

Using WS-Security and SAML for Internet Single Sign On

Darren Miller

Abstract

Single Sign On solutions are desirable to reduce the number of usernames and passwords that each user has to manage. Managing multiple accounts and multiple sites can be burdensome to the user and can lead to security exposures. SAML and WS-Security are emerging technologies for use in allowing authentication assertions to be passed across disparate systems and across company boundaries. These protocols are open standards can be used for creating solutions that enable single sign on. This paper looks at how these protocols meet the need for credential passing in a distributed environment of multiple service providers

Introduction

Single sign-on (SSO) is a growing field of interest in the area of e-commerce. Companies are deploying applications intended for user populations that were previously distinct: internal users, business partners, and customers. In turn, users are accessing a variety of sites and applications in order to conduct business online. In general, users who access many different, unrelated sites are faced with difficulties in maintaining IDs, passwords, and account information for each of the sites.

Internet based single sign-on will help individual users by simplifying access to web resources. It will also help businesses manage identities of their users. This paper will research the considerations needed to create a reliable single sign-on environment for a population of users accessing a population of unrelated web sites.

The goal is to analyze emerging technologies, such as SAML and WS-Security for their applicability to creating workable single sign-on solutions. The scope assumes that existing authentication systems are in place and adequate for authenticating the user. This paper will explore the ability to share credential information in a secure manner.

Overview

Many organizations, especially larger ones, have a variety of systems, often with varying authentication schemes. There's a desire to reduce the number of usernames and passwords that users have to manage, and limit the number of times users need to login to access resources internal to the company. There are also operational benefits by streamlining the processes for creating and disabling user accounts and granting access to resources. It simplifies the user experience and can reduce the number of support calls from users who've forgotten their passwords to access certain rarely-used systems.

Some organizations have made some progress in deploying single sign-on solutions within the confines of their own set of systems. There are a number of architectures and vendor products to address this desire. This can be challenging even within a single organization's environment. However, the Internet environment, where users access multiple sites, is even more complex and can raise a number of issues.

Issues with multiple sign-on

Having accounts at multiple sites creates issues of username and password management. The issue of lost usernames or passwords represents one of the largest support issues that organizations face.

Single password synchronization

One simple and very common approach for users who access multiple sites involves the user creating an account with the same username for each site and keeping passwords in sync between them. This stop-gap solution is challenging given the complexities of the many-to-many user-to-site relationships.

Keeping usernames and passwords in sync can be particularly unwieldy in an Internet environment. For one thing, username requirements may vary from site to site, meaning that a username that is legitimate at one site may not be acceptable at another. In addition, the user's preferred username may already be in use at some sites.

Different sites will also have different policies regarding the composition of passwords, the length of time before a password must be changed, and the number of previous passwords that are preserved to prevent repeating passwords.

This solution also creates a risk in that, since each site has to know the user's credentials, it could use that information to impersonate that user at the other sites. This weakens the integrity of the username/password scheme as a method of identification. A way that this could be exploited is for a malicious site to be set up, and the user induced to create an account there, in the hopes of gaining access to the user's credentials. This sort of attack, known as "phishing", is a frequent occurrence and users have had their identities compromised.

In addition, if any of the sites with which the user transacts business becomes compromised, and if the password is stored in a recoverable format, the intruder will have gained the ability to impersonate the user. The user's identity becomes only as strong as the weakest site's security, and a compromise can result in identity theft.

Multiple weak passwords

A user may have separate usernames and password at the various sites. However, users often find it hard to pick out good strong, yet memorable, passwords, and trying to create multiple passwords can be challenging. As a result, passwords are often weak and easily breakable.

Multiple unrelated passwords could be more secure, but they create management headaches for the users, who often fall into bad habits to make things easier.

Writing down username/passwords

Another approach users take is to write down their passwords on a list. Keeping the list up to date can be a hassle, as the user joins new web sites and as their passwords change. These lists may be stored in a file, which in many cases are printed and taped right to their monitors, tucked under the keyboards, or placed in a desk drawer. Someone with access to the user's work environment could gain access to their list of identities and

passwords. If the list is stored in an unencrypted format on the file system, hacking techniques might be exploited in an attempt to swipe the file remotely.

There are software tools available for remembering and automatically filling in password forms. For example, the Internet Explorer AutoComplete feature has the ability to remember passwords. Anyone with access to the computer could impersonate the user.

Usage scenarios

Extranet

Extranets, where a company grants access to selected systems to external business partners, is an example of a case where Internet single sign-on would be helpful, especially in the likely case where the external business partners are accessing systems at other companies as well.

An example scenario is that of the independent insurance agent. Independent agents do business with a variety of insurance carriers. Agents log in to the carriers' sites to use systems for quoting, rating, issuance, and customer service applications. In some cases, agents use integrated agency management systems that attempt to front-end business transactions on the agent's behalf.

Each insurance carrier will most likely have its own username format and password requirements. As a result, agents must have IDs for each of the insurance companies that they represent and have to keep track of the IDs and passwords. This can lead to where the agents have the same password, or variations, across the different sites. Some create "cheat sheets" of their usernames and passwords. It can also lead to the sharing of IDs and passwords between agents within the same office if one agent is having problems getting into a certain site.

Insurance carriers are financial institutions that have non-public information on their insureds and agents. From the perspective of the insurance carrier, they are bound by ethical considerations and various state and federal laws to protect their data. Furthermore, no insurance carriers would want to trust another carrier to use or store its credentials, or to authenticate users on its behalf. A carrier wants to ensure that the individual on whose behalf it is issuing business is in fact the agent in question.

End-user e-commerce

In another example, Internet users often access many different merchant sites. The user may not want to create and maintain accounts at each of the individual merchant sites. The user may forget usernames or passwords to sites that are very rarely used.

The merchant, on the other hand, needs to get the information needed to complete the transaction in a reliable way and to be assured that it the user is the one submitting the order or accessing account information.

If a merchant requires a login for the user to access the site or to place an order, the merchant is left in a position where it must provide mechanisms for password reset or recovery. A help desk may be required to handle call-in complaints, or an automated self-service system may have to be provided.

SOAP and Web Services

The Simple Object Access Protocol (SOAP) is a method for encoding remote method calls into XML messages for application-to-application communication. Web Services use SOAP messages transmitted via HTTP. Users do not interact with Web Services directly. Instead, a client or web application may invoke one or many Web Services to complete the user's request. The Web Services

themselves may call other Web Services in the course of carrying out the request. Web Services are being viewed as an open means of allowing applications to talk to one another within a company or even through firewalls across company boundaries.

In this case, the applications invoking Web Services may need to pass the user context. This user context may need to be carried along through various systems and between companies as requests are being processed. An example often cited for such systems are travel bureau sites, which may contact airlines, hotels, or auto rental sites on behalf of the user. Another could be a financial institution, such as a bank, that may provide integrated access to various other service providers, such as brokerages, analysts, and such. In either scenario, it might not be reasonable for the user to maintain individual accounts at all the sites involved in the transactions.

Obstacles to Internet Single Sign-on

Limitation of cookies

HTTP is by nature a stateless protocol. Each client request is considered independently of any previous request. Even multiple requests within a single TCP session must be treated as independent requests.

A single user session typically spans many TCP sessions, even when interacting with a single service provider. A user session can involve different web servers, due to load balancing or the distribution of functionality across multiple servers. Client-side addresses can also change during a session due to dynamic addressing, network address translation, and proxying.

In order to maintain state between user requests, a mechanism known as "HTTP cookies" is described in RFC2965 [1]. An authenticated user session requires the ability to link each request to an authentication credential, and cookies are often used in various ways to achieve this. A cookie can contain a session identifier, a user token, or actual user credentials (though not recommended). Cookies can be considered a connectionless way to maintain state (and thus user authentication) because the information is sent back and forth with each request, independently of the underlying transport.

However, cookies have some limitations that make them difficult to use for Internet SSO. For one thing, cookies are only sent back to the same domain from which they originated. This greatly limits their ability to be used as a mechanism for cross-domain authentication.

Cookies are also easily stolen. An eavesdropper could intercept the cookies in transit, or a hacker could use client-side scripting techniques or cross-site scripting to try and acquire the cookies. All cookies for a given site are always sent with each request to that site, which means they may pass over the Internet much more than required for session management, making them more susceptible to interception or leaking.

In a shared hosting environment, cookies might be exposed to other web sites sharing the same domain if care isn't taken to limit them. The "Domain," "Path," and "Port" attributes are can limit the scope of the cookie and prevent it from being accessed elsewhere. There also is a "Secure" cookie attribute, which tells the client not to play the cookie over a non-secure session. However, the implementation of these attributes is optional.

Some cookies are session cookies, which means that they are only valid for a given browser session. If the user doesn't close the browser, the cookie maybe considered

still valid, if the site doesn't time out idle sessions. Persistent cookies are stored to disk. If someone gains access to a computer with an active browser session and active session cookies, or with persistent cookies, the user can be impersonated.

Limitations of certificate-based authentication

When a client initiates an HTTPS session, by the SSL/TLS protocol, the server responds with information, including its certificate, which the client can use to authenticate the server. For most HTTPS sessions, that is all that is required. This allows clients to make ad-hoc requests for server resources securely.

However, the SSL/TLS protocol also allows for the web server to require a certificate of the client. This certificate then can be mapped back to user credentials to authenticate the user.

Possession of the private key associated with a valid certificate can be considered to be proof of identity of the user. During SSL/TLS session setup, proof of ownership of the private key corresponding to the certificate is established by having the party sign a nonce with the private key. The other party is assured that the private key is known by decrypting the signed value with the public key. The ownership of the certificate can also be combined with an additional check, such as prompting the user for the password associated with the account that owns the certificate for stronger two-factor authentication.

Certificates are cryptographically signed by a Certificate Authority, making them tamper-proof. Any attempts to modify the information on a certificate will make them invalid.

This authentication method is connection-oriented, since it is tied to the underlying transport. In fact, without a proper exchange of certificates, the SSL/TLS session will fail. The web server can use the provided certificate for subsequent authorization functions by reading the information contained in the server's (implementation specific) Request variables. The mechanism is still stateless unless state is maintained in some other way, such as with cookies.

In spite of the strengths of certificates, there are still limitations. Certificates are stronger authenticators as a means of proving identity than a username and password, since ownership of the private key is required in order to use the certificate. However, unless a smart-card is used, the certificate is stored on the client. This makes it unsuitable for use on an uncontrolled workstation, such as an Internet kiosk.

Use of certificates for cross-site authentication requires implementation of a public-key infrastructure (PKI) on an Internet-wide scale. At present, no such infrastructure of this scale exists supporting such a large population of users in a cross-organizational mode.

Use of client side authentication via certificates creates issues with proxies. Since SSL/TLS is designed for end-to-end authentication, there are issues with having an intermediary trying to pass the client side certificate all the way through to the server.

Also, SSL/TLS would require that all of the sessions be encrypted, even though most of the session may not require encryption. This can be a performance impact.

Problems with home-brew authentication

A given community that desires to share credentials could base their authentication on a custom-developed cross-site authentication scheme. This solution raises

problems of interoperability, since any new groups that desire to join the community will need to implement the custom protocols. Some decisions may be based on the underlying architectures of the incumbent systems, which may create lock-in.

More importantly, it is notoriously difficult to design good security protocols. Even protocols that have withstood mathematical evaluation have been shown to have exploitable flaws [3], [4]. Although design principles for protocols exist, it takes a great deal of expertise to build a protocol that is robust, secure, and provides the desired characteristics. Even "correct" protocols can be implemented incorrectly or can be used assuming they provide characteristics such as non-repudiation that they might not in fact guarantee [4].

Furthermore, the design of a home-brew authentication scheme precludes the use of vendor provided SDKs which can simplify the implementation. Some of the flaws in authentication systems can come from improper use or handling of secrets, such as private keys, in code. There are also issues that arise from coding errors, like buffer overruns, canonicalization-based mistakes, or random-number generation errors [5]. The upshot is that beyond the difficulty of designing security protocols is the additional challenge of implementing those protocols correctly. This implementation challenge is multiplied by the number of participants in the community.

Terminology

The terminology for SSO varies from provider to provider. The following terms are being used:

An **Authentication Provider (AP)** is a site that authenticates the user's credentials. This is also known in the literature as an Authentication Service Provider (ASP). An AP may be required to provide assurances that this authentication has taken place.

A **Service Provider (SP)** is an entity that contains protected resources or services that the User wishes to access.

An **Authentication Assertion (AA)** is a "token" provided by the AP to the User to be presented to the SP to identify the User and provide assurances that authentication has been performed by the AP.

An **Authentication Assertion Artifact (AAA)** is a reference ID to an AA provided by the AP to the User to be presented to the SP. The SP then makes a separate request presenting the AAA to the AP to get the full AA.

Internet Single Sign-on Protocol

The following steps describe the generic protocol flow for an Internet SSO.

1. User authenticates to the AP (optional).
2. User establishes a session to SP, requesting access to a protected resource.
3. SP redirects the User to the AP, providing a return SP URL identifier.
4. If User hasn't yet authenticated, or if authentication session has timed out, authenticate the User.
5. AP creates an AA to bind the User and the SP. Either the AA or an AAA is provided to the User with a redirect back to the SP.
6. The User is redirected back to the SP and presents the AA or AAA.

7. If an AAA is used, the SP makes a secondary request to the AP to get the full AA.
8. Once the AA is validated, the SP grants access to the site or resources.

This flow is generic (see Figure 1). Implementations will have nuanced variations of the above flow.

It is assumed that the AP has the ability to authenticate and maintain session information regarding the user. Once the user is validated to the SP, the SP may maintain state in some other locally significant way, such as cookies.

Requirements for Internet Single Sign-on

Pashalidis and Mitchell [6] discuss a number of considerations for SSO solution. These considerations include privacy and anonymity protection, ability of the user to access from different places including untrusted environments such as public terminals, operational aspects such as costs, and policy issues such as trust and ability to support conflict resolution and law enforcement.

A real-world Internet SSO solution must also address a number of concerns that to meet business needs.

Operational concerns surrounding APs. An AP must not be a single point of failure, it must have strong security practices and audit tracking. These goals may present difficulties for a small SSO community, but as the community grows in size it becomes more and more important. APs must maintain high availability and distributed operations.

Ability to provision accounts and disable them within a reasonable period of time. This needs to be able to happen in a dynamic fashion as employees come and go from companies. Given the growing concern of identity fraud, an AP needs the ability to block an account across all the member sites as soon as it becomes aware of such activity.

Solution must allow for new users, SPs, and APs entering the community. As businesses come and go within the real world, the community needs to be dynamic to reflect this.

Solution must limit the ability or risks of collusion. Malicious parties working together can undermine the SSO. Scenarios to be considered are collusion between users and SPs, between users and APs, between APs and SPs, between users, between SPs, and between APs. An AP has a particularly responsible role, since it could fabricate AA's of its users. The other participants in the community have to trust the AP to be able to manage its user lifecycle and to perform authentication of the users.

Any given SP must not be able to spoof the identity of any of the users at another SP. Even though an SP receives AA information in the normal course of interacting with the user, it must not be able to take that AA and use it to access another SP as the user.

Ability to support multiple roles. Users have different roles in life. A person may be an insurance agent for an agency accessing an insurance carrier, and may be a private customer of that insurance carrier.

Existing Technologies

Microsoft Passport

Microsoft has a web-based SSO service known as Passport, which has been in place since 1999. This service allows access to the Microsoft-owned web properties and participating merchant sites. In Passport, when a user attempts to access a resource site (SP) for the first time,

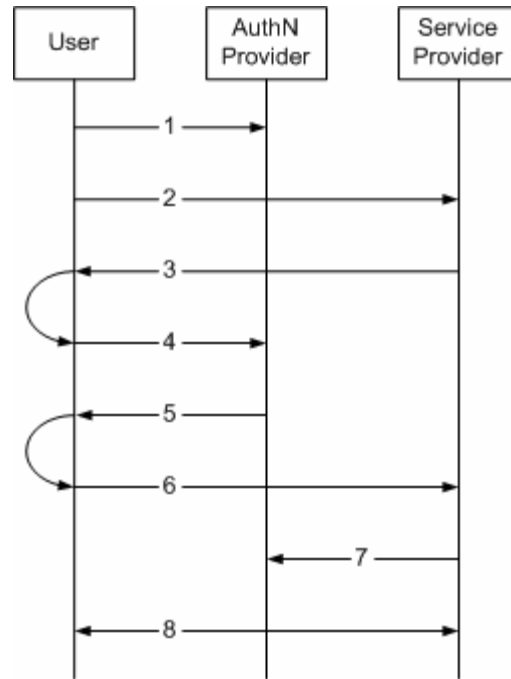


Figure 1: Generic SSO Protocol Flow

they are checked to see if they have a valid Passport cookie. If so, they are granted access to the resource. Otherwise, they are redirected to the Passport AP site. If the user doesn't have a current Passport session cookie, they are prompted to authenticate. Once authenticated, the Passport AP redirects the user back to the SP with an AA in the passed browser URL. The SP then sets a Passport cookie that it can access for the remainder of that session.

Passport is designed to work with existing browser technologies. As a result, it has some exposures based on limitations in the existing technologies [7], [8]. Some problems include interoperability with non-Microsoft browsers, the multiplicity of default root certifications, and dependencies on SSL and DNS. There are also social exposures such as the risk of a site spoofing the AP through "phishing" attacks and from bogus merchant sites.

Passport users are strictly end consumers. Microsoft does not offer Passport communities targeted to be business solutions. A business that wants to implement its own SSO system won't be able to use Passport.

Passport is an authentication system owned by a single entity. As a result, it represents a single point of attack for hackers to obtain credentials or against which to launch denial of service attacks. It also puts a lot of power into the hands of a single organization. Microsoft has made claims about opening up their framework for a more distributed model [9].

Passport is web-based. As such, its application to other protocols is limited.

Kerberos

Kerberos is a SSO system developed at MIT as a network authentication protocol. This protocol allows all participants within a single *realm* to access resources by means of an AA ticket. The AP and SP share secret keys used to encrypt and authenticate the requests.

Cross-site authentication is achieved by setting up explicit cross-certification trusts between realms. Cross-site trust can also be set up based on a hierarchical model

[10]. The requirement for shared keys creates difficulties in deploying Kerberos in a multi-organization context.

Kerberos supports multiple different protocols, as long as they are Kerberos-aware.

Emerging Technologies

SAML

SAML, Security Assertion Markup Language, is a standardized way of expressing authentication assertions in an XML format in a way that can be shared between participants in a federated identity environment. It is an open standard submitted by Liberty Alliance to OASIS.

SAML is not the actual authentication mechanism. It is a means of transforming facts about an authentication event that has taken place. SAML would be used in an Internet SSO environment as a means of passing an AA between the AP and the SP. The SAML bindings specifications leave the details of implementation to profiles, which describe the specifics of how to do this. Two profiles currently exist, the Browser/Artifact model and the Browser/POST model [11].

SAML relies on the concept of an “inter-site transport service” residing at the AP. This service receives the redirect from the SP, captures the information required to build the AA, and redirects the user back to the SP.

Browser/Artifact Profile

The Browser/Artifact model operates similar to how the Passport model works. Using the Browser/Artifact model, the user is redirected to the AP with information about the requested SP. After the user is authenticated, AP redirects the user to the SP with AAA information being passed by means of the browser query string. The profile prescribes that information sent between the AP, SP, and user should be sent over SSL/TLS to protect against interception of the query string and AA information.

The Browser/Artifact model is designed to work within the limitations of current browsers by passing information back and forth via URL redirects. Browsers and servers impose limitations on the length of URLs and so sometimes the full information can't be sent via the URL redirect. What is sent is a reference token to the full assertion at the AP site. In this case, the SP makes a second request to collect the full AA from the AP.

Browser/POST profile

The Browser/POST model is an alternative profile for exchanging information between the user, the AP and the SP. In this model the user with an enabled browser client receives the full SAML assertion which it in turn POSTs back to the SP. This is more complex to implement, as it requires either client-side automation via scripting or user interaction, or an enabled client.

Weaknesses

Like any Internet traffic, SAML is susceptible to eavesdropping attacks. Because of this, the standards recommend implementation of SSL/TLS based protection for confidentiality. Earlier versions of SAML-based profiles also suffered from Man-In-The-Middle attacks [12], [13]. Updates to the standards require the inclusion of an SP URL identifier in the authentication transfer between the SP to AP. In this way, the user can ensure that request is only being sent to the SP for which it was requested. Further, a rogue SP couldn't forward the AA unmodified to a second SP to impersonate user, since the second SP would receive an AA that was not valid for it.

Project Liberty Framework

The Liberty Alliance has a framework for wide-scale federated identity management, of which SAML is a piece. The framework designs a series of specifications designed to address more comprehensive aspects such as privacy, trust. Project Liberty has a profile for SAML usage called the Liberty-Enabled Client Profile [14]. This profile is an extension of the Browser/POST profile. It presumes that client side extensions are installed for automation.

Shibboleth

Shibboleth is an Internet2 project intended to create an Internet SSO solution to authorize a user from one organization to accessing resources at another site while being authorized by the user's home security domain. It uses SAML as the underlying technology for authentication assertions. Shibboleth is being implemented as a means of enabling cross-authorization of users between participating educational institutions. A number of universities are participating in alpha testing along with several well-known content providers [15].

WS-Security

WS-Security is an OASIS standard by an industry group led by IBM, Microsoft, and Verisign. It is a framework for attaching security header information to web services SOAP requests. SOAP Message Security has three components: authentication tokens, digital signatures, and confidentiality. WS-Security provides a flexible framework for exchanging different types of security tokens, including XML Username/Password Tokens, X.509 Certificates, and Kerberos tickets [16]. SAML tokens can be embedded within the SOAP headers [17], [18]. The WS-Encryption portion of the WS-Security suite allows for the preservation of message confidentiality independently of the underlying transport.

WS-Federation

WS-Federation is part of the Web Services Initiative, or WS-Security suite. It goes into specifications of how to use the other protocols to achieve the solution for federated identity management, including issues of policy and trust. A comparative analysis by Liberty Alliance [19] outlines the overlap between the specifications, but overall, the standards are not mutually exclusive and interoperability is likely, especially given that SAML is a common protocol between them.

Security Considerations

Profiles for SAML and WS-Security are guidelines for implementation that specify requirements for mitigating risks. These protocols still suffer from weaknesses endemic to the underlying technologies: browser leakage, cookies, DNS, and NTP. There are also still issues with social threats such as phishing, web site spoofing, and fraudulent SPs. The AP is a ripe point of attack for trying to gather user credentials. A site that pretends to be the AP could trick users into logging in. Tricks such as those described in [7], [8] could be applied to these SSO communities as well.

Most browsers aren't configured to actually check the server-side certificate in an SSL session. This gap creates the risk of a Man-In-The-Middle attacks are as a continued a concern. SAML and WS-Security reduce the exposure, since the user's credentials need not be sent to the individual SPs. As a result, the user session to the AP is the most at risk of compromise.

Implementation Concerns

Flexibility

SAML and WS-Security are flexible meet the needs of different communities interested in create a single sign –on environments. An employer could use it to authenticate its employees and pass the assertions to an external benefits providers, for example. The protocols' foundation in XML with its open extensible nature allows them to change to meet changing business needs.

Cost

These protocols represent an additional layer of complexity for developers. Implementing these protocols doesn't alleviate the need to have an authentication mechanism. However, a large, complex organization can use these protocols as a means of passing credentials back and forth between disparate systems and across organization boundaries.

Developing to standards-based protocols for SSO will also reduce development costs, as vendors provide SDKs for integrating their products with these standards. This reduces the risks of a faulty implementation and lowers the costs to develop, implement, and maintain them.

User acceptance

Users are interested in reducing the number of accounts and number of times they are required to log in. As such, there is likely to be a great deal of interest, as can already be seen by the popularity of Microsoft Passport. Companies can mandate the use of SAML or WS-Security based frameworks to their employees. Shibboleth is already involved in getting buy-in from the academic community. As more communities come on board, the adoption will spread.

Conclusions

New technologies are emerging to provide standard frameworks for Internet Single Sign-On. They are still fairly recent, but present opportunities for simplifying the process of maintaining credentials at multiple sites and for passing credentials between organizations.

Many of the concerns with a single sign-on community or federation are policy issues, such as requirements for trust for APs that are not directly addressed by SAML. These have to be addressed within a larger framework, such as Liberty, WS-Federation, and Shibboleth. These three seem to represent the three industry sectors interested in this: business, technology, and academia respectively.

Internet SSO systems create privacy concerns, including dangers of user tracking, account linking, or over-disclosure of information to SPs. Other concerns are operational, such as scalability, resistance to failure and attack.

References

- [1] IETF. "HTTP State Management Mechanism," RFC 2965. <http://www.ietf.org/rfc/rfc2965.txt>
- [2] Abadi M., Needham R., "Prudent Engineering Practice for Cryptographic protocols." In Proceedings of the 1994 IEEE Symposium on Security and Privacy, (1994) 122-136, IEEE Computer Society Press.
- [3] Anderson, R. and Needham, R. "Robustness principles for public key protocols." In Proc. Int'l. Conference on Advances in Cryptology (CRYPTO 95), volume 963 of Lecture Notes in Computer Science, pages 236--247. Springer-Verlag, (1995).
- [4] Syverson, P. and Cervesato, I. "The logic of authentication protocols." In Proceedings of 9th International Conference on Cooperative Information Systems (Trento, Italy, Sept. 2001), C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, Eds., vol. 2172 of Lecture Notes in Computer Science, Springer Verlag.
- [5] Howard, M and LeBlanc, D. Writing Secure Code, 2nd Ed. Microsoft Press, 2003.
- [6] Pashalidis, A. and Mitchell, C.J. "A taxonomy of single sign-on systems," in R. Safavi-Naini and J. Seberry (editors), Information Security and Privacy - 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11 2003, Proceedings, Springer-Verlag (LNCS 2727), Berlin (2003), pp.249-264.
<http://www.isg.rhul.ac.uk/~xrte/cv/ssotax.pdf>.
- [7] Rubin, A. and Korman, D. "Risks of the Passport Single Signon Protocol." IEEE Computer Networks, July, 2000.
- [8] Oppliger, R. "Microsoft .NET Passport: A Security Analysis." IEEE Computer Society/0018-9162/03/29-35
- [9] Liberty Alliance. "Identity Systems and Liberty Specification Version 1.1 Interoperability." [Technical White Paper], 02/14/2003.
- [10] Internet Engineering Task Force. RFC 1510: The Kerberos Network Authentication Service (V5), September 1993.
- [11] OASIS. "Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1." [OASIS Standard] 09/02/2003.
- [12] Groß, T. "Security Analysis of the SAML Single Sign-on Browser/Artifact Profile." In Proceedings of 19th Annual Computer Security Applications Conference, (2003), pp. 298-307.
- [13] Pfitzmann, B. and Waidner, M. "Analysis of Liberty Single-Sign-on with Enabled Clients." IEEE Internet Computing. Nov/Dec 2003, pp. 38-44.
- [14] Liberty Alliance. "Liberty Bindings and Profiles Specification Version 1.1." (Jan 15, 2003).
- [15] Intenet2.edu. "Shibboleth Project Home Page" <http://shibboleth.internet2.edu/index.html>
- [16] OASIS. "Web Services Security: SOAP Message Security 1.0." (Jan 2004).
- [17] OASIS. "Web Services Security: SAML Token Profile, Working Draft 09." (Jan 2004).
- [18] OASIS. "Web Services Security: SAML Interop 1 Scenarios, Working Draft 04", Jan 29, 2004.
- [19] Liberty Alliance. "Liberty Alliance & WS-Federation: A Comparative Overview." (Oct 14, 2003).
- [20] Kearns, D. "SAML tops federation projects survey." Network World Fusion, Identity Management Newsletter. Online at <http://www.nwfusion.com/newsletters/dir/2004/0119i d1.html> (01/19/04).

