



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

**OASIS Standard Committee Draft, 12
FebruarySeptember 2006**

Document identifier:

wss-v1.1-spec-~~eserrata-cd~~-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 Committee Draft incorporating errata changes to the OASIS Standard document** 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

35 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
36 might be claimed to pertain to the implementation or use of the technology described in this document or
37 the extent to which any license under such rights might or might not be available; neither does it represent
38 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
39 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
40 available for publication and any assurances of licenses to be made available, or the result of an attempt
41 made to obtain a general license or permission for the use of such proprietary rights by implementors or
42 users of this specification, can be obtained from the OASIS Executive Director.

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

46 Copyright (C) OASIS Open 2004-2006. All Rights Reserved.

47 This document and translations of it may be copied and furnished to others, and derivative works that
48 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
49 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
50 this paragraph are included on all such copies and derivative works. However, this document itself may
51 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
52 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
53 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
54 into languages other than English.

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

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

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

63 Table of Contents

64	1 Introduction.....	4
65	2 Notations and Terminology.....	6
66	2.1 Notational Conventions.....	6
67	2.1.1 Namespaces.....	6
68	2.1.2 Acronyms and Abbreviations.....	7
69	2.2 Normative References.....	7
70	2.3 Non-normative References.....	8
71	3 MIME Processing.....	9
72	4 XML Attachments.....	10
73	5 Securing SOAP With Attachments.....	11
74	5.1 Primary SOAP Envelope.....	11
75	5.2 Referencing Attachments.....	11
76	5.3 MIME Part Reference Transforms.....	12
77	5.3.1 Attachment-Content-Signature-Transform.....	12
78	5.3.2 Attachment-Complete-Signature-Transform.....	12
79	5.3.3 Attachment-Ciphertext-Transform.....	13
80	5.4 Integrity and Data Origin Authentication	13
81	5.4.1 MIME header canonicalization.....	13
82	5.4.2 MIME Content Canonicalization.....	15
83	5.4.3 Protecting against attachment insertion threat.....	15
84	5.4.4 Processing Rules for Attachment Signing.....	15
85	5.4.5 Processing Rules for Attachment Signature Verification.....	16
86	5.4.6 Example Signed Message.....	16
87	5.5 Encryption.....	17
88	5.5.1 MIME Part CipherReference.....	18
89	5.5.2 Encryption Processing Rules.....	18
90	5.5.3 Decryption Processing Rules.....	19
91	5.5.4 Example.....	20
92	5.6 Signing and Encryption.....	21

1 Introduction

93

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

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

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

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

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

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

113 Goals of this profile include the following:

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

121 Non-goals include:

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

133 The existence of this profile does not preclude using other mechanisms to secure attachments conveyed
134 in conjunction with SOAP messages, including the use of XML security technologies at the application

135 layer or the use of security for the XML Infoset before a serialization that uses attachment technology
136 [MTOM]. The requirements in this profile only apply when securing SwA attachments explicitly according
137 to this profile.

2 Notations and Terminology

138

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

2.1 Notational Conventions

140

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

144

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

145

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

146

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

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

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

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

2.1.1 Namespaces

157

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

163 The namespaces used in this document are shown in the following table (note that for brevity, the
164 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#

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

166 2.1.2 Acronyms and Abbreviations

167 The following (non-normative) table defines acronyms and abbreviations for this document, beyond those
168 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]

169 2.2 Normative References

- 170 [RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
171 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- 172 [CHARSETS] Character sets assigned by IANA. See [ftp://ftp.isi.edu/in-](ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets)
173 [notes/iana/assignments/character-sets](ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets).
- 174 [Excl-Canon] "Exclusive XML Canonicalization, Version 1.0", W3C Recommendation, 18 July
175 2002. <http://www.w3.org/TR/xml-exc-c14n/>.
- 176 [RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet
177 Message Bodies", IETF RFC 2045, November 1996,
178 <http://www.ietf.org/rfc/rfc2045.txt>.
- 179 [RFC2046] "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", IETF
180 RFC 2046, November 1996, <http://www.ietf.org/rfc/rfc2046.txt>.
- 181 [RFC2047] "Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header
182 Extensions for Non-ASCII Text", IETF RFC 2047, November 1996,
183 <http://www.ietf.org/rfc/rfc2047.txt>.
- 184 [RFC2048] "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration
185 Procedures", <http://www.ietf.org/rfc/rfc2048.txt>.
- 186 [RFC2049] "Multipurpose Internet Mail Extensions(MIME) Part Five: Conformance Criteria
187 and Examples", <http://www.ietf.org/rfc/rfc2049.txt>.
- 188 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", IETF
189 RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 190 [RFC2184] P. Resnick, "MIME Parameter Value and Encoded Word Extensions: Character
191 Sets, Languages, and Continuations", IETF RFC 2184, August 1997,
192 <http://www.ietf.org/rfc/rfc2184.txt>.
- 193 [RFC2392] E. Levinson, "Content-ID and Message-ID Uniform Resource Locators", IETF
194 RFC 2392, <http://www.ietf.org/rfc/rfc2392.txt>.
- 195 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI):
196 Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August
197 1998, <http://www.ietf.org/rfc/rfc2396.txt>.
- 198 [RFC2557] "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", IETF
199 RFC 2557, March 1999, <http://www.ietf.org/rfc/rfc2557.txt>.
- 200 [RFC2633] Ramsdell B., "S/MIME Version 3 Message Specification", Standards Track RFC
201 2633, June 1999. <http://www.ietf.org/rfc/rfc2633.txt>.
- 202 [RFC2822] "Internet Message Format", IETF RFC 2822, April 2001,
203 <http://www.ietf.org/rfc/rfc2822.txt>.
- 204 [SECGLO] "Internet Security Glossary," Informational RFC 2828, May 2000.

205	[SOAP11]	"SOAP: Simple Object Access Protocol 1.1", W3C Note, 08 May 2000.
206	[SwA]	"SOAP Messages with Attachments", W3C Note, 11 December 2000,
207		http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211 .
208	[WS-I-AP]	"Attachments Profile Version 1.0", <i>Final Material</i> , 2004-08-24, http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html .
209		
210	[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 .
211		
212		
213	[XML-Schema]	W3C Recommendation, "XML Schema Part 1: Structures," 2 May 2001,
214		http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/ .
215		W3C Recommendation, "XML Schema Part 2: Datatypes," 2 May 2001,
216		http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ .
217	[XML-Sig]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February
218		2002, http://www.w3.org/TR/xmlsig-core/ .
219	[XPath]	W3C Recommendation, "XML Path Language", 16 November 1999,
220		http://www.w3.org/TR/xpath .

221 2.3 Non-normative References

222	[DecryptT]	M. Hughes et al, "Decryption Transform for XML Signature", W3C
223		Recommendation, 10 December 2002. http://www.w3.org/TR/xmlenc-decrypt/ .
224	[MTOM]	"SOAP Message Transmission Optimization Mechanism", W3C
225		Recommendation, 25 January 2005, http://www.w3.org/TR/soap12-mtom/ .

3 MIME Processing

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

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

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

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

265

4 XML Attachments

266
267
268

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.

269
270
271
272
273

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.

274
275
276
277
278

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.

279
280

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.

281 5 Securing SOAP With Attachments

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

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

290 5.1 Primary SOAP Envelope

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

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

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

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

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

```
302 --xyl
303 Content-Type: text/xml; charset=utf-8
304 Content-ID: <foo>

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

307 5.2 Referencing Attachments

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

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

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

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

321 **5.3 MIME Part Reference Transforms**

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

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

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

330 **5.3.1 Attachment-Content-Signature-Transform**

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

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

335 When this transform is used the content of the MIME part should be canonicalized as defined in section
336 [54.4.2](#).

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

340 The output of the transform is an octet stream consisting of the canonicalized serialization of the
341 attachment content. All of the MIME headers associated with the MIME part are ignored and not included
342 in the output octet stream. The canonicalization of the content is described in section [54.4.2](#) of this
343 specification.

344 **5.3.2 Attachment-Complete-Signature-Transform**

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

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

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

- 351 • Content-Description
- 352 • Content-Disposition
- 353 • Content-ID
- 354 • Content-Location
- 355 • Content-Type

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

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

361 When this transform is used the MIME headers should be canonicalized as defined in section 54.4.1 and
362 the MIME content should be canonicalized as defined in section 54.4.2.

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

366 The output of the transform is an octet stream consisting of concatenation of the MIME canonicalized
367 MIME headers selected by the transform followed by the canonicalized attachment content. The
368 canonicalization of headers and content are described in sections 54.4.1 and 54.4.2 of this specification.

369 5.3.3 Attachment-Ciphertext-Transform

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

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

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

378 The output of the transform is an octet stream consisting of the ciphertext as conveyed in the MIME part
379 content. All of the MIME headers associated with the MIME part are ignored and not included in the output
380 octet stream. The MIME text canonicalization of the content is described in section 54.4.2 of this
381 specification.

382 5.4 Integrity and Data Origin Authentication

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

390 5.4.1 MIME header canonicalization

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

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

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

- 399 without a Content-Type header MUST be treated as having a Content-Type header with the value
400 "Content-Type: text/plain; charset=us-ascii". MIME headers not listed in the Attachment-Complete
401 transform section of this specification are to be ignored by the transform.
- 402 3. The MIME headers MUST be processed by the Attachment-Complete transform in lexicographic order
403 (ascending).
 - 404 4. The MIME header names MUST be processed by the transform as having the case according to the
405 MIME specifications (as shown in the Attachment-Complete section).
 - 406 5. The MIME header values MUST be unfolded [[RFC2822](#)].
 - 407 6. Any Content-Description MIME header containing RFC2047 encoding MUST be decoded [[RFC2047](#)].
 - 408 7. When a Content-ID header is processed, the "<>" characters associated with the msg-id MUST be
409 included in the transform input. The reason is that although semantically these angle bracket
410 characters are not part of the msg-id (RFC 2822) they are a standard part of the header lexicographic
411 representation. If these characters are not integrity protected then an attacker could remove them
412 causing the CID transformation specified in RFC2392 to fail.
 - 413 8. Folding whitespace in structured MIME headers (e.g. Content-Disposition, Content-ID, Content-
414 Location, Content-Type) that is not within quotes MUST be removed. Folding whitespace in structured
415 MIME headers that is within quotes MUST be preserved. Folding whitespace in unstructured MIME
416 headers (e.g. Content-Description) MUST be preserved [[RFC2822](#)]. For example, whitespace
417 immediately following the colon delimiter in the structured Content-Type header MUST be removed, but
418 whitespace immediately following the colon delimiter in the unstructured Content-Description header
419 MUST be preserved.
 - 420 9. Comments in MIME header values MUST be removed [[RFC2822](#)].
 - 421 10. Case-insensitive MIME header values (e.g. media type/subtype values and disposition-type values)
422 MUST be converted to lowercase. Case-sensitive MIME header values MUST be left as is with
423 respect to case [[RFC2045](#)].
 - 424 11. Quoted characters other than double-quote and backslash ("\") in quoted strings in structured MIME
425 headers (e.g. Content-ID) MUST be unquoted. Double-quote and backslash ("\") characters in quoted
426 strings in structured MIME headers MUST be character encoded [[RFC2822](#)].
 - 427 12. Canonicalization of a MIME header MUST generate a UTF-8 encoded octet stream containing the
428 following: the MIME header name, a colon (":"), the MIME header value, and the result of
429 canonicalizing the MIME header parameters in lexicographic order (ascending) as described below.
 - 430 13. MIME header parameter names MUST be converted to lowercase [[RFC2045](#)].
 - 431 14. MIME parameter values containing RFC2184 character set, language, and continuations MUST be
432 decoded. The resulting canonical output MUST not contain the RFC2184 encoding [[RFC2184](#)].
 - 433 15. Case-insensitive MIME header parameter values MUST be converted to lowercase. Case-sensitive
434 MIME header parameter values MUST be left as is with respect to case [[RFC2045](#)].
 - 435 16. Enclosing double-quotes MUST be added to MIME header parameter values that do not already
436 contain enclosing quotes. Quoted characters other than double-quote and backslash ("\") in MIME
437 header parameter values MUST be unquoted. Double-quote and backslash characters in MIME
438 parameter values MUST be character encoded.
 - 439 17. Canonicalization of a MIME header parameter MUST generate a UTF-8 encoded octet stream
440 containing the following: a semi-colon (";"), the parameter name (lowercase), an equals sign ("="), and
441 the double-quoted parameter value.
 - 442 18. Each header MUST be terminated by a single CRLF pair, without any trailing whitespace.

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

444 **5.4.2 MIME Content Canonicalization**

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

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

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

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

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

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

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

469 **5.4.3 Protecting against attachment insertion threat**

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

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

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

479 **5.4.4 Processing Rules for Attachment Signing**

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

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

485 For each attachment Reference, perform the following steps:

- 486 1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as
487 outlined in section 54.4.2. Attachments of an XML content type require Exclusive XML Canonicalization
488 without comments[Excl-Canon].
- 489 2. If MIME headers are to be included in the signature, perform MIME header canonicalization as
490 outlined in section 54.4.1.
- 491 3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL
492 attribute value to this URL.
- 493 4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST
494 include a <ds:Transform> element with the Algorithm attribute having the full URL value specified
495 earlier in this profile – corresponding to either the Attachment-Complete-Signature-Transform or
496 Attachment-Content-Signature-Transform, depending on what is to be included in the hash calculation.
497 This MUST be the first transform listed. The <ds:Transform> element MUST NOT contain any
498 transform for a MIME transfer encoding purpose (e.g. base64 encoding) since transfer encoding is left
499 to the MIME layer as noted in section 2. This does not preclude the use of XML Transforms, including a
500 base64 transform, for other purposes.
- 501 5. Extract the appropriate portion of the MIME part consistent with the selected transform.
- 502 6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature
503 Recommendation.

504 5.4.5 Processing Rules for Attachment Signature Verification

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

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

- 509 1. Find the attachment corresponding to the <ds:Reference> URL attribute value. This value MUST
510 correspond to the Content-ID for the attachment[SwA].
- 511 2. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as
512 outlined in section 54.4.2. Attachments of an XML content type require Exclusive XML Canonicalization
513 without comments[Excl-Canon]. The MIME content to be MIME canonicalized MUST have had any
514 transfer-encoding processed at the MIME layer before this step is performed.
- 515 3. If MIME headers were included in the signature, perform MIME header canonicalization as outlined in
516 section 54.4.1.
- 517 4. Extract the appropriate portion of the MIME part according to the MIME Part Signature Transform
518 value.
- 519 5. Calculate the reference hash and verify the reference.

520 5.4.6 Example Signed Message

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

```

522 --BoundaryStr
523 Content-Type: text/xml

524 <S11:Envelope xmlns:S11="..." xmlns:wss="..." xmlns:wsu="..." xmlns:ds="..."
525 xmlns:xenc="...">
526   <S11:Header>
527     <wsse:Security>

528       <wsse:BinarySecurityToken wsu:Id="CertAssociatedWithSigningKey"
529         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
530 wss-soap-message-security-1.0#Base64Binary"
531         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
532 x509-token-profile-1.0#x509v3">
533         ...
534       </wsse:BinarySecurityToken>

535       <ds:Signature>
536         <ds:SignedInfo>
537           <ds:CanonicalizationMethod Algorithm=
538 'http://www.w3.org/2001/10/xml-exc-c14n#' />
539           <ds:SignatureMethod Algorithm=
540             'http://www.w3.org/2000/09/xmldsig#rsa-sha1' />
541           <ds:Reference URI="cid:bar">
542             <ds:Transforms>
543               <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
544 wss-SwAProfile-1.1#Attachment-Content-Signature-Transform"/>
545             </ds:Transforms>
546           <ds:DigestMethod Algorithm=
547             "http://www.w3.org/2000/09/xmldsig#sha1"/>
548           <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</ds:DigestValue>
549           </ds:Reference>
550         </ds:SignedInfo>
551         <ds:SignatureValue>DeadBeef</ds:SignatureValue>

552       <ds:KeyInfo>
553         <wsse:SecurityTokenReference>
554           <wsse:Reference URI="#CertAssociatedWithSigningKey"/>
555         </wsse:SecurityTokenReference>
556       </ds:KeyInfo>

557     </ds:Signature>
558   </wsse:Security>
559 </S11:Header>
560 <S11:Body>
561   some items
562 </S11:Body>
563 </S11:Envelope>
564 --BoundaryStr
565 Content-Type: image/png
566 Content-ID: <bar>
567 Content-Transfer-Encoding: base64

568 the image

```

569 5.5 Encryption

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

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

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

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

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

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

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

596 **5.5.1 MIME Part CipherReference**

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

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

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

606 **5.5.2 Encryption Processing Rules**

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

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

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

- 614 receiver should be able to process the <xenc:EncryptedKey> before the
615 <xenc:EncryptedData> element for the attachment.
- 616 2. Encrypt the attachment part using XML Encryption, according to the rules of XML Encryption. Encrypt
617 either the attachment including content and selected MIME headers or only the attachment content.
- 618 When encryption includes MIME headers, only the headers listed in this specification for the Attachment-
619 Complete-~~ReferenceSignature~~-Transform (Section 54.3.2) are to be included in the encryption. If a
620 header listed in the profile is present it MUST be included in the encryption. If a header is not listed in this
621 profile, then it MUST NOT be included in the encryption.
- 622 3. Set the <xenc:EncryptedData> Type attribute value to a URI that specifies adherence to this profile and
623 that specifies what was encrypted (MIME content or entire MIME part including headers). The following
624 URIs MUST be used for this purpose:
- 625 • Content Only:
- 626 `http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-`
627 `Only`
- 628 • Content and headers:
- 629 `http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete`
- 630 4. Set the <xenc:EncryptedData> MimeType attribute to match the attachment MIME part Content-Type
631 header before encryption when the Content-Only URI is specified for the Type attribute value. The
632 MimeType attribute value MAY be set when the AttachmentComplete Type attribute value is specified.
- 633 5. Optionally set the <xenc:EncryptedData> Encoding attribute to reflect the attachment content
634 encoding, as visible to the security layer at the time of encryption. This is advisory information to the
635 decryption security layer. It should be understood that this has no relation with the actual encoding that
636 could be performed independently by the MIME layer later for transfer purposes.
- 637 6. Set the <xenc:EncryptedData> <xenc:CipherReference> to the same reference URL for the attachment
638 that was used before encryption . This MUST be a CID scheme URL referring to the attachment part
639 Content-ID. Ensure this MIME header is in the part conveying the cipher data after encryption.
- 640 7. Include the Attachment-Ciphertext-Transform in the <xenc:CipherReference> <xenc:Transforms> list.
- 641 8. Prepend the <xenc:EncryptedData> element to the <wsse:Security> SOAP header block and then
642 prepend the associated optional <xenc:ReferenceList> element.
- 643 9. Update the attachment MIME part, replacing the original content with the cipher text generated by the
644 XML Encryption step.
- 645 10. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the
646 cipher data.

647 **5.5.3 Decryption Processing Rules**

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

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

- 653 The following decryption steps must be performed so that the result is as if they were performed in this
 654 order:
- 655 1. Extract the cipher text from the attachment referenced by the <xenc:CipherReference> URL attribute.
 656 The Attachment-Ciphertext-Transform defined in this profile indicates that the MIME part content is
 657 extracted.
 - 658 2. Decrypt the cipher text using the information present in the appropriate <xenc:EncryptedData> element
 659 and possibly other out of band information, according to the XML Encryption Standard.
 - 660 3. If the <xenc:EncryptedData>Type attribute indicates that selected MIME headers were encrypted, then
 661 those MIME headers MUST be replaced by the result of decryption, as well as the MIME part content.
 - 662 4. If the <xenc:EncryptedData>Type attribute indicates that only the content of the MIME part was
 663 encrypted, then the cipher text content of the attachment part MUST be replaced by the result of
 664 decryption. In this case the MIME part Content-Type header value MUST be replaced by the
 665 <xenc:EncryptedData> MimeType attribute value.
 - 666 5. If the <xenc:EncryptedData> Encoding attribute is present then the decryption security layer may pass
 667 this advisory information to the application.

668 5.5.4 Example

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

```

671 Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"
672 --BoundaryStr
673 Content-Type: text/xml

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

680   <S11:Header>
681     <wsse:Security>

682       <wsse:BinarySecurityToken wsu:Id="Acert"
683         EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
684 wss-soap-message-security-1.0#Base64Binary"
685         ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
686 x509-token-profile-1.0#x509v3">
687         ...
688       </wsse:BinarySecurityToken>

689       <xenc:EncryptedKey Id='EK'>
690         <EncryptionMethod
691           Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
692         <ds:KeyInfo Id="keyinfo">
693           <wsse:SecurityTokenReference>
694             <ds:X509Data>
695               <ds:X509IssuerSerial>
696                 <ds:X509IssuerName>
697                   DC=ACMECorp, DC=com
698                 </ds:X509IssuerName>
699                 <ds:X509SerialNumber>12345678</X509SerialNumber>
700               </ds:X509IssuerSerial>

```

```

701     </ds:X509Data>
702     </wsse:SecurityTokenReference>
703 </ds:KeyInfo>
704 <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
705 <ReferenceList>
706     <DataReference URI='#EA' />
707     <DataReference URI='#ED' />
708 </ReferenceList>
709 </EncryptedKey>

710 <xenc:EncryptedData
711     Id='EA'
712     Type="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-
713 1.1#Attachment-Content-Only"
714     MimeType="image/png">
715     <xenc:EncryptionMethod
716         Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc' />
717     <xenc:CipherData>
718         <xenc:CipherReference URI=cid:bar">
719             <xenc:Transforms>
720                 <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
721 wss-SwAProfile-1.1#Attachment-Ciphertext-Transform"/>
722             </xenc:Transforms>
723         </xenc:CipherReference>
724     </xenc:CipherData>
725 </xenc:EncryptedData>

726 </wsse:Security>
727 </S11:Header>
728 <S11:Body>
729     <xenc:EncryptedData Id='ED'
730     <xenc:EncryptionMethod
731         Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc' />
732     <xenc:CipherData>
733         <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
734     </xenc:CipherData>
735 </xenc:EncryptedData>
736 </S11:Body>
737 </S11:Envelope>
738 --BoundaryStr
739 Content-Type: application/octet-stream
740 Content-ID: <bar>
741 Content-Transfer-Encoding: binary

742 BinaryCipherData

```

743 5.6 Signing and Encryption

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

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

A. Acknowledgments

758 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

