



XML Secured Payload Profile

Version 1.0 draft — 1 September 2008

Draft for comment

National E-Health Transition Authority Ltd

Level 25

56 Pitt Street

Sydney, NSW, 2000

Australia.

www.nehta.gov.au

Disclaimer

NEHTA makes the information and other material (“Information”) in this document available in good faith but without any representation or warranty as to its accuracy or completeness. NEHTA cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

Document Control

This document is maintained in electronic form. The current revision of this document is located on the NEHTA Web site and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is of the latest revision.

Copyright © 2008, NEHTA.

This document contains information which is protected by copyright. All Rights Reserved. No part of this work may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without the permission of NEHTA. All copies of this document must include the copyright and other information contained on this page.

Table of contents

Table of contents	iii
Document information	v
Change history	v
1 Introduction	7
1.1 Background	7
1.2 Purpose	7
1.3 Scope	7
1.3.1 Content	7
1.3.2 Related Publications	7
1.3.3 Audience	8
1.4 Status	8
1.5 Normative references	8
1.6 Definitions	9
1.6.1 Acronyms	9
1.6.2 Namespaces	9
1.7 Conformance	9
1.8 Overview	9
2 Profiles	10
2.1 Signed profile	10
2.2 Encrypted profile	10
3 Containers	11
3.1 XML Schema definitions	11
3.1.1 SignedPayload	11
3.1.2 EncryptedPayload	13
4 XML Signature	15
Signature Process	15
4.1 Standards	15
4.1.1 XML Signature	15
4.1.2 Detached signature	16
4.2 ds:Signature	16
4.3 ds:SignedInfo	16
4.3.1 ds:CanonicalizationMethod	17
4.3.2 ds:SignatureMethod	17
4.3.3 ds:Reference	18
4.3.4 ds:Transforms	19
4.3.5 ds:Transform	19
4.3.6 ds:DigestMethod	20
4.3.7 ds:DigestValue	20
4.4 ds:SignatureValue	20
4.5 ds:KeyInfo	20
4.5.1 ds:KeyInfo (in Signature)	20
4.5.2 ds:X509Data	21
4.6 ds:Object	21
4.6.1 ds:Object	21
5 XML Encryption	22
Encryption Process	22
5.1 Standards	22
5.1.1 XML Encryption	22
5.2 xenc:EncryptedData	23
5.2.1 Encryption granularity	23

5.2.2	MimeType and Encoding.....	24
5.3	xenc:EncryptionMethod	24
5.3.1	Symmetric encryption using AES-256	24
5.4	ds:KeyInfo.....	25
5.4.1	ds:KeyInfo (in EncryptedData).....	25
5.5	xenc:CipherData.....	25
5.5.1	xenc:CipherData (EncryptedData)	25
5.6	xenc:EncryptionProperties	26
5.6.1	xenc:EncryptionProperties	26
5.7	ds:EncryptedKey	26
5.7.2	Asymmetric encryption using RSA 1.5	27
5.7.3	ds:KeyInfo (in xenc:EncryptedKey)	28
5.7.4	ds:X509Data	28
5.7.5	xenc:CipherData (EncryptedKey).....	28
Appendix A: Informative references		30

Document information

Change history

Version	Date	Comments
1.0 draft	2008-09-01	Draft for comment

This page intentionally left blank.

1 Introduction

1.1 Background

The National E-Health Transition Authority (NEHTA) has recommended Web services as the mechanism for application-to-application communications in Australia's e-health environment.

Web services use the Extensible Markup Language (XML) as the format for representing data.

1.2 Purpose

This document defines a set of mechanisms for securing XML formatted data and representing that secured data in XML. These mechanisms are based on XML Signature [XSSP2002] and XML Encryption [XESP2002]. These mechanisms are grouped into profiles.

A profile specifies a set of obligations that must be met. These obligations identify the standards to use and how they are to be applied.

Profiles provide a clear definition of what must be done. Profiles are necessary because often there are many alternative standards that can be used and those standards can be interpreted and used in different ways. Profiles have been defined to enable different implementations to be integrated together to exchange information. These profiles have been developed by taking into account the standards and practical implementations of those standards.

The primary purpose of these profiles is to provide mechanisms for securing payloads used inside Web services SOAP messages. This is necessary when sending SOAP messages through intermediaries and when only the ultimate recipient can read the contents of the SOAP message. In the context of Web services, security only protects the message between endpoints so additional payload security maybe required.

1.3 Scope

1.3.1 Content

The profiles define ways to implement the payload security patterns defined in the *Concepts and Patterns for Implementing Services* [CPIS2008]. Those patterns identify different approaches for satisfying different security requirements on message payloads. Those security requirements involve different levels of confidentiality, integrity and authentication.

The profiles in this document are designed for data represented as XML.

These profiles can also be used independently of SOAP and Web services. For example, for securing XML data that is stored in databases. However, for convenience the data will be referred to as the "payload".

1.3.2 Related Publications

A familiarity with the *Concepts and Patterns for Implementing Services* [CPIS2008] is useful for understanding the motivation behind the profiles in this document.

Examples of how these profiles can be implemented in JAX-WS, WCF and WSE can be found in *Example Technical Implementation of Interoperable Web Services*, [EIJAX2008], [EIWCF2008] and [EIWSE2008].

1.3.3 Audience

This document is intended for:

- Specification authors who create service interface specifications or software specifications. They will use this document by referencing these profiles in their specifications.
- Software developers who create implementations of those specifications. They will use this document by implementing the profiles in their software. The profiles chosen will depend on the specification being implemented. Note developers will usually use existing toolkits or libraries for performing XML Encryption and XML Signature operations. They will not normally implement them from scratch.
- Testers who check an implementation for conformance to specifications. They will use this document as a set of conformance criteria.

The reader is expected to have detailed knowledge of XML, XML Encryption, XML Signature and Public Key Infrastructure (PKI).

1.4 Status

This document is a draft and has been released for comment and feedback purposes.

1.5 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- [EXC2002] W3C, *Exclusive XML Canonicalization*, Version 1.0, W3C Recommendation, 18 July 2002, <http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/>
- [FIPS197] FIPS, *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, 26 November 2001.
- [PKCS1-1993] RSA, *PKCS #1: RSA Encryption Standard*, Version 1.5, RSA Laboratories Technical Note, 1 November 1993.
- [RFC2119] IETF, *RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner, March 1997, <http://ietf.org/rfc/rfc2119.txt>
- [XESP2002] W3C, *XML Encryption Syntax and Processing*, W3C Recommendation, 10 December 2002, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>
- [XEXSD2002] W3C, *XML Encryption Syntax and Processing*, W3C Recommendation XML Schema, 10 December 2002, <http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd>
- [XSSP2002] W3C, *XML Signature Syntax and Processing*, W3C Recommendation, 12 February 2002 <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>
- [XSXSD2002] W3C, *XML Signature Syntax and Processing*, W3C Recommendation XML Schema, 12 February 2002 <http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd>

Non-normative references can be found in Appendix A: "Informative references".

1.6 Definitions

1.6.1 Acronyms

AES	Advanced Encryption Standard
PKI	Public Key Infrastructure
SHA	Secure Hash Algorithm
SKI	Subject Key Identifier
XML	Extensible Markup Language

1.6.2 Namespaces

The prefix "sp" is used to refer to the secured payload namespace, "urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901".

The prefix "ds" is used to refer to the XML Signature namespace, "http://www.w3.org/2000/09/xmldsig#".

The prefix "xenc" is used to refer to the XML Encryption namespace, "http://www.w3.org/2001/04/xmlenc#".

1.7 Conformance

To conform to a particular profile, an artefact **MUST** satisfy every criterion in that profile.

The keywords **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** in this document are to be interpreted as described in IETF's RFC 2119 [RFC2119].

1.8 Overview

Chapter 2 "Profiles" defines the profiles. These profiles make use of the containers defined in Chapter 3.

Chapter 3 "Containers" specifies XML Schema datatype definitions for the representations of the secured payloads. Criteria for these containers make references to the criteria in Chapters 5 and 4.

Chapter 4 "XML Signature" specifies the criteria for using the XML Signature standard.

Chapter 5 "XML Encryption" specifies the criteria for using the XML Encryption standard.

2 Profiles

A profile specifies a set of obligations that must be met. These obligations identify standards to use and how they are to be applied.

This document defines two profiles which correspond to two of the payload security patterns defined in the *Concepts and Patterns for Implementing Services 2.0* [CPIS2008]. These profiles are designed to implement those patterns for XML formatted data.

The two profiles are:

- Signed profile
- Encrypted profile

Note that the *Concepts and Patterns for Implementing Services 2.0* also defines a "None" and "Signed Before Encrypted" pattern. The "None" pattern indicates that no additional security over the web services point-to-point security mechanisms defined in [WSP2008] is required. The "Signed Before Encrypted" pattern indicates the payload must be signed and then encrypted. This pattern can be realised by combining the "Signed profile" and "Encrypted profile".

2.1 Signed profile

The Signed profile requires the payload to be signed.

This profile can be used when authentication of the data creator and/or message integrity are required. In the context of services, this ensures that the ultimate recipient can verify the identity of the entity that created the payload and the data has not been modified in transit.

Conformance to this profile requires conformance to the criteria of:

- `sp:SignedPayload` (section 3.1.1)

2.2 Encrypted profile

The Encrypted profile requires the payload to be encrypted.

This profile can be used when confidentiality is required. In the context of services, this ensures that only the ultimate recipient can read the payload.

Conformance to this profile requires conformance to the criteria of:

- `sp:EncryptedPayload` (section 3.1.2)

3 Containers

This chapter defines XML datatypes which are used as containers for representing the secured data.

The datatypes can be used within a service specification or independently when payload security is required between two entities. There are no restrictions of how many containers can be used. The specifications that use the containers define which container and how many will be used.

3.1 XML Schema definitions

Two container data-types are defined using XML Schema [XSD2004a] [XSD2004b]. These datatypes are:

- `sp:SignedPayload` — for holding a payload and a signature of the payload.
- `sp:EncryptedPayload` — for holding an encrypted payload.

3.1.1 SignedPayload

The `sp:SignedPayload` is a container for representing the content and a digital signature over that content.

3.1.1.1 Obligations

XS 3.1.1.1-1 Implementations **MUST** comply with the `sp:SignedPayload` XML Schema complex type shown in XML Schema Fragment 1.

XML Schema Fragment 1:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901"
  xmlns:tns="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="qualified">

  <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>

  <xsd:complexType name="SignedPayloadData">
    <xsd:sequence>
      <xsd:any processContents="lax" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="SignatureList">
    <xsd:sequence>
      <xsd:element ref="ds:Signature" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="SignedPayload">
    <xsd:sequence>
      <xsd:element name="signatures" type="tns:SignatureList" minOccurs="1"
        maxOccurs="1"/>
      <xsd:element name="signedPayloadData" type="tns:SignedPayloadData"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="signedPayload" type="tns:SignedPayload"/>

</xsd:schema>
```

XS 3.1.1.1-2 The `ds:Signature` element **MUST** comply with all the criteria in chapter 4.

- xs 3.1.1.1-3 The `xsd:any` element **MUST** be the data that is being secured.
- xs 3.1.1.1-4 The `id` attribute **MUST** be set to a value unique in the XML document.
- xs 3.1.1.1-5 The `sp:signedPayloadData` element and payload **MUST** be signed.

3.1.1.2 Notes (non-normative)

The `sp:signatures` element can contain one or more signatures of the `sp:signedPayloadData` element. This allows multiple entities to sign the payload.

The `xsd:any` element within the `sp:SignedPayloadData` type is a placeholder for the payload content. When this container is used with actual content, the definition of that content would replace the `xsd:any` element.

The `id` attribute on the data element is used to hold a generated unique ID that is used by XML Signature as a reference to the data that's been signed. The value of the ID must be unique in the scope of the whole XML document the payload is a part of.

The `sp:signedPayloadData` element, including the `id` attribute and the payload are signed to create the `ds:Signature` element. The `ds:Signature` element is added after the data element within the container.

3.1.1.3 Example (non-normative)

The following shows an example instance of the `sp:SignedPayload` container:

```

<sp:signedPayload
  xmlns:sp="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901">
  <sp:signatures>
    <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="#a0ede6c4-21e4-44d5-93b3-4b2f02f9e1f8">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <DigestValue>Orn5miXZzb9EYhwcMg4ZTWavRds=</DigestValue>
        </Reference>
      </SignedInfo>
      <SignatureValue>ffzZtvV4ms46VbZRLM6r0eMoOirKpwd1kOAVxhLqs/TPgtIjDqgX4HMqmRbjNibAtO
4KVmnLkx6NUq3PbNFYODZCF9xTEw2HTOCSRvRpxlL0DPbVc5JylU0v3e1hbcglCmhqfRyI9003b3se9WpF
/nkDSZGAov2NmhWnbnNcqjE=</SignatureValue>
      <KeyInfo>
        <X509Data>
          <X509Certificate>MIICBjCCAW+gAwIBAgIBBjANBgkqhkiG9w0BAQUFADAeMRwwGgYDVQQDExNORUhuUQ
SBFSUlXUyBkZW1vIENBMB4XDTA3MDYyMjAwMDI1NFoXDTE3MDYxOTAwMDI1NFowFzEVMBMGA1UEAxQMamF
4d3NfY2xpZW50MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDVNtVihAhN76KhBevo/YTzBx1oB7K8Y
qRhfG3ca/Y9qFv1Mk6tUNjC9LxBqtLcQbcLpeZOFsKumtygvvZUBAZlpTR7qMOeHcFMYCPlsVS4mToTR
8P33au2x6VvTj7AWaXr8Diljv13hB/luPKRuflyw0hV7mzvSkclwHVBbn0QIDAQABolswWTAMBgNVHRMBA
f8EAJAAMB0GA1UdDgQWBBS1A5kUs7rHCR8wZEX7Nb4m3k1zOzAfBgNVHSMEGDAWgBQL0DcCBAjrsOSF5Usi
wo0701htHBzAJBgNVHREEjAAMA0GCSqGSIb3DQEBAQUAA4GBAGEPlkuOg5RftVWrfP+PSgHueUugWHhOq
UsUB5w3OPyhkIawVfbrXu4cq+wSo96yIP/89xsxTwmWiWa0LQ02xmEZWf2F9FrcO2Ni9nZllulREtrd5Hu
ino80GmEB4AJWDhGV0GAT45Ze/tuVg+Xa+YmvHuuYseLVGeirnEOmoPS2</X509Certificate>
        </X509Data>
      </KeyInfo>
    </Signature>
  </sp:signatures>
  <sp:signedPayloadData id="a0ede6c4-21e4-44d5-93b3-4b2f02f9e1f8">
    <pathologyreport xmlns="http://www.example.org">
      <name>John Doe</name>
      <tests>
        <test name="test1">1000</test>
      </tests>
    </pathologyreport>
  </sp:signedPayloadData>
</sp:signedPayload>

```

3.1.2 EncryptedPayload

The `sp:EncryptedPayload` is a container for representing content which has been encrypted.

3.1.2.1 Obligations

- XS 3.1.2.1-1 Implementations **MUST** comply with the `sp:EncryptedPayload` XML Schema complex type shown in XML Schema Fragment 2.

XML Schema Fragment 2:

```
<xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901"
  xmlns:tns="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  elementFormDefault="qualified">

  <xsd:import namespace="http://www.w3.org/2001/04/xmlenc#"
    schemaLocation="http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd"/>

  <xsd:complexType name="EncryptedPayloadData">
    <xsd:sequence>
      <xsd:element ref="xenc:EncryptedData" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="KeyList">
    <xsd:sequence>
      <xsd:element ref="xenc:EncryptedKey" minOccurs="1"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="EncryptedPayload">
    <xsd:sequence>
      <xsd:element name="keys" type="tns:KeyList" minOccurs="1"
        maxOccurs="1"/>
      <xsd:element name="encryptedPayloadData" type="tns:EncryptedPayloadData"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="encryptedPayload" type="tns:EncryptedPayload"/>

</xsd:schema>
```

- XS 3.1.2.1-2 The `xenc:EncryptedData` element **MUST** comply with all the criteria in chapter 5.
- XS 3.1.2.1-3 The `xenc:EncryptedKey` element **MUST** comply with all the criteria in chapter 5, section 5.7.
- XS 3.1.2.1-4 The plaintext contents of the `xenc:EncryptedData` element **MUST** be the contents being secured.

3.1.2.2 Notes (non-normative)

The `sp:KeyList` element can hold one or more encrypted keys. An encrypted key can be used to decrypt the data. This allows the message to be decrypted by multiple entities. Each entity that needs to be able to decrypt the payload has its own encrypted key which contains a common session key.

The `sp:encryptedPayloadData` element contains the encrypted data.

3.1.2.3 Example (non-normative)

The following shows an example instance of the `sp:EncryptedPayload` container:

```
<sp:encryptedPayload
  xmlns:sp="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901">
  <sp:keys>
    <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Data>
          <X509SKI>pQOZFLO6xwkmGRF+zW+Jt5Nczs=</X509SKI>
        </X509Data>
      </KeyInfo>
      <CipherData>
        <CipherValue>
uQJemj3rJTgfm4hcPe5EebpoX3O6IBv9uLuOCSP6oKXYrFnPjCjI0A7423fkRC0dqbvMkn3xOtQ94yzhF4H
py2Fs6YIEf3Xf3r6s8I+huDecpEa410ZP4/+uVVL/FEmdZerVf0ayhp0+miRD0rOtFP2peiNjT6G7ggIK5
vkOnNzw=
        </CipherValue>
      </CipherData>
      <ReferenceList>
        <DataReference URI="#_1" />
      </ReferenceList>
    </EncryptedKey>
  </sp:keys>
  <sp:encryptedPayloadData>
    <EncryptedData Id="_1"
      Type="http://www.w3.org/2001/04/xmlenc#Element"
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
      <CipherData>
        <CipherValue>
BQ6byRM5fjzm0IU1AYIHxd5ufsu7/ctJEGxRfo04JfGA4qa3PgTjaBry/9YN8wqblIoGxiNeyGAzenxONY
oky+AVMAdqtsh6QzKx1Ze3Xnj5I8oTHMA8EfL0R/w8ObuHhQxHnfxbf6yNIZik09dS6Z4lpLAVyphEotaU
v+TWE+fdhds/ti3wnxqSSA8EsDIY8d61N5P8A3AlZY73dynwWw8gJu8pWctYQYzz+ezVlNg=
        </CipherValue>
      </CipherData>
    </EncryptedData>
  </sp:encryptedPayloadData>
</sp:encryptedPayload>
```

The following shows an example instance of the `sp:EncryptedPayload` container in its decrypted form:

```
<sp:encryptedPayload
  xmlns:sp="urn:xml-gov-au:nehta:types:SecuredPayload:1.0-draft-20080901">
  <sp:keys>
    ...
  </sp:keys>
  <sp:encryptedPayloadData>
    <pathologyreport xmlns="http://www.example.org">
      <name>John Doe</name>
      <tests>
        <test name="test1">1000</test>
      </tests>
    </pathologyreport>
  </sp:encryptedPayloadData>
</sp:encryptedPayload>
```

4 XML Signature

This chapter contains criteria on using XML Signature.

This chapter is organised into six sections:

- Standards, containing obligations on the standards to use;
- `ds:Signature`;
- `ds:SignedInfo`;
- `ds:SignatureValue`;
- `ds:KeyInfo`.; and
- `ds:Object`.

The last four correspond to the four possible child elements in `ds:Signature`, the root element in XML Signature.

Signature Process

The obligations in this chapter are designed to support the normal practice of signing using PKI.

The creator steps:

1. Sign the plaintext data using their private key.

The consumer steps:

2. Validates the signature using the signer's public certificate. When there are multiple signatures, this step is repeated for each signature that needs to be validated.

4.1 Standards

4.1.1 XML Signature

4.1.1.1 Obligations

XS 4.1.1.1-1 Implementations **MUST** use the *XML-Signature Syntax and Processing Recommendation* from W3C [XSSP2002] and be valid against the XML Signature XML Schema [XSXSD2002].

4.1.1.2 Notes (non-normative)

The *XML-Signature Syntax and Processing Recommendation* from the W3C specifies a process for signing data and representing the result as XML.

The XML Signature specification defines a standard for signing and verifying data. Any data that can be referenced via a URI can be signed. Signatures on data can ensure that the data being sent hasn't been tampered with and can also assert the identity of who signed the data. XML Signature allows multiple references to be signed within the one signature, although the criteria in this document restrict it to only one.

The result of the signing process is a `ds:Signature` XML Element. The `ds:Signature` element has four possible child elements. The criteria for these are described in the following sections:

- `ds:SignedInfo` (see section 4.3)
- `ds:SignatureValue` (see section 4.4)
- `ds:KeyInfo` (see section 4.5)

- `ds:Object` (see section 4.6)

4.1.1.3 Example (non-normative)

The result of using XML Signature produces an instance of the `ds:Signature` element. This is an example of this element and its contents:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <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="#1eb025e2-a40b-4406-87a0-0b4646eb8d90">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <ds:DigestValue>ZoEDHpJkU4+88f+aSEntb1lMlQk=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>ip0f7HK...</ds:SignatureValue>
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509Certificate>
C9A3AgQI6zkheVLIIsKDuzpR7RwcwCQYDVR0RBAlwADANBgkqhkiG9w0BAQUFAAOBgQC9palkOey2uxHn/t
CtC0xOstVedXJ+1+HrQngIz3QK+XDq5jeT7QmDc6UtGntu0Y4cXY35Au/gOeTIZkCC0RW9BPG6MJqUfInf
5K6uCWC4aYGrTcpcn0/Skb0/SvA5VKc9/CQtzoTafSud/CHP8Jc3ZUB1SdHOMgBMAcUt55z8jA==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

4.1.2 Detached signature

4.1.2.1 Obligations

XS 4.1.2.1-1 Implementations **MUST** use the detached signature form of XML Signature.

4.1.2.2 Notes (non-normative)

XML Signature allows the signature to be related to the data being signed in three different ways.

- Enveloping signature, where the data is embedded inside the XML Signature.
- Enveloped signature, where the XML signature is embedded inside the data.
- Detached signature, where the signature is separate from the data.

This obligation specifies the use of a detached signature.

4.2 ds:Signature

There are no additional obligations on the use of the `ds:Signature` element.

4.3 ds:SignedInfo

The `ds:SignedInfo` element contains the following child elements:

- `ds:CanonicalizationMethod`
- `ds:SignatureMethod`
- `ds:Reference`

Obligations for using these are described in this section.

4.3.1 ds:CanonicalizationMethod

4.3.1.1 Obligations

XS 4.3.1.1-1 Implementations **MUST** use the Exclusive XML Canonicalization as specified in [EXC2002] over the `ds:SignedInfo` element and indicate this by setting the `Algorithm` attribute on the `ds:CanonicalizationMethod` element to `"http://www.w3.org/2001/10/xml-exc-c14n#" .`

4.3.1.2 Notes (non-normative)

XML Signature allows different canonicalization algorithms to be applied to the `ds:SignedInfo` element for creating the canonicalized form of the `ds:SignedInfo` element which is signed to create the signature value.

There are two types of canonicalization, Exclusive and Inclusive. These obligations specify that the Exclusive XML Canonicalization algorithm is to be used. Exclusive canonicalization ignores the context in which the XML fragment appears, and only keeps the namespaces which are visibly utilised within the fragment of XML being signed. Inclusive Canonicalization inherits the context in which the XML fragment appears, including the namespaces that are in scope where the fragment appears. Therefore, Exclusive Canonicalization is necessary when the fragment is inserted into a different XML document, such as a container, which could have other namespaces in scope. Inclusive canonicalization cannot be used as it copies all namespace declarations into the fragment of XML being signed. This would include namespaces outside the fragment and would invalidate the signature if the context is different.

The canonicalization method specified on the `ds:SignedInfo` element only applies to `ds:SignedInfo` element and its contents. A canonicalization method is also applied to the data being signed via the `ds:Transform` element, see section 4.3.5 for details.

4.3.1.3 Example (non-normative)

The example below shows the representation of the `ds:CanonicalizationMethod` element and the `Algorithm` attribute with exclusive canonicalization method specified:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    ...
  </ds:SignedInfo>
</ds:Signature>
```

4.3.2 ds:SignatureMethod

4.3.2.1 Obligations

XS 4.3.2.1-1 Implementations **MUST** use the RSA-SHA1 algorithm for generating the signature and indicate this by setting the `Algorithm` attribute on the `ds:SignatureMethod` element to `"http://www.w3.org/2000/09/xmldsig#rsa-sha1" .`

The RSA-SHA1 algorithm is defined in section 6.4.2 of *XML-Signature Syntax and Processing* [XSSP2002].

4.3.2.2 Notes (non-normative)

XML Signature allows different algorithms to be applied to the canonicalized data to calculate the signature value.

These obligations specify that the RSA-SHA1 algorithms are to be used. In future versions of the profile other algorithms could be specified.

4.3.2.3 Example (non-normative)

The example below shows the `ds:SignatureMethod` element and the `Algorithm` attribute with the RSA-SHA1 algorithm specified:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    ...
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    ...
  </ds:SignedInfo>
</ds:Signature>
```

4.3.3 ds:Reference

4.3.3.1 Obligations

- XS 4.3.3.1-1 There **MUST** only be one `ds:Reference` element in `ds:SignedInfo` element.
- XS 4.3.3.1-2 The URI attribute on the `ds:Reference` element **MUST** be present.
- XS 4.3.3.1-3 The URI attribute value **MUST** be a fragment identifier of the element being signed.

4.3.3.2 Notes (non-normative)

The reference element specifies which elements have been signed within the source XML document. The URI attribute value is a fragment identifier which uses the value of the `id` attribute of the signed data. A fragment identifier consists of the '#' character followed by a unique identifier. For example, when the `id` attribute value is '12345', the fragment identifier would be '#12345'. Note any method or algorithm for generating an ID can be used as long as the generated ID is unique.

XML Signature allows for one or more `ds:Reference` elements, to allow for signatures over one or more fragments of XML. These obligations specify that only one fragment is signed. This criterion is designed for use with the `sp:SignedPayload` defined in section 3.1.1 which has one and only one fragment to sign.

4.3.3.3 Example (non-normative)

The example below shows the single `ds:Reference` element and URI attribute set to a unique value:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    ...
    <ds:Reference URI="#317c500e-ce82-428f-8987-f23636287b4e">
      ...
    </ds:Reference>
  </ds:SignedInfo>
</ds:Signature>
```

The example below shows the `id` attribute on the fragment that's been signed:

```
...
<data id="317c500e-ce82-428f-8987-f23636287b4e">
  <pathologyreport xmlns="http://www.example.org">
    <name>John Doe</name>
    <tests>
      <test name="test1">1000</test>
    </tests>
  </pathologyreport>
</data>
```

```

    </tests>
  </pathologyreport>
</sp:data>
...

```

4.3.4 ds:Transforms

4.3.4.1 Obligations

XS 4.3.4.1-1 The `ds:Transforms` element in the `ds:Reference` element **MUST** be present.

4.3.4.2 Notes (non-normative)

XML Signature allows the `ds:Transforms` element to be optional. If it is not present, no transforms are performed on the data being signed. The transforms are applied to the content to create the representation that is actually signed. This is useful when the content can have multiple syntactic forms which are considered the same. The transforms are used to create the same representation between the entity signing the data and the entity verifying the signature.

These obligations specify that there will always be transforms. The particular transform is described in section 4.3.5.

4.3.5 ds:Transform

4.3.5.1 Obligations

XS 4.3.5.1-1 There **MUST** be only one `ds:Transform` element in the `ds:Transforms` element.

XS 4.3.5.1-2 Implementations **MUST** use the Exclusive XML Canonicalization method over the contents being signed and indicate this by setting the `Algorithm` attribute on the `ds:Transform` element to `"http://www.w3.org/2001/10/xml-enc-c14n#" .`

4.3.5.2 Notes (non-normative)

XML Signature allows for zero or more transformations to be applied to the data being signed before the digest is calculated.

These obligations specify that there will be exactly one transformation and that transformation is the Exclusive XML Canonicalization method.

The canonicalization method specified on the `ds:Transform` element only applies to the data being signed. A canonicalization method is also applied to the `ds:SignedInfo` element, see section 4.3.1 for details.

4.3.5.3 Example (non-normative)

The example below shows the `ds:Transforms` element, `ds:Transform` element, and `Algorithm` attribute set to the Exclusive XML Canonicalization method:

```

<ds:Signature Id="_1">
  <ds:SignedInfo>
    ...
    <ds:Reference URI="#_5002">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-enc-c14n#" />
      </ds:Transforms>
    ...
  </ds:Reference>
  ...

```

```

</ds:SignedInfo>
...
</ds:Signature>

```

4.3.6 ds:DigestMethod

4.3.6.1 Obligations

XS 4.3.6.1-1 Implementations **MUST** use the SHA-1 digest method and indicate this by setting the `Algorithm` attribute on the `ds:DigestMethod` element to "http://www.w3.org/2000/09/xmldsig#sha1".

The SHA-1 algorithm is defined in section 6.2.1 of *XML-Signature Syntax and Processing* [XSSP2002].

4.3.6.2 Notes (non-normative)

XML Signature allows different algorithms to be used to calculate the digest value. A digest is a fixed length "fingerprint" that is calculated from a block of data. When the data the digest is calculated on changes, the digest also changes. The digest is used as part of the signature to ensure the integrity of the data has not been compromised.

These obligations specify that a particular algorithm is always used.

In future versions of this profile other algorithms could be specified.

4.3.6.3 Examples (non-normative)

The example below shows the `ds:DigestMethod` element and the `Algorithm` attribute set to the SHA1 digest method.

```

<ds:Signature Id="_1">
  <ds:SignedInfo>
    ...
    <ds:Reference URI="#_5002">
      ...
      <ds:DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <ds:DigestValue>KjbslWMIoEM/612TlLnUroSHRIA=</ds:DigestValue>
    </ds:Reference>
    ...
  </ds:SignedInfo>
  ...
</ds:Signature>

```

4.3.7 ds:DigestValue

There are no additional obligations on the use of the `ds:DigestValue` element.

4.4 ds:SignatureValue

There are no additional obligations on the use of the `ds:SignatureValue` element.

4.5 ds:KeyInfo

4.5.1 ds:KeyInfo (in Signature)

4.5.1.1 Obligations

XS 4.5.1.1-1 The `ds:KeyInfo` element in the `ds:Signature` element **MUST** be present.

4.5.1.2 Notes (non-normative)

XML Signature allows an optional `ds:KeyInfo` element in the `ds:Signature` element. If it is not present, the key used for the signature is known implicitly or indicated by an external mechanism.

These obligations specify that the key is explicitly included so that an external mechanism is not needed.

4.5.2 ds:X509Data

4.5.2.1 Obligations

XS 4.5.2.1-1 The `ds:X509Data` element in the `ds:KeyInfo` element **MUST** be present.

XS 4.5.2.1-2 The `ds:X509Certificate` element in the `ds:X509Data` element **MUST** be present and contain the encoded value of the signing certificate unless the service specification states a different mechanism is used.

4.5.2.2 Notes (non-normative)

Allowing the certificate of the signer to be included in the signature allows the recipient to verify the signature without having to externally retrieve the certificate.

These obligations specify that the X509 certificate is included.

4.5.2.3 Example (non-normative)

The example below shows the `ds:X509Data` element and the `ds:X509Certificate` element with a certificate encoded using base64:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig">
  ...
  <ds:KeyInfo >
    <ds:X509Data>
      <ds:X509Certificate>657f9bup... </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

4.6 ds:Object

4.6.1 ds:Object

4.6.1.1 Obligations

XS 4.6.1.1-1 The `ds:Object` element in the `ds:Signature` element **MUST NOT** be present.

4.6.1.2 Notes (non-normative)

XML Signature allows for zero or more `ds:Object` elements at the end of the `ds:Signature` element. These contain arbitrary data. Typically they are used to contain the data being signed when the enveloping signatures method is used.

Since detached signatures are used (see section 4.1.2), the `ds:Object` element is not required.

5 XML Encryption

This chapter contains criteria on using XML Encryption.

This chapter is organised into seven sections:

- Standards, containing obligations on the XML Encryption standard;
- `xenc:EncryptedData`;
 - `xenc:EncryptionMethod`;
 - `xenc:KeyInfo`;
 - `xenc:CipherData`; and
 - `xenc:EncryptionProperties`; and
- `xenc:EncryptedKey`.

The second section, `xenc:EncryptedData`, covers the root element of XML Encryption. The four sections after it (`xenc:EncryptionMethod` through to `xenc:EncryptionProperties`) cover each of the child elements of `xenc:EncryptedData`. The last section covers the `xenc:EncryptedKey`.

Encryption Process

The obligations in this chapter are designed to support the normal practice of encrypting using PKI.

The creator steps:

1. Generate a random session key that will only be used for this encryption operation.
2. Encrypt the plaintext data using a symmetric cryptographic algorithm and the session key.
3. Encrypt the session key using the public certificate of the consumer who will decrypt the data. When there are multiple public certificates used for encrypting, this step is repeated for each one.

The consumer steps:

4. Decrypt the encrypted session key using their private key.
5. Decrypt the encrypted data using the session key it has obtained from step 4.

5.1 Standards

5.1.1 XML Encryption

5.1.1.1 Obligations

XS 5.1.1.1-1 Implementations **MUST** use the *XML Encryption Syntax and Processing* Recommendation from W3C [XESP2002] and be valid against the XML Encryption XML Schema [EXSD2002].

5.1.1.2 Notes (non-normative)

The *XML Encryption Syntax and Processing* Recommendation from the W3C specifies a process for encrypting data and representing the result as XML.

The result of the encryption process is an `xenc:EncryptedData` element. The `xenc:EncryptedData` element has four possible child elements. The criteria for these are described in the following sections:

- `xenc:EncryptionMethod` (see section 5.3)
- `ds:KeyInfo` (see section 5.4)
- `xenc:CipherData` (see section 5.5); and
- `xenc:EncryptionProperties` (see section 5.6).

5.1.1.3 Example (non-normative)

The result of using XML Encryption produces an instance of the `xenc:EncryptedData` element. This is an example of this element:

```
<xenc:EncryptedData Id="_1" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
  <xenc:CipherData>
    <xenc:CipherValue>
iVBORwOKGgoAAAANSUHEUGAAACQAAAAMCAIAAACFnYiUAAAAIGNIUk0AAH0LAACA
gwAA+f8AAIDpAAB1MAAA6mAAADqYAAAXb5JfxUYAAAAJcEhZcwAACxMAAAATAQCa
nBgAAAAJdnBBZwAAACQAAAAMACVatSUAAMqSURBVDjLrVRfaFNnFD/fd29uc9Mt
iakasWQ102H80wjVsxHr8+Zch3zRlYKPCtuLsqfBUBRFQVSKT/6BMmuh6oNsyigm
s1Ksi7RK1YImrWln2uxem5jem4vJ/fN9x4ebUvRBKHpePjh8v/Pn9zvnEESExRgi
ctsWPB4gZFFAAKCLBei56Rv79pcV5b0SStlJU9c/cTK7UskPP+SMveM150aBg9PD
Ix/Giu6jPnlkPaKhvppIJMMbNzTv3kUoBQDOeSaTKRQKwWAwFouJokgIkQMBQ1HT
f90yCrOb0joav1qjjD7Rs5PZxD/lmfyKeHM43vz85i1ldJQ7TuPXW2Pt01zOa509
7um9/N2Pjy51W4Zx5/CRE6fOujkSiUQ6nfb5fJlMjplMIoAgSdWS1vztd3VsTB0b
u7q7w1DU8v+KqRvF8YmX94b06Zmqpj2+0muVDSKKd48dHzzxcKfWRPz74K/XO/cy
x0HE0Z7e863fIOfpTKavr69arSKiYRjd3d3K7Kw2NXVu/aZcKuUCz2/d9rTvGiL+
8f328du3XSdyjoiOaer5/P2zXRdav3UsExFrNCLjwaYIFQQAqF+2VJAKYLxYLFYq
lYGBAcYYIYRS6jgOZ9zjkwORiAusDy93LAsAuOMw05rXtXr32PHc0L/eYPBNseCR
fchxQTMA4LYzn5gBASDAOA+FQmltbYwxAJAkqc7rft0+gciZbdc+c+6qCwhUrEW7
39U1NTi489LFwBeRmeGRO4e0AucLmiFnOD9giMgdBghNTU2KopTLZb/f7/f7NU2r
miYhwBmD+elExpBzAOCMabmcS1JpcnLZ+nVLY2uFurpc6oFjmu6AlGqhkktqpNoY
C4Ioy4w5kcbG1paW/v7+hoYG27Yty/qhvZ1SQfivsvrBriLReBkoBIN65Z+j02Wc3
/ox3/NT6y8/XO/f27Gj3eL2VkuYNBWqR3QuiT88QSj5fuRIATF031Feh1V+6/MzN
zRUKBVEUw+GwLMvMNEtT/y2JrhIkCQBeZ7NyMciHQgCQH3k418slbtkciERKL7Iv
U61QNBqONxuquiQaJZSSxZ6rj7G32Ee8Hsuzt6IAAAAASUVORK5CYII=
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>
```

5.2 xenc:EncryptedData

5.2.1 Encryption granularity

5.2.1.1 Obligations

- XS 5.2.1.1-1 Implementations **MUST** use XML Encryption “element” level encryption as defined in [XESP2002] and indicate this by including the `Type` attribute on the `xenc:EncryptedData` element and setting it to “`http://www.w3.org/2001/04/xmlenc#Element`”.
- XS 5.2.1.1-2 The `Id` attribute on the `xenc:EncryptedData` element **MUST** be set to a unique value.

5.2.1.2 Notes (non-normative)

XML Encryption can be used with either ‘element’ or ‘content’ level encryption. Content level encryption leaves the root element of the data being encrypted and replaces its contents with an `xenc:EncryptedData` element. Element level encryption encrypts everything including the root element and replaces it with the `xenc:EncryptedData` element. This obligation specifies element level encryption to ensure that information in the name of the root element and its attributes will not be exposed, thus increasing security.

XML Encryption has an optional `Type` attribute on the `xenc:EncryptedData` element to indicate the level of encryption granularity. These obligations

make the `Type` attribute mandatory, so that the encryption granularity is explicit.

The `Id` attribute can be used when the `xenc:EncryptedKey` element is detached from the `xenc:EncryptedData` element. This allows the encrypted key to reference data it was used to encrypt. The encrypted key will use the `Id` value within its reference list. Note any method or algorithm for generating an ID can be used as long as the generated ID is unique.

5.2.1.3 Example (non-normative)

The example below shows the `xenc:EncryptedData` element with the `Type` attribute set to use 'element' level encryption:

```
<xenc:EncryptedData Id="_1"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
  ...
</xenc:EncryptedData>
```

5.2.2 MIME Type and Encoding

5.2.2.1 Obligations

XS 5.2.2.1-1 The `Encoding` attribute on the `xenc:EncryptedData` element **MUST NOT** be present.

XS 5.2.2.1-2 The `MimeType` attribute on the `xenc:EncryptedData` element **MUST NOT** be present.

5.2.2.2 Notes (non-normative)

XML Encryption provides optional `MimeType` and `Encoding` attributes on its `xenc:EncryptedData` element. These attributes are purely advisory.

These obligations state that these attributes are not to be used. It is expected that the context in which the XML Encryption is used will precisely define the types and encoding of the data. Therefore, it is redundant to specify them using these attributes.

5.3 xenc:EncryptionMethod

5.3.1 Symmetric encryption using AES-256

5.3.1.1 Obligations

XS 5.3.1.1-1 The `xenc:EncryptionMethod` element in the `xenc:EncryptedData` element **MUST** be present.

XS 5.3.1.1-2 Implementations **MUST** encrypt the content data using AES-256 and indicate this by including the `Algorithm` attribute on the `xenc:EncryptionMethod` element and setting it to "http://www.w3.org/2001/04/xmlenc#aes256-cbc". AES-256 is defined in [FIPS197].

XS 5.3.1.1-3 The `xenc:KeySize` element in the `xenc:EncryptionMethod` element **MAY** be present.

XS 5.3.1.1-4 The `xenc:KeySize` element **MUST** have the value of "256" if present.

XS 5.3.1.1-5 The `xenc:OAEPparams` element in the `xenc:EncryptionMethod` element **MUST NOT** be present.

5.3.1.2 Notes (non-normative)

The obligations in this section cover the symmetric encryption from the Encryption Process: step 2 for the creator and step 5 for the consumer.

AES-256 was chosen because it is endorsed by [ACSI33].

The `xenc:OAEPparams` element is not used when AES encryption is used.

In the future other algorithms could be specified.

5.3.1.3 Example (non-normative)

The example below shows the `xenc:EncryptionMethod` element with the `Algorithm` attribute set to use AES-256:

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
  ...
</xenc:EncryptedData>
```

5.4 ds:KeyInfo

5.4.1 ds:KeyInfo (in EncryptedData)

5.4.1.1 Obligations

xs 5.4.1.1-1 The `ds:KeyInfo` element in the `xenc:EncryptedData` element **MUST NOT** be present.

5.4.1.2 Notes (non-normative)

XML Encryption allows information about the keys to be optional. If it is not present it is implicitly known or is defined by an external mechanism. Since the `xenc:EncryptedKey` has been detached, the `ds:KeyInfo` is not required. Note that the `ds:KeyInfo` within the `xenc:EncryptedKey` element is still required.

5.5 xenc:CipherData

5.5.1 xenc:CipherData (EncryptedData)

5.5.1.1 Obligations

xs 5.5.1.1-1 The `xenc:CipherData` element in the `xenc:EncryptedData` element **MUST** be present.

xs 5.5.1.1-2 The `xenc:CipherReference` element in the `xenc:CipherData` element **MUST NOT** be present.

xs 5.5.1.1-3 The `xenc:CipherValue` element in the `xenc:CipherData` element **MUST** be present.

5.5.1.2 Notes (non-normative)

The `xenc:CipherData` element contains the raw encrypted data. XML Encryption allows the cipher value to be referenced externally or embedded.

These obligations specify that the cipher value of the encrypted data is embedded.

5.5.1.3 Example (non-normative)

The example below shows the `xenc:CipherData` element and the `xenc:CipherValue` element of the `xenc:EncryptedData` element which contains the cipher value:

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
...
  <xenc:CipherData>
    <xenc:CipherValue>
eQXAd06hM2ZX3HmUHO83au473wWMFzB29CCE+zGNQggqldudMKJhyWMm329bWzS7qXYgBacKR8tD
DXnytZD4lbs1QMULCXvWDYDe3f1AbmEen2KuDK+FQpPd3t3ViuX518ViS+KJqekLXvdo9Q8SIRSM
/4tuLcygBJKB8iXT/PVln2yisinC6KdtTlQylhlMv+6gE+juvLXIV+5lQ36Op/xTrwv2rLdRJ9YN
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>
```

5.6 xenc:EncryptionProperties

5.6.1 xenc:EncryptionProperties

5.6.1.1 Obligations

XS 5.6.1.1-1 The `xenc:EncryptionProperties` element in the `xenc:EncryptedData` element **MUST NOT** be present.

5.6.1.2 Notes (non-normative)

In XML Encryption, the `xenc:EncryptionProperties` is optional. It is designed to contain additional information about the generation of the `xenc:EncryptedData` or `xenc:EncryptedKey`.

This obligation specifies that the `xenc:EncryptionProperties` element is not used, because no additional information needs to be conveyed.

5.7 ds:EncryptedKey

5.7.1.1 Obligations

- XS 5.7.1.1-1 The `MimeType` attribute on the `xenc:EncryptedKey` element **MUST NOT** be present.
- XS 5.7.1.1-2 The `Encoding` attribute on the `xenc:EncryptedKey` element **MUST NOT** be present.
- XS 5.7.1.1-3 The `Recipient` attribute on the `xenc:EncryptedKey` element **MUST NOT** be present.
- XS 5.7.1.1-4 The `xenc:ReferenceList` element in the `xenc:EncryptedKey` element **MUST** be present and have a single `xenc:DataReference` attribute with the `URI` attribute set to the `Id` value of the `xenc:EncryptedData` element.
- XS 5.7.1.1-5 The `xenc:CarriedKeyName` element in the `xenc:EncryptedKey` element **MUST NOT** be present.
- XS 5.7.1.1-6 A new session key **MUST** be generated each time a payload is encrypted.

5.7.1.2 Notes (non-normative)

The obligations in this section cover the asymmetric encryption of the session key from the Encryption Process: step 1 and 2 for the creator and step 4 for the consumer.

In XML Encryption, the `xenc:EncryptedKey` in the `ds:KeyInfo` element is mandatory. These criteria specify that the `MimeType`, `Encoding` and `Recipient` attributes on that element are not used because they are not needed.

The `xenc:ReferenceList` element is required when the `xenc:EncryptedKey` element is detached from the `xenc:EncryptedData` element. The `URI` attribute of the `xenc:DataReference` element has a unique fragment identifier which references the `xenc:EncryptedData` element the key was used to encrypt. The value of the `URI` attribute must be the same value as the `Id` attribute on the `xenc:EncryptedData` element in section 5.2.

5.7.1.3 Example (non-normative)

The example below shows the `xenc:EncryptedKey` element:

```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
  ...
  <xenc:CipherData>
    <xenc:CipherValue>
ZwafQcp4ccXi2yaffEVZGHHmx80jc9Rsgghkyu2CmJ6hRUt1LUHfy8/UuXXi0SDeg75k3kJpIUuxV
cvDUCvB4x8/HMJT2wGwug59trb51SV01q/VV1qfUBA0YkFb+WlSh6Zt2cWmcVz+Q0D2gOLk1YeQO
7cZmvGK3QBliKkXJ0bJVkT+Fa8DN4uOqxm7urm++JzSfyqjK6w63U1vi0ngRuAb0+LsIAz9wCjTp
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedKey>
```

5.7.2 Asymmetric encryption using RSA 1.5

5.7.2.1 Obligations

XS 5.7.2.1-1 The `xenc:EncryptionMethod` element in the `xenc:EncryptedKey` element **MUST** be present.

XS 5.7.2.1-2 Implementations **MUST** use the RSA 1.5 algorithm to encrypt the session key and indicate this by including the `Algorithm` attribute on the `xenc:EncryptionMethod` element and setting it to "http://www.w3.org/2001/04/xmlenc#rsa-1.5"

RSA 1.5 is defined in [PKCS1-1993].

5.7.2.2 Notes (non-normative)

The obligations in this section cover the asymmetric encryption from the Encryption Process: step 3 for the creator and step 4 for the consumer.

RSA 1.5 was chosen because it is endorsed by [ACSI33].

Public Key Infrastructure (PKI) security tokens were chosen to match the authentication mechanism used by NEHTA's national e-health environment.

In the future other algorithms could be specified.

5.7.2.3 Example (non-normative)

The example below shows the `xenc:EncryptionMethod` element with the `Algorithm` attribute set to use RSA 1.5:

```
<xenc:EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  ...
</xenc:EncryptedKey>
```

5.7.3 ds:KeyInfo (in xenc:EncryptedKey)

5.7.3.1 Obligations

XS 5.7.3.1-1 The ds:KeyInfo element in the xenc:EncryptedKey element **MUST** be present.

5.7.3.2 Notes (non-normative)

The ds:KeyInfo element is required because it specifies which key was used to encrypt the encrypted key.

5.7.4 ds:X509Data

5.7.4.1 Obligations

XS 5.7.4.1-1 Implementations **MUST** encrypt the symmetric session key using the public asymmetric key of the Consumer.

XS 5.7.4.1-2 The ds:X509Data element in the ds:KeyInfo (EncryptedKey) element **MUST** be present as the one and only child element.

XS 5.7.4.1-3 The ds:X509SKI element in the ds:X509Data element **MUST** be present and have the Subject Key Identifier (SKI) of the certificate that was used to encrypt the session key as its value.

5.7.4.2 Notes (non-normative)

The obligations in this section cover the asymmetric encryption from the Encryption Process: step 3 for the creator and step 4 for the consumer.

The Subject Key Identifier (SKI) is used to uniquely identify the certificate used to encrypt the data. This is consistent with the usage of Subject Key Identifiers in NEHTA e-health environment.

5.7.4.3 Example (non-normative)

The example below shows the ds:KeyInfo element that contains the ds:X509Data element and ds:X509SKI element:

```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <ds:KeyInfo>
    <ds:X509Data>
      <ds:X509SKI>657f9bupP2LWtSvgCc3XpbRSiec=</ds:X509SKI>
    </ds:X509Data>
  </ds:KeyInfo>
  ...
</xenc:EncryptedKey>
```

5.7.5 xenc:CipherData (EncryptedKey)

5.7.5.1 Obligations

XS 5.7.5.1-1 The xenc:CipherData element in the xenc:EncryptedKey element **MUST** be present.

XS 5.7.5.1-2 The xenc:CipherReference element in the xenc:CipherData element **MUST NOT** be present.

XS 5.7.5.1-3 The xenc:CipherValue element in the xenc:CipherData element **MUST** be present.

5.7.5.2 Notes (non-normative)

The `xenc:CipherData` element contains the raw encrypted data. XML Encryption allows the cipher value to be referenced externally or embedded.

These obligations specify that the cipher value of the encrypted key is embedded within the `xenc:CipherData` element.

5.7.5.3 Example (non-normative)

The example below shows the `xenc:CipherData` element and the `xenc:CipherValue` element of the `xenc:EncryptedKey` element which contains the cipher value:

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
  <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <xenc:EncryptedKey>
      ...
      <xenc:CipherData>
        <xenc:CipherValue>
KCWjMTZPFMs6XV1I8mLAn2Ah+Y7F4uzfEnltUU7M1hZ3moke2y4LRUw13dRssOf3lgS2IoisW5DWfiQVZQ
mT+3S0Jtw+0npNegjKe8ucm0S5F0HzOtkJiHA8veEWOrtw3PKab8QIN7TbdgPhfU5WkmLpHJy9hZBQv0LL
PfFNXvU=
        </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedKey>
  </ds:KeyInfo>
  ...
</xenc:EncryptedData>
```

Appendix A: Informative references

- [ACSI33] Defence Signals Directorate, *Australian Government Information and Communications Technology Security Manual*, ACSI 33, 19 September 2005.
- [XSD2004a] W3C, *XML Schema Part 1: Structures, Second Edition*, W3C Recommendation, 28 October 2004
<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- [XSD2004b] W3C, *XML Schema Part 2: Datatypes, Second Edition*, W3C Recommendation, 28 October 2004
<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- [CPIS2008] NEHTA, *Concepts and Patterns for Implementing Services 2.0*, 26 May 2008
- [WSP2008] NEHTA, *Web Services Profile 3.0*, 26 May 2008
- [EIJAX2008] NEHTA, *Example Technical Implementation of Interoperable Web Services JAX-WS 3.0*, 3 July 2007
- [EIWCF2008] NEHTA, *Example Technical Implementation of Interoperable Web Services WCF 3.0*, 3 July 2007
- [EIWSE2008] NEHTA, *Example Technical Implementation of Interoperable Web Services WSE 3.0*, 3 July 2007
- [XEXSD2002] W3C, *XML Encryption Syntax and Processing*, W3C Recommendation XML Schema, 10 December 2002,
<http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd>
- [XSXSD2002] W3C, *XML Signature Syntax and Processing*, W3C Recommendation XML Schema, 12 February 2002
<http://www.w3.org/TR/xmlsig-core/xmlsig-core-schema.xsd>