Web Services Security: Interop 2 Scenarios

Working Draft 03, 26 Aug 2003

Document identifier:
wss-interop2-draft-03.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>

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 list. Others should subscribe to and send comments to the wss-comment@lists.oasis-open.org list. To subscribe, send an email message to wss-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Table of Contents

Introduction

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

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

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

1.1 Terminology

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

2 Test Application

All three scenarios use the same, simple application.
The Requester sends a Ping element with a value of a string.
The Responder returns a PingResponse element with a value of the same string.

3 Scenario #4 Session Key

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

3.1 Agreements

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

3.1.1 SESSION-KEY-VALUE

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

3.1.2 CERT-VALUE

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

3.1.3 Signature Trust Root

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

3.2 Parameters

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

No parameters are required.

3.3 General Message Flow

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

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

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

3.4 First Message - Request

3.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand=”1” attribute. 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
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

3.4.2 Message Creation

3.4.2.1 Security

The Security element MUST contain the mustUnderstand=“1” attribute.

3.4.2.2 ReferenceList

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

3.4.2.3 BinarySecurityToken

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

3.4.2.4 Signature

The signature is over the entire SOAP body.

3.4.2.4.1 SignedInfo

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

3.4.2.4.2 SignatureValue

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

3.4.2.4.3 KeyInfo

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

3.4.2.5 Timestamp

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

3.4.2.6 Body

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

3.4.2.7 EncryptedData

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

The Type MUST have the value of #Content.

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

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

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

3.4.3 Message Processing

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

3.4.3.1 Security

3.4.3.2 ReferenceList

The ReferenceList indicates the data to be decrypted.

3.4.3.3 Timestamp

The Timestamp element MUST be ignored.

3.4.3.4 Body

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

3.4.3.5 EncryptedData

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

3.4.3.6 BinarySecurityToken

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

3.4.3.7 Signature

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

3.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
     <xenc:DataReference URI="#enc" />
   </xenc:ReferenceList>
   <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>
   <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>
```

3.5 Second Message - Response

3.5.1 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
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

3.5.2 Message Creation

3.5.2.1 Security

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

3.5.2.2 ReferenceList

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

3.5.2.3 Signature

The signature is over the entire SOAP body.

3.5.2.3.1 SignedInfo

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

3.5.2.3.2 SignatureValue

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

3.5.2.3.3 KeyInfo

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

3.5.2.4 Timestamp

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

3.5.2.5 Body

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

3.5.2.6 EncryptedData

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

The Type MUST have the value of #Content.

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

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

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

3.5.3 Message Processing

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

3.5.3.1 Security

3.5.3.2 ReferenceList

The ReferenceList indicates the data to be decrypted

3.5.3.3 Timestamp

The Timestamp element MUST be ignored.

3.5.3.4 Body

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

3.5.3.5 EncryptedData

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

3.5.3.6 Signature

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

3.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
     <xenc:DataReference URI="#enc" />
   </xenc:ReferenceList>
   <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>KxW...5B=</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>8Hkd...al7=</SignatureValue>
    <KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier
ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
     </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>d2s...GQ=</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

3.6 Other processing

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

3.6.1 Requester

No additional processing is required.

3.6.2 Responder

No additional processing is required.

3.7 Expected Security Properties

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

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

4 Scenario #5 – Overlapping Signatures

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

4.1 Agreements

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

4.1.1 CERT-VALUE

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

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

4.1.2 Signature Trust Root

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

4.2 Parameters

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

No parameters are required.

4.3 General Message Flow

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

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

4.4 First Message - Request

4.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand="1" attribute. 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
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

4.4.2 Message Creation

4.4.2.1 Security

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

4.4.2.2 Signature

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

4.4.2.2.1 SignedInfo

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

4.4.2.2.2 SignatureValue

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

4.4.2.2.3 KeyInfo

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

4.4.2.3 BinarySecurityToken

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

4.4.2.4 Signature

This signature is over the entire SOAP body.

4.4.2.4.1 SignedInfo

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

4.4.2.4.2 SignatureValue

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

4.4.2.4.3 KeyInfo

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

4.4.2.5 Timestamp

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

4.4.2.6 Body

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

4.4.3 Message Processing

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

4.4.3.1 Security

4.4.3.2 Signature

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

4.4.3.3 BinarySecurityToken

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

4.4.3.4 Signature

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

4.4.3.5 Timestamp

The Timestamp element MUST be ignored.

4.4.3.6 Body

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

4.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <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>AXK...Fe=</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>MQwx...agv=</SignatureValue>
    <KeyInfo>
     <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="#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>
```

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.

4.5.3 Message Processing

The body is passed to the application without modification.

4.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PingResponse xmlns="http://xmlsoap.org/Ping">
   <text>Hello</text>
  </PingResponse>
</soap:Body>
</soap:Envelope>
```

4.6 Other processing

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

4.6.1 Requester

No additional processing is required.

4.6.2 Responder

No additional processing is required.

4.7 Expected Security Properties

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

5 Scenario #6 – Encrypt and Sign

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

5.1 Agreements

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

5.1.1 CERT-VALUE

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

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

5.1.2 Signature Trust Root

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

5.2 Parameters

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

No parameters are required.

5.3 General Message Flow

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

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

5.4 First Message - Request

5.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand="1" attribute. 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
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

5.4.2 Message Creation

5.4.2.1 Security

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

5.4.2.2 BinarySecurityToken

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

5.4.2.3 Signature

The signature is over the entire SOAP body.

5.4.2.3.1 SignedInfo

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

5.4.2.3.2 SignatureValue

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

5.4.2.3.3 KeyInfo

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

5.4.2.4 EncryptedKey

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

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

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

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

5.4.2.5 Timestamp

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

5.4.2.6 Body

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

5.4.2.7 EncryptedData

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

The Type MUST have the value of #Content.

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

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

5.4.3 Message Processing

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

5.4.3.1 Security

5.4.3.2 BinarySecurityToken

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

5.4.3.3 Signature

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

5.4.3.4 EncryptedKey

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

5.4.3.5 Timestamp

The Timestamp element MUST be ignored.

5.4.3.6 Body

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

5.4.3.7 EncryptedData

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

5.4.4 Example (Non-normative)

Here is an example request.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <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>
   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
/>
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
     <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier
ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#enc" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
  <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:CipherData>
    <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

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
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

5.5.2 Message Creation

5.5.2.1 Security

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

5.5.2.2 Signature

The signature is over the entire SOAP body.

5.5.2.2.1 SignedInfo

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

5.5.2.2.2 SignatureValue

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

5.5.2.2.3 KeyInfo

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

5.5.2.3 BinarySecurityToken

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

5.5.2.4 EncryptedKey

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

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

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

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

5.5.2.5 Timestamp

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

5.5.2.6 Body

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

5.5.2.7 EncryptedData

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

The Type MUST have the value of #Content.

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

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

5.5.3 Message Processing

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

5.5.3.1 Security

5.5.3.2 Timestamp

The Timestamp element MUST be ignored.

5.5.3.3 Body

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

5.5.3.4 EncryptedData

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

5.5.3.5 Signature

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

5.5.3.6 BinarySecurityToken

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

5.5.3.7 EncryptedKey

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

5.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <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>KxW...5B=</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>8Hkd...al7=</SignatureValue>
    <KeyInfo>
     <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>
   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
/>
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
     <wsse:SecurityTokenReference>
      <wsse:Reference URI="#myCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#enc" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
  <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:CipherData>
    <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

5.6 Other processing

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

5.6.1 Requester

No additional processing is required.

5.6.2 Responder

No additional processing is required.

5.7 Expected Security Properties

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

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

6 Scenario #7 – Signed Token

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

6.1 Agreements

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

6.1.1 CERT-VALUE

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

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

6.1.2 Signature Trust Root

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

6.2 Parameters

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

No parameters are required.

6.3 General Message Flow

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

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

6.4 First Message - Request

6.4.1 Message Elements and Attributes

Items not listed in the following table MAY be present, but MUST NOT be marked with the mustUnderstand=”1” attribute. 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
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

6.4.2 Message Creation

6.4.2.1 Security

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

6.4.2.2 EncryptedKey

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

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

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

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

6.4.2.3 BinarySecurityToken

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

6.4.2.4 Signature

The signature is over the entire SOAP body.

6.4.2.4.1 SignedInfo

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

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

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

6.4.2.4.2 SignatureValue

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

6.4.2.4.3 KeyInfo

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

6.4.2.5 Timestamp

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

6.4.2.6 Body

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

6.4.2.7 EncryptedData

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

The Type MUST have the value of #Content.

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

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

6.4.3 Message Processing

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

6.4.3.1 Security

6.4.3.2 EncryptedKey

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

6.4.3.3 Timestamp

The Timestamp element MUST be ignored.

6.4.3.4 Body

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

6.4.3.5 EncryptedData

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

6.4.3.6 BinarySecurityToken

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

6.4.3.7 Signature

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

6.4.4 Example (Non-normative)

Here is an example request.

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

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
/>
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
     <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier
ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#enc" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
   <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="#Token">
      <Transforms>
       <Transform Algorithm=http://schemas.xmlsoap.org/2003/06/STR-Transform">
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#/>
       </Transform>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>pHrr...xK=</DigestValue>
     </Reference>
     <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 wsu:Id="Token">
      <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:CipherData>
    <xenc:CipherValue>AYb...Y8=</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

6.5 Second Message - Response

6.5.1 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
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

6.5.2 Message Creation

6.5.2.1 Security

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

6.5.2.2 BinarySecurityToken

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

6.5.2.3 EncryptedKey

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

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

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

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

6.5.2.4 Signature

The signature is over the entire SOAP body.

6.5.2.4.1 SignedInfo

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

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

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

6.5.2.4.2 SignatureValue

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

6.5.2.4.3 KeyInfo

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

6.5.2.5 Timestamp

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

6.5.2.6 Body

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

6.5.2.7 EncryptedData

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

The Type MUST have the value of #Content.

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

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

6.5.3 Message Processing

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

6.5.3.1 Security

6.5.3.2 BinarySecurityToken

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

6.5.3.3 EncryptedKey

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

6.5.3.4 Timestamp

The Timestamp element MUST be ignored.

6.5.3.5 Body

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

6.5.3.6 EncryptedData

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

6.5.3.7 Signature

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

6.5.4 Example (Non-normative)

Here is an example response.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Header>
  <wsse:Security soap:mustUnderstand="1"
xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
   <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>
   <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
/>
    <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
     <wsse:SecurityTokenReference>
      <wsse:Reference URI="#myCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#enc" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>
   <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="#Token">
      <Transforms>
       <Transform Algorithm="http://schemas.xmlsoap.org/2003/06/STR-Transform">
        <CanonicalizationMethod Algorithm=""http://www.w3.org/2001/10/xml-exc-
c14n#"/>
       </Transform>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue>B4j...Xv=</DigestValue>
     </Reference>
     <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>KxW...5B=</DigestValue>
     </Reference>
    </SignedInfo>
    <SignatureValue>8Hkd...al7=</SignatureValue>
    <KeyInfo>
     <wsse:SecurityTokenReference wsu:Id="Token">
      <wsse:KeyIdentifier
ValueType="wsse:X509v3">B39R...mY=</wsse:KeyIdentifier>
     </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:CipherData>
    <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </soap:Body>
</soap:Envelope>
```

6.6 Other processing

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

6.6.1 Requester

No additional processing is required.

6.6.2 Responder

No additional processing is required.

6.7 Expected Security Properties

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

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

7 References

7.1 Normative

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

Appendix A. Ping Application WSDL File

```
<definitions xmlns:tns="http://xmlsoap.org/Ping"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://xmlsoap.org/Ping" name="Ping">
   <types>
          <schema targetNamespace="http://xmlsoap.org/Ping"
xmlns="http://www.w3.org/2001/XMLSchema">
                 <complexType name="ping">
                        <sequence>
                               <element name="text" type="xsd:string"
nillable="true"/>
                               <element name="ticket" type="xsd:string"
minOccurs="0"/>
                        </sequence>
                 </complexType>
                 <complexType name="pingResponse">
                        <sequence>
                               <element name="text" type="xsd:string"
nillable="true"/>
                        </sequence>
                 </complexType>
                 <element name="Ping" type="tns:ping"/>
                 <element name="PingResponse" type="tns:pingResponse"/>
          </schema>
   </types>
   <message name="PingRequest">
          <part name="ping" element="tns:Ping"/>
   </message>
   <message name="PingResponse">
          <part name="pingResponse" element="tns:PingResponse"/>
   </message>
   <portType name="PingPort">
          <operation name="Ping">
                 <input message="tns:PingRequest"/>
                 <output message="tns:PingResponse"/>
          </operation>
   </portType>
   <binding name="PingBinding" type="tns:PingPort">
          <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
          <operation name="Ping">
                 <soap:operation/>
                 <input>
                        <soap:body use="literal"/>
                 </input>
                 <output>
                        <soap:body use="literal"/>
                 </output>
          </operation>
   </binding>
   <service name="PingService">
          <port name="PingPort" binding="tns:PingBinding">
                 <soap:address
location="http://localhost:8080/pingejb/Ping"/>
          </port>
   </service>
```

```
</definitions>
```

Appendix B. Revision History

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

Appendix C. Notices