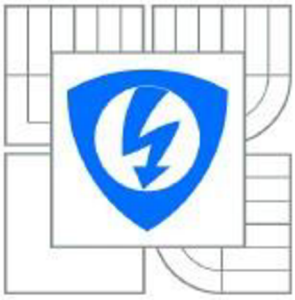
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MULTIAPLIKAČNÍ ČIPOVÉ KARTY

MULTIAPPLICATIONAL SMART CARDS

DIPLOMOVÁ PPRÁCE

MASTER´S THESIS

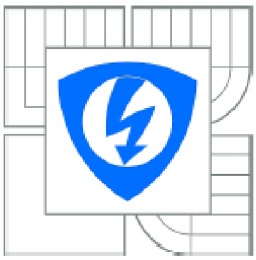
AUTOR PRÁCE Bc. IVO MELUZÍN

AUTHOR

VEDOUCÍ PRÁCE Ing. JIŘÍ SOBOTKA

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Ivo Meluzín	***ID:***	98152
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Multiaplikační čipové karty

POKYNY PRO VYPRACOVÁNÍ:

Zabývejte se výzkumem použití Smart Card jako autentizačního prvku pro více aplikací současně. Analyzujte podmínky multiaplikačního využití Smart Card. Zaměřte se především na problematiku umístění více aplikací na jedné kartě. Navrhněte systém, kde jedna Smart Card slouží pro autentizaci uživatele v jedné aplikaci a zároveň jako elektronická peněženka v jiné aplikaci, přičemž jedna aplikace by neměla mít přístup k datům druhé aplikace. Navržený systém se pokuste implementovat.

DOPORUČENÁ LITERATURA:

[1] RANKL, Wolfgang; EFFING, Wolfgang. Smart Card Handbook. 4th edition. Munich : John Wiley & Sons, Ltd., 2010. 1043 s. ISBN 978-0-470-74367-6
[2] RANKL, Wolfgang. Smart Card Applications. West Sussex (England) : John Wiley & Sons, Ltd., 2007. 217 s. ISBN 978-0-470-05882-4.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Jiří Sobotka

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Anotace

Cílem diplomové práce je nejprve popsat možnosti širokého využití čipových karet v odvětvích běžného života. Je nutné následně zmínit hardwarové a softwarové vybavení chytrých čipových karet, jejich komunikaci s terminálem a datové zabezpečení. Zaměříme se v dalším pokračování práce na prostředí Java Card, ve kterém se pokusíme vyzkoušet dvě aplikace, pro elektronickou peněženku a kontrolu identifikace uživatele s PIN kódem. Na základě tohoto je třeba zmínit multiaplikační pravidla sdílení dat a objektů kartou.
V závěru je nutné zaměřit se na možnosti vzájemného ovlivňování a také zabezpečení proti útoku z vnějšku i budoucí využití technologie.

Klíčová slova

Čipové karty, Smart Card, Java Card, autentizace uživatele, elektronická peněženka, applet, APDU, jazyk Java, zabezpečení Java Card.

Abstract

The goal of the first part of the thesis is to describe the options of wide usage of chip cards in different segments of our life. Consequently it is necessary to mention hardware and software equipment of smart card, its communication with terminal and data security. In this thesis we focus on Java Card environment in which we will try to create two applications, one for electronic purse and the second for user identification. Basically, we need to mention multiapplicational rules of sharing data and objects. At the end of the thesis we are focusing on the possibility of mutual interference between the applications and on protection against attacks. Also future applications of this technology are described.

Key words

Chip cards, Smart Card, Java Card, user authenticate, electronic purse, applet, APDU, Java language, security of Java Card.

MELUZÍN, I. *Multiaplikační čipové karty*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 83 s. Vedoucí diplomové práce Ing. Jiří Sobotka.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma "Multiaplikační čipové karty" jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 25. května 2011 .....

podpis autora

Děkuji svému vedoucímu práce Ing. Jiřímu Sobotkovi, za užitečné rady a pomoc při zpracování diplomové práce.

Obsah

Seznam obrázků, tabulek, kódů

ÚVOD

Lidé se snaží odjakživa si ve svém životě ulehčit cestu k nějakému cíli. Každý jedinec neustále získává informace a snaží se je nějakým způsobem uchovat co nejdéle. Z důvodu touhy po skladování informací a dat na jednotném médiu se už několik desetiletí pracuje na různých paměťových periferiích a na zvětšování jejich kapacity. Vědci a vývojáři vždy tíhli k miniaturizaci a integraci výrobků. Zvláště v elektrotechnickém průmyslu, včetně oblasti mikročipů. Vývoj paměťových čipů a též výpočetních mikroprocesorů se od svých prvních krůčků těšil velké oblibě. Po roce 1970 se v oblastech paměťových médií mluví o tzv. čipových kartách. První z nich se šířily z Německa a využívaly se jako telefonní karta. Ta obsahovala jednotky pro volání a po jejich vyčerpání pozbyla významu. Až později se začínaly objevovat karty s multiaplikačním využitím.

V průběhu dlouhého vývoje se karty začali osazovat programovatelnými čipy, které se mohly vybavit mnoha funkcemi a lze už tak mluvit o tzv. chytrých kartách (Smart cards). Od prvních paměťových karet ušla technologie Smart Card dlouhou cestu. Nabízí nyní uplatnění v celém spektru oblastí a služeb. Čipové karty se používají hlavně v bankovnictví, dopravě, oblasti kontroly přístupu, zdravotní péči a telekomunikacích. Velkými oblastmi využití jsou platební karty bankovních domů a použití karty jako autentizačního prostředku uživatele.

V diplomové práci jako celku se zaměříme na pochopení technologie Smart Card. S pomocí vývojového prostředí Java Card Technology se pokusíme navrhnout i aplikovat na čipovou kartu funkce pro identifikaci a elektronickou peněženku. Záleží nám na tom, dokázat schopnost oddělení těchto dvou aplikací v rámci karty. Případně se seznámit s možnostmi ovlivnění jedné aplikace druhou. Součástí práce je také popsat současná pravidla bezpečnosti na kartách Java Card platformy a jejich využití dnes i v budoucnu.

Strukturu práce můžeme popsat následovně. V první kapitole se blíže seznamujeme s historickým vývojem čipových karet, jejich použitím pro sdílení dat a aplikací v různých světových průmyslových oblastech. U druhé části se zmiňujeme o konstrukci a typech karet. V třetí kapitole se již zaměříme na technologii Java Card pro čipové karty, postavené na programovacím jazyce Java a jeho schopnostech. Pro následující část si přiblížíme základní stavební jednotku Java Card, tj. applet. Seznámíme se s appletem, jeho principem, komunikací i zabezpečením. Vyzkoušíme návrh v prostředí JCS Suite v3.0. V závěrečné fázi klademe důraz na seznámení se s možnými útoky na kartu, včetně obranných mechanizmů. Zhodnotíme vhodnost karty pro více aplikací současně. Také definujeme pravidla správného používání a bezpečnosti. Na konec shrneme směry dnešního multiplikačního využití karet s budoucími nároky na ně.

1 POUŽITÍ ČIPOVÝCH KARET

Obrovský rozvoj internetu a bezdrátových komunikací radikálně změnil způsob komunikace a propojování mezi lidmi. Jakmile svět začal být více elektronicky propojený, běžný model obchodování se přiblížil od komunikace tváří v tvář k transakcím prováděným on-line v reálném čase, jen díky pár kliknutím myši v našem domově.

Úspěch obchodování elektronicky závisí na důvěře společností, které sice vystavěly svůj obchod na osobním kontaktu, ale pár posledních let již kladou stejnou váhu i na neosobní elektronické transakce. Bezpečnost a mobilita čipových karet, dále jen „Smart Card“, zajistí spolehlivou a pohodlnou cestu k provádění e-obchodování a dalším požitkům moderní doby, ve spojení s oblastmi cestování, placení mýtného, kontroly fyzického přístupu a identifikaci.

Chytrá “Smart card“ [4] sdílí a zpracovává informace přes elektronické integrované obvody zapuštěné v plastovém substrátu těla karty. Díky čipu jde o jistý druh přenosného počítače. Tato technologie umožňuje nést informaci a také výpočetní sílu pro ni. Proto si vystačí během transakcí bez dalších větších databází.

1.1 Úvodní fakta

Myšlenka zapuštění integrovaných obvodů (IO) do plastové karty vyšla z hlav dvou německých vědců, Jurgena Dethloffa a Helmuta Grotruppa v roce 1968 [6]. Zajistili si na svůj vynález patent. Zcela nezávisle na nich o dva roky později, si pan Kunitaka Arimura z Japonska nechal patentovat myšlenku čipových karet. Pokroku dosáhl Roland Moreno se svým 47. patentem Smart Card technologie v době, kdy společnost CIII-Honeywell-Bull začala komerčně nabízet čipové karty a uvedla na trh karty s mikroprocesorovým čipem.

První zkušební čipové karty byly zavedeny ve Francii a Německu po roce 1980, které fungovaly jako předplacené telefonní karty a bankovní kreditní karty. Následný úspěch prokázal velký potenciál díky přizpůsobivosti.

Později prostřednictvím vývoje výkonných čipů a kryptografie se karty staly výkonnější. V dnešní době jsou mimo jiné používány jako elektronická peněženka, též jako přístup k lékařským záznamům člověka, k nabídce televizních a satelitních programů i jako základní identifikace pro mobilní komunikace nebo poslouží při řízení přístupu s identifikací.

Po několik let je tato technika běžně používána všude ve světě v různých odvětvích denního života. Samozřejmě se dále díky pokročilým metodám zabezpečení vyvíjí a proniká do dalších aspektů lidských potřeb.

Čipové karty (ČK) ve výsledku nabízí mnoho výhod. První přednost, výpočetní síla umístěná v malém prostoru. Přenositelnost, bezpečnost a jednoduchost v použití jsou další z nich.

Zjednodušeně – procesor, paměť, kanály pro vstup a výstup jsou zabaleny do jednoho celku v plastové kartě. ČK je imunní vůči většině známých útoků, protože nepotřebuje vnější potenciálně náchylné zdroje dat nebo napájení. K proniknutí do informací na kartě je nutné disponovat krom samotné karty i znalostmi o použití hardwaru, softwaru a dodatečných periferií. Kryptografické funkce posilují odolnost vůči vnějším útokům. Sdílená data jsou zakódována v rámci fyzické paměti, stejně tak jsou digitálně podepsána a v některých případech zašifrována při výměně s vnějším světem [4].

Další výhodou je přenositelnost. Svá cenná data můžete přenášet z místa na místo a kdekoli mít k ním okamžitý přístup. Stačí jednoduše vložit do čtecího zařízení a začít pracovat.

1.2 Využití téměř bez hranic

Karty Smart cards (SC) velmi často zajišťují uskladnění dat, přístup k nim, a také autentizaci při všemožných transakcích. Konkrétní příklady jsou uvedeny dále [1].

V bankovní sféře figurují rozšířené kreditní a úvěrové karty. Obsahují magnetický pruh na zadní straně, nicméně jsou vybaveny ještě výpočetními schopnostmi Smart Card pro bezhotovostní transakce a ověřování přístupu k účtu. Nové trendy placení v bankovním prostředí, jak je známe dnes, zahrnují ještě elektronickou peněženku.. Jde o menší platby, ke kterým není nutno vlastnit kreditní kartu s vysokým úvěrovým stropem.

V rámci věrnostních programů při nákupech, umožňuje karta zákazníkům maloobchodních společností odměnit za jejich častý nákup, což vede k větší spokojenosti na obou stranách. Zákazník může slevové body využít k nákupu zboží za nižší cenu, prominutí poštovného aj. Příkladem může být karta Clubcard obchodního řetězce Tesco. Obchodníci tak mohou i lépe porozumět požadavkům klientů.

U hromadné dopravy SC suplují jízdenky a dlouhodobé transportní poukázky. V listopadu roku 2010 nasadilo město Praha do oběhu právě speciální kartu Opencard pro různé využití služeb. Obecně v rámci autodopravy se nabízí možnost platit takto mýtné nebo parkovné i jízdné v hromadné dopravě, což se v podstatě podobá předplaceným telefonním kartám používaných už odnepaměti.

V oblasti zdravotní péče technologie Smart karet redukuje složitost správy záznamů lékařské karty pacienta a informuje o rozsahu zdravotního pojištění. Hlavní předností je mobilita a jednoduché sdílení informací o pacientovi mezi jednotlivými lékaři, nemocnicí a pojišťovnou.

Na internetu se velmi často setkáme s řízením přístupu a autentifikace díky SC. Pro veřejnost, využívající to k přenášení soukromého klíče, elektronického podpisu – tak nutnému k přístupu do dnešního elektronického světa, je to lákavá nabídka. Jak víme, soukromý klíč v kombinaci se známým veřejným klíčem slouží dobře v kryptografických

oborech. A samozřejmě v kombinaci s digitálním podpisem funguje jako spolehlivý autentizační systém.

Dále možné využít na webovém prostředí k přístupu na jednotlivé stránky, i schválení on-line transakcí.

V uzavřeném prostředí firem i univerzit , MULTIAPLIKAČNÍ čipové karty jednak poskytnou kontrolu fyzického přístupu do budovy a učeben, jednak přístup k síti, také možnost administrace dat uložených na kartě. Další výhodou je použití té stejné karty k placení a objednávání obědů a přístupu k dalším službám běžného života.
Technologie Smart Card má široké použití, tudíž si elektronické karty nachází a budou nacházet cestu do našich peněženek.

1.2.1 Moderní směry

Finanční
+ Elektronická peněženka zastupuje mince a drobné bankovky formou elektronických peněz pro nákup v prodejních automatech.
+ Kreditní, platební karty využívá i magnetického pruhu.
+ Platby přes internet.

Telekomunikace
+ Zabezpečené přihlašování a používání nejen GSM sítí (formou SIM karet v GSM telefonech).
+ Placení Pay-TV služeb internetových televizí.

Státní programy
+ Benefitní programy vlád, elektronické poukazy na stravování.

Zabezpečení přístupu k informacím
+ Přístupové karty zaměstnanců, studentů s použitím hesel a známých biometrických kontrol pro přístup k počítači nebo síti.

Kontrola fyzického přístupu
+ Většinou zaměstnanecké přístupové karty s ID identifikátorem osoby.
+ Řízení přístupu biometrikou – kontrola otisku prstů, tvar sítnice, kontrola obličeje.
+ Přístup do pokojů hotelů.

Dopravní
+ Řidičské průkazy, elektronické občanky.
+ Platba placených úseků.

Věrnostní programy

+ Obchodníci věrným zákazníkům nabízí slevy, bonusové nákupy, dárky.

Zdravotní péče
+ Zdravotní integrovaná karta pacienta s informací o pojištění a důležitá lékařská data.

Identifikace studentů na vysokých školách
+ Studentské karty (isic atd.) ID, umožňující více aplikací – studentský peněžní účet pro menzy, ubytování; přístupy do učeben, knihoven.

Dále se blíže zaměříme blíže na několik velkých oblastí pro využití čipových karet.

1.3 Elektronické platby a typy platebních karet

Systémy elektronických plateb i elektronické peněženky nabízí nejen již dávno známé služby. Pro banky snižují náklady spojené s manipulací s hotovostí. Peněženky pro off-line používání snižují cenu telekomunikační výměny dat. Pro obchodníky je urychlení transakcí tímto způsobem velice výhodné. Výdejní automaty a jízdenkové terminály nemusí být vyráběny tak draze a složitě, bez potřeby zpracování a uchovávání hotovosti. Částka zaplacená kartou může být pomocí komunikačních kanálů poslána jinam. Dochází k urychlení transakcí a ušetření času strávených běžným placením v obchodech a podobných místech. Technologie Smart Card vylepšuje běžné kreditní karty o několik schopností a o několik úrovní zvyšuje bezpečnost. Samozřejmě ale bezhotovostní transakce, alespoň v blízké budoucnosti, určitě zcela nenahradí klasické způsoby platby.

Lze hovořit o třech skupinách elektronických plateb – kreditní karty, debetní karty a elektronická peněženka [4].

1.3.1 Bloky platebních systémů

Elektronické platby založené na SC technologii můžou mít různé prvky odlišné. Neexistuje jednoznačný model charakterizující všechny systémy z důvodu různých požadavků v různých odvětvích. Zmíníme jen několik základních principů, rozdělených do čtyř částí.

Řídící systém
Dělí se na dvě stěžejní podskupiny. Bezhotovostní subsystém spravuje všechny účty jednotlivých bank a držitelů karet. Zabezpečuje i monitorovací funkce.

Řídící subsystém kontroluje administrativní postupy, vydávání nových verzí klíčů, dodávání nových verzí softwaru do terminálů. Zajišťuje bezpečnost všech operací v systému řízení.

Síť
Spojuje řídící systém se všemi terminály (přenosnými, jako jsou v obchodech u pokladny i nepřenosnými). Jde o transparentní síť, která nesmí měnit posílané zprávy mezi jednotlivými póly komunikace.

Terminály
Automatické terminály, známé jako bankomaty, slouží především k výběru hotovosti z účtu. Přenosné terminály jsou u pokladen obchodů a benzínových pump.

Čipové karty
Široké spektrum použití ji předurčuje nejen k platebním transakcím, ale i jako přístupové prostředky do budov, do sítě atd. Placení parkovného, placení průjezdu zpoplatněnou veřejnou komunikací. Pro větší bezpečí mají i samotné karty moduly pro šifrování a ochranu citlivých dat na čipu.

Karty jako dříve Quick v Rakousku, Geldkarte v Německu, tak dnes světově rozšířené VISA, MasterCARD a jim podobné z velké části splňují výše napsané a jde samozřejmě o chytré platební karty s čipem. Pro multiaplikační použití karet je běžné, že se využívá distribuovaných řídících systémů, kdy jich komunikuje více dohromady.

1.3.2 Elektronická peněženka

Nápady vytvořit elektronické platební prostředky použitím čipových karet vznikaly už od počátku technologie Smart Card. Po roce 1990 byl navrhnut první koncept systému plateb.

Pokud si představíme klasickou peněženku plnou mincí, bankovek a dokladů; můžeme si představit, jaké můžou být požadavky ze strany uživatelů na tu elektronickou. Je nutné mít možnost naplnit peněženku před použitím, platba musí proběhnout jednoduše. Všechny platby mají být pokud možno anonymní, přesněji řečeno, aby nemohlo být někým dohledáno kdo a kdy co koupil. V dnešní době samozřejmě víme, že pro vyšetřování policií a dalších věcí, je možné tyto věci dohledat.

1.3.3 Platební standard EMV a Eurocheque

V průběhu let se množily různé specifikace a normy pro využití čipových karet v širokém spektru oblastí. Tento standard charakterizuje dobré i stinné stránky kreditních

karet s mikročipem, jak jej známe dnes. Od roku 1993 tři mezinárodní kreditní společnosti Europay, MasterCard a Visa zahájili spolupráci na ustanovení pravidel "IC Card Specifications for Payment System". Následovalo několik verzí, ta v té době nejdůležitější – EMV 96 vydána po několika obměnách roku 1998. Po roce 2000 začala fungovat verze 4 pojmenována EMV 2000. Poslední a aktuální specifikací je EMV 4.2 od června 2008 [4].

Zmíněné ustanovení (skládající se ze čtyř knih) definuje jen minimum nejzákladnějších požadavků na vydavatele karet. S každou verzí se objevují nové otázky. Mnoho dalších pravidel a schopností je definováno samotnými výdejci karet dle potřeby. EMV a Eurocheque daly základ dnešním platebním systémům poskytovanými bankami.

Čipové karty v rámci jiného ustanovení Eurocheque jsou tzv. hybridní karty disponující čipem i magnetickým pruhem na zadní straně. Triple 3DES je použit pro šifrování. Byl definován jeden z prvních operačních systémů na kartě (OS). Všechny soubory nahrané zde mají uděleny práva pro zápis a čtení od výrobce. To funguje již ve více než 50 zemích Evropy, Asie, Afriky a Ameriky.

Po inicializaci spojení se systémem poskytovatele služby, je možné výměnu dat ochránit dle ISO 7816-4. Symetrický šifrovací algoritmus určený pro koncovou komunikaci se jmenuje 3DES ("Triple DES") [2]. Obě strany komunikace sdílejí privátní klíč bezpečně. Při přístupu k terminálu, je nutné ověřit identitu uživatele nějakou formou podpisu. Uživatel je také většinou vyzván k zadání 4-místného PIN kódu (Personal Identification Number).

1.4 Systémy identifikace přístupu osob

V tomto směru využití čipových karet tíhne obecně k ustanovení řízeného přístupu do nějakých místností v rámci domu, firmy, univerzity, případně přístupu k pracovním stanicím. U dveří i u pracovních stanic je umístěn terminál, který umožní lidem projít přes zabezpečené dveře nebo použít zabezpečený počítač díky identifikaci pomocí chytrých karet. Je důležité stanovit odlišné úrovně bezpečnosti, neboli přístup je limitován pro úzkou skupinu uživatelů. Přístup je povolen po úspěšném procesu autentizace uživatele. Existují různé procesy ověření identity, od vlastnictví správné karty, zadání hesla PIN, až k biometrickým kontrolám otisků prstu, sítnice apod. Terminál musí disponovat černou listinou neplatných karet a karet se zakázaným přístupem [7].

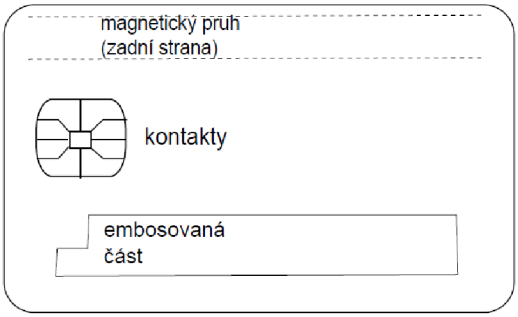
Největší snahou je minimalizovat dobu komunikace terminál – karta, dnes už během méně než sekundy se veškerá data bezdrátově vymění a rozhodne se o povolení přístupu. Zároveň je nutné zvážit zabezpečení terminálů a karet proti neoprávněným útokům a snahám zmanipulovat systém ke svému prospěchu.

V dnešním světě bývá technologie identifikace kartou použita hlavně v oblasti elektronického zabezpečovacího systému objektů.

2 TECHNOLOGIE SMART CARD

Tzv. “Chytré karty“ (smart cards - SC) nazýváme také jinak jako čipové karty, nebo karty s integrovanými obvody (ICC). Obvody jsou zabudovány do plastikové formy, obsahují prostředky pro výměnu dat, jejich skladování prostřednictvím čipu. Karty Smart Card mohou mít na zadní straně embossovanou část, dále i magnetický pruh. Obrázek 1.1 poskytuje podrobnější náhled na vzhled karty [1]. Jako Smart Card kartu označujeme tu, která už je schopna provádět složitější výpočty a neslouží jen jako paměťový prostor pro data. V tom jde o běžnou čipovou paměťovou kartu. SC rozměry vyhovují normě ISO 7816. To a ostatní fakta o kartách budou popsány na následujících stránkách.

Komunikace karty s ostatním světem, v podobě uživatelského terminálu, probíhá je-li karta v anebo blízko čtecímu zařízení, které je spojeno s PC.



Obr. 2.1: Čipová karta

Čipové karty je možné dělit do několika skupin [1]. Nejdříve na paměťové karty a mikroprocesorové karty. Smart cards mohou být i kontaktní a bezkontaktní; vztaženo ku přístupu k datům. Jako tzv. „chytrá karta“ je nazývána čipová karta s vestavěnými integrovanými obvody v rámci mikroprocesoru, schopného složitých výpočtů. To vše v rámci dostatečného paměťového místa pro větší nabídku aplikací.

2.1 Paměťové a mikroprocesorové karty

První SC vyráběny v masivním počtu byly ty paměťové. Nejde ještě o „chytrou kartu“ jako takovou, protože zde chybí programovatelná logika a je tu pouze paměťový čip. Slouží pro předplacené služby.

Nejdříve neměly karty žádnou výpočetní jednotku (CPU). Jejich možnosti zpracování dat byly omezeny na jednoduchý obvod, kde byly definovány jednoduché příkazy, které nemohly být přeprogramovány. Tudíž po vyčerpání předplaceného kreditu karta pozbyla funkčnosti.

I když se daly chránit data proti zásahu zvenčí bezpečnostní logikou, konkrétně telefonní karty mohly být jednoduše zneužity.

Výhodou těchto typů karet je ale nízká cena a nenáročná technologie.

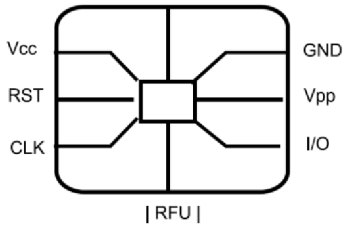
Oproti tomu, mikroprocesorové karty nabízí mnohem větší bezpečnost dat a multifunkčnost. Hlavně data nejsou nikdy přístupny zvenčí. Kontrola přístupu k datům řídí CPU, např. pomocí kódování (oblast kryptografie) a hesla pro identifikaci externích zařízení. Užitek skýtá možnost použít kartu pro jednu nebo více funkcí. Omezení určují paměťové zdroje a výpočetní výkon.

Tyto vlastnosti skýtají mnoho využití v praxi – přístupové systémy, bankovnictví, bezdrátové technologie. Cenová relace karet od počátků výroby klesla na několik desítek až stovek korun, v závislosti na nabízených službách.

Pojem Smart card odpovídá především druhému typu, naopak mezi čipové karty můžeme zařadit oba typy. Z důvodu komunikace čtecích terminálů a karet je nutnost přítomnosti procesoru zabudovaném na kartě, proto se dále zmiňuji jen o těch mikroprocesorových.

2.2 Kontaktní a bezkontaktní karty

Kontaktní karty se musí vložit do čtecího zařízení a pro komunikaci používají rozhraní s 6 kontakty (dle obr. 2.1) [1].



Obr. 2.2: Kontakty čipu čipové karty

Protože vkládání karty do čtečky ve správném směru je zdlouhavé, někdy se používají karty nekontaktní při rychlých operacích, jako je veřejná doprava a přístup do budovy.

Jak je všeobecně známo, tyto prostředky nepotřebují fyzický kontakt se zařízením neb je použita zabudovaná anténa. Napájení je zprostředkováno díky interní baterii či anténě. Vysílání probíhá přes elektromagnetické vlny. Výhody tedy jsou, že nedochází k opotřebení kontaktů a nemusí odpovídat standardní velikosti karty. V průběhu výměny dat musí být SC v dostatečné blízkosti s komunikujícím zařízením, aby při případném vzdálení se od něj, nedocházelo ke ztrátám dat. Samozřejmě může dojít i k nevědomému zachycení citlivých dat. A jednou z větších nevýhod je i cena. V rámci technologie Java Card (viz kapitola 3) je obvykle použita karta kontaktní.

2.3 Hardware Smart Card

Hlavními prvky jsou kontaktní body vsazené na povrchu plastového těla, centrální jednotka (CPU) a ještě paměťový blok, případně méně se vyskytující procesor matematických operací.

2.3.1 Kontaktní body

Dle obrázku 2.2 (čipová jednotka) je dáno 8 kontaktních pólů [1] o různých funkcích (dle ISO 7816).

- Vcc bod slouží jako zdroj energie čipu. Napětí se pohybuje okolo 3 V (SIM karta) nebo 5 V ±10 %.
- Bod RST se využije pro poslání signálu pro tzv. "warm reset". Naopak tzv. "cold reset" probíhá při odpojení od napájení nebo vytažení karty ze čtečky.
- CLK pól přivádí externí hodinový signál, dle kterého se vnitřní hodiny řídí.
- GND pro zem s určenou hodnotou 0 V.
- Vpp bod figuruje jen ve starších kartách. Přivádí dvě hladiny programovacího napětí. Změny napětí jsou vyžadovány při zápisu do paměti na dřívějších modelech.
- Input/Output (I/O) bod má užitek při výměně dat a příkazů v half-duplex módu. Tzn. data mohou být posílány jen v jednom směru v každém čase.
- RFU pro budoucí využití (bezkontaktní) je rezervováno.

2.3.2 Centrální procesor (CPU)

Nejběžnějším typem na SC je 8bitový mikrokontrolér (Motorola, Intel) pracující nad frekvenci 5 MHz. Vysoce účinné karty obsahují ještě časový násobič dovolující pracovat na 40 Mhz a více.

Dnešní čipy mají 16-bitový či 32-bitový mikrokontrolér, případně mohou mít menší sadu příkazů. Skutečně se používají ve velkém.

2.3.3 SC ko-procesor

S důrazem na bezpečnost přenosu a přístupu k datům na kartě, pomáhá s komplexnějšími výpočty týkající se šifrování. Výpočty různých aritmetických posloupností ve velkém číselném rozsahu jsou požadovány např. pro aplikaci RSA algoritmu (Rivest Shamir Adleman – veřejný klíč) [2]. Obsažením koprocesoru se zvedá cena.

2.3.4 Paměťové registry

Nalezneme zde tři typy pamětí; trvalá neproměnlivá (ROM), trvalá proměnlivá (EEPROM) a dočasná proměnlivá (RAM) a platí pro ně [7]:

- ROM (Read-only-memory) je používána pro sdílení dat továrního nastavení uloženém při výrobě. Po uvedení na trh není možné tuto paměť měnit. Obsahuje operační systém a funkční aplikace. Při výrobním procesu dochází při binárním zápisu k tzv. maskování dat.
- EEPROM (electrical erasable programmable read-only memory), stejně jako předchozí, dokáže uchovat data při odpojení od zdroje. Rozdíl je, že obsah může být měněn během užívání karty. V podstatě se jedná o jakýsi hard-disk. Hlavními parametry je rychlost přístupu, doba udržení dat (cca.10 let) a množství zápisů do konce životnosti (100,000). Rychlost čtení je stejná jako u RAM, ale zápis je o mnoho pomalejší.
- RAM (Random-acces-memory) – se používá jako dočasné úložiště pro data, se kterými se aktuálně pracuje. V případě odpojení zdroje se data ztrácí. Výhodné je, že disponuje neomezeným počtem přístupů.

Paměťové místo v RAM je největší. Naopak ROM je nejmenší a tím nejlevnější.
Dalšími rozvíjejícími se typy pamětí jsou tzv. flash paměti podobající se EEPROM. Slouží ke sdílení velkého bloku dat, ke kterým se přistupuje jako k celku.

2.4 Komunikační prostředky

2.4.1 Čtecí zařízení a uživatelské aplikace

CAD (card acceptance device) připojené přes některé rozhraní k počítači, slouží k vložení karty Smart Card (dále také SC). Existují dva typy přístupových zařízení – čtečky a terminály.

Čtecí zařízení jsou připojeny většinou přes USB port, případně sériový a paralelní. Díky slotu pro kartu, je možné vložit kartu, v případě bezkontaktním, probíhá komunikace elektromagnetickými vlnami. Čtečka zprostředkuje komunikační kanál pro výměnu dat mezi PC a kartou. Tyto zařízení sami o sobě nedokážou pracovat s daty, ale disponují schopností opravy dat a kontrolou přístupu pomocí transportních protokolů.

Terminály jsou už vlastně takové poloviční počítače. Disponují čtecím rozhraním, i procesní jednotkou pro práci s daty. Využití dnes nachází na benzinových stanicích a na pokladnách obchodů při placení při komunikaci s bankovními účty především.

Na těchto dvou zařízeních pracují programy, ovládající tok dat mezi koncovými zařízeními, říkáme jim uživatelské aplikace.

2.4.2 Komunikační model SC

Kanál mezi kartou a uživatelským PC je polo-duplexní; tedy buď posílá data jeden nebo druhý. Když komunikují dva počítače, posílají si datové pakety tvořené na základě transportního protokolu (TCP/IP viz [4]). Podobně čipové karty vysílají vlastní datové pakety – APDU (application protocol data units). Může být tvaru příkaz nebo odpověď.

V praxi zde funguje model master-slave (jeden aktivní, druhý pasivní). Čtecí zařízení (slave) s kartou čeká na příchozí příkaz ve formě APDU od uživatelské stanice (master). Odpovědí je poslán paket s informací, že zadaný příkaz byl vykonán atd.

2.4.3 APDU protokol

Dle normy (viz kapitola o standardech) ISO 7816-4 se tento protokol týká aplikační vrstvy. Zprávy APDU [4],[1] zahrnují dvě struktury: první používána uživatelem na straně čtecího zařízení k posílání příkazů a druhá používána kartou k odeslání odpovědí. Každému příkazu musí vždy následovat odpověď.

Tab. 2.1 : Příkazový paket (C-APDU)

Hlavička				***Tělo***		
CLA	INS	P1	P2	Lc	Datové pole	Le

Tab. 2.2 : Paket odezvy (R-APDU)

Tělo	***Povinné návěstí***	
Datové pole	SW1	SW2

Hlavička příkazového C-APDU se skládá ze 4 bajtů. CLA (class of instruction), INS (instruction mode), P1 a P2 (parametr 1 a 2).
CLA – identifikuje charakteristickou skupinu příkazu a odpovědi.
INS – zde je obsažen popis instrukce.
P1,P2 – obsahují upřesnění instrukcí.

Po hlavičce následuje v paketu hlavní tělo proměnné délky.
Lc – určuje délku datového pole v bytech.
Datové pole – soubor dat posílána do karty nutný pro provedení instrukce z hlavičky.
Le – očekávaný počet bajtů na straně uživatele v rámci odpovědi.

Paket (R-APDU) pro odpověď má dvě části. Tělo a povinné návěstí.
Datové pole – je určeno počtem bytů očekávané v Le části příkazového paketu.

Návěstí s parametry SW1 SW2 – neboli tzv. stavové slovo určuje do jakého stavu má přejít karta po provedení příkazu APDU např. (0x9000 pro stav: dokončeno). V obou typech paketů je datové pole nepovinné.

2.4.4 TPDU protokol

Transportní protokol pro výměnu paketů použitelný pro kartu i uživatelské prostředí (transmission protocol data units - TPDU) je definován v normě ISO 7816-3 (viz.kap. 2.6.1). Obsahuje dva pod-protokoly T=0 a T=1. V prvním orientovaném je jako nejmenší možná vysílatelná jednotka bajt. U druhého objektově orientovaného je to slovo, sekvence bajtů [4].

2.4.5 ATR

Po připojení karty do čtecího zařízení vysílá technologie Smart Card na ní obsažená, zprávu ATR (Answer to reset) [4] do uživatelského PC. Jde o jakousi inicializaci, při které se posílají parametry pro vytvoření komunikačního kanálu. Informuje uživatele o použitém transportním protokolu, rychlosti výměny dat, informace o čipu a všechny ostatní potřebné informace. Dodatečně se zaměříme na využití k identifikaci appletu v kapitole 4.

2.5 Operační systém karet

Smart Card čipové karty disponují operačními systémy podporující sbírku instrukcí pro vytváření a řízení aplikací, podobné počítačovým OS. Tyto systémy dokáží zpracovat hlavně APDU příkazy i další zadané od výrobců.

V tomto případě, uživatelská aplikace je vlastně složka dat sdílející řídící informace pro právě řízenou aplikaci. Instrukce pro přístup k těmto datům implementuje operační systém. Ukazuje se, že jde o úzkou spolupráci těchto protipólů.

Dnes tyto centralizované operační systémy jsou zastoupeny na všech dnešních čipových kartách. Novější OS podporují lepší oddělení jednotlivých vrstev dat a lepší schopnosti nahrávání zdrojových kódů.

2.6 Řízení, normy a specifikace

Větší část softwarového vybavení a systémových periferií funguje na uživatelské straně. Systémové vybavení rozpozná kartu SC a inicializuje komunikaci mezi čtečkou a PC. Ruku v ruce jdou s tímto vybavením i prostředky pro správu a zabezpečení karty. Běžný uživatel svými prostředky disponuje sadou ADPU (viz. kap. 2.4.3) příkazů. Programové vybavení na uživatelské straně je napsáno jedním z široce používaných jazyků C nebo Java.

Software běžící na samotné kartě také zahrnuje některé systémové a uživatelem definované prostředky. Řídí se odtud vstupně výstupní komunikace, systém souborů a vlastnosti potřebné pro běh programů. Software karty může být definován v jazyce assembleru v čipovém mikroprocesoru nebo pomocí programovacího jazyka.

K zajištění součinnosti softwaru od různých výrobců, vývojové skupiny Open Platform nebo OpenCard Framework (viz níže) nabízí určité sjednocené prostředí pro uživatelskou veřejnost.

Java CARD prostředí umožní napsat aplikaci v jazyce Java a zprovoznit ji v rámci nějakých pravidel na kartě podporující Smart Card technologii.

2.6.1 Standardy

V průběhu let bylo ustanoveno několik pravidel a norem, které zmíníme níže.

ISO 7816

Pojmenována: Idenifikační karty – Čipové karty s integrovanými okruhy. Definuje dle společnosti International Organization for Standardization (ISO) charakteristiky čipových karet s elektrickými kontakty, a můžeme ji dělit na několik podskupin [6].

ISO 7816-1: fyzické parametry
ISO 7816-2: umístění elektronických kontaktů čipu
ISO 7816-3: protokoly výměny dat a elektronických signálů
ISO 7816-4: vnitřní příkazy
ISO 7816-5: identifikace aplikací
ISO 7816-6: interní datové složky
ISO 7816-7: příkazy pro programování

EMV standard (Europay, MasterCard, Visa)– zakládá se na ISO normách, ale upřesňuje požadavky na využití v bankovním průmyslu.

Open Card Framework

Je spravováno společenstvím OpenCard seskupující hlavní průmyslové giganty v Smart Card. Jde o prostředí na uživatelské straně, které komunikuje s čtečkami karet i se samotnými kartami. Snaží se minimalizovat závislost na výrobcích a prodejcích čipových zařízení. Tato aplikace umožňuje pro PC stanice využívat síťové prostředky disponující prostředím Java. Jednoznačně ulehčuje práci s programováním karet pro běžného uživatele znalého základu programovacích jazyků.

3 PROSTŘEDÍ JAVA CARD

Nyní si popíšeme technologii využitou v práci. Java Card technologie (dále také JC) umožňuje nahrát a spustit na čipové kartě programy pro různé typy aplikace, i zdrojové kódy psané ve vyšším programovacím jazyku Java. Je třeba se tedy seznámit s jednotlivými částmi a architekturou technologie Java Card firmy Oracle [3] (dříve Sun Microsystems).

3.1 Vývoj aplikací pro chytré karty

V průběhu let se vždy ukázalo, že přijít s novým a novým uplatněním technologie je běh na dlouhou trať. Je tomu tak, přestože karty jsou dle různých norem standardizovány ve většině svých parametrů i vlastností. Velikost, tvar a komunikační protokoly jsou více méně stejné, vnitřní rozhranní se liší výrobce od výrobce. Vývojářské nástroje jsou vytvořeny samotnými výrobci Smart Card (SC). Základním stavebním kamenem je ucelený programovací jazyk obsažen v každém vývojovém balíčku (hlavně Java Card Technology) a také simulátory hardwaru podporovány výrobci a prodejci karet a karetních čteček. Není ani možné, aby třetí strana vyvíjela aplikace nezávisle na těchto dvou opěrných pilířích. Nicméně je zvykem, že vytváření aplikací je výsadou úzké skupiny kvalifikovaných vývojářů se znalostí prostředí určitého softwaru.

Jelikož nejsou přesně definovány rozhranní pro aplikace na vyšších vrstvách, vývojáři jsou nuceni pracovat s komunikačními protokoly a pamětí nižších vrstev. Směrům využití SC karet v dnešní době předcházel dlouhý výzkum od základů. V praxi trvá zavedení nové techniky do prodeje 2 roky. A převoditelnost na jiné platformy je téměř nemožná.

Tak vývoj určený přímo pro příslušnou platformu zapříčiňuje to, že jakékoli technologie od jiných poskytovatelů nemůžou fungovat na jedné kartě společně. Absence větší spolupráce mírně brzdí vývoj technologie SC.

3.1.1 Jazyk Java pro Smart Card

Java CARD Technology (JCT) [1] je prostředí nabízející mnoho řešení při vyvíjení nových funkcí pro čipové karty Smart Card. Umožní spuštění hlavní instance (pojmenována „applet“) napsané v jazyce Java. JCT představuje platformu využívající programovací jazyk ruku v ruce s bezpečným a přenosným prostředím multiaplikačních čipových karet. Má několik charakteristických bodů.

Pohodlný vývoj – Může těžit z dalších zásuvných modulů od mnohých výrobců, implantovaných v prostředí Java. JCT nabízí open-source platformu zahrnující API (application programming interface – uživatelsky přehledné programovací rozhranní). Applety mohou čerpat z knihoven vytvořených někým jiným.

Bezpečnost –Ve vývojovém prostředí se zabudované bezpečnostní funkce doplňují s technologií SC karet. Jednotlivé applety při multifunkčním použití karty, jsou odděleny firewallem (bezpečné oddělení každé aplikace vysvětleno později). Tím se systém dokáže bránit pokusům poškození jednotlivých částí útočníkem.

Hardware nezávislost – JCT je nezávislá na použitém zařízení.

Sdílení a řízení – Obrovskou výhodou pro použití hlavně v dnešní době integrace, je možnost aplikovat více funkcí na jednu kartu. Je to umožněno díky zmiňovaným oddělením prostředků firewallem, který znemožňuje ovlivňovat jeden skript appletu jiným.

Existují mnohé mezinárodní standardy (ISO 7816) a hlavně instance appletu můžou komunikovat nejen na bázi JCT zařízení, ale i s jinými běžně používanými čtecími terminály.

Základy prvního vývojového prostředí Java Card API (application programming interface) byly uvedeny v roce 1996 inženýry z vývojového centra Schlumberger v Texasu. Ve snaze najít zabezpečené prostředí pro vývoj, se ukázalo jako nejlepší integrace do jazyku Java. Později vzniklo společenství, které založilo první Java Card forum, za účelem vyvíjení přínosných aplikací pro budoucnost.

Od roku 2001 se na technologii pracuje a vyvíjejí se nové verze Java Card. Prostředí je licencováno desítkami výrobců čipových karet. Poslední verzí je Java Card v2.2.2 či nejnověji upravovaná v3.0 [3].

3.2 Obecná architektura

SC je docela malá výpočetní jednotka používaná už léta. Největší vývojářská výzva JCT (Java Card Technology) je zprovoznit tento software na čipové kartě ruku v ruce se zachováním paměťového prostoru pro všechny možné oblasti použití. Proto čipové karty podporují jen omezené prostředky jazyka Java a jako podporu je třeba implementovat virtuální stroj [1] neboli Java Card Virtual Machine (JCVM).

Díky ní je možné virtuálně, chcete-li nanečisto, zkoušet a testovat aplikace za absence reálné čipové karty. Dělíme jí na dvě oblasti – jedna část běží na kartě, druhá část funguje a běží bez karty. Mnohé výpočetní úlohy nejsou nuceny spouštět se při samotné exekuci programu a aplikace. Jsou to především: kontrola syntaxe, načítání tříd, linkování, optimalizace. Všechno je provedeno mimo prostředí karty na počítači s JCVM.

Výpočetní prostředí na kartách SC vzdáleně připomíná počítač. Pro lepší podporu Java, je v rámci technologie Java Card vymezeno vývojové prostředí podporující práci s pamětí, komunikaci, bezpečnost a umožnění spuštění modelu programu (ISO 7816). Toto

prostředí dokáže obalit stěžejní jádro systému karty a oddělit ji od přídavných aplikací určených pro jiné použití..

Víceméně JCT definuje platformu, díky níž vytvořené programy v Javě mohou fungovat na čipových kartách - v následujícím textu nazýváme programy jako – **Applety.**

Architekturu celého systému pro vývoj aplikací s technologií Smart Card dělíme na tři bloky [1]:

- Java Card Virtual Machine (JCVM) – definuje omezená pravidla pro použití jazyku Java a pro použití k práci s kartami.

- Java Card Runtime Environment (JCRE) – určuje chování při kompilaci a spuštění programu Java – popisuje správu paměťové oblasti, zacházení s appletem atd.

- Java Card Application Programming Interface (API) – jde o jádro a soubor rozšířených balíčků nebo tříd určených pro použití v technologii Smart Card.

3.2.1 Java Card Virtual Machine

Hlavní rozdíl od klasického Java virtuálního prostředí tkví v dělení se na dvě části. Pomocí karty spouštěný překladač (interpreter) a převodník (converter) běžící v prostředí uživatelského PC. Společně umožní načíst Java třídy a provést výpočet programu na základě daných pravidel.

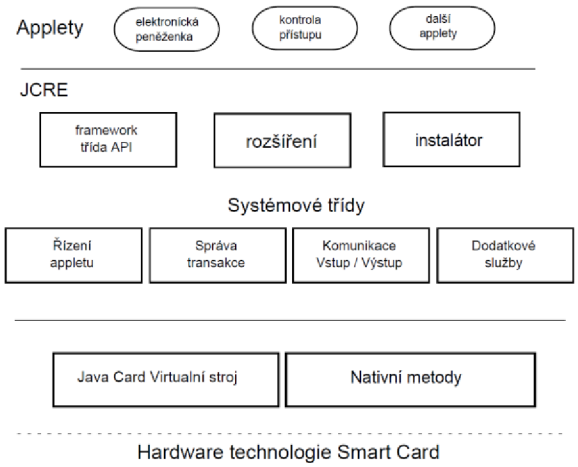
Převodník načte a připraví data všech tříd a vytvoří z nich balíček CAP (converted applet) doprovázený skupinou dat pro API (tzv. složka Export). Převedený applet (viz dále) je pak nahrán na kartu a spuštěn překladačem. Veškeré prostředky Java nepodporované Java Card jsou detekovány už převodníkem.

CAP soubor obsahuje spustitelnou binární reprezentaci použitých tříd z balíčků Java Card. Jde o vlastně speciální reprezentaci spustitelného archivu v Javě s příponou ".jar". Spolu se zdrojovým kódem jsou zde uchovány linkovací vazby, informace o třídách. V Java prostředí je třída centrálním bodem architektury. Určuje jakýsi standard pro kompatibilitu platforem Java. Soubor je konceptem toho, v jaké podobě je software nahrán na kartu. V rámci prostředí karty se tím umožní dynamické načítání potřebných tříd appletu. Největší výhodou je tedy pravidlo, kdy po správné kompilaci zdrojového kódu appletu je možné jej spustit na kterékoliv kartě podporující tuto technologii. Popíšeme si CAP přehledněji v jedné z dalších kapitol o návrhu aplikací.

Průvodní soubor Export vytvořený při kompilaci obsahuje informace o jejím účelu. Obsahuje veřejné informace o API pro balíček tříd a podklady pro přenositelnost.

3.2.2 Java Card Runtime Environment

Neboli prostředí pro exekuci Java Card Appletů (JCRE) již v prostředí karty (také viz 3.1). Jsou ji předány veškeré data z počítače či terminálu, na kterém běží virtuální stroj (VM). Přebírá dohled nad síťovou komunikací, exekuci appletu, jádra systému a zabezpečení na kartě. Můžeme ho považovat při absenci jiných systémů, za operační systém čipové karty.



Obr. 3.1: Objektové schéma vývojového prostředí JCRE [2]

Dle obrázku (3.1) zahrnuje pod sebou výše zmíněné prostředí Virtuálního stroje (Java Card Virtual Machine) a konstrukční základ tříd (vlastně API zmíněné níže). Díky tomuto, je applet (Aplikace spustitelná na kartách psaná v Javě) snazší vytvořit a spustit na odlišných technologiích různých výrobců.

Blok vestavěných metod (nativní) pracuje s transportními protokoly, kryptografickou kontrolou a správou paměti.

Oblast Systémových tříd se stará o správu transakcí a o běh programů, a také o výměnu dat mezi kartou a koncovým uživatelským zařízením.

Aplikační soustava dává meze využití API tříd a balíčků. Applety mají přístup k JCRE právě přes API, dostatečně to ulehčuje vývoj programů v tomto prostředí.

Instalátor zajistí nahráti appletu do karty, v praxi po její produkci z výroby. Pracuje se soubory CAP (kap 3.3.1). Jde o volitelný nástroj, ovšem bez něj není možné nahrávání dodatečných appletů, neboť vše je tom případě nahráno již při výrobním procesu.

Vývojové prostředí Java Card je inicializováno jen jednou v době procesu – provede se při ní spuštění JCVM a vytvoření objektů pro jednotlivé služby. Instance appletů vytvoří v objektech místo pro sdílení dat. V případě náhlého i úmyslného restartu se rozběhne JCVM od posledního začátku smyčky. Na rozdíl od toho stav prostředí a objekty jsou v kartě uchovány v trvalé paměti EEPROM.

3.2.3 Rozhraní API

Zjednodušeně řečeno – programovací rozhraní pro tvorbu zdrojových kódů využitím balíčků a knihoven technologie Smart Card. Skládá se z definovaných tříd pro různé funkce za účelem využití na kartě. Obsaženy jsou zde tři hlavní balíčky a knihovny.

V `java.lang` třída Object definuje kořenovou oblast hierarchie Java Card, přídavná třída `Throwable` je základem pro tvorbu výjimek a detekci chyb.

`Javacard.framework` určuje třídy pro běh jádra appletu. Obsahuje konkrétně třídu Applet pro definici jeho spuštění. Uvedeme ještě třídu PIN, která určuje pravidla nejčastěji používané autorizace pro kartu.

Balíček `javacard.security` obsahuje pravidla pro kryptografické šifry např. DES symetrický nebo RSA asymetrický šifrovací algoritmus (viz.kapitola 4.2).

4 VYTVÁŘENÍ APPLETŮ JAVA CARD

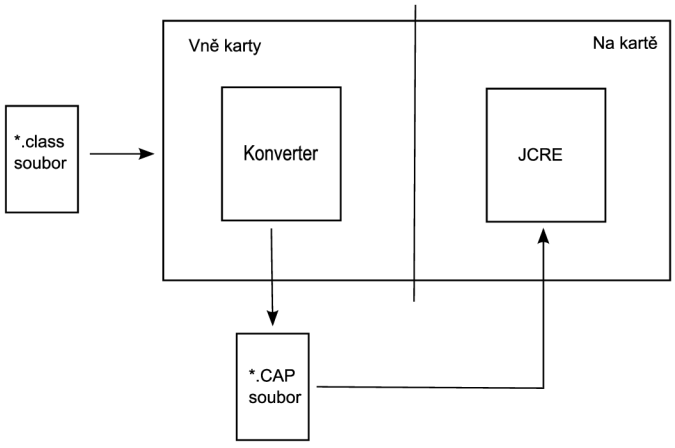
4.1 Návrh appletů pro multiaplikační kartu

Již zmiňované applety jsou základní stavební jednotky, přesněji tedy aplikace, prostředí Smart Card. Budeme mluvit ve zkratce o způsobu práce s applety v prostředí JCRE a dále se zaměříme na pravidla a způsob jejich zápisu. To vše za účelem nastínění tvorby a úpravy aplikace elektronické peněženky a aplikace pro kontrolu přístupu pomocí identifikace např. PINem.

Applet je program napsaný v programovacím jazyce Java [3] podléhající pravidlům Java Card. Skládá se ze sledu příkazů a pravidel pro spuštění ve výpočetním prostředí JCRE bloku. V jádru reprezentuje applet instanci několika tříd v balíčku `javacard.framework.Applet`. Ty jsou nahrány na kartě po dobu jejího života, i když mnohé speciální karty podporují volitelné vymazání aplikací. Systém tvorby podporuje multiaplikační přístup a každý applet má unikátní identifikační číslo AID (viz výše).

Principy tvorby v Java Card

Klasické Java technologické nástroje se používají k vytvoření a kompilaci zdrojového kódu appletů platformy Java Card. Po tomto procesu vzniká z kódu soubor s příponou *.class, předzpracovaná verze appletu. Dále pomocí převaděče (JC converter) vytvoříme soubor schopný nahrání na kartu (CAP). Zde už je díky CAP možné stáhnout aplikaci na kartu.



Obr. 4.1 : Konverze třídy na CAP [1]

4.1.1 CAP soubor

Jde o mezikrok mezi tvorbou kódu a nahráváním hotového appletu na kartu. Obecně jsme si princip popsali v kapitole o JCVM a nyní přidáme několik informací. Načtením souborů tříd Java kódu v balíčku vytvořeného projektu se dále provede převod na konvertovaný

applet tedy CAP soubor. Konvertor těchto souborů je součástí virtuálního stroje Java Card Virtual Machine. Po nahrávání zmíněného speciálního souboru na kartu, se na základě jeho binárního kódu vytvoří v paměti karty nový applet.

Soubor tzv. Converted Applet (CAP) [1] obsahuje formu binární reprezentace tříd z vytvořeného balíčku. Součástí CAP je několik boků komponent. Ty se zabývají např. popisem informací o třídách, reprezentací bajtového kódu, kompilací a ověřování napsaného kódu. Tento typ souboru stanovuje standard pro přenositelnost na ostatní platformy. Podle tohoto principu je software nahráván na kartu. Dodatkovým souborem při konverzi kódu na CAP je tzv. export file. Funguje obdobně, jako hlavička v souborech psaných v jazyku C. Obsahuje informace o API a spojovacích mechanizmech. Java Card prostředí se od běžné Java aplikace v mnoha částech liší (viz kap. 3).

4.1.2 Obecná metoda návrhu

Samotný proces tvorby appletu začíná u návrhu balíčku, stejně jako v jazyce Java. Po uvedení některých hlavních částí kódu (konkrétně peněženky) si popíšeme další postup.

Výpis kódu 4.1 : Část kódu appletu peněženky

```
Package src.purse ;
Import javacard.framework.* ;

Public class Wallet extends Applet {
Final static byte Debit = (byte) 0x20 ;    //definice možných
           //instrukcí, např. zde konkrétně čerpání peněz)
…
Byte balance ;
OwnerPIN pin;
Byte buffer [ ] ;              // proměnné / APDU zásobník)
Private Wallet () {
…
Pin = new OwnerPIN ( PinTryLimit, MaxPinSize)  ;   //nastavení PINu)
Register () ;     }    //alokování pamětí v konstruktoru)
…
Public static void install (APDU apdu) {

New Wallet () ;   }          //metoda install pro instalaci appletu)
Public boolean select ()  {

Pin.reset() ;               //metoda select pro volbu appletu)
…. }
```

Výše uvedené skripty, jsou výňatky z kódu. Hlavním blokem je třída `Wallet`, v níž jsou definovány proměnné a určené výrazy pro instrukce. Metody `install` a `select` zajistí výběr appletu po nahrátí na kartu.

Výpis kódu 4.2: Další části peněžního appletu

```
…
Public void process (APDU apdu) {
Buffer = apdu.getBuffer (); //(metoda pro čtení ADPU paketů v JCRE )

If (buffer [ISO.OFFSET_CLA]  != = Wallet_CLA) ISOException.throwIT
(ISO.SW_CLA_NOT _SUPPORTED)    //ošetření chyb instrukce

Switch (buffer[ISO.OFFSET_INS]) {
…
Case Debit: debit(apdu) ; return ; }       //příklad zpracování APDU
}
…                              //nyní se definují jednotlivé metody
Private void debit (APDU apdu)  {
…
Balance = (byte) (balance - buffer [ISO.OFFSET_CDATA]) ;
…   }
Private void validate (APDU apdu) {        //k zabránění
                          //nepovolenému přístupu slouží PIN

Pin.check(buffer, ISO.OFFSET_CDATA, byteRead) ;
}
```

K předchozím metodám je třeba ještě uvést metodu `process`. Ty společně tvoří základní páteřní kameny [17] každé aplikace v Java Card. Dodatečná definice metody pro čerpání peněz `debit`, ověření PIN kódu pro autentizaci, případně další (zde nevypsané) metody, jsou definovány dle potřeb konkrétních appletů.

Instalace appletu

Instalování appletu může proběhnout v rámci výrobního procesu karty, také u výdejce, ale i pomocí bezpečné instalace u uživatele. Konkrétně jde o stažení digitálně podepsaného appletu a ověření jeho správnosti JCRE v prvním případě. Při otevřeném tzv. OPEN stavu, instalace na kartu probíhá, u našeho případu, v rámci programu JCT Suite pomocí bezpečností domény v kartě (Issuer Security domain).

Třídy v programu na platformě JC nehovoří přímo se čtecím zařízením nebo terminálem. To se děje pomocí řídícího prostředí. Třídy appletů tak moho vyměňovat data s JCRE nebo ostatními applety v rámci daných mezí. Samotná JCRE zvolí applet a pak mu posílá přijaté APDU příkazy. Tím řídící prostředí take odděluje uživatele od protokolové komunikace a čipu karty.

JC Applet může být nedosažitelný, v případě vymazání odkazu v řídícím registru. I když v paměti stale figuruje. Instalací aplikace se aktivují její statické články. Inicializují se proměnné.

Třída appletu

Každý applet v Java Card dědí třídu `Applet` z knihovny. Tato abstraktní třída má několik metod, které musí být definovány každým nově vytvořeným appletem. Hlavní z nich si probereme. Při instalaci samotné je volána metoda `install()` pomocí JCRE jen jednou. Je nutné definovat ji jako statickou, z důvodu volání ještě před aktivací appletu. Konstruktor není volán přímo.

Jde ale o první krok, který metoda udělá – zavolá svůj konstruktor. V tomto bodu se inicializují nestatické proměnné všech tříd deklarovaných v appletu. Stejné to je i pro statické metody. Tudíž statické metody jsou alokovány ještě před spouštěcím procesem, a to při nahrávání celé aplikace. Od chvíle, kdy applet sám má za úkol sestavit data a své metody, musí být konstruktor ve stavu `public`, `private`, `protected`. V případě „public“ může být konstruktor volán jiným appletem. Metody a data nejsou viditelné bez ohledu na jejich práva.

Objekt APDU postoupený do metody `install()` dokáže předat jednorázovou inicializaci či identifikační data. V našem případě peněženková aplikace dědí metody z třídy Applet v závislosti na obecné implementaci. Třída tak zahrnuje prvky JCRE – hlavní metody jsou:

Výpis kódu 4.3 : Metody definované pro JCRE

```
Public abstract class Applet {

Protected Applet () { }
...
public static void install (APDU apdu) throws ISOException {
ISOException.throwIt(ISO.SW_FUNC_NOT_SUPPORTED);
}
```

```
public void process (APDU apdu) throws ISOException {
byte buffer[] = apdu.getBuffer();
...
ISOException.throwIt(Util.makeShort((byte)0x9F, buffer));
}

protected boolean select() {
...
return true;
}

public void deselect () {}
...
protected final void register () {
AppTable.register (this);
...
}
}
```

V uvedené ukázce jsou vytipovány některé důležité části kódu, které minumálně musí dědící třída `Applet` obsahovat. Ta dále volá metodu `process()` jako způsob přijímání dat z čtecího zařízení. Konkrétní informace je zabalena v parametru APDU paketu pro metodu. Výše zmíněná implementace je všeobecně platná. V tomto případě je ještě zde volána chyba pro případ příchozí instrukce, kterou systém nepodporuje.

Metoda `install()` v případě chybového hlášení ukazuje, že applet nemůže být nainstalován. Pokud se tak stane, dokud nebude syntax upravena, nebude applet nahrátý na kartu. Všechny úvodní jednorázové inicializace (registrace, vytváření objektů pomocí `new`) by měly být provedeny právě v této třídě.

Registrace appletu

Např. v appletu peněženky i autentizace je APDU přeposláno do konstruktoru, i když to není striktně vyžadováno. Konstruktor pak volá jediné – `register()` za účelem přidání odkazu do tabulky instalovaných appletů v JCRE.

Zmíněnou metodu musí volat každý a to v rámci `install` metody. Zajistíme si tímto správnou komunikaci s řídícím JCRE. Applet na něm závisí v oblasti komunikace s některou z metod pro aktivaci, deaktivaci. Nesmíme zapomenout, že jednovláknová JCRE v případě nekonečné smyčky v appletu potřebuje odpojení karty k tomu, aby se mohla vrátit do výchozího stavu.

Aktivace zvoleného appletu

Dalšími metodami v třídě `Applet` jsou `select()` a `deselect()`. Oproti předchozím tyto nepodporují příjem APDU paketů. JCRE filtruje datový tok APDU mezi applety a čtecím zařízením. Metody pro aktivaci jsou volány jen v případě, když je přijat APDU s příkazem k vybrání appletu.

Jak už bylo několikrát zmíněno, Java Card applet má přiřazeno unikátní číslo AID (Aplikační ID). Určuje vybraný applet v rámci selekce. Přijde-li do JCRE paket APDU typu SELECT, je předchozí applet deaktivován a vybere se applet nový, dle dat v příkazovém paketu. Je zavolána metoda `select()` před posláním APDU typu SELECT do metody `process()` vlastněné nově vybraným appletem. Dojde k volání tedy hned dvakrát. Metoda, která starému appletu předá informaci o zrušení výběru je metoda `deselect()`.

Existují dva typy APDU s příkazem SELECT pro výběr:

1.) Pro zvolení appletu.
2.) Pro zvolení souboru ve vybraném appletu.

Po přijetí takového paketu si JCRE převezme AID číslo z dat a hledá v tabulce nahraných appletů. V případě neúspěchu považuje JCRE tento příkaz jako volba souboru a předá toto do zpracující metody aktuálně vybraného appletu.

Obecná implementace `select()` navrací hodnotu TRUE, při správné aktivaci appletu a jeho nové schopnosti přijímat APDU. Jinak se vrací hodnota chybového hlášení. Metoda `deselect()` nevrací nic. Dvě poslední metody musí být vytvořeny, aby byly volány JCRE a nedošlo k případné výjimce. Vše by se ještě jinak muselo provádět při zpracování metody `install()`. Implementace metody `register()` není přepisovatelná. Vkládá odkaz na objekty appletu do tabulky registrovaných appletů. Díky tomuto je možné následně volat metody pro výběr appletu. Jakmile dojde k aktivaci, je možné zasílat příkazy v paketu APDU a nechávat zpracovat applet zvolené instrukce.

Princip APDU zpracování

První zpracování následných APDU příkazů do appletu se děje voláním `APDU.getBuffer()` k získání datového paketu odpovídajícího APDU. Ten obsahuje hlavičku a data s příkazem. JCRE poskytuje prostředky pro rozpoznávání druhů paketů. Použití ISO.OFFSET_XXX a obecnou skladbu APDU dokumentuje následný příklad kódu (Tab. 5.6). Stavba paketu odpovídá tabulkám 2.1 nebo 2.2. Podrobnější princip a ukázky datové části APDU i bloku hlavičky si popíšeme v kategorii ukázky návrhu v JCT Suite Smart Café.

Výpis kódu 4.4: Získávání hodnot hlavičky APDU ve zdrojovém kódu

```
byte buf[] = apdu.getBuffer();
byte cla = buf[ISO.OFFSET_CLA];
byte ins = buf[ISO.OFFSET_INS];
byte lc = buf[ISO.OFFSET_LC];
byte p1 = buf[ISO.OFFSET_P1];
byte p2 = buf[ISO.OFFSET_P2];

//pro zjištění některého údaje voláme:
Util.arrayCopy(buf, ISO.OFFSET_CDATA, databuf, 0, lc);
```

Práce s pakety APDU

S APDU příkazy se setkáváme v metodě `process` potřebné ve všech appletech psaných v Javě pro Java Card.

Už zmíněný tzv. buffer slouží k uložení APDU hlavičky příkazu. Applet si zavolá o referenci na APDU buffer. Provede činnost dle typu příkazu a vrátí odpověďní APDU paket. JCRE potřebuje zajistit, že odkazy na APDU objekt nebo buffer nesmí být sdíleny v proměnných třídy, instancích proměnných a polí. Applet může ale sdílet odkazy v rámci lokálních proměnných a metod, což omezí použití pouze v rámci těchto metod. Toto všechno je kvůli zajištění omezení zisku reference na ADPU data jiného appletu.

Skupina některých příkazů může být ve vývojovém prostředí automatizována. V našem případě u programu JCT Suite je inicializační mechanismus spuštěn po vybrání karty. Sled autorizačních procesů je proveden několika APDU výzva – odpověď. Naopak je zde umožněno také prosté posílání APDU (výzva k PINu, nabití karty atd.). Po vyplnění jednotlivých položek hlavičky a dat lze zprávu odeslat manuálně.

4.2 Pracovní prostředí JCS Suite Smart Café v3.0

Ve světě čipových karet, se mluví o ekvivalenci Java Card technologie a programování klasických Java aplikací. Pro JCS Smart Café [19] je třeba speciálních pravidel a nástrojů, pro podporu vývojářů appletů zaměřených na JC. Ty dokážou plně využít nabízených schopností.

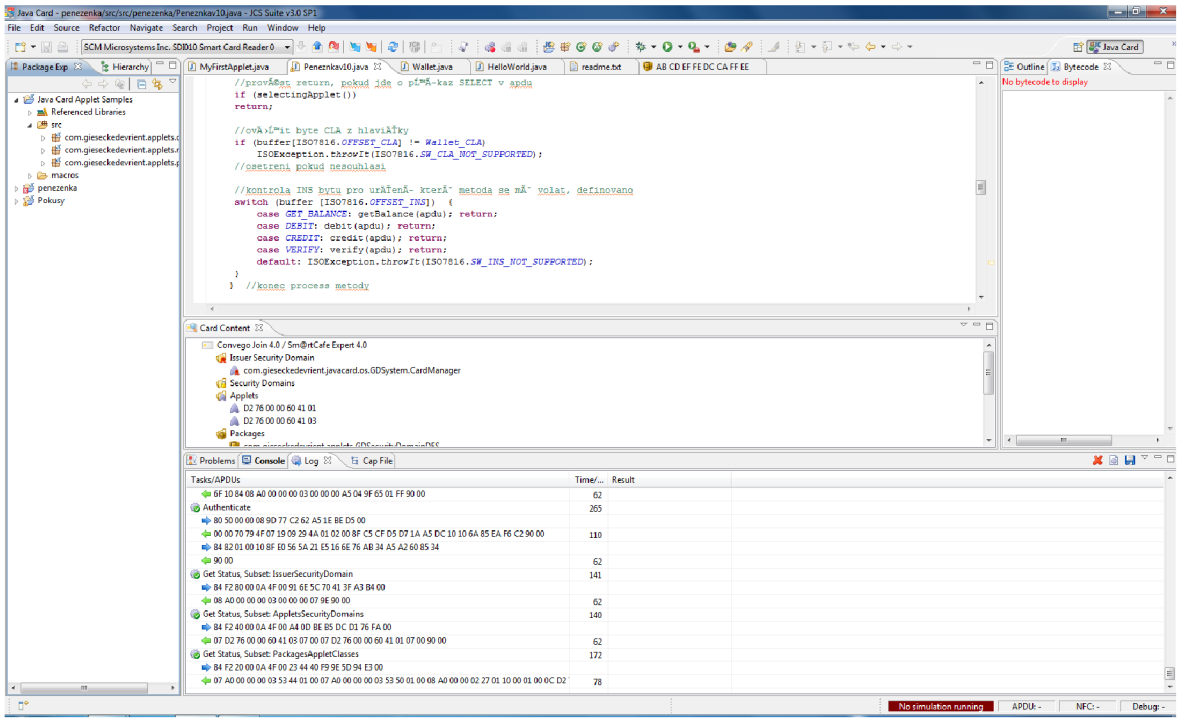
První aplikace pro vývoj appletů na karty, se stala v roce 1998 Sm@rtCafé Java Card od firmy Giesecke & Devrient (G&D). Součástí bylo IDE vývojové prostředí propojené s balíčkem Sm@rtCafé.

Postupným vylepšením dalších verzí, které se staly kompatibilní s programovacími prostředími Eclipse, NetBeans apod., zatím dosáhl vývoj poslední aplikace nazvané JCS

Suite v3.0. Kompletní sada nástrojů v nové verzi nabídne přehledné rozhraní pro podporu ve vývoji, testování a kompilaci appletů JC pro karty podporující SmartCafé a další.

Celý systém JCS Suite 3.0 [9] dostupný pro naši práci není tolik rozšířený. Jde spíše o vývojové prostředí pro laboratorní účely společnosti G&D. Masovému použití brání vyšší cena, dále také existence jednodušších nástrojů pro Java Card.

Pro představu na dalším obrázku vidíme celé prostředí popisované platformy. Všechny obrázky z prostředí JCS jsou v lepší formě uvedeny na přiloženém médiu diplomové práce.

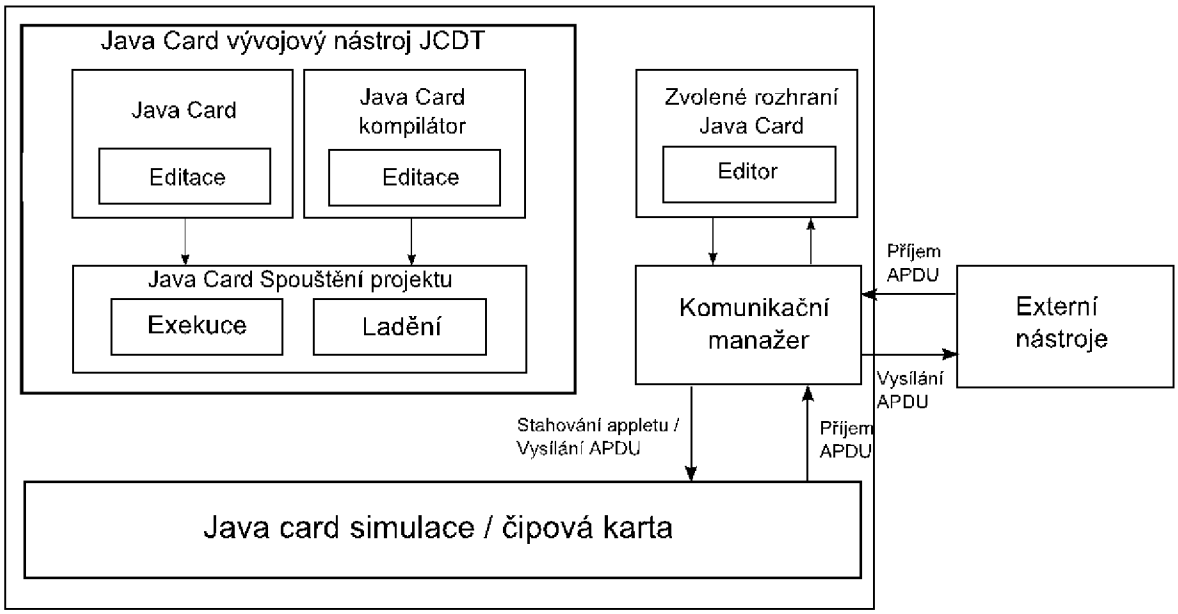


Obr. 4.2: Prostředí JCS Suite v3.0

4.2.1 Hlavní dispozice

JCSuite nabízí ucelené prostředí testování k vyvíjení appletů a Java aplikací. Součástí jsou např.:

- IDE vývojové prostředí založené na Eclipse.
- Podpora JCVM a starších verzí Smart Café.
- Standardy OpenPlatform a Java Card 2.x.x.
- Nahrávání appletů na skutečné čipové karty.
- Makra a knihovny pro posílání APDU příkazů.
- Hardware čtečka SCM Microsystems SDI 010.



Obr. 4.3: Architektura JCS prostředí [9]

Hlavními komponenty prostředí jsou Java Card Simulace, Exekutor a Komunikační manažer. Důležitými schopnostmi se přiblížíme reálnému použití:

- JC Simulace provádí spuštění v terminálu
- JC Spouštěč nahrává applet do simulátoru a nabízí spuštění a proces debug
- JC Manažer zodpovídá za stažení celého programu do simulátoru nebo karty, provádí též správu APDU příkazů a maker
- Externí spuštění Java aplikací pro komunikaci s kartou je umožněno speciálním TCP/IP rozhraním

4.2.2 JC vývoj a kompilace

Integrace s prostředím Eclipse (dodatečně NetBeans) umožňuje využít nástroje pro vývoj a správu JC appletů. Pro náš případ byl použitý právě JCS Suite pracující na jádru Eclipse. Využívá ho jeden z největších výrobců a distributorů karet společnost G&D z Německa. Dostupný virtuální simulátor dokáže emulovat vlastnosti fyzických karet.

Všechen vývoj, kompilace a finální testování je zapouzdřeno do vytvořeného JC projektu. (využijeme průzkumník projektu). Každému z nich jsou nabídnuty volitelné vlastnosti simulace včetně knihoven.

4.2.3 JC tvorba apletů

Proces vzniku appletů není nepodobný vzniku jakéhokoli programu v jazyce Java. Pouze jsou zde navíc prostředky k vytvoření speciálních tříd a balíčků pro Java Card. Je třeba zprvu určit vlastnosti appletů – identifikační čísla AID, verze, volby pro stažení na karty. Pro kompilaci je využito klasického JCDK vývojového prostředí od Oracle, s charakterizací případných chyb a doporučení řešení.

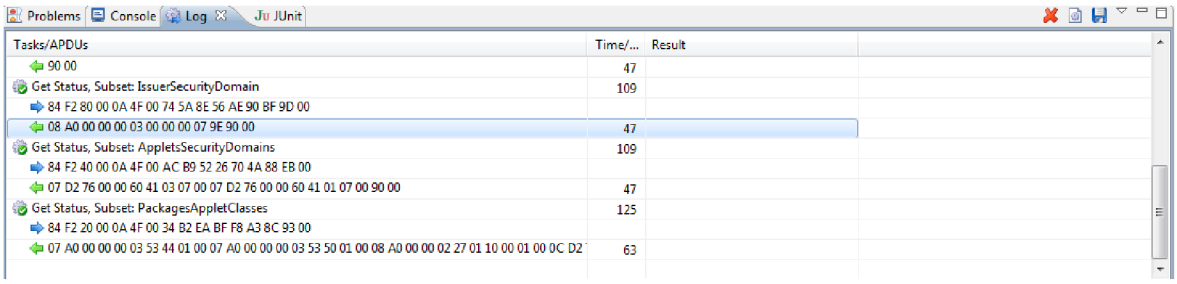
4.2.4 Spuštění appletu

Projekty mohou být odsimulovány nezávisle. Stěžejní bod spočívá ve spuštění appletu v režimu DEBUG nebo RUN. Je možné využít simulátoru, kdy není třeba karty, a to hlavně v případě ladění. V ostatních případech je při procesu aktivováno spojení s kartou a následuje nahrávání appletu. Můžeme konfigurovat APDU příkazy do maker a ty posílat do appletu, případně očekávat reakci na ně.

K zastavení běhu programu dochází, pokud uživatel uzná za vhodné, případně při výskytu chyby. V případě přerušení násilného, se po znovuzapojení objekty v zásobníku karty vrací do posledních uložených pozic. Stejně tak je možné nechat kartu rovnou vyresetovat. Jde o jeden bezpečnostní prvek v Java Card – atomicitu, nebo-li uvedení proměnných a transakcí do výchozího stavu před přerušením operací.

4.2.5 Ladění JC

Spuštěné applety lze ladit, krok za krokem v rámci debug módu. V rámci prostředí JCS je myšleno i na použití skutečných čipových karet. Nachází se zde duplexní rozhraní a ovladače pro komunikaci se čtečkou. Celá komunikace APDU se objevuje ve výpisu APDU ve speciální tabulce na způsob následujícího.



Obr. 4.4: Výpisy komunikačních paketů APDU

4.2.6 Nastavení Java Card

Schopnosti JC

- Zobrazení a modifikace různých parametrů karty, víceúrovňové autentizace, protokolová podpora
- APDU příkazy a sestavování maker, zabezpečení JCVM
- Nahrání appletů a práce s nimi, zobrazení komunikace do přehledné konzole
- Výpis informací ISD bezpečnostní domény karty

Konfigurace a manažer

Podle vybrané verze Smart Caffé (karta podporuje určité prostředí a příkazy), jsou zobrazeny informace o simulátoru nebo kartě. Životní cyklus karty (CPLC) s jeho daty je do určité míry možné měnit.

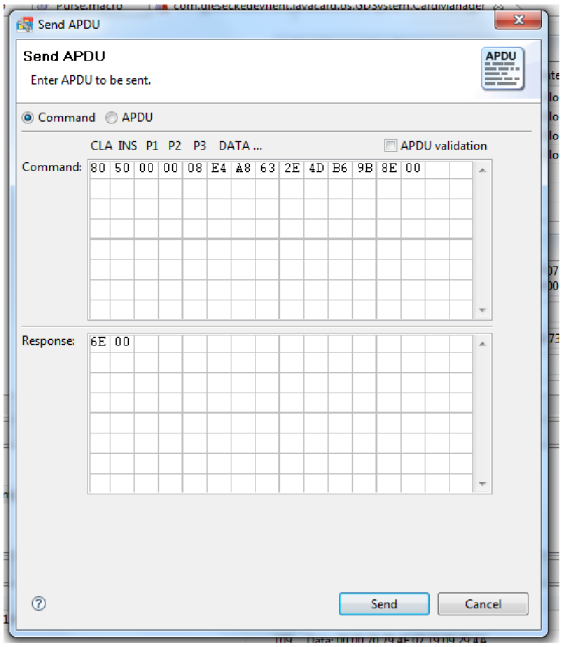
Manažer tedy poskytuje kontrolu nad kartou promocí komunikačního okna Issuer Security Domain (ISD), poskytující ucelený pohled na zbývající volné místo v pamětích, číslo AID, nastavené procesy autentizace klíči, a další důležité informace.

Komunikace s JC

V první řadě uvedeme nejběžnější formu komunikace, využitím nástroje pro tvorbu APDU. Lze nastavit i očekávanou odpověď na ADU dotazy. Seznam všech se nachází v normě ISO, která nebyla k dispozici, z finančních důvodů. Jsme omezeni na několik základních typů, které se používají v rámci manuálu. Rozpoznávání dat a instrukcí z APDU příkazu je velice komplexní a bylo popsáno v jedné z dřívějších kapitol.

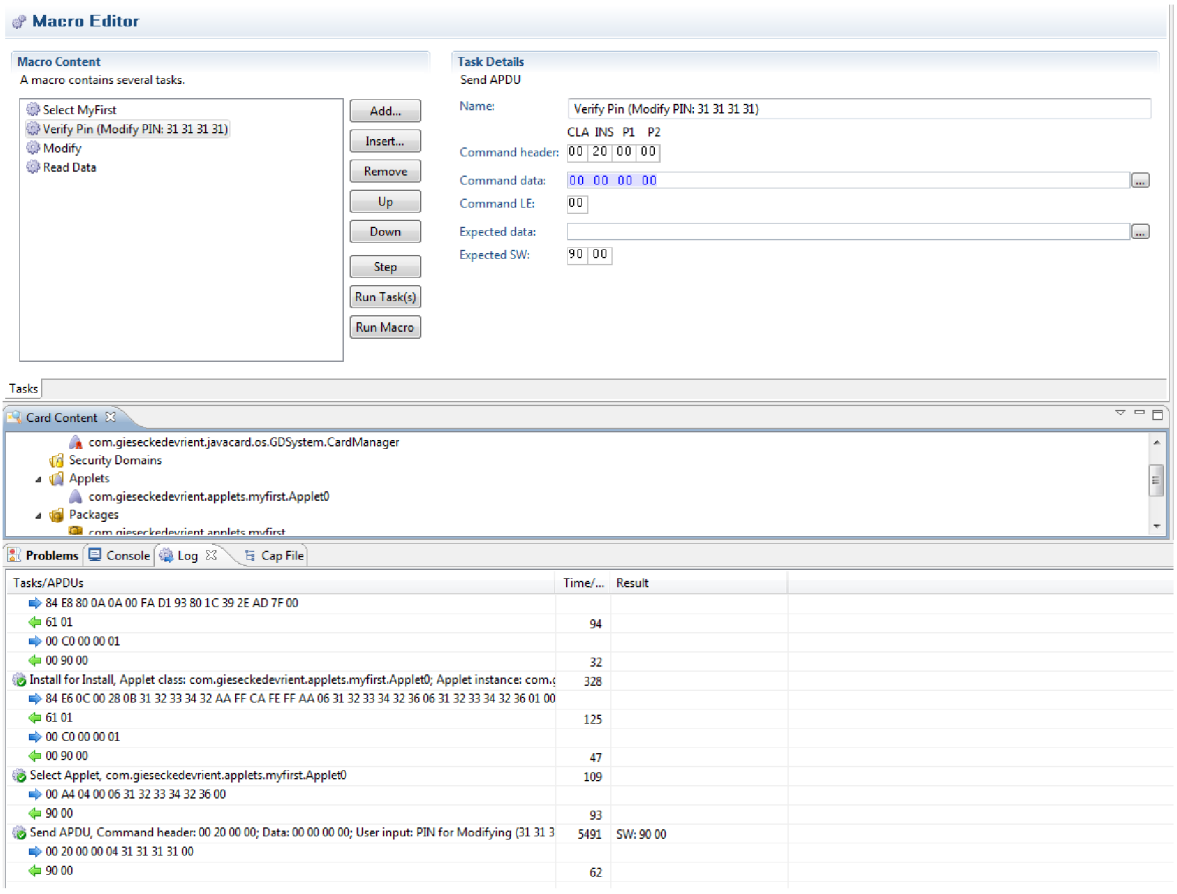
Makra

Editor maker poskytuje nabídku generování APDU a zadávání sekvencí na sobě závisejících, speciálně v souvislosti s nahráváním balíčků appletů. Pokud neuvažujeme některé automatické sledy paketů poskytnuté v prostředí JCS Suite, je nám dovoleno zadávat příkazy ručně. To sebou nese větší pravděpodobnost chyby odezvy. Lze si tak lépe vyzkoušet celý princip. V následném obrázku vidíme tabulku pro zadávání jednotlivých parametrů APDU.



Obr. 4.5 : Manuální zadání APDU

Speciálně v případě aplikace pro zadávání PINu pro autentizaci jako jednoho ze zkoumaných appletů, bylo použito maker vypadajících následovně. Jde o pravý opak principu zadávání ručně.

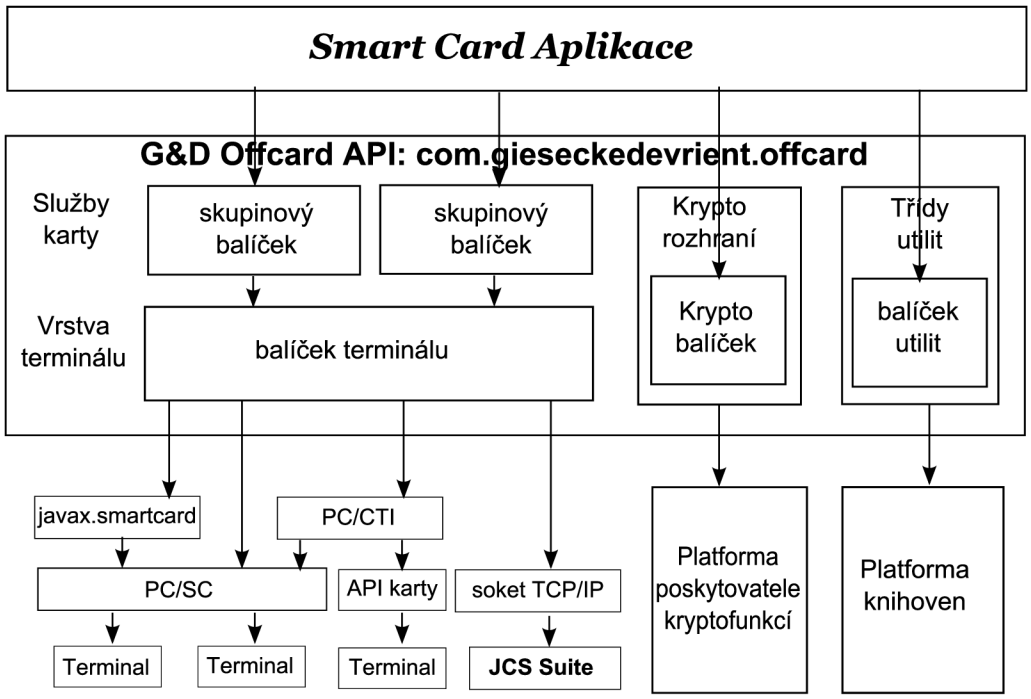


Obr. 4.6 : Zadávání APDU pomocí maker u autentizačního appletu

4.2.7 Kód na kartě a propojení s běžnou Java aplikací

Rozhraní vně karty, tedy Off-Card umožní uživateli zahrnout kód v Java Card do běžné Java aplikace. Lze vytvořit program spuštěný v JCS Suite na počítači, komunikující s appletem na čipové kartě. Šlo hlavně o případ peněženky.

Můžeme konstatovat, že systém vytvořený v rámci Java Card tvoří dvě části. Applet na kartě a k němu související aplikace mimo kartu (v PC – není podmínkou). Aplikace na počítači komunikuje přes terminál a bezpečnostní logický kanál.



Obr. 4.7 : API rozhraní na hostitelské stanici pro možné připojení více terminálů [9]

Program může použít vyšší službu uživatelského rozhraní API pro komunikaci s nižší službou. Je možné projít krok za krokem celý běh aplikace, pokud jsou na obou stranách komunikační rozhranní spuštěny současně. Jednoduchý applet bez rozšířenějšího zabezpečení simuluje elektronickou peněženku. Můžeme nabíjet finanční částku a samozřejmě čerpat. Simulačně, i na kartě, lze dále spustit odlišný applet pro autorizaci PIN kódem. Odezva je zobrazena přímo ve vývojovém prostředí

4.3 Návrhy

Po seznámení se základními principy Java Card a vývoji aplikací pro tuto platformu v rámci prostředí JCS Suite můžeme předvést několik konkrétních ukázek práce s elektronickou peněženkou a autentizace PIN kódem v rámci dvou oddělených appletů.

Zdrojový kód appletu pro určitou identifikaci uživatele na základě znalosti tajného kódu je v celkové formě uložen na přiloženém médiu k diplomové práci (viz příloha A – Obsah CD). Applet peněženky lze nalézt spolu s dalšími dodatečnými materiály na stejném místě.

Formy APDU

Seznam nejobecnějších typů APDU příkazů s jejich charakteristikou lze najít v příloze CD. Zde zmíníme několik základních pro práci s bezpečnostní doménou a pro komunikaci appletů. Kompletní seznam možných APDU se nachází v normě ISO 7816-4, ale ty používané v JCS Suite jsou v informačním dokumentu Smart Café [17]. Některé zmíníme.

Select

Pomocí něj volíme applet na otevřeném kanálu. Každé další příkazy APDU směřují jemu. Pokud se kanál ještě neotevřel, provede se a dle AID se prohledá seznam appletů v databázi. Při shodě se aktivuje.

Tab. 4.1: Forma APDU příkazu Select

CLA	**INS**	**P1**	**P2**	**Lc**	**Data**
0X	A4	04	00		

CLA – Zvolení appletu na vybraném kanálu.

P2 – Zvolí se jeden Applet dle plného jména, anebo větší počet shodujících se s částečným názvem.

Lc – Délka AID 5 až 15 bajtů.

Data – Obsahuje konkrétní identifikační AID, případně parciální pro zvolení více appletů.

Delete

Provádí operaci smazání pro balíček nebo applet. V bloku APDU – Data se objevují speciální TLV identifikátory mazaných položek (T – typ, L – délka, V – hodnota).

Tab. 4.2: Forma APDU příkazu Delete

CLA	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**
	E4	00				01

CLA – „80“ zajistí komunikaci přes bezpečný kanál 0.

P2 – „00“ určuje mazání objektu, „80“ maže i relevantní data.

Lc – Délka příkazu v Data položce.

Data – Obsahuje identifikátor aplikace, kterou mažeme. Tvar je následující:

Tab. 4.3: Forma APDU příkazu TLV Delete

T	**L**	**V**
4F	05 až 10	AID appletu / balíčku

External Authenticate

Autorizuje hosta a určuje úroveň zabezpečení komunikace. V položce Data je obsažen kryptogram.

Tab. 4.4: Forma APDU příkazu EA

CLA	**INS**	**P1**	**P2**	**Lc**	**Data**
84	82		00	10	„Kryptogram“

P1 – Právě tento parametr určuje zabezpečení, „00“ platí pro před-zabezpečenou kartu, „01“všechny příkazy musí obsahovat kódovou autentizací zprávu MAC pro hlavičku a data „03“ to stejné, navíc musí být šifrováno.

Get Data

Příkaz umožňuje získat z karty některá charakteristická data a informace o paměti (velikost EEPROM, privátní klíče, informace ISD). V paketu APDU záleží na CLA parametru, a odpovídající paket obsahuje Data s informací o vyžádané službě.

Tab. 4.5: Forma APDU příkazu GetData

CLA	**INS**	**P1**	**P2**	**Le**
	CA			

CLA – „00“ volán jak příkaz dle ISO, „8X“ zvolen bezpečnostní kanál x.

P1/P2 – Ukazatel (značka) na požadovaná data, „00 66“ je např. identifikační číslo výrobce

Le – Délka očekávaných dat.

Data – Obsahuje hodnotu obdržené položky ve formě identifikátoru či TLV kódu.

Get Status

Příkaz vyžádá informace o životním cyklu karty pro aplikace, balíčky a ISD. V položce Data definujeme AID či status hledané položky (initialized, secured, atd.). V tomto případě je bezpečnostní kanál vyžadován.

Tab. 4.6: Forma APDU příkazu GetStatus

CLA	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**
	F2				Kritéria	

CLA – Úroveň zabezpečení kanálu, např. „80“.

P1 – Zahrnuje pod-složky stavů, které chceme zahrnout v odpovědi, „40“ pro aplikace ISD

P2 – Značí požadavek na vyslání dat pro přesnou velikost APDU, či pro jinou než předchozí, tedy „00“ a „01“.

Lc – Délka dat příkazu.

Data – Buď AID, nebo životní cyklus hledané položky, příkazová část „4F 00“ bity + AID.

Initialize Update

Využití nachází v momentu potřeby posílání dat mezi dvěma komunikujícími stranami v Java Card. Jejím použitím získáme i nové klíče pro relaci a nastavení bezpečného kanálu.

Tab. 4.7: Forma APDU příkazu Update

CLA	**INS**	**P1**	**P2**	**Lc**	**Data**	**Le**
80	50		00	08		1C

P1 – Volí se číslo verze klíče určující sadu klíčů v rámci ISD pro vytvoření zabezpečené relace, „00“ je výchozí nastavení sady klíčů

Data – Výzva od hosta, náhodné 8 bajtové číslo.

Install for Load

Zasílá informace kartě o možných krocích, které jsou požadovány k nahrávání appletů.

Tab. 4.8: Forma APDU příkazu Install for Load

CLA	INS	P1	P2	Lc	Data	Le
	E6	02	00			01

CLA – Volí se číslo bezpečnostního kanálu (Secure Channel 0 až 3 viz. [x]).

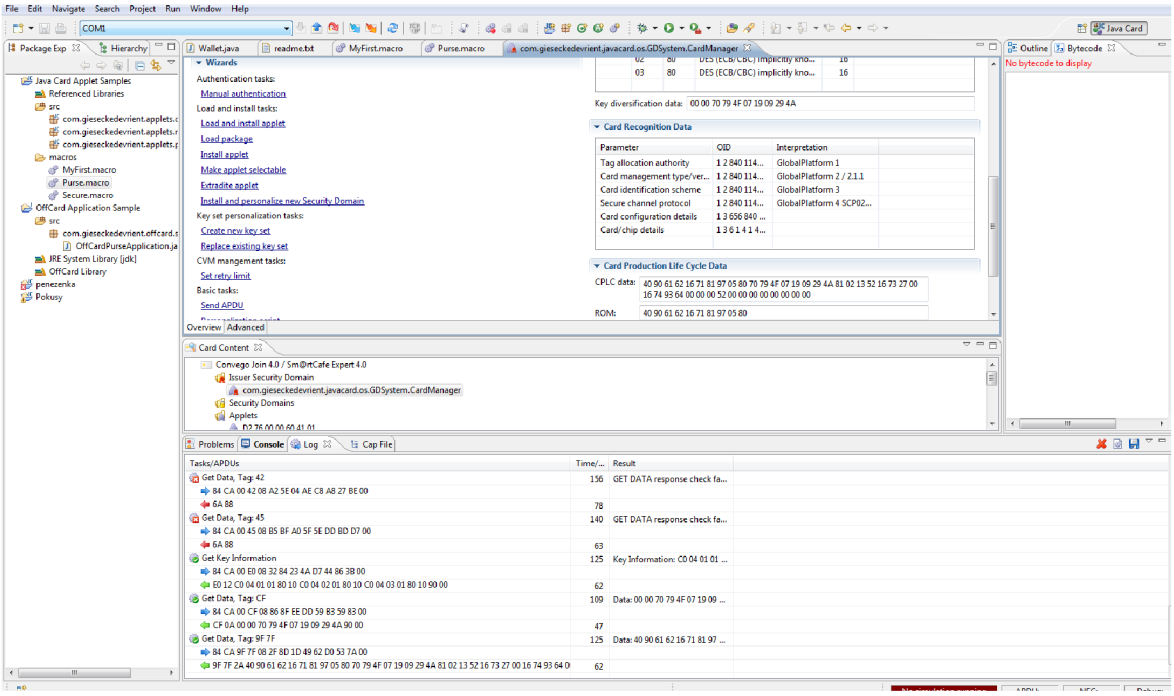
Lc – Délka dat v příkazu.

Data – Struktura informačních dat v této položce může obsahovat např. balíčkovou identifikaci AID, bezpečnostní doménu a její bajtovou délku, různé parametry.

Potvrzení správnosti se jako vždy provede odpovědí „90 00“, případně jiným kódem s definovanou správou o chybě. Podrobněji samozřejmě v manuálu k Smart Caffé Suite.

Testování

Karta Convego Join 4.0 Smart Café disponovala kontaktním i bezkontaktním rozhraním komunikace. Po vložení karty do čtecího zařízení firmy SCM Microsystem je nutné kartu zaregistrovat. Poté provádíme autorizaci karty a výměnu tajných klíčů mezi prostředím JCS a bezpečnostní doménou ISD na čipové kartě. Sled jednotlivých APDU názorně ukazuje výstřižek z námi využívaného prostředí.



Obr. 4.8 : Náhled na výpis APDU při autorizaci

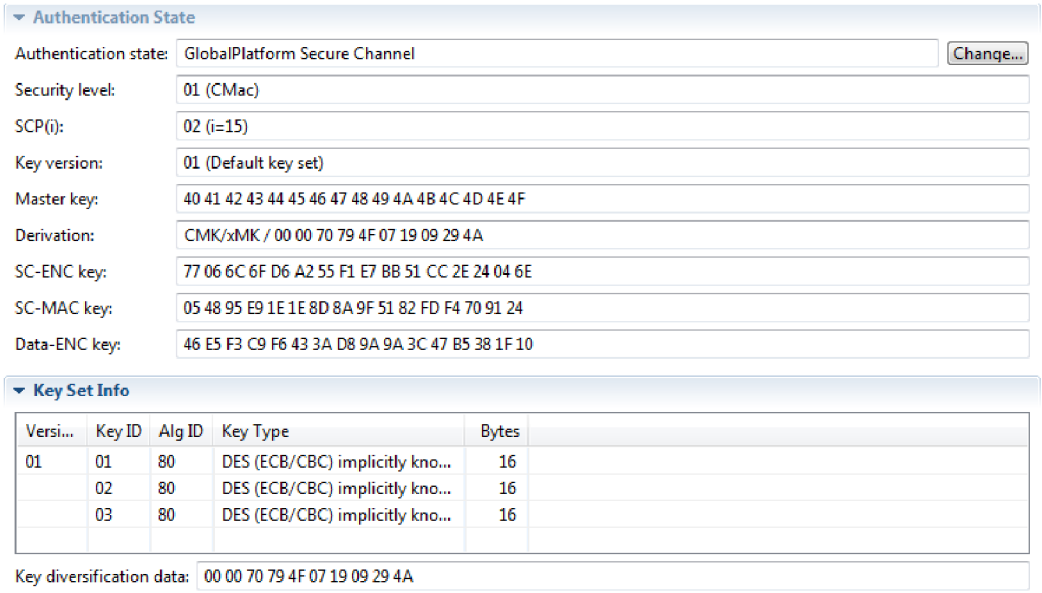
Celkový sled výměny dat lze exportovat v rámci XML dokumentu a uložit na libovolné místo. Menší část dokumentu je zobrazena níže. Vidíme zde informační popis akce. Dále vzhled příkazového a odpovídajícího APDU paketu a čas operace. Pro celkové výpisy je možné nahlédnout na přiložené médium. V následujícím obrázku jsou uvozovací prvky jazyku XML pro informaci o operaci (<info>), dále ohraničení příkazu APDU (<CommandApdu>) a odpověď (<ResponseApdu>). Položky typu <time> zahrnují informaci o době celé operace v milisekundách.

Výpis kódu 4.5: Exportovaný soubor s APDU příkazy při aktivaci karty

```
<Info>Connect, SCM Microsystems Inc. SDI010 Smart Card Reader 0</Info>
<CommandApdu>80 CA 9F 7F 00</CommandApdu>

 <ResponseApdu Time="31">9F 7F 2A 40 90 61 62 16 71 81 97 05 80 70 79 4F 07 19 09 29
4A 81 02 13 52 16 73 27 00 16 74 93 64 00 00 00 52 00 00 00 00 00 00 00 00 90
00</ResponseApdu>

<CommandApdu>00 A4 04 00 00</CommandApdu>

<ResponseApdu Time="16">6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF 90
00</ResponseApdu> </Task>

-<Task Time="63"> <Info>Select Applet,
com.gieseckedevrient.javacard.os.GDSystem.CardManager</Info>

<Result>FCI: 6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF</Result>

 <CommandApdu>00 A4 04 00 08 A0 00 00 00 03 00 00 00 00</CommandApdu>
```

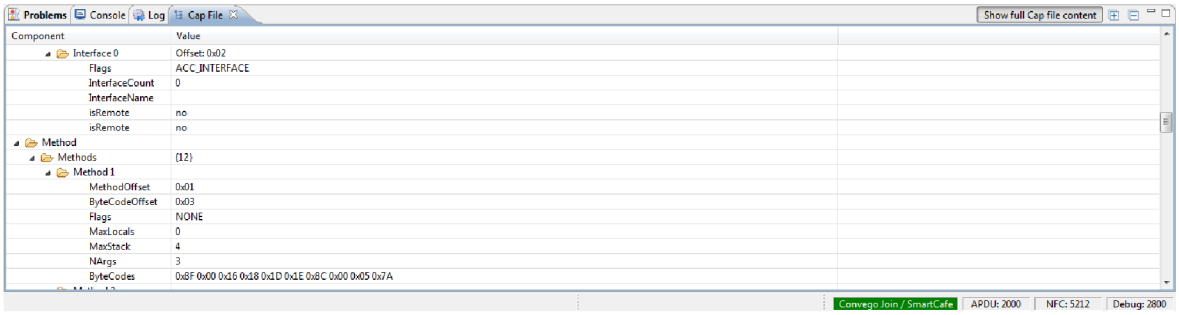
Ve zkratce v zobrazeném skriptu se aktivuje karta a používá se metoda `select` pro vybrání appletu pomocí bezpečnostní domény. Doména zároveň informuje uživatele o svých možnostech, velikosti paměti, variaci klíčů a verzi systému. Celá tato doména ovládá fungování čipové karty. Prostřednictvím ní se mění životní cykly karty, zjišťují podporované kryptografické metody a určují vlastnosti.

Pomocí ISD domény se dá nahrát applet na kartu, pokud nevyužijeme přímou možnost spuštění zdrojového kódu v prostředí JCS, které automaticky převede celý správně zkompilovaný kód do formy CAP a nabídne se samotné stažení appletu na kartu. Z obrázku pod tímto textem, ukazující na část rozšířených informací o doméně karty, můžeme odvodit stav pro autentizaci, klíče i využití např. DES algoritmů.



Obr. 4.9: Údaje z ISD o použitých klíčích

Jako další krok jsme stáhli převedený applet pro každou z aplikací na kartu. V rámci logu pro zobrazení CAP, jsme mohli vysledovat obecnou strukturu CAP souboru s jeho podsložkami. Zmíněný náhled ukazuje část bajtového kódu rozhraní a jedné metody z původního zdrojového kódu. Každá třída i metoda má v CAP převedeném appletu svoji reprezentaci.



Obr. 4.10 : Práce se souborem CAP

Pro peněžní aplikaci se nejdříve aktivoval applet pro zpětnou komunikaci na kartě. Ten byl převeden, již známým způsobem, do výchozí podoby ze tří původních *.class souborů. Jsou v nich informace o použitých metodách, atributech potřebných pro chtěné operace. V konzolovém okně jsme ověřili nahrání na kartu. Prostřednictvím obyčejné a jednoduché Java aplikace spuštěné na počítači též v prostředí JCS, komunikovaly obě strany (Java aplikace => applet na kartě) mezi sebou na základě následujícího způsobu (výzva - odpověď).

Výpis kódu. 4.6: Výpis z konzoly při změně PINu

```
Start Installing Applet --> Class AID: '31 32 33 34 31 36'

Instance AID: '31 32 33 34 31 36'

Applet '31 32 33 34 31 36' installed.

...

Purse Balance: 10000

Available Operations (Activate Console, press key and <RETURN>):

a - Verify Credit PIN

b - Verify Debit PIN

c - Credit Purse

d - Debit Purse

q - Quit

a

Enter Credit PIN (4 digits, default 2000):
2000
```

Jak vidíme, interakce je zajištěna pomocí zadávání písmen z klávesnice. Ani jednoduché grafické GUI rozhraní, není v této verzi Java Card stále ještě povoleno. Jde o jednoduché nastínění principu elektronické peněženky, která v mnohem složitější formě může fungovat jako aplikace od dnešních výrobců karet v běžném životě.

Několik příkazů APDU pro první z appletů, kdy se autentizuje přístup a načítá bezpečnostní doména i klíče u peněžního appletu, je uvedeno níže.

Výpis kódu 4.7: Purse_Applet_APDU.xml

```
-<Task Time="312"> <Info>Authenticate</Info>

<CommandApdu>80 50 00 00 08 06 2F 00 5A D8 77 28 3B 00</CommandApdu>

<ResponseApdu Time="125">61 1C</ResponseApdu>

<CommandApdu>84 82 01 00 10 DC A2 E2 32 96 DA 15 7D F2 B9 82 EB 15 3C
8C CB</CommandApdu>

<ResponseApdu Time="62">90 00</ResponseApdu> </Task>
...
-<Task Time="187"> <Info>Get Status, Subset:
IssuerSecurityDomain</Info>

<CommandApdu>84 F2 80 00 0A 4F 00 F3 1A A6 3B 3F CA 49 39
00</CommandApdu>

<ResponseApdu Time="78">61 0B</ResponseApdu>

<CommandApdu>00 C0 00 00 0B</CommandApdu>

<ResponseApdu Time="47">08 A0 00 00 00 03 00 00 00 01 9E 90
00</ResponseApdu> </Task>
...
-<Task Time="125"> <Info>Get Key Information</Info>

<Result>Key Information: C0 04 01 01 80 10 C0 04 02 01 80 10 C0 04 03
01 80 10</Result>

<CommandApdu>84 CA 00 E0 08 32 84 23 4A D7 44 86 3B 00</CommandApdu>

<ResponseApdu Time="62">E0 12 C0 04 01 01 80 10 C0 04 02 01 80 10 C0
04 03 01 80 10 90 00</ResponseApdu> </Task>
```

APDU příkazy jsou zapisovány v hexadecimální soustavě. Prvních šest čísel definuje hlavičku příkazu. Odpověď je vždy kratší a nemusí obsahovat data (potvrzení : „90 00“).

Podrobnější náhled na APDU skýtá kapitola 4.3 a 2.4.3.

Zde si ukážeme vybraný sled APDU pro vybrání appletu autentizace a následná možnost jednoho z vytvořených příkazů makra pro změnu PINu nebo provedení vybrání daného appletu (zde obsahuje APDU informace o AID appletu `32 31 31 32 32 30 30 31` apod.).

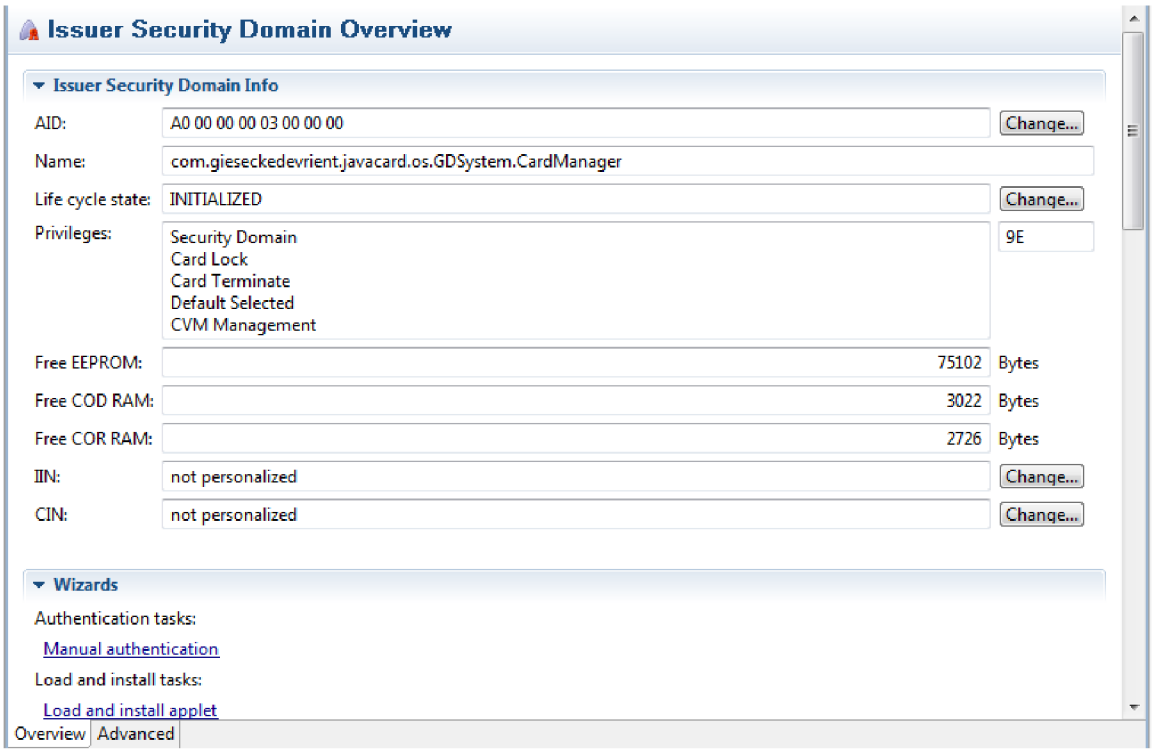
Výpis kódu 4.8 : Autentizace_Applet_APDU.xml

```
-<Task Time="109"> <Info>Select Applet,
com.applets.myfirstAutent.Applet0</Info>

<CommandApdu>00 A4 04 00 06 31 32 33 34 32 36 00</CommandApdu>

<ResponseApdu Time="93">90 00</ResponseApdu> </Task>

-<Task Time="5491"> <Info>Send APDU, Command header: 00 20 00 00;
Data: 00 00 00 00; User input: PIN for Modifying (31 31 31 31) (0,
4)</Info>

<Result>SW: 90 00</Result>

<CommandApdu>00 20 00 00 04 31 31 31 31 00</CommandApdu>

<ResponseApdu Time="62">90 00</ResponseApdu> </Task>

-<Task Time="156"> <Info>Send APDU, Command header: 00 D6 00 00;
Data: 4A 6F 68 6E FF 44 6F 65 FF 32 31 31 32 32 30 30 31 FF 31 36
32 31 31 32 30 31 53 78 78 78 FF</Info>

<Result>SW: 90 00</Result>
```

Ověřili jsme si možnost platformy Java Card, pouze jednoho aktivního appletu v jednom momentě. Příkazy do appletu autentizace se zadávaly pomocí makra, jako je v Obr. 4.6. Peněžní aplikace poskytne komunikaci na základě výměny dat mezi appletem na kartě a jednoduchým programem na hostitelské stanici. Pomocí zadávání na klávesnici nebo pomocí maker jsou provedeny operace odpovídající příkazům v jednotlivých APDU. V následující části stručně popíšeme využitý hardware v laboratorních podmínkách. V kapitole 5 se zaměříme na bezpečnost celého systému, nutných pravidel pro správné používání Java Card i čipových karet v každodenním životě.

4.4 Využitý hardware

Dostatečně kvalitní počítačová sestava vybavená systémem Windows 7 a softwarem JCS Suite v3.0 byla základním předpokladem. Přes USB rozhraní bylo možné napojit čtecí zařízení s duálním rozhraním SCM Microsystems SDI011. Čipová karta pro laboratorní a testovací účely byla od společnosti Giesecke & Devrient (G&D) typu Convego Join 4.0. Nadcházející obrázek poskytuje jedno ze základních oken informací o bezpečnostní doméně, zobrazující parametry karty.



Obr. 4.11: Základní parametry karty zobrazené v ISD

4.4.1 Karta

Společnost G&D, od které pochází prostředí JCS Suite - Smart Caffé, se zabývá hlavně vývojem čipových karet [18] a jejich distribuce už delší dobu. Tato německá firma se zaměřuje hlavně na vývoj pro platební aplikace a produkty do celého světa. Patří sem prostředky vybavené specifikací EMV (viz kap. 2), karty s duálním rozhraním drátové/bezdrátové komunikace s terminálem. Kreditní čipové karty jsou dostupné pro platformy založené na bezpečných operačních systémech. Dále nabízejí servisní služby a projekt management. Díky vysokému postavení na trhu poskytuje firma podporu adaptace na nové technologie.

Jednou z oblastí poskytovaných zákazníkům je řada produktů GD Convego Join Multi-pplication. V jednodušších verzích ji lze rozšířit mezi koncové zákazníky pro laboratorní využití.

Světové bankovní domy používají produkty Convego Element pro zabezpečenou EMV a Eurocheque standardizaci. Na platformě Convego Element s podporou Open Platform jsou postaveny karty Visa a Mastercard. Ještě zmíníme Convego Air pro bezdrátové potřeby čipových karet.

Convego Join pro laboratorní účely, jsou navrženy pro řešení více appletů na jedné kartě. Zvýrazňují podporu Java Card specifikace v2.2.1, v2.2.2. Jsou zde zahrnuty funkce

a algoritmy pro co nejschopnější míru zabezpečení. Disponují promyšlenou komunikací a formou paměťového systému.

Kartu specifikujeme [17]:

- název: GD GmbH – Convego Join 4.0 DI / SMart Caffé Expert 4.0 MDI36-1,
- komunikační protokol: T0/T1,
- autentizace: statická i dynamická,
- datové a adresní formáty: 8,16 bitové,
- paměti: EEPROM 36 KB, ROM 236 KB, RAM 6 KB,
- garantováno 500,000 přepsání EEPROM,
- výrobce čipu: Philips Infineon – SLE66CLX360PEM.

4.4.2 Čtecí zařízení

Zařízení společnosti SCM Microsystems disponuje možností spojení s kartou kontaktně nebo bezdrátově. Konkrétní model SDI011 má několik schopností:

Obr. 4.12 : Čtečka čipových karet SCM SDI011

- dvojí rozhraní pro komunikaci s kartou ISO 7816, ISO 14443,
- velikost 63 x 64 x 10 mm,
- nová verze SmartOS firmware,
- ovladače pro Windows, Mac OS i Linux,
- rozhraní USB pro komunikaci s počítačem a napájení,
- použitelnost pro e-Banking, elektronickou ID.

Nové znalosti

Nyní zkusíme shrnout několik nových poznatků. Obě spuštěné aplikace se nechovaly nijak nepřiměřeně a ve své podstatě fungovaly správně. Některé problémy se naskytly při snaze zprovoznit oba applety současně. Dále některým zadaným APDU applet neporozuměl. Šlo ale především o testování a vyzkoušení si schopností prostředí. Pomocí konzoly jsme mohli zadávat hodnotu čerpání z účtu fiktivní peněženky, na druhou stranu přístup k operacím potřeboval zadání PINu. Co se týče možnosti ovlivnění dvou odlišných aplikací, platforma nedokázala při určité obměně kódu applet znovu spustit. Nebo spíše chybějící hlubší znalosti principů v Java Card a nedostatek zkušenosti v programování, nedovolil širšího využití ke zkoumání bezpečnosti praktickými pokusy. Celkové výpisy APDU paketů pro různé operace v daném prostředí, zdrojové kódy, interakce konzole a seznam dostupných APDU včetně dalších obrázků je obsažen na vloženém médiu. Prostředí JCS 3.0 patří zcela jistě ke komplexněji založenému tvůrčímu rozhraní vhodnému pro pokročilejší vývojáře programů. Důležitým krokem bylo obohatit se především o základní poznatky ve vyvíjení appletů pro karty. Dnešní využití a další budoucnost čipových karet s platformou Java Card uvedeme v závěrečné kapitole 5.7 a ve shrnutí práce.

5 ZABEZPEČENÍ A POTENCIÁLNÍ ÚTOKY

Kromě možností vytváření aplikací, popisujeme v diplomové práci bezpečnostní prvky celé platformy. Zmíníme tedy oddělovací mechanismy, ale i povolené způsoby sdílení dat napříč applety. Se zajištěním integrity souvisí kryptografické nástroje a dodržení pravidel výměny dat mezi koncovými prvky. V závěrečné fázi popíšeme některé dnešní využití čipových karet u nás, případně v celosvětovém měřítku. Stanovíme předpoklady pro další vývoj technologie Java Card.

5.1 Objektový firewall

Platforma Java Card vždy byla a bude multiaplikačním prostředím. Účelové applety od různých tvůrců společně fungují na jedné kartě, s možností nahrávání dalších i po výrobním procesu. Skripty v nich mohou obsahovat velmi citlivé informace – otisky prstů, elektronické peníze, privátní klíče šifrování. Proto musí být sdílení dat mezi aplikacemi nutně omezeno.

Toho je docíleno použitím mechanismu firewallu [1], ochraňujícího applet od přístupu jeho objektů do objektů jiných a naopak. Pro spolupráci aplikací (peněženka, řízení přístupu, identifikace) na jedné kartě jsou v Java Card zavedena bezpečnostní pravidla pro sdílení dat.

Krom oddělení appletů, poskytuje firewall ochranu proti nejvíce očekávaným bezpečnostním problémům. Chyba ve výrobním procesu a předimenzovaný design může zapříčinit nežádoucí přetékání důležitých dat do oblastí určených pro jiné využití. Vykazuje také určitou ochranu proti nabourání se vetřelcem. Applet může získat odkaz na objekt ve veřejně dostupném prostředí, ale pokud ten patří pod applet z jiného balíčku, firewall zabrání přístup k němu. Nemůže tedy dojít k poškození a změně průběhu operací aplikace jinou, pokud nevyužijeme několika málo způsobů ke sdílení.

5.1.1 Kontext

Firewall rozdělí systém objektů na oddělené chráněné objektové prostory – tzv. kontexty [4] a tvoří tak hranici mezi nimi. Pokud je instance nějakého appletu vytvořena, řídící JCRE ji přiřadí objektový prostor ve formě kontextu. Instance jednoho společného balíčku Java Card sdílejí jeden skupinový kontext. Přístup k objektům je povolen instancím jen v tomto objektovém prostoru, do jiných nemohou proniknout díky bariéře.

Samotné jádro Java Card Runtime Environment má vlastní objektový prostor. Jde o systémový kontext se speciálními možnostmi. Může mít přístup do všech podřazených appletů, ovšem ty do JCRE systému vidět nemohou.

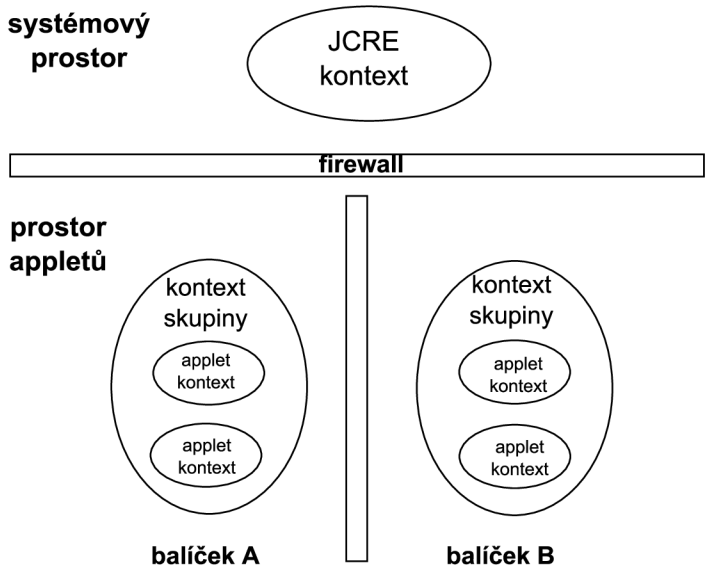
5.2 Vzájemná ovlivnění appletů

Některé způsoby datové výměny mezi jednotlivými aplikacemi jsou povoleny. Sdílení krom přímé definice ve zdrojovém kódu samozřejmě souvisí s komunikačními prostředky; především s APDU pakety. S těmi se setkáváme v metodě `process`, potřebné ve všech appletech psaných v Javě. Už zmíněný buffer, chcete-li zásobník dat, slouží k uložení APDU hlavičky příkazu. Applet si zavolá sám o referenci na APDU buffer (viz kap. 2.4.3 nebo [1]).

JCRE potřebuje zajistit, že odkazy na APDU objekt nesmí být sdíleny v proměnných třídy, instančních proměnných a polích. Applet může jen sdílet odkazy v rámci lokálních proměnných, což omezí možnost použití pouze v rámci daných metod. Toto všechno je kvůli zajištění proti zisku reference na data jiného appletu.

5.2.1 Vlastnictví objektů

Vždy existuje jediný aktivní kontext (viz výše nebo [1],[4]), ať už do něj patří applety jedné skupiny nebo jde-li o kontext JCRE. Pokud je vytvořen objekt, je vlastněn aktivním appletem v kontextu, ve kterém je iniciován. V případě statických polí vytvořených před nahráním na kartu, je vlastníkem kontext balíčku (odpovídající stejnému kontextu appletu).



Obr. 5.1: Systém oddělení appletů firewallem [1]

Objekt je základním stavebním kamenem programování v jazyku Java. Volání objektů podléhá přístupovým pravidlům. Např. privátní instance nelze použít mimo vlastní objekt. Kontext vlastnící objekt je porovnáván s aktivním kontextem. Pokud nejde o ten samý, je přístup odmítnut některým z definovaných chybových hlášení v závislosti na situaci. Jen instance třídy, tj. objekty, jsou vlastněny kontexty. Samotné třídy ovšem ne. Pouze statické metody typu `throw.It` ve třídě `ISOException` sdílí všichni. Pokud mají přídomek `private`, pak jsou dostupné jen z dané třídy.

Při volání statické metody se vše vykoná v rámci volajícího kontextu. Statická pole jsou dostupná ve všech kontextech, ale platí, že v polích uvnitř volaných objektů patří appletu, který ho vytvořil.

5.2.2 Sdílení objektu mezi kontexty

S podvědomím, že vždy je aktivní jen jeden applet, prostředky sdílení v případě potřeby, zapínají přepínání kontextů. JCVM rozhoduje, které požadavky autorizovat, a které ne. Jednoduchá žádost o přístup do objektu jiného kontextu stejně nespustí jejich přepínání. Pouze JCRE kontext může mít přístup k polím instance v jiném kontextu. Přepínání je aktivní jen při přechodu z a pak do metody objektu nacházejícím se v jiné části. Postup je takový, že aktuální kontext je uložen ve svém stavu; dojde k přepnutí a aktivizaci kontextu volaného s jejími právy. Po provedení metody tohoto kontextu, se původní zase obnoví a stane se hlavním.

V rámci JCRE systémového kontextu jsou obsažena speciální práva. Může libovolně volat metody objektů na celé kartě. Toto exekuční prostředí řídí zdroje a spravuje applety. Právě přes toto místo je na základě APDU volána metoda `select` a `deselect` pro aktivaci appletu. Dochází k přepnutí a ztrátě privilegií JCRE kontextu.

5.2.3 JCRE přístupové body

Běžný kontext appletu nemůže přistupovat ke kontextu řídícího prostředí JCRE. K některým částem lze ovšem povolit přístup přes vstupní přístupové body EPO. Jde o speciální objekty s metodami určenými k přístupu. Jiný způsob komunikace s JCRE firewall zachytí. Tedy veřejné metody JCRE objektů můžeme volat z různých míst. Jejich pole a data už přístupná nejsou. Přechodné přístupové body jsou dostupné všude, ale reference na tyto objekty nelze sdílet v proměnných tříd apod. Mezi hlavní objekty z přístupových bodů patří APDU objekt. Trvalé přístupové body mají možnost být sdíleny všude a patří sem AID instance sloužící k zabalení identifikačního čísla appletu. Za definici těchto schopností odpovídají návrháři. Díky nim může k určitým službám JCRE přistoupit v omezeném každý applet – ovšem jde o přístup k zabaleným složkám.

5.2.4 Globální pole

Data pro globální použití umožňuje firewall zahrnout do primitivních polí. Jde o sdílenou paměť přístupnou všem. Existuje zde jen jedna metoda odvozená z třídy `Object`. Možnosti může přidělovat jen JCRE. Vyvoláním metody dochází k řízenému přepnutí kontextu. Platí zde podobné podmínky jak v dočasných přístupových bodech. V rámci Java Card prostředí jde o globální pole APDU buffer a parametry v metodě `install`. Jak víme, globální pole je viditelné všude, proto je používáme omezeně, vždy při selekci nového appletu, nebo při příjmu nového příkazu v APDU. Zabrání se zpětnému zjištění citlivých dat jinou stranou

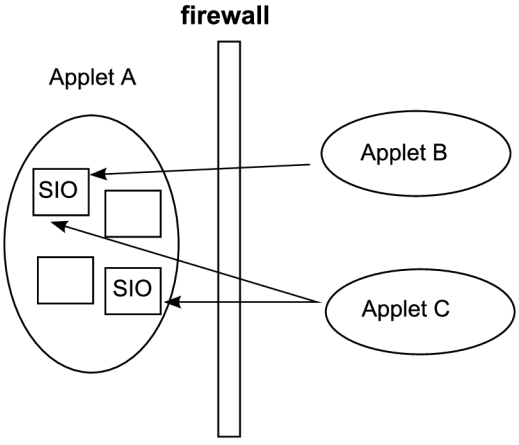
Toto vše platí pro sdílení dat appletu a JCRE. V následujících řádcích je nutné zmínit přístup dat mezi nahranými aplikacemi.

5.2.5 Sdílené rozhraní

Mechanizmy sdíleného rozhraní zabezpečují vzájemné propojení a sdílení mezi samotnými applety. Rozhraní při definici dědí rozhraní balíčku `javacard.framework.Shareable`:

```
Public interface Shareable{}
```

Sdílející rozhraní definuje metody dostupné v ostatních appletech. Třída může implementovat různé množství těchto rozhraní a také dědit další třídy. Pro doplnění, objekt třídy implementující rozhraní sdílení neboli SIO, je pro vlastní kontext jako každý obyčejný objekt. Pro okolní kontexty se jeví jako speciální objekt, který zpřístupní jen metody definované v bloku sdíleného rozhraní. Této metody sdílení dat se kupříkladu použije, když má uživatel na kartě aplikovánu peněženku a sbírá bonusové body za nakupování v nějakém e-shopu. Na základě placené částky musí být informována aplikace bonusových bodů a zvýšit svůj počet.



Obr 5.2: Přístup ke sdílenému rozhraní [1]

5.3 Útoky a obranné mechanizmy Java Card

Zajímavým poznatkem vzhledem k zabezpečení je rozdíl mezi právě standardní Java aplikací a aplikací JC. V rámci Smart Card až na výjimky není k dispozici žádný verifikátor kódu běžící na kartě. To otevírá možnost útoku pomocí škodlivého kódu. Tento přístup zkouší omezit jisté prostředky Java Card, např. označování částí kódu. Nyní si tedy popíšeme některé možnosti útoku [10] lstivým způsobem ze třetí strany, včetně známých prostředků k ochraně. Známé jsou i fyzické útoky tzv. postranními kanály, kdy se měřením zjišťují změny fyzikálních veličin daného zařízení pomocí různých metod.

5.3.1 Přehled

V jazyku Java není takové nebezpečí napadení paměťového prostoru jako u C jazyků. Určitou jistotu nám dává fakt, že známe chování části kódu, aniž bychom znali kompletní program. Na JC čipových kartách se sporadicky vyskytuje kontrolor bytového kódu. To přiživuje možnost použití záludného kódu sloužícího k poškození a získání informací z aplikace na kartě.

Platforma JC z části poskytuje ochranu proti těmto způsobům, ověřováním digitálních podpisů appletů. Ovšem vyvstává otázka, jak moc odolné jsou proti cizímu škodlivému kódu. Musíme se zamyslet, zda je možné ohodnotit kód z hlediska bezpečnosti, i bez znalosti ostatních appletů na kartě. Víme, že obranné mechanismy na kartě nemusí být dobře pochopitelné z hlediska funkčnosti. Nachází se zde firewall jako důležitý obranný blok. Zastoupí málo používaný tzv. byte-code verifier (BCV) pro Smart card? Proti jakým druhům útoku postranními kanály garantuje výrobce ochranu?

V následujících řádcích se zmíníme o zranitelnosti Java Card platformy proti škodlivým kódům. Budeme se snažit určit nějaké způsoby vsazení těchto kódů na kartu. Též jak může jeden applet poškodit druhý a protiopatření proti tomu všemu. Není mnoho dokumentů zabývajících se právě konkrétní bezpečností a útoky. Před nějakou dobou bylo provedeno pár laboratorních pokusů útoku na vybrané karty [10].

Popíšeme tedy ještě jednou ve zkratce firewall a ověřování kódu pomocí BCV. Prvním krokem pro útok je dostat falešný kód na kartu. Potřebujeme nějak prolomit systém, například změnou paměti typovou obměnou. Dále je možné diskutovat o způsobech vytvoření typového zmatení a dalších postupech útoků na karty. Proti zvoleným útokům zmíníme protiopatření v rámci karet.

5.3.2 Obranné možnosti

Verifikátor bytového kódu

Typovou správnost kódu v Java zajistí BCV a stejně tak i ochranu paměti. Bez podpory dynamických tříd, je kontrola kódu provedena při instalaci appletu. Některé karty spoléhají na ověření digitálního podpisu autorizovanou třetí stranou, a to mimo kartu. Některé kontroly probíhají za běhu aplikace. Appletový firewall potřebuje také fungovat při běhu. Obranný virtuální stroj provádí typovou kontrolu vykonané bytové instrukce. Dochází k režii v čase i v používání paměťového prostoru. To může zajistit naprostou ochranu proti škodlivým kódům.

Prolomení firewallu

Je třeba tento ochranný prostředek opět zmínit. Provádí kontrolu přístupu appletů k jiným appletům. Každý objekt má svůj kontext, a pro jakékoli pole či metodu je kontrolováno právo přístupu. Pouze řídící prostředí JCRE má právo přístupu ke všemu.

Obranné mechanizmy všeobecně mohou buď každý podporovat různé záruky a doplňovat se, nebo všechny stejné záruky a zastupovat jeden druhého. Firewall nabízí něco navíc, než ověřování kódu. Rozumně napsaný applet může zpřístupnit některá data zvnějšku metodou public. Ověřováním kódů nezabrání přístupu appletů navzájem, ale firewall všem správně napsaným nabízí kontrolovaný přístup mezi sebou (např. SIO sdílené objekty).

Firewall nezajistí ochranu proti skriptům se škodlivým kódem. V několika málo případech ovšem poskytne jakoby druhou vlnu ochrany. Pokud se něco pokouší zmocnit odkazu na objekt patřící cizímu appletu, prochází to opět obrannými prostředky firewallu, omezujícími tento čin.

Pravidla jeho fungování jsou komplikovaná. Může se objevit několik chybných nastavení, které sníží účinnost. Díky výzkumníkům na Radboud univerzitě v Holandsku [10], se podařilo prozkoumat pár odchylek od uváděných specifikací některých typů karet. Může tak docházet k neomezení přístupu přes sdílené rozhraní. I když by právě v případě aktivního appletu na jiném logickém kanále, měl být přístup omezen.

5.3.3 Škodlivý kód na kartách

Je známo několik způsobů nahrání škodlivých kódů na čipové karty. Užitím zmanipulovaného CAP souboru, napadeným objektem sdíleného rozhraní či změněním nahrávacího mechanizmu.

Manipulace s CAP

Je to nejjednodušší cesta k vnesení kódu na kartu. Bezproblémové by to mělo být v případě absence ověřování kódu BCV na kartě. CAP (Converted APplet) soubor se vytváří

po kompilaci zdrojových kódů na *.class soubory. Soubory CAP (viz kapitola 4) jsou poté nahrávány na kartu přes terminál. Konvertované applety a jejich modifikace zajistí např. zmatení mezi odkazy a skutečnými hodnotami primitivních typů. Pouze pevné znemožnění nahrávání appletů na kartu, případně kontrola jejich nahrávání s digitálně podepsaným kódem zajistí ochranu proti manipulaci s CAP.

Napadení SIO

Jde o další způsob útoku na konzistenci a funkci appletů. Prostředek sdíleného rozhraní povoluje komunikaci mezi applety (mezi různými kontexty). Lze takto legálně překračovat hranice firewallu. K vytvoření útoku necháme dva applety komunikovat přes rozhraní. Každý z nich má ale svoji vlastní CAP, ve které se definice sdíleného rozhraní liší. To protože applety jsou separátně kompilovány. Možnou obranou některých karet je znemožnění nahrání skriptu používající definici sdíleného rozhraní. BCV ověřování kódu běžící vždy pro jednotlivý applet nepozná, že je něco v nepořádku.

Napadení transakce výměny dat

Jde o možná nejlépe napadnutelnou část platformy JC. Může být zdrojem chybových operací na kartě. Transakční mechanismus dovolí použít bajtové instrukce k návratové operaci při nečekanému výpadku napájení karty nebo restartu. Návratový mechanismus by měl zrušit odkaz na objekty alokované při průběhu zrušené operace a vynulovat je. Toto může být cílem úmyslného útoku na primitivní typy. Pokud jsou odkazy na pole a proměnné roztroušeny všude možně, může být pro návratovou operaci složité určit všechny odkazy, které mají být vynulovány. Obranou proti tomuto útoku je implementace správně řešeného procesu vyčištění v případě chtěného či nevynuceného přerušení transakce. Případně je dobré mít ošetřenou schopnost vypnutí karty při přerušení nějaké aplikace.

5.3.4 Typové útoky v JC

Pomocí zmíněných metod je uskutečnitelné nahrát trochu jiný typ nakaženého kódu. Určitým nápadem je využít typové zmatení (prolínání [10],[11]) mezi poli z různých typů. Byte a Short záměna pole se použije pro přístup k hodnotám za hranicemi pole. Také připadá v úvahu využití zmatení typů mezi polem a objektem. Například zapsáním odkazu na objekt k provedení výpočtu ukazatele. Záleží na reprezentaci v paměti.

Bytové pole využité jako pole Short

Jde o snahu vynutit si u appletu dvojité čtení nebo zápis o dvojnásobné velikosti původního bytového pole. Díky drobným úpravám kódu se považuje pole typu byte jako pole typu short. Kupříkladu pole byte délky dvacet čtené jako pole short dovolí přístup 20x k poli short, tj. 40 bajtů, jehož polovina reprezentace může patřit už jinému appletu v paměti.

Zvažujeme 3 typy polí byte: globální (APDU buffer), pevné a dále i přechodné kontextové pole. Na většině karet se tato pole chovají stejně a mají stejnou odezvu při této snaze obelstít ochranu. Slabě zabezpečené typy karet umožní nahlédnout do částí souborů CAP instalovaných vzápětí na kartu. Applet první může ovlivnit data appletu instalovaného později. Karty s běžící typovým ověřováním neumožnili tento útok, dle [10] dokumentu na straně 9.

Přistup k objektu jako k poli

Dochází ke zmatení mezi polem a objektem, dle příkladu:

```
Public class Vzor { short pole = (short) 0x7FFF; }
```

Celkový útok závisí na skutečné reprezentaci polí a objektů v paměti. K úspěchu musí být informace o délce pole sdílená ve stejném rozmezí fyzické paměti jako parametr "pole" objektu Vzor. Když dokážeme donutit JCVM přistupovat k objektu Vzor jako k poli, délka pole bude 0x7FFF – což poskytuje přístup do pamětové části o velikosti 32 Kb. Když můžeme přistupovat k informační hodnotě pole prostřednictvím reference na objekt, můžeme již délku pole volit libovolně.

Z toho plyne, že můžeme následně odkazované pole číst a zapisovat jako číselné hodnoty. Pokud máme přístup k referencím na objekty, můžeme je měnit za jinou referenci jiného typu. Vše je jednoduché, dokud se nesnažíme překročit hranice objektů. Na některých kartách dochází k dynamické kontrole hranic objektu. Zabraňuje se zde přepínání referencí pryč z povolených oblastí přístupu. A takto se brání přístupu do jiných dat appletu.

Nabízelo by se nastavení pole v jednom appletu k odkazování na objekt z jiného, to ovšem ve většině případů omezuje firewall karty.

AID zneužití

Možnost přepínání referencí může být zneužito k úpravě systémového identifikačního čísla objektu – AID. Applet Identifier je sekvence čísel identifikující applet na kartě. AID objekt vlastní své pole, ukazující na bajtové pole s informací o aktuálním identifikátoru.

Problémem může být firewall. Jinak změna AID appletu vlastněna systémem způsobí, že poškozený applet naruší formu registru. Navíc omezí funkci jiných spoluaplikací.

5.3.5 Protiopatření

Java Card a její specifikace nechává trochu volnosti pro obranné mechanizmy zasazené ve virtuálním stroji (JCVM). Jakmile je spuštěn správně napsaný kód, tyto mechanizmy nemusí být odhalitelné. Pravidla JC při exekuci správného kódu na různých kartách garantují, že výsledky budou stejné. V rámci podvrženého kódu nic takového zajištěno není. Zavádí se tedy další dynamické obranné prostředky [11][15].

Ověřování typů – tento proces v době průběhu aplikace na některých kartách běží a činí je odolnými vůči záměrně škodlivému kódu změnou typu.

Ověření hranic působnosti objektu – virtuální stroj provádí kontrolu omezených hranic působnosti pole při přístupu k jeho prvkům. Podobný proces může být požadován při přístupu pole instance a volání metod objektu. Při konverzi bajtového kódu na CAP soubor, jsou jmenní reprezentanti nahrazeni číselnými pro lepší kontrolu délky polí.

Fyzická kontrola hranic pole – kontrola může být provedena jak použitím logické velikosti pole, tak i díky fyzické velikosti složek pole v bajtech.

Kontrola firewallem – pracuje bezpodmínečně za běhu. Některé útoky změnou pole byte na pole short a záměnou AID ukazuje na ne zcela postačující obranu. Útoky za účelem záměny referencí jsou neškodné, pokud firewall dokáže zabránit přístupu do jiných kontextů. Způsobem přístupu k objektu jakoby šlo o pole, můžeme v mnoha případech firewall obejít.

Paměťová integrita – K aplikování dynamických kontrol hranic pole potřebuje firewall z virtuálního stroje nahrát metadata do paměti představující objekt. Při kontrole dodatečných dat může JCVM omezit další možné útoky.

Všeobecně schopnost nahrání zákeřného kódu na kartu ještě nezaručí úspěch. Kontrola skriptů samozřejmě není jediným řešením. Transakční mechanizmy mohou podkopat bezpečnost při ověřování kódu. Řešením je lepší kontrola za běhu, než pouze před nahráváním. Omezuje to ovšem výkon JCVM.

Tyto problémy s náchylnosti k poškození platí u karet s možností nahrávání dodatečných appletů. V případě využívání karet v běžném životě, by povýrobní nahrávání appletů mělo být zakázáno, nebo alespoň podmíněno digitálními podpisy a autoritami. Bylo by dobré mít jistotu, že neexistují žádné možnosti ovlivnění appletů navzájem ve špatném slova smyslu. Nebezpečí zde ale stále trvá.

Na okraj zmíníme využívaný venkovní útok měřením chování karty. Postranními kanály lze pozorovat a měřit fyzikální změny zařízení v průběhu výpočtu nějaké operace. Jedná se o elektromagnetické vlivy, změnu odběru napětí a teplotu. Jedná se o neinvazivní metody bez nutnosti poškození čipu karty. Proti zjištění citlivých infromací analýzou operací výpočtu šifrovacíh algoritmů, klíčů nebo PIN kódu, je třeba se spolehnout na ochranné mechanizmy výrobce.

5.4 Kryptografie a ochranná pravidla

Poukazujeme zde na typické schopnosti a vlastnosti u čipových karet a jejich aplikace pro různá zabezpečení [2]. Bezpečná výměna zpráv, použitý transportní protokol, to vše může být založeno na tajném šifrování a klíčích. Nejprve je nutno zmínit základní požadavky – aby byla zajištěna samotná aplikace i výměna dat mezi komunikujícími stranami, tedy kartou a uživatelem. Naopak použití karty jako třeba autorizačního tokenu zajišťuje bezpečnost pro různé systémy v běžném životě.

5.4.1 Aplikační zabezpečení

V případě sdílení finanční částky ve formě elektronických peněz, je prvním bodem zájmu zajištění autentizace. Obě strany komunikace musí poznat, že druhá strana a její data jsou legitimní. Zajištěním prostředků z kryptografie budou obě strany autorizovány. Data mohou obsahovat citlivé informace o PINu, částce či bankovním účtu. Pro znemožnění odposlechu třetí stranou se data šifrují. Dále je proti změně integrity dat během transportu do APDU příkazů přidáváno za uživatelská data kontrolní číslo MAC. To se vypočítá většinou z již zašifrované zprávy.

Čipové karty různých velikostí jsou z velké části považovány za autorizační prvek a prostředek pro šifrování v rámci uživatelského a průmyslového využití v širokém spektru oborů (viz kap. 1). Například určitá aplikace může autentizací držet tajný klíč pro přístup do sítě. Držitel karty pak zažádá o přístup (do sítě, případně nějakého prostoru), spravující server nebo terminál vygeneruje kupříkladu náhodné číslo a pošle k uživateli. V rámci tokenu se data zašifrují a pošlou zpět do terminálu nebo serveru. A ten, pokud je dešifruje a najde shodu se svou již vygenerovanou hodnotou, zobrazí informaci, že autorizace proběhla v pořádku.

5.4.2 Obecné požadavky transakcí

Důležitým pravidlem jsou implementační jedinečnost, algoritmická nezávislost a přenositelnost. Cílem je nechat uživatele si upravit kryptovací služby, tedy šifry,

a digitální podpisy bez znalosti algoritmu pro ně. Můžeme i do API rozhraní Java Card dodávat i brát různé šifrovací algoritmy. V API jde z velké části o abstraktní třídy. Je zde definován typ služeb šifrování a přístupné rozhraní k těmto službám. Například JCRE dokáže dědit třídu v API z balíčku `javacard.security` a nabídne implementaci doslova v pár řádcích. Jsou k dispozici jak symetrické, tak asymetrické šifry (viz kap. 5.4.2).

Tato část nabízí racionální způsob zabezpečení v rámci peněženky, která by se dala použít v praxi a též karty pro kontrolu přístupu. Popisují se jednotlivé položky – cíle, chování a mechanizmy, i způsob provedení [14,15].

Cíle zabezpečení

- Žádná možnost neoprávněného zvýšení částky.
- Nemožnost náhodného vymazání části peněž.
- Pouze majitel bankovního účtu z něj může nabít částku do peněženky.

Bezpečnostní politika

Pro cíle pro oblast zabezpečení je určena bezpečnostní politika.

1) Zabránění nepovolenému přidání peněz:
- Terminál se čtečkou i karta musí sebe navzájem autentizovat. Není možné, aby provedené autentizační mechanizmy byly zpětně detekovatelné.
- Příkaz pro dobíjení musí být autorizován terminálem.
- Při nabíjení peněženky penězi, musí být odečtena z účtu odpovídající hodnota.

2) Ztížení poškození nebo vymazání částky v peněžence:
- Čerpání částky je umožněno, pokud to velikost zbývajících financí dovolí.
- Dochází k dotázání se bankovního účtu o povolení provedení transakce.
- Možná ztráta při přerušení operace.

3) Autorizace uživatele u peněženky i bankovního účtu:
- Zadání PIN je vyžadováno. Je omezen počet neúspěšných zadání.
- Použití bezpečnostních certifikátů po zadání pin kódu.

Dalším omezením neoprávněného použití může být limitování týdenní hranice pro čerpání. Jednoduše existují stropy pro maximální výběr, dobití, částku v transakci i obsah identifikačních PINů.

Mechanizmy autentizace

Komunikace probíhá v zabezpečených kanálech. Jsou použity další mechanizmy:

- Kritické příkazy jsou provedeny po oboustranné autentizaci, založené na sdílených tajných klíčích a na čítačích. Různé klíče pro různé úrovně zabezpečení operace.
- Podpisy a klíče zabezpečí nejen původ příkazů, ale i integritu a správné pořadí. Proti zpětnému dohledání podpisových zpráv se používá:
  - Možné digitální podpisy zakládající se na sdílených klíčích. Ty fungují jen určitou dobu od otevření do uzavření kanálu.
  - Podepsání zprávy záleží na předchozí hodnotě v podpisu zprávy. Zajištěna souvislost a pořadí.
  - Dočasné klíče zajistí i ochranu proti kryptoanalýze, zjišťující hodnoty z klíčů zpětně z historie zpráv.
- Tajné informace včetně PINu jsou navíc posílány zašifrované.
- Šifrování příkazu i odpovědi.

Pravidla pro přístupové applety na kartách

Pro všechny applety a aplikace na kartách pod platformou Java Card platí i předešlá pravidla a ještě několik dalších. Takový vlastník karty s funkcí přístupu do budov, firemních areálů, dveří, nebo také k přístupu k počítači, připojení k internetu se musí o svoji kartu pečlivě starat a chránit ji před odcizením a zneužitím.

Všeobecně elektromechanické zařízení přebírá roli klíčové dírky a karta působí jako druh elektronického klíče. Každý má přístup tam, kam mu byla nastavena práva a musí znát svůj PIN.

Většinou dochází k přiložení karty k terminálu a zadání soukromého kódu. Komunikace by měla probíhat také v zabezpečených kanálech (viz výše). Ve vývojovém prostředí JCS Suite na PC se simuloval přístup zadáním PINu do konzoly programu za běhu appletu na kartě (kapitola 4.3).

5.5 Čipové karty a bezpečí Java Card

Toto spojení je dnes velice úspěšné a hojně používané vývojáři i koncovými výrobci karet. Můžeme nyní popsat vzájemné rozpoznávání terminálu a karty v něm. Pro zabezpečení proti zneužití jsou komunikace a případné bloky programů šifrovány symetrickými a asymetrickými algoritmy, které nyní popíšeme [4]. V části o vytváření a nahrávání appletů (kap. 4.3) jsou uvedena použití vzájemné výměny autorizačních klíčů uložených v bezpečnostní doméně karty. Na konci této podkapitoly se zabýváme riziky platformy.

5.5.1 Šifrování

Šifrovací algoritmy [2], jak známo, dělíme na symetrické a asymetrické. V rámci prvního se používá stejný klíč pro šifrování a dešifrování. V druhém se používají dva různé klíče. Důležitým prvkem je množství různých klíčů použitelných pro daný algoritmus.

Dalším platným faktorem je, že výpočetní čas procedury nesmí záležet na velikosti klíče, nebo šifrovaného textu – jinak je možné algoritmus rychle prolomit. Jako „prakticky bezpečné“ se označují systémy, které neprolomí útočník jakkoli omezený časem a prostředky. Po prolomení systému již není garantována spolehlivost a autentičnost. Útočník dekóduje data, nejspíše za účelem změny, pokud před tím zjistí tajný klíč patřící algoritmu.

5.5.2 Symetrické algoritmy

Jsou založeny na použití společného privátního klíče pro šifrování i dešifrování – proto symetrické. V následující části popíšeme hlavní použitelné algoritmy v Java Card.

DES algoritmus

Při vývoji tohoto algoritmu byly brány v potaz dva důležité principy. Jedná se o metody zaměňování a prolínání. Statistiky šifrovaného textu mají vliv na statistiky původního textu pro ztížení dekódování útočníkem. Dále prolínáním každého bitu klíče a bitů šifrovaného textu dokážou výsledné šifrované slovo natáhnout jak jen je to možné.

V rámci DES má vstupní blok a zašifrovaný blok textu stejnou velikost 64 bitů. Jsou zde obsaženy i paritní bity, což snižuje délku pole využitelnou pro klíč. Znamená to, že je zde přibližně 2^{56} různých klíčů. Může se to zdát hodně, ovšem limit počtu možných privátních klíčů je hlavní slabinou algoritmu DES. V případě odposlouchávání komunikace mezi terminálem a čipovou kartou, je možné nasadit útok hrubou silou, využíváním paralelních procesů a k získání vstupního textu použitím všech možných klíčů. Porovnávají se předchozí šifrované texty s aktuálním. Tento typ algoritmu se stal zastaralým a nedostatečným, proto se v pozdějších letech přešlo k Triple-DES (3DES) mechanizmu (viz níže). Dvojité DES šifrování používá dva odlišné klíče, které není možné odhalit pomocí jednoduchého šifrování použitím třetího klíče. Speciální 3DES metoda dovoluje použít tři sekvenční CBC-DES módy ke třem odlišným metodám šifrování, potažmo dešifrování. Zašifrovaný blok použitím sekvence operací bude dešifrován v opačném sledu těchto zmíněných operací. Pokud jsou použity právě tři různé klíče v každé operaci, jde o Triple-DES algoritmy. Tři nezávislé klíče o velikosti 3x56 bitů.

AES šifrování

Jde o blokový symetrický mechanizmus s bloky o délce 128 bitů, využitelných pro tři různé klíče. Z tohoto důvodu existují AES-128, 192, 256 v závislosti na délce klíče. Proto je prostor $4.7x10^{21}$ pro AES klíč bezkonkurenčně větší než u DES. V rámci softwarové implementace na kartě zabírá 4 kB paměti ROM.

Existují 4 formy použití blokových symetrických šifer. Z toho převážně jedna, CBC (cipher block chaining) [4] je běžně používána v technologii Smart Card. V tomto módu jsou datové řetězce v blocích spojovány pomocí XOR operátoru v průběhu šifrování. Každý krok je tedy závislý na předchozím. Podobné bloky vstupního textu jsou převedeny do absolutně odlišných šifrovaných bloků. Inicializační vektor a první blok vstupního textu jsou smíchány v operátoru XOR a následně šifrovány DES algoritmem. Tento šifrovaný blok je poslán do operátoru XOR společně s dalším blokem vstupního textu atd.

5.5.3 Asymetrické šifrování

Jde o využití dvou různých klíčů. Veřejným klíčem může osoba zašifrovat zprávu, ale pouze oprávněná osoba vlastnící soukromý klíč ji může dešifrovat či obráceně. Využívá se u digitálních podpisů.

Šifra RSA

Nejrozšířenější asymetrický algoritmus, který je založen na aritmetice velkých čísel. Před samotným kódováním je třeba vstupní texty rozdělit na bloky dle velikosti klíče RSA (Rolland Shamir Adleman). Samotné šifrování je zavedeno umocněním vstupního textu následovaným modulovou operací. Výsledek je šifrovaný text. Mezi dešifrováním a šifrováním existuje analogie. Je nutné znát soukromý klíč. Z důvodu nevelké paměti je na čipových kartách aplikována metoda modulového umocnění, kde průběžné výsledky nikdy nepřesáhnou hodnotu modulu. Nejvíce se používají klíče délek 768 a 1024 bitů.

V posledních letech jsou klíče z 2048 bitů. S délkou klíče stoupá výpočetní náročnost. Samotné CPU jednotky, osazené na čipech karet, by nezvládali výpočet RSA. Aritmetický koprocesor vyřeší tento problém, protože je určen pro rozsáhlé matematické výpočty.

DSS algoritmus

Norma DSS byla ustanovena ve Spojených státech a popisuje systém digitálních podpisů DSA (Digital Signal Algorithm). Zabývá se spíše vytvořením elektronického podpisu, než samotným šifrováním, jako je tomu u RSA. Podpisová zpráva musí být zmenšena do dané

velikosti použitím hashovací funkce (hash algorithm). Bylo doporučeno použít transformační funkci SHA-1. Ta vygeneruje 160-bitovou hodnotu z nějaké zprávy.

5.5.4 Identifikování komunikace

Základem je možnost komunikovat oběma směry při výměně dat terminálu a čipové karty. K dispozici je pouze jeden kanál. Komunikace se nazývá half-duplex, kdy vysílá data jedna nebo druhá strana a řídí se pravidly. Základy jsme se zabývali v kapitole 2.4.2. Karty s full-duplex režimem se zatím dostatečně nerozšířily. Inicializaci komunikace provádí vždy terminál. Karta odpovídá na jeho dotazy, sama bez výzvy data neposílá. Jde o jakýsi typ master – slave operace. Po vložení karty do čtecího přístroje (obecně terminálu) se mechanicky i elektricky propojí kontakty čipu. Karta provede restart napájení a odešle zprávu Answer To Reset (ATR) k terminálu. Ta obsahuje inicializační parametry karty a druh komunikačního protokolu (Protocol Parameter Selection - PPS). Po schválení vyšle terminál první příkaz APDU. Následně karta toto zpracuje a odešle odpověď zpět atd. [14]

Obecně po připojení napájení, nastavení časovače a resetu signálu přijde na řadu vyslání ATR zprávy přes komunikační kanál. Obsahuje dílčí informace o transportním protokolu, kartě a používané aplikaci. Není běžné, aby zpráva měla svojí maximální velikost 33 bytů. V rámci platebních systémů je kladen důraz na rychlost, proto je ATR zpráva zkrácena, aby se dosáhlo menšího zpoždění. Pokud v rámci milisekund terminál nedetekuje zprávu, snaží se opakovat aktivační sekvenci, popřípadě označí kartu za nefunkční. V rámci ATR je důležitý indikátor kategorie. Nachází se zde informace o struktuře ATR. Jde o seznam podporovaných služeb karty a OS. Pak se provádí výpočty založené na zdrojovém kódu. Metody [4] pro připomenutí jsou následující.

Výpis kódu. 5.1: Applet a jeho základní metody

```
Import javacard.framework.*;
Public class Vzor extends Applet {
Public static void install (byte[] bArray, short bOffset, byte bLength)
//Tuto statickou metodu volá JCRE pro utvoření instance třídy Applet

protected final void register()
//toto se používá k registraci do systému a vsazení AID čísla do souboru cap
//kompilovaného appletu

protected final void register(byte[] bArray, short bOffset, byte bLength)
// toto se používá k registraci spuštěné instance appletu v JCRE a přiřadit
//appletu identifikaci AID v poli bArray

public boolean select()
//vývojové prostředí volá metodu pro oznámení applet, že byl vybrán pro
//komunikaci

public abstract void process(APDU apdu)
//Přikazuje appletu zpracovat příchozí příkaz APDU bloku
public void deselect()
//volání metody následuje při oznámení zvolení jiného appletu }
```

5.5.5 Logické kanály a AID identifikátor

V jednotlivých čipových kartách je většinou obsaženo více nezávislých multiplikačních programů, můžeme s nimi komunikovat pomocí logických kanálů pro každou z nich. Až pro čtyři možné aplikace je zde možnost komunikovat souběžně přes vlastní logické kanály. Dva bity v APDU bloku se používají k určení, který příkaz patří jaké aplikaci. Z důvodu absence identifikace kanálů nelze odeslat další a další příkaz, dokud odpověď na ten poslední nedorazí k terminálu. Dalo by se říci, že každý logický kanál, obrazně řečeno, představuje samostatnou kartu se všemi jejími prostředky. Operační systém v případě komunikace s více aplikacemi musí toto paměťově zvládat souběžně. Zahrnutím všech možných procesů autentizace tak zvýší požadavky na paměť RAM mikročipu karty.

Identifikátor aplikace (AID) je jedinečné číslo příslušné jednotlivým aplikacím od výrobců karet dle normy ISO/IEC 7816-5. Číslo AID může být národně, případně mezinárodně registrováno.

Toto značení dělíme na dvě složky. První datový blok je číslo Registrovaného aplikačního identifikátoru (RID) o délce 5 bytů. Obsahuje kód země, kategorii a číslo poskytovatele aplikace. Tento numerický kód zajistí jedinečnost RID a identifikaci vybrané aplikace celosvětově. Druhý blok může dodatečně dodat poskytovatel služby. Rozšíření identifikátoru o délce asi 11 bytů (zkráceně PIX) obsahuje např. sériové licenční číslo pro správu aplikace. Jde tedy hlavně o identifikaci používanou renomovanými výrobci karet a programů pro ně (příklad v Tab. 5.1) [4].

Tab. 5.1: Vzor bloku čísla identifikátoru AID

RID	**PIX**	**Popis**
A0 00 00 00 63	50 4B 43 53 2D 31 35	ID karta ve Finsku

5.6 Výhody a nevýhody Java Card

Mnoho ze základních pravidel bezpečnosti Java není zahrnuto do technologie Java Card (JC). Tedy celý koncept ochrany JC appletů je odlišný od běžných appletů např. pro webové stránky. Paradoxně některé chybějící prvky vylepšují situaci, a naopak některé vytváří nová potenciální rizika.

Snižování rizik

Je problém, že Java musí vyřešit zajištění typové ochrany, když ve stejné chvíli zpřístupňuje dynamické třídy. Pokud máme v podvědomí zmatek o manipulaci s typy objektů, můžeme tím poškodit zabezpečení programu. V běžné situaci, JC se stará o vyřešení odstraněním dynamických tříd. Všeobecně nahrávání tříd bylo vždy zdrojem bezpečnostních „děr“ v Javě.

Výhodou může být absence zřetězení v JC z důvodu lepší přehlednosti kódu. Zřetězení totiž zahlcuje JCVM. I když na kartě může být více appletů, systém dovolí v jedné chvíli pracovat právě s jedním z nich. Větší počet aplikací s možným vzájemným ovlivněním je nebezpečí už samo pro sebe.

Zvýšení rizik

Chybějící řetězení a dynamické třídy mohou mít také špatný vliv na vlastnosti JC. Můžeme zmínit několik dalších rizik:

- chybějící sběr odpadu (garbage collector)
- správa chybových hlášení
- ochrana multi-aplikací firewallem
- přístup nativního kódu a problém sdílení objektů.

Tyto **oblasti** [16] si dále přiblížíme.

Tzv. Garbage collector (sběr odpadu a dat v paměti) je dobrým příkladem vlastnosti s bezpečnostním dopadem. Bez nějakého způsobu uvolňování paměti může nastat problém s odmítáním služeb. JC specifikace nepotřebuje implementovat sběr odpadu na kartě. Výsledkem ale může být způsobení zmatku díky zvláštním chybám. Paměťové trhliny (zaplnění paměti při neuváženém vytváření a rušení objektů) se objevují hlavně v jazyce C++. Jestliže JC toto nepodporuje, logické chyby mohou vyčerpat paměť a degradovat kartu.

Dalším problémem je tzv. „volný ukazatel“. V tomto případě program vyprázdní paměť, ale stále si uchovává ukazatel na toto místo. Do této volné pozice je možné později uložit nová data, ale při použití starého odkazu může dojít k pádu aplikace. Je vidět, že tato schopnost garbage collectoru může chybět. K zajištění bezproblémového využití paměti slouží správné testování a analýza appletového kódu.

Projektování výjimek je zajímavým úkolem pro JC, neboť nezachycení výjimek vede k destabilizaci karty a omezení použití. Nevyřešením této oblasti dochází k zvláštním situacím a nevyžádaným přerušením programu.

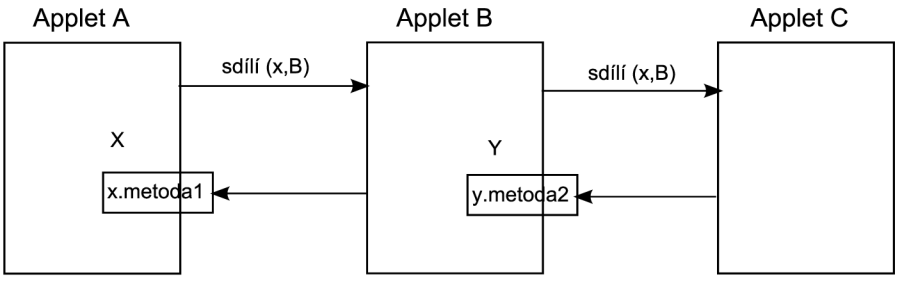
Od chvíle, kdy JC povoluje multiplikační systém na jedné kartě, existuje ohrožení útoky mezi oddělenými aplikacemi. Riziko je nasnadě v situacích, kdy applety poskytují různí výrobci. Jak už bylo několikrát zmíněno, firewally ochraňují právě applety navzájem, i když občas není jasné, jak je na konkrétní kartě tato technologie řešena.

Tudíž asi hlavním způsobem ochrany může být správa paměti, přesněji, programy nemají mít přístup do libovolné její části. Povolen je přístup k objektům, které vlastní na základě instalace appletu. Politika paměťové ochrany musí být striktně zabudována pro spolehlivé použití. Samozřejmě existují případy (a bude jich přibývat), kdy potřebují na kartě applety sdílet data mezi sebou. Budoucí využití jedné karty pro všechny možné služby denního života je otázkou několika let.

Důležitým požadavkem v rámci více aplikací a zajištění jejich bezpečnostních pravidel na jedné kartě, je právě schopnost sdílení objektů, citována na stránkách Sun Microsystems Oracle:

„JCRE udržuje cestu k právě zvolenému appletu stejně jako k právě aktivnímu appletu. Aktivní hodnota appletu je označena jako jeho prováděcí kontext. Pokud je pak volána virtuální metoda v objektu, kontext je změněn ke korespondujícímu appletu vlastnící daný objekt. Po návratu z metody se vrací ukazatel kontextu zpět k původnímu programu. Kontext a pravidla sdílení objektu tak určují, zda je přístup ke sdílení vůbec povolen.“ [13]

Uvedeme příklad s třemi applety (obr 5.3). Applet A sdílí vlastní objekty „x“ s B. B sdílí svoje „y“ s appletem C. Metoda v objektu části B pojmenovaná y.metoda2(), sdílená s C vyvolá metodu v objektu z A pod jménem x.metoda1(). Applet C je vybrán a má přístup k vyvolání metod objektů v B. To zahrnuje, že může v jednom kroku zvolit i metodu v objektu z appletu A. Jednoduše - C nepřímo volá x.metoda1() z A. Problémem je, že A si může myslet, že sdílí x.metoda1 pouze s B. Ovšem applet B může otevřít pro sdílení do C tuto metodu prostřednictvím jiné y.metoda2. Applet A vůbec nemusí vědět, že sdílí do více appletů, než původně zamýšlelo.



Obr 5.3: Nevyžádané sdílení metody mezi applety [15]

Krátce řečeno, virtuální metody, pokud jsou použity, umožní appletu přístup k objektům a zároveň se zajistí také schopnost přidělit přístup i pro jiný applet. To značně oslabuje zajištění ochrany objektu. Statické metody nemění kontext a uplatňovat omezení pro dynamické metody přináší složitosti v procesech sdílení. Může dojít k donucení appletu, aby sdílel všechny objekty jinak vyžadující udělení přístupu.

Zdaleka největším problémem v návrhu JC je schopnost prodejce použít původní nativní metody pro danou platformu. Applety nemohou být přenášeny na odlišné karty a komunikovat zároveň s jinou kartou, což také vystavuje kartu nebezpečnému kódu z vnější strany za firewallem. Pokud jsou nativní metody dostupné, zhoršuje to schopnost právě firewallu. JCRE toto neomezuje a dochází k překročení pravidel (nativní programy psané v jiném jazyce sem také spadají). Útočící applet využije nativních metod k vynesení operací, které by jinak okamžitě zastavilo prostředí JCRE. Po spuštění kódu už JCRE a bezpečnostní mechanizmy nic nezmohou. Proto se nativním metodám snažíme vyhnout.

Existuje mnoho dobrých i špatných bezpečnostních přístupů na kartách. Můžeme zvolit takovýto přístup: Vše na kartě je zabezpečené, neboť data byla nahrána bezpečným způsobem a karta je imunní proti manipulaci ilegálním způsobem. Jinak řečeno, vše v domě je bezpečné, protože to tam dal majitel a jen on má klíč. Tedy čipové karty jsou odolné, protože bezpečnostní návrh při výrobě a vývoji neumožní dodatečnou nepřiměřenou manipulaci. Pouze v rámci nahrávání dat a objektů na kartu i z ní, funguje přístupový seznam povolených akcí.

Smart Cards fungují skoro jako organizované soubory, tedy nastavením schopností pro manipulaci s daty a příslušného typu klíče, je možné zaručit lepší zabezpečení pro čtení, zápis apod. Klíč jednoho člověka umožní z karty jen číst, naopak u jiné osoby je povoleno i zapisovat do souboru. Pro přiřazování schopností jednotlivým osobám je třeba přiřadit pověřené osobě zase jiný klíč přístupu (klíče na principu např. PIN kódu). Existuje mnoho ověřovacích technik, z nichž každá funguje trochu jinak. V praxi např. čipové karty platformy OpenCard použité bankovními domy, potřebují zadání bezpečného PINu. To lze provést prostřednictvím klávesnice na čtecím zařízení - bankomatu.

5.7 Použití čipových karet v praxi

Krom všeobecně známých platebních a všech možných kreditních karet s čipem poskytovaných bankovními domy, jsou tu např. zaměstnanecké přístupové ID karty do prostor podniků. Studentské karty ISIC integrující více funkcí v jednom. Konkrétními známými příklady u nás jsou CLUB CARD pro Tesco obchodního řetězce a Open CARD pražských integrovaných služeb, která má stále své problémy. V rámci Tesco věrnostního programu sbírá zákazník body a uplatňuje je na slevách. Karta disponuje PIN soukromým číslem pro zajištění ochrany spotřebitele. V rámci druhého příkladu pražský magistrát hl. m. Prahy takto poskytuje multiplikační nástroj ke svým službám. Díky možnosti nabíjení

peněžních jednotek do čipu karty, je možné ji využívat k placení parkovného, jako náhradu průkazky pro městkou dopravu nebo jako přístupový prostředek do knihoven. Všechno výše zmíněné je prováděno za účelem integrace informačních technologií do života a k posílení ekonomiky státu. V blízké budoucnosti by mělo také dojít k masovému rozšíření čipových ID jednotlivých osob, chcete-li elektronické občanské průkazy. Mnohé státy už s touto technologií bilancují a testují je každodenním používáním.

5.8 Průřez dosaženými výsledky

Zaměřili jsme se na bezpečnostní prvky platformy Java Card. V další části shrneme a popíšeme mechanizmy uplatnění těchto prvků a krátce zmíníme využití ve vyzkoušených appletech. Zmíníme zkušenosti s vývojovým prostředí JCT Suite 3.0 vystavěné na jádře Eclipse API používající technologii Java Card. Shrneme obranné prostředky pro dnešní využití a slibnou budoucnost čipových karet na této i ostatních platformách.

Bezpečnostní prvky JC

Snoubí se zde základní bezpečnost Java systémů a dodatečných částí přidaných z JC. Podmínka inicializace proměnné před jejím použitím a kontrola úrovně přístupových práv pro třídy a metody je hlavní propozicí platforem Java. Doplněním prostředků ze světa čipových karet, které mají zvláštní požadavky, nacházíme další pilíře bezpečnosti a správného fungování platformy Java Card.

V JC jsou objekty sdílené v přechodné paměti v RAM. Při restartu karty jsou veškeré objekty stanoveny na základní hodnotu (zero, false). Data jsou přechovávána také v dočasné paměti, ale jsou ošetřena pravidly při případném výpadku napájení a pádu karty. Tedy v případě chyby, bude položka nějakého pole obnovena na předchozí uloženou hodnotu. To znamená, že všechna data v bytech, již správně nakopírována, jsou obnovena do svých předchozích hodnot. V předchozích kapitolách zmíněný firewall mezi applety funguje výborně. Odděluje paměťový prostor appletů a prostor pro systém. Applety nemohou komunikovat spolu, pouze pokud se nachází ve stejném balíčku. V oblasti sdílení, JCRE má přístup do všech appletů. Applety navzájem jen díky zvláštním přístupovým bodům a díky sdílenému rozhraní. Nativní metody pro přístup za hranice JCVM či do jiných systémů je povolena pouze v rámci zabudovaných systémů od výrobce karet.

Mechanizmy konverze

Kód Java v jedné či více *.java souborů je kompilována využitím standardních prostředků prostředí Java Card. Zmíníme třeba Oracle Java Develoment Kit nebo právě Smart Caffé. Je snahou odstranit co nejvíce chyb již při tomto procesu.

V rámci prostředí Java virtuálního stroje jsou soubory ověřeny ještě před spuštěním. Zjišťuje se správný formát a struktura. Předchází se tím výpadkům paměti a její nestabilitě. Zajišťuje se nedostupnost privátních metod z vnějších periferií mimo jejich třídu. Každý objekt je určen jen pro svoji jedinečnou funkci.

Naopak v JCVM máme dvě hlavní části. Konvertor běžící mimo kartu na hostitelské části systému. Jako vstup vyžaduje právě *.class soubory. Jsou podrobeny podobnému procesu jako v tradičním sledu ověření, ovšem s více specifickými pravidly. Pro zajištění schopností v mezích pravidel Java Card proběhne ještě proces kontroly datových typů, nežádoucí přítomnosti více vláken či přetečení. Pak se převádí do CAP souboru pro kartu.

Applety peněženky a kontroly přístupu

V použitém prostředí v laboratoři jsme si vyzkoušeli vytváření a nahrávání appletů na kartu. Vše proběhlo na počítači s daným softwarem. Vyzkoušení funkčnosti dvou hlavních způsobů proběhlo v konzolovém textovém prostředí pomocí zadávání příkazů. Toto omezení ohraničilo zaměření naší práce. Avšak skutečně šlo o určité využití jediné čipové karty Convego Join 4.0, jako multiplikačního prostředku. Možné vyzkoušení samotné karty mimo vývojové prostředí by mohlo být námětem dalšího výzkumu. Zaměřili bychom se kupříkladu na nějaký námi upravený terminál patřící např. dveřím, které bychom synchronizovali s kartou (a appletem pro tuto funkci nainstalovaným) a pokusili se díky ní dveře otevřít. Případné využití promyšleného appletu pro kontrolu peněžního obnosu na stejné kartě, by se dalo vyzkoušet v interakci s rozhraním jednoduchého čtecího zařízení s naprogramovaným nákupním systémem. Veškeré prostředky by samozřejmě skýtaly větší finanční podporu. Bylo by nutné najít vhodného výrobce terminálu, případně součástky pro výrobu i další software.

Během posledního roku nastupuje do podvědomí vývojářů nová specifikace Java Card Platform 3.0 ve dvou verzích. Verze Connected oproti Classic nabízí krom běžných funkcí a i podporu komunikace využitím protokolů HTTP přes TCP/IP (z rozhraní USB). Jde tedy o jistý druh webového serveru. Otevírají se pak nové možnosti. Platforma nové verze využívá lepší čipové karty s 32 bitovými procesory. Není potřeba převádět *.class soubory do různých formátů a je dovoleno svůj zdrojový kód přímo nahrávat. Důležitým prvkem je zavedení čištění nealokované paměti a její obnovování. Jde vlastně o inovativní kombinaci využití internetové komunikace a zavedení vícevláknových operací do prostředí Java Card. Z pohledu bezpečnosti se pro komunikaci používají nově i HTTPS a SSL protokoly.

ZÁVĚR

Náplní diplomové práci bylo v první části popsat moderní směry využití čipových karet. Druhý blok se zaměřoval na fyzické vlastnosti čipů na tzv. chytrých kartách (Smart Card). Jednou z používaných platforem těchto karet je Java Card společnosti Oracle, kterou jsme popsali v další kapitole. Důležitým prvkem je princip paketů APDU. Následující blok textu se už věnuje obecnému návrhu aplikací. V programu JCS Suite vybaveném jádrem na bázi vývojového prostředí Eclipse došlo k analýze a návrhu appletů včetně simulace výměny dat. K dispozici proto bylo kvalitní hardwarové vybavení. Důležité informace pro pochopení zabezpečení platformy Java Card přibližuje pátá kapitola práce. Přes definici firewallu a omezení sdílení dat se dostáváme k přehledu útoků zákeřným kódem a celkové obrany čipových karet. Pro kryptografii i metody autentizace je vyhrazena další část. Na závěr jsme využili znalostí pro shrnutí výhod a nevýhod celé platformy. Zakončili jsme zmínkou o praktickém využití a potenciálu čipových karet do budoucna. Včetně shrnutí vhodnosti karet na platformě Java Card pro využití dvou i více aplikací současně.

Popsání technologie Java Card pro Smart Cards (čipové karty), jejích bezpečnostních podmínek a vhodnosti pro multiplikační využití pro více appletů na jedné kartě, bylo stěžejním bodem zájmu závěrečné práce.

Princip technologie je založen na programovacím jazyku Java. Tento fakt už zajišťuje určitou míru bezpečnosti. Některé schopnosti Java Card dědí nebo přidává, jiné potlačuje, to vše v rámci zjednodušení syntaxe jazyka pro kartu. Programy na tuto platformu jsou objemově menší, mají pevně určenou základní formu (metody process, install, komunikace APDU atd.) a pravidla pro psaní základních jednotek neboli appletů. V průběhu návrhu dvou aplikací na kartu vyplynulo několik důležitých faktů. Systém zde disponuje nástrojem pro oddělení jednotlivých aplikací – firewallem. Ten zabraňuje jakékoliv výměně dat mezi různými applety navzájem. Se všemi applety může komunikovat jen řídící blok JCRE prostředí. Existují pouze pečlivě vybrané prostředky povolující do jisté míry přístup jednoho appletu do jiného. Nejvýznamnější je sdílené rozhraní, speciální blok v těle appletu, kam např. vývojář vloží metody, které může za nějakým účelem cizí applet vidět. K ostatním částem aplikace applety za hranicí firewallu přístup nemají. Existují zde dále ochranné podmínky při náhlém výpadku napájení karty, kdy se prováděné operace a další použité objekty vrací do výchozích stavů díky uložení v paměti. Čtecí zařízení nebo terminál komunikuje s kartou prostřednictvím paketů APDU. Ty mají stanovenou formu obsahující různé druhy instrukcí, které jsou všechny uvedeny v normě ISO 7816. Na straně terminálu bývá v praxi Java aplikace. Komunikuje s karetním systémem formou výzva – odpověď.

Při konkrétním návrhu appletu jsme si potvrdili, že správnou funkci zdrojového kódu zajistí jen správná syntaxe podle pravidel definovaných např. v Java Card Development Kit verze 2.2.x od společnosti Oracle. Tomuto vyhovovala právě testovaná karta Convego

Join 4.0. V rámci appletu fungují pouze metody, které jsou v něm definovány. Pokud by nějaká třetí strana posílala instrukci v paketu APDU, která v těle zdrojového kódu cíleného appletu není zmíněna, dojde k chybovému hlášení. V konkrétním případě našich appletů, PIN je odeslán ve zprávě jen jednou a při schválení již další instrukce posílá terminál příkazy a data s podvědomím, že applet je autorizován. To snižuje možnost odposlechu tajného hesla, ale na druhou stranu zvyšuje úspěšnost zcizení informací díky náhodnému uhodnutí hesla. Z pohledu bezpečnosti je větší hrozbou škodlivý kód, reprezentovaným části kódu nebo celým appletem. Může zde být snaha přečíst datové typy za jejich hranicí reprezentace v paměti a dostat se k datům jiných aplikací. Určitou ochranou je ověřování bajtového kódu před konverzí appletu. A dále, při inicializaci komunikace jsou autorizačními APDU vyměňovány šifrovací klíče, ověřena identita terminálů a karty. Co se týče formy souboru nahrávatelné na čip karty, vytvoří se ze zdrojových kódů CAP pomocí konvertoru zabudovaného ve virtuálním stroji JCVM. Jde podle mne o krok na víc, který ruší Java Card verze 3. Celkově po ověření vlastností Java Card můžeme říci, že systém je vhodný pro použití více aplikací najednou. Je potřeba ovšem přidat do každé konkrétní aplikace vyhovující zabezpečení. A nepustit vědomě třetí stranu k našim programům.

Dnešní forma čipových karet (s Java Card apod.) nedisponuje nějakou formou mechanické ochrany. Každý majitel karty si musí dle svého uvážení chránit čip, který při špatných provozních podmínkách může i přestat kompletně fungovat. Vlastník karty si nebude určitě chtít poškodit svá citlivá data a musí se vyvarovat ztrátě nebo odcizení. Pouze útočník disponující určitými znalostmi může mechanicky, s využitím potřebného vybavení, vysledovat a upravit funkční prostředky čipu na kartě, vymazat paměť apod.

Na bázi nové platformy Java Card 3, se applety z karty přes uzpůsobený terminál už propojí s webovým serverem poskytující danou službu se zabezpečením formou HTTPS a SSH protokolů.

Trendem je jednoznačně integrace a miniaturizace. Dnešních několik druhů karet pro různé využití vystřídá jedna na všechno. To bude klást obrovské nároky na bezpečnost. Můžeme jít v představách ještě dále, kdy plastová forma, jak ji známe, vymizí a nastoupí pouze drobné čipy voperované do tkáně člověka.

LITERATURA

[1] CHEN, Zhigun. *Java Card Technology for Smart Cards : Architecture and Programmer´s Guide*. 2. vydání. Messachusetts : Addison Wesley, 2004. 368 s. ISBN 0201703297.

[2] Doc. Ing. BURDA, Karel CSc. *Bezpečnost informačních systémů*. Brno : VUT FEKT Brno, 2005. 104 s.

[3] *ORACLE : Java Card Technology* [online]. Oracle Corporation, 500 Oracle Parkway, Red Wood Shores : 2010 [cit. 2010-12-05]. Dostupné z WWW: <http://www.oracle.com/technetwork/java/javacard/overview/index.html>.

[4] RANKL, Wolfgang; EFFING, Wolfgang. *Smart Card Handbook*. 3. vydání. Munich : John Wiley & Sons Ltd, 2003. 1043 s. ISBN 0470856688.

[5] RANKL, Wolfgang. Smart Card Applications. West Sussex (England) : John Wiley & Sons, Ltd., 2007. 217 s. ISBN 978-0-470-05882-4

[6] *Smart Card Alliance* [online]. 1997-2010. New Jersey (USA) : 2010 [cit. 2010-12-05]. Dostupné z WWW: <http://www.smartcardalliance.org/>.

[7] *Smart Card Basics* [online].Card Logic Corporation, 2010 [cit. 2010-12-05]. Dostupné z WWW: <http://www.smartcardbasics.com/>.

[8] *Smart Card Forum 2010* [online]. Verze 2010. květen 2010 [cit. 2010-12-05]. Dostupné z WWW: <http://www.smartcardforum.cz/>.

[9] *Giesecke & Devrient GmbH : Technical White PaperJCS Suite v3.0* [online]. 2008. Munich : Giesecke & Devrient, 2008 [cit. 2010-12-14]. JCS Suite v3.0. Dostupné z WWW: <http://www.gi-de.com/pls/portal/maia.display_custom_items. DOWNLOAD_SEEALSO_FILE?p_ID=6932&p_page_id=156036&p_pg_id=42>.

[10] MOSTOWSKI, Wojciech; POLL, Erik. *Radboud University Nijmegen* [online]. 2007 [cit. 2011-05-13]. Malicious Code on Java Card Smartcards : Attacks and Countermeasures. Dostupné z WWW: <http://www.cs.ru.nl/~erikpoll/papers/cardis08.pdf>.

[11] DI GORGIO, Rinaldo. *Javaworld.com* [online]. 01.02.1998 [cit. 2011-05-13]. Get a jumpstart on the Java Card. Dostupné z WWW: <http://www.javaworld.com/jw-02-1998/jw-02-javadev.html?page=1>.

[12] *Augsburg University - Institut fur Informatik* [online]. V1.10. Palo Alto, USA : Sun Microsystems, Inc., 18. 07. 1998 [cit. 2011-05-13]. Java Card Applet Developer's Guide. Dostupné z WWW: <http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/teaching/fruehere_semester/ws0708/javacard/Dokumentation/JCADG.pdf>.

[13] *ORACLE* [online]. California, USA : Sun Microsystems, Inc., 2006 [cit. 2011-05-13]. Overview of the Java Card™ Protection Profile Collection. Dostupné z WWW: <https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=java_card_pp-2.2.2-overview-oth-JPR@CDS-CDS_Developer>.

[14] MARLET, Renaud. *Http://www.siveroni.com/ - Igor A. Siveroni, Ph.D.* [online]. 2002 [cit. 2011-05-13]. Demoney: A Demonstrative Electronic Purse Card Specification. Dostupné z WWW: <http://www.siveroni.com/imperial/secsafe/docs/secsafe-tl-007-0.8.pdf>.

[15] MCGRAW , Gary; FELTEN, Edward. *Http://www.securingjava.com/ : Section 7* [online]. 1999 [cit. 2011-05-13]. Java Card Security: How Smart Cards and Java Mix . Dostupné z WWW: <http://www.securingjava.com/chapter-eight/chapter-eight-7.html>.

[16] CHEN, Zhigun. *Www.javaworld.com* [online]. 1999 [cit. 2011-05-13]. How to write a Java Card applet: A developer's guide. Dostupné z WWW: <http://www.javaworld.com/jw-07-1999/jw-07-javacard.html?page=1>.

[17] *SmartCafé Expert 4.0 : Smart Card Platform*. Edition 05.2008. Munchen : Giesecke & Devrient GmbH, 2008. 195 s.

[18] *Http://www.gi-de.de* [online]. Munich, Germany : Giesecke & Devrient GmbH, 2010 [cit. 2011-05-16]. Convego Platform and Applications. Dostupné z WWW: <http://www.gi-de.de/gd_media/media/en/documents/brochures/mobile_security_2/cspa/Convego-Platforms-and-Applications.pdf>

ZKRATKY

AES - Advanced Encryption Standard / symetrický šifrovací algoritmus standard
AID - identifikátor aplikace
APDU - Application Protocol Data Unit - Paket výměny dat
API - Application Programming Interface / Programovací rozhranní
ATR - answer to Reset - Inicializační informace po připojení karty
BCV – byte code verifier / kontrolér kódu
C-APDU - Command APDU / Příkazový paket
CAD - Card Acceptance Device / Čtecí zařízení pro kartu
CAP - Converted Applet / Zkompilovaný applet
CBC-DES - Cipher Block Chaining DES / Speciální typ šifrovacího standardu
CLA - Class of Isntruction / Skupina isntrukcí
CLK - Clock / Hodinový časovač
CPLC – Card Process Life Cycle / životní cyklus karet
CPU - Central Processing Unit / Centrální řídící jednotka
ČK - čipová karta
DES - Data Encryption Standard / šifrovací standard
DSA - Digital Signature Algorithm / Digitální podpis
EEPROM - Electrical Erasable Programable Read-only-memory / programovatelná paměť
EMV - standard Europay Mastercard Visa
EPO – vstupní přístupové body JCRE
G&D – Giesecke & Devrient německá společnost
GND - Ground / Zem
GSM - Globalní systém pro mobilní komunikace
HTTPS - Hypertext Transfer Protocol Secure / Zabezpečený protokol
I/O - Input/Output / Vstup a výstup
ID – identifikátor
IDE – Integrated Development Environment / vývojové prostředí
INS - Instruction Mode / Typ instrukcí
IO - integrované obvody
ISD – Issuer Security Domain / bezpečnostní doména karty
ISO - International Organization in Standardization / Mezinárodní organizace pro standardy
JCRE - Java Card Runtime Environmnent / Vývojové prostředí Java Card
JCS - Java Smart Caffé / Software pro správu karet
JCT - Java Card Technology / Technologie Java Card
JCVM - Java Card Virtual Machine / JC Virtuální stroj
Lc, Le - délka datového pole paketu, počet bytů
MAC – kontrolní číslo dat paketů
OS - operační systém
PC - počítačová stanice
PIN - Personal Identification Number / Soukromé identifikační číslo
PIX - Pokračování RID (seriové číslo atd.)
PPS - Protocol Parameter Selection / Výběr protokolu
R-APDU - Response APDU / Paket odpovědi
RAM - Random Acces Memory / Paměť náhodného přístupu
RFU - Radio Frequency Unit - Jendotka pro radiovou komunikaci
RID - Registrovaný unikátní identifikátor
ROM - Read Only Memory / Paměť pro čtení
RSA - RIver-Shamir-Adleman algoritmus
RST – restart
SC - Smart Card
SHA - Secure Hash Algorithm / Hash funkce pro šifrování
SIM - Subscriber Identity Module / Modul identity uživatele

SIO – Shareable Interfeace Object / objekt sdíleného rozhraní
SSL – Secure Socket Layer
SWx - stavová slova v odpovědi
TCP/IP - Transmission Control Prot. / Internet Prot. / Protokolová sada
TLV – způsob kódování dat Type-Length-Value
TPDU - Transmission Proovol Data Unit / Paket transportního protokolu
USB - Universal Serial Bus / rozhraní
Vcc - přívod napětí
VISA - typ platební karty
Vpp - programovací signál
XML – Extensible Markbale language / značkovací jazyk
XOR - Exclusive OR / operátor

PŘÍLOHA A
Obsah CD

- návod.txt
- složky:
- Applety – zdrojové kódy dvou appletů
- Diplomová práce – dokument v pdf
- Obrázky – různé exportované obrázky prostředí JCS Suite
- XML výpisy APDU – důležité výpisy komunikace pomocí APDU

PŘÍLOHA B

Příklad APDU autentizace karty v XML kódu


```
<Info>Connect, SCM Microsystems Inc. SDI010 Smart Card Reader 0</Info> <CommandApdu>80 CA
9F 7F 00</CommandApdu>

 <ResponseApdu Time="31">9F 7F 2A 40 90 61 62 16 71 81 97 05 80 70 79 4F 07 19 09 29 4A 81
02 13 52 16 73 27 00 16 74 93 64 00 00 00 52 00 00 00 00 00 00 00 00 90 00</ResponseApdu>

<CommandApdu>00 A4 04 00 00</CommandApdu>

<ResponseApdu Time="16">6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF 90
00</ResponseApdu> </Task>

-<Task Time="63"> <Info>Select Applet,
com.gieseckedevrient.javacard.os.GDSystem.CardManager</Info>

<Result>FCI: 6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF</Result>

 <CommandApdu>00 A4 04 00 08 A0 00 00 00 03 00 00 00 00</CommandApdu>

 <ResponseApdu Time="47">6F 10 84 08 A0 00 00 00 03 00 00 00 A5 04 9F 65 01 FF 90
00</ResponseApdu> </Task>

 -<Task Time="187"> <Info>Authenticate</Info>

 <CommandApdu>80 50 00 00 08 FC B6 2D A5 27 79 3C 4D 00</CommandApdu>

<ResponseApdu Time="94">00 00 70 79 4F 07 19 09 29 4A 01 02 00 92 13 DA 44 7B 54 96 09 96
E0 97 A8 92 AC D2 90 00</ResponseApdu>

<CommandApdu>84 82 01 00 10 60 4F F6 6E 01 65 F7 49 1F 27 A0 BC F6 49 73 06</CommandApdu>

 <ResponseApdu Time="62">90 00</ResponseApdu> </Task>

 -<Task Time="109"> <Info>Get Status, Subset: IssuerSecurityDomain</Info>

<CommandApdu>84 F2 80 00 0A 4F 00 D8 AC FC 79 CC B8 C6 18 00</CommandApdu>

<ResponseApdu Time="63">08 A0 00 00 00 03 00 00 00 07 9E 90 00</ResponseApdu> </Task>

 -<Task Time="156"> <Info>Get Status, Subset: AppletsSecurityDomains</Info>

<CommandApdu>84 F2 40 00 0A 4F 00 BC 28 D4 BD F1 26 FB BD 00</CommandApdu>

<ResponseApdu Time="78">07 D2 76 00 00 60 41 03 07 00 07 D2 76 00 00 60 41 01 07 00 90
00</ResponseApdu> </Task>

 -<Task Time="156"> <Info>Get Status, Subset: PackagesAppletClasses</Info>

<CommandApdu>84 F2 20 00 0A 4F 00 1B AE 9D E2 D0 10 0C C1 00</CommandApdu>

<ResponseApdu Time="78">07 A0 00 00 00 03 53 44 01 00 07 A0 00 00 00 03 53 50 01 00 08 A0
00 00 02 27 01 10 00 01 00 0C D2 76 00 00 05 AA 04 03 60 01 04 10 01 00 07 D2 76 00 00 60
50 03 01 00 07 D2 76 00 00 60 50 01 01 00 08 AB CD EF FE DC CA FF EE 01 00 90
00</ResponseApdu> </Task>

 -<Task Time="140"> <Info>Get Data, Tag: 66</Info>

 <Result>Data: 73 4A 06 07 2A 86 48 86 FC 6B 01 60 0C 06 0A 2A 86 48 86 FC 6B 02 02 01 01
63 09 06 07 2A 86 48 86 FC 6B 03 64 0B 06 09 2A 86 48 86 FC 6B 04 02 15 65 0B 06 09 2B 85
10 86 48 64 02 01 03 66 0C 06 0A 2B 06 01 04 01 2A 02 6E 01 02</Result>

 <CommandApdu>84 CA 00 66 08 80 EF 3A 9C CF A4 2B A2 00</CommandApdu>

<ResponseApdu Time="62">66 4C 73 4A 06 07 2A 86 48 86 FC 6B 01 60 0C 06 0A 2A 86 48 86 FC
6B 02 02 01 01 63 09 06 07 2A 86 48 86 FC 6B 03 64 0B 06 09 2A 86 48 86 FC 6B 04 02 15 65
0B 06 09 2B 85 10 86 48 64 02 01 03 66 0C 06 0A 2B 06 01 04 01 2A 02 6E 01 02 90
00</ResponseApdu> </Task>
```

```
-<Task Time="140"> <Info>Get Free Space, EEPROM</Info>

<Result>Free space: 74573 bytes</Result>

 <CommandApdu>84 CA 00 C6 08 A8 D4 70 6E D0 4A 47 7E 00</CommandApdu>

<ResponseApdu Time="63">C6 04 00 01 23 4D 90 00</ResponseApdu> </Task>

-<Task Time="140"> <Info>Get Free Space, transient COD</Info>

<Result>Free space: 3022 bytes</Result>

 <CommandApdu>84 CA 00 C5 08 73 BB 42 52 5B 50 D9 6C 00</CommandApdu>

<ResponseApdu Time="62">C5 04 00 00 0B CE 90 00</ResponseApdu> </Task>

-<Task Time="140"> <Info>Get Free Space, transient COR</Info>

<Result>Free space: 2726 bytes</Result> <CommandApdu>84 CA 00 C7 08 C0 1A D7 3C DE 7F 38 D3
00</CommandApdu>

<ResponseApdu Time="62">C7 04 00 00 0A A6 90 00</ResponseApdu> </Task>

 -<Task Time="156"> <Info>Get Data, Tag: 42</Info> <Error>GET DATA response check failed
with SW 6A 88</Error>

 <CommandApdu>84 CA 00 42 08 A2 5E 04 AE C8 A8 27 BE 00</CommandApdu>

<ResponseApdu Time="78">6A 88</ResponseApdu> </Task>

<CommandApdu>84 CA 00 45 08 B5 BF A0 5F 5E DD BD D7 00</CommandApdu>

<ResponseApdu Time="63">6A 88</ResponseApdu> </Task>

-<Task Time="125"> <Info>Get Key Information</Info>

<Result>Key Information: C0 04 01 01 80 10 C0 04 02 01 80 10 C0 04 03 01 80 10</Result>

<CommandApdu>84 CA 00 E0 08 32 84 23 4A D7 44 86 3B 00</CommandApdu>

<ResponseApdu Time="62">E0 12 C0 04 01 01 80 10 C0 04 02 01 80 10 C0 04 03 01 80 10 90
00</ResponseApdu> </Task>

 -<Task Time="109"> <Info>Get Data, Tag: CF</Info>

<Result>Data: 00 00 70 79 4F 07 19 09 29 4A</Result>

 <CommandApdu>84 CA 00 CF 08 86 8F EE DD 59 B3 59 83 00</CommandApdu>

<ResponseApdu Time="47">CF 0A 00 00 70 79 4F 07 19 09 29 4A 90 00</ResponseApdu> </Task>

-<Task Time="125"> <Info>Get Data, Tag: 9F 7F</Info>

 <Result>Data: 40 90 61 62 16 71 81 97 05 80 70 79 4F 07 19 09 29 4A 81 02 13 52 16 73 27
00 16 74 93 64 00 00 00 52 00 00 00 00 00 00 00 00</Result>

<CommandApdu>84 CA 9F 7F 08 2F 8D 1D 49 62 D0 53 7A 00</CommandApdu>

<ResponseApdu Time="62">9F 7F 2A 40 90 61 62 16 71 81 97 05 80 70 79 4F 07 19 09 29 4A 81
02 13 52 16 73 27 00 16 74 93 64 00 00 00 52 00 00 00 00 00 00 00 00 90 00</ResponseApdu>
</Task>

 </TaskLog>
```