Java Card aplet kao orkestrator komponenti

Danijel Venus, Petar Bjeljac, Goran Čeko, Branko Perišić
Katedra za informatiku/Departman za računarstvo i automatiku
Fakultet tehničkih nauka
Novi Sad, Republika Srbija
danijelvenus@gmail.com, pbjeljac@uns.ac.rs, gceko@uns.ac.rs, perisic@uns.ac.rs

Sadržaj—**U radu je analizirana Java Card tehnologija i konceptualna realizacija aplikacije kao kompozicija komponenti, kao i primer jedne takve realizacije u kontekstu GlobalPlatform specifikacije. Kompozicija komponenti se oslanja na radni okvir (Service framework). Poseban akcenat je stavljen na razvoj pseudo fajl sistema, sigurnosnih mehanizama i servisa personalne prirode na kojima se dalje temelje Java Card bazirani servisi (aplikacije).**

Ključne riječi-java card; smart card; csf; global platform

I. UVOD

Savremeni pristup servisima u domenu informacionih tehnologija se sve više oslanja na aktivne komponente koje korisnici nose sa sobom. Smart kartice predstavljaju tipičan primer. Pod smart karticama se podrazumevaju kartice koje poseduju „čip“. Na osnovu vrste sadržanog čipa dele se na: kartice sa memorijskim čipom i kartice sa mikrokontrolerom. Kartice sa memorijskim čipom se dele na one koje poseduju i one koje ne poseduju sigurnosnu logiku. Kartice sa mikrokontrolerom mogu posedovati sporedni mikrokontroler (numerički koprocesor) namenjen za podršku kripto-zaštiti, sa danas najrasprostranjenijim osloncem na asimetrične kriptografske algoritme.

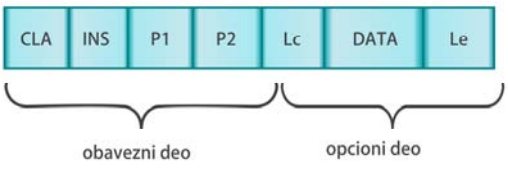
S obzirom na domen primene kod smart kartica je esencijalno postojanje standarda čijom doslednom primenom se omogućava korišćenje, u sklopu istog sistema, kartica različitih proizvođača kao i kartica koje su proizvedene uz oslonac na različite tehnologije. ISO/IEC-7816 (1-15) predstavlja jedan od najzastupljenijih specifikaciju u ovoj oblasti. Delovi 1 i 2 ove specifikacije posvećeni su o fizičkim i električnim karakteristikama a deo 3 transmisionim protokolima. Komunikacija sa karticom se na aplikacionom nivou obavlja putem razmene APDU (Application Processing Data Unit) komandi i zasnovana je na klijent-server arhitekturi.

Na nižem, transmisionom nivou, APDU se najčešće prenosi korišćenjem dva protokola označena kao T0 (bajt) i T1 (blok). Zbog svoje prirode T0 protokoli su jako zavisni od aplikacionog sloja dok je protokol T1 potpuno transparentan. U konkretnim primenama zbog svoje jednostavnosti T0 protokol je dominantan.

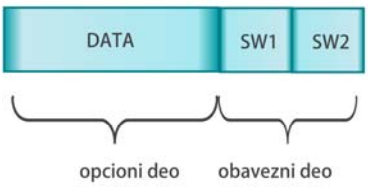
Komunikacija je zasnovana na konceptu komanda-odgovor. Za svaku komandu koju CAD (Card Accepting Device) pošalje kartici, neophodan je povratni odgovor. C-APDU su komande koje se šalju prema kartici i njihov format je prikazana na Slici. 1.

- *CLA* polje je dužine 1 bajt i sadrži informacije o identifikatoru logičkog kanala kroz koji se šalju komande (bitovi b2,b1), indikatoru da li je u pitanju sigurnosna komanda (bitovi b4,b3) i samoj vrednosti klase komande kodiranoj u bitovima b8-b5 Identifikator logičkog kanala je ceo broj u intervalu [0,3].
- *INS* polje je dužine 1 bajt i predstavlja kôd instrukcije komande.
- *P1* i *P2* su dužine 1 bajt svaki i predstavljaju dodatne parametre same komande.
- *Lc* polje je dužine 1 bajt ako postoji i predstavlja broj bajtova u *DATA* polju.
- *DATA* polje je dužine *Lc* bajtova i predstavlja podatke komande.
- *Le* polje je dužine 1 bajt ako postoji i predstavlja maksimalni (očekivani) broj bajtova koje kartica nakon obrade komande vraća klijentu.

R-APDU su komande koje kartica vraća kao odgovor. Format ovih komandi je prikazan na Slici 2.



Slika 1. Format C-APDU komande



Slika 2. Format R-APDU komande

- *DATA* polje je maksimalno dužine *Le* bajtova i predstavlja podatke odgovora ako postoje.
- *SW1* i *SW2* su dužine 1 bajt svaki i predstavljaju status obrade C-APDU komande koji može biti iz klasa

uspešno obrađenih, obrađenih sa upozorenjem i neobrađenih sa greškom.

JavaCard™ specifikacija [1] opisuje pravila funkcionisanja JVM (Java Virtual Machine) na kartici kao i celokupno izvršno okruženje JCRE (Java Card Runtime Environment). U glavne elemente koji su opisani JCRE specifikacijom spadaju:

- Životni vek JVM. JVM nikad ne prestaje sa radom. Pri prestanku napajanja kartice iz CAD uređaja, JVM prelazi u pasivno stanje, čekajući ponovo kontakt sa CAD.
- Životni vek Java Card aplikacija (apleta).
- Selekcija apleta i logički kanali komunikacije.
- Definicija dve vrste objekata u aplikaciji (perzistentni i tranzijentni).
- Kontekst izvršavanja apleta i firewall mehanizmi
- Transakcioni mehanizmi
- Specifikacija RMI (Remote Method Invocation) servisa
- Instalacija i deinstalacija apleta.

Pored *JavaCard*™ postoje i druge dve dominantne platforme: *Windows® Powered Smart Cards* i *MULTOS*™.

Za razliku od *JavaCard*™, *GlobalPlatform*® specifikacija [2] se temelji na propisivanje načina upravljanja karticom. Ona definiše način funkcionisanja više aplikacija različitih dobavljača na istoj kartici uvođenjem novog elementa nazvanog – *Security Domain*. Svaka instalirana aplikacija na kartici ima instancu sigurnosnog domena koji predstavlja dobavljača aplikacije ili izdavača kartice. Sigurnosni domeni u ime aplikacije izvršavaju „osetljive“ operacije (otvaranje sigurnosnih kanala, obrada sigurnosnih komandi, postavljanje statusa aplikacija, učitavanje novih i brisanje postojećih aplikacija ili modula sa kartice, ažuriranje ključeva i sl.).

II. JAVA CARD

Razvoj Java Card aplikacija se ne razlikuje bitno od razvoja standardnih Java aplikacija. Apleti i ostali referentni elementi, koji zajedno čine izvršivi modul na kartici, se pišu na Java programskom jeziku. Vrši se standardna kompilacija izvornih datoteka odgovarajućim Java kompajlerom. Konverter zatim vrši konverziju *class* datoteka u jednu datoteku sa ekstenzijom cap. Cap datoteka je po formatu identična jar datoteci i poseduje dovoljno informacija da bi se učitala na karticu. Postoji i implementacija konvertera koji se dobija zajedno sa JCDE (Java Card Development Environment) razvojnim okruženjem.

Resursi kartice uslovljavaju niz ograničenja prilikom pisanja Java Card aplikacija. Najvažnija ograničenja obuhvataju: ne postojanje dinamičkog učitavanja klasa, niti, kloniranja i finalizacije objekata. Podskup ključnih reči (*native*, *synchronized*, *transient*, *volatile*, *strictfp*), kao i podskup tipova (*char*, *double*, *float*, *long* i *int)* nisu podržani ili predstavljaju opcije. Mehanizam brisanja nekorišćenih objekata (*Garbage Collector*) je opciono podržan preko metode *JCSystem.requestObjectDeletion*. Mehanizmi uvedeni od verzije Java 1.5 (generički tipovi, anotacije i enum) uglavnom nisu podržani [3].

Apleti primaju APDU komande, obrađuju ih i klijentu šalju odgovor. Pre bilo kakve komunikacije sa apletom, on se mora selektovati na nekom od otvorenih logičkih kanala primenom ISO/IEC-7816 SELECT komande. Razvijani aplet mora naslediti klasu *javacard.fremwork.Applet* i definisati statičku metodu *install* koja se poziva samo jednom od strane JCRE u toku instalacije apleta na karticu. U *install* metodi se mora instancirati dati aplet i pozvati njegova metod *register* koja registruje instalirani aplet u izvršnom okruženju. Nakon uspešne registracije apleta, on postaje sposoban za selekciju i obradu komandi. Obrada komandi se vrši preko metode *process* koja prima objekat APDU komande.

JCRE poseduje sigurnosne mehanizme za izvršavanje pojedinih apleta. Svi apleti iz istog paketa dele jedan kontekst izvršavanja koji je ograničen mehanizmom zaštitne barijere (*firewall*). Objekat iz jednog paketa ne može pristupati čak ni javnim metodama i poljima objekta drugog paketa. Ovaj mehanizam provere se vrši na nivou JVM čime se sprečava narušavanje integriteta funkcionisanja aplikacija iz jednog paketa instaliranjem „*zlonamernih*“ apleta na kartici. Ukoliko je neophodno ostvariti određen nivo komunikacije između dva apleta iz različitih konteksta izvršavanja, moguće je drugom apletu izvesti određen broj metoda preko deklarisanog interfejsa, koji nasleđuje *javacard.framework.Shareable*. Izvoženje kontroliše sam aplet koji izvozi dati objekat i poseduje mogućnost provere ko traži objekat, tako da je moguće i odbiti zahtev u slučaju ako ciljni aplet nije „prepoznat“. Zahtev za dobijanje interfejsa se izvodi preko metode *JCSystem.getAppletShareableInterfaceObject*.

Vek trajanja objekata je realizovan obrnuto od standardnog Java izdanja. Svi objekti kreirani na standardni način su perzistentni u memoriji. Memorija koja se koristi za ovakve slučajeve je EEPROM (Electrical Erasable Programmable Read Only Memory) i eventualno ROM (Read Only Memory) za slučajeve fabričke inicijalizacije objekata na kartici. Da bi se kreirao objekat tranzijentne prirode koristi se metoda sa prefiksom u nazivima *JCSystem.makeTransient...* Tranzijentni objekti nisu u punom smislu reči nepostojani, već samo njihov sadržaj. Objekat je perzistentan, ali njegov sadržaj nije i on se resetuje na inicijalne vrednosti (byte na 0 i sl.) prilikom ispunjenja određenih uslova. Postoje dve vrste tranzijentnih objekata: objekti čiji se sadržaj resetuje prilikom resetovanja kartice i objekti čiji se sadržaj resetuje prilikom deselekcije aplikacije. Sadržaj tranzijentnih objekata se obično nalazi u RAM memoriji kartice.

Specifikacija takođe definiše transakcione mehanizme preko metoda *JCSystem.beginTransaction* i *JCSystem.commitTransaction.* Dovoljno detaljno je specifikovan način na koji se potvrđuje transakcija ili povlači. Unutar transakcije svi perzistentni objekti su uslovno ažurirani. Transakcija se ne povlači kompletno u trenutku ako na primer prestane napajanje kartice usled izvlačenja iste iz CAD terminala, već se prilikom sledećeg kontakta sa CAD terminalom vrši ažuriranje sadržaja odnosno samo povlačenje transakcije pre bilo kakvih drugih obrada pristiglih APDU

komandi koje bi potencijalno narušile integritet objekata. Tranzijentni objekti ne učestvuju u transakciji. Transakcija je ograničena memorijskim sposobnostima kartice. Provera koliko je prostora ostalo se može obaviti pozivom metoda *JCSystem.getMaxCommitCapacity* i *JCSystem.getUnusedCommitCapacity*.

III. JAVA CARD SERVICE FRAMEWORK

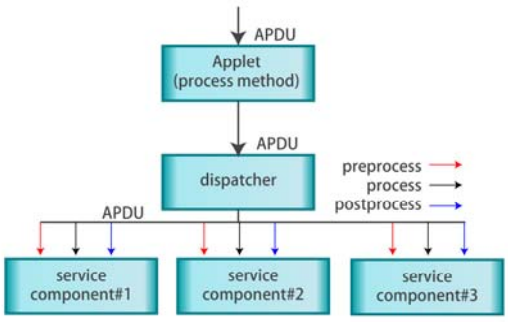
Java Card raspolaže sa jednostavnim radnim okvirom za građenje apleta agregacijom servis komponenti. Servis komponenta implementira interfejs *javacard.framework.service.Service*.

Interfejs poseduje metode za obradu APDU komandi, pre-procesiranje i post-procesiranje komandi. Postoji i pogodna klasa *BasicService* koja realizuje uobičajene poslove rukovanja sa APDU komandama odnosno definisanja CSF (*Common Service Format*) podataka u APDU baferu. Agregiranje servis komponenti se obavlja pomoću klase *Dispatcher*. Klasa je takođe zadužena za prosleđivanje komandi servis komponentama.

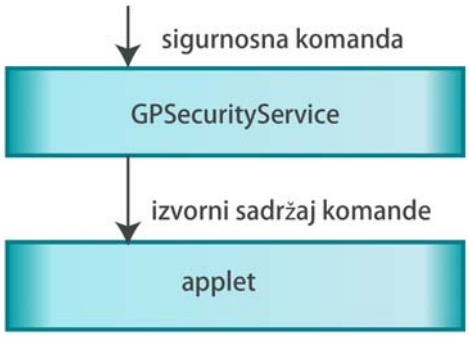
Orkestracija servis komponenti je urađena u formi *pipeline* arhitekture. Prvi servis u nizu prima komandu i ako je prepozna obrađuje je i signalizira dispečera da je komanda obrađena. U suprotno komanda se prosleđuje sledećoj servis komponenti u nizu. Na Slici 3. je prikazan tipičan primer servis orijentisane aplikacije.

Jedina implementacija servis komponente koja je uključena u standardnu Java Card ediciju jeste *RMIService*. Komponenta na klijentskoj strani *stub* vrši prevođenje poziva Java metode u APDU komandu koja stiže preko aplikacije na kartici do *RMIService* komponente koja isti APDU transformiše u poziv metode klase koja implementira *rmi* interfejs na kartici.

Korisna posledica servis komponenti je implementacija komponente *GPSecurityService* koja vrši sigurnosne komunikacije kroz komunikacioni kanal između klijenta i kartice u ime apleta (Slika 4.).



Slika 3. Servis orijentisan aplet



Slika 4. Transparencija sigurnosnih komandi

Jednom implementirana servis komponenta, može se koristiti u svim apletima kojima je potrebno otvaranje sigurnosnog kanala sa klijentom. Samo otvaranje kanala i sigurnosna komunikacija sa strane kartice se oslanja na korišćenje API GlobalPlatform standarda čija se implementacija nalazi na kartici.

Ovakvu komponentu treba uvezati pre svih ostalih kako bi obavila neophodno pre-procesiranje komandi (u zavisnosti od nivoa sigurnosti, odmotavanje komandi koja je šifrirana i sl.). Na taj način razvoj samog apleta i komunikacija sa klijentom postaje transparentna u odnosu na sigurnosne mehanizme.

Pristupanje servisima pridruženog domena aplet obavlja uz oslonac na metode klase *org.globalplatfrom.GPSystem* i, specijalno za radnje sa sigurnosnim kanalom, preko interfejsa *SecureChannel* koji se dobija preko klase *GPSystem*. Značajne metode interfejsa *SecureChannel* su: *processSecurity* koji procesira komande iz GlobalPlatform specifikacije relevantne za rukovanje sa sigurnosnom komunikacijom, *wrap* koji umotava komande pre slanja kao odgovora klijentu, *unwrap* koji odmotava komande pre procesiranja od strane ostalih servis komponenti. Odmotavanje vrši dekripciju sadržaja ako je on bio pre toga kriptovan, proveravanje integriteta podataka i autentičnosti. Umotavanje radi slične stvari u zavisnosti od traženog sigurnosnog protokola.

IV. PRIMER JAVA CARD APLIKACIJE UPOTREBOM SERVIS KOMPONENTI

Kao primer agregiranja servis komponenti odabran je razvoj pseudo fajl sistema i komponente za pružanje servisa personalne prirode.

Postoji skup komandi iz ISO/IEC-7816 standarda koji se bave manipulacijom fajl sistemom kartice. Takođe isti standard definiše i organizaciju samog fajl sistema. Ovakav fajl sistem Java Card kartice nije direktno vidljiv krajnjem klijentu iz bezbednosnih razloga [4]. Pošto se sami apleti i drugi osetljivi podaci takođe nalaze u sklopu fajl sistema, kao i sadržaj aplikacije u vreme izvršavanja, direktna manipulacija sa fajl sistemom, čak i sa sigurnosnom politikom zaštite pristupa pojedinim datotekama, nije preporučljiva. Zbog toga JCRE „maskira“ ceo fajl sistem i klijentu nudi samo mogućnost selekcije aplikacije i vršenje komunikacije uz oslonac na APDU komande.

Realizacija fajl sistema je urađena kao servis komponenta *JCFileSystemService* tako da ju je moguće inkorporirati u bilo koji aplet koji zahteva skladištenje podataka. Pošto je preporučljiva praksa da se celokupni memorijski prostor apleta (a time i prostor servis komponente) unapred alocira, to radi i ova komponenta. Dodavanje datoteka i logički brisanje datoteka i sadržaja je realizovana logički. Fajl sistem je organizovan hijerarhijski sa *root* datotekom na vrhu. Svaka datoteka sadrži skup zapisa *records* jednake ili varijabilne dužine. Takođe svaka datoteka referencira skup drugih datoteka. Pored same definicije strukture, datoteka poseduje pravila vezana za pristup: privilegije za čitanje, pisanje i postavljanje statusa (logički brisanje i dodavanje) zapisa i datoteke. Struktura i pravila se definišu pri samom instanciranju servisa, kao i alokacija potrebnog memorijskog prostora. U Tabeli I. je prikazan skup komandi koje ova servis komponenta prepoznaje. Pošto je struktura hijerarhijska, referenciranje pojedinih zapisa i datoteka se vrši preko odgovarajućih putanja u okviru *root* datoteke.

READ RECORD komanda vrši čitanje sadržaja jednog zapisa. Vrednosti parametara *P1* i *P2* određuju proces čitanja (inicijalizacija čitanja na zadatu putanju, sledeći blok podataka zapisa i slično).

UPDATE RECORD komanda vrši ažuriranje sadržaja jednog zapisa. Vrednosti parametara *P1* i *P2* određuju proces ažuriranja (inicijalizacija ažuriranja, sledeći blok prenosa podataka na karticu, potvrda ažuriranja).

GET INDICES komanda vraća statuse svih zapisa, ili svih datoteka u zavisnosti od vrednosti parametra P1*, referencirane datoteke.*

SET STATUS komanda vrši postavljanje statusa zapisa ili datoteke u zavisnosti od vrednosti parametra *P1* i *P2*. Status je logički aktivan ili logički pasivan.

Prilikom izvršenja ovih komandi proverava se stanje logičkog kanala komunikacije. Ako stanje ispunjava uslove definisane u referenciranoj datoteci, komanda se uspešno izvršava, ako ne ispunjava, vraća se odgovor sa statusom greške. Skup pravila za pristup datoteci je neka od sledećih:

- Nema pravila (slobodan pristup datoteci).
- Klijent je autentikovan.
- Klijent je autentikovan i komunikacija zadržava integritet prenetih podataka.
- Klijent je autentikovan i komunikacija zadržava integritet i tajnost prenetih podataka.
- Kombinacija sa verifikovanim vlasnikom kartice za prethodna četiri slučaja (PIN-Personal Identification Number verifikovan).

TABELA I. SPISAK KOMANDI *JCFILESYSTEMSERVICE* KOMPONENTE

	CLA (hex)	INS (hex)
READ RECORD	0x80	0x02
UPDATE RECORD	0x80	0x04
GET INDICES	0x80	0x06
SET STATUS	0x80	0x08

Provera se oslanja na *GlobalPlatform* API. Nivo sigurnosti kanala se proverava metodom *getSecurityLevel* interfejsa *SecurityChannel*, dok se verifikacija PIN vrednosti ili provera da li je PIN verifikovan u trenutnoj sesiji između CAD terminala i kartice vrši metodom *verify* i *isVerified* interfejsa *CVM* (Card Holder Verification Method) kojim se pristupa preko klase *GPSystem*.

Agregacijom servis komponente *JCFileSystemService* i *GPSecurityService* se dobija aplikacija koja ima sposobnost pružanja usluga čitanja i skladištenja podataka kako „neosetljive“ tako i „osetljive“ prirode.

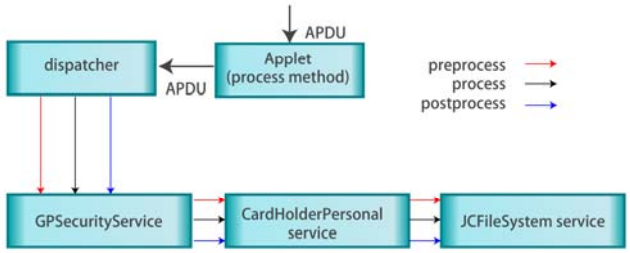
Pod servisima personalne prirode koje su najčešće zastupljene u domenu korišćenja smart kartica jesu: upravljanje PIN-om kao mehanizmom za garanciju da lice kao reprezentativ kartice ima pristup osetljivim radnjama koje su personalno zavisne, digitalno potpisivanje dokumenata, enkripcija i dekripcija simetričnim ključem i digitalna verifikacija dokumenata kao potvrda da je zaista dokument digitalno potpisan sa datom karticom. Iako prva tri navedena servisa moraju isključivo da budu dostupna sa kartice, treći tip servisa može biti lociran i u klijentskom sloju pod uslovom da za verifikaciju koristi sertifikat sa kartice. Za realizaciju ova tri tipa servisa uvedena je komponenta *CardHolderPersonalService*. APDU komande definisane ovom komponentom su:

- VERIFY_PIN - verifikacija PIN-a unetog preko terminala.
- UPDATE_PIN - ažuriranje vrednosti PIN-a na kartici.
- TRIES_REMAINING - broj preostalih pokušaja verifikacije PIN-a.
- IS_PIN_VERIFIED - provera da li je pin verifikovan.
- PUT_KEY - ažuriranje vrednosti ključeva (simetričnih i asimetričnih) potrebni za enkripciju, dekripciju i digitalno potpisivanje.
- SIGN - komanda za digitalno potpisivanje dokumenata. Dokument se mora u celosti preneti na karticu nizom komandi u formi bajtova.
- VERIFY - komanda za verifikaciju potpisa na kartici, tzv. interna verifikacija potpisa pri čemu se kompletan dokument i potpis šalju na kartici radi verifikacije. Znatno je sporiji od eksterne verifikacije ali je zato sigurniji jer je rizik upada u kanal komunikacije sveden na minimum.
- CIPHER - enkripcija i dekripcija podataka upotrebom simetričnog ključa.
- READ_PUBLIC_KEY - čitanje javnog ključa iako se pristup javnom ključu može izvršiti i preko sertifikata koji je skladišten u pseudo fajl sistemu kartice.

Da bi u praksi funkcionisao servis ovakve personalne prirode potrebno je i implementirati komponentu za rukovanje sigurnosnom komunikacijom između klijenta i kartice. Iako je suštinski ovaj posao već urađen preko implementacije *GlobalPlatform* specifikacije, potrebno ga je uvezati sa *Java Card* preko već pomenute komponente *GPSecurityService*.

Kompletna skica vezivanja ovih komponenti u kompletnu sliku pružanja servisa personalne prirode može se videti na Slici 5.

GPSecurityService ide na prvom mestu radi enkripcije, dekripcije i validacije samih APDU komandi kako bi se postigla transparentnost komunikacionog kanala. Ostale dve komponente mogu ići proizvoljnim redosledom jer nisu u apsolutnoj zavisnosti.



Slika 5. Kompletna slika aplikacije za pružanje personalnih servisa

V. ZAKLJUČAK

Postoji veliki broj standarda, kako industrijskih tako i globalnih, koji diktiraju razvoj u domenu smart kartica. Mnogi među njima moraju garantovati kompatibilnost. Primer su Java Card, GlobalPlatform i ISO/IEC-7816 standardi. Svaki od njih samostalno ne čini infrastrukturu koja može funkcionisati u realnom okruženju.

Java Card definiše aspekte razvoja i funkcionisanja aplikacija za smart card platformu, dok GlobalPlatform propisuje arhitekturu upravljanja karticom i njene sigurnosne aspekte.

Service framework i konstrukcija apleta agregiranjem servis komponenti je fleksibilno rešenje u većini slučajeva razvoja apleta.

Primer je realizacija apleta koji komunikaciju sa klijentom ostvaruju RMI metodom. Jedan način za dodatnu obradu APDU komandi (npr. umotavanje i odmotavanje komandi) je ubacivanjem još jedne servis komponente pored komponente *RMIService*.

Drugi značajan primer radnog okvira servisa je omogućavanje transparentnog razvoja i funkcionisanja apleta u odnosu na sigurnosne mehanizme koje nudi asocirani sigurnosni domen za tu aplikaciju.

LITERATURA

[1] *Sun Microsystems, "Runtime Environment Specification, Java Card™ Platform, Version 2.2.1", California, October 2003.*

[2] *Global Platform, "Card Specification, Version 2.1.1", March 2003.*

[3] *Sun Microsystems, "Virtual Machine Specification, Java Card™ Platform, Version 2.2.1", California, October 2003.*

[4] *W. Rankl, W.Effing, "Smart Card Handbook, Third Edition", © Carl Hanser Verlag, Munich/FRG, 2002.*

ABSTRACT

In this article there are Java Card technology and conceptual realization of application as composition of components based on *Service framework*. Example of such a realization is given in the context of the GlobalPlatform specification. The practical usage is demonstrated on a pseudo file system, personal services and a security mechanism plug-in development.

Java Card Applet as components orchestrator

Danijel Venus, Petar Bjeljac, Goran Čeko, Branko Perišić