



1 Web Services Security

2 SOAP Messages with Attachments

3 (SwA) Profile 1.1

4 **OASIS Committee Specification, 14 November 2005**

5 **Document identifier:**

6 wss-v1.1-spec-cs-SwAProfile

7 **Location:**

8 <http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1>

9 **Technical Committee:**

10 OASIS Web Services Security (WSS) TC

11 **Chair(s):**

12 Kelvin Lawrence, IBM

13 Chris Kaler, Microsoft

14 **Editors:**

15 Frederick Hirsch, Nokia

16 **Abstract:**

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

19 **Status:**

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

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

27 For information on whether any patents have been disclosed that may be essential to
28 implementing this specification, and any offers of patent licensing terms, please refer to the
29 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
30 at www.oasis-open.org/committees/wss.
31

Notices

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

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

44 Copyright (C) OASIS Open 2004-2005. All Rights Reserved.

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

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

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

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

61 **Table of Contents**

62	1 Introduction.....	4
63	2 Notations and Terminology.....	6
64	2.1 Notational Conventions.....	6
65	2.1.1 Namespaces.....	6
66	2.1.2 Acronyms and Abbreviations.....	7
67	2.2 Normative References.....	7
68	2.3 Non-normative References.....	8
69	3 MIME Processing.....	9
70	4 XML Attachments.....	10
71	5 Securing SOAP With Attachments.....	11
72	5.1 Primary SOAP Envelope.....	11
73	5.2 Referencing Attachments.....	11
74	5.3 MIME Part Reference Transforms.....	12
75	5.3.1 Attachment-Content-Signature-Transform.....	12
76	5.3.2 Attachment-Complete-Signature-Transform.....	12
77	5.3.3 Attachment-Ciphertext-Transform.....	13
78	5.4 Integrity and Data Origin Authentication	13
79	5.4.1 MIME header canonicalization.....	13
80	5.4.2 MIME Content Canonicalization.....	15
81	5.4.3 Protecting against attachment insertion threat.....	15
82	5.4.4 Processing Rules for Attachment Signing.....	15
83	5.4.5 Processing Rules for Attachment Signature Verification.....	16
84	5.4.6 Example Signed Message.....	16
85	5.5 Encryption.....	17
86	5.5.1 MIME Part CipherReference.....	18
87	5.5.2 Encryption Processing Rules.....	18
88	5.5.3 Decryption Processing Rules.....	19
89	5.5.4 Example.....	20
90	5.6 Signing and Encryption.....	21

1 Introduction

91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133

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

134 [MTOM]. The requirements in this profile only apply when securing SwA attachments explicitly according
135 to this profile.

2 Notations and Terminology

136

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

137

2.1 Notational Conventions

138

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

139

140

141

Listings of productions or other normative code appear like this.

142

Example code listings appear like this.

143

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

144

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

145

146

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

147

148

149

150

151

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

152

153

154

2.1.1 Namespaces

155

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

156

157

158

159

160

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

161

162

Prefix	Namespace
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

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

163

164 2.1.2 Acronyms and Abbreviations

165 The following (non-normative) table defines acronyms and abbreviations for this document, beyond those
166 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]

167 2.2 Normative References

168	[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 .
169		
170	[CHARSETS]	Character sets assigned by IANA. See ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets .
171		
172	[Excl-Canon]	"Exclusive XML Canonicalization, Version 1.0", W3C Recommendation, 18 July 2002. http://www.w3.org/TR/xml-exc-c14n/ .
173		
174	[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 .
175		
176		
177	[RFC2046]	"Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", IETF RFC 2046, November 1996, http://www.ietf.org/rfc/rfc2046.txt .
178		
179	[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 .
180		
181		
182	[RFC2048]	"Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", http://www.ietf.org/rfc/rfc2048.txt .
183		
184	[RFC2049]	"Multipurpose Internet Mail Extensions(MIME) Part Five: Conformance Criteria and Examples", http://www.ietf.org/rfc/rfc2049.txt .
185		
186	[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 .
187		
188	[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 .
189		
190		
191	[RFC2392]	E. Levinson, "Content-ID and Message-ID Uniform Resource Locators", IETF RFC 2392, http://www.ietf.org/rfc/rfc2392.txt .
192		
193	[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 .
194		
195		
196	[RFC2557]	"MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", IETF RFC 2557, March 1999, http://www.ietf.org/rfc/rfc2557.txt .
197		
198	[RFC2633]	Ramsdell B., "S/MIME Version 3 Message Specification", Standards Track RFC 2633, June 1999. http://www.ietf.org/rfc/rfc2633.txt .
199		
200	[RFC2822]	"Internet Message Format", IETF RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt .
201		
202	[SECGLO]	"Internet Security Glossary," Informational RFC 2828, May 2000.
203	[SOAP11]	"SOAP: Simple Object Access Protocol 1.1", W3C Note, 08 May 2000.
204	[SwA]	"SOAP Messages with Attachments", W3C Note, 11 December 2000, http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211 .
205		

206 **[WS-I-AP]** “Attachments Profile Version 1.0”, *Final Material*, 2004-08-24, <http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>.
207

208 **[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>.
209
210

211 **[XML-Schema]** W3C Recommendation, “XML Schema Part 1: Structures,” 2 May 2001, <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.
212

213 W3C Recommendation, “XML Schema Part 2: Datatypes,” 2 May 2001, <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.
214

215 **[XML-Sig]** W3C Recommendation, “XML-Signature Syntax and Processing”, 12 February 2002, <http://www.w3.org/TR/xmlsig-core/>.
216

217 **[XPath]** W3C Recommendation, “XML Path Language”, 16 November 1999, <http://www.w3.org/TR/xpath>.
218

219 **2.3 Non-normative References**

220 **[DecryptT]** M. Hughes et al, “Decryption Transform for XML Signature”, W3C Recommendation, 10 December 2002. <http://www.w3.org/TR/xmlenc-decrypt/>.
221

222 **[MTOM]** “SOAP Message Transmission Optimization Mechanism”, W3C Recommendation, 25 January 2005, <http://www.w3.org/TR/soap12-mtom/>.
223

224

3 MIME Processing

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

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

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

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

263

4 XML Attachments

264
265
266

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.

267
268
269
270
271

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.

272
273
274
275
276

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.

277
278

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.

279 5 Securing SOAP With Attachments

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

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

288 5.1 Primary SOAP Envelope

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

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

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

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

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

```
300 --xyl  
301 Content-Type: text/xml; charset=utf-8  
302 Content-ID: <foo>  
  
303 <?xml version='1.0' ?>  
304 <s11:Envelope xmlns:s11="http://schemas.xmlsoap.org/soap/envelope/" />
```

305 5.2 Referencing Attachments

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

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

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

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

319 **5.3 MIME Part Reference Transforms**

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

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

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

328 **5.3.1 Attachment-Content-Signature-Transform**

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

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

333 When this transform is used the content of the MIME part should be canonicalized as defined in section
334 4.4.2.

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

338 The output of the transform is an octet stream consisting of the canonicalized serialization of the
339 attachment content. All of the MIME headers associated with the MIME part are ignored and not included
340 in the output octet stream. The canonicalization of the content is described in section 4.4.2 of this
341 specification.

342 **5.3.2 Attachment-Complete-Signature-Transform**

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

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

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

- 349 • Content-Description
- 350 • Content-Disposition
- 351 • Content-ID
- 352 • Content-Location
- 353 • Content-Type

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

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

359 When this transform is used the MIME headers should be canonicalized as defined in section 4.4.1 and
360 the MIME content should be canonicalized as defined in section 4.4.2.

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

364 The output of the transform is an octet stream consisting of concatenation of the MIME canonicalized
365 MIME headers selected by the transform followed by the canonicalized attachment content. The
366 canonicalization of headers and content are described in sections 4.4.1 and 4.4.2 of this specification.

367 **5.3.3 Attachment-Ciphertext-Transform**

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

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

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

376 The output of the transform is an octet stream consisting of the ciphertext as conveyed in the MIME part
377 content. All of the MIME headers associated with the MIME part are ignored and not included in the output
378 octet stream. The MIME text canonicalization of the content is described in section 4.4.2 of this
379 specification.

380 **5.4 Integrity and Data Origin Authentication**

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

388 **5.4.1 MIME header canonicalization**

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

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

- 394 1. The transform MUST process MIME headers before the MIME content.
- 395 2. The transform MUST only process MIME headers that are explicitly present in the attachment part and
396 are listed in the Attachment-Complete transform section of this specification, except that a MIME part
397 without a Content-Type header MUST be treated as having a Content-Type header with the value

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

442 **5.4.2 MIME Content Canonicalization**

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

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

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

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

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

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

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

467 **5.4.3 Protecting against attachment insertion threat**

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

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

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

477 **5.4.4 Processing Rules for Attachment Signing**

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

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

483 For each attachment Reference, perform the following steps:

- 484 1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as
485 outlined in section 4.4.2 Attachments of an XML content type require Exclusive XML Canonicalization
486 without comments[[Excl-Canon](#)].
- 487 2. If MIME headers are to be included in the signature, perform MIME header canonicalization as
488 outlined in section 4.4.1.
- 489 3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL
490 attribute value to this URL.
- 491 4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST
492 include a <ds:Transform> element with the Algorithm attribute having the full URL value specified
493 earlier in this profile – corresponding to either the Attachment-Complete-Signature-Transform or
494 Attachment-Content-Signature-Transform, depending on what is to be included in the hash calculation.
495 This MUST be the first transform listed. The <ds:Transform> element MUST NOT contain any
496 transform for a MIME transfer encoding purpose (e.g. base64 encoding) since transfer encoding is left
497 to the MIME layer as noted in section 2. This does not preclude the use of XML Transforms, including
498 a base64 transform, for other purposes.
- 499 5. Extract the appropriate portion of the MIME part consistent with the selected transform.
- 500 6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature
501 Recommendation.

502 **5.4.5 Processing Rules for Attachment Signature Verification**

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

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

- 507 1. Find the attachment corresponding to the <ds:Reference> URL attribute value. This value MUST
508 correspond to the Content-ID for the attachment[[SwA](#)].
- 509 2. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part, as
510 outlined in section 4.4.2. Attachments of an XML content type require Exclusive XML Canonicalization
511 without comments[[Excl-Canon](#)]. The MIME content to be MIME canonicalized MUST have had any
512 transfer-encoding processed at the MIME layer before this step is performed.
- 513 3. If MIME headers were included in the signature, perform MIME header canonicalization as outlined in
514 section 4.4.1.
- 515 4. Extract the appropriate portion of the MIME part according to the MIME Part Signature Transform
516 value.
- 517 5. Calculate the reference hash and verify the reference.

518 **5.4.6 Example Signed Message**

```
519 Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"  
520 --BoundaryStr  
521 Content-Type: text/xml  
522 <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..."  
523 xmlns:xenc="...">  
524   <S11:Header>  
525     <wsse:Security>
```



```

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

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

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

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

```

567 5.5 Encryption

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

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

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

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

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

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

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

594 **5.5.1 MIME Part CipherReference**

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

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

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

604 **5.5.2 Encryption Processing Rules**

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

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

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

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

616 When encryption includes MIME headers, only the headers listed in this specification for the Attachment-
617 Complete Reference Transform (Section 4.3.2) are to be included in the encryption. If a header listed in
618 the profile is present it MUST be included in the encryption. If a header is not listed in this profile, then it
619 MUST NOT be included in the encryption.

- 620 3. Set the <xenc:EncryptedData> Type attribute value to a URI that specifies adherence to this profile and
621 that specifies what was encrypted (MIME content or entire MIME part including headers). The following
622 URIs MUST be used for this purpose:
- 623 • Content Only:
624 `http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Content-`
625 `Only`
 - 626 • Content and headers:
627 `http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-1.1#Attachment-Complete`
- 628 4. Set the <xenc:EncryptedData> MimeType attribute to match the attachment MIME part Content-Type
629 header before encryption when the Content-Only URI is specified for the Type attribute value. The
630 MimeType attribute value MAY be set when the AttachmentComplete Type attribute value is specified.
- 631 5. Optionally set the <xenc:EncryptedData> Encoding attribute to reflect the attachment content
632 encoding, as visible to the security layer at the time of encryption. This is advisory information to the
633 decryption security layer. It should be understood that this has no relation with the actual encoding that
634 could be performed independently by the MIME layer later for transfer purposes.
- 635 6. Set the <xenc:EncryptedData> <xenc:CipherReference> to the same reference URL for the
636 attachment that was used before encryption . This MUST be a CID scheme URL referring to the
637 attachment part Content-ID. Ensure this MIME header is in the part conveying the cipher data after
638 encryption.
- 639 7. Include the Attachment-Ciphertext-Transform in the <xenc:CipherReference> <xenc:Transforms> list.
- 640 8. Prepend the <xenc:EncryptedData> element to the <wsse:Security> SOAP header block and then
641 prepend the associated optional <xenc:ReferenceList> element.
- 642 9. Update the attachment MIME part, replacing the original content with the cipher text generated by the
643 XML Encryption step.
- 644 10. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the
645 cipher data.

646 **5.5.3 Decryption Processing Rules**

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

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

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

- 654 1. Extract the cipher text from the attachment referenced by the <xenc:CipherReference> URL attribute.
655 The Attachment-Ciphertext-Transform defined in this profile indicates that the MIME part content is
656 extracted.
- 657 2. Decrypt the cipher text using the information present in the appropriate <xenc:EncryptedData> element
658 and possibly other out of band information, according to the XML Encryption Standard.
- 659 3. If the <xenc:EncryptedData>Type attribute indicates that selected MIME headers were encrypted, then
660 those MIME headers MUST be replaced by the result of decryption, as well as the MIME part content.

- 661 4. If the <xenc:EncryptedData>Type attribute indicates that only the content of the MIME part was
 662 encrypted, then the cipher text content of the attachment part MUST be replaced by the result of
 663 decryption. In this case the MIME part Content-Type header value MUST be replaced by the
 664 <xenc:EncryptedData> MimeType attribute value.
- 665 5. If the <xenc:EncryptedData> Encoding attribute is present then the decryption security layer may pass
 666 this advisory information to the application.

667 5.5.4 Example

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

```

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

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

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

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

690     <xenc:EncryptedKey Id='EK'>
691       <EncryptionMethod
692         Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
693       <ds:KeyInfo Id="keyinfo">
694         <wsse:SecurityTokenReference>
695           <ds:X509Data>
696             <ds:X509IssuerSerial>
697               <ds:X509IssuerName>
698                 DC=ACMECorp, DC=com
699               </ds:X509IssuerName>
700             <ds:X509SerialNumber>12345678</ds:X509SerialNumber>
701             </ds:X509IssuerSerial>
702           </ds:X509Data>
703         </wsse:SecurityTokenReference>
704       </ds:KeyInfo>
705       <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
706       <ReferenceList>
707         <DataReference URI='#EA' />
708         <DataReference URI='#ED' />
709       </ReferenceList>
710     </xenc:EncryptedKey>

711     <xenc:EncryptedData
712       Id='EA'
713       Type="http://docs.oasis-open.org/wss/oasis-wss-SwAProfile-
714   1.1#Attachment-Content-Only"
715       MimeType="image/png">

```

```

716     <xenc:EncryptionMethod
717         Algorithm='http://www.w3.org/2001/04/xmlenc#aes128-cbc' />
718     <xenc:CipherData>
719         <xenc:CipherReference URI="cid:bar">
720             <xenc:Transforms>
721                 <ds:Transform Algorithm="http://docs.oasis-open.org/wss/oasis-
722 wss-SwAProfile-1.1#Attachment-Ciphertext-Transform" />
723             </xenc:Transforms>
724         </xenc:CipherReference>
725     </xenc:CipherData>
726 </xenc:EncryptedData>

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

743 BinaryCipherData

```

744 5.6 Signing and Encryption

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

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

A. Acknowledgments

759 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

760 **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	Obliv

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

