Open Metering System

Technical Report 03
XML Key Exchange

Issue 1.0.2 / 2015-10

RELEASE

Document History

Version	Date	Comment	Editor
1.0.0	2013-10 to 2014-06	Creation of this document	Th. Blank, A. Bolder, U. Pahl, W. Ton, M. Weidinger
1.0.1	2015-01 to 2015-09	Comments due to Voting report on ND 078: vote on OMS_DTR03 – XML Key Exchange 1.0.0 / 2014-06 and resolution of comments	Th. Blank, A. Bolder, U. Pahl, W. Ton, M. Weidinger
1.0.2	2015-10	Issue for release	U. Pahl, A.Bolder

Table of contents

List of figures

List of tables

1 Introduction

This document describes how encryption keys of meters can be distributed in a secure way between market members. The solution builds upon XML in order to be cross-platform compatible, and it uses XSD schemata to structure and validate the contents of the files.

2 Why use XSD files

XSD-files make it possible to enforce a mandatory structure and properties in an XML-file, while at the same time giving vendors the opportunity to fill in extra information. Thus vendors may add in manufacturer specific extra data (e.g. meter name, production information, extra encryption keys etc.).

3 Security

For security reasons, the key material in the XML-files shall be encrypted. This shall be done by encrypting each of the Key elements in the XML-files (see the examples in Section 5). The encryption algorithm shall be AES wrapper (RFC3394) (Algorithm='http://www.w3.org/2001/04/xmlenc#kw-aes128'). The key for encrypting a Key element shall either be included in the XML-file itself using a TransportKey element or be supplied by some out-of-band mechanism. If the TransportKey method is used, the encryption algorithm shall be RSA OAEP (Algorithm='http://www.w3.org/2001/04/xmlenc#kw-aes256').

Different Key elements in the same XML-file may be encrypted with the same or different keys, as needed. A mandatory signature which covers the device and key material guarantee the integrity of the whole file. This signature shall be done with RSA-SHA256.

The examples in Section 5 present XML-files which are compliant with these requirements.

4 The schema file

The XML schema specified in the file named OMS_KEY_EXCH_v2_1.xsd defines the standard structure and format to adhere to in order to comply with this specification.

4.1 Explanations for selected names

5 Table 1 gives explanations for selected Type names used in the schema OMS_KEY_EXCH_v2_1.xsd.

Table 1 – Explanations for selected Type names

Type name	Description
CryptoKey	A complex type name representing a DeviceKey, which may contain one or more KeyInterfaces, a KeyMode, a KeyDefinition, and one or more Keys.
CryptoMethods	Provide a selection of standardised crypto methods or cipher suites. One selection defines a set of definition like key length, key derivation function, Key truncation, MAC-length etc. besides the applicable crypto method itself. For the same crypto method (like CMAC with AES 128) several combinations may be defined (like 8 or 16 Byte MAC-length). **NOTE**: These predefined crypto methods allow interoperable devices.
CryptoMethodType	This field describes applicable cryptographic methods (e.g. encryption) or cipher suites (like OMS-SecProfile_B) which shall be applied in context with this key. Either a preconfigured crypto method is selected using <CryptoMethod> or a new crypto method has to be defined using <CustomCryptoMethod>.
Device	A complex type name which represents the device in the xml schema with the DeviceID, the DeviceKey and the VendorDeviceData.
DeviceId	A complex type name which contains the MbusAddress as well as the DinAddress.
DinAddress	A simple string type to declare the format of a 14 digit DIN address, e.g. 3OMS2F12345678.
HexByte	A simple string type to declare a format for a hexadecimal byte representation, e.g. 3FB8.
IdentificationNo	The identification number of the device from EN 13757-3, which is an 8-digit number.
InterfaceTypes	This field selects device interfaces to which the key is applicable.
KeyApplicationType	The key may only be used for a given application like firmware update. This field is used to select one of the predetermined types of applications for which the key shall be applied.
KeyDefinitions	A complex type name which contains information about the KeyType, KeyID and KeyUsage for a specific DeviceKey.
KeyList	A complex type name which represents the actual key contained in EncryptedDataType

Type name	Description
KeyTypes	Different crypto methods require different types of key. A simple encryption uses an encryption key. The OMS-Security profile B applies a master key for both encryption and authorisation. Certificates have to be used for TLS-based cipher methods exchanging a certificate.
KeyUsage	A complex type name containing either a standard key usage expressed by the KeyApplication element or a non-standard key usage expressed by the CustomKeyApplication string.
Manufacturer	ASCII code of the manufacturer ID (three uppercase letters) according to EN 62056-21 or DIN 43863-5, e.g. QDS or KAM. The flag association, UK (http://www.dlms.com) administers the allocation of manufacturer IDs.
MbusAddress	A complex type name which represents a technical M-Bus address.
VendorData	A complex type name for a container of manufacturer specific data.

Table 2 gives explanations for selected Element names used in the schema OMS_KEY_EXCH_v2_1.xsd.

Table 2 – Explanations for selected Element names

Element name	Description
CustomCryptoMethod	Allows the definition of a custom specific crypto method. Use a unique name with a hint to the applied specification (e.g. "CMAC_16BYTE_RFC4493 "). It is recommended to add an explanation field with a web reference to the specification of the crypto standard applied (e.g. "<!-- http://www.rfc-base.org/rfc-4493.html -->").
CryptoMethod	Provides a set of predefined crypto methods. If the selected crypto method for the device is not covered by this set of crypto methods, then use "CustomCryptoMethod".

5

4.2 OMS key exchange schema

The content of the file OMS_KEY_EXCH_v2_1.xsd is listed in Figure 1. The file itself is also attached to this document.

Figure 1 – OMS_KEY_EXCH_v2_1.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://localhost/OMS_KEY_EXCH_v2_1"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://localhost/OMS_KEY_EXCH_v2_1"
elementFormDefault="qualified">
    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="xmldsig-core-schema.xsd"/>
    <xs:import namespace="http://www.w3.org/2001/04/xmlenc#"
schemaLocation="xenc-schema.xsd"/>
    <xs:annotation>
```

```
        <xs:documentation xml:lang="en">
      Distribution format for OMS Meter information.
      Copyright 2014 All rights reserved.
      Version 1.0 changes:
      - Created the document
      Version 1.1 changes:
      - Define KeyType and KeyTypes
      - Define omsKeyExchange element
      - Specify maxOccurs="unbounded" for CryptoKey elements in an
        omsDevice
      - Import xenc-schema.xsd and reference EncryptedKey element
      Version 1.2 changes:
      - Define omsKeyExchange element
      Version 1.3 changes:
      -
      Version 1.4 changes:
      - Move vendor specific data to VendorData element
      - Allow DeviceKey unbounded
      Version 1.5 changes:
      - Add vendor specific data element to omsKeyExchange for order
        specific info
      Version 1.6 changes:
      - Removed mode 1-4
      - Changed addressing
      Version 1.7 changes (Meeting in Mannheim):
      - change KeyTypes
      - change KeyIdx to KeyIndex
      - implement KeyApplication, KeyUsage, KeyID, KeyList
      Version 1.8 changes:
      - CryptoMethods and KeyUsage as choice types
      - implement KeyDefinitons
      Version 1.9 changes:
      - add attribute DeviceIndex
      - Limit vendor specific data like DevideData and OrderData to max
        occurance 1 and call it "Vendor..."
      Version 2.0 changes:
      - MbusAddress is optional in DeviceID
      - change TariffData to TariffSetting
      - add KeyApplicationType "All"
      Version 2.1 changes:
      - Introduce shortStringType
      - Use unsignedShort and unsignedInt instead of nonNegativeInteger
      - Introduce limits on elements KeyInterface, Key, and DeviceKey
        instead of being unbounded
      - Rename Device complex type to DeviceType
      - provide a Signature element
        </xs:documentation>
    </xs:annotation>
    <xs:simpleType name="shortStringType">
        <xs:restriction base="xs:token">
            <xs:maxLength value="127"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="InterfaceTypes">
        <xs:restriction base="xs:string">
            <xs:enumeration value="RemoteWireless"/>
            <xs:enumeration value="RemoteWired"/>
            <xs:enumeration value="Local"/>
            <xs:enumeration value="All"/>
            <xs:enumeration value="Other"/>
        </xs:restriction>
```

```
    </xs:simpleType>
    <!-- Definition of CryptoMethods or CipherSuites -->
    <xs:simpleType name="CryptoMethodType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OMS-SecProfile_A"/>
            <xs:enumeration value="OMS-SecProfile_B"/>
            <xs:enumeration value="OMS-SecProfile_C"/>
            <xs:enumeration value="EN13757-4_ENC-subfield1"/>
            <!--http://oms-group.org/fileadmin/pdf/OMS-
              Spec_Vol2_Primary_v402.pdf-->
            <!--http://oms-group.org/fileadmin/pdf/OMS-
              Spec_Vol2_Primary_v402.pdf-->
            <!--http://oms-group.org/fileadmin/pdf/OMS-
              Spec_Vol2_Primary_v402.pdf-->
            <!--http://www.beuth.de/en/standard/
              din-en-13757-4/198860297-->
            <!--FNN-SecProfile 123-->
            <!--FNN-SecProfile 45-->
            <!--DSMR 4-->
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="CryptoMethods">
        <xs:choice>
            <xs:element name="CryptoMethod" type="tns:CryptoMethodType"/>
            <xs:element name="CustomCryptoMethod"
             type="tns:shortStringType"/>
        </xs:choice>
    </xs:complexType>
    <xs:simpleType name="KeyTypes">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MasterKey"/>
            <xs:enumeration value="Certificate"/>
            <xs:enumeration value="EncKey"/>
            <xs:enumeration value="MacKey"/>
            <xs:enumeration value="KeyEncKey"/>
            <xs:enumeration value="Other"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="KeyApplicationType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="All"/>
            <xs:enumeration value="Data"/>
            <xs:enumeration value="TariffSetting"/>
            <xs:enumeration value="KeyTransfer"/>
            <xs:enumeration value="FirmwareUpdate"/>
            <xs:enumeration value="Prepayment"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="KeyUsage">
        <xs:choice>
            <xs:element name="KeyApplication"
             type="tns:KeyApplicationType"/>
            <xs:element name="CustomKeyApplication"
             type="tns:shortStringType"/>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="KeyDefinitions">
        <xs:sequence>
            <xs:element name="KeyType" type="tns:KeyTypes"/>
            <xs:element name="KeyID" type="xs:unsignedShort"
             minOccurs="0"/>
```

```
            <!-- KeyID represents the exact definiton of the KeyID in the
M-Bus, means ConfigurationField or future AFL MAC ID -->
            <xs:element name="KeyUsage" type="tns:KeyUsage"
             minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="KeyList">
        <xs:sequence>
            <xs:element name="KeyData" type="xenc:EncryptedDataType"/>
        </xs:sequence>
        <xs:attribute name="KeyVersion" type="xs:unsignedShort"
         default="0"/>
        <!-- KeyVersion is 0 as default. This value is applied even if
KeyVersion is not present. -->
    </xs:complexType>
    <xs:complexType name="CryptoKey">
        <xs:sequence>
            <xs:element name="KeyInterface" type="tns:InterfaceTypes"
minOccurs="0" maxOccurs="16"/>
            <xs:element name="KeyMode" type="tns:CryptoMethods"/>
            <xs:element name="KeyDefinition" type="tns:KeyDefinitions"/>
            <xs:element name="Key" type="tns:KeyList" maxOccurs="64"/>
        </xs:sequence>
        <xs:attribute name="KeyIndex" type="xs:unsignedShort"
         use="required"/>
        <!-- KeyIndex is just a running number of all keys of one device
starting with 0 (zero) -->
    </xs:complexType>
    <xs:simpleType name="Manufacturer">
        <xs:restriction base="xs:token">
            <xs:pattern value="[A-Z]{3}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="IdentificationNo">
        <xs:restriction base="xs:token">
            <xs:pattern value="[0-9]{8}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="HexByte">
        <xs:restriction base="xs:token">
            <xs:pattern value="[A-F0-9]{2}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="MbusAddress">
        <xs:sequence>
            <xs:element name="Manufacturer" type="tns:Manufacturer"/>
            <xs:element name="IdentificationNo"
             type="tns:IdentificationNo"/>
            <xs:element name="Version" type="tns:HexByte"/>
            <xs:element name="DeviceType" type="tns:HexByte"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="DinAddress">
        <xs:restriction base="xs:token">
            <xs:pattern value="[A-F0-9]{1}[A-Z]{3}[A-F0-9]{2}[0-9]{8}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="DeviceId">
        <xs:sequence>
            <xs:element name="MbusAddress" type="tns:MbusAddress"
             minOccurs="0"/>
```

```
                <xs:element name="DinAddress" type="tns:DinAddress"/>
            </xs:sequence>
        </xs:complexType>
        <!-- Allows vendors to add any element and/or attribute -->
        <xs:complexType name="VendorData">
            <xs:sequence>
                <xs:any namespace="##any" minOccurs="0"
                 maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <!-- Definition of the OMS meter information -->
        <xs:complexType name="DeviceType">
            <xs:sequence>
                <xs:element name="DeviceId" type="tns:DeviceId"/>
                <xs:element name="DeviceKey" type="tns:CryptoKey"
                 minOccurs="0" maxOccurs="64"/>
                <xs:element name="VendorDeviceData" type="tns:VendorData"
                 minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="DeviceIndex" type="xs:unsignedInt"/>
            <xs:anyAttribute/>
            <!-- DeviceIndex is just a running number of all devices in the
file -->
        </xs:complexType>
        <xs:element name="OMSKeyExchange">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="VendorOrderData" type="tns:VendorData"
                     minOccurs="0"/>
                    <xs:element name="TransportKey"
type="xenc:EncryptedKeyType" minOccurs="0"/>
                    <xs:element name="Device" type="tns:DeviceType"
                     maxOccurs="unbounded">
                        <xs:unique name="uniqueKeyIndex">
                            <xs:selector xpath="tns:DeviceKey"/>
                            <xs:field xpath="@KeyIndex"/>
                        </xs:unique>
                    </xs:element>
                    <xs:element ref="ds:Signature"/>
                </xs:sequence>
            </xs:complexType>
            <xs:unique name="uniqueDeviceIndex">
                <xs:selector xpath="tns:Device"/>
                <xs:field xpath="@DeviceIndex"/>
            </xs:unique>
        </xs:element>
</xs:schema>
```

The OMS_KEY_EXCH_v2_1.xsd schema references an additional schema, namely xenc-schema.xsd (providing the namespace "http://www.w3.org/2001/04/xmlenc#"), which in turn references the schema xmldsig-core-schema.xsd (providing the namespace "http://www.w3.org/2000/09/xmldsig#"). In order to enable XML validation without an Internet connection, these two additional XSD-files need to be included alongside the OMS_KEY_EXCH_v2_1.xsd file. Furthermore, the xenc-schema.xsd schema from W3C (located at http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd) needs to have its import statement modified to point to a local file instead of an Internet address. This modification has been carried out in the description included in Annex B.1 xenc-schema.xsd. The xmldsig-core-schema.xsd file is also included in Annex B.2 for the sake of completeness.

5 Examples

Note that the XSD-files need to be available to the software which parses the XML-file in order to validate them against the schema. The XSD-files may be shipped alongside the XML-file or included in the software, ensuring that systems without an Internet connection are able to validate the XML.

5.1 Example 1: OMS specific data

This example presents an XML-file with mainly mandatory information which complies with the OMS-Specification. The key elements are encrypted with a key which is supplied out-of-band.

The content of the file OMS_Example1_v2_1.xml is listed in Figure 2. The file itself is also attached to this document.

Figure 2 – OMS_Example1_v2_1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<OMSKeyExchange xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://localhost/OMS_KEY_EXCH_v2_1">
  <Device>
    <DeviceId>
      <MbusAddress>
        <Manufacturer>DIN</Manufacturer>
        <IdentificationNo>00001111</IdentificationNo>
        <Version>1E</Version>
        <DeviceType>04</DeviceType>
      </MbusAddress>
      <DinAddress>6DIN1E00001111</DinAddress>
    </DeviceId>
    <DeviceKey KeyIndex="0">
      <KeyInterface>RemoteWireless</KeyInterface>
      <KeyMode>
        <CryptoMethod>OMS-SecProfile_A</CryptoMethod>
      </KeyMode>
      <KeyDefinition>
        <KeyType>EncKey</KeyType>
        <KeyUsage>
          <KeyApplication>Data</KeyApplication>
        </KeyUsage>
      </KeyDefinition>
      <Key KeyVersion="1">
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Preset Key from Factory</KeyName>
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>Hlb7hqyFbZNcsi/3FOgFwVbv5l93qKVT</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
      <Key KeyVersion="2">
```

```
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Replacement Key - change with Service tool</KeyName>
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>cmf/4BzRxrd1qro+LsorlTy0wLmMspRg</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
    </DeviceKey>
  </Device>
  <Device>
    <DeviceId>
      <MbusAddress>
        <Manufacturer>DIN</Manufacturer>
        <IdentificationNo>00002222</IdentificationNo>
        <Version>00</Version>
        <DeviceType>03</DeviceType>
      </MbusAddress>
      <DinAddress>7DIN0000002222</DinAddress>
    </DeviceId>
    <DeviceKey KeyIndex="0">
      <KeyInterface>RemoteWireless</KeyInterface>
      <KeyMode>
        <CryptoMethod>OMS-SecProfile_B</CryptoMethod>
      </KeyMode>
      <KeyDefinition>
        <KeyType>MasterKey</KeyType>
        <KeyUsage>
          <KeyApplication>Data</KeyApplication>
        </KeyUsage>
      </KeyDefinition>
      <Key>
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>3RhdU0deKNiYGdb0HKbx+U48ZDqre/uL</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
    </DeviceKey>
    <DeviceKey KeyIndex="1">
      <KeyInterface>Local</KeyInterface>
      <KeyMode>
        <CustomCryptoMethod>a user defined Crypto Method
        </CustomCryptoMethod>
      </KeyMode>
      <KeyDefinition>
        <KeyType>MasterKey</KeyType>
        <KeyUsage>
          <KeyApplication>Data</KeyApplication>
        </KeyUsage>
```

```
      </KeyDefinition>
      <Key>
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>3RhdU0deKNiYGdb0HKbx+U48ZDqre/uL</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
    </DeviceKey>
  </Device>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
       Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <SignatureMethod
       Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
      <Reference URI="">
        <Transforms>
          <Transform
           Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
                      signature" />
          <Transform
           Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
        </Transforms>
        <DigestMethod
         Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
        <DigestValue>sRLVuThdu7fRIYsK04iJecm47SbhSDpoKLACd7rQmvY=
        </DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
      ORARLtQ3mwgOikg2VkqSJFITU7Ayz0Mb+FSFNZI54NE3fwZo7SGXPhCSEaRFo7nfRW2E
      eBtoL7guRqcugtOOXz5eebbRZLqyX17sUZyJQXRN+NirIzRD9H6hBPX+Gmn6JFoGHiGK
      diIdYW2zveQDjVzf/L3KECNIXWpFHgUNqqQ=
    </SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>
          <Modulus>
            pri1OecbYESZ9IiXZSrq1oqh2SuLvV3VIP3XRoOKFHtAgPSDP2WAPnAEli7g
            b+bKrWDf+A8+QcEV3uWvRQa0hfMz0t9ssyGn9jtQzGCx1yYjufz4IpkMtEoe
            ejgzZskW1gB+fBV+OKYNnF42qVfBPsggWhSTVqII+mIebGx+nl0=
          </Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
</OMSKeyExchange>
```

This example is generated and parsed using the example implementation in C# listed in Annex B.4 program.cs.ex1. This implementation is meant as an example only, and it is not to be used in production code.

The plaintext keys transported in this example are:

- 0x11335577113355771133557711335577
- 0x22446688224466882244668822446688
- 0xAACCEE00AACCEE00AACCEE00AACCEE00
- 0xBBDDFF11BBDDFF11BBDDFF11BBDDFF11

The SessionKey is supplied out of band and is noted below in C#-Syntax:

```
{0xDE, 0xAD, 0xBE, 0xEF, 0x00, 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xCA, 0xFE, 0xBA, 0xBE, 0x00}.
```

5.2 Example 2: Manufacturer specific data

Manufacturer specific data may be included as child elements in the DeviceData element (scope is the specific device) or in the OrderData element (scope are all devices in the XML document). If needed, vendors may include their own XSD schemata in addition to the OMS_KEY_EXCH_v2_1.xsd schema.

In this example, the key named “SessionKey” is used for decrypting the Key elements. This key is included within the XML-file in the TransportKey element. The “SessionKey” key has itself been encrypted with an asymmetric key named “KeyID”.

The choice of the encryption algorithm is in compliance with the requirements stated in Section 3.

The schema file to verify the vendor data is provided in Annex B.3 MyVendorNameSpace.xsd.

The content of the file OMS_Example2_v2_1.xml is listed in Figure 3. The file itself is also attached to this document.

Figure 3 – OMS_Example2_v2_1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<OMSKeyExchange xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://localhost/OMS_KEY_EXCH_v2_1">
  <VendorOrderData>
    <ordernumber xmlns="MyVendorNamespace">12345</ordernumber>
    <customer xmlns="MyVendorNamespace">Customer ABC</customer>
    <street xmlns="MyVendorNamespace">West Road 123</street>
    <zip xmlns="MyVendorNamespace">12345</zip>
    <city xmlns="MyVendorNamespace">Middle City</city>
  </VendorOrderData>
  <TransportKey Id="KeyId">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
    oaep-mgf1p" xmlns="http://www.w3.org/2001/04/xmlenc#">
      <KeySize>3072</KeySize>
    </EncryptionMethod>
    <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
      <CipherValue>
        clUv2B9b1Wn+zZBCyy5b0XujJcYgxGAWrdTNFC7Cp/xB3t740n4GsZSvVzvXFTMGn
        neXKgv3nDC2vUaUPdJkrAiKNzvHCYK6MujnWJJKe/SukxXIW1ZC87tQ+NnoMJHUy8
        OHBQroeunHhCSQBw4fUoIif45OJrnYD0IW1Wg9mlX7nvwnxKQFbZKQMACA6ZwUFtA
        el87mRx2iTd+snCNzmqEog7wv+SixX2pP8WIcIXfNS6LclosT03DOVRNeKZyQwRSI
        wgtkIzV5y8gzv5iaXEGQ5pW2SgsU/aWY6PKjfpdmfrG0cuvipGEfq9hb0ZE7YWGNC
        L4W3G+MyHKH2yqyY46dRbjm3ifg3LqoSyRoNO5qfMKj6h1y+PrUArS3Ea+2oAXxoo
        4FveWQ3SXwIrdWMm7xteSqWrY5xMPG7trL8KSeXUaA4lv54Lf+h60OBjKg2C0QJPu
        fiPFZTfBEY7gcRYa78Ybzbiz+K6z6vxBWMmzEusreHqGMrq4fHtLGs7p5
      </CipherValue>
    </CipherData>
```

```
    <CarriedKeyName xmlns="http://www.w3.org/2001/04/xmlenc#">SessionKey
    </CarriedKeyName>
  </TransportKey>
  <Device DeviceIndex="1">
    <DeviceId>
      <MbusAddress>
        <Manufacturer>XXX</Manufacturer>
        <IdentificationNo>33334444</IdentificationNo>
        <Version>0A</Version>
        <DeviceType>02</DeviceType>
      </MbusAddress>
      <DinAddress>1XXX0A33334444</DinAddress>
    </DeviceId>
    <DeviceKey KeyIndex="0">
      <KeyInterface>RemoteWireless</KeyInterface>
      <KeyMode>
        <CustomCryptoMethod>RFC6188_AES-192_CTR</CustomCryptoMethod>
      </KeyMode>
      <KeyDefinition>
        <KeyType>EncKey</KeyType>
        <KeyID>1</KeyID>
        <KeyUsage>
          <KeyApplication>Data</KeyApplication>
        </KeyUsage>
      </KeyDefinition>
      <Key>
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Valid_for_2015</KeyName>
            <RetrievalMethod URI="#SessionKey"
             Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>wwpTGnRN5ihNOXMq6U8m7ALJdM4ISPLN</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
      <Key KeyVersion="1">
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Valid_for_2016</KeyName>
            <RetrievalMethod URI="#SessionKey"
             Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>Ukl1KRUPlfiEdntveAOhOH5XVjzJIHuS</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
      <Key KeyVersion="2">
        <KeyData>
```

```
            <EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
             xmlns="http://www.w3.org/2001/04/xmlenc#">
              <KeySize>128</KeySize>
            </EncryptionMethod>
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>Valid_for_2017</KeyName>
              <RetrievalMethod URI="#SessionKey"
               Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
            </KeyInfo>
            <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
              <CipherValue>Ns2qEjvrb3phMD7Ax6cZ7h7sznGrfJ2r</CipherValue>
            </CipherData>
          </KeyData>
        </Key>
      </DeviceKey>
      <DeviceKey KeyIndex="1">
        <KeyInterface>RemoteWireless</KeyInterface>
        <KeyMode>
          <CustomCryptoMethod>RFC6188_AES-192_CTR</CustomCryptoMethod>
        </KeyMode>
        <KeyDefinition>
          <KeyType>EncKey</KeyType>
          <KeyID>2</KeyID>
          <KeyUsage>
            <KeyApplication>FirmwareUpdate</KeyApplication>
          </KeyUsage>
        </KeyDefinition>
        <Key>
          <KeyData>
            <EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
             xmlns="http://www.w3.org/2001/04/xmlenc#">
              <KeySize>128</KeySize>
            </EncryptionMethod>
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>Valid_for_2017</KeyName>
              <RetrievalMethod URI="#SessionKey"
               Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
            </KeyInfo>
            <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
              <CipherValue>jlF1fTjlteUniiyMm9ukIaPlQs71SNdw</CipherValue>
            </CipherData>
          </KeyData>
        </Key>
      </DeviceKey>
      <DeviceKey KeyIndex="2">
        <KeyInterface>RemoteWireless</KeyInterface>
        <KeyMode>
          <CustomCryptoMethod>RFC618_AES-192_CTR</CustomCryptoMethod>
        </KeyMode>
        <KeyDefinition>
          <KeyType>EncKey</KeyType>
          <KeyID>3</KeyID>
          <KeyUsage>
            <CustomKeyApplication>Backup Key</CustomKeyApplication>
          </KeyUsage>
        </KeyDefinition>
        <Key>
          <KeyData>
            <EncryptionMethod
```

```
            Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
            xmlns="http://www.w3.org/2001/04/xmlenc#">
              <KeySize>128</KeySize>
            </EncryptionMethod>
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>Valid_only_once</KeyName>
              <RetrievalMethod URI="#SessionKey"
               Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
            </KeyInfo>
            <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
              <CipherValue>60Kj/g6oPH77JlHMyqjJF0/f+b4y8E19</CipherValue>
            </CipherData>
          </KeyData>
        </Key>
      </DeviceKey>
      <DeviceKey KeyIndex="3">
        <KeyInterface>RemoteWireless</KeyInterface>
        <KeyMode>
          <CustomCryptoMethod>RFC4493_CMAC_AES128_trunc_4Byte
          </CustomCryptoMethod>
        </KeyMode>
        <KeyDefinition>
          <KeyType>MacKey</KeyType>
          <KeyID>1</KeyID>
          <KeyUsage>
            <CustomKeyApplication>MAC_For_All_EncModes
            </CustomKeyApplication>
          </KeyUsage>
        </KeyDefinition>
        <Key>
          <KeyData>
            <EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
             xmlns="http://www.w3.org/2001/04/xmlenc#">
              <KeySize>128</KeySize>
            </EncryptionMethod>
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>MacKey</KeyName>
              <RetrievalMethod URI="#SessionKey"
               Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
            </KeyInfo>
            <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
              <CipherValue>3RhdU0deKNiYGdb0HKbx+U48ZDqre/uL</CipherValue>
            </CipherData>
          </KeyData>
        </Key>
      </DeviceKey>
      <VendorDeviceData>
        <meterinfo xmlns="MyVendorNamespace">
          <MeterName>Electricity meter</MeterName>
          <ConsumptionType>kWh</ConsumptionType>
          <IMEI>357805023984942</IMEI>
        </meterinfo>
      </VendorDeviceData>
    </Device>
    <Device DeviceIndex="2">
      <DeviceId>
        <MbusAddress>
          <Manufacturer>DIN</Manufacturer>
          <IdentificationNo>00002222</IdentificationNo>
          <Version>00</Version>
```

```
        <DeviceType>03</DeviceType>
      </MbusAddress>
      <DinAddress>7DIN0000002222</DinAddress>
    </DeviceId>
    <DeviceKey KeyIndex="0">
      <KeyInterface>RemoteWireless</KeyInterface>
      <KeyMode>
        <CryptoMethod>OMS-SecProfile_A</CryptoMethod>
      </KeyMode>
      <KeyDefinition>
        <KeyType>EncKey</KeyType>
        <KeyUsage>
          <KeyApplication>Data</KeyApplication>
        </KeyUsage>
      </KeyDefinition>
      <Key KeyVersion="1">
        <KeyData>
          <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
           xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Preset Key from Factory</KeyName>
            <RetrievalMethod URI="#SessionKey"
             Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>/+1d03RQ1LwqaLbuQdQYGY+wrZpxjjMT</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
      <Key KeyVersion="2">
        <KeyData>
                              <EncryptionMethod
           Algorithm="http://www.w3.org/2001/04/xmlenc#kw-aes128"
                              xmlns="http://www.w3.org/2001/04/xmlenc#">
            <KeySize>128</KeySize>
          </EncryptionMethod>
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Replacement Key - change with Service tool</KeyName>
            <RetrievalMethod URI="#SessionKey"
             Type="http://www.w3.org/2001/04/xmlenc#EncryptedKey" />
          </KeyInfo>
          <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">
            <CipherValue>ZqlXchQofxchjzNRD+vWuFnXRUig9+9r</CipherValue>
          </CipherData>
        </KeyData>
      </Key>
    </DeviceKey>
  </Device>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
       Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <SignatureMethod
       Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
      <Reference URI="">
        <Transforms>
          <Transform
```

```
            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
                       signature" />
          <Transform
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
        </Transforms>
        <DigestMethod
         Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
        <DigestValue>Mhc/YjnHnVRNHM0mGHaGsuGABK9HvCdp2P/dWkED3Wk=
        </DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>
      EVbxVWeUZV12IUSJOT0JxlNZtjYtfnvfSL81NF8+ByECpIAiMN/i5RH7Eq06kVvjYLNC
      LammFOGmwGhkJkBBC0IzDCa/cVbznqfUX6dbhg1xE++HnGN8baUPp7SvX3VAtFplxjA9
      gqQh+Wk9lWQ6CwBRXTQTGv1DT/zwI6xP67Y=
    </SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>
          <Modulus>
            pri1OecbYESZ9IiXZSrq1oqh2SuLvV3VIP3XRoOKFHtAgPSDP2WAPnAEli7g
            b+bKrWDf+A8+QcEV3uWvRQa0hfMz0t9ssyGn9jtQzGCx1yYjufz4IpkMtEoe
            ejgzZskW1gB+fBV+OKYNnF42qVfBPsggWhSTVqII+mIebGx+nl0=
          </Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
</OMSKeyExchange>
```

This example is generated and parsed using the example implementation in C# listed in Annex B.5 program.cs.ex2. This implementation is meant as an example only, and it is not to be used in production code.

The plaintext keys transported in this example are:

- 0x123456789ABCDEF011223344556677
- 0x22446688AACCEF0112233445566778
- 0x32547698BADCFF1213243546576879
- 0x426486A8CAED0F231425364758697A
- 0x527496B8DAFD1F3415263748596A7B
- 0x6284A6C8EB0D2F45162738495A6B7C
- 0xE30527496B8DAFCD1E2F4051627384
- 0xF31537597B9DBFDE1F304152637485

The example makes use of the RSA private/public key pair listed in Annex B.7 rsa3072_pu.xml and Annex B.6 rsa3072_pr.xml.

Annex A: Requirements to OMS-Meter manufacturers

- A Manufacturer of OMS-devices should apply this XML-Key-File to share the secret key-information with the operator of devices carefully.
- The Manufacturer should use the predefined “CryptoMethodType” for communication interfaces according to the OMS Specifications. Table A.1 describes applicable crypto parameters depending on the selected security profile.

Table A.1 – Applicable key types for OMS Security profiles

Type Name	Security Profile A	Security Profile B	Security Profile C
<CryptoMethod>	“OMS-SecProfile_A”	“OMS-SecProfile_B”	“OMS-SecProfile_C”
<KeyType>	“EncKey”	“MasterKey”	“Certificate”; “MasterKey”

- Each meter should support at least “Data” or “All” <KeyApplication>.
- Each OMS-device should provide both <MbusAddress> and <DinAddress>.

Annex B: Useful files

B.1 xenc-schema.xsd

The content of the file xenc-schema.xsd is listed below. The file itself is also attached to this document.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE schema PUBLIC "-//W3C//DTD XMLSchema 200102//EN"
 "http://www.w3.org/2001/XMLSchema.dtd"
 [
   <!ATTLIST schema
     xmlns:xenc CDATA #FIXED 'http://www.w3.org/2001/04/xmlenc#'
     xmlns:ds CDATA #FIXED 'http://www.w3.org/2000/09/xmldsig#'>
   <!ENTITY xenc 'http://www.w3.org/2001/04/xmlenc#'>
   <!ENTITY % p ''>
   <!ENTITY % s ''>
 ]>

<schema xmlns='http://www.w3.org/2001/XMLSchema' version='1.0'
        xmlns:xenc='http://www.w3.org/2001/04/xmlenc#'
        xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
        targetNamespace='http://www.w3.org/2001/04/xmlenc#'
        elementFormDefault='qualified'>

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

  <complexType name='EncryptedType' abstract='true'>
    <sequence>
      <element name='EncryptionMethod' type='xenc:EncryptionMethodType'
       minOccurs='0'/>
      <element ref='ds:KeyInfo' minOccurs='0'/>
      <element ref='xenc:CipherData'/>
      <element ref='xenc:EncryptionProperties' minOccurs='0'/>
    </sequence>
    <attribute name='Id' type='ID' use='optional'/>
    <attribute name='Type' type='anyURI' use='optional'/>
    <attribute name='MimeType' type='string' use='optional'/>
    <attribute name='Encoding' type='anyURI' use='optional'/>
  </complexType>

  <complexType name='EncryptionMethodType' mixed='true'>
    <sequence>
      <element name='KeySize' minOccurs='0' type='xenc:KeySizeType'/>
      <element name='OAEPparams' minOccurs='0' type='base64Binary'/>
      <any namespace='##other' minOccurs='0' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='Algorithm' type='anyURI' use='required'/>
  </complexType>

    <simpleType name='KeySizeType'>
      <restriction base="integer"/>
    </simpleType>

  <element name='CipherData' type='xenc:CipherDataType'/>
  <complexType name='CipherDataType'>
    <choice>
      <element name='CipherValue' type='base64Binary'/>
```

```
      <element ref='xenc:CipherReference'/>
    </choice>
  </complexType>

  <element name='CipherReference' type='xenc:CipherReferenceType'/>
  <complexType name='CipherReferenceType'>
    <choice>
      <element name='Transforms'
       type='xenc:TransformsType' minOccurs='0'/>
    </choice>
    <attribute name='URI' type='anyURI' use='required'/>
  </complexType>

  <complexType name='TransformsType'>
    <sequence>
      <element ref='ds:Transform' maxOccurs='unbounded'/>
    </sequence>
    </complexType>

  <element name='EncryptedData' type='xenc:EncryptedDataType'/>
  <complexType name='EncryptedDataType'>
    <complexContent>
      <extension base='xenc:EncryptedType'>
      </extension>
    </complexContent>
    </complexType>

  <!-- Children of ds:KeyInfo -->
  <element name='EncryptedKey' type='xenc:EncryptedKeyType'/>
  <complexType name='EncryptedKeyType'>
    <complexContent>
      <extension base='xenc:EncryptedType'>
        <sequence>
          <element ref='xenc:ReferenceList' minOccurs='0'/>
          <element name='CarriedKeyName' type='string' minOccurs='0'/>
        </sequence>
        <attribute name='Recipient' type='string' use='optional'/>
      </extension>
    </complexContent>
  </complexType>

  <element name="AgreementMethod" type="xenc:AgreementMethodType"/>
  <complexType name="AgreementMethodType" mixed="true">
    <sequence>
      <element name="KA-Nonce" minOccurs="0" type="base64Binary"/>
      <!-- <element ref="ds:DigestMethod" minOccurs="0"/> -->
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      <element name="OriginatorKeyInfo" minOccurs="0"
       type="ds:KeyInfoType"/>
      <element name="RecipientKeyInfo" minOccurs="0"
       type="ds:KeyInfoType"/>
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>
  <!-- End Children of ds:KeyInfo -->

  <element name='ReferenceList'>
    <complexType>
      <choice minOccurs='1' maxOccurs='unbounded'>
        <element name='DataReference' type='xenc:ReferenceType'/>
        <element name='KeyReference' type='xenc:ReferenceType'/>
```

```
      </choice>
    </complexType>
  </element>

  <complexType name='ReferenceType'>
    <sequence>
      <any namespace='##other' minOccurs='0' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='URI' type='anyURI' use='required'/>
  </complexType>

  <element name='EncryptionProperties'
   type='xenc:EncryptionPropertiesType'/>
  <complexType name='EncryptionPropertiesType'>
    <sequence>
      <element ref='xenc:EncryptionProperty' maxOccurs='unbounded'/>
    </sequence>
    <attribute name='Id' type='ID' use='optional'/>
  </complexType>

  <element name='EncryptionProperty' type='xenc:EncryptionPropertyType'/>
  <complexType name='EncryptionPropertyType' mixed='true'>
    <choice maxOccurs='unbounded'>
      <any namespace='##other' processContents='lax'/>
    </choice>
    <attribute name='Target' type='anyURI' use='optional'/>
    <attribute name='Id' type='ID' use='optional'/>
    <anyAttribute namespace="http://www.w3.org/XML/1998/namespace"/>
  </complexType>

</schema>
```

B.2 xmldsig-core-schema.xsd

The content of the file xmldsig-core-schema.xsd is listed below. The file itself is also attached to this document.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE schema
  PUBLIC "-//W3C//DTD XMLSchema 200102//EN"
 "http://www.w3.org/2001/XMLSchema.dtd"
 [
   <!ATTLIST schema
     xmlns:ds CDATA #FIXED 'http://www.w3.org/2000/09/xmldsig#'>
   <!ENTITY dsig 'http://www.w3.org/2000/09/xmldsig#'>
   <!ENTITY % p ''>
   <!ENTITY % s ''>
 ]>

<!-- Schema for XML Signatures
    http://www.w3.org/2000/09/xmldsig#
    $Revision: 1.1 $ on $Date: 2002/02/08 20:32:26 $ by $Author: reagle $

    Copyright 2001 The Internet Society and W3C (Massachusetts Institute
    of Technology, Institut National de Recherche en Informatique et en
    Automatique, Keio University). All Rights Reserved.
    http://www.w3.org/Consortium/Legal/

    This document is governed by the W3C Software License [1] as described
    in the FAQ [2].
```

```
    [1] http://www.w3.org/Consortium/Legal/copyright-software-19980720
    [2] http://www.w3.org/Consortium/Legal/IPR-FAQ-20000620.html#DTD
-->

<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        targetNamespace="http://www.w3.org/2000/09/xmldsig#"
        version="0.1" elementFormDefault="qualified">

<!-- Basic Types Defined for Signatures -->
  <simpleType name="CryptoBinary">
    <restriction base="base64Binary">
    </restriction>
  </simpleType>

  <!-- Start Signature -->
  <element name="Signature" type="ds:SignatureType"/>
  <complexType name="SignatureType">
    <sequence>
      <element ref="ds:SignedInfo"/>
      <element ref="ds:SignatureValue"/>
      <element ref="ds:KeyInfo" minOccurs="0"/>
      <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

  <element name="SignatureValue" type="ds:SignatureValueType"/>
  <complexType name="SignatureValueType">
    <simpleContent>
      <extension base="base64Binary">
        <attribute name="Id" type="ID" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

    <!-- Start SignedInfo -->
  <element name="SignedInfo" type="ds:SignedInfoType"/>
  <complexType name="SignedInfoType">
    <sequence>
      <element ref="ds:CanonicalizationMethod"/>
      <element ref="ds:SignatureMethod"/>
      <element ref="ds:Reference" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

  <element name="CanonicalizationMethod"
   type="ds:CanonicalizationMethodType"/>
  <complexType name="CanonicalizationMethodType" mixed="true">
    <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
      <!-- (0,unbounded) elements from (1,1) namespace -->
    </sequence>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>

  <element name="SignatureMethod" type="ds:SignatureMethodType"/>
  <complexType name="SignatureMethodType" mixed="true">
    <sequence>
      <element name="HMACOutputLength" minOccurs="0"
```

```
       type="ds:HMACOutputLengthType"/>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      <!-- (0,unbounded) elements from (1,1) external namespace -->
   </sequence>
   <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>

     <!-- Start Reference -->
<element name="Reference" type="ds:ReferenceType"/>
<complexType name="ReferenceType">
   <sequence>
     <element ref="ds:Transforms" minOccurs="0"/>
     <element ref="ds:DigestMethod"/>
     <element ref="ds:DigestValue"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional"/>
   <attribute name="URI" type="anyURI" use="optional"/>
   <attribute name="Type" type="anyURI" use="optional"/>
</complexType>

<element name="Transforms" type="ds:TransformsType"/>
<complexType name="TransformsType">
   <sequence>
     <element ref="ds:Transform" maxOccurs="unbounded"/>
   </sequence>
</complexType>

<element name="Transform" type="ds:TransformType"/>
<complexType name="TransformType" mixed="true">
   <choice minOccurs="0" maxOccurs="unbounded">
     <any namespace="##other" processContents="lax"/>
     <!-- (1,1) elements from (0,unbounded) namespaces -->
     <element name="XPath" type="string"/>
   </choice>
   <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>
     <!-- End Reference -->

<element name="DigestMethod" type="ds:DigestMethodType"/>
<complexType name="DigestMethodType" mixed="true">
   <sequence>
     <any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Algorithm" type="anyURI" use="required"/>
</complexType>

<element name="DigestValue" type="ds:DigestValueType"/>
<simpleType name="DigestValueType">
   <restriction base="base64Binary"/>
</simpleType>
   <!-- End SignedInfo -->

   <!-- Start KeyInfo -->
<element name="KeyInfo" type="ds:KeyInfoType"/>
<complexType name="KeyInfoType" mixed="true">
   <choice maxOccurs="unbounded">
     <element ref="ds:KeyName"/>
     <element ref="ds:KeyValue"/>
     <element ref="ds:RetrievalMethod"/>
     <element ref="ds:X509Data"/>
```

```
      <element ref="ds:PGPData"/>
      <element ref="ds:SPKIData"/>
      <element ref="ds:MgmtData"/>
      <any processContents="lax" namespace="##other"/>
      <!-- (1,1) elements from (0,unbounded) namespaces -->
    </choice>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>

  <element name="KeyName" type="string"/>
  <element name="MgmtData" type="string"/>

  <element name="KeyValue" type="ds:KeyValueType"/>
  <complexType name="KeyValueType" mixed="true">
    <choice>
      <element ref="ds:DSAKeyValue"/>
      <element ref="ds:RSAKeyValue"/>
      <any namespace="##other" processContents="lax"/>
    </choice>
  </complexType>

  <element name="RetrievalMethod" type="ds:RetrievalMethodType"/>
  <complexType name="RetrievalMethodType">
    <sequence>
      <element ref="ds:Transforms" minOccurs="0"/>
    </sequence>
    <attribute name="URI" type="anyURI"/>
    <attribute name="Type" type="anyURI" use="optional"/>
  </complexType>

      <!-- Start X509Data -->
  <element name="X509Data" type="ds:X509DataType"/>
  <complexType name="X509DataType">
    <sequence maxOccurs="unbounded">
      <choice>
        <element name="X509IssuerSerial" type="ds:X509IssuerSerialType"/>
        <element name="X509SKI" type="base64Binary"/>
        <element name="X509SubjectName" type="string"/>
        <element name="X509Certificate" type="base64Binary"/>
        <element name="X509CRL" type="base64Binary"/>
        <any namespace="##other" processContents="lax"/>
      </choice>
    </sequence>
  </complexType>

  <complexType name="X509IssuerSerialType">
    <sequence>
      <element name="X509IssuerName" type="string"/>
      <element name="X509SerialNumber" type="integer"/>
    </sequence>
  </complexType>
      <!-- End X509Data -->

      <!-- Begin PGPData -->
  <element name="PGPData" type="ds:PGPDataType"/>
  <complexType name="PGPDataType">
    <choice>
      <sequence>
        <element name="PGPKeyID" type="base64Binary"/>
        <element name="PGPKeyPacket" type="base64Binary" minOccurs="0"/>
        <any namespace="##other" processContents="lax" minOccurs="0"
```

```
          maxOccurs="unbounded"/>
      </sequence>
      <sequence>
        <element name="PGPKeyPacket" type="base64Binary"/>
        <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
      </sequence>
    </choice>
</complexType>
     <!-- End PGPData -->

     <!-- Begin SPKIData -->
<element name="SPKIData" type="ds:SPKIDataType"/>
<complexType name="SPKIDataType">
   <sequence maxOccurs="unbounded">
     <element name="SPKISexp" type="base64Binary"/>
     <any namespace="##other" processContents="lax" minOccurs="0"/>
   </sequence>
</complexType>
     <!-- End SPKIData -->
   <!-- End KeyInfo -->

   <!-- Start Object (Manifest, SignatureProperty) -->
<element name="Object" type="ds:ObjectType"/>
<complexType name="ObjectType" mixed="true">
   <sequence minOccurs="0" maxOccurs="unbounded">
     <any namespace="##any" processContents="lax"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional"/>
   <attribute name="MimeType" type="string" use="optional"/>
    <!-- add a grep facet -->
   <attribute name="Encoding" type="anyURI" use="optional"/>
</complexType>

<element name="Manifest" type="ds:ManifestType"/>
<complexType name="ManifestType">
   <sequence>
     <element ref="ds:Reference" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional"/>
</complexType>

<element name="SignatureProperties" type="ds:SignaturePropertiesType"/>
<complexType name="SignaturePropertiesType">
   <sequence>
     <element ref="ds:SignatureProperty" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="Id" type="ID" use="optional"/>
</complexType>

<element name="SignatureProperty" type="ds:SignaturePropertyType"/>
<complexType name="SignaturePropertyType" mixed="true">
   <choice maxOccurs="unbounded">
     <any namespace="##other" processContents="lax"/>
     <!-- (1,1) elements from (1,unbounded) namespaces -->
   </choice>
   <attribute name="Target" type="anyURI" use="required"/>
   <attribute name="Id" type="ID" use="optional"/>
</complexType>
   <!-- End Object (Manifest, SignatureProperty) -->
```

```
    <!-- Start Algorithm Parameters -->
  <simpleType name="HMACOutputLengthType">
    <restriction base="integer"/>
  </simpleType>

      <!-- Start KeyValue Element-types -->
  <element name="DSAKeyValue" type="ds:DSAKeyValueType"/>
  <complexType name="DSAKeyValueType">
    <sequence>
      <sequence minOccurs="0">
        <element name="P" type="ds:CryptoBinary"/>
        <element name="Q" type="ds:CryptoBinary"/>
      </sequence>
      <element name="G" type="ds:CryptoBinary" minOccurs="0"/>
      <element name="Y" type="ds:CryptoBinary"/>
      <element name="J" type="ds:CryptoBinary" minOccurs="0"/>
      <sequence minOccurs="0">
        <element name="Seed" type="ds:CryptoBinary"/>
        <element name="PgenCounter" type="ds:CryptoBinary"/>
      </sequence>
    </sequence>
  </complexType>

  <element name="RSAKeyValue" type="ds:RSAKeyValueType"/>
  <complexType name="RSAKeyValueType">
    <sequence>
      <element name="Modulus" type="ds:CryptoBinary"/>
      <element name="Exponent" type="ds:CryptoBinary"/>
    </sequence>
  </complexType>
      <!-- End KeyValue Element-types -->
    <!-- End Algorithm Parameters -->
  <!-- End Signature -->

</schema>
```

B.3 MyVendorNameSpace.xsd

The content of the file MyVendorNameSpace.xsd is listed below. The file itself is also attached to this document.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://localhost/MyVendorNameSpace"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://localhost/MyVendorNameSpace"
           elementFormDefault="qualified">
  <xs:element name="ordernumber" type="xs:token"/>
  <xs:element name="customer" type="xs:token"/>
  <xs:element name="street" type="xs:token"/>
  <xs:element name="zip" type="xs:token"/>
  <xs:element name="city" type="xs:token"/>
  <xs:element name="MeterName" type="xs:token"/>
  <xs:element name="ConsumptionType" type="xs:token"/>
  <xs:element name="IMEI" type="xs:token"/>
</xs:schema>
```

B.4 program.cs.ex1

This implementation is meant as an example only, and it is not to be used in production code.

The content of the file program.cs.ex1 is listed below. The file itself is also attached to this document.

```
/* Copyright 2015 Landis + Gyr.
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or any
   later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>. */
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;
using System.Xml;
using System.Xml.Serialization;
using System.Xml.XPath;
using RFC3394;
namespace oms1
{
   class Program
   {
      static void Main(string[] args)
      {
         int ex = 0;
         byte[] TheKey = {0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0x00,0x11,0x22,0x33,
            0x44,0x55,0x66,0x77};

         byte[, ,] Exkey = {{
   {0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77},
   {0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88},
   {0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00},
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11},
   {0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88},
   {0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00},
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11},
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11}}
,
   {{0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77}
,
   {0x22,0x44,0x66,0x88,0xAA,0xCC,0xEF,0x01,0x12,0x23,0x34,0x45,0x56,0x67,0x78,0x89},
   {0x32,0x54,0x76,0x98,0xBA,0xDC,0xFF,0x12,0x13,0x24,0x35,0x46,0x57,0x68,0x79,0x8A},
   {0x42,0x64,0x86,0xA8,0xCA,0xED,0x0F,0x23,0x14,0x25,0x36,0x47,0x58,0x69,0x7A,0x8B},
   {0x52,0x74,0x96,0xB8,0xDA,0xFD,0x1F,0x34,0x15,0x26,0x37,0x48,0x59,0x6A,0x7B,0x8C},
   {0x62,0x84,0xA6,0xC8,0xEB,0x0D,0x2F,0x45,0x16,0x27,0x38,0x49,0x5A,0x6B,0x7C,0x8D},
   {0xE3,0x05,0x27,0x49,0x6B,0x8D,0xAF,0xCD,0x1E,0x2F,0x40,0x51,0x62,0x73,0x84,0x8E},
   {0xF3,0x15,0x37,0x59,0x7B,0x9D,0xBF,0xDE,0x1F,0x30,0x41,0x52,0x63,0x74,0x85,0x8F}}

      };
```

```
        byte[] TheSessionKey = {0xDE,0xAD,0xBE,0xEF,0x00,0x12,0x34,0x56,0x78,0x9A,
           0xBC,0xCA,0xFE,0xBA,0xBE,0x00};
        String Filename = "mbus1.xml";

        XmlSerializer serializer = new XmlSerializer(typeof(OMSKeyExchange));
        //Create XML File objects
        OMSKeyExchange clsOMS = new OMSKeyExchange();
        StreamWriter writer = new StreamWriter(Filename);
      // a key container.
        const int iMbusLen = 16;
        clsRFC3394 wrap = new clsRFC3394(TheSessionKey);
        EncryptionMethodType kem = new EncryptionMethodType();
        if (TheSessionKey.Length == 16)
        {
           kem.Algorithm = "http://www.w3.org/2001/04/xmlenc#kw-aes128";
           kem.KeySize = "128";
        }
        else if (TheSessionKey.Length == 32)
        {
           kem.Algorithm = "http://www.w3.org/2001/04/xmlenc#kw-aes256";
           kem.KeySize = "256";
        }
        else
        {
           Console.Write("Wrong KEK length: " + TheSessionKey.Length);
           return;
        }

        //=============================================================================
        // First device
        CryptoMethodType CryptoMethod2 = new CryptoMethodType();

        CryptoMethod2 = CryptoMethodType.OMSSecProfile_A;

        DeviceType[] Devices = new DeviceType[2];
        DeviceType Device = new DeviceType();
        DeviceId DeviceId = new DeviceId();
        DeviceId.DinAddress = "6DIN1E00001111";

        MbusAddress MbusAddress = new MbusAddress();
        MbusAddress.Manufacturer = "DIN";
        MbusAddress.IdentificationNo = "00001111";
        MbusAddress.Version = "1E";
        MbusAddress.DeviceType = "04";
        DeviceId.MbusAddress = MbusAddress;
        Device.DeviceId = DeviceId;
        Device.DeviceKey = new CryptoKey[2];
        CryptoKey DeviceKey1 = new CryptoKey();
        int j = 0;
        EncryptedDataType EncDevKey = new EncryptedDataType();
        KeyList Key = new KeyList();
        //-----------------------------------------------------------------------------
        KeyList[] Keys1 = new KeyList[2];
        // First DeviceKey
        DeviceKey1.KeyIndex = 0;
        InterfaceTypes KeyInterface1 = new InterfaceTypes();
        DeviceKey1.KeyInterface = new InterfaceTypes[1];
        KeyInterface1 = InterfaceTypes.RemoteWireless;
        DeviceKey1.KeyInterface[0] = KeyInterface1;
        DeviceKey1.KeyMode = new CryptoMethods();
        DeviceKey1.KeyMode.Item = CryptoMethod2;

        KeyDefinitions KeyDefinition1 = new KeyDefinitions();
        KeyDefinition1.KeyType = KeyTypes.EncKey;

        KeyUsage KeyUsage1 = new KeyUsage();
        KeyUsage1.Item = KeyApplicationType.Data;
        KeyDefinition1.KeyUsage = KeyUsage1;
```

```
        DeviceKey1.KeyDefinition = KeyDefinition1;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[ex, 0, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData1 = new CipherDataType();
        EncData1.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey1 = new EncryptedDataType();
        EncDevKey1.EncryptionMethod = kem;
        EncDevKey1.CipherData = EncData1;
        KeyInfoType kit1 = new KeyInfoType();
        kit1.ItemsElementName = new ItemsChoiceType2[1];
        kit1.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit1.Items = new String[1];
        kit1.Items[0] = "Preset Key from Factory";
        EncDevKey1.KeyInfo = kit1;
        KeyList Key1 = new KeyList();
        Key1.KeyData = EncDevKey1;
        Key1.KeyVersion = 1;
        Keys1[0] = Key1;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = TheKey[i] = Exkey[ex, 1, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData2 = new CipherDataType();
        EncData2.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey2 = new EncryptedDataType();
        EncDevKey2.EncryptionMethod = kem;
        EncDevKey2.CipherData = EncData2;
        KeyInfoType kit2 = new KeyInfoType();
        kit2.ItemsElementName = new ItemsChoiceType2[1];
        kit2.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit2.Items = new String[1];
        kit2.Items[0] = "Replacement Key - change with Service tool";
        EncDevKey2.KeyInfo = kit2;
        KeyList Key2 = new KeyList();
        Key2.KeyData = EncDevKey2;
        Key2.KeyVersion = 2;
        Keys1[1] = Key2;
        DeviceKey1.Key = Keys1;
        Device.DeviceKey[0] = DeviceKey1;
           //-----------------------------------------------------------------------
        Devices[0] = Device;

        //=========================================================================
        // Second device

        DeviceType Device2 = new DeviceType();
        Device2.DeviceKey = new CryptoKey[1];
        KeyList[] Keys2 = new KeyList[2];
        DeviceId DeviceId2 = new DeviceId();
        DeviceId2.DinAddress = "7DIN0000002222";

        MbusAddress MbusAddress2 = new MbusAddress();
        MbusAddress2.Manufacturer = "DIN";
        MbusAddress2.IdentificationNo = "00002222";
        MbusAddress2.Version = "00";
        MbusAddress2.DeviceType = "03";
        DeviceId2.MbusAddress = MbusAddress2;
        Device2.DeviceId = DeviceId2;

        Device2.DeviceKey = new CryptoKey[2];
        CryptoKey DeviceKey2 = new CryptoKey();
        DeviceKey2.KeyIndex = 0;
        InterfaceTypes KeyInterfaceA = new InterfaceTypes();

        KeyInterfaceA = InterfaceTypes.RemoteWireless;
        DeviceKey2.KeyInterface = new InterfaceTypes[1];
```

```
        DeviceKey2.KeyInterface[0] = KeyInterfaceA;

        KeyList KeyA = new KeyList();
        CryptoMethods CryptoMethodA = new CryptoMethods();
        CryptoMethodA.Item = CryptoMethodType.OMSSecProfile_B;
        DeviceKey2.KeyMode = new CryptoMethods();
        DeviceKey2.KeyMode = CryptoMethodA;
        KeyDefinitions KeyDefinitionA = new KeyDefinitions();
        KeyDefinitionA.KeyType = KeyTypes.MasterKey;
        KeyUsage KeyUsageA = new KeyUsage();
        KeyUsageA.Item = KeyApplicationType.Data;
        KeyDefinitionA.KeyUsage = KeyUsageA;
        DeviceKey2.KeyDefinition = KeyDefinitionA;
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[ex, 2, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();

        CipherDataType EncDataA = new CipherDataType();
        EncDataA.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKeyA = new EncryptedDataType();
        EncDevKeyA.CipherData = EncDataA;
        EncDevKeyA.EncryptionMethod = kem;

        KeyA.KeyData = EncDevKeyA;
        Keys2[0] = KeyA;
        DeviceKey2.Key = Keys2;
        Device2.DeviceKey[0] = DeviceKey2;
        //-------------------------------------------------------------
        KeyList KeyB = new KeyList();
        KeyList[] Keys3 = new KeyList[1];
        InterfaceTypes KeyInterfaceB = new InterfaceTypes();
        CryptoKey DeviceKey3 = new CryptoKey();
        DeviceKey3.KeyIndex = 1;
        KeyInterfaceB = InterfaceTypes.Local;
        DeviceKey3.KeyInterface = new InterfaceTypes[1];
        DeviceKey3.KeyInterface[0] = KeyInterfaceB;
        KeyUsage KeyUsageB = new KeyUsage();
        KeyUsageB.Item = KeyApplicationType.Data;
        KeyDefinitions KeyDefinitionB = new KeyDefinitions();
        KeyDefinitionB.KeyUsage = KeyUsageB;
        DeviceKey3.KeyDefinition = KeyDefinitionB;
        DeviceKey3.KeyMode = new CryptoMethods();
        CryptoMethods CryptoMethodB = new CryptoMethods();
        CryptoMethodB.Item = "a user defined Crypto Method";
        DeviceKey3.KeyMode = CryptoMethodB;
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[ex, 3, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncDataB = new CipherDataType();
        EncDataB.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKeyB = new EncryptedDataType();
        EncDevKeyB.CipherData = EncDataB;
        EncDevKeyB.EncryptionMethod = kem;

        KeyB.KeyData = EncDevKeyB;
        Keys3[0] = KeyB;
        DeviceKey3.Key = Keys3;

        DeviceKey3.Key = Keys2;
        Device2.DeviceKey[1] = DeviceKey3;
        Devices[1] = Device2;
        //=============================================================================
        clsOMS.Device = Devices;

        serializer.Serialize(writer, clsOMS);
        writer.Close();
        // ----------now sign the document ----------------------------
        CspParameters cspParamsSig = new CspParameters();
```

```
        cspParamsSig.KeyContainerName = "XML_SIG_RSA_KEY_3072";
        // Specify an signature key.
        cspParamsSig.KeyNumber = (int)KeyNumber.Signature;
        // Get the RSA key from the container. This key will sign the XML document.
        RSACryptoServiceProvider rsaKeySig = new
RSACryptoServiceProvider(cspParamsSig);
        //Console.WriteLine("Signature key retrieved from container : \n {0}",
rsaKeySig.ToXmlString(true));

        // Create a new XML document.
        XmlDocument xmlDoc = new XmlDocument();

        // Load an XML file into the XmlDocument object.
        xmlDoc.PreserveWhitespace = true;
        xmlDoc.Load(Filename);

        // Canonalize and convert to bytes
        XmlDsigC14NTransform can = new XmlDsigC14NTransform(false);
        can.LoadInput(xmlDoc);
        var ms = (MemoryStream)can.GetOutput(typeof(Stream));
        ms.Flush();
        byte[] bytes = new byte[ms.Length * sizeof(char)];
        bytes = ms.ToArray();
        SHA256 hash = new SHA256Managed();
        hash.Initialize();
        hash.TransformFinalBlock(bytes, 0, bytes.Length);

        byte[] SignatureValue = new byte[rsaKeySig.KeySize / 8];
        SignatureValue = rsaKeySig.SignHash(hash.Hash,
CryptoConfig.MapNameToOID("SHA256"));

        var Signature = new Signature();
        Signature.SignatureValue = new byte[rsaKeySig.KeySize / 8]; ;
        Signature.SignatureValue = SignatureValue;
        Signature.KeyInfo = new KeyInfo();
        Signature.SignedInfo = new SignedInfo();

        KeyValueType Skv = new KeyValueType();
        Skv.Text = new String[1];
        Skv.Text[0] = rsaKeySig.ToXmlString(false);

        KeyInfo kiS = new KeyInfo();

        kiS.AddClause(new RSAKeyValue(rsaKeySig));

        Signature.KeyInfo = kiS;
        // Create a reference to be signed.
        Reference reference = new Reference();
        reference.Uri = "";
        // Add an enveloped transformation to the reference.
        XmlDsigEnvelopedSignatureTransform env = new
XmlDsigEnvelopedSignatureTransform();
        env.Algorithm = "http://www.w3.org/2000/09/xmldsig#enveloped-signature";

        reference.AddTransform(env);
        reference.DigestValue = new byte[hash.Hash.Length];
        reference.DigestValue = hash.Hash;
        reference.DigestMethod = "http://www.w3.org/2001/04/xmlenc#sha256";
        reference.AddTransform(can);

        SignedInfo SignedInfo = new SignedInfo();
        SignedInfo.CanonicalizationMethod = "http://www.w3.org/TR/2001/REC-xml-c14n-
20010315";
        SignedInfo.AddReference(reference);
        SignedInfo.SignatureMethod = "http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256";

        Signature.SignedInfo = new SignedInfo();
```

```
        Signature.SignedInfo = SignedInfo;

        XmlElement xmlDigitalSignature;
        xmlDigitalSignature = Signature.GetXml();
        xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature,
true));

        Console.WriteLine("XML file " + Filename + " signed.");

        // Save the document.
        xmlDoc.Save(Filename);
     }
   }
}
```

B.5 program.cs.ex2

This implementation is meant as an example only, and it is not to be used in production code.

The content of the file program.cs.ex2 is listed below. The file itself is also attached to this document.

```
/* Copyright 2015 Landis + Gyr.
   This program is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or any
   later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program.  If not, see <http://www.gnu.org/licenses/>. */
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security;
using System.Security.Cryptography;
using System.Security.Cryptography.Xml;
using System.Xml;
using System.Xml.Serialization;
using System.Xml.XPath;
using RFC3394;
namespace oms1
{
   class Program
   {
      static void Main(string[] args)
      {
         int ex = 0;
         byte[] TheKey =
{0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77};

         byte[, ,] Exkey = {{
   {0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77,0x11,0x33,0x55,0x77},
   {0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88},
   {0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00},
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11},
   {0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88,0x22,0x44,0x66,0x88},
   {0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00,0xAA,0xCC,0xEE,0x00},
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11},
```

```
   {0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11,0xBB,0xDD,0xFF,0x11}}
,
   {{0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77}
,
   {0x22,0x44,0x66,0x88,0xAA,0xCC,0xEF,0x01,0x12,0x23,0x34,0x45,0x56,0x67,0x78,0x89},
   {0x32,0x54,0x76,0x98,0xBA,0xDC,0xFF,0x12,0x13,0x24,0x35,0x46,0x57,0x68,0x79,0x8A},
   {0x42,0x64,0x86,0xA8,0xCA,0xED,0x0F,0x23,0x14,0x25,0x36,0x47,0x58,0x69,0x7A,0x8B},
   {0x52,0x74,0x96,0xB8,0xDA,0xFD,0x1F,0x34,0x15,0x26,0x37,0x48,0x59,0x6A,0x7B,0x8C},
   {0x62,0x84,0xA6,0xC8,0xEB,0x0D,0x2F,0x45,0x16,0x27,0x38,0x49,0x5A,0x6B,0x7C,0x8D},
   {0xE3,0x05,0x27,0x49,0x6B,0x8D,0xAF,0xCD,0x1E,0x2F,0x40,0x51,0x62,0x73,0x84,0x8E},
   {0xF3,0x15,0x37,0x59,0x7B,0x9D,0xBF,0xDE,0x1F,0x30,0x41,0x52,0x63,0x74,0x85,0x8F}}

     };

        byte[] TheSessionKey =
{0xDE,0xAD,0xBE,0xEF,0x00,0x12,0x34,0x56,0x78,0x9A,0xBC,0xCA,0xFE,0xBA,0xBE,0x00};
        String Filename = "mbus2.xml";
        // Create a new CspParasmeters object to specify
        // a key container.
        const int iMbusLen = 16;

        EncryptionMethodType kem = new EncryptionMethodType();
        if (TheSessionKey.Length == 16)
        {
           kem.Algorithm = "http://www.w3.org/2001/04/xmlenc#kw-aes128";
           kem.KeySize = "128";
        }
        else if (TheSessionKey.Length == 32)
        {
           kem.Algorithm = "http://www.w3.org/2001/04/xmlenc#kw-aes256";
           kem.KeySize = "256";
        }
        else
        {
           Console.Write("Wrong KEK length: " + TheSessionKey.Length);
           return;
        }

        CspParameters cspParamsEnc = new CspParameters();
        cspParamsEnc.KeyContainerName = "XML_ENC_RSA_KEY_3072";
        // Specify an exchange key.
        cspParamsEnc.KeyNumber = (int)KeyNumber.Exchange;
        // Get the RSA key from the container. This key will encrypt
        // a symmetric key, which will then be encryped in the XML document.
        RSACryptoServiceProvider rsaKeyEnc = new
RSACryptoServiceProvider(cspParamsEnc);
        // Console.WriteLine("Encryption key retrieved from container : \n {0}",
rsaKeyEnc.ToXmlString(true));


        clsRFC3394 wrap = new clsRFC3394(TheSessionKey);

        // Encrypt the session key and add it to an EncryptedKey element.
        EncryptedKey ek = new EncryptedKey();
        EncryptedKeyType ekt = new EncryptedKeyType();
        byte[] encryptedKey = EncryptedXml.EncryptKey(TheSessionKey, rsaKeyEnc,
false);
        CipherDataType cd = new CipherDataType();
        EncryptionMethodType em = new EncryptionMethodType();
        em.Algorithm = "http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p";
        em.KeySize = rsaKeyEnc.KeySize.ToString();

        RetrievalMethodType rm = new RetrievalMethodType();
        rm.URI = "#SessionKey";
        rm.Type = "http://www.w3.org/2001/04/xmlenc#EncryptedKey";
        cd.Item = encryptedKey;
        ekt.CipherData = cd;
        ekt.EncryptionMethod = em;
```

```
        ekt.Id = "KeyId";
        ekt.CarriedKeyName = "SessionKey";

        XmlSerializer serializer = new XmlSerializer(typeof(OMSKeyExchange));
        //Create XML File objects
        OMSKeyExchange clsOMS = new OMSKeyExchange();
        StreamWriter writer = new StreamWriter(Filename);

        clsOMS.TransportKey = ekt;
        DataReference dRef = new DataReference();

        // Specify the EncryptedData URI.
        dRef.Uri = "#SessionKey";
        KeyInfoType kit = new KeyInfoType();
        kit.ItemsElementName = new ItemsChoiceType2[1];
        kit.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit.Items = new String[1];
        kit.Items[0] = "SessionKey";
        //The above is the same as the previous (1.6) example
        VendorData OrderInfo = new VendorData();
        OrderInfo.Any = new XmlElement[5];
        XmlDocument domO = new XmlDocument();
        XmlNode node0 = domO.CreateNode(XmlNodeType.Element, "ordernumber",
"MyVendorNamespace");
        node0.InnerText = "12345";
        OrderInfo.Any[0] = (XmlElement)node0;
        XmlNode node1 = domO.CreateNode(XmlNodeType.Element, "customer",
"MyVendorNamespace");
        node1.InnerText = "Customer ABC";
        OrderInfo.Any[1] = (XmlElement)node1;
        XmlNode node2 = domO.CreateNode(XmlNodeType.Element, "street",
"MyVendorNamespace");
        node2.InnerText = "West Road 123";
        OrderInfo.Any[2] = (XmlElement)node2;
        XmlNode node3 = domO.CreateNode(XmlNodeType.Element, "zip",
"MyVendorNamespace");
        node3.InnerText = "12345";
        OrderInfo.Any[3] = (XmlElement)node3;
        XmlNode node4 = domO.CreateNode(XmlNodeType.Element, "city",
"MyVendorNamespace");
        node4.InnerText = "Middle City";
        OrderInfo.Any[4] = (XmlElement)node4;

           clsOMS.VendorOrderData = new VendorData();
           clsOMS.VendorOrderData = OrderInfo;
        //=========================================================================
        // First device
        DeviceType[] Devices = new DeviceType[2];

        DeviceType Device = new DeviceType();

        DeviceId DeviceId = new DeviceId();

           Device.DeviceIndex = 1;
           DeviceId.DinAddress = "1XXX0A33334444";

           MbusAddress MbusAddress = new MbusAddress();
           MbusAddress.Manufacturer = "XXX";
           MbusAddress.IdentificationNo = "33334444";
           MbusAddress.Version = "0A";
           MbusAddress.DeviceType = "02";
           DeviceId.MbusAddress = MbusAddress;

        VendorData VendorDeviceData = new VendorData();
        VendorDeviceData.Any = new XmlElement[1];
        XmlDocument domV = new XmlDocument();
        XmlNode nodeM = domV.CreateNode(XmlNodeType.Element, "meterinfo",
"MyVendorNamespace");
```

```
        XmlNode nodeM0 = domV.CreateNode(XmlNodeType.Element, "MeterName",
"MyVendorNamespace");
        nodeM0.InnerText = "Electricity meter";
        XmlNode nodeM1 = domV.CreateNode(XmlNodeType.Element, "ConsumptionType",
"MyVendorNamespace");
        nodeM1.InnerText = "kWh";
        XmlNode nodeM2 = domV.CreateNode(XmlNodeType.Element, "IMEI",
"MyVendorNamespace");
        nodeM2.InnerText = "357805023984942";
        nodeM.AppendChild(nodeM0);
        nodeM.AppendChild(nodeM1);
        nodeM.AppendChild(nodeM2);
           VendorDeviceData.Any[0] = (XmlElement)nodeM;
           Device.VendorDeviceData = VendorDeviceData;
        Device.DeviceId = DeviceId;
        Device.DeviceIndex = 1;
        Device.DeviceIndexSpecified = true;
        Device.DeviceKey = new CryptoKey[4];
        CryptoKey DeviceKey1 = new CryptoKey();
        int j = 0;
        EncryptedDataType EncDevKey = new EncryptedDataType();
        KeyList Key = new KeyList();
        //-----------------------------------------------------------------------------
        // First DeviceKey
        DeviceKey1.KeyIndex = 0;
        InterfaceTypes KeyInterface1 = new InterfaceTypes();
        DeviceKey1.KeyInterface = new InterfaceTypes[1];
        KeyInterface1 = InterfaceTypes.RemoteWireless;
        DeviceKey1.KeyInterface[0] = KeyInterface1;

        CryptoMethods CryptoMethod1 = new CryptoMethods();
        CryptoMethod1.Item = "RFC6188_AES-192_CTR";

        DeviceKey1.KeyMode = new CryptoMethods();
        DeviceKey1.KeyMode = CryptoMethod1;

        KeyDefinitions KeyDefinition1 = new KeyDefinitions();
        KeyDefinition1.KeyType = KeyTypes.EncKey;

        KeyDefinition1.KeyID = 1;
        KeyDefinition1.KeyIDSpecified = true;
        KeyUsage KeyUsage1 = new KeyUsage();
        KeyUsage1.Item = KeyApplicationType.Data;
        KeyDefinition1.KeyUsage = KeyUsage1;
        DeviceKey1.KeyDefinition = KeyDefinition1;

        KeyList[] Keys3 = new KeyList[3];
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 0, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData1 = new CipherDataType();
        EncData1.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey1 = new EncryptedDataType();
        EncDevKey1.EncryptionMethod = kem;
        EncDevKey1.CipherData = EncData1;
        KeyInfoType kit1 = new KeyInfoType();
        kit1.ItemsElementName = new ItemsChoiceType2[2];
        kit1.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit1.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit1.Items = new Object[2];
        kit1.Items[0] = "Valid_for_2015";
        kit1.Items[1] = rm;

        EncDevKey1.KeyInfo = kit1;
        KeyList Key1 = new KeyList();
        Key1.KeyData = EncDevKey1;
        Key1.KeyVersion = 0;
        Keys3[0] = Key1;
```

```
        j++;
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = TheKey[i] = Exkey[1, 1, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData2 = new CipherDataType();
        EncData2.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey2 = new EncryptedDataType();
        EncDevKey2.EncryptionMethod = kem;
        KeyInfoType kit2 = new KeyInfoType();
        kit2.ItemsElementName = new ItemsChoiceType2[2];
        kit2.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit2.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit2.Items = new Object[2];
        kit2.Items[0] = "Valid_for_2016";
        kit2.Items[1] = rm;
        EncDevKey2.CipherData = EncData2;
        EncDevKey2.KeyInfo = kit2;
        KeyList Key2 = new KeyList();
        Key2.KeyData = EncDevKey2;
        Key2.KeyVersion = 1;
        Keys3[1] = Key2;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 2, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData3 = new CipherDataType();
        EncData3.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey3 = new EncryptedDataType();
        EncDevKey3.EncryptionMethod = kem;
        KeyInfoType kit3 = new KeyInfoType();
        kit3.ItemsElementName = new ItemsChoiceType2[2];
        kit3.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit3.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit3.Items = new Object[2];
        kit3.Items[1] = rm;
        kit3.Items[0] = "Valid_for_2017";
        EncDevKey3.CipherData = EncData3;
        EncDevKey3.KeyInfo = kit3;
        KeyList Key3 = new KeyList();
        Key3.KeyData = EncDevKey3;
        Key3.KeyVersion = 2;
        Keys3[2] = Key3;

        DeviceKey1.Key = Keys3;
        Device.DeviceKey[0] = DeviceKey1;

        //-------------------------------------------------------------------------
        //Second DeviceKey
        KeyList[] Keys1A = new KeyList[1];
        CryptoKey DeviceKey2 = new CryptoKey();
        DeviceKey2.KeyIndex = 1;

        InterfaceTypes KeyInterface2 = new InterfaceTypes();
        DeviceKey2.KeyInterface = new InterfaceTypes[1];
        KeyInterface2 = InterfaceTypes.RemoteWireless;
        DeviceKey2.KeyInterface[0] = KeyInterface2;
        CryptoMethods CryptoMethod3 = new CryptoMethods();
        CryptoMethod3.Item = "RFC6188_AES-192_CTR";
        DeviceKey2.KeyMode = new CryptoMethods();
        DeviceKey2.KeyMode = CryptoMethod3;

        KeyDefinitions KeyDefinition2 = new KeyDefinitions();
        KeyDefinition2.KeyType = KeyTypes.EncKey;
        KeyDefinition2.KeyID = 2;
        KeyDefinition2.KeyIDSpecified = true;
        KeyUsage KeyUsage2 = new KeyUsage();
        KeyUsage2.Item = KeyApplicationType.FirmwareUpdate;
```

```
        KeyDefinition2.KeyUsage = KeyUsage2;
        DeviceKey2.KeyDefinition = KeyDefinition2;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 3, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData4 = new CipherDataType();
        EncData4.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey4 = new EncryptedDataType();
        EncDevKey4.EncryptionMethod = kem;
        KeyInfoType kit4 = new KeyInfoType();
        kit4.ItemsElementName = new ItemsChoiceType2[2];
        kit4.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit4.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit4.Items = new Object[2];
        kit4.Items[1] = rm;
        kit4.Items[0] = "Valid_for_2017";
        EncDevKey4.CipherData = EncData4;
        EncDevKey4.KeyInfo = kit4;
        KeyList Key4 = new KeyList();
        Key4.KeyData = EncDevKey4;
        Key4.KeyVersion = 0;
        Keys1A[0] = Key4;
        DeviceKey2.Key = Keys1A;
        Device.DeviceKey[1] = DeviceKey2;

        //-------------------------------------------------------------------------
        //Third DeviceKey
        KeyList[] Keys1B = new KeyList[1];
        CryptoKey DeviceKey3 = new CryptoKey();
        DeviceKey3.KeyIndex = 2;

        InterfaceTypes KeyInterface3 = new InterfaceTypes();
        DeviceKey3.KeyInterface = new InterfaceTypes[1];
        KeyInterface3 = InterfaceTypes.RemoteWireless;
        DeviceKey3.KeyInterface[0] = KeyInterface3;

        CryptoMethods CryptoMethod4 = new CryptoMethods();
        CryptoMethod4.Item = "RFC618_AES-192_CTR";
        DeviceKey3.KeyMode = new CryptoMethods();
        DeviceKey3.KeyMode = CryptoMethod4;

        KeyDefinitions KeyDefinition3 = new KeyDefinitions();
        KeyDefinition3.KeyType = KeyTypes.EncKey;
        KeyDefinition3.KeyID = 3;
        KeyDefinition3.KeyIDSpecified = true;
        KeyUsage KeyUsage3 = new KeyUsage();
        KeyUsage3.Item = "Backup Key";
        KeyDefinition3.KeyUsage = KeyUsage3;
        DeviceKey3.KeyDefinition = KeyDefinition3;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 4, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData5 = new CipherDataType();
        EncData5.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey5 = new EncryptedDataType();
        EncDevKey5.EncryptionMethod = kem;
        KeyInfoType kit5 = new KeyInfoType();
        kit5.ItemsElementName = new ItemsChoiceType2[2];
        kit5.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit5.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit5.Items = new Object[2];
        kit5.Items[1] = rm;
        kit5.Items[0] = "Valid_only_once";
        EncDevKey5.CipherData = EncData5;
        EncDevKey5.KeyInfo = kit5;
        KeyList Key5 = new KeyList();
```

```
        Key5.KeyData = EncDevKey5;
        //Key5.KeyVersion = "1";
        Keys1B[0] = Key5;
        DeviceKey3.Key = Keys1B;
        Device.DeviceKey[2] = DeviceKey3;

        //-------------------------------------------------------------------------
        //Fourth DeviceKey
        KeyList[] Keys1C = new KeyList[1];
        CryptoKey DeviceKey4 = new CryptoKey();
        DeviceKey4.KeyIndex = 3;

        InterfaceTypes KeyInterface4 = new InterfaceTypes();
        DeviceKey4.KeyInterface = new InterfaceTypes[1];
        KeyInterface4 = InterfaceTypes.RemoteWireless;
        DeviceKey4.KeyInterface[0] = KeyInterface4;

        CryptoMethods CryptoMethod5 = new CryptoMethods();
        CryptoMethod5.Item = "RFC4493_CMAC_AES128_trunc_4Byte";
        DeviceKey4.KeyMode = new CryptoMethods();
        DeviceKey4.KeyMode = CryptoMethod5;

        KeyDefinitions KeyDefinition4 = new KeyDefinitions();
        KeyDefinition4.KeyID = 1;
        KeyDefinition4.KeyIDSpecified = true;
        KeyDefinition4.KeyType = KeyTypes.MacKey;
        KeyUsage KeyUsage4 = new KeyUsage();
        KeyUsage4.Item = "MAC_For_All_EncModes";
        KeyDefinition4.KeyUsage = KeyUsage4;
        DeviceKey4.KeyDefinition = KeyDefinition4;

        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[ex, 5, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncData6 = new CipherDataType();
        EncData6.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKey6 = new EncryptedDataType();
        EncDevKey6.EncryptionMethod = kem;
        KeyInfoType kit6 = new KeyInfoType();
        kit6.ItemsElementName = new ItemsChoiceType2[2];
        kit6.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kit6.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kit6.Items = new Object[2];
        kit6.Items[1] = rm;
        kit6.Items[0] = "MacKey";

        EncDevKey6.CipherData = EncData6;
        EncDevKey6.KeyInfo = kit6;
        KeyList Key6 = new KeyList();
        Key6.KeyData = EncDevKey6;
        //Key6.KeyVersion = "2";
        Keys1C[0] = Key6;
        DeviceKey4.Key = Keys1C;
        Device.DeviceKey[3] = DeviceKey4;
        //-------------------------------------------------------------------------
        Devices[0] = Device;

        //=========================================================================
        // Second device

        DeviceType Device2 = new DeviceType();
        Device2.DeviceIndex = 2;
        Device2.DeviceKey = new CryptoKey[1];
        KeyList[] Keys2 = new KeyList[2];
        DeviceId DeviceId2 = new DeviceId();
        DeviceId2.DinAddress = "7DIN0000002222";

        MbusAddress MbusAddress2 = new MbusAddress();
```

```
        MbusAddress2.Manufacturer = "DIN";
        MbusAddress2.IdentificationNo = "00002222";
        MbusAddress2.Version = "00";
        MbusAddress2.DeviceType = "03";
        DeviceId2.MbusAddress = MbusAddress2;
        Device2.DeviceId = DeviceId2;
        Device2.DeviceIndex = 2;
        Device2.DeviceIndexSpecified = true;
        Device2.DeviceKey = new CryptoKey[1];
        CryptoKey DeviceKey = new CryptoKey();
        DeviceKey.KeyIndex = 0;
        InterfaceTypes KeyInterfaceA = new InterfaceTypes();

        KeyInterfaceA = InterfaceTypes.RemoteWireless;
        DeviceKey.KeyInterface = new InterfaceTypes[1];
        DeviceKey.KeyInterface[0] = KeyInterfaceA;

        KeyList KeyA = new KeyList();
        CryptoMethods CryptoMethodA = new CryptoMethods();
        CryptoMethodA.Item = CryptoMethodType.OMSSecProfile_A;
        DeviceKey.KeyMode = new CryptoMethods();
        DeviceKey.KeyMode = CryptoMethodA;
        KeyDefinitions KeyDefinitionA = new KeyDefinitions();
        KeyDefinitionA.KeyType = KeyTypes.EncKey;
        KeyUsage KeyUsageA = new KeyUsage();
        KeyUsageA.Item = KeyApplicationType.Data;
        KeyDefinitionA.KeyUsage = KeyUsageA;
        DeviceKey.KeyDefinition = KeyDefinitionA;
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 6, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();

        CipherDataType EncDataA = new CipherDataType();
        EncDataA.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKeyA = new EncryptedDataType();
        EncDevKeyA.CipherData = EncDataA;
        EncDevKeyA.EncryptionMethod = kem;

        KeyInfoType kitA = new KeyInfoType();
        kitA.ItemsElementName = new ItemsChoiceType2[2];
        kitA.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kitA.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kitA.Items = new Object[2];
        kitA.Items[1] = rm;
        kitA.Items[0] = "Preset Key from Factory";

        EncDevKeyA.KeyInfo = kitA;
        KeyA.KeyData = EncDevKeyA;
        KeyA.KeyVersion = 1;
        Keys2[0] = KeyA;

        j++;
        //-----------------------------------------------------------------------------
        KeyList KeyB = new KeyList();
        for (int i = 0; i < iMbusLen; i++) { TheKey[i] = Exkey[1, 7, i];
Console.Write(TheKey[i].ToString("x")); }
        Console.WriteLine();
        CipherDataType EncDataB = new CipherDataType();
        EncDataB.Item = wrap.WrapKey(TheKey);
        EncryptedDataType EncDevKeyB = new EncryptedDataType();
        EncDevKeyB.CipherData = EncDataB;
        EncDevKeyB.EncryptionMethod = kem;
        KeyInfoType kitB = new KeyInfoType();
        kitB.ItemsElementName = new ItemsChoiceType2[2];
        kitB.ItemsElementName[0] = ItemsChoiceType2.KeyName;
        kitB.ItemsElementName[1] = ItemsChoiceType2.RetrievalMethod;
        kitB.Items = new Object[2];
        kitB.Items[1] = rm;
```

```
        kitB.Items[0] = "Replacement Key - change with Service tool";
        EncDevKeyB.KeyInfo = kitB;
        KeyB.KeyData = EncDevKeyB;
        KeyB.KeyVersion = 2;
        Keys2[1] = KeyB;
        j++;

        DeviceKey.Key = Keys2;
        Device2.DeviceKey[0] = DeviceKey;
        Devices[1] = Device2;
        //=============================================================================
        clsOMS.Device = Devices;

        serializer.Serialize(writer, clsOMS);
        writer.Close();
        // ----------now sign the document ----------------------------------------
        CspParameters cspParamsSig = new CspParameters();
        cspParamsSig.KeyContainerName = "XML_SIG_RSA_KEY_3072";
        // Specify an signature key.
        cspParamsSig.KeyNumber = (int)KeyNumber.Signature;
        // Get the RSA key from the container. This key will sign the XML document.
        RSACryptoServiceProvider rsaKeySig = new
RSACryptoServiceProvider(cspParamsSig);
        //Console.WriteLine("Signature key retrieved from container : \n {0}",
rsaKeySig.ToXmlString(true));

        // Create a new XML document.
        XmlDocument xmlDoc = new XmlDocument();

        // Load an XML file into the XmlDocument object.
        xmlDoc.PreserveWhitespace = true;
        xmlDoc.Load(Filename);

        // Canonalize and convert to bytes
        XmlDsigC14NTransform can = new XmlDsigC14NTransform(false);
        can.LoadInput(xmlDoc);
        var ms = (MemoryStream)can.GetOutput(typeof(Stream));
        ms.Flush();
        byte[] bytes = new byte[ms.Length * sizeof(char)];
        bytes = ms.ToArray();
        SHA256 hash = new SHA256Managed();
        hash.Initialize();
        hash.TransformFinalBlock(bytes, 0, bytes.Length);

        byte[] SignatureValue = new byte[rsaKeySig.KeySize / 8];
        SignatureValue = rsaKeySig.SignHash(hash.Hash,
CryptoConfig.MapNameToOID("SHA256"));

        var Signature = new Signature();
        Signature.SignatureValue = new byte[rsaKeySig.KeySize / 8]; ;
        Signature.SignatureValue = SignatureValue;
        Signature.KeyInfo = new KeyInfo();
        Signature.SignedInfo = new SignedInfo();

        KeyValueType Skv = new KeyValueType();
        Skv.Text = new String[1];
        Skv.Text[0] = rsaKeySig.ToXmlString(false);

        KeyInfo kiS = new KeyInfo();

        kiS.AddClause(new RSAKeyValue(rsaKeySig));

        Signature.KeyInfo = kiS;
        // Create a reference to be signed.
        Reference reference = new Reference();
        reference.Uri = "";
        // Add an enveloped transformation to the reference.
        XmlDsigEnvelopedSignatureTransform env = new
```

```
XmlDsigEnvelopedSignatureTransform();
        env.Algorithm = "http://www.w3.org/2000/09/xmldsig#enveloped-signature";

        reference.AddTransform(env);
        reference.DigestValue = new byte[hash.Hash.Length];
        reference.DigestValue = hash.Hash;
        reference.DigestMethod = "http://www.w3.org/2001/04/xmlenc#sha256";
        reference.AddTransform(can);

        SignedInfo SignedInfo = new SignedInfo();
        SignedInfo.CanonicalizationMethod = "http://www.w3.org/TR/2001/REC-xml-c14n-
20010315";
        SignedInfo.AddReference(reference);
        SignedInfo.SignatureMethod = "http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256";

        Signature.SignedInfo = new SignedInfo();
        Signature.SignedInfo = SignedInfo;

        XmlElement xmlDigitalSignature;
        xmlDigitalSignature = Signature.GetXml();
        xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature,
true));

        Console.WriteLine("XML file " + Filename + " signed.");

        // Save the document.
        xmlDoc.Save(Filename);
    }
  }
}
```

B.6 rsa3072_pr.xml

The content of the file rsa3072_pr.xml which contains the private key is listed below. The file itself is also attached to this document.

IMPORTANT!

The private key shall never be transmitted. It is only showed here to allow a verification of the example.

```
<RSAKeyValue>
<Modulus>lIRU8V1F0bQ4IORe1bdMhTEl1cguTZAkM5FIP6gf00nT8Dn7bNv7BWTTRdnofI5OUZr
lCZzRju/QqmFVw/qjxelkXvV0+1dMPLPmmdgdi4XcVDZJvCXTr30dBjE+gpIYm1ggB8EfLfPxPcZ
0chrMSXkUyVK3k1YGSh92Vka+yfLkk18gkJcURgX7HQ3A/wRpW+GZVl/BMsQKYE7uQVGWEIwSKFm
6XkjZIr5iNDP7fCVt1inFxk7AkriP8u42jpvcQ19HprBdOqYtGTxyBqWUkkXr0xTE3eF550MzfBW
o8iMMsoOdF9kd1M1c+rpAOxX2fsOxkAya5BXGcR6wXB6x1dhZvACdpgwanqINoJYf6nmgYEIb2/H
mZkvAVlMjwBuDHyE7armLFrotRYYQg5rlC3d4FKw5W2KnJNhvB2KPXK2m2p+/ZRghIbUSmCl1a9A
fsQM6pb771Ou5G16s8UQhh4HMSPSpClKTkgxzhSrBQt2HCm+q6qQR5J6o
6ldatWS9</Modulus>
<Exponent>AQAB</Exponent>
<P>zOtNbazLZptuP6phxgZ68rZNS49rbfSZW3qqfEHnCwZbg6iPUSdRNJZl8b64xrPkggg+uZom/
4paYFXdr6KVABM8wSWWGMN95NfZXrS/4enFiZhmui27uWqnYf7gubDBU5uFkUAVCUvx7Q7M3MZBv
/BrV5P2uyMejoqA2tzuiITcOdG0CQ3o57+aC+I0XauSNt0oShE6+z1UWxuuoagL31xqWHZQc/Ct0
g4bsXAxCBkA/d4zVgk5x1ijzP9mEeMd</P>
<Q>uYnH9w5am6p71MP5XsJ9h2oUVM10YR7PFqloryICuHPMlrYOI3xO/lo8NdhVpontUiZmE3jBP
DZv32vPozMExTKZ79wEiFzgjPOrXlfMcM6Pwe6wea6ifXn7Hd2dn4JFHG08E3j81Tiw4v7MaVzmz
NTJFyQj+CAw1KGaD2IdRtTgTrP9m3wVpokTrQWS1Zuwe6Q887dfLzClAO3YiKDYwDJKK1IvxZegq
brwTnTM8ivbAoevxqZGqYCks1z/SzYh</Q>
<DP>btwRnHbx50XJRDoqARa6ZxpHAMzLQsYkPUBEgUqp8uXylmyXdEEABqIi7sDOd/kUc2CAhqql
1HOF5z4s23rqfwrpQFjUNft03VwE5d6FPHPbAVBoqupY97uCaFqaOvsr685kWStA9jGW5TevjvTW
1z7643tR7PXX7ydxBHanuUWfWsxfSNK0oMOgVDOBPve192Z4CHH/5bHNNCjqtdwS6Di387m2VrLr
ZotpYfofHJFiXvVZ8n61Ompy21GEZ8Ll</DP>
<DQ>baEf6rcsirZdnQZ/HqLFEXJ4uPL9RTWzH6zSEkmCWgLybEr+Z7FX66rd0OMXGOR+uf++aNr9
```

```
Mtrnl5hYhBHtS6fulW/nrbtTY3XHWGOAhVdIUkr7M6sgkAu3eZotPMb4lydkTA09N6/btQThO5CN
P5Krj2TnbAF5WOpz4YsTyXqXX2my/GCBWnm8iou0i0sHKMeOzGMZdwTDOp1PSMvhxw7wDFDvsU06
vY+b1n/laWLIceJC1poBPr6okuLsZkCB</DQ>
<InverseQ>fuwJFD4Q5E3mE2EZmH61o1thBlSS+4HB8Wdb8RFkodCCJSZftkAIaT1Y67pzn6Ojv7
OUW8S9ltZLCwQFglPrlV02tJyV3nIhOqary0zyQt1LM5YujTNwGflZ3hlxVNyXVr0y8/L72qm52f
GQHwpBwCTbuSYb8L6X2vYFM0dx1YCrpQO2STRyWRb7ltktzqNJ7kUQby47mN5KejMOzqpJe6c/aa
5XjXexydRx09lSzWBLJQJyheSclyU9llIWVw58</InverseQ>
<D>XEzp5Z+MNsoEYBzN+5CRk+15kCV68WV1uj6+YiGAtfWaLCA86+jDVxTUYQMOg+qFLh5P3euCK
Y1NZ7ZacRQiGGv2nzV5YINetBMya8ZuhjnJNuUOakowB06AbUrJiKh3doQMVAVn9U04TqY51HhnI
fqIOp6XzI+YFRT8nJ4XELdN63EhcfsS9HnfGK1llpgNSrcobw/PqFdCA5zKYfnsg5gp6L3rH5Gys
O8VZVAxKXSxkHohmh9zezwM6nO1oGeoR+RdG7TkfrsEq8XRXsmWkUMzYlX7m3niS6HumXf+U2dPu
ma0IaCq1Z+y1Gl5GG8OSp2bcslwiUFYAu3FZ03wEMydeIYIQFSpfExQgukLuzK8EMQyHqxTGPLDD
nhqT0yJwOA7Ph0pOQ+9X0hhAuiZUzxC4q6XJvFjvs/07q8VoQ3YAPKEPXmYb3rcLgxJ1ITqo5nZI
WV0SVshgi3O7p1Ruu3PhhDtxOUQjseRXLiAp2uKURDHBm7KvuKaCwOW2ysB</D>
</RSAKeyValue>
```

B.7 rsa3072_pu.xml

The content of the file rsa3072_pu.xml which contains the public key is listed below. The file itself is also attached to this document.

```
<RSAKeyValue>
<Modulus>lIRU8V1F0bQ4IORe1bdMhTEl1cguTZAkM5FIP6gf00nT8Dn7bNv7BWTTRdnofI5OUZr
lCZzRju/QqmFVw/qjxelkXvV0+1dMPLPmmdgdi4XcVDZJvCXTr30dBjE+gpIYm1ggB8EfLfPxPcZ
0chrMSXkUyVK3k1YGSh92Vka+yfLkk18gkJcURgX7HQ3A/wRpW+GZVl/BMsQKYE7uQVGWEIwSKFm
6XkjZIr5iNDP7fCVt1inFxk7AkriP8u42jpvcQ19HprBdOqYtGTxyBqWUkkXr0xTE3eF550MzfBW
o8iMMsoOdF9kd1M1c+rpAOxX2fsOxkAya5BXGcR6wXB6x1dhZvACdpgwanqINoJYf6nmgYEIb2/H
mZkvAVlMjwBuDHyE7armLFrotRYYQg5rlC3d4FKw5W2KnJNhvB2KPXK2m2p+/ZRghIbUSmCl1a9A
fsQM6pb771Ou5G16s8UQhh4HMSPSpClKTkgxzhSrBQt2HCm+q6qQR5J6o
6ldatWS9</Modulus>
<Exponent>AQAB</Exponent>
</RSAKeyValue>
```