Petri Hirvi

Web services -yhteyskäytäntö sähköisten kuittitietojen välityksessä

Tietotekniikan
pro gradu -tutkielma
18. marraskuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Petri Hirvi
Yhteystiedot: petri.j.hirvi@student.jyu.fi
Puhelinnumero: +358 400 602 998
Ohjaaja: Ismo Hakala ja Risto Honkanen
Työn nimi: Web services -yhteyskäytäntö sähköisten kuittitietojen välityksessä
Title in English: Web services -protocol for the transmission of electronic receipt
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 56+8
Tiivistelmä: Pro gradu -tutkielman teoriaosuudessa tarkastellaan web services -palvelun toteutusta pankkiyhteysohjelmiston kehittäjän kannalta. Tutkielmassa tutustutaan web services -standardeihin ja käsitellään XML-dokumenttien prosessointia pankkien palveluissa. Pro gradu -tutkielman empiirisessä osuudessa kuvataan web services -palvelun käyttöä asiakkaan ja pankin välisessä sähköisten kuittitietojen välityksessä. Siirrettävät aineistot salataan SSL-protokollan avulla ja asiakkaan tunnistaminen tehdään pankin julkaisemalla Public Key Infrastructure -varmenteella (PKI). Web services -yhteyskäytäntöä käytetään myös muiden aineistojen käsittelyyn kuten viitemaksujen ja konekielisten tiliotteiden välityksessä. Sähköisten kuittitietojen käsittelyn muodostamassa ekosysteemissä yhdistyvät siihen liittyvät teknologia ja sopimukset sekä tuotteet ja palvelut. Kaikki nämä yhdessä luovat infrastruktuurin eli alustan, jonka avulla yritykset voivat tarjota uusia palveluita käyttäjilleen ja asiakkailleen.
Avainsanat: Web services, SOAP, XML, WSDL, SSL, REST, sähköinen kuitti
Abstract: The theoretical part of the master's thesis examines the implementation of the web services -solution from the perspective of a banking software developer. The dissertation introduces web services -standards and deals with the processing of XML documents in banking services. The empirical part of the master's thesis presents the web services -protocol in the transmission of electronic receipt information between the customer and the bank. The files to be transferred are encrypted using the SSL protocol and the customer is identified by a Public Key Infrastructure (PKI) certificate issued by the bank. The web services -protocol is also used to process other materials, such as the transmission of reference payments and machine-language account statements. Electronic receipt data processing ecosystem combines related technologies and contracts with products and services. Together, these create the infrastructure, the platform, for companies to offer new services to their users and customers.

Keywords: Web services, SOAP, XML, WSDL, SSL, REST, electronic receipt

Sanasto

B2BAI	Sovellusliittymät yhdistävät yritysten ohjelmistot palveluina B2B-protokolliin (Business to Business Application Integration)
CA	Varmenteen eli sertifikaatin myöntäjä (Certificate Authority)
CPS	Varmennekäytäntö lausuma, joka määrittää käytänteet sertifikaattien myöntämiseen ja hallinnointiin (Certificate Practice Statement)
CRL	Sertifiointielimen ylläpitämä lista suljetuista varmenteista (Certificate Revocation List)
EAI	Integraatiossa yrityssovellukset keskustelevat keskenään (Enterprise Application Integration)
Ekosysteemi	Itsestään järjestäytyvä tekninen infrastruktuuri
eOsoite	Yrityksien itse hallinnoima verkko-osoite tuote-, sopimus- tai kuittitietojen välitykseen
eTasku	Mobiilisovellus kuittien hallintaan
Finvoice	Pankkien kehittämä verkkolaskuformaatti
GDPR	Yleinen tietosuoja-asetus (General Data Protection Regulation)
HTTP	Selaimien ja www-palvelimien käyttämä tiedonsiirtoprotokolla (Hypertext Transfer Protocol)
HTTPS	HTTP-protokollan salattu versio (Hypertext Transfer Protocol Secure)
Infrastruktuuri	Tuotteiden, palveluiden ja teknologian muodostama alusta
Issuer	Kyseisen palvelun liikkeellelaskija

PKI	Public Key Infrastructure. Kansainvälinen määritys yhteyden osapuolen tunnistamiseksi
RA	Rekisteröintielin, joka vastaa käyttäjien ja sertifiointielimen vuorovaikutuksesta (Registration Authority)
REST	HTTP-protokollaan perustuva arkkitehtuurimalli (Representational State Transfer)
RPC	Tietoliikenneprotokolla korkean tason käyttöjärjestelmätoimintoihin (Remote Procedure Call)
SOAP	Standardoitu web services -yhteyden sanomamuoto
SSL	Secure Socets Layer. Internet-yhteyksissä käytetty salaustekniikka
UDDI	Alustariippumaton ja avoin palvelurekisteristandardi (Universal Description, Discovery and Integration)
URI	Sisältää tietoverkossa sijaitsevan tiedon sijainnin ja nimen (Uniform Resource Identifier)
URL	Yksilöi internetissä olevan resurssin sijainnin sekä sen käyttöön tarvittavan yhteyskäytännön (Uniform Resource Locator)
URN	Yksilöi internetissä olevan resurssin URN-skeeman avulla ottamatta kantaa sen saavutettavuuteen (Uniform Resource Name Identifier)
WSDL	XML-pohjainen kieli web-teknologioihin perustuvien palvelujen kuvaamiseen (Web Services Description Language)
XML	Rakenteellinen kuvauskieli tiedonvälityksessä (Extensible Markup Language)
XSD	Teknologia XML-dokumenttien rakenteiden kuvaamiseen (XML Schema Definition)

Sisältö

1 Johdanto

Tässä tutkielmassa esitellään web services -teknologiaa ja sen soveltamista web services -palvelun toteutuksessa. Palvelun toteuttamista varten tutkitaan XML-tekniikoita ja XML-dokumenttien digitaalisia allekirjoituksia. Web services -palvelun osalta tutkitaan erityisesti SOAP-tekniikkaa, mihin tutkielman empiirisessä osuudessa kuvatun palvelun toteutus perustuu. Empiirisessä osuudessa selvitetään myös, miten web services -palvelulla muodostetaan istunto yrityksen ja pankin välisessä sähköisten kuittitietojen välityksessä.

Sähköisten kuittitietojen välitys muodostaa osan eKuitti-ekosysteemiä, joka tarjoaa sääntöjä, käytäntöjä ja standardeja, joita ekosysteemissä toimivien organisaatioiden on noudatettava. Teknologiateollisuuden [41] mukaan sähköisten kuittitietojen hyödyntäminen muodostaa yhteisen perustan, jonka avulla yritykset voivat tarjota uusia ja innovatiivisia palveluita käyttäjilleen ja asiakkailleen.

1.1 Tutkielman taustaa

Digitalisaatio ja sähköisen kaupankäynnin kasvu yhdessä globaalin kilpailun kanssa ovat tehneet menestyksekkäästä liiketoiminnasta vaikeampaa kuin koskaan ennen. Beheshtin [2] mukaan voidakseen menestyä ja olla kilpailukykyinen, yrityksen on pystyttävä vähentämään tuotantokustannuksia, tehostettava liiketoimintaprosesseja ja pystyttävä tarjoamaan tuotevalikoimaa, joka vastaa asiakkaiden tarpeita ja odotuksia. Tietotekniikalla on tärkeä rooli globaalisti toimivien yrityksien kilpailustrategiassa. Beheshti [2] esittää, että strateginen painopiste on vahvasti siirtynyt liiketoimintayksiköiden toimintojen integrointiin tietotekniikan avulla, jolloin organisaatiolla olisi käytössä jo varhaisessa vaiheessa enemmän tietoa päätöksenteon pohjalle.

Sähköisen laskutuksen ja kuittitietojen käytön kasvu on hyvä esimerkki tietotekniikan hyödyntämisestä, jolla on merkittäviä vaikutuksia organisaatiossa. Penttisen [34] mielestä hyvä esimerkki tietotekniikan suorasta vaikutuksesta on sähköisen laskutuksen käytön kasvu, joka mahdollistaa yksittäiselle yritykselle laajan kumppaniverkoston maksuliikenteen hoitamiseen organisaatioiden välillä. Penttinen [34] kui-

tenkin korostaa, että avoimien standardien leviäminen luonnollisesti edistää sähköisen laskutuksen skaalautuvuutta, toisaalta suurilla yrityksillä on pitkät perinteet pakottaa pienemmät kumppaninsa sähköiseen point-to-point kaupankäyntiin. Haagin ja muiden [18] mukaan siirtyminen avoimiin standardeihin on kuitenkin mahdollistanut yrityksille sähköisten aineistojen hyödyntämisen liiketoiminnassaan.

1.2 Tutkimusongelmat

Pro gradu -tutkielmassa käsitellään yrityksen ja pankin välistä web services -yhteyskäytäntöä, sekä sen hyödyntämistä sähköisten kuittitietojen käsittelyssä yrityksen kululaskujärjestelmässä. Tutkielman teoriaosassa selvitetään web services -tekniikoiden lisäksi XML-allekirjoitusten ja -salauksien käyttöä aineistojen välityksessä eri osapuolten välillä. Tutkielman empiirisessä osassa kuvataan miten web services -palvelua käytetään sähköisten kuittitietojen käsittelyssä yrityksen ja pankin tietojärjestelmissä.

Pro gradu -tutkielman tutkimuskysymyksinä esitetään:

- Miten web services -palvelu toteutetaan sähköisen kuittiaineiston välityksessä?
- Miten XML-allekirjoitusta ja XML-salausta käytetään aineiston salauksessa?
- Miten pankin ja pankkiyhteysohjelmiston välinen istunto muodostetaan?

1.3 Tutkielman rakenne

Tämä pro gradu -tutkielma on jaettu viiteen käsittelylukuun. Tutkielman luvussa 2 tarkastellaan web services -yhteyskäytäntöä palvelukeskeisten arkkitehtuurien käytäntönä ja sen soveltuvuutta hajautettujen järjestelmien kehittämisessä sekä ohjelmistojen välisissä rajapinnoissa. Web services -teknologioista esitellään REST- ja SOAP-pohjaisten ratkaisujen periaatteet, sekä niiden käyttöä aineiston välityksessä pankkiyhteysohjelman kannalta.

Luvussa 3 tarkastellaan XML-merkintäkieltä, johon web services -viestit ja SOAP-pohjaisissa palveluissa käytetty WSDL-dokumentti perustuvat. Luvussa 4 tarkastellaan salausprotokollien sekä XML-salauksien ja XML-allekirjoitusten käyttöä web services -palveluissa. Koska palveluilla on useita eri osapuolia, täytyy tietoturva

ylettyä palveluiden viestitasolle asti, jotta saavutetaan tarvittava datan eheys ja kiistämättömyys. Luvussa 5 esitellään eKuitin ekosysteemiä ja miten sen tuottamaa sähköistä kuittitietoa, strukturoidussa muodossa, hyödynnetään rivitasolla yrityksen kululaskujärjestelmässä.

Luvussa 6 esitellään web services -palvelun toteutusta yrityksen ja pankin välillä. Luvussa kuvataan käyttötapauksina sähköisen kuittiaineiston käsittely pankin tietojärjestelmästä yrityksen kululaskujärjestelmään. Tutkielman viimeinen luku sisältää tutkielman yhteenvedon.

2 Web services -yhteyskäytäntö

Web services -yhteyskäytäntö tunnetaan palvelukeskeisten arkkitehtuurien käytäntönä, joka mahdollistaa viestinnän ohjelmistokomponenttien tai sovellusten välillä internetissä [11]. Koska eri ohjelmistoja ja sovelluksia toteutetaan eri ohjelmointikielillä, laitteistoille ja alustoille, varsinaisten verkkopalvelujen määritykset eivät ole web services -palvelusta riippuvaisia. Tämä mahdollistaa standardin ja universaalin tiedonvälitysmenetelmän eri verkkopalvelujen välillä [11]. Web services helpottaa hajautettujen sovellusten kehittämistä ja eri ohjelmistojen välistä nopeaa integrointia [27]. Tässä luvussa esitellään web services -teknologian ratkaisuja sekä tarkastellaan REST- ja SOAP-pohjaisten palveluiden eroja ja soveltuvuutta pankkiyhteysohjelmiston kehityksessä. Lisäksi esitellään web services -palvelujen arkkitehtuuria ja toteutuksen eri vaiheita sekä mitä etuja ja haittoja web services -yhteyskäytäntö voi sisältää.

Web services toimii väliohjelmistoratkaisuna, missä määritellään tietoliikenneprotokollat ja säännöt datan välitystä varten [11]. Väliohjelmistot toimivat abstraktiokerroksena, joka toimii käyttöliittymänä sovelluksen eri komponenttien viestinnälle ja datan välitykselle [3]. RPC-protokolla (Remote Procedure Call) eli etäkäsittelykutsut otettiin käyttöön 1980-luvulla ja niiden avulla toteutettiin ensimmäiset etätoiminnot hajautetuissa järjestelmissä. RPC-protokolla toimii kerroksena, joka tarjoaa mekanismin hajautettujen proseduurikutsujen toteuttamiseen ottamatta kantaa käytettyyn alustaan tai ohjelmointikieleen [5].

Web services -palvelun tarjoaja ja käyttäjä kommunikoivat keskenään erilaisten XML-pohjaisten protokollien avulla. Protokollat sisältävät kolme erillistä komponenttia, jotka ovat SOAP (Simple Object Acces Protocol), WSDL (Web Services Description Language) ja UDDI (Universal Description, Discovery and Integration). Tässä tutkielmassa ei esitellä UDDI-rekisterin kautta käytettyä web services -palvelua, koska sellaista ei ole julkaistu pankkiyhteyskäyttöön [15].

Web services -teknologioilla toteutetut palvelut voidaan jakaa kahteen eri ryhmään, REST- ja SOAP-pohjaisiin palveluihin toteutuksen mukaan [23]. REST-palvelujen etuja ovat keveys ja yksinkertaisuus. Esimerkiksi REST-palveluissa viestien formaattina ei ole pakko käyttää SOAP-standardissa määriteltyjä viestejä, vaan nii-

den rakenne ja sisältö voidaan tarvittaessa itse määritellä. REST-palvelun kuvauksessa ei ole myöskään pakko käyttää WSDL-dokumenttia. Vaikka REST:n käyttö painottuu pääasiassa avoimien rajapintojen käyttöön, on REST kuitenkin yksi tapa toteuttaa HTTP-protokollaan perustuvia web services -palveluja. Web-rajapintoja kutsutaan usein virheellisesti REST-rajapinnoiksi, ottamatta kantaa kuinka tarkasti REST-tyyliä niissä on noudatettu [36].

SOAP-pohjaisissa palveluissa viestit ovat aina SOAP-viestejä ja palveluista on usein saatavilla myös WSDL-dokumentti [35]. Tarkemman määrittelyn ja paremman ohjelmointityökalujen tuen takia SOAP-viesteihin perustuvia web services -palveluja käytetään pankkien maksuliikenneaineistojen välittämisessä. Tutkielman empiirisessä osassa esitellään SOAP-pohjaisen web services -palvelun käyttö kuittitietojen välityksessä yrityksen ja pankin välillä.

2.1 Web services -palveluarkkitehtuuri

Kregerin [26] mukaan web services -palvelu sisältää kolme eri toimijaa: palveluntarjoaja, palvelupyynnön esittäjä ja palvelurekisteri. Palveluntarjoaja toimii verkkopalvelun omistajana, joka hallinnoi kyseistä verkkopalvelua internetissä. Palveluntarjoaja kuuntelee pyyntöjä, käsittelee ne ja toimittaa tiedot takaisin pyynnön esittäjälle. Palvelunpyynnön esittäjä tunnistaa verkkopalvelun käyttäjät, jotka voivat olla yksittäisen käyttäjän web-selain tai muu käytössä oleva verkkopalvelu. Palvelurekisteri sisältää keskitetysti tiedot kaikista olemassa olevista palvelukuvauksista.

Palvelu ja palvelukuvaus ovat kaksi sovellusta, joissa palvelu on palveluntarjoajan alustalle asennettu ohjelmistomoduuli. Palvelupyynnön esittäjä saa moduulin käyttöönsä kutsumalla sitä tai palvelu voidaan toteuttaa pyynnön esittäjänä, joka on yhteydessä muiden verkkopalvelujen kanssa. Palvelukuvaus sisältää tiedot palvelun käyttöliittymästä ja toteutuksesta. Tiedot koostuvat tietotyypeistä, toiminnoista, kutsumisohjeista, palvelun sijainnista verkossa ja tarvittavista metatiedoista, jotka helpottavat palvelun käyttöä [26].

Fenselin ja muiden [11] mukaan palvelu julkaistaan yhdessä palvelukuvauksen kanssa, jotta palvelunpyytäjä löytää sen. Palvelu julkaistaan vaatimuksista riippuen tuotanto- tai kehitysympäristössä. Find-toiminnolla saadaan yksityiskohtaista tietoa palvelukuvauksesta ja sen käytöstä palvelupyynnön esittäjän näkökulmasta. Invoke-toiminnolla palvelupyynnön esittäjä kutsuu käytettävää palvelua hyödyntäen Find-toiminnolla saatuja tietoja palvelun sijainnista [26].

2.2 Web services -palvelun toteutuksen vaiheet

Kreger [26] määrittelee verkkopalvelun toteutuksessa neljä erillistä vaihetta. Ensimmäinen vaihe sisältää verkkopalvelun julkaisemiseen tarvittavan suunnittelun, analysoinnin, toteutuksen ja testaamisen. Tavoitteena on julkaista toimiva ja testattu palvelu, joka vastaa sille asetettuja liiketoiminnan vaatimuksia. Tavoitteen saavuttamiseksi on toteutuksessa ja testauksessa huomioitava kehitysprojektin tavoitteiden ja menettelytapojen määrittäminen, liiketoiminnan tarpeiden analysointi sekä palvelumääritykset ja niiden suunnittelu [33].

Toinen vaihe käsittää käyttöönoton ja verkkopalvelun julkaisemisen käyttäjille. Käyttöönotto sisältää palveluliittymien julkaisun ja julkaisupalvelun toteutuksen eri vaiheissa [6]. Tässä vaiheessa esitellään myös verkkopalvelun yksityiskohdat, kuten sijainti, versio ja ohjeet toteutuksesta. Käyttöönotto pitää sisältää suunnitelman ja toimintaympäristön asetukset. Kolmannessa vaiheessa eli suoritusvaiheessa verkkopalvelu on julkaistu käytettäväksi internetin kautta, jolloin palvelunpyytäjä voi suorittaa find- ja invoke-toimintoja [26].

Neljäs vaihe eli hallintavaihe sisältää verkkopalvelusovelluksen hallinnan, ylläpidon ja arvioinnin. Tarkoituksena on monitoroida verkkopalvelun suorituskykyä toimintaympäristössä, jotta tiedetään, ovatko liiketoimintaprosessit, palvelun laatu ja kustannukset huomioitu riittävällä tasolla [33]. Palvelun vasteaikoja ja suorituskykyä mittaavia tietoja tallennetaan erilaisilla mittareilla, jotta saadaan selville vastaavatko ne palvelulle asettuja vaatimuksia [33].

2.3 REST:n käyttö

REST (Representational State Transfer) on HTTP-protokollaan perustuva arkkitehtuurimalli sovellusrajapintojen toteutuksissa. REST-arkkitehtuuri sisältää kolme pääkäsitettä: resurssi, esitys ja tila [10]. Resurssi voi olla mikä tahansa tieto, joka on tallennettu tietokoneelle ja voidaan esittää bittivirtana. Resurssit erotellaan toisistaan tunnuksella eri komponenttien välisessä tiedonsiirrossa, kun taas esitys sisältää resurssin metatietoja. REST-arkkitehtuurissa on kaksi eri tilatyyppiä: resurssitila ja sovellustila. Resurssitila sisältää palvelimen tuottamaa tietoa resurssista ja sovellustila kertoo missä tilassa asiakassovellus on suhteessa palvelimeen. REST pohjautuu tietoelementteihin, niiden tulkintoihin ja eri sovelluskomponenttien vuorovaikutukseen [10].

2.3.1 Historia

REST-arkkitehtuurin kehittäjänä pidetty Roy Fielding määrittelee sen sovellusarkkitehtuurityyliksi hajautetuille hypermediajärjestelmille. Hypermediajärjestelmällä tarkoitetaan järjestelmää, joka muodostuu toisiinsa linkitetyistä mediadokumenteista. Suurin syy REST-arkkitehtuurityylin kehitykselle oli tarve hallita web-protokollien, kuten HTTP:n ja URI:n kehitystä. Vuonna 1994 Fielding kehitti työryhmänsä kanssa HTTP/1.0-protokollan ja suunnitteli samalla jo seuraavaa versiota HTTP/1.1:stä, kun samaan aikaan kehitettiin ensimmäinen versio REST:stä. REST ohjasi HTTP:n kehitystä, mutta myös REST:iä kehitettiin HTTP:n rinnalla esiin tulleiden tarpeiden pohjalta [12].

REST pyrkii muodostamaan suunnitelman siitä, miten web-sovelluksen tulee toimia. Web-sovellus muodostuu kokoelmasta toisiinsa linkitetyistä sivuista, missä kukin sivu esittää (represent) käyttäjälle yhden tilan (state) sovelluksesta. Siirtyminen (transfer) tilasta toiseen tapahtuu sivujen sisältämien linkkien kautta [13].

2.3.2 Rajoitteeet käytössä

REST-arkkitehtuurissa määritellään kuusi eri rajoitetta, joita REST-pohjaisten järjestelmien tulee noudattaa. Rajoitteet ovat ohjeellisia ja hyväksi todettuja käytäntöjä paremman REST-palvelun rakentamiseen. Toisin kuin SOAP-pohjaisissa toteutuksissa, REST:n sisältämiä rajoitteita ei ole tarkoitus noudattaa orjallisesti. Rajoitteet tulee kuitenkin ottaa huomioon toteutuksessa, jos se on teknisesti perusteltua. Tästä syystä REST-pohjaiset palvelut mielletään usein kehittäjän näkökulmasta helpommin ylläpidettäviksi [17].

Fieldingin [12] esittämien REST-arkkitehtuurimallin rajoitteiden tarkoituksena on vastata uusien web-sovellusten vaatimuksiin. Uuden arkkitehtuurityylin vaatimukset web-sovelluksille ovat skaalautuvuus, parempi tietoturva, rajapintojen geneerisyys, komponenttien riippumattomuus ja välikomponenttien käyttö viiveen pienentämisessä komponenttien välillä sekä järjestelmän vanhojen osien kapselointi. Vaatimusten takia REST:ssä on mukana rajoitteita muista tietoverkkopohjaisista arkkitehtuurityyleistä. REST:n päärajoitteita ovat:

- asiakas-palvelin (client-server)
- tilaton (stateless)
- välimuisti (cache)

- yhdenmukainen rajapinta (uniform interface)
- kerroksittainen järjestelmä (layered system)
- ladattava koodi (code-on-demand)

Asiakas-palvelin-rajoitteen tarkoitus on erottaa sovelluksen toiminnallisuudesta kaksi eri osaa, joista toinen on asiakkaan ja toinen palvelimen vastuulla. Yleisin tapa vastuunjaosta on se, että asiakas vastaa käyttöliittymästä, kun taas palvelimen vastuulla on tietojen tallennus. Selkeä vastuunjako mahdollistaa yksinkertaisemmat komponentit, mikä parantaa järjestelmän skaalautuvuutta. Mikäli asiakkaan ja palvelimen väliseen rajapintaan ei tehdä muutoksia, voidaan molempia kehittää toisistaan riippumatta [12].

Fieldingin [12] mukaan tilattomuus-rajoite on asiakas-palvelin-rajoitteen lisärajoite. Tilattomuus tarkoittaa, ettei asiakas-palvelin-session tilaa tallenneta palvelimelle. Kun asiakas muodostaa pyynnön palvelimelle, pyyntö sisältää kaiken tiedon, mitä palvelin tarvitsee pyynnön käsittelyssä. Koska palvelimen ei tarvitse tallentaa tilaa asiakkaan pyyntöjen välillä, palvelin voi tarpeen mukaan vapauttaa omia resurssejaan hyvinkin nopeasti. Myös järjestelmän näkyvyyden ja luotettavuuden parantuminen ovat tilattomuuden etuja. Näkyvyydellä tarkoitetaan sitä, että asiakkaan pyynnön todellinen tila on suoraan nähtävissä kussakin pyynnössä. Luotettavuuden parantuminen tarkoittaa puolestaan sitä, että tilaton järjestelmä pystyy palautumaan vikatilanteista paremmin. Tilattomuuden haittana on verkon mahdollinen ylikuormittuminen, koska kaikki relevantti data täytyy siirtää jokaisen pyynnön mukana. Sovelluksen tilan tallentamista asiakkaalla voidaan pitää myös haittana, koska tällöin palvelimella ei ole kontrollia sovelluksen käyttäytymisestä.

Verkon mahdollisen ylikuormittumisongelman vähentämiseksi on REST:ssä olemassa välimuisti-rajoite. Välimuisti-rajoite tarkoittaa sitä, että palvelimen vastaus tulee sisältää tiedon, voidaanko vastaus tallentaa asiakkaan välimuistiin. Tällöin asiakkaan ei tarvitse tehdä uudestaan samaa pyyntöä palvelimelle, vaan pyynnön vastaus voidaan hakea uudestaan välimuistista. Välimuisti-rajoitteen käyttö mahdollistaa sen, että osa pyynnöistä voidaan jättää joko kokonaan tai osittain suorittamatta. Tämä parantaa järjestelmän suorituskykyä ja pienentää järjestelmän eri toimintojen välistä viivettä. Asiakkaan välimuisti voi sisältää myös vanhentunutta dataa, mikä toisaalta huonontaa järjestelmän luotettavuutta [12].

Fielding [12] esittää yhdenmukaisen rajapinta-rajoitteen yksinkertaistavan järjestelmän kokonaisarkkitehtuuria ja parantavan näin komponenttien näkyvyyttä.

Yhdenmukaisessa rajapinnassa toimivien komponenttien itsenäinen kehittäminen on mahdollista asiakas-palvelin-järjestelmässä. Koska rajapinnassa siirrettävä data noudattaa aina samaa formaattia riippumatta siitä, mikä olisi paras muoto kullekin sovellukselle, voi se tarkoittaa järjestelmän suorituskyvyssä ongelmia. Yhdenmukaiseen rajapintaan liitetään usein neljä lisärajoitetta: itseselitteiset viestit, resurssien tunnistaminen, hypermedian käyttö sovelluksen tilakoneena ja resurssien manipulointi esitysten kautta. Viestien itseselitteisyys tarkoittaa, että kaikki viestin käsittelyyn tarvittava tieto on viestissä itsessään. Resurssilla on yksiselitteinen tunniste ja esitys, resurssin esitys koostuu datasta ja metadatasta. Resurssin esitys voi muuttua ajan kuluessa, vaikka itse resurssi pysyisi samana. Kaikki hypermedian sisäiset viittaukset kohdistuvat aina johonkin resurssiin. Fredrichin [17] mukaan SOAP-pohjaisissa toteutuksissa palveluntarjoaja määrittelee kutsuttavat operaatiot. REST-pohjaisissa järjestelmissä operaatiot suoritetaan samalla tavalla resurssista riippumatta. Yhtenäisellä rajapinnalla turvataan se, että kaikkia resursseja on mahdollista käsitellä samalla tavalla, eikä jokaiseen resurssiin tarvita erillisiä komentoja.

Kerroksittainen järjestelmä-rajoite muodostuu keskenään kommunikoivista kerroksista. Kerrokset ovat järjestäytyneet hierarkkisesti, jokainen kerros voi kommunikoida vain ylä- ja alapuolellaan olevien kerrosten kanssa. Asiakkaan ja palvelimen välissä sijaitsevien kerrosten avulla voidaan kapseloida järjestelmän vanhentuneita osia. Kapseloinnissa vanhentuneen palvelun päälle rakennetaan uusi rajapintakerros, jonka avulla vanhentuneet asiakassovellukset eristetään uusista palveluista. Kerroksittaisen järjestelmän haittana nähdään viiveiden lisääntyminen, mikä johtuu datan käsittelyn määrästä. Prosessoinnin kuormitusta voidaan kuitenkin vähentää välimuistin käytöllä [12].

Fieldingin [12] mukaan ladattava koodi -rajoite mahdollistaa asiakassovelluksen toiminnallisuuden laajentamista. Palvelin voi lähettää vastauksen mukana koodia, joka suoritetaan asiakaspäässä, jolloin asiakassovellusten kehittäminen helpottuu. Ladattava koodi heikentää järjestelmän näkyvyyttä ja siitä syystä ladattava koodi -rajoite on vapaaehtoinen. Vapaaehtoisen ladattava koodi -rajoitteen vaikutukset ovat voimassa vain niissä järjestelmän osissa, joissa se on käytössä. Yrityksen omassa sisäverkossa asiakassovellukset voivat käyttää ladattava koodi-rajoitetta ainoastaan yrityksen omilta palvelimilta, yrityksen ulkopuoliset asiakassovellukset eivät käytä ladattava koodi -rajoitetta.

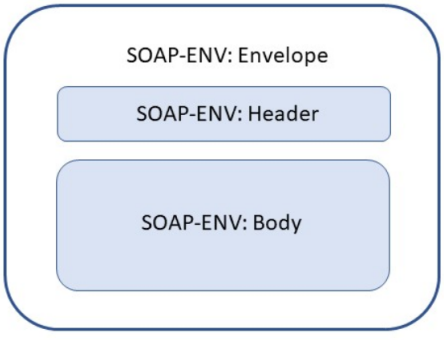
2.4 SOAP:n käyttö

SOAP (Simple Object Access Protocol) on protokolla, jonka avulla käyttäjä voi lähettää komennon palvelulle ja saada siitä vastauksen. Microsoftin kehittämä SOAP-protokolla perustuu XML-pohjaiseen strukturoituun tiedon välittämiseen verkkosovellusten välillä. SOAP-viesti rakentuu kolmesta eri osasta ja se on riippumaton kuljetusprotokollasta [37, 49].

Yksinkertaisimmillaan SOAP:a voidaan käyttää yksisuuntaiseen sanomien välitykseen, missä asiakassovellus lähettää SOAP-pyynnön palvelimelle, joka käsittelee pyynnön ja lähettää SOAP-vastauksen saadun tiedon perusteella [49]. Empiirisessä osassa esitellyssä web services -ratkaisussa käytetään yksisuuntaista kommunikointia, jossa pankkiyhteysohjelmasta lähetetään aineiston hakupyyntö pankin palvelimelle. SOAP-viestin sisältö esitetään XML-muodossa, mikä ei ota kantaa millä ohjelmointikielellä tai minkä käyttöjärjestelmän päälle SOAP-viestin käsittely on tehty [38].

2.4.1 SOAP-kirjekuori

SOAP-viestejä kutsutaan kirjekuoriksi (envelope). SOAP-kirjekuori on XML-muotoinen dokumentti ja se alkaa aina Envelope-elemetillä. SOAP-kirjekuori sisältää pakollisen runko-osan eli Body-elementin ja valinnaisen otsikon eli Header-elementin [21].



Kuva 2.1: SOAP-viestin osat

Kuvassa 2.1 luetellaan SOAP-viestin osat: kirjekuori (envelope), otsikko (header) ja runko (body), ja ne voidaan lähettää tai vastaanottaa eri verkkoprotokollien avulla, kuten HTTP, SMTP tai FTP [33]. Kirjekuori on pakollinen elementti ja se määritte-

lee viestin alun ja lopun. Samalla se kertoo viestin vastaanottajalle, onko viesti vastaanotettu ja voidaanko se käsitellä. Otsikko on valinnainen elementti, joka sisältää koodaussääntöjä, esimerkiksi salasanasuojatun verkkopalvelun käyttämisestä digitaalisella allekirjoituksella. Myös runko on pakollinen elementti ja se sisältää tietoja palvelun kutsuista, vastauksista ja siirrettävästä XML-muotoisesta datasta [26].

2.4.2 SOAP-otsikko

SOAP-kirjekuoren otsikko alkaa Header-elementillä. Header-elementin lapsielementtiä kutsutaan otsikkomerkinnäksi. Ensimmäisen tason otsikkomerkinnöissä voidaan käyttää kolmea SOAP-määritykseen kuuluvaa attribuuttia [48]:

- encodingStyle
- actor
- mustUnderstand

EncodinStyle-attribuutilla määritellään, kuinka tietoja tulisi välittää. Actor-attribuutti ilmaisee, mille sovellukselle kyseinen otsikkomerkintä on tarkoitettu. Kun sovellus tunnistaa itsensä actor-attribuutista, sovellus poistaa kyseiset otsikkomerkinnät ennen viestin välittämistä eteenpäin. MustUnderstand-attribuutilla määritellään otsikkotiedon pakollisuus. Jos otsikkotieto on määritelty pakolliseksi, sovelluksen on pystyttävä käsittelemään kyseinen otsikkotieto. Virhetilanteessa sovellus ei tunnista otsikkotietoa, jolloin se palauttaa SOAP-viestin lähettäjälle virheilmoituksen [49].

```
<eb:MessageHeader xmlns:eb="http://www.oasis-open.org/msg-header-2_0.xsd" SOAP-ENV:mustUnderstand="1" eb:id="375">
<eb:From>
<eb:PartyId>FI4950009420028730</eb:PartyId>
<eb:Role>Sender</eb:Role>
</eb:From>
<eb:From>
<eb:PartyId>OKOYFIHH</eb:PartyId>
<eb:Role>Intermediator</eb:Role>
</eb:From>
<eb:To>
<eb:PartyId>FI6833010002100048</eb:PartyId>
<eb:Role>Receiver</eb:Role>
</eb:To>
<eb:To>
<eb:PartyId>ESSEFIHX</eb:PartyId>
<eb:Role>Intermediator</eb:Role>
</eb:To>
<eb:MessageData>
<eb:MessageId>375</eb:MessageId>
<eb:Timestamp>2020-03-17T16:23:21+02</eb:Timestamp>
<eb:RefToMessageId/>
</eb:MessageData>
</eb:MessageHeader>
```

Kuva 2.2: SOAP:n Header-elementti

Kuvassa 2.2 on esimerkki Finvoice-verkkolaskun SOAP-kehyksestä. SOAP-sanoman header-elementin from- ja to-tageihin liittyy eb:Role-tagi, joka ilmaisee sanoman lähettäjän ja vastaanottajan. Laskun vastaanottaja ilmoittaa lähettäjälle oman tunnuksensa receiver-tagissa ja mahdollisen palveluntarjoajan (intermediator) tunnuksen intermediator-tagissa. Jos lasku lähetetään suoraan vastaanottajalle, ei vastaanottajan palveluntarjoajaa tarvita. Osapuolet käyttävät from- ja to-tageissa joko intermediaattorin kanssa tai keskenään sovittuja tunnisteita. Kun asiakkaan laskut välitetään laskuoperaattorin kautta, asiakas sopii operaattorinsa kanssa käytettävästä osoitteesta [14].

Sähköisten kuittitietojen välityksessä SOAP-kehystä ei yleensä tarvita, koska tarvittavat tiedot ilmoitetaan kuittisanoman MessageTransmissionDetails- ja BuyerPartyDetails-tiedoissa. Verkkolaskuoperaattoreiden toiminta vaatii kuitenkin SOAP-kehystä, mikä mahdollistaa kuittitietojen välittämisen suoraan myyjältä ostajalle tai kolmannen osapuolen kautta [14].

2.4.3 SOAP-runko

Runko-osa eli Body-elementti on SOAP-viestin pakollinen osa. Runko-osa sisältää XML-muotoista dataa, RPC-kutsun parametreja tai virheilmoituksia. Koska SOAP-standardissa ei määritellä Body-elementin tarkkaa sisältöä, kaikilla runko-osan elementeillä tulisi olla täydellinen nimi. SOAP-viestin vastaanottajan on pystyttävä tunnistamaan runko-osassa olevien elementtien merkitys, vaikkei elementeille määritelläkään mustUnderstand-attribuuttia [21].

```
<SOAP-ENV:Body>
<eb:Manifest eb:id="Manifest" eb:version="2.0">
<eb:Reference eb:id="Finvoice" xlink:href="375">
 <eb:schema eb:location="http://www.finanssiala.fi/finvoice/finvoice.xsd" eb:version="2.0"/>
</eb:Reference>
</eb:Manifest>
</SOAP-ENV:Body>
```

Kuva 2.3: SOAP:n Body-elementti

Kuvassa 2.3 Finvoice-verkkolaskun SOAP-ENV:Bodyssä viitataan alkuperäiseen verkkolaskuun. Reference eb:id kertoo, mistä viitteestä href:ssä on kyse eli tässä esimerkissä arvo 375, joka voi olla sama kuin eb:MessageId. MessageId:n pituus on maksimissaan 48 merkkiä ja SOAP-ENV:Bodyn lopetus-tagi päättää SOAP-sanoman body-elementin [14].

2.4.4 SOAP-virheiden käsittely

Kun clientin ja palvelimen välisessä kommunikoinnissa tapahtuu virhe, palvelin lähettää virheilmoituksen SOAP-viestin lähettäjälle SOAP-vastauksen runko-osassa olevan fault-elementin avulla. SOAP-virheilmoitus sisältää virhekoodin, virheen kuvauksen, tiedon virheen suorittaneesta osasta sekä virheen yksityiskohdat. Virhekoodityyppejä ovat VersionMismatch, MustUnderstand, Server ja Client [38]. VersionMismatch tarkoittaa virheellistä SOAP-kirjekuoren nimeä. MustUnderstand tarkoittaa, ettei lapsielementin mustUnderstand-attribuutin arvoa 1 osattu käsitellä. Server tarkoittaa palvelinongelmaa, ja client ilmaisee, että SOAP-viesti oli muotoiltu väärin tai se sisälsi virheellistä tietoa [44]. SOAP-standardi määrittelee Fault-elementille neljä lapsielementtiä [21]:

- faultcode
- faultstring
- faultfactor
- detail.

Faultcode-elementti on pakollinen ja siinä välitetään virheen konekielinen syy. Faultstring-elementti on myös pakollinen ja se sisältää virheilmoituksen selväkielisenä kuten 'File access denied'. Faultfactor-elementti on valinnainen tieto ja siinä kerrotaan SOAP-virheen aiheuttaneen sovelluksen URI-osoite. Detail-elementti on joko pakollinen tai valinnainen tieto ja se sisältää virhetilanteen tarkemman kuvauksen. Jos SOAP-viestin runko-osan käsittely on aiheuttanut virhetilanteen, detail-elementti on pakollinen [21].

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode>
         <faultstring xsi:type = "xsd:string">
            Failed to locate method (ValidateCreditCard) in class (examplesCreditCard) at
               /usr/local/ActivePerl-5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
         </faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Kuva 2.4: SOAP:n Fault-elementti

Kuvassa 2.4 Fault-elementin Faultcode-lapsielementtiin kirjoitetaan virheen konekielinen syy ja sen merkitys selväkielisenä Faultstring-lapsielementtiin. Molemmat lapsielemntit ovat pakollisia, kun SOAP-viestin lähettäjälle palautetaan virheilmoitus SOAP-vastauksen runko-osan fault-elementissä.

2.5 Edut ja haitat

Web services -teknologia valittiin yhteysratkaisuksi pankkien ja yritysasiakkaiden väliseen automatisoituun sovelluksien väliseen viestintään. Pankkien päätös perustui siihen, että web services -toteutukset pohjautuvat kansainvälisiin standardeihin ja ne voidaan toteuttaa nykyaikaisilla ohjelmistokehitystyökaluilla [15]. Tähtinen [45] kuvaa web services -yhteiskäytännön etuja integraatioprosessin näkökulmasta. Suurin etu integroitavien tietojärjestelmien kannalta saavutetaan sillä, että teknologian käyttö on alustariippumatonta, koska web services -rajapinnat perustuvat itsensä kuvaaviin XML-sanomiin. Tämä tarkoittaa sitä, että asiakas- ja palvelinsovellukset voivat toimia eri alustoilla, riippumatta käyttöjärjestelmästä, ja ne voivat olla kirjoitettu eri ohjelmointikielillä.

SOAP-viestien välityksessä käytetään usein HTTP-protokollaa, jolloin siirrettävä data reititetään saman tietoliikenneportin kautta kuin web-selainten liikennekin. Yrityksen tietoturvan kannalta tällöin päästään eroon haastavien TCP/IP-porttien käytöltä. SOAP-protokolla on kevyt ja suoraviivainen käyttää, eikä se vaadi suuria suoritustehoja järjestelmässä käytettävien laitteiden prosessoinnilta [29].

Hogg ja muut [19] tarkastelevat yhteiskäytännön etuja järjestelmäintegraatioiden kannalta ja mainitsee web services -palveluiden mahdollistavan yrityksen sisäisen EAI (Enterprise Application Integration) ja yritysten välisen B2BAI (Business to Business Application Integration) ohjelmistojen integroinnin. Tästä seuraa se, että järjestelmien tarjoamat palvelut ovat paremmin käytettävissä (access enablement). Web services -tekniikoiden avulla saadaan aikaiseksi ketteriä ohjelmistokomponentteja, joiden avulla järjestelmän olemassa olevia toiminnallisuuksia voidaan laajentaa. Käytetyt tekniikat mahdollistavat myös ohjelmistojen evolutionaariseen ja inkrementaaliseen käyttöönottoon, jolloin muutoksia olemassa oleviin järjestelmiin ei tarvita. Kaikki tämä vähentää järjestelmien integraatioon vaativien resurssien tarvetta.

Luon ja muiden [27] mukaan HTTP-protokollalla on negatiivisia vaikutuksia web services -ratkaisujen toteutuksissa. HTTP:n käyttötarkoitus oli alkujaan tilatto-

maan synkroniseen dokumenttien hakeminen web-selaimella. HTTP ei kuitenkaan ole kovin turvallinen sovellusten välisessä viestinnässä, koska viestien asynkronisuutta ja varmaa läpimenoa ei voida toteuttaa tiedonsiirtotasolla. Web services -tekniikoiden kehittämistä ja toteuttamista palvelukeskeisissä arkkitehtuureissa ovat hidastaneet useat erilaiset spesifikaatiot, joita eri sovelluskehykset tukevat niiden kehittäjien mukaan. Tämä on osaltaan vaikeuttanut tekniikoiden laajamittaista hyväksyntää ja käyttöä. Jotta web services -tekniikat lunastavat odotukset palvelukeskeisen ohjelmistojen integroinnin välineenä, on näitä tekniikoita tuettava integraatiototeutuksissa. Yhtä selkeää ohjetta ei kuitenkaan ole olemassa, mitä spesifikaatioita tulisi noudattaa palvelukeskeisen arkkitehtuurin toteutuksissa.

3 XML Extensible Markup Language

Web services -palveluita käytetään pääasiassa järjestelmien tai ohjelmien väliseen vuorovaikutukseen. Palveluiden palauttamat vastaukset esitetään usein XML- merkintäkielellä (Extensible Markup Language). Voidaankin todeta, että web services -yhteyskäytäntö pohjautuu HTTP- ja XML-standardeihin [43]. Tässä luvussa tarkastellaan XML-merkintäkielen rakenteita, nimiavaruuksia ja skeemaa esimerkkien avulla. Lisäksi esitellään web services -palveluiden SOAP-pohjaisissa toteutuksissa käytettyä WSDL-dokumenttia, joka pohjautuu niin ikään XML-merkintäkieleen. WSDL-dokumentissa kerrotaan käytetystä palvelusta ja sen toiminnasta palvelun käyttäjille kaikki oleellinen tieto pyyntöjen muodostamista varten [40].

XML-merkintäkielen avulla välitetään komentoja ja viestejä eri ohjelmointikielillä implementoitujen sovelluksien välillä. XML:ää voidaan kuvauskielenä verrata HTML-kieleen, jolla luodaan pääasiassa www-sivuja. XML:n käyttö keskittyy datan tallentamiseen ja välittämiseen, kun HTML:ää käytetään pääasiassa data esittämiseen. XML-kielellä kuvataan mitä välitettävä data on ja elementtien nimet usein kertovat suoraan varsinaisen datan sisällön [43].

3.1 XML-dokumentin rakenne

XML-dokumentti muodostuu juurielementeistä ja niiden lapsielementeistä sekä elementeissä käytetyistä attribuuteista. Juurielementin attribuutit sisältävät XML-dokumentin nimiavaruudet ja ne kuvaavat dokumentin rakennetta. Elementtien tietotyypit kuvataan XML-skeemassa, joita käytetään niin ikään juurielementin attribuutteina.

```
<?xml version="1.0"?>
- <Finvoice>
  - <InvoiceRow>
      <ArticleIdentifier>tuote1</ArticleIdentifier>
      <ArticleName>Tuote1 nimi</ArticleName>
      <DeliveredQuantity QuantityUnitCode="kpl">3,00</DeliveredQuantity>
      <UnitPriceAmount AmountCurrencyIdentifier="EUR">100,00</UnitPriceAmount>
      <RowVatRatePercent>24,00</RowVatRatePercent>
      <RowVatAmount AmountCurrencyIdentifier="EUR">72,00</RowVatAmount>
      <RowVatExcludedAmount AmountCurrencyIdentifier="EUR">300,00</RowVatExcludedAmount>
      <RowAmount AmountCurrencyIdentifier="EUR">372,00</RowAmount>
    </InvoiceRow>
  - <InvoiceRow>
      <ArticleIdentifier>tuote2</ArticleIdentifier>
      <ArticleName>Tuote2 nimi</ArticleName>
      <DeliveredQuantity QuantityUnitCode="kpl">6,00</DeliveredQuantity>
      <UnitPriceAmount AmountCurrencyIdentifier="EUR">100,00</UnitPriceAmount>
      <RowVatRatePercent>24,00</RowVatRatePercent>
      <RowVatAmount AmountCurrencyIdentifier="EUR">144,00</RowVatAmount>
      <RowVatExcludedAmount AmountCurrencyIdentifier="EUR">600,00</RowVatExcludedAmount>
      <RowAmount AmountCurrencyIdentifier="EUR">744,00</RowAmount>
    </InvoiceRow>
</Finvoice>
```

Kuva 3.1: XML-dokumentin rakenne

Kuvassa 3.1 esitellään XML-dokumentin rakennetta esimerkin avulla. Ensimmäinen rivi sisältää prosessointikäskyn, joka kertoo käytetyn XML-version ja merkistön. Toisella rivillä on dokumentin juurielementti Finvoice. XML-dokumentin rakenne voi sisältää vain yhden juurielementin, jonka sisällä koko XML-dokumentti kuvataan. Juurielementin alla on InvoiceRow-elementti, jonka alla on kahdeksan lapsielementtiä. DeliveredQuantity-elementillä on attribuuttina QuantityUnitCode, jolla ilmaistaan toimitetun määrän yksikkö. Attribuutit kertovat minkä tyyppistä dataa elementti sisältää ja ne voivat sisältää vain yhden arvon. InvoiceRow-elementin lapsielementissä RowVatAmount attiribuuttina on AmountCurrencyIdentifier, joka kertoo käytetyn valuutan [46].

3.2 XML-dokumentin nimiavaruus

XML-dokumenteissä elementtien nimet voivat aiheuttaa ristiriitoja eri sovelluksien välillä. Samanimiset elementit voi tarkoittaa eri asioita tai niiden sisällöt voivat olla täysin erilaiset, tällöin sovellus ei pysty käsittelemään XML-dokumenttia luotettavasti. Elementtien väliset nimeämisongelmat ratkaistaan käyttämällä nimiavaruuksia (XML-Namespace) [46].

Nimiavaruudessa elementin nimen mukana on viittaus mihin nimiavaruuteen

elementti liittyy. Nimiavaruus määritellään elementin xmlns-attribuutilla ja määrittely tehdään dokumentin juurielementissä. Attribuutti antaa nimiavaruudelle uniikin nimen. Nimiavaruuden attribuutti sisältää yleensä URI-osoitteen (Uniform Resource Identifier). Osoitteesta ei tarkasteta mitään, vaan se sisältää tietoa nimiavaruudesta. URI-osoitteella tarkoitetaan internetissä olevan resurssin yksilöivää tunnusta, resurssi voi olla esimerkiksi tiedosto tai palvelu. URL-osoite (Uniform Resource Locator) ja URN-tunnus (Uniform Resource Name) ovat URI:n alaosoitteita. URL-osoite yksilöi internetissä olevan resurssin sijainnin sekä sen käyttöön tarvittavan yhteyskäytännön. URN-tunnus yksilöi internetissä olevan resurssin URN-skeeman avulla ottamatta kantaa sen saavutettavuuteen, URN-tunnus on pysyvä ja ainutkertainen. URI-osoite voi olla samanaikaisesti URL-osoite ja URN-tunnus [43].

```
<?xml version="1.0"?>
<ns1invoice xmlns:ns1="http://www.w3.org" xmlns:ns2="http://www.w3.org">
  - <ns1:InvoiceRow>
        <ns1:ArticleIdentifier>tuote1</ns1:ArticleIdentifier>
        <ns1:ArticleName>Tuote1 nimi</ns1:ArticleName>
    </ns1:InvoiceRow>
  - <ns2:InvoiceRow>
        <ns2:td>tuote1</ns2:td>
        <ns2:td>tuote2</ns2:td>
    </ns2:InvoiceRow>
</ns1invoice>
```

Kuva 3.2: XML-dokumentin nimiavaruus

Kuvassa 3.2 XML-dokumentissa on kaksi samanimistä, mutta sisällöltään erilaista InvoiceRoW-elementtiä. Elementit ovat eroteltu toisistaan ns1- ja ns2-attribuuteilla, jotta ne voidaan käsitellä eri ohjelmistojen välillä. Molemmille nimialueiden attribuuteilla on määritelty URI tiedonhakua varten.

3.3 XML-skeema

XML-skeemassa määritellään XML-dokumentin rakenne. Skeemaa käytetään XML-dokumentin rakenteen ja sisällön tarkastamiseen. XML-skeeman avulla voidaan varmistaa, että XML-dokumentti sisältää järjestelmän käyttämää dataa. Esimerkiksi sähköisestä kuittiaineistosta tarkastetaan, että kaikki pakolliset ostajan tiedot ovat vastaanotetussa aineistossa. XML-skeemassa määritellään mitä elementtejä dokumentissa voidaan esittää. Sen lisäksi skeemassa määritellään elementit ja niiden lapsielementit sekä niiden järjestys ja määrä. Myös tiedon pakollisuus ja sisältö määritellään skeemassa. Elementtien ja attribuuttien osalta määritellään lisäksi tietotyypit

sekä niiden oletusarvot [43, 46].

```
<?xml version="1.0"?>
<InvoiceDetails>
    <InvoiceTypeCode>INV01</InvoiceTypeCode>
    <InvoiceTypeCodeUN>380</InvoiceTypeCodeUN>
    <InvoiceTypeText>LASKU</InvoiceTypeText>
    <OriginCode>Original</OriginCode>
    <InvoiceNumber>375</InvoiceNumber>
    <InvoiceDate Format="CCYYMMDD">20200621</InvoiceDate>
</InvoiceDetails>
```

Kuva 3.3: XML-dokumentin skeema

Kun dataa siirretään kahden eri ohjelmiston välillä, voidaan elementtien sisältöä tulkita eri tavoilla Kuvassa 3.3 XML-dokumentissa InvoiceDate-elementin päivämäärä voidaan tulkita joko suomalaisella tai amerikkalaisella merkintätavalla. Kun elementille annetaan attribuuttina Format="CCYYMMDD", tiedon käsittelijät ymmärtävät päivämäärän amerikkalaisessa muodossa [43, 46].

3.4 WSDL:n käyttö

WSDL on XML-muotoon perustuva määrittelykieli, joka kehitettiin kuvaamaan ja julkaisemaan standardoidulla tavalla verkkopalvelujen käytettävät protokollat, toiminnot ja sanomaformaatit. W3C (World Wide Web Consortium) hyväksyi ja julkaisi WSDL:n version 1.1:n maaliskuussa 2001. Vaikka WSDL:n julkaisu on edelleen vain esitys, pidetään sitä käytännössä virallisena tapana määritellä web services -palvelu [30]. WSDL-dokumenttia käytetään ensisijaisesti SOAP-pohjaisissa web services -palveluissa. REST-palveluissa vastaavaa dokumenttia ei yleensä tarvita, koska palvelu tehdään itseään kuvaavaksi, jolloin erillistä dokumenttia ei tarvita. REST-palvelun käyttöön tarvittavat toiminnot esitellään usein listauksena palveluntarjoajan verkkosivuilla [40].

WSDL:ssä sanomien rakenne kuvataan yleensä XSD-tyyppimäärittelyllä (XML Schema Definition). WSDL-elementeillä kuvataan operaatiot, miten sanoman vastaanottaja palvelua käyttää sekä käytettävä protokolla ja osoite eli miten sanoma lähetetään web-palvelulle [30]. Taulukossa 3.1 on kuvattu WSDL:n tärkeimmät elementit.

Taulukko 3.1: WSDL:n elementit

Termi	Elementti	Selite
Datatyypit	types	Yleensä XSD-tyyppimäärityksiin perustuva viestinvälityksessä käytetyn tietotyypin ilmaisu.
Viesti	message	Abstraktia, tietotyypeiltään määrättyä sovellusten välillä kul-kevaa tietoa.
Operaatio	operation	Asbstrakti kuvaus operaatiosta tietylle viestille.
Porttityyppi	portType	Abstrakti määrittely operaatioista, jotka liittyy tiettyyn päätepisteeseen; määrittelee joukon operaatioita sidosta varten.
Sidos	binding	Sitoo porttityypillä määritellyt operaatiot ja viestit todelliseen protokollaan ja sanomatyyppiin.
Portti	port	Liittää sidoksen palvelun verkko-osoitteeseen.
Palvelu	service	Määritteleen palveluun kuuluvat portit ja mahdolliset määrittel-lyyn liittyvät laajennukset.

WSDL:n elementit muodostavat viisi osakokonaisuutta, joihin dokumentti voidaan jakaa. Types-elementissä määritellään ne tietotyypit, joissa viestin rakenne määritellään. Message-elementissä määritellään viestit, joita web services -palvelun kautta voidaan välittää. Viestit ovat rakenteellisia ja käyttävät types-osassa määritettyjä tietotyyppejä. PortType-elementissä määritellään palvelun tukemat operaatiot, jotka muodostuvat syöte- ja tulosteviestistä. Binding-elementissä määritellään viestien tekniset yksityiskohdat välityksessä, kuten mitä data formaattia käytetään. Service-elementissä kerrotaan porttimäärityksinä, mistä verkko-osoitteesta web services-palvelut löytyvät [21].

3.4.1 Types-elementti

Types-elementissä määritellään tietotyypit, joita sanomissa käytetään. WSDL:ssä tietotyyppien määrittely voidaan toteuttaa useille eri tekniikoilla, mutta yleensä siihen käytetään XML-skeemaa. Pankkien käyttämässä varmennepalvelussa CertificateRequest- ja ResponseHeader-tietotyypit sisältävät lähettäjän tiedot sekä sanoman aikaleimatiedot. Response-sanoma sisältää omat elementit vastauskoodille ja vastaustekstille. ApplicationRequest- ja Responce-elementeissä käytetty base64Binary-tietotyyppi määrittelee elementin sisällön RFC 2045 -standardin mukaiseksi base64-koodatuksi dataksi. CertificateServiceFault-tietotyyppi sisältää elementit järjestelmävirhesanomille [21].

3.4.2 Message-elementti

Message-elementti määrittää sovellusten välillä välitettävän viestin, joita yhdessä WSDL-dokumentissa voi olla useita. Elementtien rakenne on melko yksinkertainen ja sillä voi olla vain yksi name-attribuutti, jolla kerrotaan viestin nimi. Message-elementti voi sisältää vain yhden lapsielementin, jolla voi olla vain kolme eri attribuuttia [21].

```
<wsdl:message name="certificateServiceFault">
    <wsdl:part name="certificateServiceFault" element="tns:certificateServiceFaultElement"/>
</wsdl:message>
<wsdl:message name="getCertificateRequest">
    <wsdl:part name="getCertificatein" element="tns:getCertificatein"/>
</wsdl:message>
<wsdl:message name="getCertificateResponse">
    <wsdl:part name="getCertificateout" element="tns:getCertificateout"/>
</wsdl:message>
```

Kuva 3.4: WSDL:n Message-elementti

Kuvassa 3.4 esitellään varmennepalvelun viestejä. Message-elementillä on ainoastaan name-attribuutti, joka kertoo viestin nimen. Varmennepalvelun WDSL:ssä viestit ovat yksiosaisia, joten niillä on vain yksi part-lapsielementti. Element-attribuutilla viitataan types-elementissä määriteltyyn tietotyyppiin [21].

3.4.3 portType-elementti

PortType-elementti tarkoittaa joko käyttäjän tai palvelimen tukemaa operaatiojoukkoa. Porttityypillä voi olla yksi tai useampi operaatio, jossa määritellään operaa-

tiossa välitettävät viestit ja viestin kulkusuunta. Name-attribuutti kertoo operaation nimen, joita ei yhden porttityypin sisällä saa olla kuin yksi samanniminen. Input-, output- ja fault-elementit ovat operaation lapsielementtejä, joista input- ja output-elementtien järjestys ja esiintyminen kertovat operaation tyypin. WSDL määrittelee neljä erityyppistä tiedonvälitystapaa [21]:

- yksisuuntainen (one-way)
- pyyntö-vastaus (request-response)
- anottu vastaus (solicit-response)
- ilmoitus (notification)

Yksisuuntaisessa tiedonvälityksessä päätepiste vastaanottaa viestin, mutta ei vastaa viestiin, itse operaatio sisältää vain input-elementin. Pyyntö-vastaus-tapauksessa päätepiste vastaanottaa viestin ja palauttaa vastauksen, operaatio sisältää input- ja output-elementit tässä järjestyksessä. Anottu vastaus sisältää päätepisteen lähettämän viestin ja saadun vastauksen, muuten operaatio sama kuin pyyntö-vastaus-tapauksessa. Ilmoituksessa päätepiste lähettää viestin, mutta ei saa vastausta, operaatio ei sisällä kuin output-elementin [21].

```
<wsdl:portType name="OPCertificateServicePortType">
  - <wsdl:operation name="getCertificate">
        <wsdl:input name="getCertificateRequest" message="tns:getCertificateRequest"/>
        <wsdl:output name="getCertificateResponse" message="tns:getCertificateResponse"/>
        <wsdl:fault name="certificateServiceFault" message="tns:certificateServiceFault"/>
    </wsdl:operation>
</wsdl:portType>
```

Kuva 3.5: WSDL:n portType-elementti

Kuvassa 3.5 esitellään varmennepalvelun porttityypit. Input-, output- ja fault-elementtien message-attribuutilla viitataan vastaanotettavaan viestiin. Fault-elementti määrittää vastausviestin sisällön, mikäli saatua viestiä ei pystytty käsittelemään. Fault-elementin name-attribuutti on pakollinen, input- ja output-elementeille name-attribuutti on valinnainen [21].

3.4.4 Binding-elementti

Binding-elementissä kerrotaan tiedonsiirrossa käytetyn viestin muoto ja protokolla. Elementin pakollisia tietoja ovat name-attribuutti, sekä type-attribuutti, joka viittaa

määriteltyyn porttityyppiin. Binding-elementti sisältää operation-elementin, jonka tulee viitata sitä operaation portti-tyyppiä, joka binding-elementin type-attribuuttissa on kerrottu. WSDL määrittelee viisi kohtaa, johon voidaan lisätä laajennuselementtejä, ja joiden avulla sidonta tehdään käytettyyn tiedonsiirtoprotokollaan ja viestin formaattiin. SOAP-sidonnassa laajennuselementtejä käytetään kaikissa viidessä WSDL:n määrittelemässä kohdassa eli binding-, operation-, input-, output- ja fault-elementeissä. Laajennuselementtien alussa käytetty SOAP-tunnus määrittää elementtien käyttämän nimiavaruuden, minkä johdosta WSDL-standardin elementit ovat eri nimiavaruudesa kuin laajennuselementit [21],[47].

```
<wsdl:binding name="OPCertificateServiceHttpBinding" type="tns:OPCertificateServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  - <wsdl:operation name="getCertificate">
        <soap:operation soapAction=""/>
      - <wsdl:input name="getCertificateRequest">
            <soap:body use="literal"/>
        </wsdl:input>
      - <wsdl:output name="getCertificateResponse">
            <soap:body use="literal"/>
        </wsdl:output>
      - <wsdl:fault name="certificateServiceFault">
            <soap:fault use="literal"/>
        </wsdl:fault>
    </wsdl:operation>
</wsdl:binding>
```

Kuva 3.6: WSDL:n Binding-elementti

Kuvassa 3.6 esitellään varmennepalvelun elementeistä binding ja operation. Bindingelementin attribuuteilla määritellään SOAP-tyyli ja käytettävä protokolla. Esimerkissä Style-attribuutti viittaa SOAP:n dokumentti-tyyliseen sanomaan ja transport-attribuutti määrittää protokollan. Binding-elementti on aina pakollinen WSDL-dokumentissa, jos sanoman välitykseen käytetään SOAP-protokollaa. Operation-elementti ja sen soapAction-attribuutti ovat pakollisia, jos SOAP:n kanssa käytetään HTTP-protokollaa. Laajennuselementti soap:body määrittää, kuinka viestien osat tallennetaan SOAP-viestin body-elementtiin. Kuvassa 3.6 elementin use-attribuutin arvona on "literal", jolloin body-osassa välitettävää sisältöä ei salata. Mikäli sisällön salaus halutaan purkaa, attribuutin arvo on "encoded". Laajennuselementti soap:fault määrittää sen, kuinka SOAP-virheilmoitus muodostetaan, jollei SOAP-sanoman vastaanottaja ole ymmärtänyt lähettäjän pyyntöä tai pyyntöä ei voitu toteuttaa [21],[47].

3.4.5 Service- ja port-elementit

Service-elementti kokoaa yhteenkuuluvat portit yhdeksi palveluksi. Elementti sisältää port-lapsielementin, jonka binding-attribuutti kertoo käytetyn sidokseen. SOAP-sidonnassa port-elementti sisältää yhden soap:address-elementin, jonka location-attribuutti kertoo sen URI-osoitteen, jolla SOAP-palvelua voidaan kutsua [21],[47].

```
- <wsdl:service name="OPCertificateService">
    - <wsdl:port name="OPCertificateServiceHttpPort" binding="tns:OPCertificateServiceHttpBinding">
        <soap:address location="http://domain.fi:1001/wsdl/CertificateService.xml"/>
      </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Kuva 3.7: WSDL:n Service- ja port-elementit

Kuvassa 3.7 esitellään varmennepalvelun service- ja port-elementit. Yksi service-elementti voi sisältää useita port-elementtejä eli palvelulla voi olla useita verkko-osoitteita. Kun port-elementti viittaa samaan sidokseen, yhdellä operaatiolla voi olla vaihtoehtoisia osoitteita [21],[47].

4 Salausprotokollat

Tässä luvussa esitellään web services -palvelun salaukseen ja XML-aineistojen allekirjoituksiin käytettyjä tekniikoita. Verkkopalveluissa viestit siirtyvät tekstimuodossa palvelimien välillä. Kun palvelulla on useampia osapuolia, käytetty siirtokerros ei takaa riittävä suojaa, vaan vaadittavaa tietoturva täytyy ulottaa viestitasolle. Tarkoituksena on suojata viestien ja sen osien luottamuksellisuus, eheys sekä kiistämättömyys. Tietoturvan kannalta on olennaista, että viesti toimitetaan muuttumattomana halutulle vastaanottajalle ja varmistutaan siitä, ettei viestiä pystytä käsittelemään, vaikka se päätyisikin väärälle vastaanottajalle [25].

Pankkien web services -palveluissa tietoliikenne salataan SSL-yhteyden (Security Socket Layer) avulla. Palvelun käyttäjän eli asiakkaan tunnistaminen perustuu PKI-varmenteisiin (Public Key Infrastructure) eli avaimiin. Pankki varmenteiden julkaisijana antaa avaimet asiakkaalle [31]. Pankkien web Services -palveluiden tietoturvaratkaisut mahdollistavat jokaisen pankin valitsevan itse varmenteita myöntävät viranomaiset (Certificate Authority). Salausalgoritmeilla ja avainten pituuksilla tulee olla 20 vuoden elinkaari, jotta saavutetaan riittävän vahva tietoturva. Pankin ja yrityksen järjestelmien välinen tietoliikenneturva rajataan point to point -yhteyksiin, riippumatta siitä, onko kyseessä loppuasiakas, palvelukeskus tai muu välittäjä. XML-aineistojen salaus pohjautuu XML-standardiin, jota käytetään tietosuojaan [15].

4.1 SSL-protokolla

Kearney [24] toteaa, että verkkopalveluiden yleisin tietoturvaratkaisu on SSL, jonka uusin versio tunnetaan lyhenteenä TLS (Transport Layer Security). SSL käyttää julkisen avaimen tekniikkaa todentamaan yhden tai molemmat päätepisteet, ja se perustuu symmetriseen avaimeen, jota käytetään lähetyspakettien salaamiseen. Liikenne päätepisteiden välillä tapahtuu salattuna, millä estetään kolmansien osapuolien puuttumisen datan sisältöön.

SSL tarjoaa asiakkaalle palvelimen identiteetin todentamisen. Asiakkaan henkilöllisyys voidaan välittää palvelimelle myös käyttämällä asiakaspäätevarmenteita.

Varmenteilla turvataan eheys ja luottamuksellisuus asiakkaan ja palvelimen välisen siirron aikana. SSL-protokolla ei kuitenkaan tarkista sähköisillä varmenteilla tunnistettujen kohteiden luotettavuutta. Se vain varmistaa, että varmenteen on myöntänyt luotettu varmentaja, jolla on sertifikaatti [24].

Yu ja muut [50] toteavat, että SSL-tunnelissa viestin sisältö on suojattu, mutta SSL ei ota kantaa siihen mitä viestille tehdään yhteyden eri päissä. Jos tunnelissa tarvitaan SOAP-viestin käsittelyä, tällöin SSL-yhteys on katkaistava. Kun SSL-yhteys avataan uudestaan, päätepisteet ovat menettäneet yhteyden sillä välin hetkeksi. Kun viestit ovat SSL-salattuja, sovellustason palomuuri ei voi tutkia viestin sisältöä, jolloin yhteys on katkaistava hetkeksi uudelleen tai vaihtoehtoisesti viestit jätetään tutkimatta. Tästä syystä SSL-protokolla ei tarjoa kokonaisratkaisua, mikä suojaisi sovellusten kautta arkaluonteiset tiedot palvelupyynnöstä palomuuriin, sovelluspalvelimiin ja yrityksen sisäiseen verkkoon [4]. SSL on ns. point to point -ratkaisu eli data turvataan pisteestä pisteeseen siirron aikana [24]. Jos jokin päätepiste tai välityspalvelin ei ole luotettava, kuljetuskerroksen suojaus ei tarjoa riittävää suojaa ulkopuolisia uhkia vastaan [28].

SSL tarjoaa turvallisen yhteyden selaimen ja palvelimen välillä vain silloin, kun hyökkäyksen mahdollisuus on epätodennäköisin. SSL-protokolla ei huomioi viestien autentikointia, datan eheyttä, eikä viestien kiistämättömyyttä. SSL on luotettava tekniikka kahden koneen välisen liikenteen suojaamiseksi, mutta se ei tarjoa riittävää suojaa useimmille verkkopalveluille [50].

4.2 XML-allekirjoitukset

Curpehyn [8] mukaan isokokoisten XML-tiedostojen kokonaan salaaminen ei ole tarkoituksenmukaista, vaan tällöin on järkevää salata tiedostosta vain tietyt osat. Julkisella avaimella salatun viestin saa auki vain salaisen avaimen haltija. Tällä varmistetaan viestin luottamuksellisuus sekä haluttu vastaanottaja. Viestin allekirjoittamisella varmistetaan, ettei viestin sisältö ole muuttunut siirron aikana [24].

Dournaee [9] toteaa, että XML-allekirjoituksen suunnittelussa otettiin erityisesti huomioon skaalautuvuus ja joustavuus. XML-allekirjoitus on hyvin muotoiltu XML-dokumentti, joten sitä voidaan käsitellä ja lukea normaalisti, lukuun ottamatta itse allekirjoitusosaa ja sinetin arvoa. Kaikki allekirjoituksen käsittelemiseen tarvittavat tiedot, mukaan lukien varmennustiedot, voidaan sisällyttää allekirjoitukseen. Verrattuna perinteiseen digitaaliseen allekirjoitukseen, XML-allekirjoituksessa

on useita kehittyneitä ominaisuuksia. Perinteistä digitaalista allekirjoitusta voidaan käyttää vain koko asiakirjan allekirjoittaminen. XML-allekirjoitusta voidaan käyttää vain tietyn osan allekirjoittamiseen, jolloin asiakirjan muut osat ovat edelleen muokattavissa rikkomatta allekirjoitusta. Asiakirjan eri osia voidaan allekirjoittaa milloin tahansa ja eri käyttäjien toimesta, eikä allekirjoituksien lukumäärällä ole rajoituksia.

Bartel ja muut [1] kuvaavat XML-allekirjoituksen tavaksi luoda yhteys avaimen ja datan välille. XML-allekirjoituksessa ei huomioida allekirjoitetun datan sisältöä, eikä varmisteta itse avaimien ja niiden haltijoiden välistä yhteyttä. Osana XML:n tietoturvaa XML-allekirjoitus ei yksin ratkaise palveluiden turvallisuus- ja luottamusongelmia. Bartel ja muut [1] luettelevat XML Signature-määrittelyssä joukon toimenpiteitä, jotka suoritetaan allekirjoituksen luomisen ja varmennuksen aikana. Dournanee [9] kuvaa XML-allekirjoituksen luomista ytimen tuottamiseksi, joka muodostuu viitteiden ja allekirjoituksen tuottamisesta. XML-allekirjoituksen ytimen varmistaminen muodostuu myös kahdesta eri osasta, viitteiden ja allekirjoituksen varmistamisesta. XML-allekirjoituksella varmistetaan, että allekirjoitettu data ei ole muuttunut ja että allekirjoitus on tehty allekirjoittajan salaisella avaimella. XML-dokumenttien allekirjoitus on määritelty W3C:n XML Signature Syntax and Processing-standardissa [48].

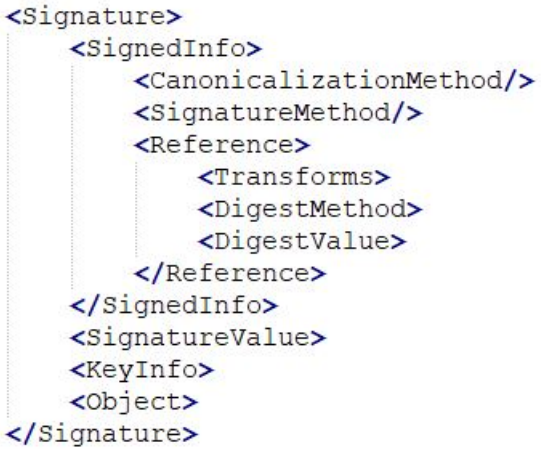
4.2.1 Allekirjoituksen luominen

Dournaee [9] toteaa, että XML-allekirjoitusta voidaan käyttää minkä tahansa digitaalisen sisällön kanssa, pääpaino kuitenkin XML-dokumenttien allekirjoittamisessa. Koska XML-allekirjoitus on hyvin muodostettu XML-dokumentti, sitä voidaan lukea ja käsitellä normaalisti. Kaikki allekirjoituksen käsittelyyn tarvittava tieto, myös varmennusinformaatio, voidaan sisällyttää allekirjoitukseen. XML-allekirjoituksen luominen muodostuu viitteiden ja allekirjoituksen tuottamisesta. XML-allekirjoituksen varmistus sisältää kaksi erillistä osaa: viitteiden ja allekirjoituksen varmistaminen [9]. Taulukossa 4.1 esitellään XML-allekirjoituksen elementit.

Taulukko 4.1: XML-allekirjoituksen elementit

Elementti	Selite
CanonicalizationMethod	Algoritmi SignedInfo-elementin kanonisoimiseen.
DigestMethod	Algoritmi tiivisteen laskemiseen allekirjoitettavasta datasta.
DigestValue	Laskettu tiiviste.
KeyInfo	Allekirjoituksen varmentamisen avain.
Object	Valinnaista dataa.
Reference	Allekirjoitettavan kohteen URI-attribuuttiviittaus.
Signature	Allekirjoituksen identifioiva juurielementti.
SignatureMethod	Salausalgoritmi SignedInfo-arvon muuntamiseen SignatureValue-elementin arvoksi.
SignatureValue	Allekirjoituksen arvo.
SignedInfo	Tiedot allekirjoitettavista kohteista.

XML-allekirjoitus muodostetaan Signature-elementin avulla. Kyseisellä elementillä voi olla ID-attribuutti, joka yksilöi allekirjoituksen, jos dokumentissa on useita allekirjoituksia. Jokaista allekirjoitettavaa dataobjektia varten on luotava Reference-elementti. Reference-elementti yksilöi allekirjoitettavan kohteen valinnaisen URI-attribuuttiviittauksen avulla. Ennen elementin luomista data muokataan halutulla tavalla, jonka jälkeen datasta lasketaan tiivistearvo. Saatu tiivistearvo viedään DigestValue-elementtiin ja arvon laskemisessa käytetty algoritmi Digest-Method-elementtiin. DigestMethod määrittää siis algoritmin, jota on käytetty tiivisteen laskemiseen allekirjoitettavasta datasta [1].



Kuva 4.1: XML-allekirjoituksen rakenne

Kuvassa 4.1 esitellään XML-allekirjoituksen rakenne ja elementtien järjestys rakenteen sisällä. Varsinainen XML-allekirjoituksen luomisessa tehdään ensin SignedInfo-elementti ja sille lapsielementeiksi SignatureMethod, CanonicalizationMethod sekä Reference-elementit. SignedInfo sisältää tiedon kaikista allekirjoitettavista kohteista. Elementin valinnaista Id-attribuuttia voidaan käyttää, jos on tarvetta viitata toisista allekirjoituksista tai kohteista [1].

SignatureMethod kertoo pakollisessa Algorithm-attribuutissa salausalgoritmin, jota käytetään kanonisoidun SignedInfo-arvon muuntamiseen SignatureValue-elementin arvoksi. Kanonisoinnissa fyysisesti erilaisista, mutta sisällöltään samanlaisista elementeistä muodostetaan identtiset elementit. CanonicalizationMethod puolestaan määrää SignedInfo-elementin kanonisoimiseen käytettävän algoritmin pakollisessa Algorithm-attribuutissaan.-elementtiin lasketaan arvo, joka saadaan käyttämällä SignedInfo-elementin algoritmia ja arvoa. Lopuksi muodostetaan Signature-elementti ja sille lapsielementeiksi aiemmin luodut SignedInfo- ja SignatureValue-elementit. Signature on allekirjoituksen juurielementti, joka identifioi allekirjoituksen. Elementin valinnaista Id-attribuuttia voidaan käyttää yksilöimiseen, jos allekirjoituksia on useampia [1].

4.2.2 Allekirjoituksen varmentaminen

XML-allekirjoituksen varmentaminen muodostuu Reference- ja Signature-elementtien varmentamisesta. Reference-elementin varmennus todistaa sen, että kaikki viit-

teet ovat ehyitä ja koskemattomia. Varmentamisessa muodostetaan ensin SignedInfo-elementti CanonicalizationMethod-elementissä kerrotun algoritmin mukaisesti. Seuraavaksi datasta lasketaan tiiviste DigestMethod-elementissä ilmoitetulla algoritmilla. Laskennan tulosta verrataan DigestValue-elementin arvoon ja jos arvot täsmäävät, varmennus on onnistunut [1].

Signature-elementtien varmennuksella todetaan, onko allekirjoitus aito ja kiistämätön. Varmennusta varten haetaan avainta koskeva tieto KeyInfo-elementistä tai mahdollisesta ulkoisesta lähteestä. KeyInfo kertoo allekirjoituksen varmentamiseen käytettävän avaimen sekä sen nimen ja sertifikaatin. Lopuksi lasketaan tiivistearvo SignatureMethod-elementin arvosta käyttämällä CanonicalizationMethod-elementtiä yhdessä aiemmin haetun avaintiedon kanssa. Laskennan tuloksena saatua tiivistearvoa verrataan KeyInfo-elementin arvoa ja jos arvot täsmäävät, varmennus on onnistunut [1].

4.3 XML-salaukset

Imamura ja muut [20] kuvaavat XML Encryption Syntax and Processing -julkaisussa menetelmän datan salauksen XML-dokumenteissa ja sen elementeissä. Salauksessa data viedään muodostetun XML-elementin lapsielementtiin. XML-salaus voidaan kohdistaa dokumentin eri osiin, mutta salauksen käyttöä dokumentin sisällä on syytä harkita, koska salatun datan käsittely hidastaa sen käyttöä. Koska yhden XML-dokumentin sisällä salaukset voivat olla eri osapuolten tekemiä, eivät kaikki dokumentin käyttäjät välttämättä pysty käsittelemään kaikkia dokumentin osia.

Trivedi ja muut [42] kuvaavat kirjassaan menetelmän, jossa salattu data salataan uudestaan. Menetelmä tunnetaan nimellä supersalaus, missä XML-dokumentin koko elementti salataan, ei siis pelkästään elementin sisältö.

4.3.1 Salauksen toteutus

Imamura ja muut [20] luettelevat toimenpiteet, joilla salaus toteutetaan. XML-salaus perustuu EncryptedData-elementtiin, joka on salauksen juurielementti ja identifioi salatun osion. Elementti korvaa salattavan datan dokumentissa. Ennen salausta valitaan salauksessa käytettävä algoritmi ja hankitaan salaukselle avain. Jos avainkin salataan, tehdään sille samat toimenpiteet kuin varsinaiselle salattavalle datalle.

Jos salattava data tai sen sisältö on XML-elementti, muunnetaan data UTF-8-

muotoon. UTF-8 on vaihtuvanpituinen koodaustapa, jossa merkkejä käsitellään kahdeksanbittisinä tavuina eli oktetteina. Salauksessa muodosteteaan EncryptedData- tai EncryptedKey-elementit rakenteineen. EncryptedKey-elementti sisältää käytetyn avaimen salatussa muodossa. Lopuksi alkuperäinen data korvataan EncryptedData-elementillä, jos salattu data tai sen sisältö on XML-elementti, muutoin EncryptedData-elementistä tehdään uuden XML-dokumentin juurielementti [20].

4.3.2 Salauksen purkaminen

Imamura ja muut [20] määrittelevät, miten salauksen purkamisessa haetaan ensin EncryptedData- tai EncryptedKey-elementistä salauksessa käytetty algoritmi ja parametrit. Salauksen käytetyn avaimen tiedot saadaan ds:KeyInfo-elementistä, jonka jälkeen avaimen salaus puretaan, jos avain on salattu. Seuraavaksi data puretaan saaduilla algoritmeilla ja avaimilla. Jos purettava data tai sen sisältö on XML-elementti, EncryptedData-elementti korvataan UTF-8-muodossa olevalla puretulla rakenteella. Kun EncryptedKey-elementti puretaan, palautetaan puretut tiedot salatun datan purkamisen jatkamista varten.

Trivedin ja muiden [42] mukaan XML-salauksessa ei oteta kantaa EncryptedData- tai EncryptedKey-elementtien eheyteen, mikä aiheuttaa turvallisuusongelman, jos hyökkääjä muuttaa salattavan datan rakenteita. Nämä hyökkäykset voidaan estää käyttämällä XML-allekirjoitusta. Toinen huomioitava tietoturvaongelma on salattavan datan sisälle piilotetut tiedostot, joita palomuurit eivät huomioi. Salauksissa tulisi käyttää satunnaistettuja algoritmeja, jotka eivät käytä samaan dataan samaa salausta ja samaa avainta. Tällä estetään, ettei hyökkääjä pysty keräämään avaimen ja salatun datan muodostamia pareja.

Imamura ja muut [20] korostavat, ettei salattua dataa kannata välttämättä allekirjoittaa, ellei tiedetä varmasti sen sisältöä. XML-allekirjoituksen ja XML-salauksen käyttö samassa dokumentissa saattaa aiheuttaa, että tiiviste jää näkyviin Reference-elementin sisälle selkokielisenä, kun XML-allekirjoitettua dataa salataan.

4.4 PKI Public Key Infrastructure

XML-sanomien allekirjoitus pohjautuu asymmetriseen salaustekniikkaan, joka muodostuu julkisesta ja yksityisestä salausavaimesta. Julkisella avaimilla voidaan purkaa yksityisellä avaimella tehty salaus, mutta julkinen avain ei yksistään riitä sala-

tun tiedon avaamiseen, ellei yksityisen avaimen haltijaa tiedetä. Varmenteen myöntäjä (Certificate Authority) allekirjoittaa sähköisen todistuksen eli varmenteen, jonka vaatimat tiedot varmenteen hakija toimittaa varmenteen myöntäjälle julkisen avaimen lisäksi. Tämän jälkeen hakijan toimittamat tiedot tarkistetaan ja jos ne todetaan oikeiksi, varmenteen myöntäjä laskee varmennepyynnössä olevista tiedoista tiivisteen ja allekirjoittaa varmenteen sähköisesti. Palvelu, joka varmennetta käyttää, tarvitsee varmenteen myöntäjän julkisen avaimen tiivisteen avaamiseen. Jos itse laskettu tiiviste täsmää julkisella avaimella purettuun tiivisteeseen, varmenteen sisältämää tietoa voidaan pitää luotettavana [22].

Varmenteiden jakelu ja ylläpito hoidetaan PKI-järjestelmällä, jonka keskeisin toimija on varmenteen myöntäjä CA (Cetficate Authority). Vaikka varmenteen myöntäjä voisi vastata kaikista varmennukseen liittyvistä toiminnoista, käytännössä varmenteen hakijalla on vastuu julkisen ja yksityisen avaimien muodostuksesta. Siksi varmenteen hakijan tiedot tarkastetaan usein erillisen rekisteröijän RA (Registration Authority) toimesta. Hakijan tunnistaminen ja rekisteröinti on oltava tehtynä ennen web services -palvelun käyttöönottoa. Asioitaessa pankin kanssa, yrityksen käyttämä pankkiyhteysohjelma muodostaa julkisen ja yksityisen avaimen pankkivarmenteen noudossa, jolloin varmenteen myöntäjänä toimii pankin tietojärjestelmää operoiva osapuoli. Jotta varmenteiden käyttöön olisi turvallista ja sujuvaa, PKI-järjestelmän pitää pystyä tarjoamaan seuraavat perustoiminnot [22]:

- rekisteröinti
- varmenteen luonti
- varmenteen jakelu
- varmennehakemiston ylläpito
- sulkulistapalvelu
- ylläpidolliset tukipalvelut
- toimintakuvaus

Rekisteröinnissä tarkastetaan varmennettavan hakevan henkilön tiedot. Varmenteen luonnissa hakijan varmennepyyntö muutetaan varmenteeksi ja varmenne allekirjoitetaan sähköisesti. Varmenteen jakelu pitää pystyä järjestämään haltijalle turvallisesti ja varmennehakemisto täytyy sisältää ajantasaiset tiedot kaikista CA:n myöntämistä varmenteista. Sulkulista eli CRL (Certificate Revocation List) tulee sisältää

kaikkien ennen vanhentumista mitätöityjen varmenteiden sarjanumerot. Sen lisäksi PKI-järjestelmän tulee tarjota tukipalveluina vikatilanteiden selvittelyt, käyttäjien kyselyt ja varmenteiden uusinta niiden voimassaoloajan päättyessä. PKI-järjestelmän toiminnot tulee olla kuvattua toimintakuvauksessa, jota kutsutaan varmennekäytäntölausumaksi, lyhenteenä CPS (Certificate Practice Statement) [22, 32].

5 Mikä on eKuitti?

Sähköisten kuittitietojen välityksessä käytetty web services -palvelu pohjautuu XML-standardeihin ja aineistojen strukturoituihin tietomalleihin, mikä mahdollistaa tietojen siirtymisen järjestelmien tai sovellusten välillä. Taloushallintoliiton [39] yhdessä Liikenne- ja viestintäministeriön käynnistämän TAL- TIO-hankkeen yhtenä tavoitteena oli saada taloustieto välittymään tietojärjestelmästä toiseen standardissa ja strukturoidussa muodossa. Tavoitteena on aluksi luoda kotimainen standardi, jonka käyttöä voitaisiin myöhemmin laajentaa myös muihin maihin EU:n sisällä. Taloushallintoliiton [39] mukaan standardi edesauttaa integraatioita tietojärjestelmien ja ohjelmistojen välille ja mahdollistaa pk-yritysten ja yhteisöjen digitaalisen talouden hallinnan standardin mukaisena tietovarastoratkaisuna. Tässä luvussa tarkastellaan eKuitti-ekosysteemin palveluita ja toimijoita sekä esitellään eKuitin strukturoitua tietomallia esimerkkien avulla.

eKuitti-ekosysteemin tarkoitus on tuottaa ostotietoja jäsennellyssä muodossa rivitasolla. Teknologiateollisuuden [41] määrityksien mukaan sähköiset ostotiedot siirretään reaaliajassa, jolloin ostotapahtumaan liittyvä maksutapahtuma välitetään oman kanavansa kautta kuittitietoja tarvitseville osapuolille. eKuitti ei ole maksutapahtuma, vaan eritelmä ostetuista ja maksetuista tuotteista ja palveluista. Finanssialan julkaisussa [16] eKuitti on konekielinen jäsennelty standardimuotoinen kuitti, missä tietokentät käsitellään automaattisesti ilman kuittitietojen manuaalista syöttämistä, tästä syystä paperi- tai PDF-kuittia ei pidetä eKuittina.

Siirryttäessä sähköisten kuittitietojen käsittelyyn tietojärjestelmissä, yritykset joutuvat arvioimaan käyttöönottoon liittyviä mahdollisuuksia ja uhkia. Cedillon ja muiden julkaisemassa artikkelissa [7] sähköisen tiedonsiirron käyttöönotossa tulisi huomioida seuraavat asiat:

- tuottavuus
- yhteensopivuus
- käytettävyys
- testattavuus

- näkyvyys.

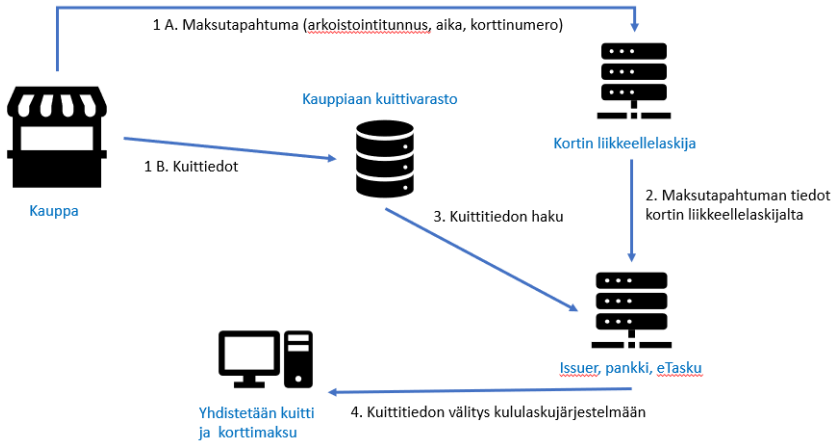
Tuottavuuden suhteen on arvioitava lisääkö ratkaisu tuottavuutta yrityksen eri prosesseissa. Sähköisellä kuittitietojen käsittelyllä manuaalisten virheiden määrä pienenee ja kuittitietojen käsittelyaika lyhenee. Yhteensopivuuden määrittelemisellä voidaan todentaa onko ratkaisu johdonmukainen jo olemassa olevan toimintamallin kanssa. Siirtyminen sähköisen kuittitietojen käsittelyyn tulee olla yksinkertaista, eikä toimintamalli saa olla liian vaikeasti ymmärrettävä tai käytettävä. Käyttöönoton kannalta testaamisella on tärkeä merkitys, jotta toiminnot voidaan todeta toimiviksi tuotannon mukaisessa ympäristössä, sekä hyödyntää testauksessa saadut palautteet käyttäjiltä. Yrityksen julkisen imagon ja näkyvyyden kannalta on tärkeää, ettei liiketoiminta häiriydy uusien prosessien käyttöönoton takia [7].

5.1 eKuitin ekosysteemi

eKuitin vastaanottajalla tulee olla oikeus valita, mitä yritystä kuittien vastaanottoon käytetään. Finanssiala [16] pitää tärkeänä, että kauppiaat voivat valita käyttämänsä kuittien välitysoperaattorin eKuittien välittämiseksi. Nelikulma-mallin mukaisesti sekä ostaja ja myyjä tekevät kumpikin omat sopimuksensa operaattorien kanssa.

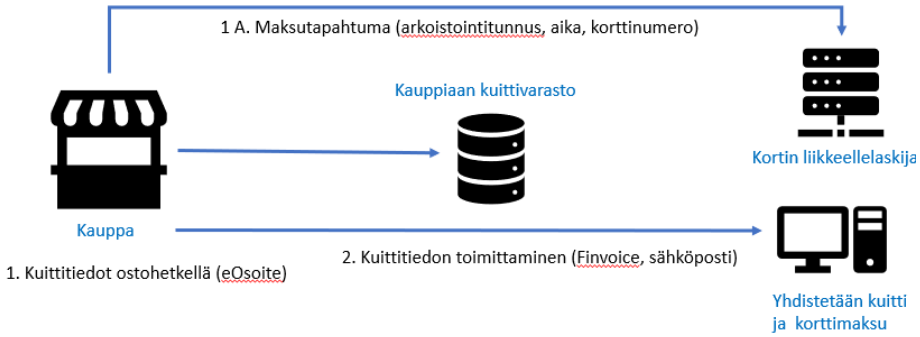
Monissa yrityksissä matkalaskun tekijä liittää sähköiset kuitit matkalaskuihin matkalaskujärjestelmässä, jolloin käyttäjän ei tarvitse enää jälkikäteen kuittitietoja käsitellä. Matkalaskujärjestelmissä tulee huomioida, ettei eKuitti-ostoja makseta työntekijän pankkitilille, jos maksu on tapahtunut yrityskortilla. Yrityksen kirjanpidossa eKuitit käsitellään ostolaskujen kierrätysjärjestelmässä kuten ostolaskut. eKuittien kierrätys on välttämätön, jotta oston muodostuneet kulut voidaan kohdistaa oikealla projektille tai kustannuspaikalle. Kierrätyksen jälkeen eKuitit tallennetaan sähköiseen arkistoon verkkolaskujen tapaan.

Teknologiateollisuuden [41] esittämässä mallissa kuvataan kaksi eri tapaa, miten eKuitti-tiedot saadaan välitettyä yrityksen kululaskujärjestelmään, sen mukaan onko maksukortin käyttäjällä eOsoite käytössä vai ei. eOsoite on yrityksien itse hallinnoima verkko-osoite tuote-, sopimus- tai kuittitietojen välitykseen. Yhtä yhteistä eOsoite-rekisteriä ei ole vielä olemassa, mutta Taloushallintoliitto [39] on esittänyt, että eOsoite-rekisteri voisi tulevaisuudessa korvata nykyisen verkkolaskurekisterin.



Kuva 5.1: Maksu myyntipisteellä - kaikki maksutavat ja eOsoite tiedetään ostohetkellä

Kuvassa 5.1 yrityksen maksukortin ostoon liittyvä maksutapahtuma välitetään kortin liikkeellelaskijan järjestelmään ja oston kuittitiedot kauppiaan kuittivaraston kautta käyttäjän valitseman eKuitti-palveluntarjoan järjestelmään. Palveluntarjoajan järjestelmästä maksu- ja kuittitiedot välitetään yrityksen kululaskujärjestelmään.

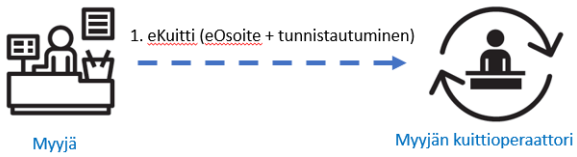


Kuva 5.2: Maksu maksukortilla myyntipisteellä - kuittitiedot haetaan erikseen arkistointitunnuksen perusteella

Ostoon liittyvä maksutapahtuma välitetään kortin liikkeellelaskijan järjestelmään kuvassa 5.2 ja oston kuittitiedot käyttäjän ilmoittaman eOsoitteen avulla yrityksen kululaskujärjestelmään, missä kuittitietoon yhdistetään kortin maksutapahtuma arkistointitunnuksen perusteella.

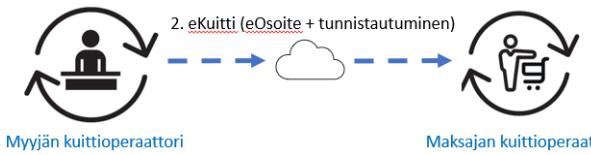
5.2 eKuitin nelikulma-malli

eKuitti-ekosysteemin toimintamalliksi suositellaan nelikulma-mallia. Mallissa maksaja ja myyjä tekevät kumpikin omat sopimuksensa kuittioperaattoriensa kanssa. Nelikulma-mallissa maksajan kuittitiedot välitetään ensin myyjän kuittioperaattorille, jonka jälkeen operaattori välittää kuittitiedot edelleen maksajan kuittioperaattorille. Maksaja kuittioperaattori vastaanottaa kuittitiedot ja välittää ne maksajan mobiilisovellukselle käyttäjätunnuksen perusteella. Nelikulma-mallin merkitys kasvaa, kun kauppaa käydään suurten kansainvälisten ja rajat ylittävien kauppiaiden kanssa. Finanssialan [16] tavoitteena on aikaansaada kaksisuuntainen sähköinen kuitti, jonka avulla myyjä voi ottaa yhteyttä asiakkaaseen maksun jälkeen, esimerkiksi verkkolinkin avulla. eKuitti-tietojen käsittelyn on oltava GDPR-asetuksen (General Data Protection Regulation) mukaista. GDPR on EU:n yleinen tietosuoja-asetus, joka otettiin käyttöön 25.5.2018 alkaen kaikissa EU:n jäsenmaissa. Tietosuoja-asetusta sovelletaan erityisesti henkilötietojen käsittelyyn, mutta yleiseen käyttöön muodostettu kuittidata ostoksista on anonymisoitua dataa. Asiakkaan yksilöidyt tiedot ovat hänen omassa hallinnassa ja asiakkaalla on oikeus hyödyntää, salata ja siirtää tietojaan palveluntarjoajalta toiselle [16].



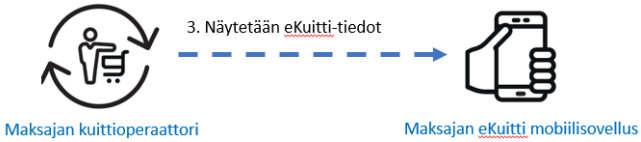
Kuva 5.3: eKuitti-tiedot myyjän kuittioperaattorille

Kuvassa 5.3 myyntipistelaite generoi sähköiset kuittitiedot ja lisää tietoihin maksajan kuittioperaattorin sähköisen osoitteen ja käyttäjätunnuksen, sekä toimittaa kuitin myyjän kuittioperaattorille.



Kuva 5.4: eKuitti-tiedot maksajan kuittioperaattorille

Seuraavassa vaiheessa kuvassa 5.4 myyjän kuittioperaattori vastaanottaa kuitin ja mikäli kuitti kuuluu toiselle kuittioperaattorille, lähettää operaattori kuittitiedot edelleen maksajan kuittioperaattorille.



Kuva 5.5: eKuitti-tiedot maksajan mobiilisovelluksessa

Kuvassa 5.5 maksajan kuittioperaattori vastaanottaa kuitin ja välittää kuittitiedot maksajan mobiilisovellukselle. Käyttäjän sovellus vastaanottaa kuittitiedot ja tallentaa ne käyttäjän tilille käyttäjätunnuksen perusteella. Tarvittaessa maksajan nimi ja osoite, sekä ostajan tunnistetiedot voidaan lisätä kuittitietoihin. Lopuksi käyttäjä näkee omat kuittitietonsa kuittidata-sovelluksessaan.

5.3 eKuitin strukturoitu tietomalli

eKuitin tietomalli on strukturoitu, mikä mahdollistaa eKuitin siirtämisen jäsennellyssä muodossa myyjän järjestelmästä ostajan käyttämään järjestelmään. Yhteisten toimintamallien ja standardoidun eKuitin tietomallin avulla kaikki toimijat voivat kehittää erilaisia palveluita yrityksille ja kuluttajille. Strukturoidussa eKuitissa liikkuva materiaali ei voi sisältää kuvia, mutta eKuitti voi sisältää linkin myyjän tarjoamiin palveluihin. Finanssialan [16] määrityksissä eKuitin tulee olla vakiomuotoinen lähtökohtana Finvoice-tietovaatimukset.

Finassialan Finvoice 3.0 soveltamisohjeessa [14] eKuitti on määritelty käyttäen XML-syntaksia. XML mahdollistaa eKuitin käsittelyn sovelluksen ymmärtämässä muodossa, selaimella eKuitti voidaan esittää loppukäyttäjälle paperikuittia vastaavassa muodossa. Selaimella esitetty kuitti voidaan tulostaa paperikuitiksi ja käsitellä

manuaalisesti. Finassialan Finvoice 3.0 soveltamisohjeessa [14] määritellään eKuittitietojen tarkemmmat tietoelementit. Myyjän ja ostajan tietoelementit käsitellään taulukoissa 5.1 ja 5.2, muiden osapuolten ja laskun tietoelementit taulukoissa 5.3 ja 5.4.

Myyjän tiedot lähetetään SellerPartyDetails-rakenteessa. Pakollisten laskutietojen lisäksi on välitettävä yhteystiedot, jos niitä vaaditaan. Taulukossa 5.1 esitetyt myyjän tiedot ovat pakollisia, jotta myyjä voidaan tunnistaa. Taloushallintoliitto [39] on esittänyt OVT-tunnuksen tilalle eOsoite-tunnusta, mutta yhtä yhteistä eOsoiterekisteriä ei ole vielä olemassa.

Taulukko 5.1: Myyjän tiedot (Seller information).

Data element	Explanation	Example
SellerOrganisation UnitNumber	Seller's OVT	003704904840
SellerParty Identifier	Seller's ID	0490484-0
SellerOrganisation TaxCode	Seller's VAT	FI04904840

Jos asiakas on tunnistettu, ostajatietojen on vastattava kuitin tietoja. Jos osto on tehty verkkokaupassa, ostajan osoite annetaan taulukon 5.2 DeliveryPartyDetails-rakenteessa ja toimituspäivämäärä DeliveryDetails/DeliveryDate-rakenteessa. Taulukossa 5.2 esitetyistä ostajan yhteystiedoista pakollisia ovat BuyerOrganisationName, BuyerPartyIdentifier ja BuyerCode Attribute, AgreementIdentifier ja BuyerOrganisationTaxCode.

Taulukko 5.2: Ostajan tiedot (Buyer information).

Data element	Explanation	Example
BuyerOrganisationName	Buyer's name	CARD
BuyerPartyIdentifier	Buyer's ID	
BuyerCode Attribute:IDType	Entity Id	
AgreementIdentifier	CustomerID(loyalty program)	0118283371
BuyerOrganisationTaxCode	Buyer's VAT	FI17897111
BuyerPhoneNumberIdentifier	Buyer's phone	09-123456
BuyerEmailaddressIdentifier	Buyer's email	
BuyerPostalAddressDetails	Buyer's postaddress	

Taulukon 5.3 AnyPartyDetails-rakenne sisältää tietoja kaupasta tai myyntipistelaitteesta. Muiden osapuolten tiedot eivät ole pakollisia, mutta niiden avulla voidaan myöhemmin tehdä asiakasvirta-analyyseja liitettynä kaupan (AnyPartyIdentifier) tai myyyntipistelaitteen (AnyPartySiteCode) paikkatietoihin.

Taulukko 5.3: Muiden osapuolten tiedot (Information on other parties).

Data element	Explanation	Example
AnyPartyText	Name of shop	Store 2
AnyPartyCode		Site
AnyPartyIdentifier	Shop's business ID	
AnyPartyOrganisationName	Organisation's name	Test Store
AnyPartyCommunicationDetails	Contact information	
AnyPartyPhoneNumberIdentifier	Phone number	09-5422 5422
AnyPartyStreetName	Street address	Streetname 11
AnyPartyTownName	City	LAHTI
AnyPartyPostCodeIdentifier	Postal code (ZIP)	15111
AnyPartySiteCode	Cash register number	Cash register 1

Taulukon 5.4 InvoiceDetails-rakenteessa välitetään kuitin ostorivit ja niiden arvonlisävero. Laskurivillä välitetään myös tieto maksutavasta, sekä mitä kortteja maksamiseen on käytetty. Summat ja arvonlisäverotiedot on ilmoitettava samassa valuutassa kuin lasku.

Taulukko 5.4: InvoiceDetails (laskun tiedot).

Data element	Explanation	Example
InvoiceTypeCode	Invoice type REC01	REC01
InvoiceTypeText	Info message text	e-Receipt
OriginCode	Type: original or copy	Original
InvoiceNumber	Invoice reference	45
InvoiceDate	Date of purchase	20170822
InvoicingPeriod Start/EndDate	Invoicing period	
SellerReference Identifier	Seller's reference	232199645
SellersBuyer Identifier	Customer ID(seller)	1012
PaymentCardInfo	Payment card info	
CardHolderName	Card holder name	

6 Web services -palvelun käyttö kuittitietojen välityksessä

Web services -teknologiaa käytetään asiakkaan ja pankin välisiin eräsiirtoaineistojen välitykseen. Asiakkaan ja pankin välinen liikenne salataan SSL-protokollan avulla. Asiakkaan tunnistaminen tehdään pankin julkaisemalla PKI-varmenteella. Web services -yhteyskäytäntöä käytetään myös muiden aineistojen käsittelyyn kuten viitemaksujen ja konekielisten tiliotteiden välittämiseen [31]. Tässä luvussa esitellään sähköisten kuittitietojen välityksessä käytettävää web services -palvelua ja kuvataan palvelun toteutusratkaisua käyttötapausesimerkkien avulla.

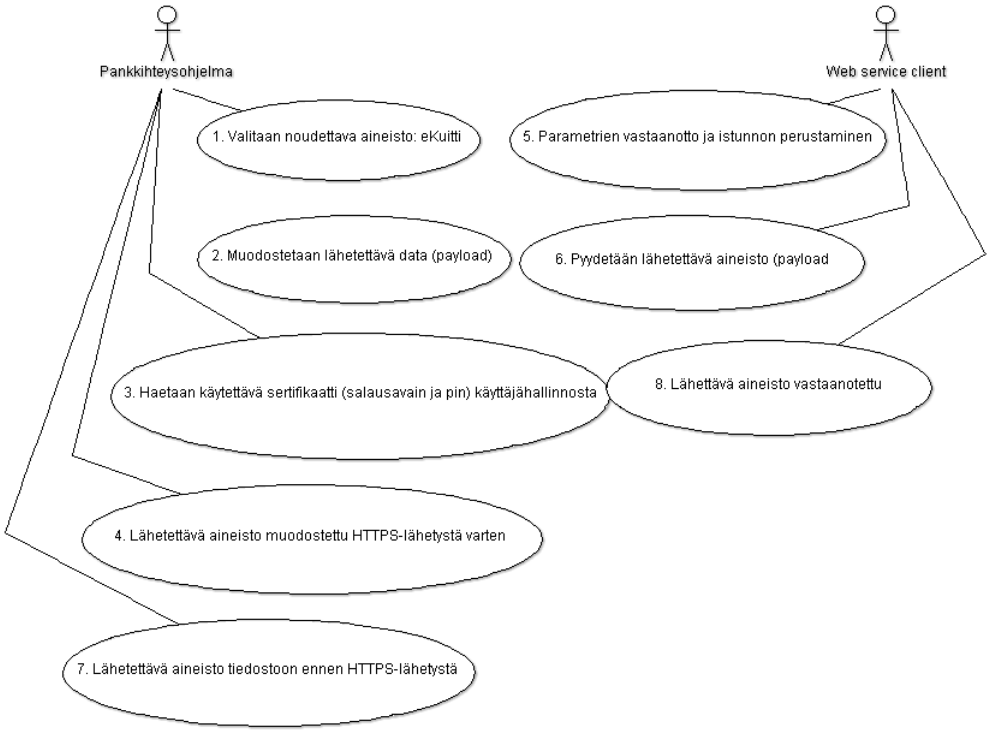
Yhteyden muodostuksessa asiakas on aktiivinen osapuoli. Kun aineistoja lähetetään pankkiin tai noudetaan pankista, asiakas avaa ensin yhteyden. Web services -yhteyskäytäntöä varten yrityksellä tulee olla sitä tukeva pankkiyhteysohjelmisto. Web services -yhteyden käytöstä tehdään sopimus (Web Services Agreement) asiakkaan ja pankin välillä. Sopimuksen perusteella asiakas voi noutaa pankin eräsiirtoaineistoja web services -yhteydellä käyttäen PKI-turvakäytäntöä. Palvelun käyttöoikeus tarkistetaan asiakkaan digitaalisesta allekirjoituksesta ApplicationRequest-sanomassa [31].

Pankin web services -palveluissa käytetään SOAP-pohjaista toteutusta, koska pankit, palvelun tarjoajina, määrittelevät itse rajapinnat eli kutsuttavat operaatiot. REST-pohjaisissa toteutuksissa lähetettävän datan sisältö on tärkeämpää, eikä operaatioilla ole valmiiksi määriteltyä muotoa, kuten SOAP:ssa message- ja envelope-elementit [40].

Web services -palvelussa pankin ja asiakkaan välinen aineistojen siirto kuvataan client-sovelluksen teknisessä kuvauksessa, WSDL-kuvaustiedostossa. WSDL on client-sovelluksen automaattista käsittelyä varten toteutettu konfigurointitiedosto ja sitä käytetään toiminnallisuuden rakentamiseen palvelimen kanssa. Suomessa pankeilla on yksi yhteinen WSDL-tiedosto, jota kaikki pankit käyttävät [31].

6.1 Lähetettävän aineiston muodostus

Ennen kuin web services -palvelua voidaan käyttää, asiakkaalla pitää olla voimassa oleva sopimus pankin kanssa web services -palvelun käytöstä. Asiakkaalla pitää olla sopimuksen teon yhteydessä pankista saatu käyttäjä- tai yrityskohtainen tunnus, jolla PKI-varmenne on ladattu pankista asiakkaan pankkiyhteysohjelmistoon. Varmenteeseen perustuva digitaalinen allekirjoitus, käyttäjän todennus ja oikeudet käyttää pankin tarjoamia palveluita tarkistetaan pankissa varmenteen perusteella [31].



Kuva 6.1: Lähetettävän aineiston muodostus käyttötapauskaaviona

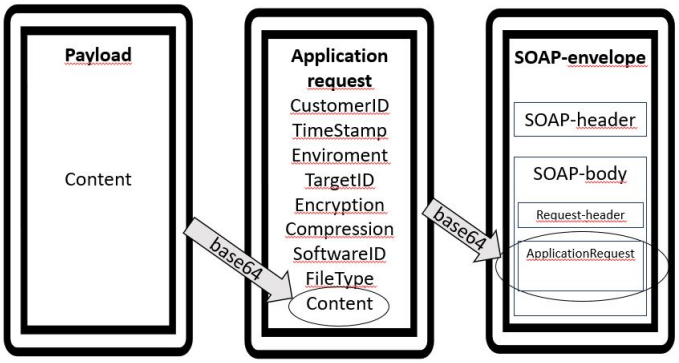
Kuvassa 6.1 kuvataan vaiheittain pankkiin lähetettävän aineiston muodostus pankkiyhteysohjelmassa. Vaiheessa 1 pankkiyhteysohjelman käyttöliittymässä valitaan mitä aineistoa pankista noudetaan. Aineistojen noutoon liittyy aina toimenpidepyyntö (Command), joka tässä tapauksessa on pyyntö noutaa aineisto (Downfile) pankista. Vaiheessa 2 muodostetaan pankkiin lähettävä aineisto, jota kutsutaan nimellä Payload. Payload on pankin kanssa sovittu palvelun tietosisältö ja se sisältää noudettavan aineiston parametrit pankin käsittelyä varten. Payloadin sisältö

on base64-koodattu, mikä estää sen näkymisen web services -kanavalla. Sisällöllä ei ole merkitystä käytetyn siirtokanavan suhteen, mikä mahdollistaa varakanavan käytön, jos asiakkaan tunnistaminen onnistuu ja valtuudet kyseisen kanavan käyttämiseen ovat voimassa.

Vaiheessa 3 pankkiyhteysohjelma hakee käyttäjähallinnosta käytettävän PKI-varmenteen ja PIN-koodin. Varmenne on suojattu asiakkaan antamalla PIN-koodilla, eikä varmennetta voida käyttää ilman PIN-koodia. Vaiheessa 4 pankkiyhteysohjelman muodostama aineisto jää odottamaan HTTPS-lähetystä pankkiin. Web services -client (ws-client) muodostaa istunnon pankin kanssa todentamalla palvelupyynnön käyttäjä- tai yrityskohtaisen varmenteen perusteella vaiheessa 5. Palvelupyynnössä asiakkaan eli aineiston allekirjoittajan täytyy vastata web services sopimuksen tehnyttä yritystä. Vaiheessa 6 istunnon perustamisen jälkeen, ws-client pyytää lähetettävän aineiston pankkiyhteysohjelmalta. Vaiheissa 7-8 pankkiyhteysohjelma lähettää aineiston (Payload) ws-clientille HTTPS-protokollaa käyttäen.

6.2 Lähetettävän aineiston salaus

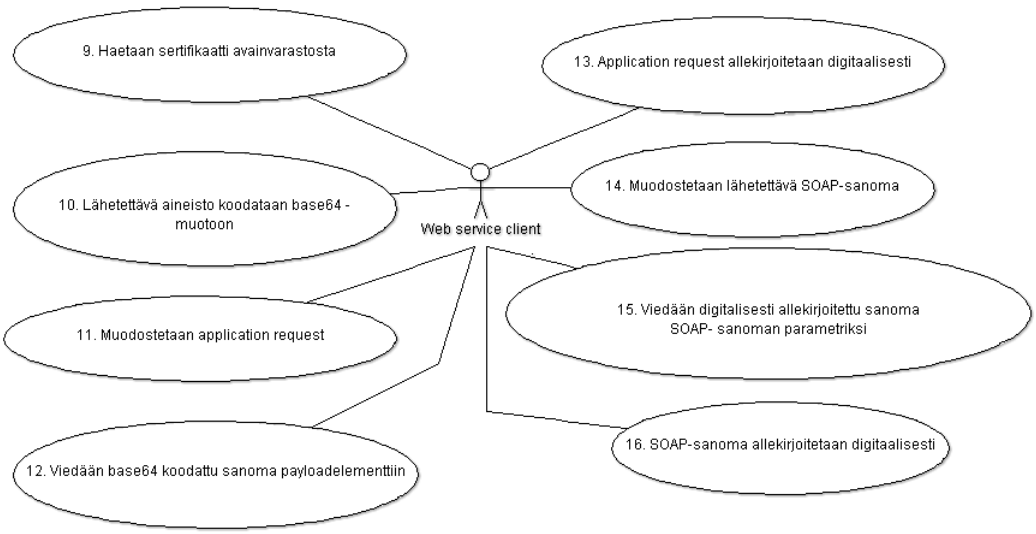
Varmenteella varmistetaan, että sanoman lähettäjä on oikea taho. Autentikoinnilla pyritään todentamaan myös se, että viestin allekirjoittajalla on lupa tai valtuudet käyttää pankin palvelua. Palvelun käyttöoikeus pyritään vahvistamaan tarkistamalla allekirjoitus ApplicationRequest-sanomasta. ApplicationRequest-sanomaa käytetään aineiston pyytämiseen pankista (DownloadFile). ApplicationRequest toimii myös kehyksenä lähetettäville tiedoille (Payload) [31].



Kuva 6.2: Lähetettävän aineiston ja SOAP-sanoman väliset suhteet

Kuvassa 6.2 kuvataan pankkiin lähetettävien SOAP-sanoman ja Payloadin vä-

lisiä riippuvuuksia. Allekirjoitettu ja base64-koodattu ApplicationRequest-sanoma kirjoitetaan SOAP-sanoman body-osaan, sen ApplicationRequest-kenttään. SOAP-sanoman rakenne on määritelty WSDL-dokumentissä ja se allekirjoitetaan digitaalisesti aineiston lähettäjän varmenteen yksityisellä avaimella. Avain voi olla sama kuin ApplicationRequest-sanomalla allekirjoitettu avain. Seuraavaksi SOAP-sanoma välitetään, web services -protokollaa käyttäen, pankin web services -palvelulle ja odotetaan vastausta.



Kuva 6.3: Lähetettävän aineiston salaus käyttötapauskaaviona

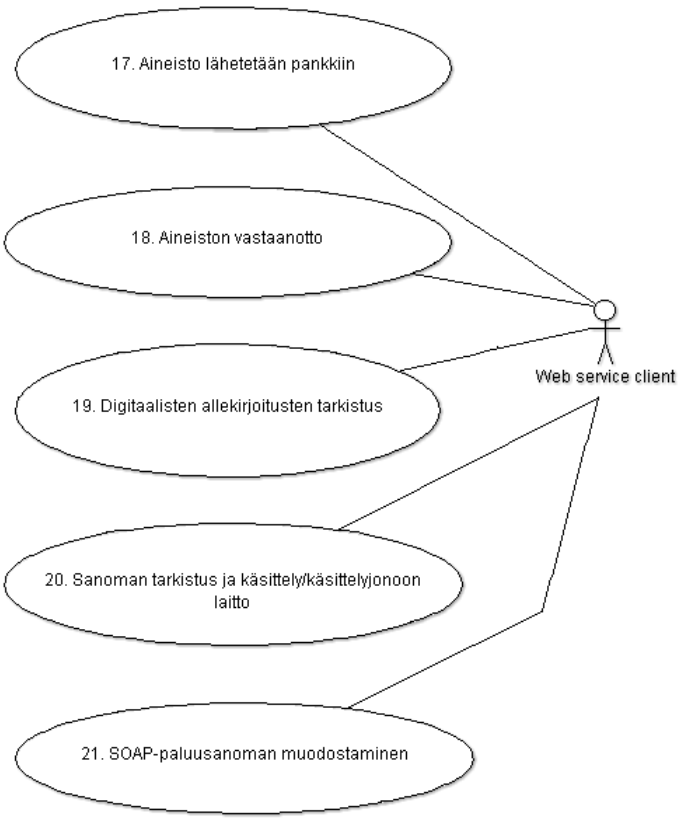
Kuvassa 6.3 kuvataan pankkiin lähettävän aineiston salaus ws-clientilla. Vaiheessa 9 ws-client hakee käytettävän sertifikaatin digitaalista allekirjoitusta varten. Ennen lähetystä aiemmin muodostettu Payload-aineisto koodataan base64-muotoon vaiheessa 10. Tämän jälkeen vaiheessa 11 ws-client muodostaa XML-muotoisen Application request-sanoman, jonka payload-nimiseen elementtiin Payload-sanoma viedään vaiheessa 12. Kun XML-muotoiseen sanoman elementtiin viedään toinen XML-muotoinen sanoma, sisempi sanoma koodataan base64-muotoon.

Vaiheessa 13 ApplicationRequest-sanoma allekirjoitetaan digitaalisesti käyttäjä- tai yrityskohtaisella varmenteella ja yksityisellä avaimella, samalla allekirjoitettu sanoma koodataan base64-muotoon. Seuraavaksi muodostetaan WSDL-dokumentin perusteella SOAP-sanoma, jonka body-elementtiin lisätään allekirjoitettu ja base64-koodattu sovelluspyyntö vaiheissa 14-15. Lopuksi SOAP-sanoma allekirjoitetaan digitaalisesti aineiston lähettäjän sertifikaatin yksityisellä avaimella. Allekirjoitus kir-

joitetaan SOAP-sanoman header-elementtiin vaiheessa 16.

6.3 Aineiston lähetys pankkiin

Aineiston lähetyksessä pankkiin on mukana digitaalisesti allekirjoitettu ApplicationRequest-sanoma, jossa on haluttu toimenpidepyyntö (Command). ApplicationRequest on allekirjoitettu sopimuksessa mainitun käyttäjän tai yrityksen varmenteen yksityisellä avaimella. Web services -client lähettää aineiston pankille web services-sanomana digitaalisella allekirjoituksella, jolla pankki tunnistaa ja todentaa asiakkaan [31].



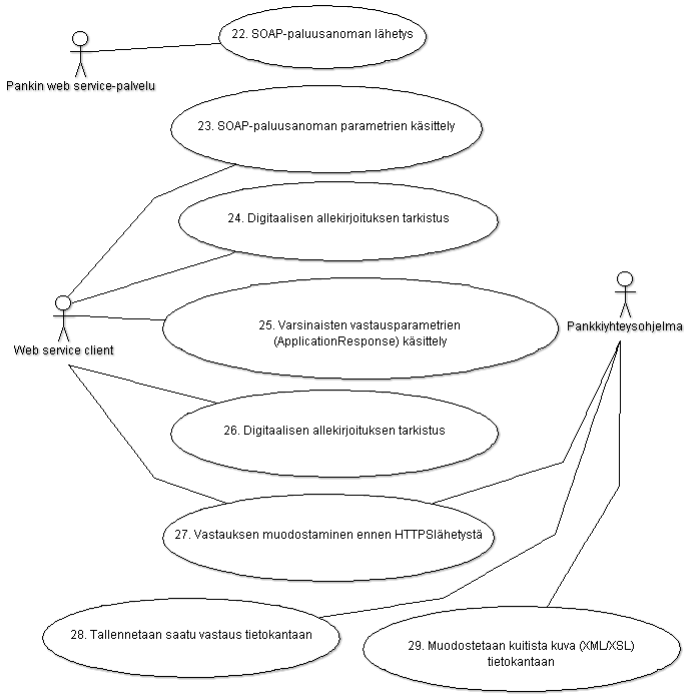
Kuva 6.4: Aineiston lähetys pankkiin käyttötapauskaaviona

Kuvassa 6.4 kuvataan vaiheittain aineiston lähetys pankkiin. Vaiheessa 17 aiemmin perustetussa istunnossa ws-client lähettää aineiston pankkiin ja jää odottamaan vastausta. Pankin web services -palvelu vastaanottaa aineiston ja tarkistaa digitaalisesta allekirjoituksesta lähettäjän oikeellisuuden vaiheissa 18-19. Välittömästi aineiston vastaanottamisen jälkeen, pankin web services -palvelu lähettää kuittauksen takaisin aineiston lähettäjälle. Jos asiakasta ei tunnisteta tai asiakkaalle ei ole oikeutta palvelun käyttöön, kuittauksessa palautetaan virhekoodi aineiston lähettäjälle. Jos asiakkaan todentaminen ja valtuutus ovat kunnossa, vaiheessa 20 vastaanotettu palvelupyyntö laitetaan käsittelyjonoon ja käsitellään pankin omissa tietojärjestelmissä.

Vaiheessa 21 pankin web services -palvelu muodostaa ja lähettää takaisin SOAP-paluusanoman, joka sisältää digitaalisesti allekirjoitetun AppliationResponse-sanoman. Pyydetty aineisto asetetaan ApplicationResponse-sanoman Content-kentäään base64-koodattuna. Jos pyydettyä aineistoa ei ole saatavilla, ApplicatioResponse-sanoma sisältää virheilmoituksen.

6.4 Aineiston vastaanotto

Pankki allekirjoittaa ApplicationResponse-sanoman muodostaessaan vastauksen asiakkaalle, joten asiakas voi tarkistaa, että sanoma on tullut sovitulta osapuolelta ja ettei se ole muuttunut allekirjoituksen jälkeen. Vastauksen sisältö noudattaa ApplicationResponse-sanomaa, joka on määritelty pankkien yhteisessä web services -kuvauksessa [31].



Kuva 6.5: Aineiston vastaanotto käyttötapauskaaviona

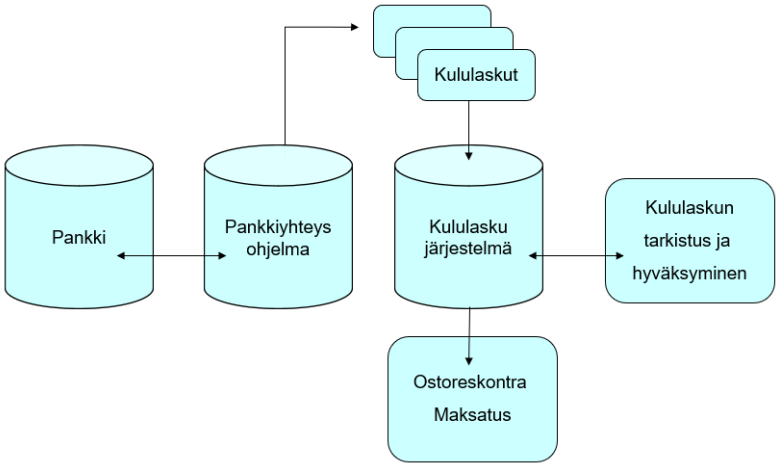
Kuvassa 6.5 kuvataan vaiheittain aineiston vastaanotto ws-clientilla. Vaiheissa 22-24 ws-client vastaanottaa pankin lähettämän SOAP-paluusanoman ja tarkistaa sen digitaalisen allekirjoituksen. Tarkistamisella varmistetaan, ettei ulkopuolinen taho ole muokannut viestiä tai allekirjoitusta. Seuraavaksi ws-client tarkistaa ApplicationResponse-sanoman digitaalisen allekirjoituksen vaiheissa 25-26. Tarkistuksella todennetaan, että sanoma on tullut oikealta osapuolelta, eikä tieto ole muuttunut siirron aikana. ApplicationResponse-sanomaa käytetään aineiston vastaanottoon pankista (DownloadFile). ApplicationResponse toimii kehyksenä vastaanotettaville tiedoille (Payload). Vaiheessa 27 ws-client lähettää ApplicationResponse-sanoman Contens-kentässä saadun aineiston pankkiyhteysohjelmalle HTTPS-protokollaa käyttäen. Lopuksi pankkiyhteysohjelma tallentaa saadun datan sen käyttämään tietokantaan vaiheissa 28-29.

6.5 Sähköinen kuittiaineisto kululaskujärjestelmässä

Sähköisten kuittien käyttöönotto aiheuttaa muutoksia yrityksen tietojärjestelmiin ja prosesseihin. Kuitin käsittely kululaskujärjestelmässä eroaa sen mukaan, käyttää-

kö yrityksen työntekijä omaa vai yrityksen maksukorttia. Maksukorteilla ostetaan yleensä työmatkoihin liittyviä pienempiä hankintoja tai edustamiseen liittyviä tuotteita tai palveluita. Korttiostot voivat olla myös hankintoja yrityksen tuotantoa varten, jolloin kuittien sisältö tarvitaan oston tarkastus- ja hyväksymisprosesseja varten.

Kun sähköinen kuittiaineisto on noudettu pankista, kululaskut tarkastetaan ja hyväksytään maksettaviksi korvauksina kululaskun laatijalle kululaskujärjestelmässä. Kululaskujärjestelmässä on valmiiksi kiinteitä tietoja, joita ovat henkilötiedot, organisaatio ja hyväksymisoikeudet. Ennen kululaskun hyväksymistä, kululaskun laatija voi tarvittaessa täydentää kululaskun seurantaan liittyviä tietoja kuten kustannuspaikka-tiedot. Kustannuspaikkoja käytetään kustannusten hallinnassa ja sen avulla seurataan yrityksen toiminnan tehokkuutta ja tuloksellisuutta.



Kuva 6.6: Sähköinen kuittiaineisto kululaskujärjestelmässä

Kuvassa 6.6 esimies tai muu vastaava henkilö saa sähköpostiinsa tiedon hyväksymättömistä kululaskuista kululaskujärjestelmässä. Jos kululaskun hyväksyminen vaatii useamman hyväksyjän, kukin hyväksyjä saa tiedon tapahtumasta sähköpostiinsa. Kululasku voidaan hyväksyä myös rivikohtaisesti, jolloin esimies hyväksyy vain ne rivit, jotka ovat hänen vastuulla. Kun kaikki kululaskun rivit ovat hyväksytty, lasku siirtyy ostoreskontraan maksettavaksi korvauksena kululaskun laatijalle.

6.6 Tulosten analyysiä

Sähköisten kuittitietojen välityksessä web services -teknologia on kompleksinen monitasoisen sanomarakenteen takia. Peruslogiikka aineistojen käsittelyssä ei poikkea eri pankkien välillä, mutta sanomien muodostussäännöt eroavat tosistaan. Pankit tarjoavat kattavaa dokumentaatiota ohjelmistotaloille web services -palveluiden käytöstä, mikä on helpottanut niiden integrointia pankkiyhteysohjelmistoihin. Pankkien dokumentaatiossa on kuitenkin puutteita ja erojakin eri pankkien välillä, osasta dokumenteista ei löytynyt tarkkoja ohjeita käytetyn web services -palvelupyynnön vaatimista elementeistä. Välitettävälle aineistolle tehtävät base64-koodaukset kasvattivat sanomien kokoa merkittävästi, koska aineistojen pakkausta ei juurikaan käytetä pankkien web services -palveluissa.

Työssä tutkittiin web services ja XML -tekniikoita, joihin perustuen palvelun toteutusta kuvattiin. Web services -palvelun toteuttaminen vaati tutustumista myös XML-dokumenttien digitaalisiin allekirjoituksiin. Pankin SOAP-sanomien ja palvelupyyntöjen aitous ja muuttumattomuus varmistetaan digitaalisella allekirjoituksella. Allekirjoitukset todennetaan varmenteella eli julkisella avaimella, joiden jakelun pankit järjestävät itsenäisesti.

Koska pankin web services -palvelimen ja yrityksen pankkiyhteysohjelmiston välinen keskustelu perustuu XML-pohjaisiin SOAP-sanomiin, työssä tutkittiin palvelun toteuttamista erityisesti SOAP-protokollan kannalta. Pankkien SOAP-pohjaisissa palveluissa viestit ovat aina SOAP-viestejä ja palveluista on saatavilla WSDL-dokumentti. Dokumentti on tekninen kuvaus pankin web services -palvelun käytöstä ja sen XML-muoto mahdollistaa dokumentin käsittelyn pankkiyhteysohjelmistossa. WSDL-dokumentin käsittelyä helpotti myös se, että dokumentti on yhteinen kaikilla pankeilla.

Tutkielmassa saatiin vastaukset teoriaosuudessa esitettyihin tutkimuskysymyksiin ja työn tavoitteet saavutettiin osittain, koska selvitystä ei tehty kaikkien pankkien osalta. Vallitsevasta pandemiatilanteesta ja aikatauluongelmista johtuen tutkielman empiirinen osuus sisälsi web services -palvelun esittelyn ja toteutusratkaisun kuvauksen, jonka tuloksia tullaan hyödyntämään myöhemmin pankkiyhteysohjelmiston jatkokehityksessä.

7 Yhteenveto

Pro gradu -tutkielman tavoitteena oli selvittää, miten web services -palvelu toteutetaan yrityksen pankkiyhteysohjelmiston ja pankin välille. Tutkielman teoriaosuudessa tutkittiin web services- ja XML-tekniikoita, joiden pohjalta palvelu toteutettaisiin. Web services -palvelun toteutuksessa tutkittiin erityisesti SOAP-sanomien käyttöä, koska se standardoitu web services -yhteyden sanomamuoto pankkiyhteyskäytössä. Empiirisessä osassa esitellyn web services -palvelun toteutus vaati tutustumista myös XML-salauksiin ja XML-dokumenttien digitaalisiin allekirjoituksiin.

Web services -yhteyskäytäntö mahdollistaa juostavan ohjelmistokehityksen pankkiyhteysohjelmistojen tekijöille ja se tarjoaa uusia mahdollisuuksia pankin tarjoamissa palveluissa sanomien käytön laajennusmahdollisuuksien johdosta, esimerkkinä sähköisten kuittitietojen hyödyntäminen. Sähköisten kuittitietojen käsittely osana eKuitti-ekosysteemiä mahdollistaa yrityksille mallin, missä strukturoitu sähköinen kuitti korvaa paperisten kuittien käsittelyn ja samalla kuittitietojen manuaalinen tallentaminen yritysten tietojärjestelmiin tulee tarpeettomaksi. Myös kotitaloudet hyötyvät eKuitin käyttöönotosta, koska kuluttajat voivat käsitellä ostoksien kuitit sähköisesti, esimerkiksi takuuasioissa, palveluntarjoajien kuittipalveluissa.

Yritysten matkalaskujen käsittelyssä sähköiset kuittitiedot ovat käytettävissä sähköisistä kuittiarkistosta, matkalaskuihin ei tarvitse liittää paperikuitteja tai niiden kuvia. Sähköisten kuittien mukana tulevan datan jatkohyödyntämiselle on kasvava tarve yrityksien eri liiketoimintaprosesseissa. Tietoa voidaan välittää esimerkiksi vakuutusyhtiön sähköiseen korvaushakemukseen. Myös Verohallinnolla on suunnitelmia sähköisten kuittien osto- ja myyntitietojen hyödyntämisestä yritysten arvonlisäveroraportoinnissa. Haagin ja muiden [18] mukaan siirtyminen sähköisten kuittitietojen käsittelyyn vaatii yrityksien tietojärjestelmien uudistamista sekä muutoksia myös organisaation liiketoimintaprosesseihin. Uudet toimintatavat ja käytänteet vaativat henkilöstön kouluttamista ja uusien työtehtävien määrittelyä. Haag ja muut [18] arvioivat digitalisoinnin tuovan yrityksille merkittäviä taloudellisia hyötyjä ja siksi kannustavat siirtymistä sähköisen laskutukseen ja sähköisten kuittitietojen hyödyntämiseen organisaation eri liiketoimintaprosesseissa.

Lähteet

[1] BARTEL, M., BOYER, J., FOX, B., LAMACCHIA, B., JA SIMON, E. XML-signature syntax and processing, w3c recommendation. URL http://www.w3.org/TR/2002/REC-xmldsig-core-20020212, viitattu 12.5.2020.

[2] BEHESHTI, H. M. What managers should know about ERP/ERP II. *Management Research News 29* (2006), 184–193.

[3] BERNSTEIN, P. A. *Middleware: An Architecture for Distributed System Services*. Digital Equipment Corporation Cambridge Research Lab, Cambridge, U.S.A., 1993.

[4] BEZNOSOV, K., FLINN, D. J., KAWAMOTO, S., JA HARTMAN, B. Introduction to web services and their security. *Information Security Technical report 10* (2005), 2–14.

[5] BIRELL, A. D., JA NELSON, B. J. Implementing remote procedure calls. *ACM Transactions on Computer Systems 2* (1984), 39–59.

[6] BRITTENHAM, P., CUBERA, F., EHNEBUSKE, D., JA GRAHAM, S. Understanding WSDL in a UDDI registry. URL https://www.ibm.com/developerworks/webservices/library/ws-wsdl/index.html, viitattu 13.4.2020.

[7] CEDILLO, P., GARCÍA, A., CÁRDENAS, J. D., JA BERMEO, A. A systematic literature review of electronic invoicing, platforms and notification systems. Julkaisusarjassa *2018 International Conference on eDemocracy eGovernment* (Ambato, Ecuador, April 2018), IEEE, 150–157.

[8] CURPHEY, M. Web services: Developers dream or hackers heaven? *Information Security Technical report 10* (2005), 228–235.

[9] DOURNAEE, B. *XML Security*. McGraw-Hill, New York, U.S.A., 2002.

[10] FENG, X., SHEN, J., JA FAN, Y. Rest: An alternative to RPC for web services architecture. Julkaisusarjassa *2009 First International Conference on Future Information Networks* (Beijing, China, November 2009), IEEE, 7–10.

[11] FENSEL, D., LAUSEN, H., POLLERES, A., DE BRUIJN, J., STOLLBERG, M., ROMAN, D., JA DOMINGUE, J. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag, Berlin, Germany, 2007.

[12] FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, U.S.A., January 2000.

[13] FIELDING, R. T., JA TAYLOR, R. N. Principled design of the modern web architecture. *ACM Transactions on Internet Technology 2* (2002).

[14] FINANSSIALA RY. Finvoice 3.0 soveltamisohje. URL `https://www.finanssiala.fi/finvoice/dokumentit/Finvoice_3_0_soveltamisohje.pdf`, viitattu 22.11.2019.

[15] FINANSSIALA RY. Security and message specification for financial messages using web services. URL `https://www.finanssiala.fi/maksujenvalitys/dokumentit/WebServices_Messages_v110_20200504.pdf`, viitattu 24.9.2020.

[16] FINANSSIALA RY. Transmitting card purchase receipts in the finvoice format. URL `https://www.finanssiala.fi/finvoice/dokumentit/Transmitting_card_purchase_receipts_in_the_finvoice_format.pdf`, viitattu 22.11.2019.

[17] FREDRICH, T. RESTful service best practices. best-practices document. URL `https://s3.amazonaws.com/tfpearsonecollege/bestpractices/RESTful+Best+Practices.pdf`, viitattu 20.9.2020.

[18] HAAG, S., BORN, F., KREUZER, S., JA BERNIUS, S. Organizational resistance to e-invoicing results from an empirical investigation among SMEs. Julkaisusarjassa *12th International Conference on Electronic Government* (Koblenz, Germany, September 2013), IEEE, 286–297.

[19] HOGG, K., CHILCOTT, P., NOLAN, M., JA SRINIVASAN, B. An evaluation of web services in the design of a b2b application. Julkaisusarjassa *27th Australian Computer Science Confrence* (Dunedin, New Zealand, 2004), 331–340.

[20] IMAMURA, T., DILLAWAY, B., SIMON, E., YIU, K., JA NYSTROM, M. XML encryption syntax and processing version 1.1. w3c recommendation 11 april 2013. URL https://www.w3.org/TR/xmlenc-core1/Overview_diff_rec.html, viitattu 8.5.2020.

[21] JÄRVINEN, J. *Hajautetut verkkopalvelut*. Docendo Finland Oy, Jyväskylä, 2002.

[22] JÄRVINEN, P. *Salausmenetelmät*. Docendo Finland Oy, Jyväskylä, 2003.

[23] KALIN, M. *Java Web Services: Up and Running*. O'Reilly Media Inc., Sebastopol, U.S.A., 2009.

[24] KEARNEY, P. Message level security for web services. *Information Security Technical report 10* (2005), 41–50.

[25] KLEINER, E., JA ROSCOE, A. W. On the relationship between web services security and traditional protocols. *Electronic Notes in Theoretical Computer Science 155* (2006), 583–603.

[26] KREGER, H. *Web Services Conceptual Architecture (WSCA 1.0)*. Tekninen raportti, IBM Software Group, January 2001.

[27] LUO, M., ENDREI, M., COMTE, P., KROGDAHL, P., ANG, J., NEWLING, T., ARSANJANI, A., JA CHUA, S. *IBM: Patterns: Service-oriented Architecture and Web Services*. IBM, New York, U.S.A., 2004.

[28] MACPHEE, A., JA ONEILL, M. Notes from the field: Implementing a security solution for web services. *Information Security Technical report 10* (2005), 25–32.

[29] MCLAUGHLIN, B. *Java and XML Data Binding*. O'Reilly Media Inc., Sebastopol, U.S.A., 2002.

[30] NEWCOMER, E. *Understanding Web Services : XML, WSDL, SOAP and UDDI*. Pearson, Indianapolis, U.S.A., 2002.

[31] NORDEA BANK OYJ. Web services security and communication. URL https://www.nordea.fi/Images/146-163432/web-services-security-and-communication-description.pdf, viitattu 10.5.2020.

[32] OY SAMLINK AB. Samlink Customer CA Varmennuskäytäntö. URL https://www.samlink.fi/wp-content/uploads/2015/12/Varmennuska%CC%88yta%CC%88nto%CC%88_pdf.pdf, viitattu 27.06.2020.

[33] PAPAZOGLOU, M. P. *Web Services: Principles and Technology*. Pearson Education Limited, Essex, U.K., 2008.

[34] PENTTINEN, E. Electronic invoicing as a platform for exchanging accounting information. Julkaisusarjassa *Aalto-yliopiston Kauppakorkeakoulun julkaisuja* (2010), Aalto-Print.

[35] RESNICK, S., CRANE, R., JA BOWEN, C. *Essential Windows Communication Foundation (WCF): For .NET Framework 3.5*. Addison-Wesley, Upper Saddle River, U.S.A., 2008.

[36] RICHARDSON, L., AMUNDSEN, M., JA RUBY, S. *RESTful Web APIs: Services for a Changing World*. O'Reilly Media Inc., Sebastopol, U.S.A., 2013.

[37] SKONNARD, A. Understanding SOAP. URL `https://msdn.microsoft.com/en-us/library/ms995800.aspx`, viitattu 01.04.2020.

[38] SNELL, J., TIDWELL, D., JA KULCHENKO, P. *Programming Web services with SOAP*. O'Reilly Media Inc., Sebastopol, U.S.A., 2002.

[39] SUOMEN TALOUSHALLINTOLIITTO RY. Taltio-hankkeen loppuraportti. URL `https://taloushallintoliitto.fi/sites/default/files/dokumentit/page/fields/field_related_attachments/taltio-hankkeen_loppuraportti_yleinen_31.10.2017.pdf`, viitattu 22.11.2019.

[40] TAMPEREEN TEKNILLINEN YLIOPISTO. Web-palveluiden toteutustekniikat ohj-5201. URL `http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/4.pdf`, viitattu 20.9.2020.

[41] TEKNOLOGIATEOLLISUUS RY. eReceipt guidelines. URL `https://teknologiateollisuus.fi/sites/default/files/file_attachments/2018_ekuitti_eng_sisus_vedos_6.pdf`, viitattu 22.11.2019.

[42] TRIVEDI, R., WHITNEY, D., GALBRAITH, B., V, P. D., JANAKIRAMAN, M., HIOTIS, A., JA HANKISON, W. *Professional Web Services Security*. WROX Press, Birmingham, U.K., 2002.

[43] TUIKKA, T., JA KANALA, S. *XML-Ohjelmoinnin perusteet*. Edita Publishing Oy, Helsinki, 2001.

[44] TUTORIALS POINT. SOAP tutorial. URL https://www.tutorialspoint.com/soap/soap_fault.htm, viitattu 13.09.2020.

[45] TÄHTINEN, S. *Järjestelmäintegraatio*. Talentum Oyj, Helsinki, 2005.

[46] W3SCHOOL. XML tutorial. URL http://www.w3schools.com/xml/default.asp, viitattu 20.06.2020.

[47] WORLD WIDE WEB CONSORTIUM. Web services description language WSDL. URL https://www.w3.org/TR/wsdl20/, viitattu 02.06.2020.

[48] WORLD WIDE WEB CONSORTIUM. XML signature syntax and processing. URL http://www.w3.org/TR/xmldsig-core/, viitattu 02.06.2020.

[49] WORLD WIDE WEB CONSORTIUM W3C. Simple object access protocol. URL http://www.w3.org/TR/2000/NOTE-SOAP-20000508, viitattu 10.5.2020.

[50] YU, W., SUPTHAWEESUK, P., JA ARAVIND, D. Trustworthy Web Services Based on Testing. Julkaisusarjassa *Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering* (Shanghai, China, June 2005), IEEE.

A Liite 1: eKuitti xml-tiedosto

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<?xml-stylesheet href="Finvoice.xsl" type="text/xsl"?>
<Finvoice Version="3.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Finvoice3.0.xsd">
<MessageTransmissionDetails>
<MessageSenderDetails>
<FromIdentifier>FI04904840</FromIdentifier>
<FromIntermediator>NONE</FromIntermediator>
</MessageSenderDetails>
<MessageReceiverDetails>
<ToIdentifier>Findity</ToIdentifier>
<ToIntermediator>NONE</ToIntermediator>
</MessageReceiverDetails>
<MessageDetails>
<MessageIdentifier>ElectronicReceipt</MessageIdentifier>
<MessageTimeStamp>2018-06-01T22:30:47Z</MessageTimeStamp>
<ImplementationCode>ECR1</ImplementationCode>
</MessageDetails>
</MessageTransmissionDetails>
<SellerPartyDetails>
<SellerPartyIdentifier>0490484-0</SellerPartyIdentifier>
<SellerOrganisationName>Oy Yritys Ab</SellerOrganisationName>
<SellerOrganisationTaxCode>FI04904840</SellerOrganisationTaxCode>
<SellerPostalAddressDetails>
<SellerStreetName>KESKUSKATU 12 C</SellerStreetName>
<SellerTownName>HÄMEENLINNA</SellerTownName>
<SellerPostCodeIdentifier>13300</SellerPostCodeIdentifier>
</SellerPostalAddressDetails>
</SellerPartyDetails>
<SellerOrganisationUnitNumber>003704904840</SellerOrganisationUnitNumber>
<BuyerPartyDetails>
<BuyerOrganisationName>CARD</BuyerOrganisationName>
</BuyerPartyDetails>
<AnyPartyDetails>
<AnyPartyText AnyPartyCode="Site">Ketju1</AnyPartyText>
<AnyPartyOrganisationName>Myymälä 123</AnyPartyOrganisationName>
<AnyPartyCommunicationDetails>
<AnyPartyPhoneNumberIdentifier>03-5422 5422</AnyPartyPhoneNumberIdentifier>
</AnyPartyCommunicationDetails>
<AnyPartyPostalAddressDetails>
<AnyPartyStreetName>Kirkkokatu 23</AnyPartyStreetName>
<AnyPartyTownName>HOLLOLA</AnyPartyTownName>
<AnyPartyPostCodeIdentifier>15111</AnyPartyPostCodeIdentifier>
</AnyPartyPostalAddressDetails>
<AnyPartySiteCode>Kassa 1</AnyPartySiteCode>
</AnyPartyDetails>
<InvoiceDetails>
<InvoiceTypeCode>REC01</InvoiceTypeCode>
<InvoiceTypeCodeUN>632</InvoiceTypeCodeUN>
<InvoiceTypeText>Sähköinen kuitti</InvoiceTypeText>
<OriginCode>Original</OriginCode>
<InvoiceNumber>45</InvoiceNumber>
<InvoiceDate Format="CCYYMMDD">20180522</InvoiceDate>
<SellerReferenceIdentifier>232199645</SellerReferenceIdentifier>
<SellersBuyerIdentifier>1012</SellersBuyerIdentifier>
<AgreementIdentifier>0118283371</AgreementIdentifier>
<InvoiceTotalVatExcludedAmount AmountCurrencyIdentifier="EUR">706,37</
InvoiceTotalVatExcludedAmount>
<InvoiceTotalVatAmount AmountCurrencyIdentifier="EUR">49,53</InvoiceTotalVatAmount>
<InvoiceTotalVatIncludedAmount AmountCurrencyIdentifier="EUR">755,90</
InvoiceTotalVatIncludedAmount>
<VatSpecificationDetails>
```

```
<VatBaseAmount AmountCurrencyIdentifier="EUR">206,37</VatBaseAmount>
<VatRatePercent>24,00</VatRatePercent>
<VatCode>S</VatCode>
<VatRateAmount AmountCurrencyIdentifier="EUR">49,53</VatRateAmount>
</VatSpecificationDetails>
<VatSpecificationDetails>
<VatBaseAmount AmountCurrencyIdentifier="EUR">500,00</VatBaseAmount>
<VatRatePercent>0,00</VatRatePercent>
<VatCode>O</VatCode>
<VatRateAmount AmountCurrencyIdentifier="EUR">0,00</VatRateAmount>
</VatSpecificationDetails>
</InvoiceDetails>
<PaymentCardInfo>
<PrimaryAccountNumber>524342xxxxxx1401</PrimaryAccountNumber>
</PaymentCardInfo>
<PaymentStatusDetails>
<PaymentStatusCode>PAID</PaymentStatusCode>
</PaymentStatusDetails>
<InvoiceRow>
<ArticleIdentifier>123456</ArticleIdentifier>
<ArticleName>HILAVITKUTIN</ArticleName>
<InvoicedQuantity QuantityUnitCode="M2">1.0000</InvoicedQuantity>
<UnitPriceAmount AmountCurrencyIdentifier="EUR">7,98</UnitPriceAmount>
<RowVatRatePercent>24,00</RowVatRatePercent>
<RowVatCode>S</RowVatCode>
<RowVatAmount AmountCurrencyIdentifier="EUR">1,92</RowVatAmount>
<RowVatExcludedAmount AmountCurrencyIdentifier="EUR">7,98</RowVatExcludedAmount>
<RowAmount AmountCurrencyIdentifier="EUR">9,90</RowAmount>
</InvoiceRow>
<InvoiceRow>
<ArticleIdentifier>234567</ArticleIdentifier>
<ArticleName>RUUVIMEISSELI</ArticleName>
<EanCode>0075678164125</EanCode>
<InvoicedQuantity QuantityUnitCode="KPL">2.0000</InvoicedQuantity>
<UnitPriceAmount AmountCurrencyIdentifier="EUR">99,19</UnitPriceAmount>
<RowVatRatePercent>24,00</RowVatRatePercent>
<RowVatCode>S</RowVatCode>
<RowVatAmount AmountCurrencyIdentifier="EUR">47,61</RowVatAmount>
<RowVatExcludedAmount AmountCurrencyIdentifier="EUR">198,39</RowVatExcludedAmount>
<RowAmount AmountCurrencyIdentifier="EUR">246,00</RowAmount>
</InvoiceRow>
<InvoiceRow>
<ArticleIdentifier>888888</ArticleIdentifier>
<ArticleName>PORAKONE</ArticleName>
<InvoicedQuantity QuantityUnitCode="KPL">1.0000</InvoicedQuantity>
<UnitPriceAmount AmountCurrencyIdentifier="EUR">500,00</UnitPriceAmount>
<RowVatRatePercent>0,00</RowVatRatePercent>
<RowVatCode>O</RowVatCode>
<RowVatAmount AmountCurrencyIdentifier="EUR">0,00</RowVatAmount>
<RowVatExcludedAmount AmountCurrencyIdentifier="EUR">500,00</RowVatExcludedAmount>
<RowAmount AmountCurrencyIdentifier="EUR">500,00</RowAmount>
</InvoiceRow>
<InvoiceRow>
<SubInvoiceRow>
<SubIdentifier>PAYMENT</SubIdentifier>
<SubArticleIdentifier>Kortti</SubArticleIdentifier>
<SubArticleName>MAKSUTAPA</SubArticleName>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00000">CardMaskedNumber</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>524342xxxxxx1401</SubRowDefinitionValue>
</SubRowDefinitionDetails>
```

```
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00001">ReferenceNumber</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>151222010038</SubRowDefinitionValue>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00002">TimeStamp</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>151222100715</SubRowDefinitionValue>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00003">RequestedAmount</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>755,90</SubRowDefinitionValue>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00004">AuthorizationCode</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue/>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00005">MerchantNumber</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue/>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00006">AuthorizingTermID</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue/>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00007">VerifiedByPINFlag</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue/>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00008">TenderAuthorizationMethodType</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>ChipPin</SubRowDefinitionValue>
</SubRowDefinitionDetails>
<SubRowDefinitionDetails>
<SubRowDefinitionHeaderText DefinitionCode="CARD00009">CreditCardCompanyCode</
SubRowDefinitionHeaderText>
<SubRowDefinitionValue>L5</SubRowDefinitionValue>
</SubRowDefinitionDetails>
<SubRowAmount AmountCurrencyIdentifier="EUR">755,90</SubRowAmount>
</SubInvoiceRow>
</InvoiceRow>
<SpecificationDetails>
<SpecificationFreeText>KORTTITAPAHTUMA</SpecificationFreeText>
<SpecificationFreeText/>
<SpecificationFreeText>Kortti:                          MC Debit</SpecificationFreeText>
<SpecificationFreeText>**** **** **** 1401 CP</SpecificationFreeText>
<SpecificationFreeText>Sovellus:              A0000000004101001</SpecificationFreeText>
<SpecificationFreeText>Tap.nro/Varmennus:         00038/179143</SpecificationFreeText>
<SpecificationFreeText>Yritys/Ala:           111111111111/5399</SpecificationFreeText>
<SpecificationFreeText/>
<SpecificationFreeText>Autentisointi:         03072B69B36642F7</SpecificationFreeText>
<SpecificationFreeText>Viite:                     151222010038</SpecificationFreeText>
<SpecificationFreeText/>
<SpecificationFreeText>Debit/Veloitus               755,90 EUR</SpecificationFreeText>
<SpecificationFreeText/>
```

```
</SpecificationDetails>
<EpiDetails>
<EpiIdentificationDetails>
<EpiDate Format="CCYYMMDD">20151222</EpiDate>
<EpiReference/>
</EpiIdentificationDetails>
<EpiPartyDetails>
<EpiBfiPartyDetails>
<EpiBfiIdentifier IdentificationSchemeName="BIC">BANKFIHH</EpiBfiIdentifier>
<EpiBfiName>NORDEA</EpiBfiName>
</EpiBfiPartyDetails>
<EpiBeneficiaryPartyDetails>
<EpiAccountID IdentificationSchemeName="IBAN">FI04904840131313</EpiAccountID>
</EpiBeneficiaryPartyDetails>
</EpiPartyDetails>
<EpiPaymentInstructionDetails>
<EpiInstructedAmount AmountCurrencyIdentifier="EUR">755,90</EpiInstructedAmount>
<EpiCharge ChargeOption="SLEV">SLEV</EpiCharge>
<EpiDateOptionDate Format="CCYYMMDD">20180522</EpiDateOptionDate>
<EpiPaymentMeansCode>54</EpiPaymentMeansCode></EpiPaymentInstructionDetails>
</EpiDetails>
</Finvoice>
```

B Liite 2: eKuitti layout

Sähköinen kuitti

Myyjä:
Y-tunnus: 0490484-0
Oy Yritys Ab
KESKUSKATU 12 C
13300 HÄMEENLINNA
Ostaja:
CARD

Laskun päivä:	22.5.2018
Laskunro:	45
Myyjän tilausnro:	232199645
Sopimus:	0118283371
Asiakasnro:	1012
Maksettava:	755,90 euroa
Eräpäivä:	22.5.2018
IBAN:	FI04 9048 4013 1313
BIC:	BANKFIHH
Maksukortti:	524342xxxxxx1401
Maksun tilanne:	Maksettu

Kuvaus HILAVITKUTIN Laskutettu 1.0000 M2	Tuotetunnus 123456		Yksikköhinta veroton ja määrä 7,98 => 7,98	Alv (S) Normaali veroprosentti Alv-määrä 1,92 (24,00 %)	Yhteensä verollinen 9,90
Kuvaus RUUVIMEISSELI Laskutettu 2.0000 KPL	Tuotetunnus 234567	EAN-koodi 0075678164125	Yksikköhinta veroton ja määrä 99,19 => 198,39	Alv (S) Normaali veroprosentti Alv-määrä 47,61 (24,00 %)	Yhteensä verollinen 246,00
Kuvaus PORAKONE Laskutettu 1.0000 KPL	Tuotetunnus 888888		Yksikköhinta veroton ja määrä 500,00 => 500,00	Alv (O) Veroton palvelu Alv-määrä 0,00 (0,00 %)	Yhteensä verollinen 500,00
Kuvaus MAKSUTAPA	Tuotetunnus Kortti CardMaskedNumber 524342xxxxxx1401 ReferenceNumber 151222010038 TimeStamp 151222100715 RequestedAmount 755,90 TenderAuthorizationMethodType ChipPin CreditCardCompanyCode L5				Yhteensä verollinen 755,90

ALV-erittely:

Yhteensä veroton:	706,37 euroa	
ALV yhteensä:	49,53 euroa	LASKU YHTEENSÄ: 755,90 euroa
Alv 24,00 % S:	49,53 euroa (206,37 euroa)	
Alv 0,00 % O:	0,00 euroa (500,00 euroa)	

KORTTITAPAHTUMA

Kortti: MC Debit
**** **** **** 1401 CP
Sovellus: A000000004101001
Tap.nro/Varmennus: 00038/179143
Yritys/Ala: 111111111111/5399

Autentisointi: 03072B69B36642F7
Viite: 151222010038

Debit/Veloitus 755,90 EUR

Ketju1 (Site) :

Nimi:	Myymälä 123
Postiosoite:	Kirkkokatu 23 15111 / HOLLOLA
Yhteystiedot:	03-5422 5422
Toimipiste:	Kassa 1

Oy Yritys Ab	ALV-numero: FI04904840 OVT-tunnus: 003704904840