Metropolia Ammattikorkeakoulu
Mediatekniikan koulutusohjelma

Antti Hämäläinen

CAT_TP-tiedonsiirtoprotokollan testausjärjestelmä

Insinöörityö 4.5.2009

Ohjaaja: diplomi-insinööri Teemu Asikainen
Ohjaava opettaja: lehtori Petri Vesikivi

Metropolia Ammattikorkeakoulu Insinöörityön tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Antti-Ville Hämäläinen CAT_TP-tiedonsiirtoprotokollan testausjärjestelmä 79 sivua 4.5.2009
Koulutusohjelma	mediatekniikka
Tutkinto	insinööri (AMK)
Ohjaaja Ohjaava opettaja	diplomi-insinööri Teemu Asikainen lehtori Petri Vesikivi

Insinöörityön aiheena oli CAT_TP (card application toolkit transport protocol) -tiedonsiirtoprotokollan testausjärjestelmä ja työn tilaajana matkapuhelinteknologia-alan yritys. CAT_TP-tiedonsiirtoprotokolla on yksi useista OTA (over-the-air) -tekniikoista, joita käytetään älykorttien etähallintaan. CAT_TP mahdollistaa palvelimen ja älykortin välisen kaksisuuntaisen kommunikoinnin. Yritys toimii NFC (near field communication) -ekosysteemissä Trusted Services Managerina (TSM), joka hallinnoi palveluntarjoajien sovelluksia älykorteilla niiden koko elinkaaren ajan.

NFC-teknologia on lyhyen matkan kontaktiton yhteysteknologia. Se tekee mahdolliseksi, että matkapuhelin voi toimia esimerkiksi pankkikorttina. Matkapuhelimen (U)SIM (universal subscriber identity module) -kortille tallennetaan palveluntarjoajan, kuten pankin, NFC-sovellus. NFC-sovellukset ladataan langattomasti OTA-tekniikoilla, kuten CAT_TP:llä, (U)SIM-kortille TSM:n toimesta.

Testausjärjestelmän tarkoituksena oli varmistaa, että yrityksen CAT_TP-toteutus toteuttaa CAT_TP-spesifikaation. Testausjärjestelmä toteutettiin ETSI (European Telecommunication Standards Institute) -järjestön määrittämän CAT_TP-testausspesifikaation pohjalta. Testausjärjestelmän testit ovat JUnit-testejä, jotka voidaan ajaa täysin automatisoidusti.

CAT_TP-toteutuksen testaamista varten kehitettiin 106 testiä, jotka testaavat sitä kaikilla mahdollisilla tavoilla. Testeissä huomioitiin niin toivotut kuin epätoivotut tilanteet. Testausjärjestelmän tuella yritys pystyy kehittämään helpommin taustajärjestelmänsä CAT_TP-toteutusta ja varmistamaan, että lopullinen toteutus on varmasti spesifikaation mukainen. Näin voidaan varmistaa, että yrityksen taustajärjestelmän ja älykortin välinen kommunikointi on mahdollisimman saumatonta. Testausjärjestelmä on integroituna yrityksen CAT_TP-toteutukseen, ja alustavia testejä on jo suoritettu.

Hakusanat	CAT_TP, BIP, OTA, älykortit, NFC

Helsinki Metropolia University of Applied Sciences **Abstract**

Author	Antti-Ville Hämäläinen
Title	CAT_TP conformance test bench
Number of pages	79 pages
Date	4.5.2009
Degree programme	Media Technology
Degree	Bachelor of Engineering
Instructor	Teemu Asikainen, M.Sc. Techn.
Supervisor	Petri Vesikivi, Lecturer

The topic of the Bachelor's thesis was CAT_TP (card application toolkit transport protocol) transport protocol's conformance test bench subscribed by a company in mobile technology business. CAT_TP protocol is one of many over-the-air (OTA) technologies which are used for remote management of smart cards. CAT_TP protocol allows a data channel to be established between a smart card and a server. The company plays the role of a trusted services manager (TSM) in the near field communication (NFC) ecosystem. They manage remotely service provider's applications in the smart cards through their whole life-cycle.

NFC technology is a short range contactless connectivity technology. It enables that a mobile phone can act as a bank card. Service provider's (e.g. a bank) NFC application is stored to the mobile phone's universal subscriber identity module (USIM). NFC applications are downloaded wirelessly to the USIM with OTA technologies, such as CAT_TP, by a TSM.

The aim of the conformance test bench is to ensure that the company's CAT_TP implementation specifies the CAT_TP specification. The conformance test bench was implemented with the help of a test specification provided by European telecommunications standards institute (ETSI). All test cases in the conformance test bench are JUnit tests which can be executed in a fully automated way.

106 test cases were implemented for the test bench which tested the CAT_TP implementation in all possible ways. The expected as well as the unexpected cases were taken into account. With the conformance test bench the company can ensure that their CAT_TP implementation specifies the CAT_TP specification, but also the development of the implementation is easier with the help of it. When the CAT_TP implementation is implemented in accordance to the CAT_TP specification, the interoperability between it and a remote CAT_TP entity can be ensured. The CAT_TP test bench is integrated to the company's CAT_TP implementation and some tentative tests has already been done.

Keywords	CAT_TP, BIP, OTA, Smart cards, NFC

Sisällys

Liitteet

Lyhenteet

3G	Third Generation
3GPP	3rd Generation Partnership Project
AID	Application Identifier
APDU	Application Protocol Data Unit
ATR	Answer To Reset
BIP	Bearer Independent Protocol
CAT	Card Application Toolkit
CAT_TP	Card Application Toolkit Transport Protocol
DES	Data Encryption Standard
ECMA	European Computer Manufacturers Association
EDGE	Enhanced Data rates for Global Evolution
ETSI	European Telecommunication Standards Institute
FID	File Identifier
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HCI	Host Controller Interface
IEC	International Electrotechnical Commission
IrDA	Infrared Data Association
ISD	Issuer's Security Domain
ISO	International Organization for Standardization
JCRE	Java Card Run Environment
JCVM	Java Card Virtual Machine
NFC	Near Field Communication
OTA	Over The Air
PCD	Proximity Coupling Device
PDU	Protocol Data Unit
PIX	Proprietary application Identifier Extension
RAM	Remote Application management

RFID	Radio Frequency Identification
RFM	Remote File Management
RID	Registered Identifier
RSA	Ron Rivest, Adi Shamir ja Leonard Adleman
(U)SAT	(Universal) Sim Application Toolkit.
SDU	Service Data Unit
(U)SIM	(Universal) Subscriber Identity Module
SMS	Short Message Service
SMSC	Short Message Service Center
SMS-PP	Short Message Service Point to Point
SSD	Suplementary Security Domain
SWP	Single Wire Protocol
TCP	Transmission Control Protocol
TSM	Trusted Services Manager.
UDP	User Datagram Protocol
UICC	Universal Integrated Circuit Card
VCD	Vicinity Coupling Device

1 Johdanto

Insinöörityön tarkoituksena on tehdä CAT_TP-tiedonsiirtoprotokollan testausjärjestelmä, ja työn tilaajana on Venyon Oy. Venyon hoitaa palveluntarjoajien, kuten pankkien ja luottokortti- ja joukkoliikenneyhtiöiden, sovellusten turvallisen lataamisen ja koko elinkaaren aikaisen hallinnoinnin matkapuhelimessa. CAT_TP (card application toolkit transport protocol) on tiedonsiirtoprotokolla älykortin ja taustajärjestelmän väliseen kommunikointiin. Sitä käytetään älykorttien langattomaan hallinnointiin, kuten sovellusten lataamiseen matkapuhelimen UICC (universal integrated circuit card) -kortille. CAT_TP on luotettava vaihtoehto muille OTA (over the air) -tekniikoille, kuten SMS-viesteille ja toisille BIP (bearer independent protocol) -protokollille. Matkapuhelimen UICC-kortti tunnetaan yleisemmin myös nimellä (U)SIM (universal subscriber identity module) -kortti.

NFC (near field communication) -teknologia mahdollistaa sen, että matkapuhelin voi toimia luottokorttina, matkakorttina ja vaikka avaimena. NFC:tä tukevan puhelimen avulla voidaan yhdistää kaikki lompakon kortit yhteen, sen avulla pystytään jakamaan dataa ja hankkimaan lisätietoa. NFC-matkapuhelin tuo etuja verrattuna tavalliseen muoviseen luottokorttiin, koska siinä on käyttöliittymä. Näin saadaan lisävuorovaikutusta käyttäjän ja kortin välille. Kuluttaja pystyy tarkistamaan suoraan kortilta viimeisimmät nostot ja seuraamaan matkakorttinsa saldoa. NFC:tä voidaan myös käyttää Bluetooth-yhteyden tavoin kahden NFC:tä tukevan laitteen väliseen kommunikointiin, joten sitä voidaan soveltaa lukuisin eri tavoin.

Koska NFC-teknologia mahdollistaa matkapuhelimen toimivan luotto- ja pankkikorttina, on turvallisuus kriittinen vaatimus useille NFC-sovelluksille. Näin ollen on parasta sijoittaa NFC-sovellus erillisen turvaelementin (secure element) sisään. Matkapuhelimissa se on yleensä sen (U)SIM-kortti, mutta turvaelementtinä voi toimia myös erillinen älykortti. OTA-tekniikan avulla voidaan toimittaa NFC-sovelluksia matkapuhelimen UICC-kortille ja hallinnoida niitä siellä. OTA-tekniikan ansiosta matkapuhelinta ei tarvitse koskaan viedä fyysisesti mihinkään asiakaspalvelupisteeseen, jotta siihen saataisiin NFC-

toiminnallisuutta. OTA-tekniikka mahdollistaa älykortin ja taustajärjestelmän välisen kommunikoinnin langattomasti.

Venyon toimii TSM:nä (trusted services manager) NFC-ekosysteemissä eli se hallinnoi NFC-sovelluksia matkapuhelimessa koko niiden elinkaaren ajan. Tähän se käyttää OTA-tekniikkaa. Venyonin nykyinen toteutus hallinnoi matkapuhelimen UICC-korttia erillisen Java Midlet -sovelluksen kautta, mutta tulevaisuudessa tarkoituksena on hallinnoida UICC-korttia ilman sitä, käyttäen CAT_TP-protokollaa. Insinöörityönä tehtävän CAT_TP-testausjärjestelmän avulla on tarkoitus varmistaa, että taustajärjestelmän CAT_TP-toteutus menettelee kaikissa tilanteissa standardien mukaisesti kommunikoidessaan älykortin kanssa.

Insinöörityöraportissa käsittelen CAT_TP-testausjärjestelmän toiminnallisuuden ja toteutuksen lisäksi NFC-teknologiaa matkapuhelimissa painottaen älykortin roolia ja tekniikoita hallinnoida sitä langattomasti.

2 NFC-teknologia

2.1 NFC-arkkitehtuuri

Nykypäivänä esiintyy monia kontaktittomia yhteysteknologioita, mutta yksi kiinnostavimmista on NFC-teknologia. Kontaktiton yhteysteknologia on älykortti-valmistajien kehittämä termi, jolla tarkoitetaan laitteiden välistä kommunikaatiota lyhyen matkan päästä. NFC-teknologia polveutuu kontaktittomien tunnistamis- ja yhteys-teknologioiden yhdistelmästä, kuten Philipsin Mifare- ja Sonyn FeliCa-teknologioista, jotka ovat laajasti tunnettuja kontaktittomia älykorttiteknologioita. Siinä yhdistyy lukijalaiteen ja älykortin toiminnot ja vertaisverkkokommunikaatio. NFC on lyhyen kantaman langaton yhteysteknologia, joka mahdollistaa lyhyen matkan, noin 0–10 cm, turvallisen kontaktittoman tunnistamisen ja kommunikoinnin laitteiden välillä. NFC-kommunikaatio on nopeaa, yksinkertaista ja turvallista. NFC-teknologian kehityksen aloittivat Sony

Corporation ja Royal Philips Electronics syksyllä 2002, ja myöhemmin se hyväksyttiin ECMA (European Computer Manufacturers Association) - ja ISO/IEC (International Organization for Standardization) -standardeiksi [1]. NFC:tä kehitetään pääosin matkapuhelimille, mutta myös muiden elektronisten laitteiden väliseen kommunikaatioon. Matkapuhelin-NFC:ssä yhdistyvät NFC-yhteysteknologia, matkapuhelintekniikka ja UICC-kortti, mikä tuo merkittäviä etuja pelkkään perinteiseen kontaktittomaan älykorttiin verrattuna. Näin pystytään yhdistämään NFC-kommunikointi, matkapuhelimen näyttö ja näppäimistö ja tietoliikenneverkkoyhteys. Matkapuhelin-NFC mahdollistaa esimerkiksi nopeat ja vaivattomat pankki- ja luottokorttimaksut. [2, s. 4.]

Kahden laitteen välinen NFC-kommunikaatio voidaan aloittaa yksinkertaisesti viemällä laitteet hyvin lähelle toisiaan. Laitteet muodostavat linkin toistensa välille hyvin nopeasti, noin 100–200 millisekunnissa [3]. NFC-laitteet kommunikoivat sähkömagneettisen kentän välityksellä. Vähintään toisen kommunikaation osapuolen täytyy muodostaa kenttä. NFC-teknologia toimii 13,56 megahertsin taajuudella. NFC-laite pystyy siirtämään dataa 106, 212 tai 424 kilobittiä sekunnissa [4, s. 5–6]. 13,56 megahertsin taajuudella ei tarvita kommunikaatiolle lisenssejä, eikä sillä ole suurempia rajoitteita, paitsi tietenkin joitakin maakohtaisia voi esiintyä [5, s. 7].

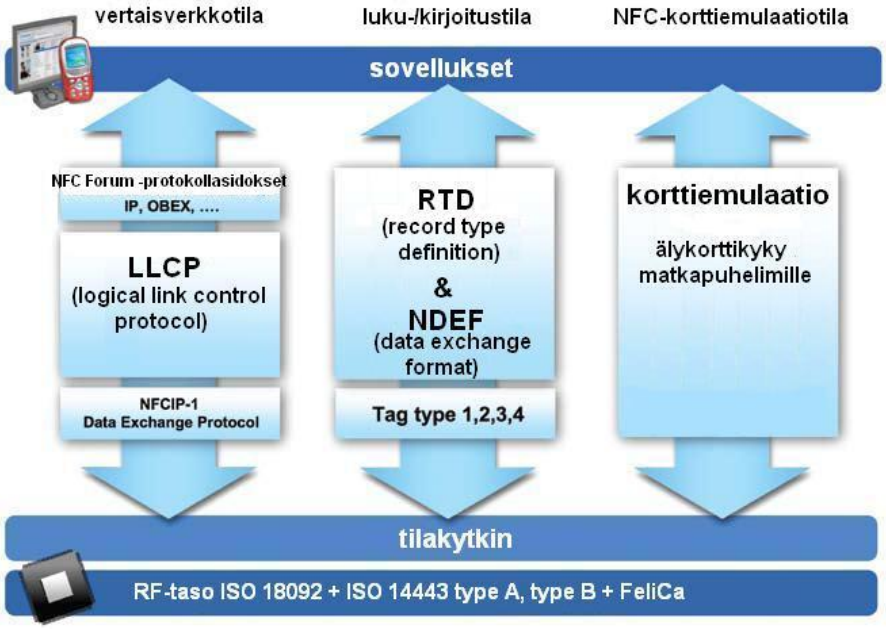
NFC-teknologia pohjautuu RFID (radio frequency indentification) -tekniikkaan. RFID on yleinen termi, jolla tarkoitetaan teknologioita, jotka käyttävät radioaaltoja tunnistamiseen. RFID-tekniikassa lukijalaite lähettää antennin kautta radioaaltoja tunnisteille pyytäen niitä lähettämään tietonsa takaisin lukijalaitteelle. Tunnisteet koostuvat mikrosirusta ja antennista. Tunnisteiden mikrosiruille voidaan tallentaa ja niistä voidaan hakea dataa ja jokaisen tunnisteen mikrosirulla on sille ominaiset tunnistetiedot tallennettuna. Pyynnön saadessaan tunniste hakee mikrosirultaan tarvittavat tiedot ja lähettää ne takaisin lukijalaitteelle. Tiedonsiirto tapahtuu amplitudimoduloimalla lukijalaitteen lähettämää radioaaltoa, jolloin lukijalaite havaitsee radioaallon muutokset, jotka se osaa muuntaa digitaaliseksi informaatioksi. Lukijalaite vastaa aina yhteyden muodostamisesta, varmentamisesta ja törmäysten estämisestä. Tunnisteilla ei ole omaa virtalähdettä, joten ne

saavat tarvittavan virran lukijalaitteen lähettämistä sähkömagneettisista aalloista. Tunniste indusoi sähkömagneettiset aallot tarvittavaksi virraksi. RFID-järjestelmät voivat käyttää monia eri toimintataajuuksia, matalataajuuksista hyvin korkeisiin taajuuksiin. Näin ollen sen kantama voi vaihdella nollasta kolmeen metriin. [6.]

NFC-laitteet voivat kommunikoida joko perinteisellä asiakas-palvelinmallilla tai muodostamalla vertaisverkon (peer-to-peer), jolloin molemmat laitteet toimivat sekä asiakkaana että palvelimena. Laite, joka aloittaa NFC-kommunikaation, on aloitteentekijä (initiator). Nimensä mukaisesti se aloittaa kommunikaation ja hallitsee datan siirtoa. Toinen laitteista toimii kohteena (target). Kohde vastaa aloitteentekijän kutsuihin. [5, s. 7.]

NFC-kommunikaatio voi olla aktiivista tai passiivista. Aktiivisessa kommunikaatiossa molemmat laiteet, aloitteentekijä ja kohde, muodostavat omat sähkömagneettiset kenttänsä tiedonsiirtoon. Aktiivista kommunikaatiota kutsutaan myös vertaisverkkotilaksi (peer-to-peer mode). Passiivisessa kommunikaatiossa kohde indusoi tarvittavan virran aloitteentekijän luomasta sähkömagneettisesta kentästä. Passiivinen kommunikaatio on hyvin tärkeää, varsinkin puhuttaessa laitteista, jotka saavat virtansa akuista, koska tällöin laitteen ei tarvitse turhaan käyttää akkua kentän muodostamiseen ja joissakin tilanteissa matkapuhelimessa ei tarvitse olla edes akkua kommunikointiin toisen laitteen kanssa. Tällöin matkapuhelin toimii vain perinteisen kontaktittoman älykortin tavoin. Kun matkapuhelin luo sähkömagneettisen kentän passiivisessa kommunikaatiossa, tilaa kutsutaan luku-/kirjoitustilaksi (read/write mode). Passiivinen kommunikaatio, jossa matkapuhelin indusoi tarvittavan virran, on NFC-korttiemulaatiotila (NFC card emulation mode). [5, s. 7–8; 7; 8.] NFC-laite on laite, joka pystyy aktiiviseen tilaa eli luomaan oman sähkömagneettisen kentän. NFC-tunnisteita ovat laitteet, jotka ovat aina passiivisessa tilassa. NFC-tunnisteista yleensä luetaan dataa NFC-laitteella, mutta niihin voidaan myös kirjoittaa dataa. NFC-laitteille on myös ominaista, että ne voivat vaihdella tilaansa. [9.]

Kuvassa 1 on esitetty NFC-arkkitehtuuri. Siitä nähdään NFC-kommunikaation mahdolliset eri tilat ja standardit, jotka niiden pitää määrittää.



Kuva 1. NFC-arkkitehtuuri [10].

Tilakytkin (mode switch) on yksi NFC:n peruskomponenteista. Sen vastuulla on muiden laitteiden havainnointi, laitteen oman olemassaolon tiedottaminen ja laitteiden välisen yhteyden sulkeminen. Tilakytkimen tulee tukea kaikkia NFC-standardeja.

2.2 NFC-teknologian standardit

NFC:n määrittävät ECMA 340 (NFC IP-1) -, ISO/IEC 18092 (NFC IP-1) -, ECMA 352 (NFC IP-2)- ja ISO/IEC 21481 (NFC IP-2) -standardit. NFC IP-1 ja NFC IP-2 ovat molemmat ECMA:n määrittämiä standardeja, jotka on myöhemmin hyväksytty myös ISO/IEC-standardeiksi. NFC IP-1 on myös alaspäin yhteensopiva ISO\IEC 14443-A (Mifare)- ja ISO\IEC 14443-C (FeliCa) -standardien kanssa. ISO\IEC 14443 -standardi on neliosainen kansainvälinen standardi lyhyen kantaman kontaktittomille älykorteille, jotka

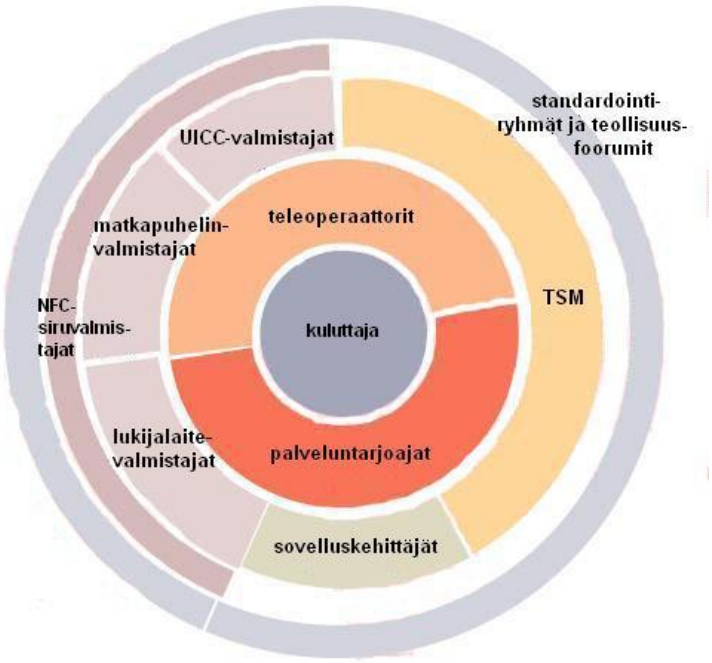
toimivat 13,56 megahertsin taajuudella [11, s. 15]. ISO\IEC 14443 -standardin neljä osaa määrittävät kortin fyysiset ominaisuudet (ISO 14443-1), sähkömagneettisen kentän signaalirajapinnan (ISO 14443-2), protokollat yhteyden alustamiseen ja törmäyksenhallintaan (ISO 14443-3) ja protokollat tiedonsiirtoon (ISO 14443-4). [12.] Alaspäin yhteensopivuuden ansiosta nykyisiä RFID-lukijoita ja -tunnisteita ei tarvitse vaihtaa, jotta niistä saadaan NFC-yhteensopivia, mikä onkin yksi palveluntarjoajien vaatimus NFC-teknologialta. NFC IP-1 -standardi määrittää NFC:n aktiivisen ja passiivisen tilan, sähkömagneettisen kentän rajapinnan, toimintamallin yhteyden alustamiselle ja törmäyksenhallintaan sekä protokollat tiedonsiirtoon [13].

NFC IP-1- ja ISO/IEC 14443 -standardit on molemmat määritetty toimintataajuudelle 13,56 MHz. Lisäksi samalle taajuudelle on määritetty ISO/IEC 15693 -standardi, joka on standardi älykorteille, jotka pystyvät kommunikoimaan pidemmän matkan päähän, noin 1–1,5 metriin. Vaikka NFC IP-1-, ISO/IEC 14443- ja ISO/IEC 15693 -standardit määrittävät kaikki toimintataajuudekseen 13,56 MHz, niiden kommunikaatiomallit eriävät. NFC IP-1 käyttää NFC-kommunikaatiomallia, ISO/IEC 14443 -standardissa on määritelty PCD (proximity coupling device) -kommunikaatiomalli ja ISO/IEC 15693 -standardissa on määritelty VCD (vicinity coupling device) -kommunikaatiomalli. NFC IP-2 -standardi on määritelty havaitsemaan nämä eri kommunikaatiomallit ja valitsemaan niistä tarvittava. Se on suunniteltu niin, että se ei häiritse meneillään olevia yhteyksiä samalla taajuudella. [14, s. 5.]

2.3 NFC-ekosysteemi

NFC-teknologiasta puhuttaessa tulee usein esiin termi NFC-ekosysteemi. Aivan kuten biologian ekosysteemi, niin myös NFC koostuu monista toisistaan riippuvaisista entiteeteistä: on laitteiden ja älykorttien valmistajia, palveluntarjoajia, teleoperaattoreita, TSM:iä, standardien kehittäjiä ja tietenkin loppukäyttäjiä. Niiden kaikkien täytyy tehdä tiivistä yhteistyötä, jotta tarjolle saadaan parempia NFC-palveluita.

Kuvassa 2 on esitetty NFC-ekosysteemin entiteetit. Kaiken keskellä on loppukäyttäjä, jolle muut entiteetit yrittävät yhdessä tuottaa mahdollisimman helppokäyttöisiä ja luotettavia palveluita.



Kuva 2. NFC-ekosysteemi [11, s. 40].

Käyttäjää ympyröivät palveluntarjoajat ja teleoperaattorit, joilta käyttäjät tilaavat palveluita. Palveluntarjoajat ja teleoperaattorit yhdistää luotettava kolmas osapuoli, TSM, eivätkä niiden palvelut olisi mahdollisia ilman NFC-siru-, älykortti-, matkapuhelin- ja lukijalaitevalmistajia. Täytyy myös olla jokin taho, joka toteuttaa sovellukset, eli ohjelmistokehittäjät. Kaikkea ympäröivät standardien kehittäjät, joiden työ edistää muiden NFC-ekosysteemin entiteettien saumatonta yhteistyötä.

NFC-entiteetit ja niiden roolit

Teleoperaattorit ovat avainasemassa NFC-ekosysteemissä, sillä yhteistyö niiden ja muiden NFC-ekosysteemin entiteettien kesken on välttämätöntä NFC:n menestymisen kannalta.

NFC-sovellukset sijoitetaan niiden omistamille UICC-korteille. Teleoperaattorin tulee tarjota sovellukselle oma SSD (suplementary security domain) -tietoturva-alue UICC-kortilta ja avaimet sen hallinnointiin. Teleoperaattorin (U)SIM-sovelluksen ja palveluntarjoajien NFC-sovelluksien täytyy olla samalla UICC-kortilla. Teleoperaattorin tulee myös mahdollistaa OTA-tekniikka kortille, jotta voidaan toimittaa NFC-sovellus ja ylläpitää sitä käyttäjän UICC-kortilla sen toimituksen jälkeen. Käyttäjät tilaavat teleoperaattorilta matkapuhelinverkon käyttöönsä, ja operaattori toimittaa käyttäjälle UICC-kortin. Teleoperaattori tarjoaa käyttäjälle kaikki matkapuhelinpalvelut ja asiakaspalvelun. [15, s. 16.]

Palveluntarjoajat tarjoavat lopullisen konkreettisen palvelun käyttäjälle ja tahtovat, että se sijoitetaan matkapuhelimen (U)SIM-kortille. Palveluntarjoajia ovat pankit, luottokorttiyhtiöt, joukkoliikenneyhtiöt, tapahtumajärjestäjät, vähittäiskauppiaat ja niin edelleen. Myös teleoperaattori voi toimia palveluntarjoajana. Palveluntarjoajat tarjoavat palvelua ostosten ja joukkoliikenteen ja tapahtumien lippujen maksamiseen matkapuhelimella. Palveluntarjoajat pyytävät TSM:iltä apua sovellusten lataamiseksi, asentamiseksi ja koko elinkaaren aikaiseen hallinnointiin teleoperaattorin (U)SIM-kortilla. Palveluntarjoajat ovat myös avainasemassa, koska niiden palveluiden toimivuus määrittää NFC:n tulevaisuuden. Niiden pitää tarjota oikeanlaisia, helppokäyttöisiä ja korkean tason turvallisuudella varmistettuja palveluita käyttäjille, jotta nämä myös tulevaisuudessa haluavat käyttää niitä. Palveluntarjoajien tavoitteena on tietenkin saada sovelluksensa mahdollisimman monen käyttäjän matkapuhelimiin ja mahdollisimman monen teleoperaattorin (U)SIM-kortille. Vähittäiskauppiailta, kuten myymälöiltä ja ravintoloilta, odotetaan, että ne tarjoavat vaihtoehdon maksaa NFC-tekniikalla. Oletetaan kuitenkin, että myymälöihin tulee NFC:n yleistyessä, enemmin tai myöhemmin, mahdollisuus maksaa matkapuhelimella. [2, s. 11; 16, s. 5.]

Tietenkään NFC ei ole mahdollinen ilman laitteiden valmistajia. Tarvitaan valmistajat NFC-siruille, älykorteille, NFC-yhteensopiville matkapuhelimille ja kortinlukijoille. NFC-laitteet ja kaikki sen komponentit ovat NFC-teknologian kannalta elintärkeitä. Kaikki NFC-

sovellukset ovat käyttäjien kannalta hyödyttömiä, jollei käyttäjien saataville saada tarpeeksi paljon niihin kykeneviä laitteita. Jotta kaikki olisi mahdollisimman yhteensopivaa, täytyy valmistajien ja kehittäjien tarkasti seurata NFC-teknologian kansainvälisiä standardeja. Standardien tarkka noudattaminen varmistaa, että NFC-palvelut toimivat saumattomasti NFC:n kaikilla tasoilla, laitteista käyttäjään. Mahdollisimman tarkoista ja hyvistä standardeista ja spesifikaatioista vastaavat sellaiset tahot kuin NFC Forum, ECMA, ETSI (european telecommunication standards institute), 3GPP (3rd generation partnership project) ja ISO/IEC. Näissä tahoissa ovat mukana maailman suurimmat matkapuhelin-, älykortti-, NFC-siru- ja lukijalaitevalmistajat. Ne kaikki yhdessä pyrkivät kehittämään NFC-teknologiaa eteenpäin. Niiden lisäksi tarvitaan ohjelmistokehittäjiä, jotka tekevät NFC-sovellukset. Usein palveluntarjoajat vain määrittävät sovelluksen, mutta jokin muu taho toteuttaa sen.

TSM-entiteetti

TSM eli Trusted Services Manager on NFC-ekosysteemin uusin entiteetti. TSM:t toimittavat ja hallitsevat palveluntarjoajien palveluita käyttäjien matkapuhelimien UICC-korteilla. Ne hallitsevat palveluita matkapuhelimissa niiden koko elinkaaren ajan, lataamisesta poistoon. TSM toimii luotettavana välikätenä teleoperaattorin ja palveluntarjoajan välissä ja palveluntarjoajan ja loppukäyttäjän välissä. TSM:n rooli tulee sopia tarkasti päätekijöiden kesken, jotta loppukäyttäjälle voidaan järjestää palveluita tehokkaasti. TSM ei vaikuta NFC-sovelluksen transaktiovaiheeseen mitenkään, mutta varmistaa, että palveluntarjoajan toimintamalli ei rikkoudu missään välissä. TSM:n tulee pystyä toimimaan yhden tai useamman palveluntarjoajan ja teleoperaattorin välikätenä. TSM:iltä odotetaan hyvää mainetta ja luotettavaa palveluiden käsittelyä ja sen alustalta hyvää turvallisuutta, saatavuutta ja skaalattavuutta. [2, s. 13; 15, s. 21.] Liitteessä 1 on esimerkkikuva päätekijöiden välisistä suhteista, kun käytössä on sovellus maksamiseen matkapuhelimella.

TSM:n roolin voi yksityisen kolmannen osapuolen sijaan hoitaa myös teleoperaattori, palveluntarjoaja tai useampi edellä mainituista yhdessä. TSM-palveluita tarjoavan kolmannen osapuolen hyödyntämisessä on suuri etu, koska pelkästään TSM-palveluihin erikoistuneet tahot ovat yleensä kokeneempia kuin ne, jotka pyrkivät vaikuttamaan useilla NFC:n alueilla. Erillisen kolmannen osapuolen hyödyntäminen auttaa teleoperaattoreita ja palveluntarjoajia, kun niiden tarvitsee yhdistää infrastruktuurinsa vain yhden tahon kanssa. Ilman erillistä TSM:ää, palveluntarjoajien pitää integroida palvelunsa jokaiselle teleoperaattorille erikseen. Tämä onkin yksi syy, miksi TSM-roolia on ehdotettu NFC-ekosysteemiin. Myös jos teleoperaattori tai palveluntarjoaja hoitaa lisäksi TSM:n roolia, voi tulla epäselvyyksiä, mitä kunkin osapuolen tulee määrittää. TSM-infrastruktuurin rakentaminen erikseen, oman muun infrastruktuurin lisäksi, on myös suuri työ. Kun kaikki osapuolet hoitavat sen roolin, johon niillä on eniten ammattitaitoa, tai kaikki luovat yhdessä toimivan ympäristön, on paremmat mahdollisuudet tuottaa loppukäyttäjälle hyvä, toimiva palvelu. [15, s. 19–20.]

Teleoperaattori myöntää oman SSD-tietoturva-alueen TSM:lle, se määrittää sille tarvittavat oikeudet ja muistikiintiöt ja antaa avaimet sen hallinnointia ja OTA-alustaa varten. Sovellusten OTA-hallinnointia varten tietoturva-alueella täytyy olla RAM (remote application management) -hallintaohjelma. RAM-hallintaohjelmaa käytetään nimen mukaisesti sovellusten etähallinnointiin OTA-tekniikalla. Saatuaan tarvittavat avaimet teleoperaattorilta TSM pystyy lataamaan (U)SIM-kortille palveluntarjoajien pyynnöistä sovelluksia. TSM ja palveluntarjoaja vaihtavat tietoturva-alueen avaimia ja sovelluskohtaisia yksityiskohtia. Palveluntarjoajat voivat TSM:n rajapinnan kautta seurata kutsuja ja sovellusten statuksia, ja ne voivat pyytää TSM:ää tekemään sovelluspäivityksiä. TSM:n tulee myös toteuttaa teleoperaattorille rajapinta, jonka kautta voidaan hoitaa raportointia, laskutusta, asiakaspalvelua ja latausten oikeuttaminen. Palveluntarjoaja ei ole ainoa, joka voi tilata TSM:ltä palvelua, vaan joskus myös teleoperaattori voi tilata TSM:ltä palvelua (U)SIM-kortin tietoturva-alueiden muistikiintiöiden hallintaan.

Palveluntarjoajille TSM voi tarjota kommunikaatiokanavan sovellukseen. Sen kautta asiakas pystyy esimerkiksi lataamaan sovellukseensa lisää aikaa, rahaa tai pääsyoikeutta. Esimerkiksi julkisen liikenteen yhtiöt voivat tarjota mahdollisuutta ladata lisää matkustusaikaa matkakorttisovellukseen matkapuhelimessa. Tähän käytetään OTA-tekniikoita, jotta lataus voidaan tehdä mistä vain ja mihin aikaan vain. Tällöin käyttäjä tekee tilauksen ja TSM toimittaa tilauksen käyttäjän UICC-kortille. On tärkeää, että TSM:n matkapuhelinrajapinta pystyy kommunikoimaan kaikkien teleoperaattoreiden myöntämien (U)SIM-korttien, kaikentyyppisten UICC-korttien ja kaikkien matkapuhelinmallien kanssa. Tätä varten se joutuu luultavimmin tekemään alustalleen toteutukset useammille OTA-tekniikoille.

TSM:n alustan tulee olla hyvin turvallinen. Sitä käytetään sellaisten sovelluksien lataamiseen ja hallinnointiin, joille turvallisuus on kriittistä. Jotkin sovellukset, kuten maksusovellukset, vaativat, että alustalla on Visan ja Mastercardin sertifikaatit.

2.4 NFC:n käyttökohteet

NFC-teknologialta odotetaan maailman mullistamista. Odotetaan, että sen helppo ja luotettava kahdensuuntainen kommunikointi elektronisten laitteiden kesken tulee olemaan kaikkialla läsnä. On lukemattomia käyttökohteita, joissa pystyttäisiin hyödyntämään NFC-teknologiaa. Sen pääkäyttökohteet ovat laitteiden langaton kommunikointi, pääsy digitaaliseen informaatioon ja transaktiot, jotka sisältävät ostosten ja lippujen maksamisen matkapuhelimella. [10, s. 1.]

Langattomaan kommunikointiin laitteiden kesken on myös monia muita teknologioita, kuten Bluetooth-, IrDA- ja Wi-Fi-yhteydet, mutta NFC:n etu on sen helppo ja nopea linkin muodostus laitteiden välille. Esimerkiksi Bluetooth-yhteyden muodostaminen laitteiden välille vie aikaa, kun yhteys pitää manuaalisesti alustaa. NFC luo laitteiden välille linkin välittömästi, kun ne viedään lähelle toisiaan. NFC:n avulla voidaan hetkessä vaihtaa laitteiden kesken multimediatiedostoja, käyntikortteja tai kalenterimerkintöjä. Jos tarvitaan

nopeampaa Bluetooth-yhteyttä laitteiden väliseen tiedonvaihtoon, NFC mahdollistaa myös Bluetooth-yhteyden muodostamisen laitteiden välillä ilman, että käyttäjän tarvitsee itse manuaalisesti alustaa yhteyttä. Matkapuhelimien lisäksi sitä voidaan hyödyntää digitaalikameroissa, televisioissa, tulostimissa ja monissa muissa laitteissa. Esimerkiksi kuvien tulostaminen digitaalikamerasta voidaan laukaista viemällä kamera tulostimen lähelle. [7, s. 3–4.] Liitteessä 2 on tarkemmin esitetty eri langattomien yhteysteknologioiden eroavaisuuksia.

Yksi NFC:n pääkäyttökohteista on digitaalisen informaation jakaminen. NFC:tä voidaan hyödyntää esimerkiksi aikataulujen jakamiseen linja-autopysäkeillä. Viemällä matkapuhelin lähelle pysäkillä olevaa NFC-tunnistetta saadaan puhelimeen linja-auton aikataulu ja verkosta voidaan saada tietoa mahdollisista myöhästymisistä. Samantapaisia keinoja voidaan hyödyntää älyjulisteissa (smart poster), kuten esimerkiksi katumainoksissa. Kadulla oleviin mainostauluihin voidaan sijoittaa NFC-tunnisteita, joiden kautta pystytään jakamaan lisää tietoa mainoksesta. Tunniste voi esimerkiksi käynnistää matkapuhelimen selaimen ja siirtyä mainostajan verkkosivuille. Näin käyttäjän ei tarvitse itse painaa URL-osoitetta muistiin tai syöttää sitä selaimelle. Älyjulisteista on myös mahdollista ladata esimerkiksi elokuvan ennakkomainos tai vaikka uusi soittoääni matkapuhelimeen. Matkustettaessa voitaisiin hakea turisti-informaatio omalla äidinkielellä NFC-turistipisteistä. NFC-tunnisteita voidaan käyttää hyväksi lukemattomin tavoin. [10, s. 3.]

NFC:n suurin käyttökohde tulee kuitenkin olemaan ostosten ja julkisen liikenteen ja tapahtumien lippujen maksaminen matkapuhelimilla. NFC-teknologian avulla voidaan maksaa luotettavasti myymälöissä ja ravintoloissa heilauttamalla matkapuhelinta lukijassa. Myös useissa maissa käytössä olevat matkakortit voidaan korvata matkapuhelimella. Etu perinteisiin luotto-, pankki- ja matkakortteihin verrattuna tulee olemaan se, että matkapuhelimen avulla näihin palveluihin saadaan lisäarvoa (value added services). Matkapuhelimien tuotettu lisäarvo on näyttö, näppäimistö ja tietoliikenneverkko, jotka tekevät mahdolliseksi esimerkiksi nähdä viimeisimmät ostot matkapuhelimessa tai matkakortin arvon. Myös matkapuhelimella voidaan verkon kautta OTA-tekniikalla

nopeasti ladata matkakortti ilman, että pitää käydä erillisessä latauspisteessä (top-up). Tämä nopeuttaa palvelun käyttöä ja vähentää palveluntarjoajan kustannuksia, koska erillisen asiakaspalvelupisteen ylläpitäminen on kallista. [10, s. 2–3; 16, s. 3.]

Matkapuhelimella maksaminen tapahtuu helposti heilauttamalla puhelinta maksulaitteessa. Matkapuhelimen (U)SIM-kortille on tallennettu kaikki tilit, luotot ja ladatut rahat, joista käyttäjä voi helposti valita haluamansa maksuvälineen. Maksun suorittamiseksi tarvitaan jokin salasana. Tieto maksun onnistumisesta ja tilien saldo voidaan myös tarkistaa helposti jälkeenpäin. [10, s. 3.]

2.5 NFC-matkapuhelimet ja NFC-tunnisteet

NFC-matkapuhelimet

NFC-teknologian yleistymistä markkinoilla hidastaa NFC-matkapuhelimien vähäisyys. Nokialta ovat markkinoilla mallit 3220, 6131 NFC ja 6212 classic. Niistä mikään ei tue (U)SIM-korttia turvaelementtinä. 6131 NFC- ja 6212 classic -malleissa turvaelementti on sisäinen, ja 3220-mallia varten pitää ostaa erillinen matkapuhelinkuori, jossa turvaelementti ja NFC-siru ovat. Muilta valmistajilta matkapuhelimia on yhtä kehnosti. Nokian lisäksi ainakin Samsung, Benq, Motorola ja Sagem ovat esitelleet NFC-matkapuhelimia, mutta niistä harva on päässyt markkinoille. Jotta (U)SIM-korttia voidaan käyttää NFC:n turvaelementtinä, täytyy matkapuhelimen NFC-sirun tukea HCI:tä (host controller interface) ja SWP (single wire protocol) -protokollaa. SWP-protokollaa käytetään (U)SIM-kortin ja NFC-sirun väliseen kommunikointiin. [17.]

Matkapuhelinvalmistajat ovat kehitelleet matkapuhelimia, jotka tukevat SWP-protokollaa ja UICC-korttia turvaelementtinä. Niillä on tehty joitakin NFC-kokeiluja, mutta markkinoilla ne eivät ole. SWP-protokollaa tukevia matkapuhelinprototyyppejä ovat Sagem my700X-, LG L600V-, Nokia 6131 SWP- ja Motorola L7 (SLVR) -mallit. Kaikissa näissä malleissa käytetään Inside Contactless NFC -sirua. Teleoperaattorit vaativat

matkapuhelinvalmistajilta markkinoille nopeasti lisää matkapuhelimia, jotka tukevat SWP-protokollaa. [17.]

NFC-tunnisteet

NFC-tunnisteita on neljää eri tyyppiä, NFC Forum type 1/2/3/4 -tunnisteet, jotka on määrittänyt NFC Forum. Tunnisteen tyyppi määrittää sirun muistikapasiteetin ja käytetyn tiedonsiirtonopeuden. NFC Forum type 1 -tunnisteet ja NFC Forum type 2 -tunnisteet perustuvat ISO\IEC 14443A -standardiin. Type 1 -tunnisteilla on muistia 96 tavusta 2:een kilotavua, ja niissä käytetään Topaz-mikrosirua. Type 2 -tunnisteilla on muistia 48 tavusta 2:een kilotavua ja niissä käytetään Mifare Ultralight -mikrosirua. Molemmille tunnistetyypeille voidaan kirjoittaa ja uudelleenkirjoittaa, ja ne voidaan kirjoitussuojata. Niiden tiedonsiirtonopeus on 106 kilobittiä sekunnissa. [18; 19.]

NFC Forum type 3 -tunniste perustuu JIS (japanese industrial standard) X 6319-4 -standardiin, joka tunnetaan paremmin nimellä FeliCa. Myös käytettyä mikrosirua kutsutaan FeliCaksi. Type 3 -tunnisteet esiasettaa jo valmistaja kirjoitettaviksi, uudelleenkirjoitettaviksi tai pelkästään luettaviksi. Tunnisteen muistikapasiteetti vaihtelee, mutta teoreettinen kyky on 1 megatavu sovellusta kohden. Se pystyy siirtämään dataa 212 tai 424 kilobittiä sekunnissa. [18; 19.]

NFC Forum type 4 -tunniste on täysin yhteensopiva sekä ISO 14443A- että ISO 14443B -standardien kanssa. Myös Type 4 -tunnisteille valmistaja esiasettaa jonkin lukuoikeuden, kuten kirjoitus- tai uudelleenkirjoitusoikeuden tai pelkän lukuoikeuden. Tunnisteiden muistikapasiteetti vaihtelee, mutta se voi olla peräti 32 kilotavua sovellusta kohden. Sen tiedonsiirtonopeus on 424 kilobittiä sekunnissa. Type 4 -tunnisteet soveltuvat hyvin sellaiseen käyttöön, missä tarvitaan suurta muistikapasiteettia ja nopeaa tiedonsiirtoa. Mikrosiruna siinä käytetään usein Mifare DESfire -sirua. [18; 19.]

3 Älykortit

3.1 Älykorttien käyttö NFC-teknologiassa

Älykortit tarjoavat sekä loogisen että fyysisen turvallisuuden matkapuhelimilla käytettäville NFC-sovelluksille. NFC-sovellukset, kuten esimerkiksi maksusovellukset, sisältävät arkaluontoista informaatiota, joka pitää tallentaa siten, ettei kukaan ulkopuolinen pääse käsittelemään sitä. Älykortit tarjoavat turvallisen ympäristön, sillä niiden tekniikka mahdollistaa datan salauksen sekä kopioinnin ja väärentämisen eston. Lisäksi älykortit ovat paljon pitkäikäisempiä kuin useat matkapuhelimet ja ne ovat helposti siirrettävissä sovelluksineen matkapuhelimesta toiseen. Useat älykortit toteuttavat Global Platform -rajapinnan, joten niiden käsittely on älykortista riippumatta hyvin samanlaista. Älykortit ovat hyvin standardisoituja. Niille on standardit sovellusten säilyttämiseen, OTA-kommunikointiin, yksityisyyden turvaamiseen ja koko elinkaaren aikaiseen hallinnointiin. OTA-tekniikalla niiden sovellukset pystytään tarvittaessa välittömästi estämään tai aktivoimaan. NFC:n kannalta on myös tärkeää, että (U)SIM-kortin omistaa teleoperaattori, joka vastaa kaikista matkapuhelinpalveluista. (U)SIM-kortti on henkilökohtainen, mikä yhdessä älykorttitekniikan kanssa helpottaa käyttäjän tunnistamista ja hallinnointia. (U)SIM-korttia voidaan jopa joissain tilanteissa käyttää NFC:hen, vaikka matkapuhelimessa ei olisi akkua. [2, s. 15–16.]

Älykorttien muistikapasiteetti kasvaa jatkuvasti, ja älykorteille voidaan tallentaa yhä enemmän tietoa ja sovelluksia. Näin esimerkiksi telekommunikaatiossa käytettäville (U)SIM-korteille on järkevää sijoittaa enemmän sovelluksia matkapuhelimen sijaan. Kun kaikki käyttäjälle tärkeä on tallennettu (U)SIM-kortille, käyttäjä ei ole enää riippuvainen käytettävästä matkapuhelimesta.

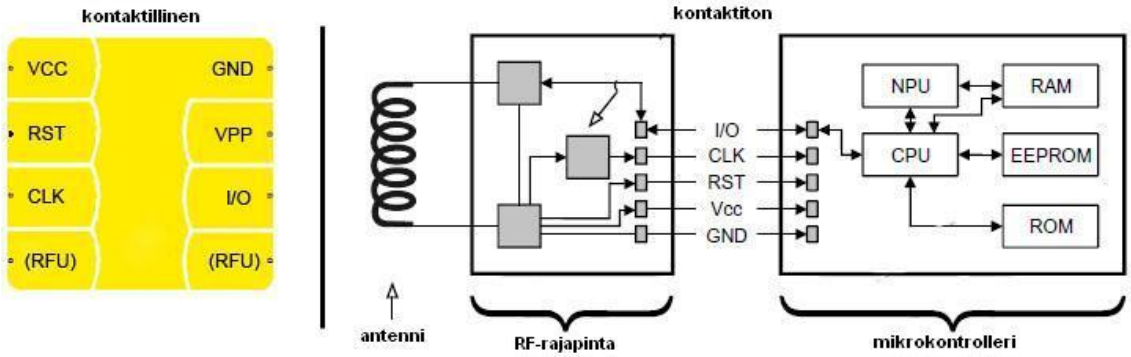
3.2 Älykorttien tekniikka

Älykortit ovat laitteita, joihin on upotettu joko mikrokontrolleri tai pelkkä muistisiru. Älykorteille, joissa on pelkkää muistia, voidaan vain turvallisesti tallentaa ja sieltä voidaan noutaa dataa. Vastoin nimen antamaa vaikutelmaa, niitä ei voida pitää todellisesti älykkäinä. Mikrokontrolleriälykorteissa on mikroprosessori ja muistia, ja ne voivat ajaa funktioita, kuten poistaa, lisätä ja manipuloida informaatiota. Niillä on käyttöjärjestelmä ja muistia sisäänrakennetuin turvallisuusominaisuuksin. Älykortit pystyvät kommunikoimaan älykortinlukijoiden kanssa joko kontaktillisesti tai kontaktittomasti. Älykortit täyttävät ISO/IEC 7816- ja/tai ISO/IEC 14443 -standardien eri vaatimukset. Nimi ”älykortti” on hieman harhaanjohtava, sillä älykortit eivät aina ole mukautettuna korteissa, vaan siru voidaan yhdistää moniin muihinkin asioihin. Älykortteja käytetään maksuvälinekorteissa, kulkukorteissa, matkapuhelimien UICC-kortteina, jotka yleisemmin tunnetaan (U)SIM-kortteina, avaimenperäavaimissa, kelloissa ja monissa muissa kohteissa. [20.]

Älykortit voivat olla kontaktittomia, kontaktillisia tai molempien yhdistelmiä eli hybridejä. Kontaktittomien korttien ei tarvitse koskea lukijalaitteeseen pystyäkseen kommunikoimaan sen kanssa. Kontaktittomat älykortit käyttävät lukijan muodostamaa sähkömagneettista kenttää tarvittavan virran saamiseen ja kommunikointiin. Kontaktillisen kortin taas pitää olla fyysisessä kosketuksessa lukijan kanssa. Kontaktillisen älykortin tunnistaa kortin päällä näkyvästä kultaisesta kontaktista. Kontaktittomia älykortteja pidetään luotettavampina kuin kontaktillisia, koska kontaktillisia kortteja haittaa kortilla oleva lika ja huolimaton kortin asettaminen lukijalaitteeseen voi vahingoittaa korttia. Kontaktilliset kortit pitää lisäksi asettaa täsmälleen oikeaan kohtaa, jotta kommunikointi lukijan kanssa voidaan aloittaa. Kontaktittomien korttien pitää vain olla sähkömagneettisen kentän ulottuvissa. Nykyään yhä useampi älykortti on kontaktiton. [21, s. 21.]

Kontaktillisen älykortin ja lukijan yhdistää älykortin kontaktirajapinta, jossa on 6–8 kontaktia, joiden paikat ja toiminnallisuus on tarkoin standardoitu. Se toimii myös suojakotelona sirulle. Kontaktittomassa älykortissa on radiotaajuusrajapinta, jonka kautta

sen kanssa voi kommunikoida. Radiotaajuusrajapinnan ja mikrokontrollerin välissä on melkein samanlainen rajapinta kuin kontaktillisen kontaktirajapinta. Koteloa kutsutaan sirumoduuliksi (chip module). Kuvasta 3 nähdään molempien, kontaktillisten ja kontaktittomien, älykorttien kontaktirajapinnat.



Kuva 3. Älykortin kontaktit [23; 22, s. 20].

Kontaktit tarjoavat tarvittavan virran (VCC), tyhjäys (RST)- ja kellotaajuussignaalit (CLK). Niissä on kaksisuuntainen rajapinta tiedonvaihtoon (I/O). Kuudes signaali (VPP) on tarkoitettu antamaan virtaa EEPROM-muistin ohjelmoinnille, mutta sitä harvoin käytetään. Kaksi viimeistä kontaktia (RFU) on jätetty mahdollista myöhempää käyttöä varten. [21, s. 20–21.]

Kontaktittomat kortit määrittävät ISO/IEC 14443 -standardin, joka on standardi lyhyen kantaman kontaktittomille älykorteille, jotka toimivat 13,56 megahertsin taajuudella. ISO/IEC 7816 -standardi on 15-osainen, ja sen kolme ensimäistä osaa määrittävät kontaktillisen älykortin olemuksen ja rajapinnat. Loput standardin osat koskevat eri älykorttityyppien ominaisuuksia. Niissä määritetään kortin looginen rakenne, komentoja kortin peruskäyttöön, sovellusten hallinnointi ja valitseminen, biometrinen varmennus, salauspalvelut ja sovellusten nimeäminen. Muita standardeja kontaktittomille älykorteille ovat ISO/IEC 10536 -standardi, joka on harvemmin käytetty standardi lyhyen kantaman kontaktittomaan kommunikointiin, ISO\IEC 15693 -standardi, joka on pitemmän matkan

kommunikointia varten kehitetty standardi, ja ISO\IEC 18092 -standardi, joka määrittää NFC-teknologian. [21, s. 25.]

Muita standardeja ja spesifikaatioita älykorteille ovat EMV 2000 -standardi, jonka ovat kehittäneet yhdessä Europay, Mastercard ja Visa. EMV 2000 -standardi on kehitetty maksukortteja varten. ETSI on myös määrittänyt telekommunikaatiossa käytettäville älykorteille monia tärkeitä standardeja ja spesifikaatioita, kuten GSM 11.11- ja TS 102 221 -standardit. GSM 11.11 -standardiperhe määrittää rajapinnan (U)SIM-kortin ja matkapuhelimen välille. Olennainen tekninen spesifikaatio UICC-korteille telekommunikaatiossa on ETSI 102 221 -spesifikaatio, joka luonnehtii niiden fyysiset ja loogiset parametrit. [22, s. 5; 22, s. 13; 22, s. 790.]

Mikrokontrolleriälykorttien tärkeimmät osat ovat mikroprosessori, ROM (read only memory)-, EEPROM (electronically erasable programmable read-only memory)- ja RAM (random access memory) -muistialueet sekä dataväylät ja rajapinta kommunikointiin ulkomaailman kanssa. Älykorteissa voi olla pääprosessorin lisäksi toinen prosessori, joka avustaa datan salauksessa. Nykyisten älykorttien prosessorit ovat yleensä 8–32-bittisiä. ROM-muisti on tarkoitettu käyttöjärjestelmää varten. Se on sama koko älykortin elinkaaren ajan, sinne ei pysty tallettamaan, eikä sieltä pysty poistamaan mitään kortin valmistuksen jälkeen.

Käyttöjärjestelmä valvoo, että sovellukset käyttävät vain niille määrätyn verran muistia. Samalle älykortille on mahdollista ladata useita sovelluksia, jotka toimivat täysin toisistaan riippumatta. Kaikki sovelluskohtainen data pysyy suojassa toisilta sovelluksilta. Sovellukset voidaan sijoittaa kortille valmistusvaiheen jälkeenkin käyttämällä OTA-tekniikoita. EEPROM-muistiin tallennetaan sovellukset ja muu informaatio. Sovelluksille tarkoitettu muisti on yleensä jaettu moniin sektoreihin ja alueisiin, joille on ominaiset avaimet, pääsyoikeudet ja kirjoitus- ja lukuoikeudet. Näiden avulla yhdistetään oikeutetut käyttäjät, laitteet ja sovellukset niille rajoitettuun muistialueeseen. Yleisimpien älykorttien muistikapasiteetti vaihtelee nykyään 16:n ja 256 kilotavun välillä. EEPROM-muistia

voidaan myös antaa käyttöjärjestelmälle. EEPROM-muistiin voidaan sen elinkaaren aikana kirjoittaa noin 100 000–1 000 000 kertaa. RAM-muisti on älykortin työskentelymuisti, se toimii käyttöjärjestelmän välimuistina. RAM-muisti ei ole pysyvää muistia, kuten ROM- ja EEPROM-muisti, vaan virran katketessa RAM-muisti tyhjenee. RAM-muistiin voidaan kirjoittaa loputtomasti. Flash EEPROM -muistia, josta usein käytetään termiä Flash-muisti, hyödynnetään myös älykorteilla. Sen avulla pystytään nostamaan älykorttien muistikapasiteettia huomattavasti. Sille voi kirjoittaa noin 10 000 kertaa. [22, s.70–79; 21, s. 19.]

Älykortit ovat luotettavia ja turvallisia sähköisiä tunnistevälineitä (electronic identification token). Ne ovat hyvin suojattuja, ja niitä on todella vaikea monistaa tai väärentää. Datansuojaus älykortilla on tärkeää, koska ne ovat yleensä personoitu yhtä käyttäjää varten, jolloin niille on tallennettu yksityistä informaatiota. Älykorttien päätarkoitus onkin oikeuksien todentaminen ja käyttäjän tunnistaminen. Niiden laitteisto ja ohjelmisto sisältävät monia tapoja havainnoida ja reagoida mahdollisiin informaation peukalointeihin. Ne on päällystetty ylimääräisillä metallilevyillä, ja niillä on sensorit, jotka havaitsevat lämpö- ja ultraviolettivalohyökkäykset, ja lisäohjelmistoa ja -mikropiiristöä estämässä virta-analyysit. [20.]

Älykortit tarjoavat varmennusta varten mekanismin, jota voidaan hyödyntää käyttäjiin, laitteisiin ja sovelluksiin, jotka tahtovat päästä käsittelemään älykortin dataa. Varmennuksen avulla voidaan varmistaa, että data on älykortilla turvassa ja sitä pääsevät käsittelemään vain oikeat osapuolet. Näin älykortille voidaan tallentaa yksityistä tietoa ulkoisten tietokantojen sijaan. Älykortit toimivat siis myös kannettavina tietokantoina. [20.]

Älykortit mahdollistavat datansalauksen. Niillä on salausominaisuuksia, kuten avaimien generointi, hajautus, digitaalinen allekirjoittaminen ja muistitila salausavaimille, joka näkyy vain älykortin sovelluksille. Salauksen avulla pystytään varmistamaan, että dataa ei ole peukaloitu. DES (data encryption standard) -algoritmi on perinteinen salausalgoritmi, jota käytetään telekommunikaatiossa ja rahatransaktiossa. DES-salaus on symmetrinen, eli siinä

käytetään samaa avainta salaukseen ja salauksen purkuun. Älykorttien mikrokontrollereissa on usein omat yksiköt DES-salauksen laskemista varten, mikä nopeuttaa selvästi prosessointiaikoja. Älykortit pystyvät myös RSA (Ron Rivest, Adi Shamir ja Leonard Adleman) -salausalgoritmiin, joka perustuu julkisiin ja yksityisiin salausavaimiin. RSA-salausalgoritmi on asymmetrinen. Salauksessa käytetään eri avainta datan salaukseen ja salauksen purkuun. RSA-salaus vaatii paljon laskutehoa, mutta on hyvin luotettava. [20; 22, s. 88.]

Lukija ja sovellus käyttävät APDU (application protocol data unit) -viestejä kommunikoidessaan keskenään. APDU-viestit määrittää ISO/IEC 7816-4 -standardi. APDU-viestit ovat kansainvälinen datayksikköstandardi sovelluskerroksella (application Layer). Niitä on kahdentyyppisiä: komento-APDU (command APDU) ja vastaus-APDU (response APDU). Komennot tulevat lukijalta älykortille, ja ne koostuvat ylätunnisteesta ja pääosasta. Ylätunniste on 5 tavua pitkä ja koostuu neljästä osasta: CLA (class byte) ja INS (instruction byte) sekä kaksi parametritavua (parameter bytes P1 & P2). Lisäksi sillä on vielä P3-tavu, jota voidaan käyttää lisänä määrittämään komennon pituus tai vastauksen pituus, muuten se merkitään nollaksi. Ylätunnisteen tarkoitus on kertoa sovellukselle komennon luonne. CLA-tavu kertoo komennon luokan. INS-tavu määrittää tarkan ohjeen kortille CLA-tavun puitteissa. ISO/IEC 7816-4 -standardi määrittää INS-tavulle 18 ohjetta. P1- ja P2-tavut antavat vielä lisämääritystä ohjeille. Jos INS-tavun määrittämä ohje ei niitä tarvitse, ne merkitään nollaksi.

Älykortit taas lähettävät vastaus-APDU:t lukijoille. Ne ovat yksinkertaisempia kuin komennot. Riippuen tulleesta komennosta siinä on pääosa, joka sisältää dataa tai voi olla määrittämätön. Vastauksen kaksi viimeistä tavua kertovat statusinformaatiota. Niistä toinen määrittää virhetilakategorian ja toinen määrittää komentokohtaisen statuksen tai virheilmoituksen. Virhetilakategoria kertoo, onnistuiko komento tai minkätyyppinen virhetila esiintyi. Seuraava tavu vielä tarkentaa esiintynyttä tilaa. Jos sovellus käyttää vain ISO/IEC 7816-4 -standardin määrittämiä ohjeita, sitä voidaan ajaa kaikentyyppisillä älykorteilla. Monesti kuitenkin jonkin muun standardin määrittämiä ohjeita käytetään

myös, kuten telekommunikaatiossa ETSI:n määrittämiä. Lukijalaitteen ja älykortin välisen kommunikaation lisäksi APDU-komentoja lähetetään myös OTA-tekniikalla, kuten CAT_TP:llä, taustajärjestelmästä älykortille. [21, s. 72.]

ISO/IEC 7816-4 -standardi määrittää älykorteille hierarkkisen tiedostonhallinta-järjestelmän. Tiedostonhallintajärjestelmällä on oliopohjainen rakenne, eli kaikki informaatio tiedostosta tallennetaan itse tiedostoon. Lisäksi tiedoston pitää ensin olla valittuna, ennen kuin voi suorittaa mitään operaatioita. Tällainen järjestelmä pakottaa jakamaan tiedostot kahteen osaan, ylätunnisteeseen ja pääosaan. Ylätunnisteeseen merkitään tiedoston rakenne ja sen pääsyoikeudet. Muokattava data taltioidaan pääosaan, ja pääosa on linkitetty ylätunnisteeseen osoittimella. Ylätunniste ja pääosa tallennetaan eri muistisivuille EEPROM-muistiin, ja tällöin ei voida tahallisesti aiheuttaa virhetiloja, joiden avulla pystyttäisiin muuttamaan ylätunnisteessa olevia pääsyoikeuksia pääosan informaatioon. Joskus käyttöjärjestelmä voi myös tehdä kaksi ylätunnistetta kahdelle eri sovellukselle, jolloin ne voivat jakaa tietoa. Tällöin ylätunnisteiden täytyy olla identtiset. [22, s. 254.]

Käytännössä älykorteilla voi olla kahdentyyppisiä tiedostoja: hakemistotiedostoja eli DF-tiedostoja (dedicated files) ja EF-tiedostoja (elementary files). DF-tiedostot toimivat kuin kansiot: niissä on viittaus alempiin DF-tiedostoihin tai EF-tiedostoihin. EF-tiedostoissa pidetään itse sovelluksen tai käyttäjän dataa. Kantatiedostoa (root) kutsutaan MF-tiedostoksi (master file). MF-tiedosto sisältää kaikki kansiot ja tiedostot älykortilla, ja se määrittää koko muistikapasiteetin tietoturva-alueelle. EF-tiedostot jaetaan vielä näkyvyyden perusteella ulkomaailmasta käsiteltäviin EF-tiedostoihin (working EFs) ja pelkästään käyttöjärjestelmälle tarkoitettuihin tiedostoihin (internal EFs). Working EF -tiedostoihin tallennetaan kaikki informaatio, joka on tarkoitettu ulkomaailmaan älykortista katsoen. Internal EF -tiedostoihin taas tallennetaan sovellusten ajamista koskeva data, salausavaimet ja data käyttöjärjestelmää varten. Pääsyä näihin tiedostoihin suojelee käyttöjärjestelmä. [22, s. 254–255.]

Jokaisella tiedostolla on kahden tavun suuruinen tunnistetieto (FID), jota käytetään tiedostojen valitsemiseen. Tunnisteiden täytyy olla samassa kansiossa uniikkeja, ja myös kansion tunnisteen täytyy olla uniikki. On muutamia tunnisteita, jotka on varattu tiettyyn käyttöön, eivätkä niitä voi muut tiedostot tai kansiot käyttää. [22, s. 258.] Taulukossa 1 on esitetty nämä tiedostotunnisteet.

Taulukko 1. Varatut tiedostotunnisteet [22, s. 258].

FID	**Nimi ja käyttö**	**Standardit**
’2F00’	Varattu EF_{DIR}-tiedostolle, johon tallennetaan sovellustunnisteita (application identifier) ja hakemistopolkuja kyseisiin sovelluksiin.	ISO/IEC 7816-4
’2F01	Varattu EF_{ATR}-tiedostolle, joka sisältää tarkenteita ATR:ään.	ISO/IEC 7816-4
’3F00’	Varattu MF-hakemistotiedostolle, joka on kaikkien älykortin tiedostojen kanta.	ISO/IEC 7816-4, GSM 11.11, TS 102.221, EMV
’3FFF’	Varattu tiedoston valitsemiselle, kun käytetään polun nimeä.	ISO/IEC 7816-4
’FFFF’	ISO/IEC on varannut tämän myöhempää käyttöä varten.	ISO/IEC 7816-4

EF_{DIR}-tiedoston tarkoitus on näyttää terminaalille kaikki älykortin sovellukset standardoidulla tavalla. DF-tiedostoilla on FID (file identifier) -tunnisteen lisäksi 1–16 tavun pituinen nimi (DF name), jota voidaan käyttää myös tiedoston valitsemiseen. Tämä on varatoimi, sillä on mahdollista, että uniikit kahden tavun FID-tunnisteet eivät riitä. DF-tiedoston nimeä käytetään aina ISO/IEC 7816-5 -standardin mukaisesti AID (application identifier) -tunnisteen kanssa, jolloin voidaan varmasti tietää, että ollaan yhteydessä oikeaan sovellukseen. AID-tunniste on jokaiselle älykorttisovellukselle 5–16 tavun pituinen uniikki tunniste. AID-tunniste koostuu RID (registered identifier) -tunnisteesta ja vapaaehtoisesta PIX (proprietary application identifier extension) -tunnisteesta. RID-tunniste on 5 tavun pituinen, ja se muodostetaan maakoodista, sovelluksen tyypistä ja numerosta, joka viittaa sovelluksen toimittajaan. RID-tunnisteen määrää kansallinen tai

kansainvälinen rekisteröintitaho, joten jokainen AID-tunniste on aidosti uniikki ja sitä voidaan käyttää sovelluksen tunnistamiseen maailmanlaajuisesti. ISO-organisaatiolta voi hakea uniikkia RID-koodia [23, s. 9]. PIX-tunniste on AID-tunnisteessa vapaaehtoinen, se voidaan koostaa esimerkiksi sarjanumerosta ja versionumerosta ja sillä voi olla pituutta 0–11 tavua. [22, s. 259–262.]

Älykorteille on useita eri käyttöjärjestelmiä. Käyttöjärjestelmän valitseminen riippuu pitkälti siitä, millaisia sovelluksia sillä tullaan käyttämään. Jos älykortilla on tarkoitus käyttää vain yhtä sovellusta, jota ei tarvitse muokata ja sovellus laitetaan sinne jo valmistajan toimesta, siinä ei tarvitse olla monimutkaista käyttöjärjestelmää. Useiden sovellusten älykorteilla, joihin on mahdollista ladata ja hallinnoida sovelluksia valmistajan toimituksen jälkeen, tarvitaan monimutkaisempi käyttöjärjestelmä, kuten Multos tai Java Card.

Multos-käyttöjärjestelmä on suunniteltu älykorteille, ja sen alusta noudattaa hyvin korkeaa turvallisuutta. Multos-kortille voidaan tallentaa useita sovelluksia ja dataa turvatulla ja kontrolloidulla tavalla. Sille voidaan ladata sovelluksia kortin toimittamisen jälkeenkin ilman riskiä, että ne voivat vahingoittaa kortin sisältöä. Multos-kortille on määritetty käyttöjärjestelmän lisäksi sille ominainen ohjelmointikieli MEL (multos executable language), tavat hallinnoida korttia ja ladata sovelluksia ja sertifikaatilla todentaminen. Sertifikaattien avulla älykortin toimittaja pystyy hallinnoimaan kolmansia osapuolia, jotka haluavat sijoittaa sovelluksiaan älykorteille. Vaikka Multos-kortille on oma ohjelmointikielensä, sen sovellukset yleensä kirjoitetaan C-kielellä. Kortin kääntäjä kääntää ne ajossa MEL-kieleksi. Multos-kortilla on oma virtuaalikone, joka hoitaa kääntämisen. [21, s. 91–92.]

Myös Microsoft on tuonut panoksensa älykorttiteknologiaan julkaisemalla .NET-älykorttikäyttöjärjestelmän, joka mahdollistaa Microsoftin .NET-kielillä kehitettyjen sovellusten ajamisen älykortilla. Tämä käyttöjärjestelmä ei ole kuitenkaan kovin suosittu. [21, s. 100–101.]

3.3 Global Platform -spesifikaatio

Global Platform on yhtymä, joka määrittää spesifikaatioita ja standardeja monen sovelluksen älykorteille. Sen tavoitteena on luoda ja ylläpitää spesifikaatioita ja ajaa standardien omaksumista, jotta saadaan avoin ja yhteentoimiva infrastruktuuri älykorteille, laitteille ja järjestelmille, mikä helpottaa ja nopeuttaa sovellusten kehitystä, järjestämistä ja hallinnointia eri toimialoilla. Global Platformin spesifikaatiot määrittävät prosessit sovellusten lataamiseen älykortille ja hallinnointiin siellä. Global Platform -yhtymä on jaettu kolmeen komiteaan teknisen toimialan mukaan: älykortti-, laite- ja järjestelmäkomiteoihin. Komiteat hoitavat oman teknisen toimialansa mukaiset spesifikaatiot. Global Platform -spesifikaatio mahdollistaa OTA-tekniikan käytön älykortilla, sovellusten lataamisen ja hallinnoinnin älykortilla. Global Platform -spesifikaatiot ovat alustasta ja sovelluksesta riippumattomia, joten niitä voidaan soveltaa kaikkiin käyttöjärjestelmiin ja älykortteihin. Tämän kirjoitushetkellä uusimman spesifikaation versionumero on älykorteille 2.2. [24.]

Kortin myöntäjille Global Platform tarjoaa vallan hallinnoida älykortteja joustavasti ja jakaa hallintaoikeutta joihinkin kortin osiin yhteistyökumppaneille. Valta säilyy kortin myöntäjällä, mutta hallintaoikeuden jaon kautta yhteistyökumppanit voivat ladata ja hallinnoida omia sovelluksa kortin myöntäjän älykortilla. Global Platform -spesifikaatio siis mahdollistaa sen, että teleoperaattorit voivat myöntää TSM:ille omia tietoturva-alueita (U)SIM-korteilta. [25, s. 17.] Liitteessä 3 on kuvattu Global Platformin määrittämä arkkitehtuuri, kun kortilla on kortinmyöntäjän sovellusten lisäksi muilta palveluntarjoajilta sovelluksia.

Global Platform määrittää älykorteille tietoturva-alueet (security domain), ja älykortit voivat sisältää niitä yhden tai useampia. Tietoturva-alueiden funktio on järjestää avaimet ja salauspalvelut sovelluksille. Jokaiselle tietoturva-alueelle on omat avaimet hallinnointia varten. Älykortin myöntäjä omistaa kortin ISD (issuers security domain) -tietoturva-alueen, (U)SIM-korteilla sen omistaa teleoperaattori. ISD isännöi teleoperaattorin sovelluksia,

kuten (U)SIM- ja (U)SAT (universal SIM application toolkit) -sovelluksia sekä RAM- ja RFM (remote file management) -hallintaohjelmia. ISD-tietoturva-alue on ainoa pakollinen tietoturva-alue älykortilla. Kortin myöntäjä pystyy luomaan toisia lisätietoturva-alueita (supplementary security domains), joille voidaan sijoittaa esimerkiksi NFC-sovelluksia. Lisätietoturva-alueet tulevat usein tarpeeseen tilanteissa, joissa älykortille ladataan eri organisaation kuin kortin myöntäjän sovellus, kuten esimerkiksi (U)SIM-kortille pankin maksusovellus. Kortin myöntäjä omistaa avaimet ISD-tietoturva-alueeseen ja OTA-hallinnointiin. Kun uusi tietoturva-alue luodaan jollekin taholle, se taho saa avaimet tietoturva-aluetta varten. Kaikki tietoturva-alueiden avaimet ovat toisistaan erillisiä. Tietoturva-alueet mahdollistavat sen, että monen sovelluksen älykortilla sovellukset voivat toimia täysin toisistaan riippumatta. [25, s. 19; 21, s. 84.]

Yksi Global Platformin keskeisimmistä komponenteista on älykorttihallintaohjelma (card manager). Se toimii älykortin ylläpitäjänä ja portinvartijana. Se hallitsee kortin toimittajan tietoturva-aluetta, valvoo elämänkaarta ja myöntää Global PIN -koodin sovelluksille. Se valvoo, mitä sovelluksia voi ladata kortille ja että ne käyttävät vain niille määrätyn verran muistia. Se tarkistaa jokaisen sisään tulevan komennon (APDU) ja toimittaa kommennot valitulle sovellukselle. Hallintaohjelma valvoo toimittajan avaimia, ja se alustaa suojatut yhteydet kortin ulkopuolelle. Kun älykortilla on useita tietoturva-alueita, niiden pitää jakaa vastuuta ja hallintaohjelman pitää aina tarkistaa tietoturva-alueen oikeudet, ennen kuin antaa sen suorittaa operaatioita. Kortin toimittaja määrää muiden tietoturva-alueiden oikeudet. [21, s. 81.]

Erityisen ohjelmistorajapinnan (global platform API) kautta älykortit voivat käyttää Global Platformin määrittämiä palveluita. Ohjelmistorajapinta yhdistetään Java Card -älykortin virtuaalikoneeseen, ja se voidaan myös yhdistää Multos- ja .Net-käyttöjärjestelmiin. Tämän rajapinnan avulla älykortti voi lukittautua, jos sen turvallisuus on uhattu. Se voi muodostaa suojatun kommunikaatiokanavan kortin ulkopuolelle, kuten CAT_TP-yhteyden ja se voi tarkistaa avaimet ennen niiden latausta älykortille. [21, s. 83.]

Global Platform -spesifikaatio määrittää älykortin, älykortin tietoturva-alueiden ja sovellusten elinkaaren ja elinkaaren aikaiset eri tilat. Älykortin elinkaari alkaa OP_READY-tilasta, jolloin kortilla on ajoympäristö ja ISD-tietoturva-alue pystyy käsittelemään sisään tulevia komentoja. Tässä tilassa voidaan asentaa lisätietoturva-alueita ja asettaa niille avaimia. Seuraava tila on INITIALIZED-tila. Tässä tilassa älykortille on jo laitettu jotakin alustavaa dataa, mutta se ei ole vielä valmis annettavaksi käyttäjälle. SECURED-tilassa ISD-tietoturva-alue sisältää kaikki tarvittavat avaimet ja turvallisuuselementit. Tässä vaiheessa kortti on täysin valmis käyttöön ja voidaan antaa käyttäjälle. Tämän jälkeen kortti voi elinkaarensa aikana enää mennä vain CARD_LOCKED-tilaan, jossa kortilta ei voi valita mitään sovelluksia tai tietoturva-alueita, tai TERMINATED-tilaan. Älykortin elinkaari loppuu TERMINATED-tilaan. CARD_LOCKED-tila on ainoa, josta voidaan palata edelliseen tilaan, muut tilamuutokset ovat peruuttamattomia. [25, s. 29–31.]

Spesifikaation mukaan sovelluksen elinkaari alkaa, kun sovellus on ladattu ja asennettu älykortille. Silloin sen tilaksi määritetään INSTALLED-tila. Tällöin sovellusta ei vielä voida valita kortin ulkopuolelta, mutta se on kuitenkin onnistuneesti taltioitu sille varattuun muistiin ja linkitetty käyttöjärjestelmään. Seuraava elinkaaren tila on SELECTABLE-tila. Tässä tilassa sovellus voi saada komentoja kortin ulkopuolelta. Ennen kuin sovellus voi mennä SELECTABLE-tilaan, sen pitää olla kunnolla asennettu älykortille ja täysin toimintavalmiudessa. Tästä mahdollinen seuraava tila on LOCKED-tila, jolloin sovellusta ei voi käyttää, mutta tästä tilasta voidaan palata takaisin edelliseen tilaan. LOCKED-tilaa hyödynnetään, kun esimerkiksi esiintyy turvallisuusongelmia. NFC-puhelimen kadotessa voidaan estää sovellukset (U)SIM-kortilla, jolloin sovelluksia ei voida käyttää, jos matkapuhelin päätyy vääriin käsiin. LOCKED-tila on ainoa, josta voidaan palata takaisin edelliseen tilaan, muut tilamuutokset ovat peruuttamattomia. Tietoturva-alueille on muuten samat elinkaaren tilat kuin sovelluksille, mutta niillä on vielä erikseen SELECTABLE-tilan jälkeen PERSONALIZED-tila, jossa niille annetaan tarvittavat avaimet ja data. Siinä varmistetaan, että tietoturva-alue on täysin käyttövalmiina. Molempien sovellusten ja tietoturva-alueiden elinkaari päättyy niiden fyysiseen poistoon älykortilta. [25, s. 33–37.]

Global Platform myös määrittää turvallisuusarkkitehtuurin, jonka päämääränä on taata älykortin komponenttien turvallisuus ja koskemattomuus, ja se takaa kortille tallennetun datan koskemattomuuden. Käytetty turvallisuuspolitiikka on täysin kortin myöntäjän vastuulla. Global Platform kuitenkin tarjoaa hyvin suunnitellut keinot suojata älykorttia ja sen sisältöä. [25, s. 22.]

3.4 Java Card -älykortti

Java Card -teknologia mahdollistaa, että älykortit ja muut muistin osalta rajoittuneet laitteet pystyvät ajamaan Java-sovelluksia. Java Card -alusta takaa turvallisen ympäristön, johon voidaan tallentaa useita sovelluksia, myös kortin toimittamisen jälkeen. Java-ohjelmointikieli tarjoaa useita etuja älykorttiteknologialle. Java on kehitetty alustasta riippumattomaksi ohjelmointikieleksi, ja se tunnetaan turvallisena ohjelmointikielenä. Java-ohjelmointikielen turvallisuusominaisuudet ja Java Card -älykortin arkkitehtuuri mahdollistavat usean sovelluksen rinnakkaiselon älykortilla. Älykortin JCRE (Java card runtime environment) -ajoympäristön turvallisuusominaisuudet pakottavat palomuurit erottamaan jokaisen sovelluksen toisistaan, millä varmistetaan, että yhden Java Appletin luomaa objektia ei voi käyttää toinen appletti. Java-ohjelmointikielen käyttö avaa älykorttien ympäristön yhä useammalle ohjelmistokehittäjälle. Jos hallitsee Java-ohjelmointikielen, voi periaatteessa kehittää sovelluksia älykorteille. Java Card -älykortit ovat päässeet teollisuuden suosioon älykortteina, joille tallennetaan useita sovelluksia. Java Card -älykortti on kaikkein käytetyin avoimen alustan älykortti. Java Card -teknologia noudattaa ISO/IEC 7816 -standardia. [26, s. 17–18.] Liitteessä 3 on kuva Java Card -älykortin arkkitehtuurista.

Java Card -teknologia määrittää Java Card -ajoympäristö JCRE ja luokkia ja funktiota helpottaakseen sovelluskehittäjiä kehittämään Java Appleteja. Java Appletit ovat sovelluksia, jotka on kirjoitettu Java-ohjelmointikielellä, ja niitä ajaa Java-virtuaalikone. Älykorteille kehitettyjä Java Appleteja voidaan myös kutsua Cardleteiksi. JCRE-ajoympäristö mahdollistaa Java Appletien ajamisen älykortilla, sen alustasta riippumatta.

JCRE sisältää älykortin laitteiston, käyttöjärjestelmän, JCVM (Java card virtual machine) -virtuaalikoneen ja Java-luokkakirjastot. Virtuaalikoneeseen ja luokkakirjastoihin on valmiit toteutukset, mutta laitteiston ja käyttöjärjestelmän älykorttivalmistaja joutuu toteuttamaan itse. Global Platform -spesifikaatio määrittää JCRE:lle asennusohjelman (installer), jonka avulla voidaan asentaa sovelluksia älykortille sen toimituksen jälkeen. Ilman asennusohjelmaa sovellukset pitää asentaa älykortille älykortin asennusvaiheessa. Tällaisia Java Card -älykortteja kutsutaan staattisiksi. [21, s. 78–79.]

JCVM-virtuaalikoneella on kaikki tarvittava tietämys ja resurssit Java-tavukoodin ajoon älykortilla. Se eroaa perinteisestä Java-virtuaalikoneesta siten, että se on jaettu kahteen osaan. Älykortin virtuaalikone sisältää vain ne toiminnot, jotka ovat relevantteja sovellusten ajonaikaiseen ajamiseen. Muut toiminnot, kuten luokkien lataus, linkitys ja tavukoodin tarkistus, tekee muunninohjelma (converter), jota ajetaan tietokoneella. Tietokoneille löytyy myös JCVM-emulaattoreita, joiden avulla voidaan testata sovellusta, ennen kuin se asennetaan älykortille, jolloin älykorttien muistia, jonka käyttökerrat ovat rajalliset, ei tarvitse turhaan rasittaa. JCVM-virtuaalikoneen tehtävä on ajaa Java-tavukoodi ja valvoa pääsyä kaikkiin älykortin resursseihin, kuten muistiin ja sisääntulo- ja ulostuloportteihin. Lisäksi se sallii sovellusten lataamisen älykortille sen toimituksen jälkeen, mikäli JCRE:ssä tämä on toteutettu. Java Appletit kehitetään tietokoneella, minkä jälkeen ne käännetään ja muunnetaan CAP (converted applet) -tiedostoiksi. Java Appletti asennetaan CAP-tiedostona älykortille, joka on virtuaalikoneen ajettavassa muodossa. Tässä prosessissa saadaan kitkettyä mahdollisia sovelluskehityksessä esiintyneitä virheitä pois sovelluksesta, ennen kuin se asennetaan älykortille, ja älykortilla oleva virtuaalikone voidaan pitää mahdollisimman pienenä, kun sovelluksen kääntäminen voidaan tehdä kortin ulkopuolella. [21, s. 78; 26, s. 18.]

Java Card -älykortille on neljä ohjelmistorajapintaa, jotka helpottavat sovellusten kehitystä älykorteille. Kolme niistä on pakollisia kaikille älykorteille, joissa tietyt salausalgoritmit eivät ole välttämättömiä. Javan perustan luo java.lang-paketti, joka sisältää kaikki perinteiset luokat virhetiloille. Sitä täydentää javacard.framework-paketti, joka sisältää

ydinluokat Java Appletille älykortilla. Se sisältää perusluokat sovellusten hallintaan ja tiedonvaihtoon terminaalin kanssa ja monenlaisia ISO/IEC 7816-4 -standardin määrittämiä vakioita. Tietoturvaa edistää javacard.security-paketti, jossa on funktioita ja rajapintoja useisiin salausalgoritmeihin. Vaihtoehtoinen javacardx.crypto-paketti sisältää rajapinnat salausten purkuun. Lisäksi Java Card -älykorteille voidaan ladata lisäohjelmistorajapintoja, kuten SIM Application Toolkit. [26, s. 18.]

3.5 SIM Application Toolkit -rajapinta

Kaikki GSM-matkapuhelimet käyttävät USIM- tai SIM-sovellusta, joka on matkapuhelimen UICC-kortilla. (U)SIM-sovelluksen rooli on toimia teleoperaattorin turvamoduulina matkapuhelimessa, ja se vahvistaa teleoperaattorin verkon käyttöoikeuden. Joka kerta, kun matkapuhelimella yritetään käyttää matkapuhelin- tai tietoverkkoa, teleoperaattorin varmennuskeskus pyytää (U)SIM-sovellusta varmentamaan käyttöoikeutensa. Ero USIM-sovelluksen ja SIM-sovelluksen välillä on se, että USIM-sovellusta käytetään 3G-puhelimissa. [21, s. 131.]

SIM Application Toolkit (SAT) tarjoaa rajapinnan (U)SIM-kortin ja matkapuhelimen välille. Näin (U)SIM-sovellus voi hyödyntää matkapuhelimen näyttöä, näppäimistöä ja kommunikaatioyhteyksiä. Se sallii myös datan ja sovellusten latauksen (U)SIM-kortille. SAT-rajapinnan avulla voidaan antaa lisäarvoa sovelluksista käyttäjälle, kun sovellus voi hyödyntää näitä matkapuhelimen ominaisuuksia. Esimerkiksi matkapuhelimella tehdyn maksun jälkeen voidaan tarkistaa oma saldo. USAT:a käytetään USIM-korteilla, joissa on 3G-tuki. SAT:n on kehittänyt ETSI-järjestö vuonna 1999. Lisäksi on kehitetty Card Application Toolkit (CAT), joka on SAT:n ylästandardi. Siitä on riisuttu kaikki GSM-palveluihin liittyvä pois, joten se on käytettävästä laitteesta riippumaton. CAT:n määrittää ETSI TS 102 223 -spesifikaatio. [21, s. 132.]

SIM Application Toolkit antaa (U)SIM-sovelluksille mahdollisuuden tehdä monia toimintoja. Kaikki matkapuhelimet eivät kuitenkaan välttämättä tue näitä kaikkia

toimintoja, joten (U)SIM:n tulee jotenkin tietää, mihin kaikkeen matkapuhelin kykenee. Tämän tiedon se saa SAT:n Profile download -ohjeella. Tämän ohjeen matkapuhelin lähettää (U)SIM:lle sen käyttöönottovaiheessa. Proactive SIM on mekanismi, jolla (U)SIM voi aloittaa jonkin tapahtuman matkapuhelimessa. Proactive-komentoja on kymmeniä, ja niiden avulla pystytään esimerkiksi aloittamaan puheluita, soittamaan soittoääniä, avaamaan selain, avaamaan yhteyskanava, lähettämään dataa, pyytämään käyttäjäsyötettä ja laukaisemaan monia muita tapahtumia. Komennon jälkeen matkapuhelin kertoo (U)SIM:lle sen onnistumisesta. Jatko on täysin (U)SIM:stä kiinni. Se on vastuussa tapahtumista, jotka lähtevät (U)SIM:ltä. Jos käsky epäonnistuu, matkapuhelin palauttaa virheen syyn (U)SIM:lle. [27, s. 13; 27, s. 15–17.]

SAT:n avulla voidaan saada matkapuhelimeen tulleesta tekstiviestistä dataa ilman, että matkapuhelin hälyttää käyttäjälle tekstiviestistä, eikä viesti mene matkapuhelimen saapuneiden viestien laatikkoon eli se on läpinäkyvä käyttäjälle. Tämän mahdollistaa SMS-PP data download -palvelu. Jos palvelu on aktivoitu, matkapuhelin tarkistaa sisään tulevien tekstiviestien sisällön. Jos viesti sisältää SIM data download -protokollatunnisteen, matkapuhelin antaa viestin (U)SIM:lle käsiteltäväksi. CAT_TP-tiedonsiirtoprotokollan kannalta SMS-PP data download -palvelu on tärkeä, koska yleensä taustajärjestelmä pyytää(U)SIM-korttia aloittamaan CAT_TP-yhteyden. Tämän se tekee lähettämällä SMS-PP-viestin (U)SIM-kortille, jossa on komento CAT_TP-yhteyden avaamisesta tiettyyn porttiin. [27, s. 41.]

BIP-kanavan avaus on määritetty SAT-spesifikaatiossa. (U)SIM voi pyytää matkapuhelinta avaamaan GPRS- tai 3G-yhteyden johonkin määrättyyn porttiin. Käytetty yhteys valitaan open channel -komennon yhteydessä. SAT-spesifikaatio määrittää komennot (U)SIM:lle kanavan avaukseen ja sulkemiseen, datan lähettämiseen ja vastaanotetun datan saamiseen ja yhteyskanavan statuksen saamiseen. BIP-kanavan kautta (U)SIM-kortti voi kommunikoida taustajärjestelmän kanssa. BIP-protokollaa käytetään OTA-tekniikassa esimerkiksi sovellusten lataamiseen ja asentamiseen (U)SIM-kortille. [28, s. 20.]

SAT määrittää myös keinot monitoroida matkapuhelimen tapahtumia, kuten soittoja, tekstiviestejä, sijaintia ja useita muita. Se mahdollistaa myös puheluiden ja tekstiviestien kontrolloinnin. SAT määrittää myös turvallisuuden kaikille SAT-spesifikaation määrittämille tapahtumille. [27, s. 12.]

3.6 Älykortin sovellukset

Älykortit ovat olleet käytössä jo yli 30 vuotta, ja ajan kuluessa niille on keksitty yhä enemmän käyttökohteita. Ensimmäinen älykortin käyttökohde olivat ennakolta maksetut puhelinkortit (prepaid). Niissä käytetyt älykortit sisälsivät vain pelkkää muistia, joten ne eivät pystyneet prosessoimaan dataa. Nykyään useimmat älykortit sisältävät mikrokontrollerin ja niitä käytetään hyvin laajasti. Matkapuhelimien, luotto- ja pankkikorttien lisäksi älykortteja käytetään tunnistamiseen ja varmentamiseen monilla eri aloilla, terveystietojen ja -vakuutusten sekä muun henkilökohtaisen tiedon tallentamiseen, maksu-TV:hen, pääsy- ja matkalippuina, asiakasuskollisuuteen (bonuskortit), kirjastokortteina ja moniin muihin tarkoituksiin.

Älykortin looginen ja fyysinen turvallisuus takaavat hyvän ympäristön turvallisuutta vaativille sovelluksille. Älykortin käytön myötä voidaan ladata yhä enemmän informaatiota käyttäjästä älykortille, ja voidaan olla varmoja, että informaatio kortilla ei ole väärennettyä. Siksi älykorttien käyttöä harkitaan yhä useammalla toimialalla käyttäjien henkilökohtaisen tiedon tallentamiseen. Älykortit mitä luultavimmin syrjäyttävät magneettiraitakortit markkinoilla.

Nykyään, kun älykorttien muistikapasiteetti on kasvanut ja prosessorit parantuneet, on järkevää sijoittaa useita sovelluksia samaan älykorttiin. Matkapuhelimen UICC-kortti tuntuu täydelliseltä ratkaisulta siihen, koska on nykypäivää, että melkein kaikilla ihmisillä on matkapuhelin aina mukana kulkiessaan. Käyttäjää voi kuitenkin pelottaa ajatus, että jos hän hukkaa (U)SIM-kortin, hän hukkaa kaikki pankki-, luotto- ja matkakortit samalla. Vaikka (U)SIM-kortin sovellukset pystytäänkin sulkemaan välittömästi, käyttäjä on silti

menettänyt kaiken, kunnes kortti löytyy tai uusi myönnetään. Kun kaikki kortit ovat erillään, hukkaustilanteessa ei välttämättä kadota kaikkea.

4 Over-the-air-tekniikka

4.1 OTA-tekniikka

OTA (over the air) -tekniikka tekee mahdolliseksi luoda linkin taustajärjestelmän ja (U)SIM-kortin välille. Tällainen linkki mahdollistaa komennon lähettämisen suoraan taustajärjestelmästä näkymättömästi matkapuhelimen (U)SIM-kortille. OTA-tekniikka mahdollistaa sovellusten ja datan lataamisen, poistamisen ja muokkaamisen, ja sen avulla sovellukset voidaan aktivoida tai estää (U)SIM-kortilla. OTA-tekniikka on myös perusta NFC-sovellusten lisäarvolle, sillä sen avulla voidaan vaihtaa tietoa palveluntarjoajan taustajärjestelmän ja asiakkaan (U)SIM-kortin välillä. Älykorttien suurempi muistikapasiteetti sallii ladata useampia sovelluksia ja enemmän henkilökohtaista tietoa älykorteille.

OTA-tekniikan avulla älykorttien hallinta on huomattavasti parantunut. Se on edullisempaa palveluntarjoajan kannalta ja helpompaa käyttäjän, sillä erillisiä palvelupisteitä ei tarvita ollenkaan. Esimerkiksi ladatakseen lisää rahaa joukkoliikenne-lippusovellukseen käyttäjän ei tarvitse lähteä johonkin palvelupisteeseen tekemään tätä, vaan hän voi tehdä sen suoraan siellä, missä on, ja mihin kellonaikaan vain. Asiakaspalvelupisteiden ylläpito on palveluntarjoajille kallista. OTA-tekniikan ansiosta (U)SIM-kortit eivät enää ole staattisia komponentteja, jotka eivät kehity ajan myötä. Nyt niille voidaan ladata enemmän informaatiota ja ne ovat arvokkaampia käyttäjälle. Niitä ei tarvitse vaihtaa tehtäessä pieniä päivityksiä korteille. OTA- ja NFC-teknologia lisäävät matkapuhelinten ja tietoverkon käyttöä entisestään, mikä tuo lisätuloja teleoperaattorille, mutta tietenkin suurempia matkapuhelinlaskuja käyttäjälle. [29, s. 4.]

OTA-tekniikka perustuu asiakas-palvelinmalliin. Sen toisessa päässä on palvelin, joka kuuluu jollekin palveluntarjoajalle, teleoperaattorille, TSM:lle tai vastaavalle. Palvelimen pitää olla yhteydessä matkapuhelinverkkoon joko LAN:n tai Internetin välityksellä. Yhteyden toisessa päässä on (U)SIM-kortti, joka käyttää matkapuhelinta terminaalina. OTA-teknologian keskeiset komponentit ovat taustajärjestelmä, joka lähettää komennot (U)SIM-kortille, OTA-yhdyskäytävä, tekstiviestikeskus ja laite, joka toimii (U)SIM-kortin terminaalina. OTA-yhdyskäytävä muuntaa taustajärjestelmän komennot (U)SIM-kortille luettavaan muotoon. OTA-yhdyskäytävä on yhteydessä SIM-korttitietokantaan; se generoi komennon vastaanottavalle (U)SIM-kortille ymmärrettäväksi (U)SIM-kortin tyypin mukaisesti. Lisäksi tarvitaan SMSC (short message service center) -tekstiviestikeskus, joka pystyy lähettämään OTA-yhdyskäytävältä saadun datan SMS-viesteinä (U)SIM-kortille. Myös muita tiedonsiirtoyhteyksiä voidaan tarvita, kuten TCP (transmission control protocol)- ja UDP (user datagram protocol) -yhteyttä. Komentojen vastaanottoon tarvitaan matkapuhelin, joka toimittaa komennot (U)SIM-kortille. Matkapuhelimessa pitää olla SAT asennettuna. [29, s. 5–6.] Liitteessä 5 näkyy kuvana OTA-arkkitehtuuri.

Kommunikaatio taustajärjestelmän ja (U)SIM-kortin välillä voidaan tehdä useammalla tavalla: SMS-viesteillä tai BIP-protokollalla. Perinteisin tapa on käyttää SMS-viestejä. Siinä taustajärjestelmä lähettää datan OTA-yhdysväylää käyttäen. Yhdysväylä muuntaa datan SMS-viesteiksi ja antaa ne tekstiviestikeskukselle, joka puolestaan lähettää ne (U)SIM-kortille. SMS-viesteillä on tietty yläpituusraja, 160 merkkiä, joten SMS-lähestymistavassa data pitää jakaa useampiin viesteihin, jos se ylittää tuon rajan. Tämän lähestymistavan ongelma on se, että jos sovelluksen lähetyksessä yksi tai useampi viesti katoaa, täytyy koko sovellus lähettää uudelleen (U)SIM-kortille. Datan lähettäminen SMS-viestein on myös hidasta. [30, s. 17.]

Toinen OTA-tekniikan lähestymistapa on BIP-protokolla. BIP-protokollan avulla dataa voidaan lähettää nopeammin ja tehokkaammin kuin SMS-viestejä käyttäen. BIP-protokollan avulla (U)SIM-kortti voi käyttää kommunikointiin matkapuhelimen GPRS-, 3G-, EDGE-, Bluetooth-, IrDA- tai Wi-Fi-yhteyttä. Se on tiedonvälityskanavasta ja

sovelluksista täysin riippumaton. BIP on kuitenkin suunniteltu ajatellen IP (internet protocol) -pohjaisia yhteyksiä, kuten EDGE-, GPRS- ja 3G-yhteydet. BIP-protokollaa varten on kaksi erilaista toteutusta: CAT_TP- ja TCP-tiedonsiirtoprotokollat. BIP-protokollaa käyttäen data voidaan lähettää taustajärjestelmästä (U)SIM-kortille suurempina osina kuin käyttäen SMS-viestejä. Oikeastaan kerralla lähetetyn datan suuruuden määrittää täysin (U)SIM-kortin kyky käsitellä sitä. BIP-protokollan tiedonvälityskanava antaa reaaliaikaisen varmennuksen, joka kertoo lähetetyn datan menneen perille. Jos yhden tai useamman paketin lähetys epäonnistuu, paketit voidaan lähettää heti uudelleen ilman, että jo toimitettuja paketteja tarvitsisi lähettää uudestaan. Markkinoilla on kuitenkin vain rajoitettu määrä matkapuhelimia, jotka mahdollistavat BIP-protokollan käytön. [30, s. 5–6.]

OTA-tekniikan tiedonvälityskanavan valitseminen määrittyy pitkälti vastaanottavan matkapuhelimen ominaisuuksista. SMS-viestejä käytetään OTA-tekniikassa tilanteissa, joissa matkapuhelin ei tue muita tiedonvälityskanavia, kuten GPRS- tai 3G-yhteyttä. GPRS-yhteyden käyttö saattaa nostaa suorituskykyä 100-kertaiseksi SMS-viesteihin nähden. 3G-yhteys parantaa suorituskykyä vielä entisestään GPRS-yhteyteen nähden. [30, s. 9.]

RFM on älykortin tiedostojen hallinnointia OTA-tekniikalla. RFM:llä pystyy käsittelemään kaikkia MF-tiedostossa viitattuja DF- ja EF-tiedostoja. ETSI TS 102 221- ja ETSI TS 102 222 -standardit määrittävät komennot tiedostojen hallinnointiin OTA-tekniikalla. Liitteessä 6 on esitetty RFM-komennot. RAM taas on älykortilla olevien sovellusten hallintaa OTA-tekniikalla. RAM sisältää tavat sovelluksien asennustiedostojen (load file) lataamiseen ja poistamiseen sekä sovellusten lataamiseen, asentamiseen ja poistamiseen. Lisäksi sen avulla voi lukita ja avata sovelluksia. Global Platform -spesifikaatio määrittää prosessit kaikille edellä mainituille. Jotta jonkin tietoturva-alueen tiedostoja ja sovelluksia pystyy muokkaamaan, pitää olla tarvittavat oikeudet myönnettynä. [31, s. 14–15.] Komennot sovellusten hallinnointiin on esitetty liitteessä 7.

OTA-hallinnointi voidaan jakaa kolmeen pääalueeseen. Ensimmäisenä on (U)SIM- ja (U)SAT-sovellusten hallinnointi (U)SIM-kortilla. Tämäntyyppistä hallinnointia tehdään kaikille (U)SIM-korteille, niillekin, joita ei käytetä NFC-teknologiaan. Sille tyypillistä on (U)SIM-tiedostojärjestelmän päivittäminen tai sen puhelinluetteloon tallentaminen. Sitä voidaan myös hyödyntää (U)SIM Application Toolkitin lataamiseen kortille. Toinen pääalue on TSM:n OTA-hallinnointi palveluntarjoajien sovelluksille. TSM turvallisesti lataa ja personoi sovelluksen sille myönnettyyn tietoturva-alueeseen ja hallinnoi niitä siellä koko niiden elinkaaren ajan. Kolmas pääalue on tietoturva-alueiden muistikiintiöiden hallinnointi OTA-tekniikalla, minkä tekee yleensä teleoperaattori, mutta joskus sekin saattaa ulkoistaa sen TSM:lle.

4.2 BIP-protokolla

BIP-protokolla sallii tiedon latauksen (U)SIM-kortille matkapuhelimen GPRS-, EDGE- tai 3G-yhteyden kautta, mutta myös muita yhteyskanavia voidaan käyttää, sillä BIP on tiedonvälityskanavasta riippumaton. Sen määrittä ETSI TS 102 223 -standardi. RAM- ja RFM-palvelut ovat huomattavasti nopeampia BIP-kanavan läpi kuin SMS-viestien käyttö, ja siksi BIP-kanava soveltuu hyvin älykortin hallinnointiin, kuten lataamiseen, päivittämiseen, estämiseen, aktivoimiseen ja poistamiseen. BIP-protokolla itsessään ei takaa turvallisuutta tai varmuutta lähetetylle datalle, joten niiden pitää olla kunnossa käytetyllä tiedonvälityskanavalla. [30, s. 17.]

Käytännössä BIP on joukko SAT-komentoja, jotka määrittävät rajapinnan matkapuhelimen ja (U)SIM-kortin välille. Komentoihin kuuluvat halutun yhteyskanavan avaaminen ja sulkeminen, datan lähetys ja vastaanotto sekä yhteyden statuksen saaminen. Lisäksi, jos tiedonvälityskanavana käytetään Bluetooth- tai IrDA-yhteyttä, niille on vielä muutama lisäkomento yhteyksien alustamiseen. Yhteyden avaamiseen käytetään OPEN_CHANNEL-komentoa, jossa pitää määrittää käytettävä yhteyskanava, osoite ja puskurin koko eli datan maksimikoko, jonka (U)SIM-kortti pystyy käsittelemään. Taulukossa 2 on esitetty BIP-protokollan käyttämät SAT-kommennot.

Taulukko 2. BIP-komennot [32, s. 101].

BIP-komento	**Kuvaus**
OPEN CHANNEL	Avaa yhteyskanavan määrättyyn osoitteeseen.
CLOSE CHANNEL	Sulkee yhteyskanavan.
RECEIVE DATA	Hakee matkapuhelimeen vastaanotetun datan (U)SIM-kortille.
SEND DATA	Lähettää dataa, kun yhteyskanava on avattuna.
GET CHANNEL STATUS	Palauttaa yhteyskanavan statuksen.
SERVICE SEARCH	Etsii palvelua matkapuhelimen lähiympäristöstä (*).
GET SERVICE INFORMATION	Palauttaa kaikki tiedot tietystä palvelusta (*).
DECLARE SERVICE	Lataa puhelimen palvelulistaan ne palvelut, jotka (U)SIM-kortti tarjoaa palvelimena (*).
*) Vain palveluille, kuten Bluetooth tai IrDA	

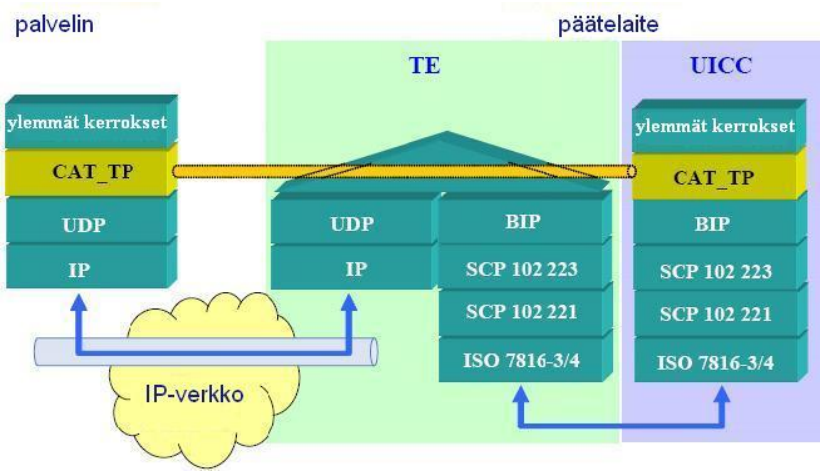
Kuten mainittu, BIP-protokolla käyttää kahta toteutusta: CAT_TP ja TCP. CAT_TP-protokolla on suunniteltu(U) SIM-kortin ja palvelimen väliseen kommunikointiin. TCP-protokolla taas on ollut jo kauan käytössä ja tunnetaan laajasti Internet-protokollana. Matkapuhelimissa TCP-protokollaa on käytetty pitkään jo WAP-selaimen ja -palvelimen väliseen kommunikointiin. BIP-protokollassa hyödynnetään TCP:n luotettavaa yhteyttä. Suurin ero TCP:n ja CAT_TP:n välillä on, että TCP-protokolla on toteutettu matkapuhelimeen, kun taas CAT_TP-toteutus on (U)SIM-kortilla. TCP-protokolla on ollut kauan käytössä Internet-protokollana, joten sille löytyy palvelimelle monia valmiita toteutuksia. CAT_TP-toteuksen joutuu todennäköisemmin tekemään alusta asti itse, jos sitä ei ole vielä palvelimelle toteutettu. UDP-protokollan yläpuolelle toteutettu CAT_TP-protokolla tarjoaa kuitenkin nopeamman yhteyskanavan kuin TCP-protokolla ja siinä on yhtäläinen varmistus pakettien perille pääsystä. [30, s. 6.]

BIP-protokolla mahdollistaa sen, että kommunikaation molemmat osapuolet, taustajärjestelmä ja (U)SIM-kortti, voivat aloittaa kommunikaation, riippuen tietenkin (U)SIM-kortilla olevasta sovelluksesta. Sovellusaloitteinen palvelukutsu menee niin sanotun pull-mekanismin mukaisesti ja palvelinaloitteinen push-mekanismin. (U)SIM-kortilla oleva sovellus, johon on toteutettu SAT:n mukaisesti valikko, jossa on vaihtoehto

tiedonhakuun taustajärjestelmästä, voi aloittaa käyttäjän toimesta yhteyden taustajärjestelmän kanssa. Tällöin (U)SIM-kortti pyyttää matkapuhelinta OPEN_CHANNEL-kutsulla avaamaan yhteyskanavan taustajärjestelmään ja kanavan avattua lähettää komennon taustajärjestelmälle. Push-mekanismissa palvelin tekee aloitteen kommunikaation aloittamiseksi. Se ei kuitenkaan tapahdu aivan niin yksinkertaisesti kuin sovellusaloitteinen yhteyskanava, koska matkapuhelin ei ole yhteydessä tietoverkkoihin jatkuvasti. Kun taustajärjestelmä tahtoo avata kommunikaatiokanavan (U)SIM-kortin ja sen välille, se lähettää (U)SIM-kortille PushSMS-tekstiviestin, jossa se pyytää (U)SIM-korttia avaamaan yhteyden määrättyyn osoitteeseen ja siellä tiettyyn porttiin. SAT:n toimesta käyttäjä ei huomaa PushSMS-viestin saapumista matkapuhelimeen eli se tulee käyttäjälle näkymättömästi. Saatuaan viestin (U)SIM-kortti avaa yhteyden taustajärjestelmään ja taustajärjestelmä voi suorittaa halutut operaatiot. [30, s. 14–16.]

4.3 CAT_TP-tiedonsiirtoprotokolla

CAT_TP-tiedonsiirtoprotokolla mahdollistaa (U)SIM-kortin ja palvelimen välisen kommunikoinnin ilman, että ne ovat fyysisesti kiinni toisissaan. CAT_TP-protokolla käyttää tähän matkapuhelimen GPRS- tai 3G-yhteyksiä. Se on suunniteltu varta vasten juuri (U)SIM-kortin ja palvelimen väliseen kommunikointiin. CAT_TP-protokollan määrittää ETSI TS 102 127 -spesifikaatio. Tiedonvaihto palvelimen ja (U)SIM-kortin välillä tapahtuu UDP-porttien kautta. UDP-protokolla ei itsessään takaa tiedon eheyttä, ja datapaketit voivat kadota tai tulla väärässä järjestyksessä ilman huomautusta. Tämän takia on kehitetty CAT_TP-protokolla, joka hoitaa pakettien kuittauksen, segmentoinnin, järjestämisen, uudelleen lähettämisen ja niin edelleen. CAT_TP:n pitää olla toteutettu kommunikaation molemmissa päissä, (U)SIM-kortilla ja palvelimella. [30, s. 6.] Kuvassa 4 on esitetty CAT_TP-ympäristö ja sen eri kerrokset. Taustajärjestelmä (remote entity) antaa komentoja CAT_TP-protokollan välityksellä älykortille. CAT_TP-protokolla käyttää BIP-protokollan mukaisesti matkapuhelimen (TE) 3G- tai GPRS-yhteyttä. Tieto siirtyy kommunikaation osapuolten välillä UDP-porteista.



Kuva 4. CAT_TP-ympäristö ja sen kerrokset [33, s. 10].

CAT_TP-yhteys on kaksisuuntainen (full-duplex), joten se sallii tiedon lähettämisen ja vastaanottamisen yhtäaikaisesti. Siinä tieto lähetetään luotettavasti kommunikaation toiselle osapuolelle, ja jos ongelmia esiintyy, ne ilmoitetaan ylemmälle kerrokselle. CAT_TP-protokolla huomaa kaikki virheelliset PDU (protocol data unit) -paketit ja hylkää ne. Lähettäessään dataa CAT_TP segmentoi suuret SDU (service data unit) -paketit pienemmiksi PDU-paketeiksi, ja vastaanottaessaan se puolestaan kokoaa PDU-paketeissa saadun datan takaisin SDU-paketiksi. PDU-paketit järjestetään niille annetun sekvenssinumeron (sequence number) mukaisesti. Näin saadaan koottua SDU-paketti oikein, vaikka PDU-paketit saapuisivat väärässä järjestyksessä vastaanottavalle osapuolelle. [33, s. 9–10.]

SDU-paketti sisältää älykortin komennot eli APDU:t. CAT_TP ei ole kiinnostunut SDU-paketin sisällöstä, se vain segmentoi sen tarvittaessa CAT_TP-yhteydessä määritettyjen rajojen mukaisesti PDU-paketeiksi. Data liikkuu CAT_TP-yhteyden läpi PDU-paketeissa, jotka on koostettu ylätunnisteesta ja datasta. Aina yhteyden luomisen aikana kommunikaation osapuolet ilmoittavat toisilleen suurimman mahdollisen PDU- ja SDU-paketin koon. Jos älykortille lähetettävän komennon pituus ylittää PDU-paketin maksimikokorajan, se segmentoidaan useampiin PDU-paketteihin. PDU-paketit lähetetään mahdollisimman pian niiden luomisen jälkeen vastaanottavalle osapuolelle. Lähettäjä

säilyttää lähetetyt PDU-paketit, kunnes toinen osapuoli kuittaa ne. Vastaanottavalla puolella CAT_TP-yhteys varastoi vastaanotettuja paketteja, kunnes koko SDU-paketti on vastaanotettu. Kun kokonainen SDU-paketti on vastaanotettu, se annetaan ylemmälle kerrokselle käsiteltäväksi. CAT_TP tietää vastaanotettujen pakettien ylätunnisteiden perusteella, milloin SDU-paketti on vastaanotettu kokonaisuudessaan. Jos lähetettävän SDU-paketin koko ylittää vastaanottavan osapuolen määrittämän suurimman mahdollisen SDU-paketin koon, täytyy komento jakaa useampiin SDU-paketteihin. Jos vastaanotettu PDU-paketti on suurempi kuin alussa määritetty PDU-paketin maksimikoko, paketti hylätään. [33, s. 10.]

CAT_TP-protokolla kostuu kahdesta kerroksesta: segmentoinnin hallinnoinnista (segment management) ja tiedonsiirron hallinnoinnista (transport management). Segmentoinnin hallinnoinnin tarkoitus on nimen mukaisesti segmentoida SDU-paketit PDU-paketeiksi lähettäjän ja vastaanottajan määrittämissä rajoissa. Se myös vastaanotettaessa hallinnoi segmentoitujen pakettien kooston. PDU-paketin ylätunnisteen SEG (segment) -bitti osoittaa, onko vielä tulossa lisää dataa, ennen kuin SDU-paketti voidaan koostaa. Kun SEG-bitti on nolla, data on tullut joko kokonaisuudessaan yhdessä paketissa tai kyseinen PDU-paketti on sarjan viimeinen eli sisältää SDU-paketin loput datat. Kun se on yksi, se viestii vastaanottajalle, että samaan katkelmaan on tulossa vielä lisää dataa. [33, s. 12.]

CAT_TP PDU-paketin ylätunniste kertoo paketin tyypin, ylätunnisteen pituuden, lähdeportin, määränpääportin, datan pituuden, sekvenssinumeron, kuittausnumeron, ikkunan koon ja tarkistussumman. Ylätunniste on 18– (18+N) tavua pitkä. Muutamille pakettityypeille ylätunnisteeseen tulee vielä lisämuuttujia, joiden pituus on N-tavua pitkä. Esimerkiksi yhteyden avauspyynnössä lähetetty SYN PDU -paketti sisältää edellä mainittujen lisäksi vastaanotettavien PDU- ja SDU-pakettien maksimikoot ja mahdollisen ID-tunnisteen. ID-tunnistetta voidaan käyttää yhteyden avaamisessa, jos sille on tarvetta. Sen avulla voidaan tunnistaa yhteyden osapuolet. ID-tunnisteen käyttö on kuitenkin vapaaehtoista: jos sitä ei tarvita, se merkitään nollaksi. Lisämuuttujiin ylätunnisteessa merkitään myös EACK PDU -paketissa kuitatut epäjärjestyksessä saapuneet paketit.

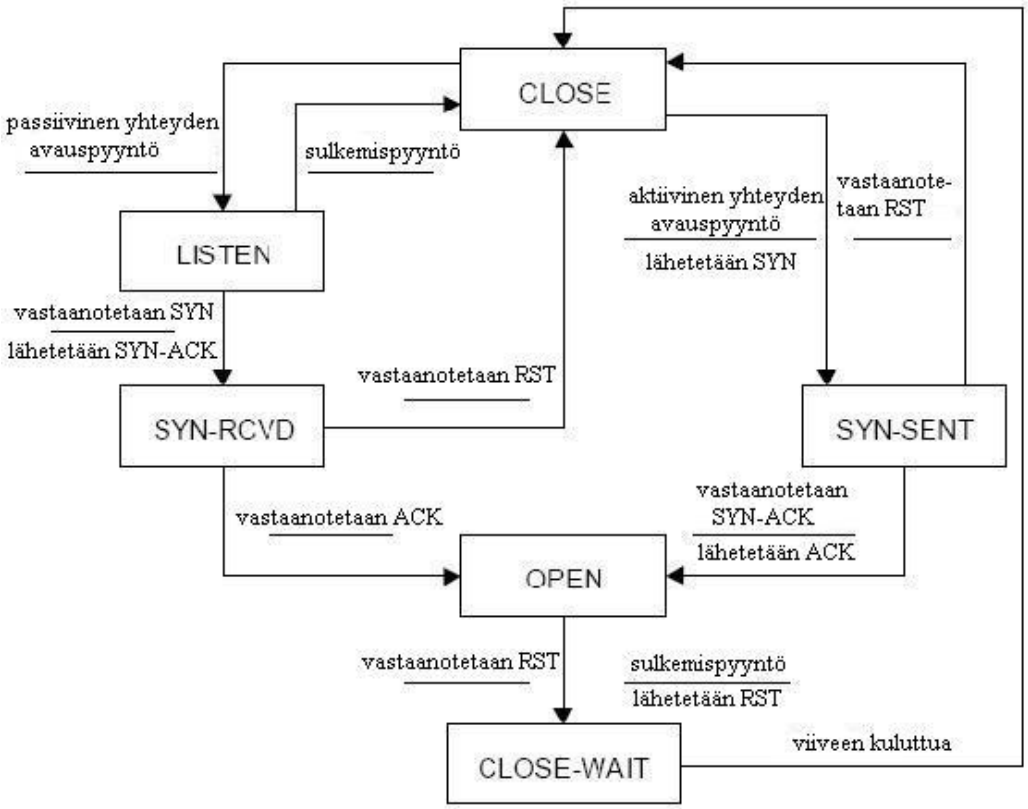
Yhteyden nollaukseen tarkoitetuilla RST PDU -paketeilla on syykoodi (reason code), joka selittää, miksi yhteys katkaistaan. [33, s. 43–44.] Syykoodit löytyvät liitteestä 8. Taulukossa 3 on vielä esitetty kaikki erilaiset CAT_TP-pakettien tyypit.

Taulukko 3. Mahdolliset CAT_TP PDU -paketit.

Paketin nimi	**Mihin käytetään?**
SYN PDU	Yhteyden luontiin ja sekvenssinumeroiden synkronointiin
SYN-ACK PDU	Yhteyden luonnin kuittaukseen ja sekvenssinumeroiden synkronointiin
ACK PDU	Onnistuneesti vastaanotetun paketin kuittaukseen ja ikkunan koon ilmoittamiseen
ACK-SEG-DATA PDU	Segmentoidun datan lähettämiseen (ei sarjan viimeinen tai ainut)
ACK-UNSEG-DATA PDU	Datan lähettämiseen (joko segmentoidun datan viimeinen tai yksittäinen)
ACK-EACK PDU	Epäjärjestyksessä saapuneiden pakettien kuittaukseen
ACK-EACK-UNSEG-DATA PDU	Epäjärjestyksessä saapuneiden pakettien kuittaukseen ja datan lähettämiseen
NUL-ACK PDU	Yhteyden tilan tarkistamiseen ja vastaanotetun paketin kuittaamiseen
RST PDU	Yhteyden nollaamiseen
RST-ACK PDU	Yhteyden nollaamiseen ja vastaanotetun paketin kuittaamiseen
NUL-EACK PDU	Yhteyden tilan tarkistamiseen ja epäjärjestyksessä saapuneiden pakettien kuittaamiseen

Tiedonsiirronhallinnointikerros vastaa yhteydestä. Se vastaa yhteyden avaamisesta, käytetystä portista, yhteyden eri tiloista, yhteyden muuttujista, yhteyden sulkemisesta, käytetystä versionumerosta ja puoliavoimen tai toimettoman yhteyden havaitsemisesta. Yhteys on puoliavoin, kun toisen kommunikaation osapuolen yhteys on katkennut. Yhteyden tilan voi varmistaa lähettämällä NUL PDU -paketin vastapuolelle. Yhteys on toimeton, kun kaikki vastaanotetut paketit ovat kuitattu, yhtään pakettia ei ole prosessoinnissa, kaikki data on lähetetty eikä liikennettä ole esiintynyt kanavalla hetkeen. [33, s. 13–17.]

CAT_TP-yhteys etenee useissa yhteystiloissa elinkaarensa aikana. CAT_TP-protokollan eri yhteystilat ovat Closed-, Listen-, Syn-Rcvd-, Syn-Sent-, Open- ja Close-Wait-tila. Kuvassa 5 esitetään CAT_TP-protokollan määrittämät tilat, ja nuolet kuvassa osoittavat suunnan, johon kyseiset tilat voivat edetä. Kuvassa esitetään myös tapahtumat, jotka saavat CAT_TP-yhteyden muuttamaan tilaansa.



Kuva 5. CAT_TP-yhteyden yhteystilat [33, s. 14].

CAT_TP-yhteys on Closed-tilassa, kun mitään yhteyttä ei esiinny kahden portin välillä. CAT_TP-yhteyden voi avata kahdella eri tavalla, joko aktiivisesti tai passiivisesti. Aktiivisessa avaamisessa CAT_TP-toteutus menee Syn-Sent-tilaan ja lähettää SYN PDU -paketissa pyynnön yhteyden avaamisesta CAT_TP-palvelimelle (CAT_TP server). CAT_TP-palvelin on se yhteyden osapuoli, joka vastaanottaa yhteyden avauspyynnön, CAT_TP-asiakas (CAT_TP client) taas lähettää sen. Molemmat, taustajärjestelmän

palvelinkone ja matkapuhelimen UICC-kortti, voivat toimia molempina CAT_TP-palvelimina ja -asiakkaina, eli molemmat pystyvät avaamaan yhteyden ja vastaamaan avauspyyntöihin. Aktiivisen yhteyden avaavan pitää tietää CAT_TP-palvelimen kuuntelema portti. Yleensä älykortti tekee aktiivisen yhteyden avaamisen, ja se saa tarvittavat tiedot siihen PushSMS-viestissä, jonka palvelinkone on sille aikaisemmin lähettänyt.

Passiivisessa yhteyden avaamisessa CAT_TP-toteutus siirtyy Listen-tilaan ja alkaa kuunnella porttiaan SYN PDU -paketin vastaanottamiseksi. CAT_TP-yhteys siirtyy Syn-Rcvd-tilaan Listen-tilasta, kun SYN PDU -paketin avauspyyntö saapuu ja siihen on lähetetty kuittaus (SYN-ACK PDU) CAT_TP-asiakkaalle. Kun CAT_TP-asiakas saa CAT_TP-palvelimelta kuittauksen (SYN-ACK PDU) lähettämälleen avauspyynnölle (SYN PDU), se tallentaa CAT_TP-palvelimen asettamat yhteystiedot, kuten maksimikoot PDU- ja SDU-paketeille, ikkunan koon (window size) ja sekvenssinumeron (sequence number), ja lähettää vielä kuittauksen (ACK PDU) tästä CAT_TP-palvelimelle. Tämän jälkeen CAT_TP-asiakas siirtyy Open-tilaan. Myös CAT_TP-palvelin siirtyy Open-tilaan, kun saa CAT_TP-asiakkaalta kuittauksen (ACK PDU) lähettämälleen avauspyynnönkuittaukselle (SYN-ACK PDU).

CAT_TP-yhteys on avattu, kun CAT_TP-toteutus on Open-tilassa. Tällöin yhteyden osapuolet voivat aloittaa kaksisuuntaisen tiedonvaihdon keskenään. Close-Wait-tilaan siirrytään Open-tilasta, jos vastaanotetaan sulkemispyyntö ylemmältä kerrokselta tai vastaanotetaan RST PDU -paketti. Saadessaan sulkemispyynnön ylemmältä kerrokselta CAT_TP-toteutuksen pitää lähettää RST PDU -paketti toiselle osapuolelle, jotta sekin osaa sulkea yhteyden. CAT_TP-toteutus pysyy hetken Close-Wait-tilassa, minkä jälkeen se sulkeutuu kunnolla takaisin Close-tilaan. Close-Wait-tilassa kaikki vastaanotetut paketit hylätään. [33, s. 13–15.]

CAT_TP-protokollaa pidetään luotettavana tiedonsiirtoprotokollana, koska sen useat mekanismit edistävät luotettavaa kommunikointia. CAT_TP-protokollassa määritetään

kaikille lähetettäville paketeille jokin sekvenssinumero, joka identifioi ne saman yhteyden muista paketeista. CAT_TP-toteutuksen sekvenssinumeroa korotetaan aina yhdellä, kun on lähetetty paketti, joka vaatii vastapuolelta kuittausta. Sekvenssinumeroiden perusteella vastaanottava osapuoli tietää pakettien oikean järjestyksen, ja sen avulla se pystyy kuittaamaan lähettäjälle, että paketti on vastaanotettu. Pakettien kuittauksen avulla tiedetään, että lähetetty paketti on varmasti vastaanotettu. Jos kuittausta ei saada tietyn ajan kuluessa, CAT_TP-toteutus lähettää sen uudelleen. Uudelleenlähetystä varten jokaiselle yhteydelle määritetään, kuinka monesti paketti lähetetään uudelleen. CAT_TP-protokolla pitää huolen laskurin avulla, että paketti lähetetään niin monta kertaa, kuin on määritetty. Jos sitä ei kuitata tämänkään jälkeen, yhteys suljetaan. Paketit kuitataan vain silloin, kun ne on oikein vastaanotettu ja hyväksytty. Kaikkia paketteja ei kuitenkaan kuitata; esimerkiksi normaalia kuittausta, jossa ei ole dataa, ei kuitata takaisin. Jokainen lähetetty paketti vielä varustetaan tarkistussummalla (checksum). Tarkistussumman avulla vastaanottaja voi tarkistaa, että paketti ei ole virheellinen. CAT_TP käyttää 16-bittistä TCP-tarkistussummaa, joka on määritetty RFC 793 -standardissa. Tarkistussumma on paketin kaikkien 16-bittisten sanojen summan yhden komplementin yhden komplementti. [33, s. 17–18.]

CAT_TP-protokollassa sen osapuolet ilmoittavat toisilleen jonkin ikkunan koon, jonka sisällä ne ovat valmiita vastaanottamaan paketteja. CAT_TP-toteutus ilmoittaa senhetkisen ikkunan kokonsa aina kuittausten yhteydessä vastapuolelle. Ikkunan koko määrittää käytännössä, kuinka monta dataa sisältävää pakettia voidaan lähettää toiselle osapuolelle ilman, että tarvitsee odottaa kuittauksia. Suurin sallittu paketin sekvenssinumero on viimeksi tulleessa kuittauksessa kuitatun paketin sekvenssinumeron (acknowledgement number) ja siinä määritetyn ikkunan koon summa. Jos ikkunan raja saavutetaan, keskeytyy lähetys, kunnes vastapuoli ilmoittaa uuden ikkunan koon. Jos jompikumpi osapuoli ilmoittaa ikkunan kookseen nolla, lähetys keskeytetään, kunnes se taas ilmoitetaan suuremmaksi NUL PDU -paketissa. Kaikki lähetetyt paketit, joiden sekvenssinumero osuu vastaanottavan ikkunan sisään, pitää kuitata vastapuolelle. [33, s. 18–14.]

5 CAT_TP-tiedonsiirtoprotokollan testausjärjestelmä

5.1 Tilaajan tarve

Venyonin nykyinen (U)SIM-kortinhallintatoteutus käyttää (U)SIM-kortin hallinnointiin puhelimeen asennettavaa Java Midletiä. Tämä lähestymistapa ei ole kuitenkaan teleoperaattoreiden suosima. Ne pitävät ideaalisimpana hallinnoida matkapuhelimien (U)SIM-korttia OTA-tekniikoilla, joissa ollaan yhteydessä suoraan (U)SIM-korttiin. SIM-protokollista teleoperaattorit suosivat CAT_TP-protokollaa ylitse TCP-protokollan, sillä se on suunniteltu UDP-protokollan yläpuolelle. TCP-protokolla on luotettavampi kuin UDP-protokolla yksistään, sillä UDP-protokolla ei takaa lähetettyjen pakettien perille menoa tai sitä, että ne saapuvat oikeassa järjestyksessä. TCP-protokolla taas varmistaa, että kaikki paketit saapuvat oikeassa järjestyksessä ja varmasti perille. TCP-protokolla laskee nopeuttaan aina 50 prosenttia, jos paketti eksyy matkalla. Tämän avulla on tarkoitus estää mahdolliset tukkeutumiset. Näin ollen, jos TCP-protokollaa käytettäessä katoaa paketteja, siitä tulee hidas yhteyskanava. TCP-protokollaa hidastaa myös sen ylimääräinen virhetilojen tarkistaminen. Kun UDP-protokollan yläpuolella käytetään CAT_TP-protokollaa, siitä saadaan nopea ja luotettava yhteyskanava laitteiden välille, koska CAT_TP varmistaa pakettien saapumisen ja järjestää ne, jos niitä vastaanotetaan epäjärjestyksessä. CAT_TP-protokollassa hyödynnetään UDP-protokollan nopeus, mutta siinä on TCP-protokollan tapainen varmistus, että kaikki paketit saadaan varmasti toimitettua perille. [34.]

Koska CAT_TP-protokollasta ei ole saatavilla kaupallisia toteutuksia, se pitää toteuttaa itse. On tärkeää, että se on toteutettu täysin CAT_TP-protokollan määrittämän ETSI TS 102 127 -spesifikaation mukaisesti. ETSI on määritellyt CAT_TP-protokollan testausta varten oman testispesifikaation (ETSI TS 102 431). Venyonin tarve oli, että tämän spesifikaation perusteella kehitetään testausjärjestelmä, jolla pystytään testaamaan sen CAT_TP-toteutus.

5.2 Ohjelmistotestaus

Ohjelmistotestauksen periaatteita

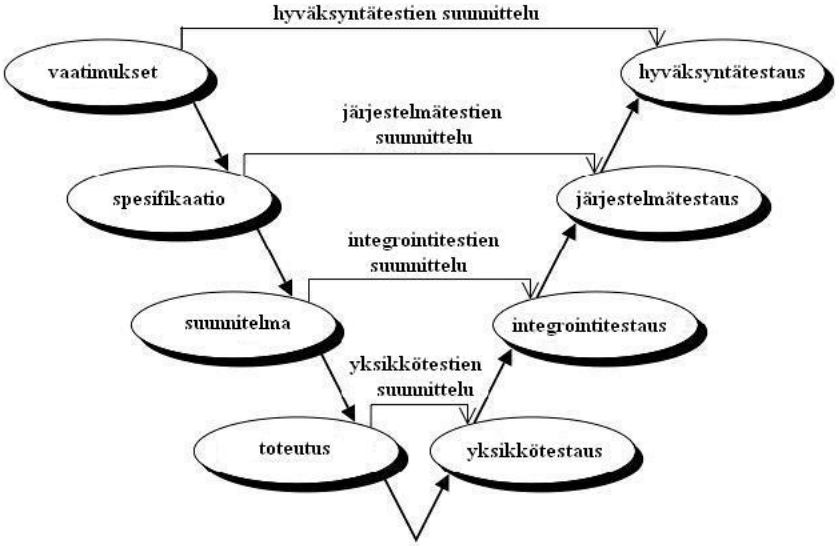
Testauksessa testataan, että sovellus toimii kunnolla, sen jokaisena hetkenä, myös silloin, kun tapahtuu jotain odottamatonta. Testaus jaetaan kahteen osaan: positiiviseen ja negatiiviseen testaukseen. Positiivisessa testauksessa testataan, että sovellus toimii, kuten on vaadittu. Negatiivisessa taas pyritään etsimään virheitä tahallisesti. Käytännön ohjelmistotestaus on näiden yhdistelmä: siinä pyritään vakuuttamaan, että sovellus toimii vaatimusten mukaisesti, mutta samalla yritetään löytää sovelluksesta virheitä pois. Täydentävinä testaustapoina pidetään white box- ja black box -testausta. White box -testauksessa suunnitellaan testaus tietäen, miten testattava toteutus on rakennettu. Tämän testauksen usein suorittaa jo ohjelmistokehittäjä, ja sillä testataan pienten kokonaisuuksien toimivuutta. Black box -testauksessa taas päinvastoin ei tiedetä, miten testattava toteutus on rakennettu, ja siinä testaussuunnitelma tehdään ulkoisen käyttäytymisen pohjalta. Black box -testausta tehdään yleensä testausvaiheen loppuvaiheessa. Testauksella ei voida koskaan tehdä sovelluksesta täydellistä, koska tietokonejärjestelmät ovat niin monimutkaisia, mutta sen avulla pystytään määrittämään, onko toteutus tarpeeksi vakaa siihen käyttöön, mihin se on tarkoitettu. [35, s. 9–11; 35, s. 16–17.]

Testauksen lähtökohta on määrittää tarkasti testin haluttu lopputulos. Kuulostaa ilmiselvältä, mutta monesti testauksen lopputulosta ei ole tarkasti määritetty ja testauksen läpäisevät sellaiset testit, joiden pitäisi epäonnistua siinä. Tämä johtuu siitä, että kun ei ole tarkkaa lopputulosta, testaaja saattaa tulkita joitakin tuloksia liian sinisilmäisesti, varsinkin silloin, kun testaaja itse on kehittänyt sovelluksen. Siksi testeille pitää aina määrittää tarkasti, mitä testataan ja mikä on odotettu lopputulos. Saatu lopputulos pitää myös aina tarkastaa läpikotaisesti. Yleinen sääntö on, että sovelluksen kehittäjän tulisi välttää oman sovelluksen lopullinen testaus, sillä monesti omasta sovelluksesta ei tahdota löytää virheitä tai, jos sovelluskehittäjä on ymmärtänyt jonkin sovelluksen spesifikaation osan väärin, hän myös todennäköisesti testaa sen väärin. Sovelluskehittäjän on kuitenkin hyvä tehdä

yksikkötestausta sovellukselleen jo kehitysvaiheessa, jotta ilmiselvät virheet saadaan kitkettyä pois.

Testauksessa tulee tehdä testit niin odotetuille kuin odottamattomille tapahtumille. Se, että tarkastelee sovellusta siltä kannalta, mitä sen pitäisi tehdä, on vain puolet testauksesta. Sovellusta pitää myös tarkastella toiselta suunnalta eli siltä, mitä sen ei pitäisi tehdä. On hyvin yleistä, että sovellusta katsellaan vaan siltä kannalta, mitä sen pitäisi tehdä, mutta virheet sovelluksesta löydetään suuremmalla todennäköisyydellä silloin, kun se testataan odottamattomilla tapahtumilla. Yhden testauskierroksen ja siinä esiintyneiden virheiden korjauksen jälkeen on tärkeää testata koko sovellus uudelleen. Täytyy varmistaa, että korjaukset jossakin kohtaa sovellusta eivät aiheuta niitä lisää toisessa. Testausta ei saa suunnitella ajatellen, että virheitä ei tule löytymään, koska se sotii koko testauksen periaatetta vastaan. Testauksessa juuri pyritään löytämään virheitä. Koska testauksessa tulee osata ajatella sovellusta monelta kannalta, se on erittäin haastavaa ja luovaa. Testaukselle tulee jättää aina tarpeeksi aikaa, eikä sitä tule tehdä vain nopeasti juuri ennen sovelluksen toimituksen aikarajaa. [36, s. 14–20.]

V-malli (V model) on hyväksi todettu malli sovellusten toteuttamiseen ja testaamiseen. Sen vasen sakara osoittaa sovelluksen kehitysvaiheet ja oikea testausvaiheet. V-mallin jokainen testausvaihe on linkitetty johonkin kehitysvaiheeseen, ja jokaiselle testausvaiheelle tehdään kehitysvaiheen dokumentaatiota vastaava testaussuunnitelma. Testauksen jälkeen testaussuunnitelmia ja vastaavien kehitysvaiheiden dokumentteja verrataan. Kuvassa 6 on esitetty V-mallin määrittämä sovelluskehitysprosessi.



Kuva 6. V-testausmalli [35, s. 40].

Testaus jaetaan useisiin vaiheisiin, kuten V-malli osoittaa. Yksikkötestaus (unit testing) on alimmalla testaustasolla, sen alemmaksi ei voi mennä. Siinä testataan yksittäisiä sovelluksen komponentteja, kuten funktiota, ja sen on tarkoitus selvittää, että komponentti yksistään toimii, kuten vaaditaan. Siinä yksittäinen funktio otetaan pois kokonaisuudesta ja testataan sen toimivuutta. Yksikkötestaus suoritetaan yleensä white box -testauksen tavoin. Yksikkötestauksen suorittavat yleensä ohjelmistokehittäjät jo sovelluksen kehityksen aikana. [35, s. 35.]

Integrointitestauksen (integration testing) päämääränä on testata, että moduulit, jotka yhdessä muodostavat testattavan toteutuksen, toimivat yhdessä oikealla tavalla, yhdenmukaisesti ja vakaasti. Integrointitestauksen tekee yleensä sovelluskehitystyöryhmä, mutta suuremmissa organisaatiossa sen tekevät erilliset testaajat. Monesti, kun integroidaan erillisiä osia yhteen kokonaisuudeksi, esiintyy jotain virheitä, joko välittömästi tai jonkin käyttöajan kuluessa. Integrointitestauksessa voivat tulla esille esimerkiksi sovelluskehittäjien spesifikaation tulkitsemisvirheet. Koska yksikkötestauksen usein tekee sovelluskehittäjä itse, spesifikaation väärästä tulkitsemisesta johtuvat virheet voivat jäädä huomaamatta. [35, s. 36.]

Järjestelmätestauksen (system testing) päämääränä on saada luottamus, että käyttäjäryhmä tai sovelluksen tilaaja tulee hyväksymään sovelluksen. Järjestelmätestauksen suorittaa erillinen testausryhmä, joka ei ole ollut osallisena sovelluksen kehitykseen. Järjestelmätestauksessa testataan järjestelmän toiminnallista ja rakenteellista kestävyyttä sekä suorituskykyä ja luotettavuutta. Se tehdään black box -testausmekanismilla. Lisäksi, jos testattava järjestelmä on käyttöympäristössään vuorovaikutuksessa muiden järjestelmien kanssa, sille tehdään järjestelmäintegrointitestaus (system integration testing). Siinä testataan, että testattavalla toteutuksella ei ole haittavaikutuksia muihin järjestelmiin ja että muilla järjestelmillä ei ole haittavaikutuksia testattavaan toteutukseen. [35, s. 59.]

Viimeisenä tehdään hyväksyntätestaus (acceptance testing), jonka tarkoitus on varmistaa, että järjestelmä täyttää kaikki sovellukselle määritetyt vaatimukset, ja osoittaa, että sovellus toimii oikein. Tämän jälkeen sovellus voidaan todeta käyttövalmiiksi, ja se voidaan toimittaa tilaajille. Hyväksyntätestaus jaetaan yleisesti kahteen osaan: käyttäjähyväksyntä-testaukseen (user acceprance testing) ja operaatiohyväksyntätestaukseen (operations acceptance testing). Käyttäjähyväksynnän tekevät sovelluksen käyttäjät ja operaatiohyväksynnän sovelluksen hallinnoijat. [35, s. 38.]

JUnit-testausohjelmisto

JUnit on avoimen lähdekoodin runko (framework) testaamiseen. Sen avulla voidaan kirjoittaa ja ajaa automatisoituja testejä omalle sovellukselle. JUnitia pidetään standardi-kirjastona testausta varten Java-ohjelmointikielelle. Sen myötä ohjelmakoodin testaus on helpottunut huomattavasti. JUnitia käytetään pääasiassa yksikkötestaukseen. [37.]

Pääasiassa JUnit tarjoaa joukon funktioita, joiden avulla pystytään testaamaan, että koodi toimii, kuten on oletettu. Sen avulla pystytään vertailemaan objekteja, funktion palautusarvoja ja muuttujia johonkin oletettuun arvoon. Vertailun epäonnistuessa se raportoi eroavaisuudet ja osoittaa kohtaan, jossa testi epäonnistui. Kun JUnit-testi onnistuu, se näyttää vihreää palkkia, ja virhetilanteissa joko sinistä tai punaista. Sen avulla pystytään

ajamaan kerralla useita testejä ja saadaan heti tietää, miten testit onnistuivat. JUnitin Assert-luokka määrittää tärkeimmät funktiot testaamiseen. [38, s. 3–4.] Taulukossa 4 on kuvattu niistä muutama tärkein.

Taulukko 4. Assert-funktiot [39].

Funktio	**Kuvaus**
AssertEquals(String message, Object expected, Object actual)	AssertEquals vertaa kahta objektia keskenään. Jos ne ovat samat, testi hyväksytään.
AssertFalse(boolean condition)	Jos ehto palauttaa false-totuusarvon, testi hyväksytään.
AssertNull(Object object)	Jos objekti on NULL, testi hyväksytään. Testille myös vastakohta AssertNotNull.
AssertTrue(boolean condition)	Jos ehto palauttaa true-totuusarvon, testi hyväksytään.
Fail(String message)	Fail hylkää testin ja näyttää sille annetun viestin.

JUnit tarjoaa itse testaukseen käytettävien funktioiden lisäksi keinot alustaa testi ja jälkikäsitellä se. Sen avulla voidaan määrittää joitakin toimenpiteitä, jotka pitää tehdä aina, ennen kuin testi ajetaan ja voidaan määrittää toimenpiteet, jotka testin jälkeen tehdään. Hyvin usein testin jälkeen kaikki nollataan, jotta seuraava testi voidaan suorittaa taas puhtaalta pöydältä. [38, s. 14.]

JUnit on toteutettu alun perin Java-ohjelmointikielelle, mutta se on saatavilla myös muille ohjelmointikielille. Sillä on helppo tehdä testejä, koska se on suunniteltu ajatellen ohjelmoijia, jotka eivät ole testaajia. [38, s. 9.]

5.3 Testausjärjestelmä

Insinöörityön osana tehty testausjärjestelmä kehitettiin ETSI TS 102 431 -spesifikaation pohjalta, joka on testaussuunnitelma CAT_TP-toteutuksien testaamiseen. Spesifikaation tavoitteena on, että sen testaussuunnitelman avulla lokaali CAT_TP-toteutus pystyy saumattomasti kommunikoimaan ulkoisen CAT_TP-toteutuksen kanssa. Se määrittää

tekniset ominaisuudet ja tavat testata CAT_TP-toteutusta. Liitteessä 9 on esitetty kaikki toiminnallisuus, joka tulee testata. Koska testausspesifikaatio oli jo valmiiksi määritetty, tehtäväkseni jäi testauksen toteuttaminen Venyonin CAT_TP-toteutukseen ja virheiden raportointi sekä korjaaminen.

Testausjärjestelmä koostuu CAT_TP-toteutuksesta, jota testataan ja testioperaattorista. Testioperaattori hoitaa testien ajamisen, tarkistaa CAT_TP-toteutuksen vastausviestit ja ohjailee CAT_TP-toteutusta antamalla komentoja ylemmältä kerrokselta, ja sen pitää pystyä tarkistamaan CAT_TP-toteutuksen ylemmälle kerrokselle antamat SDU-paketit. Jokainen testi on itsenäinen, eli kaikki pitää nollata jokaisen testin jälkeen, jotta seuraava testi voidaan suorittaa puhtaalta pöydältä.

Testispesifikaatio esittää jokaiselle testille alkuehdot, jotka sen tulee täyttää, jotta testi on validi. Jokainen testausjärjestelmän testi alustettiin näiden alkuehtojen mukaan. Alkuehdoissa määritetään, mitä CAT_TP-toiminnallisuuksia CAT_TP-toteutuksen tulee tukea, ja niissä annettiin yhteydelle joitakin arvoja, kuten maksimikoot PDU- ja SDU-paketeille, ikkunan koko, sekvenssinumero ja niin edelleen.

Testispesifikaatio määrittää vaatimukset testin suoritukselle ja tuloksille. Menettelytavoissa kuvataan testiskenaario. Siinä on askel askeleelta määritetty, mitä testattavalle toteutukselle lähetetään, mitä komentoja sille annetaan ja mitä oletetaan vastaukseksi. Testattavan toteutuksen vastausviestien lisäksi ohjeistettiin tarkistamaan, että testattava viestii mahdolliset virhetilat ylemmälle kerrokselle ja tarkistamaan sen vastaanottamat ja koostamat SDU-paketit. Katsotaan, että testi on hyväksytysti suoritettu, jos se on määrittänyt alkuehdot, edennyt menettelytavan mukaisesti ja saavuttanut oletetut tulokset eli testattavan vastauspaketit ja ylemmälle kerrokselle viestitty sisältö on testispesifikaation mukaista. Testausspesifikaatio määrittää myös syyt, miksi testit tulee toteuttaa.

CAT_TP-toteutuksen testaus suoritettiin Junit 4.4 -testausohjelmistolla. JUnit tarjoaa hyvän kirjaston testaukseen, ja sen avulla voidaan automatisoidusti testata kaikki CAT_TP-

testausjärjestelmän testit. Se ilmoittaa heti, onko testi mennyt läpi, ja epäonnistuessaan se kertoo syyn epäonnistumiselle. JUnitin tarjoamat keinot alustaa testi ja jälkikäsitellä se tekee siitä ideaalisen työkalun testata CAT_TP-toteutusta. Testit jaettiin niiden lähtökohtaisten CAT_TP-tilojen mukaan. Vaikka jokainen testi tulee aloittaa puhtaalta pöydältä eli Close-tilasta, usein itse testattava ominaisuus, oletettavastikin, esiintyi jossain muussa CAT_TP-tilassa ja yhteyden alustaminen tiettyyn tilaan tapahtui aina samalla tavalla, ellei toisin määritetty. Näin ollen pystyttiin helposti alustamaan jokainen testi Close-tilasta siihen tilaan, jossa testattava ominaisuus oli, ja sitten ajamaan itse testi. Jokaisen testin jälkeen kaikki nollattiin ja CAT_TP-toteutus palautettiin Close-tilaan. Testin jälkeen vielä poistettiin koko CAT_TP-objekti, ja se luotiin aina testin alustuksessa uudelleen, mistä se sitten alustettiin vielä haluttuun tilaan. Tämä varmisti sen, että jokainen testi oli varmasti riippumaton muista. Omat testausluokat luotiin siis Close-, Listen-, Syn-Sent-, Syn-Rcvd- ja Open-tiloille, joissa kaikissa testattiin useampia ominaisuuksia. Close-Wait-tilalle testispesifikaatio ei määrittänyt omia testejä, sillä aika, jonka se odottaa ennen kuin siirtyy Close-tilaan, on täysin CAT_TP-toteutuksesta riippuvainen. Testioperaattori ei saa vaikuttaa siihen.

Testausjärjestelmä määrittää rajapinnan, joka testattavan järjestelmän tulee toteuttaa. Rajapinta määrittää funktioita CAT_TP-toteutuksen ja yhteyden alustamiseen, ylemmän kerroksen komentoihin, muutamien muuttujien arvojen ja testattavan tilan saamiseen ja PDU-pakettien syöttämiseen. Liitteessä 10 on esitetty kokonaisuudessa testausjärjestelmän määrittämä rajapinta.

Testausjärjestelmän testit tehtiin black box -testausmekanismilla. Testauksessa suoritetaan testausspesifikaation määrittämiä toimintoja ja tarkistetaan, että testattava toteutus käyttäytyy oikealla tavalla. Testausjärjestelmä ei välitä, miten testattavan toteutuksen prosessointi on toteutettu, kunhan sen lopputulos on CAT_TP-spesifikaation mukainen. Testauksessa annetaan CAT_TP-toteutukselle jokin komento tai paketti prosessoitavaksi ja testataan, että se toimii sen pohjalta, huomioiden testin alkuehdot, spesifikaation mukaisesti. CAT_TP-toteutuksen saatua komennon tai paketin prosessoitavaksi siltä

voidaan odottaa viittä eri lopputulosta: vastauspakettia, viestiä virheestä ylemmälle kerrokselle, tilamuutosta, edellisten yhdistelmiä tai saadun paketin hylkäystä ilman, että se vaikuttaa mihinkään. Esitän muutamilla esimerkeillä testausjärjestelmän periaatteen ja toteutuksen. Kaikkia testausjärjestelmän testejä en käy läpi, koska niitä tuli 106 kaiken kaikkiaan.

Yksi yksinkertaisimmista testeistä on testausspesifikaation otsikkonumerolla 7.2.1.1 numeroitu testi, jossa testataan CAT_TP-yhteyden normaalia passiivista avausta. Testin alkuehdoiksi on annettu, että testattava (CE) on Listen-tilassa ja kommunikaation vastapuolelle, joka on tässä tilanteessa testioperaattori (CS), on määrätty alkusekvenssinumeroksi 200. PDU-paketin maksimikooksi on määrätty 500 tavua ja SDU-paketin maksimikooksi 1 000 tavua sekä lähdeportiksi portti 1024. Testin tarkoituksena on määrittää, että testattava on Open-tilassa lähetettyään SYN-ACK PDU -paketin ja saatuaan siihen vastaukseksi ACK PDU -paketin. Testistä on lisäksi kaksi skenaariota: vastaanotettu ACK PDU -paketti saapuu järjestyksessä muiden pakettien kanssa ja se saapuu epäjärjestyksessä. Kuvassa 7 on esitetty testin skenaario, kun ACK PDU saapuu järjestyksessä.

		CE		CS
Time	State			
1.	LISTEN		<---	<SEQ=200><SYN>
2.	SYN-RCVD	<SEQ=X><ACK=200><SYN,ACK>	--->	
3.	SYN-RCVD		<---	<SEQ=201><ACK=X><ACK>
4.	OPEN		<---	<SEQ=201><ACK!=X><NUL,ACK>
5.	OPEN	<SEQ=X+1><ACK=201><ACK>	--->	

Kuva 7. Skenaario CAT_TP-yhteyden normaalista passiivisesta avaamisesta [27, s. 40].

Testissä testioperaattori toimittaa testattavalle SYN PDU:n ja tarkistaa, että saa vastaukseksi SYN-ACK PDU:n muotoiltuna oikein ja että testattava on Syn-Rcvd-tilassa. Sitten testioperaattori toimittaa testattavalle ACK PDU:n ja tarkistaa, että sen tila on muuttunut Open-tilaan. Lopuksi tämä vielä varmistetaan lähettämällä testattavalle NUL PDU, johon odotetaan vastaukseksi ACK PDU:ta, jossa on sekvenssinumerona ja kuittausnumerona oikeat arvot. Testi voidaan hyväksyä, jos se on mennyt täysin

testausspesifikaation mukaisesti. Liitteessä 11 on testin toteutus. Ennen testin ajoa testattava alustettiin Listen-tilaan eli sille annettiin passiivinen avauskutsu.

Toisessa esimerkissä testattavalle annetaan komento ja siitä odotetaan poikkeustilaviestiä takaisin ylemmälle kerrokselle. Testi on numeroitu testispesifikaatiossa numerolla 7.2.2.8, ja siinä annetaan Syn-Rcvd-tilassa olevalle CAT_TP-toteutukselle Send-komento. Testin tarkoituksena on varmistaa, että testattava ei ota tehtäväkseen datan lähettämistä ja pysyy Syn-Rcvd-tilassa. Lisäksi varmistetaan, että se antaa poikkeustilaviestin ”yhteys ei ole avattu” takaisin ylemmälle kerrokselle. Datalla, joka annetaan Send-komennon yhteydessä, ei ole testin kannalta merkitystä. Testin alkuehtoina on, että testattava on Syn-Rcvd-tilassa. Testioperaattorille alkuehdot ovat samat kuin edellisessä esimerkissä. Kuten testijärjestelmän rajapinnasta nähdään, ylemmän kerroksen komennot pystyvät heittämään CatTpException-poikkeustiloja. Tässä testissä odotetaan, että Send-komento heittää ”yhteys ei ole avattu” -poikkeustilan. Testin toteutuksessa se testataan try-catch-rakenteella. Heti testattavan saatua Send-komennon Syn-Rcvd-tilassa se heittää poikkeustilan, jonka nappaa catch-osio. Catch-osiossa verrataan, että poikkeuksen syy on sama kuin mitä testispesifikaatiossa odotetaan. Testin skenaariosta ei ole kuvaa. Liitteessä 12 on testin toteutus.

Viimeisessä esimerkissä testataan yhteyden sulkemista Close-komennolla, kun testattava on Open-tilassa, ja varmistetaan, että se lähettää RST PDU:n, vaikka yhteyden vastapuoli on ilmoittanut ikkunan kookseen nolla. Testispesifikaatiossa testi kulkee numerolla 9.1.13. Testin alkuehtoina on, että testattava on Open-tilassa. Testioperaattorille alkuehdot ovat samat kuin ensimmäisessä esimerkissä. Kuvassa 8 on esitetty testin skenaario.

		CE		CS
Time	State			
1.	OPEN	<SEQ=X+1><ACK=200><ACK, Data><SEG=0>	--->	
2.	OPEN		<---	<SEQ=201><ACK=X+1/0><ACK,Data><SEG=0>
3.	OPEN	<SEQ=X+2><ACK=201><ACK>	--->	
4.	OPEN	<SEQ=X+2><ACK=201><RST>	--->	
5.	CLOSE			

Kuva 8. CAT_TP-yhdeyden sulkeminen, kun CS:n ikkunan koko on 0 [27, s. 149].

Aluksi testissä annetaan testattavalle Send-komento, johon testioperaattori vastaa ACK-DATA PDU:lla, joka ilmoittaa sen ikkunan kooksi nolla. Testioperaattorin ACK-DATA PDU:hun odotetaan kuittausta testattavalta. Yleensä, jos vastapuoli ilmoittaa ikkunan kookseen 0, lähetys keskeytetään, mutta RST PDU voidaan kuitenkin toimittaa. Testioperaattorin saatua kuittauksen ACK-DATA PDU:hun, se antaa testattavalle Close-komennon ja odottaa, että testattava menee Close-Wait-tilaan ja toimittaa RST PDU:n testioperaattorille. Liitteessä 13 on testin toteutus.

Yllä olevien esimerkkien tapaisesti testattiin kaikki testispesifikaation määrittämät testit. Monet testeistä pystyttiin suorittamaan melkein yhtä helposti kuin edelliset esimerkit, mutta jotkin testit olivat hankalampia.

5.4 Tulokset

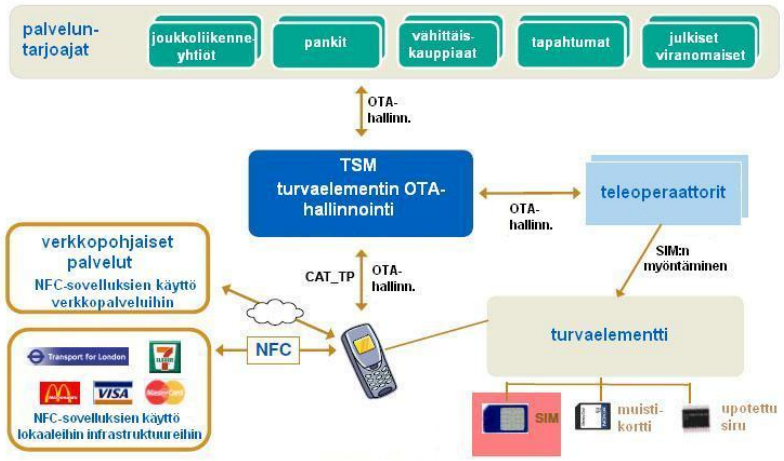
Testijärjestelmää varten toteutettiin 106 testiä, jotka testaavat CAT_TP-toteutusta kaikilla mahdollisilla tavoilla. Useita testien skenaarioita tuskin edes ilmenee CAT_TP-toteutuksen lopullisessa käytössä, mutta on silti tärkeää testata ne läpi. Testit oli melko helppo toteuttaa, kun keksi tarpeeksi yksinkertaisen lähestymistavan. Ensimmäiset suunnitelmat testijärjestelmää varten olivat huomattavasti monimutkaisempia, ja ne olisivat loppujen lopuksi varmasti vain heikentäneet testijärjestelmän laatua. Testijärjestelmän toteutus eli koko sen kehittämisprosessin ajan, ja lähestymistapa testijärjestelmään yksinkertaistui jatkuvasti, mikä paransi sen laatua.

Testijärjestelmää kehitettäessä havaittiin yksi suuri ongelma. ETSI-järjestön kehittämä testispesifikaatio ETSI 102 431 erosi useita kertoja CAT_TP-tiedonsiirtoprotokollan määrittämästä ETSI TS 102 127 -spesifikaatiosta. Näissä tilanteissa päätettiin toteuttaa testit CAT_TP-spesifikaation pohjalta, koska voidaan olettaa, että se on tehty huomattavasti huolellisemmin kuin testausspesifikaatio CAT_TP:lle. Ennen jokaisen testin toteuttamista piti käydä tarkasti CAT_TP-spesifikaation perusteella testi läpi, jotta saatiin selville, että testausspesifikaation määrittämä skenaario oli tehty oikein.

Tätä työtä kirjoittaessani testijärjestelmä on jo integroitu Venyonin tulevan CAT_TP-toteutuksen kanssa. Alustavissa testauksissa on pystytty havaitsemaan monia virheitä ja poistamaan ne alustavasta toteutuksesta. Toteutuksen keskeneräisyyden vuoksi se ei kuitenkaan vielä läpäise kaikkia testejä. Nyt, kun testijärjestelmä on olemassa ja integroituna CAT_TP-toteutukseen, toteutuksen kehitysvaiheessa voidaan testata sen oikeellisuutta spesifikaatiota vastaan, mikä helpottaa sen kehitystä loppuun. Sen avulla osataan jo kehitysvaiheessa huomioida monia seikkoja, jotka voisivat muuten jäädä huomioimatta.

6 Yhteenveto

CAT_TP-tiedonsiirtoprotokolla on yksi pieni osa suurempaa kokonaisuutta: NFC-teknologiaa. NFC-teknologian sanotaan muuttavan maailmaan. Se tekee mahdolliseksi, että kaikki lompakon kortit voidaan yhdistää yhteen korttiin: (U)SIM-korttiin. NFC:n myötä on mahdollista, että tulevaisuudessa käytetään matkapuhelinta julkisen liikenteen matkakorttina ja sillä voidaan maksaa ostokset myymälässä. Tämän tason sovellukset matkapuhelimessa vaativat korkeaa turvallisuutta. Sovellus on sijoitettava mahdollisimman turvalliseen ympäristöön matkapuhelimissa, jotta voidaan olla varmoja, etteivät sitä pääse käsittelemään väärät tahot. Kuvassa 7 on esitetty koko NFC-ympäristö.



Kuva 7. NFC-ympäristö [40, s. 6].

Tietoturvan NFC-sovelluksille antavat älykortit. Jokaisessa matkapuhelimessa on (U)SIM-korttina älykortti, joten sovelluksen sijoittaminen sinne tuntuu luontevalta. Älykorttien tekniikka varmistaa, että korttien tietoa ei pystytä peukaloimaan tai kopioimaan. Älykorttien korkea turvallisuus erottaa kaikki kortilla olevat sovellukset toisistaan, joten muut sovellukset eivät pääse käsittelemään toisille sovelluksille määrättyä muistia. NFC-sovelluksia kehittävien palveluntarjoajien kannalta on myös tärkeää, että teleoperaattorit omistavat (U)SIM-kortit. Näin niiden hallinnointi on helpompaa. Teleoperaattorit myöntävät palveluntarjoajille tietoturva-alueen (U)SIM-kortilta, johon sovellus voidaan sijoittaa, ja antavat niille henkilökohtaiset avaimet sen hallinnointia varten. TSM eli trusted services manager toimii yhteyspisteenä useampien palveluntarjoajien ja teleoperaattoreiden välissä, jotta näiden ei tarvitsisi tehdä jokaista varten erillisiä infrastruktuureja. TSM hoitaa palveluntarjoajan puolesta sovellusten toimittamisen kortille ja sen koko elinkaaren aikaisen hallinnoinnin.

NFC-sovellukset yleensä tahdotaan sijoittaa (U)SIM-kortille vasta kortin toimittamisen jälkeen. Tätä varten TSM käyttää OTA-tekniikkaa sovellusten lataamiseen ja asentamiseen (U)SIM-kortille ja hallinnointiin siellä. Lisäksi sitä käytetään tuomaan sovelluksille lisäarvoa. Sen avulla voidaan esimerkiksi ladata matkakorttiin lisää aikaa. OTA-tekniikat tarjoavat keinon kommunikointiin (U)SIM-kortin ja TSM:n taustajärjestelmän kanssa. OTA-tekniikoista kiinnostavin on BIP-protokolla ja siitä vielä tarkemmin CAT_TP-protokolla. CAT_TP-protokolla on luotettava tiedonsiirtoprotokolla, joka on suunniteltu varta vasten (U)SIM-kortin ja ulkoisen palvelimen väliseen tiedonsiirtoon. Siinä (U)SIM-kortti käyttää matkapuhelimen GPRS- tai 3G-yhteyttä yhteyskanavan luomiseen palvelimen kanssa.

Insinöörityön tilaaja Venyon toimii TSM:nä NFC-ekosysteemissä. Venyonin nykyinen (U)SIM-kortinhallintatoteutus käyttää (U)SIM-kortin hallinnointiin kuitenkin erillistä Java Midletiä matkapuhelimessa. Tämä lähestymistapa ei ole kaikkien teleoperaattoreiden suosima vaihtoehto (U)SIM-kortin kontrollointiin. Siksi Venyon on toteuttamassa itselleen CAT_TP-rajapinnan. Jotta CAT_TP-toteutusta voidaan pitää luotettavana, sen on täytettävä

täysin ETSI-järjestön määrittämä spesifikaatio. Venyonin insinöörityönä tilaama testausjärjestelmä on tätä tarkoitusta varten.

Testausjärjestelmä rakennettiin ETSI-järjestön CAT_TP-testausspesifikaation pohjalta. Testausjärjestelmän testit toteutettiin JUnit-testeinä, ja ne testaavat CAT_TP-toteutusta kaikilla mahdollisilla tavoilla. Testausjärjestelmä koostuu testattavasta CAT_TP-toteutuksesta ja testioperaattorista. CAT_TP-toteutuksen tulee määrittää testausjärjestelmän määritäämä rajapinta, jotta testaus voidaan suorittaa. Testausjärjestelmän avulla on jo pystytty kehittämään Venyonin CAT_TP-toteutusta paremmaksi.

NFC-teknologiasta on tehty jo useita pilotteja ympäri maailmaa. Sitä on käytetty ostosten ja joukkoliikenteen lippujen maksamiseen. Myös monet suuret yhtiöt kehittävät palveluita sitä varten. Kuitenkin NFC:tä vaivaa vielä yhteensopivien laitteiden vähäisyys markkinoilla. Laitteenvalmistajien pitää pikaisesti tarjota käyttäjille enemmän NFC-laitteita ja palveluntarjoajien käytännöllisiä palveluita. NFC:llä on vielä paljon esteitä voitettavana, jotta se hyväksytään käyttäjien keskuudessa. Uskotaan, että jo vuonna 2011 NFC olisi hyvin monella käyttäjällä käytössä. Itse pidän lukua liian optimistisena. Uskon, että käyttäjiltä vie aikaa sulattaa ajatus, että matkapuhelimella voi maksaa ostokset myymälöissä tai että se toimii matkakorttina. Uskon, että käyttäjät eivät vielä vuosiin luota saavansa suurempaa hyötyä NFC-matkapuhelimen käytöstä verrattuna perinteisiin luotto-, pankki- ja muihin älykortteihin. Kuitenkin ajan kuluessa, kun NFC-laitteet yleistyvät ja palveluita tulee saataville, NFC:tä luultavasti ruvetaan käyttämään enenevässä määrin.

Lähteet

1 NFC-technology. (WWW-dokumentti.) Sony Corporation. <http://www.sony.net/SonyInfo/News/Press_Archive/200312/03-059E/>. 8.12.2003. Luettu 10.2.2009.

2 Mobile NFC services. Version 1.0. (WWW-dokumentti.) GSMA. <http://www.gsmworld.com/documents/nfc_services_0207.pdf>. Helmikuu 2007. Luettu 2.2.2009.

3 Ghannam, Taoufik. How NFC can to speed Bluetooth transactions. (WWW-dokumentti.) Wireless Net DesignLine. <http://www.wirelessnetdesignline.com/howto/180201430>. 14.2.2004. Luettu 2.2.2009.

4 Near Field Communication Interface and Protocol (NFCIP-1). (WWW-dokumentti.) ECMA International. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-340.pdf>. 14.12.2004. Luettu 2.2.2009.

5 Near Field Communication White paper. (WWW-dokumentti.) ECMA International. <http://www.ecma-international.org/activities/Communications/2004tg19-001.pdf>. 2004. Luettu 2.2.2009.

6 RFID-teknologia. (WWW-dokumentti.) RFID Lab Finland. <http://www.rfidlab.fi/default.asp?1;2;800;0;61;&t=1&f=2&p=800&subp=800&subp0=800&did=61>. 21.8.2007. Luettu 6.1.2009.

7 NFC definition. (WWW-dokumentti.) Gemalto. <http://www.gemalto.com/nfc/definition.html>. Luettu 2.2.2009.

8 PN511 Transmission module. (WWW-dokumentti.) NXP. <http://www.nxp.com/acrobat_download/other/identification/082733.pdf>. 13.6.2007. Luettu 9.2.2009.

9 Difference between a NFC-enabled device and a NFC tag. (WWW-dokumentti.) NFC Forum. <http://www.nfc-forum.org/resources/faqs#tag>. Luettu 9.2.2009.

10 NFC-arkkitehtuuri. (WWW-dokumentti.) NFC-Forum. <http://www.nfc-forum.org/resources/white_papers/nfc_forum_marketing_white_paper.pdf/>. 31.10.2007. Luettu 10.2.2009.

11 Romen, Gerald. Near Field Communication Technology and the road ahead. (WWW-dokumentti.) NFC Forum & GSM Association. <http://www.nfc-

forum.org/resources/multimedia/NFC_Forum_14Feb07_Press_and_Analyst_Briefing_Slides.pdf>. 14.2.2007. Luettu 3.2.2009.

12 Requirements of ISO/IEC 14443 Type B Proximity Contactless Identification Cards. (WWW-dokumentti.) Atmel Corporation. <http://www.atmel.com/dyn/resources/prod_documents/doc2056.pdf>. Marraskuu 2005. Luettu 10.2.2009.

13 NFC IP-1. (WWW-dokumentti.) ECMA International. <http://www.ecma-international.org/publications/standards/Ecma-340.htm>. Joulukuu 2004. Luettu 10.2.2009.

14 NFC IP-2. (WWW-dokumentti.) ECMA International. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-352.pdf>. 7.5.2007. Luettu 10.2.2009.

15 Pay-Buy-Mobile Business Opportunity - Public White Paper Analysis. (WWW-dokumentti.) GSM Association. <http://www.gsmworld.com/documents/gsma_pbm_white_paper_11_2007.pdf>. Marraskuu 2007. Luettu 5.2.2009.

16 White paper - Near Field Communication. (WWW-dokumentti.) Nokia. <http://wiki.forum.nokia.com/images/6/6b/Nokia_NFC_white_paper.pdf>. 2007. Luettu 13.3.2009.

17 NFC-matkapuhelimet. (WWW-dokumentti.) NFC Research Lab. <http://www.nfc-research.at/index.php?id=45>. Luettu 14.3.2009.

18 NFC specifications. (WWW-dokumentti.) NFC Forum. <http://www.nfc-forum.org/specs/>. Luettu 9.2.2009.

19 NFC-tunnisteet. (WWW-dokumentti.) Top tunniste. <http://www.toptunniste.fi/index.php?id=128>. Luettu 9.2.2009.

20 Smart Cards. (WWW-dokumentti.) Smart Card Alliance. <http://www.smartcardalliance.org/pages/smart-cards-faq#what-is-a-smart-card>. Luettu 11.2.2009.

21 Hendry, Mike. Multi-application Smart Cards Technology and Applications. New York: Cambridge University Press, 2007.

22 Rankl, Wolfgang & Effing, Wolfgang. Smart Card Handbook 3rd ed. Munich: John Wiley & Sons, 2003.

23 Jun, Won J. Smart Card Technology Capabilities. (WWW-dokumentti.) Giesecke & Devrient. <http://csrc.nist.gov/publications/nistir/IR-7056/Capabilities/Jun-SmartCardTech.pdf>. 8.6.2003. Luettu 19.2.2009.

24 Global Platform. (WWW-dokumentti.) Global Platform Inc. <http://www.globalplatform.org/specifications.asp>. 2008. Luettu 15.3.2009.

25 Global Platform - Card specification version 2.2. (WWW-dokumentti.) Global Platform Inc. <http://www.globalplatform.org/specificationdownload.asp?id=6483>. Maaliskuu 2006. Luettu 15.3.2009.

26 Java Card™ & STK Applet Development Guidelines. (WWW-dokumentti.) Gemalto. <http://developer.gemalto.com/fileadmin/contrib/downloads/pdf/Java_Card_STK_Applet_Development_Guidelines.pdf >. Luettu 2.3.2209.

27 ETSI TS 151 014 - Digital cellular telecommunications system (Phase 2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (3GPP TS 51.014 version 4.5.0 Release 4). (WWW-dokumentti.) European Telecommunications Standards Institute. <http://pda.etsi.org/exchangefolder/ts_151014v040500p.pdf>. Joulukuu 2004. Luettu 15.3.2009.

28 ETSI TS 102 223 - Smart Cards; Card Application Toolkit (CAT) (Release 8, v8.2.0). (WWW-dokumentti.) European Telecommunications Standards Institute. <http://pda.etsi.org/exchangefolder/ts_102223v080200p.pdf>. Tammikuu 2009. Luettu 15.3.2009.

29 OTA White Paper - Using Over-The-Air Technology to Remotely Manage Value Added Services. (WWW-dokumentti.) Gemplus. <http://www.gemalto.com/dwnld/1839_ota_white_paper_nov2001.zip>. 14.11.2004. Luettu 18.3.2009.

30 Bearer Independent protocol (BIP). (WWW-dokumentti.) Giesecke & Devrient. <http://www.gdai.com/pls/portal/BIP_Whitepaper_final.pdf>. 2006. Luettu 4.3.2009.

31 ETSI TS 102 226 - Smart Cards; Remote APDU structure for UICC based applications (Release 7) v7.5.0. (WWW-dokumentti.) European Telecommunications Standards Institute. <http://pda.etsi.org/exchangefolder/ts_102226v080000p.pdf>. Lokakuu 2007. Luettu 4.3.2009.

32 Interoperability Stepping Stonen (Release 6 v1.2.0). (WWW-dokumentti.) SIM Alliance. <http://www.simalliance.org/servlets/sfs?t=/documentManager/sfdoc.file.supply&e=UTF-8&i=1185787014303&l=0&fileID=1194014148577>. 20.2.2007. Luettu 18.3.2009.

33 ETSI TS 102 127 - Smart Cards; Transport protocol for CAT applications; Stage 2 (Release 6) v.6.12.0. (WWW-dokumentti.) European Telecommunications Standards

Institute. <http://pda.etsi.org/exchangefolder/ts_102127v061200p.pdf>. Syyskuu 2008. Luettu 11.3.2009.

34 UDP vs TCP. (WWW-dokumentti.) Devmaster.net. <http://www.devmaster.net/wiki/UDP_vs_TCP>. 26.2.2008. Luettu 11.3.2009.

35 Watkins, John. Testing IT: An Off-the-Shelf Software Testing Handbook. USA: Cambridge University Press, 2000.

36 Myers, G., Sandler, C., Badgett, T. Art of Software Testing 2nd ed. New Jersey: John Wiley & Sons Inc, 2004.

37 JUnit. (WWW-dokumentti.) Sourceforge. <http://sourceforge.net/projects/junit/>. Luettu 11.3.2009.

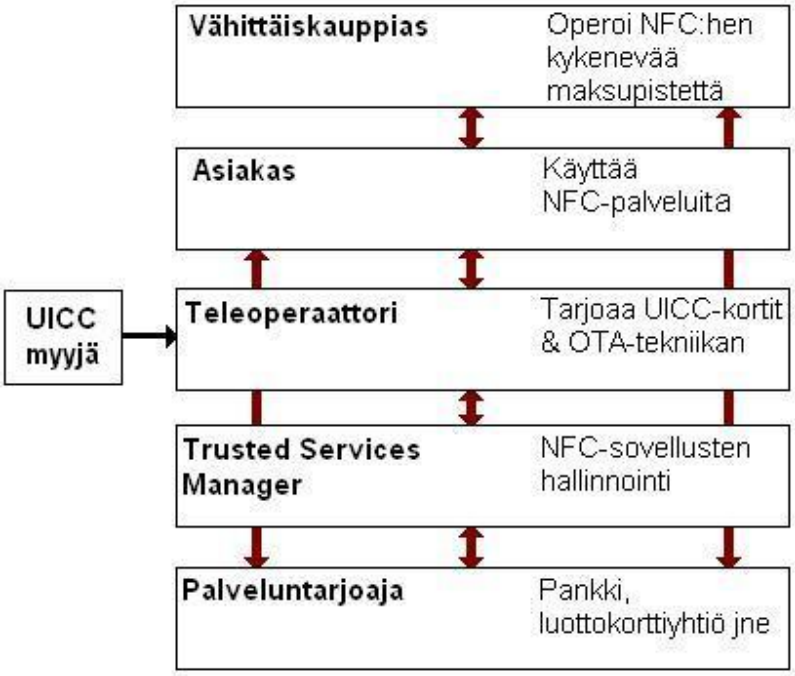
38 Beck, Kent. JUnit pocket guide. USA: O'Reilly Media, 2005.

39 Org.junit.assert-luokka. (WWW-dokumentti.) JUnit. <http://junit.org/apidocs/org/junit/Assert.html>. Luettu 20.3.2009.

40 Nordlund, Sirpa. Secure Over-The-Air Services in NFC Ecosystems. (WWW-dokumentti.) Venyon Oy. <www.nfc-research.at/fileadmin/congress/2007/slides/05_Venyon_Sirpa_Nordlund.pdf>. 20.3.2007. Luettu 21.3.2009.

41 ETSI TS 102 431 - Smart Cards; Test specification for the Transport Protocol of CAT Applications (CAT_TP) validation (Release 7) v.7.1.0. European Telecommunications Standards Institute. (WWW-dokumentti.) <http://pda.etsi.org/exchangefolder/ts_102431v070100p.pdf>. Maaliskuu 2008. Luettu 11.3.2009.

Liite 1: NFC-entiteettien väliset roolit



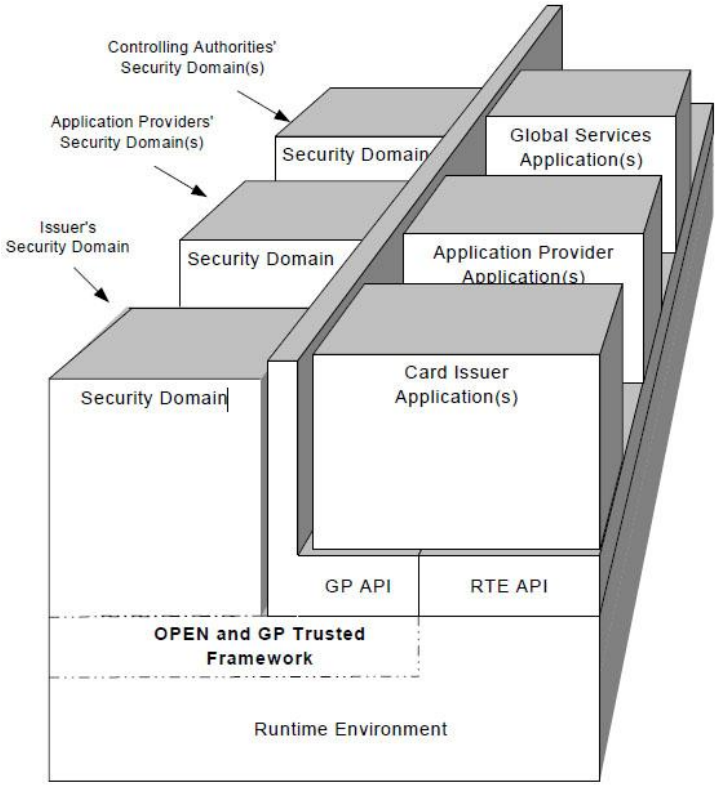
Kuva 1. Mobiilimaksamiseen liittyvien NFC-entiteettien suhteet [2, s. 14].

Liite 2: Langattomat yhteysteknologiat

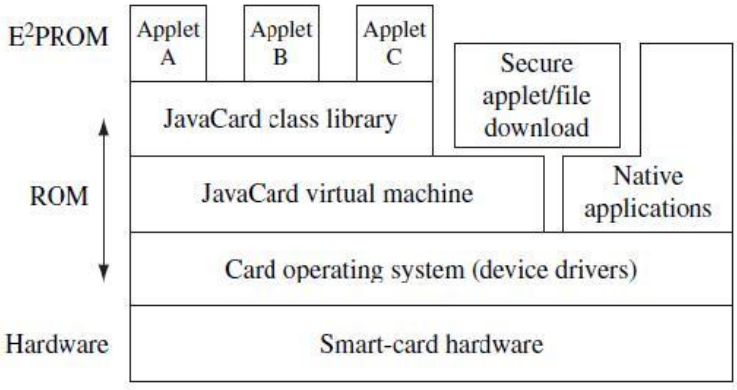
Taulukko 1. Langattomien yhteysteknologioiden eroja [11, s. 49].

	NFC	**RFID**	**IrDA**	**Bluetooth**
Yhteyden alustusaika	<0.1ms	<0.1ms	~0.5s	~6s
Kantomatka	~10cm	~3m	~5m	~30m
Käytettävyys	Käyttäjäkeskeinen, nopea, helppo, vaistonvarainen	Laitekeskeinen, helppo	Datakeskeinen, helppo	Datakeskeinen, keskinkertainen
Valikoivuus	Korkea, sovittu, turvallinen	Osittain sovittu	Se, joka näköakselilla	Ei varmuutta yhteyden vastapuolesta
Käyttökohteita	Maksaminen, pääsyn oikeuttaminen, tiedonvaihto, toisten yhteyksien alustaminen	Esineiden seuraus	Kontrollointi, tiedonvaihto	Tiedonvaihto
Käyttäjä kokemukset	Helppo yhteyden alustaminen pelkällä kosketuksella	Informaation saanti	Helppo	Tarvitsee manuaalista yhteyden alustamista

Liite 3: Global Platform- ja Java Card -arkkitehtuurit

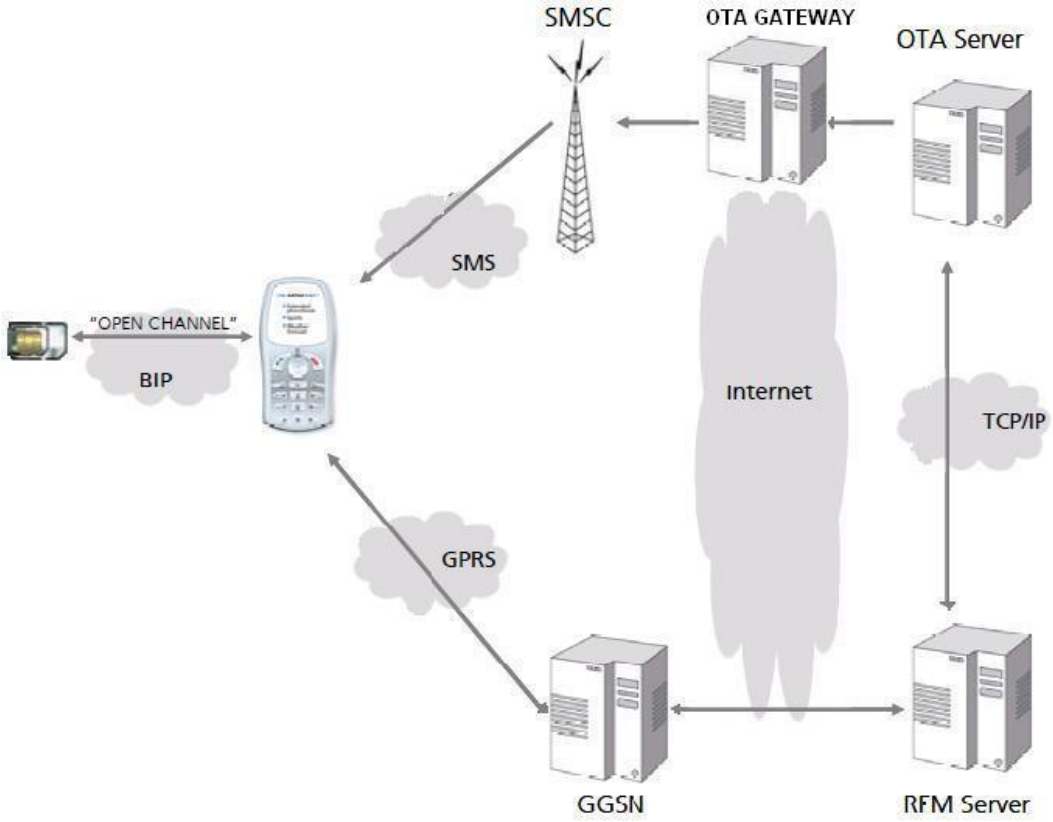


Kuva 1. Global Platform -arkkitehtuuri [26, s. 18].



Kuva 2. Java Card -älykortin arkkitehtuuri [22, s. 77].

Liite 4: OTA-arkkitehtuuri



Kuva 1. OTA-arkkitehtuuri [31, s. 13; 30, s. 5].

Liite 5: RFM- ja RAM-komennot

Taulukko 1. RFM-kommennot [32, s. 14].

Komennot
SELECT
UPDATE BINARY
UPDATE RECORD
SEARCH RECORD
INCREASE
VERIFY PIN
CHANGE PIN
DISABLE PIN
ENABLE PIN
UNBLOCK PIN
DEACTIVATE FILE
ACTIVATE FILE
READ BINARY
READ RECORD
CREATE FILE
DELETE FILE
RESIZE FILE

Taulukko 2. RAM-kommennot [32, s. 16].

Komennot
DELETE
SET STATUS
INSTALL
LOAD
PUT KEY
GET STATUS
GET DATA

Liite 6: RST PDU -paketin mahdolliset syykoodit

Taulukko 1. RST PDU -pakettien syykoodit [34, s. 52].

Syykoodi	**Syy**
00	Normaali päättyminen
01	Yhteyden alustaminen epäonnistui, virheellisiä parametreja
02	Väliaikaisesti kykenemätön alustamaan yhteyttä
03	Kysytty portti tavoittamattomissa
04	Aavistamaton paketti vastaanotettu
05	Maksimi määrä uudelleen lähetyksiä saavutettu
06	Versionumero ei ole tuettu
07	Jätetty mahdollista myöhempää käyttöä varten (RFU)

Liite 7: Testattavat toiminnallisuudet

Taulukko 1. Testattavat toiminnallisuudet [42, s. 16].

Toiminnallisuus	**Pakollisuus**
Tarkistussumma	Kyllä
Uudelleen lähetysten laskuri	Kyllä
Virheellisten PDU-pakettien havaitseminen ja hylkääminen	Kyllä
Vastaanotetun PDU-paketin tyypin käsittely	Kyllä
Maksimi PDU-paketin koon käsittely	Kyllä
Maksimi SDU-paketin koon käsittely	Kyllä
Kuittausnumeron käsittely	Kyllä
Sekvenssinumeroiden käsittely	Kyllä
Sekvenssinumeron asettaminen	Kyllä
CAT_TP-linkin uniikki identifiointi (Samaa porttia ei jaeta kahden CAT_TP-yhteyden kesken)	Katso kommentti 1
Epäjärjestyksessä saapuneiden pakettien käsittely	Katso kommentti 2
Kuittausten käsittely	Kyllä
Jokaisen kuittaamattoman paketin uudelleen lähetys	Kyllä
CAT_TP-linkin sulkeminen, mihin aikaan vaan	Kyllä
SDU-paketin toimittaminen järjestyksessä	Kyllä
Pakettien järjestäminen	Kyllä
Lähetettävien SDU-pakettien segmentointi	Katso kommentti 3
Kaksisuuntainen datan vaihto	Kyllä
ID-tunnisteen oikeellinen muodostaminen	Kyllä
Portin käsittely aktiivisessa avauspyynnössä	Katso kommentti 4
Ikkunan koon käsittely	Kyllä
Ylemmän kerroksen pyyntöjen käsittely	Kyllä
Kommentti 1: CAT_TP-toteutus tukee useamman yhtäaikaisen yhteyden muodostamista. **Kommentti 2:** CAT_TP-toteutus tukee suurempaa ikkunan kokoa kuin 1. **Kommentti 3:** CAT_TP-toteutus tukee lähetettävien pakettien segmentointia. **Kommentti 4:** CAT_TP-toteutus tukee lokaalin portin määrittämistä aktiivisessa avauspyynnössä.	

Liite 8: CAT_TP-testaujärjestelmän määrittämä rajapinta

```
public interface CatTpTestAdapter {
      // CAT_TP-yhteyden avauspyyntö
      void open(boolean passive, int localPort, int destPort,int maxPduSize,
                int maxSduSize, byte[] id) throws CatTpException;
      // CAT_TP-yhteyden sulkemispyyntö
      void close() throws CatTpException;
      // Lähettää dataa
      void send(byte[] data) throws CatTpException;
      // Lähettää NUL PDU
      void sendNUL() throws CatTpException;
      // Palauttaa ylemmälle tasolle vastaanotetun SDU
      byte[] receive() throws CatTpException;

      // Sulkee ja nollaa CAT_TP-yhteyden
      void setClosed();
      // Luo CAT_TP-objektin
      void setOpen(int CE_PORT);
      // Palauttaa CAT_TP-entiteetin nykyisen tilan
      CatTpStatus getCurrentState();
      // Antaa PDU:n CAT_TP-entiteetille käsiteltäväksi
       void processPDU(byte[] pdu) throws CatTpException;
      // Rekisteröi ID-tunnisteen
      void registerConnection(byte[] id);
      // Seuraavat palauttaa nimen mukaiset muuttujat
      int getSndIniSeqNb();
      int getSndWinSize();
      int getRcvWinSize();
      int getRcvSduSizeMax();
      int getRcvPduSizeMax();
      int getSndSduSizeMax();
      int getSndPduSizeMax();
      int getRetries();
      byte[] getID();
      //Palauttaa serverin jota CAT_TP-entiteetti käyttää
       Server getServer();
}
```

Kuva 1. CAT_TP-testausjärjestelmän määrittämä rajapinta

Liite 9: Testi tavallisen CAT_TP-yhteyden passiiviseen avaamiseen

```
/**
 *  Testing reception of SYN PDU (ETSI 102431 - 7.2.1.1 [page 40])
 */
public void testRcvSyn(){
    log.logInfo("Testing reception of SYN PDU [7.2.1.1]");
    assertEquals("Verifying that CE is in LISTEN state","ListenState", CE.getCurrentState());
    PduGenerator pdugen = new PduGenerator();
    BasePDU pdu = pdugen.SYN_PDU(CS_PORT, CE_PORT, SEQ, WIN_SIZE, MAX_PDU_SIZE, MAX_SDU_SIZE, id);
    assertTrue("Comfirming that CS SYN PDU is valid.",pdu.isValid());
    try{
        log.logPduMovement(pdu,  CE.getCurrentState(), false);
        CE.processPDU(pdu.getBuffer()); //<-- Gives the SYN PDU for CE to process

        BasePDU synack = waitPDU(0); //<-- Waits for the CE response(SYN-ACK PDU)
        log.logPduMovement(synack, CE.getCurrentState(),true);
        assertTrue("Verifying that CEs SYN-ACK PDU is valid.",synack.isValid());
        assertTrue("Verifying that CEs SYN-ACK PDU is SYN set.",synack.isSYNSet());
        assertTrue("Verifying that CEs SYN-ACK PDU isACK set.",synack.isACKSet());
        assertEquals("SYN-ACK sequence number ",CE_SEQ, synack.getSEQ());
        assertEquals("Verifying that CE is in SYNRCVD state after having issued the SYN-ACK PDU",
                "SynRcvState", CE.getCurrentState());

        pdu = pdugen.ACK_PDU(CS_PORT,CE_PORT, (short)(SEQ+1),CE_SEQ, WIN_SIZE);
        assertTrue("Verifying that CSs ACK PDU is valid.",pdu.isValid());
        CE.processPDU(pdu.getBuffer()); //<-- Gives the ACK PDU for CE to process
        log.logPduMovement(pdu,  CE.getCurrentState(), false);

        assertEquals("Verifying that CE is in OPEN state","OpenState", CE.getCurrentState());

        BasePDU nulpdu = pdugen.NUL_ACK_PDU(CS_PORT,CE_PORT, (short)(SEQ+1),CE_SEQ, WIN_SIZE);
        assertTrue("Comfirming that CS NUL PDU is valid.",nulpdu.isValid());
        log.logPduMovement(nulpdu,  CE.getCurrentState(), false);
        CE.processPDU(nulpdu.getBuffer());

        BasePDU ceAck = waitPDU(1);
        log.logPduMovement(ceAck, CE.getCurrentState(),true);
        assertTrue("Comfirming that CE response PDU is valid.",ceAck.isValid());
        assertFalse("Comfirming that CE response PDU is ACK PDU.",ceAck.isSYNSet());
        assertTrue("Comfirming that CE response PDU is ACK PDU.",ceAck.isACKSet());
        assertEquals("Checking that CS NUL PDU is correctly acknowledged",SEQ+1, ceAck.getAckNum());

    }catch(Exception e){fail("Test case failed,caught exception: "+e.getMessage());}
    assertEquals("Verifying that CE is in OPEN state",
            "OpenState", CE.getCurrentState());
    log.logInfo("END OF: Testing reception of SYN PDU [7.2.1.1]");
}
```

Kuva 1. Testi tavallisen CAT_TP-yhteyden passiiviseen avaamiseen.

Liite 10: Testi Send-komennon vastaanottamisesta Syn-Rcvd-tilassa

```
/**
 *  Testing Send request (ETSI 102431 - 7.2.2.8 [page 66])
 *  To verify that CE does not take into account the send request
 *  and stays in its current state.
 */
public void testSendReq(){
    log.logInfo("Testing Send request [7.2.2.8]");
    assertEquals("Verifying that CE is in SYNRCVD state after having issued the SYN-ACK PDU",
            "SynRcvState",CE.getCurrentState());
    try{
        CE.send(new byte[4]);
        fail("Exception expected from send request");
    }catch(CatException e){
        assertEquals("Expecting CONNECTION NOT OPEN exception",3,e.getReason());
    }
    assertEquals("Verifying that CE stays in its current state",
            "SynRcvState", CE.getCurrentState());
    log.logInfo("END OF: Testing Send request [7.2.2.8]");
}
```

Kuva 1. Testi Send-komennon vastaanottamisesta Syn-Rcvd-tilassa.

Liite 11: Testi yhteyden sulkemisesta Open-tilassa, kun vastapuoli on ilmoittanut ikkunan kooksi nollan

```
public void testCloseReq2(){
    log.logInfo("Testing CLOSE request in the 'OPEN' state - CS's window size = 0 [9.1.13]");
    assertEquals("Verifying that CE is in OPEN state after having issued the SYN-ACK PDU and receiv
            "OpenState", CE.getCurrentState());
    try{
        CE.send(new byte[]{(byte)99,(byte)88});

        BasePDU ack = waitPDU(1);
        log.logPduMovement(ack,  CE.getCurrentState(), true);
        assertTrue("Comfirming that CEs ACK-UNSEG-DATA PDU is valid.",ack.isValid());
        assertTrue("Comfirming that CEs ACK-UNSEG-DATA PDU is ACK set.",ack.isACKSet());
        assertFalse("Comfirming that CEs ACK-UNSEG-DATA PDU is not SEG set.",ack.isSEGSet());
        assertEquals("Sequence number",CE_SEQ+1, ack.getSEQ());
        assertEquals("Acknowledgement number",SEQ, ack.getAckNum());
        assertEquals("Verifying that CE is in OPEN state","OpenState", CE.getCurrentState());

        PduGenerator pdugen = new PduGenerator();
        BasePDU pdu = pdugen.ACK_UNSEG_DATA_PDU(CS_PORT,CE_PORT, (short)(SEQ+1),(short)(CE_SEQ+1),
                (short)0,new byte[]{(byte)01,(byte)23});
        assertTrue("Comfirming that CS ACK-UNSEG-DATA PDU is valid.",pdu.isValid());
        log.logPduMovement(pdu,  CE.getCurrentState(), false);
        CE.processPDU(pdu.getBuffer());

        BasePDU ack2 = waitPDU(2);
        log.logPduMovement(ack2,  CE.getCurrentState(), true);
        assertTrue("Comfirming that CEs ACK PDU is valid.",ack2.isValid());
        assertTrue("Comfirming that CEs ACK PDU is ACK set.",ack2.isACKSet());
        assertEquals("Sequence number",CE_SEQ+2, ack2.getSEQ());
        assertEquals("Acknowledgement number",SEQ+1, ack2.getAckNum());
        assertEquals("Verifying that CE is in OPEN state","OpenState", CE.getCurrentState());
    }catch(CatException e){
        fail("Sending data failed.");
    }

    try{
        CE.close();

        BasePDU rst = waitPDU(3);
        log.logPduMovement(rst,  CE.getCurrentState(), true);
        assertTrue("Comfirming that CEs RST PDU is valid.",rst.isValid());
        assertTrue("Comfirming that CEs ACK PDU is RST set.",rst.isRSTSet());
        assertEquals("Sequence number",CE_SEQ+2, rst.getSEQ());
        assertEquals("Reason code",00, rst.getReason());
        assertEquals("Verifying that CE is in Close-Wait state","CloseWaitState", CE.getCurrentState());

        CE.close();
        fail("Exception expected from close request");
    }catch(CatException e){
         assertEquals("Verifying that CE signals 'CONNECTION_CLOSING'",1, e.getReason());
    }
    assertEquals("Verifying that CE is in Close-Wait state","CloseWaitState", CE.getCurrentState());
    log.logInfo("END OF: Testing CLOSE request in the 'OPEN' state - CS's window size = 0 [9.1.13]");
}
```

Kuva 1. Testi yhteyden sulkemisesta Open-tilassa, kun vastapuoli on ilmoittanut ikkunan kooksi nollan.