SAML V2.0 Holder-of-Key Web Browser SSO Profile Version 1.0

Committee Specification 02

10 August 2010

Specification URIs:

This Version:

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cs-02.html

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cs-02.odt (Authoritative)

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cs-02.pdf

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.xsd

Previous Version:

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cd-03.html

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cd-03.odt (Authoritative)

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso-cd-03.pdf

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.xsd

Latest Version:

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.html

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.odt

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.pdf

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.xsd

Technical Committee:

OASIS Security Services TC

Chair(s):

Hal Lockhart, BEA Systems, Inc.
Thomas Hardjono , MIT

Editors:

Nate Klingenstein, Internet2
Tom Scavo, National Center for Supercomputing Applications (NCSA)

Related Work:

This specification is a cryptographically strong alternative to the SAML V2.0 Web Browser SSO Profile described in the SAML V2.0 Profiles specification [SAML2Prof].

Declared XML Namespace(s):

urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser

Abstract:

The SAML V2.0 Holder-of-Key Web Browser SSO Profile allows for transport of holder-of-key assertions by standard HTTP user agents with no modification of client software and maximum compatibility with existing deployments. The flow is similar to standard Web Browser SSO, but an X.509 certificate presented by the user agent via a TLS handshake supplies a key to be used in a holder-of-key assertion. Proof of possession of the private key corresponding to the public key in the certificate resulting from the TLS handshake strengthens the assurance of the resulting authentication context and protects against credential theft. Neither the identity provider nor the service provider is required to validate the certificate.

Status

This document was last revised or approved by the SSTC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC by using the "Send A Comment" button on the TC's web page at 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 IPR section of the TC web page (http://www.oasis-open.org/committees/security/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/security.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

In the scenario addressed by this profile, which is an alternate version of the SAML V2.0 Web Browser SSO Profile [SAML2Prof], a principal uses an HTTP user agent to access a web-based resource at a service provider. To do so, the user agent presents a holder-of-key SAML assertion acquired from its preferred identity provider.

The user may first acquire an authentication request from the service provider or a third party. The user agent transports the authentication request to the identity provider by making an HTTP request over TLS. An X.509 certificate supplied as a result of the TLS handshake supplies a public key that is associated with the principal. The identity provider authenticates the principal by any method of its choosing and then produces a response containing at least one assertion with holder-of-key subject confirmation and an authentication statement for the user agent to transport to the service provider. The assertion is then presented by the user agent to the service provider by making an HTTP request over TLS. An X.509 certificate supplied as a result of the TLS handshake proves possession of the private key matching the public key bound to the assertion. Finally, the service provider consumes the assertion to create a security context for the principal.

In what follows, a profile of the SAML Authentication Request Protocol [SAML2Core] is used in conjunction with an HTTP binding (section 2.5). It is assumed that the user wields an HTTP user agent, such as a standard web browser, capable of presenting client certificates in conjunction with a TLS handshake.

1.1 Notation

This specification uses normative text. The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119]:

> …they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)…

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

```
Listings of XML schemas appear like this.
```

```
Example code listings appear like this.
```

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
`md:`	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace defined in the SAML V2.0 metadata specification [SAML2Meta].
`ds:`	http://www.w3.org/2000/09/xmldsig#	This is the XML digital signature namespace defined in the XML Signature Syntax and Processing specification [XMLSig].
`hoksso:`	urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser	This is the web browser holder-of-key namespace defined by this document and its accompanying schema [HoKSSO-XSD].
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace

Prefix	XML Namespace	Comments
		defined in the SAML V2.0 core specification [SAML2Core].
`samlp:`	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace defined in the SAML V2.0 core specification [SAML2Core].
`xs:`	http://www.w3.org/2001/XMLSchema	This is the XML Schema namespace [Schema1].

This specification uses the following typographical conventions in text: `<SAMLElement>`, `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

1.2 Terminology

The term *TLS* as used in this specification refers to either the Secure Sockets Layer (SSL) Protocol 3.0 [SSL3] or any version of the Transport Layer Security (TLS) Protocol [RFC2246] [RFC4346] [RFC5246]. As used in this specification, the term *TLS* specifically does **not** refer to the SSL Protocol 2.0 [SSL2].

Unless otherwise noted, the term *X.509 certificate* refers to an X.509 client certificate as specified in the relevant version of the TLS protocol.

1.3 Normative References

[HoKSSO-XSD] OASIS Committee Specification 02, *Schema for SAML V2.0 Holder-of-Key Web Browser SSO Profile.* August 2010. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.xsd

[RFC2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt

[RFC2246] T. Dierks, C. Allen. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. http://www.ietf.org/rfc/rfc2246.txt

[RFC4346] T. Dierks, E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*. IETF RFC 4346, April 2006. http://www.ietf.org/rfc/rfc4346.txt

[RFC5246] T. Dierks, E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. IETF RFC 5246, August 2008. http://www.ietf.org/rfc/rfc5246.txt

[RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 5280, May 2008. http://www.ietf.org/rfc/rfc5280.txt

[SAML2Bind] OASIS Standard, *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[SAML2Core] OASIS Standard, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[SAML2HoKAP] OASIS Committee Specification 02, *SAML V2.0 Holder-of-Key Assertion Profile*. January 2010. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml2-holder-of-key-cs-02.pdf

[SAML2Meta] OASIS Standard, *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[SAML2Prof] OASIS Standard, *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[Schema1] H. S. Thompson et al. *XML Schema Part 1: Structures.* World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[SSL3] A. Freier, P. Karlton, P. Kocher. *The SSL Protocol Version 3.0*. Netscape Communications Corp., November 18, 1996. http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

[XMLSig] D. Eastlake, J. Reagle, D. Solo, F. Hirsch, T. Roessler. *XML Signature Syntax and Processing (Second Edition)*. World Wide Web Consortium Recommendation, 10 June 2008. http://www.w3.org/TR/xmldsig-core/

1.4 Non-normative References

[AIXCM] T. Moreau. *Auto Issued X.509 Certificate Mechanism (AIXCM).* IETF Internet-Draft, 6 August 2008. http://www.ietf.org/internet-drafts/draft-moreau-pkix-aixcm-00.txt

[IDPDisco] OASIS Committee Specification 01, *Identity Provider Discovery Service Protocol and Profile*., October 2007. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery-cs-01.pdf

[NISTEAuth] W. E. Burr et al. *Electronic Authentication Guideline*. National Institute of Standards and Technology, Draft Special Publication 800-63-1, 12 December 2008. http://csrc.nist.gov/publications/drafts/800-63-rev1/SP800-63-Rev1_Dec2008.pdf

[RFC3820] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile.* IETF RFC 3820, June 2004. http://www.ietf.org/rfc/rfc3820.txt

[SAML2Secure] OASIS Standard, *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf

[SAML2Simple] OASIS Committee Draft 04, *SAMLv2.0 HTTP POST "SimpleSign" Binding*. December 2008. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-binding-simplesign-cd-04.pdf

[SSL2] K. Hickman. *The SSL Protocol*. Netscape Communications Corp., February 9, 1995. http://www.mozilla.org/projects/security/pki/nss/ssl/draft02.html

[SSTC2NIST] "Suggested revisions to Draft NIST Special Publication 800-63-1 and the use of Assertions at Level-of-Assurance 4." OASIS SSTC, 4 November 2008. http://www.oasis-open.org/committees/download.php/29904/NIST-800-63-LOA-4-Letter-v2.pdf

2 Holder-of-Key Web Browser Profile

2.1 Required Information

Identification: `urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser`

Contact information: security-services-comment@lists.oasis-open.org

SAML Confirmation Method Identifiers: The SAML V2.0 “holder-of-key” confirmation method identifier, `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key`, is included in all assertions issued under this profile.

Description: Given below.

Updates: Provides an alternative to the SAML V2.0 Web Browser SSO Profile [SAML2Prof].

2.2 Background

This profile is designed to enhance the security of SAML assertion and message exchange without requiring modifications to client software. A holder-of-key SAML assertion is delivered to the service provider via an HTTP binding (section 2.5) over TLS. The user agent presents an X.509 certificate previously vetted by the identity provider, resulting in a strong association of the resulting security context with the intended user and elimination of numerous attacks (section 4).

Enhanced security is the primary benefit associated with the use of this profile. Under ordinary Web Browser SSO, there is a small chance that a bearer token will be stolen in transit, as described in [SAML2Secure]. Confirming that the presenter of the token is the intended subject through public key cryptography virtually eliminates this chance, improving the viability of SAML Web Browser SSO for sensitive applications.

Related to this, NIST has recently revised its E-Authentication Guideline [NISTEAuth], and in the revision, in response to a public comment from the SSTC [SSTC2NIST], NIST has clarified the use of "assertions" at NIST level-of-assurance 4. As a result of this revised E-Authentication Guideline, “holder-of-key assertions may be used” as level 4 security tokens provided certain requirements are met (section 10.3.2.4 of [NISTEAuth]). We believe that holder-of-key SAML assertions obtained via the SAML V2.0 Holder-of-Key Web Browser SSO Profile are cryptographically strong authentication tokens that meet the NIST requirements.

2.3 Profile Overview

Figure 1 illustrates the basic template for achieving Web Browser SSO under this profile. The following steps are described by the profile. Within an individual step, there may be one or more actual message exchanges depending on the binding used for that step and other deployment-specific behavior.

1. **HTTP Request to Service Provider** (section 2.6.1)

   The principal, via an HTTP user agent, makes an HTTP request for a secured resource at the service provider. at this step, the user agent may or may not present an X.509 certificate to the service provider in conjunction with a TLS handshake. In any event, the service provider determines that no security context exists and subsequently initiates Holder-of-Key Web Browser SSO.

2. **Service Provider Determines Identity Provider** (section 2.6.2)

   The service provider determines the principal's preferred identity provider by unspecified means.

3. **Service Provider Issues <samlp:AuthnRequest> to Identity Provider** (section 2.6.3)

   The service provider issues a <samlp:AuthnRequest> message to be delivered by the user agent to the identity provider. An HTTP binding is used (section 2.5) to transport the message to the identity provider through the user agent. The user agent presents the message to the identity provider in an HTTP request over TLS. In conjunction with TLS, the user agent presents an X.509 certificate to the identity provider as described in section 2.4.

4. **Identity Provider Identifies Principal and Verifies Key Possession** (section 2.6.4)

   The principal is identified by the identity provider at this step. The identity provider identifies the principal using any authentication method at its disposal while honoring any requirements imposed by the service provider in the <samlp:AuthnRequest> message. The identity provider must establish that the user agent holds the private key corresponding to the public key bound to the X.509 certificate and that the public key does in fact belong to the principal.

5. **Identity Provider Issues <samlp:Response> to Service Provider** (section 2.6.5)

   The identity provider issues a <samlp:Response> message to be delivered by the user agent to the service provider. The response either indicates an error or includes at least an authentication statement in a holder-of-key assertion. An HTTP binding is used (section 2.5) to transport the message to the service provider through the user agent. The user agent presents the message to the service provider in an HTTP request over TLS. As in step 3, the user agent presents an X.509 certificate to the service provider as described in section 2.4.

6. **Service Provider Grants or Denies Access to Principal** (section 2.6.6)

   The SAML response is consumed by the service provider who either responds to the principal's user agent by establishing a security context for the principal and returning the requested resource, or by returning an error.

Note that an identity provider can initiate this profile at step 5 by issuing a <samlp:Response> message to a service provider without the preceding steps. The user agent or a third party may also initiate this profile by submitting an unsigned request at step 3.

2.4 TLS Usage

As noted in the introduction, this profile is an alternative to ordinary SAML Web Browser SSO [SAML2Prof]. The primary difference between that profile and this Holder-of-Key Web Browser SSO Profile is that the principal MUST present an X.509 certificate and prove possession of the private key associated with the public key bound to the certificate. This leads to holder-of-key subject confirmation [SAML2HoKAP], a type of subject confirmation that is stronger than the bearer subject confirmation inherent in ordinary Web Browser SSO.

The user agent presents an X.509 certificate in conjunction with a TLS handshake. It is important to realize that the presented certificate need not be a trusted certificate (although this is certainly permitted). However, the certificate MUST be presented via TLS. This proves possession of the corresponding private key.

According to the TLS protocol, validation of the client certificate is optional. Likewise this Holder-of-Key Web Browser SSO Profile does not require TLS client authentication, which is strictly OPTIONAL (but see section 4.3). Moreover, the authentication method by which the identity provider identifies the principal is unspecified.

According to the TLS handshake protocol, if the TLS server can not validate the client certificate, the server may either continue the handshake or prematurely terminate the handshake by returning a fatal alert to the client. Moreover, if the TLS server chooses to send a fatal alert, it must immediately close the HTTP connection according to the TLS protocol. Clearly this is undesirable, so the TLS server MUST be

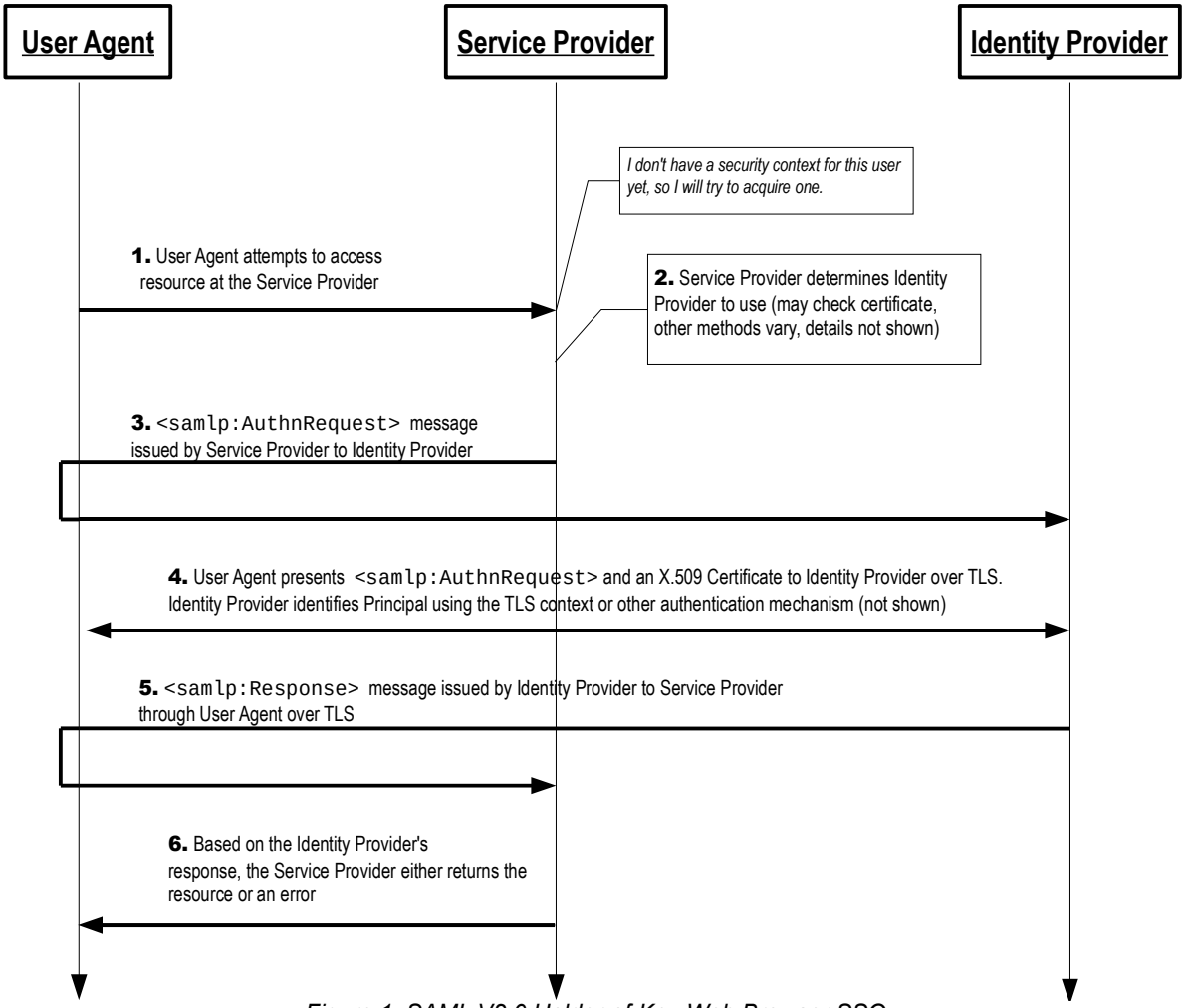

Figure 1: SAML V2.0 Holder-of-Key Web Browser SSO

configured to continue the TLS handshake to completion even in the presence of an untrusted client certificate. The method of doing so depends on the chosen TLS implementation and is therefore out of scope with respect to this profile.

In summary, the principal MUST present an X.509 certificate (via TLS) and prove possession of the private key at steps 3 and 5 (sections 2.6.3 and 2.6.5, resp.). However, the presentation of an X.509 certificate at step 1 (section 2.6.1) is strictly OPTIONAL.

At the conclusion of the TLS handshake, the identity provider (resp., the service provider) MUST be able to retrieve the X.509 certificate presented by the user agent at step 4 (resp., step 6). The consequences of a failure to do so is discussed in detail in section 2.6.4 (resp., section 2.6.6).

At either of steps 3 or 5 (or both), the identity provider or the service provider (resp.) MAY use the public key bound to the certificate or the TLS session key to create a security context for the principal. Also, at step 1, the service provider MAY use the public key bound to the certificate or the TLS session key to associate any subsequent exchange with the original request.

2.5 Choice of Binding

The identity provider and the service provider MUST use a browser-based HTTP binding to transmit the SAML protocol message to the other party. A SAML HTTP binding [SAML2Bind] MAY be used for this purpose:

1. HTTP Redirect
2. HTTP POST
3. HTTP Artifact

This profile does not preclude the use of other browser-based HTTP bindings (such as the SAML V2.0 SimpleSign binding [SAML2Simple]).

The identity provider and the service provider independently choose their preferred binding (subject to the other party's desire or ability to comply). The service provider chooses an HTTP binding to transmit the `<samlp:AuthnRequest>` message to the identity provider. Later, independent of the service provider's choice of binding, the identity provider chooses an HTTP binding to transmit the `<samlp:Response>` message to the service provider. The identity provider MUST NOT use the HTTP Redirect binding since the response typically exceeds the URL length permitted by most HTTP user agents.

If the service provider uses either the HTTP Redirect or HTTP POST binding, the `<samlp:AuthnRequest>` message is delivered directly to the identity provider at step 3 (section 2.6.3). If the service provider uses the HTTP Artifact binding, the identity provider uses the Artifact Resolution Profile [SAML2Prof] to make a callback to the service provider to retrieve the `<samlp:AuthnRequest>` message.

Similarly, if the identity provider uses the HTTP POST binding, the `<samlp:Response>` message is delivered directly to the service provider at step 5 (section 2.6.5). If the identity provider uses the HTTP Artifact binding, the service provider uses the Artifact Resolution Profile to make a callback to the identity provider to retrieve the `<samlp:Response>` message.

2.6 Profile Description

The SAML V2.0 Holder-of-Key Web Browser SSO Profile is a profile of the SAML V2.0 Authentication Request Protocol [SAML2Core]. Where this Holder-of-Key Web Browser SSO specification conflicts with Core, the former takes precedence.

If the request is initiated by the service provider, begin with section 2.6.1. If the request is initiated by the user agent or a third party, begin with section 2.6.4. If the identity provider issues a response without a corresponding request, begin with section 2.6.5. The descriptions refer to a single sign-on service and assertion consumer service in accordance with their use described in section 4.1.3 of [SAML2Prof]. Processing rules for all messages are specified in section 2.7 of this profile.

2.6.1 HTTP Request to Service Provider

The profile may be initiated by an arbitrary HTTP request to the service provider. The service provider is free to use any means it wishes to associate the subsequent interactions with the original request. For example, each of the SAML HTTP bindings discussed in section 2.5 provides a `RelayState` mechanism that the service provider MAY use to associate any subsequent exchange with the original request.

2.6.2 Service Provider Determines Identity Provider

The service provider determines the principal's preferred identity provider by any means at its disposal, including but not limited to the SAML V2.0 Identity Provider Discovery Profile [SAML2Prof] or the Identity

Provider Discovery Service Protocol and Profile [IDPDisco]. If the user agent presents an X.509 certificate at the previous step, the service provider MAY use the X.509 certificate as a means of discovery. Use of the X.509 certificate in this way is out of scope. However, see section 4.2 for relevant discussion.

2.6.3 Service Provider issues <samlp:AuthnRequest> to Identity Provider

Once an identity provider has been selected, the location of the single sign-on service to which to send a `<samlp:AuthnRequest>` message is determined based on the SAML binding chosen by the service provider (section 2.5). Metadata as described in section 2.8 MAY be used for this purpose. Following the HTTP request by the user agent in section 2.6.1, an HTTP response is returned containing a `<samlp:AuthnRequest>` message or an artifact, depending on the SAML binding used, to be delivered to the identity provider's single sign-on service.

Profile-specific rules for the contents of the `<samlp:AuthnRequest>` element are given in section 2.7.1.

2.6.4 Identity Provider Identifies Principal and Verifies Key Possession

The identity provider must perform two functions in this step: identify the principal presenting the `<samlp:AuthnRequest>` message and verify that the principal possesses the private key associated with the public key bound to the presented X.509 certificate. The identity provider subsequently binds X.509 data from the certificate (or the certificate itself) to a `<saml:SubjectConfirmation>` element.

The identity provider MUST establish the identity of the principal (unless it will return an error) prior to the issuance of the `<samlp:Response>` message. If the `ForceAuthn` attribute on the `<samlp:AuthnRequest>` element is present and true, the identity provider MUST freshly establish this identity rather than relying on any existing session it may have with the principal. Otherwise, and in all other respects, the identity provider may use any means to authenticate the user agent, subject to any requirements called out in the `<samlp:AuthnRequest>` message. In particular, the identity provider MAY use TLS client authentication to identify the principal. That is, the identity provider MAY validate the presented X.509 certificate as described in [RFC5280], but this is by no means a requirement. See section 2.4 for details.

As described in section 2.4, it is REQUIRED that the `<samlp:AuthnRequest>` message be presented to the identity provider via an HTTP request over TLS that supplies the identity provider with an X.509 certificate and establishes the user agent's possession of the corresponding private key. The certificate resulting from the TLS handshake MUST be used to construct any holder-of-key `<saml:SubjectConfirmation>` elements in the issued `<samlp:Response>` element.

Any holder-of-key `<saml:SubjectConfirmation>` elements included in the response MUST conform to the SAML V2.0 Holder-of-Key Assertion Profile [SAML2HoKAP]. See section 2.7.3 for consequences of this dependency. In addition, note well that the Holder-of-Key Assertion Profile requires that the X.509 certificate obtained as a result of the TLS handshake MUST be known to be associated with the principal (see section 2.4 of [SAML2HoKAP]). Precisely how the identity provider satisfies this requirement is out of scope, but see section 4.3.

If the principal is unable to prove possession of the private key corresponding to the public key in the certificate (via TLS), or the identity provider is unable to retrieve the X.509 certificate resulting from the TLS handshake, the identity provider MUST return an error. Otherwise, the identity provider processes the request following the rules specified in section 2.7.2.

2.6.5 Identity Provider Issues <samlp:Response> to Service Provider

Depending on the SAML binding used (section 2.5), the identity provider returns an HTTP response to the user agent containing a `<samlp:Response>` message or an artifact, to be delivered to the service

provider's assertion consumer service. Profile-specific rules regarding the contents of the <samlp:Response> element are included in section 2.7.3.

2.6.6 Service Provider Grants or Denies Access to Principal

As specified in section 2.4, the HTTP request that transports the response issued at the previous step MUST be made over TLS. This supplies proof of possession of the private key and an X.509 certificate to be checked against the X.509 data bound to the assertion's <saml:SubjectConfirmation> element. The TLS protocol also maintains the confidentiality and integrity of the message exchange.

If the principal is unable to prove possession of the private key corresponding to the public key in the certificate (via TLS), or the service provider is unable to retrieve the X.509 certificate resulting from the TLS handshake, the subject is not confirmed and the service provider SHOULD NOT create a security context for the principal.

Otherwise, the service provider MUST process the <samlp:Response> message and any enclosed <saml:Assertion> elements as described in [SAML2Core] and section 2.7.4 below. Any subsequent use of the <saml:Assertion> elements is at the discretion of the service provider and other relying parties, subject to any restrictions on use contained within the assertions themselves or previously established out-of-band policy governing interactions between the identity provider and the service provider.

To complete the profile, the service provider creates a security context for the user. The service provider MAY establish a security context with the user agent using any session mechanism it chooses. In particular, the public key or the TLS session key MAY be used to create the security context as discussed in section 2.4.

2.7 Use of Authentication Request Protocol

This profile builds upon the Authentication Request Protocol [SAML2Core]. In the nomenclature of actors enumerated in section 3.4 of Core, the service provider is the request issuer and the relying party, the user agent is the attesting entity and the presenter, and the principal is the requested subject. There may be additional relying parties at the discretion of the identity provider.

2.7.1 <samlp:AuthnRequest> Usage

A service provider MAY include any <samlp:AuthnRequest> message content as specified in [SAML2Core]. Additionally, the request MUST conform to the following rules:

- The <saml:Issuer> element MUST be present and MUST contain the unique identifier of the requesting service provider. The Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- The <samlp:AuthnRequest> message MAY be signed. The choice of signing method is a joint policy decision between the identity provider and the service provider.
- If the request message is signed, the service provider SHOULD include the AssertionConsumerServiceURL and AssertionConsumerServiceIndex attributes on the <samlp:AuthnRequest> element. Doing so often makes it easier for the identity provider to process the request.

2.7.2 <samlp:AuthnRequest> Message Processing Rules

The identity provider MUST follow all processing rules specified in [SAML2Core]. If the identity provider cannot or will not satisfy the request, it MUST respond with an error containing one or more error status codes.

If the `<samlp:AuthnRequest>` element is signed, and the signature can be successfully verified, the identity provider MAY (subject to policy) choose to accept the content of the request message without further processing. In particular, the identity provider MAY accept the values of the `AssertionConsumerServiceURL` or `AssertionConsumerServiceIndex` attributes on the `<samlp:AuthnRequest>` element (if any) without further processing.

If the `<samlp:AuthnRequest>` element is not signed, the identity provider MUST verify the content of the request message by some out-of-band means. In particular, the identity provider MUST verify that the values of the `AssertionConsumerServiceURL` or `AssertionConsumerServiceIndex` attributes on the `<samlp:AuthnRequest>` element (if any) belong to the target service provider.

If the `AssertionConsumerServiceURL` and `AssertionConsumerServiceIndex` attributes on the `<samlp:AuthnRequest>` element are absent, the identity provider determines the endpoint location of the assertion consumer service that will consume the response. The identity provider MUST be certain that the chosen endpoint location does in fact belong to the target service provider.

Even if the `<samlp:AuthnRequest>` element is signed, the identity provider MAY (subject to policy) choose to verify the request content by some out-of-band means. In all cases, the method by which the identity provider verifies the request content is unspecified. For instance, SAML metadata MAY be used for this purpose as described in section 2.8.

2.7.3 <samlp:Response> Usage

If the identity provider wishes to return an error in response to a request, it MUST NOT include any assertions in the `<samlp:Response>` message. Otherwise, the `<samlp:Response>` element MUST conform to the following rules:

- The `<saml:Issuer>` element of the `<samlp:Response>` element MAY be omitted, but if present it MUST contain the unique identifier of the issuing identity provider. The `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
- The response MUST contain at least one `<saml:Assertion>` element. Each assertion's `<saml:Issuer>` element MUST contain the unique identifier of the issuing identity provider, and the `Format` attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`.
- The `<saml:Subject>` element of every assertion returned by the identity provider MUST refer to the authenticated principal. Any holder-of-key assertions issued by the identity provider MUST fully conform to the SAML V2.0 Holder-of-Key Assertion Profile [SAML2HoKAP].
- Any `<saml:Subject>` elements in the response MUST strongly match the `<saml:Subject>` element in the `<samlp:AuthnRequest>` element (if any) as required by [SAML2Core]. If the `<samlp:AuthnRequest>` element contains an explicit `<saml:SubjectConfirmation>` element and the identity provider is unable to produce a strongly matching `<saml:Subject>` element for any reason, the identity provider MUST return an error.
- If the `<samlp:AuthnRequest>` element does not include a `<saml:Subject>` element, or the `<saml:Subject>` element in the request does not contain a `<saml:SubjectConfirmation>` element, every holder-of-key assertion in the response MUST contain a

<saml:SubjectConfirmation> element containing a <ds:X509Certificate> element. Other X.509 data MAY be included in additional child elements of the <ds:X509Data> element as specified in [SAML2HoKAP].

- Additional <saml:SubjectConfirmation> elements MAY be included in any assertion, though deployers should be aware of the implications of allowing weaker confirmation as the processing as defined in section 2.4.1.1 of [SAML2Core] is effectively satisfy-any. See section 3 for related considerations.
- Any assertion issued for consumption under this profile MUST contain a <saml:AudienceRestriction> element including the service provider's unique identifier in its <saml:Audience> element. Other conditions as defined in section 2.5 of [SAML2Core] (and other <saml:Audience> elements) MAY be included as requested by the service provider or at the discretion of the identity provider. All such conditions MUST be understood by and accepted by the service provider in order for the assertion to be considered valid.
- The set of one or more holder-of-key assertions MUST contain at least one <saml:AuthnStatement> element that reflects the authentication of the principal to the identity provider. Additional statements MAY be included in a holder-of-key assertion at the discretion of the identity provider.
- If the identity provider supports the Single Logout Profile [SAML2Prof], a <saml:AuthnStatement> element issued for consumption using this profile MUST include a SessionIndex attribute to enable per-session logout requests by the service provider.

As indicated above, the identity provider MUST issue at least one <saml:AuthnStatement> element. The identity provider typically issues exactly one such element but MAY issue multiple <saml:AuthnStatement> elements (in multiple assertions) if the service provider requires multiple assertions for various purposes.

If the identity provider issues multiple <saml:AuthnStatement> elements, the values of the IssueInstant attributes and the content of the <saml:SubjectLocality> elements MUST be identical across the <saml:AuthnStatement> elements. The content of the <saml:AuthnContext> elements MAY vary across the <saml:AuthnStatement> elements, presumably because the consumers of the various assertions have different requirements with respect to authentication context.

If the SAML HTTP POST binding (or a derivative of HTTP POST such as the SAML V2.0 SimpleSign binding [SAML2Simple]) is used to deliver the <samlp:Response> message to the service provider, every assertion in the response MUST be protected by digital signature. This can be accomplished either by signing each individual <saml:Assertion> element or by signing the <samlp:Response> element (or both).

2.7.4 <samlp:Response> Message Processing Rules

Regardless of the SAML binding used, the service provider MUST do the following:

- Verify any signatures present on the assertion(s) and/or the response.
- Verify that any assertions relied upon are valid according to processing rules in [SAML2Core].
- Using the X.509 certificate resulting from the TLS handshake, any holder-of-key assertions in the response MUST be confirmed in accordance with the SAML V2.0 Holder-of-Key Assertion Profile [SAML2HoKAP].

Any assertion that is not valid, or whose subject confirmation requirements cannot be met, SHOULD be discarded and SHOULD NOT be used to establish a security context for the principal.

If the response contains multiple assertions with multiple <saml:AuthnStatement> elements, the service provider MAY consume any one of them at its discretion. How the service provider makes this decision is unspecified.

2.7.5 Artifact-Specific <samlp:Response> Message Processing Rules

If the HTTP Artifact binding (section 2.5) is used to deliver the <samlp:Response> message to the service provider, the dereferencing of the artifact using the Artifact Resolution Profile [SAML2Prof] MUST be mutually authenticated, integrity protected, and confidential. Mutually authenticated TLS or message signatures MAY be used to authenticate the parties and protect the messages.

The identity provider MUST ensure that only the service provider to whom the <samlp:Response> message has been issued is given the message as the result of a <samlp:ArtifactResolve> request. To partially satisfy this requirement, the identity provider MAY encrypt the assertions in the response.

2.8 Use of Metadata

[SAML2Meta] defines metadata elements that describe supported bindings and endpoint locations for SAML entities. However, the metadata specification offers no way to distinguish the profile supported by an endpoint. A boolean flag extension is not sufficient to signal use of this profile because SAML implementations that don't implement this profile would ignore this optional attribute. As a result, an implementation could send users to an inappropriate endpoint, potentially impacting interoperability and the user experience.

Rather than define new endpoint elements, this profile specifies the use of the Binding attribute to disambiguate between this Holder-of-Key Web Browser SSO Profile and the original Web Browser SSO Profile. The URI of the actual binding is instead placed into an extension attribute on the same endpoint element. The combined information is sufficient to distinguish the correct profile and binding when making a request to an endpoint.

All <md:SingleSignOnService> endpoints and all <md:AssertionConsumerService> endpoints to be used exclusively with this profile MUST have a Binding attribute of:

```
urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser
```

If an endpoint calls out the above Binding attribute value, it MUST also include the extension attribute hoksso:ProtocolBinding as described below. The XML attribute hoksso:ProtocolBinding contains the identifier of the desired protocol binding.

The following schema fragment defines the hoksso:ProtocolBinding attribute [HoKSSO-XSD]:

```
<xs:attribute name="ProtocolBinding" type="anyURI" use="optional"/>
```

An example of a conforming <md:SingleSignOnService> element is as follows:

```
<md:SingleSignOnService
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:hoksso="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser"
  hoksso:ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
  Binding="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser"
  Location="https://your-idp.example.org/some/path"/>
```

Similarly, an example of a conforming <md:AssertionConsumerService> element is as follows:

```
<md:AssertionConsumerService index="1" isDefault="true"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:hoksso="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser"
  hoksso:ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
  Binding="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser"
  Location="https://your-sp.example.org/some/path"/>
```

3 Compatibility

Like the SAML V2.0 Web Browser SSO Profile [SAML2Prof], this Holder-of-Key Web Browser SSO Profile is a profile of the SAML V2.0 Authentication Request Protocol [SAML2Core]. The primary difference between the original Web Browser SSO Profile and this Holder-of-Key Web Browser SSO Profile is the mandate for holder-of-key subject confirmation, made possible by the user agent's ability to present an X.509 certificate in conjunction with a TLS handshake. Although the SAML V2.0 Holder-of-Key Web Browser SSO Profile is technically compatible with the original Web Browser SSO Profile, it is RECOMMENDED that separate endpoints be used to ensure all processing is performed in accordance with each profile's requirements and to avoid any negative impact on the user experience.

The SAML V2.0 Holder-of-Key Web Browser SSO Profile does not preclude the addition of bearer `<saml:SubjectConfirmation>` elements in conforming assertions. This peculiar combination of `<saml:SubjectConfirmation>` elements is permitted since it is believed that carefully crafted deployments and use cases may find it useful. However, such hybrid assertions must be issued only after due deliberation and care. Technically, an assertion containing both bearer and holder-of-key `<saml:SubjectConfirmation>` elements may be accepted as valid with no proof of possession of the private key, reintroducing attacks such as man-in-the-middle and replay. Such assertions require security precautions appropriate for standard bearer assertions as described in section 7.1.1 of [SAML2Secure].

4 Security and Privacy Considerations

Assertions issued under the Holder-of-Key Web Browser SSO Profile have different security and privacy characteristics than the bearer assertions used in the original Web Browser SSO Profile (see section 3). as specified, holder-of-key subject confirmation minimizes the potential for assertion theft and virtually eliminates man-in-the-middle attacks. Potential replay attacks that would otherwise require the tracking and checking of assertion `ID` attributes are also prevented by holder-of-key subject confirmation.

Since the content of a `<ds:X509Certificate>` element is simplest to produce and consume [SAML2HoKAP], deployments are encouraged to use the `<ds:X509Certificate>` element whenever possible. If, on the other hand, the service provider asks for specific X.509 data (other than the default `<ds:X509Certificate>` element), the identity provider is obliged to comply. In this case, however, the service provider will have already decoded and parsed the ASN.1-encoded certificate. It is likely, therefore, that the identity provider will be able to do the same. Thus the ability of each party to ASN.1-decode the certificate (always a concern when dealing with X.509 certificates from unknown issuers) is reasonably assured. (See section 2.6.1 of [SAML2HoKAP] for more information about ASN.1 encodings.)

Like the original Web Browser SSO Profile, this profile specifies that the `<samlp:AuthnRequest>` element MAY be signed, whereas Core specifies that the `<samlp:AuthnRequest>` element (and protocol requests in general) SHOULD be signed. Unlike the Web Browser SSO Profile, however, the identity provider MAY (subject to policy) accept the content of a signed request message without further processing, that is, without resorting to some out-of-band means of verification. This gives deployments more flexibility than what is allowed in the original Web Browser SSO Profile, especially if the presented X.509 certificate is signed by a trusted issuer.

4.1 X.509 Certificate Usage

As suggested in section 1.2, any client certificate compatible with the TLS protocol can be used by this profile. In particular, the use of self-signed certificates is not precluded. However, self-signed certificates should be used with care since it is well known that their use may break some implementations. For maximum interoperability, deployers are encouraged to use standard X.509 end-entity certificates [RFC5280] whenever possible. For those deployments that wish to avoid or do not require an X.509-based public key infrastructure (PKI), yet wish to maintain interoperability, note that so-called "meaningless X.509 certificates" [AIXCM] satisfy the formal requirements of X.509 end-entity certificates without belaboring the assumption of an underlying trust model.

As a hypothetical example, suppose the user (or a browser plug-in operating on behalf of the user) issues an X.509 proxy certificate [RFC3820] signed by a “meaningless end-entity credential,” that is, an X.509 credential whose public key certificate is signed by an untrusted CA such as the inherently untrusted Meaningless CA [AIXCM]. Such a proxy certificate is completely usable by this profile since the focus is on the public-private key pair, not the trustworthiness of the certificate issuer.

As a further by-product of using X.509 certificates, as discussed in section 2.4, a security context resulting from an exchange conforming to the Holder-of-Key Web Browser SSO Profile can be keyed using the public key bound to the certificate or the TLS session key. Application-layer sessions, such as those maintained by cookies, are often poorly protected by user agents, allowing for theft of the session and impersonation of the user. A session based on the public key or the TLS session key has no such limitations, however.

4.1.1 Privacy Issues

In terms of privacy, there may be limitations on the degree to which users can remain anonymous under this profile since an X.509 certificate is presented to the service provider. An X.509 certificate typically contains a globally unique distinguished name for the subject often containing personally identifying information. Additional information about the subject may be implicitly revealed through other fields or

extensions in the certificate. Furthermore, unless a new key pair is subsequently issued, the public key in the presented certificate is a de-facto persistent identifier, as discussed in [SAML2Secure].

4.2 Identity Provider Discovery

If the user accesses the service provider first, and presents an X.509 certificate to the service provider, discovery of the user's identity provider may be performed by examining fields or extensions within the presented certificate. For instance, if the user presents an X.509 certificate in conjunction with the initial request as described in section 2.6.1, the service provider may decode and parse the presented certificate and use the X.509 subject distinguished name or other field or extension in the certificate to determine the principal's preferred identity provider and/or single sign-on service endpoint. Such use of the X.509 certificate is beyond the scope of this specification, however.

As a specific example how this might be accomplished, suppose that the proxy certificate of the hypothetical example in section 4.1 contains a self-issued SAML attribute assertion bound to a non-critical X.509 certificate extension (which implies a v3 certificate, by the way, a basic requirement called out in the TLS protocol). Suppose further that the X.509-bound SAML token contains the following self-asserted attribute:

```
<saml:Attribute
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
  Name="urn:oasis:names:tc:SAML:2.0:nameid-format:entity"
  FriendlyName="entityID">
  <saml:AttributeValue>
    https://idp.example.org/saml
  </saml:AttributeValue>
</saml:Attribute>
```

Such an attribute could be used for identity provider discovery by the service provider at step 2.

4.3 TLS Client Authentication

The identity provider's requirements for user authentication and keying material as described in section 2.6.4 can be simultaneously addressed by validating the presented X.509 certificate as described in [RFC5280]. This is not mandatory, however, unless such an authentication context is specifically requested by the service provider. Note that phishing is virtually eliminated in the presence of X.509 client authentication, as there are greater challenges and no benefits to tricking the user into authenticating with a legitimate X.509 credential to a fraudulent party.

This profile offers potential usability benefits as well. If a certificate is used for principal authentication, there is no need for the user to further confirm its identity, and potentially no user interaction is required.

4.4 SAML vs. X.509 PKI

The SAML V2.0 Holder-of-Key Web Browser SSO profile realizes the benefits of a standard TLS session in which both parties exchange X.509 certificates. These benefits include TLS server authentication, transport-level data integrity and confidentiality, and most importantly, client-side proof of possession of the private key corresponding to the public key bound to the presented X.509 client certificate. In the case of the (untrusted) client certificate, the focus is on the proof of possession step. The fact that the client certificate is an untrusted certificate is actually an advantage since it avoids the difficulty of an X.509-based public key infrastructure (PKI).

This profile offers meaningful advantages over traditional X.509-based PKI. For instance, there is no requirement for a mutually trusted root certification authority (CA), distributed OCSP or CRL-based revocation lists, or X.509 certificate path validation (particularly at the SP). Moreover, not all participants in the SSO exchange need leverage the presented X.509 certificate to realize the benefits of this profile.

710 Furthermore, the presented X.509 certificate can be customized for each transaction, including fresh
711 attributes and appropriate revelation of principal identity as required.

712 4.4.1 An Illustration

713 As described in section 2.7.2, if the service provider signs the request with a trusted key, the identity
714 provider MAY accept the content of the `<samlp:AuthnRequest>` element without further processing. If
715 the identity provider and the service provider share a common X.509-based PKI, request signing makes
716 sense since everything the identity provider needs to know to formulate the response may be included in
717 the signed request. In the absence of such a PKI, signing serves little or no purpose. Indeed, a SAML-
718 based PKI based on trusted SAML metadata makes request signing unnecessary. Since a service
719 provider that signs requests must mitigate the threat of key theft, and since such a service provider is
720 more susceptible to denial-of-service attacks, infrastructure based on SAML metadata is preferred.

5 Conformance

All parties, including the identity provider, the service provider, and the HTTP user agent, MUST conform to section 2.4. In particular, the user agent MUST have the ability to present an X.509 certificate in conjunction with a TLS handshake.

The identity provider and the service provider MUST support the HTTP POST and HTTP Redirect bindings as discussed in section 2.5. Other binding support provided by the two parties is strictly OPTIONAL. In particular, support for the HTTP Artifact binding is OPTIONAL.

5.0.1 Identity Provider Conformance

In addition to the relevant requirements in section 5 above, an identity provider that conforms to this profile MUST adhere to the normative text in sections 2.6.4, 2.6.5, 2.7.2, and 2.7.3, and the relevant portions of section 2.7.5. If the identity provider uses SAML metadata, it MUST also conform to section 2.8 of this profile.

In addition to the above requirements, a conforming identity provider MUST meet the conformance requirements of the SAML V2.0 Holder-of-Key Assertion Profile [SAML2HoKAP].

5.0.2 Service Provider Conformance

In addition to the relevant requirements in section 5 above, a service provider that conforms to this profile MUST adhere to the normative text in sections 2.6.1, 2.6.2, 2.6.3, 2.6.6, 2.7.1, and 2.7.4, and the relevant portions of section 2.7.5. If the service provider uses SAML metadata, it MUST also conform to section 2.8 of this profile.

In addition to the above requirements, a conforming service provider MUST meet the conformance requirements of the SAML V2.0 Holder-of-Key Assertion Profile [SAML2HoKAP].

Appendix A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS Security Services (SAML) Technical Committee, whose voting members at the time of publication were:

- John Bradley, Individual
- Scott Cantor, Internet2
- Duane DeCouteau, Veterans Health Administration
- Christian Guenther, Nokia Siemens Networks GmbH & Co.
- Thomas Hardjono, M.I.T.
- Frederick Hirsch, Nokia Corporation
- Ari Kermaier, Oracle Corporation
- Nathan Klingenstein, Internet2
- Hal Lockhart, Oracle Corporation
- Paul Madsen, NTT Corporation
- Kyle Meadors, Drummond Group Inc.
- Bob Morgan, Internet2
- Thinh Nguyenphu, Nokia Siemens Networks GmbH & Co.
- Rob Philpott, EMC Corporation
- Anil Saldhana, Red Hat
- Tom Scavo, National Center for Supercomputing Applications
- Kent Spaulding, Skyworth TTG Holdings Limited
- David Staggs, Veterans Health Administration
- Emily Xu, Sun Microsystems

In addition, the editors would like to thank the National Institute of Informatics (Japan) and the UPKI initiative for their support of this work.

The editors would also like to acknowledge the following contributors:

- Scott Cantor, Internet2 (United States)
- Paul Friedrichs, Defense Information Services Agency (United States)
- Patrick Harding, Ping Identity Corporation (United States)
- Enrique de la Hoz, University of Alcala de Henares (Spain)
- Toshiyuki Kataoka, National Institute of Informatics (Japan)
- Chad La Joie, SWITCH (Switzerland)
- Diego Lopez, RedIRIS (Spain)
- David Waite, Ping Identity Corporation (United States)
- Peter Sylvester, EdelWeb (France)
- Marc Stern, Approach Belgium

Appendix B. Revision History

Document ID	Date	Committer	Comment
sstc-saml-holder-of-key-browser-sso-draft-1	27 Feb 2008	N. Klingenstein	Initial draft
sstc-saml-holder-of-key-browser-sso-draft-2	21 Apr 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-3	17 Jun 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-4	22 Jun 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-5	4 Aug 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-6	26 Aug 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-7	23 Sep 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-8	2 Nov 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-9	11 Nov 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-10	12 Dec 2008	N. Klingenstein	
sstc-saml-holder-of-key-browser-sso-draft-11	11 Jan 2009	T. Scavo	Technical editing and refactoring
sstc-saml-holder-of-key-browser-sso-cd-01	9 Mar 2009	T. Scavo	Committee Draft 01
sstc-saml-holder-of-key-browser-sso-draft-12	14 Jun 2009	T. Scavo	Response to Public Comments
sstc-saml-holder-of-key-browser-sso-cd-02	5 Jul 2009	T. Scavo	Committee Draft 02
sstc-saml-holder-of-key-browser-sso-cs-01	29 Jul 2009	tc-admin	Committee Specification 01
sstc-saml-holder-of-key-browser-sso-draft-13	4 Oct 2009	T. Scavo	Fixed bugs in CS 01
sstc-saml-holder-of-key-browser-sso-cd-03	3 Nov 2009	T. Scavo	Committee Draft 03