A Brief Introduction to Liberty

Susan Landau and Jeff Hodges

13 February 2003

1 Introduction

For the man on the street, the businesswoman in her office, the shopper or investor at home, identity on the Internet is a straightforward idea with a complex solution. Using Amazon, there is one sign-on and password; using United Airlines, another; connecting to L.L. Bean, yet another, and with Fidelity Investments, a fourth. Within the enterprise, each service — on-line corporate travel, 401(k) account management, employee benefits — may require its own sign-on and password. The same holds for business-to-business interactions. The result is cumbersome, the user experience offputting. The first challenge of Web services is a simple and secure identity mechanism, the second, and equally important, concern is privacy protection.

Single sign-on and federated network identity (a system for binding multiple accounts for a given user) are key to solving this. A federated system allows businesses to manage *their own* resources including customer data. Federated network identity enables consumers to retain some control over which companies have access to their information.

The purpose of Liberty is to develop technical specifications for network identity in a federated environment. It is a project of the *Liberty Alliance,* a group organized to establish open standards for network-based identity interactions.

Developing single-sign-on architecture for federated network identity system in the 2002 Internet environment when the only common Web security tool is SSL (Secure Socket Layer) is technically challenging. Doing so without controlling the most important tool for user interaction — the browser

— makes it even more so.[1]

Liberty's Version 1 architecture is dependent on current browser tools — SSL, Web redirects, and cookies — all of which have limited security functionality. Version 2 can assume smarter clients. But Version 1 must rely on the installed browser base, including Netscape 3.0 and all versions of Internet Explorer, as well as the limited storage of Web browsers on cell phones (which is currently restricted on some to a 256-byte URL).

This document was written as an introduction for those unfamiliar with the Liberty technical specifications. This document briefly describes the technologies on which Liberty is built and explains Liberty's fundamental protocols. Liberty's protocols are complex and this introductory document does not cover them in full detail. Instead, this document attempts to give the reader an overview of Liberty Version 1. The current document has liberally borrowed from a number of documents written by the Liberty team [1], [3], [4], [6], [5].

The next section discusses Liberty technologies. The reader is assumed to have some understanding of network protocols, including TCP/IP. Basic Web protocols will be presented, but since the emphasis of this document is Liberty protocols, discussion will be brief.

2 Web Browser Basics

The Web browser is the fundamental tool for Liberty interactions. A browser is simply a program that allows the user to visit Web sites, read news groups, order items, etc. We begin with a brief overview of HTTP (Hypertext Transfer Protocol), the request-response protocol that defines the World Wide Web.

2.1 HTTP

HTTP was developed by Tim Berners-Lee in the early 1990s as a simple general-purpose way to transfer information over the Internet. It is an "applications-level" protocol, sitting above TCP/IP, the Internet transport mechanism.

[1] As of March 2002, 93% of Web browsers were Internet Explorer [9].

HTTP is implemented in a client program, the browser, and a server program, called simply a Web server. Downloading information, posting information (for example, when filling in an on-line form) is accomplished through a series of HTTP interchanges.

A Web page consists of either a properly structured document in the hypertext markup language (HTML) or one of several type of objects that can be pointed to by an HTML document but cannot themselves point to anything else. These include plain text pages, images (jpeg and gif), sound files (wav, mp3), and more. For the Liberty protocol communications, we are concerned only with the information in the base HTML file.

On the Web, resources are identified by URIs, uniform resource identifiers. A URI can be classified as a locator, a name, or both. The reader is probably more familiar with URLs, or uniform resource locators; a URL is the locator form of a URI and identifies a resource through its main access mechanism. URNs, uniform resource names, are names permanently associated with a resource (for example, much in the way that ISBN numbers function for books). Liberty protocols primarily use URLs.

A URL is a string consisting of three parts: the scheme name denoting the protocol for communicating with the server (typically HTTP, though it can be ftp, telnet, etc.), the name of the server, and the path name within the server of the resource being sought. In the URL:

http://www.projectliberty.org/faq.html

http:	denotes the scheme name;
www.projectliberty.org (with optional ":"port number as in servicedesk.central:8080)	denotes the host;
/faq.html (with possible "?" query)	denotes the path.

2.2 How a Web Connection is Established

Things connected to the Internet are identified by Internet Protocol (IP) addresses, which are 32-bit numbers (usually written in the form of four decimal

numbers, each between 0 and 255, separated by periods, e.g. 152.70.8.105). For use by humans, Internet addresses are given hierarchical names in the Domain Naming System (DNS). It is these names, rather than IP addresses, that are included in URIs. The first step a browser must take is to resolve the URI into an Internet destination. When a user clicks on a URI, the browser sends a query to a DNS (domain name server), which replies with the numeric Internet address of the host named in the URI. The browser then sets up a TCP connection with the host. Once this connection is established, the browser makes an HTTP "Request" and the destination server replies with an HTTP "Response."

The format of an HTTP Request is a request line, followed by optional headers and an optional message body. The request line includes a request method, the URI being requested, and the HTTP protocol version. There are a number of different types of request methods; for Liberty, the important ones are:

- GET: retrieves whatever information is identified by the Request-URI;
- HEAD: similar to GET, except that no message body is retrieved, only a header
- POST: While HEAD and GET retrieve information, POST updates information, e.g., by filling out requested information in a form. Since a POST is pushing information onto the server, users that make a POST request must demonstrate that they have appropriate authority to post the information.

  A POST can also be used to take the body of a request and use it as input to a program identified by the Request-URI in the Request-Line, perhaps a mail handler or a bulletin board manager.

  The function that the POST does is determined by the server and depends on the Request-URI [10].

For the example http://www.projectliberty.org/faq.html, the HTTP Request message above might be:

GET /faq.html/ HTTP/1.1.

(Using a query statement into a database, the GET command can also be used to submit information to a program. There are subtle differences between this use of GET and POST; we will not discuss these here.)

Optional headers may include identification information about the user's browser, e.g., Mozilla 4.0, encoding information about the form of the response, e.g., fr (French), etc. A typical request message might be:

GET /faq.html/ HTTP/1.1
Host: www.projectliberty.org
User-Agent: Mozilla 4.0
CRLF

All browser requests end with CRLF (carriage return, line feed).

The HTTP Response begins with a Status-Line that has three fields: the server's protocol version number, the response code, and a natural language phrase. Response codes include:

- 200 OK
- 204 No Content[2]
- 301 Moved Permanently (in this case a new URI is specified in Location: header of the response message)
- 302 Found (The requested resource temporarily resides under a different URI.)
- 307 Temporary Redirect
- 400 Bad Request (syntax unrecognizable by server)
- 403 Forbidden
- 404 Not Found
- 505 HTTP Version Not Supported[3]

A typical HTTP response might be:

HTTP/1.1 200 OK

As with the HTTP Request, the HTTP Response has optional header lines. These can include the type of connection (whether the server will close the

[2]The response to a POST request is 200 OK or 204 No Content[10].

[3]See [10] p. 39 for a full list of response status codes.

TCP connection after sending the message), the date (time and date of HTTP response), Last-Modified (so that the client browser can know whether to retrieve the requested object from a local cache instead), etc. The following is a typical HTTP response:

```
HTTP/1.1 200 OK
Connection: close
Date: Monday, 6 May 2002 14:17:17 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Tuesday, 30 April 2002 18:03:02 GMT
Content-Length: 5117
Content-Type: text/html
```

3 Liberty Technologies

The Liberty user, who is called the *Principal,* interacts with two types of entities: *Service Providers* (businesses and information providers of the on-line world) and *Identity Providers*: entities that maintain and manage identity information. Single sign-on — the ability of the Principal to authenticate herself once per session with an Identity Provider and then use that authentication to create sessions with various Service Providers and perhaps even other Identity Providers *without having to reauthenticate herself* — is a key feature of Liberty. The Principal needs a simple way to authenticate herself at the Service Provider even if she has not already visited an Identity Provider. This requires a mechanism by which the Service Provider obtains information from the Identity Provider confirming the Principal's identity. This should also work for a Principal who is visiting a Service Provider and who has previously authenticated herself with an Identity Provider and now wants the Service Provider to accept this authentication.

There are several protocols needed. At present, *Authentication* on the Web is normally accomplished by presenting a name and demonstrating knowledge of a corresponding password. This method is also expected to predominate in the first release of Liberty, but Liberty does not limit Identity Providers to a particular scheme. More complex and secure mechanisms involve cryptographic signatures, certificates, or challenge/responses protocols. *Identity Federation* is a way of binding or associating multiple accounts

for a given Principal at various Liberty-enabled entities — Service Providers and (perhaps multiple) Identity Providers).

One way the on-line world differs from the 'real' world is the ease with which a Principal's movements can be traced and a dossier of her Web experiences compiled. For this reason, Liberty supports *Pseudonyms*, the assignment of an arbitrary name by the Identity or Service Provider to identify a Principal. The pseudonym has meaning only in the context of the relationship between the Identity Provider and Service Provider. Thus a Principal's Web experience is harder to follow. Finally, an important part of the Liberty experience is *Single Logout*.

To sign on, a Liberty Principal needs to be directed from the Service Provider to an Identity Provider, and to federate identities, a Liberty Principal also needs to be redirected from a Service Provider to an Identity Provider, or vice versa. To dissolve federating links, the same is true. The Principal could accomplish this by connecting to a new URI, conduct business (Sign-On, perform on-line Identity Federation, etc.), and return. But Liberty seeks "seamless" user-friendly solutions, where the redirection is done not by the Principal, but automatically by the system.

Liberty protocols use Web redirects, in optional combination with SOAP (Simple Object Access Protocol), to exchange SAML-encoded [11] identity information and accomplish Single Sign-On, Identity Federation, etc. Liberty protocols use SSL to encrypt these Web communications and provide user confidentiality.

3.1 Web Redirects

Liberty makes use of the HTTP command 302 "Temporary Redirect," (which was originally designed for a somewhat different purpose.[4] Such 302 Web Redirects work by placing the URI of another location in the Location field of an HTTP Response. The browser receiving such a response is obliged to perform an HTTP GET specifying the URI so conveyed in the HTTP response. This allows Liberty to create a "communications channel" between Identity Providers and Service Providers, as will be explained in detail in the next section.

[4] According to RFC 2616, "a 302 Temporary Redirect means that the requested resource temporarily resides under a different URI."

During a Liberty Web redirect, some private information about the user often travels in the HTTP message. This information needs to be protected. This is accomplished through the use of HTTPS, the secure version of HTTP that uses SSL (Secure Socket Layer) for transporting HTTP messages. A Web Redirect HTTP Response looks like:

HTTP/1.1 302 FOUND
<other headers>
Location: https://< host name and path>?<query>
<other HTTP 1.1 components>

3.2 SSL

SSL encrypts all HTTP communications between the client and server, so that even if an HTTP message is intercepted, the user information is protected. SSL has three main steps: the browser authenticates the server, the browser generates a session key, the server and browser agree all further communications will be encrypted. In slightly more detail, the protocol is given below. For the casual reader of this document, the brief explanation above should be sufficient (more details may be found in [13].)

1. The Principal's browser sends the server the client browser's SSL version number and cryptographic preferences. (SSL supports a choice of cryptographic algorithms.)

2. The server responds with its SSL version number, cryptographic preferences, and public-key certificate.

3. The Principal's browser verifies that the server's certificate is valid. If it is, the browser uses the CA's public key to verify the certificate's signature and determine the server's public key. If the certificate is not valid, what happens next depends upon the browser. The browser may inform the user that a secure and authenticated connection cannot be established; the browser may provide a pop-up window saying, "Expired Certificate; do you want to continue anyway?," etc.

4. The browser generates a session key, encrypts this with the server's public key and sends the encrypted key to the server.

5. This is the key in which all future messages will be encrypted. The browser sends another message saying that the handshake (key establishment) part of the communications is over; this message is encrypted.

6. The server sends a message to the browser saying that all future messages will be encrypted. The server also sends a message saying that the handshake part of the negotiation is now over; this message is also encrypted.

7. The session key has been established and all future communications between the client and server are encrypted.

SSL gives a choice of cryptoalgorithms for the computation, including RSA for the key exchange, and RC4, DES, Triple-DES, and — in new browsers — AES for encryption.

3.3 Cookies in Liberty and the Introduction Problem

HTTP is a stateless protocol; by itself it does not enable the maintenance of state information about or on behalf of the user. Cookies are a way to get around this state of statelessness and enable the server to store information, e.g., about users. State information is stored at the client and is then sent to the server the next time the user accesses that server. A cookie is an HTTP header written by the server to the browser's memory. Put another way, a cookie is a text string consisting of domain, path, lifetime, and value (a variable the website sets) that can save information during an HTTP session or between sessions.

If the cookie's lifetime has not elapsed when the browser is shut off, the cookie is written to a file on the browser's hard drive. In that case, we call it a *persistent* cookie. Each time the browser is turned on, the cookie file is read from disk and each time the browser is shut off, the cookie file is written to disk. Cookies are deleted when their lifetimes expire *or* when the cookie cache is full. (Netscape, for example, only allows 300 cookies in the cookie file.) In that case, the oldest cookies are deleted first.

Browsers typically give users a choice about how they can handle cookies, e.g.:

- Accept all cookies

- Only accept cookies originating from the same server as the page being viewed[5]
- Do not accept or send cookies
- Warn me before accepting a cookie [Netscape 4.76]

Liberty designers cannot assume that "Accept all cookies" is enabled by default in the installed browser base. Furthermore, "Accept all cookies" does not allow the user control over her system and is a poor security choice. So cookies cannot be used to satisfy the cross-DNS-domain information sharing needed in Liberty. But cookies can be and are used in Liberty.

Liberty Trust Circles (also called Circles of Trust) are federations of Identity and Service Providers engaged in on-line interactions based on Liberty-enabled technologies. These companies have agreed on how they will verify on-line identities.

Suppose there is more than one Identity Provider in such a Trust Circle. Providers will need a way to see which Identity Providers a user uses, and this must work across DNS domains. This is known as the "Introduction Problem." An optional Liberty approach is to create a *common domain* for the Trust Circle that will be accessible to all parties in the Trust Circle. For example, if the Identity Provider is Airline.inc, the Service Provider is CarRental.inc, and the Trust Group is AAG.inc (Airline Affinity Group.inc), then the DNS domain will be XXX.AAG.inc. That is, the Identity Provider will be Airline.AAG.inc, the Service Provider will be CarRental.AAG.inc, etc. ([1], pp. 8-13.)[6]

[5] This means only accept cookies with the same address as the page being viewed.

[6] The common domain cookie enables the Service Provider to discover with which Identity Providers the Principal has recently interacted. So the Service Provider has to obtain the cookie. It does so via the Web redirection trick.

The Service Provider sends an HTTP Redirect to the Principal, with the location header set to the URI hosting the cookie transfer service in the common domain. The transfer service URI must specify HTTPS as the transport scheme. The URI in the location header includes another URI embedded in the query parameter which the cookie transfer service will use to redirect the response back to the Service Provider.

The Principal sends an HTTPS GET to the Common Domain. The Commom Domain cookie transfer service responds to the Principal with an HTTP Redirect to the Service Provider. The location header is the return URI for the Service Provider. Again, HTTPS is specified for the transmission. The cookie value is included in the query component of

When a Principal authenticates herself with a particular Identity Provider, the Identity Provider will redirect the Principal's browser to the Identity Provider's common domain service with a parameter indicating that the Principal is using that Identity Provider[7]. The common domain service writes a cookie with that preference and redirects back to the Identity Provider. Thereafter, Service Providers and other Identity Providers in the Trust Circle will be able to tell which Identity Provider the Principal used.

As noted, this common domain cookies approach is optional. Liberty implementations may use other methods, e.g., simple hand configuration to list "known" Identity Providers, thus also solving the Introduction Problem.

3.4 Exchanging Identity Information: SOAP, SAML, and Web Redirects

Liberty protocols exchange identity information through pre-existing protocols and languages, SOAP and SAML, respectively, as well as Web redirects. SOAP, Simple Object Access Protocol, is a peer-to-peer protocol for exchanging structured and typed information between peers in a distributed environment. SAML, Security Assertion Markup Language, is an XML-based security assertion framework [12].

The SOAP envelope is a framework for expressing what is in a message, who should handle it, and whether it is optional or mandatory. SOAP also has encoding rules for exchanging application-defined datatypes, exactly what is needed in Liberty. SOAP also has Remote Procedure Calls (RPCs), which are used in Liberty.

SAML defines three types of assertions: authentication, attribute, and authorization decision. Liberty uses authentication assertions, which state that subject S was authenticated at time T by means M. Note: SAML does not manage credentials; it only reports on them.

In summary, SAML authentication assertions are conveyed by either SOAP or Web Redirects in order to accomplish exchange of identity and authentication information.

---

the return URI; it is the list of Identity Providers. The Principal's browser redirects the HTTPS to the Service Provider, which reads the cookie to determine the Identity Provider for the Principal. (This discussion is taken from [1], pp. 35-39.)

[7]This interaction is specified as the "Identity Provider Introduction" Profile in [3], p. 45.

4 Liberty Version 1 Protocols

4.1 The Liberty "Actors"

Liberty posits three actors: Principals, Service Providers, and Identity Providers:

A Principal is a user.

A Service Provider is an organization offering Web-based services to Principals. It can be car rental or airline, it can be a seller of widgets for just-in-time production, an on-line *New York Times* subscription, government agencies providing information, financial services , etc.

An Identity Provider is a Service Provider offering additional services and incentives so that other Service Providers affiliate with the Identity Provider and Principals choose to use the Service Provider as their Identity Provider. The services may be financial, they may include connections with other types of Web services (e.g., calendar or address-book related features), they may be a corporate identity authority that enables access to various company Service Providers (e.g., an on-line travel agency, an outsourced benefits office), etc. The important service an Identity Provider gives is authentication of the Principal.

4.2 Requirements Summary

Liberty seeks to build a mechanism for federated identity. Key to Liberty is interoperability across a wide range of platforms. The Engineering Requirements Document states that the technology must provide the "widest possible support for [different] operating systems, [different] programming languages, [and different] network infrastructures" ([4], p. 5). Thus Liberty technologies:

- should enable individuals (customers and businesses) to *securely* control their *personal information*;
- should be *open standards* for single sign-on allowing *decentralized authentication*;
- should be *open standards* for network identity spanning *all* types of network devices; and

- should work internationally ([4], p.4).

Liberty Version 1 supports four main functionalities: Authentication, Pseudonym, Identity Federation, and Single Logout. Liberty's broad principles translate into the following recommendations for Liberty-enabled implementations:

- Identity Federation
  - The Principal is informed about Identity Federation and Federation Termination (this is also called Defederation).
  - Service Providers and Identity Providers inform each other about identity defederation.
  - Identity Providers notify appropriate Service Providers about account termination at the Identity Provider.
  - Service Providers and Identity Providers provide the Principal with a list of federated identities at the Service Provider/Identity Provider in question. ([4] pp. 5-6)
- Authentication
  - Liberty-enabled Identity Providers and Service Providers will support all methods of navigation including, but not limited to, clickthroughs, bookmarks/favorites, URIs.
  - The Identity Providers provide the Principal with an authenticated identity prior to the Principal presenting any credentials or personally identifiable information.
  - Identity Providers and Service Providers will mutually authenticate each other; both Identity Providers and Service Providers ensure confidentiality, integrity, and authenticity of the information they exchange about Principals.
  - Identity Providers and Service Providers will support various types of authentication mechanisms.
  - To preserve Principals' privacy, Identity Providers and Service Providers will exchange a minimal set of information about each Principal: authentication status, instant (the time at which authentication occurred), method, pseudonym. ([4], pp.6-8)

- Pseudonym
  - To preserve Principal anonymity, Identity Providers and Service Providers will support unique pseudonyms on a per Identity Federation basis across all Identity Providers and Service Providers. Thus a Principal might be Scott00003 to the Identity Provider, and ScottMcN to the Service Provider. ([4], p. 8)
- Single Logout
  - When a Principal logs out of an Identity Provider, all appropriate Service Providers will be notified ([4], p. 9).

These requirements have more detailed instantiations in the Engineering Requirements Document [4].

4.3 Liberty Protocols

Liberty Version 1 defines four protocols:

- Single Sign-On and Federation
- Name Registration
- Federation Termination Notification
- Single Logout [6].

The first is described in detail below. Name Registration is an optional protocol and is not further described here. Federation Termination and Single Logout are briefly described below.

Each protocol may be may be concretely implemented in more than one way. Each way is called a "profile" of the protocol. In order to cover the widest possible range of possible users[8] four "profiles" of this protocol are defined: Liberty Browser Artifact[9] (using GET), Liberty Browser POST,

[8]These include web browsers, mobile handsets, and other configurations (e.g., those involving proxies),

[9]This is through a SAML artifact, which is a small random number designed to point to SAML assertions. The SAML artifact is passed between sites by the browser via the URL query strings.

Liberty WML POST (WML is the markup language typical of mobile handsets), and Liberty Enabled Client/Proxy. The differences in the profiles are minimal.[10] This document seeks to give a clear but concise explanation of Liberty. It cannot include all cases and all details. So with the exception of the Identity Federation protocol, where all variations are shown, the protocols are presented only in their most typical fashion using HTTP GETs.

Single Sign-On and Federation

Single Sign-On and Federation is the most complex of the Liberty protocols. We will first describe the protocol from the Principal's vantage point, then sketch in moderate detail what the protocol is actually doing, and conclude with some of the technical details. This discussion appears with greater precision in ([3], pp. 12-15).

Assume the Principal is at a Service Provider. For simplicity, assume that the Principal has already opened an account at a Liberty-enabled Identity Provider but has not necessarily authenticated herself to the Identity Provider during the current Web session. Furthermore, assume all HTTP Requests are HTTP GETs (other profiles of this protocol are illustrated in the figure on page 19).

The Principal is at a Service Provider and she decides she wants a service, perhaps updating her benefits selection with an outsourced on-line benefits office, perhaps ordering widgets needed from another site in just-in-time factory production. She needs some means to inform the Service Provider that she wants to login with her Identity Provider. There may be a button to an affinity group site on the Service Provider's Web page, or the Service Provider may have some Liberty-enabled Identity Providers listed, or the Service Provider may have a separate log-in page. Whatever the mechanism, the Principal says "I want to authenticate myself." The Service Provider sends the Principal's browser over to the appropriate Identity Provider, who produces a credential (there will be more detail later on the exact form of the credential) for the Service Provider upon Principal login, and redirects the Principal's browser back to the Service Provider. Although complex in detail, this set of actions produce a seamless experience for the Principal.

[10] The overview for these protocols may be found in the *Liberty Architecture Overview*. For detailed instructions, the reader should see the *Liberty Bindings and Profiles Specifications*.

Sketching the Single Sign-On and Federation protocol in more detail, we have it beginning with the Principal's browser sending an HTTP GET to the Service Provider. The Service Provider's website responds with an HTTP 302 Redirect containing two important URIs. The first is the URI to which the Principal's browser should now redirect the original GET command; the other is an embedded URI that refers back to the Service Provider.

Following the redirect response, the Principal's browser sends an HTTP request to the Identity Provider specifying the URI from the previous response, including the embedded Service Provider's URI. The Identity Provider responds to the Principal with a credential (more on this later) via an HTTP Redirect pointing back to the Service Provider; this information has been extracted from the previously embedded URI that the Service Provider had supplied. The Principal's browser sends an HTTP Request to the Service Provider specifying the URI taken from the location field of the response returned from the Identity Provider. Note that this URI may itself contain an embedded URI pointing back to the Identity Provider.

The Service Provider now has a form of credential for the Principal, either an assertion or an artifact. If it is an assertion, the Service Provider will send an HTTP Request with the artifact to the Identity Provider who will respond with an assertion (or not). The Service Provider processes the assertion, and sends an HTTP Response to the Principal, completing the original HTTP request that began this sketch.

Restating the above protocol in more detail.

1. At the Service Provider website, the Principal selects an Identity Provider for her identity authentication (or the Service Provider selects one for her based on whatever approach is being used for introduction; see Section 3.3). She may press a "button" at the Service Provider's website to select an Identity Provider, she may fill in a form at the Service Provider's website (or she may be requesting some resource and the Service Provider determines the Identity Provider from context (see Section 3.3), the mechanism that is used to transmit this information is an HTTP Request submitted by the Principal's browser.

2. The Service Provider determines the address of the Identity Provider and concocts an alternate URI pointing at the Identity Provider.

3. The Service Provider responds to the Principal's browser with an HTTP Response 302 ("Redirect") and an alternate URI in the location header field. The alternate URI field has a second, embedded URI pointing back to the Service Provider.

   Now the Liberty protocol varies, depending on whether it is based on GETs, POSTs, or WML browsers. What will be explained here and onwards is the most common scenario, HTTP GETs.

4. The Principal's browser performs an HTTP GET Request to the Identity Provider with the URI from the location field received in Step 3 as the argument of the Request.

5. The Identity Provider processes the Principal's HTTP GET Request. If the Principal has not yet been authenticated by the Identity Provider, authentication occurs now.

6. The Identity Provider responds with an authentication assertion, a SAML artifact, or an error. The response is conveyed using HTTP Redirect 302 whose location header contains the URI pointing to the Service Provider (this has been extracted from the GET argument URI of Step 4).

7. The Principal's browser obtains the artifact or assertion.

   The Principal sends an HTTP GET Request to the Service Provider using the complete URI taken from location field of the response received in Step 6.

8. This step occurs only if the Step 6 response was a SAML artifact. (Recall that a SAML artifact is a small random number designed to point to SAML assertions. It is passed between sites using Web redirects and embedding it in URL query strings.)

   The Service Provider now needs to get an authentication assertion corresponding to the artifact it received. The Service Provider sends a SOAP message to the Identity Provider, requesting the assertion.

9. This step occurs only if the Step 6 response was a SAML artifact.

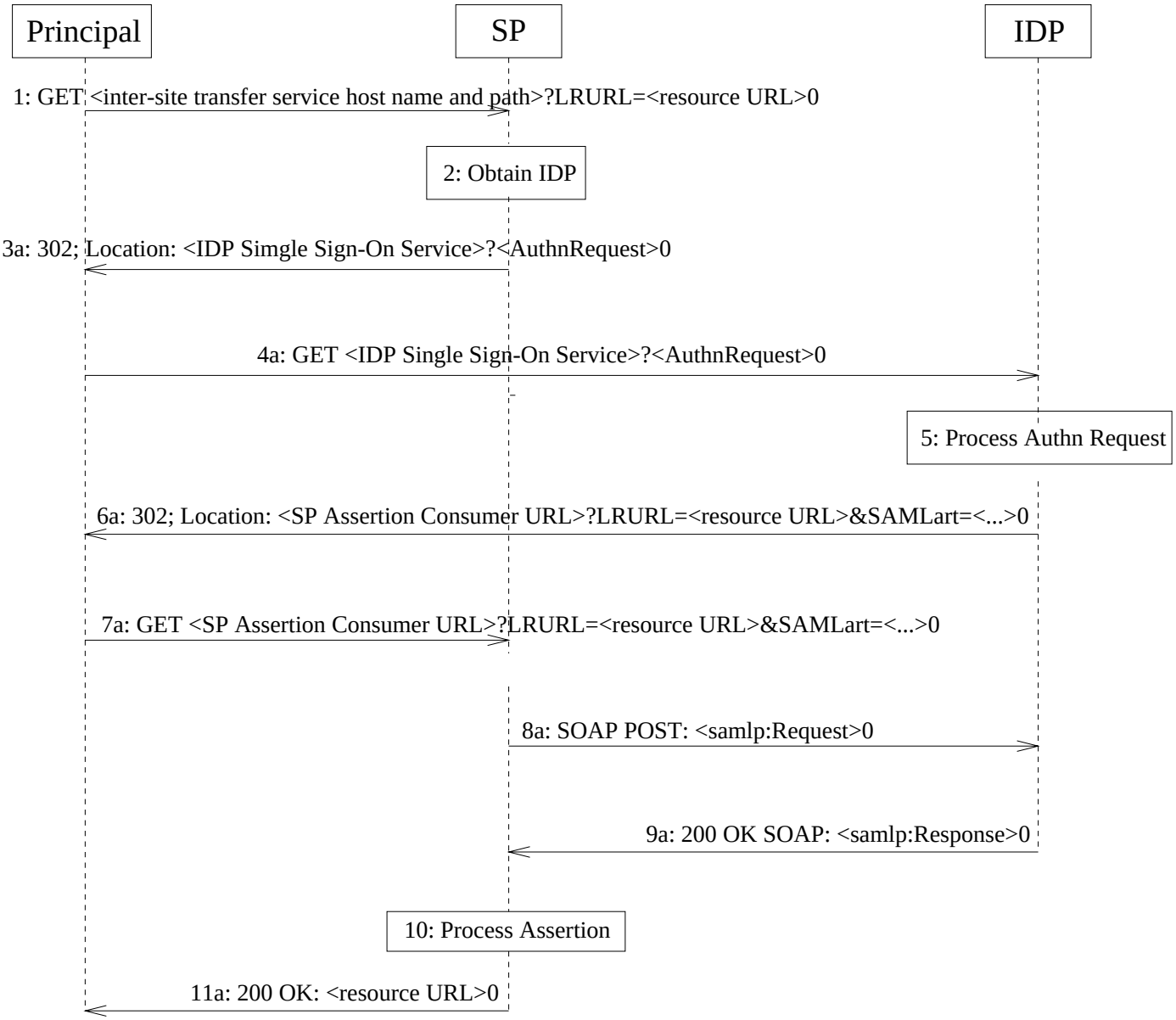
   The Identity Provider processes the request, and responds with the corresponding assertion.

10. The Service Provider sends an HTTP Response completing the Principal's original HTTP request in Step 1.

(A more detailed discussion can be found at [3], pp. 12-15, from which this discussion is excerpted.)

The figure below shows the set of actions described above, along with variations for (b) Single Sign-on Browser POST, (c) Single Sign-on WML POST, and (d) Single Sign-on Liberty Enabled Client/Proxy.[11]

[11] Profiles (a), (b), (c), and (d) are, respectively, figures 2, 3, 4, and 5 in [3].



3b: 302 Location < IDP Single Signon>?<Auth Request>
3c: 200 OK w/ WML POST <IDP: Single Signon Service: LAREQ= AuthnRequest>
3d: 200 OK < AuthnRequest Envelope>; Liberty-Enabled Header
4b: GET <IDP Single Signon >?<AuthnRequest>
4c: POST <IDP Single Signon Service>?LAREQ=AuthnRequest

4d: SOAP POST <AuthnRequest>; Liberty-Enabled Header()

6b: 200; FORM; METHOD=POST;ACTION=<SP assertion consumer URL:LARES=<AuthnResponse>

6c: POST <SP Assertion Consumer URL: LARES=AuthnResponse>

6d: 200 OK SOAP: <SP AuthnResponseEnvelope>;Liberty-Enabled Header()

7b: POST <SP assertion consumer URL; LARES=<AuthnResponse>

7c: POST <SP Assertion Consumer URL>: LARES=<AuthnResponse>

7d: POST <SP Assertion Consumer URL>: LARES=<AuthnResponse>Liberty-Enabled Header

8b:, 8c:, 8d:, 9b:, 9c:, 9d: —

11b: HTTP Response; Liberty-Enabled Header

11c: HTTP Response; Liberty-Enabled Header

11d: HTTP Response; Liberty-Enabled Header[12]

4.4 Federation Termination

Just as Principals can federate their identities, they also need to be able to perform a federation revocation; this is also called *Federation Termination* or *defederation*. Federation Termination can be initiated at an Identity Provider or a Service Provider.

We begin with the case where the Principal initiates the request at an Identity Provider.

1. The Principal sends an HTTP Request to the Identity Provider specifying the Service Provider with whom Federation Termination should occur.

2. The Identity Provider's Federation Termination service responds with an HTTP Redirect to the Service Provider. The location header field is set to the Service Provider's Federation Termination URI, which must specify HTTPS as the URL scheme.

[12] The exact form of the HTTP response in this final step will vary depending upon how the overall interaction began. If the Principal clicked on an explicit "Login" link, then this response is likely to be a "200 OK." If the login interaction was begun because the Principal had selected a protected resource, it may be a 200 OK or some error depending on whether the Principal successfully authenticated herself and is authorized to access the resource.

3. The Principal sends an HTTPS Request to the Service Provider with the FederationTerminationNotification information attached to the URI of the HTTP GET.

4. The Service Provider processes the request.

5. The Service Provider's Federation Termination sends an HTTP Redirect to the Principal with the location header URI being the Identity Provider.

6. The Principal sends an HTTP Get to the Identity Provider.

7. The Identity Provider responds with an HTTP 200 OK and a textual message, confirming that it has requested terminating the identity federation with the Service Provider.

The case where the request is initiated at the Service Provider is handled as above, with the roles of the Identity Provider and Service Provider switched. (For a more detailed discussion of Federation Termination, see [3], pp. 28-34, from which this discussion is excerpted.)

4.5 Pseudonyms

Liberty, with its system of multiple Identity and Service Providers needs a mechanism by which Principals can be uniquely identified across multiple domains. At the same time, Principal privacy demands that a Principal *not* be required to have a single identifier across all domains.

Liberty solves this problem by having each member of a Trust Circle create a "handle" for each Principal. Members can create multiple handles for each Principal, but each Identity Provider must have a single handle for each (Principal, Service Provider) pair, even if the Service Provider maintains multiple websites.

The Identity Provider and Service Provider in a federation need to remember each other's handle for the Principal, so they create entries in their Principal directories for each other and record each other's handle for the Principal.

Since identities are verified across a chain (there is no "transitive" trust in Version 1 Liberty), there is no value in a "universal" identity.

4.6 Single Logout

Like Federation Termination, Single Logout can be initiated at either the Identity Provider or Service Provider. In either case, the Identity Provider then communicates a logout notification to each Service Provider with which it has established a session for the Principal. A more detailed discussion can be found at [3], pp. 34-45, from which this discussion is excerpted.

We will begin with the Principal initiating a logout request at the Identity Provider.

1. The Principal signals she wants to log out; this is done via an HTTP GET to the single logout service URI at the Identity Provider.

2. For each Service Provider for which Identity Provider has provided an authentication assertion during the Principal's current session, the Identity Provider's single logout service sends an HTTP Redirect to the Principal. For each redirect, the location header is set to the Service Provider's single logout service URI; the location header contains a query component (LogoutNotification).

3. For each of these Service Providers, the Principal fulfills the redirect.

4. Each Service Provider processes the redirect.

5. Each Service Provider's single logout service responds and sends the Principal an HTTP redirect with location header being the Identity Provider (obtained, as per usual, from the location header information provided by the HTTP Redirect that came from the Identity Provider).

6. The Principal's browser fulfills this redirect request. The Identity Provider repeats the process beginning with Step 2 if there are more Service Providers to contact, otherwise it proceeds to Step 7.

7. The Identity Provider replies with a HTTP 200 OK message ([3], pp. 35-41).

If the Principal initiates the single logout notification at a Service Provider instead, then the process is only slightly more complicated. The Principal sends an HTTP Request to the Service Provider indicating the session logout

at the associated Identity Provider. The Service Provider responds with an HTTP Redirect to the Identity Provider with the location head URI set to the Identity Provider's single logout service and things proceed — essentially — as above ([3], pp. 41-45). (For a more detailed discussion of Single Logout, see [3], pp. 34-45, from which this discussion is excerpted.)

5 Liberty Security

The intent of the Liberty Version 1 specifications is to enable single sign-on at multiple sites substantially as secure as giving a name and password at each site. We believe Liberty has succeeded in these goals. But because Liberty Version 1 is built on top of present-day Internet technlogy, some security issues remain. The following is based on [8], which discusses these issues in greater detail.

5.1 Risks of Single Sign-on

We are not aware of any particular vulnerabilities in the Liberty Version 1 Single Sign-on protocols per se. However, these protocols exacerbate well-known Internet insecurities, including:

- Risks of Weak Passwords

  "Reusable" passwords — a password that is used multiple times to gain access to a site — are a known vulnerability, as are weak, guessable passwords. Such poorly-choosen passwords can be particularly problematic in a single sign-on environment.

  If a principal uses strong and unrelated passwords at multiple sites, she has better security than in the single sign-on situation. But if she does not, then the vulnerability of multiple sites is essentially the same regardless of whether single sign-on is used.

  Note Liberty Version 1 does not specify any particular authentication technique. Identity providers can choose to have stronger forms of identification.

- Risks of Embedded Login Forms

In the interests of providing a seamless login service, one option is for a Service Provider to offer an Identity-Provider embedded form for authentication. While there is security risk in this, the embedded form mechanism does run the risk of teaching the user to do the wrong thing.

The principal is potentially revealing her authentication credentials to the service provider in clear text. If authentication is to be done via embedded forms, contract terms between Identity Providers and Service Providers regarding privacy and data ownership should be clear.

- Risks of Internet Deployment

  If a principal accesses a Liberty-enabled browser at a public site (e.g., an airport kiosk), and the browser subsequently hangs before the user has completed her session, there is a danger that a rogue may take over the legitimate user's session. This is, of course, a general problem, not a Liberty-specific one.

  The Security and Privacy Issues document [8] recommends that Liberty-based identity servcies have a mechanism that enable the user to later return, via a different browser session, and kill off the old session. To prevent session hijacking, [8] recommends short session times for Identity Providers.

- Risks of Weak Cryptography

  The privacy and security protection of the principal's data is principally accomplished through the use of SSL/TLS. One of the most vexing issues in securing web services is that the currently-installed browser base includes many browsers that only have weak 40-bit cryptography enabled. This poses a serious privacy and security risk since the principal's encrypted communications can be easily recovered to its unencrypted form.

  The solution, clearly, is that the cipher suites have minimum effective key size of 112-key bits. The Liberty specifications recommend this.

The real lesson is that principals, Identity Providers, and Service Providers should practice due diligence.

5.2 Liberty Security Requirements

As the *New Yorker* cartoon says, "On the Internet, no one knows you're a dog." That presents the most serious type of threat to Liberty interactions. How does a Principal, Service Provider, or Internet Provider know that the party on the other end is who they saw they are?

Liberty specifications attempt to address this issue.

- Service Providers are required to authenticate Identity Providers using Identity Provider Server-side Certificates.
- Each Service Provider is required to be configured with a list of authorized Identity Providers; each Identity Provider is required to be configured with with a list of authorized Service Providers.
- The authenticated Identity of an Identity Provider must be displayed to a Principal before the Principal presents personal authentication data to that Identity Provider.
- Liberty protocol messages and some of their components are generally required to be digitally signed and verified.
- In interactions between Service Providers and Identity Providers, requests are to be protected against replay attacks. (A replay attack is one in which challenges and responses are recorded and then the responses are used again later by an impersonator.)

(These are excerpted from [1] pp. 16-17.)

There are also communication and message security requirements in Version 1 Liberty([1], p. 16):

Security Mechanism	**Channel Security**	**Message Security (for Requests, Assertions)**
Confidentiality	Required	Required
Per-message data integrity	Required	Required
Transaction Integrity	—	Required
Peer-entity authentication	Identity Provider — Required Service Provider — Optional	—
Data Origin Authentication	—	Required
Nonrepudiation	—	Required

Nonetheless, spoofing attacks and other types of attacks are possible.

5.3 Threat Scenarios

The Version 1 Liberty protocols subscribe to a typical Internet threat model, which assumes that the intermediary and end-systems participating in Liberty protocol exchanges have not been compromised. This is a rather strong assumption. The Version 1 protocols include some protections against such compromise, but the Version 1 protocols are not fully hardened. This is a design decision. It really is impossible to have any other choice if Liberty is to be launched now on the currently-installed browser base. For more information on the threat scenarios considered and the applicable countermeasures, see [3].

Liberty designers have classified attacks by:

- Type of Actors:
  - Rogue Entities: Rogue entities are those that misuse their privileges. Rogue actors may be Principals, Identity Providers, or Service Providers. They attempt to use their relationship to escalate privileges or masquerade as another entity.
  - Spurious Entities: Spurious entities are those who masquerade as a legitimate entity or are completely unknown to the system.
- Type of Attack:
  - Passive: The adversary monitors the communication channel.
  - Active: The adversary may transmit messages to obtain information.
- Collusion: Two or more entities may collude to launch an attack.
- Denial-of-Service Attacks: The prevention of access to a resource caused by overloading the resource with spurious requests.

Through the use of SSL and federation, Version 1 Liberty protocols protect Principals' privacy. The protocols are designed to prevent:

- replay attacks, in which an entity captures an authentication assertion and attempts to "replay" it and thus impersonate a principal;
- denial-of-service attacks, through signing of authentication requests (this is a partial solution only);
- collusion between two principals, through matching of IP address with the AuthenticationLocality of the SAML artifact (this is only a partial solution, since it may be defeated by source IP address spoofing).

There are no protections against a rogue Service Provider impersonating a Principal at another Service Provider, or a rogue Identity Provider, nor against Service Providers colluding to violate the privacy of the Principal. In a sense, these are out-of-band privacy and security issues.

For more detail on these issues, see [3] pp. 51-55., from which this discussion has been excerpted.

6 Conclusion

Liberty Version 1 was designed to simplify the process of signing on to multiple sites while still preserving users' privacy and security. The design decision was that this should be done building on top of the currently-installed user base. As this document and the *Security and Privacy Concerns in Liberty Alliance Version 1.0 Specifications* [8] make clear, in the Internet environment, security is fragile, and implementors — and users — must exercise due diligence. As mentioned earlier, the intent was that single sign-on should be substantially as secure as giving a name and password at each site. We believe that Liberty Version 1 has met those goals.

Acknowledgements: As noted in the text, we have extensively used material from the Liberty documents below. We benefitted from comments from Gary Ellison, Whitfield Diffie, and Seth Proctor. Any remaining errors are our responsibility.

References

[1] J. Hodges (ed.), *Liberty Architecture Overview*, Version 1.1, 15 January 2003 http://www.projectliberty.org/specs/liberty-architecture-overview-v1.1.pdf

[2] L. Kannappan, M. Lachance, and J. Kemp (ed.), *Liberty Architecture Implementation Guidelines,* Version 1.1, 15 January 2003, http://www.projectliberty.org/specs/liberty-architecture-implementation-guidelines.v1.1.pdf

[3] J. Roualt and T. Wason (ed.), *Liberty Bindings and Profiles Specification,* Version 1.1, 15 January 2003, http://www.projectliberty.org/specs/liberty-architecture-bindings-and-profiles-v1.1.pdf

[4] S. Allavarpu (ed.), *Liberty Alliance Engineering Requirements Document,* Version 0.2, 9 March 2002.

[5] H. Mauldin and T. Wason (ed.), *Liberty Architecture Glossary,* Version 1.1, 15 January 2003, http://www.projectliberty.org/specs/liberty-architecture-tech-glossary-v1.1.pdf

[6] J. Beatty and J. Kemp (ed.), *Liberty Protocols and Schemas Specification,* Version 1.1, 15 January 2003, http://www.projectliberty.org/specs/liberty-architecture-protocols-schemas-v1.1.pdf

[7] *Project Liberty: Marketing Requirements Document: Phase I — Authentication Sharing*, 17 April 2002.

[8] G. Ellison, J. Hodges, and S. Landau, *Security and Privacy Concerns in Liberty Alliance Version 1.1 Specifications,* 12 February 2003.

[9] S. Pruitt, "Microsoft's Newest Browser Has Gained Significant Market Share, so What Will Become of its Closest Rival?," *PC World,* 27 March 2002, http://www.pcworld.com/news/article/0,aid,91483,00.asp

[10] RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999.

[11] RFC2396, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998.

[12] SAML v. 1.0, Specification Set, 31 May 2002, http://www.oasis-open.org/committees/security/♯documents

[13] A.Frier, P. Karlton, and P. Kocher, "The SSL Protocol Version 3.0," ftp.netscape.com/pub/review/ssl-spec.tar.Z, 4 March 1996.

Glossary

DNS: The Internet Directory System, a distributed database implemented in a hierarchy of name servers which translate the mnemonic URI hostnames to Internet addresses

Circle of Trust: A group of Service and Identity Providers that have business relationships built on top of Liberty-enabled technology and guidelines.[5]

client: program that establishes a connection for the purpose of sending requests, e.g., a browser [10].

federated network identity: a system for binding multiple accounts for a given user.

network identity : the global set of attributes composed from an individual's various account(s) [1].

About the Authors

Susan Landau is Senior Staff Engineer at Sun Microsystems Laboratories. Before joining Sun, she was a faculty member at the University of Massachusetts and Wesleyan University, and held visiting positions at Yale, Cornell, and the Mathematical Sciences Research Institute at Berkeley. She and Whitfield Diffie have written "Privacy on the Line: The Politics of Wiretapping and Encryption," which won 1998 Donald McGannon Communication Policy Research Award, and the 1999 IEEE-USA Award for Distinguished Literary Contributions Furthering Public Understanding of the Profession. Landau is also primary author of the 1994 Association for Computing Machinery report "Codes, Keys, and Conflicts: Issues in US Crypto Policy." Before becoming involved in policy, Landau had worked in symbolic computation and algebraic algorithms, discovering several polynomial-time algorithms for problems that previously only had exponential-time solutions. She is a Fellow of the American Association for the Advancement of Science. Landau is a member of the National Institute of Standards and Technology's Computer System Security and Privacy Advisory Board, as well as a member of the Association for Computing Machinery's Advisory Committee on Privacy and Security, and ACM's Committee on Law and Computing Technology. She has appeared on NPR several times, and has had articles published in the "Chicago Tribune," the "Christian Science Monitor," "Scientific American," as well as numerous scientific journals.

Jeff Hodges is an Architect at Sun Microsystems, working in the areas of identity solutions, distributed infrastructure, and security. His interests lie in the nature of "network identity" and its realization via composition of authentication, security, and directory technologies. He is a participant in the Liberty Alliance effort and editor of the Liberty Version 1.0 Architecture Overview. He served as co-chair of the OASIS Security Services Technical Committee, shepherding and contributing to the work on Security Assertion Markup Language (SAML). His past work has included contributions to the design of the LDAPv3 directory access protocol – specifically, co-authoring of RFCs 2829 and 2830. Additionally, he contributed to the design and deployment of Stanford University's SUNet ID and Registry/Directory infrastructure. Prior to joining Sun, he's held engineering positions at Oblix, Inc., Stanford University, and Xerox.