

# Lausunto: Ohjelmistojen samankaltaisuus, VF Partner vs. System

24.2.1997

Prof. Ilkka Haikala,

TTKK/Ohjelmistotekniikka

PL 553

33101 Tampere

(03)3652911

ijh@cs.tut.fi

Olen jo aikaisemmin ilmaissut käsitykseni VF Partnerin ja Systemin välisestä maksupääteohjelman mahdollista kopiointia koskevasta kiistasta (lausunto 8.12.1996). Asianajaja Miika Pellolta saamani tiedon mukaan kiistassa on sittemmin tuotu esiin muutamia uusia kopioituksi väitettyjä ohjelmarivejä. Aikaisemmin antamassani lausunnossa kävin kopioituksi väitettyjä rivejä läpi, ja totesin, että mielestäni kopiointia ei ole tapahtunut. Miika Pellon pyynnöstä ryhdyn vielä kerran vertailemaan em. ohjelmankohdista, tällä kertaa perusteellisemmin kuin aikaisemmin -- ja toivottavasti nyt myös viimeisen kerran.

## 1. Vertailun lähtökohdat

Arvioitavakseni on toimitettu luettelo, jossa on noin 40 samankaltaiseksi väitettyä muistipaikkaa. Kuten jo VTT:n lausunnossa on todettu (22.12.1994, Unto Pulkkinen), nämä edustavat vain pientä osaa koko ohjelmistosta (Pulkkinen mukaan n. 10%). Koska VF Partnerin ohjelman ovat kirjoittaneet samat ohjelmoijat kuin Systemin aikaisemman ohjelman, ei samantapaisten ohjelmarivien määrä ole mielestäni tavanomaisuudesta poikkeava. Syitä samankaltaisuudelle olen esittänyt edellisessä lausunnossani.

Käytin uuteen selvitystyöhön noin kaksi työpäivää, yhteensä noin 20 tuntia. Tänä aikana ehdin käydä kopioituksi väitetyt ohjelman osat käsky käskyltä lähes kokonaan läpi. Tarkastelun ulkopuolelle jäi vain muutamia ns. PATU-standardiin liittyviä rivejä, joihin aikani ei riittänyt. Tarkasteluni ei myöskään kata saamani listan ulkopuolelle jääviä muistipaikkoja.

Vaikka ohjelmakoodi on periaatteessa yksinkertaista, on se käytetyn kryptisen TCL-ohjelmointikielen ja ohjelmien puutteellisen kommentoinnin takia vaikeaselkoista. Toisaalta käytettävissä ollut työaika oli tehtävään nähden verraten lyhyt. Tästä syystä seuraavassa esityksessä on mahdollisesti muutamia pikku virheitä, joiden poistamiseksi ohjelmakoodi olisi pitänyt käydä vielä toistamiseen läpi. Tällaisilla lipsahduksilla ei kuitenkaan ole kokonaisuuden kannalta merkitystä.

Olen seuraavassa yrittänyt ryhmitellä samaan kokonaisuuteen kuuluvia muistipaikkoja yhteen. Kunkin kohdan alussa on taulukko, josta selviävät käsitellyt muistipaikat. Muistipaikkojen numeroinnissa käytän Systemin ohjelmalistauksen numerointia.

Oheiset kuvaukset saattavat paikoitellen vaikuttaa tarpeettoman yksityiskohtaisilta. Tein ne lähinnä itseäni ajatellen siltä varalta, että asiaan joudutaan vielä palaamaan. Ilman muistiinpanoja asiaan palaaminen parin kuukauden päästä vaatisi jälleen muutaman päivän työn.

## 2. Sanoman numerojen pakkaus

VF	436	Systemek	717
VF	435	Systemek	718
VF	434	Systemek	885
VF	432	Systemek	921

Pankkiyhteydellä käytettävässä numeroiden esityksessä käytetään ns. pakattua BCD-esitystä. Em. ohjelmarivit muuttavat sanoman tähän muotoon.

Ohjelman logiikka on erittäin yksinkertainen. Sanoma käydään läpi, ja aina kaksi perättäistä merkkiä yhdistetään yhdeksi merkiksi laskemalla ne yhteen kaavalla  $\text{merkki1} * 16 + \text{merkki2}$ . Näin 16 bitin sijasta kaksi peräkkäistä merkkiä saadaan esitettyä 8 bitillä. Tapa, jolla koodaus tehdään, on määritelty Maksupäättevarmennuksen standardissa kohdassa L6. Koodia monimutkaistaa jonkin verran standardin edellyttämien täytemerkkien lisääminen ja muutamien standardissa esitettyjen erikoistilanteiden ottaminen huomioon (mm. merkien # ja & käyttö).

## 3. Sanoman pituuden lasku

VF	424	Systemek	884
----	-----	----------	-----

Ohjelmakoodissa lisätään pankkiyhteysosan alkuun sen pituus. Laskutapa on määritelty standardissa sivulla L6-2 (kohta 1.2). Tämä kohta on malliesimerkki kohdasta, joka muodostuu ohjelmointiympäristön antamien puitteiden ja pankkiyhteystandardin määräyksistä johtuen pakostakin samankaltaiseksi. Jopa kaksi eri ohjelmoijaa tuottaisi todennäköisesti lähes saman koodin, puhumattakaan saman ohjelmoijan uudelleen toteuttamasta koodista.

## 4. Kermitin CRC:n laskeminen

VF	475	Systemek	874
VF	474	Systemek	329
VF	490	Systemek	875
VF	489	Systemek	343
VF	427	Systemek	918
VF	426	Systemek	882
VF	425	Systemek	363

Tässä lasketaan tavanomaisella tavalla Kermit-protokollan määrittelemä tarkistussumma, joka on löydettävissä Kermit-protokollan määrittelystä. Algoritmista käydään sanoma läpi ja summataan sen tavut yhteen. Lopussa summa lyhennetään 8-bittisestä 6-bittiseksi summaamalla kaksi ylintä bittiä tarkistussummaan ja nollaamalla yläpään bitit (Kermitin tarkistussumma on 6-bittinen, tavussa on 8 bittiä). Sama algoritmi on koodattu kolmeen eri kohtaan, ensin rivit 874 ja 329, sitten 875 ja 343 ja lopulta rivit 918, 882 ja 363.

## 5. Kieltoluettelotarkistusosoite

VF	464	System	313
VF	458	System	922
VF	457	System	379
VF	447	System	715
VF	423	System	390
VF	446	System	924
VF	430	System	925
VF	596	System	932
VF	597	System	935
VF	595	System	936

Tarkistusosoite koostuu verraten pitkstä suoritusketjusta, johon osallistuvista muistipaikoista osaa väitetään kopioituiksi.

Muistipaikassa 464 rakennetaan varmistussanomien bittikarttaa (64 bittiä), joka on vakioitu ja määritelty pankkistandardissa. Tästä syystä koodi ei oikeastaan voi olla kovin erilaista, koska bittikartan kokoaminen on mekaaninen tehtävä. Tämä koskee varsinkin 32 ensimmäistä bittiä. Viimeisten 32 osalta bitin ohjelmat poikkeavat toisistaan yllättävänkin paljon.

Muistipaikassa 922 lisätään sanomaan kenttiä, edelleen pankkistandardin mukaisesti. Koodi on mielestäni standardin vaatimukset huomioon ottaen erilaista, ei samanlaista.

Muistipaikassa 379 sanomaan lisätään pankkistandardin vaatimia vakioituja kenttiä. Muita osin koodi on erilaista.

Muistipaikassa 715 sanomaan lisätään alku ja loppumerkit ja se lähetetään. Tämän jälkeen odotetaan linjalta vastaukseksi SOH-merkkiä. Kun SOH on saatu, otetaan vastaan vastaussanomien merkit merkkiin CR asti. Toiminta on pankkistandardin määrittelemä. Koodinpätkät muistuttavat toisiaan rakenteellisesti. Kyseessä on kuitenkin aivan tavallinen rutiinikoodi.

Muistipaikassa 390 olevassa aliohjelmassa puretaan vastaussanomien mahdollisesti sisältämät toistomerkit. Pankkistandardin mukaan (sivu L6-3) saman merkin toistuessa useaan kertaan se voidaan (sanomien lyhentämiseksi) esittää muodossa ~nX, missä n on tietyllä tavalla koodattu toistojen määrä ja X on toistettava merkki. Siis esimerkiksi AAAA voidaan esittää muodossa ~4A (koodaustavasta johtuen luku 4 on tosin muodossa 4+32, eli siis 36, jolloin sanoma näyttää sisältävän merkkijonon ~\$A). Ohjelma käy vastaussanomien läpi merkki merkiltä ja löytäessään merkin ~ korvaa merkkijonon ~nX vastaavalla määrällä merkkejä X. Molempien ohjelmien koodi on tässä paikassa samanlaista. Aliohjelma vastaa vaikeustasoltaan korkeakoulun ensimmäisen ohjelmointikurssin helpohkoja kotitehtäviä.

Muistipaikassa 924 oleva koodi purkaa vastaussanomien. Sanomien keskeltä etsitään merkkijono 'TR00' (tai VP:n koodissa VPF), joka on maksupäätteen tunnus. Pankkistandardin sivulla L1-1 on kuvattu sanomien muoto: TR00:n edellä on kahden merkin koodi, joka kertoo, onko varmennus kunnossa ja vielä senkin edellä on 6-merkin varmistuskoodi, joka on otettava talteen (ja näytettävä myös päätteen näytöllä, se tosin tehdään vasta myöhemmin). Ohjelmakoodi on noin ensimmäisen puoliskon osalta samankaltainen (koodin ja varmistuskoodin talteenotto), loppuosa on erilainen.

Muistipaikassa 925 tutkitaan, onko epäonnistunutta varmistusyritystä yritetty jo uudelleen (ohjelma yrittää maksimissaan kolme kertaa). Laskuria vähennetään joka kerta yhdellä, ja jos se nolautuu, generoidaan käyttäjälle ilmoitus. Muussa tapauksessa muodostetaan hieman erimuotoinen sanoma kuin ensimmäisellä kerralla ja aloitetaan koko kysely pankista uudelleen. Sanomien muodon määrää pankkistandardi (sivu L1-1). Ohjelma koodi on suoraviivaista ja molemmissa ohjelmissa lähes samanlaista. Samalaisuuteen ohjaa myös halu säilyttää ohjelman käyttöliittymä muuttumattomana.

Muistipaikat 932, 935 ja 936 muodostavat kokonaisuuden, jossa päätteen käyttäjälle kerrotaan varmistuksen epäonnistumisen syy. Syykoodit on lueteltu pankkistandardissa sivulla L2-2. Koodit käydään läpi yksi kerrallaan, ja koodista riippuen muodostetaan sopiva tulostusrivi. Muistipaikoissa 932 ja 935 ohjelma koostuu toisiaan seuraavista vertailuista: jos koodi = '14' niin tulostus= "tuntematon", jos taas koodi on '02' niin tulostus= .... eli käydään kaikki pankkistandardin antamat vaihtoehdot läpi. Muistipaikassa 936 tulostetaan ilmoitus maksupäätteelle. Kaikkien muistipaikkojen osalta koodi on hyvin samankaltaista. Muistipaikkojen 932 ja 935 osalta tämä johtunee pankkistandardista ja muistipaikan 936

osalta myös halusta pitää päätteen käyttöliittymä molemmissa ohjelmissa samanlaisena.

## 6. Kermitin init-paketin käsittely

VF	492	System	869
VF	491	System	344
VF	485	System	871
VF	487	System	870

Muistipaikat 869 ja 344 muodostavat rutiinin, joka lähettää init-paketin. 869 on vain 2 käskyn mittainen alkuosa. Ensin rakennetaan kermit-protokollan määräämä init-paketti, yhteensä kymmenkunta tavua. Tämän jälkeen lasketaan CRC (tarkistussumma, mp 875) paketin loppuun ja lisätään sen alkuun SOH-merkki ja loppuun CR-merkki. Tämän jälkeen sanoma lähetetään ja otetaan pankin vastaus muistipaikan 870 rutiinilla. Jos vastaus ei ole ok, yritetään uudelleen maksimissaan 5 kertaa. Rutiini on alkuosaltaan (sanoman muodostaminen) samanlainen molemmissa ohjelmissa. Loppuosat poikkeavat toisistaan. Kermit-protokolla määrää paketin sisällön.

Muistipaikka 870 sisältää rutiinin, jossa otetaan vastaan init-sanoman vastaussanoma. Vastaussanoma on samanmuotoinen kuin init-sanoma, ohjelma tosin tarkastaa siitä tässä vain, että se alkaa SOH-merkillä, lopussa on CR-merkki ja tarkistussumma on oikein. Rutiini jatkuu tästä muistipaikkaan 342, jossa tarkastetaan sanoman juokseva sanomanumero (en tarkastanut, koska muistipaikkaa ei ole lueteltu samankaltaisten listalla). Molemmissa ohjelmissa on tässä samankaltaisuutta.

Muistipaikka 871 on vain kahden käskyn mittainen ja muistuttaa muistipaikan 869 käyttöä muistipaikan 344 rutiinin alkuosana. Otin se esille tässä kohdassa, koska myös sieltä tullaan muistipaikan 345 kautta muistipaikan 870 rutiiniin.

## 7. Sanomanumeron käsittely

VF	483	System	876
VF	482	System	877

Kermit-protokollan sanomat numeroidaan siirtoyhteydellä häviävien pakettien havaitsemiseksi. Numerointi juoksee arvosta 32 arvoon 95. Kun paketti numerolla 95 on lähetetty, aloitetaan numerointi uudelleen arvosta 32. Numeron kasvatus tapahtuu muistipaikassa 876 olevalla koodilla. Joidenkin virhetilanteiden yhteydessä numeroa joudutaan vähentämään. Vähentävä koodi on muistipaikassa 877. Koodi on sama molemmissa ohjelmissa. Tässä tapauksessa laskukaava on kuitenkin niin yksinkertainen, että ei olisi hämmästyttävää, vaikka kaksi eri ohjelmoijaa kirjoittaisi saman ohjelmakoodin.

Tämä kohta on samalla hyvä esimerkki siitä, miten sama ohjelmakohta kirjoitetaan suurella todennäköisyydellä samalla tavalla. Kehittyneemmällä ohjelmointikielellä ohjelmoija kirjoittaisi yhden ainoan käskyn, esimerkiksi

```
sanomanumero := (sanomanumero + 1) mod 64 + 32.
```

Tässä mod on jakojäännöksen antava operaattori. Toinen vaihtoehto olisi käyttää ehdollista rakennetta, jota on sovellettu muistipaikoissa 876 ja 877:

```
sanomanumero := sanomanumero+1  
if sanomanumero = 96 then sanomanumero := 32
```

Muutamia muitakin vaihtoehtoja on, mutta todennäköisyys, että valitaan jompi kumpi ylläolevista on erittäin suuri, vaikka kyseessä olisi kaksi eri ohjelmoijaa. Ohjelmoijalla on yleensä suosikkitaapansa, jota hän aina käyttää.

## 8. Soitto modeemilinjalle

VF	499	Systemek	863
VF	498	Systemek	923

Muistipaikoissa 499 ja 498 on rutiini, jolla yritetään soittaa puhelimella pankin tietokoneelle. Rutiinin alussa on testi, jonka avulla testataan, ettei soittoyrityksiä tehdä loputtomasti, jos pankki ei vastaa. Tämän jälkeen tutkitaan linjan tila. Jos puhelin ei ole käytössä ja linja on kunnossa, yritetään soittaa pankkiin. Vastausta odotellaan silmukassa. Kun yhteys on saatu, palataan kutsuneeseen ohjelmaan.

Koodissa on samankaltaisuutta. Samankaltaisuus selittyy mm. laitteiston edellyttämiltä toiminnoilta ja käyttöliittymän samankaltaisuudesta.

## 9. Tulostusta kirjoittimelle

VF	855	Systemek	280
----	-----	----------	-----

Muistipaikan 280 koodinpätkä on osa laajempaa tulostuskokonaisuutta. Koodi on täysin sarjallista, ja siinä toistuu rakenne: vie kentän nimenä toimiva vakioteksti tulostuspuskuriin (esimerkiksi "KASSA: ") - - vie kentän arvo puskuriin -- tulosta. Kun tulosteet ja käyttöliittymä on haluttu säilyttää samoina, on lopputuloksena lähes identtinen ohjelmakoodi.

## 10. Tietojen talletus tiedostoon 10

VF	825	Systemek	633
----	-----	----------	-----

Muistipaikan 633 koodinpätkässä muodostetaan tiedostoon talletettavaa tietuetta. Lopuksi tietue lisätään tiedostoon. Kun molemmissa ohjelmissa viedään samat tiedot, muistuttavat ohjelmien koodit tässä kohdassa toisiaan.

## 11. Yhteyden katkaisu

VF	224	Systemek	413
----	-----	----------	-----

Yksinkertainen ohjelmanpätkä yhteyden katkaisemiseksi, ohjelmakoodissa on samantapaisia kohtia, jotka on helposti selitettävissä. Puutuinkin jo edellisessä lausunnossani tässä kohdassa VTT:n lausunnossa olleeseen virheeseen (Pulkkinen oli tulkinnut tulostustekstin vahingossa kommentiksi).

## 12. Päivän numeron laskeminen

VF	251	Systemek	780
----	-----	----------	-----

Muistipaikan 780 rutiinissa lasketaan päivämäärästä (kuukausi ja pvm) päivän järjestysnumero vuoden alusta. Koska kuukausissa on eri määrä päiviä, on päivien määrä talletettu merkkijonoon (31 29 31 30...), joka silmukan joka kierroksella kopioidaan puskuriin, siitä poistetaan jo käsiteltyjen kuukausien merkit, ja kasvatetaan summaa ko. kuukauden päivien määrällä. Silmukan joka kierroksella saadaan siis lisättyä yhden kuukauden päivien määrä.

Tämä kohta oli tutkimassani koodissa ainoa, jossa ohjelmassa oli havaittavissa jonkinasteista innovatiivisuutta (keksin tosin itse helposti paremman ratkaisun). Innovatiivisuutta ei tässäkään tarvita siksi, että ongelma olisi jotenkin hankala, vaan siksi, että TCL-kieli on niin alkeellinen, ettei se sisällä indeksoituja muuttujia. Lähes missä hyvänsä muussa ohjelmointikielessä koko algoritmi olisi korvattavissa yhdellä ainoalla (tai muutamalla) käskyrivillä.

Ohjelmien koodit ovat lähes samat. Ohjelmoija on varmaankin aikoinaan miettinyt jonkin aikaa, ennen kuin on keksinyt tämän ratkaisun. Tällöin ei ole ihmeellistä, että ohjelmoija myöhemmin muistaa keksimänsä ratkaisun ja käyttää sitä myöhemmin uudelleen.

### 13. PATU-standardiin liittyvät kohdat

VF	393	Systek	658
VF	397	Systek	656
VF	391	Systek	657
VF	396	Systek	671
VF	573	Systek	668
VF	56	Systek	659

Nämä viimeiseksi jättämäni ohjelmarivit liittyvät pankkien turvastandardiin. Ajan puutteen vuoksi minulla ei ollut mahdollisuutta käydä niitä yksityiskohtaisesti läpi. Koska ko. kohtien algoritmit otaksuttavasti määräytyvät PATU-standardin perusteella, en usko, että niissä havaitut samankaltaisuudet tuovat mitään uutta esille.

### 14. Yhteenveto

Läpikäymäni ohjelmakoodi tukee aikaisempia selvityksiä (VTT ja allekirjoittaneen aikaisempi selvitys), eikä mielestäni tuo mitään oleellista uutta esille. VTT:n raportissa todettiin, että "toisiaan muistuttavien kohtien yhteenlaskettu osuus on suuruusluokkaa 10% koko ohjelman koosta" (täysin samanlaisten kohtien osuus on huomattavasti pienempi)<sup>1</sup>. Kuten aikaisemmassa lausunnossani totesin, ei tämä osoita laitonta kopiointia tapahtuneen, sillä samankaltaisuudet voi selvittää johtuviksi seuraavista seikoista.

- Tärkein kaikista: ohjelman ovat toteuttaneet samat kaksi ohjelmoijaa.
- Ohjelmien ulkoiset liittymät ohjaavat toteutusta. Näistä tärkeimmät ovat käyttöliittymä ja tietoliikenneliittymä. Käyttöliittymä on toteutettu uudessa ohjelmistossa (lähes) samanlaiseksi ja tietoliikenneliittymän määräävät käytettävät standardit.
- Toteutusympäristö ohjaa tietynlaisiin ratkaisuihin. Tässä suhteessa käytetty maksupäätte on primitiivisyydessään äärimmäinen esimerkki -- variointimahdollisuuksia on vähän.
- Ohjelmoijilla voi olla käytettävissään muistiinpanoja, listauksia, esimerkkejä, kurssimateriaaleja yms. joista vanhat ratkaisut ovat nähtävissä.

Läpikäymäni ohjelmakoodi on kauttaaltaan yksinkertaista eikä vaadi ohjelmoinnin peruskoulutusta kummempaa asiantuntemusta. Tyyliiltään koodi on arvioni mukaan siinä mielessä huonoa, että sen toiminnasta on paikoitellen (koodin periaatteellisen yksinkertaisuuden huomioon ottaen) aivan tarpeettoman hankala päästä selvyteen.

Itse ohjelman tekeminen ei ole tällaisen ohjelmiston tapauksessa vaikeaa, vaan vaikeaa ja työlästä on sen selvittäminen, mitä pitää tehdä. Ohjelmistoa tehdessä on varmasti kulunut paljon aikaa, kun on esimerkiksi selvitelty, miten sanomavälityksen yksityiskohdat pankkiyhteyksillä on toteutettava, miten valinnanmahdollisuuksia sisältävät parametrit kannattaa asettaa jne. Kun tämántapaiset asiat on

---

1. Lisäksi kannattaa ottaa huomioon, että maahantuoja yrityksen tekemän ohjelmiston osuus on vain hyvin pieni osa koko asiakkaalle myytävästä tuotteesta, joka koostuu maksupäätelaitteesta, VeriFonen siihen toimittamasta perusohjelmistosta sekä maahantuoja yrityksen laitteeseen laittamasta ohjelmasta.

selvitetty (mikä vaatii usein aikaavievää tiedonhankintaa, käytännön kokeilua, koulutusta jne.), on itse ohjelmiston laatiminen helppoa. Kun samat ohjelmoijat tekevät saman ohjelman toiseen kertaan, ei selvitystyötä enää tarvita: toteutus on suoraviivaista. Ohjelma substanssiarvo yritykselle ei siis ole pelkästään itse ohjelma, vaan myös se asiantuntemus, joka yritykseen kertyy ohjelmistoa tehtäessä. Asiantuntemus syntyy työn tehneille ihmisille, ja tätä asiantuntemusta yritys ei voi suojata tekijänoikeuslainsäädännön perusteella.

Myöskään liikesalaisuuteen ei voi mielestäni tässä tapauksessa vedota jo pelkästään siitä syystä, että ohjelmisto ei sisällä mitään erityisiä keksintöjä tai oivalluksia, vaan kaikki tarvittava tieto on saatavissa käsikirjojen ja standardien muodossa. Lisäsi VeriFonen maksupäätteen mahdollistaa sen sisältämän ohjelmiston selauksen. VeriFonen manuaalin "TRANZ 420 Reference Manual" sivulla 5-6 on esitetty, miten päätteen ohjelmaa voidaan vapaasti lueskella muutamalla näppäimen painalluksella. Ohjelma näkyy näytössä ns. konekielisessä muodossa, mutta VeriFonen maksupäätteen tapauksessa ohjelmoijan kirjoittama esitysmuoto ja koneen ymmärtämä esitysmuoto ovat lähes samat -- vain kommentit, jotka ohjelmistossa muutenkin ovat puutteelliset, puuttuvat. Samassa kohdassa kerrotaan, miten selailu voidaan haluttaessa estää tallettamalla muistipaikkaan 17 nollasta poikkeava luku. En yrityksistäni huolimatta onnistunut löytämään Systekin ohjelmasta muistipaikan 17 sisällön asettavaa kohtaa. Salasanan asetus on tietysti voitu tehdä jollain muullakin tavalla, mutta on todennäköistä (ja helposti varmennettavissa), että ohjelmisto on vapaasti selailtavissa miltä tahansa Systekin myymältä maksupäätteeltä.

Tampereella 24.2.1997

Ilkka Haikala