PHP-klassbibliotek för automatisering av pen-ningtransaktioner

Jonne Sundell

Examensarbete

Informationsteknik

2012

EXAMENSARBETE	
Arcada – Nylands svenska yrkeshögskola	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	3727
Författare:	Jonne Sundell
Arbetets namn:	PHP-klassbibliotek för automatisering av penningtransaktioner
Handledare (Arcada):	Göran Pulkkis
Uppdragsgivare:	Innovoice Oy

Sammandrag:

Innovoice Oy upprätthåller en fakturerings- och reskontratjänst på webben, där Innovoices kunder skapar sina fakturor och följer med sin reskontra. Innovoices kunders pengar cirkulerar via Innovoices bankkonto så att betalningarna skall kunna reskontras före de betalas vidare till kundernas bankkonton. Detta kräver dagligen manuellt arbete av Innovoices personal. Bankernas gemensamma ”Web Services” och PKI-standarder är en lösning för automatisering av dagliga banktransaktioner såsom ett företags betalningar.

Detta examensarbete är en redogörelse om vad det betyder för en programmerare att ta i bruk bankernas gemensamma ”Web Services” med programmeringsspråket PHP och PKI-standarder.

I den teoretiska delen behandlas först den PKI (Public Key Infrastructure), som används i bankernas ”Web Services” för att åstadkomma en säker kommunikation mellan kunden och banken. Därefter presenteras de olika tekniker som ingår i arkitekturen hos ”Web Services”.

I den praktiska delen beskrivs utvecklingen av ett PHP-klassbibliotek som möjliggör automatisering av ett företags betalningar (SEPA-banktransaktioner).

Nyckelord:	Innovoice Oy, Public Key Infrastructure, PKI, Web Services, SOAP, XML, PHP, SEPA, X.509
Sidantal:	61
Språk:	Svenska
Datum för godkännande:	14.12.2012

DEGREE THESIS	
Arcada University of Applied Sciences	
Degree Programme:	Information Technology
Identification number:	3727
Author:	Jonne Sundell
Title:	PHP Class Library for Automation of Payment Transactions
Supervisor (Arcada):	Göran Pulkkis
Commissioned by:	Innovoice Ltd.

Abstract:

Innovoice Ltd. provides its customers a web based invoicing and ledger software, where customers create invoices and follow a real-time ledger of their invoices. Payments from all paid invoices circulate through Innovoices´ bank account so they can be marked as paid in the ledger before the payments are transferred to the customers´ bank accounts. This requires manual work to be done daily by employees of Innovoice. Web Services and PKI standards provide a solution for automating bank transactions.

This thesis work is a report on what it means for a programmer to create PHP applications implementing the Web Services protocol and PKI standards.

The theoretical part of this thesis first describes the PKI (Public Key Infrastructure), which secures messaging between a bank and a customer. After this, the thesis describes the different techniques that are utilized in the Web Services architecture.

In the practical part of this thesis is described the development of a PHP class library that can be used to automate corporate payments (SEPA-bank transactions).

Keywords:	Innovoice Oy, Public Key Infrastructure, PKI, Web Services, SOAP, XML, PHP, SEPA, X.509
Number of pages:	61
Language:	Swedish
Date of acceptance:	14.12.2012

INNEHÅLL

Figurer

Förkortningar

CA	Certificate Authority
DSA	Digital Signature Algorithm
HTTP(S)	Hyper Text Transfer Protocol (Secure)
IMAP(S)	Internet Message Access Protocol (Secure)
ISO	International Standardization Organization
RSA	Rivest, Shamir, Adleman
PHP	Hypertext Preprocessor
PKI	Public Key Infrastructure
PKCS	Public Key Cryptography Standard
RA	Registration Authority
SOAP	Simple Object Access Protocol
SEPA	Single Euro Payment Area
S/MIME	Secure/Multipurpose Internet Mail Extensions
WS	Web Services
WS-S	Web Services-Security
WSDL	Web Services Description Language
XML	eXtensible Markup Language
XSD	XML Schema definition language

Definitioner

System	Med System refereras till Innovoice Oy:s fakturerings och reskontrasystem
”Web Services”	Med ”Web Services” refereras till en viss typs arkitektur i mjukvarusystem
WS	Med WS refereras till bankernas ”Web Services”-tjänst

FÖRORD

Jag vill tacka Göran Pulkkis som en handledare för att ha ställt upp för möten enligt min tidtabell alltid när det har behövts. Jag vill också tacka mina lärare Magnus Westerlund, Hanne Karlsson och Göran Pulkkis för att ha möjliggjort slutförandet av mina studier vid sidan om företagandet. Om ni inte hade stött mig med företagandet genom att vara flexibla med tidtabeller mm. så skulle jag troligtvis inte någonsin ha blivit färdig från Arcada. Till sist vill jag ännu hjärtligt tacka Magnus Westerlund för att ha tänt lågan hos mig till mjukvaruutveckling genom sitt intresse i programmeringskurserna och de elever som läst och jobbat hårt under hans kurser på Arcada.

1 INLEDNING

Innovoice Oy (Innovoice) erbjuder sina kunder en webb-baserad fakturerings- och reskontra-tjänst. Via webb-tjänsten skapar Innovoices kunder sina egna fakturor och ser också egna reskontran i realtid. Meningen är att systemet är så långt automatiserat, att alla dagliga uppgifter som systemet skall utföra (Utskickning av fakturor, påminnelser och indrivningar samt utförandet av banktransaktioner) inte kräver manuellt arbete av någon person.

Det kommer dagligen in betalningar på Innovoices bankkonto från Innovoices kunders betalda fakturor. Innovoice ansvarar för att utföra betalningarna vidare för sina kunder.

För tillfället krävs det dagligen manuellt arbete när en person loggar in till bankens e-tjänst och matar in betalningarna en åt gången.

1.1 Målsättning och syfte

Målsättningen med detta examensarbete är att möjliggöra automatisering av dagliga banktransaktioner genom att ta i bruk bankernas gemensamma **”Web Services” (WS) och Public Key Infrastructure (PKI) standarder**. Nyttan med detta är att man kan frigöra personer som utför manuellt arbete och säkerställa att inga mänskliga fel uppkommer vid inmatning av banktransaktionerna.

Syftet med detta arbete är att redogöra för en programmerare vad det innebär att ta i bruk och använda bankernas gemensamma WS och Public Key Infrastructure (PKI) standarder med programmeringsspråket PHP.

1.2 Metoder

Ibruktagandet av bankernas WS och PKI-standarder sker enligt dokumentationen av Nordeas WS (Nordea Bank Finland Plc. 2012), samt dokumentationen från Finansbranschens Centralförbund (FC, 2009). Med PHP-programmeringsspråket byggs ett klassbibliotek med funktionalitet för att utföra ett företags betalningar. Till klassbiblio-

teket hör också funktionalitet för att skapa ”ISO 20022”-standardiserad XML-baserad banktransaktionsinformation som används tillsammans med bankernas WS. Dessa meddelanden skapas enligt ”ISO 20022”-standardens dokumentation. Klassbiblioteket byggs som en självständig komponent, vilket möjliggör återanvändning av koden också i andra projekt. Klassbiblioteket kommer sedan att integreras in i Innovoices system.

1.3 Avgränsningar

Fastän WS-gränssnittet är definierat av Finansbranschens centralförbund tillsammans med banker, så uppehåller bankerna sina egna tjänstebeskrivningar och i detta examensarbete tas endast Nordeas tjänstebeskrivning i beaktande av bankernas egna beskrivningar.

Beskrivning av innehållet på de ”ISO 20022”-standardiserade informationsstrukturer är avgränsat från detta arbete p.g.a. bankernas WS-gränssnitt kräver inte någon viss struktur på den information som skickas till gränssnittet för att den kommer i Base64-kodat format. Läsaren visas dock varifrån man hittar beskrivningar till ”ISO 20022”-standardiserade informationsstrukturer.

2 PKI (PUBLIC KEY INFRASTRUCTURE)

PKI är en infrastruktur som baserar sig på digitala certifikat och krypteringsnycklar för att identifiera olika parter för varandra och för att kryptera meddelanden mellan parterna för att kunna åstadkomma säker kommunikation. För att infrastrukturen över huvudtaget skall kunna fungera, måste det finnas en samling av organisationer, mänskor, processer samt applikationer och algoritmer vilka övervakar och sköter utgivningen av certifikat. Eftersom infrastrukturen hålls ihop genom förtroende till de instanser som har en högre auktoritetsnivå än man själv har, kunde hela infrastrukturen falla ihop om de högsta auktoriteternas datasäkerhet blev komprometterad. (*Cryptography and Network Security*, 2006 s. 428-430)

2.1 Publika nyckelns kryptografi

Med publika nyckelns kryptografi menar man ett system där det används två olika nycklar (nyckelpar) för att kryptera och avkryptera meddelanden. Den ena av nycklarna som kallas den publika nyckeln, är som namnet säger, publik eller öppen för allmänheten och den kan vem som helst använda för att kryptera meddelanden som kan avkrypteras endast med samma nyckelpars privata nyckel. Den privata nyckeln hålls alltid hemlig av ägaren, och säkerhetens nivå i den publika nyckels kryptografi beror på den privata nyckelns säkerhet.

Denna typs kryptografi använder sig av asymmetriska krypteringsalgoritmer för att skapa krypterade meddelanden.

Den publika nyckelns kryptografi har två huvudsakliga användningsändamål:

1. Kryptering av meddelanden med den publika nyckeln
2. Signering av meddelanden med den privata nyckeln

(*Cryptography and Network Security*, 2006 s. 257-268)

2.1.1 Asymmetriska krypteringsalgoritmer

Nyckelpar kan skapas antingen för krypteringsalgoritmen RSA eller för krypteringsalgoritmen DSA. Som namnet säger så är DSA-algoritmen menad för att skapa och verifiera digitala signaturer. Med DSA kan man inte kryptera meddelanden, utan till detta måste användas någon annan krypteringsalgoritm. Ofta används Diffie-Hellman -algoritmen för att skapa en gemensam nyckel på ett säkert sätt. Sedan används symmetriska krypteringsalgoritmer där den gemensamma hemliga nyckeln används för att kryptera och avkryptera meddelanden.

RSA kan användas både för att skapa och verifiera digitala signaturer samt för att kryptera meddelanden som skall skickas över nätet.

(RFC 2631, 1999)

DSA (Digital Signature Algorithm)

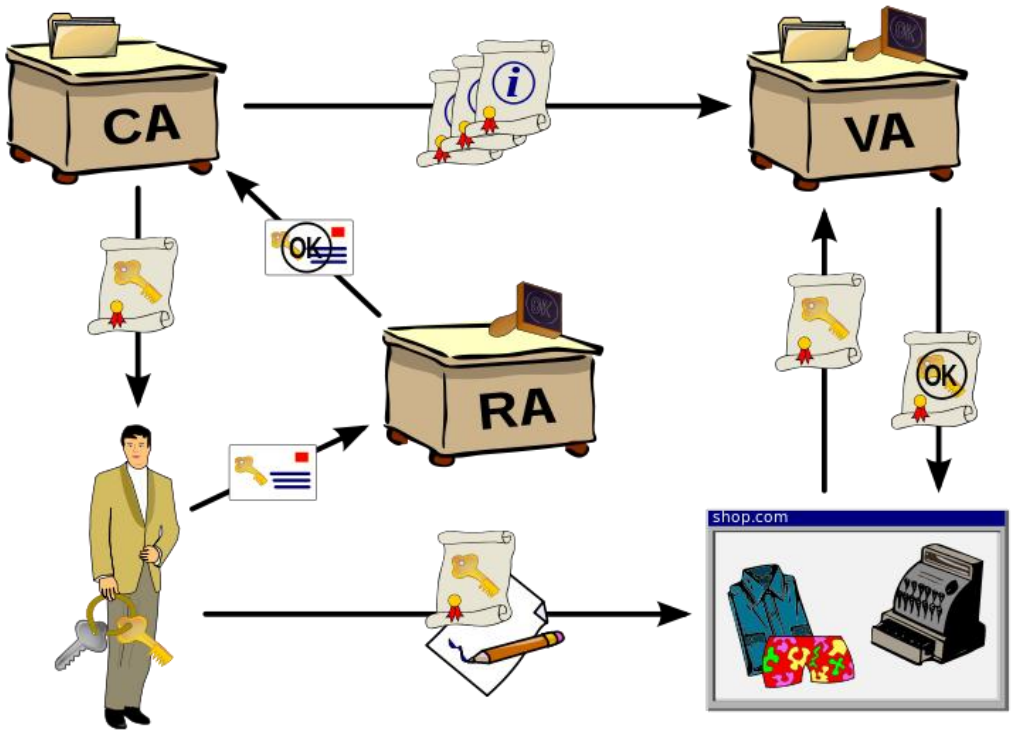
DSA är en matematisk algoritm som används för att skapa och verifiera digitala signaturer. (NIST, 2009)

RSA

RSA är världens mest använda kryptosystem. RSA kan användas för att kryptera meddelanden, utan att sändaren och mottagaren behöver ha en gemensam nyckel. (RSA Laboratories, 2002)

2.2 PKI-komponenter

De komponenter som bygger upp en PKI beskrivs i detta avsnitt. – Se Fig. 1.



Figur 1. Public Key Infrastructure (Wikipedia. 2007).

2.2.1 Certificate Authority (CA)

En CA är den instans som ger ut digitala certifikat, vilka innehåller en publik nyckel samt den publika nyckelns ägares identitet. En CA signerar sina utgivna certifikat med sin egen privata nyckel. Ett certifikat signerat av en känd CA är en verifikation på, att den publika nyckeln som ingår i certifikatet tillhör den person, organisation eller dator, som definieras av certifikatets s.k. subjekt. Det finns ett antal företag (t.ex. VeriSign,

DigiCert, Entrust, Thawte) som är färdigt anlitade av mjukvaror utgivna av stora programvaruföretag. Det är i en CA:s intresse att det digitala certifikatets ansökare är till en möjligast hög grad verifierad att vara den som han/hon/denne säger sig vara, så att CA:s anseende som en pålitlig instans inte tar skada. (NIST 2001, s. 15-17)

2.2.2 Registration Authority (RA)

RA är den instans som processerar en ansökan av ett certifikat. Dess uppgift är att verifiera ansökarens identitet och sedan registrera den i en användardatabas. Därefter uppmanas en CA att get ut ett digitalt certifikat till användaren. RA kan antingen vara skild från CA eller tillhöra samma organisation. Verifieringen av personen eller organisationen som ansöker ett certifikat görs beroende på graden av verifikation som behövs - antingen av mjukvara eller av någon person under övervakning. (NIST 2001, s. 15-17)

2.2.3 Certificate Repository

PKI-applikationer är till stor del beroende av katalogtjänster för distribution av certifikat, samt av den information som har att göra med certifikatens status.
Certificate Repository är en katalogstjänst där utgivna digitala certifikat sparas och kan verifieras vid behov av PKI-tillämpningar. (NIST 2001, s. 18)

2.2.4 Certificate Lifecycle Management

Till certifikatens livscykel tillhör ett antal procedurer och händelser såsom skapandet av nyckelparet, skapandet av ett certifikat, signering av certifikatet, publicering av certifikatet i vissa databaser där applikationer kan lätt och säkert få fram certifikat och publika nycklar. När certifikat är utgivna, kan en CA också spärra certifikaten (Certificate Revocation) vid behov.(NIST 2001, s. 30-32)

2.3 Viktiga PKI-standarder

I PKI inkluderas många standarder och specifikationer för olika procedurer, eller hur data skall struktureras. I detta kapitel presenteras ett antal vanliga och viktiga PKI-standarder.

2.3.1 X.509-certifikat

Digitala certifikat följer vanligen X.509-standarden. X.509-standarden definierar alla s.k. fält som måste finnas i ett certifikat samt acceptabla värden för alla fält.
Enligt X.509-standarden skall ett digitalt certifikat ha följande struktur:

- **Issuing CA** – Namnet på den CA som har gett ut certifikatet
- **CA Digital Signature** – Digitala signaturen av den CA som gett ut certifikatet
- **Version Number** – Den version av X.509 som detta certifikat baserar sig på
- **Serial Number** – Ett unikt nummer vilket identifierar detta certifikat
- **Subject/Owner** – Ägaren av detta certifikat. Möjlig ägare kan vara person, företag, nätverkskomponent, applikation osv.
- **Owner's Public Key –** Den publika nyckeln som tillhör detta certifikats ägare
- **Validity Period** – Datumen mellan vilka detta certifikat är i kraft
- **Certificate Usage** – Användningsändamål som detta certifikat är menat till
- **Signature Algorithm –** Hash- och ”digital signatur”-algoritmerna som har använts vid skapandet av detta certifikat

(RFC 2459, 1999)
I Fig. 2 visas innehållet av Innovoices certifikat med hjälp av OpenSSL-programmet.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            07:68:cf:da:5d:ad:63:4f:93:c3:c9:80:96:51:12:78
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com, CN=DigiCert High Assurance CA-3
        Validity
            Not Before: Sep 17 00:00:00 2012 GMT
            Not After : Dec 13 12:00:00 2013 GMT
        Subject: C=FI, ST=Uusimaa, L=Helsinki, O=Innovoice Oy, OU=ITC, CN=ssl.innovoice.fi
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b1:6d:7c:8b:87:64:d3:0a:90:1a:25:7f:bd:c3:
                    8a:78:d8:9f:1d:ea:28:df:be:28:e9:54:51:90:fe:
                    46:e4:70:3a:bc:e4:39:75:ba:e0:95:70:d6:49:3b:
                    26:11:ac:77:c1:fb:6a:03:5c:16:df:ec:7a:a4:f5:
                    73:0e:17:ab:df:3b:8c:d1:e4:49:5a:e5:ad:71:ad:
                    6d:2e:7c:9b:8a:81:70:d0:81:07:a2:c7:2d:3d:fd:
                    11:25:ef:d5:67:69:6d:43:56:db:f3:d6:a2:44:43:
                    69:7c:99:96:90:cc:97:6e:74:05:e5:0d:2b:18:b6:
                    98:a8:d8:72:39:29:af:6f:dc:8f:54:1c:62:9f:03:
                    b2:22:cf:4c:9d:92:d3:b9:0d:61:80:c5:08:78:96:
                    1c:f9:27:76:75:51:0a:52:68:45:70:7f:3d:bc:02:
                    37:91:42:96:72:03:f3:ab:7f:3d:97:9e:43:11:ed:
                    40:a1:c5:6a:f5:e8:a1:0a:20:51:e0:5a:9f:64:4e:
                    d8:c3:0b:3f:69:45:74:29:4a:64:7d:00:b6:a3:d0:
                    44:41:0d:bb:28:87:63:5c:2a:ab:d8:e8:a0:71:d1:
                    da:15:01:1e:fc:e2:c0:dc:d5:c6:c7:30:58:ab:58:
                    98:95:17:07:16:31:0d:76:85:53:3e:c0:ca:95:a4:
                    2e:3d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:50:EA:73:89:DB:29:FB:10:8F:9E:E5:01:20:D4:DE:79:99:48:83:F7

            X509v3 Subject Key Identifier:
                C9:E9:A2:E9:8B:EB:A7:D8:E2:A2:21:A7:72:4F:94:74:A1:74:B6:BC
            X509v3 Subject Alternative Name:
                DNS:ssl.innovoice.fi
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 CRL Distribution Points:
                URI:http://crl3.digicert.com/ca3-g14.crl
                URI:http://crl4.digicert.com/ca3-g14.crl

            X509v3 Certificate Policies:
                Policy: 2.16.840.1.114412.1.1
                  CPS: http://www.digicert.com/ssl-cps-repository.htm
                  User Notice:
                    Explicit Text:

            Authority Information Access:
                OCSP - URI:http://ocsp.digicert.com
                CA Issuers - URI:http://cacerts.digicert.com/DigiCertHighAssuranceCA-3.crt

            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: sha1WithRSAEncryption
        4f:96:bb:d5:32:ea:ff:84:c5:f9:92:8a:6c:a3:de:cb:4d:9f:
        af:c4:be:6b:5e:df:bc:4f:96:3b:81:77:99:78:58:36:b6:5a:
        10:34:67:d4:a6:cd:6d:b8:33:bb:0a:23:e2:3c:91:38:fc:d7:
        aa:4b:4f:dd:1d:ce:9e:89:7f:7c:5a:99:21:44:2b:eb:94:6d:
        2a:5f:69:58:87:77:cb:12:55:43:42:8a:c6:91:47:3a:f1:1f:
        82:d9:77:8d:02:a8:67:20:1f:0f:60:74:3b:05:fd:5f:c7:39:
        5a:c6:30:67:74:71:b4:96:3a:ca:45:bf:82:8b:fd:71:f0:ef:
        14:53:a0:ec:d9:db:36:3b:a5:f1:d3:c4:b0:8b:2d:9c:f5:4b:
        b4:1a:af:86:9d:03:fb:b9:f3:ea:3e:ea:46:a5:72:6b:7a:6a:
        3f:15:57:96:1d:6c:ef:08:5b:6d:ce:cf:3e:34:86:1c:b7:3e:
        88:3d:78:11:c0:6f:d8:9b:8f:01:b3:75:19:92:d8:e4:8f:95:
        39:b0:e0:0c:34:80:13:b1:18:c4:c9:85:a1:8b:99:04:93:be:
        51:28:e1:8f:f7:44:99:7d:c1:bc:ec:f2:64:59:5e:38:1e:a6:
        fe:05:95:77:06:a2:99:aa:a4:5b:8d:94:ce:85:23:e5:51:7b:
        dc:60:78:6d
```

Figur 2. Exempel på X.509-Certifikat (Innovoices servercertifikat. Certifikatutgivare: DigiCert).

2.3.2 Kryptering – PKCS

PKCS (Public Key Cryptography Standards) är en grupp av standarder utgivna av RSA Laboratories (RSA Laboratories 2012). Standarderna används inom publika nyckelns kryptografi. Viktiga standarder är

- **PKCS#1 - RSA Cryptography Standard**
  Standarden definierar realiseringen av publika nyckelns kryptografi baserad på RSA-algoritmen. Standarden beskrivs i (RSA Laboratories 2002).
- **PKCS#3 - Diffie-Hellman Key Agreement Standard**
  Standarden beskriver en metod hur två parter kan utan någon tidigare överenskommelse, komma överens om en hemlig nyckel som därefter kan användas till att kryptera meddelanden med symmetriska krypteringsalgoritmer. Standarden beskrivs i (RSA Laboratories 1993b).
- **PKCS#7 - Cryptographic Message Syntax Standard**
  Standarden beskriver en generell syntax för data såsom signerade och krypterade meddelanden. Standarden beskrivs i (RSA Laboratories 1993a).
- **PKCS#11 - Cryptographic token interface (Cryptoki, Crypto-key)**
  Denna standard definierar ett programmeringsgränssnitt (API) som kallas Cryptoki. Cryptoki definierar datatyper och funktioner tillgängliga för applikationer som programmeras med C-språket. Standarden beskrivs i (RSA Laboratories 2004).
- **PKCS#12 - Personal Information Exchange Syntax**
  Denna standard definierar ett mobilt filformat som kan användas för att spara eller transportera användarens nyckelpar och certifikat. Standarden beskrivs i (RSA Laboratories 1999).

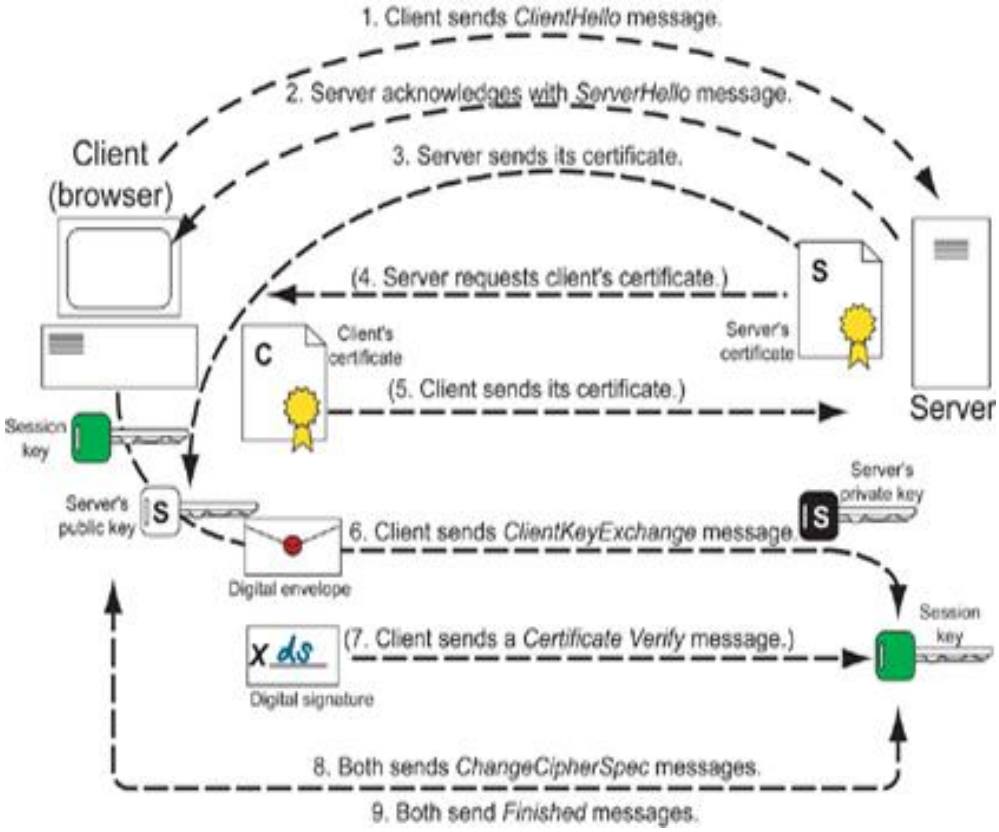
2.4 Tillämpningsområden av PKI

2.4.1 SSL-Protokollet

SSL-protokollet ger en möjlighet till både server- och klientmjukvara att autentisera sig för varandra och att skapa en säker förbindelse mellan sig, så att all trafik krypteras.

SSL-protokollet fungerar mellan transportlagret och applikationslagret.

En säker förbindelse baserad på SSL-protokollet mellan en klient och en server startar med det att klienten berättar för servern att all kommunikation skall köras över SSL. Detta görs oftast genom att till SSL ange en specifik port, som är annan än den som skulle normalt användas. Efter detta startar klienten och servern en SSL-handskakningsprocedur, där klienten och servern kommer överens om vissa parametrar, såsom krypteringsalgoritmen som skall användas samt den gemensamma nyckeln som meddelanden skall krypteras med – se Fig. 3. Till SSL-handskakningen hör också steget där servern identifierar sig för klienten med sitt digitala certifikat, varefter klienten kan verifiera att servern har den privata nyckel som är par med den publika nyckeln i certifikatet. Till SSL-protokollets karakteristik hör att servern måste presentera ett certifikat och att klienten kan ha ett certifikat.



Figur 3. SSL-handskakningsproceduren (MIT, 2003).

2.4.2 Servercertifiering

Följande tillämpningar är sådana där PKI används för att servern skall kunna autentisera sig för klienten, men klienten behöver inte autentisera för till servern.

Webbserver – HTTPS

En webbserver som använder SSL kan erbjuda säker kommunikation mellan klienten och servern. Alla sådana webbtjänster som kräver autentisering borde tillåtas endast över säkra förbindelser, annars skickas t.ex. användarnamnen och lösenordern i klartext över nätet.

e-postserver – IMAPS

E-postservrar som erbjuder en förbindelse över SSL med IMAPS-protokollet tryggar en säker förbindelse mellan användarens e-postklient och e-postservern. Ifall krypterade meddelanden har mottagits, så borde också förbindelsen mellan e-postklienten och e-postservern också vara skyddad, annars tappar man helt och hållet nyttan av att skicka krypterade meddelanden.

2.4.3 Personcertifiering

Elektroniska identitetskort

Det chipförsedda identitetskortet, som utfärdas av polisen, är försett med Befolkningsregistercentralens medborgarcertifikat. Med det elektroniska ID-kortet kan man identifiera sig själv till diverse nättjänster, digitalt signera- och kryptera meddelanden eller olika dokument. Kortet är ett tryggt sätt för dessa ändamål, eftersom det krävs att användaren matar in en PIN-kod vid användning av kortet. För inloggning på nättjänster används en identifikationskod (PIN 1). Elektronisk signatur görs med en signaturkod (PIN2). Den är juridiskt lika bindande och obestridlig som en traditionell underskrift. Kortet är giltigt i fem år. (FINeID, 2010)

Mobilcertifikat

Ett mobilcertifikat är en digital identitet i mobiltelefonens SIM-kort. Mobilcertifikatet möjliggör auktorisering av transaktioner i olika nättjänster och kan också användas för autentisering via mobiltelefonen till olika tjänster. Mobilcertifikatet sparas på SIM-kortet i mobiltelefonen. (Mobiilivarmenne, 2010)

S/MIME

S/MIME är en standard för säker e-postkommunikation där både den skickande och mottagande parten skall ha ett CA-signerat certifikat. (RFC 3851, 2004)

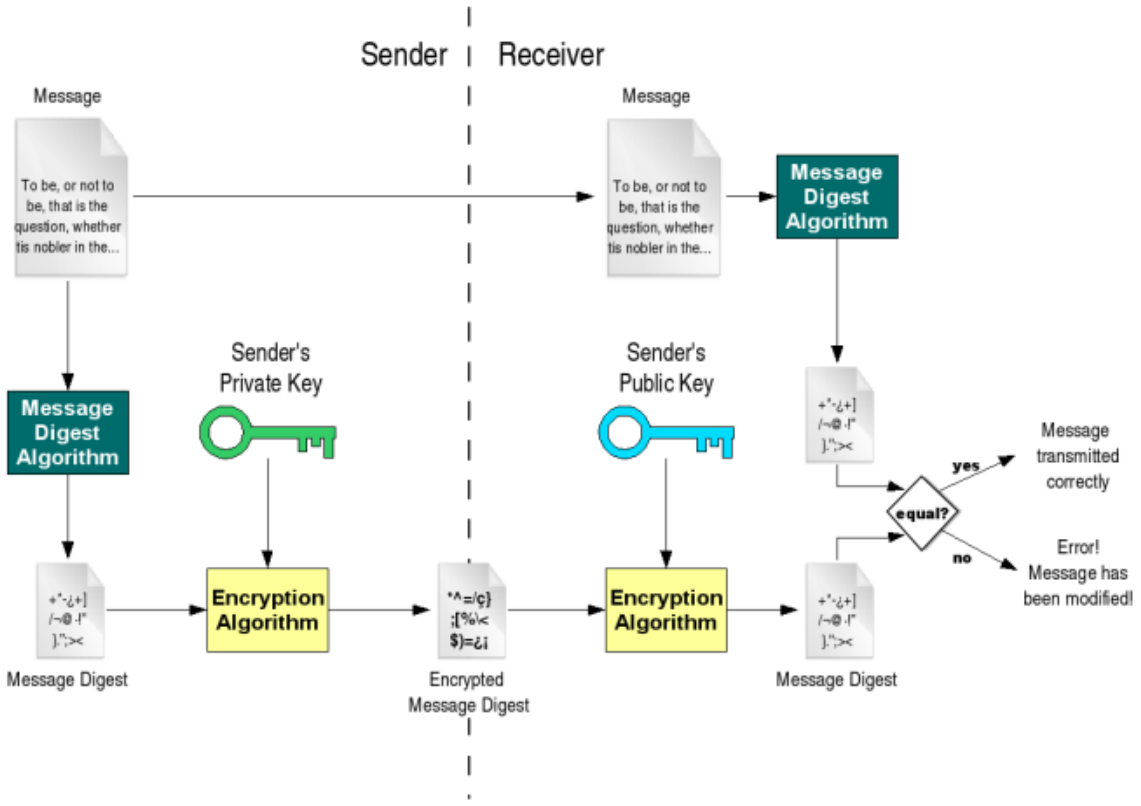
Digital signering

Digitala signaturer används för att verifiera identiteten av någon som skickat ett meddelande, samt för att kunna lita på integriteten av innehållet i meddelandet.

Digitala signaturer skapas och används på följande sätt (illustrerat i Fig. 4):

Person A vill signera ett meddelande som han vill skicka med e-post till Person B.

Person A skapar ett extrakt (Message Digest) av hela meddelandet med hjälp av en matematisk hash-funktion (Message Digest Algorithm). Extraktet är ett fingeravtryck av meddelandet och ifall någonting i hela texten skulle ändras, så kommer extraktet som skapas av hash-funktionen också att se annorlunda ut. Sedan krypterar Person A extraktet med den egna privata nyckeln. Det krypterade extraktet är den digitala signaturen. Person A skickar sedan både meddelandet och den digitala signaturen till Person B, som sedan avkrypterar digitala signaturen med Person A:s publika nyckel. Efter att Person B har avkrypterat signaturen och fått fram extraktet, kan han med samma hash-funktion skapa ett eget extrakt av hela texten och jämföra med det han fått av Person A. Ifall båda extrakten är exakt lika, kan Person B vara säker på att meddelandet har på riktigt kommit från Person A och att meddelandet inte har blivit ändrat på vägen. Digital signering av ett meddelande betyder inte, att någon skulle bli förhindrad att läsa av meddelandet som skickas över nätet. Till detta måste kryptering användas.

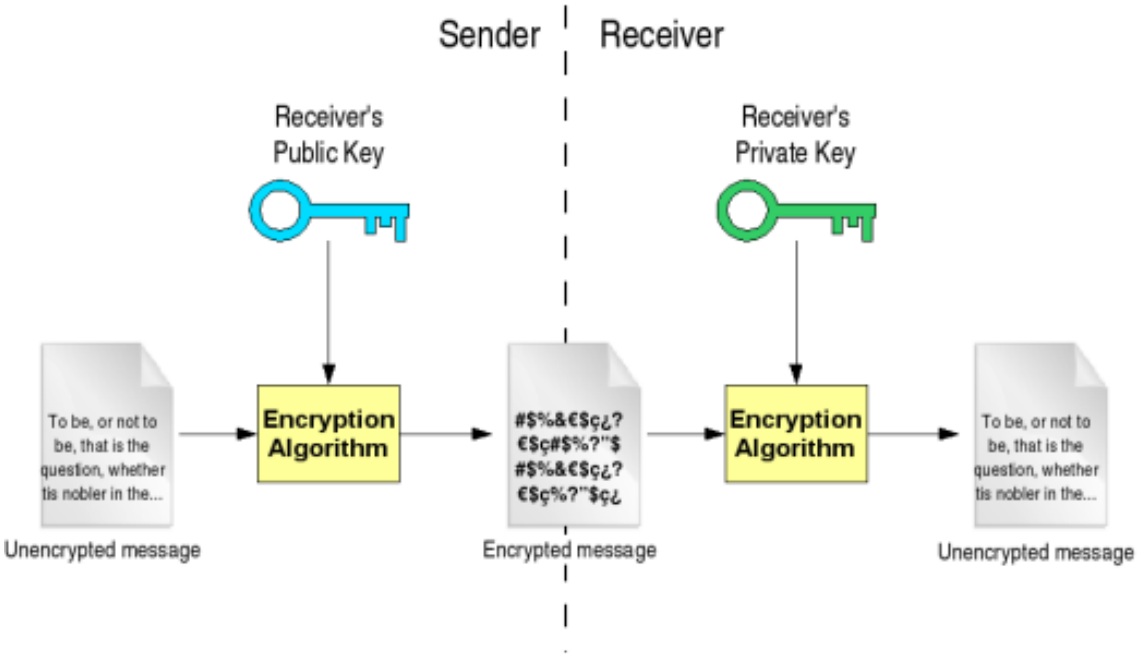


Figur 4. Processen för att digitalt signera information och verifiera en signatur (University of Chicago, 2004)

Kryptering

För att kunna säkra konfidentialiteten av ett meddelande som skickas över nätet, måste meddelandet krypteras. Kryptering av meddelanden fungerar på följande sätt (illustrerat i Fig. 5):

Person A vill skicka ett krypterat meddelande över nätet till Person B. Person A använder Person B:s publika nyckel och en asymmetrisk krypteringsalgoritm (Encryption Algorithm) såsom RSA för att kryptera meddelandet. Person A skickar meddelandet till Person B, som med hjälp av sin privata nyckel avkrypterar meddelandet som var krypterat med motsvarande publika nyckel.



Figur 5. Processen av kryptering och avkryptering av information med asymmetriska krypteringsalgoritmer (University of Chicago 2004)

Asymmetriska krypteringsalgoritmer är mycket långsammare än symmetriska algoritmer. Därför brukar man använda metoder såsom ”Diffie-Hellman Key Exchange”, för att få en krypteringsnyckel konfidentiellt utdelad till två parter och sedan använda den gemensamma krypteringsnyckeln och symmetriska krypteringsalgoritmer såsom AES (Advanced Encryption Standard) eller Blowfish för att kryptera meddelanden.

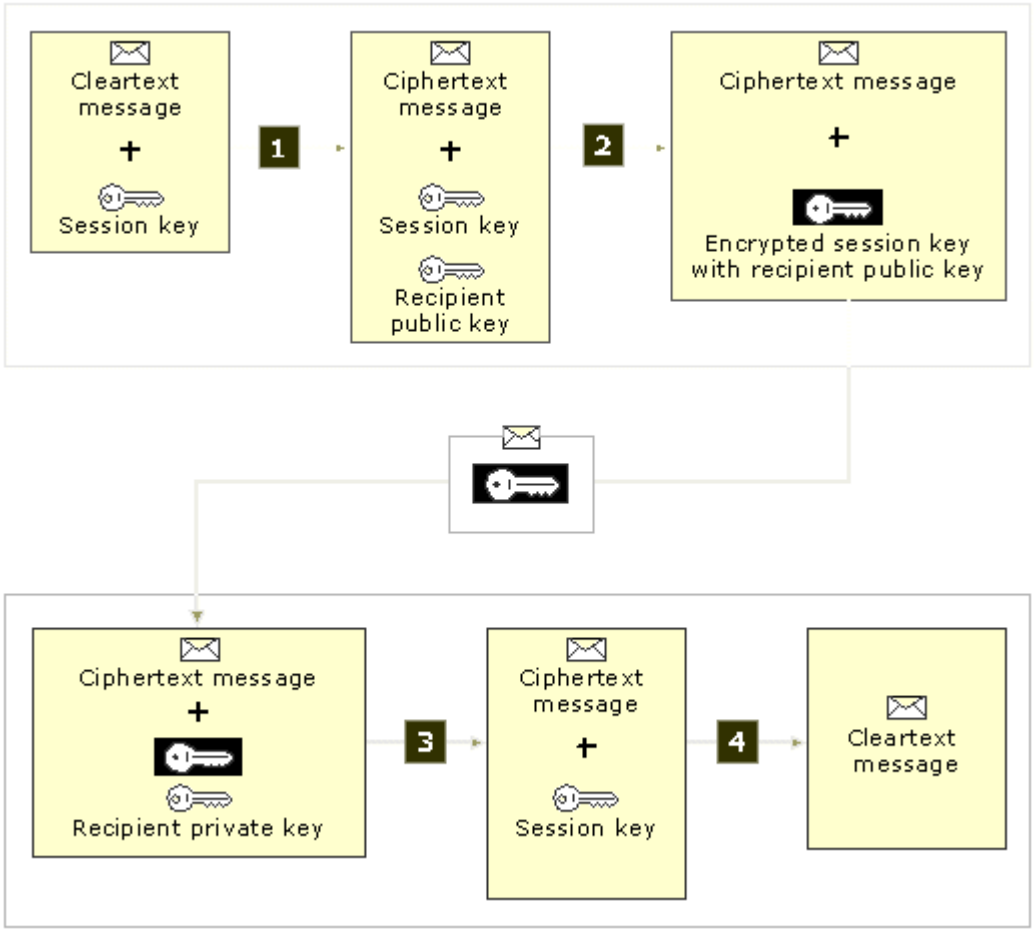
Vi använder S/MIME som exempel på hur den asymmetriska RSA-krypteringsalgoritmen och den symmetriska AES-krypteringsalgoritmen används tillsammans för att skapa en säker och effektiv kommunikation. (RSA Laboratories 1993a)

Krypterad e-post med S/MIME

Person A vill skicka ett krypterat e-postmeddelande till Person B (Illustrerat i Fig. 6).

1. Person A:s e-postprogram skapar en slumpmässig nyckel (sessionsnyckel) för användning med en symmetrisk krypteringsalgoritm (AES).
2. Person A:s e-postprogram krypterar hela meddelandet med AES-krypteringsalgoritmen och sessionsnyckeln.
3. Person A:s e-postprogram krypterar sessionsnyckeln med RSA-krypteringsalgoritmen och Person B:s publika nyckel.

4. Person A:s e-postprogram skapar ett paket av data inkluderande det krypterade meddelandet, den krypterade sessionsnyckeln, Person A:s digitala certifikat och en identifikation av den krypteringsalgoritm som har använts för att kryptera meddelandet.
5. Person A skickar e-postmeddelandet till Person B
6. Person B:s e-postprogram använder Person B:s privata nyckel för att avkryptera sessionsnyckeln.
7. Person B:s e-postprogram använder den avkrypterade sessionsnyckeln och den krypteringsalgoritm som Person A har skickat information om för att avkryptera själva meddelandet



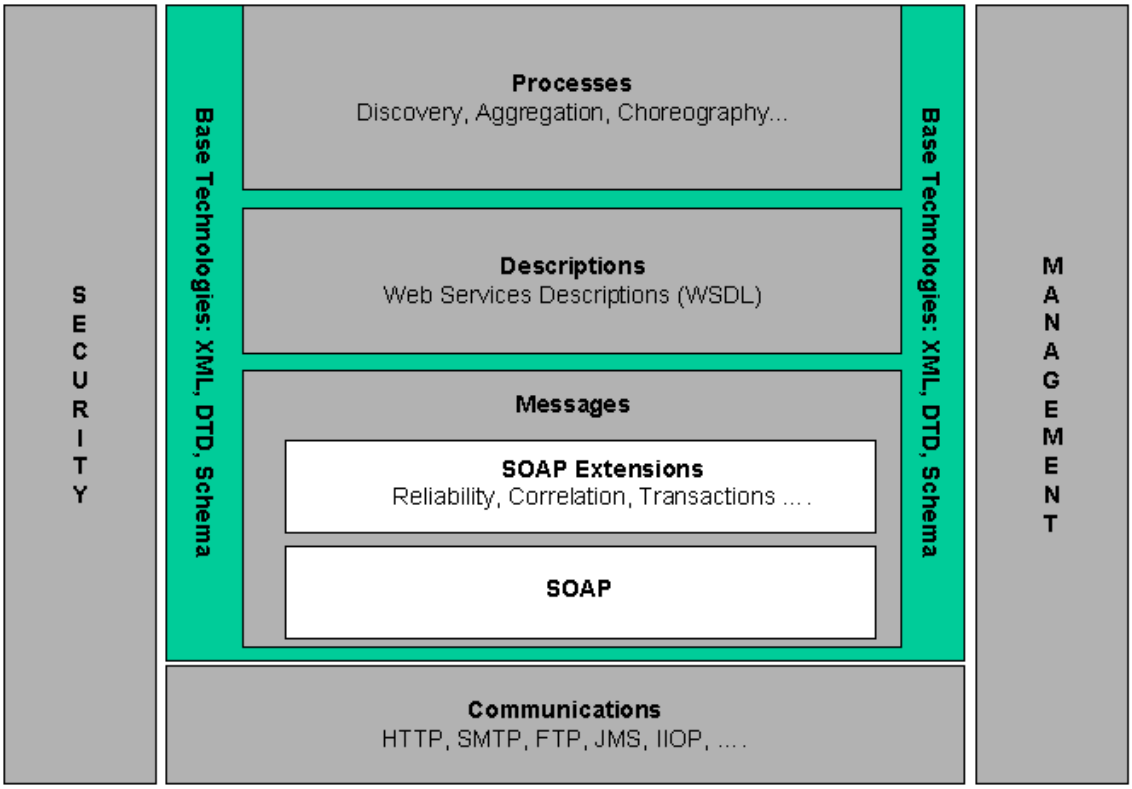
Figur 6. S/MIME-meddelandens kryptering och avkryptering (Microsoft, 2012).

3 WEB SERVICES OCH PHP

Med ”Web Services” refereras till en typ av mjukvarusystem-arkitektur. Det är ett webb-gränssnitt, som andra system och applikationer kan använda sig av för att kommunicera med ett system. W3C definierar ”Web Services” såsom ”A software system designed to support interoperable machine-to-machine interaction over a network”.

”Web Services” kan realiseras med många olika kommunikationsprotokoll och märkspråk. Den vanligaste typen av ”Web Services” är realiserad med XML som följer SOAP-standarden över HTTP. ”Web Services”-gränssnitt definieras som maskinspråk med WSDL-filer (Web Services Description Language). (W3C, 2007)

”Web Services”-arkitekturen inbäddar många olika teknologier åtskiljda på olika lager. I Fig. 7 visas W3C:s synvinkel på hur man kunde visualisera arkitekturen.



Figur 7. ”Web Services”-arkitekturens olika lager av utnyttjade teknologier (W3C, 2004a).

3.1 XML (Extensible Markup Language)

XML är ett enkelt textbaserat märkspråk för att strukturera information. XML är en av de mest använda formaten för tillgänglighet till samma strukturerade information mellan applikationer, personer och datorer mm. (W3C, 2010). I Fig. 8 visas ett enkelt exempel på hur ett kundregister kunde struktureras med märkspråket XML.

```xml
<?xml version="1.0" ?>
<kundregister>
    <kund typ="person">
        <namn>Petter Svensson</namn>
        <address>Svenssonsgatan 1</address>
        <postnummer>01234</postnummer>
        <postanstalt>Helsingfors</postanstalt>
    </kund>
    <kund typ="bolag">
        <namn>Svenssonsbolag AB</namn>
        <address>PL 123</address>
        <postnummer>01234</postnummer>
        <postanstalt>Helsingfors</postanstalt>
    </kund>
</kundregister>
```

Figur 8. Exempel på hur ett kundregister kunde byggas upp i XML-format.

3.1.1 XML digitala signaturer

”XML Signature Syntax and Processing”-dokumentet är en W3C-rekommendation om hur digitala signaturer skall användas i XML-meddelanden. XML digitala signaturer används för att säkra integriteten av information och för att autentisera signeraren av information. XML digitala signaturer kan användas för att signera vilken typs information som helst, i antingen samma eller någon annan fil som själva signaturen är beskriven i.

Det finns tre olika typer av XML digitala signaturer (W3C, 2008):

- **Enveloped**

  XML digitala signaturen är definierad i samma dokument som den signerar

- **Enveloping**

  XML digitala signaturen inkapslar den signerade informationen inom **signature**-elementet som definieras av XML digitala signaturen

- **Detached**

  XML digitala signaturen signerar ett externt dokument som refereras till med en URI (Universal Resource Identifier)

3.2 XSD (XML Schema Definition)

XML-scheman används för att definiera struktur och datatyper för XML-information. XSD är skriven med märkspråket XML. XSD kan användas för att validera XML-informationens struktur. (*W3C, 2004b*)

3.3 WSDL (Web Service Description Language)

WSDL är ett XML-baserat märkspråk för att definiera ”Web Services”-gränssnitt i sådant format som andra maskiner kan tolka. WSDL-filer kan vara mycket komplexa (se Fig. 9) och därför används det ofta diverse verktyg för att generera WSDL-filer från färdiga klasser i applikationer. I WSDL-filer definieras datatyper och värd-domän för in- och ut-parametrar till alla operationer, samt alla tillgängliga funktioner och nätverksadresserna till dessa. (*W3C*, 2007)

För att underlätta programmering mot ”Web Services”-gränssnitt, finns i många programmeringsmiljöer såsom t.ex. Java, .NET och PHP färdiga klassbibliotek med funktionalitet för att läsa in WSDL-filer och skapa korrekta datatypers objekt från dessa.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://bxd.fi/CorporateFileService"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:ns1="http://model.bxd.fi"
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:tns="http://bxd.fi/CorporateFileService" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xsd:schema targetNamespace="http://model.bxd.fi" elementFormDefault="qualified" attributeFormDefault="qualified"
            <xsd:complexType name="RequestHeader">
                <xsd:sequence>
                    <xsd:element name="SenderId" type="xsd:string" nillable="false"/>
                    <xsd:element name="RequestId" type="xsd:string" nillable="false"/>
                    <xsd:element name="Timestamp" type="xsd:dateTime" nillable="false"/>
                    <xsd:element name="Language" type="xsd:string" nillable="true"/>
                    <xsd:element name="UserAgent" type="xsd:string" nillable="true"/>
                    <xsd:element name="ReceiverId" type="xsd:string" nillable="false"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="ResponseHeader">
```

Figur 9. Extrakt från bankernas WSDL-fil (FC, 2012).

3.4 SOAP

SOAP (Ursprungligen Simple Object Access Protocol) är ett XML-baserat ramverk för att flytta information över nätverk i en distribuerad miljö. SOAP kan användas över vil-

ket som helst transportprotokoll (HTTP, TCP, SMTP osv.) och är menad för att fungera interoperabilt mellan olika plattformer och programmeringsspråk. (W3C, 2000)

3.4.1 SOAP-meddelanden

Alla meddelanden till och från SOAP-”Web Services” måste innehålla ett sk. SOAP-kuvert (eng. SOAP Envelope). Kuvertens struktur definieras av SOAP-standarderna 1.1 (W3C, 2001), och 1.2. (W3C, 2006). Ett SOAP-kuvert (se Fig. 10) måste definiera ett **Body**-element, där all information skall placeras. I ett SOAP-kuvert kan man också definiera ett **Header**-element, som är menat för övrig information, såsom digitala signaturer, certifikat och motsvarande säkerhetsdetaljer.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Figur 10. Skelettet hos ett SOAP-kuvert.

3.4.2 Felhantering i SOAP

Felhantering görs med ett standardiserat sätt: SOAP Fault. Ifall det uppkommer ett fel vid processering av ett SOAP-meddelande, så kommer sändaren att få ett **Fault**-element innanför **Body**-elementet. Fault-elementet innehåller information om felet som uppkommit, såsom en kod för felet, vad som har orsakat felet och en detaljerad beskrivning av felet. (W3C, 2000)

3.5 Web Services Security (WS-Security, WSS)

WS-Security är en utvidgning av SOAP-protokollet. WS-Security-specifikationen definierar hur olika säkerhetsmodeller (X.509, Kerberos mm.) skall förverkligas för SOAP-meddelanden för att kunna säkra integriteten och konfidentialiteten av informationen i SOAP-meddelanden. (OASIS, 2006)

WS-Security-direktiv definieras i SOAP-meddelandets **Header**-del och alla säkerhetsrelaterade direktiv skall vara underelement till ett **Security**-element.

Fig. 11 demonstrerar användningen av digitala signaturer i SOAP-meddelanden. I <*BinarySecurityToken*>-elementet skickas X.509-certifikatet i Base64-kodad form för att mottagaren skall kunna verifiera den digitala signaturen. <*SignedInfo*>-elementet innehåller information om använda algoritmer samt extraktet som är använt vid skapandet av den digitala signaturen. <*Signature*>-elementet innehåller den digitala signaturen.

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="...">
   <S11:Header>
      <wsse:Security>
         <wsse:BinarySecurityToken
                     ValueType="...#X509v3"
                     EncodingType="...#Base64Binary"
                     wsu:Id="X509Token">
                  MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
         </wsse:BinarySecurityToken>
         <ds:Signature>
            <ds:SignedInfo>
               <ds:CanonicalizationMethod Algorithm=
                     "http://www.w3.org/2001/10/xml-exc-c14n#"/>
               <ds:SignatureMethod Algorithm=
                     "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <ds:Reference URI="#myBody">
                  <ds:Transforms>
                     <ds:Transform Algorithm=
                           "http://www.w3.org/2001/10/xml-exc-c14n#"/>
                  </ds:Transforms>
                  <ds:DigestMethod Algorithm=
                        "http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <ds:DigestValue>EULddytSo1...</ds:DigestValue>
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>
              BL8jdfToEb11/vXcMZNNjPOV...
            </ds:SignatureValue>
            <ds:KeyInfo>
                <wsse:SecurityTokenReference>
                    <wsse:Reference URI="#X509Token"/>
                </wsse:SecurityTokenReference>
            </ds:KeyInfo>
         </ds:Signature>
      </wsse:Security>
   </S11:Header>
   <S11:Body wsu:Id="myBody">
      <tru:StockSymbol xmlns:tru="http://www.fabrikam123.com/payloads">
        QQQ
      </tru:StockSymbol>
   </S11:Body>
</S11:Envelope>
```

Figur 11. Användning av WS-Security och digitala signaturer i SOAP-meddelanden (OASIS, 2006).

3.5.1 PHP

PHP (Hypertext Preprocessor) är ett mycket använt skriptspråk med öppen källkod. PHP är avsett främst för webb-programmering. PHP kan inbakas i HTML-kod eller stå för sig själv. All PHP-kod exekveras av en webb-servers (Apache, IIS) PHP-kommandotolk när skriptet körs. (The PHP Group, 2005)

Exempel på ett mycket enkelt PHP-skript visas i Fig. 12.

```php
<?php
    $a = 5;
    $b = 2;
    $sum = $a + $b;
    echo "Detta är ett PHP-skript som kan addera.\n
    Summan av $a + $b är $sum";
?>
```

Vad som skrivs ut när PHP:s kommandotolk har exekverat skriptet:
Detta är ett PHP-skript som kan addera. Summan av 5 + 2 är 7

Figur 12. Exempel på ett enkelt PHP-skript som adderar ihop två tal och skriver ut resultatet.

3.6 PHP-programmering för SOAP Web Services

När man talar om PHP-programmering för SOAP Web Services, finns det ett par olika alternativ som är avsedda för tjänster definierade med WSDL-filer. Det finns såklart möjligheten att inte använda sig av WSDL-filer och manuellt skapa XML-information enligt SOAP-specifikationer samt sedan överföra informationen med hjälp av t.ex. Socket-kommunikation (Sockets, The PHP Group, 2012a). Detta kan dock vara mycket komplicerat och är avgränsat utanför detta examensarbete.

3.6.1 NuSOAP

NuSOAP är ett klassbibliotek för att skapa och använda SOAP Web Services med PHP. NuSOAP kräver inga specifika PHP-utvidgningar installerade för att fungera. NuSOAP har stöd för SOAP-specifikationen 1.1 och WSDL-specifikationen 1.1 (NuSOAP, 2002).

3.6.2 PHP:s SOAP-utvidgning

PHP:s SOAP-utvidgning (Extension) kan användas för att skapa både SOAP-servrar och SOAP-klienter med programmeringsspråket PHP. Utvidgningen kommer med i PHP version 5 och senare. Den har stöd för specifikationerna SOAP 1.1 och SOAP 1.2, samt för WSDL 1.1 (SOAP, The PHP Group, 2012b).

Utvidgningen saknar stöd för SOAP-utvidgningar, såsom **WS-Security** och lämpar sig därför bäst för ”Web Services”-gränssnitt som inte kräver säkerhetskomponenter realiserade med **WS-Security**.

I Fig. 13 visas hur SOAP-utvidgningen behöver endast fyra rader kod för att använda en tjänst över ett SOAP-gränssnitt.

```
//Specifiera addressen till en WSDL-fil, WSDL-filen definierar en metod "getText"
$wsdl = 'http://addressen-till-wsdl-filen/wsdl';

//Skapa en instans av SoapClient klassen
$soapClient = new SoapClient($wsdl);

//Kalla på "getText" metod över WS-gränssnittet. Metoden returner en textsträng
$response = $soapClient->getText();

//Skriv ut texten som gränssnittet har returnerat
echo $response->getTextResponse;
```

Figur 13. Demonstration på hur enkelt det kan vara att använda tjänster över SOAP-gränssnitt med PHP:s SOAP-utvidgning.

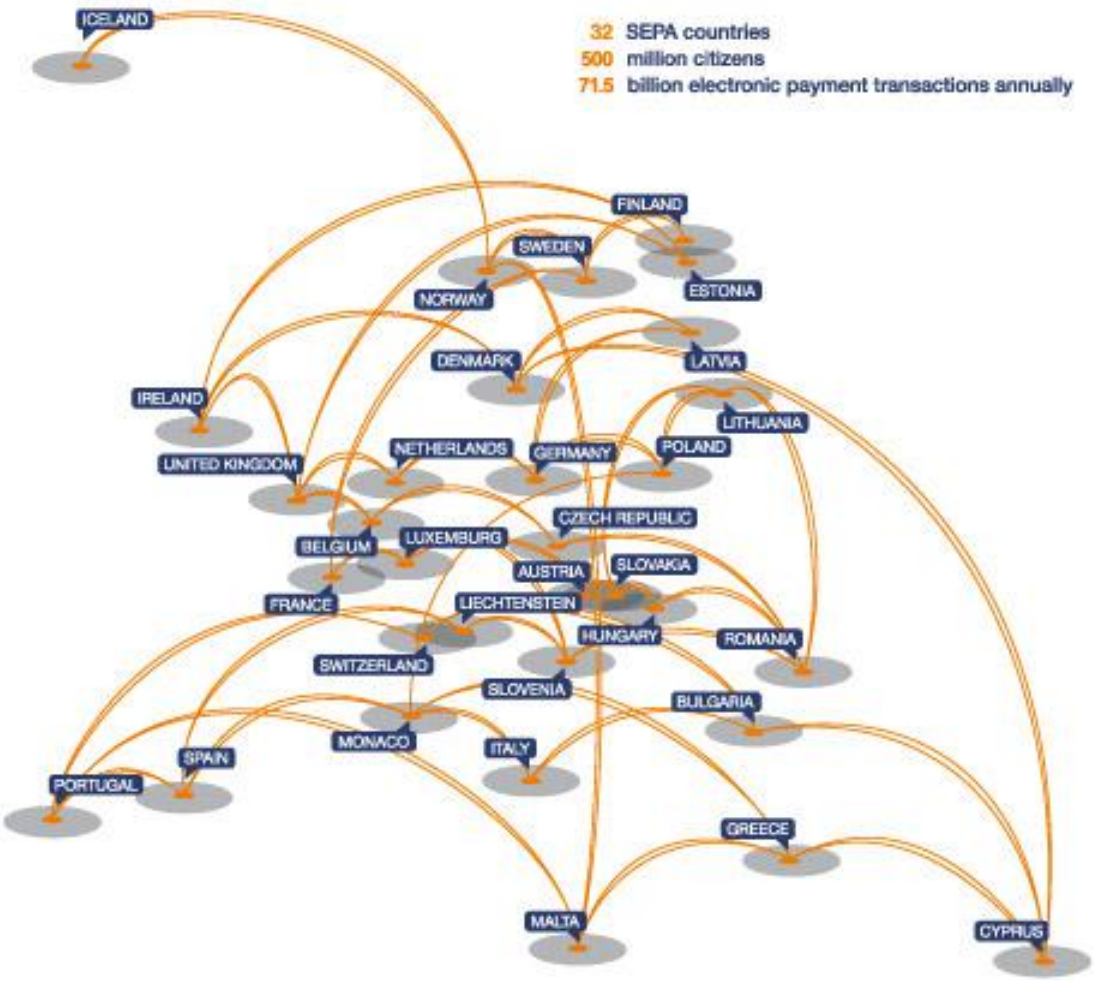
3.6.3 WSO2 Web Services Framework for PHP

WSO2 Web Services Framework for PHP är en PHP-utvidgning med öppen källkod. Ramverket fokuserar på interoperabilitet mellan distribuerade system programmerade såväl med PHP, Java eller i .NET-miljöer. WSO2 Web Services Framework for PHP är den enda PHP-utvidgningen med stöd för olika WS-* -utvidningar, såsom WS-Security och WS-Security Policy. WSO2 Web Services Framework for PHP inkluderar också funktionalitet för att generera WSDL-filer automatiskt från PHP-klassfiler. (WSO2, 2011)

4 ”WEB SERVICES” OCH PKI FÖR SEPA-BANKTRANSAKTIONER

SEPA (Single Euro Payment Area), är ett EU-initiativ för att skapa ett gemensamt betalningssystem inom EU. SEPA har hämtat med sig ”ISO 20022”-standarden, vilken definierar strukturen på XML-baserade banktransaktionsmeddelanden, som används innanför SEPA-länderna för banktransaktioner. Det finns totalt 32 länder (överblick av

dessa i Fig. 14) med i SEPA och initiativet berör 500 miljoner mänskor med 71,5 miljarder elektroniska betalningar årligen. (European Central Bank, 2012)



Figur 14. SEPA-medlemsstater (European Payments Council, 2008).

Web Services (WS) är ett protokoll för bankernas företagskunder. WS används vid överföring av information mellan kunden och banken. WS grundar sig på allmänna standarder såsom XML och SOAP. (Nordea Bank Finland Abp, 2011)

Bankernas gemensamma Web Services och PKI-standarder är utvecklade av Finansbranschens Centralförbund tillsammans med Nordea, OP-Pohjola–gruppen och Sampobanken. Ansvaret på underhåll av tjänstebeskrivningarna ligger hos bankerna. (FC, 2012)

4.1 WS-protokollet

WS-protokollet är avsett för att överföra XML-information såsom kontoutdrag och betalningar, vilka baserar sig på ”ISO 20022”-standarden (ISO 20022), eller på lokalt (bankspecifikt) material, såsom inkommande referensbetalningar. WS-protokollet är sessionslöst och alla transaktioner startas av kunden, dvs. kunden skickar till banken en sk. ApplicationRequest-begäran vid varje transaktion och banken svarar med ett ApplicationResponse-meddelande. In- och ut–meddellanden beskrivs närmare i avsnitt 4.4.

4.2 Avtal med banken

När kunden tar i bruk WS-protokollet, görs ett avtal med banken (Web Services agreement) och då väljer kunden ifall han vill ta i bruk användarspecifika eller hela företaget täckande certifikat för identifiering mot banken. De uppgifter som avtalet görs med kommer att användas när ett nyckelpar och certifikat skapas för kunden. I detta examensarbete behandlas endast fallet där hela företaget använder sig av ett certifikat.

4.3 Datasäkerheten i Web Services

Datasäkerheten i bankernas Web Services är realiserad med **PKI-standarder** (mera om PKI i avsnitt 2). Datatrafiken mellan kunden och banken sker alltid över en SSL-krypterad förbindelse. Parterna identifierar sig för varandra med **PKI-certifikat** och all information som överförs signeras digitalt med privata nycklar för att säkerställa integriteten i meddelanden. (FC, 2009)

4.3.1 Certifikat och nyckelpar

Certifikatet som används i samband med Nordeas Web Services följer standarden X.509 (IETF 1999). Certifikatutgivaren (Certificate Authority) är Nordea. Detta PKI-förhållande inkluderar endast två parter (Nordea och kunden) och skiljer sig från den mera vanliga typen av PKI, där certifikatutgivaren är en tredje part.

Certifikatet används för att Nordea skall kunna känna igen kunden som tar kontakt över WS-protokollet. Nyckelparet som ingår i certifikatet används för att digitalt signera den

information som överförs. Med den digitala signaturen kan banken verifiera, att den person eller företag som gjort avtal med banken har godkänt informationen som överförs till banken. Digitala signaturen säkerställer också att informationen inte ändrats på vägen från kunden till banken.

4.3.2 Nedladdning av certifikat

Det finns ett par olika möjligheter gällande nedladdning och sparande av certifikat.
Detta examensarbete behandlar endast fallet där nedladdningen sköts manuellt. Kunden kan ladda ner certifikatet från Nordeas hemsidor med hjälp av en kod, som den person som skrivit under avtalet får per SMS till sin mobiltelefon. Mer specifika instruktioner hittas i dokumentet ”Web Services filöverföring, tjänstebeskrivning”, avsnitt 9.1.3. (Nordea Bank Finland Abp. 2012)

4.3.3 Certifikatets utgång och förnyelse

Filbaserade certifikat utgivna av banken står i kraft i två år, varefter det inte är möjligt att öppna en WS-förbindelse till banken. För att säkerställa kontinuerlig funktion av systemet, kan certifikaten förnyas automatiskt i god tid före de går ut. Nordea erbjuder kunderna en Web Service där kunden kan ladda ner ett nytt certifikat med att digitalt signera certifikatbegäran med nuvarande giltiga certifikat.

4.4 Användning av bankernas WS

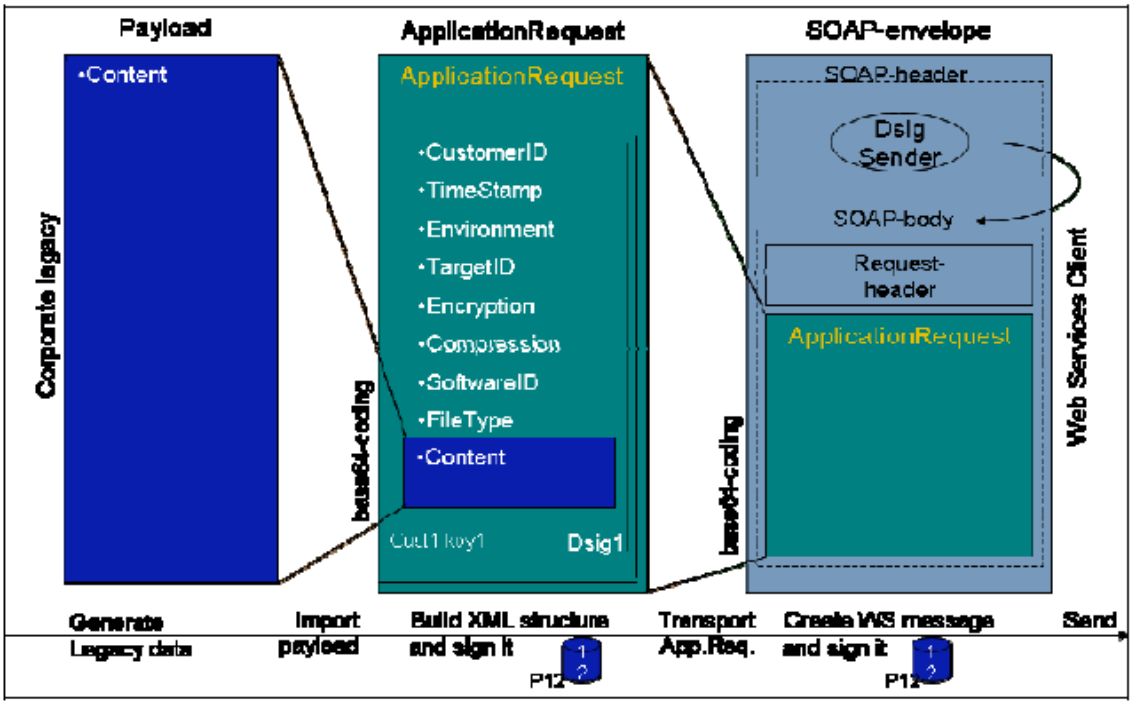
Användning av bankernas WS måste ske med en ”Nedifrån upp”-taktik, eftersom det slutliga SOAP-meddelandet som överförs består av flera datalager inkapslade innanför varandra (se Fig. 15).

Den information som skickas till banken som ett SOAP-meddelande, innehåller all den information som banken behöver för att kunna autentisera användaren. Bankernas WS används i två steg:

1. **Skapa informationen enligt dokumentationer**
2. **Skicka informationen till bankens WS**

För att beskriva **användning av bankernas WS**, måste vi dock spjälka upp processen i mindre delar och då har vi följande skeden (*FC.* 2009):

1. Skapa ”Payload”-informationen som överförs till banken. Informationen kan antingen vara enligt ”ISO 20022”-standarden eller enligt bankens egna definitioner
2. Utför Base64-kodning av ”Payload”-informationen
3. Skapa informationen för ”Application Request”-delen
4. Lägg den Base64-kodade ”Payload”-informationen i ”Application Request”-informationens ”Content”-fält
5. Utför Base64-kodning av ”Application Request”-informationen
6. Digitalt signera (Enveloped type XML Digital Signature) hela Base64-kodade ”Application Request”-informationen med den privata nyckeln i det certifierade nyckelparet.
7. Utför Base64-kodning av informationen som innehåller ”Application Request”-delen samt den digitala signaturen
8. Skapa ett SOAP-meddelande för användning av en viss operation (UploadFile, DownloadFile mm.)
9. Lägg till den signerade ”ApplicationRequest”-informationen i SOAP-meddelandets ”ApplicationRequest”-element
10. Digitalt signera (detached type XML Digital Signature) hela SOAP-meddelandet med det certifierade nyckelparets privata nyckel och lägg till den digitala signaturen i ”SOAP header”-elementet.
11. Skicka SOAP-meddelandet till bankens ”Web Service”-gränssnitt och vänta på ett svar.



Figur 15. Den information som överförs till banken via WS-gränssnittet består av tre lager: Payload, ApplicationRequest och SOAP-kuvertet (Nordea, 2011).

4.4.1 Payload-informationen

Payload-informationen är den information som definierar banktransaktionen. I företagets betalningar innehåller Payload-informationen information om betalaren, betalningens mottagare samt ett antal betalningar. I SEPA-betalningar är Payload-informationen alltid en XML-struktur som följer ”ISO 20022”-standarden. Samma XML-struktur används i alla SEPA-länder. ”ISO 20022”-information är närmare beskriven i Finanscentralens dokumentation: ISO 20022 maksut Opas (FC, 2010) och i diverse specifikationer av ”ISO 20022”-standarden (se *ISO 20022,* Universal financial industry message scheme: Full catalogue of ISO 20022 messages).

Skapandet av Payload-information är avgränsat från detta examensarbete p.g.a WS-gränssnittet inte kräver ett ”Content”-fält i ”ApplicationRequest”-informationen och detta används inte i andra funktioner än **uploadFile** (som används då man vill flytta information till banken).

4.4.2 ApplicationRequest-informationen

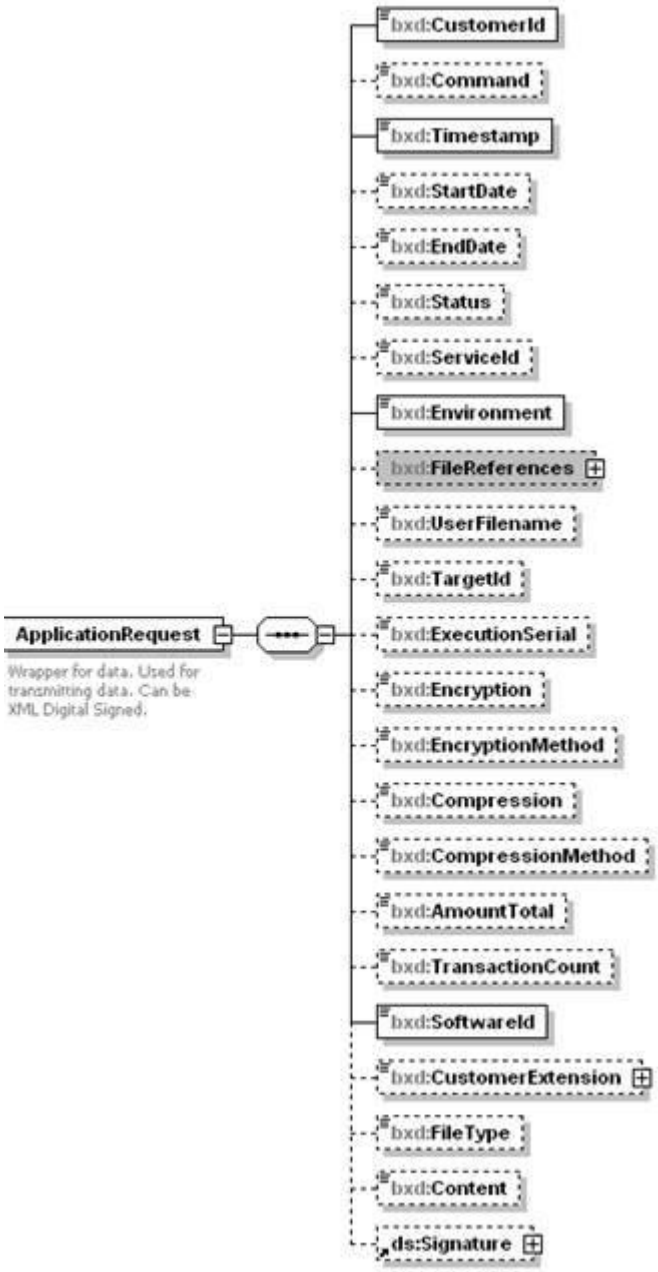
ApplicationRequest är en XML-struktur (se Fig. 16) vars huvudfunktion är att kapsla in den Base64-kodade Payload-informationen i ApplicationRequest-strukturens **Content**-element och digitalt signera denna information för att säkra integriteten i Payload-informationen. Den digitala signaturen transporteras i **Signature**-elementet. ApplicationRequest innehåller information om den operation som anropas på från bankernas WS, samt information om kunden och mjukvaran som tar kontakt.

4.4.3 ApplicationRequest-informationens XML digitala signatur

XML digitala signaturen, som används för att auktorisera användare hos banken samt för att säkra integriteten i meddelandet, är av typen ”Enveloped” (Business Signature). Denna skiljer sig från den typ av XML digitala signatur som används vid signering av SOAP-meddelanden, som är av typen ”Enveloping” (Transport Signature).

4.4.4 Validering av informationsstrukturen emot XSD

Nordea erbjuder för ApplicationRequest-informationen XSD-filer som kan nedladdas. Programmeraren kan använda sig av diverse verktyg för att validera strukturen av den skapade ApplicationRequest-informationen emot XSD-filen, både före och efter digital signering av informationen.



Figur 16. ApplicationRequest-informationens struktur (Nordea Bank Finland Plc. 2012).

4.4.5 SOAP-meddelandet

SOAP-meddelandet är den XML-baserade datastruktur (se Fig. 17) som används för att transportera informationen till banken. SOAP-meddelandets header-element innehåller WS-Security-instruktioner, kundens certifikat och den digitala signaturen av informationen som transporteras i body-elementet. Body-delen signeras digitalt med den privata nyckeln i kundens certifierade nyckelpar för att säkra att innehållet av body-elementet inte ändrats under vägen från kunden till banken. Hela SOAP-kuvertet signeras också digitalt för att banken skall kunna identifiera och autentisera kunden som har skickat informationen till bankens WS.

Ifall man inte ännu har ett kontrakt med Nordea, så erbjuder Nordea ett testcertifikat som kan användas för att testa förbindelsen till banken.

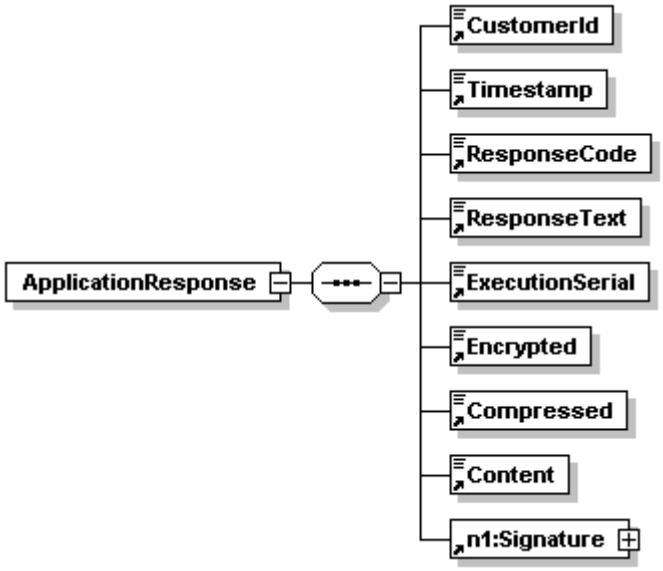
```
<soapenv:Envelope xmlns:cor="http://bxd.fi/CorporateFileService" xmlns:mod="http://model.bxd.fi"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd">
            <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="CertId-9502902"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">MIID.../DnjbkAZBo7vsj78zzdk7KNliBlqBclszdJ3dEHRWSI7FspRxyiR0NDm4lpyLwFtfw=
</wsse:BinarySecurityToken>
<ds:Signature Id="Signature-12345678" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
<ds:Reference URI="#id-4453123">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>zYeQGz0jnyy3tl5gruq+llGyzQo=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>m5fuzJnVOQGNsu4s2kfal+UTReUSz9pMxH...=</ds:SignatureValue>
<ds:KeyInfo Id="KeyId-98765432"><wsse:SecurityTokenReference wsu:Id="STRId-33454994"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wsse:Reference URI="#CertId-9502902" 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>
</soapenv:Header>
  <soapenv:Body wsu:Id="id-4453123" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
    <cor:getUserInfoin>
            <mod:RequestHeader>
              <mod:SenderId>11111111</mod:SenderId>
              <mod:RequestId>2</mod:RequestId>
              <mod:Timestamp>2009-11-30T13:18:33.000+03:00</mod:Timestamp>
              <mod:Language>FI</mod:Language>
              <mod:UserAgent>NordeaTest</mod:UserAgent>
              <mod:ReceiverId>123456789</mod:ReceiverId>
            </mod:RequestHeader>

<mod:ApplicationRequest>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiIHN0YW5kYWxvbm
U9InllcyIZGlMZVppUGdrOW5aMGFQQ1.....N4VkdXeFprK0thclVZL050RitzRnFSNit6DQpXWkM0Wnh5VV
dXcGt3YnBadjBsN3QxaGFBWE1YZVk1cmEzNGtsMTF0S0t5Z2Vlais1RXFOejVnVzdKZStBcVQwDQp4WX
p=</mod:ApplicationRequest>
    </cor:getUserInfoin>
  </soapenv:Body>
</soapenv:Envelope>
```

*Figur 17. Exempel på ett SOAP-meddelande (*Nordea Bank Finland Plc. *2012).*

4.4.6 ApplicationResponse-informationen

ApplicationResponse är den XML-struktur (se Fig. 18) som kunden får från banken som svar på en ApplicationRequest. ApplicationResponse-elementet finns i SOAP-meddelandets Body-element och själva Payload-informationen som kunden vill använda sig av finns i **Content**-elementet.



Figur 18. Strukturen hos ApplicationResponse-informationen

4.4.7 Överföring av SOAP-meddelanden

SOAP-meddelanden skickas till banken med HTTPS-protokollet. ApplicationRequest-informationen måste valideras mot respektive XML-schema före den skickas till banken.

Ifall det överförs stora mängder information åt gången, kan man komprimera Payload-informationen med antingen GZIP- eller PKZIP-algoritmer. Banken kräver att informationen komprimeras ifall dess storlek överskrider 50 MB. Den maximala storleken för en fil som överförs till banken är 50 MB. (Nordea Bank Finland Plc. 2012).

5 UTVECKLING AV ETT PHP-KLASSBIBLIOTEK (BANK WS)

I detta kapitel beskrivs utvecklingen av ett PHP-klassbibliotek (namngett Bank WS) för bankernas WS.

Följande verktyg har använts vid skapandet av PHP-klassbiblioteket:

- PHP 5.3.7
- Notepad++

- Nordeas tjänstebeskrivning.
- Nordeas testverktyg (XSD-filer)

Klassbiblioteket består logiskt av tre delar med egna ansvarsområden. Först skapades klasserna vilka ansvarar för att skapa korrekt formade XML-meddelanden enligt ”ISO 20022”-standarder. Den andra delen består av klasser vilka ansvarar för att ta kontakt till bankens ”Web Service”-gränssnitt, och överföra information från och till banken. Den sista delen av klassbiblioteket är ett tillägg till den existerande SoapClient-klassen i PHP. Denna del av klassbiblioteket ansvarar för att skapa ”WS-Security”-delarna av SOAP-meddelanden enligt OASIS ”WS-Security”-standarden (OASIS, 2006).

5.1 ”ISO 20022” XML-Meddelanden (Payload)

Dessa XML-meddelanden är i bankens specifikationer det data som refereras till som ***Payload.*** En statisk klass (ISO20022_Messages), fungerar som en ”factory” (O’Reilly, Head first design patterns) för att skapa olika XML-meddelanden baserade på ”ISO 20022”-standarder (ISO 20022, 2006). Varje meddelandetyp och -version representeras av en egen PHP klass. Den information som används för företagets betalningar (SEPA-banktransaktioner, löner, snabb-betalningar och valutabetalningar) refererar till XML-schemat **pain.001.001.03** och heter **CustomerCreditTransferInitiationV03.** Klassen består av ett par metoder som används för att skapa korrekt informationsstruktur. Strukturen skapas med hjälp av PHP:s DOM-bibliotek (DOM, The PHP Group, 2012c).

Strukturen hos XML-informationen **CustomerCreditTransferInitiationV03** är definierad av ”ISO 20022”-standarden (*ISO 20022,* 2009) men tillämpas av banken och därför har också Nordeas service-beskrivning av företagets betalningar använts (Nordea, 2012).

5.2 ”WS-Security”-tillägget till SoapClient-klassen

Klassen SoapClient_WSS är en underklass till PHP:s SoapClient-klass (PHP.net SoapClient). SoapClient_WSS-klassen tillägger funktionalitet för att kunna digitalt signera SOAP-meddelanden som skickas till banken.

XML digitala signaturen och ”WS-Security header”-elementet skapas enligt OASIS standarden för WS-Security (OASIS, 2006). För att underlätta skapandet av XML digitala signaturer, har det använts en tredje parts PHP-klassbibliotek: **wse-php** (wse-php, 2012).

5.3 WS-förbindelsen

Ansvaret för att skapa korrekt formade XML-informationsstrukturer (ApplicationRequest, SOAP Envelope), validera dessa strukturer och XML digitalt signera dessa strukturer ligger hos klassen ”WebService”.

”WebService”-klassen använder sig huvudsakligen av PHP:s DOM-funktionalitet för att skapa korrektformade XML-strukturer för ApplicationRequest-informationen och SOAP-meddelandet.

5.4 Klassbibliotekets logiska struktur

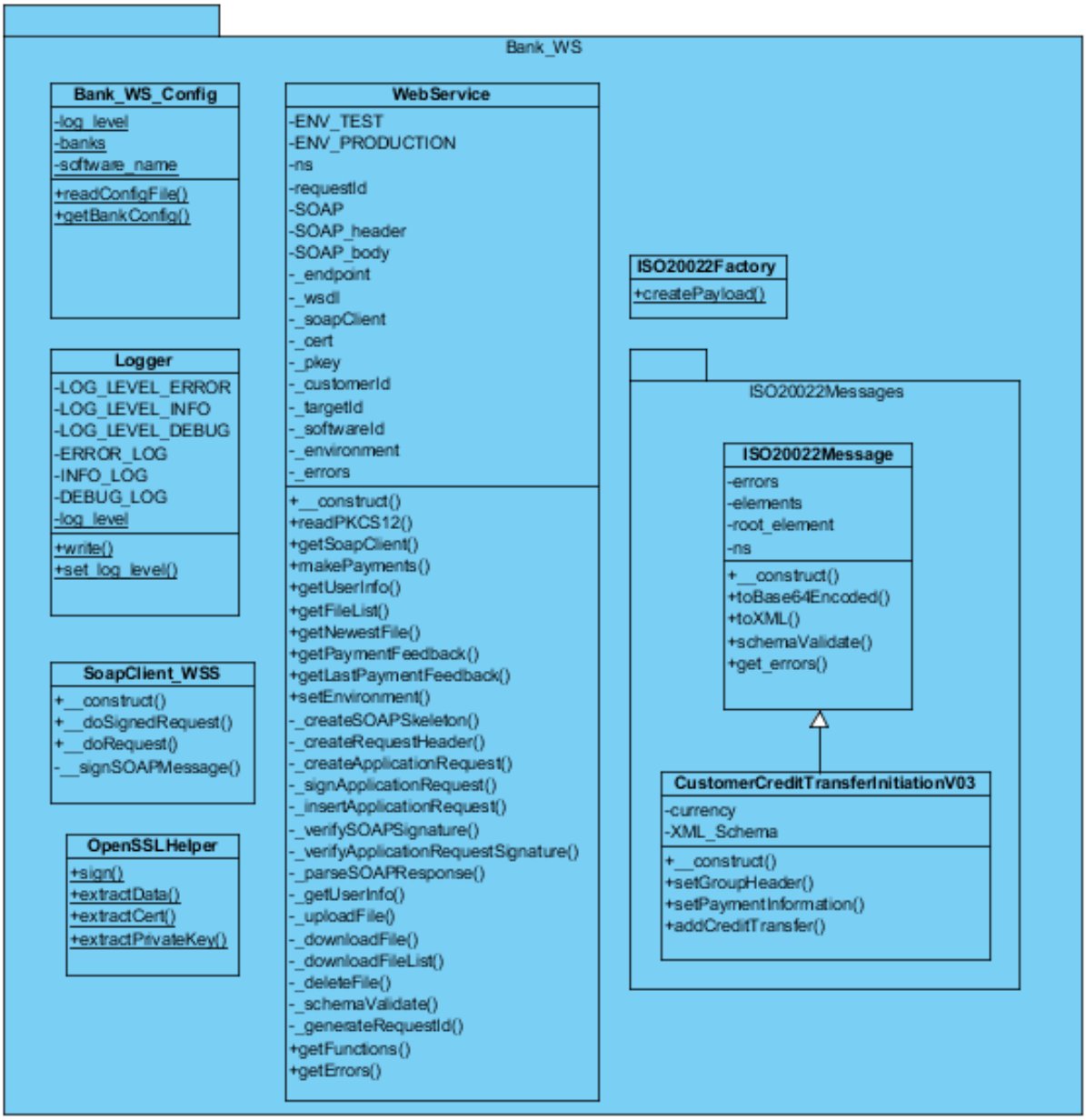
I Fig. 19 presenteras klassbibliotekets logiska struktur i ett UML-klassdiagram. Klasserna som syns i diagramet har följande funktioner:

- **ISO20022Factory**
  Skapa ”ISO 20022”-meddelanden
- **ISO20022Message**
  Huvudklassen för ”ISO 20022”-meddelanden, som realiserar metoder gemensamma för olika ”ISO 20022”-meddelanden
- **CustomerCreditTransferInitiationV03**
  Representerar en viss typs ”ISO 20022”-standardiserad informationsstruktur, i detta fall är typen ”företagets betalningar” och denna klass ansvarar för att skapa en XML-struktur enligt ISO-standarden
- **OpenSSLHelper**
  En statisk klass med hjälpfunktioner som förenklar användning av OpenSSL-funktioner såsom att läsa ut ett X.509-certifikat och den privata nyckeln från en

PKCS#12 ”certificate store”-fil som man får av Nordea efter att Nordea har certifierat nyckelparet

- **SoapClient_WSS**
  En underklass till SOAP-utvidgningens SoapClient-klass i PHP. Klassen tillägger funktionalitet för att signera SOAP-meddelanden med en transport-signatur.
- **Logger**
  En statisk klass som tillåter loggning av olika typers händelser till olika filer
- **WebService**
  Klassen som ansvarar för att definiera de olika XML-informationsstrukturer som behövs för bankernas WS (SOAP, ApplicationRequest), för signering av ApplicationRequest-informationen, för att skicka SOAP-meddelanden via SoapClient_WSS klassen, verifiera bankens signatur i ApplicationResponse-meddelandet och till sist att förvandla ApplicationResponse XML-informationen till PHP-objekt för vidare processering.

Tredje parters klasser visas ej i diagramet. Diagramet är skapat med Visual Paradigm (Visual Paradigm, 2012).



Figur 19. Klassbibliotekets logiska struktur presenterad med UML-klassdiagram

5.5 Testning av klassbiblioteket

Före XML-informationsstrukturen kan användas med bankernas WS måste den genomgå testning. Nordea ger ut XSD-filer som kan användas för att automatisk validera strukturen hos ApplicationRequest-informationen

Ifall man ännu inte har gjort ett WS-avtal med banken, kan man använda sig av ett testcertifikat för att testa förbindelsen. Testcertifikatet kan laddas ned från Nordeas webb-

sida (Nordea, 2012). Exempel på XML digitalt signerade SOAP-meddelanden samt av XML digitalt signerade ApplicationRequest-informationen kan nedladdas från Nordeas webb-sidor (Nordea 2012). Dessa kan användas för att validera strukturen av informationen som skall användas med Nordeas WS för hand. Förutom dessa så finns det inga egentliga verktyg som man kan använda sig för att testa förbindelsen.

5.6 Exempel

Alla exempel som refereras till i detta avsnitt är från en programkörning där det flyttades 0,01 euro från Innovoices bankkonto till ett annat bankkonto också ägt av Innovoice.

I Bilaga 1 visas den information som används i PHP för att utföra en banktransaktion med ”Bank WS”-klassbiblioteket. I Bilaga 2 visas exempel på en funktion som använder klassbiblioteket för att utföra en banktransaktion. Informationen som visas i Bilagorna 1 och 2 är i detta fall realiserad i Innovoices system och är exempel på den kod som behövs för att integrera klassbiblioteket i vilket system som helst.

I Bilagorna 3-5 visas de XML-informationsstrukturer som skapas av klassbiblioteket och som skickas till bankens WS-gränssnitt för att utföra SEPA-betalningar.

I Bilagorna 6-8 visas på olika sätt den information som banken skickar till kunden som svar på SOAP-meddelandet som användes för att utföra en SEPA-betalning (Bilaga 5).

6 DISKUSSION OCH SLUTSATSER

Efter att i början ha läst mycket teori angående olika möjligheter med PHP och bankernas WS, beslöt jag att bygga klassbiblioteket med hjälp av ”WSO2 Web Services Framework för PHP”-ramverket. Vid installationen av PHP-utvidgningen ”WSO2 Web Services Framework” så fick jag dock inte utvidgningen att fungera. Efter flera dagars experimentering och studier av öppna diskussioner på Internet, kom jag fram till att ”WSO2 Web Services Framework” inte fungerar tillsammans med PHP-versioner nyare än 5.4.0 (Vid skrivandet var nyaste versionen 5.4.7). Eftersom PHP går hela tiden vidare

och det utvecklas nya funktionaliteter och förbättringar, så ville jag inte ta i bruk en tredje parts komponent, vars framtid är osäker och ligger för tillfället efter i utvecklingen.

Då gjorde jag beslutet att jag tänker använda mig av PHP:s inbyggda SOAP-funktionalitet för att skapa klassbiblioteket. Den saknar stöd för att enkelt skapa ”WS-Security”-funktionalitet, vilket kräver mera arbete av programmeraren. Men enligt skribentens åsikt för tillfället, så kommer detta att leda till en bättre och mera långsiktigt fungerande lösning.

Efter att först ha prövat själv skapa med PHP:s DOM-klassbiblioket korrekt formade XML-strukturer för både ApplicationRequest-informationen och SOAP-meddelanden, men inte fått XML digitala signaturen validerad hos banken, hittades en tredje parts lösning (wse-php), som hade färdig funktionalitet för att skapa XML digitala signaturer. wse-php togs i bruk och nu skapades transportsignaturen med hjälp av wse-php, men fortfarande kunde banken inte validera signaturen. Eftersom meddelandet inte gick igenom det första säkerhetslagret hos banken (SOAP-signaturen kunde inte valideras), blev det inte heller något spår av meddelandet i bankens system för igenkänning av eventuella fel. Efter flera samtal med banken föreslog banken att jag skulle skapa ett nytt nyckelpar och en ny certifieringsbegäran, vilket jag gjorde. Efter jag hade fått ett nytt certifikat av banken började förbindelsen fungera. Nästa utmaning var att få ApplicationRequest-informationen signerad korrekt med en annan typ av XML digitala signaturer, dvs. en Enveloping Business Signature. Denna signatur gick inte heller direkt att validera hos banken av någon orsak och det kom fram att banken inte erbjuder någon service för att leta fram fel i information som skickas till banken. Det enda man kunde göra var att försöka manuellt validera ApplicationRequest-informationen mot de exempel som banken har i sin dokumentation. Jag började bygga om ApplicationRequest-strukturen från början och all kod som hade att göra med detta och till slut så fick jag signaturen validerad hos banken.

Det har varit mycket intressant att arbeta med klassbiblioteket och det har varit till stor nytta efter att det togs i bruk. Jag har lärt mig många nya saker under tiden jag har job-

bat med examensarbetet, varav man kunde speciellt lyfta fram: XML digitala signaturer, PHP:s DOM-bibliotek samt användning av OpenSSL-funktioner med PHP.

Klassbiblioteket fungerar bra för den uppgift som den var i första hand gjord för, dvs. att möjliggöra automatisering av ett företags betalningar. Klassbiblioteket möjliggör inte för tillfället användning av all funktionalitet som bankerna erbjuder via sina WS, men det är skapat för att kunna utvidgas med nya metoder, nya banker eller ny Payload-information. Troligtvis kommer det att utvecklas vidare framtiden.

Om jag fick nu välja de verktyg som skulle användas för att skapa funktionalitet för automatisering av banktransaktioner och man inte är bunden till någon viss miljö på servern, så skulle jag troligtvis välja mellan .NET eller Java p.g.a. det finns färdig funktionalitet i båda miljöerna för att arbeta med XML digitala signaturer.

KÄLLOR

European Central Bank. 2012, ECB: SEPA. Tillgänglig: http://www.ecb.int/paym/sepa/html/index.en.html Hämtad 18.10.2012

European Payments Council. 2008, Tillgänglig: http://www.europeanpaymentscouncil.eu/images/EPC-home_chart.jpg Hämtad 18.10.2012

FC. 2009, Finansbranschens Centralförbund tillsammans med Nordea, OP-Pohjola Gruppen och Sampo Banken, Security and Message Specification for Financial Messages using Web Services - Version 1.05, publicerad 22.10.2008. Tillgänglig: http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/WebServices_Messages_20081022_105.pdf Hämtad 5.8.2012

FC. 2010, Finansbranschens Centralförbund, ISO 20022 maksut, publicerad 27.12.2010. Tillgänglig: http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/ISO20022_maksut.pdf Hämtad 27.09.2012

FC. 2012, Finansbranschens Centralförbund, WSDL-fil för bankernas ”Web Services”. Tillgänglig: http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/BankCorporateFileService_20080616.wsdl Hämtad 26.7.2012

FINeID. 2010, Kansalaisvarmenne - FINEID.FI, publicerad 29.12.2010. Tillgänglig: http://fineid.fi/default.aspx?id=292 Hämtad 10.8.2012

Microsoft. 2012, Figure 2: S/MIME Message Encryption and Decryption Process. Tillgänglig: http://technet.microsoft.com/en-us/library/cc700805.aspx Hämtad 31.10.2012

MIT. 2003, MIT OpenCourseWare | Electrical Engineering and Computer Science | 6.857 Network and Computer Security, Fall 2003. Tillgänglig: http://www.ocw.nur.ac.rw/OcwWeb/Electrical-Engineering-and-Computer-Science/6-857Fall2003/CourseHome/index.htm Hämtad 30.10.2012

Mobiilivarmenne. 2010, Mobiilivarmenne lisää turvallisuutta | Mobiilivarmenne, publicerad 30.11.2010. Tillgänglig: http://www.mobiilivarmenne.fi/fi/security Hämtad 15.8.2012

NIST. 2001, National Institute of Standards and Technology, Introduction to Public Key Technology and the Federal PKI Infrastructure, publicerad 26.2.2001 Tillgänglig: http://csrc.nist.gov/publications/nistpubs/800-32/sp800-32.pdf Hämtad 20.9.2012

NIST. 2009a, National Institute of Standards and Technology, Digital Signature Standard (DSS), publicerad juni 2009. Tillgänglig: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf Hämtad 4.10.2012

Nordea. 2012, Testaus | Nordea.fi. Tillgänglig: http://www.nordea.fi/Yritykset+ja+yhteis%C3%B6t/Maksuliike/Yhteys+pankkiin/Testaus/1135012.html?lnkID=d-box_testaus_ohjeita-esimerkkiaineistoja_06-09-2012 Hämtad 20.11.2012

Nordea Bank Finland Plc. 2012, Web Services Security and Communication, description, Version: January 2012 | 44. Tillgänglig: http://www.nordea.fi/sitemod/upload/root/fi_org/liite/e/yritys/pdf/web_services_ohjelmistotalot.pdf Hämtad 15.6.2012

Nordea Bank Finland Abp. 2012. Web Services filöverföring, tjänstebeskrivning, Version:December2011|18, publicerad december 2011. Tillgänglig: http://www.nordea.fi/sitemod/upload/Root/fi_org/liite/r/yritys/pdf/web_services.pdf Hämtad 20.4.2012

Nordea. 2012, Nordea Bank Finland Abp, Yrityksen maksut –palvelu. Palvelukuvauksen esimerkkiliite Version:November2012|18. Tillgänglig: https://www.nordea.fi/sitemod/upload/root/content/nordea_fi_fi/yritysasiakkaat/laskutus_maksaminen/palvelukuvaukset/yrityksen_maksut_esimerkkiliite3.pdf Hämtad 24.10.2012

NuSOAP. 2002, NuSOAP- SOAP Toolkit for PHP, publicerad 11.6.2002. Tillgänglig: http://nusoap.sourceforge.net/ Hämtad 10.10.2012

ISO 20022. 2012, Universal financial industry message scheme. Tillgänglig: http://www.iso20022.org/about_iso20022.page Hämtad 27.9.2012

ISO 20022, Universal financial industry message scheme: Full catalogue of ISO 20022 messages. Tillgänglig: http://www.iso20022.org/full_catalogue.page Hämtad 1.8.2012

ISO 20022. 2009, Payments - Maintenance 2009 Message Definition Report s. 710-766. Tillgänglig: http://www.iso20022.org/message_archive.page Hämtad 20.9.2012

OASIS. 2006, Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), publicerad 1.2.2006, Tillgänglig: https://www.oasis-open.org/committees/download.php/16790/wss-1.1-spec-os-SOAPMessageSecurity.pdf Hämtad 29.9.2012

RFC 2459, IETF. 1999, Internet X.509 Public Key Infrastructure Certificate and CRL Profile. publicerad januari 1999. Tillgänglig http://www.ietf.org/rfc/rfc2459.txt Hämtad 27.9.2012

RFC 2631, IETF. 1999, Diffie-Hellman Key Agreement Method, publicerad juni 1999. Tillgänglig: http://www.ietf.org/rfc/rfc2631.txt Hämtad 5.10.2012

RFC 3851, IETF. 2004, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1, publicerad juli 2004, Tillgänglig: http://www.ietf.org/rfc/rfc3851.txt Hämtad 21.8.2012

RSA Laboratories. 1993a, PKCS #7: Cryptographic Message Syntax Standard, publicerad
1.11.1993 Tillgänglig: http://www.rsa.com/rsalabs/node.asp?id=2129 Hämtad 27.9.2012

RSA Laboratories. 1993b, PKCS #3: Diffie-Hellman Key Agreement Standard, publicerad
1.11.1993 Tillgänglig: http://www.rsa.com/rsalabs/node.asp?id=2126 Hämtad 27.9.2012

RSA Laboratories. 1999, PKCS #12: Personal Information Exchange Syntax Standard, publicerad
24.6.1999. Tillgänglig: http://www.rsa.com/rsalabs/node.asp?id=2138 Hämtad 27.9.2012

RSA Laboratories. 2002, PKCS #1: RSA Cryptography Standard, publicerad
14.6.2002. Tillgänglig: http://www.rsa.com/rsalabs/node.asp?id=2125 Hämtad 27.9.2012

RSA Laboratories. 2004, PKCS #11: Cryptographic Token Interface Standard, publicerad 28.6.2004. Tillgänglig: http://www.rsa.com/rsalabs/node.asp?id=2133 Hämtad 27.9.2012

RSA Laboratories. 2012, RSA Laboratories Home: RSA Security Research Center. Tillgänglig: http://www.rsa.com/rsalabs/ Hämtad 28.11.2012

Stallings, William. 2006, *Cryptography and Network Security,* 4 uppl., Upper Saddle River, NJ 07458: Pearson Education , Inc. 679 s.

The PHP Group. 2005, PHP: What is PHP? Tillgänglig: http://www.php.net/manual/en/intro-whatis.php Hämtad 20.7.2012

The PHP Group. 2012b, PHP: Sockets - Manual, Tillgänglig: http://www.php.net/manual/en/book.sockets.php Hämtad 31.10.2012

The PHP Group. 2012a, PHP: SOAP - Manual, Tillgänglig: http://www.php.net/manual/en/book.soap.php Hämtad 28.10.2012

The PHP Group. 2012c, PHP: DOM – Manual. Tillgänglig: http://www.php.net/manual/en/book.dom.php Hämtad 20.11.2012

University of Chicago. 2004, University of Chicago, Borja Sotomayor: The Globus Toolkit 4 Programmer's Tutorial. Tillgänglig http://gdp.globus.org/gt4-tutorial/multiplehtml/ch09s03.html Hämtad 30.10.2012

Visual Paradigm. 2012, UML, BPMN and Database Tool for Software Development. Tillgänglig: http://www.visual-paradigm.com/ Hämtad 23.11.2012

W3C. 2000, Simple Object Access Protocol (SOAP) 1.1, publicerad 8.5.2000, Tillgänglig: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486 Hämtad 15.7.2012

W3C. 2004a, Web Services Architecture, publicerad 11.2.2004, Tillgänglig: http://www.w3.org/TR/ws-arch/ Hämtad 24.7.2012

W3C. 2004b,. XML SCHEMA PART 0: PRIMER SECOND EDITION, publicerad 28.10.2004. Tillgänglig: http://www.w3.org/TR/xmlschema-0/ Hämtad 22.10.2012

W3C. 2007a, SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)**,** publicerad 27.04.2007. Tillgänglig: http://www.w3.org/TR/soap12-part1/ Hämtad 15.9.2012

W3C. 2007b, Web Services Description Language (WSDL) Version 2.0 Part 1:Core Language, publicerad 26.6.2007. Tillgänglig: http://www.w3.org/TR/2007/REC-wsdl20-20070626/#intro_ws Hämtad 24.7.2012.

W3C. 2008, XML Signature Syntax and Processing (Second Edition), publicerad 10.6.2008. Tillgänglig: http://www.w3.org/TR/xmldsig-core/ Hämtad 15.10.2012

W3C. 2010, XML Essentials, publicerad 2010. Tillgänglig: http://www.w3.org/standards/xml/core Hämtad 26.7.2012.

Wikipedia. 2007, Public-key infrastructure. Tillgänglig: http://en.wikipedia.org/wiki/Public-key_infrastructure Hämtad 30.10.2012

*wse-php,*2012. WS-* functionality for PHP. Tillgänglig: http://code.google.com/p/wse-php/ Hämtad 15.11.2102

WSO2. 2011, WSO2 Web Services Framwork for PHP. Tillgänglig: http://wso2.org/project/wsf/php/2.0.0/docs/ Hämtad 20.9.2012

BILAGOR

Bilaga 1. Exempel på en datastruktur för att skapa Payload-information med hjälp av ”Bank WS”-klassbiblioteket för en SEPA-betalning som utförs via bankernas WS

Denna datastruktur innehåller alla de parametrar som behövs för att kunna skapa en SEPA-betalnings Payload-information med ”Bank WS”-klassbiblioteket.

```
/*** Structure of moneyTransfer param ***/
    $moneyTransfer = array(
        'id' => 'mntId-1',
        'serviceId' => '023333983',
        'controlSum'    => 0.01,
        'numTransactions'   => 1,
        'paymentInfo' => array(
            'id' => 'pmtId-1',
            'paymentMethod' => 'TRF',
            'batchBooking' => true,
            'requestedExecutionDate' => 'today',
            'debtorName' => 'Innovoice Oy',
            'debtorIBAN' => 'FI7817453000138331',
            'debtorBIC' => 'NDEAFIHH'
        ),
        'creditTransfers' => array(
            array(
                'id' => 'ctrId-1',
                'amount' => 0.01,
                'creditorName' => 'Innovoice Oy',
                'creditorCountry' => '',
                'creditorAddress' => '',
                'creditorZipcode' => '',
                'creditorCity' => '',
                'creditorIBAN' => 'FI0317453000138323',
                'creditorBIC' => 'NDEAFIHH',
                'refNo' => '',
                'message' => 'Web Service TEST'
            ),
            // array(...) Another payment
            // array(...) One more payment
    );
```

Bilaga 2. Exempel på användning av Bankernas WS med klassbiblioteket

Denna funktion är ett exempel på hur ”Bank WS”-klassbiblioteket integreras in i ett annat system och används för att utföra en SEPA-betalning. Denna kodsnutt tillhör inte ”Bank WS”-klassbiblioteket, men demonstrerar användning av klassbiblioteket.

```
function make_payments($moneyTransfer){

//Include the Bank_WS-library
require_once(TOOLS . 'bank_ws/bank_ws.php');

//Get configurations for the profile "nordea", defined in a separate
config.php file
        $bankConfig = Bank_WS_Config::getBankConfig('nordea');

//Initialize the webservice with the bank-specific configurations
        $nordea_ws = new WebService($bankConfig);

//Read in the certificate store file so we can sign the needed parts of the
message
        $nordea_ws->readPKCS12(BANK_WS_CERTS . 'NordeaEid_3817397185.p12',
'***PASSWORD***');

//Create the Payload
        $payload =
ISO20022Factory::createPayload('CustomerCreditTransferInitiationV03');

//Set some header information for the payments in this batch
        $payload->setGroupHeader($moneyTransfer['id'],
$moneyTransfer['serviceId'], $moneyTransfer['controlSum'],
$moneyTransfer['numTransactions']);

$paymentInfo =& $moneyTransfer['paymentInfo'];

//Set payment debtor (Account from where the payments will be made)
        $payload->setPaymentInformation(
            $moneyTransfer['id'], $paymentInfo['id'],
$paymentInfo['paymentMethod'], $paymentInfo['batchBooking'],
            $paymentInfo['requestedExecutionDate'],
$paymentInfo['debtorName'], $paymentInfo['debtorIBAN'],
$paymentInfo['debtorBIC']
        );

//For each payment in this batch add a CreditTransfer-item to the payload
        foreach($moneyTransfer['creditTransfers'] as $creditTransfer){
            $payload->addCreditTransfer(
                $creditTransfer['id'], $creditTransfer['amount'],
$creditTransfer['creditorName'],
                $creditTransfer['creditorCountry'],
$creditTransfer['creditorAddress'], $creditTransfer['creditorZipcode'],
$creditTransfer['creditorCity'],
                $creditTransfer['creditorIBAN'],
$creditTransfer['creditorBIC'], $creditTransfer['refNo'],
$creditTransfer['message']
            );
        }

    //Create the correct informationstructures and send it to thebank
    $applicationResponse = $nordea_ws->makePayments($payload);

    //show the applicationResponse information
    var_dump($applicationResponse);
}
```

Bilaga 3. Exempel på Payload-informationen

Denna XML-information är ett exempel på Payload-informationen som beskriver detaljerna av en eller flera SEPA-betalningar som skickas till banken via bankernas WS som Base64-kodad information i Content-elementet av ApplicationRequest-informationen – Se Bilaga 4.

```
<?xml version="1.0" encoding="utf-8"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03
pain.001.001.03.xsd">
<CstmrCdtTrfInitn>
    <GrpHdr>
        <MsgId>1354289696</MsgId>
        <CreDtTm>2012-11-30T17:34:56+02:00</CreDtTm>
        <NbOfTxs>1</NbOfTxs>
        <CtrlSum>0.01</CtrlSum>
        <InitgPty>
            <Nm>Innovoice</Nm>
            <Id><OrgId><Othr>
                 <Id>023333983</Id>
                 <SchmeNm><Cd>BANK</Cd></SchmeNm>
            </Othr></OrgId></Id>
        </InitgPty>
    </GrpHdr>
    <PmtInf>
        <PmtInfId>pmt-1354289696</PmtInfId>
        <PmtMtd>TRF</PmtMtd>
        <BtchBookg>1</BtchBookg>
        <ReqdExctnDt>2012-11-30</ReqdExctnDt>
        <Dbtr>
            <Nm>Innovoice Oy</Nm>
            <Id>
                <OrgId><Othr><Id>023333983</Id>
                        <SchmeNm><Cd>BANK</Cd></SchmeNm>
                    </Othr></OrgId>
            </Id>
        </Dbtr>
        <DbtrAcct>
            <Id><IBAN>FI7817453000138331</IBAN></Id>
        </DbtrAcct>
        <DbtrAgt><FinInstnId><BIC>NDEAFIHH</BIC></FinInstnId></DbtrAgt>
        <CdtTrfTxInf>
            <PmtId><EndToEndId>transfer-1-1354289696</EndToEndId></PmtId>
            <Amt><InstdAmt Ccy="EUR">0.01</InstdAmt></Amt>
            <Cdtr><Nm>Innovoice Oy</Nm></Cdtr>
            <CdtrAcct>
                <Id><IBAN>FI0317453000138323</IBAN></Id>
            </CdtrAcct>
            <RmtInf>
                <Ustrd>Web Service TEST</Ustrd>
            </RmtInf>
        </CdtTrfTxInf>
    </PmtInf>
</CstmrCdtTrfInitn>
</Document>
```

Bilaga 4. Exempel på XML digitalt signerad ApplicationRequest-information

*Denna XML-information är ett exempel på XML digitalt signerad ApplicationRequest-information. Informationen beskriver den funktionen som anropas via bankernas WS och **Signature**-delen autentiserar kunden för banken.*

```
<?xml version="1.0"?>
<bxd:ApplicationRequest xmlns:bxd="http://bxd.fi/xmldata/">
<bxd:CustomerId>********</bxd:CustomerId>
<bxd:Command>UploadFile</bxd:Command>
<bxd:Timestamp>2012-11-30T19:34:56+02:00</bxd:Timestamp>
<bxd:Environment>PRODUCTION</bxd:Environment>
<bxd:TargetId>********</bxd:TargetId>
<bxd:SoftwareId>Innovoice Online</bxd:SoftwareId>
<bxd:FileType>NDCORPAYS</bxd:FileType>
<bxd:Content>PD94Eb2...(Resten av den Base64-kodade Payload-informationen är
rensad)</bxd:Content>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="">
        <ds:Transforms>
            <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>de40RMi36i9fly2N1VmwoPHggDg=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>ELEFdDb34Fz... (Resten av den Base64-kodade digitala
signaturen är rensad)</ds:SignatureValue>
<ds:KeyInfo>
    <ds:X509Data>
        <ds:X509Certificate>MIIDtjCCA... (Resten av den Base64-kodade
certifikatet är rensad)</ds:X509Certificate>
        <ds:X509IssuerSerial>
            <ds:X509IssuerName>C=SE,O=Nordea Bank AB (publ),CN=Nordea Corporate
CA 01,serialNumber=516406-0120</ds:X509IssuerName>
            <ds:X509SerialNumber>23345410</ds:X509SerialNumber>
        </ds:X509IssuerSerial>
    </ds:X509Data>
</ds:KeyInfo>
</ds:Signature>
</bxd:ApplicationRequest>
```

Bilaga 5. Exempel på ett XML digitalt signerat SOAP-meddelande

Detta är ett exempel på den XML-information som skickas till bankens WS.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cor="http://bxd.fi/CorporateFileService"
xmlns:mod="http://model.bxd.fi">
<soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">
    <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" wsu:Id="pfxbb750797-3355-0b29-8b04-
4f1581c0a792" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3">MIIDtjCC... (Resten av den Base64-kodade
certifikatet är rensad)</wsse:BinarySecurityToken>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/><ds:Reference URI="#pfxd7665340-54f7-b51c-b4bb-d133ee1e11ff">
            <ds:Transforms>
                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/></ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>Yl8gYs3J4RNRGW8gSRZaq7hd6PU=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>oKlr+W6wnXKY9g...(Resten av den Base64-kodade Digitala
signaturen är rensad)</ds:SignatureValue>
    <ds:KeyInfo>
        <wsse:SecurityTokenReference>
            <wsse:Reference URI="#pfxbb750797-3355-0b29-8b04-4f1581c0a792"/>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo></ds:Signature></wsse:Security>
</soap:Header>
<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="pfxd7665340-54f7-b51c-b4bb-d133ee1e11ff">
<cor:uploadFilein>
    <mod:RequestHeader xmlns:mod="http://model.bxd.fi">
    <mod:SenderId>********</mod:SenderId>
    <mod:RequestId>makePayments-b31a0098501e2ab6d</mod:RequestId>
    <mod:Timestamp>2012-11-30T20:34:56+02:00</mod:Timestamp>
    <mod:Language>FI</mod:Language>
    <mod:UserAgent>Innovoice Online</mod:UserAgent>
    <mod:ReceiverId>NDEAFIHH</mod:ReceiverId>
</mod:RequestHeader>
<ApplicationRequest xmlns="http://model.bxd.fi">PGJ4ZDpBcHBsa...(Resten av den
Base64-kodade ApplicationRequest-informationen är rensad)</ApplicationRequest>
</cor:uploadFilein>
</soap:Body>
</soap:Envelope>
```

Bilaga 6. Exempel på ett SOAP-meddelande som banken svarar med på en betalning gjord via bankernas WS

Denna XML-information är det som banken returnerar kunden som svar på ett funktionsanrop från bankernas WS.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:cor="http://bxd.fi/CorporateFileService" xmlns:mod="http://model.bxd.fi">
<soapenv:Header><wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">
    <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-ba495bf5-f61c-456f-9424-902ed25a2fb9">
    <wsu:Created>2012-11-30T15:34:57Z</wsu:Created>
    <wsu:Expires>2012-11-30T15:39:57Z</wsu:Expires>
</wsu:Timestamp>
<wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-ff580567-5cf8-444e-a766-
8bd99c646316" 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">MIID8TCCA...
(Resten av den Base64-kodade certifikatet är rensad)</wsse:BinarySecurityToken>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI="#Timestamp-ba495bf5-f61c-456f-9424-902ed25a2fb9">
        <Transforms>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>9MJ404NsItUJxmpirjyDw+qvxxo=</DigestValue>
    </Reference>
    <Reference URI="#Body-ea965420-6b57-43e8-a031-de59a015c4c7">
        <Transforms><Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>NwZJ9bNF+YqbDvF6sltVI/MZzCw=</DigestValue>
    </Reference>
</SignedInfo>
<SignatureValue>Nnk5tpO... (Resten av den Base64-kodade digitala signaturen är
rensad)</SignatureValue><KeyInfo><wsse:SecurityTokenReference xmlns="">
        <wsse:Reference URI="#SecurityToken-ff580567-5cf8-444e-a766-8bd99c646316"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3"/></wsse:SecurityTokenReference></KeyInfo></Signature></wsse:Security>
</soapenv:Header>
<soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="Body-ea965420-6b57-43e8-a031-de59a015c4c7">
<cor:uploadFileout>
    <mod:ResponseHeader>
        <mod:SenderId>********</mod:SenderId>
        <mod:RequestId>makePayments-b31a0098501e2ab6d</mod:RequestId>
        <mod:Timestamp>2012-11-30T16:34:57+01:00</mod:Timestamp>
        <mod:ResponseCode>00</mod:ResponseCode>
        <mod:ResponseText>OK.</mod:ResponseText>
        <mod:ReceiverId>NDEAFIHH</mod:ReceiverId>
    </mod:ResponseHeader>
    <mod:ApplicationResponse>PGMyYjpBcH...(Resten av den Base64-kodade
ApplicationRequest-informationen är rensad)</mod:ApplicationResponse>
</cor:uploadFileout>
</soapenv:Body>
</soapenv:Envelope>
```

Bilaga 7. Exempel på ApplicationResponse-informationen som innehåller data om en gjord betalning

Denna XML-information får man fram efter man har Base64-avkodat ApplicationResponse-elementet från den XML-information som visas i Bilaga 6.

```
<?xml version="1.0"?>
<c2b:ApplicationResponse xmlns:c2b="http://bxd.fi/xmldata/">
<c2b:CustomerId>********</c2b:CustomerId>
<c2b:Timestamp>2012-11-30T17:34:57.44+02:00</c2b:Timestamp>
<c2b:ResponseCode>00</c2b:ResponseCode>
<c2b:ResponseText>OK.</c2b:ResponseText>
<c2b:Encrypted>false</c2b:Encrypted>
<c2b:Compressed>false</c2b:Compressed>
<c2b:AmountTotal>0.01</c2b:AmountTotal>
<c2b:TransactionCount>1</c2b:TransactionCount>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
    <Reference URI="">
        <Transforms>
            <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
        </Transforms>
        <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>T6ziCXNAq2pMMRlV1P2DIv+FvkM=</DigestValue>
    </Reference>
</SignedInfo>
<SignatureValue>LDFlDe8L ... (Resten av den Base64-kodade digitala
signaturen är rensad)</SignatureValue>
<KeyInfo>
    <X509Data>
        <X509Certificate>MIID8TCCAtmgA ... (Resten av den Base64-
kodade certifikatet är rensad)</X509Certificate>
        <X509IssuerSerial>
            <X509IssuerName>serialNumber=516406-0120, CN=Nordea
Corporate Server CA 01, O=Nordea Bank AB (publ), C=SE</X509IssuerName>
            <X509SerialNumber>22641694</X509SerialNumber>
        </X509IssuerSerial>
    </X509Data>
</KeyInfo>
</Signature>
</c2b:ApplicationResponse>
```

Bilaga 8. Exempel på ett PHP-objekt skapat från ett SOAP-svarsmeddelande

Exempel på ett PHP-objekt som klassbiblioteket har skapat från den information som banken har skickat till kunden som svar på en SEPA-betalning – Se Bilaga 5. datastrukturen visas med PHP:s var_dump-funktion

```
object(stdClass)#40 (2) {
  ["ResponseHeader"]=>
  object(stdClass)#39 (6) {
    ["SenderId"]=>
    string(10) "********"
    ["RequestId"]=>
    string(30) "makePayments-b31a0098501e2ab6d"
    ["Timestamp"]=>
    string(25) "2012-11-30T16:34:57+01:00"
    ["ResponseCode"]=>
    string(2) "00"
    ["ResponseText"]=>
    string(3) "OK."
    ["ReceiverId"]=>
    string(8) "NDEAFIHH"
  }
  ["ApplicationResponse"]=>
  object(stdClass)#27 (8) {
    ["CustomerId"]=>
    string(10) "********"
    ["Timestamp"]=>
    string(28) "2012-11-30T17:34:57.44+02:00"
    ["ResponseCode"]=>
    string(2) "00"
    ["ResponseText"]=>
    string(3) "OK."
    ["Encrypted"]=>
    string(5) "false"
    ["Compressed"]=>
    string(5) "false"
    ["AmountTotal"]=>
    string(4) "0.01"
    ["TransactionCount"]=>
    string(1) "1"
  }
}
```