OMII Grid Security Technology Overview[1]

Syd Chapman, Alistair Dunlop, Peter Henderson and Steven Newhouse
Open Middleware Infrastructure Institute
University of Southampton
SO17 1BJ, UK

Friday, 29 July 2005

Please send comments and corrections to p.henderson@omii.ac.uk

Overview

One function of the Grid is to support collaborative science projects on the internet. While some projects may be able to operate with little or no security provided by the infrastructure, major projects have significant security requirements. Shared data or data in transit may be highly sensitive and will almost certainly need guarantees that it hasn't been tampered with or seen by unauthorised persons. The security may be provided by the application itself, but more usually it will be provided by the infrastructure.

This paper provides an introduction to some of the security issues that arise in collaborative distributed computing and some of the technology solutions that have been developed to meet these requirements.

We motivate our examination of these web service and internet technologies through a few simple usage scenarios that have emerged within Grid Computing. In this initial technology overview, prepared within OMII as part of the ongoing debate we are having in respect of interoperability of grid infrastructures, we largely restrict ourselves to authentication and authorisation.

Background

Security on the internet is provided by the judicious use of cryptography. This technology can be applied at the transport layer or at the message level. Currently, using Transport Layer Security (TLS) such as https is much easier but considerably weaker than using Message Level Security (MLS) such as WS-SecureConversation.

Using cryptographic technology, confidentiality can be established by encrypting the message contents. Encryption is done using a key (a very big number). Decryption is also done using a key. Where these encryption and decryption keys are the same, this is called symmetric-key cryptography. Its weakness is that the keys must be kept secret by both parties to the communication. Its strength is that it is fast.

The stronger, but slower, alternative is public-key cryptography. Here the keys are different, but are chosen with the important mathematical property that if you encrypt with one key you can decrypt with the other. A user then keeps one key secret (the

[1] Available at http://dedalus.ecs.soton.ac.uk/OMII_security, where later versions will also be posted

private key) but gives the other one (the public key) away to anyone who wishes to communicate with them.

Now, a message encrypted with the public key can only be read by the individual who possesses the private key. Hence any user can direct a message to a known destination, knowing that it can't be read by anyone else, simply by encrypting using the public key of that destination.

Conversely, the owner of the private key can encrypt messages with that key. These messages can be sent to anyone. Because they can be decrypted with the public key, anyone can read them. The strength of this mechanism is that the receiver of the message can be sure that it was sent by the (unique) individual who holds the private key. This is the basis of a digital signature. The message has been "signed" by the sender.

How does the receiver know that the sender is actually who they claim to be? All the receiver really knows is that the sender possesses the private key. This is the issue of trust. In practice, the sender supplies a certificate that warrants that they are indeed who they claim to be. The certificate associates a specific named entity, identified through a 'Distinguished Name', with a specific public key.

How do we know the certificate is genuine? The certificate itself has been signed (using digital signature) by a trusted third party (a certificate authority, or CA). Clearly, this leads to a chain of trust being followed, which must end with a CA that the receiver is prepared to trust.

These then are the basic underpinnings of security on the internet, made possible by the use of cryptography.

1. Confidentiality can be implemented by encryption.
2. Data integrity can be implemented by digital signature and
3. Mutual Trust can be established by an infrastructure of Certificate Authorities.

These aspects of security will be discussed in greater detail as we review the various security technologies available to us.

Whether we use certificates, or the more elementary username/password pairs, all security systems are open to attack. WSI, the Web Services Interoperability organisation, in constructing its Basic Security Profile has documented many of these attacks. Recently, Sasse and Croft have documented the problems that arise when security systems make life difficult for users. Sharing of passwords and private keys is not uncommon and in these circumstances most security solutions break. Until biometric means of identification become commonplace, these problems are likely to persist. This emphasises how important it is to make security technology easy to use, ideally to make it invisible.

Security Requirements

Security is a very broad topic, especially when discussed in terms of the attacks to which a system may be vulnerable. Rather than try to cover all of that here we will restrict ourselves effectively to those aspects of security that can be achieved by the

use of cryptography. This is not to imply that OMII is not concerned with the broader issues of security policy and its enforcement. Quite the contrary. Time has not allowed us to document our position on these broader issues. It is our intention to do that, either in a later version of this document or in a separate document, as our experience of deployment of OMII software develops.

Martin and Hopcroft have produced a more comprehensive paper on security requirements.

The principal requirements that concern us here are confidentiality, integrity, authentication, authorisation and non-repudiation all of which are achieved by the use of cryptography. We define these terms here now, for reference.

Confidentiality

Confidentiality is the assurance that a message in transit, or data held in a repository, can only be read by those who are properly authorised. Encrypted data can only be read by those who hold the decrypting-key. Clearly, confidentiality is compromised by the possibility that the decrypting key might be stolen.

For this reason, most techniques typically use a combination of symmetric and asymmetric encryption. The message is symmetrically encrypted using a short-lived symmetric key to protect it during the communications. This symmetric key is agrred between correspondents using a long-lived asymmetric key pair. The risk involved in losing the symmetric key is much less that that involved in losing the much longer-lived asymmetric private key. Obviously, it is vitally important that the user keeps the asymmetric private key very secure.

Integrity

Intregrity is the intrinsic assurance that data has not been tampered with. Again, this is ensured by encryption but used in a different way. By encrypting with a private key, the data is effectively signed by the owner of the private key. Anyone can read the data (because they have access to the public key), but only the owner of the private key can alter it. A common way to improve the performance of this mechanism is to take a digest of the data and to sign just the digest. The receiver can of course decrypt the digest, but cannot alter it in its encrypted form. The decrypted digest can be compared to the original data to check that data's integrity. When confidentiality and integrity are both required (normal) it is conventional to sign the data first and then to encrypt it for confidentiality. Signing uses the sender's private key, encryption uses the receiver's public key.

Authentication

Authentication is the assurance that a principal (user or resource) is indeed who they claim to be. The simplest form of authentication is username/password. Here, by prior arrangement, the service requiring to authenticate a user registers a password for that user. When the user requests access, the service can check that the correct password is offered. The assumption is that only the user and the service know the correct password (a shared secret). A stronger alternative to username/ password is the use of a digital certificate (usually an X509 certificate). This document binds the identity of a user to their public key. When offered in combination with data signed using the

user's (secret) private key, the certificate guarantees the authenticity of the data, thus effectively authenticating the sender. The certificate itself is signed by some mutually trusted third party.

Authorisation

Remote, shared resources are protected by only allowing authorised access. A remote resource may list the names of the principals allowed to access it or may organise access by role name or group name, in which case the user requesting access must belong to one of the authorised groups. This is the way that file stores are protected on shared machines and also the way that private web sites are protected. In general, a user authenticates themselves by some means, thus establishing their identity. The identity is then mapped to a decision about whether or not access is allowed. Authorisation is usually under control of the resource owner and is thus an action performed by that owner.

Non Repudiation

Non-repudiation is the assurance that an action performed by an individual cannot subsequently be denied. This is more demanding than digital signature alone in that both parties to a shared action (e.g. exchange of contract) must by assured at the time of agreement that the other provides guarantee. This guarantee usually therefore involves a trusted third party, who among other things verifies the signatures and is responsible for time-stamping the guarantee.

The problem that is being solved here is that parties to a transaction may subsequently default or deny participation in the transaction (repudiation). In order to try to avert subsequent repudiation or to resolve future disputes, a non-repudiation service may be invoked at the time of the transaction. A non-repudiation service "*involves the generation, verification and recording of evidence, and the subsequent retrieval and re-verification of this evidence in order to resolve disputes. Disputes cannot be resolved unless the evidence has been previously recorded*." ISO 10181-4:1997.

This is clearly more demanding than digital signatures alone. If the transacting parties are unable to resolve any future dispute themselves, they will require an adjudicator to be involved. An adjudicator will normally only accept evidence that is assured by one or more trusted third parties. Trusted third parties may be used to validate the various components of a transaction, including: time stamps, certificates, signatures, message deliveries, and transaction details.

Security Use Cases

We describe three increasingly demanding use cases for Grid security. These are by no means exhaustive of the security scenarios that we encounter in large-scale grid applications. Such a collection of use cases is urgently needed, but is beyond the current scope of this short note. However, the WSI report on Security Challenges contains some detailed scenarios which are essentially the component parts of such use-cases.

Use Case 1: Use of a nationally deployed grid service
Use Case 2: Shared Data Resources

Use Case 1: Use of a nationally deployed grid service

A number of clusters (and other specialised resources) that are geographically dispersed are to be made available to a community of users normally remote from most if not all of these resources. Users must be able to identify themselves (authentication) and are then allowed access to a restricted subset of the resources (authorisation). Users wish to run applications on these resources, accessing data on different resources, and generally federate these activities into workflows. Users require confidentiality of their data in transit, and when stored remotely, and require integrity of that data at all times (confidentiality, integrity). Moreover, workflows mean that applications on one resource will call applications on other resources and users do not wish to have to manually repeat their authentication to each of these resources (single sign-on).

Use Case 2: Shared Data Resources

A number of geographically distributed data sources are required by a single application or workflow. These may be seen as a single homogeneous resource or may be of such a heterogeneous nature that it is not possible to view them as such. The community of users that share these data resources wish to be able to manage who has access to which data and which form of access they have (authentication, authorisation). The application or workflow that they use may have rights that exceed those of the user that started it running.

Use Case 3: Virtual Organisation

A community of users wishes to share resources with all the freedoms and restrictions of use cases 1 and 2 but with the added requirement that any individual may be in many communities (virtual organisations) and may be able to federate their capabilities across the communities in which they are involved. Virtual organisations might be long-lived and relatively static, or they might be very dynamic and exist for only short periods of time. Some of the individuals involved in these communities may be computer applications (rather than people) to whom rights have been delegated.

Although we will not attempt to present specific examples of these use-cases here, in the following sections we will describe the technologies that contribute to the conventional solutions to the questions these use cases raise. In a later document, or in a later version of this document, we will attempt to show how these technologies can together tackle many of the security issues involved in these use-cases.

Security Technologies

Transport Layer Security (TLS)

Transport Layer Security manifests itself in the ubiquitous Secure Sockets Layer (SSL) and in the use of HTTP over SSL which we know as the https protocol. This

technology has been developed primarily for the simple case of a client securely accessing a server.

The technology is important because it is deployed already on a global scale. For example all serious browsers implement the client half of the https protocol and for most users and for most purposes, https is invisible to them.

The https protocol works as follows. The server has a private/public key pair. The client requests the public key (it arrives in a certificate) and encrypts an initial message to the server using this key. This message includes a newly generated symmetric key[2] to be used for encrypting all subsequent messages in this session in both directions.

So the client and server can now communicate without intrusion. Moreover, by presenting a public key (and a valid certificate) and subsequently decrypting the message encrypted with its corresponding private key, the server has authenticated itself to the client.

The usual issue of trust is established by the use of certificates signed by mutually trusted Certification Authorities (CAs). This is, for example, how you communicate confidentially with your online banking service. Note that even though it is only the server that has a public key, communication in both directions is encrypted and thus confidential. This is a significant simplifying factor in the deployment of https. There is no need for clients to have their own public/private key pairs.

The use of a public key certificate by the server effectively authenticates the server to the client. Normally the client must authenticate themselves in some other way, such as username/password (this is what your bank does).

However, the https protocol also has provision for the server authenticating the client. This requires the client to have their own certificate which is presented in the initial stages of the https protocol. Client authentication is optional and not often used, precisely because of the user's need to obtain and care for an X509 certificate. In a Grid context, however, where the user will make use of many remote resources, this requirement will be less onerous than looking after many username/password pairs.

The certificates we have been referring to generically are, in practice, X509 (version 3) certificates which have a standardised format and have themselves become ubiquitous in internet distributed computing.

The main weakness with TLS is that it is point-to-point. If A sends a message to B for onward transmission to C, using https, then A will use B's public key. The message will be decrypted by B and then encrypted with C's public key for onward transmission. B will be able to read the message. If A doesn't want B to read the message then A must first encrypt the message with a secret shared only with C before committing to the https conversation with B. This is a problem that Message Level Security (MLS) overcomes, although MLS is much more than just that.

---

[2] Actually, the client simply sends a random number. Both client and server then use this same random number to generate the same new session key

Familiar security solutions that make effective use of https manifest themselves in both secure web servers and in SSH[3], the popular secure (remote) shell implementation found on all serious Linux servers and elsewhere.

These solutions use https for encryption and use username/password for authentication and authorisation. In the case of web servers, for example, areas of the web space can be made accessible to named users or users who are assigned specific roles. Access is granted only if the correct password is given. Of course, https can arrange to pass that password to the server securely. When the username/password validation is integrated with the login requirement of the machine or the domain being accessed, this is a powerful way of establishing secure collaboration, short of implementing MLS. This use of https is also available for web services published on public servers and is adequate for many purposes.

SSH is a similar story. This gives secure command-line access, with associated file transfer, to remote machines on which the user has an account. Its deployment and use are trivial. Its power is only restricted by its point-to-point nature.

A significant requirement in Grid computing is Single Sign-On (SSO) and this has many ramifications. When using many remote resources in different administrative domains, the user does not want to have to repeatedly authenticate themselves. Signing-on once per session should be sufficient. This means that either every remote resource authenticates the user using the same password (difficult to establish and maintain) or that each resource can map the user's Grid identity to a local identity for access control purposes. The use of X509 certificates can enable SSO by using the certificate to carry sufficient additional attributes to implicitly authenticate the user to each resource in turn. The identity carried by the certificate, and authenticated by it, must be mapped locally to access rights in some way, simplest of which is to map the authenticated id to a local login.

Message Level Security (MLS)

Message Level Security, as we shall discuss it here, is a collection of technologies for securing XML messages over HTTP, in particular for securing SOAP messages.

It is thus particularly important for distributed architectures based on Web Services.

MLS technologies, or some of them, are grouped together under the WS-Security standard.

In a nutshell, WS-Security provides the means for transporting a security token (such as an X509 certificate) in a SOAP header and for using that token and/or other tokens to sign all or part of the SOAP body. It also provides the means for encrypting all or part of the body. Indeed the parts of the body can be (will be) signed and encrypted by different tokens, thus selectively allowing the body to be read by intermediaries and the ultimate receiver.

---

[3] SSH uses the SSH protocol, rather than SSL, which establishes a secure session in a similar way to https. Although they are different, to a first approximation SSH is what you get from running a remote shell over https.

If MLS encryption is use, the body parts are encrypted by the use of XML-Encryption, a well established technology from W3C that uses conventional cryptography, to encrypt XML messages in a way that is self-describing. The messages carry the identities of the encryption algorithms used.

The body parts of a SOAP message are signed by the use of XML-Signature, another well established technology from W3C that specifies how XML should be normalised and signed.

WS-Security defines how the signed and encrypted parts are arranged in the message and how the relevant tokens are carried to allow signatures to be verified. It also defines how the encryption, normalisation and signature algorithms are specified in the message.

The security tokens carried by WS-Security do not have to be X509 public key certificates, although these are the type most commonly used for signature. Security tokens can also be Kerberos tickets, encrypted username/password pairs or XML security tokens such as SAML (Security Assertion Markup Language). In fact, SAML is considered to be a proper part of WS-Security, albeit an optional part.

SAML is very powerful. It allows detailed assertions about the rights and permissions of the (sender of the) message. These can be checked by the receiver against a policy that the receiver implements.

WS-Security, on its own, achieves much of what is required in Grid architectures. However, there are more specifications in the works which extend the capabilities of WS-Security.

WS-SecureConversation is a MLS equivalent of https. WS-Security as specified above defines encryption using public keys. This can be inefficient. WS-SecureConversation defines a protocol for agreeing a shared session key, just as https does, thus making longer conversations less costly

When WS-Security is used to pass security tokens to gain access to resources, the server supporting those resources will implement policies about who is authorised to access what. The simplest access policy is implemented by access control lists (ACL) at the server where the requestor's identity is checked. Then it is adequate to pass an identity security token such as an X509 certificate. When a more elaborate policy is required, the server will use WS-Policy to specify access rules and the security token will need to carry many more attributes, in which case a SAML token is more appropriate.

Security Assertion Markup Language SAML

SAML is an XML notation for making statements about the properties of a principal in a security scenario. As such, it is referred to generically as a security token. Just as an X509 certificate or an encrypted username/password pair tells you something about a principal that offers them, then a set of SAML statements can do the same, but with much greater detail.

There are three types of SAML assertions: authentication statements, attribute statements and authorisation decision statements. These respectively say something about how a user was authenticated, what attributes a user has, and what a user is authorised to do.

An example of a SAML statement (assertion) might be that "John Doe with email jdoe@company.co.uk was authenticated by ecs.soton.ac.uk using a password at 10.00am on 1 Jan 2005"

SAML assertions can carry a significant amount of detailed information about an individual and their rights. In practice SAML will also typically carry certificates on behalf of the user it is describing. It thus combines the merits of all authentication protocols.

PrivilEge and Role Management Infrastructure Standard PERMIS.

PERMIS implements authorisation on top of an existing authentication system. It makes access decisions to resources based on policies specified in XML. This policy file is similar in concept to access/deny lists written for firewalls, but it is augmented with role information.

Fundamentally, PERMIS implements "Role Based Access Control" (RBAC). This means that access to resources is defined in terms of a user's roles (or attributes), amongst other things. E.g., Software developers are allowed access to the CVS repository. Hence privileges are associated with roles, not users. Additionally, this policy file can contain information about role hierarchies (privilege inheritance), who is trusted to allocate which roles to whom, and delegation of duties (privilege delegation).
The resulting XML policy specification is stored in an X509 Attribute Certificate (AC), signed by the policy creator, so that it is tamper resistant. Similarly role-assignment ACs, identify a user with a role. Both policy specification and role assignment ACs are stored in one or more LDAP databases and retrieved by the PERMIS access control decision function when required to check authorisation for a given user.

PERMIS can work with existing X509 authentication systems (PKI) and username/password systems. It is also compatible with Shibboleth. A Grid/Globus enabled version of the PERMIS access control decision function that responds to incoming SAML requests is also available.

The policy specification language is similar to, but predates, XACML described below. A primary difference is that PERMIS supports delegation of authority whilst XACML does not.

XML Access Control Markup Language XACML

XACML is the OASIS specification for representing access policy. XACML is yet another XML based language.

XACML provides means for flexible definition of the rules and policies used to protect a resource. It provides the means whereby an authorization decision can be based on attributes of both the subject and the resource. It provides a method for

identifying the policy that applies to a given action, based upon the values of attributes of the subjects, the resource and the action.

XACML is more elaborate and less mature than PERMIS. However, it has the significant advantage of having broader international consensus and the consideration of OASIS.

Globus Security Infrastructure GSI

The Grid has requirements for long-running collaborations that federate the use of many distributed resources. Consequently, it is necessary for (remote) programs to operate on a user's behalf without the user being present. The user must be prepared to delegate authority to a program and the program must be able to authenticate itself as an entity having that authority.

The Globus Security Infrastructure (GSI) specialises in this aspect of delegation. It has been set up to accept proxy-certificates as authenticating the user who issued them. So, a user wishing to delegate authority, acts like a CA and issues a certificate signed by themselves using their own certificate. This proxy-certificate is short lived (a few hours, typically) but gives the program possessing it access to resources with the authority of the user that signed it. Recursive delegation is of course possible.

A significant enabling mechanism for GSI is the availability of a portal to manage your client certificate. This portal is called myProxy. In order to free the client from only operating from the workstation on which their client certificate is installed, myProxy supports a mechanism for generating a proxy certificate (with the authority of the client's long-term certificate) and saving it (behind a password) on a remotely accessible database. The user can then carry out Grid operations from remote locations simply by retrieving the proxy and handing it to the service on which they require authentication. In many respects, this gives X509 the ease of use of username/password but with the added benefit of SSO over distinct domains. It is however dependent on acceptance of proxy certificates, which presents problems for some environments.

The SSO capability implicit within GSI has been integrated into the SSH protocol within the GSISSH package. This gives secure command-line access, with associated file transfer, to remote machines on which the user has an account. The GSI proxy created on the 'home' machine is passed onto the remote machine, allowing further access to other machines without the user having to respond to any further authorisation challenges – providing an SSO environment for terminal access to remote machines. The restrictive point-to-point nature of SSH (through its implicit client/server interaction) is therefore removed as following each GSISSH login an authentication token exists on the server that can be used by a client to a access further servers.

Federation, Federated Identity and Delegation

WS-Federation has the same goal of delegating authorisations through authenticated identity, but a different solution. This specification was implemented by IBM and Microsoft in late 2003 and demonstrated at a very public interop demonstration (Ferguson et al). They showed that delegation in WS-Federation worked. At the same

time they showed that IBM Web Service implementations could interoperate with Microsoft Web Service implementations.

WS-Federation is a protocol that runs over WS-Security. It supports a notion of federated identity. The demonstration given by IBM and Microsoft showed (among other things) a user in company A accessing a Web Service in company B, using the credentials that allowed the original login on a company A workstation. By accessing the remote (company B) web service as an apparent employee of company A, the WS-Federation protocol calls-back to company A, accessing an identity server, to check that the credentials offered are indeed valid.

This notion of calling-back to an identity authority has the significant benefit that the employee's credentials are only validated at (stored at) one place. If the employee is to have their rights modified, such as if they leave, there is only one location to be updated.

On a global scale Shibboleth implements a similar notion of federated identity. Identity credentials are issued by a user's principal (e.g their employer) and are used globally to access services anywhere that implements the Shibboleth protocol. It is conceivable that the very institutions that are involved in Grid computing will implement Shibboleth in the next few years and that most of what the Grid community requires will be realised by a combination of WS-SecureConverstion and Shibboleth. There is a new NMI project in the US whose purpose is to integrate Shibboleth and Globus Toolkit, so that users will be able to use their existing login credentials to run Grid jobs.

WSI Basic Security Profile

The Web Services Interoperability organisation has published a Basic Profile which determines how SOAP and the Web Services technologies should be deployed in order to simplify the problems that might arise when components developed in different organisations are required to interoperate.

To accompany this profile there is also a Basic Security Profile (BSP) that recommends how the various security technologies we have discussed should be deployed. The central technology they discus is of course WS-Security, but realities of life being what they are, BSP recognises that the MLS technology will usually be deployed on to of TLS (specifically https). The BSP therefore advises which bits of each technology should be used for which purpose and which might perhaps be more complex than necessary for the anticipated threats.

Although the BSP is neutral on favouring one technology over another, reading the report does convince one that encryption should be implemented by TLS and that integrity should be guaranteed by enforcing MLS signature. Clearly it is important for both parties to a communication to agree which parts of a message should be signed, but BSP give the minimal rules for which must be signed in order to ensure the implicit integrity of the message (e/g. these headers do belong to this body).

Interoperable Security Solutions

Notwithstanding the many technology choices that we have for implementing security in a Grid environment, or indeed possibly because of them, it is not possible to legislate a world in which all deployed systems use the same security infrastructure.

Consequently, interoperability between security solutions is going to be a vital enabling technology if the full promise of the Grid is to be realised.

Of particular importance in the Grid today are WS-Security and GSI. Although Globus are delivering a judicious mix of these technologies in GT4, others including OMII will wish to stay close to commercial standards.

We need therefore to devise means of interoperating across different technologies, and even across different deployments of the same technology.

Ironically, there is a fortunate (or unfortunate, depending on your point-of-view) relationship between the implicit weakness in security technologies and the need for them to interoperate. One technology can exploit the weaknesses in another in order to communicate with it. The converse is probably also true. If a security technology is open to interoperation, it is probably also open to attack.

Conclusions

TLS is OK for most things, ideal for many, but MLS provides greater flexibility.

TLS has significant performance advantages over MLS as well has having the significant properties of ubiquity (everyone already has it) and invisibility (they hardly know they are using it)

Can use TLS for encryption and MLS for signature in simple distributed architectures where intermediaries can be trusted. This is a very powerful and increasingly ubiquitous use of WS-Security whose deployment is consistent with the WSI BSP.

Full MLS involves more work on the user's part (eg management of certificates) as well as more infrastructure.

Implementations of WS-Security technologies are still relatively immature. They need to become as invisible as https.

WS-SecureConversation is a sensible goal for Grid middleware providers. It gives end-to-end security where https only gives point-to-point.

Delegation is important in the Grid. WS-Federation (or whatever it becomes) is a sensible longer-term goal. This may be overtaken by wide-scale deployment of Shibboleth. Meanwhile, GSI is an available option. When Globus and Shibboleth are integrated this could well offer a viable solution and migration path from where we are today.

Authorisation and virtual organisations – best contemporary options are to reproduce the username and role-based access control familiar from contemporary file systems.

Acknowledgements

Various comments from David Chadwick and other members of the STF, for which we are very grateful, have been incorporated in this version of the document.

References

1. Alfieri et al, VOMS, an authorisation system for virtual organisations
2. Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. A National-Scale Authentication Infrastructure. IEEE Computer, 33 (12). 60-66. 2000.
3. Chadwick, D, The EC PERMIS Project http://sec.isi.salford.ac.uk/openPermis/index.html
4. Chapra, V et al, Apache Tomcat Security Handbook, Wrox Press, 2003
5. DataGrid, VOMS Architecture v1.1, 2003. http://grid-auth.infn.it/docs/VOMS-v1_1.pdf
6. Ferguson, D, B Lovering, Tony Storey and John Shewchuk, Secure, Reliable, Transacted Web Services: Architecture and Composition, http://www-106.ibm.com/developerworks/webservices/library/ws-securtrans/ (2003)
7. Farrell, S. and Housley, R. An Internet Attribute Certificate Profile for Authorization. Internet Engineering Task Force, RFC 3281, 2002
8. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S., A Security Architecture for Computational Grids. 5th ACM Conference on Computer and Communications Security, 1998
9. Globus Security Team, Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective
10. Graham, S et al. Building Web Services with Java, SAMS 2004 (Chapter 9)
11. IBM and Microsoft, Security in a Web Services World: A Proposed Architecture and Roadmap, http://www-128.ibm.com/developerworks/webservices/library/ws-secmap/
12. Martin ,A and Hopcroft M, A Critical Survey of Grid Security Requirements and Technologies, Technical Report PRG-RR-03-15, Oxford University Computing Laboratory, August 2003
13. Novotny, J., Tuecke, S., and Welch, V., An Online Credential Repository for the the Grid: MyProxy
14. Oaks, S, Java Security, O'Reilly 2001
15. OASIS, Web Services Security (WS-Security). http://www.oasis-open.org/specs/
16. OASIS, "Security Assertion Markup Language (SAML) 1.1 Specification", http://www.oasis-open.org/specs/
17. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration (CAS). IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002
18. Rosenberg J and D. L. Remy, Securing Web Services with WS-Security, SAMS, 2004
19. Sasse, A and Croft B, Security Needs and Usability in e-Science Projects (2005)

20. Security Task Force, Grid Security Technology Roadmap, STF Working Document Draft 1.2 (2004?)
21. Shibboleth Project, Internet2, http://shibboleth.internet2.edu/
22. Surridge, M. Rough Guide to Security, http://eprints.ecs.soton.ac.uk/7286/01/RoughGuideToGridSecurityV1_1a.pdf
23. Tuecke, S, Von Welch, Doug Engert, Laura Pearlman, and Mary Thompson, RFC3820: Internet X.509 Public Key Infrastructure Proxy Certificate Profile. 2004
24. Von Welch, Frank Siebenlist, Sam Meder, Laura Pearlman, Use of SAML for OGSA Authorization (2003)
25. Von Welch et al, Attributes, Anonymity and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration (2004)
26. WS-I, "Web Services Interoperability (WS-I) Interoperability Profile 1.0a." http://www.ws-i.org .
27. WS-I, "Basic Security Profile Version 1.0 (Working group draft)." http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html, 2004.
28. WS-I Security Challenges, Threats and Countermeasures, http://www.ws-i.org
29. WS-Trust – http://www-106.ibm.com/developerworks/webservices/library/ws-trust
30. WS-SecurityPolicy – http://www-106.ibm.com/developerworks/webservices/library/ws-secpol
31. XML-Encryption http://www.w3.org/Encryption/2001/
32. XML-Signature http://www.w3.org/Signature/
33. WS-SecureConversation, IBM, Microsoft, RSA Security and VeriSign Web Services Secure Conversation Language, Version 1.0, 2002, http://www-106.ibm.com/developerworks/library/specification/ws-secon/

Appendix A: Maturity of Specifications and Implementations

Specification	Standards Body	Available Implementations
WS-Security	OASIS	Apache
XML-Signature	W3C	Apache
XML-Encryption	W3C	Apache
SAML	OASIS	
PERMIS		Salford
XACML	OASIS	SUN
WS-SecureConversation	see IBM+MS	
WS-Policy	see IBM+MS	
WS-SecurityPolicy	see IBM+MS	
WS-Trust	see IBM+MS	
WS-Federation	see IBM+MS	
GSI	GGF	Globus
Shibboleth		

For OASIS see http://www.oasis-open.org/specs/

For W3C see http://www.w3.org/

IBM+MS not clear whether these important specifications have yet been submitted to a standards body. See references for links or Google the specification name and follow an IBM developer works link