Abstract:
This specification defines how to use the OASIS Web Services Security: SOAP Message Security standard [WSS-Sec] with SOAP Messages with Attachments [SwA].

Status:
This is a Committee Draft.
Committee members should submit comments and potential errata to the wss@lists.oasis-open.org list. Others should submit them to the wss-comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to wss-comment-subscribe@lists.oasis-open.org with "subscribe" in the body) or use other OASIS-supported means of submitting comments.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights web page for the WSS TC (http://www.oasis-open.org/committees/wss/ipr.php).
# Table of Contents

1 Introduction.................................................................................................................................................. 3
  1.1 Notations and Terminology.................................................................................................................... 3
    1.1.1 Notational Conventions.................................................................................................................. 3
    1.1.2 Namespaces.................................................................................................................................... 4
    1.1.3 Acronyms and Abbreviations......................................................................................................... 4
  2 MIME Processing........................................................................................................................................ 5
  3 XML Attachments........................................................................................................................................ 6
  4 Securing SOAP With Attachments............................................................................................................. 7
    4.1 Primary SOAP Envelope...................................................................................................................... 7
    4.2 Referencing Attachments.................................................................................................................... 7
    4.3 MIME Part Reference Transforms....................................................................................................... 7
      4.3.1 Attachment-Content-Only Reference Transform............................................................................. 8
      4.3.2 Attachment-Complete Reference Transform................................................................................. 8
    4.4 Integrity and Data Origin Authentication .............................................................................................. 8
      4.4.1 MIME header canonicalization...................................................................................................... 9
      4.4.2 MIME Content Canonicalization.................................................................................................. 10
      4.4.3 Protecting against attachment insertion threat............................................................................. 10
      4.4.4 Processing Rules for Attachment Signing....................................................................................... 11
      4.4.5 Processing Rules for Attachment Signature Verification................................................................. 11
      4.4.6 Example Signed Message............................................................................................................. 12
    4.5 Encryption............................................................................................................................................ 12
      4.5.1 MIME Part CipherReference....................................................................................................... 13
      4.5.2 Encryption Processing Rules....................................................................................................... 13
      4.5.3 Decryption Processing Rules....................................................................................................... 14
      4.5.4 Example...................................................................................................................................... 15
    4.6 Signing and Encryption......................................................................................................................... 16
  5 References.................................................................................................................................................. 17
1 Introduction

This document describes how to use the OASIS Web Services Security: SOAP Message Security standard [WSS-Sec] with SOAP Messages with Attachments [SwA]. More specifically, it describes how a web service consumer can secure SOAP attachments using SOAP Message Security for attachment integrity, confidentiality and origin authentication, and how a receiver may process such a message.

A broad range of industries - automotive, insurance, financial, pharmaceutical, medical, retail, etc - require that their application data be secured from its originator to its ultimate consumer. While some of this data will be XML, quite a lot of it will not be. In order for these industries to deploy web service solutions, they need an interoperable standard for end-to-end security for both their XML data and their non-XML data.

Profiling SwA security may help interoperability between the firms and trading partners using attachments to convey non-XML data that is not necessarily linked to the XML payload. Many industries, such as the insurance industry require free-format document exchange in conjunction with web services messages. This profile of SwA should be of value in these cases.

In addition, some content that could be conveyed as part of the SOAP body may be conveyed as an attachment due to its large size to reduce the impact on message and XML processing, and may be secured as described in this profile.

This profile is applicable to using SOAP Message Security in conjunction with SOAP Messages with Attachments (SwA). This means the scope is limited to SOAP 1.1, the scope of SwA.

The existence of this profile does not preclude using other mechanisms to secure attachments conveyed in conjunction with SOAP messages, including the use of XML security technologies at the application layer or the use of security for the XML Infoset before a serialization that uses attachment technology [MTOM]. The requirements in this profile only apply when securing SwA attachments explicitly according to this profile.

Note that in this document, lists of processing steps are descriptive in that an implementation may use a different procedure as long as the result is the same.

1.1 Notations and Terminology

This section specifies the notations, namespaces, and terminology used in this specification.

1.1.1 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119].

Listings of productions or other normative code appear like this.

Example code listings appear like this.

Note: Non-normative notes and explanations appear like this.

When describing abstract data models, this specification uses the notational convention used by the XML Infoset. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas [XML-Schema], this specification uses the notational convention of OASIS Web Services Security: SOAP Message Security. Specifically, each member of an element’s [children] or [attributes] property is described using an XPath-like [XPath] notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).
Commonly used security terms are defined in the Internet Security Glossary [SECGLO]. Readers are presumed to be familiar with the terms in this glossary as well as the definitions in the SOAP Message Security specification [WSS-Sec].

1.1.2 Namespaces

Namespace URIs (of the general form "some-URI") represent application-dependent or context-dependent URIs as defined in RFC 2396 [URI]. This specification is designed to work with the SOAP 1.1 message structure and message processing model, the version of SOAP supported by SOAP Messages with Attachments. The current SOAP 1.1 namespace URI is used herein to provide detailed examples.

The namespaces used in this document are shown in the following table (note that for brevity, the examples use the prefixes listed below but do not include the URIs – those listed below are assumed).

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>S11</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a></td>
</tr>
<tr>
<td>wsse</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd</a></td>
</tr>
<tr>
<td>wsu</td>
<td><a href="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd</a></td>
</tr>
<tr>
<td>wsswa</td>
<td><a href="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0.xsd">http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-swa-profile-1.0.xsd</a></td>
</tr>
</tbody>
</table>

The URLs provided for the wsse and wsu namespaces can be used to obtain the schema files.

Note: When this document is finalized the wsswa URL will be updated, replacing XX values and possibly making other changes.

1.1.3 Acronyms and Abbreviations

The following (non-normative) table defines acronyms and abbreviations for this document, beyond those defined in the SOAP Message Security standard.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CID</td>
<td>Content ID scheme for URLs. Refers to Multipart MIME body part, that includes both MIME headers and content for that part. [RFC2392]</td>
</tr>
<tr>
<td>SwA</td>
<td>SOAP Messages with Attachments [SwA]</td>
</tr>
</tbody>
</table>
2 MIME Processing

This profile is concerned with the securing SOAP messages with attachments, attachments that are conveyed as MIME parts in a multi-part MIME message as outlined in SOAP Messages with Attachments. In effect this combines two processing layers, a SOAP messaging layer and a MIME wrapping. A SOAP sender effectively transmits a SOAP message and corresponding attachments by passing them to a MIME layer that serializes them. A SOAP receiver receives a message and attachments after the MIME layer processes the MIME serialization. This is important since certain aspects of the MIME processing may be changed at different intermediary transport nodes, yet remain transparent to the SOAP layer. For example, a MIME processing node may change the transfer encoding of a MIME part, transparently to the SOAP nodes. The MIME layer may translate to and from a transfer encoding upon serialization and de-serialization.

The importance to this profile is two-fold. First, it means that certain aspects of MIME processing, such as transfer encoding processing, are out of scope of the profile and do not need to be addressed. Secondly, it means that many of the MIME headers are also out of scope of the profile and the profile does not support integrity protection of these headers, since they are expected to change. If more security protection is required then it must occur at a protocol layer below the MIME layer, for example transport security (with the understanding that such security may not always apply end-end).

SOAP message security is intended to provide security at the SOAP messaging layer, including support for SOAP intermediaries. Thus this profile supports securing the attachment content, possibly including MIME headers that are associated directly with the content (such as Content-Type, Content-Length and other Content related MIME headers) and not MIME headers associated with MIME serialization. This simplifies the profile and also delineates the layering.
3 XML Attachments

A SOAP Messages with Attachments multi-part MIME structure contains a primary SOAP envelope in the root part and one or more attachments in additional MIME parts. Some of these attachments may be have a content type corresponding to XML, but do not contain the primary SOAP envelope to be processed.

Attachments associated with the SOAP body are targeted at the SOAP Ultimate Receiver along with the SOAP body, and may be processed at the application layer along with the body. This means that XML processing may not be required for such XML media type MIME attachments until application layer processing is performed. For this reason the SOAP message layer may not need to perform XML canonicalization or parsing for such attachments and SOAP Message layer security may treat these attachments as text.

Attachments might also be associated with SOAP headers and targeted toward specific SOAP intermediaries, or actors. For SOAP headers specific to an application the attachment content is processed at the application layer, logically after SOAP message processing is complete.

This profile assumes that SOAP attachments (not including the root part containing the primary SOAP envelope) need not be processed as XML at the SOAP messaging layer, so do not require SOAP canonicalization or XML parsing and may be treated as opaque data by the SOAP Message Security layer security processing. MIME part canonicalization (as described below) is required to enable effective SOAP Message Security signatures that include SOAP with Attachments.
4 Secure SOAP With Attachments

Attachments may be associated with SOAP messages, as outlined in SOAP Messages with Attachments (SwA). This profile defines how such attachments may be secured for integrity and confidentiality using the OASIS Web Services Security: SOAP Message Security standard. This does not preclude using other techniques. The requirements in this profile only apply when securing SwA attachments explicitly according to this profile.

This profile considers all attachments as opaque whether they are XML or some other content type. It is the sole responsibility of the application to perform further interpretation of attachments that happen to be XML, including the ability to sign or encrypt portions of those attachments.

4.1 Primary SOAP Envelope

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. This document assumes that a proper SOAP message package is constructed using the HTTP and MIME headers appropriate to [SwA].

The primary SOAP envelope SHOULD be conveyed in the first MIME part, but MAY be conveyed in another MIME part when the start attribute is specified in the HTTP Multipart/Related header.

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=xy1; type="text/xml"; start="<foo>"
```

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

```
--xy1
Content-Type: text/xml; charset=utf-8
Content-ID: <foo>
<?xml version='1.0' ?>
<s11:Envelope xmlns:s11="http://schemas.xmlsoap.org/soap/envelope/" />
```

4.2 Referencing Attachments

SOAP Messages with Attachments defines two MIME mechanisms for referencing attachments. The first mechanism uses a CID scheme URL to refer to the attachment that has a Content-ID MIME header with a value corresponding to the URL, as defined in [RFC 2392]. For example, a content id of “foo” may be specified in the MIME part when the start attribute is specified in the HTTP layer and not the main SOAP payload.

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

The second mechanism is to use a URL to refer to an attachment containing a Content-Location MIME header. In this case the URL may require resolution to determine the referenced attachment [RFC2557]. For simplicity and interoperability this profile limits WS-Security references to attachments to CID scheme URLs. Attachments referenced from WS-Security signature references or cipher references MUST be referenced using CID scheme URLs.

4.3 MIME Part Reference Transforms

By definition of RFC 2392, a URI reference to a MIME attachment includes the MIME headers associated
with that attachment as well as the MIME part content [RFC2392]. Since there may be some confusion as
to what is referenced, it is useful to clearly indicate what is included in the referenced attachment. In
addition, some applications may wish to only encrypt or include the attachment content in a signature
reference hash, and others may wish to include MIME headers and content.

For these reasons, this profile defines two transforms, allowing a clear and explicit statement of what is
included in a MIME reference. These transforms are called “MIME Part Reference Transforms”.

4.3.1 Attachment-Content-Only Reference Transform

The Attachment-Content-Only transform indicates that only the content of a MIME part is referenced. This
transform MUST be identified using the URI value: http://docs.oasis-open.org/wss/2004/XX/oasis-
2004XX-wss-swa-profile-1.0#Attachment-Content-Only-Transform.

Note: When this document is finalized this URL will be updated, replacing XX
values and possibly making other changes.

When this transform is used the content of the MIME part should be MIME canonicalized as defined in
section 4.4.2.

4.3.2 Attachment-Complete Reference Transform

The Attachment-Complete transform indicates that both the content and selected headers of the MIME
part are referenced. This transform MUST be identified using the URI value: http://docs.oasis-

Note: When this document is finalized this URL will be updated, replacing XX
values and possibly making other changes.

This transform specifies that in addition to the content the following MIME headers are to be included
(when present):

• Content-Description
• Content-Disposition
• Content-ID
• Content-Location
• Content-Type

These headers are included because of their common use and the risks associated with inappropriate
modification. If other headers are to be protected, other mechanisms at the application level should be
used (such as copying values into a SOAP header) and this is out of scope of this profile.

Other MIME headers associated with the MIME part serialization are not referenced by the transform and
are not to be included in signature or encryption calculations.

When this transform is used the MIME headers should be canonicalized as defined in section 4.4.1 and
the MIME content should be MIME canonicalized as defined in section 4.4.2.

4.4 Integrity and Data Origin Authentication

Integrity and data origin authentication may be provided for SwA attachments using XML Digital
Signatures, as outlined in the SOAP Message Security standard as profiled in this document. This is
useful independent of the content of the MIME part – for example, it is possible to sign a MIME part that
already contains a signed object created by an application. It may still be sensible to sign such an
attachment as part of SOAP Message security so that the receiving SOAP node may verify that all
attachments are intact before delivering them to an application. A SOAP intermediary may also choose to
perform this verification, even if the attachments are not otherwise processed by the intermediary.
### 4.4.1 MIME header canonicalization

The result of MIME header canonicalization is a UTF-8 encoded octet stream.

Each of the MIME headers listed for the Attachment-Complete transform MUST be canonicalized as part of that transform processing, as outlined in this section. This means the transform MUST perform the following actions in interpreting the MIME headers for signature creation or verification (this order is not prescriptive as long as the same result is obtained)

1. The transform MUST process MIME headers before the MIME content.

2. The transform MUST only process MIME headers that are explicitly present in the attachment part and are listed in the Attachment-Complete transform section of this specification, except that a MIME part without a Content-Type header MUST be treated as having a Content-Type header with the value "Content-Type: text/plain; charset=us-ascii". MIME headers not listed in the Attachment-Complete transform section of this specification are to be ignored by the transform.

3. The MIME headers MUST be processed by the Attachment-Complete transform in lexicographic order (ascending).

4. The MIME header names MUST be processed by the transform as having the case according to the MIME specifications (as shown in the Attachment-Complete section).

5. The MIME header values MUST be unfolded [RFC2822].

6. Any Content-Description MIME header containing RFC2047 encoding MUST be decoded [RFC2047].

7. When a Content-ID header is processed, the "<>" characters associated with the msg-id MUST be included in the transform input. The reason is that although semantically these angle bracket characters are not part of the msg-id (RFC 2822) they are a standard part of the header lexicographic representation. If these characters are not integrity protected then an attacker could remove them causing the CID transformation specified in RFC2392 to fail.

8. Folding whitespace in structured MIME headers (e.g. Content-Disposition, Content-ID, Content-Location, Content-Type) that is not within quotes MUST be removed. Folding whitespace in structured MIME headers that is within quotes MUST be preserved. Folding whitespace in unstructured MIME headers (e.g. Content-Description) MUST be preserved [RFC2822]. For example, whitespace immediately following the colon delimiter in the structured Content-Type header MUST be removed, but whitespace immediately following the colon delimiter in the unstructured Content-Description header MUST be preserved.

9. Comments in MIME header values MUST be removed [RFC2822].

10. Case-insensitive MIME header values (e.g. media type_subtype values and disposition-type values) MUST be converted to lowercase. Case-sensitive MIME header values MUST be left as is with respect to case [RFC2045].

11. Quoted characters other than double-quote and backslash (\") in quoted strings in structured MIME headers (e.g. Content-Id) MUST be unquoted. Double-quote and backslash (\") characters in quoted strings in structured MIME headers MUST be character encoded [RFC2822].

12. Canonicalization of a MIME header MUST generate a UTF-8 encoded octet stream containing the following: the MIME header name, a colon (""), the MIME header value, and the result of canonicalizing the MIME header parameters in lexicographic order (ascending) as described below.

13. MIME header parameter names MUST be converted to lowercase [RFC2045].

14. MIME parameter values containing RFC2184 character set, language, and continuations MUST be decoded. The resulting canonical output MUST not contain the RFC2184 encoding [RFC2184].

15. Case-insensitive MIME header parameter values MUST be converted to lowercase. Case-sensitive MIME header parameter values MUST be left as is with respect to case [RFC2045].
16. Enclosing double-quotes MUST be added to MIME header parameter values that do not already contain enclosing quotes. Quoted characters other than double-quote and backslash (\") in MIME header parameter values MUST be unquoted. Double-quote and backslash characters in MIME parameter values MUST be character encoded.

17. Canonicalization of a MIME header parameter MUST generate a UTF-8 encoded octet stream containing the following: a semi-colon (";"), the parameter name (lowercase), an equals sign ("="), and the double-quoted parameter value.

18. Each header MUST be terminated by a single CRLF pair, without any trailing whitespace.

19. The last header MUST be followed by a single CRLF and then the MIME content.

### 4.4.2 MIME Content Canonicalization

Before including attachment content in a signature reference hash calculation, that MIME attachment may need to be MIME canonicalized. The exact details of MIME part canonicalization depend on the Content-Type of the MIME part. To quote the S/MIME specification (section 3.1.1 “Canonicalization”) which deals with this issue [RFC2633]:

The exact details of canonicalization depend on the actual MIME type and subtype of an entity, and are not described here. Instead, the standard for the particular MIME type should be consulted. For example, canonicalization of type text/plain is different from canonicalization of audio/basic. Other than text types, most types have only one representation regardless of computing platform or environment which can be considered their canonical representation.

MIME types are registered. This registration includes a section on "Canonicalization and Format Requirements" [RFC2048] and requires each MIME type to have a canonical representation.

The MIME "text" type canonical form is defined in the MIME conformance specification (See "Canonical Encoding Model") [RFC2049]. Important aspects of "text" media type canonicalization include line ending normalization to <CR><LF> and ensuring that the charset is a registered charset (see RFC 2633 section “Canonicalization”). [RFC2633, CHARSETS, RFC2045].

MIME attachment parts (other than the part containing the primary SOAP envelope) that contain XML do NOT require XML Canonicalization according to this profile, given the rationale in the previous section on XML attachments. These parts MUST be MIME canonicalized according the MIME "text" part requirements. MIME part canonicalization must be performed before signature hash generation or verification is performed. Signature validation requires an identical hash of content requiring consistent MIME part content.

### 4.4.3 Protecting against attachment insertion threat

Including an attachment in a signature calculation enables a receiver to detect modification of that attachment. Including all attachments in a signature calculation, by providing a <ds:Reference> for each, protects against the threat of attachment removal. This does not protect against insertion of a new attachment.

The simplest protection against attachment insertion is for the receiver to know that all attachments should be included in a signature calculation – unreferenced attachments are then an indication of an attachment insertion attack.

Such information may be communicated in or out of band. Definition of these approaches is out of the scope of this profile.
4.4.4 Processing Rules for Attachment Signing

The processing rule for signing is modified based on the SOAP Message Security rules.

After determining which attachments are to be included as references in a signature, create a 
<ds:Signature> element in a <wsse:Security> header block targeted at the recipient, including a 
<ds:Reference> for each attachment to be protected by the signature. Additional <ds:Reference> 
elements may refer to content in the SOAP envelope to be included in the signature.

For each attachment Reference, perform the following steps:

1. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part.
2. If MIME headers are to be included in the signature, MIME part canonicalize the headers listed in this 
profile as outlined above.
3. Determine the CID scheme URL to be used to reference the part and set the <ds:Reference> URL 
attribute value to this URL.
4. Include a <ds:Transforms> element in the <ds:Reference>. This <ds:Transforms> element MUST 
include a <ds:Transform> element with the Algorithm attribute having the URL value specified in this 
profile - either Attachment-Complete or Attachment-Content-Only, depending on what is to be 
included in the hash calculation. This MUST be the first transform listed. The <ds:Transform> element 
MUST NOT contain any transform for a transfer encoding purpose (e.g. base64 encoding) since 
transfer encoding is left to the MIME layer as noted in section 2.
5. Extract the appropriate portion of the MIME part consistent with the selected transform.
6. Create the <ds:Reference> hash value as outlined in the W3C XML Digital Signature 
Recommendation.

4.4.5 Processing Rules for Attachment Signature Verification

Signature verification is performed as outlined in SOAP Message Security and the XML Digital Signature 
Recommendation, with the following considerations for SwA attachments.

To verify <ds:Reference> hashes for SwA attachments, the following steps must be performed for each 
reference to an attachment:

1. Find the attachment corresponding to the <ds:Reference> URL attribute value. This value MUST 
correspond to the Content-ID for the attachment[SwA].
2. MIME Part Canonicalize the content of the attachment, as appropriate to the MIME type of the part. 
The MIME content to be MIME canonicalized MUST have had any transfer-encoding processed at the 
MIME layer before this step is performed.
3. If MIME headers were included in the signature, canonicalize the headers listed in this profile as 
outlined above.
4. Extract the appropriate portion of the MIME part according to the MIME Part Signature Transform 
value.
5. Calculate the reference hash and verify the reference.
4.4.6 Example Signed Message

```xml
<Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"
--BoundaryStr
<Content-Type: text/xml
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="..." xmlns:xenc="...">
  <S11:Header>
    <wsse:Security>
      <wsse:BinarySecurityToken wsu:Id="CertAssociatedWithSigningKey"
       EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
       ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#x509v3">
        ...
      </wsse:BinarySecurityToken>

      <ds:Signature>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference URI="cid:bar">
          <ds:Transforms>
          </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <ds:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk</ds:DigestValue>
        </ds:Reference>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#CertAssociatedWithSigningKey"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body>
    some items
  </S11:Body>
</S11:Envelope>
--BoundaryStr
<Content-Type: image/png
<Content-ID: <bar>
<Content-Transfer-Encoding: base64
the image
```

4.5 Encryption

A SwA attachment may be encrypted for confidentiality protection, protecting either the MIME part content including selected MIME headers, or only the MIME part content.

This is done using XML Encryption to encrypt the attachment, placing the resulting cipher text in the
updated attachment body replacing the original content, and placing a new <xenc:EncryptedData>
 element in the <wsse:Security> header. An <xenc:CipherReference> MUST link the
 <xenc:EncryptedData> element with the cipher data.

The key used for encryption MAY be conveyed using an <xenc:EncryptedKey> element in the
<wsse:Security> header. In this case the <xenc:ReferenceList> element in the <xenc:EncryptedKey>
element MUST contain an <xenc:DataReference> with a URI attribute specifying the
<xenc:EncryptedData> element in the <wsse:Security> header corresponding to the attachment.

When the same <xenc:EncryptedKey> corresponds to multiple <xenc:EncryptedData> elements, the
<xenc:ReferenceList> in the <xenc:EncryptedKey> element SHOULD contain an <xenc:DataReference>
for each <xenc:EncryptedData> element, both for attachments and encrypted items in the primary SOAP
envelope. References should be ordered to correspond to ordering of the security header elements.

When an <xenc:EncryptedKey> element is not used when encrypting an attachment, then the
<xenc:EncryptedData> element MAY contain a <ds:KeyInfo> element to specify a key as outlined in the
SOAP Message Security standard. Different deployments may have different requirements on how keys
are referenced. When an <xenc:EncryptedKey> element is used the <xenc:EncryptedData> element
MUST NOT contain a <ds:KeyInfo> element.

When an attachment is encrypted, no <xenc:ReferenceList> element is placed as a direct child of the
<wsse:Security> header, since the <xenc:EncryptedData> element is present in the header, eliminating
the need for this reference. Although the SOAP Message Security standard recommends the use of
<xenc:ReferenceList>, this is only necessary when the <xenc:EncryptedData> element is not present in
the <wsse:Security> header. (As mentioned, when the key is conveyed in an <xenc:EncryptedKey>
element, then this element will have a ReferenceList Reference to the <xenc:EncryptedData> element).

Note: The same CID is used to refer to the attachment before encryption and after.
This avoids the need to rewrite references to the attachment, avoiding issues
related to generating unique CIDs and relating to preserving the correspondence to
the original WSDL definition.

4.5.1 MIME Part CipherReference

This profile requires that <xenc:EncryptedData> elements corresponding to encrypted SwA attachments
use a <xenc:CipherReference> to refer to the cipher text, to be conveyed in the attachment. Upon
encryption the MIME part attachment content is replaced with the encoded cipher text.

The <xenc:CipherReference> MUST have a <xenc:Transforms> child element. This element MUST have
a <ds:Transform> child having an Algorithm attribute with a URI value specifying the Content-Only MIME
Part Reference Transform. This transform explicitly indicates that when dereferencing the MIME part
reference that only the MIME part content is to be used as the cipher value.

The <xenc:CipherReference> MUST NOT contain a transform used for a transfer encoding purpose (e.g.
the base64 transform). Transfer encoding is left to the MIME layer, as noted in section 2.

4.5.2 Encryption Processing Rules

The order of the following steps is not normative, although the result should be the same as if this order
were followed.

1. When encrypting both attachments and primary SOAP envelope content using the same key, perform
the attachment processing first.

Note: The SOAP Message Security standard states that elements should be
prepended to the security header. This processing rule supports putting the
<xenc:EncryptedData> element first in the header with <xenc:EncryptedKey> and
tokens following. Thus, a receiver should be able to process the
2. Encrypt the attachment part using XML Encryption, according to the rules of XML Encryption. Encrypt either the attachment including content and selected MIME headers or only the attachment content.

3. Set the `<xenc:EncryptedData>` Type attribute value to a URI that specifies adherence to his profile and that specifies what was encrypted (MIME content or entire MIME part including headers). The following URIs MUST be used for this purpose:

   Note: When this document is finalized these URLs will be updated, replacing XX values and possibly making other changes. Note that these URLs should match the related transforms apart from `-Transform`.

4. Set the `<xenc:EncryptedData>` MimeType attribute to match the attachment MIME part Content-Type header before encryption when Content-Only URI is specified for the Type attribute value. The MimeType attribute value may also be set when the AttachmentComplete Type attribute value is specified.

5. Optionally set the `<xenc:EncryptedData>` Encoding attribute to reflect the attachment content encoding, as visible to the security layer at the time of encryption. This is advisory information to the decryption security layer. It should be understood that this has no relation with the actual encoding that could be performed independently by the MIME layer later for transfer purposes.

6. Set the `<xenc:EncryptedData>` `<xenc:CipherReference>` to the same reference URL for the attachment that was used before encryption. This MUST be a CID scheme URL referring to the attachment part Content-ID. Ensure this MIME header is in the part conveying the cipher data after encryption.

7. Include the Content-Only MIME Part Reference Transform in the `<xenc:CipherReference>` `<xenc:Transforms>` list.

8. Prepend the `<xenc:EncryptedData>` element to the `<wsse:Security>` SOAP header block. Do NOT add a `<xenc:ReferenceList>` element to the SOAP header block (even though recommended by SOAP Message Security).

9. Update the attachment MIME part, replacing the original content with the cipher text generated by the XML Encryption step.

10. Update the attachment MIME part header MIME Content-Type and Content-Length appropriate to the cipher data.

4.5.3 Decryption Processing Rules

The `<xenc:CipherReference>` URL MUST be a URL that refers to the MIME part containing the cipher text, and must also correspond to the reference value of the original attachment that was encrypted. This MUST be a CID scheme URL.

Decryption may be initiated upon locating the `<xenc:EncryptedData>` element in the `<wsse:Security>` header.

The following decryption steps must be performed so that the result is as if they were performed in this order:

1. Extract the cipher text from the attachment referenced by the `<xenc:CipherReference>` URL attribute.
The MIME Part CipherReference Transform defined in this profile indicates that the MIME part content is extracted.

2. Decrypt the cipher text using the information present in the appropriate <xenc:EncryptedData> element and possibly other out of band information, according to the XML Encryption Standard.

3. If the <xenc:EncryptedData>Type attribute indicates that selected MIME headers were encrypted, then those MIME headers MUST be replaced by the result of decryption, as well as the MIME part content.

4. If the <xenc:EncryptedData>Type attribute indicates that only the content of the MIME part was encrypted, then the cipher text content of the attachment part MUST be replaced by the result of decryption. In this case the MIME part Content-Type header value MUST be replaced by the <xenc:EncryptedData>MimeType attribute value.

5. If the <xenc:EncryptedData> Encoding attribute is present then the decryption security layer may pass this advisory information to the application.

4.5.4 Example

This example shows encryption of the primary SOAP envelope body as well as an attachment using a single symmetric key conveyed using an EncryptedKey element.

```xml
Content-Type: multipart/related; boundary="BoundaryStr" type="text/xml"
--BoundaryStr
Content-Type: text/xml

<S11:Envelope
 xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-secext-1.0.xsd"
 xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
 xmlns:id="http://www.w3.org/2000/09/xmldsig#">

<S11:Header>
 <wsse:Security>
  <wsse:BinarySecurityToken wsu:Id="Acert"
   EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
   ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">
   ...
  </wsse:BinarySecurityToken>

  <xenc:EncryptedKey Id='EK'>
   <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
   <ds:KeyInfo Id="keyinfo">
     <wsse:SecurityTokenReference>
       <ds:X509Data>
         <ds:X509IssuerSerial>
           <ds:X509IssuerName>
             DC=ACMECorp, DC=com
           </ds:X509IssuerName>
           <ds:X509SerialNumber>12345678</ds:X509SerialNumber>
         </ds:X509IssuerSerial>
       </ds:X509Data>
       <wsse:SecurityTokenReference>
         <ds:X509IssuerSerial>
           <ds:X509IssuerName>
             DC=ACMECorp, DC=com
           </ds:X509IssuerName>
           <ds:X509SerialNumber>12345678</ds:X509SerialNumber>
         </ds:X509IssuerSerial>
       </ds:X509Data>
     </wsse:SecurityTokenReference>
   </ds:KeyInfo>
   <CipherData>CipherValue>xyzabc</CipherValue></CipherData>
 </ReferenceData>
</ReferenceList>
```
When portions of content are both signed and encrypted, there is possible confusion as to whether encrypted content need first be decrypted before signature verification. This confusion can occur when the order of operations is not clear [DecryptT]. This problem may be avoided with SOAP Message Security for SwA attachments when attachments and corresponding signatures and encryptions are targeted for a single SOAP recipient (actor). The SOAP Message Security standard explicitly states that there may not be two <wsse:Security> headers targeted at the same actor, nor may there be two headers without a designated actor. In this case the SOAP Message Security and SwA profile processing rules may eliminate ambiguity since each signing or encryption produces an element in the <wsse:Security> header, and these elements are ordered. (Signing produces <ds:Signature> elements and encryption produces <xenc:EncryptedData> elements).

If an application produces different <wsse:Security> headers targeted at different recipients, these are processed independently by the recipients. Thus there is no need to correlate activities between distinct headers – the order is inherent in the SOAP node model represented by the distinct actors.
5 References


A. Acknowledgments

The editors would like to acknowledge the contributions of the OASIS WSS Technical Committee, whose voting members at the time of publication were:

- Gene Thurston, AmberPoint
- Frank Siebenlist, Argonne National Laboratory
- Hal Lockhart, BEA Systems, Inc.
- Corinna Witt, BEA Systems, Inc.
- Thomas DeMartini, ContentGuard
- Guillermo Lao, ContentGuard
- Merlin Hughes, Cybertrust
- Sam Wei, Documentum
- Tim Moses, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems, Inc.
- Toshihiro Nishimura, Fujitsu
- Kefeng Chen, GeoTrust
- Irving Reid, Hewlett-Packard
- Kojiro Nakayama, Hitachi
- Paula Austel, IBM
- Derek Fu, IBM
- Maryann Hondo, IBM
- Kelvin Lawrence, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nataraj Nagaratnam, IBM
- Ron Williams, IBM
- Ramanathan Krishnamurthy, IONA
- Kate Cherry, Lockheed Martin
- Paul Cotton, Microsoft Corporation
- Vijay Gajjala, Microsoft Corporation
- Martin Gudgin, Microsoft Corporation
- Chris Kaler, Microsoft Corporation
- Rich Levinson, Netegrity, Inc.
- Jeff Hodges, Neustar, Inc.
• Frederick Hirsch, Nokia
• Abbie Barbir, Nortel Networks
• Lloyd Burch, Novell
• Steve Anderson, OpenNetwork
• Vamsi Motukuru, Oracle
• Ramana Turlapati, Oracle
• Chong-Jen Hsu, PeopleSoft
• Prateek Mishra, Principal Identity
• Ben Hammond, RSA Security
• Rob Philpott, RSA Security
• Martijn de Boer, SAP
• Blake Dournaee, Sarvega
• Coumara Radja, Sarvega
• Pete Wenzel, SeeBeyond Technology Corporation
• Ronald Monzillo, Sun Microsystems
• Jan Alexander, Systinet
• Symon Chang, Tibco
• John Weiland, US Dept of the Navy
• Phillip Hallam-Baker, Verisign
• Maneesh Sahu, Westbridge Technology
### B. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>05/25/04</td>
<td>Frederick Hirsch</td>
<td>Initial version, put draft proposal into profile format.</td>
</tr>
<tr>
<td>2</td>
<td>05/26/04</td>
<td>Frederick Hirsch</td>
<td>Editorial and namespace suggestions from Michael McIntosh. Added rationale for SwA support to introduction. Completely rewrote processing rules for encryption and decryption.</td>
</tr>
<tr>
<td>3</td>
<td>05/28/04</td>
<td>Frederick Hirsch</td>
<td>Rewrote signature section, fixed cid references and Content-IDs, added examples.</td>
</tr>
<tr>
<td>4</td>
<td>06/12/04</td>
<td>Frederick Hirsch</td>
<td>Added Decrypt Transform section, added All-Attachments-Complete transform, changed MIME reference to v3, minor editorial changes.</td>
</tr>
<tr>
<td>5</td>
<td>07/07/04</td>
<td>Frederick Hirsch</td>
<td>Removed Decrypt transform material, since it is generally not needed and the approach had issues. Reorganized signatures section. Eliminated incorrect All-Attachments-Complete transform and replaced with discussion of attachment insertion threat. Clarified that only one wsse:Security header per actor/role minimizes signing, encryption confusion possibility. Added section for MIME Part CipherReference Transform. Editorial fixes.</td>
</tr>
<tr>
<td>6</td>
<td>07/14/04</td>
<td>Frederick Hirsch</td>
<td>** Allow use of Content-Location, consistent with SwA. ** Proposed update to signature Content-Transfer-Encoding processing rules. Needs review. Revised section on MIME canonicalization, added section on XML attachments. Only support SOAP 1.1. Clarified introduction. Added MTOM and additional MIME references. (Issue 297 should be closed – removed section on decryption transform and updated section on signing and encryption in version 5) Issue 303 – fixed, (see 3.2.4 example), Issue 306 – revised section on MIME canonicalization to close this issue. Issue 307 – revised to refer to SOAP 1.1 only, added section on XML attachments, defined MTOM and added reference. Editorial fixes.</td>
</tr>
<tr>
<td>Rev</td>
<td>Date</td>
<td>By Whom</td>
<td>What</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>07/30/04</td>
<td>Frederick Hirsch</td>
<td>Incorporate feedback from WS-I BSP. Limit MIME headers included in signature or encryption to those listed in profile. Clarify MIME layering approach. Remove processing rules associated with Content-Transfer-Encoding. Editorial correction throughout document to allow both CID and Content-Location references to attachments. Editorial revision to pull attachment referencing and reference transforms into section applicable to both signatures and encryption. Incorporated feedback from Pete Wenzel and Toshihiro Nishimura – separate URL for transform and encryption type, used Content-Only reference transform for Cipherdata as well.</td>
</tr>
<tr>
<td>8</td>
<td>08/23/04</td>
<td>Frederick Hirsch</td>
<td>Address issue 312 by clarifying use of Reference within EncryptedData element to EncryptedData for attachment when EncryptedKey is used. Processing rule related to encryption of both attachment and primary SOAP envelope items. (<a href="http://www.oasis-open.org/archives/wss/200408/msg00046.html">http://www.oasis-open.org/archives/wss/200408/msg00046.html</a>) Changed encryption example to show encryption of both primary SOAP envelop body and attachment. Include EncryptionMethod, addressing issue 309. Fix Transforms namespace to be xenc for within xenc:CipherReference (<a href="http://www.oasis-open.org/archives/wss/200408/msg00048.html">http://www.oasis-open.org/archives/wss/200408/msg00048.html</a>)</td>
</tr>
<tr>
<td>9</td>
<td>09/02/04</td>
<td>Frederick Hirsch</td>
<td>Clarify that XML attachments are opaque and remove text about XML canonicalization of attachment content. Fix typo at line 356, should state that no KeyInfo should be in EncryptedData element when EncryptedKey is used. Clarify that cipher data is base64 encoded octet stream and require CipherReference base64 transform. Revise MIME headers to be included in Attachment-Complete Reference, for signature protection. Allow continuations for these MIME headers.</td>
</tr>
<tr>
<td>Rev</td>
<td>Date</td>
<td>By Whom</td>
<td>What</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>10/02/04</td>
<td>Frederick Hirsch</td>
<td>Proposed resolutions for WSS issue-list items:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 326 part 2 - : Clarify MIME header name case, Resolution to use case per MIME specifications. See 4.3.1 item 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 326 part 3 -  Clarify transform handling of MIME parameter quoting. Retain quoting, if any, as is. Resolution in 4.3.1 item 7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 326 part 4 - Address RFC 2047 encoding. Require transform to perform RFC2047 decoding as needed. Resolution in 4.3.1, items 4-7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 329 part 1 – Strip or compress white space. No change made apart apart from preserve all whitespace in quoted strings, 4.3.1. item 10.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 329 part 2 – Order header processing alphabetically. Resolution in 4.3.1 item 3 and 4.2.2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue 329 part 3 – Show all ds:Signature elements in example in 4.3.6.</td>
</tr>
<tr>
<td>11</td>
<td>10/02/04</td>
<td>Frederick Hirsch</td>
<td>Issue 326, 329 – revision of section 4.3.1 based on feedback from Dana Kaufman and Forum Systems.</td>
</tr>
<tr>
<td>12</td>
<td>10/21/04</td>
<td>Frederick Hirsch</td>
<td>Allow cipher data to be binary data, and not use base64 transform in this case. Clarify that for base64 encoded cipher data transform or other means should be used to convey this information. Updated 4.4.1 through 4.4.4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Quoted “text/xml” in examples in 4.3.6, 4.4.4 to resolve issue 325.</td>
</tr>
<tr>
<td>13</td>
<td>10/29/04</td>
<td>Frederick Hirsch</td>
<td>Replace “7-bit” with “binary” in example 4.4.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Add clarification to sections 4.2.1 and 4.2.2 that MIME canonicalization is to be associated with the transforms, as defined in 4.3.1 and 4.3.2.</td>
</tr>
<tr>
<td>14</td>
<td>11/15/04</td>
<td>Frederick Hirsch</td>
<td>1. Only allow CID references for WS-Security references, for simplicity and interoperability.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Constrain statement on RFC2047 encoding in section 4.4.1, #6.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Clarify use of &lt;xenc:EncryptedData&gt;MimeType attribute in 4.5.2, #4. (Issue 345, #1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. Add statement from interop document regarding MIME boundary for primary SOAP envelope</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. Editorial changes to make MAY/MUST/SHOULDs capitalized where possible, other editorial fixes.</td>
</tr>
<tr>
<td>Rev</td>
<td>Date</td>
<td>By Whom</td>
<td>What</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>15</td>
<td>12/06/04</td>
<td>Frederick Hirsch</td>
<td>Explicitly allow optional use of Encryption Encoding attribute. (Section 4.5.2 #5, Issue 341)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remove base64 transform material, clarify relationship to MIME layer transform encoding. (Sections 4.4.4, 4.4.5, 4.5.1, 4.5.2, 4.5.3; Issue 344)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Add clarification that Content-ID header value &lt;&gt; included in Attachment-Complete transform for signing. (Section 4.4.1, #7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Editorial cleanup. Add KeyInfo to example 4.4.6.</td>
</tr>
<tr>
<td>cd-01</td>
<td>01/07/05</td>
<td>Frederick Hirsch</td>
<td>Change to Committee Draft - 01</td>
</tr>
</tbody>
</table>
C. Notices

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

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

Copyright © OASIS Open 2004. All Rights Reserved.

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

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

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

JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.