



Web Services Security SOAP Message with Attachments (SwA) Profile Version 1.1.1

Committee Specification Draft 01

23 March 2011

Specification URIs:

This Version:

<http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/wss-SwAProfile-v1.1.1-csd01.doc>
(Authoritative)
<http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/wss-SwAProfile-v1.1.1-csd01.pdf>
<http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/wss-SwAProfile-v1.1.1-csd01.html>

Previous Version:

<http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SwAProfile.sxw>
<http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SwAProfile.pdf>
<http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SwAProfile.html>

Latest Version:

<http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SwAProfile-v1.1.1.doc> (Authoritative)
<http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SwAProfile-v1.1.1.pdf>
<http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SwAProfile-v1.1.1.html>

Technical Committee:

OASIS Web Services Security Maintenance (WSS-M) TC

Chair(s):

David Turner, Microsoft

Editor(s):

Frederick Hirsch, Nokia
Carlo Milono, Tibco

Related work:

This specification is one part of a multi-part Work Product. The other parts include:

[Web Services Security: SOAP Message Security Version 1.1.1](#)
[Web Services Security Kerberos Token Profile Version 1.1.1](#)
[Web Services Security Rights Expression Language \(REL\) Token Profile Version 1.1.1](#)
[Web Services Security Username Token Profile Version 1.1.1](#)
[Web Services Security X.509 Certificate Token Profile Version 1.1.1](#)
[Web Services Security SAML Token Profile Version 1.1.1](#)
Schemas: <http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/xsd/>

This specification supersedes:

- [Web Services Security: SOAP Messages with Attachments \(SwA\) Profile 1.1](#), OASIS Standard incorporating Approved Errata, 1 November 2006

- [Web Services Security: SOAP Messages with Attachments \(SwA\) Profile 1.1](#), OASIS Approved Errata, 1 November 2006

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 document was last revised or approved by the [OASIS Web Services Security Maintenance \(WSS-M\) TC](#) on the above date. The level of approval is also listed above. Check the "Latest Version" location noted above for possible later revisions of 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 <http://www.oasis-open.org/committees/wss-m/>.

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 (<http://www.oasis-open.org/committees/wss-m/ipr.php>).

This document integrates specific error corrections or editorial changes to the preceding specification, within the scope of the Web Services Security and this TC.

This document introduces a third digit in the numbering convention where the third digit represents a consolidation of error corrections, bug fixes or editorial formatting changes (e.g., 1.1.1); it does not add any new features beyond those of the base specifications (e.g., 1.1).

Citation Format:

[WSS-SOAP-Attachments-Profile-V1.1.1]

Web Services Security SOAP Message with Attachments (SwA) Profile Version 1.1.1. 23 March 2011. OASIS Committee Specification Draft 01. [http://docs.oasis-open.org/wss-wss/v1.1.1/csd01/wss-SwAProfile-v1.1.1-csd01.doc](http://docs.oasis-open.org/wss/wss/v1.1.1/csd01/wss-SwAProfile-v1.1.1-csd01.doc).

Notices

Copyright © OASIS Open 2011. All Rights Reserved.

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	5
2	Notations and Terminology	6
2.1	Notational Conventions.....	6
2.1.1	Namespaces.....	6
2.1.2	Acronyms and Abbreviations.....	7
2.2	Normative References	7
2.3	Non-normative References.....	8
3	MIME Processing	9
4	XML Attachments	10
5	Securing SOAP With Attachments	11
5.1	Primary SOAP Envelope	11
5.2	Referencing Attachments	11
5.3	MIME Part Reference Transforms	12
5.3.1	Attachment-Content-Signature-Transform	12
5.3.2	Attachment-Complete-Signature-Transform	12
5.3.3	Attachment-Ciphertext-Transform	13
5.4	Integrity and Data Origin Authentication	13
5.4.1	MIME header canonicalization	13
5.4.2	MIME Content Canonicalization	14
5.4.3	Protecting against attachment insertion threat	15
5.4.4	Processing Rules for Attachment Signing	15
5.4.5	Processing Rules for Attachment Signature Verification	16
5.4.6	Example Signed Message.....	16
5.5	Encryption	17
5.5.1	MIME Part CipherReference	18
5.5.2	Encryption Processing Rules	18
5.5.3	Decryption Processing Rules	19
5.5.4	Example.....	19
5.6	Signing and Encryption.....	21
#	Conformance.....	22
A.	Acknowledgements	23
B.	Revision History.....	27

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.

47 2 Notations and Terminology

48 2.1 Notational Conventions

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

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

53 `Example code listings appear like this.`

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

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

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

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

66 2.1.1 Namespaces

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

72 The namespaces used in this document are shown in the following table (note that for brevity, the
73 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#

Comment [CM1]: No real schema – this will resolve to an HTML page discussing the namespace.

74
75 The URLs provided for the wsse and wsu namespaces can be used to obtain the schema files.

76 2.1.2 Acronyms and Abbreviations

77 The following (non-normative) table defines acronyms and abbreviations for this document, beyond those
78 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]

79 2.2 Normative References

80	[RFC 2119]	S. Bradner. <i>Key words for use in RFCs to Indicate Requirement Levels</i> . IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt .
81		
82	[CHARSETS]	Character sets assigned by IANA. See http://www.iana.org/assignments/character-sets .
83		
84	[Excl-Canon]	"Exclusive XML Canonicalization, Version 1.0", W3C Recommendation, 18 July 2002. http://www.w3.org/TR/xml-exc-c14n/ .
85		
86	[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 .
87		
88		
89	[RFC2046]	"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", IETF RFC 2046, November 1996, http://www.ietf.org/rfc/rfc2046.txt .
90		
91	[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 .
92		
93		
94	[RFC2048]	"Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", http://www.ietf.org/rfc/rfc2048.txt .
95		
96	[RFC2049]	"Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", http://www.ietf.org/rfc/rfc2049.txt .
97		
98	[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 .
99		
100	[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 .
101		
102		
103	[RFC2392]	E. Levinson, "Content-ID and Message-ID Uniform Resource Locators", IETF RFC 2392, http://www.ietf.org/rfc/rfc2392.txt .
104		
105	[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 .
106		
107		
108	[RFC2557]	"MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", IETF RFC 2557, March 1999, http://www.ietf.org/rfc/rfc2557.txt .
109		
110	[RFC2633]	Ramsdell B., "S/MIME Version 3 Message Specification", Standards Track RFC 2633, June 1999. http://www.ietf.org/rfc/rfc2633.txt .
111		
112	[RFC2822]	"Internet Message Format", IETF RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt .
113		
114	[SECGLO]	"Internet Security Glossary," Informational RFC 2828, May 2000.
115	[SOAP11]	"SOAP: Simple Object Access Protocol 1.1", W3C Note, 08 May 2000.
116	[SwA]	"SOAP Messages with Attachments", W3C Note, 11 December 2000, http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211 .
117		
118	[WS-I-AP]	"Attachments Profile Version 1.0", <i>Final Material</i> , 2004-08-24, http://www.w3.org/Profiles/AttachmentsProfile-1.0.html .
119		

- 120 **[WSS-Sec]** A. Nadalin et al., "Web Services Security: SOAP Message Security 1.1.1
121 [http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/wss-SOAPMessageSecurity-
123 v1.1.1-csd01.pdf](http://docs.oasis-open.org/wss-m/wss/v1.1.1/csd01/wss-SOAPMessageSecurity-
122 v1.1.1-csd01.pdf)
124 **[XML-Schema]** W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001,
125 <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.
126 W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001,
127 <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.
128 **[XML-Sig]** W3C Recommendation, "XML-Signature Syntax and Processing", 12 February
129 2002, <http://www.w3.org/TR/xmlsig-core/>.
130 **[XPath]** W3C Recommendation, "XML Path Language", 16 November
1999, <http://www.w3.org/TR/xpath>.

131 2.3 Non-normative References

- 132 **[DecryptT]** M. Hughes et al, "Decryption Transform for XML Signature",
133 W3C Recommendation, 10 December 2002. [http://www.w3.org/TR/xmlenc-
134 decrypt](http://www.w3.org/TR/xmlenc-decrypt)
135 **[MTOM]** "SOAP Message Transmission Optimization Mechanism", W3C
136 Recommendation, 25 January 2005, <http://www.w3.org/TR/soap12-mtom/>.

137

3 MIME Processing

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

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

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

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

176

4 XML Attachments

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

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

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

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

192 5 Securing SOAP With Attachments

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

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

201 5.1 Primary SOAP Envelope

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

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

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

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

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

```
213 --xyl  
214 Content-Type: text/xml; charset=utf-8  
215 Content-ID: <foo>  
216  
217 <?xml version='1.0' ?>  
218 <s11:Envelope xmlns:s11="http://schemas.xmlsoap.org/soap/envelope/" />
```

219 5.2 Referencing Attachments

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

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

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

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

233 5.3 MIME Part Reference Transforms

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

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

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

242 5.3.1 Attachment-Content-Signature-Transform

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

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

247
248 When this transform is used the content of the MIME part should be canonicalized as defined in section
249 5.4.2.

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

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

257 5.3.2 Attachment-Complete-Signature-Transform

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

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

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

- 264 • Content-Description
- 265 • Content-Disposition
- 266 • Content-ID
- 267 • Content-Location
- 268 • Content-Type

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

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

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

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

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

282 5.3.3 Attachment-Ciphertext-Transform

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

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

288
289 The octet stream input to this transform is the entire content of the MIME attachment associated with the
290 CID, including all the MIME headers and attachment content, as represented in the MIME part containing
291 the attachment.

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

296 5.4 Integrity and Data Origin Authentication

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

304 5.4.1 MIME header canonicalization

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

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

- 310 1. The transform MUST process MIME headers before the MIME content.
- 311 2. The transform MUST only process MIME headers that are explicitly present in the attachment part and
312 are listed in the Attachment-Complete transform section of this specification, except that a MIME part
313 without a Content-Type header MUST be treated as having a Content-Type header with the value
314 "Content-Type: text/plain; charset=us-ascii". MIME headers not listed in the Attachment-Complete
315 transform section of this specification are to be ignored by the transform.
- 316 3. The MIME headers MUST be processed by the Attachment-Complete transform in lexicographic order
317 (ascending).
- 318 4. The MIME header names MUST be processed by the transform as having the case according to the
319 MIME specifications (as shown in the Attachment-Complete section).
- 320 5. The MIME header values MUST be unfolded [RFC2822].

- 321 6. Any Content-Description MIME header containing RFC2047 encoding MUST be decoded [RFC2047].
322 7. When a Content-ID header is processed, the "<>" characters associated with the msg-id MUST be
323 included in the transform input. The reason is that although semantically these angle bracket
324 characters are not part of the msg-id (RFC 2822) they are a standard part of the header lexicographic
325 representation. If these characters are not integrity protected then an attacker could remove them
326 causing the CID transformation specified in RFC2392 to fail.
- 327 8. Folding whitespace in structured MIME headers (e.g. Content-Disposition, Content-ID, Content-
328 Location, Content-Type) that is not within quotes MUST be removed. Folding whitespace in structured
329 MIME headers that is within quotes MUST be preserved. Folding whitespace in unstructured MIME
330 headers (e.g. Content-Description) MUST be preserved [RFC2822]. For example, whitespace
331 immediately following the colon delimiter in the structured Content-Type header MUST be removed,
332 but whitespace immediately following the colon delimiter in the unstructured Content-Description
333 header MUST be preserved.
- 334 9. Comments in MIME header values MUST be removed [RFC2822].
- 335 10. Case-insensitive MIME header values (e.g. media type/subtype values and disposition-type values)
336 MUST be converted to lowercase. Case-sensitive MIME header values MUST be left as is with
337 respect to case [RFC2045].
- 338 11. Quoted characters other than double-quote and backslash ("\") in quoted strings in structured MIME
339 headers (e.g. Content-ID) MUST be unquoted. Double-quote and backslash ("\") characters in quoted
340 strings in structured MIME headers MUST be character encoded [RFC2822].
- 341 12. Canonicalization of a MIME header MUST generate a UTF-8 encoded octet stream containing the
342 following: the MIME header name, a colon (":"), the MIME header value, and the result of
343 canonicalizing the MIME header parameters in lexicographic order (ascending) as described below.
- 344 13. MIME header parameter names MUST be converted to lowercase [RFC2045].
- 345 14. MIME parameter values containing RFC2184 character set, language, and continuations MUST be
346 decoded. The resulting canonical output MUST not contain the RFC2184 encoding [RFC2184].
- 347 15. Case-insensitive MIME header parameter values MUST be converted to lowercase. Case-sensitive
348 MIME header parameter values MUST be left as is with respect to case [RFC2045].
- 349 16. Enclosing double-quotes MUST be added to MIME header parameter values that do not already
350 contain enclosing quotes. Quoted characters other than double-quote and backslash ("\") in MIME
351 header parameter values MUST be unquoted. Double-quote and backslash characters in MIME
352 parameter values MUST be character encoded.
- 353 17. Canonicalization of a MIME header parameter MUST generate a UTF-8 encoded octet stream
354 containing the following: a semi-colon (";"), the parameter name (lowercase), an equals sign ("="), and
355 the double-quoted parameter value.
- 356 18. Each header MUST be terminated by a single CRLF pair, without any trailing whitespace.
- 357 19. The last header MUST be followed by a single CRLF and then the MIME content.

358 5.4.2 MIME Content Canonicalization

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

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

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

369 To quote the S/MIME specification (section 3.1.1 "Canonicalization") which deals with this issue
370 [[RFC2633](#)]:

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

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

379

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

384 **5.4.3 Protecting against attachment insertion threat**

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

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

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

394 **5.4.4 Processing Rules for Attachment Signing**

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

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

400 For each attachment Reference, perform the following steps:

- 401 1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as
402 outlined in section 5.4.2. Attachments of an XML content type require Exclusive XML Canonicalization
403 without comments[[Excl-Canon](#)].
- 404 2. If MIME headers are to be included in the signature, perform MIME header canonicalization as
405 outlined in section 5.4.1.
- 406 3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL
407 attribute value to this URL.
- 408 4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST
409 include a <ds:Transform> element with the Algorithm attribute having the full URL value specified
410 earlier in this profile – corresponding to either the Attachment-Complete-Signature-Transform or
411 Attachment-Content-Signature-Transform, depending on what is to be included in the hash calculation.
412 This MUST be the first transform listed. The <ds:Transform> element MUST NOT contain any
413 transform for a MIME transfer encoding purpose (e.g. base64 encoding) since transfer encoding is left

414 to the MIME layer as noted in section 2. This does not preclude the use of XML Transforms, including
415 a base64 transform, for other purposes.
416 5. Extract the appropriate portion of the MIME part consistent with the selected transform.
417 6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature
418 Recommendation.

419 5.4.5 Processing Rules for Attachment Signature Verification

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

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

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

435 5.4.6 Example Signed Message

```
436 Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"
437 --BoundaryStr
438 Content-Type: text/xml
439
440 <S11:Envelope xmlns:S11="..." xmlns:wssse="..." xmlns:wssu="..." xmlns:ds="..."
441 xmlns:xenc="...">
442   <S11:Header>
443     <wsse:Security>
444
445       <wsse:BinarySecurityToken wsu:Id="CertAssociatedWithSigningKey"
446         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
447 wss-soap-message-security-1.0#Base64Binary"
448         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
449 x509-token-profile-1.0#x509v3">
450         ...
451       </wsse:BinarySecurityToken>
452
453     <ds:Signature>
454       <ds:SignedInfo>
455         <ds:CanonicalizationMethod Algorithm=
456 'http://www.w3.org/2001/10/xml-exc-c14n#' />
457         <ds:SignatureMethod Algorithm=
458 'http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
459         <ds:Reference URI="cid:bar">
460           <ds:Transforms>
461             <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
462 wss-SwAProfile-1.1#Attachment-Content-Signature-Transform"/>
463           </ds:Transforms>
464           <ds:DigestMethod Algorithm=
465             "http://www.w3.org/2000/09/xmldsig#sha1"/>
```

```

466     <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
467     </ds:Reference>
468   </ds:SignedInfo>
469   <ds:SignatureValue>DeadBeef</ds:SignatureValue>
470
471   <ds:KeyInfo>
472     <wsse:SecurityTokenReference>
473       <wsse:Reference URI="#CertAssociatedWithSigningKey"/>
474     </wsse:SecurityTokenReference>
475   </ds:KeyInfo>
476
477 </ds:Signature>
478 </wsse:Security>
479 </S11:Header>
480 <S11:Body>
481   some items
482 </S11:Body>
483 </S11:Envelope>
484 --BoundaryStr
485 Content-Type: image/png
486 Content-ID: <bar>
487 Content-Transfer-Encoding: base64
488
489 the image

```

490 5.5 Encryption

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

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

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

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

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

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

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

517 5.5.1 MIME Part CipherReference

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

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

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

527 5.5.2 Encryption Processing Rules

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

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

532 Note: The SOAP Message Security standard states that elements should be prepended
533 to the security header. This processing rule supports putting the <xenc:EncryptedData>
534 element first in the header with <xenc:EncryptedKey> and tokens following. Thus, a
535 receiver should be able to process the <xenc:EncryptedKey> before the
536 <xenc:EncryptedData> element for the attachment.

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

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

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

- 546 • Content Only:

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

- 549 • Content and headers:

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

- 551
- 552 4. Set the <xenc:EncryptedData> MimeType attribute to match the attachment MIME part Content-Type
553 header before encryption when the Content-Only URI is specified for the Type attribute value. The
554 MimeType attribute value MAY be set when the AttachmentComplete Type attribute value is specified.
- 555 5. Optionally set the <xenc:EncryptedData> Encoding attribute to reflect the attachment content
556 encoding, as visible to the security layer at the time of encryption. This is advisory information to the
557 decryption security layer. It should be understood that this has no relation with the actual encoding that
558 could be performed independently by the MIME layer later for transfer purposes.
- 559 6. Set the <xenc:EncryptedData> <xenc:CipherReference> to the same reference URL for the
560 attachment that was used before encryption. This MUST be a CID scheme URL referring to the
561 attachment part Content-ID. Ensure this MIME header is in the part conveying the cipher data after
562 encryption.

- 563 7. Include the Attachment-Ciphertext-Transform in the <xenc:CipherReference> <xenc:Transforms> list.
564 8. Prepend the <xenc:EncryptedData> element to the <wsse:Security> SOAP header block and then
565 prepend the associated optional <xenc:ReferenceList> element.
566 9. Update the attachment MIME part, replacing the original content with the cipher text generated by the
567 XML Encryption step.
568 10. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the
569 cipher data.

570 5.5.3 Decryption Processing Rules

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

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

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

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

591 5.5.4 Example

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

```
594 Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"
595 --BoundaryStr
596 Content-Type: text/xml
597
598 <S11:Envelope
599   xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
600   xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
601   wssecurity-secext-1.0.xsd"
602   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
603   xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
604
605   <S11:Header>
606     <wsse:Security>
607
608       <wsse:BinarySecurityToken wsu:Id="Acert"
609         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
610         wss-soap-message-security-1.0#Base64Binary"
611         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
612         x509-token-profile-1.0#x509v3">
```

```

613     ...
614     </wsse:BinarySecurityToken>
615
616     <xenc:EncryptedKey Id='EK'>
617         <EncryptionMethod
618             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
619         <ds:KeyInfo Id="keyinfo">
620             <wsse:SecurityTokenReference>
621                 <ds:X509Data>
622                     <ds:X509IssuerSerial>
623                         <ds:X509IssuerName>
624                             DC=ACMECorp, DC=com
625                         </ds:X509IssuerName>
626                         <ds:X509SerialNumber>12345678</X509SerialNumber>
627                     </ds:X509IssuerSerial>
628                 </ds:X509Data>
629             </wsse:SecurityTokenReference>
630         </ds:KeyInfo>
631         <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
632         <ReferenceList>
633             <DataReference URI='#EA' />
634             <DataReference URI='#ED' />
635         </ReferenceList>
636     </EncryptedKey>
637
638     <xenc:EncryptedData
639         Id='EA'
640         Type="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-
641 1.1#Attachment-Content-Only"
642         MimeType="image/png">
643         <xenc:EncryptionMethod
644             Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
645         <xenc:CipherData>
646             <xenc:CipherReference URI="cid:bar">
647                 <xenc:Transforms>
648                     <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
649 wss-SwAProfile-1.1#Attachment-Ciphertext-Transform"/>
650                 </xenc:Transforms>
651             </xenc:CipherReference>
652         </xenc:CipherData>
653     </xenc:EncryptedData>
654
655 </wsse:Security>
656 </S11:Header>
657 <S11:Body>
658     <xenc:EncryptedData Id='ED'
659         <xenc:EncryptionMethod
660             Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
661         <xenc:CipherData>
662             <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
663         </xenc:CipherData>
664     </xenc:EncryptedData>
665 </S11:Body>
666 </S11:Envelope>
667 --BoundaryStr
668 Content-Type: application/octet-stream
669 Content-ID: <bar>
670 Content-Transfer-Encoding: binary
671
672 BinaryCipherData

```

673 **5.6 Signing and Encryption**

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

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

687

688 **# Conformance**

689 The last numbered section in the specification must be the Conformance section. Conformance
690 Statements/Clauses go here.

691

A. Acknowledgements

692 The following individuals have participated in the creation of this specification and are gratefully
693 acknowledged:

694 **Participants:**

695 **Current Contributors:**

Comment [CM2]: Still TO DO

Tom	Rutt	Fujitsu Limited
Jacques	Durand	Fujitsu Limited
Calvin	Powers	IBM
Kelvin	Lawrence	IBM
Michael	McIntosh	Individual
Thomas	Hardjono	M.I.T.
David	Turner	Microsoft Corporation
Anthony	Nadalin	Microsoft Corporation
Monica	Martin	Microsoft Corporation
Marc	Goodner	Microsoft Corporation
Peter	Davis	Neustar
Hal	Lockhart	Oracle Corporation
Rich	Levinson	Oracle Corporation
Anil	Saldhana	Red Hat
Martin	Raepple	SAP AG
Federico	Rossini	Telecom Italia S.p.a.
Carlo	Milono	TIBCO Software Inc.
Don	Adams	TIBCO Software Inc.
Jerry	Smith	US Department of Defense (DoD)

696 **Previous Contributors:**

Michael	Hu	Actional
Maneesh	Sahu	Actional
Duane	Nickull	Adobe Systems
Gene	Thurston	AmberPoint
Frank	Siebenlist	Argonne National Laboratory
Peter	Dapkus	BEA
Hal	Lockhart	BEA Systems
Denis	Pilipchuk	BEA Systems
Corinna	Witt	BEA Systems
Steve	Anderson	BMC Software
Rich	Levinson	Computer Associates
Thomas	DeMartini	ContentGuard
Guillermo	Lao	ContentGuard

TJ	Pannu	ContentGuard
Xin	Wang	ContentGuard
Merlin	Hughes	Cybertrust
Dale	Moberg	Cyclone Commerce
Shawn	Sharp	Cyclone Commerce
Rich	Salz	Datapower
Ganesh	Vaideeswaran	Documentum
Sam	Wei	EMC
Tim	Moses	Entrust
Carolina	Canales-Valenzuela	Ericsson
Dana S.	Kaufman	Forum Systems
Toshihiro	Nishimura	Fujitsu
Tom	Rutt	Fujitsu
Kefeng	Chen	GeoTrust
Irving	Reid	Hewlett-Packard
Kojiro	Nakayama	Hitachi
Yutaka	Kudo	Hitachi
Jason	Rouault	HP
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
Bob	Blakley	IBM
Joel	Farrell	IBM
Satoshi	Hada	IBM
Hiroshi	Maruyama	IBM
David	Melgar	IBM
Kent	Tamura	IBM
Wayne	Vicknair	IBM
Don	Flinn	Individual
Phil	Griffin	Individual
Mark	Hayes	Individual
John	Hughes	Individual
Peter	Rostin	Individual
Davanum	Srinivas	Individual

Bob	Morgan	Individual/Internet2
Kate	Cherry	Lockheed Martin
Paul	Cotton	Microsoft
Vijay	Gajjala	Microsoft
Martin	Gudgin	Microsoft
Chris	Kaler	Microsoft
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
Frederick	Hirsch	Nokia
Senthil	Sengodan	Nokia
Abbie	Barbir	Nortel
Lloyd	Burch	Novell
Ed	Reed	Novell
Charles	Knouse	Oblix
Prateek	Mishra	Oracle
Vamsi	Motukuru	Oracle
Ramana	Turlapi	Oracle
Vipin	Samar	Oracle
Jerry	Schwarz	Oracle
Eric	Gravengaard	Reactivity
Andrew	Nash	Reactivity
Stuart	King	Reed Elsevier
Ben	Hammond	RSA Security
Rob	Philpott	RSA Security
Martijn	de Boer	SAP
Blake	Dournaee	Sarvega
Sundeep	Peechu	Sarvega

Coumara	Radja	Sarvega
Pete	Wenzel	SeeBeyond
Jonathan	Tourzan	Sony
Yassir	Elley	Sun
Manveen	Kaur	Sun Microsystems
Ronald	Monzillo	Sun Microsystems
Jan	Alexander	Systinet
Michael	Nguyen	The IDA of Singapore
Don	Adams	TIBCO
Symon	Chang	TIBCO Software
John	Weiland	US Navy
Hans	Granqvist	VeriSign
Phillip	Hallam-Baker	VeriSign
Hemma	Prafullchandra	VeriSign
Morten	Jorgensen	Vordel

697

698

B. Revision History

699

Revision	Date	Editor	Changes Made
WD01	17-January-2011	Carlo Milono	Corrected/added hyperlinks where missing; added Status section
WD02	8-February-2011	Carlo Milono	Added Related Work to reflect v1.1.1 of the specs; changed References for SOAP Message Security to reflect v1.1.1; Changed WD# to 2; Added Date; Moved Current Members to Previous and added new Current Members; saved document under wd02; entered the Revision History Merged Old Current Contributors with Old Previous, created a New Current Contributors.
WD03	16-March-2011	David Turner	Corrected and updated links
WD04	23-March-2011	David Turner	Fixed namespace in example at Lines 645-646 and removed a few errant spaces.

700

701

702