SAML Implementation Guidelines

Working Draft 01, 27 August 2004

Document identifier:
sstc-saml-implementation-guidelines-draft-01

Location:
http://www.oasis-open.org

Editor:
Charles Knouse (cknouse@oblix.com)

Contributors:
Liberty ID-FF Implementation Guideline contributors

Abstract:
This non-normative specification provides guidelines for the implementation of applications using SAML assertions, protocol, bindings, and profiles.

Status:
This is a working draft produced by the Security Services Technical Committee. Publication of this draft does not imply TC endorsement. This is an active working draft that may be updated, replaced or obsoleted at any time. See the revision history for details of changes made in this revision.

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them to the security-services-comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to security-services-comment-request@lists.oasis-open.org with "subscribe" in the body) or use other OASIS-supported means of submitting comments. The committee will publish vetted errata on the Security Services TC web page (http://www.oasis-open.org/committees/security/).

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the Security Services TC (http://www.oasis-open.org/committees/security/ipr.php).

Table of Contents

1 Introduction....4
1.1 Recommended Knowledge....4
1.2 SAML Architecture....4
1.3 SAML Environments....5
2 User Agent Considerations....6
2.1 Cookies....6
2.1.1 Constraints....6
2.1.2 Privacy and Security Considerations....7
2.1.3 URL Encoding....7
2.2 URL Length Considerations....7
2.3 Use of JavaScript/ECMAScript....7
2.4 Caching....8
3 Security Considerations....9
3.1 Channel Security....10
3.1.1 SSL and TLS....10
3.1.2 HTTP Basic and Digest Authentication....11
3.1.3 IPsec and VPNs....12
3.2 Message Security....12
3.2.1 XML Signature....12
3.2.2 XML Encryption....14
3.2.3 WS-Security....14
3.3 Trust Guidelines....15
3.3.1 Trusting Certificate Authorities....15
3.3.2 Trusting Self-Signed Certificates....15
3.3.3 Evaluating Certificates....15
4 Authentication Mechanisms....16
4.1 Authentication Mechanisms are Orthogonal to Single Sign-On....16
4.2 Identity Provider Session State Maintenance....16
4.3 Credentials....16
4.4 Authentication Type, Multi-tiered Authentication....17
4.5 Mutual Authentication....18
4.6 Validating Liveness....18
4.7 Communication Security....18
4.8 Compromised Credentials....18

5Replication Considerations....19
5.1 URL Endpoints....19
5.2 Keys....19
5.3 Artifacts....19
5.4 Received BPP Responses....19
5.5 User Identities and Aliases....19
5.6 Session State....20
6 Guidelines for Mobile Environments....21
6.1 Some Mobile-specific Deployment Considerations....21
6.2 Using the SAML SSO Profiles in Mobile Environments....21
6.2.1 Summary of the SAML Profiles in Mobile Environments....21
6.2.2 Methods of Roaming Using the SAML Protocols....22
6.2.3 WAP and the Enhanced Proxy....23
6.3 Mobile Provisioning Using Tamper-resistant Smart Cards....25
6.3.1 Obtaining EC-Provisioning Document Instances Using a (U)SIM....27
6.3.2 Updating EC-Provisioning Document Instances Using a (U)SIM....28
6.3.3 External standardization dependencies....29
6.4 SAML Authentication Context in Mobile Environments....29
7 Privacy Principles....31
7.1 Privacy Principles for Authentication....31
7.2 Privacy Principles for Federated Authentication....32
7.2.1 Passive SSO....33
7.2.2 Authentication Status....34
7.2.3 Authentication Strength....34
7.2.4 IDP Proxying....34
8 References....36

1 Introduction

This non-normative document provides implementers and deployers guidance on the usage of the SAML (Security Assertion Markup Language) specifications. Although the SAML specifications provide a basis for interoperability between implementations, implementers and deployers will make choices for authentication methods, session management and other components of a SAML implementation. Some of these choices will be dictated by the environment within which a particular implementation may operate (for example, mobile network infrastructure, or an enterprise software environment).

1.1 Recommended Knowledge

The reader is assumed to be familiar with the SAML specification family:

- SAML Assertions and Protocol ("Core") [SAMLCore]
- SAML Bindings [SAMLBind]
- SAML Profiles [SAMLProf]
- SAML Authentication Context [SAMLAuthn]
- SAML Metadata [SAMLMetadata]
- SAML Conformance Requirements [SAMLConform]
- SAML Security and Privacy Considerations [SAMLSec]

The reader should also be familiar with technologies used by SAML:

- XML [XML]
- XML Signature [XMLSig]
- XML Encryption [XMLEnc]
- SOAP [SOAP]
- SSLv3 [SSL3] and TLSv1 [RFC2246]
- HTTP [RFC2616]

1.2 SAML Architecture

In a SAML interaction, there is an *asserting party* or *SAML producer*, which provides information in the form of an XML document called an *assertion*, and a *relying party* or *SAML consumer*, which uses the asserted information for some purpose. SAML provides means by which the relying party can verify that authenticity of the assertion, including digital signatures or retrieval from the asserting party over a secure, authenticated channel. The asserting party may also be considered an *authority* for issuing types of assertions, including *authentication*, *attribute*, and *authorization decision* authorities.

In terms of the SAML protocol, a *SAML requester* sends a SAML request to obtain an assertion or perform another action, such as initiate a logout, to a *SAML responder*, which performs the action and returns a SAML response to the requester. For assertion requests, the responder may be the asserting party, or it may obtain the requested assertion from the asserting party by means outside of the scope of SAML. The requester may be the relying party, or it may pass the assertion onto the relying party by means outside of SAML.

To use the SAML protocol, the SAML requester and responder select a *binding* which specifies how the SAML messages are transmitted using another protocol. As of SAML 2.0, there are a variety of bindings from which to chose for various use cases: SOAP (usually over HTTP), PAOS (reverse SOAP), HTTP Redirect, HTTP Post, HTTP Artifact, and SAML URI. See [SAMLBind] for more information.

A primary use of SAML is *federated single sign on* (SSO), where a user authenticates once to an *identity provider* and then uses that authenticated identity to request access to resources or services from one or

more *service providers* in the same or different security domains. In this interaction, the identity provider is the asserting party and authentication authority, the assertion indicates that the user has authenticated and optionally may include user attributes, and the service providers are relying parties, using the assertion and their own access policies to determine if the user is authorized to access the requested resources or services.

An identity provider may also responsible for the definition and management of its user's identities (for example, in a directory or database), and authentication of its users (for example, through a username and password challenge). A service provider may also be responsible for the evaluation and enforcement of access policies for requested resources or services. These additional responsibilities are outside the scope of SAML, but are required for the end-to-end operation of federated SSO.

A SAML implementation may perform any combination of the roles listed above, depending on the requirements it is to fulfill and the environment in which it is deployed. Conformance to SAML profiles as specified by [SAMLConform] requires that a SAML implementation perform roles specified by the profiles. For example, the Web SSO Browser Artifact Profile (BPP) Producer requires that a conformant SAML implementation provide an asserting party for authentication assertions and a SAML responder to be used by a BPP Consumer to retrieve the assertions.

See the SAML Technical Overview [SAMLTech] for a more complete architectural discussion of SAML.

1.3 SAML Environments

SAML implementations may be designed to operate within specific environments. For example, a SAML federated SSO operation may be conducted differently if it is initiated from a mobile phone rather than from a personal computer. In such a case, the architecture implemented using the SAML specifications may be quite different than that used to enable SSO for employees to their corporate network and internal company websites. Some environments for which particular SSO implementations may exist include e-commerce, mobile phone networks, and enterprise network infrastructure. Specific guidelines are presented for some of these environments.

2 User Agent Considerations

User agents provide the interface between the end user and the SAML implementation for federated single sign on and other operations. User agents include commercial web browsers that use HTTP and mobile devices that use WAP.

2.1 Cookies

Cookies, the HTTP state management mechanism specified in [RFC2965], provide a means for web servers to store information and maintain state in the user agent. Cookies can be either session or persistent. Session cookies are deleted when the user agent terminates, while persistent cookies are stored in durable memory between user agent invocations. A persistent cookies has a lifetime set by the web server and user agents may delete an expired cookie.

In SAML implementations, cookies might be used for maintaining local session state for identity providers or service providers. A cookie (the Common Domain cookie) is also used by the Identity Provider Discover Profile [SAMLProf] to hold a list of IDs for identity providers that a service provider may use to authenticate a user. Use of the Common Domain cookie should be based on the circle of trust policy, and should be consistent within the common domain.

2.1.1 Constraints

There are a number of constraints on how cookies can be used.

2.1.1.1 Domain Restrictions

When a web site sets a cookie, it may specify the DNS domain of web servers to which the user agent is to return the cookie. RFC2965 requires that the cookie's domain domain-match the host name of the web server setting the cookie and the domain not be a top level domain. For example, a web server at `host.domain1.com` can set a cookie for the domain `.domain1.com` but not `.domain2.com` nor `.com`. This restriction is enforced by the default security setting in the predominant user agents. To permit multiple identity providers and service providers in different DNS domains to communicate using cookies, users must lower the default security settings of their user agents. This option is often an unacceptable requirement.

2.1.1.2 Cookie Size and Number Restrictions

RFC2965 recommends the user agents handle at least 300 cookies, at least 4096 bytes per cookie, and at least 20 cookies per host or domain. Some user agents use these as the maximum limits on cookies and some user agents in restricted environments (for example, mobile phones) may have lower limits or may not accept cookies at all.

2.1.1.3 Disabled Cookie Support

For security or privacy reasons, users or their organizations may disable cookie support in their user agents.

2.1.2 Privacy and Security Considerations

If a cookie contains personal or authentication information, care should be taken to protect the contents of the cookie against disclosure or alteration. The `secure` option of the HTTP `SetCookie` header can be used to require that the user agent only send the cookie over a secure channel (for example, HTTP over SSL). Information in the cookie can be encrypted or integrity-protected using a secure hash to prevent undetected alteration of its contents by the end user or an intermediate.

Persistent cookies are of special concern because they exist across invocations of the user agent. If a session authentication token is cached in a persistent cookie, the user exits the browser, and another person uses the system and relaunches the browser, then the second person could impersonate the user (unless any authentication time limits imposed by the authentication mechanism have expired).

Persistent cookies should be used *only* with the consent of the user. This consent step allows, for example, a user at a public machine to prohibit a persistent cookie that would otherwise remain in the user agent's cookie cache after the user is finished.

2.1.3 URL Encoding

A cookie should be URL-encoded before transmission on the wire, but no more than once. When receiving a cookie make sure that the cookie is URL-encoded prior to URL-decoding. This is important since it is not possible to URL-decode a cookie more times than it has been URL-encoded, and older implementations may not URL-encode a cookie. The URL-encoding format is defined in [RFC1738].

2.2 URL Length Considerations

Some user agents are restricted in the size of URL they will support. Implementors should inspect the HTTP `User-Agent` header to check whether that user-agent cannot support URLs that they construct.

It should be noted that the `User-Agent` header may not accurately describe the user-agent being used to access the service, so decisions based on the header contents should be carefully considered.

One alternative to session management using cookies is to embed a session token in all URLs provided by a service (sometimes called *URL-rewriting*). This may not be an option in environments where URL-limited user agents may access the service.

If a service detects that a user agent cannot handle a particular URL using an HTTP `GET`, then the service may use an HTTP `POST` to send the message (see [SAMLProf] for details).

It should also be noted that a SAML authentication request may be constructed (by omitting optional elements in favor of default behavior) in such a way as to be acceptable to *most* user agents. This particular issue will be of more concern in some environments than others. There is further discussion of this in Section 5.2.3 ***[CHECK]*** as it relates to a WAP 2.0 proxying mobile scenario.

2.3 Use of JavaScript/ECMAScript

Popular user agents may be configured to disable the execution of JavaScript (ECMAScript) and some user agents may not support it at all. Implementors should be aware that that content with such scripts may not function as intended and so should provide alternatives if necessary. An example is the use of JavaScript `submit` function in the form generated by the Web SSO Browser Post Profile. The form should contain a button that can be used to manually post the form if JavaScript is disabled in the user agent.

2.4 Caching

User agents, and proxies between the user agents and the web servers, may cache content to improve performance of subsequent requests. There are a number of HTTP headers to control this type of caching, for example, the `Cache-Control` header. Section 3.2.3.2 of [SAMLBind] provides more details on use of these headers. Implementors should be aware of the effect caching might have on SAML interactions and take appropriate measures. An example is the form generated by the Web SSO Browser Post Profile. A user agent might cache the form and on a subsequent federated SSO transfer, return the cached form instead of requesting a new form from the identity provider. The assertion within the cached form data may have expired before the subsequent transfer.

3 Security Considerations

The security requirements for a deployed SAML implementation are dependent on the SAML profiles used and the environments in which they are used. For example, the requirements for an implementation operating over a secure network are different than an implementation used over the Internet.

The SAML conformance document [SAMLConform] specifies some security models that mandatory for conformant SAML implementation to implement. This provides a common set of security models from which deployers of those implementations can chose. SAML implementations may also provide additional security features that deployers may use as appropriate for their environments.

General security requirements are reviewed below.

- **Confidentiality** means that data can be viewed only by the intended recipients of the data. This is usually accomplished through encryption using an algorithm such as 3DES or AES, where the intended recipients have or can obtain the key(s) needed to decrypt the data. Various proven techniques, such as Kerberos and SSL, can be used to generate and distribute keys to the proper parties.
- **Data integrity** means that data cannot be altered without detection. This is usually provided through the addition of a digital signature, a message digest computed from the data using an algorithm like SHA-1 and encrypted using a public key algorithm like RSA and the signer's private key. The recipient of the data verifies the signature by recomputing the digest, decrypting the original digest using the signer's public key, and comparing the two to detect if the data has been altered.
- **Authentication** is the process by which one party in an interaction proves its identity to another party. Typically the authenticating party demonstrates that it possesses a secret, such as a password or key, that other parties would not possess. Encryption techniques are commonly used to enhance the security of the authentication process.
    - **Peer-entity authentication** means that a party that is interacting with another party over a communications channel authenticates itself to the second party. This may involve additional interactions such as challenges between the parties.
    - **Data origin authentication** means that the recipient of data can authenticate the claimed originator of the data. This may need to be performed without (additional) interactions with the originator. Digital signatures are commonly employed for this purpose, as only the originator should possess the private key used to compute the signature.
- **Non-repudiation** means that the recipient of data has proof that the data originated with or was processed by a party. That party cannot then repudiate having originated or processed the data. Digital signatures can also be used for this requirement, as no other party should possess the private key used to compute the signature.
- **Authorization** is the process of determining if a party is allowed to perform a requested action. This involves applying policy of some sort to the requester's authenticated identity, parameters of the request, and environmental variables such as time and the requester's location.
- **Trust** means that one party has obtained security information from another party in a such a manner that the first party has a high degree of assurance that security requirements (encryption, data integrity, authentication, and so on) are being met. With public key cryptography, trust is established through the distribution of X.509 certificates that bind a subject name to the public key associated with a party's private key. A certificates is typically issued by a Certificate Authority (CA), which digitally signs it with its own private key. Trust is established in the CA, and by extension those certificates issued by the CA, by obtaining and importing the CA's own X.509 certificate with its public key.

There is always a trade-off between security on the one hand and performance and ease of deployment and use on the other. Encryption and digital signatures are expensive in terms of performance and require additional, sometimes difficult, configuration. Care should be taken to employ the right amount of security measures as required for the environment. For example, encrypting messages that are sent over an encrypted channel or signing assertions that are carried in signed SAML responses may or may not be necessary, depending on how the data is used.

3.1 Channel Security

SAML implementations may chose to use security mechanisms provided by the communication channels used between participants in SAML interactions. The primary channel used by SAML is HTTP over TCP/IP, so there are a variety of HTTP and TCP security facilities that can be used.

3.1.1 SSL and TLS

SSLv3 (Secure Sockets Layer) [SSLv3] is commonly used to provide confidentiality, data integrity, and authentication for HTTP. All popular user agents and web servers allow the use of HTTP over SSL. SSL uses secret key encryption with generated keys to encrypt the HTTP messages between the client (user agent or other program) and the server (web server or other program), and public key techniques to exchange the keys between the client and server.

There are numerous SSL packages available for use by SAML implementations, including OpenSSL and JSSE (Java Secure Sockets Extension). Each of these packages has its own method of generation and storage of keys and corresponding X.509 certificates.

SAML requesters and responders using the SOAP over HTTP binding should use HTTP over SSL for confidentiality.

TLSv1 (Transport Layer Security) [RFC2246] is an IETF standard very similar to SSLv3. TLS has some technical improvements over SSL, including a more secure message authentication code (MAC) algorithm, pseudorandom generation, and calculation of the master secret. These differences prevent SSLv2 and TLSv1 from interoperating. Consequently many products provide both protocols. In the remainder of this discussion, SSL will mean both SSL and TLS.

SSL and TLS allow a number of choices for algorithms and key lengths (known as cipher suites) to be used to encrypt data sent on the connection. The cipher suite used for a connection is negotiated between the client and server during the SSL handshake; the client and server must have at least one common cipher suite for this negotiation to succeed. The SAML Conformance Document [SAMLConform, Section 3.3] specifies the following required and optional cipher suites

- `TLS_RSA_WITH_3DEC_EDE_CBC_SHA` - required for SOAP over HTTP
- `TLS_RSA_AES_128_CBC_SHA` - optional for SOAP over HTTP
- `SSL_RSA_WITH_3DES_EDE_CBD_SHA` - required for web SSO profiles using SSL
- `TLS_RSA_WITH_3DES_EDE_CBC_SHA` - required for web SSO profiles using TLS

3.1.1.1 SSL Server Side authentication

SSL requires that the server authenticate itself to the client. To do this, the server must have a public-private key pair and an X.509 certificate that binds the public key to a subject name with the server's host name in the CN part of the name, for example, `cn=host.company.com, o=company, c=US`. During the SSL handshake a challenge from the client is signed using the server's private key, the signed challenge and the server's certificate are sent back to the client, and the client verifies the signed data with the public key from the certificate. (Note that this is a simplified view of the actual SSL handshake.) The client also verifies that the certificate has been issued by a certificate authority (CA) that it trusts. Finally, the client should also verify that the CN part of the certificate's subject name matches the host name used to connect to the server as a safeguard against spoofed server addresses.

If the server's certificate is not trusted, or the certificate and server host name do not match, user agents may issue a warning and allow the user to decide whether or not to proceed. HTTP client libraries may simply reject the connection. User agents and HTTP client libraries typically are pre-configured to trust a number of popular CAs such as VeriSign, so no additional configuration is required for these clients to trust web sites certified by these CAs.

SSL server side authentication may be used by a SAML requester using the SOAP over HTTP binding to authenticate the SAML responder. To do this, the requester must be configured to trust the CA that issued the responder's certificate.

SAML conformant implementations are required to provide the SOAP over HTTP and the URI bindings over SSL/TLS with server side authentication [SAMLConform, Section 3.3].

Server side authentication is also required by the Web SSO Browser Artifact and Post Profiles (BAP and BPP) to ensure that the user agent is connecting to the correct identity provider and service provider. As noted above, the user agent notifies the user if the server may not be trustworthy.

3.1.1.2 SSL Client Side Authentication

SSL does not require that the client authenticate itself to the server. (This is what allows millions of browser users to use HTTP over SSL without any special configuration). The server may optionally request that the client authenticate itself using an SSL client challenge. For this, the client must have a public-private key pair and an X.509 certificate that binds the public key to a subject name that identifies the client. SSL itself puts no requirement on the client's subject name, although the application using HTTP may do so. During the SSL handshake the server sends a challenge to the client, which the client signs with its private key and sends the signed challenge and its certificate back to the server for verification. (Again this is a simplified view.) Since SSL client side authentication uses long random keys with the cryptography in SSL, it is considered to be stronger that other forms of authentication that use passwords.

SSL client side authentication can be required by a SAML responder for SOAP over HTTP bindings. The SAML requester then needs to be configured with a key pair and a certificate.

SAML conformant implementations are required to provide the SOAP over HTTP and the URI bindings over SSL/TLS with client as well as server side authentication. side authentication [SAMLConform, Section 3.3]

SSL client side authentication may also be used by an identity provider to authenticate the user agent prior to the start of a federated single sign-on. This may be required by local identity provider policy or may specified in an `<AuthnRequest>` from a service provider that needs strong user authentication.

3.1.2 HTTP Basic and Digest Authentication

RFC2617 [RFC2617] specifies techniques for user agents to authenticate to web servers. These methods can also be used by SAML requesters using HTTP bindings to authenticate to SAML responders. The RFC defines an `Authorization` header to carry authentication information in HTTP requests to the server. The server uses the header to authenticate the HTTP client. After authentication the server can apply authorization policy to the request to decide whether or not to complete processing of the request.

If the server requires the `Authorization` header but it is not present in the request, the server can send a `401 Not Authorized` response to the client, which passes authentication requirements to the client. The client is then supposed to obtain the authentication information from the user and resend the request with the Authorization header. Popular user agents display a dialog box to get a username and password when they receive a `401` response. It is strongly recommended that SAML requesters and responders not depend on the `401` response to perform HTTP authentication -- the SAML requester should include an `Authorization` header in the (first) HTTP request to the SAML responder and the SAML responder should check an incoming request for an `Authorization` header. The SAML conformance document

[SAMLConform, Section 3.3] states that "the SAML requester MUST preemptively send the authorization header in the initial request".

There are two methods specified by RFC2617:

- **HTTP Basic Authentication**: The `Authorization` header includes a username and password, base-64 encoded but not encrypted. The server looks up the password it has registered with the username to authenticate the client. Since the `Authorization` header is not encrypted, basic authentication should only be used with HTTP over SSL to ensure that the username and password cannot be intercepted by a third party. All popular user agents support HTTP basic authentication.

  SAML conformance implementations must provide HTTP basic authentication for the SOAP over HTTP and URI bindings, with and without SSL/TLS [SAMLConform, Section 3.3].

- **HTTP Digest Authentication***:* The `Authorization` header includes the username and a digest of the HTTP message computed using the client's password. The actual password is not transmitted. The server looks up the password it has registered with the user name, recomputes the digest and compares it to the transmitted digest to authenticate the client. This technique also provides message integrity, since the digest comparison will fail if the message has been altered. However, most user agents and HTTP client libraries do not support HTTP digest authentication.

  HTTP digest authentication for the SOAP over HTTP binding is not required for SAML conformance.

In either form of these methods, the username and password used by the client must be communicated to the server in a trusted manner, and the server must store the password in a way that does not disclose it to other parties. For basic authentication, the password can be transmitted and stored as a SHA-1 hash, and the server can compute the hash of a received password for comparison. Passwords stored in some directories will automatically be hashed. However, for digest authentication the server must have the clear text for the password, which may further limit its usefulness.

Since HTTP authentication is based on passwords, it is considered to be weaker than SSL client authentication. There are a number of potential attacks on HTTP authentication detailed in [RFC2617]. In particular, passwords chosen by humans tend to be susceptible to dictionary and brute force attacks.

3.1.3 IPsec and VPNs

IPsec [RFC2401] is a set of security extensions to the IP layer of the TCP/IP protocol stack that provides channel security, including confidentiality, data integrity and authentication, between two host systems. There are many VPN (Virtual Private Network) products built on top of IPsec that secure connections from systems on the Internet to corporate networks. In some cases, these may provide sufficient security for SAML interactions and HTTP over SSL is not required. However, if a VPN only secures the channel to the corporate security parameter, some additional security mechanism may be required within the corporate network.

3.2 Message Security

In addition to or in place of channel security, SAML implementations may use security features of the SAML message protocol, including signature and encryption.

3.2.1 XML Signature

The W3C XML Signature standard [XMLSig] specifies the format and processing rules for digital signatures in XML documents. The `<ds:Signature>` element includes a `<SignedInfo>` element that specifies what part(s) of an XML document are signed, and what canonicalization, digest and signature algorithms were used, a `<SignatureValue>` element that contains a message digest of the signed document parts, encrypted by the signer's private key and base64-encoded, and an optional `<KeyInfo>` element that contains key information to be used to verify the signature, for example an X.509 certificate

with the signer's public key and name. The XML Signature standard allows many variations of the signed info references, algorithms, and key information. Implementations using XML Signatures either need to be prepared to accept a wide variety of signature types, or they need to use a more restrictive profile of the allowed signature features.

3.2.1.1 SAML Signature Profile

The schema for SAML assertions and messages include optional `<ds:Signature>` elements. Section 5 of the SAML Core specification [SAMLCore] provides a profile for XML Signature usage which recommends the use of the RSA and SHA-1 algorithms and mandates:

- enveloped signatures (the `<ds:Signature>` element is included in the signed document and the enveloped signature transform is used to extract it before verification),
- reference by document ID in the `<SignedInfo>` element,
- exclusive canonicalization transform to ensure that the different implementations can generate the same byte string for signing and verification, and
- no other transforms.

The Signature profile in the SAML Core does not provide restrictions on the use of the `<KeyInfo>` element. Consequently there needs to be agreement (perhaps further profiling) among interoperable SAML implementations on which `<KeyInfo>` variants are supported. Implementations that verify signatures need to have some way to determine if the data in a `<KeyInfo>` is to be trusted. Some possible `<KeyInfo>` choices are

- No `<KeyInfo>`: The verifier has to determine the signer from the context of the assertion or message. The `<Issuer>` element of the assertion or message can be used for this. The verifier must have previously obtained and stored a public key or certificate associated with the issuer to complete the verification.
- `<X509Data>`: The verifier can use the public key in the included X.509 certificate to verify the signature. The verifier has to verify the validity of the certificate, as discussed in Section 3.3, and must also verify that the certificate is appropriate for the signer, for example, by comparing the certificate subject name against the signer's configuration.
- `<KeyName>`: The verifier must have previously obtained and stored a public key or certificate associated with the key name to complete the verification.

3.2.1.2 Integrity, Data Origin Authentication, and Non-Repudiation

One of the primary uses of signatures in SAML is to protect the integrity of data that passes through untrusted components. In particular responses are signed in the Web SSO Browser Post Profile (BPP) because they pass through the user agent and hence are susceptible to tampering. Assertions obtained through a browser SSO profile might also be signed if they are to be passed to other components that need to verify their origin and integrity. Assertion and message signatures may also be employed for non-reputation of the claims and actions they represent.

3.2.1.3 Peer Authentication

Signed SAML request and response messages may be used to authenticate the requesters and responders, in lieu of the channel authentication methods discussed earlier. Since it uses public key cryptography with long random keys, signature authentication has similar strength to SSL client authentication, and it may be easier to configure. But there is significant performance overhead to signing each message.

The `<AuthnRequest>` message for federated SSO defined in [SAMLProf] may be signed by agreement between the identity provider and service provider and indicated by the `WantAuthnRequestsSigned`

attribute of the provider metadata. It may be reasonable to not sign this message in some deployment contexts, for example, an enterprise network, where access to the network and its systems is moderated by some means out of the scope of the SAML architecture.

3.2.2 XML Encryption

The W3C XML Encryption standard [XMLEnc] specifies how XML documents can carry encrypted data using `<xenc:EncryptedData>` and `<xenc:EncryptedKey>` elements. These elements are typically used as follows. The encryptor generates an encryption key, creates `<xenc:EncryptedData>` using a secret key algorithm like 3DES and AES, and creates an `<xenc:EncryptedKey>` for each recipient by encrypting the encryption key using a public key algorithm like RSA and the recipient's public key. Recipients decrypt their `<xenc:EncryptKey>` using their private keys, and then decrypt the `<xenc:EncryptedData>` using the encryption key.

The SAML schema includes a number of encrypted elements for data that might need to be confidential.

- `<EncryptedId>` can be used to protect a principal's name in a `<Subject>`.
- `<EncryptedAssertion>` can be used in `<Advice>` and `<Evidence>` to protect an entire assertion
- `<EncryptedAttribute>` can be used in an `<AttributeStatement>` to protect attribute data.

SAML does not provide any additional guidance on the use of XML Encryption for these elements. It is up to the parties involved in the SAML interactions to agree on the parameters of the encryption (algorithms, key length, and so on.) Public key trust set up for signature verification may be useful for encryption as well.

3.2.3 WS-Security

The Web Services Security family of specifications [WSS] define security mechanisms for SOAP that may be used for the SAML SOAP binding for peer authentication, confidentiality and integrity. WS-Security defines a SOAP header, `<wsse:Security>`, that carries a security token used to secure the SOAP message. There are a number of security token profiles under development.

- **Username token** [WSSUser] includes a username and a password or password digest for authentication for the SOAP client. This is roughly equivalent to the HTTP basic and digest authentication in terms of authentication strength and configuration requirements.
- **X.509 token** [WSSX509] includes an X.509 certificate used to sign or encrypt the SOAP message, using the XML Signature or XML Encryption standard previously discussed. This may provide authentication of the SOAP client, confidentiality and integrity, and non-repudiation
- **SAML token** [WSSSAML] includes a SAML assertion with claims about the SOAP client, such as authenticated identity and attributes. The SAML assertion may include key information in a `<SubjectConfirmation>` element that relates to an XML Signature in the SOAP message. This can be used to authenticate the SOAP client.
- **Kerberos token** [WSSKerb] includes a Kerberos ticket granted to the SOAP client for authentication and encryption to the SOAP server. This may be used when the client and server are within the same Kerberos realm, or when the client and server are in separate Kerberos realms that have established cross-realm trust.
- **REL token** [WSSRel] includes a ISO/IEC 21000-5 Rights Expression, used for Digital Rights Management (DRM)

3.3 Trust Guidelines

Establishing trust in other parties is a common requirement for most of the security mechanisms described above. As noted earlier, for mechanisms based on public key cryptography, this means importing and configuring trust for X.509 certificates. Each security package has its own form of trust store for this purpose. JSSE and Java XML security packages, for example, certificates are imported into a keystore file using the keytool program or the equivalent Java classes. For OpenSSL, certificates are added to the appropriate .pem files configured for the package.

3.3.1 Trusting Certificate Authorities

A SAML implementation may implicitly set up trust for all parties certificated by a Certificate Authority by importing the CA's certificate. This provides a public key to use to verify the CA's signature on the certificates it has issued. This method works best when many parties are certified by a small number of CAs. Once the SAML implementation has verified that a party's certificate is valid, it still must determine if the certificate is appropriate for that party to prevent the party from impersonating another party certified by the CA.

To obtain its own certificate from a CA, the SAML implementation has to first generate its public/private key pair and a certificate request with the public key and its desired name and other attributes. There are a number of existing tools to do this, such as the Java keytool and the openssl programs. The certificate request is sent to the CA through an interface provided by the CA, for example, an email address or a web form, the CA issues the certificate, and the SAML implementation imports its certificate into its configuration.

There may be a hierarchy of CAs, which each CA itself certified by its parent CA, to produce a certificate chain, up to a root CA. A SAML implementation may choose to import the root CA for the widest trust, and each signed or encrypted item must include the certificate chain up to but not including the root. Or the implementation may import a lower level CA for more limited trust, and each signed or encrypted item only need to include a partial chain up to but not including the trusted certificate.

3.3.2 Trusting Self-Signed Certificates

SAML implementations may use self-signed certificates as an alternative to setting up CA trust. In a self-signed certificate, the issuer and the subject of the certificate are the same; a root CA certificate is in fact self-signed. A SAML implementation must explicitly set up trust for each self-signed certificate. This works best if there are a small number of parties to trust. It is also equivalent to the case where each party uses its own CA.

Programs like keytool and openssl have the ability to generate self-signed certificates without requiring a certificate request to a CA. Consequently self-signed certificates may be easier to use than CA issued certificates in those cases where they are appropriate.

3.3.3 Evaluating Certificates

Each SAML implementation needs to evaluate the trustworthiness of the certificates it uses, either from the signed/encrypted data or from its own trust store. Suggested items to check are listed below. For certificate chains, these steps would be replied recursively up the chain.

- The current time is within the validity period (valid from and valid to) of the certificate.
- The certificate key usage, if specified in the certificate, is appropriate for the use of the certificate.
- The certificate signature by its issuer can be verified using the issuer's public key.
- If the issuer publishes certificate revocation lists (CRLs), the certificate has not been revoked.

There are protocols, including OCSP [RFC2560] and XKMS [XKMS], that can be used with CAs and other services to validate certificates.

4 Authentication Mechanisms

4.1 Authentication Mechanisms are Orthogonal to Single Sign-On

Single sign-on is a means by which a service provider or identity provider may convey to another service provider or identity provider that the user is in fact authenticated. The means by which the user was originally authenticated is called the authentication mechanism.

User authentication methods are commonly classified as being one-factor, two-factor, or three-factor. Generally, the more factors used, the more reliable the authentication method. The first factor may be "something you have" such as a smart card. The second factor might then be a PIN for the smart card, or a password -"something you know". A possible third factor might be "something you are", such as a body characteristic (fingerprint, for example).

Authentication mechanisms consist of combinations of these factors. Examples of authentication mechanisms include username with password, X.509 certificate via SSL or TLS), or Kerberos.

4.2 Identity Provider Session State Maintenance

Identity providers need to maintain authentication state information for principals. This is also known as "local session state maintenance", where "local" implies "local to the identity provider". There are several mechanisms for maintaining local session state information in HTTP [RFC2616] user agents. Cookies are one such mechanism and are specified in [RFC2965]. Identity providers use local session state information, mapped to the participating user agent, as the basis for issuing authentication assertions to service providers who are performing SSO profiles with the identity provider. Thus, when the principal uses his user agent to interact with yet another service provider, that service provider will send an `<AuthnRequest>` to the identity provider. The identity provider will check its local session state information for that user agent, and return to the service provider a `<Response>` containing an authentication assertion if its local session state information indicates the user agent's session with the identity provider is presently active.

In most cases, a service provider will submit an authentication request to an identity provider, and the recipient will return an authentication response. It is, however, possible for an identity provider and service provider to agree out-of-band to support a model whereby the identity provider creates an authentication response without first having received an authentication request. This allows single-signon for a principal at a service provider to occur without the principal first visiting that service provider, with the following provisions:

1. An SP should drop an unsolicited `<AuthnResponse>`, unless it has negotiated some out-of-band agreement with the IdP sending the response.

2. An SP can determine that a `<Response>` is unsolicited by looking for the `InResponseTo` attribute.If not found, the response is unsolicited, and they should examine the assertion to determine whether they have an out-of-band agreement with the message sender.

4.3 Credentials

Credentials are used in a number of ways in a single sign-on system and are often the basis for establishing trust with the credential bearer. Credentials may represent security-related attributes of the bearer, including the owner's identity. Sensitive credentials that require special protection, such as private cryptographic keys, must be protected from unauthorized exposure. Some credentials are intended to be

shared, such as public-key certificates. Credentials are a general notion of the data necessary to prove an assertion. For example, in a password authentication system, the user name and password would be considered credentials. However, the use of credentials is not limited to authentication. Credentials may also be relied upon in the course of making an authorization decision.

As mentioned above, certain credentials must be kept confidential. However, some credentials not only need to remain confidential, but also must be integrity-protected to prevent them from being tampered with or fabricated. Other credentials must have the properties of a nonce. A nonce is a random or non-repeating value that is included in data exchanged by a protocol, usually for guaranteeing liveness and thus detecting and protecting against replay attacks.

4.4 Authentication Type, Multi-tiered Authentication

All authentication assertions should include an authentication type that indicates the quality of the credentials and the mechanism used to vet them. Credentials used to authenticate a user or supplied to authorize a transaction and/or the authentication mechanism used to vet the credentials may not be of sufficient quality to complete the transaction.

For example, a user initially authenticates to the identity provider using username and password. The user then attempts to conduct a transaction, for instance, a bank withdrawal, which requires a stronger form of authentication. In this case the user must present a stronger assertion of identity, such as a public-key certificate or something ancillary such as birth date, mother's maiden name, etc. This act is *reauthentication* and the overall functionality is *multi-tiered authentication*. Using multi-tiered authentication can be a policy decision at the service provider and can be at the discretion of the service provider. Or it might be established as part of the contractual arrangements of the circle of trust. In this case, the circle of trust members can agree among themselves upon the trust they put in different authentication types and of each other's authentication assertions. Such an agreement's form may be similar to today's certificate practice statements (CPS) (for example, see http://www.verisign.com/repository/cps20/cps20.pdf). The information cited in such a document may include:

- User identification methods during credentials enrollment
- Credentials renewal frequency
- Methods for storing and protecting credentials (for example, smart-card, phone, encrypted file on hard drive, etc.)

While the current SAML specifications allow service providers, identity providers, and user agents to support authentication using a range of methods, the methods and their associated protocol exchanges are not specified within SAML documents. Further, the scope of the current SAML specifications does not include a means for a communicating identity provider and user agent to identify a set of methods that they are both equipped to support. As a result, support for the SAML specifications is not in itself sufficient to ensure effective interoperability between arbitrary identity providers and user agents using arbitrary methods and must, instead, be complemented with data obtained from other sources.

Also, the scope of the current SAML specifications does not include a means for a service provider to interrogate an identity provider and determine the set of authentication profiles for which a user is registered at that identity provider. As a result, effective service provider selection of specific profiles to authenticate a particular user will require access to out-of-band information describing users' capabilities. For example, members of a given circle of trust may agree that they will label an authentication assertion based on PKI technology and face-to-face user identity verification with substantiating documentation at enrollment time to be of type "Strong." Then, when an identity provider implementing these policies and procedures asserts that a user has logged in using the specified PKI-based authentication mechanism, service providers rely upon said assertion to a certain degree. This degree of reliance is likely different from the degree put into an assertion by an identity provider who uses the same PKI-based authentication mechanism, but who does not claim to subject the user to the same amount of scrutiny at enrollment time.

This issue has another dimension: Who performs the reauthentication? An identity provider or the service provider itself? This question is both an implementation and deployment issue and an operational policy issue. Implementations and deployments need to support having either the identity provider or the service

provider perform reauthentication when the business considerations dictate it (that is, the operational policy). For example, a circle of trust may decide that the risk factors are too large for having the identity provider perform reauthentication in certain high-value interactions and that the service provider taking on the risk of the interaction must be able to perform the reauthentication.

4.5 Mutual Authentication

Another dimension of the authentication type and quality space is mutual authentication. For a user authenticating himself to an identity provider, mutual authentication implies that the identity provider server authenticates itself with the user as well as vice versa. Mutual authentication is a function of the particular authentication mechanism employed. For example, any user authentication performed over SSL or TLS is mutual authentication because the server is authenticated to the client by default with SSL or TLS. This feature can be the basis of some greater assurance, but does have its set of vulnerabilities. For instance, the server may be wielding a bogus certificate, and the user may not adequately inspect it or understand the significance. It may also be the case that the server does not know if the real user is present for the authentication unless some PIN or password is bound into the authentication protocol. 483

4.6 Validating Liveness

Liveness refers to whether the user who authenticated at earlier is the same user who is about to perform a given operation. For example, a user may log in and perform various operations and then attempt to perform a given operation that the service provider considers high-value. The service provider may initiate reauthentication to attempt to validate that the user operating the system is still the same user that authenticated originally. Even though such an approach has many vulnerabilities, that is, it fails completely in the case of a rogue user, it does at least augment the service provider's audit trail. Therefore, at least some service providers will want to do it.

Authentication assertions from identity providers may contain a `<SessionOnOrAfter>` element. If this attribute was specified and the time of the user request is past the specified time, the service provider should redirect the user back to the identity provider for reauthentication.

4.7 Communication Security

A service provider can reject communications with an identity provider for various reasons. For example, it may be the policy of a service provider to require that all protocol exchanges between it and the bearer of a credential commence over a communication protocol that has certain qualities such as bilateral authentication, integrity protection, and message confidentiality.

4.8 Compromised Credentials

A service provider may discover that a user's credentials have been compromised. A preferred solution to this is to send a single logout message for that user. To aid in this, service providers should expose a SOAP single logout endpoint.

5 Replication Considerations

Components of a SAML implementation may be replicated for recovery after failure (failover) and for load balancing. This section discusses aspects of the SAML protocols that affect replication.

5.1 URL Endpoints

A SAML implementation provides URL endpoints to be used by other SAML implementations, for example, the SOAP Responder URL and the Artifact Consumer URL. The other SAML implementations must configure these endpoints, either through the use of SAML metadata or through implementation-specific configuration interfaces. It is undesirable for these endpoints to change frequently, so a replicated SAML implementation should attempt to present one set of endpoints serviced by its replicas. One way to accomplish this is through the use of front-end hardware that presents one IP address to the Internet and routes incoming requests to replicated components. The hardware must provide its own redundancy for failover recovery. Another technique is round robin DNS, where the DNS mappings for the endpoint hostnames are rotated between the replica IP addresses. OTHERS?

5.2 Keys

Similarly to the URL endpoints, a replicated SAML implementation should appear to have one set of keys for SSL and digital signatures to other SAML implementations. This can be accomplished by duplicating keystores or the equivalent among the replicas, or by retrieving keys from a replicated database.

5.3 Artifacts

For the Web SSO Browser Artifact Profile (BAP) and other profiles using the HTTP Artifact Binding, one SAML implementation associates an assertion or response with an artifact, sends the artifact to a second SAML implementation, which then sends an `<ArtifactResolve>` request back to the first implementation to retrieve the original SAML assertion or response. In a replicated implementation, one replica may create the artifact associate and a different replica may need to retrieve it. One way that this requirement can be met is by storing the artifact associations in a replicated database. To fulfill the one-time use requirement for artifacts [SAMLBind, Section 3.6.5.2], the replica responding to the artifact request must delete the artifact association for all replicas.

5.4 Received BPP Responses

The Web SSO Browser Post Profile (BPP) has a one time use requirement for posted responses similar to that for BAP [SAMLProf, Section 4.1.4.5]. A service provider is supposed to reject a response that has been replayed. To do this, the service provider needs to keep track of the responses it has received, until the expiration of the SSO assertions within the responses. Each replica must have access to the list of received responses, which can be provided through a replicated database.

5.5 User Identities and Aliases

Replicated SAML implementations in the role of identity providers need access to user information to authenticate users and create SSO assertions. This is usually provided through a user directory or database, which may itself be replicated for high availability and performance. Implementations may need to reconnect to directory or database replicas in case of failures.

710 The SAML Name Identifier Management profile [SAMLProf, Section 4.5] allows the establishment of user
711 aliases shared among identity and service providers. All replicas of a SAML implementation must have
712 access to these aliases, which can be stored in the user repository or in a separate replicated database.

713 5.6 Session State

714 The Single Logout (SLO) profile [SAMLProf, Section 4.4] requires that the SAML implementation keep
715 track of sessions initiated throught the SSO profiles. This state must be shared between replicas.

6 Guidelines for Mobile Environments

The SAML specifications recognize that the mobile environment requires specific treatment, and define specific methods which may be used by mobile devices and other systems operating in mobile networks. In particular, mobile devices may offer a more constrained environment for identity management than do other devices such as personal computers on an enterprise network. SAML defines one specific profile for mobile environments (the ECP Profile - see [SAMLProf]) and an adaptation of the Browser Artifact Profile (BPP) which enables the transmission of an artifact using WAP/WML [WML]).

It should be noted, however, that there are specific items that should be considered when implementing or deploying in a mobile environment. Some of these are noted below. The guidelines in this section relate to these specific deployment issues.

6.1 Some Mobile-specific Deployment Considerations

- **Use of the radio resource**: In some mobile environments, radio bandwidth may be restricted or costly, and in such environments, it may be desirable to limit the number of exchanges with a mobile device over the radio link.
- **Reliability/Latency**: Mobile devices may have poor network connectivity over a radio link, causing unreliable network connections, or latency over such connections.
- **Ease of deployment:** WAP 2.0 provides a different architecture for browsing from WAP 1.x. However, there remain many mobile devices deployed that are equipped with WAP 1.x-compliant user-agents. To enable handset usage of the SAML profiles may in some cases require the deployment of handsets that utilize additional or improved software.
- **Presence of SIM card**: GSM-based networks make use of the Subscriber Identity Module (SIM) card. Such cards may provide enhanced security for identity-based transactions.
- **Network roaming**: Mobile roaming business agreements established between network operators provide an important basis for trust between SAML providers.
- **Link security**: WAP 1.x does not allow for secure, encrypted links at the transport layer between a mobile device and a service provider. WAP 2.0 introduced TLS which does allow for such links. It is strongly recommended in the SAML specifications that adequate security measures be taken to protect data transmitted via the SAML protocols, including the use of transport-layer security.

6.2 Using the SAML SSO Profiles in Mobile Environments

6.2.1 Summary of the SAML Profiles in Mobile Environments

6.2.1.1 Browser Artifact Profile (BAP)

The SAML Browser Artifact Profile [SAMLProf] may be used to enable WML redirects to perform SSO. The following list summarizes this profile with respect to some of the deployment issues noted above

- **Latency**: The artifact profile will force the user agent through a minimum of three redirects. If such redirects are performed on an unreliable network, then there may be significant delays in processing of the SSO request. It should be noted, however, that the benefits that SSO provides (i.e. removing extra login pages at SAML-enabled sites) will mitigate the effect of such redirects.

- **Reliability**: As noted above, conducting three redirects over an unreliable network may prove to be a less-than desirable user experience.
- **Ease of deployment**: The artifact profile is compatible with any WAP 1.x or 2.x (x >= 0) user-agent, which implies that there is no concern about deploying new software or hardware.
- **Radio consumption**: The redirects necessary for use of this profile will use the radio link. The benefits afforded the user by use of the SSO protocols should mitigate any concern about additional consumption of radio resources.
- **Link security**: WAP 1.x does not allow for end-to-end secure transport-layer link between the device and the service provider. WAP 2.0 does provide this possibility.

6.2.1.2 Enhanced Client or Proxy (ECP) Profile

The ECP profile [SAMLProf] is an alternative to WML usage of the artifact profile. A SAML-enabled client or proxy may initiate this profile by specifying `Accept: application/vnd.paos+xml` and `PAOS` headers in an HTTP request to a service provider.

- **Latency**: If a device is performing the ECP profile, then a number of redirects are still needed for the SSO to occur. Once again, in practice, SAML deployments will cut down on the overall number of logins required, and mitigate the performance effect of these redirects. A useful optimization, however, would be to deploy an ECP proxy in environments where network latency is an issue. This may further mitigate the effect of network latency by minimizing use of the radio network.
- **Reliability**: As mentioned in the previous section, the user may experience difficulties when conducting several redirects over an unreliable network. Deploying an ECP proxy may again mitigate this issue by restricting redirects to the more reliable wired network.
- **Ease of deployment**: If a proxy server is deployed to perform the ECP profile on behalf of mobile devices, then there are no concerns about device deployment. However, if the device is performing the ECP profile itself, it may be necessary for additional software to be deployed, or upgraded on the mobile device.
- **Radio consumption**: The redirects necessary for use of this profile will be made over the radio link in the case of a mobile device performing the ECP profile. Deploying a proxy for the device may ease this concern, as the ECP redirects will be performed on the wired network.
- **Link security**: In the case of a WAP 2.0-compliant mobile device, end-to-end encrypted connections are possible between the device and the service provider at the transport layer. It should be noted that WAP 1.x does not allow for such connections. In environments where a proxy server is deployed in order to perform the ECP profile, however, it may be necessary to create two separate secure sessions - one between the mobile device and the proxy, and another between the proxy and the service provider. See Section 6.2.3 for more information about this scenario.

6.2.2 Methods of Roaming Using the SAML Protocols

As the user of a mobile device moves from place to place, they may roam from one mobile operators network to another. These operators may have in place business trust agreements that allow a user to access services from service providers that are not on the users home network. Authentication of the user may be necessary in order to access such services on the visited network. It may be necessary for identity and service providers to establish dynamic trust agreements. These allow authentication of the user by an identity provider on their home network, or for an identity provider on the roaming network to perform authentication of the user as a proxy for the home network identity provider. These methods are summarized below.

- **Proxying**: An identity provider on the visited (roaming) network proxies SAML authentication messages from the service provider on the visited network to the appropriate identity provider on the users home network, based on existing roaming contracts.
- **Direct**: The service provider on the visited network sends authentication requests directly to the appropriate identity provider on the users home network, based on an existing trust chain (such as PKI infrastructure or roaming agreements) and directed via the existing (General Packet Radio Service (GPRS)) technical infrastructure (i.e. not using any specific SAML mechanism to direct requests to the correct identity provider). Using identity provider proxying may reduce the total number of federation and/or redirect operations that occur, thus providing a better user experience.

6.2.2.1 Roaming Using the Proxy Method

Even with roaming agreements in place between the user's home network and the visited network, a service provider on the visited network may not be able to accept authentication assertions directly generated by an identity provider on the users home network. The service provider may not even know about the identity provider on the home network. This situation might necessitate the intervention of an identity provider on the visited network to bridge the two networks. Such an identity provider may act as a proxy for the identity provider on the user's home network.

For the proxying methods, the following guidelines may be followed by those deploying SAML services in a mobile environment.

- A service provider will not know *a priori* that the Principal may be roaming on the visited network, and may also not have an account with the local identity provider.
- In order to provide for such users, a service provider on the visited network should issue authentication requests with a `<NameIDPolicy>` of *any* or *onetime* **[??]** when operating in a mobile environment. This would ensure that federation of the user's identity would not be required in order for the roaming user to use services on the visited network.
- A SAML-enabled client "has, or knows how to obtain, knowledge about the identity provider that a Principal wishes to use with the service provider". In certain environments, a enhance proxy may also be present, which is "an HTTP proxy (typically a WAP gateway) that emulates a enhanced client"

The following guidelines pertain to enhanced proxy and client deployments in a mobile roaming through proxying situation

- Any identity provider on the visited network may be provisioned with information that enables them to have some knowledge of their network's roaming partners.
- Any enhanced proxy or client present on the home network should be provisioned with a list of identity providers who are partners of the user's home identity provider. It should be noted that how such provisioning is carried out is not governed by the SAML specifications.

In addition, some specific guidelines may be used with the proxy method of mobile roaming.

- In order to ensure that proxying is considered by the recipient of an authentication request, the `ProxyCount` attribute should be set to a value greater than 0. In order to minimize network latency caused by multiple proxying steps, however, it is recommended that `ProxyCount` be set no higher than 1.

6.2.3 WAP and the Enhanced Proxy

In version 1 of WAP, a *WAP Gateway* translates between WAP protocol flows and HTTP protocol flows. There is no end-to-end TLS-secured session directly between the user-agent and the (web-based) service provider. The user agent maintains one session with the WAP gateway, and the gateway maintains a separate session with the web server providing the service to the end-user.

Such an environment allows the ECP profile to be managed entirely on the WAP gateway, allowing that gateway to become a proxy for the mobile device, providing a performance benefit.

In an environment where the user-agent accessing a service may be severely constrained, unable to process URLs over a certain size; ECMAScript (thus preventing HTTP POST without user interaction), and cookie-based sessions, then using a WAP gateway to perform ECP interactions is a useful method of circumventing such user-agent deficiencies without requiring new mobile handset deployments.

In WAP version 2, the user-agent is XHTML/HTTP-compliant, and thus an end-to-end TLS-secured session is possible directly between the user-agent and the web server, without the necessity for a gateway to intermediate between the HTTP and WAP protocol flows. The enhanced client may perform the ECP profile directly, without the need for an additional intermediary. In this situation, the WAP gateway becomes an HTTP proxy server, merely forwarding HTTP requests and responses. However, in certain environments a proxy for the ECP profile may still make sense as an optimization.

In such an environment, any proxy deployed between a service provider and the user-agent will see only an encrypted HTTP protocol flow, and will not be able to perform the ECP profile, as specified, on behalf of the user agent. 652

One solution to this issue is for the service provider to redirect the user-agent to the proxy, allowing the gateway to perform the ECP profile. In this situation, the ECP profile is performed as specified in [SAMLProf] with an additional step, which allows the service provider to determine that usage of a enhanced proxy has been requested by the principal, to obtain metadata for the proxy, and finally to redirect the user agent to the WAP proxy. At this point, the ECP profile will be performed by the proxy, until the final response is received by the WAP proxy and forwarded to the user-agent.

> **Security note:** It is strongly recommended that the proxy and the identity provider in this adaptation of the ECP profile are hosted by the same entity, in the same secure facility, to avoid the possibility of a man-in-the-middle attack being launched.

6.2.3.1 ECP Profile with Enhanced Proxy

[NEED FIGURE]

1) Step 1 == ECP Step 1.

2) Step 2 is non-ECP.

The SP should recognize whether the UA uses ECP or not by metadata. The metadata could be encoded in the extension of HTTP header added by ECP (the Step 1 HTTP request from proxy to SP). It may look as follows.

```
GET /somewhere.html HTTP/1.1
...
?: LIBV=urn:liberty:iff:1.2
X-EP: http://ep.operator.com/ep/
```

[NEEDS CHECKING]

This method would require additional work for the SP. When the UA first requests an HTTPS session to the SP, the SP should read the User Agent in the HTTP header and should judge if the UA uses ECP or not. The SP then should lead the UA to non-secure HTTP connection where the EP can add the above extension on the HTTP header bound for SP.

3) Step 3 is non-ECP.

If the Principal uses ECP, redirect principal to the proxy/SSO URL written in the ECP metadata. SP's URL should also be encoded in the URL.

4) Step 4 is non-ECP.

The User Agent is now redirected to EP.

5) Step 5 is non-ECP.

The proxy decodes the requested URL, and redirects the User Agent back to SP.

Steps 6-14 are the usual ECP steps.

6.3 Mobile Provisioning Using Tamper-resistant Smart Cards

An enhanced client (EC) should be able to make decisions regarding the handling of the SAML protocols. In order to do so, the EC should know with which identity providers it should communicate and thus needs certain information about them. Given certain constraints of the mobile environment, it may not be possible for the EC to rely exclusively on acquiring metadata about identity providers using the standard SAML methods for metadata acquisition (see [SAMLMetadata]). Better performance and security may be achieved by provisioning the EC at some time prior to use of the SAML protocols, using a tamper-resistant, cryptography-enabled device with protected memory (such as a GSM SIM card). In such cases, we recommend the use of the following schema to provide a minimal set of configuration parameters that will enable the EC to be provisioned with necessary data. The metadata in this schema incorporates actual values such as the URLs of the default and partner identity providers that would be used by the mobile device.

> **Note:** Work is underway to standardize data structures, and methods of access for such provisioning data in the ETSI Project Smart Card Platform. When this work is complete, the following section will reference that work.

[CHECK]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
   targetNamespace="urn:liberty:lecprov:2003-08"
   xmlns="urn:liberty:lecprov:2003- 08"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
     schemaLocation=
         "http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
   <xs:element name="LecProv" type="LecProvType"/>
   <xs:complexType name="LecProvType">
      <xs:sequence>
         <xs:element name="IdPConfig" type="IdPConfigType"/>
         <xs:element name="Ext" type="ExtType" minOccurs="0"/>
         <xs:element ref="ds:Signature" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="optional"/>
      <xs:attribute name="release" type="xs:integer" use="required"/>
      <xs:attribute name="date" type="xs:date" use="optional"/>
   </xs:complexType>
   <xs:complexType name="IdPConfigType">
      <xs:sequence>
         <xs:element name="DefaultIdP" type="IdPEntryType"/>
         <xs:element name="PartnerIdP" type="IdPEntryType"
            maxOccurs="unbounded" minOccurs="0" />
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="IdPEntryType">
      <xs:sequence>
         <xs:element name="ProviderID" type="entityIDType"/>
```

```
            <xs:element name="ProviderDisplayName" type="xs:string"
               minOccurs="0" />
            <xs:element name="SingleSignOnServiceURL" type="versionedEndPoint"
               minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="KeyInfo" type="ds:KeyInfoType" minOccurs="0"/>
            <xs:element name="MetadataConfKey" type="ds:KeyInfoType"
               minOccurs="0" />
            <xs:element name="Ext" type="ExtType" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
      <xs:complexType name="versionedEndPoint">
         <xs:simpleContent>
            <xs:extension base="xs:anyURI">
               <xs:attribute name="version" type="xs:anyURI"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
      <xs:simpleType name="entityIDType">
         <xs:restriction base="xs:anyURI">
            <xs:maxLength id="maxlengid" value="1024"/>
         </xs:restriction>
      </xs:simpleType>
      <xs:complexType name="ExtType">
         <xs:sequence>
            <xs:any maxOccurs="unbounded" namespace="##other"
               processContents="lax"/>
         </xs:sequence>
      </xs:complexType>
</xs:schema>
```

Some guidelines apply to the use of this schema.

- Document instances should be canonicalized using XML Exclusive Canonicalization ([XMLCanon]).
- Instances of this schema should only use `<LecProv>` as a root element.
- Document instances using this schema should be constructed such that other namespaces are not imported into the EC provisioning schema, unless absolutely necessary.
- The `<ProviderID>` may be used to acquire additional metadata about the provider specified in instances of this schema.
- An extension point (element `<Ext>`) is provided to allow elements from other namespaces to be included in this schema. Given the storage constraints of this environment, implementors should take care only to add extensions that are absolutely necessary.
- The `<DefaultIdP>` should be considered by the EC to be the identity provider that should be used whenever possible.
- The `<PartnerIdP>` elements should be listed in order of preference in document instances, and should be used whenever the `<DefaultIdP>` is not available.
- It should be noted that an instance of this schema may reach a considerable size if a number of identity providers are included in the instance. In particular, the key values supplied in the `<KeyInfo>` and `<MetadataConfKey>` elements may be quite sizeable.

  It is suggested that when supplying key value information for these elements, that the key value portion of the XML be hashed (using SHA-1 for example) and then encoded base64. This value would then be placed in the `<KeyName>` element of the `<KeyInfo>` element.

When the EC must compare this stored key value against that obtained from a provider, it may simply encode the received key appropriately (for example as `RSAKeyValue`), hash it, encode base64 and then compare to the value stored.

```
<RSAKeyValue>
   <Modulus>
      xA7SEU+e0yQH5rm9kbCDN9o3aPIo7HbP7tX6WOocLZAtNfyxSZDU16ksL6W
      jubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavzyBpVRgBUwUlV
      5foK5hhmbktQhyNdy/6LpQRhDUDsTvK+g9Ucj47es 9AQJ3U=
   </Modulus>
   <Exponent>AQAB</Exponent>
</RSAKeyValue>
```

Could be encoded as described above, to create the following key information in the document instance:

```
<KeyInfo>
   <KeyName>
      reyebww23rADSaddfDFSe34ncaksdcnsdlcnk=
   </KeyName>
</KeyInfo>
```

Additionally, there are the following guidelines for the usage of the `IdPEntry` type:

- `<ProviderId>` is the identifier that the EC must use to validate the IdP as a SAML provider.
- If `<MetadataConfKey>` is present, the `<ProviderID>` may be used to acquire additional metadata about the provider specified in instances of this schema using the well-known location mechanism. Fetched metadata instance documents should be signed by the key specified by this element.
- If `<SingleSignOnServiceURL>` is present, it denotes the URL that the EC must use when issuing `<AuthnRequest>` to the IdP. Several entries of this element can appear denoting different entry points listening for different protocol version messages.
- If `<KeyInfo>` is present, IdP authentication must be achieved following the contents of this element. If not present, IdP-authentication may be achieved following any other existing mechanism, such as TLS Root-CA-List based authentication.
- If `<ProviderDisplayName>`is present, the EC should display this value to user when interacting with this IdP.

6.3.1 Obtaining EC-Provisioning Document Instances Using a (U)SIM

It is recommended that when hosted by a (U)SIM, EC-applications (or the handset firmware) should obtain the EC-Provisioning document instances through simple file I/O. A single document instance will be hosted in a read-only file (denoted herein S-EF) updated unilaterally by the (U)SIM. The (U)SIM itself will obtain the EC-Provisioning information at "personalization time" and by the means described in Section 6.3.2.

The EC application should look for the existence of this file and retrieve its contents each time the application is launched. It should also poll the (U)SIM periodically for modification or leverage the existing "SIM-REFRESH" mechanism by which the (U)SIM notifies the handset of any modification of its contents.

It should be noted that from the EC-Application perspective, the authenticity and integrity of the EC-Provisioning document is guaranteed by the fact that it is obtained directly from the U(SIM). Therefore, EC applications are not required to apply any further integrity check. For "Smart-Phone" class devices, the handset firmware and hardware must ensure that data retrieved from this source has not been tampered with by any local process or potentially malicious component.

Beyond setting the S-EF file to be read-only, it is not necessary to enforce PIN protection or any other specific access control measure on the file.

6.3.2 Updating EC-Provisioning Document Instances Using a (U)SIM

Initially, the (U)SIM will obtain the EC-Provisioning document instance during its "personalization" at manufacturing time. Later on, it would be possible to update the information by using one of two methods described below.

Independently of the mechanism used to physically obtain the updated EC-Provisioning document instances, all configuration data will be source-authenticated and integrity checked by the (U)SIM. It is recommended that the existing SIM Application Toolkit (SAT) integrity and authentication mechanisms are used for that purpose.

6.3.2.1 SAT-based (U)SIM EC-Provisioning Document Update

The first and preferred method for updating the EC-Provisioning document leverages the SAT, using SMS or a different bearer, depending on specific handset support. It is recommended that if this method is used, a complementary SIM-REFRESH command be issued to signal to the EC application that the EC-Provisioning document instance has changed.

6.3.2.2 File-based (U)SIM EC-Provisioning Document Update

The second method of provisioning may be performed by the EC via input to a specific write-only input file (denoted herein I-EF). The format of this file will be determined by the (U)SIM provisioning authority, and it is for now out of scope for these guidelines. Once the EC application has finished writing to the I-EF, it should re-read the S-EF to check for an updated version of the EC-Provisioning document instance. Beyond setting up the I-EF file as write-only, it is not necessary to enforce PIN protection or any other specific access control measure on it.

The Provisioning-source entity will deliver the following to the EC application embedded in the appropriate application-level protocol flow (such as within a `<AuthnResponseEnvelope>` if the provisioning source is also an IdP):

```
<xs:element name="newLECProvInfo">
   <xs:complexType>
      <xs:complexContent>
         <xs:extension base="xs:base64">
            <xs:attribute name="release" type="xs:integer" use="required"/>
            <xs:attribute name="date" type="xs:date" use="optional"/>
         </xs:extention>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
```

These elements would be in a separate schema.

The provisioning information itself will be in a binary format that has been base64 encoded. After base64 decoding, the EC will write the contents into the I-EF. When the EC-application finishes writing to the I-EF, the SIM will issue a SIM-REFRESH command (if the handset platform supports this). Then the EC application should read the new provisioning information from the S-EF and check that its release version is the same as the one conveyed through the container. If the SIM-REFRESH command is not supported by the handset, the EC-application should know about this limitation and instead perform a poll for a change.

If after a reasonable amount of time the contents of the S-EF are not updated accordingly, the EC-application should assume that the operation failed and signal back to the provisioning source the condition whenever the opportunity arises with:

```
<xs:element name="newLECProvInfoFailed">
   <xs:complexType>
      <xs:simpleContent>
         <xs:attribute name="release" type="xs:integer" use="required"/>
         <xs:attribute name="date" type="xs:date" use="optional"/>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

These elements would be in a separate schema.

A potential candidate protocol message for hosting this could be the extension point of EC-to-IdP `<AuthnRequest>`. But we could also create a specific SOAP call. As we are just crafting this as guidelines, we don't have to be completely specific for now on how this should be implemented.

6.3.3 External standardization dependencies

(U)SIM file identifiers (S-EF and I-EF) will eventually be registered with ETSI.

6.4 SAML Authentication Context in Mobile Environments

The SAML Authentication Context Specification [SAMLAuthn] provides a schema for describing the context surrounding the authentication of an individual. Such contextual information may be required for an organization to decide whether a particular principal's authentication was rigorous enough for their purposes. For example, a payment made via credit card, authorized with the signature of the principal may be seen as more "trustworthy" because of the presence of the signature.

SAML provides a number of authentication contexts that are specific to the mobile environment:

- **MobileTwoFactorContract**: Use of this context would involve contract registration procedures for mobile subscribers and the protocols used to control access to digital mobile networks. The two authentication factors would be a physical device, normally containing a secret key and executing symmetric cryptography, plus a mechanism that verifies that the rightful user of the device is present, e.g. the input of a CHV (Card-Holder Verification) value such as a PIN or some biometric data. Requirements for PKI-based authentication, for access to SAML services or for securing transactions, can be met if private signing keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions present in mobile network operators' (MNO) tamper-resistant smart cards (commonly called SIM cards or USIM cards).

- **MobileOneFactorContract**: Use of this context would involve contract registration procedures for mobile subscribers and the protocols used to control access to digital mobile networks. The single authentication factor would be a physical device, normally containing a secret key and executing symmetric cryptography. Requirements for PKI-based authentication, for access to SAML services or for securing transactions, can be met if private keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

- **MobileTwoFactorUnregistered**:This context may be useful when a service other than that of the MNO wishes to link their customer ID to a mobile-supplied two-factor authentication service, by capturing mobile data such as a phone number or device id at enrollment. This context would involve the protocols used to control access to digital mobile networks, but with no subscriber registration procedure employed by the MNO. The two authentication factors would be a physical device, normally containing a secret key and executing symmetric cryptography, plus a mechanism that verifies that the rightful user of the device is present, e.g. the input of a CHV (Card-Holder Verification) value such as a PIN or some biometric data . Requirements for PKI-based authentication, for access to SAML services or for securing transactions, can be met if private signing keys are deployed (e.g. using WAP's WIM),

with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

- **MobileOneFactorUnregistered**: This context may be useful when a service other than that of the MNO wishes to link their customer ID to a mobile-supplied one-factor authentication service, by capturing mobile data such as a phone number or device id at enrollment. This context would involve the protocols used to control access to digital mobile networks, but with no subscriber registration procedure by MNO. The single authentication factor would be a physical device, normally containing a secret key and executing symmetric cryptography. Requirements for PKI-based authentication, for access to SAML services or for securing transactions, can be met if private keys are deployed (e.g. using WAP's WIM), with corresponding functionality in mobile devices. Best established practice in the mobile industry is to hold secret or private key material and execute cryptographic functions in MNOs' tamper-resistant smart cards (commonly called SIM cards or USIM cards)

The above-noted authentication contexts may be appropriate in different situations. For example, two-factor authentication either with or without a MNO contract (i.e. unregistered) might be required for authorization of high-value mobile payments, whereas one-factor unregistered authentication may be sufficient for low-value transactions, or access to a corporate network. It should be noted, that other information (such as the presence of keying information on the mobile device) will also factor in to the authentication context.

7 Privacy Principles

Federated authentication systems have the potential to either enhance or hinder an individual's privacy. By simplifying authentication, such systems will ensure that a user's personal data can be better protected, and not compromised by the typical cribs that users currently typically use for simplification (e.g. reuse of password, weak passwords, stored passwords, etc). Nevertheless, if inappropriately implemented, federated authentication systems have the potential to compromise a user's privacy. While SAML has defined normative mechanisms for enhancing privacy into its specifications, other aspects of federated authentication that cannot be normatively defined can still impact privacy. This section lists guidance for such aspects.

7.1 Privacy Principles for Authentication

In a sense, there is an inherent conflict between authentication and privacy. A common definition for privacy is 'the ability for individuals to control how information about themselves is collected and shared'. As authentication systems often require that some information about the authenticating entity be collected, for a user to opt out of this collection (as is their privacy 'right') implies an inability for them to subsequently use that particular authentication system and, presumably, access those resources being protected by that system. Consequently, for every authentication system, a user must weigh the advantages afforded by accepting the requirements (with respect to collection of personal information about themselves) of the system against the potential privacy risk inherent in any such personal information sharing.

Recently, a variety of organizations have drafted "best-practice" guidelines for creating authentication systems that will respect the privacy rights of those individuals who use the systems. For example, the Center for Democracy & Technology (CDT) explored the privacy issues associated with authentication systems:

> *New technologies for authentication make possible greater realization of the Internet's potential by making online transactions more seamless, tying together information on multiple devices, and enabling yet unimagined services. However, many authentication systems will collect and share personally identifiable information, creating privacy and security risks. To mitigate these risks, it is essential that authentication systems be designed to support effective privacy practices and offer individuals greater control over their personal information.*

The CDT Working Group drafted basic privacy principles [CDT] that should be considered in the design and implementation of authentication systems. These principles are:

- **Provide user control**: The informed consent of the individual should be obtained before information is used for enrollment, authentication and any subsequent uses.
- **Support a diversity of services**: Individuals should have a choice of authentication tools and providers in the marketplace. While convenient authentication mechanisms should be available, privacy is put at risk if individuals are forced to use one single identifier for various purposes.
- **Use individual authentication only when appropriate**: Authentication systems should be designed to authenticate individuals by use of identity only when such information is needed to complete the transaction. Individual identity need not and should not be a part of all forms of authentication.
- **Provide notice**: Individuals should be provided with a clear statement about the collection and use of information upon which to make informed decisions.
- **Minimize collection and storage**: Institutions deploying or using authentication systems should collect only the information necessary to complete the intended authentication function.

- **Provide accountability**: Authentication providers should be able to verify that they are complying with applicable privacy practices.

The CDT Working Group did not specifically address the privacy issues introduced by federated authentication. Their basic guidelines can be extended to address these new issues. 983

Similarly, the EU Article 29 Data Protection Working Party published guidance on online authentication systems [Art29]. The Working Party's conclusions are as follows:

- Both those who design and those who actually implement on-line authentication systems (authentication providers) bear responsibility for the data protection aspects, although at different levels. Websites making use of these schemes (service providers) also have their own responsibility in the process. It is advisable for the different players to have clear contractual agreements between them where the obligations of each party are made explicit.
- All possible efforts should be made to allow anonymous or pseudonymous use of online authentication systems. Where this would inhibit full functionality, the system should be built to require minimal information only for the authentication of the user and to give the user full control over decisions concerning additional information (such as profile data). This choice should exist both at the level of the authentication provider and of the service providers (the sites making use of the system).
- It is vital to provide adequate information to the users concerning the data protection implications of the system (controller identity, purposes, data collected, recipients and so on). This information should be provided in an easily accessible and user-friendly way, preferably through the collection form or via a prompt box that would automatically open on the screen of the user, and in all the languages in which the service is offered.
- When personal data are to be transferred to third countries, authentication providers should work with service providers who take all necessary measures to provide adequate protection or that put in place sufficient safeguards to ensure the protection of the personal data of the users of the system, by using contracts or binding corporate rules. This should be the general rule. If in particular cases consent is used as a basis for the transfer, sufficient information and choice should be given to the users. They should have the option to agree or not to the transfer on a case by case basis.
- The use of identifiers, whatever form they take, entails data protection risks. Full consideration should be given to all possible alternatives. If user identifiers are indispensable, the possibility of allowing the user to refresh the identifier should be considered.
- The adoption of software architecture that minimizes the centralization of personal data of the Internet users would be appreciated and encouraged as a means of increasing the fault-tolerance properties of the authentication system, and of avoiding the creation of high added-value databases owned and managed by a single company or by a small set of companies and organizations.
- Users should have an easy means to exercise their rights (including their right to opt-out) and to have all their data deleted if they decide to stop using an on-line authentication system. They should also be adequately informed about the procedure they should follow if they have inquiries or complaints.
- Security plays a fundamental role in this context. Organizational and technical measures that are appropriate to the risks at stake should be taken.

7.2 Privacy Principles for Federated Authentication

In this section, we interpret the basic privacy principles listed in the previous section for federated architectures.

7.2.1 Passive SSO

Federated authentication makes possible 'invisible' authentication, i.e. a principal can be logged in at a service provider without explicit action on their part but rather based on a previous authentication event performed at an identity provider. While a valuable usability mechanism, the possibility of a principal being logged in without their knowledge could pose a privacy concern if that principal was not adequately informed of, and consented to, the process. The principal might believe they are surfing anonymously at the SP while in fact they are not.

Authentication at a (non-proxying) IDP will always be overt and active, federated authentication at an SP can be either overt/active (i.e. they click on a 'Sign in at IDP' link or button to instigate the process) or passive. Passive SSO includes both when the SP automatically directs a principal to the IDP for authentication and when the IDP sends the principal to an SP along with an unsolicited authentication assertion.

As the Principal is unknown to the SP until after it has received an authentication assertion from the IDP, it will be unable to apply policy for passive SSO at anything other than a site-wide level (e.g. always notify users). Nevertheless, SPs can enable principal-specific passive SSO policy to be applied at the IDP through the inclusion of the passive SSO details in the `<AuthnRequest>`.

> **Note:** An IDP would be unable to determine the validity of the SP's claims with respect to passive SSO - this mode relies on the SP acting in good faith. In some situations it may be the case that the IDP is 'more trusted' by the principal than the SP. In these situations, the Principal's confidence that their policy with respect to passive SSO should be based on the trust they have in the SP, not the more trusted IDP.

- When instigated by the principal, an SP must include a value of `urn:oasis:names:tc:SAML:2.0:consent:current-explicit` in the `Consent` attribute on the `<AuthnRequest>` to indicate that the principal has explicitly given consent to the sending of the `<AuthnRequest>` message.
- When instigated by the SP, the SP must include a value of either `urn:oasis:names:tc:SAML:2.0:consent:current-implicit` or `urn:oasis:names:tc:SAML:2.0:consent:prior` in the `Consent` attribute on the `<AuthnRequest>`.
- After receiving an `<AuthnRequest>` with a value of either `urn:oasis:names:tc:SAML:2.0:consent:current-implicit` or `urn:oasis:names:tc:SAML:2.0:consent:prior` in the `Consent` attribute, an IDP must ensure that the relevant principal's policy allows this.
- When sending a Principal to an SP along with an unsolicited `<AuthnResponse>`, an IDP must ensure that the principal's policy with regard to passive SSO is satisfied.
- An IDP must provide principals the opportunity to specify their policy with regard to passive SSO. An IDP should allow Principal's to specify any of:
  - They consent to authorizing passive SSO.
  - They wish to be notified of passive SSO.
  - They wish to be prompted for authorization for passive SSO.
- An IDP may give the principal the ability to specify their preferences for passive SSO on a per SP basis.

Past experience with SSO systems has shown that systems that rely upon active SSO by requiring the user to instigate the SSO process can cause confusion (the user may provide any SP credentials they have to the IDP or vice versa) and hinder adoption. Additionally, a principal who has consented to federation has presumably done so in order to enjoy a more streamlined browsing experience, this value diminished if passive SSO is ruled out.

- An IDP should implement a default policy of 'allow passive SSO', this default modified by individual Principals.
- If the Principal sets their policy at the IDP such that they choose to be notified/prompted for authorization for passive SSO, the IDP should display the name of the relevant SP at which they will be authenticated.

In general, the advantages that the different mechanisms for principal control of the SSO process will need to be weighed against the issues increased control would present for usability. This decision should however be the principals to make and IDPs must provide simple and intuitive interfaces to assist the principal in asserting their choices.

7.2.2 Authentication Status

Passive SSO, even when consented to, creates the possibility of a principal being logged in at a provider without explicit action on their part. It will sometimes be the case that, through the available access to account details, it will be readily obvious to the principal that they are logged in at the SP. However, it will also be the case that sometimes the principal will be accessing generic resources that would provide no such clue. It may be the case that a principal's behavior will be different if they believe are surfing anonymously or pseudonymously rather than as a verified individual. Consequently, in general, but especially true for passive SSO, a principal should always be able to determine their current authentication status at both identity and service providers.

- Providers must provide a mechanism to allow a Principal to determine their current status. Such status should include the following information:
  - Their current authentication status.
  - Their current session history.
  - The authenticating IDP(s) (if an SP status display).
  - The chain of IDPs (if an SP status display).
  - The current authentication context.
  - The SP's to which an assertion has been sent (if an IDP status display).

7.2.3 Authentication Strength

SAML protocols allow an SP to indicate their preferred mechanism(s) that the IDP should use to authenticate the principal by specifying the relevant preferred authentication context(s). An IDP, in its response, indicates the actual relevant authentication context used. Different authentication technologies differ in the degree to which they bind the authenticating entity to the identity claims they are making. It is expected that SPs will request stronger authentication mechanisms for more sensitive resources.

The general principle that a choice of authentication technology should be commensurate with the value of the resource being accessed can be refined for federated authentication through the following guidelines:

- An SP should request non-anonymous authentication of a principal only when it is necessary to know the principal's real identity.
- An SP should request the minimally acceptable authentication context that its access control policies define as sufficient.

7.2.4 IDP Proxying

A proxying IDP mediates SSO between another IDP (that to which the principal actually authenticates) and an SP (which, for a variety of reasons, is unable to deal directly with the first IDP). The proxying IDP consumes the authentication assertion of the original IDP and then creates another for delivery on to the SP. As the principal's user agent will be temporarily sent to the proxying IDP as part of the normal SSO

protocol, the proxying IDP has the opportunity to establish an authenticated session for that principal and set a cookie on their agent to maintain session state.

Although the Principal will necessarily have had to federate their accounts at the authenticating and proxying IDPs (with the implied consent that the proxying IDP can consume authentication assertions issued by the authenticating IDP), as well as those at the proxying IDP and the SP (with the implied consent that the proxying IDP can perform authentication for the SP), this does not necessarily mean that they consent to proxying.

- An IDP must allow principals to specify their policy for proxying. An IDP should allow principals to specify any of:
  - consent to authorizing proxying.
  - a policy to be notified of proxying.
  - a policy to be prompted for authorization of proxying at run-time.
  - a policy to limit proxying by specifying maximum hops.
- An IDP may give the principal the ability to specify their preferences for proxying on a per provider basis.

In principle, a principal can specify policy at either the authenticating IDP or the proxying IDP. For the former, once the authenticating IDP determined who the principal was, it would do a policy lookup and determine whether or not it should proceed with proxying. For the latter, the proxying IDP would do a lookup after it had received the authentication assertion from the authenticating IDP. Consequently, a proxying IDP (like the original SP) would only be able to apply policy once it had sent a request to the authenticating IDP and received the appropriate response.

- An authenticating IDP must apply any applicable proxying policy after authenticating that principal. If the principal's policy forbids proxying, the authenticating IDP must not return the requested assertion to the proxying IDP.
- A proxying IDP must apply any applicable proxying policy after receiving the authentication assertion for the principal from the authenticating IDP. If the principal's policy forbids proxying, the proxying IDP must not return the requested assertion to the SP (or other proxying IDP).

From the principal's point of view, they are trying to access a resource at the SP and they log-in at the authenticating IDP in order to do so - being logged in at the proxying IDP through a session being established there is by product of the 'visible' part of the process. Such unforeseen and unrequested authentication at the proxying IDP poses a potential privacy concern if the principal were to subsequently visit the proxying IDP site and assume that they were surfing anonymously. It may however be necessary for the proxying IDP to establish a session for the Principal in order to maintain sufficient state to enable any subsequent SLO.

- If a proxying IDP sets a cookie in the principal's browser to maintain a session for subsequent SSO, the cookie should not be bound to the principal's local identity, i.e. it should be anonymous.
- If a proxying IDP does set a cookie bound to the principal's local identity in order to establish a session, they should provide mechanisms to allow that principal to determine their SSO status if they visit the site during the lifetime of the session.

8 References

The following are cited in the text of this document:

[Art29] Rodata, Stefano, eds. *Working Document on on-line authentication services*, ARTICLE 29 Data Protection Working Party http://europa.eu.int/comm/internal_market/privacy/docs/wpdocs/2003/wp68_en.pdf

[CDT] Schwartz, Ari. eds. *Privacy Principles for Authentication Systems*, Center for Democracy & Technology http://www.cdt.org/privacy/authentication/030513interim.pdf

[RFC1738] T. Berners-Lee et al., *Uniform Resource Locators (URL)* http://www.ietf.org/rfc/rfc1738.txt

[RFC2246] *The TLS Protocol Version 1.0*, http://www.ietf.org/rfcs/rfc2246.html.

[RFC2401] S. Kent et al., *Security Architecture for the Internet Protocol*, http://www.ietf.org/rfc/rfc2401.txt

[RFC2560] M. Myers et al., *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, http://www.ietf.org/rfc/rfc2560.txt

[RFC2616] R. Fielding et al, *Hypertext Transfer Protocol -- HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] J. Franks et al, *HTTP Authentication: Basic and Digest Access Authentication*, http://www.ietf.org/rfc/rfc2617.txt

[RFC2965] D. Kristol, et al., *HTTP State Management Mechanism*, http://www.ietf.org/rfc/rfc2965.txt

[SAMLAuthn] J. Kemp et al. *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS draft August 2004. Document ID sstc-saml-authn-context-2.0. http://www.oasis-open.org/committees/security/.

[SAMLBind] E. Maler et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS draft August 2004. Document ID sstc-saml-bindings-2.0. http://www.oasis-open.org/committees/security/.

[SAMLConform] P. Mishra et al., *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS draft August 2004. Document ID oasis-sstc-saml-conformance-2.0. http://www.oasis-open.org/committees/security/.

[SAMLCore] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS draft August 2004. Document ID oasis-sstc-saml-core-2.0. http://www.oasis-open.org/committees/security/.

[SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language (SAML)*. OASIS, August 2004. Document ID oasis-sstc-saml-glossary-2.0. http://www.oasis-open.org/committees/security/.

[SAMLMetadata] J. Moreh et al. *Metadata for the OASIS Security Assertion Markup Language (SAML)*. OASIS draft August 2004. Document ID oasis-sstc-saml-metadata-2.0 http://www.oasis-open.org/committees/security/.

[SAMLProf] F. Hirsh et al., *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0,* OASIS draft August 2004. Document ID sstc-saml-profiles-2.0 . http://www.oasis-open.org/committees/security/.

[SAMLSec] F. Hirsh et al, *Security and Privacy Consideration for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS draft August 2004. Document ID sstc-saml-sec-consider-2.0. http://www.oasis-open.org/committees/security/.

[SAMLTech] J. Hughes et al., Technical Overview of the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS draft August 2004. Document ID sstc.saml-tech-overview-?. http://www.oasis-open.org/committees/security/.

[SOAP] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium W3C Note (08 May 2000). http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SSL3] *The SSL Protocol Version 3.0*, http://wp.netscape.com/eng/ssl3/draft302.txt

[WML] *Wireless Markup Language Version 1.3 Specification*, Open Mobile Alliance, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[WSS] A. Nadalin et al., *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004),* OASIS standard March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0

[WSSKerb] A. Nadalin, *Web Services Security: Kerberos Token Profile 1.0*, OASIS draft April, 2004, http://www.oasis-open.org/committees/download.php/6394/oasis-xxxxx-wss-kerberos-token-profile-1.0.pdf

[WSSREL] T. DeMartini et al., Web Services Security: Rights Expression Language (REL) Token Profile, OASIS draft May 2004. http://www.oasis-open.org/committees/download.php/6906/oasis-____-wss-REL-token-profile-1.0-draft07-clean.pdf

[WSSSAML] P. Hallam-Backer et al., *Web Services Security: SAML Token Profile*, OASIS draft April 2004, http://www.oasis-open.org/committees/download.php/6271/WSS-SAML-10.pdf

[WSSUser] A. Nadalin et al., *Web Services Security: Username Token Profile 1.0,* OASIS standard March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

[WSSX509] P. Hallam-Baker et al., *Web Services Security: X.509 Certificate Token Profile*, OASIS standard March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf

[XKMS] P. Hallam-Baker, XML Key Management Specification (XKMS 2.0), W3C candidate recommendation April 2004. http://www.w3.org/TR/xkms2/

[XML] T. Bray, et al. *Extensible Markup Language (XML) 1.0 (Second Edition)*, Recommendation, World Wide Web Consortium http://www.w3.org/TR/2000/REC-xml-20001006

[XMLEnc] D. Eastlake et al., *XML Encryption Syntax and Processing*, http://www.w3.org/TR/xmlenc-core/, World Wide Web Consortium, December 2002.

[XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*, http://www.w3.org/TR/xmldsig-core/, World Wide Web Consortium.

Acknowledgments

The editors would like to acknowledge the contributions of the OASIS SSTC Technical Committee, whose voting members at the time of publication were:

- Conor P. Cahill, AOL, Inc.
- Hal Lockhart, BEA
- Gavenraj Sodhi, Computer Associates
- Tim Alsop, CyberSafe
- John Hughes, Entegrity Solutions
- Paul Madsen, Entrust (editor)
- Miguel Pallares, Ericsson
- Irving Reid, Hewlett-Packard Company
- Paula Austel, IBM
- Maryann Hondo, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Scott Cantor, Individual
- Bob Morgan, Individual
- Prateek Mishra, Netegrity (co-chair)
- Peter Davis, Neustar
- Frederick Hirsch, Nokia
- John Kemp, Nokia
- Nicholas Sauriol, Nortel
- Charles Knouse, Oblix
- Steve Anderson, OpenNetwork
- Darren Platt, Ping Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security (co-chair)
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Jeff Hodges, Sun Microsystems
- Eve Maler, Sun Microsystems
- Ron Monzillo, Sun Microsystems
- Mike Beach, The Boeing Company
- Greg Whitehead, Trustgenix

Revision History

Rev	Date	By Whom	What
00	27 Aug 2004	Charles Knouse	Initial draft.

Notices