



1

2

Web Services Security:

3

Interop 2 Scenarios

4

Working Draft 03, 26 Aug 2003

5

Document identifier:

6

wss-interop2-draft-03.doc

7

Location:

8

<http://www.oasis-open.org/committees/wss/>

9

Editor:

10

Hal Lockhart, BEA Systems <hlockhar@bea.com>

11

Contributors:

12

Chris Kaler, Microsoft <ckaler@microsoft.com>

13

Hal Lockhart, BEA Systems <hlockhar@bea.com>

14

Peter Dapkus, BEA Systems <pdapkus@bea.com>

15

Anthony Nadalin, IBM <drsecure@us.ibm.com>

16

Abstract:

17

This document documents the four scenarios to be used in the second WSS

18

Interoperability Event.

19

Status:

20

Committee members should send comments on this specification to the [wss@lists.oasis-](mailto:wss@lists.oasis-open.org)

21

[open.org](mailto:wss-open.org) list. Others should subscribe to and send comments to the [22](mailto:wss-</p></div><div data-bbox=)

comment@lists.oasis-open.org list. To subscribe, send an email message to [23](mailto:wss-</p></div><div data-bbox=)

comment-request@lists.oasis-open.org with the word "subscribe" as the body of the

24

message.

25

Table of Contents

26	Introduction	5
27	1.1 Terminology.....	5
28	2 Test Application	6
29	3 Scenario #4 Session Key.....	7
30	3.1 Agreements	7
31	3.1.1 SESSION-KEY-VALUE	7
32	3.1.2 CERT-VALUE	7
33	3.1.3 Signature Trust Root.....	7
34	3.2 Parameters.....	7
35	3.3 General Message Flow	7
36	3.4 First Message - Request.....	8
37	3.4.1 Message Elements and Attributes	8
38	3.4.2 Message Creation.....	8
39	3.4.3 Message Processing.....	9
40	3.4.4 Example (Non-normative)	10
41	3.5 Second Message - Response	11
42	3.5.1 Message Elements and Attributes	11
43	3.5.2 Message Creation.....	12
44	3.5.3 Message Processing.....	13
45	3.5.4 Example (Non-normative)	13
46	3.6 Other processing.....	14
47	3.6.1 Requester	14
48	3.6.2 Responder	14
49	3.7 Expected Security Properties.....	14
50	4 Scenario #5 – Overlapping Signatures	15
51	4.1 Agreements	15
52	4.1.1 CERT-VALUE	15
53	4.1.2 Signature Trust Root.....	15
54	4.2 Parameters.....	15
55	4.3 General Message Flow	15
56	4.4 First Message - Request.....	15
57	4.4.1 Message Elements and Attributes	15
58	4.4.2 Message Creation.....	16
59	4.4.3 Message Processing.....	17
60	4.4.4 Example (Non-normative)	18
61	4.5 Second Message - Response	19
62	4.5.1 Message Elements and Attributes	19
63	4.5.2 Message Creation.....	19
64	4.5.3 Message Processing.....	19
65	4.5.4 Example (Non-normative)	19
66	4.6 Other processing	20
67	4.6.1 Requester	20

68	4.6.2 Responder	20
69	4.7 Expected Security Properties.....	20
70	5 Scenario #6 – Encrypt and Sign	21
71	5.1 Agreements.....	21
72	5.1.1 CERT-VALUE	21
73	5.1.2 Signature Trust Root.....	21
74	5.2 Parameters.....	21
75	5.3 General Message Flow	21
76	5.4 First Message - Request.....	21
77	5.4.1 Message Elements and Attributes	21
78	5.4.2 Message Creation.....	22
79	5.4.3 Message Processing.....	24
80	5.4.4 Example (Non-normative).....	24
81	5.5 Second Message - Response.....	25
82	5.5.1 Message Elements and Attributes	25
83	5.5.2 Message Creation.....	26
84	5.5.3 Message Processing.....	27
85	5.5.4 Example (Non-normative).....	28
86	5.6 Other processing	29
87	5.6.1 Requester	29
88	5.6.2 Responder	29
89	5.7 Expected Security Properties.....	29
90	6 Scenario #7 – Signed Token.....	30
91	6.1 Agreements.....	30
92	6.1.1 CERT-VALUE	30
93	6.1.2 Signature Trust Root.....	30
94	6.2 Parameters.....	30
95	6.3 General Message Flow	30
96	6.4 First Message - Request.....	31
97	6.4.1 Message Elements and Attributes	31
98	6.4.2 Message Creation.....	32
99	6.4.3 Message Processing.....	33
100	6.4.4 Example (Non-normative).....	33
101	6.5 Second Message - Response.....	35
102	6.5.1 Message Elements and Attributes	35
103	6.5.2 Message Creation.....	36
104	6.5.3 Message Processing.....	37
105	6.5.4 Example (Non-normative).....	37
106	6.6 Other processing	39
107	6.6.1 Requester	39
108	6.6.2 Responder	39
109	6.7 Expected Security Properties.....	39
110	7 References.....	40
111	7.1 Normative	40

112	Appendix A. Ping Application WSDL File	41
113	Appendix B. Revision History	43
114	Appendix C. Notices	44
115		

116 Introduction

117 This document describes the four message exchanges to be tested during the second
118 interoperability event of the WSS TC. All four use the Request/Response Message Exchange
119 Pattern (MEP) with no intermediaries. All four invoke the same simple application. To avoid
120 confusion, they are called Scenario #4 through Scenario #7.

121 These scenarios are intended to test the interoperability of different implementations performing
122 common operations and to test the soundness of the various specifications and clarity and mutual
123 understanding of their meaning and proper application.

124 THESE SCENARIOS ARE NOT INTENDED TO REPRESENT REASONABLE OR USEFUL
125 PRACTICAL APPLICATIONS OF THE SPECIFICATIONS. THEY HAVE BEEN DESIGNED
126 PURELY FOR THE PURPOSES INDICATED ABOVE AND DO NOT NECESSARILY
127 REPRESENT EFFICIENT OR SECURE MEANS OF PERFORMING THE INDICATED
128 FUNCTIONS. IN PARTICULAR THESE SCENARIOS ARE KNOWN TO VIOLATE SECURITY
129 BEST PRACTICES IN SOME RESPECTS AND IN GENERAL HAVE NOT BEEN EXTENSIVELY
130 VETTED FOR ATTACKS.

131 1.1 Terminology

132 The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*,
133 and *optional* in this document are to be interpreted as described in [RFC2119].

134 **2 Test Application**

135 All three scenarios use the same, simple application.

136 The Requester sends a Ping element with a value of a string.

137 The Responder returns a PingResponse element with a value of the same string.

138 **3 Scenario #4 Session Key**

139 The Request Body contains data that has been signed and encrypted. The certificate used to
140 verify the signature is provided in the header. The symmetric key used to perform the encryption
141 is provided out-of-band. The Response Body is also signed and encrypted. The same symmetric
142 key is used to perform the encryption. The certificate used to verify the signature is provided out-
143 of-band.

144 **3.1 Agreements**

145 This section describes the agreements that must be made, directly or indirectly between parties
146 who wish to interoperate.

147 **3.1.1 SESSION-KEY-VALUE**

148 This is an opaque identifier indicating a symmetric key that has been previously agreed by
149 unspecified means.

150 **3.1.2 CERT-VALUE**

151 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
152 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
153 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of
154 digitalSignature.

155 **3.1.3 Signature Trust Root**

156 This refers generally to agreeing on at least one trusted key and any other certificates and
157 sources of revocation information sufficient to validate certificates sent for the purpose of
158 signature verification.

159 **3.2 Parameters**

160 This section describes parameters that are required to correctly create or process messages, but
161 not a matter of mutual agreement.

162 No parameters are required.

163 **3.3 General Message Flow**

164 This section provides a general overview of the flow of messages.

165 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
166 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
167 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
168 which is signed and then encrypted. The certificate for signing is included in the message. The
169 encryption is performed using a previously agreed session key.

170 The Responder decrypts the body and then verifies the signature. If no errors are detected it
171 returns the response signing and encrypting the message body. The response is also signed and
172 encrypted. The signing key is provided externally. The encryption is done using the same
173 previously agreed session key.

174 **3.4 First Message - Request**

175 **3.4.1 Message Elements and Attributes**

176 Items not listed in the following table MAY be present, but MUST NOT be marked with the
177 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
178 Items marked optional MAY be generated and MUST be processed if present. Items MUST
179 appear in the order specified, except as noted.

180

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
ReferenceList	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
Cipherdata	Mandatory

181

182 **3.4.2 Message Creation**

183 **3.4.2.1 Security**

184 The Security element MUST contain the mustUnderstand="1" attribute.

185 **3.4.2.2 ReferenceList**

186 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
187 refers to the encrypted body of the message.

188 **3.4.2.3 BinarySecurityToken**

189 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
190 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
191 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
192 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
193 value of digitalSignature. The Requester must have access to the private key corresponding to
194 the public key in the certificate.

195 **3.4.2.4 Signature**

196 The signature is over the entire SOAP body.

197 **3.4.2.4.1 SignedInfo**

198 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
199 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
200 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
201 MUST be SHA1.

202 **3.4.2.4.2 SignatureValue**

203 The SignatureValue MUST be calculated as specified by the specification, using the private key
204 corresponding to the public key specified in the certificate in the BinarySecurityToken.

205 **3.4.2.4.3 KeyInfo**

206 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
207 indicates the BinarySecurityToken containing the certificate which will be used for signature
208 verification.

209 **3.4.2.5 Timestamp**

210 The Created element within the Timestamp SHOULD contain the current local time at the sender
211 expressed in the UTC time zone.

212 **3.4.2.6 Body**

213 The body element MUST be first signed and then its contents encrypted.

214 **3.4.2.7 EncryptedData**

215 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
216 EncryptedKey.

217 The Type MUST have the value of #Content.

218 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
219 – CBC.

220 The KeyInfo MUST contain a KeyName which is the SESSION-KEY-VALUE.

221 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
222 using the specified algorithm.

223 **3.4.3 Message Processing**

224 This section describes the processing performed by the Responder. If an error is detected, the
225 Responder MUST cease processing the message and issue a Fault with a value of
226 FailedAuthentication.

227 3.4.3.1 Security

228 3.4.3.2 ReferenceList

229 The ReferenceList indicates the data to be decrypted.

230 3.4.3.3 Timestamp

231 The Timestamp element MUST be ignored.

232 3.4.3.4 Body

233 The contents of the body MUST first be decrypted and then the signature verified. If no errors are
234 detected, the body MUST be passed to the application.

235 3.4.3.5 EncryptedData

236 The message body contents contained in the EncryptedData, referenced by the ReferenceList
237 MUST be decrypted using the key identified by SESSION-KEY-VALUE, using the specified
238 algorithm.

239 3.4.3.6 BinarySecurityToken

240 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
241 authorized entity. The public key in the certificate MUST be retained for verification of the
242 signature.

243 3.4.3.7 Signature

244 The body after decryption, MUST be verified against the signature using the specified algorithms
245 and transforms and the retained public key.

246 3.4.4 Example (Non-normative)

247 Here is an example request.

```
248 <?xml version="1.0" encoding="utf-8" ?>
249 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
250 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
251 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
252 <soap:Header>
253 <wsse:Security soap:mustUnderstand="1"
254 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
255 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
256 <xenc:DataReference URI="#enc" />
257 </xenc:ReferenceList>
258 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
259 EncodingType="wsse:Base64Binary"
260 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
261 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
262 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
263 <SignedInfo>
264 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
265 />
266 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
267 <Reference URI="#body">
268 <Transforms>
269 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
270 </Transforms>
271 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
272 <DigestValue>QTV...dw=</DigestValue>
273 </Reference>
274 </SignedInfo>
275 <SignatureValue>H+x0...gUw=</SignatureValue>
```

276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302

```

<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#myCert" />
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
<wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="body"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
    cbc" />
    <xenc:KeyInfo>
      <xenc:KeyName>SessionKey</KeyName>
    </xenc:KeyInfo>
    <xenc:CipherData>
      <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</soap:Body>
</soap:Envelope>

```

303 3.5 Second Message - Response

304 3.5.1 Message Elements and Attributes

305 Items not listed in the following table MUST NOT be created or processed. Items marked
306 mandatory MUST be generated and processed. Items marked optional MAY be generated and
307 MUST be processed if present. Items MUST appear in the order specified, except as noted.
308

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
ReferenceList	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory

EncryptionMethod	Mandatory
KeyInfo	Mandatory
Cipherdata	Mandatory

309

310 **3.5.2 Message Creation**

311 **3.5.2.1 Security**

312 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
313 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

314 **3.5.2.2 ReferenceList**

315 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
316 refers to the encrypted body of the message.

317 **3.5.2.3 Signature**

318 The signature is over the entire SOAP body.

319 **3.5.2.3.1 SignedInfo**

320 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
321 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
322 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
323 MUST be SHA1.

324 **3.5.2.3.2 SignatureValue**

325 The SignatureValue MUST be calculated as specified by the specification, using the private key
326 corresponding to the public key specified by the CERT-VALUE.

327 **3.5.2.3.3 KeyInfo**

328 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
329 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
330 MUST have the value of CERT-VALUE.

331 **3.5.2.4 Timestamp**

332 The Created element within the Timestamp SHOULD contain the current local time at the sender
333 expressed in the UTC timezone.

334 **3.5.2.5 Body**

335 The body element MUST be first signed and then its contents encrypted.

336 **3.5.2.6 EncryptedData**

337 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
338 EncryptedKey.

339 The Type MUST have the value of #Content.

340 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
341 – CBC.

342 The KeyInfo MUST contain a KeyName which is the SESSION-KEY-VALUE.

343 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
344 using the specified algorithm.

345 **3.5.3 Message Processing**

346 This section describes the processing performed by the Responder. If an error is detected, the
347 Responder MUST cease processing the message and report the fault locally with a value of
348 FailedAuthentication.

349 **3.5.3.1 Security**

350 **3.5.3.2 ReferenceList**

351 The ReferenceList indicates the data to be decrypted

352 **3.5.3.3 Timestamp**

353 The Timestamp element MUST be ignored.

354 **3.5.3.4 Body**

355 The contents of the body MUST first be decrypted and then the signature verified.

356 **3.5.3.5 EncryptedData**

357 The message body contents contained in the EncryptedData, referenced by the ReferenceList
358 MUST be decrypted using the key identified by SESSION-KEY-VALUE, using the specified
359 algorithm

360 **3.5.3.6 Signature**

361 The body after decryption, MUST be verified against the signature using the specified algorithms
362 and transforms and the indicated public key.

363 **3.5.4 Example (Non-normative)**

364 Here is an example response.

```
365 <?xml version="1.0" encoding="utf-8" ?>
366 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
367 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
368 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
369 <soap:Header>
370 <wsse:Security soap:mustUnderstand="1"
371 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
372 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
373 <xenc:DataReference URI="#enc" />
374 </xenc:ReferenceList>
375 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
376 <SignedInfo>
377 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
378 />
379 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
380 <Reference URI="#body">
381 <Transforms>
382 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
383 </Transforms>
384 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
```

```
385     <DigestValue>KxW...5B=</DigestValue>
386   </Reference>
387 </SignedInfo>
388 <SignatureValue>8Hkd...al7=</SignatureValue>
389 <KeyInfo>
390   <wsse:SecurityTokenReference>
391     <wsse:KeyIdentifier
392 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
393   </wsse:SecurityTokenReference>
394 </KeyInfo>
395 </Signature>
396 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
397   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
398 </wsu:Timestamp>
399 </wsse:Security>
400 </soap:Header>
401 <soap:Body wsu:Id="body"
402 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
403   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
404     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
405     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
406 cbc" />
407     <xenc:KeyInfo>
408       <xenc:KeyName>SessionKey</KeyName>
409     </xenc:KeyInfo>
410     <xenc:CipherData>
411       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
412     </xenc:CipherData>
413   </xenc:EncryptedData>
414 </soap:Body>
415 </soap:Envelope>
```

416

417 **3.6 Other processing**

418 This section describes processing that occurs outside of generating or processing a message.

419 **3.6.1 Requester**

420 No additional processing is required.

421 **3.6.2 Responder**

422 No additional processing is required.

423 **3.7 Expected Security Properties**

424 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
425 of the request is protected against modification and interception. The response is Authenticated
426 and protected against modification and interception. Protection against interception in both
427 directions depends on the assumption that the session key has been previously agreed in a
428 secure fashion and that it cannot be guessed.

429 The Responder must not draw any inferences about what party encrypted the message, it
430 particular it should not be assumed it was the same party who signed it.

431 **4 Scenario #5 – Overlapping Signatures**

432 The Request Body contains data that has been signed twice. First the ticket element is signed.
433 The certificate used to verify this signature is provided out-of-band. Next the entire body is
434 signed. The certificate used to verify this signature is provided in the header. The Response Body
435 is not signed or encrypted.

436 **4.1 Agreements**

437 This section describes the agreements that must be made, directly or indirectly between parties
438 who wish to interoperate.

439 **4.1.1 CERT-VALUE**

440 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
441 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
442 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of
443 digitalSignature.

444 The Responder MUST have access to the Private key corresponding to the Public key in the
445 certificate.

446 **4.1.2 Signature Trust Root**

447 This refers generally to agreeing on at least one trusted key and any other certificates and
448 sources of revocation information sufficient to validate certificates sent for the purpose of
449 signature verification.

450 **4.2 Parameters**

451 This section describes parameters that are required to correctly create or process messages, but
452 not a matter of mutual agreement.

453 No parameters are required.

454 **4.3 General Message Flow**

455 This section provides a general overview of the flow of messages.

456 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
457 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
458 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
459 which is signed twice. First the first element of the body is signed. The certificate used to verify
460 this signature is provided out-of-band. Next the entire body is signed. The certificate for this
461 signature is included in the message. The Responder verifies both signatures. If no errors are
462 detected it returns the response without any signatures.

463 **4.4 First Message - Request**

464 **4.4.1 Message Elements and Attributes**

465 Items not listed in the following table MAY be present, but MUST NOT be marked with the
466 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
467 Items marked optional MAY be generated and MUST be processed if present. Items MUST
468 appear in the order specified, except as noted.

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory

470

471 **4.4.2 Message Creation**

472 **4.4.2.1 Security**

473 The Security element MUST contain the mustUnderstand="1" attribute.

474 **4.4.2.2 Signature**

475 This signature is over the first element of the SOAP body.

476 **4.4.2.2.1 SignedInfo**

477 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
 478 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the first element under
 479 the SOAP Body element. The only Transform specified MUST be Exclusive Canonicalization. The
 480 DigestMethod MUST be SHA1.

481 **4.4.2.2 SignatureValue**

482 The SignatureValue MUST be calculated as specified by the specification, using the private key
483 corresponding to the public key specified in the certificate identified by the KeyIdentifier CERT-
484 VALUE.

485 **4.4.2.3 KeyInfo**

486 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
487 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
488 MUST have the value of CERT-VALUE.

489 **4.4.2.3 BinarySecurityToken**

490 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
491 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
492 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
493 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
494 values of digitalSignature. The Requester must have access to the private key corresponding to
495 the public key in the certificate.

496 **4.4.2.4 Signature**

497 This signature is over the entire SOAP body.

498 **4.4.2.4.1 SignedInfo**

499 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
500 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
501 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
502 MUST be SHA1.

503 **4.4.2.4.2 SignatureValue**

504 The SignatureValue MUST be calculated as specified by the specification, using the private key
505 corresponding to the public key specified in the certificate in the BinarySecurityToken.

506 **4.4.2.4.3 KeyInfo**

507 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
508 indicates the BinarySecurityToken containing the certificate which will be used for signature
509 verification.

510 **4.4.2.5 Timestamp**

511 The Created element within the Timestamp SHOULD contain the current local time at the sender
512 expressed in the UTC time zone

513 **4.4.2.6 Body**

514 The body element MUST be signed twice. The body contains two Ping requests. The first
515 signature is over only the first Ping and the second signature is over the entire body.

516 **4.4.3 Message Processing**

517 This section describes the processing performed by the Responder. If an error is detected, the
518 Responder MUST cease processing the message and issue a Fault with a value of
519 FailedAuthentication.

520 4.4.3.1 Security

521 4.4.3.2 Signature

522 The certificate referred to by the KeyIdentifier MUST be validated. The Subject of the certificate
523 MUST be an authorized entity. The first element in the body MUST be verified against the
524 signature using the specified algorithms and transforms and the indicated public key.

525 4.4.3.3 BinarySecurityToken

526 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
527 authorized entity. The public key in the certificate MUST be retained for verification of the
528 signature.

529 4.4.3.4 Signature

530 The body MUST be verified against the signature using the specified algorithms and transforms
531 and the retained public key.

532 4.4.3.5 Timestamp

533 The Timestamp element MUST be ignored.

534 4.4.3.6 Body

535 After verifying both signatures, if no errors are detected, the body MUST be passed to the
536 application.

537 4.4.4 Example (Non-normative)

538 Here is an example request.

```
539 <?xml version="1.0" encoding="utf-8" ?>
540 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
541 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
542 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
543 <soap:Header>
544 <wsse:Security soap:mustUnderstand="1"
545 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
546 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
547 <SignedInfo>
548 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
549 />
550 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
551 <Reference URI="#body">
552 <Transforms>
553 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
554 </Transforms>
555 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
556 <DigestValue>AXK...Fe=</DigestValue>
557 </Reference>
558 </SignedInfo>
559 <SignatureValue>MQwx...agv=</SignatureValue>
560 <KeyInfo>
561 <wsse:SecurityTokenReference>
562 <wsse:KeyIdentifier
563 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
564 </wsse:SecurityTokenReference>
565 </KeyInfo>
566 </Signature>
567 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
568 EncodingType="wsse:Base64Binary"
569 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
570 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
571 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
```

572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603

```
<SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <Reference URI="#tick">
    <Transforms>
      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <DigestValue>QTV...dw</DigestValue>
  </Reference>
</SignedInfo>
<SignatureValue>H+x0...gUw</SignatureValue>
<KeyInfo>
  <wsse:SecurityTokenReference>
    <wsse:Reference URI="#myCert" />
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
<wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
  <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="body">
  <Ping xmlns="http://xmlsoap.org/Ping">
    <text>Hello</text>
    <ticket wsu:Id="tick">1234567</ticket>
  </Ping>
</soap:Body>
</soap:Envelope>
```

604 4.5 Second Message - Response

605 4.5.1 Message Elements and Attributes

606 Items not listed in the following table MUST NOT be created or processed. Items marked
607 mandatory MUST be generated and processed. Items marked optional MAY be generated and
608 MUST be processed if present. Items MUST appear in the order specified, except as noted.
609

Name	Mandatory?
Body	Mandatory

610

611 4.5.2 Message Creation

612 The response message must not contain a <wsse:Security> header. Any other header elements
613 MUST NOT be labeled with a mustUnderstand="1" attribute.

614

615 4.5.3 Message Processing

616 The body is passed to the application without modification.

617 4.5.4 Example (Non-normative)

618 Here is an example response.

```
619 <?xml version="1.0" encoding="utf-8" ?>
620 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
621 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
622 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
623 <soap:Body>
624 <PingResponse xmlns="http://xmlsoap.org/Ping">
625 <text>Hello</text>
626 </PingResponse>
627 </soap:Body>
628 </soap:Envelope>
```

629 **4.6 Other processing**

630 This section describes processing that occurs outside of generating or processing a message.

631 **4.6.1 Requester**

632 No additional processing is required.

633 **4.6.2 Responder**

634 No additional processing is required.

635 **4.7 Expected Security Properties**

636 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
637 of the request is protected against modification. The response is not protected in any way.

638 **5 Scenario #6 – Encrypt and Sign**

639 The Request Body contains data that has been encrypted and signed. The certificate associated
640 with the encryption is provided out-of-band. The certificate used to verify the signature is provided
641 in the header. The Response Body is also encrypted and signed, reversing the roles of the key
642 pairs identified by the certificates.

643 **5.1 Agreements**

644 This section describes the agreements that must be made, directly or indirectly between parties
645 who wish to interoperate.

646 **5.1.1 CERT-VALUE**

647 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
648 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
649 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of
650 keyEncipherment, dataEncipherment and digitalSignature.

651 The Responder MUST have access to the Private key corresponding to the Public key in the
652 certificate.

653 **5.1.2 Signature Trust Root**

654 This refers generally to agreeing on at least one trusted key and any other certificates and
655 sources of revocation information sufficient to validate certificates sent for the purpose of
656 signature verification.

657 **5.2 Parameters**

658 This section describes parameters that are required to correctly create or process messages, but
659 not a matter of mutual agreement.

660 No parameters are required.

661 **5.3 General Message Flow**

662 This section provides a general overview of the flow of messages.

663 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
664 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
665 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
666 which is encrypted and then signed. The certificate for encryption is provided externally. The
667 certificate for signing is included in the message The Responder verifies the signature and then
668 decrypts the body. If no errors are detected it returns the response encrypting and signing the
669 message body. The roles of the key pairs are reversed from that of the request, using the
670 encryption key to sign and the signing key to encrypt.

671 **5.4 First Message - Request**

672 **5.4.1 Message Elements and Attributes**

673 Items not listed in the following table MAY be present, but MUST NOT be marked with the
674 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.

675 Items marked optional MAY be generated and MUST be processed if present. Items MUST
 676 appear in the order specified, except as noted.
 677

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

678

679 **5.4.2 Message Creation**

680 **5.4.2.1 Security**

681 The Security element MUST contain the mustUnderstand="1" attribute.

682 **5.4.2.2 BinarySecurityToken**

683 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
 684 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate

685 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
686 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
687 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have
688 access to the private key corresponding to the public key in the certificate.

689 **5.4.2.3 Signature**

690 The signature is over the entire SOAP body.

691 **5.4.2.3.1 SignedInfo**

692 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
693 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
694 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
695 MUST be SHA1.

696 **5.4.2.3.2 SignatureValue**

697 The SignatureValue MUST be calculated as specified by the specification, using the private key
698 corresponding to the public key specified in the certificate in the BinarySecurityToken.

699 **5.4.2.3.3 KeyInfo**

700 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
701 indicates the BinarySecurityToken containing the certificate which will be used for signature
702 verification.

703 **5.4.2.4 EncryptedKey**

704 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

705 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
706 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
707 MUST have the value of CERT-VALUE.

708 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
709 Key specified in the specified X.509 certificate, using the specified algorithm.

710 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
711 refers to the encrypted body of the message.

712 **5.4.2.5 Timestamp**

713 The Created element within the Timestamp SHOULD contain the current local time at the sender
714 expressed in the UTC time zone.

715 **5.4.2.6 Body**

716 The contents of the body element MUST be first encrypted and then the entire element signed.

717 **5.4.2.7 EncryptedData**

718 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
719 EncryptedKey.

720 The Type MUST have the value of #Content.

721 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
722 – CBC.

723 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
724 using the specified algorithm.

725 **5.4.3 Message Processing**

726 This section describes the processing performed by the Responder. If an error is detected, the
727 Responder MUST cease processing the message and issue a Fault with a value of
728 FailedAuthentication.

729 **5.4.3.1 Security**

730 **5.4.3.2 BinarySecurityToken**

731 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
732 authorized entity. The public key in the certificate MUST be retained for verification of the
733 signature.

734 **5.4.3.3 Signature**

735 The body after decryption, MUST be verified against the signature using the specified algorithms
736 and transforms and the retained public key.

737 **5.4.3.4 EncryptedKey**

738 The random key contained in the CipherData MUST be decrypted using the private key
739 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

740 **5.4.3.5 Timestamp**

741 The Timestamp element MUST be ignored.

742 **5.4.3.6 Body**

743 The signature over the body MUST first be verified decrypted and then its contents decrypted. If
744 no errors are detected, the body MUST be passed to the application.

745 **5.4.3.7 EncryptedData**

746 The message body contents contained in the EncryptedData, referenced by the ReferenceList
747 MUST be decrypted using the random key, using the specified algorithm.

748 **5.4.4 Example (Non-normative)**

749 Here is an example request.

```
750 <?xml version="1.0" encoding="utf-8" ?>
751 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
752 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
753 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
754 <soap:Header>
755 <wsse:Security soap:mustUnderstand="1"
756 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
757 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
758 EncodingType="wsse:Base64Binary"
759 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
760 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
761 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
762 <SignedInfo>
763 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
764 />
765 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
766 <Reference URI="#body">
767 <Transforms>
768 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
769 </Transforms>
```



```

770     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
771     <DigestValue>QTV...dw=</DigestValue>
772     </Reference>
773 </SignedInfo>
774 <SignatureValue>H+x0...gUw=</SignatureValue>
775 <KeyInfo>
776   <wsse:SecurityTokenReference>
777     <wsse:Reference URI="#myCert" />
778   </wsse:SecurityTokenReference>
779 </KeyInfo>
780 </Signature>
781 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
782   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
783 />
784   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
785     <wsse:SecurityTokenReference>
786       <wsse:KeyIdentifier
787 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
788     </wsse:SecurityTokenReference>
789   </KeyInfo>
790   <xenc:CipherData>
791     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
792   </xenc:CipherData>
793   <xenc:ReferenceList>
794     <xenc:DataReference URI="#enc" />
795   </xenc:ReferenceList>
796 </xenc:EncryptedKey>
797 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
798   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
799 </wsu:Timestamp>
800 </wsse:Security>
801 </soap:Header>
802 <soap:Body wsu:Id="body"
803 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
804   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
805     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
806     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
807 cbc" />
808     <xenc:CipherData>
809       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
810     </xenc:CipherData>
811   </xenc:EncryptedData>
812 </soap:Body>
813 </soap:Envelope>

```

814

815 5.5 Second Message - Response

816 5.5.1 Message Elements and Attributes

817 Items not listed in the following table MUST NOT be created or processed. Items marked
818 mandatory MUST be generated and processed. Items marked optional MAY be generated and
819 MUST be processed if present. Items MUST appear in the order specified, except as noted.

820

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Signature	Mandatory
SignedInfo	Mandatory

CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

821

822 **5.5.2 Message Creation**

823 **5.5.2.1 Security**

824 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
825 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

826 **5.5.2.2 Signature**

827 The signature is over the entire SOAP body.

828 **5.5.2.2.1 SignedInfo**

829 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
830 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body
831 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod
832 MUST be SHA1.

833 **5.5.2.2.2 SignatureValue**

834 The SignatureValue MUST be calculated as specified by the specification, using the private key
835 corresponding to the public key specified in the certificate in the BinarySecurityToken.

836 **5.5.2.2.3 KeyInfo**

837 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
838 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
839 MUST have the value of CERT-VALUE.

840 **5.5.2.3 BinarySecurityToken**

841 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
842 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent
843 in the request.

844 **5.5.2.4 EncryptedKey**

845 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

846 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
847 indicates the BinarySecurityToken containing the certificate which will be used for signature
848 verification.

849 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
850 Key specified in the specified X.509 certificate, using the specified algorithm.

851 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
852 refers to the encrypted body of the message.

853 **5.5.2.5 Timestamp**

854 The Created element within the Timestamp SHOULD contain the current local time at the sender
855 expressed in the UTC time zone.

856 **5.5.2.6 Body**

857 The contents of the body element MUST be first encrypted and then the entire element signed.

858 **5.5.2.7 EncryptedData**

859 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
860 EncryptedKey.

861 The Type MUST have the value of #Content.

862 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
863 – CBC.

864 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
865 using the specified algorithm.

866 **5.5.3 Message Processing**

867 This section describes the processing performed by the Responder. If an error is detected, the
868 Responder MUST cease processing the message and report the fault locally with a value of
869 FailedAuthentication.

870 **5.5.3.1 Security**

871 **5.5.3.2 Timestamp**

872 The Timestamp element MUST be ignored.

873 5.5.3.3 Body

874 The contents of the body MUST first be decrypted and then the signature verified.

875 5.5.3.4 EncryptedData

876 The message body contents contained in the EncryptedData, referenced by the ReferenceList
877 MUST be decrypted using the random key, using the specified algorithm.

878 5.5.3.5 Signature

879 The body after decryption, MUST be verified against the signature using the specified algorithms
880 and transforms and the indicated public key.

881 5.5.3.6 BinarySecurityToken

882 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
883 authorized entity. The certificate is used to identify the private key to be used for decryption.

884 5.5.3.7 EncryptedKey

885 The random key contained in the CipherData MUST be decrypted using the private key
886 corresponding to the certificate specified by the Reference, using the specified algorithm.

887 5.5.4 Example (Non-normative)

888 Here is an example response.

```
889 <?xml version="1.0" encoding="utf-8" ?>
890 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
891 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
892 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
893 <soap:Header>
894 <wsse:Security soap:mustUnderstand="1"
895 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
896 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
897 <SignedInfo>
898 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
899 />
900 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
901 <Reference URI="#body">
902 <Transforms>
903 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
904 </Transforms>
905 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
906 <DigestValue>KxW...5B=</DigestValue>
907 </Reference>
908 </SignedInfo>
909 <SignatureValue>8Hkd...a17=</SignatureValue>
910 <KeyInfo>
911 <wsse:SecurityTokenReference>
912 <wsse:KeyIdentifier
913 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
914 </wsse:SecurityTokenReference>
915 </KeyInfo>
916 </Signature>
917 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
918 EncodingType="wsse:Base64Binary"
919 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
920 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
921 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmenc#">
922 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#rsa-1_5"
923 />
924 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
925 <wsse:SecurityTokenReference>
926 <wsse:Reference URI="#myCert" />
```

```
927     </wsse:SecurityTokenReference>
928 </KeyInfo>
929 <xenc:CipherData>
930   <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
931 </xenc:CipherData>
932 <xenc:ReferenceList>
933   <xenc>DataReference URI="#enc" />
934 </xenc:ReferenceList>
935 </xenc:EncryptedKey>
936 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
937   <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
938 </wsu:Timestamp>
939 </wsse:Security>
940 </soap:Header>
941 <soap:Body wsu:Id="body"
942 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
943   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
944     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
945     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
946 cbc" />
947     <xenc:CipherData>
948       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
949     </xenc:CipherData>
950   </xenc:EncryptedData>
951 </soap:Body>
952 </soap:Envelope>
```

953

954 **5.6 Other processing**

955 This section describes processing that occurs outside of generating or processing a message.

956 **5.6.1 Requester**

957 No additional processing is required.

958 **5.6.2 Responder**

959 No additional processing is required.

960 **5.7 Expected Security Properties**

961 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
962 of the request is protected against modification and interception. The response is Authenticated
963 and protected against modification and interception. Note that the fact that the signature is over
964 the cyphertext may raise doubts as to whether the signing entity was aware what was signed.

965 The cleartext SignatureValue may also assist a known plaintext attack. The Responder must not
966 draw any inferences about what party encrypted the message, it particular it should not be
967 assumed it was the same party who signed it.

968 **6 Scenario #7 – Signed Token**

969 The Request Body contains data that has been signed and encrypted. The signature also
970 protects an enclosed Security Token by means of the STR Dereference Transform. The
971 certificate used to verify the signature is provided in the header. The certificate associated with
972 the encryption is provided out-of-band. The Response Body is also signed and encrypted,
973 reversing the roles of the key pairs identified by the certificates.

974 **6.1 Agreements**

975 This section describes the agreements that must be made, directly or indirectly between parties
976 who wish to interoperate.

977 **6.1.1 CERT-VALUE**

978 This is an opaque identifier indicating the X.509 certificate to be used. The certificate in question
979 MUST be obtained by the Requester by unspecified means. The certificate SHOULD NOT have a
980 KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of
981 keyEncipherment, dataEncipherment and digitalSignature.

982 The Responder MUST have access to the Private key corresponding to the Public key in the
983 certificate.

984 **6.1.2 Signature Trust Root**

985 This refers generally to agreeing on at least one trusted key and any other certificates and
986 sources of revocation information sufficient to validate certificates sent for the purpose of
987 signature verification.

988 **6.2 Parameters**

989 This section describes parameters that are required to correctly create or process messages, but
990 not a matter of mutual agreement.

991 No parameters are required.

992 **6.3 General Message Flow**

993 This section provides a general overview of the flow of messages.

994 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.
995 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a
996 null string may be used. The recipient SHOULD ignore the value. The request contains a body,
997 which is signed and then encrypted. The signature also covers the Token used for signing. The
998 certificate for signing is included in the message. The certificate for encryption is provided
999 externally. The Responder decrypts the body and then verifies the signature. If no errors are
1000 detected it returns the response signing and encrypting the message body. The roles of the key
1001 pairs are reversed from that of the request, using the signing key to encrypt and the encryption
1002 key to sign. The signature also covers the Token used for signing.

1003 **6.4 First Message - Request**

1004 **6.4.1 Message Elements and Attributes**

1005 Items not listed in the following table MAY be present, but MUST NOT be marked with the
 1006 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.
 1007 Items marked optional MAY be generated and MUST be processed if present. Items MUST
 1008 appear in the order specified, except as noted.

1009

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1010

1011 **6.4.2 Message Creation**

1012 **6.4.2.1 Security**

1013 The Security element MUST contain the mustUnderstand="1" attribute.

1014 **6.4.2.2 EncryptedKey**

1015 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1016 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
1017 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
1018 MUST have the value of CERT-VALUE.

1019 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
1020 Key specified in the specified X.509 certificate, using the specified algorithm.

1021 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
1022 refers to the encrypted body of the message.

1023 **6.4.2.3 BinarySecurityToken**

1024 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
1025 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate
1026 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT
1027 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the
1028 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have
1029 access to the private key corresponding to the public key in the certificate.

1030 **6.4.2.4 Signature**

1031 The signature is over the entire SOAP body.

1032 **6.4.2.4.1 SignedInfo**

1033 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
1034 be RSA-SHA1.

1035 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference
1036 contained in the Signature. The STR Dereference Transform with a parameter of the Exclusive
1037 Canonicalization Transform MUST be specified. The DigestMethod MUST be SHA1.

1038 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The
1039 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be
1040 SHA1.

1041 **6.4.2.4.2 SignatureValue**

1042 The SignatureValue MUST be calculated as specified by the specification, using the private key
1043 corresponding to the public key specified in the certificate in the BinarySecurityToken.

1044 **6.4.2.4.3 KeyInfo**

1045 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
1046 indicates the BinarySecurityToken containing the certificate which will be used for signature
1047 verification.

1048 **6.4.2.5 Timestamp**

1049 The Created element within the Timestamp SHOULD contain the current local time at the sender
1050 expressed in the UTC time zone.

1051 **6.4.2.6 Body**

1052 The body element MUST be first signed and then its contents encrypted.

1053 **6.4.2.7 EncryptedData**

1054 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
1055 EncryptedKey.

1056 The Type MUST have the value of #Content.

1057 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
1058 – CBC.

1059 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
1060 using the specified algorithm.

1061 **6.4.3 Message Processing**

1062 This section describes the processing performed by the Responder. If an error is detected, the
1063 Responder MUST cease processing the message and issue a Fault with a value of
1064 FailedAuthentication.

1065 **6.4.3.1 Security**

1066 **6.4.3.2 EncryptedKey**

1067 The random key contained in the CipherData MUST be decrypted using the private key
1068 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

1069 **6.4.3.3 Timestamp**

1070 The Timestamp element MUST be ignored.

1071 **6.4.3.4 Body**

1072 The contents of the body MUST first be decrypted and then the signature verified. If no errors are
1073 detected, the body MUST be passed to the application.

1074 **6.4.3.5 EncryptedData**

1075 The message body contents contained in the EncryptedData, referenced by the ReferenceList
1076 MUST be decrypted using the random key, using the specified algorithm.

1077 **6.4.3.6 BinarySecurityToken**

1078 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
1079 authorized entity. The public key in the certificate MUST be retained for verification of the
1080 signature.

1081 **6.4.3.7 Signature**

1082 The body after decryption, MUST be verified against the signature using the specified algorithms
1083 and transforms and the retained public key.

1084 **6.4.4 Example (Non-normative)**

1085 Here is an example request.

1086 `<?xml version="1.0" encoding="utf-8" ?>`

```

1087 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1088 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1089 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1090 <soap:Header>
1091 <wsse:Security soap:mustUnderstand="1"
1092 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1093 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1094 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1095 />
1096 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1097 <wsse:SecurityTokenReference>
1098 <wsse:KeyIdentifier
1099 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
1100 </wsse:SecurityTokenReference>
1101 </KeyInfo>
1102 <xenc:CipherData>
1103 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
1104 </xenc:CipherData>
1105 <xenc:ReferenceList>
1106 <xenc:DataReference URI="#enc" />
1107 </xenc:ReferenceList>
1108 </xenc:EncryptedKey>
1109 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1110 EncodingType="wsse:Base64Binary"
1111 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1112 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1113 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1114 <SignedInfo>
1115 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1116 />
1117 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1118 <Reference URI="#Token">
1119 <Transforms>
1120 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
1121 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1122 c14n#"/>
1123 </Transform>
1124 </Transforms>
1125 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1126 <DigestValue>pHrr...xK=</DigestValue>
1127 </Reference>
1128 <Reference URI="#body">
1129 <Transforms>
1130 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1131 </Transforms>
1132 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1133 <DigestValue>QTV...dw=</DigestValue>
1134 </Reference>
1135 </SignedInfo>
1136 <SignatureValue>H+x0...gUw=</SignatureValue>
1137 <KeyInfo>
1138 <wsse:SecurityTokenReference wsu:Id="Token">
1139 <wsse:Reference URI="#myCert" />
1140 </wsse:SecurityTokenReference>
1141 </KeyInfo>
1142 </Signature>
1143 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1144 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1145 </wsu:Timestamp>
1146 </wsse:Security>
1147 </soap:Header>
1148 <soap:Body wsu:Id="body"
1149 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1150 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1151 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1152 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
1153 cbc" />
1154 <xenc:CipherData>
1155 <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
1156 </xenc:CipherData>
1157 </xenc:EncryptedData>

```

1158
1159
1160

```
</soap:Body>  
</soap:Envelope>
```

1161

6.5 Second Message - Response

1162

6.5.1 Message Elements and Attributes

1163
1164
1165
1166

Items not listed in the following table MUST NOT be created or processed. Items marked mandatory MUST be generated and processed. Items marked optional MAY be generated and MUST be processed if present. Items MUST appear in the order specified, except as noted.

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
KeyIdentifier	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Reference	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Timestamp	Mandatory
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1167

1168 **6.5.2 Message Creation**

1169 **6.5.2.1 Security**

1170 The Security element MUST contain the mustUnderstand="1" attribute. Any other header
1171 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

1172 **6.5.2.2 BinarySecurityToken**

1173 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be
1174 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent
1175 in the request.

1176 **6.5.2.3 EncryptedKey**

1177 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1178 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which
1179 indicates the BinarySecurityToken containing the certificate which will be used for signature
1180 verification.

1181 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public
1182 Key specified in the specified X.509 certificate, using the specified algorithm.

1183 The ReferenceList MUST contain a DataReference which has the value of a relative URI that
1184 refers to the encrypted body of the message.

1185 **6.5.2.4 Signature**

1186 The signature is over the entire SOAP body.

1187 **6.5.2.4.1 SignedInfo**

1188 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST
1189 be RSA-SHA1.

1190 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference
1191 contained in the Signature. The STR Dereference Transform with a parameter of the Exclusive
1192 Canonicalization Transform MUST be specified. The DigestMethod MUST be SHA1.

1193 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The
1194 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be
1195 SHA1.

1196 **6.5.2.4.2 SignatureValue**

1197 The SignatureValue MUST be calculated as specified by the specification, using the private key
1198 corresponding to the public key specified in the certificate in the BinarySecurityToken.

1199 **6.5.2.4.3 KeyInfo**

1200 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST
1201 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier
1202 MUST have the value of CERT-VALUE.

1203 **6.5.2.5 Timestamp**

1204 The Created element within the Timestamp SHOULD contain the current local time at the sender
1205 expressed in the UTC time zone.

1206 **6.5.2.6 Body**

1207 The body element MUST be first signed and then its contents encrypted.

1208 **6.5.2.7 EncryptedData**

1209 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the
1210 EncryptedKey.

1211 The Type MUST have the value of #Content.

1212 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES
1213 – CBC.

1214 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,
1215 using the specified algorithm.

1216 **6.5.3 Message Processing**

1217 This section describes the processing performed by the Responder. If an error is detected, the
1218 Responder MUST cease processing the message and report the fault locally with a value of
1219 FailedAuthentication.

1220 **6.5.3.1 Security**

1221 **6.5.3.2 BinarySecurityToken**

1222 The certificate in the token MUST be validated. The Subject of the certificate MUST be an
1223 authorized entity. The certificate is used to identify the private key to be used for decryption.

1224 **6.5.3.3 EncryptedKey**

1225 The random key contained in the CipherData MUST be decrypted using the private key
1226 corresponding to the certificate specified by the Reference, using the specified algorithm.

1227 **6.5.3.4 Timestamp**

1228 The Timestamp element MUST be ignored.

1229 **6.5.3.5 Body**

1230 The contents of the body MUST first be decrypted and then the signature verified.

1231 **6.5.3.6 EncryptedData**

1232 The message body contents contained in the EncryptedData, referenced by the ReferenceList
1233 MUST be decrypted using the random key, using the specified algorithm.

1234 **6.5.3.7 Signature**

1235 The body after decryption, MUST be verified against the signature using the specified algorithms
1236 and transforms and the indicated public key.

1237 **6.5.4 Example (Non-normative)**

1238 Here is an example response.

```
1239 <?xml version="1.0" encoding="utf-8" ?>  
1240 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
1241 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1242 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```

1243 <soap:Header>
1244 <wsse:Security soap:mustUnderstand="1"
1245 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1246 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1247 EncodingType="wsse:Base64Binary"
1248 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1249 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1250 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1251 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1252 />
1253 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1254 <wsse:SecurityTokenReference>
1255 <wsse:Reference URI="#myCert" />
1256 </wsse:SecurityTokenReference>
1257 </KeyInfo>
1258 <xenc:CipherData>
1259 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
1260 </xenc:CipherData>
1261 <xenc:ReferenceList>
1262 <xenc:DataReference URI="#enc" />
1263 </xenc:ReferenceList>
1264 </xenc:EncryptedKey>
1265 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1266 <SignedInfo>
1267 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1268 />
1269 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1270 <Reference URI="#Token">
1271 <Transforms>
1272 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
1273 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1274 c14n#" />
1275 </Transform>
1276 </Transforms>
1277 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1278 <DigestValue>B4j...Xv</DigestValue>
1279 </Reference>
1280 <Reference URI="#body">
1281 <Transforms>
1282 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1283 </Transforms>
1284 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1285 <DigestValue>KxW...5B</DigestValue>
1286 </Reference>
1287 </SignedInfo>
1288 <SignatureValue>8Hkd...a17</SignatureValue>
1289 <KeyInfo>
1290 <wsse:SecurityTokenReference wsu:Id="Token">
1291 <wsse:KeyIdentifier
1292 ValueType="wsse:X509v3">B39R...mY</wsse:KeyIdentifier>
1293 </wsse:SecurityTokenReference>
1294 </KeyInfo>
1295 </Signature>
1296 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1297 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1298 </wsu:Timestamp>
1299 </wsse:Security>
1300 </soap:Header>
1301 <soap:Body wsu:Id="body"
1302 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1303 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1304 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1305 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
1306 cbc" />
1307 <xenc:CipherData>
1308 <xenc:CipherValue>d2s...GQ</xenc:CipherValue>
1309 </xenc:CipherData>
1310 </xenc:EncryptedData>
1311 </soap:Body>
1312 </soap:Envelope>

```

1313

1314 **6.6 Other processing**

1315 This section describes processing that occurs outside of generating or processing a message.

1316 **6.6.1 Requester**

1317 No additional processing is required.

1318 **6.6.2 Responder**

1319 No additional processing is required.

1320 **6.7 Expected Security Properties**

1321 Use of the service is restricted to authorized parties that sign the Body of the request. The Body
1322 of the request is protected against modification and interception. The response is Authenticated
1323 and protected against modification and interception. The signature over the signature token binds
1324 it to the message, preventing a repudiation attack by certificate substitution.

1325 The Responder must not draw any inferences about what party encrypted the message, it
1326 particular it should not be assumed it was the same party who signed it.

1327 **7 References**

1328 **7.1 Normative**

1329 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1330 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.

Appendix A. Ping Application WSDL File

```

1332 <definitions xmlns:tns="http://xmlsoap.org/Ping"
1333 xmlns="http://schemas.xmlsoap.org/wsdl/"
1334 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1335 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1336 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
1337   <types>
1338     <schema targetNamespace="http://xmlsoap.org/Ping"
1339     xmlns="http://www.w3.org/2001/XMLSchema">
1340       <complexType name="ping">
1341         <sequence>
1342           <element name="text" type="xsd:string"
1343 nillable="true"/>
1344           <element name="ticket" type="xsd:string"
1345 minOccurs="0"/>
1346         </sequence>
1347       </complexType>
1348       <complexType name="pingResponse">
1349         <sequence>
1350           <element name="text" type="xsd:string"
1351 nillable="true"/>
1352         </sequence>
1353       </complexType>
1354       <element name="Ping" type="tns:ping"/>
1355       <element name="PingResponse" type="tns:pingResponse"/>
1356     </schema>
1357   </types>
1358   <message name="PingRequest">
1359     <part name="ping" element="tns:Ping"/>
1360   </message>
1361   <message name="PingResponse">
1362     <part name="pingResponse" element="tns:PingResponse"/>
1363   </message>
1364   <portType name="PingPort">
1365     <operation name="Ping">
1366       <input message="tns:PingRequest"/>
1367       <output message="tns:PingResponse"/>
1368     </operation>
1369   </portType>
1370   <binding name="PingBinding" type="tns:PingPort">
1371     <soap:binding style="document"
1372 transport="http://schemas.xmlsoap.org/soap/http"/>
1373     <operation name="Ping">
1374       <soap:operation/>
1375       <input>
1376         <soap:body use="literal"/>
1377       </input>
1378       <output>
1379         <soap:body use="literal"/>
1380       </output>
1381     </operation>
1382   </binding>
1383   <service name="PingService">
1384     <port name="PingPort" binding="tns:PingBinding">
1385       <soap:address
1386 location="http://localhost:8080/pingejb/Ping"/>
1387     </port>
1388   </service>

```

1389

```
</definitions>
```

1390

1391

Appendix B. Revision History

1392

Rev	Date	By Whom	What
wss-01	2003-07-28	Hal Lockhart	Initial version
wss-02	2003-08-25	Hal Lockhart	Timestamp is created first – Appears as last element under Security Made c14n method a parameter to the STR Dereference Transform in scenario 7 Scenario 5 is altered to have a single ping element as required by the WS-I BP, a ticket element is added to Ping to provide a target for the inner signature
wss-03	2003-08-26	Hal Lockhart	Correct syntax of c14n parameter to STR Dereference Transform Change scenario 7 to sign the STR referring to the signature token rather than the encryption token Added ticket element to Ping schema in WSDL file

1393

1394

Appendix C. Notices

1395 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1396 that might be claimed to pertain to the implementation or use of the technology described in this
1397 document or the extent to which any license under such rights might or might not be available;
1398 neither does it represent that it has made any effort to identify any such rights. Information on
1399 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
1400 website. Copies of claims of rights made available for publication and any assurances of licenses
1401 to be made available, or the result of an attempt made to obtain a general license or permission
1402 for the use of such proprietary rights by implementors or users of this specification, can be
1403 obtained from the OASIS Executive Director.

1404 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1405 applications, or other proprietary rights which may cover technology that may be required to
1406 implement this specification. Please address the information to the OASIS Executive Director.

1407 **Copyright © OASIS Open 2002. All Rights Reserved.**

1408 This document and translations of it may be copied and furnished to others, and derivative works
1409 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1410 published and distributed, in whole or in part, without restriction of any kind, provided that the
1411 above copyright notice and this paragraph are included on all such copies and derivative works.
1412 However, this document itself does not be modified in any way, such as by removing the
1413 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1414 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1415 Property Rights document must be followed, or as required to translate it into languages other
1416 than English.

1417 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1418 successors or assigns.

1419 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1420 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1421 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1422 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1423 PARTICULAR PURPOSE.