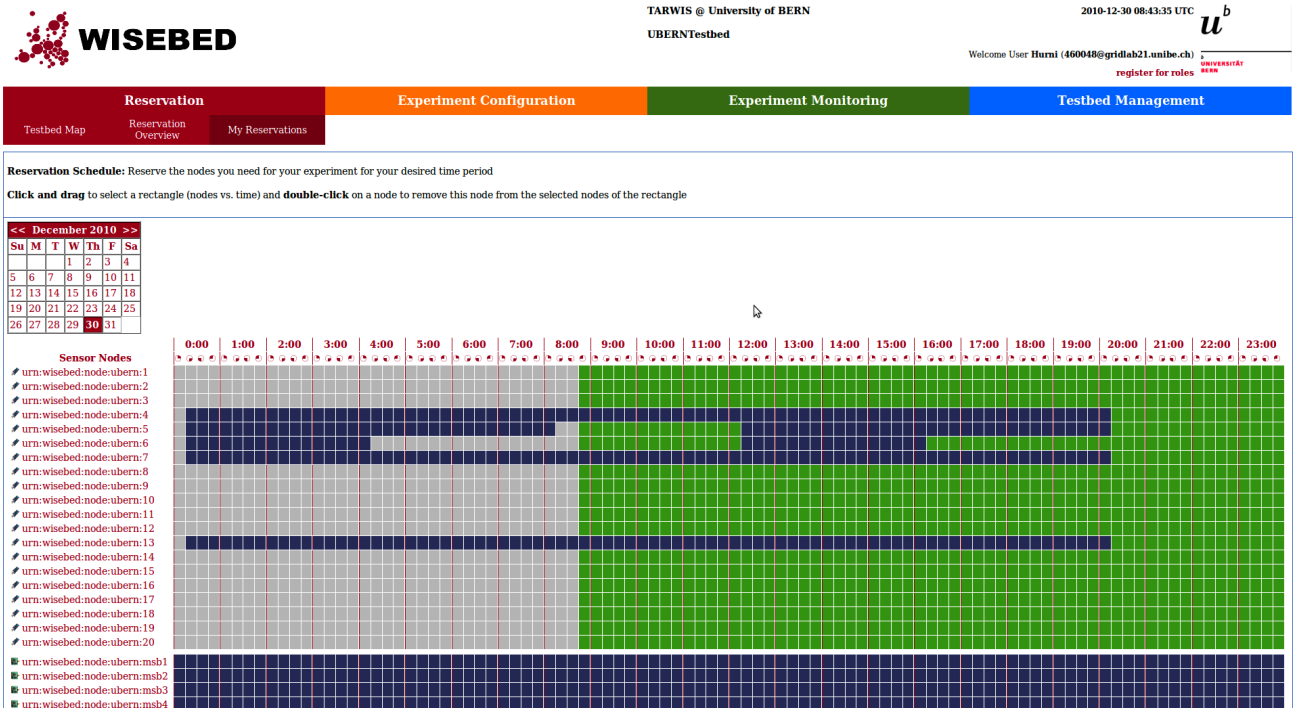
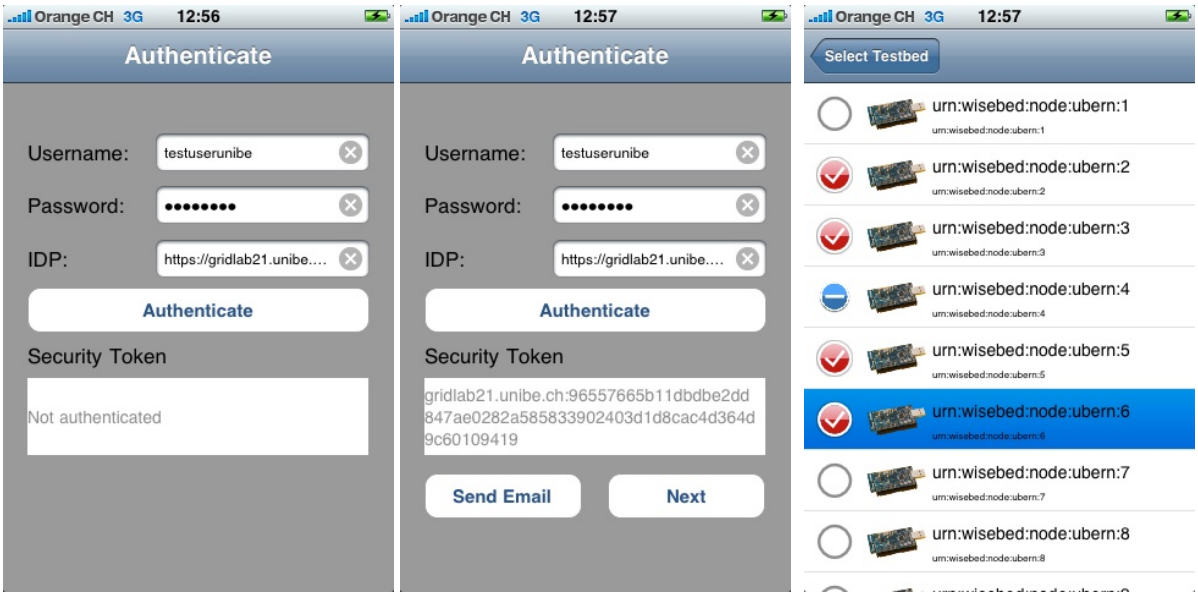


Figure 10. Screenshots of the iPhone application (authentication, authentication response, and node reservation).



4.3. iPhone Application

An iPhone application has been developed as a proof-of-concept for a non-browser client using the developed AAI web service authentication. This application provides a front-end for the reservation

system (see Figure 10). Obviously, the iPhone application has limited functionality compared with a web-based GUI as presented in Subsection 4.2. A certain subset of operations such as executing and displaying resource reservations can be conveniently performed using the iPhone application, while more sophisticated operations such as configuring WSN experiments might be better done using the web-based GUI.

The iPhone application allows users to manage their reservations for the provided wireless sensor testbeds, whereas the user authentication is handled by the AAI web service authentication system. The user provides his/her credentials, *i.e.*, username and password, to the IDP for authentication. After successful user authentication, the iPhone application uses the *temporary authentication key* to call reservation services of the wireless sensor network testbed and manages the resources for an experiment of the user. Although the initial purpose of the iPhone application was to support resource reservations for federated WSN testbeds, it provides a generic example for the usage of the AAI web service authentication concept.

5. Conclusions

The Shibboleth authentication and authorization infrastructure provides only protection for browser-based resources. To protect web resources, which are accessible using web services, we designed and implemented a web service extension for Shibboleth. Our extension does not make any modification to the standard Shibboleth installation.

In case of wireless sensor network testbeds, simple authorization is not sufficient. To provide an exclusive access on a resource or a part of a resource during a certain time interval, we implemented a reservation system. A web interface has been implemented to perform and manage reservations. It has been integrated into the TARWIS GUI [15] to administrate all user reservations. Furthermore, we implemented an iPhone application as a non-browser based client.

References

1. Shibboleth: What's Shibboleth? Available online: http://shibboleth.net (accessed on 31 January 2013).
2. SWITCH, The Swiss Education and Research Network Zurich, Switzerland: Authentication and Authorization Infrastructure in a nutshell. Available online: http://www.switch.ch/aai/docs/AAI-Flyer_en.pdf (accessed on 31 January 2013).
3. GridShib for Shibboleth2. Available online: http://gridshib.globus.org (accessed on 31 January 2013).
4. Jie, W.; Arshad, J.; Sinnott, R.; Townend, P.; Lei, Z. A review of grid authentication and authorization technologies and support for federated access control. *ACM Comput. Surv.* **2011**, *43*, No. 2, Article 12.
5. D-Grid Initiative: IVOM. Available online: http://www.d-grid.de/ (accessed on 31 January 2013).
6. ESP-Grid. Available online: http://wiki.oucs.ox.ac.uk/esp-grid/ (accessed on 31 January 2013).
7. Condor Shibboleth Merger Project. Available online: http://www.cs.wisc.edu/condor/ (accessed on 31 January 2013).

8. OIOSAML. Available online: http://digitaliser.dk/group/42063/resources (accessed on 31 January 2013).
9. Predic8. Available online: http://www.predic8.com/shibboleth-web-services-sso-en.htm (accessed on 31 January 2013).
10. Hardt, D. The OAuth 2.0 Authorization Framework. IETF RFC 6749, October 2012. Available online: http://tools.ietf.org/html/rfc6749 (accessed on 31 January 2013).
11. OASIS: Web Services Security: SOAP Message Security 1.1. OASIS Standard Specification. 1 February 2006.
12. Hurni, P.; Staub, T.; Wagenknecht, G.; Anwander, M.; Braun, T. A Secure Remote Authentication, Operation and Management Infrastructure for Distributed Wireless Sensor Network Testbeds, In Proceedings of the First Workshop on Global Sensor Networks (GSN '09), Kassel, Germany, 6–7 March 2009.
13. Want, R. iPhone: Smarter than the Average Phone. *IEEE Pervasive Comput.* **2010**, *9*, 6–9.
14. Coulson, G.; Porter, B.; Chatzigiannakis, I.; Koninis, C.; Fischer, S.; Pfisterer, D.; Bimschas, D.; Braun, T.; Hurni, P.; Anwander, M.; *et al*. Flexible Experimentation in Wireless Sensor Networks. *Commun. ACM* **2012**, *55*, 82–90.
15. Hurni, P.; Wagenknecht, G.; Anwander, M.; Braun, T. A Testbed Management Architecture for Wireless Sensor Network Testbeds (TARWIS). In Proceedings of the EWSN10, Coimbra, Portugal, 17–19 February 2010.