Tullverket

Example to manually check and create EDIFACT signatures

2022-11-18

Contents

1 Introduction

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 or the certificates is made and the commands will not work with arbitrary EDIFACT files. The examples presume that there is only one single CUSDEC message and an AUTACK message in the exchange.

The term *message digest* is used in this document for *hash value* or *cryptographic checksum*.

1.1 Links to files used in the examples

Links to example EDIFACT files and shell script can be found at “EDI – Security Concept” (English) https://www.tullverket.se/edisecurity and “EDI – Säkerhet” (Swedish) https://www.tullverket.se/edisakerhet.

1.2 Links to specifications

Technical specifications SCTS-SC
https://www.tullverket.se/foretag/ansokochdeklarera/deklareradigitalt/ediforsystemutvecklare/editekniskaspecifikationer/sctssc

2 Concept

One exchange (one file) can contain two or more messages, of which at least one CUSDEC or CUSRES message and exactly one AUTACK message.

The AUTACK message contains electronic signatures of each message, referring to each message in the exchange.

The following EDIFACT exchange with one CUSDEC message and one AUTACK message is used in the example. Line breaks are added for increased readability but beaks the electronic signature.

```
UNA:+.? '
UNB+UNOC:3+9999999999+2021000969+220302:0920+ITX22030220366+ChangeThis+CUSDECI-
IE++++1'
UNH+MESREF-000001+CUSDEC:D:96B:UN:145050'
BGM+FRA'
RFF+ZTI:TJ37183522'
RFF+AEI:TJ30158935'
RFF+AEI:TJ30158893'
RFF+AEI:TJ30158901'
RFF+CLU:Signature test ääöÅÄÖ'
UNS+D'
UNS+S'
UNT+10+MESREF-000001'
UNH+AUTMESSREF0001+AUTACK:4:1:UN:SEA02A'
USH+7+SRN01+3++2++1+ZH1::SE9999999999++1:20220302:092003'
USA+1:::48'
USC+E4bHuNu?+Us5Ehao4KnCVtQ==+4:i2b2cOkL1ZshQVI4jyepMonS/3M=+3'
USA+6:::10'
USX+ITX22030220366++++++MESREF-000001+CUSDEC:D:96B:UN'
USY+SRN01+ZS3:de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114'
USY+SRN01+ZS4:9440e74f0bbeac8761010a4008d41b55c4bf71bab3f06165aea93821afcd16301dd17bd
4d941e410d0a8a389a77eb6843895a016b2ee9e173446b83e59c427e99433abab7d7398e753f9950f23c7
874f7b9406b0eddbae904c65796bf461f3c0e358e080929f81d90f97dfe89eb1ae74bf8292949e804f2c8
327e5c3e5ba783e1e76f2151a3d0737f1665a8303badbdccd2ec443cdf434df7f7ddf97940c45db7a42c1
fb35368ad09c34cac26f69595aa8d96d481850d708d9842ce4d30313e4d9e5cf5f15d5f84dc7615e5311b
```

```
d0001c2d1a77d17ab753fbd920bc2a0a971f036ed767d71d50de5a2709baa0e5f12b036ec7a812640d1fa
b3233c104683cfff'
UNT+9+AUTMESSREF0001'
UNZ+2+ITX22030220366'
```

2.1 The data that are to be signed

The electronic signature is calculated on each message, in this case a CUSDEC message.

Line breaks are added for increased readability but beaks the electronic signature. The signature is calculated from the beginning of the UNH segment, including UNH, to the end of the UNT segment, including the apostrophe sign.

```
UNH+MESREF-000001+CUSDEC:D:96B:UN:145050'
BGM+FRA'
RFF+ZTI:TJ37183522'
RFF+AEI:TJ30158935'
RFF+AEI:TJ30158893'
RFF+AEI:TJ30158901'
RFF+CLU:Signature test ääöÄÄÖ'
UNS+D'
UNS+S'
UNT+10+MESREF-000001'
```

The message reference in the example is MESREF-000001

2.2 The electronic signature

In the AUTACK message, the segment group consisting of one USX and two USY segments, is repeated for each CUSDEC or CUSRES message.

The USX message uniquely references the corresponding message, that must be in the same exchange.

The USY segments contain the actual signature.

```
UNH+AUTMESSREF0001+AUTACK:4:1:UN:SEA02A'
USH+7+SRN01+3++2++1+ZH1::SE9999999999++1:20220302:092003'
USA+1:::48'
USC+E4bHuNu?+Us5Ehao4KnCVtQ==+4:i2b2cOkL1ZshQVI4jyepMonS/3M=+3'
USA+6:::10'
USX+ITX22030220366++++++MESREF-000001+CUSDEC:D:96B:UN'
USY+SRN01+ZS3:de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114'
USY+SRN01+ZS4:9440e74f0bbeac8761010a4008d41b55c4bf71bab3f06165aea93821afcd16301dd17bd
4d941e410d0a8a389a77eb6843895a016b2ee9e173446b83e59c427e99433abab7d7398e753f9950f23c7
874f7b9406b0eddbae904c65796bf461f3c0e358e080929f81d90f97dfe89eb1ae74bf8292949e804f2c8
327e5c3e5ba783e1e76f2151a3d0737f1665a8303badbdccd2ec443cdf434df7f7ddf97940c45db7a42c1
fb35368ad09c34cac26f69595aa8d96d481850d708d9842ce4d30313e4d9e5cf5f15d5f84dc7615e5311b
d0001c2d1a77d17ab753fbd920bc2a0a971f036ed767d71d50de5a2709baa0e5f12b036ec7a812640d1fa
b3233c104683cfff'
UNT+9+AUTMESSREF0001'
```

2.3 The certificate and private keys

The certificate used for the signature is not included in the exchange but a reference to the used certificate is included in the USC segment. To sign a message, both the certificate and the corresponding private key must be available. To verify a message, only the signature certificate (and the certificate hierarchy) is needed.

The Swedish Customs Signature Certificate, used to verify messages sent from Swedish Customs, can be downloaded from http://ca.tullverket.se/ca/TullverketEDIcurrent.crt.

Since Swedish Customs issues the Company Signature Certificates, these certificates are available to Swedish Customs to verify messages sent from companies to Swedish Customs.

In the example, a test certificate from a test certificate hierarchy is used.

3 Check and calculate signature using openssl

The examples assume that we have the file your_edifact_message.txt with the complete EDIFACT message and the file your_certificate.pem with the used Signature Certificate in PEM format.

3.1 Check the signature

The message digests for each CUSDEC or CUSRES message are calculated and compared with the values stored in the signature in the AUTACK USY segments.

3.1.1 Extract the signature from the EDIFACT file

The signature can be extracted from the corresponding USY segment, qualified by ZS4 (USX(*).USY[9](S508.0563=ZS4).S508.0560).

Extract the signature from the EDIFACT file:

```
grep '+ZS4:' your_edifact_message.txt | sed 's/^.*+ZS4://' | sed
"s/'.*//" > signature.hex
```

Convert the extracted hexadecimal signature to binary format.

```
xxd -r -p < signature.hex > signature.bin
```

3.1.2 Extract the public key from the certificate

In the example, we assume that we know which certificate is used. In a real system, you have to look up which certificate is used. (In the later signing example, we calculate the certificate references to be used in the USC segment.)

```
openssl x509 -in your_certificate.pem -pubkey -noout > cert.pub
```

3.1.3 Verify the signature cryptographically

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 signature.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 - de 4b 49 25 bf 84 35 cd-93 cb e3 e0 cd b9 d0 26   .KI%..5........&
      0010 - 0d 82 a0 2c 94 b3 05 60-6b 7b ed 28 4b 7c 61 14   ...,...`k{.(K|a.
```

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, but this will be done in the near future.)

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 signature.bin -pubin -
asn1parse| grep '-' | cut -b14-60 | tr '-' ' ' | xxd -r -p | xxd -p
-c256
```

Result:

```
de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114
```

3.1.4 Calculate message digest for message

Extract the data from the EDIFACT file that is supposed to be signed (UNH-UNT), in this example only one single CUSDEC message.

```
cat your_edifact_message.txt | sed 's/UNH/\x00/g' | sed
's/^.[^\x00]*\x00/UNH/' | sed "s/\x00.*//" | tr -d '\n' >
your_signed_data.txt
```

Calculate the SHA-256 hash

```
cat your_signed_data.txt | openssl dgst -sha256
```

Result:

```
(stdin)=
de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114
```

3.1.5 Extract the message digest from AUTACK

The stated message digest can be extracted from the corresponding USY segment, qualified by ZS3 (USX(*).USY[9](S508.0563=ZS3).S508.0560).

```
grep '+ZS3:' your_edifact_message.txt | sed 's/^.*+ZS3://' | sed
"s/'.*//"

de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114
```

The message digest in the AUTACK message must match the calculated message digest and the message digest in the signature.

3.1.6 Compare calculated message digest for message with message digest from signature

Compare the calculated message digest for the CUSDEC message,

```
(stdin)=
de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114
```

and the extracted message digest from the electronic signature:

```
de4b4925bf8435cd93cbe3e0cdb9d0260d82a02c94b305606b7bed284b7c6114
```

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

3.2 Create electronic signature

In this example, a small shell script, “create_AUTACK” is used. The different steps are explained in the shell script.

A “template file” with place holders for the variable parts is used in this example, but that is not a stable and secure solution to be used in a production environment.

Links to the files used in the example, including the used signature certificate and the corresponding private key, can be found in chapter 1.1 above.

3.2.1 Shell script “create_AUTACK”

```
#!/bin/sh

# The certificates used is this example are included in this zip-archive, but can
also be downloaded from
# curl -O https://ftgtest-
meddelandevalidering.tullverket.se/valideringstjanst/validering/andrafiler/Swedish_Cu
stoms_TEST_CA_0.1.zip

# The paths for the certificate and private key used in this script will be correct
using
# unzip Swedish_Customs_TEST_CA_0.1.zip

# Unsigned template file
EDIFACTFILEIN="EDIFACT_unsigned_template"
# The private key necessary for signing
PRIVATEKEY="Swedish_Customs_TEST_CA_0.1_2015-09-14/testcompany.key"
# The certificate necessary for validation of signature and for extracting parameters
when signing
CERTIFICATE="Swedish_Customs_TEST_CA_0.1_2015-09-14/testcompany.crt"


# The resulting signed output file
EDIFACTFILEOUT="temp_EDIFACT_signed_output"

# Public key extracted from certificate
PUBLICKEY="temp_public_key"

# The extracted CUSDEC or CUSRES message (UNH to UNT') to be signed
EDIFACTTOSIGN="temp_EDIFACT_to_sign"

# Calculated binary message digest (hash value) to be used in the signature
HASHTOSIGN="temp_hash_to_sign"

# Calculated digital signature
SIGNATURE="temp_digital_signature"

echo
echo "1. Extract the EDIFACT message to be signed,"
echo "from the beginning of the UNH segment, including UNH, to the end of the UNT
segment, including the apostrophe sign."
echo
cat ${EDIFACTFILEIN} | sed 's/UNH/\x00/g' | sed 's/^.[^\x00]*\x00/UNH/' | sed
"s/\x00.*//" | tr -d '\n' > ${EDIFACTTOSIGN}
echo "EDIFACT exchange file: ${EDIFACTFILEIN}"
echo "EDIFACT message: ${EDIFACTTOSIGN}"
echo

echo
echo "2. Calculate message digest (hash value) on the EDIFACT message"
echo
```

```
cat ${EDIFACTTOSIGN} | openssl dgst -sha256 -binary > ${HASHTOSIGN}
HASHVALUE=`xxd -p -c 32 ${HASHTOSIGN}`
echo "Hash value (hexadecimal form):"
echo "${HASHVALUE}"
echo
echo "This value is to be used in the AUTACK message."
echo

echo
echo "Exampel on common error: Calculate *** WRONG *** hash value - DO *** NOT *** DO
THIS!"
WRONGHASHVALUE=`cat ${EDIFACTTOSIGN} | openssl dgst -sha256 -binary | xxd -p -c 32 |
tr -d '\n' | openssl dgst -sha256 -binary | xxd -p -c 32`
echo "*** WRONG - COMMON ERROR *** Hash value:"
echo "${WRONGHASHVALUE}"
echo
echo "This is a SHA-256 hash of the hexadecimal form of the correct SHA-256 hash. The
original binary hash must be used in the signature!"
echo

echo
echo "3. Sign the binary hash in ${HASHTOSIGN}, signature in: ${SIGNATURE}"
echo "Indicate in DigestInfo in the signature that the SHA256 algorithm has been
used"
openssl pkeyutl -sign -inkey ${PRIVATEKEY} -in ${HASHTOSIGN} -out ${SIGNATURE} -
pkeyopt digest:sha256
DIGITALSIGNATURE=`xxd -p ${SIGNATURE} | tr -d '\n'`
echo
echo "Digital signature:"
echo "${DIGITALSIGNATURE}"
echo
echo "This value is to be used in the AUTACK message."
echo

echo
echo "4. Calculate Certificate Reference, Base64 encoded certificate serial number,
may include '+' characters."
CERTIFICATEREFERENCE=`openssl x509 -noout -serial -in ${CERTIFICATE} | sed
's/serial=//' | tr -d '\n' | xxd -r -p | openssl enc -a`
echo "Certificate reference: ${CERTIFICATEREFERENCE}"
echo "Insert EDIFACT release character '?' in case of '+' characters, which can occur
in Base64 encoding."
CERTIFICATEREFERENCEEDIFACT=`echo -n "${CERTIFICATEREFERENCE}" | sed 's/+/\?+/g'`
echo
echo "Certificate reference EDIFACT: ${CERTIFICATEREFERENCEEDIFACT}"
echo
echo "This value is to be used in the AUTACK message."
echo

echo
echo "5. Calculate Key Name, Base64 encoded authority key identifier
(authorityKeyIdentifier[keyIdentifier]), may include '+' characters."
KEYNAME=`openssl x509 -noout -text -in ${CERTIFICATE} | grep keyid | sed
's/^.*keyid://' | xxd -r -p | openssl enc -a`
echo "Key Name: ${KEYNAME}"
echo "Insert EDIFACT release character '?' in case of '+' characters, which can occur
in Base64 encoding."
```

```
KEYNAMEEDIFACT=`echo -n "${KEYNAME}" | sed 's/+/\?+/g'`
echo
echo "Key Name  EDIFACT: ${KEYNAMEEDIFACT}"
echo
echo "This value is to be used in the AUTACK message."
echo

echo
echo "Extra step to verify the calculated signature"
echo "Extract public key from certificate"
openssl x509 -in ${CERTIFICATE} -pubkey -noout > ${PUBLICKEY}

echo
echo "Verify signature ${SIGNATURE}"
openssl rsautl -verify -inkey ${PUBLICKEY} -in ${SIGNATURE} -pubin -asn1parse

echo
echo "6. Crate the signed output file: ${EDIFACTFILEOUT}"
# Substituting variable values in a template file, as in this example, is generally
not a good and secure solution
eval "cat <<EOF
$(<${EDIFACTFILEIN})
EOF
" > ${EDIFACTFILEOUT}
```

3.2.2 Sample template file

Line breaks are added for increased readability but are not present when the electronic signature is calculated. The variable parts are ${CERTIFICATEREFERENCEEDIFACT}, ${KEYNAMEEDIFACT}, ${HASHVALUE} and ${DIGITALSIGNATURE}.

```
UNA:+.? '
UNB+UNOC:3+9999999999+2021000969+220302:0920+ITX22030220366+ChangeThis+CUSDECI-
IE++++1'
UNH+MESREF-000001+CUSDEC:D:96B:UN:145050'
BGM+FRA'
RFF+ZTI:TJ37183522'
RFF+AEI:TJ30158935'
RFF+AEI:TJ30158893'
RFF+AEI:TJ30158901'
RFF+CLU:Signature test ääöÅÄÖ'
UNS+D'
UNS+S'
UNT+10+MESREF-000001'
UNH+AUTMESSREF0001+AUTACK:4:1:UN:SEA02A'
USH+7+SRN01+3++2++1+ZH1::SE9999999999++1:20220302:092003'
USA+1:::48'
USC+${CERTIFICATEREFERENCEEDIFACT}+4:${KEYNAMEEDIFACT}+3'
USA+6:::10'
USX+ITX22030220366++++++MESREF-000001+CUSDEC:D:96B:UN'
USY+SRN01+ZS3:${HASHVALUE}'
USY+SRN01+ZS4:${DIGITALSIGNATURE}'
UNT+9+AUTMESSREF0001'
UNZ+2+ITX22030220366'
```