hyväksymispäivä arvosana

arvostelija

Julkisen avaimen menetelmä ja ETSI:n MSS-määritys

Jyrki Saarinen

Helsinki 15.9.2006

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Jyrki Saarinen

Työn nimi — Arbetets titel — Title

Julkisen avaimen menetelmä ja ETSI:n MSS-määritys

Oppiaine — Läroämne — Subject

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	15.9.2006	76 sivua + 13 liitesivua

Tiivistelmä — Referat — Abstract

Julkisen avaimen menetelmä on ollut käytössä Suomessa jo jonkin aikaa esimerkiksi HST-korteilla, joissa on Väestörekisterikeskuksen myöntämä kansalaisvarmenne. Kansalaisvarmenteella voidaan tehdä mm. laillisesti sitovia digitaalisia allekirjoituksia ja salata sähköpostia. Viime aikoina myös eri matkapuhelinoperaattorit ovat alkaneet tarjota mobiilia kansalaisvarmennetta, jossa salainen avain on sijoitettu matkapuhelimen SIM-kortille.

Tässä pro gradu -työssä kuvataan ensin julkisen avaimen menetelmä ja sen käyttöalueet. Lisäksi mainitaan muutamia julkisen avaimen algoritmeja ja julkisen avaimen menetelmiin liittyviä ongelmia. Tämän jälkeen kuvataan ETSI:n MSS-määritys. ETSI:n MSS-määritys kuvaa palvelun rajapinnan, jossa loppukäyttäjältä pyydetään digitaalinen allekirjoitus. Lopuksi kuvataan Valimo Wireless Oy:n toteutus ETSI:n MSS-määrityksestä.

ACM Computing Classification System (CCS):
E.3 [Data Encryption],
D.2.[Software Engineering]

Avainsanat — Nyckelord — Keywords
julkisen avaimen menetelmä, varmenne, digitaalinen allekirjoitus, ETSI, MSS

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto **1**

2 Julkisen avaimen salaus **1**

2.1 Digitaalinen allekirjoitus 3

2.2 RSA 4

2.3 Diskreettiin logaritmiin perustuvat menetelmät 6

2.3.1 ElGamal 6

2.3.2 Diffie-Hellman 7

2.3.3 Digital Signature Algorithm 8

2.4 Elliptiset käyrät 9

2.5 Varmenteet 11

2.5.1 Luottamushierarkia 14

2.6 Eri algoritmien suorituskyvystä 15

3 ETSI:n MSS-määritys **17**

3.1 Mobiiliallekirjoitus 19

3.2 Viestien formaateista 19

3.3 Eri viestintämoodit 20

3.4 Muut MSS-palvelut 24

3.5 MSS-tietotyyppien määrittelyt 27

3.6 SOAP-alivirhekoodit 33

3.7 MSS-paluukoodit 34

4 ETSI:n MSS-standardin toteutus **34**

4.1 Eri viestintämoodien toteutus 40

4.1.1 Synkroninen asiakas-palvelin -moodi 40

4.1.2 Asynkroninen asiakas-palvelin -moodi 41

4.1.3 Asynkroninen palvelin-palvelin -moodi 42

4.2 Tarvittavat kolmansien osapuolten kirjastot 44
4.3 Modulijako 45
4.4 Module-luokka 45
4.5 Application Provider Data Manager -moduli 46
4.6 Billing Data Manager -moduli 47
4.7 Bridge-moduli 48
4.8 Confirmer-moduli 50
4.9 Hibernate-moduli 53
4.10 JMS-moduli 59
4.11 JMX-moduli 61
4.12 LDAP-moduli 61
4.13 SDK-moduli 63
4.14 Signing Gateway -moduli 66
4.15 SOAP-moduli 69
4.16 Transaction Data Manager -moduli 70

5 Suorituskyvystä **70**

6 Yhteenveto **71**

Lähteet **72**

Liitteet

1 XML-skeema -määritykset

A MSS-viestien XML-skeema -määrittelyt **1**

A.1 XML-skeema -nimiavaruuksista 1
A.2 MessageAbstractType 1
A.3 MSS_Signature-palvelu 2
A.4 MSS_StatusQuery-palvelu 4

A.5 MSS_ProfileQuery-palvelu 5
A.6 MSS_Registration-palvelu 6
A.7 MSS_Receipt-palvelu 8
A.8 MSS_Handshake-palvelu 9
A.9 MSS_Notification-palvelu 12

1 Johdanto

Julkisen avaimen salausalgoritmit ovat olleet Internetissä käytössä jo pitkään SSL- ja TLS -protokollissa [DA99] ja Secure Shell-palvelussa (SSH) [YL05], mutta muuten sen käyttö - esimerkiksi sähköpostin salaamisessa - on ollut vähäistä. Suomessa Väestörekisterikeskus (VRK) on jo jonkin aikaa myöntänyt HST-älykorteilla (Henkilön Sähköinen Tunnistaminen) ns. kansalaisvarmenteita, joita voidaan käyttää esimerkiksi laillisesti sitovan digitaalisen allekirjoituksen luontiin ja sähköpostin salaukseen. Korttien menekki ei kuitenkaan ole ollut toivottua, sillä HST-kortilla käytettäviä palveluita ei ole juurikaan ollut ja kortinlukijoiden saatavuus on ollut heikkoa. Kyseessä on siis jonkinasteinen muna-kana -ongelma.

Ratkaisuksi näihin ongelmiin on esitetty mobiilia kansalaisvarmennetta, jossa salainen avain on sijoitettu matkapuhelimen SIM-kortille. Tämän ratkaisun on toivottu tuovan lisäkiinnostusta kansalaisvarmenteeseen, sillä tässä tapauksessa matkapuhelin toimii kortinlukijana, matkapuhelin on henkilökohtainen ja melkein aina käyttäjän mukana. Suomessa tällaisia 'PKI-SIM'-kortteja myöntävät esimerkiksi TeliaSonera ja Elisa.

ETSI (European Telecommunications Standards Institute) on määritellyt SOAP:iin [GHM+03] perustuvan MSS-määrityksen (Mobile Signature Service), joka määrittelee rajapinnan palveluntarjoajan (Application Provider, AP) ja matkapuhelinoperaattorin tarjoaman MSS-palvelun (MSS Provider, MSSP) välille. MSS-rajapinnan toteutus siis mahdollistaa mm. digitaalisen allekirjoituksen pyytämisen loppukäyttäjältä palveluntarjoajalle. MSS-määritys kuvataan myöhemmin. Lisäksi kuvataan erään MSS-toteutuksen arkkitehtuuri ja suunnittelu kuvataan myöhemmin.

2 Julkisen avaimen salaus

Jotta tiedonvälitys avoimissa tietoverkoissa olisi luottamuksellista (confidentiality), tarvitaan tiedon salausta (encryption). Salaisen avaimen menetelmissä, joita kutsutaan myös symmetrisiksi salausmenetelmiksi, on kommunikoivilla osapuolilla hallussaan salainen avain K. Lähetettävä tieto eli selväteksti P salataan salausalgoritmilla f ja saadaan salateksti $C = f_K(P)$. Vastaanottaja saa salatekstin haltuunsa ja purkaa sen salaisella avaimella K saaden selvätekstin $P = f_K^{-1}(f_K(P)) = f_K^{-1}(C)$. Menetelmän turvallisuus perustuu siihen, että salatekstistä C on laskennallisesti vaikeaa

[1] saada selville selvätekstiä P.

Avain K on siis pidettävä salassa, sillä kuka tahansa kenellä on se hallussaan voi kuunnella osapuolten välistä kommunikointia. Ongelmaksi muodostuu suuri tarvittavien avainten määrä. Jos n osapuolta haluaa kommunikoida keskenään, tarvitaan $n^2 - n$ avainta.

Näiden avainten jakelun ja hallinnoinnin ongelmaan (key distribution problem) Diffie ja Hellman esittivät ratkaisuksi klassikkoartikkelissaan julkisen avaimen menetelmän [DH76] periaatteet.

Julkisen avaimen salauksessa ajatuksena on, että salaisen avaimen sijaan osapuolella on kaksi avainta, salainen avain K_{pr} ja julkinen avain K_{pu}. Nämä avaimet ovat keskenään matemaattisesti sellaisessa suhteessa, että toisen laskeminen toisen avulla on laskennallisesti vaikeaa. Diffien ja Hellmanin avaintenvaihdossa laskennallinen vaikeus on diskreettien logaritmien laskemisessa, RSA:ssa taas on kyse kokonaislukujen tekijöihinjaon vaikeudesta [2].

Julkisen avaimen menetelmässä luottamuksellisuus saavutetaan seuraavasti. Lähettäjä hankkii vastaanottajan B julkisen avaimen K_{pu_B} ja muodostaa salatekstin $C = f_{K_{pu_B}}(M)$. Vastaanottaja purkaa salatekstin C omalla salaisella avaimellaan K_{pr_B} saaden selvätekstin $P = f^{-1}_{K_{pr_B}}(f_{K_{pu_B}}(M)) = f^{-1}_{K_{pr_B}}(C)$.

Käytännössä päästä-päähän luottamuksellisuuden (point-to-point encryption) saavuttamiseksi käytetään istuntoavainta (session key) ja symmetrisiä salausmenetelmiä, sillä julkisen avaimen menetelmät ovat tyypillisesti monta kertaluokkaa hitaampia [Sta98]. Julkisen avaimen menetelmää kuten RSA tai Diffie-Hellman käytetään vain istuntoavaimen välittämiseen osapuolten välillä (key exchange).

Todentaminen (authentication) tapahtuu siten, että todennettava osapuoli A salaa todentamista haluavalta osapuolelta B saamansa haasteviestin (challenge text) M salaisella avaimellaan K_{pr_A} saaden salatekstin $C = f^{-1}_{K_{pr_A}}(M)$. Vastaanottaja B purkaa salatekstin A:n julkisella avaimella K_{pu_A} saaden selvätekstin
$M' = f_{K_{pu_A}}(f^{-1}_{K_{pr_A}}(M)) = f_{K_{pu_A}}(C)$. Jos $M' = M$, niin osapuoli A on todennettu ja voidaan sanoa A:n olevan se kuka hän väittääkin olevansa.

Julkisen avaimen menetelmässä ongelmana on osapuolen julkinen avain K_{pu}. Voidaanko olla varmoja siitä, kuuluuko K_{pu} todellakin sille entiteetille, joka väittää

[1] Murtamisen vaikeus raa'an voiman (brute force) menetelmällä on luokkaa $O(2^n)$ missä n on avaimen pituus.

[2] Tekijöihin jaon uskotaan olevan vaikea ongelma, mutta varmuutta ei asiasta vielä ole.

omistavansa K_{pu}:n? Tähän ongelmaan ratkaisuna on luotettu kolmas osapuoli, varmentaja (Certificate Authority, CA), josta myöhemmin lisää.

Järjestelmää, jossa on mukana myöhemmin esiteltävät varmenteet ja varmentajat, kutsutaan julkisen avaimen infrastruktuuriksi (Public Key Infrastructure, PKI).

2.1 Digitaalinen allekirjoitus

Dokumentti D voidaan digitaalisesti allekirjoittaa salaamalla se allekirjoittajan A salaisella avaimella: $DS = f_{K_{pr_A}}(D)$. Näin saadaan digitaalinen allekirjoitus DS. On vielä mainittava että tämä vaatii salausfunktiolta f sen että $D = f_{K_{pr_A}}(f_{K_{pu_A}}(D)) = f_{K_{pu_A}}(f_{K_{pr_A}}(D))$.

Digitaalinen allekirjoitus takaa että dokumentti D ei muuttunut matkalla pyytäjältä B allekirjoittajalle A (integrity) ja että sen on todellakin allekirjoittanut A, jos B luottaa A:n julkiseen avaimeen (varmenteeseen) (authentication). Kiistämättömyys (non-repudiation) saavutetaan siten, että luotettu osapuoli tallettaa DS:n. Tällöin allekirjoittaja A ei voi kiistää luoneensa DS:ää.

Koska dokumentti D:n voi olla mielivaltaisen suuri, salataan usein dokumentista D laskettu kryptografinen tiiviste (message digest, hash code) $MD = H(D)$, jottei DS:stakin tulisi mielivaltaisen suurta. MD on kooltaan vakio, ja on dokumentin D digitaalinen 'sormenjälki'. Tällaisia tiivistefunktioita ovat esimerkiksi MD5 [Riv92] ja SHA-1 [NIS95]. Näitä algoritmeja ei kuitenkaan suositella käytettäväksi uusissa digitaalisen allekirjoituksen sovelluksissa niistä löydettyjen heikkouksien vuoksi [Ran05], [Szy05].

Nyt $DS = f_{K_{pr_A}}(H(D))$ voidaan lähettää osapuolelle B, joka voi *tarkistaa* digitaalisen allekirjoituksen aitouden.

Digitaalinen allekirjoitus DS voidaan tarkistaa seuraavasti: käyttäjä B purkaa DS:n ja jos $H(D) = f_{K_{pu_A}}(DS)$, on DS tarkistettu oikein.

Kuten jo aikaisemmin mainittiin, ongelmana on, voidaanko luottaa siihen, että A:n julkinen avain K_{pu_A} todellakin on hänen? Tähän ongelmaan kuvataan ratkaisu myöhemmin.

2.2 RSA

RSA-algoritmissa avainpari tuotetaan seuraavien kappaleiden tavoin [Sch96]. Valitaan kaksi salaista alkulukua, p ja q, $p \neq q$. Tämän jälkeen lasketaan $n = p * q$. Lukua n kutsutaan (julkiseksi) modulukseksi, p ja q hävitetään.

Eulerin funktio $\phi(n)$ on niiden kokonaislukujen määrä jotka ovat pienempiä kuin n ja n:n kanssa suhteellisesti jaottomia (relatively prime). Jos p ja q ovat alkulukuja, niin $\phi(p * q) = (p-1)(q-1)$, joten $\phi(n)$ on helposti laskettavissa.

Julkinen eksponentti e valitaan siten, että $syt(\phi(n), e) = 1$ ja $1 < e < \phi(n)$. Julkinen avain on nyt $K_{pu} = (e, n)$. Salainen avain on $K_{pr} = (d, n)$, missä $d \equiv e^{-1} \pmod{\phi(n)}$.

Olkoon M salattava selväteksti. Tällöin salateksti C on $C = M^e \pmod{n}$. Salateksti C puretaan selvätekstiksi M laskemalla $M = C^d = (M^e)^d = M^{ed} = M^1 \pmod{n}$.

Salauksen ja sen purun suorituskykyä rajoittaa potenssiin korottaminen modulo n. Laskettavana on siis $a^b \pmod{n}$. Potenssin $a^b \pmod{n}$:n laskeminen $a * ... * a \pmod{n}$ vaatisi valtavia välituloksia, mutta onneksi modulo n voidaan suorittaa jokaisen iteraation jälkeen: $((a \pmod{n}) * (b \pmod{n})) \pmod{n} = (a * b) \pmod{n}$.

Seuraavassa on algoritmi joka suorittaa potenssiin korotuksen $a^b \pmod{n}$ [Sta98]. Eksponentti b on ilmaistu binäärilukuna $b_k, b_{k-1}, ..., b_0$.

```
d := 1
for i := k downto 0
  do
    d := (d * d) mod n
    if bit i set in b then
      d := (d * a) mod n
```

Kuten algoritmista voidaan nähdä, se toimii ajassa $O(k)$ (olettaen että modulo n:n laskeminen vie vakioajan $O(1)$) missä k on eksponentin ykkösbittien määrä. Näin ollen eksoponentiksi kannattaa valita luku jossa on mahdollisimman vähän ykkösbittejä, usein e:ksi valitaan $65537_{10} = 10000000000000001_2$. Tällöin julkisen avaimen operaatiot eli tiedon salaus tai digitaalisen allekirjoituksen tarkistus tapahtuvat nopeasti, mutta vastaavasti salaisen avaimen operaatiot kuten salauksen purku tai digitaalisen allekirjoituksen luonti vievät enemmän aikaa. Vastaavasti salaisen

avaimen eksponentiksi d voidaan valita luku joka sisältää vähän ykkösbittejä jolloin edellämainitut operaatiot nopeutuvat ja toiset vastaavasti hidastuvat [Wie98]. Riippuu siis käyttötarkoituksesta valitaanko e:ksi vai d:ksi vähän ykkösbittejä sisältävä eksponentti.

RSA:n turvallisuuden uskotaan perustuvan tekijöihin jaon ongelmaan [Sch96], johon ei tunneta polynomiaalisessa ajassa toimivia tehokkaita algoritmeja. Jos hyökkääjä saa jaettua n:n tekijöihinsä p ja q, voi hän helposti laskea arvon $\phi(n) = (p-1)(q-1)$ ja saada salaisen avaimen haltuunsa: $d \equiv e^{-1} \pmod{\phi(n)}$.

Hyökkääjä voi myös yrittää selvittää salaista eksponenttia d suoraan julkisesta avaimesta (e, n), mutta tämän uskotaan olevan yhtä vaikeaa kuin tekijöihin jako [KR95].

Toisaalta on olemassa jonkinlaista näyttöä siitä, että RSA:n murtaminen voisi olla helpompaa kuin tekijöihin jako silloin kun $d \equiv e^{-1} \pmod{\phi(n)}$ on pieni [BD98]. Itseasiassa näin on silloin kun $d < n^{0.292}$ [BD99].

Lyhyitä selvätekstejä käytettäessä RSA:ssa on ongelma. Olkoon salattava selväteksti m 56-bittinen DES-avain, jonka suuruus on luokkaa 10^{17} numeroa. Tämä salataan julkisella avaimella (e, n) ja saadaan salateksti $c \equiv m^e \pmod{n}$.

Nyt hyökkääjä tekee kaksi listaa:

1. $cx^{-e} \pmod{n}$ kaikille $0 \leq x \leq 10^9$
2. $y^e \pmod{n}$ kaikille $0 \leq y \leq 10^9$

Tämän jälkeen hyökkääjä etsii parin siten, että $cx^{-e} \equiv y^e \pmod{n}$ jollakin x, y. Tämä on sama kuin $c \equiv (xy)^e \pmod{n}$, ja $m \equiv xy \pmod{n}$, joten hyökkääjä on saanut selvätekstin m selville, olettaen että x ja y ovat kokonaislukuja ja pienempiä kuin 10^9. Moni selväteksti m on kahden tällaisen luvun tulo.

Tämä hyökkäys on paljon tehokkaampi kuin käydä läpi kaikki 10^{17} m:n arvoa. Tämän hyökkäyksen esto on helppoa. Lisätään satunnaisia bittejä m:n alkuun ja loppuun. Salatekstin c vastaanottaja tietää tämän ja osaa poistaa nämä ylimääräiset bitit.

Hieman avaimien pituuksista. Vuonna 2003 vähintään 73 bittiä pitkä symmetrisen salauksen avain tuo turvan raa'an voiman hyökkäystä vastaan [LV01]. Raa'an voiman tarvittavaksi laskentavoimaksi on arvioitu $3,51 * 10^{10}$ MIPS-vuotta.

RSA:n tapauksessa avaimen pituus tarkoittaa julkisen moduluksen n pituutta. Vuonna 2003 vähintään 1068 bittiä pitkän n uskotaan tuovan turvan raa'an voiman hyök-

käystä vastaan. Tämä osoittaa selvästi miten paljon suurempia avaimen pituuksia julkisen avaimen menetelmät (1068 vs. 73 bittiä) kuten RSA vaativat verrattuna symmetrisiin salausmenetelmiin.

2.3 Diskreettiin logaritmiin perustuvat menetelmät

Olkoon G äärellinen, multiplikatiivinen syklinen ryhmä jossa on n alkiota. Olkoon b ryhmän G virittäjä. Tällä tarkoitetaan sitä, että jokainen ryhmän G alkio $x \in G$ voidaan ilmaista virittäjän potenssina $x = b^k, k \in \mathbf{N}$.

Diskreetti logaritmi kannalle b on funktio: $log_b : G \rightarrow \mathbf{Z}_n$. Olkoon luku $y \in G$ muotoa $y = b^x$. Tällöin diskreetin logaritmin ongelma on laskea luku $x = log_b(y), x \in \mathbf{Z}_n$.

Diskreetin logaritmin laskentaongelman (DLP:n) uskotaan olevan vaikea joillakin ryhmillä G, mutta käänteinen ongelma eli diskreetti potenssiin korotus on helppo. Tätä asymmetriaa hyödynnetään esimerkiksi ElGamal-algoritmissa ja Diffien ja Hellmanin avaintenvaihtoalgoritmissa.

Diskreettiin logaritmiiin perustuvissa algoritmeissa avaimen pituus tarkoittaa p:n pituutta ryhmän G ollessa $\mathbf{Z}_p = \{1, 2, ..., p-1\}$.

Koska DLP:n ja tekijöihin jako -ongelman uskotaan olevan yhtä vaikeita, vähintään 1068 pitkän p:n uskotaan suojaavan raa'an voiman hyökkäyksiltä vuonna 2003 [LV01].

2.3.1 ElGamal

Avainparin generointi tapahtuu seuraavasti. Valitaan suuri alkuluku p siten, että diskreetti logaritmi -ongelma syklisessä multiplikatiivisessa ryhmässä $\mathbf{Z}_p^*$ on vaikea [Wie98].

Seuraavaksi valitaan ryhmän $\mathbf{Z}_p^*$ virittäjä g, joka korotetaan satunnaiseen potenssiin k; $h = g^k \pmod p, 1 < k < p-1$. Nyt julkinen avain K_{pu} on kolmikko (p, g, h) ja salainen avain K_{pr} on kolmikko (p, g, k).

Viestin M salaus tapahtuu seuraavasti. Viesti M koodataan lukuna $m, 1 \leq m \leq p-1$. Tämän jälkeen valitaan satunnaisluku $1 < s < p-1$. Salattu viesti C on nyt pari $C = (c_1, c_2)$, missä $c_1 = g^s \pmod p$ ja $c_2 = m * h^s \pmod p$.

Salatekstin $C = (c_1, c_2)$ purku tapahtuu salaisella avaimella $K_{pr} = (p, g, k)$ seuraavasti: $c_1^{-k} * c_2 = (g^s)^{-k} * m * h^s = (g^k)^{-s} * m * h^s = h^{-s} * m * h^s = m \pmod p$.

ElGamal-menetelmällä voidaan myös luoda digitaalinen allekirjoitus. Viestistä M, joka on koodattu numeroksi m, tuotetaan digitaalinen allekirjoitus salaisella avaimella $K_{pr} = (p, g, k)$ seuraavasti. Valitaan satunnainen $1 < s < p - 1$ siten, että s ja $p - 1$ ovat keskenään jaottomat. Tämän jälkeen lasketaan $s_1 = g^s \pmod p$ ja $s_2 = (m - k * s_1)s^{-1} \pmod{p-1}$. Digitaalinen allekirjoitus DS on nyt pari (s_1, s_2).

2.3.2 Diffie-Hellman

Diffien ja Hellmanin menetelmällä [DH76] voidaan kahden osapuolen välillä turvattomassa tietoverkossa vaihtaa jokin salaisuus, joka usein on symmetrisen salausmenetelmän istuntoavain.

Ensin valitaan alkuluku p ja kokonaisluku α siten, että α on p:n primitiivijuuri ja $\alpha < p$. Tällöin p:n primitiivijuuri generoi kaikki kokonaisluvut $1, ..., p-1$ potenssien kautta.

Primitiivijuuren α potenssit $\alpha \pmod p, \alpha^2 \pmod p, \alpha^3 \pmod p, ..., \alpha^{p-1} \pmod p$ tuottavat kokonaisluvut $1, ..., p - 1$ jossakin permutaatiossa, joten α on $\mathbf{Z_p}^*$:n viritäjä.

Olkoot p ja α julkisia. Jos A ja B haluavat vaihtaa salaisuuden K, A valitsee salaisen satunnaisluvun $X_A < p$ ja laskee $Y_A = \alpha^{X_A} \pmod p$.

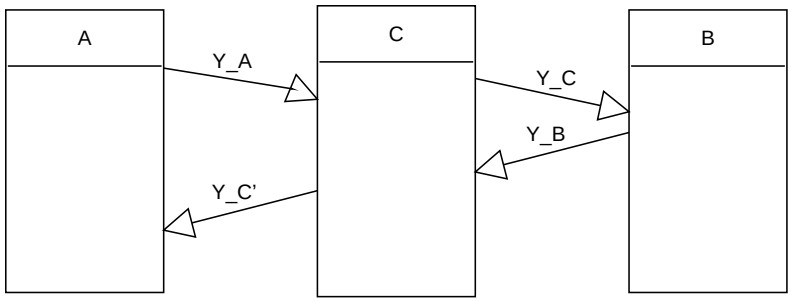
Samaan tapaan B valitsee salaisen satunnaisluvun $X_B < p$ ja laskee $Y_B = \alpha^{X_B} \pmod p$.

Seuraavaksi A ja B vaihtavat julkiset luvut Y_A ja Y_B keskenään. Sitten molemmat osapuolet korottavat toisen osapuolen julkisen Y:n salaiseen potenssiin X saaden salaisuuden K:

$K = Y_B^{X_A} \pmod p$
$= (\alpha^{X_B} \pmod p)^{X_A} \pmod p$
$= (\alpha^{X_B})^{X_A} \pmod p$
$= \alpha^{X_B * X_A} \pmod p$
$= (\alpha^{X_A})^{X_B} \pmod p$
$= (\alpha^{X_A} \pmod p)^{X_B} \pmod p$
$= Y_A^{X_B} \pmod p$.

Hyökkääjällä joka haluaa selvittää K:n ei ole muuta julkista tietoa saatavilla kuin (q, α, Y_A, Y_B). Hänen täytyy siis laskea diskreetti logaritmi $X_A = log_\alpha(Y_A)$ joka on tunnetusti vaikeaa tietyille ryhmille [MW96] sillä tehokasta algoritmia ei tunneta.

Diffie-Hellman on myös altis välimieshyökkäykselle (man in the middle attack). Hyökkäävä osapuoli voi kaapata sekä A:n B:lle lähettämällä julkisen luvun Y_A ja lähettämällä B:lle oman julkisen luvun Y_C että B:n A:lle lähettämän julkisen luvun Y_B ja lähettämällä A:lle oman julkisen luvun Y'_C. Näin hyökkääjä voi salakuunnella A:n ja B:n välistä keskustelua kaappaamalla A:n B:lle lähettämän viestin, purkamalla sen A:n ja hyökkääjän kanssa luodun salaisen avaimen avulla. Tämän jälkeen hyökkääjä uudelleensalaa viestin B:n ja hyökkääjän kanssa luodulla salaisella avaimella. A ja B eivät huomaa tätä toimintaa mitenkään. Tämä välimieshyökkäys on estettävissä todentamalla osapuolet A ja B ennen istuntoavaimen vaihtoa esimerkiksi digitaalisella allekirjoituksella. Tämä välimieshyökkäys on esitetty kuvassa 1.



Kuva 1: Diffie-Hellman välimieshyökkäys

2.3.3 Digital Signature Algorithm

Digital Signature Algorithm:n turvallisuus perustuu diskreetin logaritmin laskemisen vaikeuteen [Sta98]. Se suunniteltiin pelkästään digitaalisen allekirjoituksen laskemiseen, ei salaukseen tai avainten vaihtoon.

Ensin generoidaan avainpari. Valitaan L-bittinen alkuluku p, $512 \leq L \leq 1024$. Sitten valitaan 160-bittinen alkuluku q siten, että $p-1 = qz, z \in \mathbf{Z}$. Tämän jälkeen valitaan sekä h, $1 < h < p-1$ siten, että $g = h^z \pmod p$ että satunnainen $0 < x < q$. Julkinen avain on nyt $K_{pu} = (p, q, g, y)$ missä $y = g^x \pmod p$. Salainen avain on $K_{pr} = x$.

Viestistä M lasketaan digitaalinen allekirjoitus seuraavasti. Valitaan satunnaisluku s, $1 < s < q$. Sitten lasketaan $s_1 = (g^s \pmod p) \pmod q$ ja $s_2 = (H(M) - s_1 * x)^{s^{-1}} \pmod q$, missä H on SHA-1 -tiivistefunktio. Digitaalinen allekirjoitus DS on nyt pari (s_1, s_2).

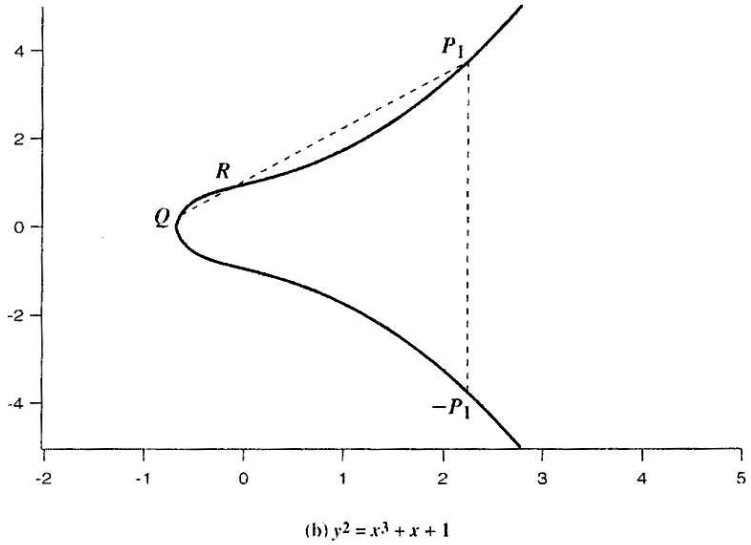
DS tarkistetaan seuraavasti:

1. $w = s_2^{-1} \pmod{q}$
2. $u_1 = H(M) * w \pmod{q}$
3. $u_2 = s_1 * w \pmod{q}$
4. $s_1' = ((g^{u_1} * y^{u_2}) \pmod{p}) \pmod{q}$

Tarkistus on onnistunut jos $s_1 = s_1'$.

2.4 Elliptiset käyrät

Reaaliset elliptiset käyrät määritellään muotoa $y^2+axy+by = x^3+cx^2+dx+e$ olevien yhtälöiden avulla missä $a, b, c, d, e \in \mathbf{R}$. Olkoon elliptisen käyrän määrittelyssä myös piste O, eli piste äärettömyydessä (point at infinity) tai nollapiste (zero point). Tällöin voidaan määritellä yhteenlasku pisteiden välille [Sta98]:



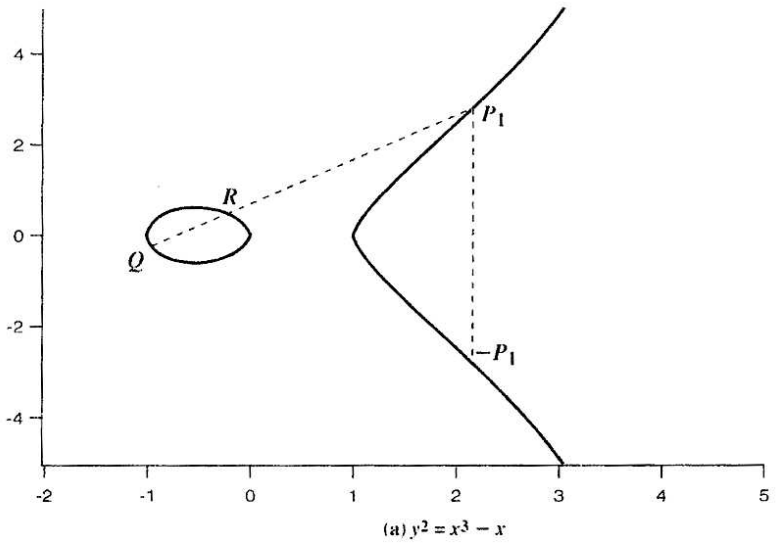
Kuva 2: Esimerkki elliptisestä käyrästä

1. Piste O on additiivinen identiteetti, eli $O = -O$. Tällöin jokaiselle pisteelle $P \in E$ pätee $P + 0 = 0 + P = P, P \in E$.
2. Pystysuora suora leikkaa elliptisen käyrän kahdessa pisteessä joilla on sama x-koordinaatti. Olkoon pisteet $P_1 = (x, y)$ ja $P_2 = (x, -y)$. Suora leikkaa myös nollapisteen. Täten $P_1 + P_2 + O = O$ ja $P_1 = -P_2$. Tällöin pisteen $P = (x, y)$ 'negaatio' $-P$ on piste $(x, -y)$; tämä on esitetty kuvissa 2 ja 3.

3. Kahden pisteen Q ja R; joilla on eri x-koordinaatit; yhteenlasku suoritetaan seuraavasti. Piirretään niiden välille suora ja haetaan kolmas (leikkaus)piste P_1. Täten $Q+R+P_1 = 0$ joten $Q+R = -P_1$; tämä on esitetty kuvissa 2 ja 3.

4. Pisteen Q tuplapiste saadaan piirtämällä pisteeseen Q tangentti ja hakemalla leikkauspiste S elliptisellä käyrällä. Tällöin $Q + Q = 2Q = -S$.

Yllä esitetyt yhteenlaskun säännöt noudattavat yhteenlaskun normaaleja ominaisuuksia kuten kommutatiivisuutta ja assosiatiivisuutta. Pisteen $P \in E$ kertominen kokonaisluvulla $k \in \mathbf{N}$ on määritelty seuraavasti: $2P = P + P$, $3P = P + P + P, ...$



Kuva 3: Esimerkki elliptisestä käyrästä

Elliptisten käyrien kryptografiassa (ECC) käytetään elliptisiä käyriä, jotka ovat määritelty äärellisen kunnan suhteen. Riittää myös rajoittua vain tietyn muotoisiin käyriin. Useimmiten kuntana käytetään kuntaa Z_p, missä p on alkuluku. Tällöin elliptinen käytä voidaan määritellä yhtälön $y^2 \equiv x^3 + ax + b \pmod{p}$ avulla, missä $0 < a < p$ ja $0 < b < p$ ja $4a^3 + 27b^2 \pmod{p} \neq 0$. Merkitään tällaista käyrää symbolilla $E_p(a, b)$.

ECC:n yhteenlasku vastaa RSA:n kertolaskua modulo n. Kokonaisluvulla kertomista vastaa potenssiinkorotus modulo n. Olkoon $Q = kP, P, Q \in E, 0 < k < p$. Q:n laskeminen on helppoa kun k ja p on annettu, mutta k:n laskeminen on vaikeaa kun P ja Q on annettu. Tätä ongelmaa kutsutaan elliptisen käyrän diskreetin logaritmin ongelmaksi (ECDLP, Elliptic Curve Discrete Logarithm Problem).

Diffien ja Hellmanin avaintenvaihtoalgoritmi voidaan muuntaa käyttämään elliptisiä käyriä (ECDH, Elliptic Curve Diffie-Hellman) seuraavasti [3].

Ensin valitaan alkuluku p siten, että p:n pituus on toivotun avaimen pituus. Sitten valitaan ryhmälle $E_p(a,b)$ parametrit a ja b. Tämän jälkeen valitaan virittäjäpiste $G(x_1, y_1) \in E_p(a,b)$, tämä vastaa virittäjää α normaalissa Diffie-Hellmanissa. $G(x_1, y_1)$ ja $E_p(a,b)$ ovat julkisia. Avaintenvaihto käyttäjien A ja B välillä tapahtuu seuraavasti:

1. Käyttäjä A valitsee kokonaisluvun n_A, tämä on A:n salainen avain. Seuraavaksi A laskee julkisen avaimen $P_A = n_A * G \in E_p(a,b)$.
2. Käyttäjä B valitsee kokonaisluvun n_B, tämä on B:n salainen avain. Seuraavaksi B laskee julkisen avaimen $P_B = n_B * G \in E_p(a,b)$.
3. A ja B vaihtavat julkiset avaimet P_A ja P_B keskenään.
4. A laskee salaisuuden $K = n_A * P_B$ ja B laskee salaisuuden $K = n_B * P_A$.

Hyökkääjän täytyisi laskea k kun tiedossa on G ja kG, eli kyse on ECDLP:stä. Uskotaan, etteivät diskreetin logaritmin ongelma multiplikatiivisessa ryhmässä äärellisen kunnan yli ja ECDLP ole samoja ongelmia. Lisäksi uskotaan että ECDLP on huomattavasti vaikeampi ongelma kuin DLP [Sta98].

Elliptisten käyrien kryptografiassa avaimen pituus on sen äärellisen kunnan koko, jonka yli elliptinen käyrä on määritelty. Vuonna 2003 on arvioitu, että alkuluku p, jonka pituus on vähintään 136-140 bittiä, tuo vastaavan turvan kun 83 bittiä symmetrisessä järjestelmässä [LV01]. Täten ECC tarjoaa saman tietoturvan tason paljon pienemmällä avaimen koolla kuin RSA ja diskreettiin logaritmiin perustuvat menetelmät. Tämä tekee ECC:stä sopivan sovelluksiin, joissa prosessoriteho ja/tai muisti ovat rajoitettuja resursseja [Wie98]. Käyttöä kuitenkin rajoittaa Certicom-yhtiön lisenssipolitiikka. Yhtiöllä on yli 130 patenttia liittyen elliptisiin käyriin.

2.5 Varmenteet

Varmenne (Digital Certificate) on haltijansa digitaalinen 'henkilökortti'. Varmentajan myöntämässä varmenteessa on haltijansa julkisen avaimen lisäksi muun muas-

[3] Diskreettiin logaritmiin perustuvat menetelmät kuten DSA voidaan samaan tapaan muuntaa käyttämään elliptisiä käyriä, jolloin kyse on ECDSA:sta (Elliptic Curve Digital Signature Algorithm)

sa tietoa varmenteen haltijasta sekä varmenteen voimassaolosta. Varmenne sisältää myös myöntäjän eli varmentajan tunnisteen. Tärkeimpänä varmenne sisältää vielä varmentajan digitaalisen allekirjoituksen, jolla varmentaja takaa käyttäjän varmenteen oikeellisuuden.

Yleisesti käytössä oleva varmenteen formaatti on X.509-varmenne.

Sen ASN.1-määritelty rakenne on kuvattu seuraavassa. *Certificate*-rakenteessa on itse varmenteen sisältö *tbsCertificate* sekä varmentajan digitaalinen allekirjoitus *signature* ja algoritmi, *signatureAlgoritm*, jolla allekirjoitus on muodostettu [Sta98].

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING  }
```

Varmenne koostuu oleellisilta osiltaan seuraavasti: *serialNumber* on yksikäsitteinen sarjanumero varmentajan sisällä. Varmentajan ja varmenteen haltijan 'nimet' ilmoitetaan *issuer*- ja *subject*-kentissä. Nämä nimet ovat DN-nimiä (Distinguished Name) [M. 97]. Esimerkkinä DN-nimestä voisi olla 'CN=Jyrki Saarinen, OU=R&D, O=Valimo Wireless Oy, C=FI'. Lisäksi varmenteesta löytyy voimassaoloaika *validity* sekä varmenteen haltijan julkinen avain *subjectPublicKeyInfo*.

Seuraavassa *tbsCertificate*:n ASN.1-määrittely:

```
TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version must be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version must be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version must be v3
     }
```

Varmenteet myöntää siis *varmentaja* (CA, Certificate Authority), joka omalla salaisella avaimellaan digitaalisesti allekirjoittaa myöntämänsä varmenteet. Eli jos luottaa tiettyyn varmentajaan, voi myöskin luottaa tämän varmentajan myöntämiin varmenteisiin ja tätä kautta varmenteiden julkisiin avaimiin.

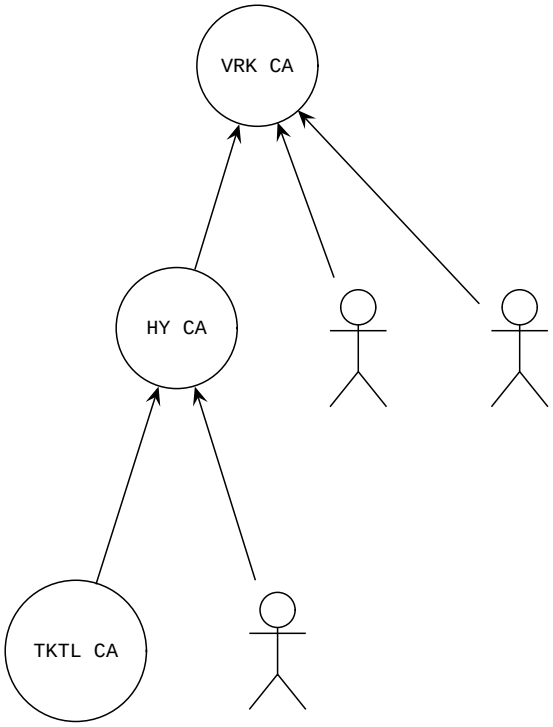
Seuraavassa vielä esimerkkinä kirjoittajan mobiilivarmenteen sisältö, joka on Elisan (entinen Radiolinja) varmentajan myöntämä:

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1033372654 (0x3d9803ee)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=FI, O=Radiolinja Certification Authorities, CN=Radiolinja CA 1
    Validity
        Not Before: Sep 30 07:57:34 2002 GMT
        Not After : Sep 30 06:57:34 2006 GMT
    Subject: C=FI, O=Radiolinja customer certificates, SN=1304236,
             CN=89358026020212001188-A0, S=SAARINEN, G=JYRKI
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:b7:39:f3:c2:4e:6e:93:31:6e:0f:e5:bd:1f:3f:
                f9:86:5f:20:18:40:a4:7e:f6:35:9e:30:4b:5b:1d:
                fd:02:6d:f2:23:6c:6c:94:0f:4b:a7:c2:f6:41:ca:
                e1:0f:10:15:15:ff:12:da:a2:60:bc:de:bb:41:72:
                83:e5:42:40:bf:d3:c5:d2:de:ea:39:13:b0:b9:92:
                39:20:2a:09:56:8b:33:0c:69:1d:34:29:64:b4:c2:
                2d:55:3b:fd:47:87:84:6a:54:4a:1c:b8:1f:aa:04:
                a6:21:42:f2:bc:00:34:bd:89:9c:a9:15:9c:13:07:
                d4:46:64:64:84:84:f4:d8:7b
            Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 CRL Distribution Points:
        URI:ldap://ldap.radiolinja.fi:389/cn=radiolinja%20ca%201,
                                         o=radiolinja%20certification%20authorities,
                                         c=fi?certificaterevocationlist;binary
     X509v3 Certificate Policies:
        Policy: 1.2.246.277.2.11.1.2
          User Notice:
            Explicit Text: Radiolinjan mobiilivarmenne. Varmenteeseen luottava taho on
                           velvollinen tarkistamaan Varmennepolitiikan (CP) ja CA:n
                           Toimintamallin (CPS) https://wpki.radiolinja.fi/ ennen
                           varmenteeseen luottamista.
          CPS: https://wpki.radiolinja.fi
 Signature Algorithm: sha1WithRSAEncryption
    aa:9e:d3:08:59:8a:41:d7:a7:eb:3a:9a:3d:d9:7f:61:bd:99:
    3f:b3:ca:fb:28:60:df:73:8c:8f:b9:95:57:3c:13:25:f9:02:
    fe:a9:ce:62:c0:17:aa:46:32:ff:62:1d:78:f9:5a:da:e5:40:
    55:0a:9e:f5:72:6e:87:c4:06:fb:05:8d:56:da:df:24:4f:50:
    5a:a7:f9:7d:94:5e:03:a6:66:06:c0:53:5d:7b:8a:51:cb:22:
    d3:31:db:a8:c7:96:be:4a:1d:ee:a3:13:8c:db:8f:ec:b2:e6:
    11:47:15:ef:82:6b:bc:68:7c:61:f0:5a:f9:ce:ce:ab:59:fe:
    3c:cb:be:6a:44:85:09:95:5f:ca:e0:75:20:51:b7:0c:5c:c8:
    fa:29:18:29:b2:e9:62:fa:18:a3:2b:be:7f:13:33:69:94:0c:
    c6:37:3b:a8:a5:a8:11:59:b8:9e:ad:a6:e7:26:e2:db:8d:d3:
```

```
5f:97:57:92:c6:f3:e9:a1:75:73:4c:c7:0c:4a:57:26:dc:fb:
0f:27:6d:d2:44:6d:8f:62:56:e7:99:fd:a3:9a:e1:1e:f1:8a:
76:2a:cd:b3:23:f7:7b:88:02:e9:8b:b0:3e:23:5a:fb:77:a8:
24:32:9a:38:54:26:41:92:ef:32:75:4b:8e:a8:be:9f:7c:b6:
7d:38:86:fc
```

2.5.1 Luottamushierarkia

Edellä olevan mukaan olisi siis yksi (juuri)varmentaja, joka myöntää käyttäjävarmenteita. Varmenteiden luottamushierarkia ei kuitenkaan ole välttämättä kaksitasoinen. Voi olla, että *juurivarmentajalla* on alivarmentajia. Esimerkiksi Suomessa Väestörekisterikeskus (VRK) voisi toimia juurivarmentajana, ja kaupalliset yritykset tai yhteisöt, kuten matkapuhelinoperaattorit tai Helsingin yliopiston ATK-keskus voisivat toimia VRK:n varmentamina (ali)varmentajina. Tämä kuvitteellinen hierarkia on kuvattu kuvassa 4.



Kuva 4: Luottamushierarkian esimerkki

Kuvassa 4 VRK:n varmentaja on myöntänyt kahdelle henkilölle käyttäjävarmenteen (käytännössä juurivarmentajat eivät yleensä myönnä käyttäjävarmenteita). VRK:n varmentaja on myös myöntänyt varmentajan varmenteen yliopiston ATK-keskukselle, joka taas on myöntänyt varmentajan varmenteen Tietojenkäsittelytieteen laitoksen

varmentajalle sekä yhden käyttäjävarmenteen.

2.6 Eri algoritmien suorituskyvystä

Kuvissa 5, 6 ja 7 vertaillaan RSA:n ja ElGamal -algoritmien suorituskykyä avainten generoinnissa, salauksessa (tai digitaalisen allekirjoituksen tarkistamisessa) ja salauksen purussa (tai digitaalisen allekirjoituksen muodostamisessa).

Algoritmit ovat kirjoittajan toteuttamia Java-ohjelmointikielellä. Mielivaltaisten suurien kokonaislukujen käsittelyyn käytettiin Javan standardikirjastosta löytyvää **java.math.BigInteger**-luokkaa.

Testit ajettiin AMD XP 1900+ PC:llä jossa oli Linux-käyttöjärjestelmä ja Sunin Java-virtuaalikone. Virtuaalikoneen versio oli 1.4.2_01 ja HotSpot server JIT -kääntäjä oli käytössä (JVM:n -server -optio) [Sun02].

Algoritmit toteutettiin kuten tässä pro gradu -työssä on kuvattu.

Testeissä vertaillaan kolmea algoritmia avaimen pituuden ollessa 512 bitistä 2048 bittiin, ja salattavan datan määrä oli 64 tavua.

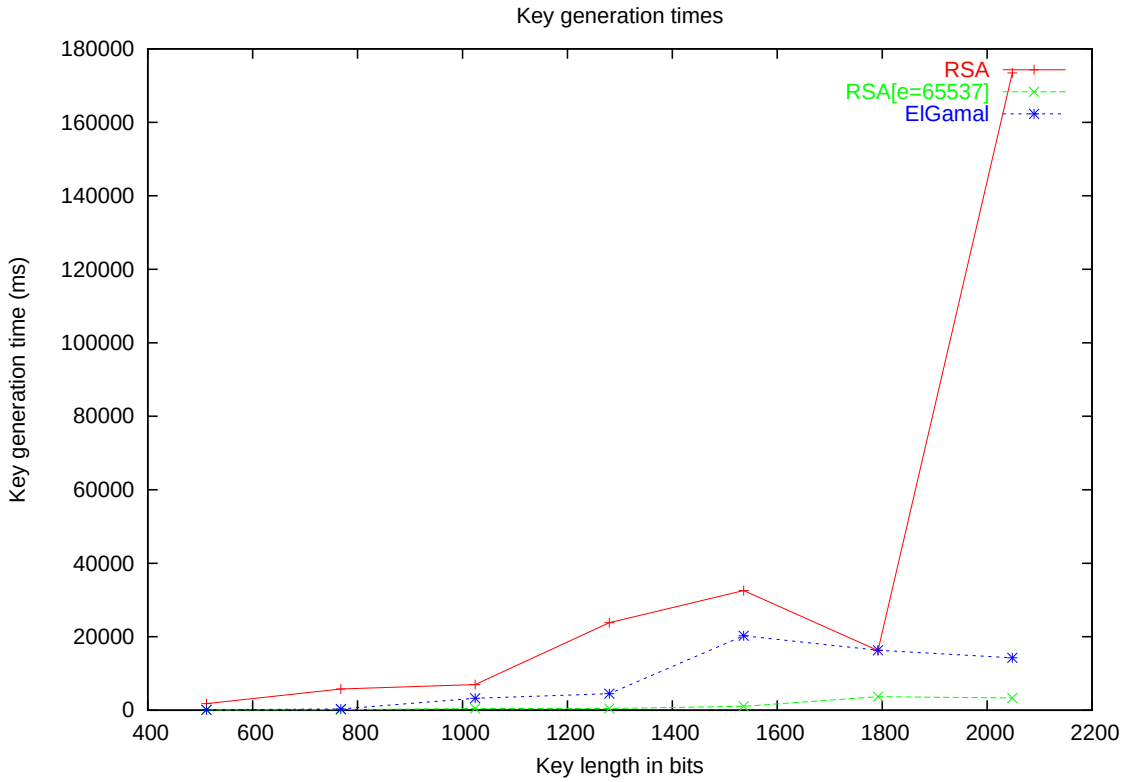
Ensimmäinen algoritmi on RSA, jossa e valittiin satunnaisesti. Toinen algoritmi on myös RSA, mutta jossa e on asetettu $65537_{10} = 1000000000000001_2$:ksi. Kolmas algoritmi on ElGamal, jossa eksponentti k on satunnainen siten, että $1 < k < p-1$.

Kuvassa 5 on avainparin luonnin vertailua. RSA kiinteällä eksponentilla e on selvästi nopein.

Kuvassa 6 vertaillaan salauksen tai digitaalisen allekirjoituksen tarkistuksen nopeutta. Tässä kuvassa näkyy selvästi RSA:n etu; vähän 1-bittejä sisältävällä eksponentilla e suorituskyky on huomattavasti parempi kuin muilla algoritmeilla. Täten RSA vähän 1-bittejä sisältävällä eksponentilla e on hyvä valinta sovellukseen, jossa tehdään paljon salausta tai digitaalisen allekirjoituksen tarkistusta (molemmat ovat siis julkisen avaimen operaatioita), esimerkiksi varmennepohjaisissa järjestelmissä [Wie98].

Salauksen purku tai digitaalisen allekirjoituksen generointi (molemmat ovat salaisen avaimen operaatioita) suorituskyky näkyy kuvassa 6. ElGamal-algoritmi on hieman RSA:ta suorituskykyisempi.

RSA on hitaampi salaisella avaimella operoidessa (allekirjoittaessa tai salauksen purussa) kuin diskreettiin logaritmiin perustuvat järjestelmät. Toisaalta RSA vähän

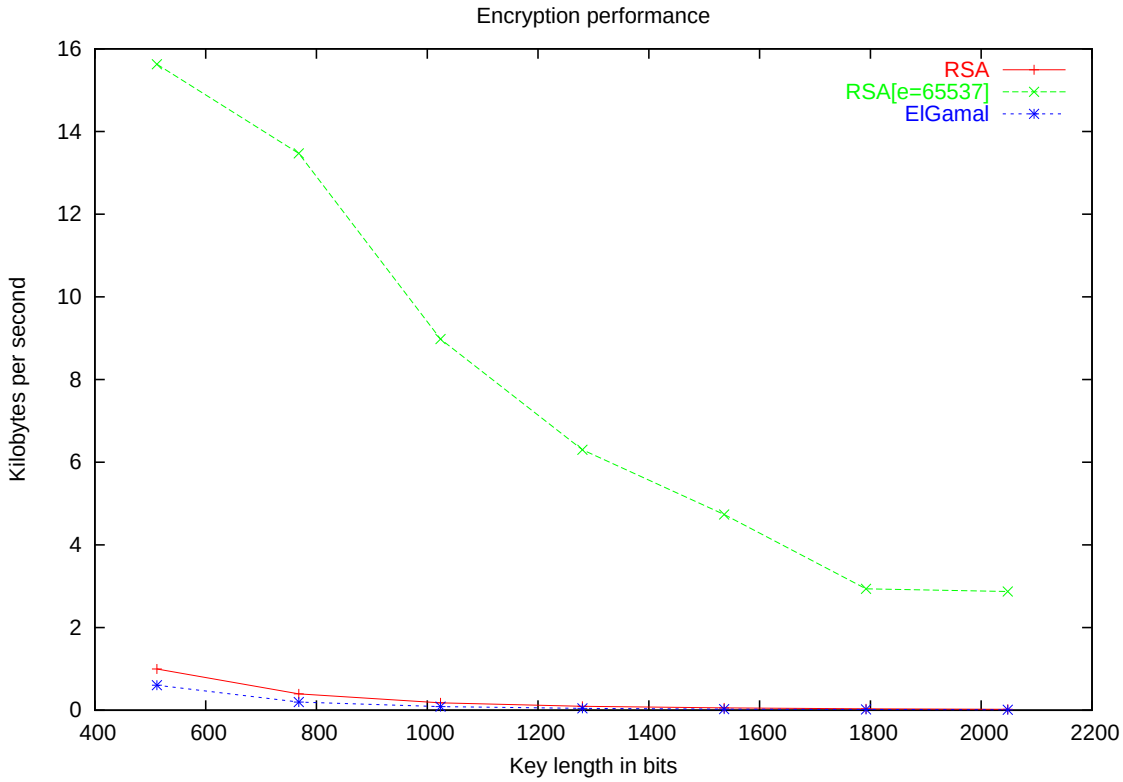


Kuva 5: Key generation speed

1-bittejä sisältävällä eksponentilla e on huomattavasti nopeampi julkisen avaimen operaatioissa (allekirjoituksen tarkistus tai salaus) verrattuna diskreettiin logaritmiin perustuviin menetelmiin [Wie98].

Seuraavan taulukon [Wie98] tulokset ovat samansuuntaisia kirjoittajan tulosten kanssa; RSA on erittäin nopea kun $e = 3$, ja tarkistetaan digitaalista allekirjoitusta tai salataan. Toisaalta diskreettiin logaritmiin perustuvat menetelmät ovat nopeampia allekirjoituksen muodostamisessa.

	RSA-1024 (e=3)	DSA-1024	ECDSA-168 (over GF(p))
sign	43	7	8
verify	0.6	27	19
key gen.	1100	7	7
param. gen.	none	6500	large (research area)



Kuva 6: Public key operations performance

3 ETSI:n MSS-määritys

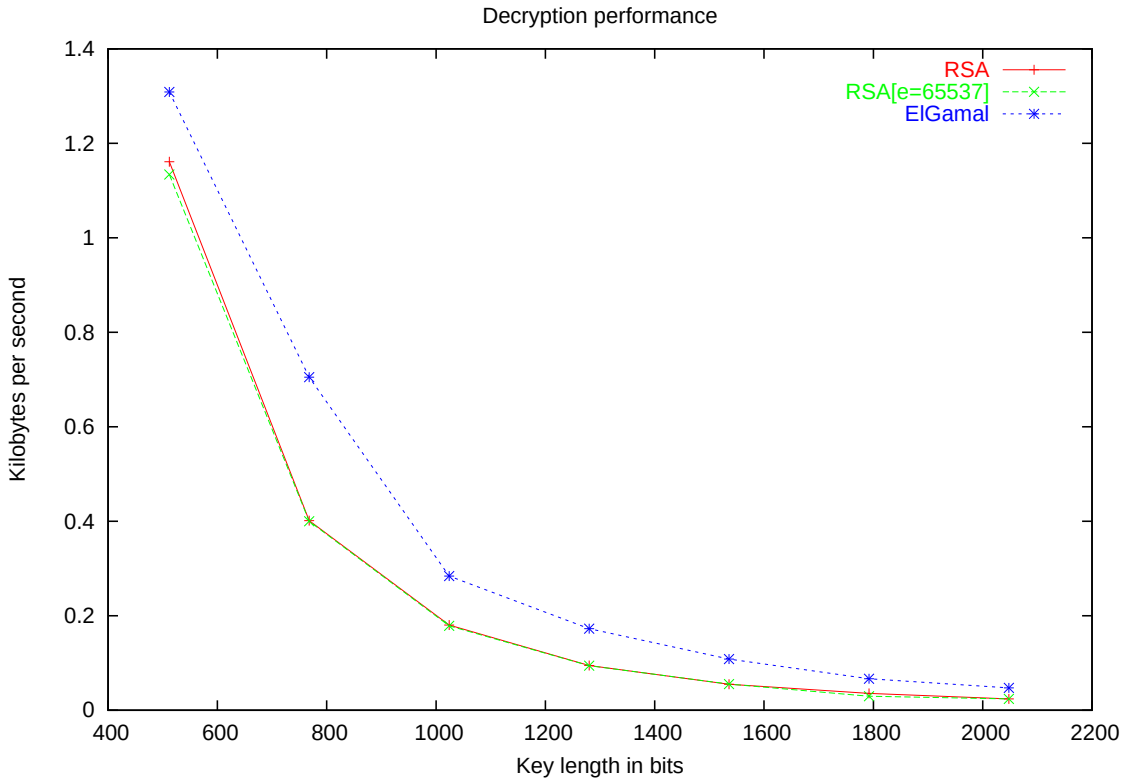
ETSI:n MSS-määritys [ETS03] määrittelee SOAP:iin [GHM+03] perustuvan rajapinnan, jolla voidaan mm. pyytää loppukäyttäjältä digitaalinen allekirjoitus. Rajapinta tarjoaa myös muitakin palveluita, kuten kuittauksen lähettämisen loppukäyttäjälle, mutta oleellisin palvelu on digitaalisen allekirjoituksen pyytäminen. Kaikki palvelut kuvataan tarkemmin myöhemmin.

Hieman SOAP:sta:

"SOAP on XML:ään perustuva, alkujaan kevyt protokolla, joka on suunniteltu viestien vaihtoon hajautetussa ympäristössä [Eng02]. SOAP on suunniteltu olemaan riippumaton ohjelmointikielistä ja muista toteutukseen liittyvistä yksityiskohdista, jotta se soveltuisi mahdollisimman moniin ympäristöihin."

Lisäksi lainaus www.w3.org:sta:

"SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML techno-



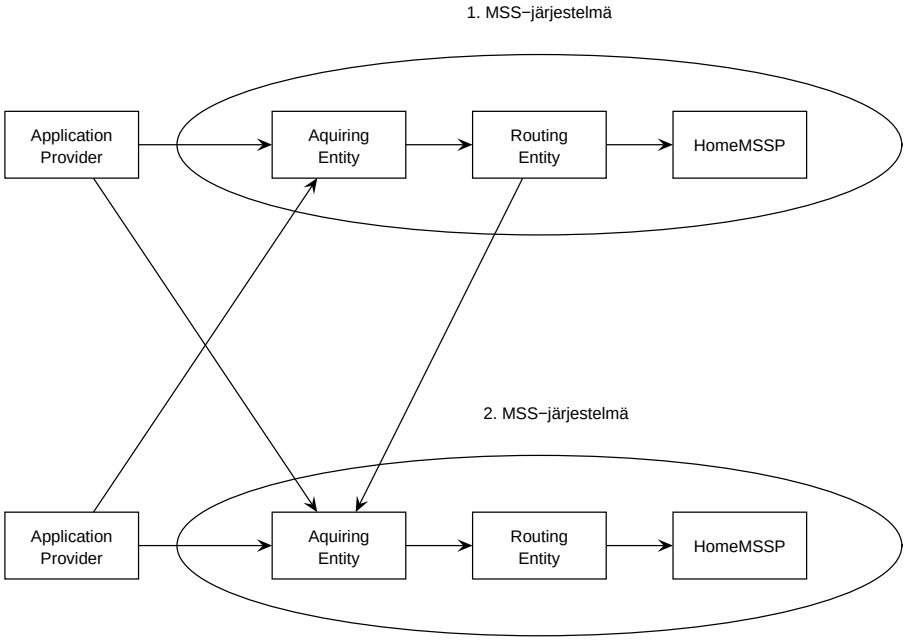
Kuva 7: Private key operations performance

logies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics."

MSS-järjestelmä koostuu seuraavista osakomponenteista, jotka voivat olla hajatettuna tai sitten toteutettuna yhtenä monoliittisena palvelimena.

- Aquiring Entity - vastaanottaa MSS-pyynnön.
- Routing Entity - mahdollisesti reitittää MSS-pyynnön toiseen MSS-järjestelmään.
- HomeMSSP - käsittelee MSS-pyynnön.

Lisäksi on vielä olemassa Application Provider (AP), joka edustaa MSS-palvelun käyttäjää eli asiakasta (client). Yllämainittu roolijako on esitetty kuvassa 8.



Kuva 8: MSS-järjestelmän roolit

3.1 Mobiiliallekirjoitus

MSS-palvelun täytyy tukea seuraavia toiminnallisuuksia, jotka ovat tarkoitettu kahteen eri tarkoitukseen:

1. Mobiiliallekirjoituksen pyytäminen loppukäyttäjältä siten, että loppukäyttäjä näkee allekirjoittamansa tekstin allekirjoituslaitteen näytöllä.

2. Mobiiliallekirjoituksen pyytäminen loppukäyttäjältä siten, että loppukäyttäjä ei näe allekirjoittamaansa tekstiä. Tämä käyttötapaus tulee ajankohtaiseksi esimerkiksi silloin, kun rekisteröintiauktoriteetti (RA, Registration Authority) haluaa loppukäyttäjän allekirjoittavan PKCS#10-muotoisen [RSA00] varmennepyynnön.

3.2 Viestien formaateista

Jokainen seuraavissa kappaleissa kuvattu palveluviesti periytyy seuraavanlaisesta MessageAbstractType-nimisestä rakenteesta, jossa on neljä osaa:

1. MajorVersion: on kokonaisluku, tällä hetkellä aina 1, ja se on pakollinen.

2. MinorVersion: on kokonaisluku, tällä hetkellä aina 1, ja se on pakollinen.

3. AP_Info: sisältää tietoa AP:sta

4. MSSP_Info: identifioi MSS-palvelun

Tarkat viestien XML-skeemat on kuvattu liitteessä 1.

AP_Info on kuvattu seuraavassa:

Nimi	Tyyppi	Kuvaus	Pakollinen
AP_ID	anyURI	AP:n tunniste	K
AP_TransID	NCName	AP:n luoma tunniste tapahtumalle	K
AP_PWD	String	AP:n salasana	K
Instant	DateTime	Ajanhetki jolloin AP loi viestin	K
AP_URL	anyURI	URI osoitteeseen johon takaisinkutsu tehdään, jos kyse palvelin-palvelin -viestintämoodista	E

MSSP_Info on kuvattu seuraavassa:

Nimi	Tyyppi	Kuvaus	Pakollinen
MSSP_ID	MeshMemberType	MSS-palvelun identifioiva tieto	K
Instant	DateTime	Ajanhetki jolloin MSS-palvelu loi viestin	E

3.3 Eri viestintämoodit

MSS-määritys määrittelee MSS_Signature-palvelulle kolme eri viestintämoodia. Ensimmäinen on synkroninen asiakas-palvelin -moodi, jossa MSS_Signature-palvelu odottaa ("blokkaa"), kunnes allekirjoitus on saapunut puhelimelta tai aikakatkaisu on tapahtunut:

1. AP lähettää MSS_SignatureRequest-viestin.

2. Loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö push-viestinä.

3. Loppukäyttäjä näppäilee PIN-koodinsa ja puhelin luo digitaalisen allekirjoituksen, joka palaa MSS-palvelimelle.

4. Digitaalinen allekirjoitus käsitellään MSS-palvelimella ja AP:lle vastataan MSS_SignatureRespo viestillä.

Tämä prosessi on esitetty kuvassa 9.

MSS_Signature-palvelulla pyydetään loppukäyttäjältä digitaalinen allekirjoitus. Palvelun vastauksena on digitaalinen allekirjoitus sisältäen loppukäyttäjän varmenteen, jos tapahtuma meni odotetusti. Vastauksena voi tietysti olla myös jokin virhekoodi, esimerkiksi aikakatkaisu (timeout) jos loppukäyttäjä ei reagoinut, eli syöttänyt PIN-koodiaan ajoissa.

MSS_SignatureRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
DataToBeSigned	DataType	Allekirjoitettava teksti	K
DataToBeDisplayed	DataType	Näytettävä teksti	E
SignatureProfile	mssURIType	Toivottu allekirjoitusprofiili	E
AdditionalServices	AdditionalServiceType[]	Lista lisäarvopalveluista	E
MSS_Format	mssURIType	Mitä allekirj. form. toivotaan	E
KeyReference	KeyReferenceType	Mitä avainta tulisi käyttää	E
SignatureProfileComparison	SignatureProfileComparisonType		E
ValidityDate	DateTime	Kauanko tapahtuma on voimassa	E
TimeOut	PositiveInteger		E
MessagingMode	MessagingModeType	Haluttu viestintämoodi	K

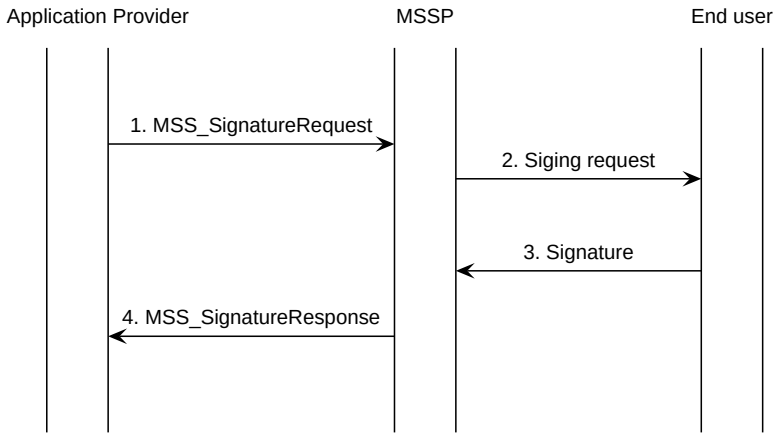
Seuraavassa on kuvattu MSS_SignatureResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
Status	StatusType	Tapahtuman tila	K
SignatureProfile	mssURIType	Mitä allekirjoitusprofiilia käytettiin	E
MSS_Signature	SignatureType	Jos tapahtuma on valmis, allekirjoitus	E
MSSP_TransID	NCName	MSS-palvelun luoma tapahtumatunnus	K

MSS_Signature-palvelua voidaan käyttää kolmella erilaisella viestintämoodilla, jotka on kuvattu seuraavassa. MSS-palvelun täytyy tukea kaikkia kolmea eri viestintämoodia.

Toinen moodi on asynkroninen asiakas-palvelin -moodi, jossa MSS-asiakas kysyy ("pollaa") tapahtuman tilaa, kunnes se on valmis:

1. AP lähettää MSS_SignatureRequest-viestin.
2. Loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö push-viestinä.
3. AP kysyy tapahtuman tilaa lähettämällä MSS_StatusQueryRequest-viestin, mutta tapahtuma ei ole vielä valmis. Vastaus tapahtuu MSS_StatusQueryResponse-viestillä.

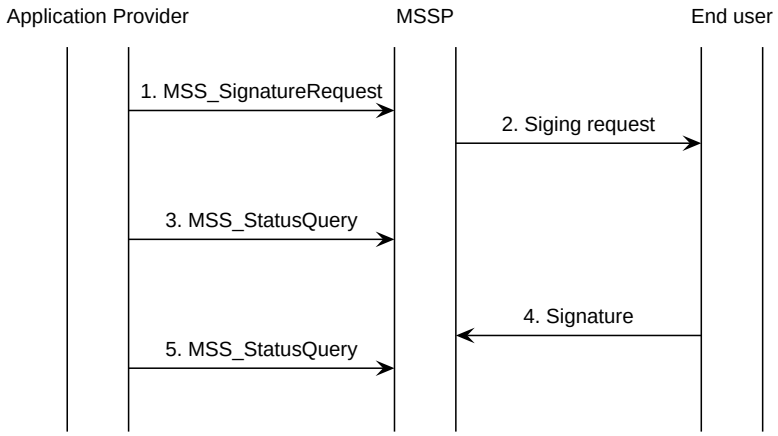


Kuva 9: Synkroninen asiakas-palvelin -moodi

4. Loppukäyttäjä näppäilee PIN-koodinsa ja puhelin luo digitaalisen allekirjoituksen, joka palaa MSS-palvelimelle. Digitaalinen allekirjoitus käsitellään MSS-palvelimella.

5. AP kysyy tapahtuman tilaa lähettämällä MSS_StatusQueryRequest-viestin, johon vastatataan MSS_StatusQueryResponse-viestillä, ja tapahtuman tila on valmis.

Tämä prosessi on esitetty kuvassa 10.



Kuva 10: Asynkroninen asiakas-palvelin -moodi

MSS_StatusQuery-palvelu on kuvattu seuraavassa.

MSS_StatusQuery-palvelulla kysytään MSS_Signature-tapahtuman tilaa. Vastauksena on joko 1) tapahtuma on valmis ja allekirjoitus on paluuviestissä mukana, 2) tapahtuma on kesken tai 3) on tapahtunut jokin virhe (esimerkiksi aikakatkaisu, EXPIRED_TRANSACTION).

MSS_StatusQueryRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MSSP_Trans_ID	NCName	Minkä tapahtuman tilaa kysytään	K

Tyyppi NCName on määritelty seuraavasti: [BHL99]

```
NCName ::= (Letter | "_") (NCNameChar)*
NCNameChar ::= Letter | Digit | "." | "-" | "_" | CombiningChar | Extender
```

Seuraavassa on kuvattu MSS_StatusQueryResponse-viestin kentät:

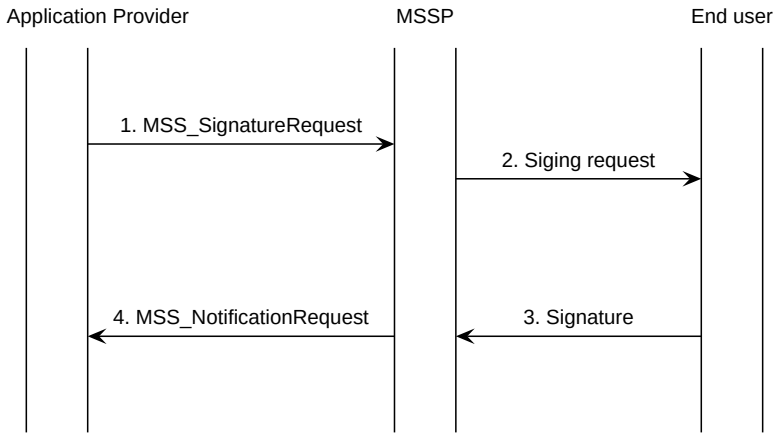
Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
MSS_Signature	SignatureType	Jos tapahtuma on valmis, allekirjoitus	E
Status	StatusType	Tapahtuman tila	K

Kolmas moodi on asynkroninen palvelin-palvelin -moodi, jossa MSS-asiakkaalle tehdään takaisinkutsu (callback) kun tapahtuman tila on valmis.

1. AP lähettää MSS_SignatureRequest-viestin.
2. Loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö push-viestinä.
3. Loppukäyttäjä näppäilee PIN-koodinsa ja puhelin luo digitaalisen allekirjoituksen, joka palaa MSS-palvelimelle.
4. Digitaalinen allekirjoitus käsitellään MSS-palvelimella, ja AP:lle vastataan tekemällä MSS_NotificationRequest-takaisinkutsu MSS_SignatureResponse-viestillä.

Tämä prosessi on esitetty kuvassa 11.

Eri viestintämoodeilla on omat hyvät puolensa. Synkronisessa asiakas-palvelin -moodissa AP kuluttaa palvelimen resursseja (tämänhetkisellä Java Servlet API:lla kuluu yksi säie per MSS-kutsu) kunnes tapahtuma on valmis, kun taas asynkronisessa asiakas-palvelin -moodissa AP kysyy tapahtuman tilaa kuormittaen palvelinta. Täten ei voida sanoa kumpi viestintämoodi on parempi, se riippuu täysin käyttötapauksesta ja tilanteesta. Asynkroninen palvelin-palvelin -moodi on tehokkain; se ei varaa säiettä pitkäaikaisesti palvelimella kuten synkroninen asiakas-palvelin -moodi, eikö myöskään tehoa huku AP:n kysyessä tapahtuman tilaa, kuten asynkronisessa asiakas-palvelin -moodissa tapahtuu.



Kuva 11: Asynkroninen palvelin-palvelin -moodi

3.4 Muut MSS-palvelut

MSS_ProfileQuery-palvelulla AP kysyy MSS-palvelulta allekirjoitusprofiilia, joka voi olla erilainen riippuen loppukäyttäjästä. Erot voivat olla esimerkiksi erilaisia kryptografisia tekniikoita [ETS03]. Tyypillisesti allekirjoitusprofiilia kysytään ennen MSS_Signature-palvelun käyttöä. Vastauksena on URI, joka on viittaus tiettyyn allekirjoitusprofiiliin.

Seuraavassa on kuvattu MSS_ProfileQueryRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	E

Seuraavassa on kuvattu MSS_ProfileQueryResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileSignatureProfile	mssURIType	Yksi tai useampia profiileja joita MSS-palvelu tukee	E
Status	StatusType	Tapahtuman tila	K

MSS_Registration-palvelulla voidaan rekisteröidä loppukäyttäjä MSS-palveluun. Esimerkiksi rekisteröintiauktoriteetti (RA, Registration Authority) voi pyytää MSS-palvelua lataamaan loppukäyttäjän varmenteen ja aktivoimaan allekirjoitussovelluksen SIM-kortilla.

Seuraavassa on kuvattu MSS_RegistrationRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
EncryptedData	EncryptedType		E
EncryptResponseBy	anyURI		E
CertificateURI	anyURI	Loppukäyttäjän varmenteen URI	E
X509Certificate	base64Binary	Loppukäyttäjän varmenne	E
Any	Any		E

Seuraavassa on kuvattu MSS_RegistrationResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
Status	StatusType	Tapahtuman tila. Täytyy olla koodi 408	K
EncryptResponseBy	anyURI		E
CertificateURI	anyURI	Loppukäyttäjän varmenteen URI	E
X509Certificate	base64Binary	Loppukäyttäjän varmenne	E
PublicKey	base64Binary	Loppukäyttäjän julkinen avain	E

MSS_Receipt-palvelulla voidaan lähettää kuittaus loppukäyttäjän matkapuhelimeen. Tyypillisesti tätä palvelua käytetään MSS_Signature-palvelun lopuksi informoimaan loppukäyttäjää siitä, että onnistuiko allekirjoitustapahtuma vai ei.

Seuraavassa on kuvattu MSS_ReceiptRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
Status	StatusType	Tapahtuman tila	E
Message	DataType	Näytettävä teksti	E
SignedReceipt	XMLSignature	AP:n antama allekirjoitettu kuittaus	E
MSSP_TransID	NCName	Kuittauksen täytyy viittata edell. tapahtumaan	K

Seuraavassa on kuvattu MSS_ReceiptResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
Status	StatusType	Tapahtuman tämänhetkinen tila	K

MSS_Handshake-palvelua käytetään kun AP ja MSS-palvelu haluavat tulla yhteisymmärrykseen käytettyjen XML-allekirjoitusten muodosta. On olemassa kaksi käyttötapausta:

1. AP ja MSS-palvelu eivät tunne toisiaan, ja haluavat selvittää toistensa ominaisuudet.

2. AP:lla ja MSS-palvelulla on sopimussuhde, ja ne tuntevat toistensa ominaisuudet, mutta AP tai MSS-palvelu haluaa saada tapahtumankäsittelyyn lisäturvaa esimerkiksi käyttämällä XML-allekirjoituksia viestinnässä.

Seuraavassa on kuvattu MSS_HandshakeRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
SecureMethods	ComplexType	Mitkä MSS-palvelut AP haluaa turvattuna XML-allekirjoituksella	K
Certificates	ComplexType	AP antaa listan varmenteita joita se voi käyttää XML-allekirjoituksen tekoon	K
RootCAs	ComplexType	AP antaa listan CA-varmenteita joita se voi käyttää XML-allekirjoituksen tarkistukseen	K
SignatureAlgList	ComplexType	Lista allekirjoitusformaateista joita AP tukee	K

Seuraavassa on kuvattu MSS_HandshakeResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MSSP_TransID	NCName	MSS-palvelun luoma tapahtuman tunniste	K
SecureMethods	ComplexType	Mitkä MSS-palvelut MSS-palvelu haluaa turvattuna XML-allekirjoituksella	K
MatchingMSSPCertificates	ComplexType	MSS-palvelun juurivarmenteet jotka sopivat AP:n pyynnössä antamaan listaan	K
MatchingAPCertificates	ComplexType	AP:n pyynnössä antamat varmenteet joihin MSS-palvelu voi luottaa tarkistukseen	K
MatchingSigAlgList	ComplexType	Lista allekirjoitusformaateista joita AP ja MSS-palvelu tukevat	K

MSS_Notification-palvelulla ilmoitetaan AP:lle, että tapahtuma on valmis. Tämä tapahtuu takaisinkutsuna (callback), joten AP toimii tässä tilanteessa palvelimena.

Seuraavassa on kuvattu MSS_NotificationRequest-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
Status	StatusType	Tapahtuman tila	K
SignatureProfile	mssURIType	Mitä allekirjoitusprofiilia käytettiin	E
MSS_Signature	SignatureType	Jos tapahtuma on valmis, allekirjoitus	E
MSSP_TransID	NCName	MSS-palvelun luoma tapahtumatunnus	K

Seuraavassa on kuvattu MSS_NotificationResponse-viestin kentät:

Nimi	Tyyppi	Kuvaus	Pakollinen
MobileUser	MobileUserType	Loppukäyttäjän MSISDN	K
MSS_Signature	SignatureType	Jos tapahtuma on valmis, allekirjoitus	E
Status	StatusType	Tapahtuman tila	K

3.5 MSS-tietotyyppien määrittelyt

Seuraavassa on esitetty MSS-nimiavaruuden tietotyyppien määrittelyt XML-skeeman muodossa.

URI-tietotyypillä [BLMF+98] ilmaistaan MSS-palvelussa seuraavia asioita:

- Application Providereita
- Itse MSS-palvelua
- Allekirjoitusformaatteja
- Lisäpalveluita, kuten aikaleimausta, allekirjoituksen tarkistusta tai allekirjoituksen arkistointia
- Allekirjoitusprofiileja
- MSS-palvelun XML-skeeman nimiavaruutta

MeshMemberType-tietotyypillä ilmaistaan jonkin entiteetin osoite, esimerkiksi MSS-palvelun osoite. Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="DNSName"
                type="xs:string"
                minOccurs="0"/>
    <xs:element name="IPAddress"
                type="xs:string"
                minOccurs="0"/>
    <xs:element name="URI"
                type="xs:anyURI"
                minOccurs="0"/>
    <xs:element name="IdentifierString"
                type="xs:string"
                minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

DigestAlgAndValueType-tietotyyppi on säiliö kryptografiselle tiivisteelle ja algoritmille millä se on luotu. Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="DigestAlgAndValueType>
  <xs:sequence>
    <xs:element name="DigestMethod"
                type="ds:DigestMethodType"
```

```
                minOccurs="0"/>
    <xs:element name="DigestValue"
                type="ds:DigestValueType"/>
  </xs:sequence>
</xs:complexType>
```

mssURIType-tietotyypillä ilmaistaan esimerkiksi allekirjoitusprofiilin URI, ja valinnaisesti mukana on myös URI:n kryptografinen tiiviste. Tietotyyppi 'any' mahdollistaa mielivaltaisen tiedon lisäämisen tähän tietotyyppiin. Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="mssURIType>
  <xs:sequence>
    <xs:element name="mssURI"
                type="xs:anyURI"/>
    <xs:element name="DigestAlgAndValue"
                type="mss:DigestAlgAndValueType"
                minOccurs="0"/>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Seuraavilla URI:lla määritellään eri allekirjoitusformaatteja:

- **"http://uri.etsi.org/TS102204v1.1.2/#XMLSignature"** määrittelee XML-allekirjoituksen [BBF+02a].
- **"http://uri.etsi.org/TS102204v1.1.2/#PKCS7"** määrittelee PKCS#7-allekirjoituksen [RSA93].
- **"http://uri.etsi.org/TS102204v1.1.2/#CMS-Signature"** määrittelee CMS-allekirjoituksen [HS99].
- **"http://uri.etsi.org/TS102204v1.1.2/#PKCS10"** määrittelee PKCS#10-allekirjoituksen [RSA00].

MobileUserType-tietotyyppi määrittelee loppukäyttäjän yksikäsitteisesti. Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="MobileUserType>
  <xs:sequence>
    <xs:element name="IdentityIssuer"
                type="mss:MeshMemberType"
```

```
                minOccurs="0"/>
    <xs:element name="UserIdentifier"
                type="xs:string"
                minOccurs="0"/>
    <xs:element name="HomeMSSP"
                type="mss:MeshMemberType"
                minOccurs="0"/>
    <xs:element name="MSISDN"
                type="xs:string"
                minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

MessagingModeType-tietotyypillä määritellään miten AP ja MSS-palvelu kommunikoivat keskenään. Kolme mahdollista moodia ovat:

1. Synkroninen asiakas-palvelin -moodi, perinteinen pyyntö-vaste -viestintä.
2. Asynkroninen asiakas-palvelin -moodi, useita pyyntö-vaste -viestejä.
3. Asynkroninen palvelin-palvelin -moodi, kaksisuuntainen viestintä. Asiakkaalta vaaditaan palvelinominaisuuksia sillä MSS-palvelu tekee takaisinkutsun asiakkaalle.

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:simpleType name="MessagingModeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="synch"/>
    <xs:enumeration value="asynchClientServer"/>
    <xs:enumeration value="asynchServerServer"/>
  </xs:restriction>
</xs:simpleType>
```

DataType-tietotyyppiä käytetään säiliönä kun loppukäyttäjälle lähetetään viestejä allekirjoitettavaksi tai näytettäväksi. MIME-tyyppi on määritelty seuraavassa dokumentissa [FIBV96a] ja koodaus seuraavassa [FIBV96b].

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="DataType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="MimeType"
                    type="xs:string"
                    use="optional"/>
      <xs:attribute name="Encoding"
                    type="xs:string"
```

```
                use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

KeyReferenceType-tietotyypillä ilmaistaan se julkinen avain jota halutaan käyttää, kun loppukäyttäjä tekee digitaalisen allekirjoituksen. Avain voidaan määritellä seuraavasti:

- URI joka osoittaa loppukäyttäjän varmenteeseen
- Loppukäyttäjän varmenteen myöntäjän DN (Distinguished Name) [M. 97]
- Avaimen kryptografinen tiiviste ja tiivistealgoritmi
- Loppukäyttäjän varmenteen myöntäjän varmenteen kryptografinen tiiviste ja tiivistealgoritmi
- Vapaavalintainen menetelmä (xs:any-elementti)

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="KeyReferenceType">
  <xs:sequence>
    <xs:element name="CertificateURL"
                type="xs:anyURI"
                minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="CertificateIssuerDN"
                type="xs:string"
                minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="HashOfUsersPublicKey"
                type="mss:DigestAlgAndValueType"
                minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="HashOfCAPublicKey"
                type="mss:DigestAlgAndValueType"
                minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

AdditionalServiceType-tietotyypillä AP ilmaisee MSS-palvelulle mitä lisäarvopalveluita se haluaisi, kuten allekirjoituksen tarkastusta tai allekirjoituksen arkistointia.

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="AdditionalServiceType">
  <xs:sequence>
    <xs:element name="Description"
                type="mss:mssURIType"/>
    <xs:element name="Entity"
                type="mss:MeshMemberType"
                minOccurs="0"/>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Seuraavilla URI:lla määritellään eri lisäarvopalveluita:

- **"http://uri.etsi.org/TS102204v1.1.2/#validate"**
  määrittelee allekirjoituksen tarkistuksen
- **"http://uri.etsi.org/TS102204v1.1.2/#timestamp"**
  määrittelee allekirjoituksen aikaleimauksen
- **"http://uri.etsi.org/TS102204v1.1.2/#archive"**
  määrittelee allekirjoituksen arkistoinnin

SignatureProfileComparisonType-tietotyypillä määritellään miten MSS-palvelu käyttäytyy kun se saa tälläisen tietoalkion MSS_SignatureRequest-pyynnössä:

- Jos 'exact', niin MSS-palvelun täytyy löytää ainakin yksi sopiva allekirjoitusprofiili
- Jos 'minimum', niin MSS-palvelua pyydetään käyttämään allekirjoitusprofiilia joka on vähintään yhtä 'hyvä' kuin MSS_SignatureRequest-pyynnössä pyydetään
- Jos 'better', niin MSS-palvelua pyydetään käyttämään allekirjoitusprofiilia joka on 'paras' mitä on tarjoilla.

Se mitä 'hyvä' ja 'paras' tarkoittavat, ei MSS-määritys sen kummemmin määrittele.

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:simpleType name="AdditionalServiceType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="exact"/>
    <xs:enumeration value="minimum"/>
    <xs:enumeration value="better"/>
  </xs:restriction>
</xs:simpleType>
```

SignatureType-tietotyyppi on säiliö loppukäyttäjän luomalle ja AP:lle palautettavalle digitaaliselle allekirjoitukselle. Sisältönä voi olla joko XML-allekirjoitus (ds:SignatureType), PKCS#7-allekirjoitus (xs:base64Binary) tai CMS-allekirjoitus (xs:base64Binary).

Tyyppi toimii myös säiliönä muulle datalle (xs:any-tyyppi) jota MSS-palvelu mahdollisesti tuottaa allekirjoitustapahtuman yhteydessä, kuten validointidataa, tarkistusdataa, aikaleimoja jne.

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="SignatureType>
  <xs:choice>
    <xs:element name="XMLSignature"
                type="ds:SignatureType"/>
    <xs:element name="Base64Signature"
                type="xs:base64Binary"/>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:choice>
</xs:complexType>
```

StatusType-tietotyyppi ilmoittaa MSS-tapahtuman tilan, joten siinä on pakollinen statuskoodikenttä. Lisäksi mukana on valinnainen tilaan liittyvä viesti.

Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="StatusType>
  <xs:sequence>
    <xs:element name="StatusCode"
                type="mss:StatusCodeType"/>
    <xs:element name="StatusMessage"
                type="xs:string"
                minOccurs="0"/>
    <xs:element name="StatusDetail"
```

```
              type="mss:StatusDetailType"
              minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

StatusCodeType-tietotyyppi sisältää MSS-palvelulta palautuvan statuskoodin. Määritys mahdollistaa sisäkkäiset statuskoodit. Seuraavassa tietotyypin määrittelevä XML-skeema:

```
<xs:complexType name="StatusCodeType>
  <xs:sequence>
    <xs:element name="StatusCode"
                type="mss:StatusCodeType"/>
  </xs:sequence>
  <xs:attribute name="Value"
                type="xs:integer"
                use="required"/>
</xs:complexType>
```

StatusDetailType-tietotyyppi tarjoaa lisätietoa tapahtuman tilasta ja on varsin vapaamuotoinen kuten seuraava XML-skeema -määritys osoittaa:

```
<xs:complexType name="StatusDetailType>
  <xs:sequence>
    <xs:any namespace="##other"
            processContents="lax"
            minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3.6 SOAP-alivirhekoodit

Seuraavassa on lueteltu SOAP-alivirhekoodit (SOAP fault subcodes) [GHM+03], joita MSS-palvelu voi palauttaa vasteena MSS-pyyntöön.

Koodi	Syy	Selitys
101	WRONG_PARAM	Virheellinen parametri MSS-pyynnössä
102	MISSING_PARAM	Puuttuva parametri MSS-pyynnössä
103	WRONG_DATA_LENGTH	DataToBeSigned liian suuri
104	UNAUTHORIZED_ACCESS	AP:ta ei tunneta tai salasana on väärä
105	UNKNOWN_CLIENT	AP:n haluamaa loppukäyttäjää ei tunneta
106	HANDSHAKE_REQUIRED	MSS-palvelu haluaa tehdä ensin kättelyn
107	INAPPRORIATE_DATA	MSS-palvelu ei tunne MIME-tyyppiä
108	INCOMPATIBLE_INTERFACE	MinorVersion ja/tai MajorVersion vääriä
109	UNSUPPORTED_PROFILE	AP yrittää käyttää väärää allekirjoitusprofiilia
208	EXPIRED_TRANSACTION	Tapahtumalle on tapahtunut aikakatkaisu
209	OTA_ERROR	MSS-palvelu ei pysty keskustelemaan loppukäyttäjän kanssa
401	USER_CANCEL	Loppukäyttäjä hylkäsi tapahtuman
402	PIN_NR_BLOCKED	Tapahtuman aikana tapahtui virhe loppukäyttäjän
403	CARD_BLOCKED	allekirjoituslaitteessa
404	NO_KEY_FOUND	
405	NO_URL_FOUND	
406	PB_SIGNATURE_PROCESS	
407	REGISTRATION_NOK	
422	NO_CERT_FOUND	Loppukäyttäjän varmennetta ei löytynyt
423	CRL_PB	Tapahtui virhe loppukäyttäjän varmenteen validoinnin
424	CRL_EXPIRED	aikana
425	ERROR_CERTIFICATE	
900	INTERNAL_ERROR	MSS-palvelun sisäinen virhe

3.7 MSS-paluukoodit

Seuraavassa on lueteltu MSS-statuskoodit, joita MSS-palvelu voi palauttaa vasteena (StatusCodeType:n attribuuttina) MSS-pyyntöön.

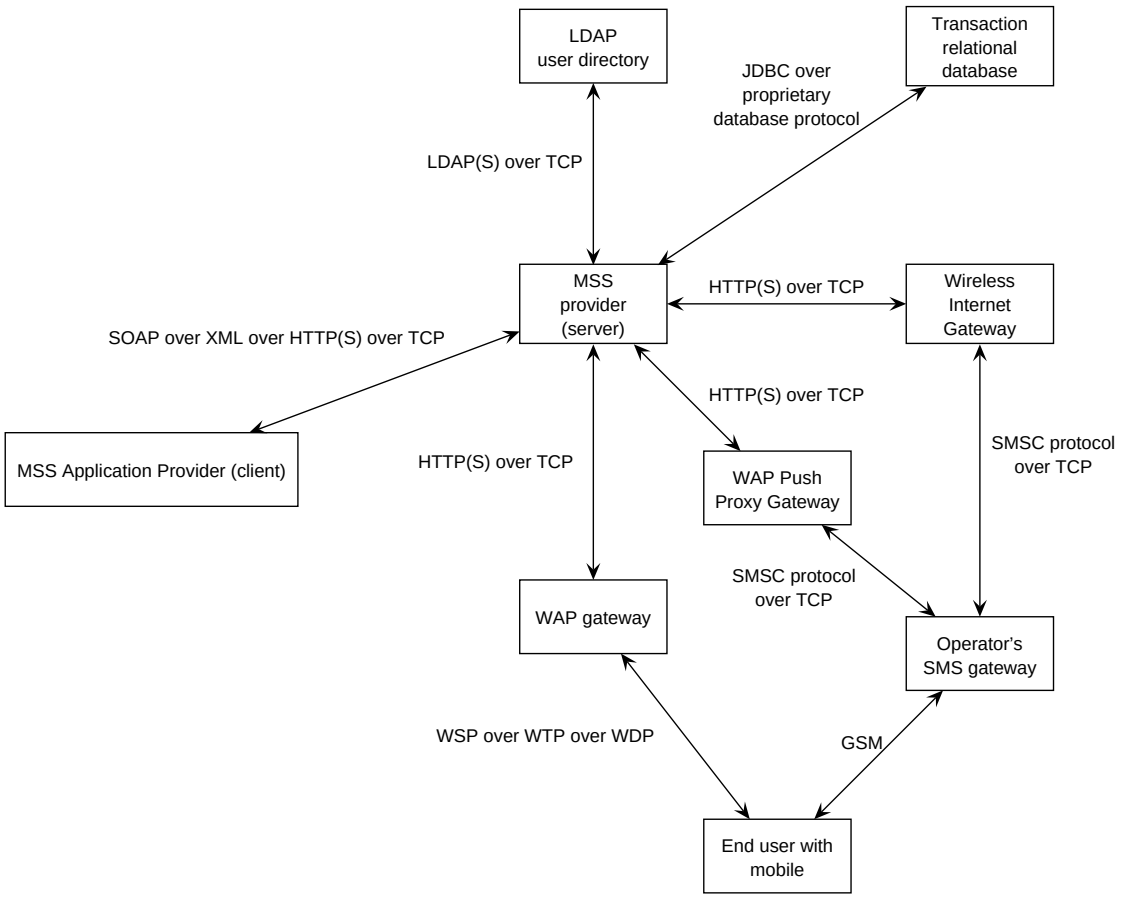
Koodi	Syy	Selitys
100	REQUEST_OK	Vastaanottaja (AP tai MSS-palvelu) hyväksyi pyynnön
400	USER_SIGN	Loppukäyttäjä on vastaanottanut allekirjoituspyynnön
408	REGISTRATION_OK	MSS_Registration-palvelu onnistui
500	SIGNATURE	Vasteviesti sisältää allekirjoituksen
501	REVOKED_CERTIFICATE	Loppukäyttäjän varmenne on sulkulistalla
502	VALID_SIGNATURE	Vasteviesti sisältää allekirjoituksen ja se on tarkastettu oikein
503	INVALID_SIGNATURE	Vasteviesti sisältää allekirjoituksen, mutta sen tarkastus epäonnistui
504	OUTSTANDING_TRANSACTION	Allekirjoitustapahtuma on kesken
600	OK with PUSH confirmation	
601	OK without PUSH confirmation	
602	NOK with PUSH confirmation	
603	NOK without PUSH confirmation	

4 ETSI:n MSS-standardin toteutus

Seuraavassa kuvataan Valimo Wireless Oy:n toteutus MSS-määrityksestä, jonka nimi on Valimo Validator - MSSP. Se toteuttaa itse MSS-määrityksen lisäksi myös FiCom Ry:n soveltamisohjeen [FiC05]. Toteutetut MSS-palvelut ovat:

- MSS_Signature
- MSS_StatusQuery
- MSS_Receipt

Kyseinen toteutus sisältää kaikki MSS-määrityksen kolme komponenttia (Aquiring Entity, Routing Entiry ja HomeMSSP) yhdessä. Kuvassa 12 on kuvattu koko järjestelmä.

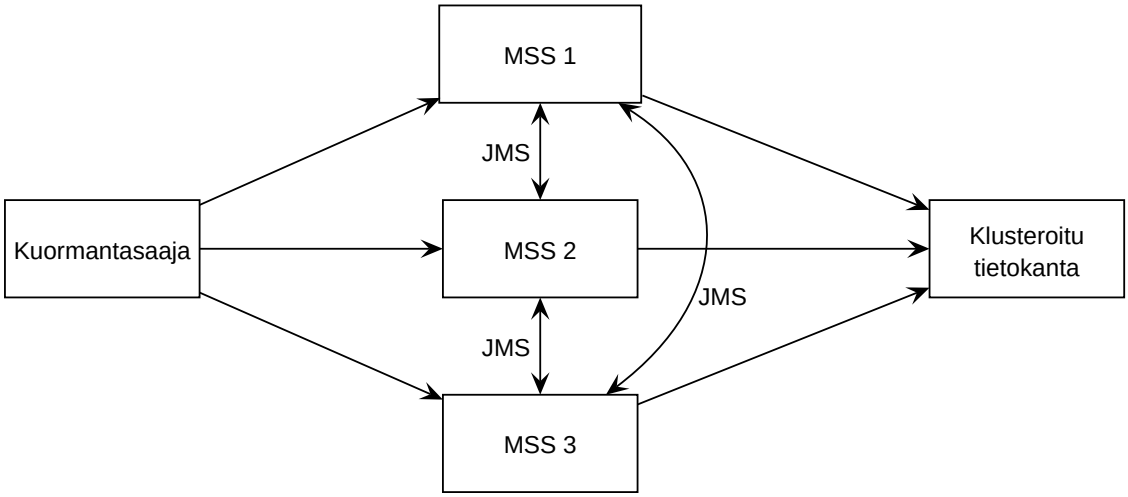


Kuva 12: MSS-järjestelmä

Kyseinen MSS-määrityksen toteutus on ohjelmoitu Java-ohjelmointikielellä, ja J2SE 1.4.2:n standardikirjastojen lisäksi toteutuksessa on käytetty useita kolmansien osapuolien vapaasti saatavilla olevia kirjastoja.

Kyseinen toteutus on ns. korkeasti käytettävä (highly available), eli useita MSS-palveluprosesseja voidaan käyttää tuomaan lisäsuorituskykyä ja/tai nostamaan käytettävyyden tasoa. Keskeisenä teknologiana tähän on prosessien väliseen kommuni-

kointiin tarkoitettu JMS-rajapinta [Sunb]. Kuvassa 13 on kuvattu korkeasti käytettävä MSS-järjestelmä jossa kuormantasaaja on kiinteässä IP-osoitteessa, ja joka jakaa kuormaa ryppään (klusterin) MSS-prosesseille. MSS-prosessien takana on ryvästetty tietokanta.



Kuva 13: MSS-klusteri

Kuvassa 12 tuettuja allekirjoituslaitteita ovat WAP 1.2.1 -yhteensopivat puhelimet WIM:n (Wiresless Identity Module) sisältävällä SIM-kortilla. WAP 1.2.1-määrityksen WMLScript-kielinen funktio Crypto.signText() tuottaa SignedContent-nimisen DER-koodatun ASN.1-rakenteen joka sisältää mm. allekirjoituksen.

Toinen vaihtoehto ovat GSM phase 2+ -yhteensopivat puhelimet (käytännössä kaikki puhelimet) joissa on SmartTrustin WIB-selain (Wireless Internet Browser) [Smab] ja RSA-valmius [Smaa]. RSA-valmiudella varustetulla SIM-kortilla on usein kaksi RSA-pistokasta (plugin), P7-pistokas ja FP-pistokas. P7-pistokas tuottaa SignedContent-rakenteen, joten se on siinä suhteessa yhteensopiva WAP 1.2.1-määrityksen kanssa. P7-pistokas näyttää loppukäyttäjälle allekirjoitettavan tekstin, kun taas FP (FingerPrint) -pistokas ei näytä allekirjoitettavaa tekstiä. FP-pistokas tuottaa DER-koodatun ASN.1-rakenteen jonka nimi on WrappedContent. WrappedContent-rakenne sisältää PKCS#1-yhteensopivan digitaalisen allekirjoituksen.

SignedContent-rakenne on kuvattu seuraavassa:

```
enum
{
  null(0),
  rsa_sha_pkcs1(1),
```

```
  ecdsa_sha_p1363(2),
  (255)
} DataSignatureAlgorithm;

struct
{
  DataSignatureAlgorithm algorithm,
  switch (algorithm)
  {
    case null:
      struct {};
    default:
      opaque signature<0..2^16-1>;
  };
} Signature;

enum
{
  implicit(0),
  sha_key_hash(1),
  wtls_certificate(2),
  x509_certificate(3),
  x968_certificate(4),
  certificate_url(5),
  (255)
} SignerInfoType;

struct
{
  SignerInfoType signer_info_type;
  switch (signer_info_type)
  {
    case implicit:
      struct {};
    case sha_key_hash:
      opaque hash[20];
    case wtls_certificate:
      WTLSCertificate;
    case x509_certificate:
      opaque x509_certificate<0..2^16-1>;
    case x968_certificate:
      opaque x968_certificate<0..2^16-1>;
    case certificate_url:
      opaque url<0..255>;
  };
} SignerInfo;

enum
{
  text(1),
  data(2),
  (255)
} ContentType;

enum
```

```
{
  false(0),
  true(1)
} Boolean;

struct
{
  ContentType content_type;
  uint16 content_encoding;
  Boolean content_present;
  switch (content_present)
  {
    case false:
      struct {};
    case true:
      opaque content<0..2^16-1>;
  };
} ContentInfo;

enum
{
  gmt_utc_time(1),
  signer_nonce(2),
  (255)
} AttributeType;

struct
{
  AttributeType attribute_type;
  switch (attribute_type)
  {
    case gmt_utc_time:
      unit8[12];
    case signer_nonce:
      opaque signer_nonce[8];
  };
} AuthenticatedAttribute;

struct
{
  unit8 version;
  Signature signature;
  SignerInfo signer_infos<0..2^16-1>;
  ContentInfo content_info;
  AuthenticatedAttribute authenticated_attributes<0..255>;
} SignedContent;
```

WrappedContent-rakenne on kuvattu seuraavassa:

```
struct
{
  opaque signature<0..2^16-1>;
} Signature;
```

```
enum
{
  implicit(0),
  sha_key_hash(1),
  certificate_url(5),
  iccid(128),
  key_usage_id(129),
  (255)
} SignerInfoType;

struct
{
  SignerInfoType signer_info_type;
  switch (signer_info_type)
  {
    case implicit:
      struct {};
    case sha_key_hash:
      opaque hash[20];
    case certificate_url:
      opaque url<0..255>;
    case iccid:
      opaque iccid[10];
    case key_usage_id:
      uint8;
  };
} SignerInfo;

struct
{
  unit8 version;
  Signature signature;
  SignerInfo signer_infos<0..2^16-1>;
} WrappedContent;
```

Yksi MSS_Signature-tapahtuma, eli allekirjoituksen pyytäminen, menee pääpiirteittäin seuraavasti, kuten kuvassa 12 on esitetty:

1. Application Provider (AP) eli MSS-asiakas lähettää palvelupyynnön (MSS_SignatureRequest).

2. MSS-palvelu tallettaa tietokantaan merkinnän että tapahtuma on aloitettu, ja lähettää allekirjoituspyynnön push-viestinä joko Push Proxy Gatewayn (jos kyseessä on WAP 1.2.1 -allekirjoituslaite) tai Wireless Internet Gatewayn (jos kyseessä on WIB-allekirjoituslaite) kautta loppukäyttäjän puhelimeen.

3. Loppukäyttäjä syöttää PIN-koodinsa ja puhelin laskee digitaalisen allekirjoituksen.

4. Allekirjoitus saapuu puhelimelta MSS-palvelimelle joko WAP-yhdyskäytävän tai WIG-yhdyskäytävän läpi.

5. MSS-palvelu hakee käyttäjän varmenteen LDAP-hakemistosta, validoi sen ja tarkistaa allekirjoituksen varmenteen julkisella avaimella. Lopuksi tallennetaan tietokantaan merkintä onnistuneesta tai epäonnistuneesta tapahtumasta.

6. MSS-palvelu palauttaa vastauksen (MSS_SignatureResponse), joka sisältää mm. loppukäyttäjän digitaalisen allekirjoituksen ja varmenteen, MSS-asiakkaalle eli AP:lle.

4.1 Eri viestintämoodien toteutus

Seuraavissa kappaleissa on esitetty kontrollin kulku järjestelmän eri modulien välillä jokaisen eri viestintämoodin kohdalla.

4.1.1 Synkroninen asiakas-palvelin -moodi

Synkronisen asiakas-palvelin -viestintämoodin MSS_Signature-tapahtuma etenee MSS-toteutuksessa seuraavasti:

1. AP tekee MSS_Signature-kutsun lähettämällä MSS_SignatureRequest-viestin MSS-SOAP-moduliin.

2. MSS-SOAP-moduli kutsuu Transaction Data Manager -modulia luodakseen uuden rivin tietokantaan tapahtumaa varten.

3. Signing Gateway -modulin avustuksella loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö.

4. MSS-SOAP-moduli menee JMS-moduliin nukkumaan, odottamaan tapahtuman valmistumista.

5. Allekirjoitus saapuuu WIG/WAP-yhdyskäytävältä Signing Gateway -modulille, joka purkaa allekirjoituksen.

6. Jäsennetty allekirjoitus annetaan Bridge-modulille.

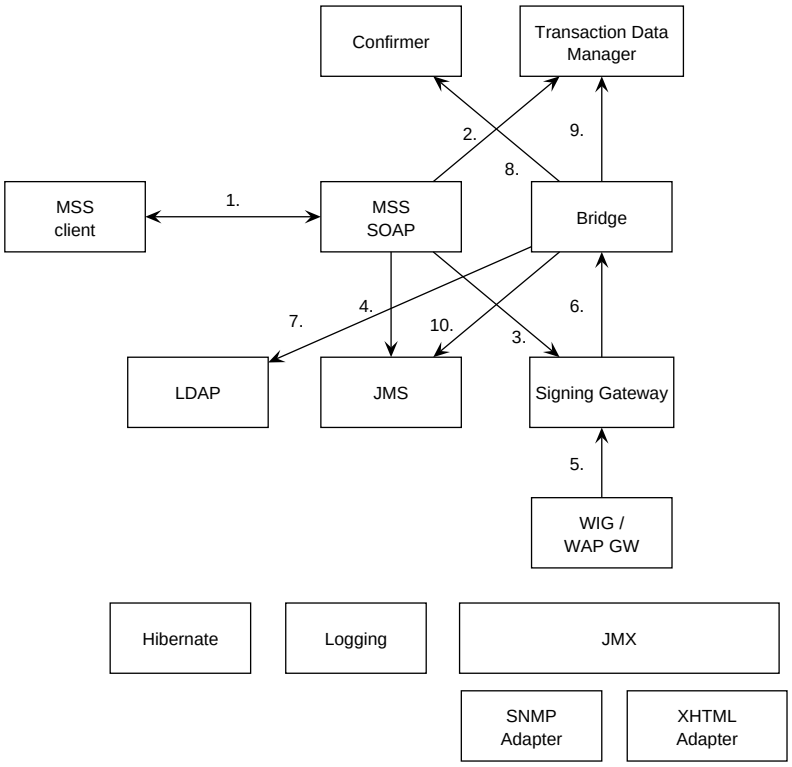
7. Bridge-moduli hakee LDAP-modulia käyttäen loppukäyttäjän varmenteen.

8. Seuraavaksi allekirjoitus tarkistetaan loppukäyttäjän varmenteen julkisella avaimella sekä varmenne validoidaan Confirmer-modulilla.

9. Tapahtuman tila on nyt valmis, tulos tallennetaan tietokantaan

10. Lähetetään JMS-signaali, että tapahtuma on valmis, jolloin kohdassa 4. nukkumaan jäänyt säie herää jossakin MSS-palveluprosessissa, tapahtuman tulos luetaan tietokannasta, ja AP:lle palautetaan MSS_SignatureResponse.

Synkroninen asiakas-palvelin -viestintämoodi on kuvattu kuvassa 14.



Kuva 14: Synkroninen asiakas-palvelin -viestintämoodi

4.1.2 Asynkroninen asiakas-palvelin -moodi

Asynkronisen asiakas-palvelin -viestintämoodin MSS_Signature-tapahtuma etenee MSS-toteutuksessa seuraavasti:

1. AP tekee MSS_Signature-kutsun lähettämällä MSS_SignatureRequest-viestin MSS-SOAP-moduliin.

2. MSS-SOAP-moduli kutsuu Transaction Data Manager -modulia luodakseen uuden rivin tietokantaan tapahtumaa varten.

3. Signing Gateway -modulin avustuksella loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö.

4. AP tekee MSS_StatusQuery-kyselyn olisiko tapahtuma jo valmis, mutta se ei ole vielä valmis.

5. Allekirjoitus saapuuu WIG/WAP-yhdyskäytävältä Signing Gateway -modulille, joka purkaa allekirjoituksen.

6. Jäsennetty allekirjoitus annetaan Bridge-modulille.

7. Bridge-moduli hakee LDAP-modulia käyttäen loppukäyttäjän varmenteen.

8. Seuraavaksi allekirjoitus tarkistetaan loppukäyttäjän varmenteen julkisella avaimella sekä varmenne validoidaan Confirmer-modulilla.

9. Tapahtuman tila on nyt valmis, tulos tallennetaan tietokantaan

AP voisi kysyä nyt tapahtuman tilaa, ja se olisi valmis ja allekirjoitus olisi noudettavissa.

Asynkroninen asiakas-palvelin -viestintämoodi on kuvattu kuvassa 15.

4.1.3 Asynkroninen palvelin-palvelin -moodi

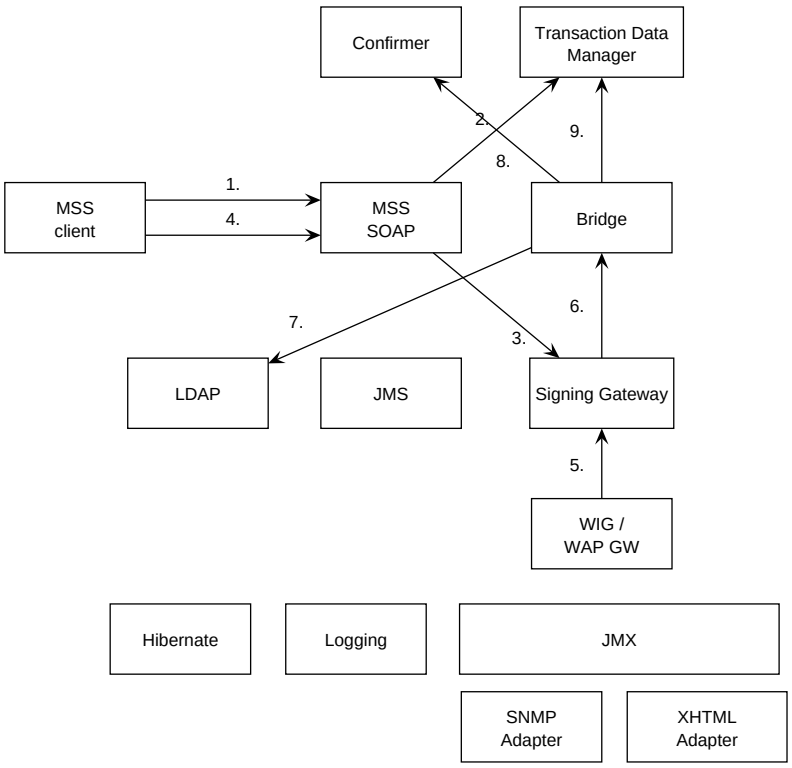
Asynkronisen palvelin-palvelin -viestintämoodin MSS_Signature-tapahtuma etenee MSS-toteutuksessa seuraavasti:

1. AP tekee MSS_Signature-kutsun lähettämällä MSS_SignatureRequest-viestin MSS-SOAP-moduliin.

2. MSS-SOAP-moduli kutsuu Transaction Data Manager -modulia luodakseen uuden rivin tietokantaan tapahtumaa varten.

3. Signing Gateway -modulin avustuksella loppukäyttäjän puhelimeen lähetetään allekirjoituspyyntö.

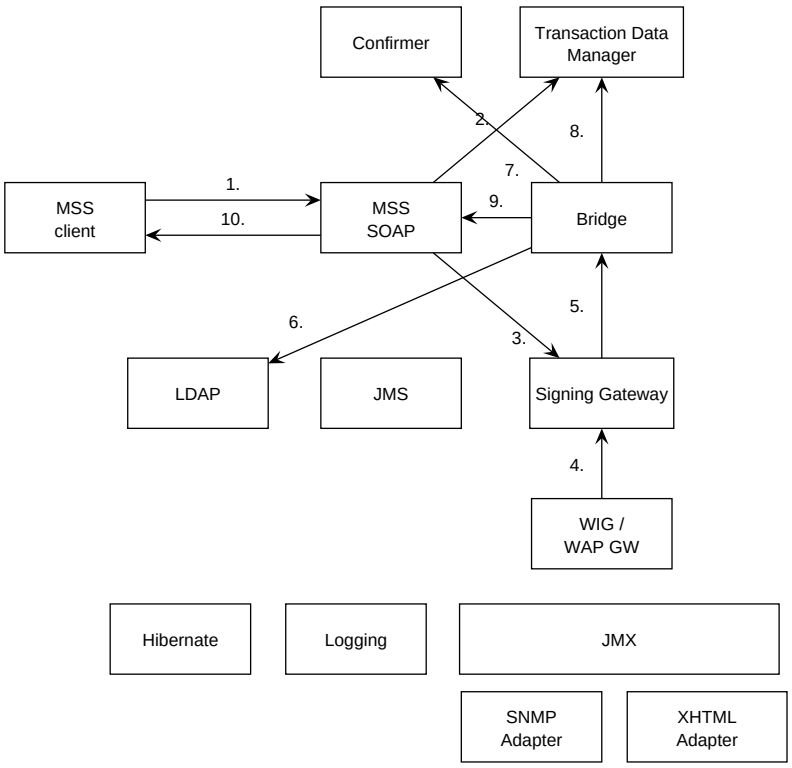
4. Allekirjoitus saapuuu WIG/WAP-yhdyskäytävältä Signing Gateway -modulille, joka purkaa allekirjoituksen.



Kuva 15: Asynkroninen asiakas-palvelin -viestintämoodi

5. Jäsennetty allekirjoitus annetaan Bridge-modulille.
6. Bridge-moduli hakee LDAP-modulia käyttäen loppukäyttäjän varmenteen.
7. Seuraavaksi allekirjoitus tarkistetaan loppukäyttäjän varmenteen julkisella avaimella sekä varmenne validoidaan Confirmer-modulilla.
8. Tapahtuman tila on nyt valmis, tulos tallennetaan tietokantaan
9. Ilmoitetaan MSS-SOAP-modulille että tapahtuma olisi valmis ja pitäisi tehdä takaisinkutsu (callback).
10. AP:lle vastataan tekemällä MSS_NotificationRequest-takaisinkutsu, joka sisältää MSS_SignatureResponse-viestiä vastaavat tiedot.

Asynkroninen palvelin-palvelin -viestintämoodi on kuvattu kuvassa 16.



Kuva 16: Asynkroninen palvelin-palvelin -viestintämoodi

4.2 Tarvittavat kolmansien osapuolten kirjastot

Valimo Wireless Oy:n MSS-toteutus hyödyntää useita kolmansien osapuolten vapaasti saatavia kirjastoa. Seuraavassa on lueteltu olennaisimmat.

Apache Axis [Apa] on SOAP 1.2 -yhteensopiva SOAP-toteutus. Sen **wsdl2java**-työkalulla tuotetaan MSS-palvelun WSDL-kuvauksesta Java-kieliset palvelinpään luurangot (skeletons) ja asiakaspään tyngät (stubs). Lisäksi sitä käytetään ajonaikaisena SOAP-moottorina.

Bouncy Castle -kryptokirjastoa tarvitaan DER-koodattujen ASN.1-rakenteiden purkamiseen allekirjoitusten vastaanotossa.

Doug Lean util.concurrent-kirjasto [Lea04] sisältää tehokkaita ja yleiskäyttöisiä toteutuksia eri synkronointiprimitiiveistä, kuten semaforeista, lukoista ja jonoista. Tämä Java-pakkaus on integroitu J2SE5:een (pakkaus javax.util.concurrent), mutta tässä MSS-toteutuksessa ei voida käyttää kuin J2SDK:n versiota 1.4.2 ulkoisten rajoitteiden vuoksi, joten tämä kirjasto on tarpeen erillisenä.

Hibernate [Hib] on kirjasto, jolla oliot saadaan näppärästi relaatiotietokantaan. Sen

etuina on se, ettei tarvitse kirjoittaa tietokantamoottorikohtaista ohjelmakoodia (kuten usein täytyy vaikka JDBC-rajapinta pyrkiikin olemaan tietokantariippumaton, mutta eroja silti ilmenenee). Lisäksi käytetään XDoclet-työkalua [XDo], jolla saadaan sopivasti koristelluista Java-lähdekoodeista luotua Hibernaten tarvitsemat XML-pohjaiset kuvaustiedostot, joissa kerrotaan miten jonkin luokan oliot kuvautuvat tietokantaan. Muuten nämä XML-pohjaiset kuvaustiedostot joutuisi kirjoittamaan itse.

MSS-toteutus käyttää Java Messaging Service -rajapintaa (JMS) [Sunb] MSS-prosessien väliseen kommunikointiin. MSS-toteutus ei ota kantaa siihen, miten JMS-rajapinta on toteutettu, toteutuksia on monia, ja MSS-toteutuksen asentaja voi valita tarkoitukseensa sopivimman JMS-toteutuksen. JMS-rajapintaa käytetään ns. publish-subscribe -moodissa, jossa prosessi julkaisee viestin kanavaan, ja josta kiinnostuneet prosessit voivat sitten poimia viestin.

Java Management Extensions (JMX) [Suna] mahdollistaa Java-olioiden tilan katsomisen, muuttamisen ja operaatioiden kutsumisen ulkopuolelta. Tällaisia Java-olioita kutsutaan MBeaneiksi. Jokainen MBean siis paljastaa itsestään attribuutteja, jotka voivat olla read-only tai read-write, sekä operaatioita joita voidaan JMX:n avulla kutsua. Tietojen paljastaminen tapahtuu tekemällä ns. MBean-rajapintoja jotka Java-luokka sitten toteuttaa.

Kaikki järjestelmän lokitietojen hallinta hoidetaan log4j-kirjastolla [Gül02].

Varmenteiden ja sulkulistojen (Certificate Revocation List, CRL) hakuun LDAP-hakemistoista käytetään Novellin LDAP-kirjastoa [Nov].

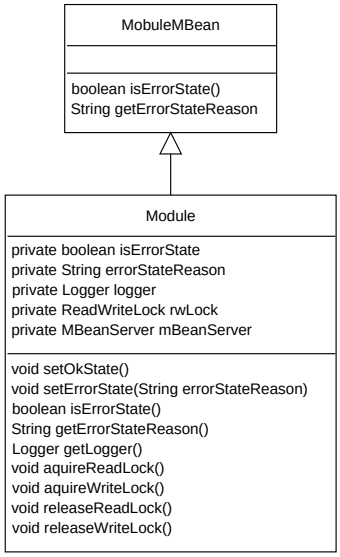
4.3 Modulijako

Seuraavissa kappaleissa esitetään MSS-toteutuksen modulien osalta niiden toiminnallisuus luokkakaavioin, rajapintakuvauksin ja kirjallisin selityksin.

4.4 Module-luokka

Module-luokka tarjoaa kaikille muille moduleille palveluita, ts. muut modulit periytyvät tästä luokasta. Se tarjoaa mm. lokituspalvelut sekä konstrutorissaan rekiströinnin JMX-palvelimeen. Lisäksi se tarjoaa Doug Lean luku/kirjoituslukon synkronointia varten. Lisäksi jokainen moduli voi olla virhetilassa, ja tätä varten Module-luokassa on lippu joka kertoo virhetilan, ja jos kyseessä on virhetila, kertoo jäsen-

muuttuja myös virhetilan syyn. ModuleMBean-rajapinta antaa mahdollisuuden kysyä mahdollista virhetilaa JMX:n kautta. Luokkakaavio on kuvassa 17:



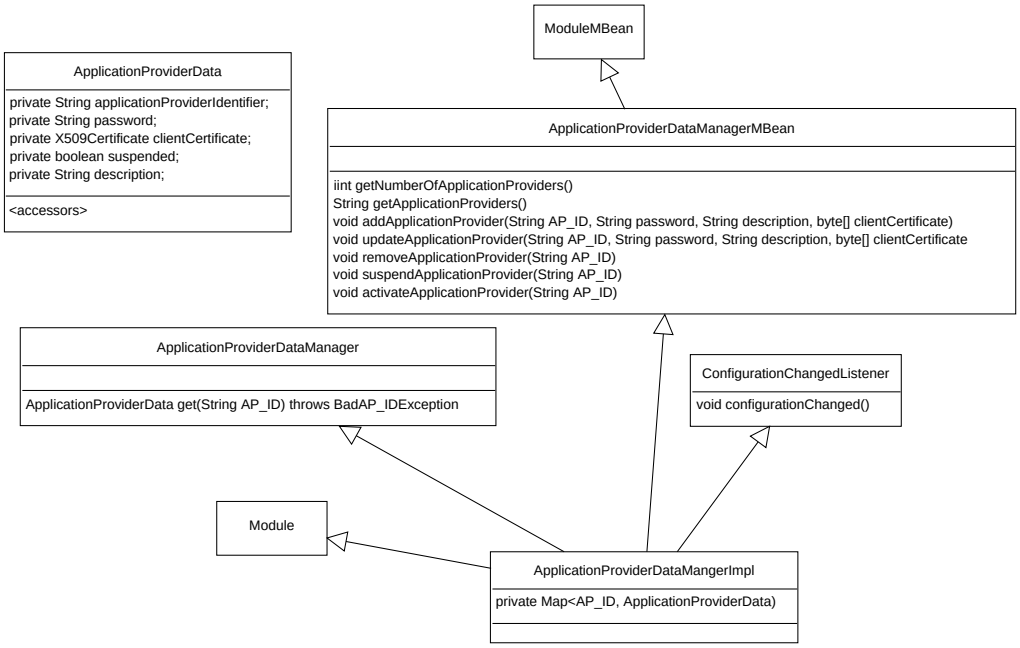
Kuva 17: Module-luokan luokkakaavio

4.5 Application Provider Data Manager -moduli

Tämä moduli tarjoaa näkymän AP-tietokantaan. Sillä voi hakea AP:n tiedot kun tiedossa on AP:n tunniste. Luokkakaavio on esitetty kuvassa 18. Tietoja ei haeta tietokannasta kysyttäessä, vaan ne ovat hajautustaulukossa välimuistissa. Kun moduli saa tiedon MSS-järjestelmän konfiguraation muuttumisesta ConfigurationChangedListener-rajapinnan kautta, se lataa kaikki AP:t tietokannasta välimuistiinsa. Näin säästetään tietokantahakuja. ApplicationProviderData-luokka on luokka jonka olioita tallennetaan ja ladataan tietokantaan Hibernaten avustuksella.

Seuraavassa on Application Provider Data Manager -modulin rajapintakuvaus JavaDoc-muodossa:

```
public interface ApplicationProviderDataManager
{
  /**
   * Returns information of a AP based on the AP_ID.
   *
   * @param applicationProviderIdentifier
   *  AP_ID.
   * @return
   *  The data of the AP.
```



Kuva 18: Application Provider Manager -modulin luokkakaavio

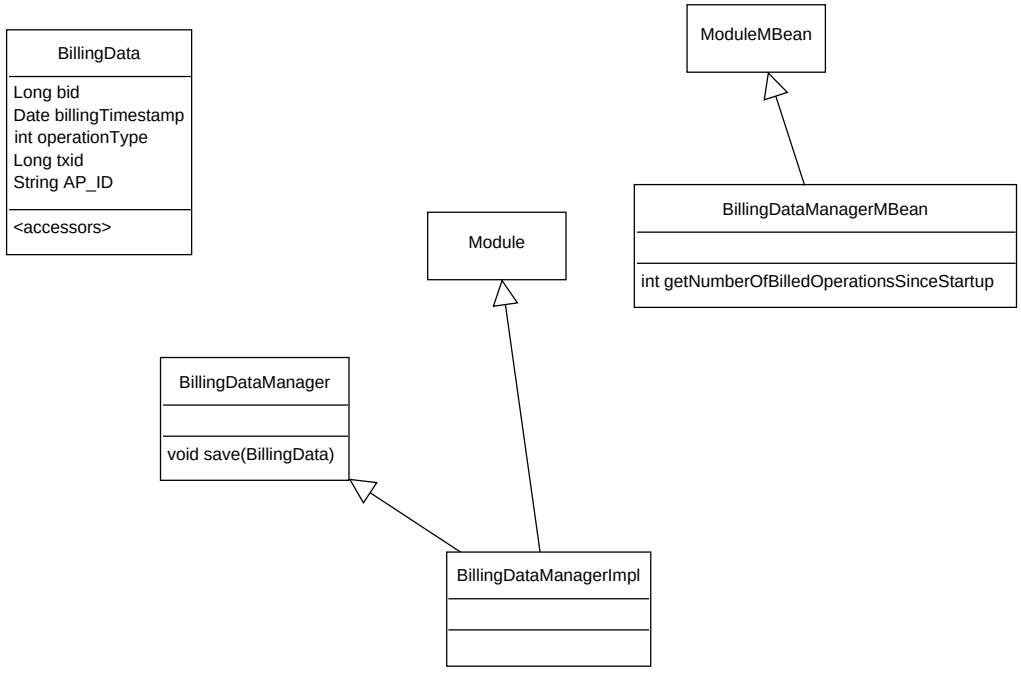
```
   * @throws BadApplicationProviderException
   *  When such AP didn't exist.
   */
  ApplicationProviderData get(String applicationProviderIdentifier) throws BadApplicationProviderException;
}
```

4.6 Billing Data Manager -moduli

Billing Data Manager -moduli tarjoaa palveluita joilla voidaan laskuttaa asiakasta - eli AP:ta - sen tekemistä MSS_Signature-palvelukutsuista. Sen rajapinta on seuraavassa JavaDoc-muodossa:

```
public interface BillingDataManager
{
  /**
   * Inserts a new row to the database.
   *
   * @param billingData
   *  Billing data object to be saved.
   * @throws HibernateException
   *  When something went wrong in the database.
   */
  void save(BillingData billingData) throws HibernateException;
}
```

Billing Data Manager -modulin luokkakaavio on kuvassa 19:



Kuva 19: Billing Data Manager -modulin luokkakaavio

4.7 Bridge-moduli

Bridge-moduli ottaa allekirjoituksia vastaan Signing Gateway -modulilta, tarkistaa allekirjoituksen, validoi varmenteen (molemmat toimenpiteet käyttäen Confirmer-modulia) ja lopuksi suorittaa valitusta viestintämoodista riipppuvaiset lopputoimenpiteet. Esimerkiksi synkronisessa asiakas-palvelin -moodissa tapahtuman lopuksi täytyy lähettää JMS-signaali JMS-modulia käyttäen, jotta säie joka odottaa JMS-modulissa tämän tapahtuman valmistumista, heräisi.

Seuraavassa on Bridge-modulin rajapintakuvaus JavaDoc-muodossa:

```
public interface Bridge
{
  /**
   * Receives a PKCS#7 signature from signing device. If in validating mode,
   * transaction state will be updated accordingly.
   * Transaction status is returned to signing device so it
   * can send feedback.
   *
   * @param pkcs7Constructor
   *  An algorithm that constructs a PKCS#7 object when given cleartext and signer's certificate.
```

```
 * @param urlToSignerCertificate
 *  LDAP URL to signer's certificate.
 * @param ldapUsername
 *  Username to LDAP server or null.
 * @param ldapPassword
 *  Password to LDAP server or null.
 * @param transactionData
 *  Transaction data of this transaction.
 * @throws InternalErrorException
 *  An internal error happened, HTTP 500 should be returned to the signing device.
 * @throws IPCException
 *  JMS error happened.
 * @throws HibernateException
 *  In case of database trouble.
 */
ProcessingResult handlePKCS7Signature(PKCS7Constructor pkcs7Constructor,
                                      String urlToSignerCertificate,
                                      String ldapUsername,
                                      String ldapPassword,
                                      TransactionData transactionData)
    throws
      InternalErrorException,
      IPCException,
      HibernateException;

/**
 * Receives a PKCS#1 signature from signing device.
 *
 * @param pkcs1
 *  PKCS#1 data.
 * @param transactionData
 *  Transaction data of this transaction.
 * @throws InternalErrorException
 *  An internal error happened, HTTP 500 should be returned to the signing device.
 * @throws IPCException
 *  JMS error happened.
 * @throws HibernateException
 *  In case of database trouble.
 */
ProcessingResult handlePKCS1Signature(byte[] pkcs1,
                                      TransactionData transactionData)
    throws
      InternalErrorException,
      IPCException,
      HibernateException;

/**
 * Receives a PKCS#1 signature from signing device.
 *
 * @param pkcs1
 *  PKCS#1 data.
 * @param transactionData
 *  Transaction data of this transaction.
 * @param ldapUsername
 *  Username to LDAP server or null.
```

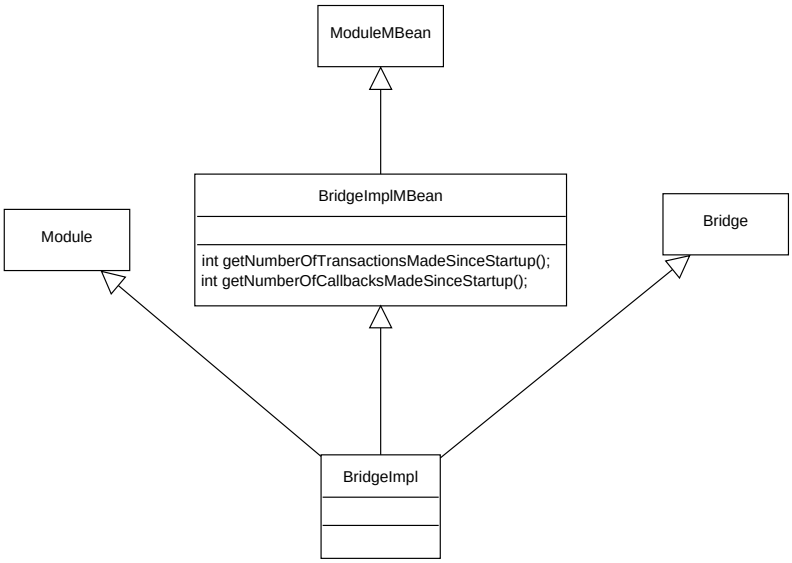
```
   * @param ldapPassword
   *  Password to LDAP server or null.
   * @throws InternalErrorException
   *  An internal error happened, HTTP 500 should be returned to the signing device.
   * @throws IPCException
   *  JMS error happened.
   * @throws HibernateException
   *  In case of database trouble.
   */
  ProcessingResult handlePKCS1Signature(byte[] pkcs1,
                                        TransactionData transactionData,
                                        String urlToSignerCertificate,
                                        String ldapUsername,
                                        String ldapPassword)
      throws
        InternalErrorException,
        IPCException,
        HibernateException;

  /**
   * A timeout happened on a MSS_Signature transaction.
   *
   * @param transactionData
   *  Transaction data of timed out transaction.
   *
   * @return
   * @throws InternalErrorException
   *  An internal error happened, HTTP 500 should be returned to the signing device.
   */
  ProcessingResult handleTimeout(TransactionData transactionData) throws InternalErrorException;
}
```

Seuraavassa on Bridge-modulin luokkakaavio kuvassa 20. Bridge-rajapinta on jo edellä esitelty joten se on jätetty luokkakaaviosta pois. BridgeImpl siis perii Module-luokan ja toteuttaa sekä Bridge-rajapinnan että BrigeImplMBean-rajapinnan. BridgeImplMBean-rajapinta kertoo mitä attribuutteja tai operaatioita JMX:n kautta tarjotaan näkyville.

4.8 Confirmer-moduli

Confirmer-moduli tarkistaa digitaalisia allekirjoituksia, validoi varmenteita ja hakee allekirjoittajien varmenteita. Varmenteen validointi tapahtuu seuraavasti:

1. Onko varmenne voimassa juuri nyt?
2. Onko varmenteen myöntäjä allekirjoittanut sen? Ts. tarkistuuko varmenteen digitaalisen allekirjoitus myöntäjän varmenteen julkista avainta vasten?



Kuva 20: Bridge-modulin luokkakaavio

3. Onko varmenne sulkulistalla (Certificate Revocation List, CRL)?

Seuraavassa on Confirmer-modulin rajapinta JavaDoc-muodossa:

```
public interface Validation
{
  /**
   * Validates a user certificate.
   *
   * @param x509Certificate
   *  The certificate to be validated.
   * @throws SignerCertificateRevokedException
   *  The certificate was revoked.
   * @throws SignerCertificateNotTrustedException
   *  Any CA didn't trust certificate.
   * @throws ConfirmerException
   *  Other problem.
   */
  void validate(X509Certificate x509Certificate)
  throws
  SignerCertificateRevokedException,
  SignerCertificateNotTrustedException,
  ConfirmerException;
}

public interface Verification
{
  /**
   * Verifies a signature.
   *
   * @param clearTextData
```

```
   *  What was signed.
   * @param signature
   *  The signature to be verified.
   * @param signerCertificate
   *  To which public key against the signature is verified.
   * @throws SignatureException
   *  When signature verification fails.
   */
  void verify(ClearTextData clearTextData,
              VerifiableSignature signature,
              X509Certificate signerCertificate) throws SignatureException;
}

public interface Confirmer extends Verification, Validation
{
  /**
   * Returns signer's certificate.
   *
   * @param urlToSignerCertificate
   *  LDAP URL to signer certificate.
   * @param ldapUsername
   *  Username to LDAP server or null.
   * @param ldapPassword
   *  Password to LDAP server or null.
   * @return
   *  The signer's certificate.
   * @throws MalformedURLException
   *  Bad LDAP URL.
   * @throws CertificateException
   *  Certificate object couldn't be constructed from LDAP data.
   * @throws LDAPException
   *  Problem communicating with LDAP server.
   */
  X509Certificate getSignerCertificate(String urlToSignerCertificate, String ldapUsername, String ldapPassword)
  throws
  MalformedURLException,
  CertificateException,
  LDAPException;
}
```

Itse varmenteiden myöntäjät muodostavat verkon. Verkossa voi olla kolmenlaisia solmuja:

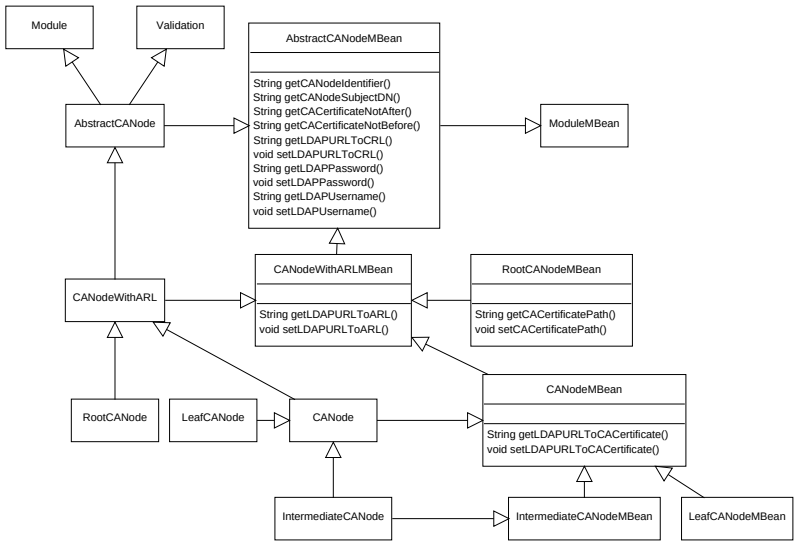
- IntermediateCANode - alivarmentajille
- LeafCANode - alivarmentajille
- RootCANode - juurivarmentajille

Seuraavat kolme luokkaa ovat vain helpottamassa luokkahierarkiaa, niillä ei ole varsinaista käyttöä itse Confirmer-modulin konfiguroinnissa:

- AbstractCANode
- CANodeWithARL
- CANode

Jokaista CA-solmu-luokkaa vastaa vastaava Data-loppuinen (esimerkiksi RootCA-NodeData) luokka joka sisältää tiedot jota kyseinen solmu tarvitsee. Nämä Data-luokat säilötään ja ladataan tietokannan avulla muistiin käyttäen Hibernatea. Jokainen CA-solmu pitää siis sisällään viitteen vastaavaan Data-luokan ilmentymään. Tässä on siis kyseessä Confirmer-modulin konfiguraatio joka on tietokannassa.

Tämä luokkakaavio on kuvattu kuvassa 21, siitä puuttuu Data-luokat mutta niiden hierarkia on vastaava kuten ym. solmuluokkienkin:



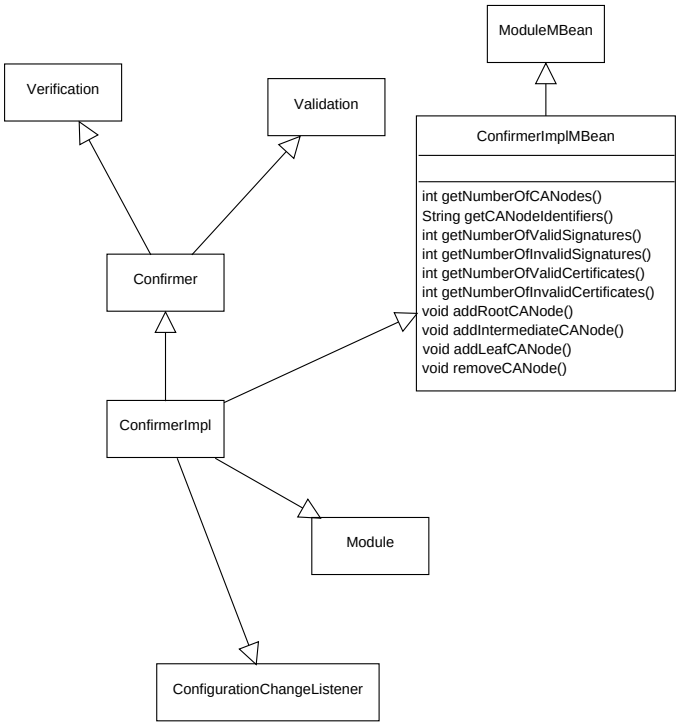
Kuva 21: Confirmer -modulin solmut

Osa Confirmer-modulin luokkakaaviota on esitetty seuraavassa 22:

Confirmer-moduli käyttää monia eri poikkeuksia eri tilanteissa. Luokkakaaviossa 23 on kuvattu ne. Poikkeusten nimet lienevät itsedokumentoivia.

4.9 Hibernate-moduli

Hibernate-moduli tarjoaa muille moduleille mahdollisuuden tallettaa (INSERT), päivittää (UPDATE) ja hakea (SELECT) Java-olioita relaatiotietokannassa. Hibernate-



Kuva 22: Confirmer -modulin luokkakaaviota

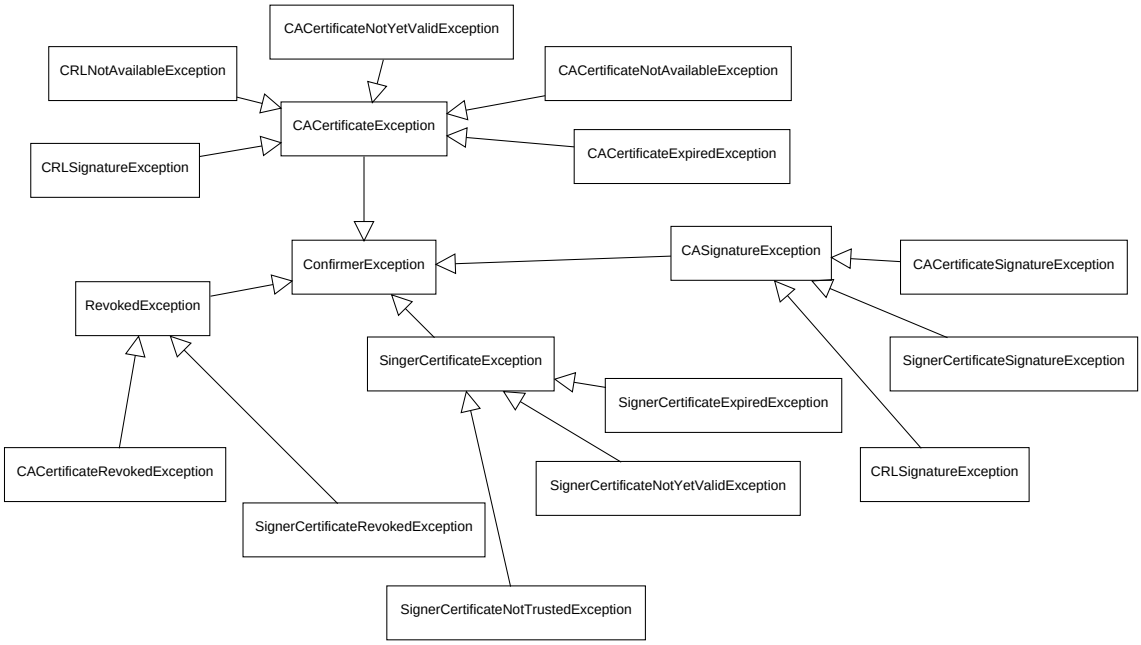
moduli on ns. Singleton-olio [GHJV95], eli siitä on olemassa vain yksi instanssi, jonka viite on saatavilla Main-Singleton -olion kautta.

Hibernate-modulia käytettäessä voi käyttäjä antaa sen hallita transaktiorajoja (withTx-metodit), tai sitten käyttäjä hoitaa transaktiorajat itse beginTx()- ja endTx()-metodeja kutsumalla.

Hibernate-moduli on oikeastaan vain kääröluokka (wrapper) Hibernate:lle, ja kääröluokka on tehty paremmaksi sopimaan MSS-toteutuksen käyttötavoille. SessionAndTransaction-luokka on säiliöluokka, joka sisältää ainoastaan Hibernate-session ja Hibernate-transaktion.

Seuraavassa on Hibernate-modulin rajapinta JavaDoc-muodossa:

```
public interface Hibernate
{
  /**
   * Begins a transaction.
   *
   * @param log
   *  Whether it should be logged how long a beginTx() takes.
   * @return
   *  SessionAndTransaction object that is to be given to endTx().
   * @throws HibernateException
```



Kuva 23: Confirmer -modulin poikkeusten luokkakaavio

```
 *  In case of database trouble.
 */
public SessionAndTransaction beginTx(boolean log) throws HibernateException;

/**
 * Ends a transaction.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param commit
 *  Whether to commit or rollback the transaction.
 * @param log
 *  Whether it should be logged how long a endTx() takes.
 * @throws HibernateException
 *  In case of database trouble.
 */
public void endTx(SessionAndTransaction sessionAndTransaction, boolean commit, boolean log) throws
  HibernateException;

/**
 * Returns a list of Hibernated objects based
 * on HQL query. Transactions are managed by this
 * method.
 *
 * @param hql
 *  The query string in HQL.
 * @return
 *  A list of Hibernated objects.
 *
```

```
 * @throws HibernateException
 *  In case of database trouble.
 */
public List loadWithTx(String hql) throws HibernateException;

/**
 * Returns a list of Hibernated objects based
 * on HQL query. Transactions are not managed by this method.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param hql
 *  The query string in HQL.
 * @param log
 *  Whether it should be logged how long a load() takes.
 * @return
 *  A list of Hibernated objects.
 * @throws HibernateException
 *  In case of database trouble.
 */
public List load(SessionAndTransaction sessionAndTransaction, String hql, boolean log) throws HibernateException;

/**
 * Returns a list of Hibernated objects based
 * on HQL query. Transactions are not managed by this method.
 *    *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param hql
 *  The query string in HQL.
 * @param params
 *  An array of parameters in the HQL query.
 * @param types
 *  An array of Hiberate types of the objects to be loaded.
 * @param log
 *  Whether it should be logged how long a load() takes.
 * @return
 *  A list of Hibernated objects.
 * @throws HibernateException
 *  In case of database trouble.
 */
public List load(SessionAndTransaction sessionAndTransaction,
                 String hql,
                 Object[] params,
                 Type[] types,
                 boolean log) throws HibernateException;

/**
 * Returns a Hibernated object based
 * on a class and a primary key. Transactions are not managed by this method.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param clazz
 *  Class of the object to be loaded.
```

```
 * @param primaryKey
 *  Primary key for the object.
 * @param log
 *  Whether it should be logged how long a load() takes.
 * @return
 *  The Hibernated object.
 * @throws HibernateException
 *  In case of database trouble.
 */
public Object load(SessionAndTransaction sessionAndTransaction, Class clazz, Serializable primaryKey, boolean log)
  throws HibernateException;

/**
 * Returns a Hibernated object based
 * on a class and a primary key. Transactions are managed by this method.
 *
 * @param clazz
 *  Class of the object to be loaded.
 * @param primaryKey
 *  Primary key for the object.
 * @param log
 *  Whether it should be logged how long a load() takes.
 * @return
 *  The Hibernated object.
 * @throws HibernateException
 *  In case of database trouble.
 */
public Object loadWithTx(Class clazz, Serializable primaryKey, boolean log) throws HibernateException;

/**
 * Deletes a number of rows based on given HQL query.
 *
 * @param hql
 * @return
 *  The number of rows deleted.
 * @throws HibernateException
 *  In case of database trouble.
 */
public int deleteWithTx(String hql) throws HibernateException;

/**
 * Deletes a number of rows based on given HQL query.
 * Does not manage transactions.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param hql
 *  The HQL query.
 * @return The number of rows deleted.
 * @throws HibernateException
 */
public int delete(SessionAndTransaction sessionAndTransaction, String hql) throws HibernateException;

/**
 * Deletes a (Hibernated) persistent object (makes it transient)
```

```
 * This method manages transactions by itself.
 *
 * @param object
 *  Object to be removed from the database.
 * @throws HibernateException
 *  In case of database trouble.
 */
public void deleteWithTx(Object object) throws HibernateException;

/**
 * Deletes a (Hibernated) persistent object (makes it transient)
 * This method does not manage transactions by itself.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param object
 *  Object to removed from the database.
 * @throws HibernateException
 *  In case of database trouble.
 */
public void delete(SessionAndTransaction sessionAndTransaction, Object object) throws HibernateException;

/**
 * Inserts a new object, makes it persistent
 * This methods manages transactions by itself.
 *
 * @param object
 *  Object to be inserted.
 * @throws HibernateException
 *  In case of database trouble.
 */
public void saveWithTx(Object object) throws HibernateException;

/**
 * Inserts a new object, makes it persistent.
 * This methods does not manage transactions by itself.
 *
 * @param sessionAndTransaction
 *  Holds transaction state.
 * @param object
 *  Object to be inserted.
 * @throws HibernateException
 *  In case of database trouble.
 */
public void save(SessionAndTransaction sessionAndTransaction, Object object) throws HibernateException;

/**
 * Updates a (Hibernated) persistent object.
 * This methods does not manage transactions by itself.
 *
 * @param object
 *  Object to be inserted.
 * @throws HibernateException
 *  In case of database trouble.
 */
```

```
  public void updateWithTx(Object object) throws HibernateException;

  /**
   * Updates a (Hibernated) persistent object.
   * This methods does not manage transactions by itself.
   *
   * @param sessionAndTransaction
   *  Holds transaction state.
   * @param object
   *  Object to be updated.
   * @throws HibernateException
   *  In case of database trouble.
   */
  public void update(SessionAndTransaction sessionAndTransaction, Object object) throws HibernateException;
}
```

4.10 JMS-moduli

JMS-moduli mahdollistaa MSS-prosessien välisen kommunikoinnin käyttäen JMS-rajapintaa:

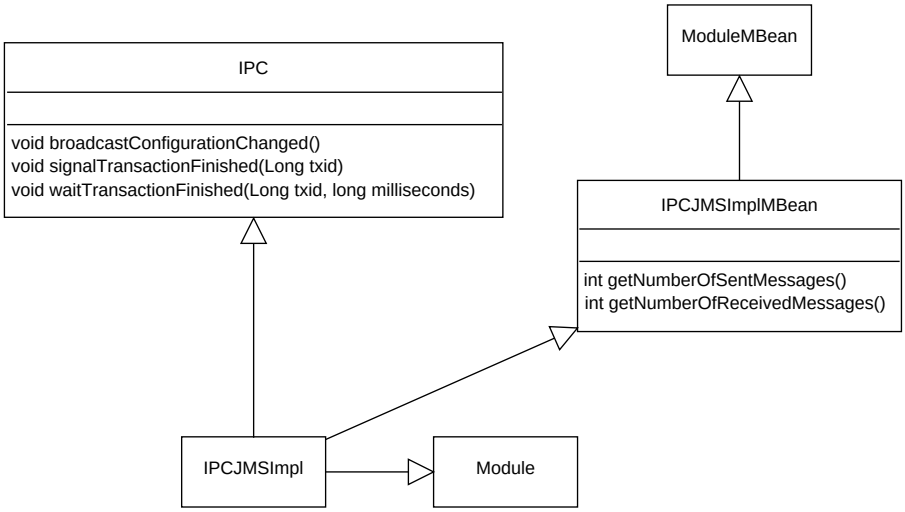
1. Lähettää "tapahtuma on nyt valmis-signaaleja
2. Odottaa "tapahtuma on nyt valmis-signaaleja
3. Lähettää "konfiguraatio on muuttunut, päivittäkää konfiguraationne-signaaleja

Seuraavassa on kuvassa JMS-modulin rajapinta:

```
/**
 * Defines API for MSSP related communication
 * between JVM's (and inside one JVM).
 */
public interface IPC
{
  /**
   * Broadcasts a message to all interested parties,
   * that configuration data has been changed, p
   * lease refresh your local configuration data
   * (or please invalidate your cache).
   *
   * @throws IPCException When something went wrong in communications.
   */
  void broadcastConfigurationChanged() throws IPCException;

  /**
   * Broadcasts a message that the state of a transaction
   * has been changed (it has been completed in an way,
```

```
   * also timeout is one of the reasons).
   *
   * @param txid Transaction identifier of transaction that was completed.
   * @throws IPCException When something went wrong in communications.
   */
  void signalTransactionFinished(Long txid) throws IPCException;

  /**
   * Waits until a transaction is completed, or waiting timeout
   * is reached.
   *
   * @param txid         Transaction identifier of a transaction that the waiter
   *                     is interested in.
   * @param milliseconds The waiting timeout in milliseconds. Zero means infinite timeout.
   * @throws IPCException When something went wrong in communications.
   */
  void waitTransactionFinished(Long txid, long milliseconds) throws IPCException;
}
```

Kuvassa 24 on kuvattu JMS-modulin luokkakaavio.



Kuva 24: JMS-modulin luokkakaavio

Seuraavassa ConfigurationChangeListener-rajapinnan kuvaus, jotka ne modulit, joka ovat kiinnostuneita konfiguraation muutoksista voivat toteuttaa.

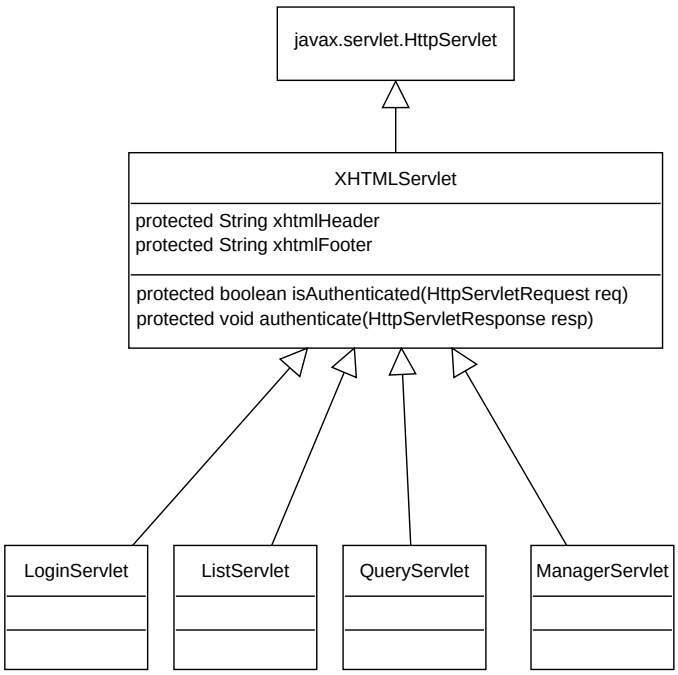
```
public interface ConfigurationChangeListener
{
  /**
   * This callback method is called by the IPC
   * implementation thread after a configuration
   * changed signal was received.
   */
  void configurationChanged();
}
```

4.11 JMX-moduli

Tämä moduli tarjoaa selainpohjaisen käyttöliittymän MSS-toteutuksen konfigurointiin ja monitorointiin. Se perustuu Javan Servlet-rajapintaan [Sunc].

LoginServlet hoitaa sisäänkirjautumisen. ListServlet listaa kaikki saatavilla olevat MBeanit. QueryServlet listaa tietyn MBeanin attribuutit ja operaatiot, jotka se on julkaissut näkyville. ManagerServlet muuttaa MBeanin attribuuttia tai käynnistää jonkun MBeanin julkaiseman operaation.

Luokkakaavio on esitetty kuvassa 25.



Kuva 25: JMX -modulin luokkakaavio

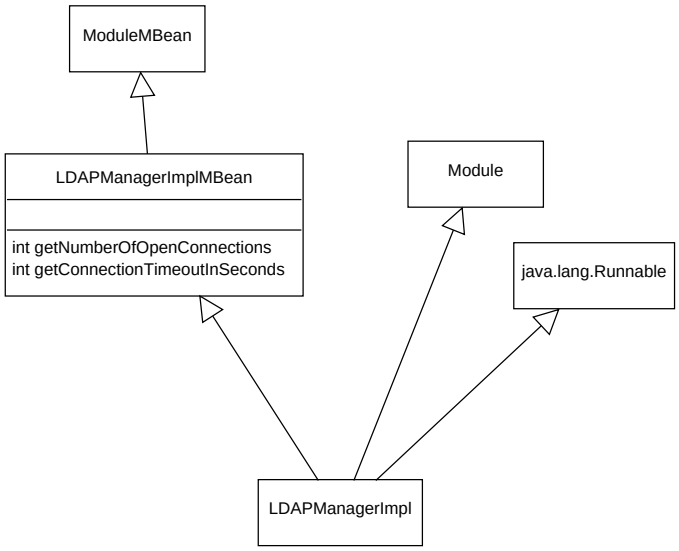
4.12 LDAP-moduli

LDAP-moduli tarjoaa rajapinnan jolla hakea varmenteita sekä sulkulistoja (CRL). Seuraavassa modulin JavaDoc-rajapinta:

```
public interface LDAPManager
{
  /**
   * Fetches a CRL from LDAP.
   *
```

```
   * @param ldapUrl
   *  URL to the CRL.
   * @return
   *  The CRL object.
   * @throws CRLException
   *  When the CRL object couldn't be constructed
   * @throws LDAPException
   *  When the CRL material couldn't be fetched from LDAP.
   */
  X509CRL fetchCRL(LDAPUrl ldapUrl, String ldapUsername, String ldapPassword) throws
    CRLException,
    LDAPException,
    MalformedURLException,
    CertificateException;

  /**
   * Fetches a CA certificate from LDAP.
   *
   * @param ldapUrl
   *  URL to the CA certificate.
   * @return
   *  The CA certificate object.
   * @throws CertificateException
   *  When the CA certificate object couldn't be constructed
   * @throws LDAPException
   *  When the CA certificate material couldn't be fetched from LDAP.
   */
  X509Certificate fetchCACertifiacte(LDAPUrl ldapUrl, String ldapUsername, String ldapPassword) throws
    CertificateException,
    LDAPException,
    MalformedURLException;

  /**
   * Fetches a signer certificate from LDAP.
   *
   * @param signerCertificateReference
   *  A reference to the signer's certificate, LDAP URL
   *  for example.
   * @return
   *  The signer certificate object.
   * @throws CertificateException
   *  When the signer certificate object couldn't be constructed
   * @throws LDAPException
   *  When the signer certificate material couldn't be fetched from LDAP.
   */
  X509Certificate[] fetchSignerCertificate(String signerCertificateReference, String ldapUsername, String ldapPassword) th
    CertificateException,
    LDAPException,
    MalformedURLException;
}
```

Luokkakaavio on esitetty kuvassa 26.



Kuva 26: LDAP -modulin luokkakaavio

4.13 SDK-moduli

SDK-moduli piilottaa Axis:n käsittelyn rajapintojen taakse. Sen käyttäjän ei siis tarvitse tuntea SOAP:ia tai Axis:ta. Seuraavassa synkroninen asiakas-palvelin -viestintämoodin JavaDoc-rajapinta sekä asynkronisen asiakas-palvelin -viestintämoodin JavaDoc-rajapinta:

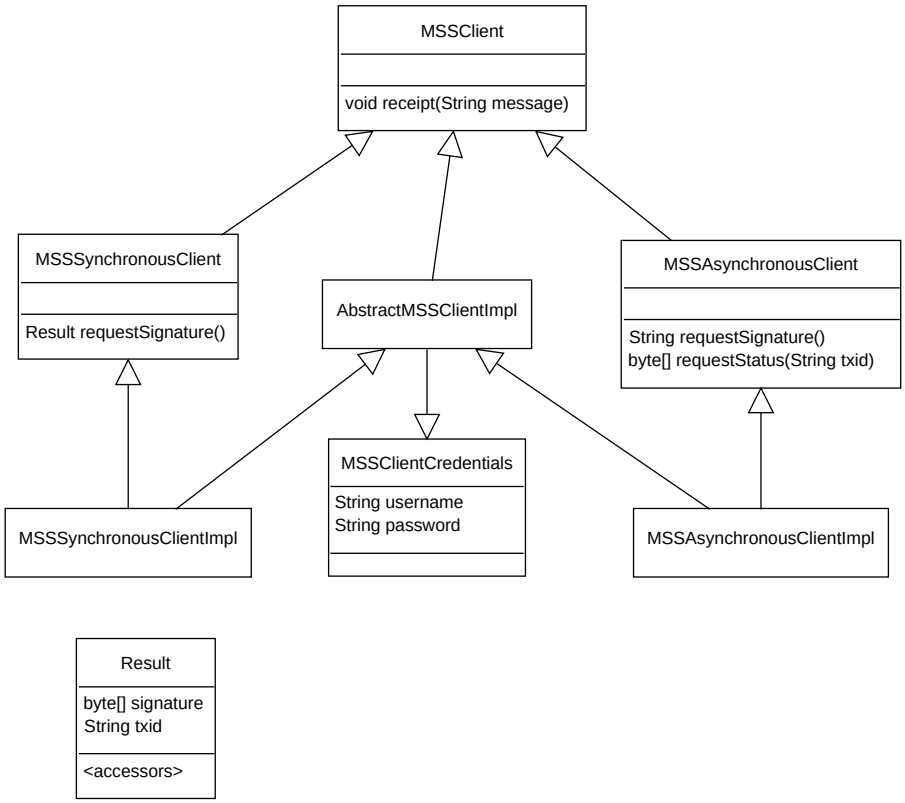
```
public interface MSSClient
{
  /**
   * Sends a receipt tp a mobile user.
   *
   * @param msisdn
   *  MSISDN of the mobile.
   * @param receiptMessage
   *  Message to be sent.
   * @param txid
   *  Transaction identifier from requestSignature().
   * @throws URI.MalformedURIException
   * @throws RemoteException
   * @throws ServiceException
   */
  void receipt(String msisdn, String receiptMessage, String txid) throws URI.MalformedURIException, RemoteException, Serv
}

public interface MSSSynchronousClient extends MSSClient
{
  /**
   * Starts a MSS transaction (MSS_Signature)
   * and returns a PKCS#7 signature or throws
```

```
   * an exception in case of validation failed or
   * transaction timed out (when user didn't sign in time).
   *
   * @param msisdn
   *  MSISDN of the signer.
   * @param textToBeSigned
   *  What is signed.
   * @param dataToBeDisplayed
   *  What is displayed on mobile before signing (optional, can be null).
   * @param signatureProfile
   *  What signing device should be used. This goes to SignatureProfile.
   * @param sessionID
   *  Session ID (see FiCom spec) or null if it is not wanted.
   * @param civilIDCheckRegExp
   *  If not null, is used to check whether subject DN of the signer matches this regexp. If it doesn't match,
   *  signature validation will fail.
   * @param validate
   *  If this is true, transaction is validated.
   *  If this is false, validation is done or not done dependening on the MSSP server configuration.
   * @return
   *  PKCS#7 data.
   * @throws SignatureException
   *  Signature verification failed.
   * @throws RevokedException
   *  Signer's certificate was revoked.
   * @throws CertificateException
   *  Other reason caused certificate validation to fail (not trusted, not signed by issuer, not valid at the time, ...)
   * @throws TimeoutException
   *  Signer didn't sign in time.
   * @throws IOException
   *  Something nasty happened during communication with the server (SSL handshake failed for example).
   * @throws GeneralSecurityException
   *  PKCS#7 parsing failure.
   */
  Result requestSignature(String msisdn,
                          String textToBeSigned,
                          String dataToBeDisplayed,
                          String signatureProfile,
                          String sessionID,
                          String civilIDCheckRegExp,
                          boolean validate)
    throws
    SignatureException,
    RevokedException,
    CertificateException,
    TimeoutException,
    ServiceException,
    RemoteException,
    IOException,
    GeneralSecurityException,
    AuthenticationException;
}

public interface MSSAsynchronousClient extends MSSClient
{
```

```
/**
 * Starts a MSS transaction (MSS_Signature).
 *
 * @param msisdn
 *  MSISDN of the signer.
 * @param textToBeSigned
 *  What is signed.
 * @param dataToBeDisplayed
 *  What is displayed on mobile before signing (optional, can be nulL).
 * @param signatureProfile
 *  What signing device should be used. This goes to SignatureProfile.
 * @param sessionID
 *  Session ID (see FiCom spec) or null if it is wanted.
 * @param civilIDCheckRegExp
 *  If not null, is used to check whether subject DN of the signer matches this regexp. If it doesn't match,
 *  signature validation will fail.
 * @param validate
 *  If this is true, transaction is validated.
 *  If this is false, validation is done or not done dependening on the MSSP server configuration.
 * @return
 *  Transaction identifier which is used to ask the status of the transaction.
 * @throws IOException
 *  Something nasty happened during communication with the server (SSL handshake failed for example).
 */
String requestSignature(String msisdn,
                        String textToBeSigned,
                        String dataToBeDisplayed,
                        String signatureProfile,
                        String sessionID,
                        String civilIDCheckRegExp,
                        boolean validate)
  throws
  ServiceException,
  RemoteException,
  IOException;

/**
 * Queries the result of a transaction (MSS_StatusQuery).
 *
 * @param txid
 *  The transaction identifier given from starting the transaction.
 * @return
 *  PKCS#7 data if the transaction was finished. If transaction wasn't finished. null is returned.
 * @throws SignatureException
 *  Signature verification failed.
 * @throws RevokedException
 *  Signer's certificate was revoked.
 * @throws CertificateException
 *  Other reason caused certificate validation to fail (not trusted, not signed by issuer, not valid at the time, ...)
 * @throws TimeoutException
 *  Signer didn't sign in time.
 * @throws IOException
 *  Something nasty happened during communication with the server (SSL handshake failed for example).
 * @throws GeneralSecurityException
 *  PKCS#7 parsing failure.
```

```
   */
  byte[] requestStatus(String txid)
    throws
    SignatureException,
    RevokedException,
    CertificateException,
    TimeoutException,
    ServiceException,
    RemoteException,
    IOException,
    GeneralSecurityException, AuthenticationException;
}
```

SDK-modulin luokkakaavio on esitetty seuraavassa kuvassa 27:



Kuva 27: SDK -modulin luokkakaavio

4.14 Signing Gateway -moduli

Tämä moduli lähettää loppukäyttäjän puhelimeen allekirjoituspyyntöjä ja vastaanottaa allekirjoituksia.

Seuraavassa on Signing Gateway -modulin rajapintakuvaus:

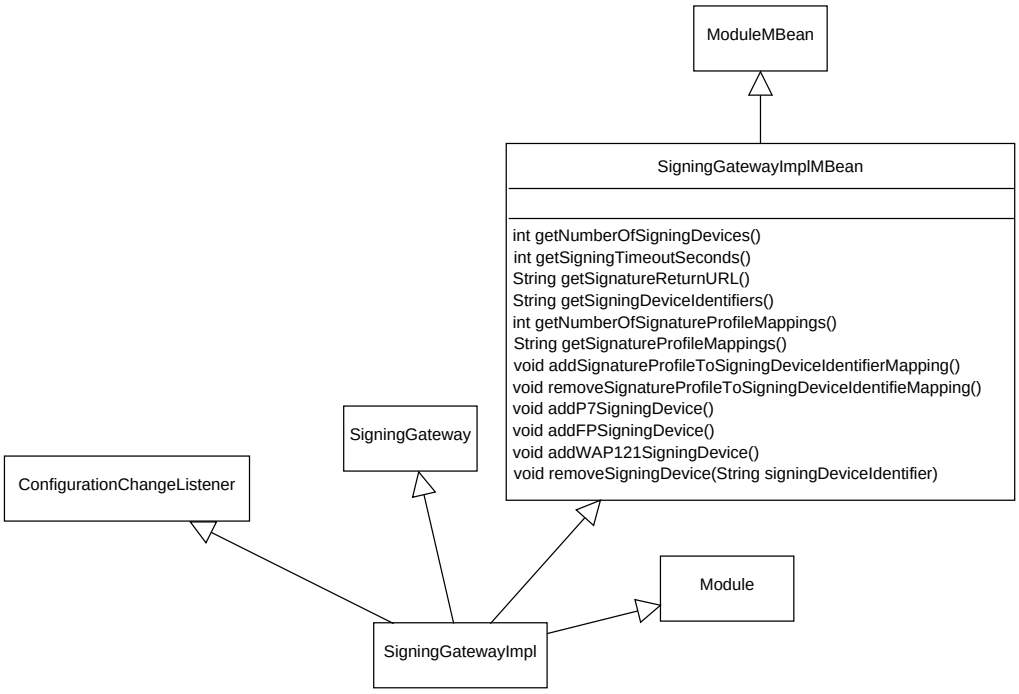
```
public interface SigningGateway
{
  /**
   * Returns signatureProfile -> signing device identifier mapping.
   *
   * @param signatureProfile
   *  The signature profile.
   * @return
   *  Sigining device identifier.
   */
  String getSignatureProfileMapping(String signatureProfile);

  /**
   * Returns signing device based on signing device identifier.
   *
   * @param signingDeviceIdentifier
   *  The identifier.
   * @return
   *  Signing device object.
   */
  SigningDevice getSigningDevice(String signingDeviceIdentifier);

  /**
   * Returns an URL where signatures are expected to come back.
   *
   * @return
   *  The URL.
   */
  String getSignatureReturnURL();

  int getSigningTimeoutSeconds();
}
```

Kuvassa 28 on Signing Gatewayn luokkakaavio:

Varsinainen toiminta tapahtuu SigningDevice-luokassa ja sen aliluokissa. Signing Gateway on lähinnä varasto niille. Niiden luokkakaavio on esitetty kuvassa 29.

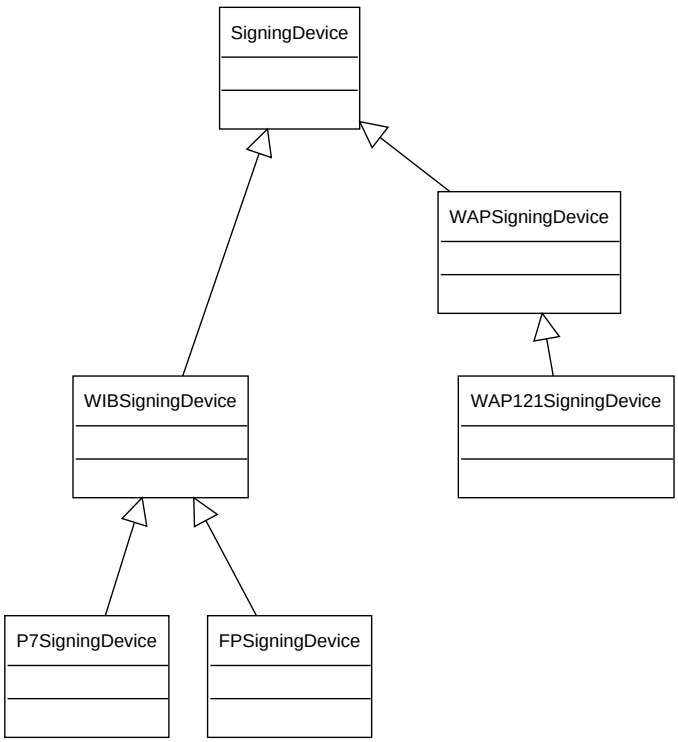
Seuraavassa on abstraktin Signing Device -luokan rajapinta:

```
  /**
   * Sends a request for digital signature.
   *
   * @param signingRequestParameters
   *  Required parameters for the request.
   * @throws IOException
   *  Error happened during send of the receipt.
   * @throws TooLongTextToBeSignedException
   *  Text that was to be signed was too long for this device type.
   * @throws OTAErrorException
   *  Over The Air error happened.
   */
  public abstract void sendSigningRequest(SigningRequestParameters signingRequestParameters)
    throws IOException, TooLongTextToBeSignedException, OTAErrorException;
```



Kuva 28: Signing Gateway -modulin luokkakaavio

```
/**
 * Sends a receipt to a user.
 *
 * @param receiptParameters
 *  Parameters of the receipt.
 * @throws IOException
 *  Error happened during send of the receipt.
 * @throws OTAErrorException
 *  Over The Air error happened.
 */
public abstract void sendReceipt(ReceiptParameters receiptParameters)
  throws IOException, OTAErrorException;

/**
 * Checks whether this signing device supports this MIME type of data for signing.
 *
 * @param mimeType
 *  The MIME type.
 * @throws MimeTypeNotSupportedException
 */
public abstract void checkMimeType(String mimeType)
  throws MimeTypeNotSupportedException;

/**
 * Should handle incoming digital signature.
 *
 * @param parameters
```



Kuva 29: Allekirjoituslaitteiden luokkakaavio

```
   *  Parameters of the HTTP request containing signature among other things.
   * @param txid
   *  Transaction identifier.
   * @return
   *  Response to the phone (gateway).
   */
  public abstract HttpResponseContent handleSignature(Map parameters, Long txid);
```

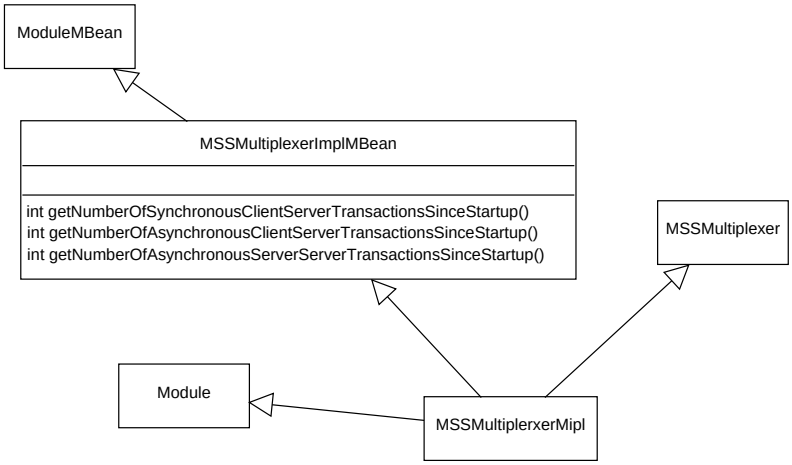
4.15 SOAP-moduli

SOAP-moduli vastaanottaaa MSS-pyynnöt Axis:n generoimissa luokissa (skeletons, luurangot). Näistä luokista nämä pyynnöt "multipleksataan"yhteen luokkaan (MSS-MultiplexerImpl). Seuraavassa MSSMultiplexer:n rajapintakuvaus:

```
public interface MSSMultiplexer
{
  MSS_HandshakeRespType MSS_Handshake(MSS_HandshakeReqType MSS_HandshakeReq) throws RemoteException;

  MSS_StatusRespType MSS_Notification(MSS_SignatureRespType MSS_SignatureResp) throws RemoteException;

  MSS_ProfileRespType MSS_ProfileQuery(MSS_ProfileReqType MSS_ProfileReq) throws RemoteException;
```

```
  MSS_ReceiptRespType MSS_Receipt(MSS_ReceiptReqType MSS_ReceiptReq) throws RemoteException;

  MSS_RegistrationRespType MSS_Registration(MSS_RegistrationReqType MSS_RegistrationReq) throws RemoteException;

  MSS_SignatureRespType MSS_Signature(MSS_SignatureReqType MSS_SignatureReq, Object extraParam) throws RemoteException;

  MSS_StatusRespType MSS_StatusQuery(MSS_StatusReqType MSS_StatusReq) throws RemoteException;

  void callbackToMSSClient(TransactionData transactionData) throws IOException, ServiceException;
}
```

Luokkakaavio on kuvassa 30.



Kuva 30: SOAP -modulin luokkakaavio

4.16 Transaction Data Manager -moduli

Tämä moduli käy ainostastaan läpi tapahtumataulua aika ajoin etsien tapahtumia, jotka vaativat aikakatkaisua (timeout). Jos tällaisia tapahtumia löytyy, niistä ilmoitetaan Bridge-modulille jotka hoitaa asian eteenpäin merkkaamalla tapahtuman tilan tietokantaan, ja signaloimalla JMS-modulin avulla mahdollisesti tapahtumaa odottavaa säiettä.

5 Suorituskyvystä

Jo aikaisemmin tässä tekstissä viitattiin korkeaan käytettävyyteen (High Availability, HA). Korkea käytettävyys nostaa myös suorituskykyä, jos MSS-prosesseja on

useita eri palvelimissa. Täten MSS-toteutus skaalautuu lähes lineaarisesti. *Lähes* lineaarisesti siksi, että JMS-viestit ovat yleislähetyksiä (broadcast), ja niistä voi tulla pullonkaula, jos MSS-prosesseja on paljon.

Seuraavassa on esitetty skaalautumista HP:n NonStop-palvelimella [HP]. NonStop-palvelimessa prosessorit ovat löyhästi kytkettyjä (loosely coupled), kyse ei ole symmetrisestä moniprosessorikoneesta. Seuraavassa taulukossa on montako tapahtumaa sekunnissa järjestelmä pystyy tekemään, kun tapahtumia on tehty 10 kappaletta ja AP:n päässä on 10 säiettä, yhteensä siis 100 tapahtumaa.

Yksi tapahtuma koostuu seuraavista askeleista, käytössä on asynkroninen asiakas-palvelin -moodi:

1. Aloitetaan tapahtuma kutsumalla MSS_Signature:a
2. Tehdään MSS_StatusQuery, tapahtuma ei ole vielä valmis
3. Tehdään MSS_StatusQuery, tapahtuma ei ole vielä valmis
4. Simuloidaan matkapuhelinta ja lähetetään allekirjoitus HTTP:lla MSS-palvelimelle
5. Tehdään MSS_StatusQuery, tapahtuma on nyt valmis

Seuraavassa taulukko, josta ilmenee suorituskyky tapahtumina per sekunti.

Prosessoreita	Tapahtumia sekunnissa
1	1,54
2	3,06

MSS-toteutuksen profilointi osoittaa, että pullonkaulalaite on CPU. NonStop-palvelin käyttää verrattaen vanhaa CPU-teknologiaa, joten yllämainitut suorituskykylukemat eivät ole kovin korkeita. Esimerkiksi Sunin moniprosessorilaitteistolla puhutaan kymmenistä MSS-tapahtumista sekunnissa.

6 Yhteenveto

Julkisen avaimen menetelmä mahdollistaa mm. henkilön vahvan tunnistamisen ja laillisesti sitovan digitaalisen allekirjoituksen luomisen. Julkisen avaimen menetelmissä on kuitenkin ongelmana esimerkiksi tarvittava avaimen pituus jotta saavutetaan sama turvallisuuden taso verrattuna symmetrisiin salausmenetelmiin. Toinen

ongelma on suorituskyky; julkisen avaimen menetelmät ovat kertaluokkaa hitaampia kuin symmetriset salausmenetelmät. Julkisen avaimen menetelmän algoritmin valinta riippuu sovellusalueesta; ei ole olemassa yhtä algoritmia joka olisi paras jokaisella alueella.

ETSI:n MSS-määritys on yksi julkisen avaimen menetelmää hyödyntävä spesifikaatio, joka määrittelee SOAP:iin perustuvan rajapinnan siihen, miten loppukäyttäjältä pyydetään digitaalinen allekirjoitus.

Valimo Wireless Oy:n toteutus MSS-määrityksestä (Valimo Validator - MSSP) tarjoaa MSS-asiakkaalle, eli Application Providerille korkean käytettävyyden sekä korkean suorituskyvyn.

Lähteet

Apa Apache <Web Services/> Project. Web Services - Axis. `http://ws.apache.org/axis/`.

BBF+02a Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML-Signature Syntax and Processing. `http://www.ietf.org/rfc/rfc3275.txt`, March 2002. RFC 3275.

BBF+02b Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML-Signature Syntax and Processing. `http://www.w3.org/TR/xmldsig-core/`, February 2002. W3C Recommendation.

BD98 Dan Boneh and Glenn Durfee. Breaking RSA may not be equivalent to factoring. In *Proceedings Eurocrypt '98, Lecture Notes in Computer Science*, volume 1233, pages 55–79. Springer-Verlag, 1998.

BD99 Dan Boneh and Glenn Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. In *Proceeding Eurocrypt '99, Lecture Notes in Computer Science*, volume 1592, pages 1–11. Springer-Verlag, 1999.

BHL99 Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. `http://www.w3.org/TR/REC-xml-names/`, January 1999. W3C.

BLMF+98 T. Berners-Lee, MIT/LCS, R. Fielding, U.C. Irvine, L. Masinter, and Xerox Corporation. Uniform Resource Identifiers (URI): Generic Syn-

tax. http://www.ietf.org/rfc/rfc2396.txt, August 1998. RFC 2396.

CCMW01 Erik Christensen, Francisco Curbera, Greg Meredith, and Sanijva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001. W3C.

DA99 T. Dierks and C. Allen. The TLS Protocol Version 1.0. http://www.ietf.org/rfc/rfc2246.txt, January 1999. RFC 2246.

DH76 Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

Eng02 Robert Englander. *Java and SOAP*. O'Reilly, 2002.

ETS03 ETSI. Mobile Commerce (M-COMM); Mobile Signature Service; Web Service Interface, August 2003. ETSI TS 102 204 v1.1.4.

FIBV96a N. Freed, Innosoft, N. Borenstein, and First Virtual. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. http://www.ietf.org/rfc/rfc2045.txt, November 1996. RFC 2045.

FIBV96b N. Freed, Innosoft, N. Borenstein, and First Virtual. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. http://www.ietf.org/rfc/rfc2046.txt, November 1996. RFC 2046.

FiC05 FiCom Ry. FICOM RY:N SOVELTAMISOHJE ETSI:N MSS-STANDARDEILLE V1.1. http://www.ficom.fi/linked/fi/MSS_FiCom_Soveltamisohje_v1.1.pdf, 2005.

GHJV95 Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1995.

GHM+03 Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP version 1.2 part 1: Messaging framework. http://www.w3.org/TR/soap12-part1/, June 2003.

Gül02 Ceki Gülcü. Short introduction to log4j. http://logging.apache.org/log4j/docs/manual.html, March 2002.

Hib Hibernate Team. Relational Persistence for Java and .NET. http://www.hibernate.org.

HP Hewlett-Packard. HP NonStop S-series servers - specifications. http://h20223.www2.hp.com/NonStopComputing/cache/255746-0-0-0-121.html.

HS99 R. Housley and SPYRUS. Cryptographic Message Syntax. http://www.ietf.org/rfc/rfc2630.txt, June 1999. RFC 2630.

IDS02 Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing. http://www.w3.org/TR/xmlenc-core/, December 2002. W3C Recommendation.

KR95 Burt Kaliski and Matt Robshaw. The secure use of RSA. *CryptoBytes*, 1(3):7–13, 1995. RSA Laboratories.

Lea04 Doug Lea. Overview of package util.concurrent Release 1.3.4. http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html, May 2004.

LV01 Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

M. 97 M. Wahl, Critical Angle Inc., S. Kille, Isode Ltd., T. Howes, Netscape Communications Corp. Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names. http://www.ietf.org/rfc/rfc2253.txt, December 1997. RFC 2253.

MW96 Ueli M. Maurer and Stefan Wolf. On the complexity of breaking the Diffie-Hellman protocol. Technical Report 244, Institute for Theoretical Computer Science ETH Zurich, 1996.

NIS95 NIST. Announcing the Secure Hash Standard, 1995. FIPS 180-1.

Nov Novell. LDAP Classes for Java. http://developer.novell.com/ndk/jldap.htm.

Ran05 James Randall. Hash Function update due to potential weakness found in SHA-1. http://www.rsasecurity.com/rsalabs/node.asp?id=2834, March 2005. RSA Laboratories.

Riv92 Ron Rivest. The MD5 Message-Digest Algoritm. http://www.ietf.org/rfc/rfc1321.txt, April 1992. RFC 1321.

RSA93 RSA. PKCS #7: Cryptographic Message Syntax Standard. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/, November 1993.

RSA00 RSA. PKCS #10 v1.7: Certification Request Syntax Standard. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/, May 2000.

Sch96 Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, Inc, 1996.

Smaa SmartTrust. SmartTrust WIB Plug-in Specification for Application Developers. http://www.smarttrust.com/support_services/pdf/Developerscorner_WIBplugin.pdf.

Smab SmartTrust. WIG WML Specification Version 4. http://www.smarttrust.com/support_services/pdf/Developerscorner_wmlspecificationwig.pdf.

Sta98 William Stallings. *Cryptography and Network Security: Principles and Practice.* Prentice-Hall, 1998.

Suna Sun. Java Management Extensions (JMX) Overview. http://java.sun.com/products/jmx/overview.html.

Sunb Sun. Java Message Service (JMS). http://java.sun.com/products/jms/index.jsp.

Sunc Sun. Java Servlet Technology Overview. http://java.sun.com/products/servlet/overview.html.

Sun02 Sun. The Java HotSpot Virtual Machine, v1.4.1. http://java.sun.com/products/hotspot/docs/whitepaper/Java_Hotspot_v1.4.1/JHS_141_WP_d2a.pdf, September 2002.

Szy05 Michael Szydlo. SHA1 collisions can be found in 2^{63} operations. http://www.rsasecurity.com/rsalabs/node.asp?id=2927, August 2005. RSA Laboratories.

TBMM04 Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures Second Edition. http://www.w3.org/TR/xmlschema-1/, October 2004. W3C Recommendation.

Wie98 Michael J. Wiener. Performance comparison of public-key cryptosystems. *CryptoBytes*, 4(1):1–5, 1998. RSA Laboratories.

XDo XDoclet Team. XDoclet - Attribute Oriented Programming. `http://xdoclet.sourceforge.net`.

YL05 Tatu Ylönen and Chris Lonvick. SSH protocol architecture. `http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-22.txt`, March 2005.

Liite 1. XML-skeema -määritykset

A MSS-viestien XML-skeema -määrittelyt

Seuraavassa on määritelty eri MSS-viestien rakenteet täsmällisesti XML-skeema -määrityksin.

A.1 XML-skeema -nimiavaruuksista

Seuraavien kappaleiden XML-skeema -määrityksissä oletetaan käytettävän seuraavaa XML-nimiavaruuksien määritystä:

```
<xs:schema
  targetNamespace="http://uri.etsi.org/TS102204/v1.1.2#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  elementFormDefault="qualified"
>
```

Tämä tarkoittaa sitä, että etuliite 'ds' viittaa W3C:n XML-allekirjoitusmäärityksen [BBF+02b] nimiavaruuteen, 'xs' viittaa XML-skeema -määrityksen [TBMM04] nimiavaruuteen, 'xenc' viittaa XML-salausmäärityksen [IDS02] ja 'env' viittaa SOAP-määrityksen [GHM+03] nimiavaruuteen. Etuliite 'mss' viittaa MSS-palvelun [ETS03] nimiavaruuteen.

A.2 MessageAbstractType

MessageAbstractType-viestistä periytyvät kaikki MSS-sanomat:

```
<xs:complexType name="MessageAbstractType" abstract="true">
  <xs:sequence>
    <xs:element name="AP_Info">
      <xs:complexType>
        <xs:attribute name="AP_ID"
                      type="xs:anyURI"
```

```
                    use="required"/>
        <xs:attribute name="AP_TransID"
                    type="xs:NCName"
                    use="required"/>
        <xs:attribute name="AP_PWD"
                    type="xs:string"
                    use="required"/>
        <xs:attribute name="Instant"
                    type="xs:dateTime"
                    use="required"/>
        <xs:attribute name="AP_URL"
                    type="xs:anyURI"
                    use="optional"/>
      </xs:complexType>
    </xs:element>
    <xs:element>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="MSSP_ID"
                    type="mss:MeshMemberType"/>
        </xs:sequence>
        <xs:attribute name="Instant"
                    type="xs:dateTime"
                    use="optional"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="MajorVersion"
                type="xs:integer"
                use="required"/>
  <xs:attribute name="MinorVersion"
                type="xs:integer"
                use="required"/>
</xs:complexType>
```

A.3 MSS_Signature-palvelu

Seuraavassa on kuvattu MSS_Signature-palvelun WSDL-kuvaus: [CCMW01]

```
<message name="MSS_SignatureInput">
  <part name="MSS_SignatureRequest"
        type="mss:MSS_SignatureRequestType"/>
</message>
<message name="MSS_SignatureOutput">
  <part name="MSS_SignatureResponse"
        type="mss:MSS_SignatureResponseType"/>
</message>

<portType name="MSS_SignaturePortType">
  <operation name="MSS_Signature">
    <input message="tns:MSS_SignatureInput"/>
    <output message="tns:MSS_SignatureOutput"/>
  </operation>
```

```
</portType>
```

MSS_SignatureRequest-viestin XML-skeema määritys:

```
<xs:element name="MSS_SignatureRequest"
            type="mss:MSS_SignatureRequestType"/>
<xs:complexType name="MSS_SignatureRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType"/>
        <xs:element name="DataToBeSigned"
                    type="mss:DataType"/>
        <xs:element name="DataToBeDisplayed"
                    type="mss:DataType"
                    minOccurs="0"/>
        <xs:element name="SignatureProfile"
                    type="mss:mssURIType"
                    minOccurs="0"/>
        <xs:element name="AdditionalServices"
                    minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Service"
                          type="mss:AdditionalServiceType"
                          maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="MSS_Format"
                    type="mss:mssURIType"
                    minOccurs="0"/>
        <xs:element name="KeyReference"
                    type="mss:KeyReferenceType"
                    minOccurs="0"/>
        <xs:element name="SignatureProfileComparison"
                    type="mss:SignatureProfileComparisonType"
                    minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="ValidityDate"
                    type="xs:dateTime"
                    use="optional"/>
      <xs:attribute name="TimeOut"
                    type="xs:positiveInteger"
                    use="optional/>
      <xs:attribute name="MessagingMode"
                    type="mss:MessagingModeType"
                    use="required/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_SignatureResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_SignatureResponse"
            type="mss:MSS_SignatureResponseType"/>
<xs:complexType name="MSS_SignatureResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType"/>
        <xs:element name="Status"
                    type="mss:StatusType"/>
        <xs:element name="SignatureProfile"
                    type="mss:mssURIType"
                    minOccurs="0"/>
        <xs:element name="MSS_Signature"
                    type="mss:SignatureType"
                    minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="MSSP_TransID"
                    type="xs:NCName"
                    use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A.4 MSS_StatusQuery-palvelu

Seuraavassa on kuvattu MSS_StatusQuery-palvelun WSDL-kuvaus:

```
<message name="MSS_StatusQueryInput">
  <part name="MSS_StatusQueryRequest"
        type="mss:MSS_StatusQueryRequestType"/>
</message>
<message name="MSS_StatusQueryOutput">
  <part name="MSS_StatusQueryResponse"
        type="mss:MSS_StatusQueryResponseType"/>
</message>

<portType name="MSS_StatusQueryPortType">
  <operation name="MSS_StatusQuery">
    <input message="tns:MSS_StatusQueryInput"/>
    <output message="tns:MSS_StatusQueryOutput"/>
  </operation>
</portType>
```

MSS_StatusQueryRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_StatusQueryRequest"
            type="mss:MSS_StatusQueryRequestType"/>
<xs:complexType name="MSS_StatusQueryRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
```

```
      <xs:attribute name="MSSP_TransID"
                    type="xs:NCName"
                    use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_StatusQueryResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_StatusQueryResponse"
            type="mss:MSS_StatusQueryResponseType"/>
<xs:complexType name="MSS_StatusQueryResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType/>
        <xs:element name="MSS_Signature"
                    type="mss:SignatureType"
                    minOccurs="0"/>
        <xs:element name="Status"
                    type="mss:StatusType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A.5 MSS_ProfileQuery-palvelu

Seuraavassa on kuvattu MSS_ProfileQuery-palvelun WSDL-kuvaus:

```
<message name="MSS_ProfileQueryInput">
  <part name="MSS_ProfileQueryRequest"
        type="mss:MSS_ProfileQueryRequestType"/>
</message>
<message name="MSS_ProfileQueryOutput">
  <part name="MSS_ProfileQueryResponse"
        type="mss:MSS_ProfileQueryResponseType"/>
</message>

<portType name="MSS_ProfileQueryPortType">
  <operation name="MSS_ProfileQuery">
    <input message="tns:MSS_ProfileQueryInput"/>
    <output message="tns:MSS_ProfileQueryOutput"/>
  </operation>
</portType>
```

MSS_ProfileQueryRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_ProfileQueryRequest"
            type="mss:MSS_ProfileQueryRequestType"/>
<xs:complexType name="MSS_ProfileQueryRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="MobileUserType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_ProfileQueryResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_ProfileQueryResponse"
            type="mss:MSS_ProfileQueryResponseType"/>
<xs:complexType name="MSS_ProfileQueryResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileSignatureProfile"
                    type="mss:mssURIType"
                    minOccurs="0"/>
        <xs:element name="Status"
                    type="mss:StatusType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A.6 MSS_Registration-palvelu

Seuraavassa on kuvattu MSS_Registration-palvelun WSDL-kuvaus:

```
<message name="MSS_RegistrationInput">
  <part name="MSS_RegistrationRequest"
        type="mss:MSS_RegistrationRequestType"/>
</message>
<message name="MSS_RegistrationOutput">
  <part name="MSS_RegistrationResponse"
        type="mss:MSS_RegistrationResponseType"/>
</message>

<portType name="MSS_RegistrationPortType">
  <operation name="MSS_Registration">
    <input message="tns:MSS_RegistrationInput"/>
    <output message="tns:MSS_RegistrationOutput"/>
  </operation>
</portType>
```

MSS_RegistrationRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_RegistrationRequest"
            type="mss:MSS_RegistrationRequestType"/>
<xs:complexType name="MSS_RegistrationRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType"/>
        <xs:element name="EncryptedData"
                    type="xenc:EncryptedType"
                    minOccurs="0"/>
        <xs:element name="EncryptResponseBy"
                    type="xs:anyURI"
                    minOccurs="0"/>
        <xs:element name="CertificateURI"
                    type="xs:anyURI"
                    minOccurs="0"/>
        <xs:element name="X509Certificate"
                    type="xs:base64Binary"
                    minOccurs="0"/>
        <xs:any namespace="##other"
                processContents="lax"
                minOccurs="0"
                maxOccurs="unbounded/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_RegistrationResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_RegistrationResponse"
            type="mss:MSS_RegistrationResponseType"/>
<xs:complexType name="MSS_RegistrationResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="Status"
                    type="mss:StatusType"/>
        <xs:element name="EncryptedData"
                    type="xenc:EncryptedType"
                    minOccurs="0"/>
        <xs:element name="CertificateURI"
                    type="xs:anyURI"
                    minOccurs="0"/>
        <xs:element name="X509Certificate"
                    type="xs:base64Binary"
                    minOccurs="0"/>
        <xs:element name="PublicKey"
                    type="xs:base64Binary"
                    minOccurs="0"/>
```

```
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A.7 MSS_Receipt-palvelu

Seuraavassa on kuvattu MSS_Receipt-palvelun WSDL-kuvaus:

```
<message name="MSS_ReceiptInput">
  <part name="MSS_ReceiptRequest"
        type="mss:MSS_ReceiptRequestType"/>
</message>
<message name="MSS_ReceiptOutput">
  <part name="MSS_ReceiptResponse"
        type="mss:MSS_ReceiptResponseType"/>
</message>

<portType name="MSS_ReceiptPortType">
  <operation name="MSS_Receipt">
    <input message="tns:MSS_ReceiptInput"/>
    <output message="tns:MSS_ReceiptOutput"/>
  </operation>
</portType>
```

MSS_ReceiptRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_ReceiptRequest"
            type="mss:MSS_ReceiptRequestType"/>
<xs:complexType name="MSS_ReceiptRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType"/>
        <xs:element name="Status"
                    type="mss:StatusType"
                    minOccurs="0"/>
        <xs:element name="Message"
                    type="mss:DataType"
                    minOccurs="0"/>
        <xs:element name="SignedReceipt"
                    type="ds:SignatureType"
                    minOccurs="0""/>
      </xs:sequence>
      <xs:attribute name="MSSP_TransID"
                    type="xs:NCName"
                    use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_ReceiptResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_ReceiptResponse"
            type="mss:MSS_ReceiptResponseType"/>
<xs:complexType name="MSS_ReceiptResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="Status"
                    type="mss:StatusType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A.8 MSS_Handshake-palvelu

Seuraavassa on kuvattu MSS_Handshake-palvelun WSDL-kuvaus:

```
<message name="MSS_HandshakeInput">
  <part name="MSS_HandshakeRequest"
        type="mss:MSS_HandshakeRequestType"/>
</message>
<message name="MSS_HandshakeOutput">
  <part name="MSS_HandshakeResponse"
        type="mss:MSS_HandshakeResponseType"/>
</message>

<portType name="MSS_HandshakePortType">
  <operation name="MSS_Handshake">
    <input message="tns:MSS_HandshakeInput"/>
    <output message="tns:MSS_HandshakeOutput"/>
  </operation>
</portType>
```

MSS_HandshakeRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_HandshakeRequest"
            type="mss:MSS_HandshakeRequestType"/>
<xs:complexType name="MSS_HandshakeRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="SecureMethods">
          <xs:complexType>
            <xs:attribute name="MSS_Signature"
                          type="xs:boolean"
                          use="required"/>
            <xs:attribute name="MSS_Registration"
                          type="xs:boolean"
```

```
                        use="required"/>
            <xs:attribute name="MSS_Notification"
                        type="xs:boolean"
                        use="required"/>
            <xs:attribute name="MSS_ProfileQuery"
                        type="xs:boolean"
                        use="required"/>
            <xs:attribute name="MSS_Receipt"
                        type="xs:boolean"
                        use="required"/>
            <xs:attribute name="MSS_Status"
                        type="xs:boolean"
                        use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="Certificates">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Certificate"
                        type="xs:base64Binary"
                        minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="RootCAs">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DN"
                        type="xs:string"
                        minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="SignatureAlgList">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="SignatureAlg"
                        type="mss:mssURIType"
                        minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_HandshakeResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_HandshakeResponse"
            type="mss:MSS_HandshakeResponseType"/>
<xs:complexType name="MSS_HandshakeResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
```

```
  <xs:sequence>
    <xs:element name="SecureMethods">
      <xs:complexType>
        <xs:attribute name="MSS_Signature"
                      type="xs:boolean"
                      use="required"/>
        <xs:attribute name="MSS_Registration"
                      type="xs:boolean"
                      use="required"/>
        <xs:attribute name="MSS_Notification"
                      type="xs:boolean"
                      use="required"/>
        <xs:attribute name="MSS_ProfileQuery"
                      type="xs:boolean"
                      use="required"/>
        <xs:attribute name="MSS_Receipt"
                      type="xs:boolean"
                      use="required"/>
        <xs:attribute name="MSS_Status"
                      type="xs:boolean"
                      use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="MatchingMSSPCertificates">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Certificate"
                      type="xs:base64Binary"
                      minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="MatchingAPCertificates">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DN"
                      type="xs:base64Binary"
                      minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="MatchingSigAlgList">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Algorithm"
                      type="mss:mssURIType"
                      minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="MSSP_TransID"
                type="xs:NCName"
                use="required"
</xs:extension>
```

```
  </xs:complexContent>
</xs:complexType>
```

A.9 MSS_Notification-palvelu

Seuraavassa on kuvattu MSS_Notification-palvelun WSDL-kuvaus:

```
<message name="MSS_NotificationInput">
  <part name="MSS_NotificationRequest"
        type="mss:MSS_NotificationRequestType"/>
</message>
<message name="MSS_NotificationOutput">
  <part name="MSS_NotificationResponse"
        type="mss:MSS_NotificationResponseType"/>
</message>

<portType name="MSS_NotificationPortType">
  <operation name="MSS_Notification">
    <input message="tns:MSS_NotificationInput"/>
    <output message="tns:MSS_NotificationOutput"/>
  </operation>
</portType>
```

MSS_NotificationRequest-viestin XML-skeema -määritys:

```
<xs:element name="MSS_NotificationRequest"
            type="mss:MSS_NotificationRequestType"/>
<xs:complexType name="MSS_NotificationRequestType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType"/>
        <xs:element name="Status"
                    type="mss:StatusType"/>
        <xs:element name="SignatureProfile"
                    type="mss:mssURIType"
                    minOccurs="0"/>
        <xs:element name="MSS_Signature"
                    type="mss:SignatureType"
                    minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="MSSP_TransID"
                    type="xs:NCName"
                    use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

MSS_NotificationResponse-viestin XML-skeema -määritys:

```
<xs:element name="MSS_NotificationResponse"
            type="mss:MSS_NotificationResponseType"/>
<xs:complexType name="MSS_NotificationResponseType">
  <xs:complexContent>
    <xs:extension base="mss:MessageAbstractType">
      <xs:sequence>
        <xs:element name="MobileUser"
                    type="mss:MobileUserType/>
        <xs:element name="MSS_Signature"
                    type="mss:SignatureType"
                    minOccurs="0"/>
        <xs:element name="Status"
                    type="mss:StatusType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```