Federated Identity Management for Android

Anders Fongen
Norwegian Defence Research Establishment
Norway
anders.fongen@ffi.no

***Abstract*—A federated identity management system (IdM) must include mobile units and must provide mutual authentication for client-server connections. Existing frameworks for identity management like SAML are unlikely to apply well to resource constrained mobile terminals like Android. The contribution of this paper is an IdM with simpler data representation and protocols for identity management and authentication, which can be deployed with fewer code lines, consume less bandwidth and require less connectivity than traditional protocols, e.g., those based on SAML and WSSec. The related service invocation mechanisms is designed to support mobile services, where object methods in mobile units can be invoked from other nodes in the network, regardless the use of NAT units and firewalls.**

Keywords-Identity management, Android, Authentication

I. INTRODUCTION

The XML protocol was once proposed as a simple replacement for SGML, which had grown very complicated and required large software stacks for processing. XML on the other hand, was simple, human readable and could be processed even with simple string operations (in case a parser was unavailable). Current XML-based standards are often seen to have lost this virtue, and are heavily dependent upon a dedicated software stack for processing, as well as they are hard to read by a human eye and even harder to verify.

Complicated protocols with many optional properties offer less interoperability than simple protocols. In the case of Identity Management (IdM) and authentication based on SAML [1] and WSSec [2] standards there is little interoperability to see, implementations only talk to themselves. This observation is based on unpublished experiments conducted by the author in order to make a Java stack and a .NET stack cooperate over a SOAP header containing WSSec and SAML data. The large number of variables (key length, key algorithm, key presentation, addressing formats etc.) were not sufficiently coordinated, the implementations were immature and software bugs added to the problem.

No one-size-fits-all solution to identity management exists, but yet it does not make sense to duplicate "stovepipe" systems in order to accommodate the different operating environments. What makes sense, however, is to differentiate in the *presentation* of the data structures involved, including the credentials used for authentication. Different presentation layers retrieve their information from the same storage of keys, roles and attributes, and put their trust upon the same identification and revocation procedures (since these procedures are costly to operate and should not be duplicated). Different presentations could be used to improve interoperability of an identity management system as well as to offer services to disadvantaged equipment.

For the purpose of offering identity management services to mobile units based on the Android platform an "extension" to an existing IdM has been built, using a presentation layer better suited for those units.

The IdM builds on existing Public Key Infrastructures (PKI) technology and storage services for user roles and attributes, and offers identity management services with related services for authenticated and encrypted service invocation. The services employ simple protocols and a thin presentation layer. The participant of the IdM (clients and service providers) exchange serialized Java objects and must therefore run on a platform with support for the Java serializing API (like Java VM, Android Dalvik, Scala etc.)

The presented IdM is built on the principles and prototype presented in [3], [4]. The contribution of this paper is to show how disadvantaged nodes and networks may be included in existing IdM systems through the provision of an adapted presentation layer. An investigation of Message Oriented Middleware like XMPP for IdM-related communication will also be presented.

The remainder of this paper is organized as follows: The next section will give a general background on Identity Management. Section III will present an outline of the GISMO IdM system. Section IV will discuss specific mechanisms related to operation across *Communitites of Interest* (COI), and Section V will present the GISMO IdMs framework for service invocation based on PDU (Protocol Data Units) with serialized Java objects. The use of a messaging protocol for improved reliability and connectivity will be discussed in Section VI, and interoperability issues related to the dual stack situation in the GISMO IdM will be discussed in Section VII. The paper provides a summary and some conclusive remarks in Section VIII.

II. MOTIVATIONAL BACKGROUND

Identity Management (IdM) are collection of services and procedures for maintaining subject information (key pair,

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9

roles) and to issue credentials for the purpose of authentication, message protection and access control. From the client perspective, the credentials issued by the IdM services enables it to access many services inside a community under the protection of mutual authentication and encryption. From the server perspective, IdM enables it to offer credentials to clients in order to provide mutual authentication.

A. Federated Identity Management

Several federated IdM schemes have been developed, some of which offer single sign on (SSO) for web clients [5], [6], [7]. The SSO protocols exploits the redirection mechanism of HTTP in combination with cookies and POST-data so that an Identity Provider (IdP) can authenticate the client once and then repeatedly issue credentials for services within the federation. This arrangement requires IdP invocation for each "login" operation, and does not offer mutual authentication, i.e., service authentication.

In the situation where the client is an application program (rather than a web browser), there are more opportunities for the client to take actively part in the protocol operations, e.g., by checking service credentials, contacting the IdP for the retrieval of own credentials, caching those credentials etc. The research efforts presented in this paper assume that the clients enjoy the freedom of custom programming.

The usual meaning of the word "federated" is that several servers share their trust in a common IdP for subject management and authentication. It does not necessarily imply any trust relationship between independent IdPs so that they can authenticate each others' clients. For the following discussion, we will call the group of clients and services which put their trust in the same IdP as a *community of interest*. A trust relation between independent IdPs is called a *cross-COI relation*.

B. Mobile and Federated IdM requirements

An essential property of an IdM is its ability to integrate with other components for management of personnel and equipment.

- An IdM should be able to use resources from the existing PKI (keys, certificates, revocation info) and offer its services to different platforms, with different presentation syntax and for different use cases.
- An IdM should also be able to tie trust relations with other IdMs in order to provide accommodation for guests and roaming clients.
- An IdM should support protocol operations for mutual authentication.

For IdM used in mobile systems, there are requirements related to the resource constraints found in these systems:

- A IdM for mobile operation must use the minimum number of protocol operation, small PDU sizes and must allow the use of caches.

C. The relation between IdM and Access Control

Services can enforce access control on the basis of the *identity* of an authenticated client, or based on *roles* or *attributes* associated with the client. For the purpose of the accommodation of roaming users, it is absolutely necessary to make access control decisions based on roles/attributes, not identity. Identity based access control requires that all roaming clients are registered into the guest IdM, which is an unscalable solution.

The principles of *Role/Attribute Based Access Control* (RBAC/ABAC) are well investigated [8]. The names and meaning of the roles/attributes that are used to make access decisions must be coordinated as a part of an IdM trust relationship. For that reasons, the number of roles/attributes used for access control needs to be kept low.

It is the obvious responsibility of an IdM to manage the roles/attributes of a subject, some of which may enter into access control decisions, others be used by the service to adapt the user interface etc. The presence of subject attributes is the main functional difference between IdM credentials and X.509 public key certificates.

III. The GISMO IdM architecture

For the purpose of authenticated service provisioning in military tactical networks (meaning wireless, mobile, multihop, multicarrier networks), an Identity Management system has been developed under the project name "GISMO" (General Information Security for Mobile Operation). The system has been previously presented in [3], [4], so its properties are only briefly listed here:

- It uses short lived *Identity Statements* containing the subject's public key and subject attributes. No revocation scheme is necessary. Identity Statements are issued by an Identity Provider (IdP).
- Cross COI relations are represented by ordinary identity statement issued from one IdP to another.
- IdPs can issue *Guest Identity Statements* when presented with a Identity Statement issued by an IdP with with whis it has a Cross COI relation. A guest identity statement contains the same information, but is signed by the different IdP.
- Authentication takes place either through a signature in the service request, or through the encryption of the service response.
- Supports Role/Attribute Based Access Control (RBAC/ABAC) through the subject attributes.
- Employs, but encapsulates an existing PKI. Clients never see X.509 certificates or revocation info.
- Identity Statements are cached and re-used during its lifetime. An IdP is invoked to issue Identity Statements, not to verify authenticity.
- Loose coupling between IdP and services/clients, and between COIs. Very little redundant registration is necessary.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9

Subject Distinguished Name
Subject Public Key
Subject Attributes
Valid from–to
Issuer Distinguished Name
Issuer Public Key
Issuer's Signature

Figure 2. The structure of the Identity Statement

The main contribution of this manuscript is an IdM for mobile systems and the related discussion on how a common IdM can accommodate different presentation layers.

Figure 1 illustrates the concepts and components of the GISMO IdM. Identity establishment, key generation and key certification happens in the (existing) PKI. Related to a CA (Certificate Authority) domain there are several Communities of Interest (COI) with one IdP common to all members of that community.

The IdP issues signed *Identity Statements*. The structure of the Identity Statement is shown in Figure 2.

Members of a COI only trust the signature of their IdP, so an Identity Statement (signed by the IdP) is not valid outside the COI unless there exists a *cross-COI Identity Statement* which links the signature of the foreign IdP to the trusted IdP. More on that later.

A. Presentation layer issues

The GISMO IdM was first developed over existing SOAP standards like SAML, WSSec, WS-addressing etc. There are libraries for Java that supports the processing of the structures, although somewhat incomplete and buggy. There are also .NET components available, but we were unsuccessful in building the IdM services on .NET. Interoperability were therefore apparently limited to Java code based on the same class library (Sun XWSS 2.0).

The second version of the GISMO IdM was built with a different presentation layer. The choice was to use *serialized Java objects*. Java serialization is a mature and well proven technology, which is available for all Java platforms except J2ME, and is also supported by the Android Dalvik virtual machine.

Even though Java is a proprietary platform as opposed to SAML/WSSec, the interoperability property of the IdM actually has improved, since it now accommodates not only J2SE platforms, but also Android Dalvik VM and programming languages that use the same serialization engine, e.g., Scala. Besides, the serialization engine consumes less resources than the XWSS library and the serialization API is straightforward and well understood.

Consequently, the second phase of the GISMO IdM uses native Java objects (POJO) for representing identity statements and service invocation PDUs. These objects are serialized during network transport. The transport protocol of choice has been HTTP and XMPP, although any reliable transport protocol (or messaging middleware) will do.

To summarize: the reasons for the use of native Java objects for PDU presentation rather than XML based standards are interoperability, network efficiency and ease of programming. The last property is of importance since this is a prototype system for experimental study.

TABLE I
ABBREVIATIONS USED IN THE FIGURES

Client X_a	Client X of COI a
IdP_a	Identity provider of COI a
PKI_a	Validation services in domain a
Server F_b	Server F in COI b
$(Id_x)_a$	Identity statement for identity x, issued by IdP_a
$(msg)S_x$	Message msg signed with private key of x
$(msg)E_x$	Message msg encrypted with public key of x

IV. CROSS COI RELATIONSHIPS

Any client will likely be a member of several COIs, reflecting the diverse tasks and responsibilities of a worker or a soldier. It is not convenient to manage the client's key pairs, attributes etc. in every COI. Most of them will naturally belong to one COI, e.g., their national military unit or the employing department, and could be regarded as "guests" in other COIs.

The ability to authenticate across COI borders is believed to be an essential requirement for a modern IdM. In the GISMO IdM, this problem has been solved by the use of *Guest Identity Statements*. One IdP can issue a Guest Identity Statement if presented for an Identity Statement issued by an IdP with which it has a trust relationship. The trust relationship is represented by a pair of *cross-COI Identity Statements* issued from one IdP to the other.

During invocation of a service in the foreign COI, the client presents the Guest Identity Statement as a part of the authentication process. The service trust the Guest Identity Statement since it is issued by "its" IdP. In order for the client to authenticate the service response, it needs the cross-COI identity statement issued by *its own IdP* to the foreign IdP so that a signature path back to its own IdP can be made. The service signs the response, includes its identity statement (signed by the foreign IdP), which together with the aforementioned cross-COI identity statement forms a signature path back to the trust anchor of the client.

Figure 3 shows the interaction between the client and the IdPs during the issuance of identity statements. Please observe that the cross-COI identity statements are issued asynchronously with regard to the client operations, but handed back to the client during issuance of a guest identity statement. Abbreviations used in the figure are explained in Table I.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9

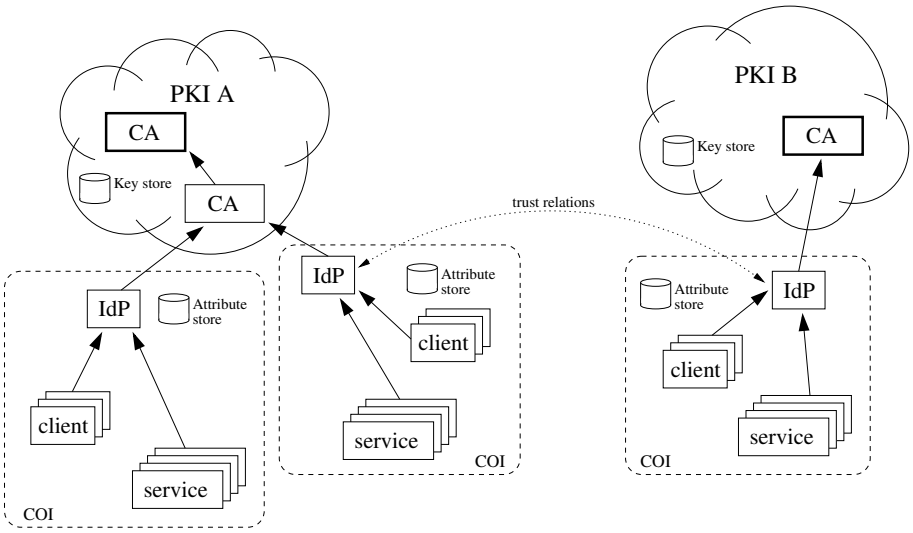

Figure 1. The functional components of a federated IdM. Observe that the IdP serves one single COI, and the trust relations are formed between COIs, not domains. Key management is handled by the PKI whereas the attribute management is done by the IdPs on the COI level

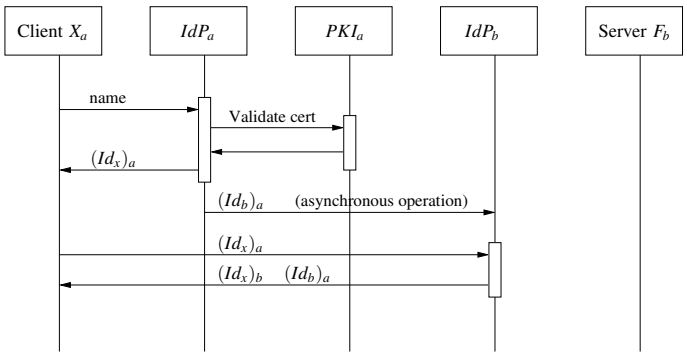

Figure 3. The identity statement issuing protocol. The IdP of COI A, termed IdP_a, issues a "native" identity statement to the client, which is given to IdP_b, which in turn issues a guest identity statement. The term PKI_a denotes a set of certificate validation services in COI a.

V. Service invocation

For service invocation using serialized POJOs as PDUs a number of interesting opportunities knock: The client may simply send a parameter object to the server containing the parameter values, and the *class* of the object identifies the service method. This arrangement eliminates the need for a separate scheme for service addressing and also eliminates the need for separate stub/skeleton compilation.

In the server, a single Java servlet hosts all services. This is possible since we do not address the service in a URL, but through association with the parameter class. The URL addresses a servlet "dispatcher" service, and the serialized parameter object included in the POST operation controls the dispatching process. The services are loaded dynamically from a JAR file repository at servlet startup and deployed through class introspection, no configuration file editing is necessary. Consequently, the deployment of services requires less configuration than a Java servlet.

One could argue that Remote Method Invocation (RMI) could have been chosen rather than a home made invocation scheme. The answer is that RMI is a full size distributed object system, whilst what is needed here is invocation of remote *procedures*. RMI is not very firewall friendly, requires distributed garbage collection and separate stub compilation and is over-specified for this particular purpose.

A. Authentication dependent on server state space

The authentication mechanisms assure the identity of the client and service during service invocation. Many different authentication protocols can be incorporated into GISMO IdM as long as they employ a public key pair corresponding to the information in the Identity Statement. It is also a requirement that the authentication can be piggybacked on the service request and should not generate separate PDUs. Two protocols have been implemented in GISMO IdM:

1) In those cases where the request must be authenticated *before* the service execution a replay protection must be in place. Replay protection requires the server to remember past requests (by their Nonce) for a while, so a clock synchronization scheme and a non-volatile stable storage must be in place (since past requests must be be remembered also across server incarnations). These requirements are rather costly.
2) In the case of a *stateless* service, where the execution of a service request does not alter the state of the service, replay protection is not necessary. A request should be signed by the client in order to protect the integrity of the message, but no Nonce for request replay protection is included. The response is *en-*

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9

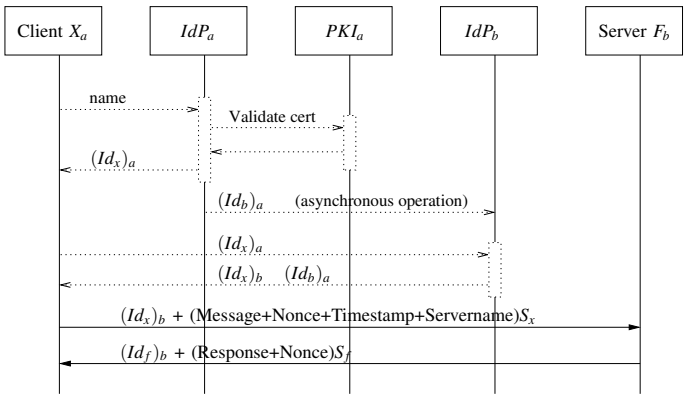

Figure 4. The authentication protocol for the stateful service. Both the request and response are signed with the sender's private key as a part of authentication process. A timestamp, a nonce and the server's name is included for replay protection.

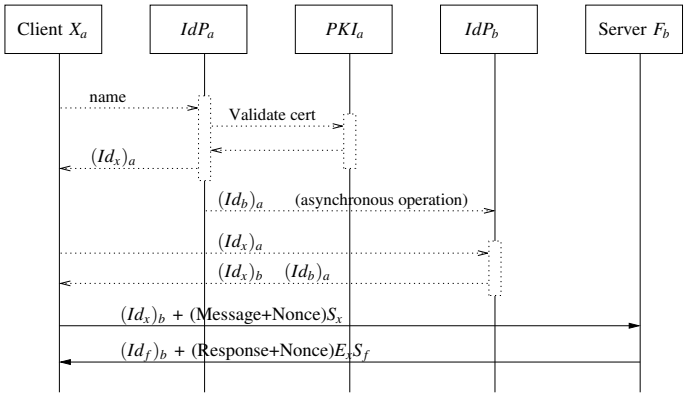

Figure 5. The authentication protocol for the stateless service. Requests are not reply protected since this is not considered as a threat, but the response need to be protected for reasons of response replay and information compromise. For the sake of integrity protection, the request is signed. The encryption of the response is a part of the authentication scheme, not a privacy measure.

crypted with the client's public key, making it useless for everyone but the holder of the private key. To a stateless server, replayed requests are not a threat and protection is not needed. Requests still need a Nonce for reasons of response replay protection, but that does not increase the state space in the client.

Figures 4 and 5 shows the two variants as an interaction diagram. The interactions shown with dotted lines are related to IdP operations and discussed in more detail in Figure 3.

B. Authentication during Identity Statement Issuance

Authentication also takes place during Identity Statement issue operations. The client simply signs the request with its private key. If the requested Identity Statement contains the corresponding public key the client is regarded as authenticated.

The Identity Statement is generally a public document and the need for authenticated requests does not always seem apparent. It is, however, likely that some subject attributes are sensitive since they reveal information about the subject's authorizations. For that reason, only authenticated requests are given the full attribute set in the Identity Statement, others receive a subset of the attributes. The selection takes place over a simple attribute name prefix convention.

VI. Messaging protocols

In a wired private network where capacity and reliability suffice, and there exist IP routes between the nodes that wish to communicate, the HTTP protocol works just fine for IdP operations and service invocations. For mobile networks this is not necessarily the case: they are slow, unreliable and consists of several partitions connected with application level gateways (from reasons of security and traffic control).

In the context of the experimental study of the GISMO IdM, an XMPP (eXtensible Messaging and Presence Protocol) network was already in place for chat communication. Through the XMPP routers (working as application gateways) otherwise isolated networks (where no IP route exists between them) can exchange chat messages. The XMPP system provides reliable and "persistent" communication in the sense that messages are stored in XMPP routers if they are undeliverable for the moment.

A. Service provision by mobile units

A messaging system creates reachable endpoints for nodes, which are disconnected at the IP layer. Nodes which reside behind a NAT unit or a firewall are unreachable from the outside world at the IP layer, yet a messaging system can send them messages. Through the XMPP protocol a mobile node can receive service requests as any other service provider. The prototype system uses a very simple service container (not a servlet), which is easily portable to a mobile Android based unit.

B. Synchronization and message persistence

The use of an "persistent" communication layer underlying an RPC system poses interesting problems related to recovery and resynchronization. In those cases where a client does not receive a timely response it simply aborts the operation and sends a similar request later. The server may have processed the first request and the response may simply be delayed. The client may now receive the delayed response as the apparent response to the repeated operation, which will be discarded. A "forward synchronization" scheme solved that problem. Under some circumstances, the XMPP nodes can be instructed not to store messages if they are of a "headline" type.

VII. SOAP vs. POJO interoperability

The GISMO IdM contains nodes which use different presentations for Identity Statements and service invocations. In

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9

order for two nodes to communicate, they must use the same communication stack, including the presentation layer. A client using serialized POJOs can therefore not communicate with an IdP or a service requiring SOAP message syntax and vice versa. For a client to reach the services it needs regardless its choice of presentation syntax three approaches can be taken:

1) Make services (and the IdP) dual-stack.
2) Make a general proxy for automatic conversion between the presentation forms (POJO and SOAP), e.g., based on JAXB.
3) Make a specific proxy for each service

Option 1 is a possible solution, but do carry a rather high cost in terms of software footprint and deployment configuration. Since SOAP services cannot employ the parameter class association scheme explained in Section V, the automatic deployment mechanism must be replaced with a manual configuration procedure.

Option 2 has not been studied in detail, but requires a combination of WSDL-compilation and JAXB-assisted conversion. It is not likely to be possible to convert on-the-fly any POJO to a SOAP message which conform to the WSDL-file of a particular web service.

Option 3 has been studied and tested, and represents an attractive approach. A service which takes the parameter values and passes them to a precompiled web services stub (generated by the WSDL compiler). The return value from the stub is passed back to the caller of the POJO service. Example code lines required for this function are shown below:

```
public class MainClass {
  public Serializable service(WeatherRequest wr,
                              Properties props) {
    try {
      Weather w = new Weather();
      String result = w.getWeatherSoap()
         .getWeather(wr.town);
      return result;
    } catch (Exception e) { return e; }
  }
}
```

Option 3 is also attractive since it gives the developer control over service aggregation and orchestration. One service call to a POJO service need not be passed on as one single web service invocation. Many individual calls may be made, and they may be sequenced or tested in any manner. Aggregated operations are useful because they potentially reduce the network traffic to and from the mobile unit, which is likely to be connected through a disadvantaged link. The proxy can even cache results for subsequent service calls.

For options 2 and 3 there is a problem related to signature values. Equivalent POJO and SOAP messages will have different signature values, and the integrity of the message is broken during a conversion. The proxy can sign the converted object using its own private key, which would require that the service accepts that the proxy vouches for the original client in the authentication phase.

VIII. Conclusions

A number of problems related to identity management of mobile units have been presented and discussed in this paper. Rather than the deployment of a separate IdM with a presentation layer and protocols adapted to a mobile environment, the addition of a separate presentation layer to an existing IdM has been proposed. The architecture of this IdM, including technical details of the use of serialized Java objects (POJO) has been described. Also, a simpler authentication protocol with fewer round trips and smaller PDUs have been proposed.

The use of serialized POJOs in the service invocation opens up interesting opportunities for easier construction and deployment of services. These aspects have been studied and described in the paper.

Future research in this field is planned to be targeted on concept demonstration in military exercises. The frameworks and suggested programming patterns will be tested in a medium scale mobile networks where military technology from several NATO countries will be tested with cooperation and interoperability in mind. During these experiment the protocols' ability to sustain service in disadvantaged networks with low bandwidth and episodic connectivity will be tested under realistic conditions.

References

[1] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, and T. Scavo, *Security Assertion Markup Language (SAML) V2.0 Technical Overview*, OASIS Committee Draft, March 2008.

[2] K. Lawrence and C. Kaler, *Web Services Security: SOAP Message Security 1.1*, OASIS Standard Specification, 2004.

[3] A. Fongen, "Identity management without revocation," in *SECURWARE 2010*. Mestre, Italy: IARIA, July 2010.

[4] ——, "Architecture patterns for a ubiquitous identity management system," in *ICONS 2011*. Saint Maartens: IARIA, Jan. 2011.

[5] "Shibboleth." [Online]. Available: http://shibboleth.internet2.edu/ [retrieved November 9, 2010]

[6] "OpenID." [Online]. Available: http://openid.net/ [retrieved November 9, 2010]

[7] "The Libery Alliance." [Online]. Available: http://www.projectliberty.org/ [retrieved November 9, 2010]

[8] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*. New York, NY, USA: ACM, 2000, pp. 47–63.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-010-9