Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1

OASIS Standard incorporating Approved Errata, 1 November 2006

Document identifier:

wss-v1.1-spec-errata-os-SwAProfile

Location:

http://docs.oasis-open.org/wss/v1.1/

Technical Committee:

OASIS Web Services Security (WSS) TC

Chair(s):

Kelvin Lawrence, IBM

Chris Kaler, Microsoft

Editors:

Frederick Hirsch, Nokia

Abstract:

This specification defines how to use the OASIS Web Services Security: SOAP Message Security standard [WSS-Sec] with SOAP Messages with Attachments [SwA].

Status:

This is an **OASIS Standard incorporating Approved Errata changes to the OASIS Standard** produced by the Web Services Security Technical Committee. The standard was approved by the OASIS membership on 1 February 2006. Check the current location noted above for possible errata to this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at www.oasis-open.org/committees/wss.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (www.oasis-open.org/committees/wss/ipr.php. The non-normative errata page for this specification is located at www.oasis-open.org/committees/wss.

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.

OASIS has been notified of intellectual property rights claimed in regard to some or all of the contents of this specification. For more information consult the online list of claimed rights.

Table of Contents

1 Introduction

This section is non-normative. Note that sections 2.2 and 5 are normative. All other sections are non-normative.

This document describes how to use the OASIS Web Services Security: SOAP Message Security standard [WSS-Sec] with SOAP Messages with Attachments [SwA]. More specifically, it describes how a web service consumer can secure SOAP attachments using SOAP Message Security for attachment integrity, confidentiality and origin authentication, and how a receiver may process such a message.

A broad range of industries - automotive, insurance, financial, pharmaceutical, medical, retail, etc - require that their application data be secured from its originator to its ultimate consumer. While some of this data will be XML, quite a lot of it will not be. In order for these industries to deploy web service solutions, they need an interoperable standard for end-to-end security for both their XML data and their non-XML data.

Profiling SwA security may help interoperability between the firms and trading partners using attachments to convey non-XML data that is not necessarily linked to the XML payload. Many industries, such as the insurance industry require free-format document exchange in conjunction with web services messages. This profile of SwA should be of value in these cases.

In addition, some content that could be conveyed as part of the SOAP body may be conveyed as an attachment due to its large size to reduce the impact on message and XML processing, and may be secured as described in this profile.

This profile is applicable to using SOAP Message Security in conjunction with SOAP Messages with Attachments (SwA). This means the scope is limited to SOAP 1.1, the scope of SwA.

Goals of this profile include the following:

- Enable those who choose to use SwA to secure these messages, including chosen attachments, using SOAP Message Security
- Allow the choice of securing MIME header information exposed to the SOAP layer, if desired.
- Do not interfere with MIME transfer mechanisms, in particular, allow MIME transfer encodings to change to support MIME transfer, despite support for integrity protection.
- Do not interfere with the SOAP processing model – in particular allow SwA messages to transit SOAP intermediaries.

Non-goals include:

- Provide guidance on which of a variety of security mechanisms are appropriate to a given application. The choice of transport layer security (e.g. SSL/TLS), S/MIME, application use of XML Signature and XML Encryption, and other SOAP attachment mechanisms (MTOM) is explicitly out of scope. This profile assumes a need and desire to secure SwA using SOAP Message security.
- Outline how different security mechanisms may be used in combination.
- Enable persisting signatures. It may be possible depending on the situation and measures taken, but is not discussed in this profile.
- Support signing and/or encryption of portions of attachments. This is not supported by this profile, but is not necessarily precluded. Application use of XML Signature and XML Encryption may be used to accomplish this. SOAP Message security may also support this in some circumstances, but this profile does not address or define such usage.

The existence of this profile does not preclude using other mechanisms to secure attachments conveyed in conjunction with SOAP messages, including the use of XML security technologies at the application layer or the use of security for the XML Infoset before a serialization that uses attachment technology [MTOM]. The requirements in this profile only apply when securing SwA attachments explicitly according to this profile.

2 Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

2.1 Notational Conventions

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.

When describing abstract data models, this specification uses the notational convention used by the XML Infoset. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas [XML-Schema], this specification uses the notational convention of OASIS Web Services Security: SOAP Message Security. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like [XPath] notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

Commonly used security terms are defined in the Internet Security Glossary [SECGLO]. Readers are presumed to be familiar with the terms in this glossary as well as the definitions in the SOAP Message Security specification [WSS-Sec] .

2.1.1 Namespaces

Namespace URIs (of the general form "some-URI") represent application-dependent or context-dependent URIs as defined in RFC 2396 [RFC2396]. This specification is designed to work with the SOAP 1.1 [SOAP11] message structure and message processing model, the version of SOAP supported by SOAP Messages with Attachments. The current SOAP 1.1 namespace URI is used herein to provide detailed examples.

The namespaces used in this document are shown in the following table (note that for brevity, the examples use the prefixes listed below but do *not* include the URIs – those listed below are assumed).

Prefix	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
S11	http://schemas.xmlsoap.org/soap/envelope/
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
wsswa	http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1.xsd
xenc	http://www.w3.org/2001/04/xmlenc#

The URLs provided for the wsse and wsu namespaces can be used to obtain the schema files.

2.1.2 Acronyms and Abbreviations

The following (non-normative) table defines acronyms and abbreviations for this document, beyond those defined in the SOAP Message Security standard.

Term	**Definition**
CID	Content ID scheme for URLs. Refers to Multipart MIME body part, that includes both MIME headers and content for that part. [RFC2392]
SwA	SOAP Messages with Attachments [SwA]

2.2 Normative References

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

[CHARSETS] Character sets assigned by IANA. See ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets.

[Excl-Canon] “Exclusive XML Canonicalization, Version 1.0”, W3C Recommendation, 18 July 2002. http://www.w3.org/TR/xml-exc-c14n/.

[RFC2045] “Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies”, IETF RFC 2045, November 1996, http://www.ietf.org/rfc/rfc2045.txt.

[**RFC2046**] “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types”, IETF RFC 2046, November 1996, http://www.ietf.org/rfc/rfc2046.txt.

[**RFC2047**] “Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text”, IETF RFC 2047, November 1996, http://www.ietf.org/rfc/rfc2047.txt.

[RFC2048] “Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures”, http://www.ietf.org/rfc/rfc2048.txt.

[RFC2049] “Multipurpose Internet Mail Extensions(MIME) Part Five: Conformance Criteria and Examples”, http://www.ietf.org/rfc/rfc2049.txt.

[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.

[RFC2184] P. Resnick, “MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations”, IETF RFC 2184, August 1997, http://www.ietf.org/rfc/rfc2184.txt.

[RFC2392] E. Levinson, “Content-ID and Message-ID Uniform Resource Locators”, IETF RFC 2392, http://www.ietf.org/rfc/rfc2392.txt.

[RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998, http://www.ietf.org/rfc/rfc2396.txt.

[RFC2557] “MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)”, IETF RFC 2557, March 1999, http://www.ietf.org/rfc/rfc2557.txt.

[RFC2633] Ramsdell B., “S/MIME Version 3 Message Specification”, Standards Track RFC 2633, June 1999. http://www.ietf.org/rfc/rfc2633.txt.

[RFC2822] “Internet Message Format”, IETF RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt.

[SECGLO] "Internet Security Glossary," Informational RFC 2828, May 2000.

[SOAP11] "SOAP: Simple Object Access Protocol 1.1", W3C Note, 08 May 2000.

[SwA] "SOAP Messages with Attachments", W3C Note, 11 December 2000, http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211.

[WS-I-AP] "Attachments Profile Version 1.0", *Final Material,* 2004-08-24, http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html.

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

[XML-Schema] W3C Recommendation, "XML Schema Part 1: Structures,"2 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[XML-Sig] W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002, http://www.w3.org/TR/xmldsig-core/.

[XPath] W3C Recommendation, "XML Path Language", 16 November 1999, http://www.w3.org/TR/xpath.

2.3 Non-normative References

[DecryptT] M. Hughes et al, "Decryption Transform for XML Signature", W3C Recommendation, 10 December 2002. http://www.w3.org/TR/xmlenc-decrypt/.

[MTOM] "SOAP Message Transmission Optimization Mechanism", W3C Recommendation, 25 January 2005, http://www.w3.org/TR/soap12-mtom/.

3 MIME Processing

This profile is concerned with the securing of SOAP messages with attachments, attachments that are conveyed as MIME parts in a multi-part MIME message as outlined in SOAP Messages with Attachments [SwA]. This involves two processing layers, SOAP messaging and MIME transfer. This specification defines processing of a merged SOAP and MIME layer, in order to meet SwA security requirements. It relies on an underlying MIME transfer layer that allows changes to MIME transfer encoding as a message transits MIME nodes. This profile does not impose restrictions on that MIME transfer layer apart from aspects that are exposed to the SOAP processing layer. Likewise, this profile does not restrict the SOAP processing model, including use of SOAP intermediaries, allowing SOAP Messages with Attachments to transit SOAP nodes.

To accommodate the ability to secure attachment headers that are exposed to the SOAP message layer and application, this profile does not assume a strict protocol layering of MIME, SOAP and application. Rather, this profile allows a SOAP sender to create a primary SOAP envelope as well as attachments to be sent with the message. It is up to the application which, if any, of the attachments are referenced from SOAP header and/or body blocks. The application may be aware of, and concerned with, certain aspects of the attachment MIME representation, including Content-Type and Content-Length headers, to give two examples. Due to this concern, the application may choose to secure these exposed headers. This does not mean, however, that the application and SOAP layer are aware or concerned with all MIME headers used for MIME transit, in particular issues related to transfer encoding. The expectation is that the MIME processing layer of the sender and receiver will handle transfer encoding issues, hiding this detail from the processing layer associated with this profile. As a result, this specification focuses on those aspects of MIME processing that are exposed and of concern to higher protocol layers, while ignoring MIME transit specific details.

This model has two implications. First, it means that certain aspects of MIME processing, such as transfer encoding processing, are out of scope of the profile and do not need to be addressed. Secondly, it means that many of the MIME headers are also out of scope of the profile and the profile does not support integrity protection of these headers, since they are expected to change. If more security protection is required then it must occur by other means, such as with a protocol layer below the MIME layer, for example transport security (with the understanding that such security may not always apply end-end).

Use of this profile is intended to be independent of MIME-specific security processing, although care must be taken when using both SOAP Message Security and S/MIME. When conveyed end-to-end, S/MIME content may be conveyed opaquely as one or more attachments, as a MIME content type. If S/MIME security is to be used between nodes that convey the SOAP message, then this may also be opaque to SOAP Message Security, as long as the attachment that was sent by the initial SOAP sender is the same as that which is received by the receiving SOAP intermediary or ultimate SOAP receiver. Care must be taken to ensure this will be the case. Clearly SOAP Message Security encryption could prevent S/MIME processing of an attachment, and likewise S/MIME encryption could prevent SOAP Message Security signature verification if these techniques are interleaved. This potential concern is out of scope of this profile.

4 XML Attachments

A SOAP Messages with Attachments multi-part MIME structure contains a primary SOAP envelope in the root part and one or more attachments in additional MIME parts. Some of these attachments may have a content type corresponding to XML, but do not contain the primary SOAP envelope to be processed.

Some attachments associated with the SOAP body may be targeted at the SOAP Ultimate Receiver along with the SOAP body and may be processed at the application layer along with the body. Others may be targeted at intermediaries. How attachments are to be processed and how these attachments are referenced from SOAP header and body blocks, if at all, is dependent on the application. In many cases the attachment content may not need to be processed as XML as the message traverses intermediaries.

Generally requiring canonicalization of XML attachments whenever transmitting them is undesirable, both due to the potential ambiguities related to the canonicalization context of the attachment (e.g. Is it an independent XML document, a portion of the primary SOAP envelope, etc) as well as the universal performance impact of such a canonicalization requirement. When XML attachment content is signed, then XML canonicalization is required, as is generally the case when signing XML.

MIME part canonicalization (as described below) is required for non-XML attachments to enable SOAP Message Security signatures that are stable despite MIME transfer processing.

5 Securing SOAP With Attachments

Attachments may be associated with SOAP messages, as outlined in SOAP Messages with Attachments [SwA]. This profile defines how such attachments may be secured for integrity and confidentiality using the OASIS Web Services Security: SOAP Message Security standard. This does not preclude using other techniques. The requirements in this profile only apply when securing SwA attachments explicitly according to this profile.

This profile considers all attachments as opaque whether they are XML or some other content type. It is the sole responsibility of the application to perform further interpretation of attachments , including the ability to sign or encrypt portions of those attachments.

5.1 Primary SOAP Envelope

When SOAP attachments are used as specified in [SwA] each SOAP message is accompanied by a MIME header and possibly multiple boundary parts. This is known as a SOAP message package. This document assumes that a proper SOAP message package is constructed using the HTTP and MIME headers appropriate to [SwA].

The primary SOAP envelope SHOULD be conveyed in the first MIME part, but MAY be conveyed in another MIME part when the start attribute is specified in the HTTP Multipart/Related header.

In particular, implementations should take care in distinguishing between the HTTP headers in the SOAP message package and the start of the SOAP payload. For example, the following Multipart/Related header belongs to the HTTP layer and not the main SOAP payload:

```
Content-Type: Multipart/Related; boundary=xy1; type=”text/xml”; start="<foo>"
```

The main SOAP payload begins with the appropriate boundary. For example:

```
 --xy1
Content-Type: text/xml; charset=utf-8
Content-ID: <foo>

<?xml version='1.0' ?>
<s11:Envelope xmlns:s11=”http://schemas.xmlsoap.org/soap/envelope/” />
```

5.2 Referencing Attachments

SOAP Messages with Attachments defines two MIME mechanisms for referencing attachments. The first mechanism uses a CID scheme URL to refer to the attachment that has a Content-ID MIME header with a value corresponding to the URL, as defined in [RFC 2392]. For example, a content id of “foo” may be specified in the MIME part with the MIME header “Content-ID: <foo>” and be referenced using the CID Schema URL “cid:foo”.

The second mechanism is to use a URL to refer to an attachment containing a Content-Location MIME header. In this case the URL may require resolution to determine the referenced attachment [RFC2557].

For simplicity and interoperability this profile limits WS-Security references to attachments to CID scheme URLs. Attachments referenced from WS-Security signature references or cipher references MUST be referenced using CID scheme URLs.

This profile assumes, since it is not defined in RFC 2396 Section 4.2, that all cid: references are not same-document references and that therefore, under XMLDSIG, dereferencing a cid: URI always yields an octet stream as input to the transform chain [RFC2396], [XMLDSIG].

5.3 MIME Part Reference Transforms

By definition of RFC 2392, a URI reference to a MIME attachment includes the MIME headers associated with that attachment as well as the MIME part content [RFC2392]. Since there may be some confusion as to what is referenced, it is useful to clearly indicate what is included in the referenced attachment. In addition, some applications may wish to only encrypt or include the attachment content in a signature reference hash, and others may wish to include MIME headers and content.

For these reasons, this profile defines reference transforms, allowing a clear and explicit statement of what is included in a MIME reference. These transforms are called "MIME Part Reference Transforms".

The input of each of these transforms is an octet stream, as defined in XML Security [XML-Sig].

5.3.1 Attachment-Content-Signature-Transform

The Attachment-Content-Signature-Transform indicates that only the content of a MIME part is referenced for signing. This transform MUST be identified using the URI value:

```
http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-
Signature-Transform
```

When this transform is used the content of the MIME part should be canonicalized as defined in section 5.4.2.

The octet stream input to this transform is the entire content of the MIME attachment associated with the CID, including all the MIME headers and attachment content, as represented in the MIME part containing the attachment.

The output of the transform is an octet stream consisting of the canonicalized serialization of the attachment content. All of the MIME headers associated with the MIME part are ignored and not included in the output octet stream. The canonicalization of the content is described in section 5.4.2 of this specification.

5.3.2 Attachment-Complete-Signature-Transform

The Attachment-Complete-Signature-Transform indicates that both the content and selected headers of the MIME part are referenced for signing. This transform MUST be identified using the URI value:

```
http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete-
Signature-Transform
```

This transform specifies that in addition to the content the following MIME headers are to be included (when present):

- Content-Description
- Content-Disposition
- Content-ID
- Content-Location
- Content-Type

These headers are included because of their common use and the risks associated with inappropriate modification. If other headers are to be protected, other mechanisms at the application level should be used (such as copying values into a SOAP header) and this is out of scope of this profile.

Other MIME headers associated with the MIME part serialization are not referenced by the transform and are not to be included in signature calculations.

When this transform is used the MIME headers should be canonicalized as defined in section 5.4.1 and the MIME content should be canonicalized as defined in section 5.4.2.

The octet stream input to this transform is the entire content of the MIME attachment associated with the CID, including all the MIME headers and attachment content, as represented in the MIME part containing the attachment.

The output of the transform is an octet stream consisting of concatenation of the MIME canonicalized MIME headers selected by the transform followed by the canonicalized attachment content. The canonicalization of headers and content are described in sections 5.4.1 and 5.4.2 of this specification.

5.3.3 Attachment-Ciphertext-Transform

The Attachment-Ciphertext-Transform indicates that only the content of a MIME part is referenced, and contains the ciphertext related to an XML EncryptedData element. This transform MUST be identified using the URI value:

```
http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Ciphertext-
Transform
```

The octet stream input to this transform is the entire content of the MIME attachment associated with the CID, including all the MIME headers and attachment content, as represented in the MIME part containing the attachment.

The output of the transform is an octet stream consisting of the ciphertext as conveyed in the MIME part content. All of the MIME headers associated with the MIME part are ignored and not included in the output octet stream. The MIME text canonicalization of the content is described in section 5.4.2 of this specification.

5.4 Integrity and Data Origin Authentication

Integrity and data origin authentication may be provided for SwA attachments using XML Signatures, as outlined in the SOAP Message Security standard as profiled in this document. This is useful independent of the content of the MIME part – for example, it is possible to sign a MIME part that already contains a signed object created by an application. It may be sensible to sign such an attachment as part of SOAP Message security so that the receiving SOAP node may verify that all attachments are intact before delivering them to an application. A SOAP intermediary may also choose to perform this verification, even if the attachments are not otherwise processed by the intermediary.

5.4.1 MIME header canonicalization

The result of MIME header canonicalization is a UTF-8 encoded octet stream.

Each of the MIME headers listed for the Attachment-Complete transform MUST be canonicalized as part of that transform processing, as outlined in this section. This means the transform MUST perform the following actions in interpreting the MIME headers for signature creation or verification (this order is not prescriptive as long as the same result is obtained)

1. The transform MUST process MIME headers before the MIME content.

2. The transform MUST only process MIME headers that are explicitly present in the attachment part and are listed in the Attachment-Complete transform section of this specification, except that a MIME part

without a Content-Type header MUST be treated as having a Content-Type header with the value "Content-Type: text/plain; charset=us-ascii". MIME headers not listed in the Attachment-Complete transform section of this specification are to be ignored by the transform.

3. The MIME headers MUST be processed by the Attachment-Complete transform in lexicographic order (ascending).

4. The MIME header names MUST be processed by the transform as having the case according to the MIME specifications (as shown in the Attachment-Complete section).

5. The MIME header values MUST be unfolded [RFC2822].

6. Any Content-Description MIME header containing RFC2047 encoding MUST be decoded [RFC2047].

7. When a Content-ID header is processed, the "<>" characters associated with the msg-id MUST be included in the transform input. The reason is that although semantically these angle bracket characters are not part of the msg-id (RFC 2822) they are a standard part of the header lexicographic representation. If these characters are not integrity protected then an attacker could remove them causing the CID transformation specified in RFC2392 to fail.

8. Folding whitespace in structured MIME headers (e.g. Content-Disposition, Content-ID, Content-Location, Content-Type) that is not within quotes MUST be removed. Folding whitespace in structured MIME headers that is within quotes MUST be preserved. Folding whitespace in unstructured MIME headers (e.g. Content-Description) MUST be preserved [RFC2822]. For example, whitespace immediately following the colon delimiter in the structured Content-Type header MUST be removed, but whitespace immediately following the colon delimiter in the unstructured Content-Description header MUST be preserved.

9. Comments in MIME header values MUST be removed [RFC2822].

10. Case-insensitive MIME header values (e.g. media type/subtype values and disposition-type values) MUST be converted to lowercase. Case-sensitive MIME header values MUST be left as is with respect to case [RFC2045].

11. Quoted characters other than double-quote and backslash ("\") in quoted strings in structured MIME headers (e.g. Content-ID) MUST be unquoted. Double-quote and backslash ("\") characters in quoted strings in structured MIME headers MUST be character encoded [RFC2822].

12. Canonicalization of a MIME header MUST generate a UTF-8 encoded octet stream containing the following: the MIME header name, a colon (":"), the MIME header value, and the result of canonicalizing the MIME header parameters in lexicographic order (ascending) as described below.

13. MIME header parameter names MUST be converted to lowercase [RFC2045].

14. MIME parameter values containing RFC2184 character set, language, and continuations MUST be decoded. The resulting canonical output MUST not contain the RFC2184 encoding [RFC2184].

15. Case-insensitive MIME header parameter values MUST be converted to lowercase. Case-sensitive MIME header parameter values MUST be left as is with respect to case [RFC2045].

16. Enclosing double-quotes MUST be added to MIME header parameter values that do not already contain enclosing quotes. Quoted characters other than double-quote and backslash ("\") in MIME header parameter values MUST be unquoted. Double-quote and backslash characters in MIME parameter values MUST be character encoded.

17. Canonicalization of a MIME header parameter MUST generate a UTF-8 encoded octet stream containing the following: a semi-colon (";"), the parameter name (lowercase), an equals sign ("="), and the double-quoted parameter value.

18. Each header MUST be terminated by a single CRLF pair, without any trailing whitespace.

19.The last header MUST be followed by a single CRLF and then the MIME content.

5.4.2 MIME Content Canonicalization

Before including attachment content in a signature reference hash calculation, that MIME attachment SHOULD be canonicalized. The reason is that signature verification requires an identical hash of content as when signing occurred.

Content of an XML Content-Type MUST be XML canonicalized using Exclusive XML Canonicalization without comments, as specified by the URI http://www.w3.org/2001/10/xml-exc-c14n# [Excl-Canon]. The reason for requiring Exclusive Canonicalization is that many implementations will support Exclusive Canonicalization for other XML Signature purposes, since this form of canonicalization supports context changes. The InclusiveNamespace PrefixList attribute SHOULD be empty or not present.

Other types of MIME content SHOULD be canonicalized according to the MIME part canonicalization mechanism appropriate to the Content-Type of the MIME part.

To quote the S/MIME specification (section 3.1.1 “Canonicalization”) which deals with this issue [RFC2633]:

> The exact details of canonicalization depend on the actual MIME type and subtype of an entity, and are not described here. Instead, the standard for the particular MIME type should be consulted. For example, canonicalization of type text/plain is different from canonicalization of audio/basic. Other than text types, most types have only one representation regardless of computing platform or environment which can be considered their canonical representation.

MIME types are registered. This registration includes a section on "Canonicalization and Format Requirements" [RFC2048] and requires each MIME type to have a canonical representation.

The MIME “text” type canonical form is defined in the MIME conformance specification (See “Canonical Encoding Model”) [RFC2049]. Important aspects of “text” media type canonicalization include line ending normalization to <CR><LF> and ensuring that the charset is a registered charset (see RFC 2633 section “Canonicalization”). [RFC2633, CHARSETS, RFC2045].

5.4.3 Protecting against attachment insertion threat

Including an attachment in a signature calculation enables a receiver to detect modification of that attachment. Including all attachments in a signature calculation, by providing a <ds:Reference> for each, protects against the threat of attachment removal. This does not protect against insertion of a new attachment.

The simplest protection against attachment insertion is for the receiver to know that all attachments should be included in a signature calculation – unreferenced attachments are then an indication of an attachment insertion attack.

Such information may be communicated in or out of band. Definition of these approaches is out of the scope of this profile.

5.4.4 Processing Rules for Attachment Signing

The processing rule for signing is modified based on the SOAP Message Security rules.

After determining which attachments are to be included as references in a signature, create a <ds:Signature> element in a <wsse:Security> header block targeted at the recipient, including a <ds:Reference> for each attachment to be protected by the signature. Additional <ds:Reference> elements may refer to content in the SOAP envelope to be included in the signature.

For each attachment Reference, perform the following steps:

1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as outlined in section 5.4.2. Attachments of an XML content type require Exclusive XML Canonicalization without comments[Excl-Canon].
2. If MIME headers are to be included in the signature, perform MIME header canonicalization as outlined in section 5.4.1.
3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL attribute value to this URL.
4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST include a <ds:Transform> element with the Algorithm attribute having the full URL value specified earlier in this profile – corresponding to either the Attachment-Complete-Signature-Transform or Attachment-Content-Signature-Transform, depending on what is to be included in the hash calculation. This MUST be the first transform listed. The <ds:Transform> element MUST NOT contain any transform for a MIME transfer encoding purpose (e.g. base64 encoding) since transfer encoding is left to the MIME layer as noted in section 2. This does not preclude the use of XML Transforms, including a base64 transform, for other purposes.
5. Extract the appropriate portion of the MIME part consistent with the selected transform.
6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature Recommendation.

5.4.5 Processing Rules for Attachment Signature Verification

Signature verification is performed as outlined in SOAP Message Security and the XML Digital Signature Recommendation, with the following considerations for SwA attachments.

To verify <ds:Reference> hashes for SwA attachments, the following steps must be performed for each reference to an attachment:

1. Find the attachment corresponding to the <ds:Reference> URL attribute value. This value MUST correspond to the Content-ID for the attachment[SwA].
2. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as outlined in section 5.4.2. Attachments of an XML content type require Exclusive XML Canonicalization without comments[Excl-Canon]. The MIME content to be MIME canonicalized MUST have had any transfer-encoding processed at the MIME layer before this step is performed.
3. If MIME headers were included in the signature, perform MIME header canonicalization as outlined in section 5.4.1.
4. Extract the appropriate portion of the MIME part according to the MIME Part Signature Transform value.
5. Calculate the reference hash and verify the reference.

5.4.6 Example Signed Message

```
Content-Type: multipart/related; boundary=”BoundaryStr” type=”text/xml”
```

```
--BoundaryStr
Content-Type: text/xml

<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..."
xmlns:xenc="...">
  <S11:Header>
     <wsse:Security>

      <wsse:BinarySecurityToken wsu:Id="CertAssociatedWithSigningKey"
           EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
           ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#x509v3">
        ...
      </wsse:BinarySecurityToken>

      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm=
'http://www.w3.org/2001/10/xml-exc-c14n#'/>
          <ds:SignatureMethod Algorithm=
              'http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
       <ds:Reference URI="cid:bar">
             <ds:Transforms>
               <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
wss-SwAProfile-1.1#Attachment-Content-Signature-Transform"/>
             </ds:Transforms>
          <ds:DigestMethod Algorithm=
              "http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>DeadBeef</ds:SignatureValue>

        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#CertAssociatedWithSigningKey"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>

    </ds:Signature>
   </wsse:Security>
  </S11:Header>
  <S11:Body>
   some items
  </S11:Body>
</S11:Envelope>
--BoundaryStr
Content-Type: image/png
Content-ID: <bar>
Content-Transfer-Encoding: base64

the image
```

5.5 Encryption

A SwA attachment may be encrypted for confidentiality protection, protecting either the MIME part content including selected MIME headers, or only the MIME part content.

This is done using XML Encryption to encrypt the attachment, placing the resulting cipher text in the updated attachment body replacing the original content, and placing a new <xenc:EncryptedData> element in the <wsse:Security> header. An <xenc:CipherReference> MUST link the <xenc:EncryptedData> element with the cipher data.

The key used for encryption MAY be conveyed using an <xenc:EncryptedKey> element in the <wsse:Security> header. In this case the <xenc:ReferenceList> element in the <xenc:EncryptedKey> element MUST contain an <xenc:DataReference> with a URI attribute specifying the <xenc:EncryptedData> element in the <wsse:Security> header corresponding to the attachment.

When the same <xenc:EncryptedKey> corresponds to multiple <xenc:EncryptedData> elements, the <xenc:ReferenceList> in the <xenc:EncryptedKey> element SHOULD contain an <xenc:DataReference> for each <xenc:EncryptedData> element, both for attachments and encrypted items in the primary SOAP envelope. References should be ordered to correspond to ordering of the security header elements.

When an <xenc:EncryptedKey> element is not used when encrypting an attachment, then the <xenc:EncryptedData> element MAY contain a <ds:KeyInfo> element to specify a key as outlined in the SOAP Message Security standard. Different deployments may have different requirements on how keys are referenced. When an <xenc:EncryptedKey> element is used the <xenc:EncryptedData> element MUST NOT contain a <ds:KeyInfo> element.

When an attachment is encrypted, an <xenc:EncryptedData> element will be placed in the <wsse:Security> header. An <xenc:ReferenceList> element associated with this <xenc:EncryptedData> element may also be added, as recommended by WSS: SOAP Message Security.

> Note: The same CID is used to refer to the attachment before encryption and after. This avoids the need to rewrite references to the attachment, avoiding issues related to generating unique CIDs and relating to preserving the correspondence to the original WSDL definition.

5.5.1 MIME Part CipherReference

This profile requires that <xenc:EncryptedData> elements corresponding to encrypted SwA attachments use a <xenc:CipherReference> to refer to the cipher text, to be conveyed in the attachment. Upon encryption the MIME part attachment content is replaced with the encoded cipher text.

The <xenc:CipherReference> MUST have a <xenc:Transforms> child element. This element MUST have a <ds:Transform> child having an Algorithm attribute with a URI value specifying the Attachment-Ciphertext-Transform. This transform explicitly indicates that when dereferencing the MIME part reference that only the MIME part content is to be used as the cipher value.

The <xenc:CipherReference> MUST NOT contain a transform used for a transfer encoding purpose (e.g. the base64 transform). Transfer encoding is left to the MIME layer, as noted in section 2.

5.5.2 Encryption Processing Rules

The order of the following steps is not normative, although the result should be the same as if this order were followed.

1. When encrypting both attachments and primary SOAP envelope content using the same key, perform the attachment processing first.

> Note: The SOAP Message Security standard states that elements should be prepended to the security header. This processing rule supports putting the <xenc:EncryptedData> element first in the header with <xenc:EncryptedKey> and tokens following. Thus, a

receiver should be able to process the <xenc:EncryptedKey> before the <xenc:EncryptedData> element for the attachment.

2. Encrypt the attachment part using XML Encryption, according to the rules of XML Encryption. Encrypt either the attachment including content and selected MIME headers or only the attachment content.

When encryption includes MIME headers, only the headers listed in this specification for the Attachment-Complete-Signature-Transform (Section 5.3.2) are to be included in the encryption. If a header listed in the profile is present it MUST be included in the encryption. If a header is not listed in this profile, then it MUST NOT be included in the encryption.

3. Set the <xenc:EncryptedData> Type attribute value to a URI that specifies adherence to this profile and that specifies what was encrypted (MIME content or entire MIME part including headers). The following URIs MUST be used for this purpose:

- Content Only:

```
http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-Only
```

- Content and headers:

```
http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete
```

4. Set the <xenc:EncryptedData> MimeType attribute to match the attachment MIME part Content-Type header before encryption when the Content-Only URI is specified for the Type attribute value. The MimeType attribute value MAY be set when the AttachmentComplete Type attribute value is specified.
5. Optionally set the <xenc:EncryptedData> Encoding attribute to reflect the attachment content encoding, as visible to the security layer at the time of encryption. This is advisory information to the decryption security layer. It should be understood that this has no relation with the actual encoding that could be performed independently by the MIME layer later for transfer purposes.
6. Set the <xenc:EncryptedData> <xenc:CipherReference> to the same reference URL for the attachment that was used before encryption . This MUST be a CID scheme URL referring to the attachment part Content-ID. Ensure this MIME header is in the part conveying the cipher data after encryption.
7. Include the Attachment-Ciphertext-Transform in the <xenc:CipherReference> <xenc:Transforms> list.
8. Prepend the <xenc:EncryptedData> element to the <wsse:Security> SOAP header block and then prepend the associated optional <xenc:ReferenceList> element.
9. Update the attachment MIME part, replacing the original content with the cipher text generated by the XML Encryption step.
10. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the cipher data.

5.5.3 Decryption Processing Rules

The <xenc:CipherReference> URL MUST be a URL that refers to the MIME part containing the cipher text, and must also correspond to the reference value of the original attachment that was encrypted. This MUST be a CID scheme URL.

Decryption may be initiated upon locating the <xenc:EncryptedData> element in the <wsse:Security> header.

The following decryption steps must be performed so that the result is as if they were performed in this order:

1. Extract the cipher text from the attachment referenced by the <xenc:CipherReference> URL attribute. The Attachment-Ciphertext-Transform defined in this profile indicates that the MIME part content is extracted.
2. Decrypt the cipher text using the information present in the appropriate <xenc:EncryptedData> element and possibly other out of band information, according to the XML Encryption Standard.
3. If the <xenc:EncryptedData>Type attribute indicates that selected MIME headers were encrypted, then those MIME headers MUST be replaced by the result of decryption, as well as the MIME part content.
4. If the <xenc:EncryptedData>Type attribute indicates that only the content of the MIME part was encrypted, then the cipher text content of the attachment part MUST be replaced by the result of decryption. In this case the MIME part Content-Type header value MUST be replaced by the <xenc:EncryptedData> MimeType attribute value.
5. If the <xenc:EncryptedData> Encoding attribute is present then the decryption security layer may pass this advisory information to the application.

5.5.4 Example

This example shows encryption of the primary SOAP envelope body as well as an attachment using a single symmetric key conveyed using an EncryptedKey element.

```
Content-Type: multipart/related; boundary=”BoundaryStr” type=”text/xml”
--BoundaryStr
Content-Type: text/xml

<S11:Envelope
  xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wsswssecurity-secext-1.0.xsd"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmls:ds="http://www.w3.org/2000/09/xmldsig#">

 <S11:Header>
   <wsse:Security>

     <wsse:BinarySecurityToken wsu:Id="Acert"
           EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
           ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#x509v3">
        ...
     </wsse:BinarySecurityToken>

     <xenc:EncryptedKey Id='EK'>
       <EncryptionMethod
         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
      <ds:KeyInfo Id="keyinfo">
        <wsse:SecurityTokenReference>
          <ds:X509Data>
            <ds:X509IssuerSerial>
              <ds:X509IssuerName>
               DC=ACMECorp, DC=com
              </ds:X509IssuerName>
               <ds:X509SerialNumber>12345678</X509SerialNumber>
```

```
            </ds:X509IssuerSerial>
          </ds:X509Data>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
      <ReferenceList>
        <DataReference URI='#EA'/>
        <DataReference URI='#ED'/>
      </ReferenceList>
     </EncryptedKey>

     <xenc:EncryptedData
       Id='EA'
       Type="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-
1.1#Attachment-Content-Only"
       MimeType="image/png">
       <xenc:EncryptionMethod
         Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc'/>
       <xenc:CipherData>
        <xenc:CipherReference URI=cid:bar">
          <xenc:Transforms>
             <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
wss-SwAProfile-1.1#Attachment-Ciphertext-Transform"/>
          </xenc:Transforms>
        </xenc:CipherReference>
       </xenc:CipherData>
     </xenc:EncryptedData>

   </wsse:Security>
  </S11:Header>
  <S11:Body>
   <xenc:EncryptedData Id='ED'>
    <xenc:EncryptionMethod
     Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc'/>
    <xenc:CipherData>
     <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
 </S11:Body>
</S11:Envelope>
--BoundaryStr
Content-Type: application/octet-stream
Content-ID: <bar>
Content-Transfer-Encoding: binary

BinaryCipherData
```

5.6 Signing and Encryption

When portions of content are both signed and encrypted, there is possible confusion as to whether encrypted content need first be decrypted before signature verification. This confusion can occur when the order of operations is not clear [DecryptT]. This problem may be avoided with SOAP Message Security for SwA attachments when attachments and corresponding signatures and encryptions are targeted for a single SOAP recipient (actor). The SOAP Message Security standard explicitly states that there may not be two <wsse:Security> headers targeted at the same actor, nor may there be two headers without a designated actor. In this case the SOAP Message Security and SwA profile processing rules may eliminate ambiguity since each signing or encryption produces an element in the <wsse:Security>

header, and these elements are ordered. (Signing produces <ds:Signature> elements and encryption produces <xenc:EncryptedData> elements).

If an application produces different <wsse:Security> headers targeted at different recipients, these are processed independently by the recipients. Thus there is no need to correlate activities between distinct headers – the order is inherent in the SOAP node model represented by the distinct actors.

A. Acknowledgments

Current Contributors:

Michael	Hu	Actional
Maneesh	Sahu	Actional
Duane	Nickull	Adobe Systems
Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Laboratory
Hal	Lockhart	BEA Systems
Denis	Pilipchuk	BEA Systems
Corinna	Witt	BEA Systems
Steve	Anderson	BMC Software
Rich	Levinson	Computer Associates
Thomas	DeMartini	ContentGuard
Merlin	Hughes	Cybertrust
Dale	Moberg	Cyclone Commerce
Rich	Salz	Datapower
Sam	Wei	EMC
Dana S.	Kaufman	Forum Systems
Toshihiro	Nishimura	Fujitsu
Kefeng	Chen	GeoTrust
Irving	Reid	Hewlett-Packard
Kojiro	Nakayama	Hitachi
Paula	Austel	IBM
Derek	Fu	IBM
Maryann	Hondo	IBM
Kelvin	Lawrence	IBM
Michael	McIntosh	IBM
Anthony	Nadalin	IBM
Nataraj	Nagaratnam	IBM
Bruce	Rich	IBM
Ron	Williams	IBM
Don	Flinn	Individual
Kate	Cherry	Lockheed Martin
Paul	Cotton	Microsoft
Vijay	Gajjala	Microsoft
Martin	Gudgin	Microsoft
Chris	Kaler	Microsoft
Frederick	Hirsch	Nokia
Abbie	Barbir	Nortel
Prateek	Mishra	Oracle
Vamsi	Motukuru	Oracle
Ramana	Turlapi	Oracle
Ben	Hammond	RSA Security
Rob	Philpott	RSA Security
Blake	Dournaee	Sarvega
Sundeep	Peechu	Sarvega
Coumara	Radja	Sarvega

Pete	Wenzel	SeeBeyond
Manveen	Kaur	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Symon	Chang	TIBCO Software
John	Weiland	US Navy
Hans	Granqvist	VeriSign
Phillip	Hallam-Baker	VeriSign
Hemma	Prafullchandra	VeriSign

759 **Previous Contributors:**

Peter	Dapkus	BEA
Guillermo	Lao	ContentGuard
TJ	Pannu	ContentGuard
Xin	Wang	ContentGuard
Shawn	Sharp	Cyclone Commerce
Ganesh	Vaideeswaran	Documentum
Tim	Moses	Entrust
Carolina	Canales-Valenzuela	Ericsson
Tom	Rutt	Fujitsu
Yutaka	Kudo	Hitachi
Jason	Rouault	HP
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Hiroshi	Maruyama	IBM
David	Melgar	IBM
Kent	Tamura	IBM
Wayne	Vicknair	IBM
Phil	Griffin	Individual
Mark	Hayes	Individual
John	Hughes	Individual
Peter	Rostin	Individual
Davanum	Srinivas	Individual
Bob	Morgan	Individual/Internet2
Bob	Atkinson	Microsoft
Keith	Ballinger	Microsoft
Allen	Brown	Microsoft
Giovanni	Della-Libera	Microsoft
Alan	Geller	Microsoft
Johannes	Klein	Microsoft
Scott	Konersmann	Microsoft
Chris	Kurt	Microsoft
Brian	LaMacchia	Microsoft
Paul	Leach	Microsoft
John	Manferdelli	Microsoft
John	Shewchuk	Microsoft
Dan	Simon	Microsoft
Hervey	Wilson	Microsoft
Jeff	Hodges	Neustar
Senthil	Sengodan	Nokia
Lloyd	Burch	Novell

Ed	Reed	Novell
Charles	Knouse	Oblix
Vipin	Samar	Oracle
Jerry	Schwarz	Oracle
Eric	Gravengaard	Reactivity
Andrew	Nash	Reactivity
Stuart	King	Reed Elsevier
Martijn	de Boer	SAP
Jonathan	Tourzan	Sony
Yassir	Elley	Sun
Michael	Nguyen	The IDA of Singapore
Don	Adams	TIBCO
Morten	Jorgensen	Vordel