



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26

---

# Web Services Security: Interop 2 Scenarios

## Working Draft 04, 26 Aug 2003

**Document identifier:**

wss-interop2-draft-04.doc

**Location:**

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

**Editor:**

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

**Contributors:**

Chris Kaler, Microsoft <ckaler@microsoft.com>  
Hal Lockhart, BEA Systems <hlockhar@bea.com>  
Peter Dapkus, BEA Systems <pdapkus@bea.com>  
Anthony Nadalin, IBM <drsecure@us.ibm.com>  
Frederick Hirsch, nokia <Frederick.Hirsch@nokia.com>

**Abstract:**

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

**Status:**

Committee members should send comments on this specification to the [wss@lists.oasis-open.org](mailto:wss@lists.oasis-open.org) list. Others should subscribe to and send comments to the [wss-comment@lists.oasis-open.org](mailto:wss-comment@lists.oasis-open.org) list. To subscribe, send an email message to [wss-comment-request@lists.oasis-open.org](mailto:wss-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

## Table of Contents

27	Introduction .....	5
28	1.1 Terminology.....	5
29	2 Test Application .....	6
30	3 Scenario #4 Session Key.....	7
31	3.1 Agreements .....	7
32	3.1.1 SESSION-KEY-VALUE .....	7
33	3.1.2 CERT-VALUE .....	7
34	3.1.3 Signature Trust Root.....	7
35	3.2 Parameters.....	7
36	3.3 General Message Flow .....	7
37	3.4 First Message - Request.....	8
38	3.4.1 Message Elements and Attributes .....	8
39	3.4.2 Message Creation.....	8
40	3.4.3 Message Processing.....	9
41	3.4.4 Example (Non-normative) .....	10
42	3.5 Second Message - Response .....	11
43	3.5.1 Message Elements and Attributes .....	11
44	3.5.2 Message Creation.....	12
45	3.5.3 Message Processing.....	13
46	3.5.4 Example (Non-normative) .....	13
47	3.6 Other processing.....	14
48	3.6.1 Requester .....	14
49	3.6.2 Responder .....	14
50	3.7 Expected Security Properties.....	14
51	4 Scenario #5 – Overlapping Signatures .....	15
52	4.1 Agreements .....	15
53	4.1.1 CERT-VALUE .....	15
54	4.1.2 Signature Trust Root.....	15
55	4.2 Parameters.....	15
56	4.3 General Message Flow .....	15
57	4.4 First Message - Request.....	15
58	4.4.1 Message Elements and Attributes .....	15
59	4.4.2 Message Creation.....	16
60	4.4.3 Message Processing.....	17
61	4.4.4 Example (Non-normative) .....	18
62	4.5 Second Message - Response .....	19
63	4.5.1 Message Elements and Attributes .....	19
64	4.5.2 Message Creation.....	19
65	4.5.3 Message Processing.....	19
66	4.5.4 Example (Non-normative) .....	19
67	4.6 Other processing .....	20
68	4.6.1 Requester .....	20

69	4.6.2 Responder .....	20
70	4.7 Expected Security Properties.....	20
71	5 Scenario #6 – Encrypt and Sign .....	21
72	5.1 Agreements.....	21
73	5.1.1 CERT-VALUE .....	21
74	5.1.2 Signature Trust Root.....	21
75	5.2 Parameters.....	21
76	5.3 General Message Flow .....	21
77	5.4 First Message - Request.....	21
78	5.4.1 Message Elements and Attributes .....	21
79	5.4.2 Message Creation.....	22
80	5.4.3 Message Processing.....	24
81	5.4.4 Example (Non-normative).....	24
82	5.5 Second Message - Response.....	25
83	5.5.1 Message Elements and Attributes .....	25
84	5.5.2 Message Creation.....	26
85	5.5.3 Message Processing.....	27
86	5.5.4 Example (Non-normative).....	28
87	5.6 Other processing .....	29
88	5.6.1 Requester .....	29
89	5.6.2 Responder .....	29
90	5.7 Expected Security Properties.....	29
91	6 Scenario #7 – Signed Token.....	30
92	6.1 Agreements.....	30
93	6.1.1 CERT-VALUE .....	30
94	6.1.2 Signature Trust Root.....	30
95	6.2 Parameters.....	30
96	6.3 General Message Flow .....	30
97	6.4 First Message - Request.....	31
98	6.4.1 Message Elements and Attributes .....	31
99	6.4.2 Message Creation.....	32
100	6.4.3 Message Processing.....	33
101	6.4.4 Example (Non-normative).....	33
102	6.5 Second Message - Response .....	35
103	6.5.1 Message Elements and Attributes .....	35
104	6.5.2 Message Creation.....	36
105	6.5.3 Message Processing.....	37
106	6.5.4 Example (Non-normative).....	37
107	6.6 Other processing .....	39
108	6.6.1 Requester .....	39
109	6.6.2 Responder .....	39
110	6.7 Expected Security Properties.....	39
111	7 References.....	40
112	7.1 Normative .....	40

113 Appendix A. Ping Application WSDL File ..... 41  
114 Appendix B. Revision History ..... 43  
115 Appendix C. Notices ..... 44  
116

---

117 **Introduction**

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

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

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

132 **1.1 Terminology**

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

---

135 **2 Test Application**

136 All three scenarios use the same, simple application.

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

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

---

## 139 **3 Scenario #4 Session Key**

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

### 145 **3.1 Agreements**

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

#### 148 **3.1.1 SESSION-KEY-VALUE**

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

#### 151 **3.1.2 CERT-VALUE**

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

#### 156 **3.1.3 Signature Trust Root**

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

### 160 **3.2 Parameters**

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

163 No parameters are required.

### 164 **3.3 General Message Flow**

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

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

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

175 **3.4 First Message - Request**

176 **3.4.1 Message Elements and Attributes**

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

181

<b>Name</b>	<b>Mandatory?</b>
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

182

183 **3.4.2 Message Creation**

184 **3.4.2.1 Security**

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

186 **3.4.2.2 ReferenceList**

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



### 189 **3.4.2.3 BinarySecurityToken**

190 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be  
191 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate  
192 suitable for verifying the signature. The certificate SHOULD NOT have a KeyUsage extension. If  
193 it does contain a KeyUsage extension, it SHOULD include the value of digitalSignature. The  
194 Requester must have access to the private key corresponding to 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 the random key  
222 identified by SESSION-KEY-VALUE, 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 ticket element is signed. The certificate used to verify this  
460 signature is provided out-of-band. Next the entire body is signed. The certificate for this signature  
461 is included in the message. The Responder verifies both signatures. If no errors are detected it  
462 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.

<b>Name</b>	<b>Mandatory?</b>
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. The certificate SHOULD NOT have a KeyUsage extension. If  
493 it does contain a KeyUsage extension, it SHOULD include the values of digitalSignature. The  
494 Requester must have access to the private key corresponding to the public key in the certificate.

#### 495 **4.4.2.4 Signature**

496 This signature is over the entire SOAP body.

#### 497 **4.4.2.4.1 SignedInfo**

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

#### 502 **4.4.2.4.2 SignatureValue**

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

#### 505 **4.4.2.4.3 KeyInfo**

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

#### 509 **4.4.2.5 Timestamp**

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

#### 512 **4.4.2.6 Body**

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

### 515 **4.4.3 Message Processing**

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

### 519 4.4.3.1 Security

### 520 4.4.3.2 Signature

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

### 524 4.4.3.3 BinarySecurityToken

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

### 528 4.4.3.4 Signature

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

### 531 4.4.3.5 Timestamp

532 The Timestamp element MUST be ignored.

### 533 4.4.3.6 Body

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

## 536 4.4.4 Example (Non-normative)

537 Here is an example request.

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

571  
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

```
<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>
```

602

## 603 4.5 Second Message - Response

### 604 4.5.1 Message Elements and Attributes

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

Name	Mandatory?
Body	Mandatory

609

### 610 4.5.2 Message Creation

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

613

### 614 4.5.3 Message Processing

615 The body is passed to the application without modification.

### 616 4.5.4 Example (Non-normative)

617 Here is an example response.

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

## 628 **4.6 Other processing**

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

### 630 **4.6.1 Requester**

631 No additional processing is required.

### 632 **4.6.2 Responder**

633 No additional processing is required.

## 634 **4.7 Expected Security Properties**

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

---

## 637 **5 Scenario #6 – Encrypt and Sign**

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

### 642 **5.1 Agreements**

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

#### 645 **5.1.1 CERT-VALUE**

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

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

#### 652 **5.1.2 Signature Trust Root**

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

### 656 **5.2 Parameters**

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

659 No parameters are required.

### 660 **5.3 General Message Flow**

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

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

### 670 **5.4 First Message - Request**

#### 671 **5.4.1 Message Elements and Attributes**

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

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

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

677

## 678 **5.4.2 Message Creation**

### 679 **5.4.2.1 Security**

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

### 681 **5.4.2.2 BinarySecurityToken**

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

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

### 688 **5.4.2.3 Signature**

689 The signature is over the entire SOAP body.

#### 690 **5.4.2.3.1 SignedInfo**

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

#### 695 **5.4.2.3.2 SignatureValue**

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

#### 698 **5.4.2.3.3 KeyInfo**

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

#### 702 **5.4.2.4 EncryptedKey**

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

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

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

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

#### 711 **5.4.2.5 Timestamp**

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

#### 714 **5.4.2.6 Body**

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

#### 716 **5.4.2.7 EncryptedData**

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

719 The Type MUST have the value of #Content.

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

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

### 724 **5.4.3 Message Processing**

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

#### 728 **5.4.3.1 Security**

#### 729 **5.4.3.2 BinarySecurityToken**

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

#### 733 **5.4.3.3 Signature**

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

#### 736 **5.4.3.4 EncryptedKey**

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

#### 739 **5.4.3.5 Timestamp**

740 The Timestamp element MUST be ignored.

#### 741 **5.4.3.6 Body**

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

#### 744 **5.4.3.7 EncryptedData**

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

### 747 **5.4.4 Example (Non-normative)**

748 Here is an example request.

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



```

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

```

813

## 814 5.5 Second Message - Response

### 815 5.5.1 Message Elements and Attributes

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

819

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

820

## 821 **5.5.2 Message Creation**

### 822 **5.5.2.1 Security**

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

### 825 **5.5.2.2 Signature**

826 The signature is over the entire SOAP body.

#### 827 **5.5.2.2.1 SignedInfo**

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

#### 832 **5.5.2.2.2 SignatureValue**

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

### 835 **5.5.2.2.3 KeyInfo**

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

### 839 **5.5.2.3 BinarySecurityToken**

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

### 843 **5.5.2.4 EncryptedKey**

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

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

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

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

### 852 **5.5.2.5 Timestamp**

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

### 855 **5.5.2.6 Body**

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

### 857 **5.5.2.7 EncryptedData**

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

860 The Type MUST have the value of #Content.

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

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

## 865 **5.5.3 Message Processing**

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

### 869 **5.5.3.1 Security**

### 870 **5.5.3.2 Timestamp**

871 The Timestamp element MUST be ignored.

### 872 5.5.3.3 Body

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

### 874 5.5.3.4 EncryptedData

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

### 877 5.5.3.5 Signature

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

### 880 5.5.3.6 BinarySecurityToken

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

### 883 5.5.3.7 EncryptedKey

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

## 886 5.5.4 Example (Non-normative)

887 Here is an example response.

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

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

952

## 953 **5.6 Other processing**

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

### 955 **5.6.1 Requester**

956 No additional processing is required.

### 957 **5.6.2 Responder**

958 No additional processing is required.

## 959 **5.7 Expected Security Properties**

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

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

---

## 967 **6 Scenario #7 – Signed Token**

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

### 973 **6.1 Agreements**

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

#### 976 **6.1.1 CERT-VALUE**

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

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

#### 983 **6.1.2 Signature Trust Root**

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

### 987 **6.2 Parameters**

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

990 No parameters are required.

### 991 **6.3 General Message Flow**

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

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

1002 **6.4 First Message - Request**

1003 **6.4.1 Message Elements and Attributes**

1004 Items not listed in the following table MAY be present, but MUST NOT be marked with the  
1005 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.  
1006 Items marked optional MAY be generated and MUST be processed if present. Items MUST  
1007 appear in the order specified, except as noted.

1008

<b>Name</b>	<b>Mandatory?</b>
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

1009

## 1010 **6.4.2 Message Creation**

### 1011 **6.4.2.1 Security**

1012 The Security element MUST contain the mustUnderstand="1" attribute.

### 1013 **6.4.2.2 EncryptedKey**

1014 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1015 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
1016 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
1017 MUST have the value of CERT-VALUE.

1018 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public  
1019 Key specified in the specified X.509 certificate, using the specified algorithm.

1020 The ReferenceList MUST contain a DataReference which has the value of a relative URI that  
1021 refers to the encrypted body of the message.

### 1022 **6.4.2.3 BinarySecurityToken**

1023 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be  
1024 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate  
1025 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT  
1026 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the  
1027 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have  
1028 access to the private key corresponding to the public key in the certificate.

### 1029 **6.4.2.4 Signature**

1030 The signature is over the entire SOAP body.

#### 1031 **6.4.2.4.1 SignedInfo**

1032 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
1033 be RSA-SHA1.

1034 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference  
1035 contained in the Signature. The STR Dereference Transform with a parameter of the Exclusive  
1036 Canonicalization Transform MUST be specified. The DigestMethod MUST be SHA1.

1037 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The  
1038 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be  
1039 SHA1.

#### 1040 **6.4.2.4.2 SignatureValue**

1041 The SignatureValue MUST be calculated as specified by the specification, using the private key  
1042 corresponding to the public key specified in the certificate in the BinarySecurityToken.

#### 1043 **6.4.2.4.3 KeyInfo**

1044 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which  
1045 indicates the BinarySecurityToken containing the certificate which will be used for signature  
1046 verification.

### 1047 **6.4.2.5 Timestamp**

1048 The Created element within the Timestamp SHOULD contain the current local time at the sender  
1049 expressed in the UTC time zone.



1050 **6.4.2.6 Body**

1051 The body element MUST be first signed and then its contents encrypted.

1052 **6.4.2.7 EncryptedData**

1053 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
1054 EncryptedKey.

1055 The Type MUST have the value of #Content.

1056 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
1057 – CBC.

1058 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,  
1059 using the specified algorithm.

1060 **6.4.3 Message Processing**

1061 This section describes the processing performed by the Responder. If an error is detected, the  
1062 Responder MUST cease processing the message and issue a Fault with a value of  
1063 FailedAuthentication.

1064 **6.4.3.1 Security**

1065 **6.4.3.2 EncryptedKey**

1066 The random key contained in the CipherData MUST be decrypted using the private key  
1067 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

1068 **6.4.3.3 Timestamp**

1069 The Timestamp element MUST be ignored.

1070 **6.4.3.4 Body**

1071 The contents of the body MUST first be decrypted and then the signature verified. If no errors are  
1072 detected, the body MUST be passed to the application.

1073 **6.4.3.5 EncryptedData**

1074 The message body contents contained in the EncryptedData, referenced by the ReferenceList  
1075 MUST be decrypted using the random key, using the specified algorithm.

1076 **6.4.3.6 BinarySecurityToken**

1077 The certificate in the token MUST be validated. The Subject of the certificate MUST be an  
1078 authorized entity. The public key in the certificate MUST be retained for verification of the  
1079 signature.

1080 **6.4.3.7 Signature**

1081 The body after decryption, MUST be verified against the signature using the specified algorithms  
1082 and transforms and the retained public key.

1083 **6.4.4 Example (Non-normative)**

1084 Here is an example request.

1085 `<?xml version="1.0" encoding="utf-8" ?>`

```

1086 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1087 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1088 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1089 <soap:Header>
1090 <wsse:Security soap:mustUnderstand="1"
1091 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1092 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1093 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1094 />
1095 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1096 <wsse:SecurityTokenReference>
1097 <wsse:KeyIdentifier
1098 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
1099 </wsse:SecurityTokenReference>
1100 </KeyInfo>
1101 <xenc:CipherData>
1102 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
1103 </xenc:CipherData>
1104 <xenc:ReferenceList>
1105 <xenc:DataReference URI="#enc" />
1106 </xenc:ReferenceList>
1107 </xenc:EncryptedKey>
1108 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1109 EncodingType="wsse:Base64Binary"
1110 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1111 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1112 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1113 <SignedInfo>
1114 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1115 />
1116 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1117 <Reference URI="#Token">
1118 <Transforms>
1119 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
1120 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1121 c14n#"/>
1122 </Transform>
1123 </Transforms>
1124 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1125 <DigestValue>pHrr...xK=</DigestValue>
1126 </Reference>
1127 <Reference URI="#body">
1128 <Transforms>
1129 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1130 </Transforms>
1131 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1132 <DigestValue>QTV...dw=</DigestValue>
1133 </Reference>
1134 </SignedInfo>
1135 <SignatureValue>H+x0...gUw=</SignatureValue>
1136 <KeyInfo>
1137 <wsse:SecurityTokenReference wsu:Id="Token">
1138 <wsse:Reference URI="#myCert" />
1139 </wsse:SecurityTokenReference>
1140 </KeyInfo>
1141 </Signature>
1142 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1143 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1144 </wsu:Timestamp>
1145 </wsse:Security>
1146 </soap:Header>
1147 <soap:Body wsu:Id="body"
1148 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1149 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1150 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1151 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
1152 cbc" />
1153 <xenc:CipherData>
1154 <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
1155 </xenc:CipherData>
1156 </xenc:EncryptedData>

```

1157  
1158  
1159

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

1160

## 6.5 Second Message - Response

1161

### 6.5.1 Message Elements and Attributes

1162  
1163  
1164  
1165

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

1166

## 1167 **6.5.2 Message Creation**

### 1168 **6.5.2.1 Security**

1169 The Security element MUST contain the mustUnderstand="1" attribute. Any other header  
1170 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

### 1171 **6.5.2.2 BinarySecurityToken**

1172 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be  
1173 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent  
1174 in the request.

### 1175 **6.5.2.3 EncryptedKey**

1176 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1177 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which  
1178 indicates the BinarySecurityToken containing the certificate which will be used for signature  
1179 verification.

1180 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public  
1181 Key specified in the specified X.509 certificate, using the specified algorithm.

1182 The ReferenceList MUST contain a DataReference which has the value of a relative URI that  
1183 refers to the encrypted body of the message.

### 1184 **6.5.2.4 Signature**

1185 The signature is over the entire SOAP body.

#### 1186 **6.5.2.4.1 SignedInfo**

1187 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
1188 be RSA-SHA1.

1189 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference  
1190 contained in the Signature. The STR Dereference Transform with a parameter of the Exclusive  
1191 Canonicalization Transform MUST be specified. The DigestMethod MUST be SHA1.

1192 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The  
1193 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be  
1194 SHA1.

#### 1195 **6.5.2.4.2 SignatureValue**

1196 The SignatureValue MUST be calculated as specified by the specification, using the private key  
1197 corresponding to the public key specified in the certificate in the BinarySecurityToken.

#### 1198 **6.5.2.4.3 KeyInfo**

1199 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
1200 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
1201 MUST have the value of CERT-VALUE.

### 1202 **6.5.2.5 Timestamp**

1203 The Created element within the Timestamp SHOULD contain the current local time at the sender  
1204 expressed in the UTC time zone.

1205 **6.5.2.6 Body**

1206 The body element MUST be first signed and then its contents encrypted.

1207 **6.5.2.7 EncryptedData**

1208 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
1209 EncryptedKey.

1210 The Type MUST have the value of #Content.

1211 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
1212 – CBC.

1213 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,  
1214 using the specified algorithm.

1215 **6.5.3 Message Processing**

1216 This section describes the processing performed by the Responder. If an error is detected, the  
1217 Responder MUST cease processing the message and report the fault locally with a value of  
1218 FailedAuthentication.

1219 **6.5.3.1 Security**

1220 **6.5.3.2 BinarySecurityToken**

1221 The certificate in the token MUST be validated. The Subject of the certificate MUST be an  
1222 authorized entity. The certificate is used to identify the private key to be used for decryption.

1223 **6.5.3.3 EncryptedKey**

1224 The random key contained in the CipherData MUST be decrypted using the private key  
1225 corresponding to the certificate specified by the Reference, using the specified algorithm.

1226 **6.5.3.4 Timestamp**

1227 The Timestamp element MUST be ignored.

1228 **6.5.3.5 Body**

1229 The contents of the body MUST first be decrypted and then the signature verified.

1230 **6.5.3.6 EncryptedData**

1231 The message body contents contained in the EncryptedData, referenced by the ReferenceList  
1232 MUST be decrypted using the random key, using the specified algorithm.

1233 **6.5.3.7 Signature**

1234 The body after decryption, MUST be verified against the signature using the specified algorithms  
1235 and transforms and the indicated public key.

1236 **6.5.4 Example (Non-normative)**

1237 Here is an example response.

```
1238 <?xml version="1.0" encoding="utf-8" ?>  
1239 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
1240 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1241 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```

1242 <soap:Header>
1243 <wsse:Security soap:mustUnderstand="1"
1244 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1245 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1246 EncodingType="wsse:Base64Binary"
1247 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1248 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1249 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1250 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1251 />
1252 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1253 <wsse:SecurityTokenReference>
1254 <wsse:Reference URI="#myCert" />
1255 </wsse:SecurityTokenReference>
1256 </KeyInfo>
1257 <xenc:CipherData>
1258 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
1259 </xenc:CipherData>
1260 <xenc:ReferenceList>
1261 <xenc:DataReference URI="#enc" />
1262 </xenc:ReferenceList>
1263 </xenc:EncryptedKey>
1264 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1265 <SignedInfo>
1266 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1267 />
1268 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1269 <Reference URI="#Token">
1270 <Transforms>
1271 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
1272 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
1273 c14n#" />
1274 </Transform>
1275 </Transforms>
1276 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1277 <DigestValue>B4j...Xv</DigestValue>
1278 </Reference>
1279 <Reference URI="#body">
1280 <Transforms>
1281 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1282 </Transforms>
1283 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1284 <DigestValue>KxW...5B</DigestValue>
1285 </Reference>
1286 </SignedInfo>
1287 <SignatureValue>8Hkd...a17</SignatureValue>
1288 <KeyInfo>
1289 <wsse:SecurityTokenReference wsu:Id="Token">
1290 <wsse:KeyIdentifier
1291 ValueType="wsse:X509v3">B39R...mY</wsse:KeyIdentifier>
1292 </wsse:SecurityTokenReference>
1293 </KeyInfo>
1294 </Signature>
1295 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1296 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1297 </wsu:Timestamp>
1298 </wsse:Security>
1299 </soap:Header>
1300 <soap:Body wsu:Id="body"
1301 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1302 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1303 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1304 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
1305 cbc" />
1306 <xenc:CipherData>
1307 <xenc:CipherValue>d2s...GQ</xenc:CipherValue>
1308 </xenc:CipherData>
1309 </xenc:EncryptedData>
1310 </soap:Body>
1311 </soap:Envelope>

```

1312

## 1313 **6.6 Other processing**

1314 This section describes processing that occurs outside of generating or processing a message.

### 1315 **6.6.1 Requester**

1316 No additional processing is required.

### 1317 **6.6.2 Responder**

1318 No additional processing is required.

## 1319 **6.7 Expected Security Properties**

1320 Use of the service is restricted to authorized parties that sign the Body of the request. The Body  
1321 of the request is protected against modification and interception. The response is Authenticated  
1322 and protected against modification and interception. The signature over the signature token binds  
1323 it to the message, preventing a repudiation attack by certificate substitution.

1324 The Responder must not draw any inferences about what party encrypted the message, it  
1325 particular it should not be assumed it was the same party who signed it.

---

1326 **7 References**

1327 **7.1 Normative**

1328 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
1329 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.



## Appendix A. Ping Application WSDL File

```

1331 <definitions xmlns:tns="http://xmlsoap.org/Ping"
1332 xmlns="http://schemas.xmlsoap.org/wsdl/"
1333 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1334 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
1335 targetNamespace="http://xmlsoap.org/Ping" name="Ping">
1336   <types>
1337     <schema targetNamespace="http://xmlsoap.org/Ping"
1338 xmlns="http://www.w3.org/2001/XMLSchema">
1339       <complexType name="ticketType">
1340         <sequence>
1341           <element name="ticket" type="xsd:string"/>
1342         </sequence>
1343         <attribute name="Id" type="wsu:Id"/>
1344       </complexType>
1345       <complexType name="ping">
1346         <sequence>
1347           <element name="ticket" type="ticketType"
1348 minOccurs="0"/>
1349           <element name="text" type="xsd:string"
1350 nillable="true"/>
1351         </sequence>
1352       </complexType>
1353       <complexType name="pingResponse">
1354         <sequence>
1355           <element name="text" type="xsd:string"
1356 nillable="true"/>
1357         </sequence>
1358       </complexType>
1359       <element name="Ping" type="tns:ping"/>
1360       <element name="PingResponse" type="tns:pingResponse"/>
1361     </schema>
1362   </types>
1363   <message name="PingRequest">
1364     <part name="ping" element="tns:Ping"/>
1365   </message>
1366   <message name="PingResponse">
1367     <part name="pingResponse" element="tns:PingResponse"/>
1368   </message>
1369   <portType name="PingPort">
1370     <operation name="Ping">
1371       <input message="tns:PingRequest"/>
1372       <output message="tns:PingResponse"/>
1373     </operation>
1374   </portType>
1375   <binding name="PingBinding" type="tns:PingPort">
1376     <soap:binding style="document"
1377 transport="http://schemas.xmlsoap.org/soap/http"/>
1378     <operation name="Ping">
1379       <soap:operation/>
1380       <input>
1381         <soap:body use="literal"/>
1382       </input>
1383       <output>
1384         <soap:body use="literal"/>
1385       </output>
1386     </operation>
1387   </binding>

```

1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395

```
<service name="PingService">  
  <port name="PingPort" binding="tns:PingBinding">  
    <soap:address  
location="http://localhost:8080/pingejb/Ping"/>  
  </port>  
</service>  
</definitions>
```

1396

## Appendix B. Revision History

1397

Rev	Date	By Whom	What
wss-01	2003-07-28	Hal Lockhart	Initial version
wss-02	2003-08-25	Hal Lockhart	<p>Timestamp is created first – Appears as last element under Security</p> <p>Made c14n method a parameter to the STR Dereference Transform in scenario 7</p> <p>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</p>
wss-03	2003-08-26	Hal Lockhart	<p>Correct syntax of c14n parameter to STR Dereference Transform</p> <p>Change scenario 7 to sign the STR referring to the signature token rather than the encryption token</p> <p>Added ticket element to Ping schema in WSDL file</p>
wss-04	2003-08-26	Hal Lockhart	<p>Fixed Ping Schema</p> <p>Fixed various typos</p>

1398

---

## Appendix C. Notices

1400 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
1401 that might be claimed to pertain to the implementation or use of the technology described in this  
1402 document or the extent to which any license under such rights might or might not be available;  
1403 neither does it represent that it has made any effort to identify any such rights. Information on  
1404 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
1405 website. Copies of claims of rights made available for publication and any assurances of licenses  
1406 to be made available, or the result of an attempt made to obtain a general license or permission  
1407 for the use of such proprietary rights by implementors or users of this specification, can be  
1408 obtained from the OASIS Executive Director.

1409 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
1410 applications, or other proprietary rights which may cover technology that may be required to  
1411 implement this specification. Please address the information to the OASIS Executive Director.

1412 **Copyright © OASIS Open 2002. All Rights Reserved.**

1413 This document and translations of it may be copied and furnished to others, and derivative works  
1414 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
1415 published and distributed, in whole or in part, without restriction of any kind, provided that the  
1416 above copyright notice and this paragraph are included on all such copies and derivative works.  
1417 However, this document itself does not be modified in any way, such as by removing the  
1418 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
1419 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
1420 Property Rights document must be followed, or as required to translate it into languages other  
1421 than English.

1422 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
1423 successors or assigns.

1424 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1425 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
1426 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
1427 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
1428 PARTICULAR PURPOSE.