

# Profiles for Conveying the Secure Communication Requirements of Web Services

Duane Merrill  
Andrew Grimshaw

University of Virginia, Department of Computer Science

## Abstract

The lack of a single authority in the Grid environment is perhaps the biggest source of security and interoperability challenges faced by Grid systems designers. A strong commitment to meaningful, interoperable security is crucial for fostering Grid adoption and buy-in. The issues of security-interoperability are twofold: (a) grids require federation of distinct trust and security domains, and (b) grid participants need to convey and discover their secure communication requirements. This paper presents two new OGF security profiles that address this latter issue.

The *Secure Communication Profile 1.0* is a refinement of the WS-SecurityPolicy specification. The goals of this profile are to impose more restrictive conformance requirements on the WS-Security mechanisms described by WS-SecurityPolicy assertions, to facilitate key distribution, and to profile normative “well-known” policy documents that identify commonly-used security mechanisms.

The *Secure Addressing Profile 1.0* refines the WS-Addressing specification in order to profile the inclusion of security policy within Endpoint References (EPRs). This approach of conveying security policy within EPRs is well-suited to the Grid paradigms of stateful Web service resources and factory patterns.

## 1 Introduction

Grid computing is fundamentally concerned with the sharing of networked resources across administrative boundaries. Today, grids provide the framework for a variety of distributed uses: adaptive computing, utility computing, high-performance computing, decentralized data management, etc. The commonalities between these genres of distributed computing revolve around the coordinated virtualization, integration, and management of computational services and data in a wide-area, heterogeneous environment having no single source of authority.

This lack of a single source of authority gives rise to many of the unique challenges that must be addressed when designing a sustainable Grid architecture. Different participants from different organizational domains contribute and consume one another’s resources, composing them in dynamic ways in order to form diverse “grid applications”. This loosely-coupled notion of resource sharing invites many opportunities for misuse, particularly when the network interconnect is the public Internet<sup>1</sup>.

The opportunity for misuse or unauthorized behavior implies that meaningful security is paramount for Grid adoption and buy-in. The security of grid infrastructure is its ability to protect its assets (information, data, services, etc) from various sources of misuse (i.e., *threat*<sup>2</sup>). Without a strong

---

<sup>1</sup> The “Internet Threat Model” [1] assumes that the end systems (i.e., hosting environments, operating systems, container software, etc.) are secure, yet the interconnection network is not.

<sup>2</sup> We use the terms *threat*, *attack*, *threat assessment*, and *security policy* as defined in the OSI security framework [2].

commitment to meaningful security, many potential adopters would be unable to participate because of undue risk and/or legal restrictions. Most types of security threat can be generalized to fall within the following three categories: (a) disclosure or theft of resources, (b) modification (including destruction) of resources, and (c) resource service interruption.

The goal of Grid security architecture is to define the security services and related mechanisms that can be appropriately applied when protection of communication between participants is required. Such services and mechanisms should make the cost of unauthorized behavior greater than the potential value of doing so. (Or make the time required to perform unauthorized behavior so great that its inherent value is lost.)

With the Grid community's adoption of the Web services paradigm, Grid software architects have at their disposal an extensive, mature set of industry-standard specifications with which for they can use to secure Web services under a wide variety of security models, including X.509 PKI [3], SAML [4], Kerberos [5], TLS/SSL [6], etc. The decision to use a particular model may depend on any number of factors, most notably:

- Suitability in terms of cost, usability, performance, and effectiveness for implementing the desired security policies.
- Accommodation of existing security infrastructure. Resources and participants are anticipated to have existing security policies for authorized behavior that are tied to extant identity or rule-based credential infrastructures.

While interoperability is fostered by the standardization of service interfaces (such as directory services [7], execution services [8], data services [9], etc.), the reality is that Grid participants will "come to the table" with different underlying security models and mechanisms. This presents distinct challenges for interoperability, even if all of the security mechanisms in question are well-specified by de-facto community standards. We can break this security-interoperability issue into two related, but separate components:

- a) *Federation of disparate trust and security domains.* As mentioned above, Grid resources are anticipated to have existing security policies for authorized behavior that are tied to existing credential infrastructures. These infrastructures are unlikely to be shared by other administrative domains within the Grid, even if the credential mechanisms are common. In an end-to-end interaction scenario within the Grid environment, the identities or roles of the participants as well as their respective credentials may not carry any syntactic or semantic meaning to the other participants, yet cannot be expected to be replaced or supplanted for the purposes of the Grid architecture. Hence the overall Grid security architecture must facilitate the integration of existing trust and security domains. [10-12]
- b) *Discovery and conveyance of secure communication requirements.* Because there is no single Grid authority, each resource provider is free to establish its own security policies and models. The mechanisms required to implement these policy semantics ultimately have an effect on message format. For example, a service endpoint may require incoming messages to contain specific types of security credentials or be signed or encrypted in a certain fashion. We consider these security impositions upon message format as being orthogonal to the application-specific message formats defined by standard service interfaces. All service implementations of a particular application (e.g., a basic execution service) must conform to the same interface, yet each is free to choose the security mechanisms that satisfy its particular security policy requirements.

We have created the *Secure Communication Profile 1.0* [13] and *Secure Addressing Profile 1.0* [14] documents in order to address this latter aspect of security-interoperability. The question is, then,

“How does a resource consumer discover the secure communication requirements for a resource that it would like to interact with?”

The *Secure Communication Profile 1.0* profile (SecComm) addresses this issue from the policy angle: how to best describe secure communication requirements. The SecComm profile has three primary goals:

- To profile the *WS-Security Policy 1.2* [15] language to accommodate the inclusion of actual security tokens within policy documents.
- To provide a point of further refinement for commonly-used security mechanisms profiled within the *WS-I Basic Security Profile 1.0* [16].
- To define normative, referenceable, composable policy documents identifying commonly-used security mechanisms.

The *Secure Addressing Profile 1.0* profile (SecAddr) addresses the discovery aspect of secure communication: how to convey such security requirements for a given Web service resource to the resource consumer. The SecAddr profile normatively refines the *WS-Addressing 1.0 – Core* [17] specification in order to facilitate the inclusion of such *WS-SecurityPolicy* assertions within WS-Addressing endpoint references, and how to do so in such a way that trust guarantees can be made to the consumer regarding the authenticity of the encapsulated policy information.

The rest of this paper is organized as follows. Section 2 discusses the overarching philosophies that provide the direction and guidance for our profile documents. We review the properties of secure communication and how it is addressed by the Web services paradigm in Section 3. Sections 4 and 5 provide overviews of the *Secure Communication Profile 1.0* and *Secure Addressing Profile 1.0* profile documents, respectively. After having discussed the assumptions and considerations made by our profiles, Section 6 presents six application use-cases that specifically leverage the mechanisms described. We’ve selected these use-cases because they are representative of generalized, recurring patterns of interaction that are benefited by our profile documents. Section 7 discusses our experiences with constructing a compliant implementation, and Section 8 concludes.

## 2 Governing Philosophies

The two profile documents presented in this paper were developed within the Open Grid Forum (OGF) community, specifically the Open Grid Services Architecture Working Group (OGSA-WG). In this context, “open” refers to the process used to develop standards that achieve interoperability. The “architecture” defines the components; their interfaces, organizations, and interactions; and the design philosophies used. The OGSA-WG establishes the following general philosophies for specifications and profiles produced under its stewardship:

- *Allow multiple policies and mechanisms to co-exist.* This is fundamental to realizing the vision of resource-sharing while maintaining local site-autonomy in an environment lacking a single source of authority.
- *Allow composition of mechanism.* As mentioned earlier, Grid resources are often shared without specific application goals in mind. Users obtain value from the Grid by constructing applications from loosely-coupled services. Without clear application requirements in mind, communication mechanisms must be profiled, but not mandated. (This is the end-to-end argument: avoid attempting to require a “least common denominator” that may not be desirable for all scenarios.)
- *Be extensible.* Standards and profile documents should be able to accommodate new mechanisms as they are developed (i.e., be made “future-proof”).

- *Be implementation agnostic.* OGSA implementations cannot presume that other Grid endpoints will be operating any particular implementation.
- *Use standard mechanisms.* Interoperability requires that security data-structures, protocols, and service interfaces adhere to published, industry-accepted specifications.

### 3 Secure Communication with Web Services

The task of realizing acceptable security for one's assets can be boiled down to the task of threat assessment. Threat assessment is the process of identifying the specific security threats against which protection is required. Participants are responsible for conducting their own threat assessments. In addition to establishing authorization requirements (i.e., who is allowed proper access), a threat assessment for a prospective resource to be exposed via the Grid may identify *confidentiality* and *integrity* requirements for the protection of communication over an untrusted network:

- Confidentiality is the assurance that only those principals intended for information access are allowed access to that information. Confidentiality properties in an insecure networked environment are usually derived through message encryption. Message confidentiality is often a co-product of authenticating the message receiver: only the cryptographically identified receiver can extract the message plaintext.
- Integrity is the assurance that unauthorized changes made to communication messages can be detected by the recipient. Cryptographic digital signatures are generally used to enable the detection of message tampering. Message integrity is often a co-product of authenticating the message sender: the digital signature also presents proof of identity.

The Web Services paradigm uses the SOAP [18] protocol for messaging between communication endpoints. Fundamentally, the security of SOAP messages is affected by two aspects:

- a) *Binding to a particular network transport protocol.* The SOAP protocol is extremely flexible. It can be enacted over virtually any other communication protocol, such as HTTP, SMTP, JMS message queues, etc. These substrate protocols may themselves provide security guarantees for properties such as authentication, integrity, and confidentiality. (For example HTTP can leverage SSL/TLS to provide transport-layer security protection.) When relying on these properties to mitigate threat, care must be taken to assure that the end-to-end notions of the substrate protocol match those of the SOAP message exchange.
- b) *Message-level credentialing and protection.* In contrast to most underlying transport protocols, the SOAP protocol is sufficiently general in that it can support virtually any credentialing system. In particular, the Web Services Security [19] (WS-Security) family of specifications defines a general-purpose mechanism for associating security credentials with message content. WS-Security then goes on to construct a set of specific profiles for encoding popular token types (e.g., X.509 [20], Kerberos [21], SAML [22], and Username-token [23] credentials). The WS-Security Core specification also defines the application of XML-Encryption [24] and XML Digital Signature [25] to provide end-to-end messaging integrity and confidentiality without the support of the underlying communication protocol.

In order to achieve real-world interoperability, the WS-I Basic Security Profile (WS-I BSP) [16] provides guidance on the use of WS-Security and its associated security token formats to resolve nuances and ambiguities between communicating implementations intending to leverage common security mechanisms.

## 4 Secure Communication: A Profile on WS-SecurityPolicy

This section presents the *Secure Communication Profile 1.0* document (SecComm). The SecComm profile is fundamentally a refinement of the *WS-SecurityPolicy 1.2* specification. We refine the WS-SecurityPolicy specification in two ways: (1) refinement of the language of policy documents that it describes, and (2) refinement of the semantics implied by these policy documents.

### 4.1 Language Refinement of WS-SecurityPolicy for Key Distribution

Like most XML specifications, WS-SecurityPolicy can be thought of as a grammar that defines a language of XML documents. In this case, the elements of the WS-SecurityPolicy language are *policy documents* that can be used to describe the secure communication requirements for Web service resources.

A policy document is composed of one or more *policy assertions*. A policy assertion is simply a specific requirement, capability, or property that impacts communication behavior. For example, a `<sp:SignedParts>` element found within a policy document is an assertion whose child elements indicate which portions of a communication message are to be digitally signed.

Policy documents can contain sets of policy assertions that are mutually exclusive of each other. These mutually exclusive groupings of assertions are called *policy alternatives*. In this fashion, one can create a policy document that conveys multiple avenues for communication, e.g., “You can either send messages to me over HTTPS with a UsernameToken credential, or over HTTP in which you sign and encrypt the message with our X.509 PKI.”

The SecComm language of policy documents is a subset of the XML documents allowed by WS-SecurityPolicy. We do not invent any new XML child elements or attributes, but rather impose some restrictions on optional elements and attributes (notably optional “any”-type elements). More specifically, we add some extra rules into the language in order to allow security policy documents to facilitate the process of key distribution.

Key distribution plays an important role in facilitating large, secure distributed systems that experience dynamic membership. Virtually every information security model that supports integrity and confidentiality uses cryptographic keys of some form in order to protect communication. In a public key infrastructure (PKI) model, an authority associates a public/private keypair with a specific entity or identity. Parties are able to securely communicate with each other after learning the other’s public key. In a symmetric key model, parties are able to securely communicate after negotiating a shared, secret key. For practical reasons of efficiency, we often combine the two models and use PKI cryptography to securely negotiate a shared secret key that is subsequently used to protect the bulk of communication between entities. The task of key distribution is one of supplying the recipient’s public key (often in the form of a digital certificate of identity) to the caller prior to communication.

The security policy document for a remote resource is an attractive vehicle for distributing its public key. A client must know both the remote resource’s message processing requirements as well as its cryptographic keys. Although WS-SecurityPolicy is good for conveying the *type* of security tokens needed as well as *how* they are to be used, it unfortunately doesn’t have any explicit provisions for embedding actual security tokens (e.g., X.509 certificates that contain public keys) within policy documents.

WS-SecurityPolicy defines a class of assertions called *token assertions* that can be used to specify different types of security tokens. These token assertions can then be included in larger constructs called *binding assertions* that carry semantic meaning for how those tokens are to be used for message exchange. For example, an “asymmetric-key” binding assertion containing two X.509

Token Assertions in <i>WS-SecurityPolicy 1.0</i>	Token Assertions in <i>Secure Communication 1.0</i>
<pre> &lt;xs:complexType name="TokenAssertionType"&gt;   &lt;xs:sequence&gt;     &lt;xs:choice minOccurs="0"&gt;       &lt;xs:element name="Issuer"         type="wsa:EndpointReferenceType"/&gt;       &lt;xs:element name="IssuerName"         type="xs:anyURI" /&gt;     &lt;/xs:choice&gt;      &lt;xs:any minOccurs="0"       maxOccurs="unbounded"       namespace="##other"       processContents="lax"/&gt;   &lt;/xs:sequence&gt;   ... &lt;/xs:complexType&gt; </pre>	<pre> &lt;xs:complexType name="TokenAssertionType"&gt;   &lt;xs:sequence&gt;     &lt;xs:choice minOccurs="0"&gt;       &lt;xs:element name="Issuer"         type="wsa:EndpointReferenceType"/&gt;       &lt;xs:element name="IssuerName"         type="xs:anyURI" /&gt;       &lt;xs:element         ref="wsse:SecurityTokenReference"/&gt;     &lt;/xs:choice&gt;      &lt;xs:any minOccurs="0"       maxOccurs="unbounded"       namespace="##other"       processContents="lax"/&gt;   &lt;/xs:sequence&gt;   ... &lt;/xs:complexType&gt; </pre>

**Table 1: Refinement of the WS-SecurityPolicy’s generic token assertion type by the Secure Communication profile.**

“initiator” and “recipient” token assertions can be used to describe the message-level communication requirements between two parties who want to sign and encrypt messages using X.509 public key infrastructure. WS-SecurityPolicy allows us to optionally annotate the recipient token assertion with either an `<sp:Issuer>` or an `<sp:IssuerName>` element to indicate a location from which to obtain the required X.509 digital certificate. The SecComm profile refines the `xsd:any` element within the token assertion schema to allow for a third option: the direct embedding of a `<wsse:SecurityTokenReference>` element within the policy document. As shown in Table 1, we mandate that the usage of the `<wsse:SecurityTokenReference>` element as a child of a token assertion be mutually exclusive of either `<sp:Issuer>` or `<sp:IssuerName>`, and recommend that the token reference be of the *embedded* or *direct* type.

#### 4.2 Semantic Refinement of WS-SecurityPolicy for Interoperability

The WS-SecurityPolicy language of security assertions was created to describe the security mechanisms and semantics defined in the WS-Security family of specifications. While the WS-Security specifications support a broad set of requirements and offer a variety of options and approaches, they can lead to interoperability challenges that result from complexity and misinterpretation. The SecComm profile constrains these options and simplifies communication by incorporating the conformance requirements of the WS-I Basic Security Profile 1.0 (WS-I BSP). Thus the secure communication mechanisms described by SecComm-compliant policy documents must adhere to the transport (HTTP over TLS) and SOAP message security clarifications and constraints of the WS-I BSP, which are designed to greatly improve the interoperability characteristics of these two technologies.

In addition to refining the semantics of the security mechanisms describable by WS-SecurityPolicy, we also found it also necessary to resolve some ambiguities between the WS-SecurityPolicy “algorithm suites” and the SSL/TLS ciphersuites. These algorithm suites are composed of cryptographic algorithms that have been defined in the XML Encryption and XML Signature specifications, but are intended by WS-SecurityPolicy to also apply to transport security mechanisms (HTTP over TLS). Our SecComm profile establishes a canonical mapping of WS-SecurityPolicy to

Transport-level Mechanism	Policy Reference URI
<ul style="list-style-type: none"> <li>Server-authenticated TLS</li> </ul>	<a href="http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLS">http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLS</a>
<ul style="list-style-type: none"> <li>Server-authenticated TLS</li> <li>Server certificate must be provided in enclosing policy document</li> </ul>	<a href="http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLSCertProvided">http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLSCertProvided</a>
<ul style="list-style-type: none"> <li>Mutually-authenticated TLS</li> </ul>	<a href="http://www.ggf.org/ogsa/2007/05/sp-secure-transport#MutualTLS">http://www.ggf.org/ogsa/2007/05/sp-secure-transport#MutualTLS</a>
<ul style="list-style-type: none"> <li>Mutually-authenticated TLS</li> <li>Server certificate must be provided in enclosing policy document</li> </ul>	<a href="http://www.ggf.org/ogsa/2007/05/sp-secure-transport#MutualTLSCertProvided">http://www.ggf.org/ogsa/2007/05/sp-secure-transport#MutualTLSCertProvided</a>

**Table 2: Commonly-used, well-known transport-level policies defined by *Secure Communication Profile 1.0***

Message-level Mechanism	Policy Reference URI
<ul style="list-style-type: none"> <li>UsernameToken</li> </ul>	<a href="http://www.ogf.org/ogsa/2007/05/sp-secure-soap#UsernameToken">http://www.ogf.org/ogsa/2007/05/sp-secure-soap#UsernameToken</a>
<ul style="list-style-type: none"> <li>Password-digest UsernameToken</li> </ul>	<a href="http://www.ogf.org/ogsa/2007/05/sp-secure-soap#PasswordDigest">http://www.ogf.org/ogsa/2007/05/sp-secure-soap#PasswordDigest</a>
<ul style="list-style-type: none"> <li>X.509 authentication of message senders to message receivers</li> <li>Includes signature over message bodies and WS-Addressing headers</li> </ul>	<a href="http://www.ogf.org/ogsa/2007/05/sp-secure-soap#MutualX509">http://www.ogf.org/ogsa/2007/05/sp-secure-soap#MutualX509</a>

**Table 3: Commonly-used, well-known message-level policies defined by *Secure Communication Profile 1.0***

TLS/SSL algorithm suites. (E.g, the *Basic256* algorithm suite maps to the *TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA* algorithm suite if using TLS 1.0/1.1 and to *SSL\_RSA\_WITH\_AES\_256\_CBC\_SHA* if using SSL 3.0).

### 4.3 Commonly-used, Well-known Policy Documents

In order to further promote interoperability, the SecComm profile defines a collection of security policy documents that represent commonly-used security mechanisms. These “well-known” policy documents are given referenceable names, allowing applications to operate more efficiently by conveying security requirements with a well-known URI rather than by communicating entire policy documents (which can be quite lengthy). The well-known policies defined by SecComm are enumerated in Tables 2 and 3.

The nestable nature of WS-SecurityPolicy facilitates the composition of multiple referenceable policies, allowing a resource to compactly define multiple alternatives for its security policy. For example, Figure 1 shows a resource’s security policy that specifies *ServerTLS* along with *UsernameToken* as one alternative, and *MutualX509* (with additional message-protection policy elements to specify encryption of the message body) as a different alternative.

```

(01) <wsp:Policy>
(02)   <wsp:ExactlyOne>
(03)
(04)     <!-- Alternative 1: Server-authenticated TLS + Username-token -->
(05)     <wsp:All>
(06)       <wsp:PolicyReference>
(07)         http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLS
(08)       </wsp:PolicyReference>
(09)       <wsp:PolicyReference>
(10)         http://www.ggf.org/ogsa/2007/05/sp-secure-soap#UsernameToken
(11)       </wsp:PolicyReference>
(12)     </wsp:All>
(13)
(14)     <!-- Alternative 2: X.509 message-level authentication, encryption -->
(15)     <wsp:All>
(16)       <wsp:PolicyReference>
(17)         http://www.ggf.org/ogsa/2007/05/sp-secure-soap#MutualX509
(18)       </wsp:PolicyReference>
(19)       <wsse:SecurityTokenReference>
(20)         <wsse:Embedded>
(21)           <wsse:BinarySecurityToken wsu:Id='RecipientMessageIdentity'
(22)             ValueType="http://...-wss-x509-token-profile-1.0#X509v3"
(23)             EncodingType="http://...-message-security-1.0#Base64Binary">
(24)               GJW5xM3aHnLxOpGVIpzSg4V486hHFe7sHET/uAEdsm/...
(25)           </wsse:BinarySecurityToken>
(26)         </wsse:Embedded>
(27)       </wsse:SecurityTokenReference>
(28)
(29)     <!-- Additional policy indicating message-level encryption -->
(30)     <wsp:Policy>
(31)       <sp:EncryptedParts>
(32)         <sp:Body/>
(33)       </sp:EncryptedParts>
(34)     </wsp:Policy>
(35)   </wsp:All>
(36)
(37) </wsp:ExactlyOne>
(38) </wsp:Policy>

```

**Figure 1: A security policy that is constructed to define two communication alternatives. The first is a composition of SecComm’s well-known *ServerTLS* and *UsernameToken* policies. The second alternative is the *MutualX509* policy (with an additional message-protection policy element to specify encryption of the message body).**

Further inspection of Table 2 reveals that we have created a set of transport-level policies in which the TLS/SSL “server certificate” must be placed within any security policy document that makes reference to these mechanism definitions. By including the server certificate with a `wsu:ID` of “RecipientTransportIdentity” in a trusted security policy document, a client can be afforded an extra measure of confidence by comparing this certificate to the one received during SSL/TLS handshake. This is particularly useful when hostname-verification is not possible in the event that the server-certificate does not contain a proper hostname or IP address. (Such certificates are used in Grid scenarios in which administrators may wish to migrate service endpoints.)

The *MutualX509* policy also deserves some further discussion. It does not provide mutual authentication in the same sense as TLS/SSL. Rather, it simply requires that message-senders authenticate to message-receivers. The implication is that, in a request/response pattern, the Grid resource is only authenticated to the client upon receipt of the response message. In order to provide true mutual authentication at the message level, additional message-protection assertions for message encryption must be specified (such as those shown in Figure 1). Such message encryption assertions should be used when authentication of the resource to the client is desired, particularly for one-way

exchanges. In order for clients to verify the response as having come from the expected source or to perform encryption of the request message, the `MutualX509` policy requires the inclusion of the resource's certificate within a greater policy document, this time with a `wsu:ID` of "RecipientMessageIdentity".

The `MutualX509` policy also contains considerations for Web service communication to Grid resources that rely on WS-Addressing headers. WS-Addressing provides, among other features, the ability for a Web service endpoint to expose multiple stateful resources (or sessions) of the same type. (For example, a Byte-IO endpoint may expose multiple stateful file-like resources.) In order to achieve this "multiplexing", WS-Addressing uses *reference parameter* message headers that are used by Web service endpoints to identify the specific resources for which operations are intended. Because the `MutualX509` policy is, at a minimum, intended to provide integrity properties, it is necessary for this policy to mandate that signatures of the message document include signing over any WS-Addressing headers.

## 5 Secure Addressing: A Profile on WS-Addressing

This section presents the *Secure Addressing Profile 1.0* document (SecAddr). The SecAddr profile is fundamentally a refinement of the *WS-Addressing 1.0 – Core* specification. The SecAddr profile establishes a mechanism for securely conveying communication policies to resource-consumers within WS-Addressing endpoint references (EPRs).

### 5.1 WS-Addressing Overview

WS-Addressing allows a single Web service endpoint to expose multiple logical resources. These resources may be object-like (e.g., file or directory resources) or session-like (e.g., data streams); the common feature of such logical resources is that they are *stateful*.

These logical resources are referenced and addressed by WS-Addressing endpoint reference (EPR) data-structures. EPRs serve as resource handles, providing an "invocation context" that describes information that can be used by a client to establish meaningful communication with their corresponding resources. EPRs are pervasively used within the OGSA Grid architecture (e.g., directory services, notification services, etc.) and within Web Services in general. They are well-suited to the factory pattern in which a Web services endpoint creates many stateful resources. For example, a Basic Execution Service (BES) creates activity resources when running jobs, Byte-IO and RNS services can "mount" files and directories from the local filesystem by exposing them as individual stateful resources, etc.

### 5.2 The Case for Conveying Security Policy within EPRs

There are three general approaches for conveying secure communication requirements (viz. SecComm policy documents) to potential resource-consumers:

1. *Convey policy within the service interface (WSDL)*. The *Web Services Policy 1.5 - Attachment* [26] (WS-PolicyAttachment) specification defines mechanisms for associating policies with the subjects to which they apply. Specifically, it defines how to decorate WSDL [27] descriptions with WS-Policy [28] documents (of which WS-SecurityPolicy is a language subset).

This approach has two fundamental shortcomings. The first is granularity. The published WSDL service interface is representative of the service endpoint. However, through WS-Addressing reference parameters, a Web Service endpoint may multiplex multiple stateful

resources, sessions, processors, etc. This scheme for attaching policy documents to WSDL is not capable of expressing different requirements for such individual resources.

The second shortcoming is availability. WSDL retrieval requires additional communication overhead and is not guaranteed to be published online. Additionally, there is no standard, profiled way of using a Web service address URL to locate the corresponding WSDL. In fact, with the standardization of interfaces, WSDL is losing relevance for dynamic discovery. Distributed Web services applications are generally built to leverage specific types of services; simply knowing the WSDL namespace URI is generally enough to infer the service interface.

2. *Convey policy through reflective metadata-retrieval operations on the resource itself.* Many distributions of grid software have followed this paradigm. For example, the caGrid [29] uses Globus GT4 [30] middleware that has been customized in order to ensure that all grid resources expose a `getServiceSecurityMetadata()` operation.

This approach has three fundamental issues. The first is that it mandates all Web service endpoints expose a common security-policy-retrieval operation. Although it would be trivial to define such a service interface specification, one developed by an open standards community does not yet exist.

The second issue is a “chicken-before-the-egg” problem: one must establish communication with a resource to discover how to communicate with it. Generally this is solved by setting “wide open” access control for the metadata retrieval operation, which may violate security policy (e.g., leaking information about whether or not a resource even exists).

The third issue is communication overhead. One must incur the cost of an additional message exchange prior to sending the intended message. Obviously this burden can be reduced by caching retrieved policy information, but care must be taken to mitigate the potential for stale policy information.

3. *Convey policy within WS-Addressing Endpoint References (EPRs).* Unfortunately, neither the WS-PolicyAttachment nor the WS-Addressing specifications normatively define how to embed secure communication policy within EPRs. The SecAddr profile remedies this deficiency by describing how WS-SecurityPolicy documents can be embedded within the extensible metadata portion of an endpoint reference. This allows the EPR to truly contain all of the information necessary to establish secure, meaningful communication with a remote resource.

This approach has one fundamental drawback: the secure communication requirements are static. They cannot be tailored to the individual requestor to indicate the particular information that authorization credentials must convey (claims) or where they may be obtained. For example, EPRs retrieved from a directory service will convey the same security information for their corresponding resources to all clients. The static nature of EPRs also makes them susceptible to the potential “staleness” issue that can arise if a resource’s communication policy is changed.

The practice of embedding secure communication requirements within EPRs is not new. The Liberty ID-WSF architecture [31] uses “security context” elements to convey semicolon-delimited URIs that indicate authentication and secure communication mechanisms. The Liberty approach lacks the fidelity to express fine-grained message protection requirements and the ability to convey key material.

```

(01) <wsa:EndpointReference>
(02)   <wsa:Address wsu:Id='TheAddress'>
(03)     http://www.example.org/some/path
(04)   </wsa:Address>
(05)
(06)   <wsa:ReferenceParameters wsu:Id='TheRefParams'>
(07)     ...
(08)   </wsa:ReferenceParameters>
(09)
(10)   <wsa:Metadata wsu:Id='TheMetadata'>
(11)
(12)     <!-- This policy attachment applies to all actions on this endpoint -->
(13)     <wsp:PolicyAttachment>
(14)       <wsp:AppliesTo>
(15)         <wsp:URI>urn:wsaaction:*</wsp:URI>
(16)       </wsp:AppliesTo>
(17)
(18)       <!-- Collection of policy alternatives -->
(19)       <wsp:Policy>
(20)         <wsp:ExactlyOne>
(21)
(22)           <!-- Alternative 1: Server-authenticated TLS + Username-token -->
(23)           <wsp:All>
(24)             <wsp:PolicyReference>
(25)               http://www.ggf.org/ogsa/2007/05/sp-secure-transport#ServerTLS
(26)             </wsp:PolicyReference>
(27)             <wsp:PolicyReference>
(28)               http://www.ggf.org/ogsa/2007/05/sp-secure-soap#UsernameToken
(29)             </wsp:PolicyReference>
(30)           </wsp:All>
(31)
(32)         </wsp:ExactlyOne>
(33)       </wsp:Policy>
(34)     </wsp:PolicyAttachment>
(35)
(36)     ...
(37)   </wsa:Metadata>
(38)   ...
(39) </wsa:EndpointReference>

```

**Figure 2: An example EPR containing security policy that is a composition of SecComm’s well-known *ServerTLS* and *UsernameToken* policies.**

### 5.3 Attaching WS-SecurityPolicy within Endpoint References

Similar to the SecComm profile described in the previous section, SecAddr creates a sub-language of WS-Addressing endpoint reference documents by imposing some restrictions on optional “any”-type attributes and elements. More specifically, SecAddr profiles the use of a `<wsp:PolicyAttachment>` element as a child of the EPR’s extensible `<wsa:Metadata>` element, as illustrated by the example EPR shown in Figure 2.

The WS-PolicyAttachment specification introduces the notion of *policy scope*: the collection of *subject* entities (e.g., endpoints, operations, messages, actions, etc.) to which a given policy document applies. In order to allow different security policies to be associated with different operations on a Web services resource, the SecAddr profile defines the domain of policy subjects for EPR-attached policy documents as the set of WS-Addressing actions available upon the referenced endpoint. For example, this can allow a directory resource to specify that no security tokens are necessary to list the contents of a directory, yet specific holder-of-key SAML credentials are needed to perform update operations. (In the event that a single secure communication policy is applicable

for all operations upon a resource, the policy attachment should specify “urn:wsaaction:\*” as the policy subject.)

#### **5.4 Digital Signature of EPRs**

In many scenarios it will be necessary to validate the integrity and trustworthiness of an EPR before using the information it contains for communication. For example, consider the use-case in which a resource-consumer obtains an EPR for a particular resource from a third party such as a directory service. The consumer would like to have some assurance that the EPR was minted by a trusted source and has not been tampered with since. (Otherwise the consumer may worry that they may actually be interacting with an imposter.) The SecAddr profile addresses this concern by profiling the use of an XML Signature `<ds:Signature>` element as a child of the EPR’s extensible `<wsa:EndpointReference>` element. Such a digital signature must cover the EPR’s `<wsa:Address>`, `<wsa:ReferenceParameters>`, and `<wsa:Metadata>` elements.

## **6 Example Application Scenarios**

In this section, we present six application scenarios that specifically leverage the mechanisms described by the SecComm and SecAddr profiles. We’ve selected these use-cases because they are representative of generalized, recurring patterns of interaction that are benefited by our profile documents.

### **6.1 Event Notification**

The notification of a relying party upon the occurrence of some event is a common activity for many distributed applications. A resource replica may notify a WS-Naming [32] resolver of its termination so that references to it will no longer be doled out to consumers in the primary resource fails. A basic execution service may notify a meta-scheduler that its available physical resources have changed. An identity-provider service may notify resources that had been given delegation privileges that a user has logged out.

At the most basic level, event notification follows a simple, *asynchronous one-way communication pattern*. Delivery may be best-effort, or depend on a reliable messaging transport (e.g., JMS, WebSphere MQ, MSMQ, etc.) to guarantee delivery.

In addition to the notification interface, the notifying party must also understand the destination’s secure communication requirements. What security tokens or credentials are required to authenticate the client’s notification message? If needed, what message protection actions, if any, must be taken to assure integrity and confidentiality? Although WS-SecurityPolicy alone can help address these issues, SecComm-compliance affords the communicating parties a greater likelihood of interoperability due to the mechanism restrictions specified in the WS-I BSP.

The second benefit to being SecComm-compliant stems from the one-way communication pattern. In order to authenticate the destination to the client and/or provide end-to-end confidentiality, the client must know the public key of the destination resource in advance in order to perform encryption at the message-level. (True one-way-ness precludes a handshake protocol that might otherwise be used to obtain key information directly from the recipient.) The SecComm profile allows for policy to easily convey this key information at the same time in which the client learns the other communication requirements.

## 6.2 Browsing a Grid Namespace

In this example, consider a browser client that is being used to traverse an RNS directory namespace. The Resource Namespace Service (RNS) specification defines a service interface that can be used to organize Grid resources in a hierarchical fashion, much like a file system or directory service. The entries for a given RNS directory are <String, EPR> tuples called *junctions*. The client retrieves the junction listings for a directory by performing a lookup operation on that directory resource.

This activity exhibits a simple *request-response communication pattern* that is complicated by the fact that the client may discover and chose to interact with resources it has never seen before. How is the browser supposed to learn the individual secure communication requirements of a given resource so that it may interact with it? A convenient solution to this discovery issue is the use of the SecAddr profile to embed WS-SecurityPolicy documents within the EPRs maintained by the RNS directory resources.

## 6.3 Firewall Traversal

Many Grid deployments services are behind corporate firewalls or network-address-translation devices (NATs). Clients cannot directly interact with the service or the container in which the service resides. The general solution to this problem is to construct proxy services that straddle these network boundaries and relay messages back and forth. This is the communication-with-intermediaries pattern.

This scenario is similar to the “Event Notification” use-case: the presence of message-passing intermediaries precludes a handshake protocol that might otherwise be used to convey the recipient’s public key information to the message sender. The SecComm profile allows for policy to easily convey this key information at the same time in which the sender learns the other communication requirements.

## 6.4 “Exported Directories”

In many data-sharing scenarios, the resources that Grid participants would like to expose are subsections of their local filesystems. This can be efficiently accommodated by operating a single Web services container that hosts both an RNS and a Byte-IO [33] service endpoint. Furthermore, the Web service implementations for these two endpoints can leverage the WS-Addressing reference properties to virtualize many local directories and files as RNS and Byte-IO resources, respectively.

In this is the *hosted-resources pattern*: incoming transport-level communication is handled at the granularity of the Web services container. (E.g., the container has a single X.509 identity that is used to authenticate all incoming service requests, regardless of which service endpoint they are ultimately destined for.) In effect, communications to stateful resources are multiplexed through a single Web service endpoint, and communications to Web service endpoints are multiplexed through a single Web services container.

In this scenario, security requirements may vary per resource, possibly requiring that each stateful resource be given its own cryptographic identity and corresponding security policy. This granularity of Grid resource leads to message-level security requirements that need to be conveyed per-resource, making the SecAddr approach of advertising policy via resource EPRs an attractive one. The SecComm profile provides value in that its “well-known” policies profile the secure treatment of WS-Addressing headers. Additionally, the SecAddr profiling of how to digitally sign endpoint references provides value in that clients do not have to rely on trusted directory systems for maintaining safe, consistent EPRs.

## 6.5 Simple Cross-domain Computing

Consider the scenario in which a computational job needs to use data currently being hosted by providers from different security domains. The client may need to obtain several credentials, all of which would be delegated to the BES service. During stage-in, the BES service would then use the appropriate credentials when requesting data for each input provider. This is the *multiple identities/attributes pattern*. Like the “Event Notification” use-case above, cross-domain interoperability is better fostered by conformance with the SecComm profile because of its restrictions on token formats and cryptographic algorithms.

## 7 Implementation Experiences

We have implemented the *Secure Communication Profile 1.0* and *Secure Addressing Profile 1.0* within our Genesis II grid platform. Genesis II is open source, standards-based grid middleware designed to support both high-throughput computing and secure data sharing. The Genesis II distribution implements many of the OGF/OGSA standard Grid service interfaces, including Basic Execution Service (BES), Byte-IO, and Resource Namespace Service (RNS). It leverages the WS-Addressing and WS-Naming specifications to expose multiple stateful resources through static Web service endpoints hosted within a web application server, and allows each individual stateful resource to have its own cryptographic identity (specifically X.509 certificates). This granularity of Grid resource makes it a good candidate for evaluating SecAddr as a mechanism for distributing resource-specific security-policies and keys.

Genesis II provides support for a handful of popular security models, notably:

- 1) Transport-level (TLS/SSL) mutual authentication of the client and the Web services container using X.509 identities
- 2) Transport-level (TLS/SSL) authentication of the Web services container’s X.509 identity to the client, message-level UsernameToken authentication of the client to the Grid resource
- 3) Message-level authentication of message-senders to message-receivers using X.509 identities and XML Signature. (Transport-level SSL/TLS is used for confidentiality.)
- 4) Message-level authentication of message-senders to message-receivers using X.509 identities and XML Signature, supplemented by one or more client-provided, holder-of-key SAML credentials indicating a signed chain of delegated rights. (Transport-level SSL/TLS is used for confidentiality.)
- 5) Message-level mutual authentication of the client and Grid resource using X.509 identities in conjunction with XML Signature and XML Encryption.
- 6) Message-level mutual authentication of the client and Grid resource using X.509 identities in conjunction with XML Signature and XML Encryption, supplemented by one or more client-provided, holder-of-key SAML credentials indicating a signed chain of delegated rights.

Figure 3 illustrates a SecComm-compliant security policy supporting all six of these mechanism compositions.

Even though we use policy references to well-known SecComm mechanisms, the size of the policy document that must be embedded into an EPR is not insignificant. The Genesis II EPR sizes for resources that have no security policy versus those that support all six mechanisms are 2.6 KB and 8.6 KB, respectively. The primary contributor to this drastic size increase is the inclusion of the

```

(01) <!-- Collection of policy alternatives -->
(02) <wsp:Policy>
(03)   <wsp:ExactlyOne>
(04)
(05)     <!-- Alternative 1:
(06)       Mutually-authenticated TLS -->
(07)     <wsp>All>
(08)       <wsp:PolicyReference>
(09)         http://www.ggf...#MutualTLS
(10)       </wsp:PolicyReference>
(11)     </wsp>All>
(12)
(13)     <!-- Alternatives 2, 3, and 4:
(14)       (Server-TLS) && ((UsernameToken) ||
(15)       (Mutually-Signed X509) ||
(16)       (Mutually-Signed X509 +
(17)       SAML delegation)) -->
(18)     <wsp>All>
(19)       <wsp:PolicyReference>
(20)         http://www.ggf...#ServerTLS
(21)       </wsp:PolicyReference>
(22)
(23)     <wsp:Policy>
(24)       <wsp:ExactlyOne>
(25)
(26)         <wsp>All>
(27)           <wsp:PolicyReference>
(28)             http://www.ggf...#UsernameToken
(29)           </wsp:PolicyReference>
(30)         </wsp>All>
(31)
(32)         <wsp>All>
(33)           <wsp:PolicyReference>
(34)             http://www.ggf...#MutualX509
(35)           </wsp:PolicyReference>
(36)
(37)           <sp:SignedSupportingTokens
(38)             wsp:Optional="true">
(39)             <wsp:Policy>
(40)               <sp:SamToken sp:IncludeToken=
(41)                 ".../AlwaysToRecipient">
(42)                 <wst:Claims Dialect=
(43)                   ".../GenesisII/v1.0"/>
(44)               <wsp:Policy>
(45)                 <sp:WssSamlV20Token1.1>
(46)               </wsp:Policy>
(47)             </sp:SamToken>
(48)           </wsp:Policy>
(49)         </sp:SignedSupportingTokens>
(50)       </wsp>All>
(51)
(52)     </wsp:ExactlyOne>
(53)   </wsp:Policy>
(54) </wsp>All>
(55)
(56) <!-- Alternatives 5 and 6:
(57)   (Message encryption) &&
(58)   ((Mutually-Signed X509) ||
(59)   (Mutually-Signed X509 +
(60)   SAML delegation)) -->
(61) <wsp>All>
(62)   <wsp:Policy>
(63)     <sp:EncryptedParts>
(64)       <sp:Body/>
(65)     </sp:EncryptedParts>
(66)   </wsp:Policy>
(67)
(68)   <wsp:PolicyReference>
(69)     http://www.ggf...#MutualX509
(70)   </wsp:PolicyReference>
(71)
(72)   <sp:SignedSupportingTokens
(73)     wsp:Optional="true">
(74)     <wsp:Policy>
(75)       <sp:SamToken sp:IncludeToken=
(76)         ".../AlwaysToRecipient">
(77)       <wst:Claims Dialect=
(78)         ".../GenesisII/v1.0"/>
(79)     <wsp:Policy>
(80)       <sp:WssSamlV20Token1.1>
(81)     </wsp:Policy>
(82)   </sp:SamToken>
(83) </wsp:Policy>
(84) </sp:SignedSupportingTokens>
(85) </wsp>All>
(86)
(87) <!-- Resource's digital certificate -->
(88) <wsse:SecurityTokenReference>
(89)   <wsse:Embedded>
(90)     <wsse:BinarySecurityToken
(91)       wsu:Id=
(92)         "RecipientMessageIdentity"
(93)       ValueType=
(94)         "http://...#X509v3"
(95)       EncodingType=
(96)         "http://...#Base64Binary">
(97)       AJpGVIpzSg4V486ET/uAEdsm/...
(98)     </wsse:BinarySecurityToken>
(99)   </wsse:Embedded>
(100) </wsse:SecurityTokenReference>
(101) </wsp:ExactlyOne>
(102) </wsp:Policy>
(103)

```

**Figure 3: An example security policy for a Genesis II resource illustrating all four possible avenues of authentication: mutually-authenticated TLS, UsernameToken, message-level X.509, and by message-level delegated message-level SAML credential.**

resource’s entire certificate chain, which presently consists of four X.509 digital certificates that cumulatively consume 4.1 KB. This “RecipientMessageIdentity” could be reduced to about 1.2 KB if it only included the resource’s end-entity certificate. This optimization would require more sophisticated trust-store configuration on behalf of Grid clients in order for them to be able to reconstruct a verified and trusted certificate chain, but would reduce the overall EPR size to 5.7 KB.

The other primary contributor to the large sizes of Genesis II EPRs is the inclusion of “implemented interface” URIs within the EPR’s extensible <wsa:Metadata> section. On average, a Genesis II resource implements eight to ten distinct service interfaces, which cumulatively account for about 1.8 KB.

To help mitigate the susceptibility to failure due to stale EPRs contained within RNS directories, Genesis II implements WS-Naming, an extension to WS-Addressing that defines the use of “resolver” services that can be used to lookup a fresh EPR for a resource when supplied with the endpoint identifier (EPI) for that resource. This mechanism works by embedding a resolver EPR within a resource’s EPR, effectively doubling the size of the resource EPR.

Our experience with the two profiles indicates that they both provide valuable, flexible solutions for conveying security requirements. We additionally note that the use of the SecAddr approach of conveying security policy within EPRs may not be appropriate for application scenarios where the transfer or storage of multi-kilobyte EPRs is not feasible, as the inclusion of key material within the EPR drastically increases its size.

## 8 Conclusions

The lack of a single authority in the Grid environment is perhaps the biggest source of security and interoperability challenges faced by Grid systems designers. In this paper, we presented two new OGF profiles that are designed to facilitate interoperable, secure communication between Grid participants: the *Secure Communication Profile 1.0* (SecComm) and *Secure Addressing Profile 1.0* (SecAddr) profile documents.

Normative profile documents are useful tools for explicitly refining behavior for implementations that adhere to industry-standard Web service specifications. Interoperability problems arise when ambiguous behavior is defined within a given specification or results from a composition of specifications (e.g., WS-SecurityPolicy, WS-Security, WS-I BSP, WS-Addressing, etc.), leading to different interpretations by different service implementers.

Although both documents concern themselves with the problem of conveying the secure communication requirements of Grid resources to clients, they address orthogonal issues and can thus be implemented independently. The SecComm profile refines the WS-SecurityPolicy specification to facilitate key distribution and impose more restrictive conformance requirements on the describable WS-Security mechanisms. These refinements as well as the normative, “well-known” policy documents profiled by SecComm can be used wherever SecurityPolicy might be conveyed (e.g., within WSDL, UDDI, reflective metadata operations, within EPRs, etc.). The SecAddr profile fills the gap that is caused by a lack of consideration by the WS-Addressing and WS-PolicyAttachment specifications for embedding security policy within EPRs. This approach is deserving of profiling within the standards community because it is well-suited to the Grid paradigm of stateful Web service resources and factory patterns.

By themselves, these profile documents are not sufficient to guarantee interoperability of all compliant Web service clients and resources. Many other security interoperability factors are out of scope of these profiles, such as the federation and integration of credentialing infrastructures. Additionally, these profiles do not establish a “lowest-common-denominator” set of security mechanisms that must be supported by all compliant resources. Rather, they adopt the view that specific secure communication requirements may vary between communities of resource providers and consumers. The intent is for a community to self-select such requirements that are appropriate and then leverage these profiles to achieve interoperability between its members (and/or cleanly discover where interoperability is not possible).

## 9 References

1. Rescorla, E. and B. Korver, *Guidelines for Writing RFC Text on Security Considerations*. 2003: RFC Editor.
2. ITU, *Security Architecture for Open Systems Interconnection for CCITT Applications*, in *CCITT Recommendation X.800*. 1991, International Telecommunication Union: Geneva.
3. ITU-T, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*, in *ITU-T Recommendation X.509*. 08/2005, International Telecommunication Union.
4. OASIS-SAML. *Assertions and Protocol for the OASIS Security Assertion Markup Language*. OASIS Standard 2003 [cited September 2 2003]; V1.1:[
5. Kohl, J. and C. Neuman, *The Kerberos Network Authentication Service (V5)*. 1993: RFC Editor.
6. Dierks, T. and C. Allen, *The TLS Protocol Version 1.0*. 1999: RFC Editor.
7. *Resource Namespace Service Specification*. July 2005, Global Grid Forum, GWD-R.
8. Grimshaw, A., et al., *OGSA Basic Execution Service*. 2007.
9. Antonioletti, M., et al., *Web Services Data Access and Integration - The Core (WS-DAI) Specification, Version 1.0* 2006, Open Grid Forum.
10. Della-Libera, G. *Web Services Trust Language (WS-Trust)*. 2002 18 December 2002 [cited; Available from: <http://www-106.ibm.com/developerworks/library/ws-trust/>].
11. David Del, V., et al., *CredEx: User-Centric Credential Management for Grid and Web Services*, in *Proceedings of the IEEE International Conference on Web Services*. 2005, IEEE Computer Society.
12. Audun, J., et al., *Trust requirements in identity management*, in *Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44*. 2005, Australian Computer Society, Inc.: Newcastle, New South Wales, Australia.
13. Merrill, D., *Secure Communication Profile 1.0*. 2007, Open Grid Forum: OGSA-WG.
14. Merrill, D., *Secure Addressing Profile 1.0*. 2007, Open Grid Forum: OGSA-WG.
15. Nadalin, A., et al. *WS-SecurityPolicy 1.2*. 2007 July 1, 2007 [cited; Available from: <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html>].
16. McIntosh, M., et al. *Basic Security Profile Version 1.1*. 2007 2007-02-20 [cited; Available from: <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.1.html>].
17. Gudgin, M., M. Hadley, and T. Rogers. *Web Services Addressing 1.0 - Core*. 2006 May 9, 2006 [cited; Available from: <http://www.w3.org/TR/ws-addr-core/>].
18. Gudgin, M., et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. 2007 April 27, 2007 [cited; Available from: <http://www.w3.org/TR/soap12-part1/>].
19. Nadalin, A., et al. *Web Services Security: SOAP Message Security 1.1*. 2006 Feb 1, 2006 [cited; Available from: <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>].
20. Nadalin, A., et al. *Web Services Security X.509 Certificate Token Profile 1.1*. 2006 Feb 1, 2006 [cited; Available from: <http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf>].
21. Nadalin, A., et al. *Web Services Security Kerberos Token Profile 1.1*. 2006 Feb 1, 2006 [cited; Available from: <http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf>].
22. Nadalin, A., et al. *Web Services Security SAML Token Profile 1.1*. 2006 Feb 1, 2006 [cited; Available from: <http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTOKENProfile.pdf>].
23. Nadalin, A., et al. *Web Services Security Username Token Profile 1.1*. 2006 Feb 1, 2006 [cited; Available from: <http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-os-UsernameTokenProfile.pdf>].
24. Imamura, T., B. Dillaway, and E. Simon. *XML Encryption Syntax and Processing*. 2002 Dec 10, 2002 [cited; Available from: <http://www.w3.org/TR/xmlenc-core/>].
25. Bartel, M., et al. *XML-Signature Syntax and Processing*. 2002 Feb 12, 2002 [cited; Available from: <http://www.w3.org/TR/xmlsig-core/>].
26. Bajaj, S., et al. *Web Services Policy 1.2 - Attachment (WS-PolicyAttachment)*. 2006 April 25, 2006 [cited; Available from: <http://www.w3.org/Submission/WS-PolicyAttachment/>].

27. Christensen, E., et al. *Web Services Description Language (WSDL) 1.1*. 2001 [cited; Available from: <http://www.w3.org/TR/wsdl>].
28. Bajaj, S., et al. *Web Services Policy 1.2 - Framework (WS-Policy)*. 2006 April 25, 2006 [cited; Available from: <http://www.w3.org/Submission/WS-Policy/>].
29. Joel, S., et al., *caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid*. 2006, Oxford University Press, p. 1910-1916.
30. Globus. [cited; Available from: <http://www.globus.org>].
31. *Liberty Alliance Project*. [cited; Available from: <http://www.projectliberty.org/>].
32. Grimshaw, A., D. Snelling, and M. Morgan, *WS-Naming Specification*. 2007, Open Grid Forum, GFD-109.
33. Morgan, M., *ByteIO Specification*. 2006, Global Grid Forum. GFD-86.