Danske Bank

MobilePay AppSwitch API Implementation Guide

Version 1.1

January 2016

Contents

1 Document log 3

2 Purpose of this guide 4

2.1 Target groups 4

2.2 Technical support 4

3 MobilePay AppSwitch API 5

3.1 Backend API services 5

4 PKI integration 7

4.1 Assumptions 7

4.2 Background information 7

4.3 Getting started 7

4.4 Obtain the root certificate 8

4.5 Install needed utilities 8

4.6 Generate certificates and private pairs 8

4.7 Create PKCS#12 files 9

4.8 Compile the supplied library 9

4.9 Utilise the sample application provided 9

5 SOAP library integration 12

5.1 How the integration works from a .NET client 12

5.2 Invoking a SOAP web service 13

5.3 Obtain necessary WSDL files 13

5.4 Obtain the DB.SoapLibrary SDK 13

5.5 Generate proxy files 13

5.6 Add proxy files to DB.SoapLibrary 13

5.7 Correct the proxy files 14

5.8 Obtain security certificates 14

5.9 Invoke the services 14

5.10 Sending requests 15

5.11 Request Example 16

5.12 Signing the request 17

5.13 Encrypting the initially signed request 18

5.14 Signing the encrypted request 19

6 Key terms and definitions 21

1 Document log

Version	Date	Amendment
1.0	15.12.2015	TKI: Document created
1.1	06.01.2016	TKI/DUY: Corrections to text and layout

2 Purpose of this guide

This implementation guide explains the implementation design of MobilePay AppSwitch API services. This includes descriptions of backend services, security, error scenarios, etc.

2.1 Target groups

The target group is project managers, system architects and developers.

2.2 Technical support

For technical questions, please contact Danske Bank Support on +45 70 114 115 or kundesupport@danskebank.dk.

Please prepare the scheme below when requesting for support.

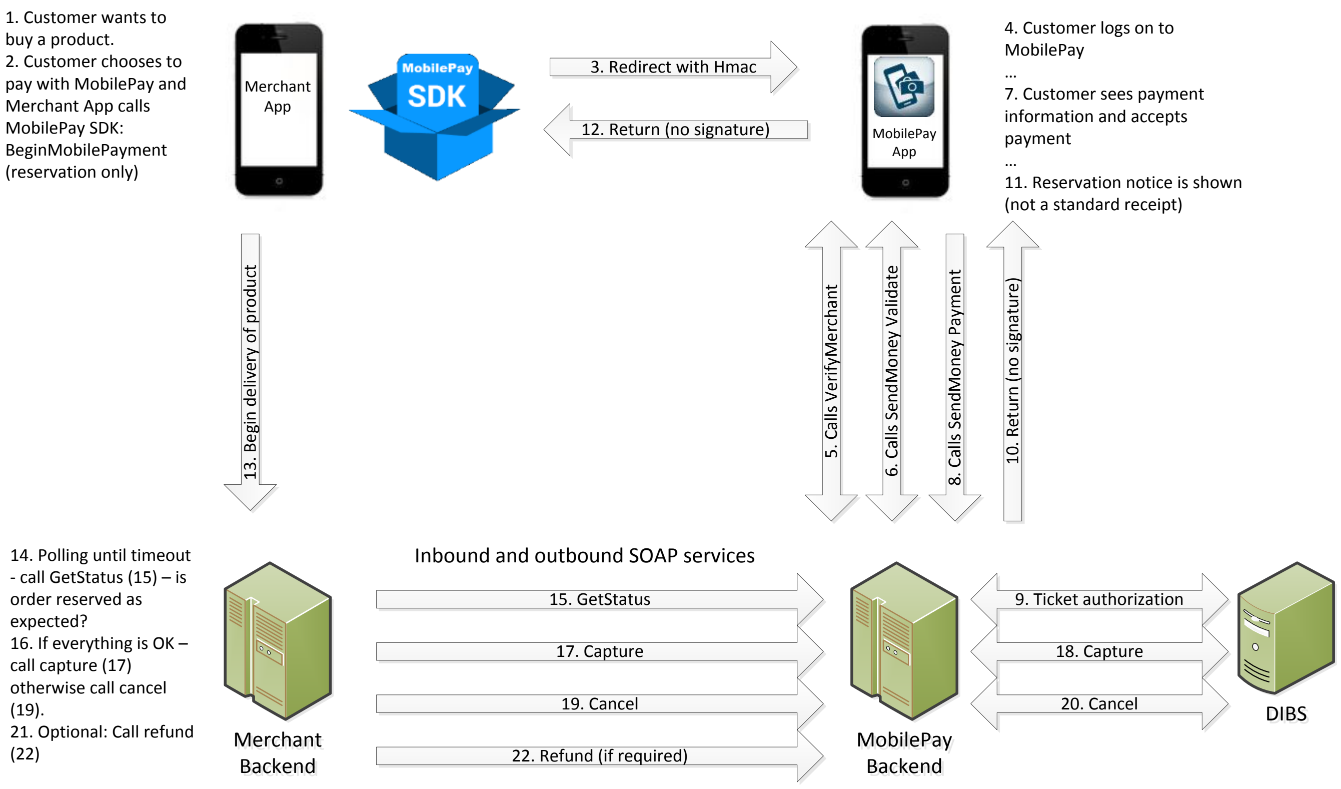
Subject	**Description / comments**
Error	*[Headline/title]*
Description	*[Error message with a short description]*
Service	*[Which service is affected?]*
Screenshot	*[Screen shot if possible]*
Date/time	*[Timestamp]*
Mobile number	*[Registered mobile number of user]*
E-mail address	*[Registered email address of merchant]*
Platform	*e.g. iOS*
OS version	*e.g. iOS 8.1*
Merchant app version	*e.g. Version 3.60*
OrderID	*e.g. 2015-06-08 000001*
TransactionID	*e.g. 1089237509*

3 MobilePay AppSwitch API

The MobilePay AppSwitch API solution enables lifecycle management of transactions; from the transaction is initiated in the merchant app to delivery of the product and finalisation of the transaction in the merchant backend.

The integrity and identity is guarded by an HMAC authentication and a signature validation, respectively. The HMAC is calculated in the SDK and sent to the MobilePay backend for verification (step 5 in below figure). This is done for all calls in MobilePay. The signature validation is only required when CaptureType = "Y".

As no signature is returned to merchant app from MobilePay when a reservation has been made, the identification verification of the current transaction must take place in merchant backend via a GetStatus API call (thus confirming Danske Bank as sender and verifying the data).


1. Customer wants to buy a product.
2. Customer chooses to pay with MobilePay and Merchant App calls MobilePay SDK: BeginMobilePayment (reservation only)
Merchant App
MobilePay SDK
3. Redirect with Hmac
12. Return (no signature)
MobilePay App
4. Customer logs on to MobilePay
...
7. Customer sees payment information and accepts payment
...
11. Reservation notice is shown (not a standard receipt)
13. Begin delivery of product
5. Calls VerifyMerchant
6. Calls SendMoney Validate
8. Calls SendMoney Payment
10. Return (no signature)
14. Polling until timeout - call GetStatus (15) – is order reserved as expected?
16. If everything is OK – call capture (17) otherwise call cancel (19).
21. Optional: Call refund (22)
Merchant Backend
Inbound and outbound SOAP services
15. GetStatus
17. Capture
19. Cancel
22. Refund (if required)
MobilePay Backend
9. Ticket authorization
18. Capture
20. Cancel
DIBS

3.1 Backend API services

Five backend API services are currently offered for the merchant backend.

1. **CancelV02** service which cancels any given reserved payment transaction.
2. **CaptureV02** service which captures any given reserved payment transaction.
3. **GetReservationsV01** service which lists all reserved payments for a given merchant ID.

4. **GetStatusV03** service which returns a status on any given payment to check whether it is authorised or not.
5. **RefundV03** service which refunds any given payment transaction.

Detailed information for each services can be found on GitHub under the MobilePay-SoapServices-SDK[1] repository.

[1] An invitation by Danske Bank is necessary for accessing the MobilePay-SoapServices-SDK repository on GitHub.

4 PKI integration

The secutity model surrounding the Danske Bank backend is proprietary and the central point is the so called PKI Factory. The PKI Factory is the service interface to create and renew certificates, and used to call the “business” endpoints. Danske Bank has developed a PKI Client that encapsulates all communication with the PKI Factory. The code, the test app, and the documentation can be found on GitHub. This standard implementation makes it easier for merchants to do proper automated certificate administration.

The client library contains a PKI Client implementation and a test application that clearly demonstrates how to interact with Danske Banks PKI Factory.

The test application can be used directly in the initial requisition of the certificates, and the PKI Client lays the ground for an easy implementation of an automated certificate renewal job.

It is highly recommended to implement an automated renewal job, since expired certificates means no access to Danske Bank’s backend APIs.

4.1 Assumptions

The setup is founded upon the following assumptions:

1. A Business Online agreement with Danske bank is obtained
2. Customer number from the agreement
3. PIN code from the agreement

4.2 Background information

The client certificates will always exist in pairs: One is used for signing and the other for encryption. Therefore, they will also be issued, revoked and renewed simultaneously.

The private key corresponding to the certificates must never be revealed by the client to anyone – including Danske Bank.

4.3 Getting started

The process can be broken down into the following steps:

1. Obtain Danske Bank’s root certificate.
2. Install needed utilities.
3. Generate two sets of certificates along with their private keys (Step 2 and 3 can be bypassed if you already have certificates and private keys).
4. Create two PKCS#12 files which contain the certificate and private keys generated above.
5. Compile the library supplied.
6. Utilise the sample application to obtain certificates issued by the bank.

4.4 Obtain the root certificate

The first thing needed to communicate with the backend is the initial bank root certificate. This will be used for encrypting the initial requests, and validating the responses.

The certificate can be obtained from the Danske Bank's website[2] . It is a self-signed certificate distributed in a container which is signed by a bank certificate issued by a major certificate authority (CA).

You must verify the signature of the zip file before proceeding. This verification process is described online[3].

4.5 Install needed utilities

To complete the following two steps you will need to have Microsoft WDK installed and its utilities present in your path.

Before you begin the installation you can test whether you have the tools already by navigating to "C:\Program Files (x86)\Windows Kits\<version>\bin\" and testing if "makecert.exe" and "pvk2pfx.exe" are present. If they are, the rest of this step may be skipped.

You can obtain WDK from the following URL:

https://msdn.microsoft.com/en-us/library/windows/hardware/ff557573(v=vs.85).aspx

After installation you should add the bin folder to your path.

If you start a command prompt, and keep it open for the next steps, your path can be temporarily modified by using the following command:

```
set PATH=%PATH%;C:\Program Files (x86)\Windows
Kits\8.1\bin\x64
```

The exact path may vary depending on the specific version of your operating system.

4.6 Generate certificates and private pairs

Two key pairs must be generated: One for signing and one for encryption.

On Windows, such a pair can be generated with the "makecert.exe" utility distributed by Microsoft as part of the Microsoft WDK.

Issuing the following commands in the command prompt started earlier will create two certificates and private keys:

```
makecert -r -pe -n "CN=DemoSigning" -b 01/01/2015 -e
01/01/2017 -len 2048 -sky exchange  -sv Signing.pvk Signing.crt
```

[2] https://www.danskebank.com/en-uk/ci/Products-Services/Transaction-Services/Online-Services/Documents/DBGROOT_1111110002.zip

[3] https://www.danskebank.com/en-uk/ci/products-services/transaction-services/online-services/pages/pki-services.aspx

```
makecert -r -pe -n "CN=DemoEncryption" -b 01/01/2015 -e
01/01/2017 -len 2048 -sky exchange  -sv Encryption.pvk
Encryption.crt
```

The parameters which can be modified are:

Parameter	Description
-n	Name of the publishers certificate (x.500 format)
-b	Earliest date the certificate is valid (mm/dd/yyyy format)
-e	Last date the certificate is valid (mm/dd/yyyy format)
-sv	Private key filename
-sv	Certificate filename

You will be prompted for a password which is used to protect the private key. **Please note that you will need this password again later**.

This will generate a certificate with a validity period and a 2048 bit key. The key size must be 2048 bits in order to be accepted by Danske Bank. Furthermore, the validity period must be two years.

4.7 Create PKCS#12 files

The certificate and private key files can be combined into a PKCS#12 container with the "pvk2pfx.exe" tool distributed by Microsoft as part of the Microsoft WDK:

```
pvk2pfx -pvk Signing.pvk -spc Signing.crt -pfx Signing.pfx
pvk2pfx -pvk Encryption.pvk -spc Encryption.crt -pfx Encryption.pfx
```

The initial certificate and private key can now be discarded:

```
del Signing.pvk
del Encryption.pvk
del Signing.crt
del Encryption.crt
```

The remaining .pfx files must be protected since they can later authenticate requests for the SOAP services backend.

4.8 Compile the supplied library

Compile the library by running the "make - copy to test app.bat" file from the DanskeBank.PKIFactory.Library folder.

This will build the library and copy it to the expected path so that the sample application can access it.

4.9 Utilise the sample application provided

The library interface itself is fully documented, and thus the sample application simply demonstrates how to use it after you have obtained the elements listed above.

The sample application utilises the Windows Certificate Store. Thus the bank root certificate obtained as well as the two locally generated PKCS#12 containers must be added to this store. This is accomplished in the following steps:

1. Creating a sub-store using the makecert utility from earlier by issuing the command: makecert –ss DanskeBank.PKIFactory
2. Start the certificate management tool by running "certmgr.msc"
3. Navigate to the newly created store
4. Remove the default generated certificate ("Joe's Software Emperium")
5. Import the bank root certificate:
    a. Right-click the "DanskeBank.PKIFactory" node -> All tasks -> Import -> Next
    b. Navigate to the root certificate file
    c. Click "Next" and then "Finish"
6. Import the PKCS#12 containers:
    a. Right-click the "DanskeBank.PKIFactory" node -> All tasks -> Import -> Next
    b. Select "Personal Information Exchange (*.pfx, *.p12) as file type
    c. Navigate to the folder containing the container
    d. Select one of the PKCS#12 containers
    e. Under "Import options" select:
        i. "Mark this key as exportable"
        ii. "Include all extended properties"
    f. Finalise the import
7. Repeat step 6 for the other PKCS#12 container
8. Set the friendly name of each imported item:
    a. Right-click the imported file in the certificate management tool -> "Properties"
    b. Enter the friendly name in the "Friendly name" field (see directly below for expected values)

Item	**Friendly name**
Bank root certificate	BankRoot
Own encryption PKCS#12	ClientGeneratedEncryption
Own signing PKCS#12	ClientGeneratedSigning

From this point, the constant Customer_Id and Customer_PIN fields in the Main method of "Program.cs" must be replaced with their proper values provided by the Bank.

Finally, the application code can be executed to request that the Bank issues certificates for use with the SOAP services gateway. If the application is executed without any further modifications, requests will be sent to Danske Bank's PKI factory in order to obtain properly signed certificates.

The certificates will be placed in the same key store used above with the following friendly names set:

Item	**Friendly name**
Bank root certificate. Application requested the latest version	BankRoot
Bank encryption certificate. Obtained from the PKI factory.	BankEncryption

Used to encrypt requests sent to the Bank.	
Bank signing certificate. Obtained from the PKI factory. Used to validate signature on responses from the Bank.	BankSigning
Client-generated encryption certificate and private key. Unchanged since initial import.	ClientGeneratedEncryption
Client-generated signing certificate and private key. Unchanged since initial import.	ClientGeneratedSigning
Client encryption certificate. This certificate is signed by the PKI factory. The bank may encrypt data using this certificate when sending it to the client.	ClientIssuedEncryption
Client-signing certificate. This certificate is signed by the PKI factory. The private key of this certificate is used to sign requests sent to the Bank.	ClientIssuedSigning

5 SOAP library integration

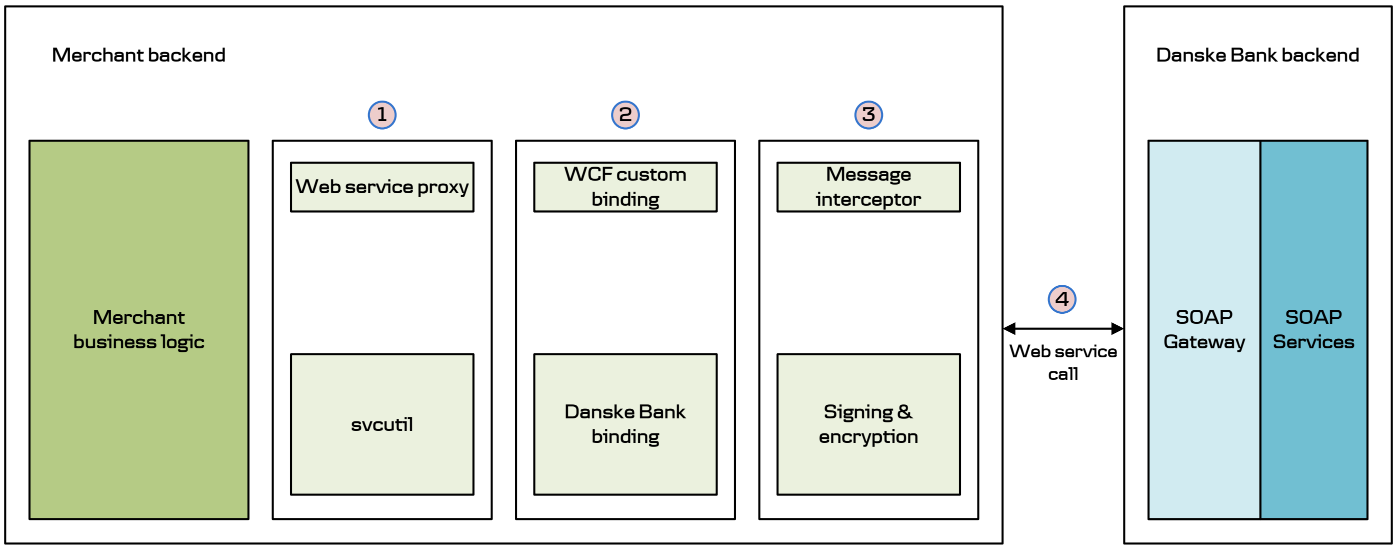
To integrate applications with services on the Danske Bank IT platform, one of the available approaches is to use SOAP web services. To access these services, a SOAP gateway is used to ensure integrity and security of the interactions.

5.1 How the integration works from a .NET client

For the communication to succeed, messages must adhere to Danske Bank-specific certificates, encryption, and signatures.

To establish communication from a .NET client to a Danske Bank SOAP web service, there are four key elements that must be addressed:

1. **Web service proxy**, which defines valid proxies and model stubs for the web service to comply with Danske Bank standards.
2. **WCF custom binding**, which handles encoding and HTTP transport of SOAP messages.
3. **Message interceptor,** which provides the Danske Bank implementation of WS-Security with respect to signature and encryption.
4. **Web service call,** which is the actual transmission of the SOAP request and response.



Web service proxy:

The necessary proxies needed to establish connection to Danske Bank are generated from WSDL using svcutil.

WCF custom binding:

The WCF binding is customised to support Danske Bank security standards and allow mes-sages to be acknowledged by the SOAP Gateway.

Since WCF does not support the Danske Bank implementation of WS-Security out-of-the-box, the custom binding is used only to encode and HTTP transport SOAP messages. Encryption and signatures are handled by the message interceptor.

Message interceptor:

The message interceptor is used to provide the Danske Bank realisation of the signing and encryption required by WS-Security.

Web service call:

The actual transmission of the SOAP message with certificates to a given HTTP end point address.

5.2 Invoking a SOAP web service

There are a number of steps you must go through to invoke a SOAP web service the first time:

1. Obtain necessary WSDL files
2. Obtain the DB.SoapLibrary SDK
3. Generate proxy files
4. Add proxy files to DB.SoapLibrary
5. Correct the proxy files
6. Obtain security certificates
7. Invoke web services

Each of these steps is detailed further below.

5.3 Obtain necessary WSDL files

The WSDL files necessary for the integration should have been provided together with web service access. Contact the Danske Bank representative if this is not the case.

5.4 Obtain the DB.SoapLibrary SDK

The SDK is manually distributed together with the WSDL files.

5.5 Generate proxy files

The proxy files are generated from the WSDL files using svcutil from the command line:

```
svcutil <file>.wsdl oasis-200401-wss-wssecurity-secext-
1.0.xsd/language:C#
```

The schema is delivered with the SDK, but can also be downloaded from OASIS.

5.6 Add proxy files to DB.SoapLibrary

The generated proxy files should be added to the DB.SoapLibrary solution.

5.7 Correct the proxy files

There are a few things that must be corrected in the generated proxy files. Specifically, this is to ensure that headers are not encrypted together with the encryption of the bodies, which is done by setting the message header attribute protection level to none.

```
// Protection level set to None to ensure that the header is not encrypted
// together with the encryption of the body

public partial class <TestService>
{
        [System.ServiceModel.MessageHeaderAttribute(
              Namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
              secext-1.0.xsd",
              ProtectionLevel = ProtectionLevel.None)]
        public SecurityHeaderType Security;
        [System.ServiceModel.MessageHeaderAttribtue(
              Namespace="http://www.danskebank.com/SecureSOAPSGW",
              ProtectionLevel = Protectionlevel.None)]
        public RequestHeaderType RequestHeader;
}
```

If multiple WSDL files are in play, each generated proxy file should be given a specific name-space.

```
// Proxies are embedded in namespaces if multiple WSDL files are in play

namespace DB.SoapLibrary.Proxy.<servicename>
{
        [System.CodeDom.Compiler.GeneratedCodeAttribute("System.ServiceModel", "4.0.0.0")]
        [System.ServiceModel.ServiceContractAttribute(Namespace =
              "http://www.danskebank.com/services/", ConfigurationName = "EchoSoapService")]
        public interface EchoSoapService
        {
              // generated code omitted
        }
}
```

5.8 Obtain security certificates

Please refer to the Danske Bank web site for how to create the first security certificate:

1. Download the root certificate for Danske Bank
2. Follow the steps in the PKI service documentation to create your first security certificate

5.9 Invoke the services

At this point, the web services can be invoked as normal from client programs. Examples are provided together with relevant API documentation.

NOTE: For traceability purposes, the SDK will log input XML and output XML for the executed web services. These files are placed in the .NET solution space.

```
// Input tracing can be turned off by inserting highlighted commenting

public object BeforeSendRequest(ref Message request, System.ServiceModel.IClientChannel channel)
{
// ...
/*
        using (var stringWriter = new StreamWriter(new FileStream("Input.xml", FileMode.Create)))
        using (var xmlTextWriter = XmlWriter.Create(stringWriter))
        {
            xmlDoc.WriteTo(xmlTextWriter);
            xmlTextWriter.Flush();
        }
*/
}

// Output tracing can be turned off by inserting highlighted commenting

public void AfterReceiveReply(ref Message reply, object correlationState)
{
/*
        LastResponseXML = reply.ToString();
        XmlDocument xmlDoc = new XmlDocument();
        xmlDoc.LoadXml(LastResponseXML);

        using (var stringWriter = new StreamWriter(new FileStream( "Output.xml", FileMode.Create)))
        using (var xmlTextWriter = XmlWriter.Create(stringWriter))
        {
            xmlDoc.WriteTo(xmlTextWriter);
            xmlTextWriter.Flush();
        }
*/
}
```

5.10 Sending requests

In order to send a request, signatures and encryption using WS-Security must be applied first. The following settings should be used:

Signatures

- Key Identifier Type: Binary Security Token
- Signature Algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1
- Signature Canonicalization: http://www.w3.org/2001/10/xml-exc-c14n#
- Digest Algorithm: http://www.w3.org/2000/09/xmldsig#sha1

Encryption

- Key Identifier Type: Binary Security Token
- Symmetric Encoding Algorithm: http://www.w3.org/2001/04/xmlenc#tripledes-cbc
- Key Encryption Algorithm:

http://www.w3.org/2001/04/xmlenc#rsa-1_5

5.11 Request Example

This is an example of a request sent to:

privatemobilepayservicescert.danskebank.com

Explanation of the RequestHeader information

- SenderId: ID of the user that performs the signature over the encrypted part. This field is required.
- SignerId1: ID of one of the signers. If DBCryptId is not defined, the response will be encrypted under this user certificate. This field is required.
- SignerId2: ID of one of the signers. This field is optional.
- SignerId3: ID of one of the signers. This field is optional.
- DBCryptId: ID of the client that will be able to decrypt the response. This field is optional. If nothing is given as input, SignerId1 will be used.
- RequestId: ID of the request. This field is required.
- Timestamp: Date and time of the request. This field is required.
- Language: Customer language. This field is required.

SignerId2 and SignerId3 need only be used when more than one signature is applied to the body.

```
<soap:Envelope      xmlns:soap="http://www.w3.org/2003/05/soap-envelope"      xmlns:sec="http://www.danskebank.com/SecureSoapSGW"
xmlns:oas="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:ser="http://www.danskebank.com/services/">
<soap:Header>
   <sec:RequestHeader>
      <sec:SenderId>1F0682</sec:SenderId>
      <sec:SignerId1>1F0682</sec:SignerId1>
      <sec:SignerId2/>
      <sec:SignerId3/>
      <sec:DBCryptId/>
      <sec:RequestId>123</sec:RequestId>
      <sec:Timestamp>2014-03-31T14:04:50Z</sec:Timestamp>
      <sec:Language>DK</sec:Language>
   </sec:RequestHeader>
   <oas:Security/>
</soap:Header>
<soap:Body>
   <ser:Echo>
      <ser:Input>
         <ser:Message>Hello</ser:Message>
      </ser:Input>
   </ser:Echo>
</soap:Body>
</soap:Envelope>
```

5.12 Signing the request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:oas="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:sec="http://www.danskebank.com/SecureSoapSGW"
xmlns:ser="http://www.danskebank.com/services/">
<soap:Header>
<wsse:Security xmlns:wsse=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="IdX509-
325E11B992A8C7CD1114216530529O117">MIID4DCCAsigAwIBAgIHFfPEjBFBMTANBgkqhkiG9w0BAQsFADCBmjEQMA4GA1UEAxM
HREJHU1dESzELMAkGA1UEBhMCREsxEzARBgNVBAcTCkNvcGVuaGFnZW4xEDAOBgNVBAgTB0Rlbm1hcmsxGjAYBgNVBAoTEURhbnNrZ
SBCYW5rIEdyb3VwMRwwGgYDVQQLExNEYW5za2UgQmFuayBEZW5tYXJrMRgwFgYDVQQFEw82MTEyNjIyODc3MzAxMDEwHhcNMTQwO
DIyMDg1MzQ2WhcNMTYwODIyMDg1MzQ2WjByMRswGQYDVQQDExJKQU4gTFV0RC1IRU5SSUtTRU4xCzAJBgNVBAYTAkRLMRAwDgYD
VQQKEwdUU1QgTUFJMTQwMgYDVQQFEytTRS10Ui9EQUJBOjAwNjExMjYyMjgtQUdSOjZDNTUwOS1VU1I6MUYwNjgyMIIBIjANBgkqhkiG9w
0BAQEFAAOCAQ8AMIIBCgKCAQEApterTgvr1L/MSKzZLHJmKdnm8lnqVe0rb1p/kT9EijSaT9m+aUqghUg+egCQy0Ou2TUcH68OdSr8EeXs/
mw51Te3CNMIzrE6E/zCfgKIhvWnl+7a1psvfWrCIOjYHzFC9I983B16wP50Ai/ZJSf/5slmNEgs4C/V3RoutvIJua08UtJ7RSd/1YFTscXq7pN9
6ia4XiYVKSeuVl8E8OmRiENlG/qbYeliBAyCJa1yBQWFktQqlwwAwM9JqYCtRYhjQS+P+nv4R6G3v8D8xA2/F977rSqQKYT35N6Woh7MaO
egL5O4c2D9TnLMrv0zWaOJ7aZc1x6yEz5F9YcRLC7lFQIDAQABo1IwUDAfBgNVHSMEGDAWgBQ05nD7QkqIQCrxigdJncTYo20tIzAdBgNV
HQ4EFgQUC6OvsGEb2GN/rO2PXIOEo1XuMVMwDgYDVR0PAQH/BAQDAgbAMA0GCSqGSIb3DQEBCwUAA4IBAQAja6BRmlEWrRbO2beZ
D7eNplSCwkVFoBY4zhPa/eVPCFe3ggUoPo6Pl0s5xDYfcbX2mUdA/A8RsgElcGQW5wKeoQT5jOYHWw+AGEK7F+kp5x0dBpafyAqENrfOFU
3CLPXQpCaj5D1YgNgGbTTKPOTOLRZRjuLCOaLyRozlzGS8twWrt3irCKILmnwxaTpWCkHr2G/wVLH1hA+mxoO+v5dJlgoFcteTjL54yl2nOhJr
orNw8uonTfQ1Sjfgla8hdE0aEC3A3Y/TT5JF4ER9h6HXnwW/MGfSV3C29ET6a1LHfq/NA70gpDVrxuPJtSsmQKX41FsVe1PyrQDucRXfvdm
p
</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-10">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser soap"/>
</ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#id-11">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
<ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser"/>
</ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>1hQbF3AD/q+9B4ywN9OoveKWygk=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>EbDCuENg07suDNEDzNqjkkcsZ+YDYupCG1hK0fwpQRbZGtXXT9XdZJ0c+ZgebB+Z1PdYmM/X7bV1
Htyx+c8fAizJp4m8oh28GewlbCX89VY+rK+Zb0hiFyB47Guz1ywJWZi+Boj1bYe/LFlb8ZjagYIR
qDwspE/77e1XaEVGCPtHAQZD+bR46WbgeGRRRylqVUIG4IW+Z8bNCuXT2GAQLM3rKDVKSF+7Ti5n
/3KI3K+sCTxpuWXmfeqa9ltbKtPWeYZSYDqb93HM+dBtXQhL0wTKPLzJX0wQvLbRCz21gJ3mU4oJ
qAe1sB0mcyDpA5lF7dwMV9QNk9GiIVHuhb/25A==</ds:SignatureValue>
<ds:KeyInfo Id="KI-325E11B992A8C7CD1114216530529O118">
<wsse:SecurityTokenReference wsu:Id="STR-325E11B992A8C7CD1114216530529O119">
<wsse:Reference URI="#IdX509-325E11B992A8C7CD1114216530529O117" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
<sec:RequestHeader>
<sec:SenderId>1F0682</sec:SenderId>
<sec:SignerId1>1F0682</sec:SignerId1>
<sec:SignerId2/>
<sec:SignerId3/>
</sec:RequestHeader>
</soap:Header>
<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-11">
<ser:Echo>
<ser:Input>
<ser:Message>Hello</ser:Message>
</ser:Input>
</ser:Echo>
</soap:Body>
</soap:Envelope>
```

5.13 Encrypting the initially signed request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:oas="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:sec="http://www.danskebank.com/SecureSoapSGW"
xmlns:ser="http://www.danskebank.com/services/">
<soap:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="Id325E11B992A8C7CD11142165489988724">MIIEATCCAumgAwI...2e54Ihb7iK+B1cDc73Sc</wsse:BinarySecurityToken>
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-325E11B992A8C7CD11142165489988723">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference>
<wsse:Reference URI="#Id325E11B992A8C7CD11142165489988724" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData><xenc:CipherValue>wQ1abihx8Z5fQ/iqeUxk4Lh/AXcN8dgKl/RAIkMVAgQZf7vhrq/ZfwbaN32Yk2wm834bplcKv4rrowP8g
7UVCyghB0Tw06gl00hrwSAFK31ijC7E7V1qghBGlUNdH6NcK+p4frJ4YI08HRN78MwpDcKjmG05dlk9w0TsWniMjrj9MtQl4lasVd0GvoCfK
Eqna66ZeMqrCAIeX+TcjL4DQntJTot6mETS7G0RkwfQ6L0QalkqJom7ys6ri3isg/EsLaW1Uuc3aQMFUX6cKlkNuxbV5m467RNb/MBuTXDEZ
geq1NDwBJfE8ttkcsfB3JDFojm0dgznsP4vcwBLidw0zA==
</xenc:CipherValue></xenc:CipherData>
<xenc:ReferenceList><xenc:DataReference URI="#ED-13"/></xenc:ReferenceList>
</xenc:EncryptedKey>
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="IdX509-
325E11B992A8C7CD11142165489984320">MIID4DCCAsigAwI...1FsVe1PyrQDucRXfvdmp</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-12">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser soap"/></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#id-11">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser"/></ds:Transform>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>sMK91Qdpv/Zs496TS62w06wLcQ0=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>VsNHEcuAs7bw7eSnEpeXUS9U024v+I9gRQ63PZT/VH/eIq8DW2egAxqqSKRr7TnyKZm3oFygf400Ehz77CbuB63TFB
Csu8YucWkNb4jZdhwZYvXJu2c+eu3sbDX5NMPFaURC4FiGp5hde/C4MulEmkaPGRHtVUBgNrK8vhMNRwBUjmiYPsoAi0lHAqby4YcjGzye
VhUQRB1Y1XAaqiC17iRkV4vmg1iR9sAM
StaYdMyBQUYGwWw7oHKuic8p/ItjlX9+Iz+GIcOw6qX9NJtbjwmMjJQgNv5oh18U31tfx8St2XDN4bnaBko5W6CZIUKhK6LFY4qQpj32JAY7
WCgB/w==</ds:SignatureValue>
<ds:KeyInfo Id="KI-325E11B992A8C7CD11142165489984321">
<wsse:SecurityTokenReference wsu:Id="STR-325E11B992A8C7CD11142165489984322">
<wsse:Reference URI="#IdX509-325E11B992A8C7CD11142165489984320" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo></ds:Signature></wsse:Security>
<sec:RequestHeader>
<sec:SenderId>1F0682</sec:SenderId>
<sec:SignerId1>1F0682</sec:SignerId1>
<sec:SignerId2/>
<sec:SignerId3/>
<sec:DBCryptId/>
<sec:RequestId>123</sec:RequestId>
<sec:Timestamp>2014-03-31T14:04:50Z</sec:Timestamp>
<sec:Language>DK</sec:Language>
</sec:RequestHeader>
</soap:Header>
<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-11">
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="ED-13" Type="http://www.w3.org/2001/04/xmlenc#Content">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
<wsse:Reference URI="#EK-325E11B992A8C7CD11142165489988723"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>y5HIv5iXioWiAuxl/mZ3jWUhka3r7fqlC1SGbC/L/u6rI721o++PVLySHyr7S+0A6PXoA7Wh0J5m91PBTHrRxy2Rzo1k+
0sYP394zTgXfURYKxafSltonn5ICNYnb/GUihAZm4R5QClz8tbWY0NR1BQJj0kqpaEeXlmXvAZnX4FFLZim8Tpg0Po70o8XKD23lL/4mpsELR
DcEpJHjrt6vdtzzA0M1Rk3zW8ofxymkbiuYUpCYeQNQxg7edKiq/BCeuQn4u5VR1UTZhHrBaCD3K4h0TCaPDW3aEGYh5DfaofxigJm7I0X9J
QzZf8oc+KheR8xUxVkYxHxUBw7bdUh3UdPxg6A/73FQhSVGWBG/y8L6ar0z5uc6BMdJKuxx4xAydSQ5V0rMxFj9/Glic3gh5d/RU/mWcDBw
NF/VKzXuc1rc4lfl1b7Dw9F2TuAiYiSzG2tSlcqXmdvLUXQujE4kwxBSwaRqwcYrlMZGy0DxSQKf7hxUAGH1rSP/ukhcvwYCLKDAFnogwlY2vm
4cNT0znbAagn85tw
fzCX4RnjTLdpnNmT5N5NvLyfldFgzKA5h+40w/JNc222MtWA1/AkA4mrN8kyY16H80pmAxHCPMxmXLdlmVqtJ5cl3VBdmXZTJ</xenc:Ciph
erValue>
</xenc:CipherData>
</xenc:EncryptedData>
```

5.14 Signing the encrypted request

```xml
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:oas="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" xmlns:sec="http://www.danskebank.com/SecureSoapSGW"
xmlns:ser="http://www.danskebank.com/services/">
<soap:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="IdX509-325E11B992A8C7CD11142165543835330">MIID5TCCAs2gAwI...JAiobmnM1d5KQI8ns=</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-17">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser soap"/></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#ED-16">
<ds:Transforms><ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="wsu oas sec ser soap"/></ds:Transform></ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>lYRwrNn8Ev23lPGVy/u9veVpSQA=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>dJ8OQ1ycgzu6OgifFGkWPuiWITlS8lysEr9toFZWVq5XZV6SKNxU8p+VP1TOOE99Yh455HcwnbYDh1NTBFrboLG9Zrl
XY7jN8YQgXyP2CQqCCk4+ZGltg+zLCLaG4GYPNCZILm39PTzNboyn8+oHmp7O87oroWUMX3h5BvgBmNqn5oEQ9U5phInPv/DlTUIzHhbb
GBy3K8MZT2ZgRxb9JmuFbx3oqdop163fbiOsNYaFhWFswKyZVrApwm3T6zcOsHkUUh1gp4dOLOO5BOfyhNTEATCpngpVirBCWLarDIztN2
wlsY6xZNpvXQaUH44w3yxVDrmRFSPq1+IQ9eRznA==</ds:SignatureValue>
<ds:KeyInfo Id="KI-325E11B992A8C7CD11142165543835331">
<wsse:SecurityTokenReference wsu:Id="STR-325E11B992A8C7CD11142165543835332"><wsse:Reference URI="#IdX509-
325E11B992A8C7CD11142165543835330" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"/></wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="Id325E11B992A8C7CD11142165543830829">MIIEATCCAumgAwI...2e54lhb7iK+B1cDc73Sc</wsse:BinarySecurityToken>
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="EK-325E11B992A8C7CD11142165543830828">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference>
<wsse:Reference URI="#Id325E11B992A8C7CD11142165543830829" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3"/></wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData><xenc:CipherValue>OzCUjsQpBQwQQulBuu3phwdlrJoXrjo1/sRXx2XlarXQcnjOL8Y/xmVhYDhM7OV4NRzlCTInEDmaCoRH
EO7/YJaBd3RJwpm8WLBPM5nKOytlXza5aavatFmgL+RsVSuslq7PKVDBpfM6/OG1gqaRw8pJwQHhCVPyP2xLH9KlOtYgtqSO9POV/Y9Gml
QiM+x4JtME2pGUovZnXtfvpnLQwREGXFvpsxvsw/vWurJLXaXJ+2hoOyEY8+jV2izO1wCd79acXbHKXZJ43JcNEE4QNFrfOimbC+xgW+315K
+PeDQLRms/zvmGv4DOREbHuCROHwESlUEa7XQLHalpZAKB6Q==</xenc:CipherValue>
</xenc:CipherData>
<xenc:ReferenceList><xenc:DataReference URI="#ED-16"/></xenc:ReferenceList>
</xenc:EncryptedKey>
<wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="IdX509-325E11B992A8C7CD11142165543827525">MIID3zCCAsegAwI...gWyjVpWV54pSo=</wsse:BinarySecurityToken>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-15">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser soap"/></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#id-11">
<ds:Transforms><ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="oas sec ser"/></ds:Transform></ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>9xgGWpo+qhRCBuoucsRODZdP2BU=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>vLGpAcUtQguHupENLWsSAUxkwBMwoY4hpe4exPSRSNET/jHeOGiphTmidq3UyOOf+Ow5REIhMIw4LxZw5tNBcIAa2c6
d8EA9OeZJNpYo8kD12UnMAbJaXMD5I4CcSJZa+XdgS3sAAdk+o7+lqnJCy3Co3O4zRGNaKe4KHSUy9NRwORiYzToOHTZHs4SbwOe7o8k
KgZPUVz8fdEBhh/A2Qdo1jqCt4lYf26w6M2d6Tvu9UjOVHtHLRS2uLOyOh5eP8mV1vhwwQ1iXtDf/9rXlH4ALncQCghm1G7lvxucT1DYv98W
tlj+Oc4uKSUmOufNoS/vrq8JEylmC9MIPGWFbPQ==</ds:SignatureValue>
<ds:KeyInfo Id="KI-325E11B992A8C7CD11142165543827526">
<wsse:SecurityTokenReference wsu:Id="STR-325E11B992A8C7CD11142165543827527">
<wsse:Reference URI="#IdX509-325E11B992A8C7CD11142165543827525" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
</ds:Signature>
</wsse:Security>
<sec:RequestHeader>
<sec:SenderId>1F0682</sec:SenderId>
<sec:SignerId1>1F0682</sec:SignerId1>
<sec:SignerId2/>
<sec:SignerId3/>
<sec:DBCryptId/>
<sec:RequestId>123</sec:RequestId>
<sec:Timestamp>2014-03-31T14:04:50Z</sec:Timestamp>
<sec:Language>DK</sec:Language>
</sec:RequestHeader>
</soap:Header>
```

```
<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-11">
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="ED-16" Type="http://www.w3.org/2001/04/xmlenc#Content">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-1.1#EncryptedKey">
<wsse:Reference URI="#EK-325E11B992A8C7CD11142165543830828"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherData>
<xenc:CipherValue>BxqzSWwdLBczl6UNqkVZ2cCSpAfvDdhhhtkYlu1ML1YKlAwrW7KWlXQ7Dxb1Ip5uJ1DR5GBZHNzLDBLGQ/wKEZ/OBB
GN9LcOO+r663r1/gv7tDzeW4uAko2LJY8FvINsqTX1VEReLHXmD7C3/Hzv+VrosstcVNGLyOJjammplRGySSuyGZBtquHeUdZ3YE7nrFJ/cjD
4OO2LJl9URNkzuo6HTHuyNw1KwuOAz22ujecWfQPPVXPvfOLiRpQebFTdR7JCbGwX51YACo5XZr5zAtNcYWd2PPNwzbnlpNK9HxJZfLLYH
p8mmPHXZjBzjOeTaNbRG4+TvNuiubdbtgJMdzWPmqtWRHioJ9Xe+z/mB2AqfzOW/RCOxUIABnvw+KOcPLVvJvQKGOSYnjO29r3oGjOYzO1
POZNvO5aLw5AaJPe+O8y6ZQPEVFv61nrPuTk+5cPly/Y1mR1yNYVYzaR966bRU6/YNCOwrCGhu7vlnDQc8AK6Zu4xQ4/Lqm8FOWyFZ5O
wEOA+AcbkkSQ+kJZoy+9QjMltfX/MwJ9OcDrPAvK8y7ssy+rMapCjqLDrNfaNy7ewekIDs6EfnshvyMVUI5MGy6dGckhF7hRgO++SVoooUgUlX
fhOj9QoGSJ2BQnt</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedData>
</soap:Body>
</soap:Envelope>
```

The final example of the signed encrypted message also exemplifies the proprietary security setup via colour codes:

- Green: The last signature on the EncryptedData element (#ED-16)

- Blue: The encryption made on the content of the Body element, which then becomes the EncryptedData element

- Yellow: The first signature on the Body element (#ID-11)

6 Key terms and definitions

Terms	Definitions
Alias	See Merchant ID
API	Application Programming Interface
DIBS	Dansk Internet Betalings System
GitHub	GitHub - Code sharing service. It is a Git repository web-based hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.
Merchant app	Synonym for an app that involves payment transactions - typically related to selling of goods or services.
Merchant ID	Merchant identification number - A unique merchant ID provided by Danske Bank.
Order ID	Order identification number - Here referring to the unique provided order ID (Shop's order ID) sent along with a payment request from a merchant (app) to MobilePay (app)
Payment ID	See Transaction ID
PKI	Public Key Infrastructure
REST	Representational state transfer (REST) is a simple stateless architecture that generally runs over HTTP. REST is used between the MobilePay app and the MobilePay backend.
SDK	Software Development Kit - MobilePay AppSwitch SDK is designed to be embedded in a merchant app.
SSL	Secure Sockets Layer- a standard cryptographic protocol designed to provide communication security over the Internet.
SOAP	Simple Object Access Protocol. SOAP is used between the merchant backend and the MobilePay backend.
Transaction ID	DIBS provided payment transaction ID
WSDL	Web Services Description Language - is an XML-based interface definition language that is used for describing the functionality offered by a web service.
WSDL file	An acronym used for any specific WSDL description of a web service, which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.