



XML Secured Payload Profile

Version 1.2 — 30 June 2009

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 © 2009, 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	1
1.1 Background	1
1.2 Purpose	1
1.3 Scope	1
1.3.1 Content	1
1.3.2 Related Publications	1
1.3.3 Audience	2
1.4 Document map	2
1.5 Normative references	3
1.6 Definitions	3
1.6.1 Acronyms	3
1.6.2 Namespaces	3
1.7 Conformance	4
1.8 Overview	4
2 Profiles	5
2.1 Signed profile	5
2.2 Encrypted profile	5
2.3 Signed container profile	5
2.4 Encrypted container profile	5
3 Containers	6
3.1 XML Schema definitions	6
3.1.1 Signed Payload	6
3.1.2 Encrypted Payload	8
4 XML Signature	11
4.1 Standards	12
4.1.1 XML Signature	12
4.1.2 Detached signature	12
4.2 ds:Signature	13
4.3 ds:SignedInfo	13
4.3.1 ds:CanonicalizationMethod	13
4.3.2 ds:SignatureMethod	14
4.3.3 ds:Reference	14
4.3.4 ds:Transforms	15
4.3.5 ds:Transform	15
4.3.6 ds:DigestMethod	16
4.3.7 ds:DigestValue	17
4.4 ds:SignatureValue	17
4.5 ds:KeyInfo	17
4.5.1 ds:KeyInfo (in Signature)	17
4.5.2 ds:X509Data	17
4.6 ds:Object	18
4.6.1 ds:Object	18
5 XML Encryption	19
5.1 Standards	19
5.1.1 XML Encryption	19
5.2 xenc:EncryptedData	20
5.2.1 Encryption granularity	20

5.2.2	MimeType and Encoding.....	21
5.3	xenc:EncryptionMethod	21
5.3.1	Symmetric encryption using AES-256	21
5.4	ds:KeyInfo.....	22
5.4.1	ds:KeyInfo (xenc:EncryptedData).....	22
5.5	xenc:CipherData.....	22
5.5.1	xenc:CipherData (xenc:EncryptedData).....	22
5.6	xenc:EncryptionProperties	23
5.6.1	xenc:EncryptionProperties	23
5.7	ds:EncryptedKey	23
5.7.2	Asymmetric encryption using RSA 1.5	24
5.7.3	ds:KeyInfo (xenc:EncryptedKey)	25
5.7.4	ds:X509Data	25
5.7.5	xenc:CipherData (xenc:EncryptedKey)	26
Appendix A: Informative references		27
Appendix B: Change log.....		28

Document information

Change history

Version	Date	Comments
1.0	2008-12-01	Final
1.1 draft	2009-02-20	Draft
1.2	2009-06-30	Final

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 communication between organisations 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 criteria that must be met. These criteria 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 may be 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 Java and .NET can be found in *Example Technical Implementation of the XML Secured Payload Profile*, [ETIJAVA2008], and [ETINET2008].

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 Document map

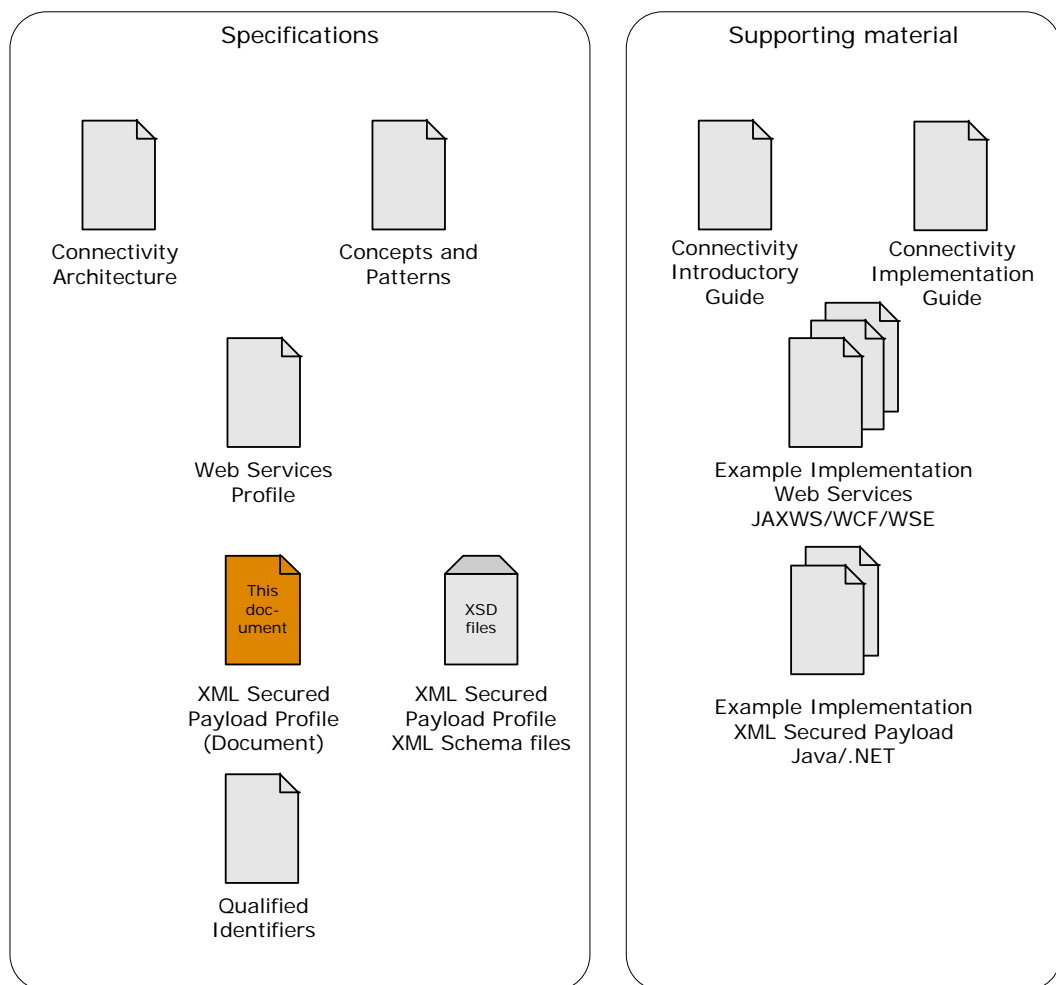


Figure 1: Document map

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/>
- [EXSD2002] W3C, *XML Encryption Syntax and Processing*, W3C Recommendation XML Schema, 10 December 2002, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd>
- [XSSP2002] W3C, *XML Signature Syntax and Processing*, W3C Recommendation, 12 February 2002 <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>
- [XSXSD2002] W3C, *XML Signature Syntax and Processing*, W3C Recommendation XML Schema, 12 February 2002 <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/xmlsig-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
IETF	Internet Engineering Task Force
PKI	Public Key Infrastructure
RFC	Request For Comments
SHA	Secure Hash Algorithm
SKI	Subject Key Identifier
XML	Extensible Markup Language

1.6.2 Namespaces

The prefix "ep" is used to refer to the Encrypted Payload namespace, "http://ns.nehta.gov.au/CoreConnectivity/Xsd/EncryptedPayload/1.2".

The prefix "sp" is used to refer to the Signed Payload namespace, "http://ns.nehta.gov.au/CoreConnectivity/Xsd/SignedPayload/1.2".

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 4 and 5.

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 criteria that must be met. These criteria identify standards to use and how they are to be applied.

This document defines four profiles. Two of the profiles specify constraints on the XML Signature and XML Encryption specifications. The other two profiles correspond to 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.

Note that the *Concepts and Patterns for Implementing Services 2.0* also defines the patterns "None" and "Signed Before Encrypted". The "None" pattern indicates that no additional security over the Web services point-to-point security mechanisms defined in *Web Services Profile* [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 specifies constraints over the XML Signature specification.

Conformance to this profile requires conformance to all the criteria specified in chapter 4, XML Signature.

2.2 Encrypted profile

The encrypted profile specifies constraints over the XML Encryption specification.

Conformance to this profile requires conformance to all the criteria specified in chapter 5, XML Encryption.

2.3 Signed container profile

The Signed container profile represents payload that is 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:

- Signed Payload (section 3.1.1)

The signed container profile makes use of the signed profile (section 2.1).

2.4 Encrypted container profile

The Encrypted container profile represents payload that is 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:

- Encrypted Payload (section 3.1.2)

The encrypted container profile makes use of the encrypted profile (section 2.2).

3 Containers

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

The data types 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 datatypes are defined using XML Schema [XSD2004a] [XSD2004b]. These data types are:

- `sp:SignedPayload` — for holding a payload and one or more signatures on the payload.
- `ep:EncryptedPayload` — for holding an encrypted payload and one or more encrypted keys.

3.1.1 Signed Payload

The Signed Payload is a container for holding a payload and one or more digital signatures on the payload.

3.1.1.1 Criteria

XS 3.1.1.1-1 Implementations **MUST** comply with the Signed Payload 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="http://ns.nehta.gov.au/CoreConnectivity/Xsd/SignedPayload/1.2"
  xmlns:tns="http://ns.nehta.gov.au/CoreConnectivity/Xsd/SignedPayload/1.2"
  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 `id` attribute **MUST** be set to a value unique in the XML document.
- xs 3.1.1.1-4 The `sp:signedPayloadData` element **MUST** be the only element signed.

3.1.1.2 Notes (non-normative)

The `sp:signedPayload` element is the root XML element of the signed payload container. It consists of the `sp:signatures` element and the `sp:signedPayloadData` element.

The `sp:signedPayloadData` element contains the payload that is to be signed. The `xsd:any` element within the `sp:signedPayloadData` element is a placeholder for the payload content. The `id` attribute on the `sp:signedPayloadData` 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 XML document that the payload is a part of.

The signer signs the `sp:signedPayloadData` element using their private key to create a digital signature, which is represented by a `ds:Signature` element. This `ds:Signature` element is added to the `sp:signatures` element.

The receiver retrieves the `ds:Signature` element within the `sp:signatures` element and validates the digital signature using the public key of the signer.

The `sp:signatures` element can contain one or more signatures of the `sp:signedPayloadData` element. This allows multiple entities to sign the payload. Each signer creates a digital signature and adds it to the `sp:signatures` element. The receiver uses the key information in each digital signature to find the public key needed to validate the digital signature.

Specifications that use this signed payload container profile should define how many signatures are expected, whether a receiver must validate all or a subset of the expected signatures, how the identity of the signer is to be established and how the trust of individual certificates is established.

The Signed profile specifies that one or more references can be present in a signature. This signed payload container further restricts it to use only one signature reference for the `sp:signedPayloadData` element.

3.1.1.3 Example (non-normative)

The following shows an example instance of the Signed Payload container:

```
<sp:signedPayload
  xmlns:sp="http://ns.nehta.gov.au/CoreConnectivity/Xsd/SignedPayload/1.2">
  <sp:signatures>
    <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="#a0ede6c4-21e4-44d5-93b3-4b2f02f9e1f8">
          <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>Orn5miXZzb9EYhwcMg4ZTWavRds=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>
        ffzZtvV4ms46VbZRLM6r0eMoOirKpwd1k0AVxhLqs/TPgtIjDggX4HMqmRbjNibAtO4KVmnLkx6NUq3PbN
        FY0DZCf9xTEw2HTOCSRvRpxlL0DPbVc5JylU0v3elhbeglCmhqfRyI9003b3se9WpF/nkDSZGAov2NmhWn
        bnNcqjE=
      </ds:SignatureValue>
    </ds:Signature>
  </sp:signatures>
</sp:signedPayload>
```

```

    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
MIICBjCCAW+gAwIBAgIBBjANBgkqhkiG9w0BAQUFADAeMRwwGgYDVQQQExNORUUhUQSBFsU1XUyBkZW1vIE
NMBB4XDTA3MDYyMjAwMDI1NFoXDTE3MDYxOTAwMDI1NFowFzEVMBMGAlUEAxQMamF4d3NfY2xpZW50MIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDVNtVihAhN76KhBevo/YTzBx1oB7K8YqRhfG3ca/Y9qFv1Mk
B6tUNjC9LxBqtLcQbcLpeZOFsKumtygvvZUBAZlpTR7qMOeHcFMYCP1sVS4mToTRt8P33au2x6VvTj7AWa
Xr8Di1jv13hB/luPKRuflyw0hV7mzvSkclwHVBBn0QIDAQABolsWWTAMBgNVHRMBAf8EAjAAMB0GA1UdDg
QWBBS1A5kUs7rHCR8wZEX7Nb4m3k1zOzAfBgNVHSMEGDAwBQL0DcCBAjrOSF5Usiwo0701HtHBzAJBgNV
HREEAjAAMA0GCSqGSIb3DQEBAQUAA4GBAGEPlkuOg5RftVWrFP+PSgHueUugWHhOqUsUB5w3OPYhkIawVf
brXu4cQ+wSo96yiP/89xsxTwMWiWa0LQ02xmEZWf2F9FrcO2Ni9nZ1lulREtrd5Huino8OGmEB4AJWDhGV
0GAT45Ze/tuVg+Xa+YmvHiuYseLVGeirnEOmoPS2
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds: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 Encrypted Payload

The Encrypted Payload is a container for holding a payload that's been encrypted. The container consists of the encrypted payload plus one or more encrypted keys.

3.1.2.1 Criteria

XS 3.1.2.1-1 Implementations **MUST** comply with the Encrypted Payload 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=
    "http://ns.nehta.gov.au/CoreConnectivity/Xsd/EncryptedPayload/1.2"
  xmlns:tns="http://ns.nehta.gov.au/CoreConnectivity/Xsd/EncryptedPayload/1.2"
  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` elements **MUST** comply with all the criteria in section 5.7.
- XS 3.1.2.1-4 The `xenc:EncryptedKey` elements **MUST** only be used to encrypt the data within the `ep:encryptedPayloadData` element.

3.1.2.2 Notes (non-normative)

The `ep:encryptedPayload` element is the root XML element of the encrypted payload container. It consists of the `ep:keys` element and the `ep:encryptedPayloadData` element.

The payload is encrypted using a symmetric algorithm with a generated session key to create a `xenc:EncryptedData` element. This `xenc:EncryptedData` element is added to the `ep:encryptedPayloadData` element.

The generated session key is encrypted using an asymmetric algorithm with the intended receiver's public key to create the `xenc:EncryptedKey` element. This `xenc:EncryptedKey` element is added to the `ep:keys` element.

The receiver uses their private key to decrypt the `xenc:EncryptedKey` element to obtain the session key. The decrypted session key is then used to decrypt the `xenc:EncryptedData` element within the `ep:encryptedPayloadData` element.

The `ep:keys` element can hold one or more encrypted keys. This allows the message to be encrypted for multiple entities. When there are multiple intended receivers, the sender encrypts the session key with the public key of each receiver. The receiver uses the key information in the `xenc:EncryptedKey` element to find an encrypted key for which they have the corresponding private key.

Specifications that use this encrypted payload container profile should define whether data can be encrypted for multiple receivers.

The Encrypted profile specifies that one or more `xenc:DataReference` elements can be present within a `xenc:ReferenceList` element of an `xenc:EncryptedKey` element. This allows one key to be used to encrypt multiple pieces of data. Since the encrypted payload container only contains a single piece of data to be encrypted, the `xenc:ReferenceList` element is constrained to have one `xenc:DataReference` element.

3.1.2.3 Example (non-normative)

The following shows an example instance of the Encrypted Payload container:

```
<ep:encryptedPayload
  xmlns:ep="http://ns.nehta.gov.au/CoreConnectivity/Xsd/EncryptedPayload/1.2">
  <ep:keys>
    <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509SKI>pQOZFLO6xwkfMGRF+zW+Jt5Nczs</ds:X509SKI>
        </ds:X509Data>
      </ds:KeyInfo>
    <xenc:CipherData>
```

```

    <xenc:CipherValue>
uQJemj3rJTgfm4hcPe5EebpoX3O6IBv9uLuOCSP6oKXYrFnPjCI0A7423fkRC0dqbvMkN3xOtQ94yzhF4H
py2Fs6YIEf3Xf3r6s8I+huDecpEa410ZP4/+uVVL/FEmdZeRvf0ayhp0+miRD0rOtFP2peiNjT6G7ggIK5
vkOnNzw=
    </xenc:CipherValue>
  </xenc:CipherData>
  <xenc:ReferenceList>
    <xenc:DataReference URI="#_1" />
  </xenc:ReferenceList>
</xenc:EncryptedKey>
</ep:keys>
<ep:encryptedPayloadData>
  <xenc:EncryptedData Id="_1"
    Type="http://www.w3.org/2001/04/xmlenc#Element"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
    <xenc:CipherData>
      <xenc:CipherValue>
BQ6byRM5fjzm0IU1AYIHxd5ufsu7/ctJEGxRfo04JfGA4qa3PgTjaBry/9YN8wqb1IoGxiNeyGAzenxONY
oky+AVMAdqtsh6QzKx1Ze3Xnj5I8oTHMA8EfL0R/w8ObuHhQxHnfxbf6yNIZik09dS6Z41pLAVyphEotaU
v+TWE+fhdhs/ti3wnxqSSA8EsDIY8d61N5P8A3A1ZY73dynwWw8gJu8pWctYQYzz+ezV1ng=
      </xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</ep:encryptedPayloadData>
</ep:encryptedPayload>

```

The following shows an example instance of the Encrypted Payload container in its decrypted form:

```

<ep:encryptedPayload
  xmlns:sp="http://ns.nehta.gov.au/CoreConnectivity/Xsd/EncryptedPayload/1.2">
  <ep:keys>
    ...
  </ep:keys>
  <ep:encryptedPayloadData>
    <pathologyreport xmlns="http://www.example.org">
      <name>John Doe</name>
      <tests>
        <test name="test1">1000</test>
      </tests>
    </pathologyreport>
  </ep:encryptedPayloadData>
</ep:encryptedPayload>

```

4 XML Signature

This chapter contains criteria on using XML Signature.

This chapter is organised into six sections:

- Standards, containing criteria 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 *XML Signature* specification [XSSP2002] defines a process for creating and validating digital signatures on XML data and representing the result in XML.

An XML Signature is created by the following steps:

1. A digest value is calculated for each XML data fragment being signed. This involves first applying a set of transforms to the XML fragment, then calculating the digest on the transformed XML fragment. The transformations ensure the XML fragment is in a normalized form. This usually includes canonicalization. The information from this step is represented using a `ds:Reference` element.
2. The `ds:Reference` elements from the previous stage are added to a `ds:SignedInfo` element. A digest value is calculated on the `ds:SignedInfo` element which involves first applying XML canonicalization. This calculated digest value is signed using the signer's private key to create the `ds:SignatureValue` element. A `ds:KeyInfo` element is used to specify which key was used to create the signature. The `ds:SignedInfo`, `ds:SignatureValue` and `ds:KeyInfo` elements are added to a `ds:Signature` element which is the resulting signature.

An XML Signature is validated by the following steps:

1. A digest value is calculated for each `ds:Reference` element within the signature. This involves applying the transforms specified in the reference, then calculating the digest value on the transformed XML fragment. The calculated digest value is compared to the one that is within the `ds:Reference` element. When they don't match, the signature validation fails.
2. A digest value is calculated on the `ds:SignedInfo` element. This involves first applying canonicalization on this element. The digest value of the `ds:SignedInfo` element is retrieved from the signature value using the signer's public key. This digest value is compared with the calculated digest value. When they don't match, the signature validation fails.

4.1 Standards

4.1.1 XML Signature

4.1.1.1 Criteria

XS 4.1.1.1-1 Implementations **MUST** use the *XML-Signature Syntax and Processing Recommendation* from W3C [DSIG2002] 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
5K6uCWC4aYGrTpcn0/SkbO/SvA5VKc9/CQtzOTafSud/CHP8Jc3ZUB1SdHOMgBMACUt55z8jA==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
```

4.1.2 Detached signature

4.1.2.1 Criteria

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 signed data is embedded inside the XML Signature.
- Enveloped signature, where the XML signature is embedded inside the data that's being signed.
- Detached signature, where the signature is separate from the data being signed.

This criterion specifies the use of a detached signature.

4.2 ds:Signature

There are no additional criteria 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`

Criteria for using these are described in this section.

4.3.1 ds:CanonicalizationMethod

4.3.1.1 Criteria

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 before the signing algorithm is applied to create the signature value. The same canonicalization is applied to the `ds:SignedInfo` element as part of signature verification to obtain the same XML that was signed.

There are two types of canonicalization, Exclusive and Inclusive. These criteria 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 verification 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 a `ds:CanonicalizationMethod` element with an `Algorithm` attribute specifying the exclusive canonicalization method:

```
<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 Criteria

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 criteria specify that the RSA-SHA1 algorithms are to be used.

4.3.2.3 Example (non-normative)

The example below shows a `ds:SignatureMethod` element with an `Algorithm` attribute specifying the RSA-SHA1 algorithm:

```
<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 Criteria

XS 4.3.3.1-1 There **MUST** be one or more `ds:Reference` elements 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. 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.

4.3.3.3 Example (non-normative)

The example below shows a single `ds:Reference` element with a 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 an `id` attribute on the fragment that has 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>
</sp:data>
...
```

4.3.4 ds:Transforms

4.3.4.1 Criteria

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 criteria 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 Criteria

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-exc-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 criteria 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 a `ds:Transforms` element containing a `ds:Transform` element with an `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-exc-c14n#" />
      </ds:Transforms>
    </ds:Reference>
  </ds:SignedInfo>
  ...
</ds:Signature>
```

4.3.6 ds:DigestMethod

4.3.6.1 Criteria

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 over which the digest is calculated on changes, the digest also changes. The digest is used as part of the signature to ensure that the data has not been modified.

These criteria specify that a particular algorithm is always used.

4.3.6.3 Examples (non-normative)

The example below shows a `ds:DigestMethod` element with an `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>
```

```
</ds:Signature>
```

4.3.7 ds:DigestValue

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

4.4 ds:SignatureValue

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

4.5 ds:KeyInfo

4.5.1 ds:KeyInfo (in Signature)

4.5.1.1 Criteria

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

4.5.2 ds:X509Data

4.5.2.1 Criteria

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)

Including the certificate of the signer in the signature allows the recipient to validate the signature without having to externally retrieve the certificate.

These criteria specify that the X509 certificate be included.

4.5.2.3 Example (non-normative)

The example below shows a `ds:X509Data` element contain a `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 Criteria

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 can 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 criteria 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 *XML Encryption* specification [XESP2002] defines a process for encrypting and decrypting XML data and representing the result in XML.

Data is encrypted using XML Encryption by the following steps:

1. A random session key is generated.
2. The data is encrypted using a symmetric algorithm with the session key. Symmetric encryption is used for the data for better performance. The encrypted data is represented using the `xenc:EncryptedData` element.
3. The session key is encrypted using an asymmetric algorithm with the public key of the receiver. The encrypted session key is represented using the `xenc:EncryptedKey` element. The `xenc:EncryptedKey` element can use a `ds:KeyInfo` element to specify which key was used. The encrypted key can be added to the `ds:KeyInfo` element of the `xenc:EncryptedData` element or it can exist independently.

Data is decrypted using XML Encryption by the following steps:

1. The encrypted session key within the `xenc:EncryptedKey` element is decrypted using a private key of the receiver. The decrypted session key is the key that was used to encrypt the data.
2. The cipher text within the `xenc:EncryptedData` element is decrypted using the session key.

5.1 Standards

5.1.1 XML Encryption

5.1.1.1 Criteria

- XS 5.1.1.1-1 Implementations **MUST** use the *XML Encryption Syntax and Processing* Recommendation from W3C [XENC2002] and be valid against the XML Encryption XML Schema [XEXSD2002].

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 XML Encryption process generates an instance of the `xenc:EncryptedData` element. An example of this element follows:

```
<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>
iVBORw0KGgoAAAANSUHEUGAAACQAAAAAMCAIAAACFnYiUAAAAIGNIUk0AAHoLAACA
gWAA+f8AAIDpAAB1MAAA6mAAADqYAAAXb5JfxUYAAAAJcEhZcwAACxMAAAAsTAQCa
nBgAAAAJdnBBZwAAACQAAAAAMACVatSUAAMqSURBVDjLrVRfaFNnFD/fd29uc9Mt
iakasWQ102H80wjVsxHr8+ZcH3zR1YKPCtuLsqfBUBRFQVSKT/6BMmuh6oNsyigm
s1Ksi7RK1YImrWln2uxem5jem4vJ/fN9x4ebUvRBKHpePjh8v/Pn9zvnEESExRgi
ctSWPB4gZFFAAKCLBei56Rv79pcV5b0SStlJU9c/cTK7UskPP+SMveM150aBg9PD
Ix/Giu6jPnlKPaKhvppIJMMbNzTv3kUoBQDOeSaTKRQKwWAwFouJokgIkQMBQ1HT
f90yCrOb0joavlqjjD7Rs5PZxD/lmfyKeHM43vz85i1ldJQ7TuPXW2Pt01zOa509
7um9/N2Pjy51W4Zx5/CR6fOujKsiUQ6nfb5fJlMJplMtoAgSdWSlvztd3VsTB0b
u7q7w1DU8v+KqRvF8YmX94b06Zmqpj2+0muVDSKKd48dHzxxckFwRPz74K/XO/cy
x0HE0Z7e863fIOfpTKavr69arSKiYRjd3d3K7Kw2NXVu/aZcKuUCz2/d9rTvGiL+
8f328du3XSdyjoiOaer5/P2zXRdav3UsExFrNCLjwaYIFQQAqF+2VJAKyLxYLFYq
lYGBAcYYIYRS6jgOZ9zjkwORiAusDy93LAsAuOMw05rXtXr32PHc0L/eYPBNseCR
fchxQTMA4LYzn5gBASDAOA+FQmltbYwxAJAkqc7rft0+gciZbdc+c+6qCwhUrEW7
39U1NTi489LFwBeRmeGR04eOAucLmiFnOD9giMgdBghNTU2KopTLZb/f7/f7NU2r
miYhwBmD+e1ExpBzAOCMabmcS1JpcnLZ+nVLY2uFurpc6oFjmu6A1GqhkktqppNoY
C4Ioy4w5kcbGlpaw/v7+hoYG27Yty/qhvZ1SQfivsvrbRiLReBkoBIN65Z+j02Wc3
/ox3/NT6y8/XO/f27Gj3eL2VkuYNBWqR3Quit88QSj5fuRIATF031FehlV+6/MzN
zRUKBVEUw+GwLMvMNET/y2JrhIkCQBeZ7NyMCiHQgCQH3k418slbtkeiERKL7Iv
U6lQNBqONxuiQaJZSSxZ6rj7G32Ee8Hsuzt6IAAAAAASUVORK5CYII=
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedData>
```

5.2 xenc:EncryptedData

5.2.1 Encryption granularity

5.2.1.1 Criteria

- 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 criterion 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 criteria 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 the `Id` attribute is of the `NCName` type. An `NCName` can only start with a letter or `'_'` character and not a digit.

5.2.1.3 Example (non-normative)

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

```
<xenc:EncryptedData Id="_12345"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element">
  ...
</xenc:EncryptedData>
```

5.2.2 MIMEType and Encoding

5.2.2.1 Criteria

- 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 criteria 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 Criteria

- 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 `xenc:EncryptionMethod` element is used to specify which encryption algorithm is applied to the plaintext.

The criteria in this section cover the symmetric encryption from the Encryption Process: step 2 for the creator and step 1 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.

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/xmenc#"
  Type="http://www.w3.org/2001/04/xmenc#Element"
  Id="_12345">
  <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmenc#aes256-cbc"/>
  ...
</xenc:EncryptedData>
```

5.4 ds:KeyInfo

5.4.1 ds:KeyInfo (xenc:EncryptedData)

5.4.1.1 Criteria

- 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 must be implicitly known or be 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 (xenc:EncryptedData)

5.5.1.1 Criteria

- 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 criteria specify that the cipher value of the encrypted data is embedded.

5.5.1.3 Example (non-normative)

The example below shows an `xenc:EncryptedData` element containing an `xenc:CipherData` element which itself contains a `xenc:CipherValue` element. The cipher value is the encrypted payload data.

```
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  Type="http://www.w3.org/2001/04/xmlenc#Element"
  Id="_12345">
  ...
  <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 Criteria

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 criterion 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 Criteria

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 one or more `xenc:DataReference` elements with the `URI` attribute set to a fragment identifier using the `Id` attribute 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 criteria 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 is set to a unique fragment identifier which references the `xenc:EncryptedData` element that the key was used to encrypt. The fragment identifier must use the value of the `Id` attribute on the `xenc:EncryptedData` element from section 5.2. 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'.

5.7.1.3 Example (non-normative)

The example below shows an `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/HMJT2wGwug59trb51SV0lq/VV1qfUBAOYkFb+WlSh6Zt2cWmcVz+Q0D2gOLk1YeQO
7cZmvGK3QBlikkXJ0bJVkT+Fa8DN4uOqxm7urm++JzSfyqjK6w63U1vi0ngRuAb0+LsIAz9wCjTp
    </xenc:CipherValue>
  </xenc:CipherData>
  <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:DataReference URI="#_12345"
      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" />
  </xenc:ReferenceList>
</xenc:EncryptedKey>
```

5.7.2 Asymmetric encryption using RSA 1.5

5.7.2.1 Criteria

- 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 criteria 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 a `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 (xenc:EncryptedKey)

5.7.3.1 Criteria

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 Criteria

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 criteria 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 a `ds:KeyInfo` element that contains the `ds:X509Data` element and a `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 (xenc:EncryptedKey)

5.7.5.1 Criteria

- 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 criteria 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 an `xenc:EncryptedKey` element containing an `xenc:CipherData` element which itself contains a `xenc:CipherValue` element. The cipher value is the encrypted key.

```
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  ...
  <xenc:CipherData>
    <xenc:CipherValue>
KCwjMTZPFMs6XV1I8mLAn2Ah+Y7F4uzfEnltUU7M1hZ3moke2y4LRUw13dRssOf3lgS2IoisW5DWfiQVZQ
mT+3S0Jtw+0npNegjKe8ucm0S5FoHzOtkJiHA8veEWoRtw3PKab8QIN7TbdgPhfU5WkmLpHJy9hZBQvO1L
PfFNXvU=
    </xenc:CipherValue>
  </xenc:CipherData>
</xenc:EncryptedKey>
```

Appendix A: Informative references

- [ACSI33] Defence Signals Directorate, *Australian Government Information and Communications Technology Security Manual*, ACSI 33, 19 September 2005.
- [CPIS2008] NEHTA, *Concepts and Patterns for Implementing Services 2.0*, 1 December 2008.
- [ETIJAVA2008] NEHTA, *Example Technical Implementation of the XML Secured Payload Java v1.0*, 1 December 2008.
- [ETINET2008] NEHTA, *Example Technical Implementation of the XML Secured Payload .NET v1.0*, 1 December 2008.
- [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/>
- [WSP2008] NEHTA, *Web Services Profile 3.0*, 1 December 2008.

Appendix B: Change log

Version 1.2

- Split the two container profiles into four profiles to create profiles of XML signatures and XML encryption that can be used in other contexts besides just the two containers defined in this document.
- Added allowing a signature to have more than one reference within the signature profile.
- Added XS 3.1.2.1-4.

Version 1.1 draft 2009-02-20

- Added document road map