



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

---

# Web Services Security: Interop 2 Scenarios Working Draft 01, 28 Jul 2003

**Document identifier:**

wss-interop2-draft-01.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>

**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

24	Introduction .....	5
25	1.1 Terminology.....	5
26	2 Test Application .....	6
27	3 Scenario #4 Session Key.....	7
28	3.1 Agreements .....	7
29	3.1.1 SESSION-KEY-VALUE .....	7
30	3.1.2 CERT-VALUE .....	7
31	3.1.3 Signature Trust Root.....	7
32	3.2 Parameters.....	7
33	3.3 General Message Flow .....	7
34	3.4 First Message - Request.....	8
35	3.4.1 Message Elements and Attributes .....	8
36	3.4.2 Message Creation.....	8
37	3.4.3 Message Processing.....	9
38	3.4.4 Example (Non-normative) .....	10
39	3.5 Second Message - Response .....	11
40	3.5.1 Message Elements and Attributes .....	11
41	3.5.2 Message Creation.....	12
42	3.5.3 Message Processing.....	13
43	3.5.4 Example (Non-normative) .....	13
44	3.6 Other processing.....	14
45	3.6.1 Requester .....	14
46	3.6.2 Responder .....	14
47	3.7 Expected Security Properties.....	14
48	4 Scenario #5 – Overlapping Signatures .....	15
49	4.1 Agreements .....	15
50	4.1.1 CERT-VALUE .....	15
51	4.1.2 Signature Trust Root.....	15
52	4.2 Parameters.....	15
53	4.3 General Message Flow .....	15
54	4.4 First Message - Request.....	15
55	4.4.1 Message Elements and Attributes .....	15
56	4.4.2 Message Creation.....	16
57	4.4.3 Message Processing.....	18
58	4.4.4 Example (Non-normative) .....	18
59	4.5 Second Message - Response .....	19
60	4.5.1 Message Elements and Attributes .....	19
61	4.5.2 Message Creation.....	19
62	4.5.3 Message Processing.....	20
63	4.5.4 Example (Non-normative) .....	20
64	4.6 Other processing .....	20
65	4.6.1 Requester .....	20

66	4.6.2 Responder .....	20
67	4.7 Expected Security Properties.....	20
68	5 Scenario #6 – Encrypt and Sign .....	21
69	5.1 Agreements.....	21
70	5.1.1 CERT-VALUE .....	21
71	5.1.2 Signature Trust Root.....	21
72	5.2 Parameters.....	21
73	5.3 General Message Flow .....	21
74	5.4 First Message - Request.....	21
75	5.4.1 Message Elements and Attributes .....	21
76	5.4.2 Message Creation.....	22
77	5.4.3 Message Processing.....	24
78	5.4.4 Example (Non-normative).....	24
79	5.5 Second Message - Response .....	25
80	5.5.1 Message Elements and Attributes .....	25
81	5.5.2 Message Creation.....	26
82	5.5.3 Message Processing.....	27
83	5.5.4 Example (Non-normative).....	28
84	5.6 Other processing .....	29
85	5.6.1 Requester .....	29
86	5.6.2 Responder .....	29
87	5.7 Expected Security Properties.....	29
88	6 Scenario #7 – Signed Token.....	30
89	6.1 Agreements.....	30
90	6.1.1 CERT-VALUE .....	30
91	6.1.2 Signature Trust Root.....	30
92	6.2 Parameters.....	30
93	6.3 General Message Flow .....	30
94	6.4 First Message - Request.....	31
95	6.4.1 Message Elements and Attributes .....	31
96	6.4.2 Message Creation.....	32
97	6.4.3 Message Processing.....	33
98	6.4.4 Example (Non-normative).....	33
99	6.5 Second Message - Response .....	35
100	6.5.1 Message Elements and Attributes .....	35
101	6.5.2 Message Creation.....	36
102	6.5.3 Message Processing.....	37
103	6.5.4 Example (Non-normative).....	37
104	6.6 Other processing .....	39
105	6.6.1 Requester .....	39
106	6.6.2 Responder .....	39
107	6.7 Expected Security Properties.....	39
108	7 References.....	40
109	7.1 Normative .....	40

110	Appendix A. Ping Application WSDL File .....	41
111	Appendix B. Revision History .....	42
112	Appendix C. Notices .....	43
113		

---

## 114 Introduction

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

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

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

### 129 1.1 Terminology

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

---

132 **2 Test Application**

133 All three scenarios use the same, simple application.

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

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

---

## 136 **3 Scenario #4 Session Key**

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

### 142 **3.1 Agreements**

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

#### 145 **3.1.1 SESSION-KEY-VALUE**

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

#### 148 **3.1.2 CERT-VALUE**

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

#### 153 **3.1.3 Signature Trust Root**

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

### 157 **3.2 Parameters**

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

160 No parameters are required.

### 161 **3.3 General Message Flow**

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

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

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

172 **3.4 First Message - Request**

173 **3.4.1 Message Elements and Attributes**

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

178

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

179

180 **3.4.2 Message Creation**

181 **3.4.2.1 Security**

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

183 **3.4.2.2 Timestamp**

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



### 186 **3.4.2.3 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 **3.4.2.4 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 and encrypting the response. The certificate SHOULD NOT  
193 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the  
194 value of digitalSignature. The Requester must have access to the private key corresponding to  
195 the public key in the certificate.

### 196 **3.4.2.5 Signature**

197 The signature is over the entire SOAP body.

#### 198 **3.4.2.5.1 SignedInfo**

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

#### 203 **3.4.2.5.2 SignatureValue**

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

#### 206 **3.4.2.5.3 KeyInfo**

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

### 210 **3.4.2.6 Body**

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

### 212 **3.4.2.7 EncryptedData**

213 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
214 EncryptedKey.

215 The Type MUST have the value of #Content.

216 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
217 – CBC.

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

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

## 221 **3.4.3 Message Processing**

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

### 225 3.4.3.1 Security

### 226 3.4.3.2 Timestamp

227 The Timestamp element MUST be ignored.

### 228 3.4.3.3 ReferenceList

229 The ReferenceList indicates the data to be decrypted.

### 230 3.4.3.4 Body

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

### 233 3.4.3.5 EncryptedData

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

### 237 3.4.3.6 BinarySecurityToken

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

### 241 3.4.3.7 Signature

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

## 244 3.4.4 Example (Non-normative)

245 Here is an example request.

```
246 <?xml version="1.0" encoding="utf-8" ?>
247 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
248 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
249 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
250 <soap:Header>
251 <wsse:Security soap:mustUnderstand="1"
252 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
253 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
254 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
255 </wsu:Timestamp>
256 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
257 <xenc:DataReference URI="#enc" />
258 </xenc:ReferenceList>
259 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
260 EncodingType="wsse:Base64Binary"
261 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
262 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
263 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
264 <SignedInfo>
265 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
266 />
267 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
268 <Reference URI="#body">
269 <Transforms>
270 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
271 </Transforms>
272 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
273 <DigestValue>QTV...dw</DigestValue>
```

274  
275  
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

```

    </Reference>
  </SignedInfo>
  <SignatureValue>H+x0...gUw=</SignatureValue>
  <KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:Reference URI="#myCert" />
    </wsse:SecurityTokenReference>
  </KeyInfo>
</Signature>
</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>

```

### 301 3.5 Second Message - Response

#### 302 3.5.1 Message Elements and Attributes

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

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

EncryptionMethod	Mandatory
KeyInfo	Mandatory
Cipherdata	Mandatory

307

## 308 **3.5.2 Message Creation**

### 309 **3.5.2.1 Security**

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

### 312 **3.5.2.2 Timestamp**

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

### 315 **3.5.2.3 ReferenceList**

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

### 318 **3.5.2.4 Signature**

319 The signature is over the entire SOAP body.

#### 320 **3.5.2.4.1 SignedInfo**

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

#### 325 **3.5.2.4.2 SignatureValue**

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

#### 328 **3.5.2.4.3 KeyInfo**

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

### 332 **3.5.2.5 Body**

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

### 334 **3.5.2.6 EncryptedData**

335 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
336 EncryptedKey.

337 The Type MUST have the value of #Content.

338 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
339 – CBC.

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

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

### 343 **3.5.3 Message Processing**

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

#### 347 **3.5.3.1 Timestamp**

348 The Timestamp element MUST be ignored.

#### 349 **3.5.3.2 Security**

##### 350 **3.5.3.3 ReferenceList**

351 The ReferenceList indicates the data to be decrypted

##### 352 **3.5.3.4 Body**

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

##### 354 **3.5.3.5 EncryptedData**

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

##### 358 **3.5.3.6 Signature**

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

### 361 **3.5.4 Example (Non-normative)**

362 Here is an example response.

```
363 <?xml version="1.0" encoding="utf-8" ?>
364 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
365 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
366 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
367 <soap:Header>
368 <wsse:Security soap:mustUnderstand="1"
369 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
370 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
371 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
372 </wsu:Timestamp>
373 <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
374 <xenc:DataReference URI="#enc" />
375 </xenc:ReferenceList>
376 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
377 <SignedInfo>
378 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
379 />
380 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
381 <Reference URI="#body">
382 <Transforms>
```

```
383     <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
384   </Transforms>
385   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
386   <DigestValue>KxW...5B=</DigestValue>
387 </Reference>
388 </SignedInfo>
389 <SignatureValue>8Hkd...a17=</SignatureValue>
390 <KeyInfo>
391   <wsse:SecurityTokenReference>
392     <wsse:KeyIdentifier
393 ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
394   </wsse:SecurityTokenReference>
395 </KeyInfo>
396 </Signature>
397 </wsse:Security>
398 </soap:Header>
399 <soap:Body wsu:Id="body"
400 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
401   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
402   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
403     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
404 cbc" />
405     <xenc:KeyInfo>
406       <xenc:KeyName>SessionKey</KeyName>
407     </xenc:KeyInfo>
408     <xenc:CipherData>
409       <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
410     </xenc:CipherData>
411   </xenc:EncryptedData>
412 </soap:Body>
413 </soap:Envelope>
```

414

## 415 **3.6 Other processing**

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

### 417 **3.6.1 Requester**

418 No additional processing is required.

### 419 **3.6.2 Responder**

420 No additional processing is required.

## 421 **3.7 Expected Security Properties**

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

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

---

## 429 **4 Scenario #5 – Overlapping Signatures**

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

### 434 **4.1 Agreements**

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

#### 437 **4.1.1 CERT-VALUE**

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

442 The Responder MUST have access to the Private key corresponding to the Public key in the  
443 certificate.

#### 444 **4.1.2 Signature Trust Root**

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

### 448 **4.2 Parameters**

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

451 No parameters are required.

### 452 **4.3 General Message Flow**

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

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

### 461 **4.4 First Message - Request**

#### 462 **4.4.1 Message Elements and Attributes**

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

467

<b>Name</b>	<b>Mandatory?</b>
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	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
Body	Mandatory

468

## 469 **4.4.2 Message Creation**

### 470 **4.4.2.1 Security**

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

### 472 **4.4.2.2 Timestamp**

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

### 475 **4.4.2.3 Signature**

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



#### 477 **4.4.2.3.1 SignedInfo**

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

#### 482 **4.4.2.3.2 SignatureValue**

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

#### 486 **4.4.2.3.3 KeyInfo**

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

#### 490 **4.4.2.4 BinarySecurityToken**

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

#### 497 **4.4.2.5 Signature**

498 This signature is over the entire SOAP body.

#### 499 **4.4.2.5.1 SignedInfo**

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

#### 504 **4.4.2.5.2 SignatureValue**

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

#### 507 **4.4.2.5.3 KeyInfo**

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

#### 511 **4.4.2.6 Body**

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

### 514 4.4.3 Message Processing

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

#### 518 4.4.3.1 Security

#### 519 4.4.3.2 Timestamp

520 The Timestamp element MUST be ignored.

#### 521 4.4.3.3 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 4.4.3.4 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 4.4.3.5 Signature

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

#### 532 4.4.3.6 Body

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

### 535 4.4.4 Example (Non-normative)

536 Here is an example request.

```
537 <?xml version="1.0" encoding="utf-8" ?>
538 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
539 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
540 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
541 <soap:Header>
542 <wsse:Security soap:mustUnderstand="1"
543 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
544 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
545 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
546 </wsu:Timestamp>
547 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
548 <SignedInfo>
549 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
550 />
551 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
552 <Reference URI="#Ping1">
553 <Transforms>
554 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
555 </Transforms>
556 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
557 <DigestValue>AXK...Fe</DigestValue>
558 </Reference>
559 </SignedInfo>
560 <SignatureValue>MQwx...agv</SignatureValue>
561 <KeyInfo>
```

562  
563  
564  
565  
566  
567  
568  
569  
570  
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  
602  
603

```
<wsse:SecurityTokenReference>
  <wsse:KeyIdentifier
ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</KeyInfo>
</Signature>
  <wsse:BinarySecurityToken ValueType="wsse:X509v3"
EncodingType="wsse:Base64Binary"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <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="#body">
      <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>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="body">
  <Ping wsu:Id="Ping1" xmlns="http://xmlsoap.org/Ping">
    <text>Hello</text>
  </Ping>
  <Ping>
    <text>Goodbye</text>
  </Ping>
</soap:Body>
</soap:Envelope>
```

## 4.5 Second Message - Response

### 4.5.1 Message Elements and Attributes

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?
Body	Mandatory

### 4.5.2 Message Creation

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

### 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     <PingResponse>
628       <text>Goodbye</text>
629     </PingResponse>
630   </soap:Body>
631 </soap:Envelope>
```

### 632 **4.6 Other processing**

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

#### 634 **4.6.1 Requester**

635 No additional processing is required.

#### 636 **4.6.2 Responder**

637 No additional processing is required.

### 638 **4.7 Expected Security Properties**

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

---

## 641 **5 Scenario #6 – Encrypt and Sign**

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

### 646 **5.1 Agreements**

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

#### 649 **5.1.1 CERT-VALUE**

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

654 The Responder MUST have access to the Private key corresponding to the Public key in the  
655 certificate.

#### 656 **5.1.2 Signature Trust Root**

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

### 660 **5.2 Parameters**

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

663 No parameters are required.

### 664 **5.3 General Message Flow**

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

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

### 674 **5.4 First Message - Request**

#### 675 **5.4.1 Message Elements and Attributes**

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

678 Items marked optional MAY be generated and MUST be processed if present. Items MUST  
 679 appear in the order specified, except as noted.  
 680

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

681

## 682 **5.4.2 Message Creation**

### 683 **5.4.2.1 Security**

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

### 685 **5.4.2.2 Timestamp**

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

### 688 **5.4.2.3 BinarySecurityToken**

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

### 695 **5.4.2.4 Signature**

696 The signature is over the entire SOAP body.

#### 697 **5.4.2.4.1 SignedInfo**

698 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
699 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body  
700 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod  
701 MUST be SHA1.

#### 702 **5.4.2.4.2 SignatureValue**

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

#### 705 **5.4.2.4.3 KeyInfo**

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

#### 709 **5.4.2.5 EncryptedKey**

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

711 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
712 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
713 MUST have the value of CERT-VALUE.

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

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

#### 718 **5.4.2.6 Body**

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

#### 720 **5.4.2.7 EncryptedData**

721 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
722 EncryptedKey.

723 The Type MUST have the value of #Content.

724 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
725 – CBC.

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

### 728 **5.4.3 Message Processing**

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

#### 732 **5.4.3.1 Security**

#### 733 **5.4.3.2 Timestamp**

734 The Timestamp element MUST be ignored.

#### 735 **5.4.3.3 BinarySecurityToken**

736 The certificate in the token MUST be validated. The Subject of the certificate MUST be an  
737 authorized entity. The public key in the certificate MUST be retained for verification of the  
738 signature.

#### 739 **5.4.3.4 Signature**

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

#### 742 **5.4.3.5 EncryptedKey**

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

#### 745 **5.4.3.6 Body**

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

#### 748 **5.4.3.7 EncryptedData**

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

### 751 **5.4.4 Example (Non-normative)**

752 Here is an example request.

```
753 <?xml version="1.0" encoding="utf-8" ?>
754 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
755 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
756 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
757 <soap:Header>
758 <wsse:Security soap:mustUnderstand="1"
759 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
760 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
761 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
762 </wsu:Timestamp>
763 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
764 EncodingType="wsse:Base64Binary"
765 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
766 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
767 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
768 <SignedInfo>
769 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
770 />
771 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
772 <Reference URI="#body">
```



```

773     <Transforms>
774       <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
775     </Transforms>
776     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
777     <DigestValue>QTV...dw=</DigestValue>
778   </Reference>
779 </SignedInfo>
780 <SignatureValue>H+x0...gUw=</SignatureValue>
781 <KeyInfo>
782   <wsse:SecurityTokenReference>
783     <wsse:Reference URI="#myCert" />
784   </wsse:SecurityTokenReference>
785 </KeyInfo>
786 </Signature>
787 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
788   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
789 />
790   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
791     <wsse:SecurityTokenReference>
792       <wsse:KeyIdentifier
793 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
794     </wsse:SecurityTokenReference>
795   </KeyInfo>
796   <xenc:CipherData>
797     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
798   </xenc:CipherData>
799   <xenc:ReferenceList>
800     <xenc:DataReference URI="#enc" />
801   </xenc:ReferenceList>
802 </xenc:EncryptedKey>
803 </wsse:Security>
804 </soap:Header>
805 <soap:Body wsu:Id="body"
806 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
807   <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
808     xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
809     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
810 cbc" />
811     <xenc:CipherData>
812       <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
813     </xenc:CipherData>
814   </xenc:EncryptedData>
815 </soap:Body>
816 </soap:Envelope>
817

```

## 5.5 Second Message - Response

### 5.5.1 Message Elements and Attributes

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
Timestamp	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

824

## 825 **5.5.2 Message Creation**

### 826 **5.5.2.1 Security**

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

### 829 **5.5.2.2 Timestamp**

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

### 832 **5.5.2.3 Signature**

833 The signature is over the entire SOAP body.

#### 834 **5.5.2.3.1 SignedInfo**

835 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
836 be RSA-SHA1. The Reference MUST specify a relative URI that refers to the SOAP Body  
837 element. The only Transform specified MUST be Exclusive Canonicalization. The DigestMethod  
838 MUST be SHA1.

### 839 **5.5.2.3.2 SignatureValue**

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

### 842 **5.5.2.3.3 KeyInfo**

843 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
844 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
845 MUST have the value of CERT-VALUE.

### 846 **5.5.2.4 BinarySecurityToken**

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

### 850 **5.5.2.5 EncryptedKey**

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

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

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

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

### 859 **5.5.2.6 Body**

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

### 861 **5.5.2.7 EncryptedData**

862 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
863 EncryptedKey.

864 The Type MUST have the value of #Content.

865 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
866 – CBC.

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

## 869 **5.5.3 Message Processing**

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

### 873 **5.5.3.1 Security**

### 874 **5.5.3.2 Timestamp**

875 The Timestamp element MUST be ignored.

### 876 5.5.3.3 Body

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

### 878 5.5.3.4 EncryptedData

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

### 881 5.5.3.5 Signature

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

### 884 5.5.3.6 BinarySecurityToken

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

### 887 5.5.3.7 EncryptedKey

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

## 890 5.5.4 Example (Non-normative)

891 Here is an example response.

```
892 <?xml version="1.0" encoding="utf-8" ?>
893 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
894 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
895 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
896 <soap:Header>
897 <wsse:Security soap:mustUnderstand="1"
898 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
899 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
900 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
901 </wsu:Timestamp>
902 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
903 <SignedInfo>
904 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
905 />
906 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
907 <Reference URI="#body">
908 <Transforms>
909 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
910 </Transforms>
911 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
912 <DigestValue>KxW...5B</DigestValue>
913 </Reference>
914 </SignedInfo>
915 <SignatureValue>8Hkd...al7</SignatureValue>
916 <KeyInfo>
917 <wsse:SecurityTokenReference>
918 <wsse:KeyIdentifier
919 Value="wsse:X509v3">B39R...mY</wsse:KeyIdentifier>
920 </wsse:SecurityTokenReference>
921 </KeyInfo>
922 </Signature>
923 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
924 EncodingType="wsse:Base64Binary"
925 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
926 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
927 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
928 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
929 />
```

```
930 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
931 <wsse:SecurityTokenReference>
932 <wsse:Reference URI="#myCert" />
933 </wsse:SecurityTokenReference>
934 </KeyInfo>
935 <xenc:CipherData>
936 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
937 </xenc:CipherData>
938 <xenc:ReferenceList>
939 <xenc:DataReference URI="#enc" />
940 </xenc:ReferenceList>
941 </xenc:EncryptedKey>
942 </wsse:Security>
943 </soap:Header>
944 <soap:Body wsu:Id="body"
945 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
946 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
947 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
948 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
949 cbc" />
950 <xenc:CipherData>
951 <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
952 </xenc:CipherData>
953 </xenc:EncryptedData>
954 </soap:Body>
955 </soap:Envelope>
```

956

## 957 **5.6 Other processing**

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

### 959 **5.6.1 Requester**

960 No additional processing is required.

### 961 **5.6.2 Responder**

962 No additional processing is required.

## 963 **5.7 Expected Security Properties**

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

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

---

## 971 **6 Scenario #7 – Signed Token**

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

### 977 **6.1 Agreements**

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

#### 980 **6.1.1 CERT-VALUE**

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

985 The Responder MUST have access to the Private key corresponding to the Public key in the  
986 certificate.

#### 987 **6.1.2 Signature Trust Root**

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

### 991 **6.2 Parameters**

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

994 No parameters are required.

### 995 **6.3 General Message Flow**

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

997 This contract covers a request/response MEP over the http binding. SOAP 1.1 MUST be used.  
998 As required by SOAP 1.1, the SOAPAction http header MUST be present. Any value, including a  
999 null string may be used. The recipient SHOULD ignore the value. The request contains a body,  
1000 which is signed and then encrypted. The signature also covers the Token used for encryption.  
1001 The certificate for signing is included in the message. The certificate for encryption is provided  
1002 externally. The Responder decrypts the body and then verifies the signature. If no errors are  
1003 detected it returns the response signing and encrypting the message body. The roles of the key  
1004 pairs are reversed from that of the request, using the signing key to encrypt and the encryption  
1005 key to sign. The signature also covers the Token used for encryption.

1006 **6.4 First Message - Request**

1007 **6.4.1 Message Elements and Attributes**

1008 Items not listed in the following table MAY be present, but MUST NOT be marked with the  
1009 mustUnderstand="1" attribute. Items marked mandatory MUST be generated and processed.  
1010 Items marked optional MAY be generated and MUST be processed if present. Items MUST  
1011 appear in the order specified, except as noted.

1012

<b>Name</b>	<b>Mandatory?</b>
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1013

## 1014 **6.4.2 Message Creation**

### 1015 **6.4.2.1 Security**

1016 The Security element MUST contain the mustUnderstand="1" attribute.

### 1017 **6.4.2.2 Timestamp**

1018 The Created element within the Timestamp SHOULD contain the current local time at the sender  
1019 expressed in the UTC time zone.

### 1020 **6.4.2.3 EncryptedKey**

1021 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1022 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
1023 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
1024 MUST have the value of CERT-VALUE.

1025 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public  
1026 Key specified in the specified X.509 certificate, using the specified algorithm.

1027 The ReferenceList MUST contain a DataReference which has the value of a relative URI that  
1028 refers to the encrypted body of the message.

### 1029 **6.4.2.4 BinarySecurityToken**

1030 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be  
1031 labeled with an Id so it can be referenced by the signature. The value MUST be a PK certificate  
1032 suitable for verifying the signature and encrypting the response. The certificate SHOULD NOT  
1033 have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the  
1034 values of keyEncipherment, dataEncipherment and digitalSignature. The Requester must have  
1035 access to the private key corresponding to the public key in the certificate.

### 1036 **6.4.2.5 Signature**

1037 The signature is over the entire SOAP body.

#### 1038 **6.4.2.5.1 SignedInfo**

1039 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
1040 be RSA-SHA1.

1041 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference  
1042 contained in the EncryptedKey. The STR Dereference Transform and Exclusive Canonicalization  
1043 Transform MUST be specified. The DigestMethod MUST be SHA1.

1044 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The  
1045 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be  
1046 SHA1.

#### 1047 **6.4.2.5.2 SignatureValue**

1048 The SignatureValue MUST be calculated as specified by the specification, using the private key  
1049 corresponding to the public key specified in the certificate in the BinarySecurityToken.

#### 1050 **6.4.2.5.3 KeyInfo**

1051 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which  
1052 indicates the BinarySecurityToken containing the certificate which will be used for signature  
1053 verification.



1054 **6.4.2.6 Body**

1055 The body element MUST be first signed and then its contents encrypted.

1056 **6.4.2.7 EncryptedData**

1057 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
1058 EncryptedKey.

1059 The Type MUST have the value of #Content.

1060 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
1061 – CBC.

1062 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,  
1063 using the specified algorithm.

1064 **6.4.3 Message Processing**

1065 This section describes the processing performed by the Responder. If an error is detected, the  
1066 Responder MUST cease processing the message and issue a Fault with a value of  
1067 FailedAuthentication.

1068 **6.4.3.1 Security**

1069 **6.4.3.2 Timestamp**

1070 The Timestamp element MUST be ignored.

1071 **6.4.3.3 EncryptedKey**

1072 The random key contained in the CipherData MUST be decrypted using the private key  
1073 corresponding to the certificate specified by the KeyIdentifier, using the specified algorithm.

1074 **6.4.3.4 Body**

1075 The contents of the body MUST first be decrypted and then the signature verified. If no errors are  
1076 detected, the body MUST be passed to the application.

1077 **6.4.3.5 EncryptedData**

1078 The message body contents contained in the EncryptedData, referenced by the ReferenceList  
1079 MUST be decrypted using the random key, using the specified algorithm.

1080 **6.4.3.6 BinarySecurityToken**

1081 The certificate in the token MUST be validated. The Subject of the certificate MUST be an  
1082 authorized entity. The public key in the certificate MUST be retained for verification of the  
1083 signature.

1084 **6.4.3.7 Signature**

1085 The body after decryption, MUST be verified against the signature using the specified algorithms  
1086 and transforms and the retained public key.

1087 **6.4.4 Example (Non-normative)**

1088 Here is an example request.

1089 `<?xml version="1.0" encoding="utf-8" ?>`

```

1090 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
1091 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1092 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
1093 <soap:Header>
1094 <wsse:Security soap:mustUnderstand="1"
1095 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1096 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1097 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1098 </wsu:Timestamp>
1099 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1100 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1101 />
1102 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1103 <wsse:SecurityTokenReference wsu:Id="Token">
1104 <wsse:KeyIdentifier
1105 Value="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
1106 </wsse:SecurityTokenReference>
1107 </KeyInfo>
1108 <xenc:CipherData>
1109 <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
1110 </xenc:CipherData>
1111 <xenc:ReferenceList>
1112 <xenc:DataReference URI="#enc" />
1113 </xenc:ReferenceList>
1114 </xenc:EncryptedKey>
1115 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1116 EncodingType="wsse:Base64Binary"
1117 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1118 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1119 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1120 <SignedInfo>
1121 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1122 />
1123 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
1124 <Reference URI="#Token">
1125 <Transforms>
1126 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform#"
1127 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1128 </Transforms>
1129 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1130 <DigestValue>pHrr...xK=</DigestValue>
1131 </Reference>
1132 <Reference URI="#body">
1133 <Transforms>
1134 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1135 </Transforms>
1136 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
1137 <DigestValue>QTV...dw=</DigestValue>
1138 </Reference>
1139 </SignedInfo>
1140 <SignatureValue>H+x0...gUw=</SignatureValue>
1141 <KeyInfo>
1142 <wsse:SecurityTokenReference>
1143 <wsse:Reference URI="#myCert" />
1144 </wsse:SecurityTokenReference>
1145 </KeyInfo>
1146 </Signature>
1147 </wsse:Security>
1148 </soap:Header>
1149 <soap:Body wsu:Id="body"
1150 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1151 <xenc:EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
1152 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1153 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-
1154 cbc" />
1155 <xenc:CipherData>
1156 <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
1157 </xenc:CipherData>
1158 </xenc:EncryptedData>
1159 </soap:Body>
1160 </soap:Envelope>

```

1161

## 1162 6.5 Second Message - Response

### 1163 6.5.1 Message Elements and Attributes

1164 Items not listed in the following table MUST NOT be created or processed. Items marked  
1165 mandatory MUST be generated and processed. Items marked optional MAY be generated and  
1166 MUST be processed if present. Items MUST appear in the order specified, except as noted.

1167

Name	Mandatory?
Security	Mandatory
mustUnderstand="1"	Mandatory
Timestamp	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
Body	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
Cipherdata	Mandatory

1168

## 1169 **6.5.2 Message Creation**

### 1170 **6.5.2.1 Security**

1171 The Security element MUST contain the mustUnderstand="1" attribute. Any other header  
1172 elements MUST NOT be labeled with a mustUnderstand="1" attribute.

### 1173 **6.5.2.2 Timestamp**

1174 The Created element within the Timestamp SHOULD contain the current local time at the sender  
1175 expressed in the UTC time zone.

### 1176 **6.5.2.3 BinarySecurityToken**

1177 The ValueType MUST be X.509 v3. The EncodingType MUST be Base 64. The token MUST be  
1178 labeled with an Id so it can be referenced by the encryption. The certificate must be the one sent  
1179 in the request.

### 1180 **6.5.2.4 EncryptedKey**

1181 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be RSA v1.5.

1182 The KeyInfo MUST contain a SecurityTokenReference with a reference to a relative URI which  
1183 indicates the BinarySecurityToken containing the certificate which will be used for signature  
1184 verification.

1185 The CipherData MUST contain the encrypted form of the random key, encrypted under the Public  
1186 Key specified in the specified X.509 certificate, using the specified algorithm.

1187 The ReferenceList MUST contain a DataReference which has the value of a relative URI that  
1188 refers to the encrypted body of the message.

### 1189 **6.5.2.5 Signature**

1190 The signature is over the entire SOAP body.

#### 1191 **6.5.2.5.1 SignedInfo**

1192 The CanonicalizationMethod MUST be Exclusive Canonicalization. The SignatureMethod MUST  
1193 be RSA-SHA1.

1194 The first Reference MUST specify a relative URI that refers to the SecurityTokenReference  
1195 contained in the EncryptedKey. The STR Dereference Transform and Exclusive Canonicalization  
1196 Transform MUST be specified. The DigestMethod MUST be SHA1.

1197 The second Reference MUST specify a relative URI that refers to the SOAP Body element. The  
1198 only Transform specified MUST be Exclusive Canonicalization. The DigestMethod MUST be  
1199 SHA1.

#### 1200 **6.5.2.5.2 SignatureValue**

1201 The SignatureValue MUST be calculated as specified by the specification, using the private key  
1202 corresponding to the public key specified in the certificate in the BinarySecurityToken.

#### 1203 **6.5.2.5.3 KeyInfo**

1204 The KeyInfo MUST contain a SecurityTokenReference. The SecurityTokenReference MUST  
1205 contain a KeyIdentifier with a ValueType attribute with a value of X509v3. The KeyIdentifier  
1206 MUST have the value of CERT-VALUE.

1207 **6.5.2.6 Body**

1208 The body element MUST be first signed and then its contents encrypted.

1209 **6.5.2.7 EncryptedData**

1210 The EncryptedData MUST be labeled with an Id referenced in the ReferenceList of the  
1211 EncryptedKey.

1212 The Type MUST have the value of #Content.

1213 The EncryptionMethod MUST contain the Algorithm attribute. The algorithm MUST be triple DES  
1214 – CBC.

1215 The CypherData MUST contain the encrypted form of the Body, encrypted under a random key,  
1216 using the specified algorithm.

1217 **6.5.3 Message Processing**

1218 This section describes the processing performed by the Responder. If an error is detected, the  
1219 Responder MUST cease processing the message and report the fault locally with a value of  
1220 FailedAuthentication.

1221 **6.5.3.1 Security**

1222 **6.5.3.2 Timestamp**

1223 The Timestamp element MUST be ignored.

1224 **6.5.3.3 BinarySecurityToken**

1225 The certificate in the token MUST be validated. The Subject of the certificate MUST be an  
1226 authorized entity. The certificate is used to identify the private key to be used for decryption.

1227 **6.5.3.4 EncryptedKey**

1228 The random key contained in the CipherData MUST be decrypted using the private key  
1229 corresponding to the certificate specified by the Reference, using the specified algorithm.

1230 **6.5.3.5 Body**

1231 The contents of the body MUST first be decrypted and then the signature verified.

1232 **6.5.3.6 EncryptedData**

1233 The message body contents contained in the EncryptedData, referenced by the ReferenceList  
1234 MUST be decrypted using the random key, using the specified algorithm.

1235 **6.5.3.7 Signature**

1236 The body after decryption, MUST be verified against the signature using the specified algorithms  
1237 and transforms and the indicated public key.

1238 **6.5.4 Example (Non-normative)**

1239 Here is an example response.

```
1240 <?xml version="1.0" encoding="utf-8" ?>  
1241 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
1242 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
1243 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```

1244 <soap:Header>
1245 <wsse:Security soap:mustUnderstand="1"
1246 xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
1247 <wsu:Timestamp xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
1248 <wsu:Created>2003-03-18T19:53:13Z</wsu:Created>
1249 </wsu:Timestamp>
1250 <wsse:BinarySecurityToken ValueType="wsse:X509v3"
1251 EncodingType="wsse:Base64Binary"
1252 xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
1253 wsu:Id="myCert">MII...hk</wsse:BinarySecurityToken>
1254 <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
1255 <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
1256 />
1257 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
1258 <wsse:SecurityTokenReference wsu:Id="Token">
1259 <wsse:Reference URI="#myCert" />
1260 </wsse:SecurityTokenReference>
1261 </KeyInfo>
1262 <xenc:CipherData>
1263 <xenc:CipherValue>dNYS...fQ</xenc:CipherValue>
1264 </xenc:CipherData>
1265 <xenc:ReferenceList>
1266 <xenc:DataReference URI="#enc" />
1267 </xenc:ReferenceList>
1268 </xenc:EncryptedKey>
1269 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
1270 <SignedInfo>
1271 <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
1272 />
1273 <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1274 <Reference URI="#Token">
1275 <Transforms>
1276 <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform#"
1277 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1278 </Transforms>
1279 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1280 <DigestValue>B4j...Xv</DigestValue>
1281 </Reference>
1282 <Reference URI="#body">
1283 <Transforms>
1284 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1285 </Transforms>
1286 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1287 <DigestValue>KxW...5B</DigestValue>
1288 </Reference>
1289 </SignedInfo>
1290 <SignatureValue>8Hkd...al7</SignatureValue>
1291 <KeyInfo>
1292 <wsse:SecurityTokenReference>
1293 <wsse:KeyIdentifier
1294 ValueType="wsse:X509v3">B39R...mY</wsse:KeyIdentifier>
1295 </wsse:SecurityTokenReference>
1296 </KeyInfo>
1297 </Signature>
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 encryption token  
1323 binds it to the message.

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

```

1388

---

## Appendix B. Revision History

1389

Rev	Date	By Whom	What
wss-01	2003-07-28	Hal Lockhart	Initial version

1390

1391

---

## Appendix C. Notices

1392 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
1393 that might be claimed to pertain to the implementation or use of the technology described in this  
1394 document or the extent to which any license under such rights might or might not be available;  
1395 neither does it represent that it has made any effort to identify any such rights. Information on  
1396 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
1397 website. Copies of claims of rights made available for publication and any assurances of licenses  
1398 to be made available, or the result of an attempt made to obtain a general license or permission  
1399 for the use of such proprietary rights by implementors or users of this specification, can be  
1400 obtained from the OASIS Executive Director.

1401 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
1402 applications, or other proprietary rights which may cover technology that may be required to  
1403 implement this specification. Please address the information to the OASIS Executive Director.

1404 **Copyright © OASIS Open 2002. All Rights Reserved.**

1405 This document and translations of it may be copied and furnished to others, and derivative works  
1406 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
1407 published and distributed, in whole or in part, without restriction of any kind, provided that the  
1408 above copyright notice and this paragraph are included on all such copies and derivative works.  
1409 However, this document itself does not be modified in any way, such as by removing the  
1410 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
1411 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
1412 Property Rights document must be followed, or as required to translate it into languages other  
1413 than English.

1414 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
1415 successors or assigns.

1416 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1417 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
1418 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
1419 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
1420 PARTICULAR PURPOSE.