Vili Lähtevänoja

Käyttäjänhallintajärjestelmä PK-yrityksen tuotannon tarpeisiin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

17.5.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Vili Lähtevänoja Käyttäjänhallintajärjestelmä PK-yrityksen tuotannon tarpeisiin 34 sivua 17.4.2012
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaajat	vanhempi järjestelmäasiantuntija Kim Engdahl yliopettaja Jarkko Vuori

Tämän työn aiheena oli kehittää TAG Systems Finland Oy:n tuotannon tarpeisiin toteutettu käyttäjänhallinta- ja sisäänkirjautumisjärjestelmä korvaamaan vanha, käytössä oleva järjestelmä. Tavoitteena oli luoda järjestelmä joka mahdollistaa käyttäjien hallinnoinnin keskitetysti palvelimelta, kaksinkertaisen tunnistautumisen älykortin ja PINin kanssa, sekä monen käyttäjän jaettu istunto.

Järjestelmän pohjana käytettiin avoimen lähdekoodin pGina-ohjelmistoa, joka mahdollistaa lisäosarajapintansa myötä omien tunnistautumislogiikoiden luonnin Windows-käyttöjärjestelmän sisäänkirjautumisprosessiin. PGinaan kehitettiin C#-ohjelmointikielellä lisäosa joka toteuttaa halutun kaksinkertaisen tunnistautumisen mahdollistavan logiikan. Lisäksi kehitettiin avustavia sovelluksia hallitsemaan muita järjestelmän toiminnallisuuksia, kuten PINin vaihtoa ja työaseman lukitusta.

Työn lopputuloksena syntyi ennalta asetetut vaatimukset toteuttava järjestelmä, joka on valmis vietäväksi tuotantoympäristöön testaukseen. Järjestelmä toimii hyvänä alustana tulevaisuudessa tapahtuvalle jatkokehitykselle, kuten biometrisen tunnistuksen lisäämiselle.

Johtopäätöksenä työstä voi vetää, että avoimen lähdekoodin järjestelmät voivat toimia hyvänä pohjana yritysten omille ratkaisuille. Ilman pGinaa järjestelmän kehitys olisi ollut erittäin hankalaa tai se olisi pitänyt ostaa toiselta taholta.

Avainsanat	C#, .NET, pGina, sovelluskehitys, tunnistautuminen, älykortti

Author(s) Title Number of Pages Date	Vili Lähtevänoja User administration system for the needs of an SME 34 pages 17 April 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Kim Engdahl, Senior Systems Specialist Jarkko Vuori, Principal Lecturer

The purpose of the project described in this thesis was to develop a user management and login system for TAG Systems Finland Oy, to replace an older system that is currently in use. The goal was to develop a system that implements centralized user management, two-factor authentication with a smart card and PIN, and multiple-user shared sessions.

A piece of open-source software called pGina was used as a base for the system. With its plugin-interface it enables the development of custom authentication logic for the login process of a Windows-operating system. A plugin was developed using the C# programming language, which implements the two-factor authentication logic that was desired.

As a result of the project, a system was developed that reaches all the goals that were set for it and which is ready to be taken into the production environment for testing. It also serves as a good base for future development, such as example addition of biometric authentication.

A conclusion can be drawn that open-source systems can serve as a good base for companies to build their own solution on, as without pGina the development of the system would have been very hard or it would have to have been bought from another party.

Keywords	C#, .NET, pGina, software development, authentication, smart card

Sisällys

Lyhenteet ja käsitteet

.NET	Microsoftin kehittämä ohjelmistoalusta.
0x	Etuliite, jota käytetään kun annettava luku on heksadesimaalimuodossa.
API	*Application Programming Interface*. Sovelluksen rajapinta, jota käyttämällä muut sovellukset voivat kommunikoida sovelluksen kanssa.
C#	Microsoftin kehittämä oliopohjainen ohjelmointikieli.
CLR	*Common Language Runtime*. Microsoftin .NET ohjelmistoalustan virtuaalikone.
CP	*Credential Provider*. GINA:n korvaaja Windows Vistasta lähtien. Tarjoaa helpompaa muokattavuutta GINA:an verrattuna.
FCL	*Framework Class Library*. Microsoftin .NET ohjelmistoalustan luokkakirjasto.
GINA	*Graphical Identification and Authentication*. Kirjasto, joka hoitaa sisäänkirjautumisia ja tunnistautumisia Microsoft Windowsissa. Tämä on korvattu Windows Vistasta lähtien Credential Providereilla.
PAN	*Primary Account Number*. Älykortin yksilöivä luku.
pGina	Avoimen lähdekoodin sisäänkirjautumisjärjestelmä Windowsille.
PIN	*Private Identification Number*. Älykortin salasana.
T0	T=0. Merkkiorientoitunut asynkroninen yksisuuntainen lähetysprotokolla.
T1	T=1. Lohko-orientoitunut asynkroninen yksisuuntainen lähetysprotokolla.

1 Johdanto

Työ on tehty TAG Systems Finland Oy:lle älykorttituotannon työpisteiden käyttäjien hallinnoinnin helpottamiseksi. Tavoitteena oli kehittää käyttäjähallintajärjestelmä, jossa tunnistautuminen tapahtuu kaksinkertaisesti (Two-factor Authentication) ja käyttäjiä voidaan hallinnoida keskitetysti palvelimella. Kaksinkertaisella tunnistautumisella tarkoitetaan tässä tapauksessa sitä, että käyttäjä tunnistautuu hänen hallussaan olevalla älykortilla sekä PIN-numerolla. Yrityksellä oli kaksinkertaisen tunnistautumisen mahdollistava järjestelmä jo entuudestaan käytössä, mutta siitä puuttui keskitetyn käyttäjienhallinnan mahdollisuus ja sen toiminta oli yleisesti epävakaata ja mahdollista kiertää.

1.1 Työn tausta

TAG Systems Finland Oy on erikoistunut turvallisten älykorttiratkaisujen kehittämiseen ja toimittamiseen pankeille, julkiselle sektorille ja yksityisyrityksille. Korttiaihiot tulevat Vantaan toimipisteeseen valmiina, ja siellä suoritetaan niiden yksilöinti. Yksilöinnillä tarkoitetaan asiakkaan tietojen viemistä kortin pintaan, magneettiraidalle sekä mikrosirulle. Prosessissa on monta eri vaihetta, ja vaiheissa tarvitaan usein koneita joita, ohjataan tietokoneilla. Visan ja Mastercardin tiukat turvavaatimukset asettavat vaatimuksia koneiden käytön valvonnalle.

Koneisiin on käyttäjänhallintaa varten asennettu nk. screenlocker–ohjelma. Ohjelma luo koneen ruudulle ikkunan, joka estää koneen käytön. Voidakseen käyttää konetta käyttäjän on kirjauduttava ohjelmaan sisään älykortin ja PINin yhdistelmällä tai vaihtoehtoisesti tunnuksella ja salasanalla. Samaa älykorttia käytetään myös muuhun kulunvalvontaan, joten käyttäjän on otettava kortti mukaansa työasemalta poistuttaessa. Tällöin ohjelma lukitsee työpisteen, eikä istunto voi jäädä vahingossa auki, jolloin joku muu käyttäjä voisi sitä käyttää. Näin varmistetaan, että aina kun konetta käytetään jää lokiin tallenne käytöstä.

Käyttäjät ovat olleet tyytymättömiä screenlocker-ohjelmaan lukuisista ongelmista johtuen. Ohjelman luoma estoikkuna voi muun muassa jumittua välillä estäen sisäänkir-

jautumisen, ja se on mahdollista kiertää tietyissä erikoistapauksissa antaen mahdollisuuden käyttää työasemaa ilman lokiin jäävää tallennetta. Lisäksi sovellus toimii vain paikallisilla käyttäjätunnuksilla, eli uutta käyttäjää lisätessä pitää käydä läpi tuotantoalueen kaikki työasemat.

Tarkasteltavana oli useita kaupallisia ratkaisuja korvaavaksi järjestelmäksi. Projektia suunniteltaessa otettiin yhteyttä muun muassa turvallisuusratkaisuja valmistaviin HID Globaliin ja SafeNetiin, joilta löytyy tarvittavan kaltaisia älykorttipohjaisia tunnistautumisratkaisuja. Vaikka nämä ratkaisut olisivat mahdollistaneet erittäin laajan ja monipuolisen käytön työasemiin ja erilaisiin palveluihin tunnistautumisessa, ne eivät kumminkaan olleet sopivia tuotannon tarpeisiin, sillä ne eivät mahdollistaneet usean käyttäjän jaettua istuntoa.

Tuotannon toiminnassa on erittäin tärkeä ominaisuus, että eri käyttäjät voivat hallita ja tarkkailla samaa tuotantoprosessia eli toimia samassa istunnossa. Kun käyttäjä A on kirjautunut sisään, käynnistänyt prosessin ja lukinnut työaseman, käyttäjän B tulee kyetä avaamaan työasema ja jatkamaan prosessin hallinnointia. Tarkastelluista ratkaisuista mikään ei tukenut tätä käyttötapaa ja ne olivat suljettuja järjestelmiä, joten tämän kaltaisen toiminnallisuuden itse lisääminen ei ollut mahdollista.

Toinen vaatimus on, että sisäänkirjautumisohjelman käyttämä älykortin lukija pitää pystyä rajaamaan vain tiettyyn lukijaan, sillä useissa koneissa on kytkettynä myös muita älykortin lukijoita, joita ei saa häiritä, sillä niitä käytetään sirun yksilöintiin tai muuhun tuotannolliseen sirun ohjelmointiin, kuten alustamiseen.

Hakukoneella tehdyllä haulla löytyi pGina-niminen ilmainen avoimen lähdekoodin tunnistautumisjärjestelmä. Järjestelmää tarkasteltaessa todettiin, että sillä kyettäisiin toteuttamaan tunnistautuminen tuotannon tarpeiden mukaisesti. Älykorttitukea ei järjestelmästä itsestään löydy, mutta hyvien rajapintojen ansiosta sen luominen katsottiin mahdolliseksi. Lisäksi katsottiin, että uusien ominaisuuksien ja toiminnallisuuksien lisääminen tulisi olemaan paljon nopeampaa ja hallitumpaa, kun järjestelmä on kehitetty yrityksen sisäisesti, etenkin kun yrityksestä löytyi jo ennestään erittäin vahvaa tietotaitoa älykorttien ja niihin perustuvien ratkaisujen saralla. Vaihtoehtojen punnitsemisen

jälkeen päätettiin alkaa rakentaa omaa ratkaisua avoimen lähdekoodin pGina-ohjelmiston päälle. [1.]

1.2 Työn tavoitteet

Työn tavoitteena oli luoda luotettava, toimiva ja helposti käytettävä sisäänkirjautumis- ja käyttäjänhallintajärjestelmä pGina-ohjelmiston päälle. Sisäänkirjautumisen tuli onnistua sekä älykortin ja PINin yhdistelmällä että tarpeen tullen paikallisen käyttäjän käyttäjätunnuksella ja salasanalla. Tulevaisuudessa tarkoituksena on lisätä sormenjälkitunnistus tunnistuskeinoksi.

Jokaista työasemaa tullaan avaamaan ja lukitsemaan mahdollisesti kymmeniä kertoja päivässä, joten toimenpiteen tulee toimia nopeasti ja sujuvasti. Työntekijän tulee tarvita vain laittaa älykortti lukijaan, syöttää PIN ja painaa Enter avatakseen työaseman. Työaseman tulee myös lukittua varmasti ja ripeästi, kun älykortti poistetaan lukijasta.

Käyttäjien hallinnan tulee olla mahdollista keskitetysti palvelimelta. Uuden käyttäjän lisääminen vaatii täten vain älykortin valmistuksen ja tietojen lisäämisen palvelimelle. Tunnistautuminen tapahtuu älykortin ja palvelimen välillä, ja työasema toimii vain tulkkina ja viestien välittäjänä.

2 Työn vaatimuksia ja toteutus

2.1 Työn rajaus

Työ rajattiin niin, että tarkoituksena oli luoda sisäänkirjautumisjärjestelmän rakenne ja logiikka sille tasolle, että sitä voitaisiin alkaa viedä tuotantoympäristöön. Tähän rajaukseen päädyttiin, koska tuotantoympäristön työasemat eivät vielä käytä Windows 7 -käyttöjärjestelmää, jota järjestelmän toiminta vaatii, eikä ole tietoa, milloin tämä päivitys tapahtuu. Tarkoituksena on käsitellä vain järjestelmää ja sen toimintaa, ei tuotantoympäristöön viemisen vaatimaa testausta ja järjestelyitä. Tuotantoympäristöön vieminen tapahtuu porrastetusti niin, että järjestelmä otetaan käyttöön ensin ei-kriittisillä

työasemilla, joilla voidaan katsoa miten järjestelmä toimii oikeassa käytössä. Tämän jälkeen kun mahdolliset ongelmat on korjattu, aletaan järjestelmää viedä kriittisempiin työasemiin. Koska tuotantoympäristö on tarkasti säännelty, järjestelmä on sinne viemistä varten myös dokumentoitava laajasti.

Järjestelmä viedään toiminnaltaan sille tasolle, että tunnistautuminen on mahdollista älykortin ja PINin yhdistelmällä. Järjestelmää tullaan mahdollisesti kehittämään tulevaisuudessa niin, että kirjautuminen onnistuu myös biometrisesti eli sormenjäljen perusteella, mutta tämän toiminnallisuuden toteuttaminen ei kuulunut työn piiriin. Tietyt asiat kortin komentorajapinnassa määräytyivät sen mukaan, että yhteensopivuus jo aiemmin toteutetun biometriikkakorttitoteutuksen kanssa säilyi.

2.2 Resurssit

2.2.1 C#-ohjelmointikieli ja .NET Framework

C# on Microsoftin kehittämä vahvasti tyyppiturvallinen olio-ohjelmointikieli. Sen juuret ovat C-perheessä, joten sen syntaksi on tuttu C-, C++- ja Java-ohjelmoijille. [2, s. 1.] C# on yleensä vahvasti yhdistetty Microsoftin .NET Framework-ympäristöön, joka tarjoaa sille ajoympäristön ja luokkakirjastot mutta joka toimii vain Windows-käyttöjärjestelmissä. Eräs vaihtoehto .NET Frameworkille olisi ollut Mono, joka on avoimen lähdekoodin ratkaisu, joka pyrkii toteuttamaan monella eri alustalla toimivan C#- ja CLR-standardien mukaisen ympäristön. Mono toimii muun muassa Windowsilla, Linuxilla, OS X:llä, BSD:llä, Androidilla ja iOS:llä. [3.]

.NET Framework on Microsoftin kehittämä ohjelmistoympäristö. Se on vahvasti sidottu C#-ohjelmointikieleen, jolle se toteuttaa monet C#-standardin määrittämistä ominaisuuksista. Vaikka .NET Frameworkia käytetään yleensä C#:n kanssa, tukee se myös monia muita kieliä, kuten C++:aa, Visual Basicia ja JavaScriptiä [4].

.NET Frameworkissa on kaksi osaa: Framework Class Library (FCL) ja Common Language Runtime (CLR). FCL tarjoaa suuren määrän luokkia, joilla voidaan toteuttaa laaja kirjo toiminnallisuuksia, kuten IO-operaatioita, graafisia käyttöliittymiä, verkon yli

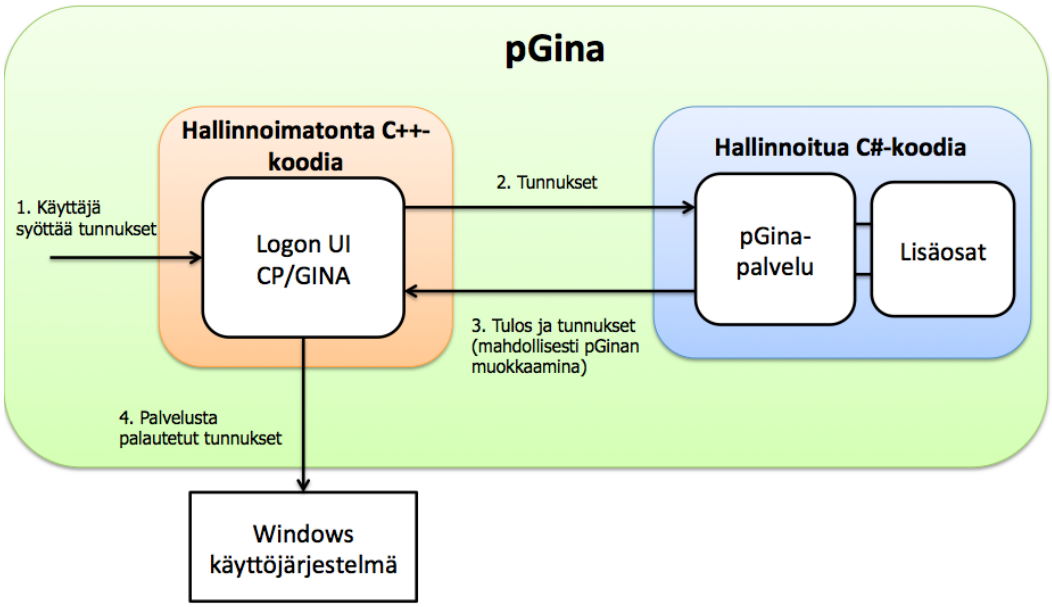
kommunikointia ja salausoperaatioita. FCL tarjoaa myös määrittelyt .NET datatyypeille. Koska datatyypit on määritelty FCL:ssä, ne ovat yhtenäiset eri .NET:iä käyttävien kielten välillä, joten niiden välinen kommunikaatio on helppoa. [5.]

Common Language Rungime (CLR) on ajoympäristö, jossa .NET ohjelmat ajetaan. Se tarjoaa muun muassa automaattista muistinhallintaa ja muistialuerajojen tarkistusta. [5.] Tämä tarkoittaa, että sillä ajetut ohjelmat voidaan toteuttaa nk. hallinnoidulla koodilla. Hallinnoidulla koodilla tarkoitetaan, että ohjelmoijan ei itse tarvitse pitää huolta esimerkiksi muistinhallinnasta, vaan se voidaan jättää sen ajoympäristön hoidettavaksi. Hallinnoimattoman koodin kirjoittaminen on mahdollista käyttämällä unsafe-avainsanaa [6].

C# ja .NET Framework valittiin alun perin toteutustekniikoiksi yksinkertaisesti siitä syystä, että pGinan lisäosarajapinta on tarkoitettu käytettäväksi näillä tekniikoilla, joten minkään muunlaisen tekniikan käyttäminen toiminnallisuuden toteuttamiseen vaatisi liian paljon työtä [7]. Myöskään esimerkiksi Javan tarjoamalle monialustaisuudelle ei ollut tarvetta, sillä järjestelmää tultaisiin käyttämään vain Windows-koneilla. Kehitystyössä huomattiin, että tekniikat olivat yleisestikin erittäin päteviä sovelluskehitykseen. C#:in selkeä ja tuttu syntaksi sekä .NET Frameworkin laaja kokoelma luokkia ja metodeja, yhdistettynä Microsoftin ylläpitämään ensiluokkaiseen ja ajan tasalla olevaan dokumentointiin, edesauttoivat kehitystyötä.

2.2.2 PGina

PGina on avoimeen lähdekoodiin perustuva, Windowsissa vakiona olevan sisäänkirjautumisjärjestelmän korvaaja. Vaikka on tarjolla monenlaisia samankaltaisia ratkaisuja, pGinan etuna on sen lisäosarajapinta. Asennuksessa tulee mukana laajan toiminnallisuuden takaava kokoelma lisäosia, ja lisäksi lisäosarajapintaa käyttäen käyttäjien on helppo ja nopea kirjoittaa lisäosia, jotka toteuttavat heidän itse haluamiansa erikoistoiminnallisuuksia.



Kuva 1. pGinan rakenne.

Kuvassa 1 on havainnoitu pGinan rakennetta. Siitä voidaan nähdä, että pGina koostuu kolmesta komponentista. Ensimmäinen komponentti on Windows XP:ssä Graphical Identification and Authentication (GINA) -komponentin ja Windows Vistasta eteenpäin Credential Provider (CP) -komponentin korvaava, hallinnoimattomalla C++-ohjelmointikielellä kirjoitettu komponentti. Tämä korvaa täysin vakiona olevan tunnistautumistoiminnallisuuden. Näin voidaan muokata muun muassa kirjautumisruudun ulkonäköä, kuten voidaan nähdä kuvasta 2.

Kuva 2. PGinan sisäänkirjautumisruutu.

Toinen komponentti on pGina Service, joka muodostaa pGinan ytimen. pGina Service saa CP/GINA:lta kirjautumistunnukset ja välittää ne käytössä oleville lisäosille. Se ei itsessään tee tunnistautumista, vaan toimii tunnusten ja tunnistautumistulosten välittäjänä CP/GINA:n ja lisäosien välillä.

Kolmantena komponenttina toimivat käytössä olevat lisäosat. Lisäosia tulee asennuksen mukana useita, mutta niiden luominen itse on myös mahdollista. Lisäosat toimivat kolmivaiheisessa prosessissa, jonka vaiheet ovat

- Tunnistus
- Oikeutus
- Portti.

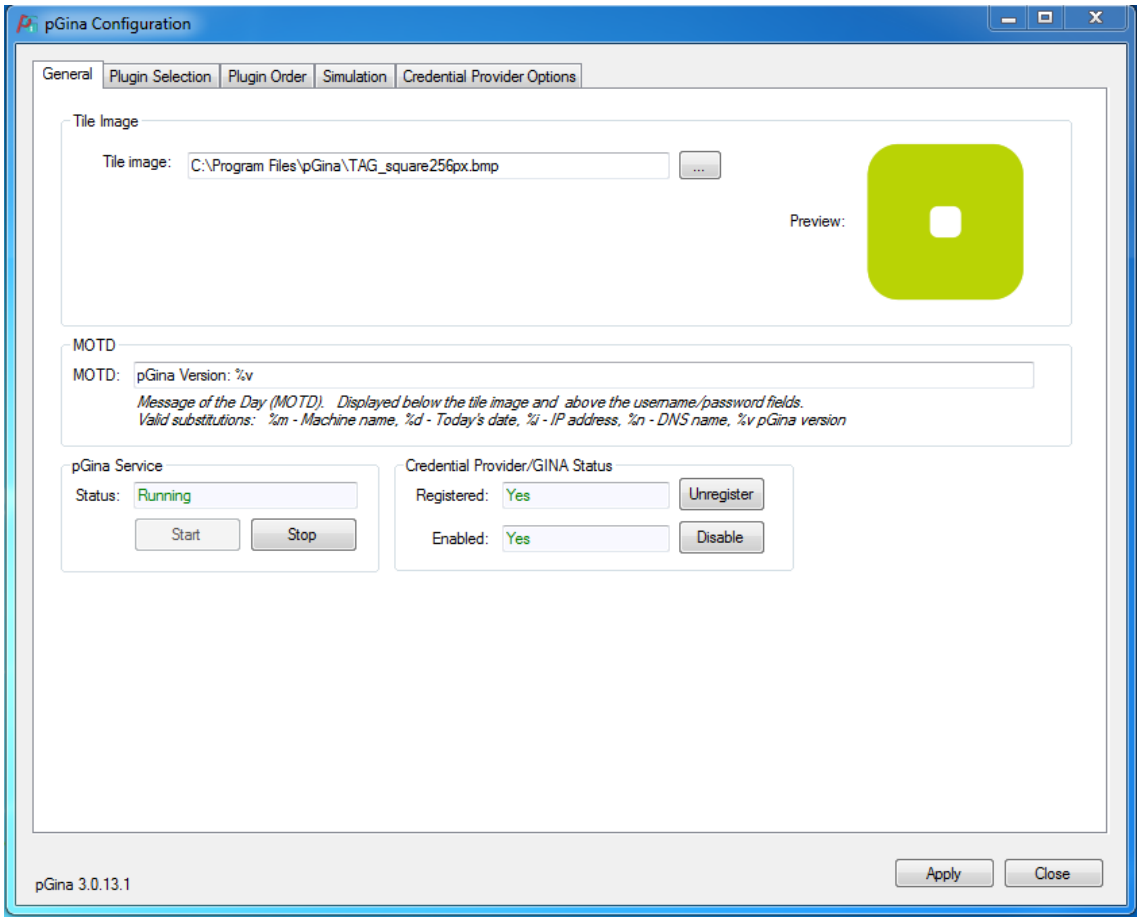
Lisäosat voivat toteuttaa 0–3 vaihetta näistä, lisäosan ominaisuuksista ja käyttäjän valinnoista riippuen. Esimerkiksi LocalMachine-lisäosa kykenee toimimaan kaikissa kolmessa vaiheessa. Jotta lisäosa pystyy toimimaan tietyssä vaiheessa, sen tulee toteuttaa sille vaiheelle määritelty rajapinta. Esimerkiksi tunnistautumistoiminnallisuuteen vaaditaan IPluginAuthentication -rajapinnan toteuttaminen. Lisäosilla voi olla lisäksi myös ilmoitustoiminnallisuus, mutta se ei osallistu sisäänkirjautumisprosessiin.

Tunnistusvaiheessa tarkistetaan, että käyttäjä on se, joka hän väittää olevansa. Tarkistus voidaan suorittaa vakiolisäosien avulla ulkoisesti esimerkiksi LDAP:ia, RADIUS:ta tai MySQL-tietokantaa vasten, mutta myös koneen paikallisia käyttäjätunnuksia vasten tunnistautuminen onnistuu. Tämän vaiheen onnistumiseen vaaditaan vähintään yhden lisäosan suorittama onnistunut tunnistautuminen. Jos yhtään lisäosaa ei ole toiminnassa, vaihe ei mene läpi.

Oikeutusvaiheessa tarkistetaan, että käyttäjällä on oikeudet päästä käsiksi niihin resursseihin, joita hän pyytää. Käyttäjältä voidaan vaatia esimerkiksi tiettyyn ryhmään kuulumista. Tämän vaiheen läpimenemiseen vaaditaan kaikkien toiminnassa olevien lisäosien onnistuneen suorituksen. Jos yhtään lisäosaa ei ole toiminnassa, vaihe ei mene läpi.

Porttivaiheessa tapahtuu tunnistautumisen jälkeistä käyttäjänhallintaa. Esimerkiksi jos LocalMachine on toiminnassa tässä vaiheessa, sen tehtävänä on luoda koneelle kirjautumistunnuksia vastaavan käyttäjän luonti työasemalle. Tämän vaiheen läpimenemiseen vaaditaan kaikkien toiminnassa olevien lisäosien onnistuneen suorituksen. Jos yhtään lisäosaa ei ole toiminnassa, vaihe ei mene läpi.

Ilmoitustoiminnallisuuden toteuttava lisäosa vastaanottaa ilmoituksia istuntotapahtumista, ja se voi suorittaa toimenpiteitä näiden tapahtumien mukaan. Istuntotapahtumia ovat muun muassa sisäänkirjautuminen, uloskirjautuminen, työaseman lukitus ja työaseman avaus. Yleisin käyttö tälle toiminnallisuudelle on lokien kirjoitus.[7.]



Kuva 3. PGinan konfigurointidialogin General-ikkuna kun palvelu on päällä.

Kuvassa 3 on yksi konfigurointidialogin ikkunoista. PGinan konfigurointidialogissa on viisi ikkunaa. General-ikkunassa voidaan asettaa sisäänkirjautumisruudun logo ja teksti sekä säätää pGina-palvelun sekä CP:n/GINAn toimintaa. Plugin Selection -ikkunassa voidaan asettaa lisäosia päälle ja pois, kun taas Plugin Order -ikkunassa voidaan säätää näiden lisäosien ajojärjestystä. Simulation-ikkunassa voidaan ajaa simulaatioita käyttäen joko ohjelmallista simulointia tai varsinaista pGina-palvelua. Credential Provider Options -ikkuna on vain Windows 7 -versiossa, ja siinä voidaan ottaa Windows 7:n alkuperäinen CP pois toiminnasta. Tätä pidetään epäsuositeltavana, sillä jos Windowsin alkuperäinen CP on pois toiminnasta ja pGina jostain syystä ajautuu epäkuntoon, voi kone olla pahimmillaan täysin käyttökelvoton.

2.2.3 PCSC-Sharp

PCSC-Sharp on Daniel Muellerin .NET Frameworkille C#-ohjelmointikielellä kirjoitettu kirjasto älykorttien kanssa tapahtuvaan kommunikaatioon. Windowsin älykorttikommunikaatio on toteutettu Winscard-kirjastolla, joka on kirjoitettu hallinnoimattomalla C-kielellä. Hallinnoimattomien C-kielen funktioiden kutsuminen hallinnoidusta C#-koodista on mahdollista, mutta se on melko työlästä. PCSC-Sharp helpottaa tätä toimimalla kääreenä Winscard-kirjastolle, jolloin ohjelmoija voi käyttää PCSC-Sharpin kautta Winscardin toiminnallisuuksia kutsumalla C#-metodeja. C-kielen funktioiden työläs kutsuminen on hoidettu siinä. PCSC-Sharp tarjoaa myös luokan, jolla voi rakentaa kansainvälisen älykorttien ominaisuuksia määrittelevän ISO-7816-standardin mukaisia komentoja.

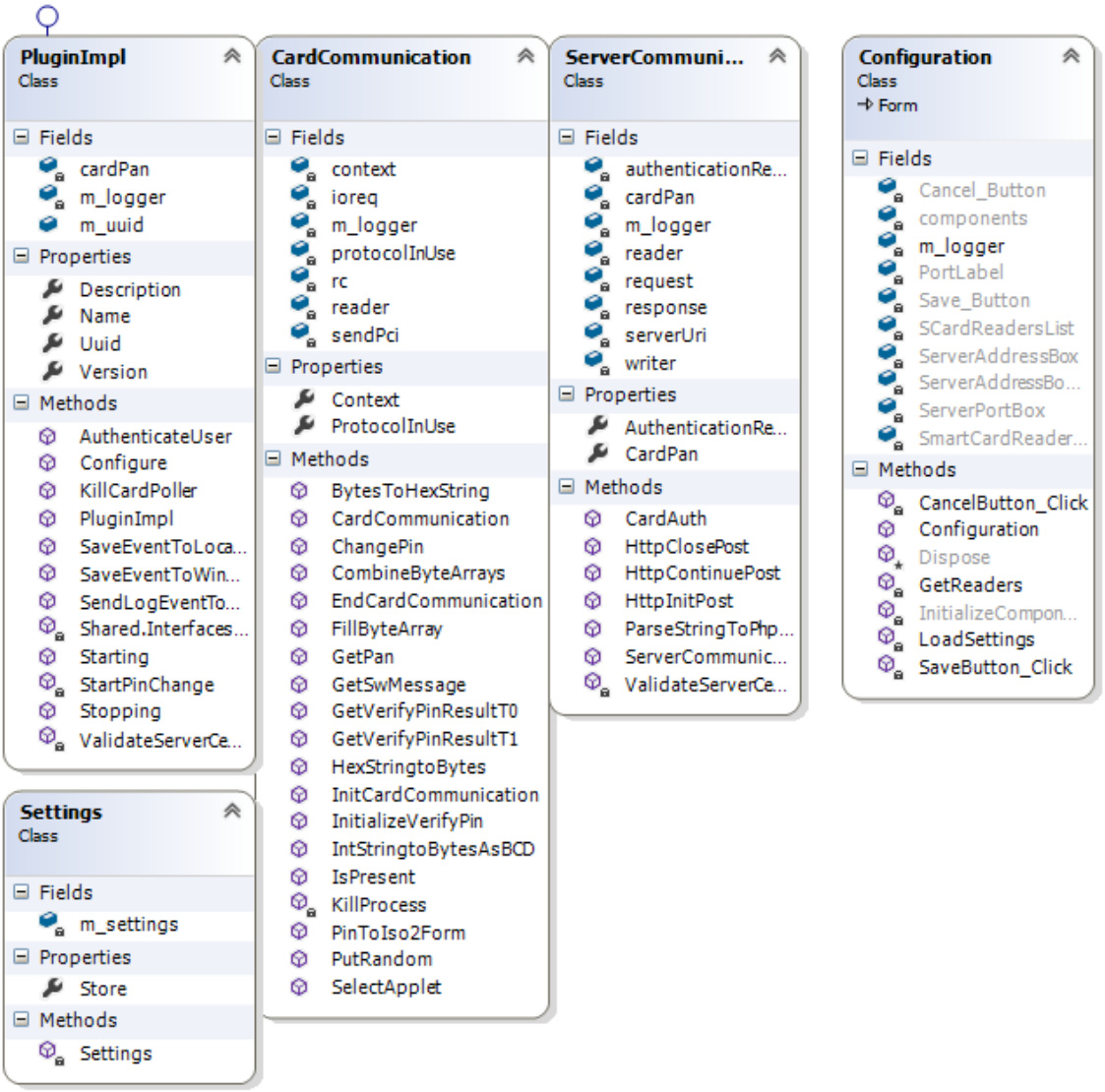
3 Toteutunut käyttäjähallintajärjestelmä

3.1 pGinaan kirjoitettu lisäosa

3.1.1 Yleiskuvaus

Koska pGina ei itsessään tarjoa älykorttitunnistautumista, täytyi se toteuttaa itse. Tämän mahdollisti pGinan lisäosarajapinta. Rajapintaa hyödyntäen lähdimme toteuttamaan TAGin-nimistä lisäosaa, johon rakensimme älykorttitunnistautumisen toiminnallisuuden.

pGina määrittelee rajapinnat tunnistus-, oikeutus- ja porttivaiheissa toimiville lisäosille, sekä ilmoitustoiminnallisuuden toteuttaville lisäosille ja konfiguraatiodialogille. Kun lisäosa toteuttaa yhden tai useamman näistä rajapinnoista, pGina tunnistaa sen ja osaa käsitellä sitä sen mukaan. TAGin toteuttaa tunnistus- sekä konfiguraatiodialogirajapinnan. Konfiguraatiodialogirajapinnan toteutus mahdollistaa sen, että käyttäjä voi säätää lisäosan asetuksia pGinan konfigurointi-ikkunasta. Lisäksi rajapinta antaa mahdollisuuden tallentaa asetukset pGinan aliavaimina, ja käyttää niiden käsittelyyn pGinan tarjoamia metodeja.



Kuva 4. PGinaan kirjoitetun lisäosan luokkakaavio.

Kuvasta 4 voidaan havaita, että TAGin-lisäosa koostuu viidestä luokasta:

- PluginImpl
- CardCommunication
- ServerCommunication
- Settings

- Configuration

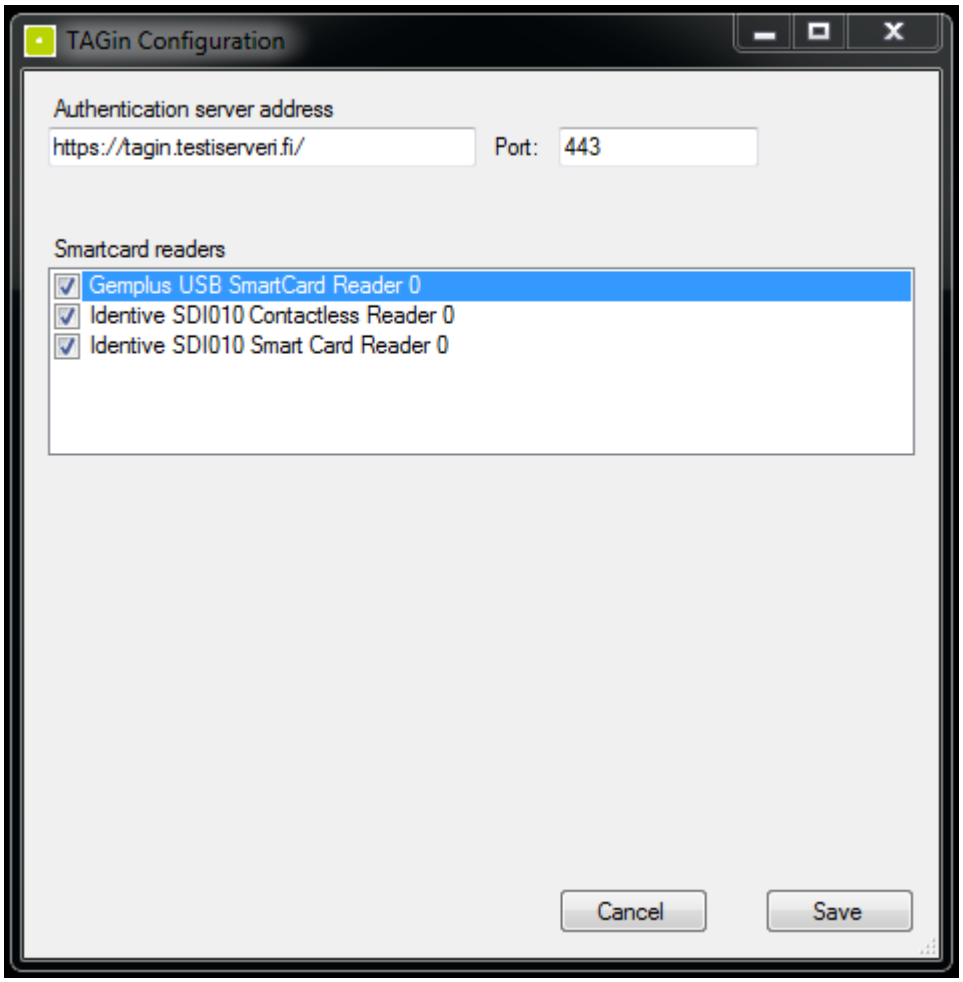
TAGinin ytimenä toimii PluginImpl-luokka, ja sen AuthenticateUser-metodi. Tämä metodi on se, jota pGina kutsuu tunnistautumista varten, ja joka palauttaa lisäosan tunnistautumistuloksen. PluginImpl-luokka sisältää myös muutaman AuthenticateUserissa käytetyn metodin, sekä muita ominaisuuksia, kuten lisäosan GUID:in (Global Unique Identifier), jota pGina käyttää lisäosien yksilöintiin.

PluginImpl-luokkaa tukevat CardCommunication- ja ServerCommunication-luokat. CardCommunication-luokan metodit mahdollistavat älykortin kanssa kommunikoinnin. Tätä varten se sisältää metodit tunnistautumisappletin valitsemiseen, PANin pyytämiseen, satunnaisluvun syöttämiseen, älykortin läsnäolon tarkistamiseen, sekä PINin tarkistamiseen ja vaihtamiseen. Lisäksi se sisältää lukuisia tukevia metodeja, jotka keskittyvät lähinnä tiedon muuntamiseen muodosta toiseen. Niillä voidaan hoitaa muun muassa tavutaulukon muunto merkkijonoksi, heksadesimaalimerkkijonon muunto tavutaulukoksi, kokonaislukumerkkijonon muunto BCD-tavutaulukoksi (Binary Coded Decimal) sekä kahden tavutaulukon yhdistäminen. Älykortille vietävä, ja siltä tuleva, tieto on heksadesimaalimuodossa ja tavuina, joten näitä muunnosmetodeja käytetään monessa paikassa.

Järjestelmän on tarkoitus tukea sekä kontaktillisia että kontaktittomia älykortteja, joka on otettu CardCommunication-luokassa huomioon. Kontaktilliset älykortit käyttävät kommunikaatiossa T0-protokollaa, ja kontaktittomat käyttävät T1-protokollaa. T0- ja T1-protokollan väliset erot ovat melko pieniä, vaikuttaen vain muutamaan operaatioon. T1-protokolla esimerkiksi palauttaa komennon vastauksessa suoraan siinä pyydetyn operaation tuloksen, kun taas T0-protokollassa palautetaan käskyn vastauksessa operaation tuloksen koko, ja itse tulos pitää noutaa erillisellä käskyllä. T1-protokolla myös vahvistaa operaation onnistumisen joissakin tapauksissa eri palautusarvolla kuin T0. Esimerkiksi onnistuneen appletin valinnan tunnistaa T1-protokollan palautusviestissä siitä, että viestin ensimmäisen tavun sisältö on heksadesimaalina 0x61. T0-protokollan palautusviestissä taas kahden viimeisen tavun sisältö on heksadesimaalina 0x9000. Kortin käyttämä protokolla saadaan selville, kun korttiin otetaan yhteys ensimmäistä kertaa tunnistautumisen aikana.

ServerCommunication-luokka kommunikoi palvelimen kanssa. Sen sisältämä CardAuth-metodi on pääasiallisessa vastuussa siitä, mitä lisäosa palauttaa tunnistautumisen tulokseksi. Tämä metodi on vastuussa SSL-suojatun yhteyden muodostamisesta palvelimelle käyttäen muita luokan sisältämiä metodeja, sekä palvelimen antamien tietojen käsittelystä. Se myös syöttää ja pyytää tietoja älykortilta käyttämällä CardCommunication-luokan metodeja. Palvelimen ja älykortin antamien viestien mukaan metodi tekee päätöksiä jatkotoimenpiteistä, viestien eteenpäin välittämisestä, sekä tunnistautumisen onnistumisesta.

Settings-luokka hallinnoi pGinan aliavaimina rekisteriin tallennettavia TAGinin asetuksia. Se sisältää dynaamisen objektin, jonka kautta asetuksia haetaan ja tallennetaan. Luokan konstruktorissa asetetaan, mitä eri asetuksia sen kautta hallinnoidaan, ja näiden asetusten vakioarvot.



Kuva 5. TAGinin konfigurointi-ikkuna.

Configuration-luokka sisältää graafisen konfigurointidialogin, joka voidaan nähdä kuvasta 5, ja siihen liittyvät metodit. Konfigurointidialogissa voidaan säätää ja tallentaa TAGinin asetuksia. Asetukset tallennetaan kahteen paikkaan: Settings-luokan avulla pGinan aliavaimiksi rekisteriin sekä erilliseen asetustiedostoon. Rekisteriin tallennettuihin asetuksiin pääsee käsiksi vain TAGin-lisäosa, joten asetustiedoston avulla voidaan tarpeelliset asetukset jakaa myös muiden järjestelmän ohjelmien käyttöön. Jos asetustiedosto turmeltuu tai sille tapahtuu jotakin, voidaan tiedoston sisältö palauttaa nopeasti käymällä TAGinin konfiguraatioruudussa ja tallentamalla asetukset. Konfigurointidialogissa voidaan määrittää TAGinin käyttämän palvelimen osoite ja portti sekä TAGinin ja järjestelmän tukevien ohjelmien käyttämät kortinlukijat.

```
2013-03-07 16:27:59,215 [1444|11|DEBUG] PluginDriver:97606c26-f2bc-4273-bcce-d67700480657: Calling 13dacac2-4f8a-566c-aaf6-d75d6ac96e79
2013-03-07 16:27:59,215 [1444|11|INFO ] pGina.Plugin.TAGin.CardCommunication: Context is valid
2013-03-07 16:27:59,215 [1444|11|INFO ] pGina.Plugin.TAGin.CardCommunication: Connecting to reader
2013-03-07 16:27:59,215 [1444|11|INFO ] pGina.Plugin.TAGin.CardCommunication: Connected to reader succesfully: Gemplus USB SmartCard Reader 0
2013-03-07 16:27:59,231 [1444|11|ERROR] pGina.Plugin.TAGin.CardCommunication: Communication with card initialized succesfully
2013-03-07 16:27:59,262 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: Applet selection successful
2013-03-07 16:27:59,324 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: PAN retrieved successfully from card
2013-03-07 16:27:59,371 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: Random number put successfully to card
2013-03-07 16:27:59,480 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: PIN verification initialized successfully
2013-03-07 16:27:59,652 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: Successfully retrieved pin verification result
2013-03-07 16:27:59,668 [1444|11|INFO ] pGina.Plugin.TAGin.ServerCommunication: Authentication successful, closing card connection
2013-03-07 16:27:59,714 [1444|11|DEBUG] pGina.Plugin.TAGin.CardCommunication: Card communication ended, reader disconnected
2013-03-07 16:27:59,714 [1444|11|DEBUG] PluginDriver:97606c26-f2bc-4273-bcce-d67700480657: 13dacac2-4f8a-566c-aaf6-d75d6ac96e79 Succeeded
```

Kuva 6. TAGin:in loki onnistuneen tunnistautumisen tapauksessa.

PGina käyttää Apachen Log4Net-kirjastoa lokien kirjoittamiseen. TAGin kirjaa tapahtumat samaan lokiin kuin pGina ja muut lisäosat, käyttäen Log4Net-kirjastoa. Kuvasta 6 voidaan nähdä, mitä lokiin kirjoitetaan onnistuneen TAGin:in tunnistautumisen tapauksessa.

3.1.2 Tunnistautumisprosessi

Taulukko 1. Laitteiden välinen kommunikaatio tunnistautumisprosessin aikana.

Vaihe	Lähettäjä	Vastaanottaja	Toimenpide
1	Työasema	Kortti	Valitse kortilta tunnistautumissovelma
2	Työasema	Kortti	Hae kortilta PAN-numero
3	Työasema	Palvelin	Vie PAN-numero palvelimelle
4	Työasema	Kortti	Vie palvelimelta saatu satunnaisluku kortille
5	Työasema	Kortti	Todenna annettu PIN-luku kortilla
6	Työasema	Palvelin	Vie kortin vastaus palvelimelle
7	Palvelin	Työasema	Tunnistautumisen tulos

Taulukossa 1 on vaiheittain kuvailtu kommunikaatiota työaseman, kortin ja palvelimen välillä tunnistautumisprosessin aikana. Jokaiseen vaiheeseen, viimeistä vaihetta lukuunottamatta, sisältyy myös vastaanottajan vastaus lähettäjän viestiin, joka sisältää joko tiedon vastaanottamisen vahvistuksen, virheilmoituksen tai lähettäjän pyytämän tiedon. Jos jokin vaihe epäonnistuu, tunnistautuminen hylätään eikä käyttäjää päästetä käyttämään työasemaa.

Tunnistautumisprosessi alkaa siitä, kun käyttäjä on syöttänyt sisäänkirjautumisruudussa PIN-luvun. PIN-luku viedään pGinaan, joka välittää sen TAGin-lisäosalle. TAGin-lisäosa ottaa yhteyden korttiin ja alkaa suorittaa tunnistautumisen vaiheita. Jos yhteyden ottaminen korttiin ei onnistu, tunnistautuminen epäonnistuu jo heti alussa.

Ensimmäisessä vaiheessa työasema valitsee kortilta SELECT-komennolla tunnistautumissovelman. Kontaktillisen T0-protokollaa käyttävän ja kontaktittoman T1-protokollaa käyttävän kortin palautusarvoissa on tämän komennon osalta eroa. Kun tunnistaudutaan kontaktillisesti eli T0-protokollaa käyttäen onnistunut sovelman valinta voidaan tunnistaa palautusarvon kahdesta viimeisestä tavusta, joiden sisältö on heksadesimaalina 0x9000. Kontaktittomassa tunnistautumisessa onnistunut appletin valinta tunnistetaan ensimmäisestä tavusta, jonka sisältö on heksadesimaalina 0x61. Jos appletin valinta ei onnistu, tunnistautuminen hylätään.

Toisessa vaiheessa työasema hakee kortilta sen PAN-numeron. Tätä numeroa käytetään kortin, ja sitä kautta käyttäjän, yksilöintiin. Jos PAN-numeron hakeminen onnistuu, palautetun tuloksen kaksi viimeistä tavua sisältävät heksadesimaalina 0x9000. Kahta viimeistä tavua edeltävät tavut sisältävät kortin PAN-numeron.

Kolmannessa vaiheessa PAN-numero viedään palvelimelle. Kommunikointi tapahtuu suojatusti HTTPS-protokollalla. Jos PANin vastaanotto onnistui ja PIN on ajan tasalla, palvelin palauttaa satunnaisluvun. Jos PIN on vanhentunut, palvelin palauttaa PINin vanhentumista ilmaisevan viestin. Tällöin vaihe menee läpi, mutta tieto otetaan huomioon, ja jos tunnistautuminen menee muuten hyväksytysti läpi, käynnistetään lopuksi PINChange-ohjelma, jolla käyttäjä voi vaihtaa PINin. Jos palvelin palauttaa muunlaisen palautusviestin tai ei mitään, tunnistautuminen hylätään.

Neljännessä vaiheessa palvelimelta saatu satunnaisluku viedään kortille. Palvelin käyttää satunnaislukua myöhemmässä vaiheessa varmistamaan, että se kommunikoi saman kortin kanssa. Jos satunnaisluvun vieminen kortille onnistuu, palautusarvon kaksi viimeistä tavua sisältävät heksadesimaalina 0x9000.

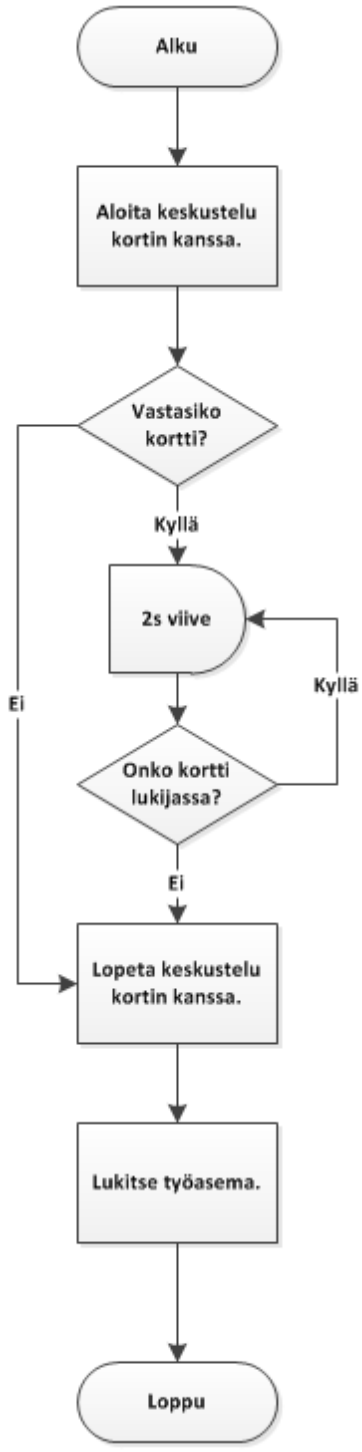
Viidennessä vaiheessa kortti todentaa annetun PINin, varmistaen, että käyttäjän antama PIN on sama kuin kortille tallennettu. Jos käyttäjän antama PIN on oikea, kortti palauttaa vastauksena salatun viestin, joka sisältää muun muassa kortin sisältämän PINin, palvelimelta saadun kahdeksan tavun pituisen satunnaisluvun, PANin sekä PINin viimeisen vaihtopäivämäärän. Tämän komennon kohdalla on taas ero kontaktillisen T0-protokollan ja kontaktittoman T1-protokollan kohdalla. T0-protokollan tapauksessa jos käyttäjän antama PIN on oikea, kortti palauttaa viestin, joka on muotoa 0x61XX, jossa XX sisältää kortin oikean palautusviestin koon. Tämä oikea palautusviesti täytyy hakea erillisellä komennolla. T1-protokollan tapauksessa ei oikeaa palautusviestiä tarvitse erikseen hakea, vaan se tulee suoraan vastauksena ensimmäiseen komentoon. Salatun viestin sisältö on ennalta määrättyä muotoa.

Kuudennessa vaiheessa kortin palauttama salattu viesti viedään palvelimelle. Palvelin purkaa viestin avaimella, jonka se löytää kolmannessa vaiheessa sille annetun PANin avulla. Kun viesti on purettu, palvelin käy läpi sen sisällön, varmistaen sen oikeellisuuden. Palvelin tarkistaa muun muassa, että viestissä annettu PIN vastaa palvelimen tietokannassa olevaa, viestin sisältämä satunnaisluku on sama kuin palvelimen antama, viestin PAN on sama kuin aikaisemmassa kommunikaatiossa. Jos kortin PIN on vaihdettu, palvelin päivittää vaihtopäivämäärän tietokantaan.

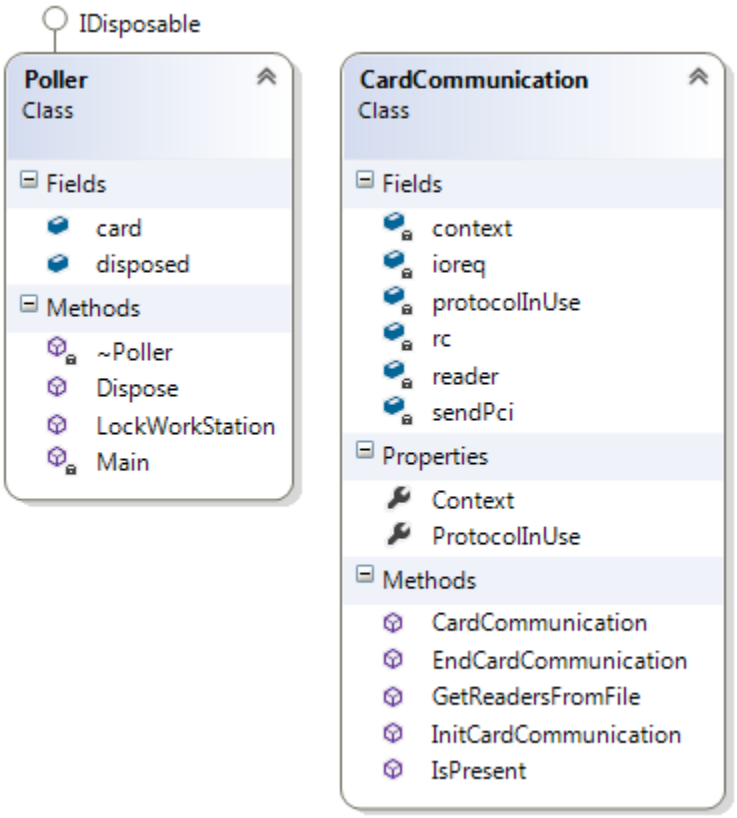
Seitsemännessä vaiheessa palvelin palauttaa tunnistautumisen tuloksen. Jos palvelin katsoo tunnistautumisen onnistuneen, TAGin palauttaa pGinalle omalta osaltaan hyväksyvän tuloksen. Jos palvelin hylkää tunnistautumisen, TAGin palauttaa pGinalle hylkäävän tuloksen ja hylkäyksen syytä kuvaavan viestin.

3.2 Tukevat sovellukset

3.2.1 Älykortin poistoa valvova sovellus



Kuva 7. Älykortin poistoa valvovan sovelluksen toiminta vuokaaviona.



Kuva 8. Älykortin poistoa valvovan sovelluksen luokkakaavio.

Älykortin poistoa valvova sovellus (CardPoller) on taustalla pyörivä näkymätön prosessi, jonka toimintaperiaate on kuvattu kuvassa 7. Sen tehtävänä on lukita työasema, kun kortinlukijassa ei ole korttia. CardPollerin toiminnallisuus on kriittinen osa kokonaisjärjestelmän toiminnallisuutta, sillä työaseman lukittuminen älykortin poistossa on tärkeä turvallisuuden, käyttäjävalvonnan ja helppokäyttöisyyden komponentti tuotantoympäristössä. Kuvasta 8 voidaan havaita, että sovelluksessa on käytetty kortin kanssa kommunikointiin vahvasti karsittua versiota TAGin-lisäosan CardCommunication-luokasta.

Sovelluksen alussa luetaan tiedostosta pGinan TAGin-lisäosan tallentamat, eriteltyjen kortinlukijoiden nimet. Sovellus tulee käyttämään vain näitä lukijoita toiminnassaan, sillä työasemaan mahdollisesti muuten liitetyt kortinlukijat voivat olla muussa käytössä. Seuraavaksi sovellus pyrkii ottamaan yhteyden korttiin, joka on yhdessä näistä lukijoista. Jos yhteyden muodostaminen ei onnistu, työasema lukitaan jo tässä vaiheessa.

Jos ensimmäinen yhteydenotto onnistuu, siirtyy sovellus silmukkaan. Silmukassa sovellus nukkuu kaksi sekuntia ja lähettää kortille viestin. Jos kortti vastaa viestiin, siirtyy sovellus taas nukkumaan. Tämä kierto jatkuu niin kauan, kunnes kortti poistetaan lukijasta tai sovellus terminoidaan.

Kun sovelluksen lähettämään viestiin ei vastata, se katsoo, että kortti on poistettu lukijasta. Tällöin kortti kutsuu työaseman lukitsevaa funktiota. Jos metodi ajetaan onnistuneesti, ohjelma lopettaa toiminnan. Jos funktion ajo epäonnistuu, ohjelma heittää poikkeuksen.

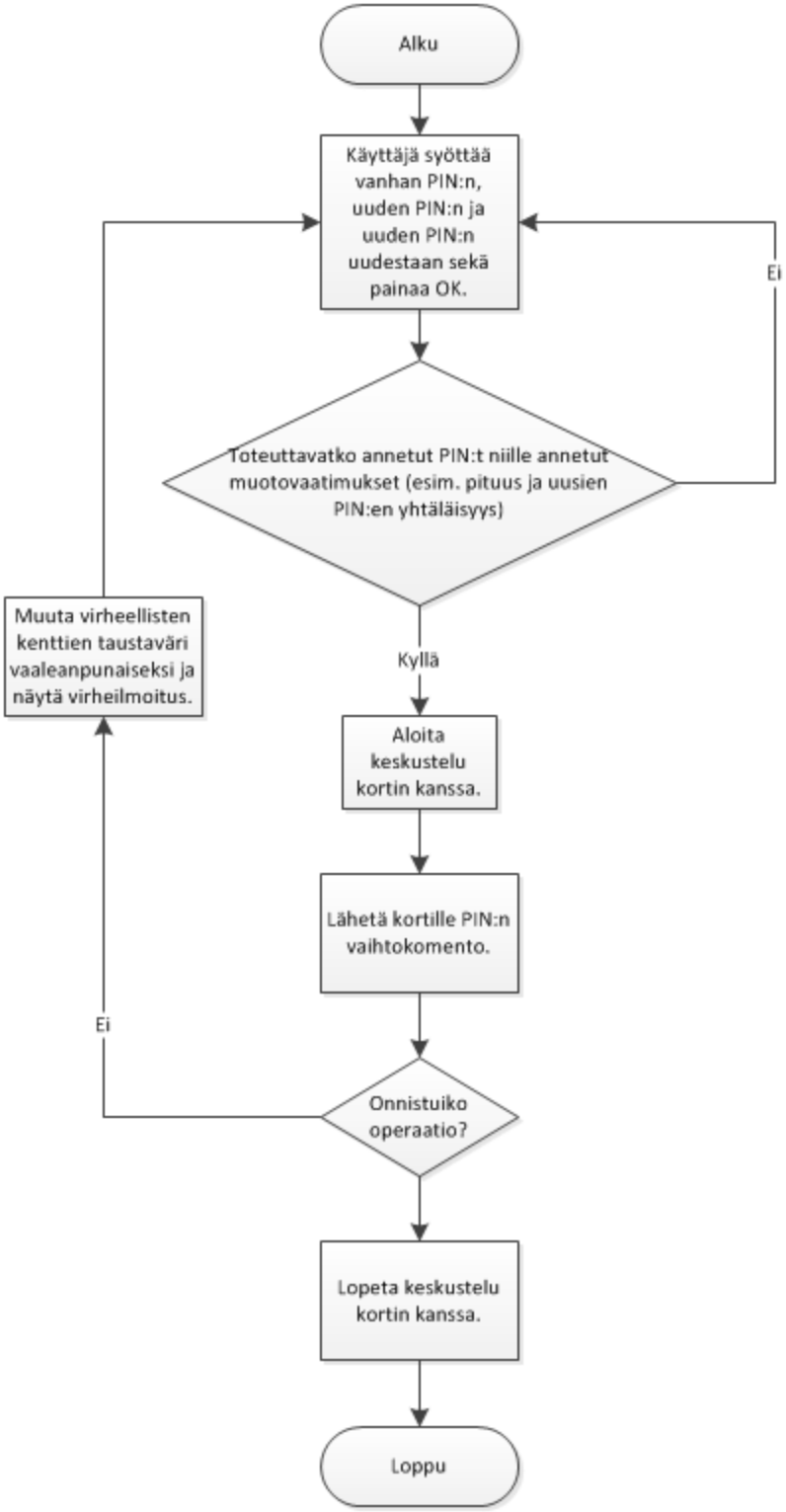
Työaseman lukitseva funktio on Windowsin API-metodi LockWorkStation, joka sijaitsee user32-kirjastossa. User32-kirjasto on kirjoitettu hallinnoimattomalla C++-kielellä, joten sen metodien liittäminen ohjelmaan vaatii Platform Invoken (P/Invoke) suorittamista. P/Invoke mahdollistaa hallinnoimattomien funktioiden kutsumisen hallinnoituun koodiin ja sitä voi käyttää System.Runtime.InteropServices-nimiavaruudesta [8].

```
[DllImport("user32", SetLastError = true)]
public static extern bool LockWorkStation();
```

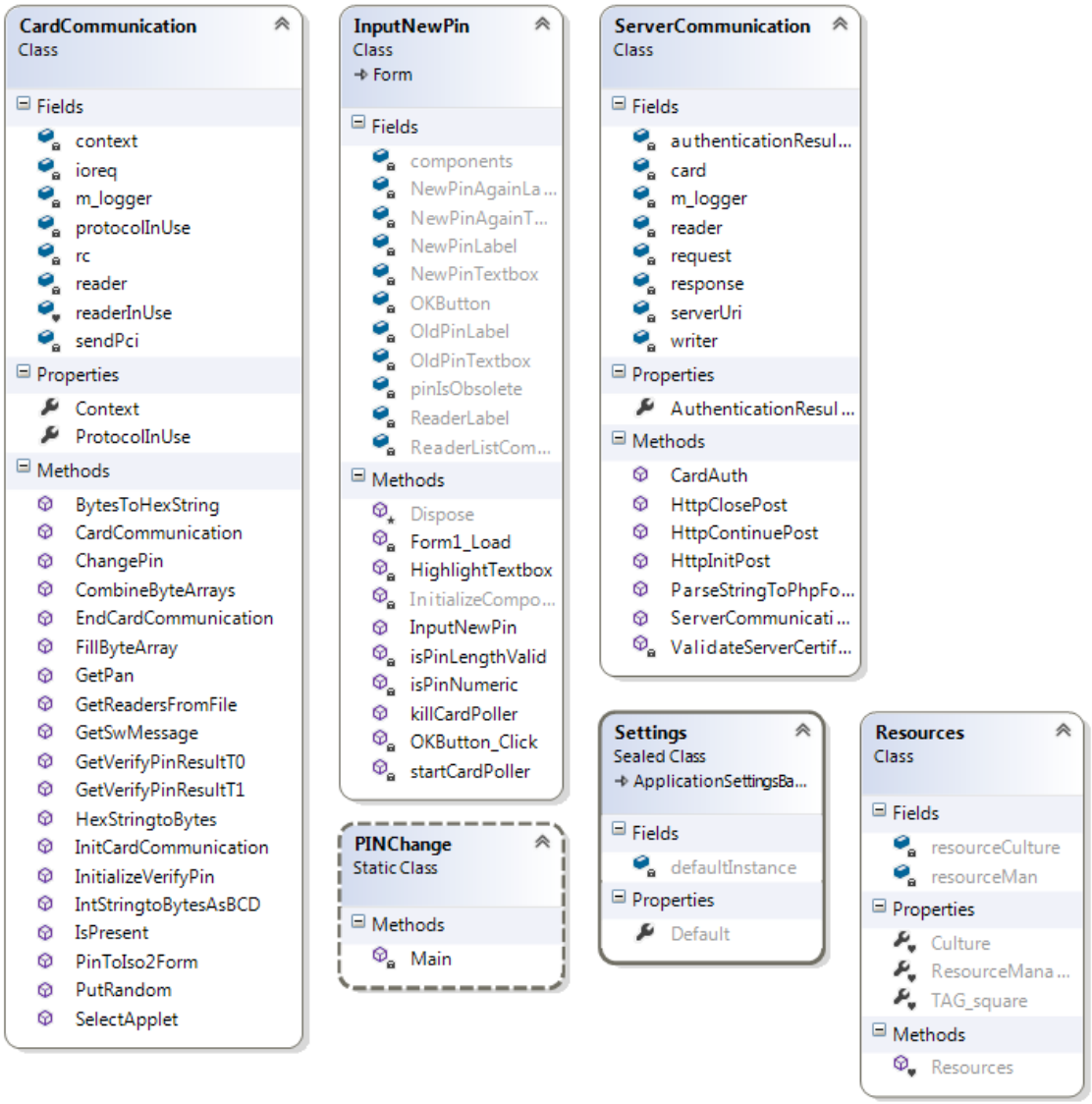
Kuva 9. Hallinnoimattoman LockWorkStation-funktion kutsu hallinnoituun koodiin P/Invokella.

Kuvassa 9 on ohjelmakoodista ote, jossa julistetaan LockWorkStation-funktio. DllImport on attribuutti, jolla määritetään, että kutsuttava metodi sijaitsee hallinnoimattomassa Dynamic-Link Library -kirjastossa (DLL) staattisena sisäänmenopisteenä. Sen sisällä annetaan kirjaston nimi ja asetetaan SetLastError-kenttä todeksi. SetLastError-kentän todeksi asettaminen tarkoittaa, että .NETin hallinnoimattoman muistin metodeja sisältävällä System.Runtime.InteropServices.Marshal-luokalla voidaan funktion ajamisen mahdollisen epäonnistumisen jälkeen saada sen syy selville kutsumalla GetLastWin32Error-metodia. [9.]

3.2.2 PINin vaihtosovellus

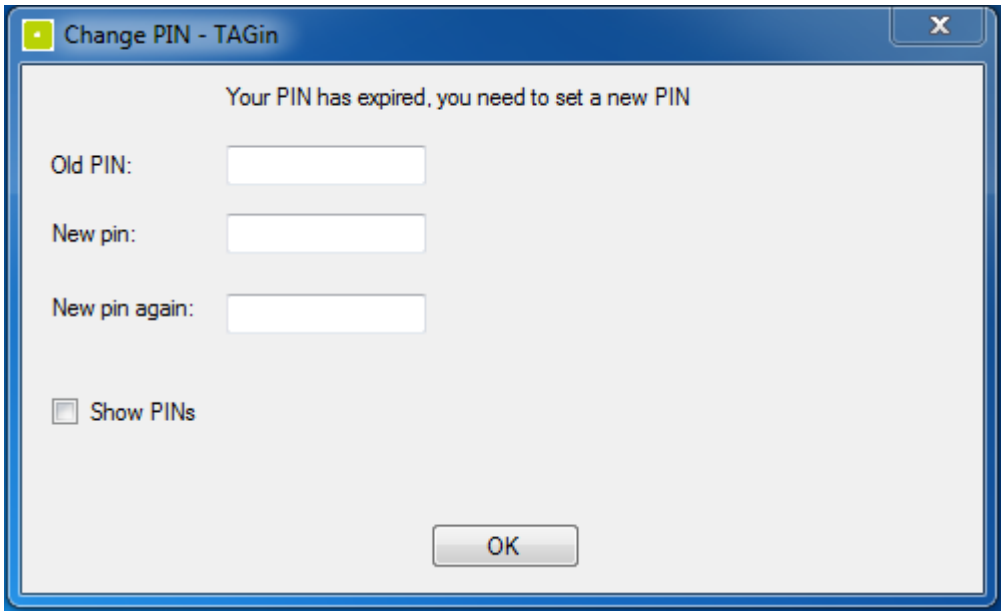


Kuva 10. PINin vaihtosovelluksen toiminta vuokaaviona.



Kuva 11. PINin vaihtosovelluksen luokkakaavio.

PINin vaihtosovellus (PINChange) on Windows Forms (WinForms) graafisen käyttöliittymän toteutusrajapinnalla toteutettu pieni sovellus, jonka toimintaperiaate voidaan nähdä kuvasta 10. Sovelluksen tarkoituksena on antaa käyttäjän vaihtaa kortin PIN-numero. Kuvasta 11 voidaan nähdä, että sovelluksessa on käytetty TAGin-lisäosan CardCommunication-luokkaa kortin kanssa kommunikoimiseen ja ServerCommunication-luokkaa palvelimen kanssa kommunikoimiseen.



Kuva 12. PINin vaihtosovelluksen pääikkuna.

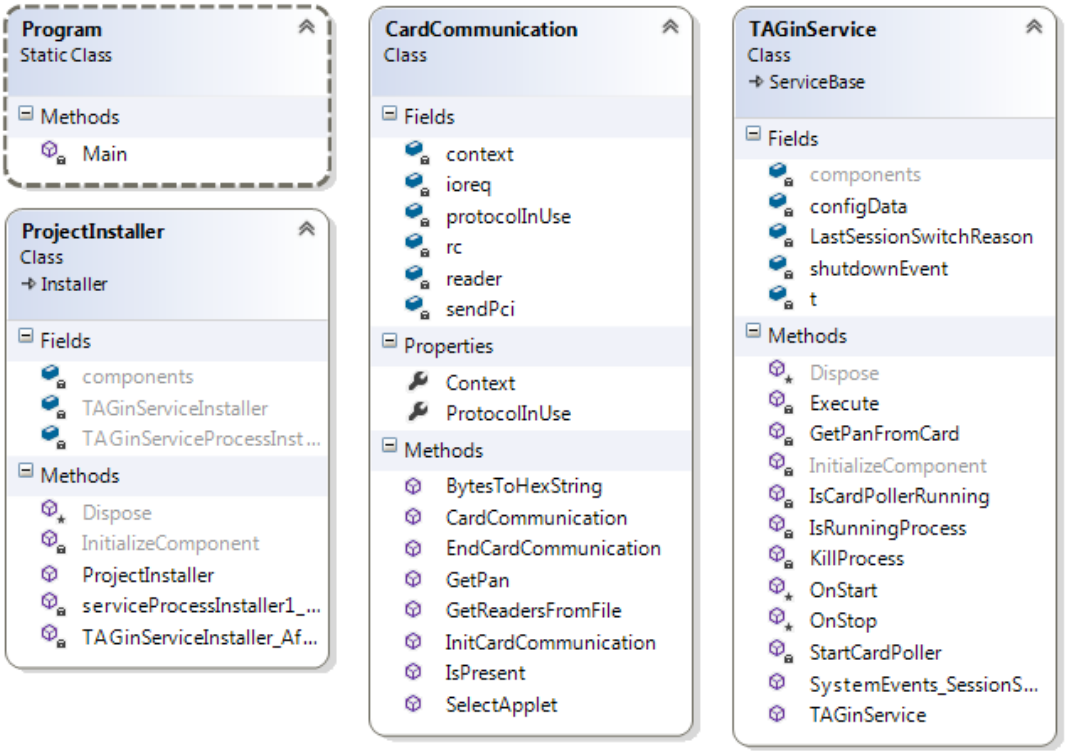
Kun sovellus käynnistetään, ilmestyy ruudulle ikkuna, joka voidaan nähdä kuvasta 7. Ikkunassa on kolme kenttää vanhalle PINille, uudelle PINille ja uuden PINin toistosyötölle. Kun käyttäjä on syöttänyt kenttiin arvot ja painanut OK, suoritetaan arvojen tarkistus. Jotta PINin vaihtoa kortille edes yritetään, tulee uusien PINien arvojen toteuttaa niille annetut vaatimukset. Molempien uusien PINien tulee olla 4–12 merkkiä pitkiä ja sisällön tulee olla vain numeerista. Lisäksi niiden tulee olla yhtenevät. Jos nämä vaatimukset eivät täyty, näytetään käyttäjälle virheikkuna, joka listaa puutteet. Lisäksi puutteita sisältävien kenttien tausta maalataan vaaleanpunaiseksi, jotta ne erottuvat. Kun vaatimukset täyttyvät, voidaan aloittaa PINin vaihto kortille.

Ennen kuin sovellus voi aloittaa kortin kanssa kommunikaation, tulee lopettaa taustalla mahdollisesti pyörivä CardPoller-sovellus, jotta korttiin voidaan ottaa yhteys. Tämän jälkeen voidaan aloittaa kommunikaatio. Kortille viedään ensin yhteensopivuussyistä kahdeksan tavun pituinen satunnaislukusarja, jonka jälkeen PINin vaihto voidaan suorittaa yhdellä komennolla. Tämän komennon mukana viedään kortille vanha ja uusi PIN. Kortti vertaa näitä kahta ja niiden täsmätessä vaihtaa PINin. Jos operaatio onnistuu, näytetään käyttäjälle sitä ilmaiseva viesti, josta käyttäjä voi napsauttaa itsensä ulos, jolloin ohjelma sulkeutuu. Sulkeutuessaan ohjelma vielä käynnistää lopuksi CardPoller-sovelluksen. Jos operaatio epäonnistuu, käyttäjälle näytetään virheviesti ja hänet ohjataan takaisin pääikkunaan.

Alun perin PINChange-sovellus oli tarkoitus käynnistää TAGin-lisäosassa siinä tapauksessa, että palvelimelta takaisin tulleessa viestissä ilmaistaan, että PIN on vanhentunut. Tästä ratkaisusta oli luovuttava, sillä pGina pyörii systeemikäyttäjän oikeuksilla, joilla ei ole mahdollista käynnistää graafista käyttöliittymää. Korvaavana, ja mahdollisuuksien mukaan väliaikaisena, ratkaisuna päädyttiin ohjeistamaan käyttäjiä vaihtamaan PIN-numero tietyin väliajoin omatoimisesti. Kun käyttäjä itse käynnistää sovelluksen, se käynnistyy tavallisen käyttäjän oikeuksilla, joten se voi näyttää graafisen käyttöliittymän.

Tapaa, jolla sovellus saa selville järjestelmän käyttämät kortinlukijat, jouduttiin myös vaihtamaan. Alkuperäisessä versiossa sovellus luki kortinlukijat samasta rekisteristä, jota myös TAGin-lisäosa käyttää niiden tallentamiseen. Sovellus kehitettiin Windows XP:llä, jossa tämä ei ollut ongelma, mutta Windows 7:ssä rekisterien lukuoikeudet olivat muuttuneet niin, ettei sovellus enää niihin päässyt. Ratkaisuna päätettiin muuttaa TAGinin toimintaa niin, että tallennettaessa lisäosan asetuksia rekisteriin tallennettiin korttilukijoiden tiedot myös ini-tiedostoon, johon PINChange-sovellus pääsee käsiksi.

3.2.3 Tukevia sovelluksia hallitseva palvelu



Kuva 13. Tukevia sovelluksia hallitsevan palvelun luokkakaavio.

Tukevia sovelluksia hallitseva palvelu (TAGin Service) on taustaprosessina pyörivä Windows-palvelu (Windows Service), joka luotiin korjaamaan Windows XP:n ja Windows 7:n välillä tiukentuneiden turva- ja käyttäjäoikeusasetusten järjestelmän toiminnalle aiheuttamia ongelmia. Se käynnistetään automaattisesti Windowsin käynnistyessä, ja se toimii CardPollerin ja PINChangen hallinnoijana ajaen näitä sovelluksia tarpeen mukaan. Kuvasta 13 voidaan havaita, että sovelluksessa on käytetty TAGin-lisäosan CardCommunication-luokasta karsittua versiota.

Kortin poistoa valvovan sovelluksen (CardPoller) tuli alun perin olla pGinan TAGin-lisäosan hallinnoima. Ajatuksena oli, että kun TAGin hyväksyy kirjautumisen, se lopuksi käynnistäisi CardPollerin ja CardPoller aloittaisi toimintansa. Tämä toimikin Windows XP:llä hyvin, mutta Windows 7:llä muuttuneet turva- ja käyttäjäoikeusasetukset estivät toiminnan. CardPollerin käynnistys toimi hyvin, kun pGinan toimintaa emuloitiin ohjelmallisesti, mutta itse palvelua käytettäessä tuli ongelmia. Emulointi ajetaan senhetkisen

sisäänkirjautuneen käyttäjän oikeuksilla, kun taas itse palvelu ajetaan järjestelmäkäyttäjän oikeuksilla. Tästä pääteltiin, että toimimattomuus johtuu näiden käyttäjien oikeuksien eroista.

pGina ja sen lisäosat ajetaan järjestelmäkäyttäjänä. Windows XP:llä tällä käyttäjällä oli laajat oikeudet, ja se pystyi toimimaan käyttäjän työpöydällä. Windows 7:llä näitä oikeuksia on supistettu, eikä järjestelmäkäyttäjä pysty toimimaan käyttäjän työpöydällä, vaan on rajoitettu taustaprosesseihin. Työaseman lukitus luetaan käyttäjän työpöydällä toimimiseksi, joten kun CardPoller ajaa työaseman lukitsevan Windowsin API-funktion, ei sillä ole mitään vaikutusta. CardPoller luulee kuitenkin suorittaneensa tehtävänsä, ja lopettaa toiminnan. Tällöin käyttäjän täytyisi lukita työasema Win+L – näppäinyhdistelmällä, joka on vastoin järjestelmän ajateltua toimintaperiaatetta, haitaten sen helppokäyttöisyyttä, sekä turvallisuutta ja käyttäjien valvomista.

Oikeanlaisen toiminnan varmistamiseksi tuli CardPoller ajaa varsinaisen käyttäjän oikeuksilla. Ohjelmallisesti tämä onnistuisi .NET:in System.Diagnostics.Process.Start-metodin yhdellä ylikuormituksella, joka ottaa parametreina käynnistettävän sovelluksen polun lisäksi käyttäjänimen, salasanan ja domainin [10]. Prosessi käynnistetään näillä käyttäjätunnuksilla, ja se kykenee silloin toimimaan halutulla tavalla.

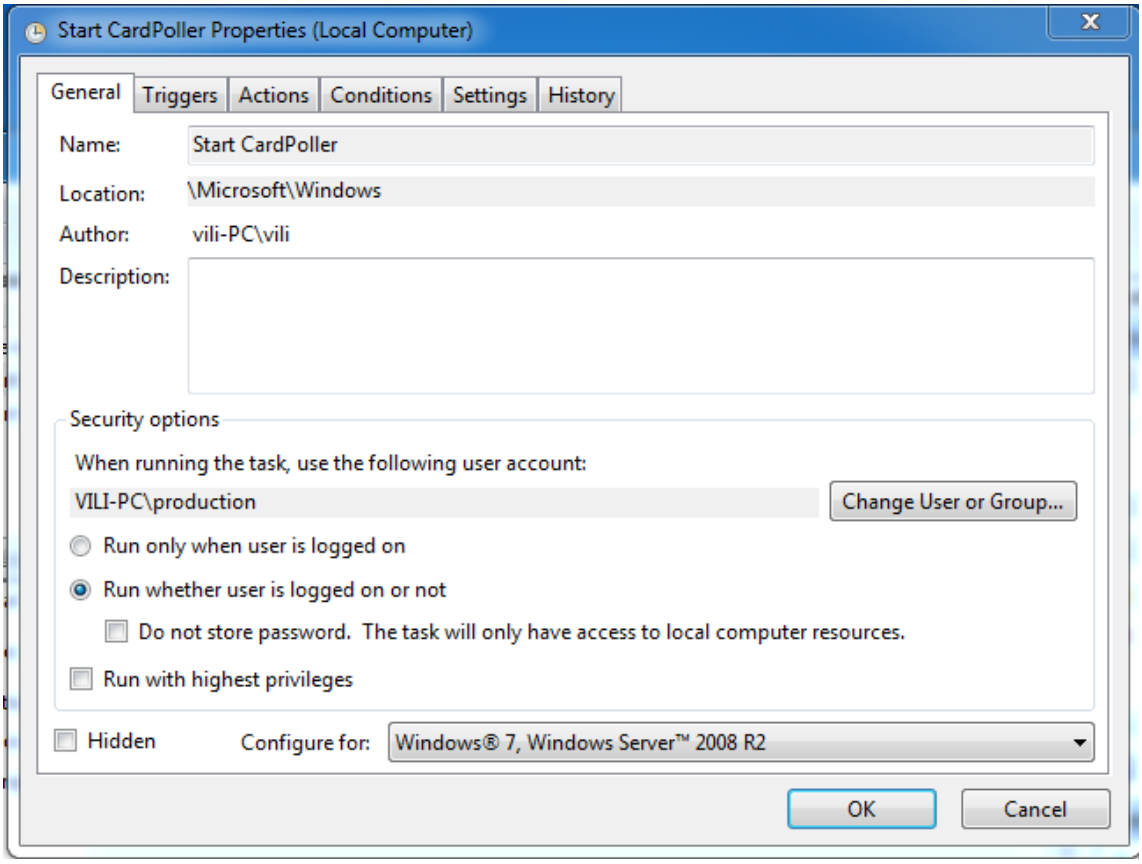
pGina käsittelee jo muutenkin käyttäjätunnuksia, joten sen sisällä prosessin käynnistäminen olisi ollut kätevää, mutta testaamalla huomattiin, että eri käyttäjänä prosessin käynnistäminen onnistuu vasta kun tunnistautuminen on tapahtunut. Tästä johtuen prosessin käynnistämisen tulisi tapahtua erillisessä, esimerkiksi StartCardPoller-nimisessä, prosessissa. Tämän toteutustavan haittana on huono käyttäjätunnusten turvallisuus.

Käyttäjätunnusten StartCardPoller-sovellukselle toimittamiseen olisi ollut kaksi tapaa. Ensimmäisessä tavassa tunnukset, etenkin salasana, tulisi olla varastoituna salattuna johonkin rekisteriin, josta ne tulisi noutaa, purkaa salaus ja käyttää prosessin käynnistämiseen. Itse toteutettu ratkaisu tunnusten varastointiin ja käsittelyyn ei kumminkaan tuntunut miellyttävältä ajatukselta.

Toisessa tavassa StartCardPollerin tulisi olla automaattisesti käynnistyvä Windows-palvelu, jolle tunnukset välitetään nimetyllä putkiyhteydellä. StartCardPoller toimisi putkiyhteydelle palvelimena, ja Single User Login -lisäosa toimisi asiakkaana. Single User Login -lisäosa avaisi yhteyden StartCardPolleriin ja lähettäisi putkella tunnusten käsittelyn yhteydessä ne StartCardPollerille, joka käyttäisi niitä CardPollerin käynnistämiseen, kun käyttäjän kirjautuminen on onnistunut. Tämä vaikutti potentiaaliselta ratkaisulta, mutta olisi vaatinut paljon muutoksia ja testaamista, joten etsittiin ratkaisua vielä muualta.

Windowsin Task Scheduler on monipuolinen työkalu, jolla voi suorittaa toimenpiteitä asetettujen liipaisimien mukaan. Liipaisimet ovat monipuolisia ja ne voivat perustua aikaan, säännöllisesti tapahtuviin tapahtumiin, kuten sisäänkirjautumistapahtumaan, tai spesifisempiin tapahtumiin. Spesifisemmällä tapahtumalla tarkoitetaan tapahtumia, joita sovellukset voivat raportoida tapahtumalokiin. Näitä tapahtumia voidaan eritellä tunnistenumeron perusteella ja niiden perusteella voidaan toimia. Vaihtoehtoja Task Schedulerin suorittamaksi toimenpiteeksi on kolme:

- Näytä viesti.
- Lähetä sähköposti.
- Aja sovellus.

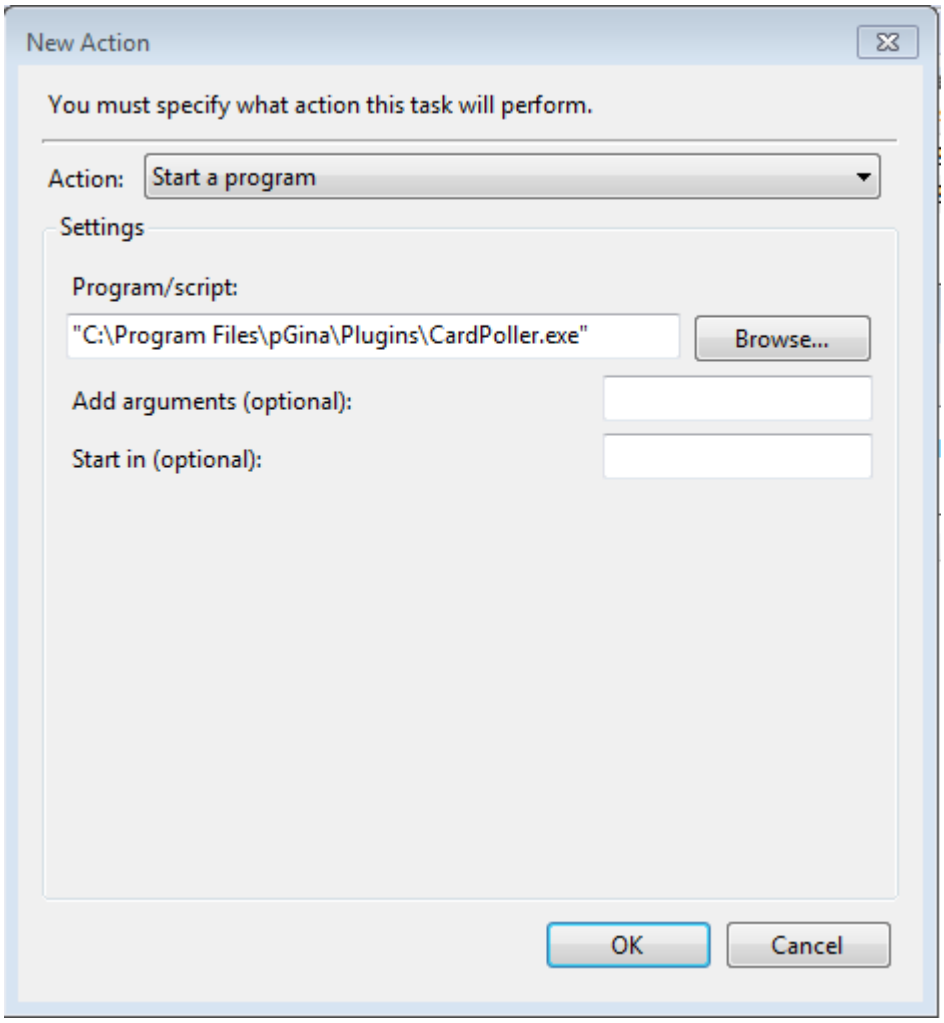


Kuva 14. Kortin poistoa valvovan sovelluksen käynnistävän tehtävän yleisten asetusten säätämisikkuna.

Näistä viimeinen oli järjestelmän kannalta kiinnostavin. Task Schedulerilla voi tehtävää luodessa määritellä käyttäjän, jonka oikeuksilla suoritettava tehtävä ajetaan. Tämä tarkoittaa sitä, että Task Schedulerin avulla voidaan määritellä ohjelma ajettavaksi joko ylläpitäjän tai tuotantokäyttäjän tunnuksilla, kuten kuvassa 14 esitetyssä ikkunassa on tehty, jolloin esimerkiksi CardPoller kykenee työpöydän lukitsemiseen.

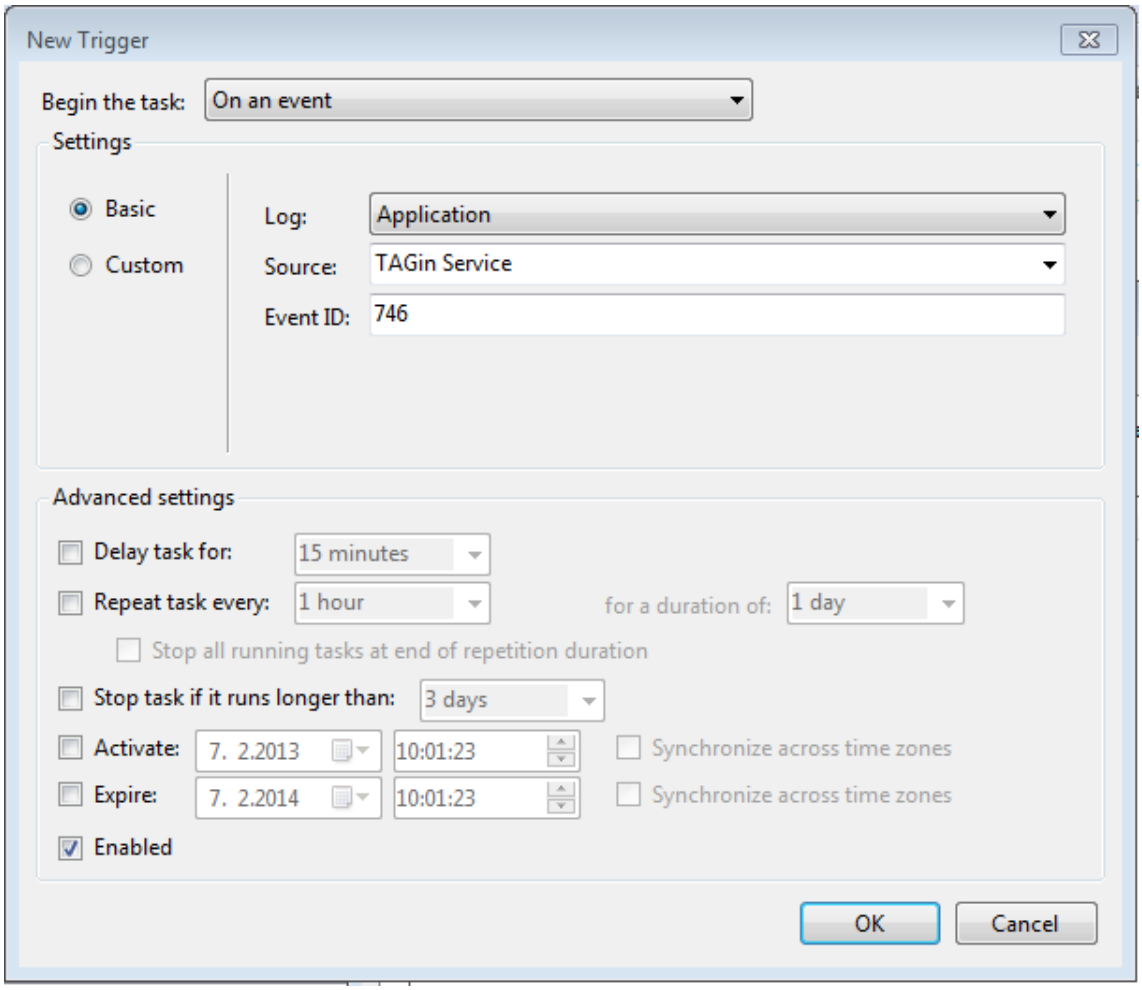
TAGin Service kirjaa tapahtumia tapahtumalokiin, joiden perusteella Task Scheduler ajaa ohjelmia. Näin saadaan yhdistettyä ohjelmallinen toiminnanvalvonta ja valvonta- ja PINin vaihtosovelluksen oikeiden ajo-oikeuksien saaminen. TAGin Servicellä ohjelmien käynnistäminen Task Scheduleria käyttäen on jopa helpompaa kuin se olisi suoraan ohjelmallisesti, sillä Task Scheduler toimii eräänlaisena abstraktiokerroksena. Kun Task Schedulerin tehtävä on konfiguroitu oikein, TAGin Servicen tulee CardPollerin käynnistääkseen vain kirjata työaseman Windowsin omaan sisäiseen tapahtumalokiin tapahtuma tietyllä tunnistenumerolla, ja Task Scheduler hoitaa loput. Palvelun ei täten tar-

vitse tietää suoritettavan sovelluksen polkua tai kenenkään käyttäjän tunnuksia, sillä ne on asetettu Task Schedulerissa. [11.]



Kuva 15. CardPoller-ohjelman käynnistävä toiminto Task Schedulerissa

Kuvasta 15 voidaan nähdä, kuinka Task Schedulerin tehtävään asetetaan toiminnoksi CardPoller-ohjelman käynnistys. Ohjelma ajetaan tehtävän asetuksissa määrätyllä käyttäjätilillä. Käynnistettävälle ohjelmalle voi myös halutessaan antaa argumenttejä sekä asettaa sen kotihakemiston muuksi kuin sen sijaintihakemistoksi.

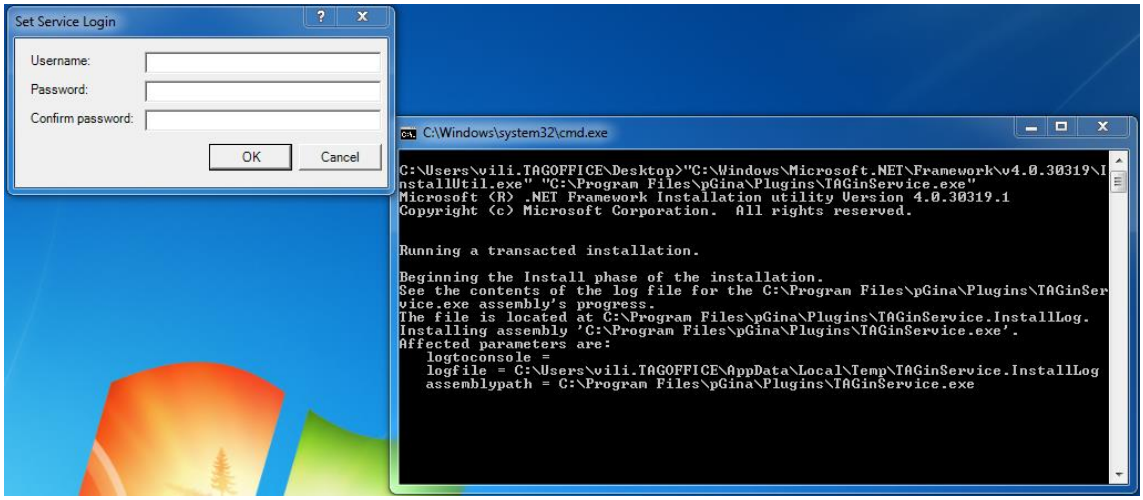


Kuva 16. CardPoller-ohjelman käynnistävän toiminnon liipaisimen asettaminen.

Kuvasta 13 voidaan nähdä, kuinka asetetaan liipaisin. Liipaisin tarkkailee Windowsin lokeja, ja kun Application-lokiin ilmestyy tapahtuma lähteestä TAGin Service tunnistenumerolla 746, laukeaa liipaisin. Kun liipaisimeen yhdistetään CardPollerin käynnistävä toiminto, voidaan tätä käyttää sen käynnistämiseen halutulla käyttäjätilillä.

Jotta Windows-palvelua voidaan käyttää, se täytyy erikseen asentaa koneelle. Tästä johtuen sovellusta kirjoitettaessa on siihen liitettävä asennin (installer). Asentimen asetuksista määritetään palvelun eri ajo-ominaisuuksia, kuten mikä on sen käynnistystapa, millä oikeuksilla se ajetaan ja millä nimellä se tunnetaan käyttöjärjestelmässä [12]. Palvelun asentamiseksi tulee ajaa sovelluksen luomiseen käytetyn .NET-version mukainen InstallUtil-sovellus [13]. TAGin Servicen tapauksessa sovellus on luotu .NET Frameworkin versiolla 4.0, joten ajettava InstallUtil.exe sijaitsee C:\Program Files\Microsoft.NET\Framework\v4.0.30319\ -kansiossa. Koska TAGin Servicen asenti-

messa on määritelty, että se ajetaan käyttäjän oikeuksilla, tulee asennuksessa syöttää käyttäjätunnukset, joilla palvelu ajetaan. Kuvasta 17 voidaan nähdä, miltä tämä asennusdialogi näyttää ajettuna.



Kuva 17. TAGin Servicen asennusdialogi

4 Jatkokehitysmahdollisuudet

Vaikka toiminnallisuuden ennalta määritelty tavoite saatiin toteutettua, on järjestelmälle useita jatkokehitysmahdollisuuksia, kuten biometrisen tunnistuksen lisääminen, asentimen luominen sekä käyttäjien palautteen mukaan tehtävät parannukset kun järjestelmä on viety tuotantoympäristöön. Nämä jatkokehitysmahdollisuudet takaavat, että järjestelmän kehitys teettää vielä töitä.

Biometrisen tunnistuksen mahdollistava sovelma älykortille on jo olemassa ja palvelimen rajapinta on suunniteltu sitä ajatellen, joten sen toteuttaminen vaatii vain ominaisuuden toteuttamista työaseman ohjelmistoihin. Biometrinen tunnistautuminen alkaa siitä, että käyttäjän sormenjälki luetaan skannerilla. Sormenjäljen kuvasta erotellaan tunnistuspiirteet ja niistä muodostetaan varmennemalline, joka lähetetään älykortille. Mallinetta verrataan kortilla kortille tallennettua mallinetta vasten, ja jos vastaavuus ylittää ennalta määritetyn raja-arvon, kortti palauttaa varmennepoletin, joka lähetetään palvelimelle. Koska palvelimen rajapinta on sama kuin PINiä käytettäessä, tarkastus

tapahtuu palvelimella samalla tavalla. Varmennepoletti sisältää tiedon siitä, käytettiinkö tunnistautumisessa PINiä vai sormenjälkeä.

Asennin koko järjestelmälle helpottaisi järjestelmän ylläpitoa tekemällä asennuksen ja päivittämisen helpommaksi, kun kaikkien komponenttien asentaminen voitaisiin hoitaa ajamalla vain yksi tiedosto. Jos järjestelmä päätetään tuotteistaa, on se myös tärkeä osa viimeistelyä ja ammattimaista kokonaisilmettä. Asennin on mahdollista luoda Microsoftin Visual Studiolla tai ulkopuolisella työkalulla, kuten Inno Setupilla. Asentimen luomisessa Visual Studiolla on etuna se, että asennin voidaan pitää projektina ratkaisussa, joka sisältää koko TAGin-järjestelmän, ja sen käyttäminen on melko yksinkertaista. Inno Setup puolestaan on monimutkaisempi, mutta sillä on mahdollista luoda erittäin monipuolisia ratkaisuja. Kun asenninta lähdetään luomaan, on punnittava näiden vaihtoehtojen etuja ja haittoja.

Kun järjestelmä lopulta viedään tuotantoympäristöön, se tulee ensimmäistä kertaa olemaan laajassa ja monipuolisessa rasituksessa. Tästä johtuen siinä tulee todennäköisesti ilmenemään ongelmia ja puutteita, joita ei ole aikaisemmassa testauksessa tullut ilmi. Lisäksi käyttäjiltä tulee luultavasti parannusehdotuksia järjestelmään. Monet Windowsin sisäisistä ratkaisuista johtuvat ongelmat onnistuttiin ratkaisemaan käyttämällä Windowsin työkaluja, mutta tulevaisuudessa voisi myös tutkia onnistuuko näiden työkalujen tekemät toimenpiteet hoitaa ohjelmallisesti. Ongelmien ja ehdotusten korjausten toteuttaminen voi jatkua koko järjestelmän elinajan.

5 Päätelmät

Työn tavoitteena oli luoda käyttäjänhallintajärjestelmä PK-yrityksen tuotannon tarpeisiin. Kehitysvaiheessa kävi ilmi, että pGina ja sen lisäosarajapinta tarjoavat joustavan ja laajennettavan tavan toteuttaa tunnistautuminen työasemalle kirjautuessa. Se mahdollistaa helposti oman tunnistautumislogiikan toteuttamisen. Suurimpia ongelmia aiheuttivatkin Windowsin sisäisten ratkaisujen kanssa kamppailu, jotka yllättivät siinä vaiheessa, kun järjestelmän kehitys käännettiin Windows 7:n tähtääväksi.

Työn tuloksena syntyi järjestelmä, joka täyttää ennalta asetetut vaatimukset ja joka on valmis vietäväksi tuotantoon testaukseen. Järjestelmä kykenee toiminnallisuudellaan korvaamaan aikaisemmin käytössä olleen nk. screenlocker-ohjelman. Sen avulla voidaan hallita käyttäjiä keskitetysti ja pitää lokia työasemien käytöstä. Järjestelmän tuotantoon vieminen aloitettaneen, kun tuotannon koneet on päivitetty Windows 7 -käyttöjärjestelmään.

Lähteet

1 Engdahl, Kim. 2012. Vanhempi järjestelmäasiantuntija. TAG Systems Finland Oy, Vantaa. Keskustelu 15.9.2012.

2 C# Language Specification Version 4.0. 2010. Microsoft Corporation.

3 About Mono. 2012. Verkkodokumentti. Novell. < http://www.mono-project.com/Main_Page>. Luettu 11.2.2013.

4 Dotnetpowered Language List. 2012. Verkkodokumentti. Dotnetpowered.com. <http://www.dotnetpowered.com/languages.aspx>. Luettu 11.3.2013.

5 Overview of the .NET Framework. 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/vstudio/zw4w595w.aspx>. Luettu 11.2.2013.

6 Unsafe. 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/chfa2zb8%28v=vs.71%29.aspx>. Luettu 11.3.2013.

7 Developing Pgina Plugins. 2012. Verkkodokumentti. Pgina Team. <http://pgina.org/docs/v3.1/plugins.html>. Luettu 11.3.2013.

8 Platform Invoke Tutorial (C#). 2003. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/aa288468%28v=vs.71%29.aspx>. Luettu 11.2.2013.

9 DllImportAttribute.SetLastError Field. 2010. Verkkodokumentti. Microsoft. < http://msdn.microsoft.com/en-us/library/system.runtime.interopservices.dllimportattribute.setlasterror%28v=vs.71%29.aspx>. Luettu 11.2.2013.

10 Process.Start Method (String, String, SecureString, String). 2010. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/sxf2saat%28v=vs.100%29.aspx>. Luettu 11.2.2013.

11 Task Scheduler (Windows). 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/windows/desktop/aa383614%28v=vs.85%29.aspx>. Luettu 11.2.2013.

12 How To: Add Installers To Your Service Application. 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/ddhy0byf.aspx>. Luettu 11.2.2013.

13 How To: Install and Uninstall Services. 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/sd8zc8ha%28v=vs.110%29.aspx>. Luettu 11.2.2013.