Tullverket

Example to manually check and create XML signatures

2022-11-18

Contents

1 Introduction

This document is based on, and replaces, the previous document in Swedish, “Exempel på att skapa och kontrollera XML-signatur manuellt.pdf”, that was published 2017-05-22. This new version is only available in English.

The examples are using only simple command line utilities in Linux. The purpose is to enhance the understanding of the technical details regarding signature calculation and to provide tools for troubleshooting.

The examples are not to be used in production environments and are to be considered as unsecure. For example, no validation of references, document contents (payload) or the certificates is made and the commands will not work with arbitrary XML files. Using *awk* and *sed* as tools to interpret and create XML documents in not recommended. The regular expressions used in the commands are specially adapted for the example file only.

1.1 Links to files used in the examples

Example XML file (ENV-Envelope-UseCase1.xml)
https://www.tullverket.se/download/18.5c3d004415b89fa6ac78bb/1496322683644/Exempelfiler.zip

Test certificate used for the example file https://ftgtest-meddelandevalidering.tullverket.se/valideringstjanst/validering/andrafiler/Swedish_Customs_TEST_CA_0.1.zip

1.2 Links to specifications

Technical specifications SCTS-ENV (Swedish only)
https://www.tullverket.se/sv/foretag/deklareradigitalt/ediforsystemutvecklare/editekniskaspecifikationer/sctsenv.4.4481397d16411b8646515ab.html

Envelope and metadata specification
https://www.tullverket.se/download/18.4b9cdbde163ab2c5bdc170/1527764751722/SCTS-ENV_Kuvertspecifikation_1.0.2.pdf

2 Concept

The *electronic signature* is calculated on *SignedInfo*, that contains message digests (hash values, cryptographic checksum) on and references to each of the parts that are to be locked.

2.1 The data that are to be locked with message digests

The example file contains three subdocuments that are to be locked:

1. **Payload** (the business message),

```
<ds:Object Id="xmldsig-03f6a207-0933-4f24-8c24-58674efa8c7a-Payload">
  <md:MetaData xmlns:md="urn:se:customs:datamodel:WCO:DocumentMetaData:1">
    <md:WCODataModelVersionCode>3.6-SE</md:WCODataModelVersionCode>
    <md:ResponsibleCountryCode>SE</md:ResponsibleCountryCode>
    <md:ResponsibleAgencyName>Swedish Customs</md:ResponsibleAgencyName>
    <md:AgencyAssignedCustomizationCode>CWHOP</md:AgencyAssignedCustomizationCode>
    <md:AgencyAssignedCustomizationVersionCode>1</md:AgencyAssignedCustomizationVersi
    <md:FunctionalDefinition>CWDS</md:FunctionalDefinition>
    <Declaration xmlns="urn:se:customs:datamodel:WCO:Declaration:1">
      <!--Declaration/response goes here!-->
    </Declaration>
  </md:MetaData>
</ds:Object>
```

2. **KeyInfo** (with the signature certificate),

```
          <ds:KeyInfo Id="xmldsig-03f6a207-0933-4f24-8c24-58674efa8c7a-Keyinfo">
            <ds:X509Data>

              <ds:X509Certificate>MIIETzCCAzegAwIBAgIQE4bHuNu+Us5Ehao4KnCVtTANBgkqhkiG9u
3TELMAkGA1UEBhMCUOUxEzARBgNVBAoMClR1bGxZZXJrZXQxGDAWBgNVBAsMD1N3
ZWRpc2ggQ3VzdG9tczE3MDUGA1UECwwuVEVTVCBQdWJsaWMgSW5OZXJtZWRpYXRl
IENlcnRpZmljYXRlIEF1dGhvcml0eTEiMCAGA1UECwwZRm9yIHRlc3RpbmcgcHVy
cG9zZXMgb25seTEVMBMGA1UEBRMMUOUyMDIxMDAwOTY5MSswKQYDVQQDDCJTd2Vk
aXNoIEN1c3RvbXMgVEVTVCBQdWJsaWMgQOEgMC4xMB4XDTE1MDkxNDA3MjMyNFoX
DTM1MDkxNDA3MjMyNFowgYkxCzAJBgNVBAYTAlNFMRUwEwYDVQQKDAxUZXNOZsO2
cmVOYWcxFjAUBgNVBAsMDUlUIGRlcGFydG1lbnQxFTATBgNVBAUTDFNFOTk5OTk5
OTk5OTEOMDIGA1UEAwwrVGVzdCBjb21wYW55IGZvciBzaWduYXR1cmUgdmFsaWRh
dGlvbiBOZXNOczCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL9njV9+
IsKK+/x7oeOwSdeOLMHM2hZ4yg3Lt6rfZ8fOhGdpvZj1uogWM4dmTtSC2ogJ4vZy
Ozq++FiZZmwOdt6ha2mzWWd6Vv3WKgyOOSLnOnWOfSiNSCvCCe2RhIybj0Z4cgJR
YtnoqyhgFKYnYh6knFDLWKQu73+cmw7mRuQzM/WnU+o7RoTSeQq2vT29kerTmHYG
XXhwKQx1MY56zoCb+xCPNzVbQe/iB41FcFShgltuhX+F1/I/xJqtSAhk4eNQlE7O
qbFOnYghMl7yS6vEQXaCCqYIHCI+AFKkBrN4AghIAUtFW4ZwnMePhOunCp+amGTy
dwaPM7Bkcz4aVcUCAwEAAaNdMFswHwYDVROjBBgwFoAUi2b2cOkL1ZshQVI4jyep
MonS/3MwDAYDVROTAQH/BAIwADALBgNVHQ8EBAMCBkAwHQYDVROOBBYEFNp+pZDV
2lSAaGDUJ7Pn+VfwVF1XMAOGCSqGSIb3DQEBCwUAA4IBAQBNI3T3U4SoUXBhwXCa
+iyuTHP67fRtmtZX5V+KNFniNDpuyptphOhv4vqaVQHq/PBA+wKrSPR9pOGrBY7F
AzRbocRHaXk+VhHa72UPcLVCs8+Tvjgc6T7cYI2mEuxbnXDVJmBbxXUapts8lRrf
PT7LZxDbkIbF9zSCgjWpOO2r/E2yWurdnwQkGDYA8LPfatge/4bQF2vHCEW4giOS
cOdOElwZSV74UDM2g+/C8AcgH5EH/XP6FvPLkxOVhVoYjsPjDgCJFTp+ciDtWkKI
lJFUD2+O/BiaN2InrNSY86nupgh5xf3+9lrwLqeBmWXXWVk58oDNS3crDpkWAVaq
dN8z</ds:X509Certificate>
</ds:X509Data>
          </ds:KeyInfo>
```

3. **SignedProperties** (with the SigningTime),

```
<ds:Object>
  <xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" Target=
    <xades:SignedProperties Id="xmldsig-03f6a207-0933-4f24-8c24-58674efa8c7a-SignedPr
      <xades:SignedSignatureProperties>
        <xades:SigningTime>2016-10-04T23:59:58Z</xades:SigningTime>
      </xades:SignedSignatureProperties>
    </xades:SignedProperties>
  </xades:QualifyingProperties>
</ds:Object>
```

2.2 The message digests that are to be signed

SignedInfo contains message digests (*DigestValue*, hash values) calculated on each of the (three) parts that are to be locked,

```
<ds:SignedInfo>
  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
  <ds:Reference Type="http://www.w3.org/2000/09/xmldsig#Object" URI="#xmldsig-03f6a207-0933-4f24-8
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>pLCUF8UL4ja3TVV1m17miE8G18SepyKAxioLVvbiyzQ=</ds:DigestValue>
  </ds:Reference>
  <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties" URI="#xmldsig-03f6a207-0933-4f24
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>Q1qEtDIjOWZBuKbvVZIP4c2OK6Ldq7hDceiJ8333MnQ=</ds:DigestValue>
  </ds:Reference>
  <ds:Reference Type="http://www.w3.org/2000/09/xmldsig#KeyInfo" URI="#xmldsig-03f6a207-0933-4f24-
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <ds:DigestValue>t4xhEROgYkZTymxlCZa7p61aRdqiC2prbckLCcLYH7A=</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>
```

2.3 The electronic signature

The electronic signature ***SignatureValue***, consisting of a single message digest calculated on *SignedInfo* (and thereby locking the referred subdocuments) and then encrypted using the private key using *SignatureMethod*,

```
          <ds:SignatureValue>T+ymEu3cIyv6X52sx1F3yvUPuJJyPtSncra//bs5Puh1Wq8rDsv62pS1FwUy9Yvg
4nTnJN1rMF8j1Zb9ksiNbLMCCLgfuY4OI5h+vkB+ZQRAZD8RUOasye7bRRIzgFkg
yZDK7bwABq/OI17ZU5o77D22mNX6j647C53CIOSmQAYry4ycBfoMCPF25LIWJkRO
GC8bOByiaOIY7Qi7D4jymME2WS2SXng8xpOjBYOflFDsI81Xpgt+hVtTWnGh159m
1qfSV84ZSRKWoMeyXbPC/WyTR9rH4azaoqSH4fJsfO6UQFCzqBRXtq+dNikH6764
FDXX5x7BG4KJ3ODVzaVOHQ==</ds:SignatureValue>
```

3 Check and calculate signature using xmllint and openssl

You can use your favourite text editor to view the example file `ENV-Envelope-UseCase1.xml`. You can also use the following command for an easier to read format:

```
xmllint --format ENV-Envelope-UseCase1.xml
```

3.1 Exclusive XML Canonicalization

Before message digests are calculated, the corresponding subdocument must be canonicalized, for example the document must be encoded in UTF-8, line breaks must be

normalized to #xA and so on. Any changes after canonicalization and before message digest calculation destroys the signature, even white space and line break changes.

In the examples, exclusive XML canonicalization is made with `xmllint --exc-c14n`. Since this is done on the complete file instead of on the subdocument, namespace must be propagated down according to the following example to make the message digest calculation correct:

Original and also after canonicalization of complete file:
```
<ds:SignedInfo>
```

After modification:
```
<ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

3.2 Calculate and compare message digests for subdocuments

The message digests for each of the subdocuments are calculated and compared with the values stored in subdocument SignedInfo.

3.2.1 Message digest for subdocument Payload

Subdocument Payload contains the business message.

Unfortunately, only exclusive canonicalization "#WithComments" can be achieved with the program xmllint, which means that the comment in the example file must be removed (<\!--.*-->).

3.2.1.1 Calculate the message digest for subdocument Payload

Extract the canonicalized subdocument and calculate the SHA-256 message digest.

```
xmllint --exc-c14n ENV-Envelope-UseCase1.xml | sed 's+<ds:Object
+<ds:Object xmlns:ds="http://www.w3.org/2000/09/xmldsig#" +' | awk
'/<ds:Object .*Payload/,/<\/ds:Object>/' | sed 's/.*<ds:Object
/<ds:Object /' | sed 's/<\/ds:Object>.*/<\/ds:Object>/' | sed
's/<\!--.*-->//' | xmllint --exc-c14n - | openssl dgst -sha256
```

Explanation of command line

1. canonicalize the XML
   ```
   xmllint --exc-c14n ENV-Envelope-UseCase1.xml
   ```

2. Modify namespace
   ```
   sed 's+<ds:Object +<ds:Object
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#" +'
   ```

3. Extract subdocument Payload
   `awk '/<ds:Object .*Payload/,/<\/ds:Object>/'`

4. Remove everything up to the start tag for the subdocument
   `sed 's/.*<ds:Object /<ds:Object /'`

5. Remove everyting after the end tag for the subdocument
   `sed 's/<\/ds:Object>.*/<\/ds:Object>/'`

6. Remove the comment but not white space surrounding the comment
   (xmllint only supports exclusive canonicalization with comments)
   `sed 's/<\!--.*-->//'`

7. Remove last trailing newline in character XML file
   (because text processing utilities are used in the example)
   `xmllint --exc-c14n -`

8. Calculate the SHA-256 message digest
   `openssl dgst -sha256`

Result:

```
(stdin)=
a4b09417c50be236b74d55659b5ee6884f0697c49ea72280c62a0b56f6e2cb34
```

3.2.1.2 Extract the message digest for subdocument Payload from SignedInfo

Extract the message digest that is saved in the signed subdocument SignedInfo. This will later be compared to the calculated message digest.

```
awk '/<ds:Reference .*Payload/,/<\/ds:Reference>/' ENV-Envelope-
UseCase1.xml | awk '/<ds:DigestValue/,/<\/ds:DigestValue/' | sed
's/.*<ds:DigestValue>//' | sed 's/<\/ds:DigestValue>.*//' | openssl
enc -d -a -A | xxd -p -c256
```

(The command `openssl enc -d -a -A` converts the value from Base64-encoded to binary format, the command `xxd -p -c256` converts the value from binary to hexadecimal format.)

Result:

```
a4b09417c50be236b74d55659b5ee6884f0697c49ea72280c62a0b56f6e2cb34
```

3.2.1.3 Compare the message digest for subdocument Payload

Compare the message digests from the steps above,

```
(stdin)=
a4b09417c50be236b74d55659b5ee6884f0697c49ea72280c62a0b56f6e2cb34
```

and

```
a4b09417c50be236b74d55659b5ee6884f0697c49ea72280c62a0b56f6e2cb34
```

Since the calculated message digest for subdocument Payload is equal to the specified value in SignedInfo, Payload is not changed since SignedInfo was calculated.

3.2.2 Message digest for subdocument KeyInfo

Similar to the calculation for subdocument Payload, but this contains the used signature certificate.

3.2.2.1 Calculate the message digest for subdocument KeyInfo

Extract the canonicalized subdocument and calculate the SHA-256 message digest.

```
xmllint --exc-c14n ENV-Envelope-UseCase1.xml | sed
's+<ds:KeyInfo+<ds:KeyInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"+' | awk
'/<ds:KeyInfo/,/<\/ds:KeyInfo>/' | sed
's/.*<ds:KeyInfo/<ds:KeyInfo/' | sed
's/<\/ds:KeyInfo>.*/<\/ds:KeyInfo>/' | xmllint --exc-c14n - |
openssl dgst -sha256
```

Result:

```
(stdin)=
b78c611113a0624653ca6c650996bba7a95a45daa20b6a6b6dc90b09c2d81fb0
```

3.2.2.2 Extract the message digest for subdocument KeyInfo from SignedInfo

Extract the message digest that is saved in the signed subdocument SignedInfo. This will later be compared to the calculated message digest.

```
awk '/<ds:Reference .*KeyInfo/,/<\/ds:Reference>/' ENV-Envelope-
UseCase1.xml | awk '/<ds:DigestValue/,/<\/ds:DigestValue/' | sed
's/.*<ds:DigestValue>//' | sed 's/<\/ds:DigestValue>.*//' | openssl
enc -d -a -A | xxd -p -c256
```

Result:

```
b78c611113a0624653ca6c650996bba7a95a45daa20b6a6b6dc90b09c2d81fb0
```

3.2.2.3 Compare the message digest for subdocument KeyInfo

Compare the message digests from the steps above,

```
(stdin)=
b78c611113a0624653ca6c650996bba7a95a45daa20b6a6b6dc90b09c2d81fb0
```

and

```
b78c611113a0624653ca6c650996bba7a95a45daa20b6a6b6dc90b09c2d81fb0
```

Since the calculated message digest for subdocument KeyInfo is equal to the specified value in SignedInfo, KeyInfo is not changed since SignedInfo was calculated.

3.2.3 Message digest for subdocument SignedProperties

Similar to the calculation for subdocument Payload, but this contains the XAdES SigningTime.

3.2.3.1 Calculate the message digest for subdocument SignedProperties

Extract the canonicalized subdocument and calculate the SHA-256 message digest.

```
xmllint --exc-c14n ENV-Envelope-UseCase1.xml | sed
's+<xades:SignedProperties +<xades:SignedProperties
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" +' | awk
'/<xades:SignedProperties/,/<\/xades:SignedProperties>/' | sed
's/.*<xades:SignedProperties /<xades:SignedProperties /' | sed
's/<\/xades:SignedProperties>.*/<\/xades:SignedProperties>/' |
xmllint --exc-c14n - | openssl dgst -sha256
```

Result:

```
(stdin)=
435a84b43223396641b8a6ef55920fe1cd8e2ba2ddabb84371e889f37df73274
```

3.2.3.2 Extract the message digest for subdocument SignedProperties from SignedInfo

Extract the message digest that is saved in the signed subdocument SignedInfo. This will later be compared to the calculated message digest.

```
awk '/<ds:Reference .*SignedProperties/,/<\/ds:Reference>/' ENV-
Envelope-UseCase1.xml | awk '/<ds:DigestValue/,/<\/ds:DigestValue/'
| sed 's/.*<ds:DigestValue>//' | sed 's/<\/ds:DigestValue>.*//' |
openssl enc -d -a -A | xxd -p -c256
```

Resultat:

```
435a84b43223396641b8a6ef55920fe1cd8e2ba2ddabb84371e889f37df73274
```

3.2.3.3 Compare the message digest for subdocument SignedProperties

Compare the message digests from the steps above,

```
(stdin)=
435a84b43223396641b8a6ef55920fe1cd8e2ba2ddabb84371e889f37df73274
```

and

```
435a84b43223396641b8a6ef55920fe1cd8e2ba2ddabb84371e889f37df73274
```

Since the calculated message digest for subdocument SignedProperties is equal to the specified value in SignedInfo, SignedProperties is not changed since SignedInfo was calculated.

3.3 Verify message digest for SignedInfo and electronic signature

3.3.1 Calculate message digest for SignedInfo

SignedInfo contains message digests on and references to each of the parts that are to be locked.

Extract the canonicalized subdocument SignedInfo and calculate the SHA-256 message digest. This will later be compared to the message digest extracted from the electronic signature.

```
xmllint --exc-c14n ENV-Envelope-UseCase1.xml | sed
's+<ds:SignedInfo>+<ds:SignedInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">+' | awk
'/<ds:SignedInfo/,/<\/ds:SignedInfo>/' | sed
's/.*<ds:SignedInfo/<ds:SignedInfo/' | sed
's/<\/ds:SignedInfo>.*/<\/ds:SignedInfo>/' | xmllint --exc-c14n - |
openssl dgst -sha256
```

Result:

```
(stdin)=
a0b60f20fde784d34654989cf757b6a6052b6ffce096e646a111f97e48a9ab5a
```

3.3.2 Extract public key from KeyInfo

Extract the public key from the signature certificate that is saved in KeyInfo. The public key is needed to check the signature.

```
awk '/<ds:X509Certificate>/,/<\/ds:X509Certificate>/' ENV-Envelope-
UseCase1.xml | sed 's/.*<ds:X509Certificate>//' | sed
's/<\/ds:X509Certificate>.*$//' | awk 'BEGIN {print "-----BEGIN
CERTIFICATE-----"} END {print "-----END CERTIFICATE-----"} {print}'
| fold -b -w 66 | openssl x509 -noout -pubkey > cert.pub
```

Contents of cert.pub, the public key:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv2eNX34iwor7/Huh7TBJ
144swczaFnjKDcu3qt9nx86EZ2m/aPW6iBYzh2ZO1ILaiAni9nI7Or74WJlmbA52
3qFrabNZZ3pW/dYqDI7RIufSdY59KI1IK8IJ7ZGEjJuPRnhyAlFi2eirKGAUpidi
HqScUMtYpC7vf5ybDuZG5DMz9adT6jtGhNJ5Cra9Pb2R6tOYdgZdeHApDHUxjnrO
```

```
gJv7EI83NVtB7+IHjUVwVKGCW26Ff4XX8j/Emq1ICGTh41CUTs6psU6diCEyXvJL
q8RBdoIKpggcIj4AUqQGs3gCCEgBS0VbhnCcx4+HS6cKn5qYZPJ3Bo8zsGRzPhpV
xQIDAQAB
-----END PUBLIC KEY-----
```

3.3.3 Show signature certificate (if wanted)

Information from the signature certificate can be shown using the following commands:

```
awk '/<ds:X509Certificate>/,/<\/ds:X509Certificate>/' ENV-Envelope-
UseCase1.xml | sed 's/.*<ds:X509Certificate>//' | sed
's/<\/ds:X509Certificate>.*$//' | awk 'BEGIN {print "-----BEGIN
CERTIFICATE-----"} END {print "-----END CERTIFICATE-----"} {print}'
| fold -b -w 66 | openssl x509 -text -nameopt multiline
```

Result:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            13:86:c7:b8:db:be:52:ce:44:85:aa:38:2a:70:95:b5
        Signature Algorithm: sha256WithRSAEncryption
        Issuer:
            countryName               = SE
            organizationName          = Tullverket
            organizationalUnitName    = Swedish Customs
            organizationalUnitName    = TEST Public Intermediate Certificate
Authority
            organizationalUnitName    = For testing purposes only
            serialNumber              = SE2021000969
            commonName                = Swedish Customs TEST Public CA 0.1
        Validity
            Not Before: Sep 14 07:23:24 2015 GMT
            Not After : Sep 14 07:23:24 2035 GMT
        Subject:
            countryName               = SE
            organizationName          = Testf\F6retag
            organizationalUnitName    = IT department
            serialNumber              = SE9999999999
            commonName                = Test company for signature
validation tests
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:bf:67:8d:5f:7e:22:c2:8a:fb:fc:7b:a1:ed:30:
                    49:d7:8e:2c:c1:cc:da:16:78:ca:0d:cb:b7:aa:df:
                    67:c7:ce:84:67:69:bf:68:f5:ba:88:16:33:87:66:
                    4e:d4:82:da:88:09:e2:f6:72:3b:3a:be:f8:58:99:
```

```
                    66:6c:0e:76:de:a1:6b:69:b3:59:67:7a:56:fd:d6:
                    2a:0c:8e:d1:22:e7:d2:75:8e:7d:28:8d:48:2b:c2:
                    09:ed:91:84:8c:9b:8f:46:78:72:02:51:62:d9:e8:
                    ab:28:60:14:a6:27:62:1e:a4:9c:50:cb:58:a4:2e:
                    ef:7f:9c:9b:0e:e6:46:e4:33:33:f5:a7:53:ea:3b:
                    46:84:d2:79:0a:b6:bd:3d:bd:91:ea:d3:98:76:06:
                    5d:78:70:29:0c:75:31:8e:7a:ce:80:9b:fb:10:8f:
                    37:35:5b:41:ef:e2:07:8d:45:70:54:a1:82:5b:6e:
                    85:7f:85:d7:f2:3f:c4:9a:ad:48:08:64:e1:e3:50:
                    94:4e:ce:a9:b1:4e:9d:88:21:32:5e:f2:4b:ab:c4:
                    41:76:82:0a:a6:08:1c:22:3e:00:52:a4:06:b3:78:
                    02:08:48:01:4b:45:5b:86:70:9c:c7:8f:87:4b:a7:
                    0a:9f:9a:98:64:f2:77:06:8f:33:b0:64:73:3e:1a:
                    55:c5
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:

keyid:8B:66:F6:70:E9:0B:D5:9B:21:41:52:38:8F:27:A9:32:89:D2:FF:73

            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage:
                Non Repudiation
            X509v3 Subject Key Identifier:
                DA:7E:A5:90:D5:DA:54:80:68:60:D4:27:B3:E7:F9:57:F0:54:5D:57
    Signature Algorithm: sha256WithRSAEncryption
         4d:23:74:f7:53:84:a8:51:70:61:c1:70:9a:fa:2c:ae:4c:73:
         fa:ed:f4:6d:9a:d6:57:e5:5f:8a:34:59:e2:34:3a:6e:ca:9b:
         69:87:48:6f:e2:fa:9a:55:01:ea:fc:f0:40:fb:02:ab:48:f4:
         7d:a7:41:ab:05:8e:c5:03:34:5b:a1:c4:47:69:79:3e:56:11:
         da:ef:65:0f:70:b5:42:b3:cf:93:be:38:1c:e9:3e:dc:60:8d:
         a6:12:ec:5b:9d:70:d5:26:60:5b:c5:75:1a:a6:db:3c:95:1a:
         df:3d:3e:cb:67:10:db:90:86:c5:f7:34:82:82:35:a9:38:ed:
         ab:fc:4d:b2:5a:ea:dd:9f:04:24:18:36:00:f0:b3:df:6a:d8:
         1e:ff:86:d0:17:6b:c7:08:45:b8:82:2d:12:70:e7:4e:12:5c:
         19:49:5e:f8:50:33:36:83:ef:c2:f0:07:20:1f:91:07:fd:73:
         fa:16:f3:cb:93:13:95:85:5a:18:8e:c3:e3:0e:00:89:15:3a:
         7e:72:20:ed:5a:42:88:94:91:54:0f:6f:b4:fc:18:9a:37:62:
         27:ac:d4:98:f3:a9:ee:a6:08:79:c5:fd:fe:f6:5a:f0:2e:a7:
         81:99:65:d7:59:59:39:f2:80:cd:4b:77:2b:0e:99:16:01:56:
         aa:74:df:33
-----BEGIN CERTIFICATE-----
MIIETzCCAzegAwIBAgIQE4bHuNu+Us5Ehao4KnCVtTANBgkqhkiG9w0BAQsFADCB
3TELMAkGA1UEBhMCU0UxEzARBgNVBAoMClR1bGx2ZXJrZXQxGDAWBgNVBAsMD1N3
ZWRpc2ggQ3VzdG9tczE3MDUGA1UECwwuVEVTVCBQdWJsaWMgSW50ZXJtZWRpYXRl
IENlcnRpZmljYXRlIEF1dGhvcml0eTEiMCAGA1UECwwZRm9yIHRlc3RpbmcgcHVy
cG9zZXMgb25seTEVMBMGA1UEBRMMU0UyMDIxMDAwOTY5MSswKQYDVQQDDCJTd2Vk
aXNoIEN1c3RvbXMgVEVTVCBQdWJsaWMgQ0EgMC4xMB4XDTE1MDkxNDA3MjMyNFoX
DTM1MDkxNDA3MjMyNFowgYkxCzAJBgNVBAYTAlNFMRUwEwYDVQQKDAxUZXN0ZsO2
```

```
cmV0YWcxFjAUBgNVBAsMDUlUIGRlcGFydG1lbnQxFTATBgNVBAUTDFNFOTk5OTk5
OTk5OTE0MDIGA1UEAwwrVGVzdCBjb21wYW55IGZvciBzaWduYXR1cmUgdmFsaWRh
dGlvbiB0ZXN0czCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAL9njV9+
IsKK+/x7oe0wSdeOLMHM2hZ4yg3Lt6rfZ8fOhGdpv2j1uogWM4dmTtSC2ogJ4vZy
Ozq++FiZZmwOdt6ha2mzWWd6Vv3WKgyO0SLn0nWOfSiNSCvCCe2RhIybj0Z4cgJR
YtnoqyhgFKYnYh6knFDLWKQu73+cmw7mRuQzM/WnU+o7RoTSeQq2vT29kerTmHYG
XXhwKQx1MY56zoCb+xCPNzVbQe/iB41FcFShgltuhX+F1/I/xJqtSAhk4eNQlE7O
qbFOnYghMl7yS6vEQXaCCqYIHCI+AFKkBrN4AghIAUtFW4ZwnMePh0unCp+amGTy
dwaPM7Bkcz4aVcUCAwEAAaNdMFswHwYDVR0jBBgwFoAUi2b2cOkL1ZshQVI4jyep
MonS/3MwDAYDVR0TAQH/BAIwADALBgNVHQ8EBAMCBkAwHQYDVR0OBBYEFNp+pZDV
2lSAaGDUJ7Pn+VfwVF1XMA0GCSqGSIb3DQEBCwUAA4IBAQBNI3T3U4SoUXBhwXCa
+iyuTHP67fRtmtZX5V+KNFniNDpuyptph0hv4vqaVQHq/PBA+wKrSPR9p0GrBY7F
AzRbocRHaXk+VhHa72UPcLVCs8+Tvjgc6T7cYI2mEuxbnXDVJmBbxXUapts8lRrf
PT7LZxDbkIbF9zSCgjWpOO2r/E2yWurdnwQkGDYA8LPfatge/4bQF2vHCEW4gi0S
cOdOElwZSV74UDM2g+/C8AcgH5EH/XP6FvPLkxOVhVoYjsPjDgCJFTp+ciDtWkKI
lJFUD2+0/BiaN2InrNSY86nupgh5xf3+9lrwLqeBmWXXWVk58oDNS3crDpkWAVaq
dN8z
-----END CERTIFICATE-----
```

3.3.4 Extract message digest from signature and check signature cryptographically

Extract the binary electronic signature to file signaturevalue.bin:

```
awk '/<ds:SignatureValue>/,/<\/ds:SignatureValue>/' ENV-Envelope-
UseCase1.xml | sed 's/.*<ds:SignatureValue>//' | sed
's/<\/ds:SignatureValue>.*$//' | tr -d '\n' | openssl enc -d -a -A -
out signaturevalue.bin
```

Verify the electronic signature cryptographically and extract the message digest in the same time from the electronic signature:

```
openssl rsautl -verify -inkey cert.pub -in signaturevalue.bin -pubin
-asn1parse
```

Result:

```
    0:d=0  hl=2 l=  49 cons: SEQUENCE
    2:d=1  hl=2 l=  13 cons:  SEQUENCE
    4:d=2  hl=2 l=   9 prim:   OBJECT            :sha256
   15:d=2  hl=2 l=   0 prim:   NULL
   17:d=1  hl=2 l=  32 prim:  OCTET STRING
      0000 - a0 b6 0f 20 fd e7 84 d3-46 54 98 9c f7 57 b6 a6   ...
....FT...W..
      0010 - 05 2b 6f fc e0 96 e6 46-a1 11 f9 7e 48 a9 ab 5a
.+o....F...~H..Z
```

Since the verification can be done without error, the specified signature certificate is used to create the electronic signature.

Checks must also be done that the certificate is issued by a trusted certificate authority and that the certificate is currently valid. (Swedish Customs does not currently publish certificate revocation lists on the Internet.)

For an easier to read format of the message digest from the electronic signature, you can use the following command:

```
openssl rsautl -verify -inkey cert.pub -in signaturevalue.bin -pubin
-asn1parse | grep '-' | cut -b14-60 | tr '-' ' ' | xxd -r -p | xxd -
p -c256
```

Result:

```
a0b60f20fde784d34654989cf757b6a6052b6ffce096e646a111f97e48a9ab5a
```

3.3.5 Compare message digest for SignedInfo with message digest from signature

Compare the calculated message digest for subdocument SignedInfo,

```
(stdin)=
a0b60f20fde784d34654989cf757b6a6052b6ffce096e646a111f97e48a9ab5a
```

and the extracted message digest from the electronic signature:

```
a0b60f20fde784d34654989cf757b6a6052b6ffce096e646a111f97e48a9ab5a
```

Since the calculated message digest for subdocument SignedInfo is equal to the extracted value from the electronic signature, the electronic signature is correct.

3.4 Create electronic signature

The message digests for each of the subdocuments are calculated in the same way as when checking the integrity of the subdocuments and are saved in subdocument SignedInfo. In this example, SignedInfo must be manually edited.

When SignedInfo is complete, message digest for SignedInfo can be calculated and later signed with the private key.

3.4.1 Calculate message digest for SignedInfo

This is done in the same way as when checking the signature.

SignedInfo contains message digests on and references to each of the parts that are to be locked.

Extract the canonicalized subdocument SignedInfo and calculate the SHA-256 message digest and save it in binary format. This will later be used to create the electronic signature.

In this example, the original `ENV-Envelope-UseCase1.xml` is used, ignoring the existing signature.

```
xmllint --exc-c14n ENV-Envelope-UseCase1.xml | sed
's+<ds:SignedInfo>+<ds:SignedInfo
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">+' | awk
'/<ds:SignedInfo/,/<\/ds:SignedInfo>/' | sed
's/.*<ds:SignedInfo/<ds:SignedInfo/' | sed
's/<\/ds:SignedInfo>.*/<\/ds:SignedInfo>/' | xmllint --exc-c14n - |
openssl dgst -sha256 -sha256 -binary > SignedInfo_binary_digest.bin
```

3.4.2 Sign binary message digest for SignedInfo

Sign the binary message digest for subdocument SignedInfo:

```
openssl pkeyutl -sign -inkey testcompany.key -pkeyopt digest:sha256
< SignedInfo_binary_digest.bin > new_signaturevalue.bin
```

Base64-encode the binary signature for inclusion in the SignatureValue element:

```
openssl enc -base64 < new_binary_signature
```

Result:

```
T+ymEu3cIyv6X52sx1F3yvUPuJJyPtSncra//bs5Puh1Wq8rDsv62pS1FwUy9Yvg
4nTnJNlrMF8jlZb9ksiNbLMCCLgfuY4OI5h+vkB+ZQRAZD8RUOasye7bRRIzgFkg
yZDK7bwABq/0Il7ZU5o77D22mNX6j647C53CIOSmQAYry4ycBfoMCPF25LIWJkR0
GC8bOByia0IY7Qi7D4jymME2WS2SXng8xp0jBYOflFDsI81Xpgt+hVtTWnGh159m
1qfSV84ZSRKWoMeyXbPC/WyTR9rH4azaoqSH4fJsf06UQFCzqBRXtq+dNikH6764
FDXX5x7BG4KJ30DVzaVOHQ==
```

This is exactly the same SignatueValue content as in the original example file, which is expected as the same data is signed and the used signature algorithm RSASSA-PKCS1-v1_5 (http://www.w3.org/2001/04/xmldsig-more#rsa-sha256) is deterministic.

4 Check and create signature using xmlsec1

This is just an example of a tool that can be used to check and create XML signatures.

"*xmlsec1*" is a command line utility included in the XML Security Library C library, https://www.aleksey.com/xmlsec/index.html.

Note that version 1.2.20, for example included in Red Hat Enterprise Linux 7, does not work when used in the examples below. Version 1.2.21 and newer does work, for example 1.2.25 included in Red Hat Enterprise Linux 8.

The test certificate hierarchy referenced in chapter 1 above is used in the following examples.

4.1 Check signature using xmlsec1

```
xmlsec1 --verify --trusted-pem TESTRootCA0.1.crt --trusted-pem
TESTPublicCA0.1.crt ENV-Envelope-UseCase1.xml
```

Result:

```
OK
SignedInfo References (ok/all): 3/3
Manifests References (ok/all): 0/0
```

4.2 Create signature using xmlsec1

To create the signature, xmlsec1 uses a template file, where the variable elements (DigestValue, SignatureValue and X509Certificate) must be emptied.

4.2.1 Create XML template file

Based on the example file ENV-Envelope-UseCase1.xml, empty the elements DigestValue, SignatureValue and X509Certificate that are to be calculated by the signature process.

Relevant parts before change:

```
              <ds:DigestValue>pLCUF8UL4j...VvbiyzQ=</ds:DigestValue>
              <ds:DigestValue>Q1qEtDIjOW...8333MnQ=</ds:DigestValue>
              <ds:DigestValue>t4xhEROgYk...CcLYH7A=</ds:DigestValue>
          <ds:SignatureValue>T+ymEu3...
FDXX5x7BG4KJ30DVzaVOHQ==</ds:SignatureValue>
            <ds:X509Certificate>MIIETzCCAzegAwIBAgIQE4bH...
dN8z</ds:X509Certificate>
```

Relevant parts after change:

```
            <ds:DigestValue/>
            <ds:DigestValue/>
            <ds:DigestValue/>
        <ds:SignatureValue/>
          <ds:X509Certificate/
```

4.2.2 Sign the template file

If no changes are made outside the elements DigestValue, SignatureValue and X509Certificate (including white space), the resulting file ENV-Envelope-UseCase1_signed_again.xml will be exactly equal to ENV-Envelope-UseCase1.xml. In the example, the private key is unencrypted.

```
xmlsec1 --sign --output ENV-Envelope-UseCase1_signed_again.xml --
privkey-pem testcompany.key,testcompany.crt ENV-Envelope-
UseCase1_unsigned_template.xml
```