Specification of secure electronic communication between organisations in the insurance sector

(Version 1.1)
1. Introduction ........................................................................................................................................ 3
  1.1 Background .................................................................................................................................. 3
  1.2 Purpose ....................................................................................................................................... 3
  1.3 Notation .................................................................................................................................... 3
  1.4 Management of the specification ............................................................................................... 4
  1.5 Versions and backwards compatibility ...................................................................................... 4
  1.6 Affiliated organisations .............................................................................................................. 4
  1.7 Standards .................................................................................................................................. 5

2. Communication ................................................................................................................................. 6
  2.1 Communication flow .................................................................................................................. 6
  2.2 An error in the communication flow ........................................................................................ 8
  2.3 Handling TxId .......................................................................................................................... 8
    2.3.1 The sender’s responsibilities ............................................................................................. 8
    2.3.2 The receiver’s responsibilities .......................................................................................... 8

3. Messages .......................................................................................................................................... 9
  3.1 Namespaces ............................................................................................................................... 9
  3.2 Algorithms ................................................................................................................................. 9
  3.3 Certificates in signed messages .............................................................................................. 9
  3.4 Technical message header ....................................................................................................... 10
  3.5 Receipts .................................................................................................................................... 11
  3.6 Soap fault .................................................................................................................................. 13
  3.8 Example of an unsigned query message ................................................................................... 15

4. Security .......................................................................................................................................... 16
  4.1 Security levels ........................................................................................................................... 17
  4.2 Secure communication ............................................................................................................. 17
  4.3 Signing ..................................................................................................................................... 18
  4.4 CAs and certificates .................................................................................................................. 18
  4.5 Revoking of certificates .......................................................................................................... 18
  4.6 Approved CAs and certificate types ......................................................................................... 19
  4.7 Handling private keys ............................................................................................................ 19

5. Appendix A Standards .................................................................................................................... 20

6. Appendix B Web Services Security ................................................................................................ 21

7. Appendix C SSEK-specific schemes .............................................................................................. 22

8. Appendix D Fault codes ................................................................................................................. 24
  8.1 Faults concerning the document ............................................................................................ 24
  8.2 Faults concerning the transaction ID ....................................................................................... 24
  8.3 Faults on the server side .......................................................................................................... 24
  8.4 Logical faults in transaction data ............................................................................................ 24
  8.5 Faults concerning certificates or signatures ............................................................................. 25
  8.6 Other faults which are not reported via SOAP fault ............................................................... 25
1. Introduction

1.1 Background

SSEK (“Specifikation av säker elektronisk kommunikation mellan aktörer i försäkringsbranschen”, in English “Specification of secure electronic communication between organisations in the insurance sector”) fulfils the business need for a standard for secure electronic communication in the insurance sector. By following the specification, organisations will reduce the risk of having to build up several different communication solutions or of insecure communication solutions being constructed.

1.2 Purpose

This document is a specification of how electronic communication should be carried out securely over the Internet. The document is ideal for use as underlying information and could be included in agreements concerning how communication should be carried out between one’s own organisation and another organisation.

1.3 Notation

In this document, the keywords SHOULD, CAN, MUST and MUST NOT have the following meaning.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHOULD</td>
<td>The text being referred to is recommended but is not a requirement for fulfilling the specification.</td>
</tr>
<tr>
<td>CAN</td>
<td>The text being referred to can be used if the transactions require it but it is not a requirement for fulfilling the specification.</td>
</tr>
<tr>
<td>MUST</td>
<td>The text being referred to is a requirement for fulfilling the specification.</td>
</tr>
<tr>
<td>MUST NOT</td>
<td>The text being referred to is not permitted under the specification.</td>
</tr>
</tbody>
</table>
1.4 Management of the specification

SSEK is being managed with the SFM (“Svenska Försäkrings Mäklares förening”, in english “the Swedish Insurance Brokers’ Association”). The group that is handling the publication of the specification has representatives from the organisations Skandia Liv, SEB Trygg-Liv, Länsförsäkringar, Alecta, Aspispronia, Danica, Folksam, SPP and SFM.

1.5 Versions and backwards compatibility

This specification is not static and may be amended in order to adapt it to new or amended standards, e.g. concerning signature management. Implementations of the specification must be able to handle different versions of the specification by programmatically determining the version of SSEK for a particular incoming message. This is handled through giving TxHeader a new namespace for each version of SSEK, where the changes are such that they affect the implementations. This applies regardless of whether or not the actual content or structure of TxHeader has been altered.

For version 1.1 of the SSEK, the following namespace applies for TxHeader:
http://schemas.ssek.org/txheader/2003-04-03/

For any future version 1.2 of SSEK the namespace for TxHeader could be:
http://schemas.ssek.org/txheader/2004-01-12/

1.6 Affiliated organisations

The organisations that have adopted the specification have decided to communicate electronically in accordance with the SSEK and have a completed platform for SSEK or are in the process of developing such a platform. It will therefore be possible for communication with affiliated organisations to be carried out after a commercially acceptable period of time by both parties. We (the group for SSEK within SFM) also naturally encourage organisations that have not adopted the specification to use the SSEK for secure electronic communication.

The following organisations have adopted SSEK.

- Skandia Liv
- SEB Trygg-Liv
- Länsförsäkringar
1.7 Standards

This document, SSEK, describes what electronic communication between parties in the insurance sector should look like. SSEK is based on a number of existing standards, specifications and recommendations from acknowledged standardising bodies such as IETF (Internet Engineering Task Force), W3C (World Wide Web Consortium) and OASIS (Organisation for the Advancement of Structured Information Standards).

For information on which standards SSEK is based on, see Appendix A Standards and Appendix B Web Services Security.

The standards and specifications that are listed in Appendix A Standards are fully covered by SSEK with the exception of WS-Security (Web Services-Security) and therefore also XMLSignature, where selected parts are used in accordance with Appendix B Web Services Security.

Like SSEK, WS-Security is based on a collection of standards and describes how electronic communication is carried out securely. SSEK is however direct adapted to the needs for electronic communication in the insurance sector. Only the parts of the WS-Security specification for which there is a commercial need are therefore used.

A major advantage of supporting parts of WS-Security is that the products that are developed with support for WS-Security can be used. SSEK messages can therefore be created and handled using the frameworks that following the WS-Security specification.
2 Communication

In connection with electronic communication, the flow MUST be specified between the communicating parties. This section describes the flow of communication for synchronous and asynchronous communication.

All communication in accordance with SSEK takes place synchronously with a query message and a reply message in the form of SOAP messages. An asynchronous flow can be achieved by combining two synchronous transmissions. To separate the synchronous transmissions in an asynchronous flow, we will hereafter refer to these as the request and the result. In order to link together an asynchronous flow, i.e. connecting the request to the result, TxId is used in accordance with 3.4 Technical message header.

2.1 Communication flow

The following diagram and description show how the flow MUST look in the case of synchronous communication when a signature is used. The signature is not essential but is used depending on the business requirement.

1. Party A sends a signed query message for a request to Party B.
2. Part B verifies the signature and the format in the query message, processes the message and returns a signed result that describes how the processing went. In the event of an error that is discovered in connection with the reception, a soap fault message is returned in accordance with section 3.6 Soap fault.
The following diagram and description show how the flow MUST look in the case of **asynchronous** communication when a signature is used. The signature is not essential but is used depending on the business requirement.

1. Party A sends a signed query message with a unique TxId for a request to Party B.
2. Party B returns a signed receipt after having verified the signature and the format in the query message. Party A have then received proof that Party B has received the message. Note that Party B has **not** given any guarantees that the message in the query message can be processed. In the event of an error in connection with the reception, a soap fault message is returned in accordance with section 3.6 Soap fault.
3. After having processed the request, Party B sends a signed query message with the original TxId and a result that describes how the processing went to Party A.
4. Party A links the result to the request using the TxId (see section 3.4 Technical message header) and returns a signed receipt after having verified the signature and the format in the result received from Party B. In the event of an error in connection with receiving, a soap fault message is returned in accordance with section 3.6 Soap fault.
2.2 An error in the communication flow

Situations can arise where there is uncertainty concerning whether a query message has been received or processed. This can for example arise if the secure channel (the SSL tunnel) goes down after the receiver has received a query message but before the sender has received a reply message. In this situation, the sender of the query message may not know whether the receiver has received the query message.

If such a situation arises, the sender of the query message MUST contact the receiver to determine whether the message has been processed. The sender MUST NOT send the message again until he or she has contacted the receiver.

Despite the fact that no receipt has been received by the sender, the receiver of a query message can process the message and send a result and receive a receipt for the result. If this is a service which is defined by the fact that archiving must be carried out, the sender of the request in this situation will have an incomplete transaction in his or her archive. There will be no receipt for the request. When requested to do so by the other party, the receiver of the query message MUST send such a receipt so that both parties can archive the complete transaction.

When the sender has contacted the receiver and it is apparent that the query message for the request has not been processed or received by the other party, the sender MUST generate a new unique TxId and send the message again.

2.3 Handling TxId

Based on the above paragraph, the following MUST apply to the sender and the receiver of a query message.

2.3.1 The sender’s responsibilities

The sender of a query message for a request is responsible for generating a unique TxId. In the event of uncertainty as to whether the message has been received by the other party, the sender MUST contact the other party in order to determine the status of the transaction.

2.3.2 The receiver's responsibilities

The receiver of a message is responsible for checking whether the incoming TxId is unique. In the event of a duplicate, the receiver MUST return a soap fault message with a fault code “Client.TxHeader.TxId.Duplicate”. In the case of asynchronous processing, the receiver is responsible for ensuring that TxId flows unaltered throughout the entire transaction.
3 Messages
Messages in communication in accordance with SSEK follow the standard SOAP 1.1.

3.1 Namespaces
The following namespaces MUST be used when creating messages in accordance with SSEK.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
<th>Used in connection with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soap</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a></td>
<td>All cases</td>
</tr>
<tr>
<td>Txh</td>
<td><a href="http://schemas.ssek.org/txheader/2003-04-03/">http://schemas.ssek.org/txheader/2003-04-03/</a></td>
<td>All cases</td>
</tr>
<tr>
<td>R</td>
<td><a href="http://schemas.ssek.org/receipt/2003-04-03/">http://schemas.ssek.org/receipt/2003-04-03/</a></td>
<td>Receipt</td>
</tr>
</tbody>
</table>

The prefix is used here to clarify the example in the document.

3.2 Algorithms
The following algorithms are used in this document. These algorithms MUST be used.

<table>
<thead>
<tr>
<th>Use</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanonicalizationMethod</td>
<td><a href="http://www.w3.org/2001/10/xml-exc-c14n#">http://www.w3.org/2001/10/xml-exc-c14n#</a></td>
</tr>
<tr>
<td>SignatureMethod</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#rsa-sha1">http://www.w3.org/2000/09/xmldsig#rsa-sha1</a></td>
</tr>
<tr>
<td>Transform</td>
<td><a href="http://www.w3.org/2001/10/xml-exc-c14n#">http://www.w3.org/2001/10/xml-exc-c14n#</a></td>
</tr>
<tr>
<td>DigestMethod</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#sha1">http://www.w3.org/2000/09/xmldsig#sha1</a></td>
</tr>
</tbody>
</table>

3.3 Certificates in signed messages
In signed messages, the certificate that is used to create the signature MUST be attached to the message in accordance with the specification for WS-Security, see Appendix B Web Services Security. The attached certificate can be used by the receiver to validate the signature. The receiver of a signed message SHOULD check that the attached certificate is correct.
3.4 Technical message header

All messages MUST include a message header with technical information, TxHeader, in accordance with the scheme in Appendix C SSEK-specific schemes with the following namespace:

The purpose of this information is to control the messages, handle asynchronous communication and obtain a manageable log of communicated documents.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>SenderId</td>
<td>ID of the sender of the document.</td>
<td>Type: type of identity which is used for the sender of a message</td>
</tr>
<tr>
<td>ReceiverId</td>
<td>ID of the recipient of the document.</td>
<td>Type: type of identity which is used for the receiver of a message</td>
</tr>
<tr>
<td>TxId</td>
<td>UUID (Universal Unique Identifier in accordance with DCE).</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>Time when the message was created in accordance with the form yyyy-mm-ddThh:mm:ss, where T is constant.</td>
<td></td>
</tr>
</tbody>
</table>

The TxId in the message header is primarily used to keep an asynchronous transaction together. TxId is not obligatory but MUST be agreed between the communicating parties for each service based on the business need.

The type of identity that is used for SenderId and ReceiverId MUST be agreed between the communicating parties. The types which can be used are defined by the scheme for Txheader and are explained in the table below:

<table>
<thead>
<tr>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP</td>
<td>Application name, can be own clients where traceability is required concerning the application that was used.</td>
</tr>
<tr>
<td>CN</td>
<td>Common Name, obtained from the client certificate.</td>
</tr>
<tr>
<td>DN</td>
<td>Distinguished Name, the entire name for the client certificate.</td>
</tr>
<tr>
<td>ORGNR</td>
<td>Organisation number</td>
</tr>
</tbody>
</table>

An example of a technical message header, TxHeader, is given below:

```xml
<txh:TxHeader soap:mustUnderstand="1" xmlns:txh="http://schemas.ssek.org/txheader/2003-04-03/">
  <txh:SenderId txh:type="CN">Company A</txh:SenderId>
  <txh:ReceiverId txh:type="CN">Company B</txh:ReceiverId>
  <txh:TxId>C61B0B07-EF5F-46a1-92B4-6E5FA574E46E</txh:TxId>
  <txh:Timestamp>2003-03-27T12:50:00</txh:Timestamp>
</txh:TxHeader>
```
3.5 Receipts

The purpose of a receipt is to enable the sender to be certain that the query message has been received by the receiver. A receipt MUST be used in connection with asynchronous communication when a message is received and approved for processing by the receiver. However, a receipt does not guarantee that the information in the received message will be processed without any errors.

In connection with the acknowledgement of a signed message, the receipt MUST contain the signature of the acknowledged message. The signature contains a representation of the original message in the form of a hash value. Representation in the form of the signature from the received message can of course only be used if the received message has been signed.

The receipt can also be used to announce the result of the processing synchronously. The scheme allows company-specific information to be added.

The namespace for a receipt is http://schemas.ssek.org/receipt/2003-04-03/. See also the scheme in Appendix C SSEK-specific schemes. The content is as follows:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResponseCode</td>
<td>Response code (OK)</td>
</tr>
<tr>
<td>ResponseMessage</td>
<td>Response message, e.g. “The message has been received and will be processed”</td>
</tr>
<tr>
<td>RequestSignatureValue</td>
<td>The signature of the acknowledged message.</td>
</tr>
</tbody>
</table>

An example of an unsigned receipt (for a request) is given below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xh="http://schemas.ssek.org/txheader/2003-04-03/"
    xmlns:xh:SenderId="CN">Company B</xh:SenderId>
    xmlns:xh:ReceiverId="CN">Company A</xh:ReceiverId>
    xmlns:xh:TxId="61B0807-6F5F-46a1-92B4-6E5FA574E46E"
    xmlns:xh:Timestamp="2003-03-27T12:50:01">
  <xh:TxHeader/>
  <soap:Body>
    <r:Receipt xmlns:r="http://schemas.ssek.org/receipt/2003-04-03/">
      <r:ResponseCode>OK</r:ResponseCode>
      <r:ResponseMessage>The message has been received and will be processed</r:ResponseMessage>
      <r:RequestSignatureValue>X1os6m3YReQJlk1JBzwvLe5hScdz09uBx4EhnJpIKZVl2uScMpjd4Fpnyy/IPB3vk159aPcbQlzqTegaHBwi1pctwRNnnO34tYeY3/HekPSiBFb9pOFqh7308qqK0v1/2K21Y0BT/WBFNdYD6n6gP8nSjAKUPx1QFC8RCUKVeZk=</r:RequestSignatureValue>
    </r:Receipt>
  </soap:Body>
</soap:Envelope>
```
An example of a signed receipt (for a request) is given below:

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:tx="http://schemas.ssek.org/txheader/2003-04-03"
               wsu:Id="txHeader">
  <tx:TxHeader soap:mustUnderstand="1" xmlns:tx="http://schemas.ssek.org/txheader/2003-04-03"
               wsu="http://schemas.xmlsoap.org/ws/2002/07/utility"
               wsu:Id="txHeader">
    <tx:SenderId tx:typen="CN">Company B</tx:SenderId>
    <tx:SenderId tx:typen="CN">Company A</tx:SenderId>
    <tx:KeyId>C61B0B07-EF5F-46a1-92B4-6E5FA574E46E</tx:KeyId>
    <tx:Timestamp>2003-03-27T12:50:01</tx:Timestamp>
  </tx:TxHeader>
  <wss:Security soap:mustUnderstand="1"
           wsu:Id="SecurityToken-aca9eb37-8ba1-4ca4-8a98-50df7b66b87f"
           MI1CcjCCAAdugAwIBAgIBAAzANBgkqhkiG9w0BA...>
    <SignatureInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
      <CanonicalizationMethod>
        <SignatureMethod>
        <Reference URI="#txHeader"/>
        <Transforms>
          <Transform Algorithm="#xml-exc-c14n#"/>
        </Transforms>
        <Transform Algorithm="#xml-exc-c14n#"/>
        </SignatureMethod>
      </Reference>
      <Reference URI="#soapBody"/>
      <Transforms>
        <Transform Algorithm="#xml-exc-c14n#"/>
      </Transforms>
      <SignatureValue>49lyLcHwMTWkmyP0tEvjaAAuN0w</SignatureValue>
    </SignatureInfo>
    <SignedInfo>
      <SignatureValue>bCtbhvWanrAVpqTD9MCp1zK6AID7iDi6r18STI+4Mco...</SignatureValue>
      <KeyInfo>
        <wss:SecurityTokenReference>
          <wss:Reference URI="#SecurityToken-aca9eb37-8ba1-4ca4-8a98-50df7b66b87f"/>
        </wss:SecurityTokenReference>
      </KeyInfo>
    </SignedInfo>
      <Receipt xmlns="http://schemas.ssek.org/receipt/2003-04-03"/>
      <ResponseCode>OK</ResponseCode>
      <ResponseMessage>The message has been received and will be processed</ResponseMessage>
      <RequestSignatureValue>X1os6m3YReQJlk1BzvLe5hScdz09uBx4JpKZ...</RequestSignatureValue>
    </soap:Body>
  </wss:Security>
</soap:Envelope>
```
3.6 **Soap fault**

In the event of an error occurring which can be notified synchronously for a query message, the receiver MUST return a soap fault message with a fault code in accordance with *Appendix D Fault codes*. The Soap fault message CAN be signed.

An example of an unsigned soap fault (for a request) is given below:

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    <soap:Body>
        <soap:Fault>
            <soap:faultcode>Client.InvalidXml</soap:faultcode>
            <soap:Faultstring>Non valid XML</soap:Faultstring>
            <soap:detail>Element 'PNR' more than 12 digits</soap:detail>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```
3.7 Example of a signed query message

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
      <wsse:BinarySecurityToken>
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
          <Reference URI="#soapHeader"/>
          <Transforms>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <Transform Algorithm="http://www.w3.org/2001/09/xmldsig-more#IdEntegrity"/>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2001/09/xmldsig#sha1"/>
          <DigestValue>Yb17z5Fk8W7cJANJzlqfEzO</DigestValue>
          <Reference/>
        </Signature>
        <KeyInfo>
          <wsse:SecurityTokenReference URI="#SecurityToken-eda24e37-4bb-4f9b-8e4b-ac0d40512623"/>
        </KeyInfo>
        <Signature>
          <wsse:Signature>
            <Reference URI="#soapHeader"/>
          </Reference>
        </Signature>
      </wsse:BinarySecurityToken>
    </wsse:BinarySecurityToken>
  </wsse:Security>
</soap:Envelope>
</m:Envelope>
```

**Arrow A** points from the hash value of the header to the part that the hash value represents.

**Arrow B** points from the hash value of the document body to the part that the hash value represents.

**Box C** contains the certificate that was used to create the signature.

**Box D** contains the signature of SignedInfo.

The actual signature is therefore contained in SignatureValue, which signs the information in SignedInfo, including the hash values which represent information in the document header and body.
### 3.8 Example of an unsigned query message

An example is given below of an unsigned query message (for a request):

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Header>
  <txh:TxHeader soap:mustUnderstand="1" xmlns:txh="http://schemas.ssek.org/txheader/2003-04-03/">
    <txh:SenderId txh:type="CN">Company A</txh:SenderId>
    <txh:ReceiverId txh:type="CN">Company B</txh:ReceiverId>
    <txh:TxId>C61B0B07-6E5F-46a1-92B4-6E5FA574E46E</txh:TxId>
    <txh:Timestamp>2003-03-27T12:50:00</txh:Timestamp>
  </txh:TxHeader>
</soap:Header><soap:Body>
  <m:Nyanm xmlns:m="http://schemas.foretagb.se/nyanm/2003-04-03/">
    <m:Part>
      <m:AGREENR>A012112233</m:AGREENR>
      <m:PERSONS>
        <m:PERSON>
          <m:PNR>1212121212</m:PNR>
          <m:FIRSTNAME>Kajsa</m:FIRSTNAME>
          <m:SURNAME>Anka</m:SURNAME>
          <m:ADDRESS>Ankeborgsvägen 2</m:ADDRESS>
          <m:POSTCODE>11111</m:POSTCODE>
          <m:TOWN>Ankeborg</m:TOWN>
          <m:COUNTRY>Disneyland</m:COUNTRY>
        </m:PERSON>
      </m:PERSONS>
    </m:Part>
  </m:Nyanm>
</soap:Body></soap:Envelope>
```
4 Security

As the information which is being communicated is sensitive and valuable, there is a need to prevent unauthorized persons from reading or manipulating the information. There may also be a need to trace information to the sender.

In order to fulfil the above requirements, SSEK describes how the following security aspects are handled in connection with electronic communication.

- **Authentication**
  Identification of the sending organisation (client) and the receiving organisation (server).

- **Confidentiality**
  Information that is sent cannot be read by external parties.

- **Correctness and Integrity**
  Information that is sent cannot be altered after it has left the sender and follows the agreed format.

- **Non-repudiation**
  The party that created and sent the information cannot deny that the information in question was created.

PKI (Public Key Infrastructure) is used in connection with the handling of the above security aspects.
4.1 Security levels

In order to fulfil all, or just a few, of the security aspects described above, signatures and client certificates can be combined. The permissible combinations are defined in four different security levels where level 4 is the highest security and level 1 is the weakest security.

Common to all SSEK security levels is the fact that an SSL tunnel is set up, through which confidentiality is maintained during the transmission and that the server authenticates itself with a server certificate.

Further information concerning certificates and signatures can be found in section 4.3 Signing and 4.4 CA and certificates.

<table>
<thead>
<tr>
<th>SSEK level</th>
<th>SSL – with server certificate</th>
<th>Signature</th>
<th>Client certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Used</td>
<td>Used</td>
<td>Used</td>
</tr>
<tr>
<td>3</td>
<td>Used</td>
<td>Used</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Used</td>
<td></td>
<td>Used</td>
</tr>
<tr>
<td>1</td>
<td>Used</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Information on the SSEK security levels:

1. Gives confidentiality in connection with transmission and authentication of the server. Does not give authentication of the client or non-repudiation because signatures and client certificates are not used. Appropriate level for public services.
2. Gives confidentiality in connection with transmission and authentication of both the server and the client. Does not give non-repudiation because signatures are not used. Appropriate level for services that are aimed at certain competent authorities but where the information sent does not subsequently have to be used as evidence.
3. Gives confidentiality in connection with transmission, authentication of the server and non-repudiation. Does not give authentication of the client because client certificates are not used. Appropriate level for services where the sender does not need to be authenticated but where the receiver has a need to demonstrate which organisation created the information that is received.
4. Gives confidentiality in connection with transmission, authentication of both the server and the client and non-repudiation. Both signatures and client certificates are used. Appropriate level for sensitive services that are aimed at certain competent authorities and where the receiver has a need to demonstrate which organisation created the information that is received.

4.2 Secure communication

Both parties can initiate the communication. A party can do this by connecting up what is known as a secure tunnel between itself and the other party. The secure tunnel MUST be created with SSL, for which the protocol HTTPS (HTTP via SSL) is used. In this case, both parties are authenticated, i.e. both client and server authentication using their certificates.

The tunnel MUST be encrypted with a secure key.
At least a 128-bit key must be used.
4.3 Signing
A digital signature gives technical proof of the organisation that created certain information. As messages are signed, the directives that are described in this document concerning signatures MUST be followed.

Certificates MUST be used to create signatures.

4.4 CAs and certificates
The certificates that are required to create the signature and authenticate each party MUST be issued by a CA (Certificate Authority) approved by both parties and be approved with regard to type and procedures for the issuing of certificates. The certificates MUST follow the standard X.509 v3 and have a sufficiently secure public key. At least a 1024-bit key applies to this.

The use of digital signatures is based on the trust that the communicating parties have in the CA that issued the certificate.

4.5 Revoking of certificates
If the certificate can no longer be considered as trustworthy, the parties that are using the certificate MUST be informed of this.

The party that uses a certificate for verifying an organisation’s identity and signatures SHOULD collect the CRL (Certificate Revocation List) issued by the certificate’s CA and check that the certificate has not been revoked each time the certificate is used. The revoking of the certificate by the certificate’s CA MUST be carried out by the owner of the certificate. In connection with this revoking, the certificate is placed in the CRL, which is regularly issued by the certificate’s CA, e.g. every 2 hours.

If there are no procedures or techniques in place for automatic revoking checks to retrieve and read a CRL, “manual revoking” MUST be used. This is carried out by the party that wants to make a certificate invalid contacting the person responsible on behalf of the other party as soon as possible, preferably by telephone. This person will then remove the certificate so that it cannot be used. In this case, the certificate MUST also be revoked by the certificate’s CA. This will therefore also work for those who actually use the CRL that is issued.

If “manual revoking” is used between two communicating parties, procedures for the two parties MUST be specified. Both parties must agree on the specified procedures.

If no procedure for revoking has been established, there is a greater risk of a fraudulent party, which has stolen the certificate's private key, using the key for a long period of time before notification that the certificate has been stolen and cannot be considered as a secure identification of the other party.
4.6 **Approved CAs and certificate types**

Telia eCommerce (issues certificates with Verisign as the CA) and Posten eSäkerhet are currently approved to issue certificates by parties who use this specification for their electronic communication.

Other types of certificate that are issued by approved CAs and used by the parties MUST be approved for use by both parties. In this case, the requirement for the certificate type is that a sufficient validation of the organisation’s identity is carried out in connection with the issuing of the certificate.

> If an insecure CA or certificate is used without adequate validation of the organisation’s identity, there is a greater risk of a fraudulent organisation managing to pass itself off as the other party in connection with communication. This could have important consequences.

4.7 **Handling private keys**

Authentication and signing is carried out with a private key, which is linked to a certificate. The authentication or signature is then used to prove that certain information has been created and sent by the sending party. The private key MUST therefore be protected in a sufficiently secure manner so that no unauthorised parties can steal the key.

If the private keys are not adequately protected, there is a greater risk of them being stolen.

A fraudulent party that has stolen a key can claim to be the organisation to which the key belongs and act in its place. The consequences can therefore be considerable.
5 Appendix A Standards

[IETF]  
IETF (Internet Engineering Task Force), http://www.ietf.org/

[OASIS]  
OASIS (Organisation for the Advancement of Structured Information Standards), http://www.oasis-open.org/

[PKI certificate]  

[Posten]  
Posten http://digitalid.postnet.se/

[SHA-1]  
http://www.itl.nist.gov/fipspubs/fip180-1.htm

[SOAP]  

[SSL]  

[WS-Security]  

[W3C]  
W3C (World Wide Web Consortium), http://www.w3.org/

[X.509 v3]  

[XML]  

[XML-C14N]  

[XML-Schema1]  

[XML-Schema2]  

[XML Signature]  
http://www.w3.org/Signature/
6 Appendix B Web Services Security

The following parts of the specification for **WS-Security** are used within SSEK.

<table>
<thead>
<tr>
<th>Section</th>
<th>Clarification</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2 Encoding Binary Security Tokens</td>
<td>Only X.509 certificates are used. X.509 certificate specified by using BinarySecurityToken:</td>
</tr>
<tr>
<td></td>
<td>&lt;wsse:BinarySecurityToken</td>
</tr>
<tr>
<td></td>
<td>Id=&quot;myToken&quot;</td>
</tr>
<tr>
<td></td>
<td>ValueType=&quot;wsse:X509v3&quot;</td>
</tr>
<tr>
<td></td>
<td>EncodingType=&quot;wsse:Base64Binary&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>MIIEZzCCA9CgAwIBAgIQLzJc0...</td>
</tr>
<tr>
<td></td>
<td>&lt;/wsse:BinarySecurityToken&gt;</td>
</tr>
</tbody>
</table>

4.3 SecurityTokenReference Element

4.5 ds:Signature

The following parts of the specification for **Web Services Security Addendum** are used within SSEK.

<table>
<thead>
<tr>
<th>Section</th>
<th>Clarification</th>
</tr>
</thead>
</table>

3.2 Id Scheme                     | X.509 certificate specified by using BinarySecurityToken                                           |

4. Placement of X.509 Certificates |                                                                                                  |
7 Appendix C SSEK-specific schemes

Scheme for TxHeader:

```xml
<xsd:schema targetNamespace="http://schemas.ssek.org/txheader/2003-04-03/"
xmlls:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified">
  <xsd:element name="TxHeader">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SenderId">
          <xsd:simpleContent>
            <xsd:complexType>
              <xsd:restriction base="xsd:anyType">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:restriction>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="ReceiverId">
          <xsd:simpleContent>
            <xsd:complexType>
              <xsd:restriction base="xsd:anyType">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:maxLength value="256"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:restriction>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="TxId" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:length value="36"/>
            </xsd:simpleType>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
Scheme for receipt:

```xml
<xsd:schema targetNamespace="http://schemas.ssek.org/receipt/2003-04-03/"
    xmlns:tns="http://schemas.ssek.org/receipt/2003-04-03/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

  <xsd:element name="Receipt">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ResponseCode" type="xsd:string" />
        <xsd:element name="ResponseMessage" type="xsd:string" minOccurs="0" />
        <xsd:element name="RequestSignatureValue" type="xsd:string" minOccurs="0" />
        <xsd:any minOccurs="0" maxOccurs="unbounded" namespace="#any"
                processContents="lax" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
8 Appendix D Fault codes

Definitions of the fault codes that can be inserted in fault messages. The values that should be set in Faultstring and Detail are not defined here. These can be set to arbitrary values that are agreed between the communicating parties. A scheme for soap fault is shown below:
http://schemas.xml.org/soap/envelope/

8.1 Faults concerning the document

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VersionMismatch</td>
<td>An unknown version of soap was specified</td>
</tr>
<tr>
<td>MustUnderstand.TxHeader</td>
<td>Incorrect format for TxHeader</td>
</tr>
<tr>
<td>MustUnderstand.Security</td>
<td>Incorrect format for security header</td>
</tr>
<tr>
<td>Client.InvalidXml</td>
<td>Incorrect format for message</td>
</tr>
<tr>
<td>Client. WebService.Unknown</td>
<td>The combination of TxHeader, MessageName and nsuri for the message is incorrect.</td>
</tr>
<tr>
<td>Client.TxHeader.SenderId.Unknown</td>
<td></td>
</tr>
<tr>
<td>Client.TxHeader.ReceiverId.Unknown</td>
<td></td>
</tr>
<tr>
<td>Client.TxHeader.Timestamp.Invalid</td>
<td></td>
</tr>
</tbody>
</table>

8.2 Faults concerning the transaction ID

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client.TxHeader.TxId.Missing</td>
<td>TxId has not been specified</td>
</tr>
<tr>
<td>Client.TxHeader.TxId.Unknown</td>
<td></td>
</tr>
<tr>
<td>Client.TxHeader.TxId.Invalid</td>
<td>Incorrect format for TxId</td>
</tr>
<tr>
<td>Client.TxHeader.TxId.Duplicate</td>
<td>TxId is not unique</td>
</tr>
<tr>
<td>Client.TxHeader.TxId.NotAllowed</td>
<td>TxId must not be specified for this service</td>
</tr>
</tbody>
</table>

8.3 Faults on the server side

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server.MessageNotProcessed</td>
<td>Internal fault</td>
</tr>
<tr>
<td>Server.WebService.Unavailable</td>
<td>The service is closed</td>
</tr>
<tr>
<td>Server.WebService.Unsupported</td>
<td>The service is not supported by the receiver</td>
</tr>
</tbody>
</table>

8.4 Logical faults in transaction data

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client.Receivername.arbitrary faultcode</td>
<td>Faults in business data concerning business logic, e.g. salary too high, can be notified in this way, e.g. “Client.Skandia. InvalidSalary”</td>
</tr>
</tbody>
</table>
### 8.5 Faults concerning certificates or signatures

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client.UnregisteredClientCertificate</td>
<td>DN in http header in client certificate does not agree with the saved DN.</td>
</tr>
<tr>
<td>Client.InvalidSecurityToken</td>
<td>The certificate is invalid with regard to the validity period.</td>
</tr>
<tr>
<td>Client.InvalidSecurityToken</td>
<td>The certificate has been revoked.</td>
</tr>
<tr>
<td>Client.FailedAuthentication</td>
<td>The specified security token could not be authenticated or authorised.</td>
</tr>
<tr>
<td>Client.UnsupportedSecurityToken</td>
<td>An unknown security token was specified, not X509.</td>
</tr>
<tr>
<td>Client.UnregisteredSecurityToken</td>
<td>DN in signaturcert does not agree with the saved DN.</td>
</tr>
<tr>
<td>Client.UnsupportedAlgorithm</td>
<td>An unknown algorithm for signature or encryption was specified</td>
</tr>
<tr>
<td>Client.InvalidSecurity</td>
<td>A fault was discovered in connection with the processing of the &lt;Security&gt; header</td>
</tr>
<tr>
<td>Client.InvalidSecurityToken</td>
<td></td>
</tr>
<tr>
<td>Client.FailedCheck</td>
<td></td>
</tr>
<tr>
<td>Client.SecurityTokenUnavailable</td>
<td></td>
</tr>
</tbody>
</table>

### 8.6 Other faults which are not reported via SOAP fault

<table>
<thead>
<tr>
<th>Fault code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>Server unavailable</td>
</tr>
<tr>
<td>TimeOut</td>
<td>The client cannot receive a response</td>
</tr>
</tbody>
</table>