A Multi-protocol Authentication Shibboleth Framework and Implementation for Identity Federation

Mengyi Li[1], Chi-Hung Chi[2](✉), Chen Ding[3], Raymond Wong[4], and Zhong She[5]

[1] Tsinghua University, Beijing, China
[2] Data61, CSIRO, Canberra, Australia
chihungchi@gmail.com
[3] Ryerson University, Toronto, Canada
[4] University of New South Wales, Kensington, Australia
[5] IntelShare Initiative, Melbourne, Australia

Abstract. One of the challenges for Single Sign-On (SSO) is the multiprotocol federation in identity management. Even though projects such as Shibboleth provide good identity management framework, they usually support single protocol such as Security Assertion Markup Language (SAML). With the movement of increasing service collaboration in the cloud, identity federation needs to be extended to cover multiple identity protocol standards. In this paper, we propose an online distributed multi-protocol identity management framework Sh-IDaaS (Shibboleth-based Identity-as-a-Service) which could discover multiple user identity services in the Shibboleth environment. The framework enables federation of various identity services by binding different identity providers to a special discovery service, even if they support different identity protocols. Based on the Shibboleth framework, we describe the detailed design and implementation of our pluggable Sh-IDaaS architecture. Analysis of interoperability and performance of our Sh-IDaaS framework prototype is also provided to justify its feasibility and practicability.

Keywords: Authentication · Identity · Single Sign-On · Identity-as-a-Service

1 Introduction

With the movement of increasing collaboration among difference cloud service providers, identity access management has become a big issue. Users not only want to aggregate multiple independent component services together to form composite services, but they also want to subscribe to more than one composite service from their partners for business process orchestration. As a result, the overhead for service providers to manage the associated user identities has become a burden; and for users, paying extra efforts to handle identity related issues is also troublesome.

The idea of single sign-on (SSO) has been proposed to solve this problem. In addition, with the increasing demand for establishing a federated environment to achieve SSO among different service providers, SSO platforms such as OpenSSO [1]

R. Beyah et al. (Eds.): SecureComm 2018, LNICST 255, pp. 81–101, 2018.
https://doi.org/10.1007/978-3-030-01704-0_5

and Shibboleth [2], and identity management related protocols such as Security Assertion Markup Language (SAML) [3], Liberty Alliance [4], Information Cards [5], and OpenID [6], emerge almost simultaneously. However, despite the effort of SSO across multiple systems, one complication is that different service providers might have different security requirements and use different standards and protocols [7].

To address this issue, it would be beneficial if a distributed, online, third party identity management service can be provided. Such a service is known as Identity-as a-Service (IDaaS). By supporting different major identity management standards and protocols, IDaaS can help manage identity related issues of software services by binding itself to them, in a similar fashion as in service composition. This not only offloads the burden of identity management from the service providers, but it also provides a robust, cost-effective way of handling identity management for them.

When involving multiple service providers in an identity federation, identity discovery becomes an essential issue. The challenge here is, for enterprise federation in which each enterprise holds its own identity management service separately, how to discover an identity provider for users. Among SSO platforms currently available, Shibboleth [8] is one of the most popular open sources for local identity and access management [9]. However, Shibboleth by itself is still not a complete IDaaS solution because it only supports SAML protocol [3]. When resources are offered by independent service providers with different access requirements using different identity protocols, it does not function anymore.

In this paper, we propose a distributed, online, Shibboleth-based, multi-protocol authentication framework Sh-IDaaS (Shibboleth-based IDaaS) to address this problem. Even though the concept of cross-protocol federation has been around for some time, there is still no multi-protocol identity federation framework available. To achieve the extensibility, we design our Sh-IDaaS as a pluggable framework so that any identity provider can be plugged in through some sort of transformation according to its related identity management standard. Being able to federate third party services with different identity protocols is extremely attractive to users because "single sign on" for the usage of multi-protocol services becomes possible.

The outline for the rest of this paper is as follows. Section 2 gives background knowledge on two main identity protocols, SAML and OpenID, together with an introduction to Shibboleth-based identity federation. It also discusses the mechanisms of the multi-protocol authentication framework for identity federation, and analyzes the reasons of choosing Shibboleth as our base reference. Section 3 reviews related work on identity federation, in particular those related to Shibboleth environment. Section 4 describes how to integrate multi-protocol identity providers together using the concept of a convertor bridge and shows the overall architectural design of the Sh-IDaaS framework. Section 5 explains in detail the actual implementation of the pluggable Sh-IDaaS prototype system. The main focus is on the SAML convertor and the architecture of the Sh-OpenID (Shibboleth-based OpenID) provider. Section 6 illustrates the interoperability and extensibility of Sh-IDaaS with two representative use cases. Section 7 presents the results of the performance tests on the handling and conversion between the two identity protocols, SAML and OpenID. And finally Sect. 8 concludes the paper.

2 Background of Shibboleth-Based Identity Federation

2.1 SAML, OpenID, and Shibboleth

Security Assertion Markup Language, or SAML [10, 11], is an XML-based open standard for exchanging authentication and authorization data between Identity Providers (IdP) and Service Providers (SP). SAML Core refers to the general syntax and semantics of SAML assertions as well as the protocol used to request and transmit these assertions from one system entity to another. By defining XML-based assertions, protocols, bindings and profiles, SAML provides important support to Web SSO.

Among all the identity related standards, the most distinguished one on handling identity discovery is the open source OpenID framework [12, 13]. Different from SAML, OpenID focuses on user-centric digital identity [14]. Due to its decentralization nature, it is easy for users to sign up and access web accounts, replacing the common login process that uses the pair of login name and password. It allows a user to login once by a unique OpenID identifier (usually an "http" or "https" URL or XRI) and gain access to the resources of OpenID Relying Parties (RP). With the user centric feature, OpenID solves the troublesome pre-configuration problem which SAML demands. No central authority must approve or register RPs to OpenID Providers (OP). An end user can freely choose which OP to use.

Based on the SAML protocol, Shibboleth Federation Software [15] was developed to provide web SSO functionality, addressing many issues including accessing online resources both inside and outside an organization, requiring multiple username-password pairs for multiple software services, facilitating rapid, effective and secure integration of disparate third-party services, and so on. For example, one of the valuable benefits of Shibboleth is described as "Build and Manage Locally, Access Globally", which means that based on Shibboleth identity and access management system for applications, an organization can use it for federation of third-party collaborators. Shibboleth has already become one of the most widely adopted federated access management software in research and education communities. In this paper, our work extends the Shibboleth federation system as a multi-protocol Sh-IDaaS framework.

2.2 Discovery Service

One limitation of the SAML protocol is its restriction on binding between SPs and multiple IdPs. Because of the passive nature of SP, how to distinguish federated deployment (also known as IdP discovery service) comes up as a fundamental problem: is there an effective and efficient solution for the SP to recognize where to send the user agent? Although numerous solutions have been put forward, most of them lead to an increased interference due to per-site prompts, or other extra work for SPs. For instance, one straightforward approach is to ask users of each SP, but this requires each user to understand how to make unambiguous selection for the SP. Obviously, these solutions put extra burden on both users and SPs. In this paper, we focus on the WAYF (Where Are You From) service provided by Shibboleth; the idea is that discovery should be regarded as a service for large federation of IdPs. In this case, Discovery Service (DS) should act as a proxy for the SP, relaying a request to the selected IdP.

Discovery Service (DS) [16] can be used during the web-based SSO when an SP needs to establish an association with a particular IdP. In theory, DS is attractive since every SP can share a single point of discovery, and the user experience can seamlessly cross over all services. Figure 1 demonstrates how the DS integrates with SPs and IdPs and how the messages are exchanged between them; it illustrates the fundamental ideas behind Sh-IDaaS (it is assumed that the user uses a standard HTTP user agent [17]). Below we list the details of the workflow, which consists of 8 steps:

1–2. When a user visits an SP for resources, the SP will first redirect the user agent to DS, which is configured in advance with a set of parameters that make up the request.

3. From the user agent's point of view, with the HTTP Redirect request to DS, it will be provided an HTML webpage with a list of all available IdPs for the user to choose from.
4. The DS sends the user agent another HTTP Redirect request to a particular IdP selected by the user in Step 3 with an HTTP Request for the login page.
5. The DS interacts with the user via the user agent to associate one or more suitable IdPs with the validity check on user's username-password pair.
6. By confirming the validity of username and password provided, the DS then sends an HTTP Redirect request again to the user agent to forward the user back to the resource initially requested in Step 1. At the same time, the user agent receives a handle that can be forwarded to the SP together with the redirect request.

7–8. The SP forwards an attribute request directly to the IdP by sending the handle it just received from Step 6. At the IdP side, the handle received from the SP gets checked. If valid, the requested user attributes are transmitted to the resource according to the principal's authority provided by the SP.

Although there are many redirections that interact with the DS, most of them are actually transparent to the user; they happen between the SP and the IdP automatically. The user only experiences an additional process of IdP selection as shown in Step 3.

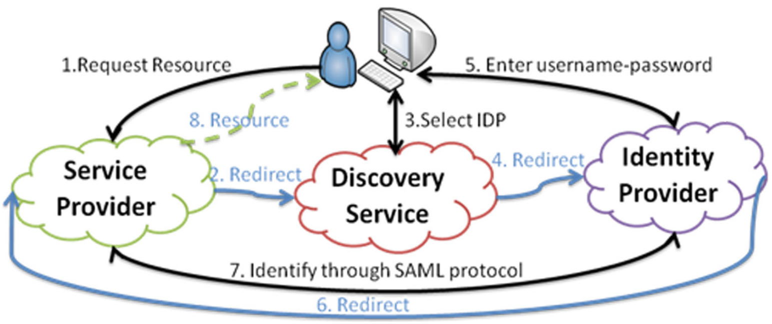

Fig. 1. Workflow between SP and IdP with DS

2.3 Analysis and Selection of Shibboleth

Even though the Shibboleth System extends the capability of Web-based applications by implementing identity management for secure access among multiple organizations, it only supports the SAML protocol. In real life, different protocols are likely to be used by different organizations. Therefore, it is very important to consider the situation when the identity federation contains multiple identity management related protocols. A more prominent related issue is whether it is possible to compose software services in the Shibboleth environment in which the identity management of these services could be handled by protocols other than SAML as well.

From the previous discussion, we argue that most of the identity related protocols (e.g., SAML and OpenID) have provided reasonable approaches for web SSO [18]. Even though these protocols use dissimilar methods on exchanging authentication and authorization data, they all rely on a third-party online service to identify the principal. Usually when a user requests resources from the SP, the corresponding IdP will pass an assertion to the SP via the DS (as shown in Fig. 1). On the basis of this assertion, the SP could make an access control decision. Based on this concept, we therefore put forward an online, extensible, identity management related, third-party service that supports multi-protocol authentication.

Through a comprehensive survey, we choose the open-source Shibboleth System (including IdP and DS) to be the basic authentication framework for our multi-protocol IDaaS because it meets our prototype requirements: Shibboleth is based on the SAML protocol, and it also handles account management with the DS. Some of the major questions that we need to ask include: What if IdPs based on different identity related protocols are interested to join in? How could the SPs connect to different IdPs sharing the same context to achieve the overall SSO? Since the combination between the SAML protocol and other identity related protocols will be the key topic of investigation in this paper, we will take OpenID, which is totally different from the SAML protocol as an example to illustrate our approach. If the framework can support the OpenID protocol, it will minimize the dependency on central servers or previous configurations, making entities more autonomous and capable of making trust decisions.

3 Related Work

In recent years, with the importance of federation issues being recognized by software services, there already exists some identity federation concepts [19–25]. Nenadic et al. [20] put forward an architecture aimed at providing multi-level user authentication service for Shibboleth. It points out that Shibboleth does not specify all the necessary components for authentication and authorization services. Therefore, by integrating FAME and PERMIS components into Shibboleth respectively, the whole architecture could provide a more complete solution for the IdP's local authentication system and the SP's access control engine. However, it leaves the task of integrating other authentication IdPs to the Shibboleth System. Hiroyuki and Takeshi [21] proposed a federation of hierarchical IdPs in the Shibboleth environment to solve the problem of complex structures of IdPs. They focused on the connection between organizations and sub-organizations through authentication delegation. However, the prototype they implemented only relies on the SAML protocol because they only used Shibboleth without considering other authentication components that may extend to different protocols. Almenárez et al. [22] mentioned the importance of federation for identity management and analyzed several federation frameworks such as the Liberty Alliance Identity Federation Framework, Shibboleth, OpenID and WS-Federation. Medsen [23] and Hatakeyama et al. [24, 25] also proposed the idea of federation proxy as the bridge for cross domain identity federation. However, it is still in the idea stage and no implementation has been done. Furthermore, information exchange among providers in these proposals is limited. There is some work on the design and implementation of web forward proxy with Shibboleth Authentication [26], but the solution does not address the multi-protocol issue.

There are "Identity 2.0" solutions such as OAuth [27] or Higgins [28] that have already integrated OpenID [29]. However, they inherit some security risks that OpenID has, including Phishing, replay attacks, and web application logic flaws. Kim et al. [19] proposed a common interoperable authentication framework regardless of the authentication methods and ID management technologies (e.g., SAML, Cardspace, and OpenID). However, it only presents the concept without defining the detailed contents such as authentication protocols, message formats, policies and operations to manage the ID lifecycle and other implementation details.

In this paper, we not only propose a well-defined authentication framework for cross-protocol identity federation, but also present the overall architecture design and implementation details, as well as the interoperability of the framework and the evaluation on its performance.

4 Proposed Solution

As mentioned in Sect. 2.3, we choose Shibboleth as the back-end authentication module so that IdPs that support other identity related protocols do not need to care about the information of principals' identities. In Sect. 3, we pointed out that the Shibboleth framework does not provide identity sources or authentication systems to keep the user information. Therefore, in order to use databases, Lightweight Directory Access Protocols (LDAP), or Kerberos connection, we need to use the Shib Proxy and the Data Connector which are integrated inside the Shibboleth project so that user's profile information can be shared in the databases or LDAP. However, there is one more intractable problem we need to solve, that is how to share the login context among multiple IdPs which adopt different identity protocols. For example, when a user logins in an IdP, through what mechanisms can other IdPs save that login session automatically, thus keeping the user in its own trust federation?

4.1 Convertor as Bridge Between Shibboleth IdP and Other IdPs

To address the issue mentioned above, we propose a universal framework that could integrate any identity protocols into the Shibboleth environment by plugging in a middleware – an identity convertor. Using this convertor as bridge between the Shibboleth IdP with other IdPs, IdPs supporting different identity protocols could share the same login context to achieve SSO when an end user sends a request for resources in the service provider. Such a middleware layer not only helps achieve the interoperability of the whole framework, but it also allows any new identity service could be added in without affecting existing services. Figure 2 illustrates the design of the middleware built to support multi-protocol IdPs in the Shibboleth environment.

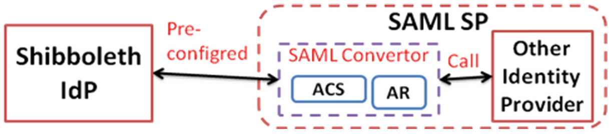

Fig. 2. Middleware design between Shibboleth IdP and other IdPs

In our design, we encapsulate a SAML convertor [30] (including ACS – Assertion Consumer Service and AR – Attribute Requestor components) that implements the basic authentication and authorization mechanism as the general SAML SP. Then we plug it into a regular IdP that may support any identity related protocol by taking the steps as we mentioned in Sect. 2.2. Note that in Fig. 2, the original IdP (the right part in Fig. 2) is now bridged to Sh-IdP. Therefore, from the perspective of a back-end Shibboleth IdP, Sh-IdP can be treated as a common SP using the SAML protocol when handling authentication and authorization. It can handle the original IdP authentication operation as well as the extended the SAML protocol to handle the final authentication process. With this architectural support, a generic structure that contains all the identity related information can be implemented using the middleware layer between Sh-IdP and other IdPs, providing the function of mapping service's identity related information from one format to the other.

4.2 Overall Architecture Design of Multi-protocol Framework

Figure 3 illustrates the overall architecture design of our multi-protocol authentication framework in the Shibboleth environment.

DS is located at the center of the framework, which is responsible for the "WAYF" service mentioned in Sect. 2.2. By presenting a front page containing a list of all the SPs, the DS allows an end user to choose which IdP for identification. Note that SPs are in fact bound to the DS directly. Any IdP that would like to join the identity federation could be transformed into a Sh-IdP by plugging a SAML convertor, and then delegating the original authentication request to the pre-configured Shibboleth IdP as shown in Fig. 2. The configuration between the Sh-IdP and the DS is the same as the deployment between the SAML SP and the DS. As long as it is configured correctly, Sh-IdP will be part of the multi-protocol framework to achieve SSO seamlessly.

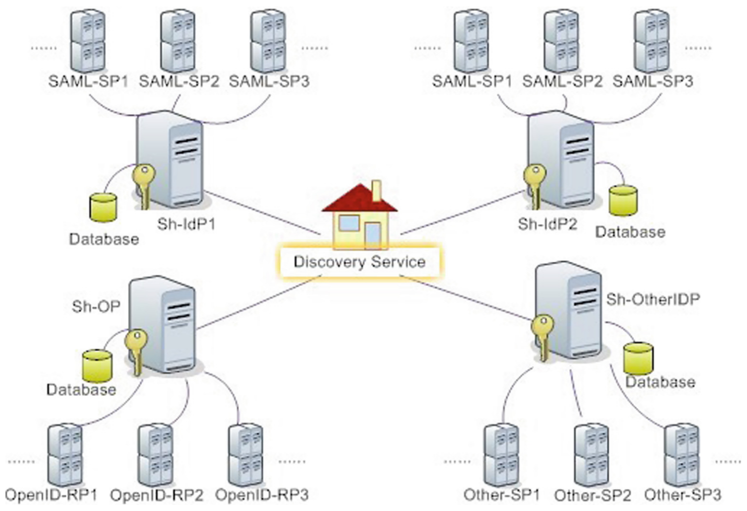

Fig. 3. Overall architecture design of our multi-protocol framework

5 Implementation of Pluggable Sh-IDaaS

5.1 Configuration of Shibboleth Environment

Shibboleth is a complex environment with multiple components and containers. The main components involved in our implementation of Shibboleth IdPs are: Apache 2.2 (version httpd-2.2.15) with mod_auth_mysql, mod_proxy_ajp, mod_ssl, Tomcat 6.0 (version apache-tomcat-6.0.26) and Shibboleth IdP (version shibbolethidp-2.0). We set SSO endpoint on port 443 by using X.509 certificate and then set Attribute Authority endpoint on port 8443 by using a self-signed certificate. In order to enable Tomcat to act as a back-end application server for the Shibboleth IdP and receiving authentication requests coming from Apache, we load mod_proxy_ajp built in Apache by adding a few requisite lines in httpd.conf.

Installation of Shibboleth IdP 2.0 is a standard procedure, just executing the install. sh file and configuring the path and hostname. However, to secure the traffic on the SSO login page and the communication between the SAML IdP and SP, X.509 certificates are required to be installed. This is to make sure that the users can verify that they are submitting their credentials to a server they trust. During the installation, we first create the key pair and the CSR (Certificate Signing Request) named idp.key and idp.crt by OpenSSL, and then load them into the metadata file folder of Shibboleth IdP. At the same time, we ensure the attribute query handler to be SSL-protected so that authentication of attribute requests is secured. These directives instruct the Apache server to request a client certificate during the SSL (Secure Socket Layer) exchange. The actual certificate processing is performed by Shibboleth, but Apache is still required to secure a connection to retrieve and feed the certificate to the IdP. This is the main reason to define a new SSL virtual host in ssl.conf on port 8443, then fill in the values of [SSLCertificateFile] and [SSLCertificateKey File] with idp.crt and idp.key which we generate in the IdP metadata file to refer to the proper key and certificate for the federation.

Since Shibboleth 2.0 does not authenticate principals, a relational database with a JDBC connector directory is recommended. Here, we choose MySQL as the database. To support MySQL, an authentication mechanism used to protect a < Location > block in Apache Basic authentication, mod_auth_mysql that is built in Apache is assumed to be in action accordingly.

5.2 Setup of a Traditional OpenID Provider

To implement a pluggable Sh-IDaaS by adding multi-protocol IdPs into the Shibboleth environment, we take OpenID Provider as an example in our prototype. The OpenID Provider (OP) is programmed in PHP and deployed on Apache Web Server. We implement the OpenID module based on some existing libraries and software packages that feature OpenID capabilities developed and maintained in openid.net [31] with version openid-php-2.2.2. Several main files and file folders we deploy are as follows:

- Folder OpenID mainly contains files that deal with OpenID server protocol and logic.
- Folder Yadis provides core PHP Yadis implementation, offering service discovery using the XRDS document format defined by OASIS.
- Folder Render contains resources and defines main pages for OP.
- File commer.php provides functions of doing or cancelling authentication between an OpenID server and a consumer.
- File session.php defines functions for session cookie actions.
- File action.php defines OpenID server action handlers. When converted into ShOP, the main file that needs to be modified for original OP is action.php. Since the functions defined in the action.php are encapsulated well, the modification of the authentication related function codes is minimal.

5.3 Implementation of SAML Convertor

The module we choose for the SAML convertor as a bridge between the Shibboleth IdP and OP is SimpleSAMLphp [32]. Since not all the modules provided by SimpleSAMLphp are needed, we extract the main modules that implement the SAML protocol as a simple SAML SP and is installed in our server as a SAML convertor. To make sure that it works well with the OP and functions as a new SAML SP, we put all the related folders into the same directory. To convert the original OP into a Shibboleth managed system, certain codes need to be modified. These include codes that communicate with the SAML convertor – the SAML SSO toolkit installed on the OP machine. All authentication issues would be delegated from OP to the SAML convertor. This delegation process has three steps. As the first step during the conversion, we need to request the SAML convertor library in the original OP. Next, in order to prepare the OP's SSO entry point, we invoke an SSO identification request, and then parse and analyze the SAML convertor's identification result. Finally, we redirect the user to his/her personal dashboard (the restricted target information) based on the information returned by the SAML convertor.

For the implementation, the main step is to convert the OP into Sh-OP is to change the original local authentication method by delegating the SimpleSAML_Auth_Simple object defined in the SAML convertor to pass the authentication operation to Shibboleth IdP, and then process authorization through the SAML protocol. Taking extensibility into consideration, we add a Sh-login function in the same file with the original login action function (both located in the action.php file of the original OP) instead of modifying the codes in the login action directly. We create a new object "$as = new SimpleSAML_Auth_Simple ('defaultsp')", and then call the method $as-> requireAuth() to establish the association between Sh-OP and Shibboleth IdP.

However, according to the SAML protocol, in order to allow the SAML convertor to collaborate with remote Shibboleth IdP servers, we need to specify the remote Shibboleth IdP's metadata in the SAML convertor's shib20-idp-remote.php file, including the attributes of entityid, metadata-set, certFingerprint, certData, SingleSignOnService, scopes, ArtifactResolutionService, and so on. All these information are located in the Shibboleth metadata folder we generated earlier.

To add a Sh-OP to the pre-defined Sh-IdP, the description of the middleware SAML convertor also needs to be added to the Shibboleth configuration files. This can be done by first creating the SAML SP metadata that contains the information required by the Sh-IdP for the converted Sh-OP, and then adding the formatted information of the SAML convertor in the relying-party.xml file that is located in the metadata folder of the Shibboleth IdP. This is a key step to enable the collaboration between the Sh-OP and the Sh-IdP via the SAML convertor. Note that the content of these files might be different according to the actual deployment of the SAML convertor middleware. Till now, we have implemented the conversion from an original OP into a special Sh-OP. The architecture and message flow details of the middleware framework will be given in the next section.

5.4 Architecture and Implementation of Our Middleware Framework

Figure 4 shows the middleware implementation we bring forward in the previous section. In our middleware framework model, we use Shibboleth 2.0 programmed in Java (as we described in Sect. 5.1) as the back-end IdP to perform the final authentication (the left part in Fig. 4). By encapsulating the SAML convertor into the regular OP and adding a Sh-SSO button in the front-page of the original OP, we transform it to a Sh-OP as we discussed in Sect. 5.3 (the right part in Fig. 4).

Through the SAML convertor, a user taking a web browser as the user agent might be able to share the login context by keeping a PHPSession cookie generated by ShOP and a JSession cookie generated by Shibboleth IdP even if they are using different programming languages and in a cross-domain environment. The main message flow Sh-OpenIDProvider handles for the cross-protocol authentication is depicted in Fig. 5.

Based on the high level architecture in Fig. 4, we integrate the converted Sh-OpenIDProvider and Shibboleth IdP with the middleware part built in the framework. We call it Sh-IDaaS (shown in Fig. 5). The main message flow is as follows:

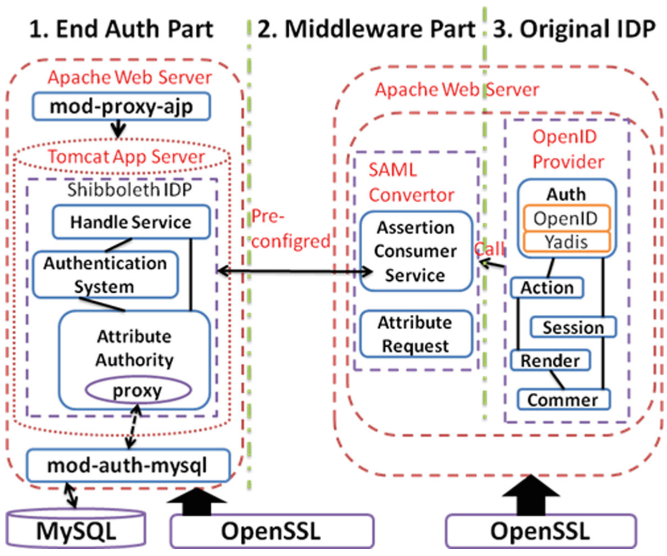

Fig. 4. Architecture of a Sh-OpenID provider

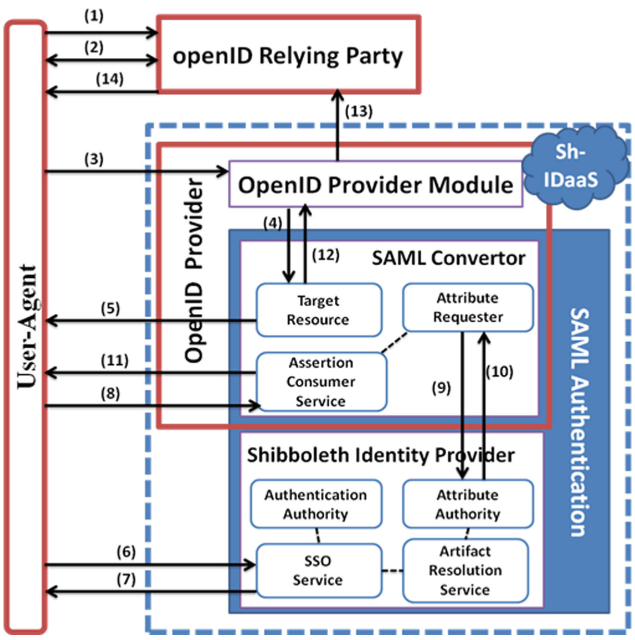

Fig. 5. Message flow inside Sh-IDaaS

1. The end user requests a target resource at the OpenID Relying Party (RP).
2. RP interacts with the end user for an OpenID URI. The user-agent is then redirected to a specified OP according to the OpenID specification.
3. The RP and the OP Module establish an association for the subsequent message flow. According to the OpenID Authentication 2.0 [29], the manner in which the end user authenticates to their OP and any policies surrounding such authentication is not in the scope of the document [6]. Hence, we assume that the end user chooses to use the federation authentication and presses the Sh-login button at the OP.
4. The OP delegates the ultimate authentication to the Shibboleth IdP by pretending to create a target resource request at the SAML convertor.
5. The SAML convertor redirects the user-agent to the SSO service at the Shibboleth IdP.
6. The user-agent requests the SSO service at the IdP. The SSO service performs a security check on whether a valid security context already exists. If not, the IdP identifies the principal. Once the principal is identified, the SSO service will obtain the authentication statement from the authentication authority.
7. The SSO service responds with a document containing an HTML form with the TARGET parameter being preserved in Step 6.
8. The user-agent issues a POST request to the assertion consumer service at the SAML convertor.
9. The assertion consumer service validates the request and dereferences the artifact created in Step 8 by sending a SAML request to the artifact resolution service at the Shibboleth IdP.
10. The artifact resolution service at the IdP returns an element containing the authentication statement to the assertion consumer service at the SAML convertor.
11. The assertion consumer service processes the authentication response, creates a security context at the SAML convertor and redirects the client to the target resource module.
12. The target resource returns to the OP Module with the assertion of the profile validation.
13. The OP Module uses the association established in Step 3 to sign messages which is sent to the OpenID RP. Then the OP redirects the user-agent back to the RP with a message about whether the authentication is approved or failed according to the assertion in Step 12.
14. The RP verifies those messages received from the OP Module and decides whether it is allowed to authorize the end user.

Step 1–14, as are shown in Fig. 5, are based on a simple use case which is mainly created to help the full understanding of the middleware framework. The major use cases that focus on the interoperability of the Sh-IDaaS architecture with multi-protocol identity federation will be discussed in the next section.

5.5 Implementation with Discovery Service

In Figs. 4 and 5, the Sh-OpenIDProvider and Shibboleth 2.0 IdP are physically separable if we configure the metadata in both SAML convertor and Shibboleth 2.0 IdP as the regular SAML SP and IdP do. Therefore, in our prototype implementation that combines SAML and OpenID protocol in the Shibboleth environment with the DS (as is shown in Fig. 3), we consider ShOP or Sh-OtherIDP as a traditional SAML SP. This implies that it can be configured with any Shibboleth IdP it trusts in the framework for its final authentication. Furthermore, we deploy a DS model as the center service and configure all the IdPs and SPs according to what are shown Fig. 3.

The DS we deploy is the version discoveryservice-1.0 from the Shibboleth project website. In order to perform the workflow of the Sh-IDaaS prototype, we deploy 11 Service Provider websites based on Drupal open source packages [33]. For each SP that supports local authentication originally, we add an SSO button in the front page, transform them into either OpenID supported or SAML supported SPs, and deploy them with a certain Sh-IdP or Sh-OpenID Provider and DS by configuring the metadata information respectively. Detailed workflow for the Sh-IDaaS prototype will be illustrated in the next section.

6 Interoperability of Sh-IDaaS Prototype

Interoperability means the ability to exchange and use information, usually in a large heterogeneous network. It is of extreme importance in a federation environment, where the sites in the community might use different federating software. Our pluggable Sh-IDaaS is designed for this purpose, offering multi-protocol support by ensuring that it will interoperate well with any software services.

To explain the interoperability of Sh-IDaaS with multi-protocol identity federation in details, two main use cases will be described in this section. Figures 6 and 7 portray the process how an end user using a web browser as a user agent handles Web-SSO through multiple protocols respectively. Here, we take Sh-OpenIDProvider and unconverted Shibboleth IdP as an instance.

6.1 Use Case When the SAML SP Is Visited First, then the OpenID Relying Party

Figure 6 shows the workflow when an end user first requests protected resources in SAML-SP with his/her username-password pair, then visits an OpenID RP with his/her OpenID identifier. Sh-IdP stands for Shibboleth IdP and Sh-OP stands for ShOpenIDProvider. The SP interacting directly with the Sh-OP is a common OpenID Relying Party (RP). The other SPs are common SAML SPs. As is shown in the figure, Step 1–6 explicate the process when an end user requests a SAML-SP for the first time. The interactions among the web browser, the SAML-SP, the selected Shibboleth IdP, and the DS are the same as Step 1–7 in Sect. 2.2. During the accessing process, Shibboleth IdP chosen by the user to implement the identity authentication would first gather a set of attributes about the principal using the attribute resolver by collecting the user data from the backend sources (in our implementation, we deploy MySQL by calling mod_auth_mysql in Apache2). Each attribute is attached with encoders for security purpose. The principal's information is thus packaged into a form ready for the response to be sent to the SAML-SP in Step 5 using the encoders attached earlier,

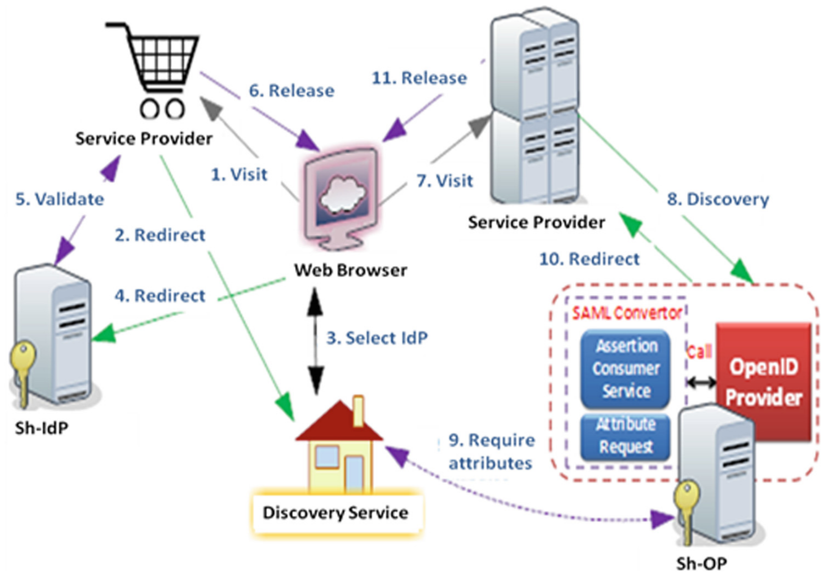

Fig. 6. Workflow of Sh-IDaaS (Use Case 1)

typically in a SAML assertion (called an artifact too). This artifact will be signed with the Sh-IdP's key and encrypted with the SAML-SP's key for security and privacy purposes. Then it is placed into a response that will be passed through the web browser back to an ACS endpoint in SAML-SP. On the other hand, in the web browser, a set of session cookies that contain the name-value pairs will be created by the DS, including an important session cookie containing a set of the principal's attributes released by the IdP. They may be encrypted for information privacy and data security purposes.

Subsequent accesses to the protected resource (Step 7–11) are granted directly until the timeout of the Shibboleth session. Afterwards, it will require a fresh handle to be issued by the DS. Without closing the web browser, the user visits the OpenID RP, filling in his/her special OpenID identifier offered by the Sh-OP (a formatted URL) portrayed in Step 7. Through the OpenID discovery process (Step 8), the web browser will establish an association to the Sh-OP. Although the original authentication phase is replaced by the SSO authentication in the Sh-OP, this is transparent to the RP. In fact, the interaction between the RP and the Sh-OP is totally the same as what the OpenID specification describes. The only difference is on how the Sh-OP validates the principal and the attributes the Sh-OP obtains from the principal. Sh-OP no longer acquires principal's attributes from an identity source or an authentication system by asking the end user directly. Instead, by preconfiguration with the DS, the SAML-convertor inside the Sh-OP would get the principal's information by resolving the HTTP response from the DS. SAMLconvertor transforms the information, and passes it to the Sh-OP. However, all these redirections between the SAML convertor and the DS are transparent to the end user. What he/she experiences is that he/she accesses the protected resources of the OpenID RP seamlessly with his/her attributes and authority issued by the Sh-IdP when he/she logs in earlier in Step 1–6.

6.2 Use Case When OpenID Relying Party Is Visited First, Then SAML SP

Figure 7 shows the interoperability of the Sh-IDaaS when an end user first visits an ordinary SP supporting identity protocol other than SAML. The workflow in this figure uses OpenID as the example. The case starts with an end user entering the OpenID RP with an OpenID identifier (Step 1). Once the RP has successfully performed discovery and created an association with the discovered Sh-OP endpoint URL, it can send an authentication request (an indirect request) to the Sh-OP to obtain an assertion. At the same time, the RP redirects the web browser to the Sh-OP for authentication (Step 2). The SAML convertor is called, performing as an ordinary SAML SP with the DS (Step 3–6). During this process, the end user should choose a Shibboleth IdP and identify himself/herself. The Shibboleth IdP then creates an artifact containing the principal's information for response to the SAML convertor and the DS keeps a set of session cookies shown in Step 1–6 in Fig. 6. According to the principal's attributes parsed from the SAML-convertor, the Sh-OP will decide whether the end user is authorized to perform the OpenID authentication. The Sh-OP redirects the web browser back to the RP with either an assertion that authentication is approved or a message that authentication fails by examining the "relay state" information returned (Step 7–8).

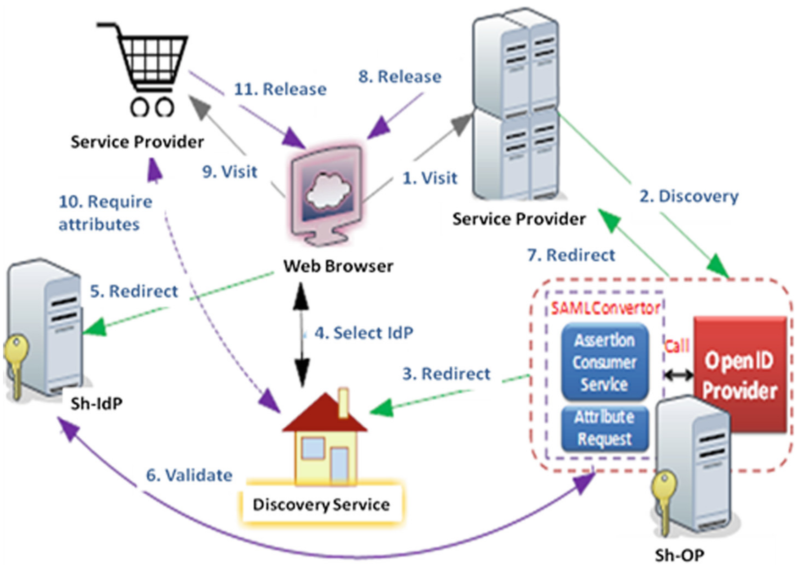

Fig. 7. Workflow of Sh-IDaaS (Use Case 2)

Without closing the browser, the end user accesses the SAML-SP (Step 9). At this time, the access occurs in the context of a set of session cookies stored within the web browser. Therefore the browser is finally redirected to the protected resource without login again as the normal SAML SSO process does in Step 9–11. All the complicated redirections are still transparent to end users.

7 Evaluation of Sh-IDaaS Prototype

In this section, we present the results of the performance tests on our Sh-IDaaS prototype. Since Sh-IDaaS contains many redirections as well as reading and writing operations which are all time consuming, we evaluate the performance of Sh-IDaaS mainly by analyzing the response time recorded in the logs.

There are two groups of experiments, one to test the basic SSO response time without using the discovery service (i.e. Group 1), and the other when using the Sh-IDaaS framework with the discovery service (i.e. Group 2). We implemented a set of test programs to simulate the accessing operations, allowing the web browser to request the resources from the SP continuously for several hours. The tests were carried out by tracking the redirections of the web browser. In order to avoid the interference between the cookies and the local cache in the web browser, we started a new process every time a new user logs on. The response time is defined as the elapsed time from the start time, which is the time when the web browser posts a resource request to an SP, to the termination time, which is the time when the web browser obtains the response resource successfully. In the test code, we filled in the prerequisite information in the login form for valid authentication such as a valid username-password pair.

In the Group 1 experiment, we collected 4 sets of data. The first set is the response time without any SSO protocols, which is the case when we simulate the general login operations that the system authenticates and authorizes the end user using the local database source built in the service provider. The second set is the response time with the OpenID protocol. To make the test more conveniently done, we make a small change to the implementation of the OpenID authentication by setting the variable "logsuccess" always as "ok" so that the test will keep going without halt. The third set is the response time with the SAML protocol. We also make a small change on the authentication code, just like what we do for the OpenID test. The last set is the response time with OpenID based on the SAML convertor, which is the case when the web browser requests for resources in an OpenID Relying Party by calling the authentication module built in the SAML convertor, and then redirects to a SAML IdP for authentication. The termination time is finally marked when the web browser responds with the valid resource. Each set of the data was collected for 2 h after the system has run for about 20 min under the same hardware condition. For about 2 h, we collected 10000 data points.

The distribution of the data is shown in Fig. 8. The x-axis is the timestamp and the y-axis is the response time of the requests in ms (mini-seconds). Since SAML and OpenID have been applied in a lot of communities with satisfying performance, we take the first 3 sets of data in Group 1 as the reference benchmark. Table 1 shows the

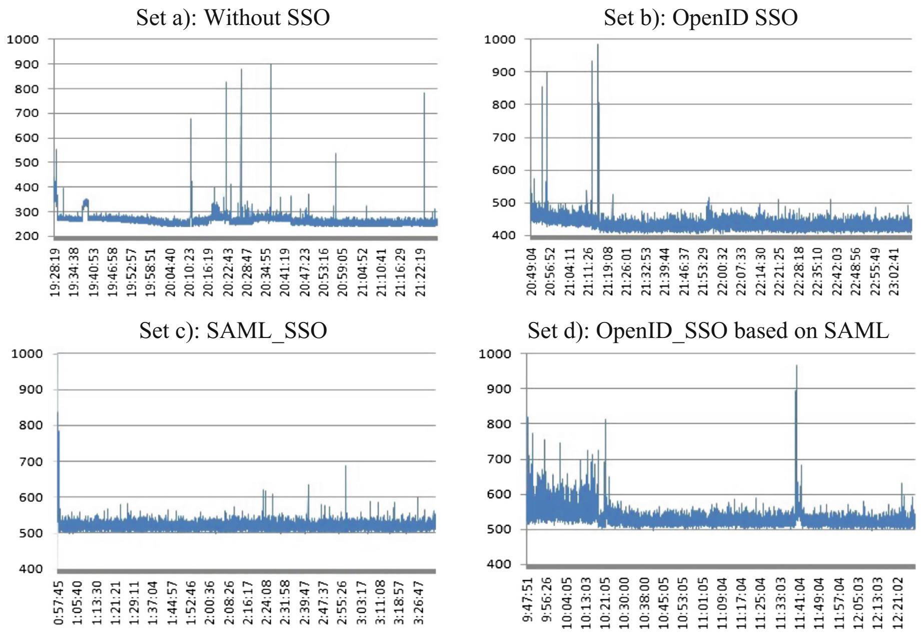

Fig. 8. Data collected from group 1 experiment on evaluation of Sh-IDaaS prototype

average response time and the standard deviation for each set respectively, which illustrates the dispersion degree of the response time under different conditions.

From Fig. 8 and Table 1, we can see that applications using OpenID or SAML services do have non-negligible overhead cost when compared to the one without using them. Furthermore, applications using SAML perform more stable than the others. Even though the average time of OpenID_SSO plus SAML_SSO is about 950 ms, when composing them together into the Sh-OpenID framework to perform the authentication process, its average time is 529 ms, which is just 9 ms more than the one for SAML_SSO alone. Therefore, the implemented prototype that plugs a SAML convertor into the OpenID authentication process has a relatively satisfactory performance.

Table 1. Data collected from group 1 experiment

	Average time (ms)	Standard deviation
With SSO	263.5781	26.2352
OpenID_SSO	434.4885	25.6883
SAML_SSO	520.5102	11.5978
OpenID based on SAML convertor	529.7865	22.1367

In the Group 2 experiment, we conducted the test based on the discovery service with more identity providers joining in. Due to the interaction between the end user and the DS, the processes we simulated are more complicated. The time consumed by operations such as choosing an IdP from the list shown in the front page of DS or filling in the login form is hard to estimate. Therefore, we collected another 4 sets of data under the assumption that the user has already logged in successfully to one of the service provider. And the response time we collected is actually the time taken by the redirections when doing the SSO.

In order to simulate the scenario, we set up 11 SPs as test models supporting either SAML or OpenID. For the data set (a), we established the connection between the SAML SP and a Shibboleth IdP through the DS first, and then requested resources from other SPs that also support the SAML protocol. Set (b) is for the scenario that an end user requests resources from a SAML SP first, and then other SPs that support the

Table 2. Data collected in group 2 experiment

	Average time (ms)	Standard deviation
Fed_SAML & SAML	484.3457	37.2062
Fed_SAML & OpenID	681.4230	40.4654
Fed_OpenID & SAML	443.8571	20.9998
Fed_OpenID & OpenID	653.5704	17.5313

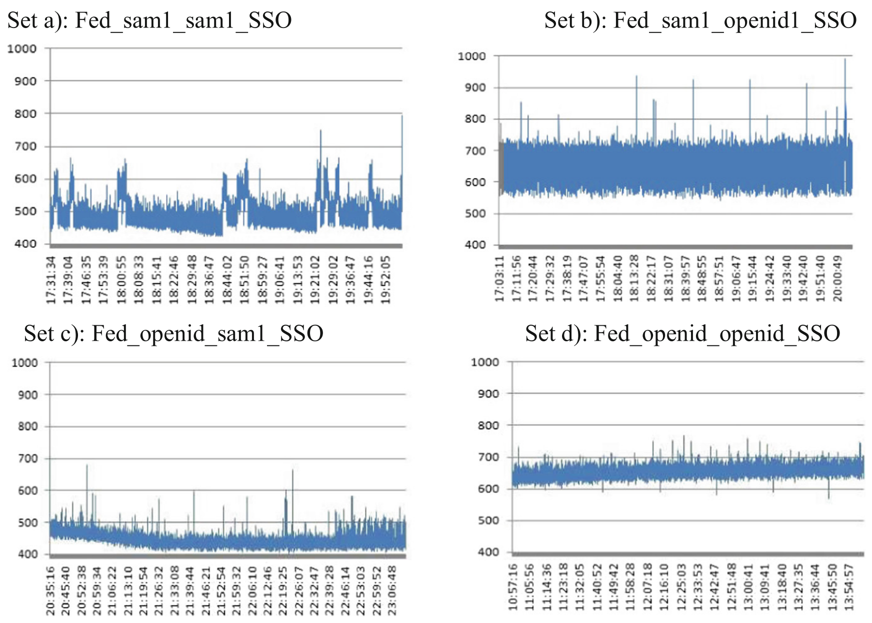

Fig. 9. Data collected from group 2 experiment on evaluation of Sh-IDaaS prototype

OpenID protocol. Set (c) is for the scenario that a user logs into an OpenID RP first, and then other SPs that support the SAML protocol. Set (d) is for the scenario that an end user successfully logs in through the OpenID protocol, and then he/she visits other SPs that also support the OpenID protocol. In our experiment, we choose data set (a) as the reference benchmark. The distribution of the data is shown in Fig. 9. The x-axis is the timestamp and the y-axis is the response time for the requests in ms.

Table 2 shows the average response time and the standard deviation calculated for each set respectively. From Fig. 9 and Table 2, we can see that in the Sh-IDaaS framework, a SSO service does have the time overhead. Since the redirections between the OpenID RP and the Shibboleth IdP through the DS operate more frequently, with the SSO OpenID RP, it almost takes 200 ms more than that for the SSO SAML SP. However, compared to the average time cost for Set (a), other sets of data are all less than 1 s, which should be acceptable by the end user. The standard deviation shown in Table 2 shows better result when compared to data of Set (a). Therefore, the pluggable Sh-IDaaS prototype we design and implement has a relatively satisfying performance.

8 Conclusion

In this paper, we propose a new Sh-IDaaS framework to address the challenge of controlling access to resources offered by multiple third-party providers supporting different identity protocols. The Sh-IDaaS prototype we design and implement achieves the combination not only between SAML IdPs but also with OpenID Providers. By supporting the OpenID protocol in the Shibboleth framework, the Sh-IDaaS solves the pre-configuration problem which SAML has. We implemented the framework by adding a convertor to delegate the authentication requests to SAML SSO in the Shibboleth framework together with the Discovery Service and the Identity Provider. Furthermore, a more generic proposal for identity federation by designing a pluggable multi-protocol authentication framework in the Shibboleth environment is also given. From the experiment results, we can see that the performance of the Sh-IDaaS prototype is satisfactory, which demonstrates both the feasibility and practicability of our proposal.

References

1. OpenSSO. https://opensso.dev.java.net/
2. The Shibboleth Project 2007. http://shibboleth.net/
3. OASIS Security Assertion Markup Language (SAML) V2.0, April 2005. http://www.oasis-open.org/
4. The Liberty Alliance Project. http://www.projectliberty.org/
5. Nanda, A.: Identity selector interoperability profile V1.0. Microsoft Corporation (2007)
6. OpenID Specifications, OpenID Foundation (2007). http://openid.net/developers/specs/
7. Blaze, M., Kannan, S., Lee, I., Sokolsky, O., Keromytis, A., Lee, W.: Dynamic trust management. IEEE Comput. **42**(2), 44–52 (2009)

8. Cantor, S. (ed.): Shibboleth Architecture. Protocols and Profiles, 10 September (2005). https://wiki.shibboleth.net/confluence/download/attachments/2162702/internet2-mace-shibboleth-archprotocols-200509.pdf
9. Grimm, C., Groeper, R.: Trust issues in Shibboleth-enabled federated grid authentication and authorization infrastructures supporting multiple grid middleware. In: Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing, pp. 569–576 (2007)
10. Ragouzis, N., et al.: Security Assertion Markup Language (SAML) V2.0 Technical Overview. OASIS Committee Draft, Document ID sstc-saml-tech-overview-2.0-cd-02, March (2008). http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf
11. Lewis, K.D., Lewis, J.E.: Web single sign-on authentication using SAML. Int. J. Comput. Sci., **2** (2009)
12. Reed, D., Chasen, L., Tan, W.: OpenID identity discovery with XRI and XRDS. In: Proceedings of the 7th Symposium on Identity and Trust on the Internet, pp. 19–25 (2008)
13. Recordon, D., Reed, D.: OpenID 2.0: a platform for user centric identity management. In: Proceedings of the 2nd ACM Workshop on Digital Identity Management, pp. 11–16 (2006)
14. Rieger, S.: User-centric identity management in heterogeneous federations. In: Proceedings of the 4th International Conference on Internet and Web Applications and Services, pp. 527–532 (2009)
15. Barton, T., et al.: Identity federation and attribute-based authorization through the globus toolkit, Shibboleth, GridShib, and MyProxy. In: Proceedings of the 5th Annual PKI R&D Workshop (2006)
16. Widdowson, R., Cantor, S. (ed.): Identity Provider Discovery Service Protocol and Profile. 27 March (2008). http://www.oasis-open.org/committees/download.php/28049/ sstc-saml-idpdiscovery-cs-01.pdf
17. RFC 2109: HTTP State Management Mechanism, http://www.ietf.org/rfc/rfc2109.txt
18. Hodges, J.: Technical Comparison: OpenID and SAML, Draft 6. 17 January (2008). http://identitymeme.org/doc/draft-hodges-saml-openid-compare-06.html
19. Kim, S.H., Jin, S.H., Lim, H.J.: A concept of interoperable authentication framework for dynamic relationship in identity management. In: Proceedings of the 12th International Conference on Advanced Communication Technology, pp. 1635–1639 (2010)
20. Nenadic, A., Zhan, N., Chin, J., Goble, C.: FAME: adding multilevel authentication to shibboleth. In: Proceedings of IEEE Conference on e-Science and Grid Computing, p. 157 (2006)
21. Hiroyuki, S., Takeshi, N.: Federated authentication in a hierarchy of IdPs by using shibboleth. In: Proceedings of the 11th IEEE/IPSJ International Symposium on Applications and the Internet, pp. 327–332 (2011)
22. Almenárez, F., Arias, P., Marín, A., Díaz, D.: Towards dynamic trust establishment for identity federation. In: Proceedings of the Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship, Article No. 25 (2009)
23. Madsen, P.: Proxy Assurance Between OpenID & SAML (2009). http://kantarainitiative.org/confluence/download/attachments/3408008/ntt-madsen-rsa/concordia.pdf
24. Hatakeyama, M., Shima, S.: Privilege federation between different user profiles for service federation. In: Proceedings of the 4th ACM Workshop on Digital Identity Management, pp. 41–50 (2008)
25. Hatakeyama, M.: Federation proxy for cross domain identity federation. In: Proceedings of the 5th ACM Workshop on Digital Identity Management, pp. 53–62 (2009)

26. Takaaki, K., Hiroaki, S., Noritoshi, D., Ken, M.: Design and implementation of web forward proxy with shibboleth authentication. In: Proceedings of the 11th IEEE/IPSJ International Symposium on Applications and the Internet, pp. 321–326 (2011)
27. OAuth. http://oauth.net/
28. Higgins. http://wiki.eclipse.org/Higgins_2.0/
29. OpenID Authentication 2.0 Final, 5 December (2007). http://openid.net/specs/openid-authentication2_0.html
30. SWITCH AAI ArpViewer, http://www.switch.ch/aai/support/tools/arpviewer.html
31. OpenID.Net, http://openid.net/developers/libraries/
32. SimpleSAMLphp. http://SimpleSAMLphp.org/
33. Drupal. http://drupal.org/