

A Unified Authentication Framework for Accessing Heterogeneous Web Services

Aaron Moss, Sandy Liu and Rene Richard

Abstract—One under-addressed issue in the field of Web Services composition is authentication between disparate services using different authentication methods or protocols. A Single Sign-On (SSO) framework reduces the burden on the end user to provide authentication credentials to these separate services; thus it is a desirable feature for systems and applications that are based on multiple Web Services. However, true SSO is not feasible in a Web Services context, as individual services can be provided by any parties; they may have arbitrary authentication methods, credential types, or protocols, and may not have an existing trust or federation arrangement with a given external authentication system. Due to these factors, it is valuable to provide a specialized service that can provide authentication information seamlessly to the services selected for a given service-enabled process. This paper introduces a unified authentication framework for accessing heterogeneous Web Services. We propose a credential storage and retrieval mechanism to store authentication data and pass that data to corresponding Web Services clients. Hence, this framework enables authenticated access to Web Services implemented with arbitrary access control methods.

Index Terms—Authentication, Credential Storage, Single Sign On, Web Services, Virtual Organization.

I. INTRODUCTION

WE are evolving into the service computing paradigm. Many software applications and processes are now built on Web Service-enabled components. By using open standards, protocols, and interfaces, any organization or individual can become a Web Service provider. The heterogeneous and independent nature of Web Services prompts to many challenges for service composition. Beyond the problems in match-making, there are also challenges in bridging incompatible data models and access control methods, etc.

Despite the challenges, with the growing accessibility of the Internet and the sophistication of the middleware tools, many people are working with remote collaborators under the notion of Virtual Organizations (VOs) [21]. A VO usually consists of a group of geographically distributed members and resources. Using the state-of-the-art cyberinfrastructure services, members in a VO can work coherently as a whole to conduct

scientific research, industrial design, or solve business problems. Typically each VO shares a pool of service-enabled resources and communication-enabled tools that may include software applications, hardware, data collections, computational power, storage, and even specialized applications for configuring optical private networks [17], [22]. Naturally, in this type of environment, there will be complex requirements for access control and authorization between the separate entities, with the possible inclusion of multiple unchangeable legacy systems. We developed a Service-oriented Architecture for Virtual Organization Infrastructure and Resources (SAVOIR¹) [5] [6] [7], to facilitate the coordination of resources in collaborative sessions.

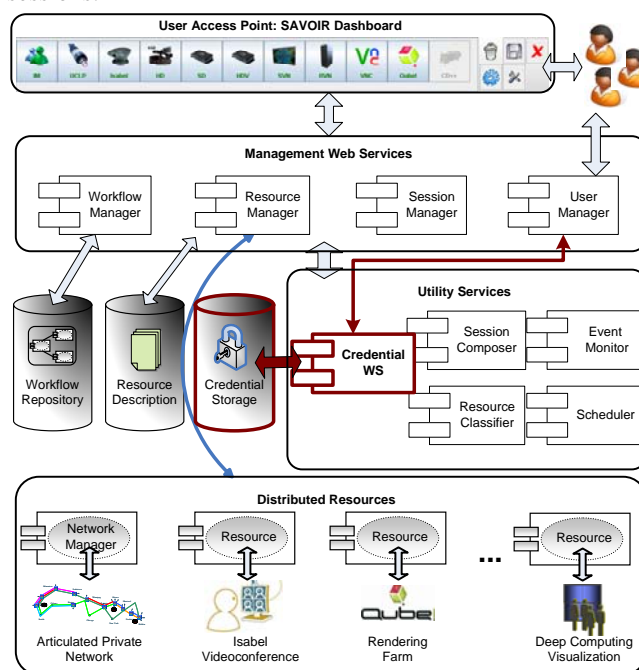


Fig. 1 The Authentication Framework embedded in SAVOIR

In SAVOIR, the pool of shared resources is made available by participants in a VO in the form of Web Services. As shown in Fig. 1 the SAVOIR dashboard acts as an integrated service client for accessing resources in a VO. The set of resources that appear on the dashboard is customizable. Each resource is represented as a widget that can be added or removed on demand. Thus the user can choose what widgets to appear in

Manuscript received December 5, 2008.

Authors are with the Institute for Information Technology, National Research Council Canada (corresponding author: Sandy Liu, phone: 506-444-0497; fax: 506-444-6114; e-mail: sandy.liu@nrc.gc.ca).

¹Previously called Eucalyptus.

the dashboard. The core part of SAVOIR includes a set of management services that manages users, resources, sessions, and session workflows with the assistance of a set of utility services. A SAVOIR session may involve the usage of multiple resources provided by different organizations participating in the VO. However, since each organization may have different and incompatible access control mechanisms, each user in the VO may be given an independent set of credentials to use with each individual resource.

To mitigate the problem of having to provide a different set of credentials to every resource participating in a SAVOIR session, we developed a unified authorization framework. This framework can be integrated into any system built on a collection of heterogeneous Web Services. It facilitates authentication between disparate services using different authentication methods or protocols. Fig. 1 shows how the credential service interacts with other components in SAVOIR. The rest of this paper describes the requirements, design and implementation of such a framework, followed by a discussion and a review of some related works.

II. REQUIREMENTS AND DESIRED FEATURES

As it is undesirable to have the end user submit multiple credential sets for a single action, some form of Single Sign-On (SSO) architecture is a useful feature for applications that require access to heterogenous Web Services. A SSO system has other benefits, including capabilities for credential delegation. In the case of service-oriented framework such as SAVOIR, a SSO system has the added benefit of requiring the end user to input only a single set of credentials, instead of multiple credentials for each service, ameliorating the well-documented problem of password fatigue [2].

implementation. Fig. 2 shows a sample SAVOIR session calling for a process that retrieves an architecture design model from a rendering Web Service, and then using a visualization Web Service to visualize the result. This process requires the composition of two Web Services, each of which has a separate authentication scheme, and cannot communicate directly with the other. It is unreasonable to ask the user to enter multiple credentials, or to have the end services change their authentication schemes to suit a composition, so a credential retrieval system becomes very useful. Fig. 2 also illustrates how the SAVOIR User Management Web Service relies on the Credential Web Service to retrieve the right credentials and communicates back to the session management Web Service to further invoke the session workflow.

Beyond simply enabling storage and retrieval for arbitrary credential types, there are other design goals that are required for a service-oriented SSO framework. Foremost among these was extensibility. As new Web Services can be added to an existing system over time, it is important that new services, with new types of credentials, can be added without any change to the system itself. This ensures that client developers do not need to learn the implementation of the storage system, instead concentrating on their primary task, service access. This also reduces system downtime due to upgrades, and lets existing credential definitions be imported more easily.

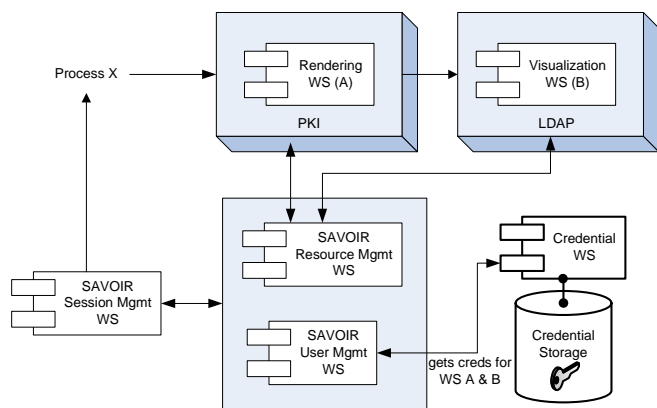


Fig. 2 Credential Storage enabling service Composition

Being in a Web Services environment, typical SSO implementations were not sufficient, in that they usually require some sort of prior arrangement between the client and the server - either some sort of existing trust arrangement, or support for a specific federation protocol. In many cases, it is not possible to impose such agreements among different service providers. Therefore, we assume that each Web Service is a black box, and we do not have any influence on its

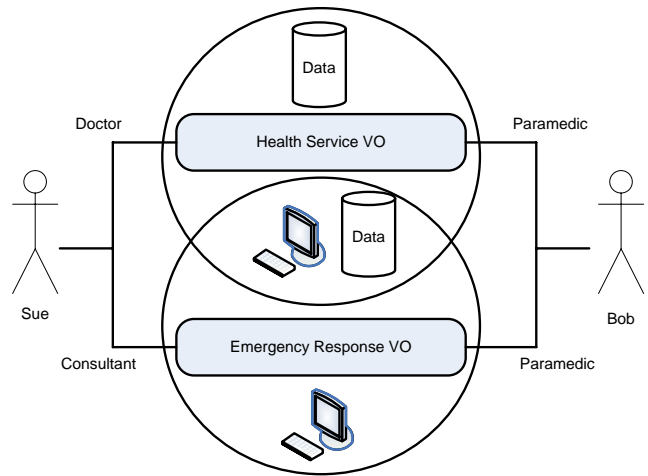


Fig. 3 Users Accessing Resources through VOs

Besides extensibility, there are two related features that can provide practical value to the system. One is the ability for a single user to have more than one credential set for a given service, and the other is the ability to assign a single credential to some set of users. For instance, in the example shown in Fig. 3, the Health Services VO may have a single password to the shared storage resource, and all consultants to the Emergency Response VO may share a password for read-only access to the same resource. If Sue wishes to access that storage resource, she should have the option to choose between her Emergency Response consultant or Health Services member credentials. If these credentials are granted to a group of users, there is no

need to store duplicate copies for each user, which would be error-prone. On the other hand, if, for instance, Bob resigns from the Health Services VO, his membership in the organization may be revoked with one operation, as opposed to individually removing his credentials for every service he was authorized to use through that membership.

Another potential requirement or feature for a service-oriented SSO system is its ability to integrate with existing authentication systems, possibly non-Web Services based such as Kerberos [10] or Lightweight Directory Access Protocol (LDAP) [13]. This furthers the goal of SSO by not requiring the user even to submit any credentials to SAVOIR or other systems that provide access to Web Services-enabled composite services.

Finally, access to this system should be exposed as a Web Service, as the other components of the SAVOIR framework are. This requires strong authentication and encryption, to maintain the security of the sensitive authorization data contained in the system.

III. SYSTEM DESIGN OVERVIEW

Our unified authentication framework is designed to function as a standalone Web Service, although it can also work as an integral component in SAVOIR. It has a common login and authentication shared with SAVOIR. Credentials are stored and transmitted as XML documents, with an XML Schema Description associated with each service for validation purposes. When a user requests credentials, all the credentials the user is authorized to access are returned to them as XML, with some metadata describing credential ownership and purpose - the service client is responsible for parsing the XML and passing the credentials to the end service. Storage and transmission of credentials, as well as other ancillary functions, such as addition of services, shall be provided by a sanitized and well-defined Web Services interface, to reduce the risk of system attackers gaining unauthorized access to the database.

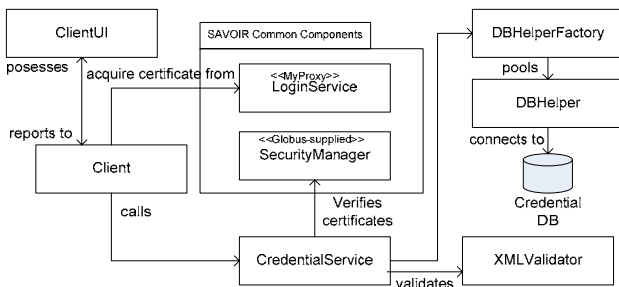


Fig. 4 Simplified Class Diagrams for the Core Entities

The credential storage and retrieval system naturally needs very strong authentication, and is thus highly dependent on the login system of SAVOIR. Fig. 4 shows the relationships between key classes in the credential system, while Fig. 5 shows an example execution flow for login and subsequent service access. The basic concept is that users log into the

system once, receiving a token that both identifies and authenticates them. When a user wants to access a secured resource, the token is presented to the SAVOIR credential service, which then returns any access credentials that this user is authorized to use for that resource. The service client then passes an appropriate credential set to the corresponding Web Service, and thereafter the user may use that service as they wish.

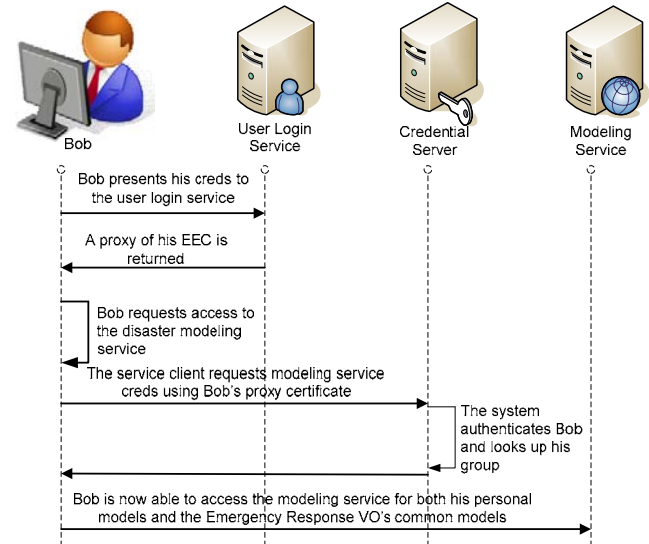


Fig. 5 The Execution Flow for Service Access

Our prototype implementation of the framework builds on Globus Toolkit's [20] Grid Security Infrastructure (GSI), mainly for the system's internal security and authorization. GSI supports WS-Security [11] and WS-SecureConversation [15] for message-level privacy and integrity, as well as TLS (formerly SSL) for transport-level security. These security mechanisms provide encryption, authentication, and message integrity, and can be combined for a better functionality to performance balance. Our prototype enables GSI by including some Globus-supplied handlers in an Apache Tomcat Servlet container [18] for server-side support, and on the client with an Apache Axis SOAP engine [19] configured to use some other Globus handlers.

One feature of GSI is support for RFC 3820 proxy certificates [25]. These proxy certificates, an extension to the standard X.509 certificates of Public Key Infrastructure (PKI) [24], allow a short-term certificate, possibly with constraints on rights, to be signed by a X.509 End Entity Certificate (EEC). (A short description of PKI and X.509 concepts can be found at [16].) In this case, the EEC acts much like a Certificate Authority. The useful application of this is that a user's private key and EEC can be stored in a secured location, and used only to sign a proxy certificate, which can then be used by the user for the duration of the session. This reduces the risk that the user's private key may be compromised, as it can be stored in a more secure location than a local filesystem, and is used in fewer communications. To provide users with proxy certificates (the authenticating and authorizing token used by

our login and authorizing services), we use a MyProxy credential management system [1]. MyProxy has support for multiple login authentication methods, including Kerberos, LDAP, Pubcookie [23], etc. In addition, another benefit to use MyProxy is that it has flexible user administration capabilities. Leveraging these benefits, we developed an administration service, with configurable, assignable, role-based access permissions, to group and manage users.

To achieve flexibility and extensibility for credential storage, credentials are stored as XML documents. This allows both definition of arbitrary types of credentials, and simple inclusion of existing XML-based credential types. Furthermore, storing and transmitting credentials in XML grants both server and client-side code language independence.

For the credential storage, MySQL [9] is used as our database engine, though any JDBC-compliant database would work. Each user of our portal corresponds to an entity in the database – these users are mapped to the Distinguished Names on the EEC's they hold in the login system. That mapping associates each user of our portal with their authorization rights when they access the storage system (providing their proxy certificate, which includes their Distinguished Name).

Users can be aggregated into groups. A group, which is represented by an entity in the database, is simply a way to aggregate users for easier allocation of credentials to a related set of people. For instance, if all students in a university need secured access to a course calendar and school phone directory, they can be added to a "Student" group, which can then be assigned access rights to these resources. Groups may also contain other groups, such that all rights of the supergroup are applied to the subgroups, but not vice-versa. This allows for fine-grained access-control, while preserving a logical hierarchy of groups. To allow maximum flexibility, a given user may belong to any number of groups, and a given group may have any number of supergroups and subgroups, nested to any level.

Each resource is also represented by an entity. These resources entities each have a unique name, and a reference to the XML Schema Description of their access credentials. Figure~\ref{xml_schema} shows a sample schema for a username and password at the top, with a corresponding credential shown at the bottom. The XML Schema is stored separately as some common types, such as username and password, may be used by multiple services. On the other hand, each credential set has a short text description stored along with the XML data of the authentication information. This description is to aid in choosing among multiple credentials a given user is authorized to access.

Finally, the core of the credential lookup system is the *authorizes* relation, which ties everything together. This relation maps a user or group and a resource to a credential. The only constraint on this relationship is that each record must be unique. This ensures that any user (or group of users) may have rights to any number of credentials, for any number of resources, and any single credential may be used by any number of users for any number of resources (as in the case

where we wish to access multiple external services sharing a login - a portal for instance, or if we wish to give two separate groups access to the same resource using the same password).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="cred">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="uname"
                    type="xs:string"/>
        <xs:element name="pword"
                    type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
...
<cred>
  <uname>sue</uname>
  <pword>g1bb3r15h!</pword>
</cred>
```

Fig. 6 Sample Credential Schema

This design provides virtually unlimited flexibility in credential storage, while still providing a unified method of storage and retrieval. The implementations of user grouping and credential authorization are also fairly loosely coupled, giving us the opportunity to use the user management system in other parts of the SAVOIR system, or even to replace it with a different implementation or model, without changing the code of the other components of the system, including the credential store.

IV. DISCUSSION AND RELATED WORKS

One common objection to SSO systems is that they provide a single point of failure for an entire architecture. The system we have designed is not as susceptible to failings of this sort, as the actual transmission of credentials is the responsibility of the client application. Thus, in the event of a service outage, a well-designed client application may provide appropriate fallback capabilities, such as asking the user to resubmit their credentials, as they would to originally store them in the system. Another objection, that a SSO system provides an attacker a single location from which to steal all of a user's credentials (the "key to the kingdom" argument) is a problem that is inherent in the nature of the system. Our system does provide flexibility for the implementation of arbitrary encryption techniques in storage, strong message security, and, on an implementation level, a modular design that eases addition of additional security functionality at a future date. These design features, combined with competent system setup and administration, should mitigate "key to the kingdom" style issues to a degree that is acceptable for many applications.

While there is significant work taking place in multiple contexts to solve the problem of SSO, the key differentiator of our project is that it does not presuppose that the a Web Service user can make any change to the end services. Changes include adding new authentication protocols, or simply setting up some new trust arrangement. This leads to a separation of responsibility between the service portal and client applications

working through it, such that storage of data needed for the provision of credentials is handled by the portal's credential storage system, while actual communication with the end service is enabled by the service client. Although we may add functionality to the portal at a later date to support common authentication protocols, such as Kerberos [10], that is outside the scope of this system. The following section briefly describes existing work in the field of SSO, and explains the differentiating factors in presumptions or applications provided by our system.

Many SSO systems are designed to work in the domain of a single institution, and thus require that end services trust the institution's login credentials. While these systems are useful within the confines of the services of one entity, the requirement for high trust between the end services and the authentication service makes them significantly less useful in our use case, considering varied Web Services. CoSign [14], a Kerberos-based system using browser cookies and developed by the University of Michigan, is one such solution. CAS [4], a similar browser-based system built on Kerberos, originally developed by Yale, is another. One more system is Pubcookie [23], a cookie-based system that follows principles similar to Kerberos. A further disadvantage of all these systems in a Web Services environment (which, admittedly, they were not designed for) is the difficulty in retrieving browser cookies from a Web Service.

Another SSO system designed for the World Wide Web is OpenID [12]. This technology provides a protocol for "relying parties", entities the user wishes to authenticate to, to correspond with "identity providers", entities that handle user authentication, and pass this data to relying parties, to establish a user's identity. Thus, a user can have one identity provider, which all the relying parties they wish to authenticate to can communicate with, giving the user a single point of sign on. While this technology shows future promise, it is currently used nearly exclusively on the World Wide Web, as opposed to Web Services, and requires support for the OpenID protocol, contradicting our design assumption that any end service may have a completely arbitrary authentication protocol.

While there are multiple Web Services standards and languages to provide SSO, these are primarily focused on federation, the transferral of trust between domains. These protocols, while they may have greater future impact, do not provide access to arbitrary Web Services using legacy authentication methods. WS-Federation [3], a protocol for the exchange of trust data between federated realms, is one example. The WS-Federation protocol provides a framework for the transformation of trust through organizational boundaries. While this may be useful for SSO within domains with strong and well-configured existing trust relationships, it is not effective in accessing arbitrary Web Services, as existing trust relationships cannot be assumed. SAML [8], an XML-based language for conveying assertions about identity, attributes, and rights of principles between services, was designed to enable SSO. While it is a useful building block in many other protocols, it still does not solve the issue of

preexisting trust relationships, and also requires the end service to be able to parse it.

V. CONCLUSIONS

We have developed a system to emulate SSO in a heterogeneous Web Services environment, freeing the end user from repeatedly entering service credentials. This was motivated by the use case of composing multiple services, each with distinct authentication schemes, assuming it is unreasonable to expect to be able to change the end service to suit our authentication, but an unnecessary load on the end user to submit multiple credentials for the same task. We introduced a unified authentication framework which stores service credentials in XML-based formats, and associates those credentials with users, roles, or organizations. This is a practical approach to deal with authentications for service-oriented systems. The key factors distinguishing our framework from other SSO implementations is that it is not limited to a single trust domain or authentication protocol, and that it is capable of storing arbitrary types of credentials, instead of a constrained set.

REFERENCES

- [1] Board of Trustees of the University of Illinois. MyProxy Credential Management Service. <http://grid.ncsa.uiuc.edu/myproxy/>, 2008.
- [2] D. Carstens, P. McCauley-Bell, L.C.Malone, and R. DeMara. Evaluation of the human impact of password authentication practices on information security. *Informing Science Journal*, 7:67.86, 2004.
- [3] M. Goodner, M. Hondo, A. Nadalin, M. McIntosh, and D. Schmidt. Understanding ws-federation. <http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-FederationSpec05282007.pdf>, May 2007.
- [4] JA-SIG. CAS 2 Architecture. http://www.ja-sig.org/products/cas/overview/cas2_architecture/index.html, 2006.
- [5] S. Liu, Y. Liang, and M. Brooks. Eucalyptus: A Web Service-enabled e-Infrastructure. In B. Spencer, M.-A. Storey, and D. Stewart, editors, *Proceedings of CASCON 2007*, pages 1.11, 10 2007.
- [6] S. Liu, Y. Liang, B. Xu, L. Zhang, B. Spencer, and M. Brooks. On demand network and application provisioning through web services. In *IEEE International Conference on Web Services (ICWS)*, pages 1120 - 1127, July 2007.
- [7] S. Liu, B. Spencer, Y. Liang, B. Xu, L. Zhang, and M. Brooks. Towards an Agile Infrastructure to Provision Devices, Applications, and Networks: A Service-oriented Approach. In *31st Annual International Computer Software and Applications Conference. COMPSAC 2007*, pages 473. 478, July 2007.
- [8] P. Madsen and E. Maler. SAML executive overview. <http://xml.coverpages.org/SAML-ExecOverviewV206-11785-20050310.pdf>, March 2005.
- [9] MySQLAB. MySQL. <http://www.mysql.com/>, 2008.
- [10] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks, volume 32, pages 33.38. September 1994.
- [11] OASIS Web Services Security (WSS) TC. Web services security. <http://www.oasis-open.org/specs/index.php#wssv1.1>, 2004.
- [12] OpenID Foundation. OpenID. <http://openid.net/what/>.
- [13] OpenLDAP Foundation. RFC4510 Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map.
- [14] Regents of the University of Michigan. CoSign. <http://www.umich.edu/~umweb/software/cosign/>, 2008.
- [15] S. Anderson et al. Web Services Secure Conversation Language (WS-SecureConversation). <http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf>, 2005.

- [16] B. Sotmayor and L. Childers. Globus Toolkit 4: Programming Java Services. Morgan Kaufmann Publishers, San Francisco, CA, 2006.
- [17] B. St.Arnaud. CA*net4 research program update – UCLP roadmap: Web Services work_ow for connecting research instruments and sensors to networks. <http://www.canarie.ca>, December 2004.
- [18] The Apache Software Foundation. Apache Tomcat. <http://tomcat.apache.org/>.
- [19] The Apache Software Foundation. Web Services - Axis. <http://ws.apache.org/axis/>.
- [20] The Globus Alliance. The globus toolkit. <http://www.globus.org/toolkit>.
- [21] The National Science Foundation Cyberinfrastructure Council. Cyberinfrastructure vision for 21st century discovery. <http://www.nsf.gov/pubs/2007/nsf0728/nsf0728.pdf>, March 2007.
- [22] The UCLP Development Team. User Controlled Lightpaths. <http://www.uclp.ca>, 2006.
- [23] University of Washington. How pubcookie works. <http://www.pubcookie.org/docs/how-pubcookie-works.html>, 2003.
- [24] J.Weise. Public key infrastructure overview. Sun BluePrints OnLine, August 2001.
- [25] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlis. X.509 Proxy Certificates for Dynamic Delegation. In 3rd Annual PKI R&D Workshop, 2004. <http://www.globus.org/alliance/publications/papers/pki04-welch-proxy-cert-final.pdf>.

Aaron Moss is a Computer Science student majoring in Theory and Computation in the University of New Brunswick. He is a Dean's List student, and has competed at the regional level in the ACM-ICPC programming competition. Aaron has strong interest in security systems and has researched various credential systems. He implemented the credential manager for the SAVOIR project during his work-term at the National Research Council. He is now working for the Canadian Security Establishment for another work-term.

Sandy Liu is a Research Officer at the Institute for Information Technology, National Research Council Canada. Sandy completed her Master of Science in Computer Science from Acadia University, and Bachelor of Business Administration in Management Information System from the University of Macau. She is also a PhD candidate in Computer Science at the University of New Brunswick. Sandy's research interests include service computing, workflow and business process management, collaborative systems, and intelligent software agents. She has over ten years of research and industrial experience. Sandy has published in various workshops, conferences and journals.

Rene Richard is an Applications Specialist and Research Support at the Institute for Information Technology, National Research Council Canada. René completed a Bachelor of Social Science in Economics from the Université de Moncton and a certificate in Internet Object Oriented Software Engineering at the Southern Alberta Institute of Technology. René has more than 15 years of industrial experience. He has a strong interest in software engineering and has designed and developed many successful software solutions for industrial and research projects.