The SOAP Profile of the OASIS Security Assertion Markup Language (SAML)

Document identifier: draft-sstc-soap-profile-01.doc

Location: http://www.oasis-open.org/committees/security/docs

Publication date: 25 March 2002

Status: Interim draft; send comments to the editor

Editor:
Prateek Mishra, Netegrity, editor (pmishra@netegrity.com)

Contributors:

Bob Blakley, Tivoli
Jeff Hodges, Sun Microsystems
Eve Maler, Sun Microsystems
Chris McLaren., Netegrity
Irving Reid, Baltimore
Krishna Sankar, Cisco Systems

Rev	Date	By Whom	What
01	March 22, 2002	Prateek Mishra	Separated SOAP Profile from Bindings Model 09
02	March 25, 2002	Prateek Mishra	Added materials from conformance draft and security considerations draft

1 Introduction

This document specifies the SOAP profile of SAML. A separate specification **[SAMLCore]** defines the SAML assertions and request-response messages themselves.

1.1 Protocol Binding and Profile Concepts

Mappings from SAML request-response message exchanges into standard messaging or communication protocols are called SAML *protocol bindings* (or just *bindings*). An instance of mapping SAML request-response message exchanges into a specific protocol <FOO> is termed a *<FOO> binding for SAML* or a *SAML <FOO> binding*.

For example, an HTTP binding for SAML describes how SAML request and response message exchanges are mapped into HTTP message exchanges. A SAML SOAP binding describes how SAML request and response message exchanges are mapped into SOAP message exchanges.

Sets of rules describing how to embed and extract SAML assertions into a framework or protocol are called *profiles of SAML*. A profile describes how SAML assertions are embedded in or combined with other objects (for example, files of various types, or protocol data units of communication protocols) by an originating party, communicated from the originating site to a destination, and subsequently processed at the destination. A particular set of rules for embedding SAML assertions into and extracting them from a specific class of <FOO> objects is termed a *<FOO> profile of SAML*.

For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages, how SOAP headers are affected by SAML assertions, and how SAML-related error states should be reflected in SOAP messages.

The intent of this specification is to specify a selected set of bindings and profiles in sufficient detail to ensure that independently implemented products will interoperate.

For other terms and concepts that are specific to SAML, refer to the SAML glossary **[SAMLGloss]**.

1.2 Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 **[RFC2119]**.

```
Listings of productions or other normative code appear like this.
```

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

Note: Non-normative notes and explanations appear like this.

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

- The prefix `saml:` stands for the SAML assertion namespace **[SAMLCore]**.
- The prefix `samlp:` stands for the SAML request-response protocol namespace **[SAMLCore]**.
- The prefix `ds:` stands for the W3C XML Signature namespace, `http://www.w3.org/2000/09/xmldsig#` **[XMLSig]**.
- The prefix `SOAP-ENV:` stands for the SOAP 1.1 namespace, `http://schemas.xmlsoap.org/soap/envelope` **[SOAP1.1]**.

This specification uses the following typographical conventions in text: `<SAMLElement>`, `<ns:ForeignElement>`, `Attribute`, `OtherCode`. In some cases, angle brackets are used to indicate nonterminals, rather than XML elements; the intent will be clear from the context.

2 SOAP Profile of SAML

See Section **Error! Reference source not found.** for the definition of the SOAP binding for SAML, as opposed to the SOAP profile of SAML.

The SOAP profile of SAML is a realization of Scenarios 3-1 and 3-3 of the SAML requirements document **[SAMLReqs]** in the context of SOAP. It is based on a single interaction between a *sender* and a *receiver*, as follows:

1. The sender obtains one or more assertions.
2. The sender attaches the assertions to a SOAP message.
3. The sender sends the SOAP message with the attached assertions to the receiver. The SOAP message may be sent over any protocol for which a SOAP protocol binding is available **[SOAP1.1]**.
4. The receiver attempts to process the attached assertions. If it cannot process them, it returns an error message. If it can process them, it does so and also processes the rest of the SOAP message in an application-dependent way.

2.1 Required Information

Identification:

http://www.oasis-open.org/security/draft-sstc-soap-profile-model-01/profiles/SOAP

Contact information:

security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

2.2 SOAP Headers

SOAP provides a flexible header mechanism, which OPTIONAL to use for extending SOAP payloads with additional information. Rules for SOAP headers are given in **[SOAP1.1]** §4.2.

SAML assertions MUST be contained within the SOAP `<SOAP-ENV:Header>` element, which is in turn contained within the `<SOAP-ENV:Envelope>` element. Two standard SOAP attributes are available for use with header elements: `actor` and `mustUnderstand`. Use of the `actor` attribute is application dependent and no normative use is specified herein.

The `mustUnderstand` attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process. SAML assertions MUST have the `mustUnderstand` attribute set to `1`; this ensures that a SOAP processor to which the SAML header is directed must process the SAML assertions as explained in **[SOAP1.1]** §4.2.3.

2.3 SAML Errors

If the receiver is able to access the SAML assertions contained in the SOAP header, but is unable to process them, the receiver SHOULD return a SOAP message with a `<SOAP-ENV:Fault>` element as the message body and with `samlp:failure` as the `<SOAP-ENV:Faultcode>` element value. Reasons why the receiver may be unable to process SAML assertions, include, but are not limited to:

1. The assertion contains a `<saml:Condition>` element that the receiver does not understand.
2. The signature on the assertion is invalid.
3. The receiver does not accept assertions from the issuer of the assertion in question.
4. The receiver does not understand the extension schema used in the assertion.

It is RECOMMENDED that the `<SOAP-ENV:Faultstring>` element contain an informative message. This specification does not specify any normative text. Sending parties MUST NOT rely on specific contents in the `<SOAP-ENV:Faultstring>` element.

Following is an example of providing fault information:

```
<SOAP-ENV:Fault>
    <SOAP-ENV:Faultcode>samlp:failure</SOAP-ENV:Faultcode>
    <SOAP-ENV:Faultstring>SAML Version Error</SOAP-ENV:Faultstring>
</SOAP-ENV:Fault>
```

2.4 Security Considerations

Every assertion MUST be signed by the issuer following the guidelines in [SAML-DSIG-Profile].

The sender and receiver MUST ensure the data integrity of SOAP messages and contained assertions. A variety of different techniques are available for providing data integrity, including, for example, use of TLS/SSL, digital signatures over the SOAP message, and IPsec.

When a receiver processes a SOAP message containing SAML assertions, it MUST make an explicit determination of the relationship between subject of the assertions and the sender. Merely obtaining a SOAP message containing assertions carries no implication about the sender's right to possess and communicate the included assertions. A variety of means are available for making such a determination, including, for example, explicit policies at the receiver, authentication of sender, and use of digital signature.

Two message formats for ensuring the data integrity of the attachment of assertions to a SOAP message, `HolderOfKey` and `SenderVouches`, are described below. The `HolderOfKey` format has the additional property that it also implies a specific relationship between the sender and subject of the assertions included within the SOAP message. Senders and receivers implementing the SOAP Profile of SAML MUST implement both formats.

2.5 HolderOfKey Format

The following sections describe the `HolderOfKey` format for ensuring the data integrity of assertions attached to a SOAP message. Both make use of XML Signature **[XMLSig]**.

2.5.1 Sender

In this case, the sender and the subject are the same entity. The sender obtains one or more assertions from one or more authorities. Each assertion MUST include the following `<saml:SubjectConfirmation>` element:

```
<saml:SubjectConfirmation>
    <saml:ConfirmationMethod>HolderOfKey</saml:ConfirmationMethod>
    <ds:KeyInfo>…</ds:KeyInfo>
</saml:SubjectConfirmation>
```

The `<saml:SubjectConfirmation>` element carries information about the sender's key within the `<ds:KeyInfo>` element. The `<ds:KeyInfo>` element provides varied ways for describing information about the sender's public or secret key.

In addition to the assertions, the sender MUST include a `<ds:Signature>` element within the SOAP `<SOAP-ENV:Header>`. The `<ds:Signature>` element MUST apply to the SAML assertion elements in the `<SOAP-ENV:Header>` element, and all the relevant portions of the `<SOAP-ENV:Body>` element, as required by the application. Specific applications might require that the signature also apply to additional elements in SOAP header.

2.5.2 Receiver

The receiver MUST verify that each assertion carries a `<saml:SubjectConfirmation>` element of the following form:

```
<saml:SubjectConfirmation>
    <saml:ConfirmationMethod>HolderOfKey</saml:ConfirmationMethod>
    <ds:KeyInfo>…</ds:KeyInfo>
</saml:SubjectConfirmation>
```

The receiving party MUST check the validity of the signature found in a `<SOAP-ENV:Envelope>/<ds:Signature>` sub-element of the SOAP message. The receiving party SHOULD use the sender's public or information about a secret key carried within the `<saml:SubjectConfirmation>/<ds:KeyInfo>` element carried within each assertion.

> **Note:** The `<ds:KeyInfo>` element is used only for checking integrity of assertion attachment (message integrity). Therefore, there is no requirement that the receiver validate the key or certificate. This suggests that, if needed, a sender can generate a public/private key pair and utilize it for this purpose.

Once the above steps have been completed, the receiver can further process the assertions and SOAP message contents with the assurance that portions of the SOAP message that fall within the scope of the digital signature have been constructed by the sender and have not been altered by an intermediary. Further, the sender has provided proof of possession of the corresponding private-key (or secret-key) component of the information included in the `<saml:SubjectConfirmation>/<ds:KeyInfo>` element included in each assertion. If the receiver believes the assertions to be valid, then the information contained in the assertions MAY be considered to be describing the sender.

2.5.3 Example

The following example illustrates the `HolderOfKey` message format:

```
<?xml:version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SOAP-ENV:Header>
        <saml:AssertionList mustUnderstand="1"
            AssertionID="192.168.2.175.1005169137985"
            IssueInstant="2001-11-07T21:38:57Z"
            Issuer="M and M Consulting"
            MajorVersion="1"
            MinorVersion="0"
            xmlns:saml="…"
            xmlns:samlp="…">
            <saml:Conditions
                NotBefore="2001-11-07T21:33:57Z"
                NotOnOrAfter="2001-11-07T21:48:57Z">
                <saml:AbstractCondition
                  xsi:type="AudienceRestrictionConditionType">
                  <saml:Audience>
                  http://www.example.com/research_finance_agreement.xml
                  </saml:Audience>
```

```
                </saml:AbstractCondition>
            </saml:Conditions>
            <saml:AuthenticationStatement
                AuthenticationInstant="2001-11-07T21:38:57Z"
                AuthenticationMethod="Password">
                <saml:Subject>
                    <saml:NameIdentifier Name="goodguy"
                        SecurityDomain="www.example.com />
                    <saml:SubjectConfirmation>HolderOfKey
                    </saml:SubjectConfirmation>
                    <ds:KeyInfo>
                        <ds:KeyValue>…</ds:KeyValue>
                        <ds:X509Data>…</ds:X509Data>
                    </ds:KeyInfo>
                </saml:Subject>
                <saml:AuthenticationLocality
                    DNSAddress="some_computer"
                    IPAddress="111.111.111.111" />
            </saml:AuthenticationStatement>
            <ds:Signature>
              <ds:SignedInfo>
                <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2000/09/WD-xml-c14n-20000119" />
                <ds:SignatureMethod Algorithm=
                  "http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
                <ds:Reference URI="">
                  <ds:Transforms>
                    <ds:Transform
     Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                  </ds:Transforms>
                  <ds:DigestMethod
                 Algorithm= "http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
                  <ds:DigestValue>GSUvQSPfYkAC9wpHbLSfPEjMllo=
                  </ds:DigestValue>
                </ds:Reference>
              </ds:SignedInfo>
              <ds:SignatureValue>
              iLJj64yusw7h4FTbiyKRvAQoALlmeCnKxhKqStrFahVXIZUXacmDJw==
              </ds:SignatureValue>
              <ds:KeyInfo>
                <ds:KeyValue>…</ds:KeyValue>
                <ds:X509Data>…</ds:X509Data>
              </ds:KeyInfo>
            </ds:Signature>
        </saml:AssertionList>
        <ds:Signature>
          <ds:SignedInfo>
            <ds:CanonicalizationMethod>
        Algorithm= "http://www.w3.org/TR/2000/09/WD-xml-c14n-20000119" />
            <ds:SignatureMethod> Algorithm=
              "http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
            <ds:Reference URI="">
              <ds:Transforms>
                <ds:Transform
     Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
              </ds:Transforms>
              <ds:DigestMethod
                Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1" />
              <ds:DigestValue>UYRsLhRffJagF7d+RfNt8CPKhbM=
              </ds:DigestValue>
            </ds:Reference>
          </ds:SignedInfo>
          <ds:SignatureValue>
```

```
            HJJWbvqW9E84vJVQkjjLLA6nNvBX7mY00TZhwBdFNDElqscSXZ5Ekw==
            </ds:SignatureValue>
          </ds:Signature>
      </SOAP-ENV:Header>
    </SOAP-ENV:Body>
      <ReportRequest>
      <TickerSymbol>SUNW</TickerSymbol>
      </ReportRequest>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.6 SenderVouches Format

The following sections describe the `SenderVouches` format for ensuring the data integrity of assertions attached to a SOAP message.

2.6.1 Sender

In this case, the sender and subject MAY be distinct entities. The sender obtains one or more assertions from one or more authorities and includes them in a SOAP message. Each assertion MUST include the following `<saml:SubjectConfirmation>` element:

```
<saml:SubjectConfirmation>
    <saml:ConfirmationMethod>SenderVouches</saml:ConfirmationMethod>
</saml:SubjectConfirmation>
```

In addition to the assertions, the sender MUST include a `<ds:Signature>` element within the SOAP `<SOAP-ENV:Header>`. The `<ds:Signature>` element MUST apply to the SAML assertion elements in the `<SOAP-ENV:Header>` element, and all the relevant portions of the `<SOAP-ENV:Body>` element, as required by the application. Specific applications might require that the signature also apply to additional elements in SOAP header.

Following the XML Signature specification, the sender MAY include a `<ds:KeyInfo>` element within the `<ds:Signature>` element. The `<ds:KeyInfo>` element provides varied ways for describing information about the sender's public or secret key. If is omitted, the receiver is expected to identify the key based on context.

2.6.2 Receiver

The receiver MUST verify that each assertion carries a `<saml:SubjectConfirmation>` element of the following form:

```
<saml:SubjectConfirmation>
    <saml:ConfirmationMethod>SenderVouches</saml:ConfirmationMethod>
</saml:SubjectConfirmation>
```

The receiving party MUST check the validity of the signature found in the `<SOAP-ENV:Envelope>/<ds:Signature>` element. Information about the sender's public or secret key either is found in the `<SOAP-ENV:Envelope>/<ds:Signature>/<ds:KeyInfo>` element carried within the SOAP envelope or is based on application context.

Once the above steps have been completed, the receiver can further process the assertions and SOAP message contents with the assurance that portions of the SOAP message that fall within

the scope of the digital signature have been constructed by the sender and have not been altered by an intermediary.

In contrast to the `HolderOfKey` case, information about the sender either is provided by the contents of the `<ds:KeyInfo>` element found within the signature or is based on application context.

2.6.3 Example

The following example illustrates the `SenderVouches` message format:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schema.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header xmlns:saml="…"
    <saml:Assertion mustUnderstand="1">…</saml:Assertion>
    <saml:Assertion mustUnderstand="1">…</saml:Assertion>
    <ds:Signature>…
        <ds:KeyInfo>…</ds:KeyInfo>
    </ds:Signature>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <message_payload/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>{PRIVATE "TYPE=PICT;ALT=Figure 3: SOAP document with
inserted assertions"}
```

2.7 Additional Security Considerations

The model described in this section does not take into account (1) replay attacks, (2) authentication of sender by receiver, (3) authentication of receiver by sender, and (4) confidentiality. These must be addressed by means other than those described in this specification.

3 Security Considerations

This profile defines methods for securely attaching SAML assertions to a SOAP document. SOAP documents are used in multiple contexts, specifically including cases where the message is transported without an active session, the message can be persisted, and the message is routed through a number of intermediaries. Such a general context of use suggests that users of this profile must be concerned with a variety of threats. In particular, no consideration has been given to the issue of sender or receiver authentication. Therefore, if required, the sender may need to authenticate the receiver using some authentication technique dependent on the context of use. Further, the receiver may need to authenticate the sender using some techniques dependent on the context of use. In the latter case, there is a possibility that the receiver may authenticate the sender utilizing the attached SAML assertions as a credential together with other information.

The SAML bindings and profiles specification **Error! Reference source not found.**, Section 4.2.3, provides more information about security considerations for this profile.

3.1 Holder of Key

This profile has one or more authorities issuing assertions that contain <SubjectConfirmation> elements that basically say "This assertion is valid if it is presented with proof that the presenter is the holder of the specified key".

A sender inserts these assertions in a message and the entire message (payload and assertions) are digitally signed using the specified key—thus providing proof to the receiver that the sender of the message held the key specified in the assertions.

3.1.1 Eavesdropping

Eavesdropping continues to be a threat in the same manner as for the SAML SOAP binding, as discussed in Section **Error! Reference source not found.**. The routable nature of SOAP adds the potential for a large number of steps and actors in the course of a message's lifetime, which means that the potential incidences of eavesdropping are increased as the number of possible times a message is in transit increases.

The persistent nature of SOAP messages adds an additional possibility of eavesdropping, in that stored items can be read from their store.

To provide maximum protection from eavesdropping, assertions should be encrypted in such a way that only the intended audiences can view the material. This removes threats of eavesdropping in transit, but does not remove risks associated with storage by the receiver or poor handling of the clear text by the receiver.

3.1.2 Replay

Binding of assertions to a document opens the door to replay attacks by a malicious user. Issuing a `HolderOfKey` assertion amounts to "blessing the user's key" for the purpose of binding assertions to documents. Once a `HolderOfKey` assertion has been issued to a user, that user can bind it to any document or documents he chooses.

While each assertion is signed, and bound by a second signature into a document, which prevents a malicious third-party (who has no access to the private key required for the binding signature) from binding the assertions to arbitrary documents, there is nothing preventing a malicious **user** (who by definition has access to the private key) from detaching a signed assertion from the document it arrived in and rebinding it to another document.

There are two lines of defense against this type of attack. The first is to consider carefully to whom you issue `HolderOfKey` assertions (can they be trusted with the right to attach the assertion to any document?) and what kind of assertions you issue as `HolderOfKey` assertions (do you want to give up control over the binding of this particular statement to a given document?). The second is a short lifetime on the assertion, to narrow the window of opportunity for this attack.

The capture and resubmission of the entire message (SAML assertions and business payload) is a threat. One counter-measure is to add information about time, or a sequence number to the digital signature included in the SOAP header. The receiver can use this information to detect duplicate messages.

3.1.3 Message Insertion

There is no message insertion attack at the level of the `HolderOfKey` format of the SOAP profile.

3.1.4 Message Deletion

There is no message deletion attack at the level of the `HolderOfKey` format of the SOAP profile.

3.1.5 Message Modification

The double signing in this profile prevents most message modification attacks. The receiver is always able to verify the signature on the assertion itself (and should be able to verify that the key used in that signing act is associated with the putative signer by means of X509v3 certificate, Certificate Revocation List checks, and so on), which provides a guarantee that the assertion is unaltered.

The receiver can also verify the binding signature to ensure that the message to which the assertion is attached is unaltered.

The profile is secure against modification within the context of an existing trust relationship. The remaining threats (compromised keys, revoked certificates being used, and so on) are outside the scope of SAML.

Note that the threat of message modification by the holder of the key exists, as discussed in the discussion of replay attacks in Section 3.1.2.

3.1.6 Man-in-the-Middle

An MITM attack is impossible for the `HolderOfKey` format of the SOAP profile, since the assertion specifies the key that must be used for the binding signature, and the assertion itself is protected against tampering by a signature.

The MITM can eavesdrop (if communication is not protected by some confidentiality scheme) but cannot alter the document without detection.

Note that a MITM could alter parts of the document unprotected by the signature (i.e. the other header elements within the <Signature> element). For example, a MITM could remove an included <KeyInfo> block from a <Signature> without affecting the validity of the signature. Theoretically this could force an XKMS lookup or other network call that could be perverted to malicious ends. However this does not pose a threat for the HolderOfKey profile since (1) the assertion has issuer info (so you know who originated the assertion came) (2) the signed assertion includes the key for the binding signature.

3.2 Sender Vouches

This profile has one or more authorities issuing assertions that contain <SubjectConfirmation> elements that basically say “Trust these if you trust the issuer and the entity who signed them”.

A collects these assertions and inserts them in a message. The sender then signs over the entire message, with the signature being used to indicate that these assertions (which are themselves signed by their issuers) are vouched for by the sender.

3.2.1 Eavesdropping

Eavesdropping continues to be a threat in the same manner as for the SAML SOAP binding, as discussed in Section **Error! Reference source not found.**. The routable nature of SOAP adds the potential for a large number of steps and actors in the course of a message's lifetime, which means that the potential incidences of eavesdropping are increased as the number of possible times a message is in transit increases.

The persistent nature of SOAP messages adds an additional possibility of eavesdropping, in that persisted items can be read from their store.

To provide maximum protection from eavesdropping, assertions should be encrypted in such a way that only the intended audiences can view the material. This removes threats of eavesdropping in transit, but does not remove risks associated with storage by the receiver or poor handling of the clear text by the receiver.

3.2.2 Replay

The fact that the sender does all binding prevents a variety of replay attacks that reuse the assertion with different documents. In this case the assertions are directly signed into the document, so separating them from the document for reuse would not benefit a malicious user. (i.e. The assertions are only as valid as the binding signature of the sender, so reusing them with a different key does not pose a risk).

Authorities should note that once a "SenderVouches" assertion has been issued, there is no control over who may use it. Any entity coming into contact with the assertion can separate these assertions and use them by signing them with their own keys. Consumers of SenderVouches assertions must, therefore, carefully decide which senders to allow to vouch for what assertions.

The capture and resubmission of the entire message (SAML assertions and business payload) is a threat. One counter-measure is to add information about time, or a sequence number to the digital signature included in the SOAP header. The receiver can use this information to detect duplicate messages.

3.2.3 Message Insertion

There is no message insertion attack at the level of the `SenderVouches` format of the SOAP profile.

3.2.4 Message Deletion

There is no message insertion attack at the level of the `SenderVouches` format of the SOAP profile.

3.2.5 Message Modification

The binding signature should prevent any message modification attacks. Selection of what parts of the document to sign should be made carefully with the possibility of this attack in mind.

Receivers should consider only the portions of the document actually bound by signature to the assertions as valid with respect to the assertions.

3.2.6 Man-in-the-Middle

The requirement for a signature here should prevent MITM attacks. Note that the verifiability of the signature is key to this step: Not only must a receiver be able to verify that a document was signed with a key, but he also needs to be able to verify the binding of key to identity. This may be accomplished by including an X509v3 certificate with the digital signature, which the receiver verifies by some means (XKMS, OCSP, CRLs) and further maps onto a known identity for the signer.

If this step is skipped, then MITM becomes a possibility: The MITM captures the original document, alters it, and passes along this new document signed with a key that purports to be from the original sender (but which is actually held by the MITM).

The MITM can eavesdrop (if communication is not protected by some confidentiality scheme) but cannot alter the document without detection.

4 Conformance

TBD

5 References

[Anders] A suggestion on how to implement SAML browser bindings without using "Artifacts", http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt.

[AuthXML] *AuthXML: A Specification for Authentication Information in XML*, http://www.oasis-open.org/committees/security/docs/draft-authxml-v2.pdf.

[MSURL] Microsoft technical support article, http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP.

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

[RFC2617] *HTTP Authentication: Basic and Digest Access Authentication*, http://www.ietf.org/rfc/rfc2617.txt, IETF RFC 2617.

[S2ML] *S2ML: Security Services Markup Language*, Version 0.8a, January 8, 2001. http://www.oasis-open.org/committees/security/docs/draft-s2ml-v08a.pdf.

[SAMLCore] Hallam-Baker, P. et al., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-21.pdf, OASIS, December 2001.

[SAMLGloss] J. Hodges et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf, OASIS, December 2001.

[SAMLSec] J. Hodges et al., Security Considerations for the OASIS *Security Assertion Markup Language (SAML)*, http://www.oasis-open.org/committees/security/docs/draft-sec-consider-02.pdf, OASIS, December 2001.

[SAMLReqs] D. Platt et al., SAML Requirements and Use Cases, OASIS, December 2001.

[Shib] Shiboleth Overview and Requirements http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.htmlhttp://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html

[ShibMarlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1, http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-architecturel-00.pdf

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

[RFC1738] Uniform Resource Locators (URL), http://www.ietf.org/rfc/rfc1738.txt

[RFC1750] Randomness Recommendations for Security. http://www.ietf.org/rfc/rfc1750.txt

[RFC1945] Hypertext Transfer Protocol -- HTTP/1.0, http://www.ietf.org/rfc/rfc1945.txt.

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

[RFC2774] An HTTP Extension Framework, http://www.ietf.org/rfc/rfc2774.txt.

565 **[SOAP1.1]** D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*,
566 http://www.w3.org/TR/SOAP, World Wide Web Consortium Note, May
567 2000.

568 **[CoreAssnEx]** Core Assertions Architecture, Examples and Explanations,
569 http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-
570 07.pdf.

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

573 **[WEBSSO]** RL "Bob" Morgan, Interactions between Shibboleth and local-site web
574 sign-on services, http://middleware.internet2.edu/shibboleth/docs/draft-
575 morgan-shibboleth-websso-00.txt

576 **[SESSION]** RL "Bob" Morgan, Support of target web server sessions in Shibboleth,
577 http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-
578 session-00.txt

579 **[SSLv3]** The SSL Protocol Version 3.0,
580 http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt

581 **[Rescorla-Sec]** E. Rescorla et al., *Guidelines for Writing RFC Text on Security*
582 *Considerations*, http://www.ietf.org/internet-drafts/draft-rescorla-sec-
583 cons-03.txt.

Appendix A. Notices

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's procedures with respect to rights in OASIS specifications can be found at 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 implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.