Web Services Security SOAP Messages with Attachments (SwA) Profile 1.0

Interop 1 Scenarios

Working Draft 06, 1 Nov 2004

Document identifier:

swa-interop1-draft-06.doc

Location:

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

Editor:

Blake Dournaee, Sarvega Inc. <blake@sarvega.com>

Contributors:

Bruce Rich, IBM <brich@us.ibm.com>
Maneesh Sahu, Actional <maneesh@actional.com>
Frederick Hirsch, Nokia <Frederick.Hirsch@nokia.com>
Manveen Kaur, Sun <Manveen.Kaur@sun.com>

Abstract:

This document formalizes the interoperability scenarios to be used in the first Web Services Security SwA Profile 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 message exchanges to be tested during the first interoperability event of the Web Services Security SOAP Message with Attachments Profile. All scenarios use the Request/Response Message Exchange Pattern (MEP) with no intermediaries. All scenarios invoke the same simple application. To avoid confusion, they are called Scenario #1 through Scenario #4.

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 four scenarios use the same, simple application.

The Requester sends a Ping element with a value of a string as the single child to a SOAP request. The value should be the name of the organization that has developed the software and the number of the scenario, e.g. “Acme Corp. – Scenario #1”.

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

Each interaction will also include a SOAP attachment secured via one of the content level security mechanisms described in **[WSS-SwA]**. For the purpose of these interoperability scenarios, the Ping request and response elements will not have security properties applied to them; they are used only to keep track of the specific scenarios.

2.1 Example Ping Element

```
<Ping xmlns="http://xmlsoap.org/Ping">
   <text>Acme Corp. – Scenario #1</text>
</Ping>
```

2.2 Example PingResponse Element

```
<PingResponse xmlns="http://xmlsoap.org/Ping">
   <text> Acme Corp. – Scenario #1</text>
</PingResponse>
```

2.3 SOAP Message Packages

When SOAP attachments are used as specified in **[SwA]** each SOAP message is accompanied by a MIME header and possibly multiple boundary parts. This is known as a SOAP message package. All interoperability scenarios in this document assume that a proper SOAP message package is constructed using the HTTP and MIME headers appropriate to **[SwA]**.

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

```
Content-Type: Multipart/Related; boundary=boundary1; type=”text/xml”; start="<foo>"
```

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

```
--boundary1
Content-Type: text/xml; charset=utf-8
Content-ID: <foo>

<?xml version='1.0' ?>
<s11:Envelope xmlns:s11=”http://schemas.xmlsoap.org/soap/envelope/”  />
```

Interoperability of the SOAP message package format, including the appropriate use of the MIME header and boundary semantics, is outside the scope of this interoperability document.

2.4 URI Shorthand Notation

For brevity, the following shorthand is used in describing URI strings:

URI	Shorthand
http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform	#Attachment-Content-Only-Transform

http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0#Attachment-Complete-Transform	#Attachment-Complete-Transform

181

3 Scenario #1: Attachment Signature

Scenario #1 tests the interoperability of a signed attachment using an X.509 certificate. The certificate used to verify the signature shall be present in the SOAP header. No security properties are applied to any part of the SOAP envelope..

3.1 Attachment Properties

This section specifies the attachment properties BEFORE security operations are applied. The Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope.

3.2 Agreements

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

3.2.1 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.

3.2.2 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.

3.3 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.

3.4 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. The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**. 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 signed attachment. The certificate used for signing is included in the message.

The Responder verifies the signature over the attachment. If no errors are detected it returns the response with no additional security properties.

3.5 First Message – Request

3.5.1 Message Elements and Attributes

Elements 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?**
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand=“1”	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

3.5.2 Message Creation

3.5.2.1 Security

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

3.5.2.2 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST be labeled with an Id so it can be referenced by the signature. The value MUST be a public key certificate suitable for verifying the signature. 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.

3.5.2.3 Signature

The signature is over the attachment content only, using the #Attachment-Content-Only-Transform

3.5.2.3.1 SignedInfo

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

3.5.2.3.2 Reference

The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the attachment. The only Transform specified MUST be #Attachment-Content-Only. The DigestMethod MUST be SHA1.

3.5.2.3.3 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.

3.5.2.3.4 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.

3.5.2.4 Post Operation Attachment Properties

This section specifies the attachment properties AFTER security operations are applied. The Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope.

3.5.3 Responder 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.

3.5.3.1 Security

3.5.3.2 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.

3.5.3.3 Signature

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

3.5.3.4 Attachment

After the attachment's signature has been verified, it should be passed to the application.

3.5.4 Example (Non-normative)

```
Content-Type: multipart/related; boundary=”sig-example”; type=”text/xml”
--sig-example
Content-Type: text/xml; charset=utf-8

<?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="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
     secext-1.0.xsd">

    <!-- This is the certificate used to verify the signature -->
    <wsse:BinarySecurityToken ValueType=“http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3”
     EncodingType=“http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary” xmlns:wsu="http://docs.oasis
     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="mySigCert">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="cid:signature">
      <Transforms>
       <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform"/>
      </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="#mySigCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
   </Signature>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>Acme Corp. - Scenario #1</text>
  </Ping>
 </soap:Body>
</soap:Envelope>

--sig-example
Content-Type: image/jpeg
Content-Id: <signature>
Content-Transfer-Encoding: base64

Dcg3AdGFcFs3764fddSArk
```

3.6 Second Message - Response

3.6.1 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?**
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

3.6.2 Message Creation

3.6.2.1 Security

There are no security properties on the response message

3.6.2.2 Body

The body element MUST be not be signed or encrypted

3.6.3 Message Processing

The response is passed to the application without modification.

3.6.4 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> Acme Corp. - Scenario #1</text>
 </PingResponse>
 </soap:Body>
</soap:Envelope>
```

3.7 Other processing

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

3.7.1 Requester

No additional processing is required.

3.7.2 Responder

No additional processing is required.

3.8 Expected Security Properties

Use of the service is restricted to authorized parties that sign the attachment. The attachment of the request is protected against modification and interception. The response does not have any security properties.

4 Scenario #2 – Attachment Encryption

The SOAP request has an attachment that has been encrypted. The encryption is done using a symmetric cipher. The symmetric encryption key is further encrypted for a specific recipient identified by an X.509 certificate. The certificate associated with the key encryption is provided to the requestor out-of-band. No security properties are applied to any part of the SOAP envelope.

4.1 Attachment Properties

This section specifies the attachment properties BEFORE security operations are applied. The Content-Type of the attachment MUST be image/jpeg. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope.

4.2 Agreements

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

4.2.1 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.

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

4.2.2 Signature Trust Root

There is no digital signature operation for this scenario

4.3 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.

4.4 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. The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**. The Content-Transfer-Encoding for the encrypted attachment MUST be base64. 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 an encrypted SOAP attachment. The attachment is encrypted with a random symmetric key, which is encrypted using a public key certificate. The certificate used for the encryption is provided to the Requestor out of band. The Responder decrypts the attachment using the symmetric key which is decrypted with the matching private key. If no errors are detected it returns the response without any security properties. If there is a decryption failure a fault is returned as outlined in section 4.5.3.

4.5 First Message - Request

4.5.1 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?**
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand=“1”	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory
CipherReference	Mandatory
Transforms	Mandatory
Transform	Mandatory
Body	Mandatory
Ping	Mandatory

4.5.2 Message Creation

4.5.2.1 Security

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

4.5.2.2 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST be labeled with an Id so it can be referenced EncryptedKey security token reference. The value MUST be a Public Key certificate suitable for symmetric key encryption. The certificate SHOULD NOT have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the values of keyEncipherment and dataEncipherment. The Responder must have access to the private key corresponding to the public key in the certificate.

4.5.2.3 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 to decrypt the symmetric key.

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 EncryptedData element that refers to the encrypted attachment.

4.5.2.4 EncryptedData

The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be present and it MUST have a value of #Attachment-Content-Only. The EncryptedData element MUST be referenced by the ReferenceList element in the EncryptedKey element. The EncryptedData MUST have a MimeType attribute with the value of image/jpeg.

4.5.2.5 EncryptionMethod

The encryption method MUST be Triple-DES in CBC mode.

4.5.2.6 CipherData

The CipherData MUST refer to the encrypted attachment with a CipherReference element. The CipherReference element MUST refer to the attachment using a URI with a cid scheme. The CipherReference must have a single Transforms element with a single Transform child with an Algorithm attribute value of #Attachment-Content-Only-Transform.

4.5.2.7 Body

The body element MUST not have any security operations applied to it.

4.5.2.8 Ping

The Ping element should contain the scenario number and the name of the entity performing the request.

4.5.2.9 Post Operation Attachment Properties

This section specifies the attachment properties AFTER security operations are applied. The Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id MUST match the Content-Id before encryption.

4.5.3 Responder 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 FailedDecryption.

4.5.3.1 Security

4.5.3.2 BinarySecurityToken

The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation of the public key is required. The responder MUST have the matching private key.

4.5.3.3 EncryptedKey

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

4.5.3.4 EncryptedData

The attachment referred to by the EncrypteData MUST be decrypted using the encrypted symmetric key.

4.5.3.5 Attachment

After decrypting the attachment, it should be passed to the application

4.5.4 Example (Non-normative)

Here is an example request.

```
Content-Type: multipart/related; boundary=”enc-example”; type=”text/xml”
--enc-example
Content-Type: text/xml; charset=utf-8

<?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="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">

   <!-- This certificate is used for symmetric key encryption -->
   <wsse:BinarySecurityToken
     ValueType=“http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3”
     EncodingType=“http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary”
   xmlns:wsu="httphttp://docs.oasis
     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="myEncCert">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="#myEncCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#encrypted-attachment" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>

    <!-- The EncryptedData portion here refers to content of the attachment -->

    <xenc:EncryptedData Id="encrypted-attachment"
    Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
1.0#Attachment-Content-Only" MimeType="image/jpeg">
   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
   <xenc:CipherData>
    <xenc:CipherReference URI="cid:enc">
      <xenc:Transforms>
        <ds:Transform
         Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
profile-1.0#Attachment-Content-Only-Transform" />
        <ds:Transform
      </xenc:Transforms>
    </xenc:CipherReference>
   </xenc:CipherData>
  </xenc:EncryptedData>

  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>Acme Corp. - Scenario #2</text>
  </Ping>
</soap:Body>
</soap:Envelope>
--enc-example
Content-Type: application/octet-stream
Content-Id: <enc>
Content-Transfer-Encoding: base64

Dsh5SA3thsRh3Dh54wafDhjaq2
```

4.6 Second Message - Response

4.6.1 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?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

4.6.2 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.

4.6.2.1 Security

There are no security properties on the response message

4.6.2.2 Body

The body element MUST be not be signed or encrypted

4.6.3 Message Processing

The response is passed to the application without modification.

4.6.4 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> Acme Corp. - Scenario #2</text>
  </PingResponse>
</soap:Body>
</soap:Envelope>
```

4.7 Other processing

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

4.7.1 Requester

No additional processing is required.

4.7.2 Responder

No additional processing is required.

4.8 Expected Security Properties

The attachment content is private for the holder of the appropriate private key. There should be no inferences made regarding the authenticity of the sender. The response is not protected in any way.

5 Scenario #3 – Attachment Signature and Encryption

The SOAP request contains an attachment that has been signed and then encrypted. The certificate associated with the encryption is provided out-of-band to the requestor. The certificate used to verify the signature is provided in the header. The Response Body is not signed or encrypted. There are two certificates in the request message. One identifiers the recipient of the encrypted attachment and one identifies the signer.

5.1 Attachment Properties

This section specifies the attachment properties BEFORE security operations are applied. The Content-Type of the attachment MUST be text/xml. The Content-Transfer-Encoding MUST be 8bit ASCII (8bit). The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope. An example of what the attachment may look like before encryption and signing is shown as follows. This example is non-normative.

```
--enc-sig-example
Content-Type: text/xml; charset=utf-8
Content-Transfer-Encoding: 8bit
Content-ID: <encsignexample>

<?xml version=1.0” encoding=”utf-8”?>
<somexml/>
```

5.2 Agreements

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

5.2.1 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.

5.2.2 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.

5.3 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.

5.4 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. The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**. The Content-Transfer-Encoding for the encrypted attachment MUST be base64. 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 an attachment, which is signed and then encrypted. The certificate for encryption is provided externally to the requestor but conveyed in the request message. The attachment is encrypted with a random symmetric key that is encrypted with a public key certificate. The certificate for signing is included in the message. The Responder decrypts the attachment using its private key and then verifies the signature using the included public key certificate. If no errors are detected it returns the Response with no security properties.

5.5 First Message - Request

5.5.1 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?**
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory
CipherReference	Mandatory

Transforms	Mandatory
Transform	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

5.5.2 Message Creation

5.5.2.1 Security

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

5.5.2.2 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST be labeled with an Id so it can be uniquely referenced. The value MUST be a PK certificate suitable for encrypting the content. The certificate SHOULD NOT have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of keyEncipherment and dataEncipherment.

5.5.2.3 EncryptedKey

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

The KeyInfo MUST contain a SecurityTokenReference with a Reference child that points to the X.509 certificate of the recipient. The Reference child should point to a relative URI which indicates the BinarySecurityToken containing the certificate which will be used to decrypt the symmetric key.

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 EncryptedData element that refers to the encrypted attachment.

5.5.2.4 EncryptedData

The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be present and it MUST have a value of #Attachment-Content-Only. The EncryptedData element MUST be referenced by the ReferenceList element in the EncryptedKey element. The EncryptedData MUST have a MimeType attribute with the value of text/xml.

5.5.2.5 EncryptionMethod

The encryption method MUST be Triple-DES in CBC mode.

5.5.2.6 CipherData

The CipherData MUST refer to the encrypted attachment with a CipherReference element. The CipherReference element MUST refer to the attachment using a URI with a cid scheme. The CipherReference must have a Transforms child with a single Transform sub child with the value of #Attachment-Content-Only-Transform.

5.5.2.7 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. 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. 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.

5.5.2.8 Signature

The signature is over the attachment content only, using the #Attachment-Content-Only-Transform

5.5.2.8.1 SignedInfo

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

5.5.2.8.2 Reference

The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the attachment. The only Transform specified MUST be #Attachment-Content-Only. The DigestMethod MUST be SHA1.

5.5.2.8.3 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.

5.5.2.8.4 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.

5.5.2.9 Body

The contents of the body MUST NOT be encrypted or signed

5.5.2.10 Post Operation Attachment Properties

This section specifies the attachment properties AFTER security operations are applied. The Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id MUST match the Content-Id before encryption.

5.5.3 Responder 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.

5.5.3.1 Security

5.5.3.2 BinarySecurityToken

The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation of the public key is required. The responder MUST have the matching private key.

5.5.3.3 EncryptedKey

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

5.5.3.4 EncryptedData

The attachment referred to by the EncryptedData MUST be decrypted using the encrypted symmetric key.

5.5.3.5 Attachment

After decrypting the attachment, it should have its signature verified

5.5.3.6 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.

5.5.3.7 Signature

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

5.5.3.8 Attachment

After the attachment’s signature has been verified, it should be passed to the application

5.5.4 Example (Non-normative)

Here is an example request.

```
Content-Type: multipart/related; boundary=”enc-sig-example”; type=”text/xml”
--enc-sig-example
Content-Type: text/xml; charset=utf-8
```

```
<?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="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">

   <!-- This certificate is used for symmetric key encryption -->
   <wsse:BinarySecurityToken
     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
     EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
   xmlns:wsu="httphttp://docs.oasis
     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="myEncCert">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="#myEncCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#encrypted-signed-attachment" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>

    <!-- The EncryptedData portion here refers to content of the attachment -->

    <xenc:EncryptedData Id="encrypted-signed-attachment"
    Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
1.0#Attachment-Content-Only" MimeType="text/xml">
   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
   <xenc:CipherData>
    <xenc:CipherReference URI="cid:encsignexample">
      <xenc:Transforms>
        <ds:Transform
         Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
profile-1.0#Attachment-Content-Only-Transform" />
        </xenc:Transforms>
    </xenc:CipherReference>
   </xenc:CipherData>
  </xenc:EncryptedData>

  <!-- This certificate is used to verify the signature -->
  <wsse:BinarySecurityToken ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" xmlns:wsu="http://docs.oasis
   open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   wsu:Id="mySigCertCert">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="cid:encsignexample">
      <Transforms>
       <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform"/>
      </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="#mySigCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
   </Signature>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>Acme Corp. - Scenario #3</text>
  </Ping>
</soap:Body>
</soap:Envelope>
--enc-sig-example
Content-Type: application/octet-stream
Content-Id: <encsignexample>
Content-Transfer-Encoding: base64

FEWMMIIfc93ASjfdjsa358sa98xsjcx
```

5.6 Second Message - Response

5.6.1 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?
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

5.6.2 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.

5.6.2.1 Security

There are no security properties on the response message

5.6.2.2 Body

The body element MUST be not be signed or encrypted

5.6.3 Message Processing

The response is passed to the application without modification.

5.6.4 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> Acme Corp. - Scenario #3</text>
  </PingResponse>
</soap:Body>
</soap:Envelope>
```

5.7 Other processing

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

5.7.1 Requester

No additional processing is required.

5.7.2 Responder

No additional processing is required.

5.8 Expected Security Properties

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

6 Scenario #4 – Attachment Signature and Encryption with MIME Headers

The SOAP request contains an attachment that has been signed and then encrypted. The certificate associated with the encryption is provided out-of-band to the requestor. The certificate used to verify the signature is provided in the header. The Response Body is not signed or encrypted. There are two certificates in the request message. One identifies the recipient of the encrypted attachment and one identifies the signer. This scenario differs from the first three scenarios in that it covers MIME headers in the signature and encryption. This means that it uses the Attachment-Complete Signature Reference Transform and Attachment-Complete EncryptedData Type.

Aside from these two changes, this scenario is identical to Scenario #3.

6.1 Attachment Properties

This section specifies the attachment properties BEFORE security operations are applied. The Content-Type of the attachment MUST be text/xml. The Content-Transfer-Encoding MUST be 8bit ASCII (8bit). The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope. An example of what the attachment may look like before encryption and signature is shown as follows:

```
--enc-sig-headers-example
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <enc-sig-headers-example>

<?xml version=1.0” encoding=”utf-8”?>
<somexml/>
```

6.2 Agreements

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

6.2.1 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.

6.2.2 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.

6.3 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.

6.4 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. The SOAP envelope MUST be wrapped in a SOAP Message Package as specified by **[SwA]**. The Content-Transfer-Encoding for the encrypted attachment MUST be base64. 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 an attachment, which is signed and then encrypted. The certificate for encryption is provided externally to the requestor but conveyed in the request message. The attachment is encrypted with a random symmetric key that is encrypted with a public key certificate. The certificate for signing is included in the message. The Responder decrypts the attachment using its private key and then verifies the signature using the included public key certificate. If no errors are detected it returns the Response with no security properties.

6.5 First Message - Request

6.5.1 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?**
Envelope	Mandatory
Header	Mandatory
Security	Mandatory
mustUnderstand="1"	Mandatory
BinarySecurityToken	Mandatory
EncryptedKey	Mandatory
EncryptionMethod	Mandatory
KeyInfo	Mandatory
SecurityTokenReference	Mandatory
CipherData	Mandatory
ReferenceList	Mandatory
EncryptedData	Mandatory
EncryptionMethod	Mandatory
CipherData	Mandatory

CipherReference	Mandatory
Transforms	Mandatory
Transform	Mandatory
BinarySecurityToken	Mandatory
Signature	Mandatory
SignedInfo	Mandatory
CanonicalizationMethod	Mandatory
SignatureMethod	Mandatory
Reference	Mandatory
Transforms	Mandatory
Transform	Mandatory
SignatureValue	Mandatory
KeyInfo	Mandatory
Body	Mandatory
Ping	Mandatory

6.5.2 Message Creation

6.5.2.1 Security

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

6.5.2.2 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. The token MUST be labeled with an Id so it can be uniquely referenced. The value MUST be a PK certificate suitable for encrypting the content. The certificate SHOULD NOT have a KeyUsage extension. If it does contain a KeyUsage extension, it SHOULD include the value of keyEncipherment and dataEncipherment.

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 child that points to the X.509 certificate of the recipient. The Reference child should point to a relative URI which indicates the BinarySecurityToken containing the certificate which will be used to decrypt the symmetric key.

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 EncryptedData element that refers to the encrypted attachment.

6.5.2.4 EncryptedData

The EncryptedData element refers to the encrypted attachment. The Type attribute MUST be present and it MUST have a value of #Attachment-Complete. The EncryptedData element MUST be referenced by the ReferenceList element in the EncryptedKey element. The EncryptedData MUST have a MimeType attribute with the value of text/xml.

6.5.2.5 EncryptionMethod

The encryption method MUST be Triple-DES in CBC mode.

6.5.2.6 CipherData

The CipherData MUST refer to the encrypted attachment with a CipherReference element. The CipherReference element MUST refer to the attachment using a URI with a cid scheme. The CipherReference must have a Transforms child with a single Transform sub child with the value of #Attachment-Content-Only-Transform.

6.5.2.7 BinarySecurityToken

The ValueType MUST be X509v3. The EncodingType MUST be Base64Binary. 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. 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.

6.5.2.8 Signature

The signature is over the attachment content only, using the #Attachment-Content-Only-Transform

6.5.2.8.1 SignedInfo

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

6.5.2.8.2 Reference

The Reference MUST specify a URI using the cid scheme that points to the Content-Id of the attachment. The only Transform specified MUST be #Attachment-Content-Only. The DigestMethod MUST be SHA1.

6.5.2.8.3 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.

6.5.2.8.4 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.

6.5.2.9 Body

The contents of the body MUST not be encrypted or signed

6.5.2.10 Post Operation Attachment Properties

This section specifies the attachment properties AFTER security operations are applied. The Content-Type of the attachment MUST be application/octet-stream. The Content-Transfer-Encoding MUST be base64. The attachment MUST have a Content-Id header that uniquely identifies the attachment. The generation of the Content-Id header is out of scope. The Content-Id MUST match the Content-Id before encryption.

6.5.3 Responder 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.

6.5.3.1 Security

6.5.3.2 BinarySecurityToken

The public key in the certificate MUST be used to decrypt the symmetric key. No trust validation of the public key is required. The responder MUST have the matching private key.

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 SecurityTokenReference, using the specified algorithm.

6.5.3.4 EncryptedData

The attachment referred to by the EncryptedData MUST be decrypted using the encrypted symmetric key.

6.5.3.5 Attachment

After decrypting the attachment, it should have its signature verified

6.5.3.6 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.

6.5.3.7 Signature

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

6.5.3.8 Attachment

After the attachment’s signature has been verified, it should be passed to the application

6.5.4 Example (Non-normative)

Here is an example request.

```
Content-Type: multipart/related; boundary=”enc-sig-headers-example”;
type=”text/xml”
```

```
--enc-sig-headers-example
Content-Type: text/xml; charset=utf-8

<?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="docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">

   <!-- This certificate is used for symmetric key encryption -->
   <wsse:BinarySecurityToken
     ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
     EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary"
   xmlns:wsu="httphttp://docs.oasis
     open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="myEncCert">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="#myEncCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
    <xenc:CipherData>
     <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
    </xenc:CipherData>
    <xenc:ReferenceList>
     <xenc:DataReference URI="#encrypted-signed-attachment" />
    </xenc:ReferenceList>
   </xenc:EncryptedKey>

    <!-- The EncryptedData portion here refers to content of the attachment -->

    <xenc:EncryptedData Id="encrypted-signed-attachment-headers"
    Type="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-
1.0#Attachment-Complete" MimeType="text/xml">
   xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
   <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
   <xenc:CipherData>
    <xenc:CipherReference URI="cid:encsign-headers-example">
      <xenc:Transforms>
        <ds:Transform
         Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-
profile-1.0#Attachment-Content-Only-Transform" />
      </xenc:Transforms>
    </xenc:CipherReference>
   </xenc:CipherData>
  </xenc:EncryptedData>

  <!-- This certificate is used to verify the signature -->
  <wsse:BinarySecurityToken ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" xmlns:wsu="http://docs.oasis
   open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
   wsu:Id="mySigCertCert">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="cid:encsign-headers-example">
      <Transforms>
       <Transform Algorithm="http://docs.oasis-open.org/wss/2004/XX/oasis-
2004XX-wss-swa-profile-1.0#Attachment-Complete-Transform"/>
      </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="#mySigCert" />
     </wsse:SecurityTokenReference>
    </KeyInfo>
   </Signature>
  </wsse:Security>
 </soap:Header>
 <soap:Body>
  <Ping xmlns="http://xmlsoap.org/Ping">
   <text>Acme Corp. - Scenario #4</text>
  </Ping>
</soap:Body>
</soap:Envelope>
--enc-sig-headers-example
Content-Type: application/octet-stream
Content-Id: <encsign-headers-example>
Content-Transfer-Encoding: base64

MW4dsa59fdsaSDr5hjdskxhMW4dsa59ffds
```

6.6 Second Message - Response

6.6.1 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?**
Envelope	Mandatory
Body	Mandatory
PingResponse	Mandatory

6.6.2 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.

6.6.2.1 Security

There are no security properties on the response message

6.6.2.2 Body

The body element MUST be not be signed or encrypted

6.6.3 Message Processing

The response is passed to the application without modification.

6.6.4 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> Acme Corp. - Scenario #4</text>
  </PingResponse>
</soap:Body>
</soap:Envelope>
```

6.7 Other processing

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

6.7.1 Requester

No additional processing is required.

6.7.2 Responder

No additional processing is required.

6.8 Expected Security Properties

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

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.

[SwA] W3C Note, "SOAP Messages with Attachments", 11 December 2000, http://www.w3.org/TR/2000/NOTE-SOAP-attachments-2001211.

[WSS-SwA] Hirsch, Frederick, Web Services Security SOAP Message with Attachments Profile 1.0, OASIS Draft 12 2004

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:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd"
targetNamespace="http://xmlsoap.org/Ping" name="Ping">
<types>
 <schema targetNamespace="http://xmlsoap.org/Ping" xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <import namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd" schemaLocation="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wsswssecurity-utility-1.0.xsd"/>

         <element name="text" type="xsd:string" nillable="true"/>
         <complexType name="ping">
                  <sequence>
                        <element ref="tns:text"/>
                  </sequence>
            </complexType>
            <complexType name="pingResponse">
                  <sequence>
                              <element ref="tns:text"/>
                        </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>
                   <mime:multipartRelated>
                     <mime:part>
                      <soap:body use="literal"/>
                    </mime:part>
                    <mime:part>
                      <mime:content type="*/*"/>
                    </mime:part>
                 </mime:multipartRelated>
                  </input>
                  <output>
                        <soap:body use="literal"/>
                  </output>
            </operation>
      </binding>
      <service name="PingService">
            <port name="Ping1" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping1"/>
            </port>
            <port name="Ping2" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping2"/>
            </port>
            <port name="Ping3" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping3"/>
            </port>
            <port name="Ping4" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping4"/>
            </port>
            <port name="Ping5" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping5"/>
            </port>
            <port name="Ping6" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping6"/>
            </port>
            <port name="Ping7" binding="tns:PingBinding">
                  <soap:address location="http://localhost:9080/pingservice/Ping7"/>
            </port>
```

```
    </service>
</definitions>
```

Appendix B. - Revision History

Rev	Date	By Whom	What
01	2004-09-07	Blake Dournaee	Initial version
02	2004-10-18	Blake Dournaee	Fixed problems with examples, specifically the quoting in the MIME headers
03	2004-10-21	Blake Dournaee	Fixed issues with examples. Pushed base64 encoding to MIME layer and removed it as a transform. Added scenario #4.
04	2004-10-22	Blake Dournaee	Fixed more problems with the examples. Clarified the meaning of the shorthand URI notation
05	2004-10-28	Blake Dournaee	Added fully qualified URIs to the examples for the X.509 token attributes ValueType and EncodingType. Added text to disambiguate between the HTTP headers and the SOAP payload.

Appendix C. Notices